

BT2102 - LAB 3

Kaustubh Jagtap – A0168820B



Queries:

4a) What were the number of trips per customer type?

Query:

```
SELECT subscription_type AS subs, COUNT(*)
FROM trip WHERE duration <= 86400
GROUP BY subs;</pre>
```

ANS:

4b) What were the shortest, average and longest trip duration?

Query:

```
MIN(duration) AS min, MAX(duration) AS max, AVG(duration) AS average FROM trip WHERE duration <= 86400;
```

ANS:

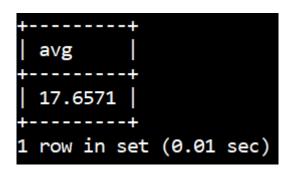
```
+----+
| min | max | average |
+-----+
| 60 | 86381 | 987.3186 |
+----+
1 row in set (0.41 sec)
```

4c) On average, how many docks are at each station?

Query:

SELECT AVG(dock_count) as avg from station;

Ans:

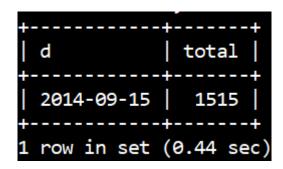


4d) Which was the busiest day of all, in terms of trips taken?

Query:

```
SELECT subq1.d, COUNT(subq1.d) AS total FROM (SELECT cast(start_date as date) as d FROM trip WHERE duration<=86400) AS subq1 GROUP BY subq1.d ORDER BY total DESC LIMIT 1;
```

ANS:

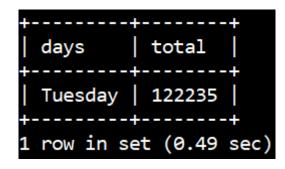


4e) Which was the busiest day of the week?

Query:

```
SELECT subq1.days, count(subq1.days) as total FROM (select DAYNAME(CAST(start_date as date)) AS days FROM trip WHERE duration<=86400) AS subq1 GROUP BY days ORDER BY total DESC LIMIT 1;
```

ANS:



4f) For this question, I wanted to find some meaningful information that could be monetized. One potential source of additional revenue could be from peak surcharge during the busiest hours. Hence, the question I arrived at was: What is the breakdown of number of rides by hour of the day? Addressing this question also allows us to smoothen our operational capabilities, by planning for the optimal number of bikes to cater for peak demans.

Query:

```
SELECT subq1.hour, COUNTsubq1.hour) as total,
TRUNCATE(COUNT(subq1.hour)/(SELECT COUNT(*) from trip) * 100, 2)
AS percentage
FROM (SELECT HOUR(start_date) as hour from trip)
AS subq1
GROUP BY subq1.hour
ORDER BY subq1.hour < 6, subq1.hour;</pre>
```

		
hour	total	percentage
+		++
6	14312	2.13
7	43939	6.55
8	85864	12.81
9	62897	9.38
10	30106	4.49
11	29141	4.34
12	34384	5.13
13	31740	4.73
14	27156	4.05
15	33223	4.95
16	59099	8.82
17	82705	12.34
18	57652	8.60
19	29188	4.35
20	16527	2.46
21	11277	1.68
22	7434	1.10
23	4450	0.66
0	2171	0.32
1	1189	0.17
2	692	0.10
3	342	0.05
4	1022	0.15
5	3449	0.51
++		

Data Processing - trip

I found a list of all California zip codes online, on the following website: http://federalgovernmentzipcodes.us/

To filter out the wrong zip codes, I matched all the entries with this list and altered the trips data. This was output into a new csv file, from which the data was loaded.

```
""Returns a processed version of trip data. Dirty zips are changed to 0"""
import csv
def read csv(csvfilename):
    Reads a csv file and returns a list of list
    containing rows in the csv file and its entries.
    rows = []
    with open(csvfilename) as csvfile:
        file_reader = csv.reader(csvfile)
        for row in file reader:
            rows.append(row)
    return rows
def store zips(): # returns a set of all california zip codes
    res = set()
    zip_codes = read_csv("../free-zipcode-database-Primary.csv")
    zip codes = zip codes[1:]
    for row in zip_codes:
        if(row[3] == "CA"):
            res.add(row[0])
    return res
```

```
def process_zips(zip_data): ## returns csv with modified zip codes
    data = read_csv("trip_raw.csv")
    temp = data[1:]
    for i in range(len(temp)):
        zc = temp[i][-1]
        if len(zc) < 5:
            temp[i][-1] = 0
        else:
            zc = zc[:5]
            if zc not in zip data:
               temp[i][-1] = 0
            else:
                temp[i][-1] = zc
   with open("trip_processed.csv", 'w', newline='') as f:
       writer = csv.writer(f, delimiter=',')
        writer.writerow(data[0])
        writer.writerows(temp)
    return f
california_zips = store_zips() # set of all california zip codes
trip_processed = process_zips(california_zips)
```

Data processing - weather

For the weather data, it was necessary to convert trace amount data (represented by 'T') to a numeric form; hence I converted those entries into 0.00. Missing columns were also replaced with 0.00.

```
Looks at weather data; wherever precipitation_inches is stored as 'T', stores 0 instead"
import csv
import codecs
def read_csv(csvfilename):
     Reads a csv file and returns a list of list
     containing rows in the csv file and its entries.
     rows = []
     with open(csvfilename) as csvfile:
           file_reader = csv.reader(csvfile)
           for row in file reader:
                rows.append(row)
     return rows
def process_weather(): ## returns csv with modified precipitation inches
   data = read_csv("weather_raw.csv")
     temp = data[1:]
     for i in range(len(temp)):
    if(temp[i][-5]=='T'):
        temp[i][-5] = 0
          for j in range(1, len(temp[i])):
    if(j!=21 and temp[i][j] == ""):
        temp[i][j] = 0
     with open("weather_processed.csv", 'w', newline='') as f:
    writer = csv.writer(f, delimiter=',')
    writer.writerow(data[0])
          writer.writerows(temp)
weather_processed = process_weather()
```

Schema

Load Data

```
LOAD DATA LOCAL INFILE 'C:/Users/ksjag/Documents/NUS/Sem 2/BT2102/Labs/Lab 3/Lab 3/station.csv'
INTO TABLE station
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
LINES TERMINATED BY '\n'
LINES TERMINATED BY '\n'
LINES (stationID, name, latitude, longitude, dock_count, city, @tmp_inst)
SET

installation_date = STR_TO_DATE(@tmp_inst, '%m/%d/%V');

-LOAD DATA LOCAL INFILE 'C:/Users/ksjag/Documents/NUS/Sem 2/BT2102/Labs/Lab 3/Lab 3/trip_processed.csv'
INTO TABLE trip
FIELDS TERMINATED BY '\n'
LINES TERMINATED BY '\n'
LINES TERMINATED BY '\n'
LINES (tripID, duration, @tmp_start, start_station_name, start_station_id, @tmp_end, end_station_name, end_station_id, bikeID,
subscription_type, zip_code)
SET

start_date = STR_TO_DATE(@tmp_start, '%m/%d/%Y %H: %i');

-Load status|
LOAD DATA LOCAL INFILE 'C:/Users/ksjag/Documents/NUS/Sem 2/BT2102/Labs/Lab 3/Lab 3/status.csv'
INTO TABLE status
FIELDS TERMINATED BY '\n'
LINES TERMINATED BY '\n'
LI
```