TOC

About This Guide

This is an archived copy of the v23 documentation provided for Logi Info v23.3 and its service packs.

Notice: Archived Documentation

This documentation is provided as a courtesy reference for a version of our software that is no longer under active development or support. The information contained herein is offered without warranties of any kind, either expressed or implied, including but not limited to warranties of accuracy, completeness, or fitness for a particular purpose.

While this archived material may assist with understanding historical functionality, please be aware that the software described is no longer maintained at this version level and may contain outdated or inaccurate information. Images may not reflect currently supported modules, support sites, or third party products. This software may not be compatible with current versions of previously compatible third party products.

To access and upgrade to current software solutions and receive ongoing support, please contact our customer support team. They can assist you in migrating to the latest appropriate software version that meets your needs. Our support team is available to help ensure a smooth transition to actively maintained alternatives that provide the functionality and reliability you require.

Dashboard for End Users

Dashboards present collections of visualizations so that you can quickly see analyses of your data. Depending on how it's been configured, you may be able to customize the Dashboard to suit your needs.

The following topics discuss Logi Dashboards:

- Working with Dashboard Tabs
- Managing Dashboard Panels
- Panel Re-Arrangement and Sizing
- Filtering Dashboard Data
- Drill to Dashboard Data
- Editing Visualizations
- Exporting Panel Content

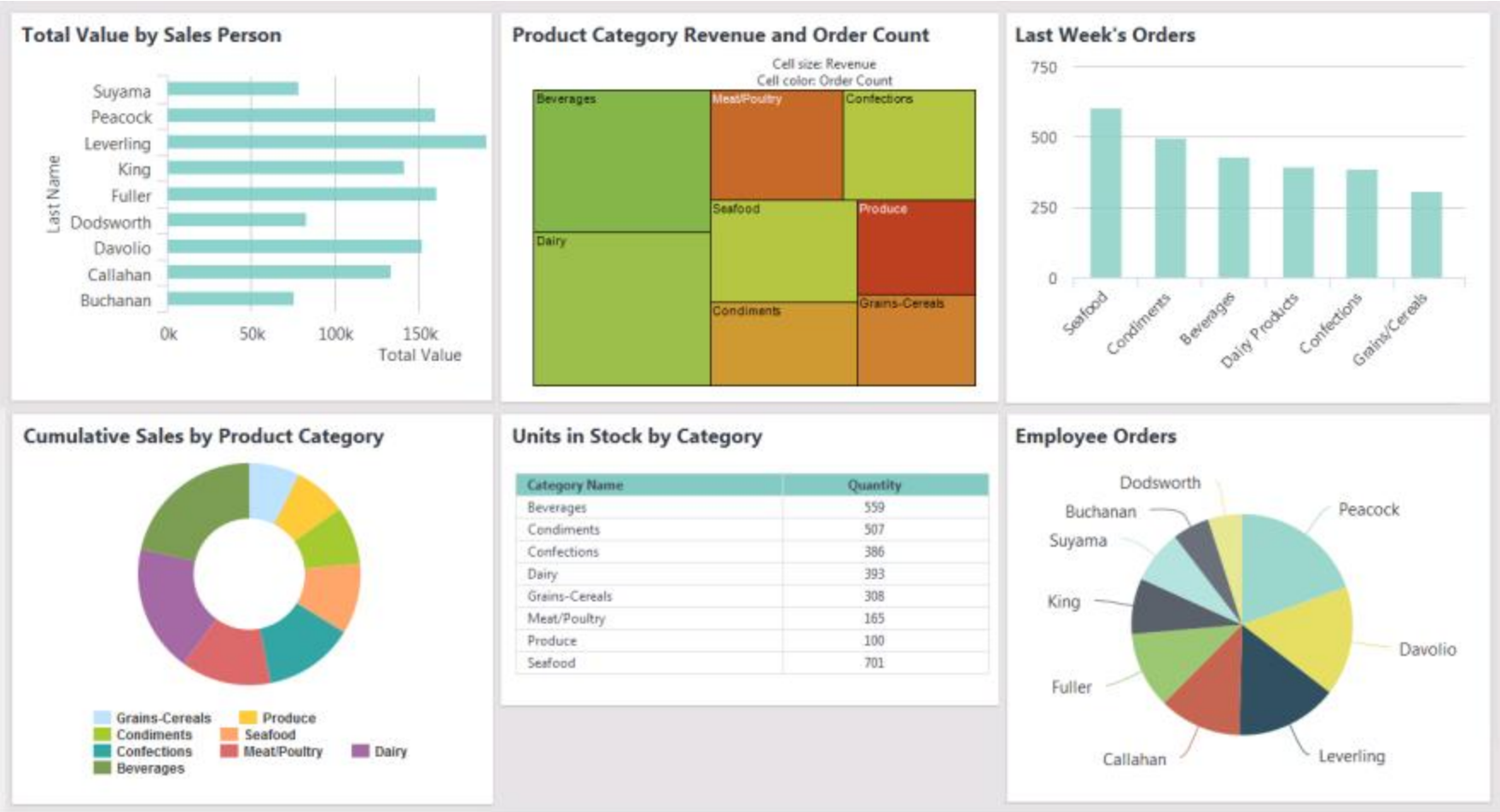
Information for *developers* is available in *Logi Info Dashboard*.

About the Dashboard

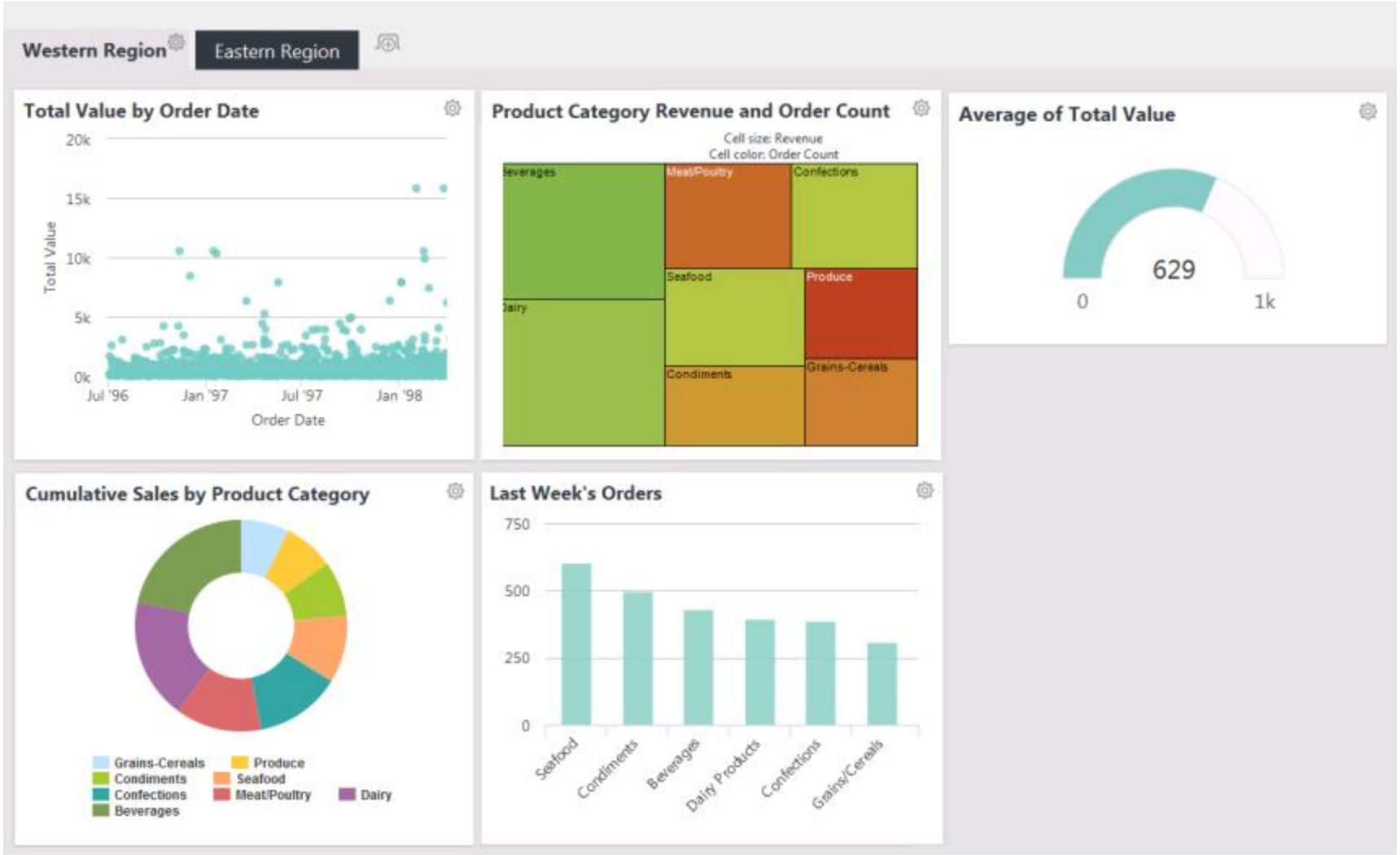
A Logi "Dashboard" is a collection of *Dashboard panels*, each containing a data visualization (table, chart, gauge, etc.). Depending on how it's been configured, you may be able to customize it by adding, removing, or rearranging these panels on the browser page. Your Dashboard may also be configured to have tabbed pages, so that different collections of Dashboard panels can be viewed.

The visualizations in Dashboard panels can be interactive, so you may have the option of varying the reporting criteria for them. Links within the reports can be used to drill-down and drill-through to detail data.

All in all, Dashboards present a very flexible way of presenting a range of information that's easy to take in with a single glance.



A simple Dashboard, with no user customization features, styled using the standard *Signal* theme, is shown above.



In the example above, we see a more complex Dashboard which includes tabs and has end-user customization enabled.

The data used in Dashboard panel visualizations is based on "metadata". If the metadata changes, such as the removal of a column, the panel will display a warning about the missing column, instead of the expected visualization.

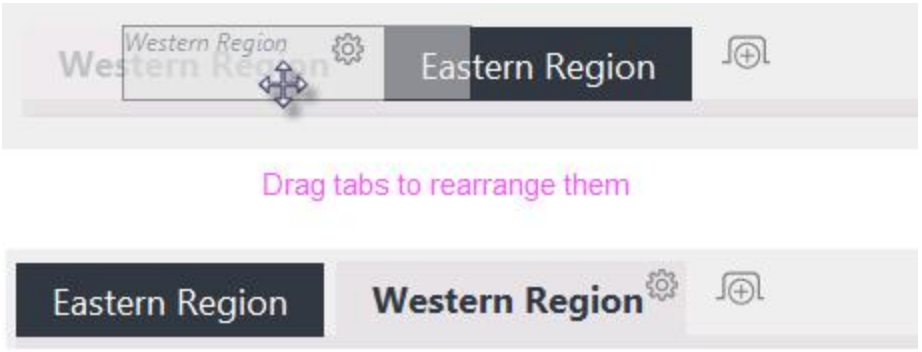
Now, let's get started using the Dashboard.

Working with Dashboard Tabs

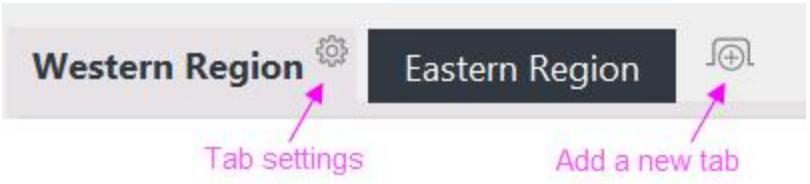
A Dashboard can consist of a single collection of panels (a base page) or a series of tabbed pages, each with its own collection of panels. Dashboards can be configured by your developer to be *static* or *adjustable.*

If a Dashboard has been configured to be adjustable *and* to have tabs, then you can add, remove, and rearrange the tabs.

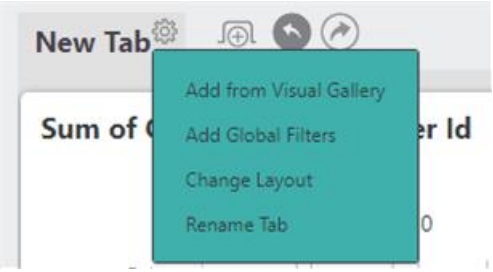
A Dashboard with tabs will have one tab created by default. In the example shown below, an additional Dashboard tab has been added. Tabs can be dragged with the mouse to rearrange them:



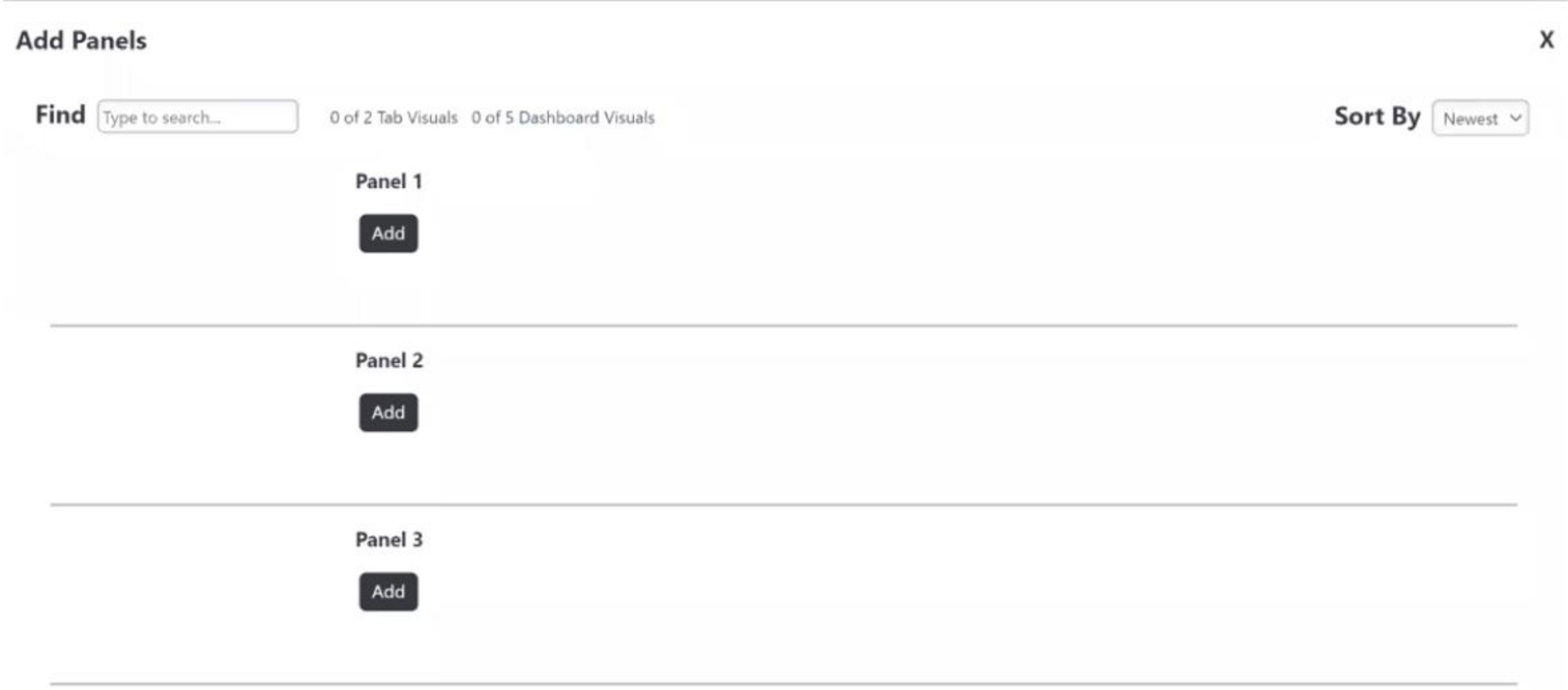
Adjustable Dashboards display this "gear" icon on the selected tab; click the **icon** to configure their settings. To add a new tab to the Dashboard, click the **plus sign** icon, as indicated below.



A tab's Settings menu, shown below, appears when its gear icon is clicked and allows you to add panels to the tab, change the number of columns being used in the tab, to rename the tab, and to remove the tab.



Dashboards can also be configured by your developer to limit the number of panels and tabs. The Add Panels pop-up panel indicates whether or not your Dashboard is configured to restrict the number of panels and tabs:



To add a new Dashboard or Tab Visual, you must first remove existing panels/tabs that are exceeding the count, based on the limitation.

Managing Dashboard Panels

Dashboard panels are the basic building blocks of the Dashboard:

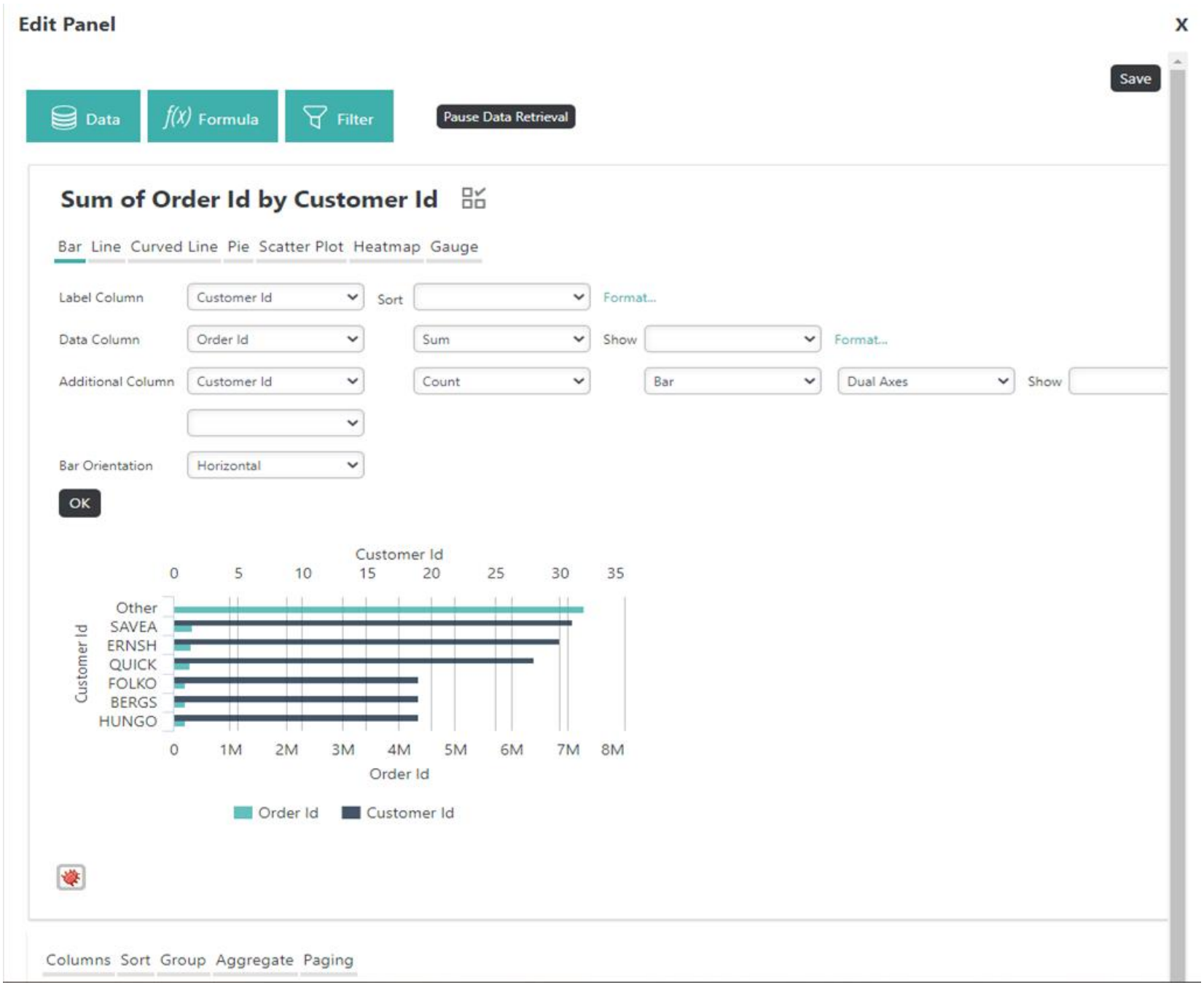
As shown above, Dashboard panels have a Caption (1), content (2), and settings (3).

If a panel is adjustable, it displays a gear icon like this ⚙ in its upper right-hand corner; click it to configure the panel's settings.

Adjustable panels can be renamed (changing their captions), rearranged on in the Dashboard or tab, or removed completely. If a panel has been created with parameters, you can adjust the parameter values or filter the data, affecting the panel content.

To make changes to the visualization in the panel, select **Edit**.

An Edit Panel pop-up displays, shown below:


Edit Panel
X
Save
Data
Formula
Filter
Pause Data Retrieval
Sum of Order Id by Customer Id
Bar Line Curved Line Pie Scatter Plot Heatmap Gauge
Label Column
Customer Id
Sort
Format...
Data Column
Order Id
Sum
Show
Format...
Additional Column
Customer Id
Count
Bar
Dual Axes
Show
Bar Orientation
Horizontal
OK
Customer Id
0 5 10 15 20 25 30 35
Other
SAVEA
ERNSH
QUICK
FOLKO
BERGS
HUNGO
Customer Id
0 1M 2M 3M 4M 5M 6M 7M 8M
Order Id
Order Id
Customer Id
Columns Sort Group Aggregate Paging

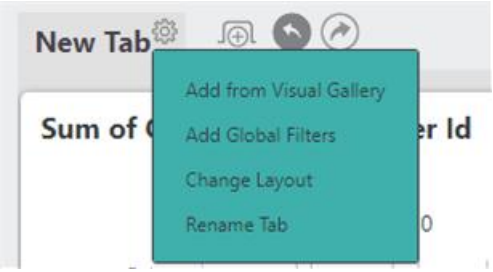
v23.1 If your Dashboard has been configured for it, the Dashboard Panel Title will update automatically according to edits that affect the Visualization Title. For example, if you change the Data Column of the visualization to calculate the Average, the Dashboard Panel Title automatically updates to "Average of.." (instead of the existing "Sum of..", as shown above).

Don't forget to select **Save** after you make your changes.

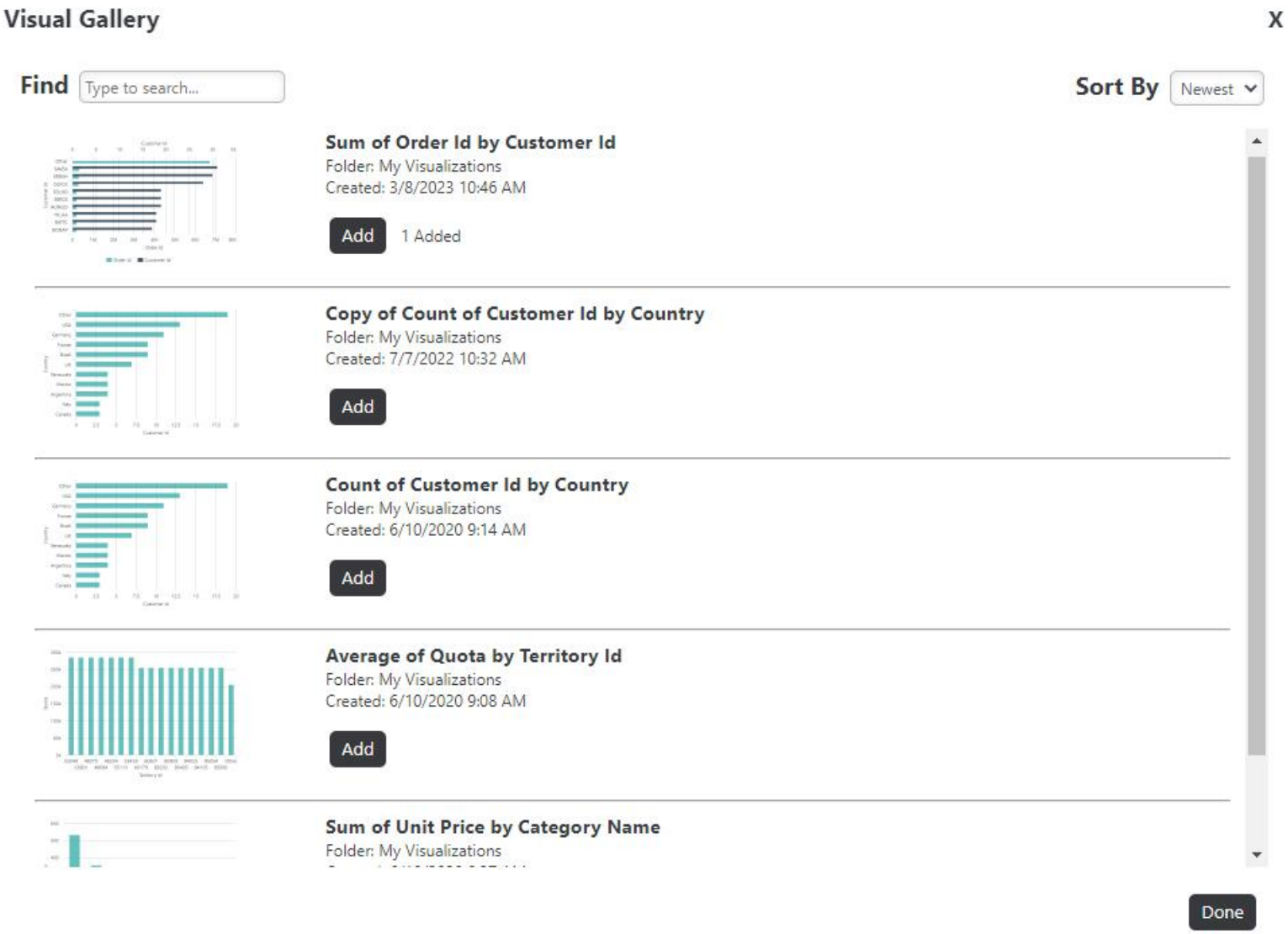
Panel Tabs

Your Dashboard may have been configured with a default set of tabs and panels, so you're not looking at a blank canvas when you browse to it the first time.

Adjustable Dashboards display a gear icon like this ⚙ on their base page or tabs. Click this icon to configure their settings and to *add* panels:



As shown in the example above, when the gear icon is clicked, a Settings pop-up menu appears. Its menu items vary depending on the Dashboard configuration, but "Add from Visual Gallery" is a common item. Click this item to display the Visual Gallery pop-up panel:

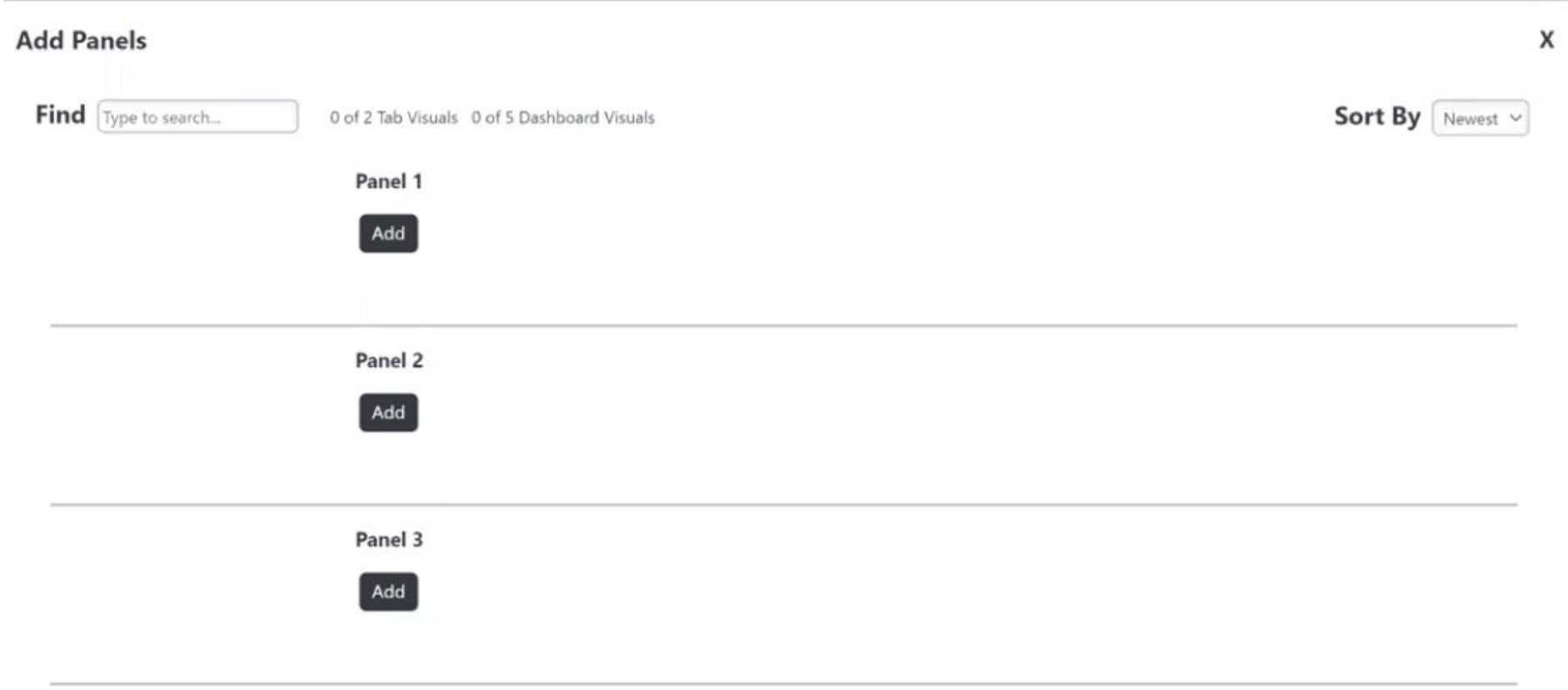

Visual Gallery
X
Find
Type to search...
Sort By
Newest
Sum of Order Id by Customer Id
Folder: My Visualizations
Created: 3/8/2023 10:46 AM
Add
1 Added
Copy of Count of Customer Id by Country
Folder: My Visualizations
Created: 7/7/2022 10:32 AM
Add
Count of Customer Id by Country
Folder: My Visualizations
Created: 6/10/2020 9:14 AM
Add
Average of Quota by Territory Id
Folder: My Visualizations
Created: 6/10/2020 9:08 AM
Add
Sum of Unit Price by Category Name
Folder: My Visualizations
Done

The Visual Gallery pop-up panel, shown above, displays the collection of panels that are available for your use in the Dashboard. If the Dashboard has never been configured before, this panel is the first thing that appears when you view the Dashboard.

The list of available Dashboard panels may include a thumbnail image, a title, a creation timestamp, and a description. Click the **Add** button to add the panel to the Dashboard or tab. After you add a panel, its Add button is replaced with an "Added" label.

Dashboard panels can be configured by your developer for *single* or *multiple* instances, determining whether they can used multiple times or not. When configured for multiple instances, the same panel can appear more than once in a Dashboard or tab. If the Dashboard has been configured to allow multiple instances, then after a panel has been added, its Add button remains available and a count of the number of instances already added is shown.

Dashboard panels can also be configured by your developer to limit the number of panels. The Add Panels pop-up panel indicates whether or not your Dashboard is configured to restrict the number of panels:

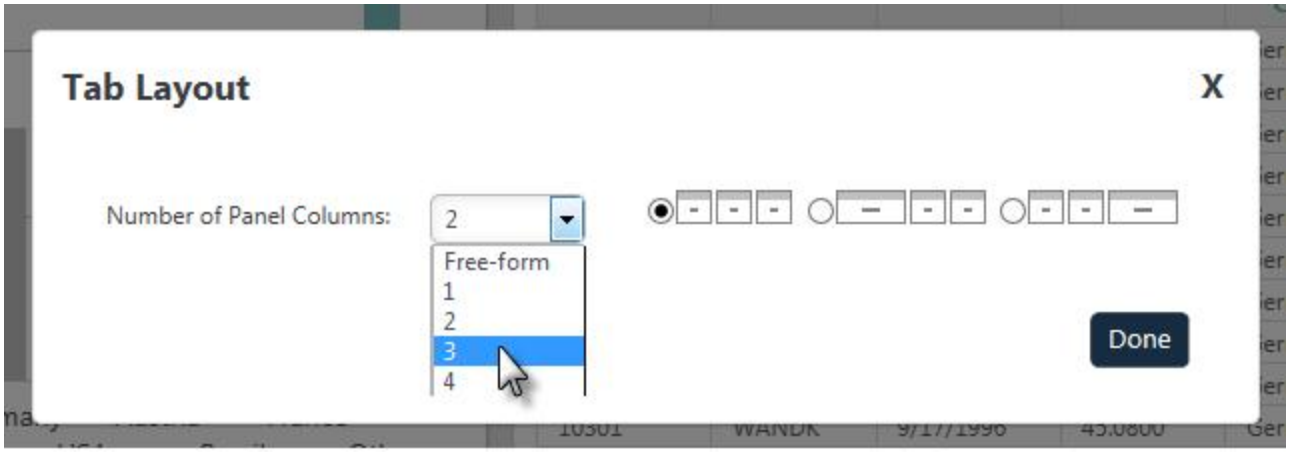


Once you reach the maximum panel allowance, the 'Add' button will be grayed out, prohibiting you from adding more panels to the Dashboard:

To add a new Dashboard or Tab Visual, you must first remove existing panels/tabs that are exceeding the count, based on the limitation.

Panel Re-Arrangement and Sizing

A Dashboard or tab's layout usually consists of one or more *columns* and, if the Dashboard is configured to be adjustable, you can drag-and-drop the panels into the columns and the arrangement of your choice.



If the Dashboard is adjustable, then by using the Dashboard or tab's gear icon and Settings menu, you can change the number of columns in a tab or Dashboard, as shown above.

Adjustable layouts let you use the "splitter bar", described below, to adjust column widths. In this case, when the layout dialog box shown above appears, you're only asked to specify the *number of panel columns* you want.

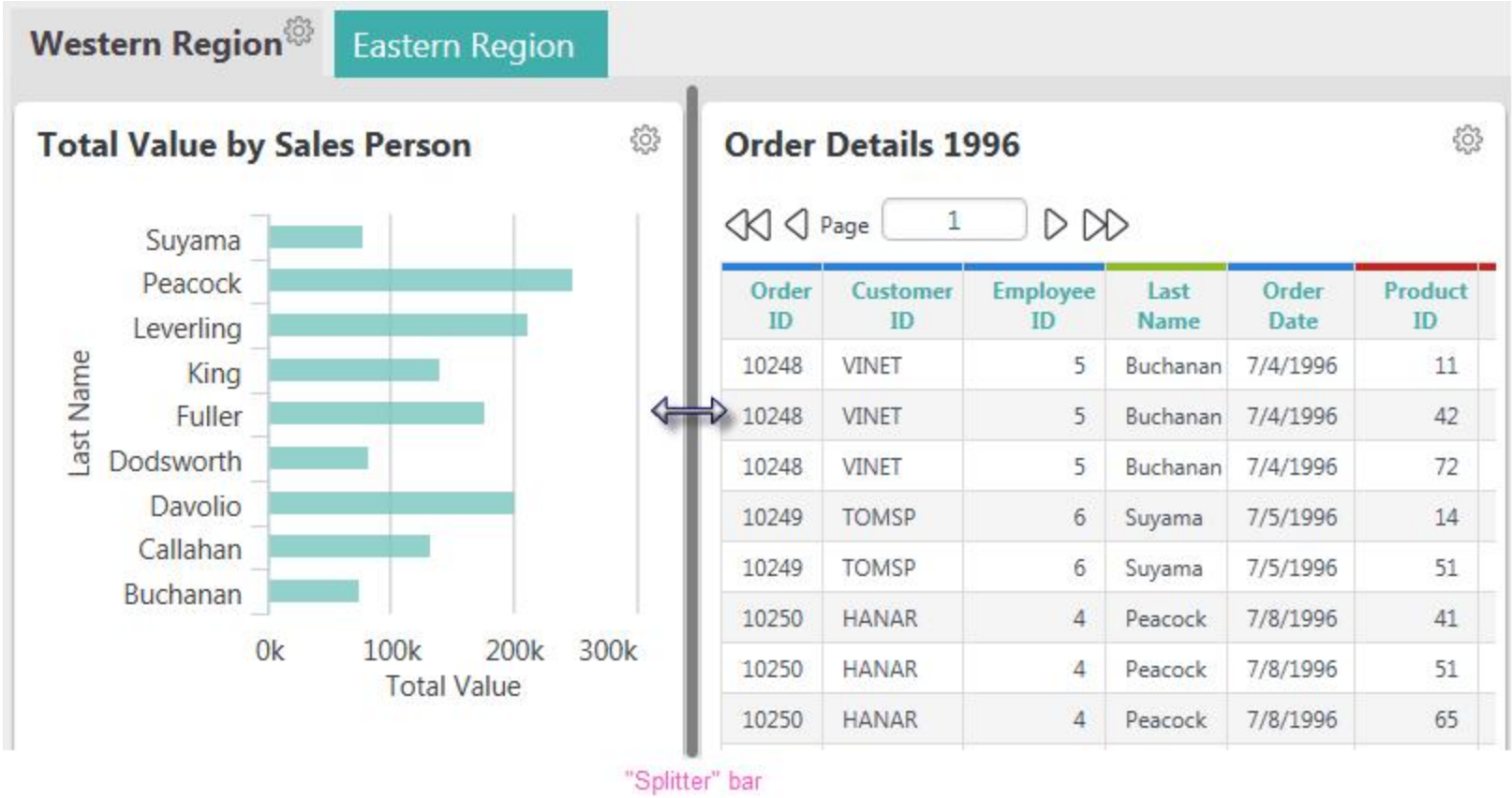
You can drag a panel to a new location by clicking on its caption bar, as shown above (note the cursor change). As the panel is dragged into a different column or position, a yellow "drop zone" indicator appears to help snap the panel into its new location.

In a columnar layout, panels will automatically resize based on the width of the column they're dropped into, and the column's width is determined by the *widest panel* it contains. If the contents of a panel are widened, then the panel and its column will widen, too.

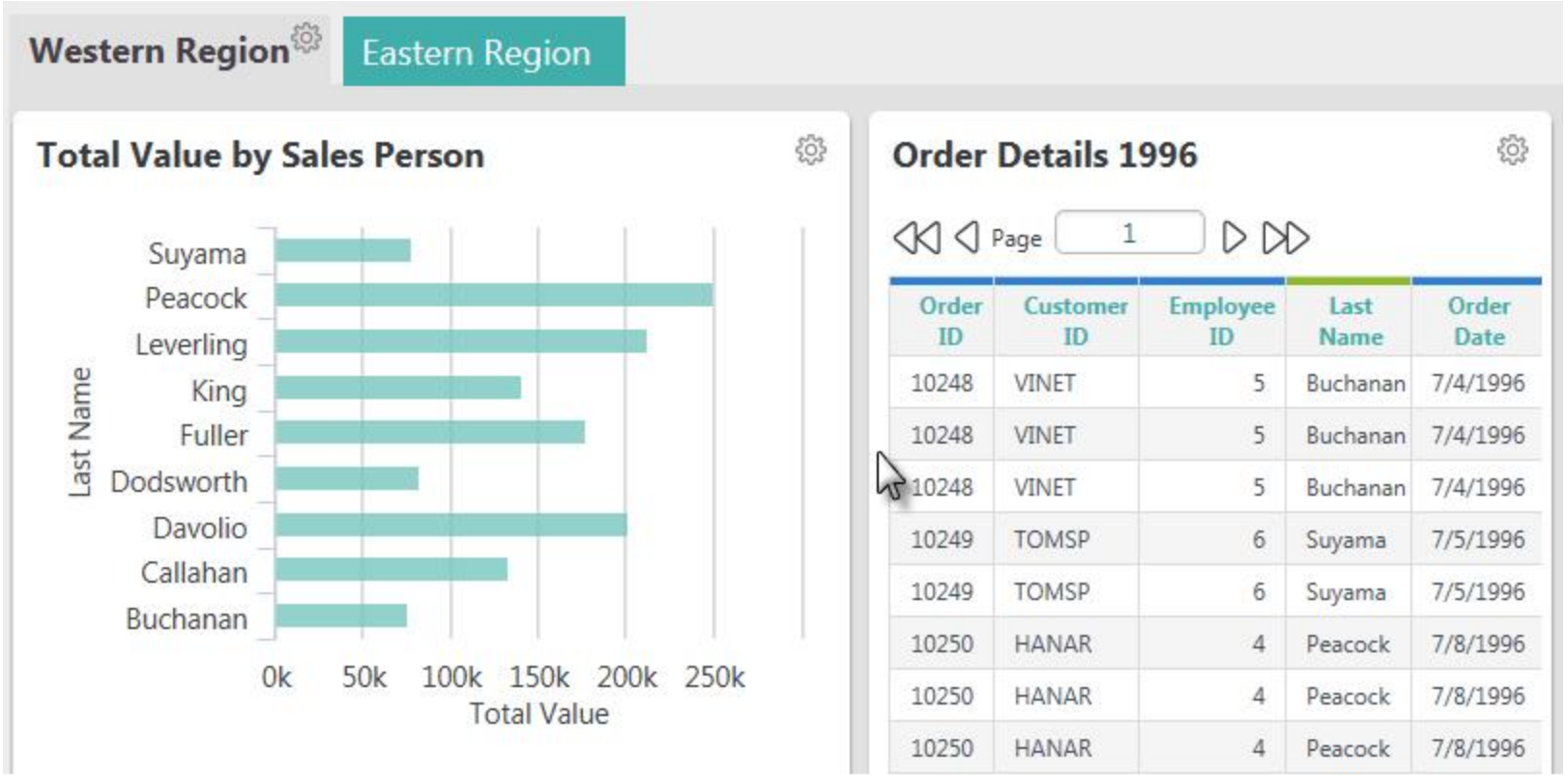
This is a change from panel behavior in previous versions! In a columnar layout, panels will automatically resize based on the width of the column they're dropped into. Column width is a *fixed width*, determined by the layout chosen and screen width. Panel content will be automatically scaled up to the maximum width that fits in the panel. In support of this change, charts in panels in columnar layouts no longer have horizontal resizing handles.

Using the Splitter Bar

You can change the widths of layout columns (and therefore panels) in Dashboards that are configured to be adjustable by using the "splitter" bar:



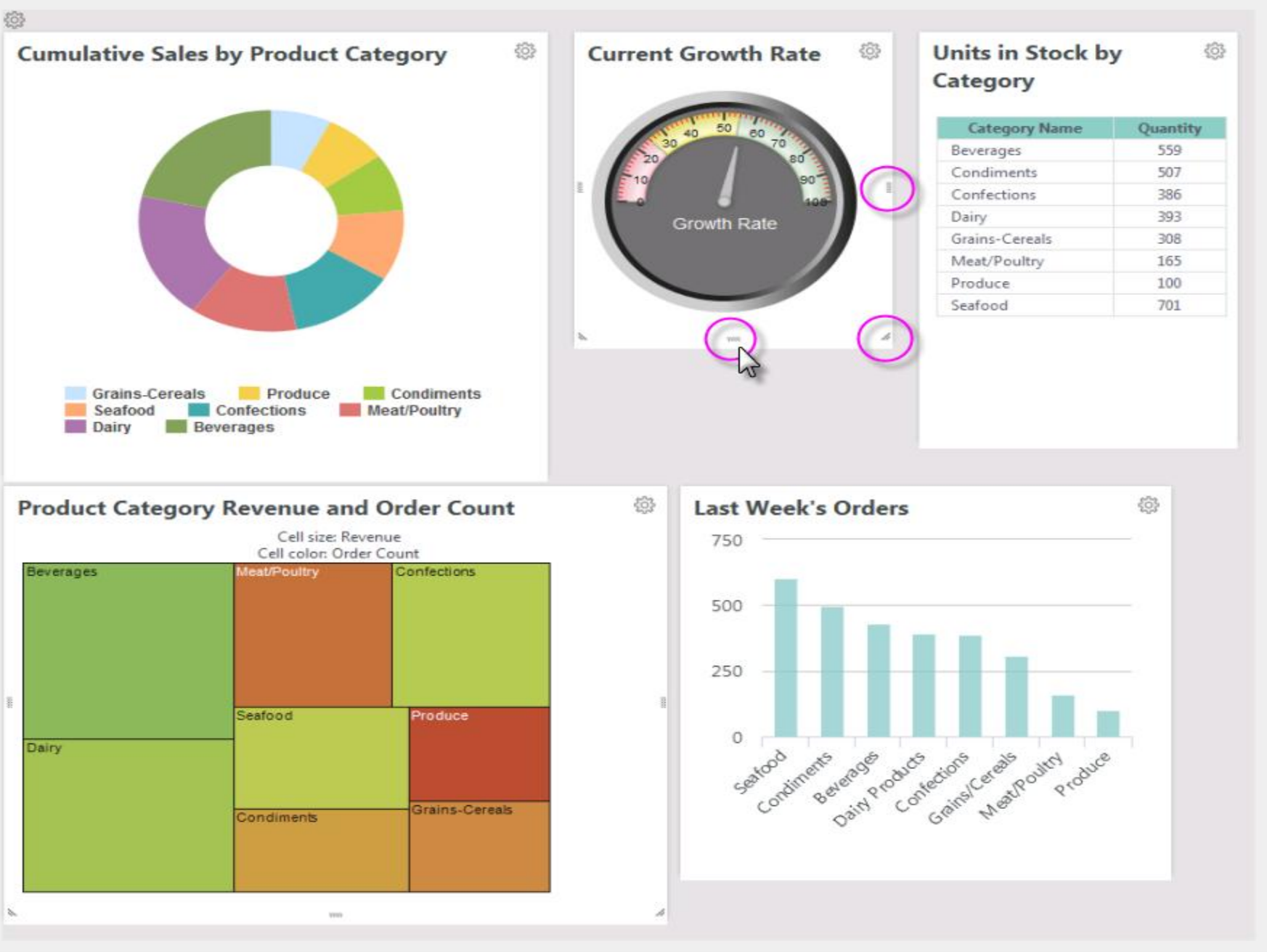
Hover your mouse cursor over the vertical space between two Dashboard panels, as shown above, and a gray splitter bar will appear and the cursor will change, as shown above.



Drag the bar right or left to the adjust panel widths, then drop the bar at the desired location. Visualization widths will automatically adjust to match the change in panel widths.

Using Free-form Layout

Depending on your application's configuration, a "Free-form Layout" option may also available. If it's enabled, you can resize and rearrange Dashboard panels without regard to any predetermined columns:

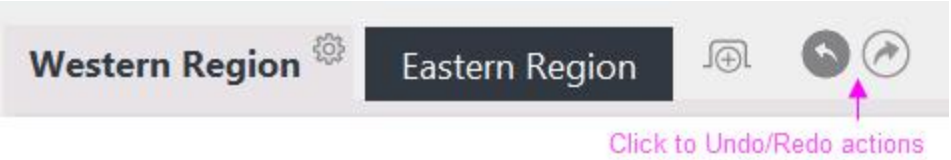

Cumulative Sales by Product Category
Grains-Cereals
Produce
Condiments
Seafood
Confections
Meat/Poultry
Dairy
Beverages
Current Growth Rate
Growth Rate
Units in Stock by Category
Category Name
Quantity
Beverages 559
Condiments 507
Confections 386
Dairy 393
Grains-Cereals 308
Meat/Poultry 165
Produce 100
Seafood 701
Product Category Revenue and Order Count
Cell size: Revenue
Cell color: Order Count
Beverages
Meat/Poultry
Confections
Seafood
Produce
Dairy
Condiments
Grains-Cereals
Last Week's Orders
750
500
250
0
Seafood
Condiments
Beverages
Dairy Products
Confections
Grains/Cereals
Meat/Poultry
Produce

In Free-form layout, shown above, any widths and placements can be achieved. When the mouse cursor hovers over a panel, "resizing handles" appear at its sides and bottom and these can be dragged to resize the panel. Internal panel scroll bars will automatically appear if the panel is made too small for its contents.

Panel side resizing handles no longer appear; use the splitter bar, described earlier, to change panel width instead.

Given that you may be able to customize a Dashboard, it's useful to be able to save those customizations and re-use them in a future session. Your developer may have enabled an automatic "bookmark" mechanism for this, or there may be a link you have to click to save your settings.

Bookmark Undo and Redo



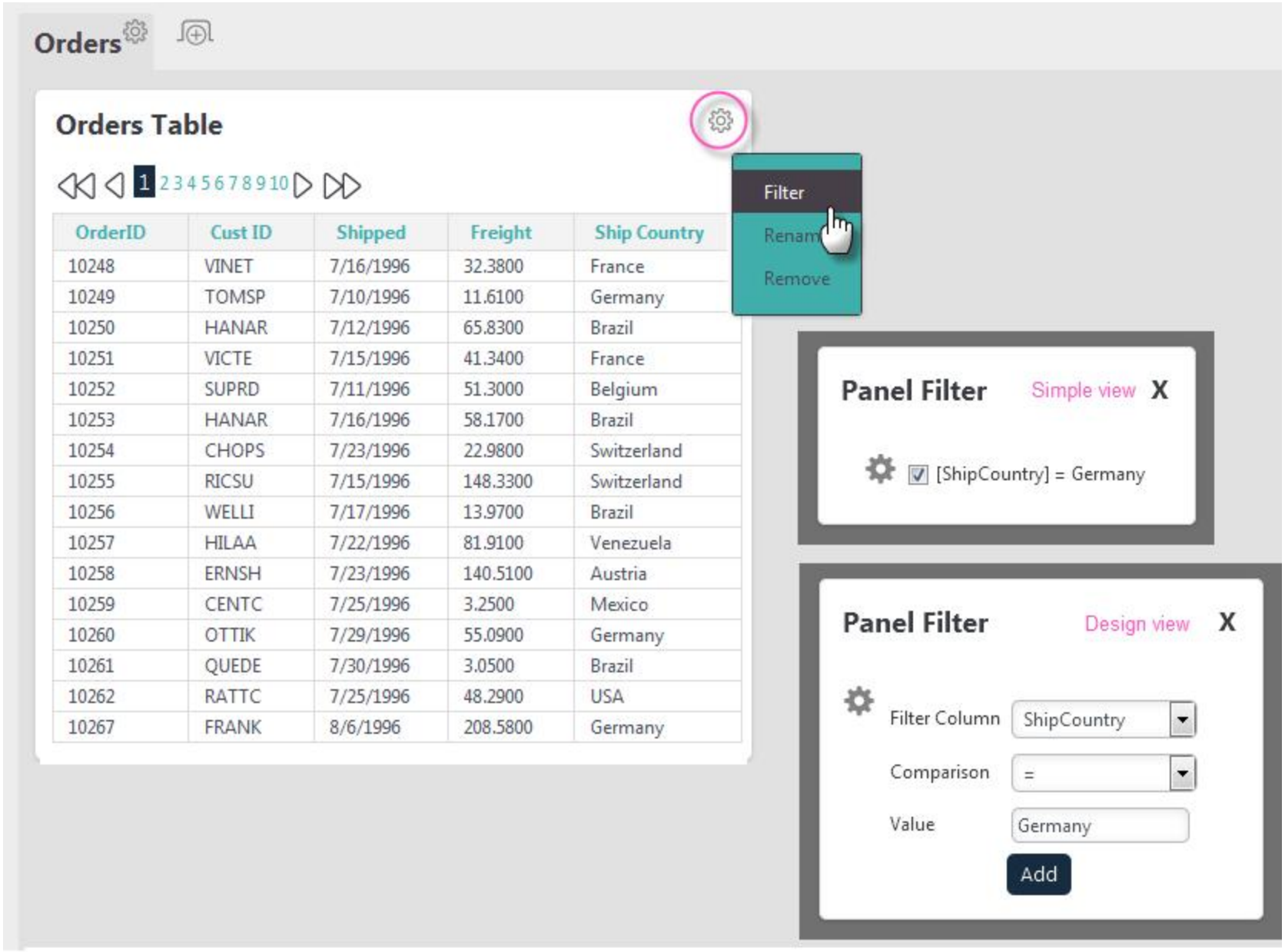
If the automatic bookmark mechanism is in use, **Undo** and **Redo** icons will appear beside the Dashboard tabs, as shown above. You can click these to undo and redo actions you've taken.

Filtering Dashboard Data

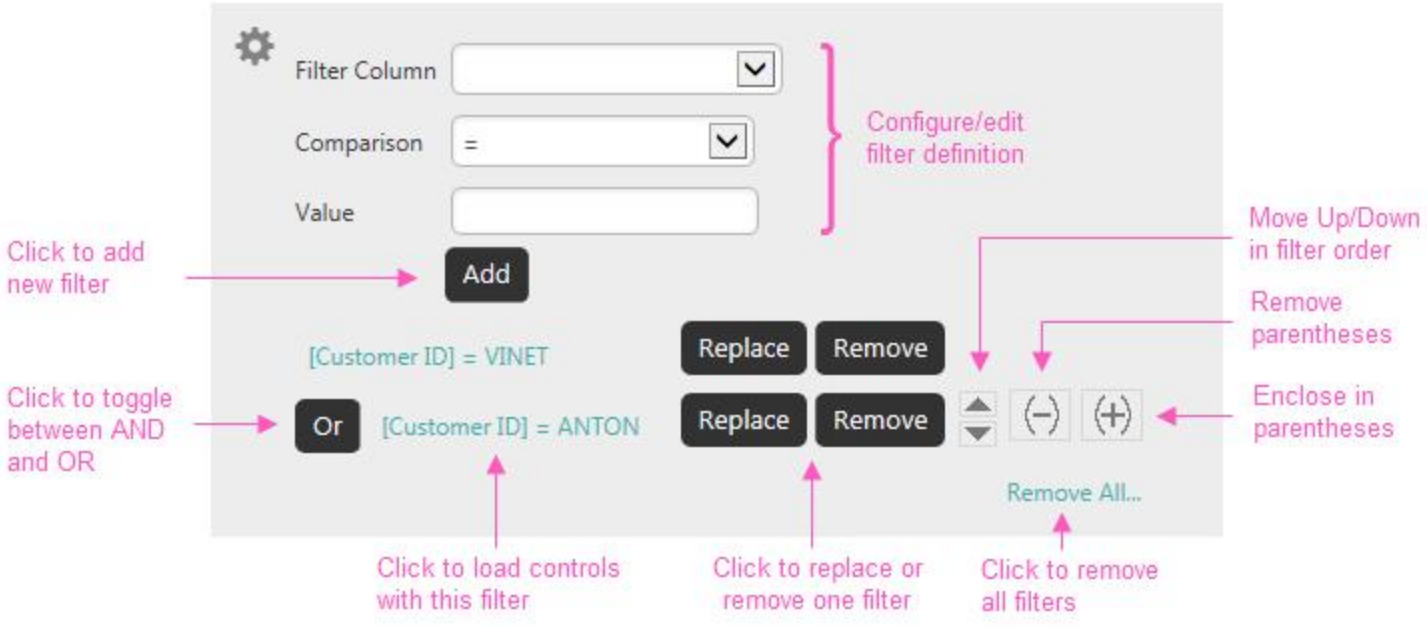
Dashboards can be configured with special features that allow you to *filter* the data used in Dashboard panels, individually and globally.

Filtering Individual Panels

If filtering has been enabled, clicking the panel's "gear" icon will display a pop-up menu that has a *Filter* option.


Orders
Orders Table
1 2 3 4 5 6 7 8 9 10
OrderID | Cust ID | Shipped | Freight | Ship Country
10248 | VINET | 7/16/1996 | 32.3800 | France
10249 | TOMSP | 7/10/1996 | 11.6100 | Germany
10250 | HANAR | 7/12/1996 | 65.8300 | Brazil
10251 | VICTE | 7/15/1996 | 41.3400 | France
10252 | SUPRD | 7/11/1996 | 51.3000 | Belgium
10253 | HANAR | 7/16/1996 | 58.1700 | Brazil
10254 | CHOPS | 7/23/1996 | 22.9800 | Switzerland
10255 | RICSU | 7/15/1996 | 148.3300 | Switzerland
10256 | WELLI | 7/17/1996 | 13.9700 | Brazil
10257 | HILAA | 7/22/1996 | 81.9100 | Venezuela
10258 | ERNSH | 7/23/1996 | 140.5100 | Austria
10259 | CENTC | 7/25/1996 | 3.2500 | Mexico
10260 | OTTIK | 7/29/1996 | 55.0900 | Germany
10261 | QUEDE | 7/30/1996 | 3.0500 | Brazil
10262 | RATTC | 7/25/1996 | 48.2900 | USA
10267 | FRANK | 8/6/1996 | 208.5800 | Germany
Filter
Rename
Remove
Panel Filter
Simple view
X
[ShipCountry] = Germany
Panel Filter
Design view
X
Filter Column
ShipCountry
Comparison
=
Value
Germany
Add

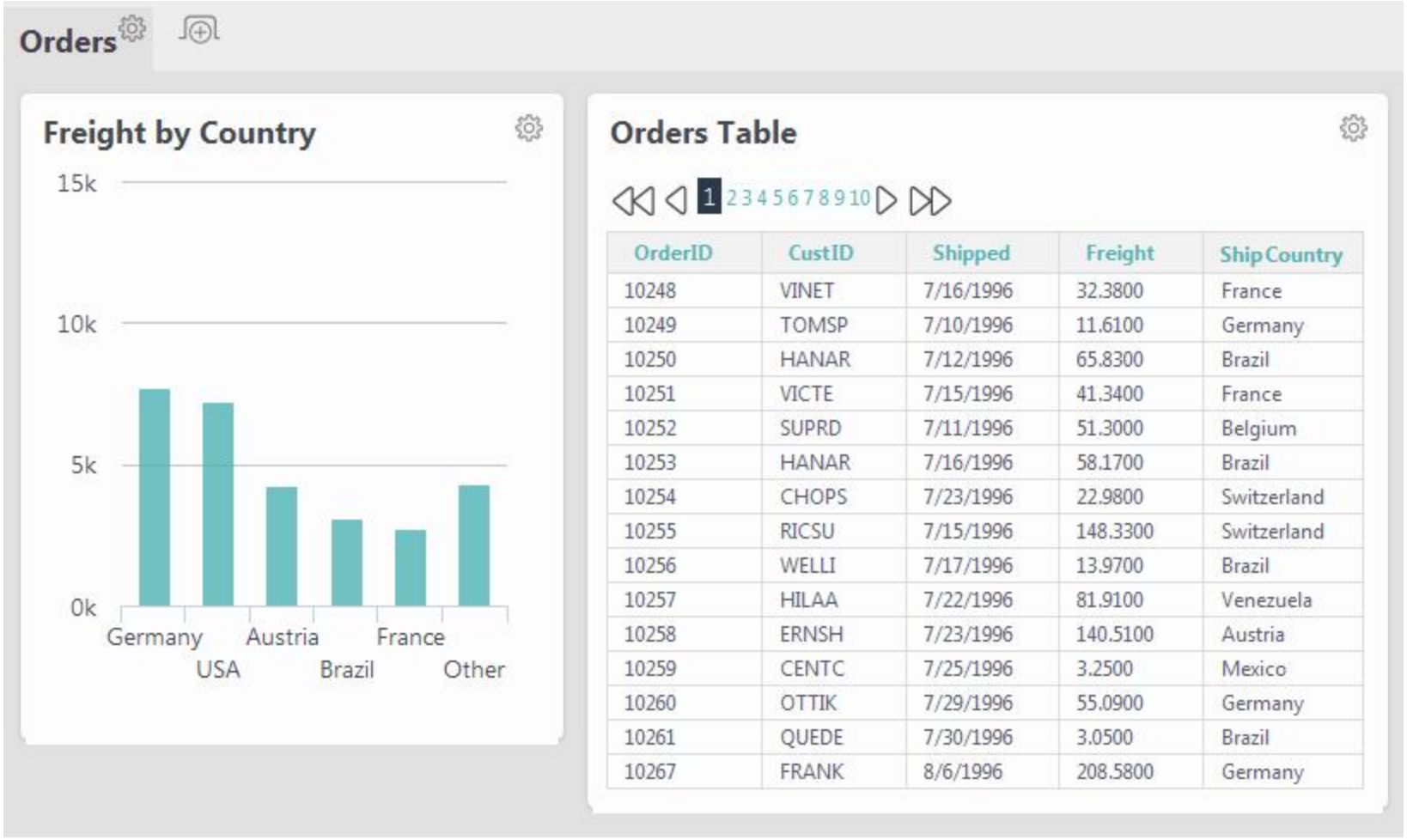
This option will open the **Panel Filter** panel, as shown above. The controls displayed in it will depend on the view mode, *Simple* or *Design*, configured by your application developer.



In Design view, you can define and combine filters, using the controls shown above. Adding, changing, or removing filters will immediately filter the data in the Dashboard panel accordingly.

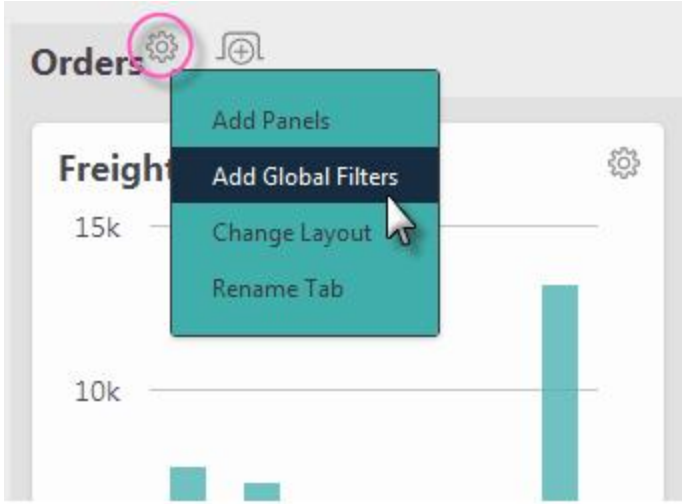
Adding a Global Filter

You can also add a "global" filter, one that affects the data used in *multiple* Dashboard panels at once. This type of filter functionality is automatically available when multiple Dashboard panels have been configured for filtering.

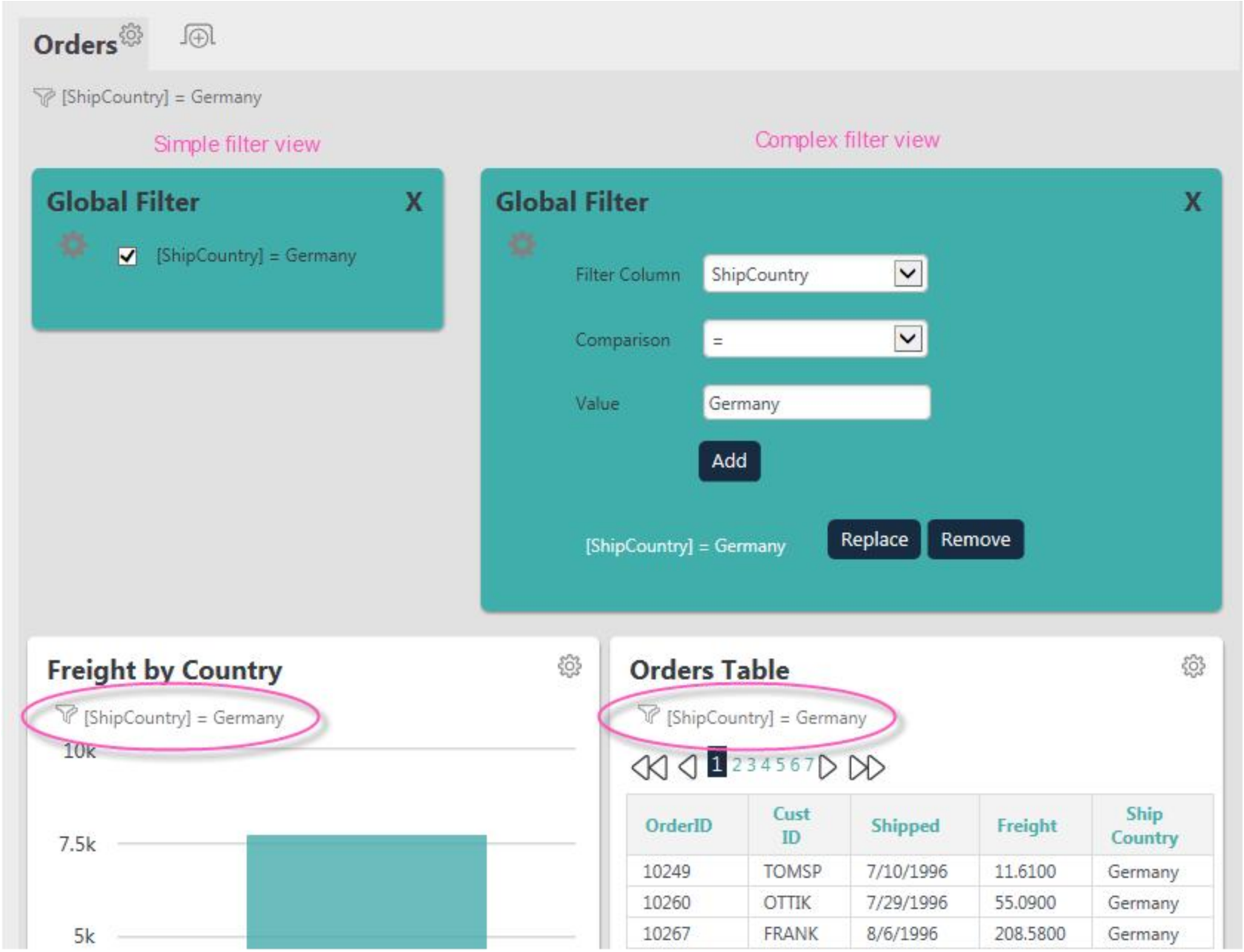


OrderID	CustID	Shipped	Freight	ShipCountry
10248	VINET	7/16/1996	32.3800	France
10249	TOMSP	7/10/1996	11.6100	Germany
10250	HANAR	7/12/1996	65.8300	Brazil
10251	VICTE	7/15/1996	41.3400	France
10252	SUPRD	7/11/1996	51.3000	Belgium
10253	HANAR	7/16/1996	58.1700	Brazil
10254	CHOPS	7/23/1996	22.9800	Switzerland
10255	RICSU	7/15/1996	148.3300	Switzerland
10256	WELLI	7/17/1996	13.9700	Brazil
10257	HILAA	7/22/1996	81.9100	Venezuela
10258	ERNSH	7/23/1996	140.5100	Austria
10259	CENTC	7/25/1996	3.2500	Mexico
10260	OTTIK	7/29/1996	55.0900	Germany
10261	QUEDE	7/30/1996	3.0500	Brazil
10267	FRANK	8/6/1996	208.5800	Germany

The example Dashboard shown above has two panels, each of which has been configured for filtering and whose visualizations use data from the ShipCountry column.

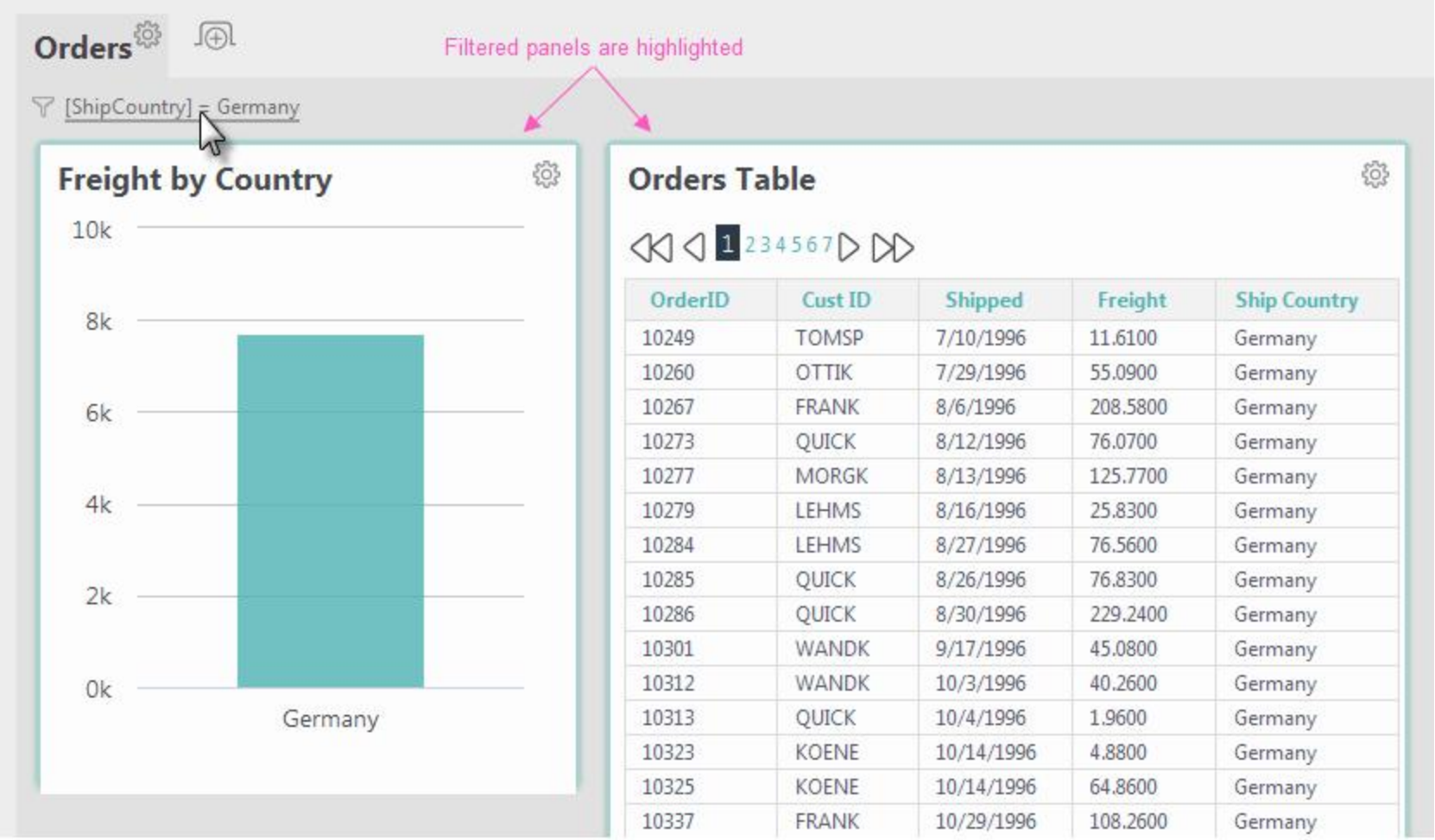


You can create a global filter by clicking the gear icon for the Dashboard tab, as shown above, and selecting the *Add Global Filters* pop-up menu option.

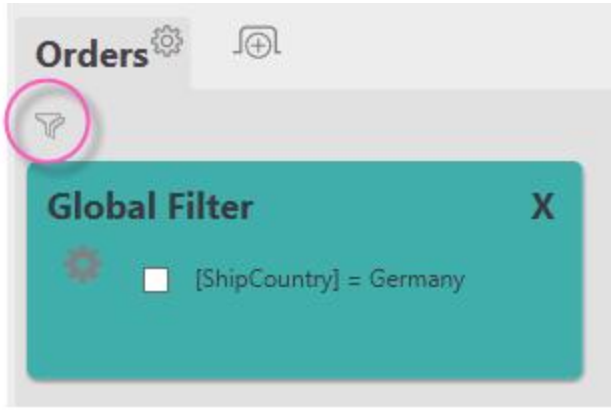

Orders
[ShipCountry] = Germany
Simple filter view
Complex filter view
Global Filter
X
[ShipCountry] = Germany
Global Filter
X
Filter Column
ShipCountry
Comparison
=
Value
Germany
Add
[ShipCountry] = Germany
Replace
Remove
Freight by Country
[ShipCountry] = Germany
10k
7.5k
5k
Orders Table
[ShipCountry] = Germany
1 2 3 4 5 6 7
OrderID
Cust ID
Shipped
Freight
Ship Country
10249
TOMSP
7/10/1996
11.6100
Germany
10260
OTTIK
7/29/1996
55.0900
Germany
10267
FRANK
8/6/1996
208.5800
Germany

The **Global Filter** panel will be displayed, as shown above. The controls displayed in it will depend on the configured view mode and have been explainedearlier. Once a filter has been created, Dashboard panels with visualizations whose data includes the filter'sdata columns will be filtered; those that do not include the filter'scolumns will not be filtered.

Filter descriptions, circled above, will be added to each filtered panel and can be clicked to re-open the Global Filter panel.



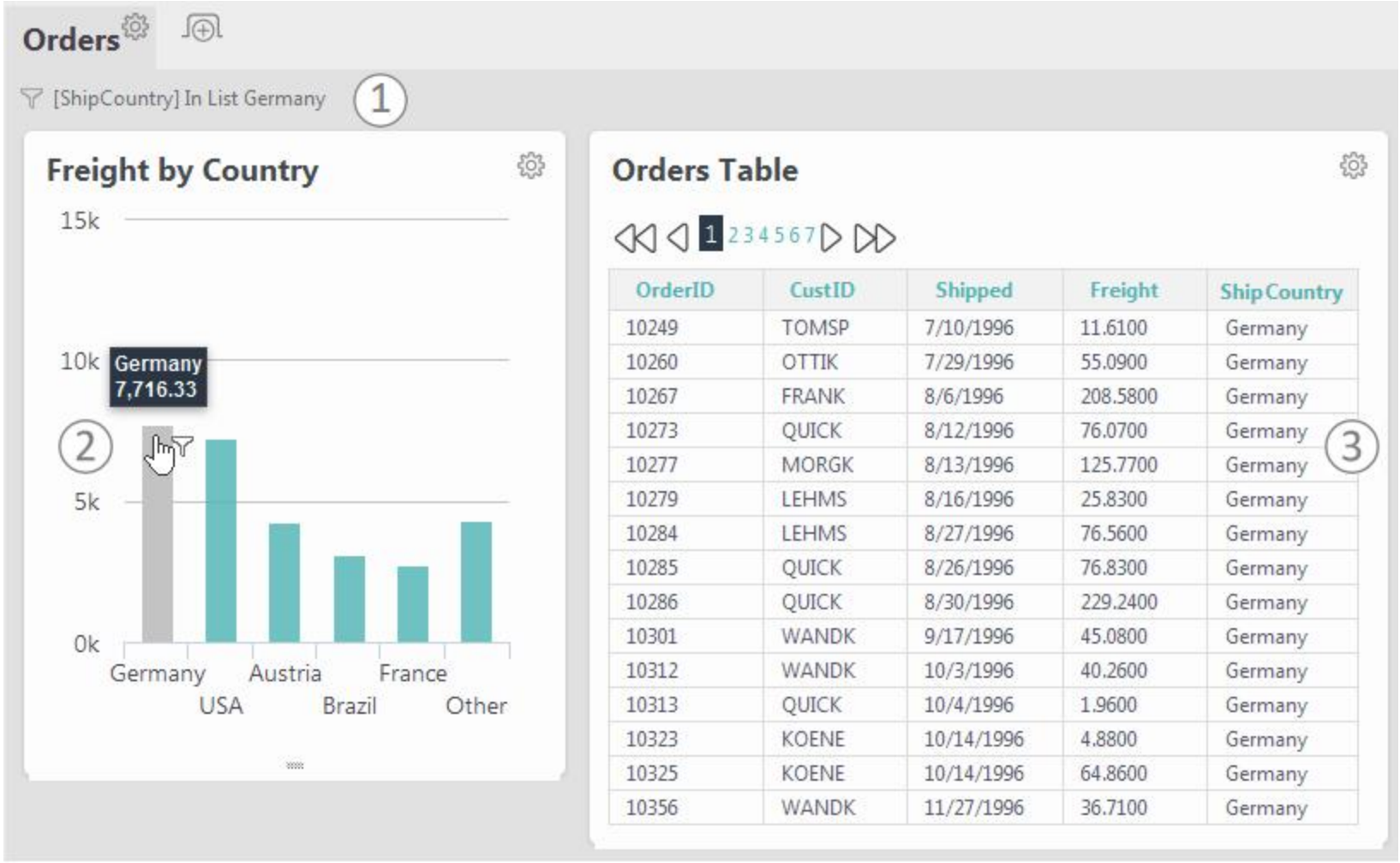
A global filter description will also appear at the top of the tab and can be used as a link to re-open the Global Filter panel. In addition, when you hover your mouse cursor over this description, the panels affected by the filter will be highlighted, as shown above.



When the Simple view mode is used, filters can be disabled by unchecking their check box. This causes the global filter description to be removed, but a filter icon, circled above, remains as an indicator that a global filter is defined and available for use. Clicking the icon will re-open the Global Filter panel.

Global Filtering using Chart Data Points

If your application has been configured to allow it, you can also create a global filter in a Dashboard by selecting data points in a chart:

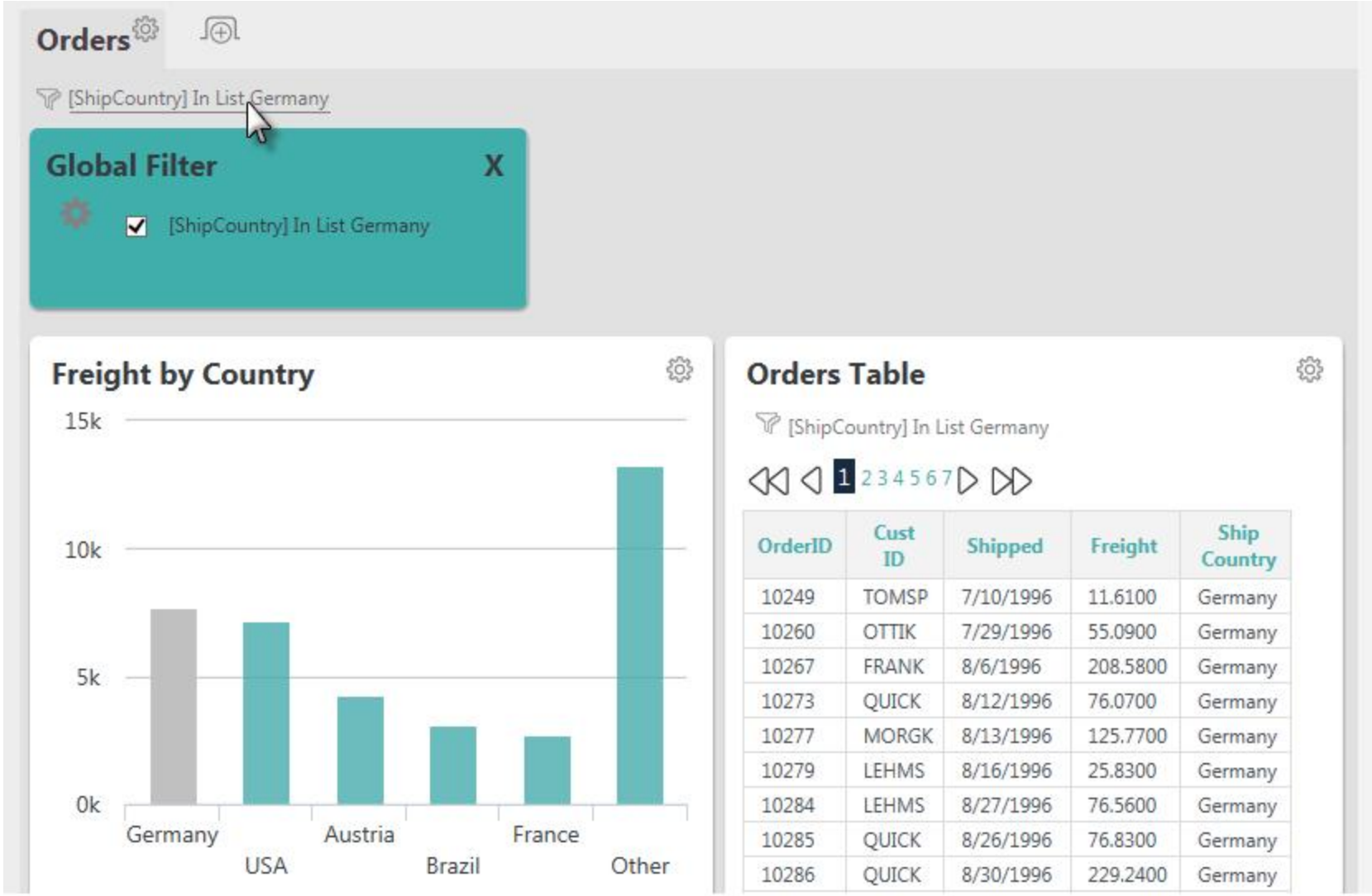


Once again, we have an example with two Dashboard panels, one with a chart and one with a table, shown above. When you click a bar in the chart, a global filter is created automatically and three other things happen:

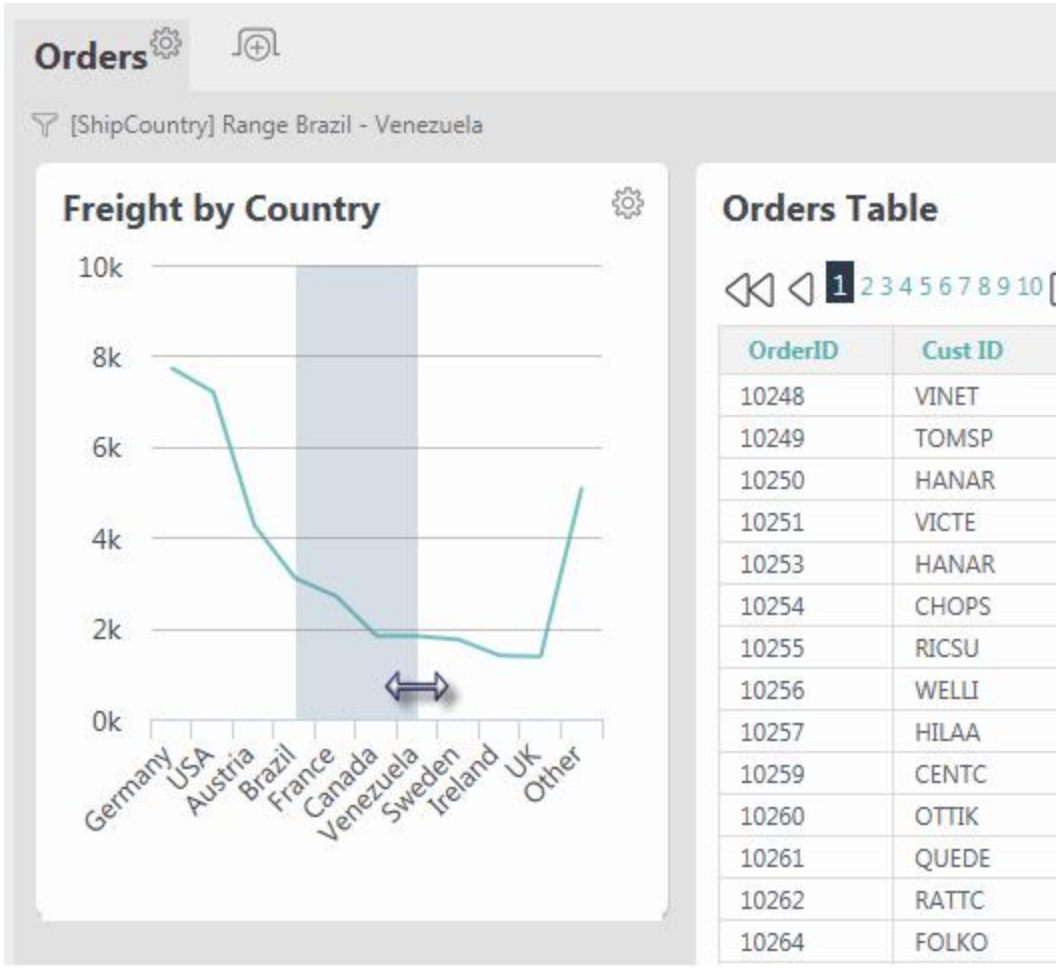
1. A Filter Description appears at the top of the Dashboard tab. This can be clicked to edit the global filter.
2. The color of the selected chart bar changes to indicate it's included in a filter.
3. The data for the table in the second panel is filtered.

This is different from the previous global filtering example because the chart that was clicked is *not* filtered.

Clicking the same chart bar a second time will cancel the filter related to that data. Clicking another bar will add additional criteria to the global filter.


Orders
[ShipCountry] In List Germany
Global Filter
X
[ShipCountry] In List Germany
Freight by Country
15k
10k
5k
0k
Germany
USA
Austria
Brazil
France
Other
Orders Table
[ShipCountry] In List Germany
1 2 3 4 5 6 7
OrderID | Cust ID | Shipped | Freight | Ship Country
10249 | TOMSP | 7/10/1996 | 11.6100 | Germany
10260 | OTTIK | 7/29/1996 | 55.0900 | Germany
10267 | FRANK | 8/6/1996 | 208.5800 | Germany
10273 | QUICK | 8/12/1996 | 76.0700 | Germany
10277 | MORGK | 8/13/1996 | 125.7700 | Germany
10279 | LEHMS | 8/16/1996 | 25.8300 | Germany
10284 | LEHMS | 8/27/1996 | 76.5600 | Germany
10285 | QUICK | 8/26/1996 | 76.8300 | Germany
10286 | QUICK | 8/30/1996 | 229.2400 | Germany

Clicking the Filter Description at the top of the Dashboard tab will cause the automatically-generated Global Filter panel to be displayed. As before, the controls displayed in it will depend on the view mode, Simple or Design, configured by your application developer (only Simple mode is shown above).



For chart types that support it, you can also create a global filter by dragging the cursor to select a range of values, as shown in the Line chart above. This will create a global filter that includes the selected range of values.

Drill To Dashboard Data

If your application has been configured for it, you can drill into Dashboards in SSRM to gain a deeper insight of the data, without leaving the context of the Dashboard Author. The Drill To feature is useful when reviewing underlying data, which makes up the aggregated number, to answer questions like: of my National sales figures, which states are performing better in the United States, and within each state, which territories are driving revenue? This feature is optional and only available in the Dashboard Author, not the Report Author.

The Drill To feature is available for the following charts:

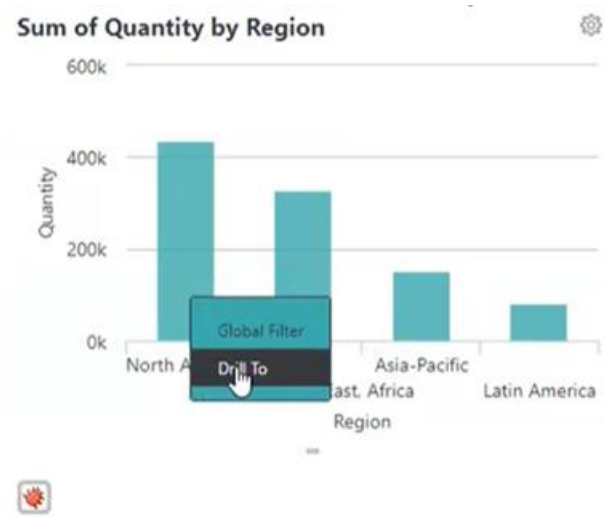
- Bar
- Pie- Date columns cannot be added
- Heat Map- Date columns cannot be added

This topic contains the following sections:

Using Drill To

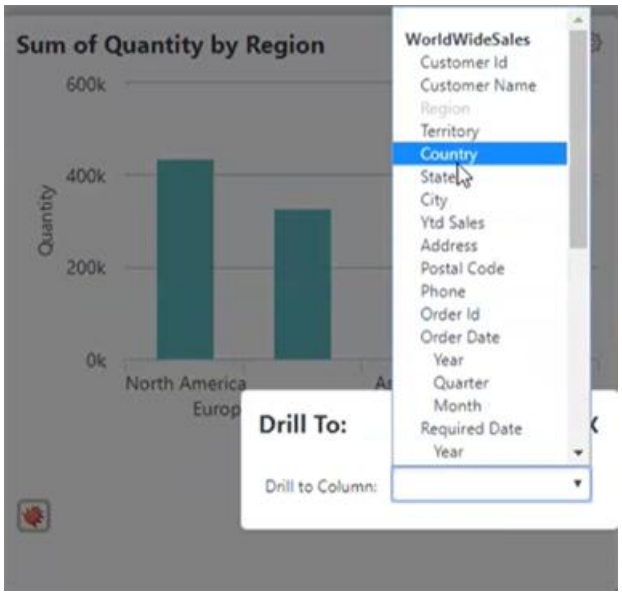
Follow the steps below to drill into your Dashboard data:

1. Select one of the available **columns**.
2. Then, select **Drill To**:



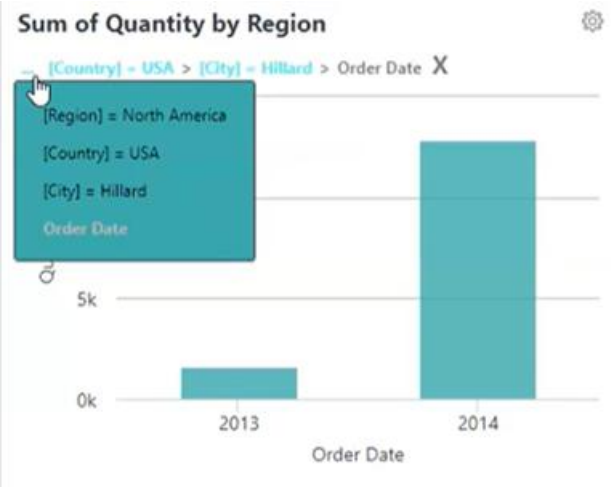
A Drill To: window displays.

3. Select the Drill to Column drop-down and select a **column** from the list, as shown below. Calculated columns that were previously established in SSRM are also available as a Drill To Column.



Info reconfigures the chart to display the data.

The Drill To path displays as a breadcrumb menu at the top of the panel. If there is not enough space on the chart, the bread-crumb menu will show the first and last node with an ellipses. Additionally, if you re-size the chart, the breadcrumb menu will automatically adjust the Drill To path to reflect these changes. In these instances, the ellipses may be the only part of the Drill To path that displays. The ellipses is selectable. Once selected, it will display a drop-down list for the Drill To path, which is also selectable:



Return to any previous drill states by selecting the node in the drill path, or, select **X** to return the chart to its original state.

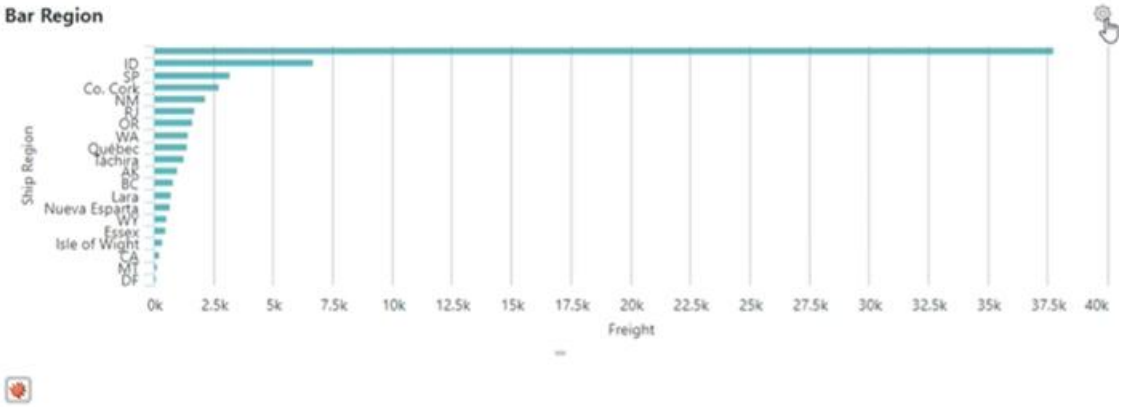
When editing a visualization from the Dashboard, your Drill To state will be kept as long as there are no changes made to the column currently used by the chart. For example, if your chart displays Freight x Order Date and you edit the visualization to display Freight Sum instead of Freight Average, the breadcrumb menu will remain intact. However, if you edit the visualization to display Ship Region x Freight, the breadcrumb menu will reset. For more information about editing a visualization, see "Editing Visualizations" on page 60.

Drill to Date/Time Columns

You can drill into Date/Time Columns by Year, Month, or Quarter in the SSRM Dashboard.

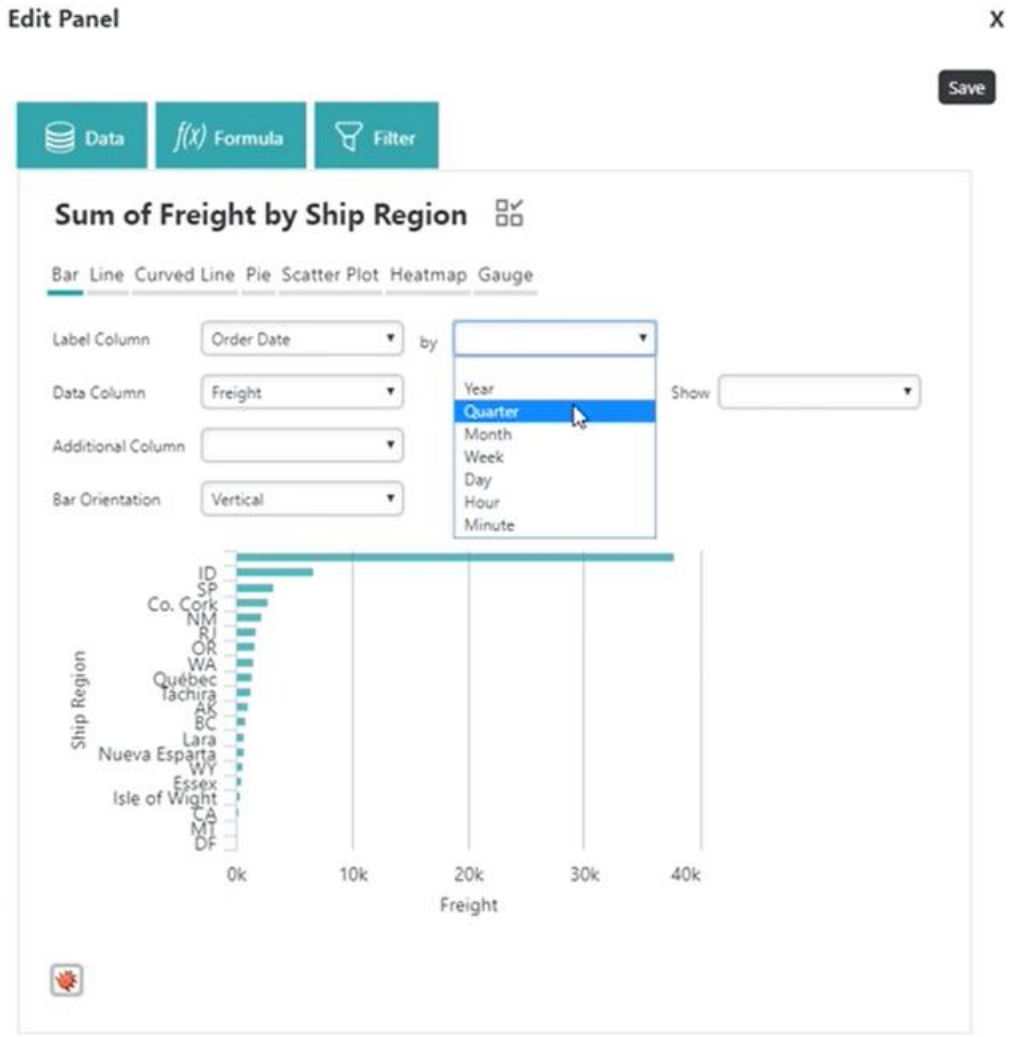
The Date/Time Drill To only applies to bar charts, not Pie Charts or Heat Maps.

1. To edit the Bar Chart from the SSRM Dashboard, select the **gear** icon:



The settings options display.

2. Select **Edit** from the list of options. An Edit Panel displays.
3. Depending on your chart axes, select the Label Column or Data Column drop-down and choose **Order Date**.
4. Then, select the Date/Time column by Year, Quarter, or Month:



You cannot drill to Week, Day, Hour, or Minute using the Dashboard Drill To feature.

5. Select **Save** to save your changes. The Edit Panel closes and the chart reconfigures to display your changes.
6. Select the desired **column** to trigger the Drill To menu. A Drill To: window displays.
7. Select the Drill to Column drop-down and select a **column** from the list:

Info reconfigures the chart to display the data.

Saving and Sharing Drill To

Upon saving, the Drill To selection will be saved in the bookmark.

Shared bookmarks will contain the Drill To selection (if any) applied by the Dashboard Owner.

If the bookmark was shared with you, not created by you, you will not be able to further Drill To or change the current Drill To selection in the Dashboard.

Editing Visualizations

If your application has been configured for it, certain visualizations shown in Dashboard panels can be edited and then saved.

If this features has been enabled, you'll see an *Edit* option in a panel's gear icon drop-down menu, as shown above. If you click that option, a pop-up panel containing the original controls used to configure the visualization will appear.

You can then change the visualization and save it back into the Dashboard. v23.1 If your Dashboard has been configured for it, the Dashboard Panel Title will update automatically according to edits that affect the Visualization Title. For example, if you

change the Data Column of the visualization to calculate the Average, the Dashboard Panel Title automatically updates to "Average of.." (instead of the existing "Sum of..", as shown above).

This feature depends on specific things about the source of the visualization so it's possible the *Edit* option might be available for some Dashboard panels but not for others.

When editing a Dashboard panel, the pop-up editing panel will contain an Analysis Grid configured with all the current settings for the panel content. You edit your panel content using it, with these restrictions:

- The Analysis Grid's *Filter* tab will only be available when editing visualizations created in Info v12.7+.
- No new charts or crosstabs can be created when editing, so their tabs will not be visible.
- You may not delete or hide/close existing charts or crosstabs when editing them.

When editing a visualization from the Dashboard, your Drill To state will be kept as long as there are no changes made to the column currently used by the chart. For example, if your chart displays Freight x Order Date and you edit the visualization to display Freight Sum instead of Freight Average, the breadcrumb menu will remain intact. However, if you edit the visualization to display Ship Region x Freight, the breadcrumb menu will reset. For more information about the Drill To Dashboard feature, see "Drill To Dashboard Data" on page 52.

Exporting Panel Content

Tables and Crosstabs in Dashboard panels can be exported to Excel or CSV, using the panel's gear icon drop-down menu:

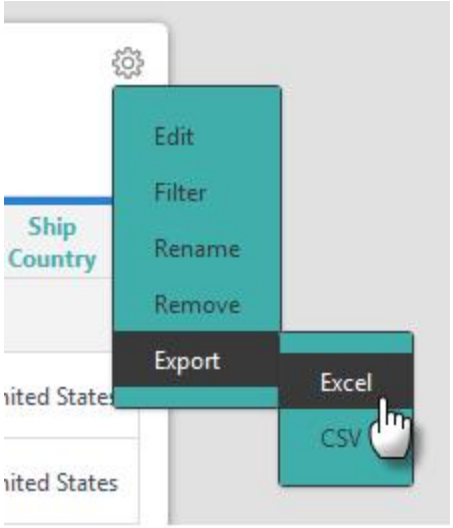


Table data will be exported to a file of the appropriate format.

The Analysis Grid for End Users

The Analysis Grid has its own user interface that allows you manipulate data, create charts, change table layouts, and much more, at runtime.

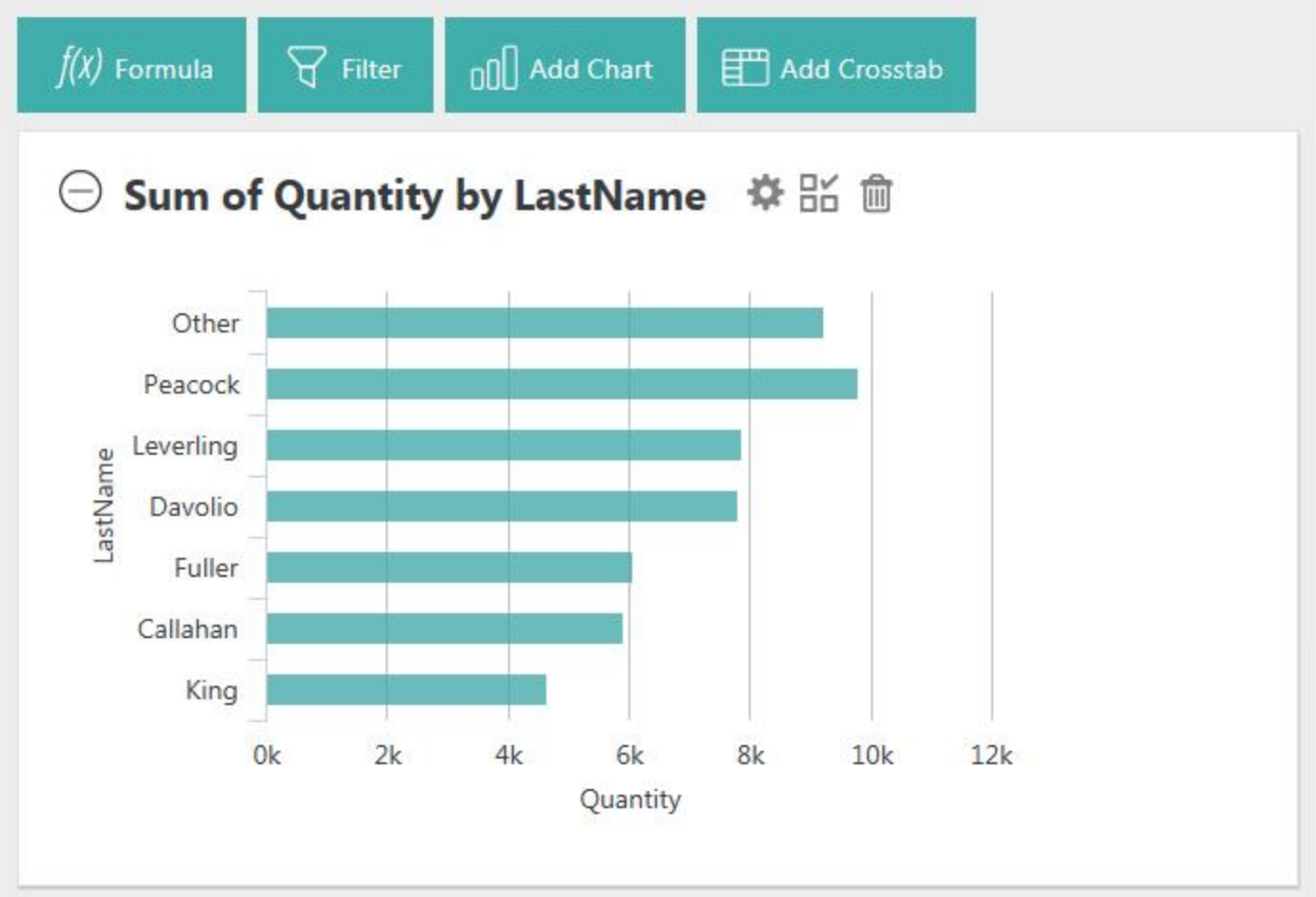
The following topics introduce Logi Info *end-users* to the Analysis Grid:

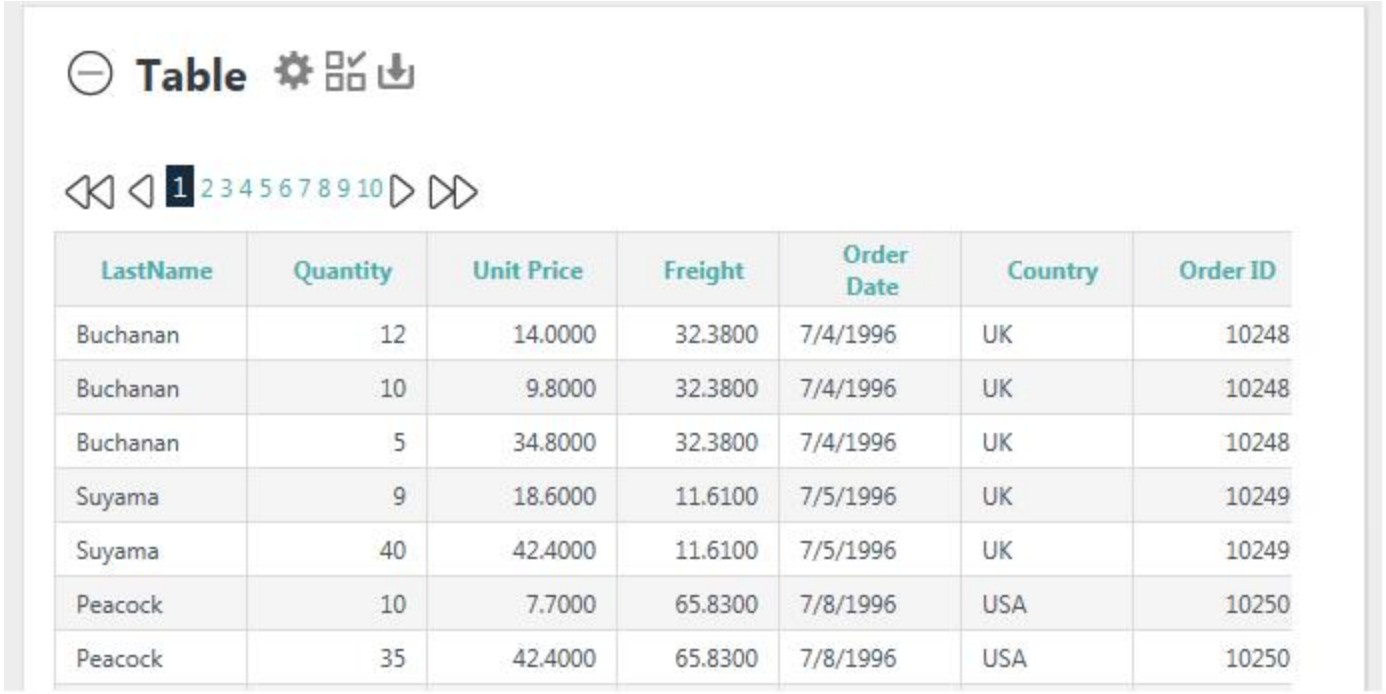
- Selecting Data
- Adding Formula Columns
- Filtering Rows
- Showing, Hiding, and Moving Columns
- Sorting Rows
- Grouping Rows
- Creating Aggregations
- Creating Custom Aggregations
- Controlling Paging
- Formatting Data
- Creating Charts and Gauges
- Editing Chart Labels and Captions
- Pivoting and Summarizing Data
- Zoom Control
- Exporting Data
- Adding Analysis Grid Charts to Dashboards

Information for *developers* is available in *The Analysis Grid for Developers*.

About the Analysis Grid

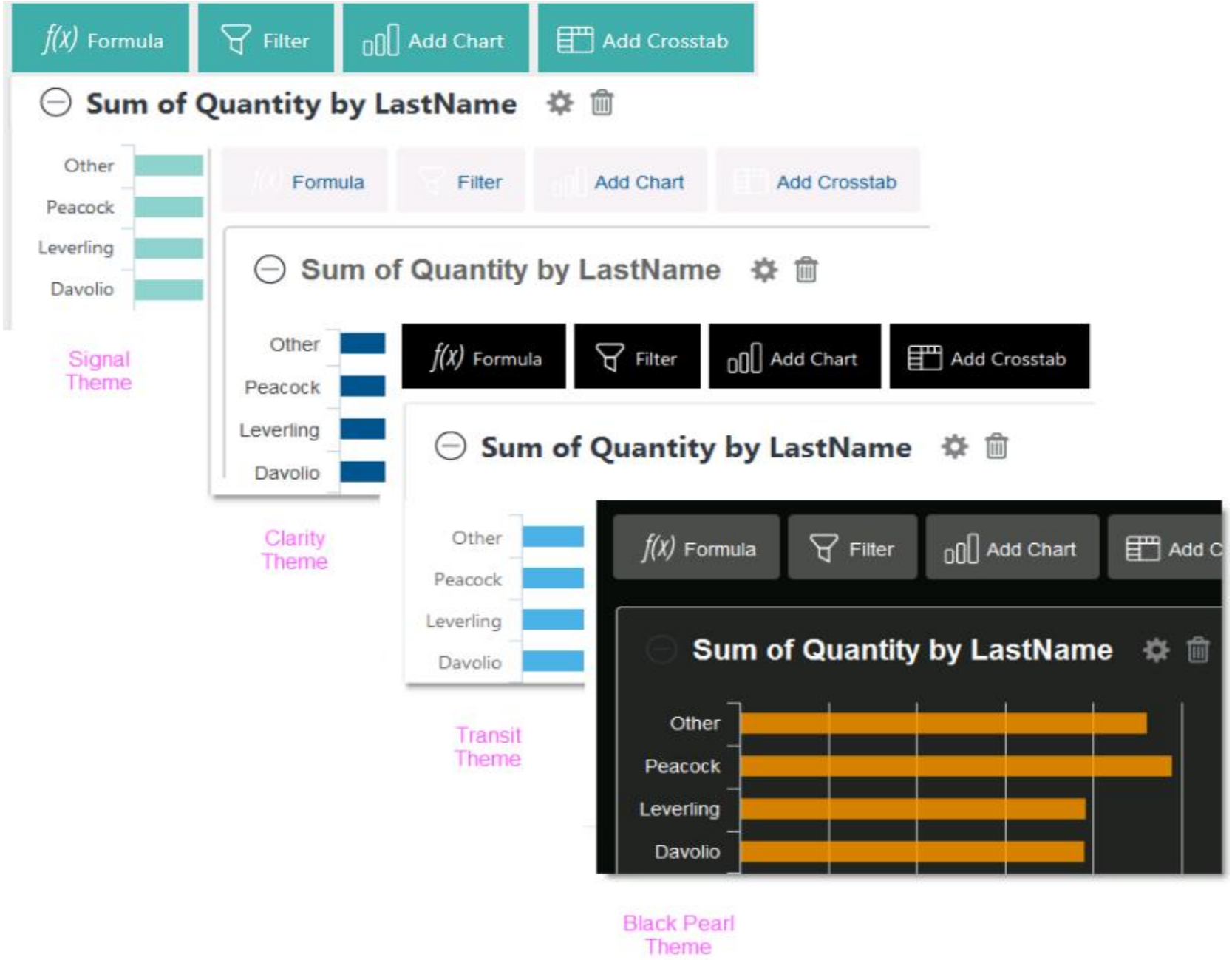
The Analysis Grid user interface consists of separate panels for controls, tables, and visualizations. At runtime, you can manipulate the controls, creating data analyses and visualizations on the fly. The Analysis Grid below displays both a chart and table:





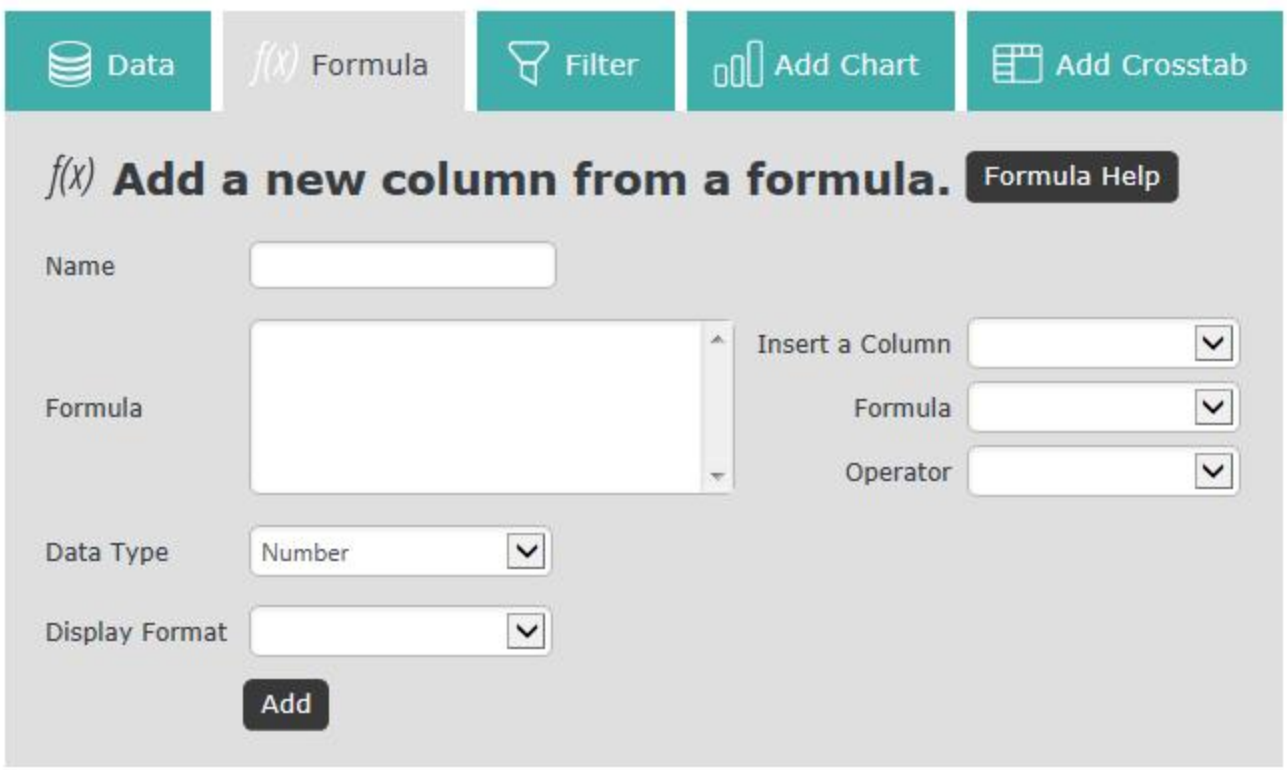
LastName	Quantity	Unit Price	Freight	Order Date	Country	Order ID
Buchanan	12	14.0000	32.3800	7/4/1996	UK	10248
Buchanan	10	9.8000	32.3800	7/4/1996	UK	10248
Buchanan	5	34.8000	32.3800	7/4/1996	UK	10248
Suyama	9	18.6000	11.6100	7/5/1996	UK	10249
Suyama	40	42.4000	11.6100	7/5/1996	UK	10249
Peacock	10	7.7000	65.8300	7/8/1996	USA	10250
Peacock	35	42.4000	65.8300	7/8/1996	USA	10250

As you can see in the examples below, the Analysis Grid can look quite different depending on the Theme being used. The examples in this topic use the Signal Theme.


Formula
Filter
Add Chart
Add Crosstab
Sum of Quantity by LastName
Other
Peacock
Leverling
Davolio
Signal Theme
Clarity Theme
Transit Theme
Black Pearl Theme

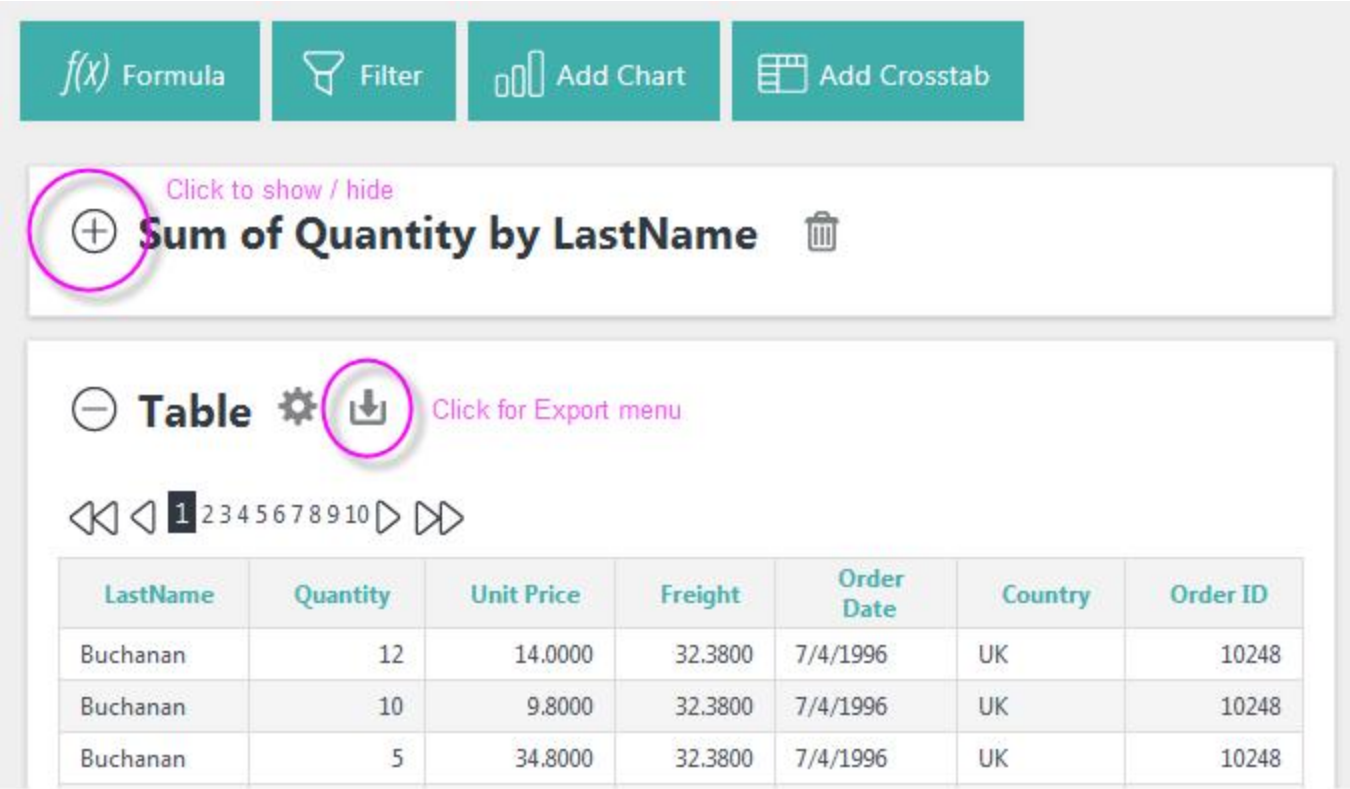
The Analysis Grid consists of a set of panels. The **Controls Panel**, at the top, includes a set of tabs or buttons that allow you to manipulate the data and visualizations at runtime. The visibility of specific controls is configured by the application developer and so you may or may not see the full set shown here.

When a tab or button is clicked, a configuration panel for that feature appears, as shown abelow. Clicking it again hides the panel.



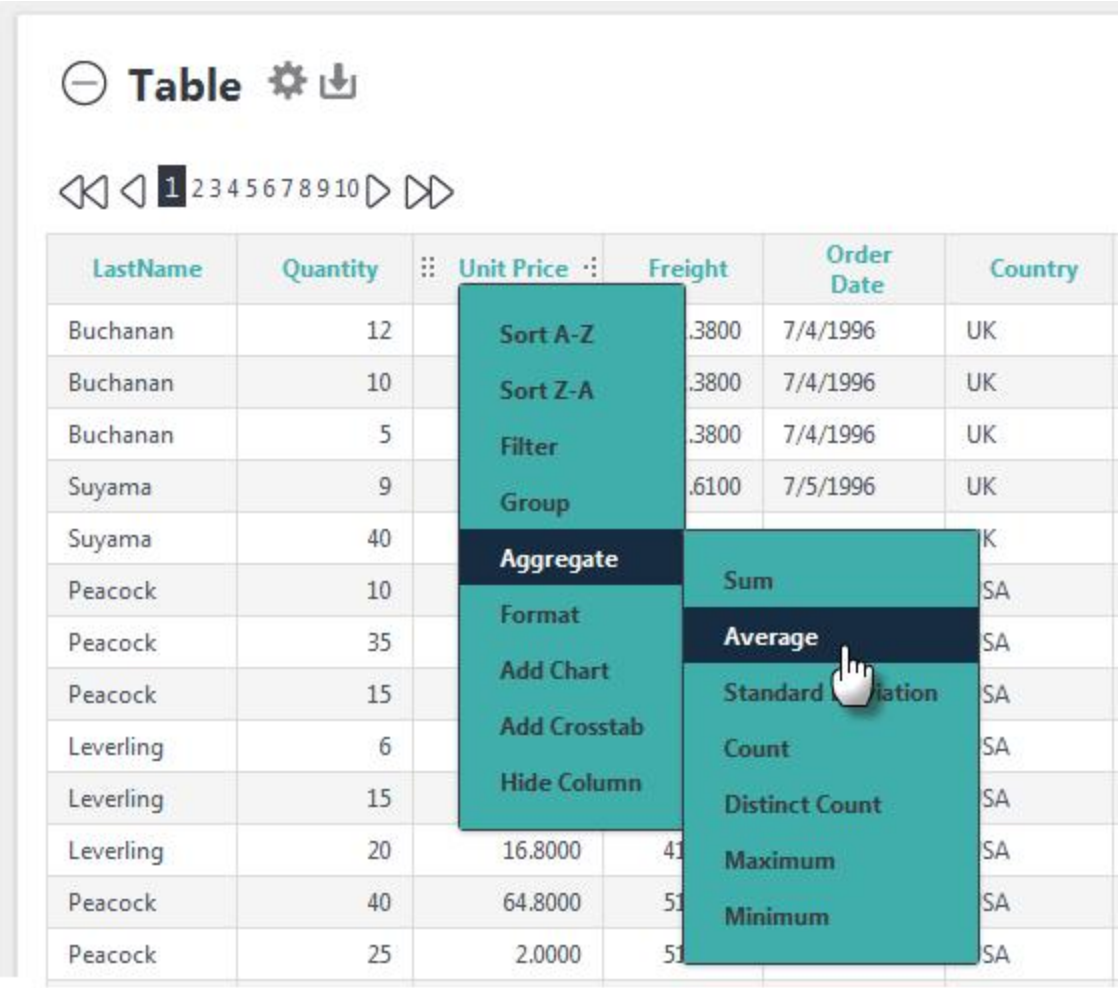
Visualization Panelscontain either a Data Table, Crosstab Table, or chart. Panels can be collapsed or expanded using the "+" and "-" icons. You can also rearrange their order by clicking with your mouse near the top of a panel, and dragging it up or down.

Panels containing tables may display an icon for exporting them to Excel, CSV, or PDF. The availability of the different types of exports is under developer control, so those available to you may vary.



Clicking a panel's **gear** icon will display a configuration area for the specific visualization. The Table configuration area, shown below, allows you to control a variety of table features.

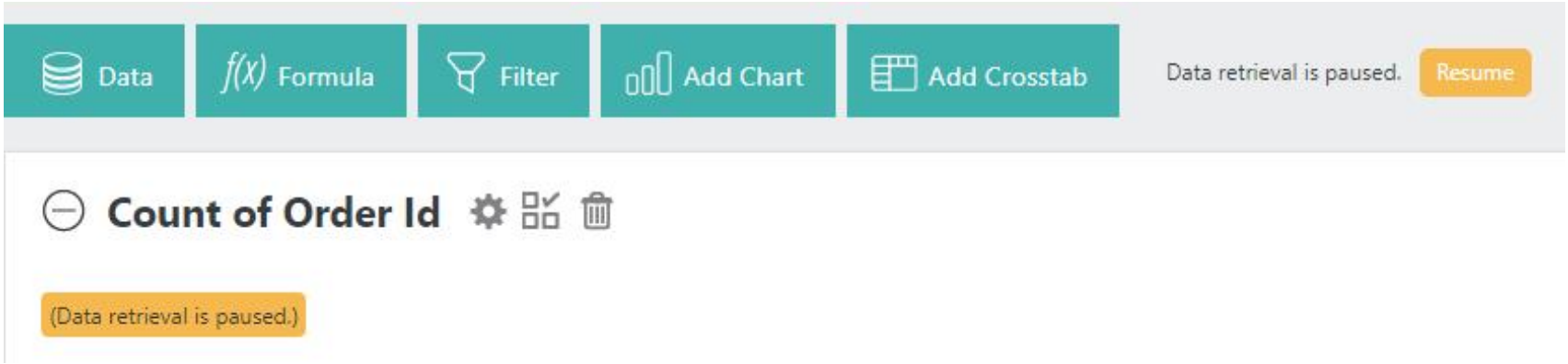
Table column *headers* contain a variety of features. They can be *dragged*to rearrange the column order or to adjust column widths. When column header text is clicked, drop-down menus provide sorting, filter, formatting, grouping, and other features, as shown below. The availability of each feature can be restricted during development.



If your application has been configured for it, **Undo** and **Redo** icons also display on the Control Panel, circled below. These allow you to undo and redo your recent actions.

Additionally, if your application has been configured for it, a **Pause Data Retrieval** button displays next to the Control Panel. When clicked, it temporarily halts data retrieval from the server. This feature is useful when data retrieval is slow and you prefer to make several changes to the Analysis Grid without having to wait for data after each change.

Click the **Resume** button, shown below, to resume retrieval.

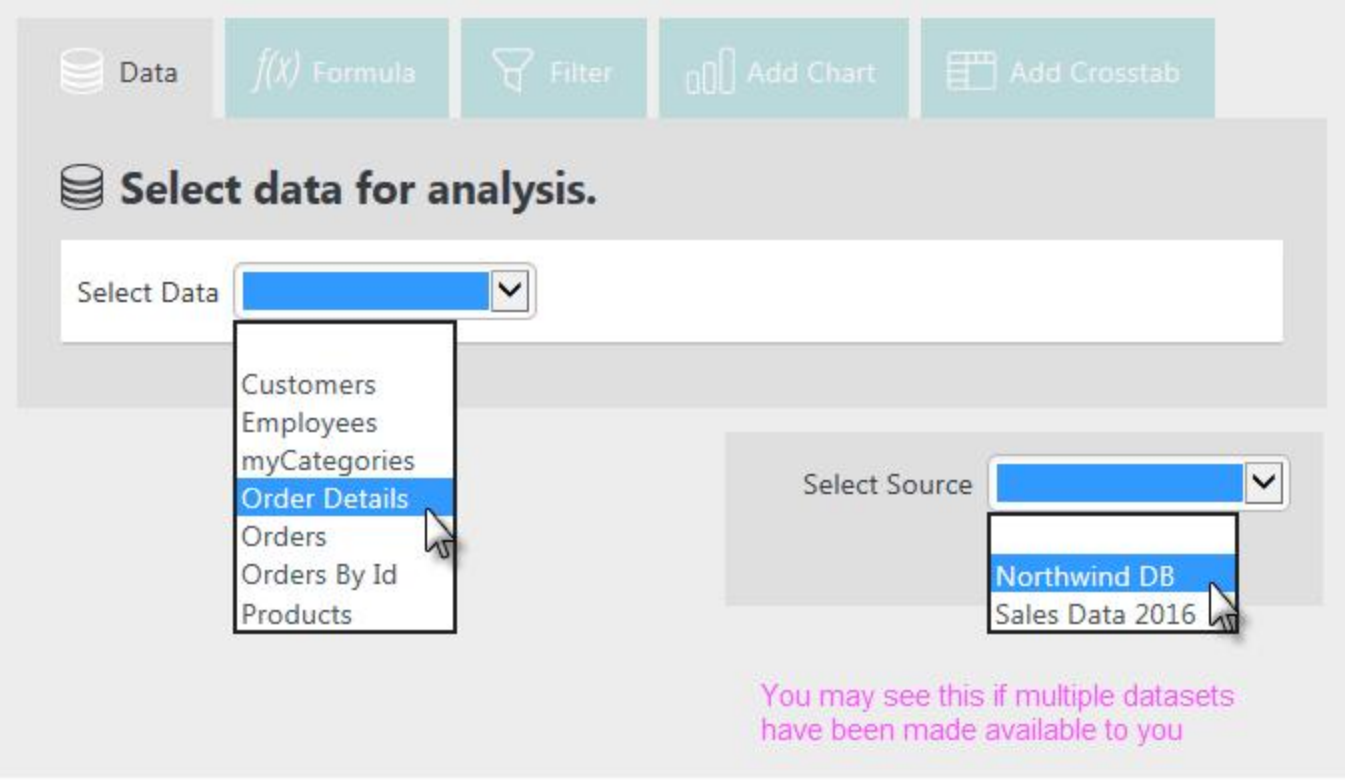


Depending on your application's configuration, this feature may be enabled by default. As always, you can select **Resume** to retrieve data results.

Analysis Grid - Selecting Data

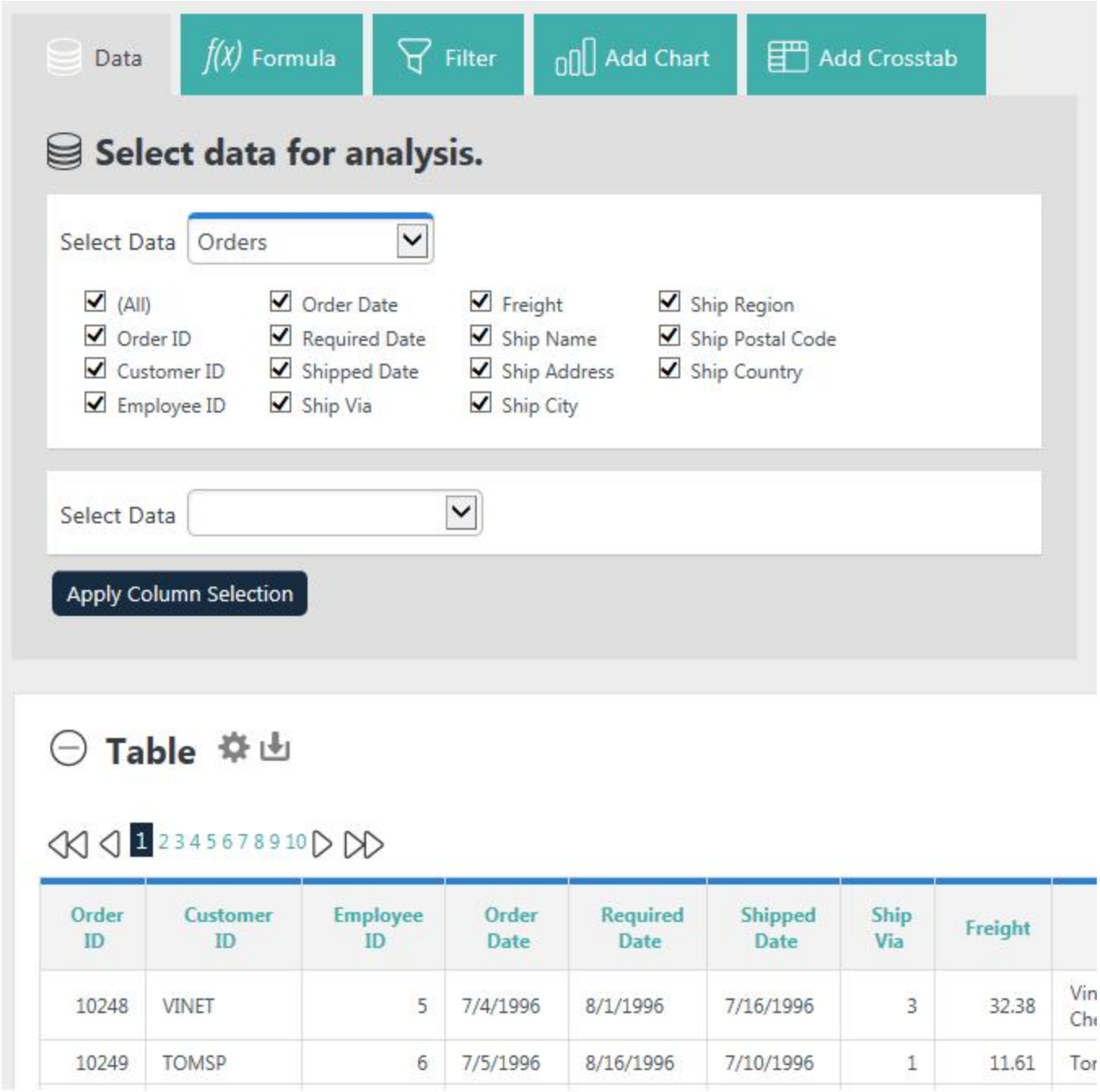
Depending on how your application has been configured, you may or may not see this feature. Skip this topic if you do not see a "Data" tab or button at the top of your Analysis Grid.

Click the "Data" tab or button to open its panel:



The first thing you'll need to do is select the Data Table or view you want to work with. The data source tables and views available to you have been determined by the application developer.

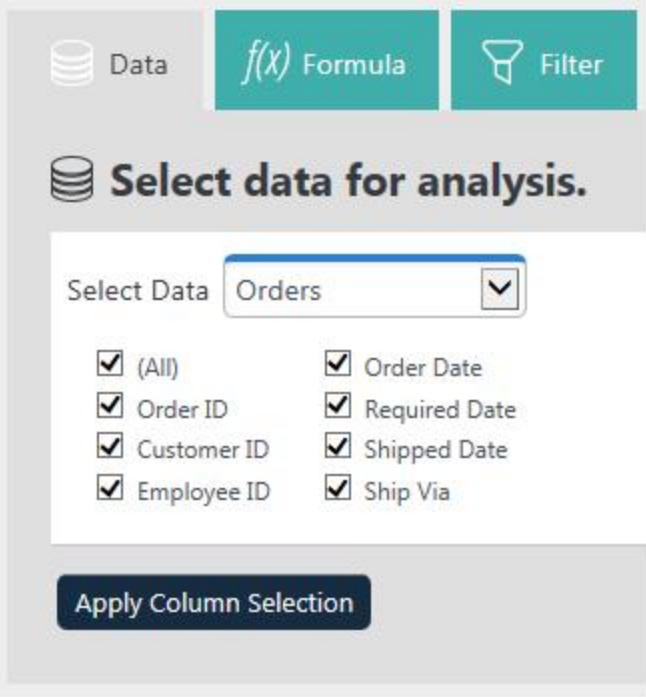
Your application may be configured for multiple data sources; if so, it will look like the example shown above, right. If that case, you'll need to select a data source first, then a table or view.



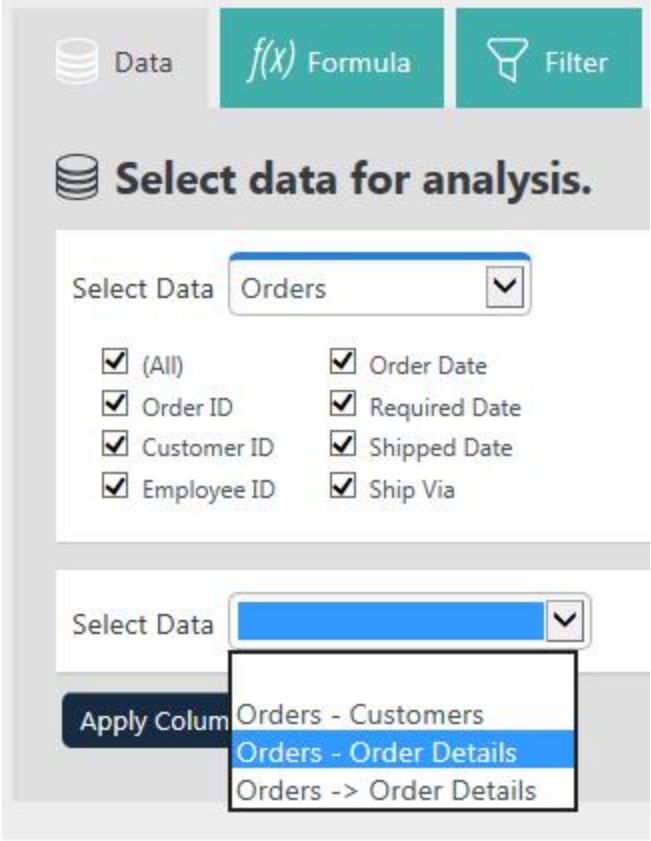
Column Selection

Data Table

As soon as you select data, a Data Table will appear showing it. A set of column selection check boxes will also appear, as shown above. You can un-check the boxes for columns you *don't* want to include in your analysis work. If it's visible, click **Apply Column Selection** to update the table, otherwise the table may update automatically.



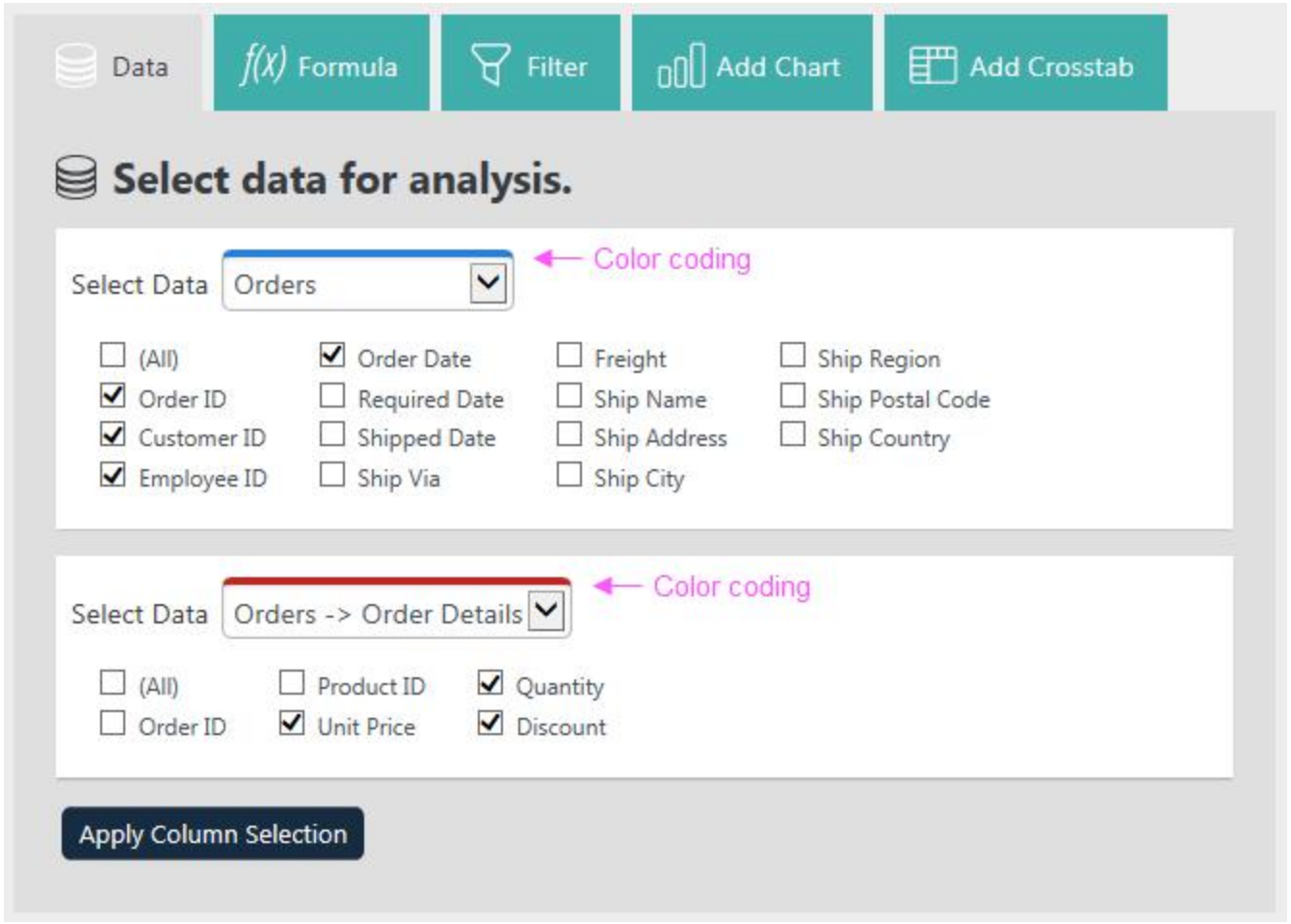
No Joins available



Multiple Joins available

Your application may be configured to allow you to "join" different Data Tables. If so, you'll see additional data selection lists, as shown above left. By default, items like `Orders - Customers` indicate an *Inner Join*, while items like `Orders` → Order Details indicate a *Left Outer Join*. However, these designations can be customized by your application's developer and other Join types may be available.

What's a "join"? A join combines two sets of data to produce a single dataset. Different types of joins produce different results. For example, an *Inner Join* selects all rows from both tables as long as there is a match between a column in both tables. A *Left Outer Join* selects all rows from the first table and adds rows from the second table that match a specified column value.


