DataEng S24: Project Assignment 2

Validate, Transform, Enhance and Store

Due date: May 12, 2024 at 10pm

By now your pipeline should be automatically gathering and transferring TriMet breadcrumb records daily but not yet doing much with the data. The next step in building your data pipeline is to get your data in shape for analysis. This includes:

A. understanding the contents of the breadcrumb data
B. reviewing the needed schema for the data
C. validating the data
D. transforming the data
E. storing the data into a database server
F. example queries

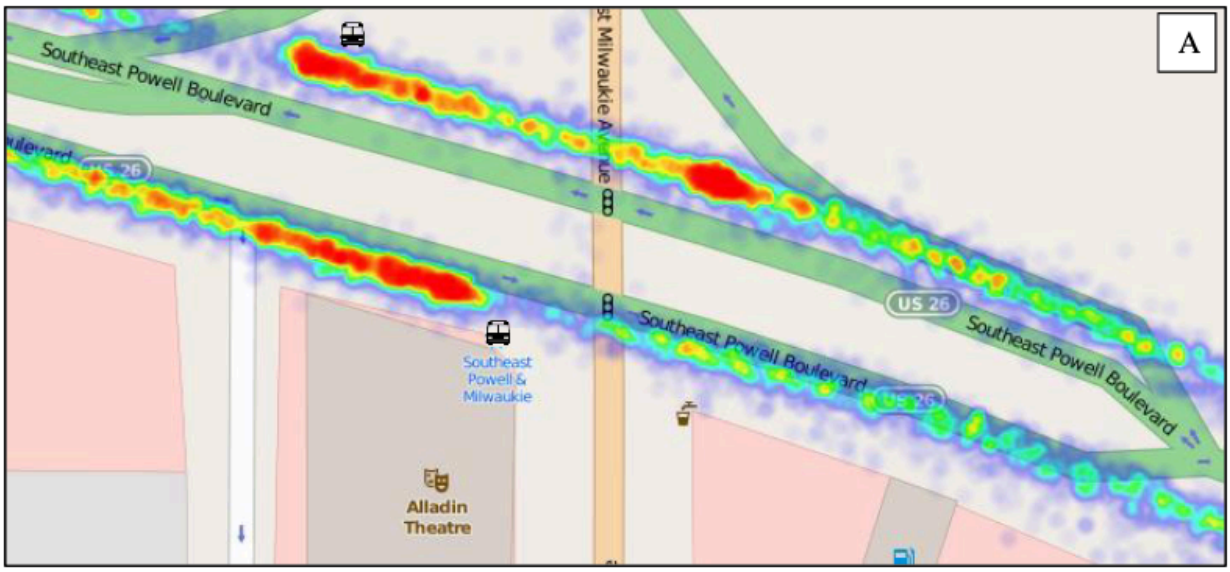
Your job is to enhance your Pub/Sub receiver to perform steps A-E so that your pipeline ingests high-quality data daily into a database server in preparation for visualization.

A. Review the Data

Have a look at a few of the data records and see if you can determine the meaning of each record attribute. Also, have a look at the only documentation that we have for the data (provided to us by TriMet). The documentation is not very good, so we are counting on you to give better descriptions of each field. In your submission document, provide an improved description of each data field.

B. Database Schema

We plan to develop a visualization application that builds visualizations similar to this:



This diagram shows vehicle speeds as a heatmap overlaid on a street map for a given latitude/longitude rectangle, route, date range and time range. Colors in the heatmap correspond to average speeds of buses at each GPS location for the selected route, date range and/or time range.

To achieve this we need you to build database tables (or views) that conform to the following schema.

