TEAM MEMBERS:
Vijayalaxmi Pujar
Susan Onesky
Chitradevi Maruthavanan
Prachi Kashyap

5/14/2023

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
5/13/2023	7	278245	Breadcrumb table = 278245 Trip table = 613
5/14/2023	1	251352	Breadcrumb table = 250958 Trip table = 616
5/10/2023	4	216463	Breadcrumb table = 216332 Trip table = 561
5/11/2023	5	193455	Breadcrumb table = 193234 Trip table = 477
5/12/2023	6	288022	Breadcrumb table = 287767 Trip table = 697
5/8/2023	2	264505	Breadcrumb table = 264230 Trip table = 640
5/17/2023	4	162054	Breadcrumb table = 161736 Trip table = 381

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 unique index number associated with a particular vehicle trip. This field acts as a foreign key referencing the trip in the veh_trip table.

EVENT_NO_STOP: The unique index number associated with a particular stop or point. This field acts as a foreign key referencing the stop in the veh_stop table.

OPD_DATE: The date of operation when the cyclic event occurred.
The format is 31DEC2022:00:00:00. The first value is the date, then month, year and 00:00:00 at the end of each. We included information about two dates of data we were able to include.

VEHICLE_ID: The unique identifier for a particular vehicle.
The number of unique vehicle IDs is:

METERS: This field represents the meters that a vehicle has traveled. The meters are in the range of 0- #

ACT_TIME: The actual time when the cyclic event occurred. This could represent the time in seconds since the start of the day. Our breadcrumbs have snapshots every five seconds

GPS_LONGITUDE: The longitude coordinate of the vehicle's position as per GPS (Global Positioning System) in WGS 84 format. The routes are in the Portland area so should be in that longitude.

GPS_LATITUDE: The latitude coordinate of the vehicle's position as per GPS in WGS 84 format.

GPS_SATELLITES: The number of satellites that were in contact with the vehicle's GPS system at the time of the cyclic event.

GPS_HDOP: The horizontal dilution of precision (HDOP) from the GPS, which indicates the reliability of the horizontal GPS position. The lower the value, the more reliable horizontal GPS position is.

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.
Then implement at least 10 of the assertions in your Kafka consumer 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.

We noticed that some of the breadcrumb keys were mismatched with their values. As of midday April 27, we have fixed this issue with the data for future served dates, but the data that you have already gathered contains errors. Your data validation process should tell you which fields are erroneous so that you can transform them in the next step.

<u>20 Data Validation Assertions</u>:

1) Some records are missing lat/lon values.
2) Existence assertion - Every trip must have a “trip no” associated with it.
3) INTER-RECORD ASSERTION - The vehicle id should remain the same for a particularly unique event trip no throughout the trip.
4) GPS_HDOP value is always in the range of 0 - 20
5) GPS_SATELLITES can never be 0, negative, or empty
6) GPS_LONGITUDE must always be a negative value (As the US is in the western hemisphere) and GPS_LATITUDE must always be a positive value (As we are in the northern hemisphere)
7) LIMIT ASSERTION - METERS must always be greater than or equal to zero
8) INTER-RECORD ASSERTION - EVENT_NO_TRIP value should always be lesser than EVENT_NO_STOP
9) (existence): No records should have repetition
10) Each EVENT_NO_TRIP has one distinct vehicle_ID
11) Each EVENT_NO_TRIP has one distinct route ID// I guess we don’t populate route ID so might not include this assertion.
12) Each unique EVENT_NO_TRIP has all distinct ACT_TIME values (no repeating timestamps for a trip)
13) Each breadcrumb in the sequence has ACT_TIME that is 5 greater than the last breadcrumb.
14) OPD_DATE first character is only a digit 1,2 or 3
15) OPT_DATE has a second character only as an integer between 0-9
16) OPD_DATE has characters 3-5 that are only DEC or JAN
17) OPD_DATE has characters 6-9 only as 2022
18) OPD_DATE has the last 9 characters only as:00:00:00
19) OPD_DATE is the same in all breadcrumbs of a day’s json file
20) Each breadcrumb in the sequence with the same EVENT_NO_TRIP does not have a maximum value of METERS greater than 178 than the last breadcrumb reading for that same EVENT_NO_TRIP.
21) There should not be any duplicate records with the same EVENT_NO_TRIP, EVENT_NO_STOP, and VEHICLE_ID.
22) EVENT_NO_STOP must be a non-negative integer and unique for each stop within a trip.
23) OPD_DATE should not be a future date

