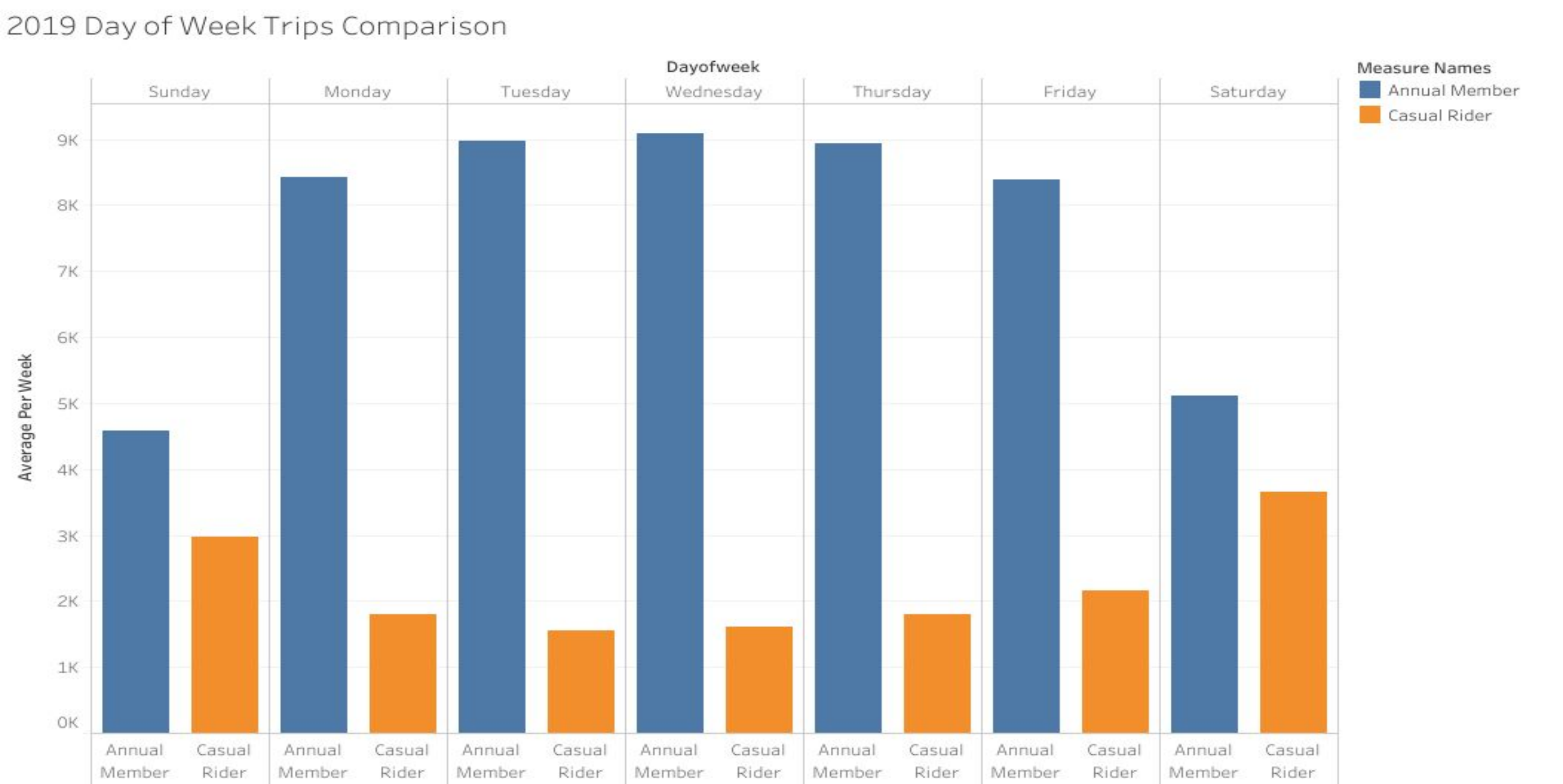
Cyclistic

Overview

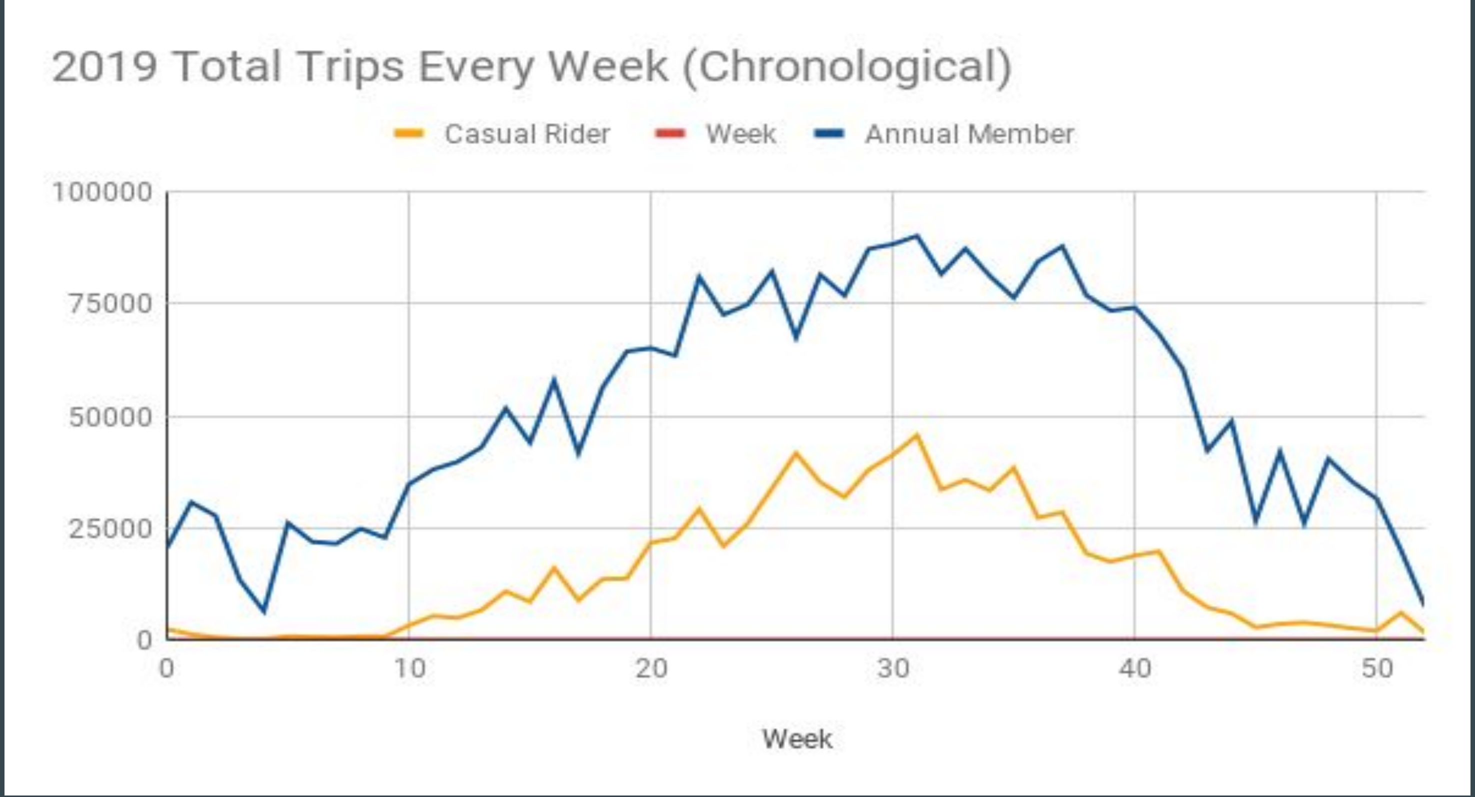
Created in 2016 and located in Chicago, Illinois, Cyclistic is a successful bike-share company ready to reach new heights. Cyclistic's business model involves casual riders (customers who purchase single-ride or full-day passes) and annual members, and its financial analysts have concluded that annual members drive better profitability. Therefore, the marketing team is now seeking ways to convert casual riders into annual members.

