# Cyclistic Case Study

Hezar K

2022-11-29

This is an analysis for Cyclistic Case Study for Google Data Analytics Course. This is an analysis for May 2021.

### **STEP ONE:** INSTALL REQUIRED PACKAGES AND IMPORT DATA

Install the required packages. **Tidyverse** package to import and wrangling the data and **ggplot2** package for visualization of the data. **Lubridate** package for date parsing and **anytime** package for the datetime conversion.

- install.packages("tidyverse")
- install.packages("ggplot2")
- install.packages("lubridate")
- install.packages("anytime")

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.4.0      ✔ purrr   0.3.5
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.1
## ✔ readr   2.1.3      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
## Loading required package: timechange
## 
## Attaching package: 'lubridate'
## 
## The following objects are masked from 'package:base':
## 
##     date, intersect, setdiff, union
```

```
library(data.table)
```

```
## 
## Attaching package: 'data.table'
## 
## The following objects are masked from 'package:lubridate':
## 
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
## 
## The following objects are masked from 'package:dplyr':
## 
##     between, first, last
## 
## The following object is masked from 'package:purrr':
## 
##     transpose
```

```
library(ggplot2)
library(anytime)
```

Import data from local drive.

```
May21 <- read_csv("C:/Users/theby/Documents/202105-divvy-tripdata.csv")
```

```
## Rows: 531633 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

### **STEP TWO:** EXAMINE THE DATA

Examine the dataframe for an overview of the data. Review column names, **colnames()**, dimensions of the dataframe by row and column, **dim()**, the first, **head()**, and the last, **tail()**, six rows in the dataframe, the summary, **summary()**, statistics on the columns of the dataframe, and review the data type structure of columns, **str()**.

