Lab: Troubleshooting Common SQL Errors with Google BigQuery

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.

We have a newly available ecommerce dataset that has millions of Google Analytics records for the Google Merchandise Store loaded into BigQuery. We've made a copy of that dataset for this lab and will be exploring the available fields and row for insights.

In this lab, we will troubleshoot common SQL errors

What you'll learn

- Using BigQuery
- Query the data-to-insights public dataset
- Practice using the Query Validator
- Troubleshoot syntax and logical SQL errors

What you'll need

- A Google Cloud Platform Project

- A Browser, such Chrome or Firefox

Setup and Requirements

Qwiklabs setup

What you'll need

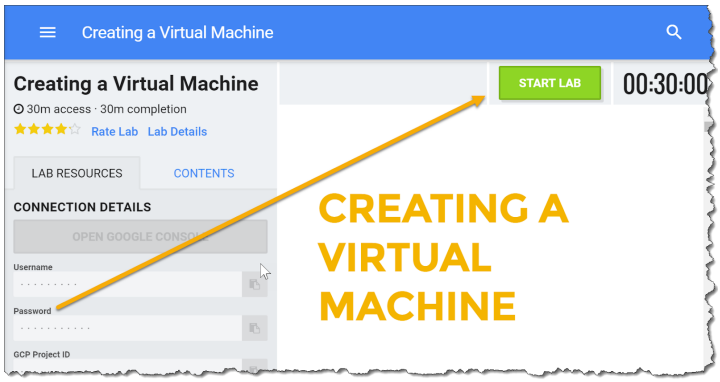
To complete this lab, you'll need:

- Access to a standard internet browser (Chrome browser recommended).
- Time. Note the lab's **Completion** time in Qwiklabs, which is an estimate of the time it should take to complete all steps. Plan your schedule so you have time to complete the lab. Once you start the lab, you will not be able to pause and return later (you begin at step 1 every time you start a lab).
- You do NOT need a Google Cloud Platform account or project. An account, project and associated resources are provided to you as part of this lab.
- If you already have your own GCP account, make sure you do not use it for this lab.
- If your lab prompts you to log into the console, **use only the student account provided to you by the lab**. This prevents you from incurring charges for lab activities in your personal GCP account.

Use a new Incognito window (Chrome) or another browser for the Qwiklabs session. Alternatively, you can log out of all other Google / Gmail accounts before beginning the labs.

Start your lab

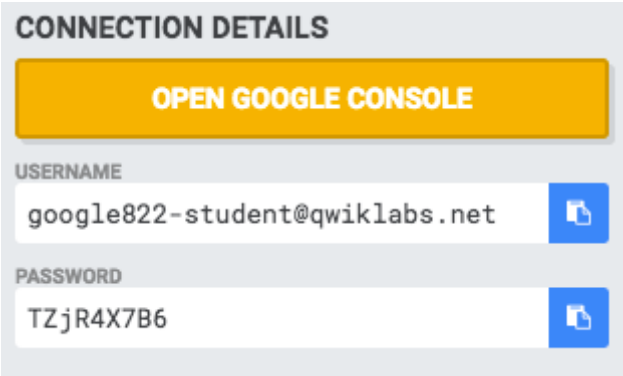
When you are ready, click **Start Lab**. You can track your lab's progress with the status bar at the top of your screen.



> **Important**: What is happening during this time?
>
> Your lab is spinning up GCP resources for you behind the scenes, including an account, a project, resources within the project, and permission for you to control the resources you will need to run the lab. This means that instead of spending time manually setting up a project and building resources from scratch as part of your lab, you can begin learning more quickly.