Key Question

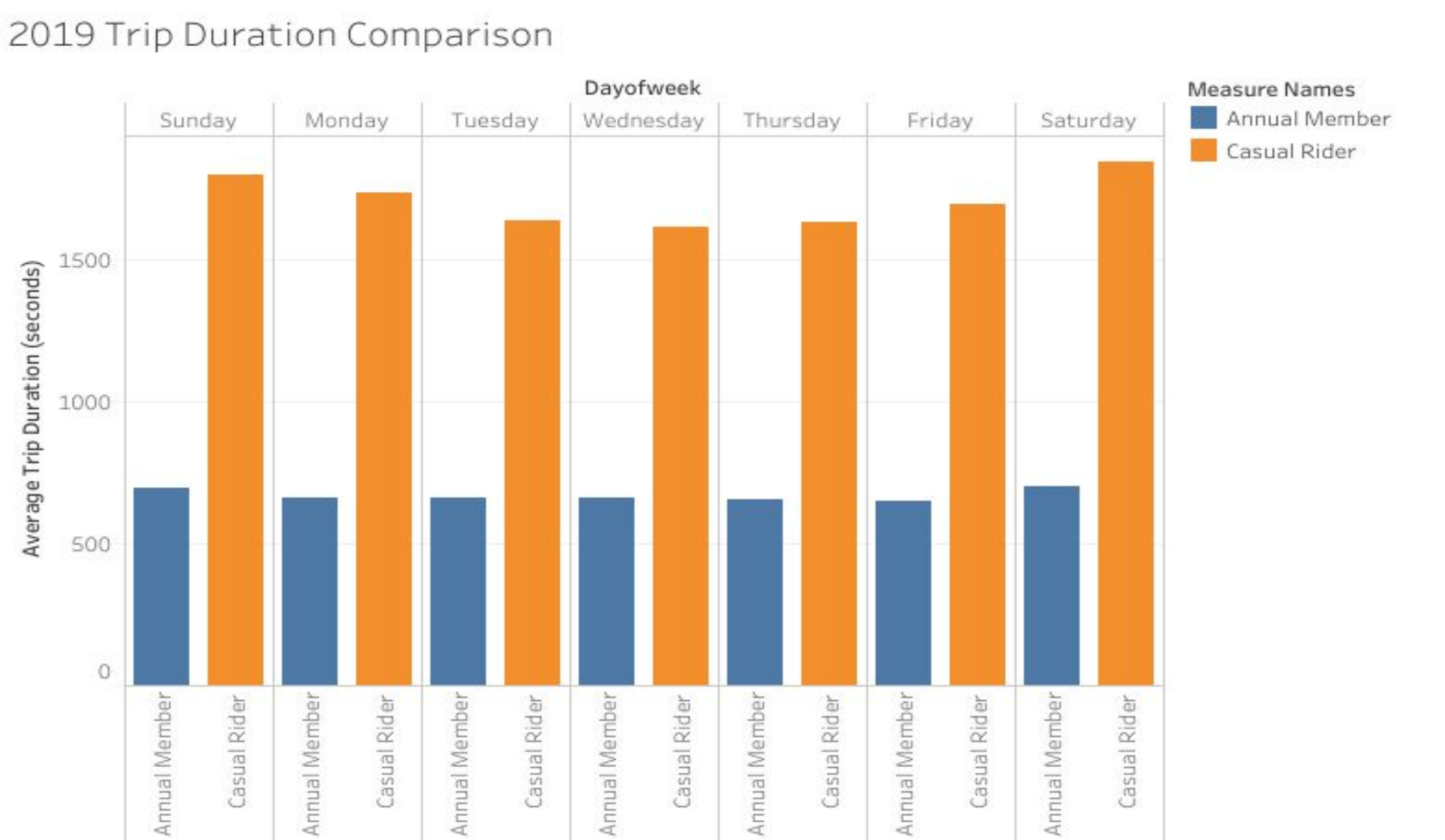
How do annual members and casual riders use Cyclistic bikes differently?



We begin our analysis by looking at which days of the week experienced the most number of trips on average. It appears that annual members prefer weekdays whereas casual riders prefer weekends.

