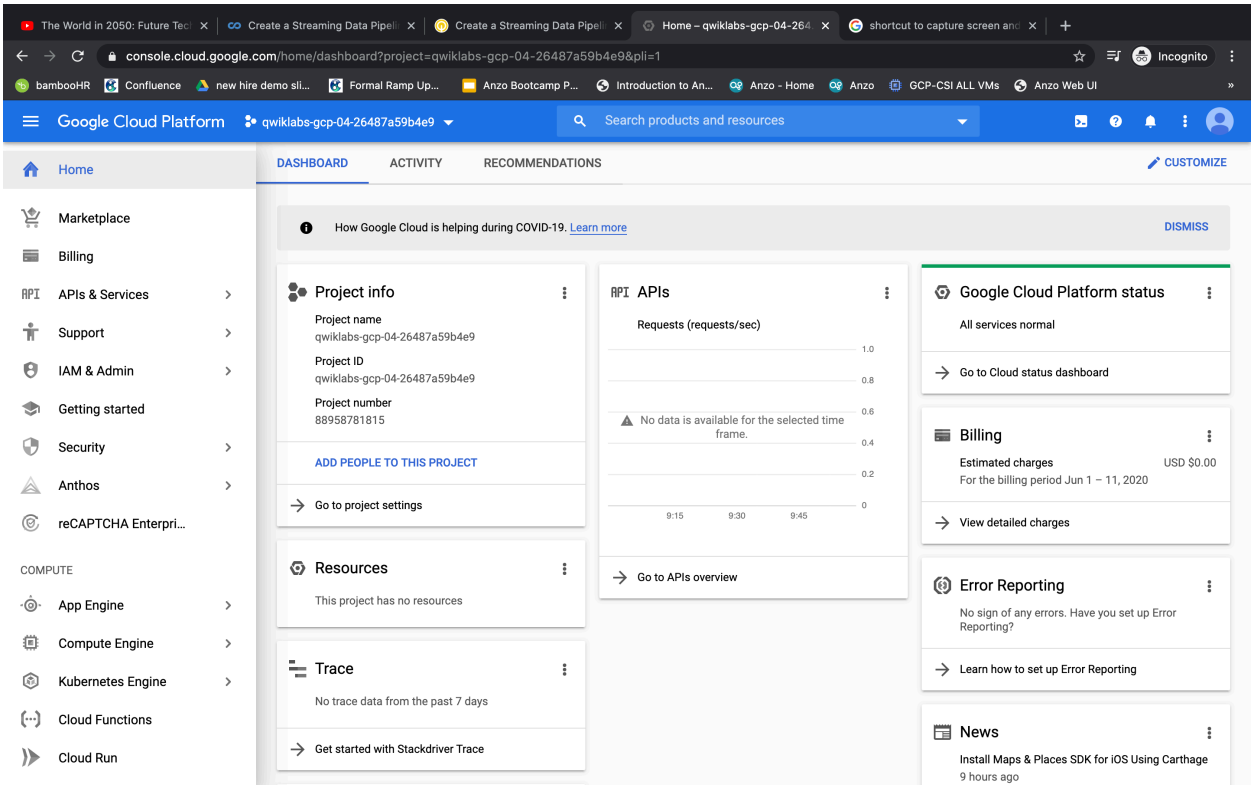
Create a Streaming Data Pipeline for a Real-Time Dashboard with Cloud Dataflow and Pub/Sub

You own a fleet of New York City taxi cabs and are looking to monitor how well your business is doing in real-time. You will build a streaming data pipeline to capture taxi revenue, passenger count, ride status, and much more and visualize the results in a management dashboard. Specifically, you will:

- Connect to a streaming data Topic in Cloud Pub/sub
- Ingest streaming data with Cloud Dataflow
- Load streaming data into BigQuery
- Analyze and visualize the results

GCP Home after login:



Create a Cloud Pub/Sub Topic

Cloud Pub/Sub is an asynchronous global messaging service. By decoupling senders and receivers, it allows for secure and highly available communication between independently written applications. Cloud Pub/Sub delivers low-latency, durable messaging.
In Cloud Pub/Sub, publisher applications and subscriber applications connect with one another through the use of a shared string called a **topic**. A publisher application creates and sends messages to a topic. Subscriber applications create a subscription to a topic to receive messages from it.
Google maintains a few public Pub/Sub streaming data topics for labs like this one.

