DataEng Project - Assignment 2 Submission Document

Chethana Muppalam
Srivashinie Dhamodharasamy
Vaishnavi Srinath

Github Link:
https://github.com/chethana613/TriMet.git

Tracking Table:

Date	Day of Week	# Sensor Readings	# rows added to your database
5/02/2024	Thursday	248962	248537
5/03/2024	Friday	341271	341066
5/04/2024	Saturday	374521	371604
5/05/2024	Sunday	382362	382271
5/06/2024	Monday	402253	396824
5/07/2024	Tuesday	256026	246272
5/08/2024	Wednesday	231207	230329
5/09/2024	Thursday	352703	308179
5/10/2024	Friday	348458	349325
5/11/2024	Saturday	332783	331326
5/12/2024	Sunday	292450	292437

Documentation of Each of the Original Data Fields:

	Column Name	Type	Documentation
1	EVENT_NO_TRIP	Numeric	Trip number is a 9 digit integer value that uniquely identifies a particular trip.
2	EVENT_NO_STOP	Numeric	Stop number is a 9 digit integer value that identifies a particular stop.
3	OPD_DATE	Timestamp	Operation date specifies the date in which the breadcrumb data was recorded.
4	VEHICLE_ID	Numeric	Vehicle Id is a unique 4 digit identification number for a vehicle. There are 100 vehicle IDs in the data.
5	METERS	Numeric	Meters is the odometer reading at a particular time. It is measured in meters.
6	ACT_TIME	Numeric	Actual time is the elapsed time from midnight of that day. It is measured in seconds. It can be from 0 to 86340.
7	GPS_LONGITUDE	Numeric	Latitude indicates the latitude coordinate of the vehicle at a particular time. The longitude is in the range of -124.0 and -122.0.
8	GPS_LATITUDE	Numeric	Longitude indicates the longitude coordinate of the vehicle at a particular time. The latitude is in the range of 45.2 and 45.7.
9	GPS_SATELLITES	Numeric	This is the count of satellites capturing the location of the vehicle. The total number of satellites are in the range of 0 and 12.
10	GPS_HDOP	Numeric	Horizontal Dilution of Precision is used to quantify the error in the horizontal position, latitude and longitude, of a receiver. It is of the range 0 and 25.5.

Data Validation Assertions:

(The first 10 assertions have been implemented in the receiver code)

1. Every breadcrumb record must have a trip number. - Existence assertion
2. Every trip must have a stop number. - Existence assertion
3. Every trip record must have an operation date and actual time. - Existence assertion
4. Every trip record must have meters value - Intra record assertion
5. The latitude values must be between 45.2 and 45.7. - Limit assertion
6. The longitude values must be between -124.0 and -122.0. - Limit assertion
7. Meters cannot have a negative value. - Limit assertion
8. For any particular vehicle ID, the combination of trip number, stop number, actual time and meters is unique. - Intra-record assertion
9. Every breadcrumb that has a latitude value must also have a longitude value - Intra-record assertion
10. For a particular day, all trip records will have the same operation date. - Summary assertion
11. The actual time or the elapsed time(ACT_TIME) must be between 0 and 86340. - Limit assertion
12. Every breadcrumb record must have a Vehicle ID. - Existence assertion
13. The total number of satellites is between 0 and 12. - Limit assertion
14. The GPS_HDOP value is in the range 0 and 26. - Limit assertion
15. For each vehicle ID, The meters are exponentially distributed over the trip number. - Distribution/statistical assertions
16. GPS_HDOP is inversely proportional to the count of satellites. - Distribution/statistical assertions
17. For any trip, As the meters increase the time must also increase. - Inter-record assertions
18. Every trip number and stop number is serviced by only one vehicle ID. - Inter-record assertion
19. The stop number is always greater than the trip number for every trip. - Inter-record assertion
20. Every trip ID is known for vehicle route - Referential integrity assertion

Data Transformations:

