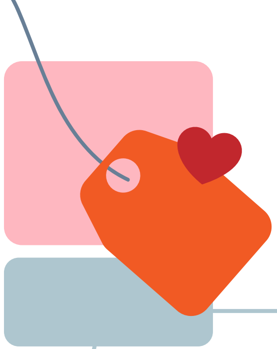


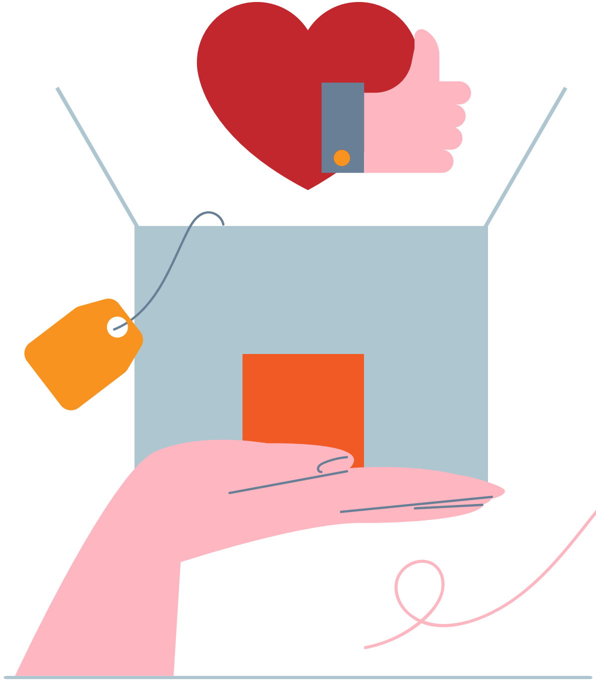
Exploring an Ecommerce Dataset using SQL in Google BigQuery

Ismi Ana Sulasiyah
(annaismi17@gmail.com)

BigQuery for Data Analysts
Google Cloud

Overview

- BigQuery is Google's fully managed, NoOps, low cost analytics database. With BigQuery we 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 us to focus on analyzing data to find meaningful insights.



Project Tasks

01 Access an ecommerce dataset

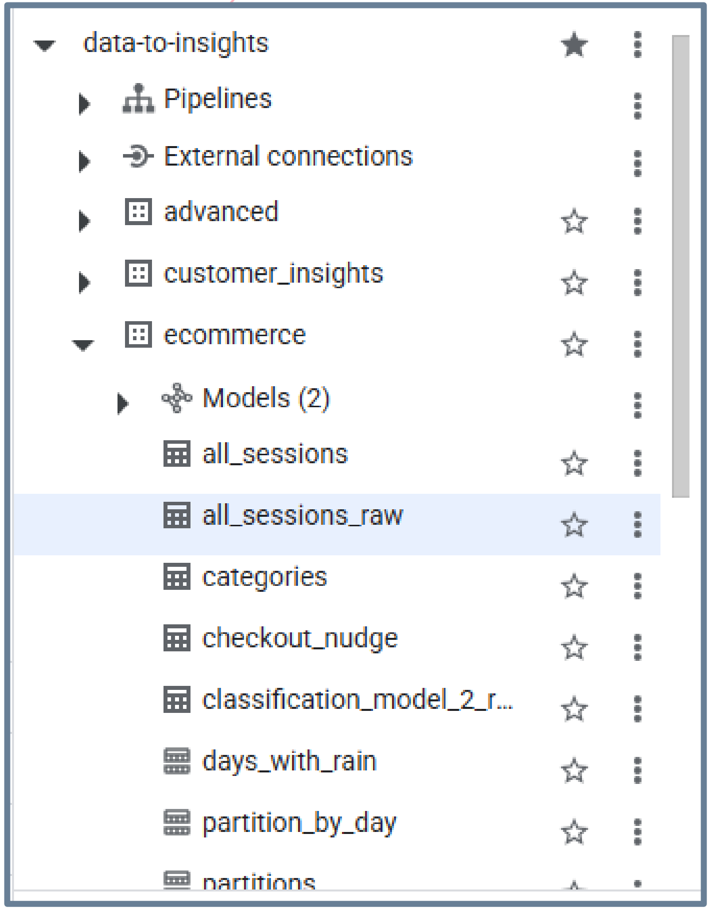
02 Explore ecommerce data and identify duplicate records