Find Your Lab's GCP Username and Password

To access the resources and console for this lab, locate the Connection Details panel in Qwiklabs. Here you will find the account ID and password for the account you will use to log in to the Google Cloud Platform:

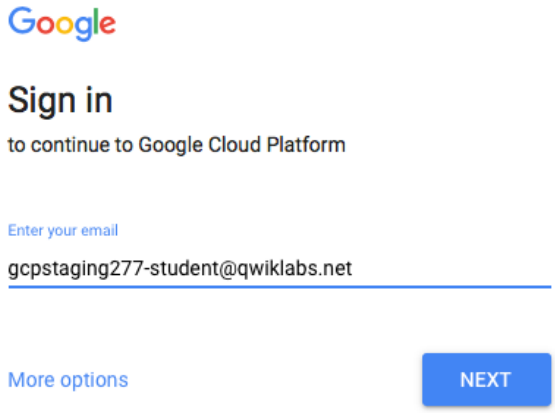


If your lab provides other resource identifiers or connection-related information, it will appear on this panel as well.

Google Cloud Platform Console

Log in to Google Cloud Console

Using the Qwiklabs browser tab/window (preferably in Incognito mode) or the separate browser you are using for the Qwiklabs session, copy the Username from the Connection Details panel and click the orange "Open Google Console" button. Paste in the Username, and then the Password as prompted:

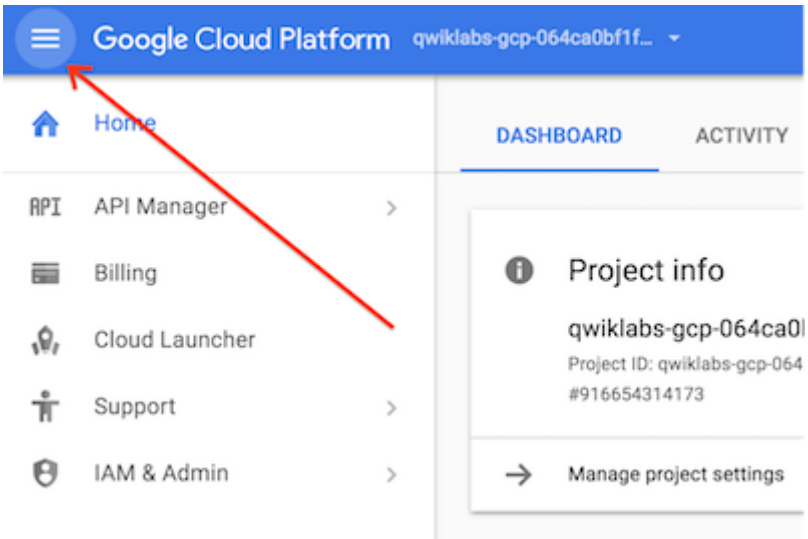


Accept the terms and conditions.

Since this is a temporary account, which you will only have access to for this one lab:

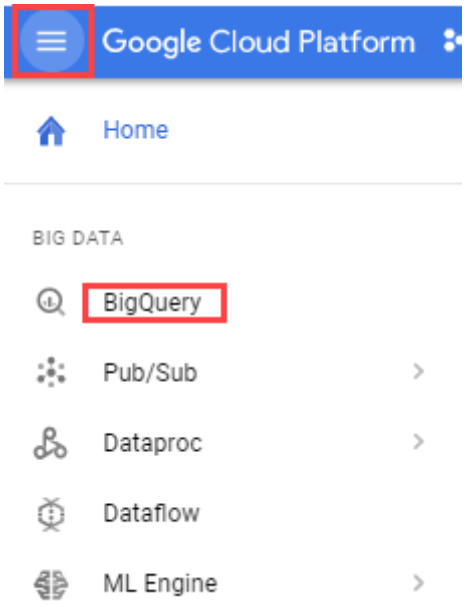
- Do not add recovery options
- Do not sign up for free trials

Note: You can view the menu with a list of GCP Products and Services by clicking the button at the top-left next to "Google Cloud Platform".



