

# McMaster University



## SEP 720 – Cloud Computing : Assignment 2

### Qwiklabs- 1

Submitted by,

Greeshma Gopal(gopalg)

ID- 400245291

## Lab 1: Exploring Your Ecommerce Dataset with SQL in Google BigQuery

- The primary objective of this lab is to explore a dataset which is already existing using BigQuery. As a first step, the data was loaded, and it is related e-commerce.

The screenshot shows the Google Cloud Platform BigQuery console. The top navigation bar includes the Google Cloud Platform logo, the project name 'My Project 50323', and a search bar. The left sidebar contains a 'Query history' section with a list of saved queries, job history, transfers, scheduled queries, reservations, and BI Engine. Below this is a 'Resources' section with a search bar and a list of datasets: 'crack-map-265614', 'bigquery-public-data', and 'data-to-insights'. The 'data-to-insights' dataset is expanded, showing sub-datasets: 'advanced', 'customer\_insights', 'ecommerce', 'irs\_990', 'ncaa', and 'ncaa\_next'. The main area is the 'Query editor', which is currently empty. Below the editor is a 'Query history' section with a 'REFRESH' button and a list of queries. The queries are sorted by date and show the following SQL statements:

Time	Status	Query
10:56 AM	✓	select Country_Name , country_code, total_fertility_rate FROM `bigquery-public-data.census_bureau_international.age_specific_fertility_rates` LIMIT 1000
10:55 AM	✓	select country_name , country_code, total_fertility_rate FROM `bigquery-public-data.census_bureau_international.age_specific_fertility_rates` LIMIT 1000
10:49 AM	✓	SELECT country_name , country_code, total_fertility_rate FROM `bigquery-public-data.census_bureau_international.age_specific_fertility_rates` LIMIT 1000
10:49 AM	✓	SELECT country_name , country_code FROM `bigquery-public-data.census_bureau_international.age_specific_fertility_rates` LIMIT 1000
10:24 AM	✓	SELECT status, status_change_date , last_update_date FROM `bigquery-public-data.austin_311_311_request` LIMIT 1000

- `all_sessions_raw` is the dataset which was explored.

The screenshot shows the Google Cloud Platform BigQuery console with the 'ecommerce' dataset selected. The 'all\_sessions\_raw' dataset is highlighted in the left sidebar. The main area displays the 'Table info' for 'all\_sessions\_raw'. The table information is as follows:

Property	Value
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 28, 2018, 4:27:04 PM
Table expiration	Never
Last modified	Jun 5, 2018, 3:57:03 PM
Data location	US

- Viewing the schema of the dataset

The screenshot shows the BigQuery console interface. On the left, a sidebar lists datasets under the 'ecommerce' project, with 'all\_sessions\_raw' selected. The main panel displays the schema for 'all\_sessions\_raw'. At the top, there are buttons for 'Run', 'Save query', 'Save view', 'Schedule query', and 'More'. Below these, the table name 'all\_sessions\_raw' is shown with links for 'QUERY TABLE', 'COPY TABLE', 'DELETE TABLE', and 'EXPORT'. The 'Schema' tab is active, showing a table with the following columns:

Field name	Type	Mode	Description
fullVisitorId	STRING	NULLABLE	
channelGrouping	STRING	NULLABLE	
time	INTEGER	NULLABLE	
country	STRING	NULLABLE	
city	STRING	NULLABLE	

- Verifying the duplicate rows using the query.

The screenshot shows the BigQuery console with a query editor and results. The query editor contains the following SQL query:

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

Below the query editor, the 'Query results' section shows the execution status: 'Query complete (14.5 sec elapsed, 5.6 GB processed)'. The results are displayed in a table with the following columns:

Row	num_duplicate_rows	fullVisitorId	channelGrouping	time	country	city	totalTransactionRevenue	transactionId
1	3	2060318762356474784	Organic Search	2542	United States	not available in demo dataset	null	n
2	2	213097446286285593	Organic Search	90	United States	not available in demo dataset	null	n
3	3	9434100155146779985	Referral	947872	United States	Los Angeles	null	n
4	2	7388247385575647094	Social	130417	India	Chennai	null	n

At the bottom, there are pagination controls: 'Rows per page: 100', '1 - 100 of 615', 'First page', '<', '>', and '>1 Last page'.

