Cyclist Capstone

Matthew S.

2025-04-11

Introduction to the capstone

In this report, I will be going over how the fictional company, Cyclistic, can maximize the number of annual subscribers. In order to do this I will be gathering data on the differences between subscribers and non-subscribers and show how they differ from one another. For this practice study, I will be in charge of the question "How do annual members and casual riders use Cyclistic bikes differently?" This document will walk you through my steps in answering this question.

*All data used is free public data made available through the license found at https://divvybikes.com/data-license-agreement

Installing packages

The following packages were used in this project and need to be installed and loaded in order for the following code chunks to work.

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
## here() starts at /cloud/project
##
##
## Attaching package: 'janitor'
##
```

```
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

Reading the csv files

We are going to first import the csv files found at https://divvy-tripdata.s3.amazonaws.com/index.html (this data is from a fictional company and should not be taken as a factual data) and give them their own variable to work with throughout this report. We can also verify they were imported correctly with the glimspe() function. You could also use the View() function if you want a full view of the file.

```
trips2019 <- read_csv("Divvy_Trips_2019_Q1.csv")
```

```
## Rows: 365069 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): from_station_name, to_station_name, usertype, gender
## dbl  (5): trip_id, bikeid, from_station_id, to_station_id, birthyear
## num  (1): tripduration
## dttm (2): start_time, end_time
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
trips2020 <- read_csv("Divvy_Trips_2020_Q1.csv")
```

```
## Rows: 426887 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(trips2019)
```

```
## Rows: 365,069
## Columns: 12
## $ trip_id           <dbl> 21742443, 21742444, 21742445, 21742446, 21742447, 21~
## $ start_time        <dttm> 2019-01-01 00:04:37, 2019-01-01 00:08:13, 2019-01-0~
## $ end_time          <dttm> 2019-01-01 00:11:07, 2019-01-01 00:15:34, 2019-01-0~
## $ bikeid            <dbl> 2167, 4386, 1524, 252, 1170, 2437, 2708, 2796, 6205,~
## $ tripduration      <dbl> 390, 441, 829, 1783, 364, 216, 177, 100, 1727, 336, ~
## $ from_station_id   <dbl> 199, 44, 15, 123, 173, 98, 98, 211, 150, 268, 299, 2~
## $ from_station_name <chr> "Wabash Ave & Grand Ave", "State St & Randolph St", ~
## $ to_station_id     <dbl> 84, 624, 644, 176, 35, 49, 49, 142, 148, 141, 295, 4~
## $ to_station_name   <chr> "Milwaukee Ave & Grand Ave", "Dearborn St & Van Bure~
## $ usertype          <chr> "Subscriber", "Subscriber", "Subscriber", "Subscribe~
## $ gender            <chr> "Male", "Female", "Female", "Male", "Male", "Female"~
## $ birthyear         <dbl> 1989, 1990, 1994, 1993, 1994, 1983, 1984, 1990, 1995~
```

```
glimpse(trips2020)
```

```
## Rows: 426,887
## Columns: 13
## $ ride_id            <chr> "EACB19130B0CDA4A", "8FED874C809DC021", "789F3C21E4~
## $ rideable_type      <chr> "docked_bike", "docked_bike", "docked_bike", "docke~
## $ started_at         <dttm> 2020-01-21 20:06:59, 2020-01-30 14:22:39, 2020-01--~
## $ ended_at           <dttm> 2020-01-21 20:14:30, 2020-01-30 14:26:22, 2020-01--~
## $ start_station_name <chr> "Western Ave & Leland Ave", "Clark St & Montrose Av~
## $ start_station_id   <dbl> 239, 234, 296, 51, 66, 212, 96, 96, 212, 38, 117, 1~
## $ end_station_name   <chr> "Clark St & Leland Ave", "Southport Ave & Irving Pa~
## $ end_station_id     <dbl> 326, 318, 117, 24, 212, 96, 212, 212, 96, 100, 632,~
## $ start_lat          <dbl> 41.9665, 41.9616, 41.9401, 41.8846, 41.8856, 41.889~
## $ start_lng          <dbl> -87.6884, -87.6660, -87.6455, -87.6319, -87.6418, --~
## $ end_lat            <dbl> 41.9671, 41.9542, 41.9402, 41.8918, 41.8899, 41.884~
## $ end_lng            <dbl> -87.6674, -87.6644, -87.6530, -87.6206, -87.6343, --~
## $ member_casual      <chr> "member", "member", "member", "member", "member", "~
```

