Cyclistic Bike-Share Analysis Case Study [R]

Mohani Lal Bhurtel

2025-01-22

1. Introduction

This case study is the Capstone Project of Google Data Analytics Professional Certificate . In this case study I am working as a junior data analyst in the marketing analyst team at Cyclistic, a fictional bike-share company in Chicago.

Cyclistic operates a bike-share program with more than 5,800 bicycles and 600 docking stations across Chicago. The company stands out by offering a diverse fleet, including reclining bikes, hand tricycles, and cargo bikes, which makes bike-sharing more accessible to people with disabilities and riders who cannot use standard two-wheeled bikes. While the majority of Cyclistic's riders use traditional bikes, approximately 8% opt for assistive options. Cyclistic riders are typically leisure users, but around 30% use the bikes for commuting to work.

Cyclistic launched its successful bike-share offering in 2016, and since then, the program has expanded to include 5,824 bicycles, which are geotracked and locked into a network of 692 stations. Riders can unlock bikes from one station and return them to any other station in the system at any time.

The **Marketing Director** believes that the future success of Cyclistic depends on increasing the number of **annual memberships**. Therefore, my team's objective is to understand the behavioral differences between casual riders and annual members, in order to design a marketing strategy that will encourage casual riders to convert to annual memberships. Our findings will be presented to the Cyclistic executive team, who must approve the recommendations. Our analysis will be supported by data-driven insights and professional data visualizations.

Cyclistic offers two types of pricing plans: **- Single-ride passes - Full-day passes - Annual memberships**

Customers who purchase single-ride or full-day passes are considered **Casual riders**, while customers who purchase annual memberships are classified as **Cyclistic members**.

To answer the key business questions, I followed the steps of the data analysis process: **ask**, **prepare**, **process**, **analyze**, **share**, and **act**.

2. Ask

The key question guiding this analysis is:

- **How do Annual members and Casual riders use Cyclistic bikes differently?**

Key Stakeholders:

- **Lily Moreno**, Director of Marketing (Manager)
- **Cyclistic Executive Team**

Objective:

- First, the Cyclistic executive team must approve our recommendations, which will be supported by data insights and visualizations.
- Based on these insights, my team will design a new marketing strategy aimed at converting casual riders into annual members.

3. Prepare

Data Description:

The data used in this analysis is the **Cyclistic Historical Trip Data**. This data covers the past 12 months, from **August 1, 2022, to July 31, 2023**, and is stored in CSV files, with each file representing one month of data. In total, there are **12 CSV files**.

The data is well-organized and structured, making it suitable for analysis. The datasets have different names to represent the various months of data, and these datasets are reliable, original, and current, as they have been provided by **Motivate International Inc.** under an appropriate license.

- **Reliability**: The data comes from a real bike-sharing company in Chicago, ensuring its authenticity.
- **Originality**: The data is directly sourced from Cyclistic's operational systems.
- **Current**: The dataset reflects data from the most recent 12-month period.
- **Credibility**: There are no issues with bias or credibility in the data.
- **Comprehensiveness**: While the data is quite comprehensive, it lacks certain information, such as data on specific bike types (e.g., reclining bikes, hand tricycles, cargo bikes), which are used by about 8% of riders.

Data Integrity:

- **Accurate**: The data is verified and accurately reflects the trip history.
- **Consistent**: The data is consistent across months and properly structured.
- **Trustworthy**: As it comes from a legitimate source, the data can be considered trustworthy.

Limitations:

There are a few limitations with the data: - **Data Privacy**: Riders' personally identifiable information (PII) is not included, so I cannot connect ride purchases to personal details such as addresses or credit card numbers. - **Financial Data**: Information about the fare for each ride is not available, limiting analysis of how much casual riders might spend compared to annual members. - **Bike Type Data**: The data does not specify the usage of reclining bikes, hand tricycles, and cargo bikes. However, it is known that about 8% of riders use these assistive bike options.

4. Process Load necessary library

```r
library(tidyverse) #helps wrangle data
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(conflicted)
library(janitor)
library(ggplot2)
```

Set dplyr::filter and dplyr::lag as the default choices

```r
conflict_prefer("filter", "dplyr")
```

```
## [conflicted] Will prefer dplyr::filter over any other package.
```

```r
conflict_prefer("lag", "dplyr")
```

```
## [conflicted] Will prefer dplyr::lag over any other package.
```

4.1 COLLECT DATA

```r
january_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202401-divvy-tripdata.c
```

```
## Rows: 144873 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
february_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202402-divvy-tripdata.
```

```
## Rows: 223164 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
march_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202403-divvy-tripdata.csv
```

```
## Rows: 301687 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
april_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202404-divvy-tripdata.csv
```

```
## Rows: 415025 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
may_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202405-divvy-tripdata.csv")
```

```
## Rows: 609493 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
june_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202406-divvy-tripdata.csv"
```

```
## Rows: 710721 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
july_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202407-divvy-tripdata.csv"
## Rows: 748962 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
august_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202408-divvy-tripdata.cs
```

```
## Rows: 755639 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
september_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202409-divvy-tripdata
```

```
## Rows: 821276 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
october_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202410-divvy-tripdata.c
```

```
## Rows: 616281 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
november_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202411-divvy-tripdata.
```

```
## Rows: 335075 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
december_2024 <- read_csv("C:/Users/Ashis/OneDrive/Desktop/Coursera/Project 2024/202412-divvy-tripdata.
```

```
## Rows: 178372 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

4.2 WRANGLE DATA AND COMBINE INTO A SINGLE FILE

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

```
colnames(january_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "start_station_id"
##  [7] "end_station_name"   "end_station_id"     "start_lat"
## [10] "start_lng"          "end_lat"            "end_lng"
## [13] "member_casual"
```

