Lab: Explore your Ecommerce Dataset with SQL in 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 find and query the data-to-insights ecommerce dataset using BigQuery.

What you'll learn

- Using BigQuery
- Query the data-to-insights public dataset
- Writing and executing queries

What you'll need

- A Google Cloud Platform Project
- A Browser, such Chrome or Firefox

Setup and Requirements

Qwiklabs setup

What you'll need

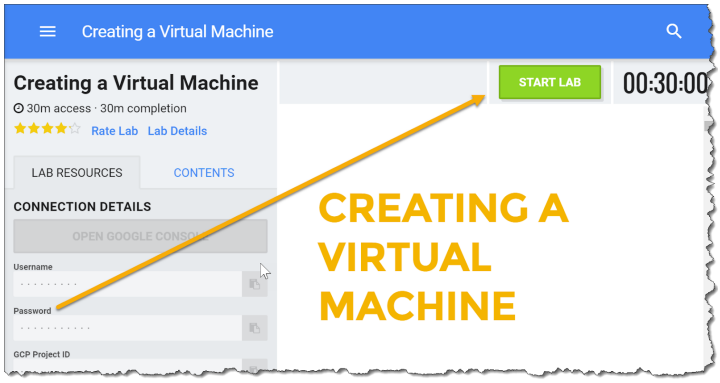
To complete this lab, you'll need:

- 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.

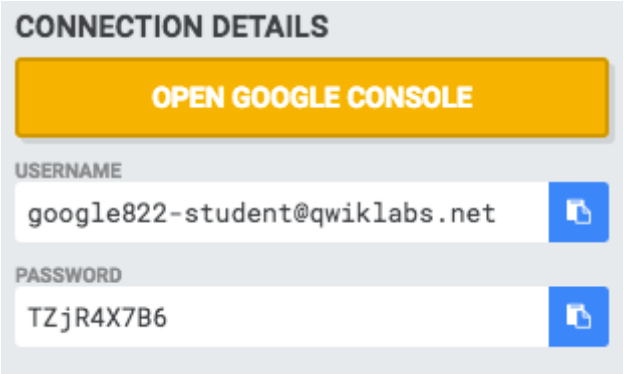


Important: What is happening during this time?

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:

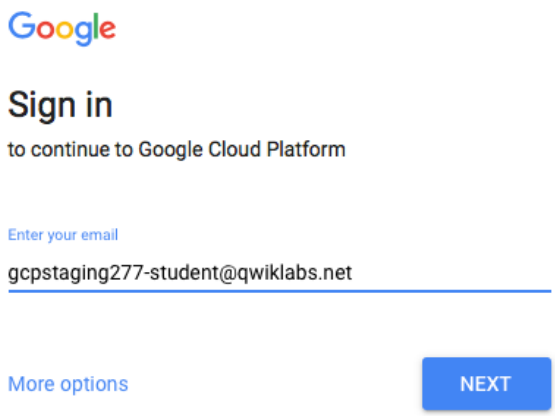


If your lab provides other resource identifiers or connection-related information, it will appear on this panel as well.

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:

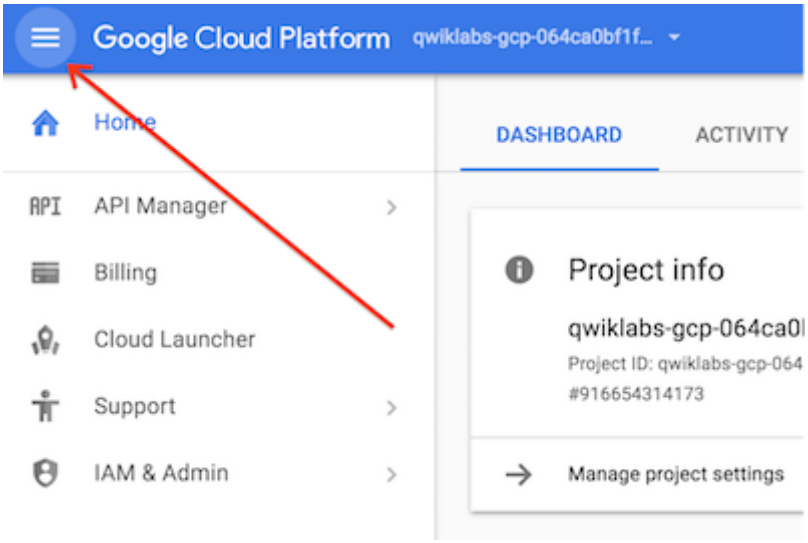


Accept the terms and conditions.

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

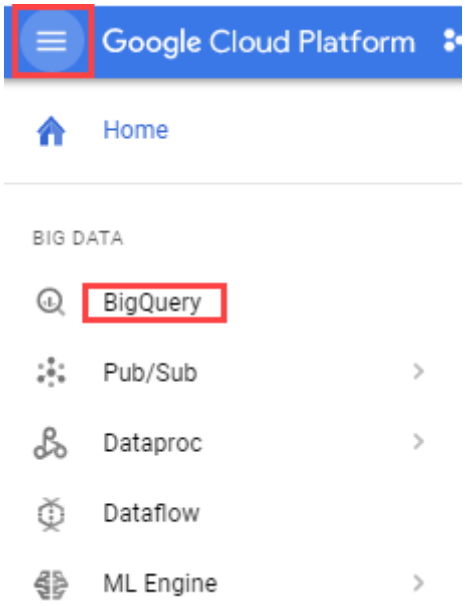
- Do not add recovery options
- Do not sign up for free trials

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".