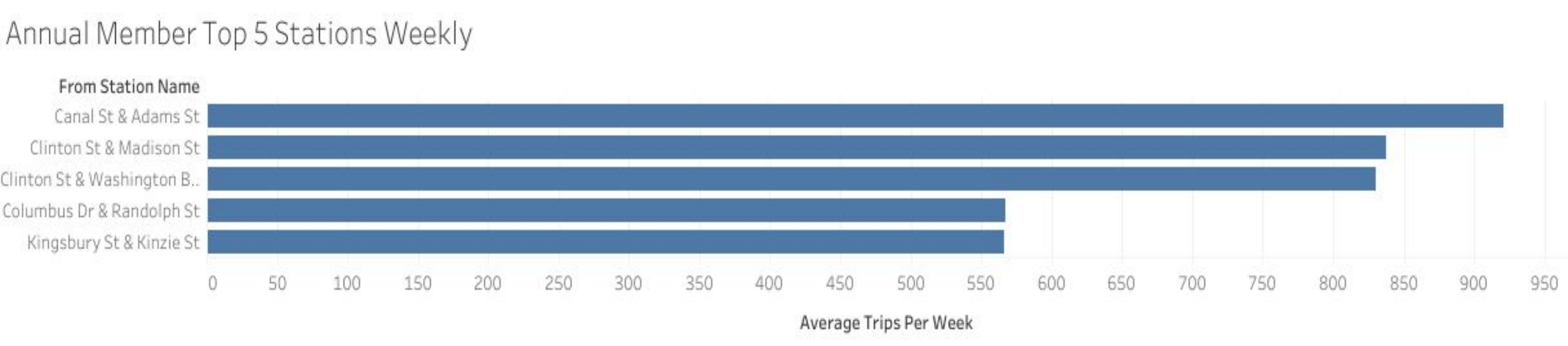


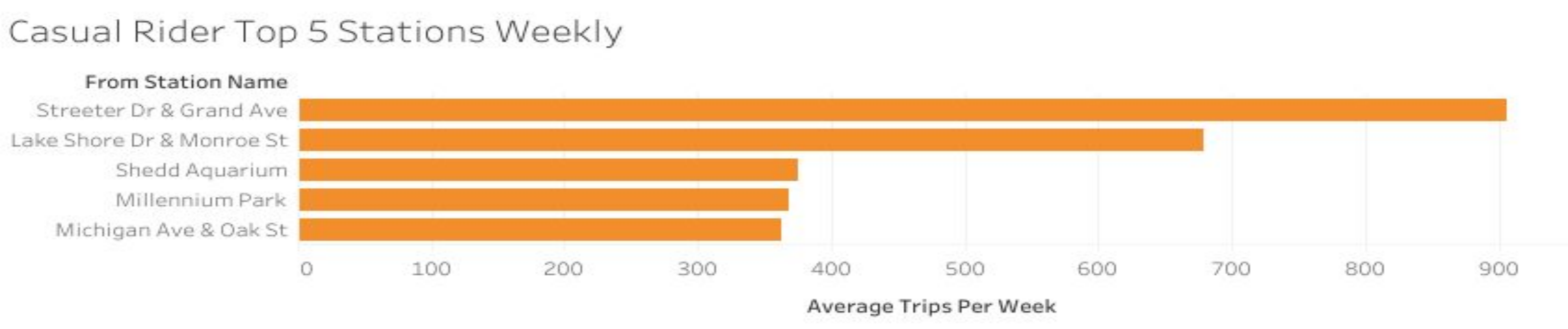
Through this time series chart, we see that annual members took a larger quantity of trips every week of the year compared to casual riders. The trends of both lines mirror closely which indicates both annual members and casual riders prefer to ride in the summer (week 25 - week 35).



2019 Trip Duration Comparison

Above we show the average time length of trips that occurred on each day of the week. Annual members tend to keep a relatively consistent, flat average whereas casual riders tend to ride for longer time lengths on the weekend and overall ride longer than annual members.





From the two charts, we see the stations with the highest average weekly trips. None of the top five stations are shared by casual riders and annual members.

Recap of Key Question: How do annual members and casual riders use Cyclistic bikes differently?

- Annual members ride more frequently per week.

- Annual members prefer weekdays whereas casual riders prefer weekends.

- Casual riders ride for longer durations and show more fluctuation in duration data.

- Casual riders and annual members most popular stations are all in different locations.

My three recommendations to drive conversion of casual riders to annual members:

1. *Advertise the benefits of annual membership at the most popular casual rider stations, with an emphasis on summertime and weekends.*

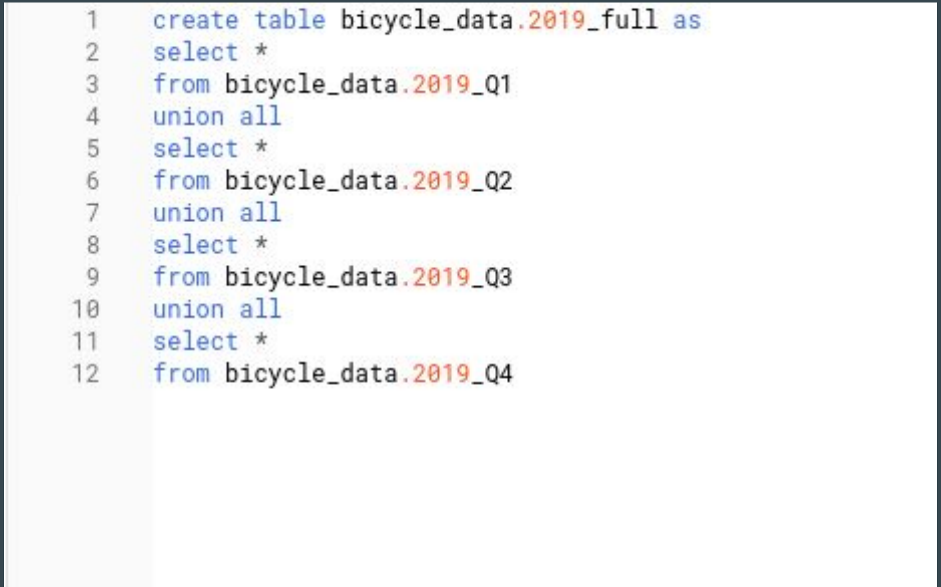
2. *Offer an annual membership free trial to casual riders after his or her account shows a high number of trips.*

3. *Set a ride time limit for casual riders after which casual riders are required to watch an advertisement on his or her phone before closing out that trip session.*

Documentation

Data Sources Used:

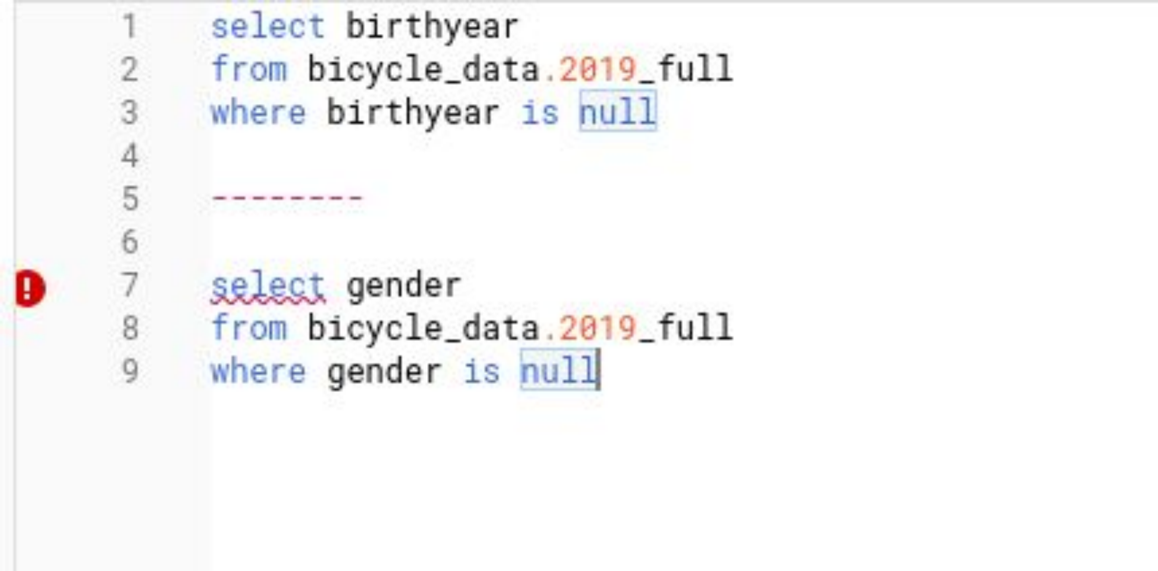
- https://divvy-tripdata.s3.amazonaws.com/Divvy_Trips_2019_Q1.zip
- https://divvy-tripdata.s3.amazonaws.com/Divvy_Trips_2019_Q2.zip
- https://divvy-tripdata.s3.amazonaws.com/Divvy_Trips_2019_Q3.zip
- https://divvy-tripdata.s3.amazonaws.com/Divvy_Trips_2019_Q4.zip



```sql
create table bicycle_data.2019_full as
select *
from bicycle_data.2019_Q1
union all
select *
from bicycle_data.2019_Q2
union all
select *
from bicycle_data.2019_Q3
union all
select *
from bicycle_data.2019_Q4
```