- For the assertion violation regarding missing latitude and longitude values, we first insert the empty latitude and longitude values into the temporary CSV file. Next, we group the records by trip ID and sort them based on the timestamp. Then, we interpolate the missing latitude and longitude values based on the subsequent values in the dataset.
- If the value of ACT_TIME exceeds 86340, the total number of seconds in a day, the date of timestamp needs to be adjusted to the next day before adding the ACT_TIME value.
- Timestamp is calculated by adding the number of seconds from the ACT_TIME to the date in the OPD_DATE.
- To compute the speed, we first saved all the records in a temporary CSV file. Then, we grouped them based on the trip number and sorted them by timestamp. After that, we iterate through each record within a trip, calculating the speed by comparing each record with its previous breadcrumb and considering the time and meters value.
- We dropped columns that are not necessary for our analysis, such as GPS_SATELLITES and GPS_HDOP. Since we had already derived the timestamp from the OPD_DATE and ACT_TIME columns, we removed these columns as well. The meters column was also dropped as it was no longer needed after calculating the speed.

Example Queries:

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

Query:

select count(*) from breadcrumb where date(tstamp) = '2023-01-01';

Screenshot:

```
postgres=# select count(*) from breadcrumb where date(tstamp) = '2023-01-01';
 count
--------
 223514
(1 row)
```

```
postgres=# select * from breadcrumb where date(tstamp) = '2023-01-01' limit 5;
       tstamp        | latitude  |  longitude  |       speed        |  trip_id
---------------------+-----------+-------------+--------------------+-----------
 2023-01-01 05:59:42 | 45.504407 | -122.843917 |                4.2 | 229979440
 2023-01-01 05:59:47 | 45.504583 | -122.843722 |                4.2 | 229979440
 2023-01-01 06:00:22 | 45.504747 |  -122.84358 | 0.6571428571428571 | 229979440
 2023-01-01 06:00:27 | 45.504977 | -122.843433 |                5.8 | 229979440
 2023-01-01 06:00:32 | 45.505127 | -122.843377 |                2.4 | 229979440
(5 rows)
```

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

Query:

select count(*) from breadcrumb where date(tstamp) = '2023-01-02';

Screenshot:

```
postgres=# select count(*) from breadcrumb where date(tstamp) = '2023-01-02';
 count
--------
 248537
(1 row)
```

```
postgres=# select * from breadcrumb where date(tstamp) = '2023-01-02' limit 5;
       tstamp        | latitude  |  longitude  | speed |  trip_id
---------------------+-----------+-------------+-------+-----------
 2023-01-02 00:00:03 |  45.41891 | -122.743597 |  24.6 | 230002538
 2023-01-02 00:00:08 | 45.419685 | -122.743527 |    17 | 230002538
 2023-01-02 00:00:13 | 45.420247 | -122.743843 |  13.2 | 230002538
 2023-01-02 00:00:18 | 45.420653 | -122.744507 |  13.6 | 230002538
 2023-01-02 00:00:23 | 45.421035 | -122.745227 |    14 | 230002538
(5 rows)
```

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

Query:

select extract(dow from timestamp) as day_of_the_week, count(*) as reading_count from breadcrumb group by extract(dow from timestamp) order by extract(dow from timestamp);

Screenshot:

```
postgres=# select extract(dow from tstamp) as day_of_the_week, count(*) as reading_count from breadcrumb group by extract(dow from tst-
amp) order by extract(dow from tstamp);
 day_of_the_week | reading_count
-----------------+---------------
               0 |        666843
               1 |       1656718
               2 |        711991
               3 |        711627
               4 |        743543
               5 |        745124
               6 |        252218
(7 rows)
```

Average count:

```
postgres=# select extract(dow from tstamp) as day_of_the_week,
count(*)::float / count(distinct date(tstamp)) as average
from      breadcrumb group by extract(dow from tstamp)
order by extract(dow from tstamp);
 day_of_the_week |      average
-----------------+--------------------
               0 |           333421.5
               1 |             828358
               2 |           355995.5
               3 |             237209
               4 | 248076.33333333334
               5 | 249739.33333333334
               6 |           126108.5
(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 t.trip_id
from breadcrumb b
join trip t on b.trip_id = t.trip_id
where date(b.tstamp) = '2023-01-01'
and b.latitude >= 45.497805 and b.latitude <= 45.504025
and b.longitude >= -122.566576 and b.longitude <= -122.563187 limit 5;
```