Data
f(x) Formula
Filter
Add Chart
Add Crosstab
Select data for analysis.
Select Data
Orders
Color coding
(All)
Order ID
Customer ID
Employee ID
Order Date
Required Date
Shipped Date
Ship Via
Freight
Ship Name
Ship Address
Ship City
Ship Region
Ship Postal Code
Ship Country
Select Data
Orders -> Order Details
Color coding
(All)
Order ID
Product ID
Unit Price
Quantity
Discount
Apply Column Selection

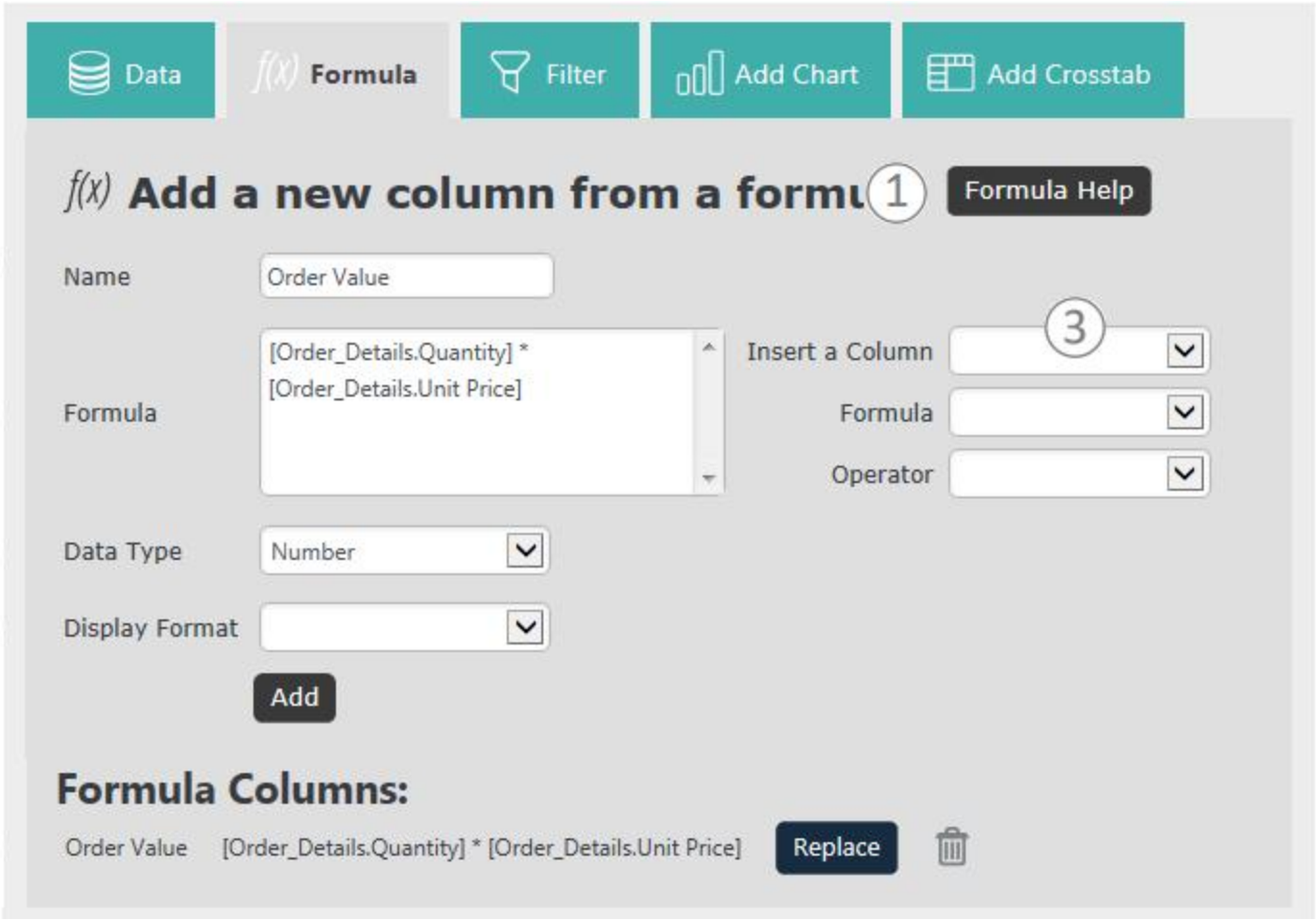
When you select data that joins tables, you'll see a color-coding scheme applied to the table that indicates where data came from. In the example shown above, table columns with a ***blue*** border came from the original table "Orders" while columns with a ***red*** border came from the selected join.

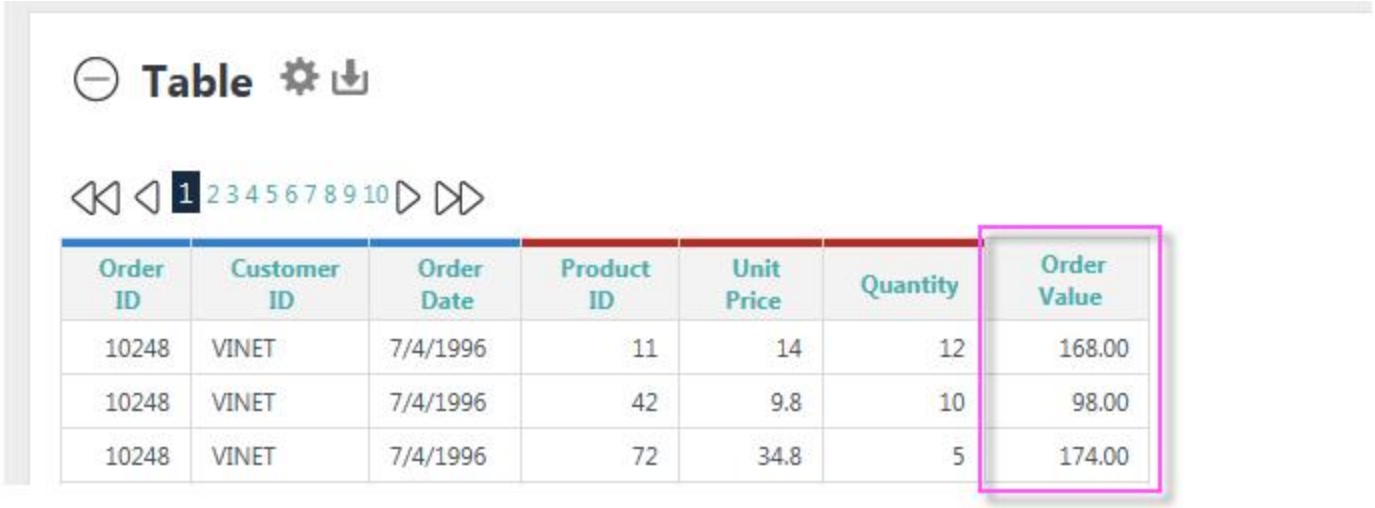
Once you've selected data, all of the other tabs or buttons at the top of the Analysis Grid become enabled.

Click the **Data** tab or button to hide the data selection controls.

Analysis Grid - Adding Formula Columns

Select the **Formula** tab to add calculated columns to the data:





New columns are added at the right side of the table but can be relocated by dragging them. Here's how to use this feature:

1. Help constructing a formula is available via the **Formula Help** button. The Formula Help pages have been relocated; if you created a browser bookmark for them, you'll need to update it.
2. Enter the **Name** for the column that will be added to the table.
3. **Insert** column names, functions, and operators into the Formula box by selecting them here.
4. And/or enter the formula by typing it into the **Formula** box. Column names should be enclosed within square brackets [ ] and typical math operator symbols, such as + - * / should be used. You can always edit or delete anything in this space. You can enter formulas that don't contain data columns. You can also add an aggregate functions into the formula.
5. Specify the **Data Type** for the new column.
6. Specify a **Display Format**. Formatting options include numeric and date formats. Click **Add** to create the new column and refresh the table.
7. As **Formula Columns** are created, they're added to the Formula Columns list. Use the adjacent **Replace** button and **Remove** (trash can) icon to manage the list. Columns that have been added are now included in the list of available columns (3) for use in other formulae.

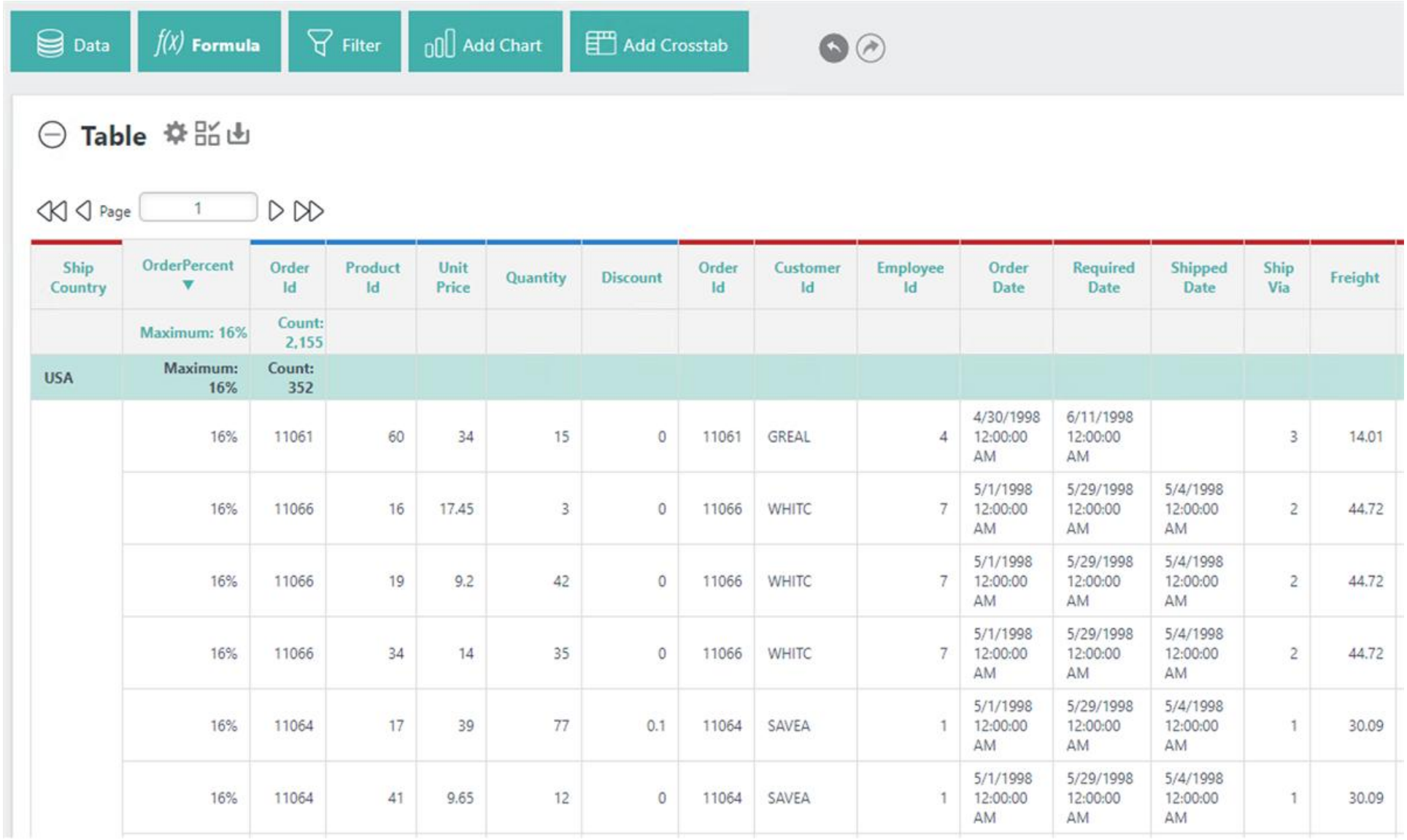
Select the **Formula** tab or button to hide the controls when done.

Your application developer will have to advise you concerning which function set is valid in the Formula box. These can vary depending on the way in which data is retrieved from the data source. In once case, JavaScript-style functions can be used, but in another case only functions supported by the database server itself can be used. The help shown when you click the **Formula Help** button will provide useful information depending on the data source.

To prevent creation of unmanageable tables, numeric type Formula columns are not available for use in a Crosstab Table as the Header Values Column or the Label Values Column.

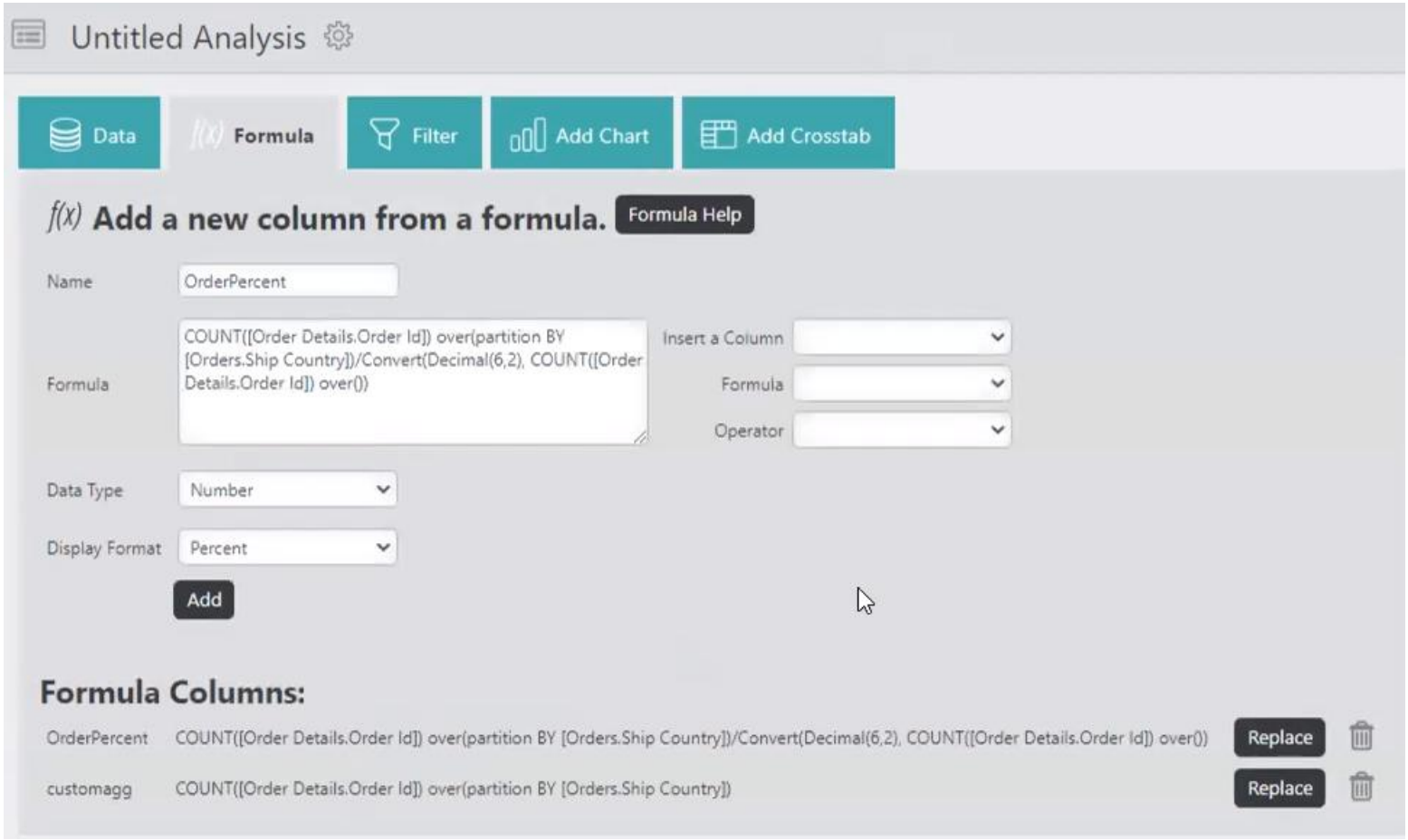
Adding Aggregate and Group Formulas

You can also use aggregates and group functions in formulas in your Analysis Grid. In the example below, there are two values that demonstrate the two-level order count for the Order Id. The first row includes the global order count, while the second row specifies the order count for that country. These values are shown as both a number and a percentage. By adding an aggregate function to the formula, you can automatically calculate the second value and have it display in your table.



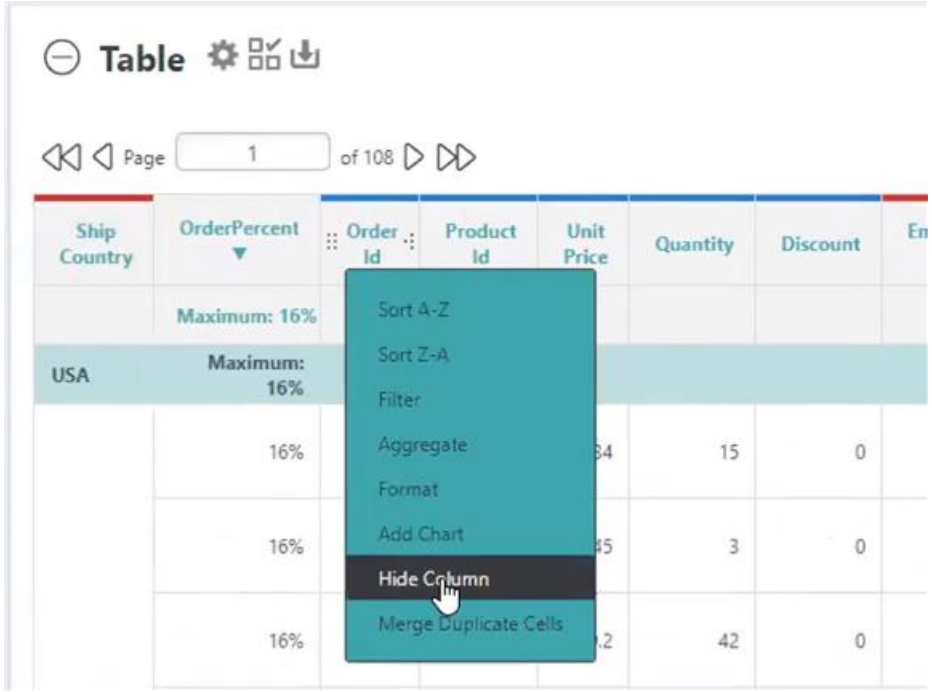
Ship Country	OrderPercent	Order Id	Product Id	Unit Price	Quantity	Discount	Order Id	Customer Id	Employee Id	Order Date	Required Date	Shipped Date	Ship Via	Freight
	Maximum: 16%	Count: 2,155												
USA	Maximum: 16%	Count: 352												
	16%	11061	60	34	15	0	11061	GREAL	4	4/30/1998 12:00:00 AM	6/11/1998 12:00:00 AM		3	14.01
	16%	11066	16	17.45	3	0	11066	WHITC	7	5/1/1998 12:00:00 AM	5/29/1998 12:00:00 AM	5/4/1998 12:00:00 AM	2	44.72
	16%	11066	19	9.2	42	0	11066	WHITC	7	5/1/1998 12:00:00 AM	5/29/1998 12:00:00 AM	5/4/1998 12:00:00 AM	2	44.72
	16%	11066	34	14	35	0	11066	WHITC	7	5/1/1998 12:00:00 AM	5/29/1998 12:00:00 AM	5/4/1998 12:00:00 AM	2	44.72
	16%	11064	17	39	77	0.1	11064	SAVEA	1	5/1/1998 12:00:00 AM	5/29/1998 12:00:00 AM	5/4/1998 12:00:00 AM	1	30.09
	16%	11064	41	9.65	12	0	11064	SAVEA	1	5/1/1998 12:00:00 AM	5/29/1998 12:00:00 AM	5/4/1998 12:00:00 AM	1	30.09

Below is an example of the new formulas, over(partition BY [xxxxxx]), used to display the two order counts:

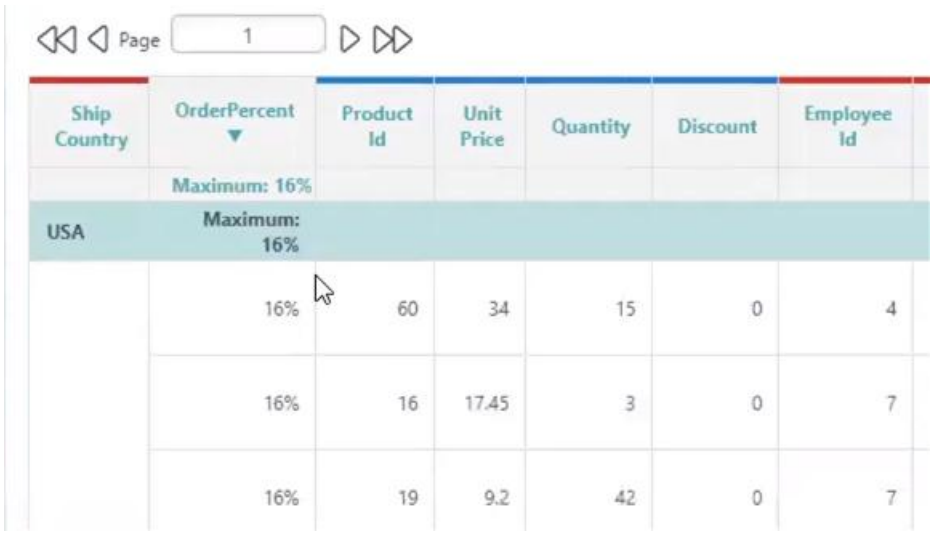


This feature can also be useful if you want to hide a column, but still want the count to be visible in your table. For example, since we created an Aggregate Formula for "customagg" that displays the Order ID and Ship Country Count, we can hide those columns in the table and still see the value.

Select the Order ID column > **Hide Column**:



The column Order ID column is no longer visible in the table, but the value for the order ID count is still available as a column (you may need to scroll to the end of the table):



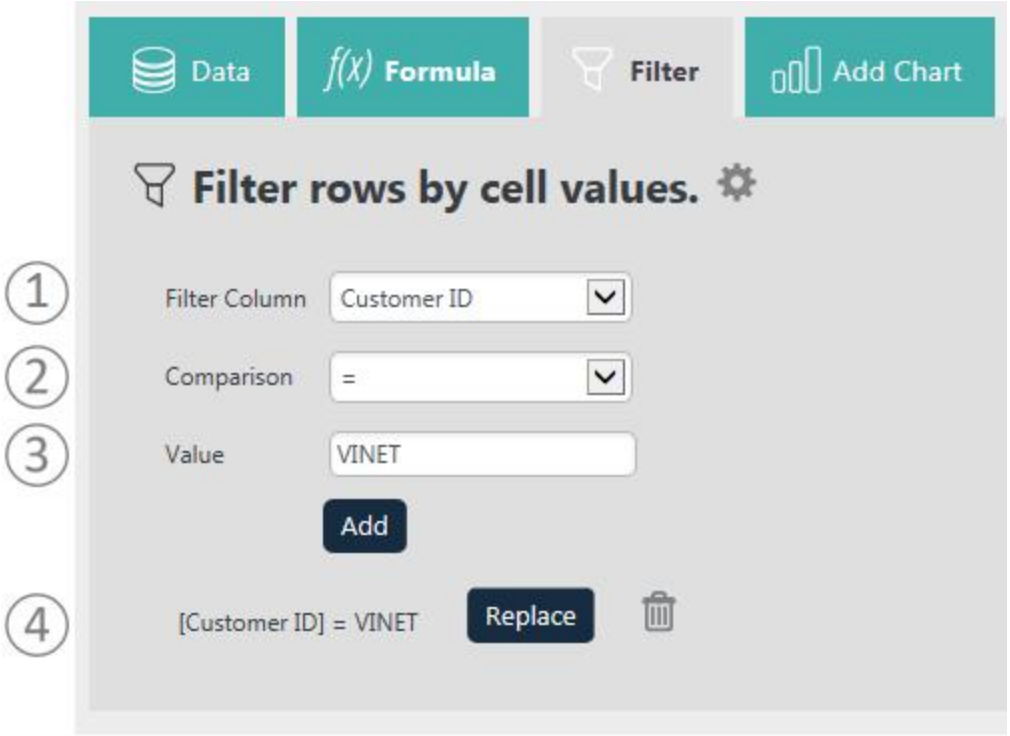
Page 1

Ship Country	OrderPercent	Product Id	Unit Price	Quantity	Discount	Employee Id
	Maximum: 16%					
USA	Maximum: 16%					
	16%	60	34	15	0	4
	16%	16	17.45	3	0	7
	16%	19	9.2	42	0	7

Product Id	Unit Price	Quantity	Discount	Employee Id	Order Date	Required Date	Shipped Date	Ship Via	Freight	Ship Name	Ship Address	Ship City	Ship Region	Ship Postal Code	Order Id	Customer Id	customagg
60	34	15	0	4	4/30/1998 12:00:00 AM	6/11/1998 12:00:00 AM		3	14.01	Great Lakes Food Market	2732 Baker Blvd.	Eugene	OR	97403			352
16	17.45	3	0	7	5/1/1998 12:00:00 AM	5/29/1998 12:00:00 AM	5/4/1998 12:00:00 AM	2	44.72	White Clover Markets	1029 - 12th Ave. S.	Seattle	WA	98124			352
19	9.2	42	0	7	5/1/1998 12:00:00 AM	5/29/1998 12:00:00 AM	5/4/1998 12:00:00 AM	2	44.72	White Clover Markets	1029 - 12th Ave. S.	Seattle	WA	98124			352

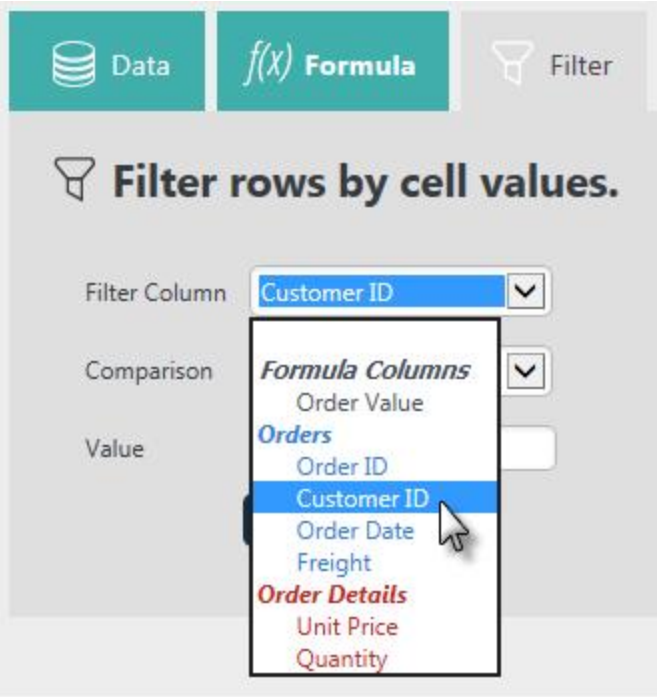
Analysis Grid - Filtering Rows

Select the **Filter** tab to remove table rows that don't meet your criteria:



Here's how to use this feature:

1. Select the **Filter Column** containing the values to be compared.



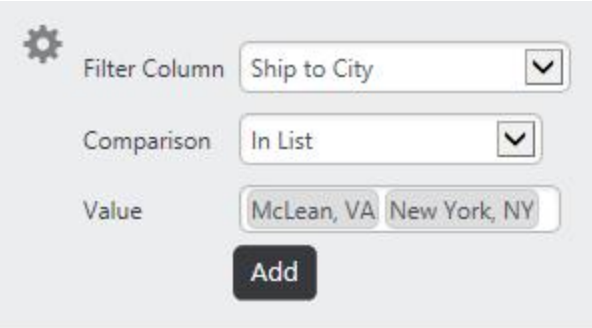
As shown above, you'll see that the options are grouped and color-coded to make it easier for you to identify them. If you created any Formula columns, they'll be in there, too.

2. Set the filtering criteria by selecting a **Comparison** operator from the list.

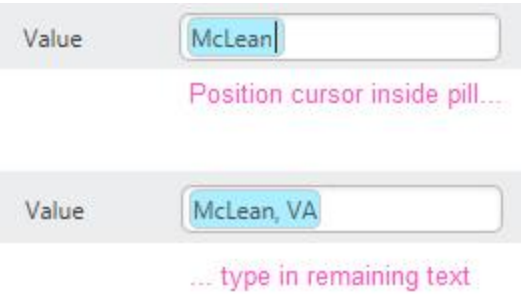
   Comparison operators include = , <, >, < =, > =, *Not* =, *Starts With*, *Not Start With*, *Contains*, *Not Contains*, Like, Not Like. If the Filter Column is a date, then *Date Range* is available and some other options are not. The *Starts With* and *Contains* operators are useful for finding values at the beginning or within text data. The *Not Contains* and *Does Not Start With* operators work in the opposite manner.

   Comparison operators *In List* and *Not In List* allow comparison against a comma-separated list of values you enter in the Value text box.

When using *In List* or *Not In List* comparisons in filters, multiple values can be entered, separated by commas. However, the values themselves may *include* commas, causing incorrect comparisons.



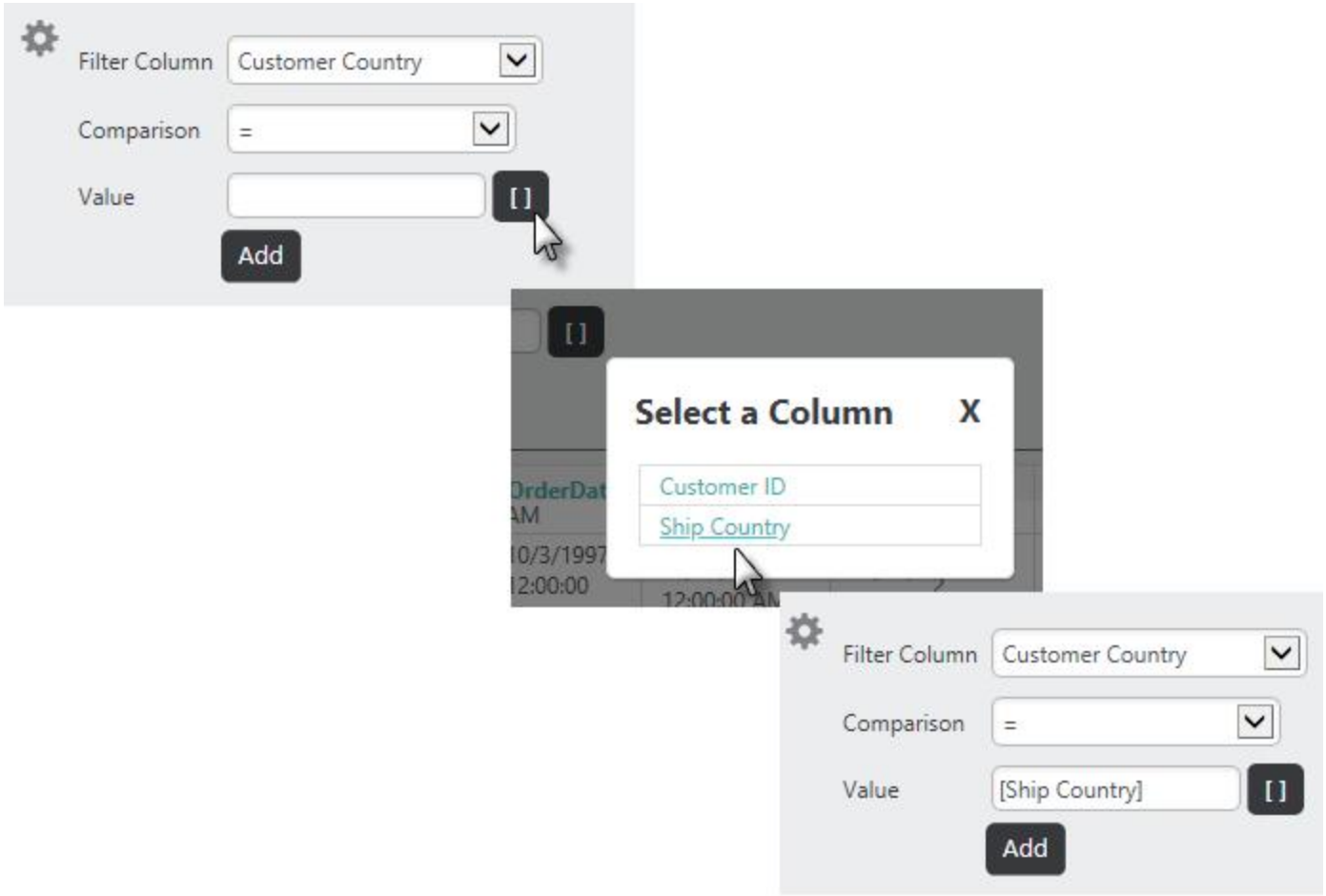
To address this, values for these types of filters are now represented in the controls by visual "pills" that enclose the complete value, as shown above. The filter uses the complete string within the pill during its comparison operation.



Pills with included commas are created by entering the first part of the value and pressing comma. This creates a blue pill as shown above. Then place your cursor *inside* the pill and type the rest of the value. Press tab to exit the pill and repeat the process for the next value, if desired.

The Filter feature is usually used to compare column data and a value you enter, but you can also directly compare the

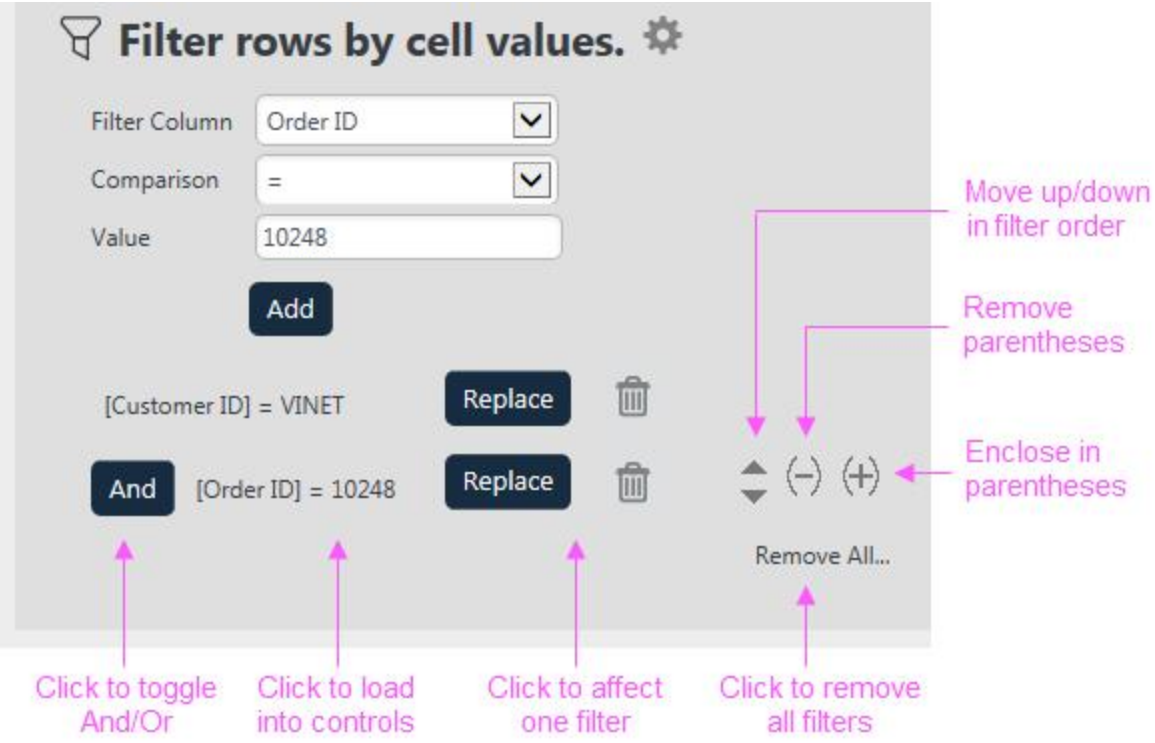
values in two data columns:



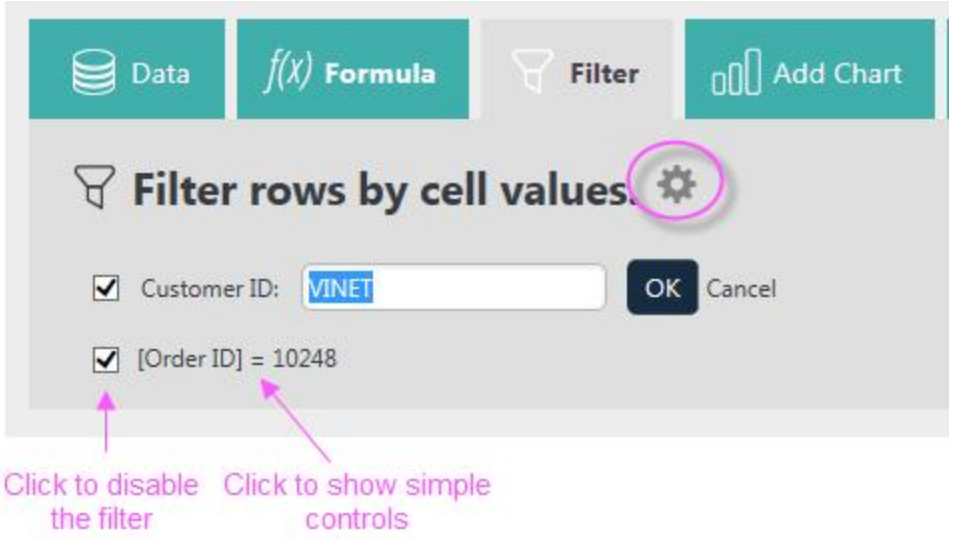
To do this, first select a Filter Column, as shown above, and a supported Comparison operator (= < > <= >= Not=). Then, select the second column from the list displayed when you click the **[]** button, or enter the column name within square brackets.

Depending on the comparison chosen, additional input controls may be displayed, for example, for date ranges. Or you may see a browse button that lets you select values for comparison from a pop-up list of choices.

3. Enter a comparison **Value**. v23.1 The Value field uses a Quick Filter function. Enter characters to filter the list of available Values. Wildcard characters (*, %) are *not* allowed in these values. Click **Add**. Rows that don't meet these criteria will be removed from the table.
4. As filters are created, they're added to the filter list. Use the adjacent **Replace** button and **Remove** (trash can) icon to manage the list.



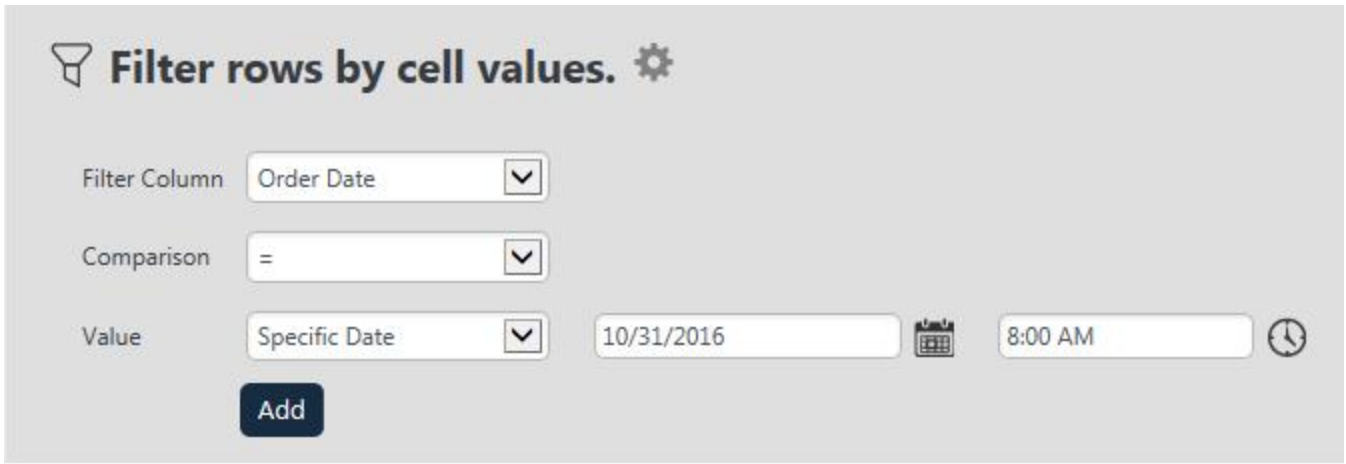
If you add multiple filters, only rows that meet *all* the conditions will be retained (an "And" situation). Clicking the *And* link in the Filters list, shown above, changes it to an *Or* link, so rows that meet *any* of the conditions will be retained. A set of four arrow icons will also appear by the trash can icon. These can be used to re-order the precedence of the filters or to group them together in various arrangements using parentheses.



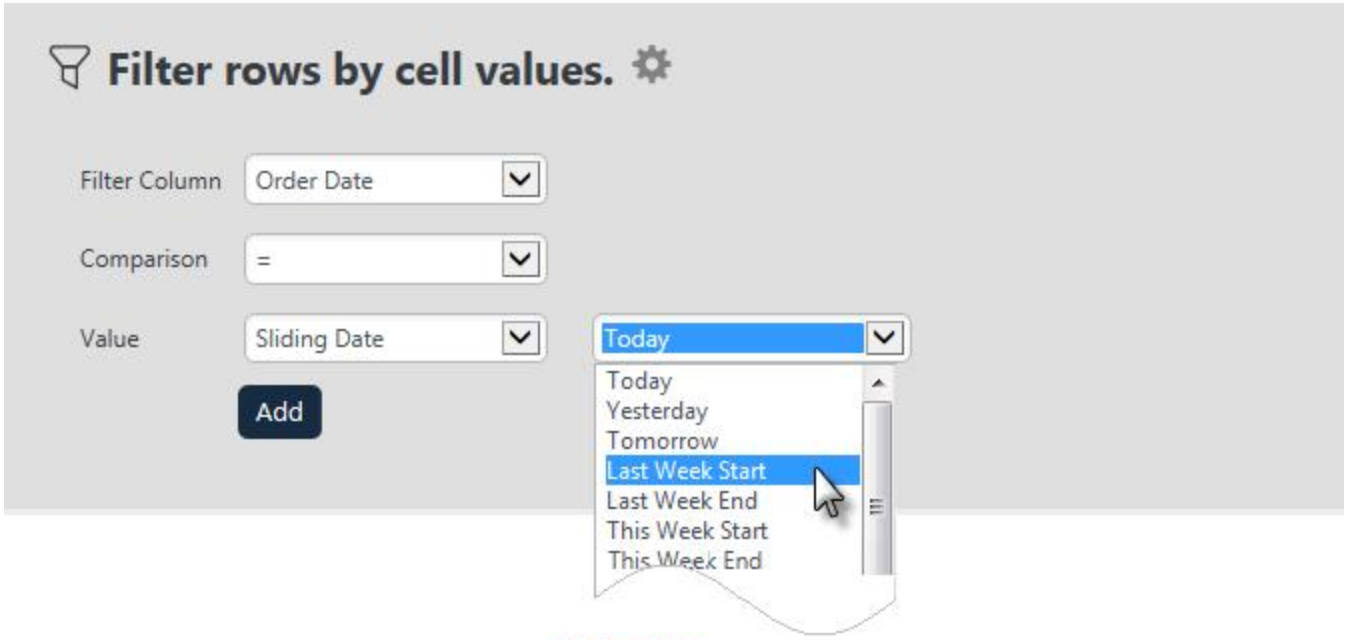
Once filters are configured, you can use the gear icon to collapse the Filter configuration area, as shown above. Check boxes and filter descriptions will remain visible in the area. Uncheck a check box to disable a filter. If you click the description text, simple controls will appear, allowing you to change the filter value.

Filtering by Dates

If the Filter Column selected is a **datetype** column, the interface presents different value controls:

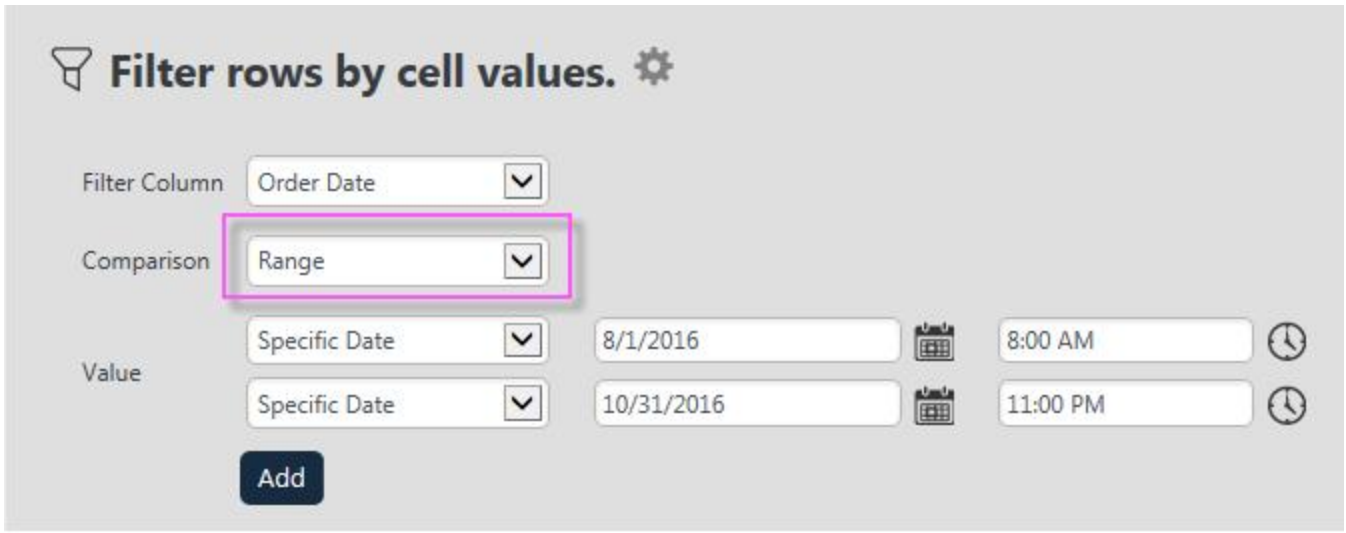


Specific Date

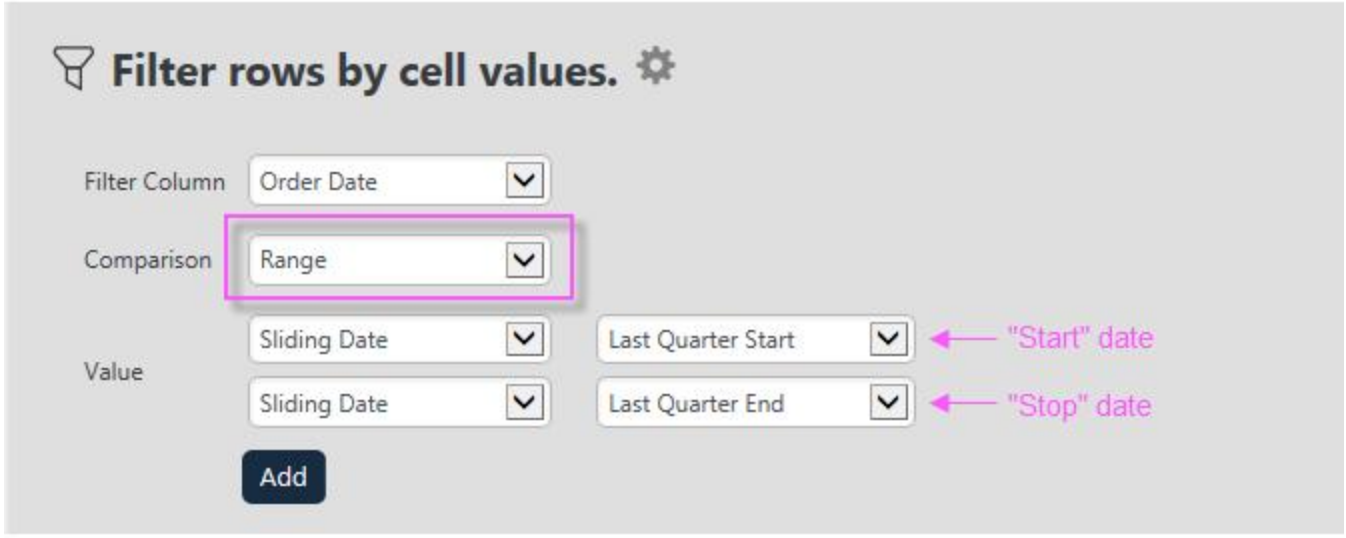


Sliding Date

You may choose to filter on a **Specific Date** and either type it in or select it from a pop-up calendar. Or, you can filter using a **Sliding Date** value and select from a long list of relative dates (*Last Week End, Last Month Start*, *90 Days Ago*, *Current Hour, Last Hour,* etc.)



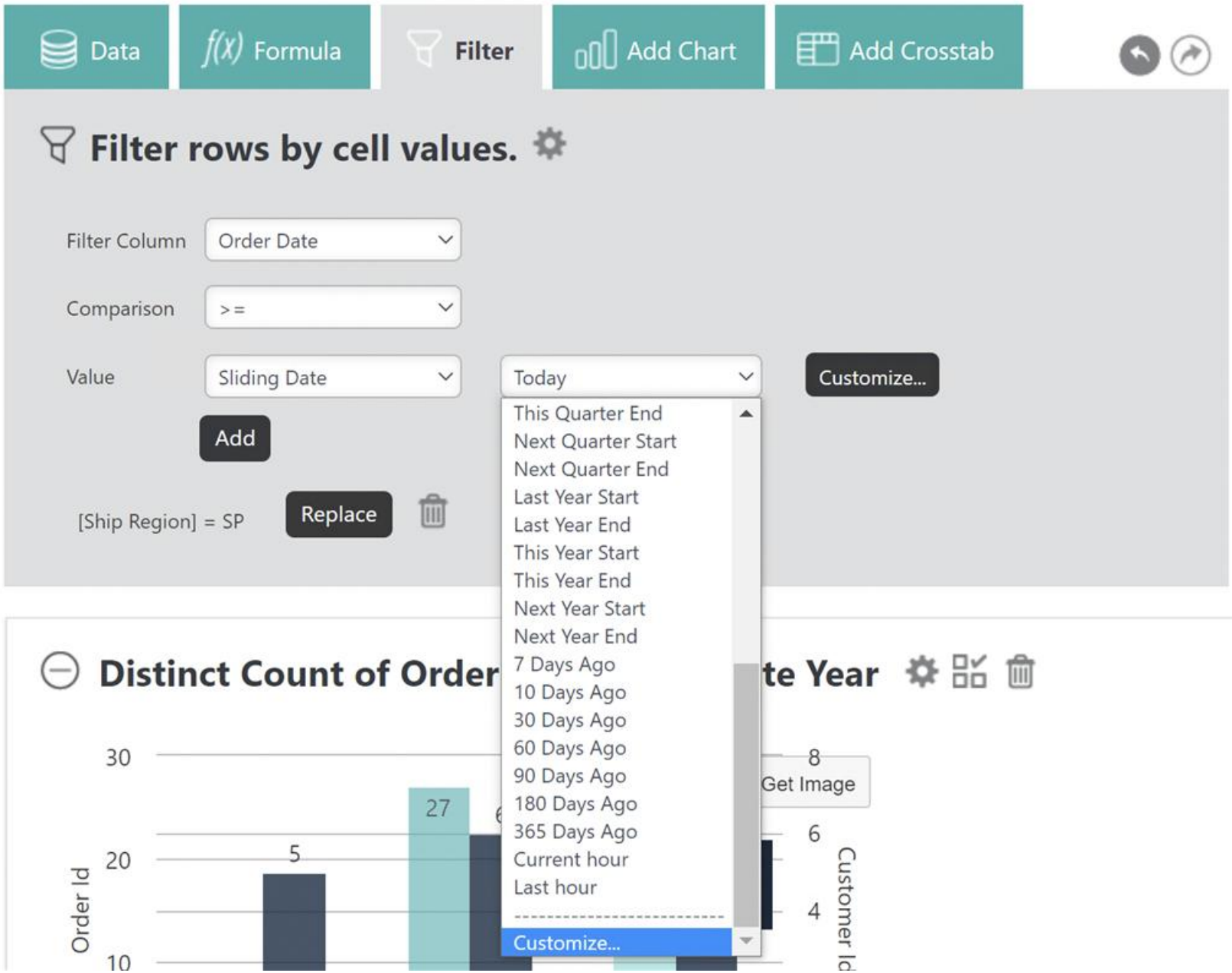
Specific Date Range



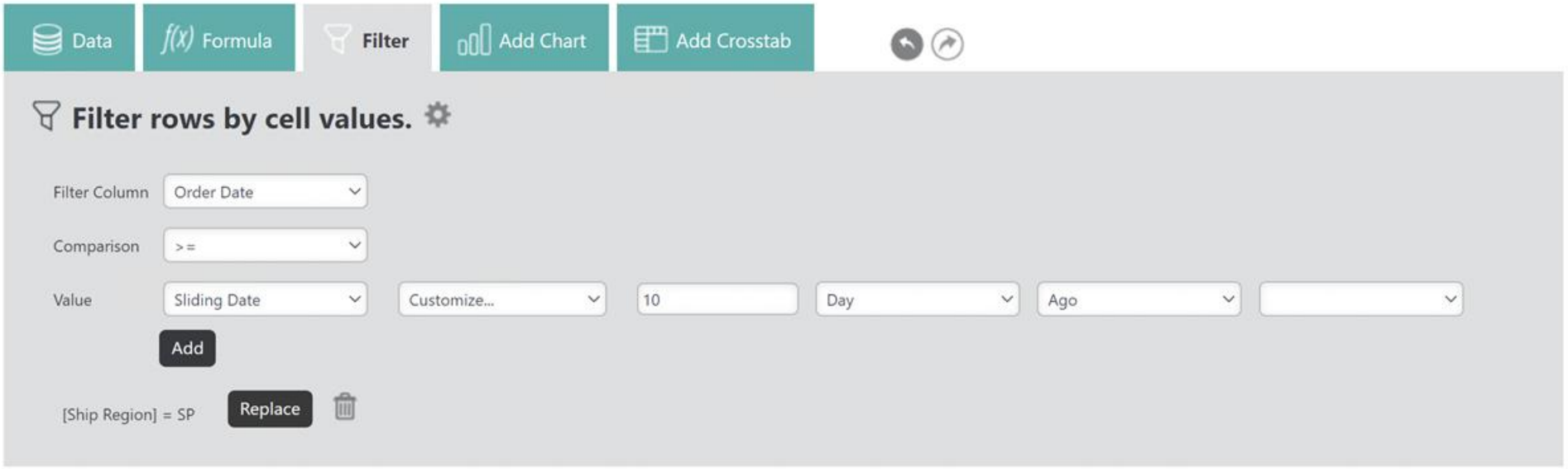
Sliding Date Range

If the Comparison option *Range* is chosen, as shown above, different value controls for Starting and Ending dates are displayed. These can be used in a variety of combinations.

Selecting the **Customize...** button, or choosing **Customize...** from the drop-down menu allows you to customize the time frame for your analysis. This option is only available for use with date/time-type data when Sliding Date is selected.

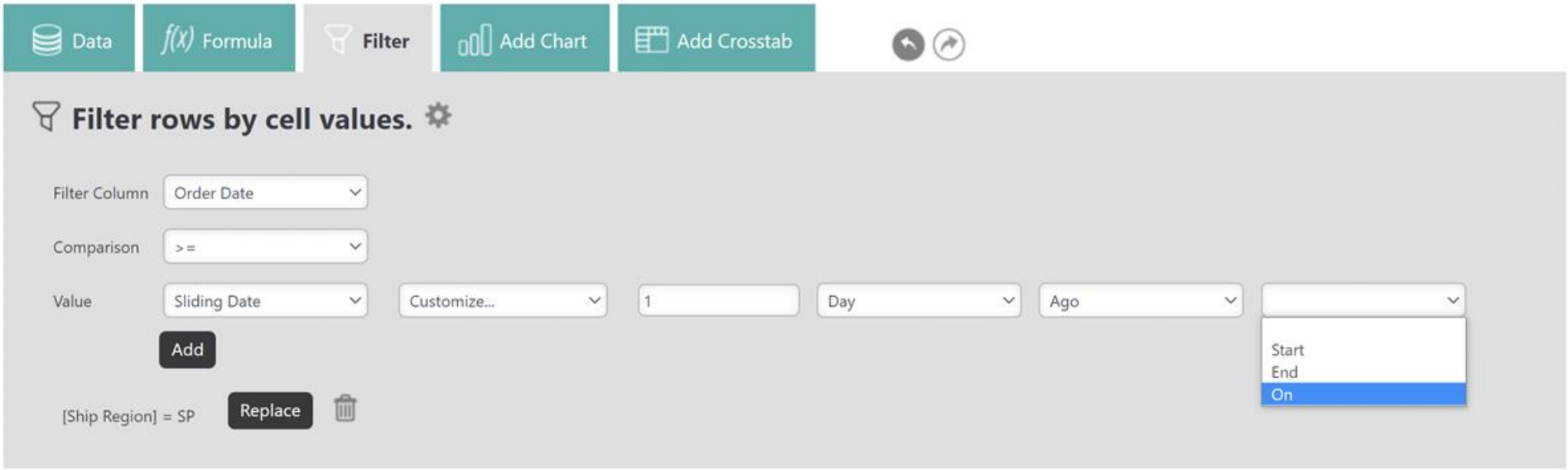

Data
Formula
Filter
Add Chart
Add Crosstab
Filter rows by cell values.
Filter Column
Order Date
Comparison
>=
Value
Sliding Date
Today
Customize...
Add
[Ship Region] = SP
Replace
This Quarter End
Next Quarter Start
Next Quarter End
Last Year Start
Last Year End
This Year Start
This Year End
Next Year Start
Next Year End
7 Days Ago
10 Days Ago
30 Days Ago
60 Days Ago
90 Days Ago
180 Days Ago
365 Days Ago
Current hour
Last hour
Customize...
Distinct Count of Order
te Year
Get Image
30
20
10
Order Id
5
27
8
6
4
Customer Id

Upon selecting Customize... additional fields generate, with default values, where you can manipulate the data:

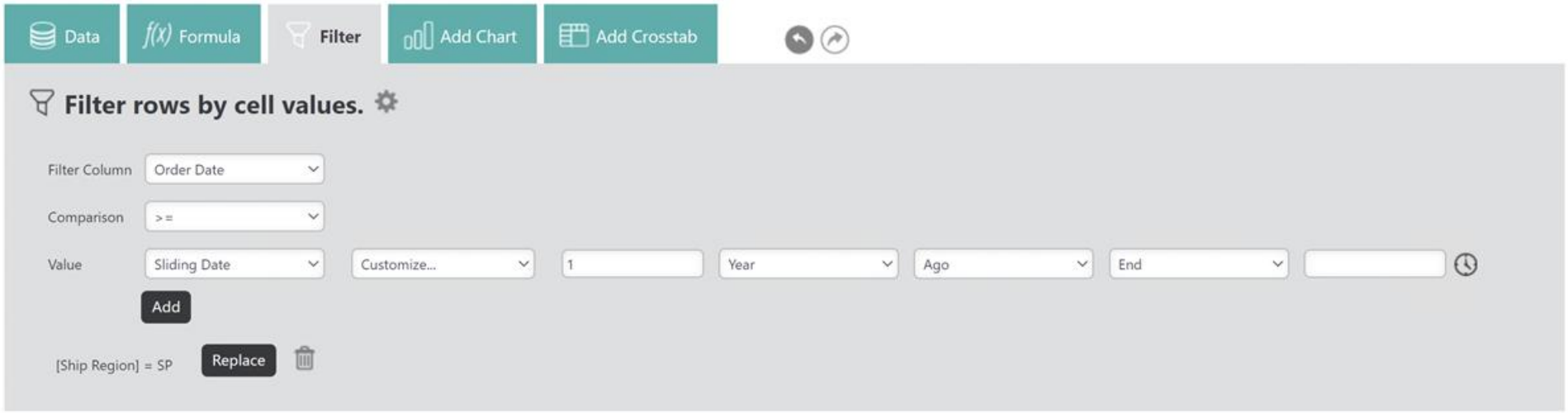


Usually *Start* and *End* is a range. You may need to change your Comparison value.

If you choose any value other than *Hour* in the drop-down list, the value *On* will be added to the Start/End drop-down list. The time picker only displays when Start, End, or On is chosen.



Specify a time by selecting the time picker:



To apply the filter, select **Replace** and **OK**:

The filtered results reflect the sliding date and specified time.

Click the **Filter** tab or button to hide the panel.

Filter by Aggregate

You can now apply filters on aggregates in your Analysis Grid. This feature is only supported on the following database versions:

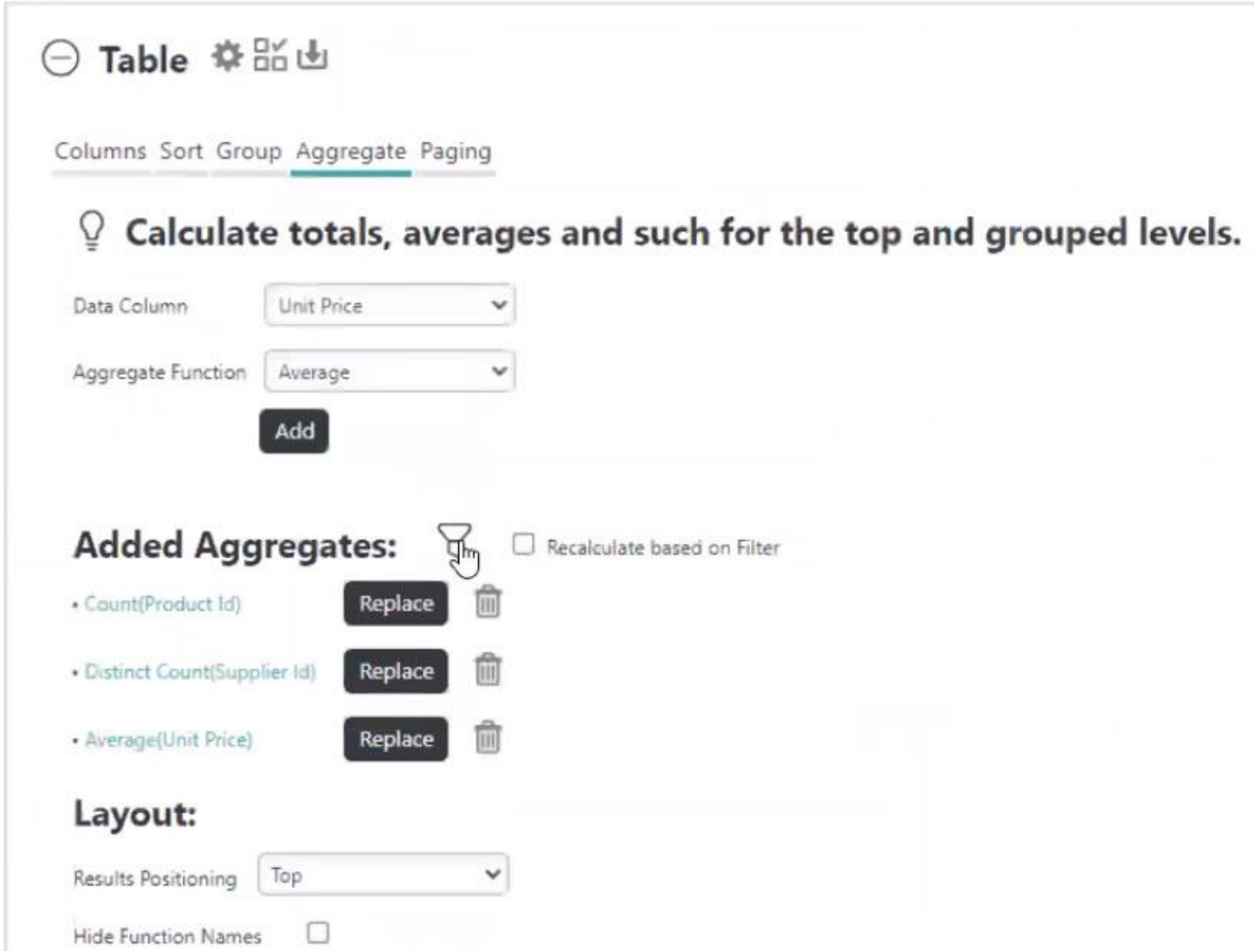
- MySQL > = 8
- SQLServer > = 2005
- Oracle > = 9
- PostegreSQL > = 8

In the example below we grouped the columns Reorder Level and Discontinued. Then, we applied an aggregate of *Count* on the Product Id column, *Discount* on the Supplier Id column, and *Average* on the Unit Price column:

Reorder Level	Discontinued	Product Id	Product Name	Supplier Id	Category Id	Quantity Per Unit	Unit Price	Units In Stock	Units On Order
		Count: 77		Distinct Count: 29			Average: 28.8663		
0		Count: 24		Distinct Count: 18			Average: 39.3404		
	False	Count: 16		Distinct Count: 14			Average: 35.3831		
		4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22	53	0
		8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40	6	0
		10	Ikura	4	8	12 - 200 ml jars	31	31	0
		12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38	86	0
		14	Tofu	6	7	40 - 100 g pkgs.	23.25	35	0
		18	Carnarvon Tigers	7	8	16 kg pkg.	62.5	42	0
		20	Sir Rodney's Marmalade	8	3	30 gift boxes	81	40	0
		26	Gumbär Gummibärchen	11	3	100 - 250 g bags	31.23	15	0
		46	Spegesild	21	8	4 - 450 g glasses	12	95	0
		47	Zaanse koeken	22	3	10 - 4 oz boxes	9.5	36	0
		59	Raclette Courdavault	28	4	5 kg pkg.	55	79	0
		60	Camembert Pierrot	28	4	15 - 300 g rounds	34	19	0
		62	Tarte au sucre	29	3	48 pies	49.3	17	0
		71	Flotemysost	15	4	10 - 500 g pkgs.	21.5	26	0
		72	Mozzarella di Giovanni	14	4	24 - 200 g pkgs.	34.8	14	0
		65	Louisiana Fiery Hot Pepper Sauce	2	2	32 - 8 oz bottles	21.05	76	0
	True	Count: 8		Distinct Count: 7			Average: 47.255		
		53	Perth Pasties	24	6	48 pieces	32.8	0	0
		42	Singaporean Hokkien Fried Mee	20	5	32 - 1 kg pkgs.	14	26	0

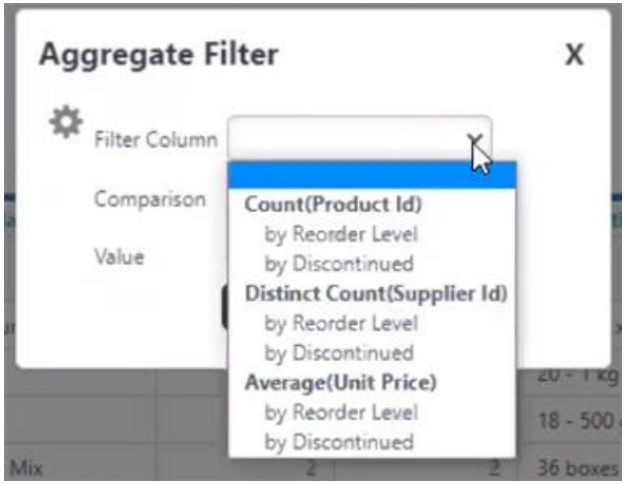
Narrow down the results of an aggregated column by applying a filter on the aggregate. In this example, we want to apply an aggregate filter that distinguishes any Count over 10.

1. To access the new feature, select the **Aggregate** tab and then select the **filter** icon:



The Aggregate Filter dialog box appears.

2. Select the **Filter Column** drop-down:



3. After you select the desired Filter Column, add a Comparison and enter a Value.

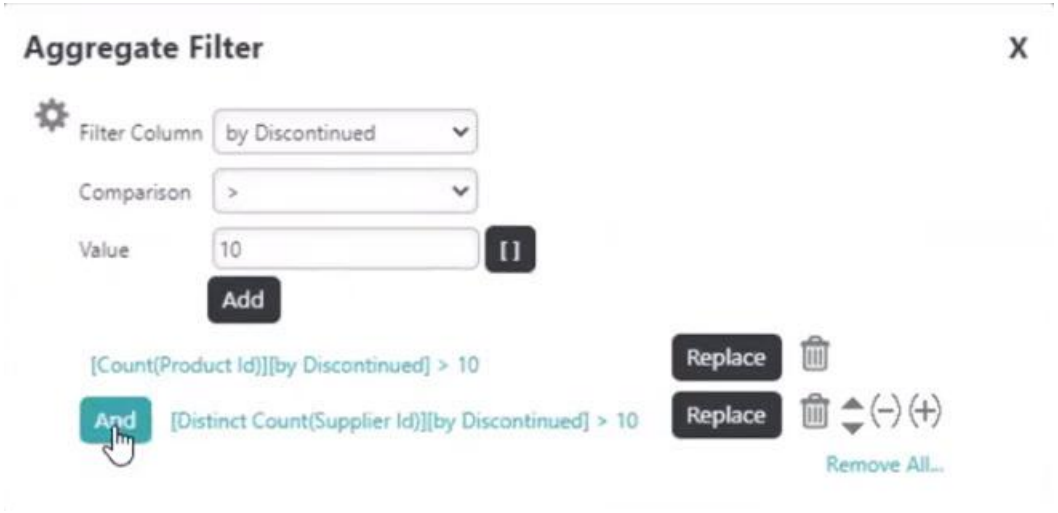
4. Then, select **Add**.

Now, our Analysis Grid displays Count values over 10 for the Product Id column:

Reorder Level	Discontinued	Product Id	Product Name	Supplier Id	Category Id	Quantity Per Unit	Unit Price	Units In Stock	Units On Order
		Count: 77		Distinct Count: 29			Average: 28.8663		
0		Count: 24		Distinct Count: 18			Average: 39.3404		
	False	Count: 16		Distinct Count: 14			Average: 35.3831		
		62	Tarte au sucre	29	3	48 pies	49.3	17	0
		59	Raclette Courdavault	28	4	5 kg pkg.	55	79	0
		60	Camembert Pierrot	28	4	15 - 300 g rounds	34	19	0
		47	Zaanse koeken	22	3	10 - 4 oz boxes	9.5	36	0
		46	Spegesild	21	8	4 - 450 g glasses	12	95	0
		71	Flotemysost	15	4	10 - 500 g pkgs.	21.5	26	0
		72	Mozzarella di Giovanni	14	4	24 - 200 g pkgs.	34.8	14	0
		26	Gumbär Gummibärchen	11	3	100 - 250 g bags	31.23	15	0
		20	Sir Rodney's Marmalade	8	3	30 gift boxes	81	40	0
		18	Carnarvon Tigers	7	8	16 kg pkg.	62.5	42	0
		14	Tofu	6	7	40 - 100 g pkgs.	23.25	35	0
		12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38	86	0
		10	Ikura	4	8	12 - 200 ml jars	31	31	0
		8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40	6	0
		4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22	53	0
		65	Louisiana Fiery Hot Pepper Sauce	2	2	32 - 8 oz bottles	21.05	76	0
25		Count: 12		Distinct Count: 10			Average: 20.3333		
	False	Count: 12		Distinct Count: 10			Average: 20.3333		
		61	Sirop d'érable	29	2	24 - 500 ml bottles	28.5	113	0
		48	Chocolade	22	3	10 pkgs.	12.75	15	70
		52	Filo Mix	24	5	16 - 2 kg boxes	7	38	0
		32	Mascarpone Fabioli	14	4	24 - 200 g pkgs.	32	9	40

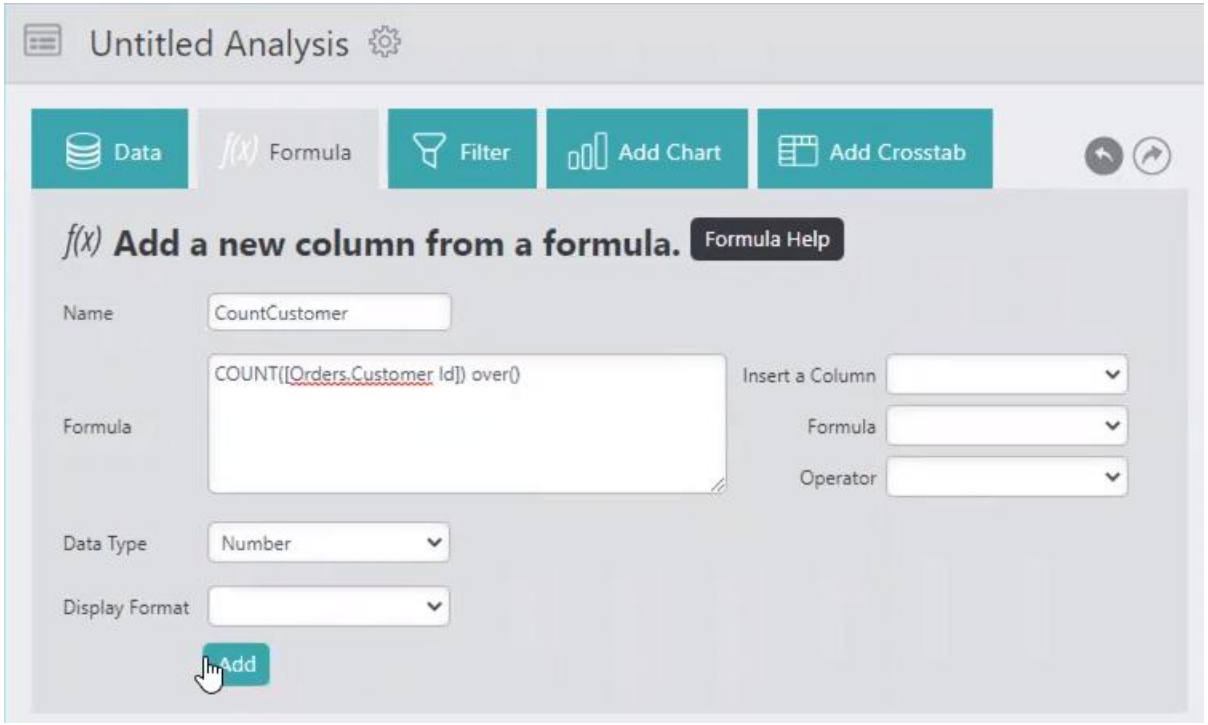
Repeat these steps to add aggregate filters to additional columns.

Currently, the relationship between the two filters is *And*. Select **And** to change the relationship to Or.

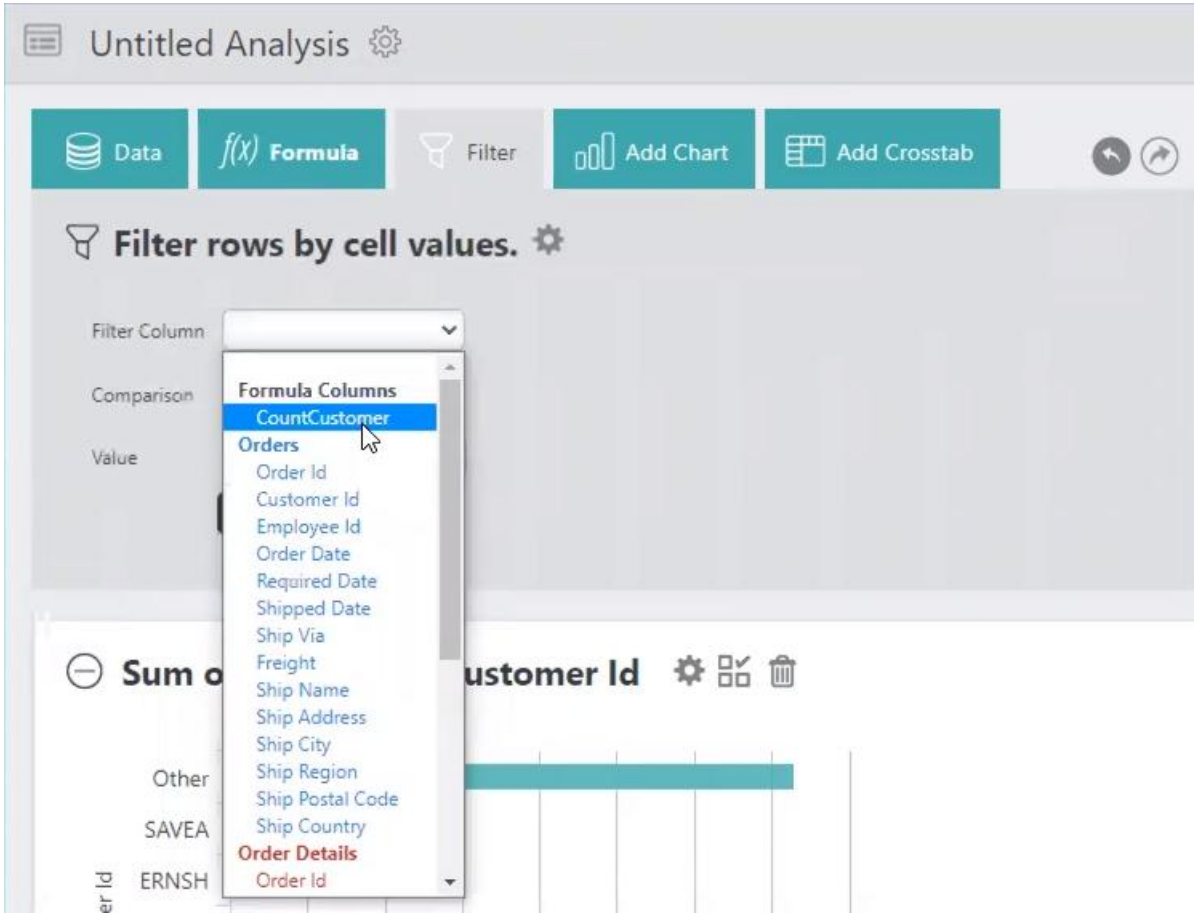


You can temporarily disable one (or both) of these filters by selecting the **gear** icon next to the Filer Column and selecting the corresponding check box:

You can also apply filters on the aggregates created through formula and it will populate as an option in the Filter Column drop-down menu. Below is an example of the CountCustomer formula we created using an aggregate:



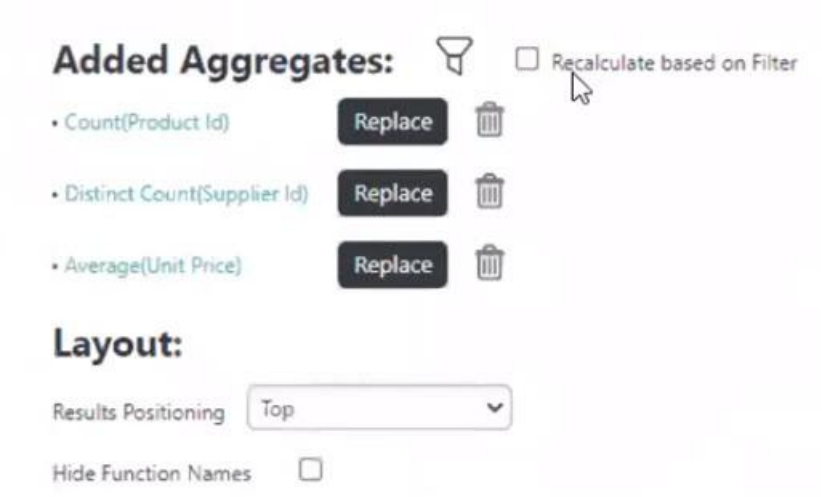
Once you select Add, the new formula will populate the Filter Column drop-down menu:



For more information about adding aggregate formulas, see "Analysis Grid - Adding Formula Columns" on page 79.

Recalculate Based on Filter

The Recalculate based on Filter check box is useful if you want to reset the column's overall aggregate value.



In this example, the Product Id column has an aggregate of Count that still reflects the overall Count (77) prior to adding the aggregate filter:

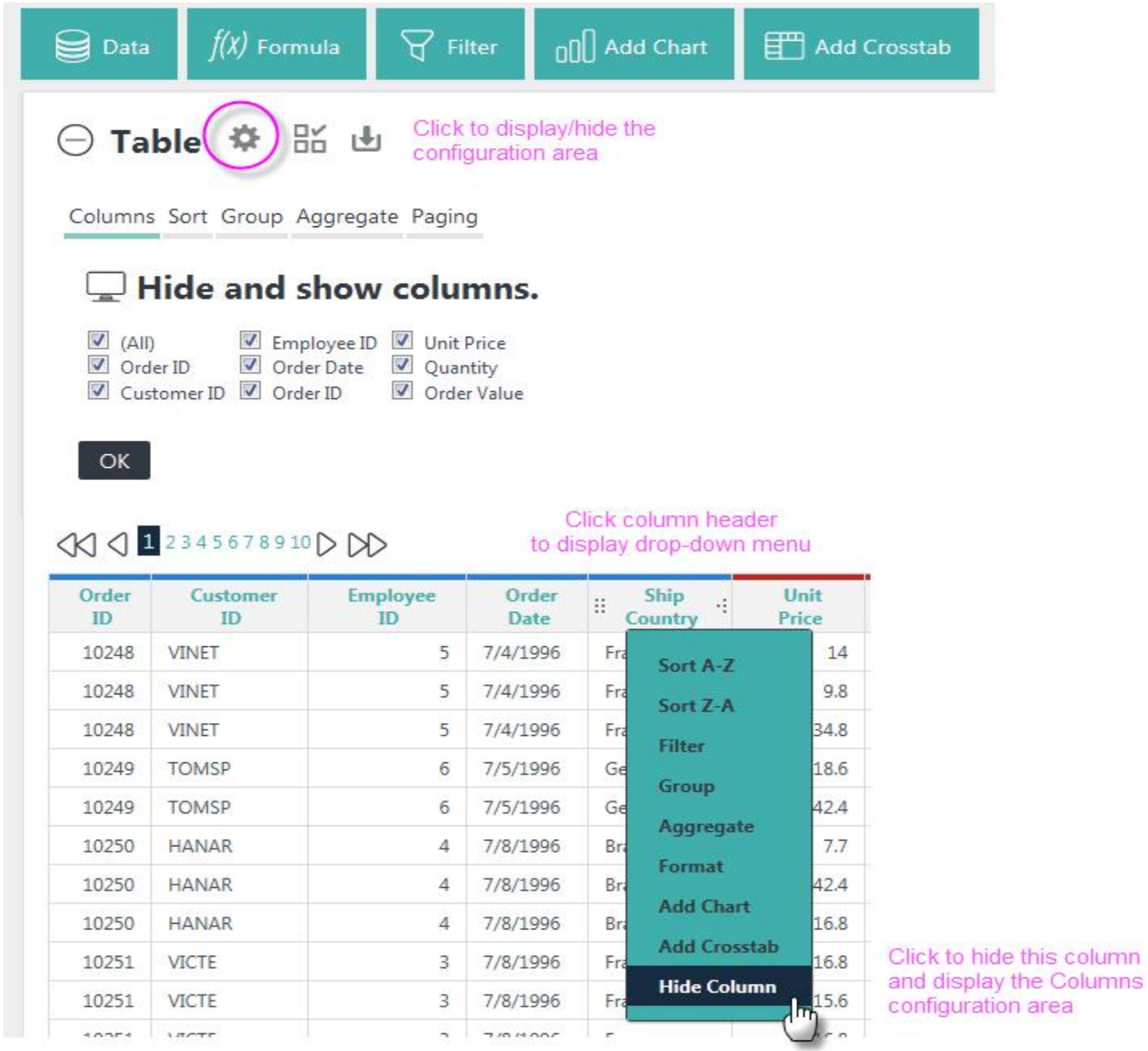
Reorder Level	Discontinued	Product Id	Product Name
		Count: 77	
0		Count: 24	
	False	Count: 16	
		62	Tarte au sucre
		59	Raclette Courdavault
		60	Camembert Pierrot
		47	Zaanse koeken
		46	Spegesild
		71	Flotemysost
		72	Mozzarella di Giovanni
		26	Gumbär Gummibärchen
		20	Sir Rodney's Marmalade
		18	Carnarvon Tigers
		14	Tofu
		12	Queso Manchego La Pastora
		10	Ikura
		8	Northwoods Cranberry Sauce
		4	Chef Anton's Cajun Seasoning
		65	Louisiana Fiery Hot Pepper Sauce
25		Count: 12	
	False	Count: 12	

After selecting the **Recalculate based on filter** check box, the new Count is 28:

Reorder Level	Discontinued	Product Id
		Count: 28
0		Count: 16
	False	Count: 16
		62
		59
		60
		47
		46
		71
		72
		20
		18
		14
		12
		10
		8
		4
		65
		26
25		Count: 12
	False	Count: 12

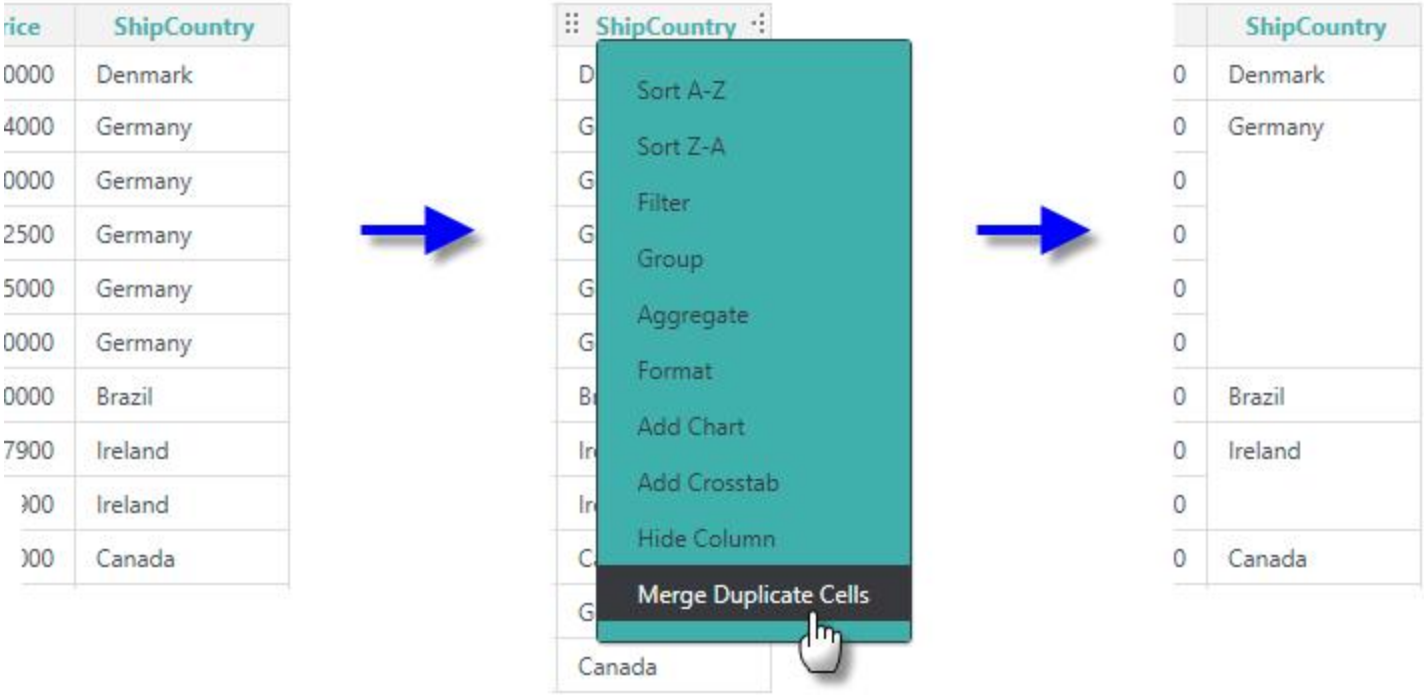
Analysis Grid - Showing, Hiding, and Moving Columns

You've selected which data columns to *include* in your working dataset but you may not want to *see* them all. The **Columns** feature in the Table configuration area, which can be displayed by clicking the **gear** icon or by clicking any **table column header**, controls column display:

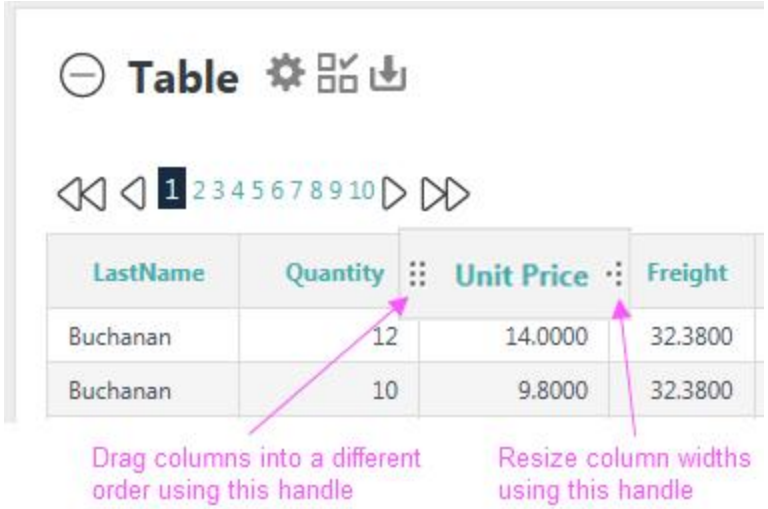

Data
Formula
Filter
Add Chart
Add Crosstab
Table
Click to display/hide the configuration area
Columns Sort Group Aggregate Paging
Hide and show columns.
(All)
Order ID
Customer ID
Employee ID
Order Date
Order ID
Unit Price
Quantity
Order Value
OK
1 2 3 4 5 6 7 8 9 10
Click column header to display drop-down menu
Order ID
Customer ID
Employee ID
Order Date
Ship Country
Unit Price
10248 VINET 5 7/4/1996 14
10248 VINET 5 7/4/1996 9.8
10248 VINET 5 7/4/1996 34.8
10249 TOMSP 6 7/5/1996 18.6
10249 TOMSP 6 7/5/1996 42.4
10250 HANAR 4 7/8/1996 7.7
10250 HANAR 4 7/8/1996 42.4
10250 HANAR 4 7/8/1996 16.8
10251 VICTE 3 7/8/1996 16.8
10251 VICTE 3 7/8/1996 15.6
Sort A-Z
Sort Z-A
Filter
Group
Aggregate
Format
Add Chart
Add Crosstab
Hide Column
Click to hide this column and display the Columns configuration area

As shown above, you can remove a column from the table by un-checking it. The **(All)** check box makes working with lots of columns easier, and you'll need to click **OK** to refresh the table with any changes. Click the **gear** icon again to hide the configuration area.

When columns contain multiple cells with the same data, you can visually group contiguous cells by *merging* them:



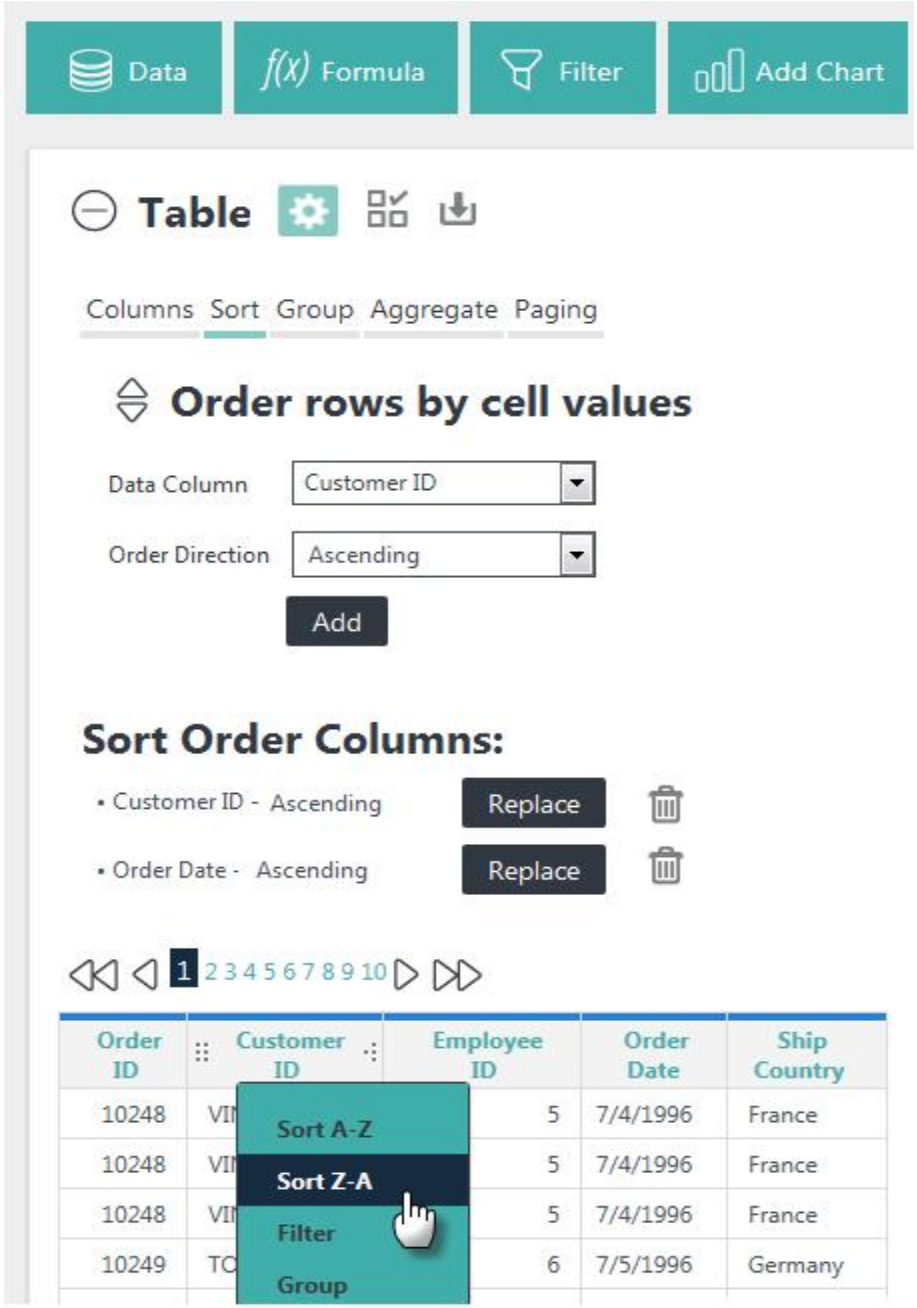
To do this, click the **column header** text and select **Merge Duplicate Cells** from the drop-down menu, as shown above. The row lines dividing duplicate cell values will be hidden. Use the same steps to reverse the process.



You can also rearrange the order, and change the widths, of table columns using two "drag handles" that appear when you hover your mouse over a column header, as shown above.

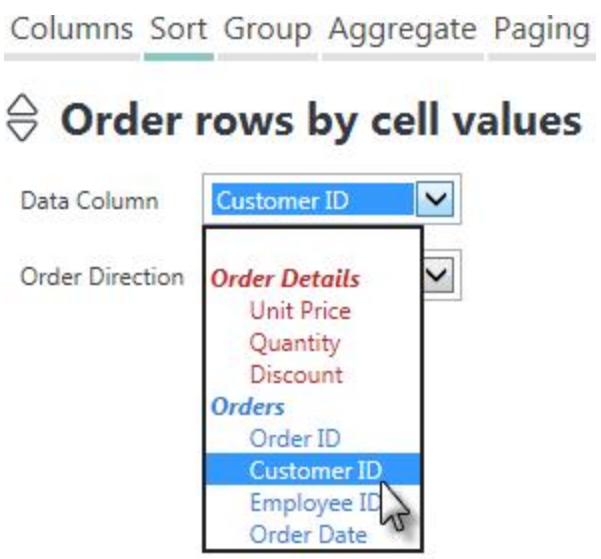
Analysis Grid - Sorting Rows

The Sort Rows feature allows you to set the *sort order* of the table column data. You can display the **Sort** configuration area by clicking the gear icon and then the Sort item, or by clicking any table column header and selecting a Sort option from the pop-up menu.


Data
Formula
Filter
Add Chart
Table
Columns Sort Group Aggregate Paging
Order rows by cell values
1
Data Column
Customer ID
2
Order Direction
Ascending
Add
Sort Order Columns:
3
• Customer ID - Ascending
Replace
• Order Date - Ascending
Replace
1 2 3 4 5 6 7 8 9 10
4
Order ID
Customer ID
Employee ID
Order Date
Ship Country
10248
5
7/4/1996
France
10248
5
7/4/1996
France
10248
5
7/4/1996
France
10249
6
7/5/1996
Germany
Sort A-Z
Sort Z-A
Filter
Group

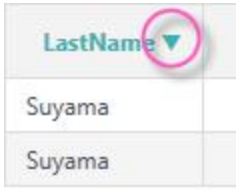
Here's how to use this feature.

1. Select a **data column** to sort on:

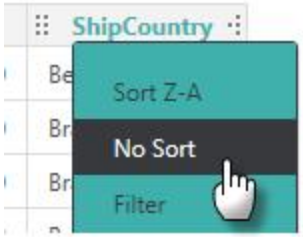


   You'll see that the available columns are grouped and color-coded to make it easier for you to identify them. If you created any Formula Columns, they'll be in there, too.
2. Select a sorting **orderdirection** - ascending or descending. Click **Add** to add the sort and refresh the table. The table will immediately be updated with the new sort. Repeat as desired for additional sorting.
3. As Sort Order columns are created, they're added to the list of sorts. Use the adjacent **Replace** button and **Remove** (trash can) icon to manage the list.
4. You can also sort a column directly,by clicking its column header and selecting the sort in the menu that appears.

Click the gear icon to hide the configuration area. The table rows will be re-displayed in the sort order created.



If your application has been configured for it, an arrow, shown circled above, will appear beside the column header text to indicate that a sort is in effect and to show its order/direction.



Once a column has been sorted, the column header menu also allows you to change the order orremove the sorting, as shown above.

Analysis Grid - Grouping Rows

The **Group** feature in the Table configuration area, which can be displayed by clicking the **gear** icon or by clicking any **table column header**, lets you group table rows:

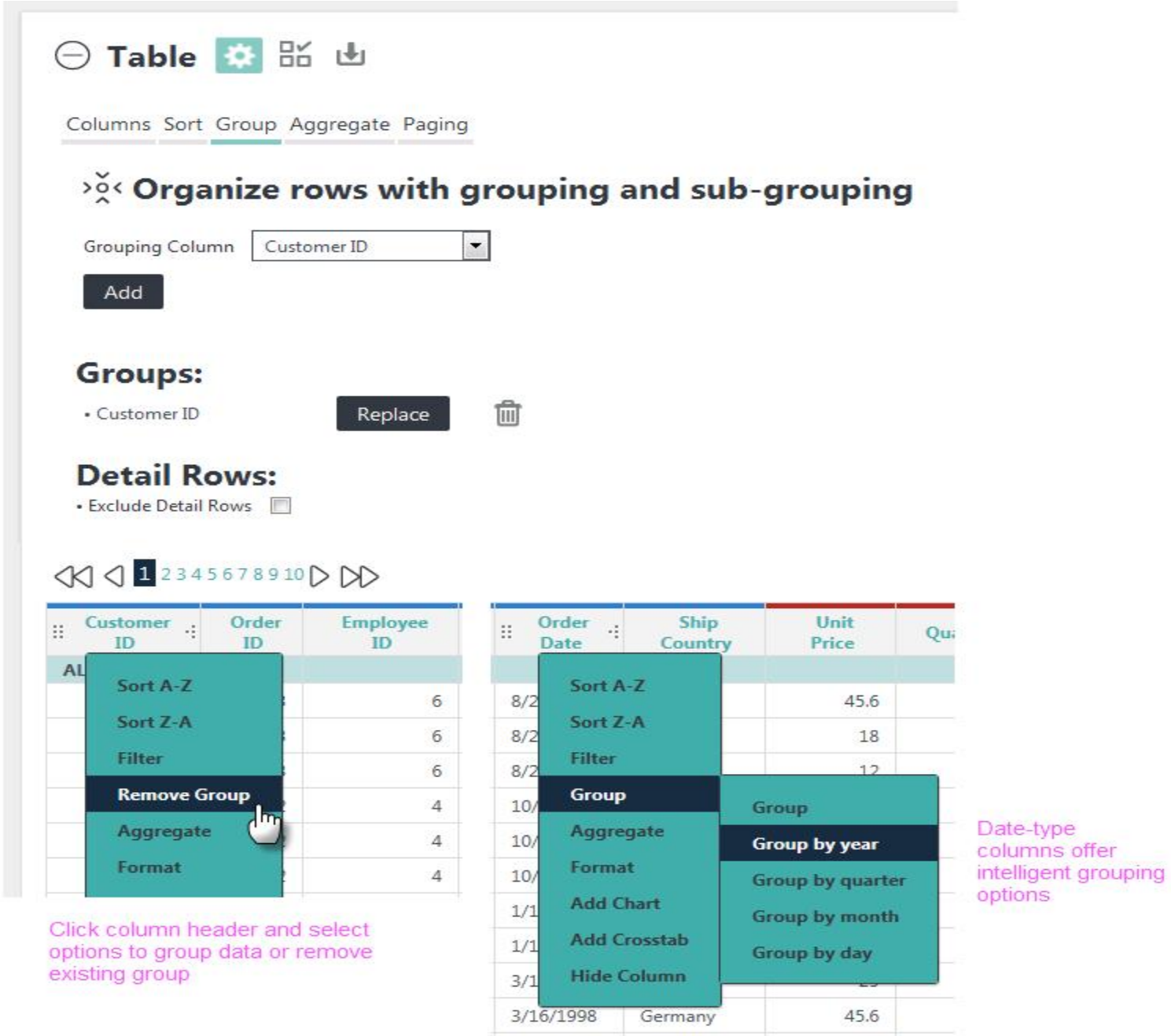


Table
Columns Sort Group Aggregate Paging
Organize rows with grouping and sub-grouping
1
Grouping Column
Customer ID
Add
Groups:
2
• Customer ID
Replace
Detail Rows:
3
• Exclude Detail Rows
1 2 3 4 5 6 7 8 9 10
4
Customer ID
Order ID
Employee ID
Sort A-Z
Sort Z-A
Filter
Remove Group
Aggregate
Format
Order Date
Ship Country
Unit Price
Group
Add Chart
Add Crosstab
Hide Column
Group by year
Group by quarter
Group by month
Group by day
3/16/1998
Germany
45.6
Click column header and select options to group data or remove existing group
Date-type columns offer intelligent grouping options

Here's how to use this feature:

1. In the configuration area, select the **Grouping Column** for the first level of grouping from the list of columns. Depending in the column's data type, additional input controls may be displayed. Click **Add** to group the data and refresh the table. Repeat as desired to create sub-groups.
2. As groups and sub-groups are created, they're added to the Groups list. Use the adjacent **Remove** button and **Replace** (trash can) icon to manage the list.
3. The **Exclude Detail Rows** check box can be used to hide the rows that have been grouped, "collapsing" each group into a single row in the table.
4. Grouping and un-grouping can be also be accomplished by clicking a column header in the table and then selecting the desired options from the context menu.

Click the **gear** icon to hide the configuration area.

Analysis Grid - Creating Aggregations

The Aggregate feature enables you to control aggregation functions in your Analysis Grid. You can combine standard calculations or create custom aggregations and save them to use more than once.

For information on how to customize an aggregation, see "Analysis Grid - Creating Custom Aggregations" on page 129.

Access this feature by selecting the **gear** icon or selecting any **table column header** when viewing a table:

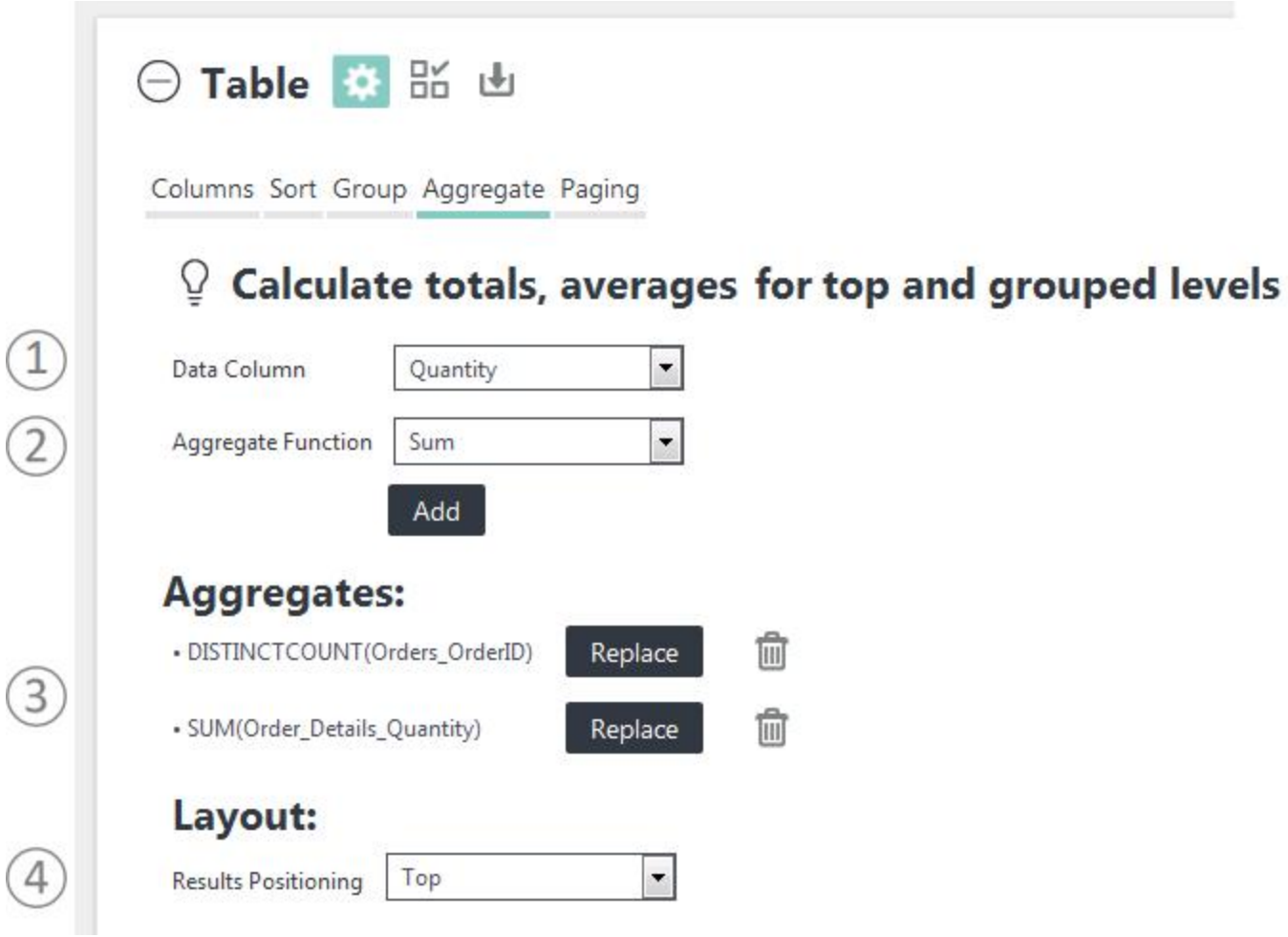


Table
Columns Sort Group Aggregate Paging
Calculate totals, averages for top and grouped levels
1
Data Column
Quantity
2
Aggregate Function
Sum
Add
Aggregates:
• DISTINCTCOUNT(Orders_OrderID)
Replace
3
• SUM(Order_Details_Quantity)
Replace
Layout:
4
Results Positioning
Top

Here's how to use standard aggregations:

1. Select the **Data Column** you want to be part of your aggregation.
2. Select the **Aggregate Function**. Options include: *Sum*, *Average*, *Standard Deviation*, *Count*, *Distinct Count*, *Minimum*, *Maximum*, and any *Custom* aggregations you may have.
3. Select **Add**. Logi Info aggregates the data and refreshes the table. Info matches the formatting of the aggregation to that of the column.
4. Logi Info adds aggregates to the Aggregate list as they are created. Select the adjacent **Replace** button or **trash can** icon to manage the list.
5. You can specify that aggregate results appear in an extra table row at the top or bottom of the table, as shown above. If *Grouping* is in effect, aggregate values will also appear at each grouping level in the table.
6. Select the **gear** icon to hide the configuration area.

By default, columns with Null values are *excluded* from aggregations. However, you can configure your Analysis Grid to include them; check with your developer for details.

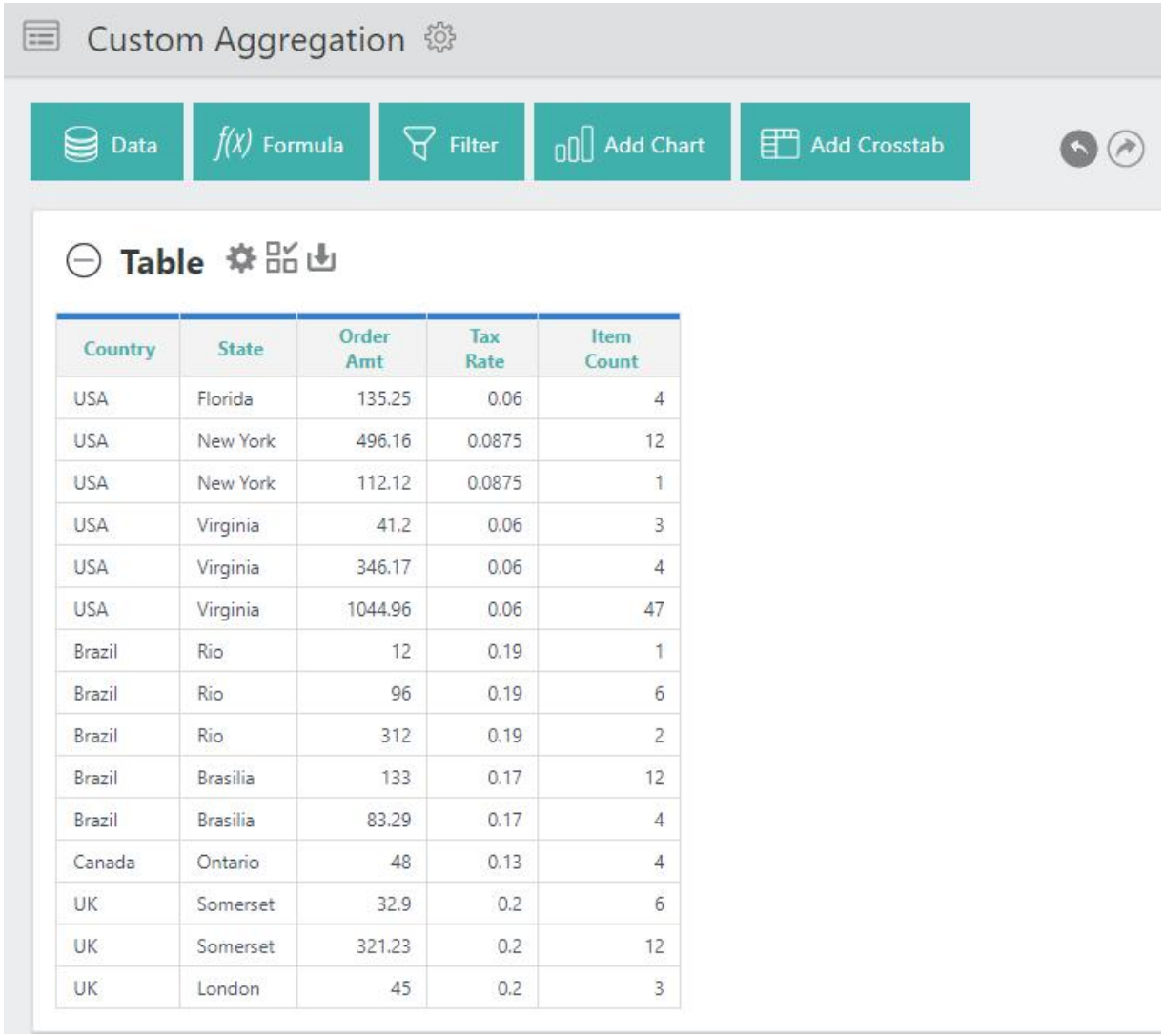
For information about filtering by aggregate, see "Analysis Grid - Filtering Rows" on page 86.

Analysis Grid - Creating Custom Aggregations

In addition to the standard aggregations, you can create custom aggregations and save them to use again in the future. There are two different types of custom aggregates you can create in the Analysis Grid: column-specific and data type-specific.

Column-Specific Custom Aggregations

For this example, we want to determine the *Average Item Cost*. Using the data provided in the table below, you'll notice we need to incorporate a Tax Rate. This additional data component indicates that we cannot find the average by simply using a standard Sum or Average aggregation. Instead, we're going to create a Custom Aggregation to calculate the Average Item Cost.



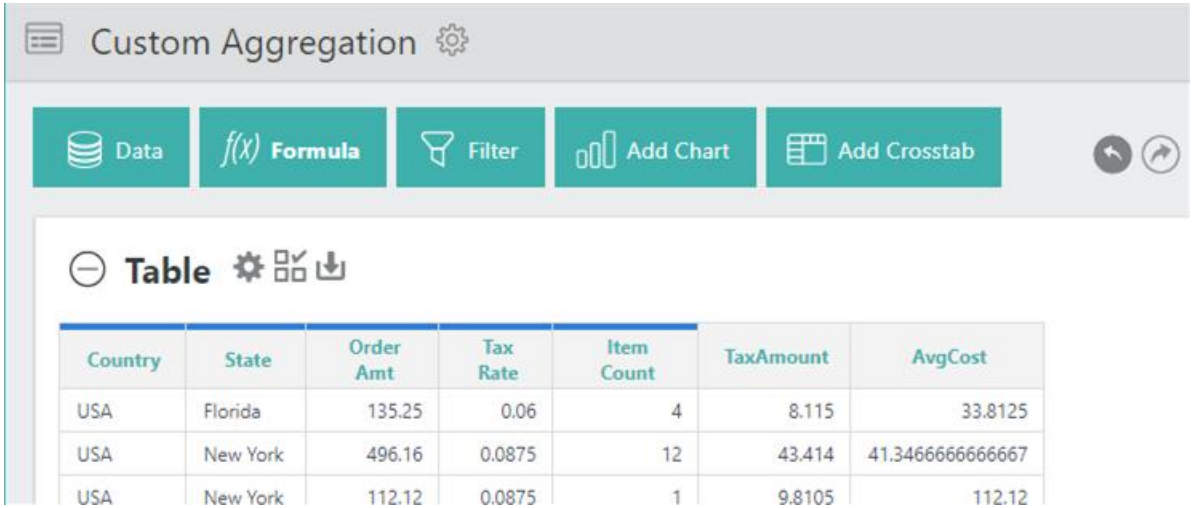
Country	State	Order Amt	Tax Rate	Item Count
USA	Florida	135.25	0.06	4
USA	New York	496.16	0.0875	12
USA	New York	112.12	0.0875	1
USA	Virginia	41.2	0.06	3
USA	Virginia	346.17	0.06	4
USA	Virginia	1044.96	0.06	47
Brazil	Rio	12	0.19	1
Brazil	Rio	96	0.19	6
Brazil	Rio	312	0.19	2
Brazil	Brasilia	133	0.17	12
Brazil	Brasilia	83.29	0.17	4
Canada	Ontario	48	0.13	4
UK	Somerset	32.9	0.2	6
UK	Somerset	321.23	0.2	12
UK	London	45	0.2	3

To calculate this number, we need to first add a formula for Tax Amount (Order Amount * Tax Rate). Additionally, we need a formula that calculates the Average Item Cost (Order Amount/Item Count). These formulas will automatically display in your Data Table as additional columns. For more information about creating formulas in your Analysis Grid, see "Analysis Grid - Adding Formula Columns" on page 79.

If you're going to aggregate a Formula column (created by executing a calculation), the "order of operations" may be important. For example, should the Analysis Grid do the calculation first, then apply the aggregation, or apply the aggregation and then do the calculation? The choice can result in completely different results.

Once you've created your formulas, follow the steps below to create a custom aggregate:

1. Select the **gear** icon to access the Aggregation feature:

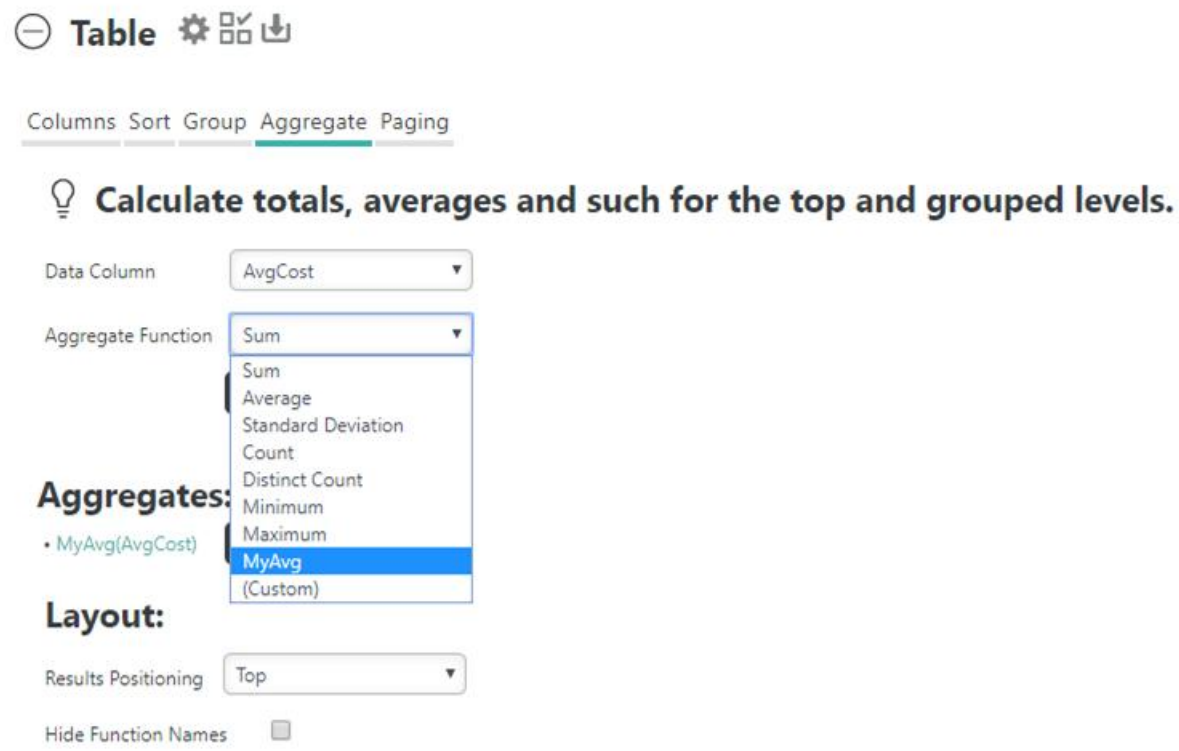


Country	State	Order Amt	Tax Rate	Item Count	TaxAmount	AvgCost
USA	Florida	135.25	0.06	4	8.115	33.8125
USA	New York	496.16	0.0875	12	43.414	41.3466666666667
USA	New York	112.12	0.0875	1	9.8105	112.12

Info displays the Table configuration options.

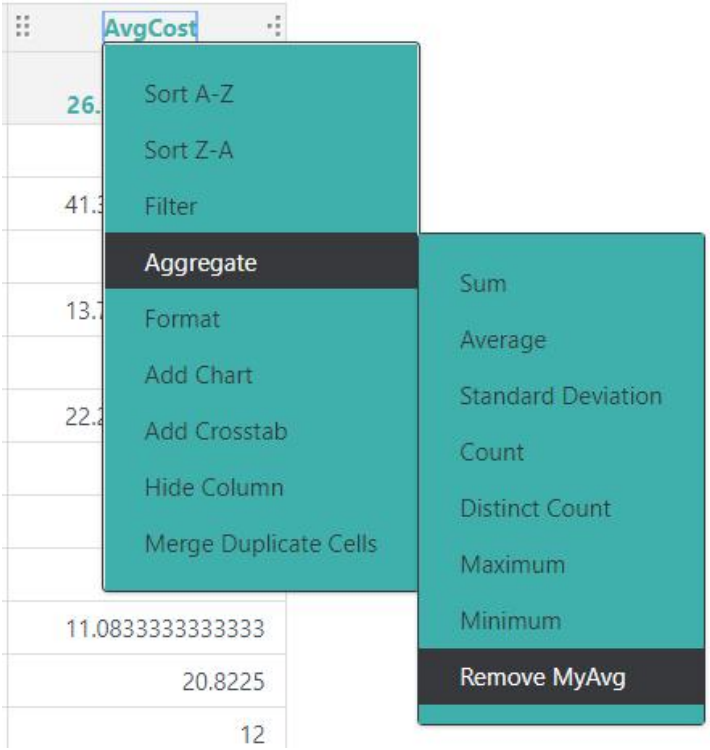
2. Select the **Aggregate** tab.
3. To add a Custom aggregation, select the **Data Column** to be aggregated from the column list.
4. Then, select **Aggregate Function** > **Custom**. Info displays the Add a custom aggregation for (name of Data Column) pop-up window:

5. Type a name for the Custom Aggregation.
6. Customize the formula by utilizing the drop-down arrows to select a **Column**, **Formula**, and **Operator**.
7. Select **Add**. Logi Info displays your custom aggregation in the Data Table and adds your new custom Aggregate Function to the drop-down list for future use:



Country	State	Order Amt	Tax Rate	Item Count	TaxAmount	AvgCost
						MyAvg: 26.9361983471074
USA	Florida	135.25	0.06	4	8.115	33.8125
USA	New York	496.16	0.0875	12	43.414	41.3466666666667

8. You can turn the custom aggregation on/off by selecting the **column header** > **Aggregations** > **Remove** (Custom Agg):



9. To edit your Custom Aggregation, select the **Aggregation hyperlink** in the Aggregates section:

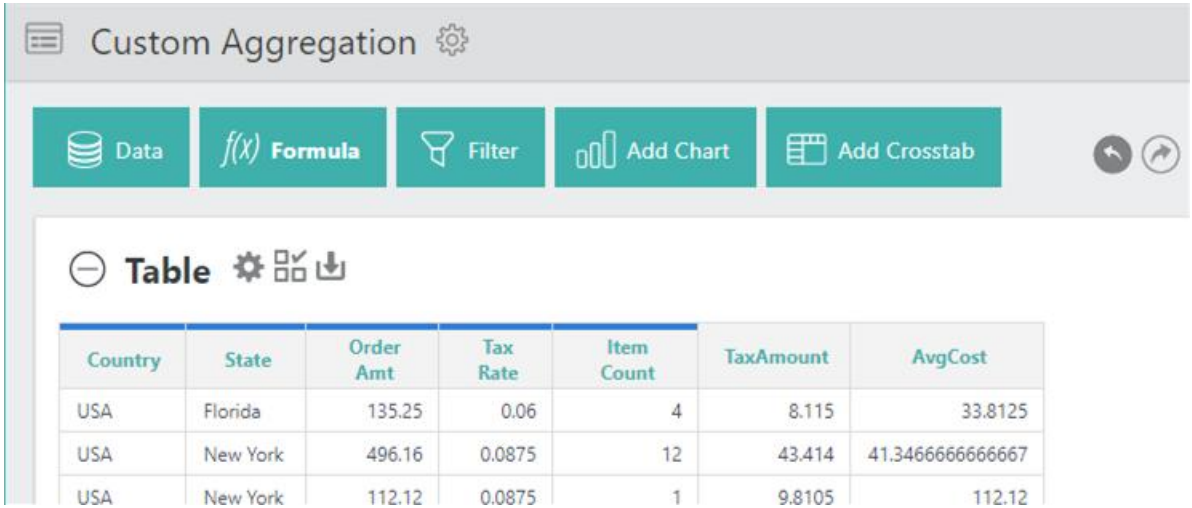
Info displays the Add a custom aggregation pop-up window.

10. Manage the list of custom aggregates by selecting the adjacent **Replace** button or **trash can** icon.

Data Type-Specific Custom Aggregation

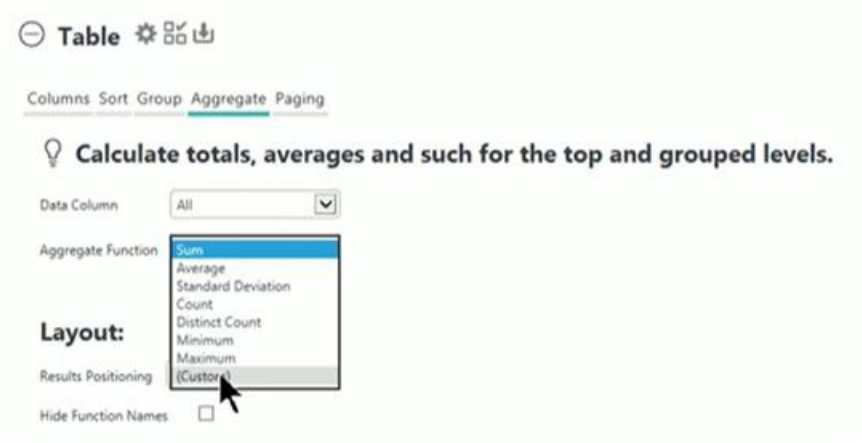
The other type of custom aggregate you can create in your Analysis Grid are applied to a specific data type(s).

1. Begin by selecting the **gear** icon to access the Aggregation feature:

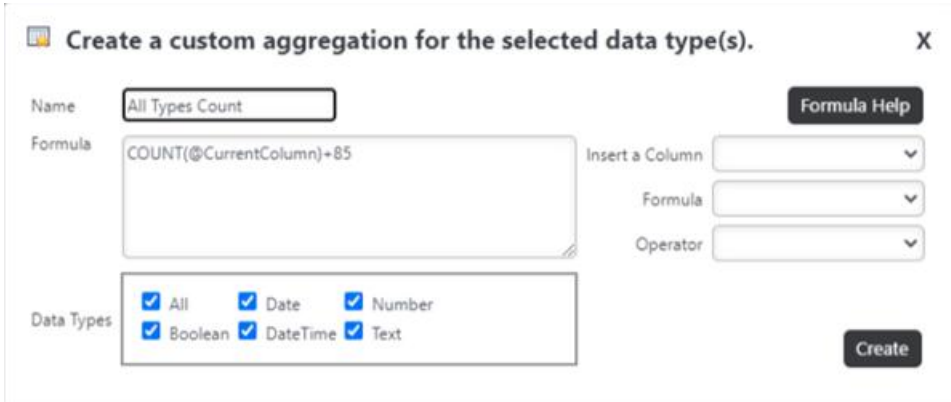


Info displays the Table configuration options.

2. Select the **Aggregate** tab.
3. Select **All** from the Data Column drop-down list.
4. Then, select **Aggregate Function** > **(Custom)**. Logi Info displays the Add a custom aggregation for the selected data type (s) pop-up window:

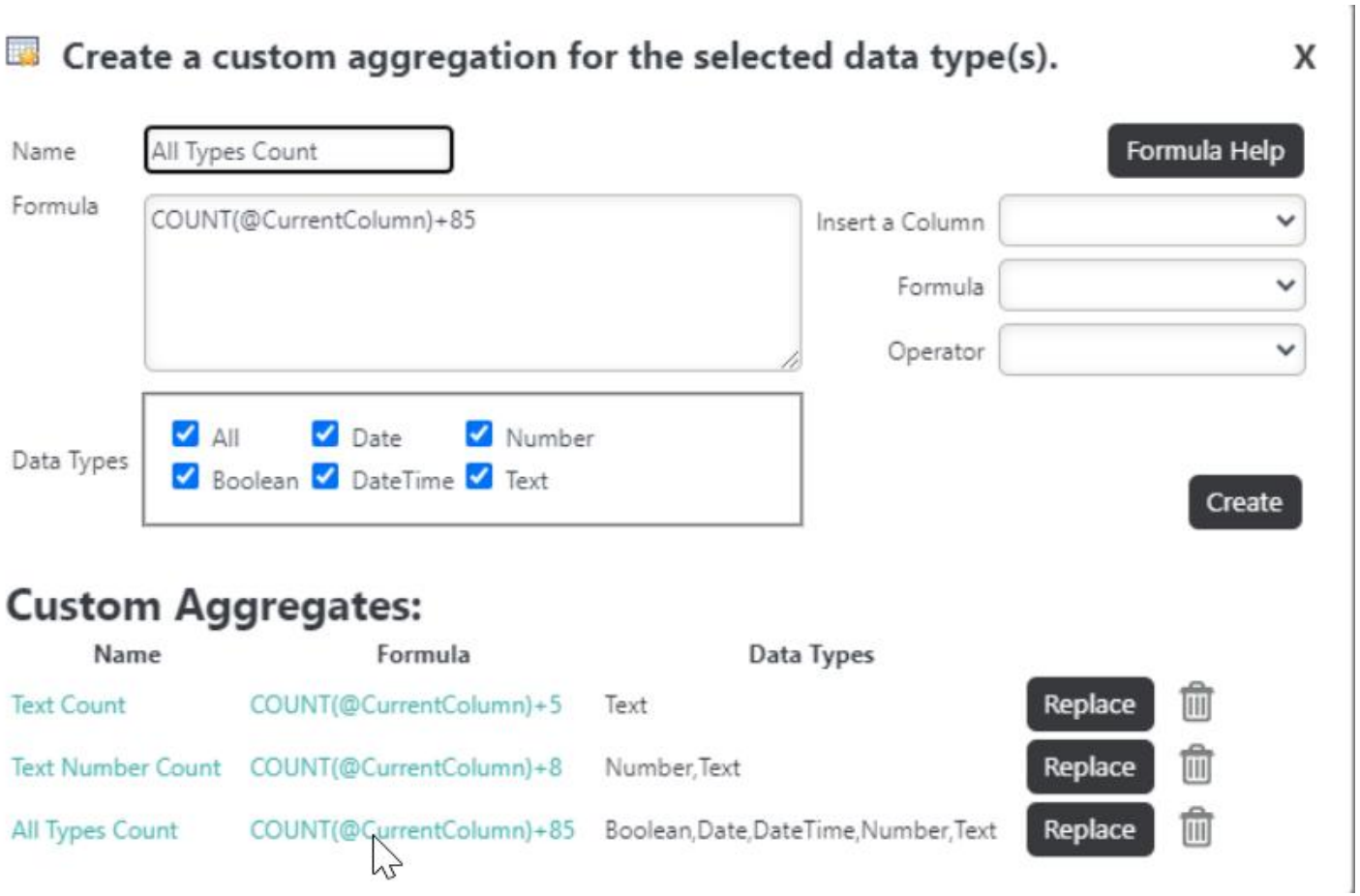


5. Type a name for the Custom Aggregation.
6. Customize the formula by entering your formula, or, utilize the drop-down arrows to select a **Column**, **Formula**, and **Operator**. Data type-specific Custom Aggregates use a @CurrentColumn token as a column reference and replace the column on which it's applied, as shown below. Reminder that you cannot apply a sum or count aggregate on a variable character field or text column.
7. Select the **All** Data Type check box, or select individual data type(s) to apply the custom aggregate:



You must select at least one check box to create the aggregate. Selecting all ensures that the custom aggregate is applied to all of the columns.

8. Select **Create**. Info displays your custom aggregation in the Data Table and adds your new custom Aggregate Function to the drop-down list for future use.
9. You can turn the custom aggregation on/off by selecting the **column header** > **Aggregations** > **Remove** (Custom Agg).
10. To edit your Custom Aggregation, select the Aggregation hyperlink in the Aggregates section:

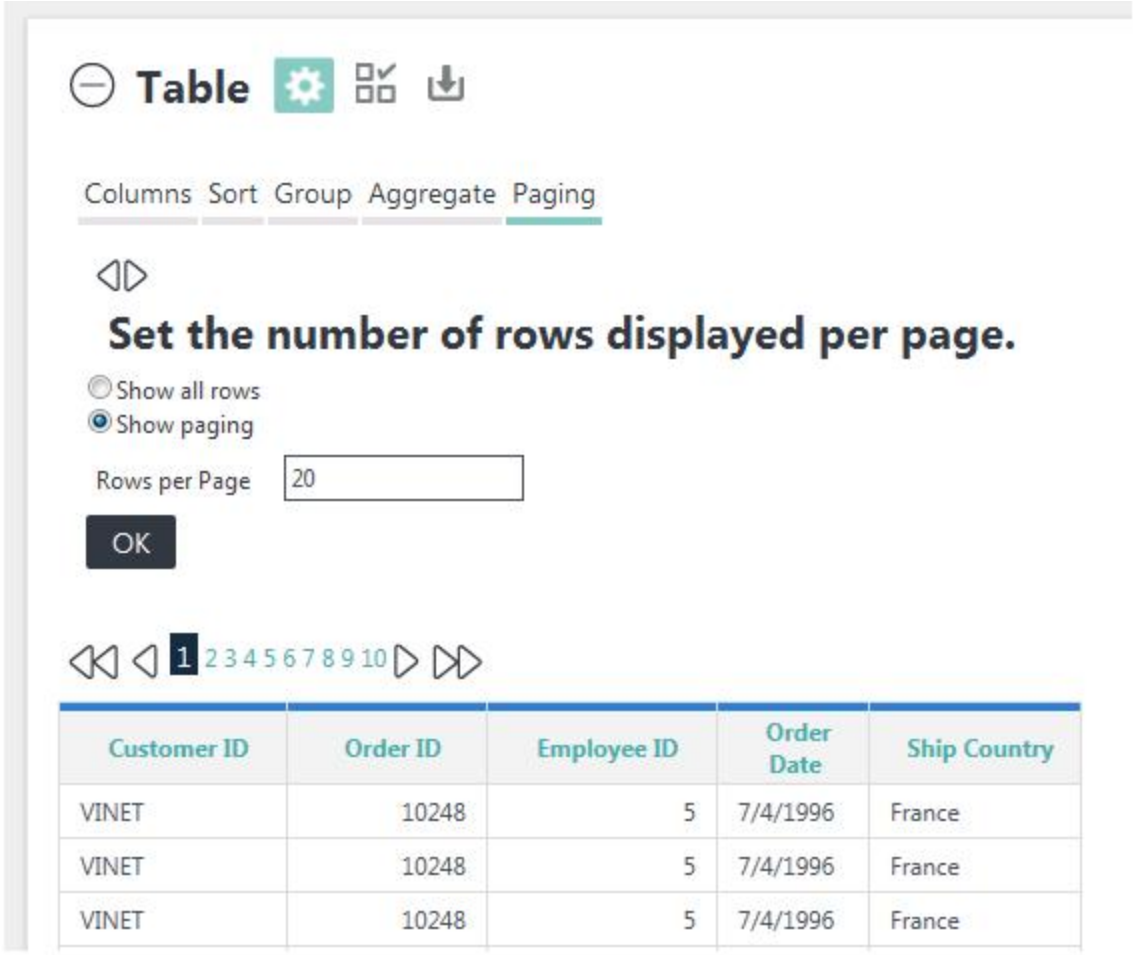


Info displays the Add a custom aggregation pop-up window.

11. Manage the list of custom aggregates by selecting the adjacent **Replace** button or **trash can** icon.

Analysis Grid - Controlling Paging

Click the table's **gear** icon and then the **Paging** option item to control the pagination of Analysis Grid tables:



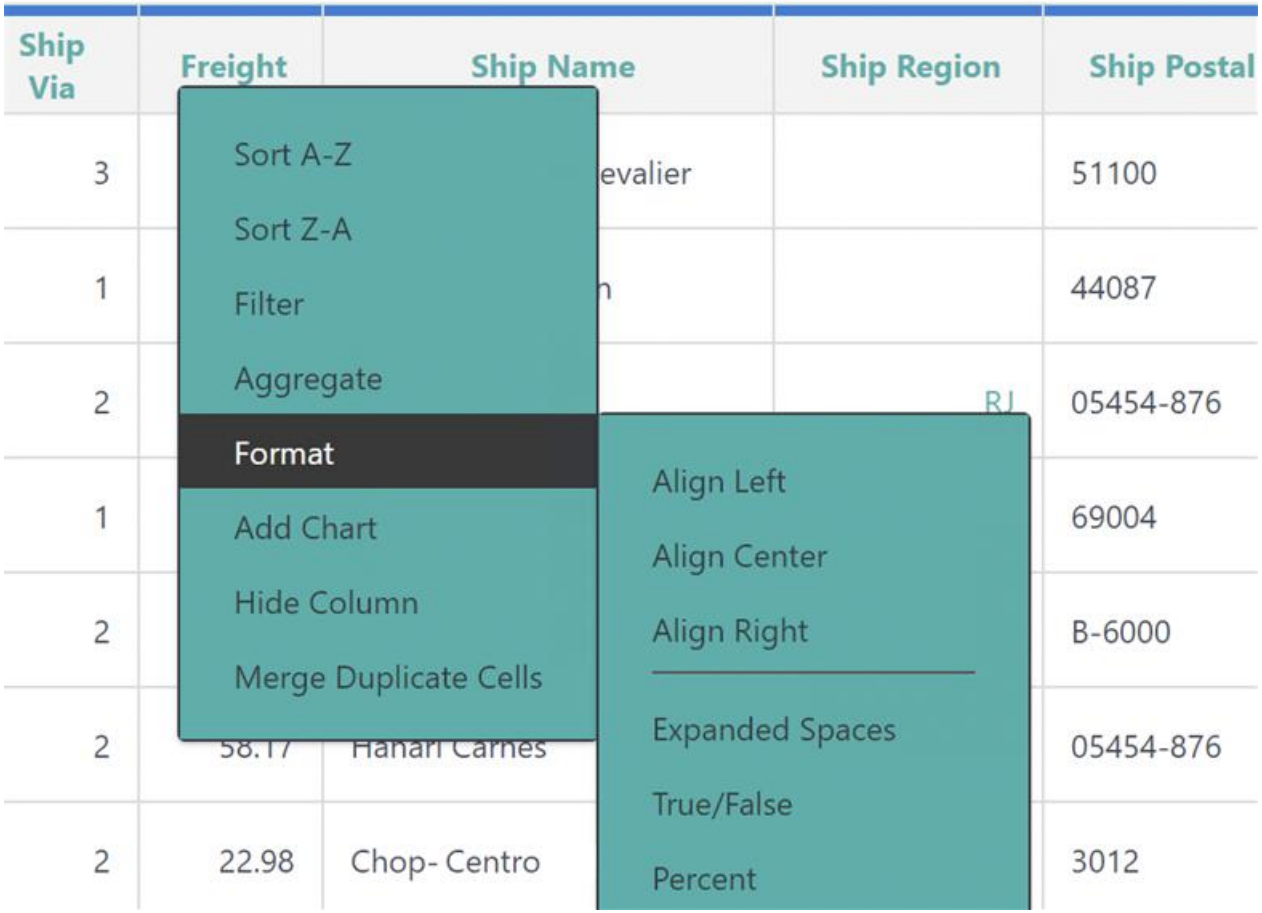
Here's what you need to do:

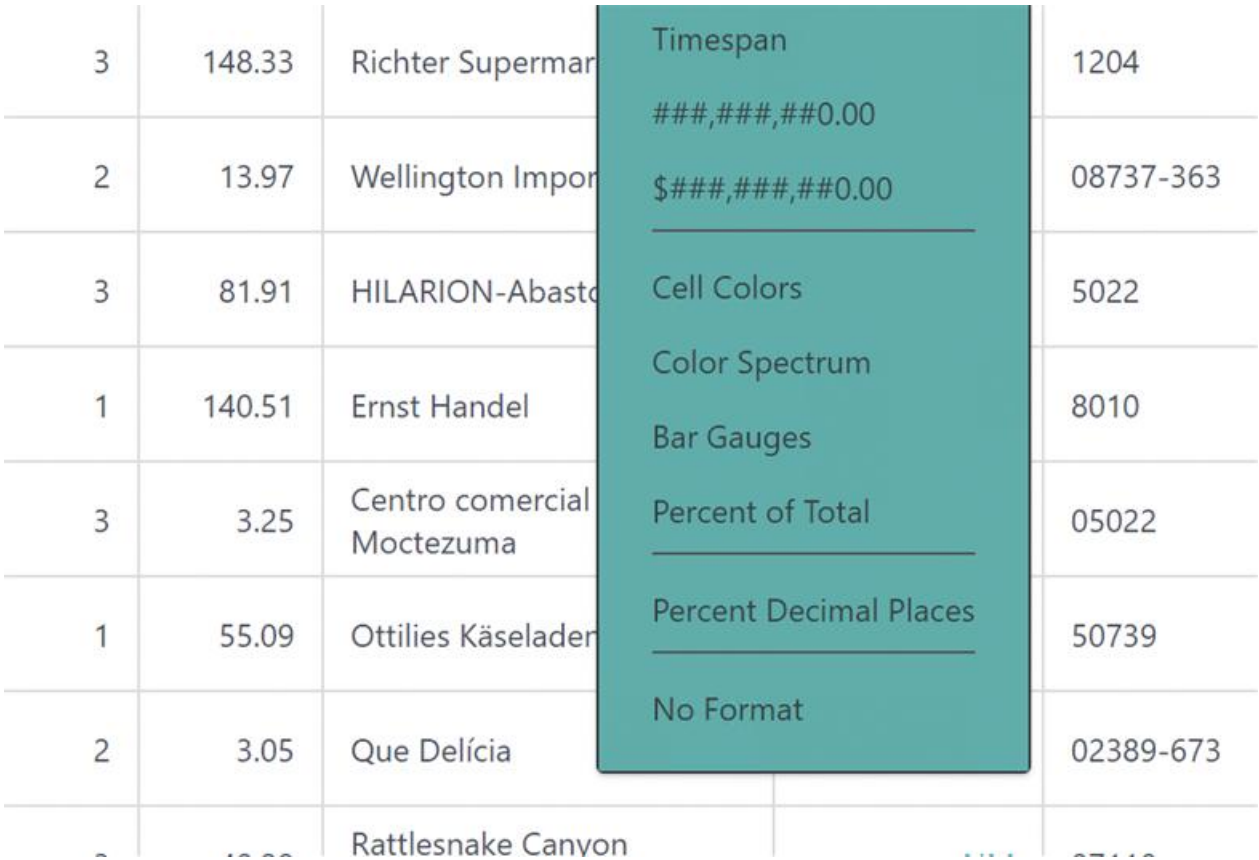
1. Choose a paging option:
    - **Show all rows** will display all of the data at once in the table. *Caution*: Selecting all rows can result in a length delay while data is retrieved.
    - **Show paging** will display a fixed number of rows per page and display the paging controls.
2. If **Show paging** has been selected, enter the number of data rows to display per table page. Click **OK** to refresh the display.

