Cyclistic Bike-Share Analysis Case Study

1. Ask

Moreno has set a clear goal: Design marketing strategies aimed at converting casual riders into annual members. In order to do that, however, the marketing analyst team needs to better understand how annual members and casual riders differ, why casual riders would buy a membership, and how digital media could affect their marketing tactics. Moreno and her team are interested in analyzing the Cyclistic historical bike trip data to identify trends.

These three questions will guide the future marketing program:

- How do annual members and casual riders use Cyclistic bikes differently?
- Why would casual riders buy Cyclistic annual memberships?
- How can Cyclistic use digital media to influence casual riders to become members?

Our first business task is to identify how annual members and casual riders use Cyclistic bikes differently. Understanding how these two different customer profiles interact with the company's offerings is key to designing effective marketing strategies that convert casual riders to annual memberships while using data to drive these decisions and fuel the company's future growth.

2. Prepare

Data Source:

This analysis uses primary source data downloaded from Cyclistic's internal data. This dataset is made available by divvy open source data. It is licensed under Motivate International Inc. This dataset consists of primary data collected from bike trackers and is all in quantitative measurements. We identify our users from their membership status and ride id. No other personal data was collected from users. For biking records, we have data on geospatial location, length of each ride, and bike type. With the available data, we will be able to examine the difference between two user types based on existing correlations but we will not be able to identify the real reasons behind trends.

Data organization:

The data used for this analysis covers the trips during the 12-month period,spanning between[January-2021 to December-2021] a set of .CSV format files organized by months and years. It comes with 13 columns, including : 1.ride_id: Unique ride id for each trip record 2.rideable_type: Type of bikes used by bikers- docked, electric and classic bikes 3.started_at: start DateTime of the ride 4.ended_at: end DateTime of the ride 5.start_station_name: Name of the station the ride started 6.start_station_id: ID of the station the ride started 7.end_station_name: Name of the station the ride ended 8.end_station_id: ID of the station the ride ended 9.start_lat: Latitude of where the ride started 10.start_lng: Longitude of where the ride started 11.end_lat: Latitude of where the ride ended 12.end_lng: Longitude of where the ride ended 13.member_casual: Type of bikers- casual riders or members.

Data Credibility Using ROCC Method:

Reliable: dataset appears to have many missing values in non-primary key columns and inconsistent station names. However, this deficiency is countered by the comprehensive latitude and longitude data. Dataset is also weak in data validation on the start and end time of bike rides, having some end times earlier than start times, but incorrect records only make up less than .01% of the whole dataset, so this isn't a significant issue with reliability. Original: data source is primary Comprehensive: dataset consists of more than 5 million rows of complete data Current: using up-to-date datasets that reflect the last 12 months of cycling data Vetted: recorded bike trips are based on real records from a primary source from an actual bike-share company*

Data acquisition process

The following process is utilized:

Each dataset is downloaded Appropriately stored in a folder for original datasets

Identifying

In order to identify issues with the data, we: - Evaluate the ride length and spot unusual observations - Filtered the data and identified missing values, duplicates… - Sorted the data and found inconsistent attribute format

3. Process

Data Cleaning - Excel: Remove negative values in column "ride_length". Since we are measuring the duration of a bike ride, our result from ended_at - started_at should not be a negative value. A negative value may occur if a bike ride's start or end time is recorded incorrectly. Sort "ride_length" in ASC order.

No negative data found.

Remove duplicate data in column "ride_id". Each record in this column represents an identifier for each unique bike ride. No bike ride should have repeat values. Check ride_id for duplicates using conditional formatting Remove duplicate values from the ride_id column by selecting the "ride_id" column, using "Remove Duplicates" in the Data tab, choose "continue with the current selection," and then click "Remove Duplicates".

No duplicates found in datasets.

```r
#install.packages("tidyverse")  #tidyverse includes core packages like
ggplot2 and readr which are helpful to wrangle data

#install.packages("lubridate")  #helps wrangle date attributes
```

```
#install.packages("ggplot2")  #helps visualize data
#install.packages("dplyr")

library(tidyverse)  #tidyverse includes core packages like ggplot2 and readr
which are helpful to wrangle data

## ── Attaching core tidyverse packages ────────────────────────
tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── 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)  #helps wrangle date attributes

library(ggplot2)  #helps visualize data

library(dplyr)

#install.packages("readxl")

library("readxl")
# xls files
data_Q1 <-
read_excel("C:/Users/dorra/OneDrive/Documents/Data/Divvy_Trips_2019_Q1.xlsx")
data_Q2 <-
read_excel("C:/Users/dorra/OneDrive/Documents/Data/Divvy_Trips_2019_Q2.xlsx")
data_Q3 <-
read_excel("C:/Users/dorra/OneDrive/Documents/Data/Divvy_Trips_2019_Q3.xlsx")

## Warning in read_fun(path = path, sheet_i = sheet, limits = limits, shim =
shim,
## : NA inserted for impossible 1900-02-29 datetime
```