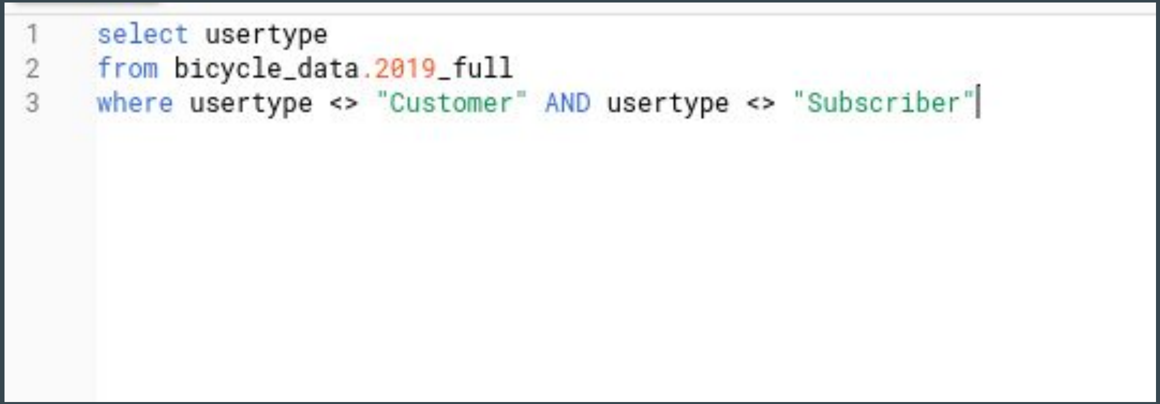
- I uploaded the prior listed data sources to Google BigQuery because that amount of data overwhelmed my excel, to hone my SQL skills, and to hopefully standout as I believe most completing this case study will follow the R notes given from the Google certificate program.

- I combined the data in BigQuery by creating a new table through the code shown above.



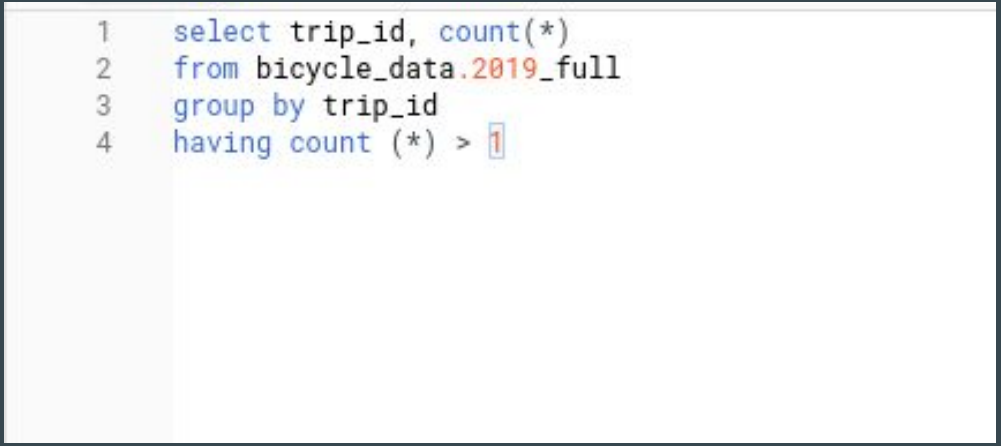
```
select birthyear
from bicycle_data.2019_full
where birthyear is null

--------

select gender
from bicycle_data.2019_full
where gender is null
```

To start the cleaning process, I searched columns for null values in which the gender and birthyear columns each returned close to 500k null values out of 3M rows. I deemed both columns contained too much unreliable data to be used.



```
select usertype
from bicycle_data.2019_full
where usertype <> "Customer" AND usertype <> "Subscriber"
```

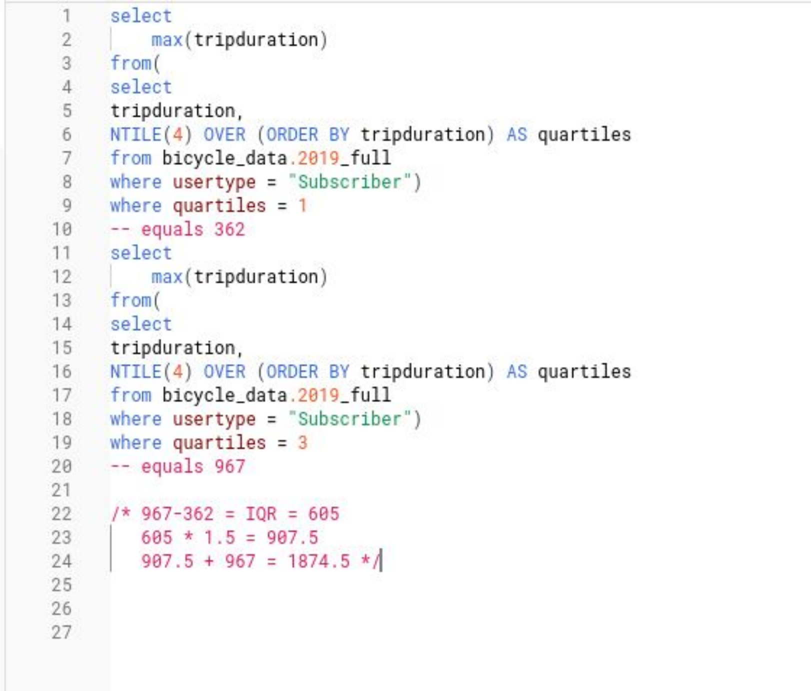
Next, I checked for any different values in the Customer and Subscriber columns.



