Untitled

Fatih CAKSEN

2024-02-04

```
library(tidyverse)
```

```
## -- 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   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- 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
```

```
library(lubridate)
library(ggplot2)
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(dplyr)
library(rmarkdown)
library(tinytex)
library(marginaleffects)
library(modelsummary)
library(sandwich)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(ggrepel)
library(stringr)
library(readr)
library(patchwork)
library(jtools)
```

```
library(knitr)
```

Step 1: Load Data

```
divvy_2019 <- read.csv("C:/Users/Fatih/Desktop/Final_Project/DataSets/Divvy_2019_Q1.csv")
divvy_2020 <- read.csv("C:/Users/Fatih/Desktop/Final_Project/DataSets/Divvy_2020_Q1.csv")
```

Step 2: Wrangle Data and Combine into a Single File

Compare Column Names Each of the Files

```
colnames(divvy_2019)
```

```
##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "bikeid"            "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "gender"            "birthyear"
```

```
colnames(divvy_2020)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "start_station_id"
##  [7] "end_station_name"   "end_station_id"     "start_lat"
## [10] "start_lng"          "end_lat"            "end_lng"
## [13] "member_casual"      "ride_length"        "ride_length_in_min"
## [16] "day_of_week"
```

#Rename Columns for Consistency

```
divvy_2019 <- divvy_2019 %>%
  rename(
    start_station_id = from_station_id,
    end_station_id = to_station_id,
    start_station_name = from_station_name,
    end_station_name = to_station_name
  )
```

```
divvy_2020 <- divvy_2020 %>%
  rename(
    trip_id = ride_id,
    usertype = member_casual,
    start_time = started_at,
    end_time = ended_at
  )
```

Inspect the dataframes and look for incongruencies

```
str(divvy_2019)
```

```
## 'data.frame':    365069 obs. of  12 variables:
##  $ trip_id           : int  21742443 21742444 21742445 21742446 21742447 21742448 21742449 21742450 2
##  $ start_time        : chr  "2019-01-01 0:04:37" "2019-01-01 0:08:13" "2019-01-01 0:13:23" "2019-01-0
```

```
##  $ end_time          : chr  "2019-01-01 0:11:07" "2019-01-01 0:15:34" "2019-01-01 0:27:12" "2019-01-
##  $ bikeid            : int  2167 4386 1524 252 1170 2437 2708 2796 6205 3939 ...
##  $ tripduration      : chr  "390" "441" "829" "1,783.00" ...
##  $ start_station_id  : int  199 44 15 123 173 98 98 211 150 268 ...
##  $ start_station_name: chr  "Wabash Ave & Grand Ave" "State St & Randolph St" "Racine Ave & 18th St"
##  $ end_station_id    : int  84 624 644 176 35 49 49 142 148 141 ...
##  $ end_station_name  : chr  "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren St (*)" "Western Av
##  $ usertype          : chr  "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
##  $ gender            : chr  "Male" "Female" "Female" "Male" ...
##  $ birthyear         : int  1989 1990 1994 1993 1994 1983 1984 1990 1995 1996 ...
```

#Remove Unnecessary Variables

```
divvy_2019 <- divvy_2019 %>%
  select(-c(gender, birthyear,bikeid,tripduration))
```

To ensure that the trip_id values stack correnctly, it's necessary to convert them to characters.

```
divvy_2019 <- mutate(divvy_2019, trip_id = as.character(trip_id))
```

```
str(divvy_2019)
```

```
## 'data.frame':    365069 obs. of  8 variables:
##  $ trip_id           : chr  "21742443" "21742444" "21742445" "21742446" ...
##  $ start_time        : chr  "2019-01-01 0:04:37" "2019-01-01 0:08:13" "2019-01-01 0:13:23" "2019-01-
##  $ end_time          : chr  "2019-01-01 0:11:07" "2019-01-01 0:15:34" "2019-01-01 0:27:12" "2019-01-
##  $ start_station_id  : int  199 44 15 123 173 98 98 211 150 268 ...
##  $ start_station_name: chr  "Wabash Ave & Grand Ave" "State St & Randolph St" "Racine Ave & 18th St"
##  $ end_station_id    : int  84 624 644 176 35 49 49 142 148 141 ...
##  $ end_station_name  : chr  "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren St (*)" "Western Av
##  $ usertype          : chr  "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
```

```
str(divvy_2020)
```

```
## 'data.frame':    426887 obs. of  16 variables:
##  $ trip_id           : chr  "EACB19130B0CDA4A" "8FED874C809DC021" "789F3C21E472CA96" "C9A388DAC6ABF3
##  $ rideable_type     : chr  "docked_bike" "docked_bike" "docked_bike" "docked_bike" ...
##  $ start_time        : chr  "2020-01-21 20:06:59" "2020-01-30 14:22:39" "2020-01-09 19:29:26" "2020-
##  $ end_time          : chr  "2020-01-21 20:14:30" "2020-01-30 14:26:22" "2020-01-09 19:32:17" "2020-
##  $ start_station_name: chr  "Western Ave & Leland Ave" "Clark St & Montrose Ave" "Broadway & Belmont
##  $ start_station_id  : int  239 234 296 51 66 212 96 96 212 38 ...
##  $ end_station_name  : chr  "Clark St & Leland Ave" "Southport Ave & Irving Park Rd" "Wilton Ave & Be
##  $ end_station_id    : int  326 318 117 24 212 96 212 212 96 100 ...
##  $ start_lat         : num  42 42 41.9 41.9 41.9 ...
##  $ start_lng         : num  -87.7 -87.7 -87.6 -87.6 -87.6 ...
##  $ end_lat           : num  42 42 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.7 -87.7 -87.7 -87.6 -87.6 ...
##  $ usertype          : chr  "member" "member" "member" "member" ...
##  $ ride_length       : chr  "0:07:31" "0:03:43" "0:02:51" "0:08:49" ...
##  $ ride_length_in_min: chr  "168:00:00" "89:12:00" "68:24:00" "211:36:00" ...
##  $ day_of_week       : int  3 5 5 2 5 6 6 6 6 6 ...
```

```
divvy_2020 <- divvy_2020 %>%
  select(-c(start_lat, end_lat,start_lng,end_lng,rideable_type,ride_length,ride_length_in_min,day_of_we
```