```
colnames(february_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "start_station_id"
##  [7] "end_station_name"   "end_station_id"     "start_lat"
## [10] "start_lng"          "end_lat"            "end_lng"
## [13] "member_casual"
```

```
colnames(march_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "start_station_id"
##  [7] "end_station_name"   "end_station_id"     "start_lat"
## [10] "start_lng"          "end_lat"            "end_lng"
## [13] "member_casual"
```

```r
colnames(april_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"        
##  [4] "ended_at"           "start_station_name" "start_station_id"  
##  [7] "end_station_name"   "end_station_id"     "start_lat"         
## [10] "start_lng"          "end_lat"            "end_lng"           
## [13] "member_casual"
```

```r
colnames(may_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"        
##  [4] "ended_at"           "start_station_name" "start_station_id"  
##  [7] "end_station_name"   "end_station_id"     "start_lat"         
## [10] "start_lng"          "end_lat"            "end_lng"           
## [13] "member_casual"
```

```r
colnames(june_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"        
##  [4] "ended_at"           "start_station_name" "start_station_id"  
##  [7] "end_station_name"   "end_station_id"     "start_lat"         
## [10] "start_lng"          "end_lat"            "end_lng"           
## [13] "member_casual"
```

```r
colnames(july_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"        
##  [4] "ended_at"           "start_station_name" "start_station_id"  
##  [7] "end_station_name"   "end_station_id"     "start_lat"         
## [10] "start_lng"          "end_lat"            "end_lng"           
## [13] "member_casual"
```

```r
colnames(august_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"        
##  [4] "ended_at"           "start_station_name" "start_station_id"  
##  [7] "end_station_name"   "end_station_id"     "start_lat"         
## [10] "start_lng"          "end_lat"            "end_lng"           
## [13] "member_casual"
```

```r
colnames(september_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"        
##  [4] "ended_at"           "start_station_name" "start_station_id"  
##  [7] "end_station_name"   "end_station_id"     "start_lat"         
## [10] "start_lng"          "end_lat"            "end_lng"           
## [13] "member_casual"
```

```r
colnames(october_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "start_station_id"
##  [7] "end_station_name"   "end_station_id"     "start_lat"
## [10] "start_lng"          "end_lat"            "end_lng"
## [13] "member_casual"
```

```r
colnames(november_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "start_station_id"
##  [7] "end_station_name"   "end_station_id"     "start_lat"
## [10] "start_lng"          "end_lat"            "end_lng"
## [13] "member_casual"
```

```r
colnames(december_2024)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "start_station_id"
##  [7] "end_station_name"   "end_station_id"     "start_lat"
## [10] "start_lng"          "end_lat"            "end_lng"
## [13] "member_casual"
```

Stack all individual month's data frames into one big data frame.

```r
all_trips <- bind_rows(january_2024,february_2024,march_2024,april_2024,may_2024,june_2024,
                        july_2024,august_2024,september_2024,october_2024,november_2024,december_2024)
```

Remove lat, long as they are no needed for analysis

```r
all_trips <- all_trips %>%
  select(-c(start_lat, start_lng, end_lat, end_lng))
```

Inspect the new dataframe and look for incongruencies

```r
str(all_trips)
```

```
## tibble [5,860,568 x 9] (S3: tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:5860568] "C1D650626C8C899A" "EECD38BDB25BFCB0" "F4A9CE78061F17F7" "0A0
##  $ rideable_type     : chr [1:5860568] "electric_bike" "electric_bike" "electric_bike" "classic_bike
##  $ started_at        : POSIXct[1:5860568], format: "2024-01-12 15:30:27" "2024-01-08 15:45:46" ...
##  $ ended_at          : POSIXct[1:5860568], format: "2024-01-12 15:37:59" "2024-01-08 15:52:59" ...
##  $ start_station_name: chr [1:5860568] "Wells St & Elm St" "Wells St & Elm St" "Wells St & Elm St" "
##  $ start_station_id  : chr [1:5860568] "KA1504000135" "KA1504000135" "KA1504000135" "TA1305000030" .
##  $ end_station_name  : chr [1:5860568] "Kingsbury St & Kinzie St" "Kingsbury St & Kinzie St" "Kingsbu
##  $ end_station_id    : chr [1:5860568] "KA1503000043" "KA1503000043" "KA1503000043" "13193" ...
##  $ member_casual     : chr [1:5860568] "member" "member" "member" "member" ...
```

Drop NA fields

```r
all_trips <- drop_na(all_trips)
```

Convert ride_id and rideable_type to character so that they can stack correctly.

```r
all_trips <-  mutate(all_trips, ride_id = as.character(ride_id))
```

4.3 CLEAN UP AND ADD DATA TO PREPARE FOR ANALYSIS

Inspect the new table that has been created. List of column names

```r
colnames(all_trips)
```

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

How many rows are in data frame?

```r
nrow(all_trips)
```

```
## [1] 4208309
```

Dimensions of the data frame?

```r
dim(all_trips)
```

```
## [1] 4208309       9
```

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

```r
head(all_trips)
```

```
## # A tibble: 6 x 9
##   ride_id          rideable_type started_at          ended_at
##   <chr>            <chr>         <dttm>              <dttm>
## 1 C1D650626C8C899A electric_bike 2024-01-12 15:30:27 2024-01-12 15:37:59
## 2 EECD38BDB25BFCB0 electric_bike 2024-01-08 15:45:46 2024-01-08 15:52:59
## 3 F4A9CE78061F17F7 electric_bike 2024-01-27 12:27:19 2024-01-27 12:35:19
## 4 0A0D9E15EE50B171 classic_bike  2024-01-29 16:26:17 2024-01-29 16:56:06
## 5 33FFC9805E3EFF9A classic_bike  2024-01-31 05:43:23 2024-01-31 06:09:35
## 6 C96080812CD285C5 classic_bike  2024-01-07 11:21:24 2024-01-07 11:30:03
## # i 5 more variables: start_station_name <chr>, start_station_id <chr>,
## #   end_station_name <chr>, end_station_id <chr>, member_casual <chr>
```

See list of columns and data types (numeric, character, etc).

