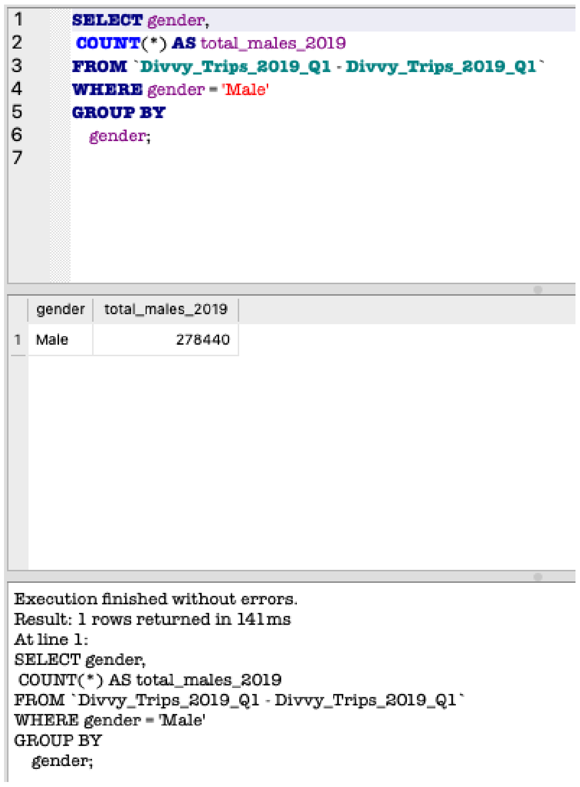
Google data Analytics Capstone Project
Part 2 (a): *2019 Q1 and 2020 Q1(Using SQL and Tableau)*

For this analyzation, SQLite was used for cleaning and transforming two data sets: Divvy_Trips_2019_Q1 and Divvy_Trips_2020_Q1. The programming started from simple SELECT, FROM and WHERE steps in SQL to calculate total number of males, females, subscribers, and customers in 2019 data set. This progressed towards more defined codes for modifying columns in 2020 data set to match those in 2019 data set, and finally combining the data sets for visualizations in Tableau. of trip duration (length of a single trip) and day of week has been made for both 2019 and 2020.

Tableau Viz: Trips Trend Analysis

Using SQl to Clean data

1. **Calculating total number of males from 2019 data set.**

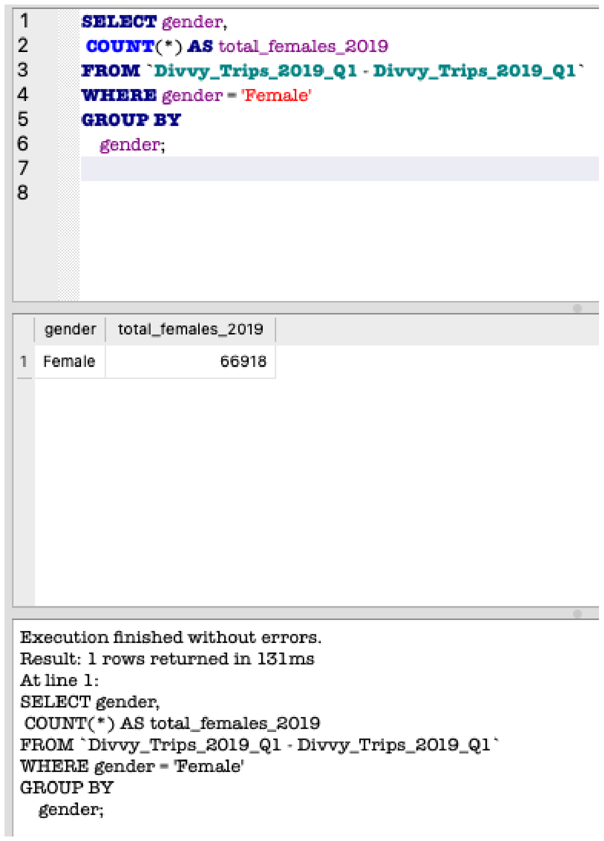


```
SELECT gender,
 COUNT(*) AS total_males_2019
FROM `Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1`
WHERE gender = 'Male'
GROUP BY
  gender;
```

```
   gender  total_males_2019
1  Male              278440
```

```
Execution finished without errors.
Result: 1 rows returned in 141ms
At line 1:
SELECT gender,
 COUNT(*) AS total_males_2019
FROM `Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1`
WHERE gender = 'Male'
GROUP BY
  gender;
```

