Predict Taxi Fare with a BigQuery ML Forecasting Model

**Open the BigQuery console**

1. In the Google Cloud Console, select **Navigation menu** > **BigQuery**.

The **Welcome to BigQuery in the Cloud Console** message box opens. This message box provides a link to the quickstart guide and the release notes.

1. Click **Done**.

The BigQuery console opens.

**Explore NYC taxi cab data**

**Question:** How many trips did Yellow taxis take each month in 2015?

Copy and paste the following SQL code into the query **EDITOR**:

#standardSQL

SELECT

TIMESTAMP\_TRUNC(pickup\_datetime,

MONTH) month,

COUNT(\*) trips

FROM

`bigquery-public-data.new\_york.tlc\_yellow\_trips\_2015`

GROUP BY

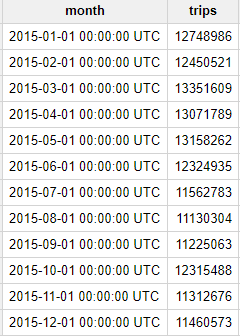
1

ORDER BY

1

Then click **Run**.

You should receive the following result:



As we see, every month in 2015 had over 10 million NYC taxi trips—no small amount!

Replace the previous query with the following and then click **Run**:

#standardSQL

SELECT

EXTRACT(HOUR

FROM

pickup\_datetime) hour,

ROUND(AVG(trip\_distance / TIMESTAMP\_DIFF(dropoff\_datetime,

pickup\_datetime,

SECOND))\*3600, 1) speed

FROM

`bigquery-public-data.new\_york.tlc\_yellow\_trips\_2015`

WHERE

trip\_distance > 0

AND fare\_amount/trip\_distance BETWEEN 2

AND 10

AND dropoff\_datetime > pickup\_datetime

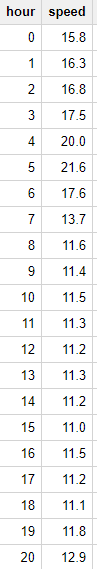
GROUP BY

1

ORDER BY

1

You should receive the following result:



During the day, the average speed is around 11-12 MPH; but at 5:00 AM the average speed almost doubles to 21 MPH. Intuitively this makes sense since there is likely less traffic on the road at 5:00 AM.

**Identify an objective**

You will now create a machine learning model in BigQuery to predict the price of a cab ride in New York City given the historical dataset of trips and trip data. Predicting the fare before the ride could be very useful for trip planning for both the rider and the taxi agency.

**Select features and create your training dataset**

The New York City Yellow Cab dataset is a [public dataset](https://cloud.google.com/bigquery/public-data/nyc-tlc-trips) provided by the city and has been loaded into BigQuery for your exploration. Browse the complete list of fields [here](https://console.cloud.google.com/bigquery?p=bigquery-public-data&d=new_york_taxi_trips&page=dataset) and then [preview the dataset](https://console.cloud.google.com/bigquery?p=bigquery-public-data&d=new_york_taxi_trips&page=dataset) to find useful features that will help a machine learning model understand the relationship between data about historical cab rides and the price of the fare.

Your team decides to test whether these below fields are good inputs to your fare forecasting model:

* Tolls Amount
* Fare Amount
* Hour of Day
* Pick up address
* Drop off address
* Number of passengers

Replace the query with the following:

#standardSQL

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 \*

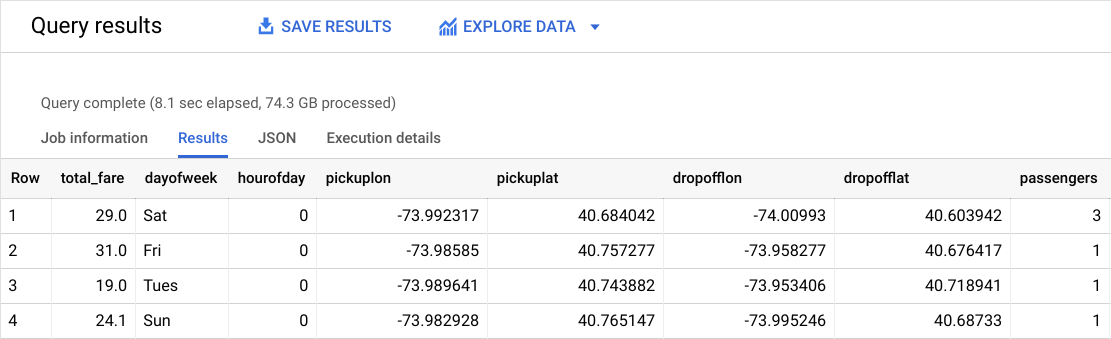
FROM taxitrips

Note a few things about the query:

1. The main part of the query is at the bottom (SELECT \* from taxitrips).
2. taxitrips does the bulk of the extraction for the NYC dataset, with the SELECT containing your training features and label.
3. The WHERE removes data that you don't want to train on.
4. The WHERE also includes a sampling clause to pick up only 1/1000th of the data.
5. Define a variable called TRAIN so that you can quickly build an independent EVAL set.

Now that you have a better understanding of this query's purpose, click **Run**.

You should receive a similar result:

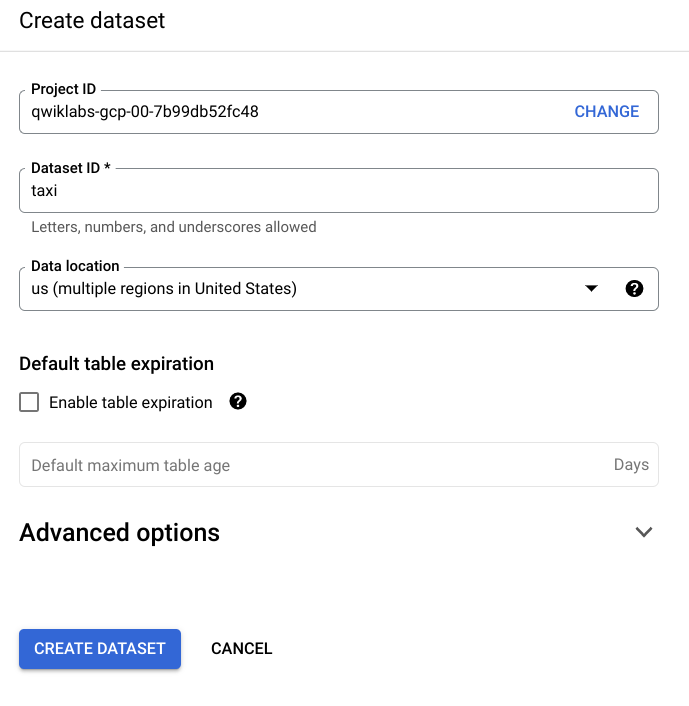


**Create a BigQuery dataset to store models**

In this section, you will create a new BigQuery dataset which will store your ML models.

1. In the left-hand *Explorer* panel, click on the **View actions** icon next to your Project ID and then click **Create dataset**.
2. In the Create Dataset dialog, enter in the following:

* For **Dataset ID**, type **taxi**.
* Select **us(multiple regions in United States)** as the **Data location**
* Leave the other values at their defaults.



1. Then click **Create dataset**.

**Select a BQML model type and specify options**

Now that you have your initial features selected, you are now ready to create your first ML model in BigQuery.

There are several model types to choose from:

* **Forecasting** numeric values like next month's sales with Linear Regression (linear\_reg).
* Binary or Multiclass **Classification** like spam or not spam email by using Logistic Regression (logistic\_reg).
* k-Means **Clustering** for when you want unsupervised learning for exploration (kmeans).

**Note:** There are many additional model types used in Machine Learning (like Neural Networks and decision trees) and available using libraries like [TensorFlow](https://www.tensorflow.org/tutorials/). At this time, BQML supports the three listed above. Follow the [BQML roadmap](https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-syntax-create) for more information.

Enter the following query to create a model and specify model options.