Creating Combined Data Frame

```
combined_data <- bind_rows(divvy_2019,divvy_2020)
```

```
colnames(combined_data)
```

```
## [1] "trip_id"            "start_time"         "end_time"
## [4] "start_station_id"   "start_station_name" "end_station_id"
## [7] "end_station_name"   "usertype"
```

```
str(combined_data)
```

```
## 'data.frame':    791956 obs. of  8 variables:
##  $ trip_id           : chr  "21742443" "21742444" "21742445" "21742446" ...
##  $ start_time        : chr  "2019-01-01 0:04:37" "2019-01-01 0:08:13" "2019-01-01 0:13:23" "2019-01-
##  $ end_time          : chr  "2019-01-01 0:11:07" "2019-01-01 0:15:34" "2019-01-01 0:27:12" "2019-01-
##  $ start_station_id  : int  199 44 15 123 173 98 98 211 150 268 ...
##  $ start_station_name: chr  "Wabash Ave & Grand Ave" "State St & Randolph St" "Racine Ave & 18th St"
##  $ end_station_id    : int  84 624 644 176 35 49 49 142 148 141 ...
##  $ end_station_name  : chr  "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren St (*)" "Western Av
##  $ usertype          : chr  "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
```

```
# I checked the new data frame
colnames(combined_data)  # Column Names
```

```
## [1] "trip_id"            "start_time"         "end_time"
## [4] "start_station_id"   "start_station_name" "end_station_id"
## [7] "end_station_name"   "usertype"
```

```
nrow(combined_data)  # Numbers of the Rows
```

```
## [1] 791956
```

```
dim(combined_data)  #Dimensions
```

```
## [1] 791956      8
```

```
head(combined_data)  # The first 6 rows of the combined data frame
```

```
##    trip_id         start_time           end_time start_station_id
## 1 21742443 2019-01-01 0:04:37 2019-01-01 0:11:07              199
## 2 21742444 2019-01-01 0:08:13 2019-01-01 0:15:34               44
## 3 21742445 2019-01-01 0:13:23 2019-01-01 0:27:12               15
## 4 21742446 2019-01-01 0:13:45 2019-01-01 0:43:28              123
## 5 21742447 2019-01-01 0:14:52 2019-01-01 0:20:56              173
## 6 21742448 2019-01-01 0:15:33 2019-01-01 0:19:09               98
##                    start_station_name end_station_id
## 1              Wabash Ave & Grand Ave             84
## 2              State St & Randolph St            624
## 3                Racine Ave & 18th St            644
## 4      California Ave & Milwaukee Ave            176
## 5 Mies van der Rohe Way & Chicago Ave             35
## 6          LaSalle St & Washington St             49
##                  end_station_name   usertype
## 1      Milwaukee Ave & Grand Ave Subscriber
## 2 Dearborn St & Van Buren St (*) Subscriber
## 3  Western Ave & Fillmore St (*) Subscriber
## 4              Clark St & Elm St Subscriber
```

```
## 5        Streeter Dr & Grand Ave Subscriber
## 6        Dearborn St & Monroe St Subscriber
```

```r
str(combined_data)  # List of Columns and Data Types
```

```
## 'data.frame':    791956 obs. of  8 variables:
##  $ trip_id           : chr  "21742443" "21742444" "21742445" "21742446" ...
##  $ start_time        : chr  "2019-01-01 0:04:37" "2019-01-01 0:08:13" "2019-01-01 0:13:23" "2019-01-0
##  $ end_time          : chr  "2019-01-01 0:11:07" "2019-01-01 0:15:34" "2019-01-01 0:27:12" "2019-01-0
##  $ start_station_id  : int  199 44 15 123 173 98 98 211 150 268 ...
##  $ start_station_name: chr  "Wabash Ave & Grand Ave" "State St & Randolph St" "Racine Ave & 18th St"
##  $ end_station_id    : int  84 624 644 176 35 49 49 142 148 141 ...
##  $ end_station_name  : chr  "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren St (*)" "Western Av
##  $ usertype          : chr  "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
```

```r
summary(combined_data) # Summary of the Combined Data Frame
```

```
##    trip_id           start_time          end_time         start_station_id
##  Length:791956      Length:791956      Length:791956      Min.   :  2.0
##  Class :character   Class :character   Class :character   1st Qu.: 77.0
##  Mode  :character   Mode  :character   Mode  :character   Median :174.0
##                                                           Mean   :204.4
##                                                           3rd Qu.:291.0
##                                                           Max.   :675.0
##
##  start_station_name end_station_id  end_station_name     usertype
##  Length:791956      Min.   :  2.0   Length:791956      Length:791956
##  Class :character   1st Qu.: 77.0   Class :character   Class :character
##  Mode  :character   Median :174.0   Mode  :character   Mode  :character
##                     Mean   :204.4
##                     3rd Qu.:291.0
##                     Max.   :675.0
##                     NA's   :1
```

Step 3: Clean Up and Add Data to Prepare for Analysis

#Remove NA and Duplicate Data

```r
# Remove rows with NAs
combined_data <- na.omit(combined_data)

# Remove duplicate rows
combined_data <- distinct(combined_data)

# Check the updated structure of the data frame
str(combined_data)
```

```
## 'data.frame':    791955 obs. of  8 variables:
##  $ trip_id           : chr  "21742443" "21742444" "21742445" "21742446" ...
##  $ start_time        : chr  "2019-01-01 0:04:37" "2019-01-01 0:08:13" "2019-01-01 0:13:23" "2019-01-0
##  $ end_time          : chr  "2019-01-01 0:11:07" "2019-01-01 0:15:34" "2019-01-01 0:27:12" "2019-01-0
##  $ start_station_id  : int  199 44 15 123 173 98 98 211 150 268 ...
##  $ start_station_name: chr  "Wabash Ave & Grand Ave" "State St & Randolph St" "Racine Ave & 18th St"
##  $ end_station_id    : int  84 624 644 176 35 49 49 142 148 141 ...
##  $ end_station_name  : chr  "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren St (*)" "Western Av
##  $ usertype          : chr  "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
```

```
##  - attr(*, "na.action")= 'omit' Named int 779496
##   ..- attr(*, "names")= chr "779496"
```

Checking variables and preparing data for analysis.

```
# We will modify the values so that the data becomes consistent.
combined_data <- combined_data %>%
  mutate(usertype = recode(usertype
                           , "member" = "Subscriber"
                           , "casual" = "Customer"))
```

Confirming normal distribution of observations.

```
table(combined_data$usertype)
```

```
##
##   Customer Subscriber
##      71642     720313
```

