Troubleshooting Common SQL Errors with BigQuery

experiment Lab schedule 50 minutes universal_currency_alt No cost

show_chart Introductory

Rate Lab

GSP408

Google Cloud Self-Paced Labs

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 you can take advantage of the pay-as-you-go model. BigQuery allows you to focus on analyzing data to find meaningful insights.

A newly available ecommerce dataset that has millions of Google Analytics records for the Google Merchandise Store has been loaded into BigQuery. You have a copy of that dataset for this lab and will explore the available fields and row for insights.

This lab steps you through the logic of troubleshooting queries. It provides activities within the context of a real-world scenario. Throughout the lab, imagine you're working with a new data analyst on your team, and they've provided you with their queries below to answer some questions on your ecommerce dataset. Use the answers to fix their queries to get a meaningful result.

What you'll learn

In this lab, you learn how to perform the following tasks:

- Pin projects to the BigQuery resource tree
- Use the BigQuery query editor and query validator to identify and troubleshoot SQL syntax and logic errors

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 Cloud 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 Cloud console** (or right-click and select **Open Link in Incognito Window** if you are running the Chrome browser).

    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** below and paste it into the **Sign in** dialog.

    ```
    "Username"
    ```
    content_c

    You can also find the **Username** in the **Lab Details** panel.

4. Click **Next**.

5. Copy the **Password** below and paste it into the **Welcome** dialog.

    ```
    "Password"
    ```
    content_c

    You can also find the **Password** in the **Lab Details** panel.

6. Click **Next**.

Important: You must use the credentials the lab provides you. Do not use your Google Cloud account credentials.

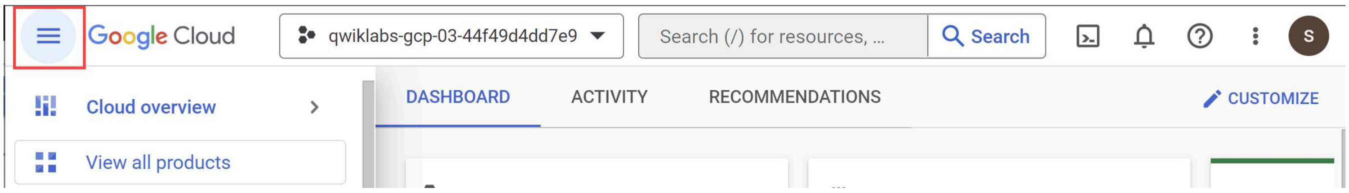
Note: Using your own Google Cloud account for this lab may incur extra charges.

7. 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 Google Cloud console opens in this tab.

Note: To view a menu with a list of Google Cloud products and services, click the **Navigation menu** at the top-left.



Task 1. Pin a project to the BigQuery resource tree

1. Click **Navigation menu** ☰ > **BigQuery**.

The Welcome to BigQuery in the Cloud Console message box opens.

Note: The Welcome to BigQuery in the Cloud Console message box provides a link to the quickstart guide and UI updates.

2. Click **Done**.

3. BigQuery public datasets are not displayed by default. To open the public datasets project, copy **data-to-insights**.

