Tony Nguyen

Dr. Shawn Bowers

CPSC 324 01

15 April 2024

Homework 5

1. Question 1

- Model information and training statistics
    - The statistics provide information on the number of iterations run, the training and evaluation loss rate, the learning rate, and the duration in seconds.
    - Among the iterations, the run time is pretty consistent without any significant changes.
    - Between the training and evaluation loss rates, the two numbers are pretty consistent as well.
    - The two loss rates and learning rates are inversely correlated. This one can easily be seen using the graph features.
- Basic syntaxes
    - Create model

```
#standardSQL
CREATE OR REPLACE MODEL `bqml_lab.sample_model`
OPTIONS(model_type='logistic_reg') AS
SELECT
  IF(totals.transactions IS NULL, 0, 1) AS label,
  IFNULL(device.operatingSystem, "") AS os,
  device.isMobile AS is_mobile,
  IFNULL(geoNetwork.country, "") AS country,
  IFNULL(totals.pageviews, 0) AS pageviews
FROM
  `bigquery-public-data.google_analytics_sample.ga_sessions_*`
WHERE
  _TABLE_SUFFIX BETWEEN '20160801' AND '20170631'
```

```
LIMIT 100000;
```

- Predict model

```
#standardSQL
SELECT
  *
FROM
  ml.EVALUATE(MODEL `bqml_lab.sample_model`, (
SELECT
  IF(totals.transactions IS NULL, 0, 1) AS label,
  IFNULL(device.operatingSystem, "") AS os,
  device.isMobile AS is_mobile,
  IFNULL(geoNetwork.country, "") AS country,
  IFNULL(totals.pageviews, 0) AS pageviews
FROM
  `bigquery-public-data.google_analytics_sample.ga_sessions_*`
WHERE
  _TABLE_SUFFIX BETWEEN '20170701' AND '20170801'));
```

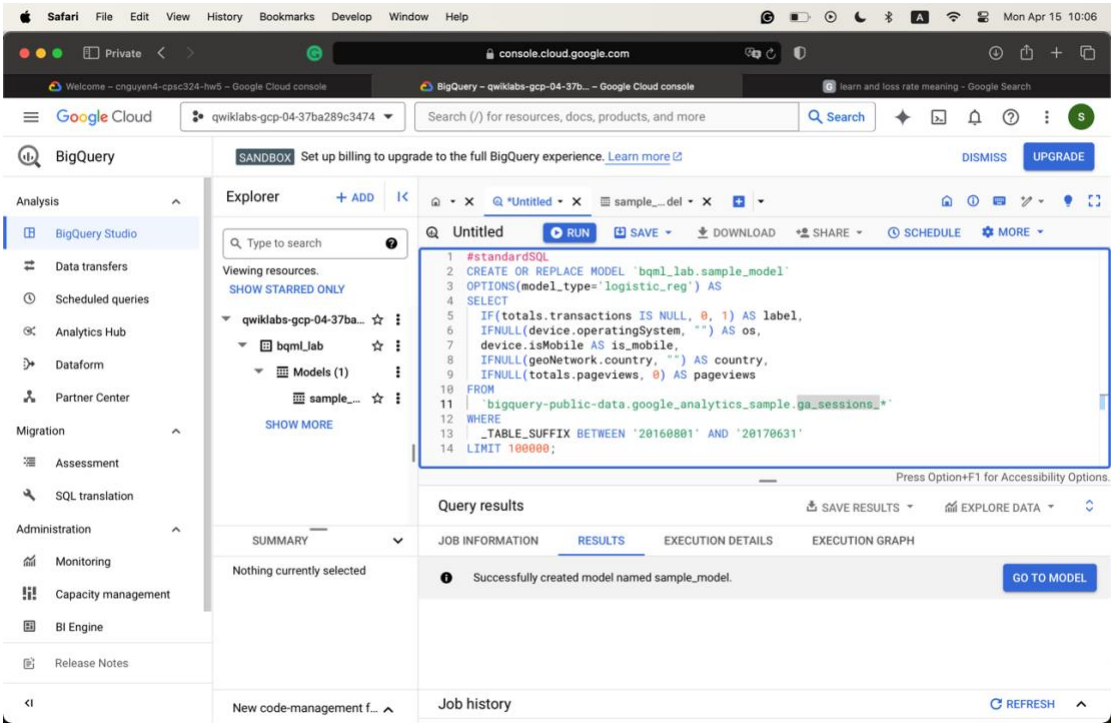
- Evaluate model

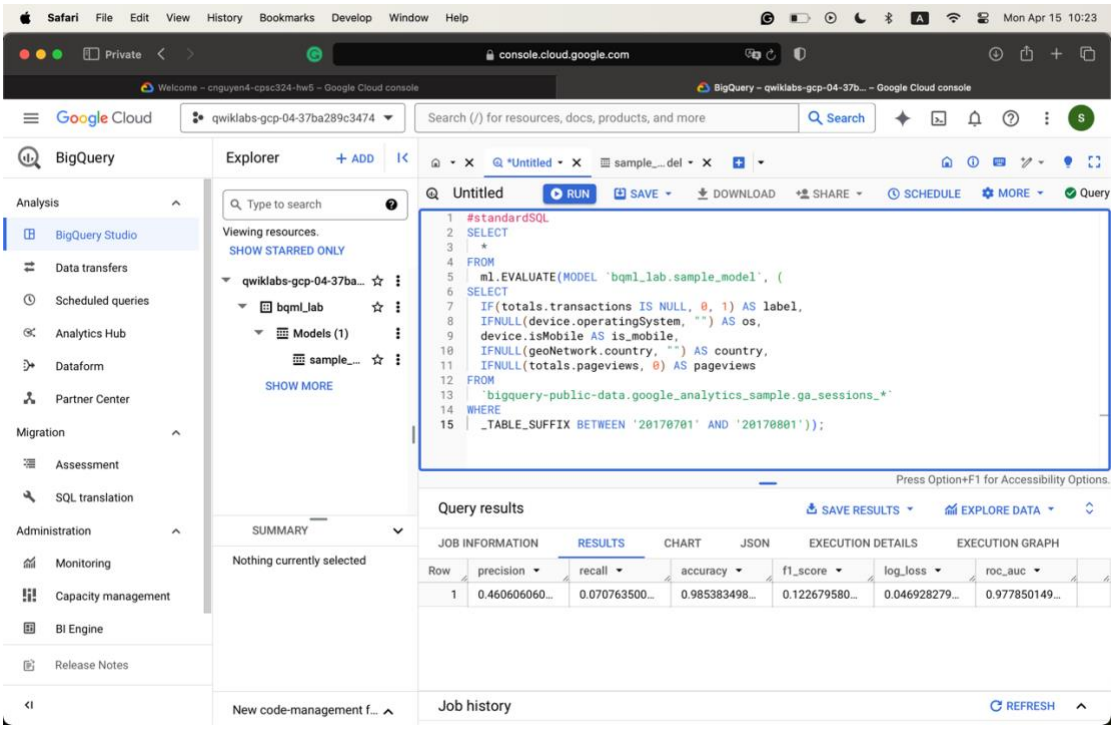
```
#standardSQL
SELECT
  country,
  SUM(predicted_label) as total_predicted_purchases
FROM
  ml.PREDICT(MODEL `bqml_lab.sample_model`, (
SELECT
  IFNULL(device.operatingSystem, "") AS os,
  device.isMobile AS is_mobile,
  IFNULL(totals.pageviews, 0) AS pageviews,
  IFNULL(geoNetwork.country, "") AS country
FROM
  `bigquery-public-data.google_analytics_sample.ga_sessions_*`
WHERE
  _TABLE_SUFFIX BETWEEN '20170701' AND '20170801'))
GROUP BY country
ORDER BY total_predicted_purchases DESC
LIMIT 10;
```

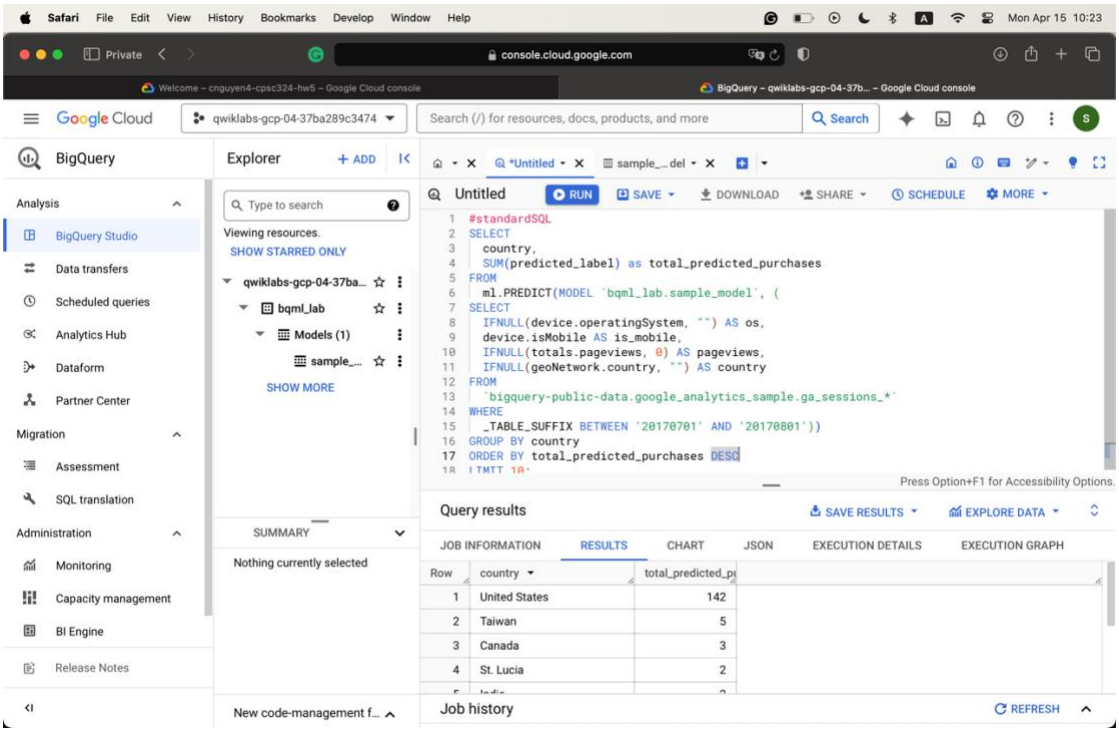
- Predicted features and features used to build the model
  - Predicted features: visitor transactions

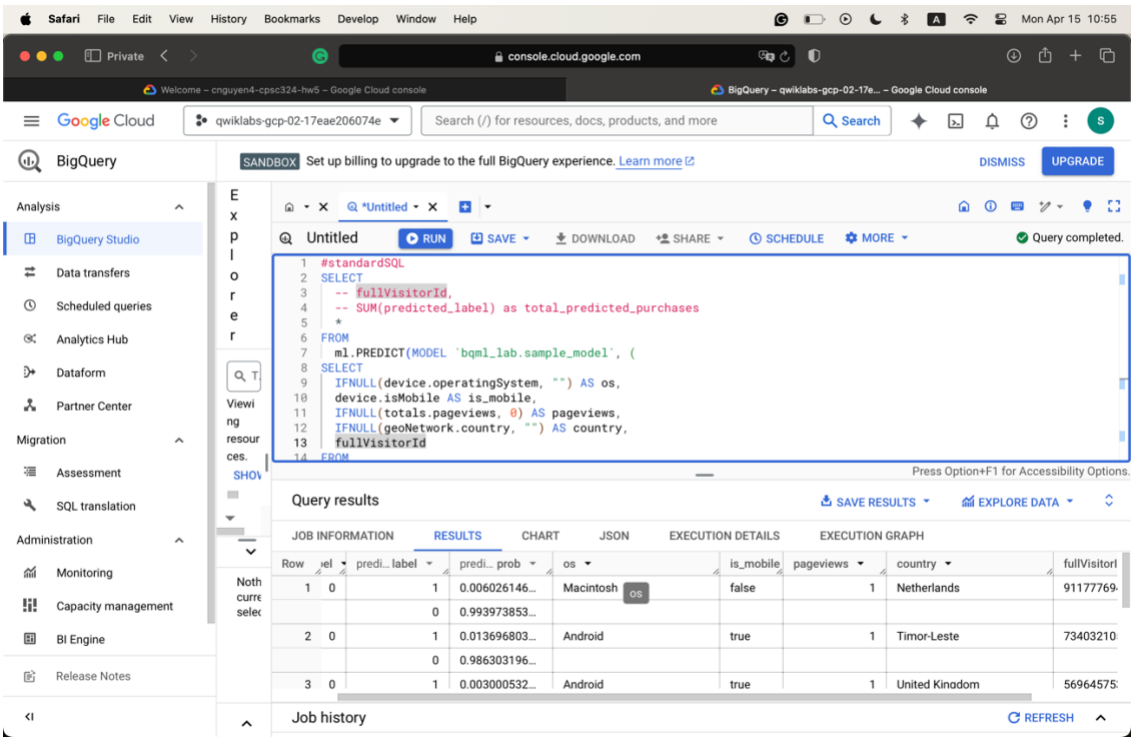
- Features used to build the model:
  - total.transactions
  - device.operatingSystem
  - device.isMobile
  - geoNetwork.country
  - totals.pageviews

- BigQuery ML returns
  - Create model
    - It does not really return anything other than saying the model has been created.
  - Evaluate model
    - It returns the statistics that are used to evaluate the model
    - In this case, we have:
      - Precision
      - Recall
      - Accuracy
      - F1_score
      - Log_loss
      - Roc_auc
  - Prediction
    - The prediction step returns the predicted label for each instance.
      - The purchase per country prediction: All columns and the predicted label

- Return all schema
    - Row
    - predicted_label
    - predicted_label_probs.label
    - predicted_label_probs.prob
    - os
    - is_mobile
    - pageviews
    - country
- Screenshots
    - 


Safari File Edit View History Bookmarks Develop Window Help
Mon Apr 15 10:23
console.cloud.google.com
Google Cloud
qwiklabs-gcp-04-37ba289c3474
Search (/) for resources, docs, products, and more
Search
BigQuery
Explorer
+ ADD
Analysis
BigQuery Studio
Data transfers
Scheduled queries
Analytics Hub
Dataform
Partner Center
Migration
Assessment
SQL translation
Administration
Monitoring
Capacity management
BI Engine
Release Notes
Type to search
Viewing resources.
SHOW STARRED ONLY
qwiklabs-gcp-04-37ba...
bqml_lab
Models (1)
sample_...
SHOW MORE
SUMMARY
Nothing currently selected
New code-management f...
Untitled
RUN
SAVE
DOWNLOAD
SHARE
SCHEDULE
MORE
Query
#standardSQL
SELECT
*
FROM
ml.EVALUATE(MODEL `bqml_lab.sample_model`, (
SELECT
IF(totals.transactions IS NULL, 0, 1) AS label,
IFNULL(device.operatingSystem, "") AS os,
device.isMobile AS is_mobile,
IFNULL(geoNetwork.country, "") AS country,
IFNULL(totals.pageviews, 0) AS pageviews
FROM
`bigquery-public-data.google_analytics_sample.ga_sessions_*`
WHERE
_TABLE_SUFFIX BETWEEN '20170701' AND '20170801'));
Press Option+F1 for Accessibility Options.
Query results
SAVE RESULTS
EXPLORE DATA
JOB INFORMATION
RESULTS
CHART
JSON
EXECUTION DETAILS
EXECUTION GRAPH
Row
precision
recall
accuracy
f1_score
log_loss
roc_auc
1
0.460606060...
0.070763500...
0.985383498...
0.122679580...
0.046928279...
0.977850149...
Job history
REFRESH


