Case Study Cyclistic

This project is the final project of the Google Data Analytics Professional Certification Course. This case study analyzes a public dataset of a fictional company provided in the course. For this analysis, R programming language will be used. R provides simple statistical analysis tools and data visualization.

Business case

You are a junior data analyst on the marketing analyst team at Cyclistic, a Chicago-based bike-share company. The marketing director believes that increasing the number of annual memberships is critical to the company's future success. As a result, your team is interested in learning how casual riders and annual members use Cyclistic bikes differently. Your team will develop a new marketing strategy based on these findings in order to convert casual riders into annual members. However, Cyclistic executives must first approve your recommendations, which must be supported by compelling data insights and professional data visualizations.

The following data analysis steps will be followed to address the business case:

- Ask
- Prepare
- Process
- Analyze
- Share
- Act

Ask

The future marketing program will be guided by three questions:

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

The man purpose of this analysis is to answer the first question that has been assigned to us by Lily Moreno (the director of marketing team): How do annual members and casual riders use Cyclistic bikes differently?

The business task

The main objective is to design marketing strategy aimed at converting casual riders to annual members by understanding how they differ.

Key stakeholders are: Director of Marketing (Lily Moreno), Marketing Analytics team, Executive team.

Prepare

Cyclistic's historical trip data to analyze and identify trends will be used in this study. Data is available to download from here

This is public data that has been made available by Motivate International Inc. under the following license

Key tasks:

- Download data and properly store it.
- Data has been downloaded, and copies have been securely stored on local computer.
- Determine how it is organized.
- The data is in CSV (comma-separated values) format, with 13 columns in total.
- Sort and filter the information.
- I will use data from 2022 (January-December) for this analysis because it is more current.
- Determine the data's credibility.
- The datasets are appropriate for this case study and will allow me to answer the business questions. However, data-privacy concerns will prevent me from using riders' personal information, preventing me from determining whether riders have purchased multiple single passes.

Steps in RStudio:

Loading necessary packages

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   1.0.1
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x 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(ggplot2)
library(dplyr)
```

Importing data into R

```
jan22 <- read_csv ("202201-divvy-tripdata.csv")
```

```
## Rows: 103770 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.
```

```
feb22 <- read_csv ("202202-divvy-tripdata.csv")
```

```
## Rows: 115609 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.
```

```
mar22 <- read_csv ("202203-divvy-tripdata.csv")
```

```
## Rows: 284042 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.
```

```
apr22 <- read_csv ("202204-divvy-tripdata.csv")
```

```
## Rows: 371249 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.
```

```
may22 <- read_csv ("202205-divvy-tripdata.csv")
```

```
## Rows: 634858 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.
```

```
jun22 <- read_csv ("202206-divvy-tripdata.csv")
```

```
## Rows: 769204 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.
```

```
jul22 <- read_csv ("202207-divvy-tripdata.csv")
```

```
## Rows: 823488 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.
```

```
aug22 <- read_csv ("202208-divvy-tripdata.csv")
```

```
## Rows: 785932 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.
```

```
sep22 <- read_csv ("202209-divvy-tripdata.csv")
```

```
## Rows: 701339 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.
```

```
oct22 <- read_csv ("202210-divvy-tripdata.csv")
```

```
## Rows: 558685 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.
```

```
nov22 <- read_csv ("202211-divvy-tripdata.csv")
```

```
## Rows: 337735 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.
```

```
dec22 <- read_csv ("202212-divvy-tripdata.csv")
```

```
## Rows: 181806 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.
```

I ensured that all 12 data sets have the same number of columns and same column names before going forward to merge them. I also checked for any inconsistencies within the data.

Merging 12 data frames into one data frame - trips_data:

```
trips_data <- bind_rows(jan22, feb22, mar22, apr22, may22, jun22, jul22, aug22, sep22, oct22, nov22, de
```

Process

To inspect new data frame I used glimpse function, which gives column names, number of rows and data type for each column. It also allows to see how each data entry looks in a data frame.

```
glimpse(trips_data)
```

```
## Rows: 5,667,717
## Columns: 13
## $ ride_id            <chr> "C2F7DD78E82EC875", "A6CF8980A652D272", "BD0F91DFF7~
## $ rideable_type      <chr> "electric_bike", "electric_bike", "classic_bike", "~
## $ started_at         <dttm> 2022-01-13 11:59:47, 2022-01-10 08:41:56, 2022-01-~
## $ ended_at           <dttm> 2022-01-13 12:02:44, 2022-01-10 08:46:17, 2022-01-~
## $ start_station_name <chr> "Glenwood Ave & Touhy Ave", "Glenwood Ave & Touhy A~
## $ start_station_id   <chr> "525", "525", "TA1306000016", "KA1504000151", "TA13~
## $ end_station_name   <chr> "Clark St & Touhy Ave", "Clark St & Touhy Ave", "Gr~
## $ end_station_id     <chr> "RP-007", "RP-007", "TA1307000001", "TA1309000021",~
## $ start_lat          <dbl> 42.01280, 42.01276, 41.92560, 41.98359, 41.87785, 4~
## $ start_lng          <dbl> -87.66591, -87.66597, -87.65371, -87.66915, -87.624~
## $ end_lat            <dbl> 42.01256, 42.01256, 41.92533, 41.96151, 41.88462, 4~
## $ end_lng            <dbl> -87.67437, -87.67437, -87.66580, -87.67139, -87.627~
## $ member_casual      <chr> "casual", "casual", "member", "casual", "member", "~
```

To ensure that all data has been imported I used head and tail functions.