Table: **BreadCrumb**

tstamp	latitude	longitude	speed	trip_id

The BreadCrumb table keeps track of each individual breadcrumb reading. Each row of the BreadCrumb table corresponds to a single GPS sensor reading from a single bus.

tstamp: the moment at which the event occurred, i.e., at which the bus's GPS location was recorded. This should be a Postgres "timestamp" type of column.

latitude: a float value indicating fractional degrees of latitude. Latitude and Longitude together indicate the location of the bus at time=tstamp.

longitude: a float value indicating the fractional degrees of longitude. Latitude and Longitude indicate the location of the bus at time=tstamp.

speed: a float value indicating the speed of the bus at time=tstamp. Speed is measured in meters per second.

trip_id: this integer column indicates the identifier of the trip in which the bus was participating. A trip is a single run of a single bus servicing a single route. This column is a foreign key referencing trip_id in the Trip table.

Table: **Trip**

trip_id	route_id	vehicle_id	service_key	direction

The Trip table keeps track of information about each bus trip. A “trip” is a single run of a single bus over a single route.

trip_id: unique integer identifier for the trip.

route_id: integer identifying the route that the trip was servicing. For this project we do not have a separate Route table, but if we did, then this would be a foreign key reference to the Route table. Note that you will not have enough information to properly populate this field during this assignment #2.

vehicle_id: integer identifying the vehicle that serviced the trip. A trip is serviced by only one vehicle but a vehicle services potentially many trips.

service_key: one of ‘Weekday’, ‘Saturday’, or ‘Sunday’ depending on the day of the week on which the trip occurred OR depending on whether the trip occurred on a Holiday recognized by TriMet. Note that you will not have enough information to properly populate this field during this assignment #2.

direction: either ‘0’ or ‘1’ indicating whether the trip traversed the route from beginning to end (‘0’) or from end to beginning (‘1’). Note that this “direction” has a completely different meaning than the same-named column in the BreadCrumb table (sorry about that). Also note that you will not have enough information to properly populate this field during assignment #2.

See the ER diagram and DDL code for the schema. Please implement this schema precisely in your database so that the visualization tool will work for you.

C. Data Validation

Create 20 distinct data validation assertions for the breadcrumb data. These should be in English (not python). Include them in your submission document (see below).

Then implement at least 10 of the assertions in your Pub/Sub receiver 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.

D. Data Transformation

Add code to your Pub/Sub receiver to transform your data. Your transformations should be driven by either the need to resolve validation assertion violations or the need to shape the data for the schema. Describe any/all needed transformations in your submission document.

A transformation that you must do is to add a speed column to conform with the BreadCrumb table schema. TriMet did not give this information in their breadcrumb service; however, we can extrapolate this information from the data that we have. To find speed, use the formula below:

$$speed = \frac{\Delta distance}{\Delta time}$$

Each breadcrumb contains data of the time it was sent and the odometer reading of the bus. You will need to find the distance traveled and the exact time between each consecutive breadcrumb of a trip to make the calculation.

Here's an example of finding the speed:

	EVENT_NO_TRIP	EVENT_NO_STOP	OPD_DATE	VEHICLE_ID	METERS	ACT_TIME	GPS_LONGITUDE	GPS_LATITUDE	GPS_SATELLITES	GPS_HDOP
47888	259173279	259173281	15FEB2023:00:00:00	4223	177061	68354	-122.525793	45.489655	12.0	0.8
47889	259173279	259173281	15FEB2023:00:00:00	4223	177118	68359	-122.525102	45.489513	12.0	0.8

The above DataFrame shows two consecutive breadcrumbs sent from the same bus on the same trip. The METERS column shows the odometer reading of the bus at the time of breadcrumb emission in meters, and the ACT_TIME column is the time in which the breadcrumb was sent in seconds. Calculate as below:

$$\frac{(METERS_{47889} - METERS_{47888})}{(ACT\_TIME_{47889} - ACT\_TIME_{47888})} = \frac{(177118 - 177061)}{(68359 - 68354)} = 11.4$$

Note that to calculate the speed of a breadcrumb, you need the previous breadcrumb of that trip. We cannot make this calculation for the very first breadcrumb in each trip. Therefore, we are expecting you to do the following: calculate the speed for the second breadcrumb of the trip; use that speed, the calculated speed of the second breadcrumb, for the first breadcrumb of the trip as well.

