DataEng S24: Project Assignment 2

Validate, Transform, Enhance and Store

Due date: May 12, 2024 at 10pm

****

Our Virtual Machine instance got deleted last week and we were unable to backup any data saved previously as discussed in class. We could secure only one day's worth of data with a fully functional data pipeline. Going forward, the receiver will directly ingest the data to database without saving to any external files

****

By now your pipeline should be automatically gathering and transferring TriMet breadcrumb records daily but not yet doing much with the data. The next step in building your data pipeline is to get your data in shape for analysis. This includes:

A. understanding the contents of the breadcrumb data
B. reviewing the needed schema for the data
C. validating the data
D. transforming the data
E. storing the data into a database server
F. example queries

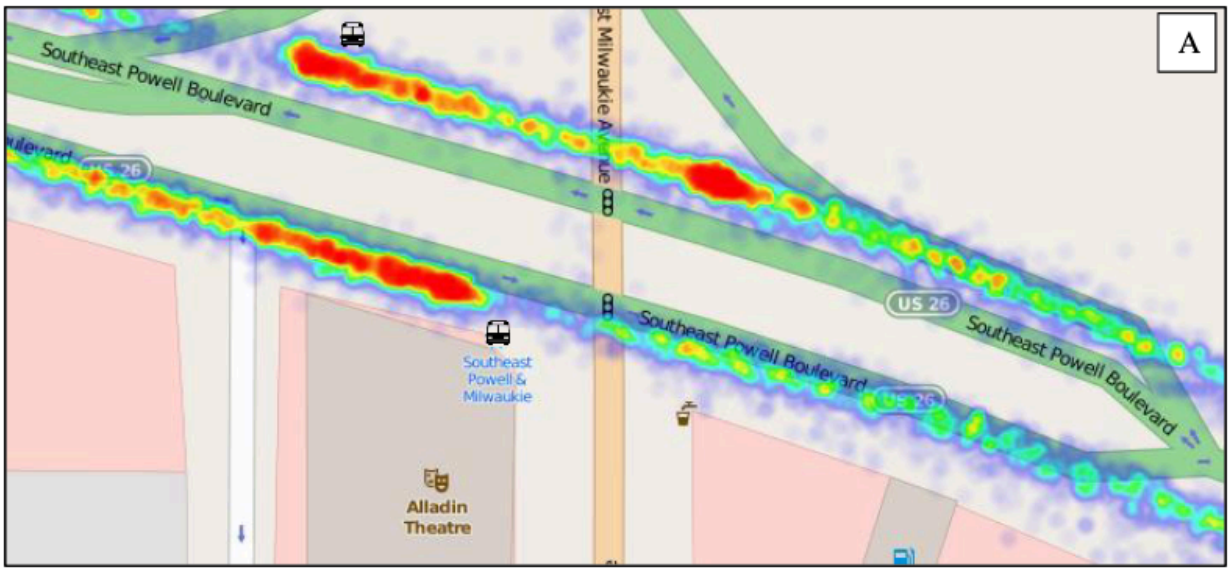
Your job is to enhance your Pub/Sub receiver to perform steps A-E so that your pipeline ingests high-quality data daily into a database server in preparation for visualization.

A. Review the Data

Have a look at a few of the data records and see if you can determine the meaning of each record attribute. Also, have a look at the only documentation that we have for the data (provided to us by TriMet). The documentation is not very good, so we are counting on you to give better descriptions of each field. In your submission document, provide an improved description of each data field.

B. Database Schema

We plan to develop a visualization application that builds visualizations similar to this:



This diagram shows vehicle speeds as a heatmap overlaid on a street map for a given latitude/longitude rectangle, route, date range and time range. Colors in the heatmap correspond to average speeds of buses at each GPS location for the selected route, date range and/or time range.

To achieve this we need you to build database tables (or views) that conform to the following schema.