```
head(trips_data)
```

```
## # A tibble: 6 x 13
##   ride_id        ridea~1 started_at          ended_at            start~2 start~3
##   <chr>          <chr>   <dttm>              <dttm>              <chr>   <chr>
## 1 C2F7DD78E82EC~ electr~ 2022-01-13 11:59:47 2022-01-13 12:02:44 Glenwo~ 525
## 2 A6CF8980A652D~ electr~ 2022-01-10 08:41:56 2022-01-10 08:46:17 Glenwo~ 525
## 3 BD0F91DFF741C~ classi~ 2022-01-25 04:53:40 2022-01-25 04:58:01 Sheffi~ TA1306~
## 4 CBB80ED419105~ classi~ 2022-01-04 00:18:04 2022-01-04 00:33:00 Clark ~ KA1504~
## 5 DDC963BFDDA51~ classi~ 2022-01-20 01:31:10 2022-01-20 01:37:12 Michig~ TA1309~
## 6 A39C6F6CC0586~ classi~ 2022-01-11 18:48:09 2022-01-11 18:51:31 Wood S~ 637
## # ... 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 1: rideable_type,
## #   2: start_station_name, 3: start_station_id
```

```
tail(trips_data)
```

```
## # A tibble: 6 x 13
##   ride_id        ridea~1 started_at          ended_at            start~2 start~3
##   <chr>          <chr>   <dttm>              <dttm>              <chr>   <chr>
## 1 7BDEDE9860418~ classi~ 2022-12-07 06:52:45 2022-12-07 06:56:36 Sangam~ 13409
## 2 43ABEE85B6E15~ classi~ 2022-12-05 06:51:04 2022-12-05 06:54:48 Sangam~ 13409
## 3 F041C89A3D1F0~ electr~ 2022-12-14 17:06:28 2022-12-14 17:19:27 Bernar~ 18016
## 4 A2BECB88430BE~ classi~ 2022-12-08 16:27:47 2022-12-08 16:32:20 Wacker~ KA1503~
## 5 37B392960E566~ classi~ 2022-12-28 09:37:38 2022-12-28 09:41:34 Sangam~ 13409
## 6 2DD1587210BA4~ classi~ 2022-12-09 00:27:25 2022-12-09 00:35:28 Southp~ 13235
```

```
## # ... 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 1: rideable_type,
## #   2: start_station_name, 3: start_station_id
```

To get statistical summary of data I used summary function

```
summary(trips_data)
```

```
##    ride_id          rideable_type        started_at
##  Length:5667717     Length:5667717     Min.   :2022-01-01 00:00:05.00
##  Class :character   Class :character   1st Qu.:2022-05-28 19:21:05.00
##  Mode  :character   Mode  :character   Median :2022-07-22 15:03:59.00
##                                        Mean   :2022-07-20 07:21:18.74
##                                        3rd Qu.:2022-09-16 07:21:29.00
##                                        Max.   :2022-12-31 23:59:26.00
##
##     ended_at                      start_station_name start_station_id
##  Min.   :2022-01-01 00:01:48.00   Length:5667717     Length:5667717
##  1st Qu.:2022-05-28 19:43:07.00   Class :character   Class :character
##  Median :2022-07-22 15:24:44.00   Mode  :character   Mode  :character
##  Mean   :2022-07-20 07:40:45.33
##  3rd Qu.:2022-09-16 07:39:03.00
##  Max.   :2023-01-02 04:56:45.00
##
##  end_station_name   end_station_id       start_lat       start_lng
##  Length:5667717     Length:5667717     Min.   :41.64   Min.   :-87.84
##  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   :-87.65
##                                        3rd Qu.:41.93   3rd Qu.:-87.63
##                                        Max.   :45.64   Max.   :-73.80
##
##     end_lat         end_lng       member_casual
##  Min.   : 0.00   Min.   :-88.14   Length:5667717
##  1st Qu.:41.88   1st Qu.:-87.66   Class :character
##  Median :41.90   Median :-87.64   Mode  :character
##  Mean   :41.90   Mean   :-87.65
##  3rd Qu.:41.93   3rd Qu.:-87.63
##  Max.   :42.37   Max.   :  0.00
##  NA's   :5858    NA's   :5858
```

Date and time are currently in one column, but for our analysis it is better to have date, month, year, day of the week in separate columns so that we can identify trends/patterns. The following chunk of code was used to achieve that:

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

To confirm that additional columns were added into the data frame I used colnames function

```
colnames(trips_data)
```

```
##  [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"      "date"               "month"
## [16] "day"                "year"               "day_of_week"
```

I also want to add ride_length calculation to our data frame. For this I will be using difftime function.

```
trips_data$ride_length <- difftime(trips_data$ended_at, trips_data$started_at)
```

Let's see what is the structure of our data at this stage.