Compare column names each of the files While the names don’t have to be in the same order, they DO need to match perfectly before we can use a command to join them into one file

```
colnames(data_Q1)

##  [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"
## [13] "ride_length"       "day_of_week"
```

```
colnames(data_Q2)
```

```
##  [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"
## [13] "ride_length"       "day_of_week"
```

```
colnames(data_Q3)
```

```
##  [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"
## [13] "ride_length"       "day_of_week"
```

All columns have the same name

```
str(data_Q1)
```

```
## tibble [365,069 × 14] (S3: tbl_df/tbl/data.frame)
##  $ trip_id          : num [1:365069] 21742443 21742444 21742445 21742446
21742447 ...
##  $ start_time       : POSIXct[1:365069], format: "2019-01-01 00:04:37"
"2019-01-01 00:08:13" ...
##  $ end_time         : POSIXct[1:365069], format: "2019-01-01 00:11:07"
"2019-01-01 00:15:34" ...
##  $ bikeid           : num [1:365069] 2167 4386 1524 252 1170 ...
##  $ tripduration     : num [1:365069] 390 441 829 1783 364 ...
##  $ from_station_id  : num [1:365069] 199 44 15 123 173 98 98 211 150 268
...
##  $ from_station_name: chr [1:365069] "Wabash Ave & Grand Ave" "State St &
Randolph St" "Racine Ave & 18th St" "California Ave & Milwaukee Ave" ...
##  $ to_station_id    : num [1:365069] 84 624 644 176 35 49 49 142 148 141
...
##  $ to_station_name  : chr [1:365069] "Milwaukee Ave & Grand Ave" "Dearborn
St & Van Buren St (*)" "Western Ave & Fillmore St (*)" "Clark St & Elm St"
...
##  $ usertype         : chr [1:365069] "Subscriber" "Subscriber"
"Subscriber" "Subscriber" ...
##  $ gender           : chr [1:365069] "Male" "Female" "Female" "Male" ...
##  $ birthyear        : num [1:365069] 1989 1990 1994 1993 1994 ...
##  $ ride_length      : POSIXct[1:365069], format: "1899-12-31 00:06:30"
"1899-12-31 00:07:21" ...
##  $ day_of_week      : num [1:365069] 3 3 3 3 3 3 3 3 3 3 ...
```

```
str(data_Q2)
```

```
## tibble [1,048,575 × 14] (S3: tbl_df/tbl/data.frame)
##  $ trip_id          : num [1:1048575] 22178529 22178530 22178531 22178532
22178533 ...
##  $ start_time       : POSIXct[1:1048575], format: "2019-04-01 00:02:22"
"2019-04-01 00:03:02" ...
##  $ end_time         : POSIXct[1:1048575], format: "2019-04-01 00:09:48"
"2019-04-01 00:20:30" ...
##  $ bikeid           : num [1:1048575] 6251 6226 5649 4151 3270 ...
##  $ tripduration     : num [1:1048575] 446 1048 252 357 1007 ...
##  $ from_station_id  : num [1:1048575] 81 317 283 26 202 420 503 260 211
211 ...
##  $ from_station_name: chr [1:1048575] "Daley Center Plaza" "Wood St &
Taylor St" "LaSalle St & Jackson Blvd" "McClurg Ct & Illinois St" ...
##  $ to_station_id    : num [1:1048575] 56 59 174 133 129 426 500 499 211
211 ...
##  $ to_station_name  : chr [1:1048575] "Desplaines St & Kinzie St" "Wabash
Ave & Roosevelt Rd" "Canal St & Madison St" "Kingsbury St & Kinzie St" ...
##  $ usertype         : chr [1:1048575] "Subscriber" "Subscriber"
"Subscriber" "Subscriber" ...
##  $ gender           : chr [1:1048575] "Male" "Female" "Male" "Male" ...
##  $ birthyear        : num [1:1048575] 1975 1984 1990 1993 1992 ...
##  $ ride_length      : POSIXct[1:1048575], format: "1899-12-31 00:07:26"
"1899-12-31 00:17:28" ...
##  $ day_of_week      : num [1:1048575] 2 2 2 2 2 2 2 2 2 2 ...
```

```
str(data_Q3)
```

```
## tibble [1,048,575 × 14] (S3: tbl_df/tbl/data.frame)
##  $ trip_id          : num [1:1048575] 23479388 23479389 23479390 23479391
23479392 ...
##  $ start_time       : POSIXct[1:1048575], format: "2019-07-01 00:00:27"
"2019-07-01 00:01:16" ...
##  $ end_time         : POSIXct[1:1048575], format: "2019-07-01 00:20:41"
"2019-07-01 00:18:44" ...
##  $ bikeid           : num [1:1048575] 3591 5353 6180 5540 6014 ...
##  $ tripduration     : num [1:1048575] 1214 1048 1554 1503 1213 ...
##  $ from_station_id  : num [1:1048575] 117 381 313 313 168 300 168 313 43
43 ...
##  $ from_station_name: chr [1:1048575] "Wilton Ave & Belmont Ave" "Western
Ave & Monroe St" "Lakeview Ave & Fullerton Pkwy" "Lakeview Ave & Fullerton
Pkwy" ...
##  $ to_station_id    : num [1:1048575] 497 203 144 144 62 232 62 144 195
195 ...
##  $ to_station_name  : chr [1:1048575] "Kimball Ave & Belmont Ave" "Western
Ave & 21st St" "Larrabee St & Webster Ave" "Larrabee St & Webster Ave" ...
##  $ usertype         : chr [1:1048575] "Subscriber" "Customer" "Customer"
"Customer" ...
##  $ gender           : chr [1:1048575] "Male" NA NA NA ...
##  $ birthyear        : num [1:1048575] 1992 NA NA NA NA ...
##  $ ride_length      : POSIXct[1:1048575], format: "1899-12-31 00:20:14"
```

```
"1899-12-31 00:17:28" ...
##  $ day_of_week      : num [1:1048575] 2 2 2 2 2 2 2 2 2 2 ...
```

