Here is a link to my code: https://github.com/NAlexH2/de-proj-cs510
You can find milestone02 in `‘milestone-submissions/milestone02’`

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:

The discrepancies between sensor readings and values actually logged are due to dropping NaN lat/long rows or speed anomalies, OR, if a specific VEHICLE_ID ran for more than a day. For example: 04-12 has more readings because a particular vehicle or vehicles were run for more than 24hrs, past the entire day of 04-11.

Date	Day of Week	# Sensor Readings	# rows added to db (trip : breadcrumb)
04-11	Thursday	314,815	314,479
04-12	Friday	295,354	295,217
04-13	Saturday	332,888	332,677
04-14	Sunday	342,189	341,945
04-15	Monday	328,164	327,921
04-16	Tuesday	235,787	235,648
04-17	Wednesday	262,890	262,728
04-18	Thursday	328,945	328,417
04-19	Friday	342,744	342,513
04-20	Saturday	340,480	340,283
04-21	Sunday	342,828	342,665
04-22	Monday	237,235	237,100
04-23	Tuesday	No data to read from API	No data to write to SQL DB
04-24	Wednesday	244,514	244,277
04-25	Thursday	230,081	229,948
04-26	Friday	356,243	355,871
04-27	Saturday	330,750	330,617

04-28	Sunday	344,805	344,588
04-29	Monday	343,842	343,608
04-30	Tuesday	266,047	265,855
05-01	Wednesday	216,700	216,463
05-02	Thursday	237,768	237,644
05-03	Friday	322,432	321,710
05-04	Saturday	333,861	333,510
05-05	Sunday	328,075	327,900
05-06	Monday	342,437	342,142
05-07	Tuesday	259,461	259,321
05-08	Wednesday	248,719	248,158
05-09	Thursday	310,386	310,036
05-10	Friday	365,326	365,074
05-11	Saturday	336,657	336,294

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:

- **The ID for every vehicle and the trip they are currently running for that specific route.**

EVENT_NO_STOP:

- **The ID for the route that the time event took place during and after the incremental change for the EVENT_NO_STOP.**

OPD_DATE:

- **The date that the bus was running, recorded as starting at midnight of that day, but is not when the event was recorded. See ACT_TIME for more information.**

VEHICLE_ID:

- **The ID for a specific vehicle ranging from 2905 to 4531, totally 100 distinct IDs to be used when making calls to the bus API to collect data for the pipeline. The values range from 2905 to 4531, but are not a complete inclusive collection of the numbers in-between, that is an ID does not increment +1 to it each time. The VID after 2935 is 3020, and after 3059 is 3103, as an example.**

METERS:

- **The distance the bus has traveled on that day in meters. This distance is not tracked by GPS signal, but by the computer on the bus itself. The buses will travel on average between 90 to 100 miles per day.**

ACT_TIME:

- **The time in seconds since data collection has commenced. These values *normally* are 5 seconds apart, but there are instances where this is no longer the case. This time is automatically corrected for at some point by either waiting to report/collect, or extending until the next 5 seconds takes place. This is used to calculate the exact timestamp of that activity in relation to OPD_DATE.**

GPS_LONGITUDE:

- **The longitude location reported by GPS for any particular event. This data can be recorded as low as 0 GPS_SATELLITES as long as there is a 23.1 GPS_HDOP. This value is never less than -123.1, or is greater than -122.2.**

GPS_LATITUDE:

- **The latitude location reported by GPS for any particular event. This data can be recorded as low as 0 GPS_SATELLITES as long as there is a 23.1 GPS_HDOP. This value is *always* between 45.3 and 45.7**

GPS_SATELLITES:

- **Number of satellites the bus is connected to. The min is 0 and the max is 12. This does not guarantee that 12 satellites can report lat and long coordinates. The majority of readings have 12 satellites.**

GPS_HDOP:

- **The horizontal dilution of precision for the bus and acts as a measurement for error propagation in satellite navigational geometry. The records missing data for lat and long all exclusively have a GPS_HDOP of 3.9 to 23.1. A bus can only have a GPS_HDOP of 23.1 and a lat/long recorded ONLY if it has 0 GPS satellites it is linked up to.**

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.

