

## COMP8210 Assignment 3

### Part 1. Basic analytics using BigQuery

P1. Count the number of rows in the *all\_sessions\_raw* table

SQL code:

```
SELECT COUNT(*) AS `number_of_rows`  
FROM `data-to-insights.ecommerce.all_sessions_raw`;
```

Result:

Row	number_of_rows
1	21552195

P2. Count the number of rows in the *all\_sessions* table

SQL code:

```
SELECT COUNT(*) AS `number_of_rows`  
FROM `data-to-insights.ecommerce.all_sessions`;
```

Result:

Row	number_of_rows
1	21493109

P3. Check the min and max date of the records in the *all\_sessions*

SQL code:

```
SELECT  
  MIN(date) AS `min_date`,  
  MAX(date) AS `max_date`  
FROM `data-to-insights.ecommerce.all_sessions`;
```

Result:

Row	min_date	max_date
1	20160801	20170801

P4. Write a query that shows total unique visitors.

SQL code:

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

Result:

Row	product_views	unique_visitors
1	21493109	389934

P5. Write a query that shows total unique visitors (*fullVisitorID*) by the referring site (*channelGrouping*)

SQL code:

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

Result:

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

P6. Write a query to list all the unique product names (*v2ProductName*) alphabetically.

SQL code:

```
SELECT  
  (v2ProductName) AS unique_product_name  
FROM `data-to-insights.ecommerce.all_sessions`  
GROUP BY unique_product_name  
ORDER BY unique_product_name;
```

Result (preview of the first 8 results):

JOB INFORMATION		RESULTS
Row	unique_product_name	
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 ...	
8	22 oz Android Bottle	

Results per page: 50 ▼ 1 – 50 of 633

## Part 2. Duplicate detection in data

P7. Write a query that returns the number of duplicate records in the “**all\_sessions\_raw**” table.

SQL code:

```
SELECT  
COUNT(*) as num_duplicate_records, *  
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_records > 1;
```

Result (preview of the first 8 results):

Row	num_duplicate_records	fullVisitorId	channelGrouping	time	country
1	2	7845114848613059919	Referral	791561	United States
2	2	7567817112637547239	Organic Search	2303834	United States
3	2	603646718962990199	Referral	106125	Netherlands
4	2	0760185147017673561	Referral	870765	United States
5	2	6379920233928997690	Organic Search	1550040	United States
6	2	9247530999213434637	Organic Search	327792	United Arab Emirates
7	2	4835082938415020542	Direct	1444297	United States
8	2	2073001123961505601	Social	522680	Egypt

Results per page: 50 1 – 50 of 615

P8. Write a query that ensures there are no duplicate records in the “all\_sessions” table.

SQL code:

```
SELECT
COUNT(*) as num_records, *
FROM
`data-to-insights.ecommerce.all_sessions`
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_records > 1;
```

Result (no duplicate records, thus no data to display):

JOB INFORMATION	RESULTS	JSON	EXECUTION DETAILS
<div>  There is no data to display. </div>			

Results per page: 50 1 – 0 of many

### Part 3. Advanced analytics using BigQuery

P9. Write a query to list the five products with the most views (*product\_views*) from all visitors (including people who have viewed the same product more than once). Your query counts the number of times a product (*v2ProductName*) was viewed (*product\_views*), puts the list in descending order, and lists the top 5 entries.

SQL code:

```
SELECT
  (v2ProductName) AS product_name,
  COUNT(*) AS product_views
FROM `data-to-insights.ecommerce.all_sessions`
GROUP BY product_name
ORDER BY product_views DESC
LIMIT 5;
```

Result:

Row	product_name	product_views
1	Google Men's 100% Cotton Short Sleeve Hero Tee White	338436
2	22 oz YouTube Bottle Infuser	226471
3	YouTube Men's Short Sleeve Hero Tee Black	217789
4	YouTube Custom Decals	211423
5	Google Men's 100% Cotton Short Sleeve Hero Tee Black	210861

P10. 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.