What data to focus on and what to ignore

If we took a look at the tables, we notice that there are some columns that do not align with our current task that we are trying to answer. One column that is redundant is in the trips2020 table with the rideable_type. If we search by unique values, it only returns one option. The columns that we want to focus on are user types, trip duration, trip start stations, and trip end stations. The names of each might be different between the two tables, we will make sure they match later on.

```r
rideType <- trips2020 %>%
  select(rideable_type)
unique(rideType)
```

```
## # A tibble: 1 x 1
##   rideable_type
##   <chr>
## 1 docked_bike
```

Cleaning and Making the Data Pretty

In this chunk of code, we will be gathering the data that we need for the 2019 table which includes the user type and the duration of their trip that was logged. Note that there are no accompanying user ID's so there are going to be trips that are logged by the same people. This does not mean that there are more subscribers than there are normal customers but it does show how many more rides that subscribers log compared to normal customers.

```r
#This is the cleaning chunk
clean2019 <- trips2019 %>%
  mutate(start_time = ymd_hms(start_time),
         end_time = ymd_hms(end_time)) %>%
  mutate(trip_duration = end_time - start_time) %>%
  select(usertype, trip_duration)

#Making the clean time information easy to read
pretty2019 <- clean2019 %>%
  mutate(timeDiffSecs = as.numeric(trip_duration*60)) %>%
  mutate(minutes = floor(timeDiffSecs/60),
         seconds = round(timeDiffSecs%%60)) %>%
  mutate(prettyTime = paste0(minutes, " minute", ifelse(minutes == 1, " ", "s "),
                             seconds, " second", ifelse(seconds == 1," ", "s")))
```

If we then start by looking at the clean2019 variable, we will see that the trip duration has time in minutes, formatted as a float. This is not very easy to understand at first glance and will be converted into a minute and second format in the pretty2019 table.

```r
glimpse(clean2019)
```

```
## Rows: 365,069
## Columns: 2
## $ usertype      <chr> "Subscriber", "Subscriber", "Subscriber", "Subscriber", ~
## $ trip_duration <drtn> 6.500000 mins, 7.350000 mins, 13.816667 mins, 29.716667~
```

```r
glimpse(pretty2019)
```

```
## Rows: 365,069
## Columns: 6
## $ usertype      <chr> "Subscriber", "Subscriber", "Subscriber", "Subscriber", ~
## $ trip_duration <drtn> 6.500000 mins, 7.350000 mins, 13.816667 mins, 29.716667~
## $ timeDiffSecs  <dbl> 390, 441, 829, 1783, 364, 216, 177, 100, 1727, 336, 886,~
## $ minutes       <dbl> 6, 7, 13, 29, 6, 3, 2, 1, 28, 5, 14, 10, 10, 9, 15, 14, ~
## $ seconds       <dbl> 30, 21, 49, 43, 4, 36, 57, 40, 47, 36, 46, 53, 1, 22, 6,~
## $ prettyTime    <chr> "6 minutes 30 seconds", "7 minutes 21 seconds", "13 minu~
```