24) The GPS_SPEED should not exceed the vehicle's maximum operational speed.
25) Meters must always be greater than or equal to 0

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.

1. Added a speed column with a given formula.
2. Added columns for direction, service key, and route id, where the direction and service key value is none and the route id is -1.
3. The vehicle id should remain the same for a particular unique event trip no throughout the trip.:For this assertion we removed the duplicate trip_ids.
4. We renamed event_no_trip as trip_id in the trip table.
5. In the BreadCrumb table we reshaped the act_time as timestamp(tstamp).
6. We dropped a few of the field which depends on the two tables which we created in Postgre.

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).

Answer the following questions about the TriMet system using your sensor data 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).

1. How many vehicles are there in the TriMet system?

```
Waiting for message or event/error in poll()
(confluent-exercise) chitram2@project-instance:~$ sudo -u postgres psql postgreses
psql (13.11 (Debian 13.11-0+deb11u1))
Type "help" for help.

postgres=# SELECT COUNT(DISTINCT vehicle_id) AS vehicle_count FROM Trip;
 vehicle_count
---------------
            66
(1 row)
```

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

```
postgres=# SELECT COUNT(*) AS event_count FROM breadcrumb WHERE DATE(tstamp) = '2023-01-01';
 event_count
-------------
      208990
(1 row)
```

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

```
postgres=# SELECT COUNT(*) AS event_count FROM breadcrumb WHERE DATE_TRUNC('day',tstamp) = '2023-01-02';
 event_count
-------------
      197591
(1 row)
```

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

```
postgres=# SELECT EXTRACT(DOW FROM tstamp) AS day_of_week, AVG(reading_count) AS average_readings
FROM (
    SELECT DATE_TRUNC('day', tstamp) AS tstamp, COUNT(*) AS reading_count
    FROM BreadCrumb
    GROUP BY DATE_TRUNC('day', tstamp), trip_id
) AS daily_readings
GROUP BY day_of_week
ORDER BY day_of_week;
 day_of_week |    average_readings
-------------+------------------------
           3 | 454.6262458471760797
           4 | 285.0000000000000000
(2 rows)
```

5. 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/lon bounding box such as [(45.497805, -122.566576), (45.504025,

-122.563187)].

```
postgres=# SELECT DISTINCT T.trip_id, T.route_id, T.vehicle_id
FROM Trip T
INNER JOIN BreadCrumb B ON T.trip_id = B.trip_id
WHERE DATE_TRUNC('day', B.tstamp)
= DATE '2023-01-01'
AND B.latitude BETWEEN 45.497805 AND 45.504025
AND B.longitude BETWEEN -122.566576 AND -122.563187;
  trip_id  | route_id | vehicle_id
-----------+----------+------------
 229984997 |       -1 |       3058
 230003642 |       -1 |       3318
 230025626 |       -1 |       3942
 230026822 |       -1 |       3945
 230117920 |       -1 |       3611
 230119964 |       -1 |       3619
 230172309 |       -1 |       3510
 230185095 |       -1 |       3715
(8 rows)
```