2. **Calculating total number of females from 2019 data set.**

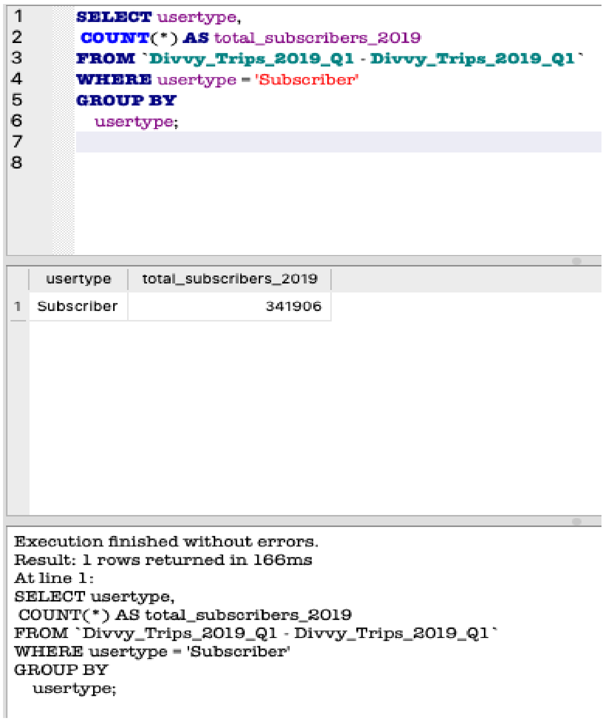


```
SELECT gender,
 COUNT(*) AS total_females_2019
FROM `Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1`
WHERE gender = 'Female'
GROUP BY
   gender;
```

	gender	total_females_2019
1	Female	66918

```
Execution finished without errors.
Result: 1 rows returned in 131ms
At line 1:
SELECT gender,
 COUNT(*) AS total_females_2019
FROM `Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1`
WHERE gender = 'Female'
GROUP BY
   gender;
```

3. **Calculating total number of Subscribers from 2019 data set.**

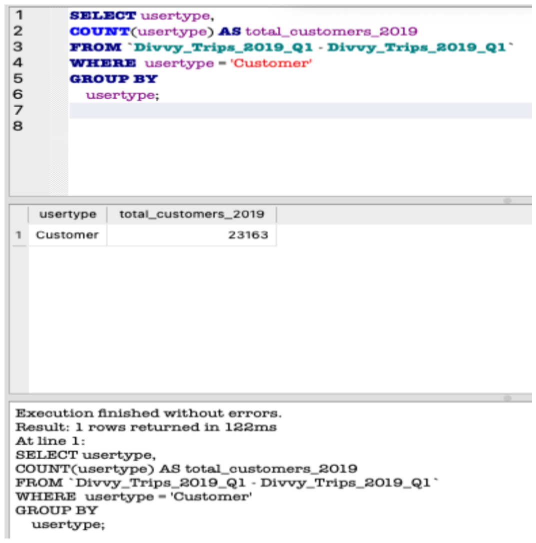


```
SELECT usertype,
 COUNT(*) AS total_subscribers_2019
FROM `Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1`
WHERE usertype = 'Subscriber'
GROUP BY
   usertype;
```

	usertype	total_subscribers_2019
1	Subscriber	341906

```
Execution finished without errors.
Result: 1 rows returned in 166ms
At line 1:
SELECT usertype,
 COUNT(*) AS total_subscribers_2019
FROM `Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1`
WHERE usertype = 'Subscriber'
GROUP BY
   usertype;
```

