Data Engineering - Project - Part2

Sri Lakshmi Polavarapu (srilakp@pdx.ed) - 914559661

GitHub link:

https://github.com/SriLakshmiPolavarapu/Trimet-Bytes--Data-Engineering-Project

A. Review the data

	Column Name	Type	Documentation
1.	EVENT_NO_TRIP	Numeric	It is a 9 digit unique integer value which identifies a particular trip.
2.	EVENT_NO_STOP	Numeric	It is a 9 digit integer value that defines a particular stop.
3.	OPD_DATE	Timestamp	Operation date is the type of date which tells when breadcrumb data was recorded. It provides a precise timestamp for each event.
4.	VEHICLE_ID	Numeric	It is the 4 digit unique number/ID for the vehicle.
5.	METERS	Numeric	This field represents the distance covered by the vehicle in meters.
6.	ACT_TIME	Numeric	Actual data is the elapsed time from midnight of that day, which is measured in seconds. It ranges from 0 to 86340.
7.	GPS_LONGITUDE	Numeric	Longitude indicates the longitude coordinate of the vehicle at a particular time.
8.	GPS_LATITUDE	Numeric	Latitude indicates the latitude coordinate of the vehicle at a particular time.
9.	GPS_SATELLITES	Numeric	This indicated the count of satellites capturing the location of the vehicle. The range of satellite count is 0 to 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.

B. Data validation

Existence:

1. There are no duplicate ID's
2. Every vehicle ID contains 4 digits
3. Every breadcrumb must have a trip number
4. The ACT_TIME of each subsequent entry is greater than the ACT_VALUE of previous entries of the same Vehicle ID
5. If meters change then ACT_TIME should also be same

Limit:

1. GPS_LONGITUDE values within the range [-180, 180]
2. GPS_LATITUDE values within the range [-90, 90]
3. GPS_HDOP values must be less than 2
4. GPS_LONGITUDE must have a negative value and GPS_LATITUDE should have a positive value
5. Satellite values are within the range [0, 12]

Intra-record check:

1. For any Vehicle ID the combination between trip number, stop number, and meters are unique
2. Every breadcrumb which has latitude value must have longitude value

Inter-record check

1. Stop number is always greater than the trio number
2. For every breadcrumb, as the meters increase time will also be increasing

Summary:

1. If we calculate speed of vehicles for each day of a week and we can check if there are any difference between weekends and weekdays
2. Check the distribution of GPS_HDOP and check id they are distributed evenly

Referential Integrity:

1. Check if the vehicle ID's in the dataset exists in any other database
2. Every ID is known for the vehicle route

Distribution/statistical:

1. When we calculate the speed of vehicles and check if we can see any patterns
2. We can check for average distance traveled by each vehicle and check for any patterns

D. Data Transformation

1. Create timestamp: Combined OPD_DATE and ACT_TIME, into a timestamp field, as it is easy when using postgreSQL for queries
2. Calculate speed: calculate speed using the change in meters and ACT_TIME of each breadcrumb. This transformation ensures the accuracy of speed calculations and handles edge cases such as the first record of each trip and negative speeds.
3. Fill in the missing values: replace both GPS_LONGITUDE and GPS_LATITUDE with 0.0 to indicate unknown locations. Filling them with zeros helps us to include these records in the dataset while indicating that the coordinates are unknown.
4. We can drop columns that are not necessary like GPS_HDOP. As we already calculated the timestamp, we no longer need OPD_DATE and ACT_TIME.

QUERIES

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

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

```
trimet_data=> select count(*) from breadcrumb where date(tstamp)='2023-01-01';
 count
--------
 246033
(1 row)

trimet_data=>
```

select * from breadcrumb where date(tstamp)='2023-01-01' limit 3;

```
trimet_data=> select * from breadcrumb where date(tstamp) = '2023-01-01' limit 3;
      tstamp        | latitude  |  longitude  | speed |  trip_id
--------------------+-----------+-------------+-------+-----------
 2023-01-01 09:29:14 | 45.501588 | -122.656437 |  14.1 | 231769742
 2023-01-01 09:29:19 | 45.501595 | -122.657493 |  16.4 | 231769742
 2023-01-01 09:29:24 | 45.501605 | -122.658618 |  17.2 | 231769742
(3 rows)

trimet_data=>
```