Let us now find the average trip duration for each user type, we can do so by grouping each user type and finding the average of their values. Running the following code chunk will make another table called tripAverage2019 and then show that table.

```r
tripAverage2019 <- pretty2019 %>%
  group_by(usertype) %>%
  summarise(averageTime = mean(timeDiffSecs, na.rm = TRUE)) %>%
  mutate(minutes = floor(averageTime/60),
         seconds = round(averageTime%%60)) %>%
  mutate(prettyTime = paste0(minutes, " minute", ifelse(minutes == 1, " ", "s "),
                             seconds, " second", ifelse(seconds == 1," ", "s"))) %>%
  select(usertype, prettyTime)

unique(tripAverage2019)
```

```
## # A tibble: 2 x 2
##   usertype   prettyTime
##   <chr>      <chr>
## 1 Customer   61 minutes 57 seconds
## 2 Subscriber 13 minutes 54 seconds
```

With this information we can see that customers that are not subscribed typically use their bikes for longer trips. Until we see the visualization we cannot assume any reasons as to why this might be.

Making the 'User Type and Their Trips' Visualization

In order to get a better understanding of why the times are so skewed we will be making a visualization of the trip duration data. This will show all the data in an easier to digest way.

```r
graphDataUserTime2019 <- pretty2019 %>%
  group_by(usertype, timeDiffSecs) %>%
  select(usertype, timeDiffSecs) %>%
  mutate(time_bracket = cut(timeDiffSecs,
                            breaks=c(0,300,600,900,1800,3600,Inf),
                            labels=c('Under 5 Min','Between 5-10 Min',
                                     'Between 10-15 Min','Between 15-30 Min',
                                     'Between 30-60 Min','Over 60 Min')
```

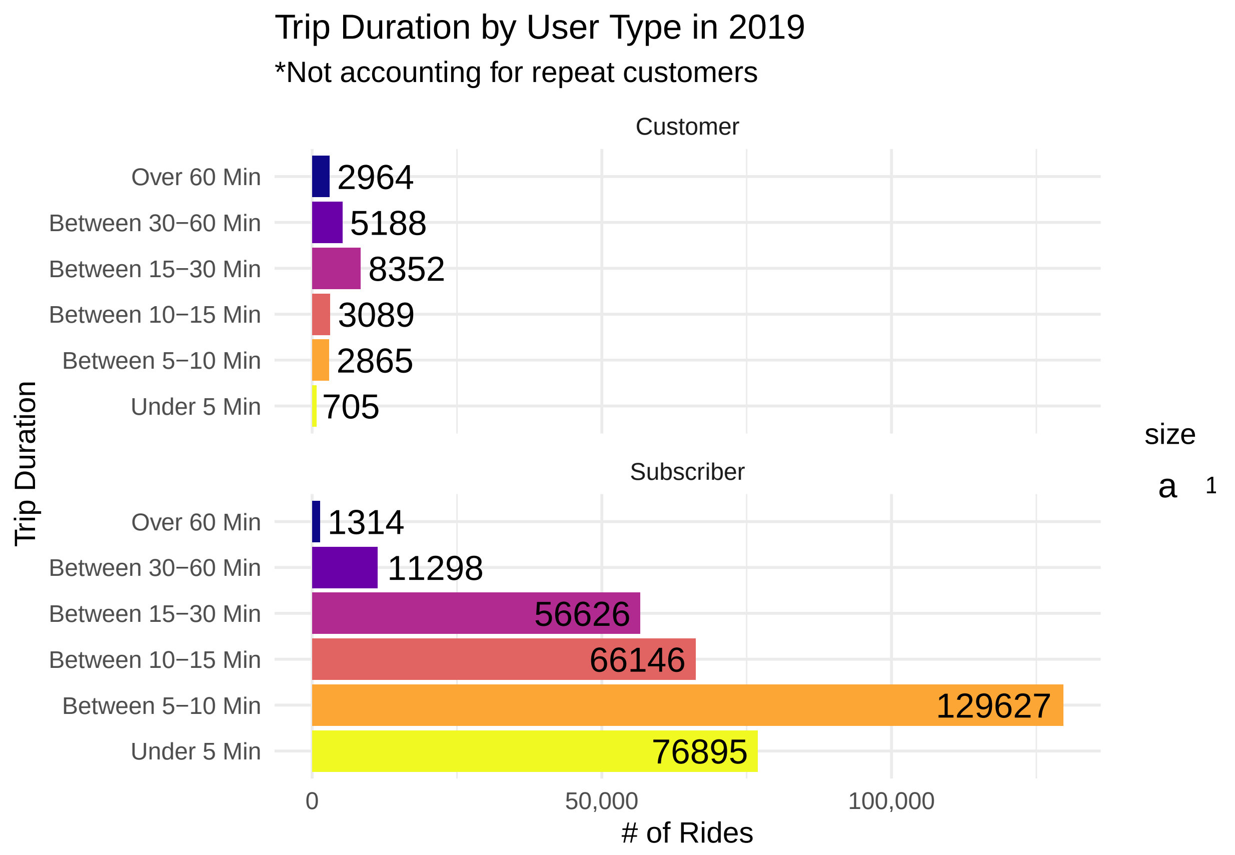
```
                          )
         ) %>%
  group_by(usertype, time_bracket) %>%
  count(time_bracket)
```

