How Does a Bike-Share Navigate Speedy Success?

Thacienne Uwimanayantumye

January 23, 2025

Project Description

Scenario

You are a junior data analyst working in the marketing analyst team at Cyclistic, a bike-share company in Chicago. The marketing director believes the company's future success depends on maximizing the number of annual memberships. Therefore, your team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, your team will design a new marketing strategy to convert casual riders into annual members. But first, Cyclistic executives must approve your recommendations, so they must be backed up with compelling data insights and professional data visualizations.

About the company

In 2016, Cyclistic launched a successful bike-share offering. Since then, the program has grown to a fleet of 5,824 bicycles that are tracked and locked into a network of 692 stations across Chicago. The bikes can be unlocked from one station and returned to any other station in the system at a time. Until now, Cyclistic's marketing strategy relied on building general awareness and appealing to broad consumer segments. One approach that helped make these things possible was the flexibility of its pricing plans: single-ride passes, full-day passes, and annual memberships. Customers who purchase single-ride or full-day passes are referred to as casual riders. Customers who purchase annual memberships are Cyclistic members.

Cyclistic's finance analysts have concluded that annual members are much more profitable than casual riders. Although the pricing flexibility helps Cyclistic attract more customers, Moreno believes that maximizing the number of annual members will be key to future growth. Rather than creating a marketing campaign targeting all-new customers, Moreno believes there is a good chance to convert casual riders into members. She notes that casual riders already know the Cyclistic program and have chosen Cyclistic for their mobility needs.

Moreno has set a clear goal: Design marketing strategies to convert 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 and why casual riders would buy a that convert digital media could affect their narcotics. Moreno and her team are interested in analyzing the Cyclistic historical bike trip data to identify trends.

Business Task.

In this project, we aim to answer the following question for the marketing manager (Moreno): How do annual members and casual riders use Cyclistic bikes differently? We use a raw data set provided by the company.

Let's start!

STEP 1: COLLECT DATA

To complete this project, we are going to use "R programming language" to consolidate downloaded Divvy data into a single data frame and then conduct a simple analysis to help answer the key question: "In what ways do members and casual riders use Divvy bikes differently?"
The data are downloaded from here and stored to local storage under the name "syclist.csv". Motivate International Inc. has made the data available under this licence. There are several data set, but for this project, we choose to use the first quarter of the year 2020. Since we already have the data, we can now prepare a cleaned dataset to be used for our analysis using Rstudio.

In R, it is a good practice to start by loading required libraries to complete the aimed tasks. In this project, we only need three libraries:

1. tidyverse for data import and wrangling.
2. lubridate for date functions
3. ggplot2 for data visualization

```
rm(list=ls()) # cleaning environment
library(tidyverse)  #helps wrangle data
```

```
## -- 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     1.0.2
## -- 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 errors
```

```
library(lubridate)  #helps wrangle date attributes
library(ggplot2)  #helps visualize data
```

Since we have the required libraries loaded, we can start uploading the Dvvy datasets to R. The data are provided into four different files each containing historical records for a quarter of a year, from the second quarter of year 2019 to the first quarter of the year 2019.

```
q2_2019 <- read_csv("Divvy_Trips_2019_Q2.csv",show_col_types = FALSE)
q3_2019 <- read_csv("Divvy_Trips_2019_Q3.csv",show_col_types = FALSE)
q4_2019 <- read_csv("Divvy_Trips_2019_Q4.csv",show_col_types = FALSE)
q1_2020 <- read_csv("Dvvy_Trips_2020_Q1.csv",show_col_types = FALSE)
```

STEP 2: WRANGLE DATA AND COMBINE INTO A SINGLE FILE

First, let's Compare the column names of each file. 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 combine them into one file.

```r
colnames(q3_2019)
colnames(q4_2019)
colnames(q2_2019)
colnames(q1_2020)
```