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 \*

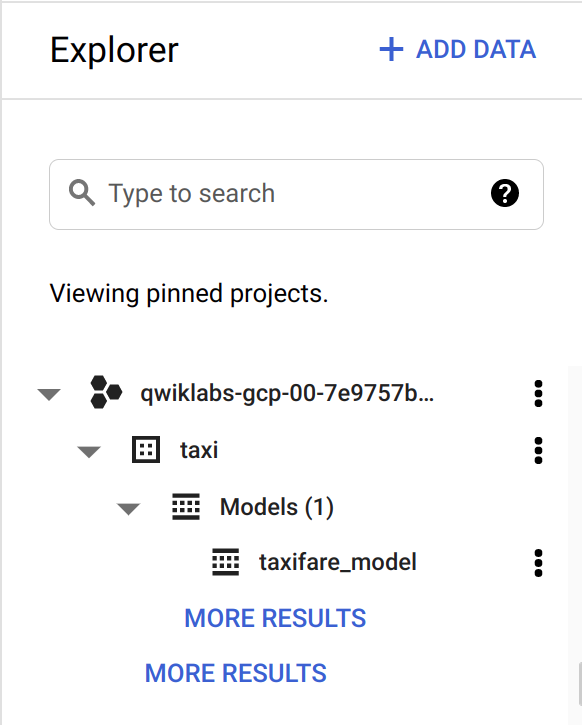
FROM taxitrips

Next, click **Run** to train your model.

Wait for the model to train (5 - 10 minutes).

After your model is trained, you will see the message "This statement will create a new model named qwiklabs-gcp-03-xxxxxxxx:taxi.taxifare\_model." which indicates that your model has been successfully trained.

Look inside your taxi dataset and confirm **taxifare\_model** now appears:



Next, you will evaluate the performance of the model against new unseen evaluation data.

**Evaluate classification model performance**

Select your performance criteria

For linear regression models you want to use a loss metric like [Root Mean Square Error (RMSE)](https://en.wikipedia.org/wiki/Root-mean-square_deviation). You want to keep training and improving the model until it has the lowest RMSE.

In BQML, mean\_squared\_error is a queryable field when evaluating your trained ML model. Add a SQRT() to get RMSE.

Now that training is complete, you can evaluate how well the model performs with this query using ML.EVALUATE. Copy and paste the following into the query **EDITOR** and click **Run**:

#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

))

You are now evaluating the model against a different set of taxi cab trips with your params.EVAL filter.

After the model runs, review your model results (your model RMSE value will vary slightly).

|  |  |
| --- | --- |
| **Row** | **rmse** |
| 1 | 9.477056435999074 |

After evaluating your model you get a **RMSE** of 9.47. Since we took the Root of the Mean Squared Error (RMSE) the 9.47 error can be evaluated in the same units as the total\_fare so it's +-$9.47.

Knowing whether or not this loss metric is acceptable to productionalize your model is entirely dependent on your benchmark criteria, which is set before model training begins. Benchmarking is establishing a minimum level of model performance and accuracy that is acceptable.

**Predict taxi fare amount**

Next you will write a query to use your new model to make predictions. Copy and paste the following into the query **EDITOR** and click **Run**:

#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

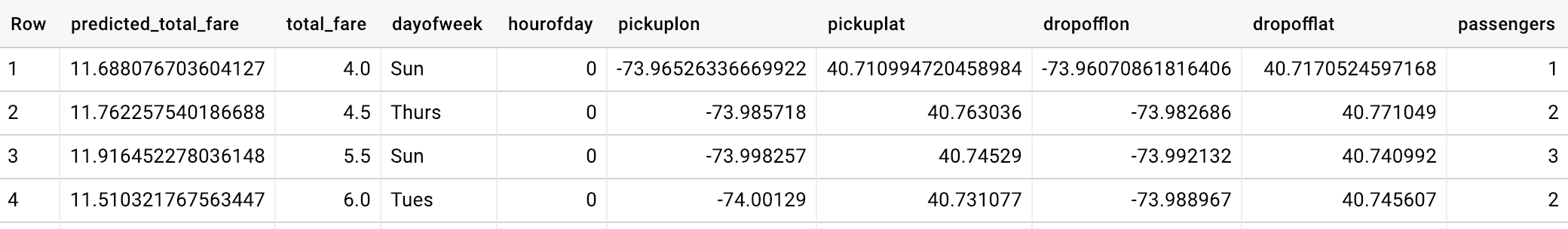
)

SELECT \*

FROM taxitrips

));

Now you will see the model's predictions for taxi fares alongside the actual fares and other features for those rides. Your results should look similar to those below:



**Improving the model with Feature Engineering**

Building Machine Learning models is an iterative process. Once we have evaluated the performance of our initial model, we often go back and prune our features and rows to see if we can get an even better model.