E. Storage in Database Server

1. Install and configure a PostgreSQL database server on your Virtual Machine. Refer to this document to learn how. For more information on PostgreSQL commands, you can see guides like this one: PostgreSQL Cheat Sheet.

2. Enhance your data pipeline to load your transformed data into your database server. You are free to use any of the data loading techniques that we discussed in class. The data should be loaded ASAP after your Pub/Sub receiver receives the data, validates the data and transforms the data so that you have a reliable end-to-end data pipeline running daily and automatically.

F. Example Queries

Answer the following questions about the TriMet system using your vehicle group's sensor data in your 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 breadcrumb reading events occurred on January 1, 2023?
```
SELECT COUNT(*)
FROM BreadCrumb
WHERE DATE(tstamp) = '2023-01-01 00:00:00' AND DATE(tstamp)< '2023-01-02 00:00:00';
```

2. How many breadcrumb reading events occurred on January 2, 2023?
```
SELECT COUNT(*)
FROM BreadCrumb
WHERE DATE(tstamp) = '2023-01-02';
```

3. On average, how many breadcrumb readings are collected on each day of the week?
```
SELECT
    EXTRACT(DOW FROM tstamp) AS day_of_week,
    AVG(count_readings) AS average_readings_per_day
FROM (
    SELECT
        DATE(tstamp) AS reading_date,
        COUNT(*) AS count_readings
    FROM BreadCrumb
    GROUP BY reading_date
) AS daily_readings
GROUP BY day_of_week
ORDER BY day_of_week;
```

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.*
FROM Trip T
JOIN BreadCrumb B ON T.trip_id = B.trip_id
WHERE B.latitude BETWEEN 45.497805 AND 45.504025
    AND B.longitude BETWEEN -122.566576 AND -122.563187
    AND DATE(B.tstamp) = '2023-01-01';
```

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?

FOR MONDAYS:

```
SELECT *
FROM BreadCrumb
WHERE latitude BETWEEN 45.506022 AND 45.516636
    AND longitude BETWEEN -122.711662 AND -122.700316
    AND EXTRACT(DOW FROM tstamp) = 1 -- Monday
    AND EXTRACT(HOUR FROM tstamp) BETWEEN 16 AND 18 -- Between 4pm and 6pm
ORDER BY tstamp;
```

FOR SUNDAYS:

```
SELECT *
FROM BreadCrumb
WHERE latitude BETWEEN 45.506022 AND 45.516636
    AND longitude BETWEEN -122.711662 AND -122.700316
    AND EXTRACT(DOW FROM tstamp) = 0 -- Sunday
    AND EXTRACT(HOUR FROM tstamp) BETWEEN 6 AND 8 -- Between 6am and 8am
ORDER BY tstamp;
```

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

```
SELECT MAX(speed) AS max_speed
FROM BreadCrumb;
```

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 speed, COUNT(DISTINCT vehicle_id) AS num_vehicles
FROM BreadCrumb
GROUP BY speed
ORDER BY num_vehicles DESC, speed;
```

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

```
SELECT
    trip_id,
    MIN(tstamp) AS start_time,
    MAX(tstamp) AS end_time,
    MAX(tstamp) - MIN(tstamp) AS trip_duration
FROM BreadCrumb
GROUP BY trip_id
ORDER BY trip_duration DESC
LIMIT 1;
```

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?

   Analyzing how many breadcrumbs are left on non-holiday Wednesdays, non-holiday Saturdays, and holidays gives us a peek into how TriMet operates on different types of days. By counting the breadcrumbs separately for each day, we can spot any interesting differences in how people use public transit and how TriMet responds. For example, weekdays without holidays might show us the usual rush hour commuting patterns, while Saturdays might reveal more leisurely travel behaviors. Holidays, with their unique festivities or events, could bring surprises in terms of service demand and passenger behaviors. Understanding these nuances can help TriMet plan services better, allocate resources wisely, and tailor operations to suit the needs of passengers across a variety of days and situations.

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).