With this code chunk we are making 'buckets' for the trip duration. I felt this is necessary as without doing so, the bar graphs would have tens of thousands of data points that would overlap each other and make it difficult to view. When I put the data into their own buckets, I count each time it does so and the bars will be sized based on that information. You will notice that the buckets have different sizes, this is to make the graph even easier to read as it cuts down on the amount of bars in each graph. If you would like to make each bucket the same you can copy this code and paste it above (make sure only replace the breaks and labels part of the code.)

breaks=c(0,300,600,900,1200,1500,1800,2100,2400,2700,3000,3300,3600,Inf), labels=c('<5 Min','>5<10 Min','>10<15 Min','>15<20 Min','>20<25 Min','>25<30 Min','>30<35 Min','>35<40 Min','>40<45 Min','>45<50 Min','>50<55 Min','>55<60 Min','>60 Min')

With this next chunk, we will be setting up the actual visualization. I am making sure that there is good contrast to be as inclusive as possible. I am also making sure to label all the necessary information and making sure it is readable.

```
ggplot(graphDataUserTime2019, aes(x=time_bracket, y=n, fill=time_bracket)) +
  geom_col(show.legend = FALSE) +
  geom_text(aes(label=n,hjust=ifelse(n>50000, 1.1,-.1),size=1)) +
  facet_wrap(~usertype, ncol=1) +
  labs(title="Trip Duration by User Type in 2019",
       subtitle="*Not accounting for repeat customers",
       x="Trip Duration",
       y="# of Rides",
       color="Frequency",) +
  scale_fill_viridis_d(option="plasma", direction=-1) +
  scale_y_continuous(labels=scales::label_comma()) +
  coord_flip() +
  theme_minimal()
```



As we can see based off of the graphs, the customers ride much less frequently than subscribers do. It is not safe to assume that the non subscribed customer only makes one trip every so often while the subscribers are likely using it as a daily commute option which is why the averages are so different.

Which Stations Are the Most Used

Now we will be seeing which stations are used the most as starting, ending, and as both starting and ending a trip. With this information we will be able to tell where the bikes should have the most stock.

The following chunk makes a variable called tripRouteSummary2019 and gets a count of each time a location is used for a trip and gets the top 15 of both starting and ending locations and puts it into a single table. With it being part of a single table, there is likely going to be N/A values since one station might be more popular as a starting location but not ending. When we make our visualization we will be sure to address this issue, but it still important to see in graph format.

