Creating a Data Warehouse Through Joins and Unions

Overview

BigQuery is Google's fully managed, NoOps, low cost analytics database. With BigQuery you can query terabytes and terabytes of data without having any infrastructure to manage or needing a database administrator. BigQuery uses SQL and can take advantage of the pay-as-you-go model. BigQuery allows you to focus on analyzing data to find meaningful insights.

The dataset you'll use is an ecommerce dataset that has millions of Google Analytics records from the Google Merchandise Store. You will explore the available fields and row for insights.

This lab focuses on how to create new reporting tables using SQL JOINS and UNIONs.

What you'll do

In this lab, you learn how to perform these tasks:

- Explore new ecommerce data on sentiment analysis
- Join together datasets and create new tables
- Append historical data with unions and table wildcards

Create a new dataset to store your tables

First, create a new dataset titled **ecommerce** in BigQuery.

1. In the left pane, click on the name of your BigQuery project (`userxx-project-xxxx`).
2. Click on the three dots next to your project name, then select **CREATE DATASET**.

The **Create dataset** dialog opens.

3. Set the *Dataset ID* to `ecommerce`, leave all other options at their default values.

Click **Create dataset**.

4. Click on the **Disable Editor Tabs** link to enable the Query Editor.

Scenario: Your marketing team provided you and your data science team all of the product reviews for your ecommerce website. You partner with them to create a data warehouse in BigQuery which joins together data from three sources:

- Website ecommerce data
- Product inventory stock levels and lead times
- Product review sentiment analysis

In this lab, you examine a new dataset based on product reviews.

BigQuery project

The project with your marketing team's dataset is **data-to-insights**. BigQuery public datasets are not displayed by default in BigQuery. The queries in this lab will use the `data-to-insights` dataset even though you cannot see it.

Since **data-to-insights** is a public dataset project, you have to pin it to your Resource Tree. To do that, do the following:

1. Click on **HIDE PREVIEW FEATURES**.
2. In a new browser window, open the public datasets project, https://console.cloud.google.com/bigquery?p=data-to-insights&page=ecommerce.
3. In the left pane, in the Resource section, click **data-to-insights**. In the right pane, click **Pin Project**.
4. Close this browser window.
5. Return to and refresh the first BigQuery browser window to refresh the BigQuery web UI.

The `data-to-insights` project is listed in the Resource section.

6. Click on **SHOW PREVIEW FEATURES**.

Enrich ecommerce data with Machine Learning

In this section, you get a feel for how the Natural Language API performs sentiment analysis on product reviews.

Sentiment analysis attempts to determine the overall attitude (positive or negative) expressed within text and is made up of *score* and *magnitude*.

- *Score* ranges between `-1.0` (negative) and `1.0` (positive) and corresponds to the overall emotional leaning of the text.
- *Magnitude* indicates the overall strength of emotion (both positive and negative) within the given text, between `0.0` and `+infinity`. Unlike score, magnitude is not normalized; each expression of emotion within the text (both positive and negative) contributes to the text's magnitude (so longer text blocks may have greater magnitudes).

Use some fictional product reviews to check out how the sentiment analysis works.

1. Open the Cloud Natural Language page in a new browser window or tab.
2. Scroll down to find the **Try the API** text box.
3. Replace the text inside the text box with the following text:
   ```
   I liked this flashlight at first. It worked ok for about a month
   but then it just stopped working. So I'm quite disappointed with
   it.
   ```
4. Click **Analyze**.
5. After the results come in, click on the **Sentiment** tab.

Analyze the following product reviews. Keep track of Scores as you go as you'll be asked. Which of those product reviews have the most positive sentiment? The most negative sentiment? The most neutral sentiment?

Review #1:

The three dog frisbees we ordered unfortunately didn't do well with our bigger German Shepherd dogs.

Review #2:

The three dog frisbees we ordered unfortunately didn't do well with our bigger German Shepherd dogs. Firstly, they had a tough time catching them in the air since the light blue models matched the color of the sky and secondly the material they were made out of wasn't strong enough to withstand more than a couple months of use before they got chewed up.

Review #3:

Honestly I've gone through quite a few umbrellas in the past but this new red Executive Umbrella is one of the best. We ended up going with red but you have a wide variety of colors to choose from. The umbrella material was excellent and didn't degrade after heavy use (we're in Seattle!).