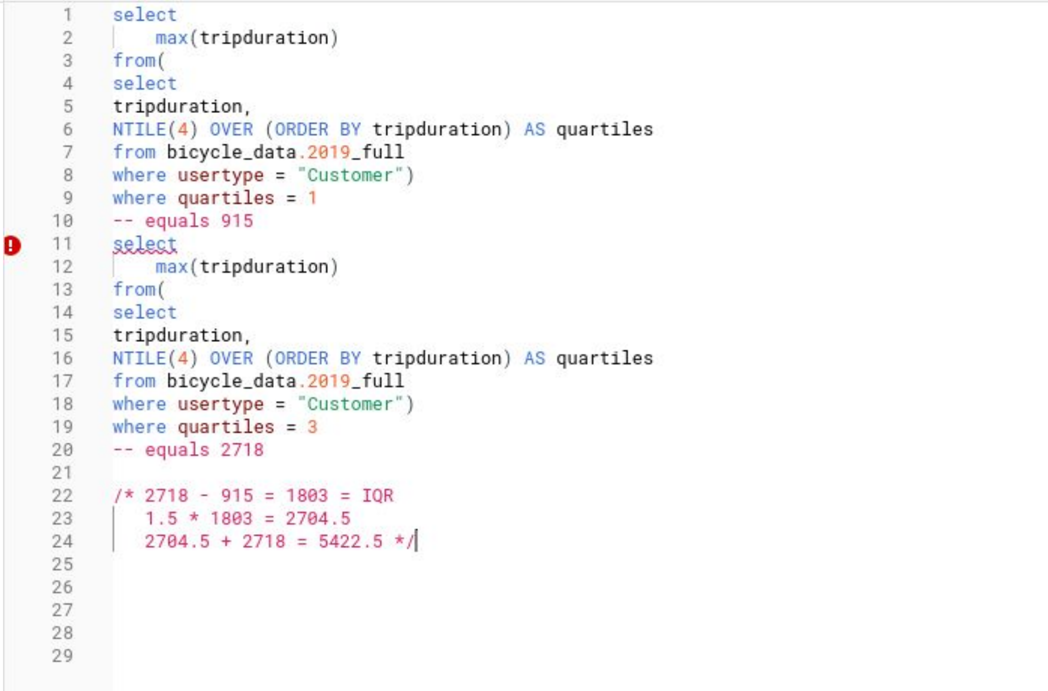
```
select trip_id, count(*)
from bicycle_data.2019_full
group by trip_id
having count (*) > 1
```

Here I checked for duplicate trip ids.

Next, I will check for outliers in the data.



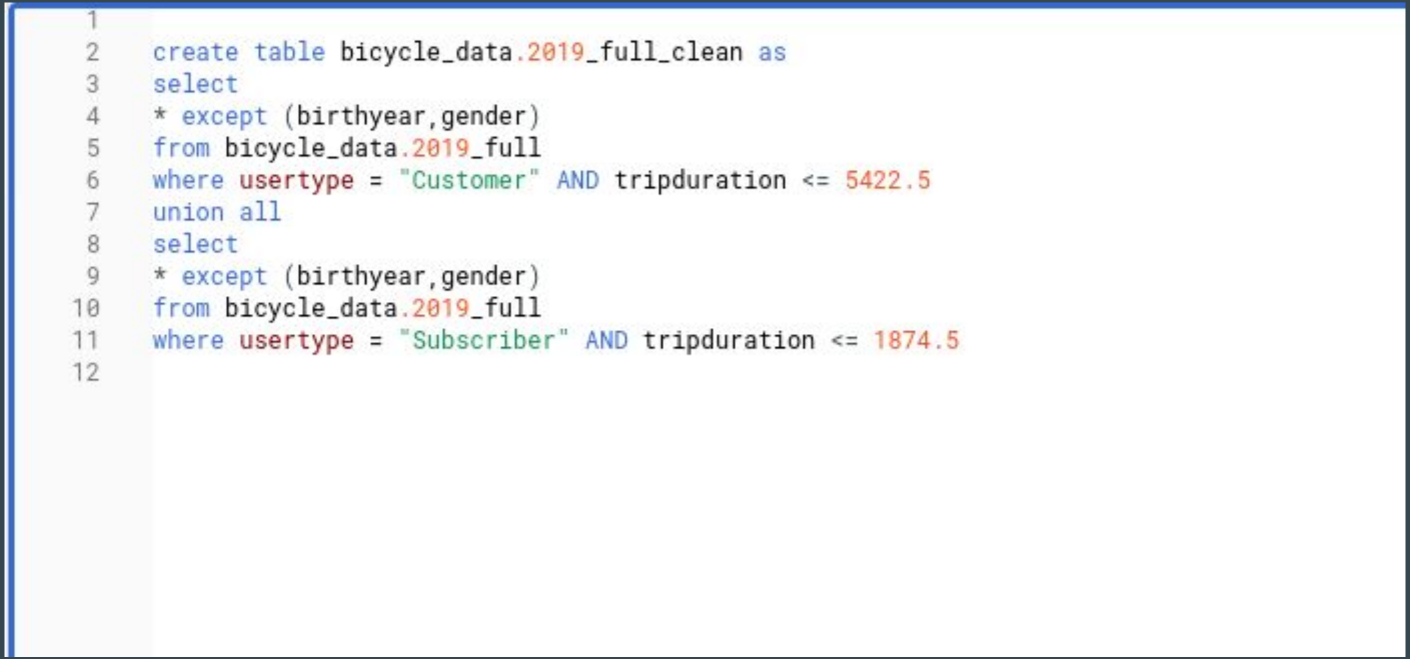
```
select
    max(tripduration)
from(
select
tripduration,
NTILE(4) OVER (ORDER BY tripduration) AS quartiles
from bicycle_data.2019_full
where usertype = "Subscriber")
where quartiles = 1
-- equals 362
select
    max(tripduration)
from(
select
tripduration,
NTILE(4) OVER (ORDER BY tripduration) AS quartiles
from bicycle_data.2019_full
where usertype = "Subscriber")
where quartiles = 3
-- equals 967

/* 967-362 = IQR = 605
   605 * 1.5 = 907.5
   907.5 + 967 = 1874.5 */
```



```
select
    max(tripduration)
from(
select
tripduration,
NTILE(4) OVER (ORDER BY tripduration) AS quartiles
from bicycle_data.2019_full
where usertype = "Customer")
where quartiles = 1
-- equals 915
select
    max(tripduration)
from(
select
tripduration,
NTILE(4) OVER (ORDER BY tripduration) AS quartiles
from bicycle_data.2019_full
where usertype = "Customer")
where quartiles = 3
-- equals 2718

/* 2718 - 915 = 1803 = IQR
   1.5 * 1803 = 2704.5
   2704.5 + 2718 = 5422.5 */
```

Using the 1.5 * IQR rule I established upper bounds at 1874.5 seconds for “Subscribers” aka Annual Members and 5422.5 seconds for “Customers” aka Casual Riders.

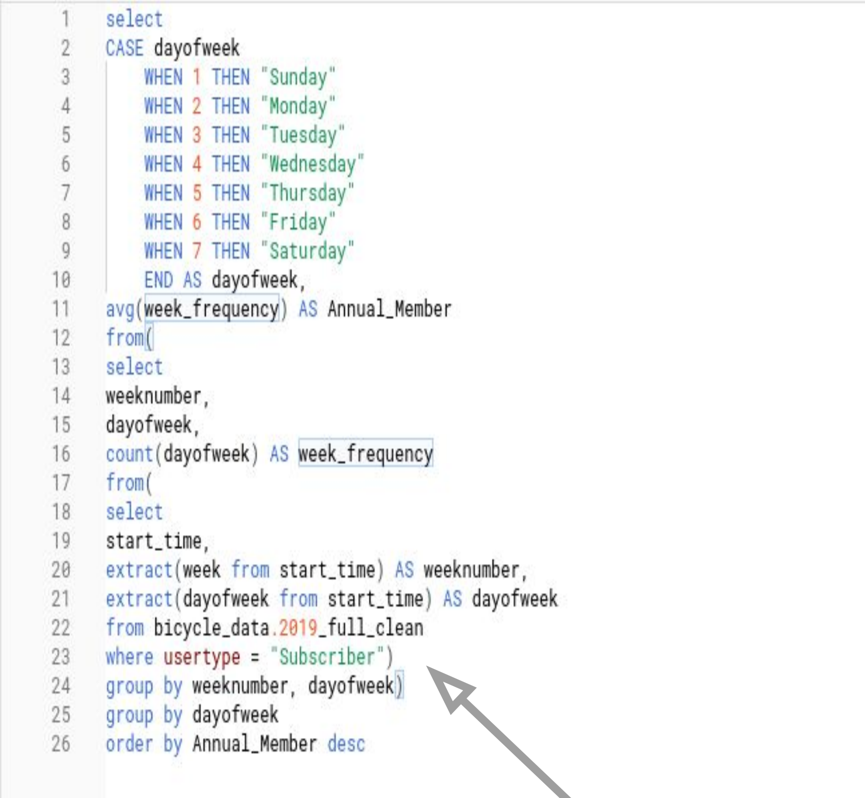
Above these bounds are suspected outliers which I removed from the data. The lower bounds for both “Subscribers” and “Customers” would have been well below zero and no data was found below a value of zero.