SQL code:

```
# first find the number of views of each unique product viewed by each visitor
WITH unique_views_by_person AS (
  SELECT
    fullVisitorId,
    (v2ProductName) AS product_name
  FROM `data-to-insights.ecommerce.all_sessions`
  GROUP BY fullVisitorId, product_name )
# then aggregate the number of unique views and sort in descending order
SELECT
  product_name,
  COUNT(*) AS unique_view
FROM unique_views_by_person
GROUP BY product_name
ORDER BY unique_view DESC
LIMIT 5
```

Result:

Row	product_name	unique_view
1	Google Men's 100% Cotton Short Sleeve Hero Tee White	152372
2	22 oz YouTube Bottle Infuser	143784
3	YouTube Men's Short Sleeve Hero Tee Black	127996
4	YouTube Twill Cap	122081
5	YouTube Custom Decals	121352

P11. expand your previous query to include the total number of distinct products ordered and the total number of total units ordered (*productQuantity*).

SQL code:

```
# find each unique product viewed, order and quantity by each visitor
WITH unique_order_by_person AS (
  SELECT
    fullVisitorId,
    (v2ProductName) AS product_name,
    COUNT(productQuantity) AS orders,
    SUM(productQuantity) AS quantity
  FROM `data-to-insights.ecommerce.all_sessions`
  GROUP BY fullVisitorId, product_name)
# then aggregate the number of unique views, orders and quantity and sort in descending
order of unique views
SELECT
  product_name,
  COUNT(*) AS unique_view,
  SUM(orders) AS total_orders,
  SUM(quantity) AS total_quantity
FROM unique_order_by_person
GROUP BY product_name
ORDER BY unique_view DESC
LIMIT 5;
```

Result:

Row	product_name	unique_view	total_orders	total_quantity
1	Google Men's 100% Cotton Short Sleeve Hero Tee White	152372	7772	15218
2	22 oz YouTube Bottle Infuser	143784	1437	10608
3	YouTube Men's Short Sleeve Hero Tee Black	127996	2330	2859
4	YouTube Twill Cap	122081	2473	10543
5	YouTube Custom Decals	121352	4444	124153

P12. 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)*).

SQL code:

```
WITH unique_order_by_person AS (
  SELECT
    fullVisitorId,
    (v2ProductName) AS product_name,
    COUNT(productQuantity) AS orders,
    SUM(productQuantity) AS quantity
  FROM `data-to-insights.ecommerce.all_sessions`
  GROUP BY fullVisitorId, product_name)
SELECT
  product_name,
  COUNT(*) AS unique_view,
  SUM(orders) AS total_orders,
  SUM(quantity) AS total_quantity,
  # calculate average quantity per order
  (SUM(quantity)/SUM(orders)) AS avg_quantity_per_order
FROM unique_order_by_person
GROUP BY product_name
ORDER BY unique_view DESC
LIMIT 5;
```

Result:

Row	product_name	unique_view	total_orders	total_quantity	avg_quantity_per_order
1	Google Men's 100% Cotton Short Sleeve Hero Tee White	152372	7772	15218	1.9580545548121462
2	22 oz YouTube Bottle Infuser	143784	1437	10608	7.3820459290187888
3	YouTube Men's Short Sleeve Hero Tee Black	127996	2330	2859	1.227038626609442
4	YouTube Twill Cap	122081	2473	10543	4.263243024666397
5	YouTube Custom Decals	121352	4444	124153	27.937218721872188

## Part 4. Schedule your analytics queries in ETL pipeline

P13. Build a pipeline that runs queries you wrote for P4, P5, and P12.

P14. Make sure the queries run in order, P4, then P5, then P12. Schedule the pipeline so that it runs every day.