4. Click + **Add** > **Star a project by name** then paste the data-to-insights name.

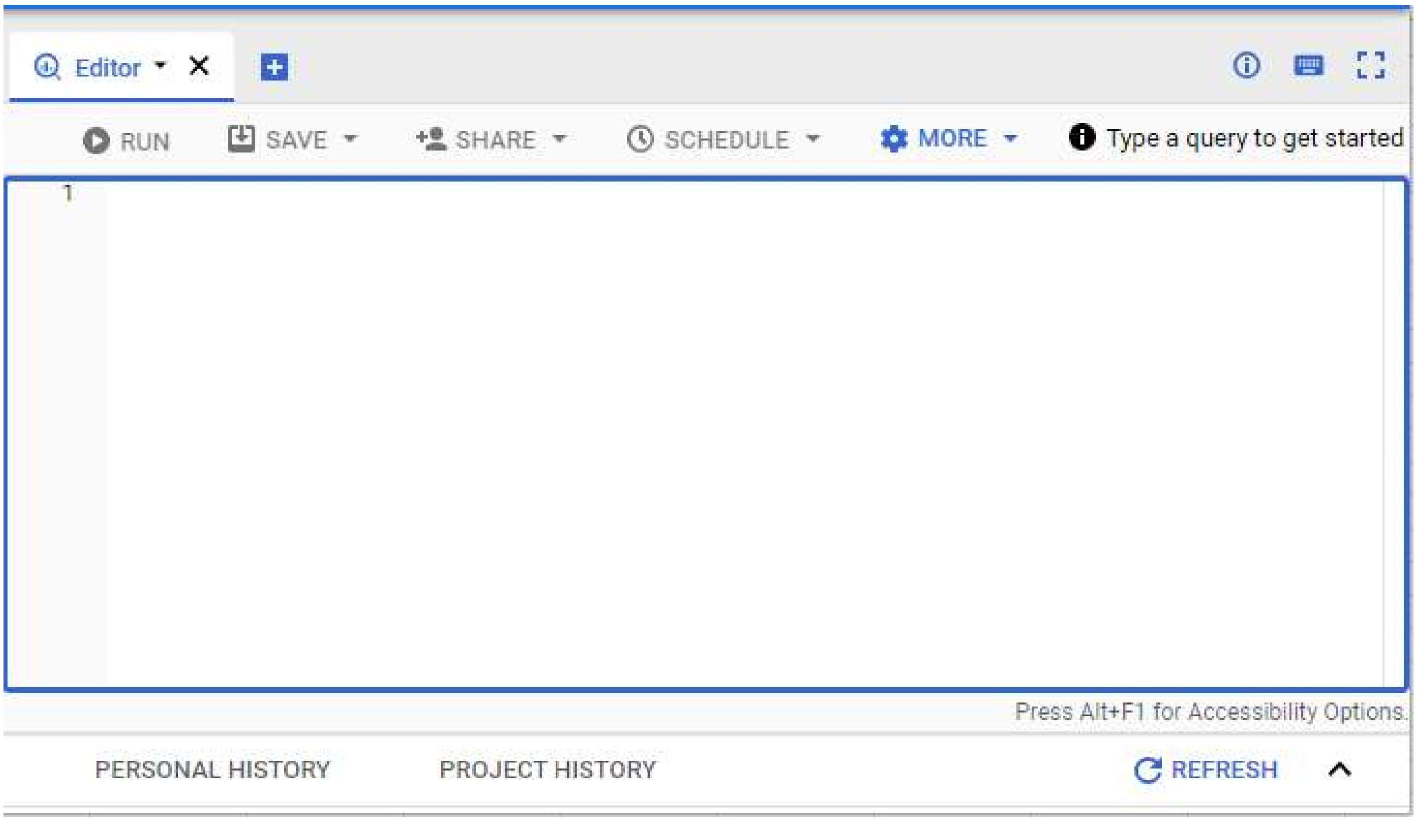
5. Click **Star**.

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

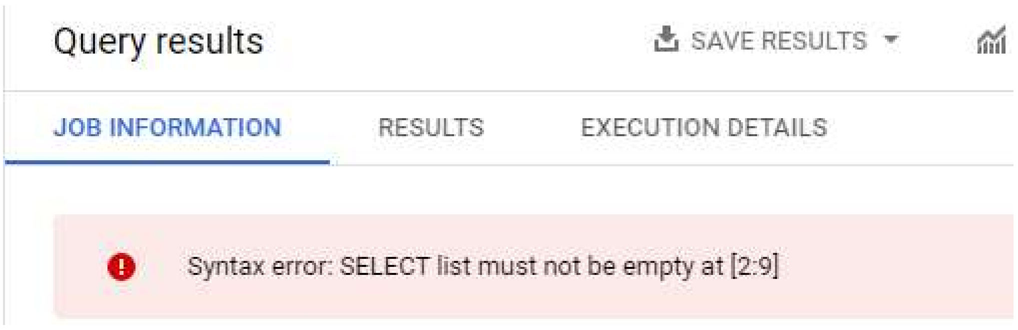
BigQuery query editor and query validator

For each activity in the following sections, this lab provides queries with common errors for you to troubleshoot. The lab directs you what to look at and suggests how to correct the syntax and return meaningful results.