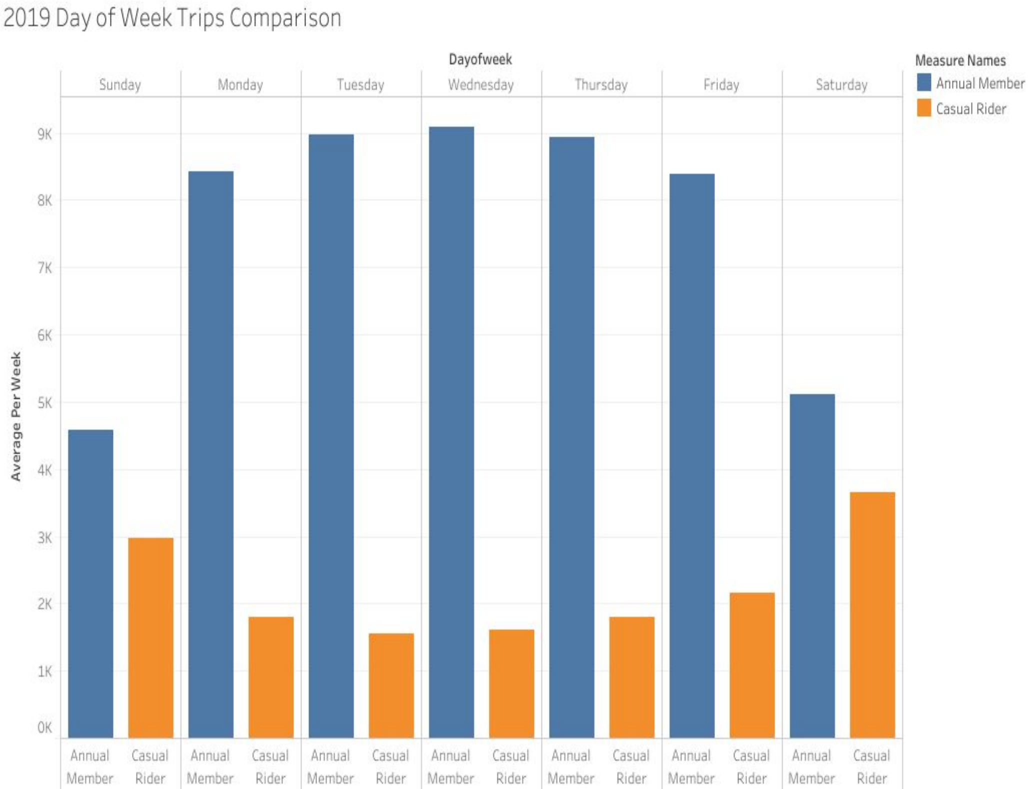
```sql

create table bicycle_data.2019_full_clean as
select
* except (birthyear,gender)
from bicycle_data.2019_full
where usertype = "Customer" AND tripduration <= 5422.5
union all
select
* except (birthyear,gender)
from bicycle_data.2019_full
where usertype = "Subscriber" AND tripduration <= 1874.5

```

To complete the cleaning process, I combined my cleaning techniques into a new table to be used for analysis.

Code Used for Charts

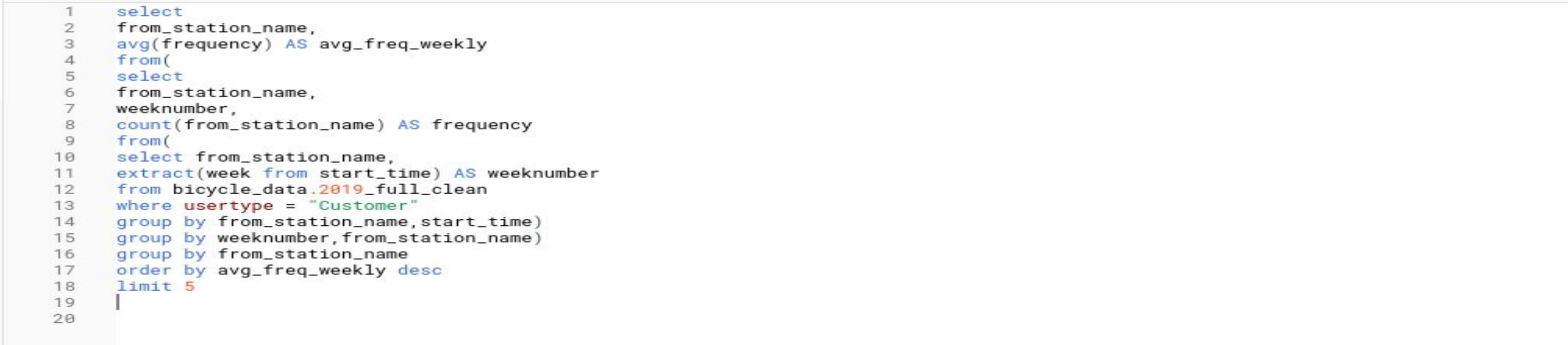


```
select
CASE dayofweek
    WHEN 1 THEN "Sunday"
    WHEN 2 THEN "Monday"
    WHEN 3 THEN "Tuesday"
    WHEN 4 THEN "Wednesday"
    WHEN 5 THEN "Thursday"
    WHEN 6 THEN "Friday"
    WHEN 7 THEN "Saturday"
    END AS dayofweek,
avg(week_frequency) AS Annual_Member
from(
select
weeknumber,
dayofweek,
count(dayofweek) AS week_frequency
from(
select
start_time,
extract(week from start_time) AS weeknumber,
extract(dayofweek from start_time) AS dayofweek
from bicycle_data.2019_full_clean
where usertype = "Subscriber")
group by weeknumber, dayofweek)
group by dayofweek
order by Annual_Member desc
```