These settings affect all tables simultaneously. Click the gear icon to hide the configuration area.

Analysis Grid - Formatting Data

You can format the appearance of the data in the Table by clicking a **table column header** and selecting **Format** and then the desired option:



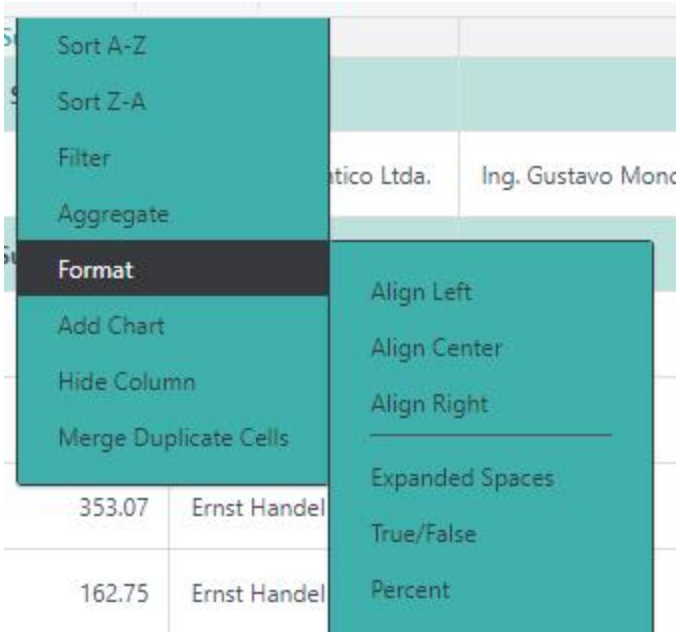


These options allow you to apply, or remove, a variety of standard formats. For more information about these formats see *Format Data*.

Selecting **No Format** will clear previously-selected numeric or color formatting but not alignment.

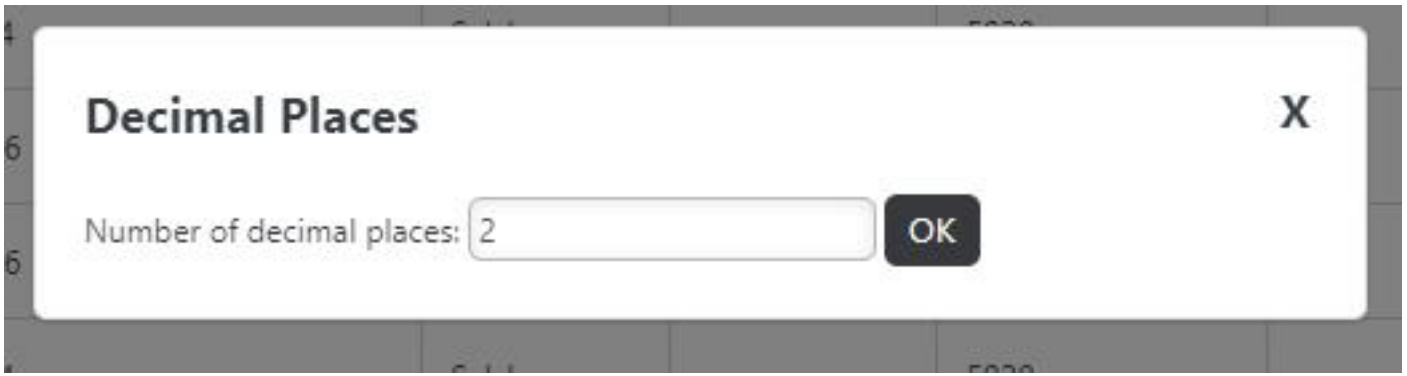
The Percent of Total format applies a calculation that determines each data value's percentage of the summed values and displays it as a percent.

When utilizing the Percent or Percent of Total formatting options you can further customize the decimal place by selecting **Percent Decimal Places**:



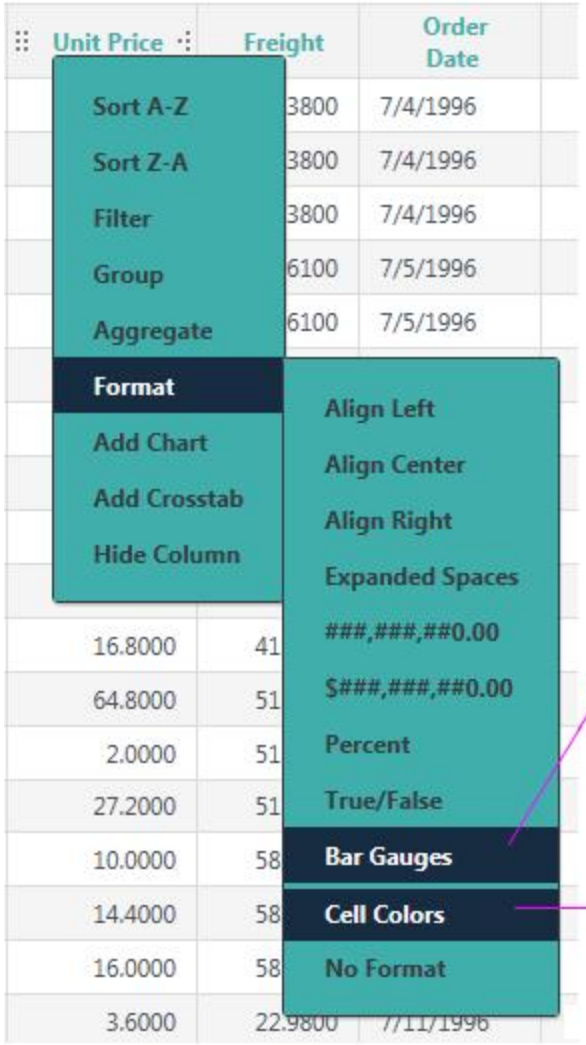
Both a number string value and an empty value is considered valid input. If the value is not a number, it will be ignored. An "empty" value will remove the setting, and decimal places will return to the default value, zero or two.

Enter the desired decimal place and select **OK**:



This setting is column-specific, meaning different columns with different percentage values can be customized to display different decimal places.

Two special format options insert visualizations right into the column:



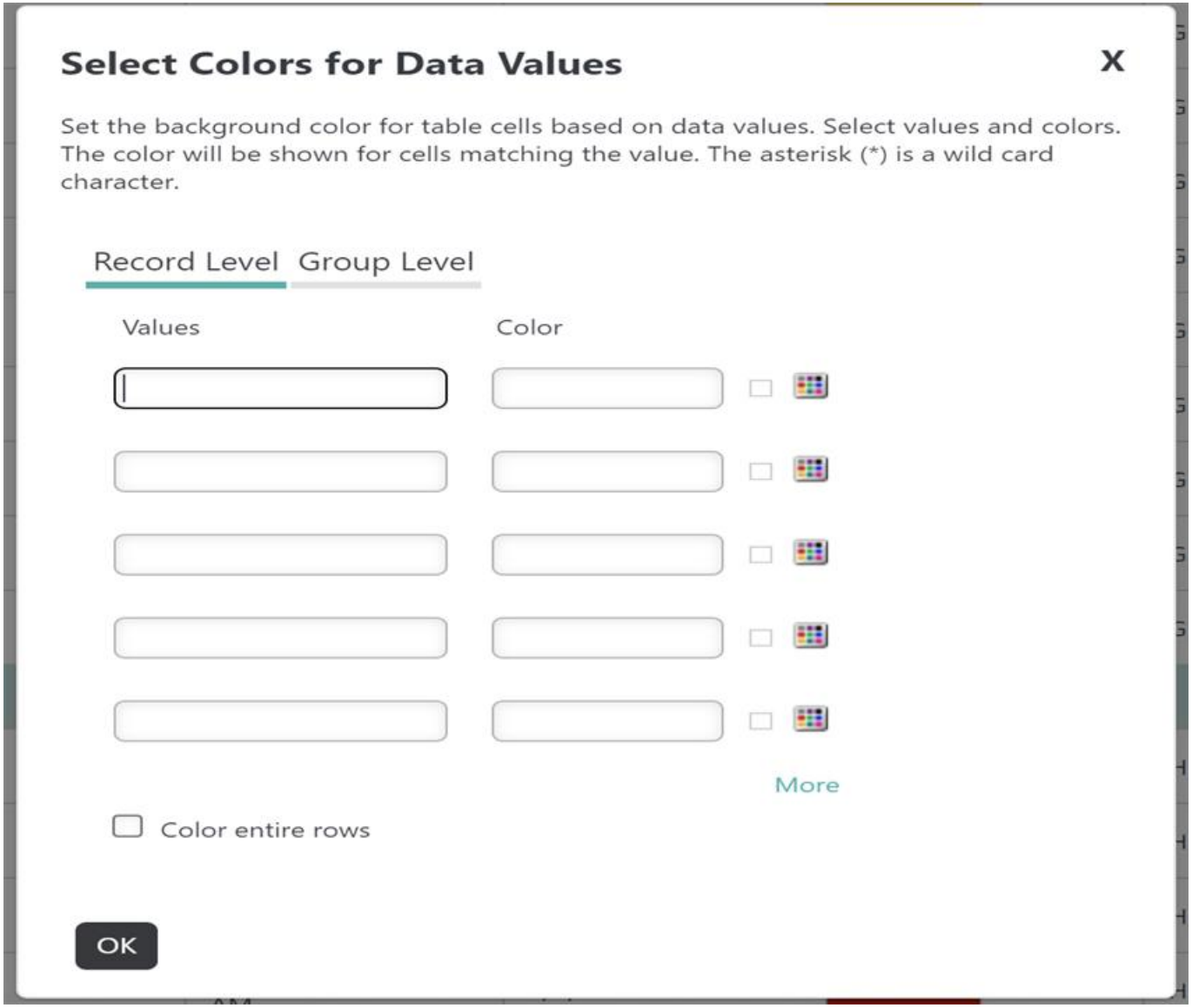
LastName	Quantity	Unit Price	Freight
Buchanan	12		32.3800
Buchanan	10		32.3800
Buchanan	5		32.3800
Suyama	9		11.6100
Suyama	40		11.6100
Peacock	10		65.8300
Peacock	35		65.8300

LastName	Quantity	Unit Price	Freight
Buchanan	12	14	32.3800
Buchanan	10	9.8	32.3800
Buchanan	5	34.8	32.3800
Suyama	9	18.6	11.6100
Suyama	40	42.4	11.6100
Peacock	10	7.7	65.8300
Peacock	35	42.4	65.8300

Selecting the **Bar Gauges** option will produce a horizontal bar gauge within the column, as shown above. The actual data value will appear in a tooltip if you hover your mouse over the gauge. Selecting the **Cell Colors** option will display the column value using an imbedded Cell Color Slider, as shown above. You can drag the slider in the column header to customize the color ranges.

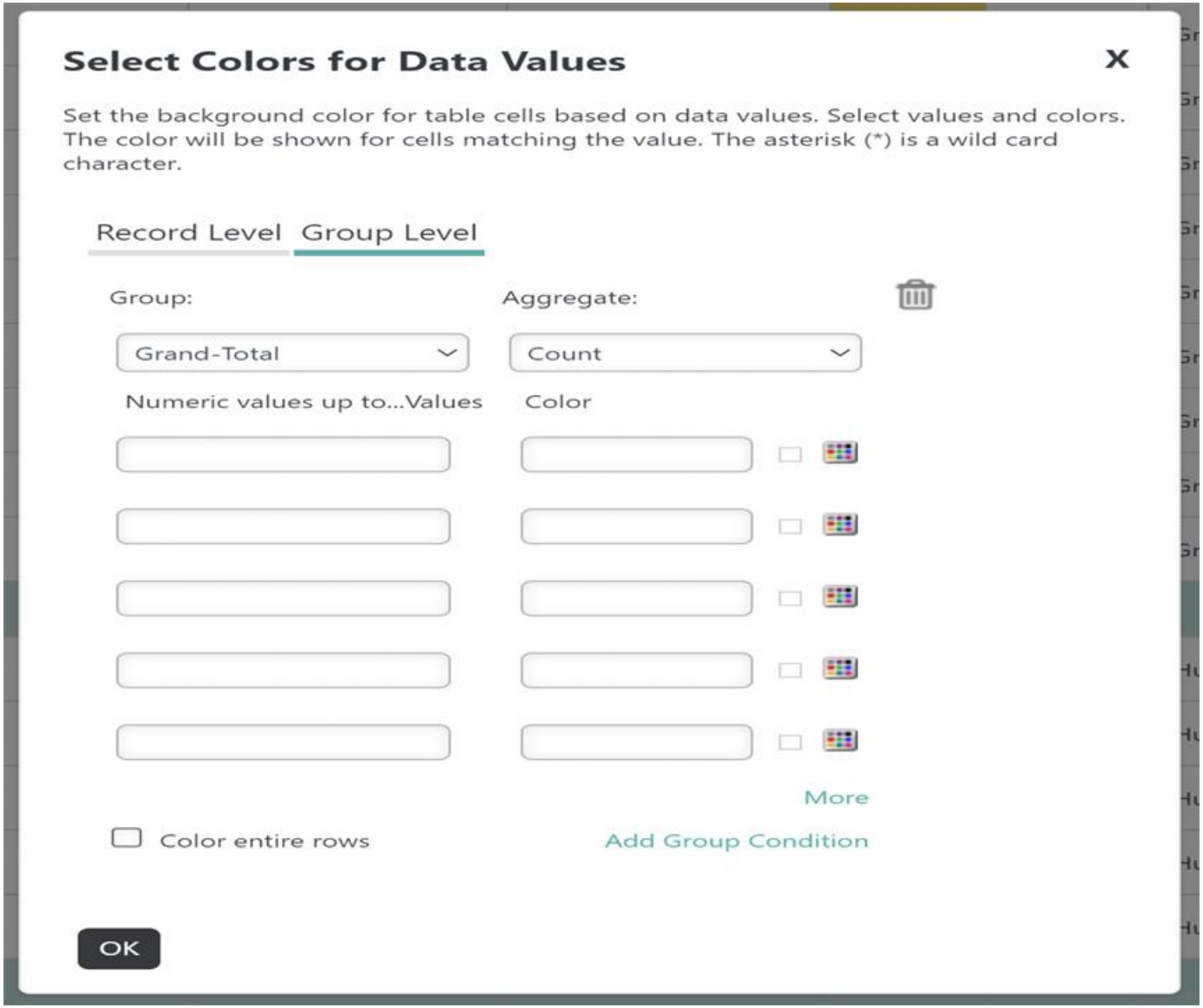
The Cell Colors feature has been changed to allow you to apply conditional colors:

Its previous functionality is now represented by a new Format menu item, Cell Color Slider, as shown above.


Select Colors for Data Values
X
Set the background color for table cells based on data values. Select values and colors. The color will be shown for cells matching the value. The asterisk (*) is a wild card character.
Record Level
Group Level
Values
Color
More
Color entire rows
OK

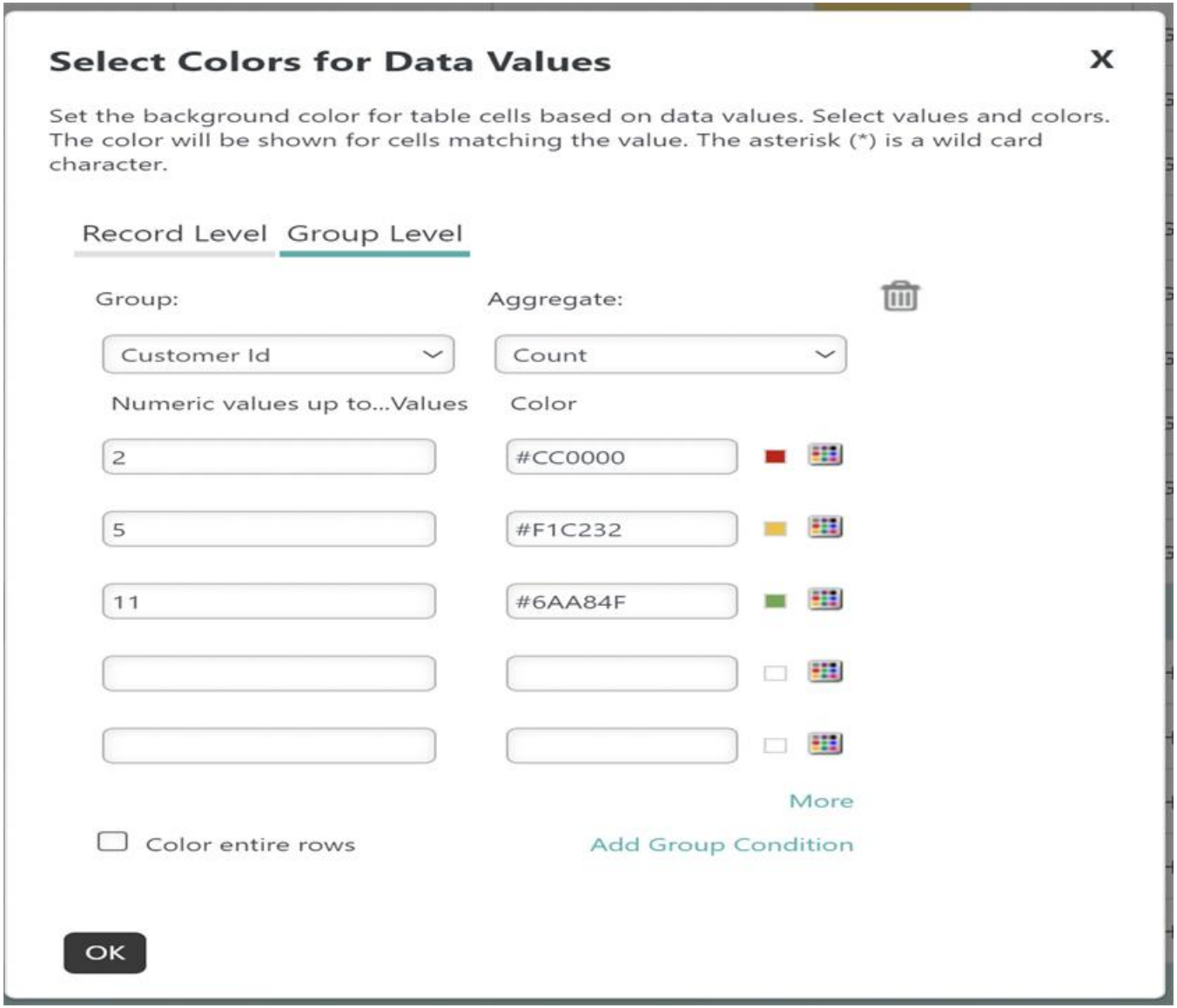
The menu's Cell Colors item now displays the pop-up dialog box shown above, where you can select data value ranges and cell background colors. The Values field can be customized to match the boolean format.

You can also format conditional colors on the group level by selecting the **Group Level** tab:


Select Colors for Data Values
X
Set the background color for table cells based on data values. Select values and colors. The color will be shown for cells matching the value. The asterisk (*) is a wild card character.
Record Level
Group Level
Group:
Aggregate:
Grand-Total
Count
Numeric values up to...Values
Color
More
Color entire rows
Add Group Condition
OK

Previously established grouping is available in the Group drop-down menu. Choose your **Group** and **Aggregate** using the drop-down menus.

Enter a numeric value in the field on the left and assign a color to the Count by entering a Hex code or using the Color Picker:



Select **OK** to apply your changes.

Your Analysis Grid reflects the conditional colors applied to the cells, like below:

Customer Id	Order Id	Employee Id	Order Date	Required Date	Shipped Date	Ship Via	Freight	Ship Name	Ship Region
Count: 122									
GREAL Count: 11									
	10528	6	May	5/20/1997 12:00:00 AM	5/9/1997 12:00:00 AM	2	3.35	Great Lakes Food Market	OR
	10589	8	Jul	8/1/1997 12:00:00 AM	7/14/1997 12:00:00 AM	2	4.42	Great Lakes Food Market	OR
	10616	1	Jul	8/28/1997 12:00:00 AM	8/5/1997 12:00:00 AM	2	116.53	Great Lakes Food Market	OR
	10617	4	Jul	8/28/1997 12:00:00 AM	8/4/1997 12:00:00 AM	2	18.53	Great Lakes Food Market	OR
	10656	6	Sep	10/2/1997 12:00:00 AM	9/10/1997 12:00:00 AM	1	57.15	Great Lakes Food Market	OR
	10681	3	Sep	10/23/1997 12:00:00 AM	9/30/1997 12:00:00 AM	3	76.13	Great Lakes Food Market	OR
	10816	4	Jan	2/3/1998 12:00:00 AM	2/4/1998 12:00:00 AM	2	719.78	Great Lakes Food Market	OR
	10936	3	Mar	4/6/1998 12:00:00 AM	3/18/1998 12:00:00 AM	2	33.68	Great Lakes Food Market	OR
	11006	3	Apr	5/5/1998 12:00:00 AM	4/15/1998 12:00:00 AM	2	25.19	Great Lakes Food Market	OR
	11040	4	Apr	5/20/1998 12:00:00 AM		3	18.84	Great Lakes Food Market	OR
	11061	4	Apr	6/11/1998 12:00:00 AM		3	14.01	Great Lakes Food Market	OR
HUNGC Count: 5									
	10660	8	Sep	10/6/1997 12:00:00 AM	10/15/1997 12:00:00 AM	1	111.29	Hungry Coyote Import Store	OR
	10600	4	Jul	8/13/1997 12:00:00 AM	7/21/1997 12:00:00 AM	1	45.13	Hungry Coyote Import Store	OR
	10375	3	Dec	1/3/1997 12:00:00 AM	12/9/1996 12:00:00 AM	2	20.12	Hungry Coyote Import Store	OR
	10394	1	Dec	1/22/1997 12:00:00 AM	1/3/1997 12:00:00 AM	3	30.34	Hungry Coyote Import Store	OR
	10415	3	Jan	2/12/1997 12:00:00 AM	1/24/1997 12:00:00 AM	1	0.2	Hungry Coyote Import Store	OR
LAZYK Count: 2									

To utilize this feature you must have an aggregation defined at the group level.

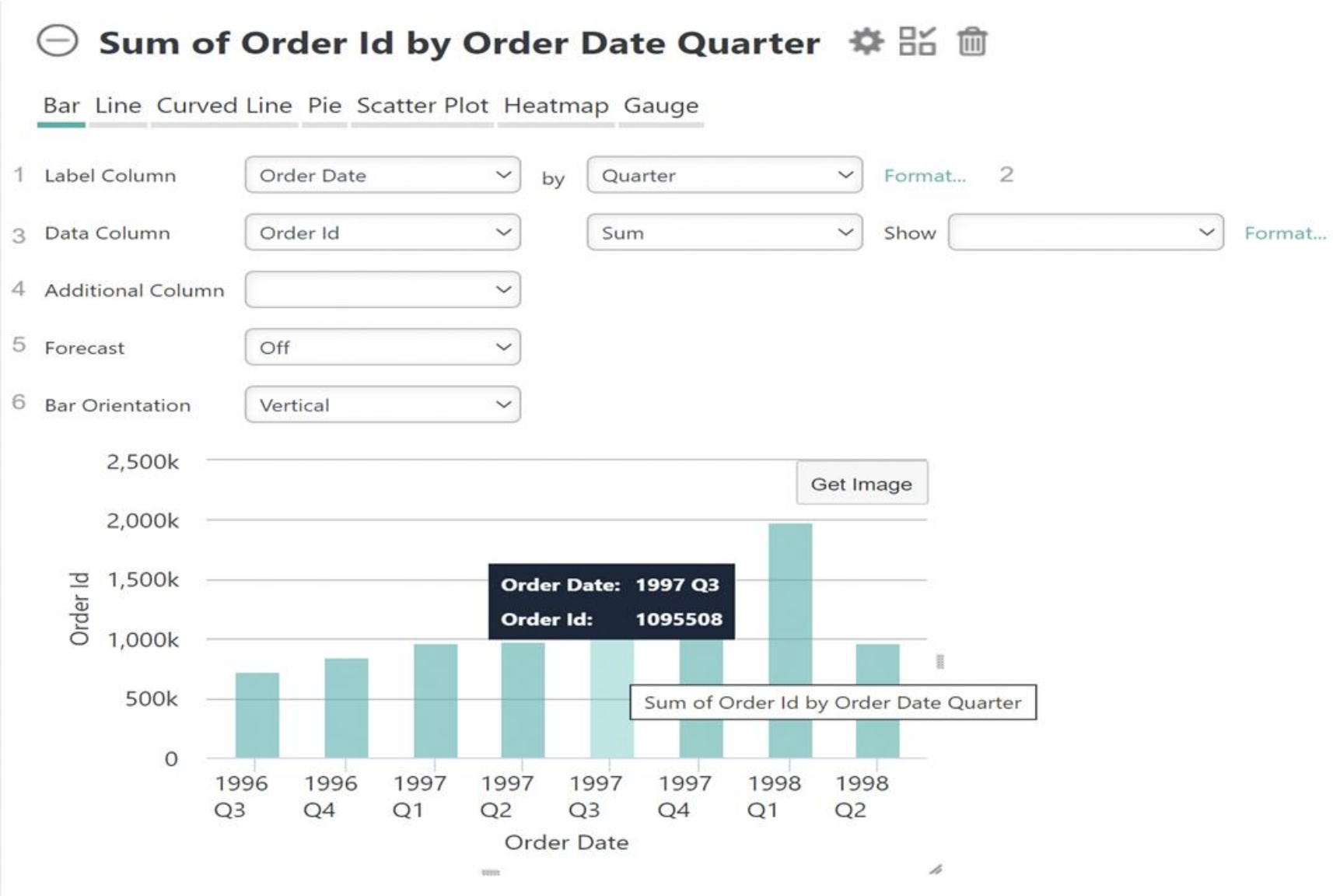
Analysis Grid - Creating Charts and Gauges

This topic introduces how to create charts and gauges in the Analysis Grid.

The following sections are covered in this topic:

Creating Charts

Click the **Add Chart** tab to use the feature that lets you create charts and gauges. A separate Chart panel with its own configuration area will be displayed:



Here's how to use this feature (some controls shown above only appear for specific data and chart types):

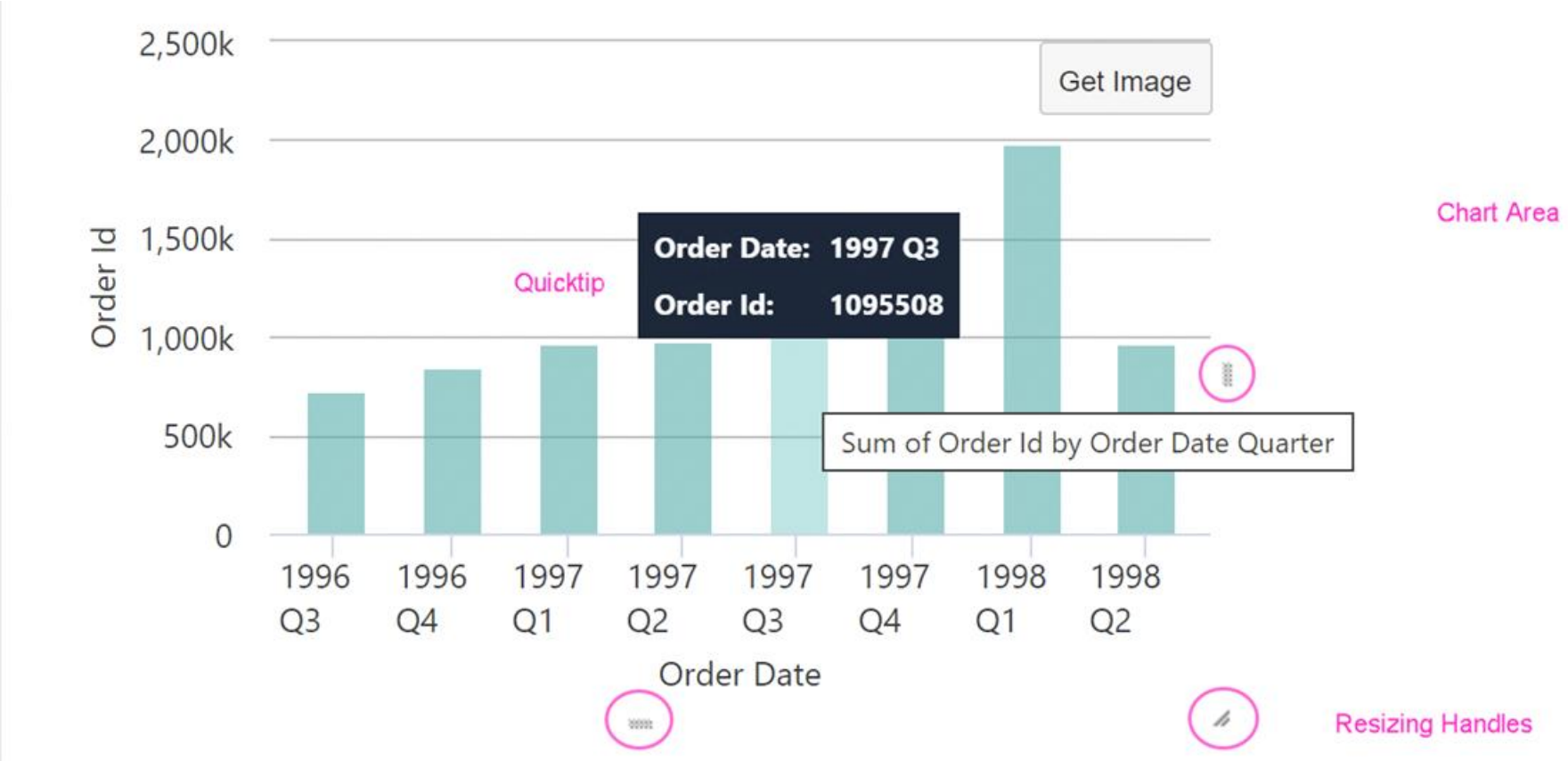
1. Assuming a **Bar** chart was selected, select the **Label Column**. This provides data for the X-axis of the chart. If the column selected is a date-type column, an interval control (*Year*, *Quarter*, *Month*, *Day*, *Hour, Minute*) will be displayed. For text type columns, you can select the Label data sort order, A-Z or Z-A, in a sort control.
2. The **Format...** option, shown next to the Label Column and Data Column, allows you to format captions and labels for your charts. If you add an Additional Column(s), this option will also be available.
3. Select the **Data Column** (Y-axis data, the "height" of each bar). You'll see that the options are grouped and color-coded to make it easier for you to identify them. If you created any Formula columns, they'll be in there, too. You can also choose to show the actual data values on the chart.
    - Select a Data Aggregation function. Options include: *Sum*, *Average*, *Standard Deviation*, *Count*, *Distinct Count*, *Minimum*, and *Maximum*. Columns with Null values are *excluded* by default from aggregations.
    - Select whether or not to show the actual data values or legend as labels within the chart.
4. The **Additional Column** specifies additional data series that will be charted along with the Data Column values for comparison. Depending on the chart type, other controls will appear for use configuring a second series, including aggregation options and charting types. You can also choose to show the actual data value on the chart.
5. Data **Forecasting** is available for Bar, Line, Curved Line, and Scatter Plot charts that use Date/Time columns. See the section below for more information.
    - If your chart uses text-type columns, the **Relevance** option replaces Forecast. Relevance allows you to tune the data shown by *Automatic*, *Rank*, and *Percentage*.
6. **Bar Orientation** allows you to choose whether the bars are vertical or horizontal arrangements for Bar charts.

Depending on how your application is configured, you may see an "OK" or "Update" button that applies all of your selection changes to the chart at once. Otherwise, your chart will be redrawn immediately as you make individual changes.

Hide the chart configuration area by clicking the **gear** icon, or delete the chart completely by clicking the **trash can** icon, shown below:

Charts are displayed in their own panels, so you can expand and collapse them using their "+" and "-" icons. You can also rearrange the order of chart and table panels by clicking with your mouse near the top of a panel, and dragging it up or down.

Charts will automatically include Quicktips, which appear when you hover your mouse over a data value, as shown below. In addition, "resizing handles" (circled below) will appear when the mouse is over the chart, allowing you to resize the chart by dragging them.



Unlike the chart shown above, Bar charts that are not time-oriented will automatically be shown in a horizontal format. This allows greater clarity in reading the "X-axis" label text.

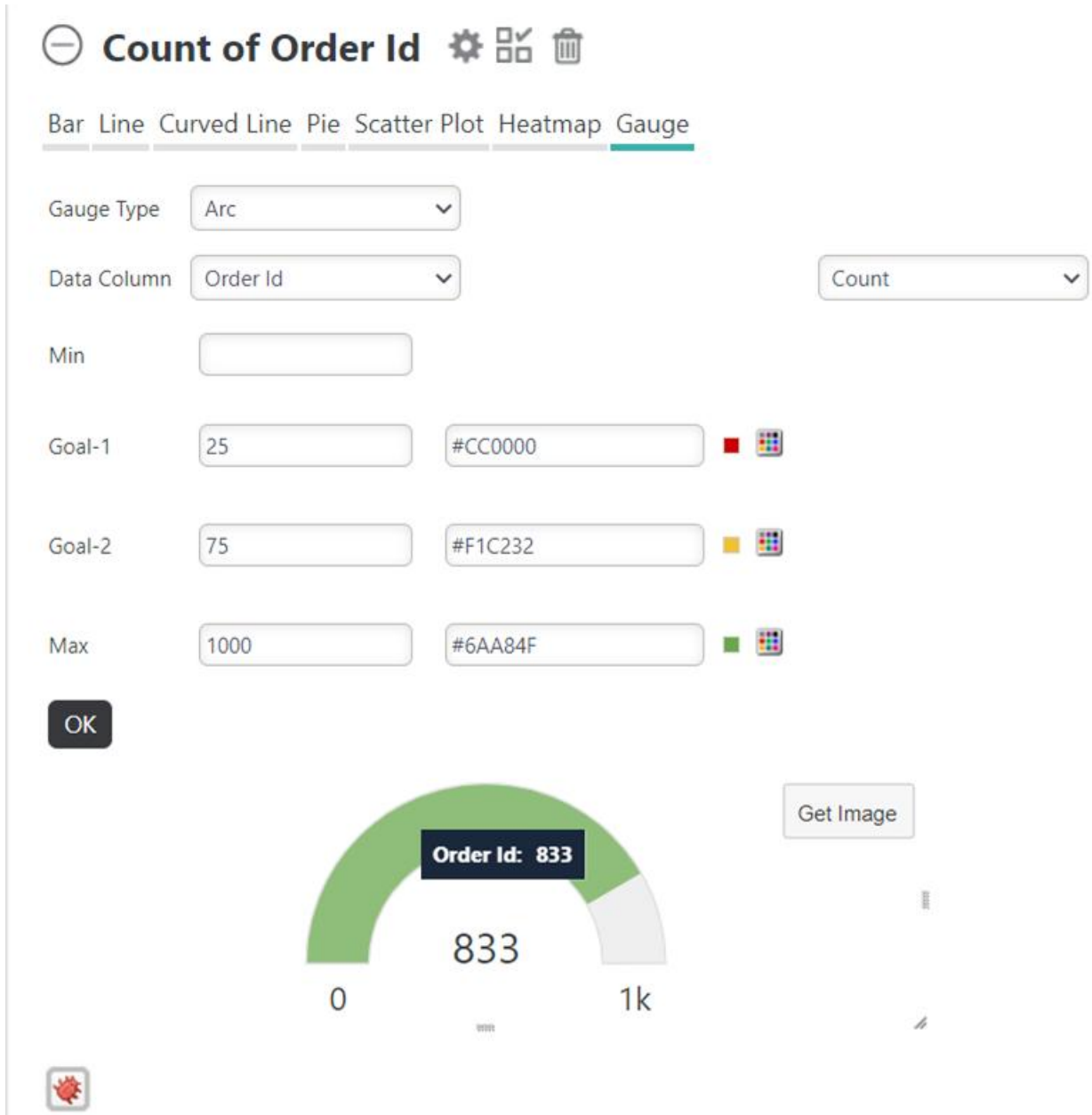
v23.1 If your Analysis Grid has been configured for it, you can utilize chart drillthrough capabilities. With drillthrough, selecting a section of the chart links to a detailed report of the data, as shown below:

Scatter Plot and Gauges do not support drillthrough.

If your application has been configured for it, you can utilize Zoom control to zoom in on your data. For more information, see "Analysis Grid - Zoom Control" on page 187.

Creating Gauges

Click the **Add Chart** tab to use the feature that lets you create charts and gauges. Then, select **Gauge** from list of available charts. A separate Chart panel with its own configuration area will be displayed:


Count of Order Id
Bar Line Curved Line Pie Scatter Plot Heatmap Gauge
Gauge Type
Arc
Data Column
Order Id
Count
Min
Goal-1
25
#CC0000
Goal-2
75
#F1C232
Max
1000
#6AA84F
OK
Get Image
Order Id: 833
833
0
1k

Here's how to use this feature:

1. Choose a **Gauge Type**. Options include: *Arc*, *Balloon Bar*, *Bullet Bar*, and *Number*.

2. Select the **Data Column**. The options are grouped and color-coded to make it easier for you to identify them. If you created any Formula columns, they appear in this column, too.
    - Select a Data Aggregation function. Options include: *Sum*, *Average*, *Standard Deviation*, *Count*, *Distinct Count*, *Minimum*, and *Maximum*. Columns with Null values are *excluded* by default from aggregations.
3. Enter a **Min** value. If you do not enter a Min value, Info automatically applies Min value of 0.
4. Set up to two **Goals**. Your goals can be set using static values, or dynamically driven by percentages. The goal will be invalid if out of range. Set color thresholds for your goals by selecting the **color picker** and choosing a **color**. When using percentages, the following rules apply:

    - If percentage-value is used, the value cannot exceed 100% (and/or less than 0%).

    - If the percentage-value and absolute-value are mixed in Goal-1/2, Max, or GuageRange-elements, the maximum value of all absolute-values is the maximum value. For example, if your settings are 10, 90%, 60, 70% and 80 respectively, the maximum value is 80, and the final range is ~ 10 ~ 56 (70%) ~ 60 ~ 72 (90%) ~ 80.

    - If ONLY percentage-value is used, the automatically-calculated-maximum-value will be taken as the maximum value. For example, if your settings are 10%, 90%, 60%, 70%, 80%, the automatically-calculated-maximum-value is 1000, and the final range is ~ 100 (10%) ~ 600 (60%) ~ 700 (70%) ~ 800 (80%) ~ 900 (90%).

5. Enter a **Max** value. If you do not enter a Max value, Info automatically applies a Max value to your chart. The color of the arc (in this example) is determined by which range contains the value. Set a color threshold for your Max value by selecting the **color picker** and choosing a **color**.

Depending on how your application is configured, you may see an "OK" button that applies all of your selection changes to the chart at once. Otherwise, your chart will be redrawn immediately as you make individual changes.

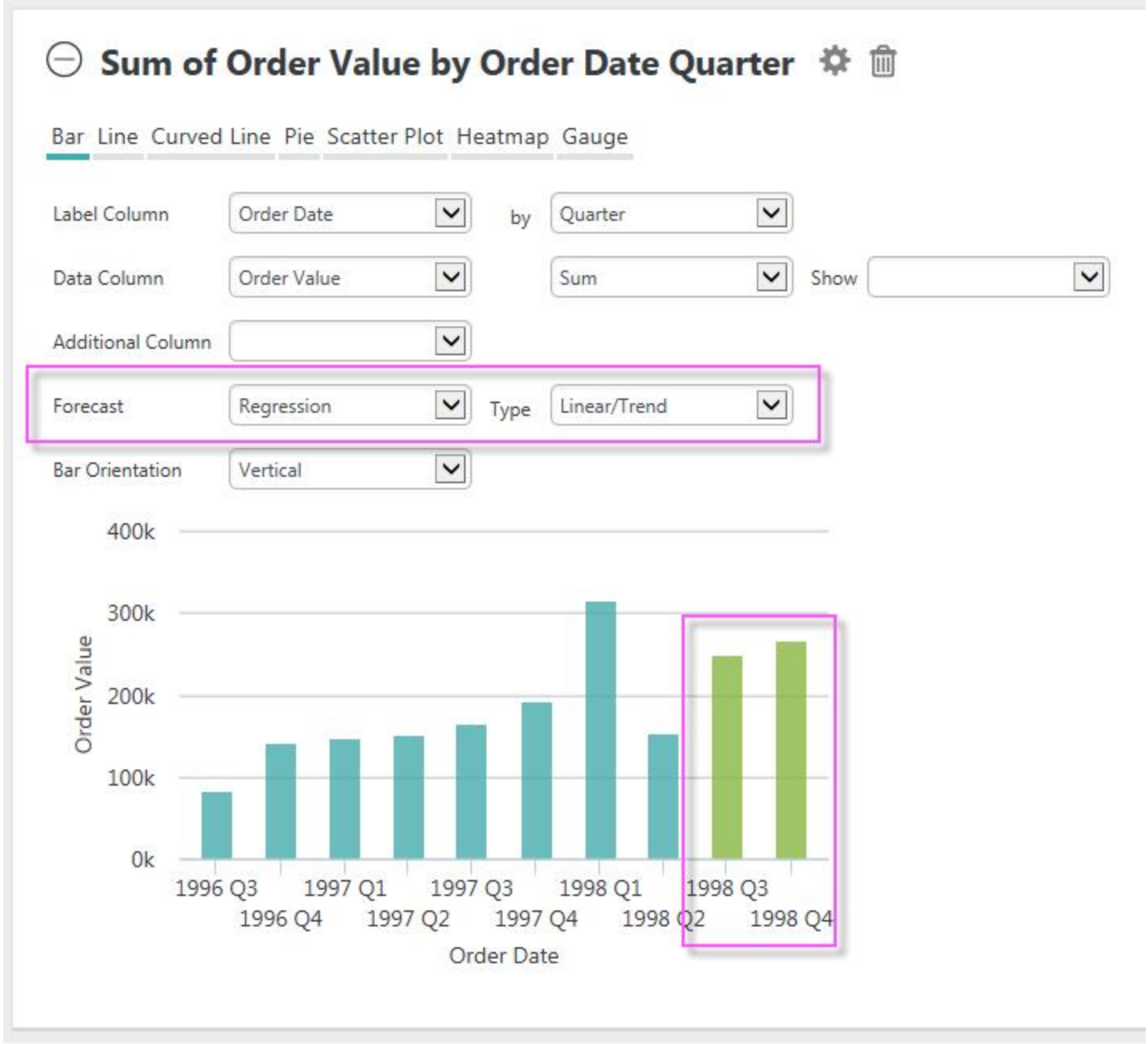
Hide the chart configuration area by clicking the **gear** icon, or delete the chart entirely by clicking the **trash can** icon.

Charts are displayed in their own panels, so you can expand and collapse them using their "**+**" and "**-**" icons. You can also rearrange the order of chart and table panels by clicking with your mouse near the top of a panel, and dragging it up or down.

Charts will automatically include Quicktips, which appear when you hover your mouse over a data value, as shown above. In addition, "resizing handles" (circled above) will appear when the mouse is over the chart, allowing you to resize the chart by dragging them.

Data Forecasting

Data Forecasting, if included by your application developer, is available for Bar, Line, Curved Line, and (in v12.2+) Scatter charts.


Sum of Order Value by Order Date Quarter
Bar Line Curved Line Pie Scatter Plot Heatmap Gauge
Label Column
Order Date
by
Quarter
Data Column
Order Value
Sum
Show
Additional Column
Forecast
Regression
Type
Linear/Trend
Bar Orientation
Vertical
400k
300k
200k
100k
0k
Order Value
1996 Q3
1996 Q4
1997 Q1
1997 Q2
1997 Q3
1997 Q4
1998 Q1
1998 Q2
1998 Q3
1998 Q4
Order Date

If it's available, extra controls for it will appear in the Chart configuration panel, as shown above.

Data forecasting is the process of generating values based on events that have *not yet* occurred. "Prediction" is a similar but more general term. Forecasting refers to formal, statistical methods that use time series, cross-sectional, or longitudinal data to produce predicted data. Typically, forecasts are displayed most effectively on charts.

Forecasting analysis options, which may vary by chart, include:

- **Time Series**(Time Series Decomposition), consisting of data in a natural, time-related order with a strong interval, where the Label Column data is of DateTime-type and the Data Column is a number.
- **Regression**, using one of several regression analysis functions. Regression analysis is recommended when the focus is on a relationship between a dependent value and one or more independent values. Available analysis functions include:
  - *Linear* - used to calculate predictive values based on a trend line.
  - *Autoregressive*- used when attempting to predict an output of a system based on previous outputs. The estimation technique used is based on "Burg's" method.
- *Exponential*, *Logarithmic*, *Polynomial*, or *Power* - non-linear types used to display the relationship between dependent and independent variables as a curvilinear function, which may provide more accuracy than a linear regression.

Analysis Grid - Editing Chart Labels and Captions

Every Analysis Grid chart has the ability to format captions and labels for your Label Column, Data Column, and Additional Column(s) in some capacity. This topic discusses how to customize the label format and edit captions for the X- and Y-axis in your charts.

Editing Captions

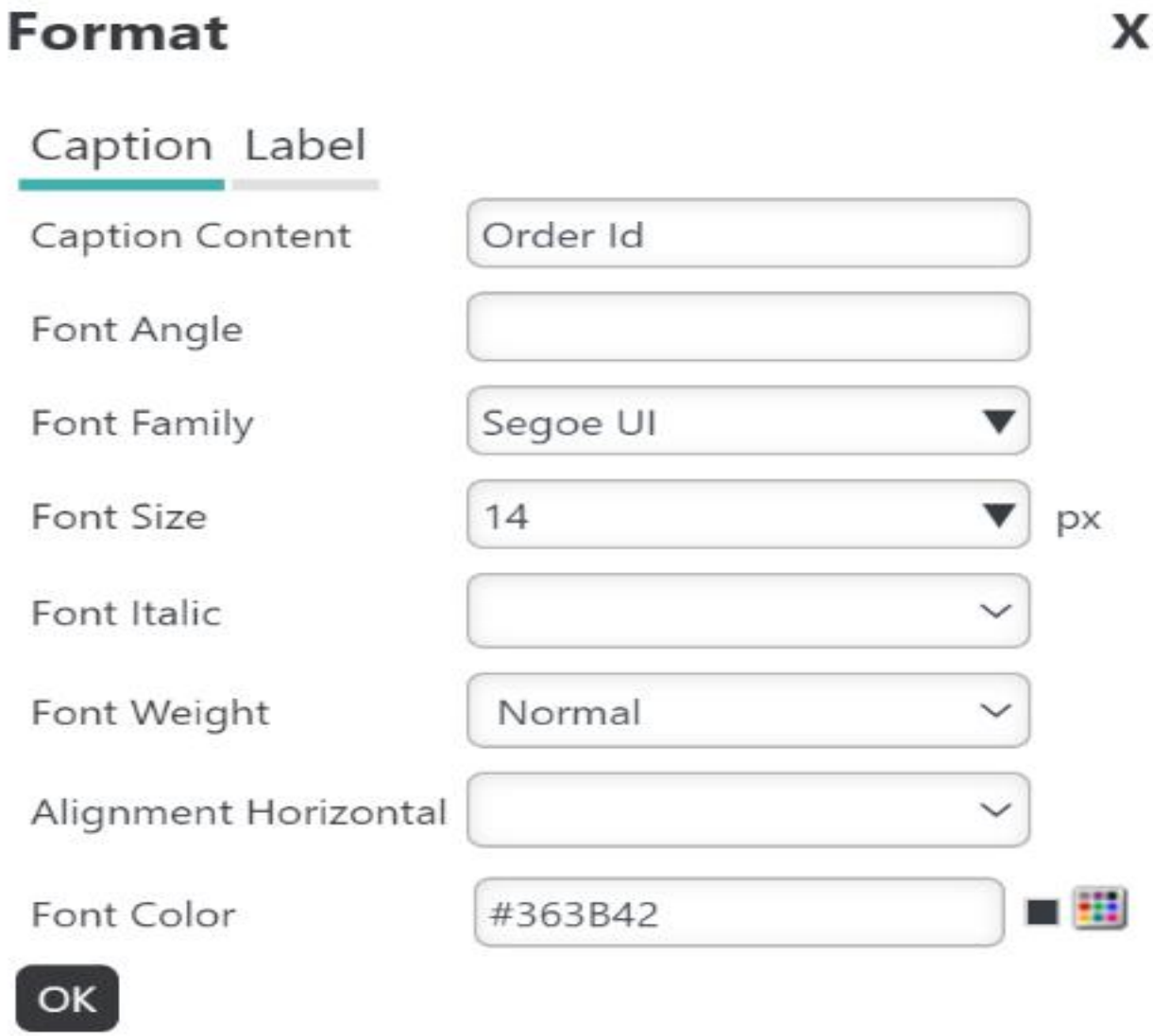
Axis caption formatting includes the ability to change the following:

- **Caption Content:** Enter the axis caption. To remove the caption, enter "".
- **Font Angle:** Enter a value between 1-360 to angle your caption.
- **Font Family:** Changes the font of your caption.
- **Font Size:** Changes the font size of your caption between 6-72 px.
- **Font Italic:** Select True or False to make your caption italicized.
- **Font Weight:** Changes the thickness of the caption font between Normal, Bold, Bolder, or Lighter. Note that the Font Family you choose may affect how the font weight displays.
- **Alignment Horizontal:** Select Left, Center, or Right to set your caption alignment.
- **Font Color:** Changes the font color and hue using a 'color picker'.

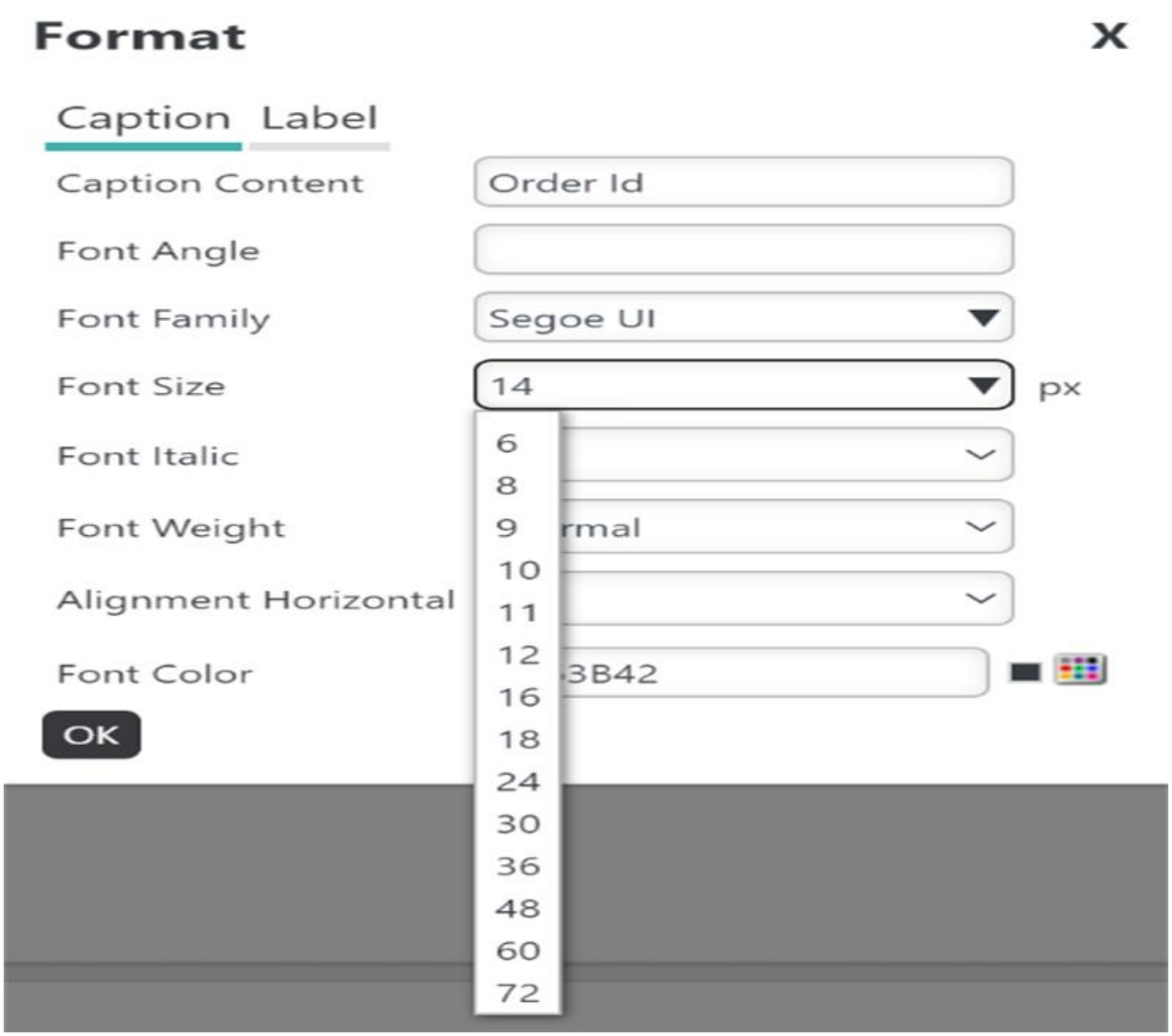
For this example, we're going to change the 'Order Id' caption's size, weight, and color.

1. To access this feature, select **Format...** in your chart:

A Format dialog window appears, shown below:

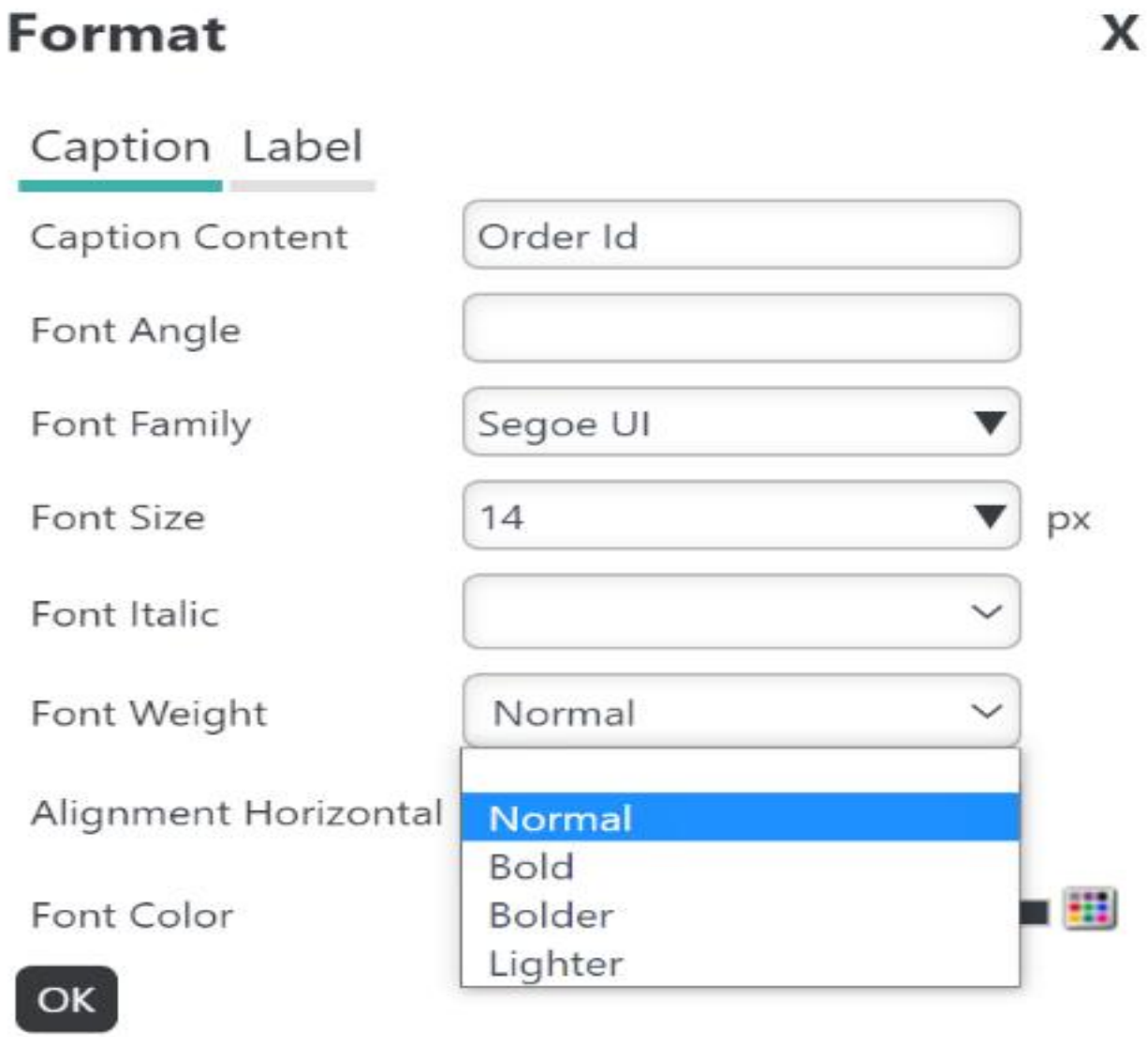


2. To change the size, select the **Font Size** drop-down menu:


Format
X
Caption
Label
Caption Content
Order Id
Font Angle
Font Family
Segoe UI
Font Size
14
px
Font Italic
Font Weight
rmal
Alignment Horizontal
Font Color
3B42
OK
6
8
9
10
11
12
16
18
24
30
36
48
60
72

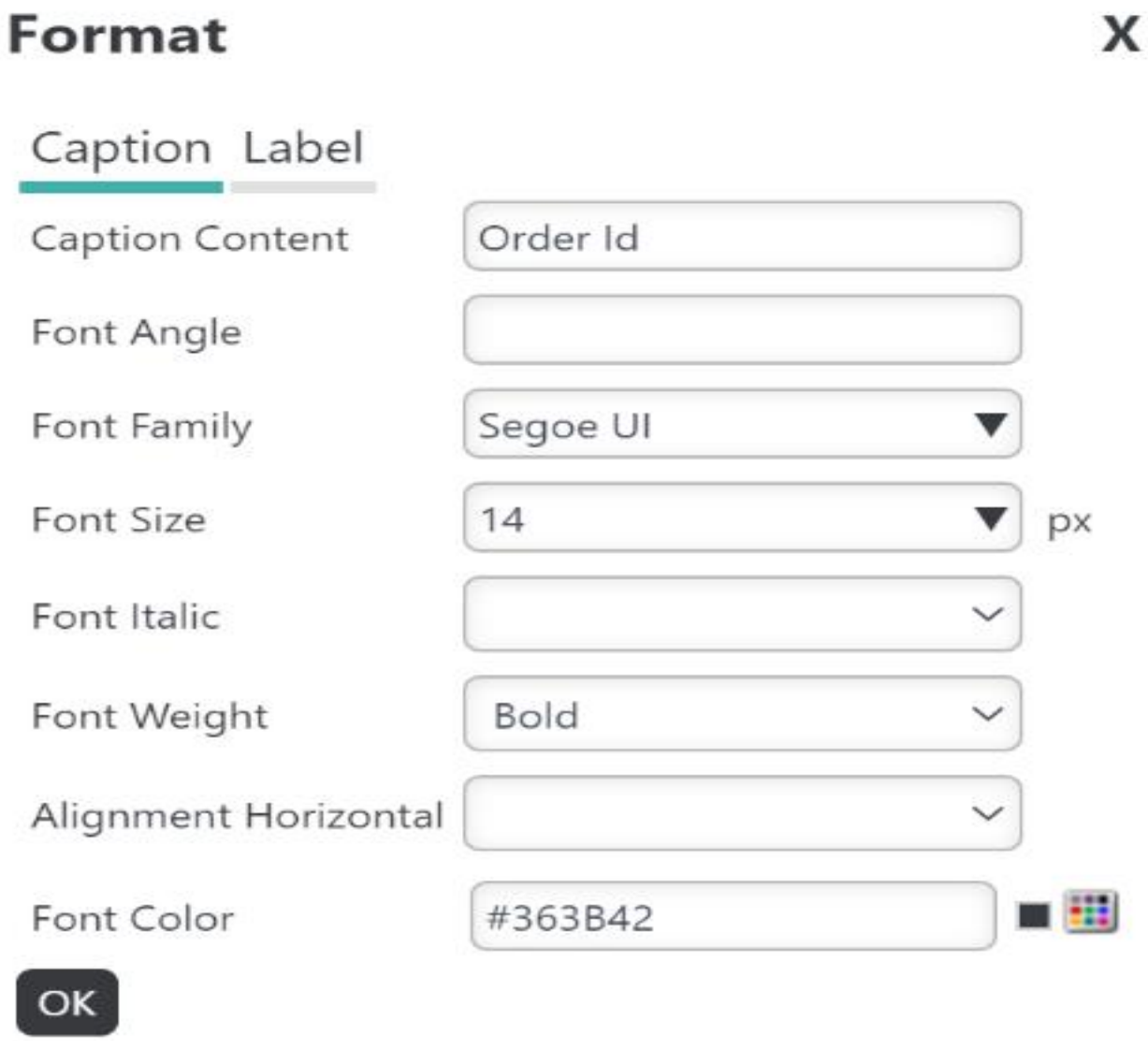
3. Choose the desired **size**. For this example, we're going to select size **14** font.

4. Then, select the **Font Weight** drop-down menu:



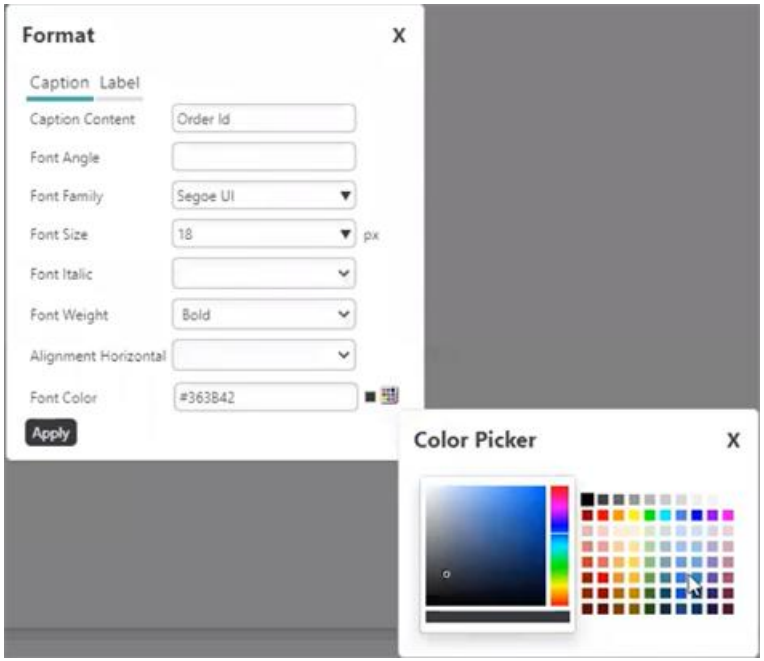
5. Choose the desired **weight**. We want our caption to be **Bold**.

6. To change the color, enter a Hex #, or choose a color by selecting the **Color Picker**:

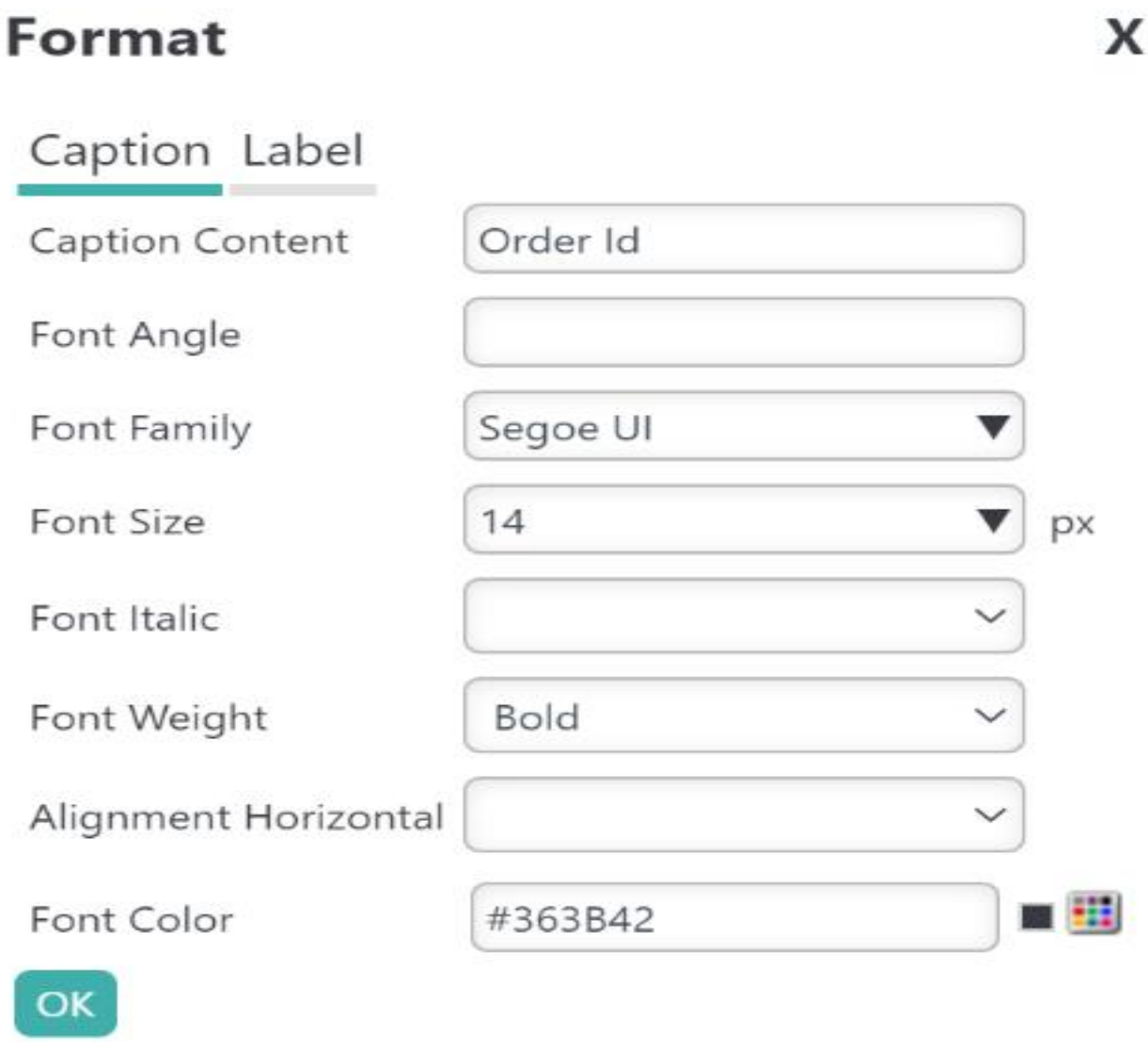


The Color Picker dialog displays.

7. Choose the desired caption **color** from the list of available colors. Change the shade of the color by moving the cursor to the left of the color panel, shown below:



8. When you're done making changes, select **OK**:



You chart automatically reflects the changes, shown below:

Since we edited the 'caption' our changes only applied to the words 'Order Id' (for this example). To change the format of the values represented in the Y axis (5M, 4M, 3M in this example), follow the steps in the section below.

Changing the chart type resets all formatting; however, changing the value in the Label, Data, or Additional Column does not reset formatting.

Editing Labels

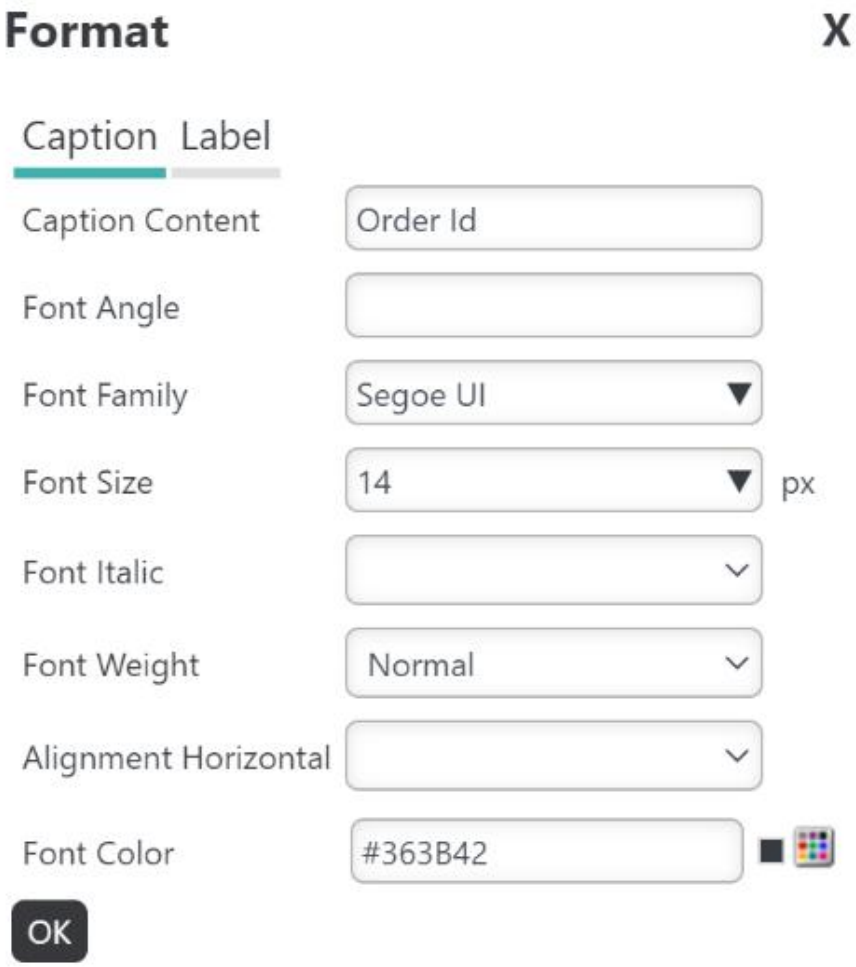
Axis label formatting includes the ability to change the following:

- **Format:** Changes the data format.
- **Font Family:** Changes the font of your label.
- **Font Size:** Changes the font size of your label between 6-72 px.
- **Font Italic:** Select True or False to make your label italicized.
- **Font Weight:** Changes the thickness of the font between Normal, Bold, Bolder, or Lighter. Note that the Font Family you choose may affect how the font weight displays.
- **Font Angle:** Enter a value between 1-360 to angle your label.
- **Font Color:** Changes the font color and hue using a 'color picker'.

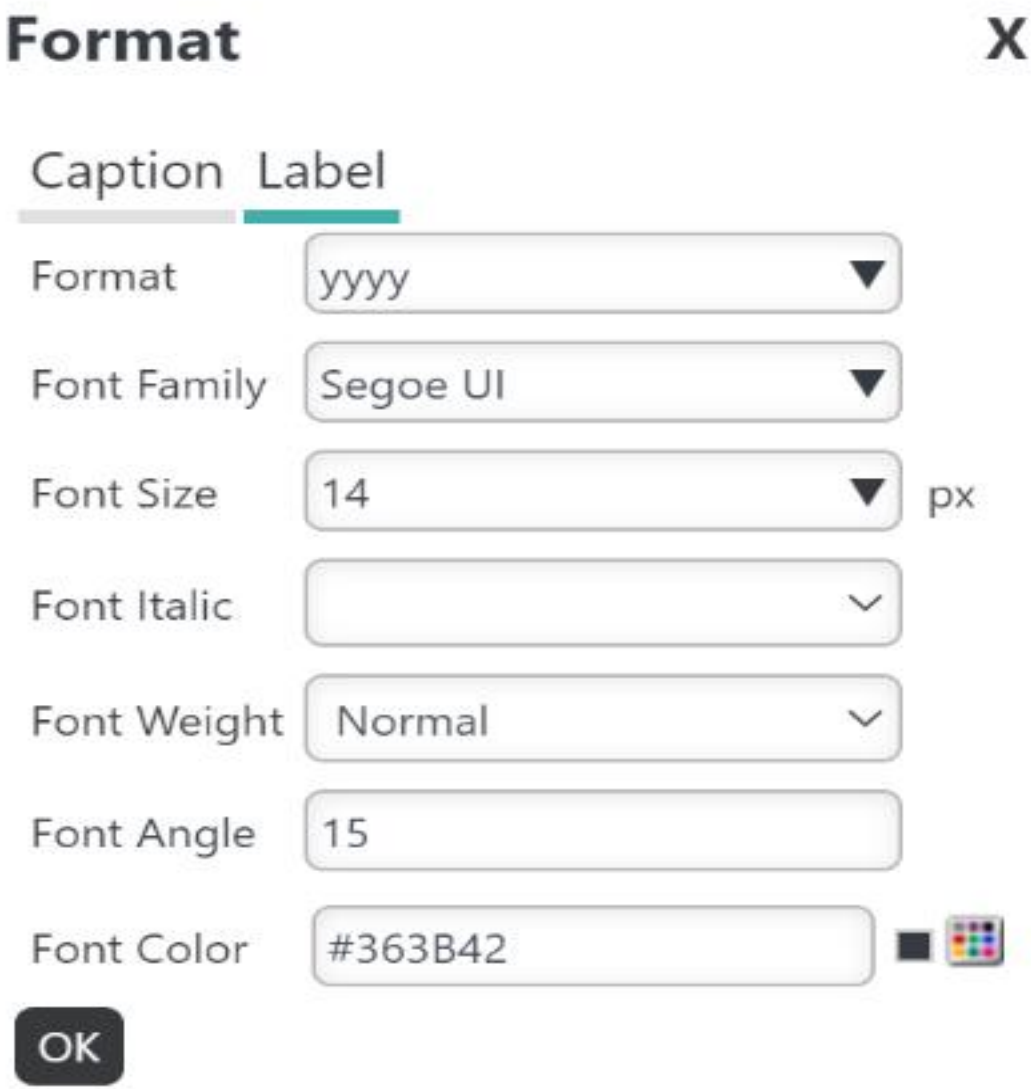
For this example, we're going to change the 'Order Id' label's format and font angle.

1. To access this feature, select **Format...** in your chart:

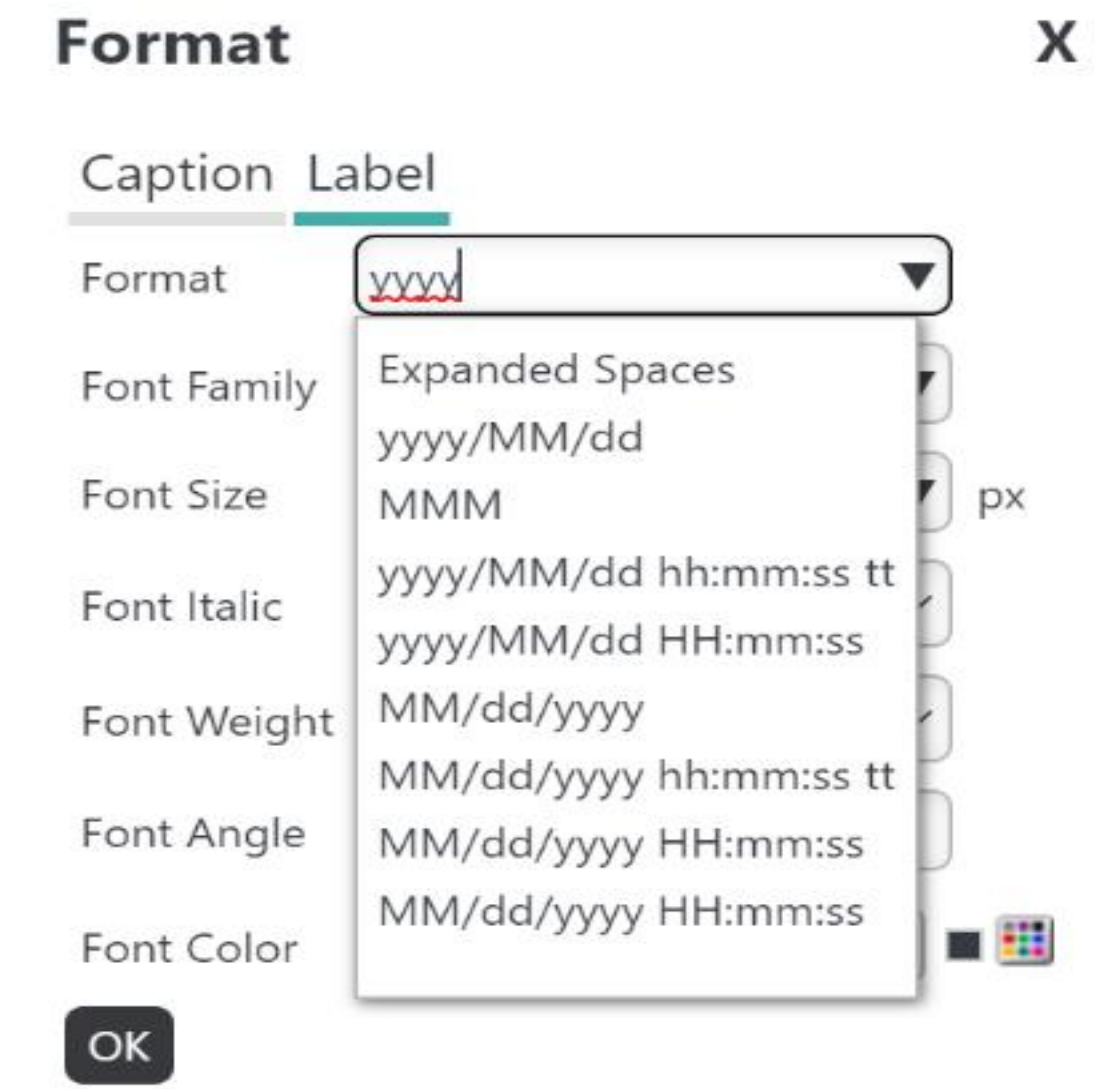
A Format dialog window appears, shown below:



2. Select the **Label** tab to edit the Data Column's label:

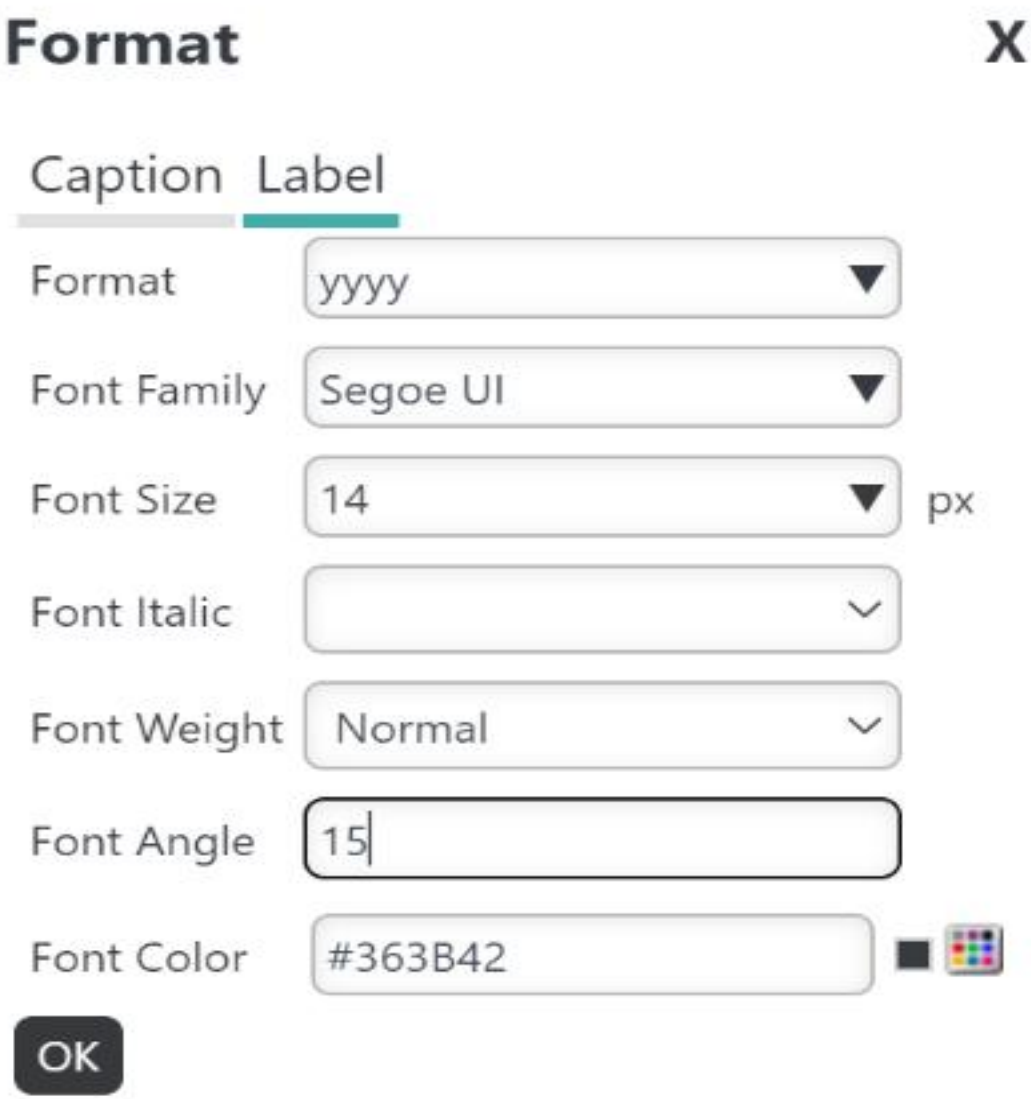

Format
X
Caption
Label
Format
yyyy
Font Family
Segoe UI
Font Size
14
px
Font Italic
Font Weight
Normal
Font Angle
15
Font Color
#363B42
OK

3. To change the format, select the **Format** drop-down menu:

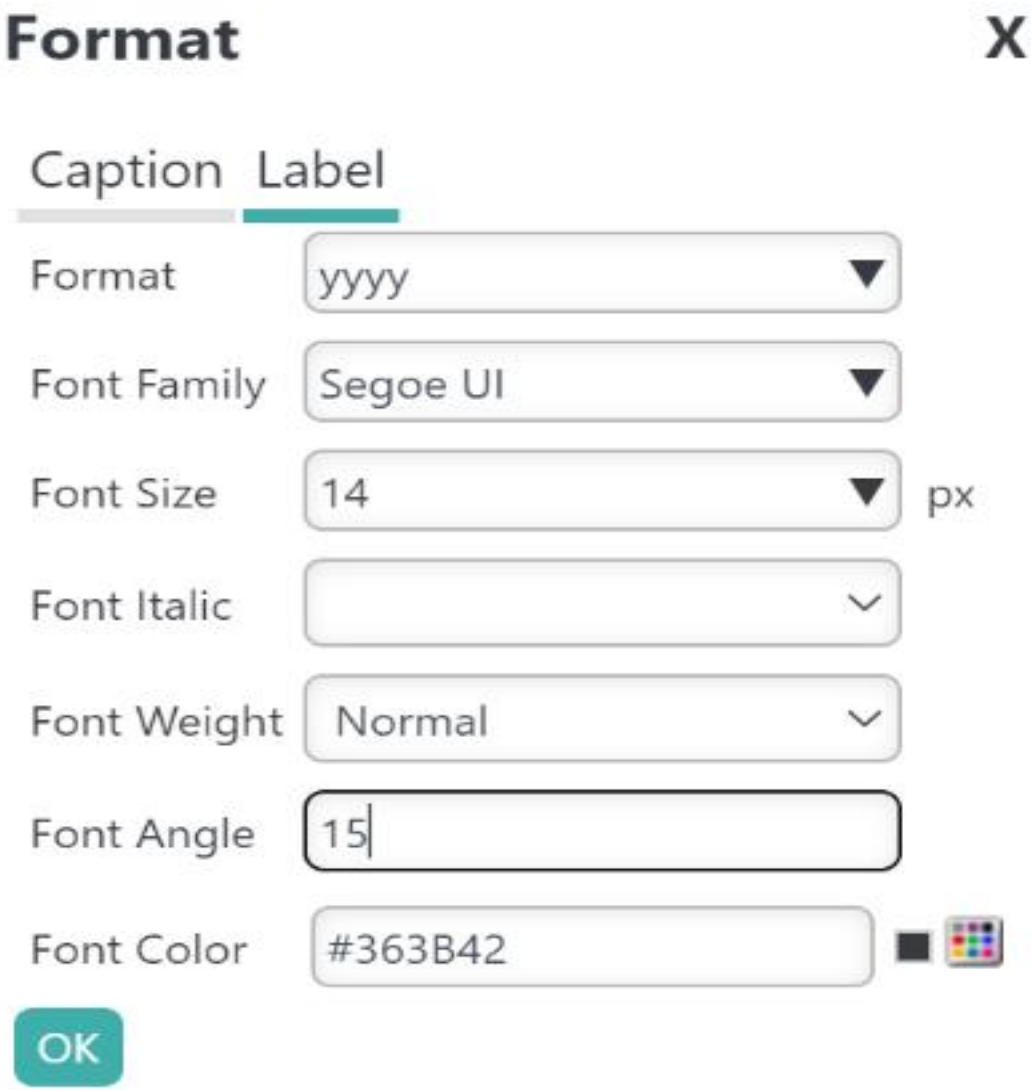


4. Choose the desired **format**, or enter your own format in the field. For this example, we're going to enter '**yyyy**'.

5. Next, we'll edit the angle of the label by entering a value (1-100) in the **Font Angle** field:


Format
X
Caption
Label
Format
yyyy
Font Family
Segoe UI
Font Size
14
px
Font Italic
Font Weight
Normal
Font Angle
15
Font Color
#363B42
OK

6. When you're done making changes, select **OK**:

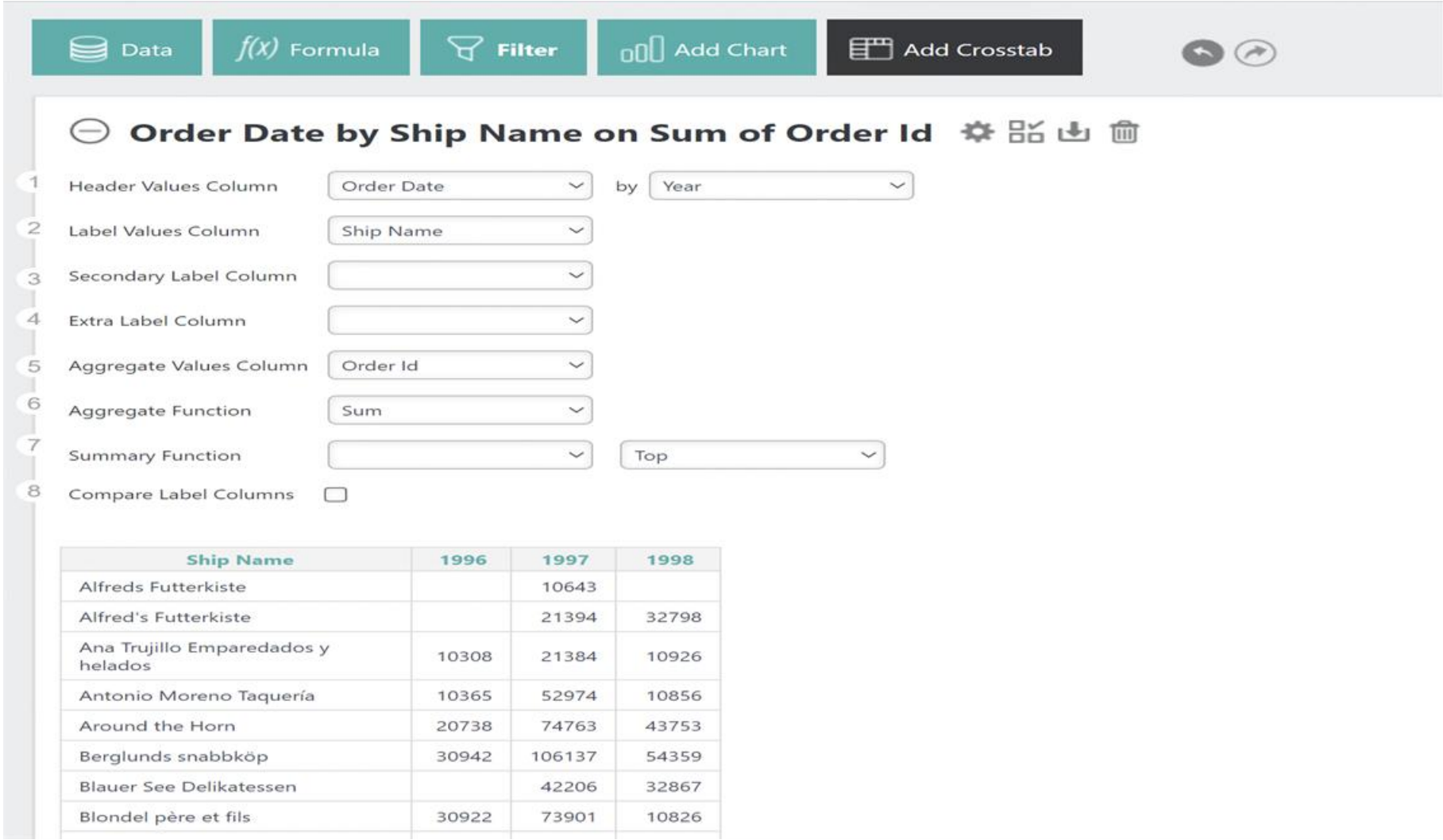


You chart automatically reflects the changes, shown below:

Changing the chart type resets all formatting; however, changing the value in the Label, Data, or Additional Column does not reset formatting.

Analysis Grid - Pivoting and Summarizing Data

Click the **Add Crosstab** tab or button to use the feature that lets you create a crosstab (also known as a "pivot") table:



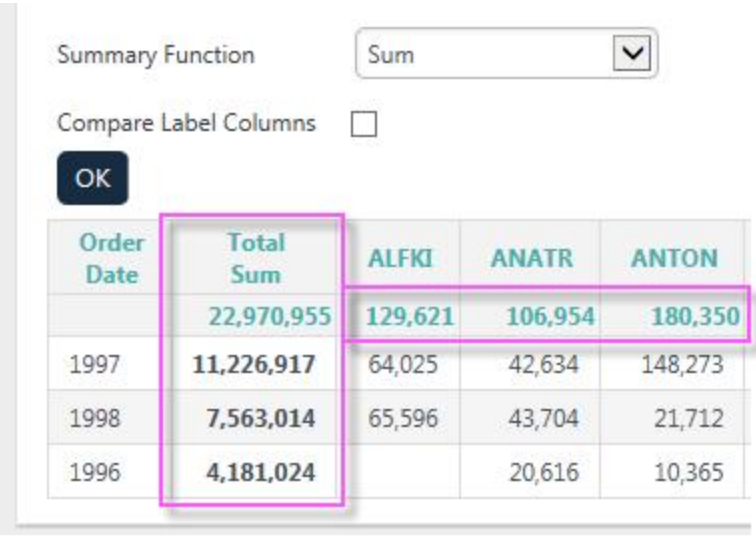
Ship Name	1996	1997	1998
Alfreds Futterkiste		10643	
Alfred's Futterkiste		21394	32798
Ana Trujillo Emparedados y helados	10308	21384	10926
Antonio Moreno Taquería	10365	52974	10856
Around the Horn	20738	74763	43753
Berglunds snabbköp	30942	106137	54359
Blauer See Delikatessen		42206	32867
Blondel père et fils	30922	73901	10826

Here's how to use this feature:

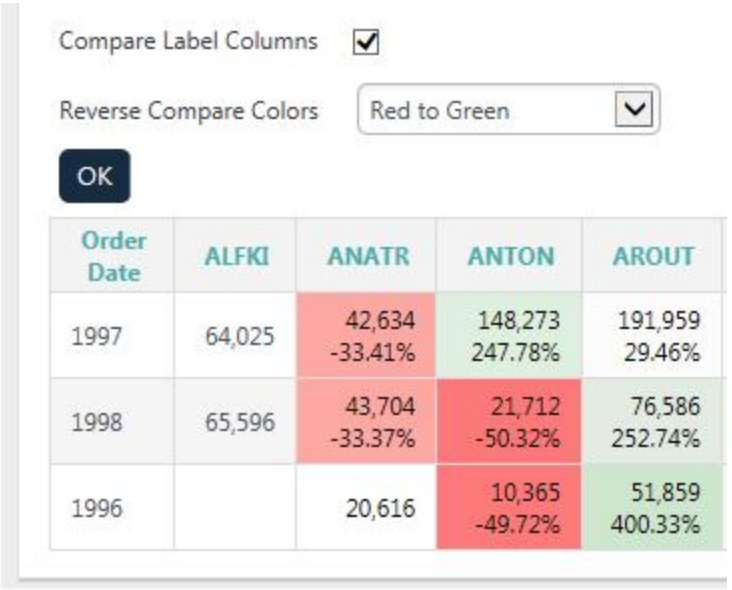
1. Select the **Header Values Column**, whose values will be shown *horizontally*, as column headers, across the top of the Crosstab Table. Each distinct value adds a crosstab column.
    - To prevent unmanageably wide tables, *numeric* columns in the original data and numeric Formula columns are not available for use here. Additional controls may appear depending on the data type of the selected column.
2. Select the **Label Values Column**, whose values will be shown *vertically*, in the left-most column of each row. Each Distinct value adds a crosstab row.
    - To prevent unmanageably long tables, *numeric* columns in the original data and numeric Formula columns are not available for use here.
3. Select the **Secondary Label Column**, whose values will be shown *vertically*, as column headers, after the Label Values Column. This column is used to group Value Columns and Label Columns together. To prevent unmanageably long tables, *numeric* columns in the original data and numeric Formula columns are not available for use here.
4. Select the **Extra Label Column**, whose values will be shown
5. Select the **Aggregate Values Column**, whose values will be *aggregated* to produce the contents for the rest of the table cells. Set the column to be aggregated into the crosstab cells.
6. Select the **Aggregate Function** to be applied to the column selected in Step 3. Options include *Sum*, *Average*, *Standard Deviation*, *Count* and *Distinct Count, Minimum* (v12.2), and *Maximum* (v12.2). Set the function for calculating the crosstab cells.
7. Select a **Summary Function** to display a summary result.
8. If you application has been configured for this, check the **Compare Label Columns** check box to compare differences between column values to be displayed, along with a cell shading indicator.

Also, depending on the application configuration, you may see an **OK** button like the one shown above that applies all of your selection changes to the table as a batch. Otherwise, your table will be updated immediately when you make individual changes.

The Summary Function feature and the Label Column Comparison feature provide interesting ways to analyze the data:



Using a Summary Function



Comparing Label columns using color

As shown above, summarizing the data totals the rows and columns, inserting a row and column to display the results. Comparing Label columns can produce color spectrum backgrounds or directional arrows, and can display the value differences as values or percentages, in the cells.

Hide the crosstab configuration area by clicking the gear icon, or delete the table entirely by clicking the Remove (trash can) icon.

Crosstab Tables are displayed in their own panels, so you can expand and collapse them using their "+" and "-" icons. You can also rearrange the order of chart and table panels by clicking with your mouse near the top of a panel, and dragging it up or down.

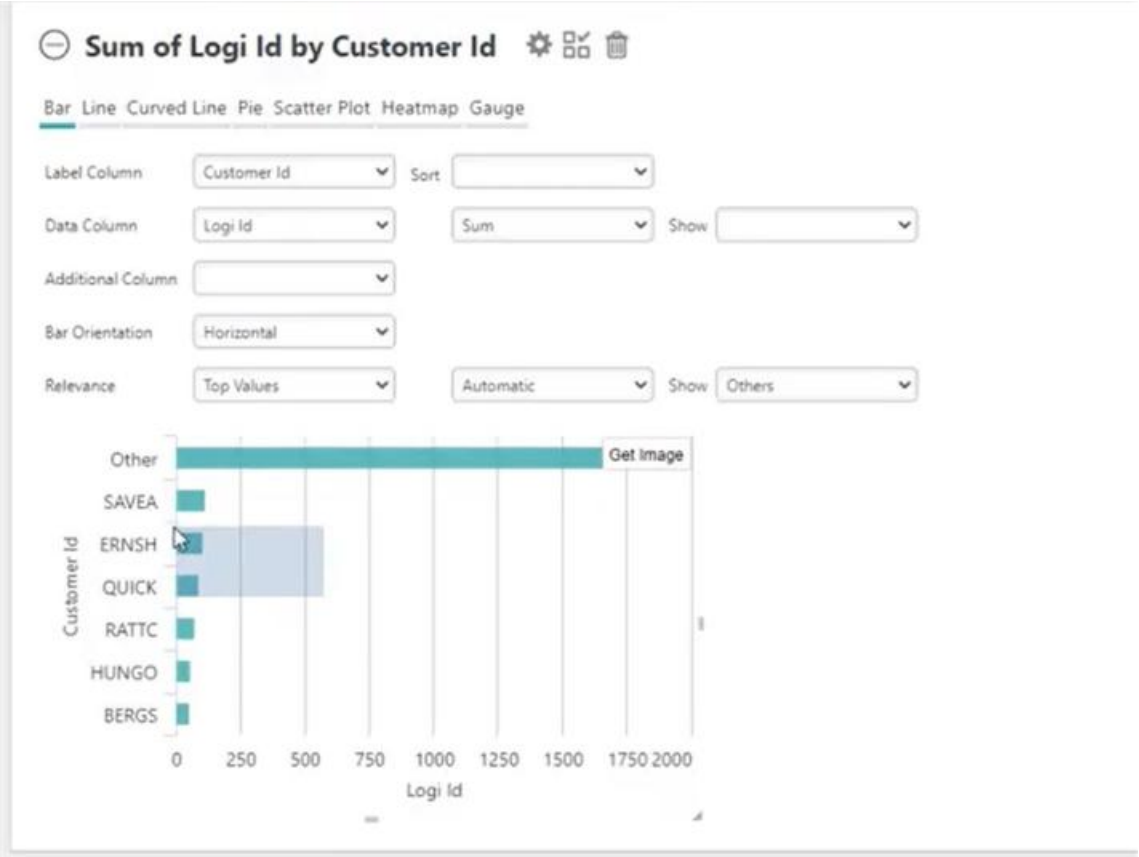
Analysis Grid - Zoom Control

Zoom control enables you to target specific data and zoom in for further analysis. If your application has been configured for it, you can now use zoom filtering on the following charts:

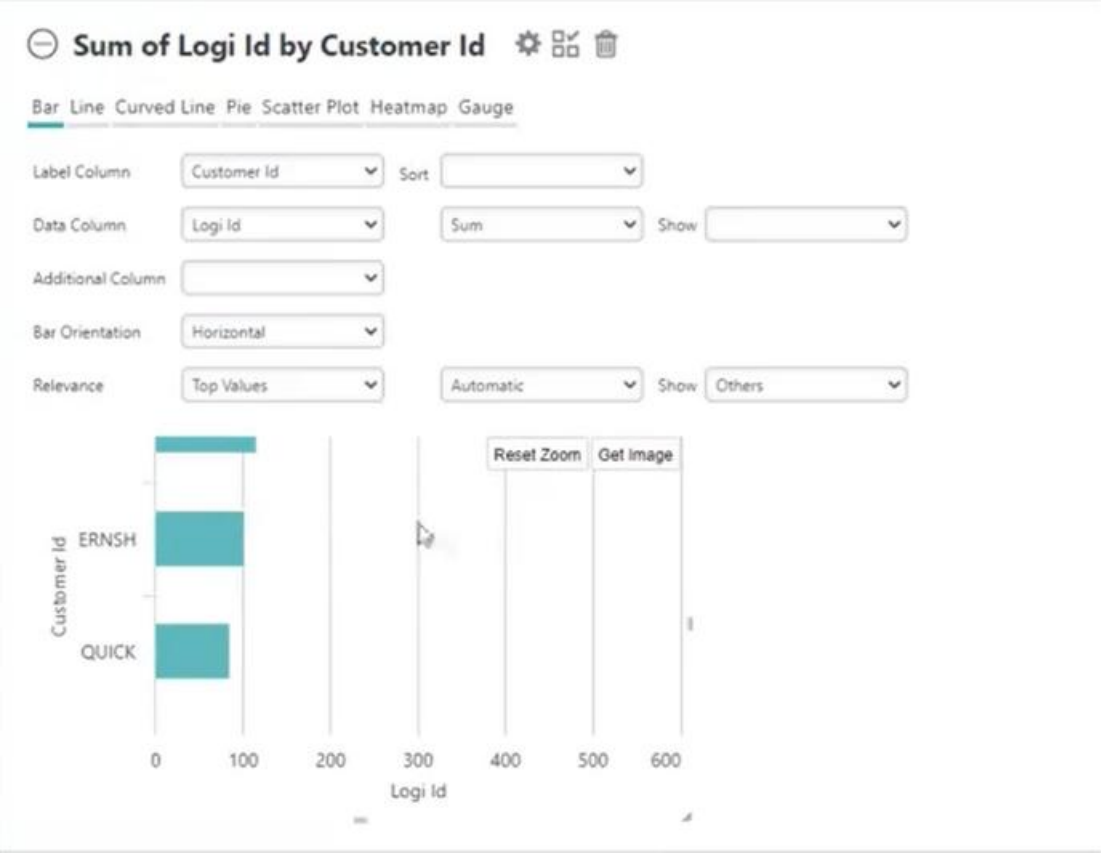
- Bar
- Line
- Curved Line
- Scatter Plot

For information on configuring this feature, see *Analysis Grid Attributes*.

To utilize this feature, simply select the data you want to zoom in on, like below:



Upon releasing your cursor, Info will re-size the chart:

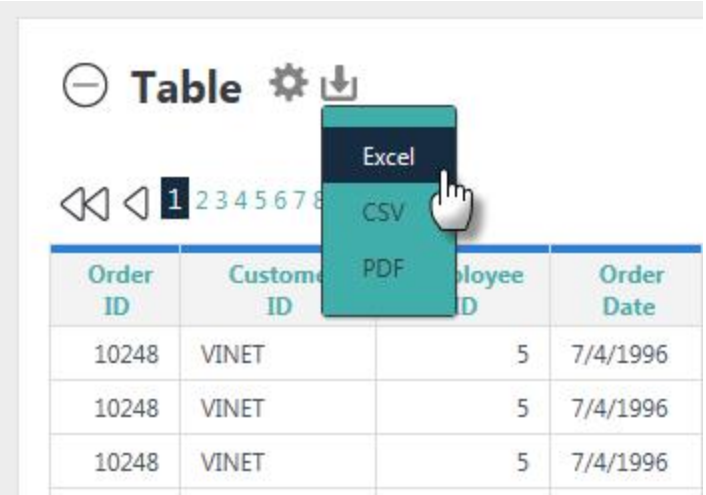


To remove the zoom and return the chart to its original state, select **Reset Zoom**.

Charts with zoom filtering can also be saved to the Dashboard for future analysis. For more information about Dashboards, see "Dashboard for End Users" on page 18.

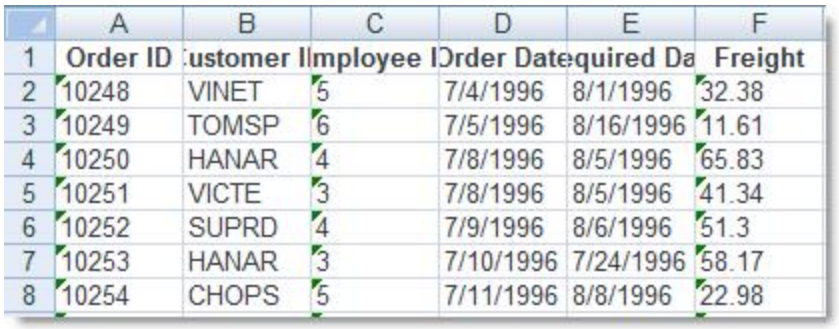
Analysis Grid - Exporting Data

Analysis Grid tables include three export functions, controlled by the **export** icon:



Depending on your application's configuration, you may see some, all, or none of the Export options shown above. They allow you to export the table's data, as follows:

Excel - The data is exported into an .xlsx or .xls file, as raw data. The file can be viewed in Excel (if installed on your computer) or can be saved to your file system. Table column headers are exported into the first row of the Excel worksheet, as shown below, and numbers are exported as text. Unless however, if your application has been configured for it, the report title and filter information display in the first row. Depending on your application's configuration, the data may be formatted and specific worksheet column widths may be set.

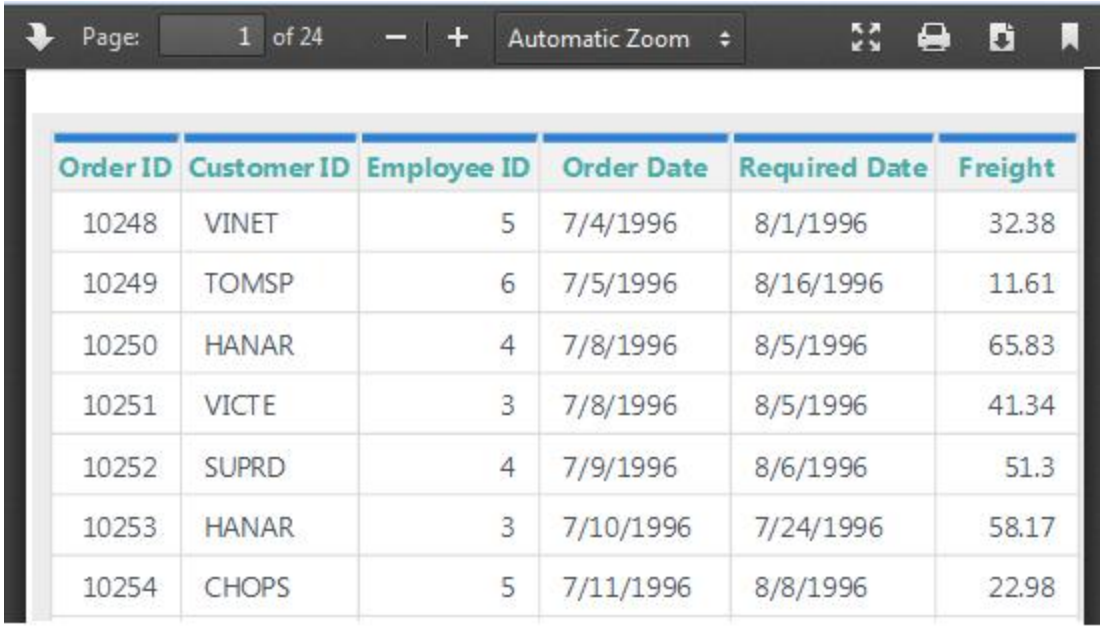


	A	B	C	D	E	F
1	Order ID	ustomer I	mployee I	Order Date	quired Da	Freight
2	10248	VINET	5	7/4/1996	8/1/1996	32.38
3	10249	TOMSP	6	7/5/1996	8/16/1996	11.61
4	10250	HANAR	4	7/8/1996	8/5/1996	65.83
5	10251	VICTE	3	7/8/1996	8/5/1996	41.34
6	10252	SUPRD	4	7/9/1996	8/6/1996	51.3
7	10253	HANAR	3	7/10/1996	7/24/1996	58.17
8	10254	CHOPS	5	7/11/1996	8/8/1996	22.98

CSV - The data is exported into a .csv file, as raw data. The file can be viewed in Notepad (or any text editor) and in Excel (if installed on your computer) or can be saved to your file system. Table column headers are exported into the first row. Unless however, if your application has been configured for it, the report title and filter information display in the first row. All fields are enclosed in double-quotes and separated by commas.

```
"Order ID","Customer ID","Employee ID","Order Date","Required Date","Freight"
"10248","VINET","5","7/4/1996","8/1/1996","32.38"
"10249","TOMSP","6","7/5/1996","8/16/1996","11.61"
"10250","HANAR","4","7/8/1996","8/5/1996","65.83"
"10251","VICTE","3","7/8/1996","8/5/1996","41.34"
"10252","SUPRD","4","7/9/1996","8/6/1996","51.3"
"10253","HANAR","3","7/10/1996","7/24/1996","58.17"
"10254","CHOPS","5","7/11/1996","8/8/1996","22.98"
```

PDF - An image of the table is exported into a temporary .pdf file. This file can be viewed in your browser using the Adobe Acrobat plug-in, similar plug-ins, or, in some cases, native browser technology. Viewers usually let you save the export as a file, if desired, or print it. Table headers will be displayed at the top of each PDF page. Unless however, if your application has been configured for it, the report title and filter information display at the top of the page.



Order ID	Customer ID	Employee ID	Order Date	Required Date	Freight
10248	VINET	5	7/4/1996	8/1/1996	32.38
10249	TOMSP	6	7/5/1996	8/16/1996	11.61
10250	HANAR	4	7/8/1996	8/5/1996	65.83
10251	VICTE	3	7/8/1996	8/5/1996	41.34
10252	SUPRD	4	7/9/1996	8/6/1996	51.3
10253	HANAR	3	7/10/1996	7/24/1996	58.17
10254	CHOPS	5	7/11/1996	8/8/1996	22.98

Export Row Limit

When working with very large data sets, exports may not be practical. To prevent unacceptably long exports, your application may have been configured with a limit on the number of rows that can be exported. If the number of rows in your Table (across all pages) exceeds this limit, the export controls will be disabled. If this is the case, you may be able to reduce the number of rows in the Table and re-enabled the export feature by adding one or more Filters on the data.

Analysis Grid - Adding Analysis Grid Charts to Dashboards

The Analysis Grid includes an optional feature that lets you create a table or chart in one report, then add that chart as a new panel in an existing Dashboard in another report.

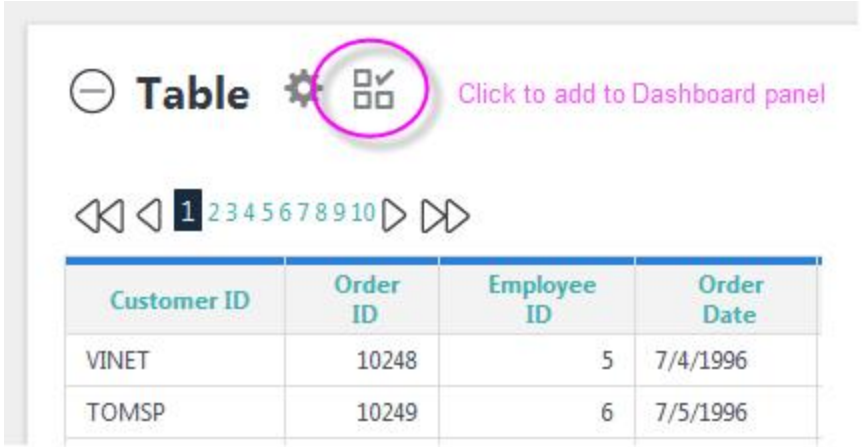


Table
Click to add to Dashboard panel
1 2 3 4 5 6 7 8 9 10
Customer ID
Order ID
Employee ID
Order Date
VINET 10248 5 7/4/1996
TOMSP 10249 6 7/5/1996

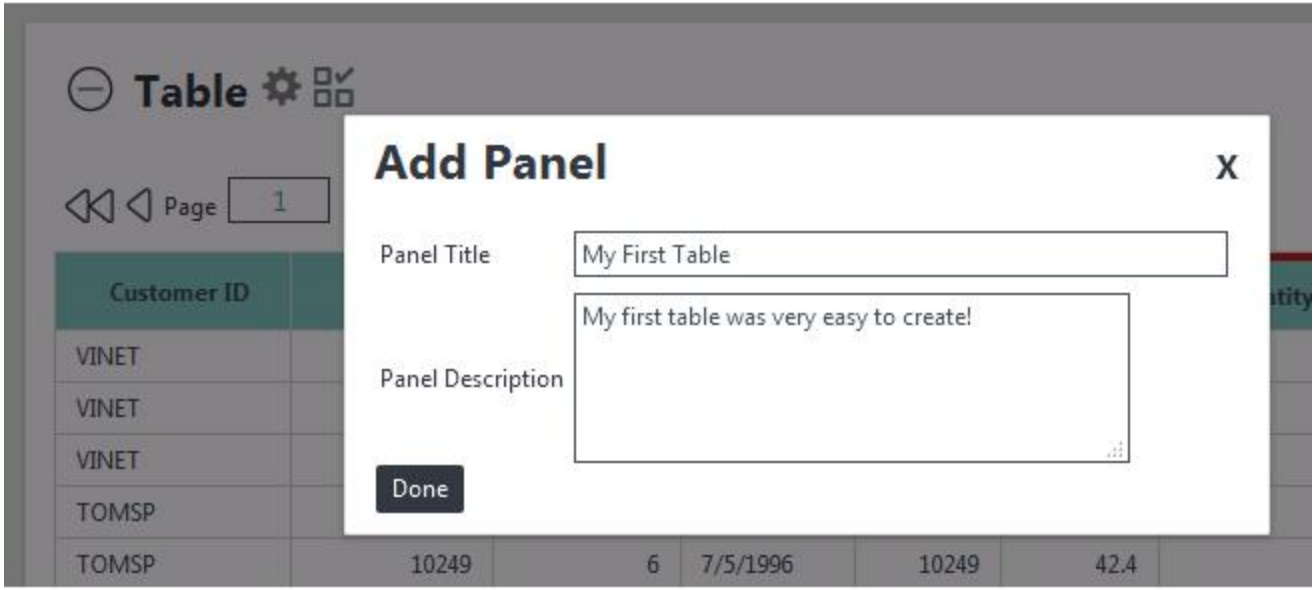


Table
Page 1
Add Panel
X
Panel Title
My First Table
Panel Description
My first table was very easy to create!
Done
Customer ID
VINET
VINET
VINET
TOMSP
TOMSP 10249 6 7/5/1996 10249 42.4

If your application has been configured for this feature, Analysis Grid tables and charts will display an **Add To Dashboard** link, as shown above. When you click the link you'll be prompted for the **Panel Title** and a description for display in the Dashboard Configuration Page.

The new table or chart panel thereafter appears in the Dashboard Configuration Page. You can insert multiple tables and charts into a Dashboard using this technique.

Charts with zoom filtering can also be saved to the Dashboard for future analysis.

For more information about Dashboards, see "Dashboard for End Users" on page 18.

Classic Charts & Gauges

Logi Info offers developers a rich selection of charts for data visualization.

The following topics discusses one of the families of charts and gauges available, Classic Charts:

- Gallery of Classic Chart and Gauge Elements
- Gallery of Deprecated Classic Charts

A number of Classic Charts have been deprecated; they are still supported and will work, but their elements are no longer available in Studio. Whenever possible use Chart Canvas Charts instead for all new development. See the gallery tables below for details.

About the Classic Charting Elements

The container or root element for charts in this family is the **Chart Canvas** element. These charts use HTML5 features and JavaScript. For more information about these charts, see *Chart Canvas Charts*.

We refer to charts and gauges that were existed *prior* to the introduction of Chart Canvas Charts as our "Classic Charts and Gauges". They fall into two categories: *Static* and *Animated*, depending on whether they can draw themselves with animation.

Classic Chart Type	Description
Animated Charts	Classic animated charts and gauges use **HTML5** features and **JavaScript** to render animated visualizations. Flash animation is *not* supported and these charts are *not exportable*.

Classic Chart Type	Description
Static Charts	Static charts and gauges are rendered on the report page as an **image**, so they do not require any special browser add-ins to be viewed and they *are exportable*.

Some individual charting elements are capable of producing several different types of charts. For example, **Chart.XY** produces *Line*, *Bar*, *Area*, and *Spline* charts. These charting elements give you the ability to adjust many aspects of your charts and, as a result, multi-type chart elements have a lot of attributes, but some of them only apply to a particular chart type and not to others.

If you're just beginning to work with Logi chart elements you'll find that it's a very iterative process and there are usually many small adjustments to be made until your charts are "just right". This is inherent in having the level of flexibility that our customers demand.

Logi Studio includes several **wizards** that can help you get started creating charts and gauges, but they generate Chart Canvas Charts, not Classic Charts.

Additional information that will allow developers to quickly get started creating charts is available in "Working with Classic Charts" on page 213.

Gallery of Classic Chart and Gauge Elements

The following table describes the charts that are available:

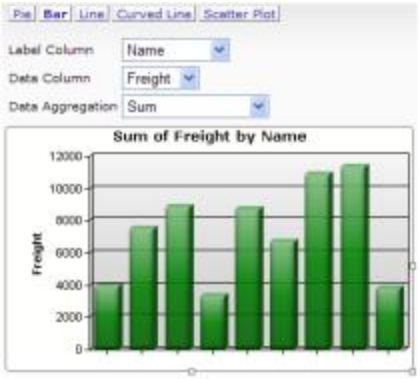
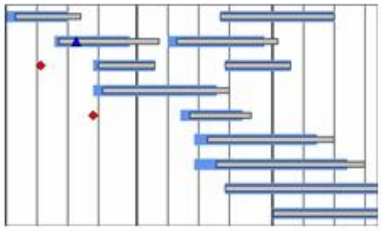
Chart	Description
	Element name: Analysis Chart The Analysis Chart is a **dynamic chart set** that offers users interactive data analysis capabilities. While viewing a report in their browser, users can select chart types and chart data. Chart options include: *Pie*, *Bar*, *Line*, *Spline*, and *Scatter*.
	Element name: Chart.Gantt A **Gantt chart** is a popular type of bar chart that illustrates a **project schedule**. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. Chart.Gantt is limited to 1,000 data rows and only supports one Extra Gantt Column child element.

Chart	Description
	Element name: Gauge.Angular **Gauge.Angular** displays an angular gauge chart, similar to a speedometer. Upper and lower values define the bounds of the gauge, and a value indicates where the needle points. Angular gauges may have labels, tick marks, and differently-colored rings.You can also customize the size of the gauge to be smaller, or larger, within the gauge area by defining a percentage in the "Gauge Size Adjustment" attribute. The attribute "Chart Description" allows you to include a description. Edit the font-related and positioning attributes using the child element "Description Style". v23.1 When the **Annotation** element is used as a child of Gauge.Angular, you can place custom labels and shapes at various points of interest.
	Element name: Gauge.Arc **Arc gauges** are simple, clean visualizations which show a beginning and ending values, with an arched color bar between. The data value number is shown within the arc. The gauge may have multiple ranges in different colors. The color of the arc is determined by which range contains the value. The attribute "Chart Description" allows you to include a description. Edit the font-related and positioning attributes using the child element "Description Style". v23.1 The attribute "Label Class Name" allows you to apply unique styling to your gauge's data labels by referencing a class name from a stylesheet in the report. v23.1 When the **Annotation** element is used as a child of Gauge.Arc, you can place custom labels and shapes at various points of interest.

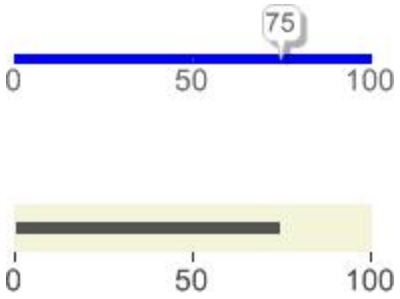
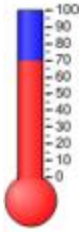
Chart	Description
	Element name: Gauge.Balloon Bar and Gauge.Bullet Bar **Balloon Bar** gauges show a value inside a prominent balloon box, with a pointer going into the bar. **Bullet Bar** gauges show a bar within a bar. These are especially good for presenting key values in an easy-to-read image. The attribute "Chart Description" allows you to include a description. Edit the font-related and positioning attributes using the child element "Description Style". v23.1 The attribute "Balloon Label Class Name" allows you to apply unique styling to your gauge's data labels by referencing a class name from a stylesheet in the report. v23.1 When the **Annotation** element is used as a child of Gauge.Balloon Bar or Gauge.Bullet Bar, you can place custom labels and shapes at various points of interest.
	Element name: Gauge.Bar A single data value is plotted along a **vertical thermometer** scale, providing a quick at-a-glance indication of its value. Alternate visual styles of this chart include *Horizontal Thermometer, Vertical Bar,* and *Horizontal Bar.*

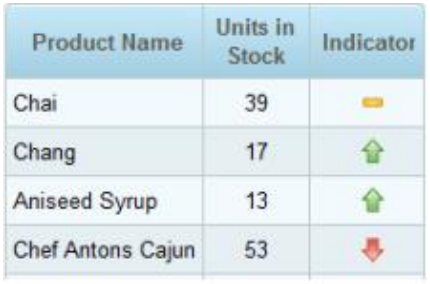
Chart	Description
	Element name: Gauge.Indicator This chart provides **multi-colored signals** associated with a single dataset, providing a quick at-a-glance indication of value. Alternate visual styles of this chart include a choice of standard signal images (*Rectangle*, *Circle, Up Triangle,* or *Down Triangle*) or your own custom images.
	Element name: Gauge.Number Displays a value as number or text. The value automatically resizes to fit the container. The value may be displayed in different colors, typically indicating how the value compares to goals defined by the ranges. v23.1 When the **Annotation** element is used as a child of Gauge.Arc, you can place custom labels and shapes at various points of interest.
	Element name: Sparkline.Area, .Bar, .Line, .WinLoss **Sparklines**are tiny charts appropriate for showing many trends at once, as a set of small timelines. They are typically drawn without axes or gridlines and are often shown inside Data Tables, alongside labels.

Chart	Description
	Element name: Text Cloud This chart creates unique visual depiction of the **relative weightings** of **words**and **phrases**. Greater weightings or frequencies can be denoted through color and font size.

Exporting Gauges as Images

Many gauges can be exported as a .PNG image, including the following:

- Gauge.Arc
- Gauge.Angular
- Gauge.Balloon Bar
- Gauge.Bullet Bar

More information about how to configure them to be exportable can be found in *Export Chart Canvas Charts*.

Gallery of Deprecated Classic Charts

These Classic Charts are still supported and will continue to work, but their elements are no longer available in Studio. Use alternate elements as indicated.

Chart	Description
  	Element name: Animated Chart.Pie Deprecated - use Chart Canvas instead. This Flash-based chart "draws itself" when the page is displayed or refreshed. A pie chart (or circle graph) is a **circular chart**divided into sectors, illustrating relative magnitudes, frequencies, or percentages. It's named for its resemblance to a pie which has been sliced. Alternate visual styles of this chart include *exploded pie wedges* and the *Doughnut* shape.

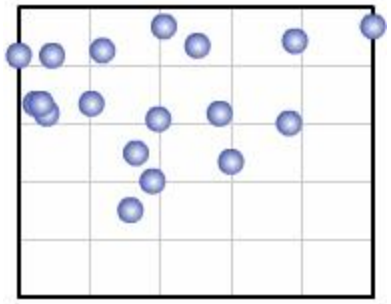
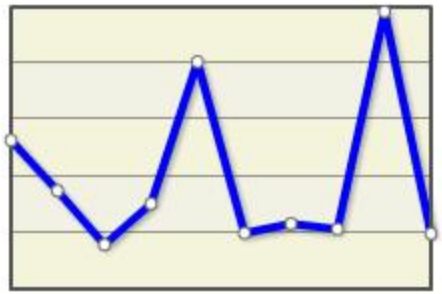
Chart	Description
	Element name: Animated Chart.Scatter Deprecated - use Chart Canvas instead. This Flash-based chart "draws itself" when the page is first displayed. A scatter chart uses **Cartesian coordinates** to display values for two variables from a dataset. The data is displayed as a **collection of points**, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis.
 	Element name: Animated Chart.XY Deprecated - use Chart Canvas instead. This Flash-based chart "draws itself" when the page is first displayed and is capable of producing **Line**, **Bar**, and **Area** charts. Cartesian coordinates are used to display values for two variables from a dataset and plot them as points on the chart. These points are used to create the visual representation of the data. Alternate visual styles of this chart include *Bar*, *Area*, and 3-Dimensional renderings.

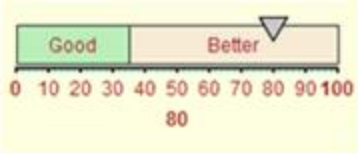
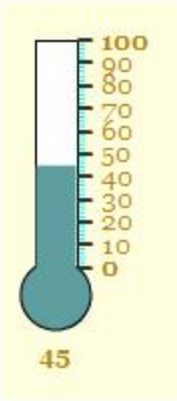
Chart	Description
 	Element name: Animated Gauge.Bar Deprecated - use Gauge.Balloon Bar, Gauge.Bullet Bar, or Gauge.Bar instead. This Flash-based chart "draws itself" when the page is first displayed. A single data value is plotted along a either a **Horizontal Bar** or a **Vertical Thermometer** scale, providing a quick at-a-glance indication of its value.

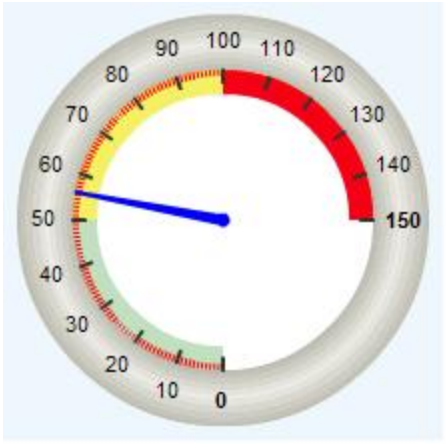
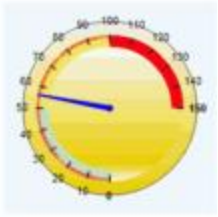
Chart	Description
  	Element name: Animated Gauge.Needle Deprecated - use Gauge.Angular instead. This Flash-based chart "draws itself" when the page is first displayed. This chart resembles an **automotive speedometer** or tachometer and provides an eye-catching data visualization. Alternate visual styles of this chart include *Bevel* and *Lens*.

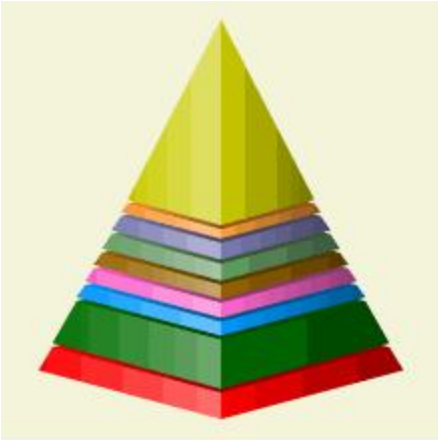
Chart	Description
 	Element name: Animated Gauge.Pyramid Deprecated - use Chart Canvas instead. This Flash-based chart "draws itself" when the page is first displayed. This chart provides a striking comparison of data, using the **thickness** of its **layers** to denote relative values. An alternate visual style for this chart is the *Funnel* shape.

Chart	Description
	Element name: Chart.Heat Map Deprecated - use Chart Canvas instead. This element produces one of the most unusual charts found in Logi reporting products, with a unique arrangement of **rectangles** representing data and relationships using **color** and **size**. It produces an image-based chart that can be exported.
 	Element name: Chart.Pie Deprecated - use Chart Canvas instead. A pie chart (or circle graph) is a **circular chart** divided into sectors, illustrating relative magnitudes, frequencies, or percentages. It is named for its resemblance to a pie which has been sliced. Alternate visual styles of this chart include *2-Dimensional*, *exploded pie wedges*, and *2-D* and *3-D doughnuts*.

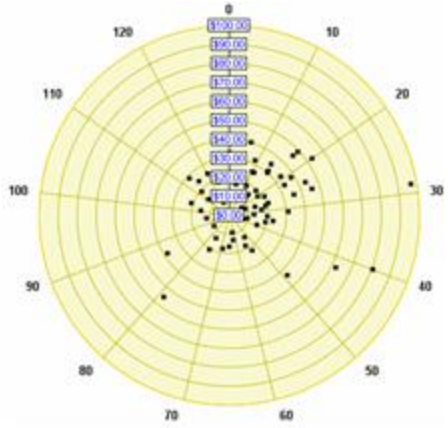
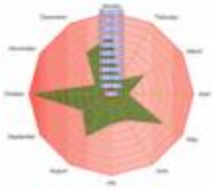
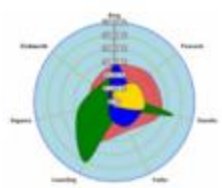
Chart	Description
  	Element name: Chart.Polar Deprecated - use Chart Canvas instead. This a circular graph in which data is displayed in terms of **values** and **angles**. Also known as a "radar chart", this is an X-Y graph drawn on a circular grid, displaying value **trends**on the basis of **angles**. Alternate visual styles of this chart include *Area,Spline Area, Scatter, Bubble, Spline Line, Line,* and *Radar*.

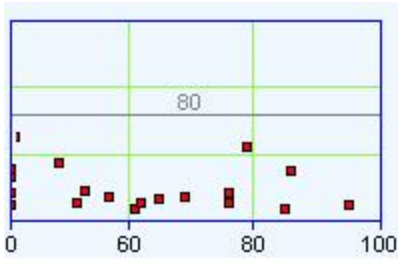
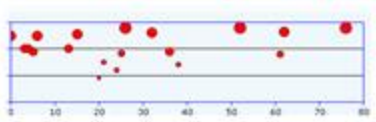
Chart	Description
 	Element name: Chart.Scatter Deprecated - use Chart Canvas instead. A scatter chart uses **Cartesian coordinates** to display values for two variables from a dataset. The data is displayed as a collection of **points**, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis. Alternate visual styles of this chart includes point symbols that are *circles, crosses, diamonds, triangles,* or *"X"s*.

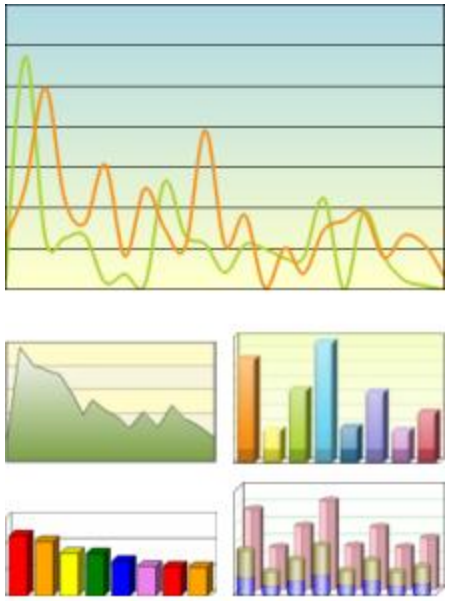
Chart	Description
	Element name: Chart.XY Deprecated - use Chart Canvas instead. This multi-faceted element can be used to product a wide variety of charts. **Cartesian coordinates** are used to display values for two variables from a dataset and plot them as points on the chart. In a **Line** chart, a line connects these points to create the visual representation of the data. **Multiple datasets** can be combined in one chart, using the same or different visual representations. Alternate visual styles of this chart include *Area, Bar, Spline, Stacked,* and *3-Dimensional*.

Chart	Description
 	Element name: Gauge.Needle Deprecated - use Gauge.Angular instead. This chart resembles an **automotive speedometer** or tachometer and provides an eye-catching data visualization, plotting a single data value on its perimeter. *Round*, *half-round, quarter-round,* and *rectangular* formats are available.

Working with Classic Charts

Logi Info includes a family of legacy **Chart** and **Gauge** visualization elements, now collectively called "Classic Charting elements", for use in reports.

The following topics discuss the use of those elements and provides guidance for implementing them:

About Working with Classic Charts

Logi Info includes a set of JavaScript-based charting elements collectively called "Chart Canvas" charts. We recommend that you use them for any new applications you create or if you update existing applications. Wizards in Logi Studio that help you create charts will create Chart Canvas charts. For more information about these chart, see *Chart Canvas Charts*.

A number of Classic Charts have been deprecated; they are still supported and will work, but their elements are no longer available in Studio. For more information, see "Classic Charts & Gauges" on page 196 and Chart Canvas Charts.

Logi Info's Classic Chart and Gauge elements provide powerful features for creating visualizations. The information presented in this topic is intended to allow you to quickly gain an understanding of how Logi charting elements work and to rapidly create simple charts.

Classic charts and gauges fall into two categories: *Static* and *Animated*, depending on whether they can draw themselves with animation.

Classic Chart Type	Description
Animated Charts	Classic animated charts and gauges use **HTML5** features and **JavaScript** to render animated visualizations. Flash animation is *not* supported and these charts are *not exportable*.
Static Charts	Static charts and gauges are rendered on the report page as an **image**, so they do not require any special browser add-ins to be viewed and they *areexportable*.

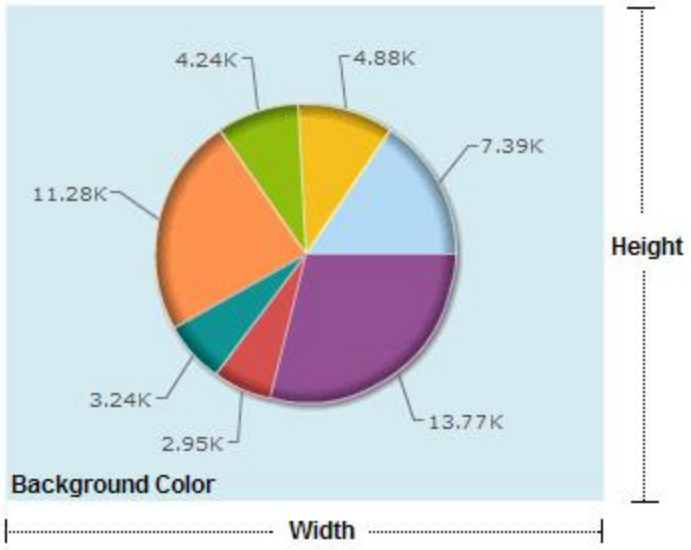
Due to the varying nature of the elements, the attributes used in examples below are not *all* available for *all* charting elements. Attributes and child "helper" elements may be quite chart-style specific; developers need to apply common sense when implementing charts based on the examples shown here.

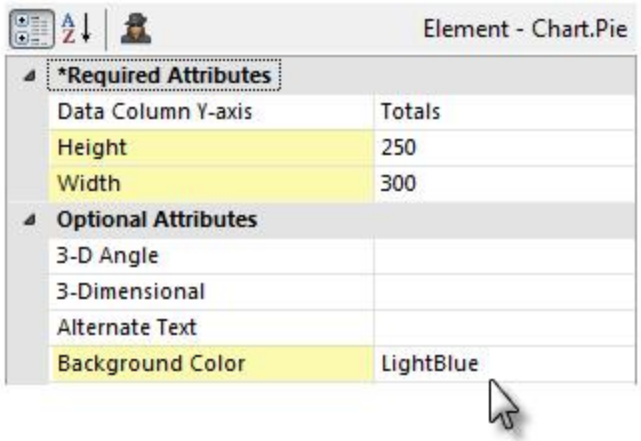
Developers unfamiliar with Logi Classic Chart elements are encouraged to read "Classic Charts & Gauges" on page 196 before proceeding.

Due to the large number of attributes available for Classic Charts, in the examples that follow the Attributes Panel screenshots have been edited to show only the relevant attributes in order to conserve space.

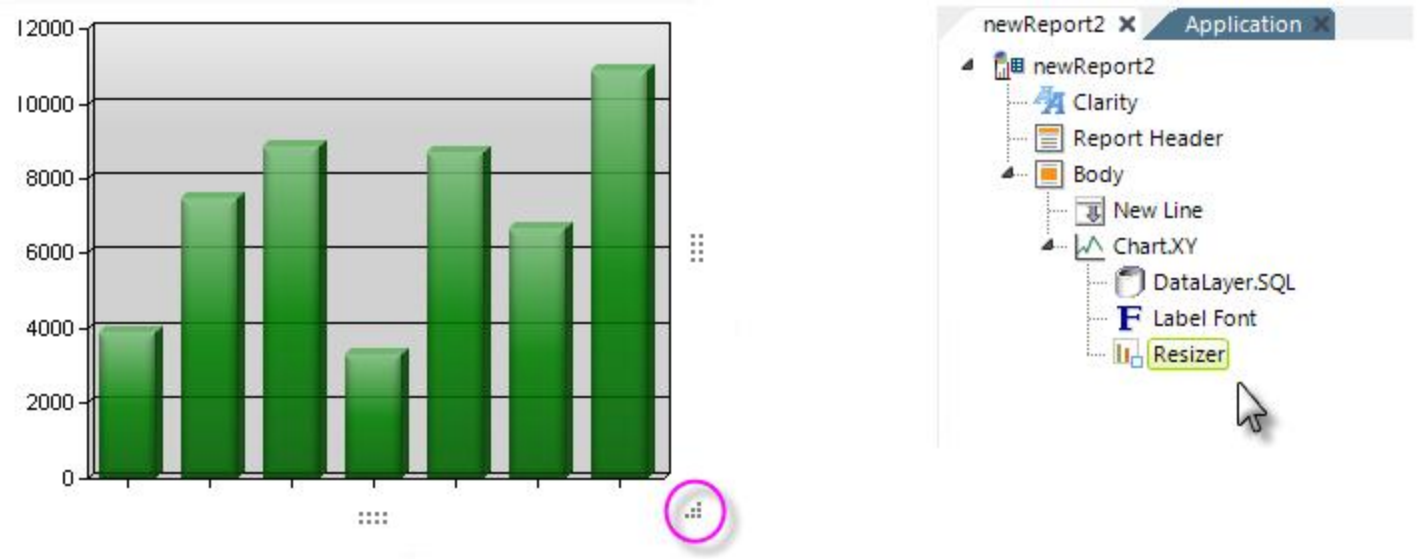
General Chart Appearance

When reports are generated, each charting element is rendered in an "image space" and the elements have attributes that allow developers to control the *size* and *background color* of that space. The shape of the image space may not match the shape of the data portion of the chart.



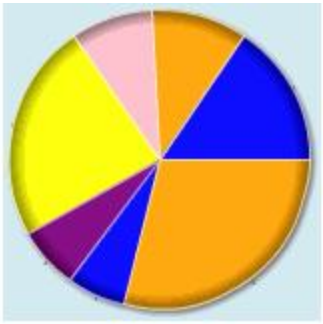


For example, as shown above, even though the chart is round, the **Height**, **Width**, and **Background Color** attributes refer to its rectangular image space. The Height and Width values use *pixels* as their unit of measurement and can be set using tokens.

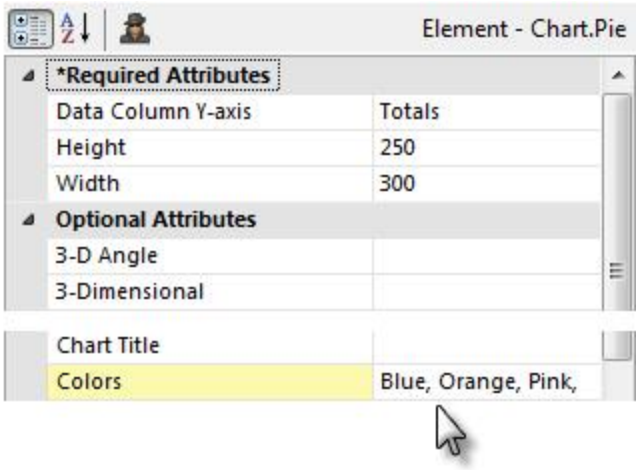


The **Resizer** element can be added as a child of Static charts. It causes "resizing handles", circled above, to be displayed when the cursor hovers over them. Users can drag these handles at runtime to *dynamically resize* the chart. Resizer changes are maintained between sessions, so the chart will maintain its size on subsequent page visits. You can specify maximum and minimum size limits for the chart.

Default Colors



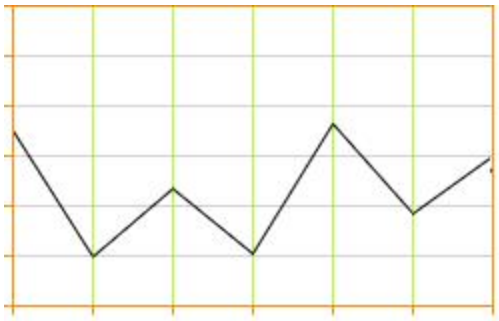
Custom Colors

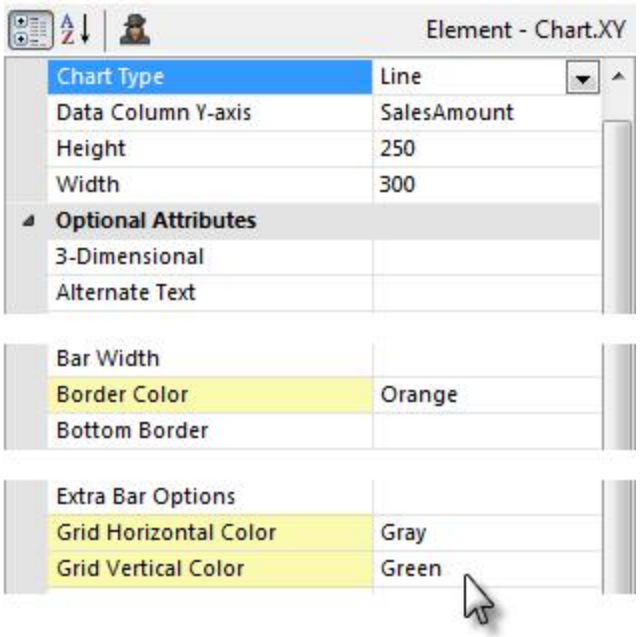


Many charts have a **Colors** attribute for setting the colors of data portions of the chart. If this attribute is left blank, a default color set is used. A custom color set can be entered, in a comma-separated list, as shown above.