Table: **BreadCrumb**

tstamp	latitude	longitude	speed	trip_id

The BreadCrumb table keeps track of each individual breadcrumb reading. Each row of the BreadCrumb table corresponds to a single GPS sensor reading from a single bus.

tstamp: the moment at which the event occurred, i.e., at which the bus's GPS location was recorded. This should be a Postgres "timestamp" type of column.

latitude: a float value indicating fractional degrees of latitude. Latitude and Longitude together indicate the location of the bus at time=tstamp.

longitude: a float value indicating the fractional degrees of longitude. Latitude and Longitude indicate the location of the bus at time=tstamp.

speed: a float value indicating the speed of the bus at time=tstamp. Speed is measured in meters per second.

trip_id: this integer column indicates the identifier of the trip in which the bus was participating. A trip is a single run of a single bus servicing a single route. This column is a foreign key referencing trip_id in the Trip table.

Table: **Trip**

trip_id	route_id	vehicle_id	service_key	direction

The Trip table keeps track of information about each bus trip. A “trip” is a single run of a single bus over a single route.

trip_id: unique integer identifier for the trip.

route_id: integer identifying the route that the trip was servicing. For this project we do not have a separate Route table, but if we did, then this would be a foreign key reference to the Route table. Note that you will not have enough information to properly populate this field during this assignment #2.

vehicle_id: integer identifying the vehicle that serviced the trip. A trip is serviced by only one vehicle but a vehicle services potentially many trips.

service_key: one of ‘Weekday’, ‘Saturday’, or ‘Sunday’ depending on the day of the week on which the trip occurred OR depending on whether the trip occurred on a Holiday recognized by TriMet. Note that you will not have enough information to properly populate this field during this assignment #2.

direction: either ‘0’ or ‘1’ indicating whether the trip traversed the route from beginning to end (‘0’) or from end to beginning (‘1’). Note that this “direction” has a completely different meaning than the same-named column in the BreadCrumb table (sorry about that). Also note that you will not have enough information to properly populate this field during assignment #2.

See the ER diagram and DDL code for the schema. Please implement this schema precisely in your database so that the visualization tool will work for you.

C. Data Validation

Create 20 distinct data validation assertions for the breadcrumb data. These should be in English (not python). Include them in your submission document (see below).

Then implement at least 10 of the assertions in your Pub/Sub receiver code. Implement a variety of different types of assertions so that you can experience with each of the major types of data validation assertions: existence, limit, intra-record check, inter-record check, summary, referential integrity, and distribution/statistical assertions.

D. Data Transformation

Add code to your Pub/Sub receiver to transform your data. Your transformations should be driven by either the need to resolve validation assertion violations or the need to shape the data for the schema. Describe any/all needed transformations in your submission document.

A transformation that you must do is to add a speed column to conform with the BreadCrumb table schema. TriMet did not give this information in their breadcrumb service; however, we can extrapolate this information from the data that we have. To find speed, use the formula below:

$$speed = \frac{\Delta distance}{\Delta time}$$

Each breadcrumb contains data of the time it was sent and the odometer reading of the bus. You will need to find the distance traveled and the exact time between each consecutive breadcrumb of a trip to make the calculation.

Here's an example of finding the speed:

	EVENT_NO_TRIP	EVENT_NO_STOP	OPD_DATE	VEHICLE_ID	METERS	ACT_TIME	GPS_LONGITUDE	GPS_LATITUDE	GPS_SATELLITES	GPS_HDOP
47888	259173279	259173281	15FEB2023:00:00:00	4223	177061	68354	-122.525793	45.489655	12.0	0.8
47889	259173279	259173281	15FEB2023:00:00:00	4223	177118	68359	-122.525102	45.489513	12.0	0.8

The above DataFrame shows two consecutive breadcrumbs sent from the same bus on the same trip. The METERS column shows the odometer reading of the bus at the time of breadcrumb emission in meters, and the ACT_TIME column is the time in which the breadcrumb was sent in seconds. Calculate as below:

$$\frac{(METERS_{47889} - METERS_{47888})}{(ACT\_TIME_{47889} - ACT\_TIME_{47888})} = \frac{(177118 - 177061)}{(68359 - 68354)} = 11.4$$

Note that to calculate the speed of a breadcrumb, you need the previous breadcrumb of that trip. We cannot make this calculation for the very first breadcrumb in each trip. Therefore, we are expecting you to do the following: calculate the speed for the second breadcrumb of the trip; use that speed, the calculated speed of the second breadcrumb, for the first breadcrumb of the trip as well.

