Creating a Data Warehouse Through Joins and Unions

cloudskillsboost.google/focuses/3640

GSP413

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

Setup and requirements

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

Access to a standard internet browser (Chrome browser recommended).

Note: Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

Time to complete the lab---remember, once you start, you cannot pause a lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab to avoid extra charges to your account.

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is the **Lab Details** panel with the following:
    - The **Open Google Console** button
    - Time remaining
    - The temporary credentials that you must use for this lab
    - Other information, if needed, to step through this lab
2. Click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.

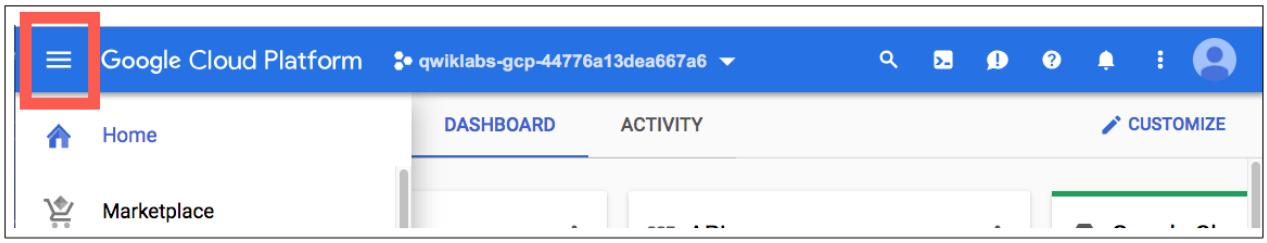
    Tip: Arrange the tabs in separate windows, side-by-side.

    Note: If you see the **Choose an account** dialog, click **Use Another Account**.
3. If necessary, copy the **Username** from the **Lab Details** panel and paste it into the **Sign in** dialog. Click **Next**.
4. Copy the **Password** from the **Lab Details** panel and paste it into the **Welcome** dialog. Click **Next**.

    Important: You must use the credentials from the left panel. Do not use your Google Cloud Skills Boost credentials. **Note:** Using your own Google Cloud account for this lab may incur extra charges.
5. Click through the subsequent pages:
    - Accept the terms and conditions.
    - Do not add recovery options or two-factor authentication (because this is a temporary account).
    - Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



The BigQuery console

Open the BigQuery console

1. In the Google Cloud Console, select **Navigation menu** > **BigQuery**.

The **Welcome to BigQuery in the Cloud Console** message box opens. This message box provides a link to the quickstart guide and the release notes.

2. Click **Done**.

The BigQuery console opens.

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 (`qwiklabs-gcp-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**.

Click *Check my progress* to verify the objective. Create a new dataset to store the tables

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.

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.

2. Click the **Schema** tab.

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
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?

Click *Check my progress* to verify the objective. Explore the product sentiment dataset

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
Click *Check my progress* to verify the objective. Join datasets to find insights

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

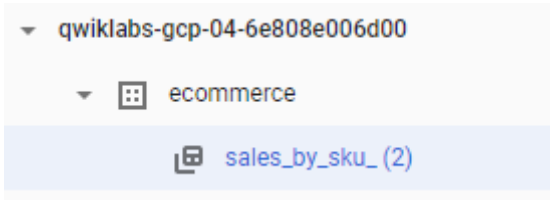
For the schema, specify the following fields:

- 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.
Click *Check my progress* to verify the objective. Append additional records

Congratulations!

You explored sample ecommerce data by creating reporting tables and then manipulating views using SQL JOINs and UNIONs.

Finish your Quest

This self-paced lab is part of the BigQuery for Data Warehousing Quest. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in this Quest and get immediate completion credit if you've taken this lab. See other available Quests.

Take your next lab

Continue with another lab in the Quest, for example Working with JSON, Arrays, and Structs in BigQuery, or check out these other labs:

- Exploring NCAA Data with BigQuery
- Cloud Composer: Copying BigQuery Tables Across Different Locations

Next steps / learn more

- Already have a Google Analytics account and want to query your own datasets in BigQuery? Follow this export guide.
- If you want to practice with more SQL syntax for JOINS, check out the BigQuery JOIN documentation.
- Try Google Dataset Search as a resource!

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. Our classes include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. Certifications help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated November 15, 2021

Lab Last Tested June 22, 2021

Copyright 2022 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.