Query editor

HIDE EDITOR

FULL SCREEN

```
4 visitId, # a visitor can have multiple visits
5 date, # session date stored as string YYYYMMDD
6 time, # time of the individual site hit (can be 0 to many per visitor session)
7 v2ProductName, # not unique since a product can have variants like Color
8 productSKU, # unique for each product
9 type, # a visitor can visit Pages and/or can trigger Events (even at the same time)
10 eCommerceAction_type, # maps to 'add to cart', 'completed checkout'
11 eCommerceAction_step,
12 eCommerceAction_option,
13 transactionRevenue, # revenue of the order
14 transactionId, # unique identifier for revenue bearing transaction
15 COUNT(*) as row_count
16 FROM
17 data-to-insights.ecommerce.all_sessions
18 GROUP BY 1,2,3 ,4, 5, 6, 7, 8, 9, 10,11,12
19 HAVING row_count > 1 # find duplicates
```

Run

Save query

Save view

Schedule query

More

This query will process 2.3 GB when run.

Query results

Query complete (8.7 sec elapsed, 2.3 GB processed)

Job information Results JSON Execution details

This query returned no results.

- Fetching the unique visitors.

Query editor

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

Run

Save query

Save view

Schedule query

More

Query results

SAVE RESULTS

EXPLORE DATA

Query complete (3.6 sec elapsed, 428 MB processed)

Job information Results JSON Execution details

Row	product_views	unique_visitors
1	21493109	389934

- Fetching the unique visitors grouping them based on the purpose of visit of site

Query editor

```

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

```

Run Save query Save view Schedule query

Query results [SAVE RESULTS](#) [EXPLORE DATA](#)

Query complete (4.1 sec elapsed, 684.6 MB processed)

Job information **Results** JSON Execution details

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)

- Querying the product names of the website

Query editor

```

1 #standardSQL
2 SELECT
3   (v2ProductName) AS ProductName
4 FROM `data-to-insights.ecommerce.all_sessions`
5 GROUP BY ProductName
6 ORDER BY ProductName

```

Run Save query Save view Schedule query More

Query results [SAVE RESULTS](#) [EXPLORE DATA](#)

Query complete (0.8 sec elapsed, 702.6 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

Rows per page: 100 1 - 100 of 633

- Fetching product views for each product

Query editor

```

2 SELECT
3   COUNT(*) AS product_views,
4   (v2ProductName) AS ProductName
5 FROM "data-to-insights.ecommerce.all_sessions"
6 WHERE type = 'PAGE'
7 GROUP BY v2ProductName
8 ORDER BY product_views DESC
9 LIMIT 5;

```

Run Save query Save view Schedule query More

Query results SAVE RESULTS EXPLORE DATA

Query complete (1.7 sec elapsed, 826.3 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

- Fetching unique views of every product

Query editor

```

1 WITH unique_product_views_by_person AS (
2   SELECT
3     fullVisitorId,
4     (v2ProductName) AS ProductName
5 FROM "data-to-insights.ecommerce.all_sessions"
6 WHERE type = 'PAGE'
7 GROUP BY fullVisitorId, v2ProductName )
8
9 SELECT
10  COUNT(*) AS unique_view_count,
11  ProductName |
12 FROM unique_product_views_by_person
13 GROUP BY ProductName
14 ORDER BY unique_view_count DESC
15 LIMIT 10

```

Run Save query Save view Schedule query More

Query results SAVE RESULTS EXPLORE DATA

Query complete (5.3 sec elapsed, 1.2 GB processed)

Job information Results JSON Execution details

Row	unique_view_count	ProductName
4	122091	YouTube Tumbler Cap
5	121288	YouTube Custom Decals
6	119600	YouTube Men's Short Sleeve Hero Tee Charcoal
7	107088	YouTube Wool Heather Cap Heather/Black
8	106279	YouTube Men's Short Sleeve Hero Tee White
9	101693	Google Men's 100% Cotton Short Sleeve Hero Tee Black

- Querying the number of orders for every product and the quantity of the same ordered.

