DataEng: Project Assignment 2 (v 2.0)

Validate, Transform, Enhance and Store

Assignment date: January 26, 2021
Due date: February 14, 2021 @10pm PT
Submit: assignment submission form

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	# updates/insertions into your database
2021-01-13	Wednesday	376234	360518
2021-01-14	Thursday	377015	355143
2021-01-15	Friday	358546	330964
2021-01-16	Saturday	170828	165741
2021-01-17	Sunday	133973	131236
2021-01-18	Monday	133479	128504

Documentation of Each of the Original Data Fields

For each of the fields of the bread crumb 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 - Each trip has a unique 9 digit trip number. Trip number could not be null.

EVENT_NO_STOP - Stop or point index. It is an integer.

OPD_DATE - Operation Date is in format DD-MMM-YY. It could not be null. It could not be a future date.

VEHICLE_ID - This is the id of the vehicle associated with each trip. It could not be null.

METERS - It is distance traveled by vehicle from the time of previous recording. It could not be a negative number. It is used to calculate the velocity.

ACT_TIME - It is the actual time of sensor recording. It could not be 0 and lesser than 0. It could not be the same as any other record in the same trip.

VELOCITY - This is the meters / (actual time of current record - actual time of previous record). This could not be greater than 5000 and less than 0.

DIRECTION - This is the geo orientation of movement of vehicle. It could not be less than 0. And this is defaulted to 0.

RADIO_QUALITY - This indicates the signal strength. It is null when value is not populated.

GPS_LONGITUDE - This is the longitude value of location. It could not be 0.0 or lesser. It could not be null.

GPS_LATITUDE - This is the latitude value of location. It could not be 0.0 or greater. It could not be null.

GPS_SATELLITES - This is count of satellites.

GPS_HDOP - This is Horizontal Dilution of Precision. It is accuracy of location data. It should range between 0.0 and 1.0.

SCHEDULE_DEVIATION - This is the delta between estimated location vs actual location of the vehicle.

Data Validation Assertions

List 20 or more data validation assertion statements here. These should be English language sentences similar to “The VELOCITY field
exceeds 5000000”. 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 RADIO_QUALITY) that might not be used in your database schema.

1. TripId and Vehicle should not be null
2. Each TripId should be associated with only one vehicle Id
3. TripId and VehicleId should be positive integers
4. Velocity should be less than 5000
5. Velocity should be a positive integer
6. Date and actual time fields should not be null
7. Date should not be future data and should not be past than 2020.
8. Actual time should be a positive integer
9. Latitude and Longitude should not be null
10. Longitude should be a negative float value.
11. Latitude should be a positive float value.
12. Latitude and longitude should not be 0.0.
13. Each latitude and longitude should have unique date time
14. Combination of TripId and datetime is unique
15. Total Trip time should not exceed 24 hours.
16. Total number of trips on weekdays should be greater than total number of trips in weekends
17. Number of trips per week for a vehicle should be normally distributed.
18. Not all trips have the same direction
19. Velocity cannot be the same value for all the breadcrumbs of the same trip.
20. Total trips should not vary more than 50% day by day and week by week throughout weekdays.

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. Drop rows if TripId or VehicleId is null
2. Drop rows if Vehicle ID is not 4 digits long
3. Default velocity to 0 if null
4. Drop rows if velocity > 5000
5. Drop rows if date and actual time are null
6. Drop rows if latitude < 0 and longitude > 0

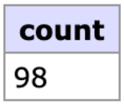
7. Default direction to 0
8. Default Service key to Weekday
9. Default route id to 0

Example Queries

Provide your responses to the questions listed in Section E 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 C-Tran system?