If you run the above code, you will see that the four datasets do not have the same column names. We need to rename some and make them consistent with the "q1-2020" dataset, as this may probably be the going-forward table design for Divvy.

```r
# Rename columns  to make them consistent with q1_2020

(q4_2019 <- rename(q4_2019
                   ,ride_id = trip_id
                   ,rideable_type = bikeid
                   ,started_at = start_time
                   ,ended_at = end_time
                   ,start_station_name = from_station_name
                   ,start_station_id = from_station_id
                   ,end_station_name = to_station_name
                   ,end_station_id = to_station_id
                   ,member_casual = usertype))
```

```
## # A tibble: 704,054 x 12
##     ride_id started_at          ended_at            rideable_type tripduration
##       <dbl> <dttm>              <dttm>                      <dbl>        <dbl>
##  1 25223640 2019-10-01 00:01:39 2019-10-01 00:17:20          2215          940
##  2 25223641 2019-10-01 00:02:16 2019-10-01 00:06:34          6328          258
##  3 25223642 2019-10-01 00:04:32 2019-10-01 00:18:43          3003          850
##  4 25223643 2019-10-01 00:04:32 2019-10-01 00:43:43          3275         2350
##  5 25223644 2019-10-01 00:04:34 2019-10-01 00:35:42          5294         1867
##  6 25223645 2019-10-01 00:04:38 2019-10-01 00:10:51          1891          373
##  7 25223646 2019-10-01 00:04:52 2019-10-01 00:22:45          1061         1072
##  8 25223647 2019-10-01 00:04:57 2019-10-01 00:29:16          1274         1458
##  9 25223648 2019-10-01 00:05:20 2019-10-01 00:29:18          6011         1437
## 10 25223649 2019-10-01 00:05:20 2019-10-01 02:23:46          2957         8306
## # i 704,044 more rows
## # i 7 more variables: start_station_id <dbl>, start_station_name <chr>,
## #   end_station_id <dbl>, end_station_name <chr>, member_casual <chr>,
## #   gender <chr>, birthyear <dbl>
```

```r
(q3_2019 <- rename(q3_2019
                   ,ride_id = trip_id
                   ,rideable_type = bikeid
                   ,started_at = start_time
                   ,ended_at = end_time
                   ,start_station_name = from_station_name
                   ,start_station_id = from_station_id
                   ,end_station_name = to_station_name
                   ,end_station_id = to_station_id
                   ,member_casual = usertype))
```

```
## # A tibble: 1,640,718 x 12
```

```
##     ride_id started_at          ended_at            rideable_type tripduration
##       <dbl> <dttm>              <dttm>                      <dbl>        <dbl>
##  1 23479388 2019-07-01 00:00:27 2019-07-01 00:20:41          3591         1214
##  2 23479389 2019-07-01 00:01:16 2019-07-01 00:18:44          5353         1048
##  3 23479390 2019-07-01 00:01:48 2019-07-01 00:27:42          6180         1554
##  4 23479391 2019-07-01 00:02:07 2019-07-01 00:27:10          5540         1503
##  5 23479392 2019-07-01 00:02:13 2019-07-01 00:22:26          6014         1213
##  6 23479393 2019-07-01 00:02:21 2019-07-01 00:07:31          4941          310
##  7 23479394 2019-07-01 00:02:24 2019-07-01 00:23:12          3770         1248
##  8 23479395 2019-07-01 00:02:26 2019-07-01 00:28:16          5442         1550
##  9 23479396 2019-07-01 00:02:34 2019-07-01 00:28:57          2957         1583
## 10 23479397 2019-07-01 00:02:45 2019-07-01 00:29:14          6091         1589
## # i 1,640,708 more rows
## # i 7 more variables: start_station_id <dbl>, start_station_name <chr>,
## #   end_station_id <dbl>, end_station_name <chr>, member_casual <chr>,
## #   gender <chr>, birthyear <dbl>
```

```
(q2_2019 <- rename(q2_2019
                  ,ride_id = "01 - Rental Details Rental ID"
                  ,rideable_type = "01 - Rental Details Bike ID"
                  ,started_at = "01 - Rental Details Local Start Time"
                  ,ended_at = "01 - Rental Details Local End Time"
                  ,start_station_name = "03 - Rental Start Station Name"
                  ,start_station_id = "03 - Rental Start Station ID"
                  ,end_station_name = "02 - Rental End Station Name"
                  ,end_station_id = "02 - Rental End Station ID"
                  ,member_casual = "User Type"))
```

```
## # A tibble: 1,108,163 x 12
##     ride_id started_at          ended_at            rideable_type
##       <dbl> <dttm>              <dttm>                      <dbl>
##  1 22178529 2019-04-01 00:02:22 2019-04-01 00:09:48          6251
##  2 22178530 2019-04-01 00:03:02 2019-04-01 00:20:30          6226
##  3 22178531 2019-04-01 00:11:07 2019-04-01 00:15:19          5649
##  4 22178532 2019-04-01 00:13:01 2019-04-01 00:18:58          4151
##  5 22178533 2019-04-01 00:19:26 2019-04-01 00:36:13          3270
##  6 22178534 2019-04-01 00:19:39 2019-04-01 00:23:56          3123
##  7 22178535 2019-04-01 00:26:33 2019-04-01 00:35:41          6418
##  8 22178536 2019-04-01 00:29:48 2019-04-01 00:36:11          4513
##  9 22178537 2019-04-01 00:32:07 2019-04-01 01:07:44          3280
## 10 22178538 2019-04-01 00:32:19 2019-04-01 01:07:39          5534
## # i 1,108,153 more rows
## # i 8 more variables: `01 - Rental Details Duration In Seconds Uncapped` <dbl>,
## #   start_station_id <dbl>, start_station_name <chr>, end_station_id <dbl>,
## #   end_station_name <chr>, member_casual <chr>, `Member Gender` <chr>,
## #   `05 - Member Details Member Birthday Year` <dbl>
```

