

# Explore ecommerce data and identify duplicate records

**Scenario:** Your 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.

Explore the `all_sessions_raw` table data:

1. Click the **Expand node** icon near **data-to-insights** to expand the project.
2. Expand **ecommerce**.
3. Click **all\_sessions\_raw**.

In the right pane, 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

4. Click the **Details** tab to view the table metadata.

FEATURES & INFO

SHORTCUT

HIDE PREVIEW FEATURES

Explorer

+ ADD DATA

Type to search

?

Viewing pinned projects.

qwiklabs-gcp-00-e4b0f55a9e21

data-to-insights

AJ\_Retail

AJ\_Retail\_Partitioned

advanced

bitcoin\_blockchain

customer\_insights

ecommerce

all\_sessions

all\_sessions\_raw

categories

checkout\_nudge

EDITOR

ALL\_SES...

all\_sessions\_raw

QUERY TABLE

COPY TABLE

Schema

Details

Preview

Description

None

Labels

None

Table info

Table ID	data-to-insights:ecommerce.all_sessions_raw
Table size	5.63 GB
Long-term storage size	5.63 GB
Number of rows	21,552,195
Created	May 29, 2018, 1:57:04 AM
Table expiration	Never
Last modified	Jun 6, 2018, 1:27:03 AM
Data location	US

Questions:

~~Preview~~

## Schema

Over 21 million

## Identify duplicate rows

Seeing a sample amount of data may give you greater intuition for what is included in the dataset. To preview sample rows from the table without using SQL, click the **preview** tab.


Scan and scroll through the rows. There is no singular field that uniquely identifies a row, so you need advanced logic to identify duplicate rows.

The query you'll use (below) 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 returns 1 as there are no other groupings of rows with the exact same value for all fields.
  - If there are multiple rows with the same values for all fields, these rows are 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. Therefore, the number of records that have duplicates will be the same as the number of rows in the resulting table.

Copy and paste the following query into the query **EDITOR**, then **RUN** query to find which records are duplicated across all columns.

```
#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;
content_copy
615
```

 Check my progress to verify the objective.

## Analyze the new `all_sessions` table

In this section you use a deduplicated table called `all_sessions`.

**Scenario:** Your data analyst team has provided you with this query, and your schema experts have identified the key fields that must be unique for each record per your [schema](#).

Run the query to confirm that no duplicates exist, this time in the `all_sessions` table:

```
#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
content_copy
```

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"

# Write basic SQL on ecommerce data

In this section, you query for insights on the ecommerce dataset.

## Write a query that shows total unique visitors

Your query determines the total views by counting product\_views and the number of unique visitors by counting fullVisitorID.

1. Click + **Compose New Query**.

2. Write this query in the editor:

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

3. To ensure that your syntax is correct, click the real-time query validator icon.

4. Click **Run**. Read the results to view the number of unique visitors.

### Results

Row	product_views	unique_visitors
1	21493109	389934

Now write a query that shows total unique visitors(fullVisitorID) by the referring site (channelGrouping):

```
#standardSQL
SELECT
  COUNT(DISTINCT fullVisitorId) AS unique_visitors,
```

```

channelGrouping
FROM `data-to-insights.ecommerce.all_sessions`
GROUP BY channelGrouping
ORDER BY channelGrouping DESC;
content_copy

```

## Results

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)

Write a query to list all the unique product names (v2ProductName) alphabetically:

```

#standardSQL
SELECT
  (v2ProductName) AS ProductName
FROM `data-to-insights.ecommerce.all_sessions`
GROUP BY ProductName
ORDER BY ProductName
content_copy

```

Tip: In SQL, the ORDER BY clause defaults to Ascending (ASC) A-->Z. If you want the reverse, try ORDER BY field\_name DESC

## GROUP BY Results

Query complete (1.422 sec elapsed, 702.56 MB processed)

Job information **Results** JSON Execution details

Row	ProductName
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
9	22 oz Mini Mountain Bottle
10	22 oz YouTube Bottle Infuser
11	22 oz. Android Mini Mountain Bottle

This query returns a total of 633 products (rows).

633

Write a query to list the five products with the most views (product\_views) from visitors (include people who have viewed the same product more than once). Your query counts number of times a product (v2ProductName) was viewed (product\_views), puts the list in descending order, and lists the top 5 entries:

Tip: In Google Analytics, a visitor can "view" a product during the following interaction types: 'page', 'screenview', 'event', 'transaction', 'item', 'social', 'exception', 'timing'. For our purposes, simply filter for only type = 'PAGE'.

```
#standardSQL
SELECT
  COUNT(*) AS product_views,
  (v2ProductName) AS ProductName
```

```
FROM `data-to-insights.ecommerce.all_sessions`
WHERE type = 'PAGE'
GROUP BY v2ProductName
ORDER BY product_views DESC
LIMIT 5;
content_copy
```

## Results

Query complete (1.554 sec elapsed, 826.31 MB processed)

Job information [Results](#) JSON Execution details

Row	product_views	ProductName
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

Bonus: Now 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.

```
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
content_copy
```

Tip: You can use the SQL WITH clause to help break apart a complex query into multiple steps. Here we 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.

## Results

Query complete (6.0 sec elapsed, 1.2 GB processed)

[Job information](#)   [Results](#)   [JSON](#)   [Execution details](#)

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

Next, expand your previous query to include the total number of distinct products ordered and the total number of total units ordered (productQuantity):

```
#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;
content_copy
```

## Results

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:

True

☐ order is the number of orders, quantity\_product\_ordered is the number of items available to be ordered

☐ order is the number of orders, quantity\_product\_ordered is the number of items ordered

Expand the query to include the average amount of product per order (total number of units ordered/total number of orders, or  $\text{SUM}(\text{productQuantity})/\text{COUNT}(\text{productQuantity})$ ).

```
#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) AS ProductName
FROM `data-to-insights.ecommerce.all_sessions`
WHERE type = 'PAGE'
GROUP BY v2ProductName
```

```
ORDER BY product_views DESC
```

```
LIMIT 5;
```

```
content_copy
```

## Results

Row	product_views	orders	quantity_product_ordered	avg_per_order	ProductName
1	316482	3158	6352	2.011399620012666	Google Men's 100% Cotton Short Sleeve Hero Tee White
2	221558	508	4769	9.387795275590552	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

Question:

YouTube Bottle Infuser

Google Mens Short Sleeve Hero Tee Black

The 22 oz YouTube Bottle Infuser had the highest avg\_per\_order with 9.38 units per order.

Click [Check my progress](#) to verify the objective.

# Congratulations!

This concludes exploring the data-to-insights ecommerce dataset! You used BigQuery to view and query the data to gain meaningful insight on various aspects of product marketing.



## Finish your Quest

This self-paced lab is part of the Qwiklabs [BigQuery for Marketing Analysts](#) and [BigQuery Basics for Data Analysts](#) Quests. 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 a Quest and get immediate completion credit if you've taken this lab. See other available [See other available Qwiklabs Quests](#).

## Take your next lab

Continue your Quest with the next lab, [Troubleshooting Common SQL Errors with BigQuery](#).

Check out other labs to learn more about BigQuery:

- [Weather Data in BigQuery](#)
- [Optimize Performance and Cost Using BigQuery Partitions](#)

## Next steps/learn more

- [Troubleshooting Common SQL Errors with BigQuery](#)
- Explore [BigQuery Public Datasets](#).
- Already 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](#).

## 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: May 5, 2021

Lab Last Tested: May 5, 2021

Copyright 2021 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.