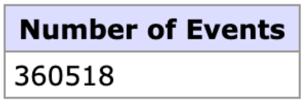
```
SELECT COUNT(DISTINCT vehicle_id)
AS Count
FROM trip
```



count
98

2. How many bread crumb reading events occurred on October 2, 2020?

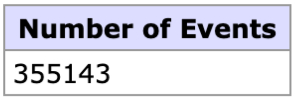
```
SELECT COUNT( * ) as "Number of Events"
FROM breadcrumb
WHERE tstamp::date = date '2020-09-02'
```



Number of Events
360518

3. How many bread crumb reading events occurred on October 3, 2020?

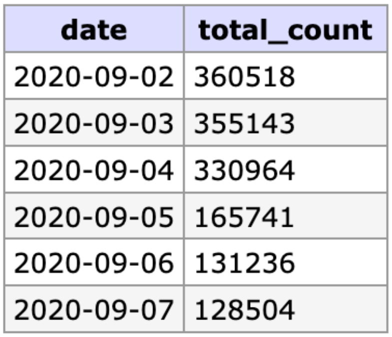
```
SELECT COUNT( * ) as "Number of Events"
FROM breadcrumb
WHERE tstamp::date = date '2020-09-03'
```



Number of Events
355143

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

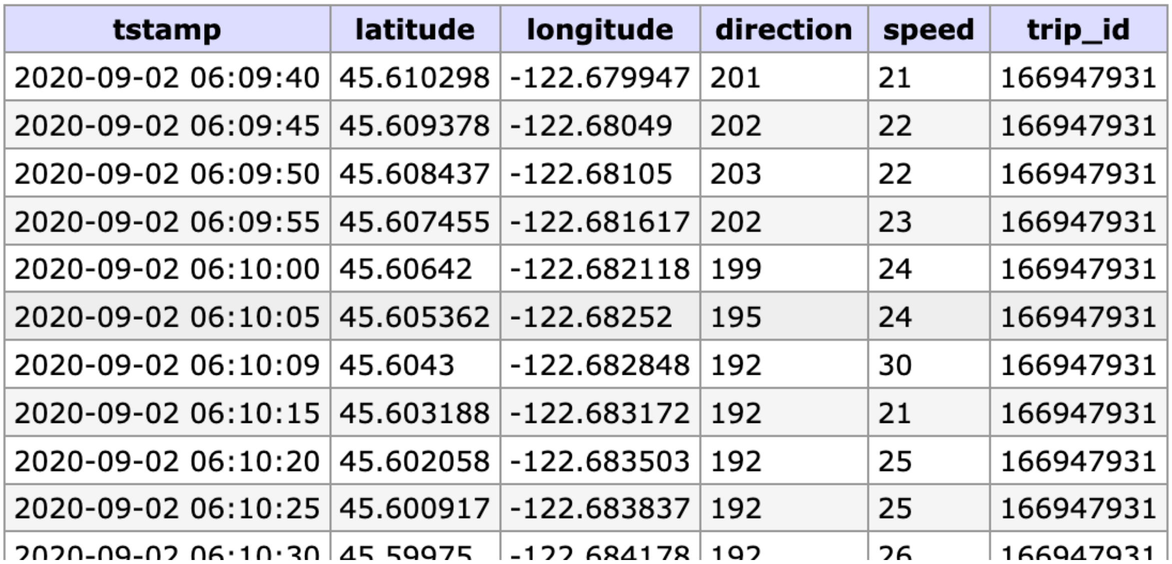
```
SELECT date(tstamp), count(tstamp) as total_count
FROM breadcrumb
GROUP BY date(tstamp)
```



date	total_count
2020-09-02	360518
2020-09-03	355143
2020-09-04	330964
2020-09-05	165741
2020-09-06	131236
2020-09-07	128504

5. List the C-Tran trips that crossed the I-5 bridge on October 2, 2020. To find this, search for all trips that have bread crumb readings that occurred within a lat/lon bounding box such as [(45.620460, -122.677744), (45.615477, -122.673624)].

```
SELECT *
FROM breadcrumb
WHERE tstamp::date = date '2020-09-02'
AND (latitude < 45.610794
AND longitude < -122.576979)
OR (latitude > 45.606989
AND longitude > -122.569501)
```



