Creating a Streaming Data Pipeline for a Real-Time Dashboard with Dataflow _lab

In this lab, 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

creating database, partitioning for performance and define schema using cloud shell

Open **Cloud Shell**

```
# create the taxirides dataset.
bq mk taxirides
#  create the taxirides.realtime table
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:string,\
passenger_count:integer -t taxirides.realtime
```

create cloud storage bucket

Create a Cloud Storage bucket
Name, paste in your **GCP Project ID**
Location type, click **Multi-region**

Set up a Dataflow Pipeline

1. Enter **streaming-taxi-pipeline** as the Job name for your Dataflow job.
2. Under **Dataflow template**, select the **Pub/Sub Topic to BigQuery** template.
3. Under **Input Pub/Sub topic**, enter projects/pubsub-public-data/topics/taxirides-realtime
4. Under **BigQuery output table**, enter <myprojectid>:taxirides.realtime
5. Under **Temporary location**, enter gs://<mybucket>/tmp/.
6. Click **Show Optional Parameters** and input the following values as listed below:
   Max workers: 2
   Number of workers: 2
7. Click the **RUN JOB** button.

Big query and click explore data

```
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 1000)
```

```
# 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
```

Stop work

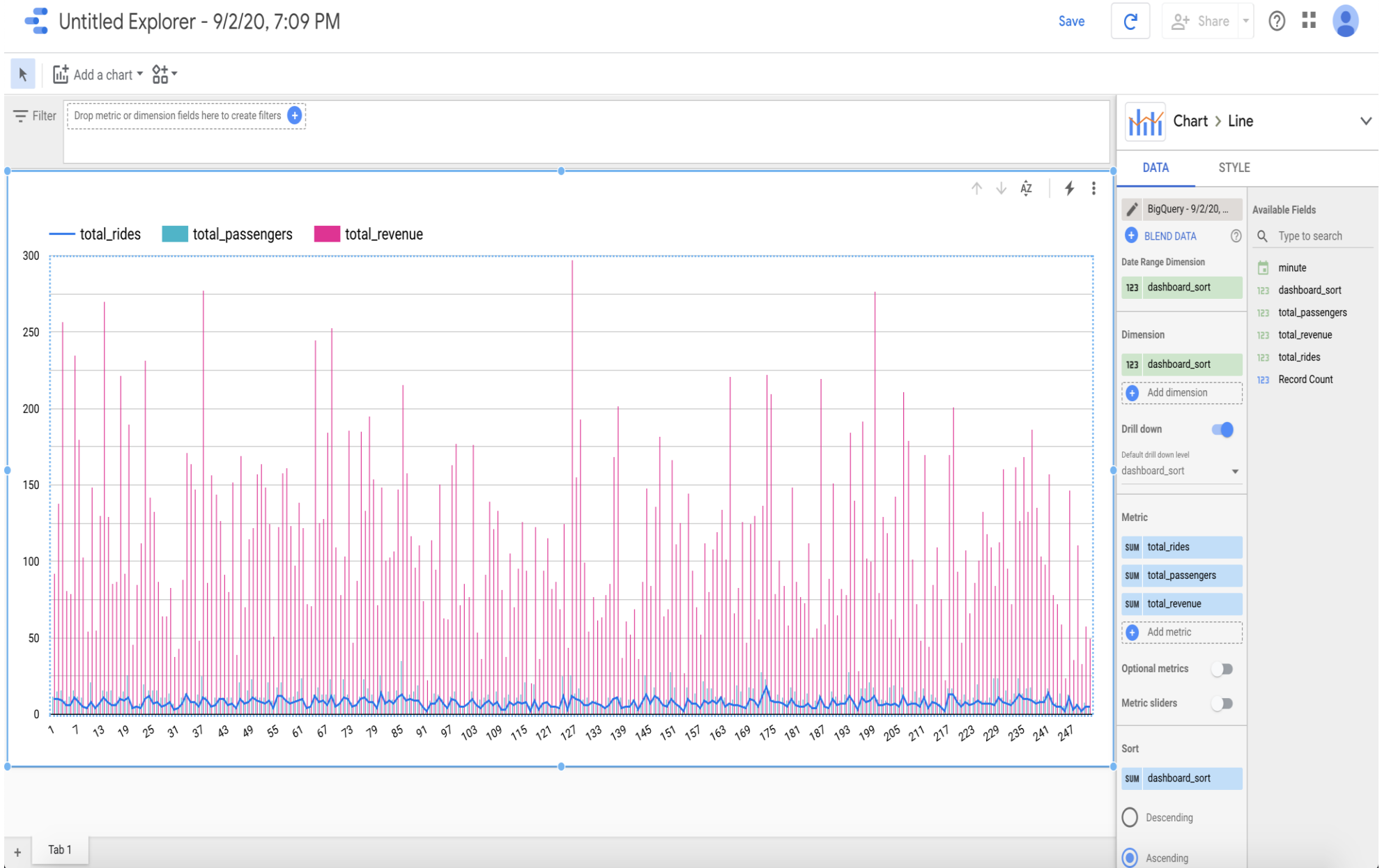
1. Navigate back to **Dataflow**.
2. Click the **streaming-taxi-pipeline** or the new job name.
3. Click **STOP** and select **Cancel** > **STOP JOB**

