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
06-05-2024	Monday	300532	300532
07-05-2024	Tuesday	221762	221762
08-05-2024	Wednesday	238366	238366
09-05-2024	Thursday	288229	288229
10-05-2024	Friday	341791	341791
11-05-2024	Saturday	304287	304287
12-05-2024	Sunday	322807	322807

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 field represents the event number of the trip. It is a unique identifier for each trip in the data. It is always an 8 digit integer. The current data in tables have it ranging from 228871317 to 233137897. There are a total of 3990 distinct trips identified in the data.

EVENT_NO_STOP

This field represents the event number of the trip. It is a unique identifier for each trip in the data. It is always a 9 digit integer. The current data in tables have it ranging from 228871319 to 233137914. There are hundreds of thousands of distinct stops identified in the data.

OPD_DATE

This field represents the date of the operation. It provides the date on which the trip occurred. The values in this field are in the format 'DDMMYYYY: HH:MM:SS'. The current data in tables have it ranging from 30DEC2022:00:00:00 to 05JAN2023:00:00:00. This does not give any information about time, the value is 00:00:00 for all readings.

VEHICLE_ID

This field represents the identification number of the vehicle. It uniquely identifies each vehicle in the data. It is always a 4 digit integer. The current data in tables have it ranging from 2910 to 4303. There are a total of 66 distinct vehicles identified in the data.

METERS

This field represents the distance covered in meters. It indicates the distance traveled between breadcrumb readings. The values in this field can be integers. The values in this field in the current data may range from 0 to 500,000. The number does not indicate the total distance covered by the vehicle in a single trip, but rather helps further in calculating the distance covered by it in each trip or over the course of many trips during the whole day.

ACT_TIME

This field represents the time of the breadcrumb reading. It provides information on when the breadcrumb reading was recorded. It is not a timestamp field by itself. It represents the number of seconds which have passed since 12 AM of a particular date. The values are always integers. The values in this field may range from 0 to 100,000 where any value greater than 86,400 means that it has crossed 24 hours from OPD_DATE of that reading. This value can further be used to calculate the actual timestamp of the breadcrumb reading.

GPS_LONGITUDE

This field represents the longitude coordinates obtained from GPS. It indicates the east-west position of the vehicle at the time of the breadcrumb reading. The values in this field are in decimal degrees format. The values in this field for the data in the tables range between -124 and -122, providing the bounds of the geographic area covered by the data. The negative sign in the value suggests that the vehicle is in the western hemisphere.

GPS_LATITUDE

This field represents the longitude coordinates obtained from GPS. It indicates the north-south position of the vehicle at the time of the breadcrumb reading. The values in this field are in decimal degrees format. The values in this field for the data in the tables range between 45 and 46, providing the bounds of the geographic area covered by the data. The positive sign in the value suggests that the vehicle is in the northern hemisphere.

GPS_SATELLITES

This field represents the number of GPS satellites used for obtaining the breadcrumb reading. It indicates the strength of the GPS signal during the reading. The number of satellites used can vary for different breadcrumb readings. The values in this field can be integers. The values for the current data in the tables range from 0 to 12.

GPS_HDOP

This field represents the Horizontal Dilution of Precision (HDOP) obtained from GPS. The values in this field can be floats or decimals. It indicates the accuracy of the GPS horizontal positioning. The values in this field can vary, and a lower value indicates higher accuracy. The values for current data in the table range from 0 to 26.

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.

