databricks Academy

DAWD 03-2 - Demo - Data Visualizations and Dashboards

The two fields below are used to customize queries used in this course. Enter your schema (database) name and username, and press "Enter" to populate necessary information in the queries on this page.

Schema Name:	hive_metastore.class_013_odg7_da_dawd
Username:	class+013@databricks.com

Lesson Objective

At the end of this lesson, you will be able to:

- Describe how to create basic visualizations using Databricks SQL
- Describe how to create a dashboard using multiple existing visualizations from Databricks SQL Queries
- Describe how to parameterize queries and dashboards to customize results and visualizations

The Counter

The Counter visualization is one of the simplest visualizations in Databricks SQL. It displays a single number by default, but it can also be configured to display a "goal" number. In this example, we are going to configure a sum of completed sales, along with a "Sales Goal." The query calculates a sum of total sales and also provides a hard-coded sales goal column.

Complete the following:

1. Open a new query tab, and run the query below
2. Save the query as "Count Total Sales"

Visualizations are stored with the queries that generate data for them. Although we probably could pick a better name than "Counter", this will help us when we build our dashboard later in this lesson. Note also that we can have multiple visualizations attached to a single query

3. In the query results section, hover over the "+" symbol, and click "Visualization"
4. Select "Counter" as the visualization type
5. For "Counter Label" type "Total Sales"
6. For "Counter Value Column" make sure the column "Total_Sales" is selected
7. For "Target Value Column" choose "Sales Goal"

Note that we can configure the counter to count rows for us if we did not aggregate our data in the query itself.

8. Click the "Format" tab
9. Optional: Change the decimal character and thousands separator
10. "Total Sales" is a dollar figure, so add "$" to "Formatting String Prefix"
11. Turn the switch, "Format Target Value" to on
12. Click "Save" in the lower-right corner
13. Click the name of the visualization (the name of the tab) and change the name to "Total Sales"
14. Make sure the query is Saved

```
USE hive_metastore.class_013_odg7_da_dawd;
SELECT sum(total_price) AS Total_Sales, 3000000 AS Sales_Goal
    FROM sales;
```

Copy

The Bar Chart

One of the most often used visualizations in data analytics is the Bar Chart. Databricks SQL supports an variety of customization options to make bar charts look beautiful. In this example, we are going to configure a bar chart

Complete the following:

1. Open a new query tab, and run the query below
2. Save the query as "Sales Over Three Months"
3. In the query results section, hover over the "+" symbol, and click "Visualization"
4. Select "Bar" as the visualization type
5. For "X Column" choose "Month"
6. For "Y Columns" click "Add column" and select "Total Sales" and "Sum"
7. Click "Add column" again and select "Total Sales" and "Count"
8. Click the "Y Axis" tab and type "Dollars" in the "Name" field (Left Y Axis)
9. Click the "Series" tab and type "Total Sales" in the first "Label" field
10. Type "Number of Sales" in the second "Label" field and change "Type" to "Line"
11. Click "Save" in the lower-right corner
12. Click the name of the visualization (the name of the tab) and change the name to "Sales by Month"
13. Make sure the query is Saved

As we can see from the visualization, the number of sales in August and October was low, but the dollar amounts of those sales was high. The opposite is true in September.

```
USE hive_metastore.class_013_odg7_da_dawd;
SELECT customer_name, total_price AS Total_Sales, date_format(order_date, "MM") AS Month,
product_category
    FROM sales
    WHERE order_date >= to_date('2019-08-01')
    AND order_date <= to_date('2019-10-31');
```

Copy

The Stacked Bar Chart

We can glean more data from the same query by adding a second visualization.

Complete the following:

1. Hover over the "+" symbol, and click "Visualization"
2. Change "Visualization Type" to "Bar"
3. For "X Column" choose "product_category"
4. Add two Y Columns and change both to "Total_Sales". Change the first to "Average" and the second to "Min"
5. Change "Stacking" to "Stack"
6. On the "X Axis" tab, change the name to "Product Category"
7. On the "Y Axis" tab, change the name to "Dollars"
8. On the "Series" tab, change the first row Label to "Average Sales" and the second row to "Minimum Sales"
9. Click "Save" in the lower-right corner
10. Click the name of the visualization (the name of the tab) and change the name to "Sales by Product Category"
11. Make sure the query is Saved

This visualization shows that, although the "Reagate" category has the highest minimum sales figure, it has the lowest average.

Maps - Choropleth

Databricks SQL has two map visualizations you can use to plot address and geolocation data: choropleth and markers. The choropleth map visualization uses color to show the count of a criterion within a specific geographic area. In this example, we are going to use customer address data to plot the number of customers in each U.S. state.