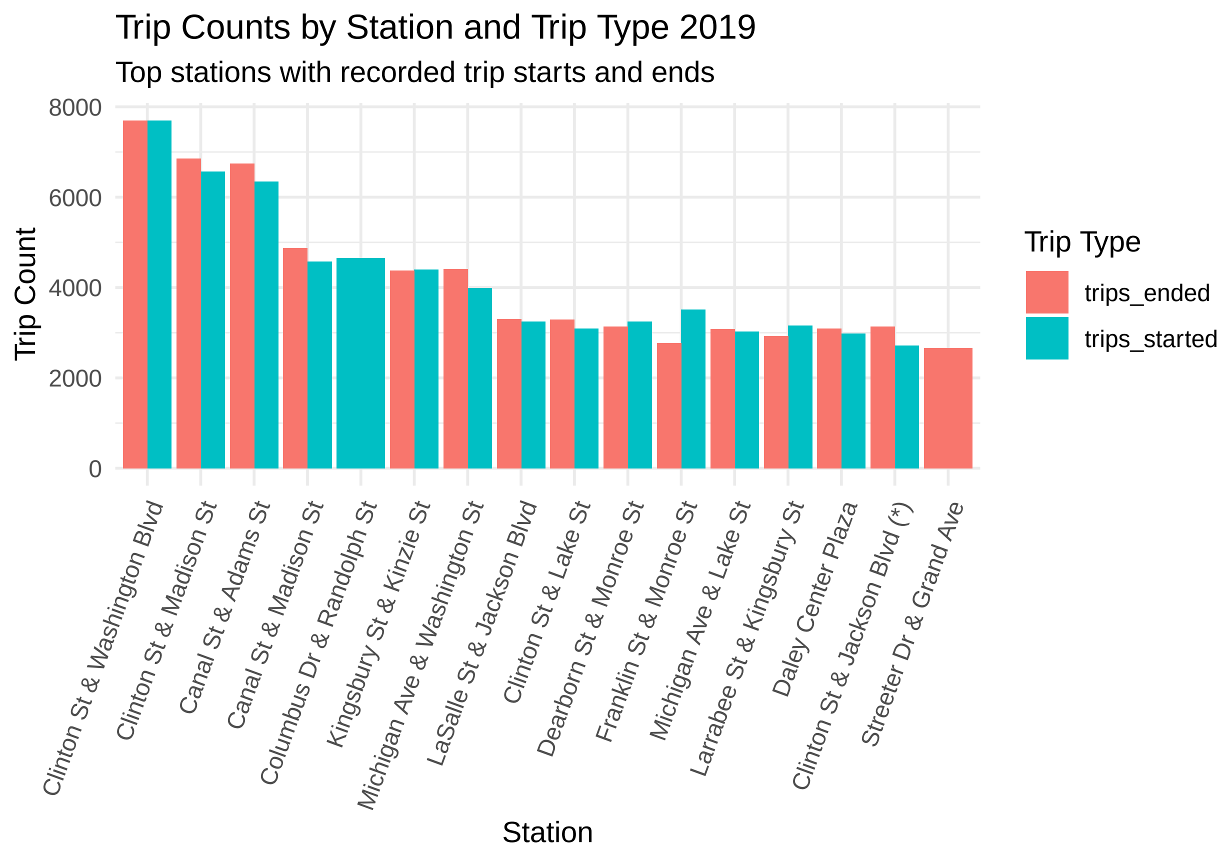
```r
tripRouteSummary2019 <- trips2019 %>%
  filter(!is.na(from_station_name) & !is.na(to_station_name)) %>%
  count(from_station_name, name="trips_started") %>%
  top_n(15, trips_started) %>%
  rename(station_name=from_station_name) %>%
  full_join(
    trips2019 %>%
      count(to_station_name, name="trips_ended") %>%
      top_n(15, trips_ended) %>%
      rename(station_name=to_station_name),
    by="station_name"
  ) %>%
  arrange(desc(coalesce(trips_started, 0) + coalesce(trips_ended, 0))) %>%
  mutate(
    station_type = case_when(
      !is.na(trips_started) & !is.na(trips_ended) ~ "Start & End",
      !is.na(trips_started) ~ "Start Only",
      !is.na(trips_ended) ~ "End Only"
    )
  ) %>%
  drop_na(station_type)
```