```
str(trips_data)
```

```
## spc_tbl_ [5,667,717 x 19] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:5667717] "C2F7DD78E82EC875" "A6CF8980A652D272" "BD0F91DFF741C66D" "CBB8
##  $ rideable_type     : chr [1:5667717] "electric_bike" "electric_bike" "classic_bike" "classic_bike"
##  $ started_at        : POSIXct[1:5667717], format: "2022-01-13 11:59:47" "2022-01-10 08:41:56" ...
##  $ ended_at          : POSIXct[1:5667717], format: "2022-01-13 12:02:44" "2022-01-10 08:46:17" ...
##  $ start_station_name: chr [1:5667717] "Glenwood Ave & Touhy Ave" "Glenwood Ave & Touhy Ave" "Sheffi
##  $ start_station_id  : chr [1:5667717] "525" "525" "TA1306000016" "KA1504000151" ...
##  $ end_station_name  : chr [1:5667717] "Clark St & Touhy Ave" "Clark St & Touhy Ave" "Greenview Ave &
##  $ end_station_id    : chr [1:5667717] "RP-007" "RP-007" "TA1307000001" "TA1309000021" ...
##  $ start_lat         : num [1:5667717] 42 42 41.9 42 41.9 ...
##  $ start_lng         : num [1:5667717] -87.7 -87.7 -87.7 -87.7 -87.6 ...
##  $ end_lat           : num [1:5667717] 42 42 41.9 42 41.9 ...
##  $ end_lng           : num [1:5667717] -87.7 -87.7 -87.7 -87.7 -87.6 ...
##  $ member_casual     : chr [1:5667717] "casual" "casual" "member" "casual" ...
##  $ date              : Date[1:5667717], format: "2022-01-13" "2022-01-10" ...
##  $ month             : chr [1:5667717] "01" "01" "01" "01" ...
##  $ day               : chr [1:5667717] "13" "10" "25" "04" ...
##  $ year              : chr [1:5667717] "2022" "2022" "2022" "2022" ...
##  $ day_of_week       : chr [1:5667717] "Thursday" "Monday" "Tuesday" "Tuesday" ...
##  $ ride_length       : 'difftime' num [1:5667717] 177 261 261 896 ...
##   ..- attr(*, "units")= chr "secs"
##  - 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>
```

In order to run calculations we need to convert ride_length from factor to numeric.I will use as.numeric function.

```
trips_data$ride_length <- as.numeric(as.character(trips_data$ride_length))
```

Let's see if it was converted correctly.

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

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

We can now add ride_distance calculation to our data frame.

```
library(geosphere)

trips_data$ride_distance <- distGeo(matrix(c(trips_data$start_lng, trips_data$start_lat), ncol=2), matri
```

Let's also convert ride_distance to km

```
trips_data$ride_distance <- trips_data$ride_distance/1000
```

There are entries in our data frame where ride length is negative or zero due to bikes being taken by Divvy for quality checks. Let's clean this.

```
trips_data_clean <- trips_data[!(trips_data$ride_length <= 0),]
```

Let's inspect data again

```
glimpse(trips_data_clean)
```

```
## Rows: 5,667,186
## Columns: 20
## $ ride_id            <chr> "C2F7DD78E82EC875", "A6CF8980A652D272", "BD0F91DFF7~
## $ rideable_type      <chr> "electric_bike", "electric_bike", "classic_bike", "~
## $ started_at         <dttm> 2022-01-13 11:59:47, 2022-01-10 08:41:56, 2022-01-~
## $ ended_at           <dttm> 2022-01-13 12:02:44, 2022-01-10 08:46:17, 2022-01-~
## $ start_station_name <chr> "Glenwood Ave & Touhy Ave", "Glenwood Ave & Touhy A~
## $ start_station_id   <chr> "525", "525", "TA1306000016", "KA1504000151", "TA13~
## $ end_station_name   <chr> "Clark St & Touhy Ave", "Clark St & Touhy Ave", "Gr~
## $ end_station_id     <chr> "RP-007", "RP-007", "TA1307000001", "TA1309000021",~
## $ start_lat          <dbl> 42.01280, 42.01276, 41.92560, 41.98359, 41.87785, 4~
## $ start_lng          <dbl> -87.66591, -87.66597, -87.65371, -87.66915, -87.624~
## $ end_lat            <dbl> 42.01256, 42.01256, 41.92533, 41.96151, 41.88462, 4~
## $ end_lng            <dbl> -87.67437, -87.67437, -87.66580, -87.67139, -87.627~
## $ member_casual      <chr> "casual", "casual", "member", "casual", "member", "~
## $ date               <date> 2022-01-13, 2022-01-10, 2022-01-25, 2022-01-04, 20~
```

```
## $ month              <chr> "01", "01", "01", "01", "01", "01", "01", "01", "01~
## $ day                <chr> "13", "10", "25", "04", "20", "11", "30", "22", "17~
## $ year               <chr> "2022", "2022", "2022", "2022", "2022", "2022", "20~
## $ day_of_week        <chr> "Thursday", "Monday", "Tuesday", "Tuesday", "Thursd~
## $ ride_length        <dbl> 177, 261, 261, 896, 362, 202, 994, 724, 1527, 443, ~
## $ ride_distance      <dbl> 0.7013791, 0.6961414, 1.0034507, 2.4601250, 0.81407~
```

Analyze

To get the summary of new data frame I use summary function again.