Python Code:

```
import datetime
import os
from airflow import DAG
from airflow import models
from airflow.contrib.operators.bigquery_operator import BigQueryOperator
from airflow.operators.dummy_operator import DummyOperator

today_date = datetime.datetime.now().strftime("%Y%m%d")
table_name_p4 = 'propane-shell-363803.ecommerce.total_unique_visitors'
table_name_p5 = 'propane-shell-363803.ecommerce.total_unique_visitors_by_channel'
table_name_p12 = 'propane-shell-363803.ecommerce.avg_quantity_per_order'
yesterday = datetime.datetime.combine(
    datetime.datetime.today() - datetime.timedelta(1),
    datetime.datetime.min.time())

default_dag_args = {
    # Setting start date as yesterday starts the DAG immediately when it is
    # detected in the Cloud Storage bucket.
    'start_date': yesterday,
    # To email on failure or retry set 'email' arg to your email and enable emailing here.
    'email_on_failure': False,
    'email_on_retry': False,
    # If a task fails, retry it once after waiting at least 5 minutes
    'retries': 0,
    'retry_delay': datetime.timedelta(minutes=5),
    'project_id': models.Variable.get('gcp_project')
}

with DAG(dag_id='daily_queries_dag',
        schedule_interval=datetime.timedelta(days=1),
        default_args=default_dag_args) as dag:
```



```
start = DummyOperator(task_id='start', dag = dag)
```

```
bq_query_p4 = BigQueryOperator(  
    task_id='bq_query_p4',  
    sql=f"""SELECT COUNT(*) AS product_views,  
    COUNT(DISTINCT fullVisitorId) AS unique_visitors  
    FROM `data-to-insights.ecommerce.all_sessions`;""",  
    destination_dataset_table=table_name_p4,  
    gcp_conn_id='bigquery_default',  
    use_legacy_sql=False,  
    write_disposition='WRITE_TRUNCATE',  
    create_disposition='CREATE_IF_NEEDED',  
    dag=dag)
```

```
bq_query_p5 = BigQueryOperator(  
    task_id='bq_query_p5',  
    sql= f"""SELECT channelGrouping AS referring_site,  
    COUNT(DISTINCT fullVisitorId) AS unique_visitors  
    FROM `data-to-insights.ecommerce.all_sessions`  
    GROUP BY referring_site  
    ORDER BY unique_visitors DESC;""",  
    destination_dataset_table=table_name_p5,  
    gcp_conn_id='bigquery_default',  
    use_legacy_sql=False,  
    write_disposition='WRITE_TRUNCATE',  
    create_disposition='CREATE_IF_NEEDED',  
    dag=dag)
```

```
bq_query_p12 = BigQueryOperator(  
    task_id='bq_query_p12',  
    sql = f"""WITH unique_order_by_person AS (  
        SELECT  
            fullVisitorId,  
            (v2ProductName) AS product_name,  
            COUNT(productQuantity) AS orders,  
            SUM(productQuantity) AS quantity  
        FROM `data-to-insights.ecommerce.all_sessions`  
        GROUP BY fullVisitorId, product_name)  
    SELECT  
        product_name,
```

```
COUNT(*) AS unique_view,  
SUM(orders) AS total_orders,  
SUM(quantity) AS total_quantity,  
(SUM(quantity)/SUM(orders)) AS avg_quantity_per_order  
FROM unique_order_by_person  
GROUP BY product_name  
ORDER BY unique_view DESC  
LIMIT 5,'',
```

```
destination_dataset_table=table_name_p12,  
gcp_conn_id='bigquery_default',  
use_legacy_sql=False,  
write_disposition='WRITE_TRUNCATE',  
create_disposition='CREATE_IF_NEEDED',  
dag=dag)
```

```
end = DummyOperator(task_id='end', dag = dag)
```

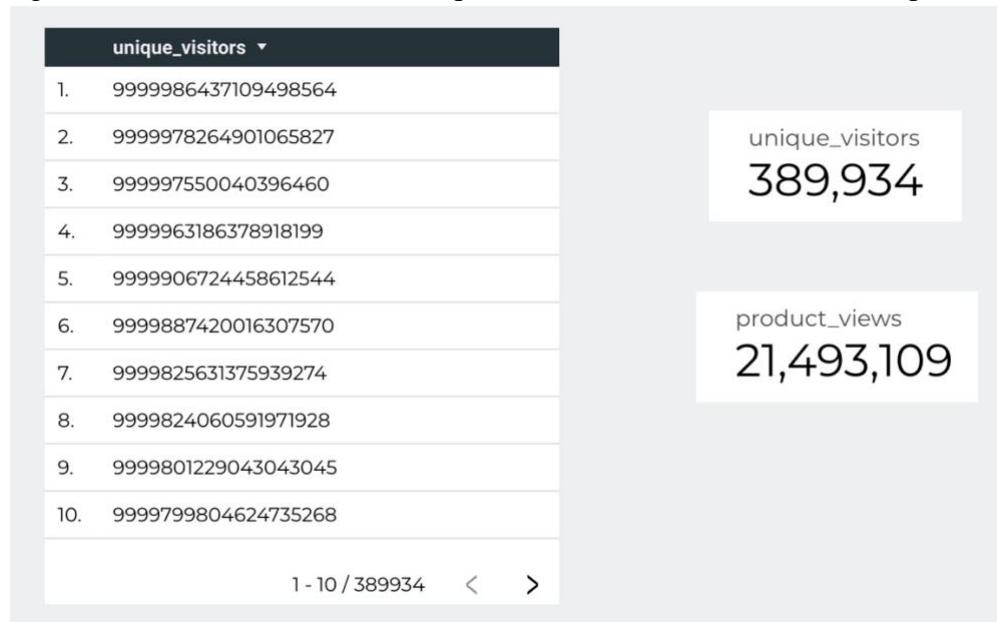
```
start >> bq_query_p4 >> bq_query_p5 >> bq_query_p12 >> end
```