03 Write basic SQL on ecommerce data

04 Practice with SQL

Task 1. Access Ecommerce Dataset

In this lab, I use a copy of Ecommerce dataset that BigQuery already provided. With the project name: data-to-insights



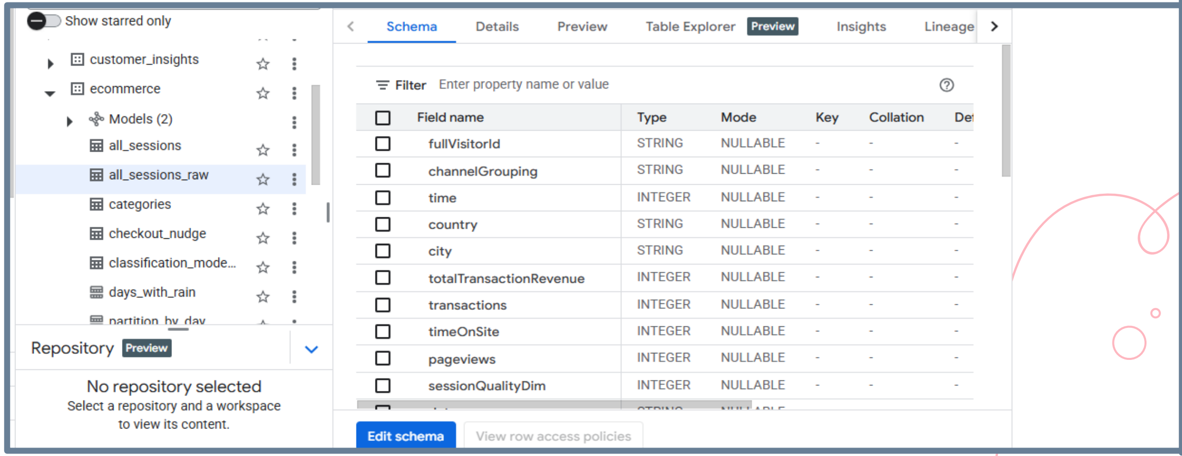
data-to-insights
- Pipelines
- External connections
- advanced
- customer_insights
- ecommerce
  - Models (2)
  - all_sessions
  - all_sessions_raw
  - categories
  - checkout_nudge
  - classification_model_2_r...
  - days_with_rain
  - partition_by_day
  - partitions

Task 2. Explore Ecommerce Data and Identify Duplicate Records

Scenario:

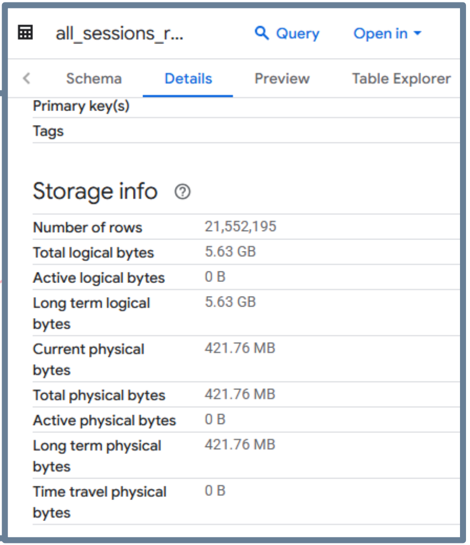
The data analyst team exported the Google Analytics logs for an ecommerce website into BigQuery and created a new table of all the raw ecommerce visitor session data. A section opens that provides 3 views of the table data:

- Schema tab: Field name, Type, Mode, and Description; the logical constraints used to organize the data
- Details tab: Table metadata
- Preview tab: Table preview



No repository selected
Select a repository and a workspace to view its content.