Convert trip_id and bikeid to character so that they can stack correctly

```
data_Q1 <-  mutate(data_Q1, trip_id = as.character(trip_id)
                   ,bikeid = as.character(bikeid))
data_Q2 <-  mutate(data_Q2, trip_id = as.character(trip_id)
                   ,bikeid = as.character(bikeid))
data_Q3 <-  mutate(data_Q3, trip_id = as.character(trip_id)
                   ,bikeid = as.character(bikeid))
```

Combining dataframes into one big dataframe Stack individual quarter's data frames into one big data frame

```
all_trips <- bind_rows(data_Q1, data_Q2, data_Q3)

# Inspect the new table that has been created

colnames(all_trips)  #List of column names

##  [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"
## [13] "ride_length"       "day_of_week"

nrow(all_trips)  #How many rows are in data frame?

## [1] 2462219

dim(all_trips)  #Dimensions of the data frame?

## [1] 2462219      14

head(all_trips)  #See the first 6 rows of data frame.  Also tail(qs_raw)

## # A tibble: 6 × 14
##   trip_id  start_time          end_time            bikeid tripduration
##   <chr>    <dttm>              <dttm>              <chr>         <dbl>
## 1 21742443 2019-01-01 00:04:37 2019-01-01 00:11:07 2167            390
## 2 21742444 2019-01-01 00:08:13 2019-01-01 00:15:34 4386            441
## 3 21742445 2019-01-01 00:13:23 2019-01-01 00:27:12 1524            829
## 4 21742446 2019-01-01 00:13:45 2019-01-01 00:43:28 252            1783
## 5 21742447 2019-01-01 00:14:52 2019-01-01 00:20:56 1170            364
## 6 21742448 2019-01-01 00:15:33 2019-01-01 00:19:09 2437            216
## # ℹ 9 more variables: from_station_id <dbl>, from_station_name <chr>,
## #   to_station_id <dbl>, to_station_name <chr>, usertype <chr>, gender
<chr>,
## #   birthyear <dbl>, ride_length <dttm>, day_of_week <dbl>

str(all_trips)  #See list of columns and data types (numeric, character, etc)
```

```
## tibble [2,462,219 × 14] (S3: tbl_df/tbl/data.frame)
##  $ trip_id          : chr [1:2462219] "21742443" "21742444" "21742445"
"21742446" ...
##  $ start_time       : POSIXct[1:2462219], format: "2019-01-01 00:04:37"
"2019-01-01 00:08:13" ...
##  $ end_time         : POSIXct[1:2462219], format: "2019-01-01 00:11:07"
"2019-01-01 00:15:34" ...
##  $ bikeid           : chr [1:2462219] "2167" "4386" "1524" "252" ...
##  $ tripduration     : num [1:2462219] 390 441 829 1783 364 ...
##  $ from_station_id  : num [1:2462219] 199 44 15 123 173 98 98 211 150 268
...
##  $ from_station_name: chr [1:2462219] "Wabash Ave & Grand Ave" "State St &
Randolph St" "Racine Ave & 18th St" "California Ave & Milwaukee Ave" ...
##  $ to_station_id    : num [1:2462219] 84 624 644 176 35 49 49 142 148 141
...
##  $ to_station_name  : chr [1:2462219] "Milwaukee Ave & Grand Ave"
"Dearborn St & Van Buren St (*)" "Western Ave & Fillmore St (*)" "Clark St &
Elm St" ...
##  $ usertype         : chr [1:2462219] "Subscriber" "Subscriber"
"Subscriber" "Subscriber" ...
##  $ gender           : chr [1:2462219] "Male" "Female" "Female" "Male" ...
##  $ birthyear        : num [1:2462219] 1989 1990 1994 1993 1994 ...
##  $ ride_length      : POSIXct[1:2462219], format: "1899-12-31 00:06:30"
"1899-12-31 00:07:21" ...
##  $ day_of_week      : num [1:2462219] 3 3 3 3 3 3 3 3 3 3 ...
```