Filtering the training dataset

Let's view the common statistics for taxi cab fares. Copy and paste the following into the query **EDITOR** and click **Run**:

SELECT

COUNT(fare\_amount) AS num\_fares,

MIN(fare\_amount) AS low\_fare,

MAX(fare\_amount) AS high\_fare,

AVG(fare\_amount) AS avg\_fare,

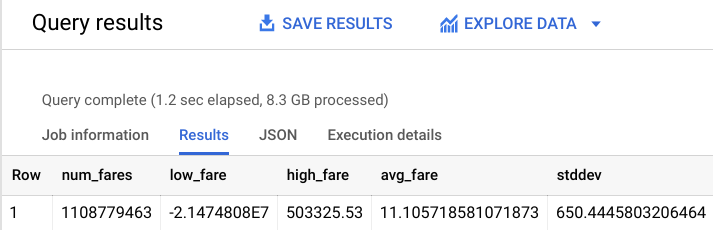
STDDEV(fare\_amount) AS stddev

FROM

`nyc-tlc.yellow.trips`

# 1,108,779,463 fares

You should receive a similar output:



As you can see, there are some strange outliers in our dataset (negative fares or fares over $50,000). Let's apply some of our subject matter expertise to help the model avoid learning on strange outliers.

Let's limit the data to only fares between $$6 and $$200. Copy and paste the following into the query **EDITOR** and click **Run**:

SELECT

COUNT(fare\_amount) AS num\_fares,

MIN(fare\_amount) AS low\_fare,

MAX(fare\_amount) AS high\_fare,

AVG(fare\_amount) AS avg\_fare,

STDDEV(fare\_amount) AS stddev

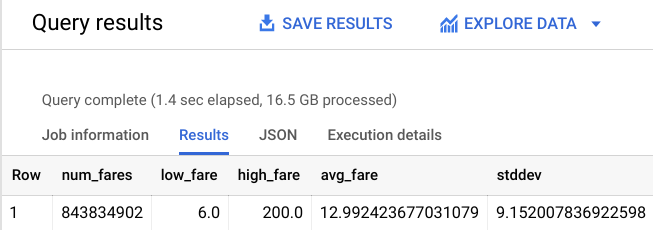
FROM

`nyc-tlc.yellow.trips`

WHERE trip\_distance > 0 AND fare\_amount BETWEEN 6 and 200

# 843,834,902 fares

You should receive a similar output:



That's a little bit better. While you're at it, let's limit the distance traveled so you're really focusing on New York City.

Copy and paste the following into the query **EDITOR** and click **Run**:

SELECT

COUNT(fare\_amount) AS num\_fares,

MIN(fare\_amount) AS low\_fare,

MAX(fare\_amount) AS high\_fare,

AVG(fare\_amount) AS avg\_fare,

STDDEV(fare\_amount) AS stddev

FROM

`nyc-tlc.yellow.trips`

WHERE trip\_distance > 0 AND fare\_amount BETWEEN 6 and 200

AND pickup\_longitude > -75 #limiting of the distance the taxis travel out

AND pickup\_longitude < -73

AND dropoff\_longitude > -75

AND dropoff\_longitude < -73

AND pickup\_latitude > 40

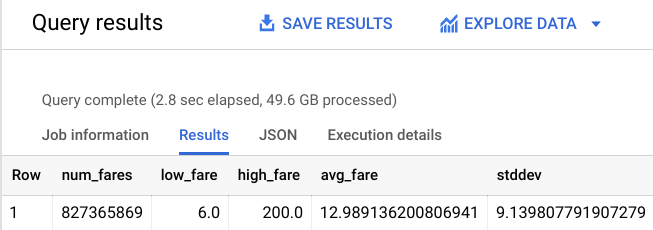
AND pickup\_latitude < 42

AND dropoff\_latitude > 40

AND dropoff\_latitude < 42

# 827,365,869 fares

You should receive a similar output:



You still have a large training dataset of over 800 million rides for our new model to learn from. Let's re-train the model with these new constraints and see how well it performs.

Retraining the model

Let's call our new model taxi.taxifare\_model\_2 and retrain our linear regression model to predict total fare. You'll note that you've also added a few calculated features for the [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance) (straight line) between the pick up and drop off.