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

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

```
trimet_data=> select count(*) from breadcrumb where date(tstamp)='2023-01-02';
 count
--------
 235763
(1 row)

trimet_data=>
```

select * from breadcrumb where date(tstamp)='2023-01-02' limit 3;

```
trimet_data=> select * from breadcrumb where date(tstamp) = '2023-01-02' limit 3;
      tstamp        | latitude  |  longitude  | speed |  trip_id
--------------------+-----------+-------------+-------+-----------
 2023-01-02 11:54:47 |  45.49076 | -122.477258 |   8.4 | 230258541
 2023-01-02 11:54:52 | 45.490647 | -122.476717 |   8.4 | 230258541
 2023-01-02 11:54:57 | 45.490607 |  -122.47655 |   2.2 | 230258541
(3 rows)

trimet_data=>
```

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

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

```
trimet_data=> select extract(dow from tstamp) as day_of_the_week, count(*) as reading_count from breadcrumb group by extract(dow from tstamp) order by extract(dow from tstamp);
 day_of_the_week | reading_count
-----------------+---------------
               0 |        489054
               1 |        235763
               2 |        349883
               3 |        389835
               4 |        669400
               5 |        712691
               6 |        494748
(7 rows)
```

select dow, cnt, avg(cnt) over(order by dow) run_avg from (select extract(dow from tstamp) dow, count(*) cnt from breadcrumb group by 1) t order by dow;

```
trimet_data=> select dow, cnt, avg(cnt) over(order by dow) run_avg from (select extract(dow from tstamp) dow, count(*) cnt from breadcrumb group by 1) t order by dow;
 dow |  cnt   |       run_avg
-----+--------+---------------------
   0 | 489054 | 489054.000000000000
   1 | 235763 | 362408.500000000000
   2 | 349883 | 358233.333333333333
   3 | 389835 | 366133.750000000000
   4 | 669400 | 426787.000000000000
   5 | 712691 | 474437.666666666667
   6 | 494748 | 477339.142857142857
(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)].

select distinct t.trip_id from breadcrumb as b join trip as t using (trip_id) where b.tstamp::date = '2023-01-01' and b.latitude between 45.497805 and 45.504025 and b.longitude between -122.566576 and -122.563187 order by t.trip_id limit 5;

```
trimet_data=> SELECT DISTINCT
  t.trip_id
FROM breadcrumb AS b
JOIN trip AS t USING (trip_id)
WHERE b.tstamp::date = '2023-01-01'
  AND b.latitude  BETWEEN 45.497805 AND 45.504025
  AND b.longitude BETWEEN -122.566576 AND -122.563187
ORDER BY t.trip_id limit 5;
  trip_id
-----------
 229816196
 229816838
 229857293
 229991566
 230011876
(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?

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;

```
trimet_data=> 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
---------------------+-----------+-------------+-------+-----------
 2023-01-02 17:16:23 | 45.515187 | -122.700512 |  19.2 | 230522280
 2023-01-02 17:16:28 | 45.514428 | -122.701188 |  19.8 | 230522280
 2023-01-02 17:16:33 | 45.513683 | -122.701895 |  19.6 | 230522280
 2023-01-02 17:16:38 | 45.513012 | -122.702732 |  19.6 | 230522280
 2023-01-02 17:16:43 | 45.512488 |  -122.70374 |  19.4 | 230522280
(5 rows)
```

```
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;
```

```
trimet_data=> 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
-------
    14
(1 row)
```

```
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;
```

```
trimet_data=> 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 06:02:46 | 45.507053 | -122.710712 |    21 | 230091000
 2023-01-01 06:02:51 | 45.507497 | -122.709527 |    21 | 230091000
 2023-01-01 06:02:56 | 45.508272 | -122.708657 |  21.8 | 230091000
 2023-01-01 06:03:01 | 45.509197 |  -122.70802 |    23 | 230091000
 2023-01-01 06:03:06 | 45.510138 | -122.707352 |  23.4 | 230091000
(5 rows)
```

```
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;
```

```
trimet_data=> 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
-------
    48
(1 row)
```

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

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

```
trimet_data=> select max(speed) as max_speed from breadcrumb;
 max_speed
-----------
      1097
(1 row)

trimet_data=>
```

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 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;
```

```
trimet_data=> 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
-------+--------------
   0.5 |          190
     1 |          190
     0 |          190
   0.2 |          190
   1.2 |          190
(5 rows)
```

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;

```
trimet_data=> 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
-----------+---------------
 227610474 | 09:08:20
(1 row)

trimet_data=>
```

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?

Holiday

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