We have now renamed the columns. Let's check for possible remaining inconsistencies. The str() function can do that.

```
# Inspect the data frames and look for inconsistencies
str(q1_2020)
```

```
## spc_tbl_ [426,887 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:426887] "EACB19130B0CDA4A" "8FED874C809DC021" "789F3C21E472CA96" "C9A38
##  $ rideable_type     : chr [1:426887] "docked_bike" "docked_bike" "docked_bike" "docked_bike" ...
##  $ started_at        : POSIXct[1:426887], format: "2020-01-21 20:06:59" "2020-01-30 14:22:39" ...
##  $ ended_at          : POSIXct[1:426887], format: "2020-01-21 20:14:30" "2020-01-30 14:26:22" ...
##  $ start_station_name: chr [1:426887] "Western Ave & Leland Ave" "Clark St & Montrose Ave" "Broadway
##  $ start_station_id  : num [1:426887] 239 234 296 51 66 212 96 96 212 38 ...
##  $ end_station_name  : chr [1:426887] "Clark St & Leland Ave" "Southport Ave & Irving Park Rd" "Wilt
##  $ end_station_id    : num [1:426887] 326 318 117 24 212 96 212 212 96 100 ...
##  $ start_lat         : num [1:426887] 42 42 41.9 41.9 41.9 ...
##  $ start_lng         : num [1:426887] -87.7 -87.7 -87.6 -87.6 -87.6 ...
##  $ end_lat           : num [1:426887] 42 42 41.9 41.9 41.9 ...
##  $ end_lng           : num [1:426887] -87.7 -87.7 -87.7 -87.6 -87.6 ...
##  $ member_casual     : chr [1:426887] "member" "member" "member" "member" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ride_id = col_character(),
##   ..   rideable_type = col_character(),
##   ..   started_at = col_datetime(format = ""),
##   ..   ended_at = col_datetime(format = ""),
##   ..   start_station_name = col_character(),
##   ..   start_station_id = col_double(),
##   ..   end_station_name = col_character(),
##   ..   end_station_id = col_double(),
##   ..   start_lat = col_double(),
##   ..   start_lng = col_double(),
##   ..   end_lat = col_double(),
##   ..   end_lng = col_double(),
##   ..   member_casual = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
str(q4_2019)
```

```
## spc_tbl_ [704,054 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : num [1:704054] 25223640 25223641 25223642 25223643 25223644 ...
##  $ started_at        : POSIXct[1:704054], format: "2019-10-01 00:01:39" "2019-10-01 00:02:16" ...
##  $ ended_at          : POSIXct[1:704054], format: "2019-10-01 00:17:20" "2019-10-01 00:06:34" ...
##  $ rideable_type     : num [1:704054] 2215 6328 3003 3275 5294 ...
##  $ tripduration      : num [1:704054] 940 258 850 2350 1867 ...
##  $ start_station_id  : num [1:704054] 20 19 84 313 210 156 84 156 156 336 ...
##  $ start_station_name: chr [1:704054] "Sheffield Ave & Kingsbury St" "Throop (Loomis) St & Taylor St
##  $ end_station_id    : num [1:704054] 309 241 199 290 382 226 142 463 463 336 ...
##  $ end_station_name  : chr [1:704054] "Leavitt St & Armitage Ave" "Morgan St & Polk St" "Wabash Ave &
##  $ member_casual     : chr [1:704054] "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
##  $ gender            : chr [1:704054] "Male" "Male" "Female" "Male" ...
##  $ birthyear         : num [1:704054] 1987 1998 1991 1990 1987 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   trip_id = col_double(),
##   ..   start_time = col_datetime(format = ""),
##   ..   end_time = col_datetime(format = ""),
##   ..   bikeid = col_double(),
##   ..   tripduration = col_number(),
```

```
##   ..   from_station_id = col_double(),
##   ..   from_station_name = col_character(),
##   ..   to_station_id = col_double(),
##   ..   to_station_name = col_character(),
##   ..   usertype = col_character(),
##   ..   gender = col_character(),
##   ..   birthyear = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
str(q3_2019)
```

```
## spc_tbl_ [1,640,718 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : num [1:1640718] 23479388 23479389 23479390 23479391 23479392 ...
##  $ started_at        : POSIXct[1:1640718], format: "2019-07-01 00:00:27" "2019-07-01 00:01:16" ...
##  $ ended_at          : POSIXct[1:1640718], format: "2019-07-01 00:20:41" "2019-07-01 00:18:44" ...
##  $ rideable_type     : num [1:1640718] 3591 5353 6180 5540 6014 ...
##  $ tripduration      : num [1:1640718] 1214 1048 1554 1503 1213 ...
##  $ start_station_id  : num [1:1640718] 117 381 313 313 168 300 168 313 43 43 ...
##  $ start_station_name: chr [1:1640718] "Wilton Ave & Belmont Ave" "Western Ave & Monroe St" "Lakevie
##  $ end_station_id    : num [1:1640718] 497 203 144 144 62 232 62 144 195 195 ...
##  $ end_station_name  : chr [1:1640718] "Kimball Ave & Belmont Ave" "Western Ave & 21st St" "Larrabee
##  $ member_casual     : chr [1:1640718] "Subscriber" "Customer" "Customer" "Customer" ...
##  $ gender            : chr [1:1640718] "Male" NA NA NA ...
##  $ birthyear         : num [1:1640718] 1992 NA NA NA NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   trip_id = col_double(),
##   ..   start_time = col_datetime(format = ""),
##   ..   end_time = col_datetime(format = ""),
##   ..   bikeid = col_double(),
##   ..   tripduration = col_number(),
##   ..   from_station_id = col_double(),
##   ..   from_station_name = col_character(),
##   ..   to_station_id = col_double(),
##   ..   to_station_name = col_character(),
##   ..   usertype = col_character(),
##   ..   gender = col_character(),
##   ..   birthyear = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
str(q2_2019)
```

```
## spc_tbl_ [1,108,163 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id                                          : num [1:1108163] 22178529 22178530 22178531 2217
##  $ started_at                                       : POSIXct[1:1108163], format: "2019-04-01 00:02:2
##  $ ended_at                                         : POSIXct[1:1108163], format: "2019-04-01 00:09:4
##  $ rideable_type                                    : num [1:1108163] 6251 6226 5649 4151 3270 ...
##  $ 01 - Rental Details Duration In Seconds Uncapped: num [1:1108163] 446 1048 252 357 1007 ...
##  $ start_station_id                                 : num [1:1108163] 81 317 283 26 202 420 503 260 2
##  $ start_station_name                               : chr [1:1108163] "Daley Center Plaza" "Wood St &
##  $ end_station_id                                   : num [1:1108163] 56 59 174 133 129 426 500 499 2
```

```
##  $ end_station_name                                  : chr [1:1108163] "Desplaines St & Kinzie St" "Wal
##  $ member_casual                                     : chr [1:1108163] "Subscriber" "Subscriber" "Subs
##  $ Member Gender                                     : chr [1:1108163] "Male" "Female" "Male" "Male" .
##  $ 05 - Member Details Member Birthday Year          : num [1:1108163] 1975 1984 1990 1993 1992 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `01 - Rental Details Rental ID` = col_double(),
##   ..   `01 - Rental Details Local Start Time` = col_datetime(format = ""),
##   ..   `01 - Rental Details Local End Time` = col_datetime(format = ""),
##   ..   `01 - Rental Details Bike ID` = col_double(),
##   ..   `01 - Rental Details Duration In Seconds Uncapped` = col_number(),
##   ..   `03 - Rental Start Station ID` = col_double(),
##   ..   `03 - Rental Start Station Name` = col_character(),
##   ..   `02 - Rental End Station ID` = col_double(),
##   ..   `02 - Rental End Station Name` = col_character(),
##   ..   `User Type` = col_character(),
##   ..   `Member Gender` = col_character(),
##   ..   `05 - Member Details Member Birthday Year` = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

It turns out that the "ride-id" and "rideable-type" do not stack correctly. Their types need to be converted into characters, which is used for the "q1-2020" dataset.