Query editor

```
1 #standardSQL
2 SELECT
3   COUNT(*) AS product_views,
4   COUNT(productQuantity) AS orders,
5   SUM(productQuantity) AS quantity_product_ordered,
6   v2ProductName
7 FROM `data-to-insights.ecommerce.all_sessions`
8 WHERE type = 'PAGE'
9 GROUP BY v2ProductName
10 ORDER BY product_views DESC
11 LIMIT 5;
```

Query results [SAVE RESULTS](#) [EXPLORE DATA](#)

Query complete (0.9 sec elapsed, 829.1 MB processed)

Job information **Results** JSON Execution details

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

```
1 SELECT
2   COUNT(*) AS product_views,
3   COUNT(productQuantity) AS orders,
4   SUM(productQuantity) AS quantity_product_ordered,
5   SUM(productQuantity) / COUNT(productQuantity) AS avg_per_order,
6   (v2ProductName) AS ProductName
7 FROM `data-to-insights.ecommerce.all_sessions`
8 WHERE type = 'PAGE'
9 GROUP BY v2ProductName
10 ORDER BY product_views DESC
11 LIMIT 5;
```

Query results [SAVE RESULTS](#) [EXPLORE DATA](#)

Query complete (0.9 sec elapsed, 829.1 MB processed)

Job information **Results** JSON Execution details

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

## Lab 2: Ingesting New Datasets into BigQuery

- Created a new table with the name 'product'. I had created it under the dataset ecommerce.

The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Resources' tree is expanded to show the project 'crack-map-265614' and the dataset 'ecommerce'. The 'products' table is selected. The main panel displays the table's schema and data. The schema includes columns: Row, SKU, name, orderedQuantity, stockLevel, and restockingLeadTime. The data shows four rows of product information.

Row	SKU	name	orderedQuantity	stockLevel	restockingLeadTime
1	GGOEGDHQ014899	20 oz Stainless Steel Insulated Tumbler	499	652	2
2	GGOEGOAB022499	Satin Black Ballpoint Pen	403	477	2
3	GGOEYHPB072210	Twill Cap	1429	1997	2
4	GGOEGEVB071799	Pocket Bluetooth Speaker	214	246	2

## Loading data from the excel

- The data for the table has been loaded with the excel which was downloaded from qwiklabs which is product.csv.

The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Resources' tree is expanded to show the project 'crack-map-265614' and the dataset 'ecommerce'. The 'products' table is selected. The main panel displays the 'Query editor' with a SQL query and the 'Query results' table.

```
1 SELECT
2 *
3 FROM
4 ecommerce.products
5 ORDER BY
6 stockLevel DESC
7 LIMIT 10
```

The query results table shows the following data:

Row	SKU	name	orderedQuantity	stockLevel	restockingLeadTime
1	GGOEGDHC018299	22 oz Water Bottle	10075	19678	4
2	GGOEGAAX0074	22 oz Water Bottle	8942	15607	2
3	GGOEGAAX0037	Sunglasses	4204	6805	5
4	GGOEGOLC013299	Spiral Notebook and Pen Set	3582	6708	11



## Loading the data from the cloud

- Instead of loading the data using excel, it was loaded from cloud. Since the schema did not match it threw the error.

The screenshot shows the Google Cloud Platform console interface. In the background, the 'Create table' dialog is open, showing the source as 'Google Cloud Storage' with the file 'greeshmagopal/products.csv' selected. The destination is 'My Project 50323' with dataset name 'ecommerce' and table type 'Native table'. An error message is displayed in the foreground: 'Error executing job' with the message 'Empty schema specified for the load job. Please specify a schema that describes the data being loaded.' Below the error message are 'CLOSE' and 'OPEN JOB HISTORY' buttons. The 'Job history' section at the bottom shows a list of jobs. The first job, at 6:49 PM, failed with the error 'Load gs://greeshmagopal/products.csv to crack-map-265614:ecommerce.products\_cloud'. The second job, at 6:26 PM, succeeded with the message 'Load uploaded file to crack-map-265614:ecommerce.products'.