Adding "ride_length" to combined_data in seconds.

```
# Convert start_time and end_time to POSIXct objects
combined_data$start_time <- as.POSIXct(combined_data$start_time)
combined_data$end_time <- as.POSIXct(combined_data$end_time)

# Calculate ride_length in seconds
combined_data$ride_length <- as.numeric(difftime(combined_data$end_time, combined_data$start_time, units

# Check the updated structure of the data frame
str(combined_data)
```

```
## 'data.frame':    791955 obs. of  9 variables:
##  $ trip_id           : chr  "21742443" "21742444" "21742445" "21742446" ...
##  $ start_time        : POSIXct, format: "2019-01-01 00:04:37" "2019-01-01 00:08:13" ...
##  $ end_time          : POSIXct, format: "2019-01-01 00:11:07" "2019-01-01 00:15:34" ...
##  $ start_station_id  : int  199 44 15 123 173 98 98 211 150 268 ...
##  $ start_station_name: chr  "Wabash Ave & Grand Ave" "State St & Randolph St" "Racine Ave & 18th St"
##  $ end_station_id    : int  84 624 644 176 35 49 49 142 148 141 ...
##  $ end_station_name  : chr  "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren St (*)" "Western Av
##  $ usertype          : chr  "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
##  $ ride_length       : num  390 441 829 1783 364 ...
##  - attr(*, "na.action")= 'omit' Named int 779496
##   ..- attr(*, "names")= chr "779496"
```

```
# Check for duplicate values in ride_length
duplicate_ride_length <- any(duplicated(combined_data$ride_length))

# Check for NA values in ride_length
na_ride_length <- any(is.na(combined_data$ride_length))

# Print the results
cat("Duplicate ride_length values:", duplicate_ride_length, "\n")
```

```
## Duplicate ride_length values: TRUE
```

```
cat("NA ride_length values:", na_ride_length, "\n")
```

```
## NA ride_length values: FALSE
```

Add date columns such as month, day and year to the combined_data frame.

This will enable us to aggregate ride data by month, day, or year.

```
combined_data$date <- as.Date(combined_data$start_time)
combined_data$month <- format(as.Date(combined_data$date), "%m")
combined_data$day <- format(as.Date(combined_data$date), "%d")
combined_data$year <- format(as.Date(combined_data$date), "%Y")
combined_data$day_of_week <- format(as.Date(combined_data$date), "%A")
```

Inspecting column structure.

```
str(combined_data)
```

```
## 'data.frame':    791955 obs. of  14 variables:
##  $ trip_id           : chr  "21742443" "21742444" "21742445" "21742446" ...
##  $ start_time        : POSIXct, format: "2019-01-01 00:04:37" "2019-01-01 00:08:13" ...
##  $ end_time          : POSIXct, format: "2019-01-01 00:11:07" "2019-01-01 00:15:34" ...
##  $ start_station_id  : int  199 44 15 123 173 98 98 211 150 268 ...
##  $ start_station_name: chr  "Wabash Ave & Grand Ave" "State St & Randolph St" "Racine Ave & 18th St"
##  $ end_station_id    : int  84 624 644 176 35 49 49 142 148 141 ...
##  $ end_station_name  : chr  "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren St (*)" "Western Av
##  $ usertype          : chr  "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
##  $ ride_length       : num  390 441 829 1783 364 ...
##  $ date              : Date, format: "2019-01-01" "2019-01-01" ...
##  $ month             : chr  "01" "01" "01" "01" ...
##  $ day               : chr  "01" "01" "01" "01" ...
##  $ year              : chr  "2019" "2019" "2019" "2019" ...
##  $ day_of_week       : chr  "Tuesday" "Tuesday" "Tuesday" "Tuesday" ...
##  - attr(*, "na.action")= 'omit' Named int 779496
##   ..- attr(*, "names")= chr "779496"
```

To ensure accuracy of data, it is necessary to remove the insufficient data from the combined_data frame which contains a some entries with negative ride_length because some bikes checked for quality by Cyclistic. Then We will create a new version of the dataframe after removing the insufficient data.

```
combined_data <- combined_data[combined_data$ride_length >= 0, ]
```

```
bike_share <- combined_data[!(combined_data$start_station_name == "HQ QR" | combined_data$ride_length<0
```

Step 4: Conduct Descriptive Analysis

Summary Statistics for Bike Share Data

```
summary(bike_share)
```

```
##    trip_id             start_time
##  Length:788189      Min.   :2019-01-01 00:04:37.00
##  Class :character   1st Qu.:2019-02-28 13:39:58.00
##  Mode  :character   Median :2020-01-07 07:59:53.00
##                     Mean   :2019-08-31 14:14:43.81
##                     3rd Qu.:2020-02-19 12:38:46.00
##                     Max.   :2020-03-31 23:51:34.00
##     end_time                       start_station_id start_station_name
##  Min.   :2019-01-01 00:11:07.00   Min.   :  2.0    Length:788189
##  1st Qu.:2019-02-28 13:51:45.00   1st Qu.: 77.0    Class :character
##  Median :2020-01-07 08:10:57.00   Median :174.0    Mode  :character
##  Mean   :2019-08-31 14:34:33.26   Mean   :202.2
##  3rd Qu.:2020-02-19 12:57:45.00   3rd Qu.:289.0
##  Max.   :2020-05-19 20:10:34.00   Max.   :673.0
##  end_station_id  end_station_name     usertype          ride_length
##  Min.   :  2.0   Length:788189      Length:788189      Min.   :       1
##  1st Qu.: 77.0   Class :character   Class :character   1st Qu.:     331
##  Median :173.0   Mode  :character   Mode  :character   Median :     539
##  Mean   :202.1                                         Mean   :    1189
##  3rd Qu.:289.0                                         3rd Qu.:     912
##  Max.   :675.0                                         Max.   :10632022
##       date               month               day                year
##  Min.   :2019-01-01   Length:788189      Length:788189      Length:788189
##  1st Qu.:2019-02-28   Class :character   Class :character   Class :character
##  Median :2020-01-07   Mode  :character   Mode  :character   Mode  :character
##  Mean   :2019-08-31
##  3rd Qu.:2020-02-19
##  Max.   :2020-04-01
##  day_of_week
##  Length:788189
##  Class :character
##  Mode  :character
##
##
##
```

