DataEng S23: Project Assignment 3

Data Integration

Due date: May 28, 2023 @10pm PT

Congratulations! By now you have a working, end-to-end data pipeline. Unfortunately, it does not have enough data to properly implement our Data Scientist's visualization. To fill out information such as "route ID" you need to access another source of data and build a new pipeline to integrate it with your initial pipeline. Here are your steps:

A. access the stop event data
B. build a new pipeline for the stop event data
C. integrate the stop event data with the breadcrumb data
D. testing

A. Stop Event Data

Access TriMet "Stop Event" data at this URL: http://www.psudataeng.com:8000/getStopEvents As with the previous data source, this data set gives all TriMet vehicle stop events for a single day of operation.

B. New Pipeline

Your job is to build a new pipeline that operates just like the previous one, including use of Kafka, automation, validation and loading.

C. Integrate Stop Events with Bread Crumbs

The two pipelines (BreadCrumb pipeline and StopEvent pipeline) must update the values in the Trip table such that all of the columns of both tables are filled correctly.

[5/20/2022] Alternatively, it would be OK to load the StopEvent data into a separate table and then use SQL views to integrate the two datasets.

D. Visualization

MapboxGL is a data visualization tool that allows you to view your breadcrumb data and display it on a map. Your job is to integrate this tool with your database tables so that you can query the breadcrumb and trip data in your database server, transform to geoJSON format and display the resulting map visualization. To get started, see this guide.

[5/20/2022] Alternatively, you may use an alternative visualization tool (such as folium) to create the required visualizations. We do not provide any guides for doing it, but you are free to do so if you prefer. The submitted visualizations must be equivalent or superior to the visualizations produced by the provided MapboxGL based visualization tool.

Submission

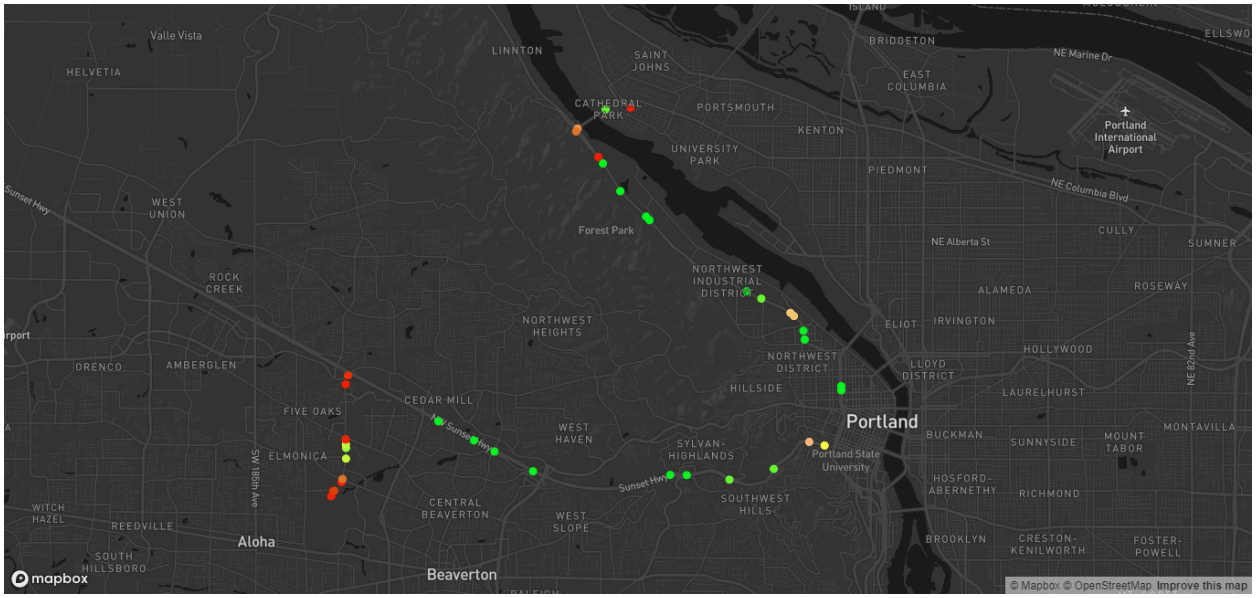
Make a copy of this document and update it to include the following visualizations. For each visualization extract from your database a list of (latitude, longitude, speed) tuples and then use the provided visualization code (see Section D above) to display bus speeds at all of the corresponding geographic coordinates. So, for example, if you are asked to visualize a “trip”, then you must query your database to find all of the (latitude, longitude, speed) tuples for that trip, and then display a map showing the recorded/calculated bus speed at each (latitude,longitude) location.