□	Field name	Type	Mode	Key	Collation	De
□	fullVisitorId	STRING	NULLABLE	-	-	-
□	channelGrouping	STRING	NULLABLE	-	-	-
□	time	INTEGER	NULLABLE	-	-	-
□	country	STRING	NULLABLE	-	-	-
□	city	STRING	NULLABLE	-	-	-
□	totalTransactionRevenue	INTEGER	NULLABLE	-	-	-
□	transactions	INTEGER	NULLABLE	-	-	-
□	timeOnSite	INTEGER	NULLABLE	-	-	-
□	pageviews	INTEGER	NULLABLE	-	-	-
□	sessionQualityDim	INTEGER	NULLABLE	-	-	-



Primary key(s)

Tags

Storage info

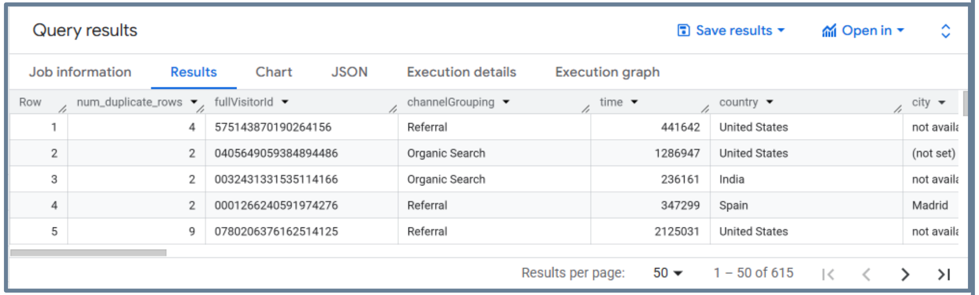
Number of rows	21,552,195
Total logical bytes	5.63 GB
Active logical bytes	0 B
Long term logical bytes	5.63 GB
Current physical bytes	421.76 MB
Total physical bytes	421.76 MB
Active physical bytes	0 B
Long term physical bytes	421.76 MB
Time travel physical bytes	0 B

Task 2. Explore Ecommerce Data and Identify Duplicate Records

Identify duplicate rows

There is no singular field that uniquely identifies a row, so I need advanced logic to identify duplicate rows. My query uses the SQL GROUP BY function on every field and counts (COUNT) where there are rows that have the same values across every field.

- If every field is unique, the COUNT will return 1 as there are no other groupings of rows with the exact same value for all fields.
- If there is a row with the same values for all fields, they will be grouped together and the COUNT will be greater than 1. The last part of the query is an aggregation filter using HAVING to only show the results that have a COUNT of duplicates greater than 1.



Row	num_duplicate_rows	fullVisitorId	channelGrouping	time	country	city
1	4	575143870190264156	Referral	441642	United States	not availa
2	2	0405649059384894486	Organic Search	1286947	United States	(not set)
3	2	0032431331535114166	Organic Search	236161	India	not availa
4	2	0001266240591974276	Referral	347299	Spain	Madrid
5	9	0780206376162514125	Referral	2125031	United States	not availa