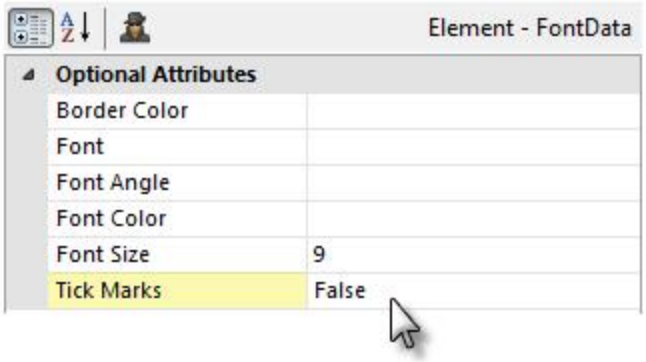
Color values may be entered as a name ("LightBlue"), a decimal RGB value (255,255,255) or a hex RGB value (#112233).

Chart.Pie and Chart.XY include a "Color Data Column" attribute that identifies a column in the datalayer as the source for the color information for static pie and bar charts, allowing you to set the colors from data.

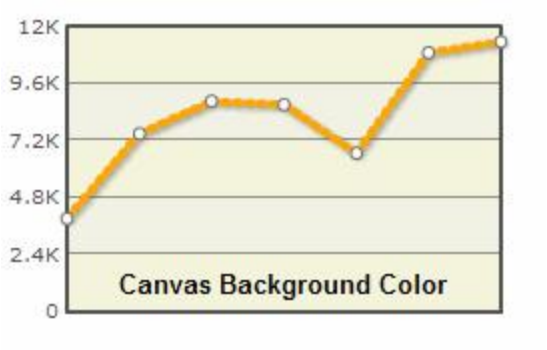


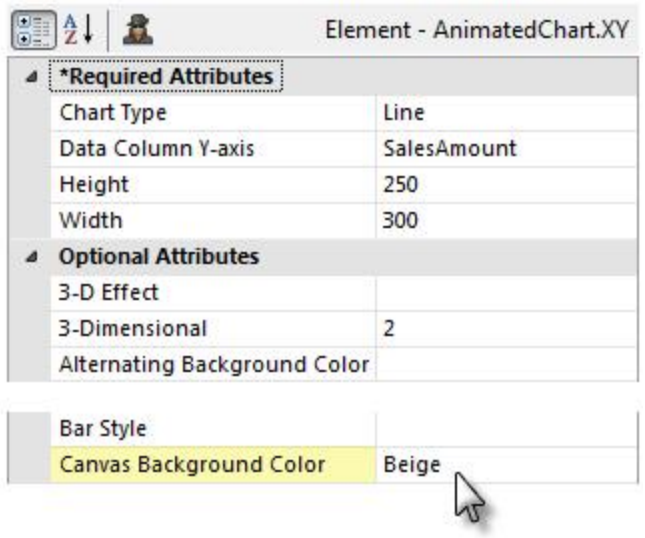


Line, Area, Bar and other charts can use a background grid of lines to help the eye follow values, as shown above. The color of the chart area *border* and *grid lines* can be set. Color values may be entered as a name ("Orange"), a decimal RGB value (255,255,255) or a hex RGB value (#112233).

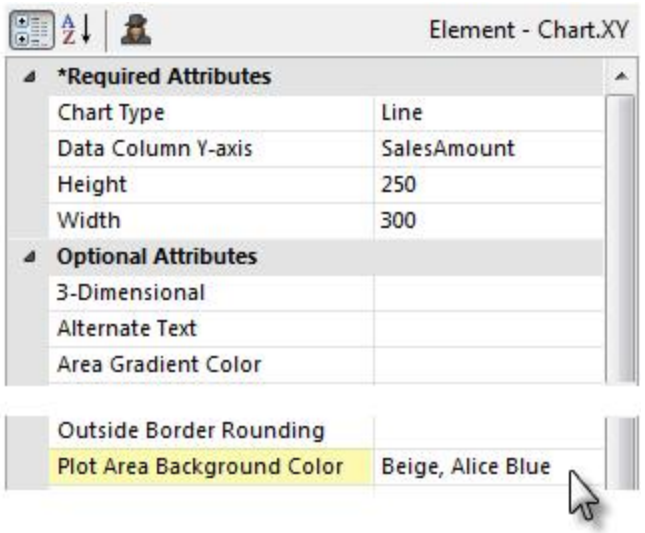


The tick marks displayed just outside the axis borders in Static Chart.XY charts can be shown or hidden, by setting the **Tick Marks** attribute in the **Data Font** and **Label Font** elements.

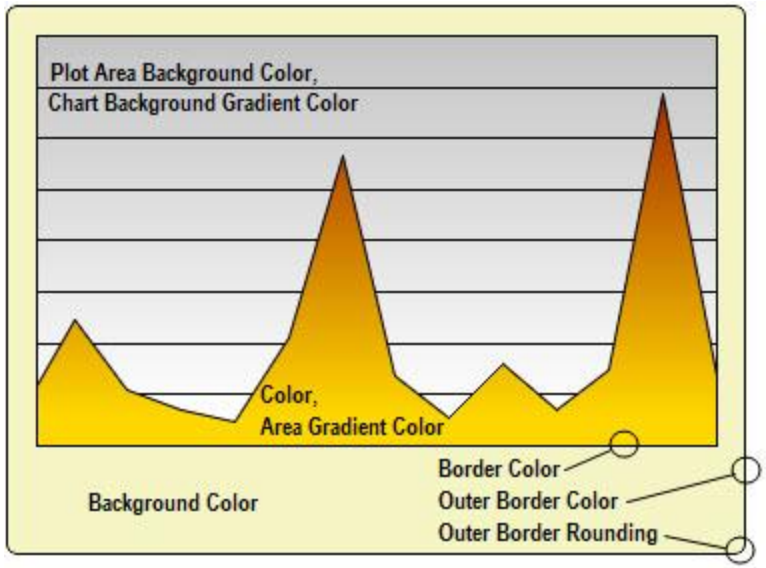


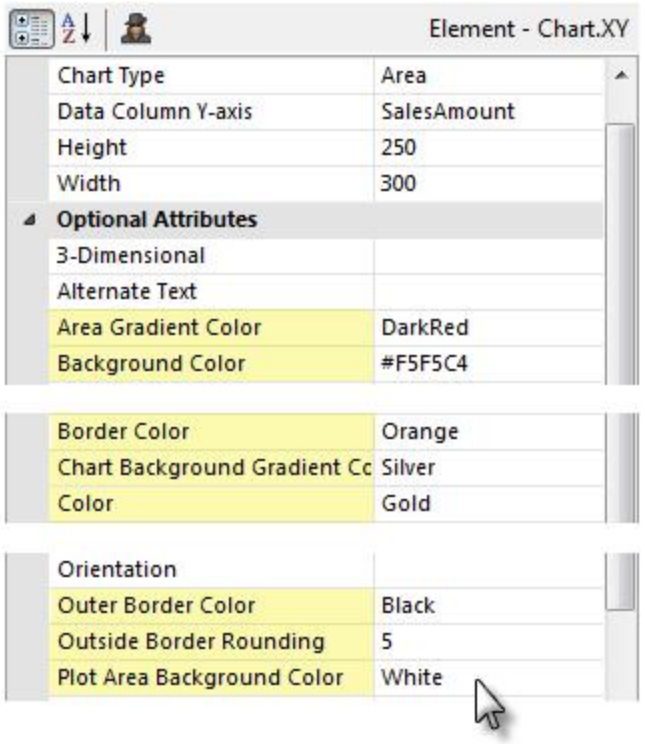


The data portions of **Animated Chart.XY** charts are rendered on a "canvas", which can have its own background color, as shown above, which is set using the **Canvas Background Color** attribute. A compatible color is automatically generated for alternating values.

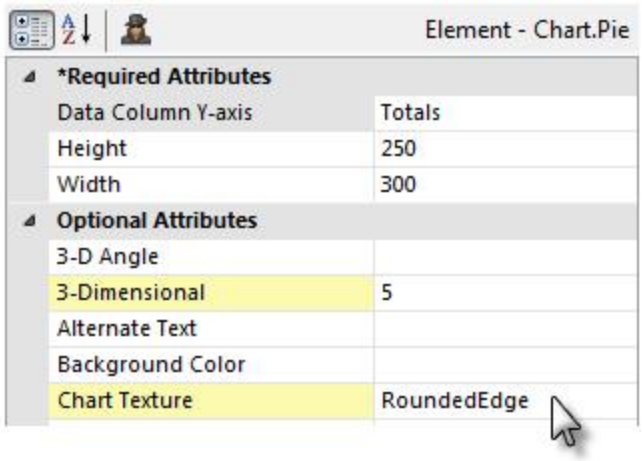


In a Static Line chart, shown above, the attribute that controls the canvas color is the **Plot Area Background Color**. Different colors for alternating values can be shown by entering a comma-separate list of values.

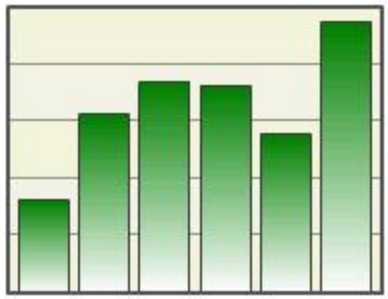




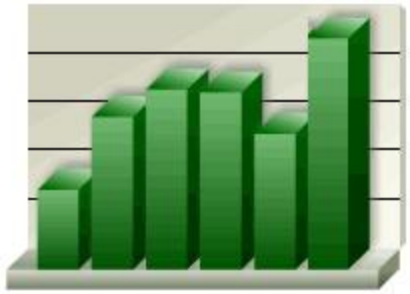
In Static Chart.XY charts (Area, Bar, Line) individual attributes control the colors of the data, plotting area background, and borders. There are also attributes for applying color gradient effects and for rounding outer border corners.



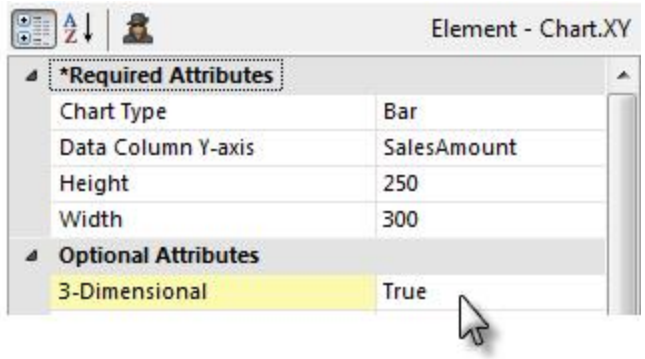
Static Chart.Pie charts include **3-Dimensional** and **Chart Texture** attributes that can be used to apply different appearances to the chart surface. 3-D depth and 3-D angle can also be configured. The Chart Texture attribute is also available for the Chart.Heat-map element, as well.



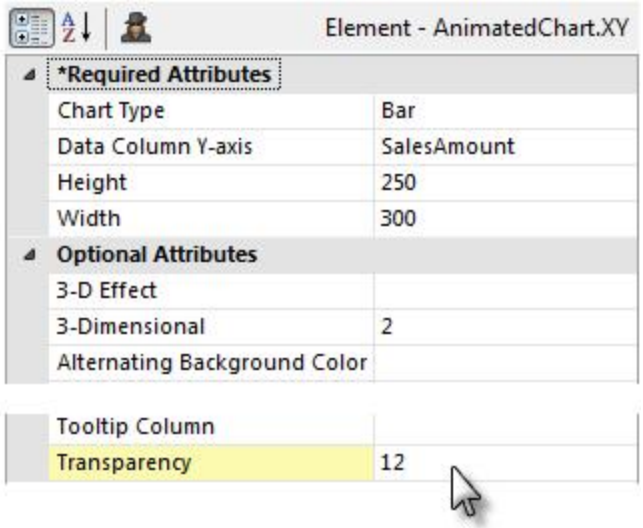
3-Dimensional = False



3-Dimensional = True



Many charts also have a **3-Dimensional** attribute which controls rendering in 2-D or 3-D, as shown above. The 3-D rendering process affects not only the data portions of the chart but the canvas, too. In the case of Gauge.Needle, the **Needle Gauge Enclosure** attribute can even add a very realistic-looking "glass cover" over the face of the gauge.



As shown above, many charts also have a **Transparency** attribute, which can be useful for letting different chart layers show through each other. The lowest value for this attribute is 0 (or blank), which makes the chart opaque, and the highest is 15, which makes the chart completely transparent.

This *does not* affect the canvas itself.

Saving Charts

All charts can be saved as image files by right-clicking them and selecting **Save** from the pop-up menu. Animated Pie charts have additional runtime features available via their pop-up menu, including **Rotation** and **Slicing** (clicking a slice to explode it out

from the pie).

Data Retrieval and Appearance

Data is provided for charts by adding a datalayer element as a child of the chart or gauge element. The datalayer retrieves the data to be visualized and supplies it, record by record, to the chart for plotting. The datalayer can make use of grouping, filtering, and all the other modifier elements generally available to datalayers in order to shape the data it retrieves. *The Formatted Column* element can be particularly useful in getting data in the datalayer formatted correctly for use in charts.

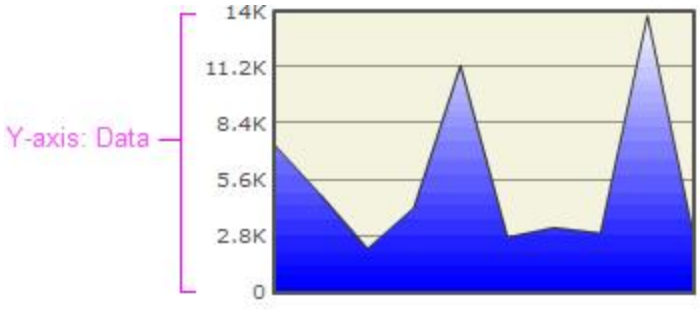

Animated Gauges can get their data from a datalayer or from tokens, including Local Data, Session, and Request tokens. However, they're not designed to be used in a report page where there are *multiple* gauges, each with its own child datalayer. In this case, all of the gauges will draw data only from the *first* child datalayer. If you have a multi-gauge page such as this in mind, design your report to use Local Data to retrieve *all* of the data needed for *all* of the gauges, then use tokens to provide the data to the gauges.

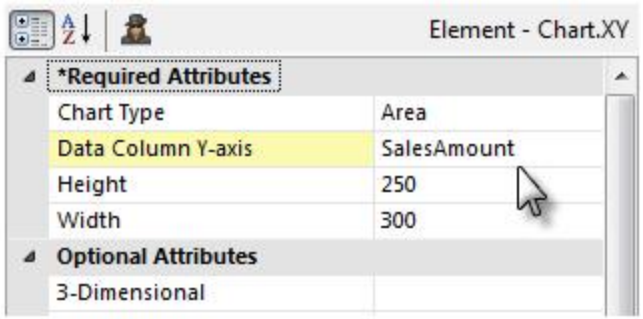
For Static charts, a **Show Wait Icon** attribute controls whether an animated "loading" image is displayed if there's a delay in retrieving data for a chart. It defaults to *True*, so set it to *False* if you don't want the animated icon to appear.

Chart processing is multi-threaded and asynchronous so, for example, charts in multiple Dashboard panels will be generated simultaneously. On web servers with multiple cores or multiple processors, chart generation tasks will be spread across them for even better performance.

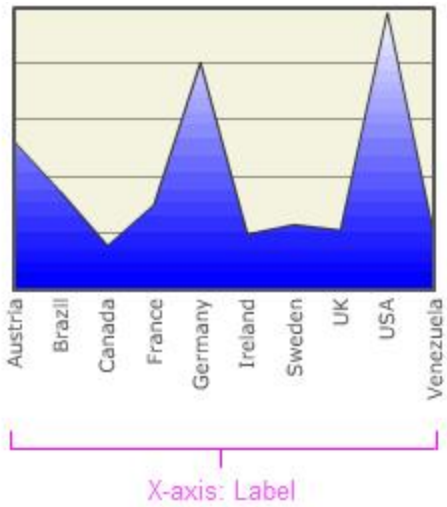
What if no data is returned to the datalayer? Instead of displaying the chart, Animated charts will automatically generate a "No data to display message". When using Static charts, you can choose to add a **Zero Rows Message** element to display a message.

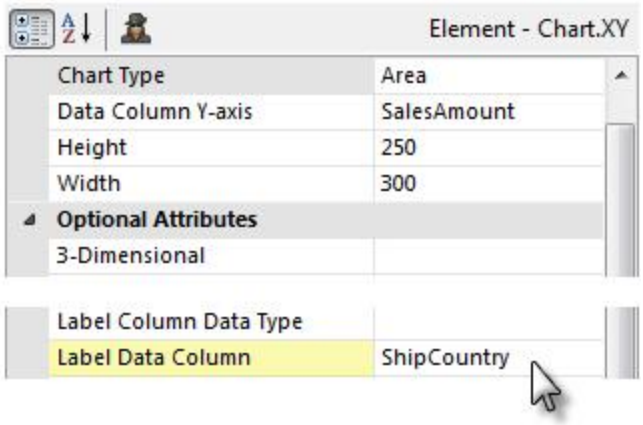
Values from the datalayer can be used to set the boundaries of graphics in the chart (the height of a bar, the size of a pie slice, etc.) and to supply text that appears in tooltips, legends, and chart labels.



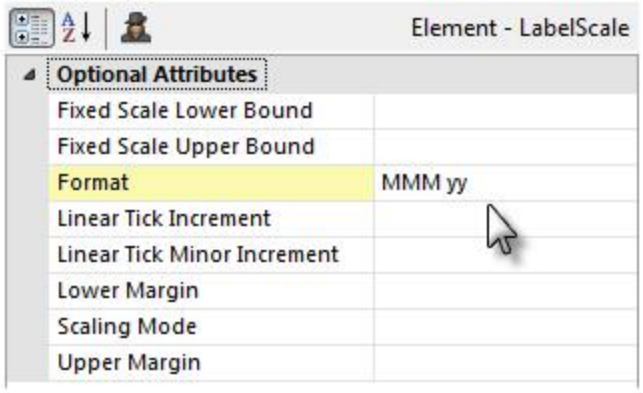


The *name* of the datalayer column that provides a chart's Y-axis ("Data") values is specified in the element's **Data Column Y-axis** attribute, as shown above. This adds the initial data series to the chart.





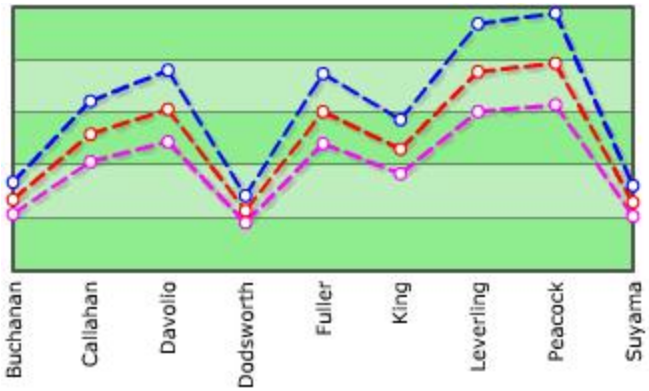
The *name* of the datalayer column that provides a chart's X-axis ("Label") values is specified in the element's **Label Data Column** attribute, as shown above.

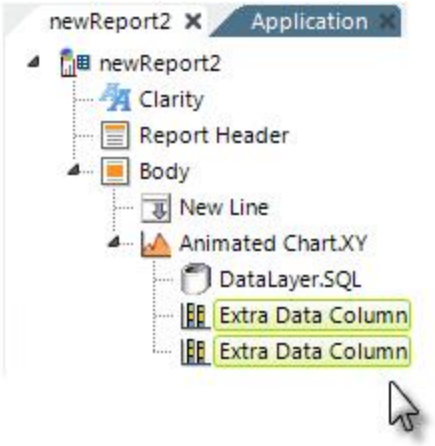


For Chart.Pie, Chart.Polar, and Chart.XY, the chart's Data values can be formatted for presentation, using the **Data Scale** and **Format Data** elements. For Label values, the **Label Scale** element provides similar functionality, as shown above.

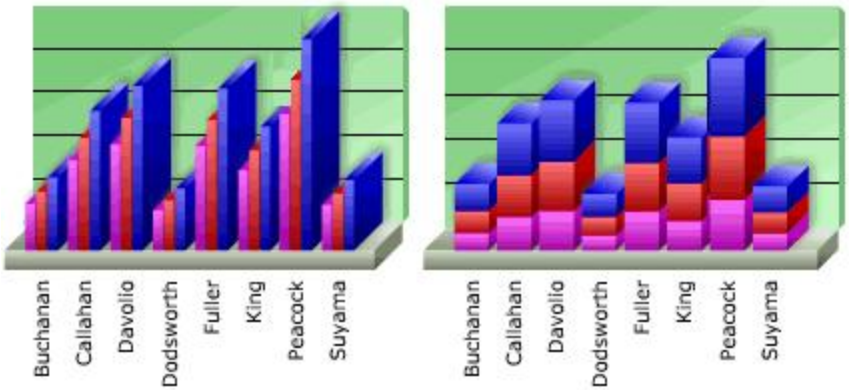
Adding Multiple Series

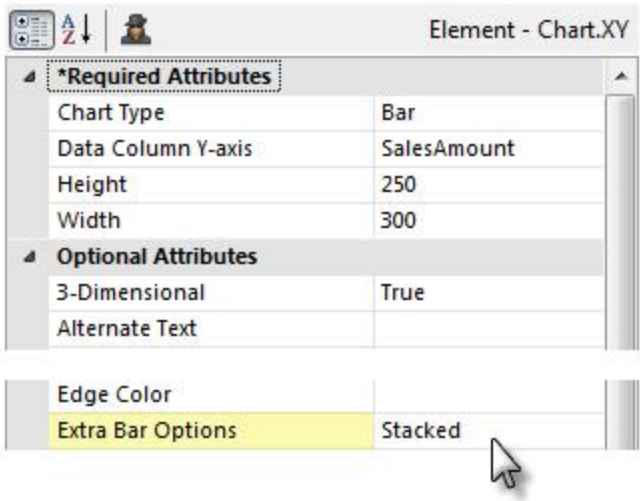
Multiple data series can be plotted on the same canvas, as shown in the following examples.



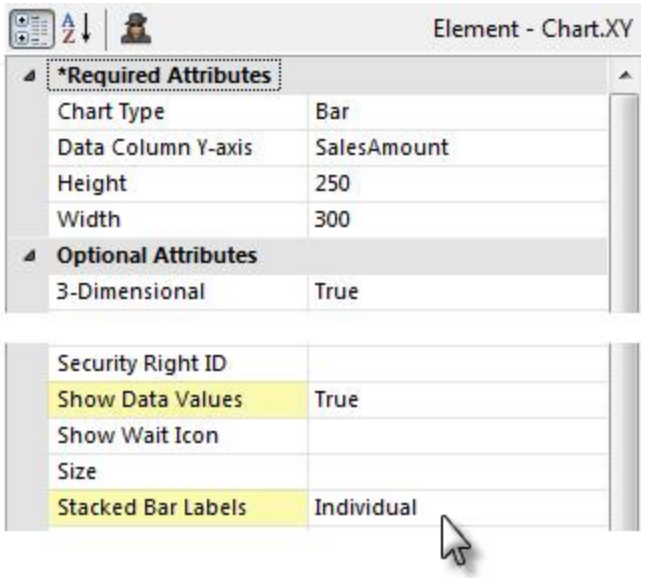


In the example above, *three* data series are shown in a Line chart. The basic chart will display one data series; to add additional series, **Extra Data Column** elements are added below the chart element. Each of these elements has an attribute that's set to the name of another column in the datalayer.

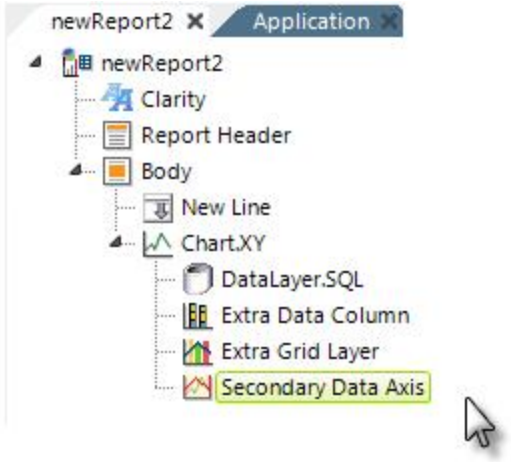




In the example above, the chart type has been changed to a Bar chart. On the left, the three data series appear as *Side-by-Side* bars, and in the middle, as *Stacked* bars. This behavior is controlled by the chart's **Extra Bar Options** attribute, shown above right.



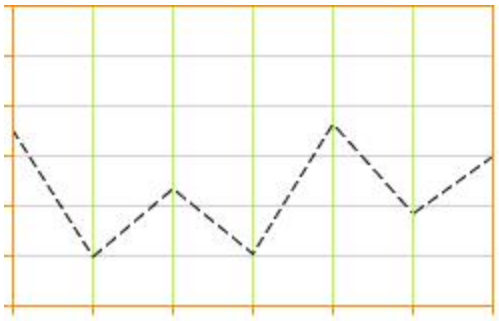
You can show the data values *inside* stacked bars in a Static Bar chart by setting the chart's **Stacked Bar Labels** attribute to *Individual*, as shown above. The default option, *Aggregate*, displays a single, combined value above the top of each stack.

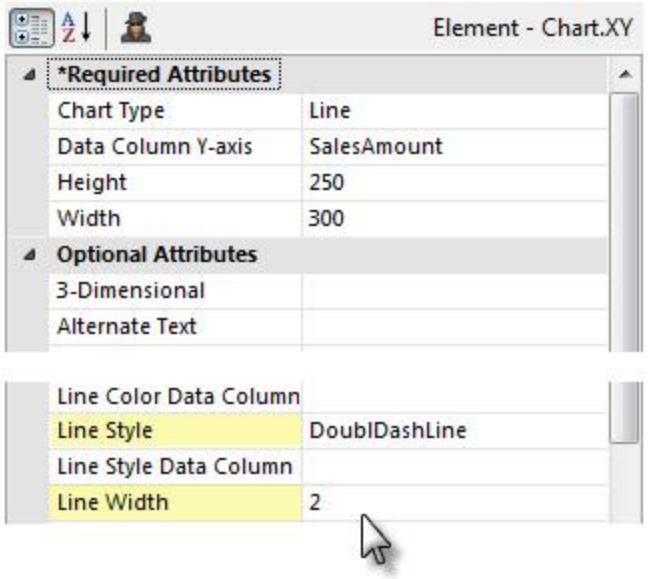


Static Chart.XY charts can have two Y-axes. The **Secondary Data Axis** element controls a secondary axis, normally appearing on the *right* side of a grid. In the example shown above, this is the "Orders" axis. This other axis represents data for an Extra Grid Layer, and has its own independent scale. Child elements can be used to format the data and secondary axis appearance.

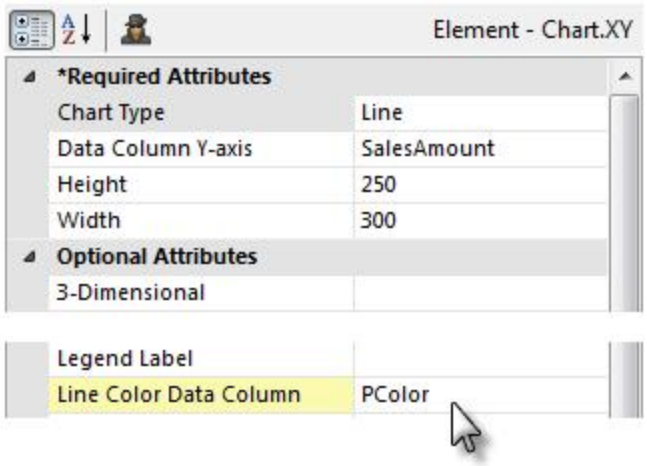
Controlling Data Appearance

The appearance of the data visualization, including colors, widths, and style, can be configured.

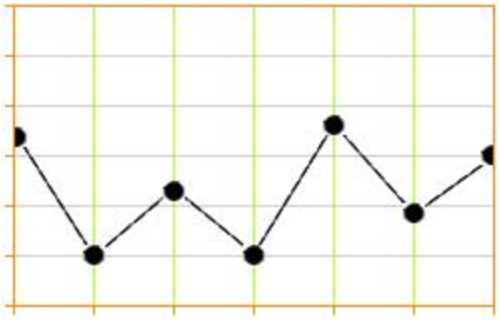


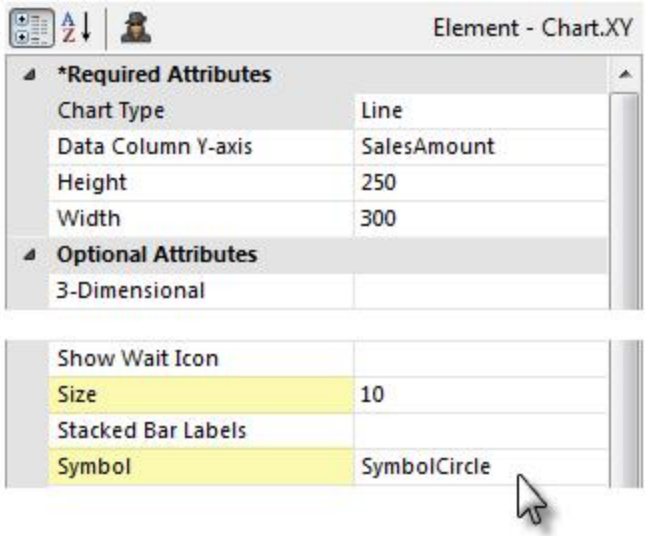


The style and width of the lines representing data in Line charts can be formatted using the **Line Style** and **Line Width** attributes, as shown above. Solid, and a variety of dotted and dashed, line styles are available.

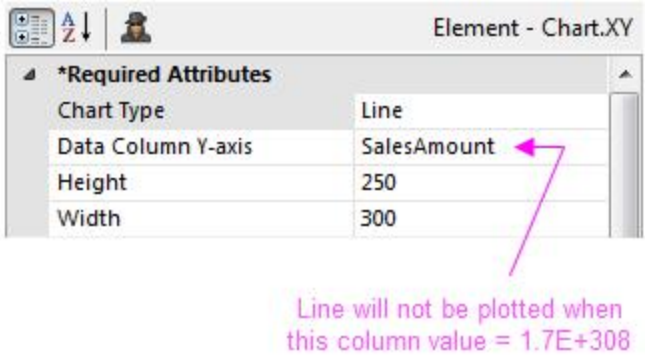


The appearance of static line charts can also be data-driven by setting the **Line Color Data Column** and **Line Style Data Column** attributes. This allows you to have different colors for different line segments, as shown above.

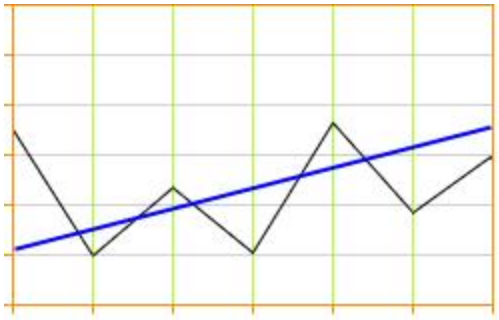


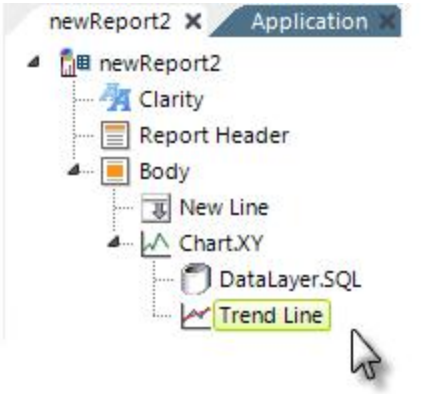


In Line and Scatter charts, the data points can be emphasized with **symbols**, as shown above. The *size* of the symbol, in pixels, can be set and symbols *choices* include Circle, Cross, Diamond, Glass Sphere, Solid Sphere, Square, Triangle, X and others. You can enter a comma-separated list of symbols as the Symbol attribute value for Line and Spline-types of Chart.XY. This allows a different symbol to be used for each data series

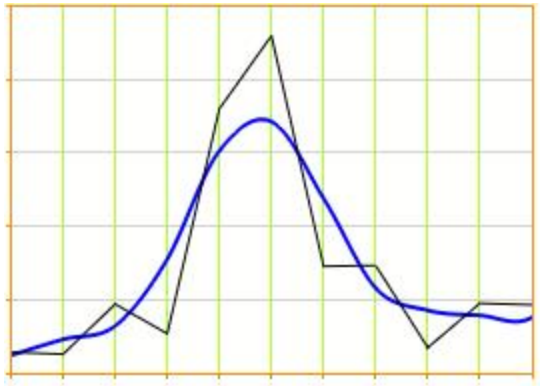


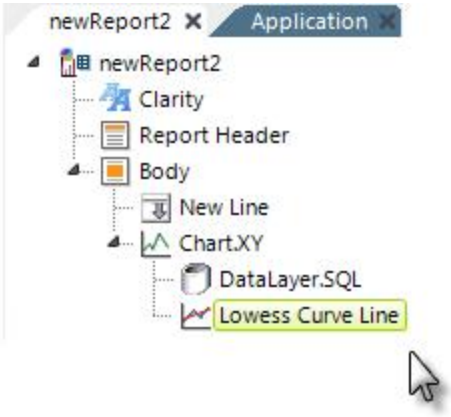
Line chart lines can be made *discontinuous* by replacing null data values with the value **1.7E+308**, producing "broken" lines.



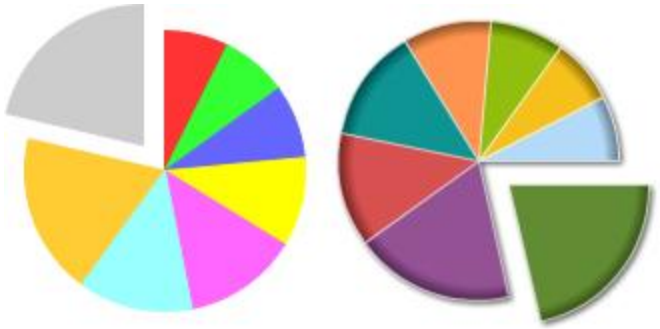


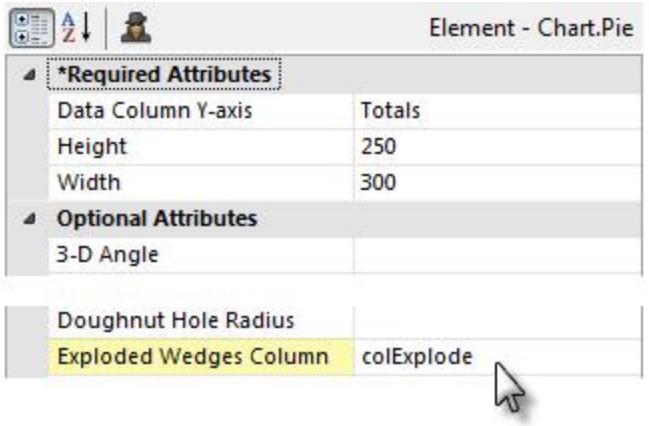
A "trend" line can be added to static reports to give a general impression of the trend of the data. The blue trend line in the example above was included by adding a **Trend Line** element as a child of the chart element. The calculation involved in determining the trend line angle occurs automatically; line width and color can be set through attributes.



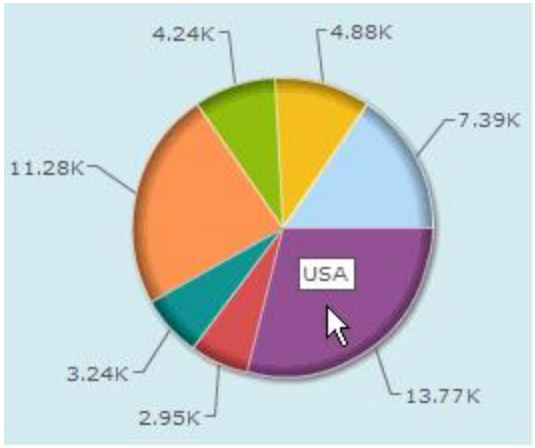


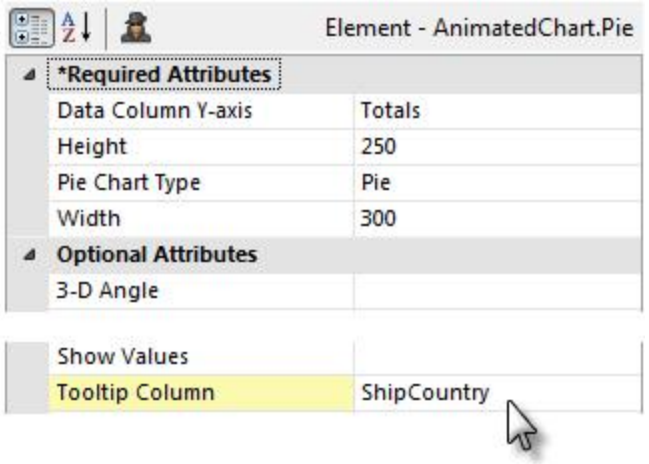
The **Lowess Curve Line** element, available with release 9.5.78, creates a trend line (in blue, above) that fits a curve through a static Chart.XY's data points using the Lowess Curve Fitting algorithm.





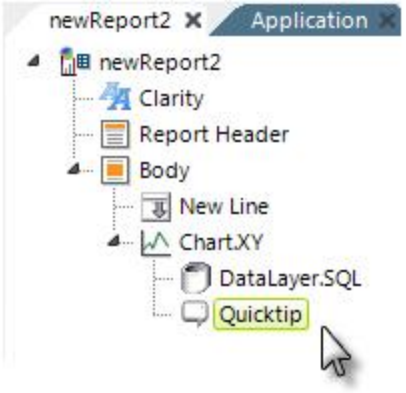
Static and Animated Pie charts can be configured so that one or more of their pie wedges are displayed as "exploded", as shown above. This is done using the **Exploded Wedges Column** attribute, which specifies a datalayer column whose values determine wedge state. Non-zero values in that column will cause the wedge for the corresponding X-axis value to be exploded. Typically, the developer will add a **Calculated Column** to the datalayer, with a formula that evaluates to *1* or *0*, and specify that column in the Exploded Wedges Column attribute.





Tooltips can be also added to charts, as shown above, by setting the **Tooltip Column** (Animated charts) or **Tooltip** (Static charts) attribute. This causes identifying data from the datalayer to be displayed as a tooltip, as shown above, when the mouse hovers over a portion of the chart. The Tooltip Column requires only the name of a column from the datalayer, while the Tooltip attribute requires the use of an @Chart token to display data. For animated charts, a **Tooltip Font** element is available for formatting the tooltip.

Certain charts can include a feature called **Quicktips**:

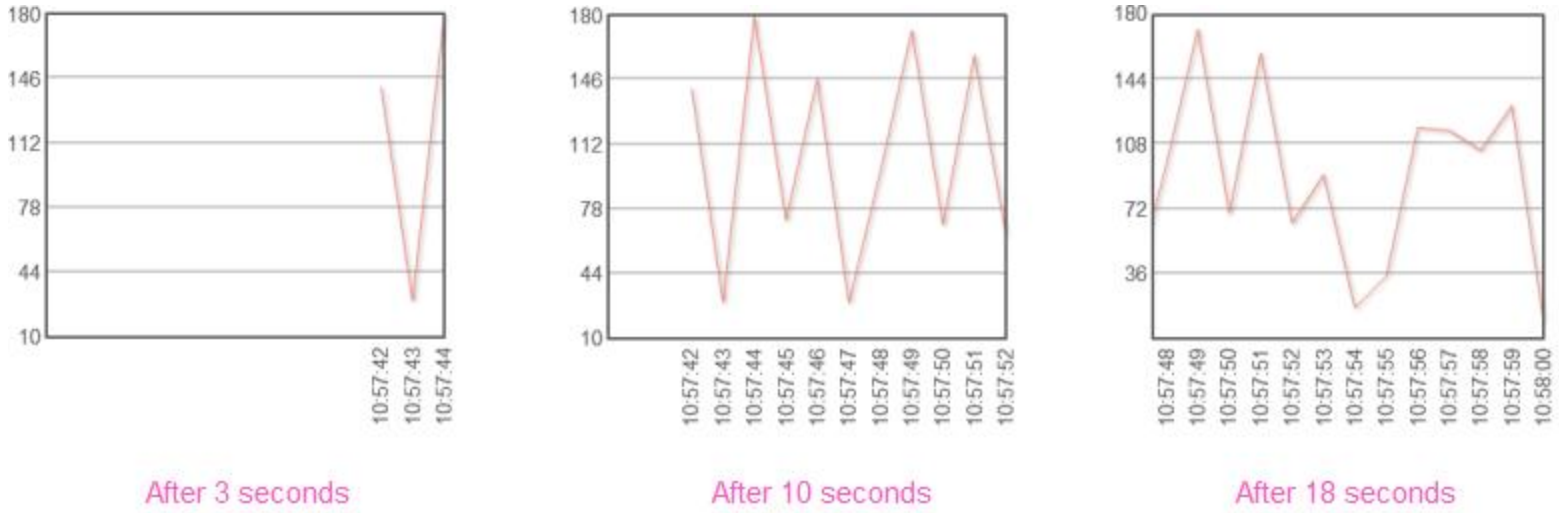


When the **Quicktip** element is added as a child of the chart element, it provides a small, pop-up "balloon" at the cursor location. The Quicktip allows the balloon to have a title and one or more rows of text. Data values can be shown by configuring the Quicktip element with the @Chart token.

In summary, data is retrieved for use in a chart by a datalayer. Typically, a chart on its own will display one data series but, with the addition of other helper elements, it can be made to show additional data series.

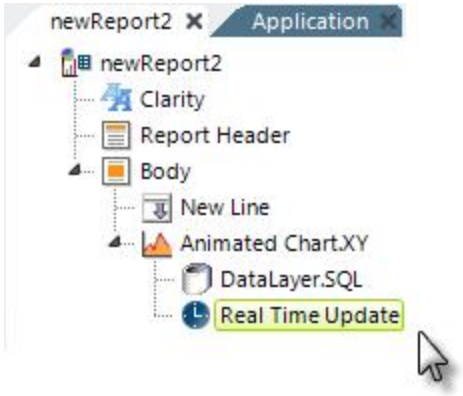
Refreshing Animated Chart.XY in Realtime

The **Realtime Update** element was introduced. It transforms an **Animated Chart.XY** into a chart that updates itself periodically, by adding new data values at its right side and scrolling the X-axis horizontally.



The example above shows a Line chart displaying a value as it changes, once per second. The Realtime Update element uses AJAX technology to automatically refresh its Animated Chart parent element, based on an interval set in seconds. It works by re-running just that portion of the report that retrieves the chart data and passing it back to the browser to update the chart. The data is accumulated internally and re-processed each time the chart is displayed.

If **Extra Data Column** elements are used, they're displayed as additional lines in Line charts, and as stacked values in Area and Bar charts.

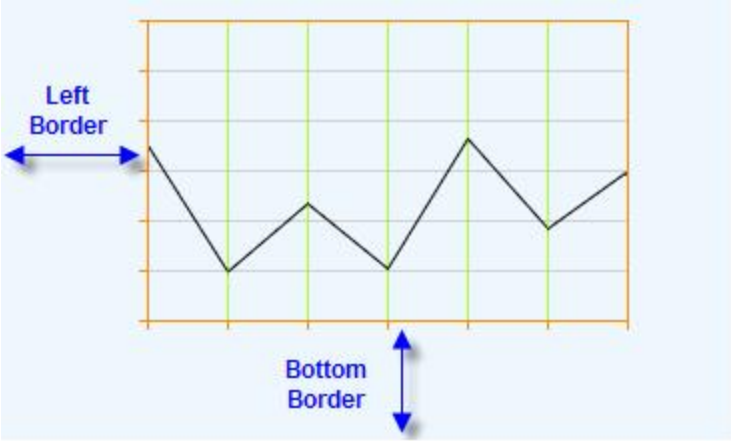


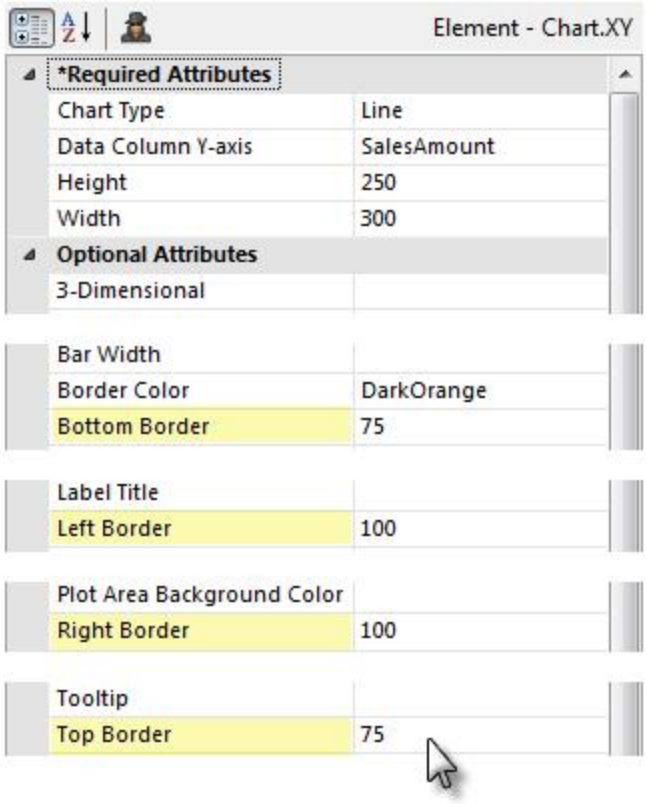
As shown above, the Realtime Update element is used by adding it beneath the Animated Chart.XY element. Its only attribute is the required **Refresh Interval** attribute, which is specifies, in seconds, how frequently to refresh the chart.

The Realtime Update element *cannot* be used with a chart whose datalayer has a **Crosstab Filter** applied to it.

Title, Labels, and Legends

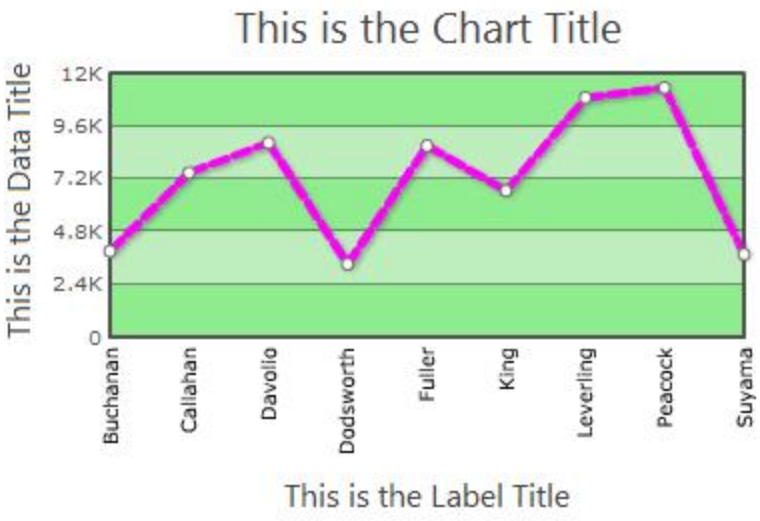
Logi Classic Chart elements include numerous attributes and supporting elements that can be used to tailor their chart titles, labels, and legends. These help developers get the right "look" for their charts, thereby helping users understand what they're seeing.

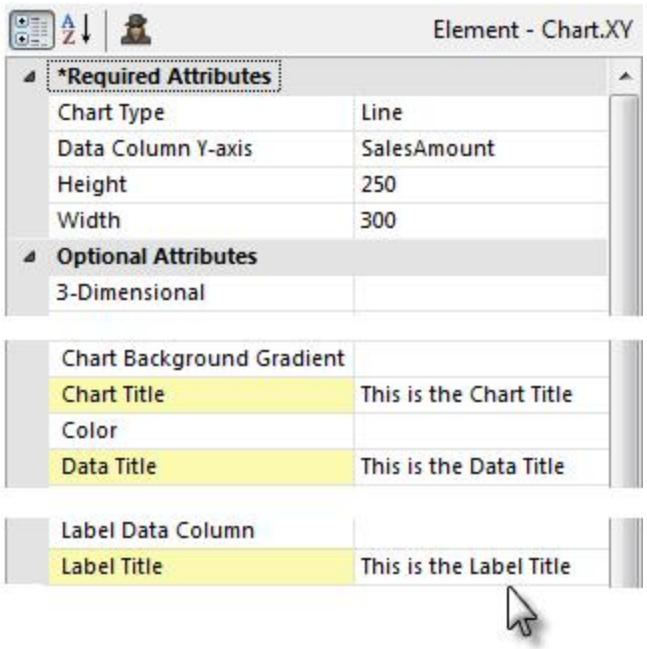




Space for titles is automatically allocated for all charts.

These indicate, in **pixels**, the distance on each side from the chart **canvas** to the **edge** of the chart image space. The overall vertical size of the chart is therefore calculated as the sum of the Top Border + Height + Bottom Border. The horizontal size calculation is Left Border + Width + Right Border.





Titles for the chart, data, and labels can be set using the **Chart Title**, **Data Title** and **Label Title** attributes as shown above.

Static Chart titles that are longer than the width of the chart itself are automatically truncated and have an ellipsis ("...") appended to them, as shown above left. In Animated Charts, the title text will wrap to another line.

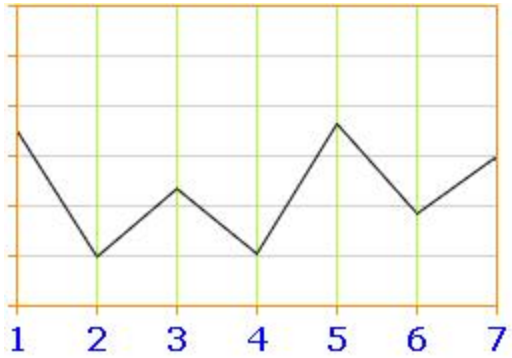
Static Charts can also use the **Chart Title Font** child element to set the title font, color, size, angle, and alignment.

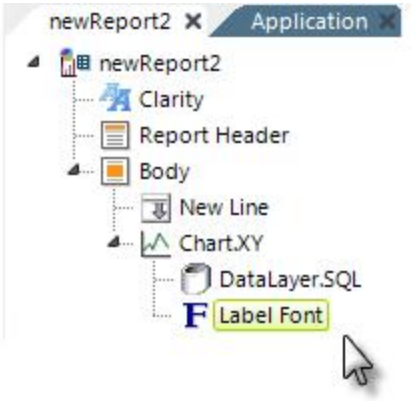
Static Chart titles can also be rendered in *multiple colors*, as shown above right, by using **CDML** tags directly in the **Chart Title** attribute value to specify different font colors, sizes, etc. Find out more about CDML in *Classic Chart Label Formatting*. The example shown above was produced using this value:

```
<*color=0000FF*>Multi-Color <*color=FFD700*>Chart <*color=BA55D3*>Titles <*color=000000*>Available Too!
```

All charts include a **Max Label Length** attribute, which specifies the maximum number of characters that will be displayed for a label before the text is trimmed and the remainder replaced with an ellipsis (...). This truncation is also applied to Legend labels for Chart.Pie, if a Legend is in use (it is *not* applied to legends for other types of charts).

Static XY and Pie charts include a **Label Column Data Type** attribute, which can be used to set the way in which the charting engine will interpret the data specified in the column named in Label Data Column X-axis. This can be useful, for example, when you want dates in the data to be presented as text in the chart's X-axis labels.

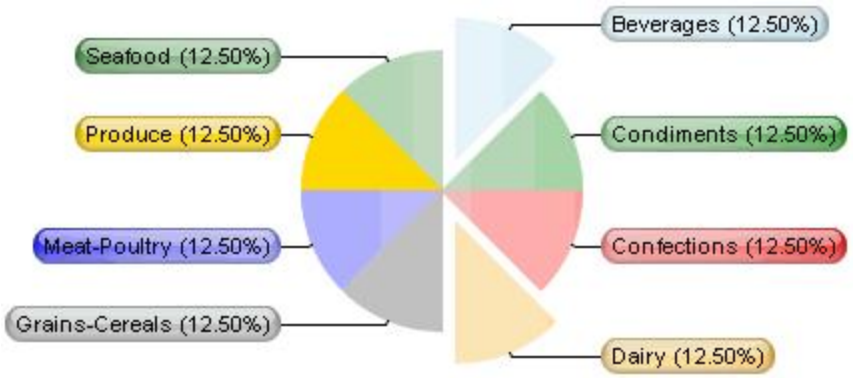


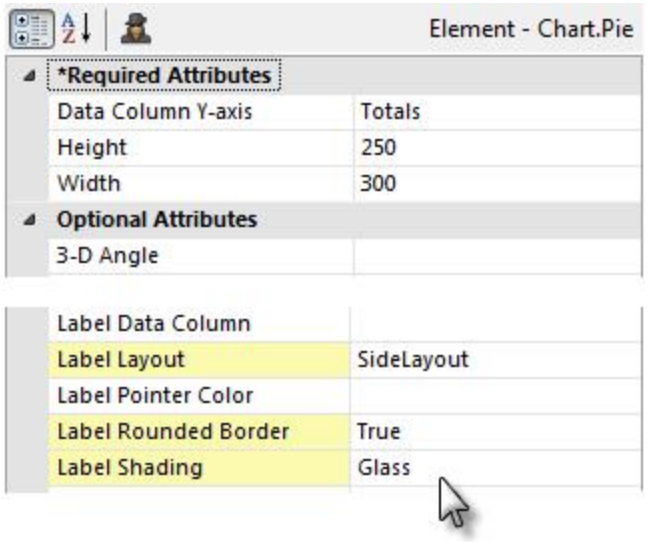


As shown above, labels can be formatted by adding a **Label Font** element, providing control over font family, size, color, angle, and decimal point appearance. *Font angle is only configurable for Static charts, not for Animated charts*.

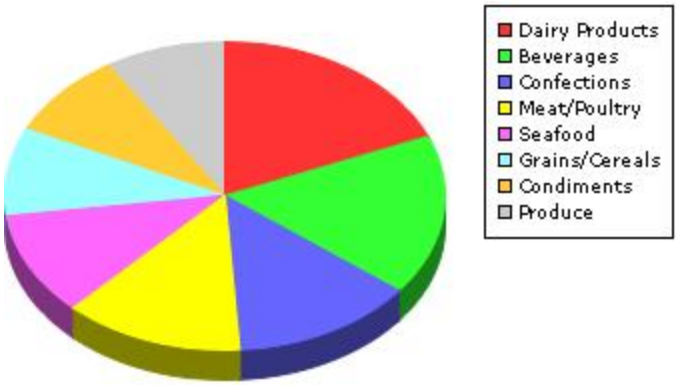
A **Label Scale** element is also available for controlling the appearance, size, and frequency of **tick marks** along the scales. One of its modes is *LinearTime*, which plots points evenly end-to-end according to an X-axis value, for use with Line and Area charts.

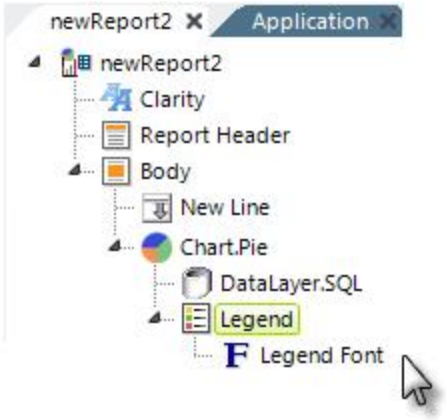
Labels can also be formatted using the **CDML** formatting language; find out more about CDML in *Classic Chart Label Formatting*.





Attributes of the Static Pie chart allow the position, shape, and shading of chart Labels to be set.





Legends provide an explanation of some of the visual information in a chart, which is often color-related. In the example above, the legend associates products with colors. In Static charts, two elements, **Legend** and **Legend Font**, shown above, control the

location, size, font, border, orientation, and background color of legends. In some Animated charts, providing a value for their **Legend Title** attribute automatically generates the legend. The association between the colors and the data is also generated automatically.

When the Legend element's **Legend Filter** attribute has been set to *True*, it becomes possible to dynamically filter the data in some Static charts, including Pie, Line, Area, Bar, and Polar charts, by clicking items in the Legend, as shown above.

Drill-down and Drill-through Features

Users frequently want to be able to get the details that support the data shown in charts and there are several ways developers can provides this. Adding "drill-down" functionality involves adding **Action** elements so that clicking on the chart will present detail information.

For example, adding an **Action.Report** and **Target.Report** element as children of the **Animated Chart.Pie** element will cause each wedge in a Pie chart to become a link to the target report.

When a chart uses its own datalayer element to retrieve data, that data is accessed with an **@Chart** token, not the @Data token that's used with Data Tables.

Data specific to each pie wedge can be passed to the target report using **Link Parameters** and an @Chart token. The target report can then take action, such as filtering its own data, based on an @Request token representing the Link Parameter value.

Automatic Drill-Through Report

Charts often use grouping to aggregate data for a consolidated view. There are times, however, when it's useful to be able to examine the detail data that was used to create that aggregation and you can provide this "drill-through" functionality with the **Group Drillthrough** element.

In the example shown above, a **Group Drillthrough** element has been added beneath a Chart.Pie element. All of the element's attributes, other than **ID**, are optional but the example shows that a custom caption and an export button selection have been entered. More information about the attributes can be found in the element's Element Reference page.

The resulting pie chart looks no different than usual, but when one of its wedges is clicked, a detail report, like the example shown below, is automatically generated, containing all of the relevant detail data.

Category Sales Detail Report

Drill down by: CategoryID="6"

Export: PDF

ProductID	CategoryID	CategoryName	Description	ProductName	calcOrderValue
17	6	Meat/Poultry	Prepared meats	Alice Mutton	1170
17	6	Meat/Poultry	Prepared meats	Alice Mutton	585
17	6	Meat/Poultry	Prepared meats	Alice Mutton	585
17	6	Meat/Poultry	Prepared meats	Alice Mutton	1560
17	6	Meat/Poultry	Prepared meats	Alice Mutton	312
17	6	Meat/Poultry	Prepared meats	Alice Mutton	780
17	6	Meat/Poultry	Prepared meats	Alice Mutton	2730
17	6	Meat/Poultry	Prepared meats	Alice Mutton	1404
17	6	Meat/Poultry	Prepared meats	Alice Mutton	78
17	6	Meat/Poultry	Prepared meats	Alice Mutton	1755
17	6	Meat/Poultry	Prepared meats	Alice Mutton	1950
17	6	Meat/Poultry	Prepa		390

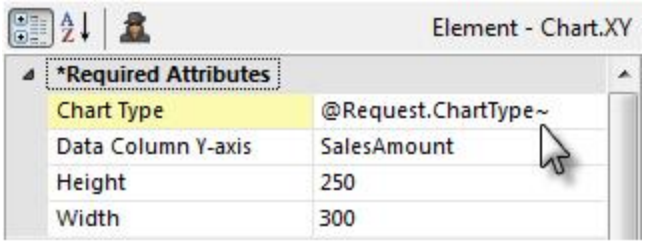
Group Drillthrough is only available for the **Chart.XY**, **Chart.Pie**, **Chart.Heatmap**, and **Chart.Scatter** elements.

The Group Drillthrough element is designed to work charts with datalayers that have a **Group Filter**, **Crosstab Filter**, or **Relevance Filter** child element. This has implications if you want to use Group Drillthrough and you're working with DataLayer.SQL; for example, you'll have to use a Group Filter element instead of doing the grouping in your query.

The Group Drillthrough element's attributes allow you to customize certain aspects of the detail report. You can also include **Drill-through Column** elements beneath the Group Drillthrough element; they allow you to set the number columns that will appear in the report and to customize their appearance.

Interactive Chart Configurations

All of the charting elements' attributes can be "tokenized"; that is, you can use tokens in them as values. This allows charts to be highly-interactive and dynamically customizable.



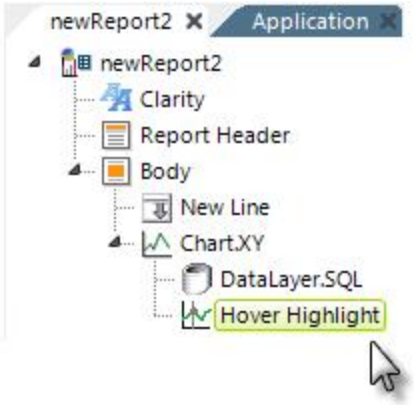
For example, a report page could include an **Input Select List** element so the user has a choice of chart types: *Bar*, *Line*, and *Area*. When the user makes a selection, the report will be refreshed and their selection is automatically passed as a Request variable**.** The resulting @Request token can then be used in the a Chart.XY element's **Chart Type** attribute, as shown above, to dynamically set the chart type. Other use-case examples include:

- Through the use of cookies and @Cookie tokens, chart configurations can be "personalized" for individual users and the settings retained between sessions.
- Charts can be included in reports as child elements of tables and have a "data-driven" configuration, using @Data tokens to provide values for chart attributes.

The chart configuration flexibility available through the use of tokens is quite extensive, providing opportunities for user-customizable chart size, colors, type, content, and more.

Hover Highlight

The **Hover Highlight** child element causes chart features, such as bars and lines, to be highlighted when the mouse pointer is hovered over them, making it easier for users to visualize the item under the pointer. The element is available for use with Static Pie, Bar, Line, Area, and Scatter charts.

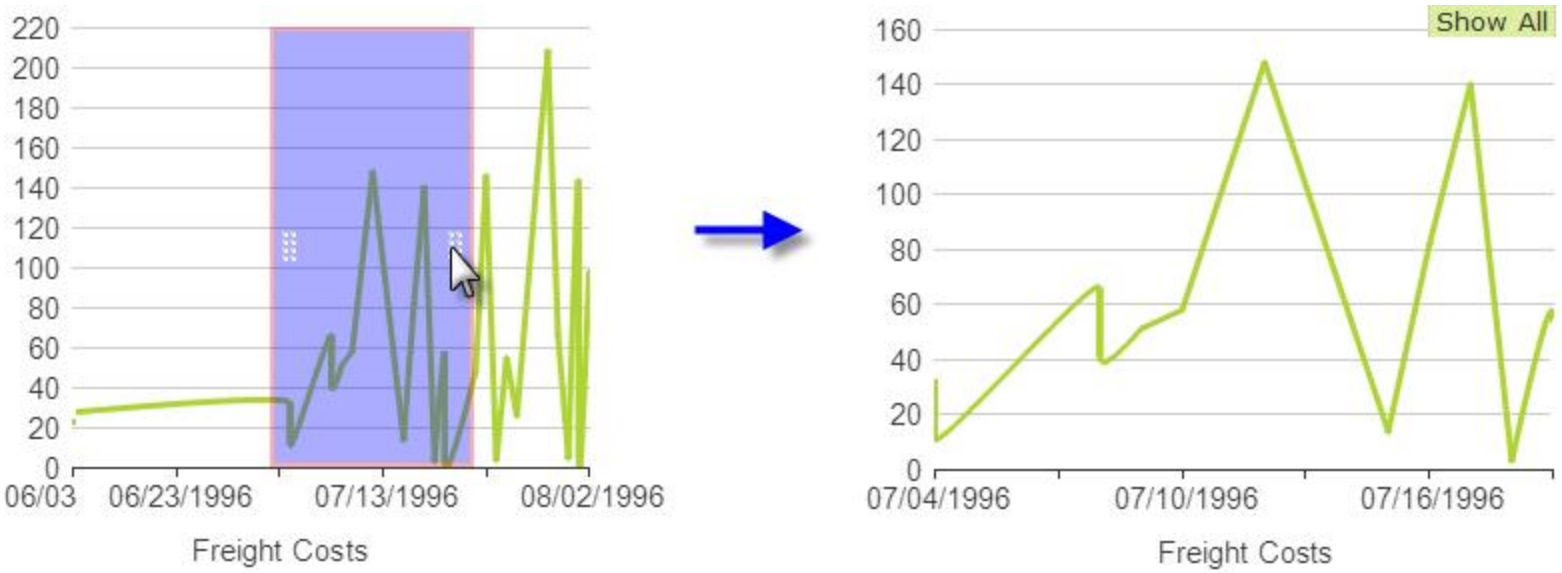


The element's attributes enable customization of the highlight overlays. The **Fill Color** attribute applies to Pie and Bar charts, the **Border Color** and **Border Width** attributes to Line, Area, and Scatter charts, and **Line Color** applies to Line and Area charts.

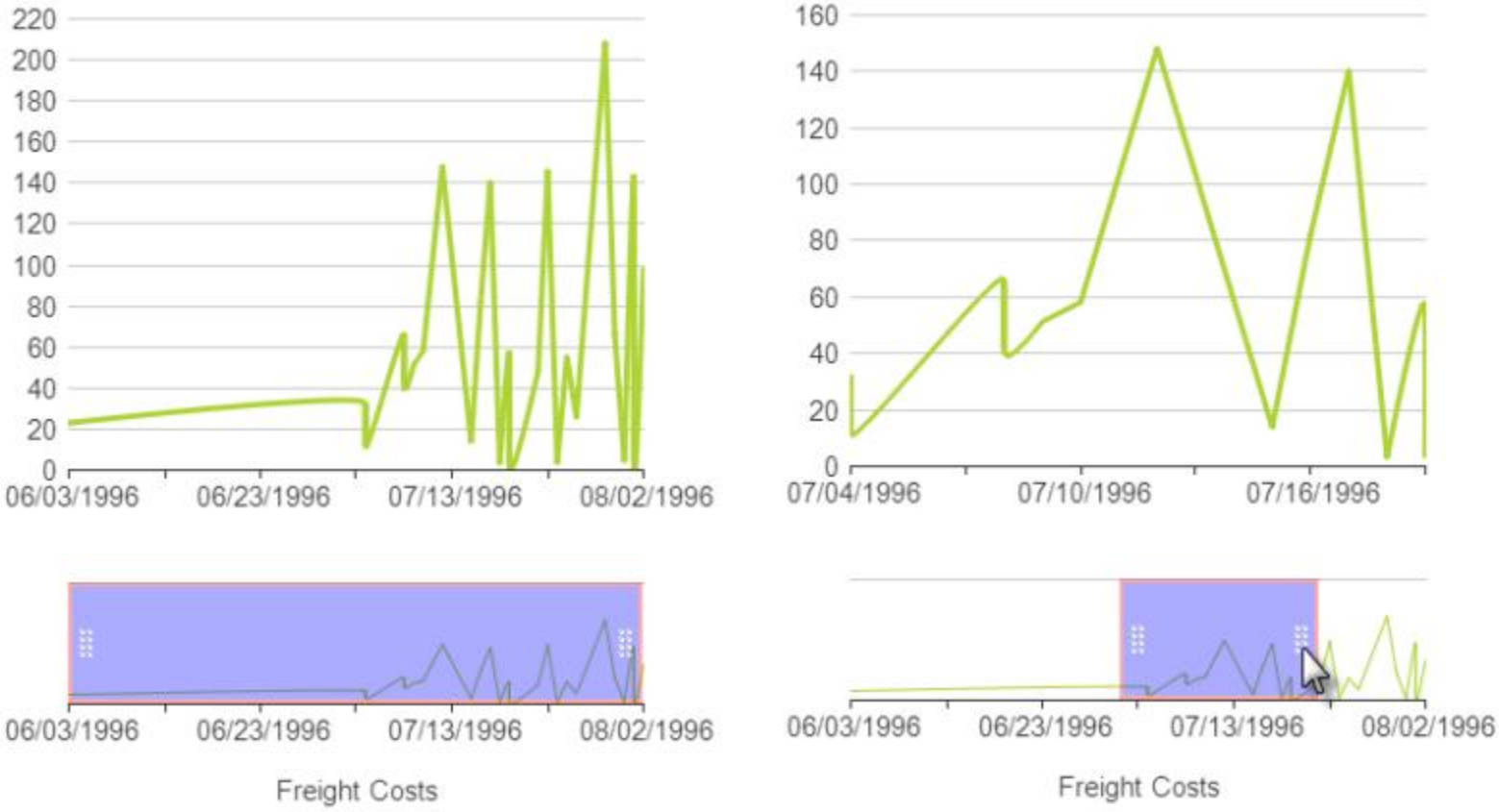
Attempts to use the *ThemeAlignCenter* class, or other CSS classes that use `text-align`, on a container such as a DIV in order to center a chart that includes Hover Highlight are likely to fail. Hover Highlight is not a simple HTML entity, like text or an image, that other content can flow around; it's wrapped in its own DIV and no longer behaves like block content. One possible alternate approach is to use an HTML table instead of a DIV as the container. Due to its non-standard behavior, this kind of centering *will* work with IE 9 only.

Zoom Chart

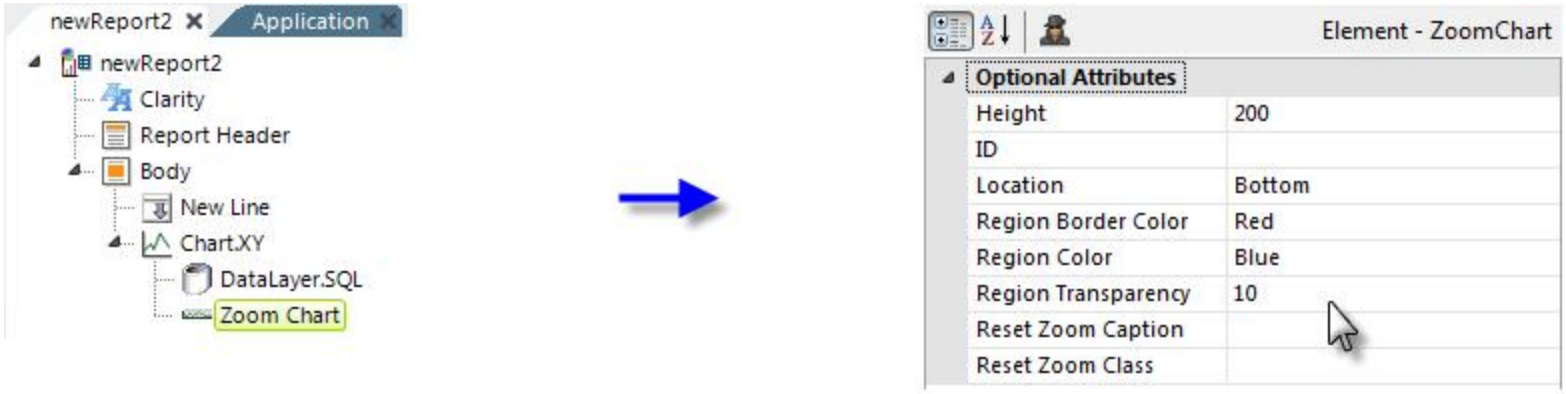
The **Zoom Chart** element provides users with the ability to "zoom" into a chart's data by selecting an area on the chart. The Zoom Chart is available for use with **static** Chart.XY and Chart.Scatter elements.



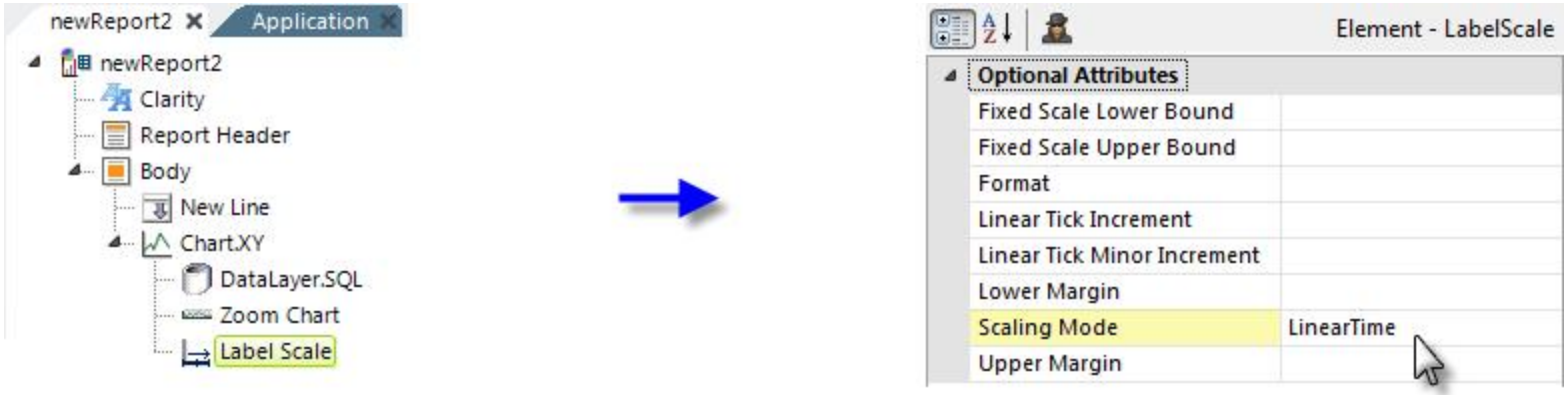
There are two configurations of the Zoom Chart. In the first, shown above left, the "selection area" appears right over the parent chart. Resizing handles allow its size to be adjusted and it can also be dragged horizontally. When the mouse button is released, the parent chart zooms in or out to include just the data within the selection area, as shown above right. Successive selections can be made on the zoomed chart. The "Show All" button returns the view to encompass all of the original data.



In the second configuration of the Zoom Chart, shown above, a smaller version of the chart, with selection area, appears *below* the chart. Changes to the selection area cause the view of the data in the chart to change correspondingly.



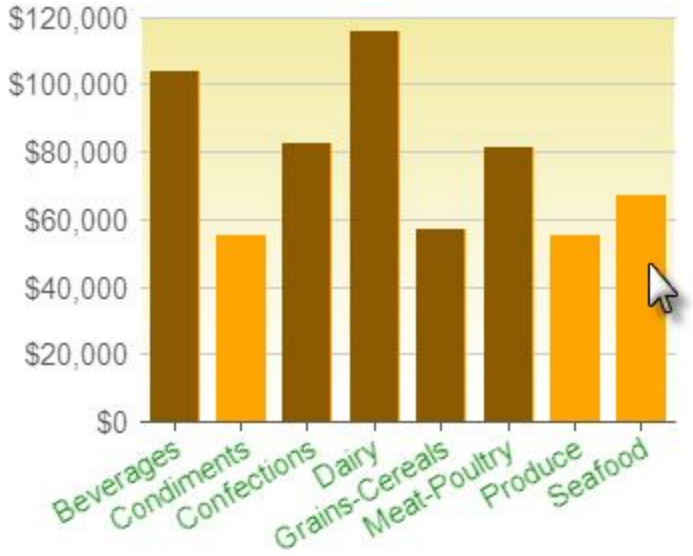
As shown above, the **Zoom Chart** element is added as a child of the parent chart and its attributes allow you to customize the selection area and chart. The **Location** attribute is set to *None* to display the selection area *on top of* the chart; setting it to *Bottom* displays the small selection area and chart beneath it.

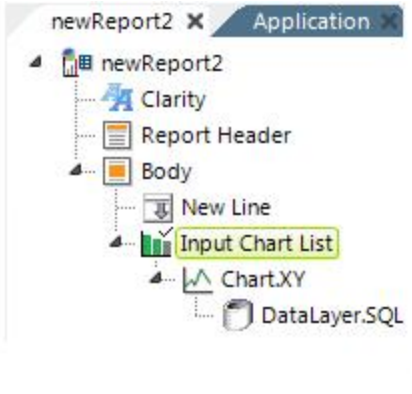


A **Label Scale** element is also *required*, as shown above, with a **Scaling Mode** set to one of the *Linear* options (based on your X-axis data type).

InputChart.List and InputChart.Range

The **InputChart.List** and **InputChart.Range** elements allow users to work with charts as *user input* elements. Users can select data values on a chart and pass them, in manner similar to other user input elements, through Request tokens to other definitions. To use either one of them, add it as the *parent* element of a Static chart element.

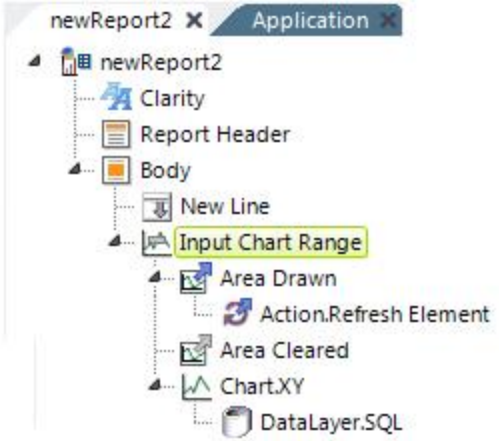




As shown above, **InputChart.List** works with Static Bar and Pie charts. At runtime, chart bars or wedges can be clicked to select them. When the report page is submitted, the X-axis values are submitted to the next definition as a Request token that includes the chart ID, whose value is comma-separated list. For example,

```
@Request.ChartXY_Bar~ = "Condiments,Produce,Seafood"
```

would result from submitting the example above. The element includes attributes common to Input elements and attributes for specifying the colors of selected and unselected bars or wedges. Standard **Action** elements (not shown) are required to submit the page.



As shown above, **InputChart.Range** works with Static Line, Spline, and Scatter charts. At runtime, a "selection area" can be "drawn" onto the chart plotting area to select values by dragging the mouse. When the report page is submitted, the maximum and minimum X-axis and Y-axis values are submitted to the next definition as individual Request tokens. For example:

```
@Request.minX_value~ = "2/20/2012"
@Request.maxX_value~ = "3/31/2012"
```

```
@Request.minY_value~ = "0"
@Request.maxY_value~ = "1200"
```

would result from submitting the example above. Clicking outside of the selection area will clear it.

The element includes attributes for specifying the names of the request variables to be associated with the value range and the colors of the selection area. Two special child event handler elements, **Area Drawn** and **Area Cleared**, can be used to fire standard Action elements, as shown, when the selection area is drawn or cleared, refreshing or submitting the page. In addition, or instead, standard, unrelated **Action** elements (not shown) can also be used to submit the page.

More information is available in *Work with User Input Elements*.

Working with Gauges

Static gauges provide a variety of interesting visualizations that can be almost photographic in their appearance. Gauges are SVG images that can be exported and scaled smoothly (they can have a child Resizer element).

Many gauges can be exported as a .PNG image, including the following:

- Gauge.Arc
- Gauge.Angular
- Gauge.Balloon Bar
- Gauge.Bullet Bar
- Gauge.Number

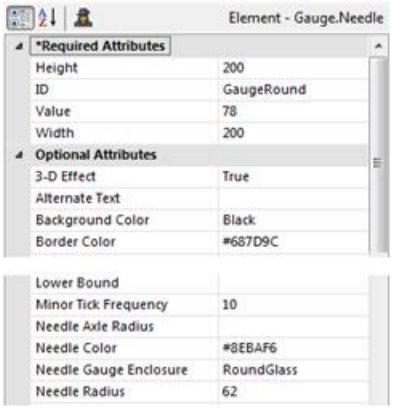
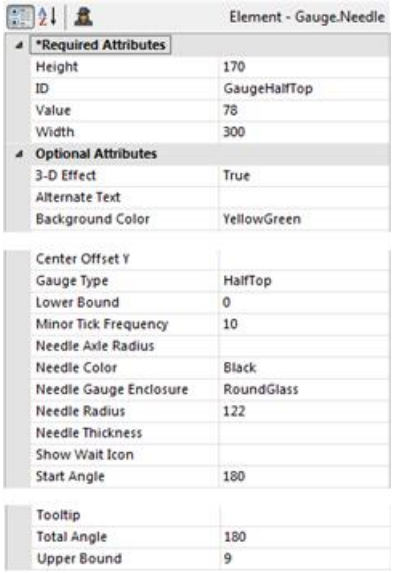
More information about how to configure gauges to be exportable can be found in *Export Chart Canvas Charts*.

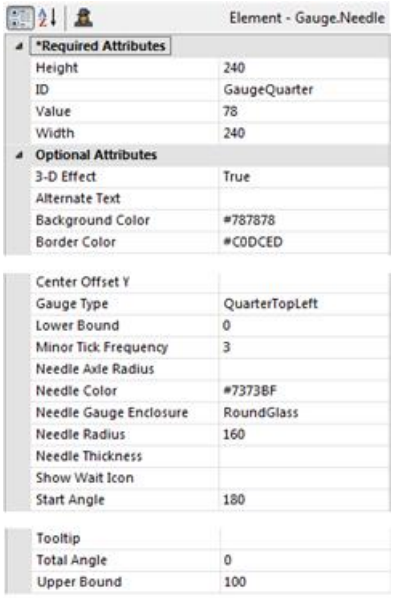
Needle Gauges

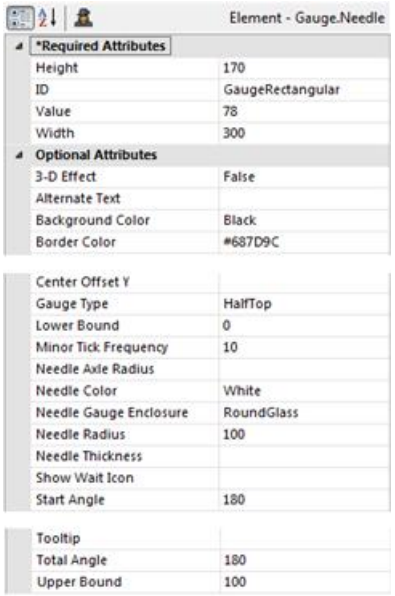
Needle gauges look like speedometers: they have a needle pointer that swings through an arc and indicates a data value on a scale. Gauge shapes include half-round, quarter-round, and rectangular.

The Gauge.Needle element has been deprecated, you should use **Gauge.Angular** instead.

The **Start Angle** and **Total Angle** attributes of the **Gauge.Needle** element govern the possible positions of the needle pointer. Here's a table of examples, showing some of the possible shapes and configurations:

Gauge Type	Configuration	Notes
Round Gauge 		Start Angle value minimum = 0 Total Angle maximum = 360.
Half-Round Gauge 		*HalfTop* (shown) Start Angle value minimum = 180 Total Angle maximum = 180 *HalfBottom* Start Angle value minimum = 0 Total Angle maximum = 180.

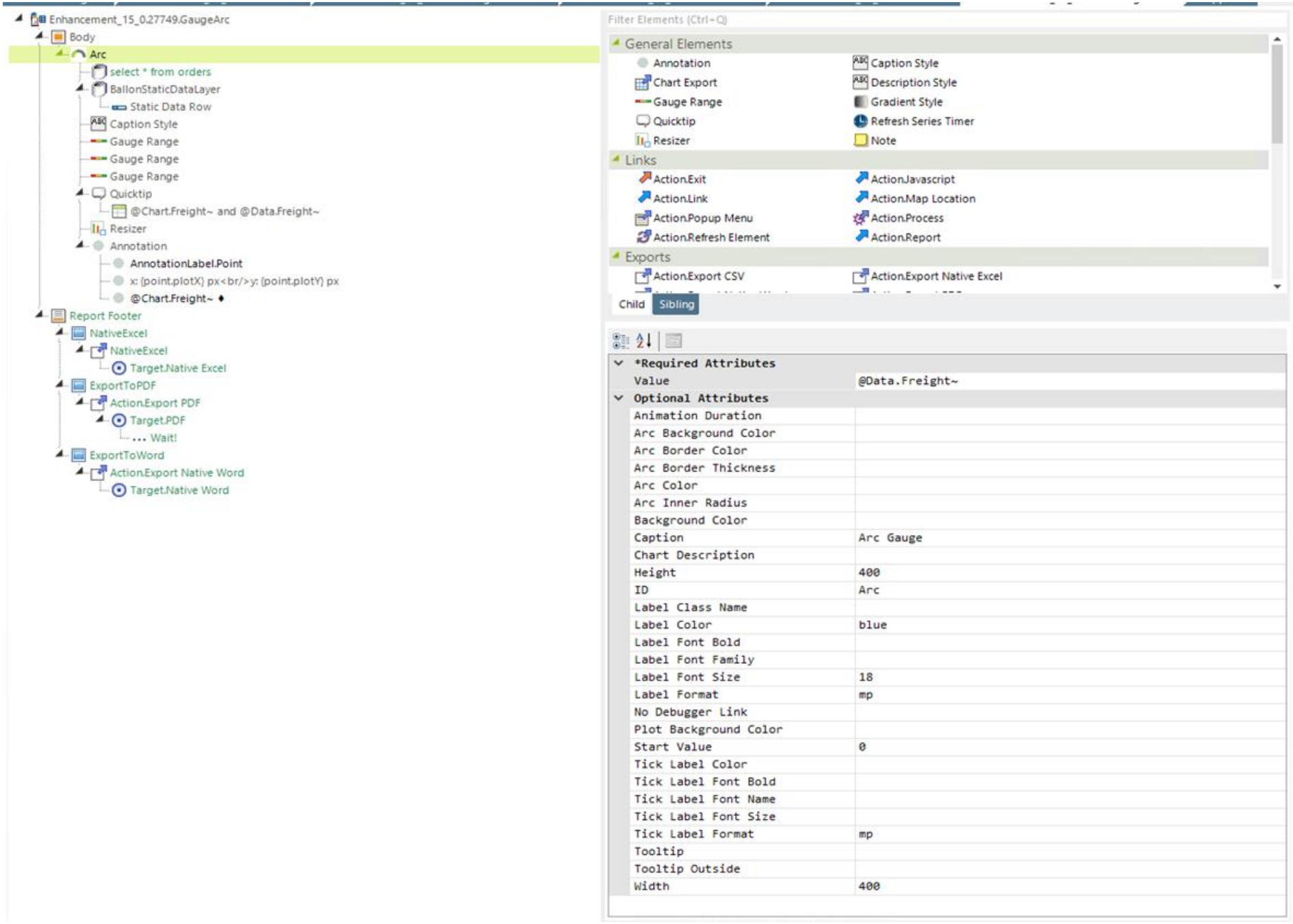
Gauge Type	Configuration	Notes
Quarter-Round Gauge 		*QuarterTopLeft* (shown) Start Angle value = 180 Total Angle = 90 *QuarterTopRight* Start Angle value = 270 Total Angle = 90 *QuarterBottomRight* Start Angle value = 0 Total Angle = 90 *QuarterBottomLeft* Start Angle value = 90 Total Angle = 90

Gauge Type	Configuration	Notes
Rectangular Gauge 		Type is set to *HalfTop*.

Arc Gauges

Arc gauges are simple, clean visualizations which show beginning and ending values, with an arched color bar between them. The data value number is shown inside the arc.

The gauge may have multiple ranges in different colors. The color of the arc is determined by which range contains the value.


Enhancement_15_0.27749.GaugeArc
Body
Arc
select * from orders
BallonStaticDataLayer
Static Data Row
Caption Style
Gauge Range
Gauge Range
Gauge Range
Quicktip
@Chart.Freight~ and @Data.Freight~
Resizer
Annotation
AnnotationLabel.Point
x: {point.plotX} px
y: {point.plotY} px
@Chart.Freight~
Report Footer
NativeExcel
NativeExcel
Target.Native Excel
ExportToPDF
Action.Export PDF
Target.PDF
... Wait!
ExportToWord
Action.Export Native Word
Target.Native Word
Filter Elements (Ctrl+Q)
General Elements
Annotation
Caption Style
Chart Export
Description Style
Gauge Range
Gradient Style
Quicktip
Refresh Series Timer
Resizer
Note
Links
Action.Exit
Action.Javascript
Action.Link
Action.Map Location
Action.Popup Menu
Action.Process
Action.Refresh Element
Action.Report
Exports
Action.Export CSV
Action.Export Native Excel
Child
Sibling
*Required Attributes
Value
@Data.Freight~
Optional Attributes
Animation Duration
Arc Background Color
Arc Border Color
Arc Border Thickness
Arc Color
Arc Inner Radius
Background Color
Caption
Arc Gauge
Chart Description
Height
400
ID
Arc
Label Class Name
Label Color
blue
Label Font Bold
Label Font Family
Label Font Size
18
Label Format
mp
No Debugger Link
Plot Background Color
Start Value
0
Tick Label Color
Tick Label Font Bold
Tick Label Font Name
Tick Label Font Size
Tick Label Format
mp
Tooltip
Tooltip Outside
Width
400

The example shows how the **Gauge.Arc** element is configured and its child Gauge Range elements, which give the arc different colors for specific data values.

v23.1 The attribute "Label Class Name" allows you to apply unique styling to your gauge's data labels by assigning a class name from a style sheet in the report.

v23.1 When the Annotation element is used as a child of Gauge.Arc, you can place custom labels and shapes at various points of interest.

The **No Debugger Link** attribute can be used to suppress individual chart/gauge debug icons that appear when debugging is turned on. You may want to do this to reduce "visual clutter" and ensure a realistic layout when debugging a page that includes relatively small charts and gauges.

Default Color Sets

If you're using one of the standard Logi themes, the theme will set the default colors. These colors can be identified by looking at the theme's Template Modifier file: `<yourAppFolder>\rdTemplate\rdTheme\<ThemeName>\ThemeModifier.xml`

If you're *not* using a theme and do not specify a color set, the following default color set will be applied.

Static Charts

#	Hex Code
1	FF3333
2	33FF33
3	6666FF
4	FFFF00
5	FF66FF
6	99FFFF
7	FFCC33
8	CCCCC

#	Hex Code
9	CC9999
10	339966
11	999900
12	CC3300
13	669999
14	993333
15	006600
16	990099
17	FF9966
18	99FF99
19	9999FF
20	CC6600
21	33CC33

#	Hex Code
22	CC99FF
23	FF6666
24	99CC66
25	009999
26	CC3333
27	9933FF
28	FF0000
29	0000FF
30	00FF00
31	FFCC99
32	999999

List continues with other colors not shown here.

Animated Charts

#	Hex Code
1	AFD8F8
2	F6BD0F
3	8BBA00
4	FF8E46
5	008E8E
6	D64646
7	8E468E
8	588526
9	B3AA00
10	008ED6
11	9D080D
12	A186BE

#	Hex Code
13	CC6600
14	FDC689
15	ABA000
16	F26D7D
17	FFF200
18	0054A6
19	F7941C
20	CC3300
21	006600
22	663300
23	6DCFF6

Colors are repeated again after 23 colors are applied.

Crosstab Filter with Chart Canvas Charts

The **Crosstab Filter** is usually associated with a table but, as an element that "pivots" the data in a datalayer, it can also be used with Chart Canvas charts.

This topic contains the following sections:

About Crosstabs

A *cross tabulation* (often abbreviated as "crosstab") is a table that displays the joint distribution of two or more variables simultaneously. Sometimes called "pivot tables", they make it easy to sort, count, and total their data. The Logi **Crosstab Table** element makes it easy to implement this kind of table and uses the **Crosstab Filter** element to shape the data in its datalayer.

Let's see how the datalayer's data is processed by the Crosstab Filter:

"Crosstab Column" data

"Label Column" data

Employee Name	2010	2011	2012
Buchanan, Steven	$17,667.20	$31,433.19	$19,691.90
Callahan, Laura	$19,160.70	$56,954.04	$47,727.95
Davolio, Nancy	$30,861.76	$95,850.40	$60,565.22
Dodsworth, Anne	$9,894.52	$24,412.89	$42,142.66
Fuller, Andrew	$17,811.46	$71,168.14	$73,790.18
King, Robert	$15,232.16	$59,827.19	$44,559.89
Leverling, Janet	$18,223.96	$103,719.08	$80,869.80
Peacock, Margaret	$49,945.12	$124,655.57	$51,163.01
Suyama, Michael	$14,519.69	$40,826.37	$17,181.58

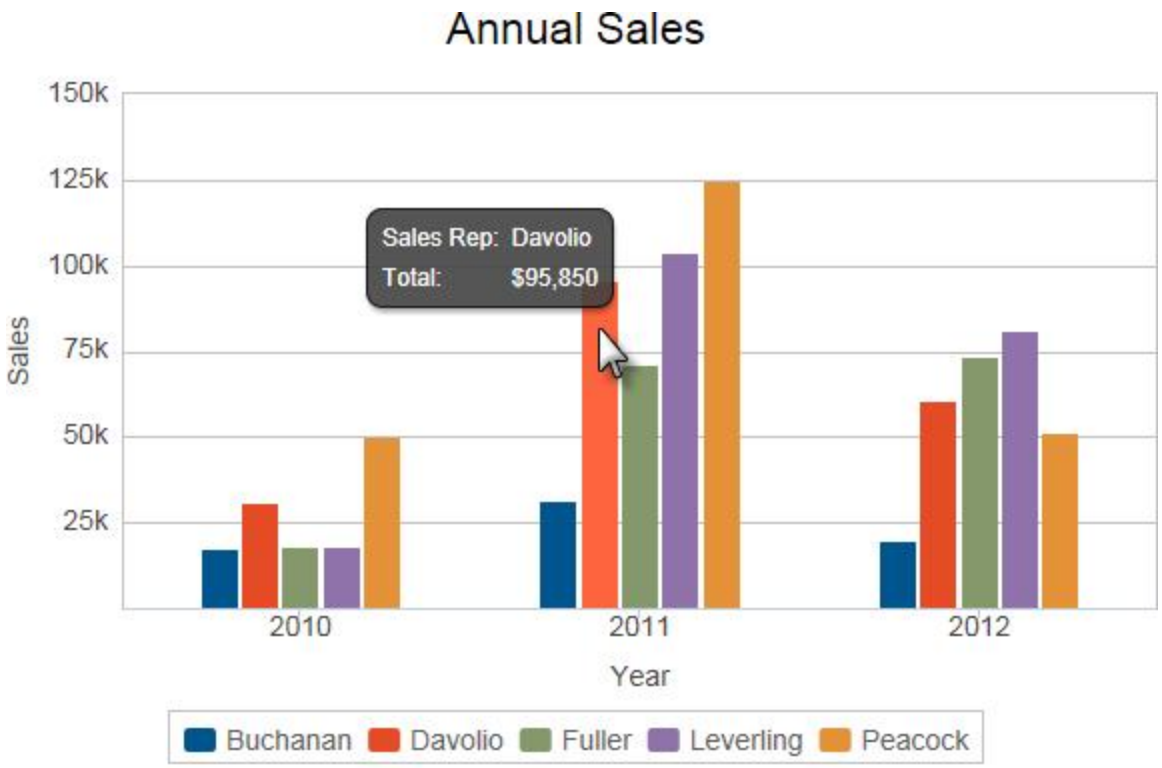
"Value Column" aggregated data

The example above shows each of the column types you specify in the filter and how they appear in a cross-tab table.

- *Unique* values in the datalayer column named as the **Crosstab Column** create new columns in the Crosstab Table.
- *Unique* values in the column named as the **Label Column** create a new row in the table.
- Values in the column named as the **Value Column** fill the cells at the intersections of the Crosstab columns and Label rows. These values are the results of an aggregating function that's part of the filter.

The example above displays years for the Crosstab column, employee names for the Label column and a sum of corresponding subtotals for the Value columns. So, when the year is *2010* and the employee is *Nancy Davolio*, the sum of all corresponding sub-totals is *$30,861.76*.

Data that has been "cross-tabbed" can also be visualized in a chart. The following example illustrates how the crosstab data categories described above work in a chart.



The chart shown above uses a crosstab filter to achieve its "grouped bars" effect.

- The Crosstab Column in this case is the employee's Last Name data, and produces an individual bar in each time period for each employee.
- Each unique value in the Label Column creates a new grouping of data by Year in the X-axis.
- Data from the Value Column provides the summed value that dictates the height of each bar.

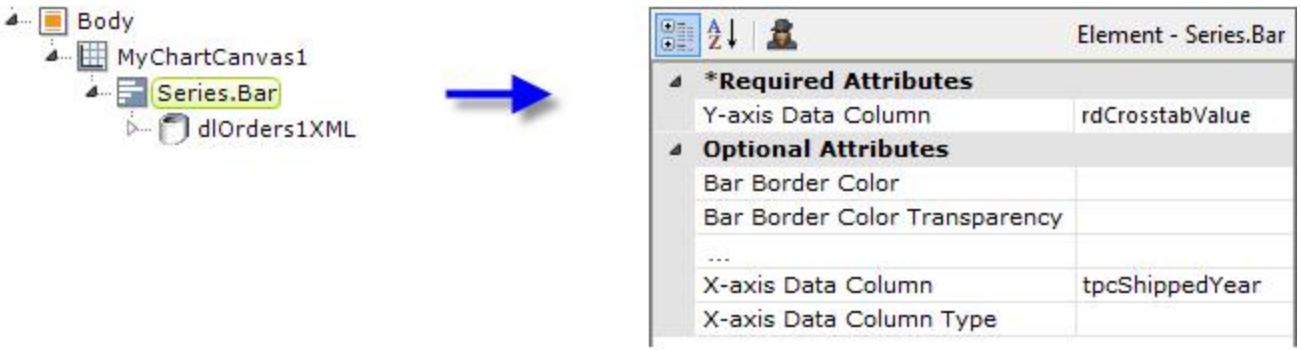
The following functions are available in the Crosstab Filter for aggregating Value columns:

- *Any* - Displays a value from any of the rows. This can be used when the specified Value column data is a string that isn't appropriate for aggregation; for example, when there's just one record to represent each cross-tab cell.
- *Average* - Returns the average of all corresponding records in the specified Value column.
- *Count* - Returns the total number of corresponding records in the specified Value column.
- *DistinctCount* - Returns the total number of unique corresponding records in the specified Value column.
- *Median* - Returns the value that separates the higher half of all values in the specified Value column from the lower half.
- *Mode* - Returns the value that occurs most frequently in records in the specified Value column.
- *StdDev* - (Standard Deviation) Returns a simple measure of the variability of data in records in the specified Value column. A low standard deviation indicates that the values tend to be very close to each other, while a high standard deviation indicates that the values are "spread out" over a large range.
- *Sum* - Returns the sum of all corresponding records in the specified Value column

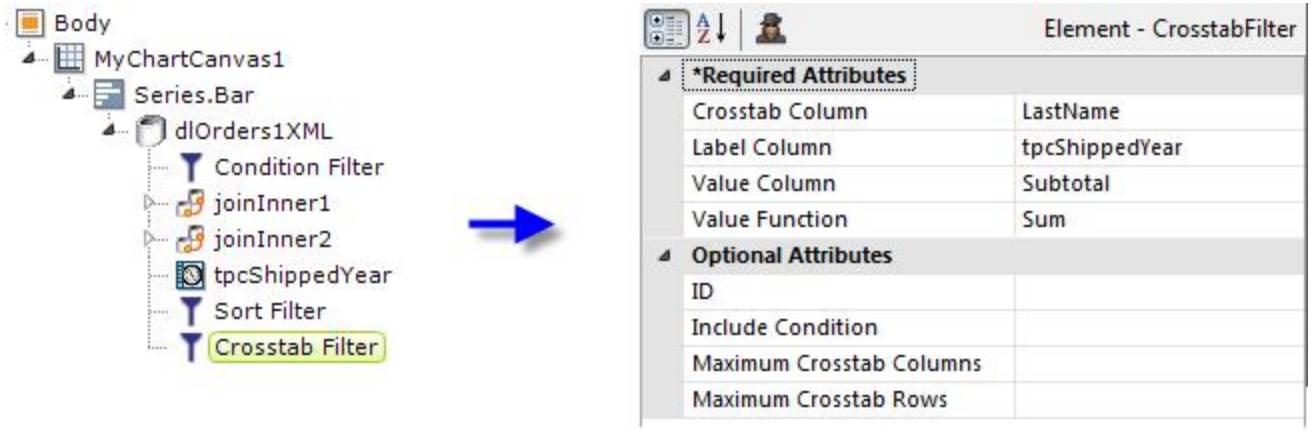
Aggregations *exclude* columns with Null values by default. They can be included by creating the constant `rdCalculationsIncludeNulls`and settingit to *True*

Creating a Cross-tabbed Chart

Here are the elements used for the cross-tabbed chart shown earlier:



In the example above, we see that a Series.Bar element has been configured. There are very few attributes needed, and you use the column the Crosstab Filter will generate, `rdCrosstabValue`, for the **Y-axis Data Column** attribute.



As shown above, a Time Period Column, a Sort Filter, and the **Crosstab Filter** have been added beneath the chart's datalayer.

The Sort Filter sorts the data on the employees' last names before the data is cross-tabbed. This puts the resulting bars in alphabetical order, left to right and, more importantly, ensures that the names are in alphabetical order in the legend. (The Legend element is not shown in the definition above).

The attribute values of the Crosstab Filter element are shown above and you can see how they correlate to the resulting chart.

Adding a Legend

In other circumstances, entering a value in the Series element's **Legend Label** attribute causes a legend to be displayed and use of the Legend element beneath Chart Canvas is *optional*, for styling and formatting purposes only.

However, when working with a Crosstab Filter, values are automatically provided to the series for use with a legend (so you can leave the Legend Label attribute blank) but you *must* add a Legend element to actually display a legend.

The use of a Crosstab Filter with charts produces interesting charts that allow visual analysis in ways that are not otherwise possible.

Heat Maps

The **Heat Map** charting element produces one of the most unusual charts found in Logi reporting products. Also known as a "tree-map", its unique arrangement of rectangles represents data and relationships using color and size. This topic introduces heat maps and the Chart.Heat Map element.

The following sections are covered in this topic:

- What is a "Heat Map"?
- Building a Heat Map

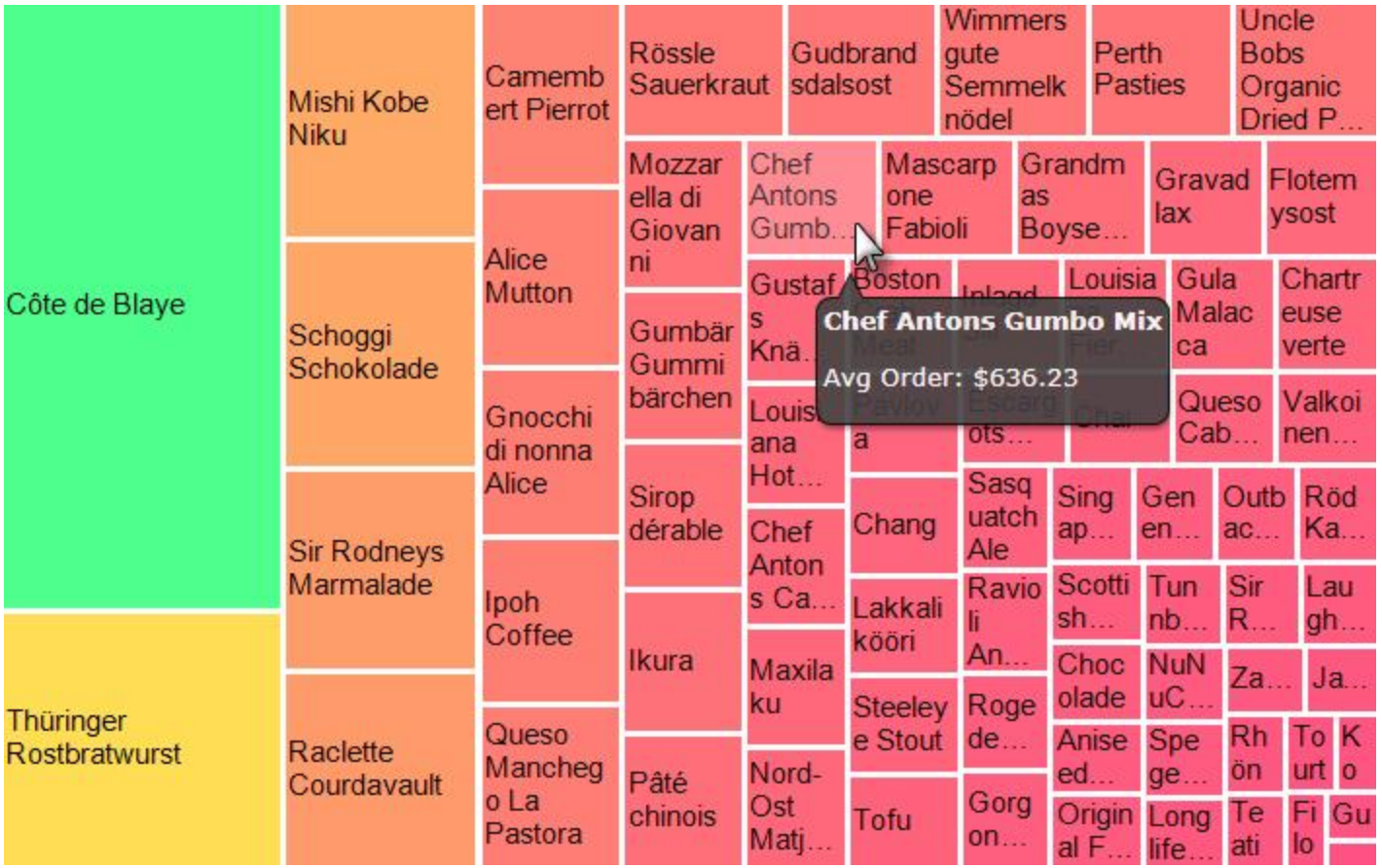
The Chart.Heatmap element has been deprecated; it's still supported and will work, but its elements are no longer available in Studio. Use Chart Canvas Chart and *Series.Heatmap*instead for all new development.

What is a "Heat Map"?

The Logi Analytics Heat Map chart is a descendant of the "treemap", first designed by Ben Shneiderman of the University of Maryland in 1991. He has an interesting treemap historypage that describes his work and shows many interesting applications of tree-maps. They're used today for analyzing data in a variety of fields, ranging from molecular biology to the stock market.

The term "Heat Map" is being used today in *several different ways*. Some heat maps are geographic maps overlaid with transparent color zones depicting average daily temperatures, homeless population densities, real estate prices, etc. Others use transparent color zones overlaid on images of web pages to indicate where visitors focus their attention and mouse clicks. And there are still others.

These other uses *do not apply* to Logi Info and are not related to Ben Shneiderman's original treemap concept. In Logi Info, a heat map is a chart, which looks like this:



It's a rectangular arrangement of small rectangles that uses **size** and **color** to visualize complex data values and relationships.

Both the color intensity (shifting within a gradient) and size of the boxes indicate **relative performance**, helping you to easily spot trends or standouts in a quick overview. This type of chart is very useful for showing **hidden trends** and relationships.

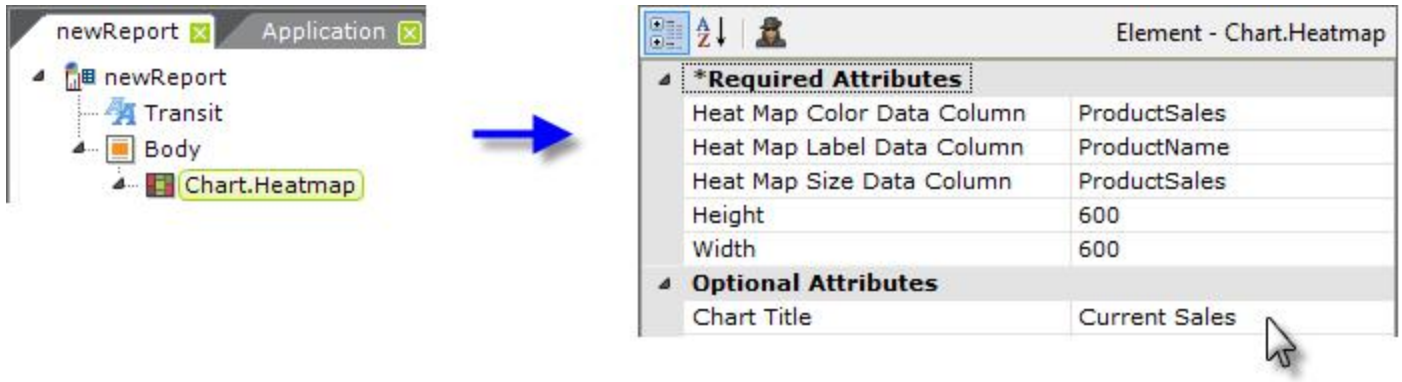
In the example above, using data from the Northwind Foods sample database, the size of the rectangles (or **cells**) represents annual revenue from sales of each food item. The cell color represents the number of orders and ranges from red (low) to green

(high). The relationship isn't linear but we can still see that items with small, red cells produced fewer sales and less revenue than items with large, green cells.

They can also be *interactive*; for example, they can be used with **Hover Highlighting**, **Quicktip** (shown above), and **Group Drillthrough** elements.

Building a Heat Map

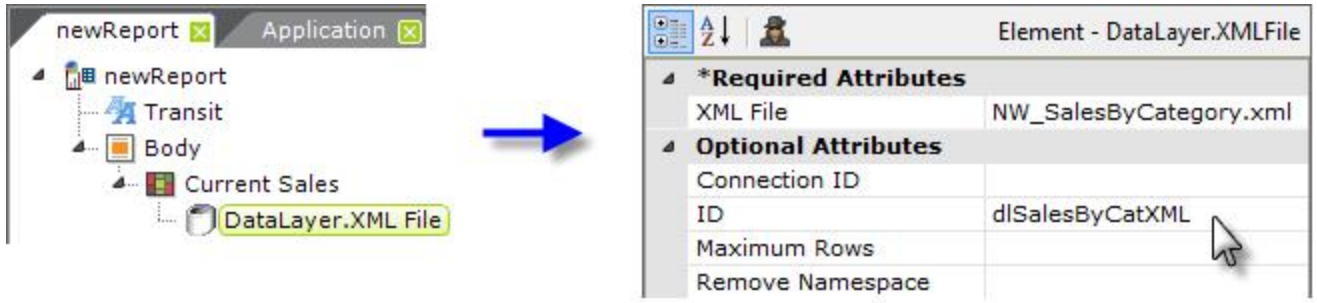
The remainder of this topic assumes you have created a Logi application with a connection to your datasource. The following examples use data from the Northwind database. As you will see, it's very easy to create a heat map.



1. In the Workspace editor, add a **Chart.Heat Map** element to the Body of your report, as shown above.
2. Set the element's attributes as shown above, right. For clarity, unused attributes are not shown. These are the key attributes:

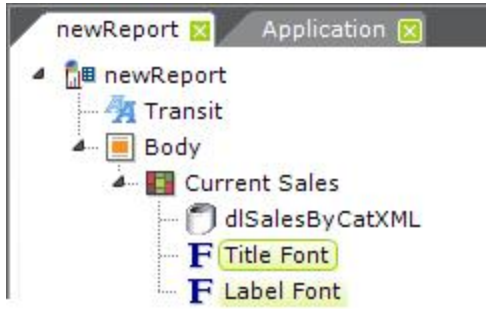
   Heat Map Color Data Column - The *color* of the chart squares will reflect product sales volume.
   Heat Map Label Data Column - The *label* within each square will be the product name.
   Heat Map Size Data Column - The *size* of the squares will also reflect product sales volume.

You can experiment later with some of the other attributes that control the borders and colors of the chart. Also, your choice of **Theme** will affect the colors used in the chart.



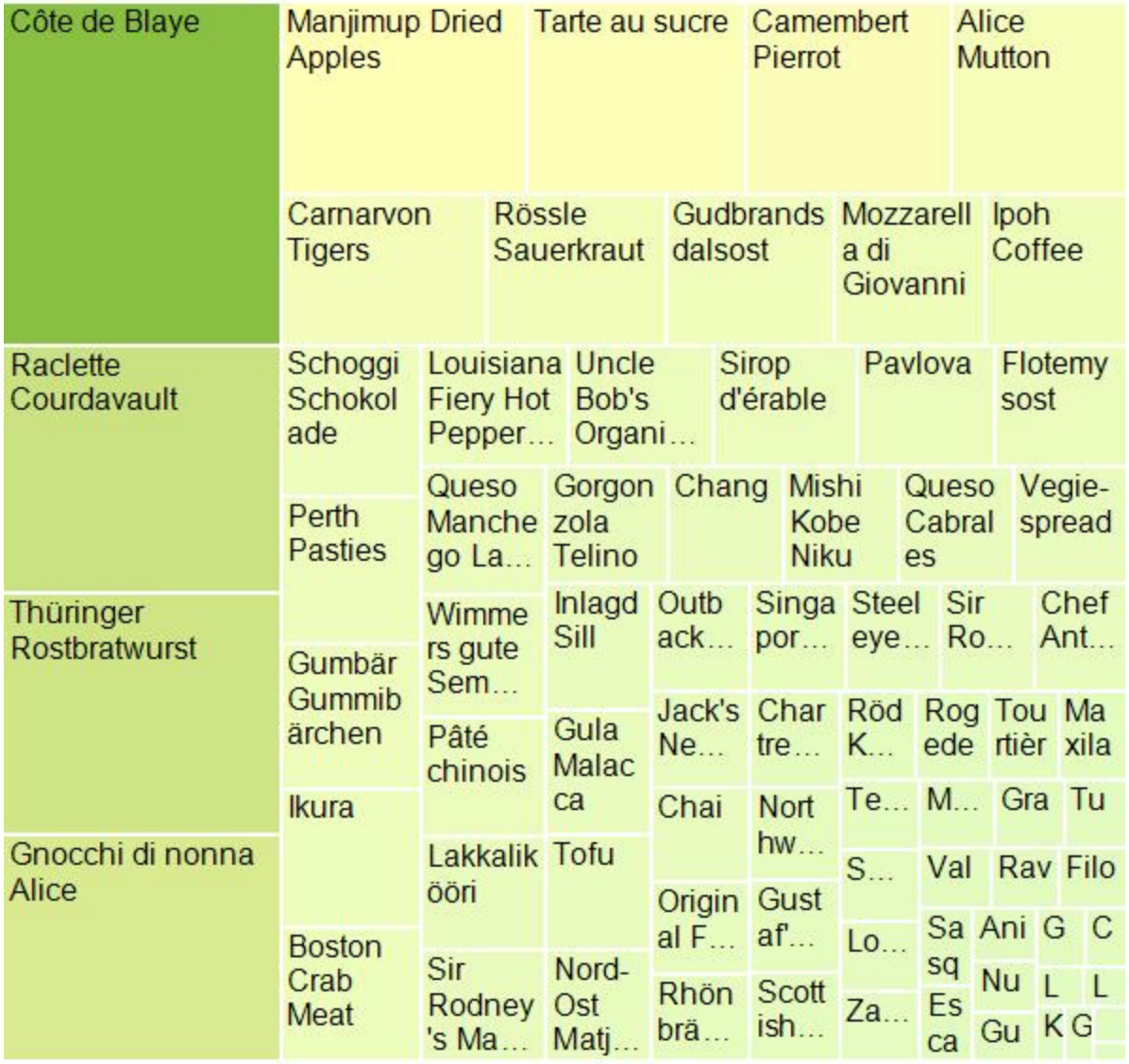
3. Next, add a **datalayer** element beneath the Chart.Heat Map element. We've used a DataLayer.XML File element in this example.

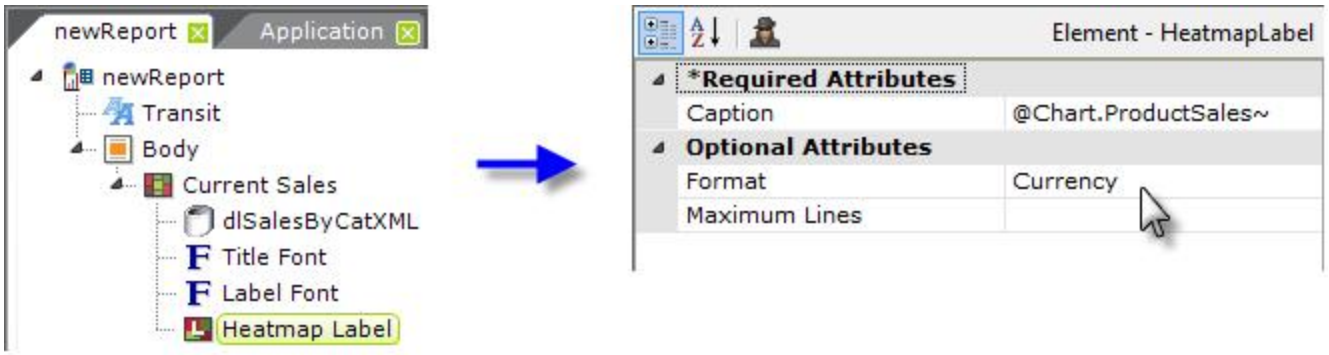
   A **Zero Rows Message** element is available for use as a child of Chart.Heat Map. This element displays a message of your choice *instead* of the chart when no data is returned into the datalayer.
4. Set the datalayer's attributes appropriately for the datasource. It's often useful to limit the number of rows returned, using the **Maximum Rows** attribute, during development so that the heat map isn't too confusing at first.



5. Optionally, you can add elements, as shown above, that allow you to format and align the overall chart title, and format the labels inside the chart squares.

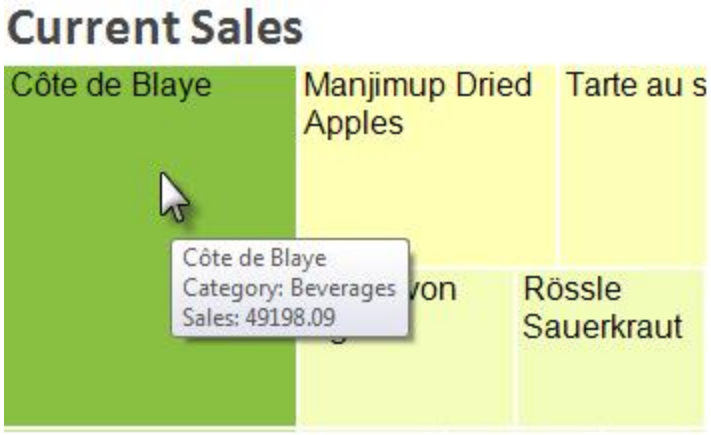
Current Sales

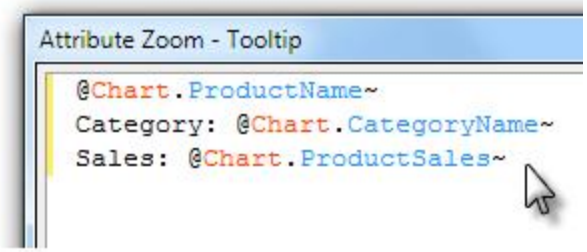


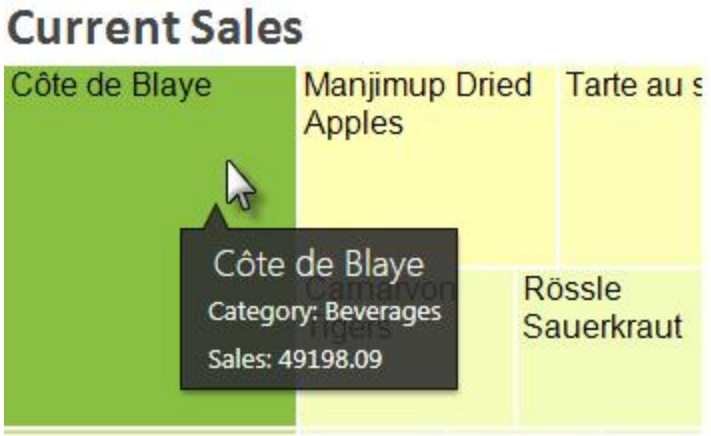


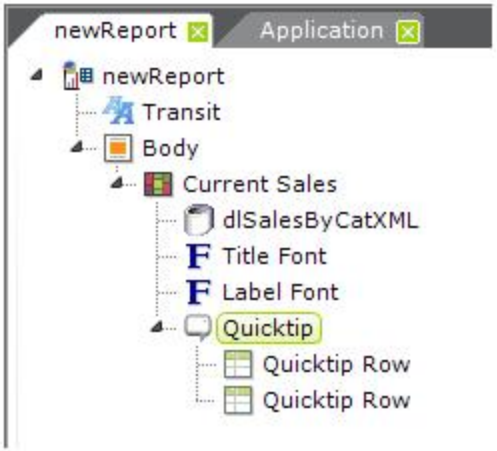
If you preview the chart at this point, you'll see something like the example shown above. You can display additional data right inside the chart's cells by using a **Heatmap Label** element, as shown above,...

... which produces the effect shown above. You can also add the **Resizer** element, if you like, which will give users the ability to resize the chart at runtime, or the **Hover Highlight** element, which lightens a cell's color to highlight it when the cursor is over it.

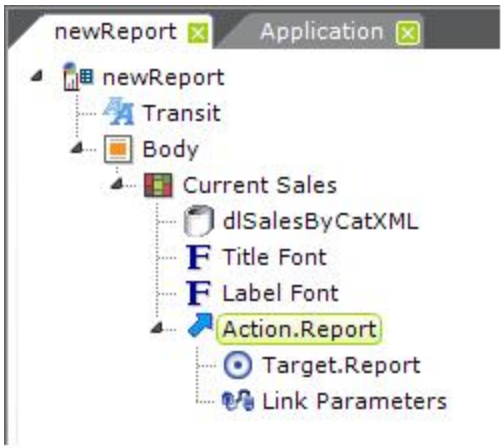


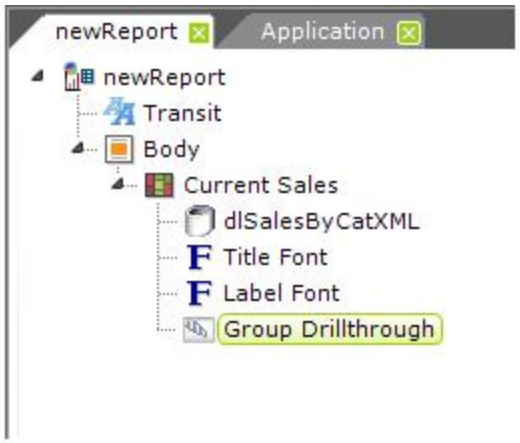




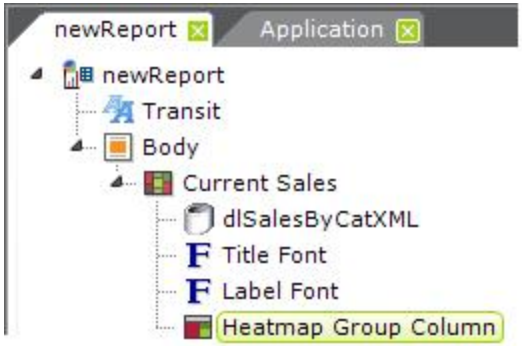


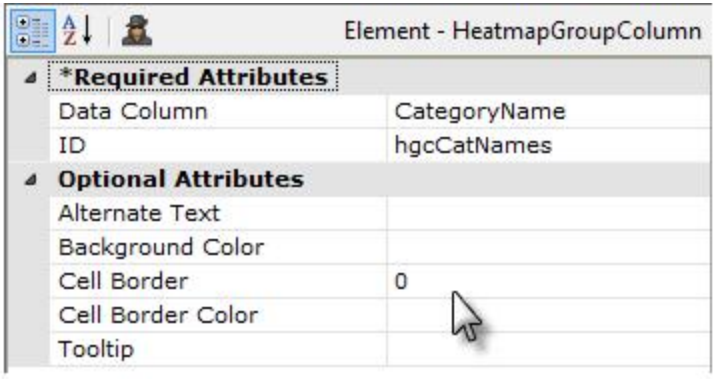
6. You can also use the Chart.Heat Map element's **Tooltip** attribute to provide additional information when the mouse is hovered over a chart cell, as shown above. @Chart tokens are used, and that placing them on different lines in the Attribute Zoom window causes them to appear on different lines in the tooltip popup. You can also use a **Quicktip** element, for a little different look, and get values on different rows in it using **Quicktip Row** elements.





7. In order to provide drill-down or drill-through functionality, you can add any of the **Action** or **Group Drillthrough** elements beneath the Chart.Heat Map element, as shown above. When passing parameters, remember to use @Chart tokens!





8. Finally, you can add one or more levels of grouping to your heat map, by using **Heatmap Group Column** elements, as shown above...

...and you can see the grouping effect in the example above. Tooltips and Action and Quicktip elements can also be used

beneath the Heatmap Group Column element to provide drill-down and drill-through functionality when groups are clicked.

Now, see what interesting uses you can come up with for this unique chart!

Polar Charts

The **Chart.Polar** element produces a circular graph in which data is displayed in terms of **values** and **angles**. Also known as a "radar chart", this is an X-Y graph drawn on a circular grid, displaying value trends on the basis of angles.

The following topics introduce the Chart.Polar element:

- Chart Types and Styles
- Attributes
- Supporting Elements

The Chart.Polar element has been deprecated; it's still supported and will work, but its elements are no longer available in Studio. Instead, use Chart Canvas Chart and set its Polar attribute for all new development.

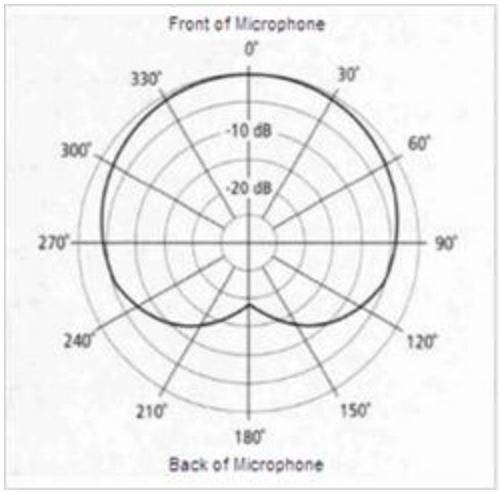
About Polar Charts

In a traditional line chart values are plotted along the **X** and **Yaxes**, with their positions designated by the familiar **Cartesian**, or horizontal and vertical, coordinates.

In a polar chart, data points are plotted using "**polar**" rather than Cartesian coordinates. In a polar chart, the **X-axis** value is used to set the **angle**, based on the values around the perimeter of the chart, in degrees. The **Y-axis** value is plotted along a **radius**, with 0 at the center of the chart.

A slight style variation provides the "radar" or "spider" chart, which **connects** its tick marks with **straight**, rather than curved, **lines** yielding a spider web-like appearance.

Multiple data sets can be plotted simultaneously as **layers**, with transparency helping users to "see through" overlapping areas. This allows quick comparison and assessment of data.



Polar charts are often very useful for present data that uses a natural cycle such as **daily temperatures** or **annual rainfall**. They're also frequently used in scientific and mathematic applications.

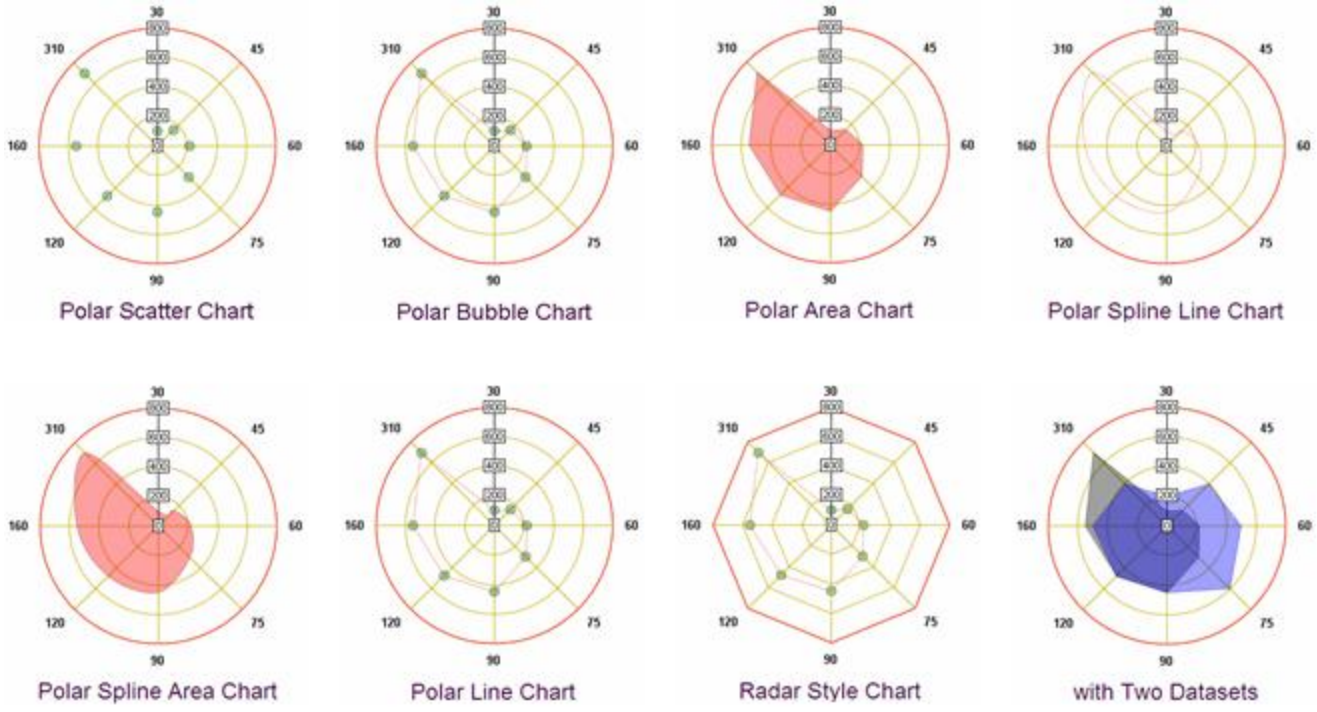
Consider the polar chart at the left. It shows the "sound pick-up pattern" for a particular type of microphone. The black line inside the chart plots how well the microphone picks up sound.

As you can see, the microphone is very good at collecting your voice if you stand **directly in front** of it.

As you move around to the **side** of the microphone, though, the sound pick-up capability starts to **diminish** and, if you stand at the **back** of the microphone, it's quite **poor**. Which is good, as this is what the microphone was designed to do; on a stage with other musicians and with an audience out front, this microphone will focus fairly exclusively on a singer's voice. And the polar chart shows this quite clearly.

Chart Types and Styles

The following images illustrate the **chart types** and **styles** available in the Chart.Polar element. All of the examples, except the last two, have been plotted with the same polar-style and a **single dataset**:



The last two examples show a "radar-style" chart and a polar-style area chart with **two overlapping datasets** (or layers). As you can see, Logi Studio provides a useful variety of styles with this new element. As with all Logi charting elements, special child elements are available to control the formatting of the borders, lines, areas, data labels, legends, and titles in polar charts.

Polar Chart Attributes

The Chart.Polar element has the following attributes:

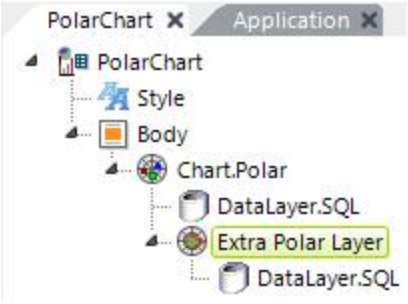
Attribute	Description
Background Color	Specifies the chart canvas background color. Enter a color by name, decimal RGB value, or hex RGB value. Prefix hex values with the pound sign, e.g. #112233.
Border Color	Specifies the chart canvas border line color. Enter a color by name, decimal RGB value, or hex RGB value. Prefix hex values with the pound sign, e.g. #112233.
Bottom Border	Specifies the distance, in pixels, between the bottom edge of the chart's image and the chart itself. Use a border to leave room for labels or a legend.
Chart Style	Specifies the style of Polar chart to be displayed *Polar* or *Spider* (Radar). The default style is *Polar*.
Chart Title	Specifies the title of the chart, which will appear at the top of the chart image.
Color	Specifies the chart color. Enter a color by name, decimal RGB value, or hex RGB value. Prefix hex values with the pound sign, e.g. #112233.
Data Column Y-axis	Specifies the name of a column in the datalayer that contains the value data.
Edge Color	Specifies the color of the line that borders filled areas. Enter a color by name, decimal RGB value, or hex RGB value. Prefix hex values with the pound sign, e.g. #112233.
Fill Color	Specifies the color of filled areas. Enter a color by name, decimal RGB value, or hex RGB value. Prefix hex values with the pound sign, e.g. #112233.
Grid Horizontal Color	Specifies the horizontal grid lines color. Enter a color by name, decimal RGB value, or hex RGB value. Prefix hex values with the pound sign, e.g. #112233.

Grid Vertical Color	Specifies the vertical grid lines color. Enter a color by name, decimal RGB value, or hex RGB value. Prefix hex values with the pound sign, e.g. #112233.
Height, Width	(Required) Size, in pixels, for the chart.
ID	Specifies a unique element ID.
Label Data Column	(Required) Specifies the name a column in the datalayer that contains angle data (in degrees).
Left Border	Specifies the distance, in pixels, between the left edge of the chart's image and the chart itself. Use a border to leave room for labels or a legend.
Legend Label	Specifies text that will be shown for this layer inside the chart legend.
Line Width	Specifies the width, in pixels, of the lines in the chart.
Maximum Label Length	Specifies the maximum number of characters that will be displayed for a label before the text is automatically trimmed and the remainder replaced with an ellipsis (...).
Outer Border Color	Specifies the chart canvas border color. Enter a color by name, decimal RGB value, or hex RGB value. Prefix hex values with the pound sign, e.g. #112233.
Outside Border Rounding	Specifies the value, in pixels, of the radii of the four corners of the chart canvas.
Plot Area Background Color	Specifies the plot area background color. Enter a color by name, decimal RGB value, or hex RGB value. Prefix hex values with the pound sign, e.g. #112233. You may also use *Transparent*.
Polar Chart Type	(Required) Specifies the type of Polar chart. Options include: *PolarLine*, *PolarScatter*, *PolarBubble*, *PolarArea*, *PolarSplineLine*, and *PolarSplineArea*.
Polar Radius	Specifies the radius, or size from the center to the perimeter, of a Polar chart, in pixels. If left blank, size is determined automatically based on

	data.
Right Border	Specifies the distance, in pixels, between the right edge of the chart's image and the chart itself. Use a border to leave room for labels or a legend.
Security Right ID	When using Logi Security, specifies one or more Security Right IDs, separated by commas. Users with these Security Right IDs will be able to view the chart, while users without them will not.
Show Wait Icon	Specifies whether or not to show the "wait" icon. Some charts may take a significant time to run on the server and display in the browser. The wait icon shows a spinning image informing the user that chart content is being processed. The default is *True;* set to *False* to disable the wait icon.
Size	Specifies the size, in pixels, of chart symbols. The default size is *4* pixels.
Symbol	Specifies the symbol to be used to identify points in the chart. Options include: *SymbolSquare*, *SymbolTriangle*, *SymbolX*, *SymbolCircle*, *SymbolCross*, and *SymbolDiamond*.
Tooltip	Specifies text that will appear when the user hovers the mouse pointer over the
Top Border	Specifies the distance, in pixels, between the top edge of the chart's image and the chart itself. Use a border to leave room for labels or a legend.
Transparency	Specifies the transparency of the chart color. The lowest value of *0* specifies that the line is opaque, with no transparency. At the other end of the scale, *15*specifies a completely transparent line. Use medium-level transparency to allow different chart layers to show through each other.

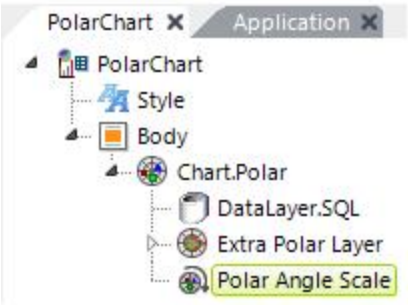
Supporting Elements

A number of elements make up the family used to create heat maps. Each element, its function, and its special attributes are discussed below:



Extra Polar Layer Element

Adds another "layer" (charted dataset) to the Polar chart, allowing multiple charts to be combined. Multiple layers in various chart styles (PolarArea, PolarLine, PolarSpline, etc.) are placed one on top of another in the same chart. The data for each layer comes from either columns in the main datalayer or from another datalayer beneath this element (shown). Attributes include: **Data Column Y-axis**- (Required) The name of the column in the associated datalayer that contains the value data for each record. **Polar Chart Type** - (Required) The type of Polar chart to be displayed: *PolarLine*, *PolarScatter*, *PolarBubble*, *PolarArea*, *PolarSplineLine*, or *PolarSplineArea*. **Legend Label** - Adds a label for this layer to the legend (if shown); a color-keyed symbol is automatically included. **Transparency** - Level of visual transparency for the plotted data in this layer. Values range from 0 (opaque) to 15 (completely transparent). Use medium-level transparency (8-10) to allow different chart layers to be seen through each other. Default value is 0.



Polar Angle Scale Element

The element controls the angular axis scaling options. Attributes include: **Fixed Scale Lower/Upper Bound** - Sets the lower and upper bound, in degrees, of the circular scale. Numbers outside of range 0-360 are ignored. If left blank, automatic scaling occurs based on the data. If Lower Bound is specified, an Upper Bound must also be specified. **Scaling Mode** - Determines the way in which angle (X-axis) data is plotted. *Standard* scaling plots the data in an enumerated manner; each point (row) returned in the datalayer is evenly spaced. This

works fine in many cases, especially when there are no missing data points. *Linear* scaling plots points based on an X-axis value; the X-axis is spread evenly end-to-end, even though the data points are not. Linear spacing works with either numeric or date/time values. **Linear Tick Increment** - When the Scaling Mode (previous attribute) is set to *LinearNumber*, this attribute controls how tick marks are created around the perimeter of the chart. Enter a number (in degrees) to specify the frequency with which tick marks will appear or, if working with date/time values, use one of the appropriate values such as *Year* or *Week*. Use *Auto* to automatically set an appropriate frequency based on the data.

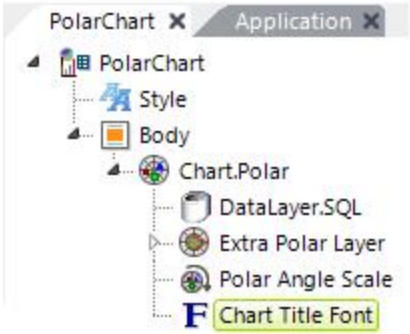
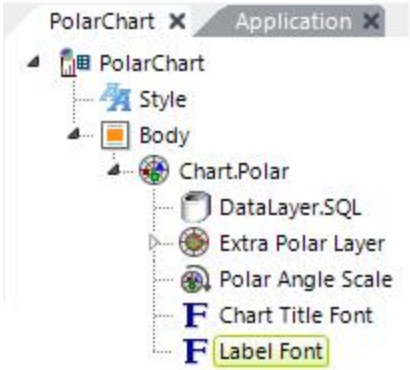


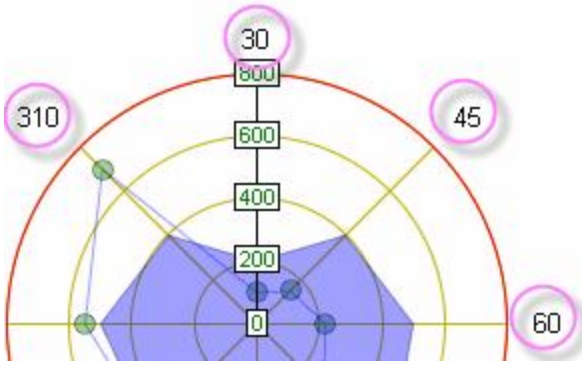
Chart Title Font Element

This optional element is available as a child to all chart elements. It controls the Font, Color, Size, and Angle (orientation) of the chart title. The title text itself is set in the Chart.Polar element.

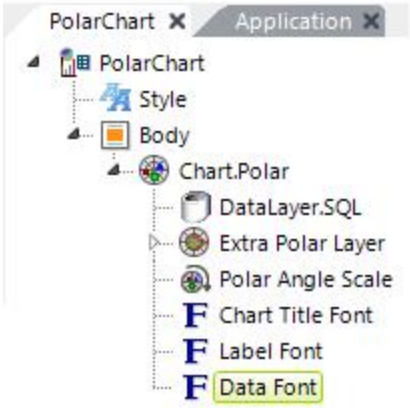


Label Font Element

This optional element is available as a child to all chart elements. In a polar chart, it controls the formatting (font, color, size, number of decimal points, and text orientation) of the angle (X-axis) data (shown circled in the image below).

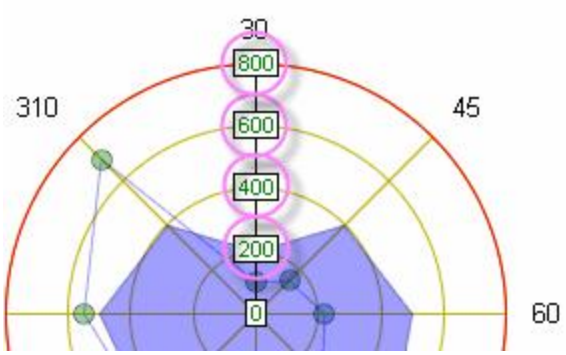


If this element is omitted or the Color attribute is left blank, the data appears in Black.

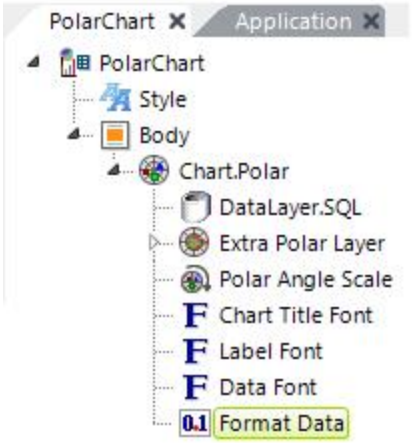


Data Font Element

This optional element is available as a child to all chart elements. In a polar chart, it controls the formatting (font, color, size, and text orientation) of the value (Y-axis) data (shown circled in the following image).