Safari File Edit View History Bookmarks Develop Window Help
Mon Apr 15 10:23
console.cloud.google.com
Google Cloud
qwiklabs-gcp-04-37ba289c3474
Search (/) for resources, docs, products, and more
Search
BigQuery
Explorer
+ ADD
Analysis
BigQuery Studio
Data transfers
Scheduled queries
Analytics Hub
Dataform
Partner Center
Migration
Assessment
SQL translation
Administration
Monitoring
Capacity management
BI Engine
Release Notes
Type to search
Viewing resources.
SHOW STARRED ONLY
qwiklabs-gcp-04-37ba...
bqml_lab
Models (1)
sample_...
SHOW MORE
SUMMARY
Nothing currently selected
New code-management f...
Untitled
RUN
SAVE
DOWNLOAD
SHARE
SCHEDULE
MORE
#standardSQL
SELECT
country,
SUM(predicted_label) as total_predicted_purchases
FROM
ml.PREDICT(MODEL `bqml_lab.sample_model`, (
SELECT
IFNULL(device.operatingSystem, "") AS os,
device.isMobile AS is_mobile,
IFNULL(totals.pageviews, 0) AS pageviews,
IFNULL(geoNetwork.country, "") AS country
FROM
`bigquery-public-data.google_analytics_sample.ga_sessions_*`
WHERE
_TABLE_SUFFIX BETWEEN '20170701' AND '20170801'))
GROUP BY country
ORDER BY total_predicted_purchases DESC
Press Option+F1 for Accessibility Options.
Query results
SAVE RESULTS
EXPLORE DATA
JOB INFORMATION
RESULTS
CHART
JSON
EXECUTION DETAILS
EXECUTION GRAPH
Row
country
total_predicted_p
1
United States
142
2
Taiwan
5
3
Canada
3
4
St. Lucia
2
Job history
REFRESH



2. Question 2
   - Describe each task
     - Task 1
       - Run SQL codes to explore the dataset
       - The first command: What percentage of the total visitors who visited our website made a purchase?
       - The second command: The top 5 selling products.
     - Task 2
       - Identify an objective to know what we should do. This can help with feature engineering.
     - Task 3
       - Select features and create the training set

- Runs several queries to see which feature should be selected to develop the model

- Task 4
  - Create a BigQuery dataset to store models
- Task 5
  - Select BigQueryML model type
  - Build a model

```
CREATE OR REPLACE MODEL `ecommerce.classification_model`
OPTIONS
(
model_type='logistic_reg',
labels = ['will_buy_on_return_visit']
)
AS

#standardSQL
SELECT
  * EXCEPT(fullVisitorId)
FROM

  # features
  (SELECT
    fullVisitorId,
    IFNULL(totals.bounces, 0) AS bounces,
    IFNULL(totals.timeOnSite, 0) AS time_on_site
  FROM
    `data-to-insights.ecommerce.web_analytics`
  WHERE
    totals.newVisits = 1
    AND date BETWEEN '20160801' AND '20170430') # train on first 9
months
  JOIN
  (SELECT
    fullvisitorid,
    IF(COUNTIF(totals.transactions > 0 AND totals.newVisits IS
NULL) > 0, 1, 0) AS will_buy_on_return_visit
  FROM
      `data-to-insights.ecommerce.web_analytics`
  GROUP BY fullvisitorid)
  USING (fullVisitorId)
```

;

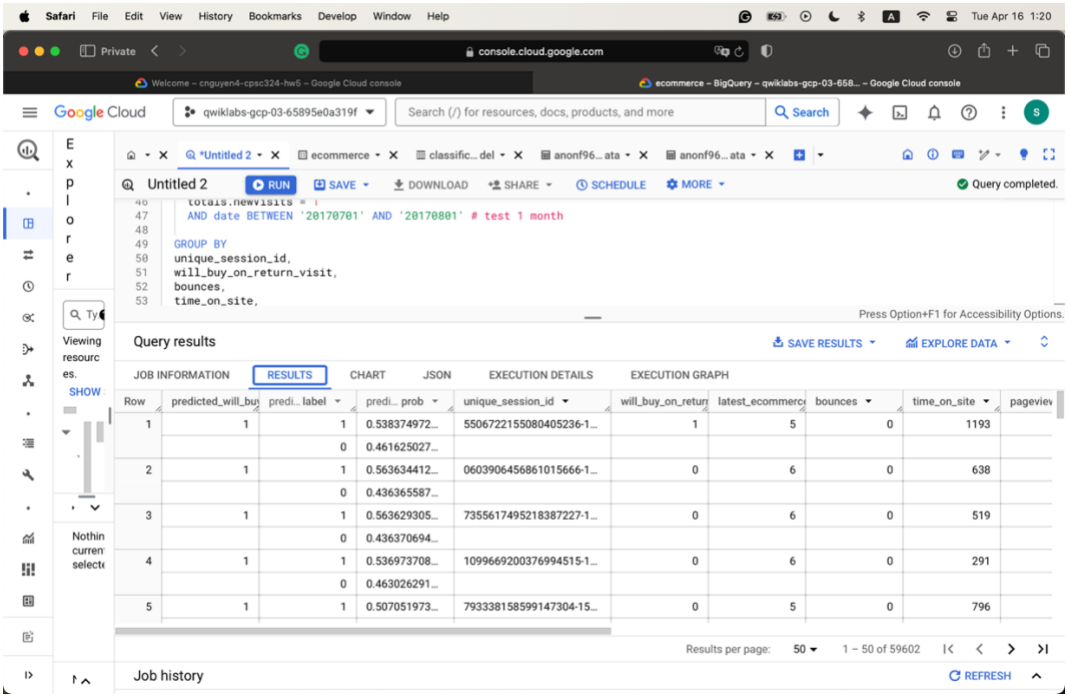
- Task 6
  - Evaluate classification model performance

```
SELECT
  roc_auc,
  CASE
    WHEN roc_auc > .9 THEN 'good'
    WHEN roc_auc > .8 THEN 'fair'
    WHEN roc_auc > .7 THEN 'decent'
    WHEN roc_auc > .6 THEN 'not great'
  ELSE 'poor' END AS model_quality
FROM
  ML.EVALUATE(MODEL ecommerce.classification_model,  (

SELECT
  * EXCEPT(fullVisitorId)
FROM

  # features
  (SELECT
    fullVisitorId,
    IFNULL(totals.bounces, 0) AS bounces,
    IFNULL(totals.timeOnSite, 0) AS time_on_site
  FROM
    `data-to-insights.ecommerce.web_analytics`
  WHERE
    totals.newVisits = 1
    AND date BETWEEN '20170501' AND '20170630') # eval on 2
months
  JOIN
  (SELECT
    fullvisitorid,
    IF(COUNTIF(totals.transactions > 0 AND totals.newVisits IS
NULL) > 0, 1, 0) AS will_buy_on_return_visit
  FROM
      `data-to-insights.ecommerce.web_analytics`
  GROUP BY fullvisitorid)
  USING (fullVisitorId)

));
```

- Task 7
  - Improve performance with feature engineering.

- Use the same code as above, but create another model to feature engineer different features and control.
- Run the evaluation again to confirm/see the same metric.

- Task 8
  - Predict which new visitor will come back and purchase
  - Similar to the run prediction method
    - This code has been applied
- Task 9:
  - Analyze results and additional information
  - Based on the prediction result

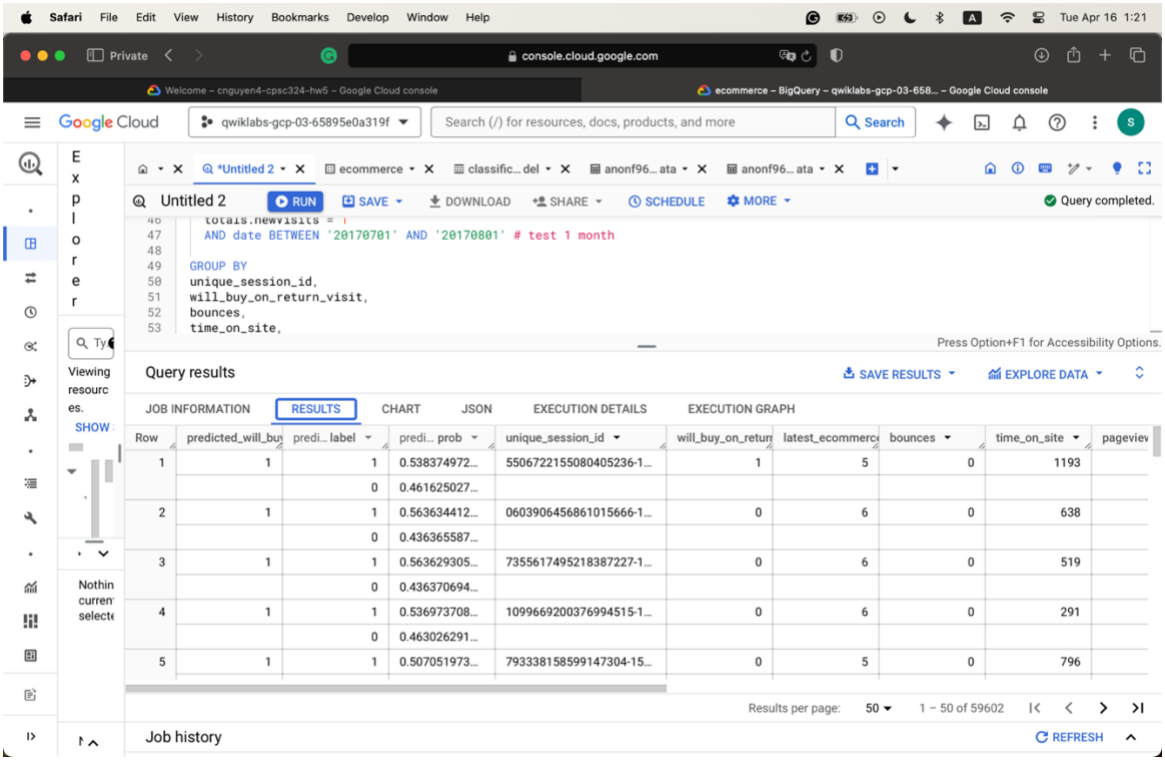


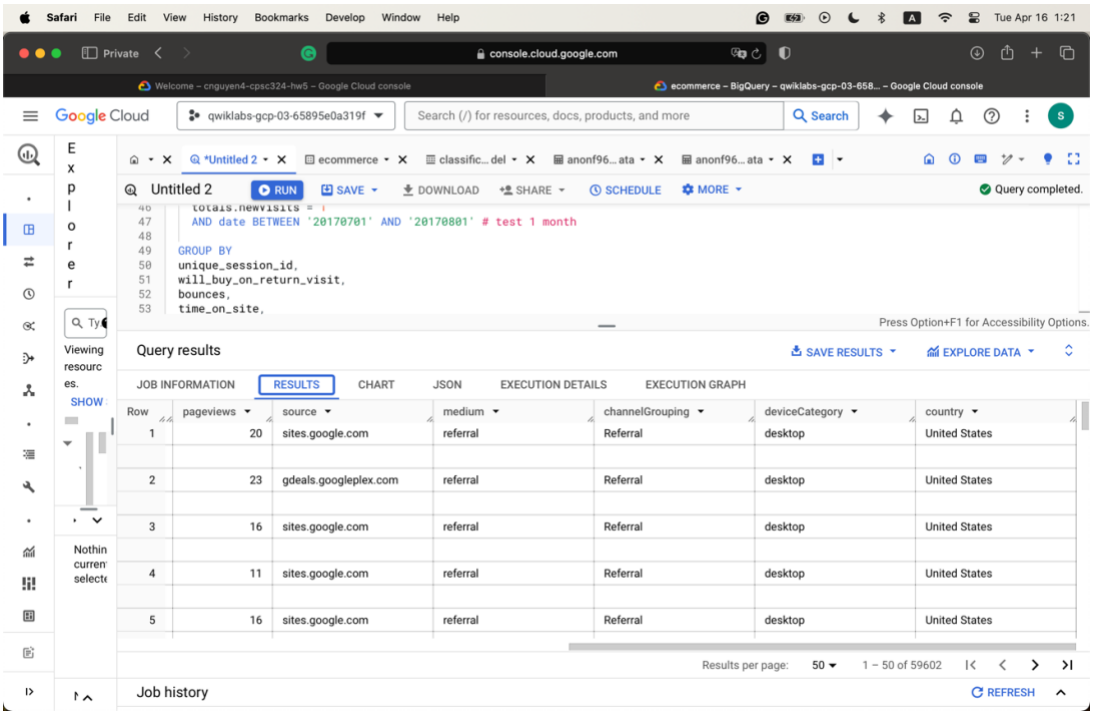
- Task 10

- Data ranges for Task 5 and Task 6. Instances used for testing
  - Task 5

        - Train: 20160801 – 20170430. Used 563,076 rows for training
    - Task 6
        - Evaluate: 20170501 – 20170630. Used 9,978 instances for testing