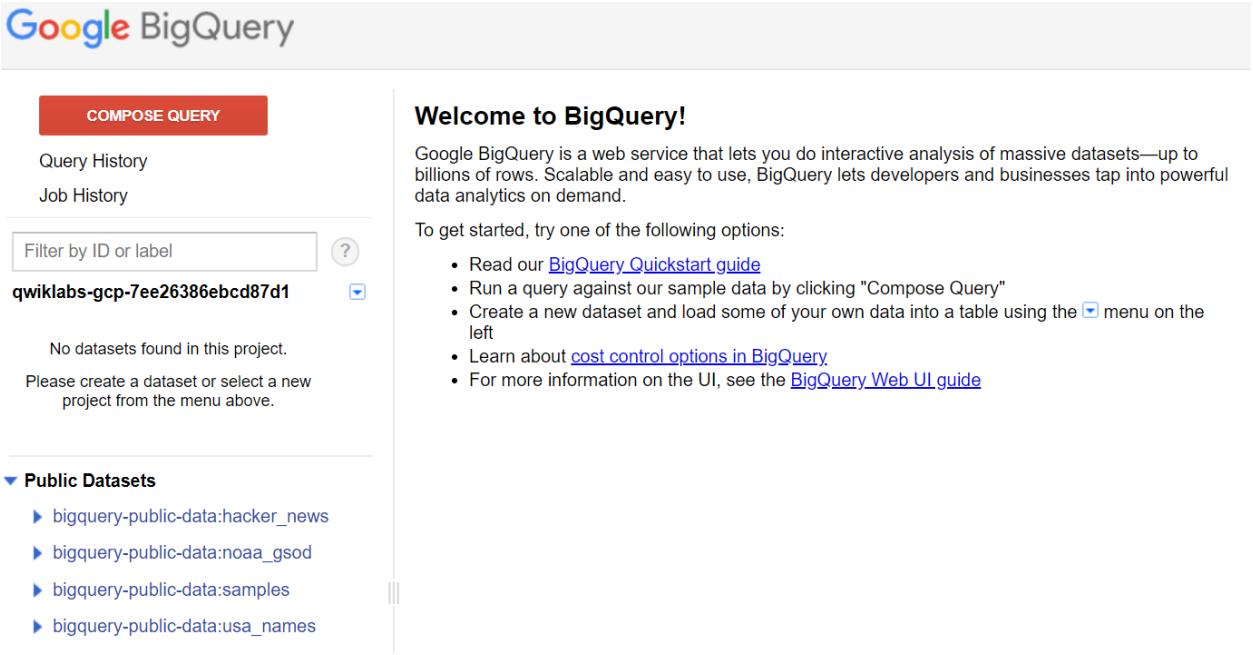
Open BigQuery Console

In the Google Cloud Console, from the Home menu, scroll down to the bottom, and click **BigQuery:**



You may be prompted to enter your lab account's password again.