To make a choropleth map, complete the following:

1. Open a new query tab, and run the query below
2. Save the query as "Count Customers by State"
3. Hover over the "+" symbol, and click "Visualization"
4. Select "Map (Choropleth)" as the visualization type
5. In the "General" tab, change "Map" to "USA", "Key Column" to state, "Target Field" to "USPS Abbreviation", and "Value Column" to "count(customer_id)"
6. Click "Save" in the lower-right corner
7. Click the name of the visualization (the name of the tab) and change the name to "Most Active States"
8. Make sure the query is Saved

```
USE hive_metastore.class_013_odg7_da_dawd;
SELECT state, count(customer_id) FROM customers
    GROUP BY state;
```

Copy

Maps - Markers

The Map (Markers) visualization type plots points on a map that signify a specific location. In this example, we have latitude and longitude data for our customer locations. We will use this to plot those locations on a map.

Complete the following:

1. Open a new query tab, and run the query below
2. Save the query as "All Customers"
3. Hover over the "+" symbol, and click "Visualization"
4. Select "Map (Markers)" as the "Visualization Type"
5. In the General tab, change "Latitude Column" to "lat", "Longitude Column" to "lon", and "Group By" to "state"
6. On the "Format" tab, enable tooltips and type "{{customer_name}}" in the "Tooltip template" field

Note: Because we are on a 2x-Small Warehouse, do not uncheck "Cluster Markers" in the "Styles" tab. The map refresh process will take a very long time to update.

1. Click "Save" in the lower-right corner
2. Click the name of the visualization (the name of the tab) and change the name to "Customer Locations"
3. Make sure the query is Saved

```
USE hive_metastore.class_013_odg7_da_dawd;
SELECT * FROM customers;
```

Copy

Dashboards

We are now going to combine all the visualizations we created above into a dashboard that will display them all at once and that we can put on a refresh schedule to keep the data that underlies each visualization up-to-date. In a future lesson, we will talk about setting a refresh schedule and subscribing stakeholders to the dashboard's output, so they can always have the newest information.

Complete the following:

1. Click "Dashboards" in the sidebar menu
2. Click "Create Dashboard"
3. Name the dashboard "Retail Organization"
4. Click "Add", and click "Visualization"

We are presented with a list of queries that have been saved. Although we named our queries based on the type of visualization we made, it makes more sense to name a query based on what it does.

5. Click "Count Total Sales"
6. Leave "Select existing visualization" selected, and drop down the list directly below. Note we have the results table from our query and our counter visualization, "Total Sales" available to us. Select "Total Sales"
7. Change "Title" to "Total Sales"
8. Optional: write a description
9. Click "Add to dashboard"
10. Repeat steps 4-9 with the "Sales Over Three Months" query ("Sales by Month" and "Sales by Product Category"), "Count Customers by State" query ("Most Active States"), and "All Customers" query ("Customer Locations")

You should have five visualizations in the dashboard

11. Click "Add", then "Text Box", and type "# Customers and Sales" in the "Edit:" box

Note that text boxes support Markdown.

12. Click "Add to dashboard"
13. Optional: Move the visualizations around by clicking and dragging each one
14. Optional: Resize each visualization by dragging the lower-right corner of the visualization
15. Optional: Click "Colors" to change the color palette used by visualizations in the dashboard
16. Click "Done Editing" in the upper-right corner
17. Run every query and refresh all visualizations all at once by clicking "Refresh"

Parameterized Queries

Before we leave this lesson, let's talk about a customization feature we can apply to our queries to give them more flexibility. Query parameters allow us to make changes to our queries without requiring new code to be written.

Complete the following:

1. Go back to the Query Editor and start a new query
2. Paste the query below into the editor and save the query as "Get Product Category"

```
USE hive_metastore.class_013_odg7_da_dawd;
SELECT DISTINCT product_category FROM sales;
```

Copy

Writing a Query with Parameters

Now that we have a query that pulls all product categories from the `Sales` table, let's use this query as a parameter in a second query.

Complete the following:

3. Start a new query, and paste the code below in the editor

Note that the query has empty single quotes.

4. Place your cursor in-between the single quotes, hover over the "+" symbol, and click "Parameter"
5. Input "category" for the "Keyword" field
6. Drop down "Type" and choose "Query Based Dropdown List"
7. For "Query" choose the query we created above: "Get Product Category"
8. Click "Add Parameter"
9. Save the query as "Total Sales by Product Category"

Note two things: First, we now have a set of double curly braces that contain the word "category". This is where are query parameter was inserted. Finally, note the dropdown list we how have just above the query results window.

10. Open the dropdown list and choose a Category from the list

11. Click "Apply Changes"

    The query is rerun with the chosen product category replacing the location of the query parameter in the query. Thus, we see the Total Sales of the Category we chose.

```
USE hive_metastore.class_013_odg7_da_dawd;
SELECT sum(total_price) AS Total_Sales FROM sales
    WHERE product_category = '';
```

Copy