Cyclistic Bike Share Analysis

Aryan Juyal

2023-07-05

Introduction

Welcome to my Google Data Analytics Certificate capstone project! As I progress through the many stages of this project, I will face real-world data analyst tasks, allowing me to exhibit my knowledge, abilities, and thought process.

About the Company

- Cyclistic is a fictitious bike-sharing program with 5,824 bicycles and 692 docking points.
- While most users ride for fun, about 30% use them to travel to work every day.
- Three pricing options are available: single-ride passes, full-day passes, and annual memberships.
- Customers that purchase a single ride or a full-day pass are casual riders.
- Member riders are customers who purchase an annual membership.

Scenario

- My title at Cyclistic is Junior data analyst in the marketing analyst team.
- Annual members are far more profitable than casual riders, according to Cyclistic's financial analysts.
- The marketing director believes that increasing the number of yearly subscriptions is critical to the company's future success.
- The purpose of the marketing strategy is to turn casual riders into annual members.

A six-step data analysis process will be used in this case study: Ask, Prepare, Process, Analyze, Share, and Act.

Step 1 : Ask questions and define the problem

This step involves the important tasks of asking the proper questions in order to acquire enough preliminary information to steer the project in the right direction. To keep everyone in the loop as you move forward, make sure you grasp the business task and identify all essential stakeholders.

1.1 Asking the right questions

The future marketing program will be guided by three questions: - **How do annual members and casual users of Cyclistic bikes differ?** - Why would a casual rider purchase a Cyclistic annual membership? - How can Cyclistic use digital media to get casual riders to join?

The first question is the topic of this case study.

1.2 Identify the business task

The business task is to investigate how casual and member riders use Cyclistic differently, with the goal of developing a new marketing approach to convert casual riders into annual members.

1.3 Identify key stakeholders

- **The marketing director** is in charge of creating campaigns and initiatives to promote the bike-share program.
- **The Cyclistic executive team**, which is known for its attention to detail, will determine whether to approve the planned marketing program.
- **The Cyclistic marketing analytics team** is a group of data analysts who collect, analyse, and report data that helps steer marketing campaigns.

Step 2 : Prepare the data by collecting and storing information

This stage entails locating and retrieving the data from its current location, evaluating its integrity, credibility, and accessibility, and saving the data in its new location.

2.1 Data Location

- Divvy, a Chicago-based bike-share firm, provided the data for this research.
- All data comes from Divvy's public data link : Divvy Trip Data.

2.2 Data Organization

- The historical trip data on Divvy is organised by month and year and saved as a zip file.
- Each csv file is organised into rows and columns.

2.3 Credibility of the Data/Data Bias

- This data is Reliable, Original, Comprehensive, Current, and Cited, and it is derived from Divvy's public historical trip data. It is credible and unbiased.

2.4 Licensing, Privacy, Security, and Accessibility

- Motivate International Inc. maintains and makes the data available under this licence.
- Divvy's trip data for public use adheres to data-privacy rules and is thus anonymized and does not contain any personally identifiable information.
- Divvy's public travel data is shared on a monthly basis and is available to everybody.

2.5 Download the Data and store it appropriately

- I chose data files for the 12 months from November 2021 through October 2022 for my project.
- To keep the files organized and easy to recognize, each file was downloaded, saved as a .csv file, and consistently labelled.

2.6 Sort and Filter the data in Excel

- Excel was used to open each monthly file.
- I made a note of the number of records in each file.
- I went through each file and looked for duplicate records.
- Each file was checked for blank/NA records.
- Identified station names and ids with "test" - will go deeper in future steps to catch everything

Summary of my initial Data Review:

File Name	No of Records	duplicate ride_id	blank start_station_name	blank start_station_id	blank end_station_name	blank end_station_id	blank end_lat
202112-divvy-tripdata	247540	no	51063	51063	53498	53498	144
202111-divvy-tripdata	359978	no	75290	75290	79187	79187	191
202210-divvy-tripdata	558685	no	91355	91355	96617	96617	475
202209-divvy-tripdata	701339	no	103780	103780	111185	111185	712
202208-divvy-tripdata	785932	no	112037	112037	120522	120522	843
202207-divvy-tripdata	823488	no	112031	112031	120951	120951	947
202206-divvy-tripdata	769204	no	92944	92944	100152	100152	1055
202205-divvy-tripdata	634858	no	86704	86704	93171	93171	722
202204-divvy-tripdata	371249	no	70877	70877	75288	75288	317
202203-divvy-tripdata	284042	no	47246	47246	51157	51157	266
202202-divvy-tripdata	115609	no	18580	18580	20355	20355	77
202201-divvy-tripdata	103770	no	16260	16260	17927	17927	86
Total	**5755694**	**0**	**878167**	**878167**	**940010**	**940010**	**5835**

Figure 1: Data Review Summary

blank end_lng	start_station_name "Hubbard Warehouse"	start_station_id "TEST"	end_station_name labeled "Hubbard Warehouse"	end_station_id "TEST"
144	9	9	28	28
191	14	14	25	25
475	22	22	19	19
712	288	288	23	23
843	417	417	43	43
947	418	418	46	46
1055	321	321	26	26
722	170	170	20	20
317	190	190	16	16
266	108	108	11	11
77	53	53	9	9
86	14	14	15	15
5835	**2024**	**2024**	**281**	**281**

Figure 2: Data Review Summary

2.7 Utilize R and RStudio

Because the files are too vast to handle in Excel, I decided to continue working on this project with the R programming language and RStudio.

2.8 Install and load R packages to use for this project

- tidyverse for data import and wrangling
- lubridate for date functions
- ggplot for visualization
- skimr provides a broad overview of a data frame

Install the packages

```
install.packages("tidyverse")
```

```
## Error in install.packages : Updating loaded packages
```

```
install.packages("lubridate")
```

```
## Error in install.packages : Updating loaded packages
```

```
install.packages("ggplot2")
```

```
## Error in install.packages : Updating loaded packages
```

```
install.packages("skimr")
```

```
## Error in install.packages : Updating loaded packages
```

Load the packages

```
library(tidyverse)
```

```
library(lubridate)
```

```
library(ggplot2)
library(skimr)
```

2.9 Upload the data files and create data frames

Data frames are the beginning point for data analysis in R, therefore I'll read the 12 csv files I've uploaded to RStudio and label them "data1", "data2", "data3",... "data12". By referring to my initial summary table, where I noted the row and column counts for data frame, I can immediately confirm the amount of rows and columns for each file.

```
data1 <- read_csv("Cyclistic_bike/202111.csv")
```

```
## Rows: 359978 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

```
data2 <- read_csv("Cyclistic_bike/202112.csv")
```

```
## Rows: 247540 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

```
data3 <- read_csv("Cyclistic_bike/202201.csv")
```

```
## Rows: 103770 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

```
data4 <- read_csv("Cyclistic_bike/202202.csv")
```

```
## Rows: 115609 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

```
data5 <- read_csv("Cyclistic_bike/202203.csv")
```

```
## Rows: 284042 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

```
data6 <- read_csv("Cyclistic_bike/202204.csv")
```

```
## Rows: 371249 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

```
data7 <- read_csv("Cyclistic_bike/202205.csv")
```

```
## Rows: 634858 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

```
data8 <- read_csv("Cyclistic_bike/202206.csv")
```

```
## Rows: 769204 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

```
data9 <- read_csv("Cyclistic_bike/202207.csv")
```

```
## Rows: 823488 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

```
data10 <- read_csv("Cyclistic_bike/202208.csv")
```

```
## Rows: 785932 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
```

```
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

```
data11 <- read_csv("Cyclistic_bike/202209.csv")
```

```
## Rows: 701339 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

```
data12 <- read_csv("Cyclistic_bike/202210.csv")
```

```
## Rows: 558685 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_station_name, end_statio
## 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.
```

2.10 Check for column consistency in all 12 dataframes

Use colnames() on each new data frame to make sure all have the same 13 columns.

```
colnames(data1)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"         "ended_at"           "start_stati
##  [7] "end_station_name"   "end_station_id"     "start_lat"          "start_lng"          "end_lat"
## [13] "member_casual"
```

2.11 Combine 12 data frames into one data frame

Use rbind to combine the 12 data frames into one data frame and name it, bike_rides

```
bike_rides <- rbind(data1, data2, data3, data4, data5, data6, data7, data8, data9, data10, data11, data
```

I used the function rm() to delete the 12 individual data frames from the environment after aggregating the 12 data frames into one data frame to free up RAM.

```
rm(data1, data2, data3, data4, data5, data6, data7, data8, data9, data10, data11, data12)
```

2.12 Inspect the new data frame

To check that my data has remained intact up to this point, I'll utilize class(), dim(), colnames(), and colSums(is.na()). At this point, I'll use my Excel observations to corroborate the following:

My new dataset has been identified as a data frame with 57,55,694 rows of data.

- 13 data columns
- There are 8,78,177 occurrences when the start_station_name and start_station_id are blank.
- There are 9,40,010 occurrences when the end_station_name and end_station_id are blank respectively.
- There are 5,835 cases when end_lat and end_lng are both blank.

Confirm that the dataset is a data frame

```
class(bike_rides)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

Confirm the no. of rows and columns

```
dim(bike_rides)
```

```
## [1] 5755694      13
```

Confirm the column names

```
colnames(bike_rides)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"         "ended_at"           "start_stati
##  [7] "end_station_name"   "end_station_id"     "start_lat"          "start_lng"          "end_lat"
## [13] "member_casual"
```

Confirm the blank data_fields

```
colSums(is.na(bike_rides))
```

```
##            ride_id      rideable_type         started_at           ended_at start_station_name   stai
##                  0                  0                  0                  0             878177
##     end_station_id          start_lat          start_lng            end_lat            end_lng      r
##             940010                  0                  0               5835               5835
```

2.13 Begin Exploring the data

Understanding the data is a critical stage that should not be overlooked. It is critical to constantly provide enough time to mentally process the facts. Understand the data structure, the data types, the parameters, the dimensions, the variables, the properties of those variables, and so on.

Let's continue on with these additional functions: select(), n(row), ncol(), length(), head(), tail(), glimpse(), str(), summary(), names(), rownames(), skim_without_charts(), View()

```
select(bike_rides)
```

```
## # A tibble: 5,755,694 x 0
```

```
nrow(bike_rides)
```

```
## [1] 5755694
```

```
ncol(bike_rides)
```

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

```
length(bike_rides)
```

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

```
head(bike_rides)
```

```
## # A tibble: 6 x 13
##   ride_id     rideable_type started_at          ended_at            start_station_name start_station_
##   <chr>       <chr>         <dttm>              <dttm>              <chr>              <chr>
## 1 7C00A93E10~ electric_bike 2021-11-27 13:27:38 2021-11-27 13:46:38 <NA>               <NA>
## 2 90854840DF~ electric_bike 2021-11-27 13:38:25 2021-11-27 13:56:10 <NA>               <NA>
## 3 0A7D10CDD1~ electric_bike 2021-11-26 22:03:34 2021-11-26 22:05:56 <NA>               <NA>
## 4 2F3BE33085~ electric_bike 2021-11-27 09:56:49 2021-11-27 10:01:50 <NA>               <NA>
## 5 D67B4781A1~ electric_bike 2021-11-26 19:09:28 2021-11-26 19:30:41 <NA>               <NA>
## 6 02F85C2C3C~ electric_bike 2021-11-26 18:34:07 2021-11-26 18:52:49 Michigan Ave & Oa~ 13042
## # i 4 more variables: start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>
```

```
tail(bike_rides)
```

```
## # A tibble: 6 x 13
##   ride_id     rideable_type started_at          ended_at            start_station_name start_station_
##   <chr>       <chr>         <dttm>              <dttm>              <chr>              <chr>
## 1 DA551F0A9C~ classic_bike  2022-10-24 17:45:38 2022-10-24 17:48:02 Sedgwick St & Nor~ TA1307000038
## 2 BC3BFA659C~ classic_bike  2022-10-30 01:41:29 2022-10-30 01:57:16 Clifton Ave & Arm~ TA1307000163
## 3 ACD6545029~ classic_bike  2022-10-30 01:41:54 2022-10-30 01:57:09 Clifton Ave & Arm~ TA1307000163
## 4 4AAC03D143~ classic_bike  2022-10-15 09:34:11 2022-10-15 10:03:21 Sedgwick St & Nor~ TA1307000038
## 5 8E6F3F2978~ classic_bike  2022-10-09 10:21:34 2022-10-09 10:43:45 Sedgwick St & Nor~ TA1307000038
## 6 8D14CBE672~ docked_bike   2022-10-22 13:17:13 2022-10-22 13:46:14 Clark St & Armita~ 13146
## # i 4 more variables: start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>
```

```r
glimpse(bike_rides)
```

```
## Rows: 5,755,694
## Columns: 13
## $ ride_id            <chr> "7C00A93E10556E47", "90854840DFD508BA", "0A7D10CDD144061C", "2F3BE33085BC
## $ rideable_type      <chr> "electric_bike", "electric_bike", "electric_bike", "electric_bike", "elec
## $ started_at         <dttm> 2021-11-27 13:27:38, 2021-11-27 13:38:25, 2021-11-26 22:03:34, 2021-11-2
## $ ended_at           <dttm> 2021-11-27 13:46:38, 2021-11-27 13:56:10, 2021-11-26 22:05:56, 2021-11-2
## $ start_station_name <chr> NA, NA, NA, NA, NA, "Michigan Ave & Oak St", NA, NA, NA, NA, NA, NA, NA, N
## $ start_station_id   <chr> NA, NA, NA, NA, NA, "13042", NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N
## $ end_station_name   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
## $ end_station_id     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
## $ start_lat          <dbl> 41.93000, 41.96000, 41.96000, 41.94000, 41.90000, 41.90086, 41.81000, 41.9
## $ start_lng          <dbl> -87.72000, -87.70000, -87.70000, -87.79000, -87.63000, -87.62379, -87.600
## $ end_lat            <dbl> 41.96000, 41.92000, 41.96000, 41.93000, 41.88000, 41.90000, 41.80000, 41.9
## $ end_lng            <dbl> -87.73000, -87.70000, -87.70000, -87.79000, -87.62000, -87.63000, -87.600
## $ member_casual      <chr> "casual", "casual", "casual", "casual", "casual", "casual", "casual", "cas
```

```r
str(bike_rides)
```

```
## spc_tbl_ [5,755,694 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:5755694] "7C00A93E10556E47" "90854840DFD508BA" "0A7D10CDD144061C" "2F3
##  $ rideable_type     : chr [1:5755694] "electric_bike" "electric_bike" "electric_bike" "electric_bik
##  $ started_at        : POSIXct[1:5755694], format: "2021-11-27 13:27:38" "2021-11-27 13:38:25" "2021
##  $ ended_at          : POSIXct[1:5755694], format: "2021-11-27 13:46:38" "2021-11-27 13:56:10" "2021
##  $ start_station_name: chr [1:5755694] NA NA NA NA ...
##  $ start_station_id  : chr [1:5755694] NA NA NA NA ...
##  $ end_station_name  : chr [1:5755694] NA NA NA NA ...
##  $ end_station_id    : chr [1:5755694] NA NA NA NA ...
##  $ start_lat         : num [1:5755694] 41.9 42 42 41.9 41.9 ...
##  $ start_lng         : num [1:5755694] -87.7 -87.7 -87.7 -87.8 -87.6 ...
##  $ end_lat           : num [1:5755694] 42 41.9 42 41.9 41.9 ...
##  $ end_lng           : num [1:5755694] -87.7 -87.7 -87.7 -87.8 -87.6 ...
##  $ member_casual     : chr [1:5755694] "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>
```

```
summary(bike_rides)
```

```
##    ride_id          rideable_type        started_at                      ended_at
##  Length:5755694     Length:5755694     Min.   :2021-11-01 00:00:14.00   Min.   :2021-11-01 00:04:06.0
##  Class :character   Class :character   1st Qu.:2022-04-27 16:40:09.00   1st Qu.:2022-04-27 16:51:40.1
##  Mode  :character   Mode  :character   Median :2022-06-30 18:31:03.00   Median :2022-06-30 18:49:28.0
##                                        Mean   :2022-06-13 23:04:32.59   Mean   :2022-06-13 23:23:58.9
##                                        3rd Qu.:2022-08-24 19:52:19.75   3rd Qu.:2022-08-24 20:10:05.7
##                                        Max.   :2022-10-31 23:59:33.00   Max.   :2022-11-07 04:53:58.0
##
##  end_station_name   end_station_id       start_lat       start_lng         end_lat         end_lng
##  Length:5755694     Length:5755694     Min.   :41.64   Min.   :-87.84   Min.   :41.39   Min.   :-88.9
##  Class :character   Class :character   1st Qu.:41.88   1st Qu.:-87.66   1st Qu.:41.88   1st Qu.:-87.6
##  Mode  :character   Mode  :character   Median :41.90   Median :-87.64   Median :41.90   Median :-87.6
##                                        Mean   :41.90   Mean   :-87.65   Mean   :41.90   Mean   :-87.6
##                                        3rd Qu.:41.93   3rd Qu.:-87.63   3rd Qu.:41.93   3rd Qu.:-87.6
##                                        Max.   :45.64   Max.   :-73.80   Max.   :42.37   Max.   :-87.3
##                                                                         NA's   :5835    NA's   :5835
```

```
names(bike_rides)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"         "ended_at"           "start_stati
##  [7] "end_station_name"   "end_station_id"     "start_lat"          "start_lng"          "end_lat"
## [13] "member_casual"
```

```
skim_without_charts(bike_rides)
```