Statistical summary of data.

Mainly for numeric

```
summary(all_trips)
```

```
##    trip_id            start_time
##  Length:2462219     Min.   :2019-01-01 00:04:37.00
##  Class :character   1st Qu.:2019-04-28 12:24:23.00
##  Mode  :character   Median :2019-06-16 14:01:48.00
##                     Mean   :2019-06-06 07:32:31.89
##                     3rd Qu.:2019-07-25 16:52:09.00
##                     Max.   :2019-08-26 17:18:28.00
##
##     end_time                           bikeid            tripduration
##  Min.   :2019-01-01 00:11:07.00   Length:2462219     Min.   :      61
##  1st Qu.:2019-04-28 12:47:19.50   Class :character   1st Qu.:     424
##  Median :2019-06-16 14:28:18.00   Mode  :character   Median :     738
##  Mean   :2019-06-06 07:57:44.78                      Mean   :    1513
##  3rd Qu.:2019-07-25 17:11:18.00                      3rd Qu.:    1343
##  Max.   :2019-10-30 14:05:17.00                      Max.   :10628400
##
##  from_station_id from_station_name  to_station_id   to_station_name
##  Min.   :  1.0   Length:2462219     Min.   :  1.0   Length:2462219
##  1st Qu.: 77.0   Class :character   1st Qu.: 77.0   Class :character
##  Median :174.0   Mode  :character   Median :174.0   Mode  :character
##  Mean   :200.6                      Mean   :201.6
##  3rd Qu.:289.0                      3rd Qu.:289.0
##  Max.   :673.0                      Max.   :673.0
##
##    usertype            gender            birthyear
##  Length:2462219     Length:2462219     Min.   :1759
##  Class :character   Class :character   1st Qu.:1979
##  Mode  :character   Mode  :character   Median :1987
##                                        Mean   :1984
##                                        3rd Qu.:1992
##                                        Max.   :2014
##                                        NA's   :372398
##   ride_length                        day_of_week
##  Min.   :1899-12-31 00:01:01.00   Min.   :1.000
##  1st Qu.:1899-12-31 00:07:04.00   1st Qu.:2.000
##  Median :1899-12-31 00:12:18.00   Median :4.000
##  Mean   :1899-12-31 00:25:10.78   Mean   :4.071
##  3rd Qu.:1899-12-31 00:22:23.00   3rd Qu.:6.000
##  Max.   :1900-05-02 01:20:22.00   Max.   :7.000
##  NA's   :1
```

```
distinct_trip_count <- all_trips %>%
  summarise(distinct_trips = n_distinct(trip_id))
distinct_trip_count
```

```
## # A tibble: 1 × 1
##   distinct_trips
##            <int>
## 1        2462219
```

There are no duplications.

```
unique_usertypes <- unique(all_trips$usertype)

print(unique_usertypes)
```

```
## [1] "Subscriber" "Customer"
```

We only have two terms for user type which is referring to casual and member

```
unique_day <- unique(all_trips$day_of_week)

print(unique_day)
```

```
## [1] 3 4 6 5 7 1 2
```

```
all_trips <- all_trips %>%
  mutate(
    # Convert ride_length to actual duration in minutes
    ride_length = hour(ride_length) * 60 + minute(ride_length) +
second(ride_length) / 60
  )

negative_rides <- all_trips %>%
  mutate(duration = as.numeric(difftime(end_time, start_time, units =
"secs"))) %>%
  filter(duration < 0)

# Count of negative durations
negative_count <- nrow(negative_rides)

# Print results
print(paste("Number of rides with negative duration:", negative_count))

## [1] "Number of rides with negative duration: 0"
```

There are no issues with time logs as well.