Dataset: https://opendata.cityofnewyork.us/

Create a BigQuery dataset

BigQuery is a serverless data warehouse. Tables in BigQuery are organized into datasets. In this lab, messages published into Pub/Sub will be aggregated and stored in BigQuery.
To create a new BigQuery dataset:

Option 1: Command Line

Cloud shell will be in the Top right of the GCP Home console. Click on that and activate cloud shell

1. Open **Cloud Shell** and run the below command to create the taxirides dataset

```
bq mk taxirides
```

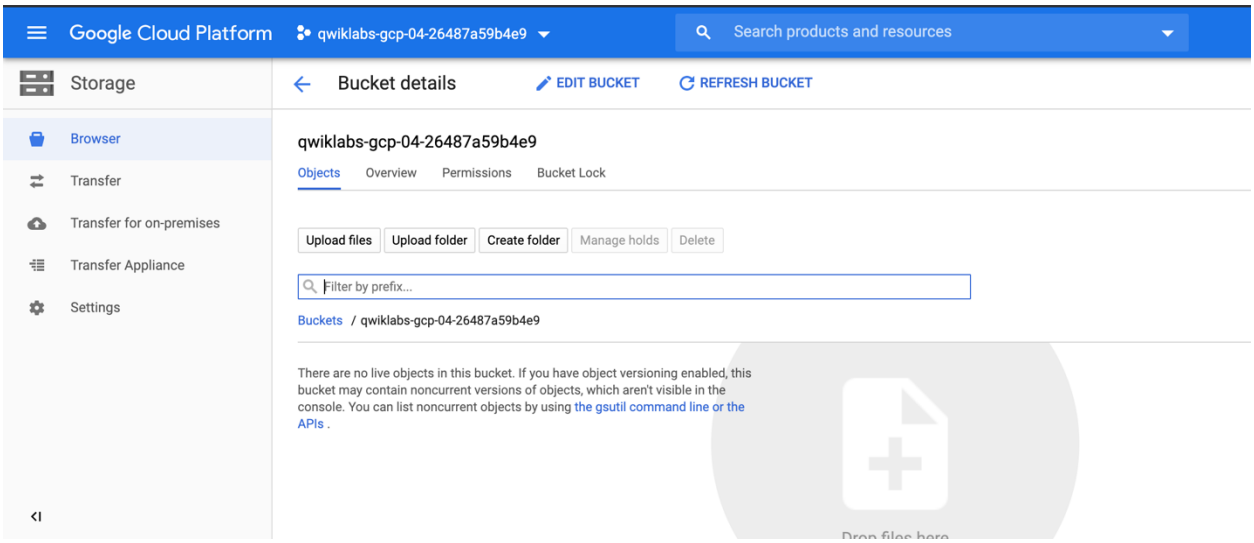
2. Run this command to create the taxirides.realtime table (empty schema we will stream into later)

```
bq mk \
--time_partitioning_field timestamp \
--schema ride_id:string,point_idx:integer,latitude:float,longitude:float,\
timestamp:timestamp,meter_reading:float,meter_increment:float,ride_status:str
ing,\
passenger_count:integer -t taxirides.realtime
```

Create a Cloud Storage Bucket

Skip this step if you already have a bucket created
Cloud Storage allows world-wide storage and retrieval of any amount of data at any time. You can use Cloud Storage for a range of scenarios including serving website content, storing data for archival and disaster recovery, or distributing large data objects to users via direct download. In this lab we will use Cloud Storage to provide working space for our Cloud Dataflow pipeline.

1. In the GCP Console, go to **Navigation menu** > **Storage**.
2. Click **CREATE BUCKET**.
3. For **Name**, paste in your GCP project ID.
4. For **Default storage class**, click **Multi-regional** if it is not already selected.
5. For **Location**, choose the selection closest to you.
6. Click **Create**.