E. Storage in Database Server

1. Install and configure a PostgreSQL database server on your Virtual Machine. Refer to this document to learn how. For more information on PostgreSQL commands, you can see guides like this one: PostgreSQL Cheat Sheet.

2. Enhance your data pipeline to load your transformed data into your database server. You are free to use any of the data loading techniques that we discussed in class. The data should be loaded ASAP after your Pub/Sub receiver receives the data, validates the data and transforms the data so that you have a reliable end-to-end data pipeline running daily and automatically.

F. Example Queries

Answer the following questions about the TriMet system using your vehicle group's sensor data in your database. In your submission document include your query code, number of rows in each query result (if applicable) and first five rows of the result (if applicable).

Submission

Congratulations! Your data pipeline is now working end-to-end!

To submit your completed Assignment 2, create a Google document containing the table shown below. Share the document as viewable by anybody at PSU who has the link. You do NOT need to share it with the individual instructors. Then include the URL of the document when using the DataEng project assignment submission form.

DataEng Project Assignment 2 Submission Document

Construct a table showing each day for which your pipeline successfully, automatically processed one complete day's worth of sensor readings. The table should look like this:

Date	Day of Week	# Sensor Readings	# rows added to your database
2023-05-13	Monday	330767	330767
			h

Documentation of Each of the Original Data Fields

For each of the fields of the breadcrumb data, provide any documentation or information that you can determine about it. Include bounds or distribution data where appropriate. For example, for something like “Vehicle ID”, say something more than “It is the identification number for the vehicle”. Instead, add useful information such as “the integers in this field range from <min val> to <max val>, and there are <n> distinct vehicles identified in the data. Every vehicle is used on weekdays but only 50% of the vehicles are active on weekends.”

EVENT_NO_TRIP:
This field denotes the event number for a trip.
It uniquely identifies each trip event.
The values are integers.
EVENT_NO_STOP:
This field represents the event number for a stop within a trip.
It uniquely identifies each stop event.
The values are integers.
OPD_DATE:
This field contains the date and time of the event.
The format is "DDMMMYYYY:HH:MM:SS", e.g., "12JAN2023:00:00:00".
It provides a precise timestamp for each event.
VEHICLE_ID:
This field denotes the identification number for the vehicle.
The integers in this field range from the minimum to the maximum observed value in the dataset.
It uniquely identifies each vehicle in the dataset.
METERS:
This field represents the distance covered by the vehicle in meters at the time of the event.
It provides information on the vehicle's movement.
ACT_TIME:
This field denotes the actual time of the event in seconds.
It provides additional timing information for each event.
GPS_SATELLITES:
This field represents the number of GPS satellites detected by the vehicle at the time of the event.

The count should not exceed 12, which is the maximum possible number of satellites in that range.
It indicates the quality of GPS signal reception.
GPS_HDOP:
This field represents the Horizontal Dilution of Precision (HDOP) value of the GPS signal at the time of the event.
It provides information about the accuracy of the GPS signal.

Data Validation Assertions

List 20 or more data validation assertion statements here. These should be English language sentences similar to “The speed of a TriMet bus should not exceed 100 miles per hour” . You will only implement a subset of them, so feel free to write assertions that might be difficult to evaluate. Create assertions for all of the fields, even those, like GPS_HDOP, that might not be used in your database schema.