```
summary(trips_data_clean)
```

```
##    ride_id          rideable_type        started_at
##  Length:5667186     Length:5667186     Min.   :2022-01-01 00:00:05.00
##  Class :character   Class :character   1st Qu.:2022-05-28 19:20:00.00
##  Mode  :character   Mode  :character   Median :2022-07-22 15:01:59.50
##                                        Mean   :2022-07-20 07:19:14.76
##                                        3rd Qu.:2022-09-16 07:18:50.75
##                                        Max.   :2022-12-31 23:59:26.00
##
##     ended_at                      start_station_name start_station_id
##  Min.   :2022-01-01 00:01:48.00   Length:5667186     Length:5667186
##  1st Qu.:2022-05-28 19:41:54.25   Class :character   Class :character
##  Median :2022-07-22 15:22:49.00   Mode  :character   Mode  :character
##  Mean   :2022-07-20 07:38:41.61
##  3rd Qu.:2022-09-16 07:36:10.75
##  Max.   :2023-01-02 04:56:45.00
##
##  end_station_name   end_station_id       start_lat       start_lng
##  Length:5667186     Length:5667186     Min.   :41.64   Min.   :-87.84
##  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   :-87.65
##                                        3rd Qu.:41.93   3rd Qu.:-87.63
##                                        Max.   :45.64   Max.   :-73.80
##
##     end_lat         end_lng       member_casual           date
##  Min.   : 0.00   Min.   :-88.14   Length:5667186     Min.   :2022-01-01
##  1st Qu.:41.88   1st Qu.:-87.66   Class :character   1st Qu.:2022-05-28
##  Median :41.90   Median :-87.64   Mode  :character   Median :2022-07-22
##  Mean   :41.90   Mean   :-87.65                      Mean   :2022-07-19
##  3rd Qu.:41.93   3rd Qu.:-87.63                      3rd Qu.:2022-09-16
##  Max.   :42.37   Max.   :  0.00                      Max.   :2022-12-31
##  NA's   :5858    NA's   :5858
##     month               day                year           day_of_week
##  Length:5667186     Length:5667186     Length:5667186     Length:5667186
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
```

```
##   ride_length      ride_distance
##  Min.   :      1   Min.   :   0.000
##  1st Qu.:    349   1st Qu.:   0.873
##  Median :    617   Median :   1.575
##  Mean   :   1167   Mean   :   2.140
##  3rd Qu.:   1108   3rd Qu.:   2.781
##  Max.   :2483235   Max.   :9817.319
##                    NA's   :5858
```

To run descriptive analysis I will use mean function (to get the average of ride length), median (to get midpoint number of ride length),max(to get longest ride) and min(to get shortest ride).

```
trips_data_clean %>%
  group_by(member_casual) %>%
  summarise(average_ride_length = mean(ride_length), median_length = median(ride_length),
            max_ride_length = max(ride_length), min_ride_length = min(ride_length))
```

```
## # A tibble: 2 x 5
##   member_casual average_ride_length median_length max_ride_length min_ride_len~1
##   <chr>                       <dbl>         <dbl>           <dbl>          <dbl>
## 1 casual                      1749.           780         2483235              1
## 2 member                       763.           530           93594              1
## # ... with abbreviated variable name 1: min_ride_length
```

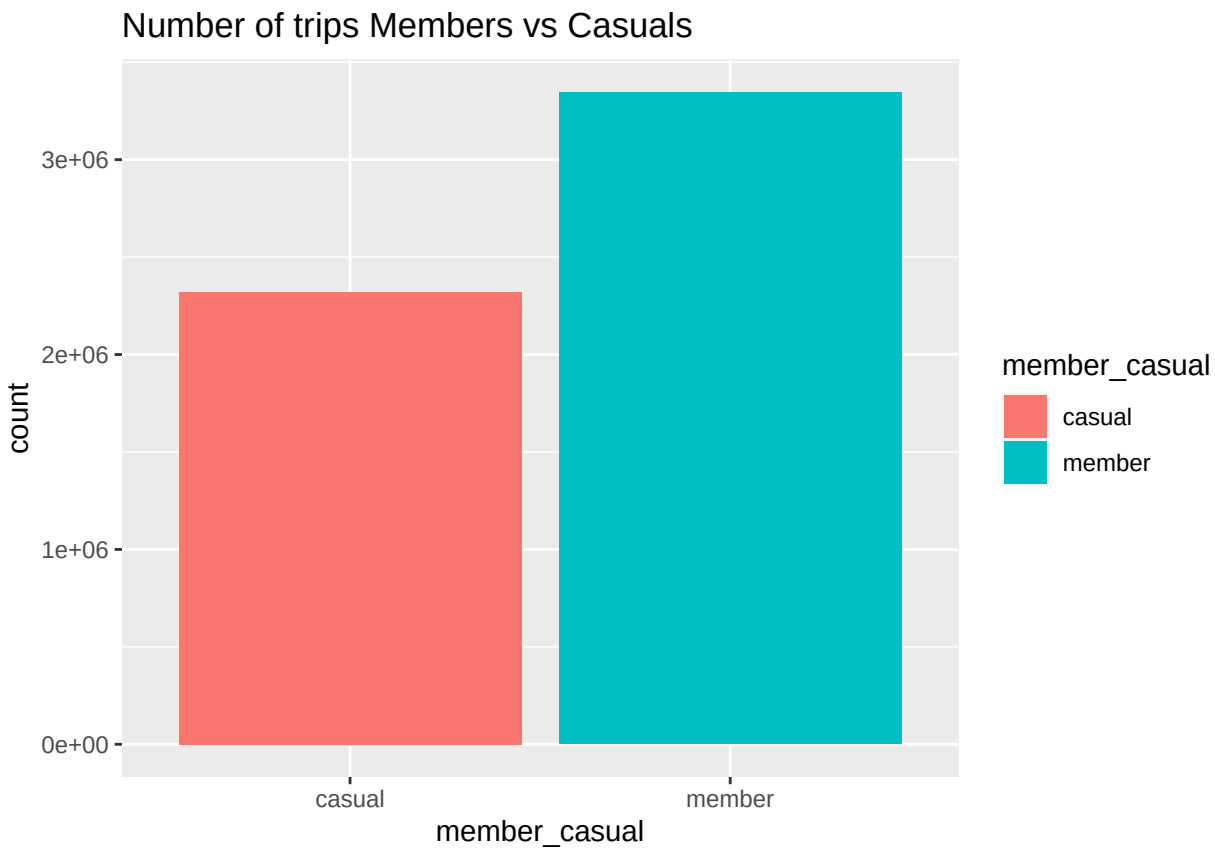
The first comparison I want to make is the number or rides taken by members vs casuals.