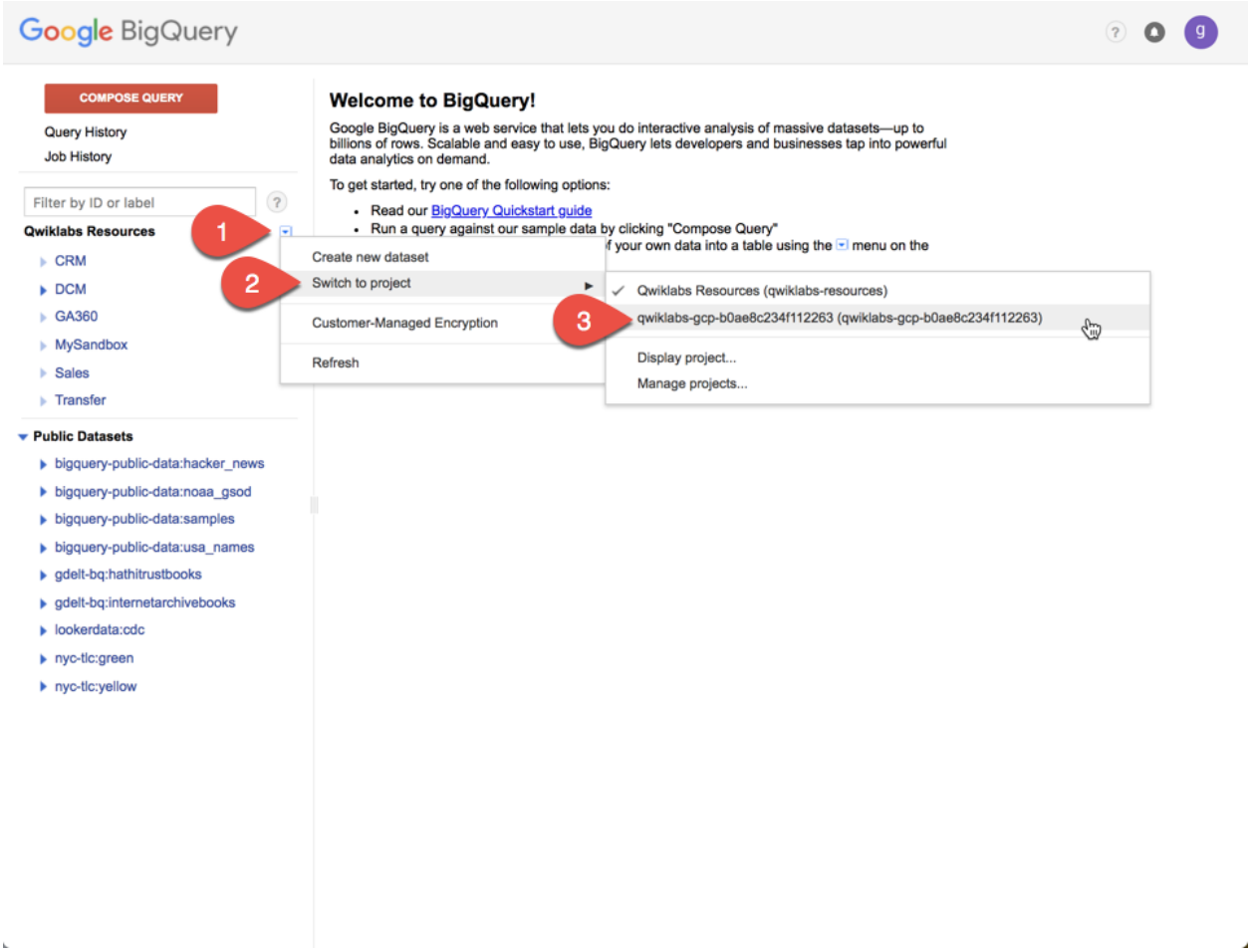
The BigQuery console will open in a new browser tab:



But there's nothing in here! Luckily, there are tons of Open Datasets available in BigQuery for you to query, and of course you can upload your own data, which you'll do in the next section.