**Create table**

Source

Create table from:  Select file from GCS bucket:  Browse File format:

Destination

Project name:  Dataset name:  Table type:

Table name:

**Error executing job**

❗ Empty schema specified for the load job. Please specify a schema that describes the data being loaded.

[CLOSE](#) [OPEN JOB HISTORY](#)

[+ Add field](#)

**Partition and cluster settings**

Partitioning:

Clustering order (optional):

Clustering order determines the sort order of the data. Clustering can only be used on a partitioned table, and works with tables partitioned either by column or ingestion

[Create table](#) [Cancel](#)

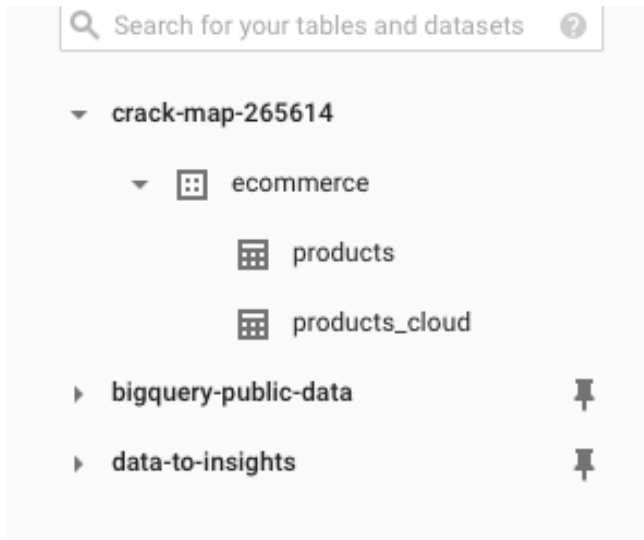
**Job history** [REFRESH](#)

[Personal history](#) [Project history](#) [Cloud Dataflow](#)

**Today**

6:49 PM	❗	Load gs://greeshmagopal/products.csv to crack-map-265614:ecommerce.products_cloud
6:26 PM	✅	Load uploaded file to crack-map-265614:ecommerce.products

- The data was uploaded to the bucket which was created earlier, and this was loaded to the table 'products\_cloud'



- Querying the data which was loaded from the cloud.

Query editor HIDE EDITOR FULL SCREEN

```

1 #standardSQL
2 SELECT
3   *,
4   SAFE_DIVIDE(orderedQuantity,stockLevel) AS ratio
5 FROM
6   ecommerce.products
7 WHERE
8   # include products that have been ordered and
9   # are 80% through their inventory
10  orderedQuantity > 0
11  AND SAFE_DIVIDE(orderedQuantity,stockLevel) >= .8
12 ORDER BY
13   restockingLeadTime DESC

```

Run Save query Save view Schedule query More This query will process 77 KB when run. ✓

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.3 sec elapsed, 77 KB processed)

Job information Results JSON Execution details

Row	SKU	name	orderedQuantity	stockLevel	restockingLeadTime	ratio
1	GGOENEBB078899	Cam Indoor Security Camera - USA	2139	2615	42	0.8179732313575526
2	GGOEGCBQ016499	SPF-15 Slim & Slender Lip Balm	3682	4069	31	0.9048906365200295
3	GGOEGQAQ020099	Four Color Retractable Pen	2002	2450	22	0.8171428571428572
4	GGOEGCBC016999	Pet Feeding Mat	105	115	19	0.9130434782608695

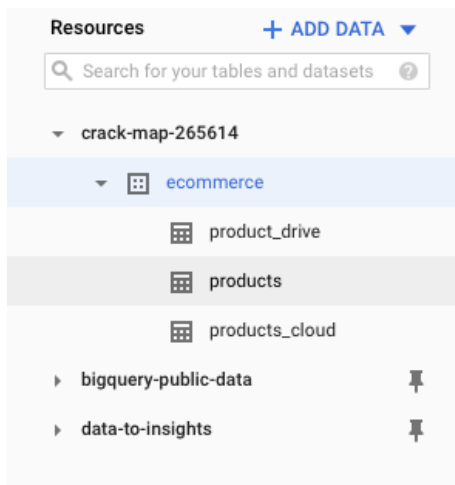
Rows per page: 100 1 - 100 of 293 First page < > >| Last page

- The results of the data is saved in google sheets. A new column 'comments' was added.

