Jacob Crisan
Shengjia He
Alex Bailey
Aditya Sharoff

DataEng: Project Assignment 2 (v 2.0)

Validate, Transform, Enhance and Store

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	# updates/insertions into your database
2/10/2021 (2020-09-30)	Wednesday	370,452	259,638
2/11/2021 (2020-10-01)	Thursday	366,506	366,178
2/12/2021 (2020-10-02)	Friday	375,777	375,387
2/13/2021 (2020-10-03)	Saturday	176,418	176,269
2/14/2021 (2020-10-04)	Sunday	135,160	135,101

Query to Check how many breadcrumb entries were inserted into the database (in accordance with our assumptions) - we inserted each one individually:

```
SELECT COUNT(*)
FROM BreadCrumb
WHERE
    (tstamp::date='2020-10-04' AND cast(tstamp as time)>='02:00:00')
    OR
    (tstamp::date='2020-10-05' AND cast(tstamp as time)<='02:00:00');
```

Query to Check how many trip entries were inserted into the database (in accordance with our assumptions) - we inserted each one individually:

```
SELECT COUNT(DISTINCT Trip.trip_id)
FROM Trip, BreadCrumb
Where
        Trip.trip_id=BreadCrumb.trip_id AND
        (
            (tstamp::date='2020-10-04' AND cast(tstamp as time)>='02:00:00')
               OR
            (tstamp::date='2020-10-05' AND cast(tstamp as time)<='02:00:00')
        );
```

Documentation of Each of the Original Data Fields

For each of the fields of the bread crumb 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:

- A single run of the bus over a single route
- “All of the bread crumb readings sharing the same EVENT_NO_TRIP should be for the same vehicle, and they all correspond to that vehicle servicing a specific route”

EVENT_NO_STOP:

- Something with a meaning that is hard to discern
- Also, a field that is not needed.

OPD_DATE:

- The date that this data was recorded. It occurs in the year 2020
- It’s formatted as a string of the form: DD-MONTH[SEP< OCT, etc]-YY

VEHICLE_ID:

- An ID identifying the bus.
- It seems to be an integer
- How many vehicles there are is still yet unknown

METERS:

- The odometer of the bus in meters
- It doesn’t have to start at 0

ACT_TIME:

- The time of day for this bus routes in seconds from midnight, however it can go above 24 hours, only restarting when the bus routes reset
- Likely in the range 0-30 (haven’t verified, but we can test to make sure that range is correct)

VELOCITY:

- The speed of bus in meters/second
- Usually less than 30
- Never negative
- Can be empty

DIRECTION:

- direction the bus is facing in degrees from 0
- Between 0-360
- 0 is north?
- Can be blank

RADIO_QUALITY:

- The quality of the bus's onboard radio?
- It's always empty

GPS_LONGITUDE:

- The longitude from GPS
- Between -122 and -123

GPS_LATITUDE:

- the latitude from GPS
- Between 45 and 46

GPS_SATELLITES:

- The number of GPS satellites used to get this reading
- Almost exclusively between 7 and 12

GPS_HDOP:

- How accurate the GPS location is based on how spread out the satellites are
- Seems to be rated on some scale between 0 and 20, with smaller values being better and larger values being worse
- Almost exclusively between 0.7 and 1.4
- GPS system telling you how confident it is in the measurement

SCHEDULE_DEVIATION:

- How far behind or ahead the bus is with its schedule, in seconds. It is negative when early and positive when late
- Range: (-1000, 1000), and it can be null at times

Data Validation Assertions

List 20 or more data validation assertion statements here. These should be English language sentences similar to “The VELOCITY field exceeds 5000000”. 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 RADIO_QUALITY) that might not be used in your database schema.

1. **Inter-record/summary assertion:** Every EVENT_NO_TRIP, ACT_TIME combo should be unique
2. **Limit assertion:** Every EVENT_NO_TRIP should be above 140000000
3. **Limit assertion:** ACT_TIME should be in the range 0-93600 (0-26 hours)
4. **Existence assertion:** Every tuple must have an EVEN_NO_TRIP
5. **Existence assertion:** Every tuple must have a VEHICLE_ID
6. **Intra-record/referential integrity assertion:** Every GPS_LATITUDE must have a corresponding GPS_LONGITUDE - *intra-record/referential integrity assertion.*
7. **Limit assertion:** Every GPS_LATITUDE cannot exceed 47 and cannot be lower than 45
8. **Limit assertion:** DIRECTION cannot be lower than 0 and higher than 360
9. **Limit assertion:** If VELOCITY exists, it must be between 0 and 30 mps
10. **Statistical assertion:** At least 40% of tuples will have a VELOCITY between (5-15)