Comparison Between Subscriber and Customer Users

```
# Comparison of subscribers and customers.
aggregate(bike_share$ride_length ~ bike_share$usertype, FUN = mean)
```

```
##   bike_share$usertype bike_share$ride_length
## 1            Customer              5372.7839
## 2          Subscriber               795.2523
```

```
aggregate(bike_share$ride_length ~ bike_share$usertype, FUN = median)
```

```
##   bike_share$usertype bike_share$ride_length
```

```
## 1            Customer                   1393
## 2          Subscriber                    508
```

```
aggregate(bike_share$ride_length ~ bike_share$usertype, FUN = max)
```

```
##   bike_share$usertype bike_share$ride_length
## 1            Customer               10632022
## 2          Subscriber                6096428
```

```
aggregate(combined_data$ride_length ~ combined_data$usertype, FUN = min)
```

```
##   combined_data$usertype combined_data$ride_length
## 1               Customer                         0
## 2             Subscriber                         1
```

```
# The average ride time for subscribers and customers per day.
aggregate (bike_share$ride_length ~ bike_share$usertype + bike_share$day_of_week, FUN = mean)
```

```
##    bike_share$usertype bike_share$day_of_week bike_share$ride_length
## 1             Customer                 Friday              6729.3254
## 2           Subscriber                 Friday               754.0477
## 3             Customer                 Monday              4511.3061
## 4           Subscriber                 Monday               816.3495
## 5             Customer               Saturday              5388.6502
## 6           Subscriber               Saturday               936.7971
## 7             Customer                 Sunday              5159.2264
## 8           Subscriber                 Sunday              1012.5387
## 9             Customer               Thursday              6997.1665
## 10          Subscriber               Thursday               715.1399
## 11            Customer                Tuesday              4414.2919
## 12          Subscriber                Tuesday               814.3137
## 13            Customer              Wednesday              4525.9530
## 14          Subscriber              Wednesday               699.3865
```

```
# I fixed the order of the days of the week.
bike_share$day_of_week <- ordered(bike_share$day_of_week, levels=c("Sunday", "Monday", "Tuesday", "Wedn
```

```
# Categorizing the bike share data by the type of riders and the day of the week.

bike_share %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%
  group_by(usertype, weekday) %>%
  summarise(
    number_of_rides = n(),
    average_duration = mean(ride_length)
  ) %>%
  arrange(usertype, weekday)
```

```
## `summarise()` has grouped output by 'usertype'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 14 x 4
## # Groups:   usertype [2]
##    usertype   weekday number_of_rides average_duration
##    <chr>      <ord>             <int>            <dbl>
##  1 Customer   Sun               18652            5061.
##  2 Customer   Mon                5591            4752.
##  3 Customer   Tue                7311            4562.
```

```
##  4 Customer   Wed               7690            4480.
##  5 Customer   Thu               7147            8452.
##  6 Customer   Fri               8013            6091.
##  7 Customer   Sat              13473            4951.
##  8 Subscriber Sun              60197             973.
##  9 Subscriber Mon             110430             822.
## 10 Subscriber Tue             127974             769.
## 11 Subscriber Wed             121902             712.
## 12 Subscriber Thu             125228             707.
## 13 Subscriber Fri             115168             797.
## 14 Subscriber Sat              59413             974.
```

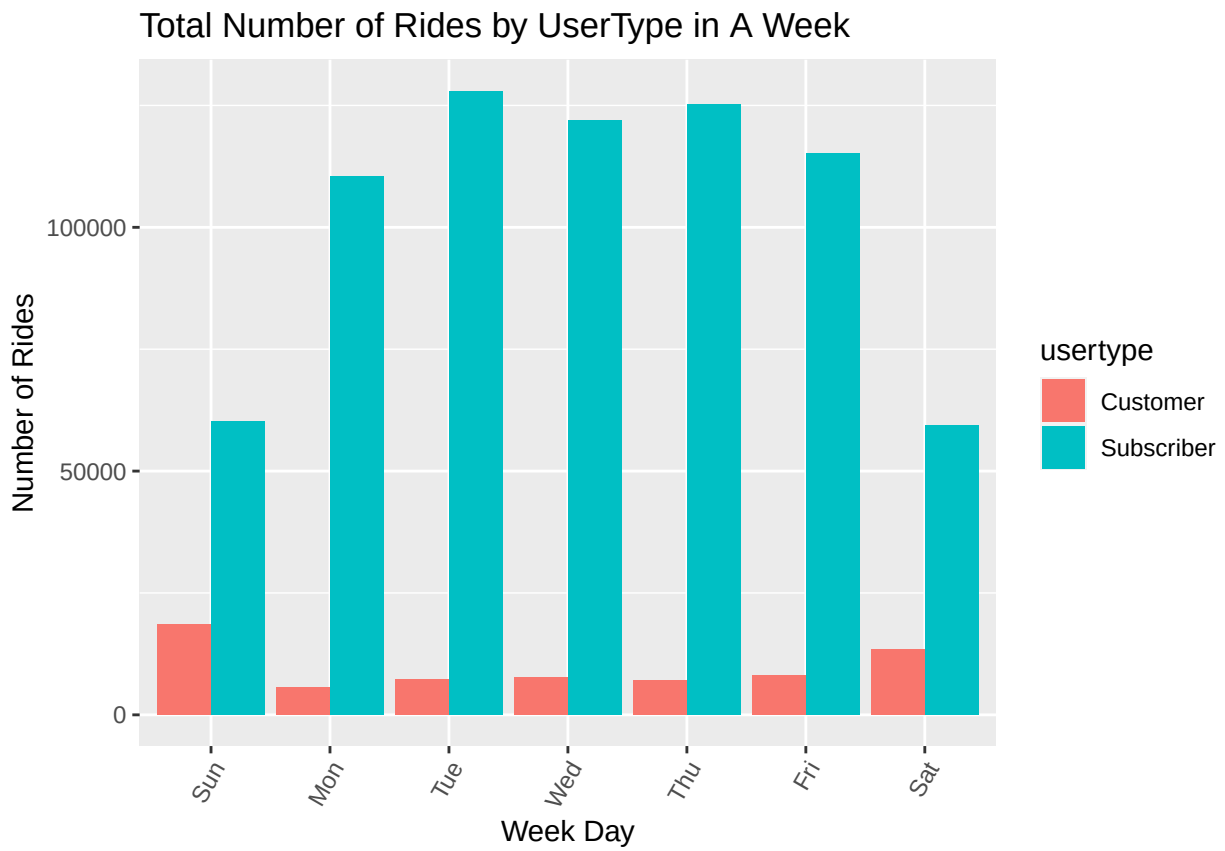
```r
# When creating a ggplot for numerical distribution, it is important to ensure that the entire number i
options(scipen=999)
```

STEP 5: Visualizations

```r
# This code calculates the total number of rides and average ride duration for each user type and day o

bike_share %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%
  group_by(usertype, weekday) %>%
  summarise(
    number_of_rides = n(),
    average_duration = mean(ride_length)
  ) %>%
  arrange(usertype, weekday) %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = usertype)) +
  geom_col(position = "dodge") +
  labs(title = "Total Number of Rides by UserType in A Week", x = "Week Day",
       y = "Number of Rides") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1))
```

```
## `summarise()` has grouped output by 'usertype'. You can override using the
## `.groups` argument.
```