Google studio

Specify the below settings:

- **Chart type:** Combo chart
- **Date range Dimension:** dashboard_sort
- **Dimension:** dashboard_sort
- **Drill Down:** dashboard_sort (Make sure that Drill down option is turned ON)
- **Metric:** SUM() total_rides, SUM() total_passengers, SUM() total_revenue
- **Sort:** dashboard_sort, Ascending (latest rides first)

Google studio



Google studio with custom query

1. select data source and go more options
2.Under **CUSTOM QUERY**, click **qwiklabs-gcp-xxxxxxx** > **Enter Custom Query**, add the following query.

```
SELECT
  *
FROM
  taxirides.realtime
WHERE
  ride_status='dropoff’
```

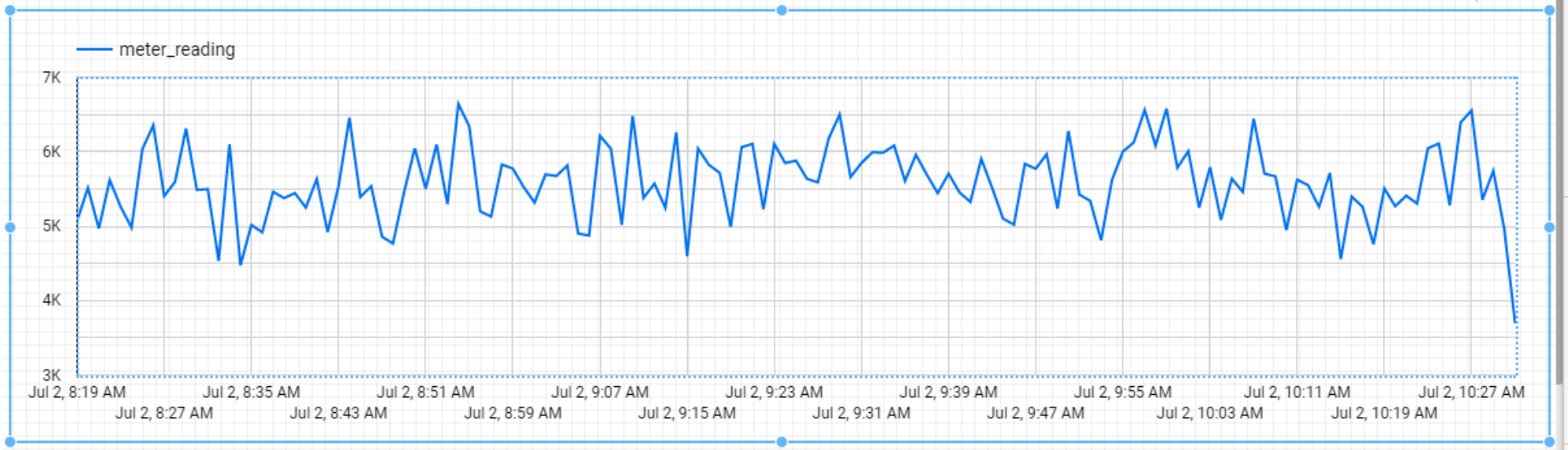
3. Add timeseries chart Change the field **timestamp** type to **Date & Time > Date Hour Minute (YYYYMMDDhhmm)**.
4.in the **Data** panel on the right, change the following:

- **Dimension:** timestamp
- **Metric:** meter_reading(SUM)

Google studio with custom query

	ride_id	Record Count
1.	00d3fdd1-d3ff-471a-a525-...	1
2.	01995720-1af1-41f7-a499-...	1
3.	039d4315-dfe4-404a-aa1f-...	1
4.	053d1eed-d0cf-4593-a095-...	1
5.	068194a7-9685-414c-b134...	1
6.	0808021e-b717-4b4f-a189-...	1
7.	086b1d76-6c8f-4920-a6f3-...	1
8.	0892d4e6-f778-445d-8235...	1

1 - 100 / 49315



DATA STYLE

Data source
BigQuery
BLEND DATA
Date Range Dimension
timestamp

Dimension
timestamp
Drill down
Breakdown Dimension
Add dimension

Metric
SUM meter_reading
Add metric
Optional metrics
Metric sliders

Default date range
Auto
Custom
Auto date range
Comparison date range
None

Filter
Time Series Filter

Available Fields
Type to search
ABC ride_id
ABC ride_status
timestamp
123 latitude
123 longitude
123 meter_increment
123 meter_reading
123 passenger_count
123 point_idx
123 Record Count
ADD A FIELD

Appendix

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. Pub/Sub delivers low-latency, durable messaging.

In 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. We'll be using the NYC Taxi & Limousine Commission's open 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.

Dataflow is a serverless way to carry out data analysis. In this lab, you 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.