Assertion 1: Check that all speed values are non-negative and reasonable.
Assertion 2: Ensure SPEED column is in the correct format.
Assertion 3: For 'EVENT_NO_TRIP' format: 9 digits long and starts with '2'.
Assertion 4: For 'VEHICLE_ID' format: must be an int and exactly 4 digits.
Assertion 5: Existence Check for Essential Columns
Assertion 6: Valid GPS Coordinate Check Intra-Record Check
Assertion 7: Distribution Check
Assertion 8: tstamp is a string and in the format DD MMM YYYY: HH:MM:SS
Assertion 9: Each event_no_trip should have only one unique vehicle_id.
Assertion 10: Meters covered by any vehicle cannot be negative.
Assertion 11: There cannot exist a trip without a vehicle id.
Assertion 12: At Least 50 trips for every vehicle in a day.
Assertion 13: Direction of each trip which took place cannot be anything other than 0 or 1.
Assertion 14: More than 50% of vehicles speed will be under 15 mph.
Assertion 15: Ensure that service_key is one of the specified valid values ('Weekday', 'Saturday', or 'Sunday')
Assertion 16: GPS_HDOP values are always decimals within the range of 0 to 24.
Assertion 17: Hundreds of trips take place each day.
Assertion 18: Every trip will have different stops which recorded details at different locations (latitude and longitude).
Assertion 19: Year should be within 2022 and 2023.
Assertion 20: Data completeness check i.e. whether data contains any null values.

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 have transformed a few columns from the consumed data like OPD_DATE and ACT_TIME and combined them to get a new column TIMESTAMP in a meaningful way and calculated the speed of each vehicle using METERS and TIMESTAMP values.
- Populated speed column by 0 by default before actual speed calculation to avoid NaN speed value for the first record of every trip.
- Used math. inNan to make sure we do not divide by NaN or null value to avoid undefined or infinite values in speed columns.

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 breadcrumb reading events occurred on January 1, 2023?

   SELECT COUNT(*) AS Jan1st2023 FROM breadcrumb WHERE DATE(tstamp)='2023-01-01';

```
postgres=# SELECT COUNT(*) AS Jan1st2023
FROM breadcrumb
WHERE DATE(tstamp)='2023-01-01';
 jan1st2023
------------
     222486
(1 row)

postgres=#
```

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

   SELECT COUNT(*) AS Jan1st2023 FROM breadcrumb WHERE DATE(tstamp)='2023-01-02';

```
postgres=# SELECT COUNT(*) AS Jan2nd2023
postgres-# FROM breadcrumb
postgres-# WHERE DATE(tstamp)='2023-01-02';
 jan2nd2023
------------
     212364
(1 row)

postgres=#
```

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

   SELECT COUNT (*)/7 AS avg_readings FROM breadcrumb WHERE DATE(tstamp) BETWEEN '2022-12-30' AND '2023-01-05';

```
postgres=# SELECT COUNT (*)/7 AS avg_readings
FROM breadcrumb
WHERE DATE(tstamp) BETWEEN '2023-01-06' AND '2023-01-12';
 avg_readings
--------------
       286469
(1 row)

postgres=#
```

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.trip_id FROM trip t
   JOIN breadCrumb b ON t.trip_id = b.trip_id
   WHERE DATE(b.tstamp) = '2023-01-01'
   AND b.latitude BETWEEN 45.497805 AND 45.504025
   AND b.longitude BETWEEN -122.566576 AND -122.563187;

```
postgres=# SELECT COUNT (*)/7 AS avg_readings
FROM breadcrumb
WHERE DATE(tstamp) BETWEEN '2023-01-06' AND '2023-01-12';
 avg_readings
--------------
       286469
(1 row)

postgres=# SELECT DISTINCT t.trip_id FROM trip t
postgres-# JOIN breadCrumb b ON t.trip_id = b.trip_id
postgres-# WHERE DATE(b.tstamp) = '2023-01-01'
postgres-# AND b.latitude BETWEEN 45.497805 AND 45.504025
postgres-# AND b.longitude BETWEEN -122.566576 AND -122.563187;
  trip_id
-----------
 229857293
 229984997
 230021750
 230039164
 230046481
 230118926
 230185893
 230194170
 230503903
(9 rows)

postgres=#
```

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?