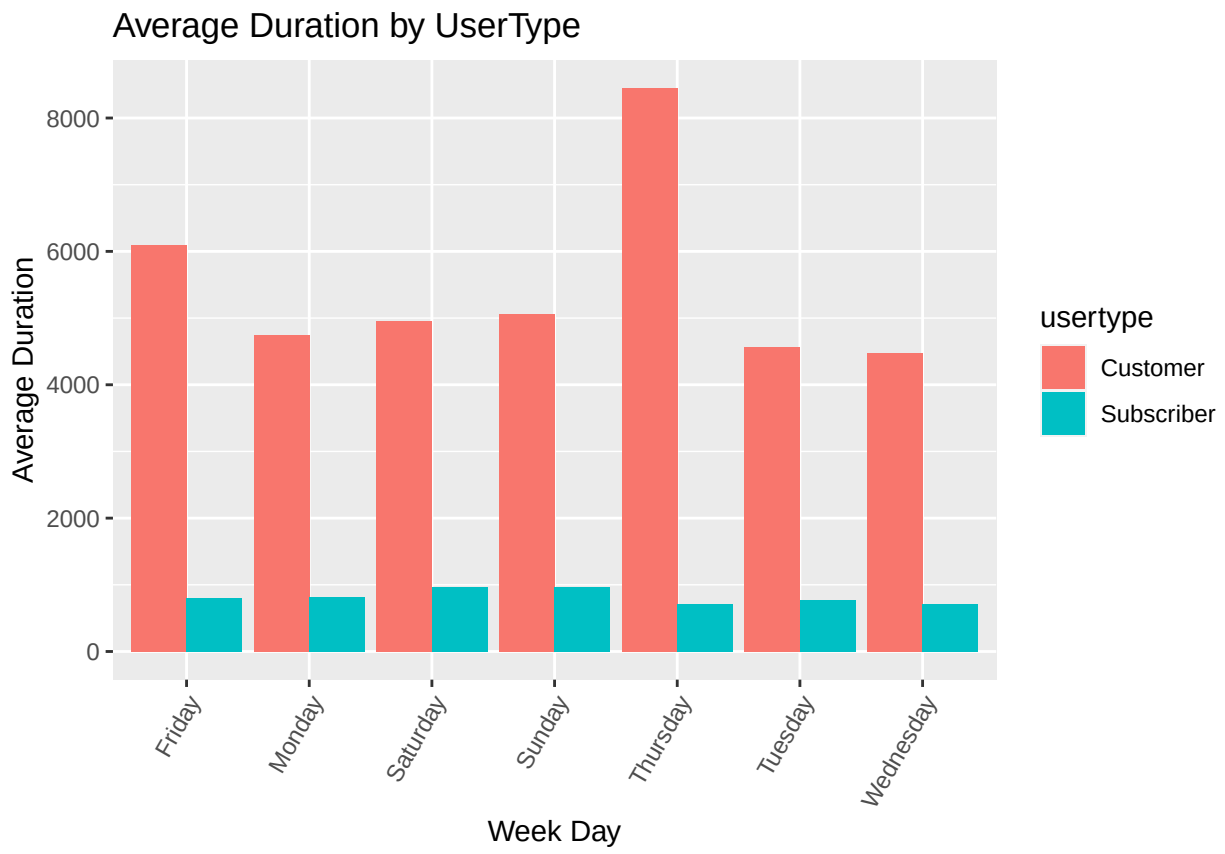


The number of rides taken by customers is significanly greater in comparison to that of subscribers. This suggests that customers tend to utilize the ride-sharing service more frequenty than subscribers.

```
# This code calculates the average ride duration by user type and day of the week, and creates a groupe

bike_share %>%
  mutate(weekday = format(start_time, "%A")) %>%
  group_by(usertype, weekday) %>%
  summarise(number_of_rides = n(), average_duration = mean(ride_length)) %>%
  arrange(usertype, weekday) %>%
  ggplot(aes(x = weekday, y = average_duration, fill = usertype)) +
  geom_col(position = "dodge") +
  labs(title = "Average Duration by UserType",
       x = "Week Day",
       y = "Average Duration") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1))
```

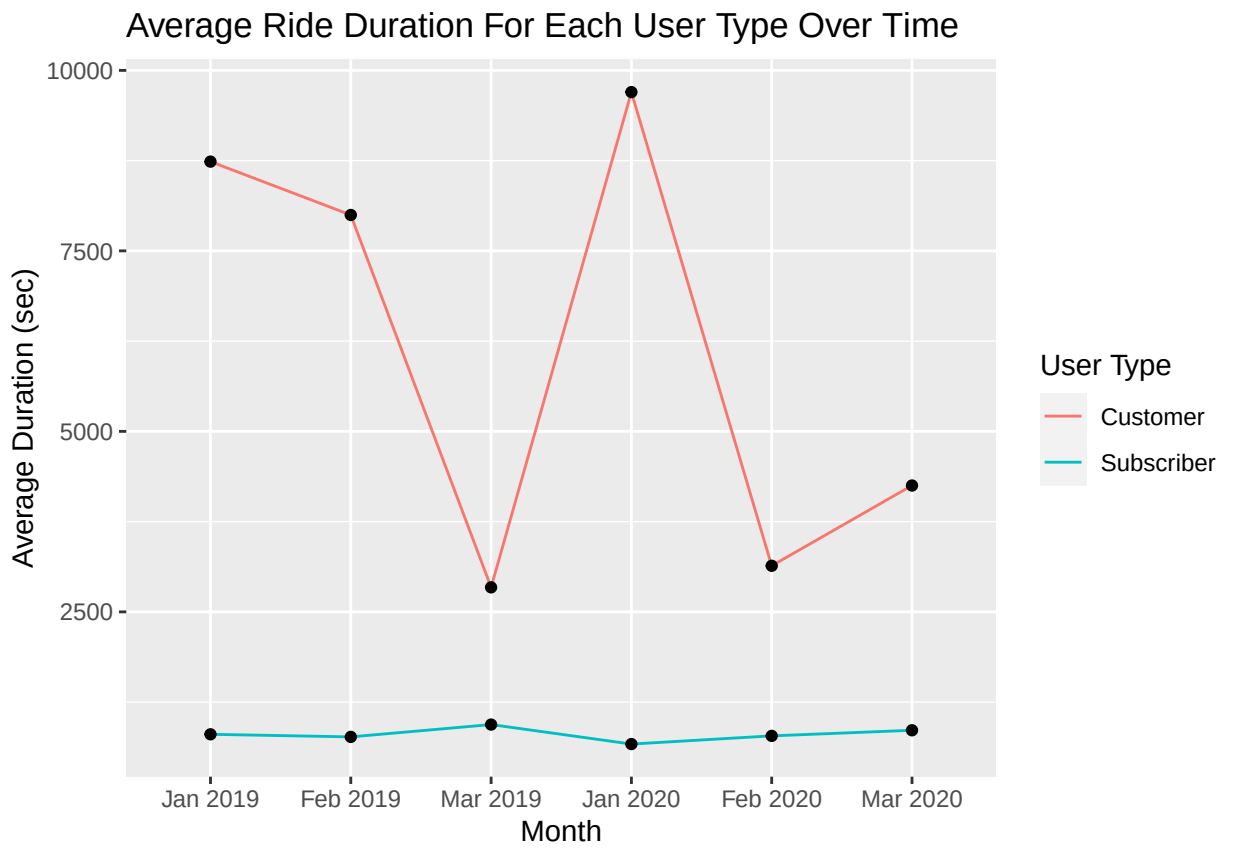
```
## `summarise()` has grouped output by 'usertype'. You can override using the
## `.groups` argument.
```