SKU	name	orderedQuantity	stockLevel	restockingLeadT	ratio
GGOEENB0078	Cam Indoor Sec	2139	2615	42	0.8179732314
GGOEGCB0016	SPP-15 Slim & S	3682	4069	31	0.9048906365
GGOEGQA0202	Four Color Retra	2002	2450	22	0.8171428571
GGOEGCB0016	Pet Feeding Mal	105	115	19	0.9130434783
GGOEQA08036	Women's Fleece	1	1	19	1
GGOEYAAJ0325	Men's Short Slee	1	1	19	1
GGOEAAWY061	Android Onesie C	2	2	19	1
GGOEAAAX058	Women's Lightw	3	3	19	1
GGOEYAEJ029C	Women's Short :	4	4	19	1
GGOEAYQ0068	Youth Girl Hoodi	4	5	19	0.8
GGOEYAEJ029C	Women's Short :	7	8	19	0.875
GGOEADJ057	Men's Performa	7	8	19	0.875
GGOEAAH0035	Android Men's Vi	8	10	19	0.8
GGOEAGVH007	Youth Girl Tee G	9	9	19	1
GGOEAAAX066	Toddler Sports T	10	10	19	1
GGOEYAB0311	Men's Short Slee	18	20	19	0.9
GGOEAAAX057	Men's Lightweig	18	20	19	0.9
GGOEAAAX066	Android Toddler :	20	22	19	0.9090909091
GGOEAAEH035	Android Men's Vi	25	27	19	0.9259259259
GGOEAGAX069	Youth Short Slee	29	31	19	0.935483871
GGOEAGALL074	Women's Short :	29	31	19	0.935483871
GGOEAGL0036	Women's Scoop	49	52	19	0.9423076923
GGOEAGAX061	Onesie Red/Gra	51	63	19	0.8095238095
GGOEGATB060	Women's Quilte	1	1	18	1

## Loading the data from google drive

- A new table 'products\_drive' was created wherein the google sheet was loaded for the data.



- Fetching the data with comments as not null(which was added newly in the google sheet).

The screenshot shows the Google Cloud Platform BigQuery console. The left sidebar contains a navigation menu with options like Query history, Saved queries, Job history, Transfers, Scheduled queries, Reservations, BI Engine, and Resources. The main area is divided into a Query editor and Query results section.

**Query editor:** The query is written in SQL: `SELECT * FROM ecommerce.product_drive WHERE comments IS NOT NULL`. An error message is displayed: "Unrecognized name: comments at [4:48]".

**Query results:** The query is complete (1.1 sec elapsed, 53.2 KB processed). The results are displayed in a table with the following columns: name, orderedQuantity, stockLevel, restockingLeadTime, ratio, and comments.

name	orderedQuantity	stockLevel	restockingLeadTime	ratio	comments	
778899	Cam Indoor Security Camera - USA	2139	2615	42	0.817973231357552	new column inserted here in the google sheet

## Lab 3: Predict Taxi Fare with a BigQuery ML Forecasting Model

- The dataset chosen was about the taxi rides in NYC. The data was loaded and queried for finding the total number of trips every month.