4. Calculating total number of Customers from 2019 data set.



```
SELECT usertype,
COUNT(usertype) AS total_customers_2019
FROM `Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1`
WHERE usertype = 'Customer'
GROUP BY
   usertype;
```

	usertype	total_customers_2019
1	Customer	23163

```
Execution finished without errors.
Result: 1 rows returned in 122ms
At line 1:
SELECT usertype,
COUNT(usertype) AS total_customers_2019
FROM `Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1`
WHERE usertype = 'Customer'
GROUP BY
   usertype;
```

5. Calculating number of trips from each station, from 2019 data-set and creating a new table named stations_2019.

```
CREATE TABLE IF NOT EXISTS stations_2019 (
   from_station_name TEXT,
   number_of_trips INTEGER
);

INSERT INTO stations_2019 (from_station_name, number_of_trips)
SELECT
   from_station_name,
   COUNT(*) AS number_of_trips
FROM `Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1`
GROUP BY from_station_name;
```

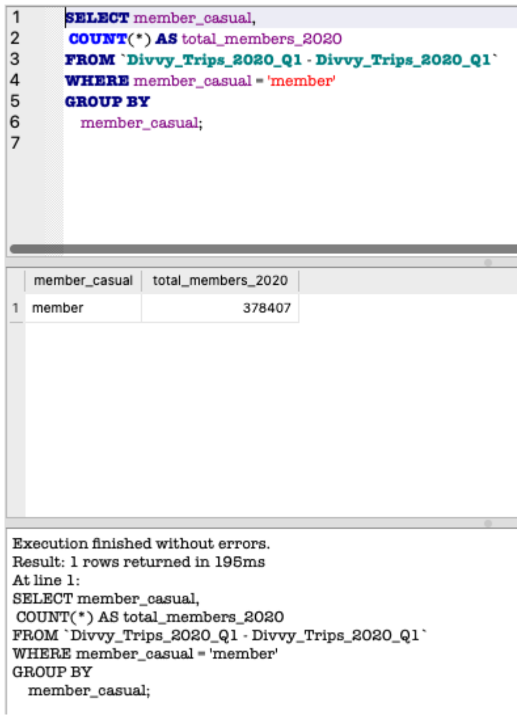
6. Creating a summary table of 2019

```
CREATE TABLE IF NOT EXISTS summary_2019 (
   from_station_name TEXT,
   station_frequency INTEGER,
         total_males INTEGER,
         total_females INTEGER,
         male_subscribers INTEGER,
         female_subscribers INTEGER,
         male_customers INTEGER,
         female_customers INTEGER,
         total_subscribers INTEGER,
         total_customers INTEGER
);

INSERT INTO summary_2019 (from_station_name, station_frequency, total_males, total_females,
male_subscribers, female_subscribers, male_customers, female_customers, total_subscribers, total_customers)
SELECT
```

```
    from_station_name,
    COUNT(*) AS station_frequency,
    SUM(CASE WHEN gender = 'Male' THEN 1 ELSE 0 END) AS total_males,
    SUM(CASE WHEN gender = 'Female' THEN 1 ELSE 0 END) AS total_females,
    SUM(CASE WHEN usertype = 'Subscriber' AND gender = 'Male' THEN 1 ELSE 0 END) AS
male_subscribers,
    SUM(CASE WHEN usertype = 'Subscriber' AND gender = 'Female' THEN 1 ELSE 0 END) AS
female_subscribers,
    SUM(CASE WHEN usertype = 'Customer' AND gender = 'Male' THEN 1 ELSE 0 END) AS male_customers,
    SUM(CASE WHEN usertype = 'Customer' AND gender = 'Female' THEN 1 ELSE 0 END) AS
female_customers,
    SUM(CASE WHEN usertype = 'Subscriber' THEN 1 ELSE 0 END) AS total_subscribers,
    SUM(CASE WHEN usertype = 'Customer' THEN 1 ELSE 0 END) AS total_customers
FROM `Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1`
GROUP BY from_station_name
ORDER BY station_frequency DESC;
```