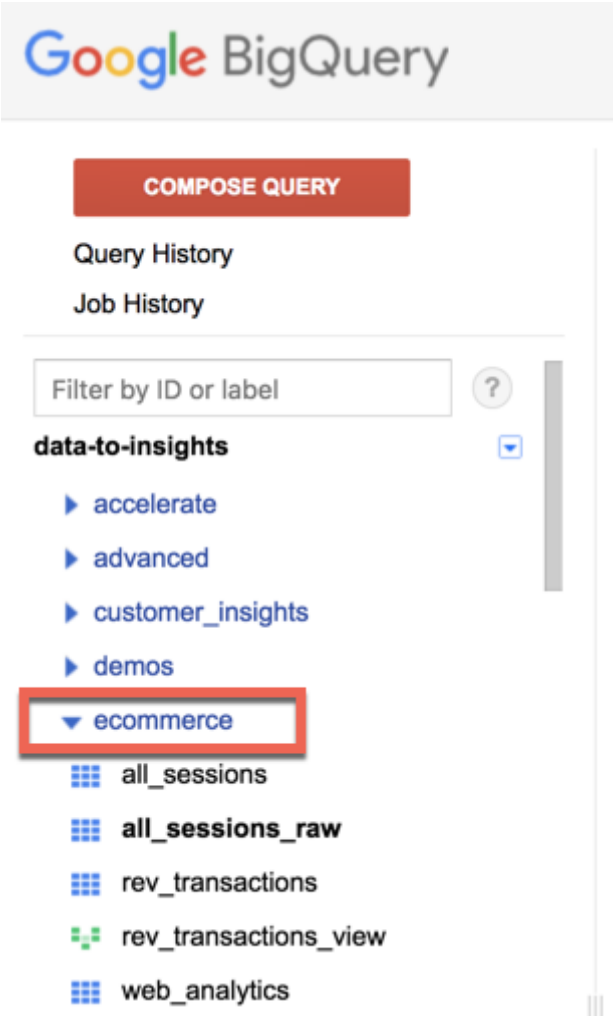
Access the Course Dataset

Ensure your BigQuery project is set to qwiklabs-gcp-123abc and not Qwiklabs Resources. If you need to switch between projects, click the drop down arrow next to the project and Switch to project as shown below:



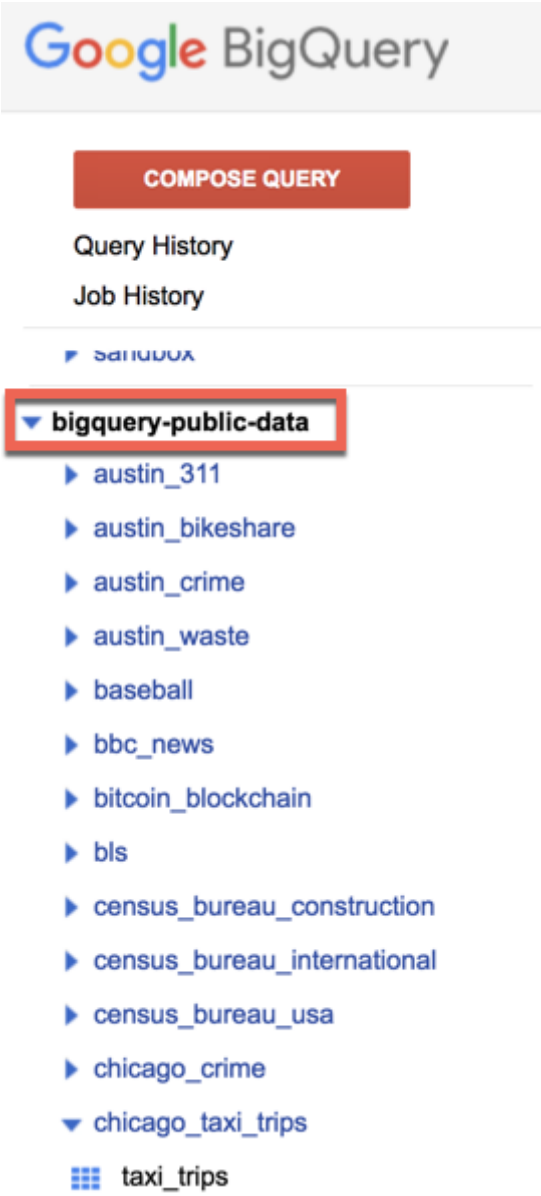
Once BigQuery is open, click on the below direct link to bring in the public data-to-insights project into your BigQuery projects panel:

- https://bigquery.cloud.google.com/table/data-to-insights:ecommerce.all_sessions_raw



Optionally, you can use this below link to bring in the bigquery-public-data project as well if you want to explore other datasets:

- https://bigquery.cloud.google.com/table/bigquery-public-data:chicago_taxi_trips.taxi_trips



The schema for the data-to-insights ecommerce dataset is below. Keep this open in a new tab for reference:

- https://support.google.com/analytics/answer/3437719?hl=en

Google Describe your issue

Field Name	Data Type	Description
clientId	STRING	Unhashed version of the Client ID for a given user associated with any given visit/session.
fullVisitorId	STRING	The unique visitor ID (also known as client ID).
visitorId	NULL	This field is deprecated. Use "fullVisitorId" instead.
userId	STRING	Overridden User ID sent to Analytics.
visitNumber	INTEGER	The session number for this user. If this is the first session, then this is set to 1.
visitId	INTEGER	An identifier for this session. This is part of the value usually stored as the _utmb cookie. This is only unique to the user. For a completely unique ID, you should use a combination of fullVisitorId and visitId.