View(May21)

```
colnames(May21)
```

```
##  [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"
```

```
nrow(May21)
```

```
## [1] 531633
```

```
dim(May21)
```

```
## [1] 531633     13
```

```
head(May21)
```

```
## # A tibble: 6 × 13
##   ride_id        ridea…¹ started_at          ended_at            start…² start…³
##   <chr>          <chr>   <dttm>              <dttm>              <chr>   <chr>
## 1 C809ED75D6160… electr… 2021-05-30 11:58:15 2021-05-30 12:10:39 <NA>    <NA>
## 2 DD59FDCE0ACAC… electr… 2021-05-30 11:29:14 2021-05-30 12:14:09 <NA>    <NA>
## 3 0AB83CB88C43E… electr… 2021-05-30 14:24:01 2021-05-30 14:25:13 <NA>    <NA>
## 4 7881AC6D39110… electr… 2021-05-30 14:25:51 2021-05-30 14:41:04 <NA>    <NA>
## 5 853FA701B4582… electr… 2021-05-30 18:15:39 2021-05-30 18:22:32 <NA>    <NA>
## 6 F5E63DFD96B2A… electr… 2021-05-30 11:33:41 2021-05-30 11:57:17 <NA>    <NA>
## # … with 7 more variables: end_station_name <chr>, end_station_id <chr>,
## #   start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, end_lng <dbl>,
## #   member_casual <chr>, and abbreviated variable names ¹rideable_type,
## #   ²start_station_name, ³start_station_id
```

```
tail(May21)
```

```
## # A tibble: 6 × 13
##   ride_id        ridea…¹ started_at          ended_at            start…² start…³
##   <chr>          <chr>   <dttm>              <dttm>              <chr>   <chr>
## 1 D0B8E59E2B3C4… electr… 2021-05-02 17:48:17 2021-05-02 17:52:19 Blacks… 13398
## 2 EF56D7D1D612A… electr… 2021-05-20 16:32:14 2021-05-20 16:35:39 Blacks… 13398
## 3 745191CB9F21D… classi… 2021-05-29 16:40:37 2021-05-29 17:22:37 Sherid… TA1307…
## 4 428575BAA5356… electr… 2021-05-31 14:24:54 2021-05-31 14:31:38 Sherid… TA1307…
## 5 FC8A4A7AB7249… electr… 2021-05-25 16:01:33 2021-05-25 16:07:37 Sherid… TA1307…
## 6 E873B8AA3EE84… docked… 2021-05-12 12:22:14 2021-05-12 12:30:27 Sherid… TA1307…
## # … with 7 more variables: end_station_name <chr>, end_station_id <chr>,
## #   start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, end_lng <dbl>,
## #   member_casual <chr>, and abbreviated variable names ¹rideable_type,
## #   ²start_station_name, ³start_station_id
```

```
summary(May21)
```

```
##    ride_id          rideable_type        started_at                   
##  Length:531633      Length:531633      Min.   :2021-05-01 00:00:11.00  
##  Class :character   Class :character   1st Qu.:2021-05-10 17:40:50.00  
##  Mode  :character   Mode  :character   Median :2021-05-19 07:44:31.00  
##                                        Mean   :2021-05-17 19:52:32.05  
##                                        3rd Qu.:2021-05-24 19:32:22.00  
##                                        Max.   :2021-05-31 23:59:16.00  
##                                                                         
##     ended_at                      start_station_name start_station_id  
##  Min.   :2021-05-01 00:03:26.00   Length:531633      Length:531633     
##  1st Qu.:2021-05-10 17:57:59.00   Class :character   Class :character  
##  Median :2021-05-19 07:59:43.00   Mode  :character   Mode  :character  
##  Mean   :2021-05-17 20:18:34.46                                        
##  3rd Qu.:2021-05-24 19:57:20.00                                        
##  Max.   :2021-06-10 22:17:11.00                                        
##                                                                         
##  end_station_name   end_station_id       start_lat       start_lng     
##  Length:531633      Length:531633      Min.   :41.65   Min.   :-87.78  
##  Class :character   Class :character   1st Qu.:41.88   1st Qu.:-87.66  
##  Mode  :character   Mode  :character   Median :41.90   Median :-87.64  
##                                        Mean   :41.90   Mean   :-87.64  
##                                        3rd Qu.:41.93   3rd Qu.:-87.63  
##                                        Max.   :42.07   Max.   :-87.52  
##                                                                         
##     end_lat         end_lng       member_casual     
##  Min.   :41.56   Min.   :-87.85   Length:531633     
##  1st Qu.:41.88   1st Qu.:-87.66   Class :character  
##  Median :41.90   Median :-87.64   Mode  :character  
##  Mean   :41.90   Mean   :-87.64                     
##  3rd Qu.:41.93   3rd Qu.:-87.63                     
##  Max.   :42.09   Max.   :-87.52                     
##  NA's   :452     NA's   :452
```

```
str(May21)
```

```
## spc_tbl_ [531,633 × 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:531633] "C809ED75D6160B2A" "DD59FDCE0ACACAF3" "0AB83CB88C43EFC2" "7881AC6D39110C
60" ...
##  $ rideable_type     : chr [1:531633] "electric_bike" "electric_bike" "electric_bike" "electric_bike" ...
##  $ started_at        : POSIXct[1:531633], format: "2021-05-30 11:58:15" "2021-05-30 11:29:14" ...
##  $ ended_at          : POSIXct[1:531633], format: "2021-05-30 12:10:39" "2021-05-30 12:14:09" ...
##  $ start_station_name: chr [1:531633] NA NA NA NA ...
##  $ start_station_id  : chr [1:531633] NA NA NA NA ...
##  $ end_station_name  : chr [1:531633] NA NA NA NA ...
##  $ end_station_id    : chr [1:531633] NA NA NA NA ...
##  $ start_lat         : num [1:531633] 41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num [1:531633] -87.6 -87.6 -87.7 -87.7 -87.7 ...
##  $ end_lat           : num [1:531633] 41.9 41.8 41.9 41.9 41.9 ...
##  $ end_lng           : num [1:531633] -87.6 -87.6 -87.7 -87.7 -87.7 ...
##  $ member_casual     : chr [1:531633] "casual" "casual" "casual" "casual" ...
##  - 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_character(),
##   ..   end_station_name = col_character(),
##   ..   end_station_id = col_character(),
##   ..   start_lat = col_double(),
##   ..   start_lng = col_double(),
##   ..   end_lat = col_double(),
##   ..   end_lng = col_double(),
##   ..   member_casual = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Create new columns as for *date*, *month*, *day*, *year*, *day_of_week*, and *ride_length* in seconds.