No need to produce software that neatly displays trips, routes, dates, times, etc. onto the visualization itself. Instead, just paste a screen capture of the map-based speed visualization into your submission document and then include a text description of the contents of the visualization. For example, text like this: “Bus Speeds for all outbound trips of route 72 between 9am and 11am on Wednesday, February 15, 2023.”

Note: We have changed questions 2 and 3 slightly since we did not have data of any Friday or Sunday, as we have stop event data of only 4 days now and need to join the breadcrumb table with it, we have discarded the previous data.

Visualization 1. A visualization of speeds for a single trip for any bus route that crosses the US-26 tunnel. You choose the day, time and route for your selected trip. To find a trip that traverses this tunnel, consider finding a trip that includes breadcrumb sensor points within this bounding box: [(45.506022, -122.711662), (45.516636, -122.700316)]. Any bus trip that includes breadcrumb points within that box either drove across the tunnel or teleported across!

SELECT * FROM breadcrumb WHERE trip_id = (SELECT trip_id FROM breadcrumb WHERE latitude BETWEEN 45.506022 AND 45.516636 AND longitude BETWEEN -122.711662 AND -122.700316 AND DATE(tstamp) = '2023-01-16' GROUP BY trip_id ORDER BY COUNT(trip_id) DESC LIMIT 1) ORDER BY tstamp;

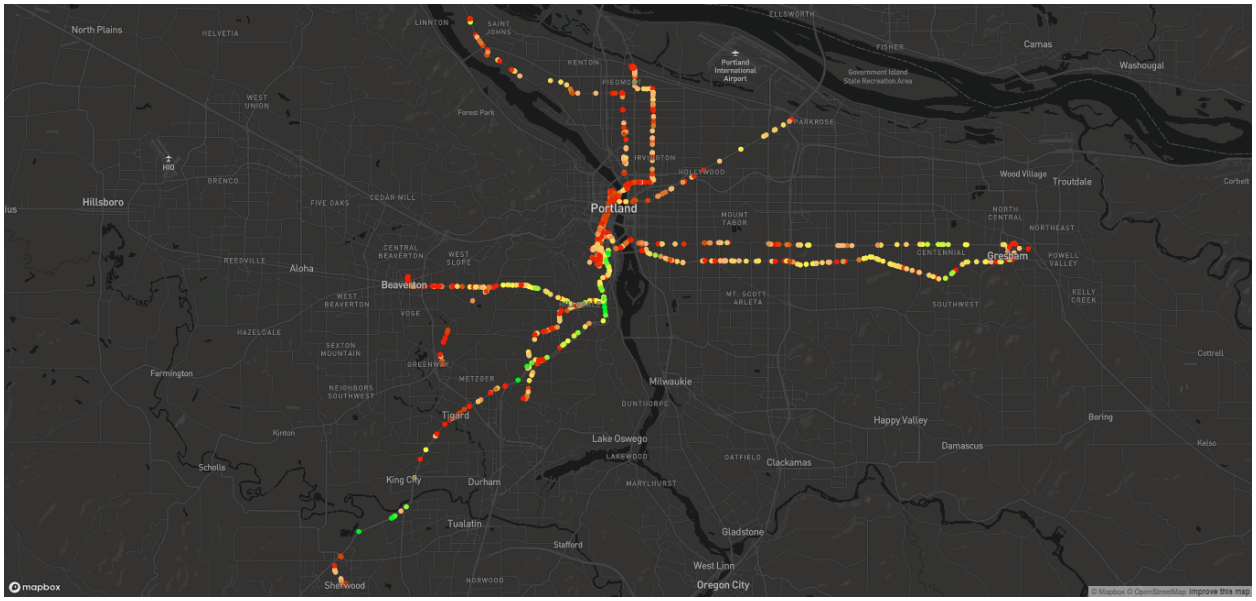


Visualization 2. All outbound trips that occurred on route 15 on a Monday between the hours of 4pm and 6pm.

SELECT * FROM breadcrumb b JOIN trip_vw t ON b.trip_id = t.trip_id WHERE route_id = 15 AND EXTRACT(DOW FROM tstamp) = 1 AND EXTRACT(HOUR FROM tstamp) >= 16 AND EXTRACT(HOUR FROM tstamp) < 18;

Visualization 3. All trips that travel to and from PSU campus on a Tuesday morning between 9am and 11am.