Table 1: Data summary

Name	bike_rides
Number of rows	5755694
Number of columns	13
Column type frequency:	
character	7
numeric	4
POSIXct	2
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ride_id	0	1.00	16	16	0	5755694	0
rideable_type	0	1.00	11	13	0	3	0
start_station_name	878177	0.85	7	64	0	1639	0
start_station_id	878177	0.85	3	44	0	1306	0
end_station_name	940010	0.84	9	64	0	1663	0
end_station_id	940010	0.84	3	44	0	1314	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
member_casual	0	1.00	6	6	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
start_lat	0	1	41.90	0.05	41.64	41.88	41.90	41.93	45.64
start_lng	0	1	-87.65	0.03	-87.84	-87.66	-87.64	-87.63	-73.80
end_lat	5835	1	41.90	0.05	41.39	41.88	41.90	41.93	42.37
end_lng	5835	1	-87.65	0.03	-88.97	-87.66	-87.64	-87.63	-87.30

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
started_at	0	1	2021-11-01 00:00:14	2022-10-31 23:59:33	2022-06-30 18:31:03	4824622
ended_at	0	1	2021-11-01 00:04:06	2022-11-07 04:53:58	2022-06-30 18:49:28	4836310

2.14 Identify missing data, limitations and other data problems

I'll use the is.na() method on each variable/column in my data frame to figure out what's going on where I have missing data.

I may use is.na() in conjunction with View() to generate a spreadsheet-like table and examine all rows to see if there are any patterns.

```
attach(bike_rides)
```

```
## The following objects are masked from bike_rides (pos = 3):
##
##     end_lat, end_lng, end_station_id, end_station_name, ended_at, member_casual, ride_id, rideable_ty
##     start_station_id, start_station_name, started_at
```

```
bike_rides[is.na(start_station_name),]
```

```
## # A tibble: 878,177 x 13
##    ride_id    rideable_type started_at          ended_at            start_station_name start_station_
##    <chr>      <chr>         <dttm>              <dttm>              <chr>              <chr>
##  1 7C00A93E1~ electric_bike 2021-11-27 13:27:38 2021-11-27 13:46:38 <NA>               <NA>
##  2 90854840D~ electric_bike 2021-11-27 13:38:25 2021-11-27 13:56:10 <NA>               <NA>
##  3 0A7D10CDD~ electric_bike 2021-11-26 22:03:34 2021-11-26 22:05:56 <NA>               <NA>
##  4 2F3BE3308~ electric_bike 2021-11-27 09:56:49 2021-11-27 10:01:50 <NA>               <NA>
##  5 D67B4781A~ electric_bike 2021-11-26 19:09:28 2021-11-26 19:30:41 <NA>               <NA>
##  6 EF780B807~ electric_bike 2021-11-27 13:31:12 2021-11-27 13:37:12 <NA>               <NA>
##  7 17069CC74~ electric_bike 2021-11-27 14:33:56 2021-11-27 14:34:38 <NA>               <NA>
##  8 93FC4662B~ electric_bike 2021-11-27 09:14:33 2021-11-27 09:19:36 <NA>               <NA>
##  9 B06B06439~ electric_bike 2021-11-27 16:13:31 2021-11-27 16:22:50 <NA>               <NA>
```

```
## 10 A2A194358~ electric_bike 2021-11-27 12:49:10 2021-11-27 12:52:47 <NA>               <NA>
## # i 878,167 more rows
## # i 4 more variables: start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>
```

```
bike_rides[is.na(start_station_id),]
```

```
## # A tibble: 878,177 x 13
##    ride_id    rideable_type started_at          ended_at            start_station_name start_station_
##    <chr>      <chr>         <dttm>              <dttm>              <chr>              <chr>
##  1 7C00A93E1~ electric_bike 2021-11-27 13:27:38 2021-11-27 13:46:38 <NA>               <NA>
##  2 90854840D~ electric_bike 2021-11-27 13:38:25 2021-11-27 13:56:10 <NA>               <NA>
##  3 0A7D10CDD~ electric_bike 2021-11-26 22:03:34 2021-11-26 22:05:56 <NA>               <NA>
##  4 2F3BE3308~ electric_bike 2021-11-27 09:56:49 2021-11-27 10:01:50 <NA>               <NA>
##  5 D67B4781A~ electric_bike 2021-11-26 19:09:28 2021-11-26 19:30:41 <NA>               <NA>
##  6 EF780B807~ electric_bike 2021-11-27 13:31:12 2021-11-27 13:37:12 <NA>               <NA>
##  7 17069CC74~ electric_bike 2021-11-27 14:33:56 2021-11-27 14:34:38 <NA>               <NA>
##  8 93FC4662B~ electric_bike 2021-11-27 09:14:33 2021-11-27 09:19:36 <NA>               <NA>
##  9 B06B06439~ electric_bike 2021-11-27 16:13:31 2021-11-27 16:22:50 <NA>               <NA>
## 10 A2A194358~ electric_bike 2021-11-27 12:49:10 2021-11-27 12:52:47 <NA>               <NA>
## # i 878,167 more rows
## # i 4 more variables: start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>
```

```
bike_rides[is.na(end_station_name),]
```

```
## # A tibble: 940,010 x 13
##    ride_id    rideable_type started_at          ended_at            start_station_name start_station_
##    <chr>      <chr>         <dttm>              <dttm>              <chr>              <chr>
##  1 7C00A93E1~ electric_bike 2021-11-27 13:27:38 2021-11-27 13:46:38 <NA>               <NA>
##  2 90854840D~ electric_bike 2021-11-27 13:38:25 2021-11-27 13:56:10 <NA>               <NA>
##  3 0A7D10CDD~ electric_bike 2021-11-26 22:03:34 2021-11-26 22:05:56 <NA>               <NA>
##  4 2F3BE3308~ electric_bike 2021-11-27 09:56:49 2021-11-27 10:01:50 <NA>               <NA>
##  5 D67B4781A~ electric_bike 2021-11-26 19:09:28 2021-11-26 19:30:41 <NA>               <NA>
##  6 02F85C2C3~ electric_bike 2021-11-26 18:34:07 2021-11-26 18:52:49 Michigan Ave & Oa~ 13042
##  7 EF780B807~ electric_bike 2021-11-27 13:31:12 2021-11-27 13:37:12 <NA>               <NA>
##  8 17069CC74~ electric_bike 2021-11-27 14:33:56 2021-11-27 14:34:38 <NA>               <NA>
##  9 93FC4662B~ electric_bike 2021-11-27 09:14:33 2021-11-27 09:19:36 <NA>               <NA>
## 10 B06B06439~ electric_bike 2021-11-27 16:13:31 2021-11-27 16:22:50 <NA>               <NA>
## # i 940,000 more rows
## # i 4 more variables: start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>
```

```
bike_rides[is.na(end_station_id),]
```

```
## # A tibble: 940,010 x 13
##    ride_id    rideable_type started_at          ended_at            start_station_name start_station_
##    <chr>      <chr>         <dttm>              <dttm>              <chr>              <chr>
##  1 7C00A93E1~ electric_bike 2021-11-27 13:27:38 2021-11-27 13:46:38 <NA>               <NA>
##  2 90854840D~ electric_bike 2021-11-27 13:38:25 2021-11-27 13:56:10 <NA>               <NA>
##  3 0A7D10CDD~ electric_bike 2021-11-26 22:03:34 2021-11-26 22:05:56 <NA>               <NA>
##  4 2F3BE3308~ electric_bike 2021-11-27 09:56:49 2021-11-27 10:01:50 <NA>               <NA>
##  5 D67B4781A~ electric_bike 2021-11-26 19:09:28 2021-11-26 19:30:41 <NA>               <NA>
##  6 02F85C2C3~ electric_bike 2021-11-26 18:34:07 2021-11-26 18:52:49 Michigan Ave & Oa~ 13042
##  7 EF780B807~ electric_bike 2021-11-27 13:31:12 2021-11-27 13:37:12 <NA>               <NA>
```

```
##  8 17069CC74~ electric_bike 2021-11-27 14:33:56 2021-11-27 14:34:38 <NA>               <NA>
##  9 93FC4662B~ electric_bike 2021-11-27 09:14:33 2021-11-27 09:19:36 <NA>               <NA>
## 10 B06B06439~ electric_bike 2021-11-27 16:13:31 2021-11-27 16:22:50 <NA>               <NA>
## # i 940,000 more rows
## # i 4 more variables: start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>
```

```
bike_rides[is.na(start_lat),]
```

```
## # A tibble: 0 x 13
## # i 13 variables: ride_id <chr>, rideable_type <chr>, started_at <dttm>, ended_at <dttm>, start_stati
## #   end_station_name <chr>, end_station_id <chr>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, en
```

```
bike_rides[is.na(start_lng),]
```

```
## # A tibble: 0 x 13
## # i 13 variables: ride_id <chr>, rideable_type <chr>, started_at <dttm>, ended_at <dttm>, start_stati
## #   end_station_name <chr>, end_station_id <chr>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, en
```

```
bike_rides[is.na(end_lat),]
```

```
## # A tibble: 5,835 x 13
##    ride_id    rideable_type started_at          ended_at            start_station_name start_station_
##    <chr>      <chr>         <dttm>              <dttm>              <chr>              <chr>
##  1 D66FB7A50~ classic_bike  2021-11-23 11:53:36 2021-11-24 12:53:30 Laflin St & Culle~ 13307
##  2 214DC891A~ classic_bike  2021-11-25 19:23:35 2021-11-26 20:23:30 Rush St & Superio~ 15530
##  3 4409AA46B~ classic_bike  2021-11-06 13:13:06 2021-11-07 13:13:01 Ashland Ave & 66t~ 16950
##  4 C4A464C28~ docked_bike   2021-11-06 16:40:58 2021-11-06 17:24:39 Millennium Park    13008
##  5 E58A224FA~ docked_bike   2021-11-25 13:56:42 2021-11-26 16:50:51 Millennium Park    13008
##  6 D893434A1~ docked_bike   2021-11-26 16:07:04 2021-11-27 17:07:05 Shedd Aquarium     15544
##  7 6653EC1EF~ classic_bike  2021-11-12 10:00:16 2021-11-13 11:00:02 University Ave & ~ KA1503000071
##  8 B346545D5~ docked_bike   2021-11-01 18:49:55 2021-11-02 07:48:18 Shedd Aquarium     15544
##  9 4FCD69A37~ classic_bike  2021-11-06 16:24:11 2021-11-07 16:24:06 Western Ave & Div~ 13241
## 10 DCAA62549~ classic_bike  2021-11-20 00:47:58 2021-11-21 01:47:53 Broadway & Wilson~ 13074
## # i 5,825 more rows
## # i 4 more variables: start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>
```

```
bike_rides[is.na(end_lng),]
```

```
## # A tibble: 5,835 x 13
##    ride_id    rideable_type started_at          ended_at            start_station_name start_station_
##    <chr>      <chr>         <dttm>              <dttm>              <chr>              <chr>
##  1 D66FB7A50~ classic_bike  2021-11-23 11:53:36 2021-11-24 12:53:30 Laflin St & Culle~ 13307
##  2 214DC891A~ classic_bike  2021-11-25 19:23:35 2021-11-26 20:23:30 Rush St & Superio~ 15530
##  3 4409AA46B~ classic_bike  2021-11-06 13:13:06 2021-11-07 13:13:01 Ashland Ave & 66t~ 16950
##  4 C4A464C28~ docked_bike   2021-11-06 16:40:58 2021-11-06 17:24:39 Millennium Park    13008
##  5 E58A224FA~ docked_bike   2021-11-25 13:56:42 2021-11-26 16:50:51 Millennium Park    13008
##  6 D893434A1~ docked_bike   2021-11-26 16:07:04 2021-11-27 17:07:05 Shedd Aquarium     15544
##  7 6653EC1EF~ classic_bike  2021-11-12 10:00:16 2021-11-13 11:00:02 University Ave & ~ KA1503000071
##  8 B346545D5~ docked_bike   2021-11-01 18:49:55 2021-11-02 07:48:18 Shedd Aquarium     15544
##  9 4FCD69A37~ classic_bike  2021-11-06 16:24:11 2021-11-07 16:24:06 Western Ave & Div~ 13241
## 10 DCAA62549~ classic_bike  2021-11-20 00:47:58 2021-11-21 01:47:53 Broadway & Wilson~ 13074
## # i 5,825 more rows
## # i 4 more variables: start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>
```

```
bike_rides[is.na(member_casual),]
```

```
## # A tibble: 0 x 13
## # i 13 variables: ride_id <chr>, rideable_type <chr>, started_at <dttm>, ended_at <dttm>, start_stati
## #   end_station_name <chr>, end_station_id <chr>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, en
```

Summary of missing data, constraints, and other issues:

Following a thorough examination of the data, I discovered the following flaws and limitations:

- This public dataset has limits due to user privacy protection.
- We can't tell if a bike user is a local or a visitor because we don't have any user demographic data;
- We don't know how many customers there are because we can't map bike trips to customers;
- We can't tell if a casual rider bike trip is associated with a single-ride pass or a full day pass because we can't map bike trips to customers.

Additionally...

- Some bike journeys appear to be "testing" in nature.
- Start and finish latitude and longitude values do not have the same decimal point; decimal points range from 2 to 6 decimal points.

These data columns have missing values, which we will rectify later in the process: * start_station_name * start_station_id * end_station_name * end_station_id * end_lat * end_lng

2.15 Ability to address the business task

The historical travel data from Divvy is appropriate for answering business issues. The data supplied will assist us in understanding how casual and member users use bikes differently. While the data files give consistent columns of data, we may expand on them by applying computations and functions to gain deeper insights.

While the data has various restrictions and concerns, there are no major faults that make the data worthless. We have a fantastic dataset from which we can derive numerous insights, including:

- Bike type usage by customer type
- Number of bike trips by customer type and time of the day, day of the week, season
- Length of bike trips by customer type and time of the day, day of the week, season

Step 3 : Process the data by cleaning and checking the information

This stage entails selecting tools that are appropriate for the amount of data you will be dealing with. Checking for data issues, cleaning the data, converting the data by adding, renaming, and removing data, and lastly ensuring that the data is clean and suitable for analysis are all part of the process. It's critical to stick to an organised procedure and document all of your actions so that coworkers can follow along and double-check your work.

3.1 Select the tools for the project

Since the combined dataset is very large with 5.7 million rows, R and RStudio has been chosen as the tool for data manipulation, cleaning, aggregation, analysis and visualization.

3.2 Transforming the data to make it work effectively

3.2.1 Renaming Columns

```
bike_rides <- rename(bike_rides, "bike_type" = "rideable_type", "user_type" = "member_casual")
```

3.2.2 Ensure datetime format is consistent throughout the started_at and ended_at columns.

```
bike_rides$started_at <- ymd_hms(bike_rides$started_at)
bike_rides$ended_at <- ymd_hms(bike_rides$ended_at)
```

3.2.3 Adding Columns

Adding a new column called, "ride_length_min".