Results per page: 50 1 – 50 of 615

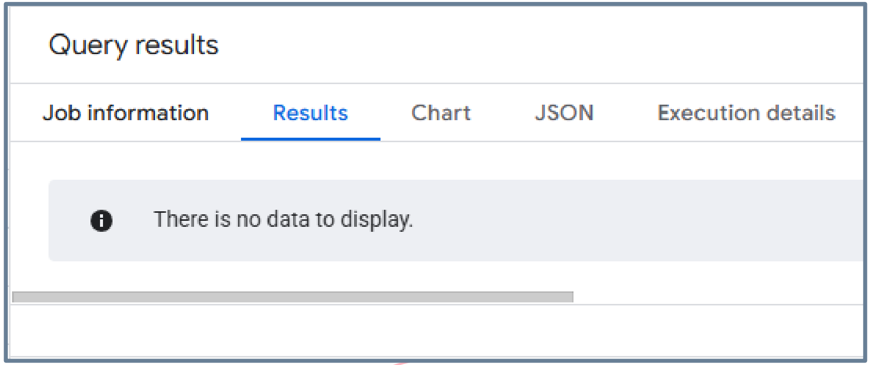
```
SELECT COUNT(*) as num_duplicate_rows, *
FROM
`data-to-
insights.ecommerce.all_sessions_raw`
GROUP BY
fullVisitorId, channelGrouping, time,
country, city, totalTransactionRevenue,
transactions, timeOnSite, pageviews,
sessionQualityDim, date, visitId, type,
productRefundAmount, productQuantity,
productPrice,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

Task 2. Explore Ecommerce Data and Identify Duplicate Records

Analyze the new all_sessions table

In this section you use a deduplicated table called all_sessions.