4.Analyze

Descriptive analysis

```
summary(all_trips$ride_length)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
##    0.5833    7.0667   12.3000   20.0774   22.3833 1439.7500         1
```

The shortest ride was 0.58 minutes (34 seconds) and the longest ride was 1439.75 minutes (almost 24 hours)

```
# Compare members and casual users
aggregate(all_trips$ride_length ~ all_trips$usertype, FUN = mean)

##   all_trips$usertype all_trips$ride_length
## 1           Customer              41.19733
## 2         Subscriber              13.36050
```

Casuals take the bike for a longer time then members.

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

##   all_trips$usertype all_trips$ride_length
## 1           Customer              26.80000
## 2         Subscriber              10.03333

aggregate(all_trips$ride_length ~ all_trips$usertype, FUN = max)

##   all_trips$usertype all_trips$ride_length
## 1           Customer               1439.15
## 2         Subscriber               1439.75

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

```
##   all_trips$usertype all_trips$ride_length
## 1           Customer             0.5833333
## 2         Subscriber             1.0166667
```

```
# Now, let's run the average ride time by each day for members vs casual
users
aggregate(all_trips$ride_length ~ all_trips$usertype + all_trips$day_of_week,
FUN = mean)
```

```
##    all_trips$usertype all_trips$day_of_week all_trips$ride_length
## 1            Customer                     1              42.90539
## 2          Subscriber                     1              15.11391
## 3            Customer                     2              42.68424
## 4          Subscriber                     2              12.84344
## 5            Customer                     3              40.05213
## 6          Subscriber                     3              12.92618
## 7            Customer                     4              38.90456
## 8          Subscriber                     4              13.00047
## 9            Customer                     5              39.59526
## 10         Subscriber                     5              13.05684
## 11           Customer                     6              40.04038
## 12         Subscriber                     6              12.98190
## 13           Customer                     7              42.11678
## 14         Subscriber                     7              15.09948
```

noting that 1 = Sunday and 7 = Saturday.

Casuals have longer rides on Sunday and Monday.

```
# analyze ridership data by type and weekday
all_trips %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%  #creates weekday
field using wday()
  group_by(usertype, weekday) %>%  #groups by usertype and weekday
  summarise(number_of_rides = n()                          #calculates the
number of rides and average duration
          ,average_duration = mean(ride_length)) %>%      # calculates the
average duration
  arrange(usertype, weekday)
```