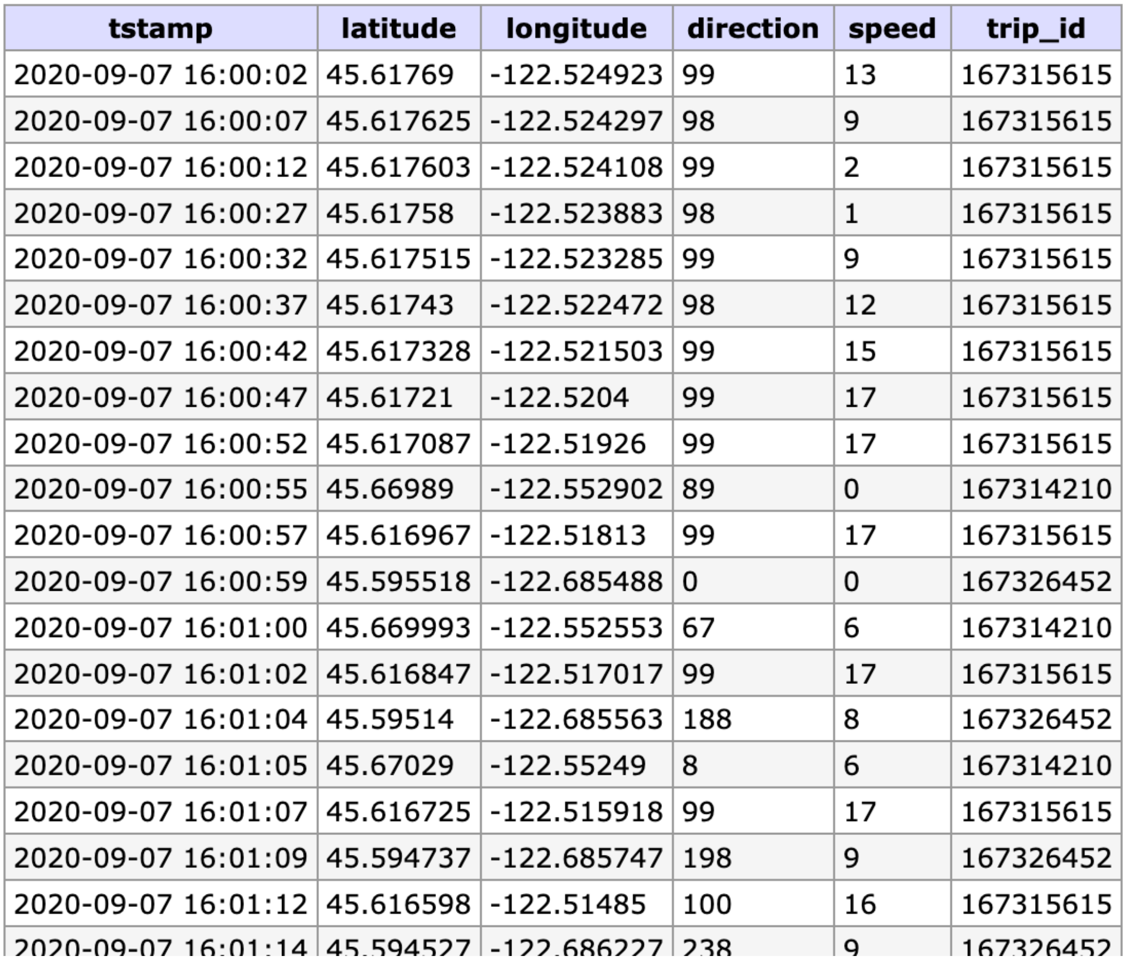
tstamp	latitude	longitude	direction	speed	trip_id
2020-09-02 06:09:40	45.610298	-122.679947	201	21	166947931
2020-09-02 06:09:45	45.609378	-122.68049	202	22	166947931
2020-09-02 06:09:50	45.608437	-122.68105	203	22	166947931
2020-09-02 06:09:55	45.607455	-122.681617	202	23	166947931
2020-09-02 06:10:00	45.60642	-122.682118	199	24	166947931
2020-09-02 06:10:05	45.605362	-122.68252	195	24	166947931
2020-09-02 06:10:09	45.6043	-122.682848	192	30	166947931
2020-09-02 06:10:15	45.603188	-122.683172	192	21	166947931
2020-09-02 06:10:20	45.602058	-122.683503	192	25	166947931
2020-09-02 06:10:25	45.600917	-122.683837	192	25	166947931
2020-09-02 06:10:30	45.59975	-122.684178	192	26	166947931

6. List all bread crumb readings for a specific portion of Highway 14 (bounding box: [(45.610794, -122.576979), (45.606989, -122.569501)]) 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?