Existence_Assertions:

- Each vehicle ID has a reasonable number of entries, not excessively high or low.
- There are no duplicate IDs.
- For every longitude data, there must be a latitude value and vice versa.
- For every vehicle ID, there must be Event_No_Trip and Event_No_Stop data.
- The ACT_TIME value for each subsequent entry is greater than the ACT_TIME value of the previous entry for the same VEHICLE_ID.
- If meters of any particular vehicle ID changes then the ACT_TIME should also change.
- The vehicle ID should only have four digits.

Limit_Assertions:

- GPS_LONGITUDE values are within the range [-180, 180].
- GPS_LATITUDE values are within the range [-90, 90].
- GPS_HDOP values should be less than 2 as those are accurate values.
- GPS_Satillite count should not be greater than 12 as those are the maximum possible number of satellites in that range.
- GPS longitude should always be negative, and GPS latitude should always be positive.
- The integral part of GPS longitude should be -122, and for GPS latitude, it should be 45.

Intra-record Check Assertion:

- Meters should be gradually increasing.
- All the columns with integer values should not have any other data types.

Inter-record Check Assertion:

- Check that the speed calculated between consecutive GPS coordinates falls within a reasonable range for the type of vehicle.
- Verify that the time difference between consecutive events is reasonable and within a certain threshold (e.g., not too short or too long).

Summary Assertion:

- Calculate the average speed of vehicles for each day of the week and check if there are any significant differences between weekdays and weekends.
- Determine the distribution of GPS_HDOP values and analyze if they are evenly distributed or skewed towards certain ranges.

Referential Integrity Assertion:

- Check if all the vehicle IDs referenced in the dataset exist in a separate vehicle registry table.
- Verify that all the trip IDs referenced in the dataset exist in a separate trip schedule table.

Distribution/Statistical Assertion:

- Analyze the distribution of speeds across all vehicles and identify any outliers or patterns.
- Determine the average distance traveled by vehicles per day and check for any trends over time.

Data Transformations

Describe any transformations that you implemented either to react to validation violations or to shape your data to fit the schema. For each, give a brief description of the transformation along with a reason for the transformation.

Timestamp Creation: transformation is to create a timestamp from the 'OPD_DATE' and 'ACT_TIME' fields. The raw data provides the operational date and the actual time separately. Combining them into a single timestamp makes it easier to work with time-related operations and queries.

Speed Creation: Calculating the speed based on the distance covered and the time elapsed between consecutive events.calculating it from distance and time, we can provide valuable insights into the vehicle's behavior during each trip. This transformation ensures the accuracy of speed calculations and handles edge cases such as the first record of each trip and negative speeds.

The third transformation fills missing values in the 'GPS_LATITUDE' and 'GPS_LONGITUDE' columns with zeros.Missing GPS coordinates can occur due to various reasons such as signal loss or device malfunction. Filling them with zeros allows us to still include these records in the dataset while indicating that the coordinates are unknown or invalid.

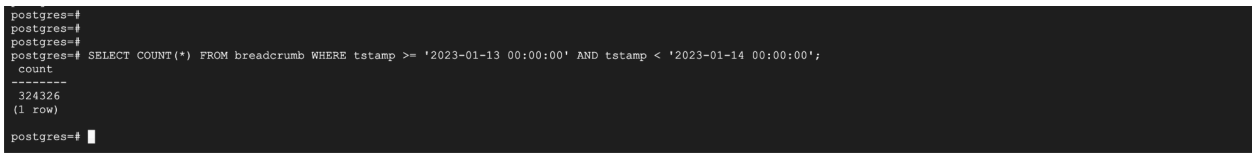
The fourth transformation adds dummy columns 'ROUTE_ID' and 'DIRECTION' to the DataFrame with default values.These dummy columns are added to prepare the DataFrame where these attributes are required. By initializing them with default values, we ensure that every row in the DataFrame has these columns, even if they are not populated with meaningful data yet.

Example Queries

Provide your responses to the questions listed in Section F above. For each question, provide the SQL you used to answer the questions along with the count of the number of rows returned (where applicable) and a listing of the first 5 rows returned (where applicable).