## Part 5. Visualisation in Data Studio

P15. Unique visitors by ID: Add the result of P4 as a number to your dashboard (5 points).

### Screenshot from Data Studio:

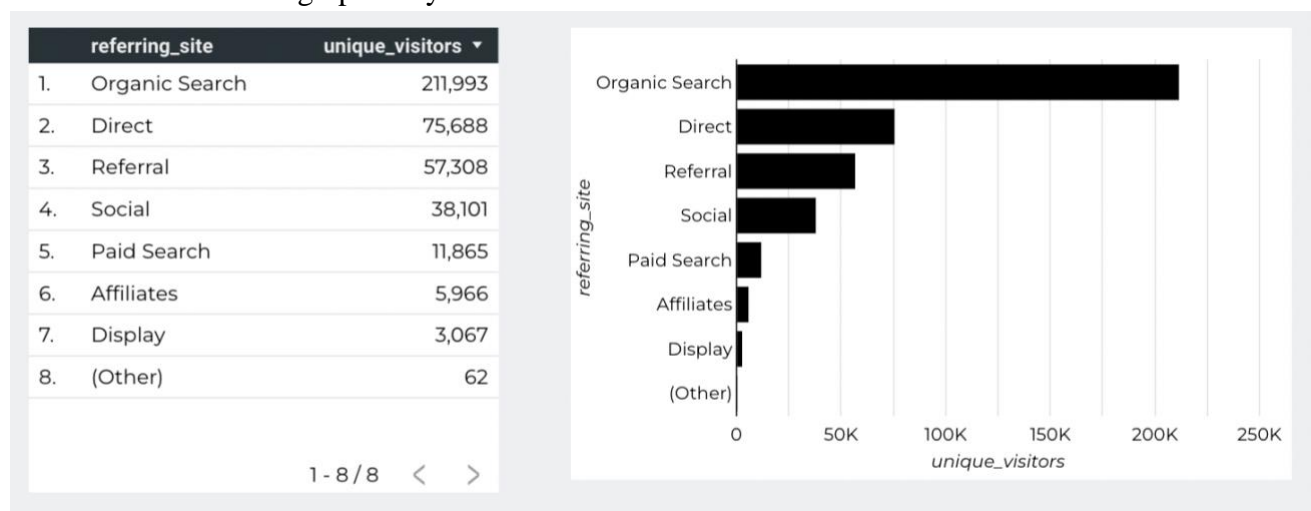
The table on the left shows a list of unique visitors by ID sorted descending. The scoreboards on the right show the number of total unique visitors and the number of total product views.



P16. Unique visitors by referring site: Add a bar chart that shows the result of the query you wrote in P5 (5 points).