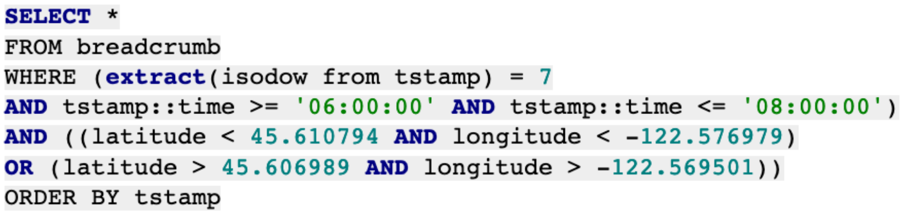
COMPARISON:

Count For Monday between 4pm to 6pm - 5159
Count For Sunday between 6am to 8am - 307

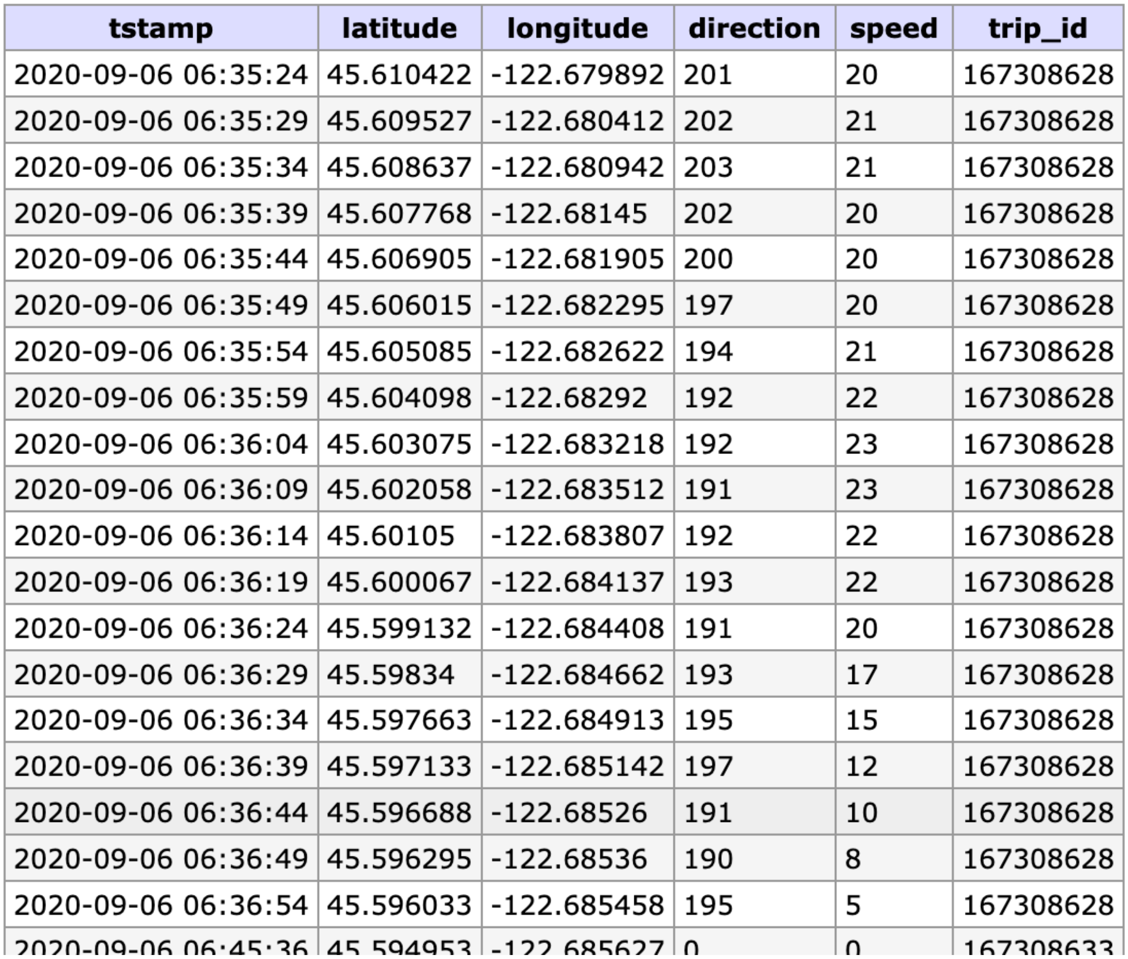
```
SELECT *
FROM breadcrumb
WHERE (extract(isodow from tstamp) = 1
AND tstamp::time >= '16:00:00' AND tstamp::time <= '18:00:00')
AND ((latitude < 45.610794 AND longitude < -122.576979)
OR (latitude > 45.606989 AND longitude > -122.569501))
ORDER BY tstamp
```