SELECT * FROM breadcrumb WHERE trip_id IN (SELECT trip_id FROM breadcrumb WHERE latitude BETWEEN 45.5080084 AND 45.5137478 AND longitude BETWEEN -122.6843533 AND -122.6809583 AND EXTRACT(DOW FROM tstamp) = 2 AND EXTRACT(HOUR FROM tstamp) >= 9 AND EXTRACT(HOUR FROM tstamp) < 11);

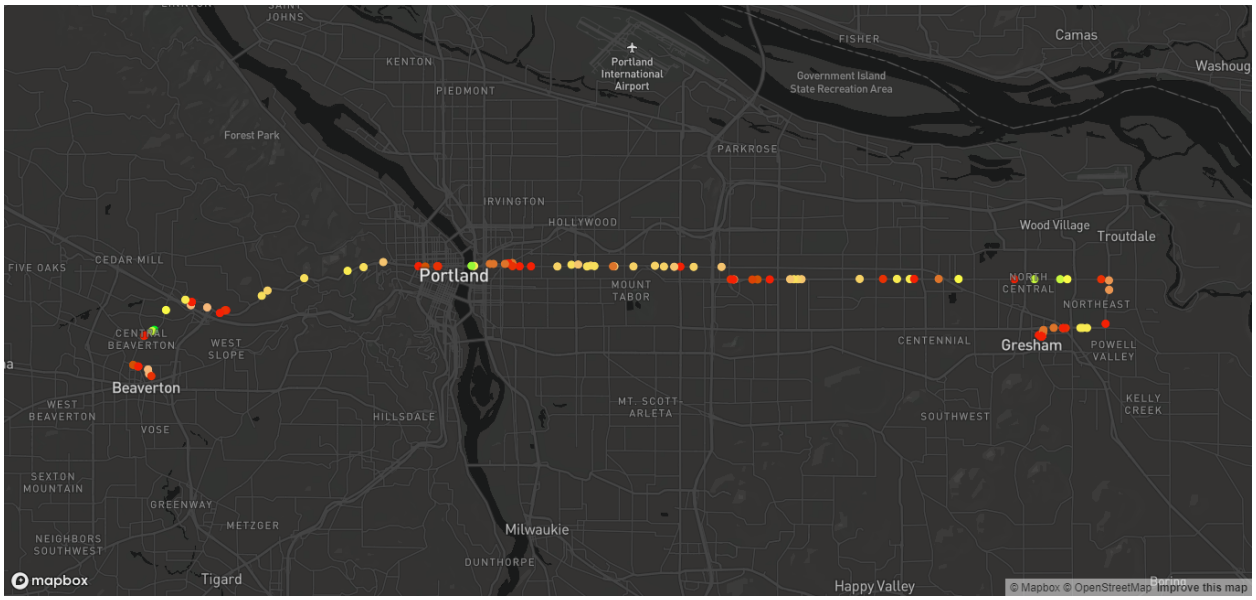


Visualization 4. The longest (as measured by time) trip in your entire data set. Indicate the date, route #, and the trip ID of the trip along with a visualization showing the entire trip.

SELECT s.trip_id, s.longtrip AS longest_trip_duration, DATE(b.tstamp), t.route_id, t.service_key, t.direction FROM (SELECT trip_id, MAX(tstamp)-MIN(tstamp) AS longtrip FROM breadcrumb GROUP BY trip_id) AS s JOIN breadcrumb b ON s.trip_id = b.trip_id JOIN trip_vw t ON b.trip_id = t.trip_id ORDER BY longtrip DESC LIMIT 1;