1. **Longitude is never less than -124 but must be greater than -122 (-124 > x <= -122)**
2. **Latitude is never more than 46 or less than 45 (46 > x >= 45)**
3. **GPS_HDOP values >= 4 and < 23.1 are missing long/lat information**
4. lat/long data is missing
5. **The fewest number of GPS_SATELLITES is 0**
6. **The most number of GPS_SATELLITES is 12**
7. **Not all 0 GPS_SATELLITES are missing a lat/long**
8. **12 GPS_SATELLITES all have lat/long**
9. **Every record has a activity time**
10. Every EVENT_NO_TRIP has one or more EVENT_NO_STOPs associated with it.
11. **Every record has a meters entry**
12. **No bus should exceed 70 mph/113 kmh (this is easier since it’s measured in meters!)**
13. There could be duplicate EVENT_NO_STOP because of a variety of reasons (traffic, waiting at a stop, etc.)
14. No VEHICLE_ID shares a EVENT_NO_TRIP with another VEHICHLE_ID
15. Most buses run during the weekday
16. More busses are active later in the day than earlier in the day
17. There are fewer busses running late at night
18. ACT_TIME does not reset when a new EVENT_NO_TRIP has been started
19. METERS does not reset when a new EVENT_NO_TRIP has started
20. There are not any null/NaN VEHICLE_IDs

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.

I removed all records with NaN values. This removal only is a concern regarding GPS coordinates missing and no other data.

I sorted the data by VEHICLE_ID and ACT_TIME prior to doing any additional transformations (after dropping columns).

I added TIMESTAMP and SPEED, which then allowed me to remove OPD_DATE and ACT_TIME.

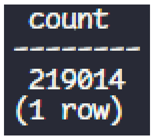
Any row with a SPEED value greater than 160KPH got removed as this was an anomaly in the data. This was due to the fact that, for whatever reason, the activity time wasn't recording in 5 second intervals (usually fewer seconds or not on a a time evenly divisible by 5), giving the illusion a bus went from about 25MPH to 110MPH in about 2 seconds.

Prior to loading into the database, I split the dataframes into two different dataframes: table and breadcrumb. These two acted as placeholders for the data that would eventually be inserted into the database. I added empty/default valued columns to the dataframes I created to make loading into the db much easier to match the schema of the two different tables.

Example Queries

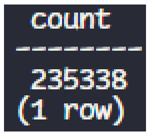
Provide your responses to the questions listed in Section F above. For each question, provide the SQL you used to answer the questions along with the count of the number of rows returned (where applicable) and a listing of the first 5 rows returned (where applicable).

1. How many breadcrumb reading events occurred on January 1, 2023?
   Query: select count (*) from breadcrumb where TO_CHAR(tstamp, 'YYYY-MM-DD')='2023-01-01';

   Answer:
   

```
 count
--------
 219014
(1 row)
```

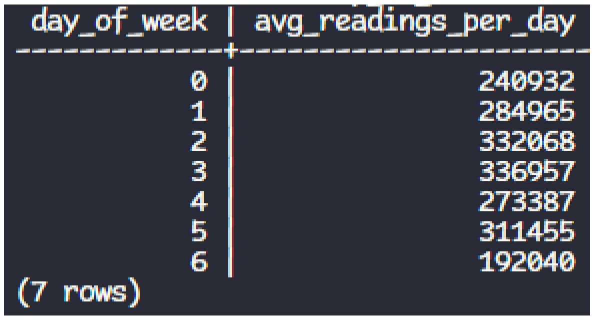
2. How many breadcrumb reading events occurred on January 2, 2023?
   Query: select count (*) from breadcrumb where TO_CHAR(tstamp, 'YYYY-MM-DD')='2023-01-02'