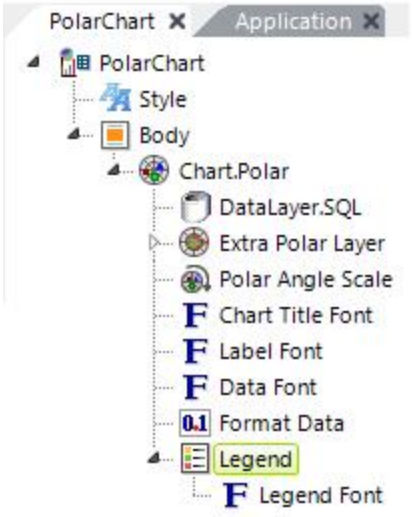


If this element is omitted or the Color attribute is left blank, the data appears in Black.



Format Data Element

This optional element is available as a child to all chart elements. It allows data values can be formatted for different decimal positions and European-style characters.

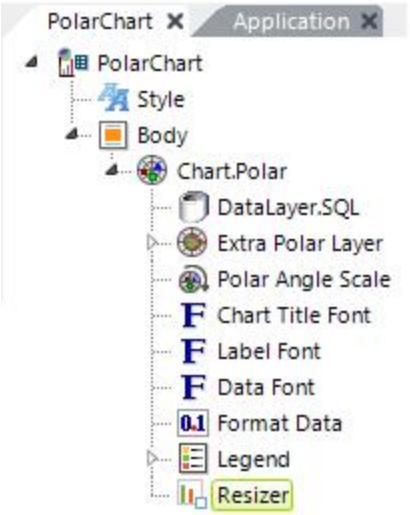


Legend Element

This optional element is available as a child to all chart elements. It displays a chart legend, in which each chart item is listed with a representative color and possible symbol. Attributes include top and left offset, border color, orientation, and background color. In a polar chart, the text for a legend entry must be set in the Chart.Polar and ExtraPolarLayer elements. This element's Legend Filtering attribute enables the ability to click legend entries at runtime and add or subtract their related objects from the chart.

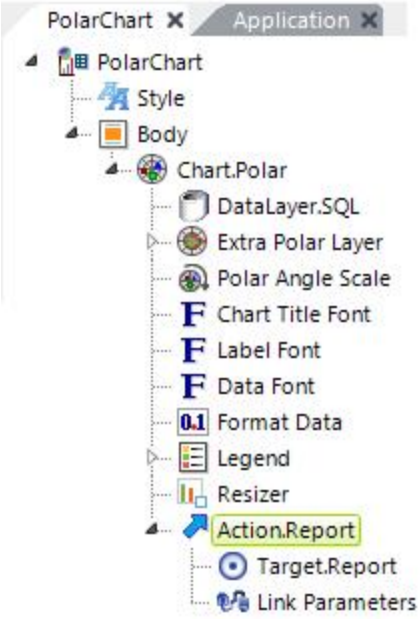
Legend Font Element

This optional elements controls the formatting (font, color, size, and text orientation) of the text in the chart legend.



Resizer Element

The Resizer element adds draggable controls to the bottom and right sides of the chart that allow the user to resize the chart at runtime. When the user completes the drag/resize operation, the chart is refreshed from the server. The new size is remembered for the chart for the current session. The default height and width of the resizer are set to 50 pixels each.



Action Element

This optional element provides click-event handling for the chart to allow a variety of actions such as redirection, process calls, etc. It works for the entire chart, not for individual data points.

Sparklines

Sparklines are tiny charts appropriate for showing many trends at once, as a set of small timelines. They are typically drawn without axes or gridlines and are often shown inside Data Tables, alongside labels.

The following topics discuss Sparklines:

- Providing Data for Sparklines
- Using Actions with Sparklines

About Sparklines

Whereas the typical chart is designed to show as much data as possible and is set off from the flow of text, Sparklines are intended to be succinct, memorable, and located where they are discussed. Logi Info provides four Sparkline elements:

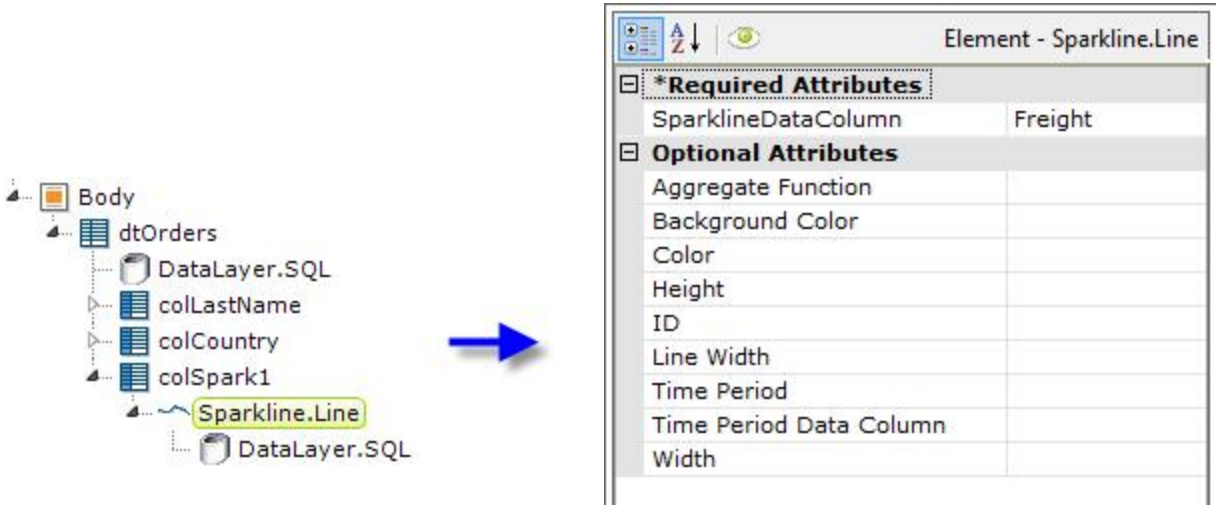
Element	Looks Like	Comment
Sparkline.Area		This element produces an area chart, and includes background, gradient, and edge color attributes.
Sparkline.Bar		This element produces a typical bar chart, with bar height varying based on data values. Negative values are displayed below the x-axis.
Sparkline.Line		This element produces a typical line chart, with a configurable line width.

Element	Looks Like	Comment
Sparkline.WinLoss		This element produces a chart similar to a bar chart, except the bar heights are all the same and simply indicate whether a data value is above or below a configurable threshold.

Sparklines are often used inside Data Table columns, but they can be used just about anywhere a regular chart can be used.

Providing Data for Sparklines

Sparklines are very easy to implement and use their own datalayers to retrieve data:



In the example shown above, a **Sparkline.Line** element has been included in a Data Table column. The Data Table has its datalayer and the Sparkline has its own, separate datalayer.

The data used by the Sparkline needs to be limited to that related to the table row that the Sparkline resides in. To do this, if using **DataLayer.SQL** to retrieve data for the Sparkline, the query can reference the data from the Data Table's datalayer using an @Data token. So the (Sparkline) query might look like:

```
SELECT * FROM orders WHERE EmployeeID = @Data.EmployeeID~
```

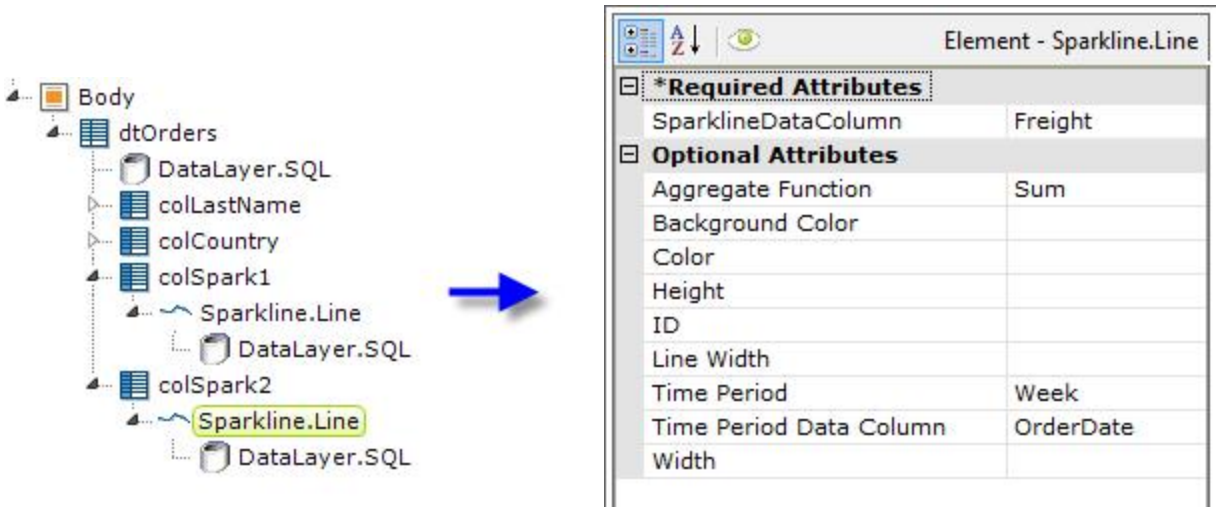
If the Sparkline data is retrieved from other datasources, a **Compare Filter** or **Condition Filter** can be used to limit the data in the datalayer to just that needed for the chart. For example, if you use a Compare Filter, its Data Column attribute value might be *EmployeeID* and its Compare Value attribute would be *@Data.EmployeeID~*.

All that's necessary to produce the Sparkline itself is to identify a data column in the **Sparkline Data Column** attribute value, as shown above, which provides the y-axis values. Each row in the datalayer is then plotted on its own chart.

Employee Name	Country	Sales
Nancy Davolio	USA	
Andrew Fuller	USA	
Janet Leverling	USA	
Margaret Peacock	USA	
Steven Buchanan	UK	

The resulting Data Table, with Sparklines, is shown above. Height and width values may be specified but, if they are not, they will be determined automatically.

Common implementations of Sparklines show data that's grouped by time period and aggregated. Sparkline elements include time-oriented grouping attributes and a built-in aggregation function, so you don't have to manipulate the datalayer data using additional elements (although you *may* do so).



In the example shown above, the **Sparkline Data Column** and **Aggregate Function** attributes are used to sum the *Freight* data column values, which provides the y-axis values. There are two attributes for grouping data by time, the **Time Period** and **Time Period Data Column** attributes, which produce the x-axis values of the chart. In the example, they're set to *Week* and the *OrderDate* data column, respectively.

Employee Name	Country	Sales	Sales by Week
Nancy Davolio	USA		
Andrew Fuller	USA		
Janet Leverling	USA		
Margaret Peacock	USA		
Steven Buchanan	UK		

The resulting Data Table is shown above. This, essentially, allows you to increase the granularity of the Sparkline.

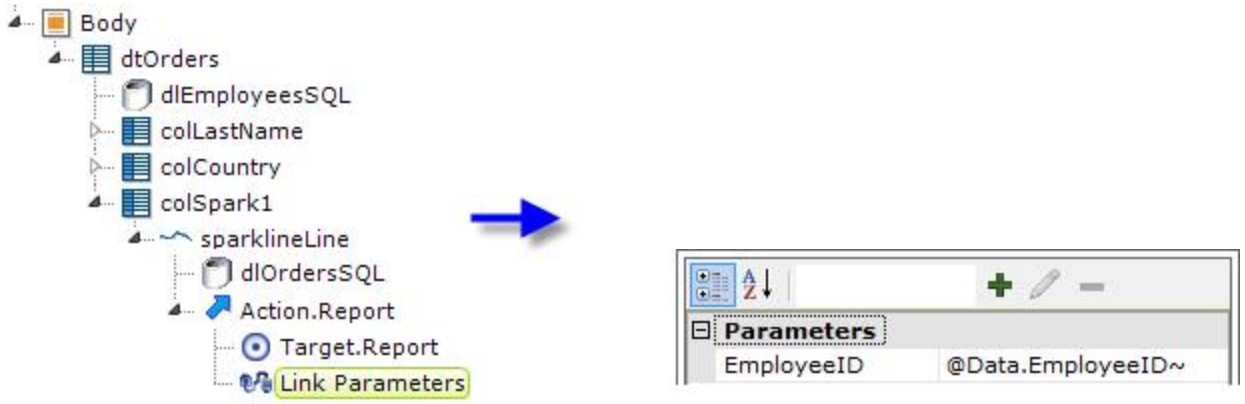
The **Time Period** attribute will only work with *date-related* values; even though they're shown in Studio, the Hour, Minute, and Second options are not valid with this element. Similarly, data in the column specified in the **Time Period Data Column** must be DateTime type data that includes date values, not just time values.

However, if you wish to display values grouped by time, add a **Time Period Column** element under your datalayer. Use a **Group Filter** to group the data on the Time Period Column and a **Group Aggregate Column** element to aggregate it. Enter the ID of the Group Aggregate Column element in the Sparkline element's Sparkline Data Column attribute.

Aggregations with Null values are *excluded* by default. To include them, create the constant `rdCalculationsIncludeNulls`and set it to *True.*

Using Actions with Sparklines

Sparklines can be made interactive by using **Action** elements with them:



In the example shown above, **Action.Report**, **Target.Report** and **Link Parameters** elements are used beneath a Sparkline element. Any of the following Action elements can be used:

- Action.Report
- Action.Link
- Action.Process
- Action.Javascript

and additional Action elements will likely be added in the near future. Link Parameters associated with these actions can reference both the @Data tokens (the Data Table data) and @Chart tokens (the Sparkline data) available at runtime.

Text Clouds

A "Text Cloud" is a visual depiction of the **weighting** of words, links, and phrases. The relative font size and color of words or phrases within the text cloud are an indication of their weighting or frequency.

The following topics discuss the elements provided in Logi Info that make working with text clouds very easy and straightforward:

- Creating a Text Cloud
- Customizing Text Cloud Appearance
- Adding Links

About Text Clouds

Text clouds (and their close cousins, **tag clouds**) are groupings of words that are visually differentiated by font size, color, etc. in order to reflect their relative weighting, frequency, or other numeric value. The words or phrases may be links, allowing users to drill-down into detail data. The text cloud is similar to another, better known visual depiction of relative weightings, the standard **pie chart**, but it allows for many more weightings, while providing somewhat less accuracy. If the text cloud is made up of individual words, their linear organization can be **alphabetical**, introducing yet another organizational scheme.

Creating a Text Cloud

Text clouds are easy to create; the data required consists of the words or phrases and a numerical weighting or frequency value. The following example application will show developers how to use the appropriate Logi Info elements to create a text cloud.

1. In Logi Info Studio, create a new application or a new report definition in an existing application.
2. Log in to DevNet.
3. Download the sample data for this exercise. If it opens in your browser, right-click it and select "View Source" or a similar source-related option. Then save it to your application's **_SupportFiles** folder as `DevNetSearches.xml`. These are the search terms and their frequencies from the DevNet site search log for a few weeks, in an XML file.

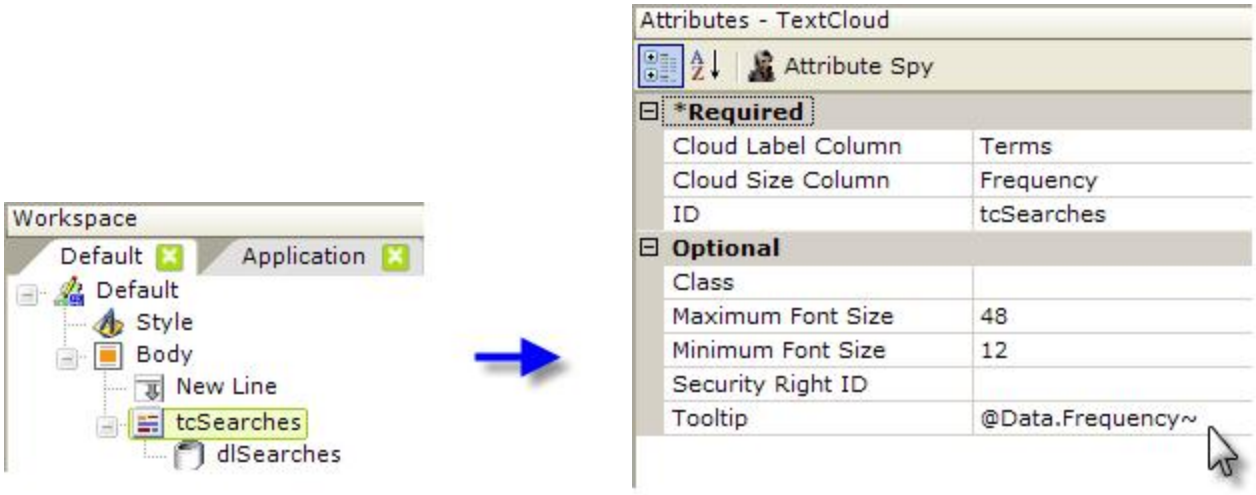
```
<ItemTerms="Connection.ODBC" Frequency="12" />
<ItemTerms="InputDate" Frequency="14" />
<ItemTerms="Show Modes" Frequency="23" />
<ItemTerms="Body" Frequency="27" />
<ItemTerms="Report" Frequency="92" />
<ItemTerms="Procedure.Script" Frequency="11" />
<ItemTerms="Chart.Pie" Frequency="38" />
```

The fragment above shows some of the data that's in the sample file. Notice that it contains two data attributes "Terms" and "Frequency". These are the data "columns" you'll be referring to later.

```
/* TextCloudStyles.css */BODY {
BACKGROUND: #ffffff;
font-family: veradana, arial;
}A:link {
text-decoration: none;
```

```
}A:hover {
text-decoration: underline;
}.fontBlue {
color: blue;
}.fontGreen {
color: green;
}.fontOrange {
color: orange;
}
```

4. In Studio, under the **_SupportFiles** folder, add a new style sheet to your application and give it a name of your choice. Copy the class definitions shown above into it, save it, and assign the style sheet to your report definition's **Style** element.



5. From within the Charts, Gauges and GIS Maps folder in the Element Toolboxpanel, add a **Text Cloud** element beneath the Body element of your report, as shown above.

6. Set the **Text Cloud** element's **Cloud Label Column** attribute value to the name of the data column containing the actual words or phrases to be displayed. In the case of the sample XML data, this column is "Terms".
7. Set its **Cloud Size Column** attribute value to the name of the data column containing the numeric weighting of each word or phrase. In the case of the sample XML data, this column is "Frequency".
8. Assign a unique **ID** to the **Text Cloud** element (entering a value here is required).
9. Set the Text Cloud's **Max** and **MinFont Size** attributes. These sizes, in **pixels**, will be assigned to the words or phrases with the largest and smallest weighting values. Words with values in between the maximum and minimum will be given font sizes proportionate to their weighting value.
10. Set the **Tooltip** attribute to the value of the Frequency data column. This will cause the actual weighting value to be displayed when the mouse hovers over any word or phrase.
11. Beneath the **Text Cloud** element, add a **DataLayer.XML File** element. Set its **ID** attribute to some unique value and set its **XML File** attribute to the name of the XML file you downloaded back in Step #2: `DevNetSearches.xml`.

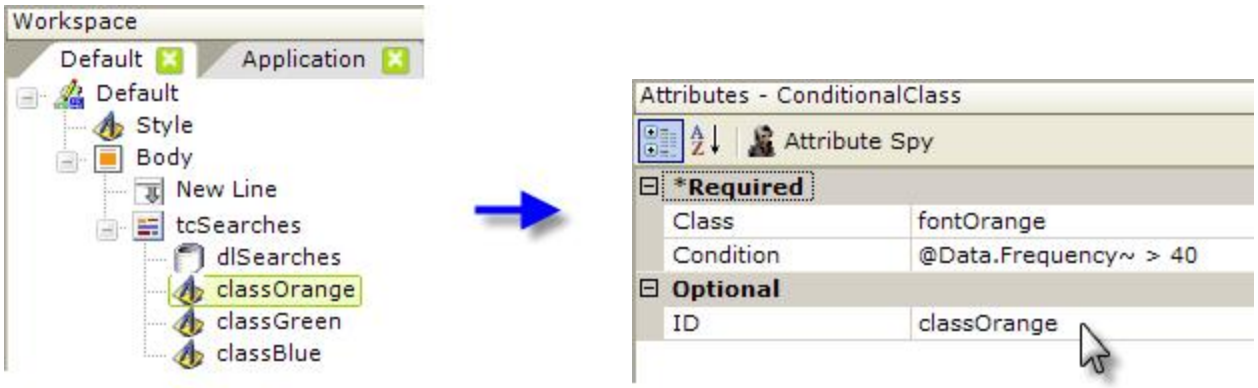
Save and preview your report and you should see the text cloud shown above. When you **hover** your mouse over the words, their actual frequency in the search log should appear as a tool tip. Notice how the *size* of the words is related to their frequency. We used an XML file as the data source for this example, but any type of datalayer element can be used to retrieve the data.

Formatting Data Using Calculated Column

As with other Logi chart elements, a **Calculated Column** element can be used to manipulate actual data and create a pseudo-column in the datalayer that is then used in the text cloud. For example, if your data had separate **first** and **last name** columns but you wanted to display them as a single, combined name in the text cloud, you'd add a Calculated Column element beneath your Text Cloud, set its **Formula** attribute to combine the two name columns (`@Data.FirstName~ @Data.LastName~`) and then use the Calculated Column element's **ID** as the value for your Text Cloud element's **Cloud Label Column** attribute.

Customizing Text Cloud Appearance

We can introduce additional visual differentiations, such as color, font style, etc., to make the weightings stand out more and to make the text cloud more visually interesting. The following example shows you have to use **Conditional Classes** to give the text cloud fonts different colors based on weighting.



1. Beneath the **Text Cloud** element ("tcSearches") add three **Conditional Class** elements, as shown above.
2. Set the first Conditional Class element's attributes as shown above. The **Condition** attribute value allows you to apply the class based on the value of the weighting. If the expression in the value evaluates to *True*, the class will be applied. In this case, all words or phrases with a frequency greater than 40 will be shown in orange.
3. Set the attributes for the other two Conditional Class elements similarly. Their attribute values should be:

Class	Condition	ID
`fontGreen`	`(@Data.Frequency~ > 30) AND (@Data.Frequency~ <= 40)`	`classGreen`
`fontBlue`	`(@Data.Frequency~ > 14) AND (@Data.Frequency~ <= 30)`	`classBlue`

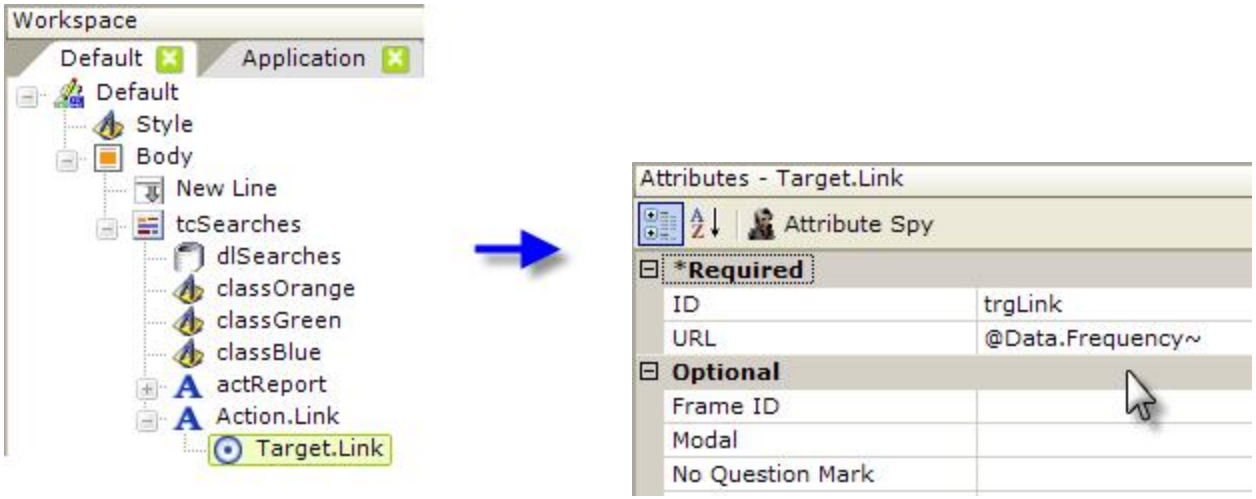
The classes assigned in the Class attributes for these elements are all defined in the style sheet you created in Step #4 in "Creating a Text Cloud" on page 307.

Conditional Class elements are evaluated *in sequence* and that the first class to evaluate to *True* will be applied and remaining Conditional Class elements will be ignored. This means that the *order* of your conditional class elements in the report definition may be significant, depending on your evaluation formulae.

Preview your report and you should see the text cloud shown above. (Any reddish quality you may see here is an artifact from the image reduction process used to fit the image to this page - only four colors: orange, green, blue, and black should be visible in your Preview Panel).

Adding Links

The text cloud can also include **links** that allow a **drill-down** capability. The sample data you used does not include a URL column, but we'll fake it to illustrate how this feature works:



1. Add **Action.Link** and **Target.Link** elements beneath your Text Cloud element, as shown above (any Action element is valid here.)
2. Set the **Target.Link** element's **URL** attribute to the column in your data that has the appropriate drill-down destination. Enter the value shown above (`@Data.Frequency~`) just so that we don't get an error when we run the preview as a URL value is required.

Preview your report. Now when you hover your mouse over a word or phrase, it should be underlined and become an active link, as shown above.

The behavior of the links (the underline only appears when hovering over one) in this case is controlled by the two link-related classes in the style sheet you created earlier.

The Text Cloud is an interesting method of displaying and comparing text-type data and, as you've seen, it's very easy to implement in Logi Info.

Animated Maps

Logi Info includes **Animated Maps**, which are animated, interactive, data-driven, JavaScript-based maps, for your applications. Data range coloring makes it easy to relate data within specified thresholds to map regions.

The following topics discuss the use of Animated Maps:

About Animated Maps

Logi Info's **Animated Maps** are animated, interactive, data-driven, JavaScript-based maps for Logi applications. Developers can use them to display geographical data distributed by category, regions, or entities. The best usage examples include statistical display of data, flight routes, office locations, election results, survey results, or business intelligence such as "Revenue by Regions" or "Revenue by States". They *do not* work with **zip codes** or **geocoded data** (latitude- & longitude-based). Like all Logi data visualizations, map portions can be interactive links that can be clicked in order to "drill-down" to other reports, charts, and URLs. These maps *do not* require Adobe's Flash Player browser-extension be installed in order to view them. Maps for 198 countries and regions are included with Logi Info; see the list of available maps below. In almost all of these maps, their subdivisions (regions, provinces, states, etc.) are referenced by aN internal numeric ID. In order to relate these ID numbers to "real world" names more likely to be encountered in your data, such as "Montana" or "Belgium", each map has a companion XML data file that can be used to cross-reference a region name to the internal ID. This is discussed in detail later in "Referencing Data" on page 322. Animated

Maps can be exported to PDF. Prior to this version, they could not be exported to PDF. Animated Maps *cannot be exported* to other formats, such as Word and Excel.

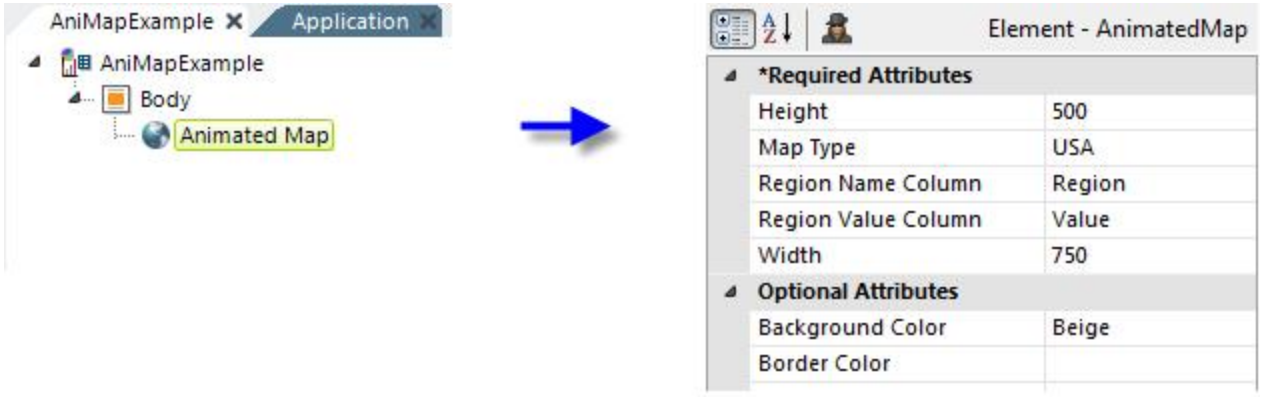
Adobe no longer supports their Flash Player, and has blocked Flash content from running in a Flash player since January 12, 2021. Select this link for more information.

Sample Application

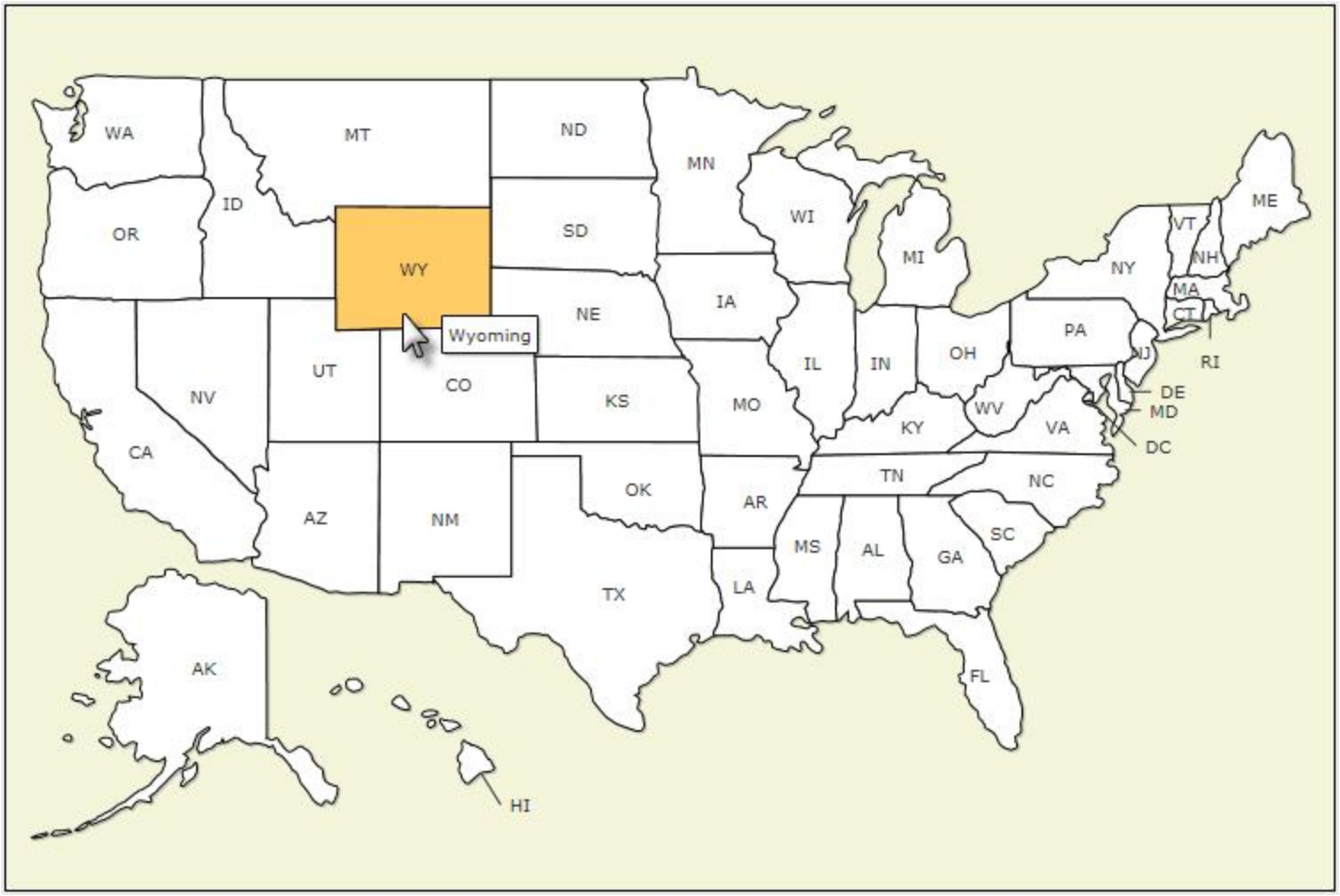
A sample application can be found on the DevNet Sample Center page.

Animated Map Element

Animated Maps are included in Logi Info applications by adding the **Animated Map** element to a report definition.



In the example above, an Animated Map element has been added and its attributes set as shown.



The resulting map is shown above. When the cursor passes over a state, its color changes and a "tooltip" appears with the full state name. The attributes for the Animated Map element are explained below. Unless otherwise noted, attribute values are optional.

Attribute	Description
Height	(Required) Specifies the height of the map, in pixels.
Map Type	(Required) Specifies the type of base map to be rendered; e.g. World, USA, China, USARegion etc. A complete list of base maps is provided at the end of this topic. Specific map type values can be found by examining the .js files in the application folder's rdTemplate\rdFusionMap folder. The India map, for example, is called `FusionCharts.HC.india.js` and the country or state name portion "india" is the value that's entered in this attribute.
Regional Name Column	(Required) Specifies the name of a column in the datalayer, with data representing the **name/ID** for each region inside a map. If a datalayer is not being used, enter an arbitrary placeholder value.
Regional Value Column	(Required) Specifies the name of a column in the datalayer, with data representing the **value** for each region to be mapped. If a datalayer is not being used, enter an arbitrary placeholder value.
Width	(Required) Specifies the width of the map, in pixels.
Background Color	Specifies the color of the map background; can be either a color name, a decimal RGB value, or a hex RGB value prefixed with a pound sign (#112233). "Transparent" is a valid value.
Border Color	Specifies the color of the map border lines, can be either a color name, a decimal RGB value, or a hex RGB value prefixed with a pound sign (#112233). "Transparent" is a valid value.
Color	Specifies the default color of the map; can be either a color name, a decimal RGB value, or a hex RGB value prefixed with a pound sign (#112233). "Transparent" is a valid value.

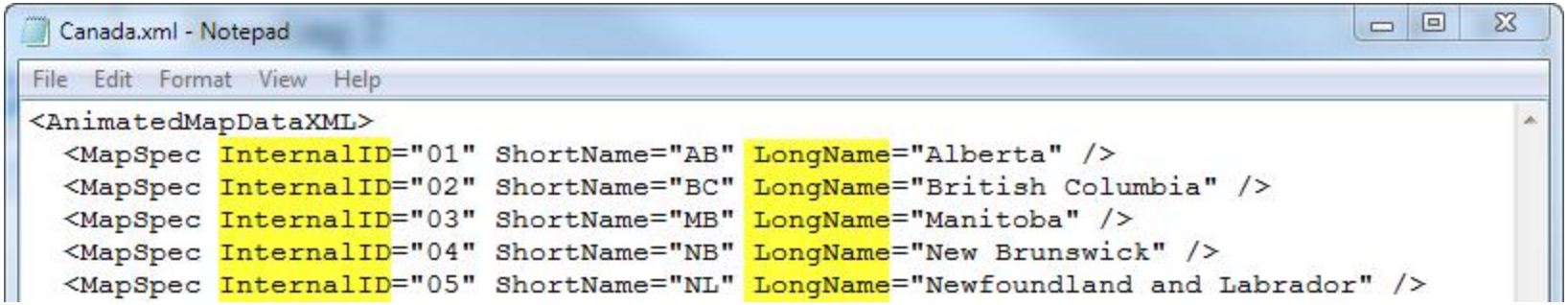
Attribute	Description
Font	Specifies the family name of font in the map. Can be followed by a space and "bold" or "italic".
Font Color	Specifies the color of the font in the map; can be either a color name, a decimal RGB value, or a hex RGB value prefixed with a pound sign (#112233).
Font Size	Specifies the size, in points, of the font in the map.
Hover Color	Specifies the background color for a map region when the mouse pointer is hovered over it; can be either a color name, a decimal RGB value, or a hex RGB value prefixed with a pound sign (#112233).
ID	Specifies the element's unique identifier.
Include Value in Labels	If working with a datalayer, set this value to *1* to have data values appear in the labels along with region identifiers.
Outer Border Color	Specifies the Specifies of the outer border of the chart canvas; can be either a color name, a decimal RGB value, or a hex RGB value prefixed with a pound sign (#112233).
Show Labels	Set this to *False* if you do not want the labels to be displayed. Default: *True*
Tooltip Column	An automatically-generated tooltip is displayed when the cursor is hovered over each map region, consisting of the data in the Regional Name Column and Regional Value Column attribute values specified earlier, separated by a comma and space. This attribute, Tooltip Column, specifies the name of a datalayer column containing the text for a custom tooltip, which will be shown instead of the automatic tooltip if a value is entered here. You can specify the name of a column returned from the datasource, or that of a Calculated Column

Attribute	Description
	you add to the datalayer to create more customized text.

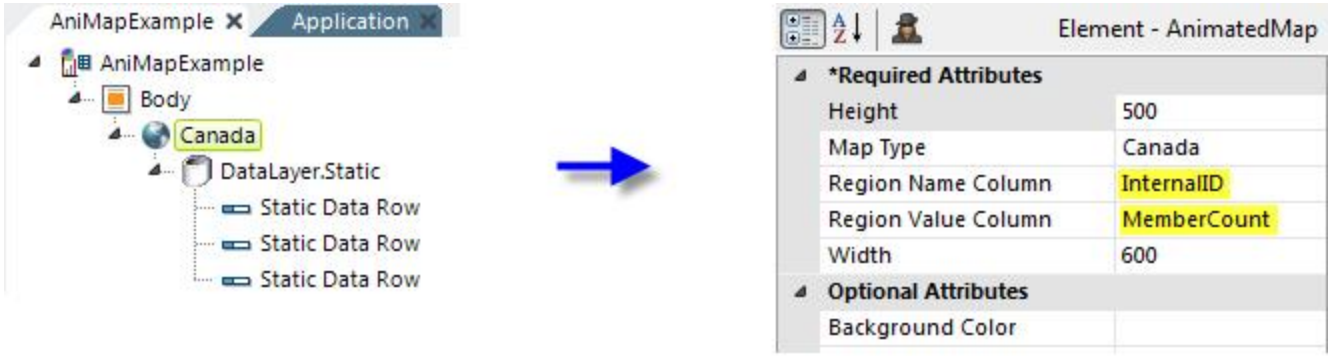
Referencing Data

It's nice to be able to show maps, but the Animated Map element becomes really useful when it associates data with map regions. To do this, a datalayer element is added to the definition. In the following example, a map of Canada will be produced, showing its provinces with their abbreviations. When the cursor passes over a province, a "tooltip" will appear showing the full province name and the number of members in a mythical book club chapter from that province.

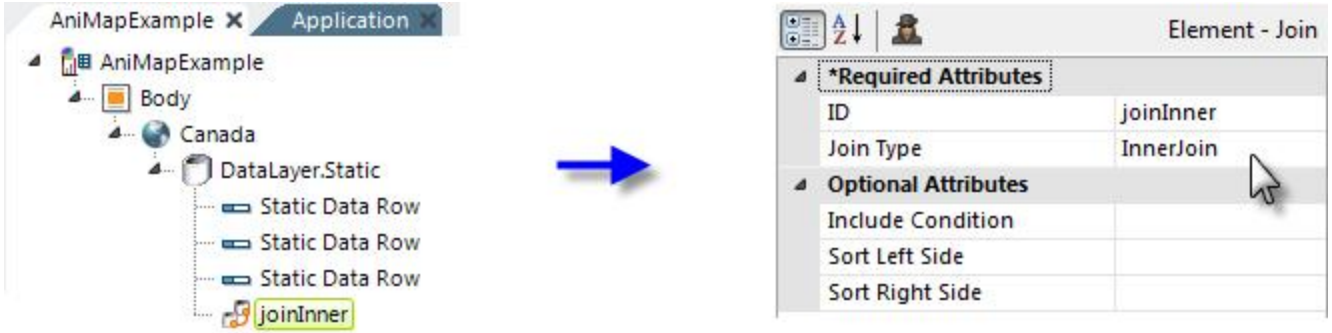
In the example definition above, a **DataLayer.Static** element has been added as a child of the Animated Map element, and some **Static Data Row** elements have been added to provide data values. Naturally, depending on the datasource, other types of datalayers can be used here.



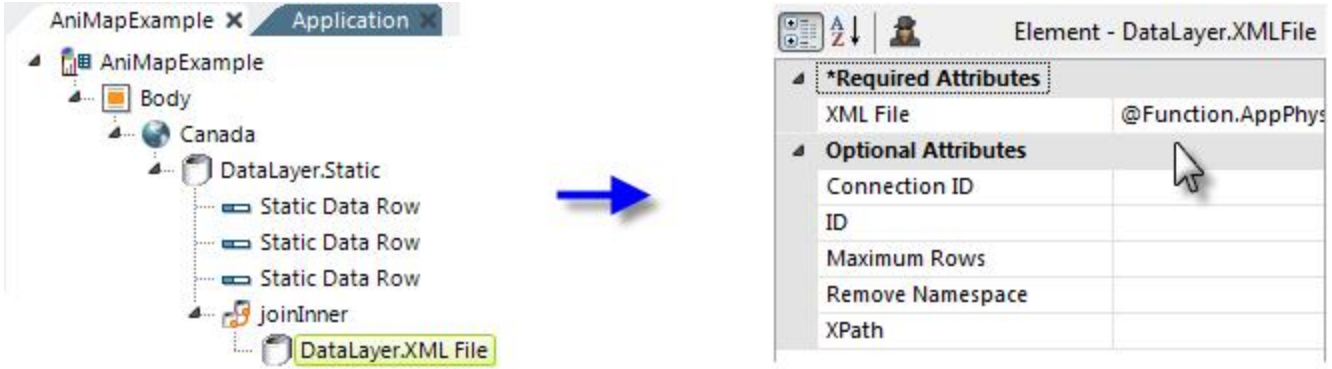
A fragment of this map's companion XML data file, Canada.xml, is shown above. Note its highlighted attributes, "InternalID" and "LongName". In order to relate the **Province** value in our data to the appropriate region in the map, a datalayer **Join** can be used between the data and the companion XML file. Here's how:



First, as shown above, the Animated Map element's **Region Name Column** attribute value is set to the InternalID in the companion XML file. The **Regional Value Column** attribute is set to the MemberCount value in the static data.



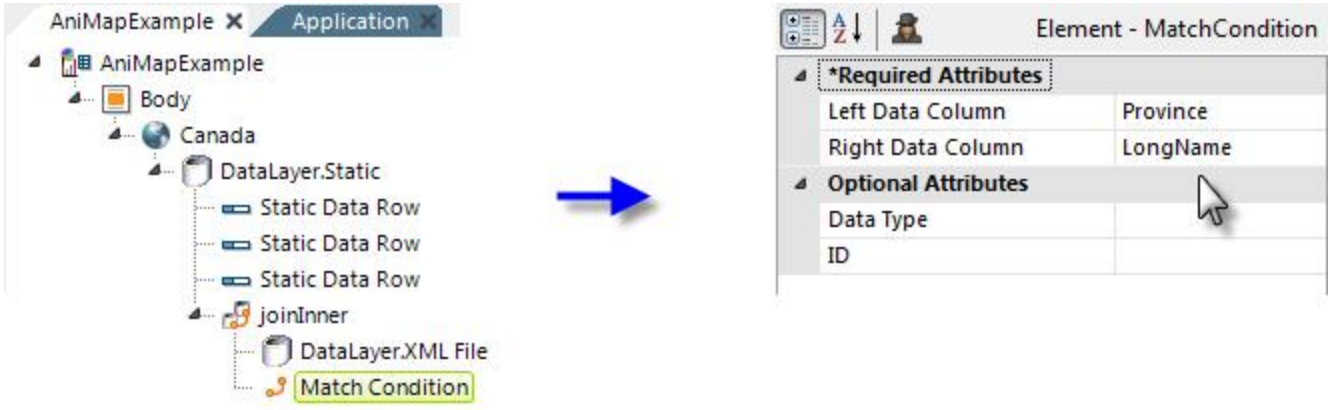
Next, a **Join** element is added, as shown above. The Join element performs the equivalent of a SQL JOIN on two datalayers. In this case, an Inner Join will used between the static datalayer and...



...a new **DataLayer.XML File** element which is added. This datalayer retrieves the data from the map's companion XML file and its **XML File** attribute value is shown but has been truncated to fit on this page. The complete XML File attribute value would be:

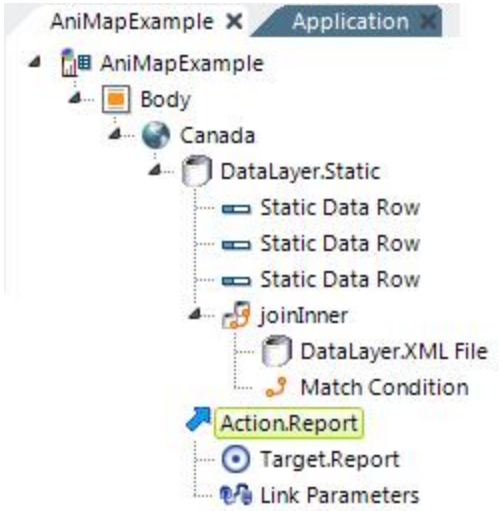
```
@Function.AppPhysicalPath~\rdTemplate\rdFusionMap\Canada.xml
```

The companion XML file names correspond to those of the map type .js files (without the "FusionCharts.HC." prefix and with an .xml file extension). So, for example, the file `FusionCharts.HC.canada.js` corresponds to the companion XML file `Canada.xml`.



Finally, a **Match Condition** element is used to provide the JOIN criteria: the Province column value in the static datalayer must match the LongName column value in the companion XML file. As in any JOIN, the choice of which column is on the left and which is on the right is significant. Generally, the Left Data Column comes from the datalayer *above* the join and the Right Data Column from the datalayer *below* it.

Now when the map is displayed, as shown above, and the cursor is placed over a region, by default the **LongName** and **MemberCount** information appear in a tooltip.

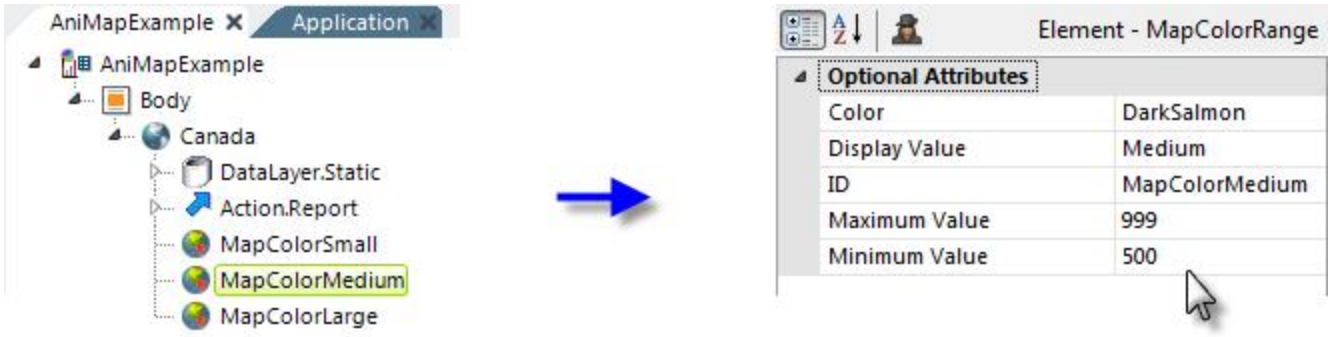


"Drill-down" capability can be added by placing an **Action** element, as shown above, beneath the Animated Map element. Region-specific data can be accessed using **@Chart** tokens and passed, as Link Parameters, to the next report or process.

The two-letter "ShortName" values in the XML data file were assigned by the map component supplier and *may not be* ISO standard country codes. If you wish to use ISO codes, you can edit the data file to use them (you may need to restart your server afterwards to clear any caching).

Adding Color Ranges

Once data values have been associated with them, individual regions in an Animated Map can be color-coded based on those values.

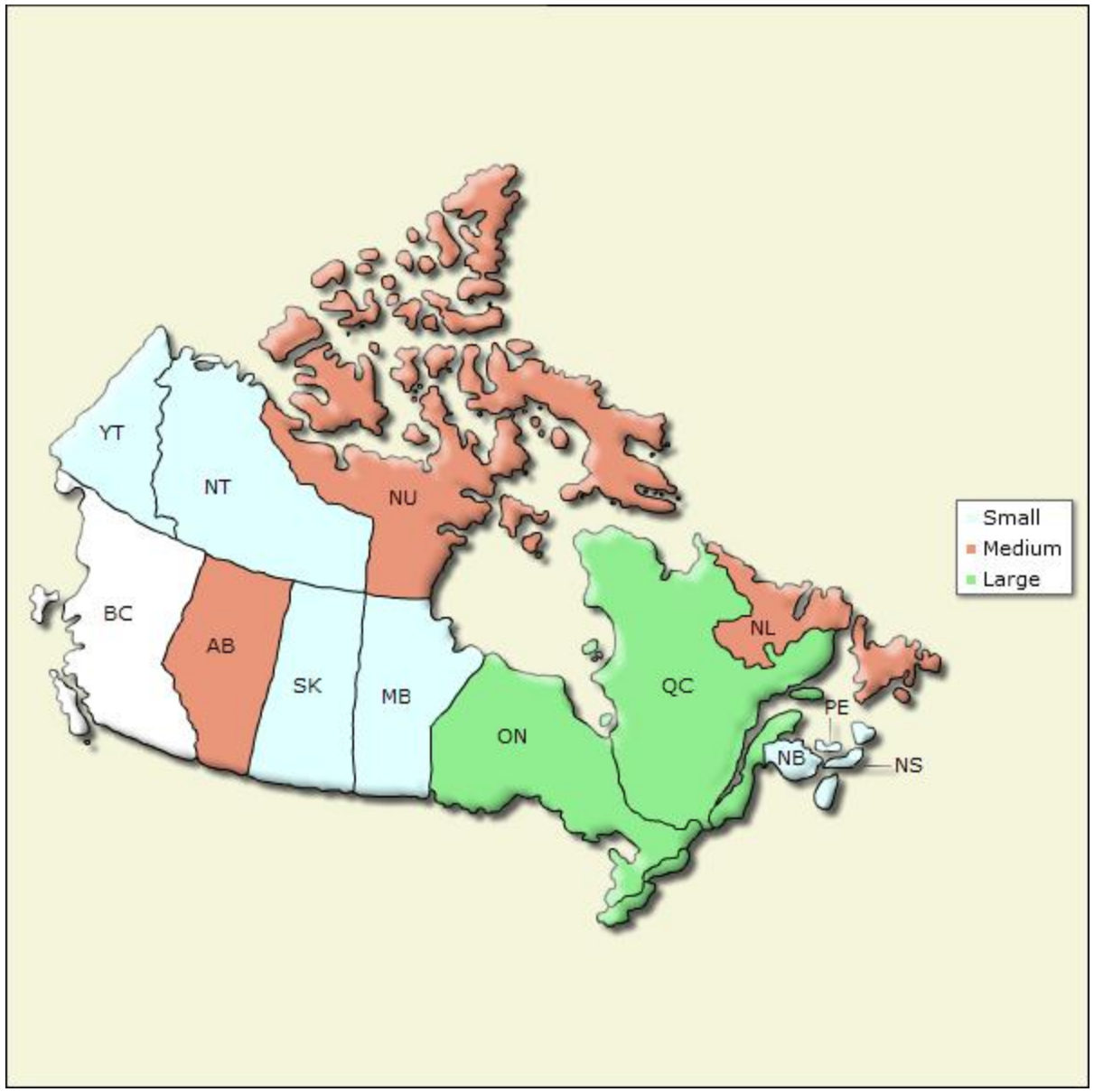


This is accomplished, as shown above, by adding one or more **Map Color Range** elements beneath the Animated Map element. Each element represents one color, which is set in its **Color** attribute. The value of this attribute can be set using @Session, @Request, and @Local tokens, but not @Data or @Chart tokens. A legend is automatically generated for the map and the text in the **DisplayValue** attribute is used in it. The assignment of a color to a particular map region is controlled by the **Max Value** and **Min Value** attributes; the color will be assigned if the data value is within this range. These values correspond to the data value identified in the Animated Map element's **Region Value Column** attribute.

Note this important distinction: in order to prevent overlapping ranges, the color is assigned if the data is *equal to* or *greater than* the **Min Value** attribute value, and *less than* the **Max Value** attribute value, and you need to provide your range values accordingly. For example,

Data Values	Min Value (= >)	Max Value (<)
1 - 5	1	6
6 - 10	6	11
11 - *n*	11	

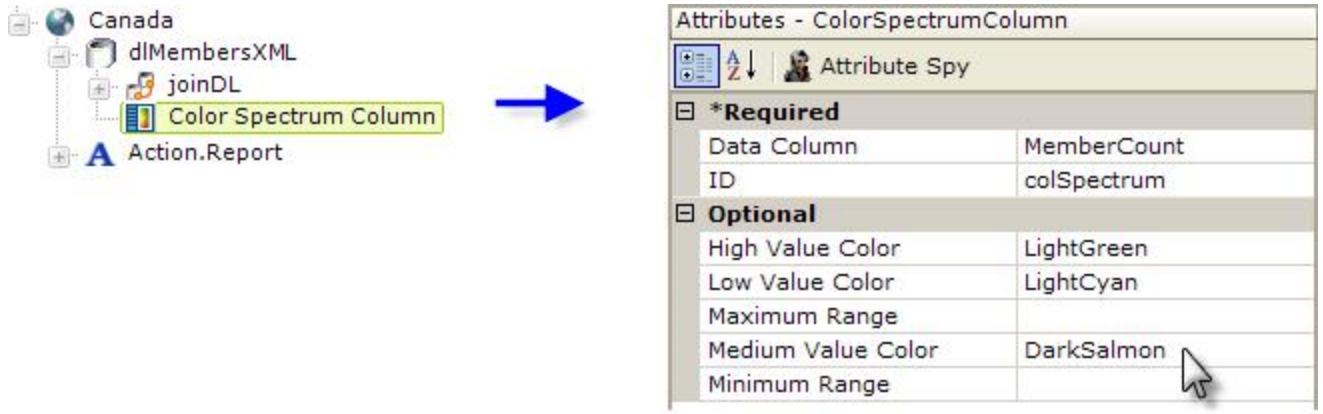
will accurately handle a data value of 10.5


YT
NT
NU
BC
AB
SK
MB
ON
QC
NL
PE
NB
NS
Small
Medium
Large

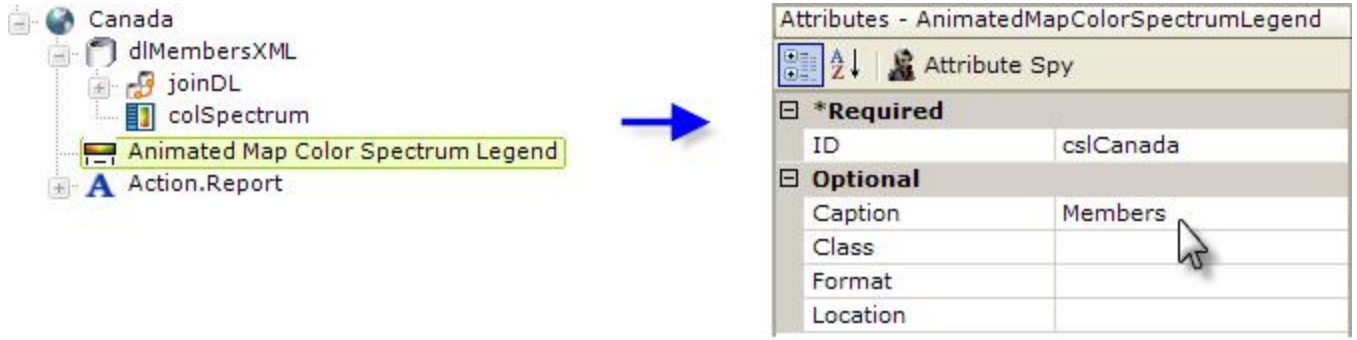
And the results can be seen in the example shown above. If no text is entered in the **Display Value** attributes, the numeric minimum and maximum values will be displayed in the legend instead.

Using a Color Spectrum Legend

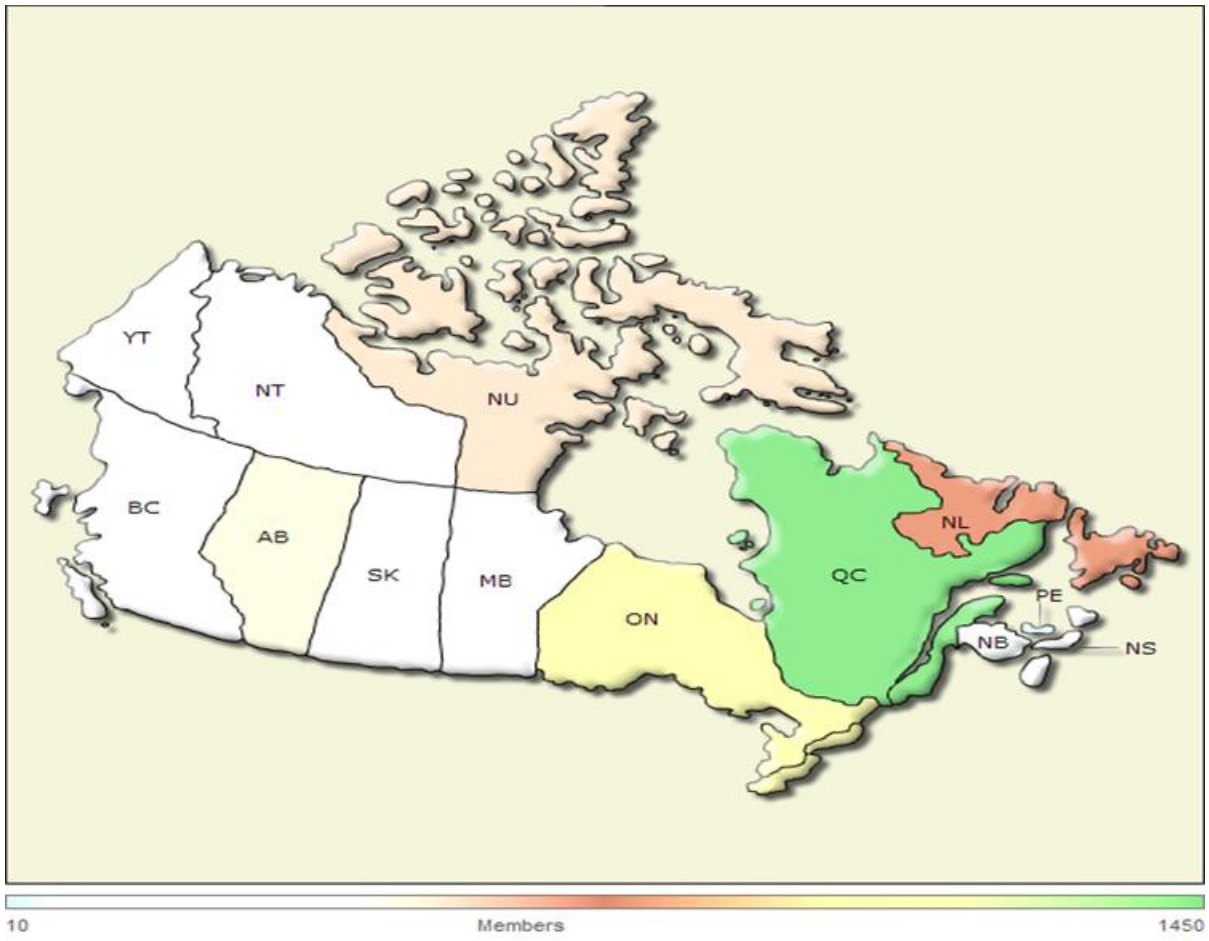
Instead of using Map Color Range elements, you can add a "color spectrum" legend to your animated map; this legend uses colors to illustrate the distribution of the data values used to generate the map and assigns those colors to the appropriate map regions.



The first step, as shown above, is to add a **Color Spectrum Column** element beneath the datalayer. Configure its attributes to identify the data column used as the Animated Map element's Region Value Column, and the desired colors for the low, medium, and high values.



The next step is to add an **Animated Map Color Spectrum Legend** element, as shown above, beneath the Animated Map and to set its caption. The legend appears by default below the map, but its **Location** attribute can be set to make it appear to the right of the map.



And the resulting map and legend are shown above.

Available Maps

Approximately 320 maps are distributed with Logi Info (as of v12.2) and they can all be seen in the

```
<yourApplication>\rdTemplate\rdFusionMap
```

folder. The following table provides a partial listing of the maps distributed with Logi Info:

Map Type	Description
World	The World, with six major regions: North and Central America, South America, Europe, Africa, Asia, and Australia.
World8	The World, with eight major regions: North America, Central America, South America, Europe, Africa, Middle East, Asia, and Oceana.
World8withAntartica	The World, with nine major regions: North America, Central America, South America, Europe, Africa, Middle East, Asia, Oceana, and Antartica.
WorldwithAntartica	The World, with seven major regions: North and Central America, South America, Europe, Africa, Asia, Australia, and Antartica.
WorldwithCountries	The World, with 204 countries identified.
NorthAmerica	Includes Greenland, Canada, United States, Mexico, Central America, and Caribbean Islands.
NorthAmericaWOCentral	Includes Greenland, Canada, United States, and Mexico.

Map Type	Description
SouthAmerica	South America, including 16 countries.
CentralAmerica	Central America, including Belize, Costa Rica, El Salvador, Guatemala, Honduras, Nicaragua, and Panama.
CentralAmericawithCaribbean	Central America, with Caribbean Islands.
Europe	Europe, with 47 countries.
EuropewithCountries	Europe, with 49 countries (adds England, Wales, Scotland, and Northern Ireland).
EuropeRegion	Europe, with five major regions: Western, Central, Eastern, Northen, Southern.
EastEuropeanRegion	Includes Belarus, Bulgaria, Moldova, Romania, Russia, and Ukraine.
CentralEuropeanRegion	Includes Austria, Czech Republic, Germany, Hungary, Liechtenstein, Poland, Slovakia, Slovenia, and Switzerland.
WestEuropeanRegion	Includes Belgium, France, Luxembourg, Monaco, and Netherlands.
NorthEuropeanRegion	Includes Denmark, Estonia, Finland, Iceland, Ireland, Latvia,Lithuania, Norway, Sweden, and the U.K.
SouthEuropeanRegion	Includes Albania, Andorra, Bosnia & Herzegovina, Croatia, Cyprus, Greece, Italy, Macedonia, Malta, Montenegro, Portugal, San Marino, Serbia, Spain, Turkey, and Vatican City.

Map Type	Description
Asia	Asia, including 49 countries.
Asia3	Asia, including 29 countries (without the Middle-East countries in the previous map).
Africa	Africa, including 80 countries.
MiddleEast	Middle East, including 20 countries.
Oceania	Australia, New Zealand, Papua New Guinea, and 12 nearby Pacific islands.
USA	United States, including all individual states.
USARegion	United States, with five regions: Northwest, Southwest, Central, Northeast, Southeast.
USANorthWestRegion	Northwest states, including Alaska, Idaho, Montana, Oregon, Washington, and Wyoming,
USASouthWestRegion	Southwest states, including Arizona, California, Colorado, Hawaii, Nevada, New Mexico, and Utah.
USACentralRegion	Central states, including Illinois, Iowa, Kansas, Michigan, Minnesota, Missouri, Nebraska, North Dakota, Oklahoma, South Dakota, Texas, and Wisconsin.
USANorthEastRegion	Northeast states, including Connecticut, Delaware, District of Columbia, Indiana, Kentucky, Maine, Maryland, Massachusetts, Michigan, New Hampshire, New Jersey, New York, Ohio, Pennsylvania, Rhode Island, Vermont, Virginia, and West Virginia.

Map Type	Description
USASouthEastRegion	Southeast states, including Alabama, Arkansas, Florida, Georgia, Louisiana, Mississippi, North Carolina, South Carolina, and Tennessee.
(Individual States)	Map Type value is the state name, without any spaces; maps show counties.
Canada	Canada, including thirteen provinces: Alberta, British Columbia, Manitoba, New Brunswick, Newfoundland & Labrador, Northwest Territories, Nova Scotia, Nunavut, Ontario, Prince Edward Island, Quebec, Saskatchewan, and Yukon Territory.
(Provinces)	Map Type value is the province name, without any spaces; maps show counties.
UK	United Kingdom, including England, North Ireland, Scotland, and Wales.
UK7	United Kingdom, including Channel Islands, England, Isle of Man, North Ireland, Scotland, and Wales.
(UK Countries)	Map Type value is one of four country names included in UK map, maps show counties.
EnglandRegion	England, with nine regions: East, East Midlands, London, North East, North West, South East, South West, West Midlands, and Yorkshire & the Humber.
ScotlandRegion	Scotland, with ten regions: Borders, Central, Dumfries & Galloway, Fife, Grampian, Highland, Lothian, Strathclyde, Tayside, and Western Isles.
(Countries)	These maps provide counties or other regional divisions. The Map Type values are: Afgh-

Map Type	Description
	anistan, Albania, Andorra, Antigua, Argentina, Armenia, AsiaGeorgia, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Bolivia, BosniaHerzegovina, Brazil, BritishColumbia, Bulgaria, Canada, Chile, China, China2 (includes HongKong, Taiwan, etc.), Columbia, CostaRica, Croatia, Cuba, Cyprus, CzechRepublic, Denmark, Dominica, DominicanRepublic, Ecuador, Egypt, ElSalvador, Estonia, Finland, France, Germany, Greece, Greenland, Grenada, Guatemala, Haiti, Honduras, Hungary, Iceland, India, Indonesia, Iraq, Ireland, Israel, Italy, Jamaica, Japan, Kenya, Latvia, Liechtenstein, Lithuania, Luxembourg, Macedonia, Malaysia, Malta, Mexico, Moldova, Monaco, Montenegro, Mozambique, Netherland, NewZealand, Nicaragua, NorthIreland, NorthKorea, Norway, Paraguay, Peru, Poland, Portugal, PuertoRico, Romania, Russia, SaintKittsAndNevia, SaintLucia, SaintVincentAndtheGrenadines, SanMarino, SaudiaArabia, Slovakia, Slovenia, SouthAfrica, SouthKorea, Spain, Suriname, Sweden, Switzerland, Taiwan, Thailand, Turkey, UAE, Ukraine, Uruguay, Vatican City, and Venezuela.
(Special Maps)	These maps provide regions or provinces: NorwayRegion, SpainProvinces.

Google Maps

The Logi Info **Google Map** element allows you to include the geo-mapping features of the Google Maps web service in your reports and you can combine it with your data to produce data-driven maps.

The following topics introduce the Google Maps web service and the Google Map element:

- What's a "Web Service"?
- Getting a Google Maps API Key
- Google Map Family of Elements
- Refreshing Google Maps
- About Mapping Data

Logi Info also supports Leaflet Maps. For more information, see "Leaflet Maps" on page 363.

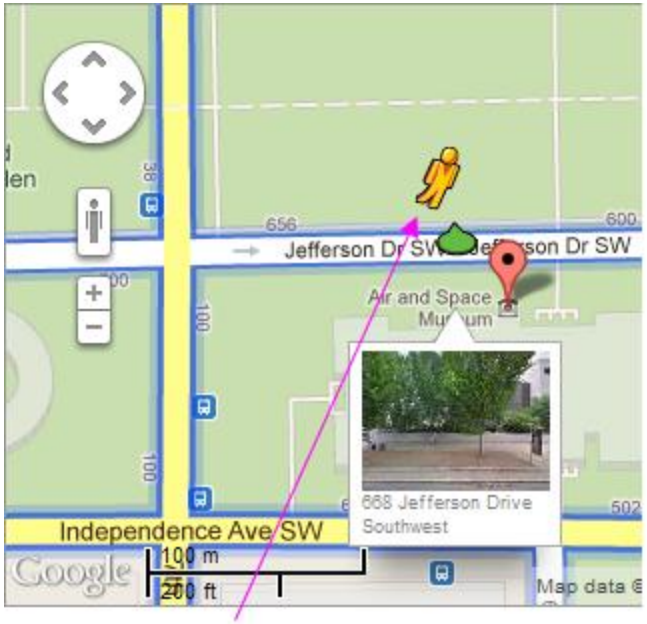
About Google Maps

As mentioned above, a Google Map is an interactive map that can use your data to identify locations and provide other features. The concept is simple: take two different data sources (your data and Google's maps) and combine them into a data-driven map.

The example above shows a Google Map of the area surrounding Washington, D.C. It's a typical map depicting roads, major land features, and significant points-of-interest. The example also identifies the available optional controls, the scale for indicating map distances, and the cursor shaped that appears when dragging the map within its viewing area.

The example above shows the same map with the **Terrain** map type selected.

This basic example shows the same map with the **Satellite** map type selected.

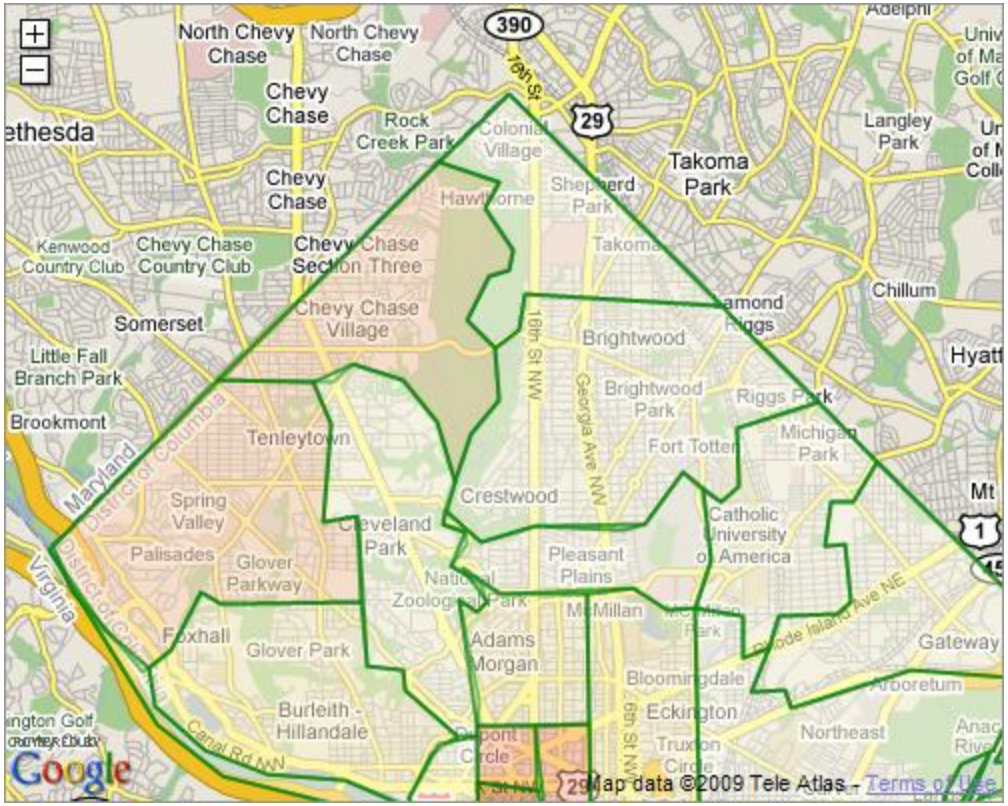


"Peg Man" icon being dragged into a usable location on the map (streets outlined in blue)

Resulting "Street View"

In **Street View** mode, Google Maps provide a street-level photographic view of a map area. Street View is engaged by dragging the "Peg Man" icon, as shown above, onto an enabled street (shown in blue when the icon is moved). Once dropped, the view switches to street-level and controls are presented allowing you to navigate and turn within the "virtual world".

Now the basic map example above includes a "mashup" of data. When identifying data is fed to the web service, the resulting output can pinpoint locations on the map. The example above shows how a geographic **Map Marker** is placed on the map to identify a specific location. Map Markers can use the default icon (shown) or a custom image, gauge, or even a chart. Optionally, the map can be configured to display a **Map MarkerInfo** window, containing additional location-specific data, when the marker is clicked.



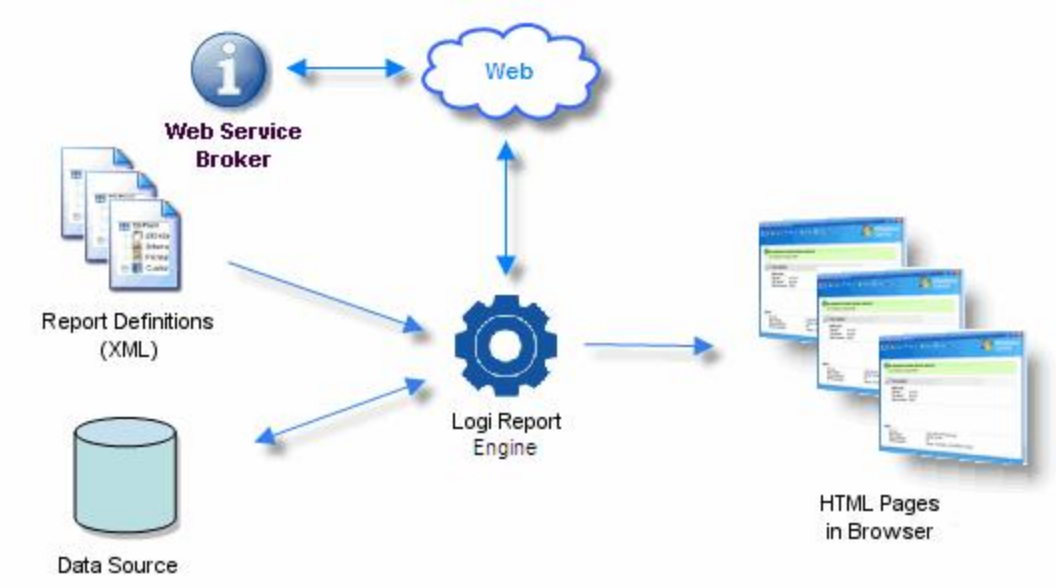
The example shown above includes data-driven, colored regions (in this case, representing postal codes), known as **Map Polygons**, overlaid on a map of Washington, DC. Logi Info can work with GIS boundary data to produce region overlays for states, counties, cities, school districts, and other areas. Like the Map Marker, regions can be clicked to display a pop-up information window with detail data. They're discussed in more detail in "Map Polygons" on page 402.

Similarly, **Map Polylines** can be plotted from data to show a route, in this case from the Orsay Museum in Paris to the Louvre Museum. Polyline color, width, and transparency level are all configurable and can be set from data values. For more information, see "Google Map Polylines" on page 391.

Google Maps can be exported to PDF. Prior to this version, they could not be exported to PDF. Google Maps *cannot be exported* to other formats, such as Word and Excel.

Google Maps - What's a "Web Service"?

The formal definition of a web service is "a software system designed to support interoperable, *machine-to-machine* interaction over a network". In plain terms, developers can think of a web service as a standard programming "function" that happens to be hosted on an external machine, one that your Logi app reaches over the Internet. You send it parameters and the web service sends you results**,** with the benefit that the whole process is language neutral.



The diagram above illustrates how this fits into the Logi application architecture. Usually, the "parameters" that you send to the web service come from a database that your application queries. The results are used in the HTML pages that are output to a browser. Logi Studio elements make the process of connecting to, and communicating with, the web service *very easy*.

Google Maps - Getting a Google Maps API Key

In order to use the Google Map elements in your application, you'll need a Google API key. Logi Info uses Google's *Maps JavaScript API* version 3.

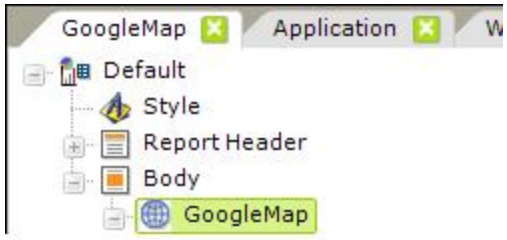
In July 2018, Google changed its licensing scheme, ending the free API access previously offered at some usage levels. This specifically affected *geocoding* in Logi applications.

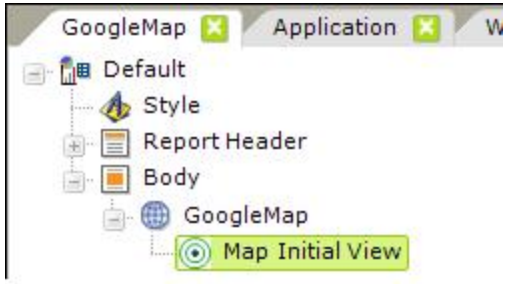
Logi Info now includes the **Connection.Nominatim** element, for connections to a free, public, OpenStreetMap (OSM) Nominatim server for geographic data. Using a simple REST API, geocoding results are returned as XML or JSON data. For more information, see *Datasource Connections*.

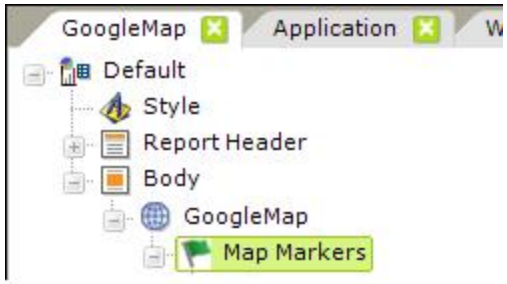
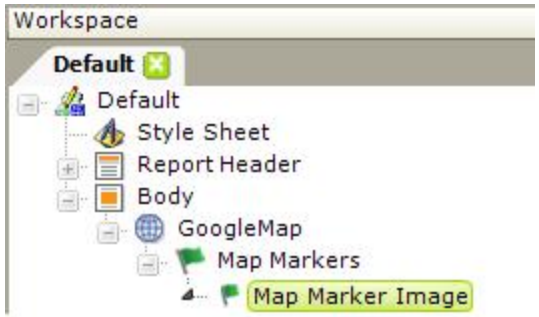
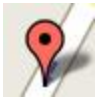
Information about how to get a Google Maps JavaScript API key is available in *Google Connections*. Please read that topic and take the necessary steps before proceeding to implement Google Map in your Logi application.

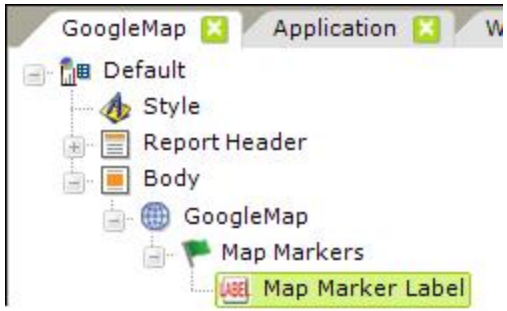
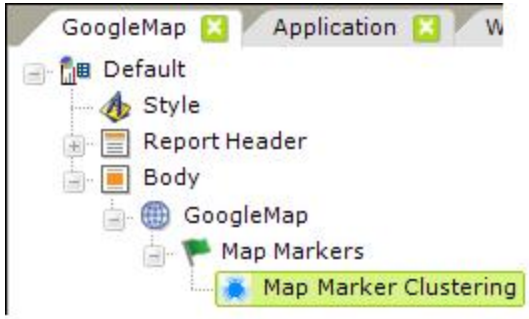
Google Maps - Family of Elements

A number of Logi Studio elements make up the family used to create Google Maps. Each element, its function, and its attributes are discussed below:

	**Connection.Google Maps -** This special type of Connection element is *required* for communicating with the web service. Like all Connection elements, it resides in your _Settings definition. See *Google Connections* for detailed information about getting an API Key and configuring this element.
	**Google Map -** The root element for implementing a Google Map in your report. Attributes include: • **ID** - (Required) A unique ID for this element. • **Height & Width** - (Required) The dimensions, in pixels, of the map. • **Connection ID** - The ID of the Connection.Google Maps element you added to your _Settings definition. If blank, the first (top-most) element of this type in _Settings will be used by default. • **Google Map Show Traffic**- Specifies whether or not vehicular traffic information will appear on the map. Default value: *False* • **Google Map Street View** - Specifies whether or not the "Street View" control appears on the map. This control contains a "peg man" icon that can be dragged onto the map to enable Street View mode, which provides a street-level photographic view of the surroundings of the icon. Default value: *True* • **Google Map Type Control** - Specifies whether or not the Map Type controls

	appear on the map. These controls allow the user to select the type of map to be shown ("Map", "Satellite", etc.). The available map types the user can select will vary depending on the map size and the value of the Google Map Types attribute. The default value is *Auto*, which selects the best control types based on map size and other factors. • **Google Map Types** - Specifies the type of map available for selection by the user when Google Map Type Control value is *True*or *Auto*. One or more map types may be entered, separated by commas, making the corresponding map types available to the user. The first map type entered will be the default type shown when the map is first displayed. Available map type values include *Map*, *Hybrid*, *Satellite*, *Traffic*, and *Terrain*. • **Google Map Zoom Control** - Specifies whether a zoom control appears on the map. The default is *True*. • **Map Scale** - Specifies whether a scale legend appears on the map. Default value: *False* • **Security Right ID** - When Logi Security is enabled, specifies the Security Right IDs of users who will be able to see this element. Multiple Right IDs, separated by commas may be entered.
	**Map Initial View** - Allows you to optionally set the initial Zoom Level and geographic location the map will open to when the page is first displayed. Attributes include: • **Map Zoom Level** - Specifies the resolution of the initial map. Values can range from *0* (the lowest zoom level, in which the entire world appears on one map) to *21+* (down to individual buildings). • **Latitude & Longitude** - Specifies the positioning values, in the data returned in the associated datalayer, for each marker. If left blank, the defaults are "@Data.Lat-

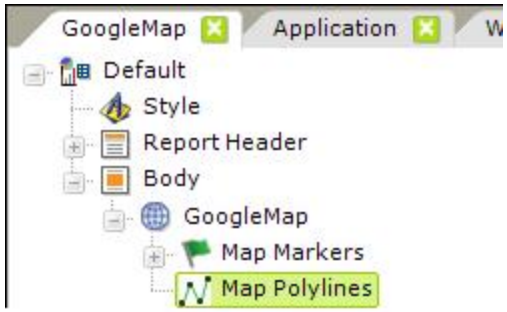
	itude~" and "@Data.Longitude~". Like all tokens, the column names are case-sensitive.
	**Map Markers -** Map Markers appear on the map to pinpoint a location, based on latitude and longitude positioning values. The graphic image for the marker can consist of an image, chart, or gauge; a default image is provided. Attributes include: • **ID** - (Required) A unique ID for this element. • **Latitude & Longitude** - Specifies the positioning values, in the data returned in the associated datalayer, for each marker. If left blank, the defaults are "@Data.Latitude~" and "@Data.Longitude~". Like all tokens, the column names are case-sensitive. • **Security Right ID** - When Logi Security is enabled, specifies the Security Right IDs of users who will be able to see the markers. Multiple Right IDs, separated by commas may be entered. You can add **Action** elements below this element so that when a user clicks a marker, an information panel appears or another report is shown.
	**Map Marker Image -** This *optional* element is a container for one Image, Gauge, or Chart element that provides the image for the Map Marker element. If this element is not included in the definition, the default marker image is used. The default image is:  Data can be used here in interesting ways. For example, an image caption (image file name) and the image size can be data from the datalayer (their values can be @Data

	tokens), so locations can be differentiated visually based on their data. The marker image can also be an actual gauge or chart which would also immediately differentiate locations based on their data.
	**Map Marker Label** - This *optional* element allows you to define a label that will appear under the marker. The label can be styled most easily by defining a CSS class. Its attributes are **Caption** and **Class**.
	**Map Marker Clustering** - When present, this *optional* element will group markers into clusters according to their distance from a cluster's center. When a marker is added, the marker cluster will find a position in all the clusters or, if it fails to find one, will create a new cluster with the marker. The number of markers in a cluster will be displayed on the cluster marker. Clusters will break apart into individual markers when the mapped is zoomed-in sufficiently. This element has no attributes.

Map Polygons - Map Polygons are regions plotted onto a map and are *optional*. They can be semi-transparent and have colors that are based on data values. Polygons are plotted from sets of latitude and longitude points. These typically come from a DataLayer.Gpx File or DataLayer.Kml File element.

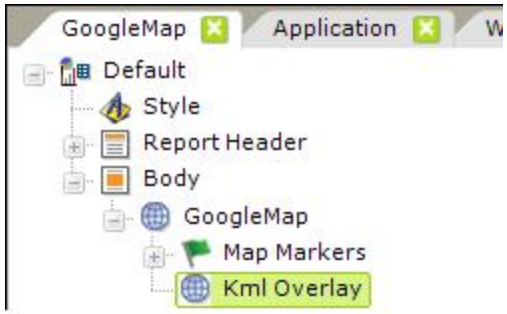
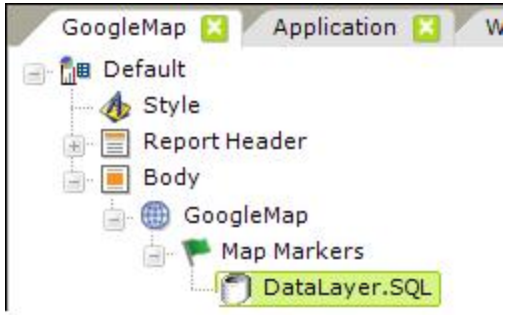
- **ID** - (Required) A unique ID for this element.
- **Border Color and Thickness** - Specifies the polygon border color and thickness in pixels. Default color: *Red*
- **Border Transparency** - Specifies a level of transparency for the border, where *0* = opaque and *15* = completely transparent. Default: *4*
- **Fill Color** - Specifies the color that fills the polygon interior. Tokens can be used here to set the color based on data. Default: *Red*
- **Fill Transparency** - Specifies a level of transparency for the interior, where *0* = opaque and *15* = completely transparent. Default: *4*

You can add **Action** elements below this element so that when a user clicks a polygon, an information panel appears or another report is shown.



Map Polylines - Map Polylines are lines plotted onto a map and are *optional*. They can be semi-transparent and have colors that are based on data values. Polylines are plotted from sets of latitude and longitude points. These typically come from a DataLayer.Gpx File or DataLayer.Kml File element.

- **ID** - (Required) A unique ID for this element.
- **Border Color and Thickness** - Specifies the polygon border color and thickness in pixels. Default color: *Red*
- **Border Transparency** - Specifies a level of transparency for the border, where *0* = opaque and *15* = completely transparent. Default: *4*

	You can add **Action** elements below this element so that when a user clicks a polygon, an information panel appears or another report is shown.
	**KML Overlay** - The KML Overlay element allows display of one or more KML files as overlays on the Google Map and is *optional*. When an overlay is used, the boundary viewport (location and zoom) are set according to the last KML file in the list. • **ID** - (Required) A unique ID for this element. • **KML URL** - The URL of a KML or KMZ file. The URL must be publicly-accessible; local intranet URLs will not work.
	**DataLayer Element -** Datalayer elements are used as usual for any data retrieval operation and can have Group Filters, etc. as child elements to shape the data. Any type of datalayer element can be used. The data for this purpose should include at least some combination of the following (see "Google Maps - About Mapping Data" on page 360): • Street address • City • State • Postal Code

Geocode Columns - This *optional* element can be used if your data does not already include geographic coordinates. It accepts address data and, via the Google web service, attempts to retrieve latitude and longitude values ("geocoding") for each record in the datalayer. The values are placed into two columns, named "Longitude" and "Latitude", whichare added to the datalayer data.

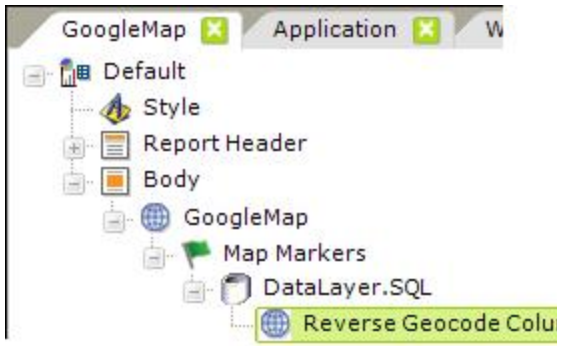
You must either secure a Google Maps API license to use this element or use an alternate geographic data source.

Logi Info now includes the **Connection.Nominatim** element, for connections to a free, public, OpenStreetMap (OSM) Nominatim server for geographic data. For more information, see *Datasource Connections*.

Follow this link for a list of the countries for which Google currently provides geocoding. Other web service brokers also provide geocoding in other countries. Element Attributes include:

- **City Data Column** - Name of a datalayer column that has the city name.
- **Connection ID** - The ID of the Connection.Google Maps element you added to your _Settings definition. If blank, the first (top-most) element of this type in _Settings will be used by default.
- **Country Data Column** - Name of a datalayer column that has the country name.
- **ID** - A unique ID for this element.
- **Include Condition** - An expression that evaluates to a value of *True* or *False*. If the attribute is blank or evaluates to *True*, geocoding will occur; if the value evaluates to *False*, the element is skipped.

- **House Number Data Column** - The name of a datalayer column that contains the street number.
- **Latitude Column ID & Longitude Column ID** - The names of the columns that the web service will add to the datalayer and fill-in with the Latitude and Longitude values. Default column names are "Latitude" and "Longitude".
- **Place Data Column** - The name of a datalayer column that has the entire address information or a place name in a single string. Use this attribute when the address data is not broken up into separate columns or when naming a point-of-interest, such as "Washington Monument" or "Grand Canyon". Use of this attribute disables the other location data columns.
- **Postal Code Data Column** - The name of a datalayer column that has the postal code data (the "Zip Code" in the U.S.)
- **State Province Data Column** - The name of a datalayer column that has the state or province name.
- **Street Data Column** - Name of a datalayer column that contains the street name.

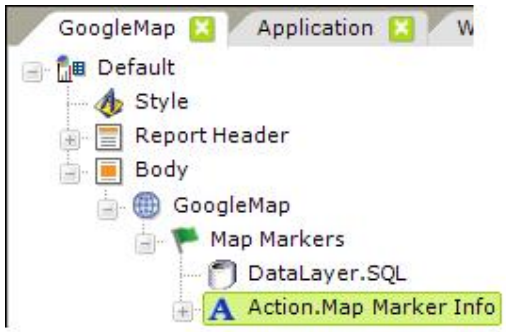
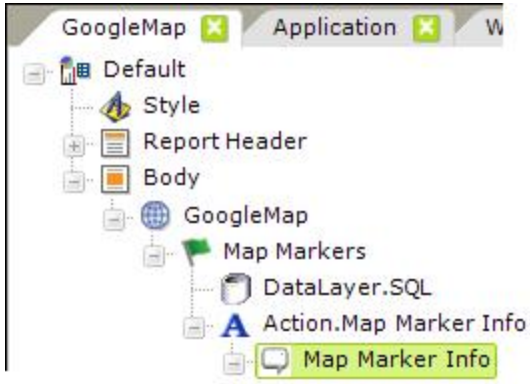


Reverse Geocode Columns - This *optional* element can beused to produce address data from geographic coordinates. Using the Google Maps web service, values returned are put into columns that are added to the datalayer, using these column names:

"StreetNumber", "StreetName", "Locality", "Sublocality", "AreaLevel1", "AreaLevel2", "AreaLevel3", "Country", "Latitude", "Longitude", and "PostalCode".

You must secure a Google Maps API license key to use this element.

Logi Info now includes the **Connection.Nominatim** element, for connections to a free, public, OpenStreetMap (OSM) Nominatim server for geographic data. For more inform-

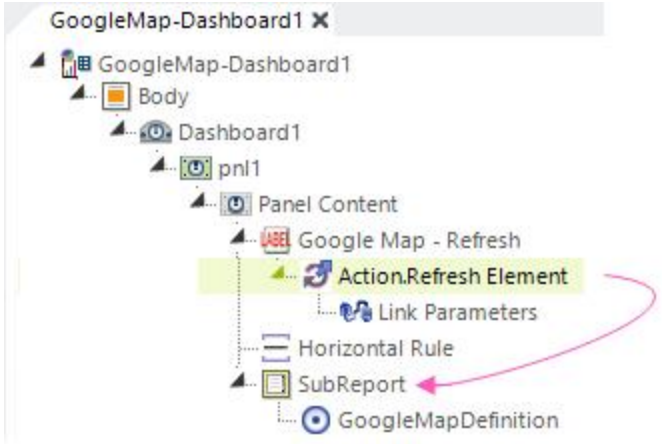
	ation, see *Datasource Connections*. Attributes include: • **ID** - (Required) A unique ID for this element. • **Latitude Data Column** - (Required) Name of the existing datalayer column that contains the latitude data. • **Longitude Data Column** - (Required) Name of the existing datalayer column that contains the longitude data.
	**Action.Map Marker Info -** This *optional* element adds click event processing to the Map Markers element. When a map marker is clicked, processing flow continues with this element's child elements. Attributes are: • **ID** - (Required) A unique ID for this element. • **Security Right ID** - Specifies which security rights can click the map and invoke an action. If blank, then access is unrestricted.
	**Map Marker Info -** This *optional* element represents the pop-up "balloon" that appears over the map when the parent Map Marker is clicked. A wide variety of elements, forming a virtual sub-report, can be placed as children below this element. These can include images, charts, links, data, and more. The data from the row associated with the clicked marker is available to the children of this element. (see the example image below).

Google Maps - Refreshing Google Maps

Developers may want to refresh their Google Maps with new data, either automatically or in response to user input. To do this, you can, of course, use an **Action.Report** element to refresh the entire report page.

Use a SubReport

If you just want to refresh the map, not the entire page, you can place your Google Map in a separate report definition and then include it in your original report as a subreport using the **SubReport** element:



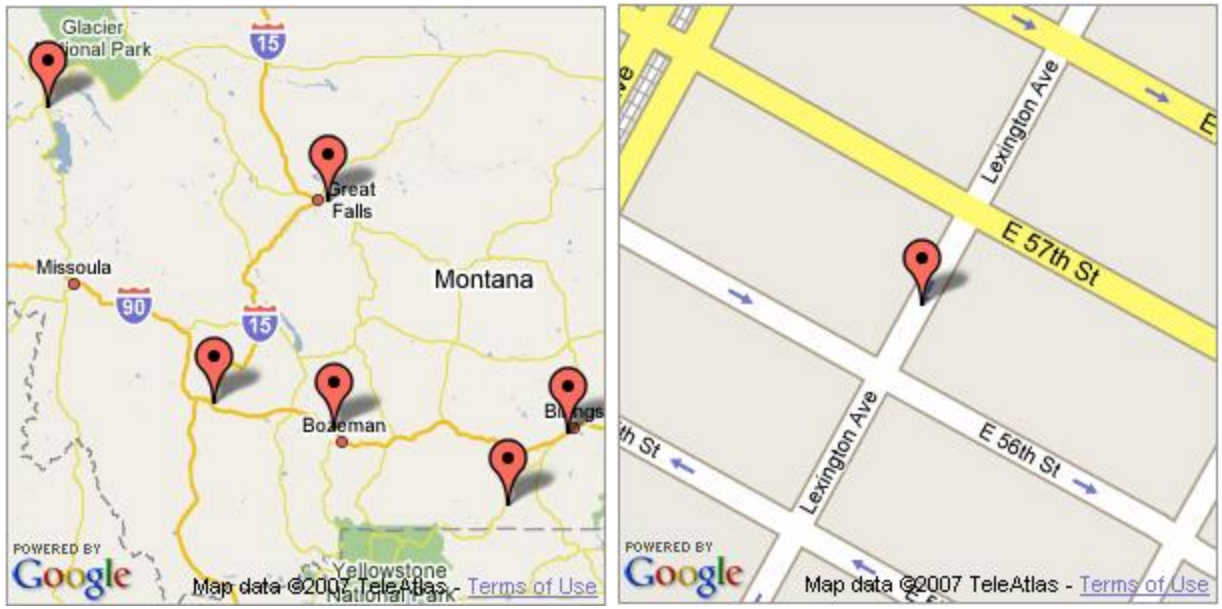
Make the SubReport element the target of an **Action.Refresh Element** or **Refresh Element Timer** element, as shown above. Use **Link Parameters** under the Action element to pass any necessary values to the Google Map. This is the recommended method of updating Google Maps in Dashboard panels.

A Google Map element *cannot* be the direct target of an Action.Refresh Element or Refresh Element Timer element; not even if you wrap the map in a Division. This is why using a SubReport is recommended.

A Google Map element *can* now be the direct target of an Action.Refresh Element or Refresh Element Timer element, however, this refreshes only the Map Markers and *not* the entire map.

Google Maps - About Mapping Data

Data used for Google Maps typically contains, at the least, address information. When this information is sent to the web service, the map generated will be scaled to fit the data into the dimensions you set for the map (in the Google Map element).



For example, if your data only includes state/province information for a single state, the map generated will be scaled to show as much of the state as necessary to include all of the map markers (above, left). This could be the result, for example, of a SQL query statement that includes a `WHERE State = 'MT'` clause.

However, if your data includes street address, city, and state/province information, the map generated will be scaled as a street-level map (above, right). This could be the result, for example, of a SQL query statement that includes a `WHERE City = 'New York'` clause.

As shown above, one of the benefits of the interactive nature of these maps is that you can display additional information when the map marker is clicked. So, your data might include business intelligence data in addition to the address information.

For example, revenue figures or employee counts. This data can be displayed, as in the example, in the Info Window that pops up when a marker is clicked. The Info Window can also contain images, charts, gauges, sub-reports, and links, all driven by data.

If your data already includes latitude and longitude positioning data, so much the better. The use of Geocode Column elements to provide that data involves making additional network "trips" to the web service and that could mean additional transaction fees. Google imposes geocoding limits based on your license type; enterprise users may have other contractual limits. See this Google Maps Geocoding Usage Billing page for more information about prices and limits.

Logi Info now includes the **Connection.Nominatim** element, for connections to a free, public, OpenStreetMap (OSM) Nominatim server for geographic data. For more information, see *Datasource Connections*.

Support for data that includes "MultiGeometry" tags is included.

This concludes this introduction to Google Maps. For additional information, see *Google Map Tutorial*.

With special thanks to contributing writer Michael Kujawski.

Leaflet Maps

The Logi Info **Leaflet Map** element, introduced in Logi Info v12.5, allows you to include geo-mapping features in your application, using the Leaflet JavaScript library and a third-party mapping server. This is an alternative to Google Maps and provides an opportunity to combine geographical data with your own information to produce data-driven maps.

The following topics discuss the use of the Leaflet Map element:

- About Map Servers
- Leaflet Map Family of Elements
- About Mapping Data
- Connecting to Map Servers
- Refreshing Maps

About Leaflet Maps

As mentioned above, a Leaflet map uses the Leaflet JavaScript API and third-party mapping servers to produce an **interactive geographic maps** that can use your data to identify locations and provide other features. The concept is simple: take two different data sources (your data and the map) and combine them into a data-driven map.

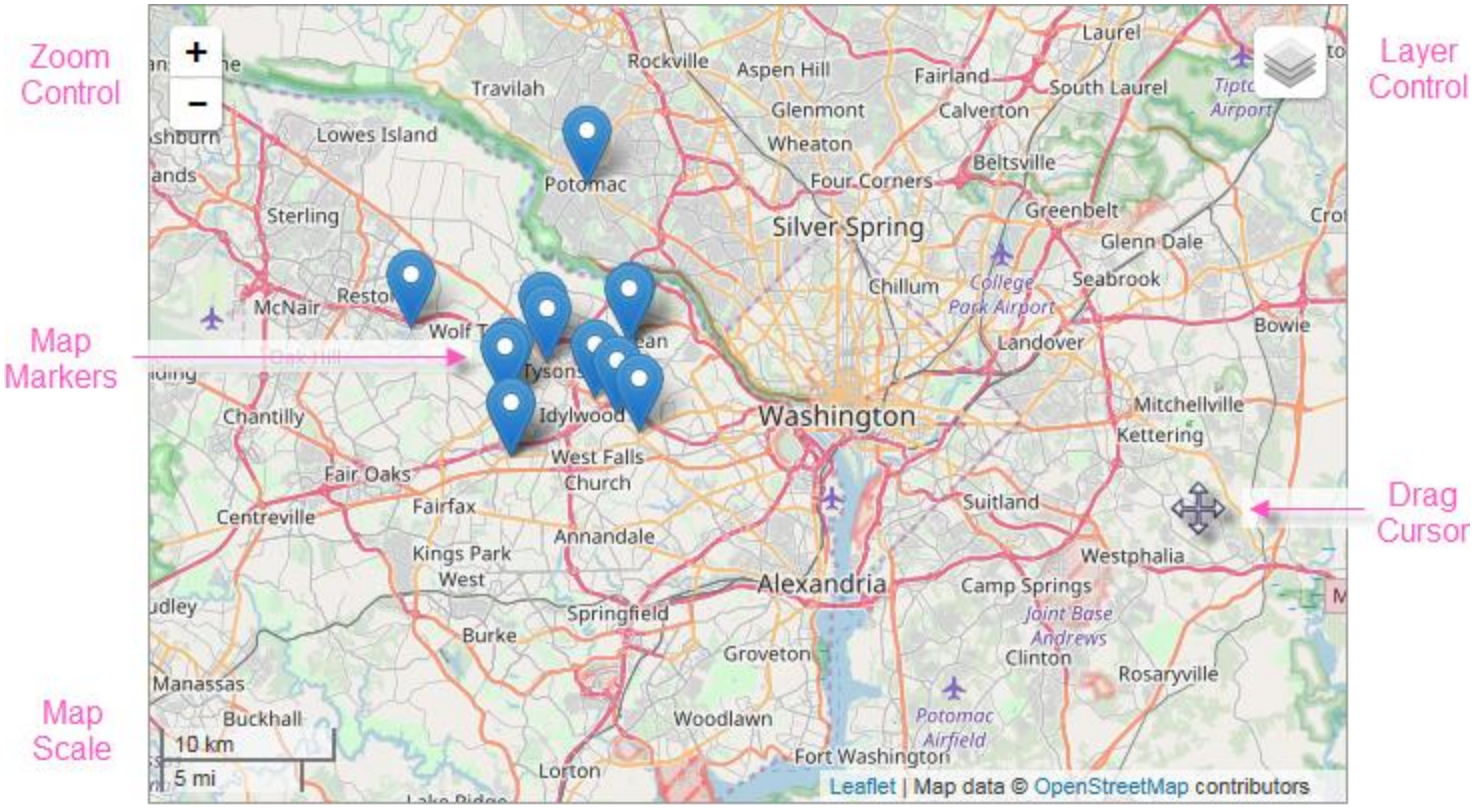
A Leaflet map may be an attractive alternative to Google Maps because it works with any Leaflet-compliant map server, as long as you observe their terms of use. At this writing, there are 70+ map servers that work with Leaflet, including:

- Bing Maps
- Here Maps
- MapBox
- MapQuest
- OpenStreets

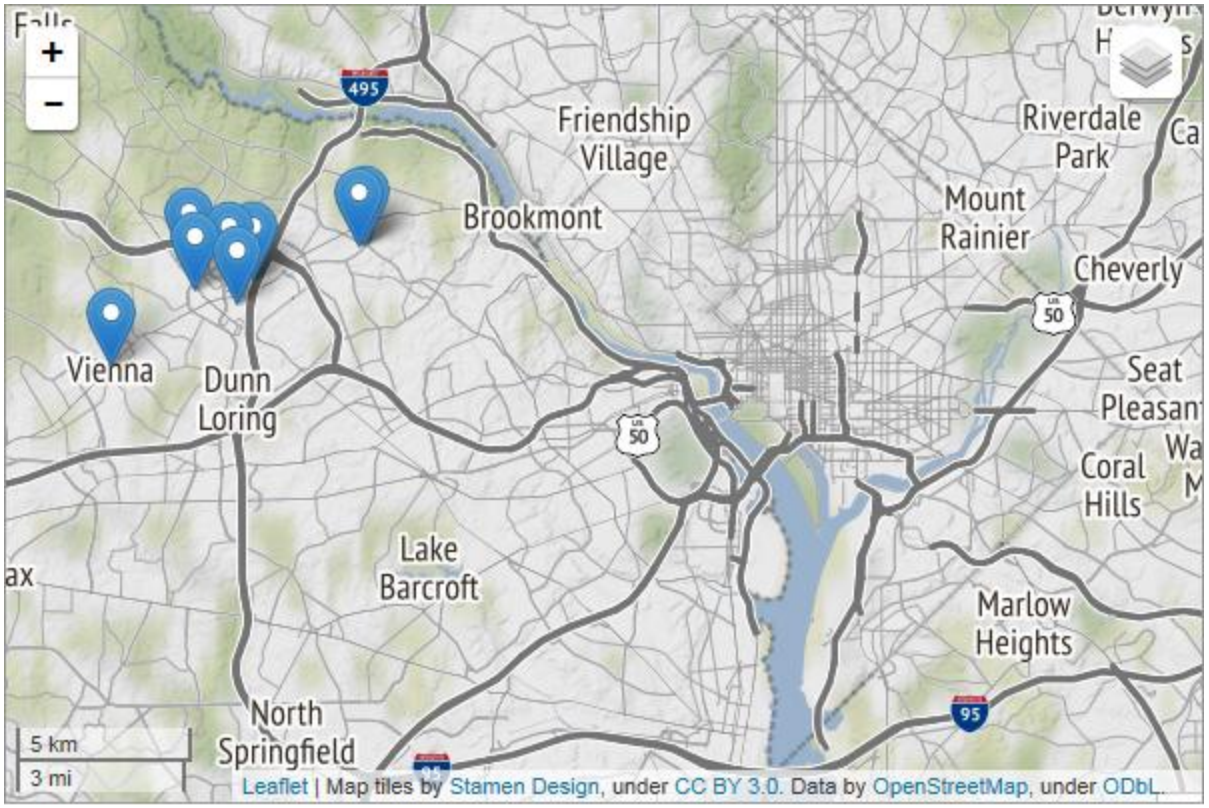
- Stamen
- Thunderforest

and many more.

In your Logi Info application, the Leaflet API is implemented as the **Leaflet Map** element. It makes use of many of the child elements, such as Map Markers and Map Polygons, that are also used for Google Map implementations.



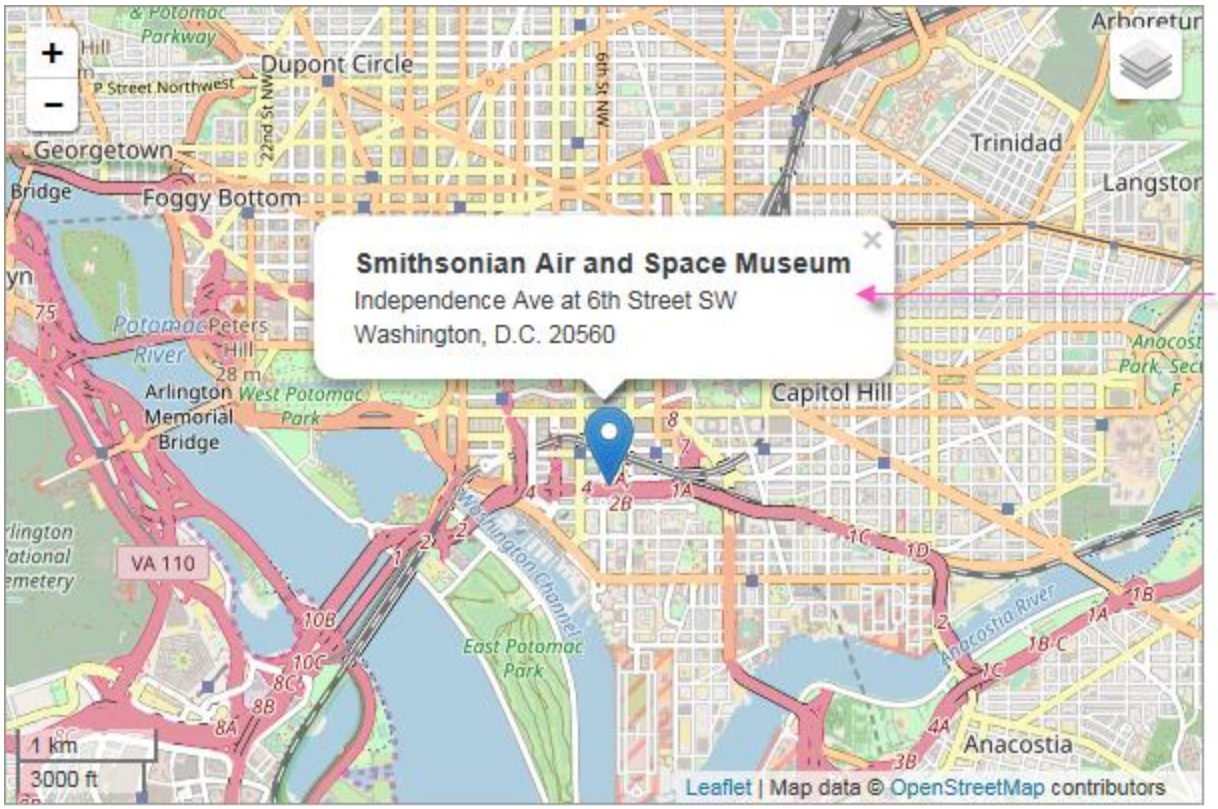
The example above shows an **OpenStreetMap** of the area surrounding Washington, D.C, created using the Leaflet Map element. It's a typical map depicting roads, major land features, and significant points-of-interest. The example also identifies the available optional controls, the scale for indicating map distances, and the cursor shaped that appears when dragging the map within its viewing area.



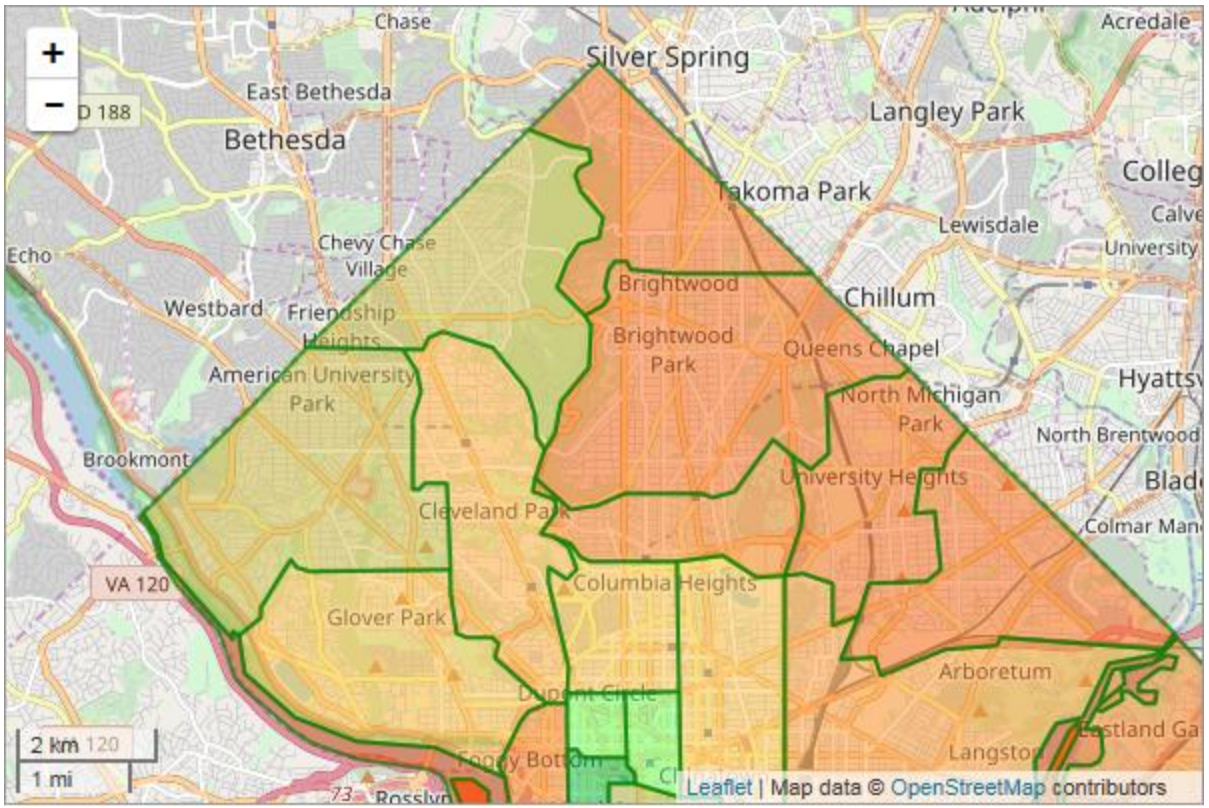
The example above shows the same area, using the **Stamen Terrain** map.

This basic example shows the a **MapQuest Satellite** map of the same area.

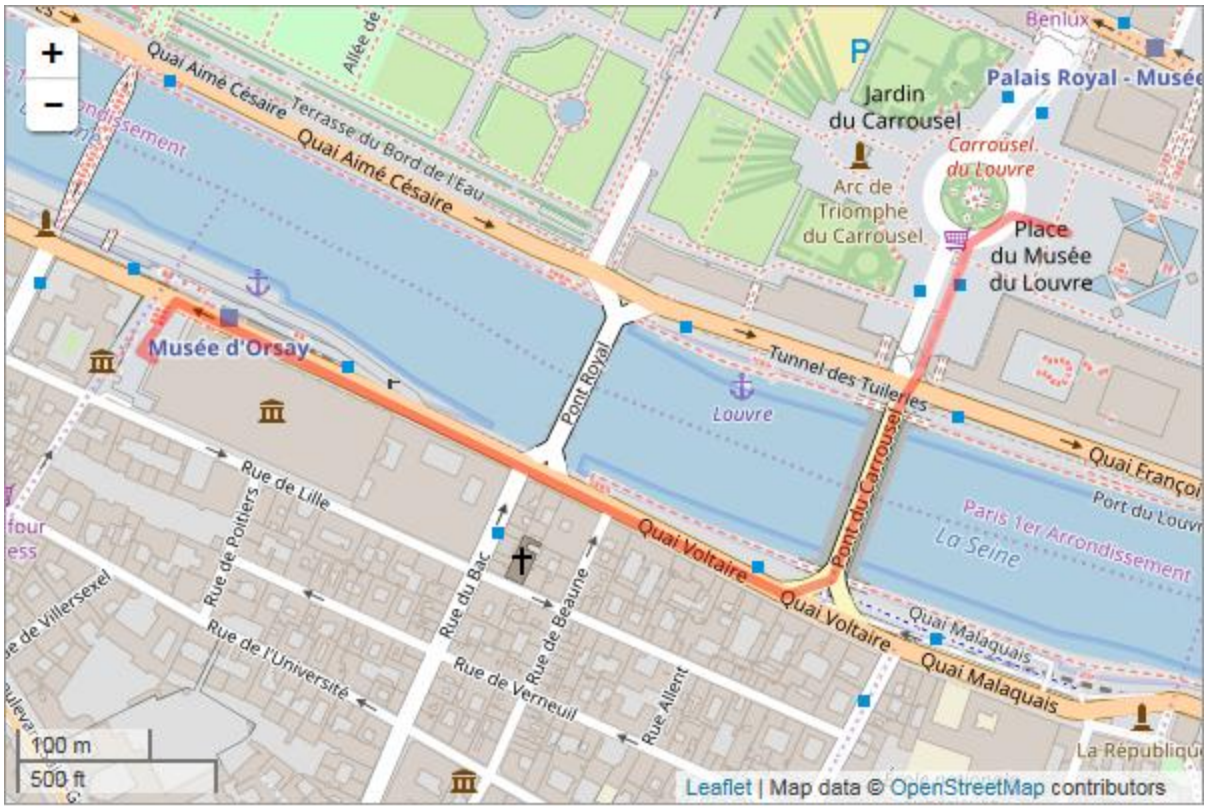
The availability of advanced features, such as Street View, is dependent on the map server.



The basic map example above includes a "mashup" of data. When identifying data is fed to the web server, the resulting output can pinpoint locations on the map. The example above shows how a geographic **Map Marker** is placed on the map to identify a specific location. Map Markers can use the default icon (shown) or a custom image, or a gauge. Optionally, the map can be configured to display a **Map MarkerInfo** window, containing additional location-specific data, when the marker is clicked.



The example shown above includes data-driven, colored regions (in this case, representing postal codes), known as "map polygons", overlaid on a map of Washington, DC. Logi Info can work with GIS boundary data to produce region overlays for states, counties, cities, school districts, and other areas. Like the Map Marker, regions can be clicked to display a pop-up information window with detail data. They're created using the **Map Polygons** element and are discussed in more detail in "Map Polygons" on page 402.



Similarly, "map polyline" overlays can be plotted from data to show a route, in this case from the Orsay Museum in Paris to the Louvre Museum. They're created using the **Map Polylines** element; line color, width, and transparency level are all configurable and can be set from data values. They're discussed in more detail in "Google Map Polylines" on page 391.

Leaflet Maps in your Logi application cannot be exported.

Google Maps Replacement

If you've implemented Google Maps in your Logi application and wish to convert to using Leaflet Maps, you'll find that most of the Google Map child elements (Map Marker Info, Map Marker Image, etc.) are compatible and can be used with Leaflet Maps without configuration changes. This makes conversion fairly easy. Because they now support both map systems, the names of the child elements have been changed and no longer start with the word "Google".

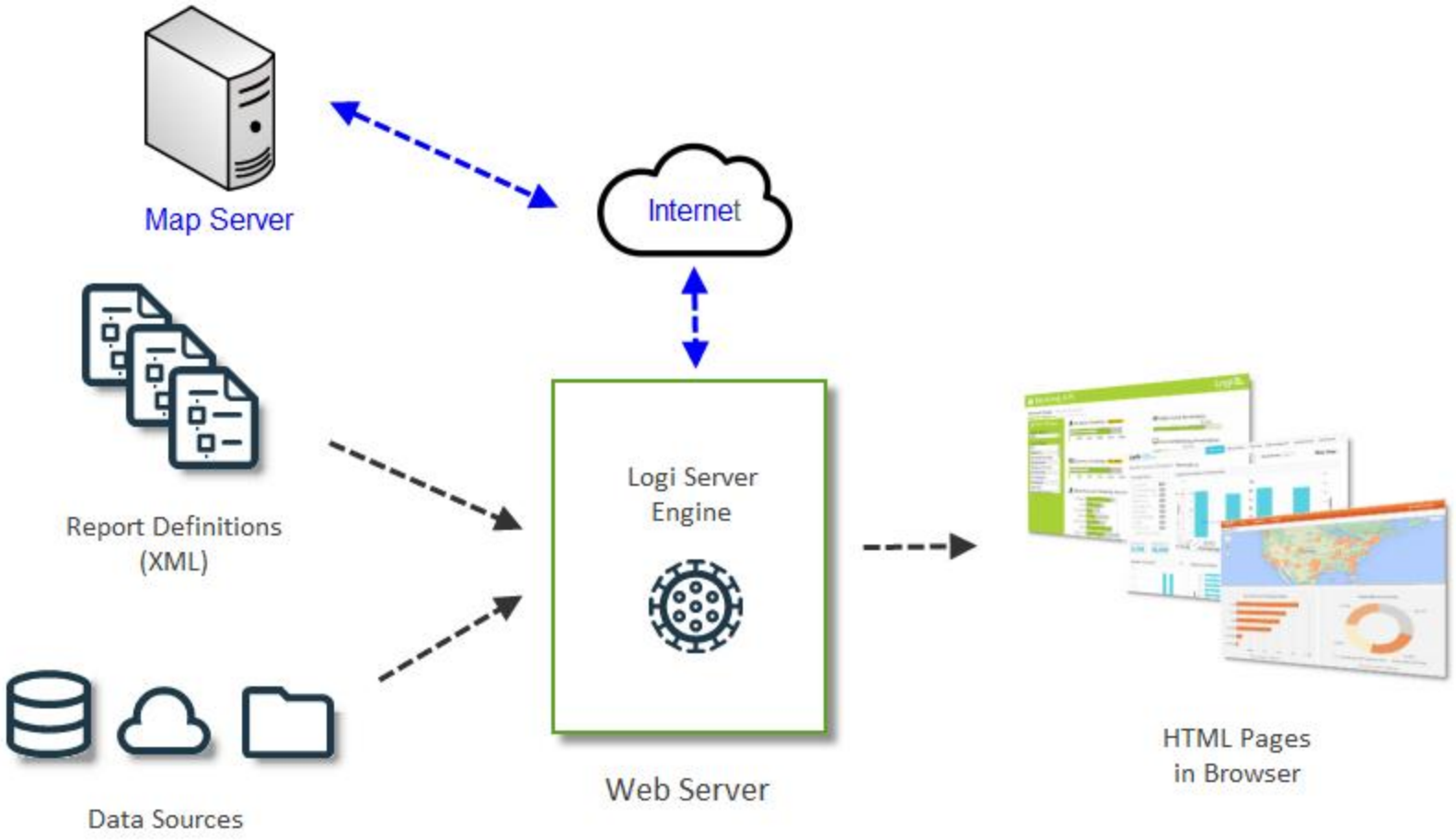
Geocoding

Geographic data for geocoding is *not* available in Logi Info from Leaflet-compatible map servers. If you want to use the **Geocode Columns** or **Reverse Geocode Columns** elements to geocode your location data, you have three choices:

1. Use Google Maps' services, which you have to license and pay for. For more information, see "Google Maps" on page 339.
2. Logi Info now includes the **Connection.Nominatim** element, for connections to a free, public, OpenStreetMap (OSM) Nominatim server for geographic data. Use this connection with the Geocode- and Reverse Geocode Columns elements to access geographic data without engaging with Google. More information is available in *Datasource Connections*.
3. Create your own geocoding *without* using the Geocode- and Reverse Geocode Columns elements. This is done by joining your data with other public data that includes geographic lattitude and longitude information. The DevNet Leaflet Maps sample application, for example, joins Starbucks store data with U.S. Postal Service data (which includes geographic information) on postal (Zip) codes to produce the Leaflet map in its Default definition.

Leaflet Maps - About Map Servers

The Leaflet API is designed to work with public map servers. These provide maps as a set of "tiles" which are assembled into a map for display, allowing for dynamic scaling. Your Logi Info app connects to, and sends parameters to, a map server and the server returns appropriate map tiles.



The diagram above illustrates how this fits into the Logi application architecture. Typically, the parameters that you send to the map server come from a datasource that your application queries. The results are used in the HTML pages that are output to a browser. Logi Studio elements make the process of connecting to, and communicating with, the map server *very easy*.

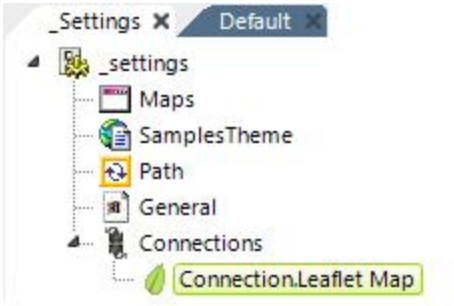
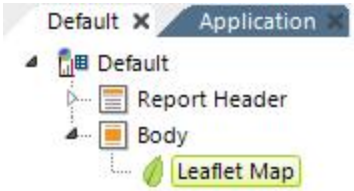
Map Service Licensing

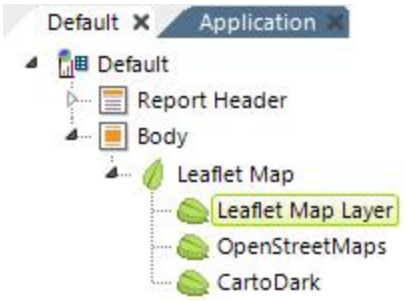
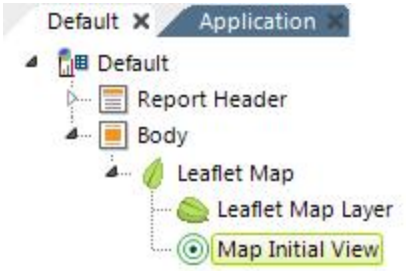
Map server providers make their services available to Internet users, sometimes charging a license fee, but more often for free while requiring attribution. You may be required to get an "API Key" from the provider web site. Providers may impose transaction limits or tiered-usage licensing.

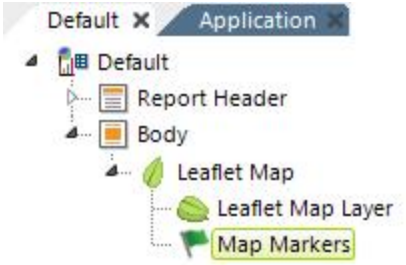
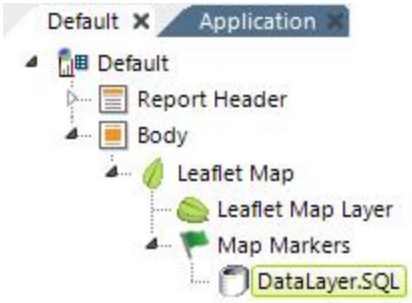
You, as the developer, are responsible for reviewing the current terms of use for the map server you choose to work with, and for taking whatever actions are legally necessary with regard to licensing.

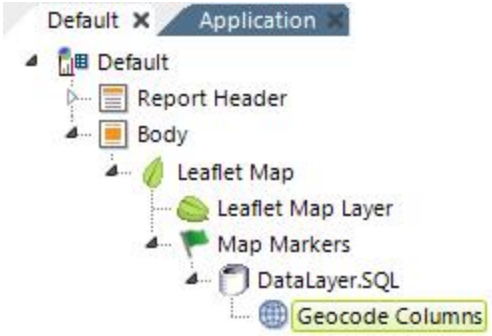
Leaflet Maps - Family of Elements

A number of Logi Studio elements make up the family used to create Leaflet Maps. Each element, its function, and its attributes are discussed below:

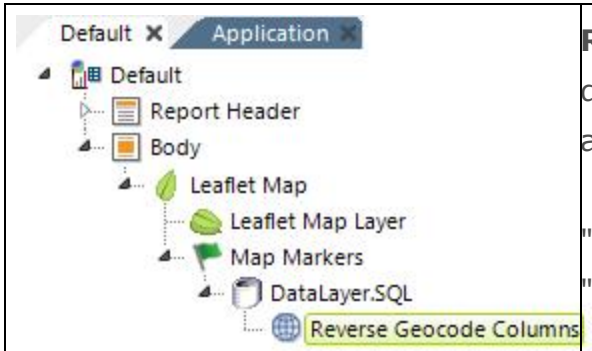
	**Connection.Leaflet Map -** This special type of Connection element is *required* for communicating with the map server. Like all Connection elements, it resides in your _Settings definition. See "Leaflet Maps - Connecting to Map Servers" on page 385 for detailed information about configuring this element.
	**Leaflet Map -** The root element for implementing a Leaflet Map in your report. Attributes include: • **Height & Width** - Specifies optional dimensions, in pixels, for the map. • **Hide Layer Control** - Specifies whether the map Layer selection control is visible on the map or not. The control will appear when there are two or more map layers configured. • **Hide Scale** - Specifies whether the Scale legend is visible on the map or not. • **Hide Zoom Control** - Specifies whether the Zoom control is visible on the map or not. • **ID -** Specifies a unique ID for this element. • **Security Right ID** - When Logi Security is enabled, specifies the Security Right IDs of users who will be able to see this element. Multiple Right IDs, separated by commas may be entered.

	**Leaflet Map Layer** - Required child element that specifies the Connection in the _Settings definition to use to display this layer of map tiles. Multiple Leaflet Map Layer elements may be used and, if so, the Layer Control on the map allows users to switch layers at runtime. This allows you to offer different map types, such as Streets,Terrain, Satellite, etc, from the same map server. Attributes include: • **Connection ID** - (Required) Specifies the ID of a Connection.Leaflet Map element in the _Settings definition. • **ID -** Specifies a unique ID for this element. • **Map Layer Control Caption** - Specifies the identifying text that will appear in the Layer Control for this layer. If left blank, the Connection ID text will appear. • **Show Layer As Overlay** - Specifies whether this layer will be available as an overlay on other layers, allowing layers to the "stacked". If set to *True*, this layer is displayed in the Layer Control with a check box instead of a radio button. If multiple Leaflet Map Layer elements are used, the first one (top-most in the Element Tree) will be the default layer used when the map is first displayed.
	**Map Initial View** - Optional child element that allows you to set the initial Zoom level and geographic location the map will open to when the page is first displayed. Attributes include: • **Map Zoom Level** - Specifies the resolution of the initial map. Values can range from *0* (the lowest zoom level, in which the entire world appears on the map) to a number reflecting the highest possible zoom level (usually less than 20, varies by map server). • **Latitude & Longitude** - Specifies the positioning values, in the data returned in

	the associated datalayer, for each marker. If left blank, the defaults are "@Data.Latitude~" and "@Data.Longitude~". Like all tokens, the column names are case-sensitive.
	**Map Markers -** Optional child element that causes Map Markers to appear on the map to pinpoint a location, based on latitude and longitude positioning values. The graphic image for the marker can consist of an image or gauge; a default image is provided. A child datalayer is used to retrieve the geographical data. Attributes include: • **ID** - (Required) Specifies a unique ID for this element. • **Latitude & Longitude** - Specifies the positioning values, in the data returned in the associated datalayer, for each marker. If left blank, the defaults are "@Data.Latitude~" and "@Data.Longitude~". Like all tokens, the column names are case-sensitive. • **Security Right ID** - When Logi Security is enabled, specifies the Security Right IDs of users who will be able to see this element. Multiple Right IDs, separated by commas may be entered. You can add Action elements below this element so that when a user clicks a marker, an information panel appears or another report is shown.
	**DataLayer -** A datalayer element is used to retrieve the data you wish to visualize on the map. As usual, the datalayer can have calculations, aggregations, grouping, etc. to shape the data. Any type of datalayer element can be used. The schema for data for this purpose should include at least some combination of the following columns (see "Leaflet Maps - About Mapping Data" on page 383):

	• Place Name • House Number • Street address • City • State • Country • Postal Code
	**Geocode Columns -** This *optional* element can be used if your data does not already include geographic coordinates. It accepts address data and, *via the Google Map web service*, attempts to retrieve latitude and longitude values ("geocoding") for each record in the datalayer. The values are placed into two columns, named "Longitude" and "Latitude", which are added to the datalayer data. Geocode Columns can also be used in DataLayers for Google Map Polygons and Google Map Polylines to automatically obtain polygon data when used with a Nominatim connection element. For connections to a free, public, OpenStreetMap (OSM) Nominatim server for geographic data. More information is available in Datasource Connections. You must secure a Google Maps API license to use this element or use an alternate geographic data source. Follow this link for a list of the countries for which Google currently provides geocoding. Other web service brokers also provide geocoding in other countries. Element Attributes include: • **City Data Column** - Name of a datalayer column that has the city name. • **Connection ID** - The ID of the Connection.Google Maps element you added to your

	_Settings definition. If blank, the first (top-most) element of this type in _Settings will be used by default. • **Country Data Column** - Name of a datalayer column that has the country name. • **ID** - A unique ID for this element. • **Include Condition** - An expression that evaluates to a value of *True* or *False*. If the attribute is blank or evaluates to *True*, geocoding will occur; if the value evaluates to *False*, the element is skipped. • **House Number Data Column** - The name of a datalayer column that contains the street number. • **Latitude Column ID & Longitude Column ID** - The names of the columns that the web service will add to the datalayer and fill-in with the Latitude and Longitude values. Default column names are "Latitude" and "Longitude". • **Place Data Column** - The name of a datalayer column that has the entire address information or a place name in a single string. Use this attribute when the address data is not broken up into separate columns or when naming a point-of-interest, such as "Washington Monument" or "Grand Canyon". Use of this attribute disables the other location data columns. • **Postal Code Data Column** - The name of a datalayer column that has the postal code data (the "Zip Code" in the U.S.) • **State Province Data Column** - The name of a datalayer column that has the state or province name. • **Street Data Column** - Name of a datalayer column that contains the street name.



Reverse Geocode Columns - This optional element can be used to produce address data from geographic coordinates. Using *the Google Maps web service*, values returned are put into columns that are added to the datalayer, using these column names:

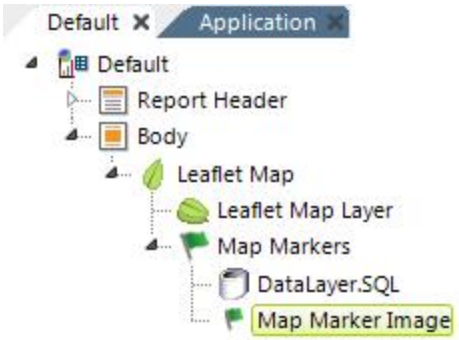
"StreetNumber", "StreetName", "Locality", "Sublocality", "AreaLevel1", "AreaLevel2", "AreaLevel3", "Country", "Latitude", "Longitude", and"PostalCode".

You must secure a Google Maps API license key to use this element or use an alternate geographic data source.

Logi Info now includes the **Connection.Nominatim** element, for connections to a free, public, OpenStreetMap (OSM) Nominatim server for geographic data. More information is available in *Datasource Connections*.

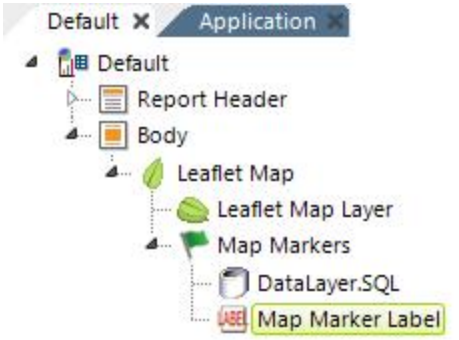
Attributes include:

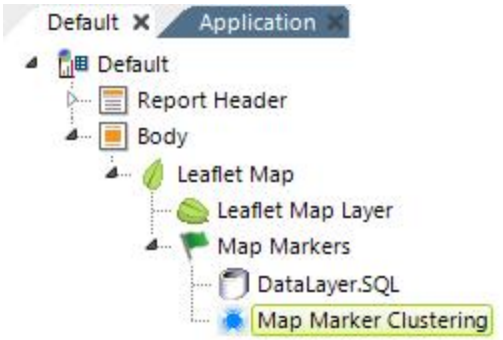
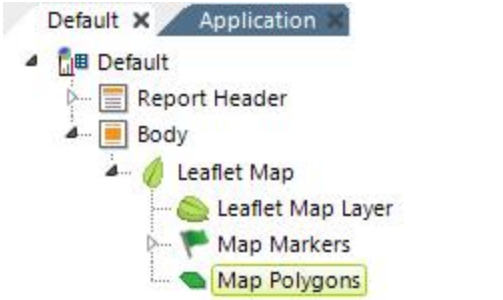
- **ID** - (Required) Specifies a unique ID for this element.
- **Latitude Data Column** - (Required) Specifies the name of the existing datalayer column that contains the latitude values.
- **Longitude Data Column** - (Required) Specifies the name of the existing datalayer column that contains the longitude values.

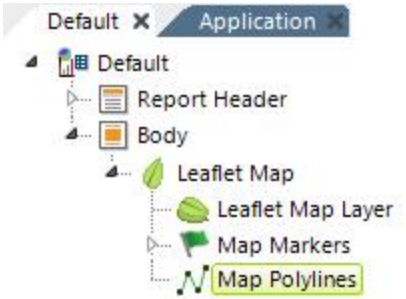
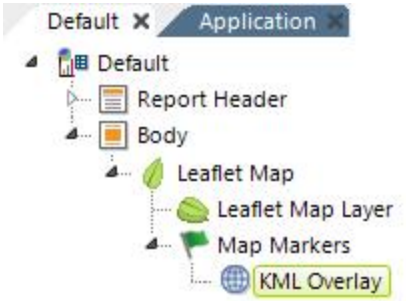
<table>
<tr>
<td> 
</td>
<td>Map Marker Image - This optional Map Markers child element is a container for an Image that provides the image for the Map Marker element. If this element is not included in the definition, the default marker image is used. The default image is:

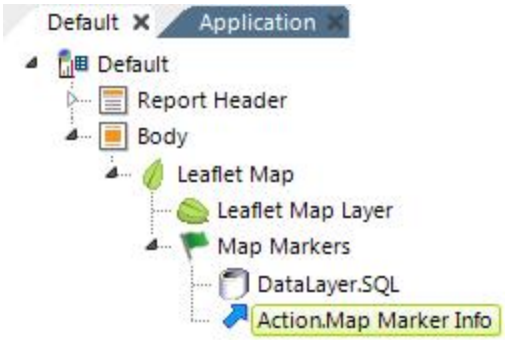
 

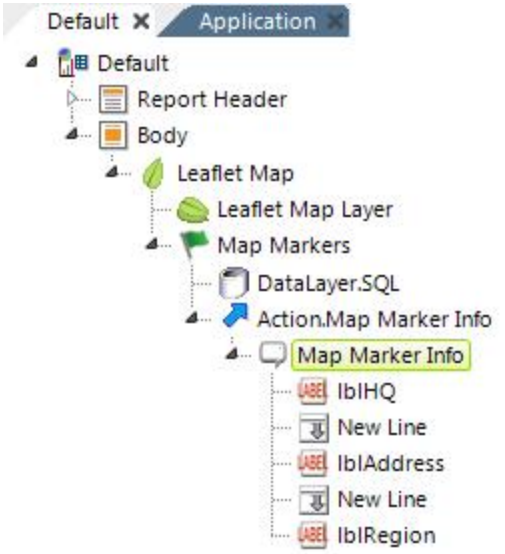
Data can be used here in interesting ways. For example, an image caption (image file name) and the image size can be data from the datalayer (their values can be @Data tokens), so locations can be differentiated visually based on their data.

You cannot add a chart canvas element under the Map Marker Image; to do this, you must first add an Action.Map Marker Info and Map Marker Info element.</td>
</tr>
<tr>
<td> 
</td>
<td>Map Marker Label - This optional Map Markers child element allows you to define a label that will appear under the marker. The label can be styled most easily using CSS.</td>
</tr>
</table>

	**Map Marker Clustering** - When present, this optional Map Markers child element will group markers into clusters according to their distance from a cluster's center. When a marker is added, the marker cluster will find a position in all the clusters or, if it fails to find one, will create a new cluster with the marker. The number of markers in a cluster will be displayed on the cluster marker. Clusters will break apart into individual markers when the mapped is zoomed-in sufficiently. This element has no attributes.
	**Map Polygons** - This optional child element plots overlaid regions onto the map. They can be semi-transparent and have colors that are based on data values. Polygons are plotted from sets of latitude and longitude points. These typically come from a DataLayer.Gpx File or DataLayer.Kml File element. Logi also supports the use of maps with WKT formatted polygons and multipolygons. • **ID** - (Required) Specifies a unique ID for this element. • **Border Color and Thickness** - Specifies the polygon border color and thickness in pixels. Default color: *Red* • **Border Transparency** - Specifies a level of transparency for the border, where *0* = opaque and *15* = completely transparent. Default: *4* • **Fill Color** - Specifies the color that fills the polygon interior. Tokens can be used here to set the color based on data. Default: *Red* • **Fill Transparency** - Specifies a level of transparency for the interior, where *0* = opaque and *15* = completely transparent. Default: *4* You can add **Action** elements below this element so that when a user clicks a polygon, an information panel appears or another report is displayed.

	You can also add a **Polygon Color Spectrum Legend** element beneath this element. The legend shows a bar displaying a spectrum of colors associated with the polygons and representing the gradation between low and high values in the data. The legend may be displayed to the right or below its parent map.
	**Map Polylines** - Map Polylines are lines plotted onto a map and are *optional*. They can be semi-transparent and have colors that are based on data values. Polylines are plotted from sets of latitude and longitude points. These typically come from a DataLayer.Gpx File or DataLayer.Kml File element. Logi also supports the use of maps with WKT formatted polylines and multipolylines. • **ID** - (Required) A unique ID for this element. • **Border Color and Thickness** - Specifies the polygon border color and thickness in pixels. Default color: *Red* • **Border Transparency** - Specifies a level of transparency for the border, where *0* = opaque and *15* = completely transparent. Default: *4* You can add **Action** elements below this element so that when a user clicks a polygon, an information panel appears or another report is shown.
	**KML Overlay** - This optional element allows display of one or more KML files as overlays on the map. When an overlay is used, the boundary viewport details (location and zoom) are set according to the last KML file in the list. • **ID** - (Required) Specifies a unique ID for this element. • **KML URL** - Specifies the URL of a KML or KMZ file. The URL must be publicly-accessible; local intranet URLs will not work.