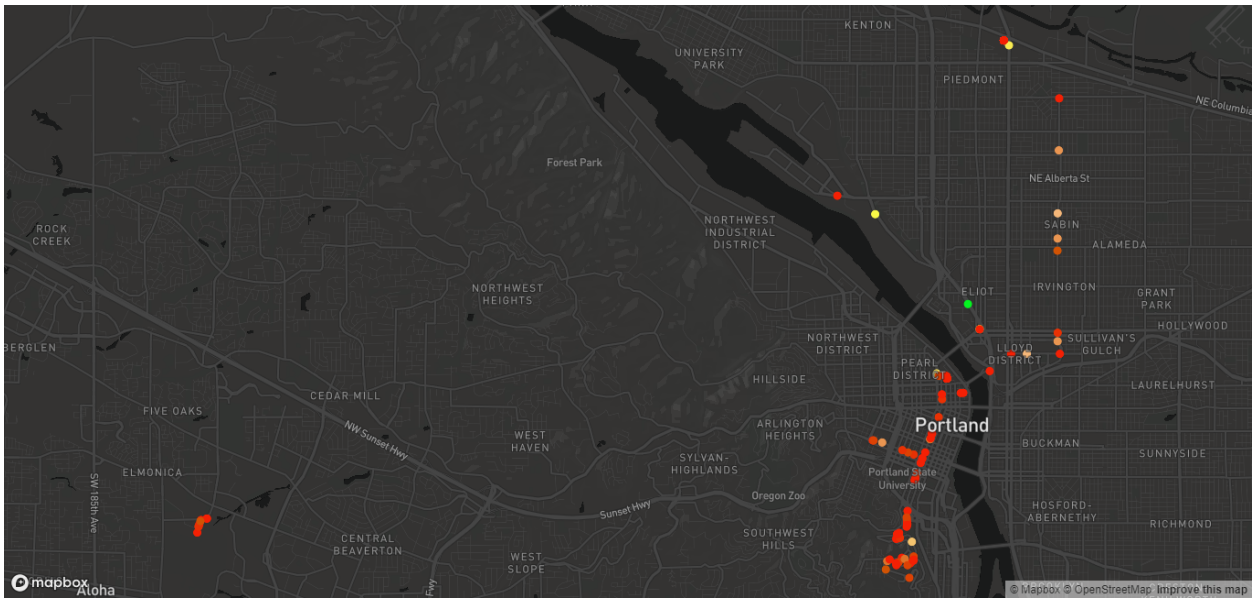
```
postgres=# SELECT s.trip_id, s.longtrip AS longest_trip_duration, DATE(b.tstamp), t.rout
e_id, t.service_key, t.direction FROM (SELECT trip_id, MAX(tstamp)-MIN(tstamp) AS longtr
ip FROM breadcrumb GROUP BY trip_id) AS s JOIN breadcrumb b ON s.trip_id = b.trip_id JOI
N trip_vw t ON b.trip_id = t.trip_id ORDER BY longtrip DESC LIMIT 1;
  trip_id  | longest_trip_duration |    date    | route_id | service_key | direction
-----------+-----------------------+------------+----------+-------------+-----------
 239247116 | 02:23:39              | 2023-01-16 |       20 | Weekday     |         0
(1 row)
```

SELECT * FROM breadcrumb b JOIN trip_vw t ON b.trip_id = t.trip_id WHERE b.trip_id = (SELECT trip_id FROM (SELECT trip_id, MAX(tstamp) - MIN(tstamp) AS trip_duration FROM breadcrumb GROUP BY trip_id) AS trip_durations ORDER BY trip_duration DESC LIMIT 1);



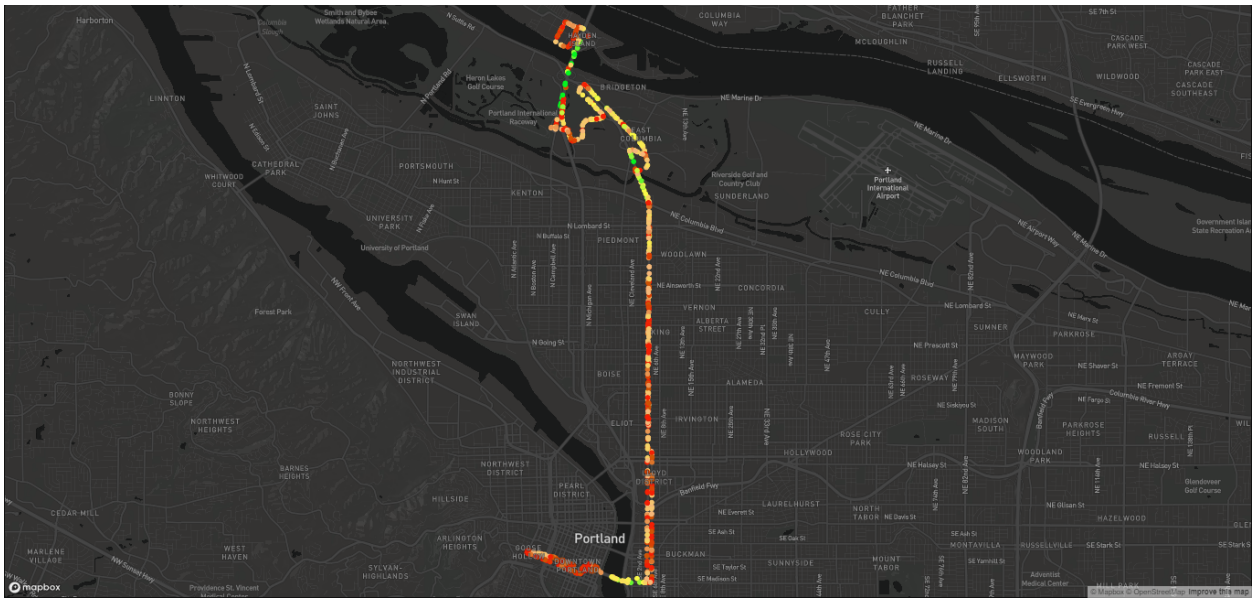
Visualization 5a. Trips with the least 5 average speeds. For choosing this visualization, we considered the slow movement of the route vehicles and that there could be potential for finding alternative ways to improve their speed and provide better service.

