Google Data Analytics Capstone

Putu Angga Kurniawan

2024-07-04

Scenario

A Chicago-based bike-sharing company, Cyclistic, wants an analysis of how its riders use their services. The customer base is divided into two categories, annual members and casual riders. Annual members are defined as those who have a yearly subscription, whereas casual riders are defined as though who use single session and day passes. Financial analysts have found that annual members are the most lucrative customer base, and so the director of marketing wants data-backed analysis that will help guide their marketing campaign to convert casual riders into annual members.

Ask

I will be using dataset from the year 2022. All data is publicly available data by the Motivate International Inc., under this license through the Google Data Analytics Certificate. The attributes within the data contain such information as rider identification number, bike type, starting and ending latitude and longitude, rider status (casual or member), and start and end station identification.

Does it pass the ROCCC test? (Reliable, Original, Comprehensive, Current, and Cited)

Reliable: Yes, the dataset comes the City of Chicago's Divvy program, which is managed by the company Lyft Bikes and Scooters, LLC, which is part of a publicly traded company.

Original: Yes, all of the data is original and collected by first-party

Comprehensive: Overall, it is comprehensive and sufficient to answer the business question at hand.

Current: No, this data is from 2022, so it is relevant but not current.

Cited: Yes, it comes from a first-party source collected from the Divvy program, who runs the bikeshare program in Chicago. The sources are credible.

Prepare

Load necessary libraries

```
library(tidyverse)
```

```
## Warning: package 'lubridate' was built under R version 4.4.1
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
```

```
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
```

```
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_confli
cts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force a
ll conflicts to become errors
```

```
library(readr)
library(lubridate)
library(ggplot2)
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.4.1
## Linking to GEOS 3.12.1, GDAL 3.8.4, PROJ 9.3.1; sf_use_s2() is TRUE
```

```
library(scales)
```

```
## Warning: package 'scales' was built under R version 4.4.1
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

Load Data

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

```
## Rows: 103770 Columns: 13
## ── Column specification ────────────────────────────────────────────────
────────

## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
```

```
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

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

```
## Rows: 115609 Columns: 13
## ── Column specification ────────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

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

```
## Rows: 284042 Columns: 13
## ── Column specification ────────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

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

```
## Rows: 371249 Columns: 13
## ── Column specification ────────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
```

```
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

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

```
## Rows: 634858 Columns: 13
## ── Column specification ────────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

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

```
## Rows: 769204 Columns: 13
## ── Column specification ────────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

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

```
## Rows: 823488 Columns: 13
## ── Column specification ────────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
```

```
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

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

```
## Rows: 785932 Columns: 13
## ── Column specification ────────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

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

```
## Rows: 701339 Columns: 13
## ── Column specification ────────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

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

```
## Rows: 558685 Columns: 13
## ── Column specification ────────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
```

```
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

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

```
## Rows: 337735 Columns: 13
## ── Column specification ────────────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

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

```
## Rows: 181806 Columns: 13
## ── Column specification ────────────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

Combine tables into one dataset

```
total_data_set <- rbind(jan_tripdata, feb_tripdata, feb_tripdata, mar_tripd
ata, apr_tripdata, may_tripdata, jun_tripdata, jul_tripdata, aug_tripdata,
sep_tripdata, oct_tripdata, nov_tripdata, dec_tripdata)
```

Inspect the columns to determine what functions will be necessary to perform functions.

```
str(total_data_set)
```

```
## spc_tbl_ [5,783,326 × 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:5783326] "C2F7DD78E82EC875" "A6CF8980A652D
272" "BD0F91DFF741C66D" "CBB80ED419105406" ...
##  $ rideable_type     : chr [1:5783326] "electric_bike" "electric_bike" "
classic_bike" "classic_bike" ...
```

```
##  $ started_at        : POSIXct[1:5783326], format: "2022-01-13 11:59:47"
"2022-01-10 08:41:56" ...

##  $ ended_at          : POSIXct[1:5783326], format: "2022-01-13 12:02:44"
"2022-01-10 08:46:17" ...

##  $ start_station_name: chr [1:5783326] "Glenwood Ave & Touhy Ave" "Glenw
ood Ave & Touhy Ave" "Sheffield Ave & Fullerton Ave" "Clark St & Bryn Mawr
Ave" ...

##  $ start_station_id  : chr [1:5783326] "525" "525" "TA1306000016" "KA150
4000151" ...

##  $ end_station_name  : chr [1:5783326] "Clark St & Touhy Ave" "Clark St
& Touhy Ave" "Greenview Ave & Fullerton Ave" "Paulina St & Montrose Ave" ..
.