7. Calculating total number of Members from 2020 data set.

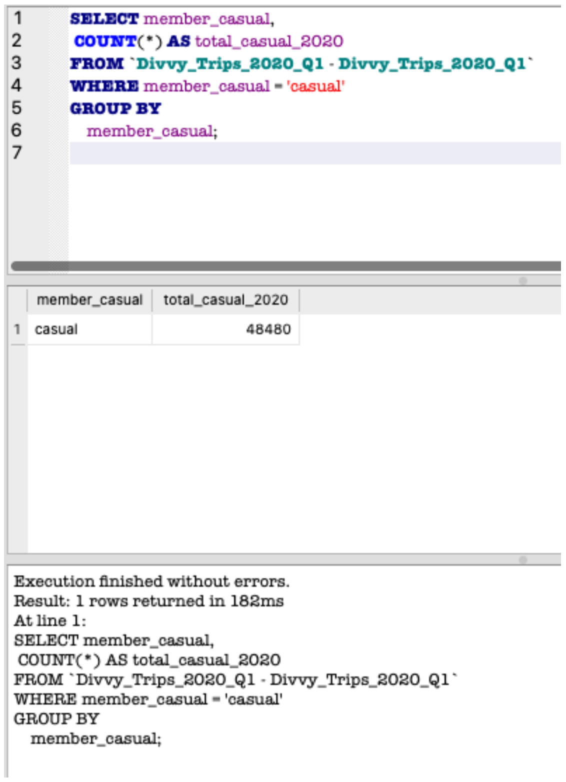


```
SELECT member_casual,
 COUNT(*) AS total_members_2020
FROM `Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1`
WHERE member_casual = 'member'
GROUP BY
   member_casual;
```

	member_casual	total_members_2020
1	member	378407

```
Execution finished without errors.
Result: 1 rows returned in 195ms
At line 1:
SELECT member_casual,
 COUNT(*) AS total_members_2020
FROM `Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1`
WHERE member_casual = 'member'
GROUP BY
   member_casual;
```

8. **Calculating total number of Casual riders from 2020 data set.**



```
SELECT member_casual,
 COUNT(*) AS total_casual_2020
FROM `Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1`
WHERE member_casual = 'casual'
GROUP BY
   member_casual;
```

	member_casual	total_casual_2020
1	casual	48480

```
Execution finished without errors.
Result: 1 rows returned in 182ms
At line 1:
SELECT member_casual,
 COUNT(*) AS total_casual_2020
FROM `Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1`
WHERE member_casual = 'casual'
GROUP BY
   member_casual;
```

9. **Calculating number of trips from each station, from 2020 data-set and creating a new table named stations_2020.**

```
CREATE TABLE IF NOT EXISTS stations_2020 (
   start_station_nameTEXT,
   number_of_trips_2020 INTEGER
);

INSERT INTO stations_2020 (start_station_name, number_of_trips_2020)
SELECT
          start_station_name,
   COUNT(*) AS number_of_trips_2020
FROM `Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1`
GROUP BY start_station_name;
```

10. **Joining two tables: stations_2019 and stations_2020 and creating a new table combined_data**

```
CREATE TABLE IF NOT EXISTS combined_data AS
SELECT
   start_time,
   end_time,
   from_station_name,
   to_station_name AS end_station_name,
   usertype
FROM "Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1"
```

```
UNION ALL
SELECT
    start_time,
    end_time,
    from_station_name,
    to_station_name,
    usertype
FROM "Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1";
```

11. Separating time from start_time in Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1

```
-- Add a new column to the existing table
ALTER TABLE "Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1"
ADD COLUMN s_time TEXT;

-- Update the new column with the desired values
UPDATE "Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1"
SET s_time = SUBSTR(start_time, 12, 8);
```

12. Separating time from end_time in Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1

```
-- Add a new column to the existing table
ALTER TABLE "Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1"
ADD COLUMN e_time TEXT;

-- Update the new column with the desired values
UPDATE "Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1"
SET e_time = SUBSTR(end_time, 12, 8);
```