Using the difftime() function to calculate the length of each trip in minutes, rounded to two decimals.

```
bike_rides$ride_length_min <- round(as.numeric(difftime(bike_rides$ended_at, bike_rides$started_at, uni
```

Verify that R recognizes my new variable as numeric so that I can perform calculations.

```
class(bike_rides$ride_length_min)
```

```
## [1] "numeric"
```

Adding columns for: date, month, day, year, day_of_week, and hour

```
bike_rides$date <- as.Date(bike_rides$started_at)
bike_rides$month <- format(as.Date(bike_rides$date), "%B")
bike_rides$day <- format(as.Date(bike_rides$date), "%d")
bike_rides$year <- format(as.Date(bike_rides$date), "%Y")
bike_rides$day_of_week <- format(as.Date(bike_rides$date), "%A")
bike_rides$hour <- lubridate::hour(bike_rides$started_at)
```

Adding a column for season

```
bike_rides <- bike_rides %>% mutate(season = recode(month,
                        December = "Winter",
                        January = "Winter",
                        February = "Spring",
                        March = "Spring",
                        April = "Spring",
                        May = "Summer",
                        June = "Summer",
                        July = "Summer",
                        August = "Fall",
                        September = "Fall",
                        October = "Fall",
                        November = "Winter"))
```

Adding a column for "time_of_day" using a case_when function

```
bike_rides <- bike_rides %>% mutate(time_of_day = case_when(
                        hour >= 6 & hour < 9 ~ "Early Morning",
                        hour >= 9 & hour < 12 ~ "Mid Morning",
                        hour >= 12 & hour < 18  ~ "Afternoon",
                        hour >= 18 & hour <= 23  ~ "Evening",
                        hour >= 0 & hour < 3  ~ "Early Night",
                        hour >= 3 & hour < 6  ~ "Late Night"))
```

3.2.4 Exploring newly created column, ride_length_min

Calculating where ride_length_min is greater than 1,440 minutes (or 24 hours)

```
sum(bike_rides$ride_length_min > 1440)
```

```
## [1] NA
```

Calculating where ride_length_min is less than 1 minute (or 60 seconds)

```
sum(bike_rides$ride_length_min < 1)
```

```
## [1] NA
```

Calculating where ride_length_min is less than 0 or a negative number.

```
sum(bike_rides$ride_length_min < 0)
```

```
## [1] NA
```

```
sum(bike_rides$started_at > bike_rides$ended_at)
```

```
## [1] NA
```

```
length(which(bike_rides$started_at > bike_rides$ended_at))
```

```
## [1] 112
```

Calculating where ride_length_min is less than 0 or a negative number

```
sum(bike_rides$ride_length_min < 0)
```

```
## [1] NA
```

Where ride_length_min is greater than 6 hours (or 360 minutes)

```
sum(bike_rides$ride_length_min > 360)
```

```
## [1] NA
```

3.3 Taking a closer look at missing values

Let's see if the missing start_station_id numbers are related to a specific bike or user type. We can observe that the majority of the issue is with electric bikes and does not apply to any specific user type.

```
bike_rides %>% filter(is.na(start_station_id)) %>%
  count(start_station_id, start_station_name, bike_type, user_type)
```

```
## # A tibble: 2 x 5
##   start_station_id start_station_name bike_type     user_type      n
##   <chr>            <chr>              <chr>         <chr>      <int>
## 1 <NA>             <NA>               electric_bike casual    363963
## 2 <NA>             <NA>               electric_bike member    514214
```

The missing end_station_id pertains to the three bike types and both user types, but again, the majority of the problem is with electric bikes.

```
bike_rides %>% filter(is.na(end_station_id)) %>%
  count(end_station_id, end_station_name, bike_type, user_type)
```

```
## # A tibble: 5 x 5
##   end_station_id end_station_name bike_type     user_type      n
##   <chr>          <chr>            <chr>         <chr>      <int>
## 1 <NA>           <NA>             classic_bike  casual      2805
## 2 <NA>           <NA>             classic_bike  member      1533
## 3 <NA>           <NA>             docked_bike   casual      2570
## 4 <NA>           <NA>             electric_bike casual    422751
## 5 <NA>           <NA>             electric_bike member    510351
```

Let's have a look at what else is going on in the absence of a start_station_id. What's going on with the start station name, start_lat, and start_lng. This demonstrates that the start_station name is missing, and both start_lat and start_lng are simply two decimal points.

```
bike_rides %>% filter(is.na(start_station_id)) %>%
  count(start_station_id, start_station_name, start_lat, start_lng)
```

```
## # A tibble: 621 x 5
##    start_station_id start_station_name start_lat start_lng     n
##    <chr>            <chr>                 <dbl>     <dbl> <int>
##  1 <NA>             <NA>                   41.6     -87.6     1
##  2 <NA>             <NA>                   41.6     -87.5     3
##  3 <NA>             <NA>                   41.6     -87.6    22
##  4 <NA>             <NA>                   41.6     -87.6     2
##  5 <NA>             <NA>                   41.6     -87.6     8
##  6 <NA>             <NA>                   41.6     -87.6     1
##  7 <NA>             <NA>                   41.6     -87.6     3
```

```
##  8 <NA>             <NA>                   41.6     -87.6    20
##  9 <NA>             <NA>                   41.6     -87.5    21
## 10 <NA>             <NA>                   41.6     -87.5     5
## # i 611 more rows
```

Let's have a look at what else is going on in the absence of an end_station_id. I'm particularly interested in what's going on with the end station name, end_lat, and end_lng. This demonstrates that the end_station name is missing, and both start_lat and start_lng only go out two decimal points.

Adding the View() function provides a table we can then also filter on.

From this view, I can also see that there are 5,961 instances where both end_lat & end_lng are missing.

```
bike_rides %>% filter(is.na(end_station_id)) %>%
  count(end_station_id, end_station_name, end_lat, end_lng)
```

```
## # A tibble: 891 x 5
##    end_station_id end_station_name end_lat end_lng     n
##    <chr>          <chr>              <dbl>   <dbl> <int>
##  1 <NA>           <NA>                41.4   -89.0     1
##  2 <NA>           <NA>                41.5   -87.6     1
##  3 <NA>           <NA>                41.6   -87.3     4
##  4 <NA>           <NA>                41.6   -87.7     2
##  5 <NA>           <NA>                41.6   -87.6     1
##  6 <NA>           <NA>                41.6   -87.7     1
##  7 <NA>           <NA>                41.6   -87.6     1
##  8 <NA>           <NA>                41.6   -87.7     1
##  9 <NA>           <NA>                41.6   -87.6     1
## 10 <NA>           <NA>                41.6   -87.6     1
## # i 881 more rows
```

3.3.1 Address missing values

To account for the missing values for both start and finish station names and ids, I'll add four new columns that display start_lat, start_lng, end_lat, and end_lng, all rounded to two decimal places. I'll then use the new rounded latitude and longitude to fill in the missing start and end station names.

Creating four new columns to show start_lat, start_lng, end_lat & end_lng all rounded to 2 decimal places.

```
bike_rides <- bike_rides %>%
  mutate(start_lat_round = round(start_lat, digits = 2),
         start_lng_round = round(start_lng, digits = 2),
         end_lat_round = round(end_lat, digits = 2),
         end_lng_round = round(end_lng, digits = 2))
```

Impute missing start station names

```
bike_rides <- bike_rides %>%
  group_by(start_lat_round, start_lng_round) %>%
  tidyr::fill(start_station_name, .direction = "downup") %>%
  ungroup()
```

Impute missing end station names

```
bike_rides <- bike_rides %>%
  group_by(end_lat_round, end_lng_round) %>%
  tidyr::fill(end_station_name, .direction = "downup") %>%
  ungroup()
```

Impute missing start_station_id

```
bike_rides <- bike_rides %>%
  group_by(start_station_name) %>%
  tidyr::fill(start_station_id, .direction = "downup") %>%
  ungroup()
```

Impute missing end_station_id

```
bike_rides <- bike_rides %>%
  group_by(end_station_name) %>%
  tidyr::fill(end_station_id, .direction = "downup") %>%
  ungroup()
```

Now that we have imputed a lot of the missing data, let's check missing values by column, again, and see what's still missing.

```
colSums(is.na(bike_rides))
```

```
##            ride_id          bike_type         started_at           ended_at start_station_name   sta
##                  0                  0                 27                 32              12097
##     end_station_id          start_lat          start_lng            end_lat            end_lng
##              24394                  0                  0               5835               5835
##               date              month                day               year        day_of_week
##                 27                 27                 27                 27                 27
##        time_of_day    start_lat_round    start_lng_round      end_lat_round      end_lng_round
##                 27                  0                  0               5835               5835
```

We don't have an end_station_name or end_station_id where end_lat and end_lng are absent, so the missing data cannot be attributed using those columns. Except for one 2-minute ride, these are all 1-minute rides.

Because these are 1-minute rides with missing end-lat and end_lng, I think they're slips and will be removed. In terms of percentage, 5,961 points that are absent are regarded as insignificant and will not affect the integrity of my dataset.

```
bike_rides %>% filter(is.na(end_lat)) %>%
  count(end_station_name, end_station_id, end_lat, end_lng, bike_type)
```

```
## # A tibble: 2 x 6
##   end_station_name end_station_id end_lat end_lng bike_type        n
##   <chr>            <chr>            <dbl>   <dbl> <chr>       <int>
## 1 <NA>             <NA>                NA      NA classic_bike  3265
## 2 <NA>             <NA>                NA      NA docked_bike   2570
```

We don't have a start_station_id to impute the missing data where the start_station_name is absent. I should be able to infer from data that has end_lat and end_lng coordinates that match. That would be more difficult because we may have numerous station names with latitude and longitude coordinates that only go out two decimal places.

For the sake of time and because the amount of data is insignificant, I'm going to leave it as is and go on. I could remove this information because it is irrelevant.

```
bike_rides %>% filter(is.na(start_station_name)) %>%
  count(start_station_name, start_station_id, start_lat, start_lng, bike_type)
```

```
## # A tibble: 112 x 6
##    start_station_name start_station_id start_lat start_lng bike_type         n
##    <chr>              <chr>                <dbl>     <dbl> <chr>         <int>
##  1 <NA>               <NA>                  41.6     -87.6 electric_bike     1
##  2 <NA>               <NA>                  41.6     -87.5 electric_bike     3
##  3 <NA>               <NA>                  41.6     -87.6 electric_bike     2
##  4 <NA>               <NA>                  41.6     -87.6 electric_bike     1
##  5 <NA>               <NA>                  41.6     -87.6 electric_bike     3
##  6 <NA>               <NA>                  41.6     -87.5 electric_bike     5
##  7 <NA>               <NA>                  41.7     -87.6 electric_bike    13
##  8 <NA>               <NA>                  41.7     -87.6 electric_bike    24
##  9 <NA>               <NA>                  41.7     -87.6 electric_bike     6
## 10 <NA>               <NA>                  41.7     -87.6 electric_bike     3
## # i 102 more rows
```

We don't have a end_station_id to impute the missing data where the end_station_name is absent. I should be able to infer from data that has end_lat and end_lng coordinates that match. That would be more difficult because we may have numerous station names with latitude and longitude coordinates that only go out two decimal places.

For the sake of time and because the amount of data is insignificant, I'm going to leave it as is and go on. I could remove this information because it is irrelevant.

```
bike_rides %>% filter(is.na(end_station_name)) %>%
  count(end_station_name, end_station_id, end_lat, end_lng, bike_type)
```

```
## # A tibble: 394 x 6
##    end_station_name end_station_id end_lat end_lng bike_type         n
##    <chr>            <chr>            <dbl>   <dbl> <chr>         <int>
##  1 <NA>             <NA>              41.4   -89.0 electric_bike     1
##  2 <NA>             <NA>              41.5   -87.6 electric_bike     1
##  3 <NA>             <NA>              41.6   -87.3 electric_bike     4
##  4 <NA>             <NA>              41.6   -87.7 electric_bike     2
##  5 <NA>             <NA>              41.6   -87.6 electric_bike     1
##  6 <NA>             <NA>              41.6   -87.7 electric_bike     1
##  7 <NA>             <NA>              41.6   -87.6 electric_bike     1
##  8 <NA>             <NA>              41.6   -87.7 electric_bike     1
##  9 <NA>             <NA>              41.6   -87.6 electric_bike     1
## 10 <NA>             <NA>              41.6   -87.6 electric_bike     1
## # i 384 more rows
```

3.4 Explore data related to "testing"

I spotted some data connected to "testing" during my initial check of the data in Excel, so I'll dig deeper to see what I find. According to the corporate website, trips are taken by employees as they service and examine the system. These trips should be eliminated. These trips should be removed. click here Divvy System Data.

I'll investigate more to see which trips are taken by personnel. I can identify the "test/warehouse" trips by looking at the start_station_id.

```
bike_rides %>%
  select(start_station_id) %>%
  count(start_station_id) %>%
  arrange(desc(n))
```

```
## # A tibble: 1,307 x 2
##    start_station_id     n
##    <chr>            <int>
##  1 13022            81600
##  2 LF-005           46373
##  3 TA1308000050     43882
##  4 KA1503000014     43191
##  5 13042            43177
##  6 13300            43063
##  7 TA1307000039     38794
##  8 13008            37492
##  9 13179            36145
## 10 KA1503000043     35709
## # i 1,297 more rows
```

Taking a look at the end_station_id allows me to identify the "test/warehouse" trips.

```
bike_rides %>%
  select(end_station_id) %>%
  count(end_station_id) %>%
  arrange(desc(n))
```

```
## # A tibble: 1,315 x 2
##    end_station_id     n
##    <chr>          <int>
##  1 13022          81735
##  2 KA1503000071   45680
##  3 TA1308000050   44828
##  4 KA1503000014   44501
##  5 LF-005         44390
##  6 13042          43001
##  7 13300          42702
##  8 TA1307000039   38609
##  9 13008          37740
## 10 13179          36954
## # i 1,305 more rows
```

I'll use the filter function to filter on these multiple specific cases by creating a vector of values c()

```
bike_rides %>% filter(start_station_id %in% c("DIVVY 001", "DIVVY 001 - Warehouse test station", "Hubba
  count(start_station_id)
```

```
## # A tibble: 8 x 2
##   start_station_id                                n
##   <chr>                                       <int>
## 1 2059 Hastings Warehouse Station               585
## 2 DIVVY 001                                      77
## 3 DIVVY 001 - Warehouse test station             14
## 4 DIVVY CASSETTE REPAIR MOBILE STATION            7
## 5 Hastings WH 2                                  15
## 6 Hubbard Bike-checking (LBS-WH-TEST)          8377
## 7 Pawel Bialowas - Test- PBSC charging station    1
## 8 Throop/Hastings Mobile Station                 37
```

I'll use the filter function to filter on these multiple specific cases by creating a vector of values c()

```
bike_rides %>% filter(end_station_id %in% c("DIVVY 001", "DIVVY 001 - Warehouse test station", "Hubbard
  count(end_station_id)
```

```
## # A tibble: 6 x 2
##   end_station_id                                  n
##   <chr>                                       <int>
## 1 2059 Hastings Warehouse Station               585
## 2 DIVVY CASSETTE REPAIR MOBILE STATION            7
## 3 Hastings WH 2                                 179
## 4 Hubbard Bike-checking (LBS-WH-TEST)           823
## 5 Pawel Bialowas - Test- PBSC charging station    2
## 6 Throop/Hastings Mobile Station                  1
```

3.5 Removing Data

Following a careful evaluation of the data, I will delete the following:

- Rides of less than 60 seconds because they may be false starts or users attempting to re-dock a bike to verify it was secure, according to the Divvy website: Data Distribution in the System
- Rides with negative ride_length are invalid because the trip start time cannot be greater than the trip end time.
- For the sake of this research, rides with a ride_length more than 24 hours are considered invalid outliers.
- Rides where end_lat and end_lng are both missing; we don't have an end_station_name or end_station_id in these circumstances, hence the missing data cannot be imputed. In terms of percentage, 5,835 missing data points are deemed insignificant. With the exception of one 2-minute ride, these are all 1-minute rides. Because they lack end-lat and end_lng, I believe they are anomalies.
- Rides where the start_station_id or end_station_id are related to "testing" as previously identified