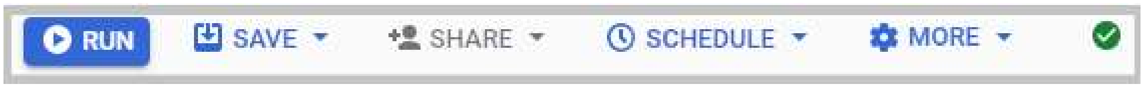
To follow along with the troubleshooting and suggestions, copy and paste the query into the BigQuery query editor. If there are errors, you see a red exclamation point at the line containing the error and in the query validator (bottom corner).



If you run the query with the errors, the query fails and the error is specified in the Job information.


Syntax error: SELECT list must not be empty at [2:9]

When the query is error free, you see a green checkmark in the query validator. When you see the green checkmark, click **Run** to run the query to view what you get for results.



Note: For information about syntax, see Standard SQL Query Syntax.

Task 2. Find the total number of customers who went through checkout

Your goal in this section is to construct a query that gives you the number of unique visitors who successfully went through the checkout process for your website. The data is in the rev_transactions table which your data analyst team has provided. They have also given you example queries to help you get started in your analysis but you're not sure they're written correctly.

Troubleshoot queries that contain query validator, alias, and comma errors

- Look at the below query and answer the following question:

```
#standardSQL
SELECT  FROM `data-to-inghts.ecommerce.rev_transactions` LIMIT 1000
```

content_c

What's wrong with the previous query to view 1000 items?

☐ There is a typo in the table name

check There is a typo in the dataset name

check We have not specified any columns in the SELECT

☐ We are using legacy SQL

Submit

- What about this updated query?

```
#standardSQL
SELECT * FROM [data-to-insights:ecommerce.rev_transactions] LIMIT 1000
```

content_c

What's wrong with the new previous query to view 1000 items?

☐ We have not specified any columns in the SELECT

☐ There is a typo in the table name

☐ There is a typo in the dataset name

check We are using legacy SQL

Submit

- What about this query that uses Standard SQL?

```
#standardSQL
SELECT FROM `data-to-insights.ecommerce.rev_transactions`
```

content_c

○

What is wrong with the previous query?

○ SELECT clause is returning all columns which leads to poor performance

○ We are still using legacy SQL

○ We are missing an ORDER BY clause

check Still no columns defined in SELECT

Submit

- What about now? This query has a column:

```
#standardSQL
SELECT
fullVisitorId
FROM `data-to-insights.ecommerce.rev_transactions`
```

content_c

○

What is wrong with the previous query?

check Without aggregations, limits, or sorting, this query is not insightful

☐ We are missing a LIMIT clause

check The page title is missing from the columns in SELECT

☐ We are missing a column alias

Submit

- What about now? The following query has a page title:

```
#standardSQL
SELECT fullVisitorId hits_page_pageTitle
FROM `data-to-insights.ecommerce.rev_transactions` LIMIT 1000
```

content_c

○

How many columns will the previous query return?

○ 2, columns named fullVisitorId and hits_page_pageTitle

check 1, a column named hits_page_pageTitle

○ 3 columns will be returned since we are missing a comma

○ 0, the query will return an error

Submit

- What about now? The missing comma has been corrected.

```
#standardSQL
SELECT
  fullVisitorId
  , hits_page_pageTitle
FROM `data-to-insights.ecommerce.rev_transactions` LIMIT 1000
```

content_c

Answer: This returns results, but are you sure visitors aren't counted twice? Also, returning only one row answers the question of how many unique visitors reached checkout. In the next section you find a way to aggregate your results.