This next chunk will make another table with the same information but will be dealing with N/A counts. This is done with the 'pivot_longer' function. This is important to do in order the make a visualization without having to deal with N/A inputs.

```r
tripRouteSumLong2019 <- tripRouteSummary2019 %>%
  pivot_longer(cols = c(trips_started, trips_ended),
               names_to="trip_type",
               values_to="count") %>%
  drop_na(count)
```

finally we will be making a graph to show which stations are most popular and clearly show if they are popular as both starting and ending trips, or by one or the other.

```r
ggplot(tripRouteSumLong2019, aes(x = reorder(station_name, -count), y = count, fill = trip_type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Trip Counts by Station and Trip Type 2019",
       subtitle = "Top stations with recorded trip starts and ends",
       x = "Station", y = "Trip Count", fill = "Trip Type") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 70, hjust = 1))
```



Cleaning and Filtering 2020 Data

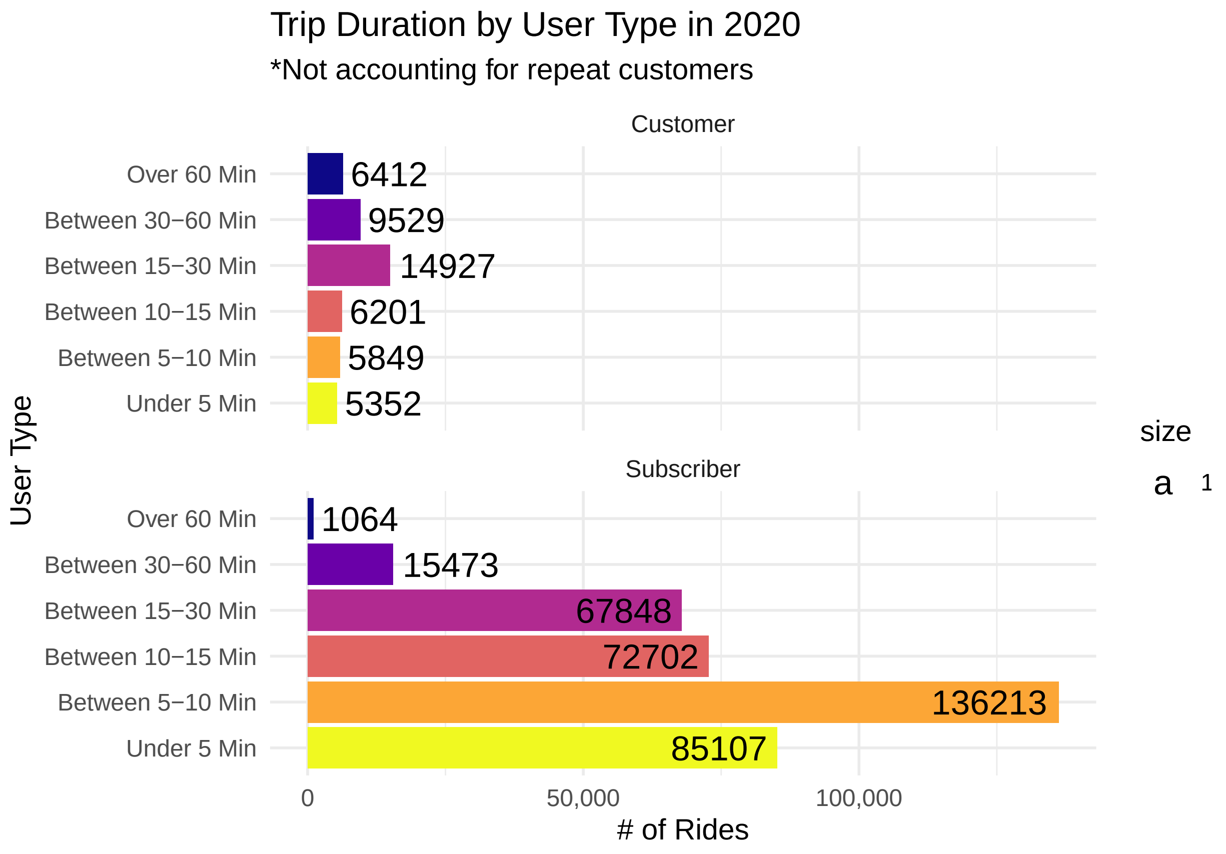
These next few chinks of code are going to be relatively the same as above. If there are any changes along the way I will describe how it is and how I cleaned it up with comments in the code chunk.

```
#Getting the same information as above but tweaking it for the 2020 table. We also want to make sure th
clean2020 <- trips2020 %>%
  rename(usertype = member_casual) %>%
  mutate(usertype = case_when(
    usertype == "member" ~ "Subscriber",
    usertype == "casual" ~ "Customer",
    TRUE ~ usertype)) %>%
  mutate(start_time = ymd_hms(started_at),