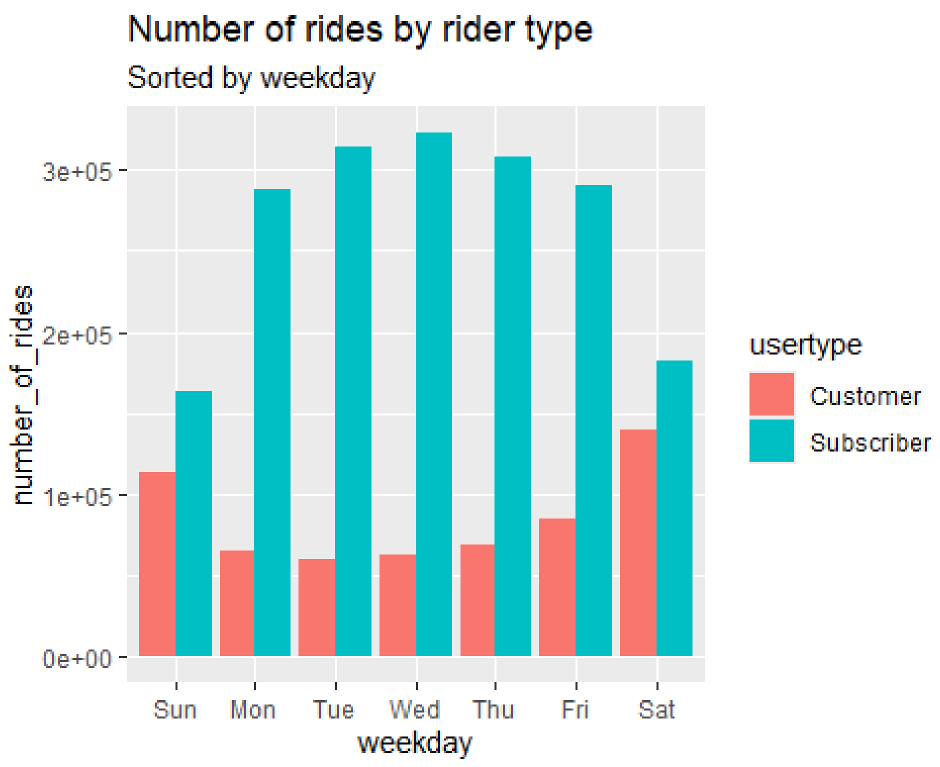
```
## `summarise()` has grouped output by 'usertype'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 14 × 4
## # Groups:   usertype [2]
##    usertype   weekday number_of_rides average_duration
##    <chr>      <ord>             <int>            <dbl>
##  1 Customer   Sun              113939             42.7
##  2 Customer   Mon               64654             40.7
##  3 Customer   Tue               60219             39.7
##  4 Customer   Wed               62799             38.9
##  5 Customer   Thu              68676             39.9
##  6 Customer   Fri              84533             40.5
##  7 Customer   Sat             139304             43.0
##  8 Subscriber Sun             163440             14.9
##  9 Subscriber Mon             287738             13.0
## 10 Subscriber Tue             313601             13.0
## 11 Subscriber Wed             322483             13.0
## 12 Subscriber Thu             308106             NA
## 13 Subscriber Fri             290346             13.0
## 14 Subscriber Sat             182381             15.1
```