    1. Question 1: What are the top 10 busiest bus stops based on the number of breadcrumb readings recorded?

```
SELECT stop_id, COUNT(*) AS num_readings
FROM BreadCrumb
GROUP BY stop_id
ORDER BY num_readings DESC
```

```
LIMIT 10;
```

2. Question 2: What is the average speed of buses during peak hours (8am - 10am) on weekdays?

```
SELECT AVG(speed) AS avg_speed
FROM BreadCrumb
WHERE EXTRACT(DOW FROM tstamp) BETWEEN 1 AND 5 -- Monday to Friday
AND EXTRACT(HOUR FROM tstamp) BETWEEN 8 AND 10;
```

3. Question 3: How does the average speed of buses vary between different routes during weekends?

```
SELECT route_id, AVG(speed) AS avg_speed
FROM BreadCrumb B
JOIN Trip T ON B.trip_id = T.trip_id
WHERE EXTRACT(DOW FROM B.tstamp) IN (0, 6) -- Sunday or Saturday
GROUP BY route_id
ORDER BY avg_speed DESC;
```

Submission

Congratulations! Your data pipeline is now working end-to-end!

To submit your completed Assignment 2, create a Google document containing the table shown below. Share the document as viewable by anybody at PSU who has the link. You do NOT need to share it with the individual instructors. Then include the URL of the document when using the DataEng project assignment submission form.

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
05/16	Thursday	74229	74229
05/17	Friday	107831	107831
05/18 09:00am	Saturday	93773	93773
05/18 12:20pm	Saturday	104540	104540

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 Trip number
EVENT_NO_STOP Stop number
OPD_DATE Date of the record (time at midnight)
VEHICLE_ID ID of the vehicle on the record
METERS Meters traveled on the current trip
ACT_TIME Time passed since OPD_DATE (add ACT_TIME to OPD_DATE to get actual datetime)
GPS_LONGITUDE Longitude coordinate of the breadcrumb
GPS_LATITUDE Latitude coordinate of the breadcrumb
GPS_SATELLITES Number of satellites recording the coordinates
GPS_HDOP Horizontal dilution of precision (see Wikipedia article)

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. The 'TIMESTAMP' column should exist in the received data.
2. The 'GPS_LATITUDE' values should be within the valid range of -90 to 90 degrees.
3. The 'GPS_LONGITUDE' values should be within the valid range of -180 to 180 degrees.
4. The 'SPEED' values should be non-negative.
5. The 'EVENT_NO_TRIP' column should exist in the received data.
6. The 'VEHICLE_ID' column should exist in the received data.
7. The 'EVENT_NO_TRIP' column should not contain null values.
8. The 'VEHICLE_ID' column should not contain null values.
9. The DataFrame should not contain any missing values.
10. The DataFrame should not contain duplicate timestamps.
11. The number of 'Men' should be non-negative and should not exceed the 'TotalPop'.
12. The number of 'Women' should be non-negative and should not exceed the 'TotalPop'.
13. The percentage of 'Hispanic' residents should be between 0 and 100.
14. The percentage of 'White' residents should be between 0 and 100.
15. The percentage of 'Black' residents should be between 0 and 100.
16. The percentage of 'Native' residents should be between 0 and 100.
17. The percentage of 'Asian' residents should be between 0 and 100.
18. The percentage of 'Pacific' residents should be between 0 and 100.
19. The number of 'VotingAgeCitizen' should be non-negative and should not exceed the 'TotalPop'.
20. Each record must have a valid 'date' in the format YYYY-MM-DD.

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 transformed the data by creating two new columns:

TIMESTAMP: This was calculated by converting the OPD_DATE of the first row (midnight) to

an hour/minutes/seconds format, and then using the seconds in ACT_TIME to determine the time of the crash since midnight.

SPEED: To calculate speed we first calculated the change in TIMESTAMP (in seconds) using the diff() function, and the change in METERS, again using the diff() function. Then we divided the difference in meters by the difference in seconds to get the SPEED column.

Both of these columns were inserted into the table, and ACT_TIME, and OPD_DATE were dropped.

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).

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).