11. **Statistical assertion:** All the entries across one week must be uniform (since the bus schedule repeats itself, they should be similar to the data from the previous week, excluding holidays maybe)
12. **Existence assertions:** Every tuple must have a DIRECTION value
13. **Limit assertions:** GPS_HDOP should be between 0.5 and 25 inclusive
14. **Limit assertion:** GPS_SATELLITES should be between 3 and 17
15. **Existence assertion:** RADIO_QUALITY must be empty
16. **Existence assertion:** The month value for OPD_DATE must be a valid month
17. **Limit assertion:** The day value for OPD_DATE must be between 1-31
18. **Limit assertions:** RADIO_QUALITY must not exist
19. **Limit assertion:** EVENT_NO_STOP should be above 140000000
20. **Limit assertion:** Schedule deviation should be in the range (-2000, 2000), if it isn’t empty

We implemented validations 1 - 10

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.

- We dropped the RADIO_QUALITY column
  We dropped the EVENT_NO_STOP column
- We kicked out any tuple that violated our assertions
- We created separate entries for the BreadCrumb and Trip table for each entry and inserted them accordingly (except that the trip insertion would fail if the table already contained the trip data)

Example Queries

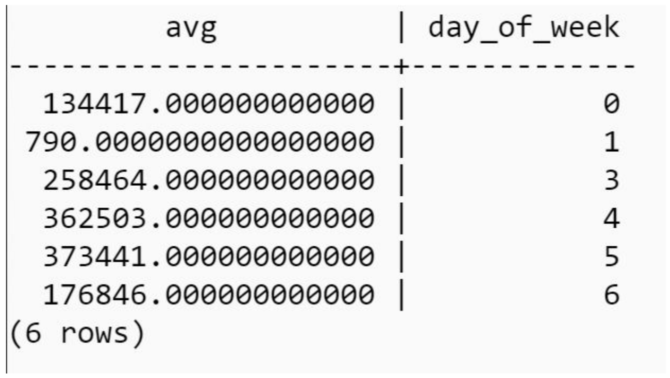
Provide your responses to the questions listed in Section E above. For each question, provide the SQL you used to answer the questions along with the count of the number of rows returned (where applicable) and a listing of the first 5 rows returned (where applicable).

How many vehicles are there in the C-Tran system?
Answer: 96
Query: SELECT COUNT(DISTINCT vehicle_id) FROM Trip;

How many bread crumb reading events occurred on October 2, 2020?
Answer: 373,441
Query: SELECT COUNT(*) FROM BreadCrumb WHERE tstamp::date = date '2020-10-02';
#
https://stackoverflow.com/questions/31433747/postgres-where-clause-compare-timestamp/31433748

How many bread crumb reading events occurred on October 3, 2020?
Answer: 176,846
Query: SELECT COUNT (*) FROM BreadCrumb WHERE tstamp::date = date '2020-10-3';

On average, how many bread crumb readings are collected on each day of the week?
Answer (we don't have that much data yet. 0 = Sunday, 6 = Saturday):



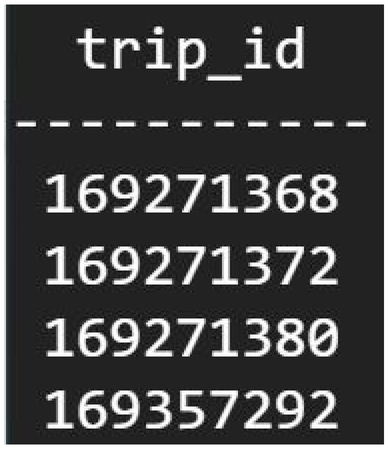
```
         avg           | day_of_week
-----------------------+-------------
  134417.000000000000  |           0
 790.0000000000000000  |           1
  258464.000000000000  |           3
  362503.000000000000  |           4
  373441.000000000000  |           5
  176846.000000000000  |           6
(6 rows)
```

Query:
```
SELECT
   AVG(num_breadcrumb_entries),
   day_of_week
FROM
```

```
    (SELECT COUNT(tstamp) as num_breadcrumb_entries,
                tstamp::date as date,
                EXTRACT(dow from tstamp::date) as day_of_week
     FROM BreadCrumb
     GROUP BY tstamp::date) as date_table
GROUP BY day_of_week
ORDER BY day_of_week ASC;
```

List the C-Tran trips that crossed the I-5 bridge on October 2, 2020. To find this, search for all trips that have bread crumb readings that occurred within a lat/lon bounding box such as [(45.620460, -122.677744), (45.615477, -122.673624)].