```r
str(all_trips)
```

```
## tibble [4,208,309 x 9] (S3: tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:4208309] "C1D650626C8C899A" "EECD38BDB25BFCB0" "F4A9CE78061F17F7" "0A0
##  $ rideable_type     : chr [1:4208309] "electric_bike" "electric_bike" "electric_bike" "classic_bike
##  $ started_at        : POSIXct[1:4208309], format: "2024-01-12 15:30:27" "2024-01-08 15:45:46" ...
##  $ ended_at          : POSIXct[1:4208309], format: "2024-01-12 15:37:59" "2024-01-08 15:52:59" ...
##  $ start_station_name: chr [1:4208309] "Wells St & Elm St" "Wells St & Elm St" "Wells St & Elm St" "
##  $ start_station_id  : chr [1:4208309] "KA1504000135" "KA1504000135" "KA1504000135" "TA1305000030" .
##  $ end_station_name  : chr [1:4208309] "Kingsbury St & Kinzie St" "Kingsbury St & Kinzie St" "Kingsb
##  $ end_station_id    : chr [1:4208309] "KA1503000043" "KA1503000043" "KA1503000043" "13193" ...
##  $ member_casual     : chr [1:4208309] "member" "member" "member" "member" ...
```

Statistical summary of data. Mainly for numerics.

```r
summary(all_trips)
```

```
##    ride_id          rideable_type        started_at                    
##  Length:4208309     Length:4208309     Min.   :2024-01-01 00:01:01.00  
##  Class :character   Class :character   1st Qu.:2024-05-18 07:55:28.00  
##  Mode  :character   Mode  :character   Median :2024-07-20 20:26:32.30  
##                                        Mean   :2024-07-15 06:53:24.81  
##                                        3rd Qu.:2024-09-17 14:39:35.17  
##                                        Max.   :2024-12-31 23:56:49.84  
##     ended_at                      start_station_name start_station_id  
##  Min.   :2024-01-01 00:07:01.00   Length:4208309     Length:4208309    
##  1st Qu.:2024-05-18 08:12:56.00   Class :character   Class :character  
##  Median :2024-07-20 20:48:15.22   Mode  :character   Mode  :character  
##  Mean   :2024-07-15 07:10:04.30                                        
##  3rd Qu.:2024-09-17 14:58:24.06                                        
##  Max.   :2024-12-31 23:59:28.81                                        
##  end_station_name   end_station_id     member_casual     
##  Length:4208309     Length:4208309     Length:4208309    
##  Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character  
##                                                          
##                                                          
## 
```

Add columns that list the date, month, day, and year of each ride. This will allow us to aggregate ride data for each month, day, or year … before completing these operations we could only aggregate at the ride level.

```r
all_trips$date <- as.Date(all_trips$started_at) #The default format is yyyy-mm-dd
all_trips$month <- format(as.Date(all_trips$date), "%m")
all_trips$day <- format(as.Date(all_trips$date), "%d")
all_trips$year <- format(as.Date(all_trips$date), "%Y")
all_trips$day_of_week <- format(as.Date(all_trips$date), "%A")
```

Add a "ride_length" calculation to all_trips (in seconds).

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

Inspect the structure of the columns.

```
str(all_trips)
```

```
## tibble [4,208,309 x 15] (S3: tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:4208309] "C1D650626C8C899A" "EECD38BDB25BFCB0" "F4A9CE78061F17F7" "0A0
##  $ rideable_type     : chr [1:4208309] "electric_bike" "electric_bike" "electric_bike" "classic_bike
##  $ started_at        : POSIXct[1:4208309], format: "2024-01-12 15:30:27" "2024-01-08 15:45:46" ...
##  $ ended_at          : POSIXct[1:4208309], format: "2024-01-12 15:37:59" "2024-01-08 15:52:59" ...
##  $ start_station_name: chr [1:4208309] "Wells St & Elm St" "Wells St & Elm St" "Wells St & Elm St" "
##  $ start_station_id  : chr [1:4208309] "KA1504000135" "KA1504000135" "KA1504000135" "TA1305000030" .
##  $ end_station_name  : chr [1:4208309] "Kingsbury St & Kinzie St" "Kingsbury St & Kinzie St" "Kingsb
##  $ end_station_id    : chr [1:4208309] "KA1503000043" "KA1503000043" "KA1503000043" "13193" ...
##  $ member_casual     : chr [1:4208309] "member" "member" "member" "member" ...
##  $ date              : Date[1:4208309], format: "2024-01-12" "2024-01-08" ...
##  $ month             : chr [1:4208309] "01" "01" "01" "01" ...
##  $ day               : chr [1:4208309] "12" "08" "27" "29" ...
##  $ year              : chr [1:4208309] "2024" "2024" "2024" "2024" ...
##  $ day_of_week       : chr [1:4208309] "Friday" "Monday" "Saturday" "Monday" ...
##  $ ride_length       : 'difftime' num [1:4208309] 452 433 480 1789 ...
##   ..- attr(*, "units")= chr "secs"
```

Convert "ride_length" from Factor to numeric so we can run calculations on the data.