```
May21$date <- as.Date(May21$started_at)
May21$month <- format(as.Date(May21$date), "%m")
May21$day <- format(as.Date(May21$date), "%d")
May21$year <- format(as.Date(May21$date), "%Y")
May21$day_of_week <- format(as.Date(May21$date), "%A")
May21$ride_length <- difftime(May21$ended_at,May21$started_at)
```

Convert *ride_length* column to numeric in order to run calculations on the data. First, check to see if the data type is numeric, and then convert if needed.

```
is.numeric(May21$ride_length)
```

```
## [1] FALSE
```

Recheck *ride_length* data type.

```
May21$ride_length <- as.numeric(as.character(May21$ride_length))
is.numeric(May21$ride_length)
```

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

### **STEP THREE:** CLEAN DATA

**na.omit()** will remove all NA from the dataframe.

```
May21 <- na.omit(May21)
```

Remove rows with the *ride_id* column character length is not 16. This will remove all the scientific ride ids that we noticed while examining the data.

```
May21 <- subset(May21, nchar(as.character(ride_id)) == 16)
```

Remove rows with the *ride_length* less than 1 minute.

```
May21 <- subset (May21, ride_length > "1")
```

### **STEP FOUR:** ANALYZE DATA

Analyze the dataframe by find the **mean**, **median**, **max** (maximum), and **min** (minimum) of *ride_length*.

```
mean(May21$ride_length)
```

```
## [1] 1590.478
```

```
median(May21$ride_length)
```

```
## [1] 840
```

```
max(May21$ride_length)
```

```
## [1] 3235296
```

```
min(May21$ride_length)
```

```
## [1] 2
```

Run a statistical summary of the *ride_length*.

```
summary(May21$ride_length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       2     466     840    1590    1557 3235296
```

Compare the members and casual users

```
aggregate(May21$ride_length ~ May21$member_casual, FUN = mean)
```

```
##   May21$member_casual May21$ride_length
## 1              casual         2378.3001
## 2              member          860.8493
```

```
aggregate(May21$ride_length ~ May21$member_casual, FUN = median)
```

```
##   May21$member_casual May21$ride_length
## 1              casual               1181
## 2              member                637
```

```
aggregate(May21$ride_length ~ May21$member_casual, FUN = max)
```

```
##   May21$member_casual May21$ride_length
## 1              casual            3235296
## 2              member              88000
```

```
aggregate(May21$ride_length ~ May21$member_casual, FUN = min)
```

```
##   May21$member_casual May21$ride_length
## 1              casual                  2
## 2              member                  2
```

Aggregate the average ride length by each day of the week for members and users.

```
aggregate(May21$ride_length ~ May21$member_casual + May21$day_of_week, FUN = mean)
```

```
##    May21$member_casual May21$day_of_week May21$ride_length
## 1               casual            Friday         2196.0943
## 2               member            Friday          812.8785
## 3               casual            Monday         2332.7619
## 4               member            Monday          833.1902
## 5               casual          Saturday         2399.1110
## 6               member          Saturday          963.7124
## 7               casual            Sunday         2882.6615
## 8               member            Sunday         1004.6901
## 9               casual          Thursday         2012.4340
## 10              member          Thursday          794.5412
## 11              casual           Tuesday         1900.6872
## 12              member           Tuesday          758.0053
## 13              casual         Wednesday         1921.6968
## 14              member         Wednesday          805.9739
```

Sort the days of the week in order.

```
May21$day_of_week <- ordered(May21$day_of_week, levels=c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday"))
```

Assign the aggregate the average ride length by each day of the week for members and users to x.

```
x <- aggregate(May21$ride_length ~ May21$member_casual + May21$day_of_week, FUN = mean)

head(x)
```

```
##   May21$member_casual May21$day_of_week May21$ride_length
## 1              casual            Sunday         2882.6615
## 2              member            Sunday         1004.6901
## 3              casual            Monday         2332.7619
## 4              member            Monday          833.1902
## 5              casual           Tuesday         1900.6872
## 6              member           Tuesday          758.0053
```

Find the average ride length of member riders and casual riders per day and assign it to y.