```
trimet_data=> select count(*) as count
from breadcrumb
where date(tstamp) = '2023-01-01';
 count
--------
 246033
(1 row)

trimet_data=>
```

Non - holiday Wednesday

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

```
trimet_data=> select count(*) as count
from breadcrumb
where date(tstamp) = '2023-01-04';
 count
--------
 364223
(1 row)

trimet_data=>
```

Non - holiday Saturday

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

```
trimet_data=> select count(*) as count
from breadcrumb
where date(tstamp) = '2023-01-07';
 count
--------
 238762
(1 row)

trimet_data=>
```

From the above 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 less 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).

Query -1: Which two vehicles have the maximum number of trips?

select vehicle_id, count(distinct trip_id) as tot_trips from trip group by vehicle_id order by tot_trips desc limit 2;

```
trimet_data=> select vehicle_id, count(distinct trip_id) as tot_trips
from trip group by vehicle_id order by tot_trips desc limit 2;
 vehicle_id | tot_trips
------------+-----------
       3043 |       171
       3003 |       150
(2 rows)

trimet_data=>
```

Query - 2: List the top 5 dates with the highest number of distinct trips, and include each date's weekday.

select date(tstamp) as trip_date, to_char(tstamp, 'fmday') as day_of_week,count(distinct trip_id) as num_trips
from breadcrumb group by date(tstamp), to_char(tstamp, 'fmday') order by num_trips desc
limit 5;

```
trimet_data=> select date(tstamp) as trip_date, to_char(tstamp, 'fmday') as day_of_week,count(distinct trip_id) as num_trips
from breadcrumb group by date(tstamp), to_char(tstamp, 'fmday') order by num_trips desc
limit 5;
 trip_date  | day_of_week | num_trips
------------+-------------+-----------
 2023-01-04 | wednesday   |      1825
 2023-01-06 | friday      |      1798
 2023-01-03 | tuesday     |      1796
 2023-01-05 | thursday    |      1782
 2022-12-30 | friday      |      1724
(5 rows)
```

Query - 3: what is the starting and ending time of each tripID?

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

```
trimet_data=> 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
-----------+---------------------+---------------------
 227193715 | 2022-12-28 13:50:44 | 2022-12-28 14:32:13
 227193840 | 2022-12-28 14:56:40 | 2022-12-28 15:18:12
 227205166 | 2022-12-28 06:40:28 | 2022-12-28 06:42:18
 227205243 | 2022-12-28 07:27:33 | 2022-12-28 07:45:18
 227205318 | 2022-12-28 08:38:25 | 2022-12-28 08:41:45
(5 rows)

trimet_data=>
```

SELECT tstamp::date AS date, COUNT(*) AS reading_count
FROM breadcrumb
GROUP BY tstamp::date
ORDER BY tstamp::date ASC;

```
trimet_data=>
trimet_data=> SELECT tstamp::date AS date, COUNT(*) AS reading_count
FROM breadcrumb
GROUP BY tstamp::date
ORDER BY tstamp::date ASC;
    date    | reading_count
------------+---------------
 2022-12-28 |         25612
 2022-12-29 |        314284
 2022-12-30 |        359230
 2022-12-31 |        255986
 2023-01-01 |        246033
 2023-01-02 |        235763
 2023-01-03 |        349883
 2023-01-04 |        364223
 2023-01-05 |        355116
 2023-01-06 |        353461
 2023-01-07 |        238762
 2023-01-08 |        243021
(12 rows)
```

Tracking Table

Date	Day of Week	# Sensor Readings	# rows added to your database
21st Apr	Monday	248962	248537
22nd Apr	Tuesday	341271	341066
23rd Apr	Wednesday	374521	371604
24th Apr	Thursday	382362	382271
25th Apr	Friday	402254	396824
26th Apr	Saturday	256026	246272
27th Apr	Sunday	352703	308179
28th Apr	Monday	491148	415991
29th Apr	Tuesday	701972	313261
30th Apr	Wednesday	655912	24578
1st May	Thursday	646356	291349
2nd May	Friday	706173	387175
3rd May	Saturday	502717	257207
4th May	Sunday	491264	247278
5th May	Monday	465570	236901
6th May	Tuesday	686552	351679
7th May	Wednesday	718025	366048
8th May	Thursday	671006	356898
9th May	Friday	700667	355229
10th May	Saturday	455513	239904
11th May	Sunday	483508	244252

Till 27th April, my readings will be wrong, but from 28th April, I have corrected the code.