I'll use the select() function to create a new data frame with only selected columns.

```
bike_rides_v2 <- select(bike_rides, c(1,2,5, 6:16, 13:16, 18:22))
```

Remove rides less than 60 seconds (or 1 minute) and greater than 24 hrs (or 1440 minutes) in length. This will remove all rides with a negative ride_length

```
bike_rides_v2 <- bike_rides_v2 %>%
  filter(ride_length_min >= 1 & ride_length_min <= 1440)
```

Remove rides where end_lat & end_lng are both missing

```
bike_rides_v2 <- bike_rides_v2 %>%
  filter(!is.na(end_lat) & !is.na(end_lng))
```

Remove rides related to test/repair stations

```
bike_rides_v2 <- bike_rides_v2 %>%
  filter(!start_station_id %in% c("DIVVY 001", "DIVVY 001 - Warehouse test station", "Hubbard Bike-chec
```

```
bike_rides_v2 <- bike_rides_v2 %>%
  filter(!end_station_id %in% c("DIVVY 001", "DIVVY 001 - Warehouse test station", "Hubbard Bike-checki
```

3.5.1 Confirm the data removal

Confirms removal where ride_length_min is greater than 1,440 minutes (24 hours)

```
sum(bike_rides_v2$ride_length_min > 1440)
```

```
## [1] 0
```

Confirms removal where ride_length_min is less than 1 minute (or 60 seconds)

```
sum(bike_rides_v2$ride_length_min < 1)
```

```
## [1] 0
```

Confirms removal of rides related to test/repair stations

```
bike_rides_v2 %>% filter(end_station_id %in% c("DIVVY 001", "DIVVY 001 - Warehouse test station", "Hubba
  count(end_station_id)
```

```
## # A tibble: 0 x 2
## # i 2 variables: end_station_id <chr>, n <int>
```

Confirms removal of rides related to test/repair stations

```
bike_rides_v2 %>% filter(end_station_id %in% c("DIVVY 001", "DIVVY 001 - Warehouse test station", "Hubba
  count(end_station_id)
```

```
## # A tibble: 0 x 2
## # i 2 variables: end_station_id <chr>, n <int>
```

Confirms removal where end_lat & end_lng were both missing

```
colSums(is.na(bike_rides_v2))
```

```
##            ride_id          bike_type start_station_name   start_station_id   end_station_name     en
##                  0                  0              11750              11750              17898
##          start_lng            end_lat            end_lng          user_type    ride_length_min
##                  0                  0                  0                  0                  0
##               year        day_of_week               hour             season        time_of_day
##                  0                  0                  0                  0                  0
```

3.6 Inspect the new data frame

Let's acquire a row count and look at our new data frame, bike_rides_v2. We had 5,621,147 rows of data after cleaning it, which implies we deleted a total of 134,547 rows.

```
nrow(bike_rides_v2)
```

```
## [1] 5621089
```

```
glimpse(bike_rides_v2)
```

```
## Rows: 5,621,089
## Columns: 19
## $ ride_id            <chr> "7C00A93E10556E47", "90854840DFD508BA", "0A7D10CDD144061C", "2F3BE33085BC
## $ bike_type          <chr> "electric_bike", "electric_bike", "electric_bike", "electric_bike", "elec
## $ start_station_name <chr> "Kosciuszko Park", "California Ave & Montrose Ave", "California Ave & Mon
## $ start_station_id   <chr> "15643", "15622", "15622", "397", "TA1306000011", "13042", "KA1503000065"
## $ end_station_name   <chr> "Keystone Ave & Montrose Ave", "Humboldt Blvd & Armitage Ave", "Californi
## $ end_station_id     <chr> "KA1504000164", "15651", "15622", "314", "13300", "TA1306000011", "KA1503
## $ start_lat          <dbl> 41.93000, 41.96000, 41.96000, 41.94000, 41.90000, 41.90086, 41.81000, 41.8
## $ start_lng          <dbl> -87.72000, -87.70000, -87.70000, -87.79000, -87.63000, -87.62379, -87.600
## $ end_lat            <dbl> 41.96000, 41.92000, 41.96000, 41.93000, 41.88000, 41.90000, 41.80000, 41.
## $ end_lng            <dbl> -87.73000, -87.70000, -87.70000, -87.79000, -87.62000, -87.63000, -87.600
## $ user_type          <chr> "casual", "casual", "casual", "casual", "casual", "casual", "casual", "cas
## $ ride_length_min    <dbl> 19.00, 17.75, 2.37, 5.02, 21.22, 18.70, 6.00, 5.05, 9.32, 3.62, 25.33, 4.
## $ date               <date> 2021-11-27, 2021-11-27, 2021-11-26, 2021-11-27, 2021-11-26, 2021-11-26,
## $ month              <chr> "November", "November", "November", "November", "November", "November", "N
## $ year               <chr> "2021", "2021", "2021", "2021", "2021", "2021", "2021", "2021", "2021", "
## $ day_of_week        <chr> "Saturday", "Saturday", "Friday", "Saturday", "Friday", "Friday", "Saturda
## $ hour               <int> 13, 13, 22, 9, 19, 18, 13, 9, 16, 12, 13, 11, 16, 12, 7, 21, 17, 6, 15, 7
## $ season             <chr> "Winter", "Winter", "Winter", "Winter", "Winter", "Winter", "Winter", "Wi
## $ time_of_day        <chr> "Afternoon", "Afternoon", "Evening", "Mid Morning", "Evening", "Evening",
```

We are still missing some start and finish station names and ids, but we have the start and end lat and lng for these, so we can work with this data if necessary. It would not be a major deal if we removed it because it accounts for a small percentage of the data. We'll keep it for the time being.

```
colSums(is.na(bike_rides_v2))
```

```
##            ride_id          bike_type start_station_name   start_station_id   end_station_name     en
##                  0                  0              11750              11750              17898
```

```
##          start_lng            end_lat            end_lng          user_type    ride_length_min
##                  0                  0                  0                  0                  0
##               year        day_of_week               hour             season        time_of_day
##                  0                  0                  0                  0                  0
```

3.7 Verify the data is clean and ready to analyze

Before proceeding to the analysis phase, I'll perform a brief check to ensure the data is now clean, correct, consistent, and complete.

- Checked for duplicates: there are no duplicate values in the data.
- Missing value checks were performed, and some data with missing values were eliminated.
- Outliers were excluded (ride lengths > 24 hours).
- Data accuracy was verified: after cleaning up the data, the remaining data remained intact.
- Data completeness has been verified: the data for the previous 12 months is complete.
- Data consistency was checked: after cleaning up the data, the data for the previous 12 months remained consistent.
- Data relevance has been verified: the dataset for the last 12 months is current and relevant.
- Checked for relevance: the data is relevant to answering the business questions.
- Data formats were checked: The columns are formatted correctly.
- Date and time format consistency: the date and time columns are consistent.
- I looked for meaningful column names and found the following: altered and added a few cols to ensure that everything is clear and significant
- Overall impression: this is a great set of data that is consistent, well-formatted, and makes sense.

Step 4 : Analyzing and Visualizing the Data

At this point, we work with the data to analyze and examine it in various ways. I'll use functions to assist me in examining relationships and trends while staying focused on investigating how annual members and casual riders use Cyclistic bikes differently.

4.1 Summary of the data at the start of analysis

```
summary(bike_rides_v2)
```

```
##    ride_id           bike_type         start_station_name start_station_id   end_station_name   end_s
##  Length:5621089     Length:5621089     Length:5621089     Length:5621089     Length:5621089     Leng
##  Class :character   Class :character   Class :character   Class :character   Class :character   Class
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode
##
##
##
##    start_lng         end_lat         end_lng        user_type         ride_length_min        date
##  Min.   :-87.84   Min.   :41.39   Min.   :-88.97   Length:5621089     Min.   :   1.00   Min.   :2021-
##  1st Qu.:-87.66   1st Qu.:41.88   1st Qu.:-87.66   Class :character   1st Qu.:   6.08   1st Qu.:2022-
##  Median :-87.64   Median :41.90   Median :-87.64   Mode  :character   Median :  10.55   Median :2022-
##  Mean   :-87.65   Mean   :41.90   Mean   :-87.65                      Mean   :  16.63   Mean   :2022-
##  3rd Qu.:-87.63   3rd Qu.:41.93   3rd Qu.:-87.63                      3rd Qu.:  18.82   3rd Qu.:2022-
##  Max.   :-87.52   Max.   :42.37   Max.   :-87.30                      Max.   :1439.85   Max.   :2022-
```

```
##      year           day_of_week             hour          season           time_of_day
##  Length:5621089     Length:5621089     Min.   : 0.00   Length:5621089     Length:5621089
##  Class :character   Class :character   1st Qu.:11.00   Class :character   Class :character
##  Mode  :character   Mode  :character   Median :15.00   Mode  :character   Mode  :character
##                                        Mean   :14.22
##                                        3rd Qu.:18.00
##                                        Max.   :23.00
```

4.2 Casual Rider vs Member Rider

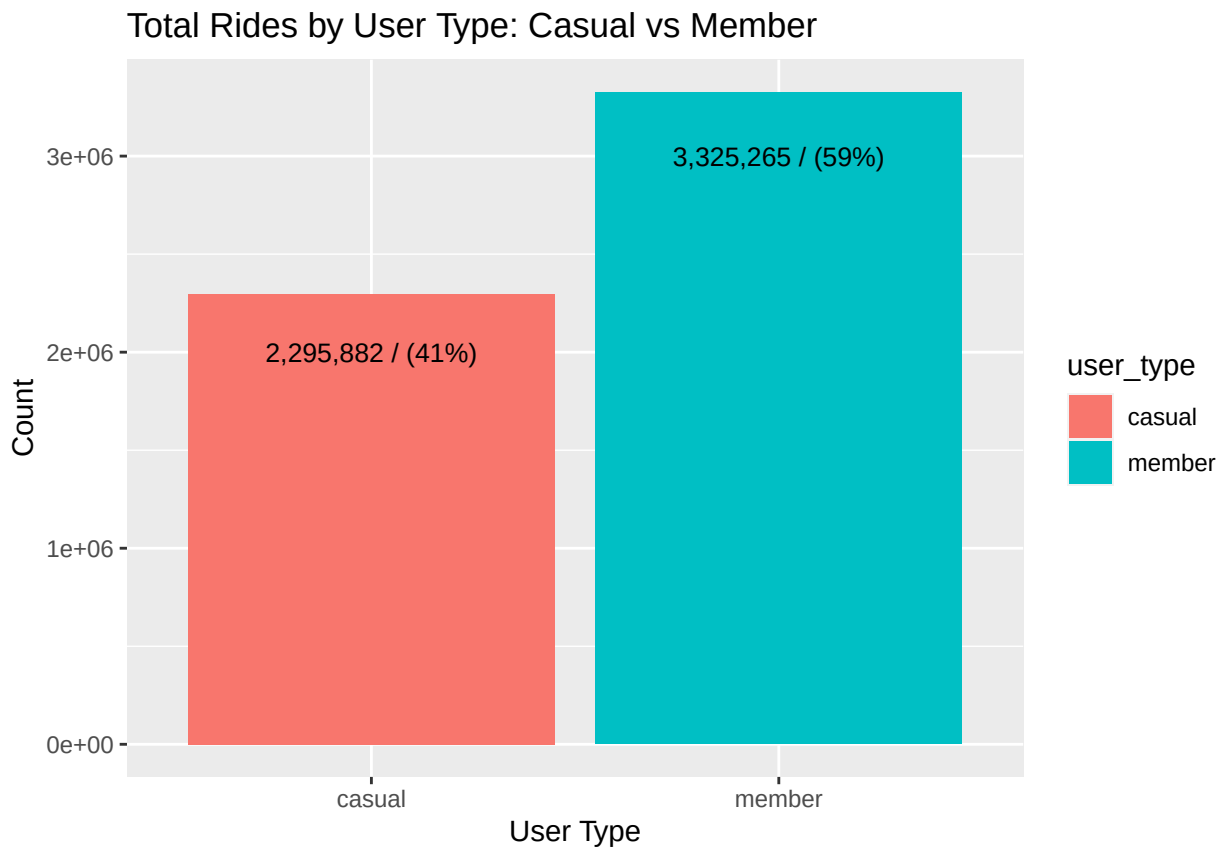
Count and Percentage breakdown of the two user types: the casual rider vs the member rider

```
bike_rides_v2 %>%
  group_by(user_type) %>%
  summarise(count = n(), Percentage = n()/nrow(bike_rides_v2)*100)
```

```
## # A tibble: 2 x 3
##   user_type   count Percentage
##   <chr>       <int>      <dbl>
## 1 casual    2295850       40.8
## 2 member    3325239       59.2
```

Visualizing total rides by user type

```
ggplot(bike_rides_v2, aes(user_type, fill = user_type)) +
  geom_bar() +
  labs(x = "User Type", y = "Count", title = "Total Rides by User Type: Casual vs Member") +
   annotate("text",x=1,y=2000000,label="2,295,882 / (41%)",color="black",size=3.5) +
   annotate("text",x=2,y=3000000,label="3,325,265 / (59%)",color="black",size=3.5)
```



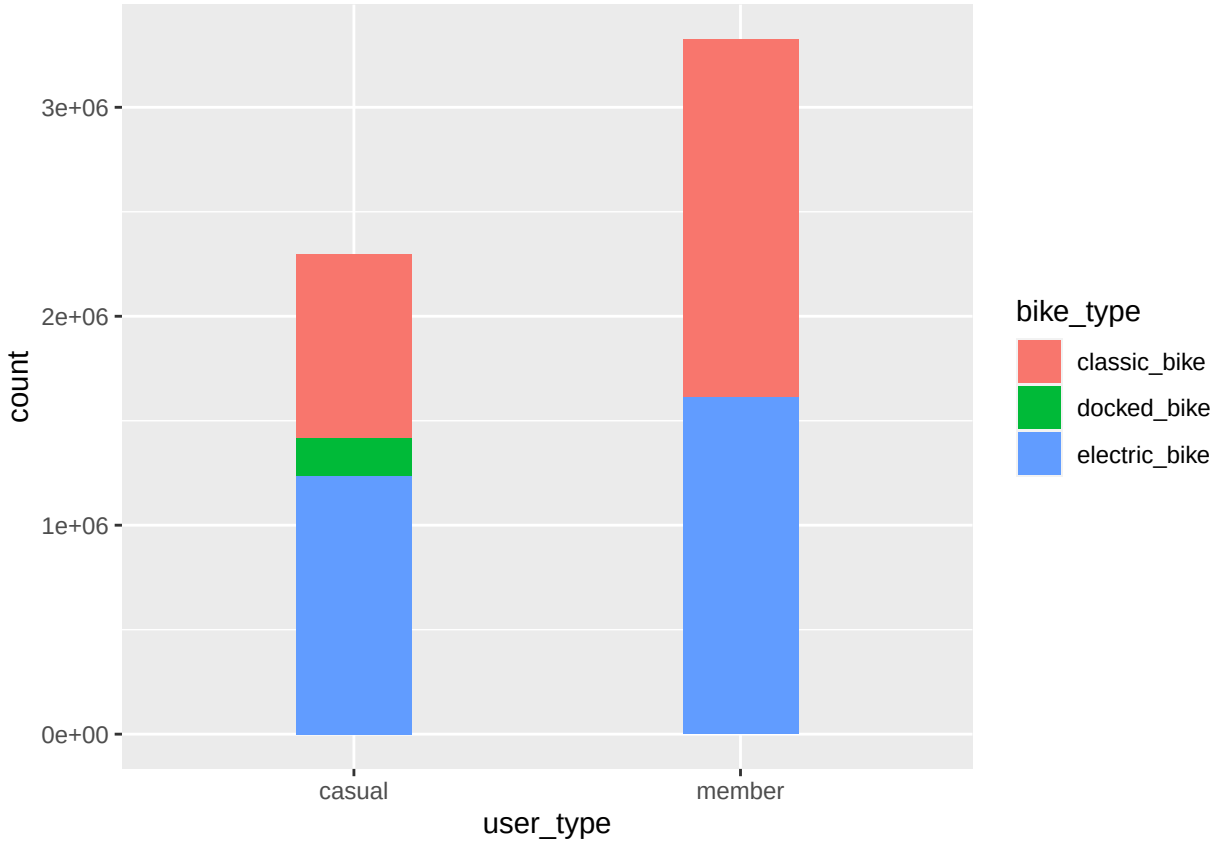
Key insight:

- 3,325,265 (or 59%) of riders are member riders, while 2,295,882 (or 41%) are casual riders

Visualizing total rides by user type and bike type

```
bike_rides_v2 %>%
  group_by(user_type, bike_type) %>%
  summarise(count = n()) %>%
  ggplot(aes(x = user_type, y = count, fill = bike_type)) +
  geom_bar(stat = "identity", width = 0.3)
```

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




```
  labs(x="Bike Type", y="Number of Rides", title = "Total Rides by user type and bike type")
```

```
## $x
## [1] "Bike Type"
##
## $y
## [1] "Number of Rides"
##
## $title
## [1] "Total Rides by user type and bike type"
##
## attr(,"class")
## [1] "labels"
```

Key insight:

- Member and casual riders use both the classic and electric bike, while only casual users use docked bikes.