Query editor

```
1 #standardSQL
2 SELECT
3   TIMESTAMP_TRUNC(pickup_datetime,
4     MONTH) month,
5   COUNT(*) trips
6 FROM
7   `bigquery-public-data.new_york.tlc_yellow_trips_2015`
8 GROUP BY
9   1
10 ORDER BY
11   1
```

Run Save query Save view Schedule query More

Query results

Query complete (2.1 sec elapsed, 1.1 GB processed)

Job information Results JSON Execution details

Row	month	trips
1	2015-01-01 00:00:00 UTC	12748986
2	2015-02-01 00:00:00 UTC	12450521
3	2015-03-01 00:00:00 UTC	13351609
4	2015-04-01 00:00:00 UTC	13071789
5	2015-05-01 00:00:00 UTC	13158262
6	2015-06-01 00:00:00 UTC	12324935

Rows per page: 100 1 - 12 of 12

- The data was explored further, to find the speed of the taxis hourly.

Google Cloud Platform

BigQuery

Query editor

```
1 #standardSQL
2 SELECT
3   EXTRACT(HOUR
4     FROM
5       pickup_datetime) hour,
6   ROUND(AVG(trip_distance /
7     TIMESTAMP_DIFF(dropoff_datetime,
8       pickup_datetime,
9       SECOND)) * 3600, 1) speed
10 FROM
11   `bigquery-public-data.new_york.tlc_yellow_trips_2015`
12 WHERE
13   trip_distance > 0
14   AND fare_amount / trip_distance BETWEEN 2
15   AND 10
16   AND dropoff_datetime > pickup_datetime
```

Run Save query Save view Schedule query More

Query results

Query complete (2.6 sec elapsed, 4.4 GB processed)

Job information Results JSON Execution details

Row	hour	speed
1	0	15.8
2	1	16.3
3	2	16.8
4	3	17.5

Rows per page: 100 1 - 24 of 24 First page Last page

- Selecting the major features to create the training dataset. The WHERE statement filters out the data which don't want to be trained on.

The screenshot shows the Google Cloud Platform BigQuery console. The query editor contains the following SQL code:

```
1 #standardSQL
2 WITH params AS (
3   SELECT
4     1 AS TRAIN,
5     2 AS EVAL
6   ),
7   daynames AS
8   (SELECT ['Sun', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'] AS daysofweek),
9   taxitrans AS (
10    SELECT
11      (tolls_amount + fare_amount) AS total_fare,
12      daysofweek[ORDINAL(EXTRACT(DAYOFWEEK FROM pickup_datetime))] AS dayofweek,
13      EXTRACT(HOUR FROM pickup_datetime) AS hourofday,
14      pickup_longitude AS pickuplon,
15      pickup_latitude AS pickuplat,
16      dropoff_longitude AS dropofflon,
17      dropoff_latitude AS dropofflat,
18      passengers
19    FROM taxi.trips
20    WHERE total_fare > 0
21  )
22 SELECT * FROM taxitrans
```

The query results table shows the following data:

Row	total_fare	dayofweek	hourofday	pickuplon	pickuplat	dropofflon	dropofflat	passengers
1	4.5	Wed	0	-73.923952	40.761339	-73.916916	40.763958	1
2	6.0	Fri	0	-73.99091339111328	40.733604431152344	-74.00401306152344	40.729209899902344	1
3	6.0	Sat	0	-73.954053	40.764005	-73.964425	40.75737	1
4	6.5	Sat	0	-73.986152	40.722517	-73.982417	40.735332	1
5	6.5	Sat	0	-73.963754	40.761287	-73.977521	40.763239	2
6	7.0	Thurs	0	-74.006762	40.735782	-74.004255	40.721887	1
7	8.0	Wed	0	0.0	0.0	-73.974238	40.760707	5
8	8.0	Sun	0	-73.966339	40.767947	-73.947692	40.775099	1

- Choosing the model for the dataset. Since the price varies linear with the distance and time, linear regression can be chosen as the model. The query result will be 'model created'.