```
# Convert ride_id and rideable_type to characters so that they can stack correctly
q4_2019 <-  mutate(q4_2019, ride_id = as.character(ride_id)
                   ,rideable_type = as.character(rideable_type))
q3_2019 <-  mutate(q3_2019, ride_id = as.character(ride_id)
                   ,rideable_type = as.character(rideable_type))
q2_2019 <-  mutate(q2_2019, ride_id = as.character(ride_id)
                   ,rideable_type = as.character(rideable_type))
```

The datasets are now ready to be combined into one dataframe for further analysis. We can also remove some columns that are not useful for our study. We can consider removing variables such as "lat", "long", "birthyear", and "gender fields" as this data was dropped beginning in 2020.

```
# Stack individual quarter's data frames into one big data frame
all_trips <- bind_rows(q2_2019, q3_2019, q4_2019, q1_2020)

# Remove lat, long, birthyear, and gender fields
all_trips <- all_trips %>%
  select(-c(start_lat, start_lng, end_lat, end_lng, birthyear, gender, "01 - Rental Details Duration In
```

STEP 3: CLEAN UP AND ADD DATA TO PREPARE FOR ANALYSIS

Now, it is time to clean up the dataset that we have saved in "all_trips" and prepare for analysis. We can first inspect it and check for possible problems that need to be fixed before conducting the analysis.

```r
# Inspect the new table that has been created

colnames(all_trips)  #List of column names
```

```
## [1] "ride_id"            "started_at"         "ended_at"
## [4] "rideable_type"      "start_station_id"   "start_station_name"
## [7] "end_station_id"     "end_station_name"   "member_casual"
```

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

```
## [1] 3879822
```

```r
dim(all_trips)  #Dimensions of the data frame?
```

```
## [1] 3879822       9
```

```r
head(all_trips)  #See the first 6 rows of the data frame. Also, the tail(qs_raw) can show the last rows
```

```
## # A tibble: 6 x 9
##   ride_id started_at          ended_at            rideable_type start_station_id
##   <chr>   <dttm>              <dttm>              <chr>                     <dbl>
## 1 221785~ 2019-04-01 00:02:22 2019-04-01 00:09:48 6251                         81
## 2 221785~ 2019-04-01 00:03:02 2019-04-01 00:20:30 6226                        317
## 3 221785~ 2019-04-01 00:11:07 2019-04-01 00:15:19 5649                        283
## 4 221785~ 2019-04-01 00:13:01 2019-04-01 00:18:58 4151                         26
## 5 221785~ 2019-04-01 00:19:26 2019-04-01 00:36:13 3270                        202
## 6 221785~ 2019-04-01 00:19:39 2019-04-01 00:23:56 3123                        420
## # i 4 more variables: start_station_name <chr>, end_station_id <dbl>,
## #   end_station_name <chr>, member_casual <chr>
```

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

```
## tibble [3,879,822 x 9] (S3: tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:3879822] "22178529" "22178530" "22178531" "22178532" ...
##  $ started_at        : POSIXct[1:3879822], format: "2019-04-01 00:02:22" "2019-04-01 00:03:02" ...
##  $ ended_at          : POSIXct[1:3879822], format: "2019-04-01 00:09:48" "2019-04-01 00:20:30" ...
##  $ rideable_type     : chr [1:3879822] "6251" "6226" "5649" "4151" ...
##  $ start_station_id  : num [1:3879822] 81 317 283 26 202 420 503 260 211 211 ...
##  $ start_station_name: chr [1:3879822] "Daley Center Plaza" "Wood St & Taylor St" "LaSalle St & Jacks
##  $ end_station_id    : num [1:3879822] 56 59 174 133 129 426 500 499 211 211 ...
##  $ end_station_name  : chr [1:3879822] "Desplaines St & Kinzie St" "Wabash Ave & Roosevelt Rd" "Canal
##  $ member_casual     : chr [1:3879822] "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
```