Troubleshoot queries that contain logic errors, GROUP BY statements, and wildcard filters

- Aggregate the following query to answer the question: How many unique visitors reached checkout?

```
#standardSQL
SELECT
  fullVisitorId
  , hits_page_pageTitle
FROM `data-to-insights.ecommerce.rev_transactions` LIMIT 1000
```

content_c

- What about this? An aggregation function, `COUNT()`, was added:

```
#standardSQL
SELECT
COUNT(fullVisitorId) AS visitor_count
, hits_page_pageTitle
FROM `data-to-insights.ecommerce.rev_transactions`
```

content_c

○

What is wrong with the previous query?

check The COUNT() function does not de-deduplicate the same fullVisitorId check It is missing a GROUP BY clause

☐ Nothing, it executes correctly

☐ A COUNT() function is used when SUM() should be used instead

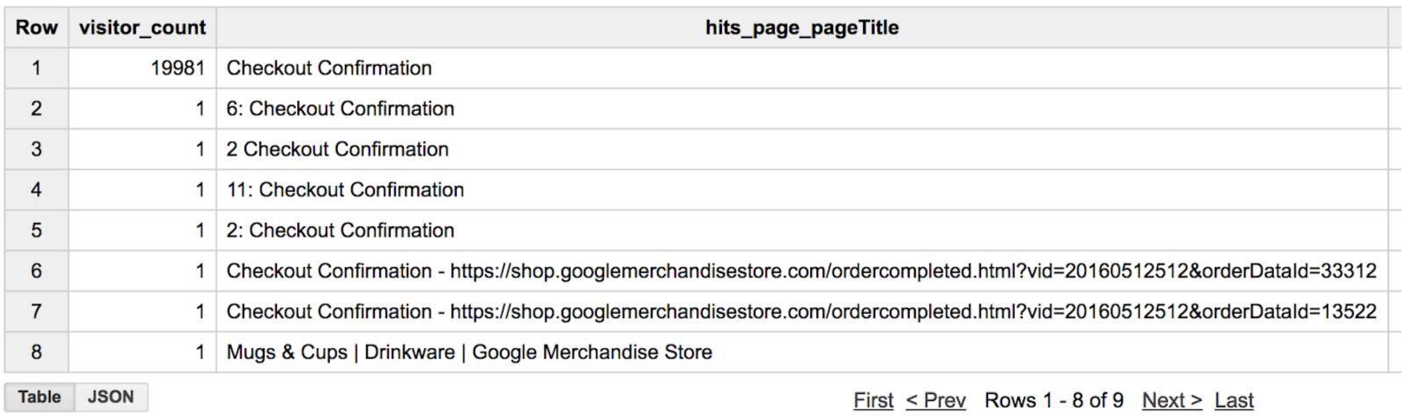
Submit

- In this next query, `GROUP BY` and `DISTINCT` statements were added:

```
#standardSQL
SELECT
COUNT(DISTINCT fullVisitorId) AS visitor_count
, hits_page_pageTitle
FROM `data-to-insights.ecommerce.rev_transactions`
GROUP BY hits_page_pageTitle
```

content_c

Results



Row	visitor_count	hits_page_pageTitle
1	19981	Checkout Confirmation
2	1	6: Checkout Confirmation
3	1	2 Checkout Confirmation
4	1	11: Checkout Confirmation
5	1	2: Checkout Confirmation
6	1	Checkout Confirmation - https://shop.googlemerchandisestore.com/ordercompleted.html?vid=20160512512&orderDataId=33312
7	1	Checkout Confirmation - https://shop.googlemerchandisestore.com/ordercompleted.html?vid=20160512512&orderDataId=13522
8	1	Mugs & Cups | Drinkware | Google Merchandise Store

Great! The results are good, but they look strange.

- Filter to just "Checkout Confirmation" in the results:

```
#standardSQL
SELECT
COUNT(DISTINCT fullVisitorId) AS visitor_count
, hits_page_pageTitle
FROM `data-to-insights.ecommerce.rev_transactions`
WHERE hits_page_pageTitle = "Checkout Confirmation"
GROUP BY hits_page_pageTitle
```

content_c

Click **Check my progress** to verify the objective.

