Team ByteBuilders

DataEng S24: Project Assignment 3

Authors -, **MANISHA Kumari** **Prathamesh Chakote**
Mohammed Taariq Amin Mansuri **Manisha Katta**

Table of Content

Github link: ChechPoint3 Github Link

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

Team ByteBuilders

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 the 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.

```
UPDATE Trip
SET route_id = ei.route_id,
      service_key = ei.service_key,
      direction = ei.direction
FROM event_info ei
WHERE Trip.trip_id = ei.trip_id
        AND Trip.vehicle_id = ei.vehicle_id;
```

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.

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.

Team ByteBuilders

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.

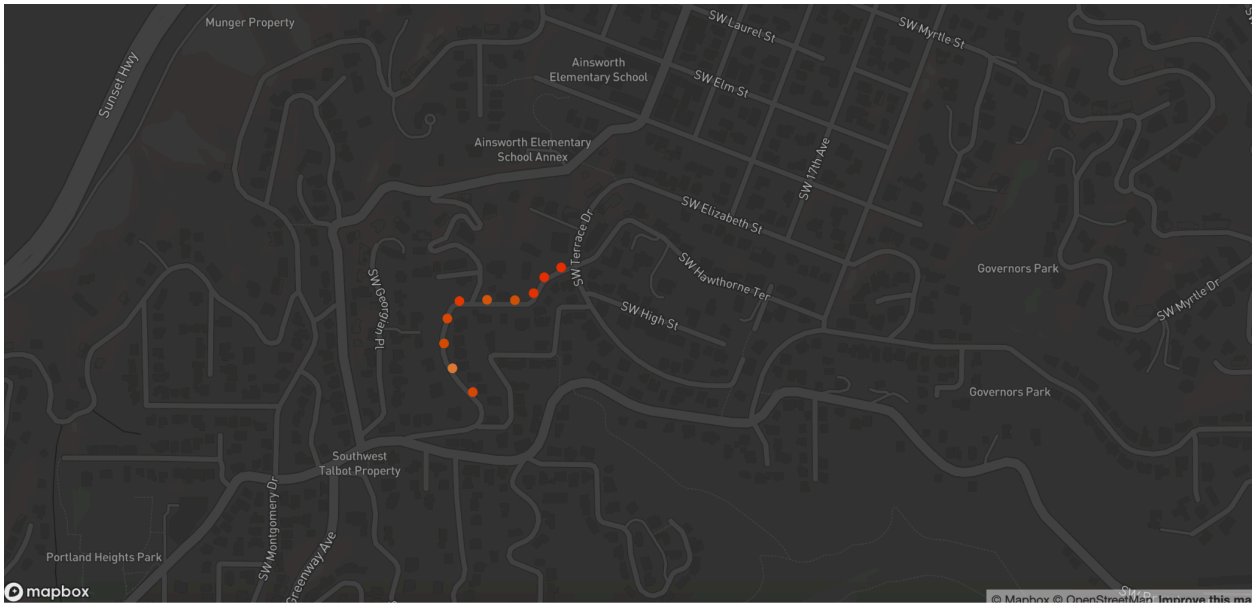
No need to produce software that neatly displays trips, routes, dates, times, etc. onto the visualization itself. Instead, just paste a screen capture of the map-based speed visualization into your submission document and then include a text description of the contents of the visualization. For example,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

Team ByteBuilders

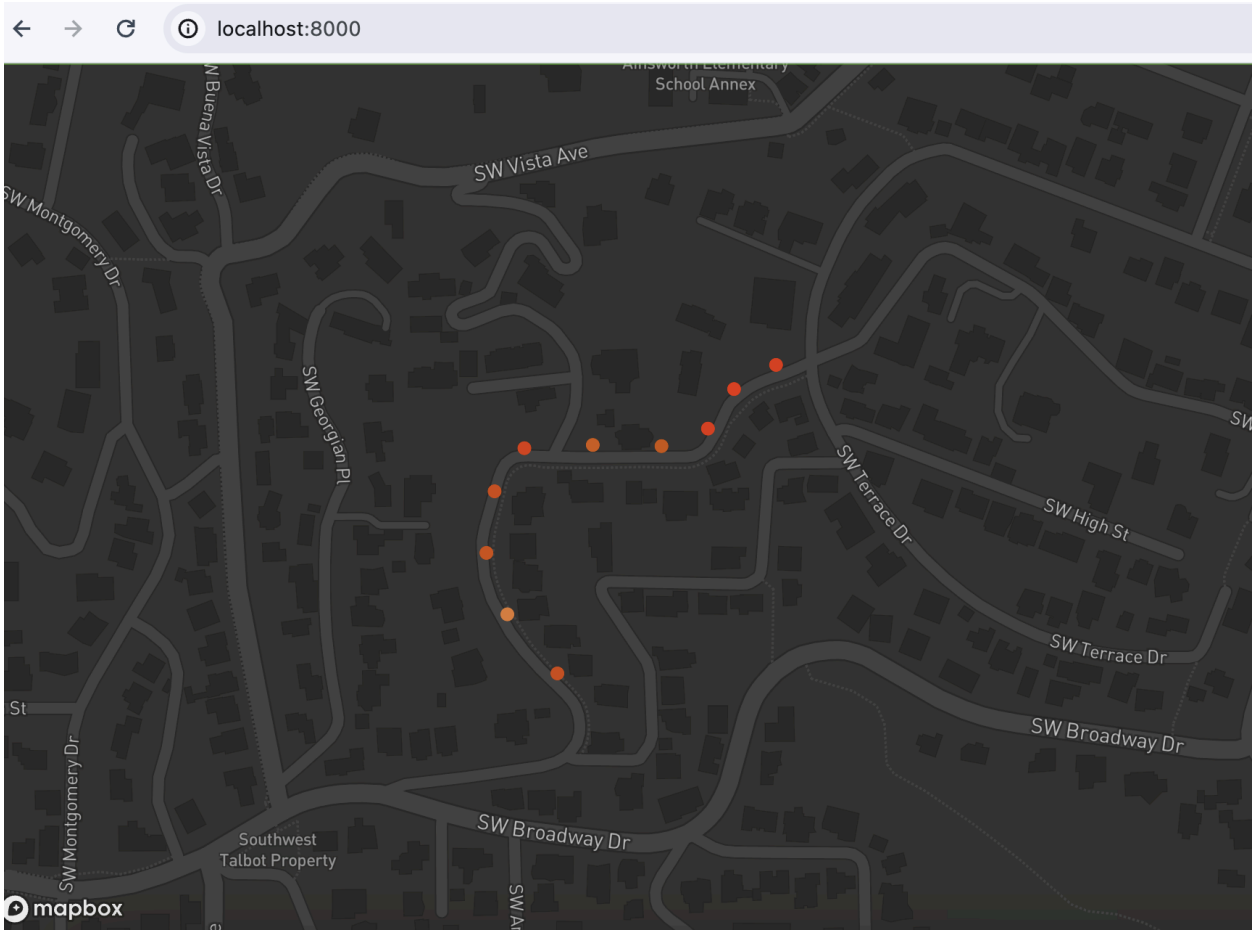
Visualization 1

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!