1. How many breadcrumb reading events occurred on January 13, 2023?

   SELECT COUNT(*) FROM breadcrumb WHERE tstamp >= '2023-01-13 00:00:00' AND tstamp < '2023-01-13 00:00:00';

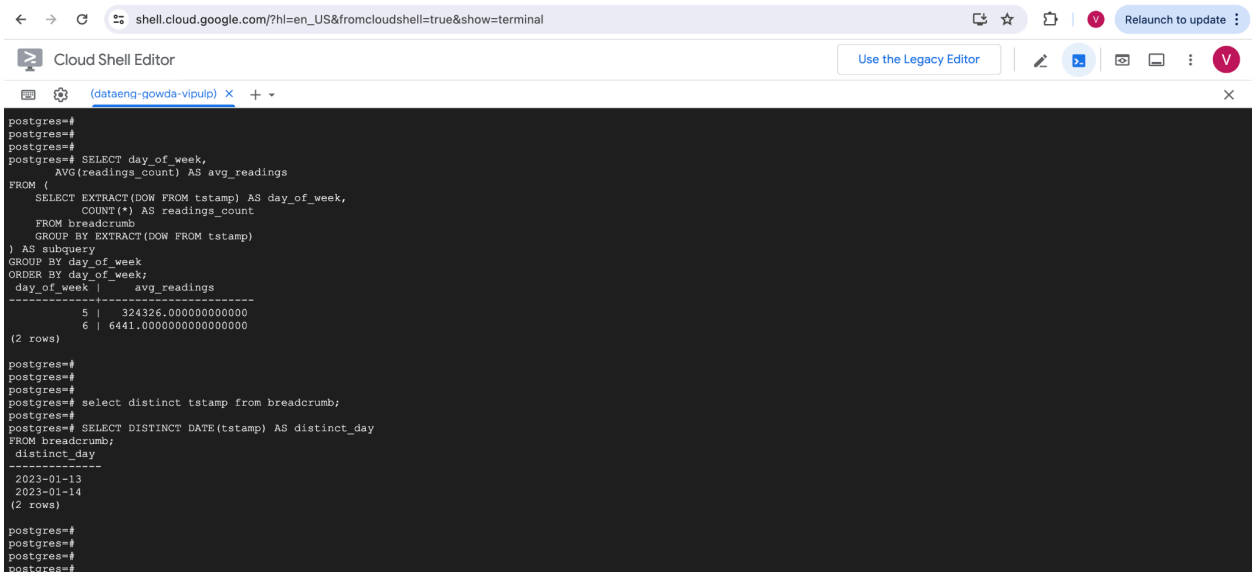


2. How many breadcrumb reading events occurred on January 14, 2023?

   SELECT COUNT(*) FROM breadcrumb WHERE tstamp >= '2023-01-14 00:00:00' AND tstamp < '2023-01-15 00:00:00';

3. On average, how many breadcrumb readings are collected on each day of the week?

   SELECT day_of_week, AVG(readings_count) AS avg_readings FROM (SELECT EXTRACT(DOW FROM tstamp) AS day_of_week, COUNT(*) AS readings_count FROM breadcrumb GROUP BY EXTRACT(DOW FROM tstamp)) AS subquery GROUP BY day_of_week ORDER BY day_of_week;



4. List the TriMet trips that traveled a section of I-205 between SE Division and SE Powell on January 13, 2023. To find this, search for all trips that have breadcrumb readings that occurred within a lat/long bounding box such as [(45.497805, -122.566576), (45.504025, -122.563187)].

```
SELECT DISTINCT t.trip_id
FROM Trip t
JOIN breadcrumb b ON t.trip_id = b.trip_id
WHERE b.tstamp >= '2023-01-13 00:00:00' AND b.tstamp < '2023-01-14 00:00:00'
  AND b.latitude BETWEEN 45.497805 AND 45.504025
  AND b.longitude BETWEEN -122.566576 AND -122.563187 limit 5;
```