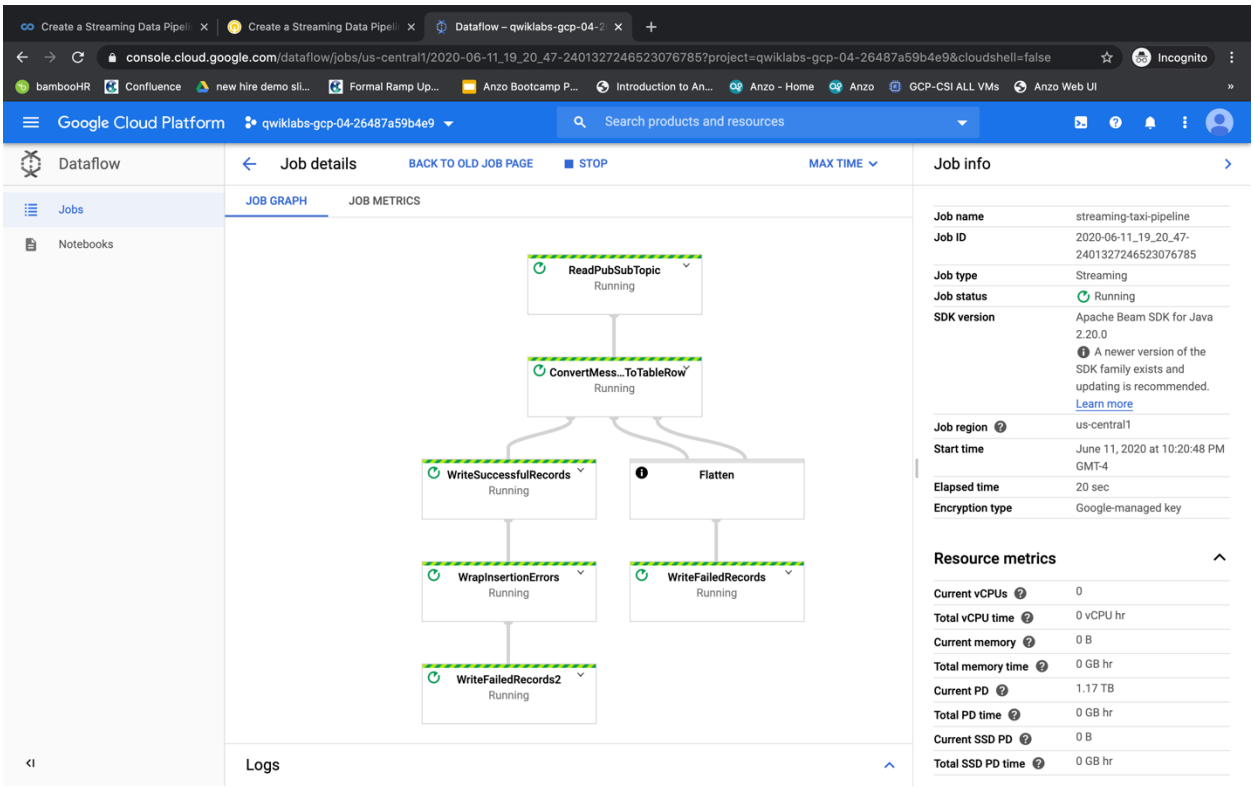


Set up a Cloud Dataflow Pipeline

Cloud Dataflow is a serverless way to carry out data analysis. In this lab, you will set up a streaming data pipeline to read sensor data from Pub/Sub, compute the maximum temperature within a time window, and write this out to BigQuery.

1. In the GCP Console, go to **Navigation menu** > **Dataflow**.
2. In the top menu bar, click **CREATE JOB FROM TEMPLATE**.
3. Enter **streaming-taxi-pipeline** as the Job name for your Cloud Dataflow job.
4. Under **Cloud Dataflow template**, select the Cloud Pub/Sub Topic to BigQuery template.
5. Under **Cloud Pub/Sub input topic**, enter `projects/pubsub-public-data/topics/taxirides-realtime`
6. Under **BigQuery output table**,
   enter `<myprojectid>:taxirides.realtime`
   Note: there is a colon : between the project and dataset name and a dot . between the dataset and table name

7. Under **Temporary Location**, enter `gs://<mybucket>/tmp/`
8. Click the **Run job** button.



Analyze the Taxi Data Using BigQuery

To analyze the data as it is streaming:

1. In the GCP Console, open the Navigation menu and select **BigQuery**.
2. Enter the following query in the Query editor and click **RUN**:

```
SELECT * FROM taxirides.realtime LIMIT 10
```

3. If no records are returned, wait another minute and re-run the above query (Dataflow takes 3-5 minutes to setup the stream). You will receive a similar output:

Query complete (1.7 sec elapsed, 0 B processed)

Job information | Results | JSON | Execution details

Row	timestamp	ride_id	meter_reading	ride_status	passenger_count
1	2019-04-24 22:09:13.734480 UTC	4bfc3d18-34c1-48db-ad93-1b9332cab8c3	21.313406	enroute	1
2	2019-04-24 22:09:13.734130 UTC	5a2099c2-7a9f-4d11-b8d4-9591990a95e0	7.5937257	enroute	1
3	2019-04-24 22:09:13.734130 UTC	1c276712-7fad-4cb9-b735-fddeed4df062	5.270588	enroute	3
4	2019-04-24 22:09:13.733910 UTC	13d7dd0f-1d81-4894-8f80-95c7d7f78a57	2.452924	enroute	2
5	2019-04-24 22:09:13.733890 UTC	c50e32e4-29ba-48ea-a026-bd47790060ff	7.9254036	enroute	1
6	2019-04-24 22:09:13.509450 UTC	a0c29640-d76d-4f43-a5b5-ba95182fbbca	8.9503765	enroute	1
7	2019-04-24 22:09:13.509260 UTC	d305d865-84be-48b1-9aae-60618333c912	19.628355	enroute	1
8	2019-04-24 22:09:13.509260 UTC	77e41112-bf33-4f8d-8217-dcd885b00ce4	19.70924	enroute	1
9	2019-04-24 22:09:13.509170 UTC	fb23b464-85e0-4e14-ad6f-10cea326b422	0.078625955	enroute	1

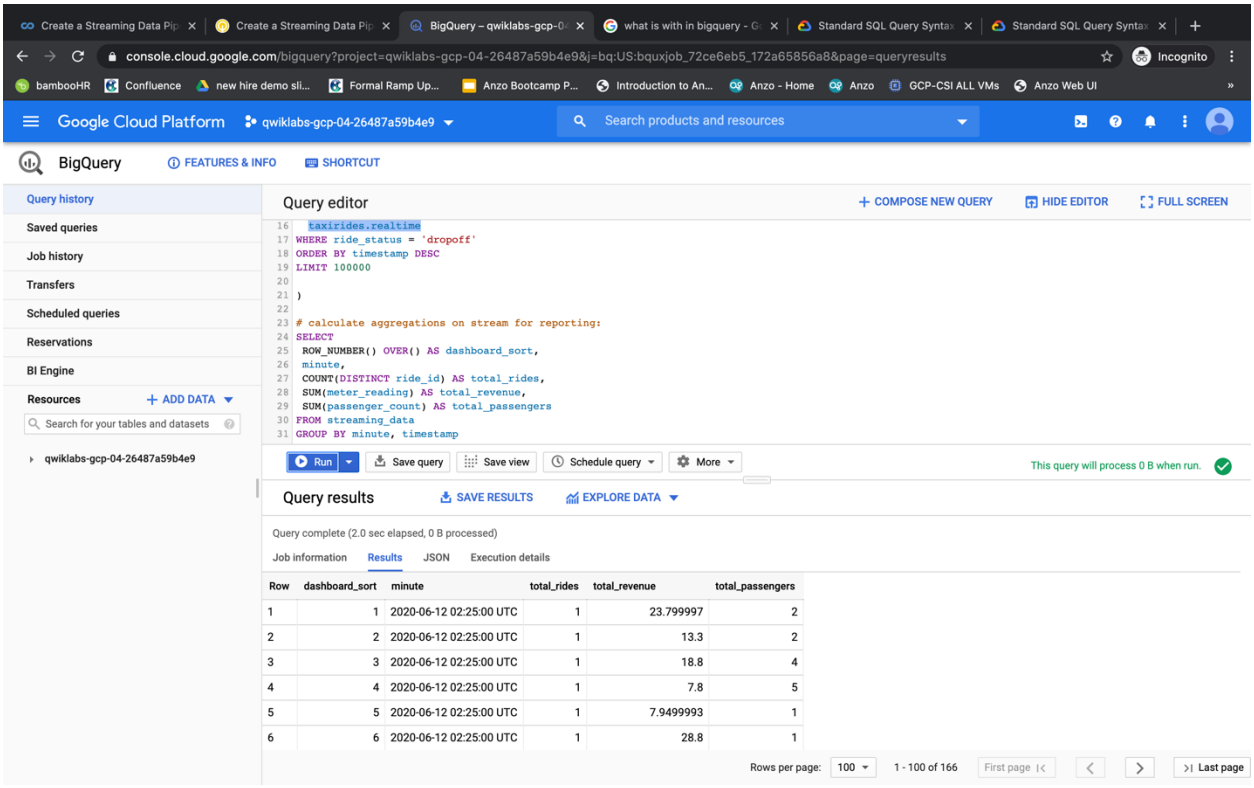
Perform aggregations on the stream for reporting