```r
summary(all_trips)  #Statistical summary of data. Mainly for numerics
```

```
##    ride_id            started_at
##  Length:3879822     Min.   :2019-04-01 00:02:22.00
##  Class :character   1st Qu.:2019-06-23 07:49:09.25
##  Mode  :character   Median :2019-08-14 17:43:38.00
```

```
##                     Mean   :2019-08-26 00:49:59.38
##                     3rd Qu.:2019-10-12 12:10:21.00
##                     Max.   :2020-03-31 23:51:34.00
##
##     ended_at                      rideable_type      start_station_id
##  Min.   :2019-04-01 00:09:48.00   Length:3879822     Min.   :  1.0
##  1st Qu.:2019-06-23 08:20:27.75   Class :character   1st Qu.: 77.0
##  Median :2019-08-14 18:02:04.00   Mode  :character   Median :174.0
##  Mean   :2019-08-26 01:14:37.06                      Mean   :202.9
##  3rd Qu.:2019-10-12 12:36:16.75                      3rd Qu.:291.0
##  Max.   :2020-05-19 20:10:34.00                      Max.   :675.0
##
##  start_station_name end_station_id  end_station_name   member_casual
##  Length:3879822     Min.   :  1.0   Length:3879822     Length:3879822
##  Class :character   1st Qu.: 77.0   Class :character   Class :character
##  Mode  :character   Median :174.0   Mode  :character   Mode  :character
##                     Mean   :203.8
##                     3rd Qu.:291.0
##                     Max.   :675.0
##                     NA's   :1
```

After the inspection, it turns out that there are a few problems that need to be fixed:

1. In the "$member - casual$" column, there are two names for members ("member" and "Subscriber") and two names for casual riders ("Customer" and "casual"). We will need to consolidate that from four to two labels.
2. The data can only be aggregated at the ride level, which is too granular. We will want to add some additional columns of data such as day, month, year..., that provide additional opportunities to aggregate the data.
3. We will want to add a calculated field for the length of a ride since the "q1-2020" data did not have the "trip duration" column. We will add "ride length" to the entire data frame for consistency.
4. There are some rides where trip duration shows up as negative, including several hundred rides where Divvy took bikes out of circulation for Quality Control reasons. We will want to delete these rides.

In the "$member - casual$" column, we replace "Subscriber" with "member" and "Customer" with "casual". Before 2020, Divvy used different labels for these two types of riders ... we will want to make our data frame consistent with their 2020 nomenclature. (N.B.: "Level" is a special property of a column that is retained even if a subset does not contain any values from a specific level.)

```
# Begin by seeing how many observations fall under each user
table(all_trips$member_casual)
```

```
##
##     casual   Customer     member Subscriber
##      48480     857474     378407    2595461
```

```
# Reassign to the desired values (we will go with the 2020 labels)
all_trips <-  all_trips %>%
  mutate(member_casual = recode(member_casual
                           ,"Subscriber" = "member"
```

```
                            ,"Customer" = "casual"))

# Check to make sure the proper number of observations were reassigned
table(all_trips$member_casual)
```

```
##
##  casual  member
##  905954 2973868
```

It turns out that the total number of Cyclistic's members is more than triple the total number of casual customers.

Next, let's add columns that list the date, month, day, and year of each ride. This will allow us to aggregate ride data for each month, day, or year... before completing these operations we could only aggregate at the ride level. We also need to add the ride length column that shows the duration of the rides.

```
#adding columns data, month, day, year, and day of the week
all_trips$date <- as.Date(all_trips$started_at) #The default format is yyyy-mm-dd
all_trips$month <- format(as.Date(all_trips$date), "%m")
all_trips$day <- format(as.Date(all_trips$date), "%d")
all_trips$year <- format(as.Date(all_trips$date), "%Y")
all_trips$day_of_week <- format(as.Date(all_trips$date), "%A")

# Add a "ride_length" calculation to all_trips (in seconds)
all_trips$ride_length <- difftime(all_trips$ended_at,all_trips$started_at)
```

To make sure that the newly created columns match our intended use, we need to inspect the newly obtained data (functions such as str() or glimpse () can do that).