5. List all breadcrumb readings on a section of US-26 west side of the tunnel (bounding box: [(45.506022, -122.711662), (45.516636, -122.700316)]) during fridays between 4pm and 6pm. Order the readings by tstamp. Then list readings for Saturdays between 6am and 8am. How do these two time periods compare for this particular location?

```
SELECT *
FROM BreadCrumb
WHERE (EXTRACT(DOW FROM tstamp) = 5 AND EXTRACT(HOUR FROM tstamp) BETWEEN 16 AND 18)
```

```
  OR (EXTRACT(DOW FROM tstamp) = 5 AND EXTRACT(HOUR FROM tstamp) BETWEEN 6
AND 8)
  AND latitude BETWEEN 45.506022 AND 45.516636
  AND longitude BETWEEN -122.711662 AND -122.700316
ORDER BY tstamp;
```

```
postgres=#
postgres=# SELECT *
FROM BreadCrumb
WHERE (EXTRACT(DOW FROM tstamp) = 5 AND EXTRACT(HOUR FROM tstamp) BETWEEN 16 AND 18)
   OR (EXTRACT(DOW FROM tstamp) = 5 AND EXTRACT(HOUR FROM tstamp) BETWEEN 6 AND 8)
   AND latitude BETWEEN 45.506022 AND 45.516636
   AND longitude BETWEEN -122.711662 AND -122.700316
ORDER BY tstamp limit 5;
       tstamp        | latitude  |  longitude  | speed |  trip_id
---------------------+-----------+-------------+-------+-----------
 2023-01-13 06:10:58 | 45.506992 | -122.710877 |  21.4 | 238583261
 2023-01-13 06:11:03 | 45.507407 |  -122.70964 |  17.2 | 238583261
 2023-01-13 06:11:08 | 45.508142 | -122.708707 |  26.2 | 238583261
 2023-01-13 06:11:13 |  45.50905 | -122.708053 |  18.2 | 238583261
 2023-01-13 06:11:18 | 45.509962 | -122.707415 |  22.4 | 238583261
(5 rows)
```

6. What is the maximum speed reached by any bus in the system?

```
SELECT MAX(speed) AS max_speed
FROM BreadCrumb;
```

```
postgres=#
postgres=#
postgres=#
postgres=# SELECT MAX(speed) AS max_speed
FROM BreadCrumb;
 max_speed
-----------
        40
(1 row)

postgres=#
```

7. List all speeds and give a count of the number of vehicles that move precisely at that speed during at least one trip. Sort the list by most frequent speed to least frequent.

```
SELECT speed, COUNT(DISTINCT t.vehicle_id) AS num_vehicles
FROM BreadCrumb b
JOIN Trip t ON b.trip_id = t.trip_id
GROUP BY speed
ORDER BY COUNT(DISTINCT t.vehicle_id) DESC, speed;
```

```
postgres=#
postgres=#
postgres=# SELECT speed, COUNT(DISTINCT t.vehicle_id) AS num_vehicles
FROM BreadCrumb b
JOIN Trip t ON b.trip_id = t.trip_id
GROUP BY speed
ORDER BY COUNT(DISTINCT t.vehicle_id) DESC, speed limit 5;
 speed | num_vehicles
-------+--------------
     0 |           73
  0.25 |           73
   0.4 |           73
   0.5 |           73
   0.6 |           73
(5 rows)

postgres=#
```

8. Which is the longest (in terms of time) trip of all trips in the data?

```
SELECT trip_id, MAX(tstamp) - MIN(tstamp) AS trip_duration
FROM BreadCrumb
GROUP BY trip_id
ORDER BY trip_duration DESC
LIMIT 1;
```

```
postgres=#
postgres=#
postgres=#
postgres=# SELECT trip_id, MAX(tstamp) - MIN(tstamp) AS trip_duration
FROM BreadCrumb
GROUP BY trip_id
ORDER BY trip_duration DESC
LIMIT 1;
  trip_id  | trip_duration
-----------+---------------
 237779705 | 02:31:13
(1 row)

postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
```

9. Are there differences in the number of breadcrumbs between a non-holiday Wednesday, a non-holiday Saturday, and a holiday? What can that tell us about TriMet's operations on those types of days?