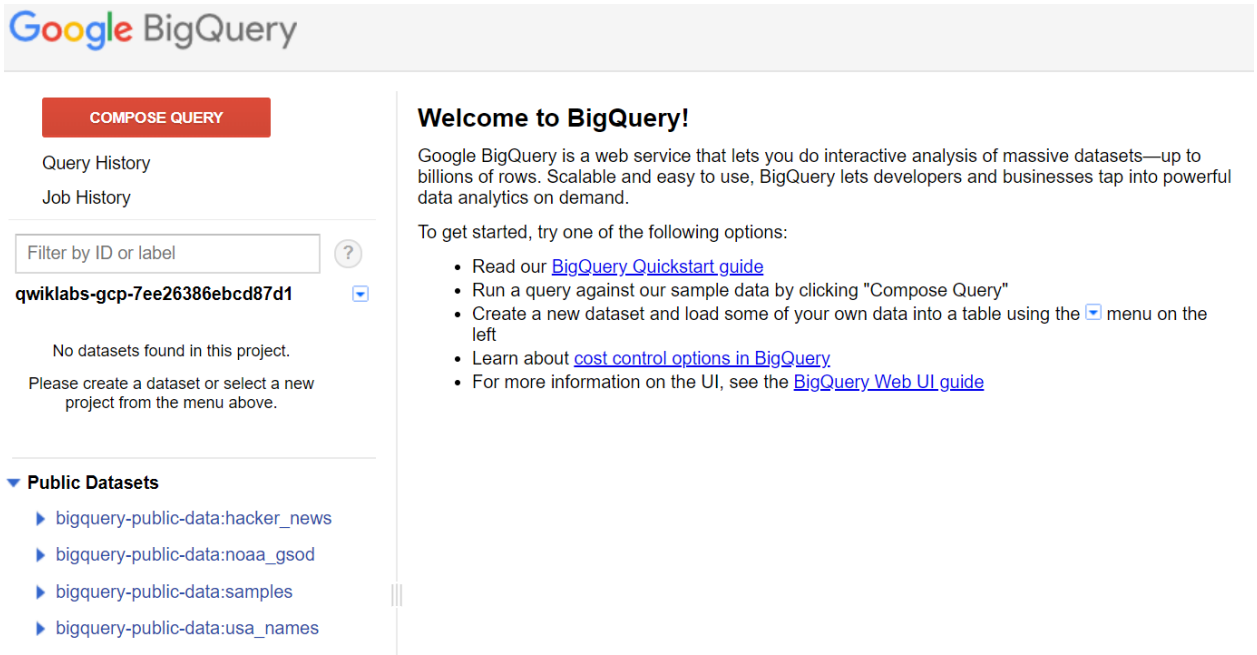
Open BigQuery Console

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.

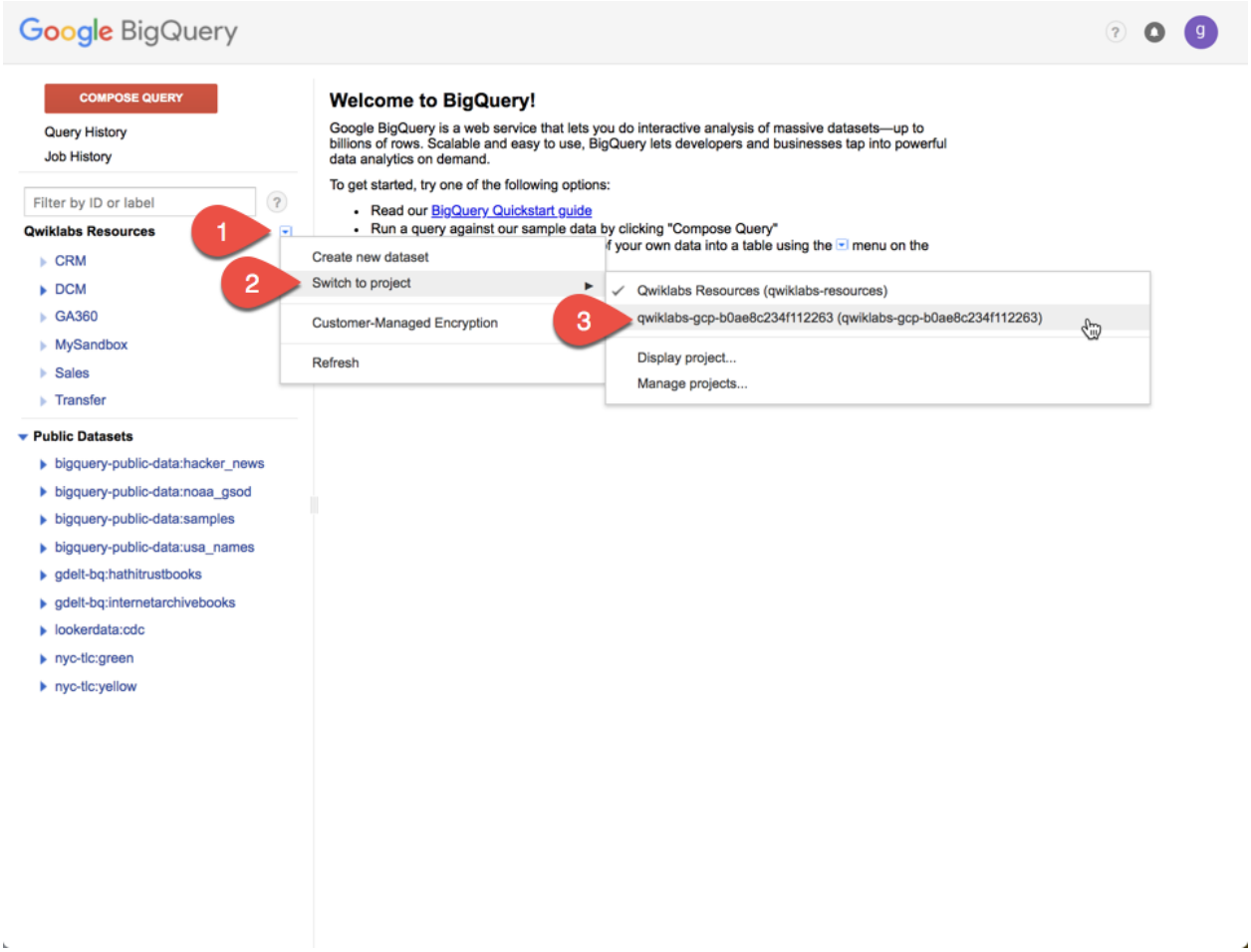
The BigQuery console will open in a new browser tab:



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.

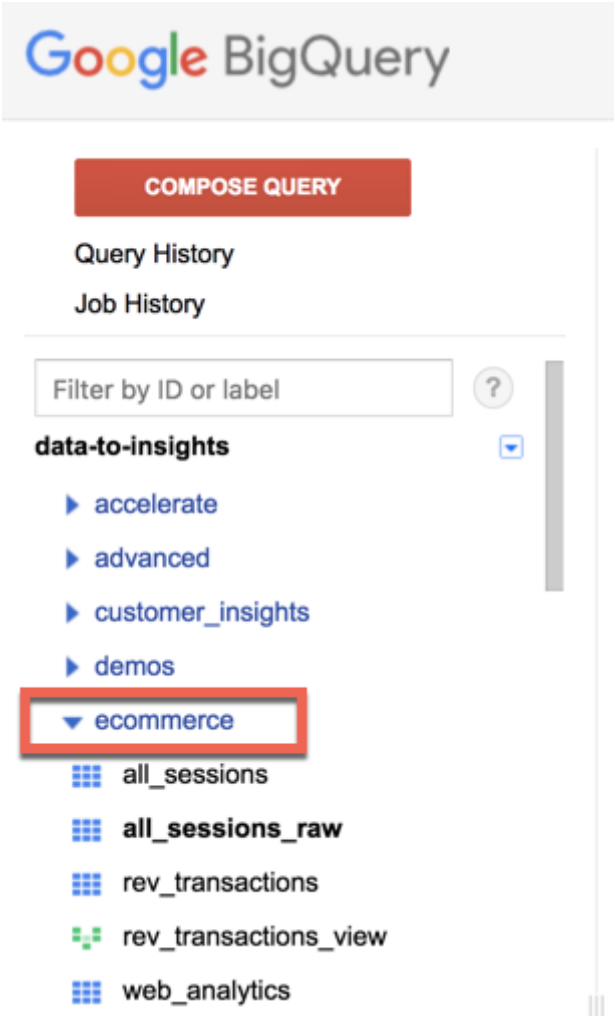
Access the Course Dataset

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:



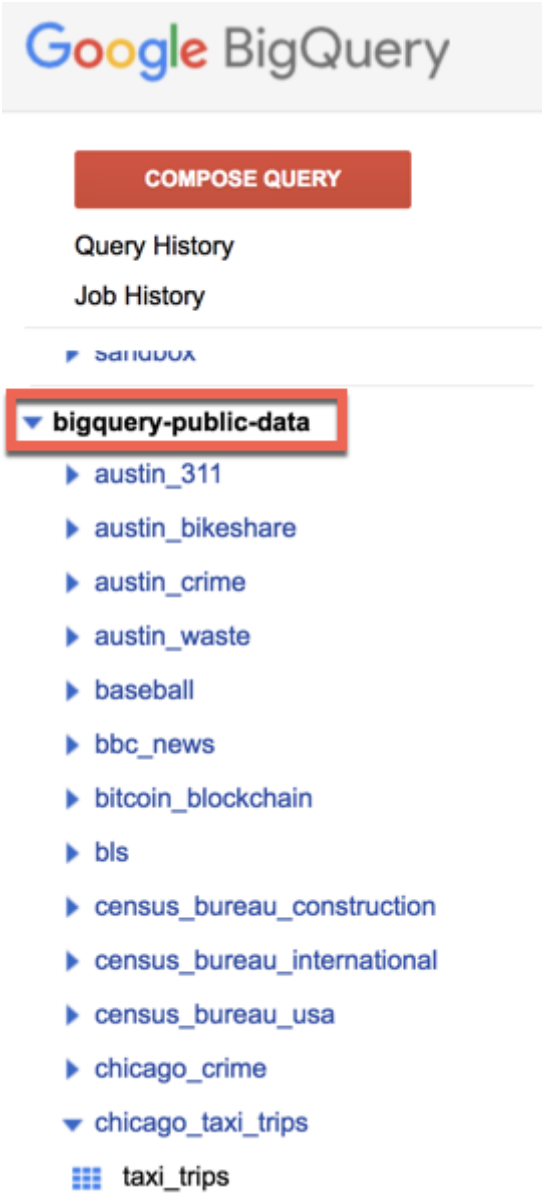
Once BigQuery is open, click on the below direct link to bring in the public data-to-insights project into your BigQuery projects panel:

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



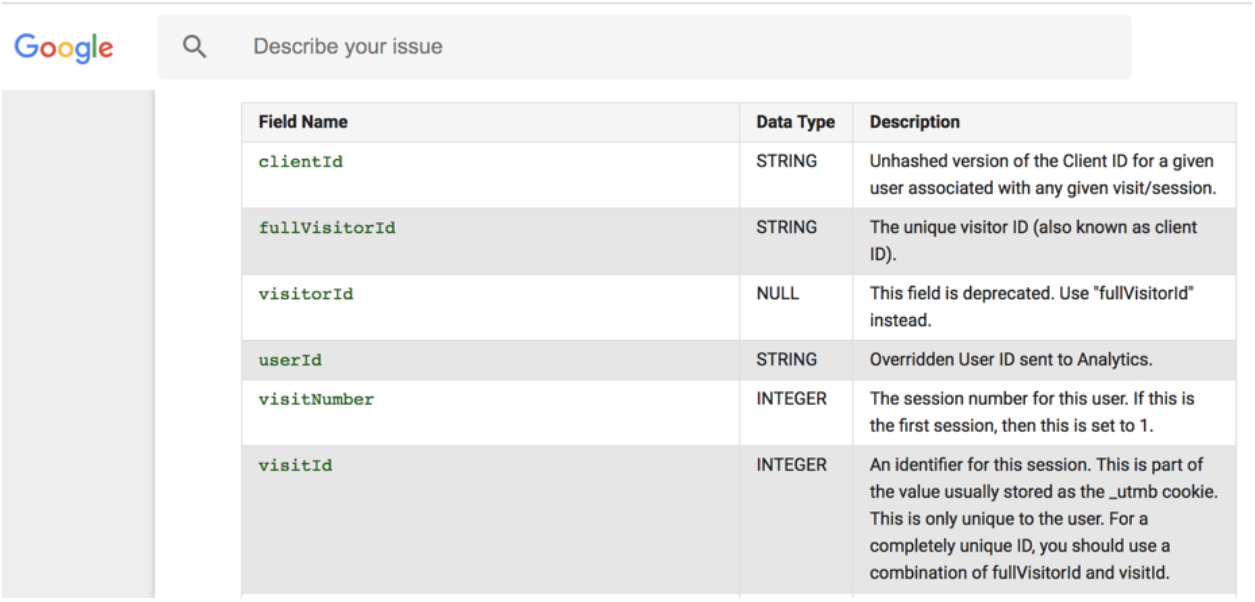
Optionally, you can use this below link to bring in the bigquery-public-data project as well if you want to explore other datasets:

- 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



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

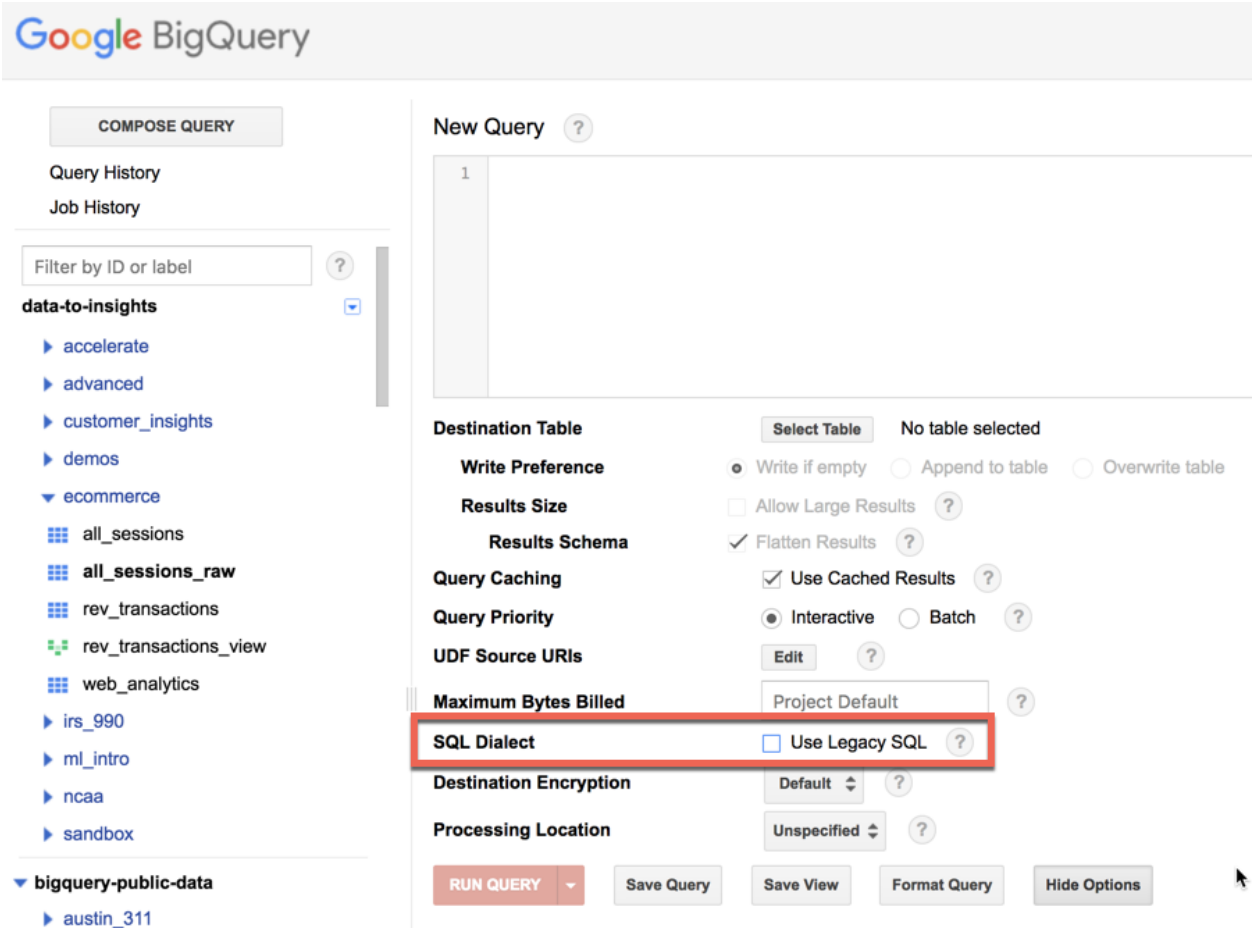
Lastly, ensure Legacy SQL is disabled by clicking:

. Compose Query

. Show Options

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

. Hide Options



Explore Ecommerce Data and Identify Duplicate Records

Your data analyst team has exported the Google Analytics logs for an ecommerce website into BigQuery and created a new table of all the raw ecommerce visitor session data for you to explore.

How large is the `all_sessions_raw` dataset (in rows and disk size)?

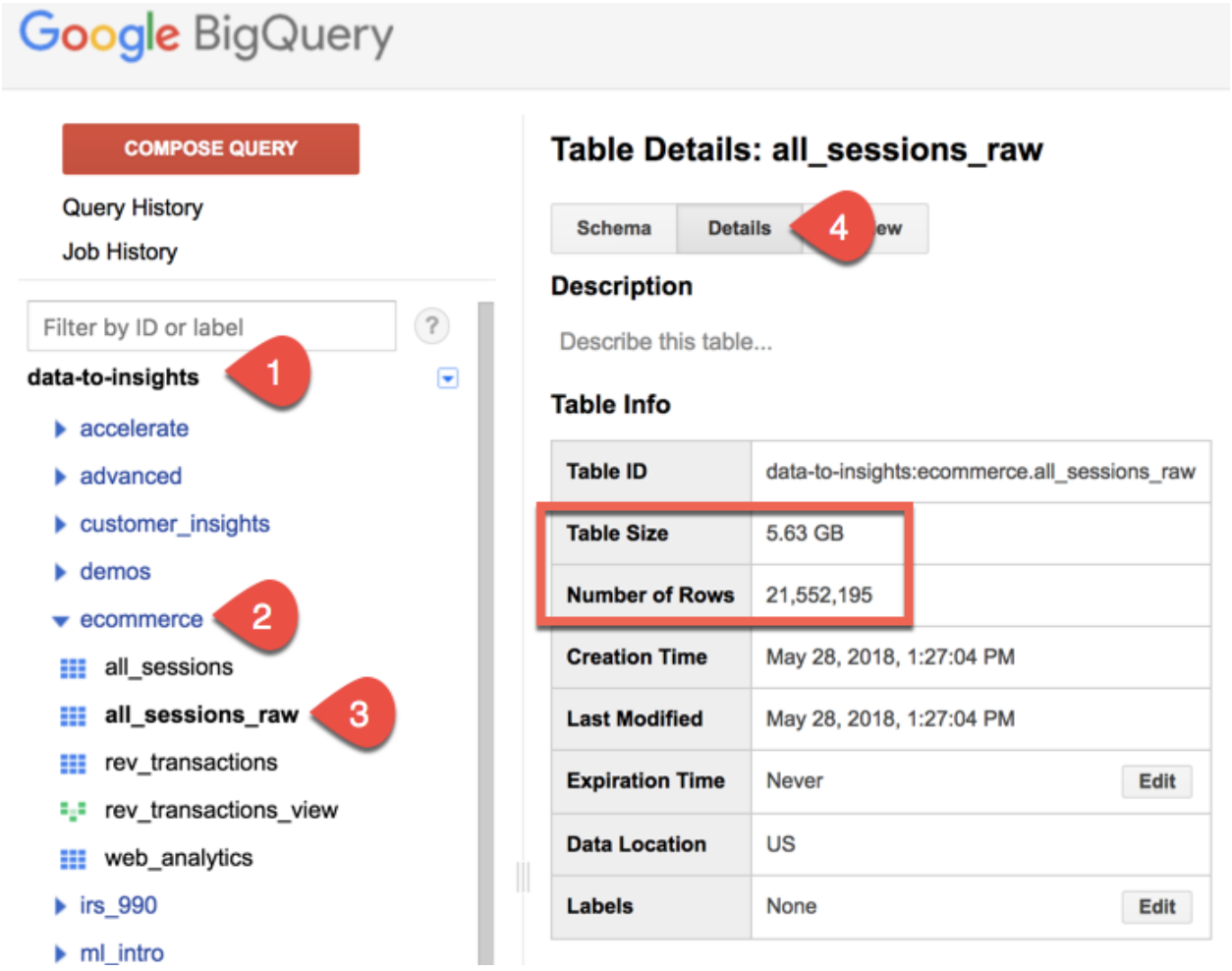
. Expand the data-to-insights project

. Click ecommerce

. Click all_sessions_raw

. Click Details

. View the table metadata



The dataset is over 21 Million rows and is currently 5.63GB in disk size.