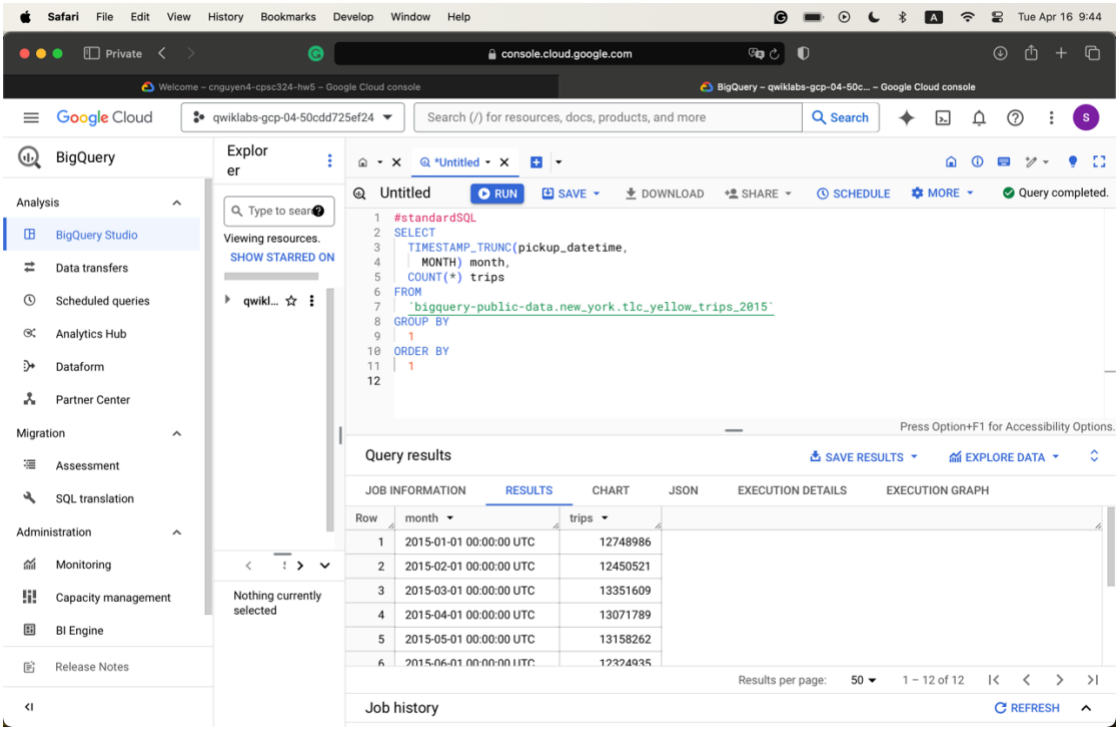
- Screenshots
    -

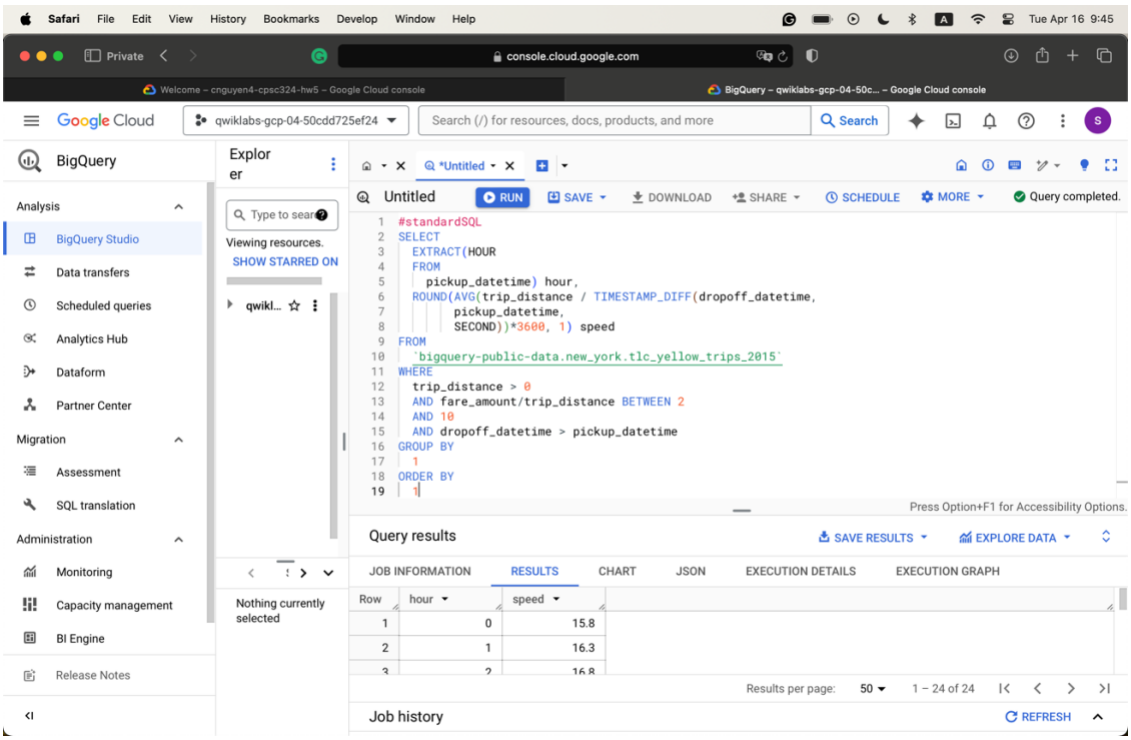




3. Question 3

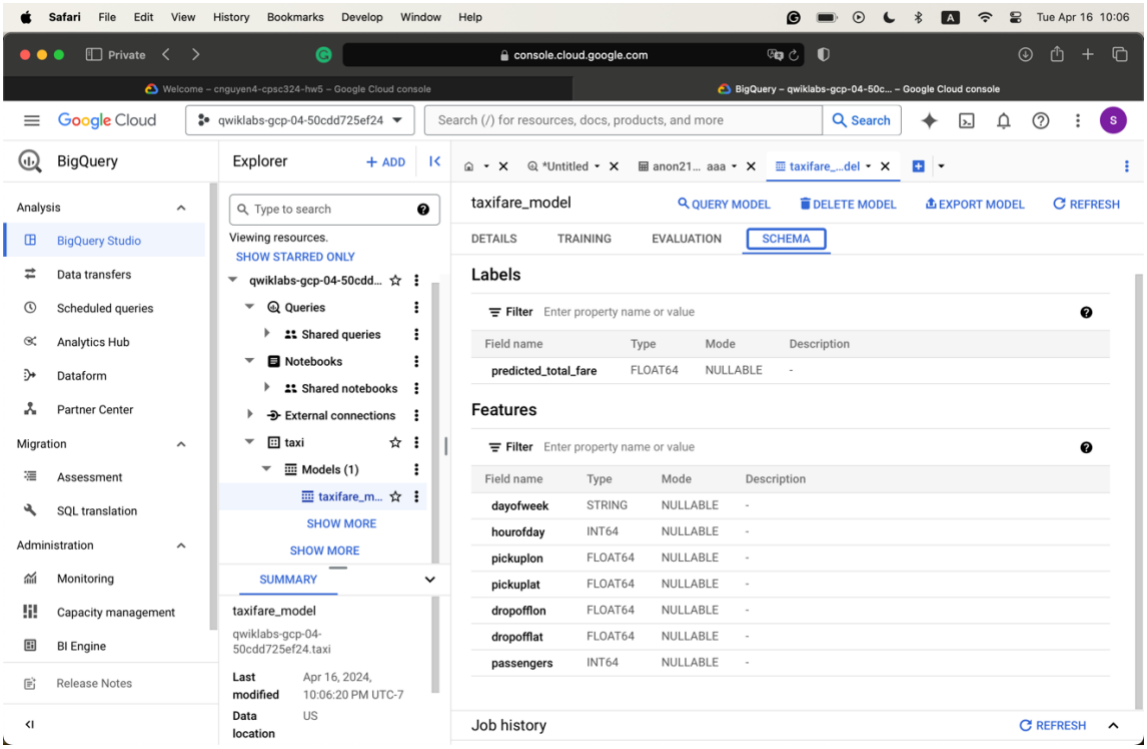
- Screenshots





- The number of rows returned in Task 3 is 1120077

- Screenshot of the schema of the training model



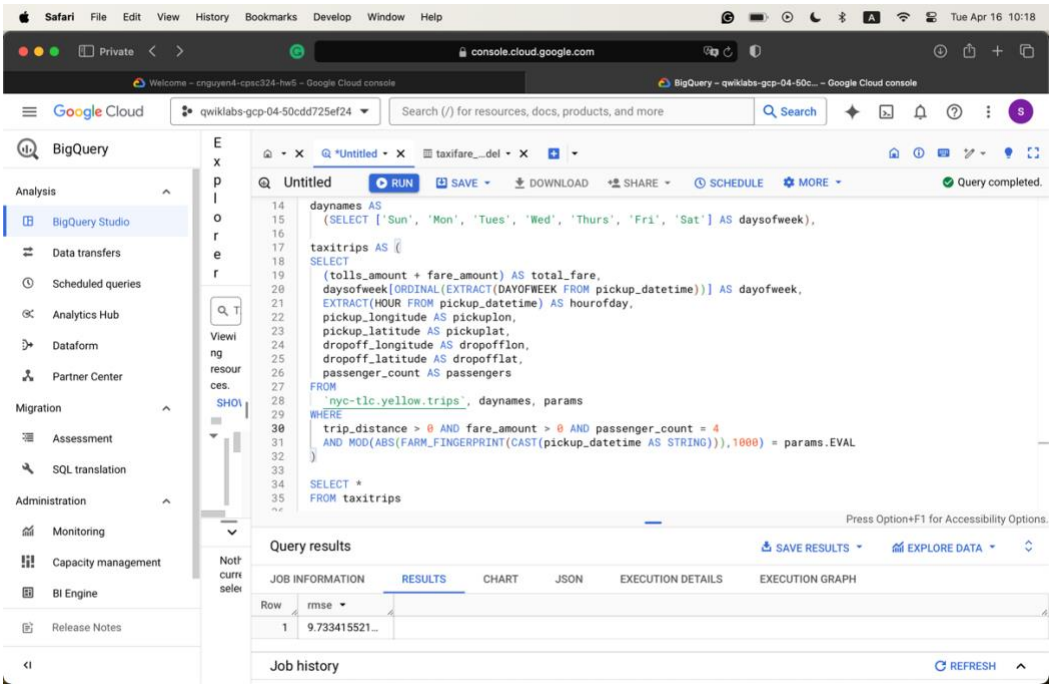
```
CREATE or REPLACE MODEL taxi.taxifare_model
OPTIONS
  (model_type='linear_reg', labels=['total_fare']) AS

WITH params AS (
    SELECT
    1 AS TRAIN,
    2 AS EVAL
    ),

  daynames AS
    (SELECT ['Sun', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'] AS
daysofweek),

  taxitrips AS (
  SELECT
    (tolls_amount + fare_amount) AS total_fare,
    daysofweek[ORDINAL(EXTRACT(DAYOFWEEK FROM pickup_datetime))] AS
dayofweek,
    EXTRACT(HOUR FROM pickup_datetime) AS hourofday,
    pickup_longitude AS pickuplon,
    pickup_latitude AS pickuplat,
    dropoff_longitude AS dropofflon,
    dropoff_latitude AS dropofflat,
    passenger_count AS passengers
  FROM
    `nyc-tlc.yellow.trips`, daynames, params
  WHERE
    trip_distance > 0 AND fare_amount > 0
    AND MOD(ABS(FARM_FINGERPRINT(CAST(pickup_datetime AS
STRING))),1000) = params.TRAIN
  )

  SELECT *
  FROM taxitrips
```

- Evaluation query
    - Number of rows returned: 1

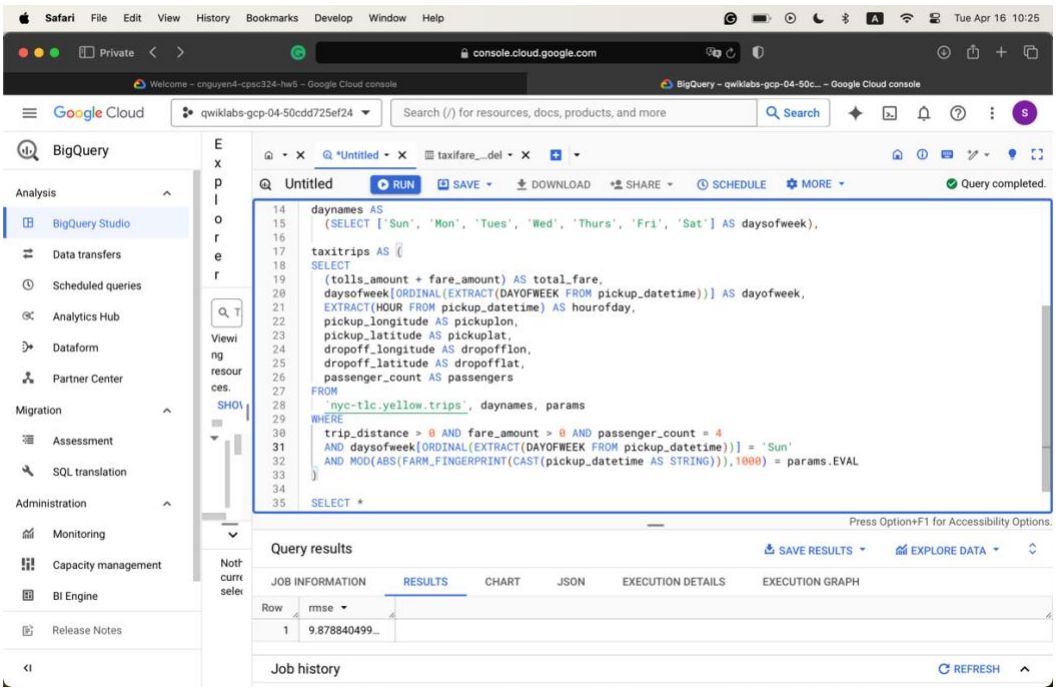
```
#standardSQL
SELECT
  SQRT(mean_squared_error) AS rmse
FROM
  ML.EVALUATE(MODEL taxi.taxifare_model,
  (

  WITH params AS (
```

```
    SELECT
    1 AS TRAIN,
    2 AS EVAL
    ),

  daynames AS
    (SELECT ['Sun', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat']
AS daysofweek),

  taxitrips AS (
  SELECT
    (tolls_amount + fare_amount) AS total_fare,
    daysofweek[ORDINAL(EXTRACT(DAYOFWEEK FROM pickup_datetime))]
AS dayofweek,
    EXTRACT(HOUR FROM pickup_datetime) AS hourofday,
    pickup_longitude AS pickuplon,
    pickup_latitude AS pickuplat,
    dropoff_longitude AS dropofflon,
    dropoff_latitude AS dropofflat,
    passenger_count AS passengers
  FROM
    `nyc-tlc.yellow.trips`, daynames, params
  WHERE
    trip_distance > 0 AND fare_amount > 0
    AND MOD(ABS(FARM_FINGERPRINT(CAST(pickup_datetime AS
STRING))),1000) = params.EVAL
  )

  SELECT *
  FROM taxitrips

  ))
```

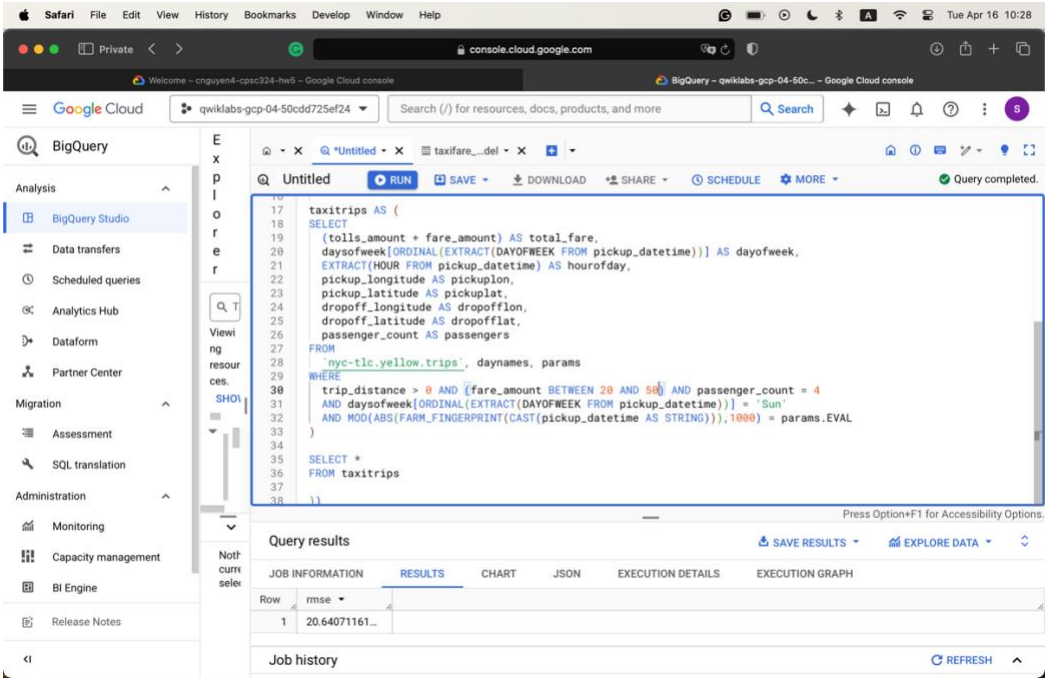
- Modifying the query
  - passengers_count = 4



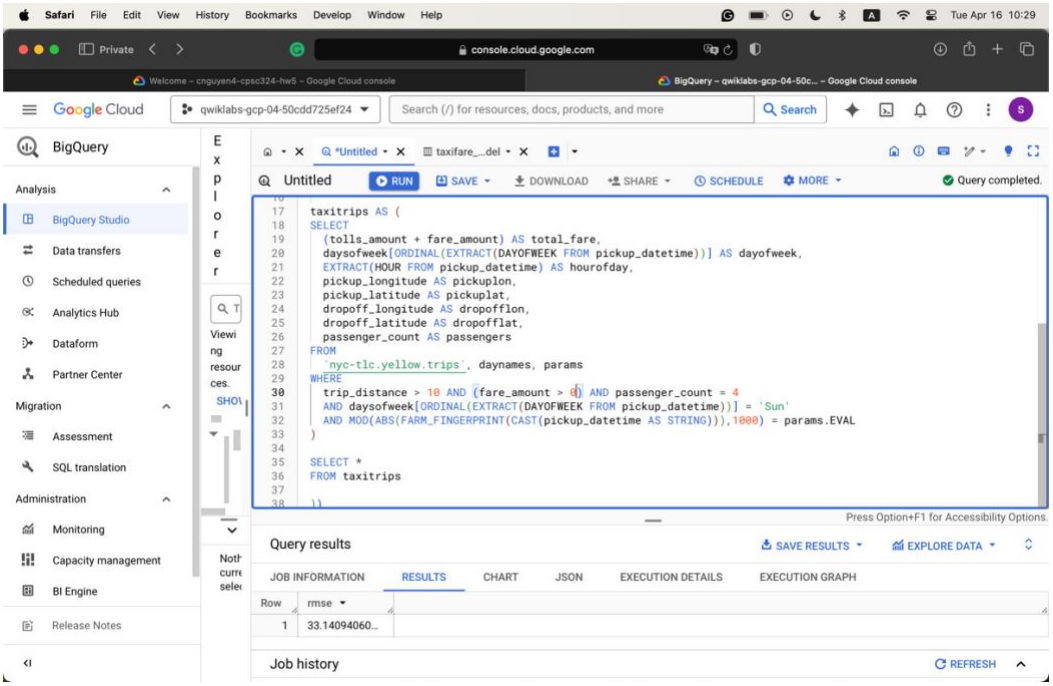
- daysofweek = ‘Sun’



