

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
5/24/23	Wednesday	152874	529

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

This value is the "trip's index". The value of this seems to be incremental, and as of 5/11/23, is a minimum of 230000000. It tends to be very stagnant compared to the other values, but does change throughout the data of the day. This number is randomly generated and assigned to each trip.

EVENT_NO_STOP

They call this value the “stop of point” index. Based on that, it seems to be used to indicate points where the vehicle stopped, either to drop off or pick up. Oddly enough, this seems to not correlate with the “meters” variable. As meters increase, the stop doesn’t necessarily update as well. However, it’s very correlated with the “EVENT_NO_TRIP”, also having a minimum value of 230000000 as of 5/11/23. The value is always based off EVENT_NO_TRIP in some way.

OPD_DATE

This value indicates the date that the records were recorded. Every time records are updated for the day, the operation date is changed. Notably, the format is in DDMonthYYYY:00:00:00, even though the time (i.e. HH:MM:SS) never actually changes and remains 00:00:00 through all the records.

VEHICLE_ID

The value is likely the way of indicating the unique vehicles in the data. As a note, this number can be repeated across multiple breadcrumbs!

METERS

This value takes the value obtained through the odometer, which explains quite a bit... It’s best to assume that as a result, distance is measured in miles. Something worth noting is that this value still goes up, even if the vehicle appears to be still. However, there are no moments in the 5/11/23 data where meters begin at 0, and instead starts at 31145.

ACT_TIME

This value indicates the actual time of day that the vehicle was recorded. Remember that this time is measured in seconds, and seconds alone. So it may be necessary to convert this time, and then add it to the operation date, as a way of faster processing.

GPS_LONGITUDE

One half of the directionals needed to pinpoint the exact location of the bus at every stop. As of 5/11/23, the data consistently marks the longitude as somewhere between -180 and 180. Note that this is horizontal, measuring from east to west. NOTE: They can have NaN. Longitude and latitude must appear together or not at all.

GPS_LATITUDE

One half of the directionals needed to pinpoint the exact location of the bus at every stop. As of 5/11/23, the data consistently marks the latitude as somewhere between -90 and 90. Note that this is vertical, measuring from north to south. NOTE: They can have NaN. Longitude and latitude must appear together or not at all.

GPS_SATELLITES

This data field’s goal is to track the number of satellites tracking the vehicle at the moment of stopping. The values have been found to range from 0 and 32, but can also be NaN.

GPS_HDOP

They call it the “horizontal dilution of precision”. Values have been found to be ranged between 0.7 and 1.5, at least as of 5/11/23 data.

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. Each GPS reading should have an ‘event_no_trip’ that represents a single run of a single bus servicing a single route.
2. Each GPS reading should have an ‘event_no_stop’.
3. ‘event_no_trip’ must have a single ‘vehicle_id’.
4. ‘Vehicle_id’ can have multiple ‘event_no_trip’.
5. ‘Act_time’ increases by 5 after each row.
6. Each GPS reading should have an ‘opd_date’ representing the operation day.
7. Each GPS reading should have a ‘vehicle_id’ representing the vehicle number.
8. Each GPS reading should have ‘meters’ representing the odometer reading.
9. Each GPS reading should have an ‘act_time’ representing the number of seconds since midnight.
10. Each GPS reading should have a ‘gps_longitude’ that represents the latitude of the vehicle’s GPS position.
11. Each GPS reading should have a ‘gps_latitude’ that represents the latitude of the vehicle’s GPS position.
12. Each GPS reading should have a ‘gps_satellites’ representing the number of satellites used to determine the vehicle’s GPS position.
13. Each GPS reading should have a ‘gps_hdop’ representing the horizontal dilution of precision for the GPS measurement.
14. The ‘opd_date’ column has a format DDMMYYYY:HH:MM:SS.
15. The ‘meters’ for a vehicle may not decrease over time.
16. The number of stops a vehicle makes in a single trip is determined by the unique ‘event_no_stop’ numbers within ‘event_no_trip’.
17. The ‘meters’ of a single trip may not increase by more than 230 with each reading.
18. The timestamp can be calculated by using ‘opd_date’ and ‘act_time’.
19. ‘Act_time’ should range from 1 - 86400 (1 day).
20. In the ‘opd_date’ the HH:MM:SS should be 00:00:00.
21. ‘Gps_hdop’ values may range from 0 - 20.
22. ‘Gps_hdop’ < 5 is appropriate for making accurate decisions.
23. ‘Gps_latitude’ ranges from -90 to 90.
24. ‘Gps_longitude’ is ranged from -180 to 180

25. 'Gps_satellites' may not be greater than 32 satellites.

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.