```
trips_data_clean %>%
  group_by(member_casual) %>%
  summarise(ride_count = length(ride_id))
```

```
## # A tibble: 2 x 2
##   member_casual ride_count
##   <chr>              <int>
## 1 casual           2321769
## 2 member           3345417
```

To visualize the above result let's use geom_bar

```
ggplot(data=trips_data_clean) + geom_bar(mapping=aes(x=member_casual, fill=member_casual)) + labs(title=
```



The graph shows that there are more member riders than casual riders based on the ride count.

Another comparison I want to make is to see total rides and average ride time by each day for members vs casual riders.

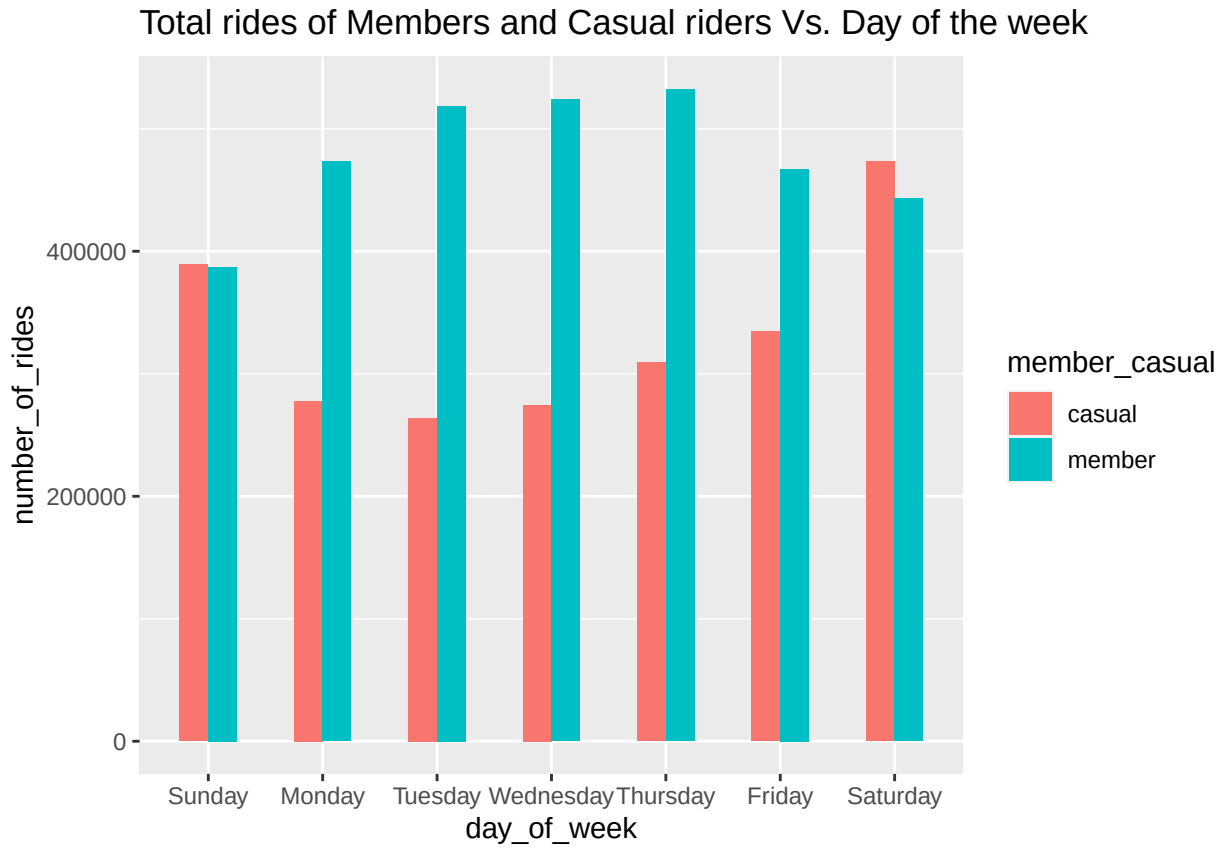
```
trips_data_clean$day_of_week <- ordered(trips_data_clean$day_of_week,
                                        levels=c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",