- Fare_amount between 20 and 50



- Trip_distance > 10



- Prediction process

```
#standardSQL
SELECT
*
FROM
```

```
  ml.PREDICT(MODEL `taxi.taxifare_model`,
   (

 WITH params AS (
    SELECT
    1 AS TRAIN,
    2 AS EVAL
    ),

  daynames AS
    (SELECT ['Sun', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri',
'Sat'] AS daysofweek),

  taxitrips AS (
  SELECT
    (tolls_amount + fare_amount) AS total_fare,
    daysofweek[ORDINAL(EXTRACT(DAYOFWEEK FROM
pickup_datetime))] AS dayofweek,
    EXTRACT(HOUR FROM pickup_datetime) AS hourofday,
    pickup_longitude AS pickuplon,
    pickup_latitude AS pickuplat,
    dropoff_longitude AS dropofflon,
    dropoff_latitude AS dropofflat,
    passenger_count AS passengers
  FROM
    `nyc-tlc.yellow.trips`, daynames, params
  WHERE
    trip_distance > 0 AND fare_amount > 0
    AND MOD(ABS(FARM_FINGERPRINT(CAST(pickup_datetime AS
STRING))),1000) = params.EVAL
  )

  SELECT *
  FROM taxitrips

));
```

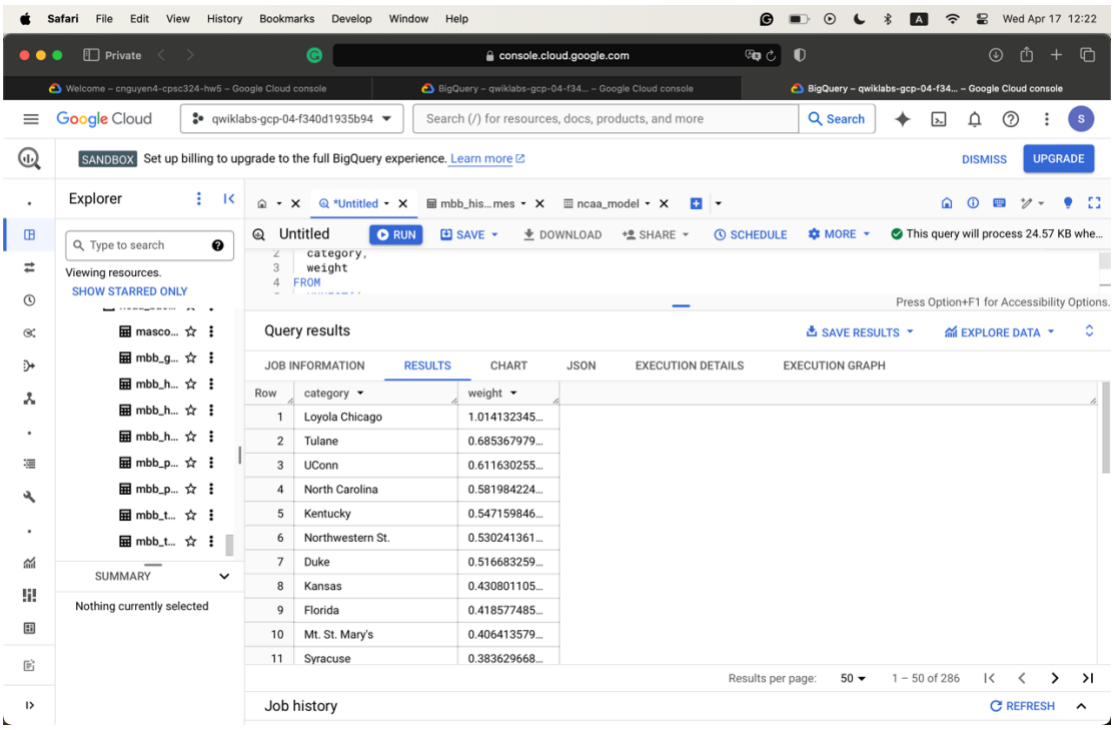
- The first model uses the raw coordinates, while the second uses the derived Euclidian distance from the coordinates. The second model also limits the fare_amount to be between 6 and 200.
    - Using the initial parameters as in the tutorial, the second model results in a lower RMSE score (5 and 9), which suggests that it is more accurate as interpreted by the RSME.

4. Question 4

- There are 33 seasons in the seasons. The data went back to 1985.
- The data used to build the model.
    - There are 2,117 instances and 27 attributes in the dataset.
    - The model only uses these attributes to train the data
        - Season
        - Win_seed
        - Win_seed_ncaa
        - Lose_seed
        - Lose_seed_ncaa
    - It covers data on NCAA men's games from 1985 to 2019.
- The training table generated from Task 7

Iteration	Training Data Loss	Evaluation Data Loss	Learn Rate	Duration (seconds)
3	0.5041	0.5784	1.6	2.68
2	0.5507	0.5929	0.8	2.65
1	0.6090	0.6298	0.4	2.49
0	0.6595	0.6671	0.2	2.39

    -
- The modified Task 7 query



- The results are the seed of each school (there are 286 of them). The lower the weight is, the more likely the team will win.
- Gonzaga's weight is 0.039

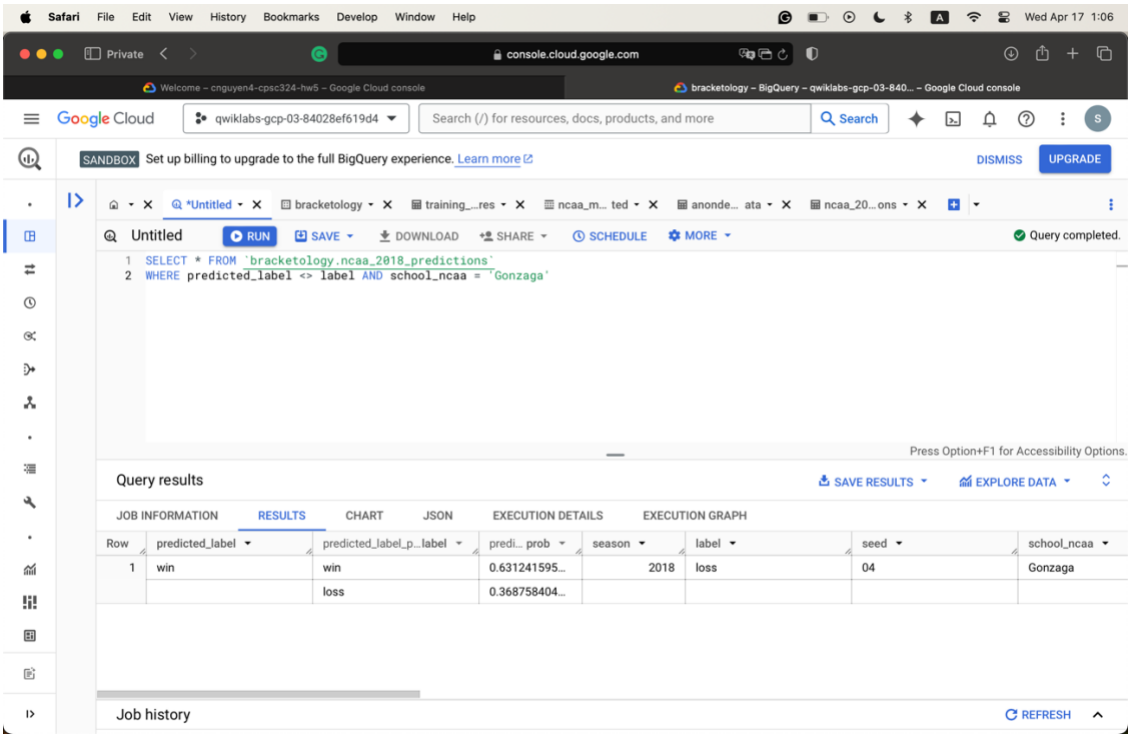
- The query result for Task 10

Row	predicted_label	predicted_label_p...label	predi... prob	season	round	days_from_epoch	game_date	day
1	win	win	0.631865055...	2018	16	17612	2018-03-22	Thursday
		loss	0.368134944...					
2	loss	win	0.466684445...	2018	32	17607	2018-03-17	Saturday
		loss	0.533315554...					

Row	day	label	seed	market	name	alias
1	Thursday	loss	04	Gonzaga	Bulldogs	GONZ
2	Saturday	win	04	Gonzaga	Bulldogs	GONZ

Row		name	alias	school_ncaa	points	opponent_seed	opponent_market
1		Bulldogs	GONZ	Gonzaga	60	09	Florida State
2		Bulldogs	GONZ	Gonzaga	90	05	Ohio State

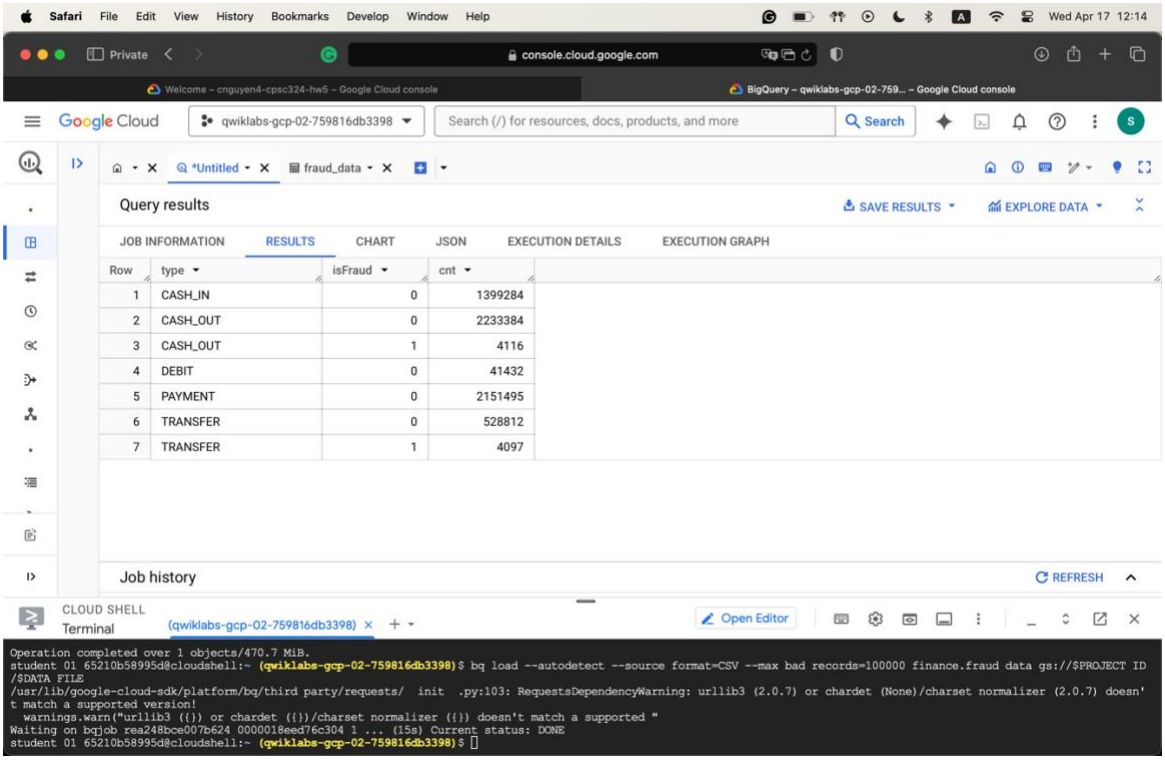
Row		opponent_seed	opponent_market	opponent_name	opponent_alias	opponent_school_ncaa	opponent_points
1	60	09	Florida State	Seminoles	FSU	Florida St.	75
2	90	05	Ohio State	Buckeyes	OSU	Ohio St.	84

- The number of rows resulting from the Task 12 query is 664.
- The number of rows used in the Task 15 training model is 438.
- Modify Task 19 query to include Gonzaga
    - 

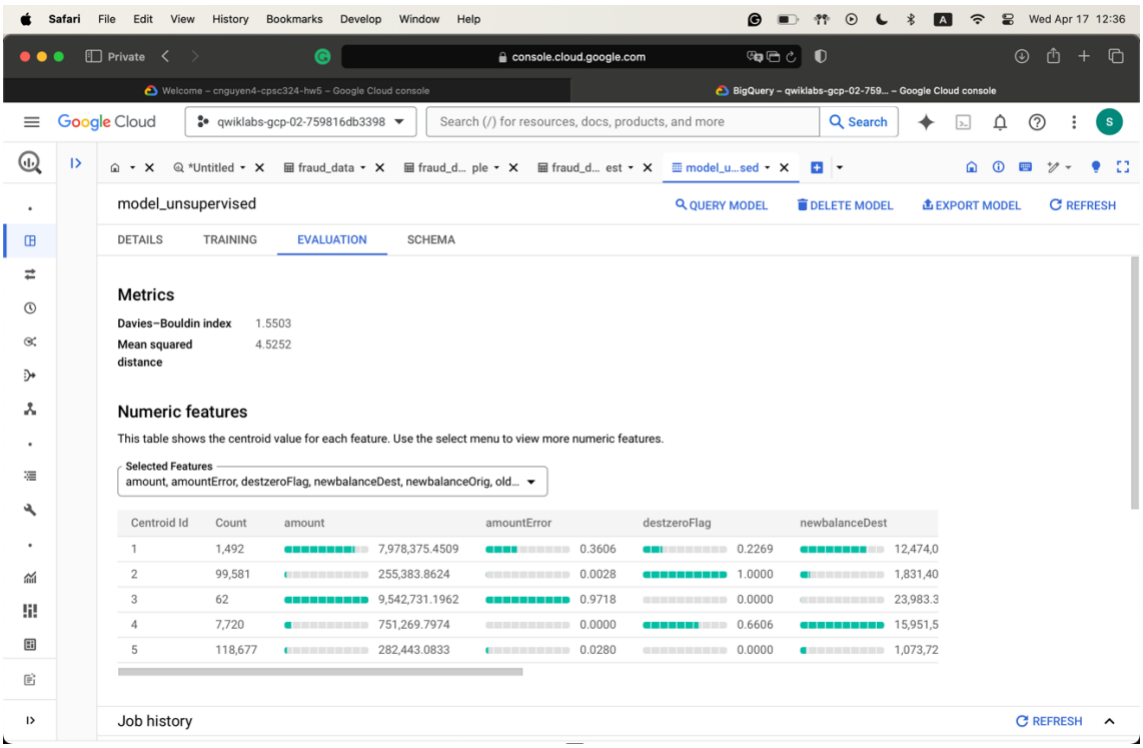
- Task 21 query
    - The query may seem a bit intimidating at first, but it is, in fact, quite easy to understand as it is straightforward. Code comments are used liberally to denote their functions.
    - The advantage is that everything is laid out. We can control what is returned and so on.
    - The disadvantage: it makes the function complicated and hard to maintain. I can imagine users can easily fall through the logic why trying to understand it.
- 2019 prediction results
    - The model returns some pretty accurate results