```
# Inspect the structure of the columns
glimpse(all_trips)
```

```
## Rows: 3,879,822
## Columns: 15
## $ ride_id            <chr> "22178529", "22178530", "22178531", "22178532", "22~
## $ started_at         <dttm> 2019-04-01 00:02:22, 2019-04-01 00:03:02, 2019-04-~
## $ ended_at           <dttm> 2019-04-01 00:09:48, 2019-04-01 00:20:30, 2019-04-~
## $ rideable_type      <chr> "6251", "6226", "5649", "4151", "3270", "3123", "64~
## $ start_station_id   <dbl> 81, 317, 283, 26, 202, 420, 503, 260, 211, 211, 304~
## $ start_station_name <chr> "Daley Center Plaza", "Wood St & Taylor St", "LaSal~
## $ end_station_id     <dbl> 56, 59, 174, 133, 129, 426, 500, 499, 211, 211, 232~
## $ end_station_name   <chr> "Desplaines St & Kinzie St", "Wabash Ave & Roosevel~
## $ member_casual      <chr> "member", "member", "member", "member", "member", "~
## $ date               <date> 2019-04-01, 2019-04-01, 2019-04-01, 2019-04-01, 20~
## $ month              <chr> "04", "04", "04", "04", "04", "04", "04", "04", "04~
## $ day                <chr> "01", "01", "01", "01", "01", "01", "01", "01", "01~
## $ year               <chr> "2019", "2019", "2019", "2019", "2019", "2019", "20~
## $ day_of_week        <chr> "Monday", "Monday", "Monday", "Monday", "Monday", "~
## $ ride_length        <drtn> 446 secs, 1048 secs, 252 secs, 357 secs, 1007 secs~
```

After the inspection, one can find that it is necessary to Convert "ride-length" from Factor to numeric so we can run calculations on the data

```r
# Convert "ride_length" from Factor to numeric
all_trips$ride_length <- as.numeric(as.character(all_trips$ride_length))
is.numeric(all_trips$ride_length)
```

```
## [1] TRUE
```

```r
sum(all_trips$ride_length<0) # checking if there are some negative values
```

```
## [1] 130
```

Remove "bad" data

The data frame includes a few hundred entries when bikes were taken out of docks and checked for quality by Divvy or when ride length was negative. Since the data is being removed, we will create a new version of the data frame (v2).

```r
all_trips_v2 <- all_trips[!(all_trips$start_station_name == "HQ QR" | all_trips$ride_length<0),]
```

Now, we can proceed with the analysis using the ready, clean, and processed dataset "all_trips_v2" for the analysis.

STEP 4: CONDUCT DESCRIPTIVE ANALYSIS

We can begin by conducting a descriptive analysis of the "ride-length" variable (in seconds). We can calculate quantiles, median ride length, mean ride length, minimum (the shortest), and maximum (the longest) ride length. The summary() function can do the work.

```r
#summary statistics
summary(all_trips_v2$ride_length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1     412     712    1479    1289 9387024
```

Next, we can compare members and casual users using descriptive measures (mean, median, max, and min).

```r
# Compare members and casual users
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = mean)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                3552.7502
## 2                     member                 850.0662
```

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = median)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                      1546
## 2                     member                       589
```

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = max)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                  9387024
## 2                     member                  9056634
```

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = min)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                        2
## 2                     member                        1
```

This gives the general difference in ride length for members and casual. But is there any difference based on days of the week?. See the average ride time by each day for members vs casual users.

```
# See the average ride time by each day for members vs casual users
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual + all_trips_v2$day_of_week, FUN = mean)
```

```
##    all_trips_v2$member_casual all_trips_v2$day_of_week all_trips_v2$ride_length
## 1                      casual                   Friday                3773.8351
## 2                      member                   Friday                 824.5305
## 3                      casual                   Monday                3372.2869
## 4                      member                   Monday                 842.5726
## 5                      casual                 Saturday                3331.9138
## 6                      member                 Saturday                 968.9337
## 7                      casual                   Sunday                3581.4054
## 8                      member                   Sunday                 919.9746
## 9                      casual                 Thursday                3682.9847
## 10                     member                 Thursday                 823.9278
## 11                     casual                  Tuesday                3596.3599
## 12                     member                  Tuesday                 826.1427
## 13                     casual                Wednesday                3718.6619
## 14                     member                Wednesday                 823.9996
```

Notice that the days of the week are out of order. Let's fix that.

```
all_trips_v2$day_of_week <- ordered(all_trips_v2$day_of_week, levels=c("Sunday", "Monday", "Tuesday", "
```

Now, let's run the average ride time for members vs casual users each day.

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual + all_trips_v2$day_of_week, FUN = mean)
```

```
##    all_trips_v2$member_casual all_trips_v2$day_of_week all_trips_v2$ride_length
## 1                      casual                   Sunday                3581.4054
## 2                      member                   Sunday                 919.9746
## 3                      casual                   Monday                3372.2869
## 4                      member                   Monday                 842.5726
## 5                      casual                  Tuesday                3596.3599
## 6                      member                  Tuesday                 826.1427
## 7                      casual                Wednesday                3718.6619
```