   SELECT * FROM breadcrumb

```
WHERE EXTRACT (DOW FROM tstamp) = 1
AND EXTRACT (HOUR FROM tstamp) BETWEEN 16 AND 18
AND latitude BETWEEN 45.506022 AND 45.516636
AND longitude BETWEEN -122.711662 AND -122.700316
ORDER BY tstamp;
```

tstamp	latitude	longitude	speed	trip_id
2023-01-02 16:06:02	45.507017	-122.710748	20.64	230711943
2023-01-02 16:06:06	45.507362	-122.709738	5.76	230711943
2023-01-02 16:06:12	45.508255	-122.708615	21.8	230711943
2023-01-02 16:06:17	45.509133	-122.708002	21.8	230711943
2023-01-02 16:06:22	45.510028	-122.707378	22	230711943
2023-01-02 16:06:27	45.510812	-122.706565	23.1	230711943
2023-01-02 16:06:32	45.511457	-122.705432	22	230711943
2023-01-02 16:06:37	45.51195	-122.70409	22.26	230711943
2023-01-02 16:06:42	45.512492	-122.702723	23.6	230711943
2023-01-02 16:06:47	45.5133	-122.701567	24.53	230711943
2023-01-02 16:06:52	45.514275	-122.700712	25.2	230711943
2023-01-02 16:25:35	45.514783	-122.700768	15.2	230711973
2023-01-02 16:25:40	45.514163	-122.701315	16.2	230711973
2023-01-02 16:25:45	45.513527	-122.701927	17.2	230711973
2023-01-02 16:25:50	45.512947	-122.702708	17.6	230711973
2023-01-02 16:25:55	45.512477	-122.703642	17.8	230711973
2023-01-02 16:26:00	45.512058	-122.704622	17.8	230711973
2023-01-02 16:26:05	45.511627	-122.705595	18	230711973
2023-01-02 16:26:10	45.511127	-122.706498	17.8	230711973
2023-01-02 16:26:15	45.51052	-122.707255	13.96	230711973
2023-01-02 16:26:20	45.509815	-122.707837	9	230711973
2023-01-02 16:26:25	45.509052	-122.708327	19.2	230711973
2023-01-02 16:26:30	45.508285	-122.70887	18.8	230711973
2023-01-02 16:26:35	45.507603	-122.709635	13.34	230711973
2023-01-02 16:26:40	45.507175	-122.710703	19.76	230711973
2023-01-02 17:15:52	45.507103	-122.71035	19.88	230712001
2023-01-02 17:15:57	45.507678	-122.709192	11.69	230712001
2023-01-02 17:16:02	45.508513	-122.70841	20.71	230712001
2023-01-02 17:16:07	45.509447	-122.707787	22	230712001
2023-01-02 17:16:12	45.510338	-122.707113	20.6	230712001
2023-01-02 17:16:17	45.51107	-122.70622	22.08	230712001
2023-01-02 17:16:22	45.51158	-122.70513	20.2	230712001
2023-01-02 17:16:27	45.512005	-122.703958	21.44	230712001
2023-01-02 17:16:32	45.512443	-122.702853	21.03	230712001
2023-01-02 17:16:37	45.513002	-122.701943	20.88	230712001
2023-01-02 17:16:42	45.513675	-122.701228	18.7	230712001
2023-01-02 17:16:47	45.5144	-122.700642	12.55	230712001
2023-01-02 17:35:23	45.515097	-122.700542	15.93	230712027
2023-01-02 17:35:28	45.514423	-122.701132	18.56	230712027
2023-01-02 17:35:33	45.513728	-122.701765	17.1	230712027
2023-01-02 17:35:38	45.513078	-122.70254	18.6	230712027
2023-01-02 17:35:43	45.512563	-122.703503	18.7	230712027
2023-01-02 17:35:48	45.51213	-122.704525	19.15	230712027
2023-01-02 17:35:53	45.511692	-122.705533	18.53	230712027
2023-01-02 17:35:58	45.511187	-122.706468	18.2	230712027
2023-01-02 17:36:03	45.510572	-122.707247	18.49	230712027
2023-01-02 17:36:08	45.509863	-122.707845	18.2	230712027
2023-01-02 17:36:13	45.509098	-122.708337	18.6	230712027
2023-01-02 17:36:18	45.508323	-122.708857	19	230712027
2023-01-02 17:36:22	45.507633	-122.7096	24.25	230712027
2023-01-02 17:36:28	45.50719	-122.710657	21.18	230712027
2023-01-02 17:58:37	45.514637	-122.700998	20.6	230519233
2023-01-02 17:58:42	45.513853	-122.701723	19.4	230519233
2023-01-02 17:58:47	45.51312	-122.702532	21.31	230519233
2023-01-02 17:58:52	45.512552	-122.703557	19.07	230519233
2023-01-02 17:58:57	45.512085	-122.70464	18.76	230519233
2023-01-02 17:59:02	45.511625	-122.705677	17.87	230519233