   Answer:
   

```
 count
--------
 235338
(1 row)
```

3. On average, how many breadcrumb readings are collected on each day of the week?
   Query: SELECT EXTRACT(DOW FROM tstamp) AS day_of_week, COUNT(*) / COUNT(DISTINCT DATE(tstamp)) AS avg_readings_per_day FROM breadcrumb GROUP BY day_of_week ORDER BY day_of_week;

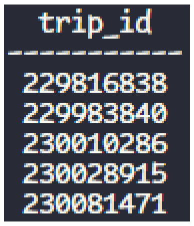
   Answer: 0 = Sunday, 6 = Saturday



```
 day_of_week | avg_readings_per_day
-------------+----------------------
           0 |               240932
           1 |               284965
           2 |               332068
           3 |               336957
           4 |               273387
           5 |               311455
           6 |               192040
(7 rows)
```

4. List the TriMet trips that traveled a section of I-205 between SE Division and SE Powell on January 1, 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)].
   Query:
   SELECT DISTINCT trip_id FROM breadcrumb WHERE tstamp >= '2023-01-01 00:00:00' AND tstamp < '2023-01-02 00:00:00' AND latitude BETWEEN 45.497805 AND 45.504025 AND longitude BETWEEN -122.566576 AND -122.563187;

   Answer:



```
  trip_id
-----------
 229816838
 229983840
 230010286
 230028915
 230081471
```

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 Mondays between 4pm and 6pm. Order the readings by tstamp. Then list readings for Sundays between 6am and 8am. How do these two time periods compare for this particular location?

Query:

Monday between 4pm and 6pm

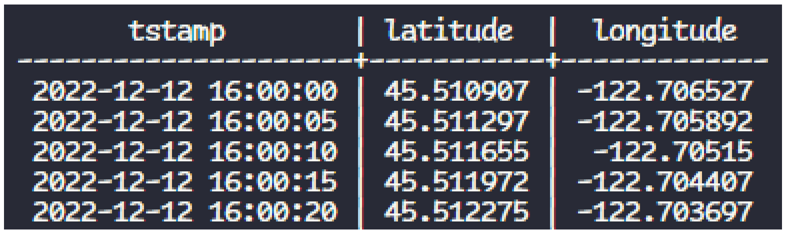
```
SELECT tstamp,
        latitude,
        longitude
FROM breadcrumb
WHERE EXTRACT(DOW FROM tstamp) = 1
  AND EXTRACT(HOUR FROM tstamp) BETWEEN 16 AND 17
  AND latitude BETWEEN 45.506022 AND 45.516636
  AND longitude BETWEEN -122.711662 AND -122.700316
ORDER BY tstamp ASC;
```

Sunday between 6am and 7am

```
SELECT tstamp,
        latitude,
        longitude
FROM breadcrumb
WHERE EXTRACT(DOW FROM tstamp) = 0
  AND EXTRACT(HOUR FROM tstamp) BETWEEN 6 AND 7
  AND latitude BETWEEN 45.506022 AND 45.516636
  AND longitude BETWEEN -122.711662 AND -122.700316
ORDER BY tstamp ASC;
```

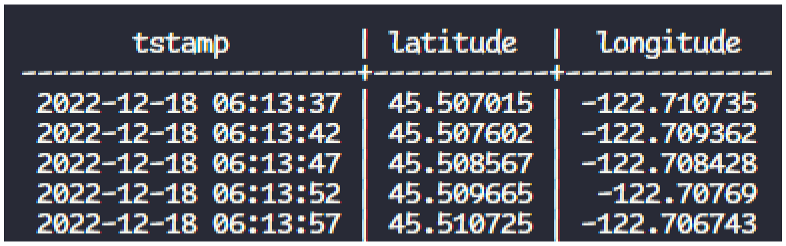
Answer:

Monday between 4pm and 6pm



```
       tstamp        | latitude  |  longitude
---------------------+-----------+-------------
 2022-12-12 16:00:00 | 45.510907 | -122.706527
 2022-12-12 16:00:05 | 45.511297 | -122.705892
 2022-12-12 16:00:10 | 45.511655 |  -122.70515
 2022-12-12 16:00:15 | 45.511972 | -122.704407
 2022-12-12 16:00:20 | 45.512275 | -122.703697
```

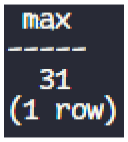
Sunday between 6am and 7am



```
       tstamp        | latitude  |  longitude
---------------------+-----------+-------------
 2022-12-18 06:13:37 | 45.507015 | -122.710735
 2022-12-18 06:13:42 | 45.507602 | -122.709362
 2022-12-18 06:13:47 | 45.508567 | -122.708428
 2022-12-18 06:13:52 | 45.509665 |  -122.70769
 2022-12-18 06:13:57 | 45.510725 | -122.706743
```

6. What is the maximum speed reached by any bus in the system?
   Query:
   select MAX(speed) from breadcrumb;

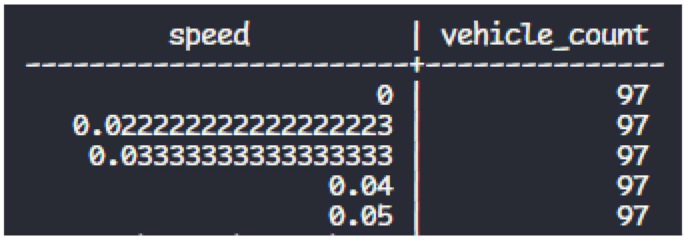
   Answer:



```
 max
-----
  31
(1 row)
```

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.
   Query:
   SELECT b.speed, COUNT(DISTINCT t.vehicle_id) AS vehicle_count
   FROM breadcrumb b
   JOIN trip t ON b.trip_id = t.trip_id
   GROUP BY b.speed
   ORDER BY vehicle_count DESC, b.speed;

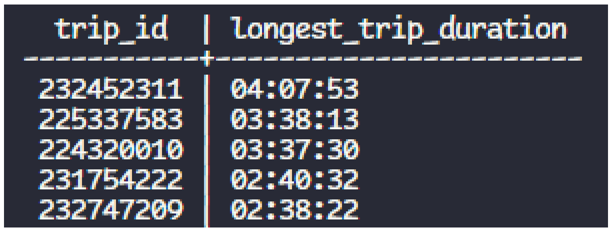
   Answer:



```
         speed          | vehicle_count
------------------------+---------------
                      0 |            97
   0.022222222222222223 |            97
    0.03333333333333333 |            97
                   0.04 |            97
                   0.05 |            97
```

8. Which is the longest (in terms of time) trip of all trips in the data?
   Query:
   SELECT trip_id, MAX(end_time - start_time) AS longest_trip_duration
   FROM (SELECT trip_id, MIN(tstamp) AS start_time, MAX(tstamp) AS end_time FROM breadcrumb GROUP BY trip_id) AS trip_times
   GROUP BY
   trip_id
   ORDER BY
   longest_trip_duration DESC;

   Answer:

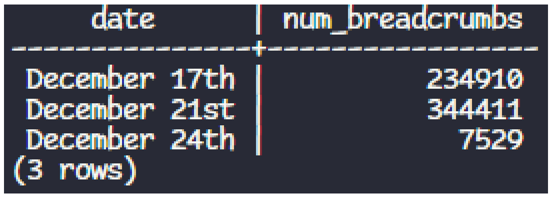


```
  trip_id  | longest_trip_duration
-----------+-----------------------
 232452311 | 04:07:53
 225337583 | 03:38:13
 224320010 | 03:37:30
 231754222 | 02:40:32
 232747209 | 02:38:22
```