trips_data_clean %>%
  group_by(member_casual, day_of_week) %>%  #groups by member_casual
  summarise(number_of_rides = n() #calculates the number of rides and average duration
            ,average_ride_length = mean(ride_length),.groups="drop") %>% # calculates the average durat
  arrange(member_casual, day_of_week) #sort
```

```
## # A tibble: 14 x 4
##    member_casual day_of_week number_of_rides average_ride_length
##    <chr>         <ord>                 <int>               <dbl>
##  1 casual        Sunday               388981               2044.
##  2 casual        Monday               277649               1751.
##  3 casual        Tuesday              263706               1550.
##  4 casual        Wednesday            274339               1485.
##  5 casual        Thursday             309297               1533.
##  6 casual        Friday               334667               1683.
##  7 casual        Saturday             473130               1957.
##  8 member        Sunday               387180                842.
##  9 member        Monday               473305                736.
```

```
## 10 member        Tuesday              518584              728.
## 11 member        Wednesday            523836              726.
## 12 member        Thursday             532215              738.
## 13 member        Friday               467051              752.
## 14 member        Saturday             443246              848.
```

To visualize the above let's use geom_col.

```
trips_data_clean %>%
  group_by(member_casual, day_of_week) %>%
  summarise(number_of_rides = n(), .groups="drop") %>%
  arrange(member_casual, day_of_week)  %>%
  ggplot(aes(x = day_of_week, y = number_of_rides, fill = member_casual)) +
  labs(title ="Total rides of Members and Casual riders Vs. Day of the week") +
  geom_col(width=0.5, position = position_dodge(width=0.5)) +
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE))
```

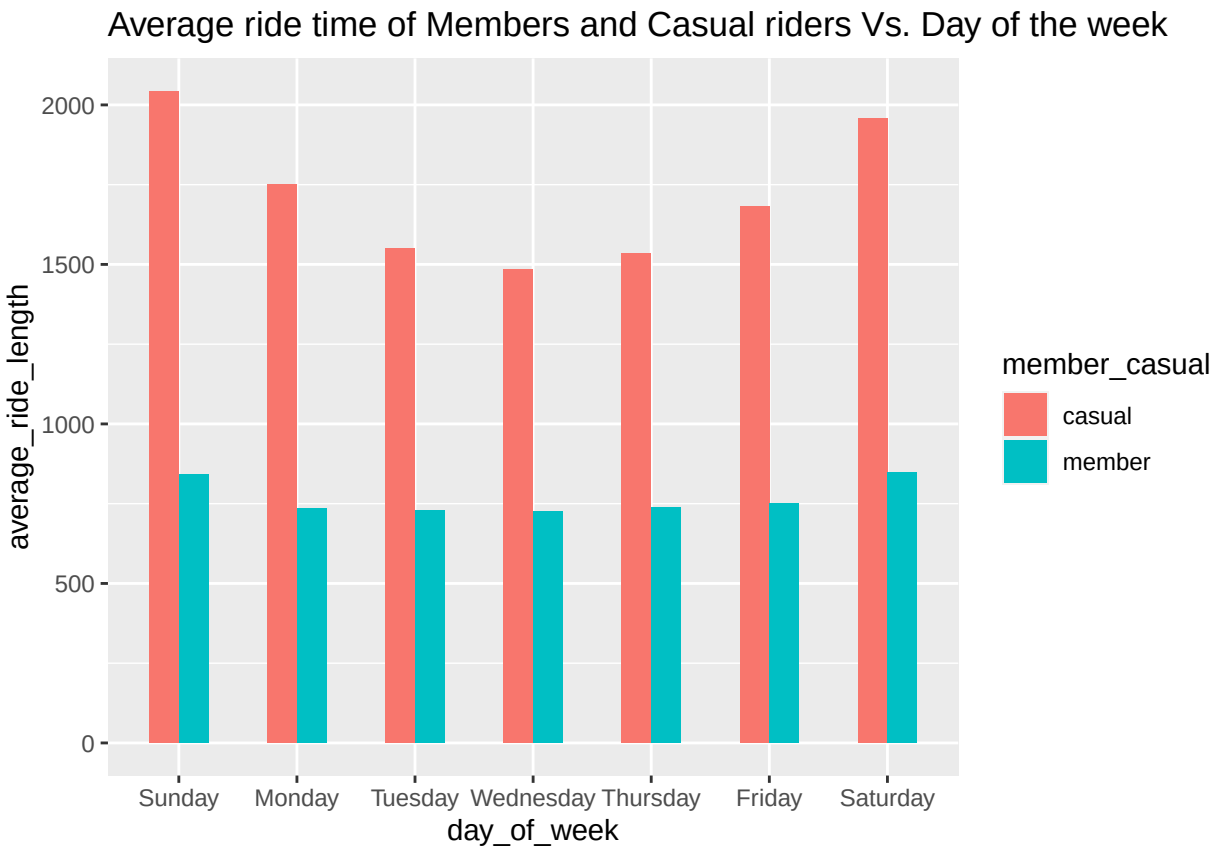


The chart shows that casual riders have the highest number of rides on Saturday, while members are quite consistent during the week, but they have the lower number of rides on the weekend.

Let's visualize the average ride by day of the week.