4.3 Analyzing Ride Length

length of each trip (in minutes)

```r
summary(bike_rides_v2$ride_length_min)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    6.08   10.55   16.63   18.82 1439.85
```

Key insights:

- Average of all ride lengths is 16.63 minutes
- Minimum of all ride lengths is 1 minute
- Maximum of all ride lengths is right under 24 hours (or 1440 minutes)
- (this aligns with our data cleaning which removed < 1 min & > 24 hours)

Average ride length of each trip (in minutes) by user type

```r
aggregate(bike_rides_v2$ride_length_min ~ bike_rides_v2$user_type, FUN = mean)
```

```
##   bike_rides_v2$user_type bike_rides_v2$ride_length_min
## 1                  casual                       22.37844
## 2                  member                       12.65979
```

Taking a closer look at ride lengths

I'm summarizing the data by ride length here to see if anything pops out. According to the table, 5,453,815 of the rides are 60 minutes or less. This is an important topic to concentrate on.

```r
bike_rides_v2 %>%
  group_by(user_type) %>%
  summarize(`<= 5min` = sum(ride_length_min <= 5),
            `<= 12min` = sum(ride_length_min <= 12),
            `<= 20min` = sum(ride_length_min <= 20),
            `<= 45min` = sum(ride_length_min <= 45),
            `<= 60min` = sum(ride_length_min <= 60),
            `> 2hrs` = sum(ride_length_min > 120),
            `> 4hrs` = sum(ride_length_min > 240),
            `> 6hrs` = sum(ride_length_min > 360)) %>%
  add_row(user_type = "Total",
          `<= 5min` = sum(.$`<= 5min`),
          `<= 12min` = sum(.$`<= 12min`),
          `<= 20min` = sum(.$`<= 20min`),
          `<= 45min` = sum(.$`<= 45min`),
          `<= 60min` = sum(.$`<= 60min`),
          `> 2hrs` = sum(.$`> 2hrs`),
          `> 4hrs` = sum(.$`> 4hrs`),
          `> 6hrs` = sum(.$`> 6hrs`))
```

```
## # A tibble: 3 x 9
##   user_type `<= 5min` `<= 12min` `<= 20min` `<= 45min` `<= 60min` `> 2hrs` `> 4hrs` `> 6hrs`
##   <chr>         <int>      <int>      <int>      <int>      <int>    <int>    <int>    <int>
## 1 casual       253457    1027015    1547609    2063938    2151546    34139     5800     3260
## 2 member       751880    2127911    2789358    3269860    3302214     5735     2058     1244
## 3 Total       1005337    3154926    4336967    5333798    5453760    39874     7858     4504
```

Key insights:

- 97% of all rides are 60 minutes or less
- 79% of all rides are 20 minutes or less
- 56% of all rides are less than 15 minutes
- In my opinion, this area is crucial to gathering more data and developing a marketing strategy.

Looking strictly at ride lengths under 12 minutes

Key insights:

- 67% of these riders are member riders
- This is double the casual riders in this category

Let's look strictly at ride lengths <= 20 minutes

Key insights:

- 64% of these riders are member riders
- As the ride length increased, we see a slight shift in the breakdown

This tables shows us that 56% of the rides are less than 12 minutes and 79% of the rides are 20 minutes or less.

```
bike_rides_v2 %>%
   group_by(user_type) %>%
  summarize("<12 min" = sum(ride_length_min < 11.99),
            "12-20 min" = sum(ride_length_min >= 12 & ride_length_min <= 20.99),
            "21-30 min" = sum(ride_length_min >= 21 & ride_length_min <= 30.99),
            "31-60 min" = sum(ride_length_min >= 31 & ride_length_min <= 60),
            "61-120 min" = sum(ride_length_min >= 60.01 & ride_length_min <= 120.99),
            "121-240 min" = sum(ride_length_min >= 121 & ride_length_min <= 240.99),
            "241+min" = sum(ride_length_min >= 241))
```

```
## # A tibble: 2 x 8
##   user_type `<12 min` `12-20 min` `21-30 min` `31-60 min` `61-120 min` `121-240 min` `241+min`
##   <chr>         <int>       <int>       <int>       <int>        <int>         <int>     <int>
## 1 casual      1025542      565686      294466      265852       110803         27750      5751
## 2 member      2125810      711234      290622      174548        17365          3608      2052
```

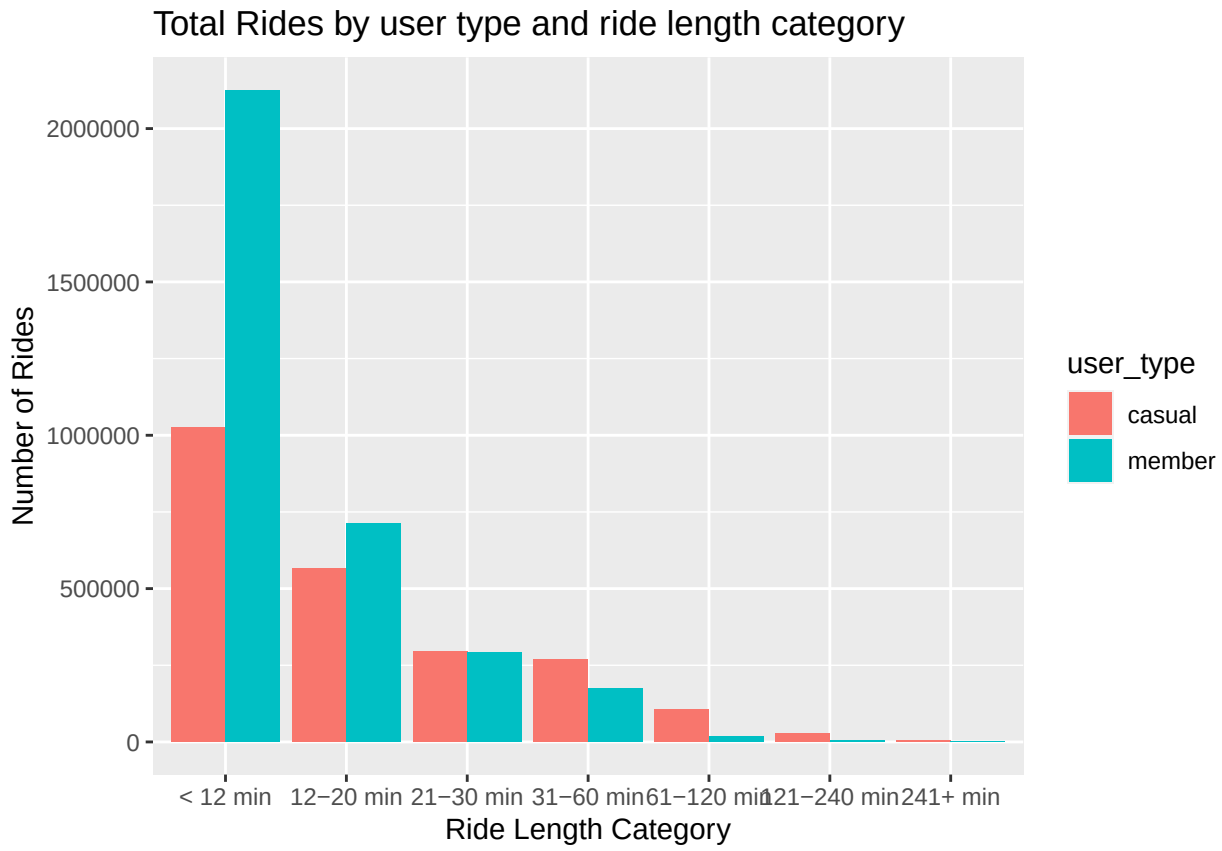
Adding a column to create ride length categories to get a better visual in R

```
bike_rides_v2 <- bike_rides_v2 %>% mutate(ride_length_cat = case_when(
                        ride_length_min < 11.99 ~ "< 12 min",
                        ride_length_min >= 12 & ride_length_min <= 20.99 ~ "12-20 min",
                        ride_length_min >= 21 & ride_length_min <= 30.99  ~ "21-30 min",
                        ride_length_min >= 31 & ride_length_min <= 60.99  ~ "31-60 min",
                        ride_length_min >= 60 & ride_length_min <= 120.99  ~ "61-120 min",
                        ride_length_min >= 121 & ride_length_min <= 240.99  ~ "121-240 min",
                        ride_length_min >= 241  ~ "241+ min"))
```

Visualizing total rides by user type and ride length category

```
bike_rides_v2 %>%
  group_by(user_type, ride_length_cat) %>%
  summarise(count = n()) %>%
  ggplot(aes(x=factor(ride_length_cat, level = c("< 12 min", "12-20 min", "21-30 min", "31-60 min","61-
  geom_col(position = "dodge") +
  labs(x="Ride Length Category", y="Number of Rides", title = "Total Rides by user type and ride length
```

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



Key insight:

- We can clearly see how the majority of rides fall into the first two categories

Average Ride length in minutes of those rides in the "< 12 min" category.

```
bike_rides_v2 %>% filter(ride_length_cat == "< 12 min") %>%
  group_by(user_type) %>%
  summarize(avg_ride_length=mean(ride_length_min))
```

```
## # A tibble: 2 x 2
##   user_type avg_ride_length
##   <chr>               <dbl>
## 1 casual               7.17
## 2 member               6.44
```

Average Ride length in minutes of those rides in the "<= 20 min" category.

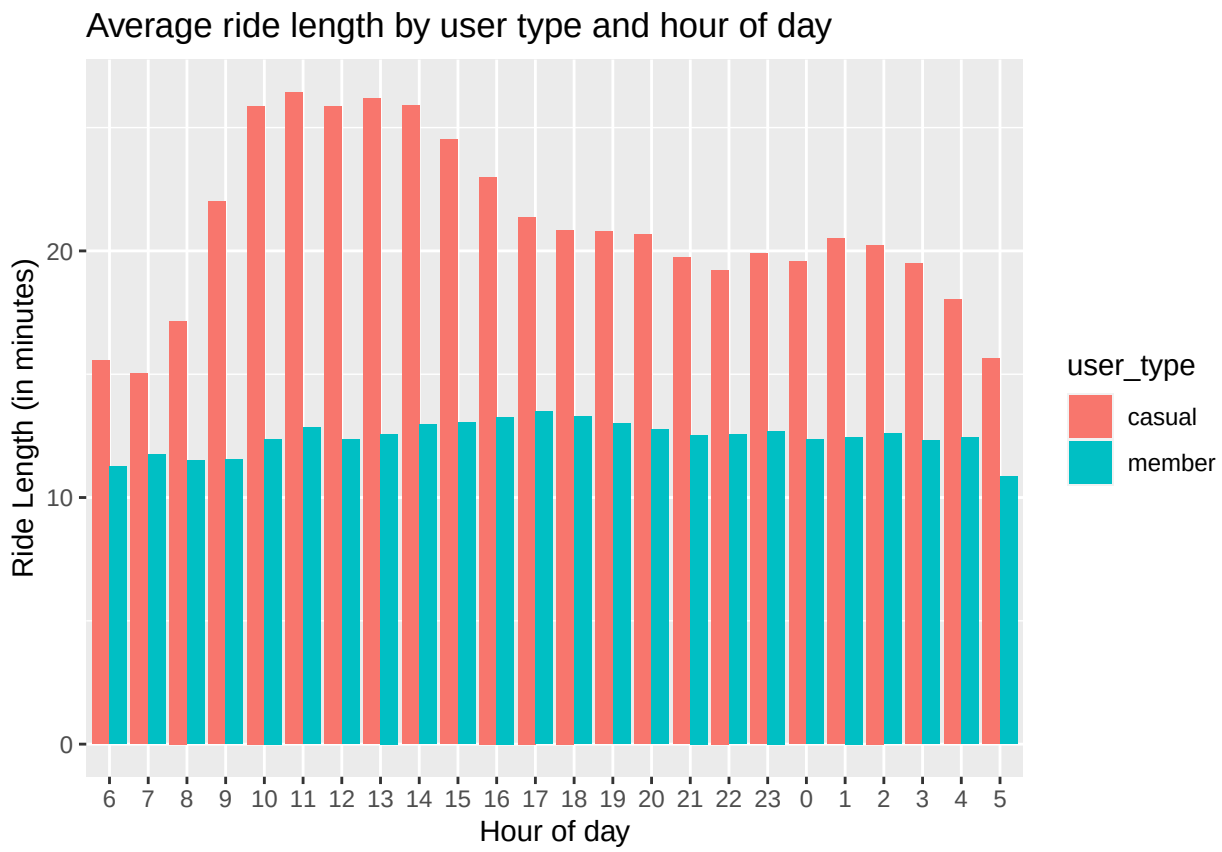
```
bike_rides_v2 %>% filter(ride_length_cat == "< 12 min" | ride_length_cat== "12-20 min") %>%
  group_by(user_type) %>%
  summarize(avg_ride_length=mean(ride_length_min))
```

```
## # A tibble: 2 x 2
##   user_type avg_ride_length
##   <chr>               <dbl>
## 1 casual               10.3
## 2 member                8.77
```

Visualizing average ride length of each trip (in minutes) by user type and hour of day

```
bike_rides_v2 %>%
  group_by(user_type, hour) %>%
  summarise(count = n(), average_ride_length=mean(ride_length_min)) %>%
  arrange(user_type, hour) %>%
  ggplot(aes(x=factor(hour, level= c(6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,0,1,2,3,4,5)), y=
  geom_col(position = "dodge") +
  labs(x="Hour of day", y="Ride Length (in minutes)", title = "Average ride length by user type and hou
```

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



Key insight:

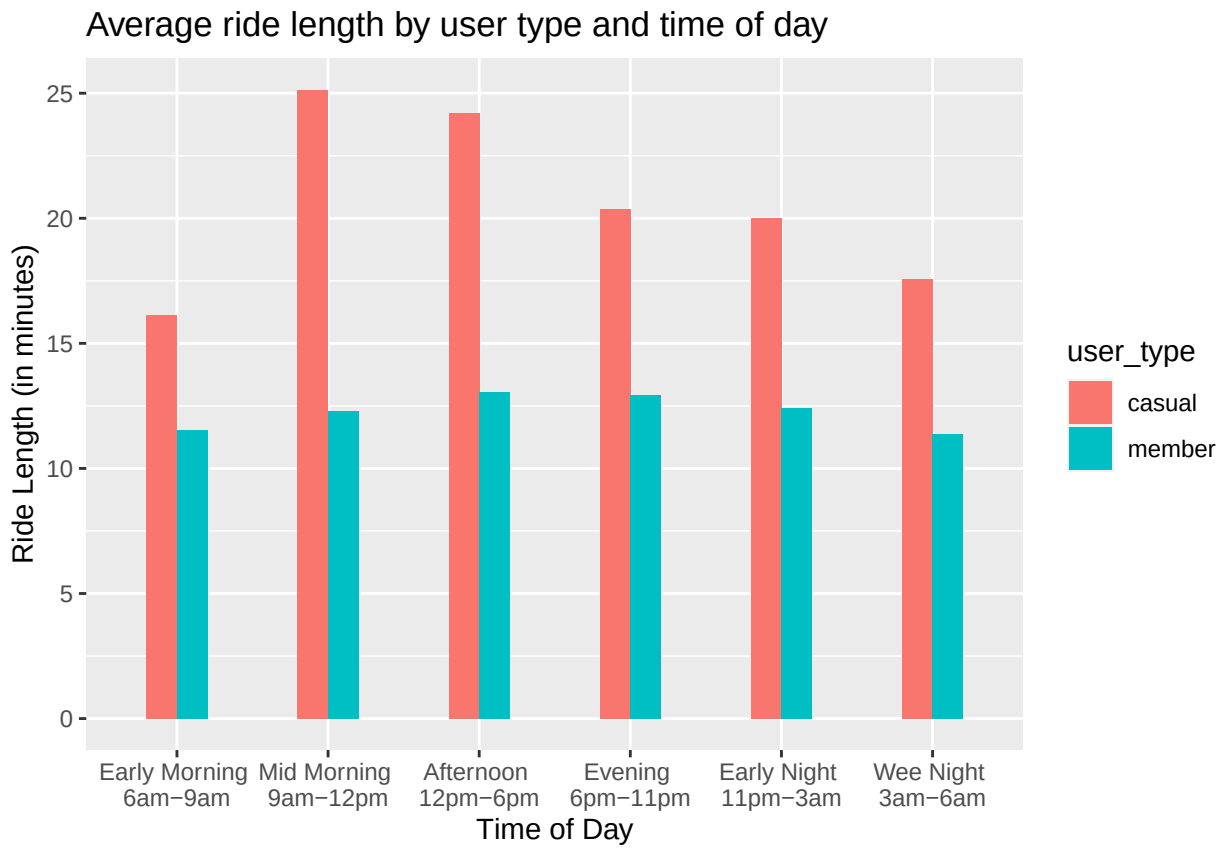
- Casual riders average longer rides than member riders, peaking between 10AM - 2PM

Visualizing average ride length of each trip (in minutes) by user type and time of day

```
axis_labels <- c("Early Morning \n6am-9am", "Mid Morning \n9am-12pm", "Afternoon \n12pm-6pm", "Evening

bike_rides_v2 %>%
  group_by(user_type, time_of_day) %>%
  summarise(count = n(), average_ride_length=mean(ride_length_min)) %>%
  ggplot(aes(x=factor(time_of_day, level= c("Early Morning", "Mid Morning", "Afternoon", "Evening", "Ea
  geom_col(position = "dodge", width = 0.4) +
  labs(x="Time of Day", y="Ride Length (in minutes)", title = "Average ride length by user type and tim
  scale_x_discrete(labels = axis_labels)
```

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



Key insights:

- Ride length for casual riders peaks mid morning through afternoon
- Ride length for member riders remains more steady throughout the day

Average ride length of each trip (in minutes) by user type and day of the week

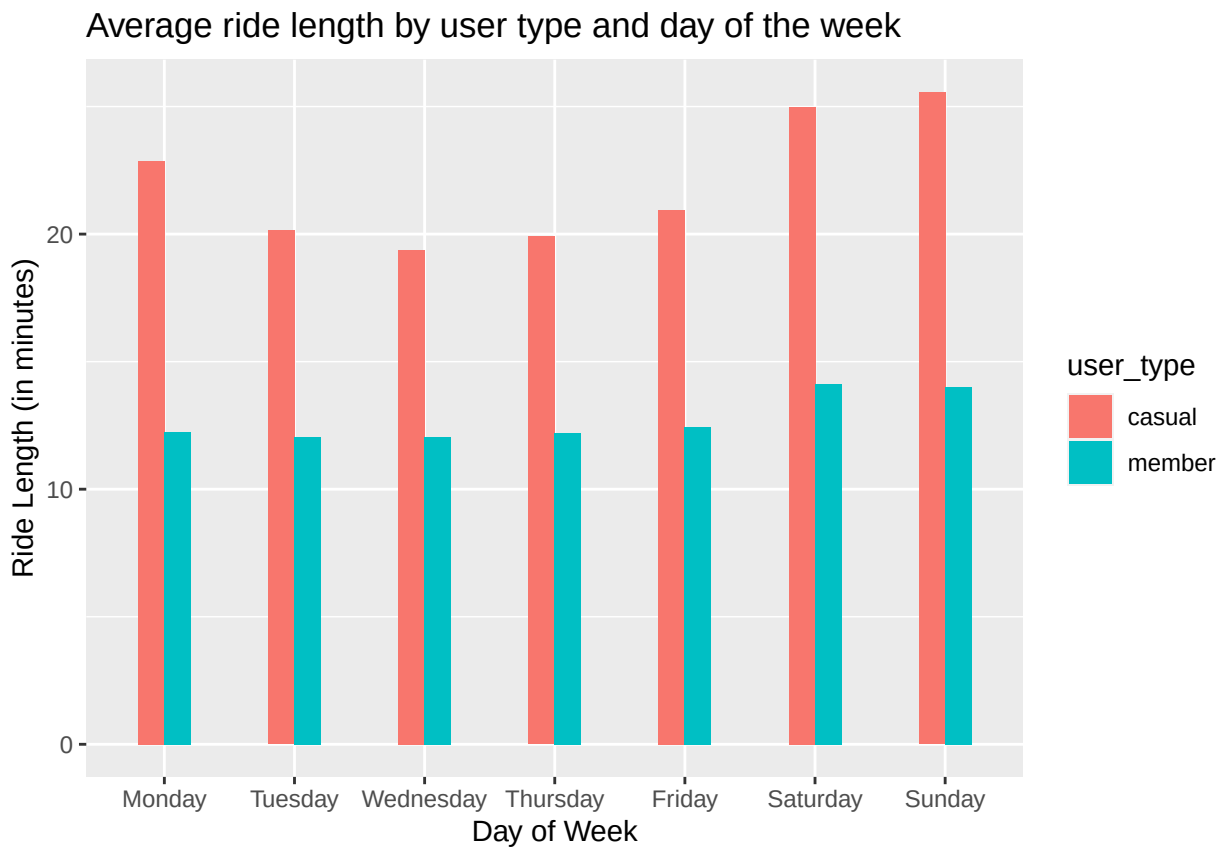
```
aggregate(bike_rides_v2$ride_length_min ~ bike_rides_v2$user_type + bike_rides_v2$day_of_week, FUN = mea
```

```
##    bike_rides_v2$user_type bike_rides_v2$day_of_week bike_rides_v2$ride_length_min
## 1                   casual                   Friday                      20.95016
## 2                   member                   Friday                      12.44000
## 3                   casual                   Monday                      22.87568
## 4                   member                   Monday                      12.23843
## 5                   casual                 Saturday                      24.97175
## 6                   member                 Saturday                      14.13371
## 7                   casual                   Sunday                      25.55731
## 8                   member                   Sunday                      14.00332
## 9                   casual                 Thursday                      19.90399
## 10                  member                 Thursday                      12.20314
## 11                  casual                  Tuesday                      20.14269
## 12                  member                  Tuesday                      12.05262
## 13                  casual                Wednesday                      19.37567
## 14                  member                Wednesday                      12.04318
```

Visualizing average ride length of each trip (in minutes) by user type and day of the week

```
bike_rides_v2 %>%
  group_by(user_type, day_of_week) %>%
  summarise(count = n(), average_ride_length=mean(ride_length_min)) %>%
  ggplot(aes(x=factor(day_of_week, level= c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Sa
  geom_col(position = "dodge", width = 0.4) +
  labs(x="Day of Week", y="Ride Length (in minutes)", title = "Average ride length by user type and day
```

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



Key insight:

- Both riders take longer rides on weekends

4.4 Recap of analysis on riders and ride length so far

- My data set consists of 5,621,147 rides
- 2,295,882 (41%) of these rides are casual riders
- 3,325,265 (59%) of these rides are member riders
- Both casual and member riders tend to use the classic and electric bikes
- Member riders take more rides
- Casual riders average longer rides
- Average of all ride lengths is 16.63 minutes
- Average ride length for all casual riders is 22.37 minutes
- Average ride length for all member riders is 12.65 minutes
- 5,453,815 (97%) of all rides are <= 60 min
- 4,428,315 (79%) of all rides are <= 20 min
- 3,151,382 (56%) of all rides are < 12 min

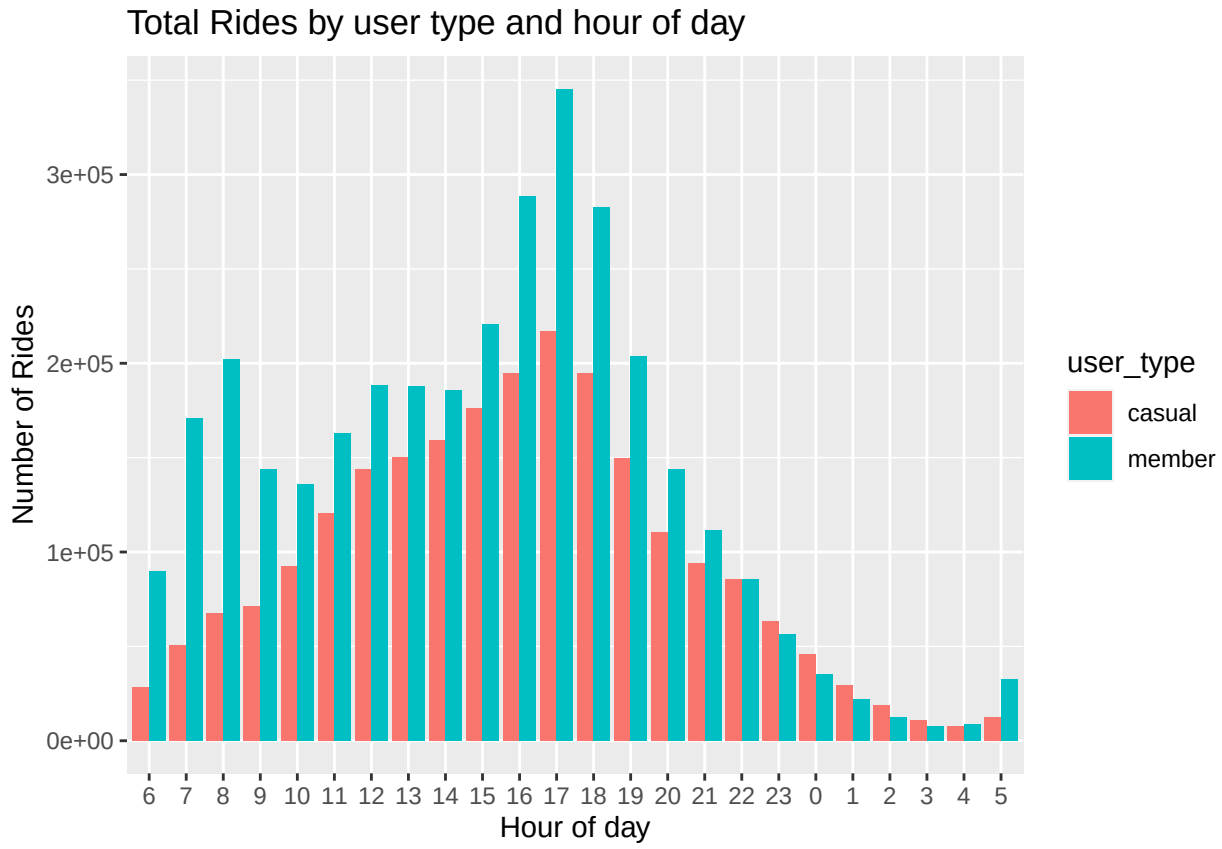
When focused on ride lengths under 12 minutes: * 67% of these riders are member riders * Average ride length for casual riders in the "<12 min" category is 7.17 minutes * Average ride length for member riders in the "<12 min" category is 6.44 minutes

When focused on ride lengths <= 20 minutes: * 64% of these riders are member riders * Average ride length for casual riders in the "<=20 min" category is 10.27 minutes * Average ride length for member riders in the "<=20 min" category is 8.76 minutes

4.5 Analyzing total rides by user type and hour of the day

```
bike_rides_v2 %>%
  group_by(user_type, hour) %>%
  summarise(count = n()) %>%
  arrange(user_type, hour) %>%
  ggplot(aes(x=factor(hour, level= c(6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,0,1,2,3,4,5)), y=
  geom_col(position = "dodge") +
  labs(x="Hour of day", y="Number of Rides", title = "Total Rides by user type and hour of day")
```

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



Key Insights:

- Member riders show a strong use of rides starting at 6am and peaking at 8am and then again peaking further from 4pm to 6pm.

- Casual rides also peak from 4pm to 6 pm. Member riders show a strong use of rides starting at 6am and peaking at 8am and then again peaking further from 4pm to 6pm. Casual rides also peak from 4pm to 6 pm.

Total rides by user type and by time of day

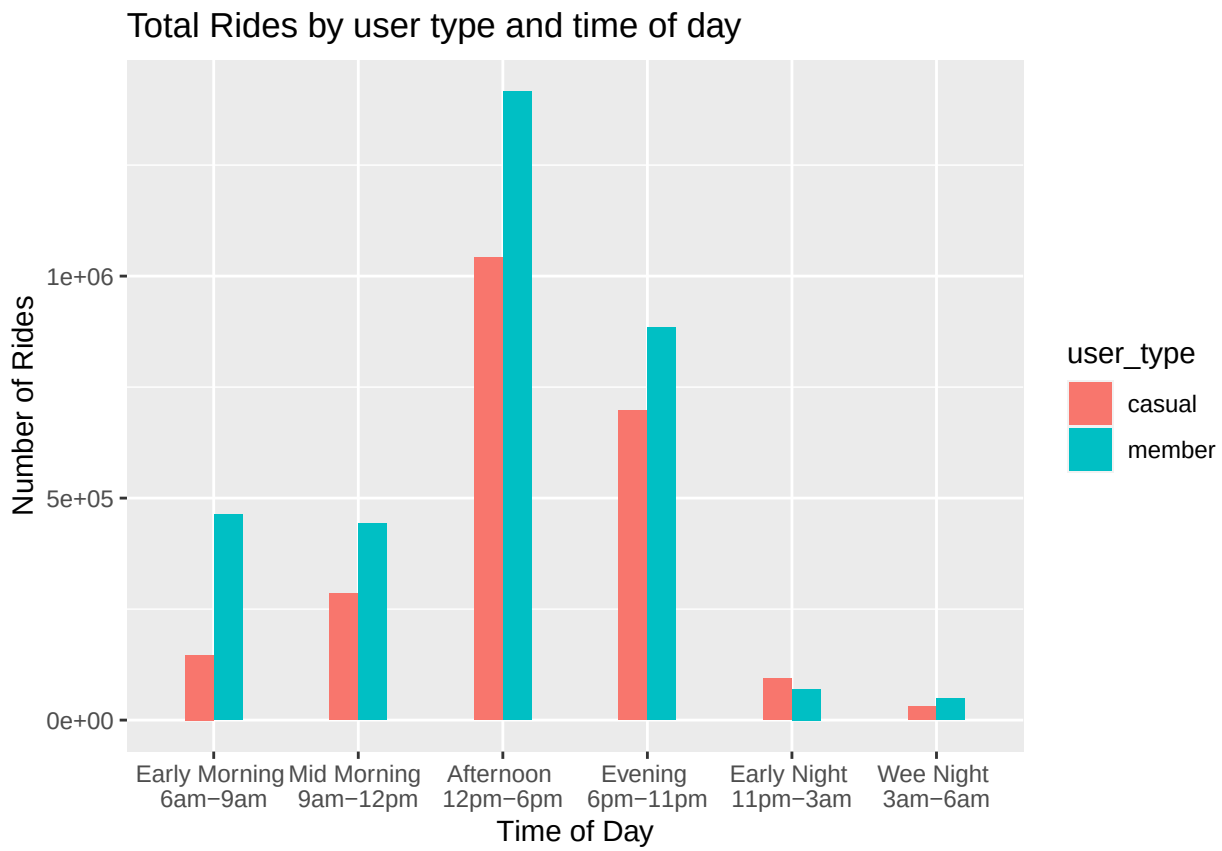
```
bike_rides_v2 %>%
   group_by(user_type) %>%
  summarize("Early Morning" = sum(time_of_day == "Early Morning"),
            "Mid Morning" = sum(time_of_day == "Mid Morning"),
            "Afternoon" = sum(time_of_day == "Afternoon"),
            "Evening" = sum(time_of_day == "Evening"),
            "Early Night" = sum(time_of_day == "Early Night"),
            "Late Night" = sum(time_of_day == "Late Night"))
```

```
## # A tibble: 2 x 7
##   user_type `Early Morning` `Mid Morning` Afternoon Evening `Early Night` `Late Night`
##   <chr>               <int>         <int>     <int>   <int>         <int>        <int>
## 1 casual             146823        284655   1041548  698017         93909        30898
## 2 member             462623        442804   1416436  884146         69939        49291
```