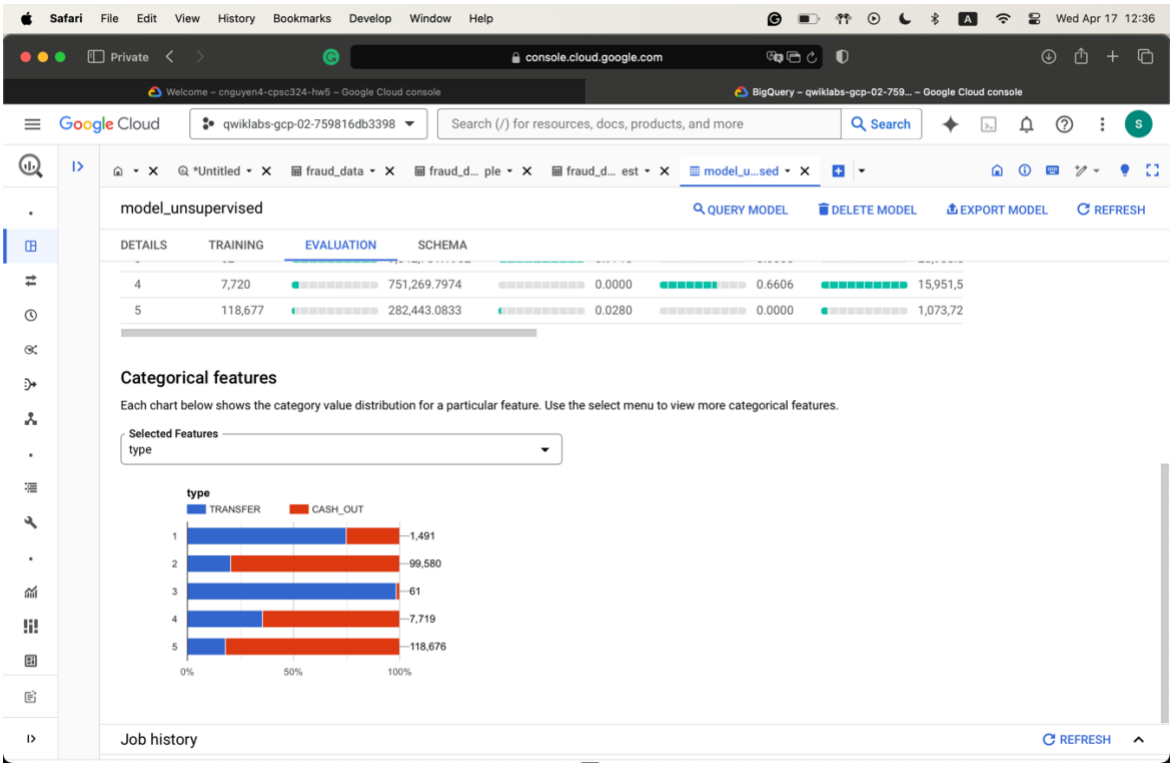
- The game that Gonzaga lost to Texas Tech was correctly predicted, though the confidence rate is not high (51% chance of losing).
- For Gonzaga v. Florida State, Gonzaga v. Baylor, and Gonzaga v. Fairleigh Dickinson, the model correctly predicted that Gonzaga won.

5. Question 5

- Result from the first query of Task 4
  - 

- Pause and reflect on Task 5
  - If there is no labeled fraud event in the data, I will probably use unsupervised learning to try to detect any abnormalities in the data. For skimming through, I can see that for CASH transactions, the amount is pretty big. I will also use association rules to try to find something that can be linked together.
- Metrics available for Task 6





- Pause and reflect on Task 6

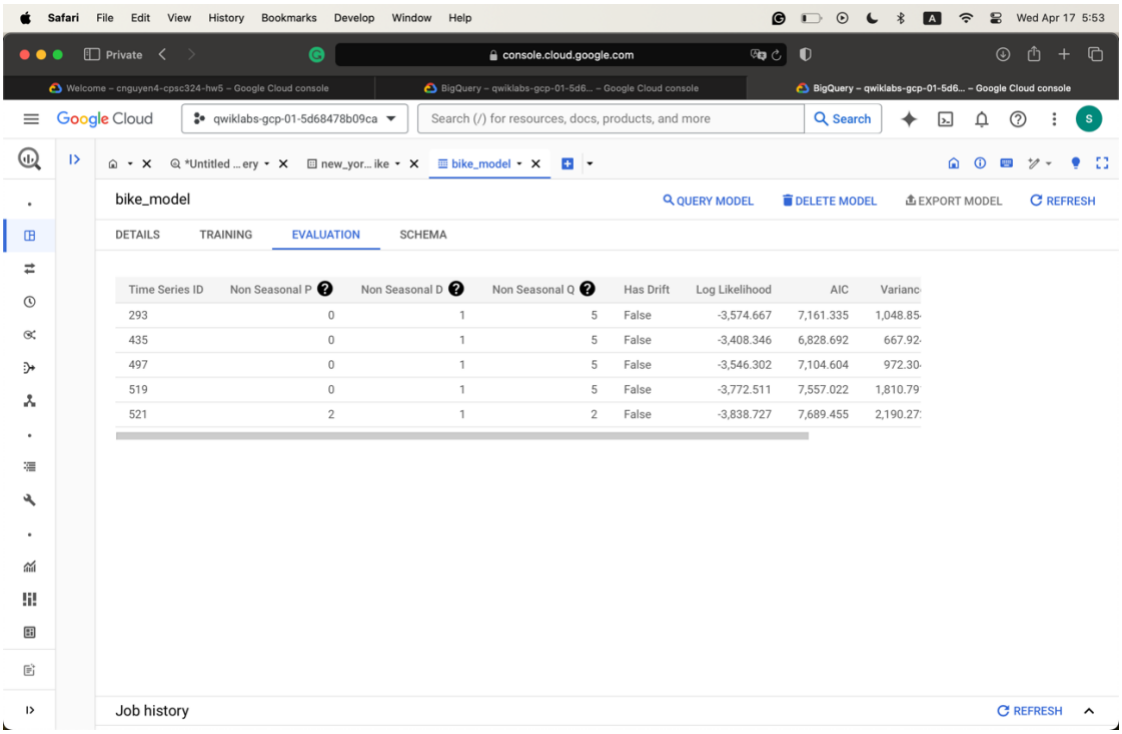
- The cluster that I think is the most interesting one is cluster 3. It is relatively small but has a very high error rate (near 100%).
- I think this is due to the fact that the instances in that cluster are really similar to each other, yet they are relatively different from the other ones.

- Pause and Reflect on Task 7
  - oldBalanceOrig and type are the two most important ones.
  - This is consistent with my response to Pause and Reflect of Task 5.
- Pause and Reflect on Task 9
  - There is a significant difference between the two models:
    -

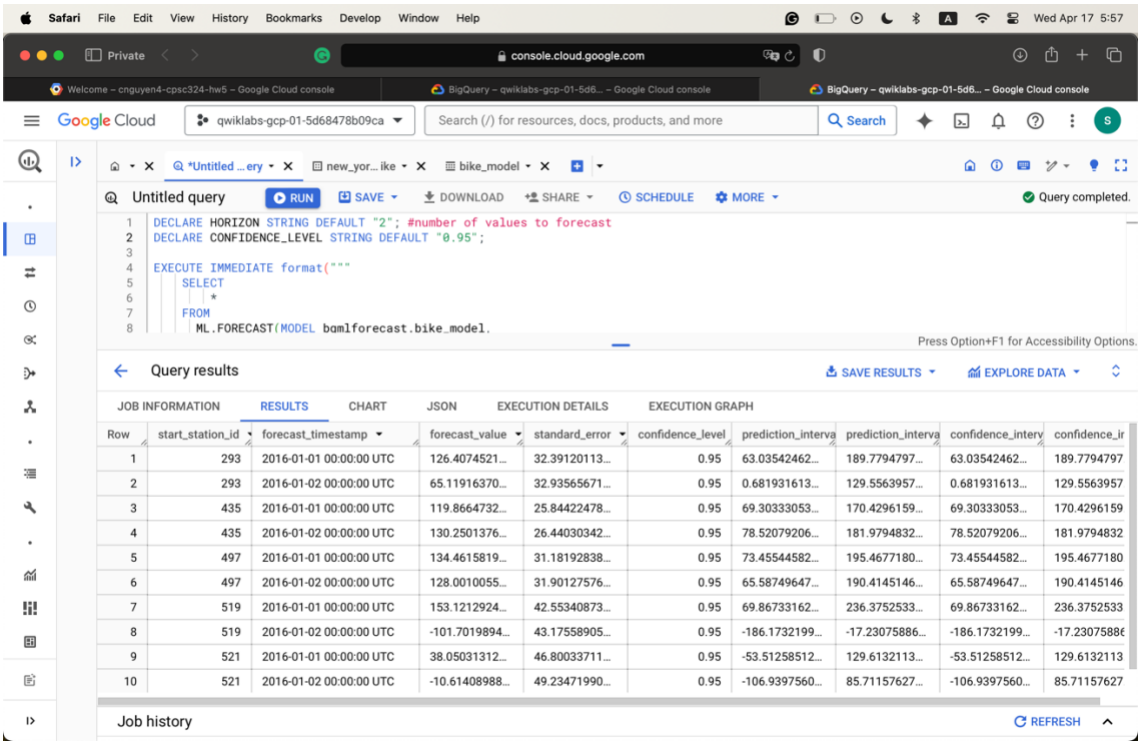
Row	model_name	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	Initial_reg	0.971905...	0.387878...	0.981919...	0.554472...	0.068082...	0.930771...
2	improved_reg	0.968573...	0.859242...	0.995108...	0.910638...	0.013089...	0.998735...

  - The improved model, which is a decision tree, yields a much better result than the initial classification one, as seen in the f1-score.
  - I noticed that the fields used are similar, and the only thing that is different is their model type.
  - I was actually surprised by the difference in the results.

6. Question 6
   - The evaluation table



- Result and query from Task 5

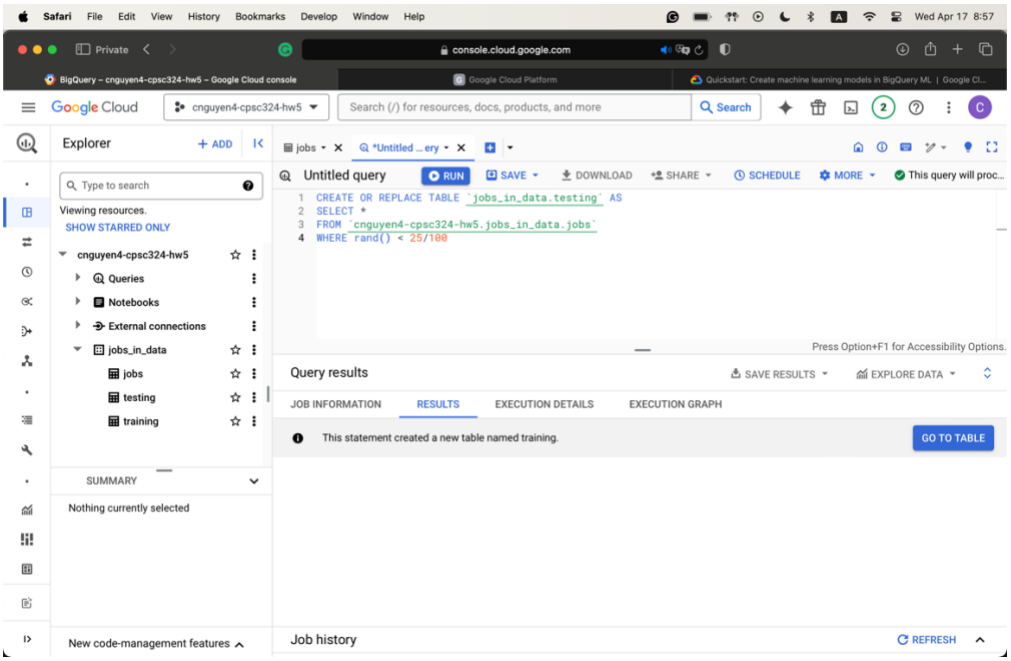



```
DECLARE HORIZON STRING DEFAULT "2"; #number of values to forecast
 DECLARE CONFIDENCE_LEVEL STRING DEFAULT "0.95";
```

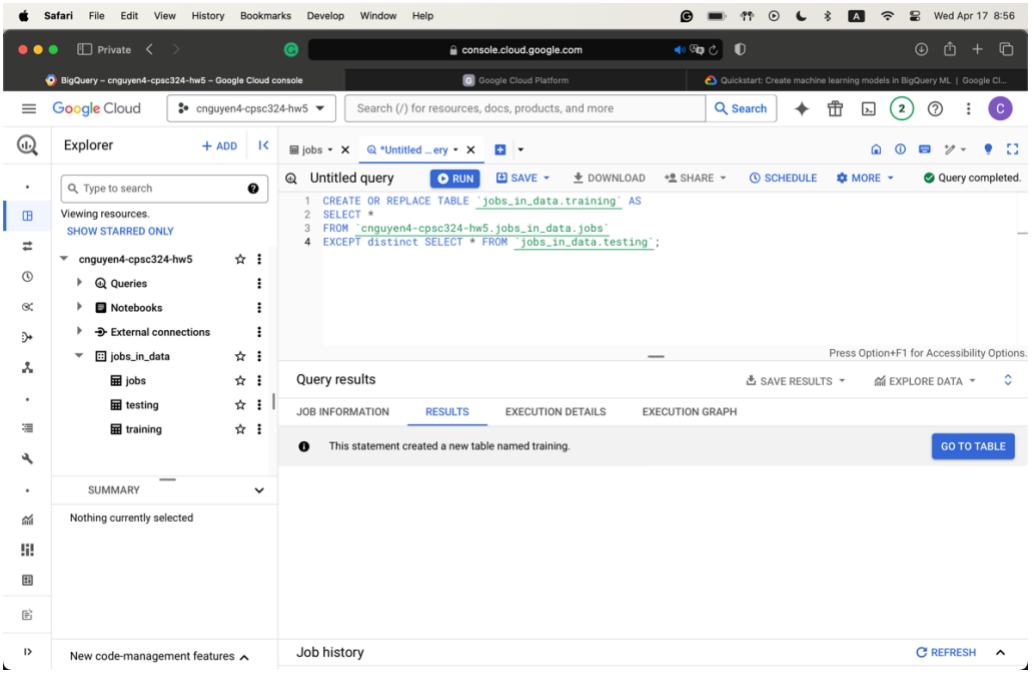
```
EXECUTE IMMEDIATE format("""
    SELECT
        *
    FROM
      ML.FORECAST(MODEL bqmlforecast.bike_model,
                  STRUCT(%s AS horizon,
                         %s AS confidence_level)
                 )
    """, HORIZON, CONFIDENCE_LEVEL)
```