13. Create a new column “length_of_trip” by subtracting s_time from e_time

```
--Add a new column to store the length of the trip
ALTER TABLE "Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1"
ADD COLUMN length_of_trip TEXT;

-- Update the new column with the calculated difference
UPDATE "Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1"
SET length_of_trip = TIME(
    julianday(e_time) * 86400 - julianday(s_time) * 86400,
    'unixepoch'
);
```

14. Separating day from start_time

```
-- Add a new column to store the day of the week
ALTER TABLE "Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1"
ADD COLUMN day_of_week TEXT;

-- Update the new column with the day of the week
UPDATE "Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1"
SET day_of_week = CASE
                    WHEN CAST(strftime('%w', start_time) AS INTEGER) = 0 THEN 'Sunday'
```

```
            WHEN CAST(strftime('%w', start_time) AS INTEGER) = 1 THEN 'Monday'
            WHEN CAST(strftime('%w', start_time) AS INTEGER) = 2 THEN 'Tuesday'
            WHEN CAST(strftime('%w', start_time) AS INTEGER) = 3 THEN 'Wednesday'
            WHEN CAST(strftime('%w', start_time) AS INTEGER) = 4 THEN 'Thursday'
            WHEN CAST(strftime('%w', start_time) AS INTEGER) = 5 THEN 'Friday'
            WHEN CAST(strftime('%w', start_time) AS INTEGER) = 6 THEN 'Saturday'
          END;
```

15. Separating day from start_time in Divvy_Trips_2020_Q1 – Divvy_Trips_2020_Q1

```
-- Add a new column to store the day of the week
ALTER TABLE "Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1"
ADD COLUMN day_of_week TEXT;

-- Update the new column with the day of the week
UPDATE "Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1"
SET day_of_week = CASE
        WHEN CAST(strftime('%w', start_time) AS INTEGER) = 0 THEN 'Sunday'
        WHEN CAST(strftime('%w', start_time) AS INTEGER) = 1 THEN 'Monday'
        WHEN CAST(strftime('%w', start_time) AS INTEGER) = 2 THEN 'Tuesday'
        WHEN CAST(strftime('%w', start_time) AS INTEGER) = 3 THEN 'Wednesday'
        WHEN CAST(strftime('%w', start_time) AS INTEGER) = 4 THEN 'Thursday'
        WHEN CAST(strftime('%w', start_time) AS INTEGER) = 5 THEN 'Friday'
        WHEN CAST(strftime('%w', start_time) AS INTEGER) = 6 THEN 'Saturday'
      END;
```

16. Separating s_time from start_time in Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1

```
-- Add a new column to the existing table
ALTER TABLE "Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1"
ADD COLUMN s_time TEXT;

-- Update the new column with the desired values
UPDATE "Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1"
SET s_time = SUBSTR(start_time, 12, 8);
```

17. Separating e_time from end_time in Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1

```
-- Add a new column to the existing table
ALTER TABLE "Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1"
ADD COLUMN e_time TEXT;

-- Update the new column with the desired values
UPDATE "Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1"
SET e_time = SUBSTR(end_time, 12, 8);
```

18. Create a new column “length_of_trip” by subtracting s_time from e_time

```
Add a new column to store the length of the trip
ALTER TABLE "Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1"
ADD COLUMN length_of_trip TEXT;
```

```
-- Update the new column with the calculated difference
UPDATE "Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1"
SET length_of_trip = TIME(
   julianday(e_time) * 86400 - julianday(s_time) * 86400,
   'unixepoch'
);
```

19. **Combining data from both tables:**

```
-- Create a new table "summary_data"
CREATE TABLE summary_data AS
SELECT from_station_name AS station_name, NULL AS end_station_name, length_of_trip, day_of_week
FROM "Divvy_Trips_2019_Q1 - Divvy_Trips_2019_Q1"
UNION ALL
SELECT from_station_name AS station_name, to_station_name AS end_station_name, length_of_trip,
day_of_week
FROM "Divvy_Trips_2020_Q1 - Divvy_Trips_2020_Q1";
```