```
SELECT fullVisitorId,visitId, date, time,
v2ProductName, productSKU, type,
eCommerceAction_type, eCommerceAction_step,
eCommerceAction_option, transactionRevenue,
transactionId, 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;
```



There is no data to display.

Run the query to confirm that no duplicates exist, this time in the all_sessions table:
‘The query returns zero records.’

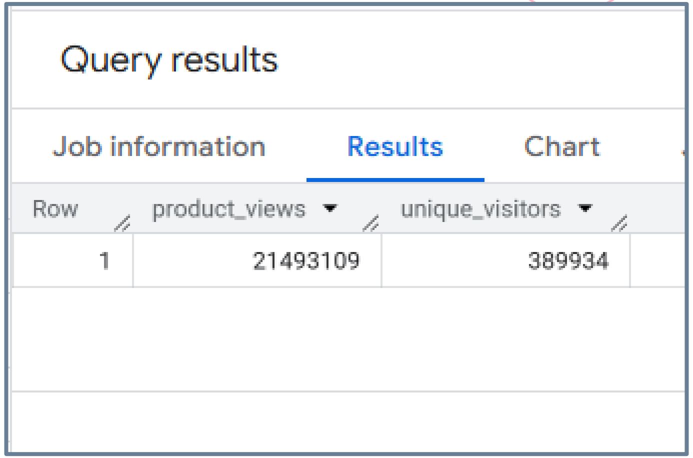
Note: In SQL, you can GROUP BY or ORDER BY the index of the column like using "GROUP BY 1" instead of "GROUP BY fullVisitorId".

Task 3. Write Basic SQL on Ecommerce Data

A query that shows total unique visitors

My query determines the total views by counting product_views and the number of unique visitors by counting fullVisitorID.

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



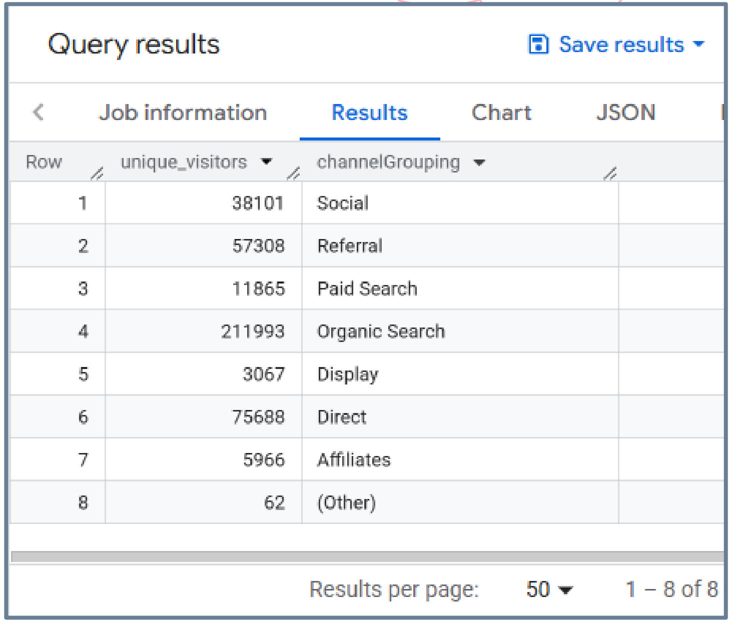
Row	product_views	unique_visitors
1	21493109	389934

Task 3. Write Basic SQL on Ecommerce Data

A query that shows total unique visitors

My query that shows total unique visitors(fullVisitorID) by the referring site (channelGrouping):

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

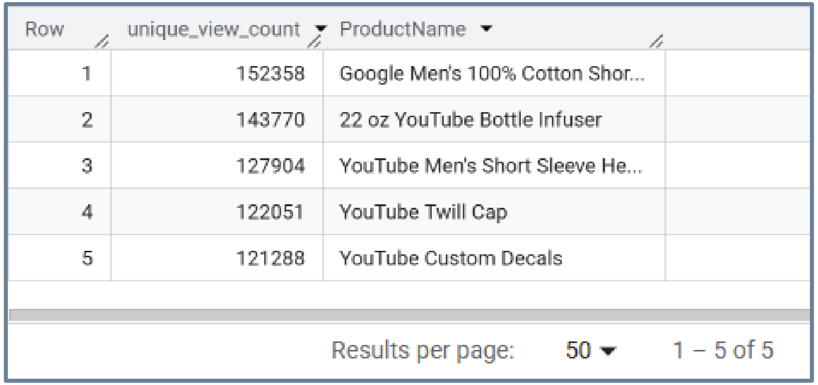


Row	unique_visitors	channelGrouping
1	38101	Social
2	57308	Referral
3	11865	Paid Search
4	211993	Organic Search
5	3067	Display
6	75688	Direct
7	5966	Affiliates
8	62	(Other)

Task 3. Write Basic SQL on Ecommerce Data

Unique view count per-user

Refine the query to no longer double-count product views for visitors who have viewed a product many times. Each distinct product view should only count once per visitor.



Row	unique_view_count	ProductName
1	152358	Google Men's 100% Cotton Shor...
2	143770	22 oz YouTube Bottle Infuser
3	127904	YouTube Men's Short Sleeve He...
4	122051	YouTube Twill Cap
5	121288	YouTube Custom Decals

SQL WITH clause to help break apart a complex query into multiple steps. Here I first create a query that finds each unique product per visitor and counts them once. Then the second query performs the aggregation across all visitors and products.

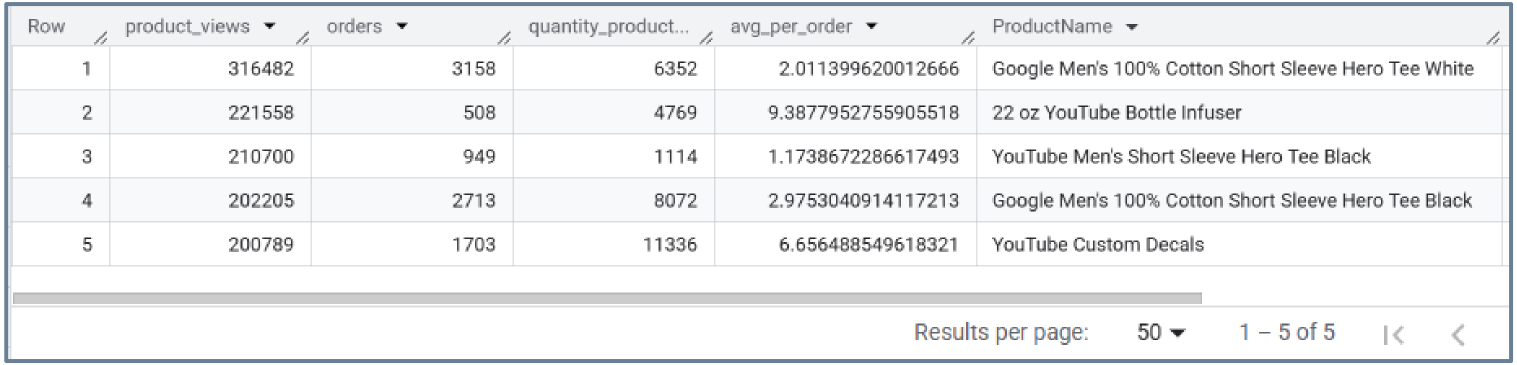
```
WITH unique_product_views_by_person AS (
-- find each unique product viewed by each
    visitor
SELECT
 fullVisitorId,
 (v2ProductName) AS ProductName
FROM `data-to-insights.ecommerce.all_sessions`
WHERE type = 'PAGE'
GROUP BY fullVisitorId, v2ProductName )