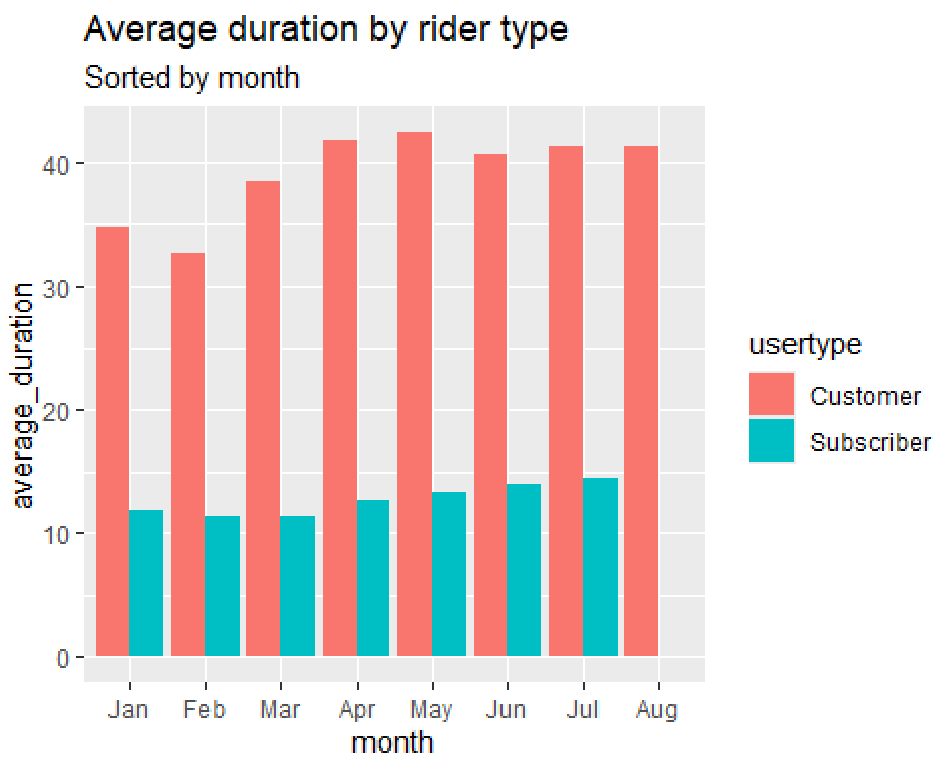
5. Share

Let’s visualize the number of rides by rider type

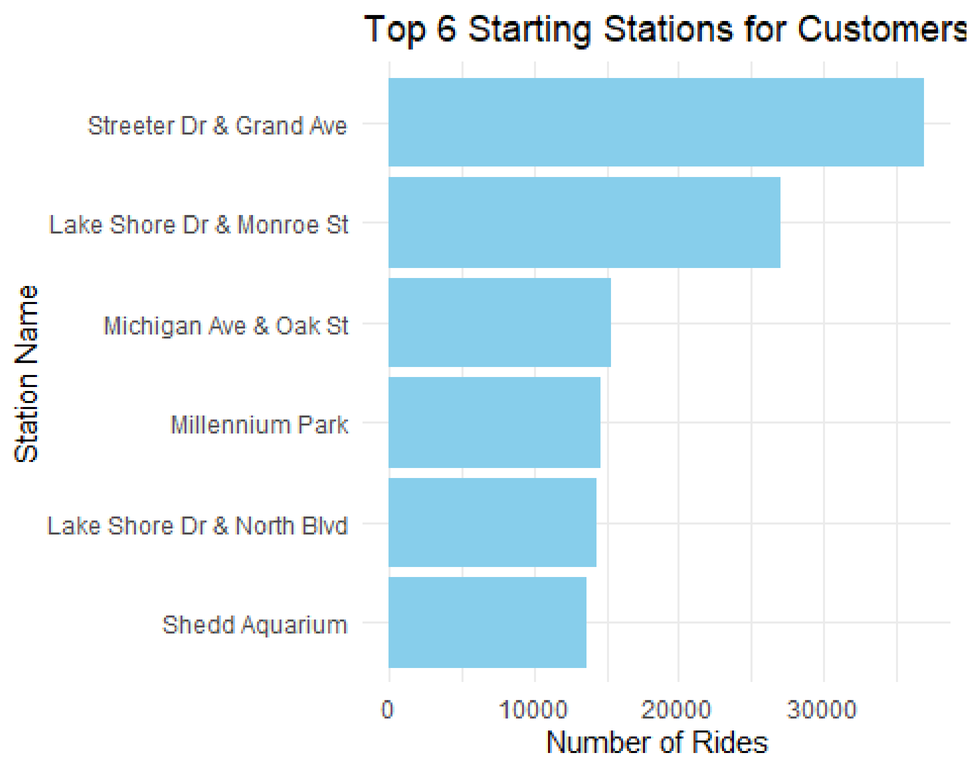
```
# Let's visualize the number of rides by rider type
library(ggplot2)
all_trips %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%
  group_by(usertype, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(usertype, weekday)  %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = usertype)) +
  geom_col(position = "dodge") +
  labs(
      title = "Number of rides by rider type",
      subtitle = "Sorted by weekday"
      )

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



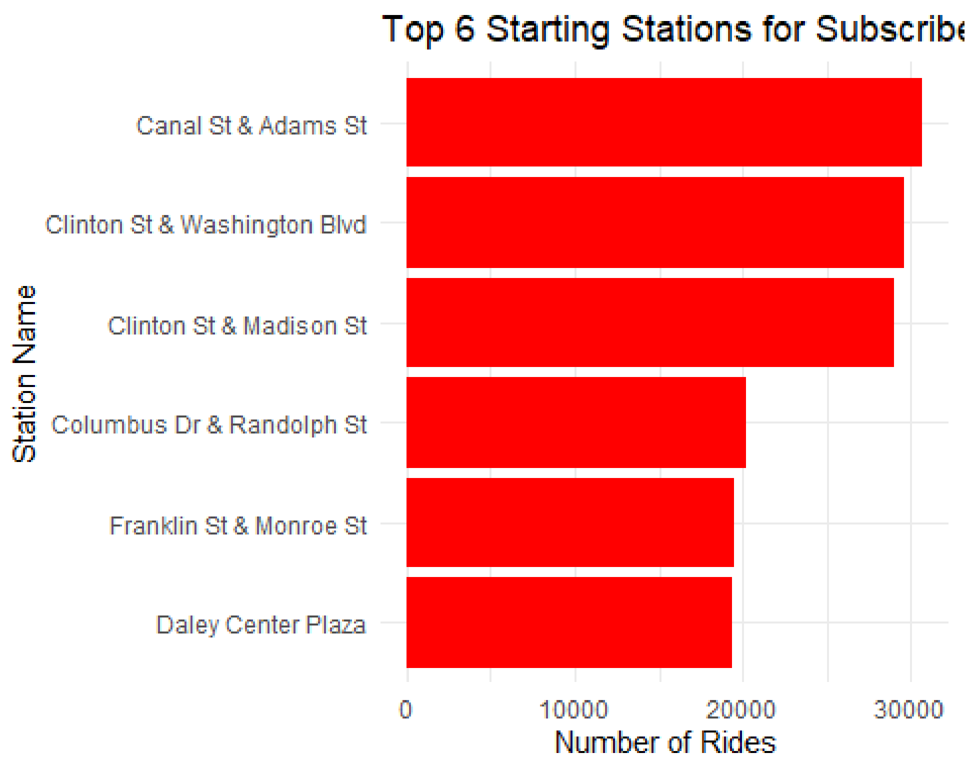
```
all_trips <- all_trips %>%
  mutate(month = month(start_time, label = TRUE))