tstamp	latitude	longitude	direction	speed	trip_id
2020-09-07 16:00:02	45.61769	-122.524923	99	13	167315615
2020-09-07 16:00:07	45.617625	-122.524297	98	9	167315615
2020-09-07 16:00:12	45.617603	-122.524108	99	2	167315615
2020-09-07 16:00:27	45.61758	-122.523883	98	1	167315615
2020-09-07 16:00:32	45.617515	-122.523285	99	9	167315615
2020-09-07 16:00:37	45.61743	-122.522472	98	12	167315615
2020-09-07 16:00:42	45.617328	-122.521503	99	15	167315615
2020-09-07 16:00:47	45.61721	-122.5204	99	17	167315615
2020-09-07 16:00:52	45.617087	-122.51926	99	17	167315615
2020-09-07 16:00:55	45.66989	-122.552902	89	0	167314210
2020-09-07 16:00:57	45.616967	-122.51813	99	17	167315615
2020-09-07 16:00:59	45.595518	-122.685488	0	0	167326452
2020-09-07 16:01:00	45.669993	-122.552553	67	6	167314210
2020-09-07 16:01:02	45.616847	-122.517017	99	17	167315615
2020-09-07 16:01:04	45.59514	-122.685563	188	8	167326452
2020-09-07 16:01:05	45.67029	-122.55249	8	6	167314210
2020-09-07 16:01:07	45.616725	-122.515918	99	17	167315615
2020-09-07 16:01:09	45.594737	-122.685747	198	9	167326452
2020-09-07 16:01:12	45.616598	-122.51485	100	16	167315615
2020-09-07 16:01:14	45.594527	-122.686227	238	9	167326452

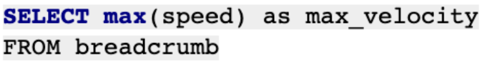


```
SELECT *
FROM breadcrumb
WHERE (extract(isodow from tstamp) = 7
AND tstamp::time >= '06:00:00' AND tstamp::time <= '08:00:00')
AND ((latitude < 45.610794 AND longitude < -122.576979)
OR (latitude > 45.606989 AND longitude > -122.569501))
ORDER BY tstamp
```

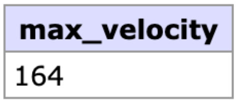


tstamp	latitude	longitude	direction	speed	trip_id
2020-09-06 06:35:24	45.610422	-122.679892	201	20	167308628
2020-09-06 06:35:29	45.609527	-122.680412	202	21	167308628
2020-09-06 06:35:34	45.608637	-122.680942	203	21	167308628
2020-09-06 06:35:39	45.607768	-122.68145	202	20	167308628
2020-09-06 06:35:44	45.606905	-122.681905	200	20	167308628
2020-09-06 06:35:49	45.606015	-122.682295	197	20	167308628
2020-09-06 06:35:54	45.605085	-122.682622	194	21	167308628
2020-09-06 06:35:59	45.604098	-122.68292	192	22	167308628
2020-09-06 06:36:04	45.603075	-122.683218	192	23	167308628
2020-09-06 06:36:09	45.602058	-122.683512	191	23	167308628
2020-09-06 06:36:14	45.60105	-122.683807	192	22	167308628
2020-09-06 06:36:19	45.600067	-122.684137	193	22	167308628
2020-09-06 06:36:24	45.599132	-122.684408	191	20	167308628
2020-09-06 06:36:29	45.59834	-122.684662	193	17	167308628
2020-09-06 06:36:34	45.597663	-122.684913	195	15	167308628
2020-09-06 06:36:39	45.597133	-122.685142	197	12	167308628
2020-09-06 06:36:44	45.596688	-122.68526	191	10	167308628
2020-09-06 06:36:49	45.596295	-122.68536	190	8	167308628
2020-09-06 06:36:54	45.596033	-122.685458	195	5	167308628
2020-09-06 06:45:36	45.594953	-122.685627	0	0	167308633