    -- aggregate the top viewed products and
    sort them
SELECT
  COUNT(*) AS unique_view_count,
  ProductName
FROM unique_product_views_by_person
GROUP BY ProductName
ORDER BY unique_view_count DESC
LIMIT 5;
```

Task 3. Write Basic SQL on Ecommerce Data

Standard Query with avg_per_order

Expand the query to include the average amount of product per order (total number of units ordered/total number of orders, or SUM(productQuantity)/COUNT(productQuantity)):

```
SELECT
  COUNT(*) AS product_views, COUNT(productQuantity) AS orders, SUM(productQuantity) AS
    quantity_product_ordered, SUM(productQuantity) / COUNT(productQuantity) AS
    avg_per_order, (v2ProductName) AS ProductName
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...	avg_per_order	ProductName
1	316482	3158	6352	2.011399620012666	Google Men's 100% Cotton Short Sleeve Hero Tee White
2	221558	508	4769	9.3877952755905518	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

Task 4. Practice with SQL

Challenge 1: Calculate a conversion rate

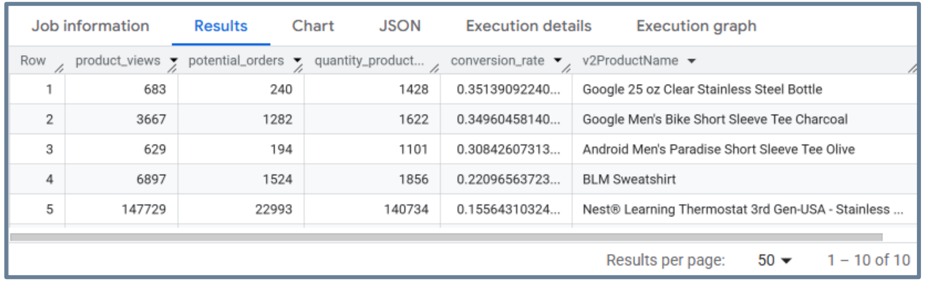
Write a conversion rate query for products with these qualities:

- More than 1000 units were added to a cart or ordered
- AND are not frisbees

Following questions:

- How many distinct times was the product part of an order (either complete or incomplete order)? 10 Times
- Which product had the highest conversion rate? Google 25 oz Clear Stainless Steel Bottle

```sql
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;
```