   As I don't have any data for wednesday, could not conclude on the differences.

```
SELECT CASE
        WHEN EXTRACT(DOW FROM tstamp) IN (0, 6) THEN 'Weekend'
        WHEN EXTRACT(DOW FROM tstamp) = 2 THEN 'Wednesday'
        ELSE 'Non-holiday'
      END AS day_type,
      COUNT(*)
FROM BreadCrumb
WHERE EXTRACT(YEAR FROM tstamp) = 2023
GROUP BY day_type;
```

```
postgres=#
postgres=#
postgres=#
postgres=#
postgres=# SELECT CASE
         WHEN EXTRACT(DOW FROM tstamp) IN (0, 6) THEN 'Weekend'
         WHEN EXTRACT(DOW FROM tstamp) = 2 THEN 'Wednesday'
         ELSE 'Non-holiday'
       END AS day_type,
       COUNT(*)
FROM BreadCrumb
WHERE EXTRACT(YEAR FROM tstamp) = 2023
GROUP BY day_type;
  day_type   | count
-------------+--------
 Non-holiday | 324326
 Weekend     |   6441
(2 rows)

postgres=#
```

10. Devise three new, interesting questions about the TriMet bus system that can be answered by your breadcrumb data. Show your questions, their answers, the SQL you used to get the answers and the results of running the SQL queries on your data (the

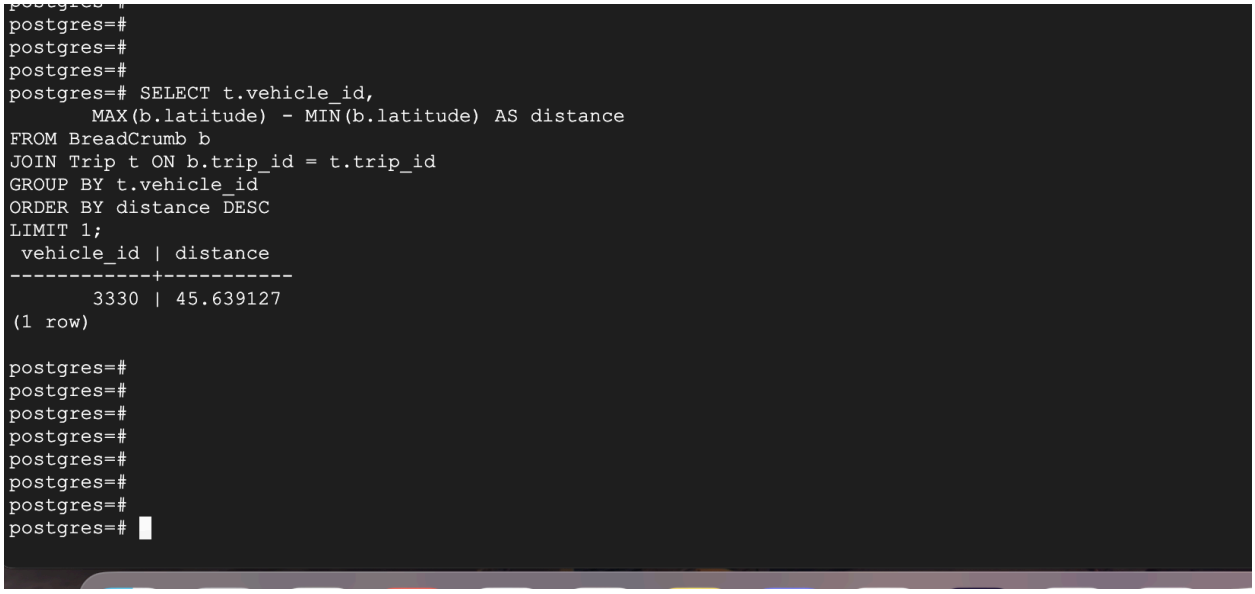
number of result rows, and first five rows returned).