Find the total number of customers went through checkout

Check my progress

Task 3. List the cities with the most transactions with your ecommerce site

Troubleshoot ordering, calculated fields, and filtering after aggregating errors

1. Complete the partially written query:

content_c

```
SELECT
geoNetwork_city,
totals_transactions,
COUNT( DISTINCT fullVisitorId) AS distinct_visitors
FROM
`data-to-insights.ecommerce.rev_transactions`
GROUP BY
```

Possible solution:

content_c

```
#standardSQL
SELECT
geoNetwork_city,
SUM(totals_transactions) AS totals_transactions,
COUNT( DISTINCT fullVisitorId) AS distinct_visitors
FROM
```

```
`data-to-insights.ecommerce.rev_transactions`
GROUP BY geoNetwork_city
```

2. Update your previous query to order the top cities first.

Which city had the most distinct visitors? Ignore the value: 'not available in this demo dataset'

- ◯ Los Angeles
- ◯ Austin
- ◯ San Jose
- ◯ Mountain View

Submit

Possible solution:

```
#standardSQL
SELECT
geoNetwork_city,
SUM(totals_transactions) AS totals_transactions,
COUNT( DISTINCT fullVisitorId) AS distinct_visitors
FROM
`data-to-insights.ecommerce.rev_transactions`
GROUP BY geoNetwork_city
ORDER BY distinct_visitors DESC
```

content_c

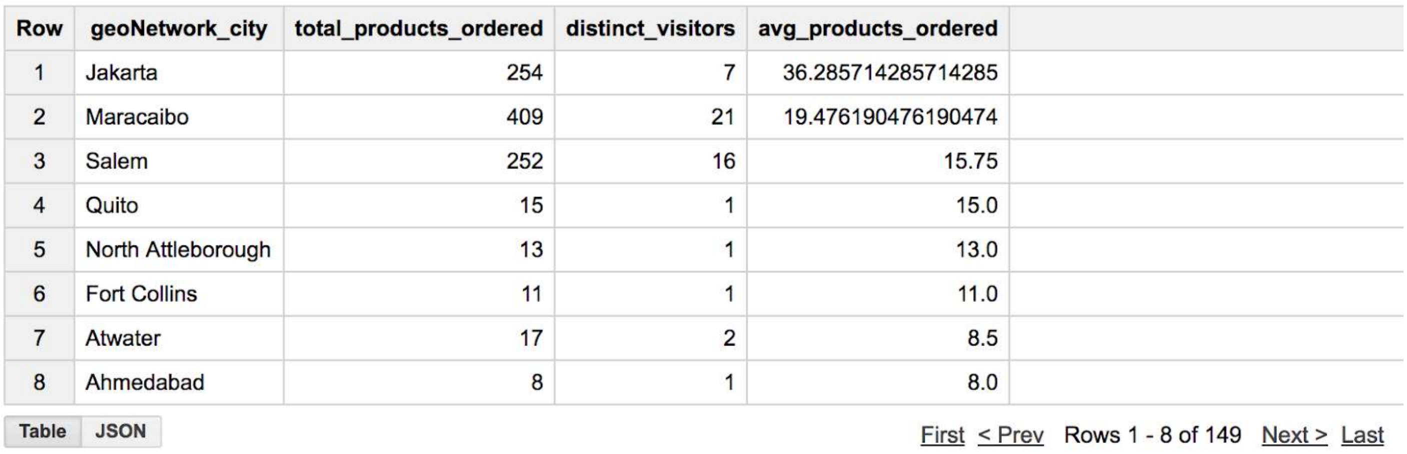
3. Update your query and create a new calculated field to return the average number of products per order by city.

Possible solution:

```
#standardSQL
SELECT
geoNetwork_city,
SUM(totals_transactions) AS total_products_ordered,
COUNT( DISTINCT fullVisitorId) AS distinct_visitors,
SUM(totals_transactions) / COUNT( DISTINCT fullVisitorId) AS
avg_products_ordered
FROM
`data-to-insights.ecommerce.rev_transactions`
GROUP BY geoNetwork_city
ORDER BY avg_products_ordered DESC
```

content_c

Results



Row	geoNetwork_city	total_products_ordered	distinct_visitors	avg_products_ordered
1	Jakarta	254	7	36.285714285714285
2	Maracaibo	409	21	19.476190476190474
3	Salem	252	16	15.75
4	Quito	15	1	15.0
5	North Attleborough	13	1	13.0
6	Fort Collins	11	1	11.0
7	Atwater	17	2	8.5
8	Ahmedabad	8	1	8.0

Table JSON First < Prev Rows 1 - 8 of 149 Next > Last

Filter your aggregated results to only return cities with more than 20 avg_products_ordered.