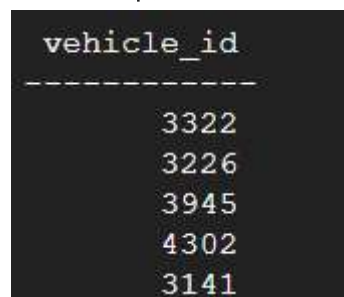
1. Speed column was created.
 - a. This took the meters and acting time between a time and its immediate next neighbor and calculated the difference between the two. The result is the speed of the vehicle at the exact moment of acting time. We want this as an additional statistic to look at for visualization.
2. Timestamp column was re-created.
 - a. This required us to separate the operation date and acting time. We stripped the date from the operation date (it was in a date/time format, not a date) and calculated the time allotted from acting time (it was in pure seconds). With these two, we get our combined timestamp and use that instead for our database. This was converted because it simplifies a lot of the querying we want and removes unnecessary columns being added, instead combining date and time together.
3. Satellite / HDOP data was removed.
 - a. The columns GPS_SATELLITES and GPS_HDOP were dropped from the original data table. For our purposes, this data was unnecessary because it's not needed for visualization later on.

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).

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

```
SELECT DISTINCT vehicle_id
FROM Trip;
```



vehicle_id
3322
3226
3945
4302
3141

b.

```

4002
3254
3234
(68 rows)

```

c.

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

- a. `SELECT COUNT(*)`
`FROM Breadcrumb`
`WHERE tstamp = '2023-1-1:00:00:00';`

```

postgres=# SELECT COUNT(*) FROM breadcrumb WHERE tstamp = '2023-1-1:00:00:00';
count
-----
      0
(1 row)

```

b.

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

- a. `SELECT COUNT(*)`
`FROM BreadCrumb`
`WHERE tstamp = '2023-1-2:00:00:00';`

```

postgres=# SELECT COUNT(*) FROM breadcrumb WHERE tstamp = '2023-1-2:00:00:00';
count
-----
      0
(1 row)

```

b.

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

- a. `SELECT AVG(*)`
`FROM (SELECT COUNT(service_key)`
`FROM breadcrumb)`
`WHERE service_key = 'Weekday';`
- b. `SELECT AVG(*)`
`FROM (SELECT COUNT(service_key)`
`FROM breadcrumb)`
`WHERE service_key = 'Saturday';`
- c. `SELECT AVG(*)`
`FROM (SELECT COUNT(service_key)`
`FROM breadcrumb)`
`WHERE service_key = 'Sunday';`

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)].

- a. `SELECT trip_id`
`FROM Breadcrumb`
`WHERE latitude BETWEEN 45.497805 AND 45.504025`
`AND longitude BETWEEN -122.566576 AND -122.563187;`

```

trip_id
-----
232473564
232473564
232473564
232473564
232473564

```

b.

```

241063248
241063248
241063248
(657 rows)

```

c.

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?

a. SELECT *

```

FROM Breadcrumb b JOIN Trip t ON b.trip_id = t.trip_id
WHERE latitude BETWEEN 45.506022 AND 45.516636
AND longitude BETWEEN -122.711662 AND -122.700316
AND service_key = 'Weekday';

```

b. SELECT *

```

FROM Breadcrumb b JOIN Trip t ON b.trip_id = t.trip_id
WHERE latitude BETWEEN 45.506022 AND 45.516636
AND longitude BETWEEN -122.711662 AND -122.700316
AND service_key = 'Sunday';

```

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

a. SELECT MAX(speed)

```

FROM Breadcrumb;

```

```

postgres=# SELECT MAX(speed) FROM breadcrumb;
          max
-----
1019.2954545454545
(1 row)

```

b.

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.

a. SELECT speed, COUNT(DISTINCT vehicle_id)

```

FROM Breadcrumb b JOIN Trip t ON b.trip_id = t.trip_id
GROUP BY speed
ORDER BY speed DESC;

```

speed	count
1019.2954545454545	1
486.67175572519085	1
439.57555555555556	1
280.67105263157896	1
272.3719806763285	1

b.

-464.11392405063293	1
-515.9620253164557	1
-559.7906976744187	1
-782.2777777777778	1
-1648.764705882353	1
(16724 rows)	

c.

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

- a. `SELECT MAX(*)`
`FROM (SELECT COUNT(trip_id)`
`FROM breadcrumb);`

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. What is the average speed of a bus?

- i. `SELECT AVG(speed)`
`FROM breadcrumb;`

```
postgres=# SELECT AVG(speed) FROM breadcrumb;
      avg
-----
 8.751509365828495
(1 row)
```

ii.

b. How many trips hit a speed of at least 30?

- i. `SELECT COUNT(t.trip_id)`
`FROM trip t JOIN breadcrumb b ON t.trip_id = b.trip_id`
`WHERE speed >= 30;`

```
postgres=# SELECT COUNT(t.trip_id) FROM trip t JOIN breadcrumb b ON t.trip_id = b.trip_id WHERE speed >= 30;
 count
-----
    130
(1 row)
```

ii.

c. How many trips occurred from May 24th to May 28th?

- i. `SELECT COUNT(timestamp)`
`FROM trip t JOIN breadcrumb b ON t.trip_id = b.trip_id`
`WHERE timestamp >= '2023-5-24:00:00:00'`
`AND timestamp <= '2023-05-28:00:00:00';`

ii.

```
postgres=# SELECT COUNT(timestamp) FROM trip t JOIN breadcrumb b ON t.trip_id = b.trip_id WHERE t
stamp >= '2023-5-24:00:00:00' AND timestamp <= '2023-05-28:00:00:00';
count
-----
      0
(1 row)
```

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).

<https://github.com/asn4psu/data-wizards-project.git>