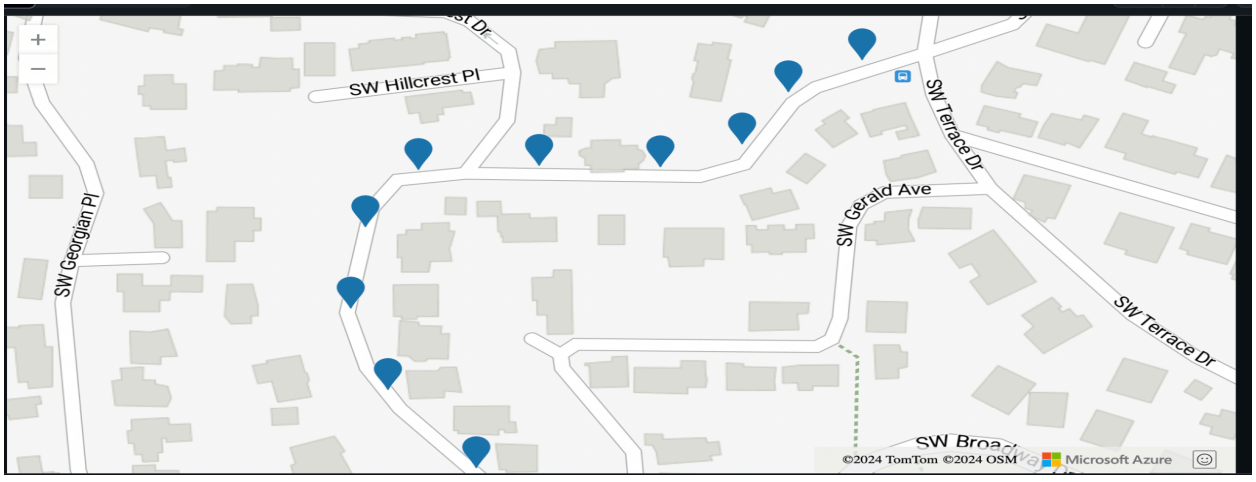
```
SELECT DISTINCT  b.longitude, b.latitude, b.speed
FROM Trip t
JOIN BreadCrumb b ON t.trip_id = b.trip_id
WHERE b.latitude BETWEEN 45.506022 AND 45.51663
  AND DATE(b.tstamp) = '2023-01-24'
  AND t.route_id = 39.0
  AND b.tstamp BETWEEN '2023-01-24 06:46:53' AND '2023-01-24 06:47:38'
  AND b.longitude BETWEEN -122.711662 AND -122.70031;
```



Team ByteBuilders



Github view:

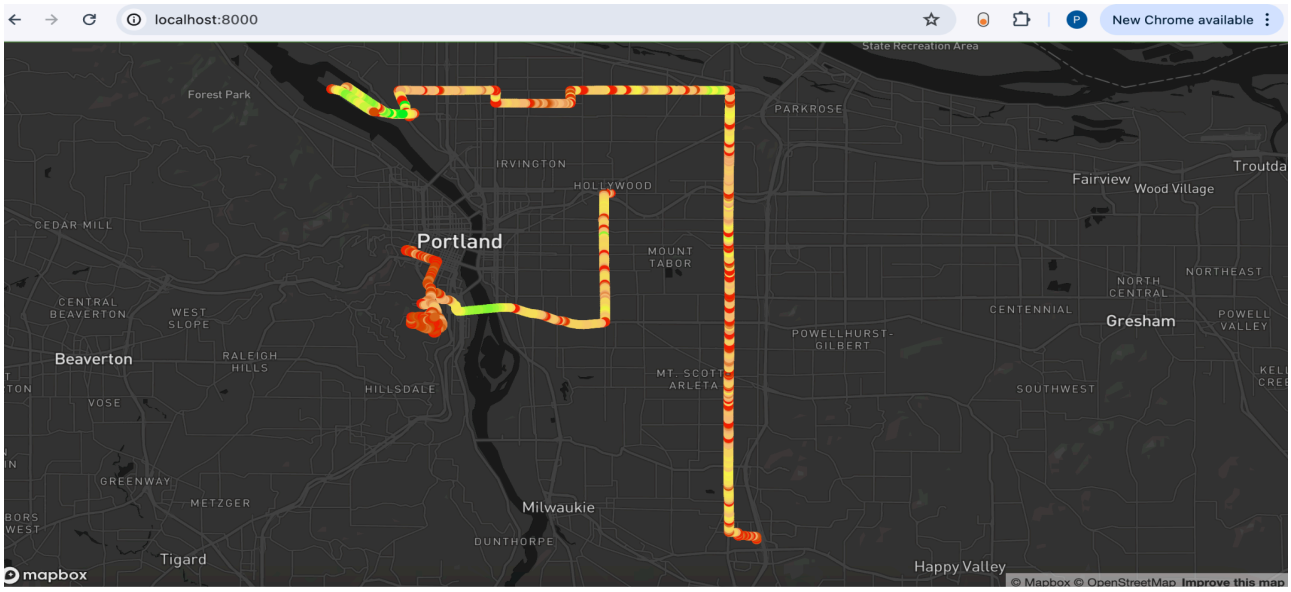


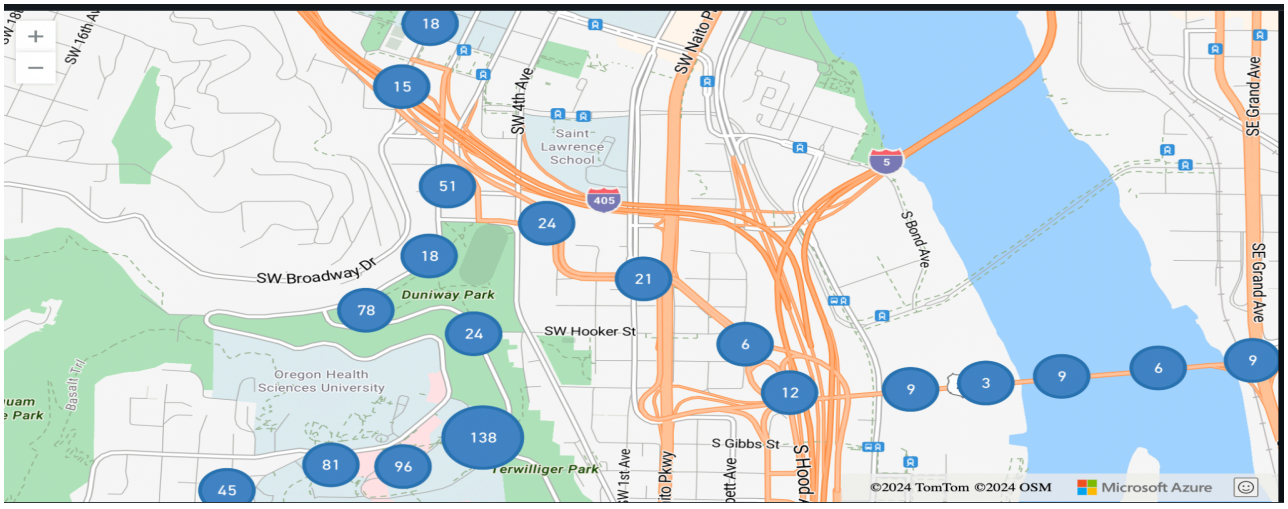
Team ByteBuilders

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 b.longitude, b.latitude, b.speed
FROM breadcrumb b
JOIN Trip t ON b.trip_id = t.trip_id
WHERE t.direction = 'Out'
        AND t.route_id = 65
        AND date(b.tstamp) = date '2023-01-20'
        AND b.tstamp >= '2023-01-20 16:00:00'
        AND b.tstamp <= '2023-01-20 18:00:00';
```





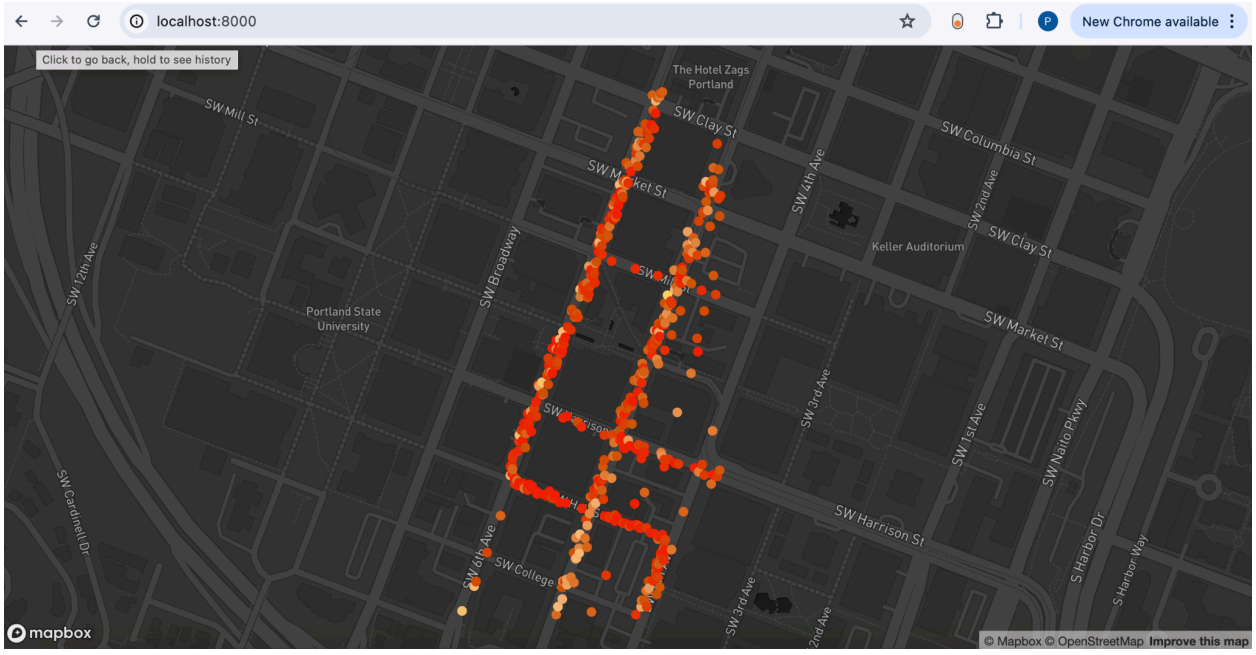
Team ByteBuilders

Visualization 3.

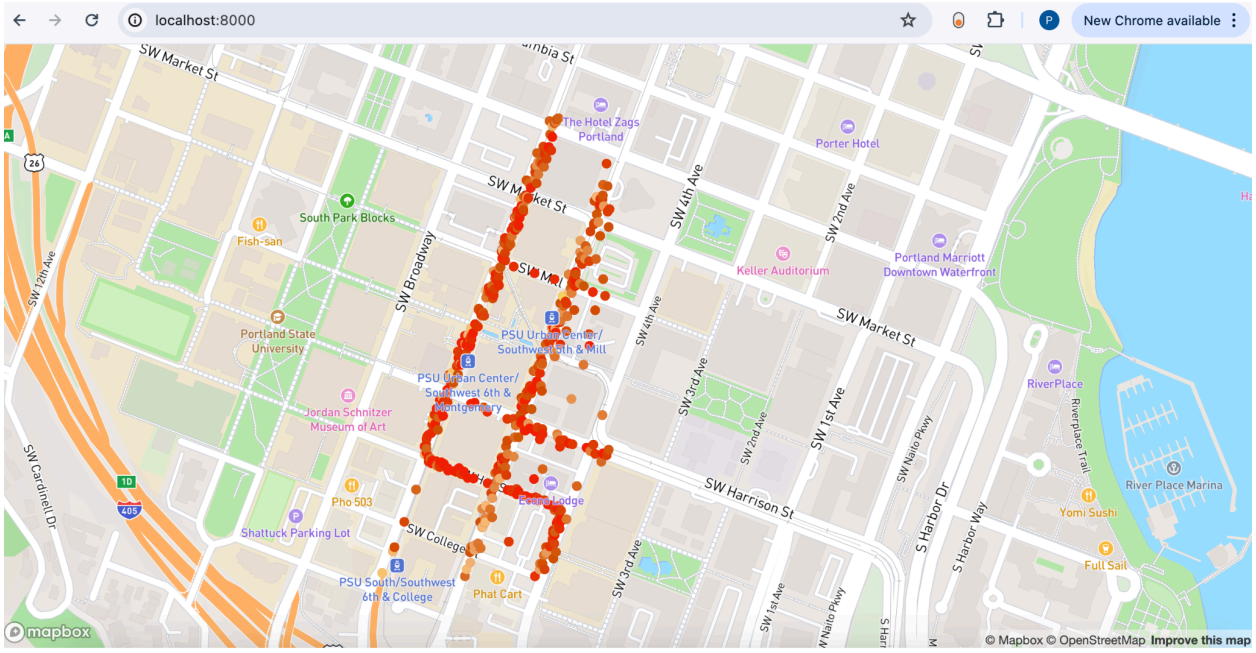
All trips that travel to and from PSU campus on any Sunday morning (you choose which Sunday) between 9am and 11am.

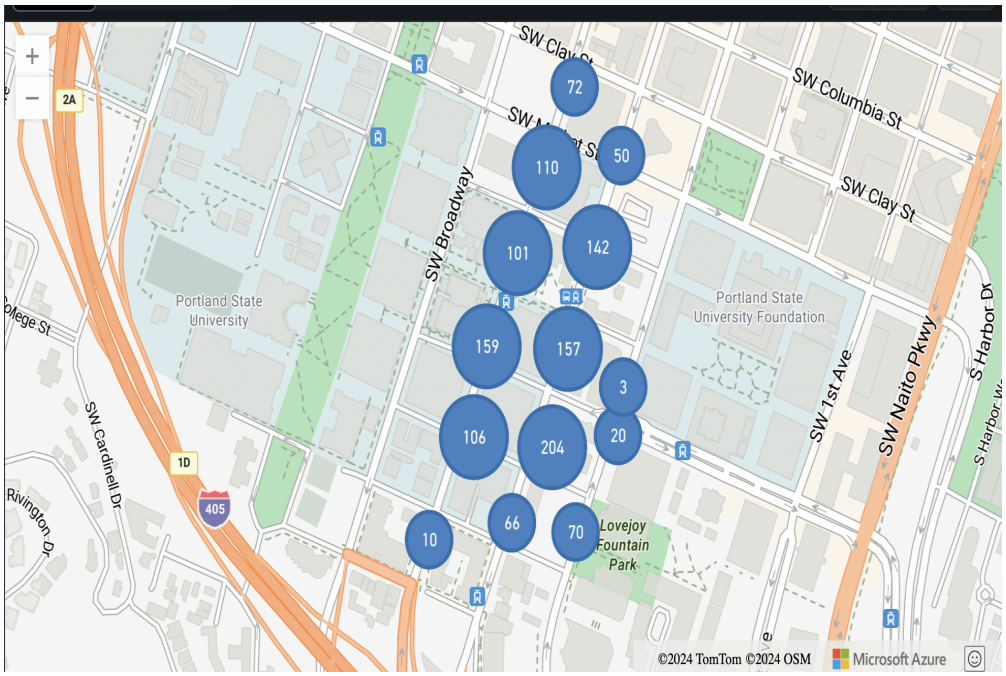
Due to cron job failure we missed the data for sunday
We don't have data for sunday so we are doing for Friday 20th january