SELECT * FROM breadcrumb WHERE trip_id IN (SELECT trip_id FROM (SELECT trip_id, AVG(speed) AS average_speed FROM breadcrumb GROUP BY trip_id ORDER BY average_speed LIMIT 5) a);



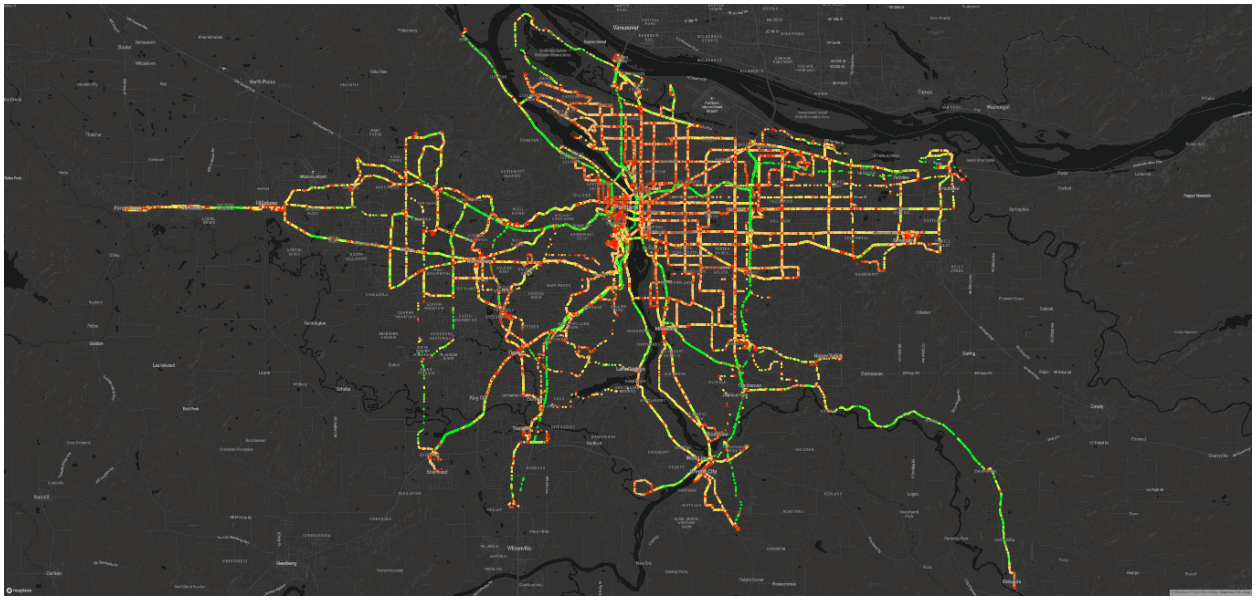
Visualization 5b. All the locations which vehicle 3028 traveled through on January 18, 2023. We chose this visualization because tracking bus locations helps with monitoring performance and punctuality, aiding in evaluating adherence to routes and schedules. It provides valuable data for historical analysis, route optimization, and data-driven decision-making in public transportation planning.

SELECT * FROM breadcrumb b JOIN trip_vw t ON b.trip_id = t.trip_id WHERE t.vehicle_id = 3028 AND DATE(b.tstamp) = '2023-01-18' ORDER BY b.tstamp;



Visualization 5c. A visualization of all the trips in the database. We chose this because we were curious to see how far and wide our data of the trimet network goes and how the entire data we have might look altogether.

SELECT * FROM breadcrumb;



Your Code

Provide a reference to the repository where you store your code. If you are keeping it private then share it with Bruce and Mina.