2023-01-02 17:59:07	45.511107	-122.706602	19.77	230519233
2023-01-02 17:59:12	45.510505	-122.70734	18.1	230519233
2023-01-02 17:59:17	45.509815	-122.707907	18.44	230519233
2023-01-02 17:59:22	45.509053	-122.708388	18.05	230519233
2023-01-02 17:59:27	45.508282	-122.708938	21	230519233
2023-01-02 17:59:32	45.50758	-122.709698	20.62	230519233
2023-01-02 17:59:37	45.507137	-122.710782	19.75	230519233
2023-01-09 16:14:46	45.507045	-122.711187	13.8	234873166
2023-01-09 16:14:51	45.507263	-122.710322	14.4	234873166
2023-01-09 16:14:56	45.507628	-122.709535	14.4	234873166
2023-01-09 16:15:01	45.50813	-122.7089	12.2	234873166
2023-01-09 16:15:06	45.508708	-122.708432	17.6	234873166
2023-01-09 16:15:11	45.509265	-122.708035	11.2	234873166
2023-01-09 16:15:16	45.509828	-122.707635	13.8	234873166
2023-01-09 16:15:21	45.51039	-122.707165	13.48	234873166
2023-01-09 16:15:26	45.510922	-122.706562	15	234873166
2023-01-09 16:15:31	45.511373	-122.705802	15.4	234873166
2023-01-09 16:15:36	45.511753	-122.704965	15.4	234873166
2023-01-09 16:15:41	45.512113	-122.704115	15.4	234873166
2023-01-09 16:15:46	45.51248	-122.70328	15	234873166
2023-01-09 16:15:51	45.5129	-122.702517	15	234873166
2023-01-09 16:15:56	45.513398	-122.701858	15	234873166
2023-01-09 16:16:01	45.51396	-122.701302	15	234873166
2023-01-09 16:16:06	45.514548	-122.700775	18.4	234873166
2023-01-09 18:30:40	45.5152	-122.700415	11.6	235080743
2023-01-09 18:30:45	45.514717	-122.700818	12.4	235080743
2023-01-09 18:30:50	45.514202	-122.701263	13.4	235080743
2023-01-09 18:30:55	45.513653	-122.701787	14.8	235080743
2023-01-09 18:31:00	45.513107	-122.702437	15.8	235080743
2023-01-09 18:31:05	45.51264	-122.703273	16.4	235080743
2023-01-09 18:31:10	45.512238	-122.704195	16.8	235080743
2023-01-09 18:31:15	45.51184	-122.70514	17.2	235080743
2023-01-09 18:31:20	45.51139	-122.706062	17.4	235080743
2023-01-09 18:31:25	45.510845	-122.706905	18	235080743
2023-01-09 18:31:30	45.510187	-122.707582	18	235080743
2023-01-09 18:31:35	45.50944	-122.70811	18.4	235080743
2023-01-09 18:31:40	45.508657	-122.708613	19	235080743
2023-01-09 18:31:45	45.507892	-122.709232	19.4	235080743
2023-01-09 18:31:50	45.507307	-122.710188	19.8	235080743
2023-01-09 18:31:55	45.50701	-122.711373	19.4	235080743

(97 rows)