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



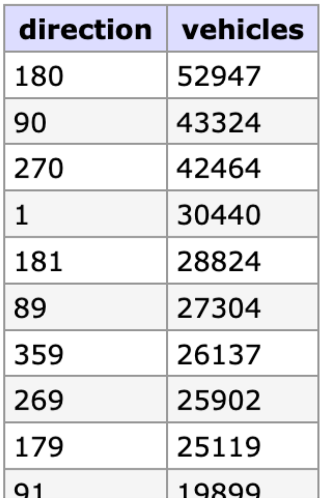
```
SELECT max(speed) as max_velocity
FROM breadcrumb
```



max_velocity
164

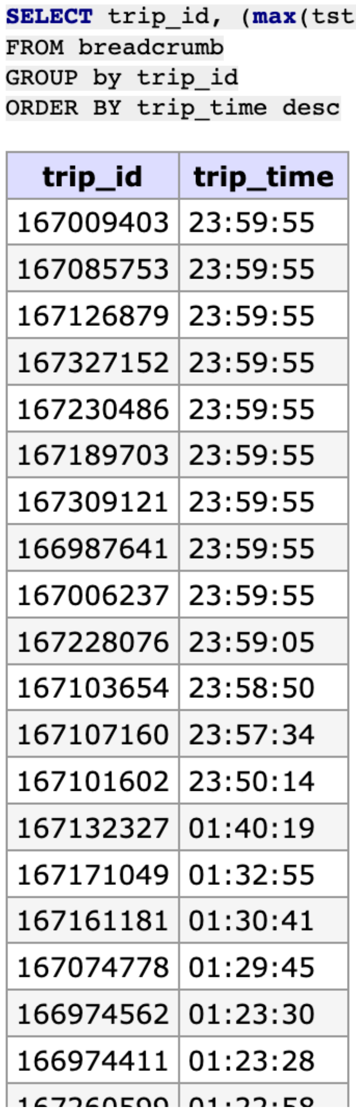
8. List all possible directions and give a count of the number of vehicles that faced precisely that direction during at least one trip. Sort the list by most frequent direction to least frequent.

```
SELECT direction, count(direction) as vehicles
FROM breadcrumb
WHERE direction > 0
GROUP by direction
ORDER BY vehicles desc
```



direction	vehicles
180	52947
90	43324
270	42464
1	30440
181	28824
89	27304
359	26137
269	25902
179	25119
91	19899

9. ~~Which is the longest (in terms of distance) trip of all trips in the data?~~ (ignore question 9)
10. Which is the longest (in terms of time) trip of all trips in the data?

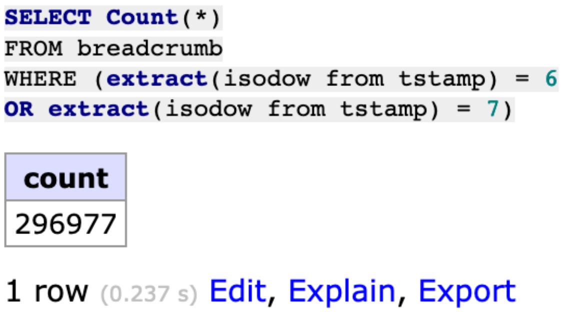


```
SELECT trip_id, (max(tstamp) - min(tstamp)) as trip_time
FROM breadcrumb
GROUP by trip_id
ORDER BY trip_time desc
```

trip_id	trip_time
167009403	23:59:55
167085753	23:59:55
167126879	23:59:55
167327152	23:59:55
167230486	23:59:55
167189703	23:59:55
167309121	23:59:55
166987641	23:59:55
167006237	23:59:55
167228076	23:59:05
167103654	23:58:50
167107160	23:57:34
167101602	23:50:14
167132327	01:40:19
167171049	01:32:55
167161181	01:30:41
167074778	01:29:45
166974562	01:23:30
166974411	01:23:28
167260599	01:22:58

11. ~~Which vehicle is the fastest? "Fastest" in this case should be measured in miles per hour averaged from the beginning of a trip to the end of the trip. That is, the total distance of the trip divided by the total time of the trip. This then should be averaged over all trips that each vehicle serviced.~~ (ignore question 11)
12. Devise three new, interesting questions about the C-Tran bus system that can be answered by your bread crumb 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).