```
is.factor(all_trips$ride_length)
```

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

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

is.numeric(all_trips$ride_length)
```

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

Remove "bad" data. The dataframe includes a few hundred entries when bikes were taken out of docks and checked for quality by Divvy or ride_length was negative. We will create a new version of the dataframe (v2) since data is being removed.

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

5.Analyze

Descriptive analysis on ride_length (all figures in seconds).

```
mean(all_trips_v2$ride_length) #straight average (total ride length / rides)
```

```
## [1] 999.5097
```

```r
median(all_trips_v2$ride_length) #midpoint number in the ascending array of ride lengths
```

```
## [1] 608.124
```

```r
max(all_trips_v2$ride_length) #longest ride
```

```
## [1] 90562
```

```r
min(all_trips_v2$ride_length) #shortest ride
```

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

Compare members and casual users.

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

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                1443.1385
## 2                     member                 748.2601
```

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

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                   804.64
## 2                     member                   529.00
```

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

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                    90562
## 2                     member                    89859
```

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

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                        0
## 2                     member                        0
```

See the average ride time by each day for members vs casual users.

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

```
##    all_trips_v2$member_casual all_trips_v2$day_of_week all_trips_v2$ride_length
## 1                      casual                   Friday                1394.2702
## 2                      member                   Friday                 726.3697
## 3                      casual                   Monday                1391.1536
## 4                   member              Monday             712.9400
## 5                   casual            Saturday            1636.3120
## 6                   member            Saturday             841.2603
## 7                   casual              Sunday            1657.6604
## 8                   member              Sunday             840.8780
## 9                   casual            Thursday            1255.5500
## 10                  member            Thursday             714.3331
## 11                  casual             Tuesday            1237.5168
## 12                  member             Tuesday             715.7616
## 13                  casual           Wednesday            1281.1430
## 14                  member           Wednesday             729.9058
```

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

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

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

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

```
##    all_trips_v2$member_casual all_trips_v2$day_of_week all_trips_v2$ride_length
## 1                      casual                   Sunday                1657.6604
## 2                      member                   Sunday                 840.8780
## 3                      casual                   Monday                1391.1536
## 4                      member                   Monday                 712.9400
## 5                      casual                  Tuesday                1237.5168
## 6                      member                  Tuesday                 715.7616
## 7                      casual                Wednesday                1281.1430
## 8                      member                Wednesday                 729.9058
## 9                      casual                 Thursday                1255.5500
## 10                     member                 Thursday                 714.3331
## 11                     casual                   Friday                1394.2702
## 12                     member                   Friday                 726.3697
## 13                     casual                 Saturday                1636.3120
## 14                     member                 Saturday                 841.2603
```

Analyze ridership data by type and weekday

```
all_trips_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%  #creates weekday field using wday()
  group_by(member_casual, weekday) %>%  #groups by usertype and weekday
  summarize(number_of_rides = n()  #calculates the number of rides and average duration
          ,average_duration = mean(ride_length)) %>%  # calculates the average duration
  arrange(member_casual, weekday)  # sorts the trip by member and weekdays
```

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

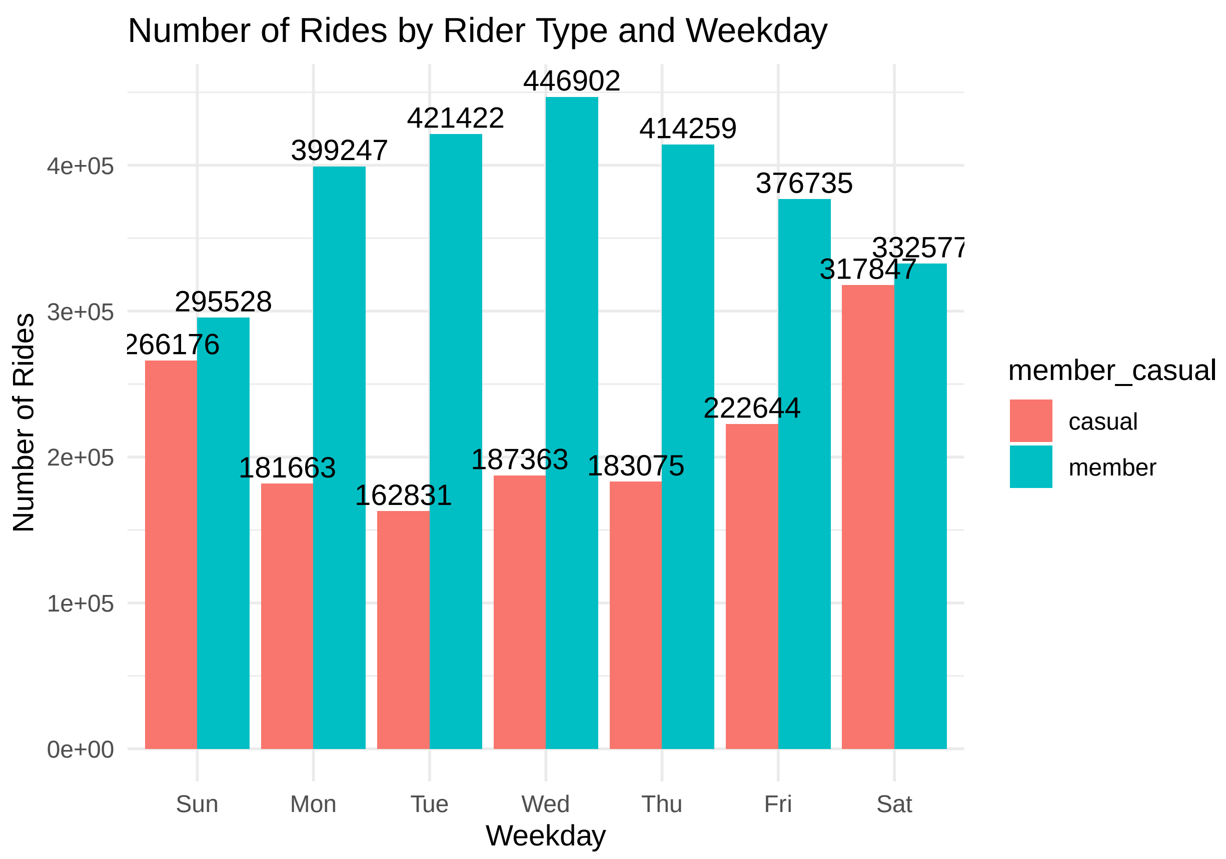
```
## # A tibble: 14 x 4
## # Groups:   member_casual [2]
##    member_casual weekday number_of_rides average_duration
##    <chr>         <ord>             <int>            <dbl>
##  1 casual        Sun              266176            1658.
##  2 casual        Mon              181663            1391.
##  3 casual        Tue              162831            1238.
##  4 casual        Wed              187363            1281.
##  5 casual        Thu              183075            1256.
##  6 casual        Fri              222644            1394.
##  7 casual        Sat              317847            1636.
##  8 member        Sun              295528             841.
##  9 member        Mon              399247             713.
## 10 member        Tue              421422             716.
## 11 member        Wed              446902             730.
## 12 member        Thu              414259             714.
## 13 member        Fri              376735             726.
## 14 member        Sat              332577             841.
```