7. Question 7

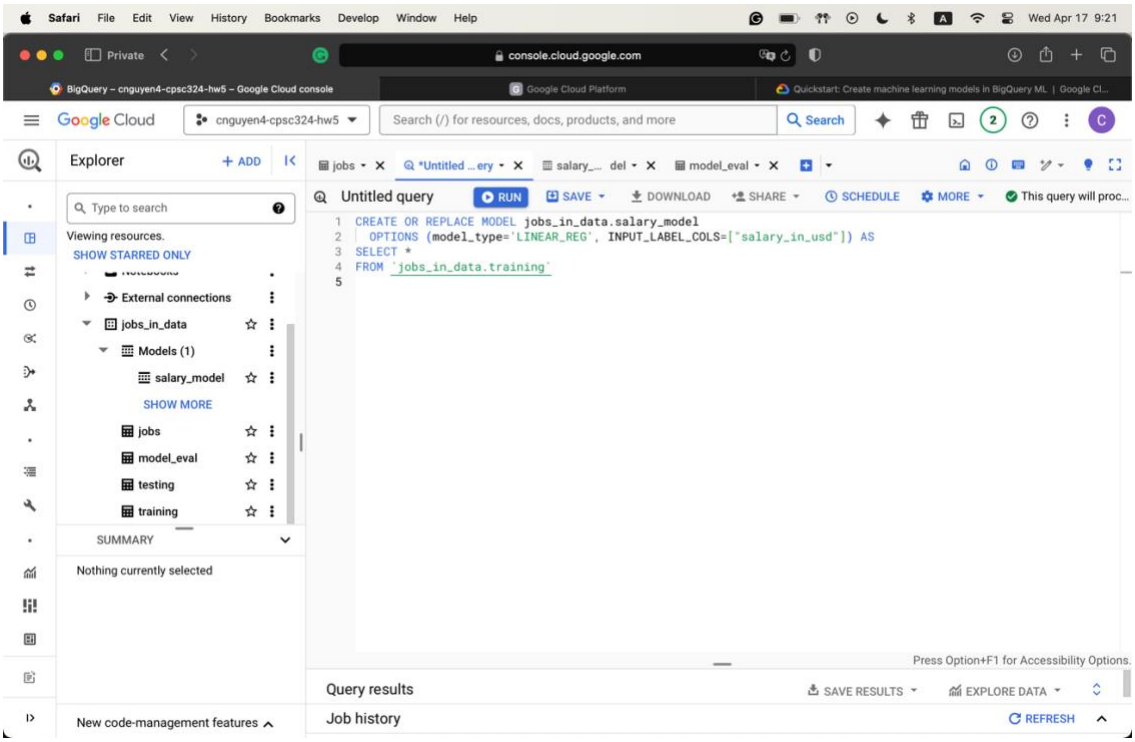
- The dataset I am using is the jobs_in_data.csv dataset from the previous homework.
    - There are 9,355 rows and 12 attributes.
- The label here is the yearly salary based on the position and other factor. This model should be able to suggest a recommended yearly salary so that it reflects the average salary in the market.
- I plan to split them up into training and testing sets using the handout method. I will keep 25% of the dataset as the testing set, and the remaining one will be the training set.
    - To create the testing set



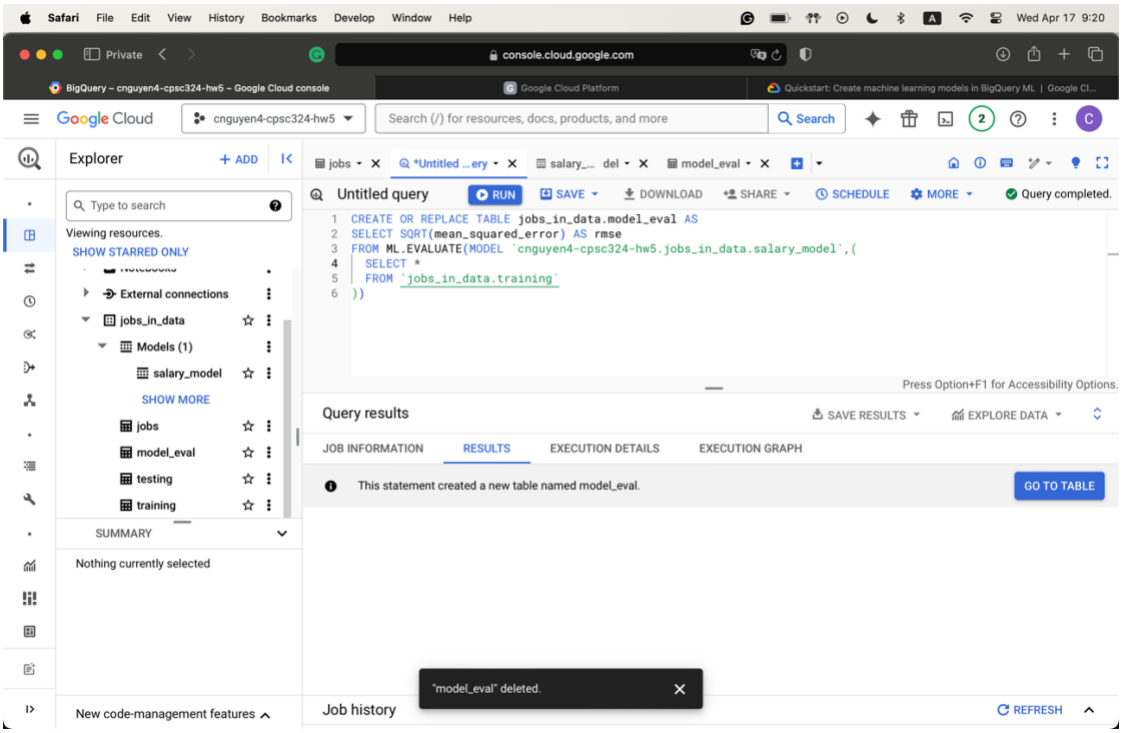
- To create the training set

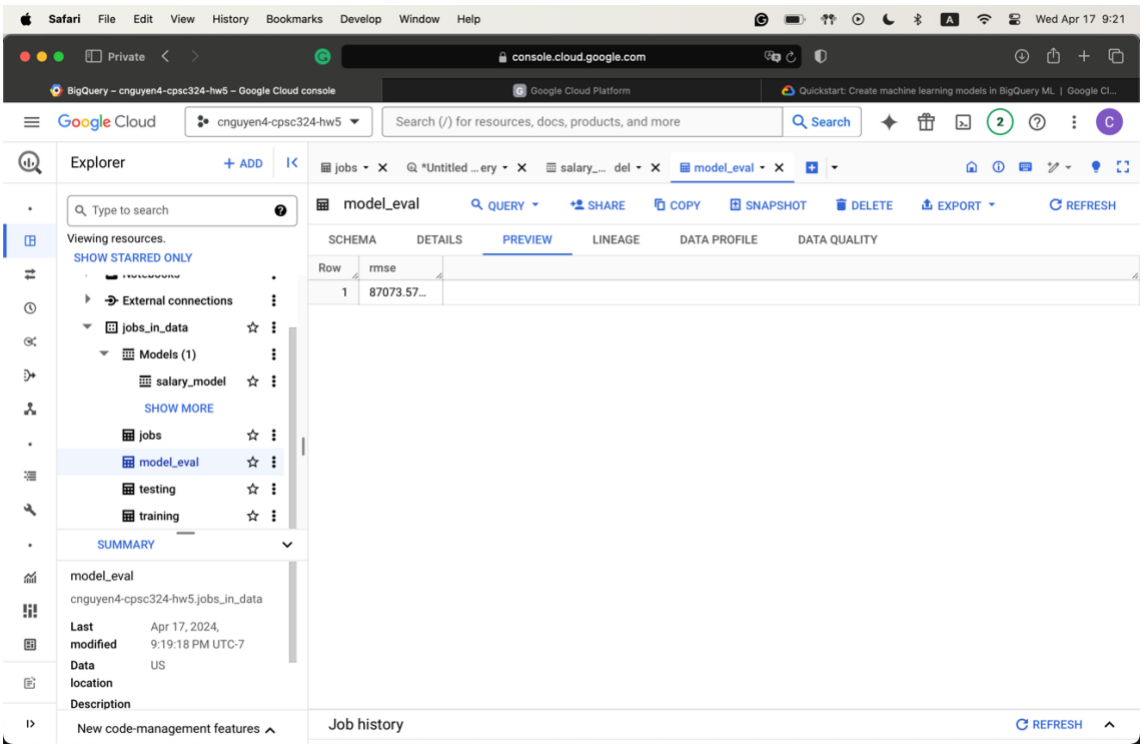


- To train the model



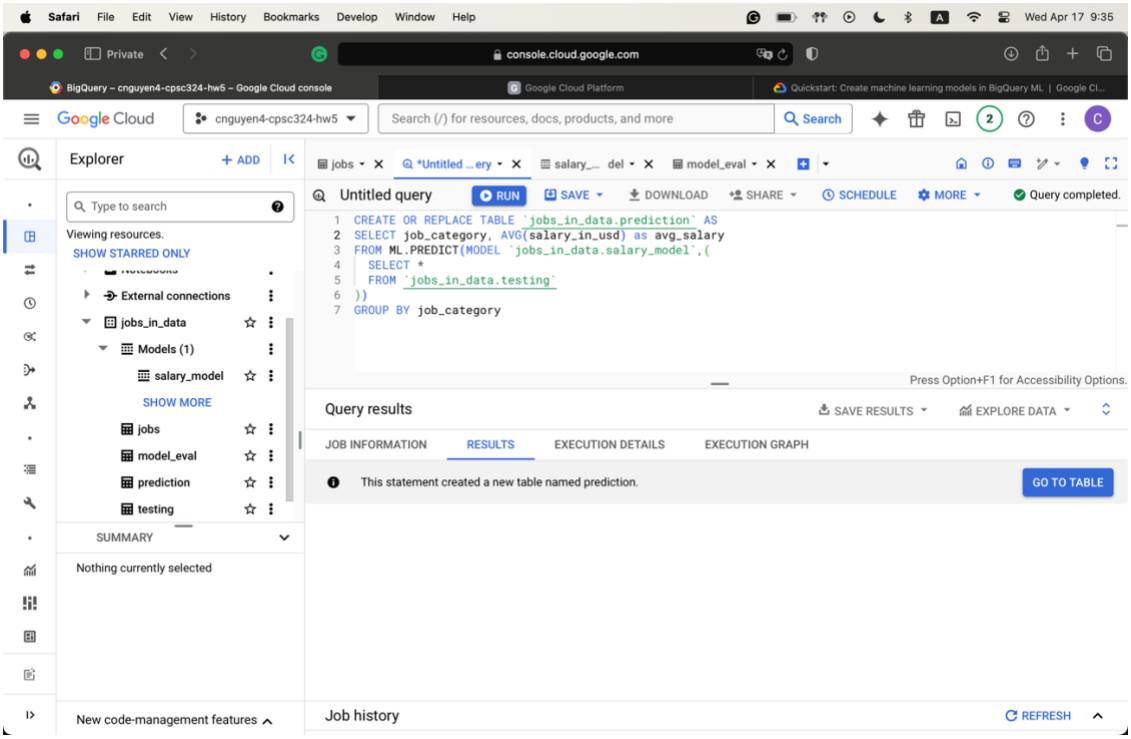
- To evaluate the model

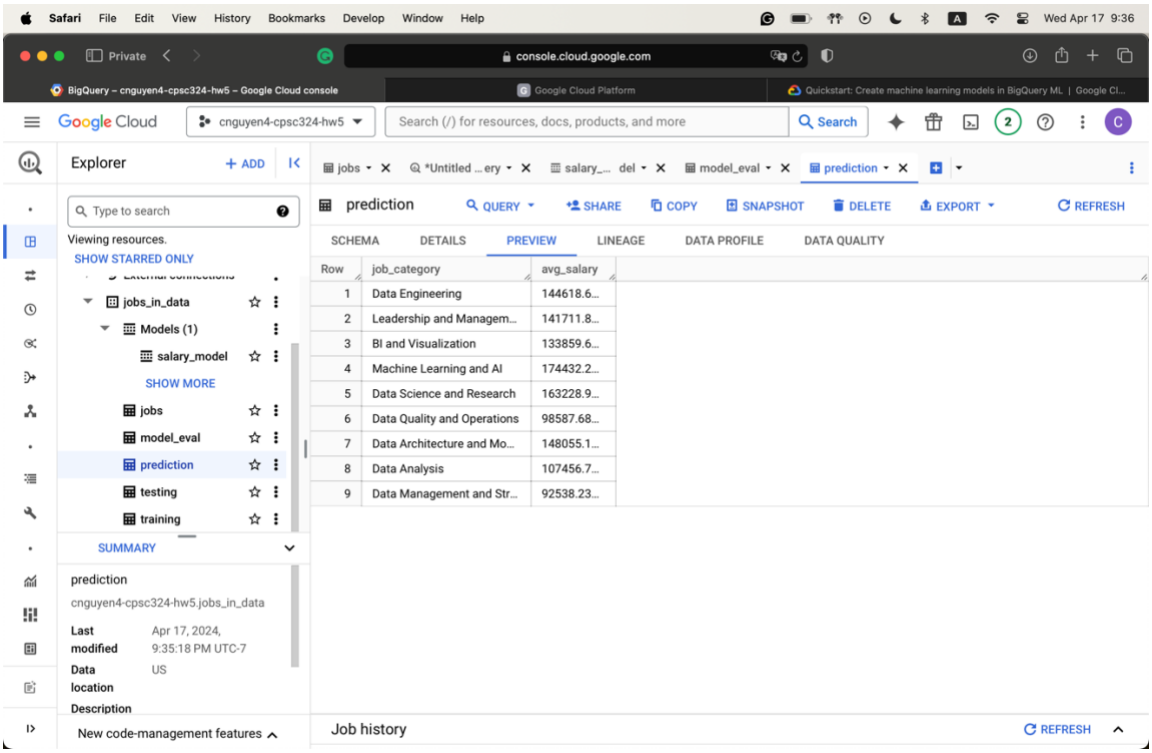




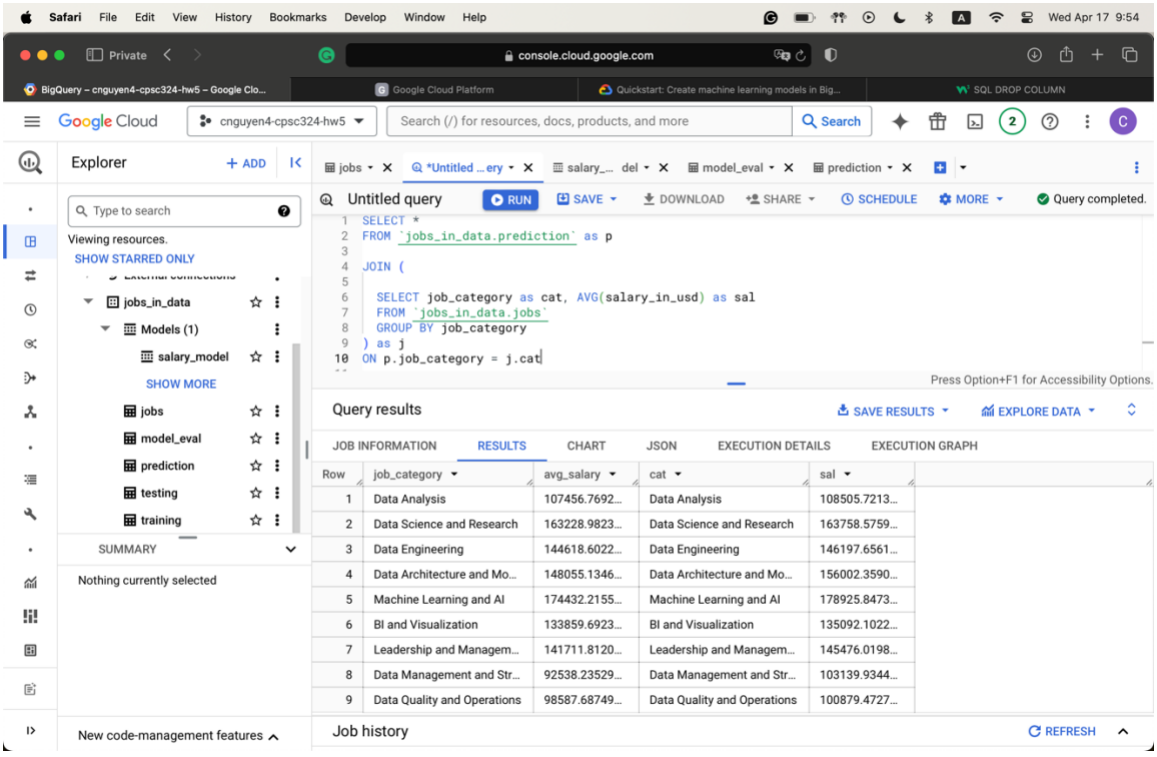
- It seems like the performance of this model is not that good.