SELECT * FROM breadcrumb

WHERE EXTRACT (DOW FROM tstamp) = 0

AND EXTRACT (HOUR FROM tstamp) BETWEEN 6 AND 8

AND latitude BETWEEN 45.506022 AND 45.516636

AND longitude BETWEEN -122.711662 AND -122.700316

ORDER BY tstamp;

tstamp	latitude	longitude	speed	trip_id
2023-01-01 07:05:07	45.515072	-122.700593	19.03	230206293
2023-01-01 07:05:12	45.514355	-122.701223	18.28	230206293
2023-01-01 07:05:17	45.51366	-122.70191	13.58	230206293
2023-01-01 07:05:22	45.51302	-122.702703	18.8	230206293
2023-01-01 07:05:27	45.512527	-122.703665	17.04	230206293
2023-01-01 07:05:32	45.512105	-122.704642	18.52	230206293
2023-01-01 07:05:37	45.511683	-122.7056	18.64	230206293
2023-01-01 07:05:42	45.511185	-122.706497	14.67	230206293
2023-01-01 07:05:47	45.510575	-122.707265	18.7	230206293
2023-01-01 07:05:52	45.509855	-122.70786	13.95	230206293
2023-01-01 07:05:57	45.509072	-122.708355	16.33	230206293
2023-01-01 07:06:02	45.508305	-122.708887	18.25	230206293
2023-01-01 07:06:07	45.507693	-122.709535	18.25	230206293
2023-01-01 07:06:12	45.507335	-122.710228	17.96	230206293
2023-01-01 07:06:17	45.507112	-122.710973	12	230206293
2023-01-01 07:08:03	45.50693	-122.711307	23	230206293
2023-01-01 07:08:08	45.507298	-122.709947	22.6	230206293
2023-01-01 07:08:13	45.508205	-122.70871	28.2	230206293
2023-01-01 07:08:18	45.509158	-122.708035	23.4	230206293
2023-01-01 07:08:23	45.510003	-122.707498	20.4	230206293
2023-01-01 07:08:29	45.510853	-122.706677	19	230206293
2023-01-01 07:08:34	45.511567	-122.705497	24.2	230206293
2023-01-01 07:08:38	45.512155	-122.704125	31	230206293
2023-01-01 07:08:43	45.512797	-122.702758	25.4	230206293
2023-01-01 07:08:48	45.513647	-122.701642	25.6	230206293
2023-01-01 07:08:53	45.514597	-122.700767	24.8	230206293
2023-01-08 06:00:51	45.507183	-122.710413	21.2	234255364
2023-01-08 06:00:56	45.507742	-122.70929	21.6	234255364
2023-01-08 06:01:01	45.508597	-122.70851	22.6	234255364
2023-01-08 06:01:06	45.509518	-122.707867	23	234255364
2023-01-08 06:01:11	45.510393	-122.707155	22.2	234255364
2023-01-08 06:01:16	45.511183	-122.706132	23.6	234255364
2023-01-08 06:01:21	45.511807	-122.70482	24.4	234255364
2023-01-08 06:01:26	45.512373	-122.703497	23.8	234255364
2023-01-08 06:01:31	45.513007	-122.702297	23.4	234255364
2023-01-08 06:01:36	45.513842	-122.701355	23.6	234255364
2023-01-08 06:01:41	45.514718	-122.700568	22.4	234255364
2023-01-08 06:09:39	45.507123	-122.71053	24	234371504
2023-01-08 06:09:44	45.507715	-122.709268	23.8	234371504
2023-01-08 06:09:49	45.508642	-122.708417	24.4	234371504
2023-01-08 06:09:54	45.509673	-122.707718	25.2	234371504
2023-01-08 06:09:59	45.510643	-122.706885	25	234371504
2023-01-08 06:10:04	45.511423	-122.705658	25.8	234371504
2023-01-08 06:10:09	45.512047	-122.704218	26.2	234371504
2023-01-08 06:10:14	45.512665	-122.702852	25	234371504
2023-01-08 06:10:19	45.513473	-122.70175	24.8	234371504
2023-01-08 06:10:24	45.514432	-122.70089	24.8	234371504
2023-01-08 08:05:01	45.506917	-122.711437	23.8	234309438
2023-01-08 08:05:06	45.507303	-122.709982	24.2	234309438
2023-01-08 08:05:11	45.508083	-122.708865	24.8	234309438
2023-01-08 08:05:16	45.509073	-122.70809	25.8	234309438
2023-01-08 08:05:21	45.510148	-122.70731	26.8	234309438
2023-01-08 08:05:56	45.513363	-122.702645	22.8	234309438
2023-01-08 08:06:01	45.513432	-122.701277	21.4	234309438
2023-01-08 08:47:33	45.507028	-122.710653	23	234170126
2023-01-08 08:47:38	45.507567	-122.709385	23.2	234170126
2023-01-08 08:47:43	45.508455	-122.708503	23.8	234170126
2023-01-08 08:47:48	45.509457	-122.707822	24.6	234170126
2023-01-08 08:47:53	45.510445	-122.707037	24.8	234170126
2023-01-08 08:47:58	45.511282	-122.705865	26	234170126
2023-01-08 08:48:03	45.51193	-122.704425	26.2	234170126
2023-01-08 08:48:08	45.512583	-122.70298	26.6	234170126
2023-01-08 08:48:13	45.51342	-122.701773	26.2	234170126
2023-01-08 08:48:18	45.514373	-122.70091	24.6	234170126