The complete BigQuery SQL reference guide is here as an additional resource

- https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax

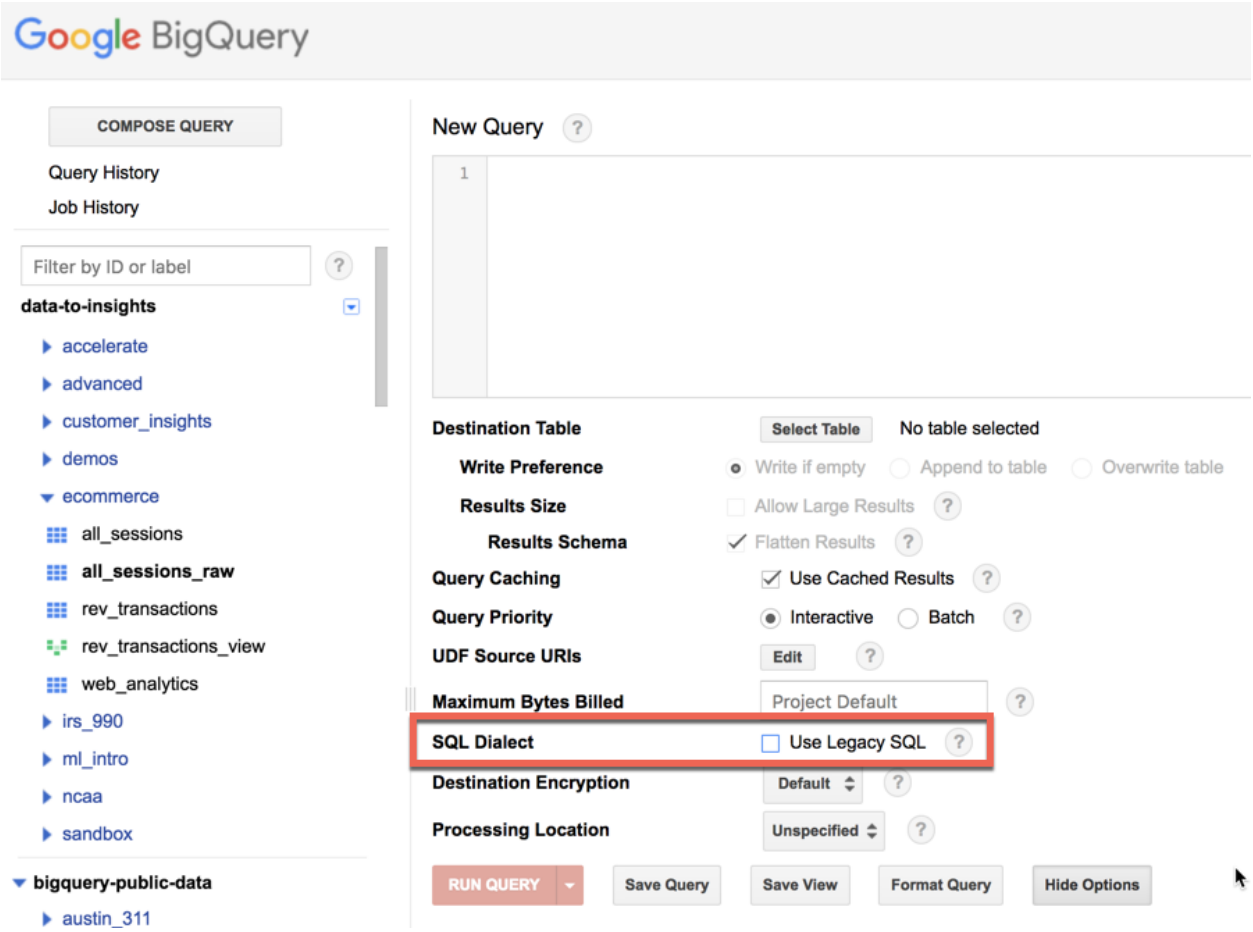
Lastly, ensure Legacy SQL is disabled by clicking:

. Compose Query

. Show Options

. Untick "Use Legacy SQL" (or leave it deselected)

. Hide Options



Background

You're working with a new data analyst on your team and they've provided you with their below queries to answer some questions on your ecommerce dataset. Validate and fix their queries to get the correct result.