6. 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?

```
postgres=# SELECT * from breadcrumb;

postgres=#
postgres=# SELECT * from BreadCrumb WHERE tstamp >= '2023-01-01 06:00:00'AND tstamp < '2023-01-01 08:00:00'
AND latitude >= 45.506022 AND latitude <= 45.516636
AND longitude >= -122.711662 AND longitude <= -122.700316
AND EXTRACT (DOW FROM tstamp) = 0
ORDER BY tstamp;
postgres=#
```

```
       tstamp        | latitude  |  longitude  |       speed        |  trip_id
---------------------+-----------+-------------+--------------------+-----------
 2023-01-01 06:00:16 | 45.507005 | -122.711178 | 17.465894465894465 | 230045092
 2023-01-01 06:00:21 | 45.507425 | -122.709808 |  23.46987951807229 | 230045092
 2023-01-01 06:00:26 | 45.508192 | -122.708822 | 22.850467289719628 | 230045092
 2023-01-01 06:00:31 | 45.509147 | -122.708142 |              23.05 | 230045092
 2023-01-01 06:00:36 |  45.51007 | -122.707495 | 23.015384615384615 | 230045092
 2023-01-01 06:00:41 |  45.51093 | -122.706577 |               23.8 | 230045092
 2023-01-01 06:00:51 |  45.51214 |  -122.70409 |               23.5 | 230045092
 2023-01-01 06:00:56 |  45.51275 | -122.702773 |               24.4 | 230045092
 2023-01-01 06:01:01 | 45.513562 | -122.701678 |               24.8 | 230045092
 2023-01-01 06:01:06 | 45.514458 | -122.700847 |               23.4 | 230045092
(10 rows)
```

7. What is the maximum velocity reached by any bus in the system?

```
postgres=# SELECT MAX(speed) AS max_velocity
FROM BreadCrumb;
 max_velocity
--------------
        40.25
(1 row)
```

8. 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.

```
postgres=# SELECT breadcrumb.speed, COUNT(*) AS vehicle_count
FROM trip ,breadcrumb
GROUP BY speed
HAVING COUNT(*) >= 1
ORDER BY vehicle_count DESC;
```

```
         speed          | vehicle_count
------------------------+---------------
                     0  |       7734097
                    10  |       3471925
                  10.4  |       3459635
                  10.6  |       3410475
                  10.8  |       3308468
```

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

```
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
-----------+---------------
 231812405 | 02:52:34
(1 row)
```

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).

    a. How many trips were taken in total within the TriMet system?

```
postgres=# SELECT COUNT(*) AS total_trips
FROM trip;
 total_trips
-------------
         613
(1 row)
```

b. What is the highest recorded speed in the TriMet system?

```
postgres=# SELECT MAX(speed) AS highest_speed
FROM breadcrumb;
 highest_speed
---------------
          25.4
(1 row)
```

c. Retrieve the latest breadcrumb for each trip:

```
postgres=# SELECT trip.trip_id, breadCrumb.*
FROM trip
JOIN breadCrumb ON trip.trip_id = breadCrumb.trip_id
WHERE breadCrumb.tstamp = (
    SELECT MAX(tstamp)
    FROM breadCrumb
    WHERE breadCrumb.trip_id = trip.trip_id
);
  trip_id  |       tstamp        | latitude  |  longitude  |       speed        |  trip_id
-----------+---------------------+-----------+-------------+--------------------+-----------
 231428659 | 2023-01-04 15:27:16 | 45.496067 | -122.682397 |  7.279671457905544 | 231428659
 231428692 | 2023-01-04 16:12:23 |  45.49136 | -122.800513 |                3.8 | 231428692
 231428717 | 2023-01-04 17:04:43 | 45.496378 | -122.682373 |  7.186301369863013 | 231428717
 231428741 | 2023-01-04 17:54:40 | 45.491013 | -122.801377 |  5.731521739130435 | 231428741
 231428770 | 2023-01-04 18:32:12 | 45.496388 | -122.682292 |  8.983333333333333 | 231428770
```

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 (bruce.irvin@gmail.com) and TA (mina8@pdx.edu).

URL - https://github.com/Chitramvanan/DataEngineering_Project/tree/main/Project%202