(64 rows)

This location has 7 different trips going through it between 4 PM and 6 PM on Monday, whereas 4 trip between 6 AM and 8AM on Sunday. Hence the first query is giving out over twice as many readings as the second.

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

   SELECT MAX(speed) as max_speed FROM breadcrumb;

```
postgres=# SELECT MAX(speed) as max_speed FROM breadcrumb;
 max_speed
-----------
       236
(1 row)
```

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(speed) AS count

   FROM breadcrumb

   GROUP BY speed

   ORDER BY COUNT(speed) DESC;

speed	count
0	114310
10	45090
11	44500
10.2	44247
10.4	43907

   Total: 2791 Rows

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

   SELECT trip_id, trip_duration

   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;

```
postgres=# SELECT trip_id, trip_duration
postgres-# FROM (
postgres(#     SELECT trip_id, MAX(tstamp) - MIN(tstamp) AS trip_duration
postgres(#     FROM breadcrumb
postgres(#     GROUP BY trip_id
postgres(# ) AS trip_durations
postgres-# ORDER BY trip_duration DESC
postgres-# LIMIT 1;
  trip_id  | trip_duration
-----------+---------------
 234873138 | 03:42:15
(1 row)

postgres=#
```

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?

```
postgres=# SELECT COUNT(DISTINCT trip_id) AS wednesday_count
FROM breadcrumb
WHERE DATE(tstamp) = '2023-01-11';
 wednesday_count
-----------------
             777
(1 row)

postgres=# SELECT COUNT(DISTINCT trip_id) AS saturday_count
FROM breadcrumb
WHERE DATE(tstamp) = '2023-01-07';
 saturday_count
----------------
            566
(1 row)

postgres=# SELECT COUNT(DISTINCT trip_id) AS holiday_count
FROM breadcrumb
WHERE DATE(tstamp) = '2023-01-01';
 holiday_count
---------------
           554
(1 row)

postgres=#
```

The data suggests that TriMet sees higher trip counts on non-holiday Wednesdays, indicating increased weekday activity. Non-holiday Saturdays show a decrease in trips, typical for weekends. Holidays exhibit a moderate decrease in trips compared to regular weekdays and Saturdays, likely due to altered travel patterns. These patterns reflect varying demand for TriMet's services based on weekdays, weekends, and holidays, highlighting the need for tailored service planning and resource allocation. Analyzing such trends can optimize service efficiency and meet customer needs effectively.

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).

    Find the 5 trips with the highest average speeds.

SELECT trip_id, AVG(speed) AS average_speed

FROM breadcrumb

GROUP BY trip_id

ORDER BY average_speed DESC

LIMIT 5;

```
postgres=# SELECT trip_id, AVG(speed) AS average_speed
postgres-# FROM breadcrumb
postgres-# GROUP BY trip_id
postgres-# ORDER BY average_speed DESC
postgres-# LIMIT 5;
  trip_id  |   average_speed
-----------+--------------------
 235583587 | 25.094755244755227
 231174055 | 24.603310344827587
 230715321 | 24.320561797752816
 232076295 | 23.059477611940295
 235736172 | 22.524938271604945
(5 rows)
```

How many vehicles are there in the TriMet system?