# Let's create a visualization for average duration
all_trips %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%
  group_by(usertype, month) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(usertype, month)  %>%
  ggplot(aes(x = month, y = average_duration, fill = usertype)) +
  geom_col(position = "dodge") +
  labs(title = "Average duration by rider type",
       subtitle = "Sorted by month")

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

## Warning: Removed 1 row containing missing values or values outside the
scale range
## (`geom_col()`).
```



```r
# Filter and count the stations
top_stations <- all_trips %>%
  filter(usertype == "Customer") %>%
  count(from_station_name, sort = TRUE) %>%
  top_n(6, n)

# Create the histogram
ggplot(top_stations, aes(x = reorder(from_station_name, n), y = n)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  coord_flip() +  # This makes it a horizontal bar chart
  labs(title = "Top 6 Starting Stations for Customers",
       x = "Station Name",
       y = "Number of Rides") +
  theme_minimal() +
  theme(axis.text.y = element_text(angle = 0, hjust = 1))
```



```r
# Filter and count the stations
top_stations <- all_trips %>%
  filter(usertype == "Subscriber") %>%
  count(from_station_name, sort = TRUE) %>%
  top_n(6, n)

# Create the histogram
ggplot(top_stations, aes(x = reorder(from_station_name, n), y = n)) +
  geom_bar(stat = "identity", fill = "red") +
  coord_flip() +  # This makes it a horizontal bar chart
  labs(title = "Top 6 Starting Stations for Subscribers",
       x = "Station Name",
       y = "Number of Rides") +
  theme_minimal() +
  theme(axis.text.y = element_text(angle = 0, hjust = 1))
```



We can clearly see the casual rides are mostly located around the center of the town (or the bay area), with all their trips located around that area points towards their bike usage pattern, which is for leisure, probably tourist or sightseeing related rides.

Members mostly use bikes all over the city including the main city area and outside the main center. This can be hypothesized as they travel for work purposes.

KEY TAKEAWAYS

- Members hold the biggest proportion of the total rides, ~10% bigger than casual riders.
- In all months we have more members than casual riders.
- For casual riders the biggest volume of data is on the weekend.
- There is a bigger volume of bikers in the afternoon.
- It could be possible that members use bikes for work purposes, this information can be backed by their bike usage in colder months, where there is significant drop in casual members in those months.

Now for how members differs from casuals: - Members have the bigger volume of data, except on saturday and sunday. On the weekend, casuals riders have the most data points. - Casuals riders have more ride length (ride duration) than members. Average ride time of member are mostly same slight increase in end of week. - We have more members during the morning, mainly between 7am and 10am. And more casuals between 3pm and 12am. - Members have a bigger preference for classic bikes, followed by electric bike. - Members have a more fixed use for bikes for routine activities. Whereas casual rider’s usage is different, mostly all activities are on the weekend. - Casual members spend time near the center of the city or the bay area, whereas members are scattered throughout the city.

6. Act

Casual riders spent more time on bikes and member riders spent less time biking than casual riders.

On weekends, the number of casual bike riders increases. The most popular day for bike renting for all subscribers was Saturday. Sunday was the second most popular day for casual users.

It seems like casual riders use bikes for picnics and short trips on weekends and casual users use bikes for general purposes like for going to school.

The data clearly showed seasonal fluctuations in ridership of all users, both in the number of rides and the average trip duration. It peaked during the summer months and was at its lowest point from the late fall to early spring, especially among casual users.

Recommendations

- Company may give discounts on weekends if casual users take annual membership.Company may Offer weekend passes. This option may lead local users to purchase an annual membership.
- Company may give more discounts on long rides. Company should also offer seasonal passes as well. If the winter season pass is offered at a discount, some casual users might be willing to try it at a fraction of a membership cost before deciding to become year-long subscribers.
- Provide a greener and echo friendly option for riders by Offering discounts to businesses, It might also motivate more casual users to cycle to work more often or all the time.