### Screenshot from Data Studio:

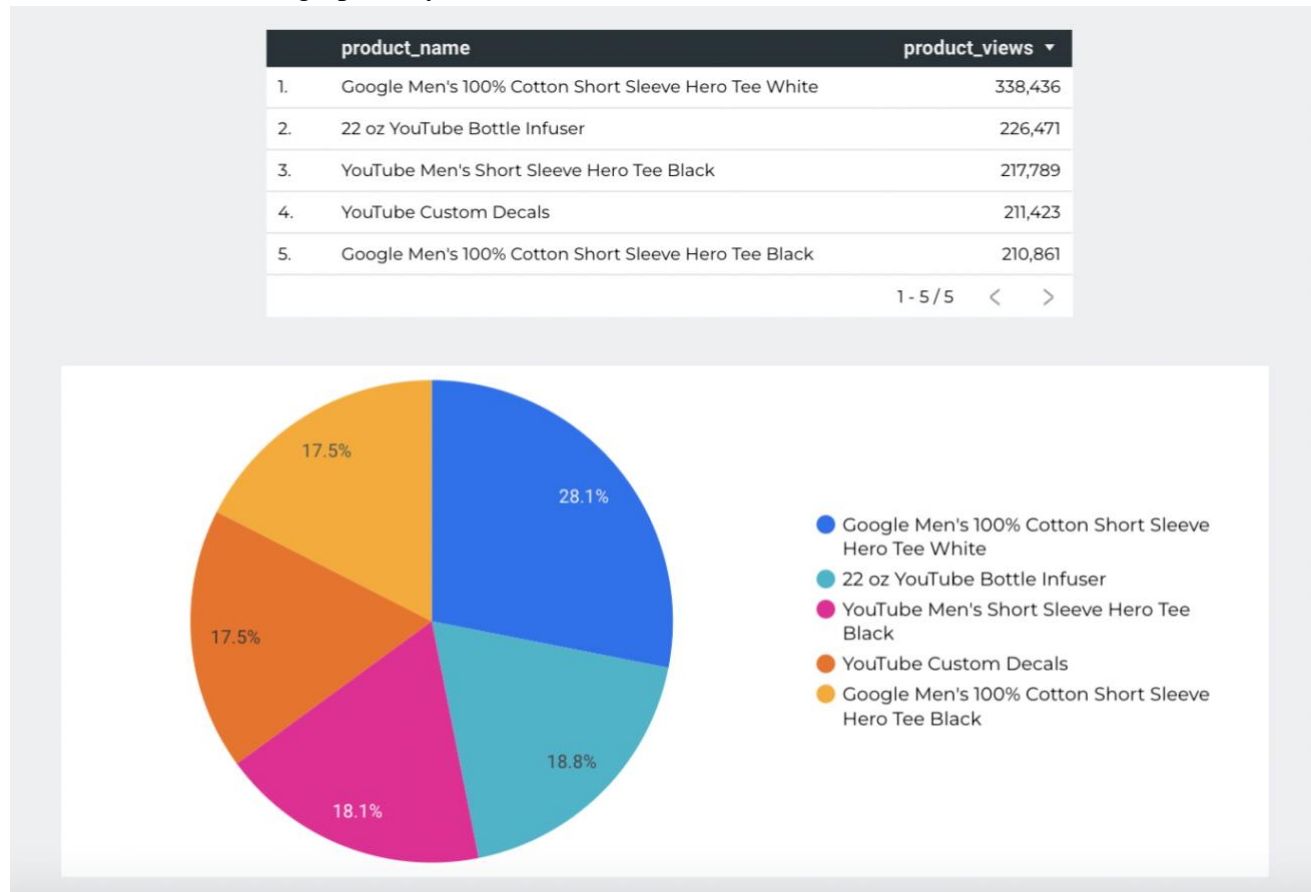
The table on the left shows the result of unique visitors by referring site. The bar chart on the right shows the same result graphically.



P17. Highly viewed products: Add a pie chart that shows the result of P9 (5 points).

Screenshot from Data Studio:

The table on the top shows the result of the top five viewed products. The pie chart on the bottom shows the same result graphically.



P18. Details of highly viewed products: Add a table that shows the result of P12 in a table (5 points).

Screenshot from Data Studio:

The table shows the details of the top five viewed products which included product name, number of unique views, total orders, total quantity and average quantity per order.

	product_name	unique_view	total_orders	total_quantity	avg_quantity_per_order
1.	Google Men's 100% Cotton Short Sleeve Hero Tee White	152,372	7,772	15,218	1.96
2.	22 oz YouTube Bottle Infuser	143,784	1,437	10,608	7.38
3.	YouTube Men's Short Sleeve Hero Tee Black	127,996	2,330	2,859	1.23
4.	YouTube Twill Cap	122,081	2,473	10,543	4.26
5.	YouTube Custom Decals	121,352	4,444	124,153	27.94