SELECT COUNT (DISTINCT vehicle_id) AS veh_count FROM trip limit 5;

```
postgres=# SELECT COUNT (DISTINCT vehicle_id) AS veh_count
postgres-# FROM trip;
 veh_count
-----------
        92
(1 row)
```

Find the average speed and total distance traveled for each vehicle during weekday and weekend

SELECT t.vehicle_id, AVG(b.speed) AS avg_speed, SUM(b.speed) AS distance

FROM trip t

JOIN breadcrumb b ON t.trip_id = b.trip_id

WHERE DATE(b.tstamp) = '2023-01-07'

GROUP BY t.vehicle_id

ORDER BY avg_speed DESC;

vehicle_id	avg_speed	distance
3650	18.710529100529097	3536.289999999999
3206	13.345536723163839	4724.319999999999
3321	13.077952336155542	41705.590000000026
4041	11.681421052631581	2219.4700000000003
3562	11.459051094890507	4709.669999999998
3035	11.427234042553192	537.08
3169	10.769305263157896	10230.840000000002
3322	10.743749999999999	1117.35
4001	10.679975209172596	68928.55999999994
3605	10.638236686390536	35957.24000000001
3621	10.608312811108188	40491.92999999996

SELECT t.vehicle_id, AVG(b.speed) AS avg_speed, SUM(b.speed) AS distance

FROM trip t

JOIN breadcrumb b ON t.trip_id = b.trip_id

WHERE DATE(b.tstamp) = '2023-01-07'

GROUP BY t.vehicle_id

ORDER BY distance DESC;

vehicle_id	avg_speed	distance
4001	10.679975209172605	68928.56
3028	10.221577411603146	59550.90999999993
3057	8.50062573789846	57600.23999999996
3247	8.671311250190818	56805.76000000005
3914	9.63922287130466	54451.97000000003
4028	9.602910296555532	52134.19999999998
3955	8.543249578414835	50661.46999999997
4042	7.976796036333607	48299.499999999985
4065	6.66723489561731	48224.11

SELECT t.vehicle_id, AVG(b.speed) AS avg_speed, SUM(b.speed) AS distance

FROM trip t

JOIN breadcrumb b ON t.trip_id = b.trip_id

WHERE DATE(b.tstamp) = '2023-01-11'

GROUP BY t.vehicle_id

ORDER BY avg_speed DESC;

vehicle_id	avg_speed	distance
3322	14.597397441552712	99276.9
3045	9.975792626114739	74948.13000000003
3264	10.405463932310347	72557.30000000005
3645	8.626322175732218	61850.73000000001
3010	8.795097564449524	61750.38000000011
3245	8.915418970032004	61284.59
3952	9.106383774884	60839.75
3650	10.214601694915258	60266.15000000002

SELECT t.vehicle_id, AVG(b.speed) AS avg_speed, SUM(b.speed) AS distance

FROM trip t

JOIN breadcrumb b ON t.trip_id = b.trip_id

WHERE DATE(b.tstamp) = '2023-01-11'

GROUP BY t.vehicle_id

ORDER BY distance DESC;

vehicle_id	avg_speed	distance
3322	14.597397441552728	99276.9000000001
4042	13.30144781144781	3950.5299999999997
3251	12.206748823040716	44078.57000000002
3247	11.898447003247735	40300.04000000008
3519	11.846950146627567	4039.8100000000004
3416	11.145357710651835	28041.720000000016
3040	10.67663248489288	58305.09000000001
3264	10.405463932310342	72557.30000000002
3005	10.389276736493937	47115.37
4034	10.311286852589639	51762.65999999999

When comparing the maximum distance travelled on weekdays is more than the weekends but the speed is more on weekends than weekdays.

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).