Untitled

Fatih CAKSEN

2024-02-04

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.4.4 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

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 21742451 21742452 ...  
## $ start\_time : chr "2019-01-01 0:04:37" "2019-01-01 0:08:13" "2019-01-01 0:13:23" "2019-01-01 0:13:45" ...  
## $ end\_time : chr "2019-01-01 0:11:07" "2019-01-01 0:15:34" "2019-01-01 0:27:12" "2019-01-01 0:43:28" ...  
## $ 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" "California Ave & Milwaukee Ave" ...  
## $ 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 Ave & Fillmore St (\*)" "Clark St & Elm St" ...  
## $ 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" "2019-01-01 0:13:45" ...  
## $ end\_time : chr "2019-01-01 0:11:07" "2019-01-01 0:15:34" "2019-01-01 0:27:12" "2019-01-01 0:43:28" ...  
## $ 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" "California Ave & Milwaukee Ave" ...  
## $ 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 Ave & Fillmore St (\*)" "Clark St & Elm St" ...  
## $ usertype : chr "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...

str(divvy\_2020)

## 'data.frame': 426887 obs. of 16 variables:  
## $ trip\_id : chr "EACB19130B0CDA4A" "8FED874C809DC021" "789F3C21E472CA96" "C9A388DAC6ABF313" ...  
## $ 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" "2020-01-06 16:17:07" ...  
## $ end\_time : chr "2020-01-21 20:14:30" "2020-01-30 14:26:22" "2020-01-09 19:32:17" "2020-01-06 16:25:56" ...  
## $ start\_station\_name: chr "Western Ave & Leland Ave" "Clark St & Montrose Ave" "Broadway & Belmont Ave" "Clark St & Randolph St" ...  
## $ 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 & Belmont Ave" "Fairbanks Ct & Grand Ave" ...  
## $ 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\_week))

# 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" "2019-01-01 0:13:45" ...  
## $ end\_time : chr "2019-01-01 0:11:07" "2019-01-01 0:15:34" "2019-01-01 0:27:12" "2019-01-01 0:43:28" ...  
## $ 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" "California Ave & Milwaukee Ave" ...  
## $ 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 Ave & Fillmore St (\*)" "Clark St & Elm St" ...  
## $ 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

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" "2019-01-01 0:13:45" ...  
## $ end\_time : chr "2019-01-01 0:11:07" "2019-01-01 0:15:34" "2019-01-01 0:27:12" "2019-01-01 0:43:28" ...  
## $ 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" "California Ave & Milwaukee Ave" ...  
## $ 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 Ave & Fillmore St (\*)" "Clark St & Elm St" ...  
## $ usertype : chr "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...

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

# 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" "2019-01-01 0:13:45" ...  
## $ end\_time : chr "2019-01-01 0:11:07" "2019-01-01 0:15:34" "2019-01-01 0:27:12" "2019-01-01 0:43:28" ...  
## $ 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" "California Ave & Milwaukee Ave" ...  
## $ 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 Ave & Fillmore St (\*)" "Clark St & Elm St" ...  
## $ 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 = "secs"))  
  
# 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" "California Ave & Milwaukee Ave" ...  
## $ 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 Ave & Fillmore St (\*)" "Clark St & Elm St" ...  
## $ 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" "California Ave & Milwaukee Ave" ...  
## $ 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 Ave & Fillmore St (\*)" "Clark St & Elm St" ...  
## $ 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", "Wednesday", "Thursday", "Friday", "Saturday"))

# 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.

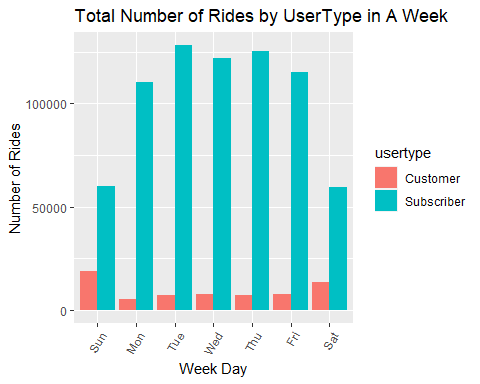
## # A tibble: 14 × 4  
## # Groups: usertype [2]  
## usertype weekday number\_of\_rides average\_duration  
## <chr> <ord> <int> <dbl>  
## 1 Customer Sun 18652 5061.  
## 2 Customer Mon 5591 4752.  
## 3 Customer Tue 7311 4562.  
## 4 Customer Wed 7690 4480.  
## 5 Customer Thu 7147 8452.  
## 6 Customer 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.