Answer:



```
  trip_id
-----------
 169271368
 169271372
 169271380
 169357292
```

266 rows

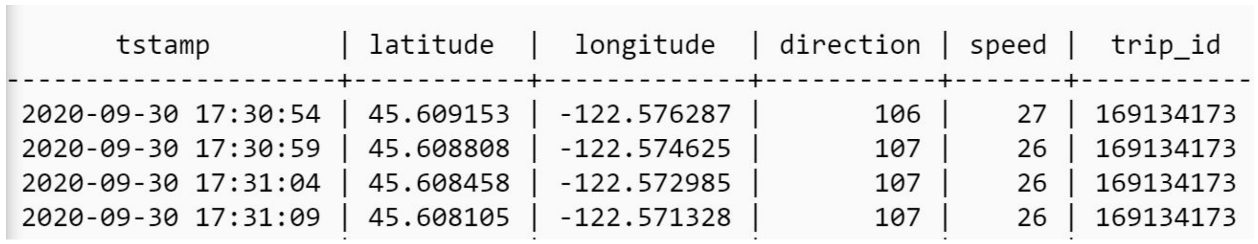
Query:

```
SELECT DISTINCT trip_id
From BreadCrumb
WHERE latitude >= 45.615477 AND
        latitude <= 45.620460 AND
        longitude <= -122.673624 AND
        longitude >= -122.677744 AND
        tstamp::date = date '2020-10-02';
```

List all bread crumb readings for a specific portion of Highway 14 (bounding box: [(45.610794, -122.576979), (45.606989, -122.569501)]) 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?

Answer (we don't have data for Monday and Sunday, so we used data for a Wednesday and Thursday):

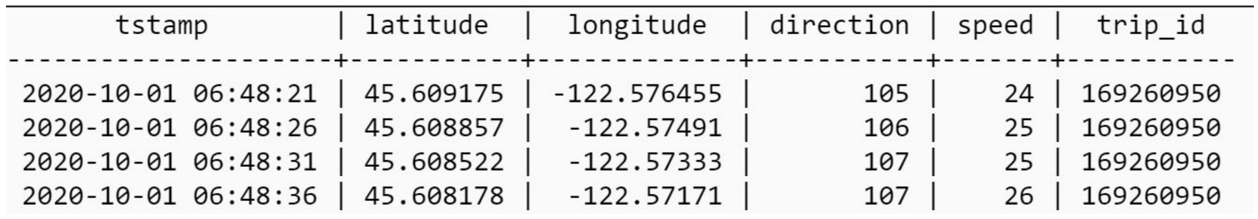
Wednesday:



```
       tstamp        | latitude  |  longitude  | direction | speed |  trip_id
---------------------+-----------+-------------+-----------+-------+-----------
 2020-09-30 17:30:54 | 45.609153 | -122.576287 |       106 |    27 | 169134173
 2020-09-30 17:30:59 | 45.608808 | -122.574625 |       107 |    26 | 169134173
 2020-09-30 17:31:04 | 45.608458 | -122.572985 |       107 |    26 | 169134173
 2020-09-30 17:31:09 | 45.608105 | -122.571328 |       107 |    26 | 169134173
```

40 rows

Thursday



```
       tstamp        | latitude  |  longitude  | direction | speed |  trip_id
---------------------+-----------+-------------+-----------+-------+-----------
 2020-10-01 06:48:21 | 45.609175 | -122.576455 |       105 |    24 | 169260950
 2020-10-01 06:48:26 | 45.608857 |  -122.57491 |       106 |    25 | 169260950
 2020-10-01 06:48:31 | 45.608522 |  -122.57333 |       107 |    25 | 169260950
 2020-10-01 06:48:36 | 45.608178 |  -122.57171 |       107 |    26 | 169260950
```

37 rows

It appears that the traffic on Wednesday and Thursday over highway 14 is pretty similar