- What's wrong with the following query?

```
#standardSQL
SELECT
geoNetwork_city,
SUM(totals_transactions) AS total_products_ordered,
COUNT( DISTINCT fullVisitorId) AS distinct_visitors,
SUM(totals_transactions) / COUNT( DISTINCT fullVisitorId) AS
avg_products_ordered
FROM
`data-to-insights.ecommerce.rev_transactions`
WHERE avg_products_ordered > 20
GROUP BY geoNetwork_city
ORDER BY avg_products_ordered DESC
```

content_c

What is wrong with the previous query?

☐ You cannot divide non-similar aggregate functions

☐ Nothing, it executes correctly

check You cannot filter on aliased fields within the `WHERE` clause

check You cannot filter aggregated fields in the `WHERE` clause (use `HAVING` instead)

Submit

Possible solution:

```
#standardSQL
SELECT
geoNetwork_city,
SUM(totals_transactions) AS total_products_ordered,
COUNT( DISTINCT fullVisitorId) AS distinct_visitors,
SUM(totals_transactions) / COUNT( DISTINCT fullVisitorId) AS
avg_products_ordered
FROM
`data-to-insights.ecommerce.rev_transactions`
GROUP BY geoNetwork_city
HAVING avg_products_ordered > 20
ORDER BY avg_products_ordered DESC
```

content_c

Click **Check my progress** to verify the objective.

List the cities with the most transactions with your ecommerce site

Check my progress

Task 4. Find the total number of products in each product category

Find the top selling products by filtering with NULL values

1. What's wrong with the following query? How can you fix it?

```
#standardSQL
SELECT hits_product_v2ProductName, hits_product_v2ProductCategory
FROM `data-to-insights.ecommerce.rev_transactions`
GROUP BY 1,2
```

content_c

What is wrong with the previous query?

☐ Nothing, it executes correctly

☐ There is a typo in the column name

check Large GROUP BYs really hurt performance (consider filtering first and/or using aggregation functions)

check No aggregate functions are used

Submit

2. What is wrong with the following query?

```
#standardSQL
SELECT
COUNT(hits_product_v2ProductName) as number_of_products,
hits_product_v2ProductCategory
FROM `data-to-insights.ecommerce.rev_transactions`
WHERE hits_product_v2ProductName IS NOT NULL
```

content_c

```
GROUP BY hits_product_v2ProductCategory
ORDER BY number_of_products DESC
```

What is wrong with the previous query which lists products?

- ○ Nothing, the query executes correctly
- check The COUNT() function is not the distinct number of products in each category
- ○ The GROUP BY contains an incorrect column
- ○ The WHERE clause should include NULL Product Names

Submit

3. Update the previous query to only count distinct products in each product category.

Possible solution:

content_c

```
#standardSQL
SELECT
COUNT(DISTINCT hits_product_v2ProductName) as number_of_products,
hits_product_v2ProductCategory
FROM `data-to-insights.ecommerce.rev_transactions`
WHERE hits_product_v2ProductName IS NOT NULL
GROUP BY hits_product_v2ProductCategory
ORDER BY number_of_products DESC
LIMIT 5
```

Which category has the most distinct number of products offered?

close ~~Office~~

check (not set)

○ ${productitem.product.origCatName}

close ~~Electronics~~

Submit

Note:

- (not set) could indicate the product has no category
- ${productitem.product.origCatName} is front-end code to render the category which may indicate the Google Analytics tracking script is firing before the page is fully-rendered

Click **Check my progress** to verify the objective.

○

Find the total number of products in each product category

Check my progress

Congratulations!

You troubleshot and fixed broken queries in BigQuery standard SQL. Remember to use the Query Validator for incorrect query syntax but also to be critical of your query results even if your query executes successfully.

Next steps / Learn more

- Explore BigQuery Public Datasets.
- Have a Google Analytics account and want to query your own datasets in BigQuery? Follow this export guide.
- Check out 15 Awesome things you probably didn’t know about BigQuery.
- Check out other labs to learn more about BigQuery:
    - Exploring Your Ecommerce Dataset with SQL in BigQuery
    - Weather Data in BigQuery
    - Creating Date-Partitioned Tables in BigQuery

Google Cloud training and 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: January 19, 2024

Lab Last Tested: August 28, 2023