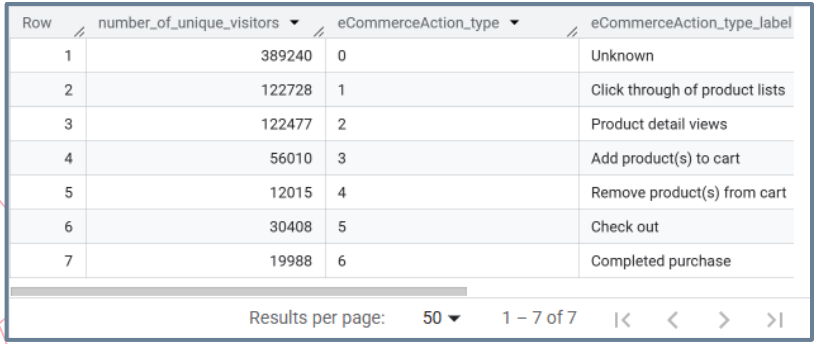
Row	product_views	potential_orders	quantity_product...	conversion_rate	v2ProductName
1	683	240	1428	0.35139092240...	Google 25 oz Clear Stainless Steel Bottle
2	3667	1282	1622	0.34960458140...	Google Men's Bike Short Sleeve Tee Charcoal
3	629	194	1101	0.30842607313...	Android Men's Paradise Short Sleeve Tee Olive
4	6897	1524	1856	0.22096563723...	BLM Sweatshirt
5	147729	22993	140734	0.15564310324...	Nest® Learning Thermostat 3rd Gen-USA - Stainless ...

Results per page: 50 1 – 10 of 10

Task 4. Practice with SQL

Challenge 2: Track visitor checkout progress

- Write a query that shows the eCommerceAction_type and the distinct count of fullVisitorId associated with each type.
- Use a Case Statement to add a new column to your previous query to display the eCommerceAction_type label (such as “Completed purchase”).



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

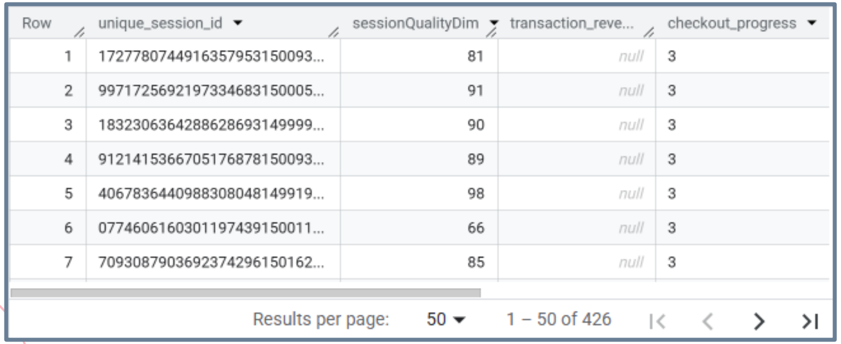
Results per page: 50 1 – 7 of 7

```
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;
```

Task 4. Practice with SQL

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).

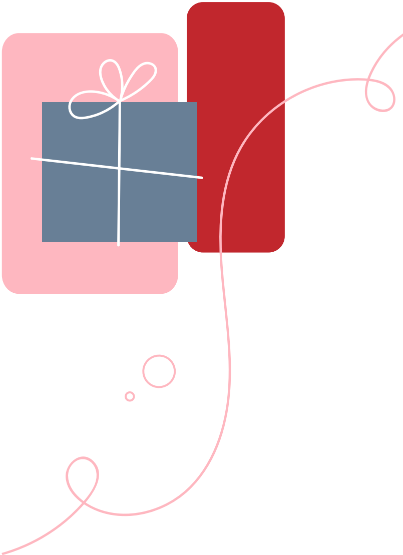


Row	unique_session_id	sessionQualityDim	transaction_reve...	checkout_progress
1	1727780744916357953150093...	81	null	3
2	9971725692197334683150005...	91	null	3
3	1832306364288628693149999...	90	null	3
4	9121415366705176878150093...	89	null	3
5	4067836440988308048149919...	98	null	3
6	0774606160301197439150011...	66	null	3
7	7093087903692374296150162...	85	null	3

Results per page: 50 ▾ 1 – 50 of 426

```
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);
```

Thanks!



For follow-up author:
annaismi17@gmail.com
id.linkedin.com/in/ismiana

Credit by Google Cloud
BigQuery for Data Analysts Course