Question 1: Find the number unique visitors who reached the checkout confirmation page in the rev_transactions table

Query Validator, Aliases, and Commas

What's wrong with the below query?

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

Answer: Typo in table name and no columns defined in SELECT

What about now? I've updated my query

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

Answer: The table name is escaped using Legacy SQL syntax (brackets) instead of Standard SQL `` (backticks)

What about now? I'm using Standard SQL

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

Answer: No columns are being SELECTED

What about now? I've added a column

```
#standardSQL
SELECT
fullVisitorId
```

```
FROM `data-to-
insights.ecommerce.rev_transactions`
```

Answer: This query will run but you likely don't need all that data returned and you're not including the page title anywhere in the query.

What about now? I've added page title

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

Answer: This query will run but only one column will return and it will be aliased in a very confusing way

What about now? I've found the missing comma and corrected it

```
#standardSQL
SELECT
  fullVisitorId
  , hits_page_pageTitle
FROM `data-to-
insights.ecommerce.rev_transactions` LIMIT
1000
# Bonus: Where should commas go?
# https://www.youtube.com/watch?
time_continue=2&v=ppioMSOi2Ho
```

Answer: This returns results but are you sure we're not double counting visitors? Also we only need one row returned to answer the question of "how many unique visitors reached checkout". Find a way to aggregate your results.

Logical Errors, Group Bys, Wildcard Filters

Aggregate the query to answer "how many unique visitors reached checkout":

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

What about this? I've added a COUNT()

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

Answer: Yes but it is missing a GROUP BY and DISTINCT

Adding GROUP BY and DISTINCT

```
#standardSQL
SELECT
COUNT(
```

DISTINCT

```
fullVisitorId) AS visitor_count
, hits_page_pageTitle
FROM `data-to-
insights.ecommerce.rev_transactions`
GROUP BY hits_page_pageTitle
```

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

Table JSON First < Prev Rows 1 - 8 of 9 Next > Last

Great! We're getting good results but they look strange. Now, 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
```

Extra Credit:

- Update the above query to accept "checkout" and "8 - Checkout" as rows returned as well.
- Find any pages in the rev_transactions table where the page title is not "checkout confirmation". Are there any? What does that mean about our data?

Question 2: Which cities that have the most transactions with our ecommerce site?

Ordering, Calculated Fields, Filtering after Aggregating

Complete the partially written query:

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

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
```

Update your previous query to order the top cities first

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
```

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
```

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 below 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
```

Answer: You cannot filter aggregated fields in the WHERE clause (use HAVING instead) and additionally you cannot filter on aliased fields within the WHERE clause.

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
```

Question 3: Find total number of products in each product category

Find Top Selling Products, Filtering with NULL values

What's wrong with the below query? How can it be fixed?

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

Answer:

- Large GROUP BYs really hurt performance (consider filtering first and / or using aggregation functions). More on this when we discuss query performance.
- No aggregate functions are used

What is wrong with the below 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
GROUP BY hits_product_v2ProductCategory
ORDER BY number_of_products DESC
```

Answer: The COUNT is not the distinct number of products in each category. The query will run but the result will not be the answer to our original question.

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

Possible solution:

```
#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
```

Extra credit:

- Does this represent all products available for sale? Why or why not? How could you address this?

Congratulations!

You've successfully fixed all of your data analyst's broken queries! Keep progressing through these labs to practice more SQL and stay tuned for advanced BigQuery concepts like how to setup your own data warehouse.

Already have a Google Analytics account and want to query your own datasets in BigQuery? Follow this export guide.