```
SELECT DISTINCT b.longitude, b.latitude, b.speed
FROM Trip t
JOIN BreadCrumb b ON t.trip_id = b.trip_id
WHERE (b.latitude BETWEEN 45.50921922110612 AND 45.513848810065525)
        AND (b.longitude BETWEEN -122.68811436369573 AND -122.68063228685133)
        And date(b.tstamp) = date '2023-01-20'
        AND b.tstamp >= '2023-01-20 09:00:00'
        AND b.tstamp <= '2023-01-20 11:00:00';
```



Team ByteBuilders



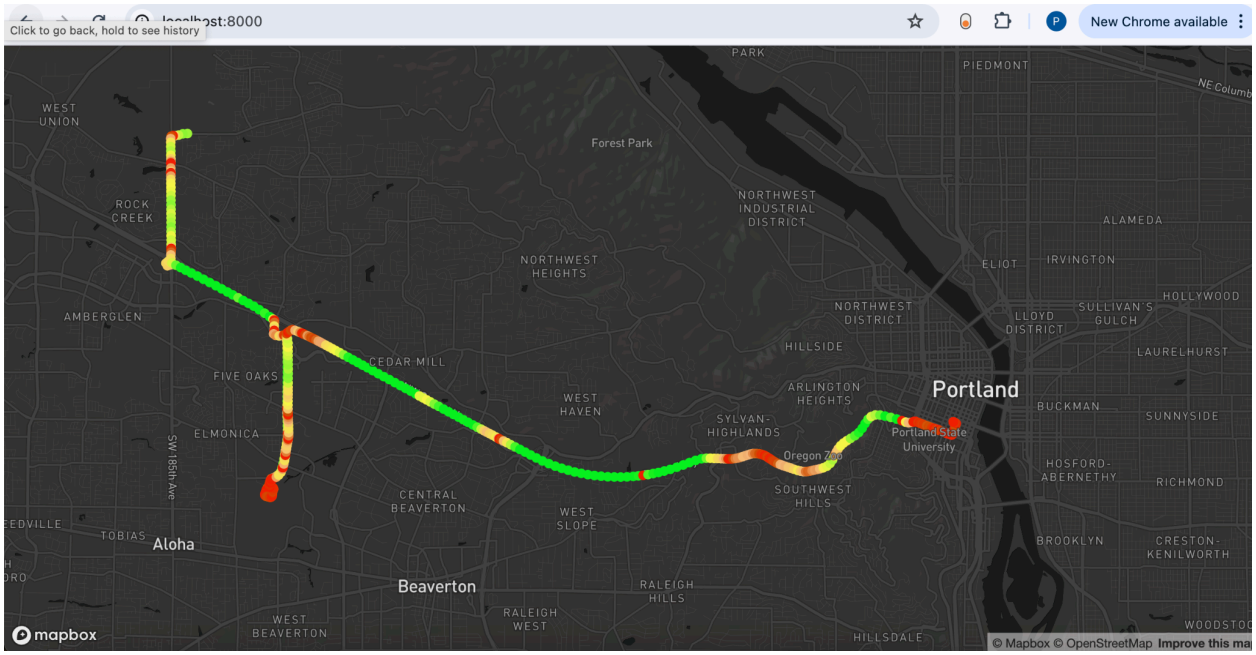


Team ByteBuilders

Visualization 4.

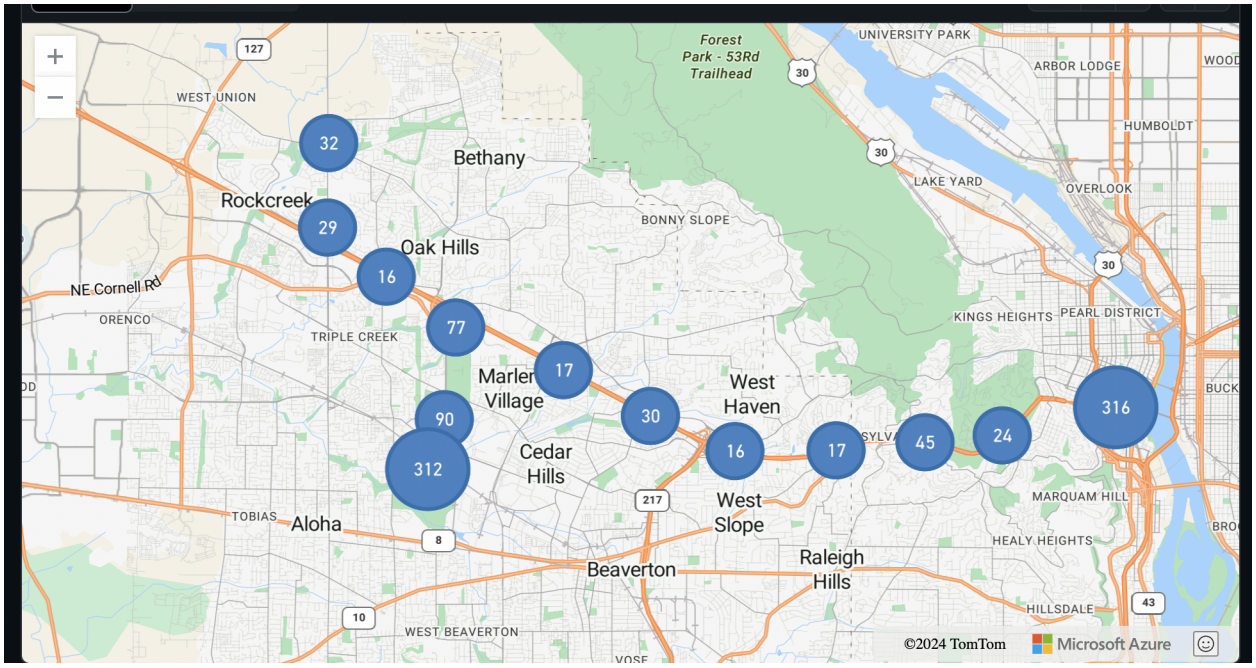
The longest (as measured by time) trip in your entire data set. Indicate the date, route #, and the trip ID of the trip along with a visualization showing the entire trip.