         end_time = ymd_hms(ended_at)) %>%
  mutate(trip_duration = end_time - start_time) %>%
  select(usertype, trip_duration)

pretty2020 <- clean2020 %>%
  mutate(timeDiffSecs = as.numeric(trip_duration)) %>%
  mutate(minutes = floor(timeDiffSecs/60),
         seconds = round(timeDiffSecs%%60)) %>%
  mutate(prettyTime = paste0(minutes, " minute", ifelse(minutes == 1, " ", "s "),
                             seconds, " second", ifelse(seconds == 1," ", "s")))

# The buckets are not the same in length of time just like above, if you would like to make them all th
graphDataUserTime2020 <- pretty2020 %>%
  group_by(usertype, timeDiffSecs) %>%
  select(usertype, timeDiffSecs) %>%
  mutate(time_bracket = cut(timeDiffSecs,
                            breaks=c(0,300,600,900,1800,3600,Inf),
                            labels=c('Under 5 Min','Between 5-10 Min','Between 10-15 Min','Between 15-3
                                     'Between 30-60 Min','Over 60 Min'))) %>%
  group_by(usertype, time_bracket) %>%
  count(time_bracket)

ggplot(graphDataUserTime2020 %>%
         filter(!is.na(time_bracket)), aes(x=time_bracket, y=n, fill=time_bracket)) +
  geom_col(show.legend = FALSE) +
  geom_text(aes(label=n,hjust=ifelse(n>50000, 1.1,-.1),size=1)) +
  facet_wrap(~usertype, ncol=1) +
  labs(title="Trip Duration by User Type in 2020",
       subtitle="*Not accounting for repeat customers",
       x="User Type",
       y="# of Rides",
       color="Frequency",) +
  scale_fill_viridis_d(option="plasma", direction=-1) +
  scale_y_continuous(labels=scales::label_comma()) +
  coord_flip() +
  theme_minimal()
```