The data indicatetes that customers tend to take longer rides compared to subscribers. Weekdays appear to be busier for customers as their average ride duration is higher during this period , while subscribers' ride duration remains consistent throughout the week. This information can help identify patterns and trends in ride behavior and assist in optimizing the service to better cater to the needs of both customers and subscribers.

```
# This code calculates average ride duration for users types and months, and  visualizes the trends ove
bike_share %>%
  mutate(month = format(start_time, "%b %Y")) %>%
  group_by(usertype, month) %>%
  summarise(average_duration = mean(ride_length)) %>%
  arrange(usertype, month) %>%
  ggplot(aes(x = month, y = average_duration, group = usertype)) +
  geom_line(aes(color = usertype)) +
  geom_point() +
  labs(title = "Average Ride Duration For Each User Type Over Time",
       x = "Month",
       y = "Average Duration (sec)",
       color = "User Type") +
  scale_x_discrete(labels = c("Jan 2019", "Feb 2019", "Mar 2019", "Jan 2020", "Feb 2020", "Mar 2020"))
```

```
## `summarise()` has grouped output by 'usertype'. You can override using the
## `.groups` argument.
```



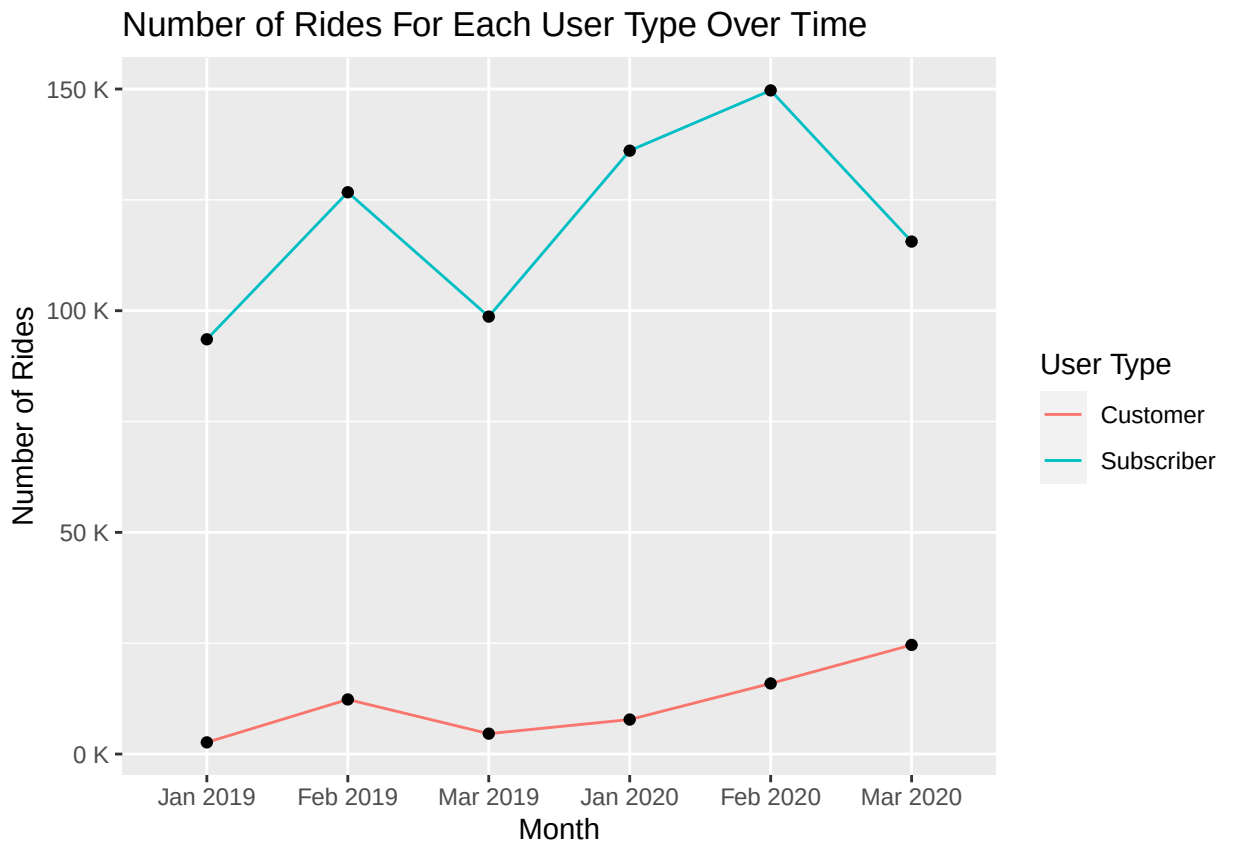
Customers rode for longer than subscribers during the first three months of 2019 and 2020, with a peak in the first two months of 2019 and a high start in 2020. The avereage ride duration dropped in the third month of 2019 and in February 2020. Subscribers' avereage ride duration remained stable in both years, indicating a more predictable travel pattern.

```
#I calculated the total rides for each user type and month, and plots a line graph using ggplot to visu

bike_share %>%
  mutate(month = format(start_time, "%b %Y")) %>%
  group_by(usertype, month) %>%
  summarise(number_of_rides = n()) %>%
  arrange(usertype, month) %>%
  ggplot(aes(x = month, y = number_of_rides, group = usertype)) +
  geom_line(aes(color = usertype)) +
  geom_point() +
  labs(title = "Number of Rides For Each User Type Over Time",
       x = "Month",
       y = "Number of Rides",
       color = "User Type") +
  scale_y_continuous(labels = scales::label_number(suffix = " K", scale = 1e-3)) +
  scale_x_discrete(labels = c("Jan 2019", "Feb 2019", "Mar 2019", "Jan 2020", "Feb 2020", "Mar 2020"))
```

```
## `summarise()` has grouped output by 'usertype'. You can override using the
```

```
## `.groups` argument.
```



The usage of bikes among subscribers far outweighs that of customers. In other words, subscribers are more inclined to use bikes than customers. While the number of bike rides by customers is on the rise, it is still significantly lower than the number of bike rides taken by subscribers.