6. Share

Let's visualize the number of rides by rider type.

```
all_trips_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday) %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge") +
  geom_text(aes(label = number_of_rides), position = position_dodge(0.9), vjust = -0.3) +
  labs(title = "Number of Rides by Rider Type and Weekday", x = "Weekday", y = "Number of Rides") +
  theme_minimal()
```

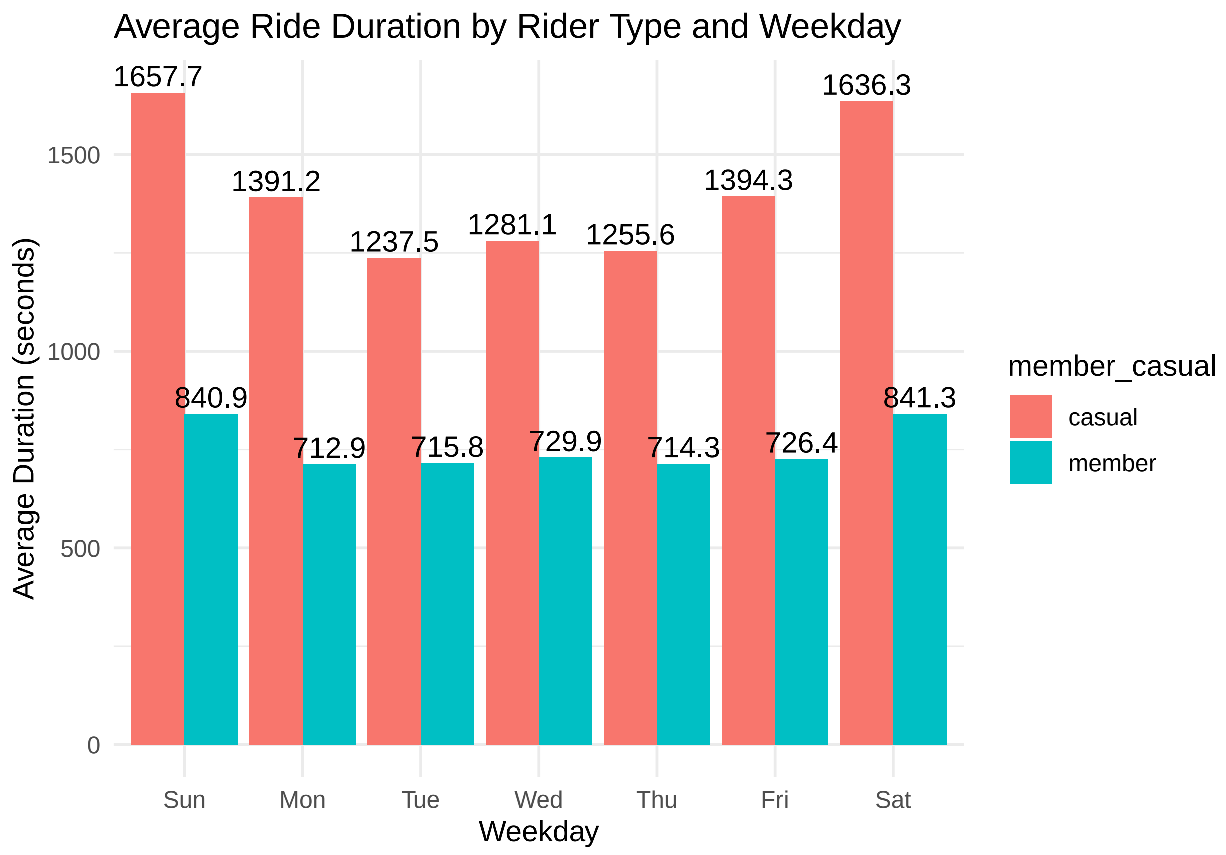
```
## `summarise()` has grouped output by 'member_casual'. You can override using the
## `.groups` argument.
```



Let's create a visualization for average duration.

```
all_trips_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday) %>%
  ggplot(aes(x = weekday, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge") +
  geom_text(aes(label = round(average_duration, 1)), position = position_dodge(0.9), vjust = -0.3) +
  labs(title = "Average Ride Duration by Rider Type and Weekday", x = "Weekday", y = "Average Duration
  theme_minimal()
```