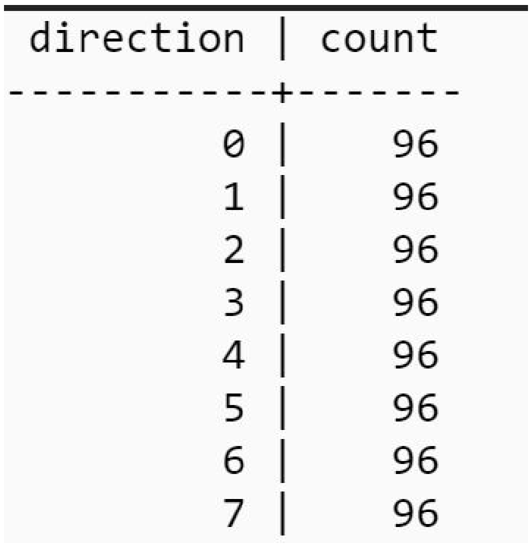
Queries:

```
SELECT *
FROM BreadCrumb
WHERE latitude >= 45.606989 AND
        latitude <= 45.610794 AND
        longitude >= -122.576979 AND
        longitude <= -122.569501 AND
        EXTRACT(dow from tstamp::date)=3 AND
        cast(tstamp as time) >= '16:00:00' AND
        cast(tstamp as time) <= '18:00:00';

SELECT *
FROM BreadCrumb
WHERE latitude >= 45.606989 AND
        latitude <= 45.610794 AND
        longitude >= -122.576979 AND
        longitude <= -122.569501 AND
        EXTRACT(dow from tstamp::date)=4 AND
        cast(tstamp as time) >= '06:00:00' AND
        cast(tstamp as time) <= '08:00:00';
```

What is the maximum velocity reached by any bus in the system?
Answer: 30 m/s
Query:
SELECT MAX(speed)
FROM BreadCrumb;

List all possible directions and give a count of the number of vehicles that faced precisely that direction during at least one trip. Sort the list by most frequent direction to least frequent.
Answer (we don't have that much data yet):

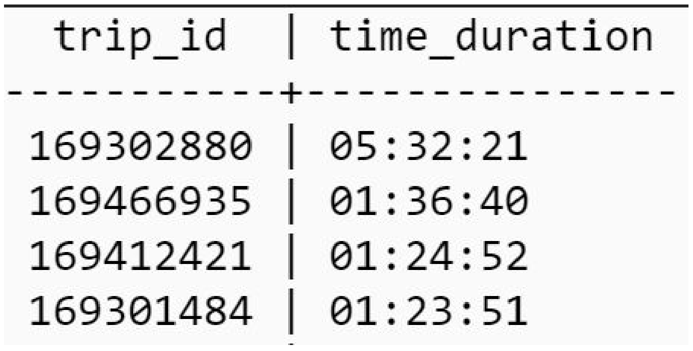


```
 direction | count
-----------+-------
         0 |    96
         1 |    96
         2 |    96
         3 |    96
         4 |    96
         5 |    96
         6 |    96
         7 |    96
```

360 Rows

Query:
SELECT BreadCrumb.direction, COUNT(DISTINCT Trip.vehicle_id) as count
FROM BreadCrumb, Trip
WHERE BreadCrumb.trip_id=Trip.trip_id
GROUP BY BreadCrumb.direction
ORDER BY count DESC;

Which is the longest (in terms of time) trip of all trips in the data?
Answer: 05:32:21

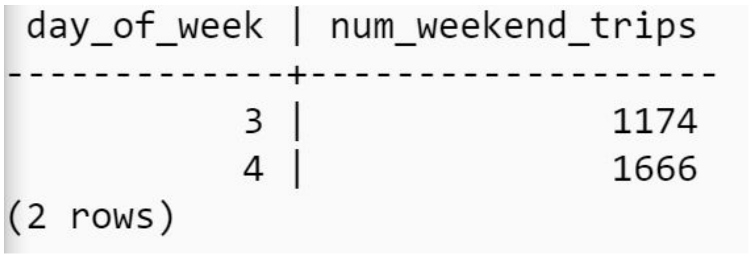


```
 trip_id   | time_duration
-----------+---------------
 169302880 | 05:32:21
 169466935 | 01:36:40
 169412421 | 01:24:52
 169301484 | 01:23:51
```

Query:
SELECT trip_id, MAX(tstamp) - MIN(tstamp) as time_duration
FROM BreadCrumb
GROUP BY trip_id
ORDER BY time_duration DESC;

Devise three new, interesting questions about the C-Tran bus system that can be answered by your bread crumb 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).