Screenshot:

```
postgres=# select distinct t.trip_id
from breadcrumb b
join trip t on b.trip_id = t.trip_id
where date(b.tstamp) = '2023-01-01'
  and b.latitude >= 45.497805 and b.latitude <= 45.504025
  and b.longitude >= -122.566576 and b.longitude <= -122.563187 limit 5;
  trip_id
-----------
 229991566
 230039164
 230131668
 230185095
 230498065
(5 rows)
```

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:

```
select *
from breadcrumb
where latitude >= 45.506022 and latitude <= 45.516636
  and longitude >= -122.711662 and longitude <= -122.700316
  and extract(dow from tstamp) = 1
  and extract(hour from tstamp) >= 16 and extract(hour from tstamp) < 18
order by tstamp limit 5;
```

Screenshot:

```
postgres=# select *
from breadcrumb
where latitude >= 45.506022 and latitude <= 45.516636
  and longitude >= -122.711662 and longitude <= -122.700316
  and extract(dow from tstamp) = 1
  and extract(hour from tstamp) >= 16 and extract(hour from tstamp) < 18
order by tstamp limit 5;
       tstamp        | latitude  |  longitude  |       speed       |  trip_id
---------------------+-----------+-------------+-------------------+-----------
 2023-01-09 16:03:52 | 45.508695 | -122.700522 |               8.4 | 235088549
 2023-01-09 16:03:57 | 45.508663 | -122.701143 |               9.8 | 235088549
 2023-01-09 16:03:57 | 45.508663 | -122.701143 | 4.927272727272728 | 235088549
 2023-01-09 16:04:02 | 45.508578 | -122.701877 |              11.8 | 235088549
 2023-01-09 16:04:02 | 45.508578 | -122.701877 |                 0 | 235088549
(5 rows)
```

Query:

```
select count(*)
from breadcrumb
where latitude >= 45.506022 and latitude <= 45.516636
 and longitude >= -122.711662 and longitude <= -122.700316
 and extract(dow from tstamp) = 1
 and extract(hour from tstamp) >= 16 and extract(hour from tstamp) < 18;
```

Screenshot:

```
postgres=# select count(*)
from breadcrumb
where latitude >= 45.506022 and latitude <= 45.516636
  and longitude >= -122.711662 and longitude <= -122.700316
  and extract(dow from tstamp) = 1
  and extract(hour from tstamp) >= 16 and extract(hour from tstamp) < 18;
 count
-------
   430
(1 row)
```

Query:

```
select *
from breadcrumb
where latitude >= 45.506022 and latitude <= 45.516636
 and longitude >= -122.711662 and longitude <= -122.700316
 and extract(dow from tstamp) = 0
 and extract(hour from tstamp) >= 6 and extract(hour from tstamp) < 8
order by tstamp limit 5;
```

Screenshot:

```
postgres=# select *
from breadcrumb
where latitude >= 45.506022 and latitude <= 45.516636
  and longitude >= -122.711662 and longitude <= -122.700316
  and extract(dow from tstamp) = 0
  and extract(hour from tstamp) >= 6 and extract(hour from tstamp) < 8
order by tstamp limit 5;
       tstamp        | latitude  |  longitude  | speed |  trip_id
---------------------+-----------+-------------+-------+-----------
 2023-01-01 06:00:16 | 45.507005 | -122.711178 |  23.2 | 230045092
 2023-01-01 06:00:21 | 45.507425 | -122.709808 |  23.4 | 230045092
 2023-01-01 06:00:26 | 45.508192 | -122.708822 |  22.8 | 230045092
 2023-01-01 06:00:31 | 45.509147 | -122.708142 |  23.6 | 230045092
 2023-01-01 06:00:36 |  45.51007 | -122.707495 |    23 | 230045092
(5 rows)
```