What does a unique row represent? Is it unique visitors?

Seeing a sample amount of data will give you greater intuition for what is included (or not) in the dataset. Without any SQL, you can preview a sample rows from the table by clicking preview

Scan and scroll through the rows to see if you can spot a unique identifier for this dataset. Is it fullVisitorId?

Answer: There is no singular field that uniquely identifies a row

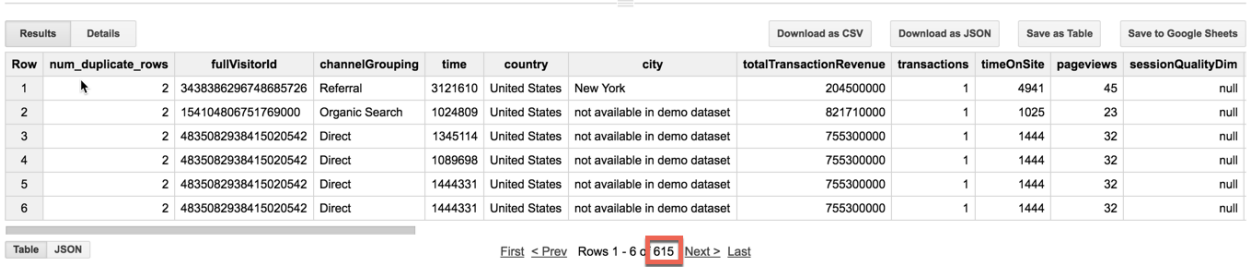
Identify duplicate rows where no key is provided

With your knowledge of SQL, how can you identify and count the occurrences of duplicate records? Hint: Use GROUP BY and HAVING in your query.

Possible Solution:

```
#standardSQL
SELECT COUNT(*) as num_duplicate_rows, *
FROM
`data-to-insights.ecommerce.all_sessions_raw`
GROUP BY
fullVisitorId, channelGrouping, time, country,
city, totalTransactionRevenue, transactions,
timeOnSite, pageviews, sessionQualityDim, date,
visitId, type, productRefundAmount,
productQuantity, productPrice, productRevenue,
productSKU, v2ProductName,
v2ProductCategory, productVariant,
currencyCode, itemQuantity, itemRevenue,
transactionRevenue, transactionId, pageTitle,
searchKeyword, pagePathLevel1,
eCommerceAction_type,
eCommerceAction_step,
eCommerceAction_option
HAVING num_duplicate_rows > 1;
```

Copy and Paste the above query and Execute it to confirm the existence of duplicate records across all columns.



Results | Details | Download as CSV | Download as JSON | Save as Table | Save to Google Sheets

Row	num_duplicate_rows	fullVisitorId	channelGrouping	time	country	city	totalTransactionRevenue	transactions	timeOnSite	pageviews	sessionQualityDim
1	2	3438386296748685726	Referral	3121610	United States	New York	204500000	1	4941	45	null
2	2	154104806751769000	Organic Search	1024809	United States	not available in demo dataset	821710000	1	1025	23	null
3	2	4835082938415020542	Direct	1345114	United States	not available in demo dataset	755300000	1	1444	32	null
4	2	4835082938415020542	Direct	1089698	United States	not available in demo dataset	755300000	1	1444	32	null
5	2	4835082938415020542	Direct	1444331	United States	not available in demo dataset	755300000	1	1444	32	null
6	2	4835082938415020542	Direct	1444331	United States	not available in demo dataset	755300000	1	1444	32	null

Table | JSON

First < Prev Rows 1 - 6 of 615 Next > Last

There are 615 duplicative records in all_sessions_raw

We will be showing how to deduplicate records with a web tool later on in this course. If you want to learn how to deduplicate using advanced SQL, checkout this query here which we will cover in greater detail later in the course. For now, your data analyst team has generated a deduplicated table called all_sessions that we will explore with SQL.

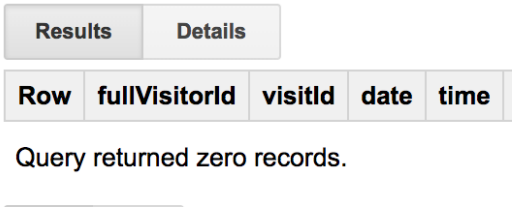
Analyze the new all_sessions table

Your data analyst team has provided you with the below query and your schema experts have identified the key fields that must be unique for each record per your schema. Run the below query to confirm no duplicates exist:

```
#standardSQL
# schema:
https://support.google.com/analytics/answer/3437719
hl=en
SELECT
fullVisitorId, # the unique visitor ID
visitId, # a visitor can have multiple visits
date, # session date stored as string
YYYYMMDD
time, # time of the individual site hit  (can be 0 to
many per visitor session)
v2ProductName, # not unique since a product can
have variants like Color
productSKU, # unique for each product
type, # a visitor can visit Pages and/or can trigger
Events (even at the same time)
eCommerceAction_type, # maps to ‘add to cart',
‘completed checkout'
eCommerceAction_step,
```

```
eCommerceAction_option,
  transactionRevenue, # revenue of the order
  transactionId, # unique identifier for revenue
bearing transaction
COUNT(*) as row_count
FROM
`data-to-insights.ecommerce.all_sessions`
GROUP BY 1,2,3 ,4, 5, 6, 7, 8, 9, 10,11,12
HAVING row_count > 1 # find duplicates
```

Results in 0 records



Now that we understand the important fields and what each row represents, it's time to query for insights on the ecommerce dataset. If you already have experience working with basic SQL, try the harder query questions in the challenge section.