```
trips_data_clean %>%
    group_by(member_casual, day_of_week) %>%
    summarise(average_ride_length = mean(ride_length), .groups="drop") %>%
    ggplot(aes(x = day_of_week, y = average_ride_length, fill = member_casual)) +
```

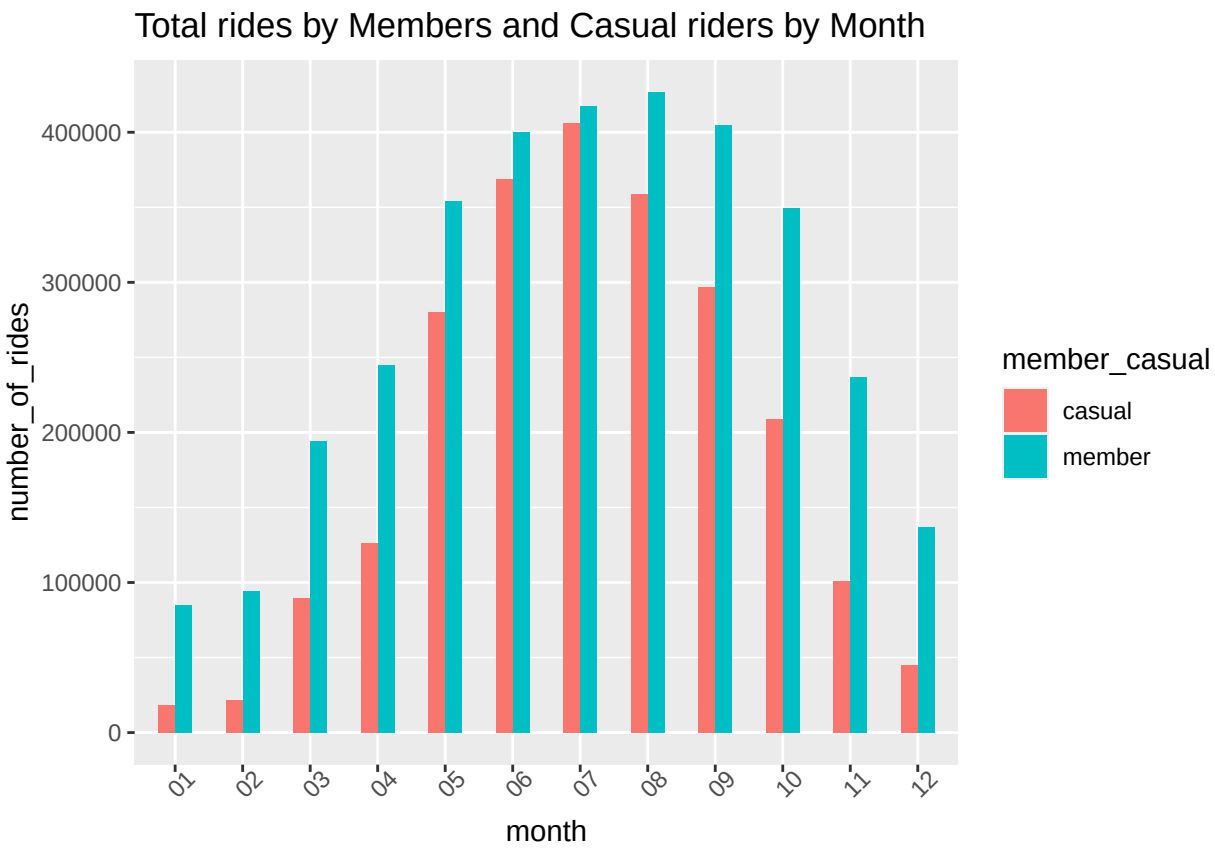
```
    geom_col(width=0.5, position = position_dodge(width=0.5)) +
    labs(title ="Average ride time of Members and Casual riders Vs. Day of the week")
```



The chart shows that casual riders ride for a longer time during the week with the highest rides on the weekends, while members ride at a consistent pace during the week with the highest rides on the weekends too.

Let's visualize the total rides taken by members and casuals by month.

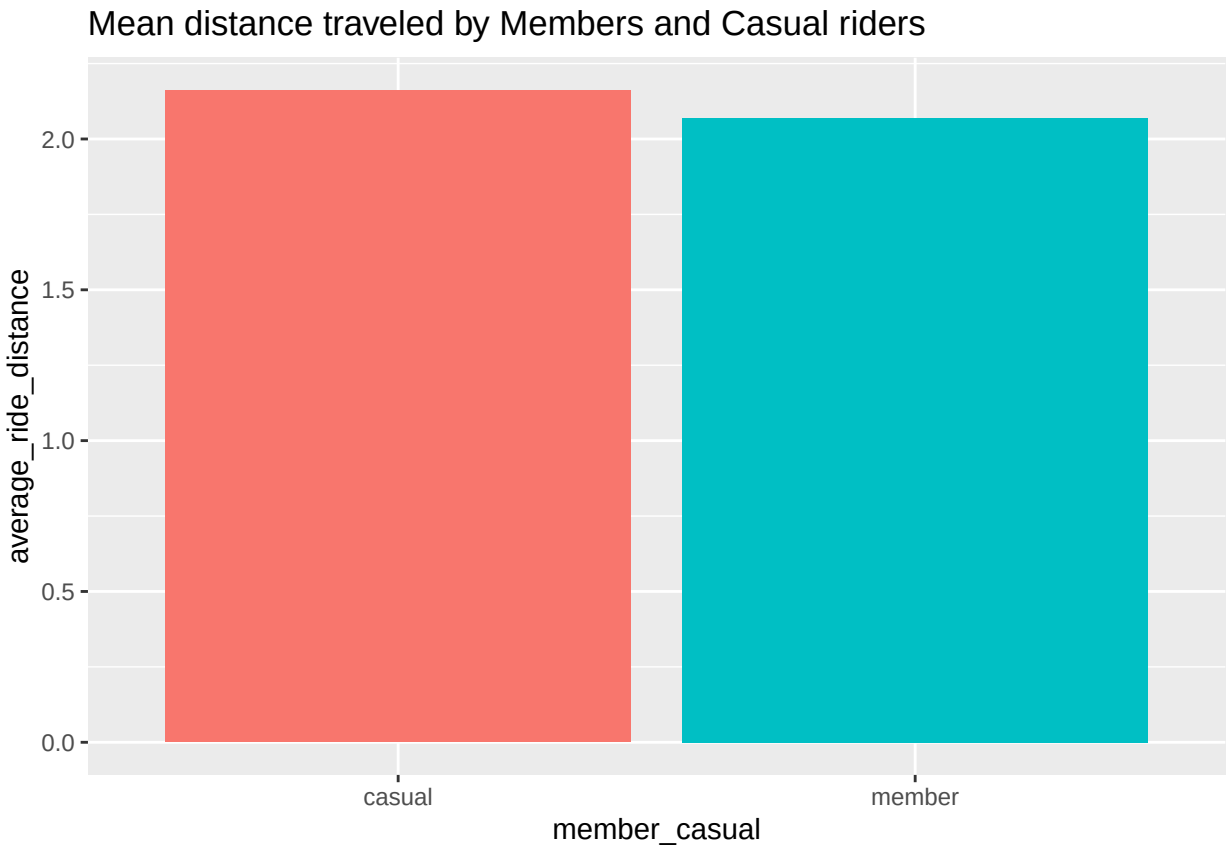
```
trips_data_clean %>%
    group_by(member_casual, month) %>%
    summarise(number_of_rides = n(),.groups="drop") %>%
    arrange(member_casual, month)  %>%
    ggplot(aes(x = month, y = number_of_rides, fill = member_casual)) +
    labs(title ="Total rides by Members and Casual riders by Month") +
    theme(axis.text.x = element_text(angle = 45)) +
    geom_col(width=0.5, position = position_dodge(width=0.5)) +
    scale_y_continuous(labels = function(x) format(x, scientific = FALSE))
```



The chart shows that that members had higher number of rides all throughout the year.The total number of rides of casual members are closer to members on warmer months May,June and July, August. July being almost equal.

Lastly, let's compare Members and Casual riders depending on ride distance

```
trips_data_clean %>%
  group_by(member_casual) %>% drop_na() %>%
  summarise(average_ride_distance = mean(ride_distance)) %>%
  ggplot() +
  geom_col(mapping= aes(x= member_casual,y= average_ride_distance,fill=member_casual), show.legend = FA
  labs(title = "Mean distance traveled by Members and Casual riders")
```



We can see that casual riders went a longer distance compared to members by a few kilometers.

Share

Conclusions

- Based on the number of rides, there are more member riders than casual riders.
- In comparison to the other days, casual riders have the most rides on Saturday, while members are quite consistent but have the fewest rides on the weekend.
- casual riders ride for a longer period of time during the week, with the most rides on weekends, whereas members ride at a consistent pace during the week, with the most rides on weekends.
- Throughout the year, members had a higher number of rides.The total number of casual rides is closer to members during the warmer months of May, June, July, and August. July is nearly equal.
- In comparison to members, casual riders traveled a few kilometers further.

Recommendations

- Hold a slash sale or promo for casual riders so they can buy more bikes and enjoy the benefits of membership.
- Encourage weekday riding by providing various coupons, offers such as free membership trials.
- Hold regular biking member competitions with prizes on weekends so that casual riders are encouraged to get membership and be able to participate in such competitions.