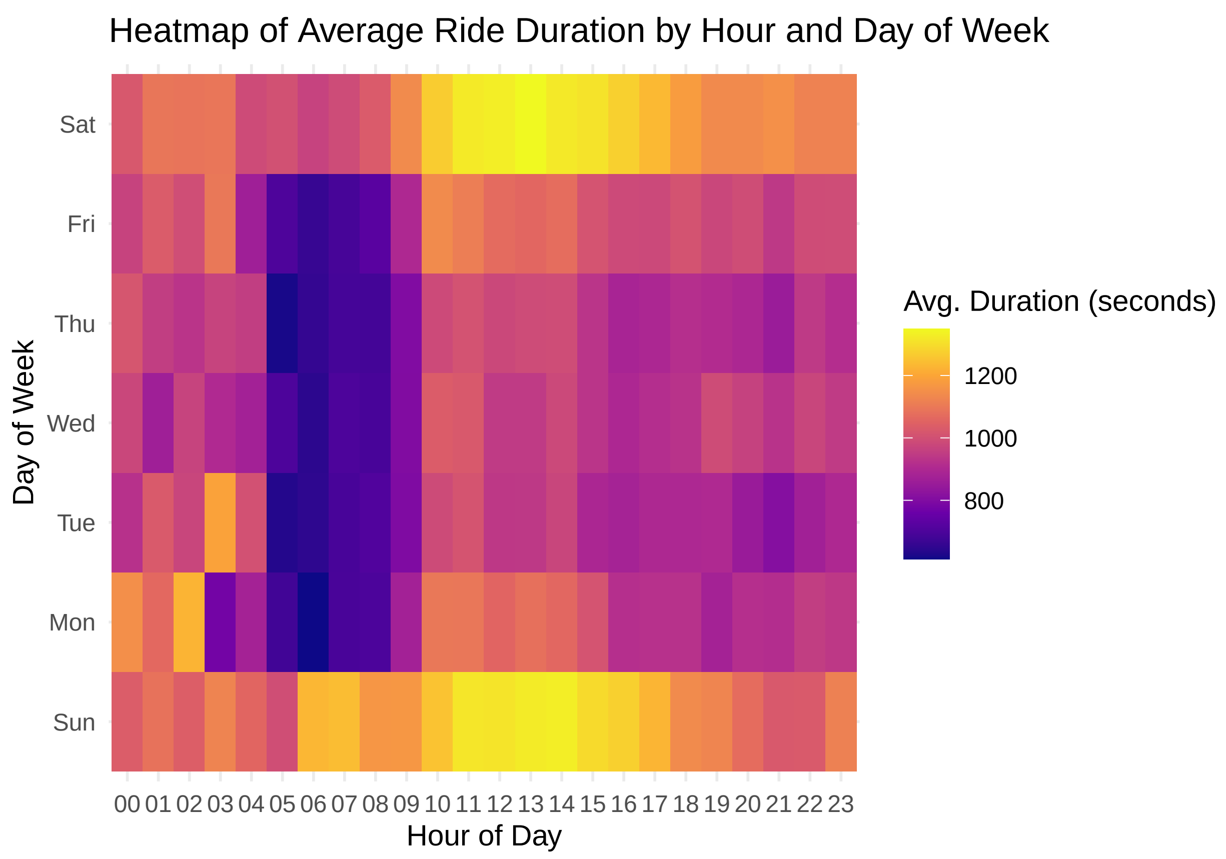
```
## `summarise()` has grouped output by 'member_casual'. You can override using the
## `.groups` argument.
```



Heatmap of average ride length by hour of day and day of the week.