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?
   Query:

```
SELECT CASE
      WHEN DATE(tstamp) = '2022-12-17' THEN 'December 17th'
      WHEN DATE(tstamp) = '2022-12-21' THEN 'December 21st'
      WHEN DATE(tstamp) = '2022-12-24' THEN 'December 24th'
  END AS date,
  COUNT(*) AS num_breadcrumbs
FROM breadcrumb WHERE
  DATE(tstamp) IN ('2022-12-17', '2022-12-21', '2022-12-24')
GROUP BY date
ORDER BY date;
```

   Answer: Unfortunately, because of something happening with the API on the day of 04-23, the data is incomplete. This is the result from the query.



```
     date      | num_breadcrumbs
---------------+-----------------
 December 17th |          234910
 December 21st |          344411
 December 24th |            7529
(3 rows)
```

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).
    Queries:

1) How many buses ran during the day on every January 3rd 2023, versus the evening. Hours from 5am to 3pm, then 4pm to 11:59 pm same day.

```
SELECT CASE
      WHEN EXTRACT(HOUR FROM tstamp) BETWEEN 5 AND 15 THEN 'Day'
      WHEN EXTRACT(HOUR FROM tstamp) BETWEEN 16 AND 23 THEN 'Evening'
  END AS time_period,
  COUNT(DISTINCT vehicle_id) AS num_buses
FROM trip t JOIN breadcrumb b ON t.trip_id = b.trip_id
WHERE
  DATE(tstamp) = '2023-01-03'
  AND EXTRACT(HOUR FROM tstamp) BETWEEN 5 AND 23
GROUP BY time_period ORDER BY time_period;
```

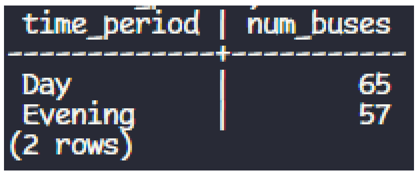
2) **Which had more trips? Even vehicle_id numbers or odd?**

```
SELECT
   CASE
      WHEN vehicle_id % 2 = 0 THEN 'Evens'
      ELSE 'Odds'
   END AS vehicle_category,
   COUNT(DISTINCT trip_id) AS num_trips
FROM trip GROUP BY vehicle_category
ORDER BY num_trips DESC;
```

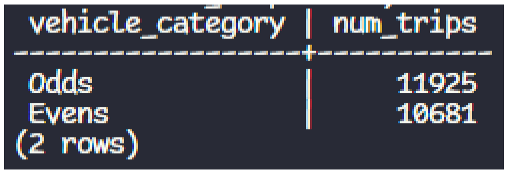
3) **Which days were the busiest based on the number of trips taken?**

```
SELECT DATE(tstamp) AS day, COUNT(DISTINCT trip_id) AS num_trips
FROM breadcrumb
GROUP BY day
ORDER BY num_trips DESC
```

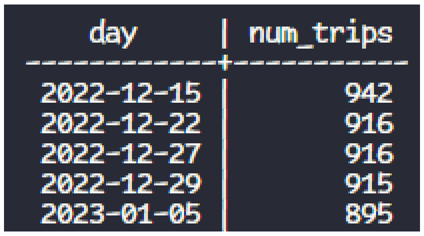
Answers:

1) 

```
 time_period | num_buses
-------------+-----------
 Day         |        65
 Evening     |        57
(2 rows)
```

2) 

```
 vehicle_category | num_trips
------------------+-----------
 Odds             |     11925
 Evens            |     10681
(2 rows)
```

3) 

```
    day     | num_trips
------------+-----------
 2022-12-15 |       942
 2022-12-22 |       916
 2022-12-27 |       916
 2022-12-29 |       915
 2023-01-05 |       895
```

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).

Here is a link to my code: https://github.com/NAlexH2/de-proj-cs510
You can find milestone02 in `‘milestone-submissions/milestone02’`