```
y <- May21 %>%
  mutate(weekday = wday(started_at)) %>%
  group_by(member_casual, weekday) %>%
  summarise(number_of_rides = n(),
            average_duration = mean(ride_length), .groups = 'drop') %>%
  arrange(member_casual, weekday)

head(y)
```

```
## # A tibble: 6 × 4
##   member_casual weekday number_of_rides average_duration
##   <chr>           <int>           <int>            <dbl>
## 1 casual              1           54125            2883.
## 2 casual              2           29003            2333.
## 3 casual              3           14990            1901.
## 4 casual              4           18926            1922.
## 5 casual              5           18585            2012.
## 6 casual              6           24956            2196.
```

Analyze the dataframe to find the frequency of member riders, casual riders, classic bikes, docked bikes, and electric bikes.

```
table(May21$member_casual)
```

```
## 
## casual member 
## 216807 234099
```

```
table(May21$rideable_type)
```

```
## 
##  classic_bike   docked_bike electric_bike 
##        308260         43351         99295
```

```
table(May21$day_of_week)
```

```
## 
##    Sunday    Monday   Tuesday Wednesday  Thursday    Friday  Saturday 
##     90058     64557     43652     52622     49093     55054     95870
```

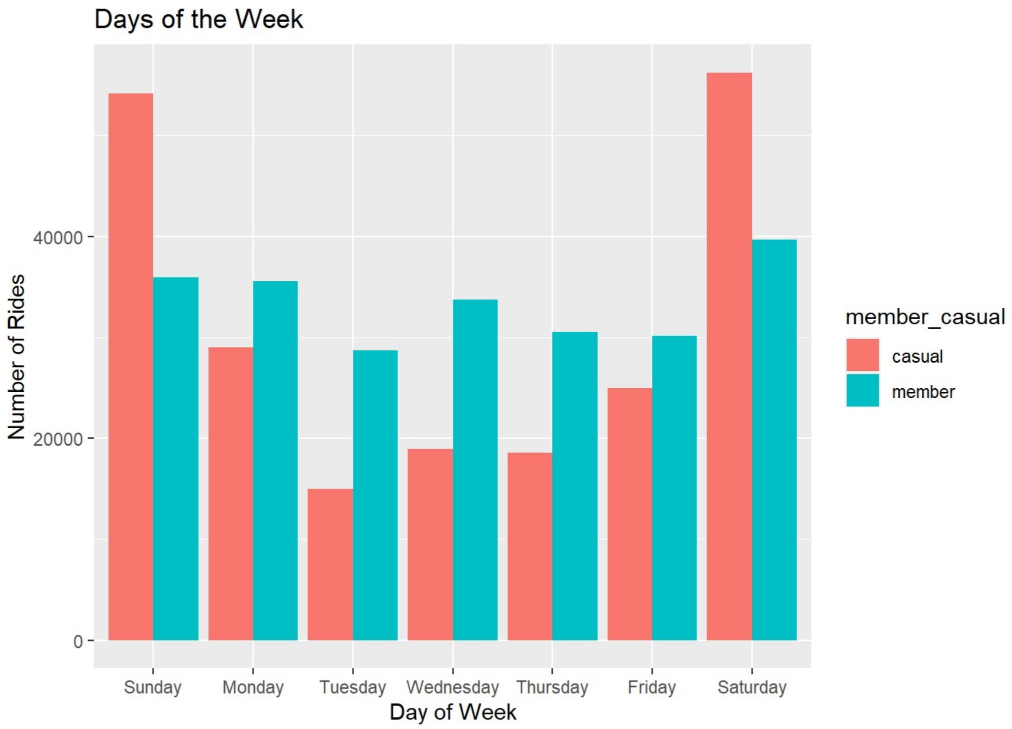
### **STEP FIVE:** VISUALIZATION

Display full digits instead of scientific number.