- Prediction

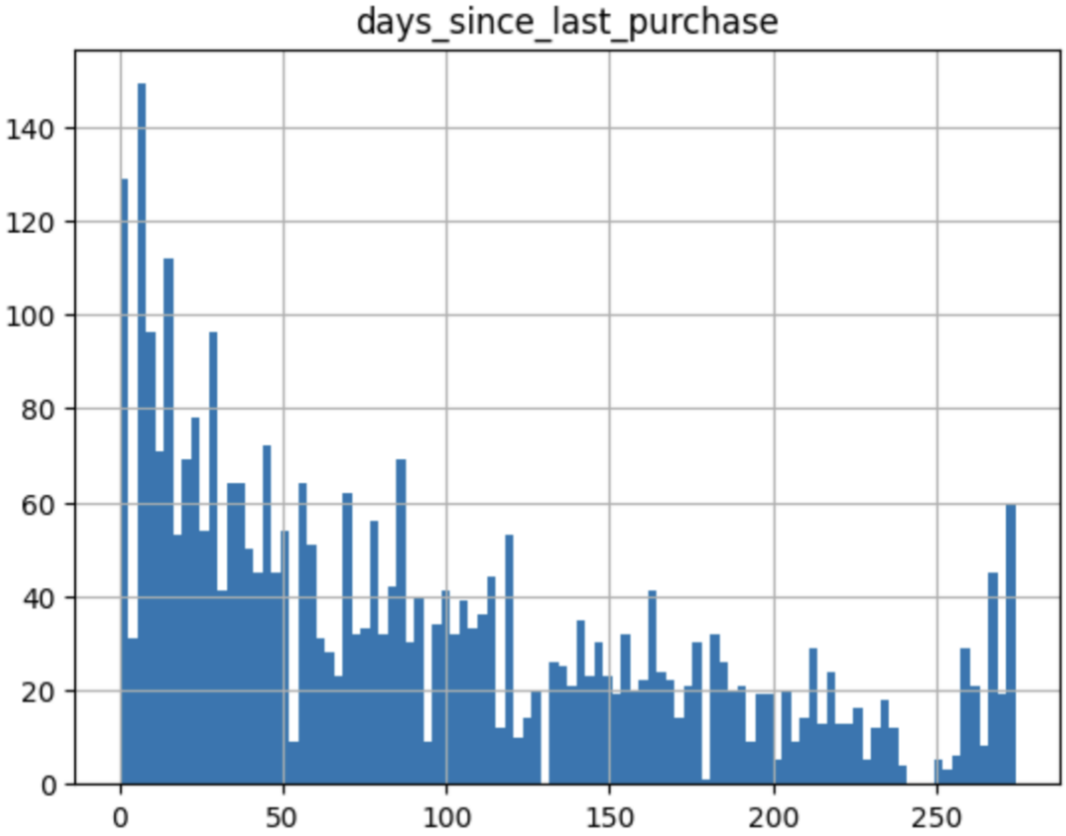


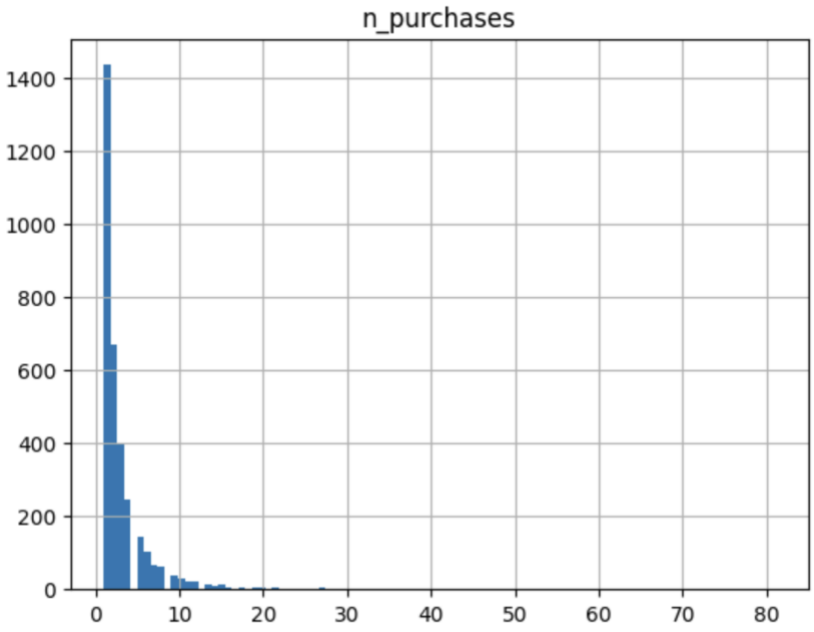


- The predicted result is pretty close to the actual average

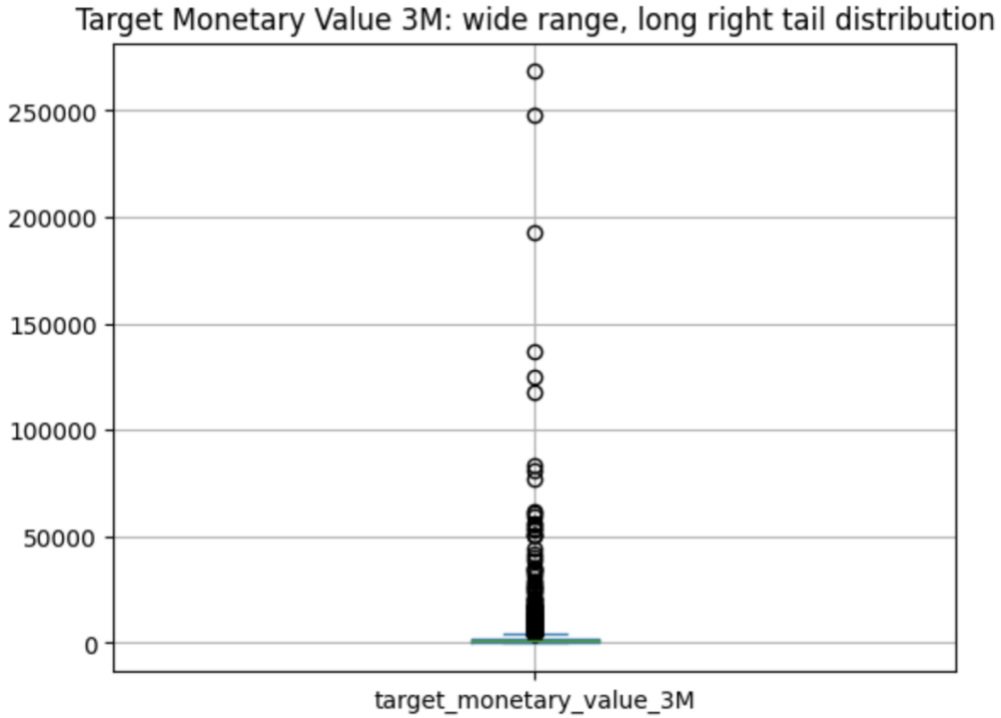


8. Question 8

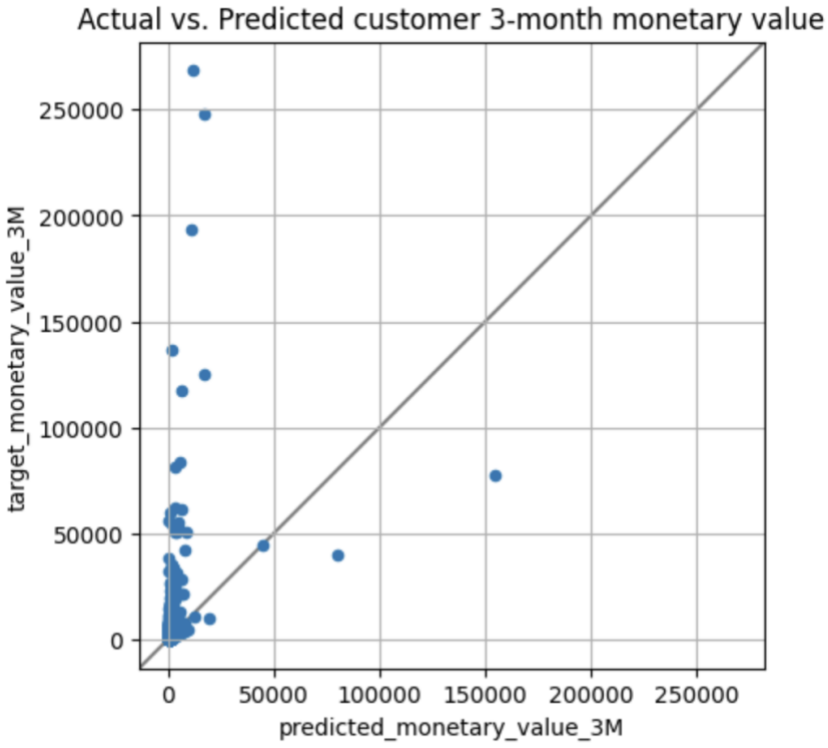
- Cell 11 histogram
  - 

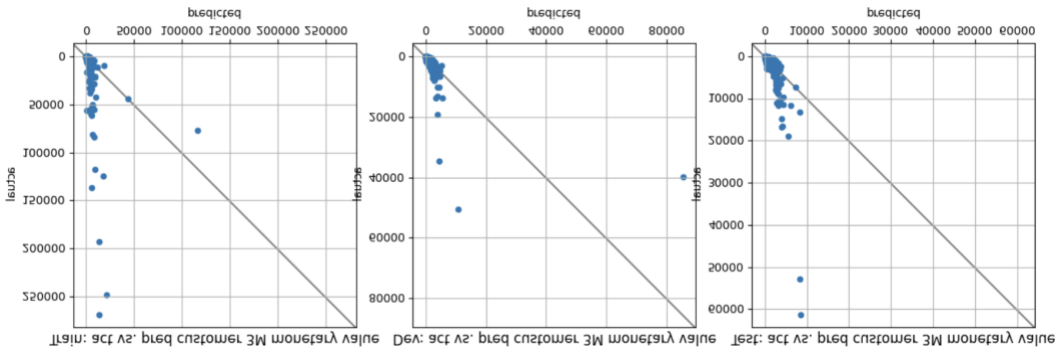
- Cell 14 histogram
  - 

- Cell 17 plot



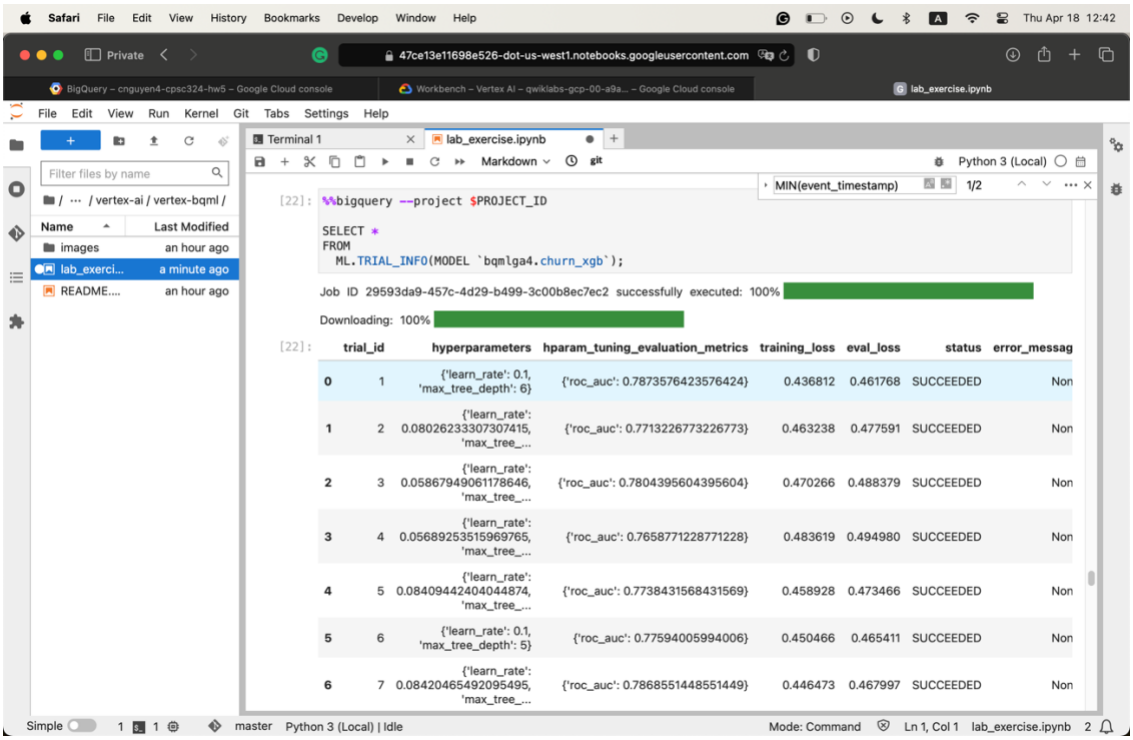
- Cell 21 plot



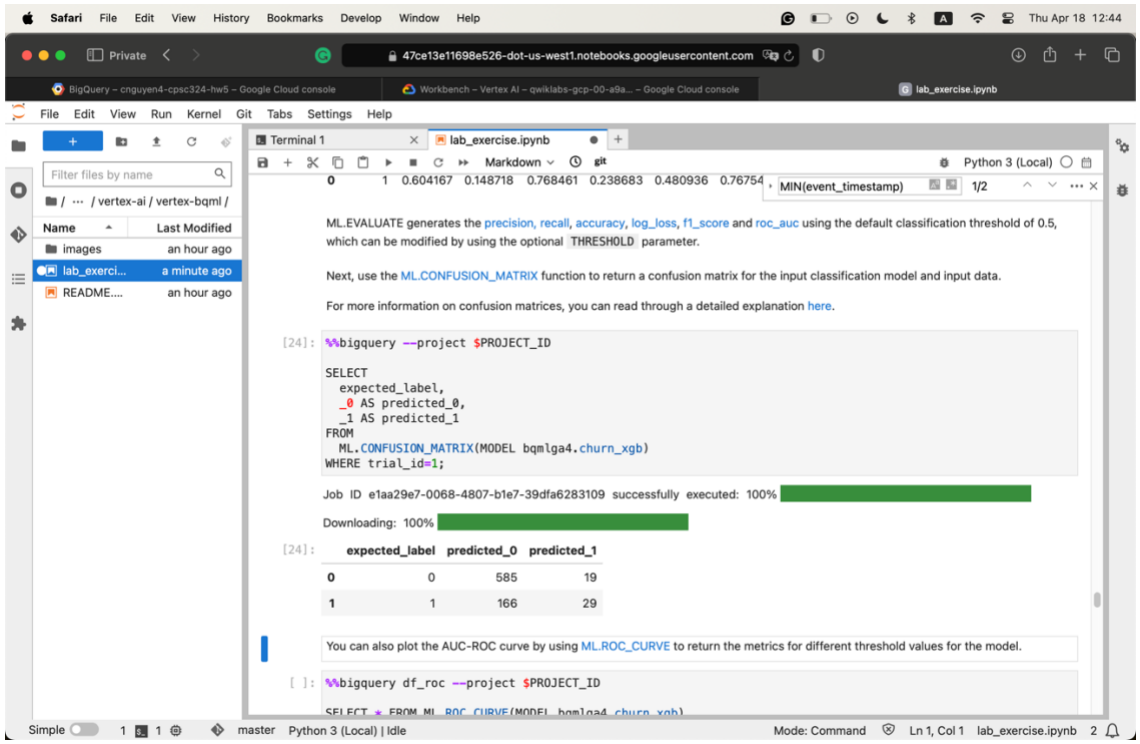
- Cell 23 plot
  - 

9. Question 9
   - Describe queries
     - Cell 10
       - Return all rows
     - Cell 11
       - Count the total number of events and the total number of unique user
     - Cell 12
       - This query returns all users that are considered bounced or churned
     - Cell 13
       - Returns a matrix of the total count of whether someone bounced, churned, or both.
     - Cell 14
       - Returns the percentage of users who have churned
     - Cell 15
       - This query returns the metadata of each user event. It shows which country they accessed it from, the operating system, and language and region.