Review #4:

I love these sunglasses. They are sturdy, look nice, and are functional. Highly recommended!

Review #5:

`I got one of the microfleece jackets as a gift and wear it most days. The material is good and not itchy.`

Review #6:

I pre-ordered a few yoga blocks but my shipment kept getting delayed because of supplier delays. Not sure what's going on there but would be great to speed up shipment.

Explore the product sentiment dataset

Your data science team has run all of your product reviews through the API and provided you with the average sentiment score and magnitude for each of your products.

First, create a copy the table that the data science team made so you can read it:

```
create or replace TABLE ecommerce.products AS
SELECT
*
FROM
`data-to-insights.ecommerce.products`
```

Note: This is only for you to review, the queries in this lab will be using the `data-to-insights` project. Click on the **ecommerce** dataset to display the `products` table.

Examine the data

1. Navigate to the **ecommerce** > **products** dataset and click the **Preview** tab to see the data.

How many Aluminum Handy Emergency Flashlight have been ordered?

- 90
- 0
- 66
- 85

2. Click the **Schema** tab.

What data type are the sentimentScore and sentimentMagnitude fields?

- FLOAT
- STRING
- RECORD
- INTERGER

Create a query that shows the top 5 products with the most positive sentiment

In the **Query Editor**, write your SQL query.

Possible Solution:

```
SELECT
  SKU,
  name,
  sentimentScore,
  sentimentMagnitude
FROM
  `data-to-insights.ecommerce.products`
ORDER BY
  sentimentScore DESC
LIMIT 5
```

What product has the highest sentiment?

- USB wired soundbar - in store only
- G Noise-reducing Bluetooth Headphones
- Stylus Pen w/ LED Light
- G Noise-reducing Bluetooth Headphones

Revise your query to show the top 5 products with the most negative sentiment.

Filter out NULL values.

Possible Solution:

```
SELECT

  SKU,

  name,

  sentimentScore,
  sentimentMagnitude
FROM
  `data-to-insights.ecommerce.products`
WHERE sentimentScore IS NOT NULL
ORDER BY
  sentimentScore
LIMIT 5
```

What is the product with the lowest sentiment?

- Womens Convertible Vest-Jacket Sea Foam Green
- 7 inch Dog Frisbee
- Mens Vintage Henley
- 4 Womens Vintage Hero Tee Platinum

Join datasets to find insights

Scenario It's the first of the month and your inventory team has informed you that the `orderedQuantity` field in the product inventory dataset is out of date. They need your help to query the total sales by product for 08/01/2017 and reference that against the current stock levels in inventory to see which products need to be resupplied first.

Calculate daily sales volume by productSKU

Create a new table in your **ecommerce** dataset with the below requirements:

- Title it `sales_by_sku_20170801`
- Source the data from `data-to-insights.ecommerce.all_sessions_raw`
- Include only distinct results
- Return `productSKU`
- Return the total quantity ordered (`productQuantity`). Hint: Use a `SUM()` `with a IFNULL` condition
- Filter for only sales on `20170801`
- `ORDER BY` the SKUs with the most orders first

Possible Solution:

```
# pull what sold on 08/01/2017
CREATE OR REPLACE TABLE ecommerce.sales_by_sku_20170801 AS
SELECT DISTINCT
```

```
          productSKU,
          SUM(IFNULL(productQuantity,0)) AS total_ordered
        FROM
          `data-to-insights.ecommerce.all_sessions_raw`
        WHERE date = '20170801'
        GROUP BY productSKU
        ORDER BY total_ordered DESC #462 skus sold
```

Click on the `sales_by_sku` table, then click the **Preview** tab. How many distinct product SKUs were sold?

Answer: 462

Next, enrich your sales data with product inventory information by joining the two datasets.