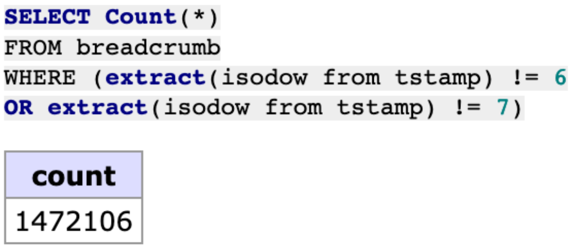
12.a. Compare total number of trips on weekdays to total number of trips on weekend



```
SELECT Count(*)
FROM breadcrumb
WHERE (extract(isodow from tstamp) = 6
OR extract(isodow from tstamp) = 7)
```

count
296977

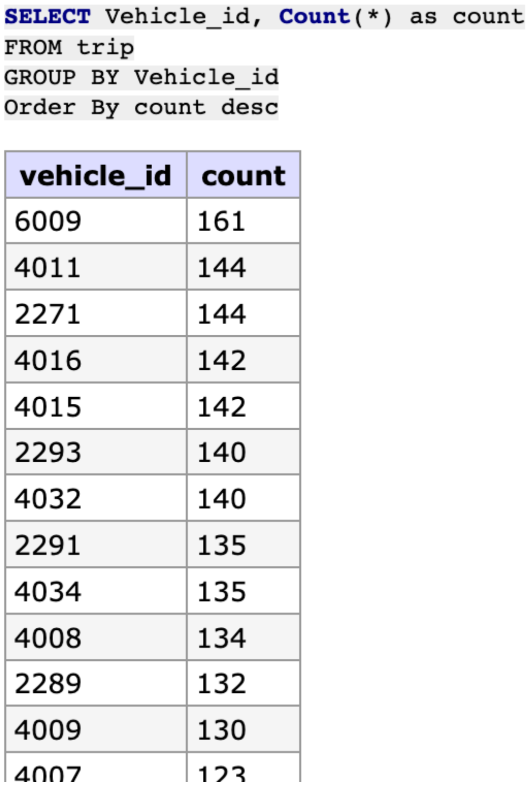
1 row (0.237 s) Edit, Explain, Export



```
SELECT Count(*)
FROM breadcrumb
WHERE (extract(isodow from tstamp) != 6
OR extract(isodow from tstamp) != 7)
```