Copy and paste the following into the query **EDITOR** and click **Run**:

CREATE OR REPLACE MODEL taxi.taxifare\_model\_2

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,

SQRT(POW((pickup\_longitude - dropoff\_longitude),2) + POW(( pickup\_latitude - dropoff\_latitude), 2)) as dist, #Euclidean distance between pickup and drop off

SQRT(POW((pickup\_longitude - dropoff\_longitude),2)) as longitude, #Euclidean distance between pickup and drop off in longitude

SQRT(POW((pickup\_latitude - dropoff\_latitude), 2)) as latitude, #Euclidean distance between pickup and drop off in latitude

passenger\_count AS passengers

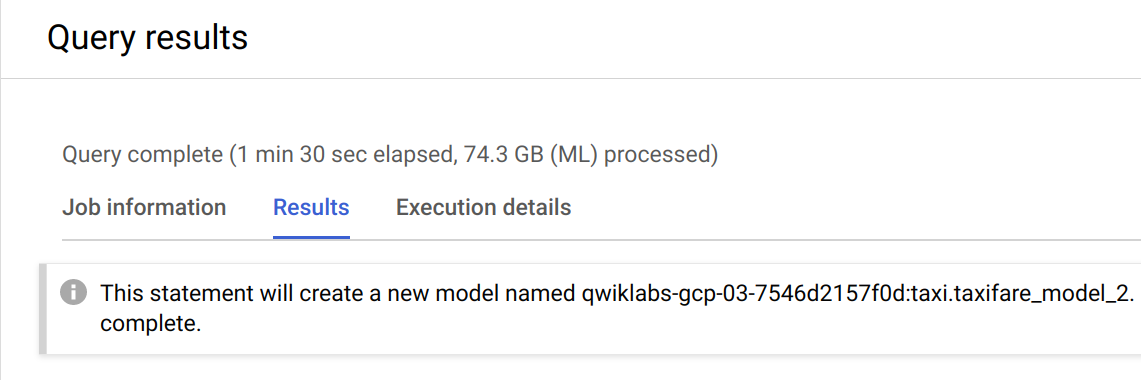
FROM

`nyc-tlc.yellow.trips`, daynames, params

WHERE trip\_distance > 0 AND fare\_amount BETWEEN 6 and 200

AND pickup\_longitude > -75 #limiting of the distance the taxis t

It may take a couple minutes to retrain the model. You can move onto the next step when you receive the following message in the Console:



Evaluate the new model

Now that our linear regression model has been optimized, let's evaluate the dataset with it and see how it performs. Copy and paste the following into the query **EDITOR** and click **Run**:

SELECT

SQRT(mean\_squared\_error) AS rmse

FROM

ML.EVALUATE(MODEL taxi.taxifare\_model\_2,

(

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,

SQRT(POW((pickup\_longitude - dropoff\_longitude),2) + POW(( pickup\_latitude - dropoff\_latitude), 2)) as dist, #Euclidean distance between pickup and drop off

SQRT(POW((pickup\_longitude - dropoff\_longitude),2)) as longitude, #Euclidean distance between pickup and drop off in longitude

SQRT(POW((pickup\_latitude - dropoff\_latitude), 2)) as latitude, #Euclidean distance between pickup and drop off in latitude

passenger\_count AS passengers

FROM

`nyc-tlc.yellow.trips`, daynames, params

WHERE trip\_distance > 0 AND fare\_amount BETWEEN 6 and 200

AND pickup\_longitude > -75 #limiting of the distance the taxis travel out

AND pickup\_longitude < -73

AND dropoff\_longitude > -75

AND dropoff\_longitude < -73

AND pickup\_latitude > 40

AND pickup\_latitude < 42

AND dropoff\_latitude > 40

AND dropoff\_latitude < 42

AND MOD(ABS(FARM\_FINGERPRINT(CAST(pickup\_datetime AS STRING))),1000) = params.EVAL

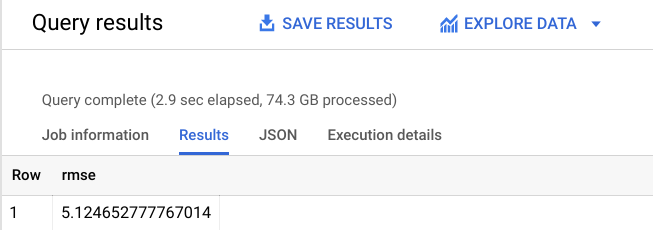
)

SELECT \*

FROM taxitrips

))

You should receive a similar output:



As you see, you've now gotten the RMSE down to: +-$$5.12 which is significantly better than +-$$9.47 for your first model.

Since RMSE defines the standard deviation of prediction errors, we see that the retrained linear regression made our model a lot more accurate.