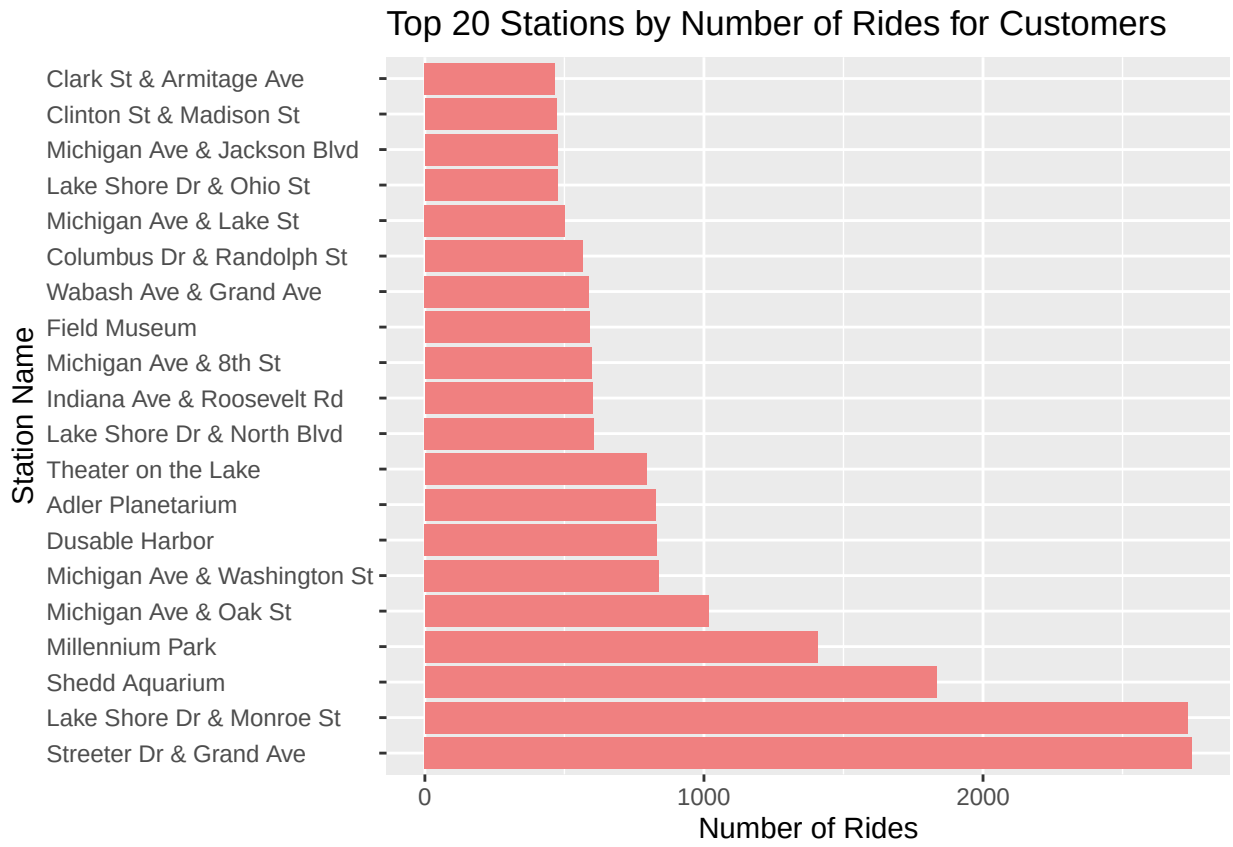
Top 20 Station Name booked for the Customers

```
# I filtered and summarized data to identify the top 20 bike share stations with the highest number of

top_stations_customers <- bike_share %>%
  filter(!is.na(start_station_name)) %>%
  filter(usertype == "Customer") %>%
  group_by(start_station_name) %>%
  summarise(number_of_rides = n(),
            avg_ride_length = mean(ride_length),
            avg_ride_length_min = mean(ride_length) / 60) %>%
  arrange(desc(number_of_rides)) %>%
  head(20)

# Horizontal bar plot for customers
ggplot(top_stations_customers, aes(x = number_of_rides, y = reorder(start_station_name, -number_of_rides
  geom_bar(stat = "identity", fill = "lightcoral") +
  labs(title = "Top 20 Stations by Number of Rides for Customers",
       x = "Number of Rides",
       y = "Station Name") +
```

```
  theme(axis.text.y = element_text(hjust = 0, vjust = 0.5))
```