Join sales data and inventory data

Using a JOIN, enrich the website ecommerce data with the following fields from the product inventory dataset:

- `name`
- `stockLevel`
- `restockingLeadTime`
- `sentimentScore`
- `sentimentMagnitude`

Complete the partially written query:

```
# join against product inventory to get name
SELECT DISTINCT
  website.productSKU,
  website.total_ordered,
  inventory.name,
  inventory.stockLevel,
  inventory.restockingLeadTime,
  inventory.sentimentScore,
  inventory.sentimentMagnitude
FROM
  ecommerce.sales_by_sku_20170801 AS website
  LEFT JOIN `data-to-insights.ecommerce.products` AS inventory
ORDER BY total_ordered DESC
```

Possible Solution:

```
# join against product inventory to get name
SELECT DISTINCT
  website.productSKU,
  website.total_ordered,
  inventory.name,
  inventory.stockLevel,
  inventory.restockingLeadTime,
  inventory.sentimentScore,
  inventory.sentimentMagnitude
FROM
  ecommerce.sales_by_sku_20170801 AS website
  LEFT JOIN `data-to-insights.ecommerce.products` AS inventory
  ON website.productSKU = inventory.SKU
ORDER BY total_ordered DESC
```

Modify the query you wrote to now include:

- A calculated field of (total_ordered / stockLevel) and alias it "ratio". Hint: Use SAFE_DIVIDE(field1,field2) to avoid divide by 0 errors when the stock level is 0.
- Filter the results to only include products that have gone through 50% or more of their inventory already at the beginning of the month

Possible Solution:

```
# calculate ratio and filter
SELECT DISTINCT
  website.productSKU,
  website.total_ordered,
  inventory.name,
  inventory.stockLevel,
  inventory.restockingLeadTime,
  inventory.sentimentScore,
  inventory.sentimentMagnitude,
  SAFE_DIVIDE(website.total_ordered, inventory.stockLevel) AS ratio
FROM
  ecommerce.sales_by_sku_20170801 AS website
  LEFT JOIN `data-to-insights.ecommerce.products` AS inventory
  ON website.productSKU = inventory.SKU
# gone through more than 50% of inventory for the month
WHERE SAFE_DIVIDE(website.total_ordered,inventory.stockLevel) >= .50
ORDER BY total_ordered DESC
```

What is the name of the top selling product and what percent of its inventory has been sold already?

- Youth Short Sleeve Tee Red with a restocking leadtime of 9
- Android Infant Short Sleeve Tee Pewter with 7 product orders out of 2 in stock
- Leather Journal-Black with 250 product orders out of 354 in stock

Append additional records

Your international team has already made in-store sales on 08/02/2017 which you want to record in your daily sales tables.

Create a new empty table to store sales by productSKU for 08/02/2017

For the schema, specify the following fields:

- table name is `ecommerce.sales_by_sku_20170802`
- `productSKU STRING`
- `total_ordered` as an `INT64` field

Possible Solution:

```
CREATE OR REPLACE TABLE ecommerce.sales_by_sku_20170802
(
productSKU STRING,
total_ordered INT64
);
```

Confirm you now have two date-shared sales tables - use the dropdown menu next to the **Sales_by_sku** table name in the table results, or refresh your browser to see it listed in the left menu.

Insert the sales record provided to you by your sales team:

```
INSERT INTO ecommerce.sales_by_sku_20170802
(productSKU, total_ordered)
VALUES('GGOEGHPA002910', 101)
```

Confirm the record appears by previewing the table - click on the table name to see the results.

Append together historical data

There are multiple ways to append together data that has the same schema. Two common ways are using UNIONs and table wildcards.

- Union is an SQL operator that appends together rows from different result sets.
- Table wildcards enable you to query multiple tables using concise SQL statements. Wildcard tables are available only in standard SQL.

Write a UNION query that will result in all records from the below two tables:

- `ecommerce.sales_by_sku_20170801`
- `ecommerce.sales_by_sku_20170802`

```
SELECT * FROM ecommerce.sales_by_sku_20170801
UNION ALL
SELECT * FROM ecommerce.sales_by_sku_20170802
```

Note: The difference between a `UNION` and `UNION ALL` is that a `UNION` will not include duplicate records.

What is a pitfall of having many daily sales tables? You will have to write many `UNION` statements chained together.

A better solution is to use the table wildcard filter and `_TABLE_SUFFIX` filter.

Write a query that uses the (*) table wildcard to select all records from `ecommerce.sales_by_sku_` for the year 2017.

Possible Solution:

```
SELECT * FROM `ecommerce.sales_by_sku_2017*`
```

Modify the previous query to add a filter to limit the results to just 08/02/2017.

Possible Solution:

```
SELECT * FROM `ecommerce.sales_by_sku_2017*`
WHERE _TABLE_SUFFIX = '0802'
```

Note: Another option to consider is to create a Partitioned Table which automatically can ingest daily sales data into the correct partition.