```
SELECT
    t.trip_id,
    t.route_id,
    t.service_key,
    t.direction,
    MAX(b.tstamp) - MIN(b.tstamp) AS trip_duration,
    ARRAY_AGG(b.latitude ORDER BY b.tstamp) AS latitude_points,
    ARRAY_AGG(b.longitude ORDER BY b.tstamp) AS longitude_points,
    ARRAY_AGG(b.speed ORDER BY b.tstamp) AS speed_points
FROM Trip t
JOIN BreadCrumb b ON t.trip_id = b.trip_id
GROUP BY t.trip_id, t.route_id, t.service_key, t.direction
ORDER BY  trip_duration DESC
LIMIT 1;
```



Team ByteBuilders

Github view:



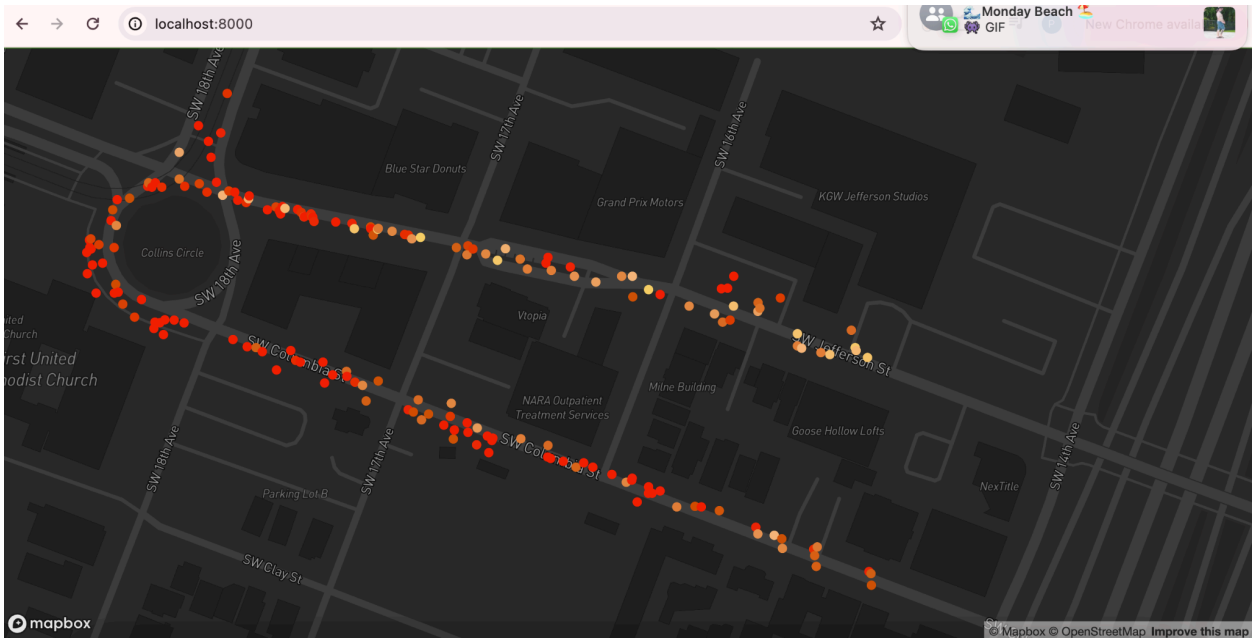
Visualization 5a, 5b, 5c, …. Three or more additional visualizations of your choice. Indicate why you chose each particular visualization.

Team ByteBuilders

Visualization 5A

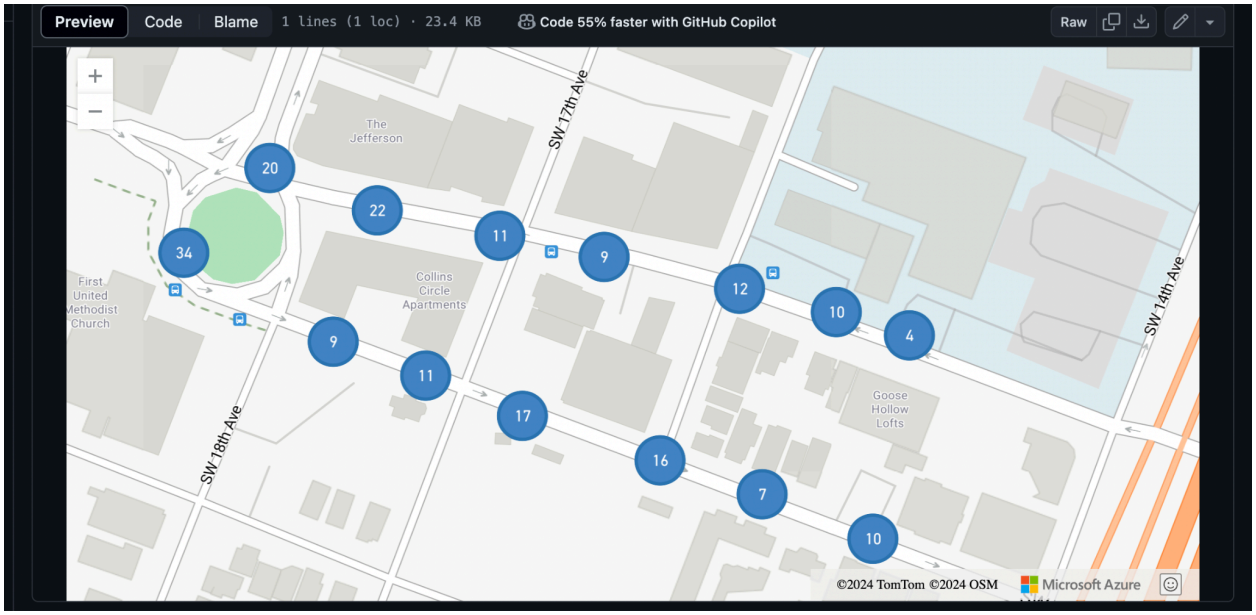
This visualization shows us all the trimet points around goose Hollow which our residency, on monday morning from 9 to 7. We were curious of the amound of trimet points around our home and its distribution so we thought of querying it and visualizing it out.