```
options(scipen=999)
```

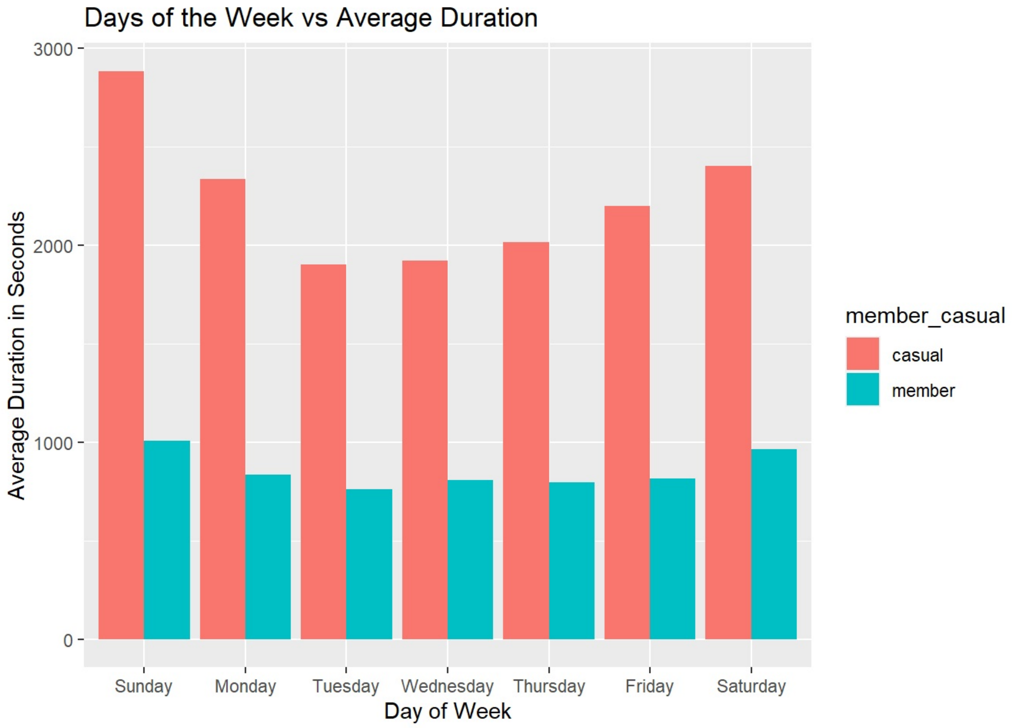
Plot the number of rides by user type during the week.

```
May21 %>% 
  mutate(day_of_week) %>% 
  group_by(member_casual,day_of_week) %>% 
  summarise(number_of_rides = n(), average_duration = mean(ride_length), .groups = 'drop') %>% 
  arrange(member_casual, day_of_week)  %>% 
  ggplot(aes(x = day_of_week, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge")+
labs(x = "Day of Week",
     y= "Number of Rides",
     title= "Days of the Week")
```



Plot the duration of the ride by user type during the week.

```
May21 %>%
  mutate(day_of_week) %>%
  group_by(member_casual, day_of_week) %>%
  summarise(number_of_rides = n(), average_duration = mean(ride_length), .groups = 'drop') %>%
  arrange(member_casual, day_of_week)  %>%
  ggplot(aes(x = day_of_week, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(x = "Day of Week",
       y= "Average Duration in Seconds",
       title= "Days of the Week vs Average Duration")
```



Create new dataframe for plots for weekday trends vs weekend trends.

```
mc<- as.data.frame(table(May21$day_of_week,May21$member_casual))
```

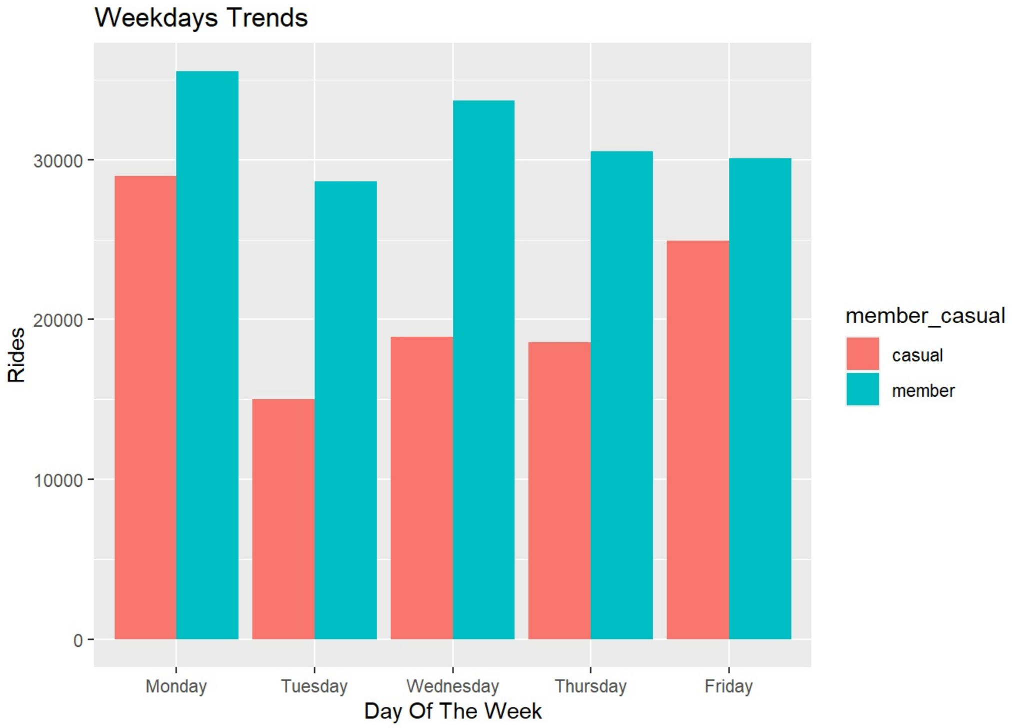
Rename columns