1. Copy and paste the below query and run

```
WITH streaming_data AS (

SELECT
  timestamp,
  TIMESTAMP_TRUNC(timestamp, HOUR, 'UTC') AS hour,
  TIMESTAMP_TRUNC(timestamp, MINUTE, 'UTC') AS minute,
  TIMESTAMP_TRUNC(timestamp, SECOND, 'UTC') AS second,
  ride_id,
  latitude,
```

```
  longitude,
  meter_reading,
  ride_status,
  passenger_count
FROM
  taxirides.realtime
WHERE ride_status = 'dropoff'
ORDER BY timestamp DESC
LIMIT 100000

)

# calculate aggregations on stream for reporting:
SELECT
 ROW_NUMBER() OVER() AS dashboard_sort,
 minute,
 COUNT(DISTINCT ride_id) AS total_rides,
 SUM(meter_reading) AS total_revenue,
 SUM(passenger_count) AS total_passengers
FROM streaming_data
GROUP BY minute, timestamp
```

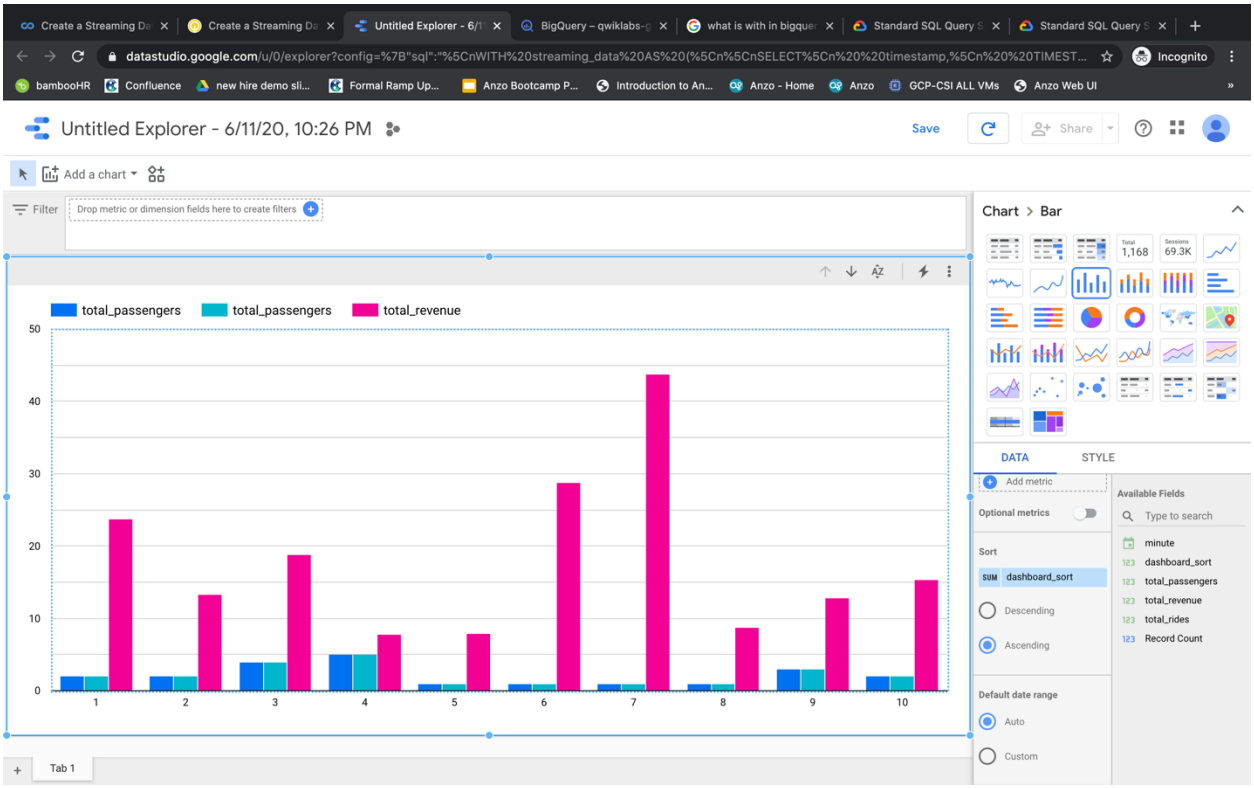
The result shows key metrics by the minute for every taxi drop-off