Practice Basic SQL on Ecommerce Data

Write a query that shows total unique visitors

. Click Compose Query

. Write your query in the editor

. Use the real-time query validator icon to ensure your syntax is correct

. Click Run Query

. View the results

Possible Solution:

```
#standardSQL
SELECT
  COUNT(*) AS product_views,
  COUNT(DISTINCT fullVisitorId) AS
unique_visitors
FROM `data-to-insights.ecommerce.all_sessions`;
```

Row	product_views	unique_visitors
1	21493109	389934

Write a query that shows total unique visitors by channel grouping (organic, referring site)

Possible Solution:

```
#standardSQL
SELECT
  COUNT(DISTINCT fullVisitorId) AS
unique_visitors,
  channelGrouping
FROM `data-to-insights.ecommerce.all_sessions`
GROUP BY 2
ORDER BY 2 DESC;
```

Row	unique_visitors	channelGrouping
1	38101	Social
2	57308	Referral
3	11865	Paid Search
4	211993	Organic Search
5	3067	Display
6	75688	Direct
7	5966	Affiliates
8	62	(Other)

What are all the unique product names listed alphabetically?

Possible Solution:

```
#standardSQL
SELECT
  v2ProductName
FROM `data-to-insights.ecommerce.all_sessions`
GROUP BY 1
ORDER BY 1;
```

Row	v2ProductName
1	1 oz Hand Sanitizer
2	14oz Ceramic Google Mug
3	15 oz Ceramic Mug
4	15" Android Squishable - Online
5	16 oz. Hot and Cold Tumbler
6	16 oz. Hot/Cold Tumbler
7	20 oz Stainless Steel Insulated Tumbler
8	22 oz Android Bottle

633 Products (rows) total

Which 5 products had the most views from unique visitors viewed each product?

Possible Solution:

```
#standardSQL
SELECT
  COUNT(*) AS product_views,
  v2ProductName
FROM `data-to-insights.ecommerce.all_sessions`
WHERE type = 'PAGE'
GROUP BY v2ProductName
ORDER BY product_views DESC
LIMIT 5;
```

Row	product_views	v2ProductName
1	316482	Google Men's 100% Cotton Short Sleeve Hero Tee White
2	221558	22 oz YouTube Bottle Infuser
3	210700	YouTube Men's Short Sleeve Hero Tee Black
4	202205	Google Men's 100% Cotton Short Sleeve Hero Tee Black
5	200789	YouTube Custom Decals

Expand your previous query to include the total number of distinct products ordered as well as the total number of total units ordered

Possible Solution:

```
#standardSQL
SELECT
  COUNT(*) AS product_views,
  COUNT(productQuantity) AS orders,
  SUM(productQuantity) AS
quantity_product_ordered,
  v2ProductName
FROM `data-to-insights.ecommerce.all_sessions`
WHERE type = 'PAGE'
GROUP BY v2ProductName
ORDER BY product_views DESC
LIMIT 5;
```

Row	product_views	orders	quantity_product_ordered	v2ProductName
1	316482	3158	6352	Google Men's 100% Cotton Short Sleeve Hero Tee White
2	221558	508	4769	22 oz YouTube Bottle Infuser
3	210700	949	1114	YouTube Men's Short Sleeve Hero Tee Black
4	202205	2713	8072	Google Men's 100% Cotton Short Sleeve Hero Tee Black
5	200789	1703	11336	YouTube Custom Decals

Questions:

- Did the product with the most views get the most orders?
- What is the difference between orders and quantity_product_ordered?

Expand the query to include the ratio of product units to order:

Possible Solution:

```
#standardSQL
SELECT
  COUNT(*) AS product_views,
  COUNT(productQuantity) AS orders,
  SUM(productQuantity) AS
quantity_product_ordered,
  SUM(productQuantity) /
COUNT(productQuantity) AS avg_per_order,
  v2ProductName
FROM `data-to-insights.ecommerce.all_sessions`
WHERE type = 'PAGE'
GROUP BY v2ProductName
ORDER BY product_views DESC
LIMIT 5;
```

Row	product_views	orders	quantity_product_ordered	avg_per_order	v2ProductName
1	316482	3158	6352	2.011399620012666	Google Men's 100% Cotton Short Sleeve Hero Tee White
2	221558	508	4769	9.387795275590552	22 oz YouTube Bottle Infuser
3	210700	949	1114	1.1738672286617493	YouTube Men's Short Sleeve Hero Tee Black
4	202205	2713	8072	2.9753040914117213	Google Men's 100% Cotton Short Sleeve Hero Tee Black
5	200789	1703	11336	6.656488549618321	YouTube Custom Decals

The 22 oz YouTube Bottle Infuser had the highest avg_per_order with 9.38 units per order

Challenge Exercises

Ready to put your SQL skills to the test? Try these challenge questions for extra credit:

Challenge 1: Calculating Conversion Rate

For products with over 1000 units that have been added to cart or ordered that are not frisbees:

- **How many distinct times was the product part of an order (either complete or incomplete order)?**
- **How many total units of the product were part of orders (either complete or incomplete)?**
- **Which product had the highest conversion rate?**

Complete the following partial query:

```
#standardSQL
SELECT
  COUNT(*) AS product_views,
  COUNT(productQuantity) AS potential_orders,
  SUM(productQuantity) AS
quantity_product_added,
  v2ProductName
FROM `data-to-insights.ecommerce.all_sessions`
WHERE v2ProductName NOT LIKE 'frisbee'
GROUP BY v2ProductName
HAVING quantity_product_added >
ORDER BY conversion_rate
LIMIT 10;
```

Possible solution:

```
#standardSQL
SELECT
  COUNT(*) AS product_views,
  COUNT(productQuantity) AS potential_orders,
  SUM(productQuantity) AS
quantity_product_added,
  (COUNT(productQuantity) / COUNT(*)) AS
conversion_rate,
  v2ProductName
FROM `data-to-insights.ecommerce.all_sessions`
```

```
WHERE LOWER(v2ProductName) NOT LIKE
'%frisbee%'
GROUP BY v2ProductName
HAVING quantity_product_added > 1000
ORDER BY conversion_rate DESC
LIMIT 10;
```

Row	product_views	potential_orders	quantity_product_added	conversion_rate	v2ProductName
1	683	240	1428	0.3513909224011713	Google 25 oz Clear Stainless Steel Bottle
2	3667	1282	1622	0.34960458140169076	Google Men's Bike Short Sleeve Tee Charcoal
3	629	194	1101	0.30842607313195547	Android Men's Paradise Short Sleeve Tee Olive
4	6897	1524	1856	0.22096563723357981	BLM Sweatshirt
5	147729	22993	140734	0.15564310324986969	Nest® Learning Thermostat 3rd Gen-USA - Stainless Steel
6	1574	207	1284	0.1315120711562897	Google Leather Journal-Black

Challenge 2: Track Visitor Checkout Progress

Write a query that shows the eCommerceAction_type and the distinct count of fullVisitorId associated with each type.

Possible Solution:

```
#standardSQL
SELECT
  COUNT(DISTINCT fullVisitorId) AS
number_of_unique_visitors,
  eCommerceAction_type
FROM `data-to-insights.ecommerce.all_sessions`
GROUP BY eCommerceAction_type
ORDER BY eCommerceAction_type;
```

Row	number_of_unique_visitors	eCommerceAction_type
1	389240	0
2	122728	1
3	122477	2
4	56010	3
5	12015	4
6	30408	5
7	19988	6

Bonus:

You are given the below mapping for the action type.

- Unknown = 0
- Click through of product lists = 1
- Product detail views = 2
- Add product(s) to cart = 3
- Remove product(s) from cart = 4
- Check out = 5
- Completed purchase = 6
- Refund of purchase = 7
- Checkout options = 8

Use a Case Statement to add a new column to your previous query to display the eCommerceAction_type label (e.g. "Completed purchase")

Possible Solution:

```
#standardSQL
SELECT
  COUNT(DISTINCT fullVisitorId) AS
number_of_unique_visitors,
  eCommerceAction_type,
  CASE eCommerceAction_type
  WHEN '0' THEN 'Unknown'
  WHEN '1' THEN 'Click through of product lists'
  WHEN '2' THEN 'Product detail views'
```

```
  WHEN '3' THEN 'Add product(s) to cart'
  WHEN '4' THEN 'Remove product(s) from cart'
  WHEN '5' THEN 'Check out'
  WHEN '6' THEN 'Completed purchase'
  WHEN '7' THEN 'Refund of purchase'
  WHEN '8' THEN 'Checkout options'
  ELSE 'ERROR'
  END AS eCommerceAction_type_label
FROM `data-to-insights.ecommerce.all_sessions`
GROUP BY eCommerceAction_type
ORDER BY eCommerceAction_type;
```

Row	number_of_unique_visitors	eCommerceAction_type	eCommerceAction_type_label
1	389240	0	Unknown
2	122728	1	Click through of product lists
3	122477	2	Product detail views
4	56010	3	Add product(s) to cart
5	12015	4	Remove product(s) from cart
6	30408	5	Check out
7	19988	6	Completed purchase

What percent of visitors who added to cart completed a purchase?

Answer: 19988 / 56010 = .3568 or 35.68%

Challenge 3: Track Abandoned Carts from High Quality Sessions

Write a query using aggregation functions that returns the unique session ids of those visitors who have added a product to their cart but never completed checkout (abandoned their shopping cart).

Possible Solution:

```
#standardSQL
# high quality abandoned carts
SELECT
  #unique_session_id
```

```
  CONCAT(fullVisitorId,CAST(visitId AS
STRING)) AS unique_session_id,
  sessionQualityDim,
  SUM(productRevenue) AS transaction_revenue,
  MAX(eCommerceAction_type) AS
checkout_progress
FROM `data-to-insights.ecommerce.all_sessions`
WHERE sessionQualityDim > 60 # high quality
session
GROUP BY unique_session_id,
sessionQualityDim
HAVING
  checkout_progress = '3' # 3 = added to cart
  AND (transaction_revenue = 0 OR
transaction_revenue IS NULL)
```

Resources:

- Solutions to regular exercises
- Solutions to challenge exercises

Congratulations!

You've successfully explored the data-to-insights ecommerce dataset! 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.