Visualizing total rides by user type and by time of day

```
bike_rides_v2 %>%
  group_by(user_type, time_of_day) %>%
  summarise(count = n()) %>%
  ggplot(aes(x=factor(time_of_day, level= c("Early Morning", "Mid Morning", "Afternoon", "Evening", "Ea
  geom_col(position = "dodge", width = 0.4) +
  labs(x="Time of Day", y="Number of Rides", title = "Total Rides by user type and time of day") +
  scale_x_discrete(labels = axis_labels)
```

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



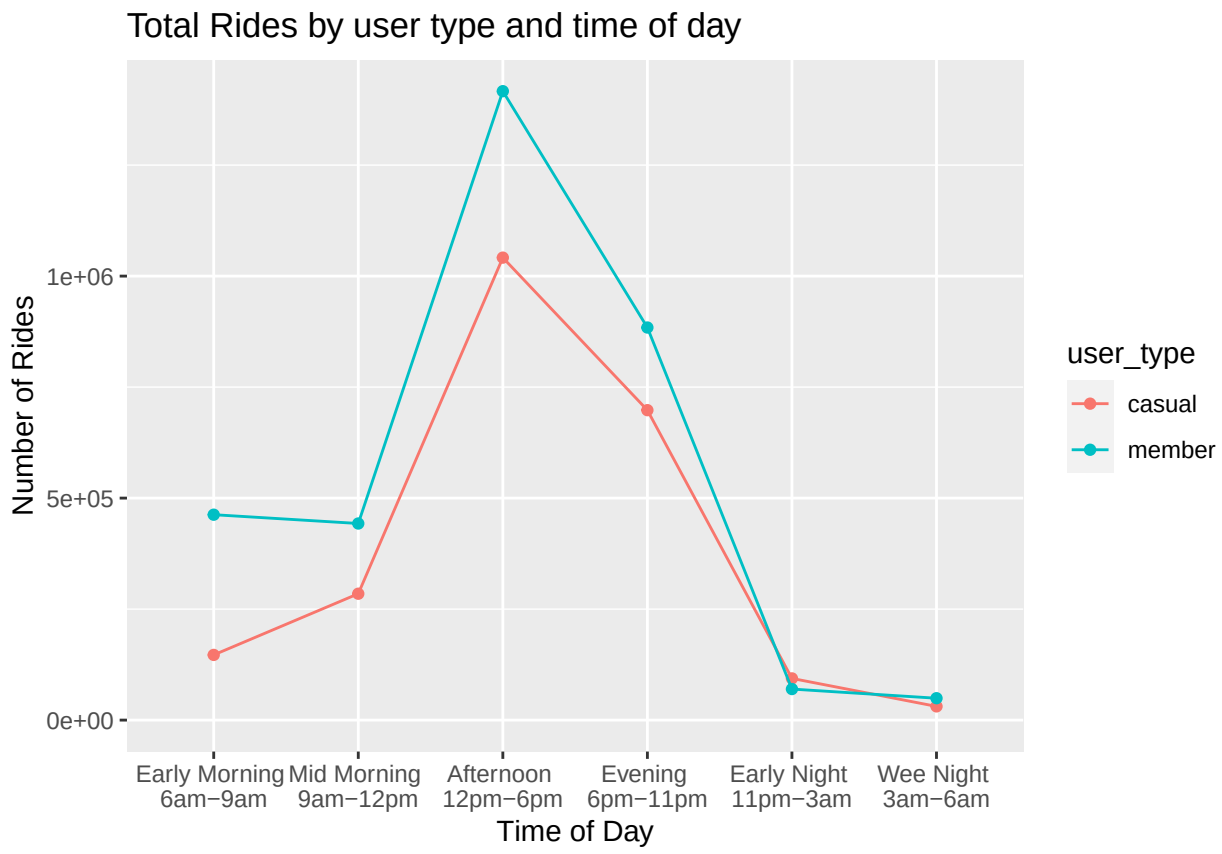
Key insight:

- 79% of early morning rides are taken by member riders.
- Rides peak in the afternoon and evening for both riders.

Another visual for total rides by user type and by time of day

```
bike_rides_v2 %>%
  group_by(user_type, time_of_day) %>%
  summarise(count = n()) %>%
  ggplot(aes(x=factor(time_of_day, level= c("Early Morning", "Mid Morning", "Afternoon", "Evening", "Ea
  geom_point() + geom_line(aes(group = user_type)) +
  labs(x="Time of Day", y="Number of Rides", title = "Total Rides by user type and time of day") + ylim
  scale_x_discrete(labels = axis_labels)
```

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



4.6 Analyzing total rides by user type and day of the week

Total rides by user type and day of the week

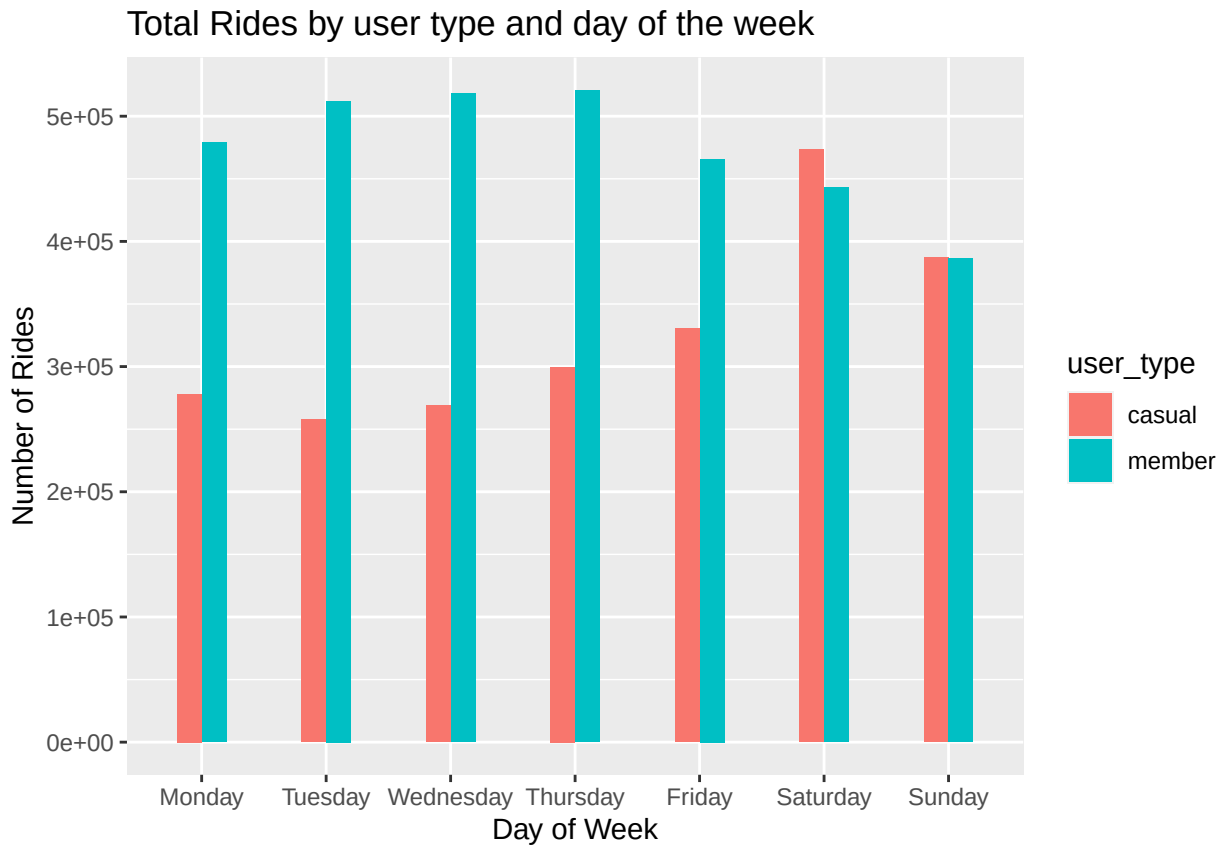
```
bike_rides_v2 %>%
   group_by(user_type) %>%
  summarize("Monday" = sum(day_of_week == "Monday"),
            "Tuesday" = sum(day_of_week == "Tuesday"),
            "Wednesday" = sum(day_of_week == "Wednesday"),
            "Thursday" = sum(day_of_week == "Thursday"),
            "Friday" = sum(day_of_week == "Friday"),
            "Saturday" = sum(day_of_week == "Saturday"),
            "Sunday" = sum(day_of_week == "Sunday"))
```

```
## # A tibble: 2 x 8
##   user_type Monday Tuesday Wednesday Thursday Friday Saturday Sunday
##   <chr>      <int>   <int>     <int>    <int>  <int>    <int>  <int>
## 1 casual    278121  257883    268819   299597 330654   473440 387336
## 2 member    478883  512240    517927   520770 465840   443277 386302
```

Visualizing total rides by user type and day of the week

```
bike_rides_v2 %>%
  group_by(user_type, day_of_week) %>%
  summarise(count = n()) %>%
  ggplot(aes(x=factor(day_of_week, level= c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Sa
  geom_col(position = "dodge", width = 0.4) +
  labs(x="Day of Week", y="Number of Rides", title = "Total Rides by user type and day of the week")
```

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



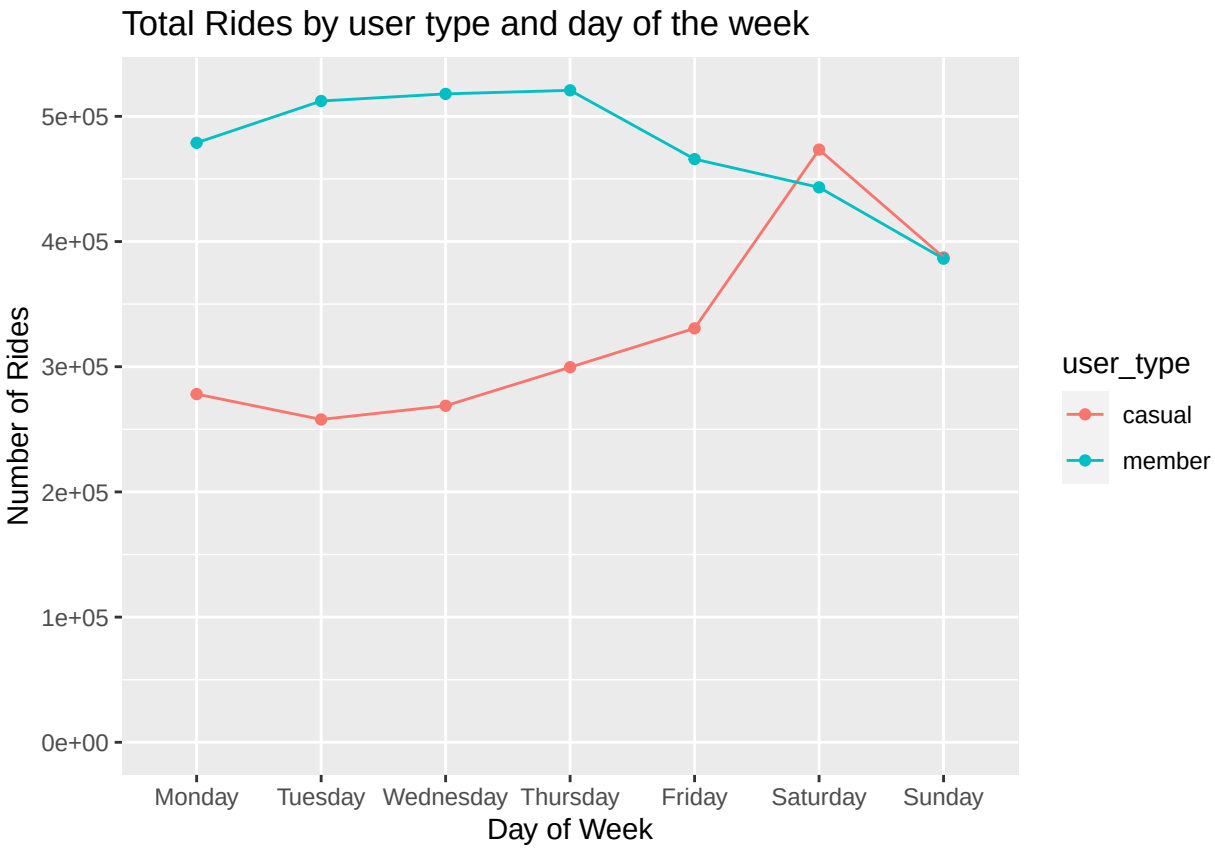
Key insight:

- Member riders take the most rides between Monday through Thursday and decline a little over the weekend * Casual riders take most rides on the weekend.

Another visualization of total rides by user type and day of the week

```
bike_rides_v2 %>%
  group_by(user_type, day_of_week) %>%
  summarise(count = n()) %>%
  ggplot(aes(x=factor(day_of_week, level= c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Sa
  geom_point() + geom_line(aes(group = user_type)) +
  labs(x="Day of Week", y="Number of Rides", title = "Total Rides by user type and day of the week") +
   ylim(0, NA)
```

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

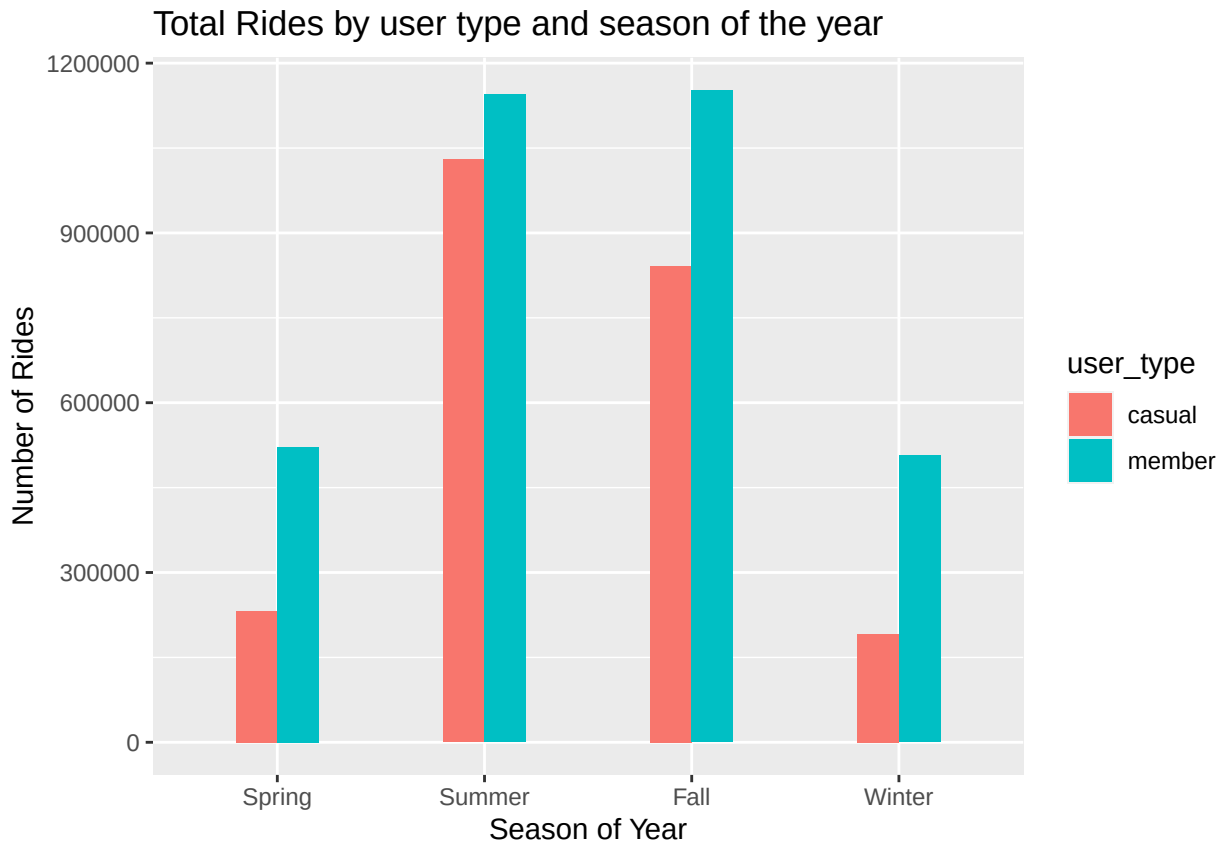


4.7 Analyzing total rides by user type and season

Visualizing total rides by user type and season

```
bike_rides_v2 %>%
  group_by(user_type, season) %>%
  summarise(count = n()) %>%
  ggplot(aes(x=factor(season, level= c("Spring", "Summer", "Fall", "Winter")), y=count, fill=user_type)
  geom_col(position = "dodge", width = 0.4) +
  labs(x="Season of Year", y="Number of Rides", title = "Total Rides by user type and season of the yea
```

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



Key insights:

- In each season we see more member rides.
- Both riders peak in summer and decline in winter.