```
## 8                    member               Wednesday                 823.9996
## 9                    casual                Thursday                3682.9847
## 10                   member                Thursday                 823.9278
## 11                   casual                  Friday                3773.8351
## 12                   member                  Friday                 824.5305
## 13                   casual                Saturday                3331.9138
## 14                   member                Saturday                 968.9337
```

Now, let's analyze ridership data by type and weekday and make a clear visualization for the number of rides by rider type (member or casual).

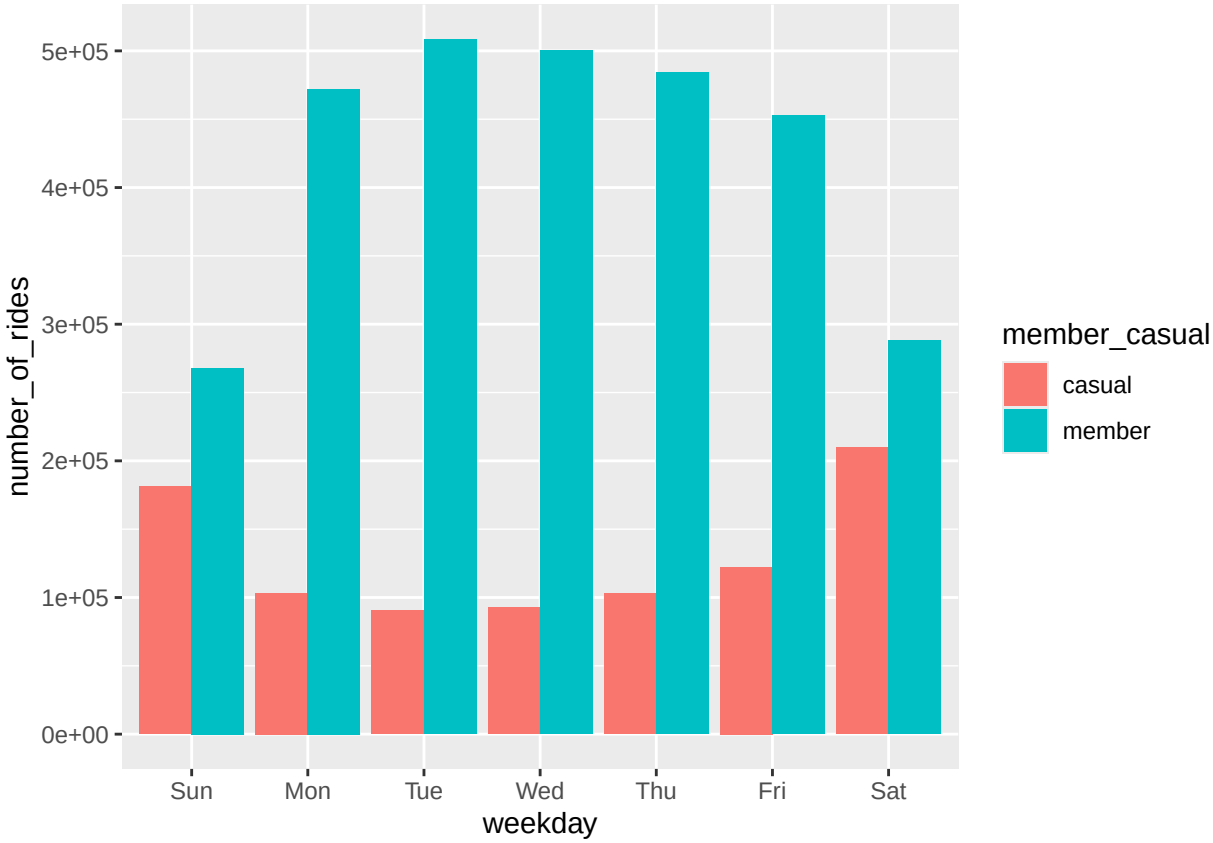
```
# analyze ridership data by type and weekday
all_trips_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%  #creates weekday field using wday()
  group_by(member_casual, weekday) %>%  #groups by user-type and weekday
  summarise(number_of_rides = n()                              #calculates the number of rides and average
  ,average_duration = mean(ride_length)) %>%        # calculates the average duration
  arrange(member_casual, weekday)                                  # sorts
```

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

```
## # A tibble: 14 x 4
## # Groups:   member_casual [2]
##    member_casual weekday number_of_rides average_duration
##    <chr>         <ord>             <int>            <dbl>
##  1 casual        Sun              181293            3581.
##  2 casual        Mon              103296            3372.
##  3 casual        Tue               90510            3596.
##  4 casual        Wed               92457            3719.
##  5 casual        Thu              102679            3683.
##  6 casual        Fri              122404            3774.
##  7 casual        Sat              209543            3332.
##  8 member        Sun              267965             920.
##  9 member        Mon              472196             843.
## 10 member        Tue              508445             826.
## 11 member        Wed              500329             824.
## 12 member        Thu              484177             824.
## 13 member        Fri              452790             825.
## 14 member        Sat              287958             969.
```