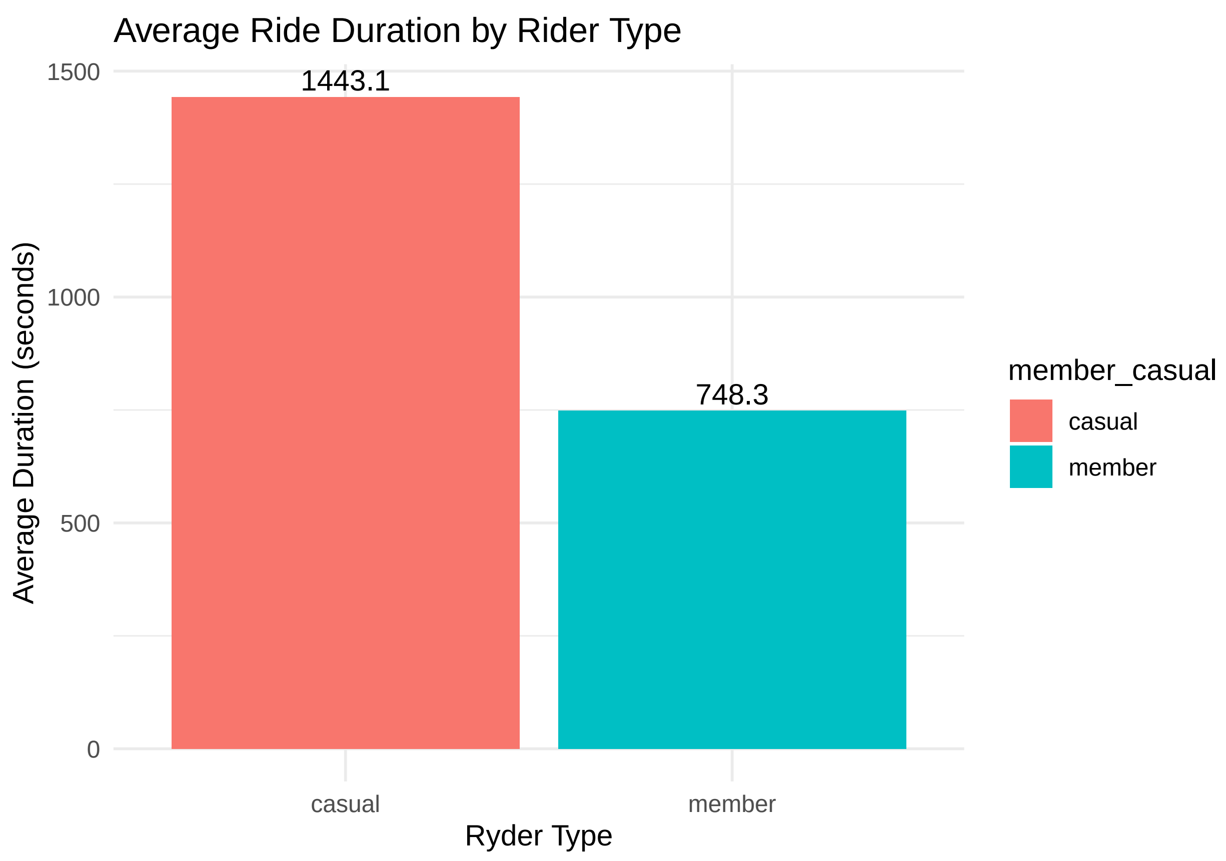
```r
all_trips_v2 %>%
  mutate(hour = format(as.POSIXct(started_at), "%H"),
         weekday = wday(started_at, label = TRUE)) %>%
  group_by(hour, weekday) %>%
  summarise(avg_duration = mean(ride_length)) %>%
  ggplot(aes(x = hour, y = weekday, fill = avg_duration)) +
  geom_tile() +
  scale_fill_viridis_c(option = "C") +
  labs(title = "Heatmap of Average Ride Duration by Hour and Day of Week",
       x = "Hour of Day", y = "Day of Week", fill = "Avg. Duration (seconds)") +
  theme_minimal()
```

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



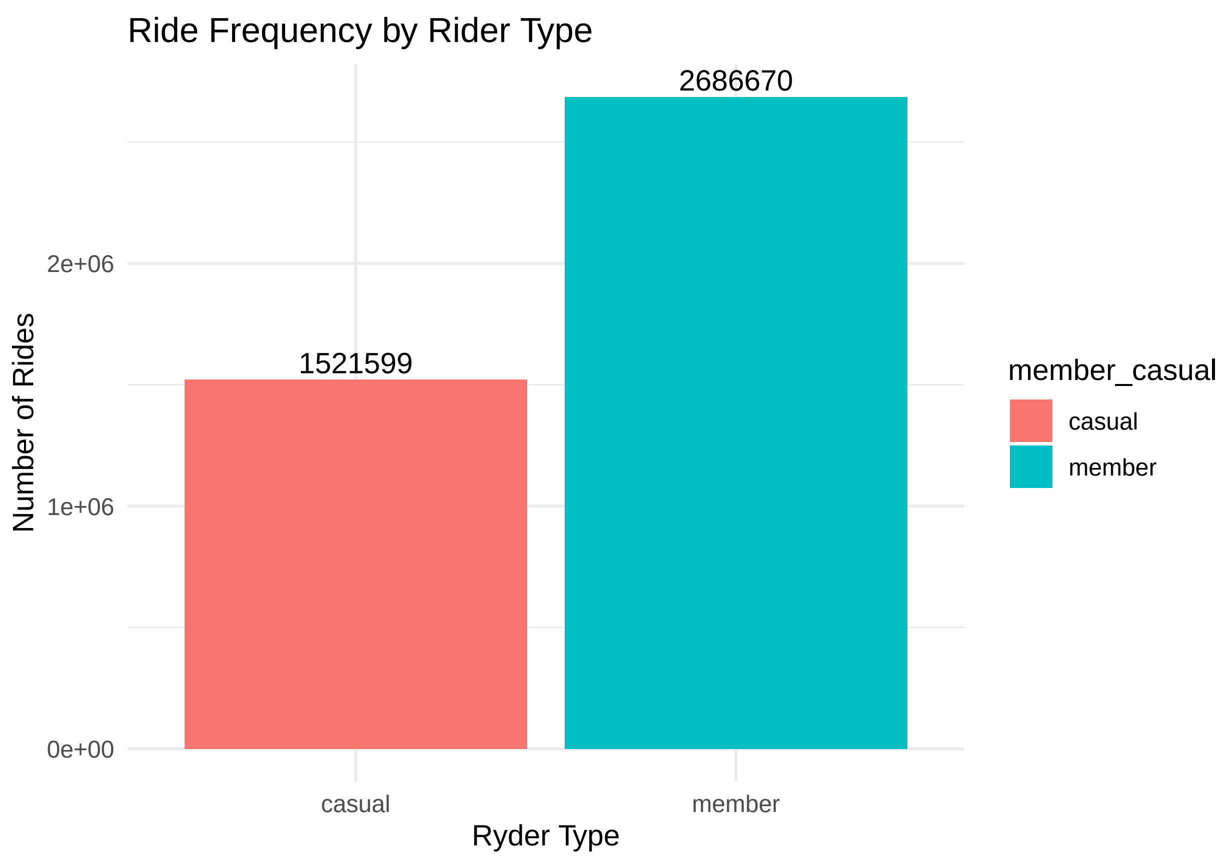
Average Duration by Ryder Type.

```
all_trips_v2 %>%
  group_by(member_casual) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(member_casual) %>%
  ggplot(aes(x=member_casual, y=average_duration, fill = member_casual)) +
  geom_col(position = "dodge") +
  geom_text(aes(label = round(average_duration, 1)), position = position_dodge(0.9), vjust = -0.3) +
  labs(title = "Average Ride Duration by Rider Type", x = "Ryder Type", y = "Average Duration (seconds)
  theme_minimal()
```