```
mc<-rename(mc, day_of_week = Var1, member_casual = Var2)
head(mc)
```

```
##   day_of_week member_casual  Freq
## 1      Sunday        casual 54125
## 2      Monday        casual 29003
## 3     Tuesday        casual 14990
## 4   Wednesday        casual 18926
## 5    Thursday        casual 18585
## 6      Friday        casual 24956
```

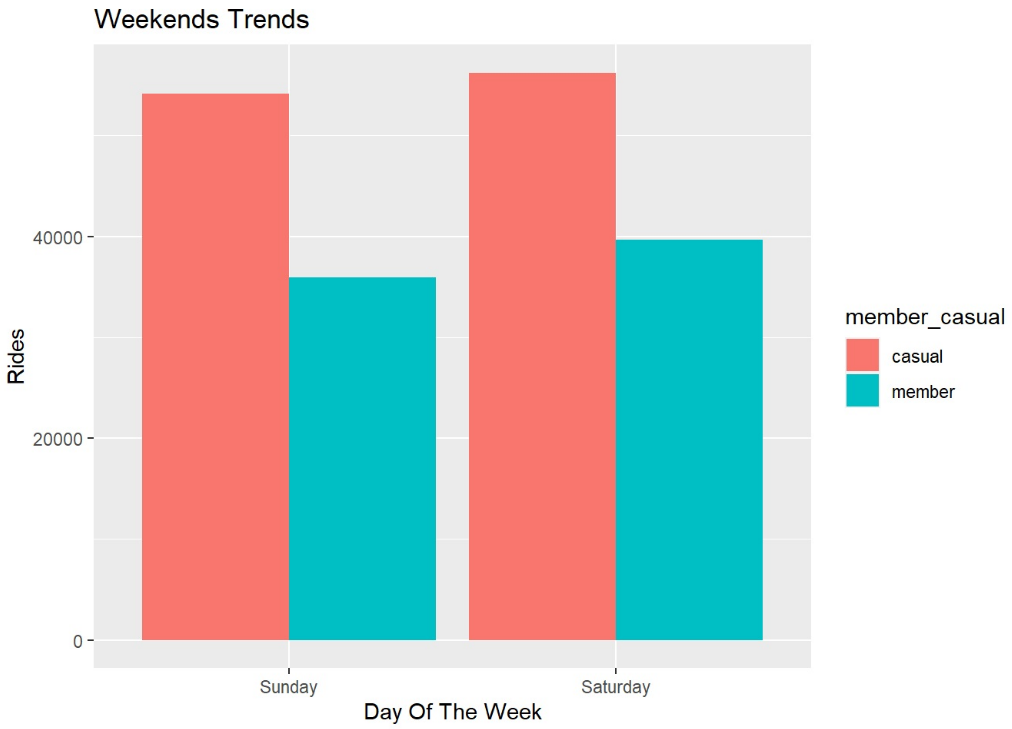
Weekday trends (Monday through Friday).