```
SELECT DISTINCT b.longitude, b.latitude, b.speed FROM Trip t
JOIN BreadCrumb b ON t.trip_id = b.trip_id
WHERE (b.latitude BETWEEN 45.51552896736652 AND 45.51829824128865)
AND (b.longitude BETWEEN -122.69432826013855 AND -122.6893552049442)
And date(b.tstamp) = date '2023-01-23'
AND b.tstamp >= '2023-01-23 09:00:00'
AND b.tstamp <= '2023-01-23 17:00:00';
```



Team ByteBuilders

Github:



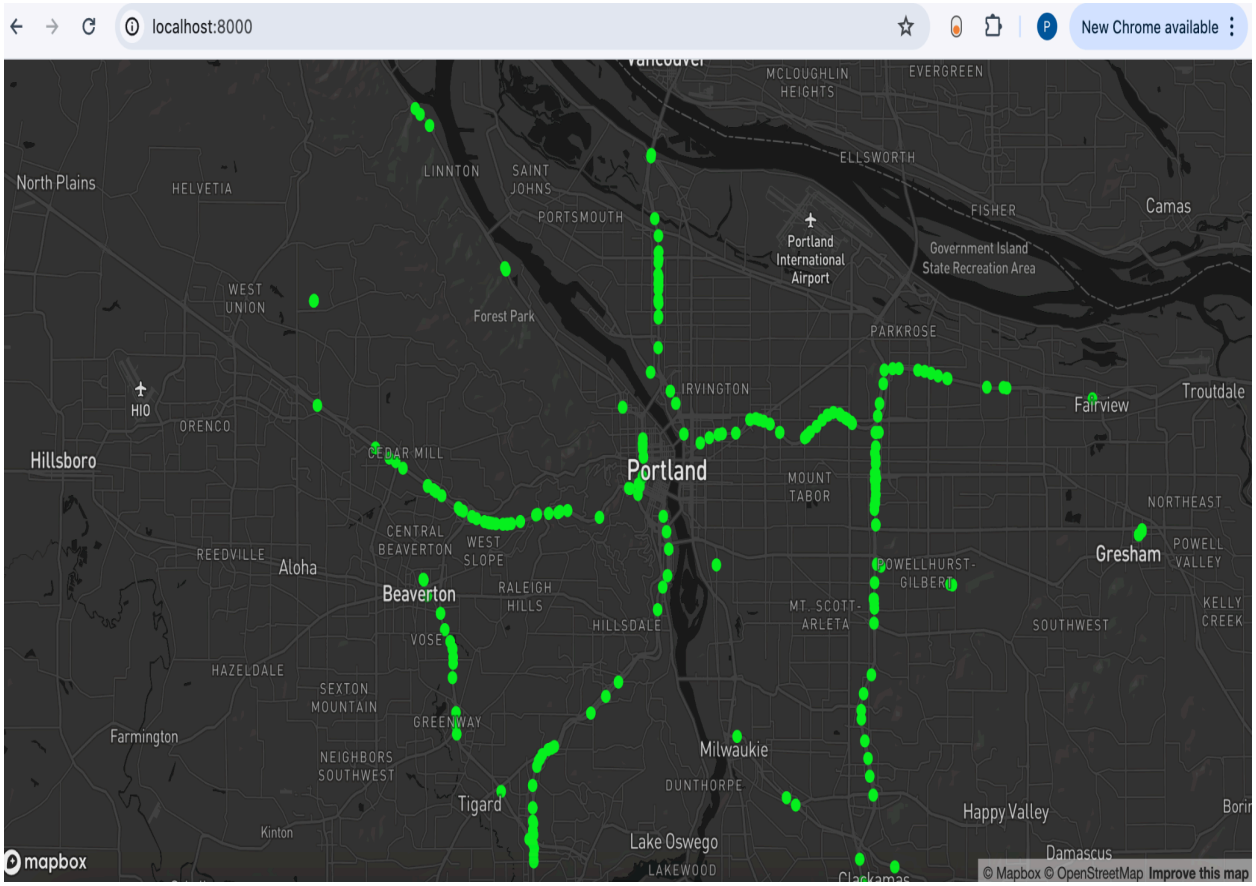
Team ByteBuilders

Visualization 5B

Data for vehicle, whose speed ranging from 30 to 50

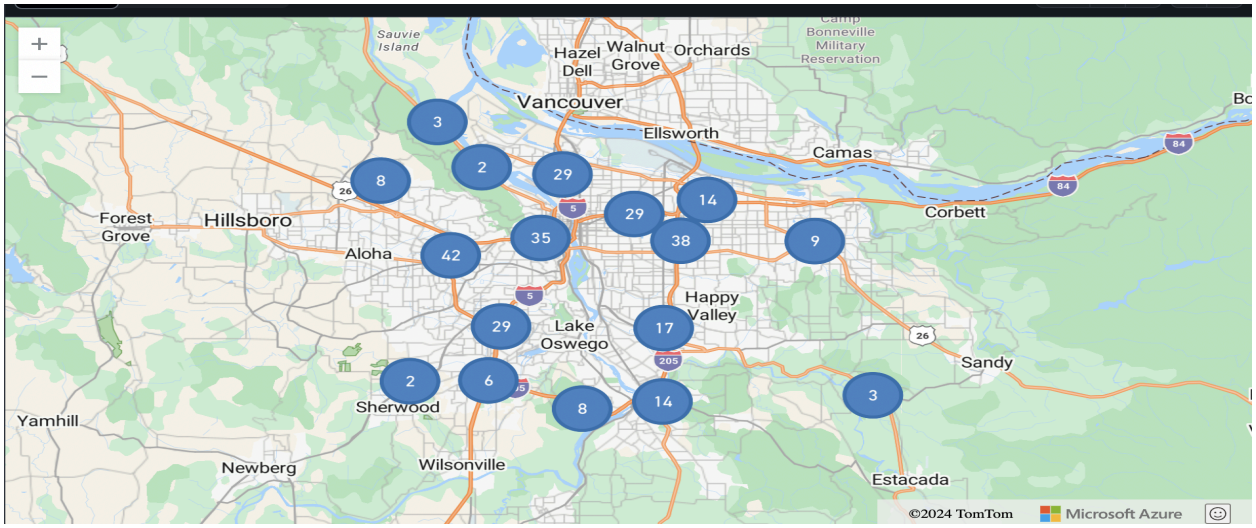
Reason: This visualization displays data for a vehicle with speeds ranging from 30 to 50 mph, illustrating its speed variations over time or distance. It helps in analyzing the vehicle's performance within this speed range.