The screenshot shows the Google Cloud Platform BigQuery console. The query editor contains the following SQL code:

```
1 CREATE or REPLACE MODEL taxi.taxifare_model
2 OPTIONS
3   (model_type='linear_reg', labels=['total_fare']) AS
4
5 WITH params AS (
6   SELECT
7     1 AS TRAIN,
8     2 AS EVAL
9   ),
10   daynames AS
11   (SELECT ['Sun', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'] AS daysofweek),
12   taxitrans AS (
13    SELECT
14      (tolls_amount + fare_amount) AS total_fare,
15      daysofweek[ORDINAL(EXTRACT(DAYOFWEEK FROM pickup_datetime))] AS dayofweek,
16      EXTRACT(HOUR FROM pickup_datetime) AS hourofday,
17      pickup_longitude AS pickuplon,
18      pickup_latitude AS pickuplat,
19      dropoff_longitude AS dropofflon,
20      dropoff_latitude AS dropofflat,
21      passengers
22    FROM taxi.trips
23    WHERE total_fare > 0
24  )
25 SELECT * FROM taxitrans
```

The query results table shows the following data:

Row	total_fare	dayofweek	hourofday	pickuplon	pickuplat	dropofflon	dropofflat	passengers
1	4.5	Wed	0	-73.923952	40.761339	-73.916916	40.763958	1
2	6.0	Fri	0	-73.99091339111328	40.733604431152344	-74.00401306152344	40.729209899902344	1
3	6.0	Sat	0	-73.954053	40.764005	-73.964425	40.75737	1
4	6.5	Sat	0	-73.986152	40.722517	-73.982417	40.735332	1
5	6.5	Sat	0	-73.963754	40.761287	-73.977521	40.763239	2
6	7.0	Thurs	0	-74.006762	40.735782	-74.004255	40.721887	1
7	8.0	Wed	0	0.0	0.0	-73.974238	40.760707	5
8	8.0	Sun	0	-73.966339	40.767947	-73.947692	40.775099	1

- Root mean square is chosen as the loss function to validate the predicted data. The rmse is higher in this case.

```

1 #standardSQL
2 SELECT
3   SQRT(mean_squared_error) AS rmse
4 FROM
5   ML.EVALUATE(MODEL taxi.taxifare_model,
6   (
7
8   WITH params AS (
9     SELECT
10      1 AS TRAIN,
11      2 AS EVAL
12    ),
13
14   daynames AS
15     (SELECT ['Sun', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'] AS daysofweek),
16
17   taxitrips AS (
18     SELECT
19       (tolls_amount + fare_amount) AS total_fare,

```

Run Save query Save view Schedule query More

Query results SAVE RESULTS EXPLORE DATA

Query complete (4.8 sec elapsed, 74.3 GB processed)

Job information Results JSON Execution details

Row	rmse
1	9.476637019770271

- Since there are outliers in the dataset (fares are negative and some are more than \$50,000), we need to filter out this data for better prediction.

Query results SAVE RESULTS EXPLORE DATA

Query complete (12.0 sec elapsed, 74.3 GB processed)

Job information Results JSON Execution details

Row	predicted_total_fare	total_fare	dayofweek	hourofday	pickuplon	pickuplat	dropofflon	dropofflat	pass
1	11.502443386235797	3.5	Wed	0	-74.003658	40.74031	-73.997352	40.737686	
2	11.95925140979752	4.0	Wed	0	-73.954907	40.76972	-73.956013	40.775472	
3	11.222868131199554	4.0	Sat	0	-73.938094	40.796674	-73.944337	40.788055	
4	11.222907321676589	4.5	Sat	0	-73.985415	40.731798	-73.987391	40.72688	
5	11.396090850265768	5.0	Tues	0	-73.98422	40.749068	-73.97328	40.743545	
6	11.648081503904997	5.5	Thurs	0	-73.997277	40.721625	-73.992042	40.734562	
7	11.502480822029849	6.0	Wed	0	-74.00328063964844	40.72758483886719	-74.00617980957031	40.73969650268555	
8	11.396114108211755	6.5	Tues	0	-73.985543	40.768008	-74.005034	40.74127	
9	11.648057063763122	6.5	Thurs	0	-73.862743	40.768847	-73.876562	40.764175	
10	11.738753301431082	6.5	Tues	0	-73.99188	40.7351	-73.999481	40.743964	
11	12.219062247997142	7.5	Thurs	0	-73.9981	40.761095	-73.996765	40.757902	