```
mc %>%
  filter(day_of_week == "Monday" |
           day_of_week == "Tuesday" |
           day_of_week == "Wednesday" |
           day_of_week == "Thursday" |
           day_of_week == "Friday") %>%
  ggplot(aes(x = day_of_week, y = Freq, fill = member_casual))+
  geom_bar(stat = "identity" , position = "dodge") +
  labs(title = "Weekdays Trends",
       x= "Day Of The Week",
       y = "Rides")
```



Weekend trends (Sunday and Saturday).

```
mc %>%
  filter(day_of_week == "Sunday" |
           day_of_week == "Saturday") %>%
  ggplot(aes(x = day_of_week, y = Freq, fill = member_casual))+
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Weekends Trends",
       x= "Day Of The Week",
       y = "Rides")
```



Create dataframe for member and casual riders vs ride type

```
rt<- as.data.frame(table(May21$rideable_type,May21$member_casual))
```

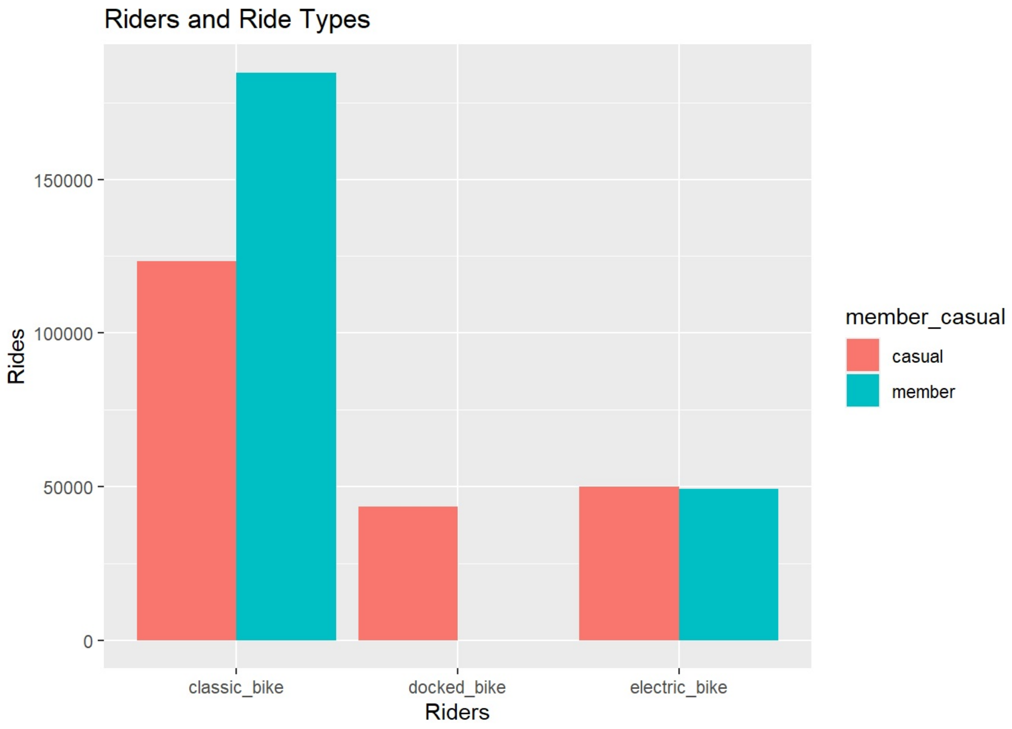
Rename columns.

```
rt<-rename(rt, rideable_type = Var1, member_casual = Var2)
head(rt)
```

```
##   rideable_type member_casual   Freq
## 1  classic_bike        casual 123455
## 2   docked_bike        casual  43351
## 3 electric_bike        casual  50001
## 4  classic_bike        member 184805
## 5   docked_bike        member      0
## 6 electric_bike        member  49294
```

Plot for bike user vs bike type.

```
rt %>%
  filter(member_casual == "member" |
           member_casual == "casual") %>%
  ggplot(aes(x = rideable_type, y = Freq, fill = member_casual))+
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Riders and Ride Types",
       x= "Riders",
       y = "Rides")
```



**STEP SIX:** EXPORT ANALYZED DATA

Save the analyzed data as a new file. fwrite(May21, “May21.csv”)