What is the average speed of buses during rush hours on weekdays?

```
SELECT AVG(speed) AS avg_speed
FROM BreadCrumb
WHERE EXTRACT(DOW FROM tstamp) BETWEEN 1 AND 5
AND EXTRACT(HOUR FROM tstamp) BETWEEN 7 AND 9
OR EXTRACT(HOUR FROM tstamp) BETWEEN 16 AND 18;
```

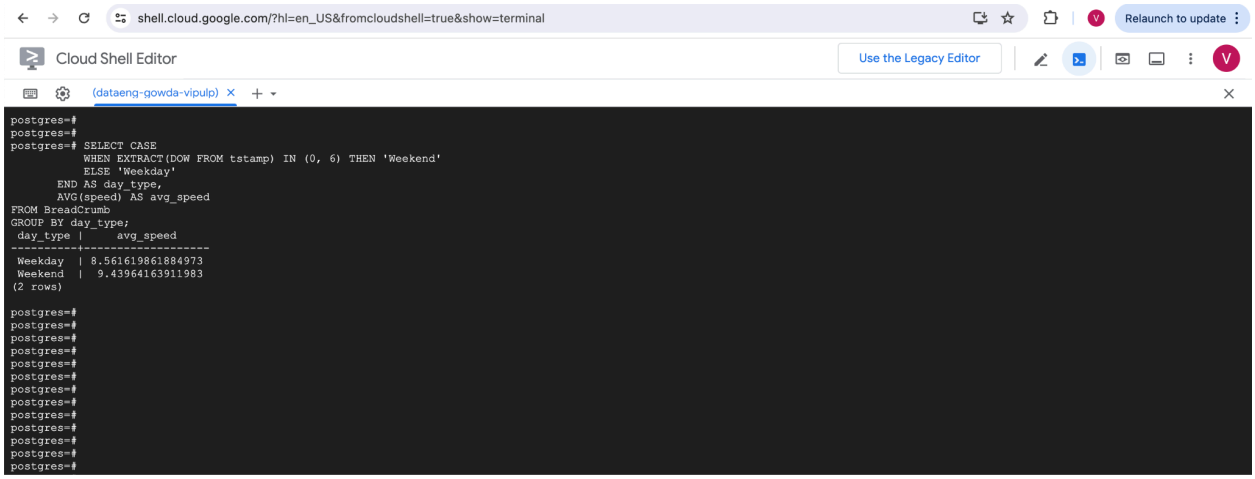
Which vehicle traveled the longest distance in a single trip?

```
SELECT t.vehicle_id,
       MAX(b.latitude) - MIN(b.latitude) AS distance
FROM BreadCrumb b
JOIN Trip t ON b.trip_id = t.trip_id
GROUP BY t.vehicle_id
ORDER BY distance DESC
LIMIT 1;
```



How does the average speed of vehicles vary by the day of the week?

```
SELECT CASE
        WHEN EXTRACT(DOW FROM tstamp) IN (0, 6) THEN 'Weekend'
        ELSE 'Weekday'
      END AS day_type,
      AVG(speed) AS avg_speed
FROM BreadCrumb
GROUP BY day_type;
```



Your Code

Provide a reference to the repository where you store your python code. If you are keeping it private then share it with the Professor (rbi@pdx.edu or mina8@pdx.edu) and TA (vysali@pdx.edu).

Team Members Roles:

Jathin: Data Validation Queries - Involved in writing assertion queries.
Raghuram:Data Validation Code - Involved in writing assertion code.
Siddhanth - Involved in writing SQL code for Example Queries and Documentation of Each of the Original Data Fields.
Vipul - Wrote the extended receiver logic and connection to the database server and ingesting data.