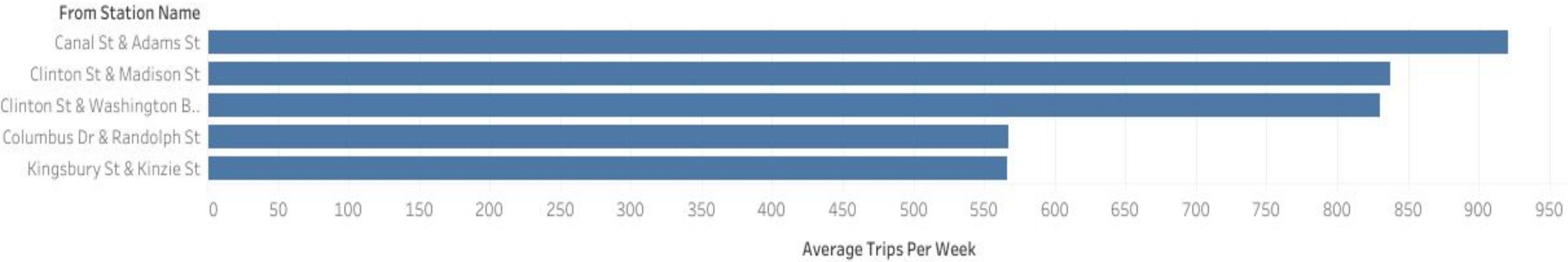
The code used to analyze "Customer" data was the same as "Subscriber". To avoid additional clutter I only showed one code chunk in all my documentations.

"Subscriber" was manually changed to Annual Member and "Customer" to Casual Rider in all my charts.

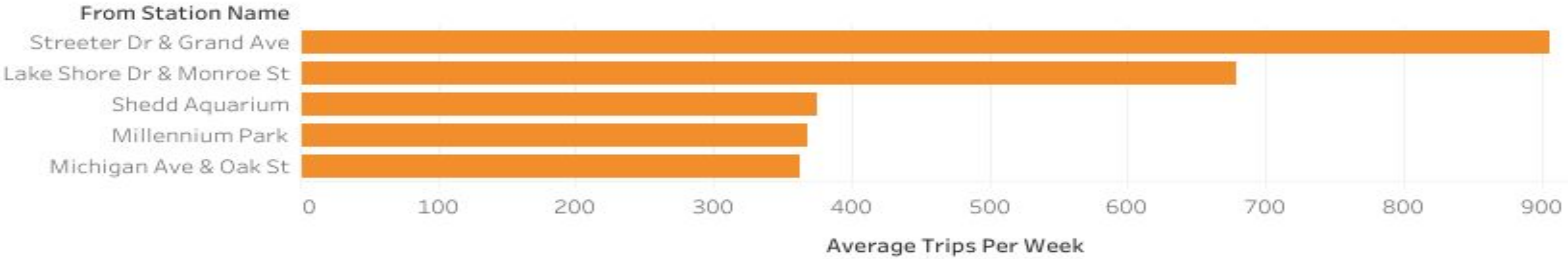


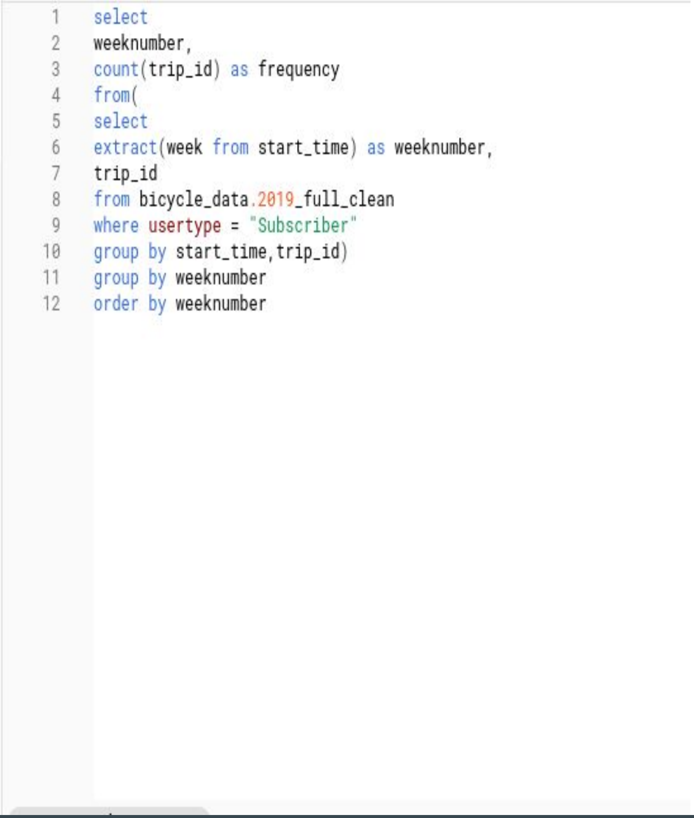
```
select
from_station_name,
avg(frequency) AS avg_freq_weekly
from(
select
from_station_name,
weeknumber,
count(from_station_name) AS frequency
from(
select from_station_name,
extract(week from start_time) AS weeknumber
from bicycle_data.2019_full_clean
where usertype = "Customer"
group by from_station_name,start_time)
group by weeknumber,from_station_name)
group by from_station_name
order by avg_freq_weekly desc
limit 5
```