Rows per page: 100 1 - 100 of 1111106 First page < > >| Last page

{{ctrl.queryPanelTitle}}Unsaved query
 Edited

```

1 SELECT
2   COUNT(fare_amount) AS num_fares,
3   MIN(fare_amount) AS low_fare,
4   MAX(fare_amount) AS high_fare,
5   AVG(fare_amount) AS avg_fare,
6   STDDEV(fare_amount) AS stddev
7 FROM
8   nyc-tlc.yellow.trips
      
```

Run

Save query

Save view

Schedule query

More

Query results

SAVE RESULTS

EXPLORE DATA

Query complete (1.5 sec elapsed, 8.3 GB processed)

Job information
 Results
 JSON
 Execution details

Row	num_fares	low_fare	high_fare	avg_fare	stddev
1	1108779463	-2.1474808E7	503325.53	11.105718581071878	650.4445803206457

- The data has been filtered to choose the ones with fares between 6 to 200.

{{ctrl.queryPanelTitle}}Query editor

```

1 SELECT
2   COUNT(fare_amount) AS num_fares,
3   MIN(fare_amount) AS low_fare,
4   MAX(fare_amount) AS high_fare,
5   AVG(fare_amount) AS avg_fare,
6   STDDEV(fare_amount) AS stddev
7 FROM
8   nyc-tlc.yellow.trips
9   WHERE trip_distance > 0 AND fare_amount BETWEEN 6 and 200
      
```

Run

Save query

Save view

Schedule query

More

Query results

SAVE RESULTS

EXPLORE DATA

Query complete (0.9 sec elapsed, 16.5 GB processed)

Job information
 Results
 JSON
 Execution details

Row	num_fares	low_fare	high_fare	avg_fare	stddev
1	843834902	6.0	200.0	12.992423677031079	9.1520078369226



- Also we will pick the data which is specific to New York city.

```

8 nyc-tlc.yellow.trips`
9 WHERE trip_distance > 0 AND fare_amount BETWEEN 6 and 200
10 AND pickup_longitude > -75
11 AND pickup_longitude < -73
12 AND dropoff_longitude > -75
13 AND dropoff_longitude < -73
14 AND pickup_latitude > 40
15 AND pickup_latitude < 42
16 AND dropoff_latitude > 40

```

Run Save query Save view Schedule query More

Query results SAVE RESULTS EXPLORE DATA

Query complete (1.8 sec elapsed, 49.6 GB processed)

Job information Results JSON Execution details

Row	num_fares	low_fare	high_fare	avg_fare	stddev
1	827365869	6.0	200.0	12.989136200806948	9.13980779190728

- The different model(same linear regression) was created taxifare\_model\_2 for retraining.

Run Save query Save view Schedule query More

Query results

Query complete (59.4 sec elapsed, 74.3 GB (ML) processed)

Job information Results JSON Execution details

*i* This statement created a new model named crack-map-265614:taxi.taxifare\_model\_2.

- The root mean square was calculated again and was lesser than the first one after when the outliers were filtered out.

Query editor

```
27 WHERE trip_distance > 0 AND fare_amount BETWEEN 6 and 200
28     AND pickup_longitude > -75 #limiting of the distance the taxis travel out
29     AND pickup_longitude < -73
30     AND dropoff_longitude > -75
31     AND dropoff_longitude < -73
32     AND pickup_latitude > 40
33     AND pickup_latitude < 42
34     AND dropoff_latitude > 40
35     AND dropoff_latitude < 42
36     AND MOD(ABS(FARM_FINGERPRINT(CAST(pickup_datetime AS STRING))),1000) = params.EVAL
37 )
38
39 SELECT *
40 FROM taxitrips
41
42 ))
```

Run

Save query

Save view

Schedule query

More

Query results

SAVE RESULTS

EXPLORE DATA

Query complete (3.5 sec elapsed, 74.3 GB processed)

Job information

Results

JSON

Execution details

Row	rmse
1	5.124652777769157