```
SELECT b.longitude, b.latitude, b.speed
FROM breadcrumb b
WHERE b.speed>=30 and b.speed<=50;
```



Github view:

Team ByteBuilders



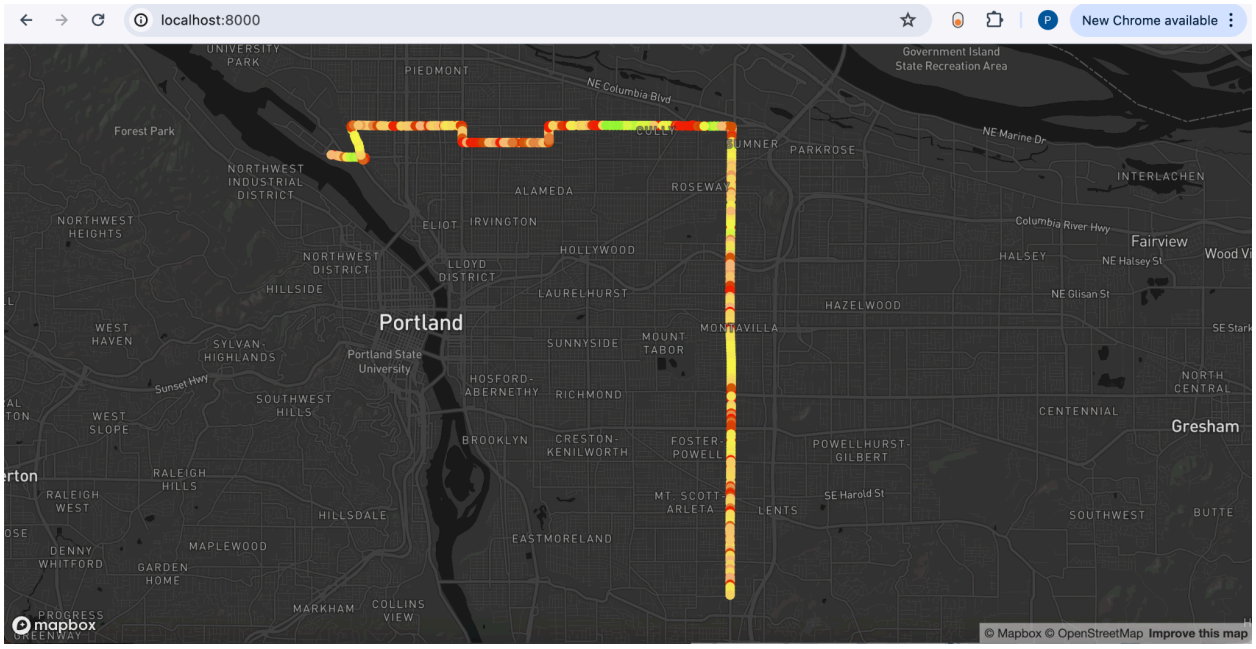
Visualization 5C

To show the path of a trip_id of 240258531

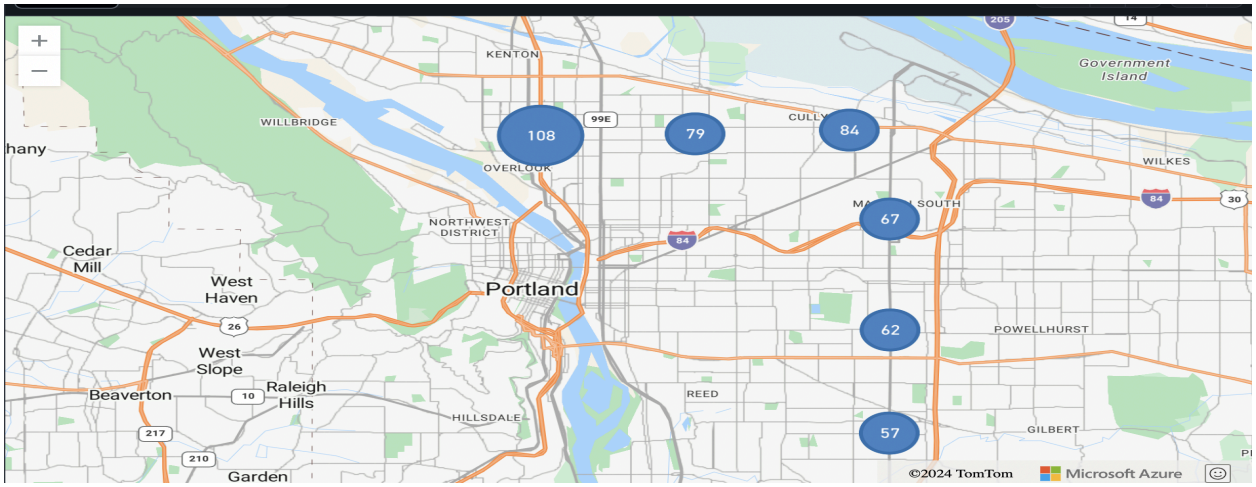
The route traveled by trip_id 240258531 is displayed in this visualization, with emphasis on the starting and finishing locations, midway stops, and overall trajectory. It facilitates geographic coverage.

```
SELECT b.longitude, b.latitude, b.speed
FROM breadcrumb b
WHERE trip_id=240258531;
```

Team ByteBuilders



Github:



Visualization 5D

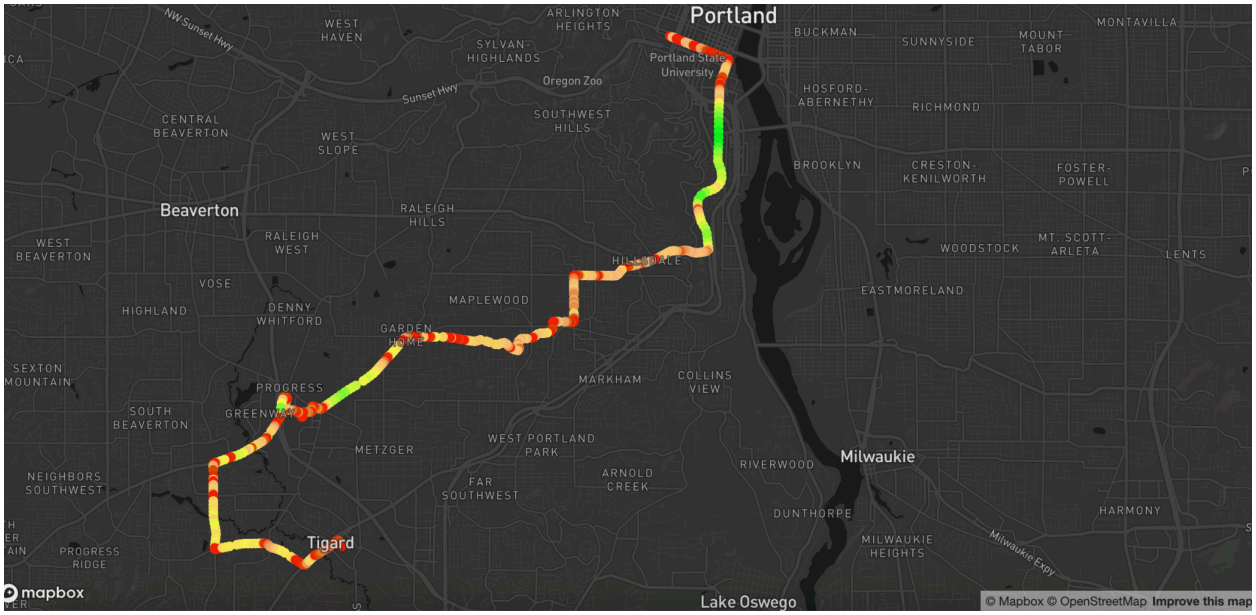
The trip which has the duration hours of 1

Reason: After seeing the maximum length trip, we were curious about the stops that particular bus takes in an hour journey,