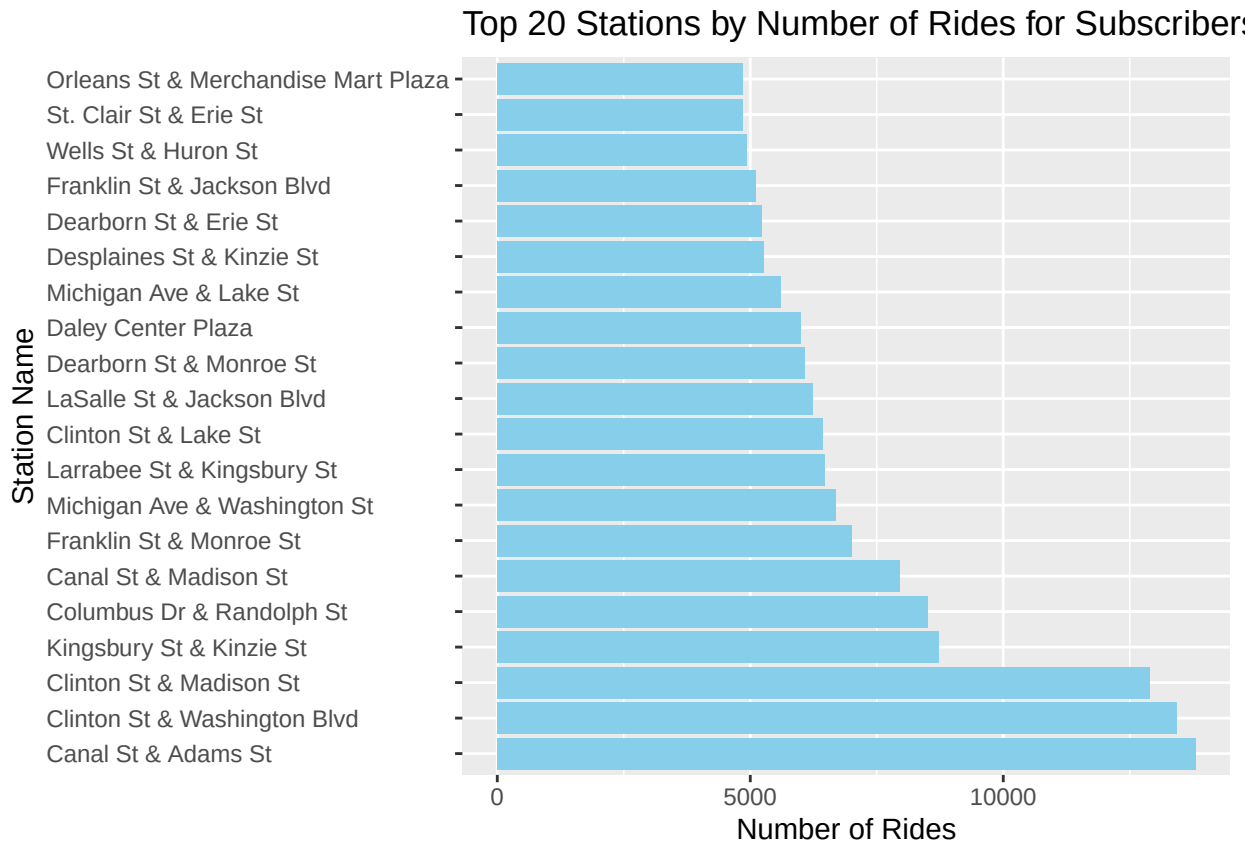


Top 20 Stations by number of rides booked for subscribers

```
#I filtered and summarized data to identify the top 20 bike share stations with the highest number of s

top_stations_subscribers <- bike_share %>%
  filter(!is.na(start_station_name)) %>%
  filter(usertype == "Subscriber") %>%
  group_by(start_station_name) %>%
  summarise(number_of_rides = n(),
            avg_ride_length = mean(ride_length),
            avg_ride_length_min = mean(ride_length) / 60) %>%
  arrange(desc(number_of_rides)) %>%
  head(20)

# Horizontal bar plot for subscribers
ggplot(top_stations_subscribers, aes(x = number_of_rides, y = reorder(start_station_name, -number_of_ri
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Top 20 Stations by Number of Rides for Subscribers",
       x = "Number of Rides",
       y = "Station Name") +
  theme(axis.text.y = element_text(hjust = 0, vjust = 0.5))
```

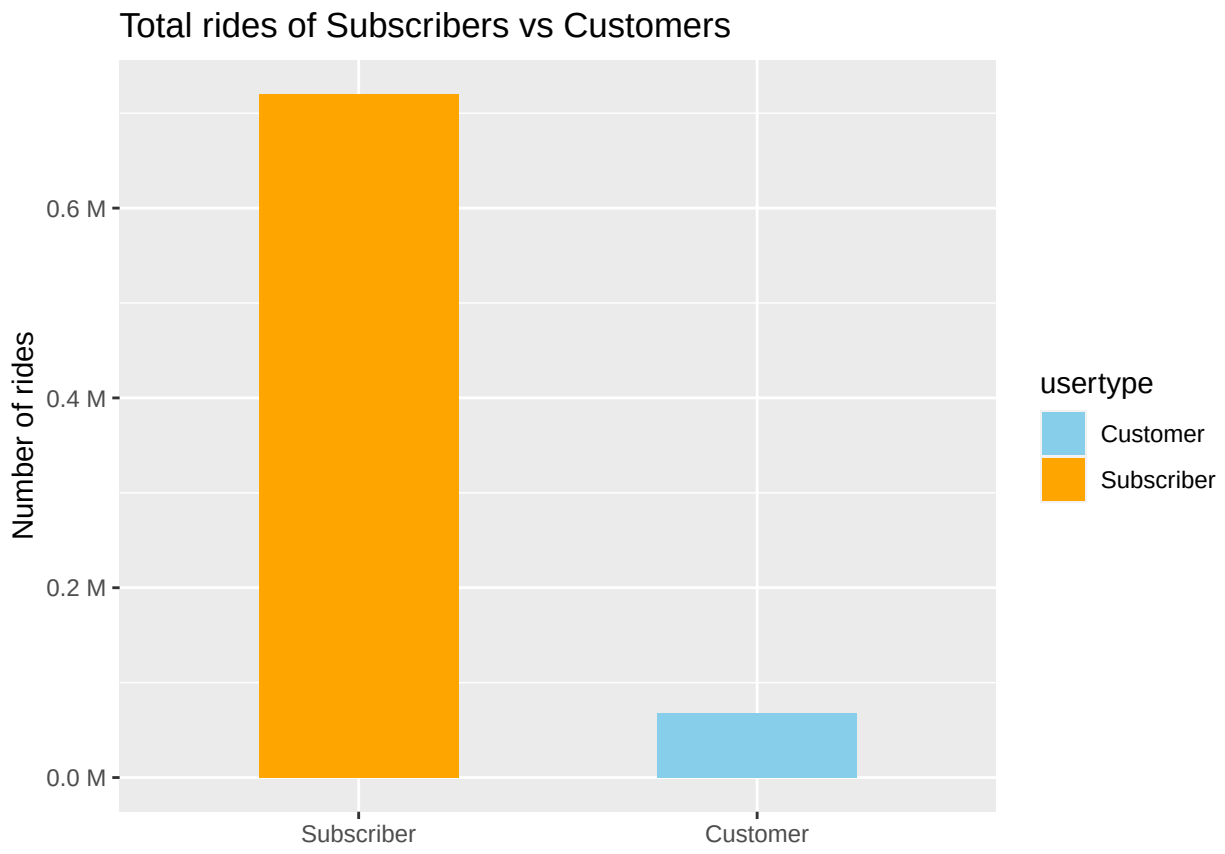


#Total rides of Subscribers vs Customer

```
#I created a bar plot that shows the total number of rides for subscribers and customers.

library(forcats)

ggplot(bike_share, aes(x = fct_infreq(usertype), fill = usertype)) +
  geom_bar(width = 0.5) +
  labs(x = NULL, y = "Number of rides", title = "Total rides of Subscribers vs Customers") +
  scale_y_continuous(labels = scales::unit_format(unit = "M", scale = 1e-6)) +
  scale_fill_manual(values = c("skyblue", "orange"))
```



Subscribers have significantly more rides than customers, as shown by the plot. This suggests that subscribers are more frequent riders, while cutomers use the service less frequently.