1) Does Wednesday or Thursday have more trips?
Answer: Thursday

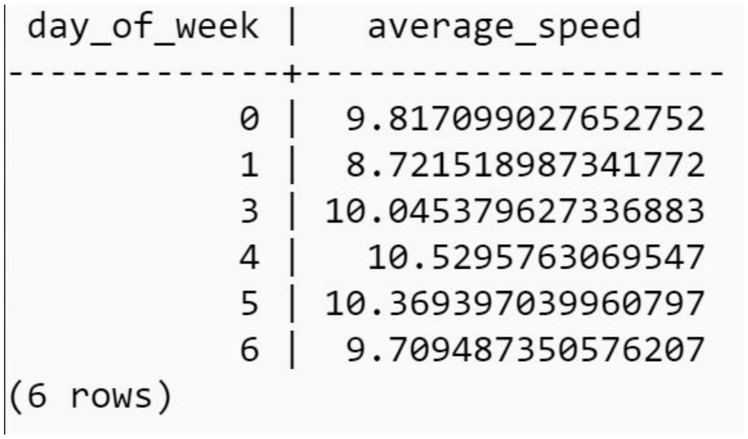


```
 day_of_week | num_weekend_trips
-------------+-------------------
           3 |              1174
           4 |              1666
(2 rows)
```

Query:
SELECT
   EXTRACT(dow from tstamp::date) as day_of_week,
   COUNT(DISTINCT trip_id) as num_weekend_trips
FROM BreadCrumb
WHERE
   EXTRACT(dow from tstamp::date) = 3 OR
   EXTRACT(dow from tstamp::date) = 4
GROUP BY EXTRACT(dow from tstamp::date);

2) What is the average speed of each day of the week?

Answer (we don't have that much data yet):



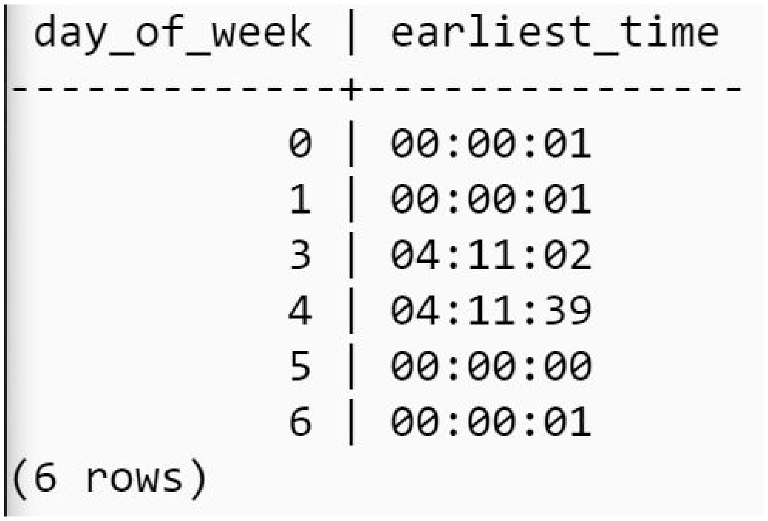
```
 day_of_week |   average_speed
-------------+--------------------
           0 |  9.817099027652752
           1 |  8.721518987341772
           3 | 10.045379627336883
           4 |   10.5295763069547
           5 | 10.369397039960797
           6 |  9.709487350576207
(6 rows)
```

Query:

```
SELECT
   EXTRACT(dow from tstamp::date) as day_of_week,
   AVG(speed) as average_speed
FROM BreadCrumb
GROUP BY EXTRACT(dow from tstamp::date);
```

3) What is the earliest time you will see a bus running on a given day of the week?

Answer (we don't have that much data yet):



```
 day_of_week | earliest_time
-------------+---------------
           0 | 00:00:01
           1 | 00:00:01
           3 | 04:11:02
           4 | 04:11:39
           5 | 00:00:00
           6 | 00:00:01
(6 rows)
```

Query:

```
SELECT EXTRACT(dow from tstamp::date) as day_of_week, MIN(CAST(tstamp as time)) as
earliest_time
FROM BreadCrumb
GROUP BY EXTRACT(dow from tstamp::date);
```

Your Code

Provide a reference to the repository where you store your python code. If you are keeping it private then share it with Bruce (bruce.irvin@gmail.com), David and Aman (github references TBD).

GitHub Repository: https://github.com/alex-bailey1/Data_Eng_project