```
# Let's visualize the number of rides by rider type
all_trips_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday)  %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge")
```

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

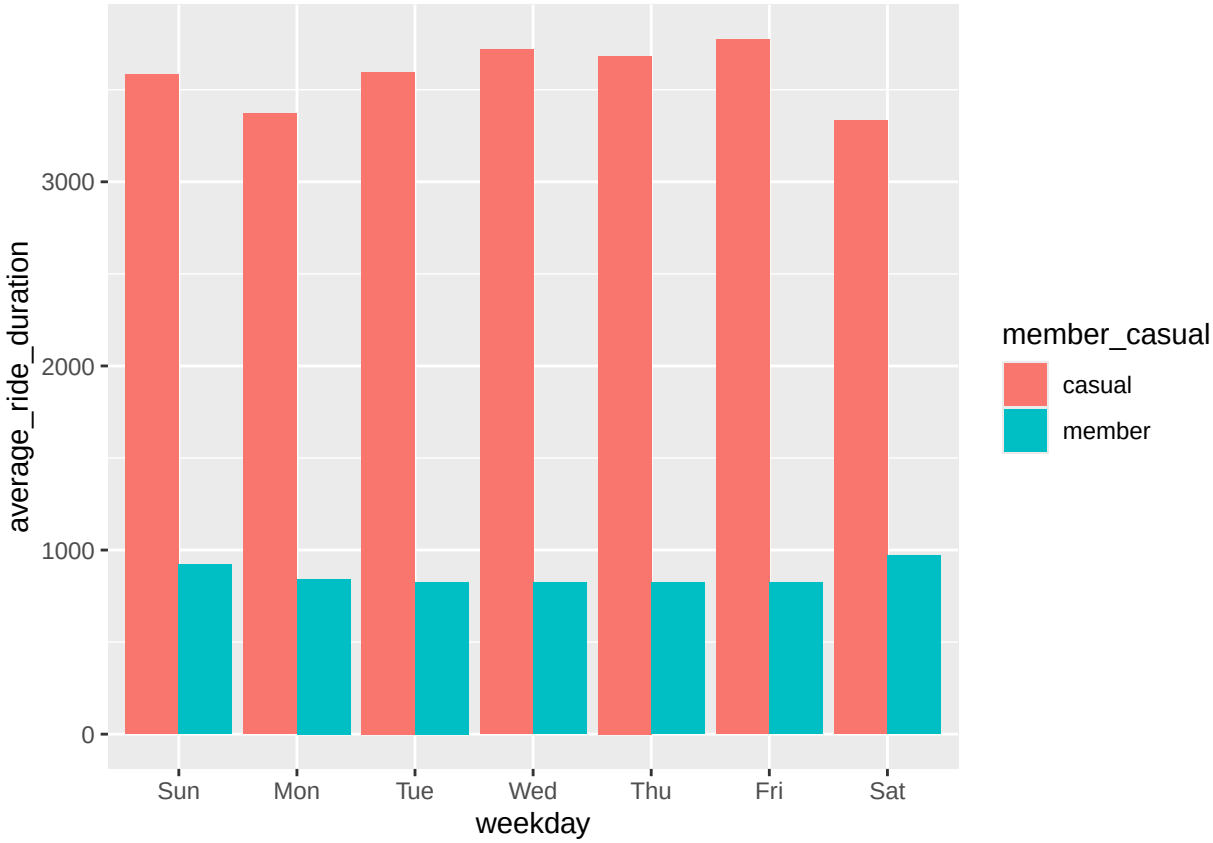


The graph above shows that members use bicycles less on weekends compared to other days of the week. The trend is the opposite for casual riders who use bicycles much on weekends. During the week, there are always more members than casual riders.

We can also visualize the average ride duration by rider type and weekday.

```
# Let's create a visualization for the average ride duration
all_trips_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%
  summarise(number_of_rides = n()
            ,average_ride_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday)  %>%
  ggplot(aes(x = weekday, y = average_ride_duration, fill = member_casual)) +
  geom_col(position = "dodge")
```

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



The graph shows that the average ride duration for casual riders is greater than the one for members.

STEP 5: EXPORT SUMMARY FILE FOR FURTHER ANALYSIS

Now that we have completed our analysis, it is necessary to save the summary file for further analysis and visualization (probably with a different software).

```
counts <- aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual + all_trips_v2$day_of_week, FU
write.csv(counts, file = 'D:/my_projects/counts.csv')
```

STEP 6: INSIGHTS AND RECOMMENDATIONS

Insights

1. **Riding Behavior by Day of the Week**:
    - Casual riders prefer weekend rides, while members have consistent usage throughout the week, primarily on weekdays.
    - Casual riders' average ride duration is significantly higher, suggesting their trips are more recreational than functional.
2. **Peak Usage Times**:

- Casual riders likely ride during leisure hours (e.g., late mornings or afternoons), while members are more likely to use bikes during peak commute times (e.g., early mornings and evenings on weekdays).

3. **Membership Value Perception**:
   - Casual riders might not perceive annual memberships as cost-effective due to infrequent weekday usage. Highlighting long-term cost savings could address this.
4. **Bicycle Availability**:
   - Higher ride durations for casual riders could indicate potential availability issues at popular locations during weekends, potentially leading to dissatisfaction.

Recommendations

1. **Weekend-Focused Membership Campaigns**:
   - Develop memberships tailored to weekend riders, emphasizing discounted weekend rides as a perk for annual members.
   - Partner with recreational destinations or events to offer ride-and-save packages for members.
2. **Targeted Digital Marketing**:
   - Use digital campaigns targeting casual riders with messages like "Save more on your weekend adventures" or "Unlimited weekend rides with our annual membership."
3. **Dynamic Pricing Models**:
   - Introduce flexible memberships, such as weekend-only passes, to ease casual riders into membership options.
4. **Enhance Accessibility During Peak Times**:
   - Ensure sufficient bike availability at high-demand locations, especially on weekends, to maintain a positive customer experience.
   - Use demand forecasting models to optimize bike reallocation.
5. **Incentives for Weekday Usage**:
   - Provide weekday ride discounts for casual riders to encourage trial usage during these days.
   - Highlight the convenience and cost benefits of weekday commuting through memberships.
6. **User Behavior Surveys**:
   - Conduct surveys targeting casual riders to better understand barriers to membership conversion and preferences for new membership structures.

These strategies aim to convert casual riders into members while improving customer satisfaction and loyalty, directly supporting Cyclistic's growth objectives.