DataEng S24: Project Assignment 3

Data Integration

**Due date:** May 26, 2024 at 10pm

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 Events Data

Access TriMet “Stop Events” data at this URL:

|  |
| --- |
| https://busdata.cs.pdx.edu/api/getStopEvents?vehicle\_num=<vehicle\_num> |

As with the previous data source, this data set gives all TriMet vehicle stop events for a single day of operation. Again, make sure to replace the vehicle\_num with the vehicle id’s assigned to you.

## B. New Pipeline

Your job is to build a new pipeline that operates just like the previous one, including use of Cloud Pub/Sub, 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.

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](https://docs.mapbox.com/mapbox-gl-js/guides/) 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](https://docs.google.com/document/d/1tMUHMDETFmGZ5kV5soPPY-elQ_U_PCw58yCSBiw-1EM/edit#heading=h.77xnwwr3x5ni).

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.”

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!

**with cte as (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)**

**SELECT longitude, latitude, speed FROM breadcrumb WHERE trip\_id = (select trip\_id from cte) ORDER BY tstamp;**

**PGPASSWORD='Project@123' psql -h localhost -p 5432 -d postgres -U postgres -c "with cte as (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)**

**SELECT longitude, latitude, speed FROM breadcrumb WHERE trip\_id = (select trip\_id from cte) ORDER BY tstamp;" -o output1.tsv -F '\t' -A**

**The below visualization shows the bus that crosses the US-26 tunnel on January 16th, 2023.**

**A map of a city

Description automatically generated**

Visualization 2. All outbound trips that occurred on route 65 on any Friday (you choose which Friday) between the hours of 4pm and 6pm.

**SELECT longitude, latitude, speed FROM breadcrumb b JOIN trip\_vw t ON b.trip\_id = t.trip\_id WHERE route\_id = 65 AND DATE(tstamp) = '2023-01-20' AND EXTRACT(HOUR FROM tstamp) >= 16 AND EXTRACT(HOUR FROM tstamp) < 18;**

**PGPASSWORD='Project@123' psql -h localhost -p 5432 -d postgres -U postgres -c "SELECT longitude, latitude, speed FROM breadcrumb b JOIN trip\_vw t ON b.trip\_id = t.trip\_id WHERE route\_id = 65 AND DATE(tstamp) = '2023-01-20' AND EXTRACT(HOUR FROM tstamp) >= 16 AND EXTRACT(HOUR FROM tstamp) < 18;" -o output2.tsv -F '\t' -A**

**The below visualization shows the bus that occurred on route 65 on January 20th, 2023, between the hours of 4pm and 6pm.**

A map of a city

Description automatically generated

Visualization 3. All trips that travel to and from PSU campus on any Sunday morning (you choose which Sunday) between 9am and 11am.

**SELECT longitude, latitude, speed 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 DATE(tstamp) = '2023-01-08' AND EXTRACT(HOUR FROM tstamp) >= 9 AND EXTRACT(HOUR FROM tstamp) < 11);**

**PGPASSWORD='Project@123' psql -h localhost -p 5432 -d postgres -U postgres -c "SELECT longitude, latitude, speed 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 DATE(tstamp) = '2023-01-08' AND EXTRACT(HOUR FROM tstamp) >= 9 AND EXTRACT(HOUR FROM tstamp) < 11);" -o output3.tsv -F '\t' -A**

**The below visualization shows all trips that travel to and from PSU campus on January 08th, 2023, between the hours of 9am and 11am.**

**A map of a city

Description automatically generated**

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 cts.trip\_id, cts.longtrip AS longest\_trip, 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 cts JOIN breadcrumb b ON cts.trip\_id = b.trip\_id JOIN trip\_vw t ON b.trip\_id = t.trip\_id ORDER BY longtrip DESC LIMIT 1;**

A computer screen with white text

Description automatically generated

**with cte as (SELECT cts.trip\_id, cts.longtrip AS longest\_trip, 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 cts JOIN breadcrumb b ON cts.trip\_id = b.trip\_id JOIN trip\_vw t ON b.trip\_id = t.trip\_id ORDER BY longtrip DESC LIMIT 1)**

**SELECT longitude, latitude, speed FROM breadcrumb b JOIN trip\_vw t ON b.trip\_id = t.trip\_id WHERE b.trip\_id = (select trip\_id from cte);**

**PGPASSWORD='Project@123' psql -h localhost -p 5432 -d postgres -U postgres -c "with cte as (SELECT cts.trip\_id, cts.longtrip AS longest\_trip, 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 cts JOIN breadcrumb b ON cts.trip\_id = b.trip\_id JOIN trip\_vw t ON b.trip\_id = t.trip\_id ORDER BY longtrip DESC LIMIT 1)**

**SELECT longitude, latitude, speed FROM breadcrumb b JOIN trip\_vw t ON b.trip\_id = t.trip\_id WHERE b.trip\_id = (select trip\_id from cte);**

**" -o output4.tsv -F '\t' -A**

**The below visualization shows the longest trip in entire data set and January 23rd, 2023, is having the longest trip with trip\_id 243495295.**

A map of a city

Description automatically generated

Visualization 5a, 5b, 5c, …. Three or more additional visualizations of your choice. Indicate why you chose each particular visualization.