4.8 Analyzing the top five starting and ending stations by user types

Top five starting stations for casual riders

```
bike_rides_v2 %>%
  filter(!(is.na(start_station_name))) %>%
  filter(user_type == "casual") %>%
  group_by(start_station_name) %>%
  summarize(count=n()) %>%
  arrange(-count) %>%
  top_n(5)
```

```
## Selecting by count
```

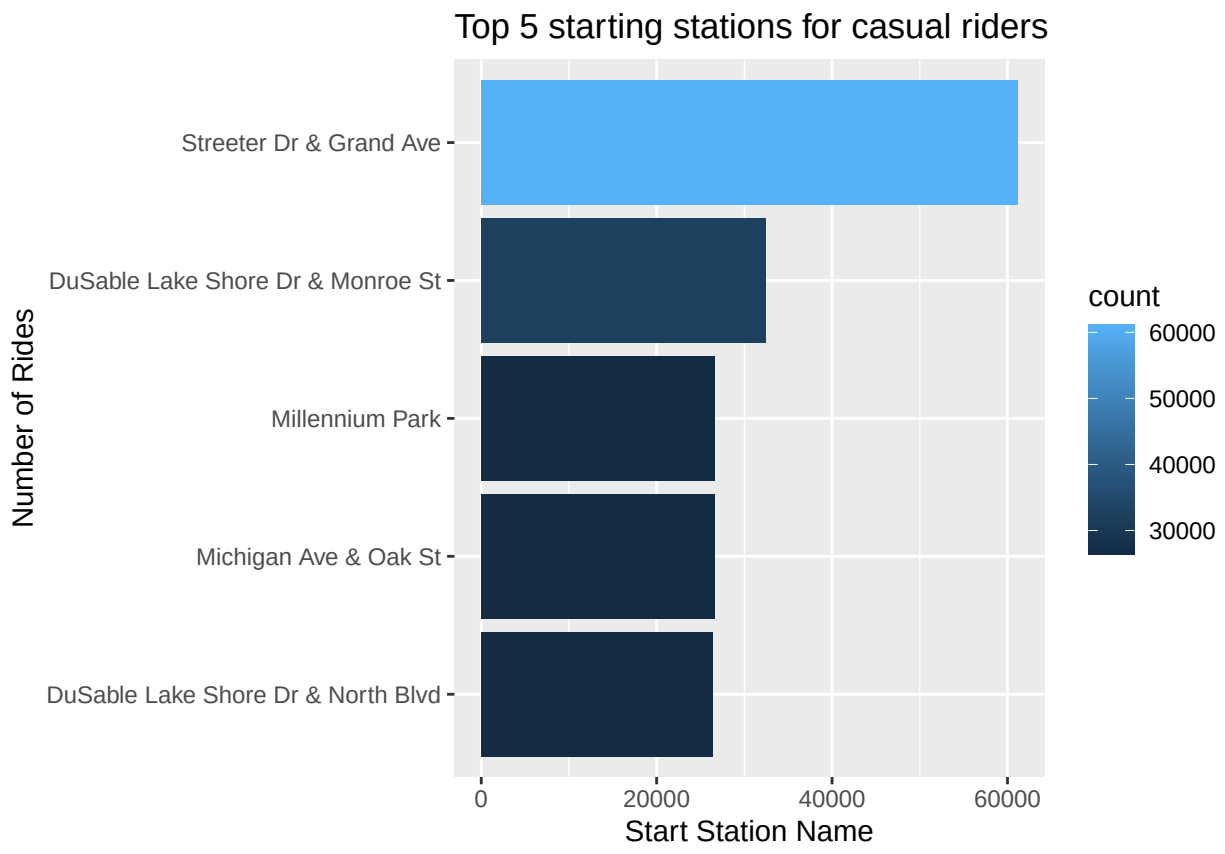
```
## # A tibble: 5 x 2
##   start_station_name                count
##   <chr>                             <int>
## 1 Streeter Dr & Grand Ave           61167
## 2 DuSable Lake Shore Dr & Monroe St  32478
```

```
## 3 Millennium Park                     26660
## 4 Michigan Ave & Oak St               26633
## 5 DuSable Lake Shore Dr & North Blvd 26398
```

Visualizing top five starting stations for casual riders

```
bike_rides_v2 %>%
  filter(!(is.na(start_station_name))) %>%
  filter(user_type == "casual") %>%
  group_by(start_station_name) %>%
  summarize(count=n()) %>%
  arrange(-count) %>%
  top_n(5) %>%
  mutate(start_station_name= fct_reorder(start_station_name, count)) %>%
  ggplot(aes(x=start_station_name, y=count, fill=count)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x="Number of Rides", y="Start Station Name", title="Top 5 starting stations for casual riders")
```

```
## Selecting by count
```



Top five ending stations for casual riders

```r
bike_rides_v2 %>%
  filter(!(is.na(end_station_name))) %>%
  filter(user_type == "casual") %>%
  group_by(end_station_name) %>%
  summarize(count=n()) %>%
  arrange(-count) %>%
  top_n(5)
```

```
## Selecting by count
```

```
## # A tibble: 5 x 2
##   end_station_name                  count
##   <chr>                             <int>
## 1 Streeter Dr & Grand Ave           63730
## 2 DuSable Lake Shore Dr & Monroe St 30864
## 3 Millennium Park                   28074
## 4 Michigan Ave & Oak St             27578
## 5 DuSable Lake Shore Dr & North Blvd 27091
```

Key insight:

- The top 5 starting and ending stations are the same for casual riders.
- The top starting and ending station for casual riders is Streeter Dr & Grand Ave - situated near a park and shoreline sightseeing - vacationers are likely visiting this area.

Top five starting stations for member riders

```r
bike_rides_v2 %>%
  filter(!(is.na(start_station_name))) %>%
  filter(user_type == "member") %>%
  group_by(start_station_name) %>%
  summarize(count=n()) %>%
  arrange(-count) %>%
  top_n(5)
```

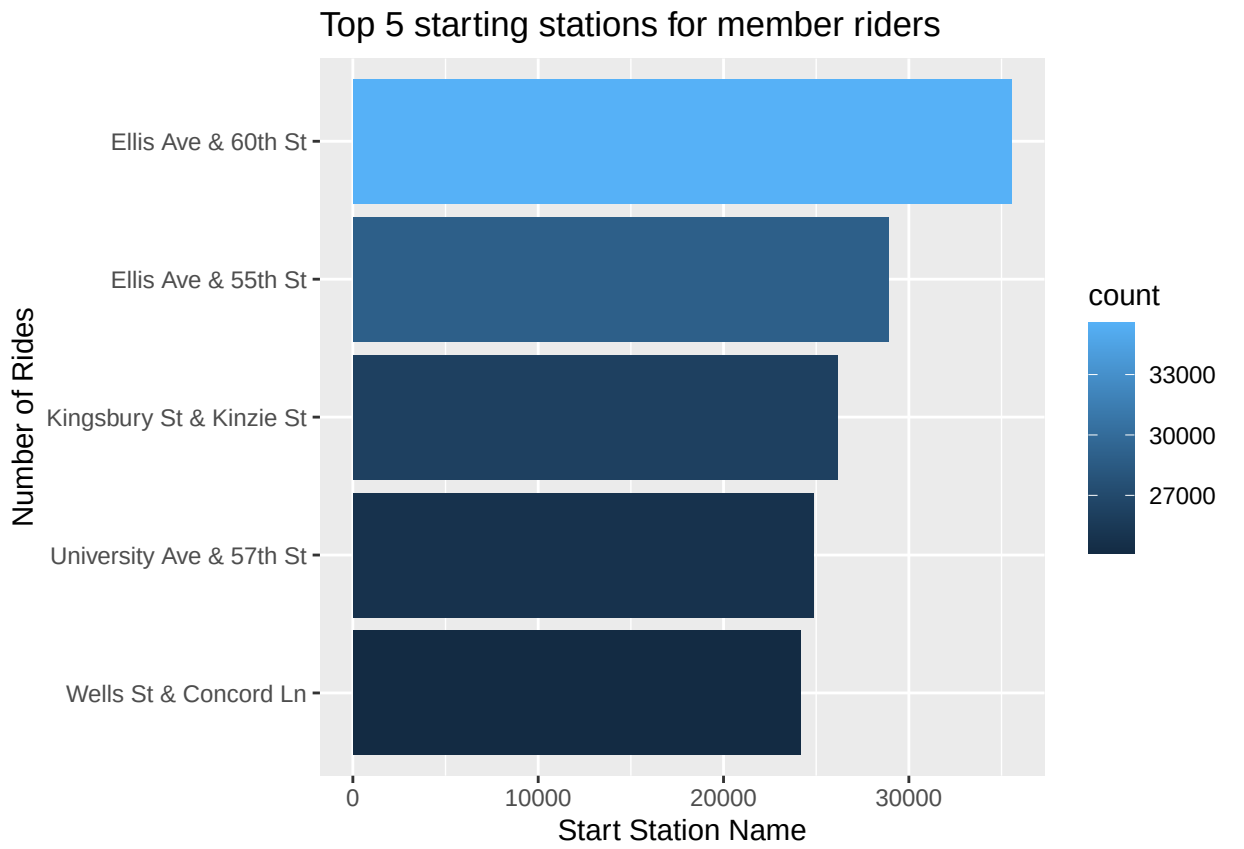
```
## Selecting by count
```

```
## # A tibble: 5 x 2
##   start_station_name       count
##   <chr>                    <int>
## 1 Ellis Ave & 60th St      35543
## 2 Ellis Ave & 55th St      28910
## 3 Kingsbury St & Kinzie St 26156
## 4 University Ave & 57th St 24859
## 5 Wells St & Concord Ln    24136
```

Visualizing top five starting stations for member riders

```r
bike_rides_v2 %>%
  filter(!(is.na(start_station_name))) %>%
  filter(user_type == "member") %>%
  group_by(start_station_name) %>%
  summarize(count=n()) %>%
  arrange(-count) %>%
  top_n(5) %>%
  mutate(start_station_name= fct_reorder(start_station_name, count)) %>%
  ggplot(aes(x=start_station_name, y=count, fill=count)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x="Number of Rides", y="Start Station Name", title="Top 5 starting stations for member riders")
```

```
## Selecting by count
```



Top five ending stations for member riders

```r
bike_rides_v2 %>%
  filter(!(is.na(end_station_name))) %>%
  filter(user_type == "member") %>%
  group_by(end_station_name) %>%
  summarize(count=n()) %>%
  arrange(-count) %>%
  top_n(5)
```

```
## Selecting by count
```

```
## # A tibble: 5 x 2
##   end_station_name         count
##   <chr>                    <int>
## 1 Ellis Ave & 60th St      36767
## 2 University Ave & 57th St 36523
## 3 Kingsbury St & Kinzie St 26479
## 4 Wells St & Concord Ln    25034
## 5 Clark St & Elm St        24106
```

Key insight:

- The top 5 starting and ending stations are the same for member riders except for one station.
- The top starting and ending station for member riders is Ellis Ave & 60th St which is situated within the University of Chicago - this suggests that students, faculty and staff are likely using this system.

4.9 Export summary file for further analysis

Creating a csv file that can be exported

```
write_csv(bike_rides_v2, file = "bike_rides_v2.csv")
```

Step 5 : Share your findings with your audience

For purposes of this project, I will knit this Rmd file to PDF and then share to a public Github repository.

Step 6: Act on the data and use the analysis results

This step entails summarizing the findings of the investigation and making recommendations to address the business questions.

6.1 Conclusions from the analysis

- 97% of all rides are 60 minutes or less
- 79% of all rides are 20 minutes or less
- 56% of all rides are less than 12 minutes
- Looking strictly at ride lengths under 12 minutes, 67% of these riders are member riders which is double the casual riders in this category.
- Looking strictly at ride lengths <=20 minutes, 64% of these riders are member riders.
- Member riders make up the majority of riders, take more rides, take shorter rides and the number of their rides peaks Monday through Thursday, declining a little over the weekend.
- 79% of early morning rides are member riders.

- The top starting station for member riders is Ellis Ave & 60th St - situated within the University of Chicago - students, faculty and staff are likely using this system
- Casual riders average longer rides, take more rides on the weekends and their rides peak in summer.
- The top starting station for casual riders is Streeter Dr & Grand Ave - situated near a park and shoreline sightseeing - vacationers are likely visiting this area.
- In each season we see more member rides, but both riders peak in summer and decline in winter.

6.2 Additional data analysis to consider

The public dataset utilized for this study has limitations due to user privacy protection. In an ideal world, the marketing team would use the entire dataset to conduct the following extra analysis:

- Determining which casual riders are locals and paying attention to their riding patterns.
- Separating single rider and day pass riders.
- How many of each category do we have?
- What is their riding style? Who rides many times every day? What are their plans?
- Examine the number and length of journeys taken by local casual riders.
- Examining member riders' riding habits more closely.
- The number of journeys, the length of the travels, and who rides many times a day? What are their plans?
- What is the surrounding scenery like for both member bikers and local casual riders?
- Do they reside in a densely populated area?
- Are there any significant employers nearby?
- Are there any educational or medical facilities nearby?
- Determine whether the local infrastructure encourages cycling to work, riding to school, riding to grocery stores, eating establishments, shopping, coffee shops, and social events.
- Collect socioeconomic and cultural demographics to determine how these factors influence these two groups' riding behaviors. Use data to create a marketing plan.

6.3 Recommendations based on data insights

Marketing is a continuous process.As a first phase, consider a plan focused on the top ride length categories: < 12 minutes and <= 20 min. Consider which places surrounding the bike stations will best complement/support/facilitate this trip length.

- Raise awareness of the convenience, affordability, accessibility, green options, and common uses for bike rides <=20 minutes
- Market to people whose home area and daily travel needs likely align well with your bike system. Focusing on dense and walkable areas, partnering with schools and companies.
  - Use social media to highlight member riders who are already riding their bikes to work, school, and the gym, and to meet friends.

  - To market sponsored member subscriptions to students and employees, partner with schools and businesses inside your bike system where your bike system may facilitate <12- to 20 minute rides.

- Furthermore, there is an opportunity to market around pricing. Calculate each rider's average usage and rider fees to identify who might benefit from enrolling in the annual membership plan. As an example:
  - One-of-a-Kind Day Pass Riders that utilize the Day Pass more than eight times each year
  - Unique Single Riders who bike more than 34 Single Rides in a calendar year or who cycle at least 3 Single Rides each month
  - Reach out to casual riders who spend money on fees and overage costs and would benefit much from converting to an annual membership. While we lose money on this member in the short term, we will build loyalty, and this consumer is likely to recommend our services.
- Offer a prize for recruiting a friend to bike to work or school for members who already ride to work or school.

Conclusion

Thank you for taking the time to look at my capstone project! Using real-world data and a business problem, this project guided me through the data analysis process from beginning to end. It's incredibly pleasant to accomplish this in R and RStudio. I'm ecstatic and eager to advance in the field of data analysis.