Query:

```
select count(*)
from breadcrumb
where latitude >= 45.506022 and latitude <= 45.516636
 and longitude >= -122.711662 and longitude <= -122.700316
 and extract(dow from tstamp) = 0
 and extract(hour from tstamp) >= 6 and extract(hour from tstamp) < 8;
```

Screenshot:

```
postgres=# select count(*)
from breadcrumb
where latitude >= 45.506022 and latitude <= 45.516636
  and longitude >= -122.711662 and longitude <= -122.700316
  and extract(dow from tstamp) = 0
  and extract(hour from tstamp) >= 6 and extract(hour from tstamp) < 8;
 count
-------
    63
(1 row)
```

From the above results we can observe that the total number of trips on Monday at 4pm to 6pm is higher than the number of trips on Sunday Morning 6am to 8am.

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

Query:

```
select max(speed) as max_speed from breadcrumb;
```

Screenshot:

```
postgres=# select max(speed) as max_speed from breadcrumb;
     max_speed
-------------------
 96.85714285714286
(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 num_vehicles
from breadcrumb b
inner join trip t on b.trip_id = t.trip_id
group by b.speed
order by num_vehicles desc limit 5;
```

Screenshot:

```
postgres=# select b.speed, count(distinct t.vehicle_id) as num_vehicles
from breadcrumb b
inner join trip t on b.trip_id = t.trip_id
group by b.speed
order by num_vehicles desc limit 5;
        speed        | num_vehicles
---------------------+--------------
                0.05 |           95
 0.06666666666666667 |           95
                   0 |           95
                0.04 |           95
                0.08 |           95
(5 rows)
```

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

Query:

```
select trip_id, max(tstamp) - min(tstamp) as trip_duration
from breadcrumb
group by trip_id
order by trip_duration desc
limit 1;
```

Screenshot:

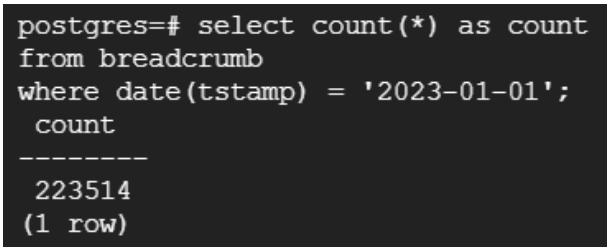
```
postgres=# select trip_id, max(tstamp) - min(tstamp) as trip_duration
from breadcrumb
group by trip_id
order by trip_duration desc
limit 1;
  trip_id  | trip_duration
-----------+---------------
 235818403 | 02:37:16
(1 row)
```

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: (Holiday)

```
select count(*) as count
from breadcrumb
where date(tstamp) = '2023-01-01';
```

Screenshot:

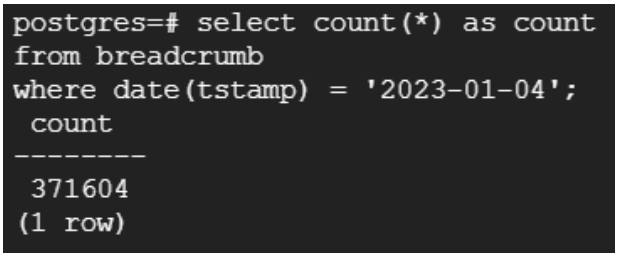


```
postgres=# select count(*) as count
from breadcrumb
where date(tstamp) = '2023-01-01';
 count
--------
 223514
(1 row)
```

Query: (Non-holiday Wednesday)

```
select count(*) as count
from breadcrumb
where date(tstamp) = '2023-01-04';
```

Screenshot:

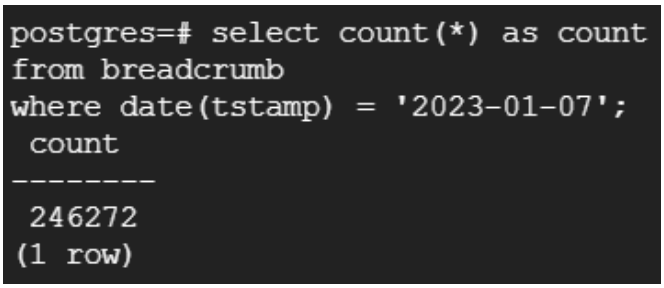


```
postgres=# select count(*) as count
from breadcrumb
where date(tstamp) = '2023-01-04';
 count
--------
 371604
(1 row)
```

Query:(Non-holiday Wednesday)

```
select count(*) as count
from breadcrumb
where date(tstamp) = '2023-01-07';
```

Screenshot:



```
postgres=# select count(*) as count
from breadcrumb
where date(tstamp) = '2023-01-07';
 count
--------
 246272
(1 row)
```

From the results we can observe that the total trips on a Holiday is less than the total number of trips on a Non-Holiday. Also we can observe that the total number of trips on a non holiday weekday is higher than the trips on a non holiday saturday.

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).

Question 1: Which vehicle has the most number of trips?

Query:

```
select vehicle_id, count(distinct trip_id) as num_trips
from trip
group by vehicle_id
order by num_trips desc
limit 1;
```

Screenshot:

```
postgres=# select vehicle_id, count(distinct trip_id) as num_trips
from trip
group by vehicle_id
order by num_trips desc
limit 1;
 vehicle_id | num_trips
------------+-----------
       3019 |       193
(1 row)
```

Question 2: Which date has the most number of trips?

Query:

```
select date_trunc('day', tstamp) as trip_date, count(distinct trip_id) as num_trips
from breadcrumb
group by trip_date
order by num_trips desc
limit 1;
```

Screenshot:

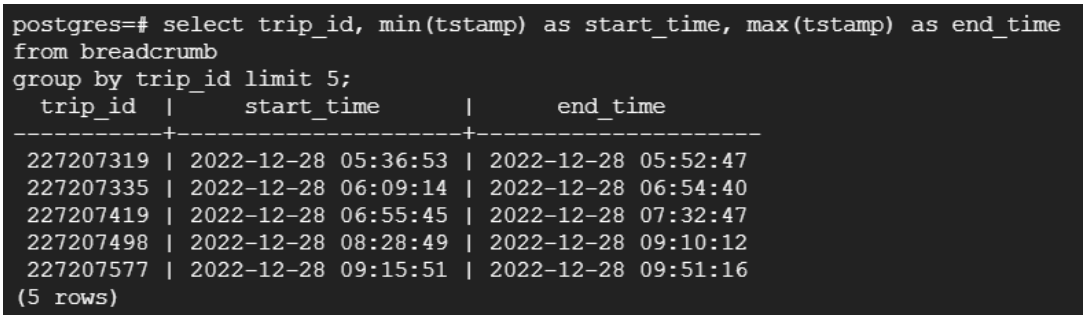
```
postgres=# select date_trunc('day', tstamp) as trip_date, count(distinct trip_id) as num_trips
from breadcrumb
group by trip_date
order by num_trips desc
limit 1;
      trip_date      | num_trips
---------------------+-----------
 2023-01-06 00:00:00 |      1068
(1 row)
```

Question 3:What is the starting and ending time of a trip?

Query:

```
select trip_id, min(tstamp) as start_time, max(tstamp) as end_time
from breadcrumb
group by trip_id limit 5;
```

Screenshot:



```
postgres=# select trip_id, min(tstamp) as start_time, max(tstamp) as end_time
from breadcrumb
group by trip_id limit 5;
  trip_id  |     start_time      |      end_time
-----------+---------------------+---------------------
 227207319 | 2022-12-28 05:36:53 | 2022-12-28 05:52:47
 227207335 | 2022-12-28 06:09:14 | 2022-12-28 06:54:40
 227207419 | 2022-12-28 06:55:45 | 2022-12-28 07:32:47
 227207498 | 2022-12-28 08:28:49 | 2022-12-28 09:10:12
 227207577 | 2022-12-28 09:15:51 | 2022-12-28 09:51:16
(5 rows)
```