**5a. We want to know all the bus services provided in one day and how far all the bus services are provided in every direction. For this visualization we have selected a specific weekday and collected all the stop points and from there we have collected its latitude and longitude points.**

**select longitude, latitude, speed from breadcrumb b join trip\_vw t on b.trip\_id = t.trip\_id where DATE(b.tstamp) = '2023-01-21' order by b.tstamp;**

**PGPASSWORD='Project@123' psql -h localhost -p 5432 -d postgres -U postgres -c "select longitude, latitude, speed from breadcrumb b join trip\_vw t on b.trip\_id = t.trip\_id where DATE(b.tstamp) = '2023-01-21' order by b.tstamp;" -o output5a.tsv -F '\t' -A**

A map of a city

Description automatically generated

**5b. We want to Identify Busiest Routes During Peak Hours i.e. from morning 7:00 AM to 9:00 AM and evening 4:00 PM to 6:00 PM. The query below identifies the routes with the most data points during peak hours (e.g., 7-9 AM and 4-6 PM). We did visualization by collecting the latitude and longitude points of the routes which are busy during peak hours. By this visualization we understood the routes which are too busy during peak hours. From the observations I found the downtown is so busy during peak hours which is obvious.**

**SELECT t.route\_id, COUNT(b.trip\_id) AS data\_point\_count FROM breadcrumb b JOIN trip\_vw t ON b.trip\_id = t.trip\_id WHERE (EXTRACT(HOUR FROM b.tstamp) BETWEEN 7 AND 9 OR EXTRACT(HOUR FROM b.tstamp) BETWEEN 16 AND 18) AND DATE(b.tstamp) = '2023-01-23' GROUP BY t.route\_id ORDER BY data\_point\_count DESC;**

A screenshot of a computer

Description automatically generated

**SELECT longitude, latitude, speed FROM breadcrumb b JOIN trip\_vw t ON b.trip\_id = t.trip\_id WHERE (EXTRACT(HOUR FROM b.tstamp) BETWEEN 7 AND 9 OR EXTRACT(HOUR FROM b.tstamp) BETWEEN 16 AND 18) AND DATE(b.tstamp) = '2023-01-23' ;**

**PGPASSWORD='Project@123' psql -h localhost -p 5432 -d postgres -U postgres -c "SELECT longitude, latitude, speed FROM breadcrumb b JOIN trip\_vw t ON b.trip\_id = t.trip\_id WHERE (EXTRACT(HOUR FROM b.tstamp) BETWEEN 7 AND 9 OR EXTRACT(HOUR FROM b.tstamp) BETWEEN 16 AND 18) AND DATE(b.tstamp) = '2023-01-23' ;" -o output5b.tsv -F '\t' -A**

**A map of a city

Description automatically generated**

**5c. We want to identify and retrieve the geographical locations (longitude and latitude) and corresponding speeds of a bus from the breadcrumb table where there were significant changes in speed. Specifically, it looks for instances where the speed difference between consecutive records within the same trip exceeds 20 units (either increase or decrease). The result helps in visualizing areas and times where abrupt speed changes occurred, which could indicate events such as rapid acceleration, sudden braking, or other anomalies during the trip.**

**WITH SpeedChanges AS (SELECT longitude, latitude, trip\_id, tstamp, speed, LAG(speed) OVER (PARTITION BY trip\_id ORDER BY tstamp) AS previous\_speed,speed - LAG(speed) OVER (PARTITION BY trip\_id ORDER BY tstamp) AS speed\_change FROM breadcrumb where latitude <> 'NaN' AND longitude <> 'NaN')**

**SELECT longitude, latitude, speed FROM SpeedChanges WHERE ABS(speed\_change) > 20;**

**PGPASSWORD='Project@123' psql -h localhost -p 5432 -d postgres -U postgres -c "WITH SpeedChanges AS (SELECT longitude, latitude, trip\_id, tstamp, speed, LAG(speed) OVER (PARTITION BY trip\_id ORDER BY tstamp) AS previous\_speed,speed - LAG(speed) OVER (PARTITION BY trip\_id ORDER BY tstamp) AS speed\_change FROM breadcrumb where latitude <> 'NaN' AND longitude <> 'NaN')**

**SELECT longitude, latitude, speed FROM SpeedChanges WHERE ABS(speed\_change) > 20;" -o output5d.tsv -F '\t' -A**

**A map of a city

Description automatically generated**

## 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](mailto:rbi@pdx.edu) or mina8@pdx.edu) and TA (vysali@pdx.edu).