Annual Member Top 5 Stations Weekly

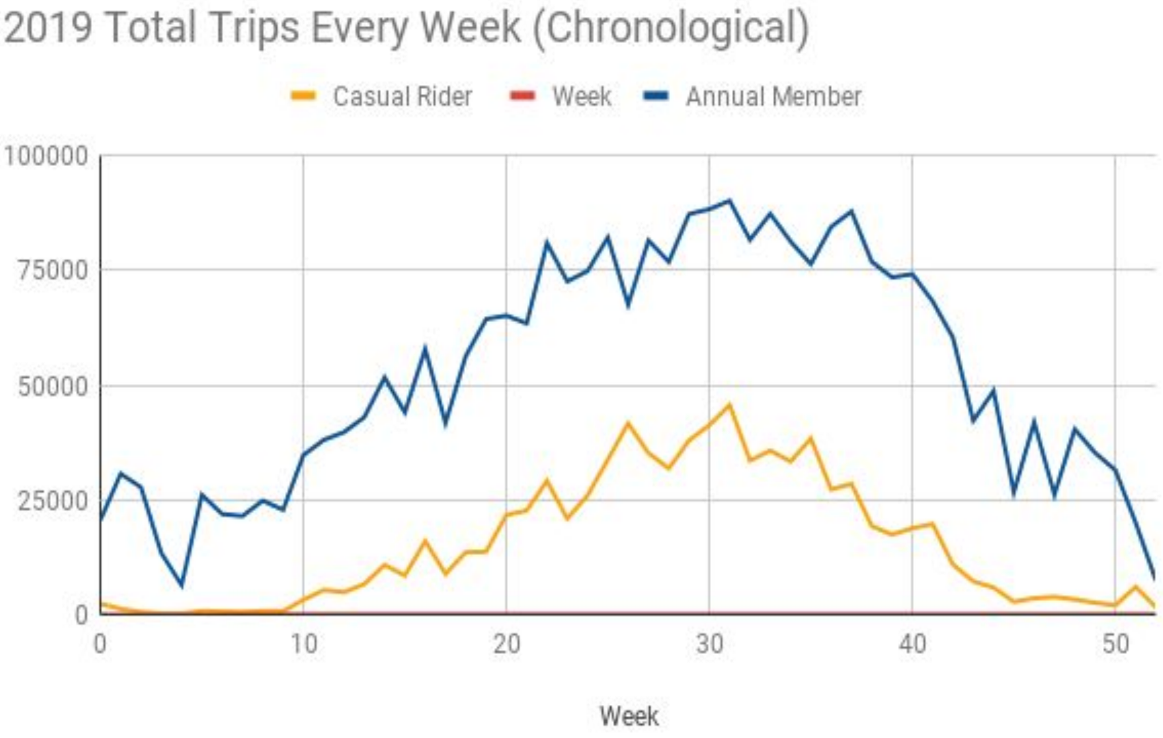


Casual Rider Top 5 Stations Weekly

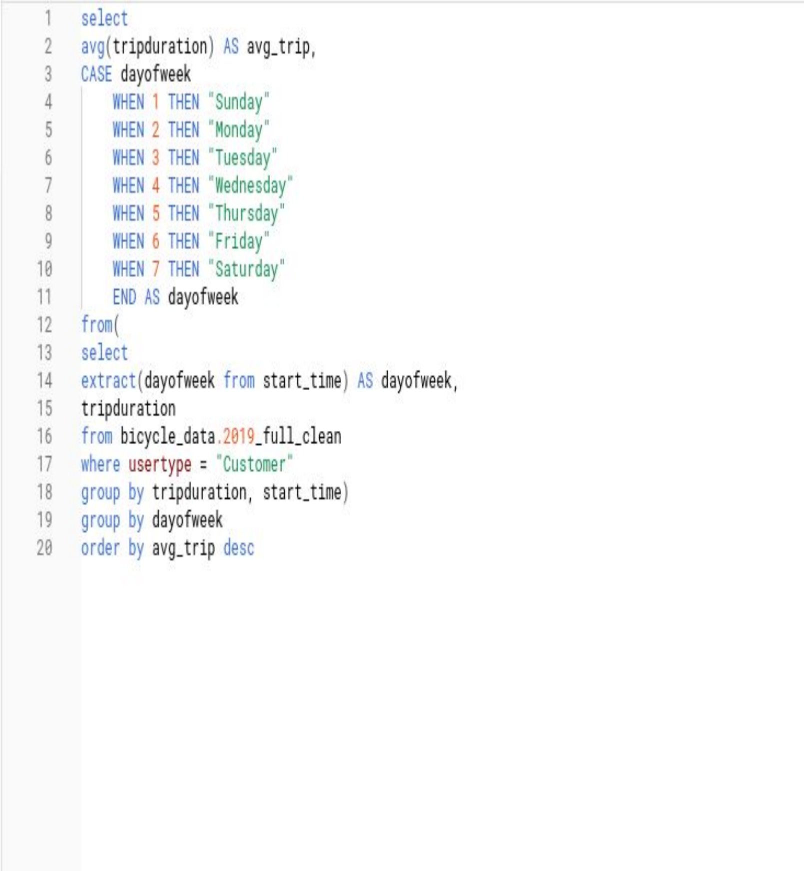




```
select
weeknumber,
count(trip_id) as frequency
from(
select
extract(week from start_time) as weeknumber,
trip_id
from bicycle_data.2019_full_clean
where usertype = "Subscriber"
group by start_time,trip_id)
group by weeknumber
order by weeknumber
```

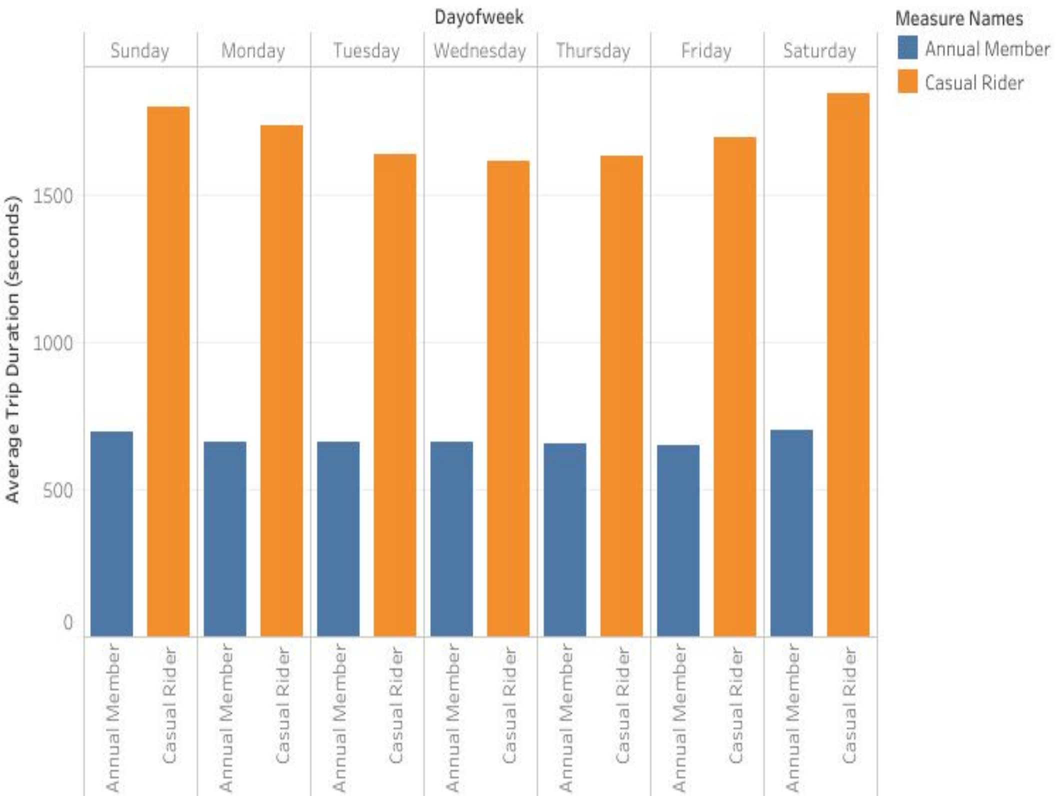


2019 Total Trips Every Week (Chronological)



```sql
select
avg(tripduration) AS avg_trip,
CASE dayofweek
    WHEN 1 THEN "Sunday"
    WHEN 2 THEN "Monday"
    WHEN 3 THEN "Tuesday"
    WHEN 4 THEN "Wednesday"
    WHEN 5 THEN "Thursday"
    WHEN 6 THEN "Friday"
    WHEN 7 THEN "Saturday"
    END AS dayofweek
from(
select
extract(dayofweek from start_time) AS dayofweek,
tripduration
from bicycle_data.2019_full_clean
where usertype = "Customer"
group by tripduration, start_time)
group by dayofweek
order by avg_trip desc
```

2019 Trip Duration Comparison



Thanks for viewing!