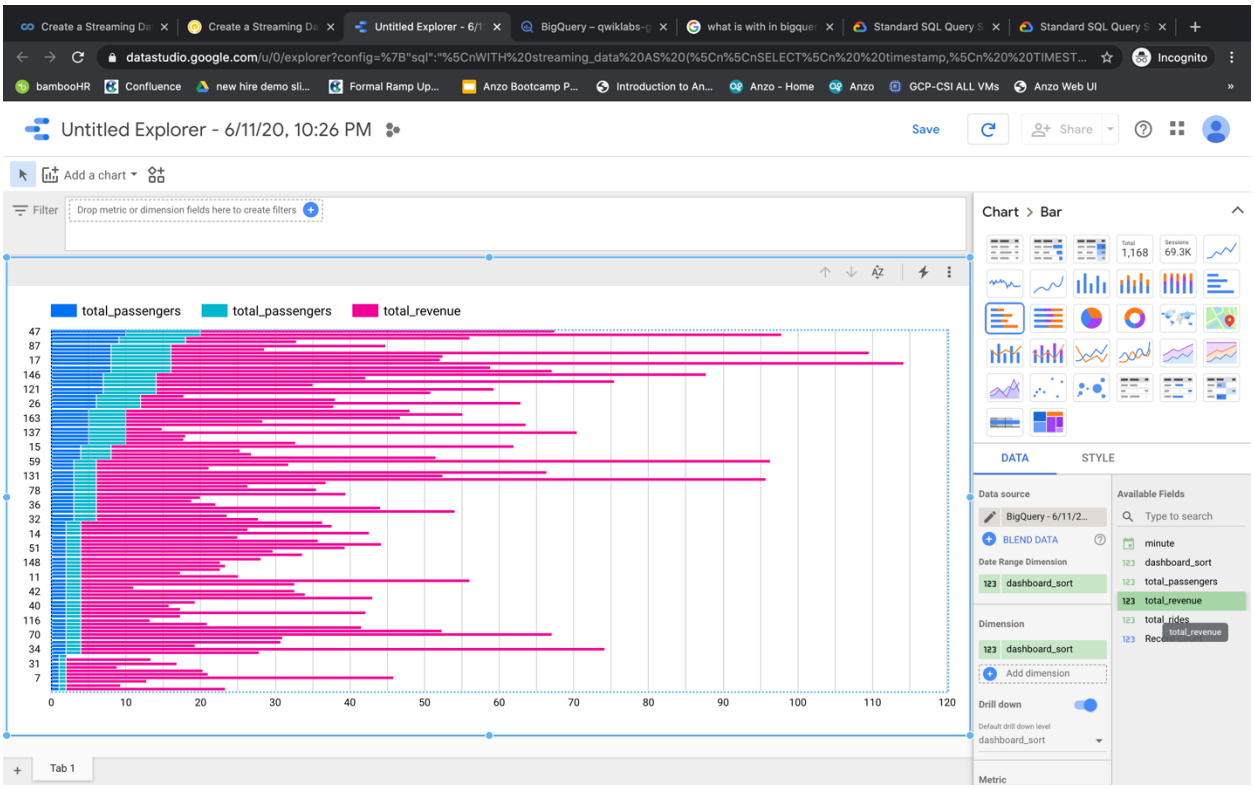


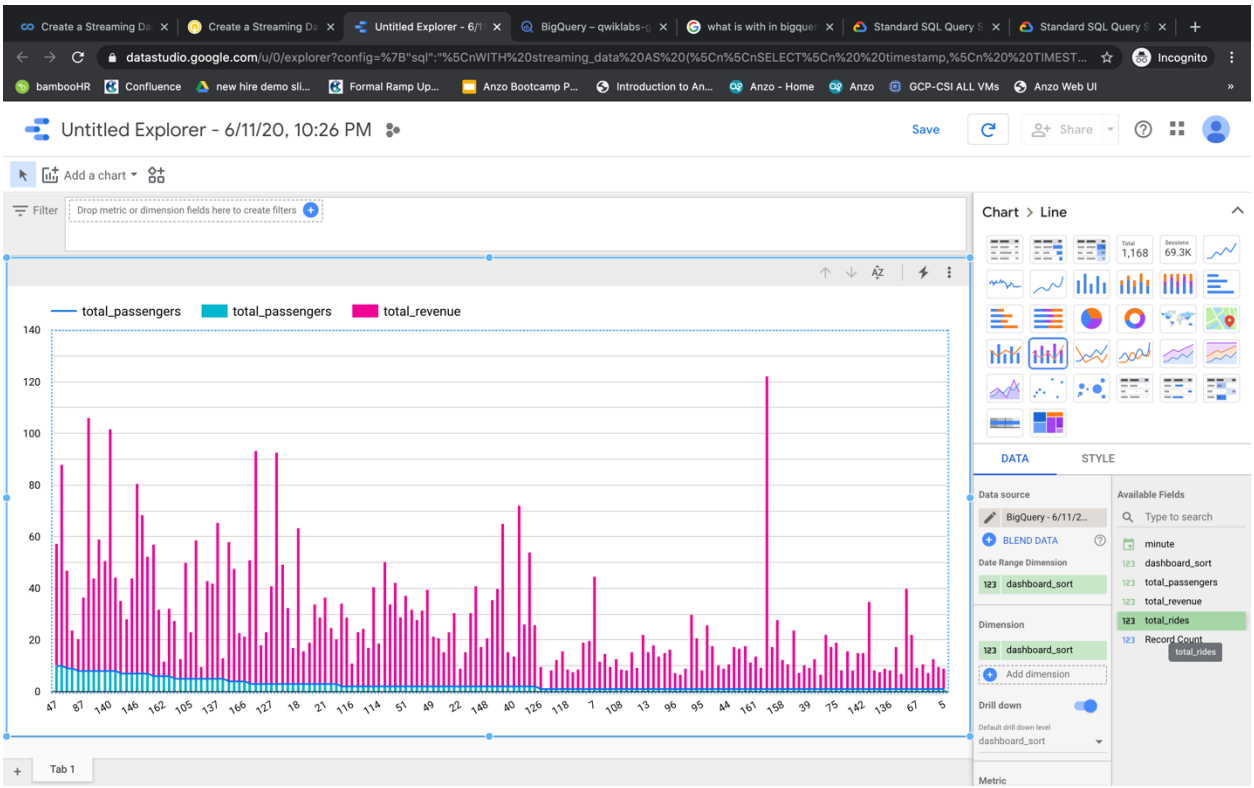
Create a Real-Time Dashboard

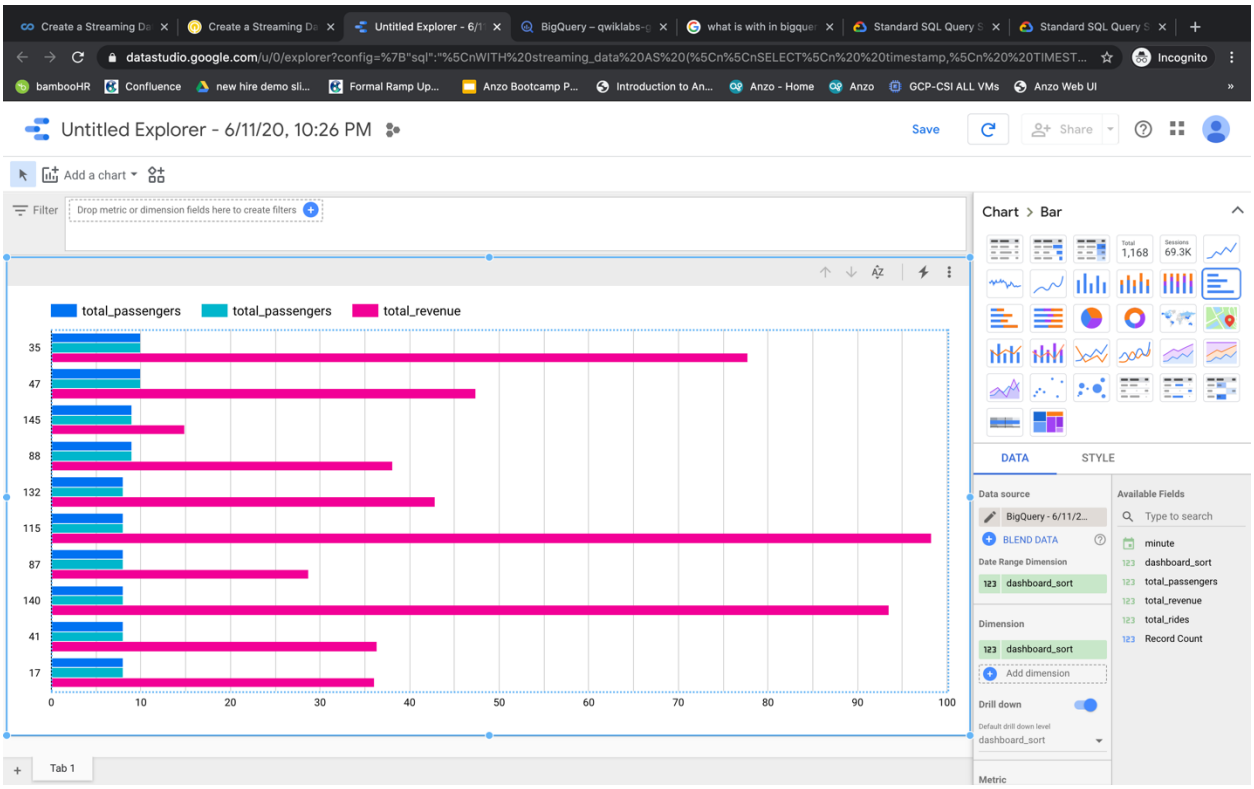
1. Click **Explore with Data Studio**
2. Specify the below settings:

- Chart type: column chart
- Date range dimension: dashboard_sort
- Dimension: dashboard_sort, minute
- Drill Down: dashboard_sort
- Metric: SUM() total_rides, SUM() total_passengers, SUM() total_revenue
- Sort: dashboard_sort Ascending (latest rides first)




Untitled Explorer - 6/11/20, 10:26 PM
Save
Share
Add a chart
Filter
Drop metric or dimension fields here to create filters
total_passengers
total_passengers
total_revenue
Chart > Bar
DATA
STYLE
Data source
BigQuery - 6/11/2...
BLEND DATA
Date Range Dimension
dashboard_sort
Dimension
dashboard_sort
Add dimension
Drill down
Default drill down level
dashboard_sort
Metric
Available Fields
Type to search
minute
dashboard_sort
total_passengers
total_revenue
total_rides
Record Count
Tab 1


Untitled Explorer - 6/11/20, 10:26 PM
Save
Share
Add a chart
Filter
Drop metric or dimension fields here to create filters
total_passengers
total_passengers
total_revenue
Chart > Line
DATA
STYLE
Data source
BigQuery - 6/11/2...
BLEND DATA
Date Range Dimension
dashboard_sort
Dimension
dashboard_sort
Add dimension
Drill down
Default drill down level
dashboard_sort
Metric
Available Fields
Type to search
minute
dashboard_sort
total_passengers
total_revenue
total_rides
Record Count
Tab 1



Stop the Cloud Dataflow job

1. Navigate back to Cloud Dataflow

2. Click the **streaming-taxi-pipeline**
3. Click **Stop Job** and **Cancel** pipeline

This will free up resources for your project