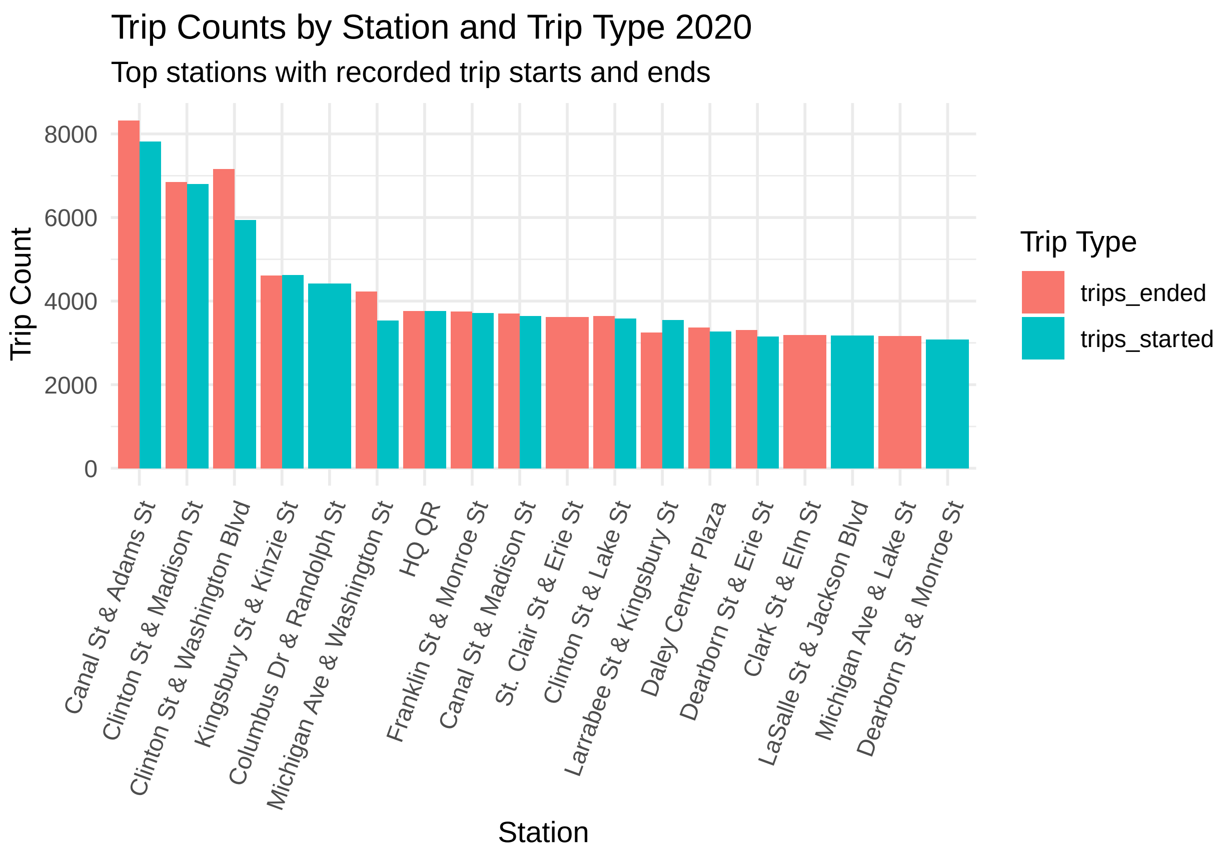


```
tripAverage2020 <- pretty2020 %>%
  group_by(usertype) %>%
  summarise(averageTime = mean(timeDiffSecs, na.rm = TRUE)) %>%
  mutate(minutes = floor(averageTime/60),
         seconds = round(averageTime%%60)) %>%
  mutate(prettyTime = paste0(minutes, " minute", ifelse(minutes == 1, " ", "s "),
                              seconds, " second", ifelse(seconds == 1," ", "s"))) %>%
  select(usertype, prettyTime)

tripRouteSummary2020 <- trips2020 %>%
  filter(!is.na(start_station_name) & !is.na(end_station_name)) %>%
  count(start_station_name, name="trips_started") %>%
  top_n(15, trips_started) %>%
  rename(station_name=start_station_name)%>%
  full_join(
    trips2020 %>%
      count(end_station_name, name="trips_ended") %>%
      top_n(15, trips_ended) %>%
      rename(station_name=end_station_name),
    by="station_name"
  ) %>%
  arrange(desc(coalesce(trips_started, 0) + coalesce(trips_ended, 0))) %>%
  mutate(
    station_type = case_when(
      !is.na(trips_started) & !is.na(trips_ended) ~ "Start & End",
      !is.na(trips_started) ~ "Start Only",
      !is.na(trips_ended) ~ "End Only"
    )
  ) %>%
  drop_na(station_type)

tripRouteSumLong2020 <- tripRouteSummary2020 %>%
  pivot_longer(cols = c(trips_started, trips_ended),
               names_to="trip_type",
               values_to="count") %>%
  drop_na(count)

ggplot(tripRouteSumLong2020, aes(x = reorder(station_name, -count), y = count, fill = trip_type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Trip Counts by Station and Trip Type 2020",
       subtitle = "Top stations with recorded trip starts and ends",
       x = "Station", y = "Trip Count", fill = "Trip Type") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 70, hjust = 1))
```



Comparing 2019 and 2020 Data.

Now that we have visualization for both 2019 and 2020 data, we can see that the company has grown based on the amount of rides.