```
SELECT
    t.trip_id,
    t.route_id,
    t.service_key,
    t.direction,
```

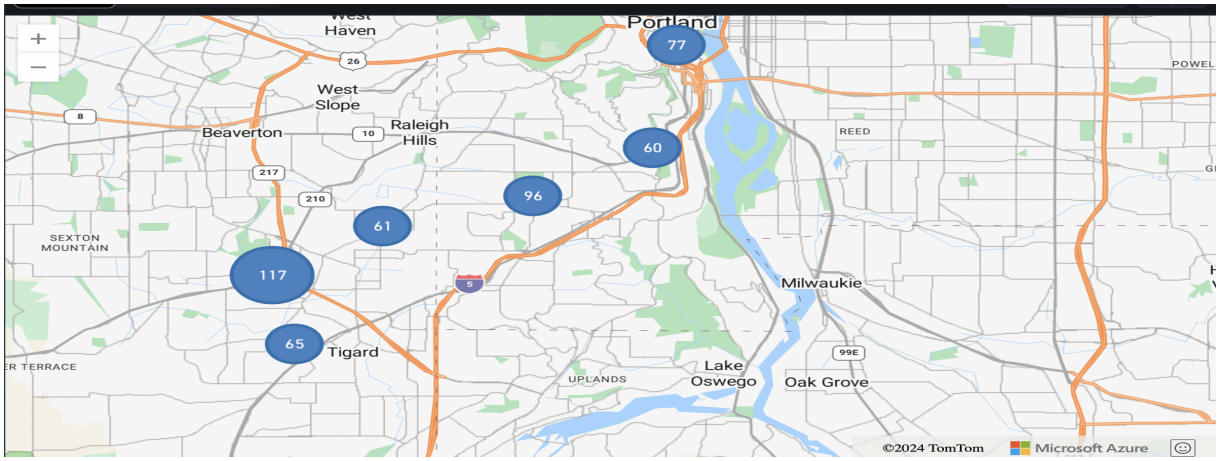
Team ByteBuilders

```
    MAX(b.tstamp) - MIN(b.tstamp) AS trip_duration,
    ARRAY_AGG(b.latitude ORDER BY b.tstamp) AS latitude_points,
    ARRAY_AGG(b.longitude ORDER BY b.tstamp) AS longitude_points,
    ARRAY_AGG(b.speed ORDER BY b.tstamp) AS speed_points
FROM
    Trip t
JOIN
    BreadCrumb b ON t.trip_id = b.trip_id
GROUP BY
    t.trip_id, t.route_id, t.service_key, t.direction
HAVING
     MAX(b.tstamp) - MIN(b.tstamp) BETWEEN INTERVAL '1 hour' AND INTERVAL '1 hour 30
minutes'
ORDER BY
    trip_duration ASC
Limit 1;
```



Team ByteBuilders

Github:



Your Code

Provide a reference to the repository where you store your Python code.

[ChechPoint3 Github Link]

If you are keeping it private then share it with the Professor (rbi@pdx.edu or mina8@pdx.edu) and TA (vysali@pdx.edu).

Extra credit:

For this activity we have named our bucket as “activity_2”
Our subscriber pulls the data from the pub/sub and pushes the data to the gcp bucket in original form i.e without any validatiosn or transformations in it

1. We have a file archiver.py which consist of code used to compress and push the data in form of a single csv file inside the bucket “activity_2”
2. This file is imported in subscriber for part2 anf part3, which then used to push the data to the bucket onces the timeout expires for subscriber.

Team ByteBuilders

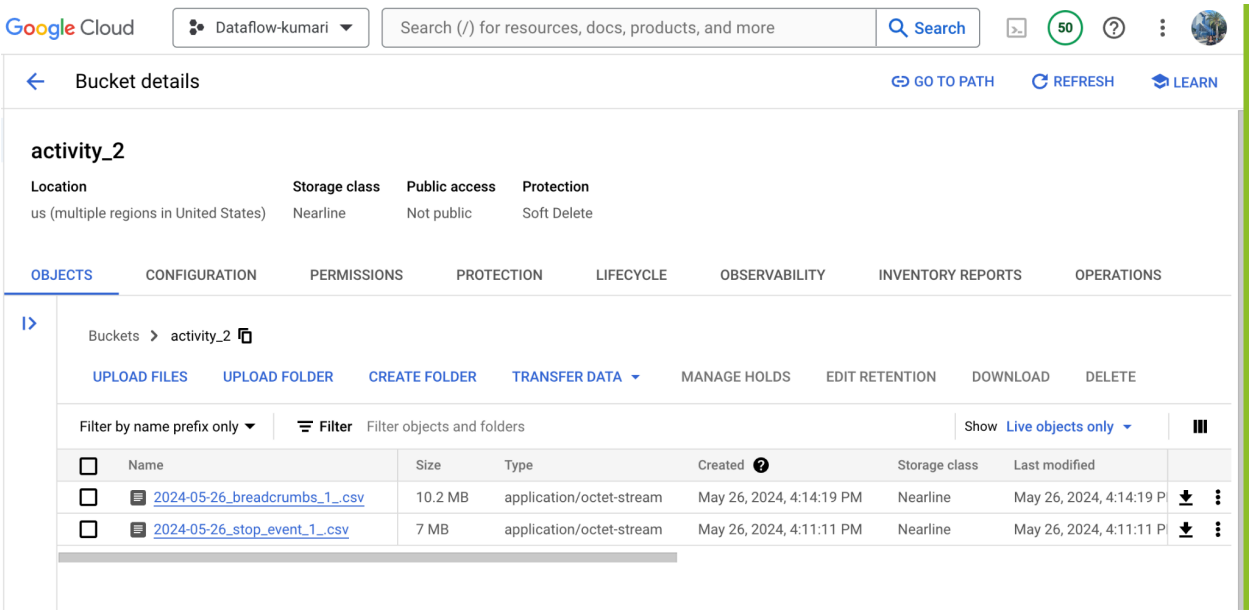


Image explanation: this image shows two CSV files, one for breadcrumbs whose size after compression is 10.2MB, and the other for event stop data, which is around 7MB.