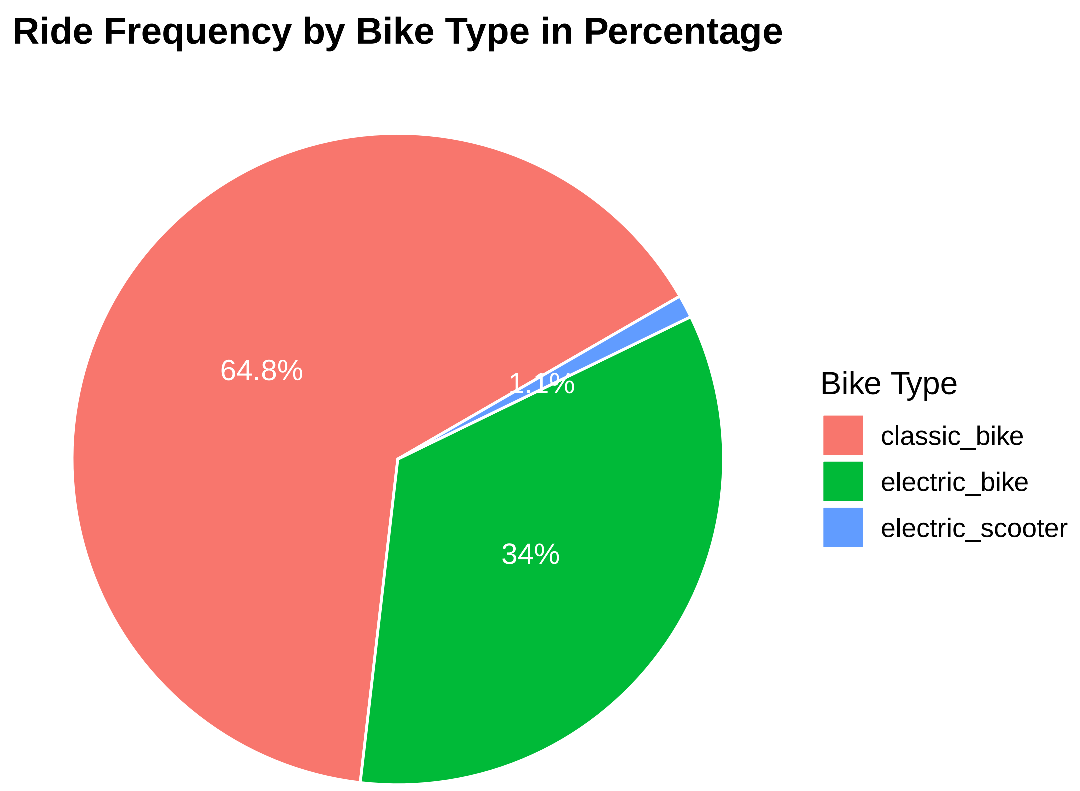
Number of rides by rider type.

```
all_trips_v2 %>%
  group_by(member_casual) %>%
  summarise(number_of_rides = n()
          ,average_duration = mean(ride_length)) %>%
  arrange(member_casual) %>%
  ggplot(aes(x = member_casual, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge") +
  geom_text(aes(label = number_of_rides), position = position_dodge(0.9), vjust = -0.3) +
  labs(title = "Ride Frequency by Rider Type", x = "Ryder Type", y = "Number of Rides") +
  theme_minimal()
```



Ride Frequency by Bike Type in Percentage.

```r
all_trips_v2 %>%
  group_by(rideable_type) %>%
  summarise(number_of_rides = n()) %>%
  arrange(rideable_type) %>%
  ggplot(aes(x = "", y = number_of_rides, fill = rideable_type)) +
  geom_bar(stat = "identity", width = 1, color = "white") +
  coord_polar("y", start = pi/3) +  # Adjust start for similar orientation
  theme_void() +
  geom_text(aes(label = paste0(round((number_of_rides / sum(number_of_rides)) * 100, 1), "%")),
            position = position_stack(vjust = 0.5), size = 4, color = "white") +  # Add percentage labe
  labs(title = "Ride Frequency by Bike Type in Percentage", fill = "Bike Type") +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    legend.position = "right",
    legend.title = element_text(size = 12),
    legend.text = element_text(size = 10)
  )
```



7. Act

Conclusion

The analysis of Divvy bike trip data from January to December 2024 reveals several insights about user behavior and ride patterns:

1. Casual riders tend to have longer ride durations compared to members, indicating that they may use the service for leisure or exploration, while members likely use it for routine commuting.
2. Casual riders are more active on weekends, suggesting recreational use, while members primarily ride during weekdays for commuting.
3. Both groups show distinct usage patterns across days of the week and hours, with casual riders contributing to greater variability.
4. Different bike types (e.g., classic or electric) are preferred by different user groups, aiding in fleet optimization.

Recommendations

1. Focus promotions on weekends for casual riders and offer membership benefits to encourage conversions.
2. Ensure bike availability during peak hours (weekdays for members, weekends for casual riders).
3. Optimize station placements based on ridership patterns.

4. Provide personalized insights (e.g., calories burned, CO2 saved) to encourage casual riders to engage more frequently.
5. Consider loyalty programs or rewards to increase member retention and attract new members.

By implementing these insights, Divvy can optimize operations, enhance customer satisfaction, and drive increased ridership.