# When creating a ggplot for numerical distribution, it is important to ensure that the entire number is displayed. To achieve this, scientific notation must be turned off. By doing so, you can provide your audience with a clear and accurate representation of your data.  
options(scipen=999)

# STEP 5: Visualizations

# This code calculates the total number of rides and average ride duration for each user type and day of the week. Then it creates a grouped bar plot with ggplot to display the total number of rides by user type on different days 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) %>%  
 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))

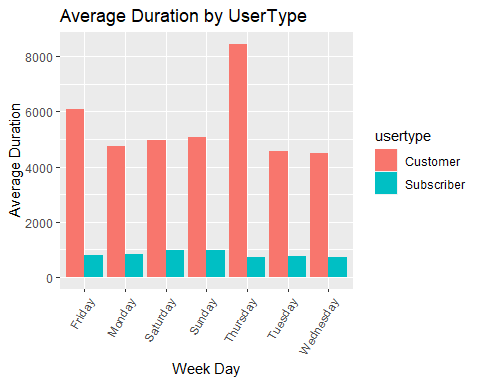
## `summarise()` has grouped output by 'usertype'. You can override using the  
## `.groups` argument.



# 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 grouped bar plot using ggplot to visualize the results.  
  
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))

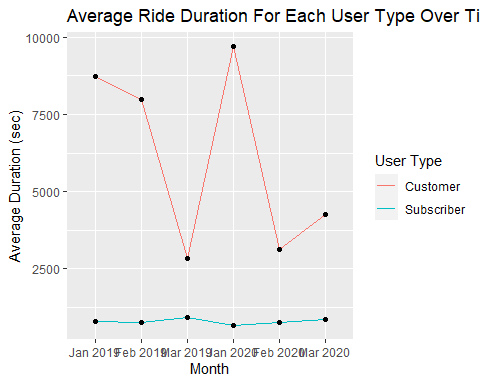
## `summarise()` has grouped output by 'usertype'. You can override using the  
## `.groups` argument.



# 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 over time using a line plot created with ggplot.   
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"))

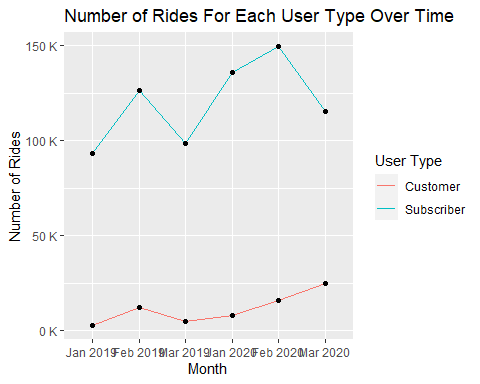
## `summarise()` has grouped output by 'usertype'. You can override using the  
## `.groups` argument.



#Bike rides 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.

#I calculated the total rides for each user type and month, and plots a line graph using ggplot to visualize changes over time.  
  
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"))

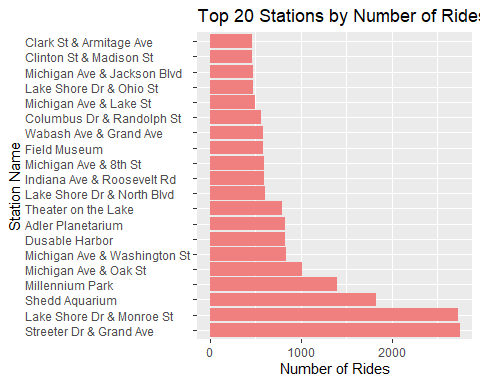
## `summarise()` has grouped output by 'usertype'. You can override using the  
## `.groups` argument.



# 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.

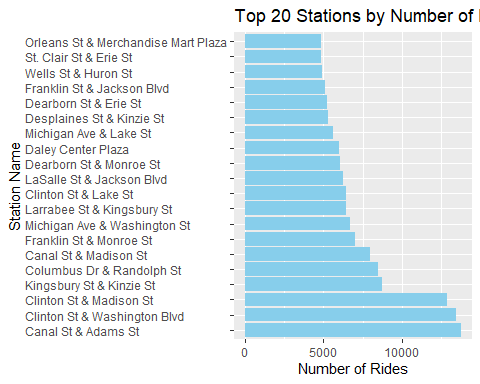
# Top 20 Station Name booked for the Customers

# I filtered and summarized data to identify the top 20 bike share stations with the highest number of customers rides. Then, I created a horizontal bar plot to visualize this information.  
  
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))



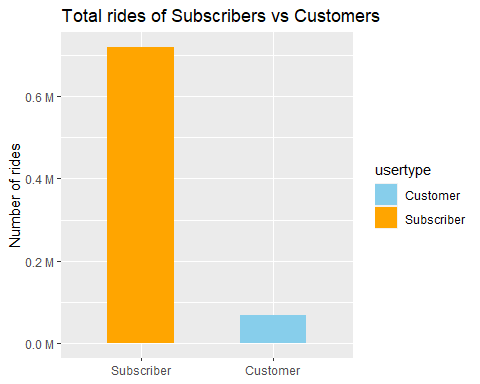
# Top 20 Stations by number of rides booked for subscribers

#I filtered and summarized data to identify the top 20 bike share stations with the highest number of subscriber rides. Then, I created a horizontal bar plot to visualize this information.  
  
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\_rides))) +  
 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.