count
1472106

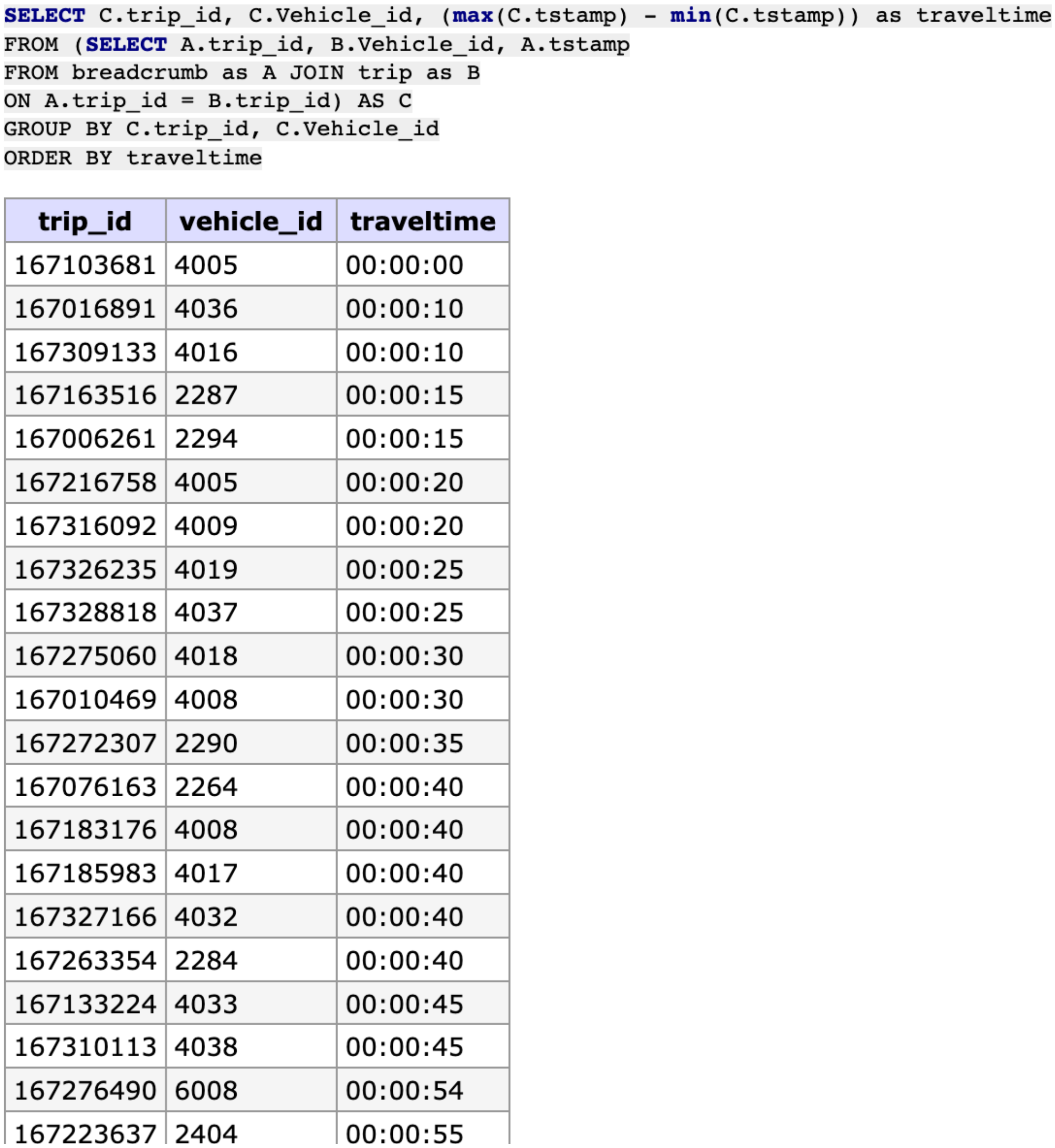
12.b. Which vehicle has done most number of trips



```
SELECT Vehicle_id, Count(*) as count
FROM trip
GROUP BY Vehicle_id
Order By count desc
```

vehicle_id	count
6009	161
4011	144
2271	144
4016	142
4015	142
2293	140
4032	140
2291	135
4034	135
4008	134
2289	132
4009	130
4007	123

12.c. Which vehicle has made the shortest trip by time.



```
SELECT C.trip_id, C.Vehicle_id, (max(C.tstamp) - min(C.tstamp)) as traveltime
FROM (SELECT A.trip_id, B.Vehicle_id, A.tstamp
FROM breadcrumb as A JOIN trip as B
ON A.trip_id = B.trip_id) AS C
GROUP BY C.trip_id, C.Vehicle_id
ORDER BY traveltime
```

trip_id	vehicle_id	traveltime
167103681	4005	00:00:00
167016891	4036	00:00:10
167309133	4016	00:00:10
167163516	2287	00:00:15
167006261	2294	00:00:15
167216758	4005	00:00:20
167316092	4009	00:00:20
167326235	4019	00:00:25
167328818	4037	00:00:25
167275060	4018	00:00:30
167010469	4008	00:00:30
167272307	2290	00:00:35
167076163	2264	00:00:40
167183176	4008	00:00:40
167185983	4017	00:00:40
167327166	4032	00:00:40
167263354	2284	00:00:40
167133224	4033	00:00:45
167310113	4038	00:00:45
167276490	6008	00:00:54
167223637	2404	00:00:55

Your Code

Provide a reference to the repository where you store your python code. If you are keeping it private then share it with Bruce (bruce.irvin@gmail.com), David and Aman (github references TBD).

Repo Link - https://github.com/Yokeshthirumoorthi/DataEngineeringWithKakfa