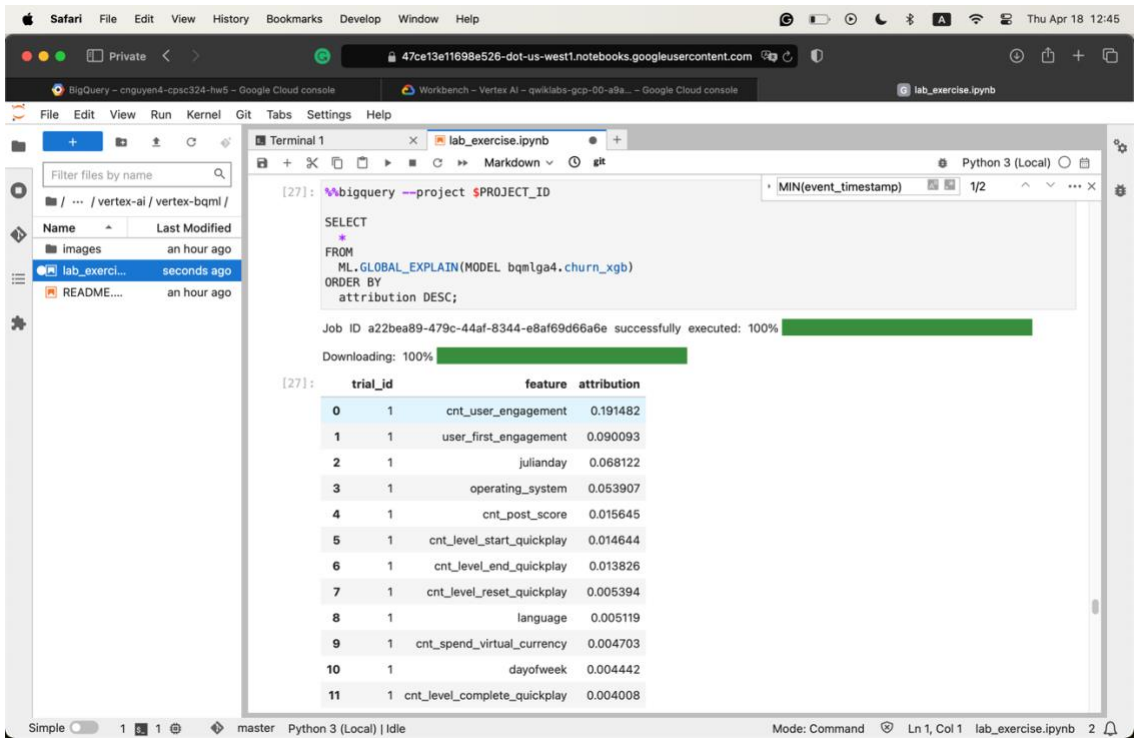
- Cell 16
  - This query counts the total number of events that occurred.
- Cell 17
  - This query counts the total number of events per user.
- Cell 18
  - This query combines these three intermediate views into one table. It also returns if an instance is used in the training and testing set.

- Cell 22
  - 

- Cell 23

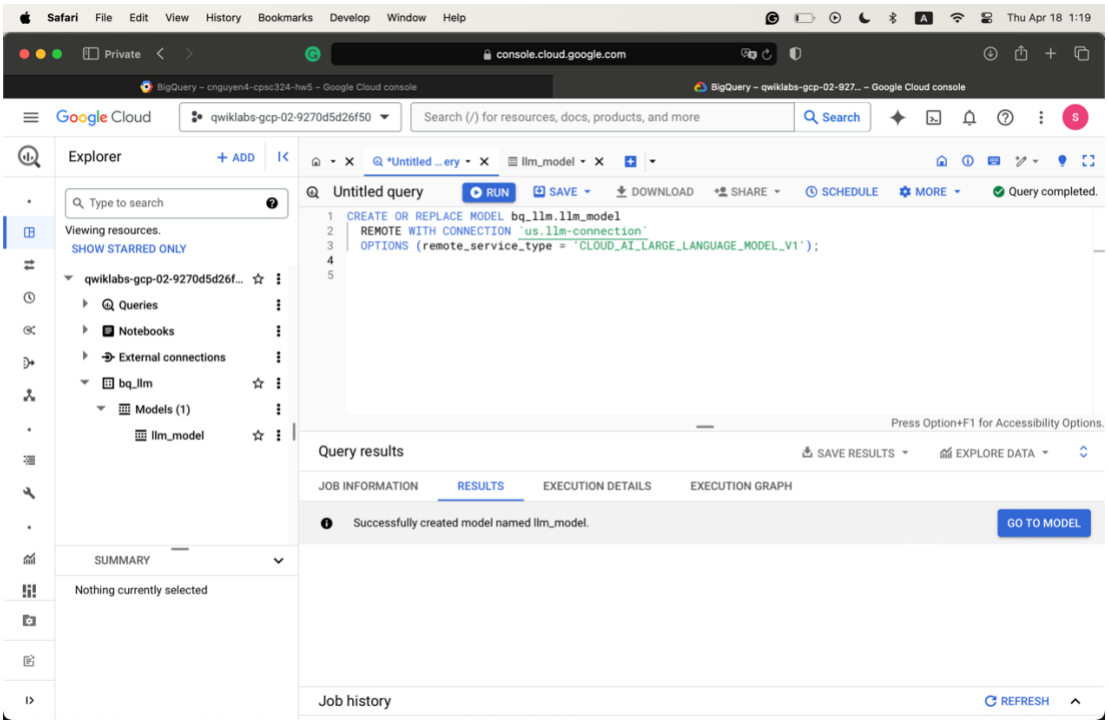
- 

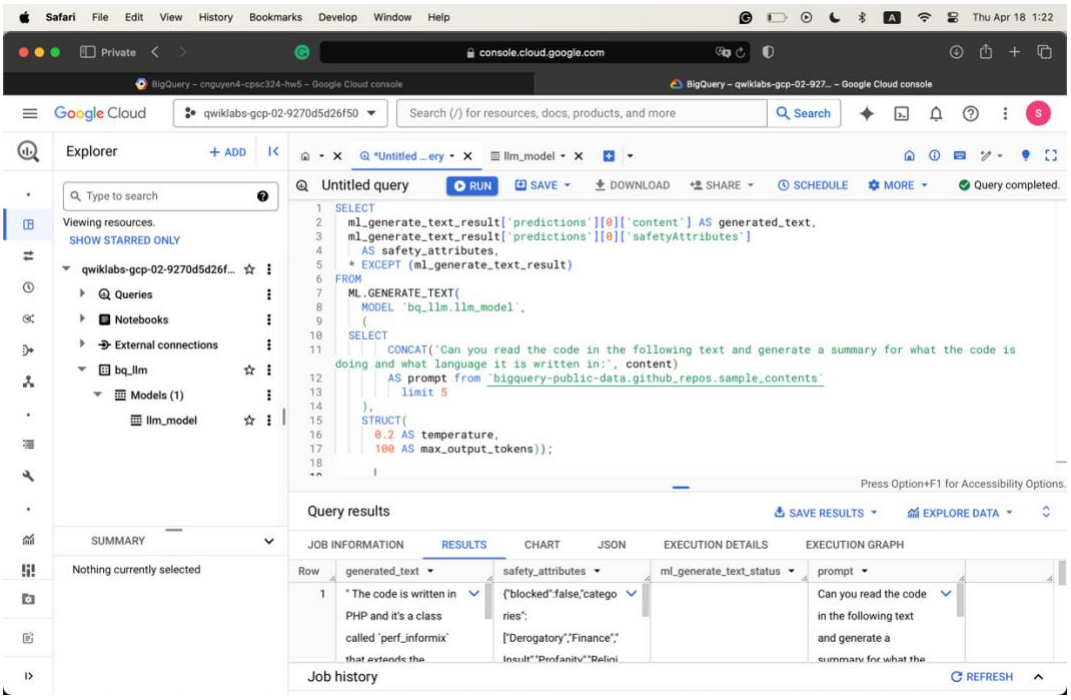
- Cell 26

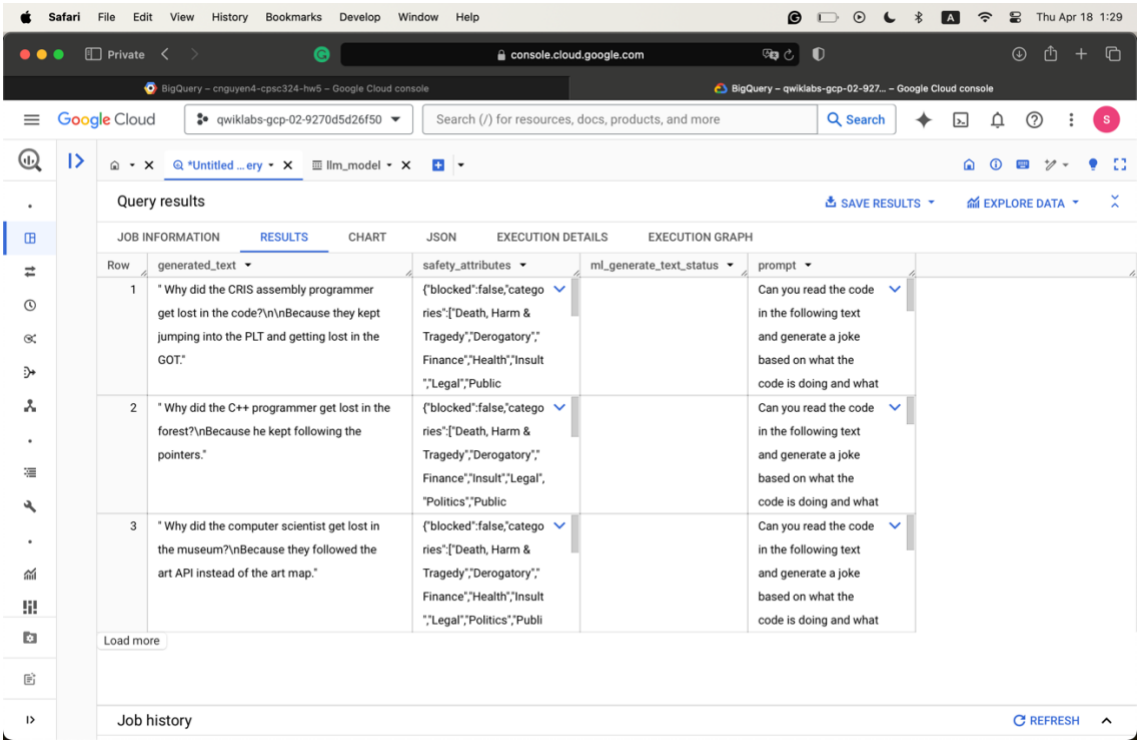
- 

  - The table above shows how much weight the model puts into each feature that is used to predict the result.

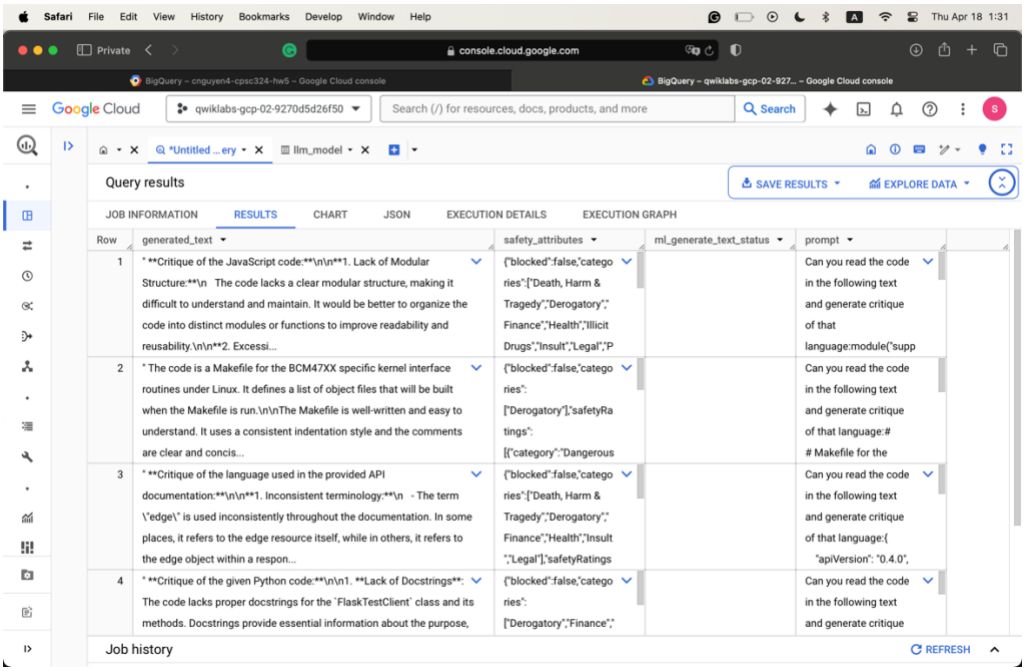
10. Question 10

- Create a model
  - 

- Generate text using the model.
  - 

- The generated_text column is the result returned by the model after passing in the prompt, as well as the data that we want to use. It returns the language in which the code was written, as well as a summary of what is going on inside.
- I tried the joke prompt, and it returned the following:
  - 

    - This is not particularly very funny, though.
- I tried another one: Can you read the code in the following text and generate a critique of that language?
  - Some examples:


Safari File Edit View History Bookmarks Develop Window Help
Thu Apr 18 1:31
Private
console.cloud.google.com
BigQuery – cnguyen4-cpsc324-hw5 – Google Cloud console
BigQuery – qwiklabs-gcp-02-927... – Google Cloud console
Google Cloud
qwiklabs-gcp-02-9270d5d26f50
Search (/) for resources, docs, products, and more
Search
*Untitled ...ery
llm_model
Query results
SAVE RESULTS
EXPLORE DATA
JOB INFORMATION
RESULTS
CHART
JSON
EXECUTION DETAILS
EXECUTION GRAPH
Row
generated_text
safety_attributes
ml_generate_text_status
prompt
1
" **Critique of the JavaScript code:**\n\n**1. Lack of Modular Structure:**\n The code lacks a clear modular structure, making it difficult to understand and maintain. It would be better to organize the code into distinct modules or functions to improve readability and reusability.\n\n**2. Excessi...
{"blocked":false,"categories":["Death, Harm & Tragedy","Derogatory","Finance","Health","Illicit Drugs","Insult","Legal","P
Can you read the code in the following text and generate critique of that language:module("supp
2
" The code is a Makefile for the BCM47XX specific kernel interface routines under Linux. It defines a list of object files that will be built when the Makefile is run.\n\nThe Makefile is well-written and easy to understand. It uses a consistent indentation style and the comments are clear and concis...
{"blocked":false,"categories":["Derogatory"],"safetyRatings":[{"category":"Dangerous
Can you read the code in the following text and generate critique of that language:# # Makefile for the
3
" **Critique of the language used in the provided API documentation:**\n\n**1. Inconsistent terminology:**\n - The term \"edge\" is used inconsistently throughout the documentation. In some places, it refers to the edge resource itself, while in others, it refers to the edge object within a respon...
{"blocked":false,"categories":["Death, Harm & Tragedy","Derogatory","Finance","Health","Insult","Legal"],"safetyRatings
Can you read the code in the following text and generate critique of that language:{ "apiVersion": "0.4.0",
4
" **Critique of the given Python code:**\n\n1. **Lack of Docstrings**: The code lacks proper docstrings for the `FlaskTestClient` class and its methods. Docstrings provide essential information about the purpose,
{"blocked":false,"categories":["Derogatory","Finance","
Can you read the code in the following text and generate critique
Job history
REFRESH