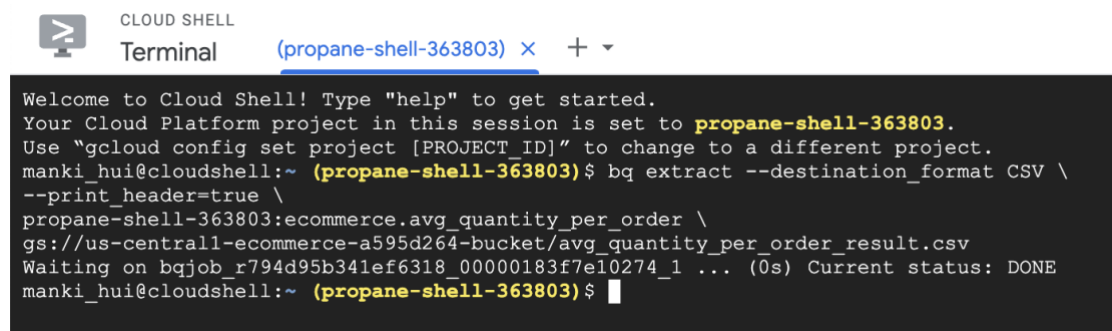
1 - 5 / 5 < >

## Part 6. Exporting the results

P19. Using a command line operation, export the result of P12 to Google Cloud storage (5 points).

```
bq extract --destination_format CSV \
--print_header=true \
propane-shell-363803:ecommerce.avg_quantity_per_order \
gs://us-central1-ecommerce-a595d264-bucket/avg_quantity_per_order_result.csv
```

Result (Cloud Shell):



```
WELCOME TO CLOUD SHELL! Type "help" to get started.
Your Cloud Platform project in this session is set to propane-shell-363803.
Use "gcloud config set project [PROJECT ID]" to change to a different project.
manki_hui@cloudshell:~ (propane-shell-363803) $ bq extract --destination_format CSV \
--print_header=true \
propane-shell-363803:ecommerce.avg_quantity_per_order \
gs://us-central1-ecommerce-a595d264-bucket/avg_quantity_per_order_result.csv
Waiting on bqjob_r794d95b341ef6318_00000183f7e10274_1 ... (0s) Current status: DONE
manki_hui@cloudshell:~ (propane-shell-363803) $
```

Result (Google Cloud Storage):

**us-central1-ecommerce-a595d264-bucket**

Location: us-central1 (Iowa) | Storage class: Standard | Public access: Subject to object ACLs | Protection: None

**OBJECTS** | CONFIGURATION | PERMISSIONS | PROTECTION | LIFECYCLE

Buckets > us-central1-ecommerce-a595d264-bucket

UPLOAD FILES | UPLOAD FOLDER | CREATE FOLDER | TRANSFER DATA | MANAGE HOLDS | DOWNLOAD | DELETE

Filter by name prefix only | Filter objects and folders

Name	Size	Type	Created	Storage class	Last modified	Public access
a595d264-4c2d-44ab-be19-05892...	6.7 KB	application/octet-stream	Oct 21, 20...	Standard	Oct 21, 20...	Not public
airflow.cfg	2.2 KB	application/octet-stream	Oct 21, 20...	Standard	Oct 21, 20...	Not public
avg_quantity_per_order_result.csv	423 B	application/octet-stream	Oct 21, 20...	Standard	Oct 21, 20...	Not public
dags/	—	Folder	—	—	—	—
data/	—	Folder	—	—	—	—

Result (generated CSV opened with Excel):

	A	B	C	D	E	F
1	product_name	unique_view	total_orders	total_quantity	avg_quantity_per_order	
2	Google Men's 100% Cotton Short Sleeve Hero Tee White	152372	7772	15218	1.958054555	
3	22 oz YouTube Bottle Infuser	143784	1437	10608	7.382045929	
4	YouTube Men's Short Sleeve Hero Tee Black	127996	2330	2859	1.227038627	
5	YouTube Twill Cap	122081	2473	10543	4.263243025	
6	YouTube Custom Decals	121352	4444	124153	27.93721872	
7						
8						

avg\_quantity\_per\_order\_result

P20. Add that step as the last step of the Airflow pipeline (5 points).

Python code (added the below code to the daily\_queries\_dag.py script):

```
export_p12 = BigQueryToCloudStorageOperator(
    task_id = 'export_p12',
    source_project_dataset_table = 'propane-shell-363803:ecommerce.avg_quantity_per_order',
    destination_cloud_storage_uris = 'gs://us-central1-ecommerce-a595d264-bucket/avg_quantity_per_order_result.csv',
    export_format='CSV', print_header=True)

start >> bq_query_p4 >> bq_query_p5 >> bq_query_p12 >> export_p12 >> end
```

Result:

Same as P19.

## Part 7. Link to YouTube

<https://youtu.be/PrQHbnC9XXE>