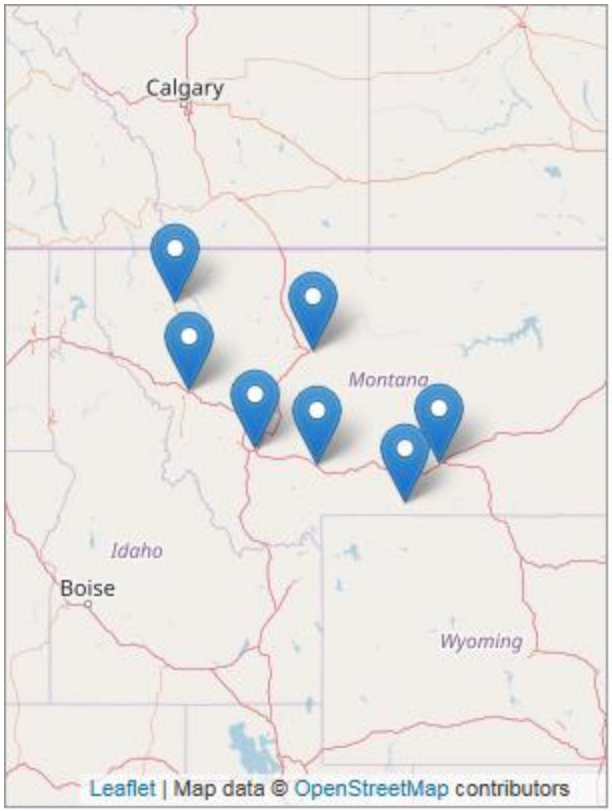
<table>
<tr>
<td> 
</td>
<td>Action.Map Marker Info - This optional element adds Click event processing to the Map Markers element. When a map marker is clicked, processing flow continues with this element's child elements. Attributes are:

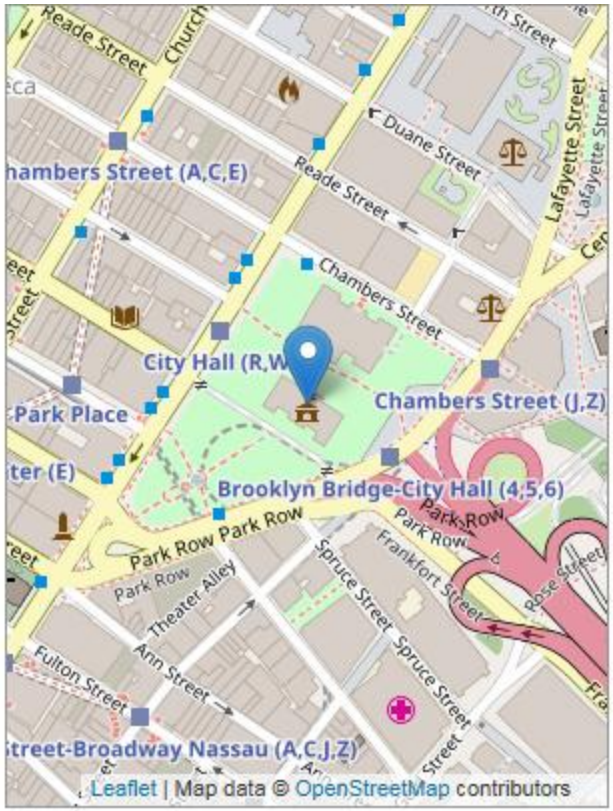
• ID - (Required) Specifies a unique ID for this element.
• Security Right ID - When Logi Security is enabled, specifies the Security Right IDs of users who will be able to click a map marker and initiate an action. Multiple Right IDs, separated by commas may be entered.</td>
</tr>
<tr>
<td> 
</td>
<td>Map Marker Info - This optional element creates the pop-up "balloon" that appears over the map when a map marker is clicked. A wide variety of elements can be placed as children below this element to display information, including images, links, data, and more. The data from the row associated with the clicked marker is available to the children of this element as @Data tokens.

If you choose to use a SubReport element to display data the pop-up balloon, you must use its Embedded subreport mode.</td>
</tr>
</table>

Leaflet Maps - About Mapping Data

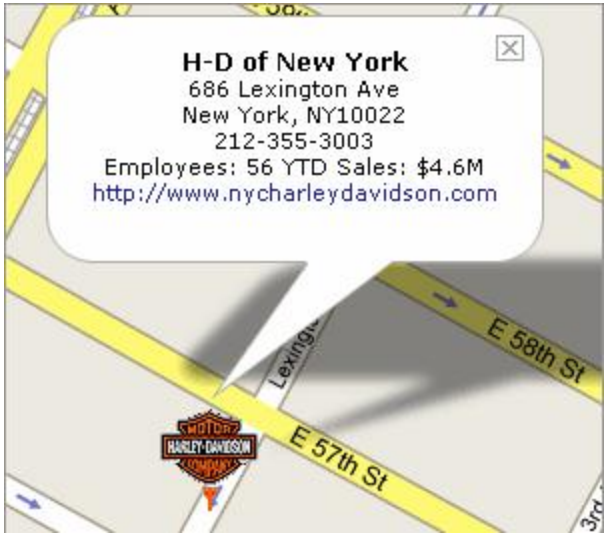
Data used for maps typically contains, at the least, address information. When this information is sent to the map server, the map generated will be scaled to fit the data into the dimensions you set for the map (in the Leaflet Map element).





For example, if your data only includes state/province information for a single state, the map generated will be scaled to show as much of the state as necessary to include all of the map markers (above, left). This could be the result, for example, of a SQL query statement that includes a `WHERE State = 'MT'` clause.

However, if your data includes street address, city, and state/province information, the map generated will be scaled as a street-level map (above, right). This could be the result, for example, of a SQL query statement that includes a `WHERE City = 'New York'` clause.



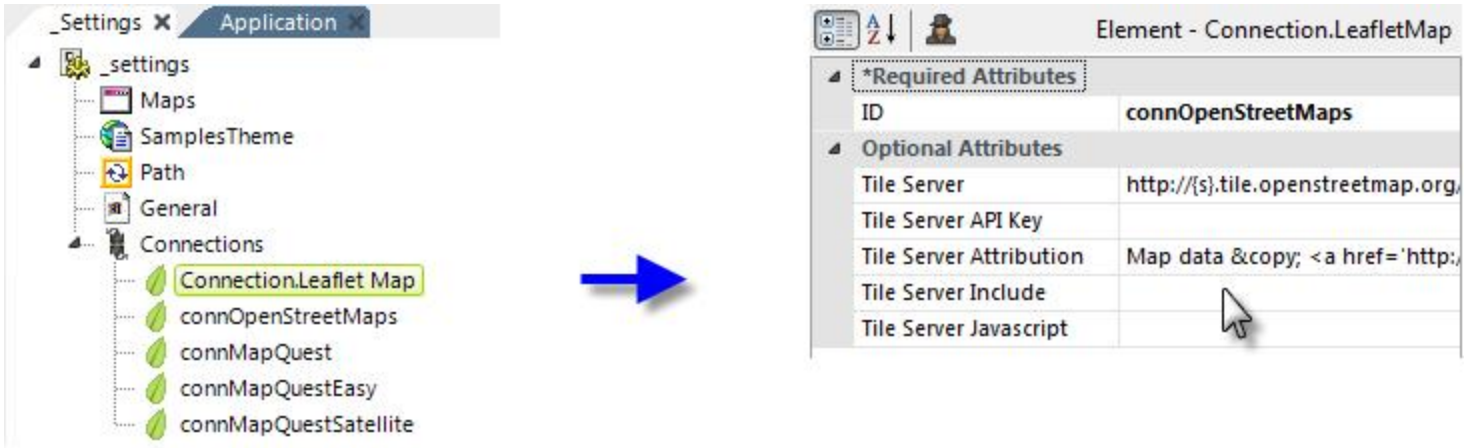
As shown above, one of the benefits of the interactive nature of these maps is that you can display additional information when the map marker is clicked. Your data might include business intelligence data in addition to the address information such as, for example, revenue figures or employee counts. This data can be displayed, as shown above, in the Info Window that pops up when a marker is clicked. The Info Window can also contain images, gauges, sub-reports, and links, all driven by data.

If your data already includes latitude and longitude positioning data, so much the better. The use of Geocode Column elements to provide that data involves additional requests to the web service and, depending on the map server's usage terms, that could mean exceeding the transaction limit or paying additional transaction fees.

Support for data that includes "MultiGeometry" tags is included.

Leaflet Maps - Connecting to Map Servers

Connections to map servers aremade in the _Settings definition, using **Connection.Leaflet Map** elements, one for each map server:



Each map, or "tile",server will have its own connection configuration requirements, and some examples are provided below. Consult the map server's web site for connection details and credentials, if necessary. The Connection.Leaflet Map element's attributes are:

Attribute	Description
ID	(Required) Specifies a unique element ID.
Tile Server	Specifies the URL of a tile server. For example: `http://my.map.tile.server/{z}/{x}/{y}.png`, where: z = map zoom level

Attribute	Description
	x, y = map coordinates The mapping element provides dynamic values for x, y, and z as the user interacts with the map. There are some pre-defined values, such as "MapQuest", which point to a public MapQuest map server. A pre-defined value may be used in place of the URL value described above. See the following section for some examples.
Tile Server API Key	Specifies an API key for map servers that require one, for example MapQuest and Thunderforest.
Tile Server Attribution	Specifies an HTML string with server attribution text, for use when the map server requires it for licensing purposes. When a map layer using this connection is displayed the attribute value is displayed in the bottom right corner of the map.
Tile Server Include	Specifies a URL to a JavaScript file that's required by some map servers. Multiple files may be included by delimiting each URL with the pipe "|" symbol.
Tile Server JavaScript	Specifies the JavaScript constructor code for creating map layers when working with certain Leaflet JavaScript plug-ins. For example, to use the official MapQuest Leaflet plug-in and retrieve their satellite map tiles, the value for this attribute should be `new MQ.satelliteLayer()`

Attribute	Description
	For some of the pre-defined map servers, such as MapQuest, this value is automatically provided. Visit http://leafletjs.com/plugins.html to learn more about Leaflet plug-ins.

Attribute Configuration Examples

These are examples of the Connection.Leaflet Map element's attribute values for selected map servers. The servers pre-defined in Logi Info have the word "Easy" in their names; only the required attributes are included in the table below.

These values may be subject to change by the map provider and are only offered here as a courtesy. If in doubt, consult the provider's website.

Map Server	Attribute Value
CartoDark	Tile Server: `http://a.basemaps.cartocdn.com/dark_all/{z}/{x}/{y}.png`
MapQuest	Tile Server Include: `https://www.mapquestapi.com/sdk/leaflet/v2.2/mq-map.js?key=<yourKey>` Tile Server JavaScript: `new MQ.mapLayer()`
MapQuestEasy	Tile Server: `MapQuest` Tile Server API Key: *<yourKey>*

Map Server	Attribute Value
MapQuestSatellite	Tile Server Include: `https://www.mapquestapi.com/sdk/leaflet/v2.2/mq-map.js?key=<yourKey>` Tile Server JavaScript: `new MQ.satelliteLayer()`
MapQuestSatelliteEasy	Tile Server: `MapQuest-Satellite` Tile Server API Key: *<yourKey>*
OpenStreetMaps	Tile Server: `http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png` Tile Server Attribution: `Map data &copy; <a href='http://openstreetmap.org'>OpenStreetMap</a> contributors`
StamenToner	Tile Server Include: `http://maps.stamen.com/js/tile.stamen.js?v1.3.0` Tile Server JavaScript: `new L.StamenTileLayer('toner')`
StamenTonerEasy	Tile Server: `Stamen-Toner`
StamenTerrain	Tile Server Include: `http://maps.stamen.com/js/tile.stamen.js?v1.3.0` Tile Server JavaScript: `new L.StamenTileLayer('terrain')`
StamenTerrainEasy	Tile Server: `Stamen-Terrain`
StamenWatercolor	Tile Server Include: `http://maps.stamen.com/js/tile.stamen.js?v1.3.0` Tile Server JavaScript: `new L.StamenTileLayer('watercolor')`
StamenWatercolorEasy	Tile Server: `Stamen-Watercolor`

Map Server	Attribute Value

ThunderforestLandscape	Tile Server:
	`https://{s}.tile.thunderforest.com/landscape/{z}/{x}/{y}.png?apikey=<yourKey>`
	Tile Server Attribution:
	`Maps &copy; <a href='http://www.thunderforest.com/'>Thunderforest</a>, Data &copy; <a href='http://www.openstreetmap.org/copyright'>OpenStreetMap contributors</a>`

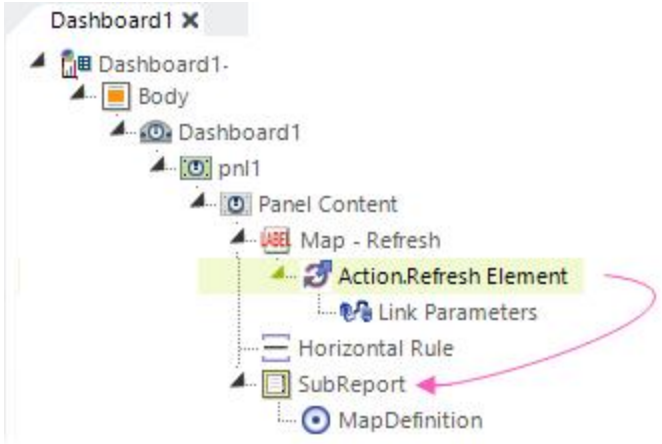
ThunderforestLandscapeEasy	Tile Server: `Thunderforest-Landscape` Tile Server API Key: *<yourKey>*

Leaflet Maps - Refreshing Maps

You may want to refresh your map with new data, either automatically or in response to user input. To do this, you can, of course, use an **Action.Report** element to refresh the entire report page.

If you just want to refresh the *data* that drives the map markers, you can make the Leaflet Map element the direct target of an **Action.Refresh Element** or **Refresh Element Timer** element.

If you just want to refresh the *entire* map, but not the entire page, you can place your map in a separate report definition and then include it in your original report as a subreport using the **SubReport** element:



Make the SubReport element the target of an **Action.Refresh Element** or **Refresh Element Timer** element, as shown above. Use **Link Parameters** under the Action element to pass any necessary values to the Leaflet Map. This is the recommended method of refreshing maps in Dashboard panels.

Special thanks to documentation contributors Michael Kujawski and Jason Scanzoni.

Google Map Polylines

If you've used Google or Leaflet maps to provide geographic mapping solutions in your Logi application, you can take advantage of another advanced feature, **Map Polylines**. In Logi Info, this allows you to overlay solid or semi-transparent, color-coded lines on a map, based on data. They're often used to depict a route from one place to another.

The following topics discuss the use of Map Polylines:

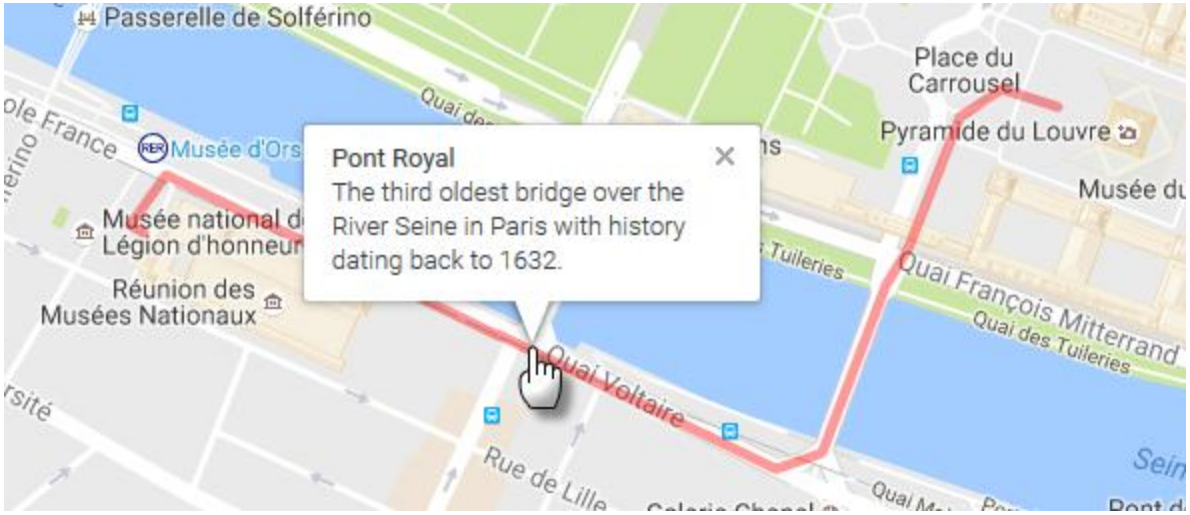
- Implementing Google Map Polylines
- Creating a Map with Polylines

If you haven't already, you should first read "Leaflet Maps" on page 363 or "Google Maps" on page 339, which provide general information.

About Map Polylines

Prior to v12.5, the element being discussed in this topic was called "Google Map Polylines". However, with the introduction of Leaflet maps, the "Google" part of the name was dropped. The element works in the same way for either mapping system. Map Polylines, plotted onto the map, are overlays that are drawn, and can be color-coded, based on data values. Here's an example:

The map above uses polylines to show a route from the Orsay Museum in Paris to the Louvre Museum. Polyline color, width, and transparency level are all configurable and can be set from data values. The line above has been configured to be semi-transparent, so that geographic features can be seen through it.



Polylines can be configured with Map Marker Info pop-ups and can be related to other data. In the example above, they've been configured so that when they're clicked, an information panel is displayed containing more detailed information about the immediate location. Other use-case examples include subway maps, traffic density indicators, and pipeline route mapping, to name but a few.

Geographic Data

Polylines are plotted using sets of **latitude** and **longitudepoints** that describe line segments. Multiple segments are used to draw a continuous line. For use with Logi elements, these typically come from GPX or KML files, which are GIS industry-standard XML data files. Special datalayers are provided in Logi Info to read these files. Because of their public nature, this data is often freely available on the Internet. Data is also widely available in *shapefiles*, a popular geospatial vector format for GIS data. Software tools, discussed in "Map Polygons" on page 402, are available for converting shapefiles into GPX or KML data files. Geographic data can also be retrieved from SQL database tables using DataLayer.SQL; see for more information.

Implementing Google Map Polylines

The following example will demonstrate how to implement Map Polylines. The purpose of the example is to create the example we've seen in "Google Map Polylines" on page 391: routes between the Louvre Museum in Paris and several other primary tourist attractions. As is often the case, the first step is to have a look at the data.

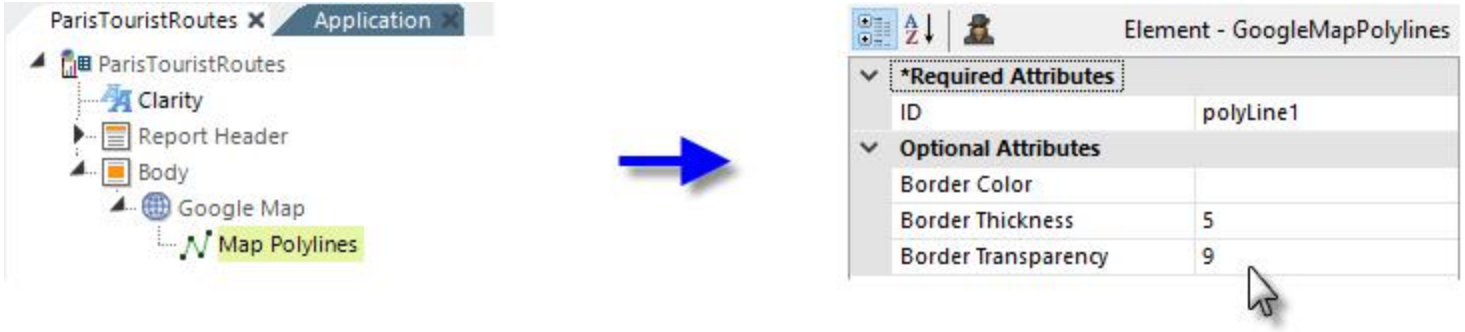
```
<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.0" xmlns="http://www.topografix.com/GPX/1/0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.topografix.com/GPX/1/0 http://www.topografix.com/GPX/1/0/gpx.xsd">
<trk>
<name>LouvreToCafeMetropole</name>
<trkseg>
 <trkpt lat="48.8610022" lon="2.3352277"></trkpt>
<trkpt lat="48.8607410" lon="2.3350883"></trkpt>
 <trkpt lat="48.8605857" lon="2.3356140"></trkpt>
 <trkpt lat="48.8598023" lon="2.3351634"></trkpt>
 <trkpt lat="48.8594352" lon="2.3366225"></trkpt>
<trkpt lat="48.8593152" lon="2.3379850"></trkpt>
<trkpt lat="48.8591670" lon="2.3390150"></trkpt>
<trkpt lat="48.8589482" lon="2.3402274"></trkpt>
<trkpt lat="48.8590470" lon="2.3402917"></trkpt>
</trkseg>
</trk>
<trk>
<name>LouvreToOrsay</name>
<trkseg>
<trkpt lat="48.8611857" lon="2.3351204"></trkpt>
```

```
<trkpt lat="48.8613198" lon="2.3345196"></trkpt>
...
```

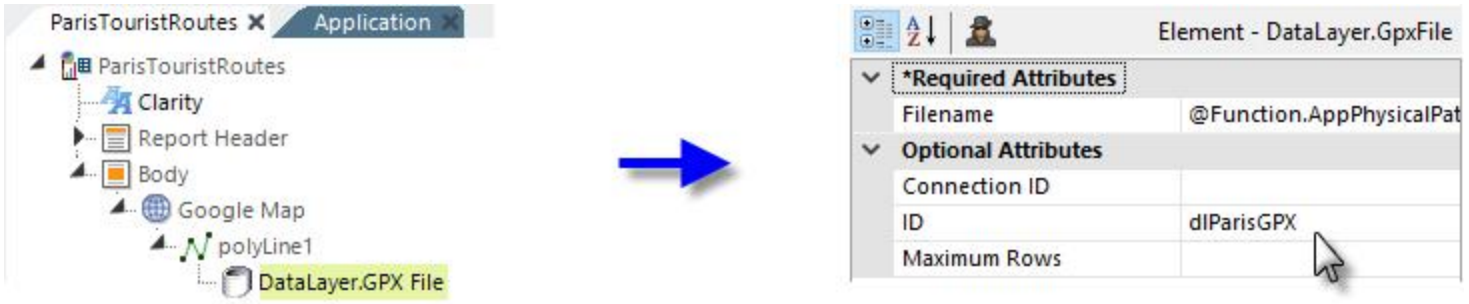
The GIS data defining the line segments for each route has been download and stored as a GPX file, and a small part of it is shown above. You can see that each route has a name and a set of latitude and longitude records, organized in a hierarchy. The format of a KML file is somewhat different but it's still XML data and you can look at it easily enough to determine relevant field names.

Creating a Map with Polylines

The following examples show an implementation using Google Maps, but the child elements described are used identically for a Leaflet Map. Before starting to create an application that uses Google Maps, you must get a **Google Maps API Key**, in order to access the Google web service. *Google Connections* provides information about getting and configuring an API Key. The example begins below with the assumption that you've gotten an API Key and that you've configured a **Connection.Google Maps** element in your _Settings definition with it.

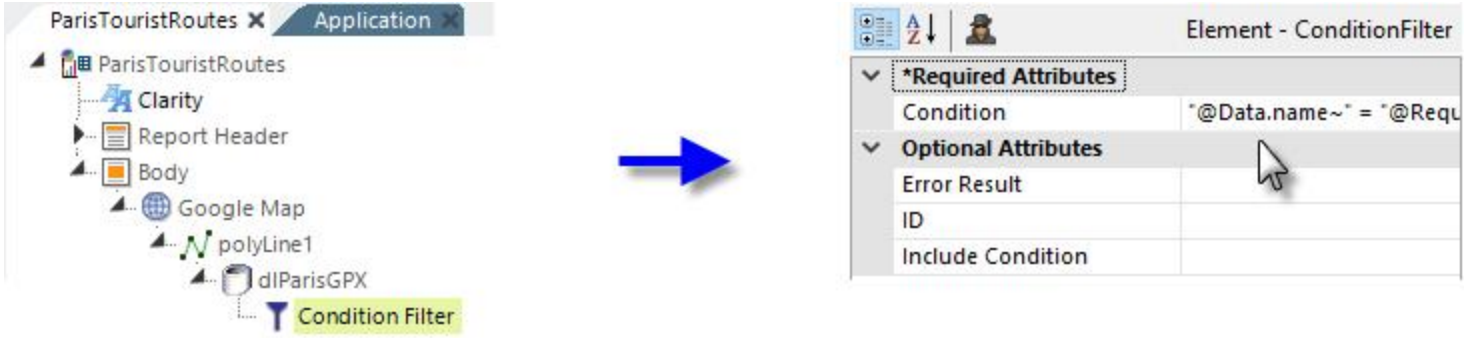


1. In your report definition, add and configure a **Google Map** element to use your Google Map connection.
2. Beneath it, add a **Map Polylines** element, as shown above. This configures the lines that will overlay the map and its **Border** attributes affect their appearance.



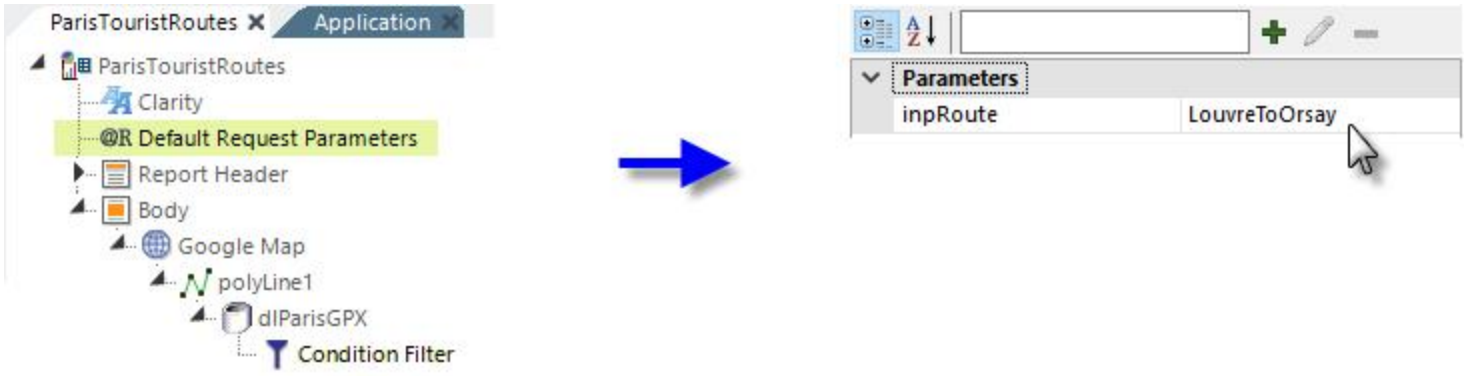
3. Beneath the Polylines element, add a **DataLayer.Gpx File** element, as shown above. It directly reads the GPX file we saw earlier to get the GIS data defining the routes. If you're working with KML files instead, there's also a **DataLayer.Kml File** element for reading them. Tokens such as @Function.AppPhysicalPath~ can make it easier to identify a file stored within your application folder. Or, add a **DataLayer.SQL File** element to the Polygon element if you are working with WKT formatted polylines and/or multipolylines. The full Filename attribute value above is:

```
@Function.AppPhysicalPath~\_SupportFiles\ParisTouristRoutes.gpx
```



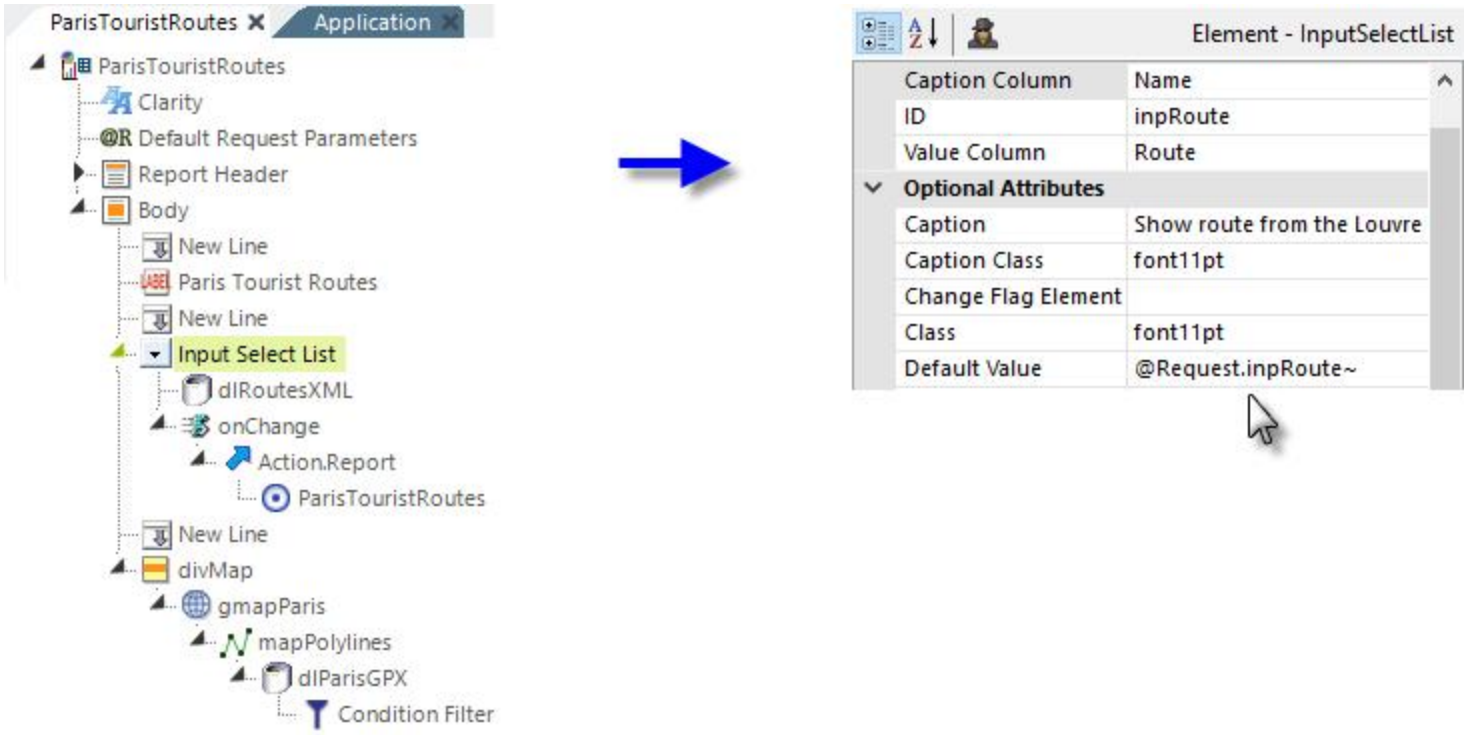
4. We want to dynamically select which individual route to display, so add a **Condition Filter** beneath the datalayer, as shown above. We'll use compare a data column value to a User Input selection we'll add later. The complete Condition attribute value is:

```
"@Data.name~" = "@Request.inpRoute~"
```



5. Let's give the filter something to work with from the initial page load: a Default Request Parameter. Add the element, as shown above, and configure a request variable and value as shown. Remember that the spelling and case of the default parameter *must* match that of the Request token in the Condition Filter element's Condition attribute.

   You should be able to test the application now and see one set of Polylines overlaid on the map.



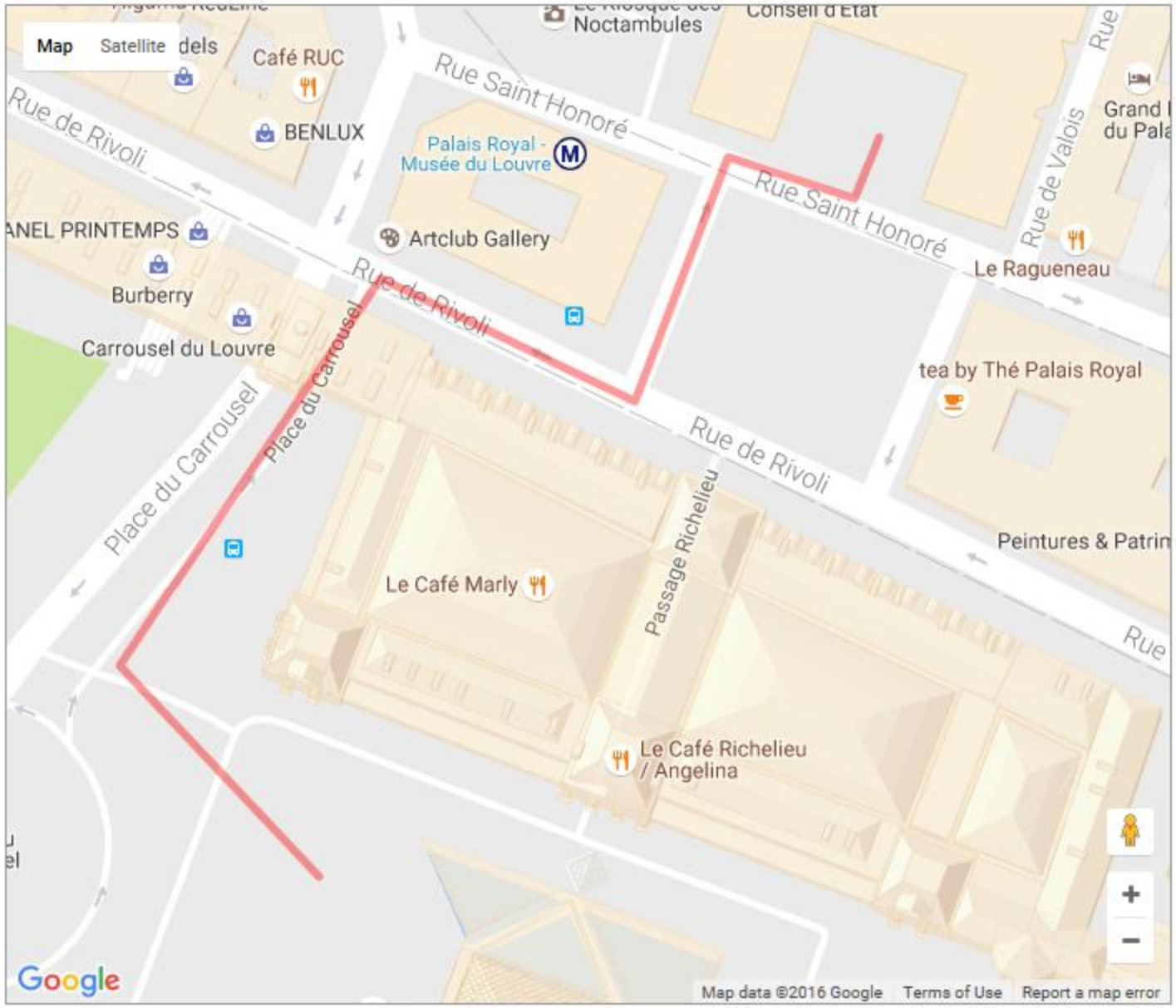
6. Now we'll pick up the pace a bit. As you can see above, we've added **New Line** and **Label** elements to give the page a title.

   And we've added an **Input Select List** element, which lets the user select which route he'd like to see drawn on the map. The datalayer reads an XML file (shown later) that contains correlates user-friendly route names to the names in the data.
   The element ID of the input element matches the spelling and name of the default request parameter we configured in the previous step.

   Finally, in the example code, an **Event Handler** is added that refreshes the page when an input selection is made.

Paris Tourist Routes

Show route from the Louvre Museum to: Hotel Palais Royal



The resulting map looks like the example shown above. The map behaves like any Google Map and can be zoomed and re-centered. Select a different route in the Input Select List to re-draw the map and show the route.

Here's that XML data file used with the Input Select List:

```
<Paris>
 <Routes Name="The Orsay Museum" Route="LouvreToOrsay" />
 <Routes Name="Cafe Metropole" Route="LouvreToCafeMetropole" />
 <Routes Name="Tuileries Gardens" Route="LouvreToTuileries" />
 <Routes Name="Angelina Chocolates" Route="LouvreToAngelina" />
 <Routes Name="Hotel Palais Royal" Route="LouvreToPalaisRoyal" />
 <Routes Name="Cafe de l'Epoque" Route="LouvreToCafedelEpoque" />
</Paris>
```

Map Polygons

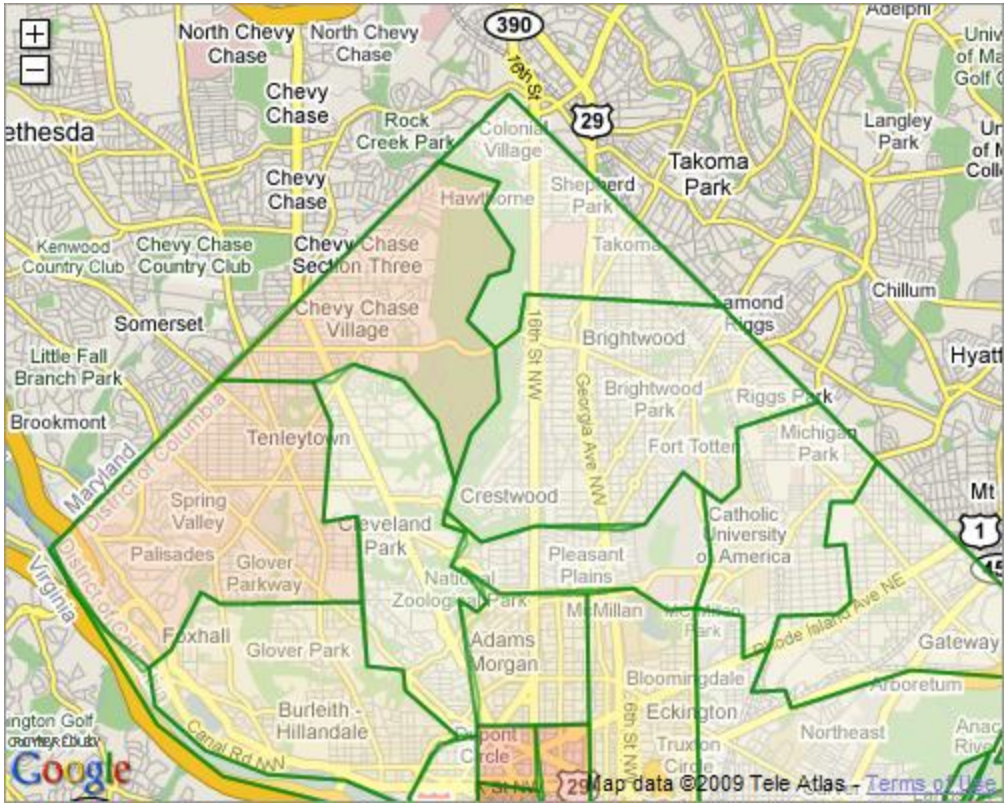
If you've used Google or Leaflet maps to provide geographic mapping solutions in your Logi application, you can take advantage of another advanced feature, **Map Polygons**. In Logi Info, this allows you to overlay solid or semi-transparent, color-coded polygons on a map, for enhanced analysis.

The following topics discuss the use of Map Polygons:

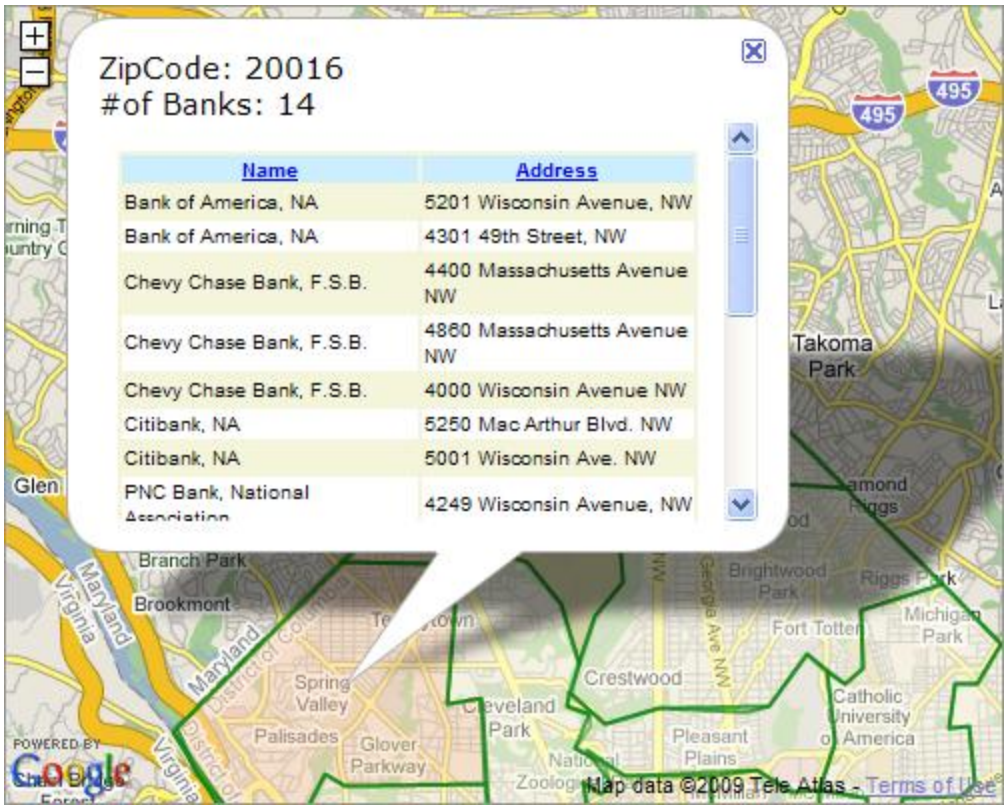
If you haven't already, you should first read "Leaflet Maps" on page 363 or "Google Maps" on page 339, which provide general information. Interested in plotting *lines* on the map? See "Google Map Polylines" on page 391.

About Map Polygons

Prior to v12.5, the element being discussed in this topic was called "Google Map Polygons". However, with the introduction of Leaflet maps, the "Google" part of the name was dropped. The element works in the same way for either mapping system. Map Polygons, plotted onto the map, are overlays or "regions"that can be color-coded based on data values. Here's an example:



The map above shows the northern half of Washington, D.C. with polygons representing different postal codes overlaid on it. The color of the polygon denotes the number of banks located within the region. The polygons have been configured to be semi-transparent, so that geographic features, such as streets, can be seen through them.



Polygons function like traditional Map Markers and can be related to other data. In the example above, they've been configured so that when they're clicked, a pop-up information panel is displayed containing more detailed information. Polygons and Map Markers can be used in the same map. Other examples of geographic polygons include political boundaries (such as states, counties, and cities), school districts, voting districts, and water management districts, to name but a few.

Geographic Area Boundary Data

Polygons are plotted using sets of **latitude** and **longitudepoints** that describe the boundaries of geographic areas. For use with Logi elements, these typically come from GPX or KML files, which are GIS industry-standard XML data files. Special datalayers are

provided in Logi Info to read these files. Because of the public nature of many of these geographic areas, their boundary data is often freely available on the Internet. Area boundaries are also widely available in *shapefiles*, a popular geospatial vector format for GIS data. Free and inexpensive software tools, discussed in "GIS Data Resources" on page 424, are available for converting shapefiles into GPX or KML data files. Geographic data can also be retrieved from SQL database tables using DataLayer.SQL - see "Using Data from SQL Server" on page 422 for more information.

General Features

The following describes additional features of Map Polygons:

- Borders, fill colors, and transparency level are all configurable and may be set from data values.
- Colors may be set in steps or as smooth color spectrums based on minimum and maximum data ranges.
- Polygons may be *included* or *excluded* based on data values.
- Polygon resolution is adjustable, and is dynamically sharpened as the user zooms the map in or out.
- Maps are initially displayed at a location and zoom level that shows all polygons.
- A special color spectrum legend element can be displayed below or alongside the map.

Implementing Map Polygons

The following example will demonstrate how to implement Map Polygons. The purpose of the example is to create a map of the state of Florida, with an overlay of all its public school districts, in order to easily compare "dropout rates" - the percentage of students who have dropped out of the school system - in the 2007-2008 school year. As is often the case, the first step is to have a look at the data. For this example, relevant annual dropout rate data from the State of Florida's Department of Education has been downloaded and stored as an XML data file.

```
<?xml version="1.0"encoding="utf-8"?>
<dropOuts District="ALACHUA" Y1999="5.7" Y2000="6.3" Y2001="6.1" Y2002="5.2" Y2003="5.1" Y2004="5.1"
Y2005="5" Y2006="6.1" Y2007="6.6" Y2008="3.6" />
<dropOuts District="BAKER" Y1999="9.7" Y2000="3" Y2001="4.2" Y2002="3.5" Y2003="3.7" Y2004="4" Y2005-
5="4.3" Y2006="3.7" Y2007="2.8" Y2008="1.8" />
<dropOuts District="BAY" Y1999="2.5" Y2000="3.5" Y2001="1.6" Y2002="1.5" Y2003="1.1" Y2004="1.8" Y2005-
5="1.2" Y2006="2" Y2007="2.5" Y2008="1.7" />
A small part of the XML dropout data is shown above; note that the field District contains the school
district name values.
```

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx xmlns="http://www.topografix.com/GPX/1/1" version="1.1" creator="ExpertGPS 3.03" xmlns:x-
si="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.topografix.com/GPX/1/1 http://www.to-
pografix.com/GPX/1/1/gpx.xsd http://www.topografix.com/GPX/gpx_overlay/0/3 http://www.topografix.com/GPX/gpx_
overlay/0/3/gpx_overlay.xsd http://www.topografix.com/GPX/gpx_modified/0/1 http://www.topografix.com/GPX/gpx_mod-
ified/0/1/gpx_modified.xsd">
<metadata>
<bounds minlat="24.54470100" minlon="-87.63493800" maxlat="31.00088800" maxlon="-80.03136200"/>
<extensions>
```

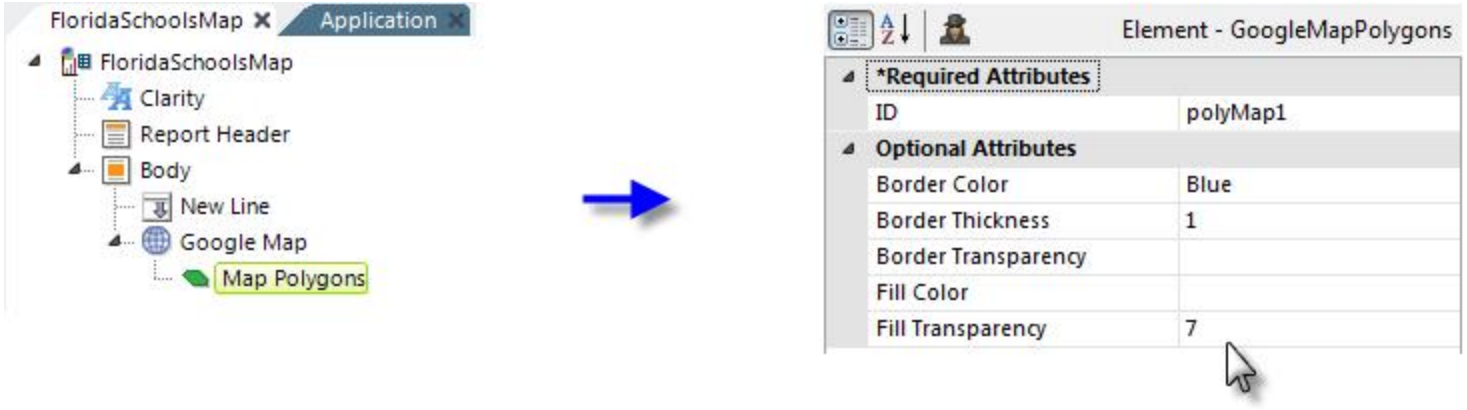
```
<time xmlns="http://www.topografix.com/GPX/gpx_modified/0/1">2009-04-20T19:40:01.535Z</time>
</extensions>
</metadata>
<extensions>
<polyline xmlns="http://www.topografix.com/GPX/gpx_overlay/0/3">
<desc>ALACHUA COUNTY SCHOOL DISTRICT</desc>
<label>
<label_text>ALACHUA COUNTY SCHOOL DISTRICT</label_text>
</label>
<extensions>
</extensions>
<points>
<pt lat="30.99692800" lon="-85.49827200"/>
<pt lat="31.00088400" lon="-85.24363200"/>
<pt lat="31.00083500" lon="-85.15445200"/>
<pt lat="31.00083400" lon="-85.15221800"/>
```

Similarly, the GIS data defining the boundaries of the state's school districts has been download and stored as a GPX file and a small part of it is shown above.

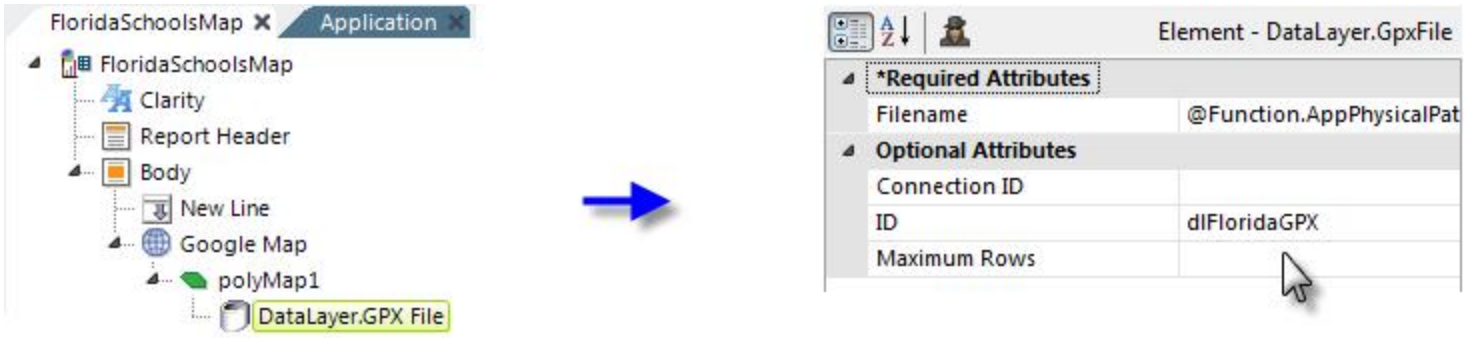
The school district name is part of the data in the *desc* field. The beginning of the latitude and longitude data for the district boundaries is at the bottom of the sample. In our Logi application, we'll relate the *schooldistrictnames* in these two files with each other in order to make our map overlay functional. The format of a KML file is somewhat different but it's still XML data and you can look at it easily enough to determine relevant field names.

Creating a Map with Polygons

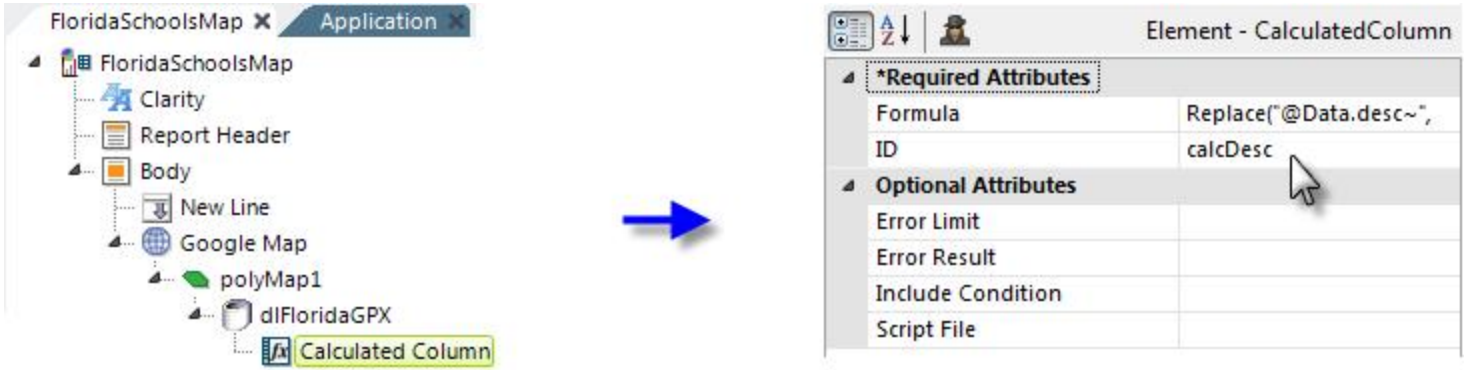
The following examples show an implementation using Google Maps, but the child elements described are used identically for a Leaflet Map. Before starting to create an application that uses Google Maps, you must get a Google Maps API Key, in order to access the Google web service. *Google Connections* provides information about getting and configuring an API Key. The example begins below with the assumption that you've gotten an API Key and that you've configured a **Connection.Google Maps** element in your _Settings definition with it.



1. In your report definition, add and configure a **Google Map** element to use your Google Map connection.
2. Beneath it, add a **Map Polygons** element, as shown above. This configures the polygons that will overlay the map and its **Border** and **Fill** attributes affect the appearance of the polygons. We'll come back later to set the **Fill Color** attribute.



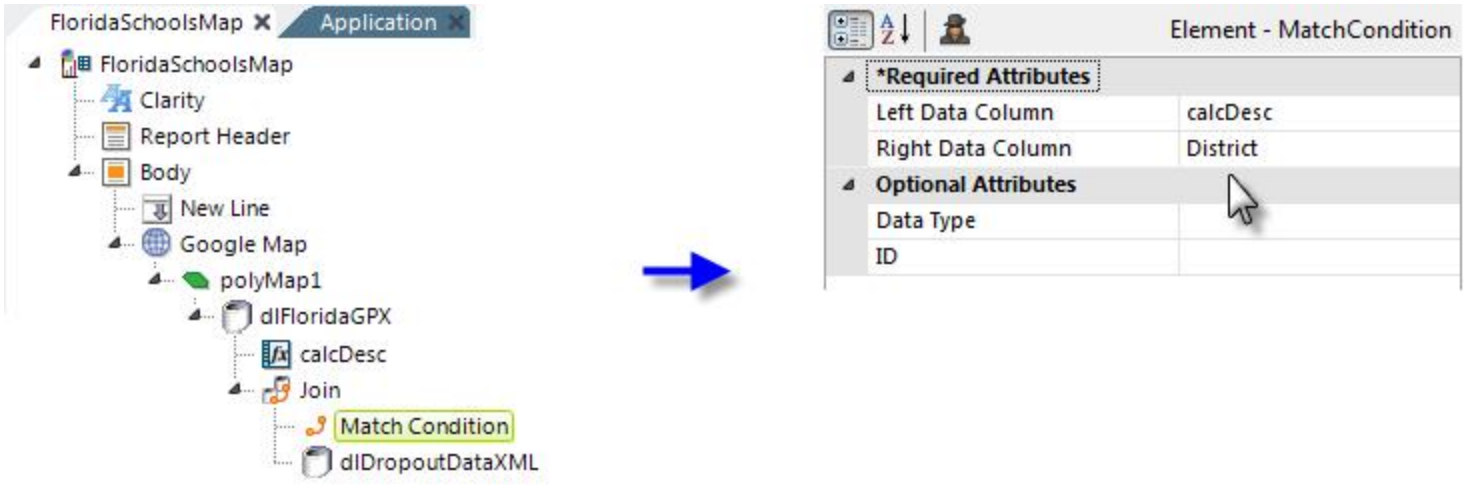
3. Beneath the Polygon element, add a **DataLayer.Gpx File** element, as shown above. It directly reads the GPX file we saw earlier to get the GIS data defining the boundaries of the state's school districts. If you're working with KML files instead, there's also a **DataLayer.Kml File** element for reading them. Tokens like @Function.AppPhysicalPath~ can make it easier to identify a file stored within your application folder. Or, add a **DataLayer.SQL File** element to the Polygon element if you are working with WKT formatted polygons and/or multipolygons.



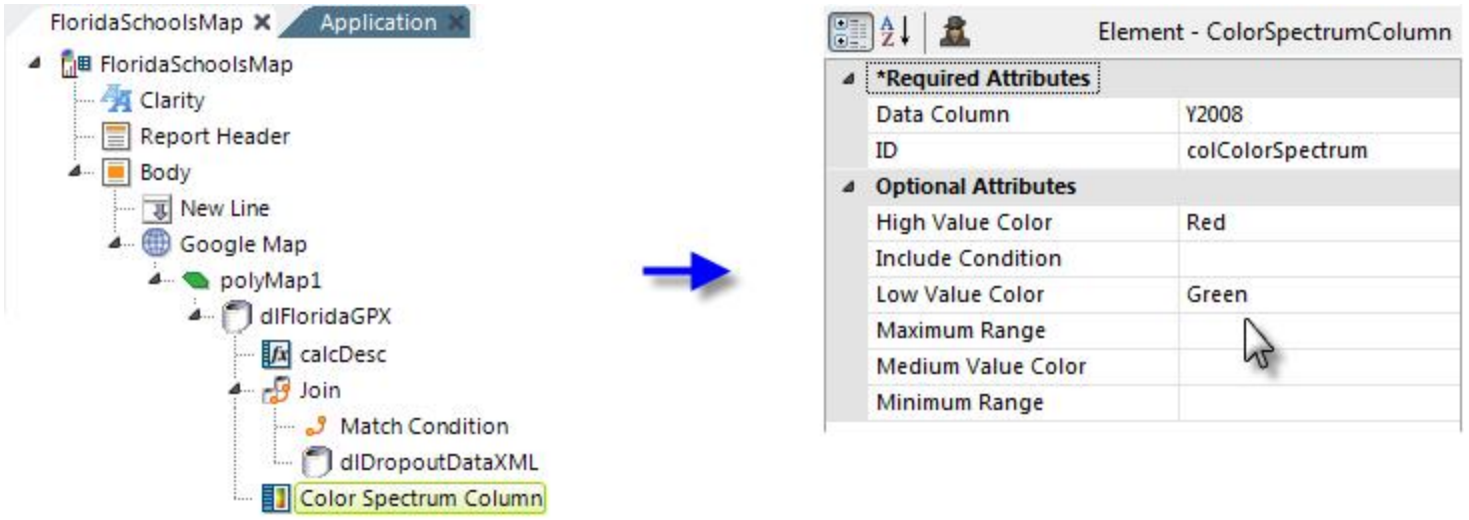
4. Beneath the datalayer, add a **Calculated Column**, as shown above. This column is used to compensate for the fact that the school district names are formatted differently in the two data files examined earlier. The complete formula attribute value is:

```
Replace("@Data.desc~", " COUNTY SCHOOL DISTRICT", "")
```

which truncates the data, creating an additional datalayer column that contains only the district name.



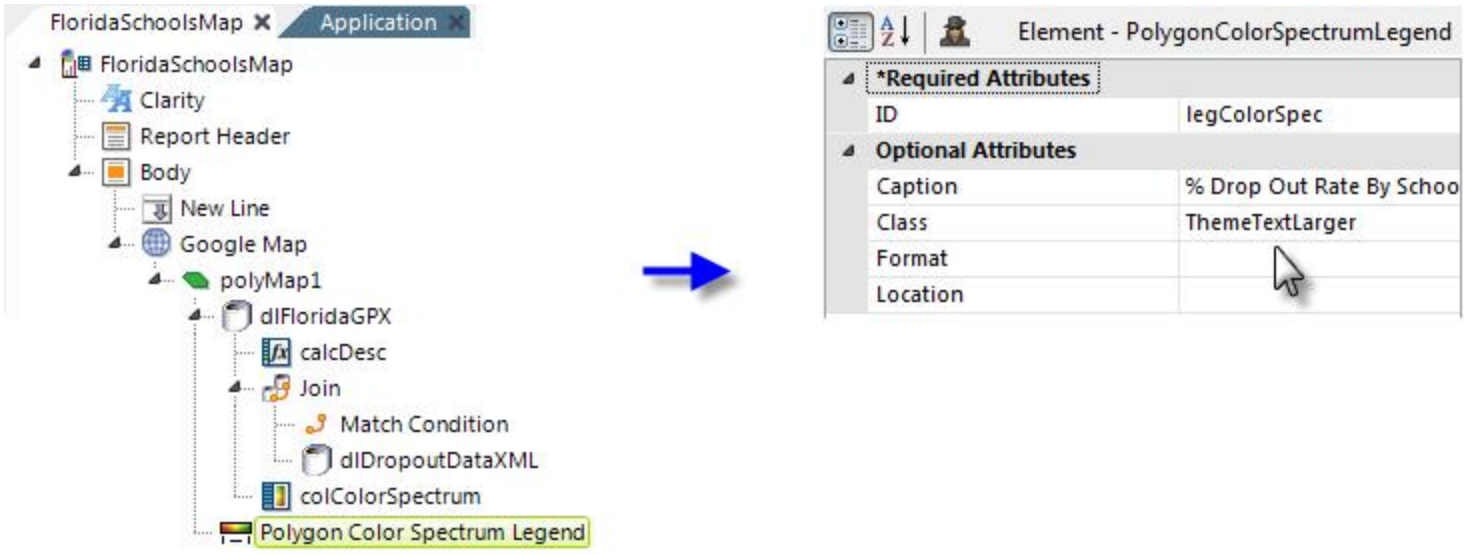
5. Next, add a **Join** element, a **DataLayer.XML** element and a **Match Condition** element, as shown above. The Join element is configured as an *Inner Join* and the datalayer retrieves data from the XML file with the dropout data we saw earlier. The Match Condition element is configured as shown above, relating the *District* column in the XML file with the calculated column *calcDesc* created in the previous step from the GPX file data.



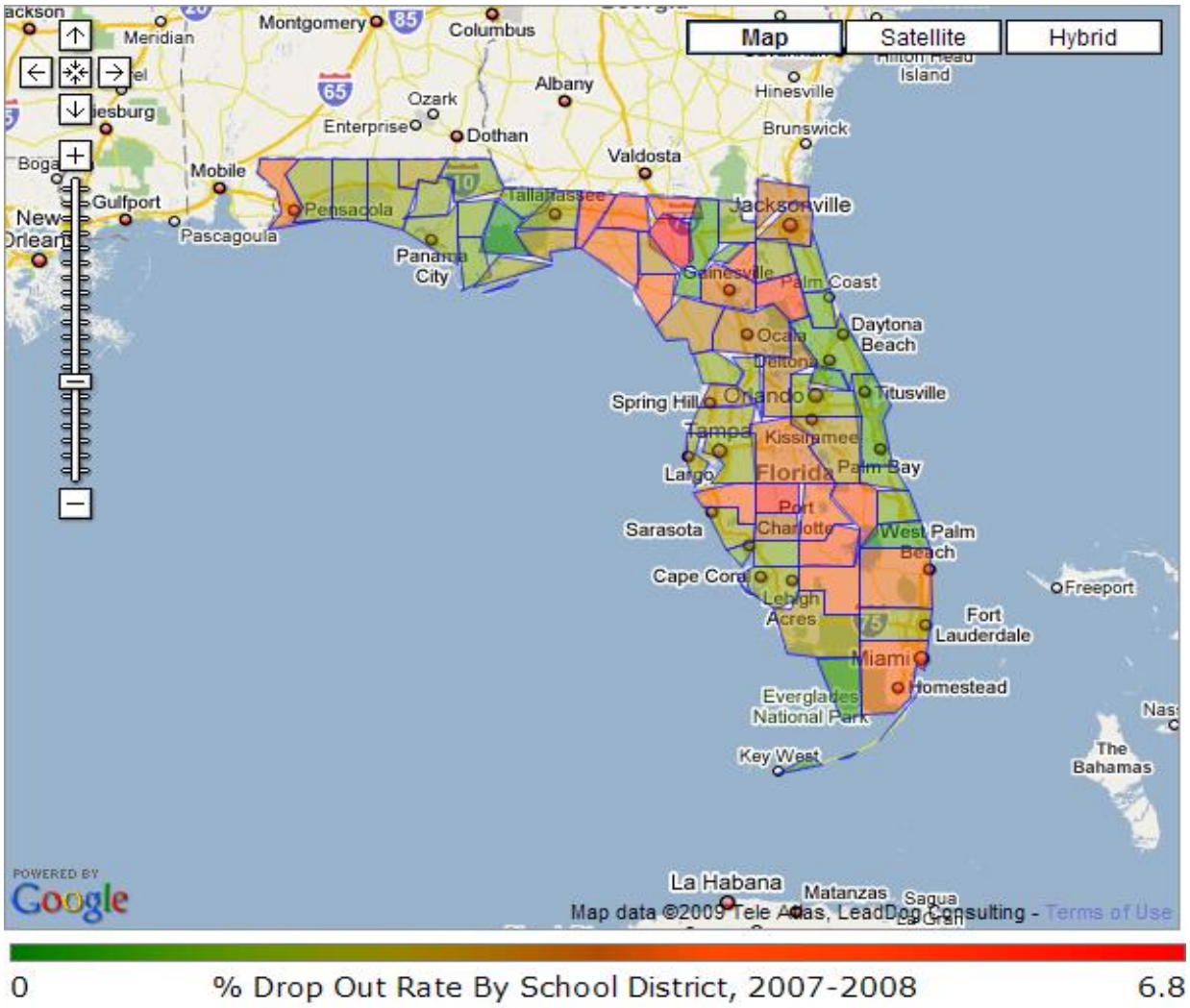
6. Select the DataLayer.Gpx File element ("dlFloridaGPX") and, beneath it, add a **Color Spectrum Column** element, as shown above. This column will contain the color value for each row in the datalayer, determining the color for each polygon on the map.

   The **Data Column** attribute value should be set to the *name* of the column with the data for the year we want to compare, 2008, and the color attributes are set to provide a color that ranges between *Red* and *Green* based on the dropout rate. Make a note of this element's **ID** attribute value, which will be used in the next step.

7. Reselect the Map Polygons element ("polyMap1"), as shown above, and set its **Fill Color** attribute value to the data token for the Color Spectrum Column added in the previous step.



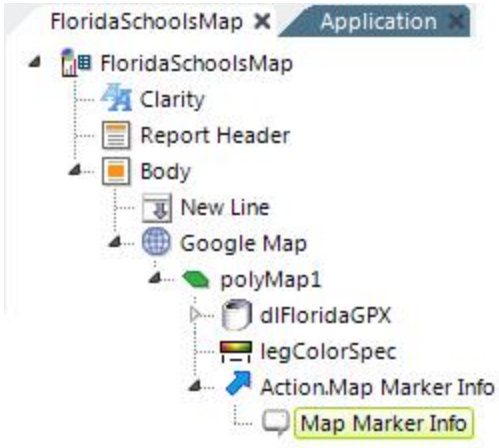
8. Finally, beneath the Map Polygons element, add a **Polygon Color Spectrum Legend** element, as shown above. This will create a color gradient legend beneath the map.



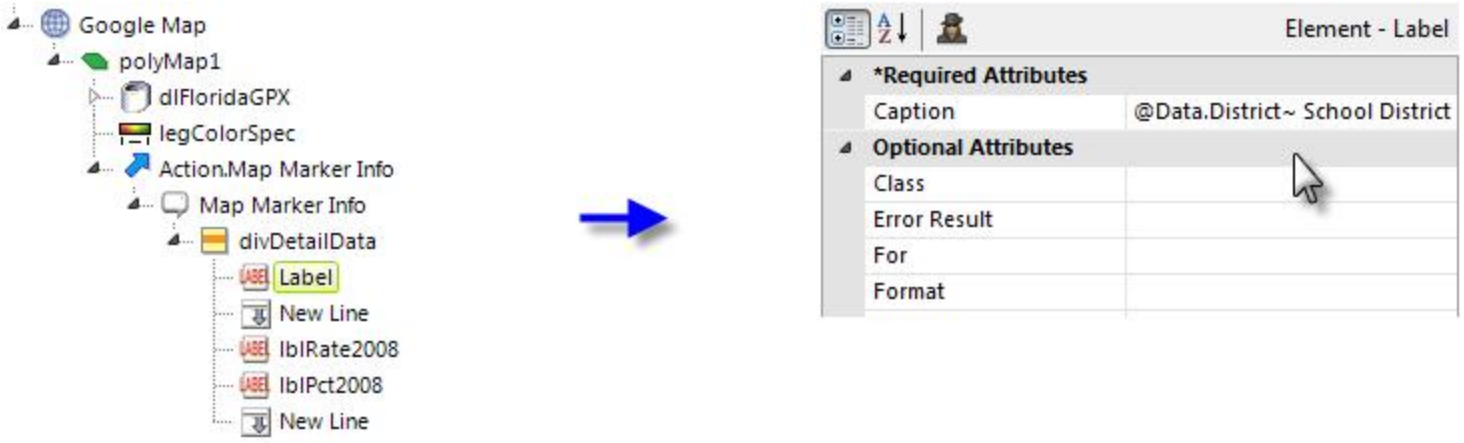
The resulting map looks like the example shown above. The map behaves like any Google Map and can be zoomed and re-centered. This topic continues on page two, with an example showing how to add drill-down capabilities to the polygons.

Adding Drill-down Capabilities

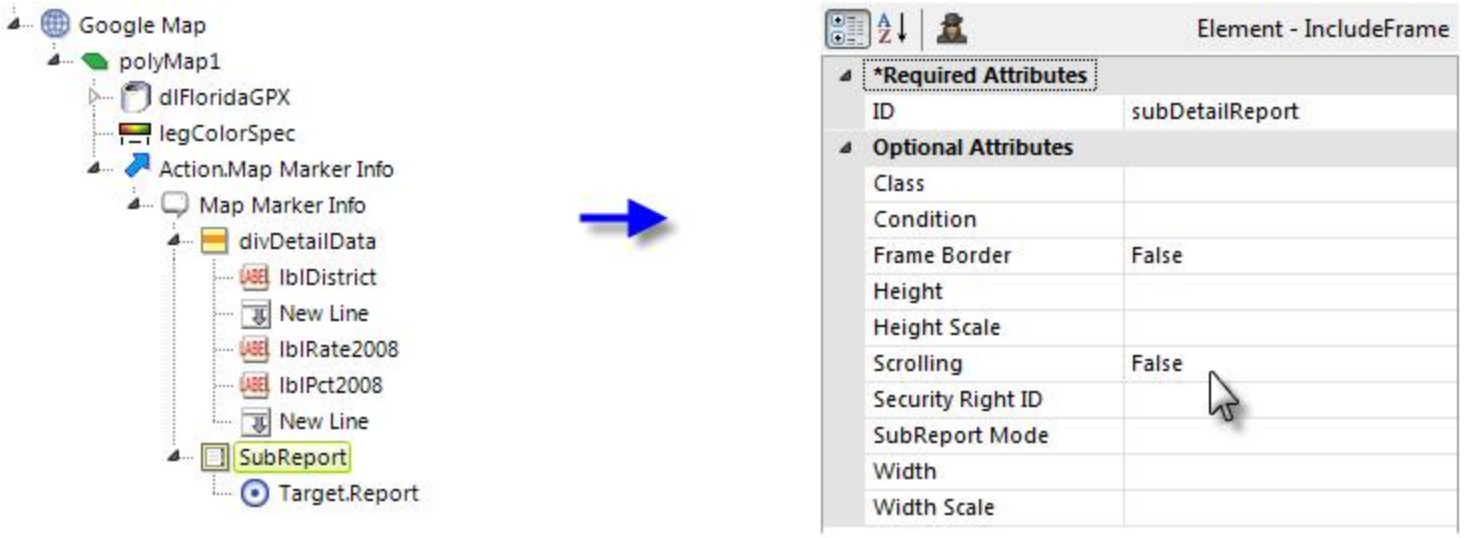
One of the most attractive features of Google Maps is their *drill-down* capabilities: clicking a Map Marker in a map causes a pop-up info panel to open, which can contain detail information. The same is true for Map Polygons - they can also be clicked in order to drill into the data. Continuing with the previous example, found in "Creating a Map with Polygons" on page 408, here's how this capability is added:



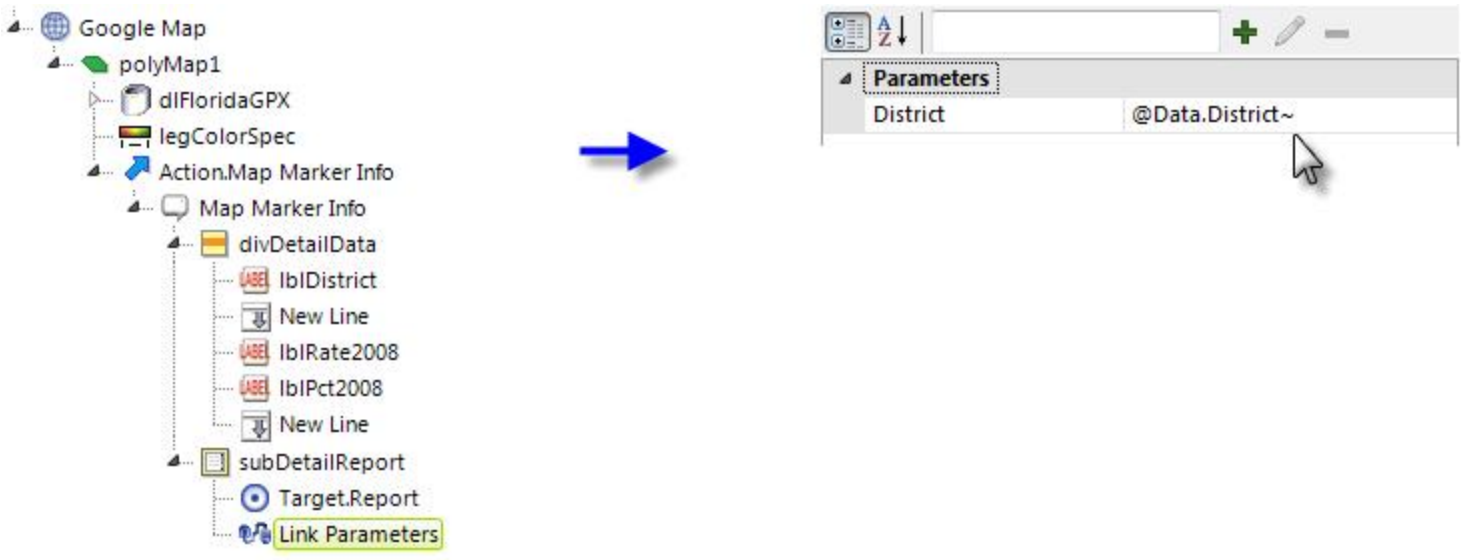
1. In the report definition, start by selecting the Map Polygons element. Beneath it add an **Action.Map Marker Info** element, and then beneath it, a **Map Marker Info** element, as shown above. Neither needs any configuration beyond being given an ID.



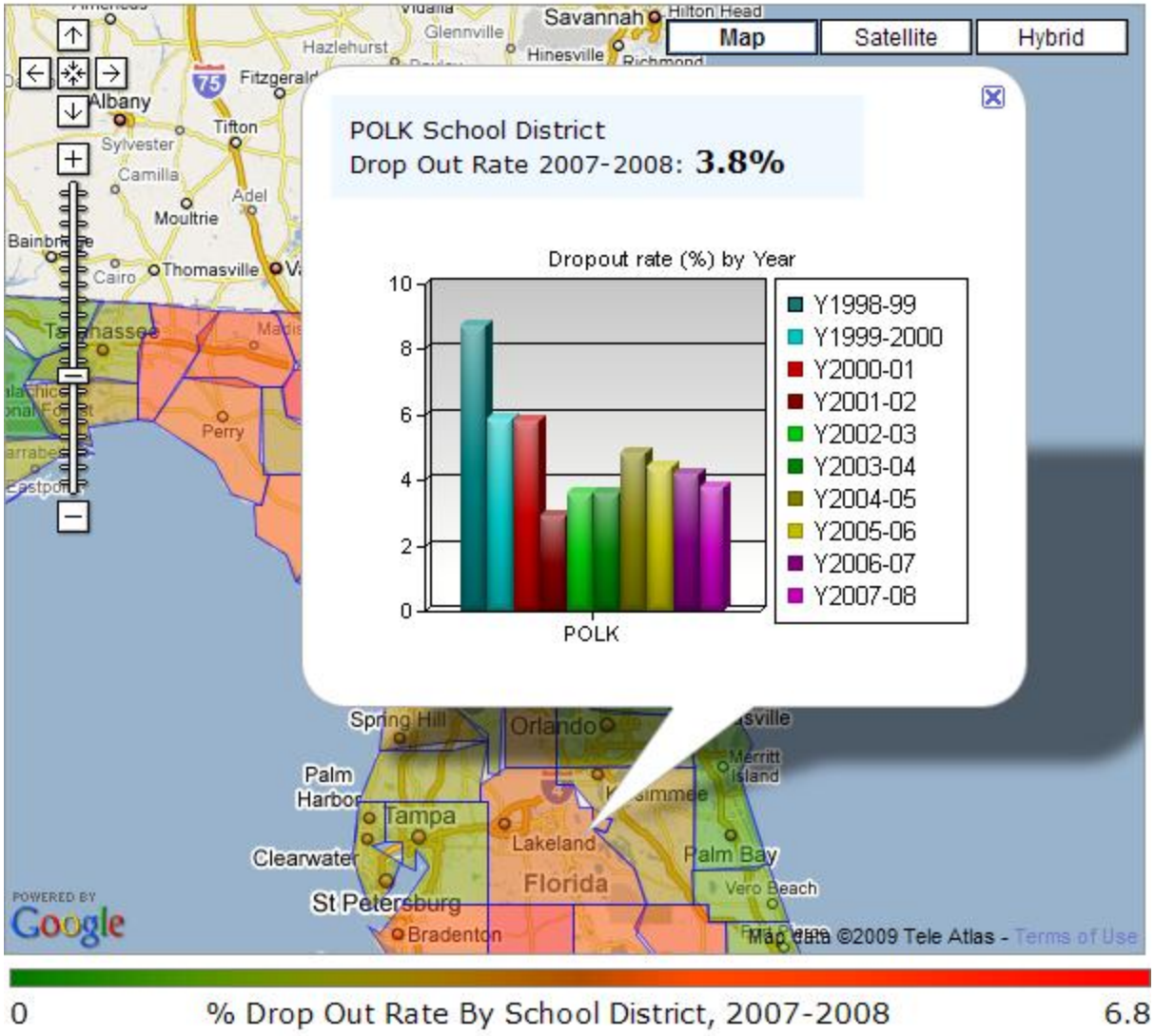
2. Beneath the Map Marker Info element, add a **Division** element and beneath it add several **Label** and **New Line** elements, as shown above. The Division allows a uniform style to be applied to the Labels, and the Labels display detail data from the datalayer, which is still in scope. Note the use of an @Data token to display the school district name.



3. Next select the Map Marker Info element again and beneath it add **SubReport** and **Target.Report** elements, as shown above. This will allow you to add a chart or table, defined in a separate definition, to the pop-up info panel. The **Frame Border** and **Scrolling** attributes are set to *False* so that the chart or table will embed smoothly into the pop-up panel.



4. Finally, a **Link Parameters** element can be added beneath the SubReport element, as shown above, to pass identifying information to the detail report.



The resulting pop-up information panel displayed when a polygon is clicked is shown above.

Tools for GIS Data Conversion

Data describing geographic region boundaries is used in Logi Info as either a **GPX** or **KML** file. In addition, data is also widely available in shapefiles (SHP). Note that these files need to use the standard Mercator projection, based on the WGS 1984 Ellipsoid projection, that Google Maps requires; "conic" projections, for example, will not work. A search of the Internet will yield many hits for software tools, free and paid, that can convert GIS data into these formats. Logi Analytics does not specifically endorse or recommend any particular tool but we have experimented with the following.

The following resources are provided to you as a courtesy. They're external resources not associated with Logi Analytics and we cannot guarantee that the links provided will remain viable. Developers are encouraged to search the Internet for their own resources.

SHP2KML 2.0

The SHP2KML program (freeware) available at http://www.zonums.com/shp2kml.htmlwill convert a SHP file into a Google KML file.

GPSBabel

The GPSBabel program (licensed under GNU - but should be fine for just about every use except re-distribution) is available at http://www.gpsbabel.org/and will convert KML to GPX format.

GPS Visualizer

http://www.gpsvisualizer.com/convert_input?convert_output=gpx
This is a free online conversion tool which is probably good for small files but not larger ones.

ExpertGPS

ExpertGPS makes it very easy to convert SHP files into GPX or KML formats and is available at http://www.expertgps.com/The program isn't free, but it's relatively low cost ($74.95 for personal use) and is available in a 30-day trial version. The program makes it very easy to convert between SHP and other file formats. When you download a .zip file containing SHP files, it will contain three files. You'll need to unzip all three files to a folder and then use the "Import" function of the ExpertGPS program to generate the GPX file (you'll only select the .shp file for import but the process reads from the other two files to get the appropriate labels and other metadata). To create the appropriate GPX file for Logi Info to work you need to make sure that you're populating the "Description" field with the boundary name from the SHP file (e.g. zip code, county, etc); it's suggested that you leave all the other fields empty to create the smallest possible GPX file.

Using Data from SQL Server

You can also set up the Map Polygons element to use DataLayer.SQL to retrieve geography data from a SQL database, for plotting geographic areas. Here, we describe examples using MS SQL Server. You may have to adjust them for your DB server, if you have things set up differently. When you use MS SQL Server Management Studio, your Geography type data is displayed as hex strings. However, if you look at the data in Logi Studio's SQL Query Builder, it appears as strings similar to:

POLYGON ((-77.15272360579938 38.899021866163743, -77.101466420629265 38.944705094496122, -77.095299197312215 38.985397708288836, -77.169278164468949 39.026928767466487, -77.220265629834756 39.006681859583395, -77.230661807874824 38.933657409186786, -77.15272360579938 38.899021866163743)) This format can also be displayed using a built-in SQL Server function called STAsText(). STAsText() (and other database versions) returns the coordinates in well-known text (WKT) standard: POLYGON ((LAT LONG,[...] , LAT LONG)). As opposed to Info, which expects the coordinate values in CSV format: Long1,Lat1 Long2,Lat2 Long3,Lat3 Long4,Lat4, with one polygon per line. To use polygons in Logi maps, geographic data needs to be transformed to include a column named *rdCoordinates*, which contains the polygon data as a string of coordinates in a comma-delimited sequence such as: `Long1,Lat1 Long2,Lat2 Long3,Lat3 Long4,Lat4` .

 While longitude and latitude values are separated by commas, coordinate pairs are separated by spaces.

Here's an example table:

Results | Messages

	state	rdCoordinates
1	AL	-87.3592810000,35.0018230000,0 -87.3492510000,35.0016620000,0 -87.2991850000,35.0009150000,...
2	AL	-88.1884692736,30.2469340542,0 -88.1950616507,30.2462354077,0 -88.2085400000,30.2448070000,...
3	AZ	-109.0452230000,36.9990840000,0 -109.0452440000,36.9694890000,0 -109.0452720000,36.96887100...
4	AR	-94.4760497582,36.4993199125,0 -94.4568835367,36.4993666550,0 -94.4525783697,36.4993771544,...
5	CA	-123.2307620000,42.0038450000,0 -123.1923610000,42.0054460000,0 -123.1549080000,42.00803600...
6	CA	-122.4187121729,37.8527169591,0 -122.4344027703,37.8524342437,0 -122.4433019345,37.85544848...
7	CA	-122.3785000544,37.8265049602,0 -122.3778785339,37.8306484157,0 -122.3699411346,37.83213667...
8	CA	-123.0139156612,37.7003546200,0 -123.0138972081,37.7044781080,0 -123.0121939825,37.70674907...
9	CA	-119.9128570000,34.0775080000,0 -119.8911300000,34.0728560000,0 -119.8735352565,34.07204559...
10	CA	-120.3622510000,34.0730560000,0 -120.3549820000,34.0592560000,0 -120.3588919552,34.05672498...
11	NY	-73.7733612559,40.8594490082,0 -73.7705522899,40.8603302522,0 -73.7663333113,40.8573166942,...

This data can be retrieved and plotted using DataLayer.SQL as a child of the Map Polygons element.

MultiPolygons

The Geography data type also supports "multipolygons", which can be thought of as a related group of polygons. For example, if a region represented a farm, a field in it might be split in two by a river running through it. Each part of the field would be a polygon, and the entire field would be a multipolygon.GoogleMapPolygons can now retrieve their coordinates from any DataLayer with the column rdCoordinates containing WKT formatted polygons and multipolygons. It is possible to create multiple rows of polygon data from a multipolygon through a JOIN. Contact Logi Customer Support for the details of this operation.

GIS Data Resources

The following resources are provided to you as a courtesy. They're external resources not associated with Logi Analytics and we cannot guarantee that the links provided will remain viable. Developers are encouraged to search the Internet for their own resources. These web sites provide free, public GIS boundary data in a variety of formats: **United States Census Bureau** - These SHP files are perfect for creating the boundaries for any US state or territory.
http://www.census.gov/geo/maps-data/data/tiger-cart-boundary.html**Texas State Data Center** - Most U.S. states will have information like this available.
http://osd.texas.gov/Data/Decennial/2010/
UC Berkley - Geographic data for most countries worldwide
http://gif.berkeley.edu/resources/data_subject.html**University of North Carolina** - Geographic data for most countries world-wide
http://guides.lib.unc.edu/content.php?pid=356445&sid=2914896
and there many other resources are available on the Internet.

Data Tables

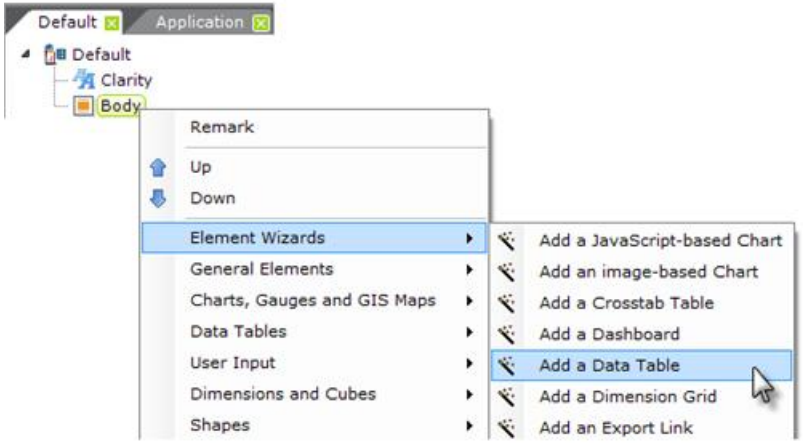
This topic introduces the Logi Data Table, which is one of the primary structures for displaying report data. It consists of a grid of rows and columns, which displays data (usually) in a row and column arrangement. These "tabular" displays in a report look good and are easy to work with.

The following topics provide more details about using Data Tables in Logi Studio:

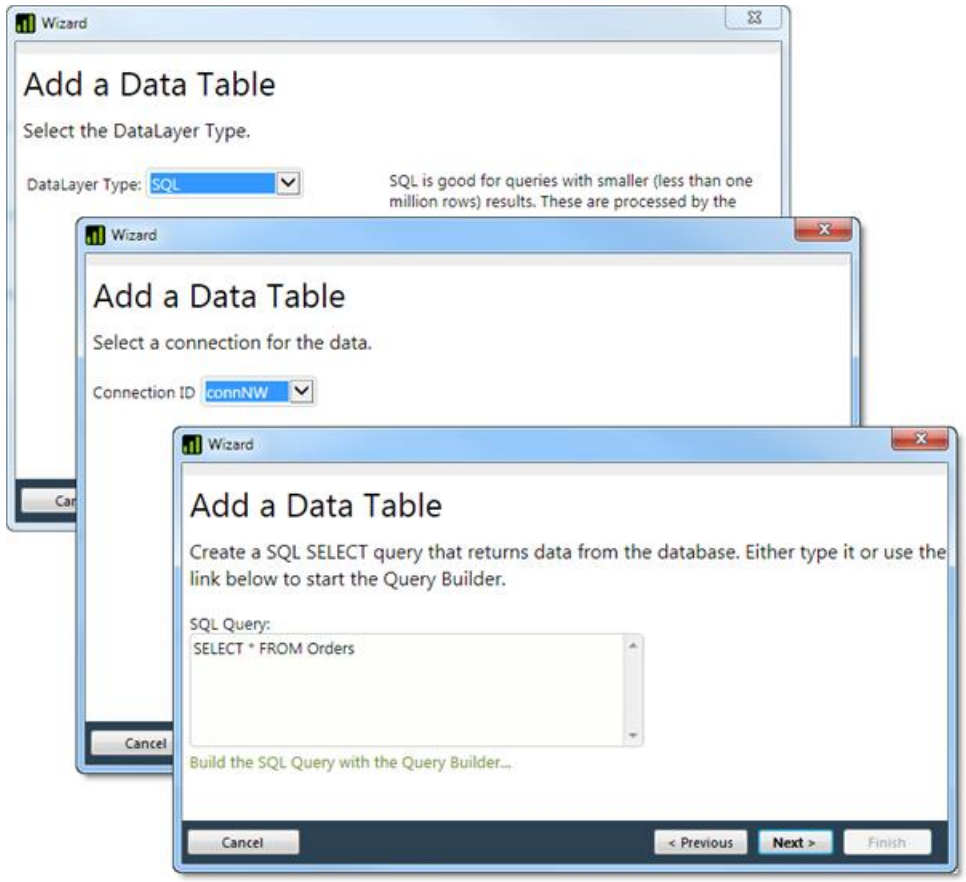
- Using the Data Table Wizard
- Data Table Basics
- Creating a Data Table Manually
- Data Table Links

Using the Data Table Wizard

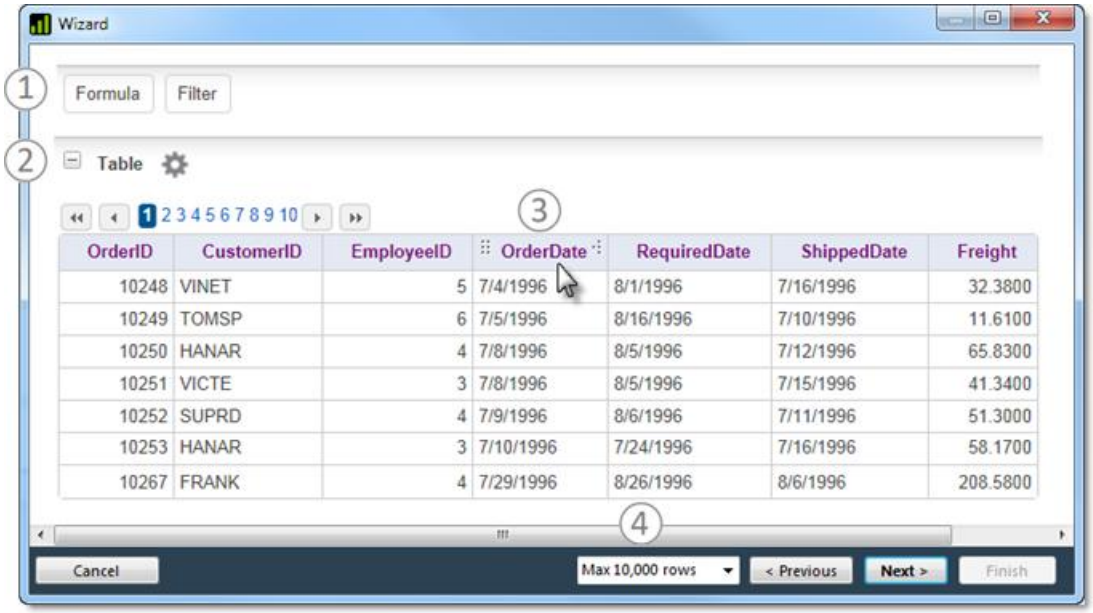
The fastest and easiest way to create a Data Table is to use the **Data Table Wizard** in Logi Studio. Here's how:



1. As shown above, in a definition in Logi Studio, select an element and either click the Data Table item in the main menu's Wizards tab, or right-click the element and select *Element Wizards*, and then select *Add a Data Table* from the secondary menu.



2. A series of dialog boxes, shown above, will be displayed. These all relate to the retrieval of the data. Make appropriate selections for your application and click **Next** to move to the next dialog box.

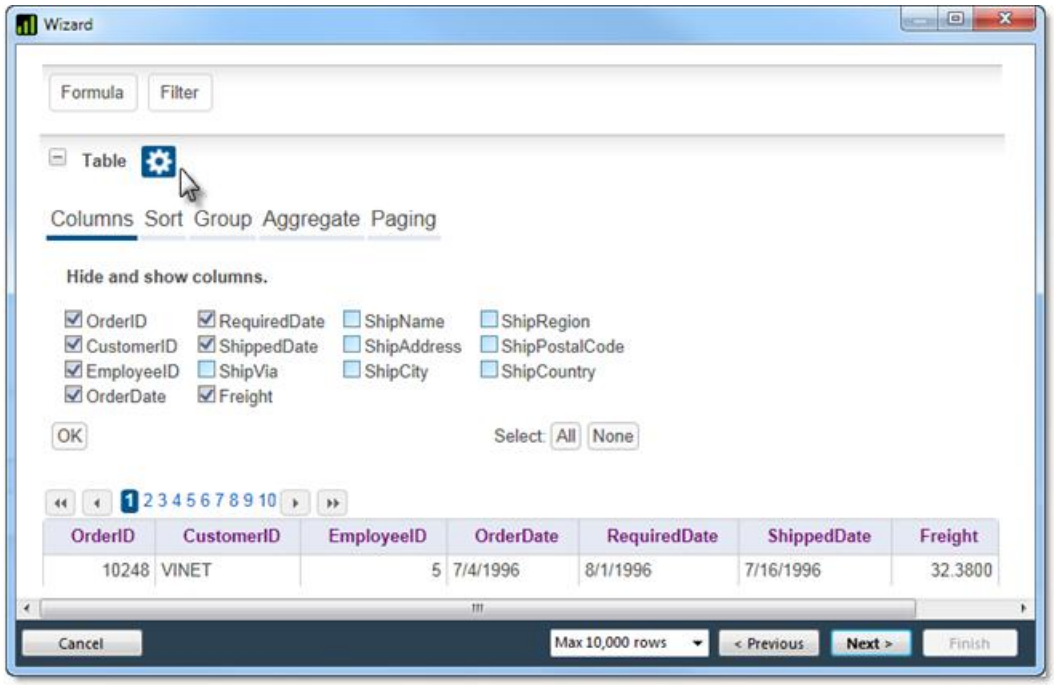


3. The wizard will display a model Data Table in a dialog box, where you can adjust it, if desired. If the table is just as you want it, click **Next** to exit the wizard. If not, you can continue the customization, using the features identified above:

   1- Click these buttons to add calculated or "Formula" columns, and to Filter the data based on column values;
   2- Click the gear icon to display the table's Configuration Panel (discussed below);
   3- Hover your cursor over a table column header and use the drag handles that appear to change the column width or rearrange column order. Click the header text to see a configuration menu for that column (discussed below);
   4- Select the maximum number of rows to be retrieved.

Table Configuration Panel

If you click the gear icon, the table's Configuration Panel will be displayed:



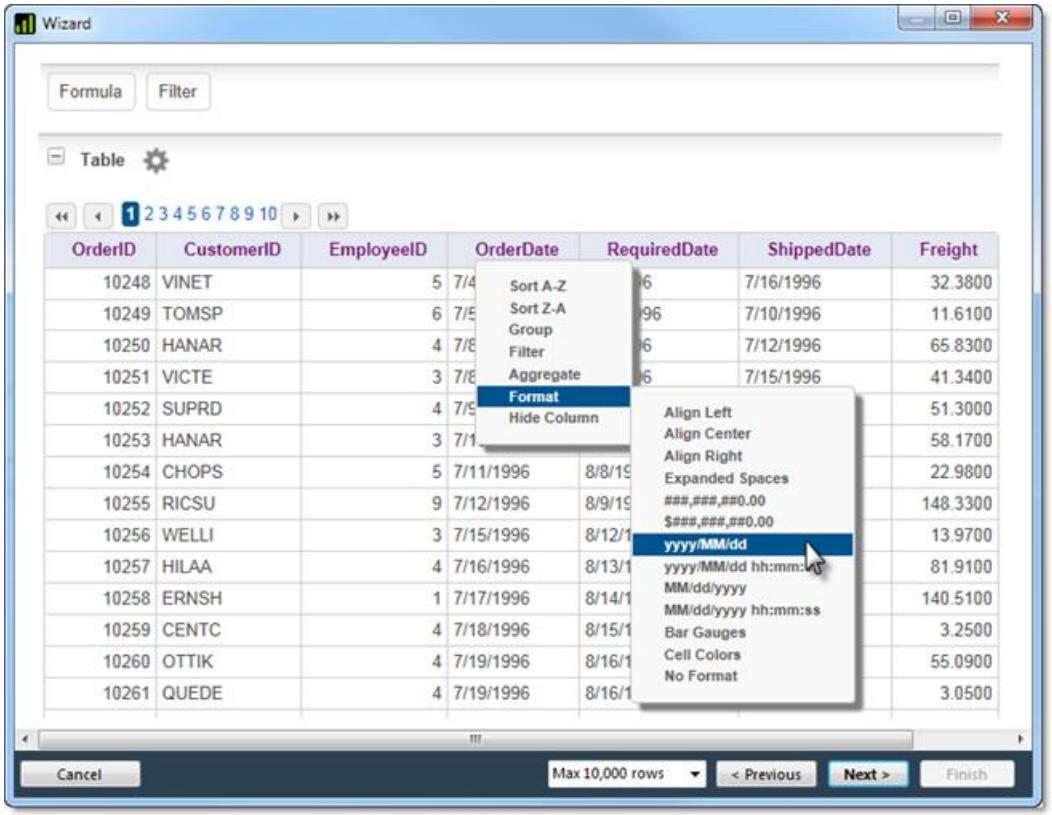
As shown above, the Configuration Panel includes these controls:

- **Columns** - Select the columns that will be displayed (displayed vs. retrieved in the data).
- **Sort** - Specify the sort order of the data after it's been retrieved.
- **Group** - Specify the grouping of the data after it's been retrieved.
- **Aggregate** - Add new columns for aggregations including *Sum, Average, Standard Deviation, Count, Distinct Count, Minimum,* and *Maximum*.
- **Paging** - Specify whether the table will use paging and, if so, how many rows will be shown per page.

Click the **gear** icon again to hide the Configuration Panel.

Column Configuration Menu

If you click a column's header text, a pop-up Configuration Menu for that column will be displayed:



Options on that menu allow you to do some of the same thing you can do in the Configuration Panel but other options, like Format, work specifically on the selected column. Changes in the displayed table will take affect immediately.

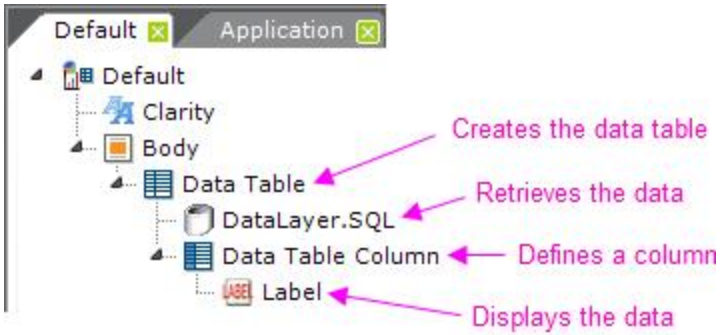
When you're done, the Wizard will add all of the elements for the table to your definition. The wizard automatically gives each column interactive sorting capabilities.

Only the elements for the table itself will be added, *not* any of the wizard's configuration controls you've just worked with.

If you already have your Data Table and datalayer elements added and configured, another wizard - "Add Data Columns" - is available when you select the **Data Table** element and is really useful if you don't have a list of table columns handy.

Data Table Basics

Data Tables are dynamic, data-driven reporting components. Unlike basic HTML tables built using the Rows, Row, and Column Cell elements, Data Tables are designed column-by-column rather than row-by-row.



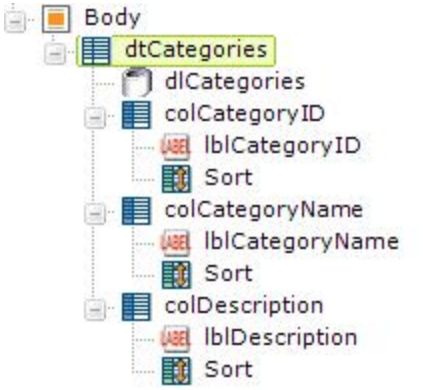
As shown above, every Data Table must have the following components:

- A **Data Table** element
- One or more **DataLayer** elements to retrieve and cache the data, possibly with child Filter elements
- One or more **Data Table Column** elements to display each column
- One or more report content elements, such as a **Label**, to present the data

This column-wise approach can be confusing. Just remember that, at runtime, *every record* in the datalayer will be iterated and rows for all of the data will be displayed in the table. Developers create and configure each column, while the number of rows is entirely dependent on the data.

The **Data Table Column** element, for example, is configured to provide the column header text, column width (if not automatically determined), and so forth.

The actual data values in the datalayer are accessed using *tokens*, such as @Data.CategoryID~, where the second part of the token is the column name in the datalayer. This case-sensitive token can be used in Label element Captions, for example, to display the data. For more information about tokens, see *Token Reference*.



ID	Category Name	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	Confections	Desserts, candies, and sweet breads
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish

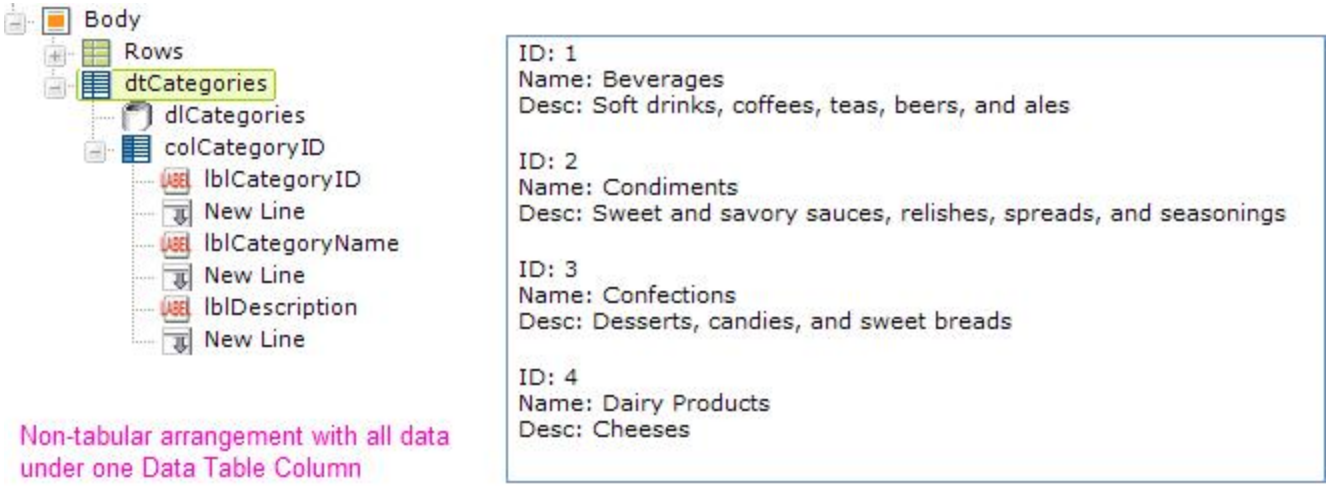
Tabular arrangement using multiple Data Table Column elements

The table shown above applied styles to it to specify its fonts, borders, and background colors. An easy way to quickly style a Data Table is to apply a Theme to your report definition. For more information, see *Working with Themes*.

The most common way to design Data Tables is to have a **Data Table Column** element for each column in the datalayer, as shown above. This works well when you have many records that you want to show in a table.

The *order* of the columns in the displayed Data Table is governed by the order of the Data Table Column elements in the definition. The top-most of these elements in the definition will be the left-most column in the displayed Data Table. Developers are free to rearrange the order of columns by selecting a Data Table Column element in Logi Studio and dragging it, or by using the Up/Down Arrow buttons in the toolbar. This is particularly useful if a wizard has been used to create the Data Table and a different column arrangement is desired.

The one-to-one correspondence of data columns to Data Table Columns elements discussed above is *not required* and developers have complete freedom over the way data is presented in a report. Suppose, for example, that you don't want the typical tabular format.



Suppose instead you want to stack the data in a kind of form when you display it. This is easy to do, as shown above. The data is presented in a form-like manner by using just *one* Data Table Column element and placing multiple **Label** elements within it. Each Label element displays the data from a different data column in the datalayer. There's no limit on the number of elements you can place beneath a single Data Table Column element and you can, in fact, even place HTML tables and charts there!

Adding **HTML Attribute Params** to your Data Table enables you to apply your application to work with other frameworks or libraries easier. Depending on the type of HTML Attribute Params you add, there will be different parameters available to use, or you can define your own parameters. If an attribute was set in both Element and HTML Attribute Params, the one set in the HTML Attribute Params will be ignored.

Important Scope Limitation The *scope* of the data in a datalayer is limited to the **Data Table** element that contains it and its child elements. So, looking at the example above, the token @Data.CategoryID~ is valid in all elements beneath the Data Table *dtCategories*, but will be empty if you attempt to use it elsewhere, such as in the Report Header element.

The only exception to this scope limitation is the **Local Data** element. This element works with a datalayer which retrieves just one result row but its data is available, using the @Local.<*columnName*>~ token, anywhere in the report definition.

Creating a Data Table Manually

You can create a Data Table in Studio manually by adding the appropriate elements to a report definition.

To do this, simply add the necessary elements to your definition in Studio. After you add your first **Data Table Column** and its child elements (Label, Sort, etc.), you can speed things up by copying and pasting it into the element tree as many times as necessary, then going back and configuring the individual elements. Copying the Data Table Column element will also copy all of its child elements, too.

In these circumstances, the issue of **spelling** becomes more important. @Data tokens are case-sensitive and column names must be spelled accurately. To ensure that you know exactly how column names are spelled, you may care to turn on **debugging** and view the actual data in the datalayer. For more information about debugging, see *Debug Reports*.

Data Table Attributes

The Data Table element includes the following attributes:

Attribute	Description
Accessible Headers	Specifies whether or not to set the HTML "Headers" attribute, which can improve the accessibility and readability of tables. Default value: *False*.
Accessible Summary	Specifies whether or not to create a "table content summary", which has no visual effect but is used by screen readers to improve accessibility. Default value: *False*.
Ajax Paging and Sorting	Enables AJAX-style paging and sorting. When the user moves to another page of table data, or sorts a column, only the table portion of the page is updated. This method prevents the web page from flickering

Attribute	Description
	or losing its scroll position. This alters the behavior of the browser's "Back" button because the page history is updated with AJAX Paging. Default value: *False*
Alternate Row Class	Specifies a style class to be used for every other row of the table, making it easier to read across the columns in single row. Themes include a class called **ThemeAlternatingRow** which you can assign, if desired.
Border	Specifies, in pixels, the width of a border, if any, to be displayed around the table. A value of 1 displays a thin border.
Caption	Specifies the text of a caption, or "title", that will appear above the table. Leave this blank to suppress it.
Caption Class	Specifies a style class to be applied to the caption text entered in the previous attribute.
Cell Spacing	Specifies the width, in pixels, of the space between table cells, if any. Default: *0*.
Class	Specifies a style class to be applied to the table as a whole.
Column Header Class	Specifies a style class to be applied to the column headers.
Draggable Columns	Specifies if the user can reposition columns by dragging the column header side-to-side. When enabled, any changed column positions are remembered for the table for the current session, and can be stored in bookmarks for future use. Draggable Columns may not be used when there are custom header rows in use with columns that span multiple columns. Default: *False*

Attribute	Description
Hide Header If No Data	Controls the ability to hide the Data Table/Crosstab Table header when there is no data to show. Default: *False*. To hide the header when there is no data, set this constant to *True*.
Hide When Zero Rows	Specifies it the Data Table is to be hidden when the datalayer returns zero rows. Default: *False*
Keep Scroll Position	When enabled, causes the browser's horizontal and vertical scroll positions to be retained when the current report is redisplayed without first going to another report. Use this to refresh the current report while avoiding resetting the scroll position to the top. Default: *False*
Layout	Specifies if the table layout should be automatically determined by all the data (*Auto*) or just the first row of data (*Fixed*). If *Fixed* is specified, the column layout is determined based on the width of the values in the *first* data row, instead of adjusting the column widths by examining data in *every* row. This can help with formatting issues, and can also significantly improve large table performance. Default: *Auto*
No Data Caption	Specifies the message that displays in the Data Table/Crosstab Table when there is no data to display. Default value: *None*, no message displays. Enter a custom message as the value for this constant, or, enter 'Default' and users will receive the message "No data to display".
Remember Sort	Specifies whether or not to retain the user's sort selection. When *True*, and the user sorts the Data Table on a particular column, the report will be redisplayed with the same sort order the next time the report is displayed, and can be stored in bookmarks for future use. Default: *False*
Resizable Columns	When enabled, a small icon appears on the right edge of the column header when the mouse hovers over it and the user can resize columns at runtime by dragging the icon right or left to a different width. The

Attribute	Description
	updated column width are remembered for the table for the duration of the current session. You may not use Resizable Columns when there are custom header rows in use with columns that span multiple columns. Default: *False*
Row Class	Specifies a style class to be applied to the table rows. This value can be an @Data token, so the class can be applied dynamically based on the data.
Security Right ID	When specified, access to this element can be controlled with Logi Security. Specify the ID of a Right defined in the application's _Settings Security element. Only users with matching rights will be able to see the element.
Sort Arrows	Specifies whether Sort direction indicators (arrows) will be displayed when the user clicks a column header to sort the table. Default: *False*
Width	Specifies the total width of the Data Table.
Width Scale	Specifies the scale to be used when applying the value of the previous attribute, *px* for pixels, or % for percentage of the table's parent container's width.

Themes automatically set a number of these attributes "behind-the-scenes".

Adding **HTML Attribute Params** to your Data Table enables you to apply your application to work with other frameworks or libraries easier. Depending on the type of HTML Attribute Params you add, there will be different parameters available to use, or you

can define your own parameters. If an attribute was set in both Element and HTML Attribute Params, the one set in the HTML Attribute Params will be ignored.

In general, the fewer attributes you have to set for a Data Table, the better. Let a Theme or the browser figure out the best combination of attributes for you.

About Table and Column Widths

Browser engines are designed to do the best job they can with HTML tables (which is the structure underlying a Data Table) when it comes to automatically figuring out column widths. Here are some tips to make widths work for you.

First, *always set a width* for a Data Table, using its **Width** attribute.

Second, for most purposes, leave the Data Table element's **Layout** attribute *blank*. This defaults to *Auto* mode, which sets columns width to the widest data.

Third, if you're concerned about ensuring that one or two columns have a specific width, then set those widths in the appropriate Data Table Column elements but leave the widths for the *other* Data Table Column elements *blank*. You don't need to try to figure out what the width of every column will be, ensuring it all adds up to 100% of the table width. The browser will try to guarantee the widths you've specifically set and will divide and allocate the remaining space among the remaining columns, when it can.

And remember that it *is* possible to put so many conflicting restrictions (via attribute values) in place that the browser will get confused and produce weird results. If that happens, try removing width attribute values for columns until things look normal again, then reconsider the issue.

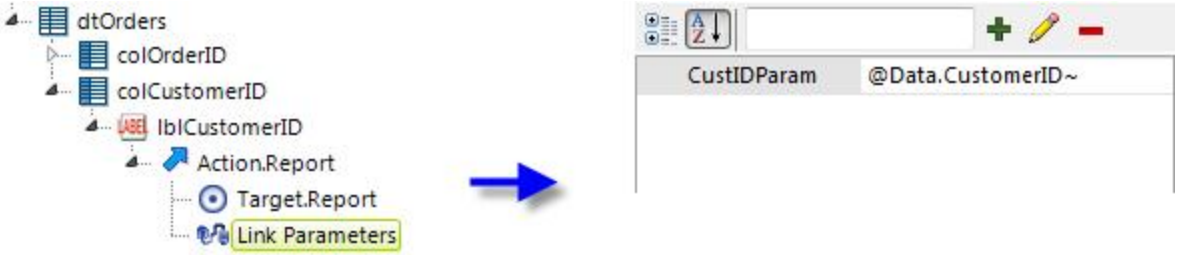
Data Table Links

A common requirement is to be able to click the actual data in a Data Table cell and do something with the clicked value.

Order ID	Cust ID	Emp ID	Order Date
10248	VINET	5	7/4/1996
10249	TOMSP	6	7/4/1996
10250	HANAR	4	7/8/1996
10251	VICTE	3	7/8/1996
10252	SUPRD	4	7/9/1996
10253	HANAR	3	7/10/1996

The image above shows an example of this, and here are the elements needed to make that work:

As shown above, an **Action** element is placed beneath the Label element used to display the data in each column. An appropriate **Target** element is also included.



Finally, a **Link Parameters** element is added, as shown above, to pass a data value to the target.

Data Table Columns

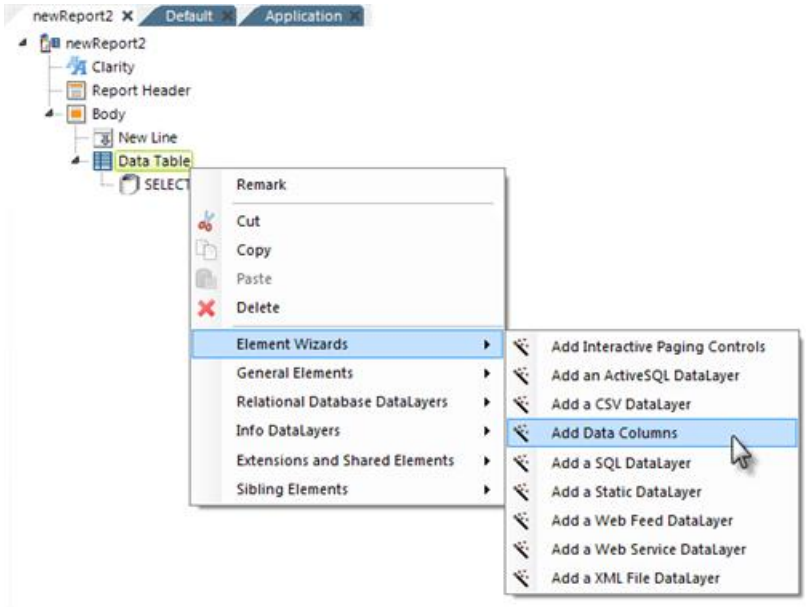
The following topics help developers customize column headers, add interactive sorting capabilities, and create various types of aggregations based on column data:

- Using the Add Data Columns Wizard
- Data Table Column Attributes
- Working with Column Headers
- Justifying Column Header Text
- Rearranging and Resizing Columns at Runtime
- Sorting Displayed Column Data
- Showing/Hiding Columns
- Setting Dynamic Column Text and Background Colors
- Summarizing Column Data
- Using Event Handlers with Column Data

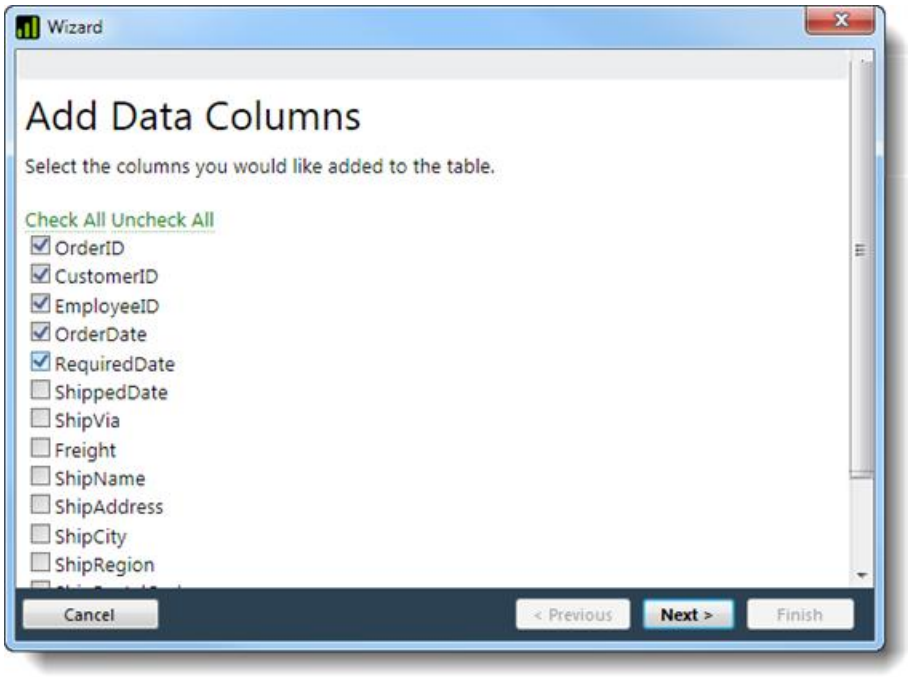
Using the Add Data Columns Wizard

You can add columns beneath a Data Table manually in Logi Studio using the Element Toolbox or context menus, but Studio also provides a convenient wizard for adding them. This wizard is especially useful if you're not sure of the exact number or spelling of column names in the table's datalayer.

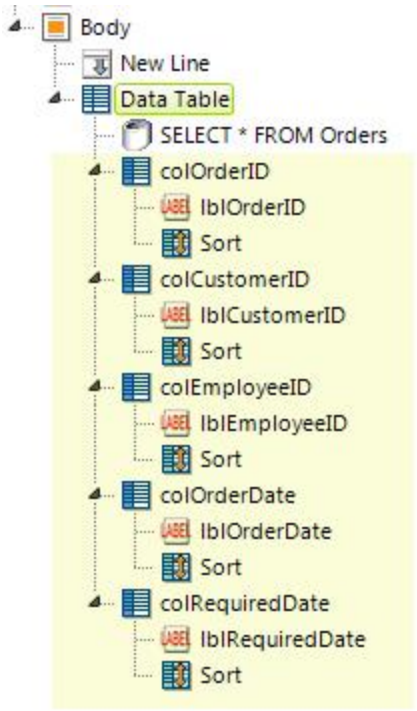
You must have correctly configured Connection (if required) and Datalayer elements in your definition before you can use this wizard.



Select and right-click a Data Table element, as shown above, and use the context menus shown to launch the **Add Data Columns** wizard.



The wizard will run the datalayer and present a list of the columns returned in the datalayer, as shown above. You can then select the columns you want added to the definition.



The wizard will insert elements for the selected columns, including a **Data Table Column**, **Label**, and **Sort** element for each column, into the report definition, as shown above. The elements are configured generically for immediate use but, at a minimum, you'll probably want to adjust the Data Table Column elements' **Column Header** attributes.

Data Table Column Attributes

The **Data Table Column** element has the following attributes:

Attribute	Description
Background Color	Specifies the background color of the of the column. Enter a color by name, decimal RGB value, or hex RGB value (prefix hex values with the pound sign, e.g. #112233). *Click the browse button to display the Color Picker tool.* Data tokens can also be used here, potentially resulting in a different background color for each table cell. For more information, see *The Color Range Column*.
Class	Specifies a style class to be applied to the column.
Column Header	Specifies the text to be displayed in the column's heading. In special cases in which an Extra Header Row element is used to make grouped column headings, and one of its columns has a Row Span greater than 1, this attribute's value may be set to *None*. This prevents the header column from getting created and thus pushing other column headers into the wrong positions. An image file can be specified here instead of text, if Header Type is set to *Image*.
Column Header Class	Specifies a class for the Column Header text or image. Use this attribute to set the alignment and other styling of the Column Header text or image. If left blank, the text or image will be centered.
Condition	Specifies an expression that evaluates to a value of *True* or *False*. If *True*, then the column is displayed, otherwise it's removed. Expressions should be in JavaScript or intrinsic functionsyntax. Typically, expressions compare values using a token, such as `@Data.value~ < 0`. Enclose both sides of the expression in quotes

Attribute	Description
	when working with strings: `"@Data.myColumn~" == "SomeValue"`.
Header Type	Specifies if the header will be text (the default) or an image.
ID	Specifies a unique element ID.
Scope Row Header	Specifies if the HTML output for data cells will be modified to improve Section 508 accessibility. Set to *True* to output a column which includes header information for each row, changing the `<TD>` tags to `<TH Scope-e="Row">` tags.
Security Right ID	When specified, access to this element can be controlled with Logi Security. Specify the ID of a Right defined in the application's _Settings Security element. Only users with matching rights will be able to see the element.
Show Modes *removed, use Condition instead*	Specifies a text string that controls whether elements will be displayed or hidden. Leave this blank for the element to always be displayed or set it to *None* to hide it (it can be displayed again later with an `Action.ShowElement` element). Set it to your own string value to have it appear only when the report's (root element's) Show Modes includes that value. Show Modes can contain multiple, comma-delimited values.
Text Color	Specifies the color of the text in the column. Enter a color by name, decimal RGB value, or hex RGB value (prefix hex values with the pound sign, e.g. #112233). *Click the browse button to display the Color Picker tool.* Data tokens can also be used here, potentially resulting in a different text color for each table cell. For more information, see *The Color Range Column*.
Tooltip	Specifies text that appears when the user hovers the mouse over the column header.

Attribute	Description
Width	Specifies the width of the element, in Width Scale units.
Width Scale	Specifies the units, pixels or percentage, to be used with the Width attribute.

Adding "HTML Attribute Params" to your Data Table enables you to apply your application to work with other frameworks or libraries easier. Depending on the type of HTML Attribute Params you add, there will be different parameters available to use, or you can define your own parameters. If an attribute was set in both Element and HTML Attribute Params, the one set in the HTML Attribute Params will be ignored.

Working with Column Headers

A *column header* contains optional information at the top of a specific Data Table column. Generally, column headers consist of text but, in a Logi application, they can include images and even user input controls, like buttons and links. Text or images may be a link that, when clicked, sorts the table on the column in alternating ascending and descending order.

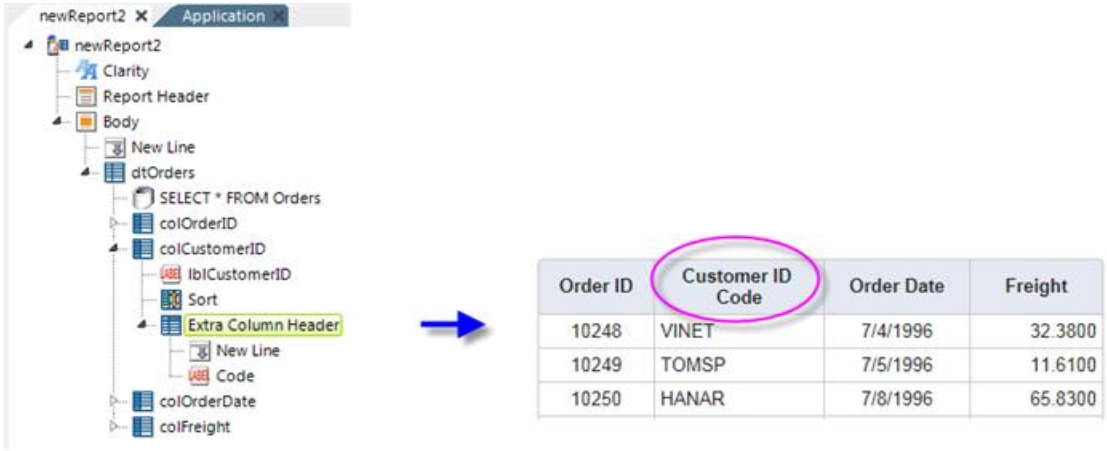
Order ID	Customer ID	Order Date	Freight
10248	VINET	7/4/1996	32.3800
10249	TOMSP	7/5/1996	11.6100
10250	HANAR	7/8/1996	65.8300

The Column Header row is the first row in the table, as shown above. To configure a column header:

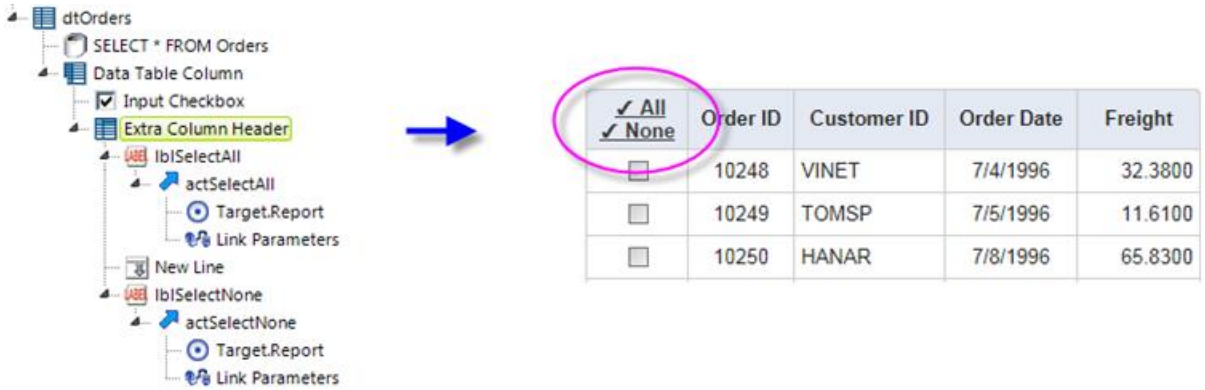
1. Select one of the **Data Table Column** elements in the definition.
2. Set the **Header Type** attribute to *Image* if an image is desired in the column header. If left blank, the default value is *Text*.
3. In the **Column Header** attribute value, type the desired text or choose an image from the drop-down list of choices.
4. The column header text is *centered* by default. However, you can apply a different class to *all* of the column headers at once by setting the parent Data Table element's **Column Header Class** attribute, or apply one individually to one or more columns by using their own Column Header Class attribute.

Using Extra Column Headers

Column headers normally consist of a single line title, but Logi Studio makes it easy for developers to add additional content if necessary. Developers can create extra column headers with multiple rows, containing text, links or images.



In the example above, an **Extra Column Header** element has been added as a child of a Data Table Column. It contains New Line and Label elements and the effect is to make a 2-line column header for the Customer ID column, as shown.

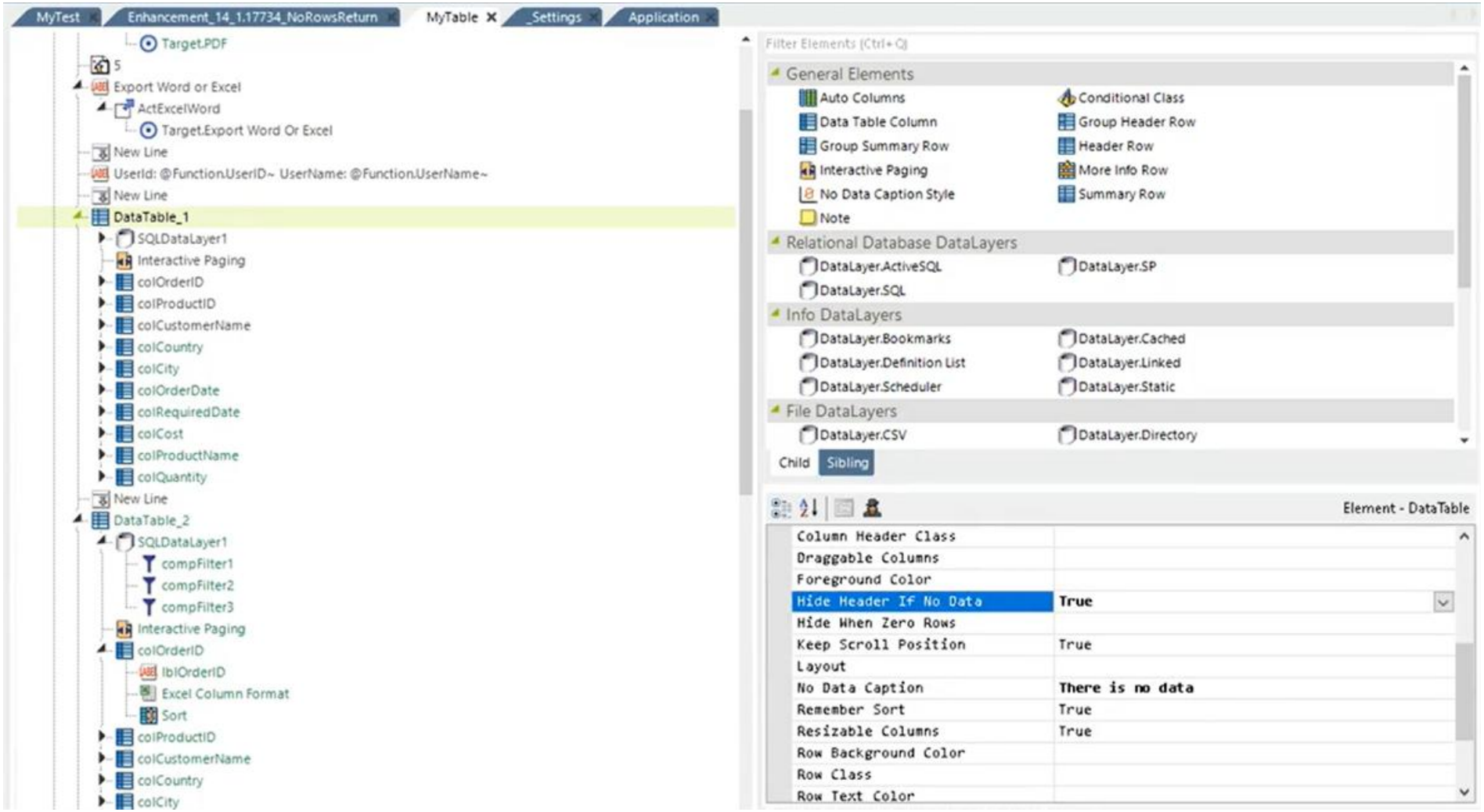


Another example, shown above, includes a column with a check box, for selecting rows, a common implementation in Data Tables. The Extra Column Header in this case includes two links that refresh the table, checking or unchecking all rows. The **Link**

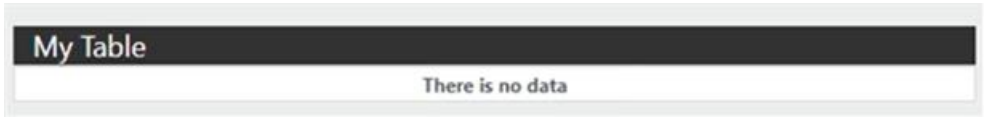
Parameters elements pass either a "1" (All) or nothing (None) when the report is refreshed, and the Input check box element uses the passed Request variable as its **Default Value** and "1" as its **Checked Value**.

Hiding Table Headers

Control whether your message includes table headers on an individual Data Table when there is no data to display by setting the new attribute 'Hide Header If No Data' to "True":



As a result, your message will look like this:



To control the header display for all Data Tables on an application level, set the InfoGo constant '`rdHideData-tableHeaderIfNoData`' to "True". By default, this constant is "False". For more information, see Configuring InfoGo Constants.

- For managed Data Tables/Crosstabs, the 'Hide Header If No Data' attribute overrides the '`rdHideData-tableHeaderIfNoData`' constant, while AG/SSRM Data Tables/Crosstabs take the constant value.
- By default, the values for both the attribute and the constant is "False" and "Default", respectively.

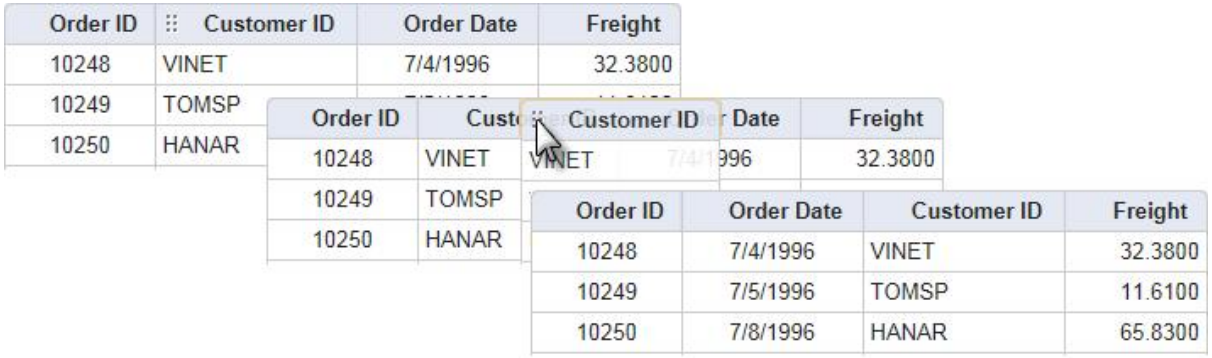
Justifying Column Header Text

The Data Table Column element has a **Class** attribute, which affects both the *data* displayed in the column and the *column header text*. By default, column header text is centered, which may not suit you. The element's **Column Header Class**, allows you to justify the header text separately from the data.

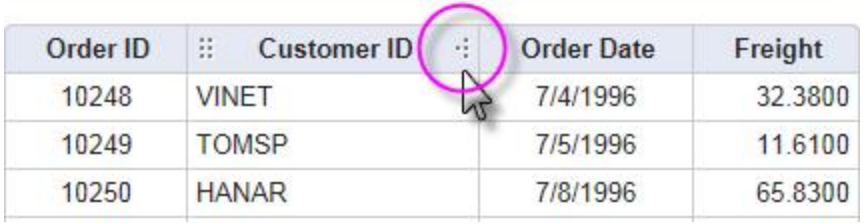
The effectiveness of other style descriptors in these style classes, such as font-size or -weight, may vary depending on any other style settings that have been made for the table, its rows, and cells, including those set by Themes.

Rearranging and Resizing Columns at Runtime

If a Data Table element's **Draggable Columns** attribute is set to *True*, users can rearrange its columns at runtime:



When this feature is enabled, as shown above, users can click on the left side of the column header and drag the column right or left to a new position. If columns can be dragged they'll display a special icon in their headers, as shown above, which isn't visible until the cursor is hovered over it. The new column arrangement can now be stored in bookmarks and "remembered".



The Data Table element's **Resizable Columns** attribute allows users to drag an icon in the column header, shown above, at runtime to increase or decrease the column width.

The default value for both of these attributes is *False*.

Column dragging and resizing functions can't be used when the Data Table includes optional special rows, such as Header rows or Summary rows, that include columns that span multiple Data Table columns.

Sorting Displayed Column Data

Developers often need to present Data Tables with their data sorted in a specific order and this can be done by sorting the datalayer data. You can also give users the ability to dynamically sort columns at runtime simply by adding a child **Sort** element beneath any Data Table Column element. This causes the column header text to become a link that users can click to sort the table based on that column.

Column data is sorted in the order specified in the **First Sort Sequence** attribute when the user clicks the link; clicking it again reverses the sort order. This defaults to *Ascending* first, then *Descending*.

To do this:	Set the following element attributes:
Create an initial sort order	Set the Sort element's **First Sort Sequence** attribute to Ascending or Descending. Default: *Ascending*
Specify a sort order for multiple columns	Set the Sort element's **Data Column** attribute to a comma-separated list of data column names, then set the **Data Type** attribute to a comma-separated list of data types that correspond with the respective column names.
Remember the user's sort selections	Set the parent Data Table element's **Remember Sort** attribute to *True.* Default: *False*
Display arrows in the header that indicate the sort order	Set the parent Data Table element's **Sort Arrows** attribute to *True*. Default: *False*

To do this:	Set the following element attributes:
Sort the data and update the table, without reloading the entire page	Set the parent Data Table element's **AJAX Paging and Sorting** attribute to *True*. AJAX is a technology that allows targeted portions of web pages to be updated, rather than the entire page. With this attribute enabled, when the user sorts a column by clicking its header, only the table portion of the web page will be updated, preventing the page from "flickering" or losing its scroll position. You should be aware that this alters the behavior of the browser's "Back" button because the page history is not updated when AJAX is used. If AJAX Paging and Sorting is enabled and sorting is initiated and there is more than a very brief delay, a spinning "Wait" icon will automatically be displayed to let the user know that processing is occurring.

Showing/Hiding Columns

The Data Table Column element's **Condition** attribute can be used to dynamically *show* or *hide* the column at runtime, and you may want to do this based on the table's data. However, there's an important restriction on this attribute: an expression used as its value *cannot* include @Data tokens. This is because any tokens used here must be available to be resolved when the table structure is being built but, because the table's datalayer doesn't run until later, there are no @Data tokens available at that time.

One solution is to use Local Data, then @Local tokens would be available when the table structure is being built and could be used in the Condition attribute. Other types of tokens, of course, can be used in Condition attribute expressions.

 We *do not* recommend using **Show Modes** for this purpose. The attribute was removed in v12.1.188 but you may encounter it in earlier versions. Use the Condition attribute instead.

Setting Dynamic Column Text and Background Colors

You can set dynamic Data Table column background and text colors using two attributes, **Background Color** and **Text Color**, which are provided for this purpose. For information about how to do this, see *The Color Range Column*.

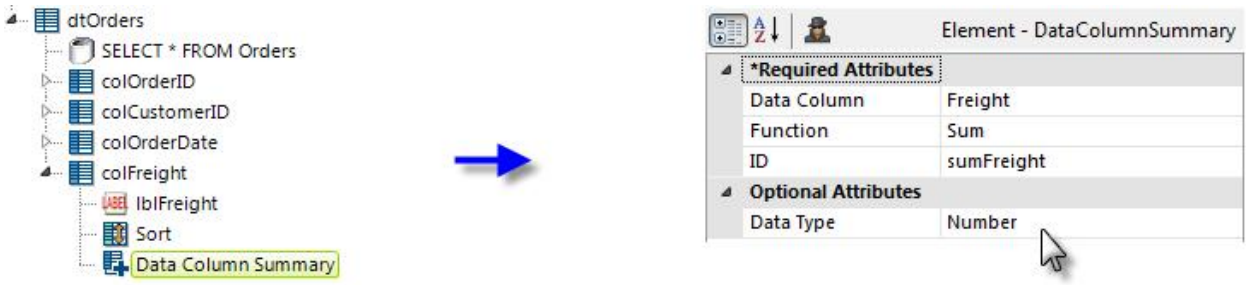
Summarizing Column Data

You can perform typical *aggregations* of column data to create summary information for Data Tables. The following aggregations are available when summarizing column data:

- Average
- AverageOfAllRows
- Count
- CountOfAllRows
- DistinctCount
- Max
- Median
- Min
- Mode
- StdDev
- Sum

By default, aggregations *exclude* columns with Null values. Create the constant `rdCalculationsIncludeNulls` in your _Settings definition and set it to *True* if you want to include them.

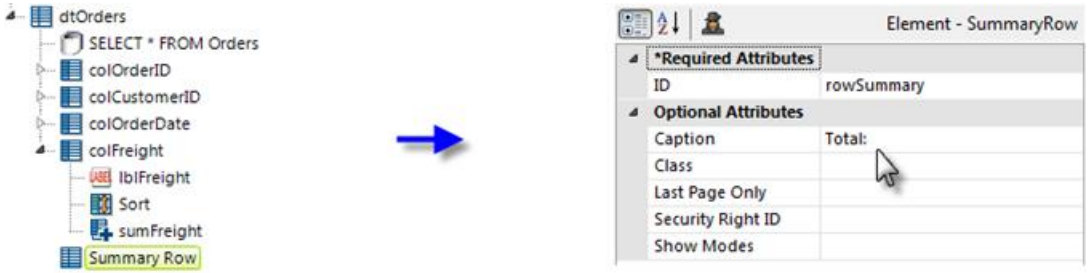
You can display the summary information at the bottom of a Data Table in a dedicated row, on every page or just the last page (if the data spans multiple pages).



To summarize a data column:

1. Add a **Data Column Summary** element beneath a Data Table Column element, as shown above.
2. Set its **Data Column** attribute value to the name of the column to be summarized.
3. Select an aggregating function in the **Function** attribute value.
4. Give the element an **ID** - this is *important* as the ID may be used later in order to reference the summarized value.
5. Set the **Data Type** attribute value to the proper data type (optional but recommended).

You can add a Data Column Summary element to every Data Table column that you want to summarize. As mentioned above, the summarized value is available using an @Data token. So, the data in the example above would be available as `@Data.sumFreight~`. However, you do not need to use this token if you use a **Summary Row** element to display your summarized data.



As shown above, the **Summary Row** element is added as a child of the Data Table (its position in the element tree does not matter) and its attributes are set as shown. This will cause a separate row to appear at the end of the report and its **Caption** will appear in the first column; the summarized values for each column that has a child Summary element will automatically appear in this row.

Order ID	Customer ID	Order Date	Freight
10248	VINET	7/4/1996	$32.38
10249	TOMSP	7/5/1996	$11.61
10250	HANAR	7/8/1996	$65.83
10251	VICTE	7/8/1996	$41.34
10252	SUPRD	7/9/1996	$51.30
Total:			$202.46

The example above shows the summary row and summarized Freight column value.

The summary row will inherit the *format* of its data column. An alternative to using data column summaries is to use datalayer aggregate columns. An aggregate column behaves like any other column in the datalayer, and developers can perform additional calculations using more datalayer column elements.

Using Event Handlers with Column Data

It's possible to add an **Event Handler** element beneath a Data Table Column element. This allows you, for example, to capture mouse-related events associated with table columns themselves and dynamically react to them. For instance, a More Info Row might be made visible or an image or a pop-up panel or a menu might be displayed when the cursor is hovered over a particular table column. For more information on More Info Rows, see "Data Table Rows" on the next page.

Data Table Rows

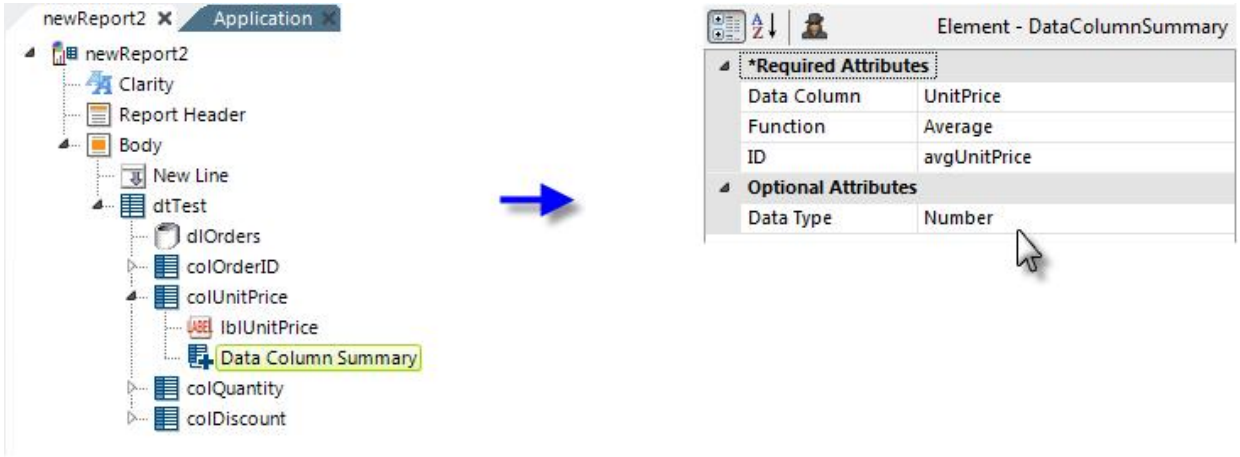
The following topics introduce several Data Table row-related elements:

- Creating Header and Summary Rows
- Highlighting Rows with CSS
- Working with "More Info Rows"

Creating Header and Summary Rows

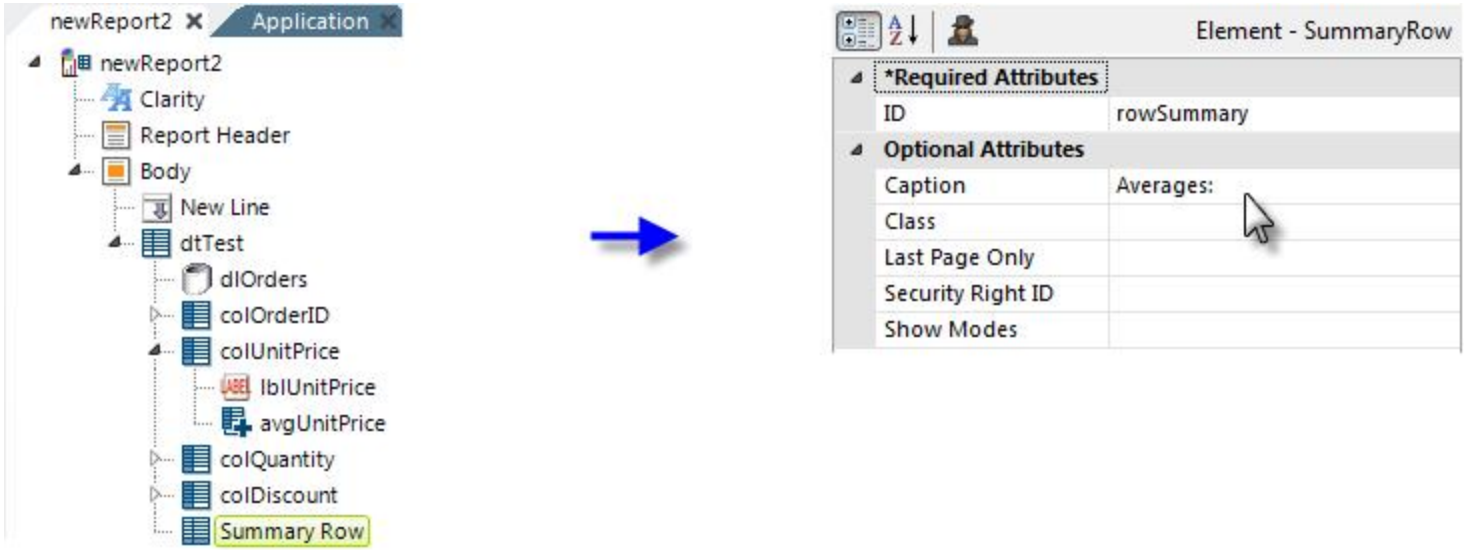
Header and **Summary** rows allow you to display data column summary information on the first (excluding the Column Header row) and/or last rows of a Data Table. You can customize these rows by adding individual **Column Cell** elements.

The easiest way to display summary information is to use special summary elements with each Data Table Column to be summarized and then create a header and/or summary row, as follows:



1. Ensure that each Data Table Column to be summarized has a child **Data Column Summary** element, as shown above.

2. Set its attributes to produce the desired summary of the desired column, as shown above.



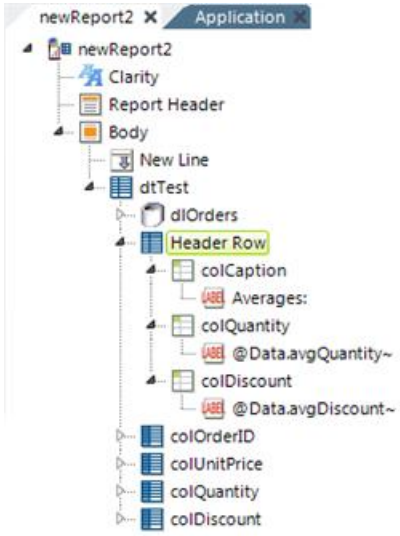
3. Select the parent Data Table element and add a **Header Row** or **Summary Row** element to the definition, as shown above.
4. Set its **Caption** attribute to display a title in the first column of the header/summary row.
5. When using a Summary Row and Interactive Paging, set its **Last Page Only** attribute to *True* to show the summary only on the last page of the Data Table, otherwise it will appear at the bottom of every page.

OrderID	UnitPrice	Quantity	Discount
10248	$14.00	12	0
10249	$18.60	9	0
10250	$7.70	10	0
10251	$16.80	6	0.05
10252	$64.80	40	0.05
Averages:	$24.38	15.4	0.02

The example above show the typical result, with the summary row at the bottom.

The cells generated for the summary row automatically inherit the formatting and alignment of the columns above them.

A standard header or summary row works well when *every* data column contains summary information. However, for Data Tables with numerous columns and/or columns with non-numeric data, a *customized* header or summary row may be a better solution. Here's how to create a customized header:



OrderID	UnitPrice	Quantity	Discount
Averages:		15.4	0.02
10248	$14.00	12	0
10249	$18.60	9	0
10250	$7.70	10	0
10251	$16.80	6	0.05
10252	$64.80	40	0.05

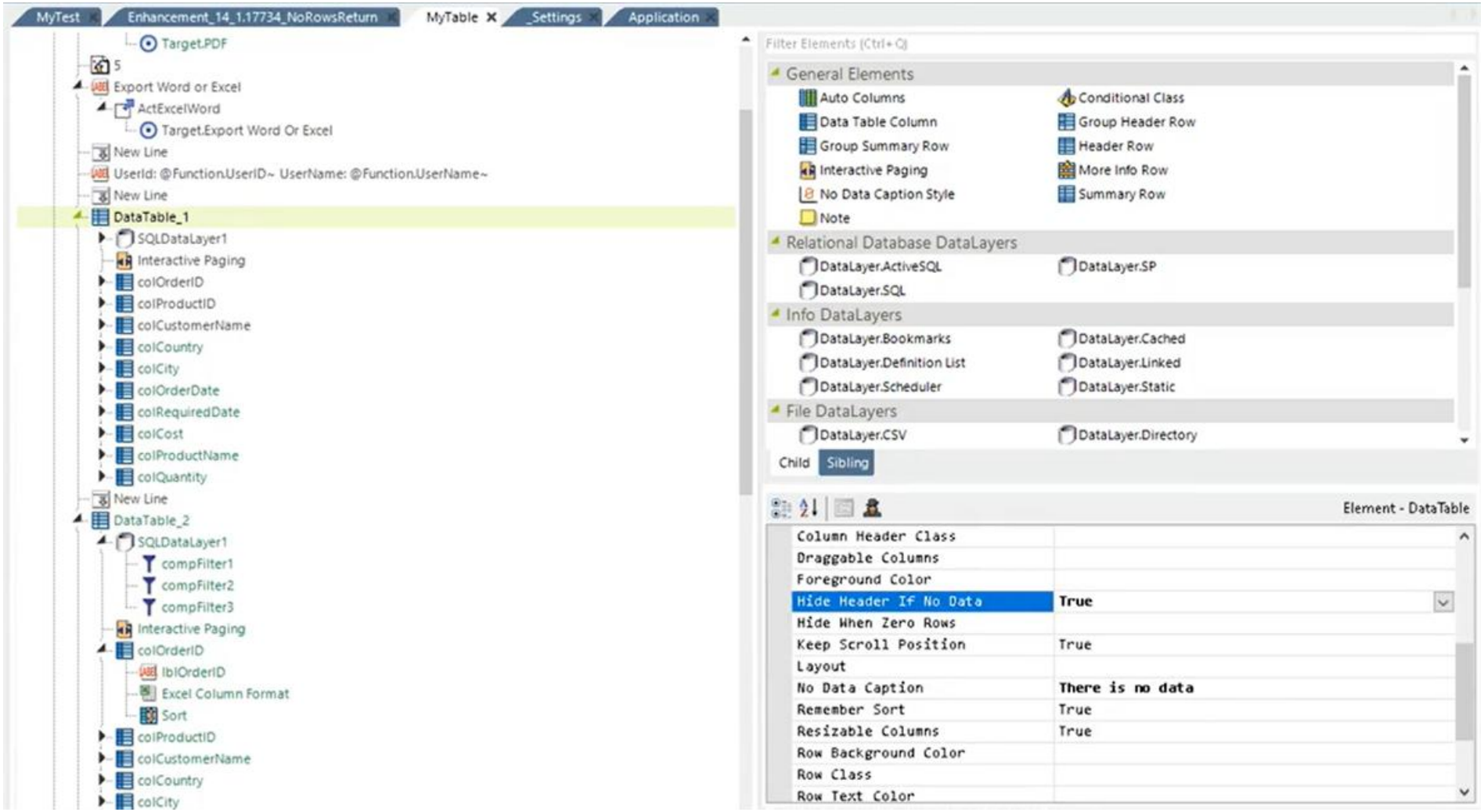
1. Ensure that each Data Table Column to be summarized has a child **Data Column Summary** element.
2. Select the parent Data Table element and add a **Header Row** element to the definition.
3. Beneath it, add one or more **Column Cell** elements.
4. Add **Label** elements beneath the Column Cell elements to display the summary data and use the @Data token to retrieve the summarized values, using on the ID of the Data Column Summary elements you added. For example, @Data.avgDiscount~.

The formatting and alignment of the Data Table columns is *not* inherited when using Column Cell elements, so you must set these individually.

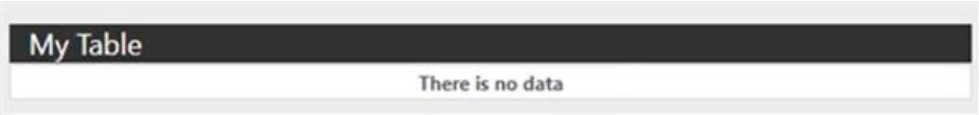
In the example shown above, the Header Row element has *three* Column Cell child elements, but the Data Table has *four* columns. This is handled by setting the first Column Cell element's **Column Span** attribute to span *2* columns, allowing for the difference.

Hiding Table Headers

Control whether your message includes table headers on an individual Data Table when there is no data to display by setting the new attribute 'Hide Header If No Data' to "True":



As a result, your message will look like this:



To control the header display for all Data Tables on an application level, set the InfoGo constant '`rdHideData-tableHeaderIfNoData`' to "True". By default, this constant is "False". For more information, see Configuring InfoGo Constants.

- For managed Data Tables/Crosstabs, the 'Hide Header If No Data' attribute overrides the '`rdHideData-tableHeaderIfNoData`' constant, while AG/SSRM Data Tables/Crosstabs take the constant value.
- By default, the values for both the attribute and the constant is "False" and "Default", respectively.

Highlighting Rows with CSS

In order to increase the "readability" of tables, especially wide tables, it's often useful to be able to highlight an entire row when the cursor is hovering over it.

OrderID	CustomerID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName
10248	VINET	7/4/2010 12:00:00 AM	8/1/2010 12:00:00 AM	7/16/2010 12:00:00 AM	3	32.38	Vins et alcools Chevalier
10249	TOMSP	7/5/2010 12:00:00 AM	8/16/2010 12:00:00 AM	7/10/2010 12:00:00 AM	1	11.61	Toms Spezialitäten
10250	HANAR	7/8/2010 12:00:00 AM	8/5/2010 12:00:00 AM	7/12/2010 12:00:00 AM	2	65.83	Hanari Carnes
10251	VICTE	7/8/2010 12:00:00 AM	8/5/2010 12:00:00 AM	7/15/2010 12:00:00 AM	1	41.34	Victuailles en stock
10252	SUPRD	7/9/2010 12:00:00 AM	8/6/2010 12:00:00 AM	7/11/2010 12:00:00 AM	2	51.3	Suprêmes délices

This effect is shown in the example above and can be created using a simple style class. The CSS used to create the effect is:

```
#yourDataTableID TR:hover TD { background-color: AliceBlue; }
```

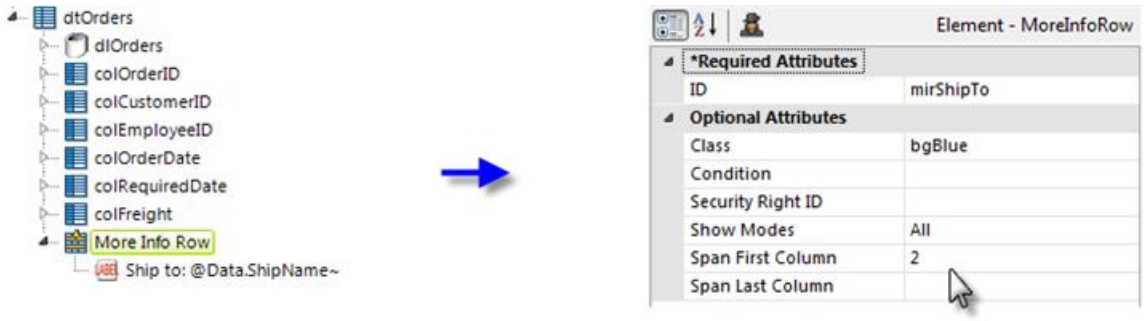
This may not work if your application doesn't have a *Doctype Declarations* set (Logi apps default to the HTML5 doctype) or with some older browser versions.

Working with "More Info Rows"

The **More Info Row** element provides a mechanism that allows detail information to appear beneath each Data Table row. This detail information can be visible all the time, or can be shown or hidden by clicking a link in one of the Data Table columns. It provides developers with an opportunity to make Data Tables less cluttered and more flexible.

Order ID	Cust ID	Emp ID	Order Date	Required Date	Freight
10248	VINET	5	7/4/2010	8/1/2010	$32.38
	Ship to: Vins et alcools Chevalier, Reims, France				
10249	TOMSP	6	7/5/2010	8/16/2010	$11.61
	Ship to: Toms Spezialitäten, Münster, Germany				
10250	HANAR	4	7/8/2010	8/5/2010	$65.83
	Ship to: Hanari Carnes, Rio de Janeiro, Brazil				
10251	VICTE	3	7/8/2010	8/5/2010	$41.34
	Ship to: Victuailles en stock, Lyon, France				
10252	SUPRD	4	7/9/2010	8/6/2010	$51.30
	Ship to: Suprêmes délices, Charleroi, Belgium				

In the example shown above, the "Ship to:" detail data is displayed beneath each data row using a More Info Row element. The font size, font color, number of regular Data Table columns spanned, background color, etc. can be set to differentiate the detail data, if desired. Here's an example of how to implement More Info Rows:



1. Beneath the parent Data Table element, add a **More Info Row** element, as shown above.
2. Set its **Show Modes** attribute value to *All*. This causes the rows to be shown at all times.
3. Set its **Span First Column** attribute value to *2*, which causes the its data to appear beginning underneath the second Data Table column (leaving this attribute and the Span Last Column attribute *blank* causes the data to span the entire Data Table).
4. Add elements beneath the More Info Row element to display the desired text and data (data from the table's datalayer is available here using @Data tokens).

The More Info Row element has a **Condition** attribute, which allows you to conditionally include the element, for greater control over when the element can be seen, primarily when the More Info Rows are displayed by default.

Showing/Hiding More Info Rows

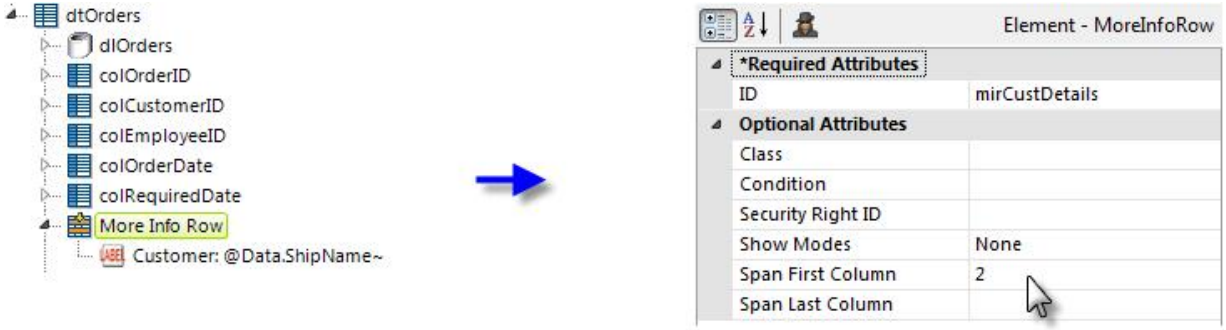
In the next example, the More Info Rows are *hidden* from view until the user takes some action to display them.

Order ID	Cust ID	Emp ID	Order Date	Required Date
10248	VINET	5	7/4/2010	8/1/2010
10249	TOMSP	6	7/5/2010	8/16/2010
10250	HANAR	4	7/8/2010	8/5/2010
10251	VICTE	3	7/8/2010	8/5/2010
10252	SUPRD	4	7/9/2010	8/6/2010

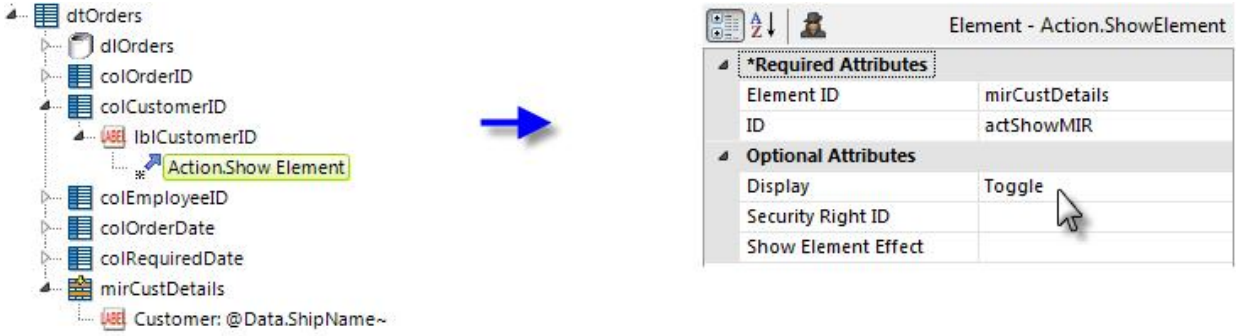
Order ID	Cust ID	Emp ID	Order Date	Required Date
10248	VINET	5	7/4/2010	8/1/2010
10249	TOMSP	6	7/5/2010	8/16/2010
	Customer: Toms Spezialitäten, Münster, Germany			
10250	HANAR	4	7/8/2010	8/5/2010
10251	VICTE	3	7/8/2010	8/5/2010
10252	SUPRD	4	7/9/2010	8/6/2010

When the user clicks on a customer ID link, the table *expands* downward to display a More Info Row that displays detail information about the customer. The new row is added right beneath the row that contains the clicked link. Depending on the element's configuration, clicking the link again will hide the new row.

The information in the More Info row can be text, as shown above, or an image, another table, or even a SubReport. Here are the key elements and attributes used to create the example above:



1. Beneath the parent Data Table element, add a **More Info Row** element, as shown above.
2. Set its **ID** to a unique value and its **Show Modes** attribute value to *None*, initially hiding all the detail rows.
3. Add elements beneath the More Info Row element to display the desired text and data (data from the table's datalayer is available here using @Data tokens).



4. Identify the data that you want to be the link that shows and hides your More Info row. Beneath it, add an **Action.Show Element** element, as shown above.
5. Set its **Element ID** attribute value to the ID of the More Info Row element you added earlier.
6. Set its **ID** attribute to a unique value and its **Display** attribute to *Toggle*.

Now run the application and click a Customer ID in the table to display the detail information. Click it again to hide it.

It is possible to "nest" More Info Row elements, so that you have one More Info Row element as a child of a table that's a child of another More Info Row. However, be aware that the lowest More Info Row contents may not export correctly, or at all, so we don't recommend this design arrangement if exporting of expanded rows is desired.

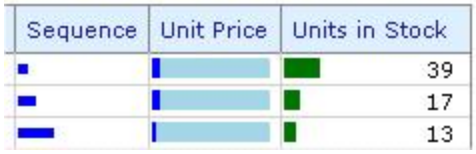
The **More Info Row Column** element can be used to closely align More Info Row columns with the columns of its parent table. When this element is used, the More Info Row element's **Span First Column** and **Span Last Column** attributes are left *blank*, and the More Info Row Column element's **Column Span** attribute is used instead to control spanning of parent table columns.

Important information about using SubReports in a More Info Row can be found in *Working with SubReports*.

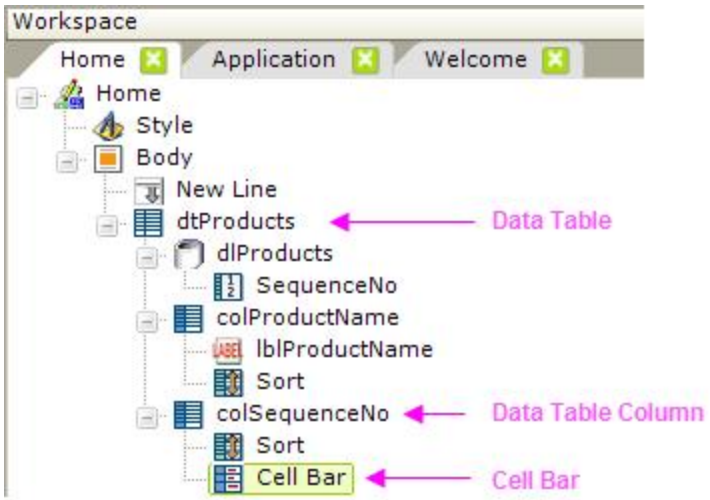
Cell Bar Gauges

This topic introduces developers to the **Cell BarGauge** element. As a child element of the Data Table Column, it provides a new method of depicting data values using color.

The Cell Bar Gauge element creates an **in-cell, horizontal bar gauge** which allows easy visual comparison of relative values in a single column. It looks like this:

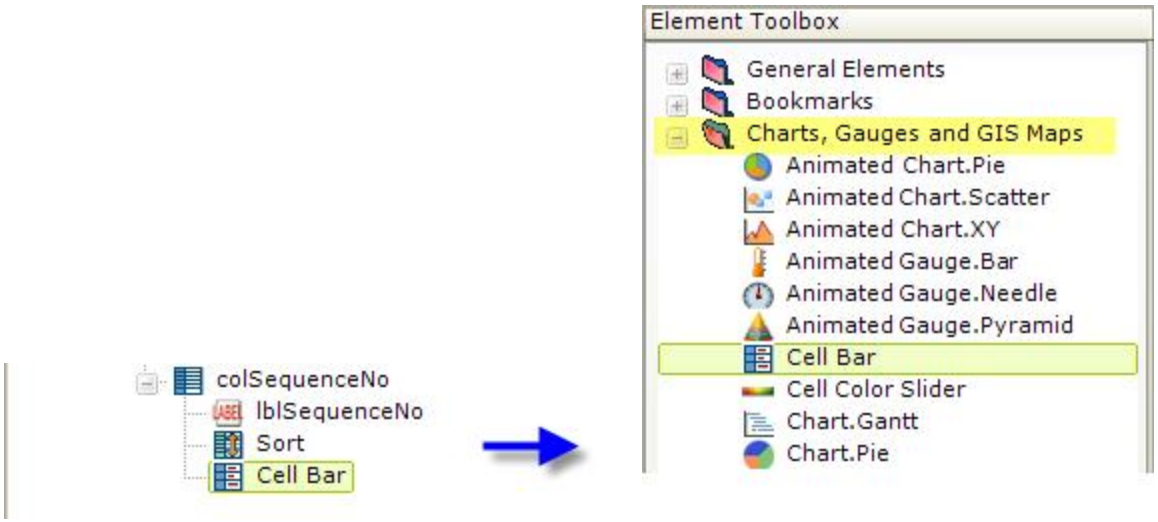


The example above shows part of a table with three data columns using the Cell Bar Gauge element. The first column, Sequence, is configured to display a plain horizontal bar who's length is based on the column's data value. The second column, Unit Price, is configured similarly but adds a background color for greater visual contrast. And the last column is configured to display both the bar and a label with the actual data value. Here are the report definition elements for the first part of the above example:



As shown above, the Cell Bar Gauge element is the child of a **Data Table Column** element, which is in turn the child of the **Data Table** element. The column with the Cell Bar Gauge can also use a Label element to display the actual data value along with the Cell Bar Gauge, or not, as desired.

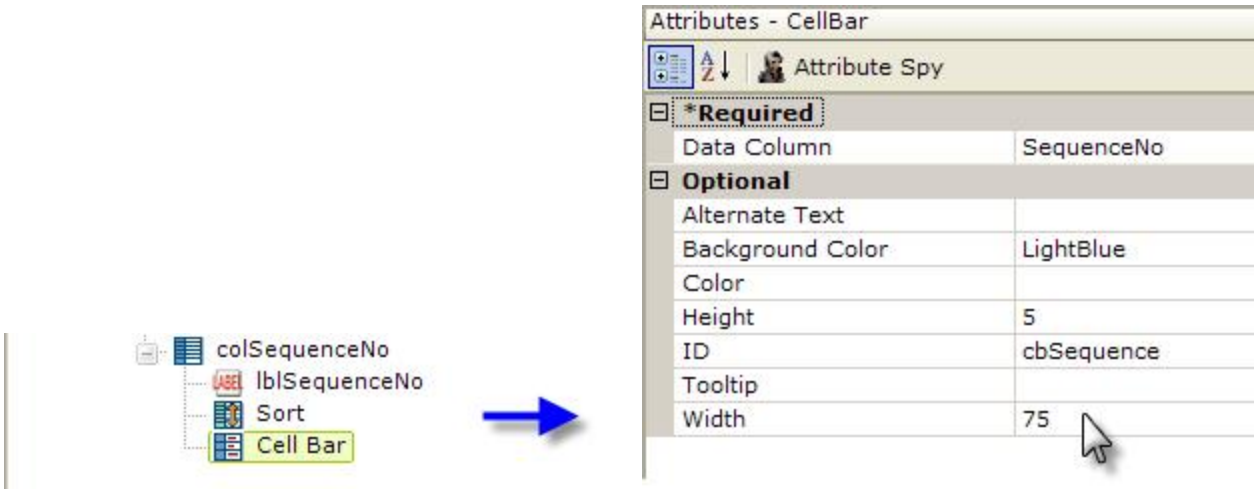
You *cannot* use the Cell Bar Gauge element in a SubData Table.



When a Data Table Column is selected, the Cell Bar Gauge child element can be found in the **Charts and Gauges** folder in the Suggestions panel, as shown above.

Setting Cell Bar Attributes

The attributes associated with the Cell Bar Gauge element in our example are shown below:



- (Required) The **Data Column** attribute has been set to the *name* of the *data column* retrieved in your Data Layer element (the actual column name, not the @Data.ColumnName~ token).
- The **Alternate Text** attribute is the text to be displayed if the bar gauge can't be rendered for some reason.
- The **Background Color** attribute sets a color for the background of the bar Here it's been set to *LightBlue*.
- The **Color** attribute sets the color for the part of the bar representing the data. The default color is *Blue*.
- **Height** and **Width** determine the physical size of the bar - it can be smaller than the cell itself; allowing you to put both a Cell Bar Gauge and a Label in a column.
- **ID** and **Tooltip** are the usual identifier and "hover text" attributes.

What About the Data?

You don't need to do anything other than identify the data column. The Cell Bar Gauge element automatically evaluates all of the values for its column and adjusts the graphics segments that make up the bar accordingly, sizing them to provide a meaningful visual range of bar lengths.

Product Name	Sequence No	Unit Price	Units In Stock
Chai			11
Chang			17
Aniseed Syrup			13
Chef Anton's Cajun Seasoning			53
Chef Anton's Gumbo Mix			0
Grandma's Boysenberry Spread			120
Uncle Bob's Organic Dried Pears			15
Northwoods Cranberry Sauce			6
Mishi Kobe Niku			29
Ikura			31

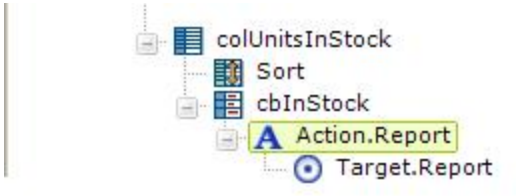
A example of a Data Table using Cell Bar Gauges is shown above.

The arrangement in the Units In Stock column was achieved by using both Cell Bar Gauge and Label elements, separated by a Space element, beneath the Data Table Column element. Set the gauge's Background Color attribute (to white, in the example) to get the data values to align on the right side of the column.

If you include and configure a Sort element in your Data Table Column, the Cell Bar Gauge will sort when the column header is clicked, just like any data value.

Cell Bars Can Be Links

Cell Bar Gauge elements, like Labels, can be **links** to other reports or URLs.

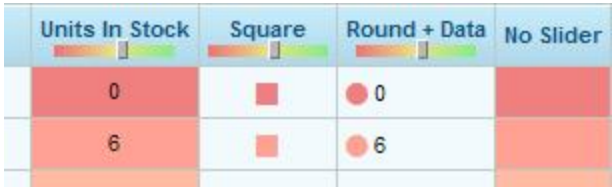


As shown above, the Cell Bar Gauge element can be a parent to any of the Action elements.

Cell Color Sliders

This topic introduces developers to the **Cell Color Slider** element. As a child element of the Data Table Column, it provides a new methods of depicting data values using color.

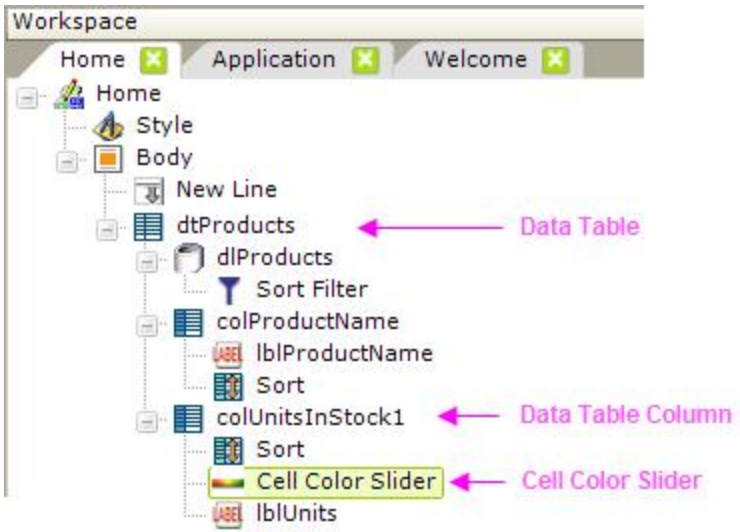
The **Cell Color Slider** element is similar to the Cell Bar Gauge element in that it generates a graphic depiction of the data within its Data Table column.



As shown above, the Cell Color Slider element shows a column's numeric values with a colored indicator or background. The color for a particular column varies depending on where its value fits into the overall population of values for that column in the table. Low values are one color, middle values a second, and high values, a third color; everything in between is given shades of those colors. The Cell Color Slider element makes it easy to visualize how values in a column relate to each other.

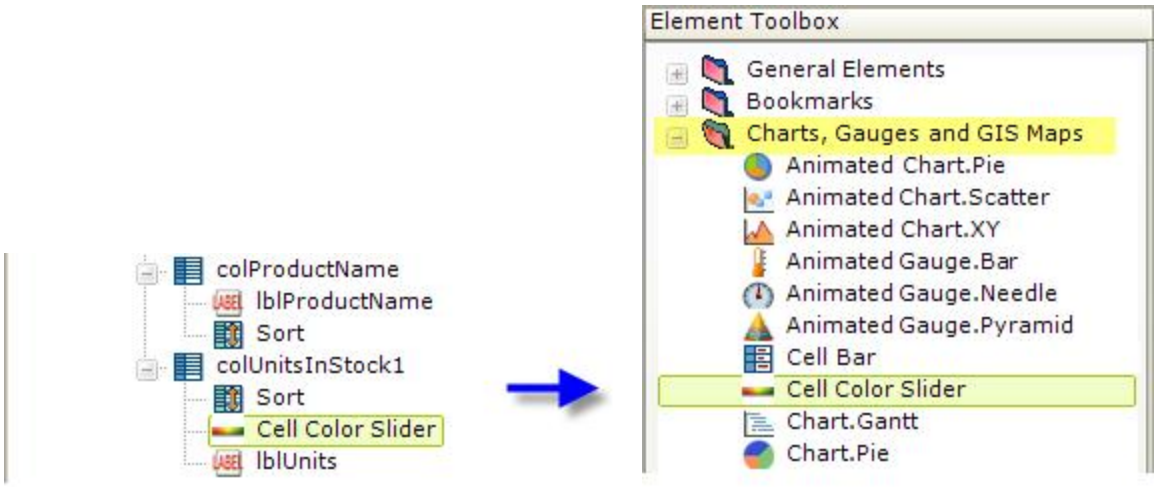
At runtime, the user can manipulate an optional slider control at the top of the column to adjust the "middle value" color and the color indicator can be a square, a circle or the entire cell background.

Cell Color Slider elements should *not* be used in reports that will be **exported**, for example, to PDF or Excel formats.



The report definition elements for the first part of the earlier example are shown above. The Cell Color Slider element is the child of a Data Table Column element, which is in turn the child of the Data Table element.

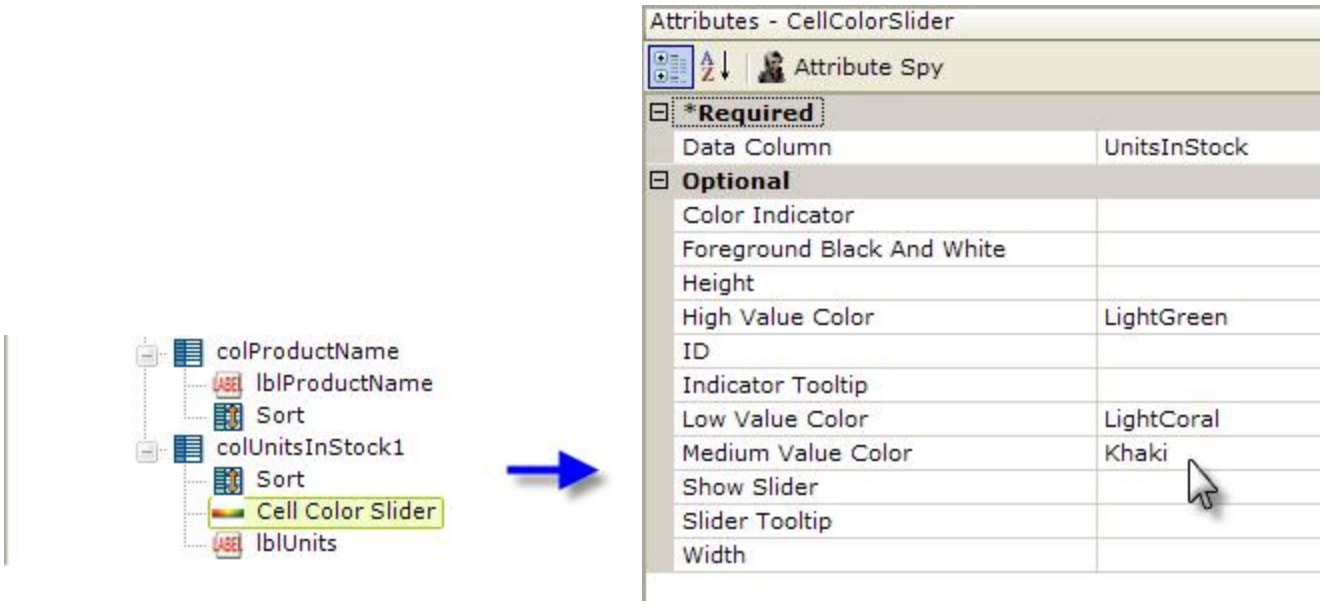
You *cannot* use the Cell Color Slider element in a SubData Table.



When a Data Table Column is selected, the Cell Color Sliderchild element can be found in the **Charts and Gauges** folder in the Suggestions panel, as shown above.

Setting Cell Color Slider Attributes

The attributes associated with the Cell Color Slider element in our example are shown below:



- The **Data Column** attribute has been set to the *name* of the *data column* retrieved in your Data Layer element (the actual column name, not the @Data.ColumnName~ token).
- The **Color Indicator** allows you to choose between Square, Circle, or Background indicators. Default: *Background*
- The **Foreground Black and White** attribute, when set to *True*, automatically sets the foreground color of text in each cell depending on the background color. If the background color is dark, the foreground font is set to white; if the background is light, the font becomes black.
- The **High Value Color** attribute sets the "high" end of the color scale; representing the highest data values. Default: *Green*
- The Middle and Low Value Color attributes set the "middle" and "low end" of the color scale; representing the median and low data values. Defaults: *Yellow* and *Red*
- The **Show Slider** attribute determines whether a slider appears below the column header caption; when visible, the slider can be moved by users at runtime to fine tune the Middle value color.
- **Height** and **Width** determine the physical size of the bar - it can be smaller than the cell itself; allowing you to put both a

Cell Color Slider and a label in a column.

- **ID** and **Slider Tooltip** are the usual identifier and "hover text" attributes.

What About the Data?

You don't need to do anything other than identify the Data Column. The Cell Color Slider element automatically evaluates all of the values for its column and adjusts the colors accordingly.

Product Name	Units In Stock	Square	Round + Data	No Slider
Chef Anton's Gumbo Mix	0		0	
Northwoods Cranberry Sauce	6		6	
Chai	11		11	
Aniseed Syrup	13		13	
Uncle Bob's Organic Dried Pears	15		15	
Chang	17		17	
Mishi Kobe Niku	29		29	
Ikura	31		31	
Chef Anton's Cajun Seasoning	53		53	
Grandma's Boysenberry Spread	120		120	

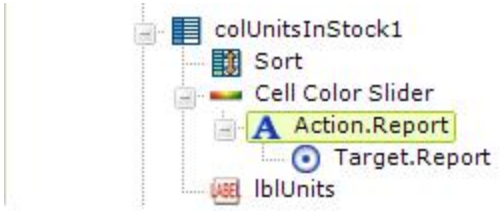
A example of a Data Table using Cell Color Slider is shown above.

The arrangement in the Units In Stock column was achieved by using both a Cell Color Slider and a Label element, and setting the Data Table Column element's **Class** attribute to *ThemeAlignCenter*.

If you include and configure a Sort element in your Data Table Column, the Cell Color Slider will sort when the column header is clicked, just like any data value.

Cell Color Sliders Can Be Links

Cell Color Slider elements, like Labels, can be **links** to other reports or URLs.



As shown above, the Cell Color Slider element can be a parent to any of the Action elements.

Hierarchical Data

Data in **hierarchicalgroups** presents a special challenge for the Logi developer and this topic introduces several different techniques for presenting it using Data Tables.

The following topics discuss Hierarchical Data:

- Using Group Header and Group Summary Rows
- Creating a Hierarchical Data Table with Indented Groups
- Creating Hierarchies by Hiding Duplicate Values
- Working with SubData Tables

See the Data Grouping Sample DevNet for an example related to this subject.

About Hierarchical Data

Hierarchical data is data that is grouped into a tree-like structure, with repeating parent/child relationships:

```
Customer 001
    Order 1
        Item 1
        Item 2
        Item 3
   Order 2
        Item 1
        Item 2

Customer 002
    Order 1
        Item 1
    Order 2
        Item 1
        Item 2
```

For example, the data shown above is in a hierarchy, grouped first by **Customer**, then by **Order**, with Order Item details. In the business intelligence reporting world, this is a very common approach to presenting data and segments the data visually in a way that makes it easy to understand.

The challenge with HTML-based reports, such as Logi applications, is to manage the screen real estate effectively when working with hierarchical data and there are several approaches to doing this. Depending on the number of columns to be displayed, a straight forward Data Table can be used. If that isn't practical, then it may be better to embed a Data Table (either as a "subdata" table or an embedded subreport) within a table to more easily manage the columns; this is discussed in "SubData Tables" on

page 512. And, finally, placing a subData Table or an embedded subreportwithin a More Info Row may help to reduce visual clutter, see "Working with "More Info Rows"" on page 474. The following examples build on the first approach, which uses a regular Data Table.

Restriction when using Input Elements

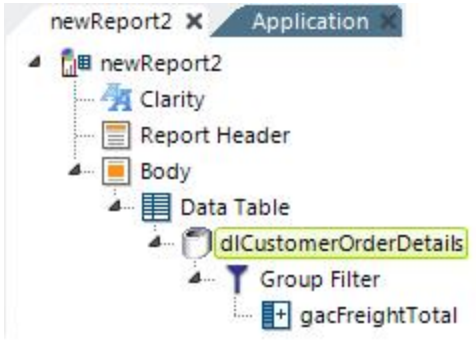
Some applications display data in a Data Table and include Inputelements in each row of the table. A common example is a "Delete" check box for every record. This can be done in Logi applications, but is *not recommended* when hierarchical data is being used. The complex data relationships involved make it impossible to uniquely identify the associated Input elements.

This prohibition applies to obvious hierarchical representations, such as the Data Table, but also to elements like the Data Tree, which use hierarchical grouping internally to create their parent-child data representations.

Using Group Header and Group Summary Rows

One effective and uncomplicated method of displaying hierarchical data in a Data Table is to use **Group Header** and **Group Summary Rows**. As is often the case, the first step is to understand the data and what groupings are required.

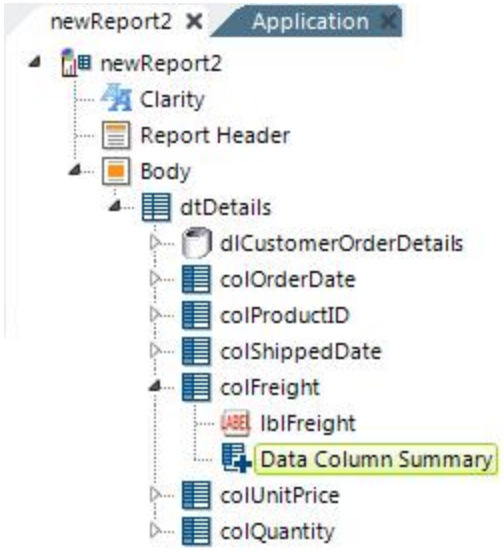
In this example, we'll group the data by Order ID. The Order ID and Customer ID will appear in a group header. We'll also summarize the Freight costs by order and show them as a total for each group in a group summary row.



The image above shows a report definition in which a SQL query has been used to retrieve data from a JOIN of Customer, Orders, OrderDetails, and Products tables. A **Group Filter** element has been added to create a hierarchical arrangement of the data, based on Order ID, and *must* be given an element **ID** ("grpOderID") for this implementation.

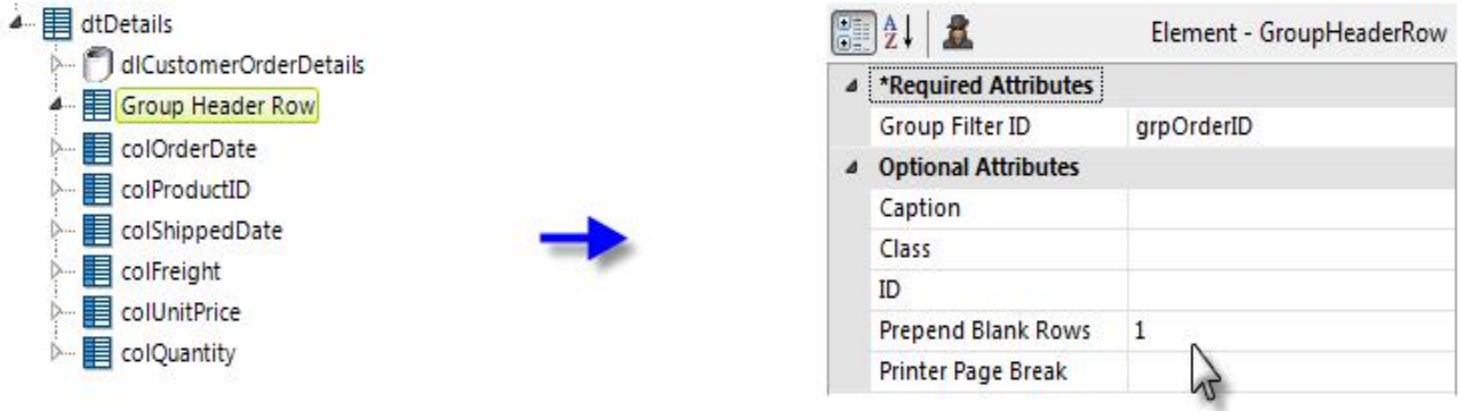
A **Group Aggregate Column** element has been added to create grouped totals of the Freight column.

Group filters create data groups by looking for unique values in a particular column. By default, the first row with a unique value is kept and the duplicate rows are discarded. Developers must choose to keep *all* grouped rows when building and presenting hierarchical data within a Data Table, by setting the Group Filter element's **Keep Grouped Rows** attribute value to *True*.



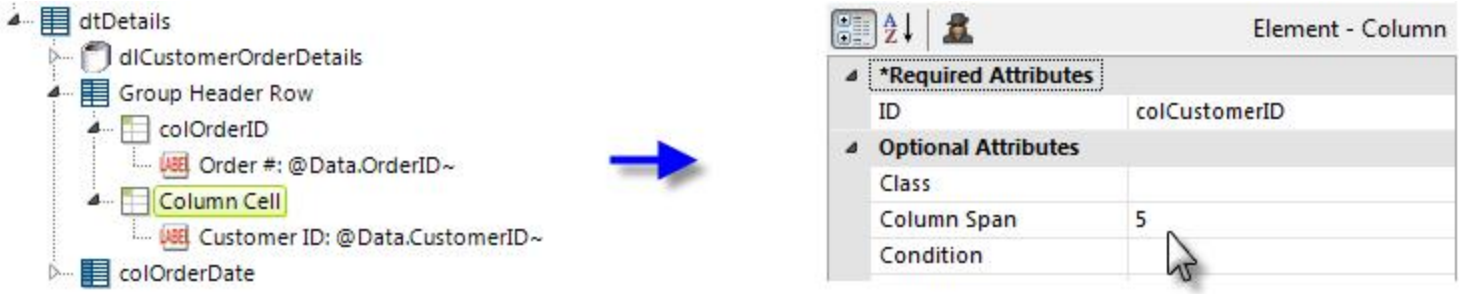
In the next step, as a shown above, **Data Table Column** elements have been added to display the detail data. Each of these columns has a child **Label** element.

In order to summarize the Freight costs, we'll add a **Data Column Summary** element to the definition beneath the Freight column. This element's attributes are then set to *Sum* the Freight column values.

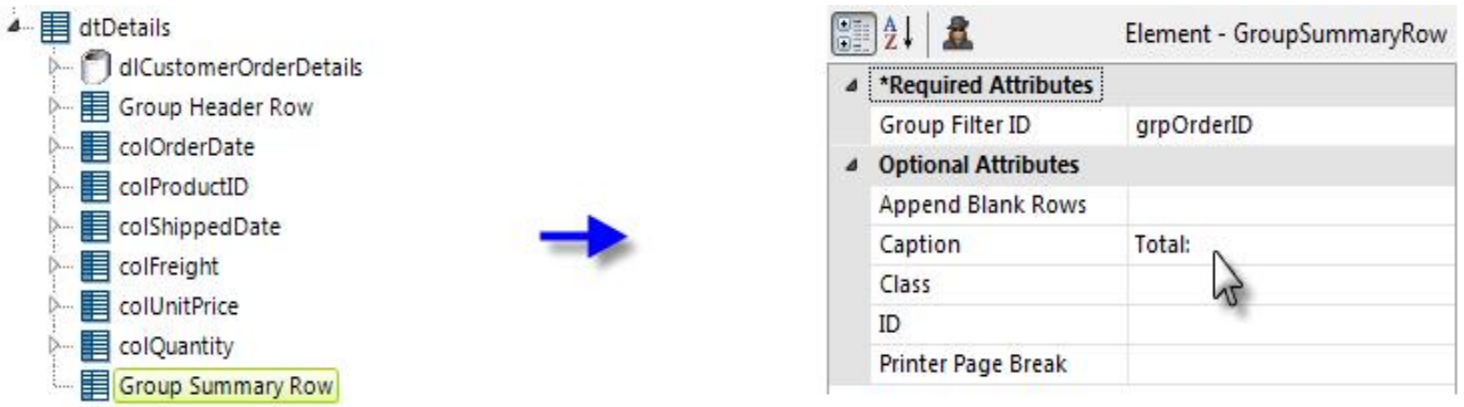


Next, a **Group Header Row** element has been added, as shown above.

The Group Header Row requires that we identify the Group Filter we used earlier (that's why we gave it an element ID), and we'll configure it to insert ("prepend") a blank row before each group except the first one.



Child elements, shown above, are added beneath the Group Header Row, and are configured to show the Order ID and Customer ID. The second Column Cell element's **Column Span** attribute is set to *5,* as there are six total columns. The Label elements are configured to display the data tokens from the appropriate columns.



Finally, a **Group Summary Row** is added, as shown above. It's configured as shown to use the appropriate Group Filter and a Caption is provided. This element will *automatically* display summaries beneath any column that has a Data Column Summary child element and the summary data will inherit the alignment and formatting of its parent column.

Order Date	Product ID	Shipped Date	Freight	Unit Price	Quantity
Order #: 10248	Customer ID: VINET				
7/4/2015	11	7/16/2015	$32.38	14.0000	12
7/4/2015	42	7/16/2015	$32.38	9.8000	10
7/4/2015	72	7/16/2015	$32.38	34.8000	5
Total:			$97.14		
Order #: 10249	Customer ID: TOMSP				
7/5/2015	14	7/10/2015	$11.61	18.6000	9
7/5/2015	51	7/10/2015	$11.61	42.4000	40
Total:			$23.22		
Order #: 10250	Customer ID: HANAR				
7/8/2015	41	7/12/2015	$65.83	7.7000	10
7/8/2015	51	7/12/2015	$65.83	42.4000	35
7/8/2015	65	7/12/2015	$65.83	16.8000	15
Total:			$197.49		

A portion of the resulting Data Table is shown above. Note how the data has been arranged in a hierarchy.

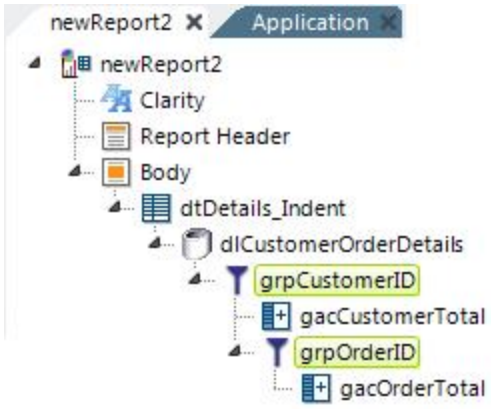
When printing reports directly from the browser or from a PDF export, developers can choose to print each data grouping on separate pages. Set the Group Header Row or Group Summary Row element's **Printer Page Break** attribute to *True* to insert a page break before each group header row.

Group Header Rows and Group Summary Row elements now include a **Show Modes** attribute, allowing you to hide them when exporting to PDF. In addition, if using these two elements with their **Append Blank Rows** or **Prepend Blank Rows** attributes set to a number, blank rows will now *not* be appended/prepended when the report is exported to PDF.

In a Data Table report, when Append Blank Rows to the Group Summary Row is set, the width of the newly added blank rows will be the same as the largest width in the existing Table Row.

Creating a Hierarchical Data Table with Indented Groups

This example produces a report that *indents* the data based on its hierarchical grouping and features two levels of grouping. Once again, the first step is to understand the data and what groupings are required.



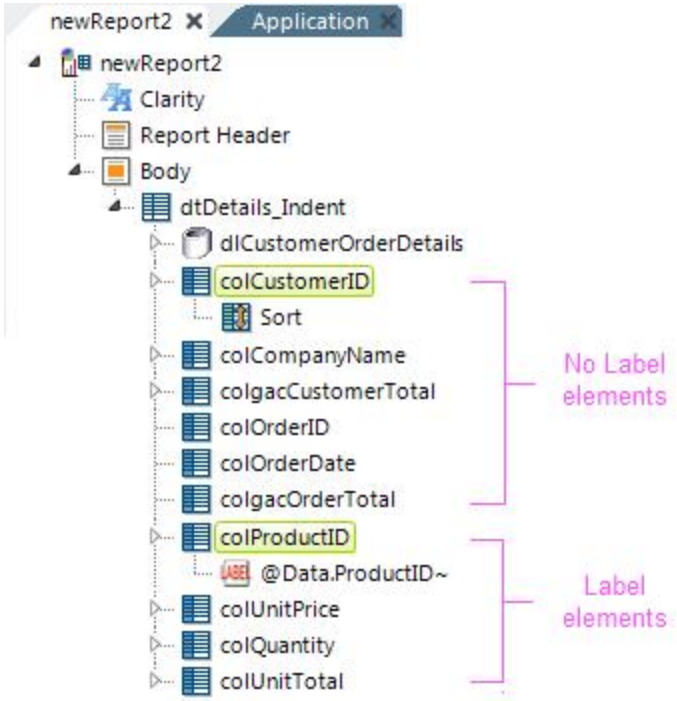
The example above shows a report definition in which a SQL query has been used to retrieve data from a JOIN of the Customer, Orders, OrderDetails, and Products tables. The query includes a calculated column created by multiplying *UnitPrice* by *Quantity* to get an *UnitTotal* value. Here's the T-SQL query used in the example:

```
     Select Orders.OrderID, Orders.CustomerID, Orders.OrderDate, [Order
Details].UnitPrice,
     [Order Details].Quantity, [Order Details].UnitPrice *
[Order Details].Quantity AS UnitTotal,
     Customers.CompanyName, Products.ProductID
     From Orders
     Inner Join [Order Details] On Orders.OrderID = [Order
```

```
Details].OrderID
     Inner Join Customers On Customers.CustomerID =
Orders.CustomerID
     Inner Join Products On Products.ProductID = [Order
Details].ProductID
```

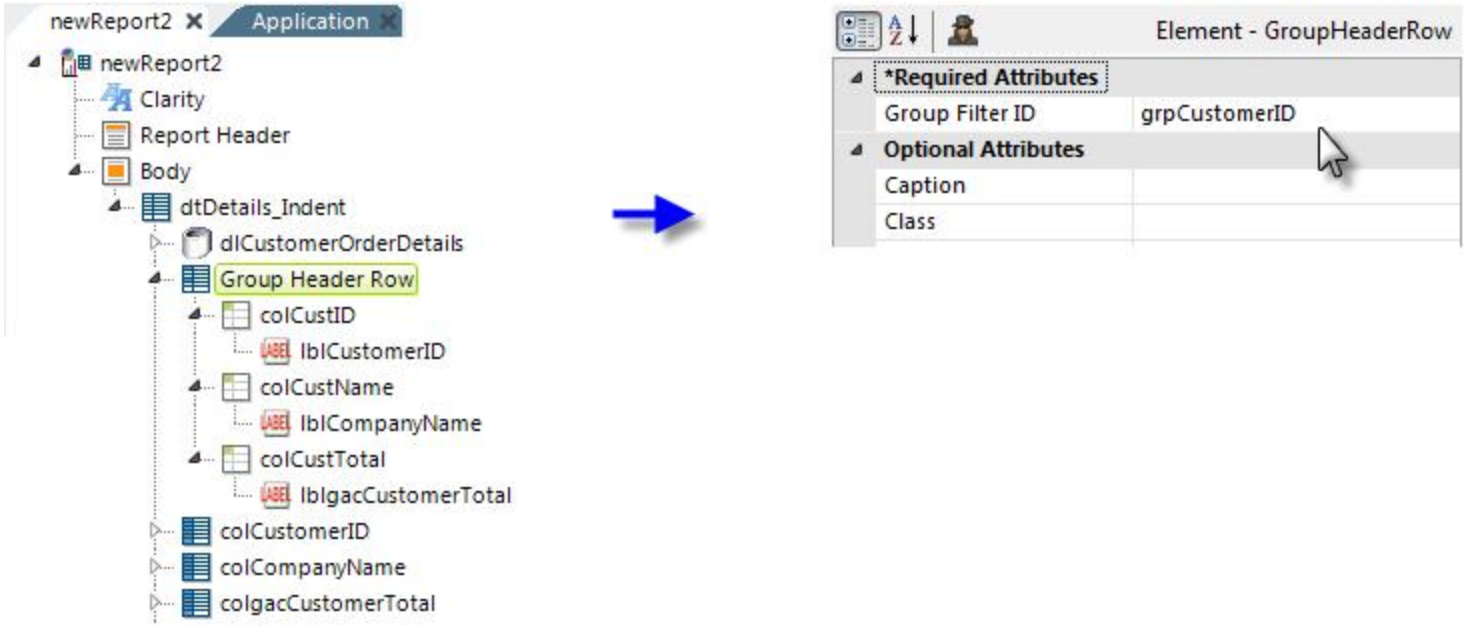
A **Group Filter** element is used to create a grouping of the data by *CustomerID* and to create a group aggregate column that sums the *UnitTotal* values for each customer. Then a second, nested Group Filter is used to group and calculate a sum for each order, again using *UnitTotal* values. Once again, we need to give both Group Filter elements a unique **ID**, which we'll use later, and we want to keep *all* of their grouped rows, by setting their **Keep Grouped Rows** attribute values to *True*.

 Because the second Group Aggregate Column element is nested, *gacOrderTotal* will not appear in Logi Studio's drop-down lists of attribute options or Intelligent Token Completion feature. You'll need to enter it manually.



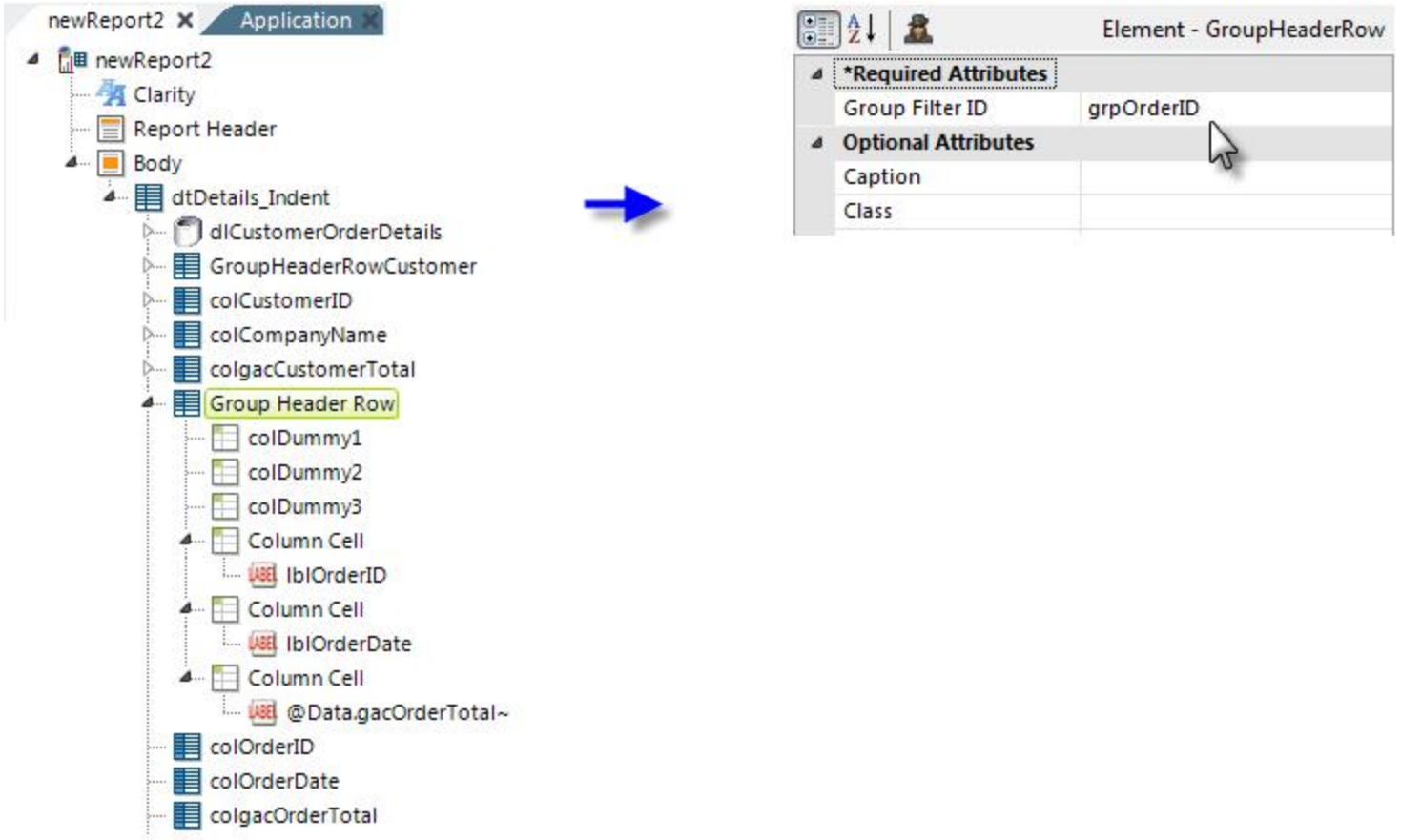
The next step, as shown above, is to provide columns for the data by adding **Data Table Column** elements to the definition. The first six of these columns will be "placeholders" - we *won't* be putting **Label** element in them to show data. Their data will be displayed using a different mechanism described later. They can, optionally, have child **Sort** elements.

For the last four Data Table columns, colProduct ID though colUnitTotal, we *will* add a **Label** element, with an @Data token in its **Caption** attribute, to display their data.



To provide data at the Customer level, a **Group Header Row** element is used, as shown above. Its **Group Filter ID** attribute value is set to the ID of the relevant group filter ("grpCustomer"). Beneath the header row, **Column Cell** and **Label** elements are added to provide the data in the first three columns of the report.

The actual location of the Group Header Row element in the element tree - at the top as shown or after its placeholder column elements - doesn't matter.



The process is repeated for the Order data, by adding another Group Header Row element, as shown above. However, this time three "dummy" Column Cell elements must be added to account for the columns holding the Customer-level data. Then three Column Cells are added to contain the Order-level data.

Cust ID	Company Name	Customer Total	Order ID	Order Date	Order Total	Product ID	Unit Price	Quantity	Unit Total
ALFKI	Alfreds Futterkiste	$4,596.20							
			10643	8/25/2015	$1,086.00				
						28	$45.60	15	$684.00
						39	$18.00	21	$378.00
						46	$12.00	2	$24.00
		2015	10692	10/3/2015	$878.00				
						63	$43.90	20	$878.00
			10702	10/13/2015	$330.00				
						3	$10.00	6	$60.00
						76	$18.00	15	$270.00
ANATR	Trujillo Emparedados	$1,402.95							
			10308	9/18/2015	$88.80				
						69	$28.80	1	$28.80
						70	$12.00	5	$60.00
			10625	8/8/2015	$479.75				
						14	$23.25	3	$69.75
						42	$14.00	5	$70.00
						60	$34.00	10	$340.00
						71	$21.50	20	$430.00

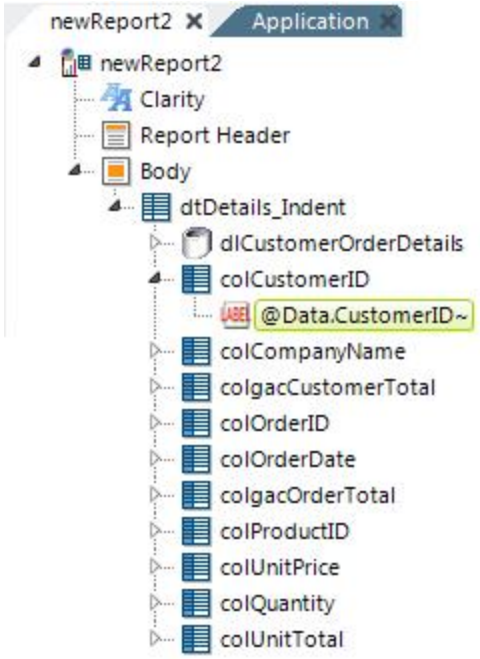
The resulting report looks like the example above. You can see that each level of grouping - Company, Order, and Product - has a different indentation level.

Group Summary Row elements could be used instead of, or in addition to, Group Header Rows in order to place the summarized Order data beneath the Order Details.

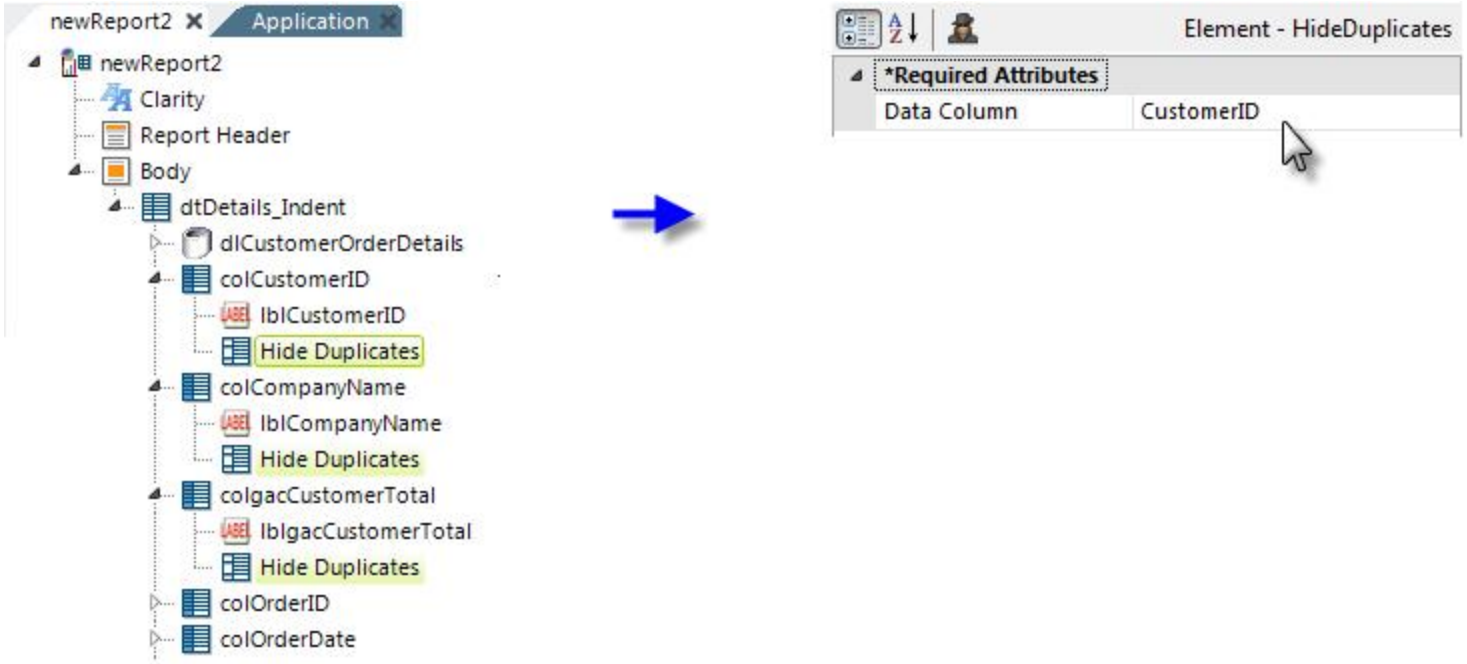
Creating Hierarchies by Hiding Duplicate Values

Use the **Hide Duplicates** element to create a hierarchical layout within a Data Table from one or more data groups.

The Hide Duplicates element can be added beneath any Data Table Column to suppress output when the record in the current row is equal to the record from the previous row. By utilizing the Hide Duplicates element in several columns, developers can achieve the effect of a hierarchical table.

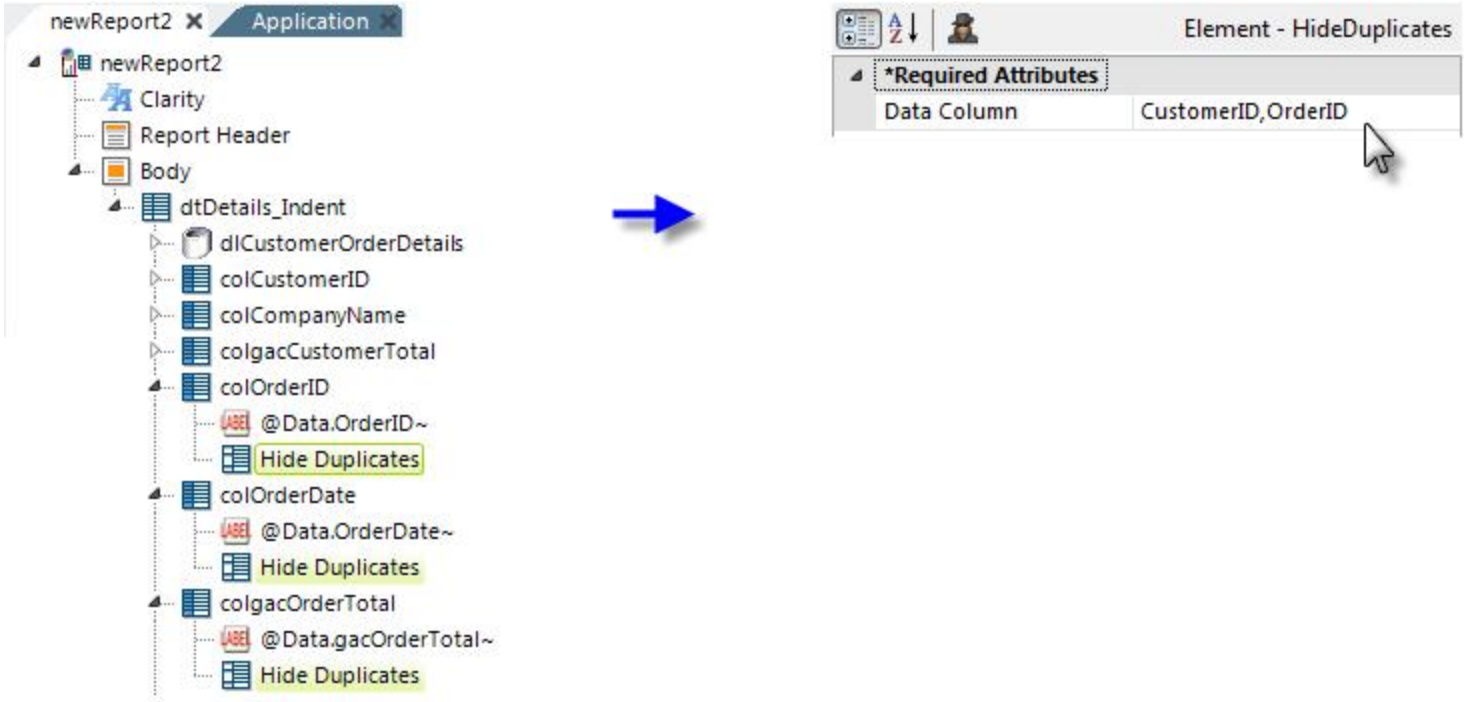


This example starts with the same datalayer used in the previous example, and uses one **Data Table Column** element for each report column, as shown above. Each Data Table Column has a child **Label** element to display the data. No **Group Header Row** or **Group Summary Row** elements, or their child elements, will be used.



Beneath *each* of the first six Data Table Column elements, add a **Hide Duplicates** element. This element has one attribute, **Data Column**, which is the name of the column to check for duplicate values.

In the first three columns (the "Customer" columns), set this value to "CustomerID", as shown above.



In the second set of three columns (the "Order" columns), set Hide Duplicates elements' Data Column attribute to a comma-separated list of "CustomerID,OrderID", as shown above.

Cust ID	Company Name	Customer Total	Order ID	Order Date	Order Total	Product ID	Unit Price	Quantity	Unit Total
ALFKI	Alfreds Futterkiste	$4,596.20	10643	8/25/2015	$1,086.00	28	$45.60	15	$684.00
						39	$18.00	21	$378.00
						46	$12.00	2	$24.00
			10692	10/3/2015	$878.00	63	$43.90	20	$878.00
			10702	10/13/2015	$330.00	3	$10.00	6	$60.00
						76	$18.00	15	$270.00
			10835	1/15/2016	$851.00	59	$55.00	15	$825.00
						77	$13.00	2	$26.00
			10952	3/16/2016	$491.20	6	$25.00	16	$400.00
						28	$45.60	2	$91.20
			11011	4/9/2016	$960.00	58	$13.25	40	$530.00
						71	$21.50	20	$430.00
ANATR	Trujillo Emparedados	$1,402.95	10308	9/18/2015	$88.80	69	$28.80	1	$28.80
						70	$12.00	5	$60.00
			10625	8/8/2016	$479.75	14	$23.25	3	$69.75

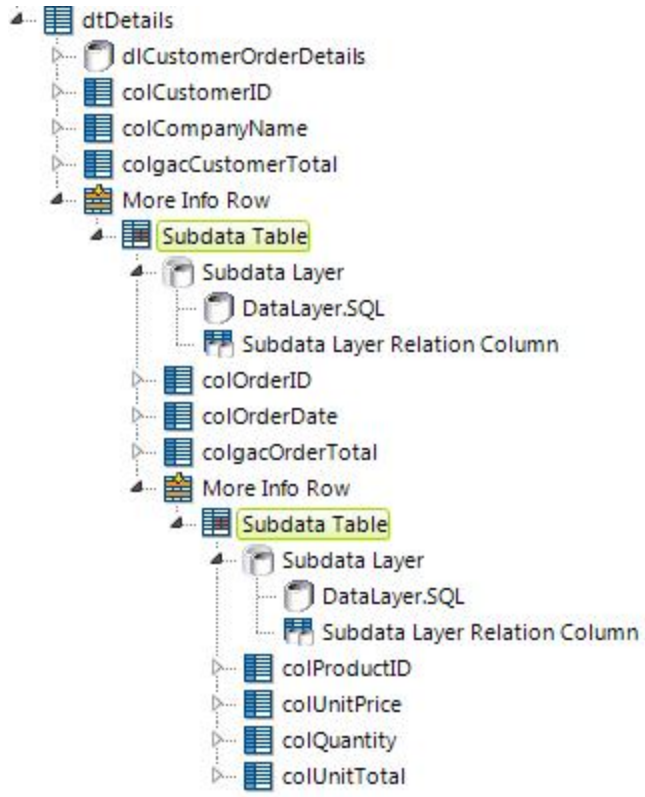
and the resulting report shows how the data is grouped and indented by virtue of suppressing duplicate data.

Working with SubData Tables

As mentioned earlier, another method of presenting hierarchical data is to use a subData Table beneath a Data Table Column or More Info Row element, see "Data Table Rows" on page 466. *We don't generally recommend this approach, although the necessary elements remain in Logi Info for backward compatibility*.

SubData Tables receive data from the **Subdata Layer** element, which uses a standard datalayer to retrieve data and then relates its columns to those of the parent Data Table's datalayer.

 Most developers find it easier to use a subreport that includes a regular Data Table, rather than a subData Table. One advantage to this approach is that a subreport, which is a separate report definition, can be independently developed and tested, and then embedded in a parent report using the SubReport element. This technique works well with the More Info Row element and does not require hierarchical data. More information is available in *Working with SubReports*.



The example definition shown above employs two More Info Row elements and two SubData Table elements to create a hierarchy of data from the previous examples. In this case, the detail data may not always be visible in the report and may require the user to do some clicking to make it visible.

For information about using SubData Tables, see "SubData Tables" on the next page.

SubData Tables

The purpose of the **SubData Table** element is to create a table that shows the child records from a hierarchical result set, or a "table within a table". This is useful for creating **sub-reports** and **drill-down** reports.

The following topics discuss the techniques required to work with SubData Tables:

- "Drilling Down" Using a SubData Table
- Grouping Data with Subdata Layers

 Many developers find it easier to use a subreport that includes a regular Data Table, rather than a subData Table. One advantage to this approach is that a subreport, which is a separate report definition, can be independently developed and tested, and then embedded in a parent report using the SubReport element. This approach works especially well with the More Info Row element (described below) and does not require hierarchical data. For more information on subreports, see *Working with SubReports*.

"Drilling Down" using a SubData Table

When presenting summarized data in table, there's often a need to "drill-down"; to display the supporting data. Sometimes this means displaying an entirely **different** detail report, but frequently the detail data can be shown right **within** the original Data Table. Here's an example:

Customer ID	Company	City	Contact	Country	Total
ALFKI	Alfreds Futterkiste	Berlin	Maria Anders	Germany	$4,596.20
ANATR	Ana Trujillo Emparedados y helados	México D.F.	Ana Trujillo	Mexico	$1,402.95
ANTON	Antonio Moreno Taquería	México D.F.	Antonio Moreno	Mexico	$7,515.35
AROUT	Around the Horn	London	Thomas Hardy	UK	$13,806.05
BERGS	Berglunds snabbköp	Luleå	Christina Berglund	Sweden	$6,968.13

Shown above is a **Data Table** that presents customer information, including their order totals. If more detailed information about a customer's orders is desired, the **Customer ID** value has been configured as a drill-down link.

Customer ID	Company	City	Contact	Country	Total
ALFKI	Alfreds Futterkiste	Berlin	Maria Anders	Germany	$4,596.20

Order ID	Order Date	Shipped Date	Shipped To	Freight	Order Total
10643	8/25/1997	9/2/1997	Berlin	29.46	1086.36
10692	10/3/1997	10/13/1997	Berlin	61.02	878.00
10702	10/13/1997	10/21/1997	Berlin	23.94	330.12
10835	1/15/1998	1/21/1998	Berlin	69.53	851.45
10952	3/16/1998	3/24/1998	Berlin	40.42	491.21
11011	4/9/1998	4/13/1998	Berlin	74.16	960.89

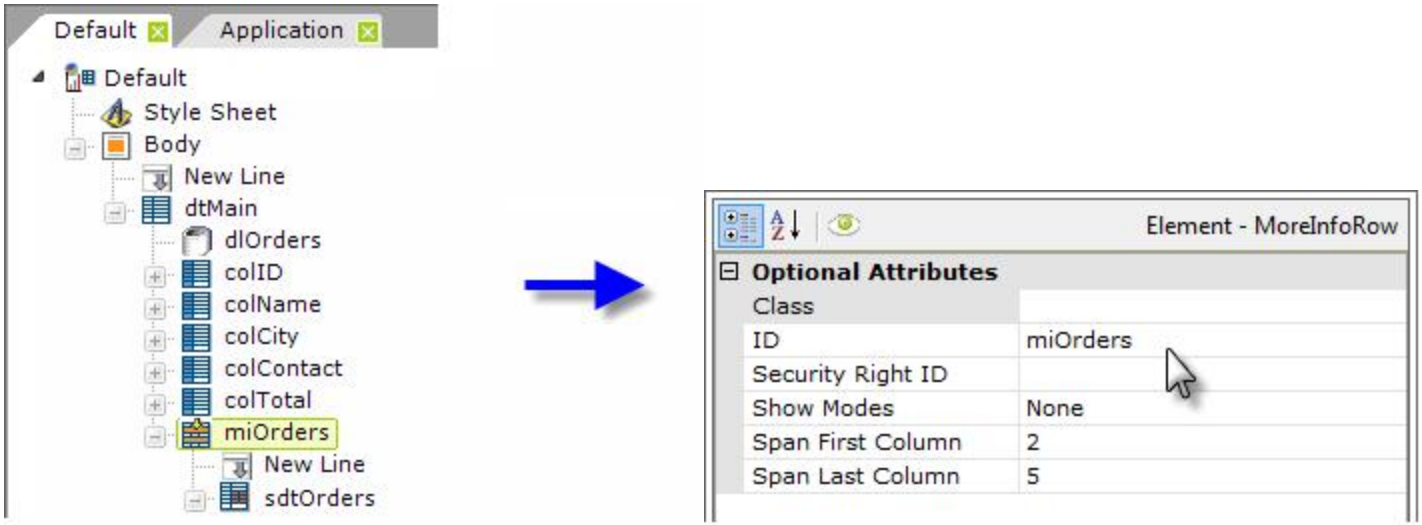
ANATR	Ana Trujillo Emparedados y helados	México D.F.	Ana Trujillo	Mexico	$1,402.95
ANTON	Antonio Moreno Taquería	México D.F.	Antonio Moreno	Mexico	$7,515.35
AROUT	Around the Horn	London	Thomas Hardy	UK	$13,806.05
BERGS	Berglunds snabbköp	Luleå	Christina Berglund	Sweden	$6,968.13

When the Customer ID link is clicked, the main table **expands** downward, as shown above, and a second or "sub" Data Table, with order information for the selected customer, is displayed within it. Clicking the link again will hide the subData Table. A More Info Row element provides the expansion capability, see "Data Table Rows" on page 466.

To generate this report, **hierarchical data** is required. This kind of data can be created with complicated SQL queries, however, Logi Studio offers an alternate, easier approach. The **Subdata Layer**, and **Subdata Layer Relation Column** elements can be used to create hierarchical data without the need for complicated SQL queries.

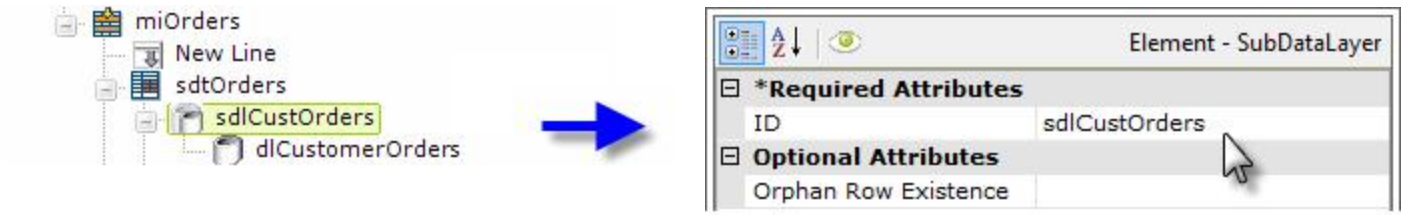
Grouping Data with Subdata Layers

The first step in creating drill-down functionality, like that shown in ""Drilling Down" using a SubData Table" on page 513, is to include **More Info Row** element and **Sub-Data Table** elements in your report.



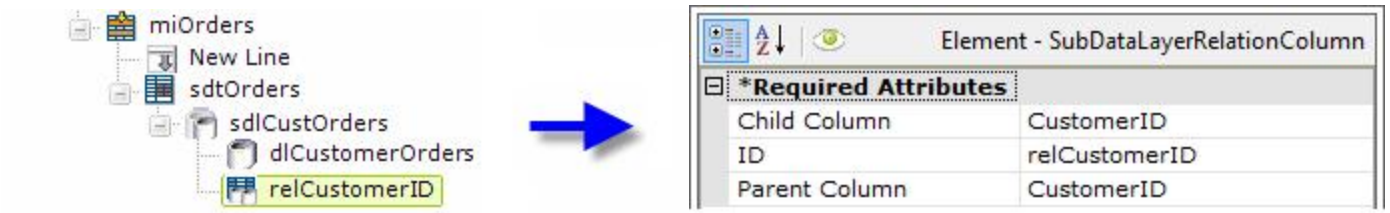
The definition example shown above includes a main Data Table, the **More Info Row** element ("miOrders") and a **SubData Table** element ("sdtOrders"). The configuration of the More Info Row's attributes is shown.

The **More Info Row Column** element can be used to closely align More Info Row columns with the columns of its parent table. When this element is used, the More Info Row element's **Span First Column** and **Span Last Column** attributes are left *blank*, and the More Info Row Column element's **Column Span** attribute is used instead to control spanning of parent table columns.



Now a **Subdata Layer** element ("sdlCustOrders") has been added beneath the SubData Table; it provides a container for a regular datalayer element ("dlCustomerOrders").

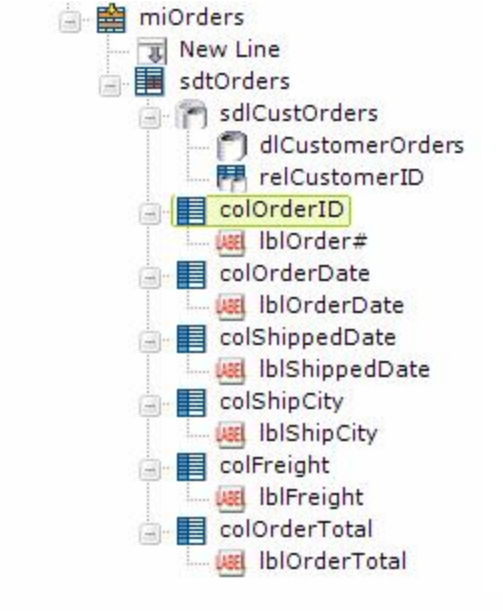
The **datasource** used by the regular datalayer element beneath a Subdata Layer element has to be the *same* as the datasource used by the main Data Table. In other words, they both get their data from the same place. This generally means that the same type of regular datalayer element will be used for both, so, if a DataLayer.SQL element is used to retrieve data for the main table, then a DataLayer.SQL element will be used with the Subdata Layer element.



Finally, a **Subdata Layer Relation Column** element is added beneath the Subdata Layer element. Its attributes are set, as shown above, to create the relationship between the data in the main table's datalayer (Parent Column) and the data in the subdatalayer (Child Column).

The practical effect of this is to create a single, hierarchical XML data object that contains child data records interspersed within their parent data records.

Depending on the data returned to the sub-datalayer, it's possible that some **child rows** will not have a matching **parent row**; these are "orphan rows". It may be possible to ensure that there will be no orphan rows through, for example, the right SQL query or external validation of the data. If that's the case, then disabling error-checking in the Subdata Layer, by setting its **Orphan Row Existence** attribute to *False*, may yield faster performance. Otherwise use the default, *True*, to prevent errors.



The definition is completed by adding the **Data Table Column** and **Label** elements that are typical for any Data Table, as shown above. Keywords: SubDataLayer, SubDataTable, SubDataLayerRelationColumn

Crosstab Tables

A **Cross Tabulation** (often abbreviated as "crosstab") is a Data Table that displays the *joint distribution* of two or more variables simultaneously. Sometimes called "pivot tables", they make it easy to sort, count, and total their data. The Logi **Crosstab Table** element makes it easy to implement this kind of table.

The following topics discuss Crosstab Tables:

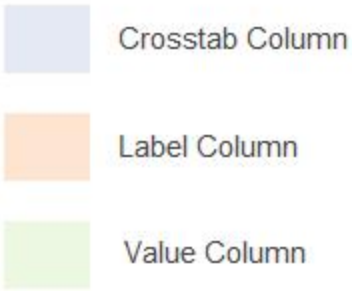
About the Crosstab Table

The Logi Info **Crosstab Table** element is a dynamic, data-driven reporting component that consists of the following three datalayer column types:

- Crosstab (also called Header) Column- Creates a new *column* in the Crosstab Table for each *unique* value. The column value appears in the table header.
- Label Column- creates a new *row* in the Crosstab Table for each *unique* value. The column value appears in the first (left-most) column of the row.
- Value Column- displays a value in each cell at the intersection of the Crosstab columns and Label rows (excluding the header row and first column).

Here's how each of these data column types appear in a Crosstab Table:

Employee Name	2013	2014	2015
Buchanan, Steven	$18,383.92	$30,716.47	$19,691.90
Callahan, Laura	$22,240.12	$56,032.62	$48,589.55
Davolio, Nancy	$35,764.51	$93,148.09	$85,797.96
Dodsworth, Anne	$9,894.52	$26,310.39	$41,103.16
Fuller, Andrew	$21,757.06	$70,444.14	$74,336.55
King, Robert	$15,232.16	$60,471.19	$48,864.88
Leverling, Janet	$18,223.96	$108,026.14	$76,562.74
Peacock, Margaret	$49,945.12	$128,809.79	$54,135.94
Suyama, Michael	$16,642.61	$43,126.37	$14,144.16



Each value displayed in the Value Column cells is the result of an *aggregation*, performed on the original data from the specified Value Column. The example above displays years for the Crosstab Column, employee names for the Label Column and a *sum* of corresponding subtotals for the Value Columns. So, when the year is *2013* and the employee is *Nancy Davolio*, the sum of all corresponding subtotals is *$35,764.51*.

To prevent creation of unmanageable tables, numeric type columns are not available for use in a Crosstab Table as the Crosstab (Header) or the Label columns.

The following functions are available for aggregating Value Columns:

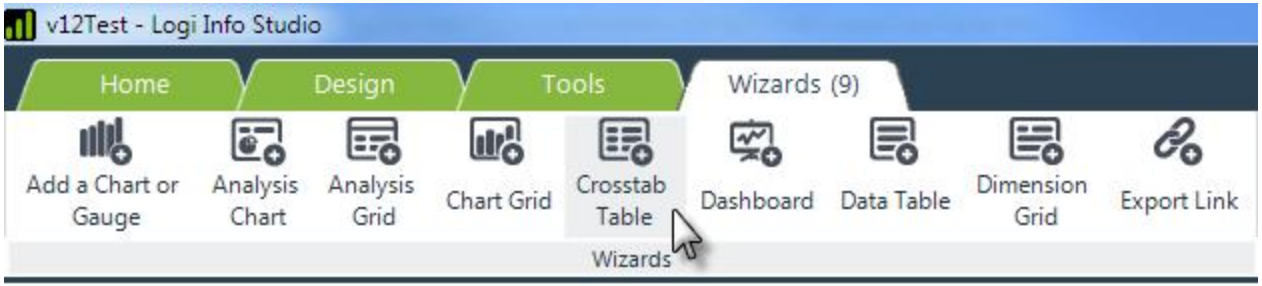
- **Any -** Displays a value from any of the rows. This can be used when the specified Value Column data is a string that isn't appropriate for aggregation; for example, when there's just one record to represent each crosstab cell.
- **Average -** Returns the average of all corresponding records in the specified Value Column.
- **Count -** Returns the total number of corresponding records in the specified Value Column.
- **DistinctCount** - Returns the total number of unique corresponding records in the specified Value Column.
- **Max** - Returns the maximum value in the specified Value Column.
- **Median -** Returns the value that separates the higher half of all values in the specified Value Column from the lower half.
- **Min** - Returns the minimum value in the specified Value Column.
- **Mode** - Returns the value that occurs the most frequently in records in the specified Value Column.
- **StdDev** - (Standard Deviation) Returns a simple measure of the variability of data in records in the specified Value Column. A low standard deviation indicates that the values tend to be very close to each other, while a high standard deviation indicates that the values are "spread out" over a large range.
- **Sum -** Returns the sum of all corresponding records in the specified Value Column

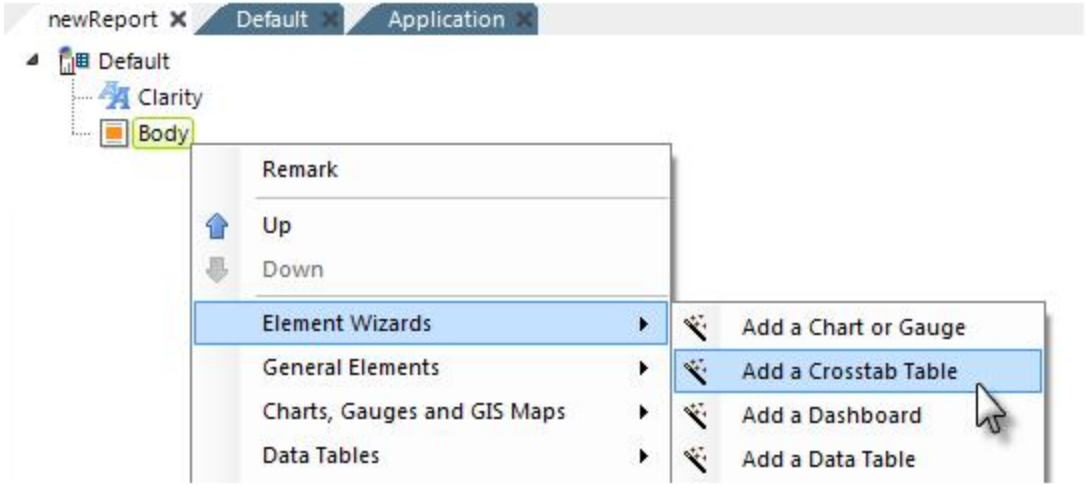
Columns with null values are *excluded* from aggregations by default. If you want to include them instead, create the constant `rdCalculationsIncludeNulls` in your _Settings definition and set it to *True.* This will affect all calculations throughout the application.

What about differences between columns, either as values or as percentages? The **Crosstab Comparison** element can display these differences in a variety of ways. The element is discussed in detail in "Comparing, Sorting, and Summarizing Columns" on page 542.

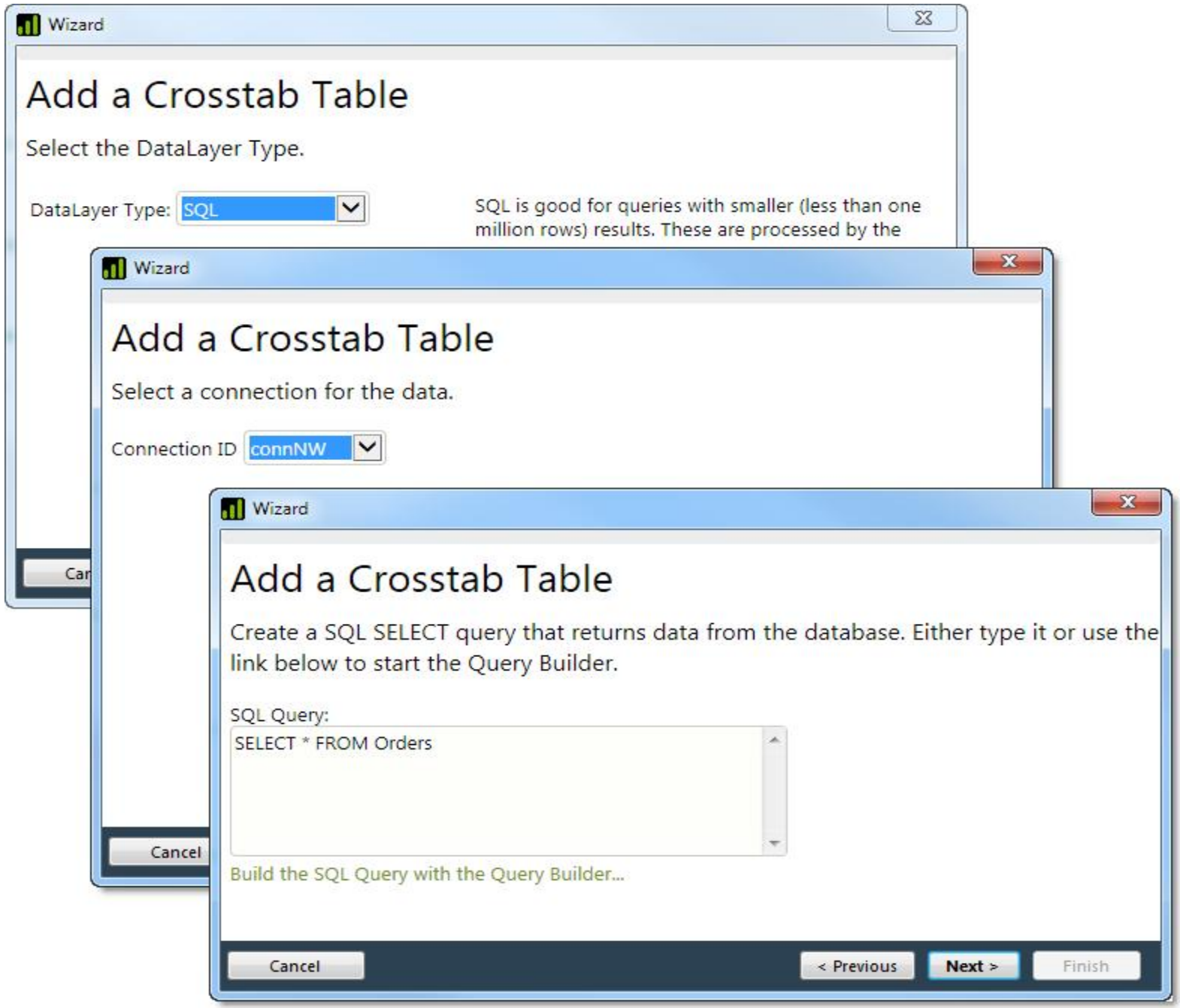
Using the Crosstab Table Wizard

The fastest and easiest way to create a Crosstab Table is to use the **Crosstab Table Wizard** in Logi Studio. Here's how:

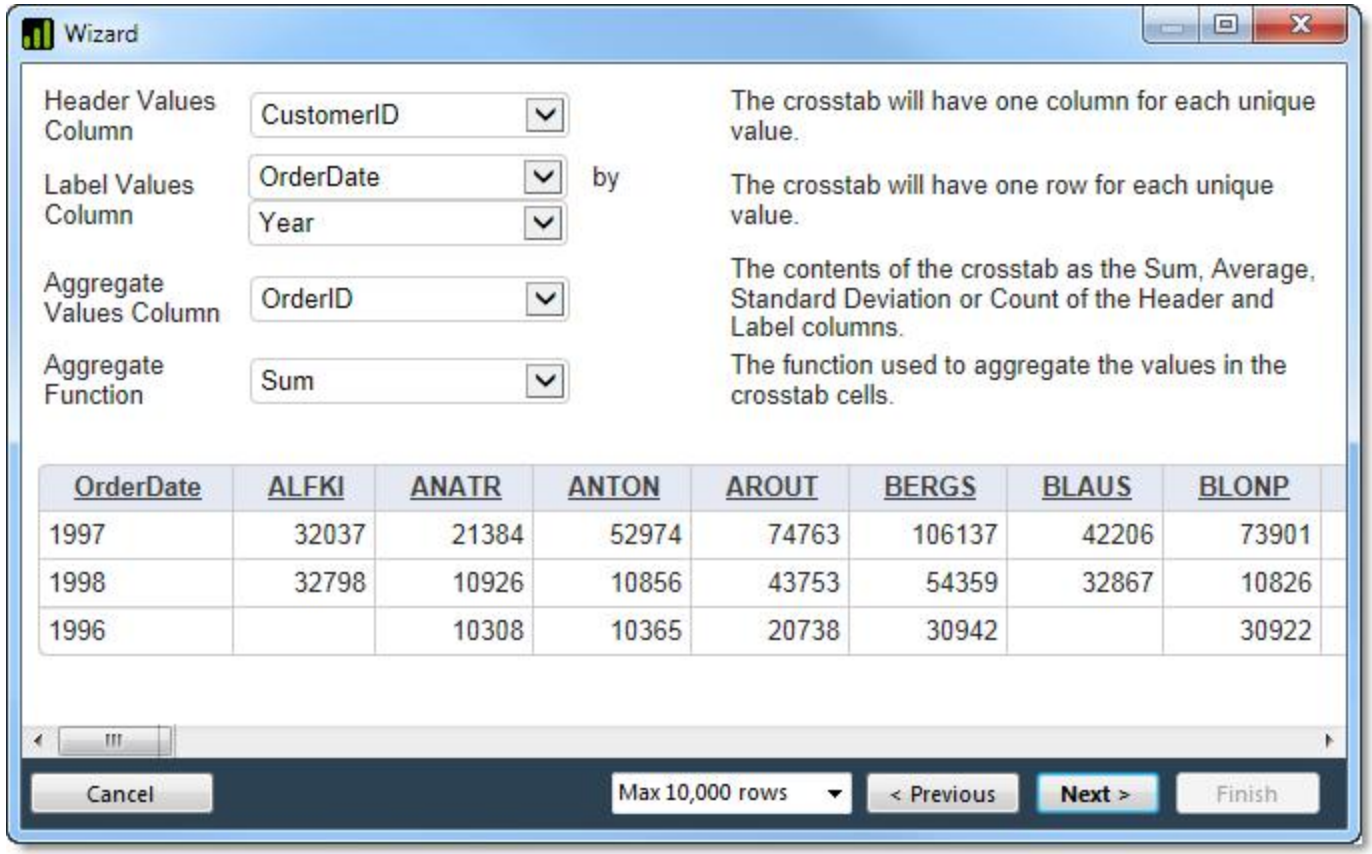




1. As shown above, in a definition in Logi Studio, select an element and either click the **Crosstab Table** item in the main menu's Wizards tab, or right-click the element and select *Element Wizards*, and then select *Add a Crosstab Table* from the secondary menu.


Wizard
Add a Crosstab Table
Select the DataLayer Type.
DataLayer Type: SQL
SQL is good for queries with smaller (less than one million rows) results. These are processed by the
Wizard
Add a Crosstab Table
Select a connection for the data.
Connection ID connNW
Cancel
Wizard
Add a Crosstab Table
Create a SQL SELECT query that returns data from the database. Either type it or use the link below to start the Query Builder.
SQL Query:
SELECT * FROM Orders
Build the SQL Query with the Query Builder...
Cancel
< Previous
Next >
Finish

2. A series of dialog boxes, shown above, will be displayed. These all relate to the retrieval of the data. Make appropriate selections for your application and click **Next** to move to the next dialog box.



3. The wizard will display a model Crosstab Table in a dialog box, where you can adjust it, if desired. If the table is just as you want it, click **Next** to exit the wizard.

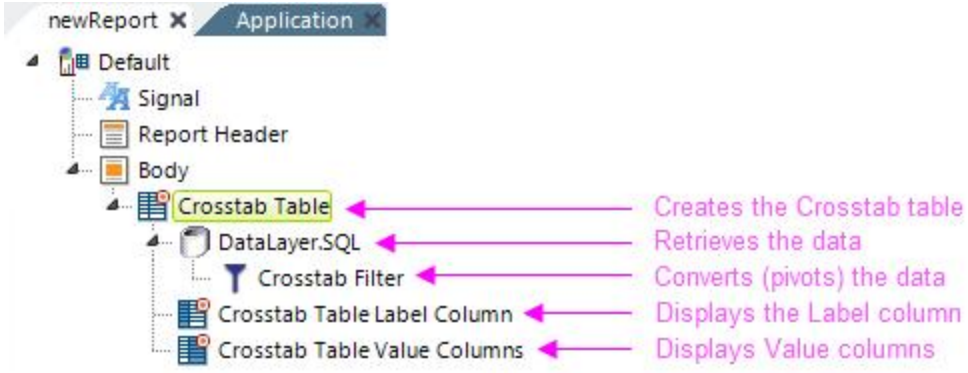
In order to prevent creation of unmanageable tables, numeric type columns are not available for use as the Header Values or Label Values columns.

The **Max Rows** option at the bottom of the dialog box controls the maximum number of rows that will be retained in the table's datalayer.

When you exit the wizard, it will finish inserting all the elements for the table, properly configured, into your report definition. Only the table itself will be added, *not* any of the configuration controls you've just worked with. Preview your application and you should see your new table.

Creating Crosstab Tables Manually

You can create a Crosstab Table in Logi Studio manually by adding the appropriate elements to a report definition. Every Crosstab Table uses the following minimum components:



- One **Crosstab Table** element
- One or more **DataLayer** elements with a **Crosstab Filter** element to retrieve and shape the data
- One or more **Crosstab Table Label Column** elements to display the Label column data
- One or more **Crosstab Table Value Columns** element to display the Value column data

The elements above define the most basic Crosstab Table. Other elements, such as **Condition Filters**, **Sort Filters**, and **Calculated Columns**, can be used to manipulate the data in the datalayer in the same way they would be used for any Data Table or chart.

If you use a **Sort Filter** beneath your datalayer, its effect on the table display may be different depending on its placement *before* (above) or *after* (below) the Crosstab Filter element. This is particularly relevant if your data returns some zero values, and you may want to experiment with this element in different locations in your definition.

The Crosstab Table Label Column element includes the **Scope Row Header** attribute. When set to *True*, the HTML output for data cells will be modified to improve Section 508 accessibility, creating a column which includes header information for each row, changing the `<TD>` tags to `<TH Scope="Row">` tags.

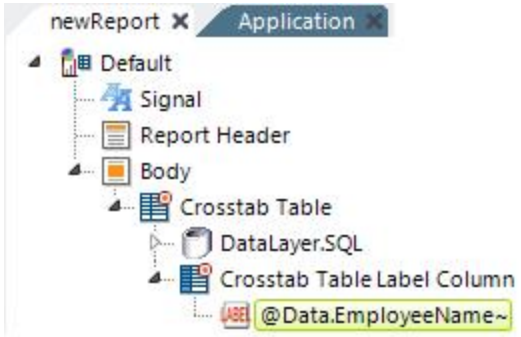
A number of elements that are available for use with regular Data Tables, such as **Header Row**, **Interactive Paging**, and **More Info Row**, can also be used with Crosstab Tables. For information about using these elements, see "Data Tables" on page 425.

Adding "HTML Attribute Params" to your Crosstab Table enables you to apply your application to work with other frameworks or libraries easier. Depending on the type of HTML Attribute Params you add, there will be different parameters available to use, or you can define your own parameters. If an attribute was set in both Element and HTML Attribute Params, the one set in the HTML Attribute Params will be ignored.

Working with Crosstab Table Columns

Instead of using the wizard, developers can manually build Crosstab Tables using two types of Crosstab Table columns: Label Columns and Value Columns. Crosstab Tables have one Label Column by default; the number of Value Column cells is determined by the number of Crosstab Column values.

Here's how to add a Crosstab Table Label column:

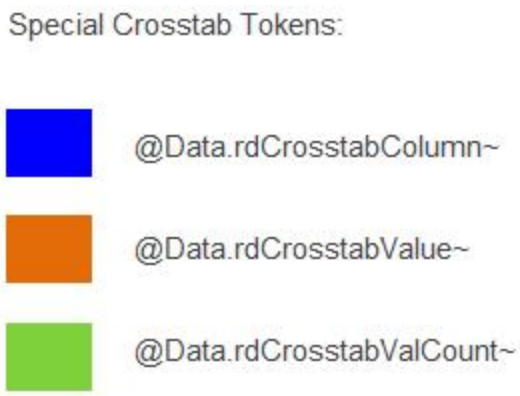


1. As shown above, in Logi Studio, select the parent Crosstab Table element and add a **Crosstab Table Label Column** element beneath it. Configure any of its optional attributes.
2. Then, add a **Label** element beneath the new column and use a regular @Data token to reference values from a column returned by the datalayer.

Next, we want to add **Crosstab Table Value Columns** elements to create the Value columns for the Crosstab Table. Value columns contain the results of the aggregations of the data from the specified column.

But the results of the aggregations do not appear in the datalayer under any of its column names, so how is it referenced? Special @Data tokens allow you to access Crosstab column and Value data. Here are the tokens:

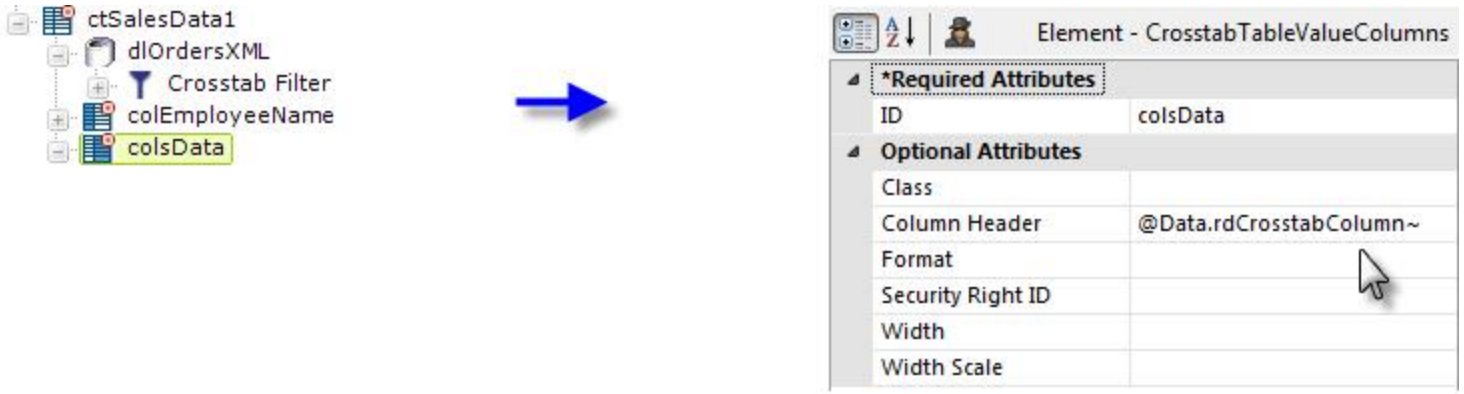
2012		2013		2014	
$954.40	Count: 23	$399.85	Count: 56	$360.00	Count: 41
$424.00	Count: 14	$1,440.00	Count: 41	$446.60	Count: 38
$1,414.80	Count: 18	$1,684.27	Count: 67	$1,332.74	Count: 42
$516.80	Count: 31	$2,097.60	Count: 78	$2,942.81	Count: 43
$642.20	Count: 10	$507.20	Count: 19	$210.20	Count: 14



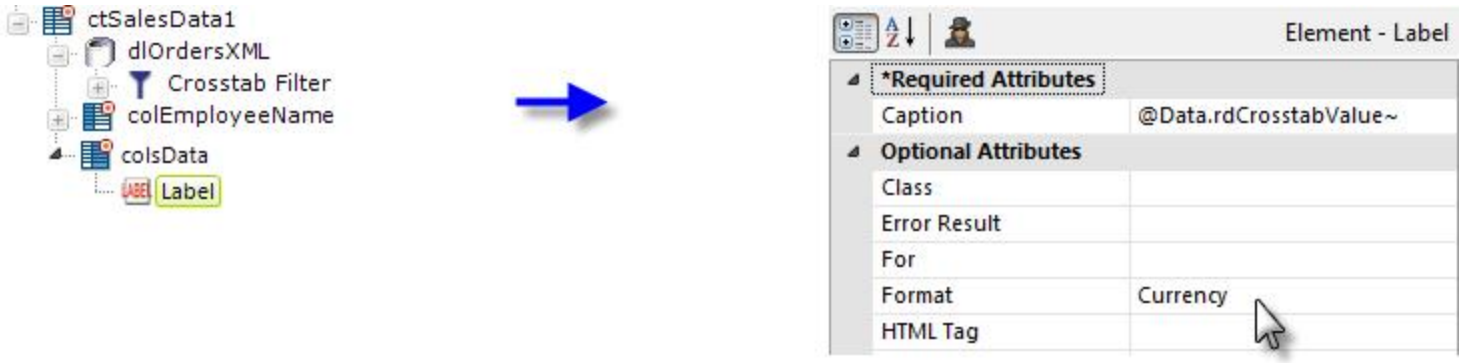
These special tokens are:

- **@Data.rdCrosstabColumn~** - Returns the values in the specified Crosstab Column
- **@Data.rdCrosstabValue~** - Returns the aggregated values in the specified Value Column
- **@Data.rdCrosstabValCount~** - Returns the number of corresponding records from the Value Column used to calculate the current aggregate value if the aggregation is *Sum*, *Count*, or *Average*.

Let's add a Value Column and set its attributes using the special tokens:



1. Beneath the Crosstab Table element, add a **Crosstab Table Value Column** element, as shown above.
2. We want to use data in the column headers, so set the element's **Column Header** attribute to one of the special tokens: *@Data.rdCrosstabColumn~.*

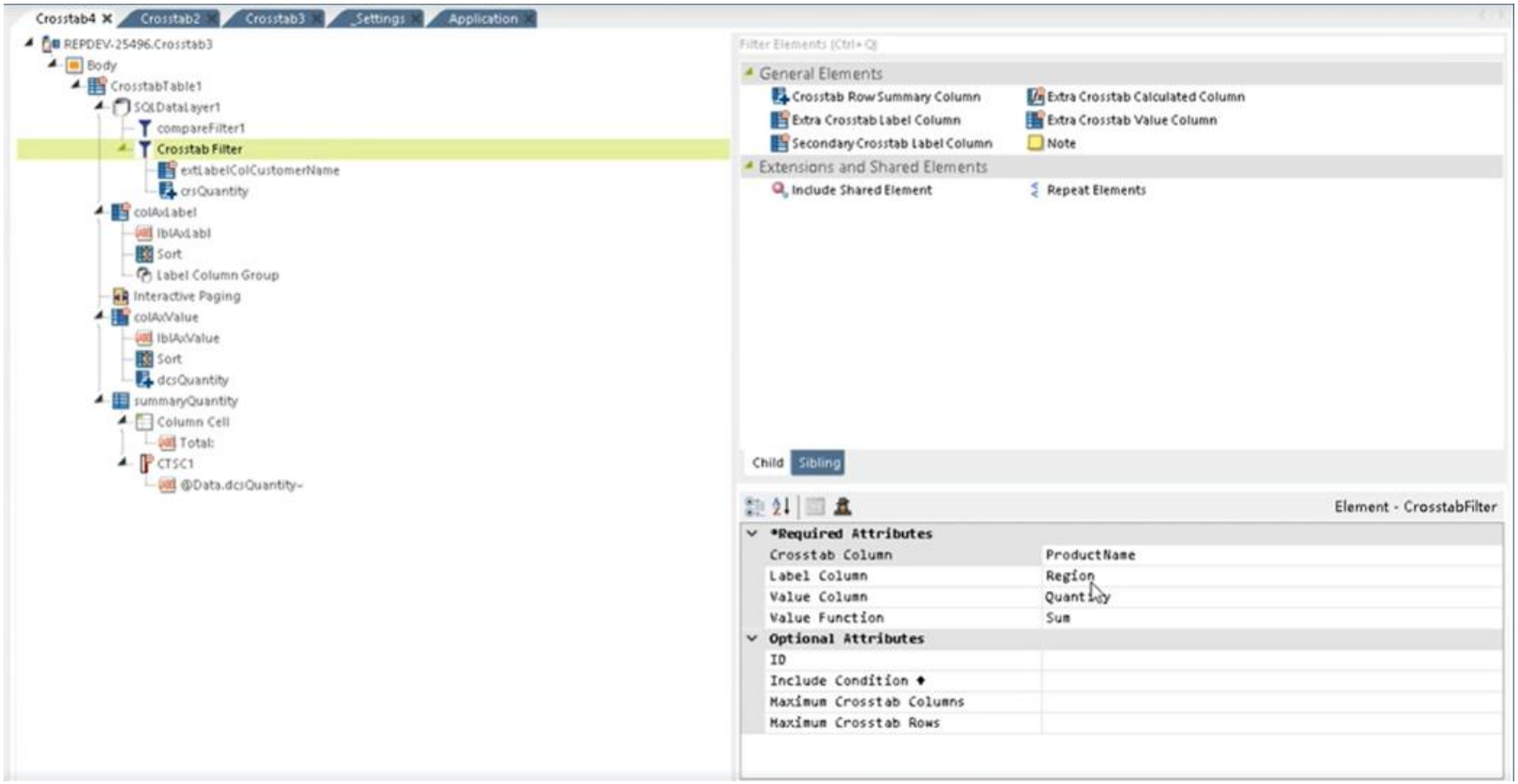


3. Add a **Label** element beneath the Crosstab Table Value Columns element and set its **Caption** attribute to *@Data.rdCrosstabValue~*, as shown above, to display the aggregated crosstab values.

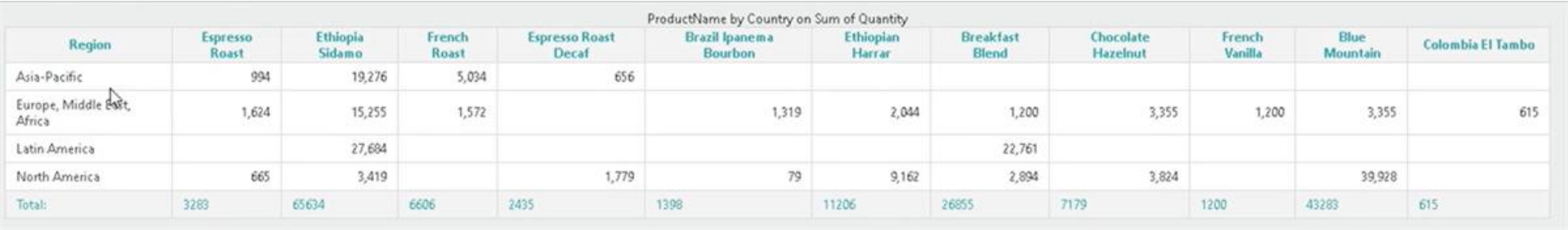
Grouping Crosstab Table Columns

You can incorporate Crosstab grouping by adding the **Secondary Crosstab Label Columns** to your Crosstab Filter. This element works like the Extra Crosstab Label Columns element, except it can be used to group Value Columns and Label Columns together.

In this example, the Crosstab Filter is grouped by the Label Column *Region*:



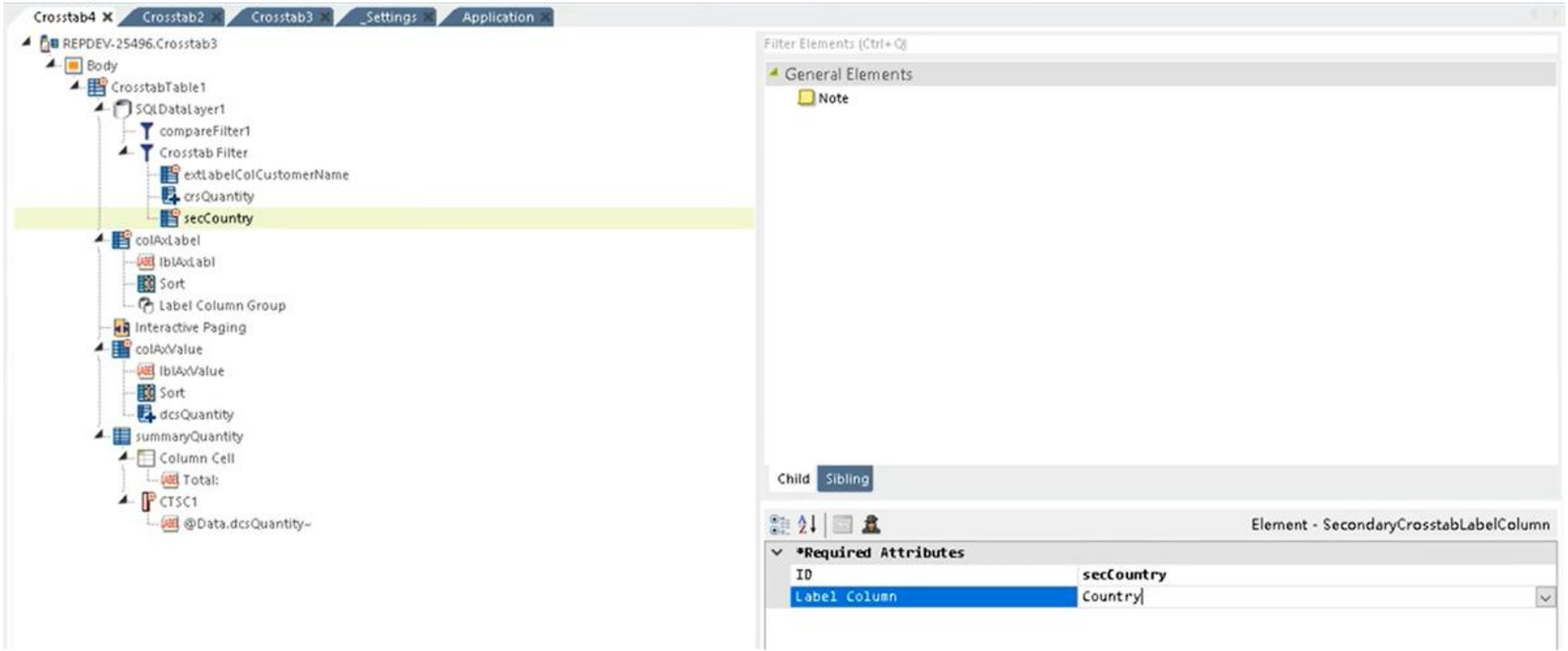
Here's what the Crosstab Table looks like, as is:



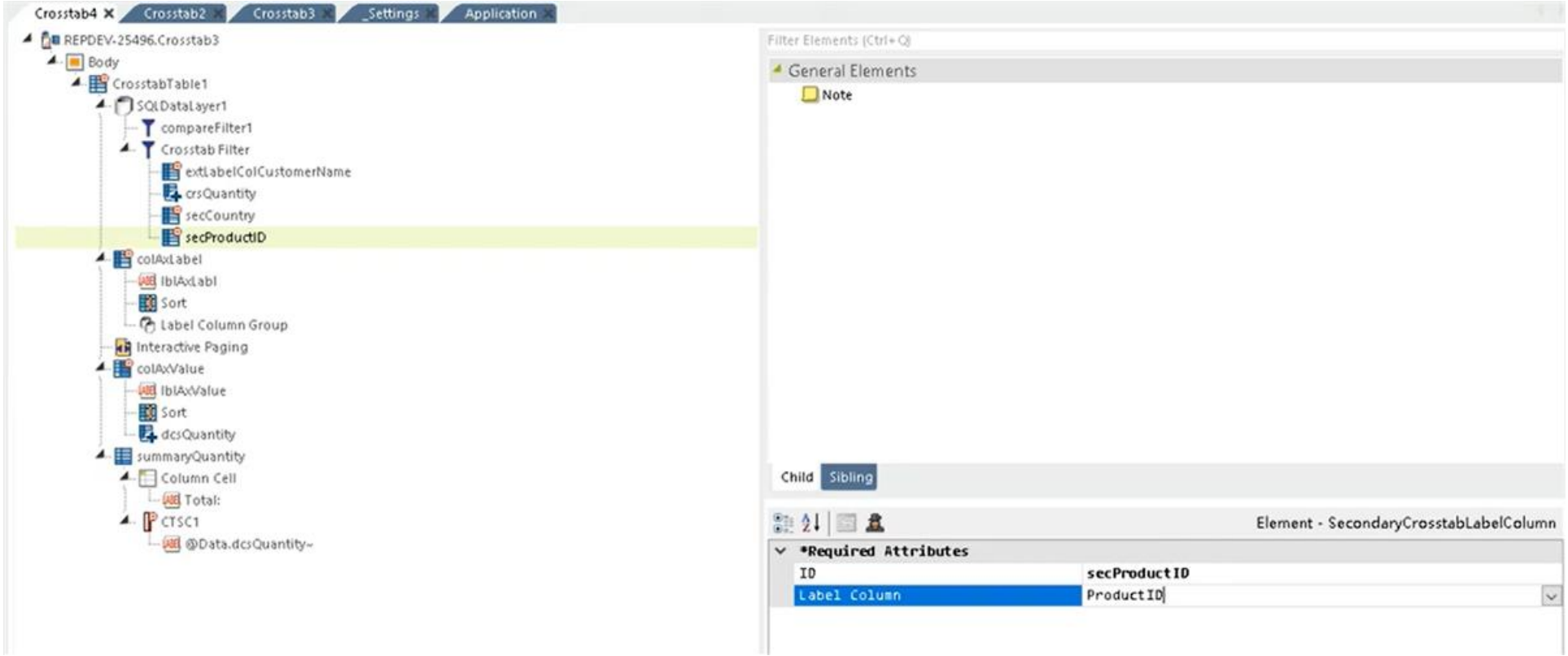
ProductName by Country on Sum of Quantity

Region	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
Asia-Pacific	994	19,276	5,034	656							
Europe, Middle East, Africa	1,624	15,255	1,572		1,319	2,044	1,200	3,355	1,200	3,355	615
Latin America		27,684					22,761				
North America	665	3,419		1,779	79	9,162	2,894	3,824		39,928	
Total:	3283	65634	6606	2435	1398	11206	26855	7179	1200	43283	615

1. First, add a **Secondary Crosstab Label Column** and give it an ID and assign a Label Column. In this example, we're adding "secCountry" with the Label Column "Country":

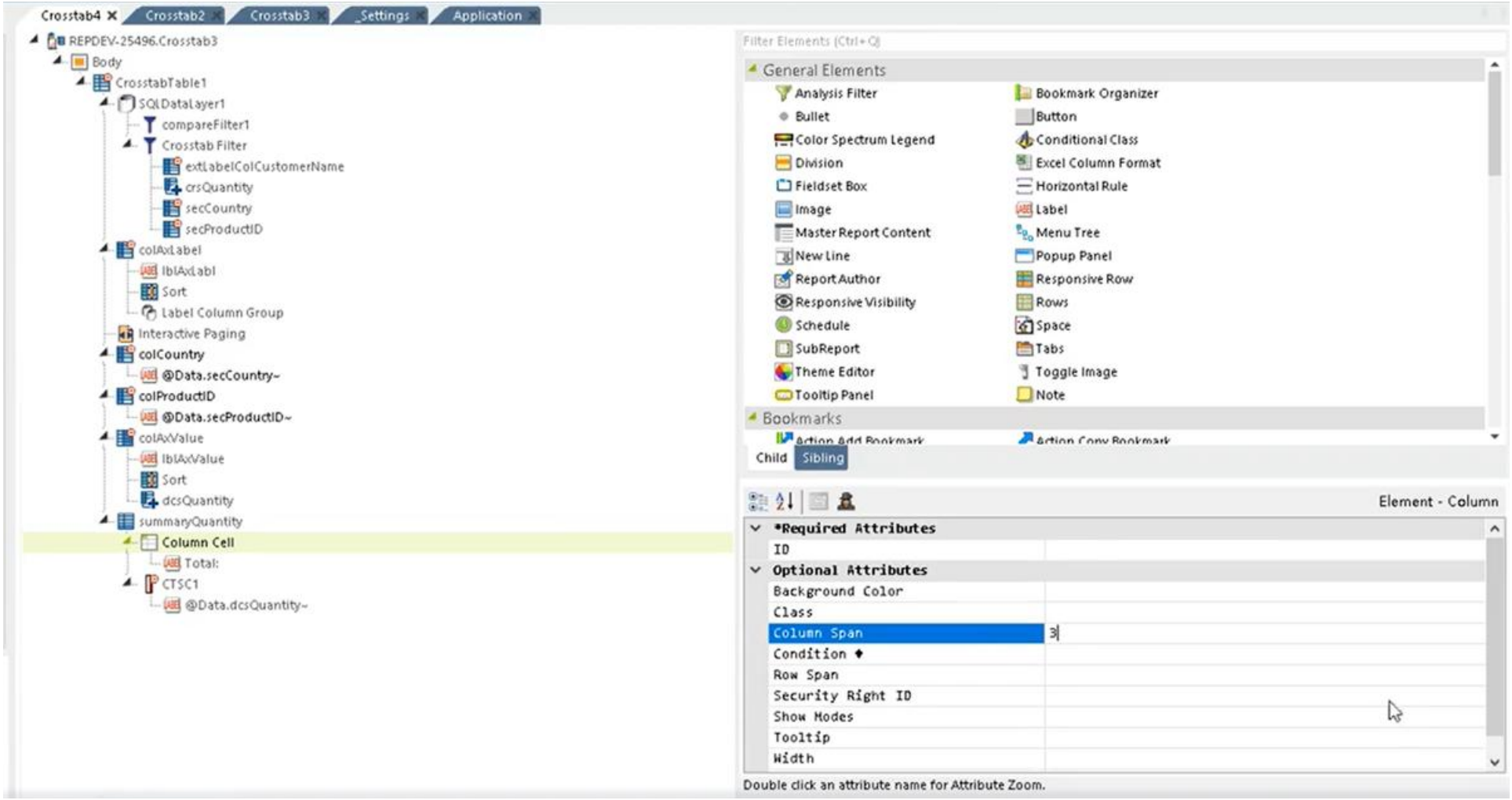


2. Then, add another **Secondary Crosstab Label Column** and give it an ID and assign a Label Column. In this example, we're adding "secProductID" with the Label Column "ProductID":



3. Add two **Crosstab Table Label Columns** to the Crosstab Filter that reflect the Secondary Crosstab Label Columns. In this example, we're adding a Crosstab Table Label Column with the ID "colCountry", Column Header titled "Country", and a Label element captioned "@Data.secCountry~" and another Crosstab Table Label Column with the ID "colProductID", Column Header titled "Product ID", and a Label element captioned "@Data.secProductID~".

4. Adjust the Column Span of the Summary Column to reflect the number of Label Columns you are grouping together. In this example, we're adjusting the span to 3 columns:

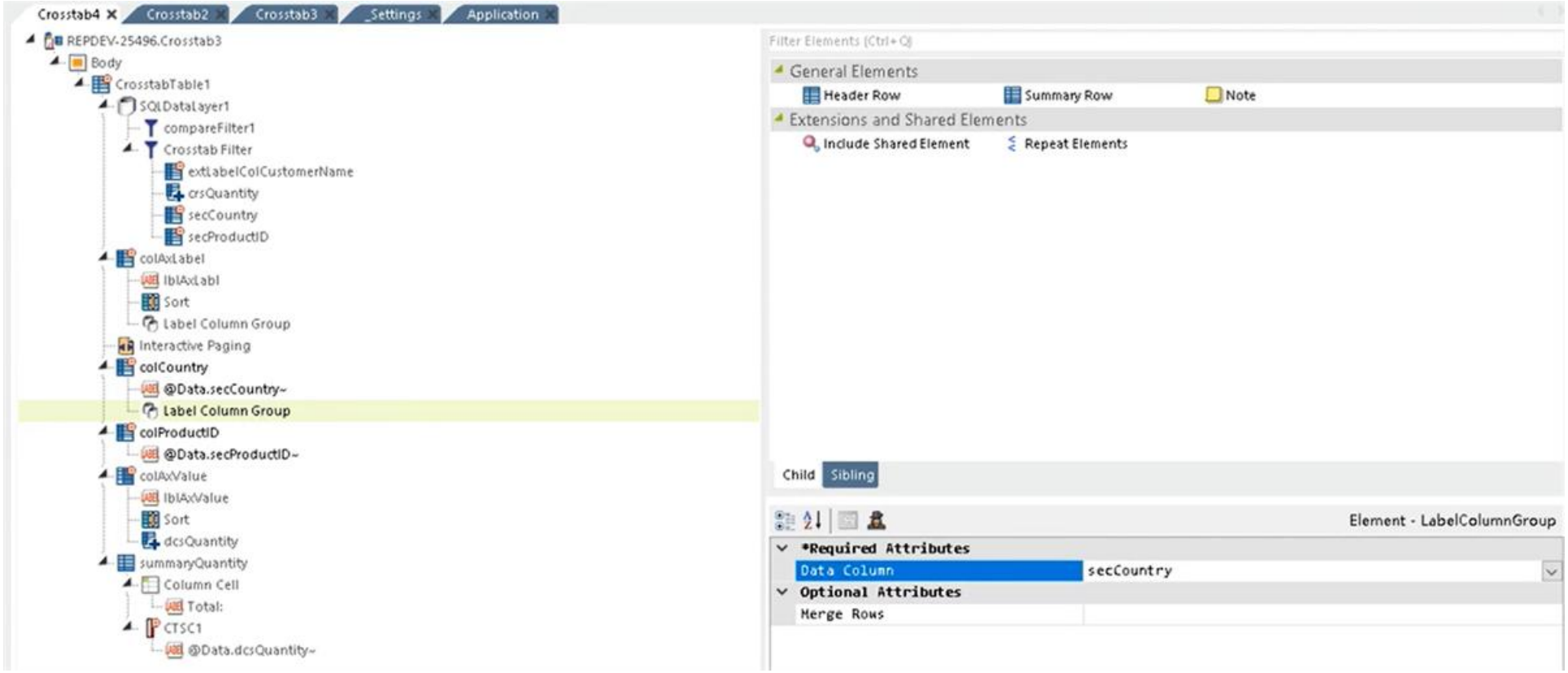


5. Select **Save** and **refresh** your report. Info displays the newly added *Country*, and *ProductID* Secondary Crosstab Label Columns:

ProductName by Country on Sum of Quantity

Region	Country	Product ID	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
Asia-Pacific	Australia	1	798										
	Australia	6		3,306									
	China	6		9,738									
	China	23			5,034								
	Japan	6		6,232									
	Malaysia	1	196										
	Malaysia	2				656							
Europe, Middle East, Africa	Belgium	1	854										
	Belgium	13					264						
	Belgium	20						2,044					
	Belgium	23			1,572								
	Poland	6		5,517									
	Poland	8							1,200				
	Russia	1	343										
	Russia	13					1,055						
	Saudi Arabia	6		9,738									
	Spain	1	427										
	Spain	25								3,355			
	Spain	26									1,200		
	Spain	28										3,355	
Total:			3283	65634	6606	2435	1398	11206	26855	7179	1200	43283	615

6. You can go one step further and add a **Label Column Group** element for each of the Data Columns you want to group. In this example, we're adding a Label Column Group for "Region" and "secCountry":



You can also change the Merge Rows attribute to "True" (default value is "False") to merge the cells grouped together. You must do this for each Label Column Group you add.

7. Select **Save** and **refresh** your report. Info groups the *Region* and *Country* columns:

ProductName by Country on Sum of Quantity

Region	Country	Product ID	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
Asia-Pacific	Australia	1	798										
		6		3,306									
	China	6		9,738									
		23			5,034								
	Japan	6		6,232									
	Malaysia	1	196										
		2				656							
Europe, Middle East, Africa	Belgium	1	854										
		13					264						
		20						2,044					
		23			1,572								
	Poland	6		5,517									
		8							1,200				
	Russia	1	343										
		13					1,055						
	Saudi Arabia	6		9,738									
	Spain	1	427										
		25								3,355			
		26									1,200		
		28										3,355	
Total:			3283	65634	6606	2435	1398	11206	26855	7179	1200	43283	615

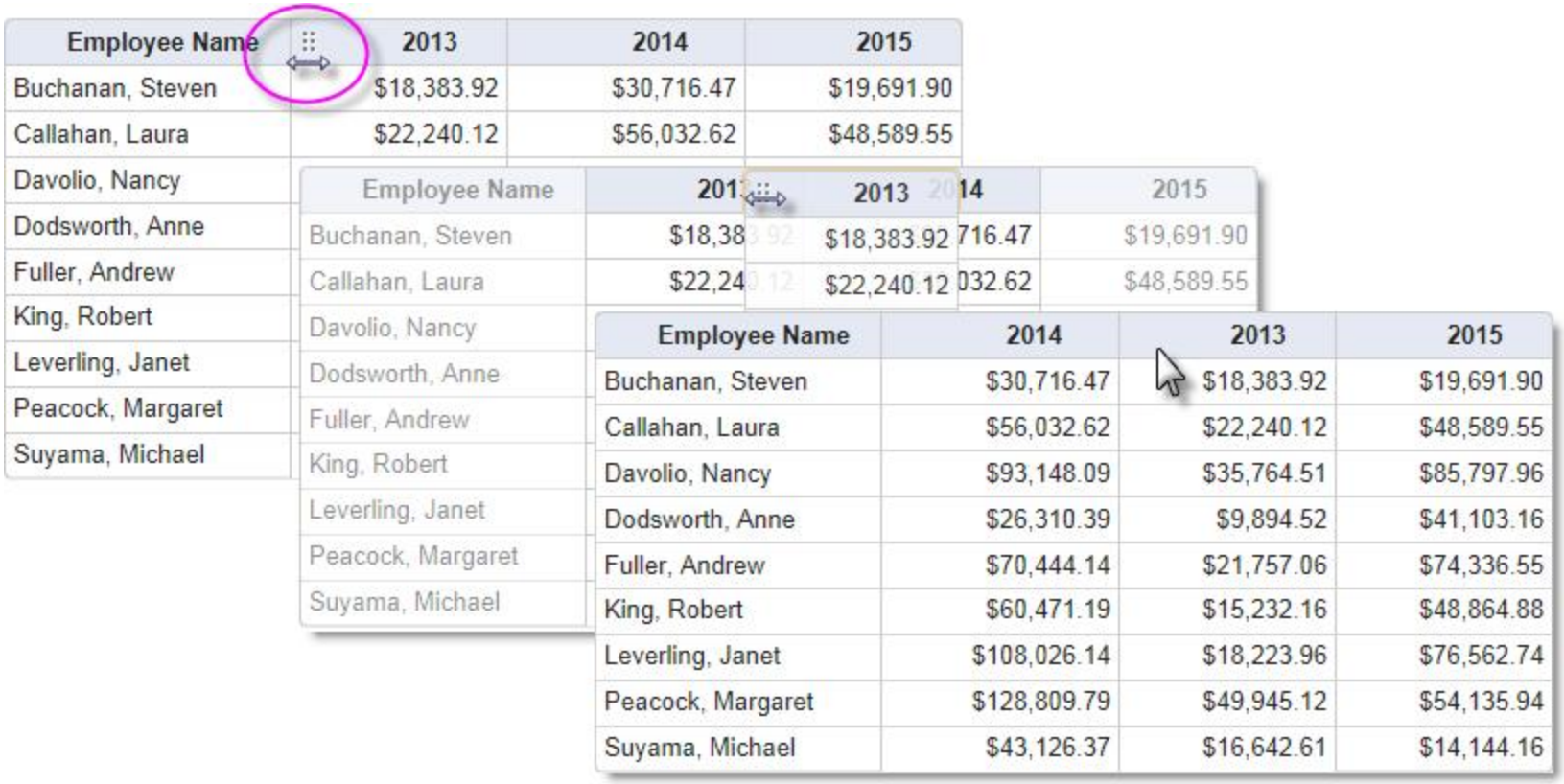
Here's what your Crosstab Table looks like if you *merged* the cells:

ProductName by Country on Sum of Quantity

Region	Country	Product ID	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
Asia-Pacific	Australia	1	798										
		6		3,306									
	China	6		9,738									
		23			5,034								
	Japan	6		6,232									
	Malaysia	1	196										
		2				656							
Europe, Middle East, Africa	Belgium	1	854										
		13					264						
		20						2,044					
		23			1,572								
	Poland	6		5,517									
		8							1,200				
	Russia	1	343										
		13					1,055						
	Saudi Arabia	6		9,738									
	Spain	1	427										
		25								3,355			
		26									1,200		
		28										3,355	
Total:			3283	65634	6606	2435	1398	11206	26855	7179	1200	43283	615

Arranging and Sizing Columns

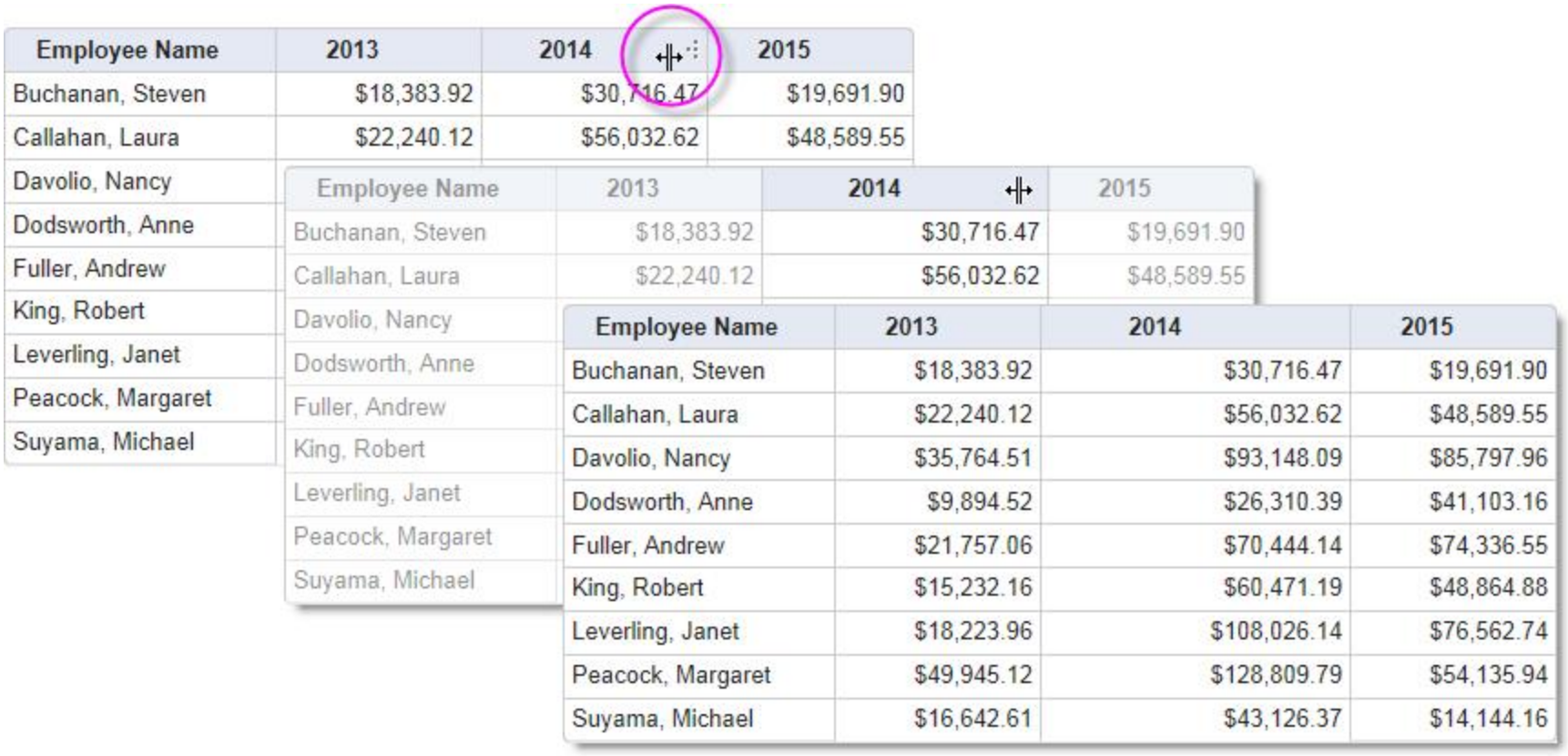
The developer can configure Crosstab Table columns so that they can be rearranged and/or resized at runtime by the user.



Employee Name	2013	2014	2015
Buchanan, Steven	$18,383.92	$30,716.47	$19,691.90
Callahan, Laura	$22,240.12	$56,032.62	$48,589.55
Davolio, Nancy	$35,764.51	$93,148.09	$85,797.96
Dodsworth, Anne	$9,894.52	$26,310.39	$41,103.16
Fuller, Andrew	$21,757.06	$70,444.14	$74,336.55
King, Robert	$15,232.16	$60,471.19	$48,864.88
Leverling, Janet	$18,223.96	$108,026.14	$76,562.74
Peacock, Margaret	$49,945.12	$128,809.79	$54,135.94
Suyama, Michael	$16,642.61	$43,126.37	$14,144.16

As shown above, when the Crosstab Table element's **Draggable Columns** attribute is set to *True*, a dotted icon appears in the column headers when the mouse is hovered over their left side and the cursor changes to left-right arrows. The entire column can then be dragged left or right and dropped into a new position. The new column arrangement will be "remembered" for the

duration of the current user session. Draggable Columns cannot be used when there are custom header rows, with columns that span multiple columns, in use. The default value for Draggable Columns is *False*.



Employee Name	2013	2014	2015
Buchanan, Steven	$18,383.92	$30,716.47	$19,691.90
Callahan, Laura	$22,240.12	$56,032.62	$48,589.55
Davolio, Nancy	$35,764.51	$93,148.09	$85,797.96
Dodsworth, Anne	$9,894.52	$26,310.39	$41,103.16
Fuller, Andrew	$21,757.06	$70,444.14	$74,336.55
King, Robert	$15,232.16	$60,471.19	$48,864.88
Leverling, Janet	$18,223.96	$108,026.14	$76,562.74
Peacock, Margaret	$49,945.12	$128,809.79	$54,135.94
Suyama, Michael	$16,642.61	$43,126.37	$14,144.16

When the **Resizable Columns** attribute is set to *True*, a dotted icon appears in the column headers when the mouse is hovered over their right side and the cursor changes shape. The icon can then be dragged left or right, contracting or expanding the width of the column. The new column width will be "remembered" for the duration of the current user session. The default value for Resizable Columns is *False*.

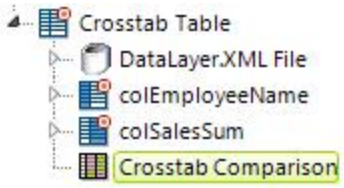
Resizable Columns cannot be used when there are custom header rows with columns that span multiple columns in use.

Comparing, Sorting, and Summarizing Columns

It's not uncommon to need to include the *difference*, either as a value or as a percentage, between two or more Value Columns in a Crosstab Table. There are two approaches to creating this information.

The Crosstab Comparison Element

The **Crosstab Comparison** element makes it easy to visually understand the differences between values in crosstab columns:



Each column is compared with the previous column and an indicator arrow or colored background is shown if the value increased or decreased. The amount of increase or decrease can be displayed as a numeric and/or percentage difference.

Employee Name	2013	2014	2015
Buchanan, Steven	$18,383.92	$30,716.47 ↑	$19,691.90 ↓
Callahan, Laura	$22,240.12	$56,032.62 ↑	$48,589.55 ↓
Davolio, Nancy	$35,764.51	$93,148.09 ↑	$85,797.96 ↓
Dodsworth, Anne	$9,894.52	$26,310.39 ↑	$41,103.16 ↑
Fuller, Andrew	$21,757.06	$70,444.14 ↑	$74,336.55 ↑
King, Robert	$15,232.16	$60,471.19 ↑	$48,864.88 ↓
Leverling, Janet	$18,223.96	$108,026.14 ↑	$76,562.74 ↓
Peacock, Margaret	$49,945.12	$128,809.79 ↑	$54,135.94 ↓
Suyama, Michael	$16,642.61	$43,126.37 ↑	$14,144.16 ↓

Comparison Style = *Arrows*

2015
$19,691.90 ↓ -35.89%
$48,589.55 ↓ -13.28%
$85,797.96 ↓ -7.89%
$41,103.16 ↑ 56.22%
$74,336.55 ↑ 5.53%
$48,864.88 ↓ -19.19%

Comparison Style = *Arrows*
Comparison Show Values = *Percent*

2015
$19,691.90 ↓
$48,589.55 ↓ -35.89%
$85,797.96 ↓
$41,103.16 ↑
$74,336.55 ↑
$48,864.88 ↓
$76,562.74 ↓
$54,135.94 ↓
$14,144.16 ↓

Comparison Style = *Arrows*
Comparison Show Tooltips = *Percent*

As shown above, comparisons between columns can be indicated by colored arrows and percentages or literal values, within each column cell or in tooltips that appear when the mouse is hovered on a value. Instead of the colored arrows, you can display a colored cell background, covering a spectrum, to indicate the differences.

The unique attributes of the Crosstab Comparison element are:

Attribute	Description
Comparison	Enables display of the numerical difference in a tooltip when the mouse is hovered over a crosstab value. Dis-

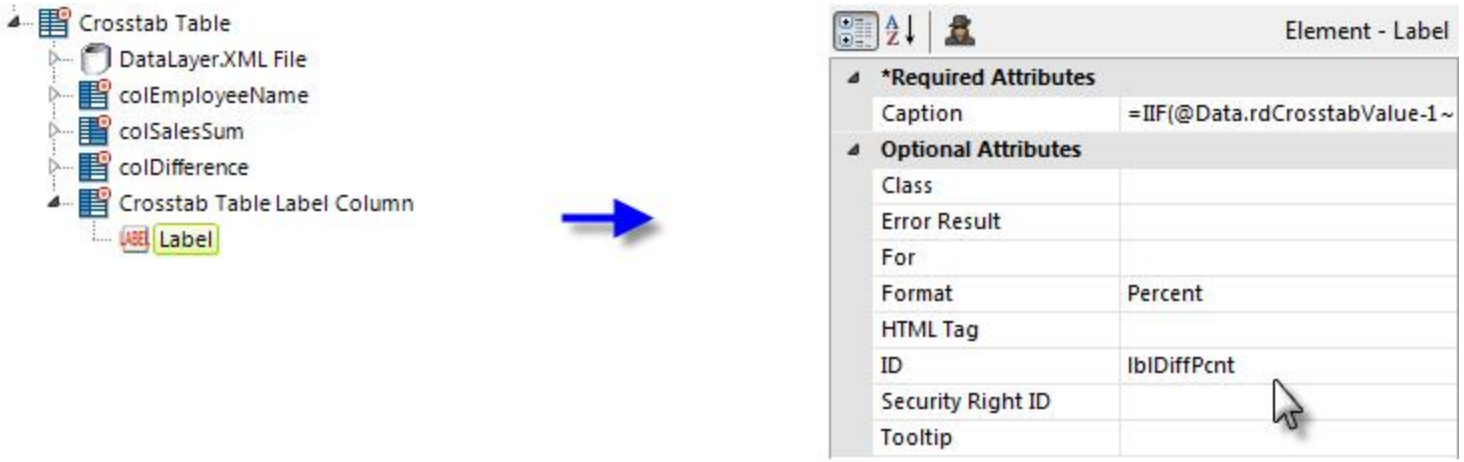
Attribute	Description
Show Tooltips	play options include *Percent*, *Value*, *All*, or the default, *None*.
Comparison Show Values	Enables display of the numerical difference in the crosstab cells. Display options include *Percent*, *Value*, *All*, or the default, *None*. The Format attribute can be used to format the displayed numbers.
Comparison Style	Specifies how differences will be indicated visually. Options include *None*, *ColorSpectrum*, and the default, *Arrows*.
Format	Specifies a format for values or percentages displayed. For more information, see *Format Data*.
High Color Value	When *ColorSpectrum* has been selected as the Comparison Style, specifies the color to be used for the highest value in the numeric range. Default: *Green*
Low Color Value	When *ColorSpectrum* has been selected as the Comparison Style, specifies the color to be used for the lowest value in the numeric range. Default: *Red*
Medium Color Value	When *ColorSpectrum* has been selected as the Comparison Style, specifies the color to be used for the value in the middle of the numeric range.

If the Crosstab Table element's **Draggable Columns** attribute has been set to *True*, and the Crosstab Comparison element is in use, when columns are dragged to change their order, their comparison values will be automatically updated.

Because the calculations made by this element do not affect the table's datalayer, it's not possible to summarize or sort the results.

Calculating Comparisons Using Tokens

The second approach to Value Column comparisons allows other calculations, such as comparisons to current dates or to external values, and is accomplished entirely within the **Label** element used to present the results.



In the example above, a **Crosstab Table Label Column** element ("colDifference") has been added with a **Label** element beneath it. The Label element's **Caption** attribute includes a formula that relies on the fact that Crosstab Table value columns can be individually addressed using the tokens @Data.rdCrosstabValue-0~, @Data.rdCrosstabValue-1~, ...@Data.rdCrosstabValue-*n*~.

To display the *difference* between two values, for example, the Label caption would use this formula:

```
=@Data.rdCrosstabValue-1~ - @Data.rdCrosstabValue-0~
```

And, we can add the same element again to display the *percentage difference* between two values. The child Label element beneath it would use this formula in its Caption attribute:

```
=IIF(@Data.rdCrosstabValue-1~ - @Data.rdCrosstabValue-0~ < 1, (@Data.rdCrosstabValue-1~ -
@Data.rdCrosstabValue-0~)/@Data.rdCrosstabValue-0~, (@Data.rdCrosstabValue-1~/@Data.rdCrosstabValue-0~) - 1
)
```

Both Label elements need to have their Format attributes set properly to display the numeric value correctly.

Employee Name	2013	2014	Diff Value	Diff Pcnt
Buchanan, Steven	$18,383.92	$30,716.47	$12,332.55	67.08%
Callahan, Laura	$22,240.12	$56,032.62	$33,792.50	151.94%
Davolio, Nancy	$35,764.51	$93,148.09	$57,383.58	160.45%
Dodsworth, Anne	$9,894.52	$26,310.39	$16,415.87	165.91%
Fuller, Andrew	$21,757.06	$70,444.14	$48,687.08	223.78%
King, Robert	$15,232.16	$60,471.19	$45,239.03	297.00%
Leverling, Janet	$18,223.96	$108,026.14	$89,802.18	492.77%
Peacock, Margaret	$49,945.12	$128,809.79	$78,864.67	157.90%
Suyama, Michael	$16,642.61	$43,126.37	$26,483.76	159.13%

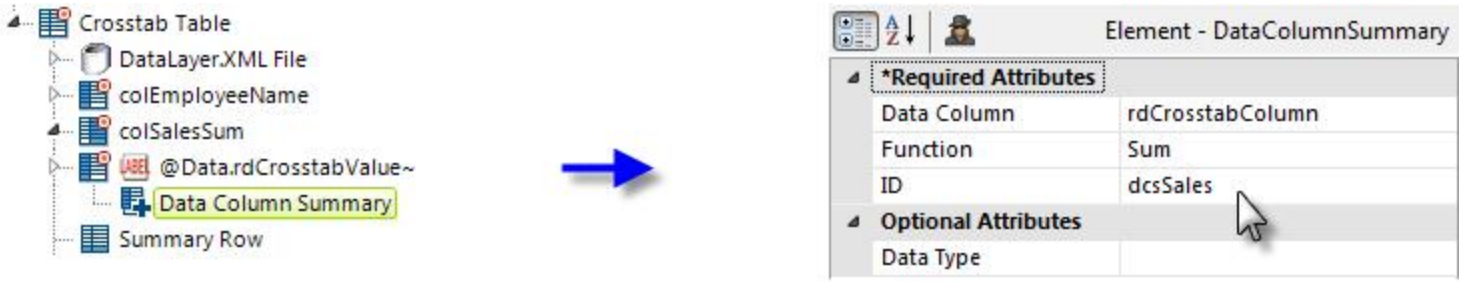
With both of these difference calculations included, the resulting Crosstab Table looks like the example shown above. Again, because the calculations do not affect the table's datalayer, it's not possible to summarize or sort the results.

Sorting and Summarizing Columns

As mentioned earlier, you can apply a Sort Filter element beneath your datalayer, before or after the Crosstab Filter, to sort the actual data. In addition, you can use sorting elements within the table columns.

The sorting and summary features for Crosstab Table Label Columns are functionally similar to those for Data Table columns: **Sort** and **Data Column Summary** child elements are added beneath them.

However, because of its special dynamic-column nature, the Crosstab Table Value Columns element has its own special sort element: the **Crosstab Value Column Sort** element is used to sort by value columns.



When using a standard Data Column Summary element to summarize crosstab Value columns, set its **Data Column** attribute to *rdCrosstabColumn*, as shown above.

A standard **Summary Row** element is added to create a row across the bottom of the Crosstab Table to display the summarized value for each column with a Data Column Summary element.

Employee Name	2013	2014	2015
Buchanan, Steven	$18,383.92	$30,716.47	$19,691.90
Callahan, Laura	$22,240.12	$56,032.62	$48,589.55
Davolio, Nancy	$35,764.51	$93,148.09	$85,797.96
Dodsworth, Anne	$9,894.52	$26,310.39	$41,103.16
Fuller, Andrew	$21,757.06	$70,444.14	$74,336.55
King, Robert	$15,232.16	$60,471.19	$48,864.88
Leverling, Janet	$18,223.96	$108,026.14	$76,562.74
Peacock, Margaret	$49,945.12	$128,809.79	$54,135.94
Suyama, Michael	$16,642.61	$43,126.37	$14,144.16
Total:	$208,083.98	$617,085.20	$463,226.84

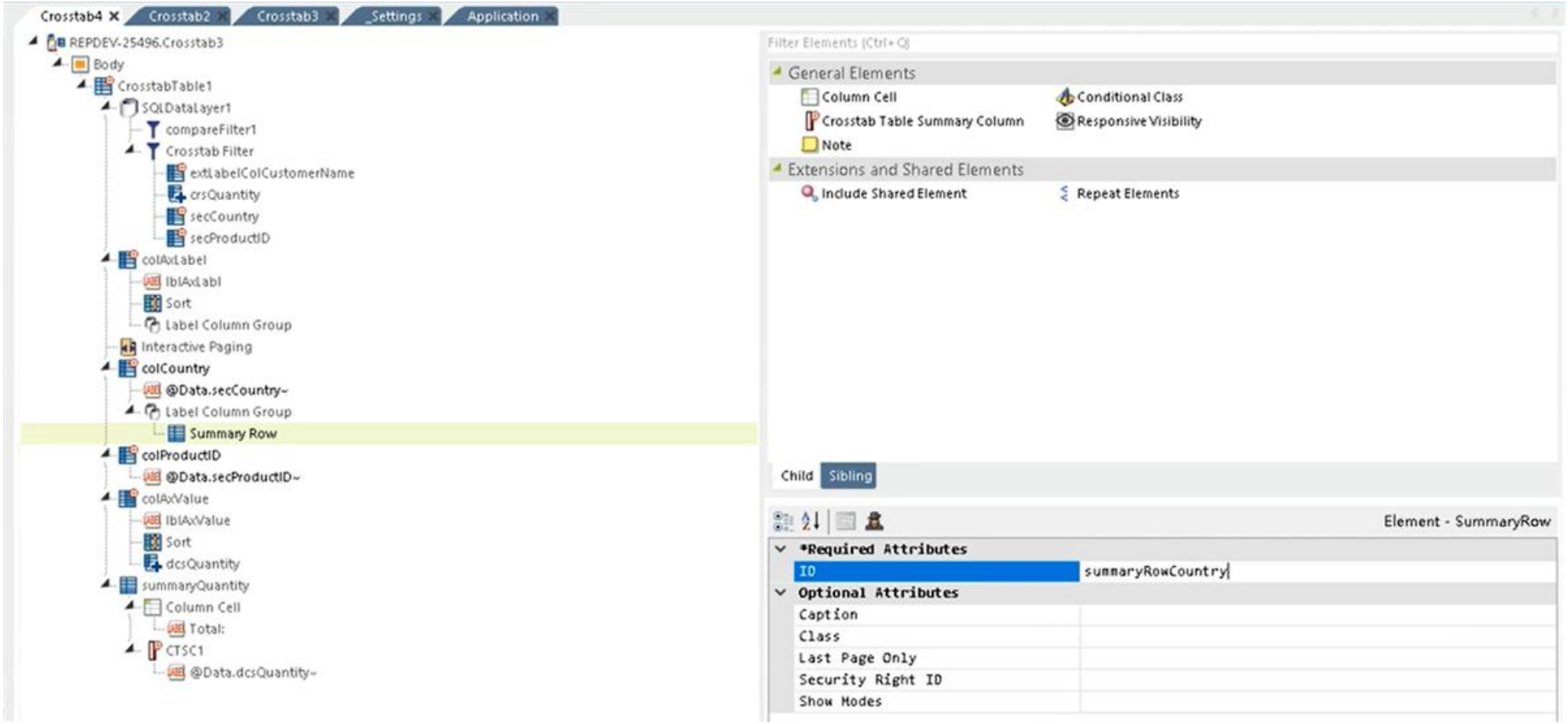
And the results look like the example shown above. By default, the Summary Row element's columns inherit the formatting and alignment of the value column above them.

For more precise control of the summary row, you may use **Column Cell** elements (beneath Crosstab Table Label Columns) and/or **Crosstab Table Summary Column** elements (beneath Crosstab Table Value Columns) to construct the row. These allow you to span rows and columns and set custom alignment and formatting.

Summarizing Groups

You can also show summary values in Label Column Group Summary Rows. This feature works like the one mentioned above, but instead of creating a row at the bottom of the Crosstab Table that displays the summarized value for each column, it allows you to total any Data Columns you have *grouped* together. For information on grouping Crosstabs, see "Working with the Crosstab Filter" on page 557.

1. Begin by adding a **Summary Row** element to your Secondary Crosstab Label Columns. In this example, we're adding a Summary Row to the "colCountry" Secondary Crosstab Label Column and giving it the ID "summaryRowCountry":



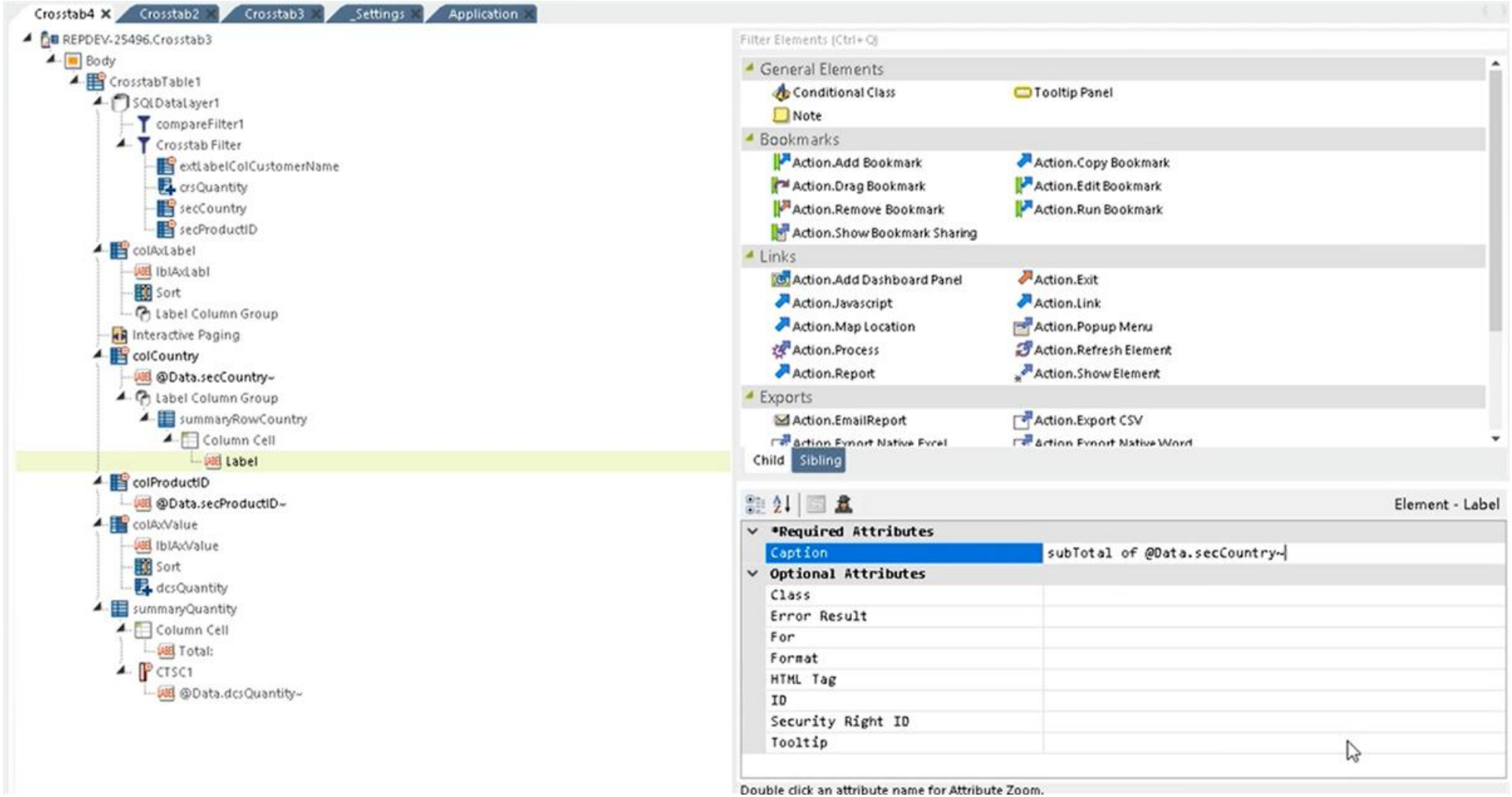
2. **Save** and **refresh** your report. Info displays the Crosstab Table with a Summary Row for the *Country* column:

ProductName by Country on Sum of Quantity

Region	Country	Product ID	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
Asia-Pacific	Australia	1	798										
		6		3,306									
			798	3,306									
	China	6		9,738									
		23			5,034								
				9,738	5,034								
	Japan	6		6,232									
				6,232									
	Malaysia	1	196										
		2				656							
			196			656							

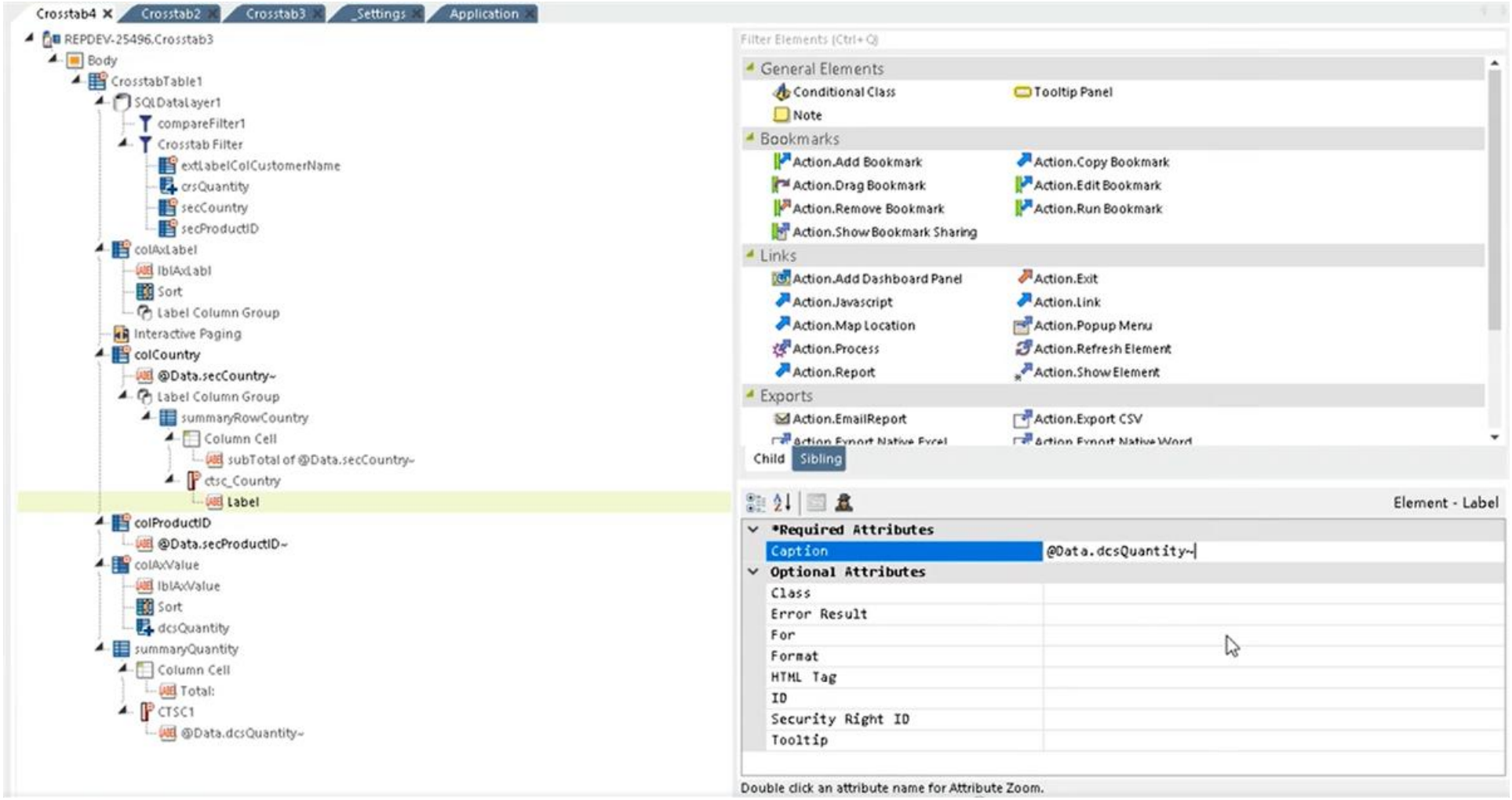
3. You can adjust the Summary Row span by adding a **Column Cell** element. In this example, we're adjusting the Column Span attribute to 2.

4. Then, add a **Label** element to the Column Cell element and add a Caption based on the Data Column you wish to tally, like below:


Crosstab4
Crosstab2
Crosstab3
_Settings
Application
REPDEV-25496.Crosstab3
Body
CrosstabTable1
SQLDataLayer1
compareFilter1
Crosstab Filter
extLabelColCustomerName
crsQuantity
secCountry
secProductID
colAxLabel
lblAxLabl
Sort
Label Column Group
Interactive Paging
colCountry
@Data.secCountry~
Label Column Group
summaryRowCountry
Column Cell
Label
colProductID
@Data.secProductID~
colAxValue
lblAxValue
Sort
dcsQuantity
summaryQuantity
Column Cell
Total:
CTSC1
@Data.dcsQuantity~
Filter Elements (Ctrl+Q)
General Elements
Conditional Class
Tooltip Panel
Note
Bookmarks
Action.Add Bookmark
Action.Copy Bookmark
Action.Drag Bookmark
Action.Edit Bookmark
Action.Remove Bookmark
Action.Run Bookmark
Action.Show Bookmark Sharing
Links
Action.Add Dashboard Panel
Action.Exit
Action.Javascript
Action.Link
Action.Map Location
Action.Popup Menu
Action.Process
Action.Refresh Element
Action.Report
Action.Show Element
Exports
Action.EmailReport
Action.Export CSV
Child
Sibling
Element - Label
*Required Attributes
Caption
subTotal of @Data.secCountry~
Optional Attributes
Class
Error Result
For
Format
HTML Tag
ID
Security Right ID
Tooltip
Double click an attribute name for Attribute Zoom.

5. Next, add a **Crosstab Table Summary Column** to your Summary Row element and give it an ID.

6. Below the Crosstab Table Summary Column, add a **Label** element and assign a Caption, based on the Data Column you are summarizing:



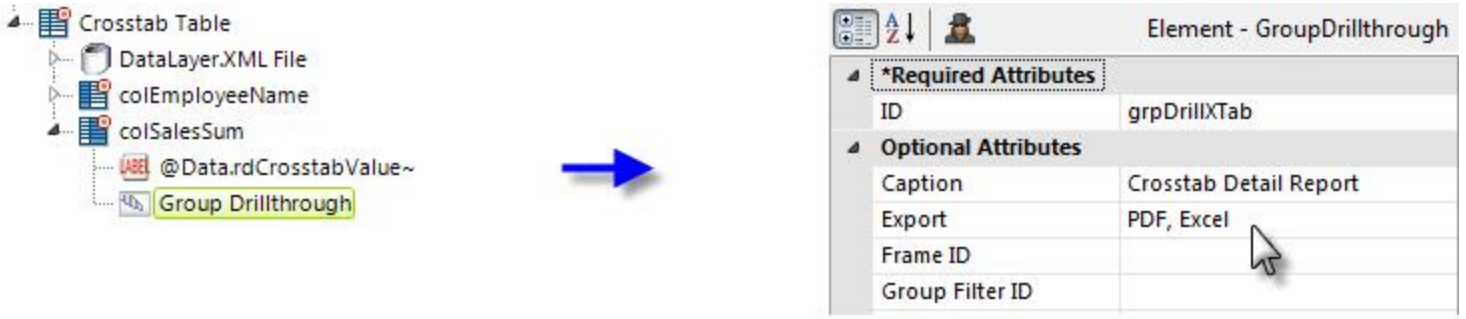
7. **Save** and **refresh** your report. Info displays the Crosstab Table with the summarized column(s):

ProductName by Country on Sum of Quantity

Region	Country	Product ID	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
Asia-Pacific	Australia	1	798										
		6		3,306									
	subTotal of Australia		798	3306									
	China	6		9,738									
		23			5,034								
	subTotal of China			9738	5034								
	Japan	6		6,232									
	subTotal of Japan			6232									
	Malaysia	1	196										
		2				656							
	subTotal of Malaysia		196			656							
Europe, Middle East, Africa	Belgium	1	854										
		13					264						
		20						2,044					
		23			1,572								
	subTotal of Belgium		854		1572		264	2044					
	Poland	6		5,517									
		8							1,200				
	subTotal of Poland			5517					1200				
	Russia	1	343										
		13					1,055						

Drillthrough to Column Detail

A Crosstab Table does a great job of aggregating data and displaying it in a manner that makes it easy to see its progression or differences. There are times, however, when it's useful to be able to examine the detail data that the crosstab filter used to create the value in each column cell. The **Group Drillthrough** element provides this functionality.



In the example above, a **Group Drillthrough** element has been added beneath a Crosstab Table Value Columns element. All of the element's attributes, other than ID, are optional but the example shows a custom caption and export button selections have been entered. More information about the attributes can be found in the element's Element Reference page.

Employee Name	2013	2014	2015
Buchanan, Steven	$18,383.92	$30,716.47	$19,691.90
Callahan, Laura	$22,240.12	$56,032.62	$48,589.55
Davolio, Nancy	$35,764.51	$93,148.09	$85,797.96

The resulting Crosstab Table is shown above. When the cursor is hovered over a value cell, a drill-through icon is displayed.

Back

Crosstab Detail Report

Drillthrough on: OrderYear="2015", FullName="Buchanan, Steven"

Export: Excel PDF

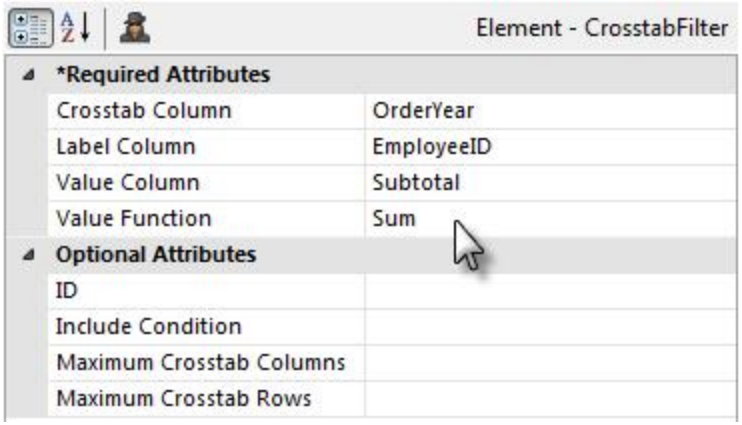
OrderID	EmployeeID	CustomerID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName	ShipAddress
10812	5	REGGC	1/2/2015 12:00:00 AM	1/30/2015 12:00:00 AM	1/12/2015 12:00:00 AM	1	59.78	Reggiani Caseifici	Strada Provinciale 124
10823	5	LILAS	1/9/2015 12:00:00 AM	2/6/2015 12:00:00 AM	1/13/2015 12:00:00 AM	2	163.97	LILA-Supermercado	Carrera 52 con Ave. Bolívar #65-98 Llano Largo
10841	5	SUPRD	1/20/2015 12:00:00 AM	2/17/2015 12:00:00 AM	1/29/2015 12:00:00 AM	2	424.3	Suprêmes délices	Boulevard Tirou #255
10851	5	RICAR	1/26/2015 12:00:00 AM	2/23/2015 12:00:00 AM	2/2/2015 12:00:00 AM	1	160.55	Ricardo Adocicados	Av. Copacabana #267
10866	5	BERGS	2/3/2015 12:00:00 AM	3/3/2015 12:00:00 AM	2/12/2015 12:00:00 AM	1	109.11	Berglunds snabbköp	Berguvsvägen 8
10869	5	SEVES	2/4/2015 12:00:00 AM	3/4/2015 12:00:00 AM	2/9/2015 12:00:00 AM	1	143.28	Seven Seas Imports	90 Wadhurst Rd.
10870	5	WOLZA	2/4/2015 12:00:00 AM	3/4/2015 12:00:00 AM	2/13/2015 12:00:00 AM	3	12.04	Wolski Zajazd	ul. Filtrowa 68

When the icon is clicked, a report containing all of the relevant detail data, like the example shown above, is automatically generated and displayed.

The Group Drillthrough element's attributes allow you to customize the icon image that appears in the Crosstab Table and certain aspects of the detail report. You can also include **Drillthrough Column** elements beneath the Group Drillthrough element; they allow you to set the number columns that will appear in the report and to customize their appearance.

Working with the Crosstab Filter

A **Crosstab Filter** element is required to convert, or "pivot", data retrieved into a datalayer for use in Crosstab Tables. In configuring this filter you specify three columns in the datalayer: the Crosstab Column, the Label Column, and the Value Column. You must also choose an aggregate function to apply to records from the Value Column.

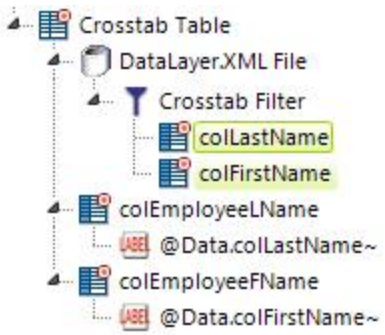


The filter groups the data based on the designated Crosstab Column. If the Crosstab Column contains numerous distinct records, many Crosstab Value columns may be generated and the table can become quite wide. This can be controlled using the optional **Maximum Crosstab Columns** attribute, which limits the number of Crosstab Value columns that appear in the table.

Similarly, the **Maximum Crosstab Rows** attribute sets the maximum number of crosstab *rows* created by the Crosstab Filter. In some cases this number should be limited so that the table does not become too large in memory. The default limit is 10,000 rows.

Extra Crosstab Label Columns

One Crosstab Table Label Column may not be sufficient to display all the necessary information in the table.



Last Name	First Name	2013	2014	2015
Buchanan	Steven	$18,383.92	$30,716.47	$19,691.90
Callahan	Laura	$22,240.12	$56,032.62	$48,589.55
Davolio	Nancy	$35,764.51	$93,148.09	$85,797.96
Dodsworth	Anne	$9,894.52	$26,310.39	$41,103.16
Fuller	Andrew	$21,757.06	$70,444.14	$74,336.55
King	Robert	$15,232.16	$60,471.19	$48,864.88
Leverling	Janet	$18,223.96	$108,026.14	$76,562.74
Peacock	Margaret	$49,945.12	$128,809.79	$54,135.94
Suyama	Michael	$16,642.61	$43,126.37	$14,144.16

As shown above, developers can add **Extra Crosstab Label Column** elements beneath a Crosstab Filter element in order to add additional columns from the datalayer as Label columns.

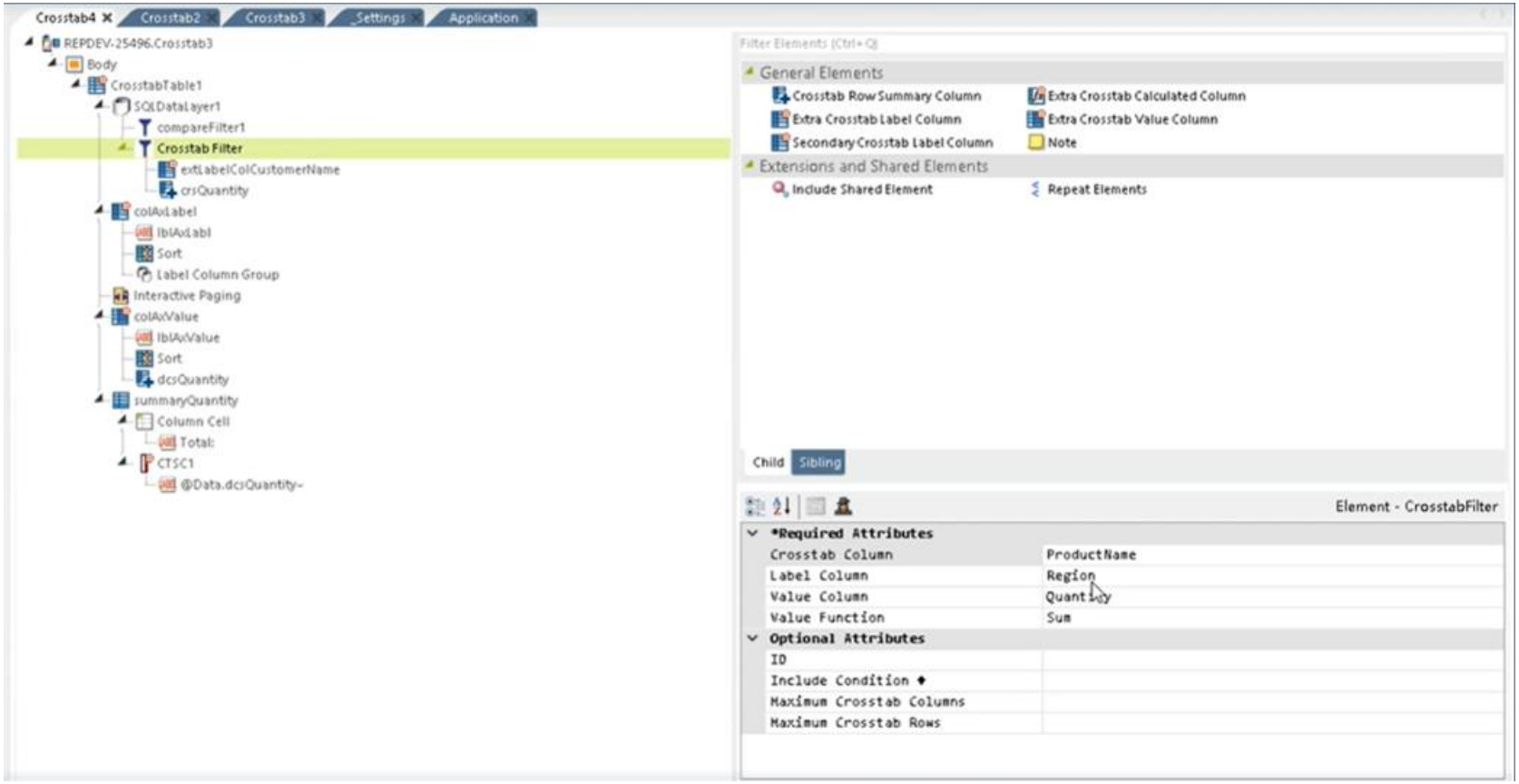
Extra Crosstab Label Columns are for display only and do not support grouping.

To display the data for these extra label columns, an @Data token is used but the token identifier matches the ID of the Extra Crosstab Label Column element, not the datalayer column it represents, as shown above.

Secondary Crosstab Label Columns

By adding the **Secondary Crosstab Label Column** element (instead of Extra Crosstab Label Columns) to your Crosstab Filter, you can group Value Columns and Label Columns together.

In this example, the Crosstab Filter is grouped by the Label Column *Region*:

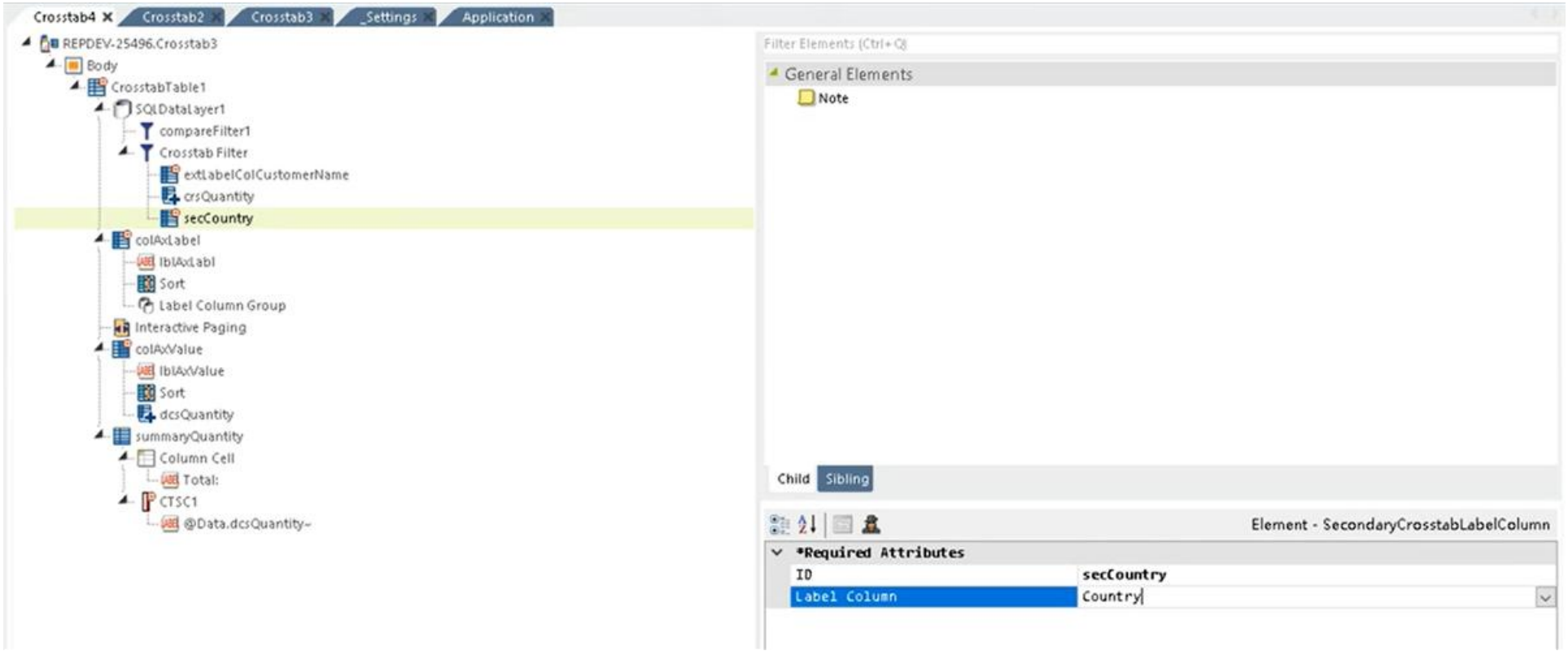


Here's what the Crosstab Table looks like, as is:

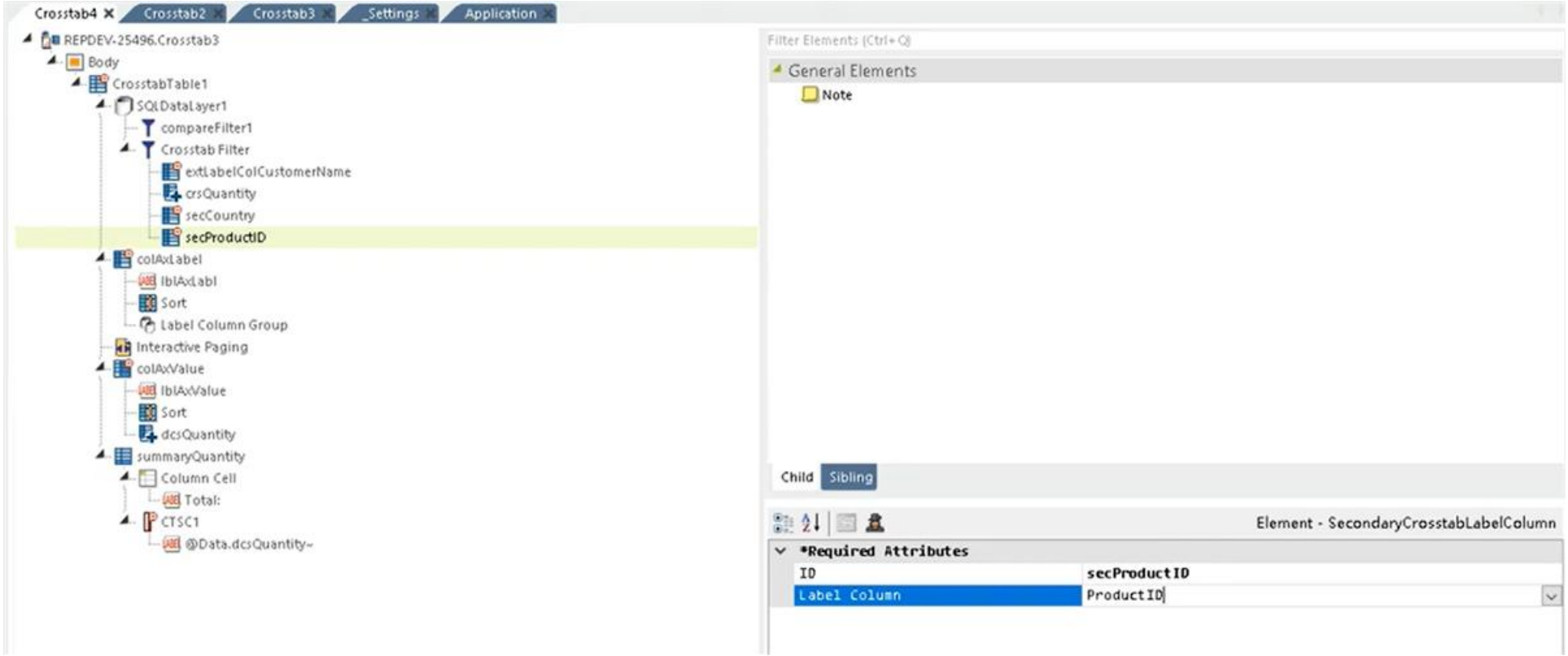
ProductName by Country on Sum of Quantity

Region	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
Asia-Pacific	994	19,276	5,034	656							
Europe, Middle East, Africa	1,624	15,255	1,572		1,319	2,044	1,200	3,355	1,200	3,355	615
Latin America		27,684					22,761				
North America	665	3,419		1,779	79	9,162	2,894	3,824		39,928	
Total:	3283	65634	6606	2435	1398	11206	26855	7179	1200	43283	615

1. First, add a **Secondary Crosstab Label Column** and give it an ID and asign a Label Column. In this example, we're adding "secCountry" with the Label Column "Country":

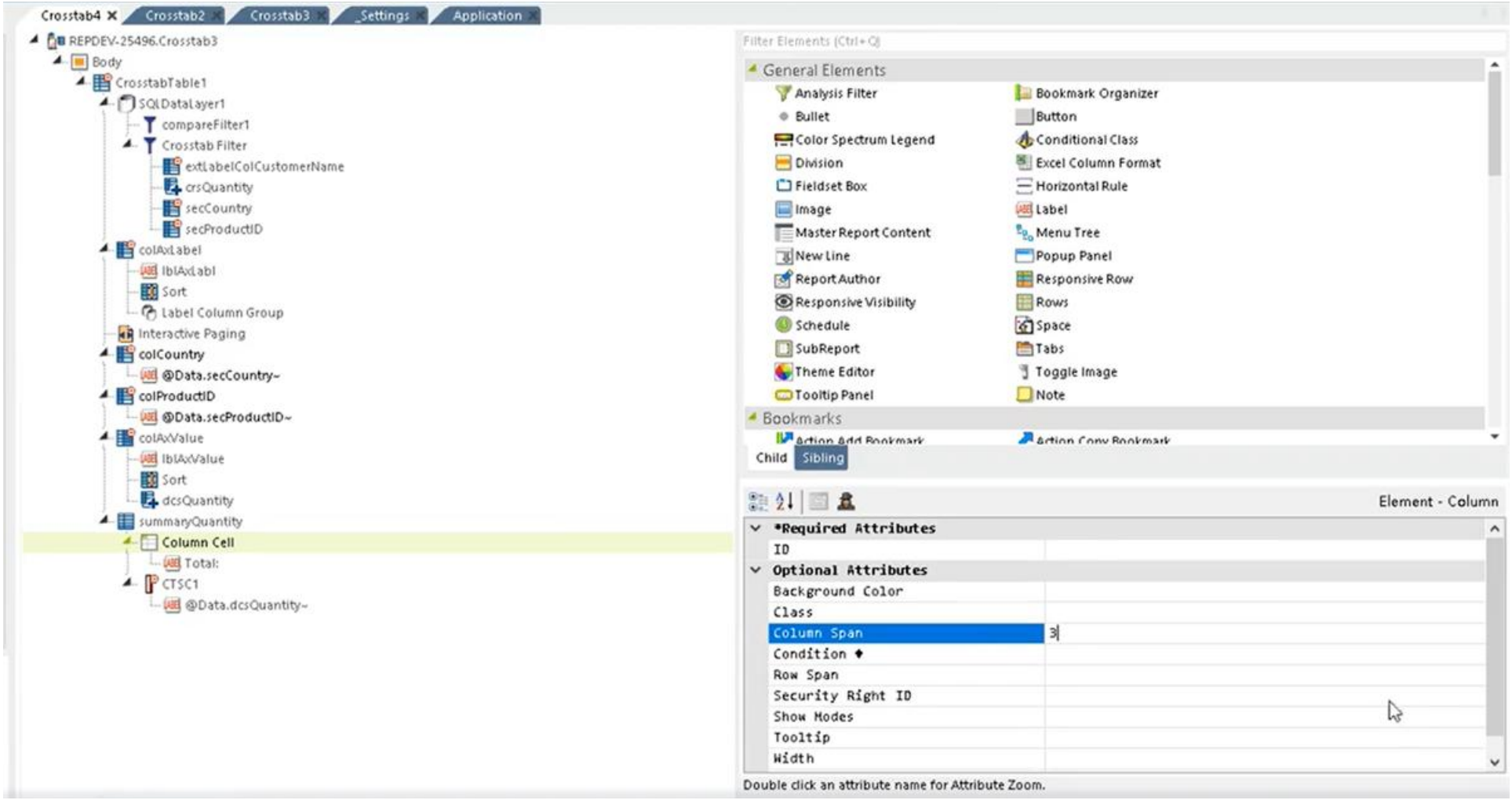


2. Then, add another **Secondary Crosstab Label Column** and give it an ID and assign a Label Column. In this example we're adding "secProductID" with the Label Column "ProductID":



3. Add two **Crosstab Table Label Columns** to the Crosstab Filter that reflect the Secondary Crosstab Label Columns. In this example, we're adding a Crosstab Table Label Column with the ID "colCountry", Column Header titled "Country", and a Label element captioned "@Data.secCountry~" and another Crosstab Table Label Column with the ID "colProductID", Column Header titled "Product ID", and a Label element captioned "@Data.secProductID~".

4. Adjust the Column Span of the Summary Column to reflect the number of Label Columns you are grouping together. In this example, we're adjusting the span to 3 columns:

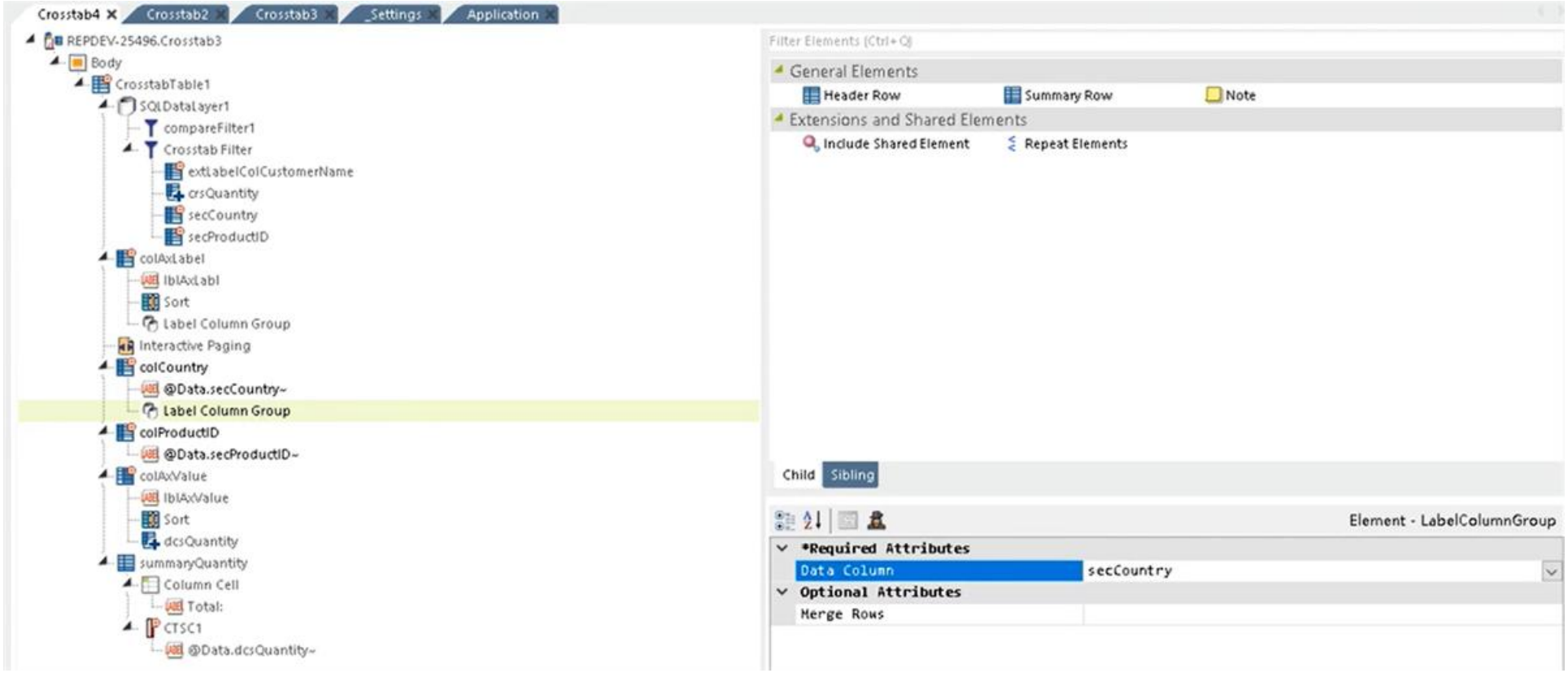


5. Select **Save** and **refresh** your report. Info displays the newly added *Country*, and *ProductID* Secondary Crosstab Label Columns:

ProductName by Country on Sum of Quantity

Region	Country	Product ID	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
Asia-Pacific	Australia	1	798										
	Australia	6		3,306									
	China	6		9,738									
	China	23			5,034								
	Japan	6		6,232									
	Malaysia	1	196										
	Malaysia	2				656							
Europe, Middle East, Africa	Belgium	1	854										
	Belgium	13					264						
	Belgium	20						2,044					
	Belgium	23			1,572								
	Poland	6		5,517									
	Poland	8							1,200				
	Russia	1	343										
	Russia	13					1,055						
	Saudi Arabia	6		9,738									
	Spain	1	427										
	Spain	25								3,355			
	Spain	26									1,200		
	Spain	28										3,355	
Total:			3283	65634	6606	2435	1398	11206	26855	7179	1200	43283	615

6. You can go one step further and add a **Label Column Group** element for each of the Data Columns you want to group. In this example, we're adding a Label Column Group for "Region" and "secCountry":



You can also change the Merge Rows attribute to "True" (default value is "False") to merge the cells grouped together. You must do this for each Label Column Group you add.

7. Select **Save** and **refresh** your report. Info groups the *Region* and *Country* columns:

ProductName by Country on Sum of Quantity

Region	Country	Product ID	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
Asia-Pacific	Australia	1	798										
		6		3,306									
	China	6		9,738									
		23			5,034								
	Japan	6		6,232									
	Malaysia	1	196										
		2				656							
Europe, Middle East, Africa	Belgium	1	854										
		13					264						
		20						2,044					
		23			1,572								
	Poland	6		5,517									
		8							1,200				
	Russia	1	343										
		13					1,055						
	Saudi Arabia	6		9,738									
	Spain	1	427										
		25								3,355			
		26									1,200		
		28										3,355	
Total:			3283	65634	6606	2435	1398	11206	26855	7179	1200	43283	615

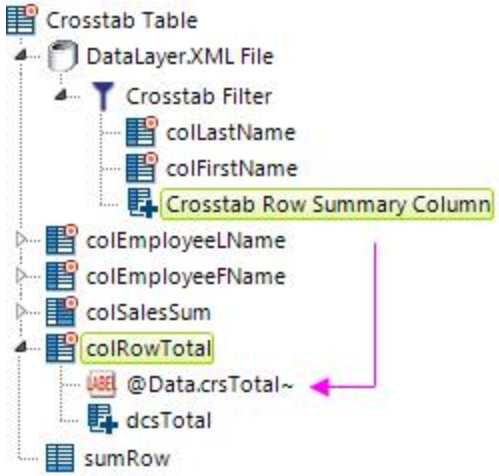
Here's what your Crosstab Table looks like if you *merged* the cells:

ProductName by Country on Sum of Quantity

Region	Country	Product ID	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
Asia-Pacific	Australia	1	798										
		6		3,306									
	China	6		9,738									
		23			5,034								
	Japan	6		6,232									
	Malaysia	1	196										
		2				656							
Europe, Middle East, Africa	Belgium	1	854										
		13					264						
		20						2,044					
		23			1,572								
	Poland	6		5,517									
		8							1,200				
	Russia	1	343										
		13					1,055						
	Saudi Arabia	6		9,738									
	Spain	1	427										
		25								3,355			
		26									1,200		
		28										3,355	
Total:			3283	65634	6606	2435	1398	11206	26855	7179	1200	43283	615

Summarizing Across a Row

In "Comparing, Sorting, and Summarizing Columns" on page 542 we saw that you can summarize crosstab *column* data (a vertical summary) and display the totals in a summary row at the bottom. You can also summarize crosstab *row* data (a horizontal summary) and show row totals in a separate column.



Last Name	First Name	2013	2014	2015	Total
Buchanan	Steven	$18,383.92	$30,716.47	$19,691.90	$68,792.29
Callahan	Laura	$22,240.12	$56,032.62	$48,589.55	$126,862.29
Davolio	Nancy	$35,764.51	$93,148.09	$85,797.96	$214,710.56
Dodsworth	Anne	$9,894.52	$26,310.39	$41,103.16	$77,308.07
Fuller	Andrew	$21,757.06	$70,444.14	$74,336.55	$166,537.75
King	Robert	$15,232.16	$60,471.19	$48,864.88	$124,568.23
Leverling	Janet	$18,223.96	$108,026.14	$76,562.74	$202,812.84
Peacock	Margaret	$49,945.12	$128,809.79	$54,135.94	$232,890.85
Suyama	Michael	$16,642.61	$43,126.37	$14,144.16	$73,913.14
Total:		$208,083.98	$617,085.20	$463,226.84	$1,288,396.02

As shown above, the **Crosstab Row Summary Column** element can be added beneath a Crosstab Filter element to generate new columns in the filtered datalayer that contain aggregations of every value in each crosstab row.

You can specify an aggregation of *Average, AverageOfAllRows, Count, CountOfAllRows,Max (v12.7), Min (v12.7), StdDev,* or *Sum.* For example, to count the number of value columns, set this attribute to *Count*; in the image above there are value columns for 2013, 2014, and 2015, so the value calculated will be 3.

Additional **Crosstab Table Label Column** elements, such as "colRowTotal" above, can be added as needed to display the summarized results. The @Data token used to retrieve the data uses the ID of the Crosstab Row Summary Column element. So if, in the example above, the Crosstab Row Summary Column element was given an ID of "crsTotal", then *@Data.crsTotal~* would be used in a Label element to display the data.

Similarly, a **Data Column Summary** element ("dcsTotal") can be added, configured to aggregate the "crsTotal" column, in order to total the new column vertically and include it in the Summary row at the bottom of the table.

Crosstab Filters can also be used with the datalayers that provide data for charts and gauges, however, they're designed for use with Crosstab Table elements and may not provide full functionality with other elements.

The Include Condition Attribute

The Crosstab Filter element has an **Include Condition** attribute and if the value of this attribute is left blank or contains a formula that evaluates to *True*, the Crosstab Filter element is applied to the datalayer. If the value evaluates to *False*, the filter is ignored and does not affect the datalayer. This powerful feature allows developers to dynamically determine if the datalayer will be manipulated or not.

Planning the Tutorial Crosstab Table

Before adding elements to a definition, it's a good idea to first *plan* the table (or chart) that you're trying to produce. A simple sketch can often provide clarity and save you from having to re-work your definition later. Look at the Crosstab Table layout model shown below:

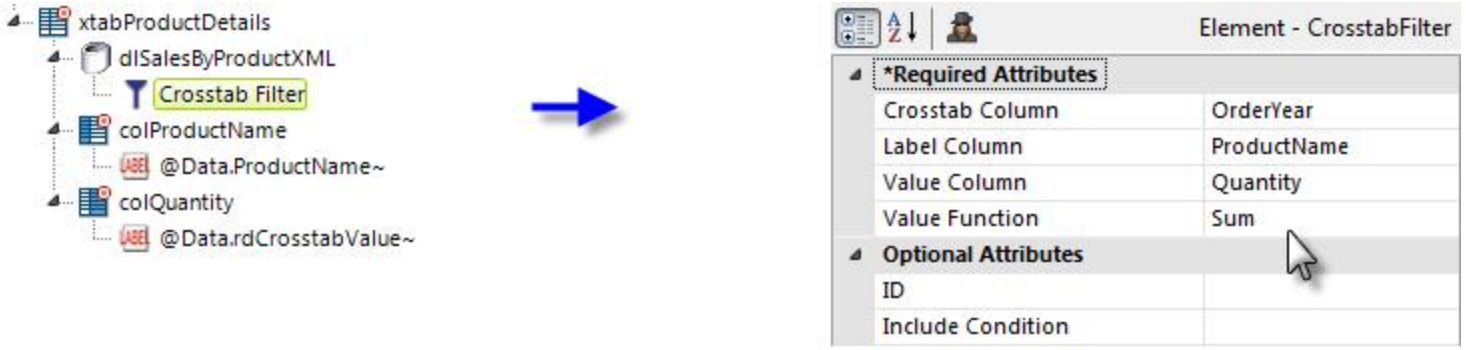
		Year 1			Year 2			Year 3			Total		
Supplier	**Product Name**	**Total Sales**	**Units Sold**	**Avg. Unit Price**	**Total Sales**	**Units Sold**	**Avg. Unit Price**	**Total Sales**	**Units Sold**	**Avg. Unit Price**	**Total Sales**	**Units Sold**	**Avg. Unit Price**
Company 1	Product 1	$9,999.99	999	$99.99									
	Product 2	$9,999.99	999	$99.99									
Company 2	Product 1	$9,999.99	999	$99.99									
	Product 2	$9,999.99	999	$99.99									
Totals:		$9,999,999.99	9,999	$999.99									

The table is limited to three years of data, groups products by company, provides summarized data for each year in several categories and for all products, and summarizes across all years as well. With this layout in hand, we can proceed to build the table.

The following examples demonstrate how to use a variety of elements to produce this table. The complete, downloadable definition for this Crosstab Table is available in the Crosstab Table Sample Application on DevNet.

Building the Basic Table Structure

As discussed in "Working with the Crosstab Filter" on page 557, the **Crosstab Filter** element determines the basic structure of the table:



The definition starts with a **Crosstab Table** element (no attributes except an element ID need be set), a datalayer to retrieve the data, and a **Crosstab Filter** element to format the data. In the filter attributes shown above, the table will have a label column for the Product Names, be grouped horizontally (crosstab) by Order Year, and the initial values will be the sum of the order Quantity.

A **Crosstab Table Label Column** element ("colProductName") and a **Crosstab Table Value Columns** element ("colQuantity") are added to the definition to display the data.

A special token is needed to access the Value column data: *@Data.rdCrosstabValue~*.

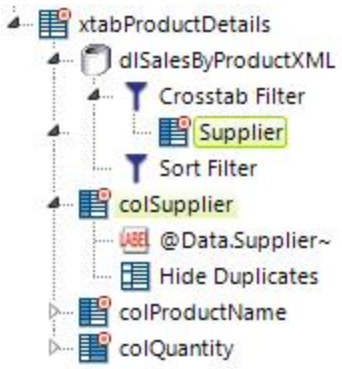
The token identifier uses the reserved word "rdCrosstabValue" and the token is used in the Caption attribute of a Label element beneath the Crosstab Table Value Columns element.

		Year 1			Year 2			Year 3		
Supplier	Product Name	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price
Company 1	Product 1	$9,999.99	999	$99.99						
	Product 2	$9,999.99	999	$99.99						
Company 2	Product 1	$9,999.99	999	$99.99						
	Product 2	$9,999.99	999	$99.99						

Referring to our crosstab layout model, the seven elements added so far provide the portions highlighted in yellow above.

Product Name	Units Sold	Units Sold	Units Sold
Aniseed Syrup	30	190	108
Chef Anton's Cajun Seasoning	107	264	82
Chef Anton's Gumbo Mix	129	19	150
Genen Shouyu	25	97	22
Grandma's Boysenberry Spread	36	100	165
Gula Malacca	138	396	67
Louisiana Fiery Hot Pepper Sauce	155	490	100
Louisiana Hot Spiced Okra	30	208	48
Northwoods Cranberry Sauce	140	114	118
Original Frankfurter grune Sosse	63	432	296
Sirop d'erable	42	396	207
Vegie-spread	109	189	147

If we preview our definition at this point, the Crosstab Table looks like the example shown above. Now let's add another Label column to display the names of the suppliers.



As discussed in "Working with the Crosstab Filter" on page 557, developers can add **Extra Crosstab Label Column** elements beneath a Crosstab Filter element to add additional columns from the datalayer as Label columns. As shown above, one of these elements is added beneath the Crosstab Filter and set to use the CompanyName datalayer column to create the "Supplier" column. Extra Crosstab Label Columns are for display only and do not support grouping.

A **Sort Filter** is added to the datalayer and set to sort on the new Supplier column to get things into order by Supplier.

To display the new column, another **Crosstab Table Label Column** element ("colSupplier") and a **Label** element to display the data are added. Finally, adding a **Hide Duplicates** element results in a hierarchical look by hiding the supplier name in consecutive rows.

		Year 1			Year 2			Year 3		
Supplier	Product Name	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price
Company 1	Product 1	$9,999.99	999	$99.99						
	Product 2	$9,999.99	999	$99.99						
Company 2	Product 1	$9,999.99	999	$99.99						
	Product 2	$9,999.99	999	$99.99						

The result is the addition of the Supplier column, highlighted in yellow above in the crosstab layout model.

Supplier	Product Name	Units Sold	Units Sold	Units Sold
Exotic Liquids	Aniseed Syrup	30	190	108
Forets d'erables	Sirop d'erable	42	396	207
Grandma Kelly's Homestead	Grandma's Boysenberry Spread	36	100	165
	Northwoods Cranberry Sauce	140	114	118
Leka Trading	Gula Malacca	138	396	67
Mayumi's	Genen Shouyu	25	97	22
New Orleans Cajun Delights	Chef Anton's Cajun Seasoning	107	264	82
	Chef Anton's Gumbo Mix	129	19	150
	Louisiana Fiery Hot Pepper Sauce	155	490	100
	Louisiana Hot Spiced Okra	30	208	48
Pavlova, Ltd.	Vegie-spread	109	189	147
Plutzer Lebensgrossmarkte	Original Frankfurter grune Sosse	63	432	296

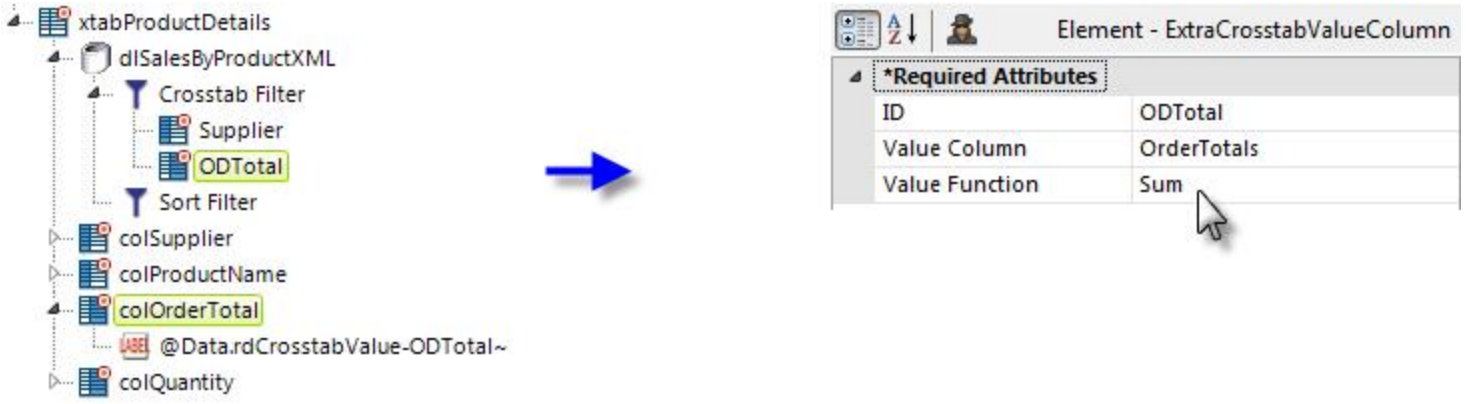
And, in a Preview, the table now looks like the image shown above.

You will need to apply formatting to various elements, such as Crosstab Table Value Columns and Column Cells, to align the data properly. This example uses the *ThemeAlignRight* style class to do this.

The **Secondary Crosstab Label Column** element works like the Extra Crosstab Label Column element, except you can use this element to group Value Columns and Label Columns together. For more information, see "Working with the Crosstab Filter" on page 557.

Adding Extra Crosstab Values

By default, the **Crosstab Filter** produces a single value for each column and row combination. In order to generate additional values for the Total Sales data for our Crosstab Table, we need to use an **Extra Crosstab Value Column** element.

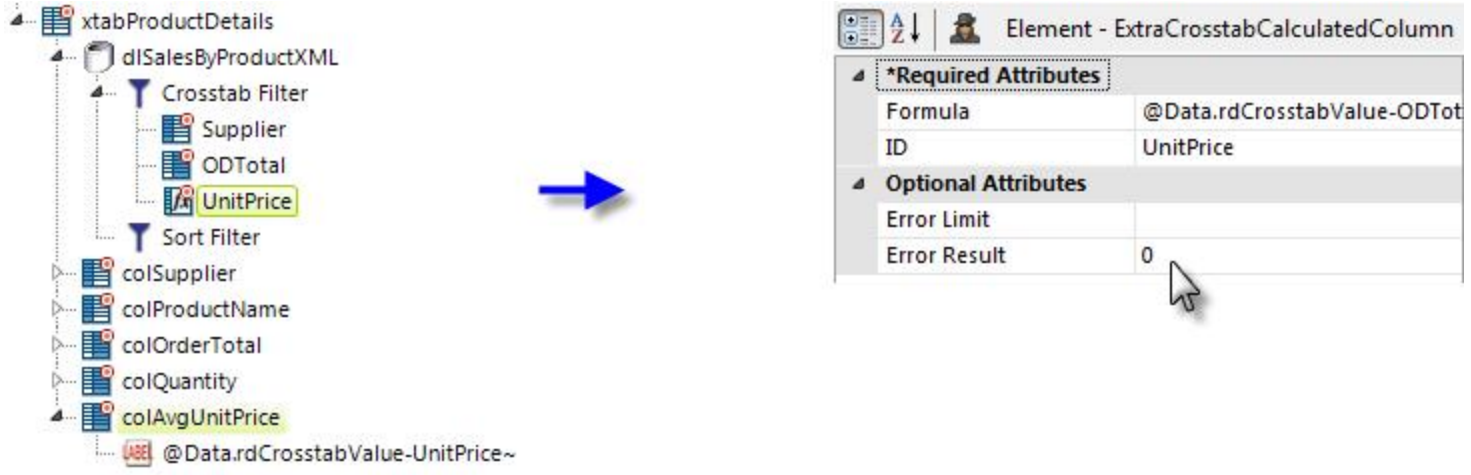


As shown above, this element simply identifies a column from the datalayer ("OrderTotals") and the aggregation to be applied to it.

A **Crosstab Table Value Columns** element ("colOrderTotal") is also added to display the values.

A special token is needed to access that data: *@Data.rdCrosstabValue-ODTotal~*.

The token identifier combines the reserved word "rdCrosstabValue", then a dash "-", and then the ID of the element beneath the Crosstab Filter that represents the value column ("ODTotal"). This token is used in the **Caption** attribute of a **Label** element beneath the Crosstab Table Value Columns element we added.



Adding "HTML Attribute Params" to your Crosstab Table enables you to apply your application to work with other frameworks or libraries easier. Depending on the type of HTML Attribute Params you add, there will be different parameters available to use, or you can define your own parameters. If an attribute was set in both Element and HTML Attribute Params, the one set in the HTML Attribute Params will be ignored.

Next, we'll repeat the previous steps in order to add an Average Unit Price column, this time using an **Extra Crosstab Calculated Column** element. In the example shown above, the average unit price is calculated using this element. The complete formula in this example is:

```
@Data.rdCrosstabValue-ODTotal~/@Data.rdCrosstabValue~
```

which divides the Order Total by the Quantity. The Order Total value is accessed using the same token discussed earlier and used to display the order total values; while the Quantity value is accessed using a similar token representing the original Value column defined in the Crosstab Filter.

		Year 1			Year 2			Year 3		
Supplier	Product Name	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price
Company 1	Product 1	$9,999.99	999	$99.99						
	Product 2	$9,999.99	999	$99.99						
Company 2	Product 1	$9,999.99	999	$99.99						
	Product 2	$9,999.99	999	$99.99						

The elements added and configured in this step add the columns highlighted above.

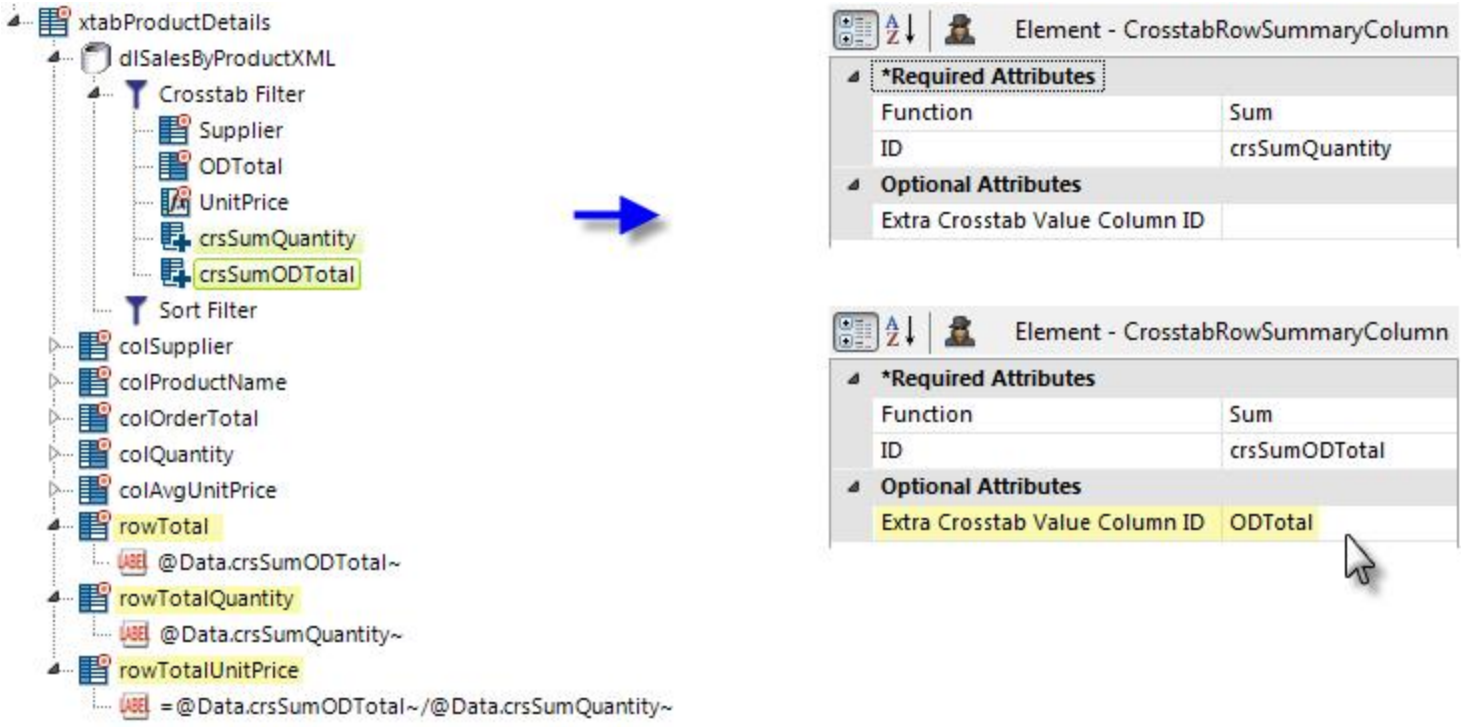
Supplier	Product Name	Total Sales	Units Sold	Avg Price	Total Sales	Units Sold	Avg Price	Total Sales	Units Sold	Avg Price
Exotic Liquids	Aniseed Syrup	$240.00	30	$8.00	$1,724.00	190	$9.07	$1,080.00	108	$10.00
Forets d'erables	Sirop d'erable	$1,113.00	42	$26.50	$9,091.50	396	$22.96	$5,261.10	207	$25.42
Grandma Kelly's Homestead	Grandma's Boysenberry Spread	$720.00	36	$20.00	$2,500.00	100	$25.00	$3,917.00	165	$23.74
	Northwoods Cranberry Sauce	$3,920.00	140	$28.00	$4,260.00	114	$37.37	$4,592.00	118	$38.92
Leka Trading	Gula Malacca	$2,042.13	138	$14.80	$6,737.94	396	$17.02	$1,135.88	67	$16.95
Mayumi's	Genen Shouyu	$310.00	25	$12.40	$1,474.83	97	$15.20	$363.00	22	$16.50
New Orleans Cajun Delights	Chef Anton's Cajun Seasoning	$1,851.52	107	$17.30	$5,214.88	264	$19.75	$1,501.50	82	$18.31
	Chef Anton's Gumbo Mix	$1,931.20	129	$14.97	$373.63	19	$19.66	$3,042.38	150	$20.28
	Louisiana Fiery Hot Pepper Sauce	$2,473.80	155	$15.96	$9,373.19	490	$19.13	$2,022.91	100	$20.23
	Louisiana Hot Spiced Okra	$408.00	30	$13.60	$2,958.00	208	$14.22	$831.36	48	$17.32
Pavlova, Ltd.	Vegie-spread	$3,348.54	109	$30.72	$6,899.26	189	$36.50	$6,453.30	147	$43.90
Plutzer Lebensgrossmarkte	Original Frankfurter grune Sosse	$655.20	63	$10.40	$4,761.38	432	$11.02	$3,755.05	296	$12.69

And, in a Preview, the table now looks like the image shown above.

You can use the **Secondary Crosstab Label Column** element to group Value Columns and Label Columns together. For more information, see "Working with the Crosstab Filter" on page 557.

Summarizing Value Rows

In this example, we're adding three columns in the table that summarize row values (horizontal summary). This is done by adding **Crosstab Row Summary Column** elements to the Crosstab Filter:



The first of these elements refers to the original Value column defined in the Crosstab Filter attributes, *Quantity*, so all that is required is that the element be added beneath the filter, be given an ID, and its summarizing function be set to *Sum*, as shown above.

The second element of this type is summarizing an Extra Crosstab Value Column ("ODTotal") so it needs to include that elements's ID, as highlighted in yellow above, in the **Extra Crosstab Value Column ID** attribute.

Three **Crosstab Table Label Columns** are also added to display the summarized values. The first two reference their data using Label elements and standard tokens *@Data.crsSumQuantity~* and *@Data.crsSumODTotal~*. The third one divides the other two using the formula `=@Data.crsSumODTotal~/@Data.crsSumQuantity~`to derive its value.

		Year 1			Year 2			Year 3			Total		
Supplier	Product Name	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price
Company 1	Product 1	$9,999.99	999	$99.99									
	Product 2	$9,999.99	999	$99.99									
Company 2	Product 1	$9,999.99	999	$99.99									
	Product 2	$9,999.99	999	$99.99									

Referring to our layout model, the addition of the Crosstab Row Summary Column elements adds the columns highlighted in yellow above.

Supplier	Product Name	Total Sales	Units Sold	Avg Price	Total Sales	Units Sold	Avg Price	Total Sales	Units Sold	Avg Price	Total Sales	Units Sold	Avg Price
Exotic Liquids	Aniseed Syrup	$240.00	30	$8.00	$1,724.00	190	$9.07	$1,080.00	108	$10.00	$3,044.00	328	$9.28
Forets d'erables	Sirop d'erable	$1,113.00	42	$26.50	$9,091.50	396	$22.96	$5,261.10	207	$25.42	$15,465.60	645	$23.98
Grandma Kelly's Homestead	Grandma's Boysenberry Spread	$720.00	36	$20.00	$2,500.00	100	$25.00	$3,917.00	165	$23.74	$7,137.00	301	$23.71
	Northwoods Cranberry Sauce	$3,920.00	140	$28.00	$4,260.00	114	$37.37	$4,592.00	118	$38.92	$12,772.00	372	$34.33
Leka Trading	Gula Malacca	$2,042.13	138	$14.80	$6,737.94	396	$17.02	$1,135.88	67	$16.95	$9,915.95	601	$16.50
Mayumi's	Genen Shouyu	$310.00	25	$12.40	$1,474.83	97	$15.20	$363.00	22	$16.50	$2,147.83	144	$14.92
New Orleans Cajun Delights	Chef Anton's Cajun Seasoning	$1,851.52	107	$17.30	$5,214.88	264	$19.75	$1,501.50	82	$18.31	$8,567.90	453	$18.91
	Chef Anton's Gumbo Mix	$1,931.20	129	$14.97	$373.63	19	$19.66	$3,042.38	150	$20.28	$5,347.20	298	$17.94
	Louisiana Fiery Hot Pepper Sauce	$2,473.80	155	$15.96	$9,373.19	490	$19.13	$2,022.91	100	$20.23	$13,869.89	745	$18.62
	Louisiana Hot Spiced Okra	$408.00	30	$13.60	$2,958.00	208	$14.22	$831.36	48	$17.32	$4,197.36	286	$14.68
Pavlova, Ltd.	Vegie-spread	$3,348.54	109	$30.72	$6,899.26	189	$36.50	$6,453.30	147	$43.90	$16,701.10	445	$37.53
Plutzer Lebensgrossmarkte	Original Frankfurter grune Sosse	$655.20	63	$10.40	$4,761.38	432	$11.02	$3,755.05	296	$12.69	$9,171.63	791	$11.59

And, in a Preview, the table now looks like the image shown above.

For information about summarizing columns, see "Summarizing Value Columns" on page 591 and "Comparing, Sorting, and Summarizing Columns" on page 542.

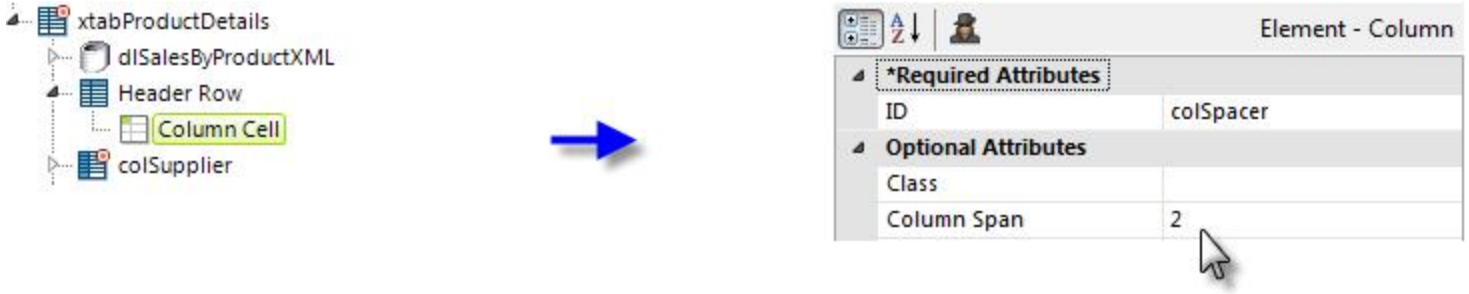
Adding Header Rows

The header row ties the crosstab value columns together and provides data-driven identification for them as well. Before proceeding, look at the header in the table below; it consists of a blank area that spans two columns, three areas that will contain data (the year number) and span three value columns each, and an area labeled "Total" that spans three columns:

		Year 1			Year 2			Year 3			Total		
Supplier	Product Name	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price
Company 1	Product 1	$9,999.99	999	$99.99									
	Product 2	$9,999.99	999	$99.99									
Company 2	Product 1	$9,999.99	999	$99.99									
	Product 2	$9,999.99	999	$99.99									

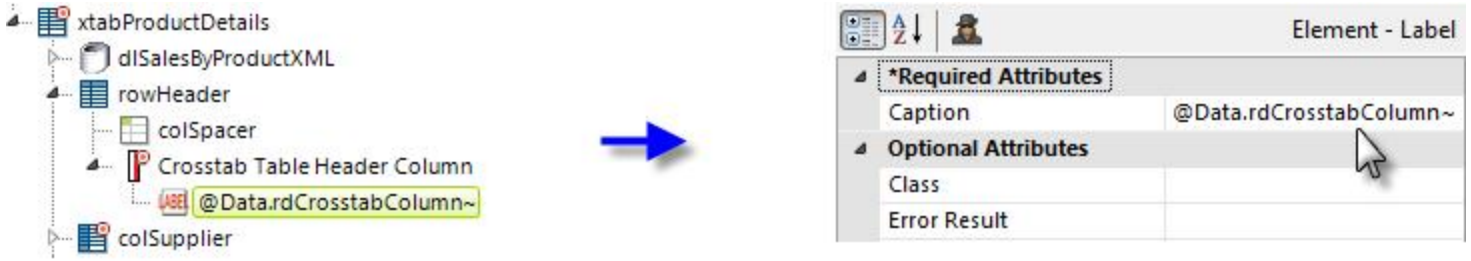
1. Start by adding a **Header Row** element to the Crosstab Table, as shown below, then "build-up" the row using a combination of elements. Set the Header Row element's required **ID** and its **Header Position** attribute to *Top*.

2. Next, add a **Column Cell** element that will be a "spacer" or placeholder; its attributes are set to span the two columns beneath it. It contains nothing.

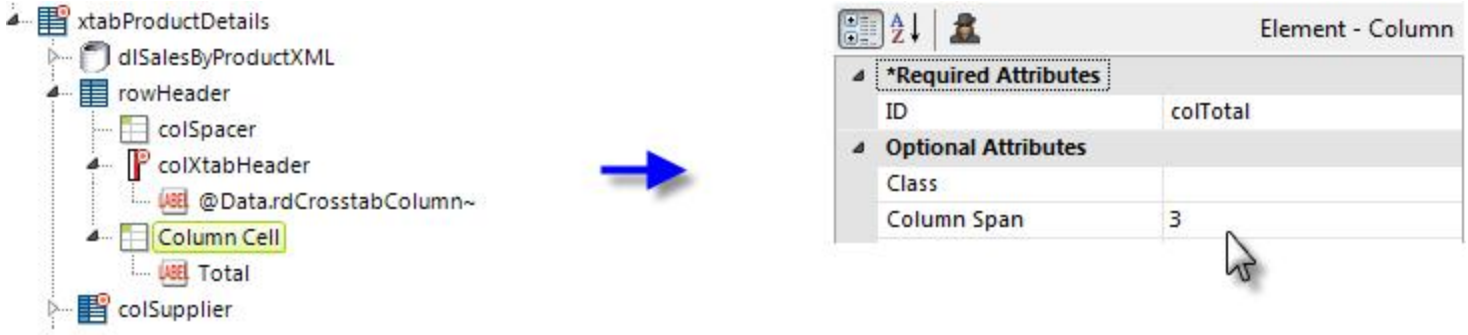


3. Then, add a **Crosstab Table Header Column** element, with a required ID and configured to span three columns.

4. Beneath the Crosstab Table Header Column add a **Label** element and set its Caption attribute value to the token *@Data.rdCrosstabColumn~.* It will display the actual data from the datalayer column identified as the Crosstab Column in the Crosstab Filter's attributes (in our example, this is "OrderYear").



5. Finally, add a second **Column Cell** element and configure it to span three columns:



6. Beneath the Column Cell add a **Label** element with the word "Total" as its **Caption** attribute value.

The elements added in this step create the header row, highlighted in yellow:

		Year 1			Year 2			Year 3			Total		
Supplier	Product Name	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price	Total Sales	Units Sold	Avg. Unit Price
Company 1	Product 1	$9,999.99	999	$99.99									
	Product 2	$9,999.99	999	$99.99									
Company 2	Product 1	$9,999.99	999	$99.99									
	Product 2	$9,999.99	999	$99.99									

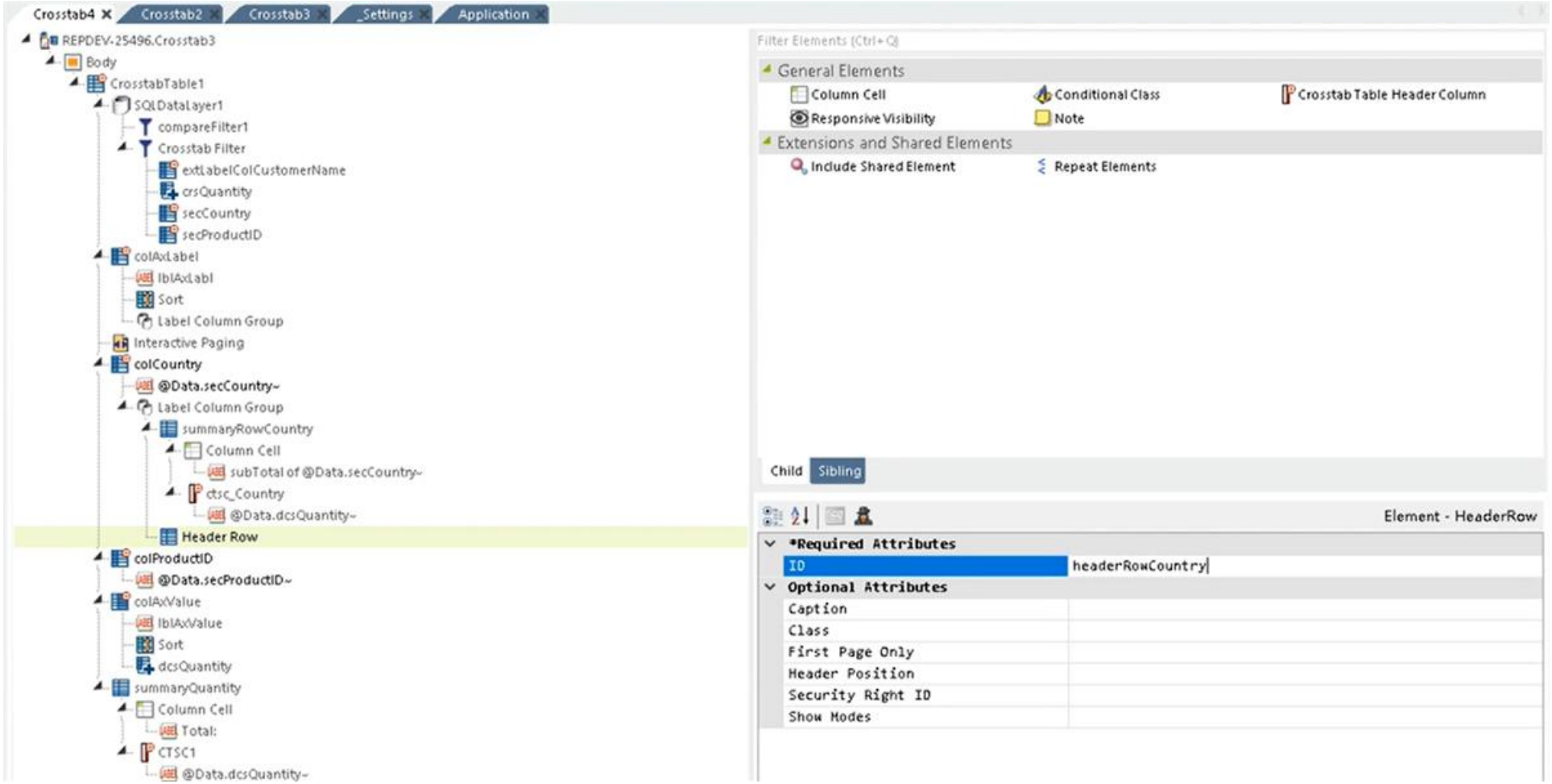
And the resulting table, with header looks like this:

		2013			2014			2015			Total		
Supplier	Product Name	Total Sales	Units Sold	Avg Price	Total Sales	Units Sold	Avg Price	Total Sales	Units Sold	Avg Price	Total Sales	Units Sold	Avg Price
Exotic Liquids	Aniseed Syrup	$240.00	30	$8.00	$1,724.00	190	$9.07	$1,080.00	108	$10.00	$3,044.00	328	$9.28
Forets d'erables	Sirop d'erable	$1,113.00	42	$26.50	$9,091.50	396	$22.96	$5,261.10	207	$25.42	$15,465.60	645	$23.98
Grandma Kelly's Homestead	Grandma's Boysenberry Spread	$720.00	36	$20.00	$2,500.00	100	$25.00	$3,917.00	165	$23.74	$7,137.00	301	$23.71
	Northwoods Cranberry Sauce	$3,920.00	140	$28.00	$4,260.00	114	$37.37	$4,592.00	118	$38.92	$12,772.00	372	$34.33
Leka Trading	Gula Malacca	$2,042.13	138	$14.80	$6,737.94	396	$17.02	$1,135.88	67	$16.95	$9,915.95	601	$16.50
Mayumi's	Genen Shouyu	$310.00	25	$12.40	$1,474.83	97	$15.20	$363.00	22	$16.50	$2,147.83	144	$14.92
New Orleans Cajun Delights	Chef Anton's Cajun Seasoning	$1,851.52	107	$17.30	$5,214.88	264	$19.75	$1,501.50	82	$18.31	$8,567.90	453	$18.91
	Chef Anton's Gumbo Mix	$1,931.20	129	$14.97	$373.63	19	$19.66	$3,042.38	150	$20.28	$5,347.20	298	$17.94
	Louisiana Fiery Hot Pepper Sauce	$2,473.80	155	$15.96	$9,373.19	490	$19.13	$2,022.91	100	$20.23	$13,869.89	745	$18.62
	Louisiana Hot Spiced Okra	$408.00	30	$13.60	$2,958.00	208	$14.22	$831.36	48	$17.32	$4,197.36	286	$14.68
Pavlova, Ltd.	Vegie-spread	$3,348.54	109	$30.72	$6,899.26	189	$36.50	$6,453.30	147	$43.90	$16,701.10	445	$37.53
Plutzer Lebensgrossmarkte	Original Frankfurter grune Sosse	$655.20	63	$10.40	$4,761.38	432	$11.02	$3,755.05	296	$12.69	$9,171.63	791	$11.59

Adding Group Header Rows

You can also add Header Rows to grouped Data or Value Columns. This feature works like the one mentioned above, but instead of creating a Header Row along the top of the Crosstab Table, it allows you to add Header Rows to any Data Columns you have *grouped* together. For information on grouping Crosstabs, see "Working with the Crosstab Filter" on page 557.

1. Add a **Header Row** element beneath a Label Column Group element and give it an ID:



2. **Save** and **refresh** your report. Info displays the Crosstab Table with the new Header Row:

ProductName by Country on Sum of Quantity

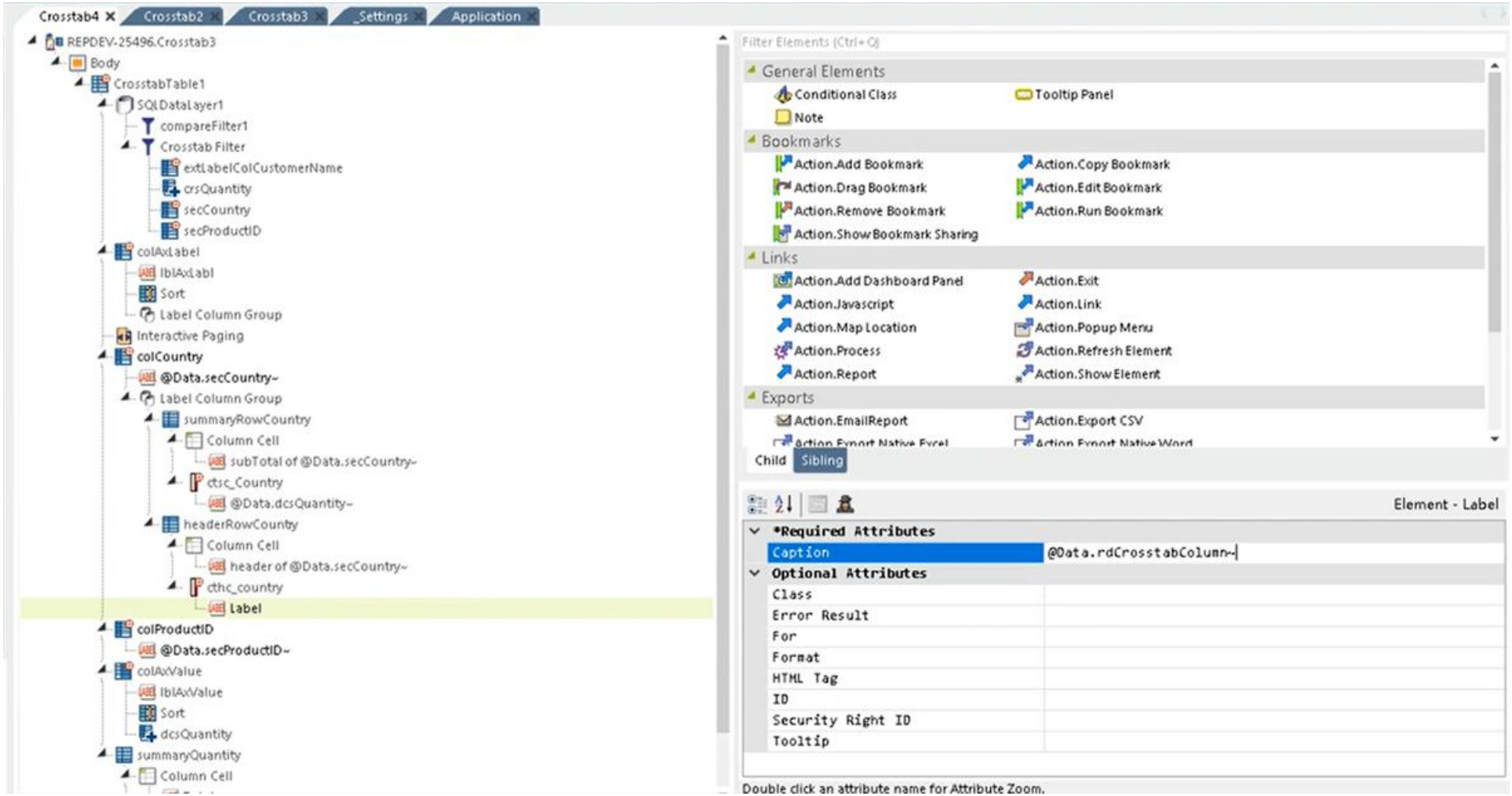
Region	Country	Product ID	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
			798	3,306									
	Australia	1	798										
		6		3,306									
	subTotal of Australia		798	3306									
				9,738	5,034								
	China	6		9,738									
		23			5,034								
Asia-Pacific	subTotal of China			9738	5034								
				6,232									
	Japan	6		6,232									
	subTotal of Japan			6232									
			196			656							
	Malaysia	1	196										
		2				656							
	subTotal of Malaysia		196			656							
			854		1,572		264	2,044					
	Belgium	1	854										
		13					264						
		20						2,044					
		23			1,572								
	subTotal of Belgium		854		1572		264	2044					

3. You can adjust the span of the Header Row by adding a **Column Cell** element and changing the Column Span attribute. In this example, we're adjusting the Column Span to 2.

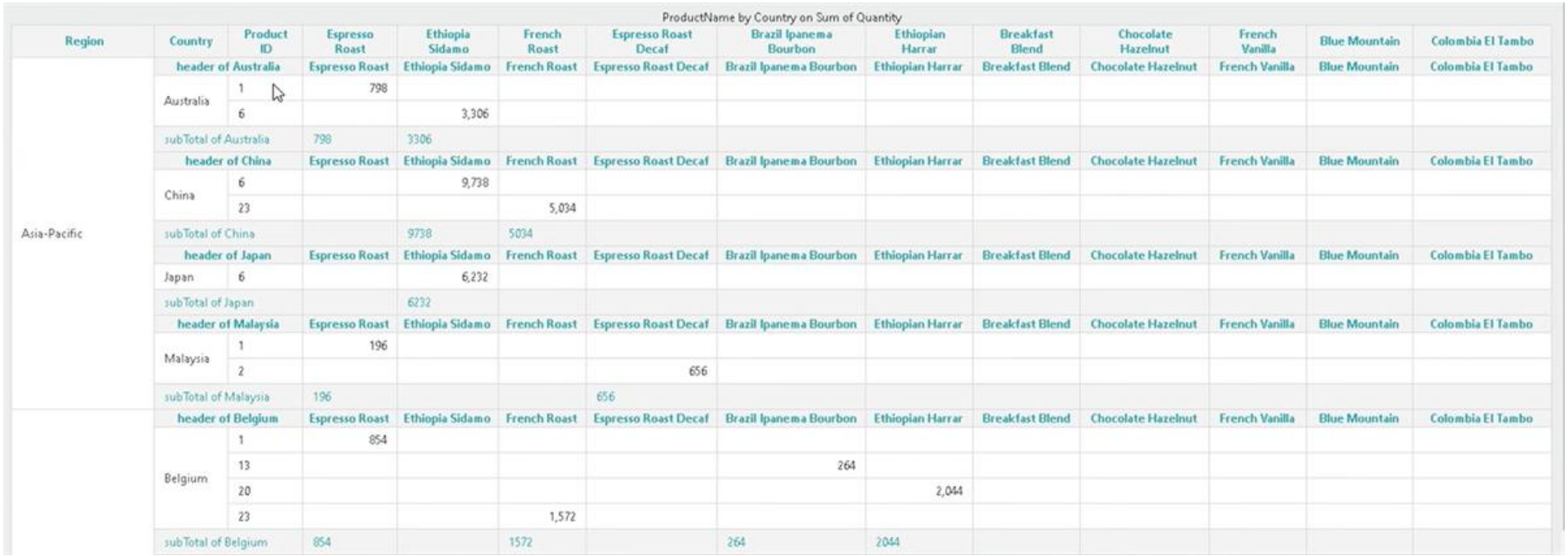
4. Add a **Label** element beneath the Column Cell and give it a Caption based on the Data Column.

5. Next, add a **Crosstab Table Header Column** and give it an ID.

6. Then, add a **Label** element beneath the Crosstab Table Header Column and give it a Caption using the special Crosstab Table token " @Data.rdCrosstabColumn~", shown below:



7. Select **Save** and **refresh** your report. Info displays the Crosstab Table with the new Header Columns:



ProductName by Country on Sum of Quantity

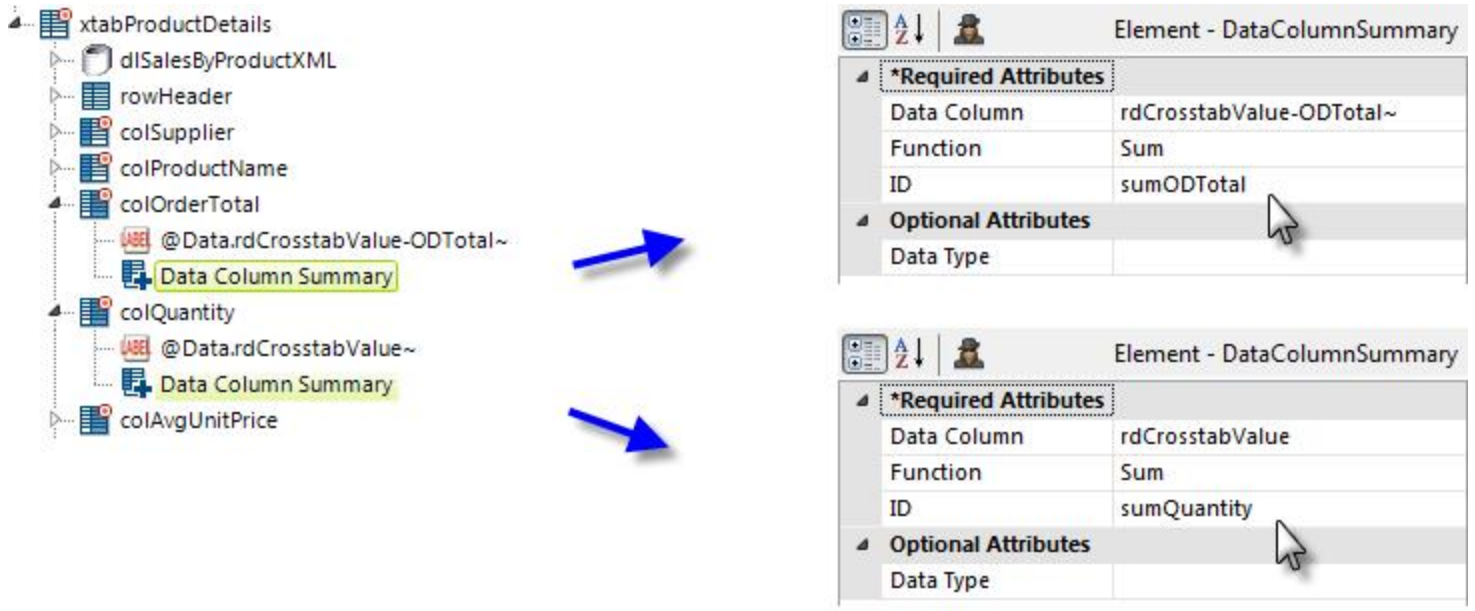
Region	Country	Product ID	Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
Asia-Pacific	header of Australia		Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
	Australia	1	798										
		6		3,306									
	subTotal of Australia		798	3306									
	header of China		Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
	China	6		9,738									
		23			5,034								
	subTotal of China			9738	5034								
	header of Japan		Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
	Japan	6		6,232									
	subTotal of Japan			6232									
	header of Malaysia		Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
	Malaysia	1	196										
		2				656							
	subTotal of Malaysia		196			656							
	header of Belgium		Espresso Roast	Ethiopia Sidamo	French Roast	Espresso Roast Decaf	Brazil Ipanema Bourbon	Ethiopian Harrar	Breakfast Blend	Chocolate Hazelnut	French Vanilla	Blue Mountain	Colombia El Tambo
	Belgium	1	854										
		13					264						
		20						2,044					
		23			1,572								
	subTotal of Belgium		854		1572		264	2044					

You can repeat the previous steps for the Crosstab Filter Label Column (in this example, *Region*) and additional Secondary Crosstab Label Column(s) (in this example, *Product Id*).

Summarizing Value Columns

The creation of a Summary Row at the bottom of a table with summarizing column values is a bit like creating the header row, in that it's built up column-by-column.

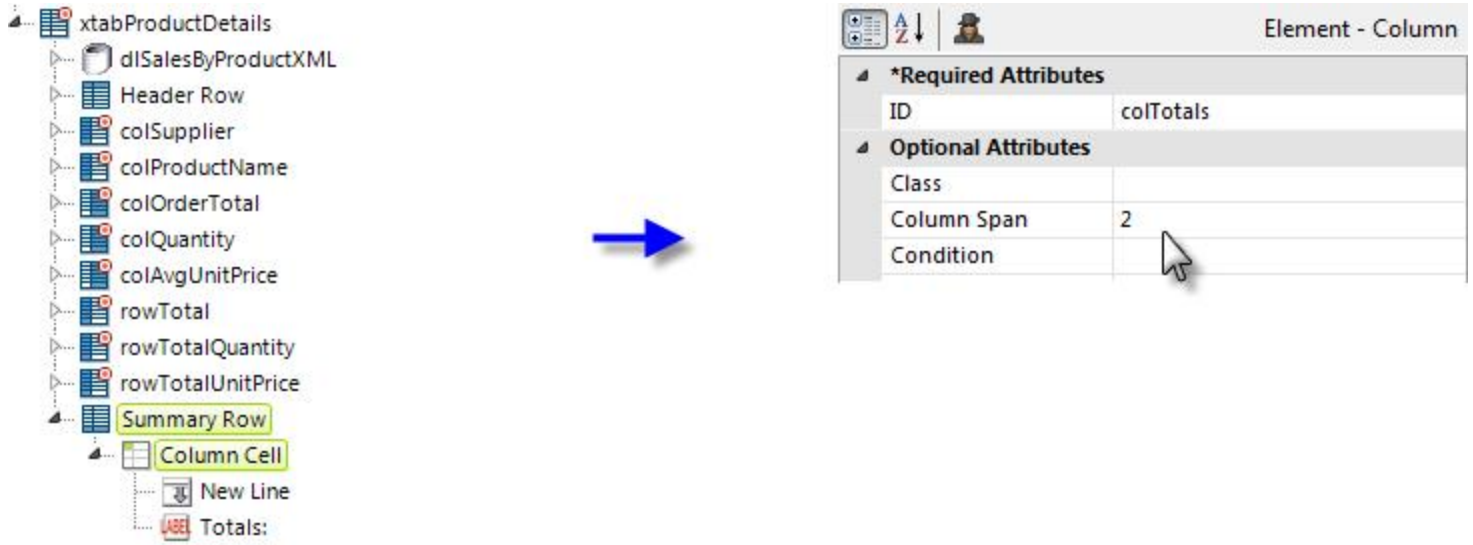
Begin by adding **Data Column Summary** elements beneath the columns to be summarized. The columns that show the average Unit Price do not need these elements as their summarized value can be derived by dividing the summarized values of two other columns.



The special column names, *rdCrosstabValue-ODTotal* and *rdCrosstabValue*, discussed earlier in "Adding Extra Crosstab Values" on page 576, are used in the Data Column Summary elements' attributes for the two extra value columns, as shown above.

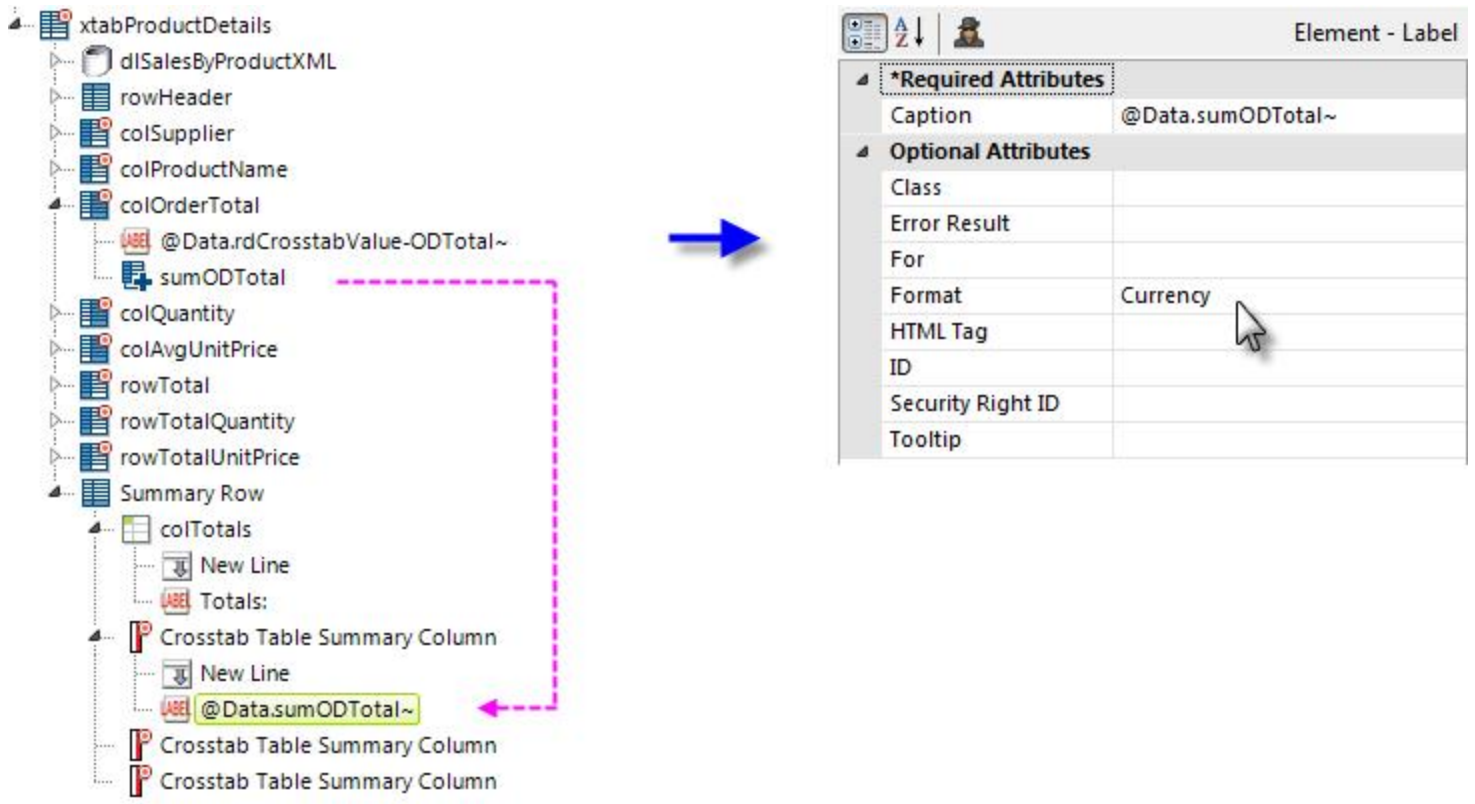
The IDs of the Data Column Summary elements ("sumODTotal", "sumQuantity", etc.) will be used in the next step with an @Data token to identify the summarized data.

Similarly, Summary Row elements are then added beneath Crosstab Label Column elements that display the row (horizontal) totals for each row.



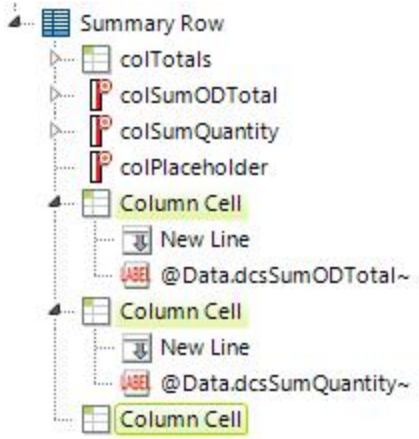
In the example above, a **Summary Row** element has been added beneath the Crosstab Table. Like the Header Row, it will contains elements that "build up" the columns needed for the summaries. **Column Cell** elements are used for non-crosstab value columns and **Crosstab Table Summary Column** elements are used for the value columns.

The first Column Cell element is set to span two columns (it will be underneath the Supplier and Product columns) and contains a **New Line** element and a **Label** element with a Caption of "Totals:".



Next, three Crosstab Table Summary Columns have been added. The first two are used to display the summarized OrderTotal and Quantity value columns. Each contains a New Line and a Label element beneath it, as shown above. The Label elements' Caption attributes are set to the @Data token for the related Data Column Summary column.

The third Crosstab Table Summary Column is just a placeholder and needs no child elements.



Finally, three Column Cell elements are added, as shown above, to display the two summary totals for the row Order Total and Quantity summary columns and a final spacer column.

		2013			2014			2015			Total		
Supplier	Product Name	Total Sales	Units Sold	Avg Price	Total Sales	Units Sold	Avg Price	Total Sales	Units Sold	Avg Price	Total Sales	Units Sold	Avg Price
Exotic Liquids	Aniseed Syrup	$240.00	30	$8.00	$1,724.00	190	$9.07	$1,080.00	108	$10.00	$3,044.00	328	$9.28
Forets d'erables	Sirop d'erable	$1,113.00	42	$26.50	$9,091.50	396	$22.96	$5,261.10	207	$25.42	$15,465.60	645	$23.98
Grandma Kelly's Homestead	Grandma's Boysenberry Spread	$720.00	36	$20.00	$2,500.00	100	$25.00	$3,917.00	165	$23.74	$7,137.00	301	$23.71
	Northwoods Cranberry Sauce	$3,920.00	140	$28.00	$4,260.00	114	$37.37	$4,592.00	118	$38.92	$12,772.00	372	$34.33
Leka Trading	Gula Malacca	$2,042.13	138	$14.80	$6,737.94	396	$17.02	$1,135.88	67	$16.95	$9,915.95	601	$16.50
Mayumi's	Genen Shouyu	$310.00	25	$12.40	$1,474.83	97	$15.20	$363.00	22	$16.50	$2,147.83	144	$14.92
New Orleans Cajun Delights	Chef Anton's Cajun Seasoning	$1,851.52	107	$17.30	$5,214.88	264	$19.75	$1,501.50	82	$18.31	$8,567.90	453	$18.91
	Chef Anton's Gumbo Mix	$1,931.20	129	$14.97	$373.63	19	$19.66	$3,042.38	150	$20.28	$5,347.20	298	$17.94
	Louisiana Fiery Hot Pepper Sauce	$2,473.80	155	$15.96	$9,373.19	490	$19.13	$2,022.91	100	$20.23	$13,869.89	745	$18.62
	Louisiana Hot Spiced Okra	$408.00	30	$13.60	$2,958.00	208	$14.22	$831.36	48	$17.32	$4,197.36	286	$14.68
Pavlova, Ltd.	Vegie-spread	$3,348.54	109	$30.72	$6,899.26	189	$36.50	$6,453.30	147	$43.90	$16,701.10	445	$37.53
Plutzer Lebensgrossmarkte	Original Frankfurter grune Sosse	$655.20	63	$10.40	$4,761.38	432	$11.02	$3,755.05	296	$12.69	$9,171.63	791	$11.59
Totals:		$19,013.39	1004		$55,368.59	2895		$33,955.47	1510		$108,337.45	5,409	

And the resulting Crosstab Table, with header and summary rows, looks like the image above.

Data Lists

The Data List element provides developers with a simple way to produce a true HTML ordered and unordered lists, using data from its datalayer.

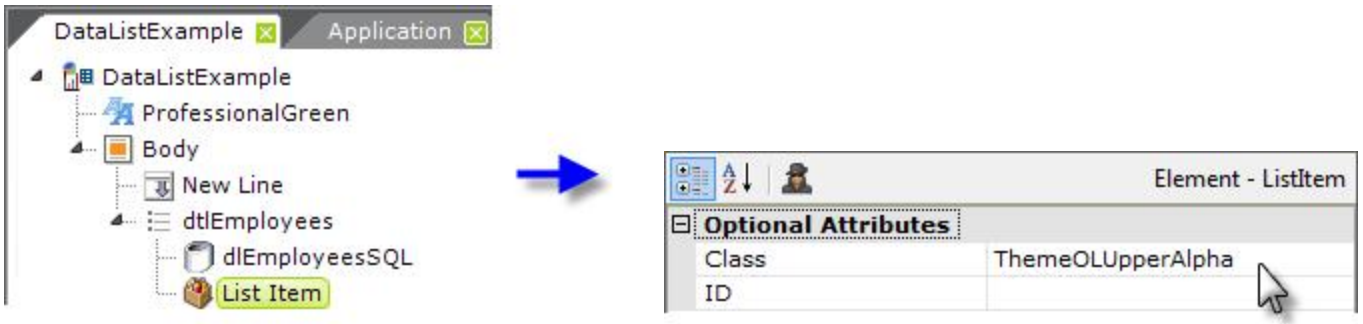
The following topics discuss working with Data Lists:

- Creating an Ordered or Unordered List
- Using the Data List with a Theme

Creating an Ordered and Unordered List

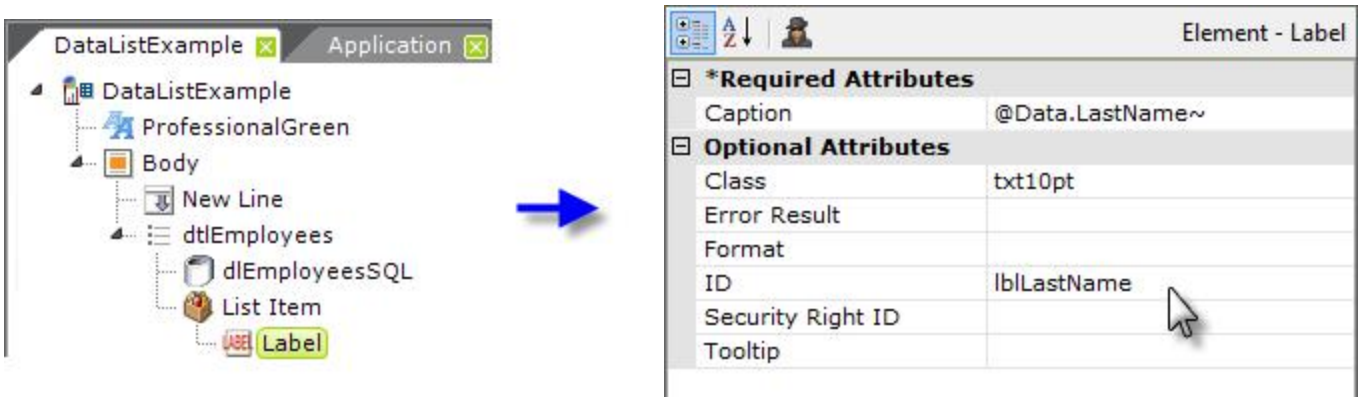
To add an ordered or unordered list to a report definition, start by adding a **Data List** element:

The Data List element is included in the **Data Tables** folder of the Element Toolbox and context menus. As shown in the example above, it uses a datalayer element to retrieve its data. The Data List element's **Ordered** attribute determines whether the list is ordered (numbered) or unordered (bulleted).



A single **List Item** element is used beneath the Data List element as a container for the list items, as shown above. Multiple List Item elements *may* be used but they are not necessary; using more than one can will result in interwoven lists of items.

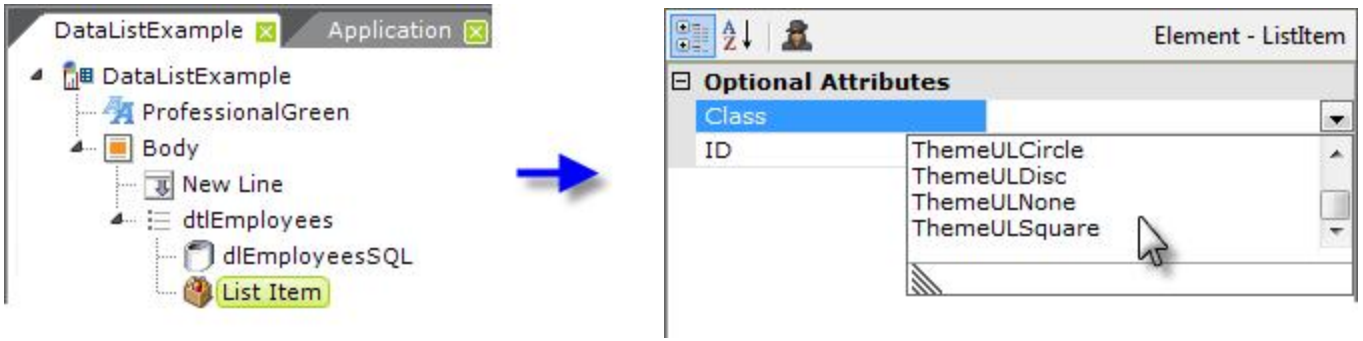
The element's **Class** attribute can be set, if desired, to one of the Theme-related Data List classes, which are discussed in "Using the Data List with a Theme" on the next page. If the attribute is left blank, an unordered list will be shown with black bullets and an ordered list will be shown with decimal numbers.



Finally, an element is added beneath the List Item to show or represent the data. This can be a **Label**, as shown above, or an Image or other elements, or several elements. The usual @Data tokens are used to reference data for display and additional style classes can be assigned.

Using the Data List with a Theme

When you assign a Theme to a Logi application, special standard classes for use with the Data List and List Item elements are made available. For more information, see *Working with Themes*.



As shown above, you can assign these classes to either the Data List or List Item elements' Class attribute (either one, not both). When the Data List is created as an *unordered* list, the **ThemeUL**... classes, shown above, can be used to set the shape of the bullet. When the Data List is created as an *ordered* list, the **ThemeOL**... classes can be used to set the style of the numbering. This table lists the classes and their effects:

Class	Effect
ThemeOLDecimal	1. Buchanan 2. Callahan 3. Davolio 4. Dodsworth

Class	Effect
ThemeOLLowerAlpha	a. Buchanan b. Callahan c. Davolio d. Dodsworth
ThemeOLLowerRoman	i. Buchanan ii. Callahan iii. Davolio iv. Dodsworth
ThemeOLUpperAlpha	A. Buchanan B. Callahan C. Davolio D. Dodsworth
ThemeOLUpperRoman	I. Buchanan II. Callahan III. Davolio IV. Dodsworth
ThemeULCircle	○ Buchanan ○ Callahan ○ Davolio ○ Dodsworth

Class	Effect
ThemeULDisc	• Buchanan • Callahan • Davolio • Dodsworth
ThemeULNone	Buchanan Callahan Davolio Dodsworth
ThemeULSquare	▪ Buchanan ▪ Callahan ▪ Davolio ▪ Dodsworth

Multi-Column Lists

A **multi-column list** is a very basic type of tabular Data Table that allows data to be arranged in columns. Unlike a traditional Data Table, with its rows and columns, a multi-column list displays all its data in cells typically arranged into a small number of columns. Developers should consider this primarily as a means of creating "lists", rather than as a way to present data in a table.

About Multi-Column Lists

The following illustrates the difference between a Data Table and a multi-column list:

Record 1 - Col 1	Col 2	Col 3	Col 4
Record 2 - Col 1	Col 2	Col 3	Col 4
Record 3 - Col 1	Col 2	Col 3	Col 4
Record 4 - Col 1	Col 2	Col 3	Col 4
Record 5 - Col 1	Col 2	Col 3	Col 4
Record 6 - Col 1	Col 2	Col 3	Col 4

Regular Data Table

Record 1 - Col 1 Col 2 Col 3 Col 4	Record 4 - Col 1 Col 2 Col 3 Col 4
Record 2 - Col 1 Col 2 Col 3 Col 4	Record 5 - Col 1 Col 2 Col 3 Col 4
Record 3 - Col 1 Col 2 Col 3 Col 4	Record 6 - Col 1 Col 2 Col 3 Col 4

Multi-Column List

As shown above, *all* of the data from **one record** is available in **one cell** of the multi-column list.

Buchanan, Steven
Sales Manager
14 Garrett Hill
London
SW1 8JR

King, Robert
Sales Representative
Edgeham Hollow Winchester
London
RG1 9SP

Callahan, Laura
Inside Sales Coordinator
4726 - 11th Ave. N.E.
Seattle, WA
98105

Leverling, Janet
Sales Representative
722 Moss Bay Blvd.
Kirkland, WA
98033

Davolio, Nancy
Sales Representative
507 - 20th Ave. E. Apt. 2A
Seattle, WA
98122

Peacock, Margaret
Sales Representative
4110 Old Redmond Rd.
Redmond, WA
98052

Dodsworth, Anne
Sales Representative
7 Houndstooth Rd.
London
WG2 7LT

Suyama, Michael
Sales Representative
Coventry House Miner Rd.
London
EC2 7JR

Fuller, Andrew
Vice President of Sales
908 W. Capital Way
Tacoma, WA
98401

The multi-column list shown above is organized so that the data is arranged from **top-to-bottom** in the first column, showing all the information from one record in individual cells, then from top-to-bottom in the second column. The number of columns is configurable.

The example above shows the same data in a **3-column** list. Because the direction of this list is *Down*, the sort order (based on Last Name) causes the data to be arranged in a top-to-bottom fashion in each column.

Now, in the example shown above, the direction has been switched to *Across*, so the sort order runs left-to-right.

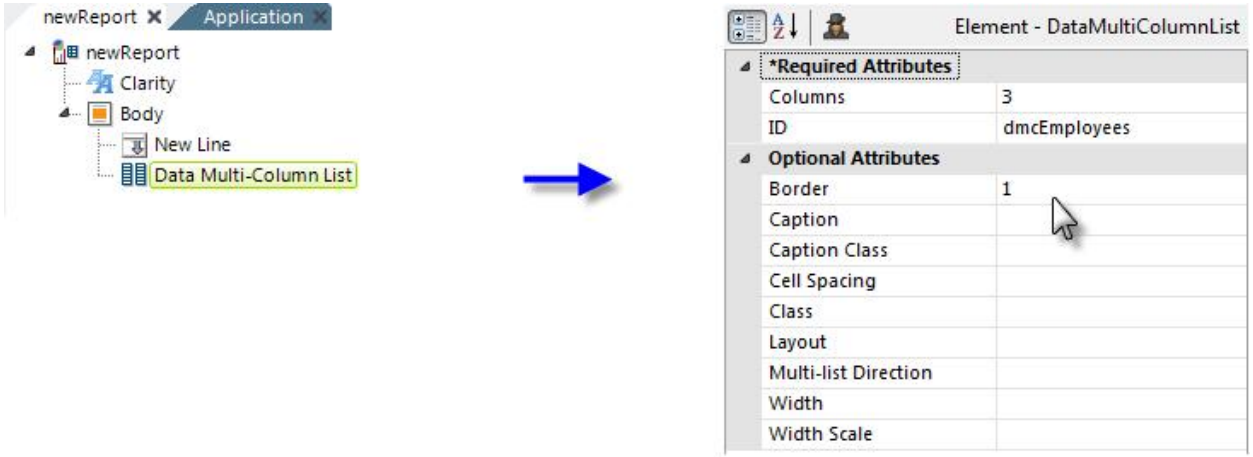
Developers can choose any type of datalayer element to retrieve data for the list. **Label** elements are used to display the data, and developers can combine them with other types of elements to organize the data.

The **Data Multi-Column List** element is used to create the list in a report definition. Multi-column lists are different from other types of Data Tables in that the number of columns that can be displayed is not dependent on the datalayer. **Data Table Column** elements *arenot* used and, instead, developers set the Data Multi-Column List element's **Columns** attribute to a numeric value that determines the fixed number of columns the list will display.

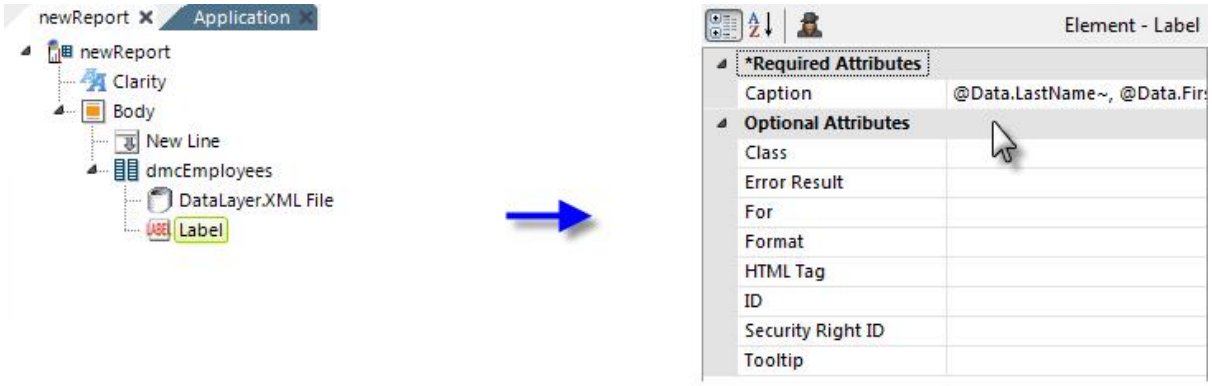
The number of rows displayed is determined automatically by distributing the data in the columns, based on the list orientation, beginning with the upper left-hand corner of the list. Any elements (other than datalayer elements) placed below a Data Multi-Column List element are repeated for each row of data in the datalayer. Content that must appear only once in the final report should not be placed within a multi-column list.

Creating a Multi-Column List

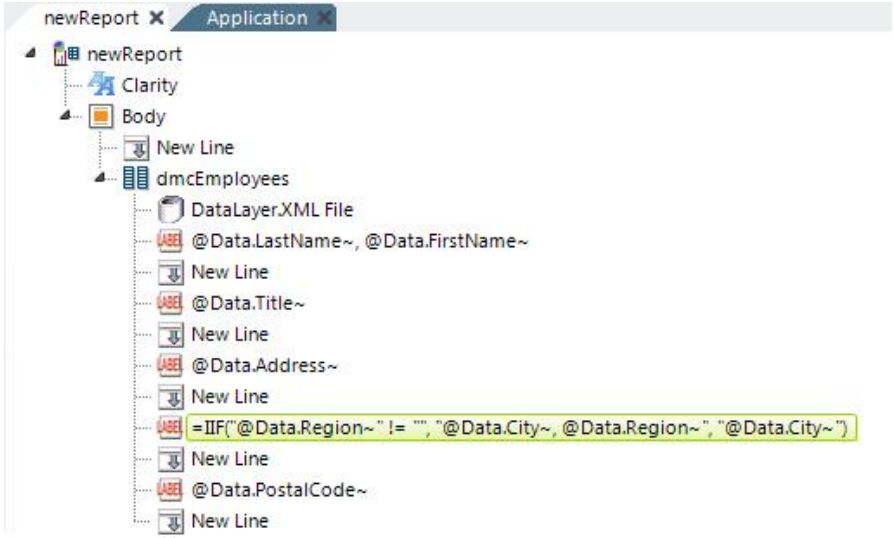
Creating a multi-column list involves configuring the list, retrieving data for the list, and displaying the list. The element itself does the work of arranging the data.



1. Add a **Data Multi-Column List** element to your definition.
2. Set the **Columns** attribute to the desired number of list columns.
3. Set the **ID** attribute.
4. Set **Multi-list Direction** attribute to control the data layout. Set it to *Across* to display data left-to-right: a new table cell is created for each row in the data layer. Leave it blank, or set it to *Down*, to display data top-to-bottom.



5. Beneath the Data Multi-Column List element, add a **DataLayer** element and configure it as necessary.
6. Beneath the Data Multi-Column List element, add **Label** elements, **New Lines**, etc. as needed. Data Table Column elements are *not* necessary. Use a standard **@Data** token for the Label element's Caption attribute to display the data.



7. In the example shown above, multiple Label and New Line elements have been added. The Label element that displays City and Region data uses a formula that suppresses the display of a comma after the City if there is no Region data.

8. And the example above shows the resulting output. The background color has been set using CSS for illustration purposes.

Glossary

A

API

API, short for Application Program Interface, is a set of routines, protocols, and tools for building software applications. In business intelligence, APIs may be used to enable end-users to directly update source systems.

Authentication

Authentication is the verification of a user's identity.

Authorization

After a user's identity has been authenticated, authorization grants or denies access to reports, columns, and records to selected users or user-groups.

B

Big Data

Refers to both the ever-growing volumes of data in use today and also to services that are specifically engineered to provide and manipulate very large data volumes.

Business Analytics

Business analytics, or business intelligence (BI), gives customers the ability to rapidly create scalable, interactive data analysis applications, and self-service capabilities users can access from anywhere and on any device.

C

Columnar Data Store

Columnar data store is a type of big data repository containing structured data in columns and rows. The main benefits are that the data can be highly compressed and is easily searchable.

CRM

A Customer Relationship Management (CRM) system is a database-based system that records a company's daily customer-related transactions. CRMs can help customer representatives to provide better service, close more deals, and increase revenue.

CSS

Cascading Style Sheets (CSS) is a technology that allows the presentation aspects of web pages to be separated from the page content. It can be used to add "styling" (e.g. apply fonts, colors, alignment, spacing, and more) to web pages.

D

Data Discovery

Data discovery is the capability to analyze data on-the-fly and uncover insights from it.

Data Enrichment

Data enrichment is a method of preparing data to make it ready for analysis and exploitation, and can include formatting, adding calculations, joining with other data, and more.

DevNet

The Logi Developer Network website.

Drill Down

Drill Down is a capability that allows the user to get a view of the underlying or supporting data used in an analysis.

Drill Through

Drill Through is similar to Drill Down but takes it one step further by applying analysis to the underlying or supporting data.

E

Elemental Development

A development approach used in Logi Info that lets developers build feature-rich applications by using reusable, pre-built elements, rather than by writing low-level code.

F

Forecasting

A technique involving data mining and analysis leading to predictions about what will happen in the future.

G

Geo Mapping

The combination of geographic and other data to produce map visualizations, such as Google or Leaflet maps.

H

Heatmap

A Heatmap chart, sometimes called a "tree map", which uses a unique arrangement of rectangles to represent data and relationships, using color and size.

I

Interpolation

The process of evaluating a literal value match containing one or more placeholders, yielding a result in which the placeholders are replaced with their corresponding values.

J

JavaScript

JavaScript is a programming language supported by the majority of modern web browsers and used by many websites.

JDBC

Java Database Connectivity (JDBC) is an API used to access relational databases. Open Database Connectivity (ODBC) is a similar API designed for use with Java.

JSON

JavaScript Object Notation (JSON) is a lightweight data-interchange format that's easy for humans to read and write, and easy for computers to parse and generate.

K

KPI

Key Performance Indicators (KPIs) are visual indicators, in the form of color-coded shapes, which are tied to a pre-defined, critical threshold.

L

LDAP

The Lightweight Directory Access Protocol (LDAP) is an Internet protocol applications use to look up information from a server and is frequently used for containing user login information.

M

My Term

My definition

N

NoSQL

"Not only SQL" (NoSQL) is an alternative to traditional relational databases, and doesn't rely on tables and a pre-determined schema. NoSQL databases are especially useful for working with large sets of distributed data.

O

ODBC

Open Database Connectivity (ODBC) is an API used to access relational databases. Java Database Connectivity (JDBC) is a similar API designed for use with Java.

OLAP

Online Analytical Processing (OLAP) is the process of analyzing data stored in multi-dimensional "cubes".

R

REST

Representational State Transfer (REST) is a type of API used to provide interoperability between computer systems on the Internet.

S

SSM

The Self-Service Module (SSM) is a package that includes Logi Info + SSRM + Discovery or Logi Platform Services.

SSRM

The Self-Service Reporting Module (SSRM) is a Logi Info add-on module that adds special elements to Info and includes the InfoGo application.

W

Write-Back

The ability to update data sources, typically by adding, editing, or deleting data.