##  $ end_station_id    : chr [1:5783326] "RP-007" "RP-007" "TA1307000001"
"TA1309000021" ...

##  $ start_lat         : num [1:5783326] 42 42 41.9 42 41.9 ...

##  $ start_lng         : num [1:5783326] -87.7 -87.7 -87.7 -87.7 -87.6 ...

##  $ end_lat           : num [1:5783326] 42 42 41.9 42 41.9 ...

##  $ end_lng           : num [1:5783326] -87.7 -87.7 -87.7 -87.7 -87.6 ...

##  $ member_casual     : chr [1:5783326] "casual" "casual" "member" "casua
l" ...

##  - 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>
```

Process

Clean the dataset of any null values.

```
clean_dataset <- na.omit(total_data_set) %>%
  distinct()
```

After cleaning out any rows with null values, I decided to scrub out any rows containing “docked_bike.” The prompt did not provide a definition, and since I do not have a manager to ask, I decided to classify them as dirty data.

```
rideable_type <- clean_dataset$rideable_type
total_dataset <- subset(clean_dataset, rideable_type != "docked_bike")
```

Next, we must add the columns “ride_length,” “month,” and “day_of_week” by creating a new table. This will allow us to make the calculations we need.

```
total_dataset <- total_dataset %>%
  mutate(ride_length = difftime(ended_at, started_at, units = "mins")) %>%
  mutate(day_of_week = wday(total_dataset$started_at)) %>%
  mutate(month = format(as.Date(total_dataset$started_at, format="%d/%m/%Y"
),"%m"))
```

Analyze

Next, we need to create the variable for “ride_length” column and then perform some calculations. This will reveal the average ride length for each month of the year.

```
ride_length <- total_dataset$ride_length


total_dataset %>%
  group_by(month, member_casual) %>%
  summarize(mean_ridelength = mean(ride_length)) %>%
  arrange(member_casual) %>%
  print(n=24)
```

```
## `summarise()` has grouped output by 'month'. You can override using the
## `.groups` argument.
## # A tibble: 24 × 3
## # Groups:   month [12]
##    month member_casual mean_ridelength
##    <chr> <chr>         <drtn>
##  1 01    casual        16.82959 mins
##  2 02    casual        19.08613 mins
##  3 03    casual        23.60740 mins
##  4 04    casual        22.57597 mins
```

```
##  5 05    casual        24.43602 mins
##  6 06    casual        22.25615 mins
##  7 07    casual        21.91729 mins
##  8 08    casual        20.58434 mins
##  9 09    casual        19.54048 mins
## 10 10    casual        18.22758 mins
## 11 11    casual        15.31033 mins
## 12 12    casual        13.36683 mins
## 13 01    member        10.26992 mins
## 14 02    member        10.64466 mins
## 15 03    member        11.79643 mins
## 16 04    member        11.60937 mins
## 17 05    member        13.30083 mins
## 18 06    member        13.68314 mins
## 19 07    member        13.50296 mins
## 20 08    member        13.10579 mins
## 21 09    member        12.62185 mins
## 22 10    member        11.67420 mins
## 23 11    member        10.82394 mins
## 24 12    member        10.20045 mins
```

We will create some additional variables from the columns for future calculations. All necessary variables are below.

```
ride_length <- (total_dataset$ride_length)
member_casual <- total_dataset$member_casual
day_of_week <- total_dataset$day_of_week
rideable_type <- total_dataset$rideable_type
```

Then we calculate average ride length for each day of the week.

```
total_dataset %>%
  group_by(day_of_week, member_casual) %>%
  summarize(mean_ridelength = mean(ride_length)) %>%
  arrange(member_casual)
```

```
## `summarise()` has grouped output by 'day_of_week'. You can override usin
g the
## `.groups` argument.
## # A tibble: 14 × 3
## # Groups:   day_of_week [7]
```

```
##    day_of_week member_casual mean_ridelength
##          <dbl> <chr>         <drtn>
##  1           1 casual        23.89903 mins
##  2           2 casual        21.27527 mins
##  3           3 casual        18.88230 mins
##  4           4 casual        18.23358 mins
##  5           5 casual        18.80030 mins
##  6           6 casual        19.90313 mins
##  7           7 casual        23.59326 mins
##  8           1 member        13.84865 mins
##  9           2 member        12.03277 mins
## 10           3 member        11.79062 mins
## 11           4 member        11.84688 mins
## 12           5 member        12.03136 mins
## 13           6 member        12.22691 mins
## 14           7 member        13.98130 mins
```

Next, we will calculate the total rides per month.

```
total_dataset %>%
  group_by(month, member_casual) %>%
  summarize(total_rides = n()) %>%
  arrange(member_casual) %>%
  print(n=24)
```

```
## `summarise()` has grouped output by 'month'. You can override using the
## `.groups` argument.
## # A tibble: 24 × 3
## # Groups:   month [12]
##    month member_casual total_rides
##    <chr> <chr>               <int>
##  1 01    casual              11662
##  2 02    casual              13800
##  3 03    casual              58934
##  4 04    casual              79917
##  5 05    casual             194126
##  6 06    casual             261856
##  7 07    casual             281079
##  8 08    casual             244207
```

```
##  9 09    casual             201427
## 10 10    casual             138928
## 11 11    casual              67762
## 12 12    casual              29633
## 13 01    member              67523
## 14 02    member              74034
## 15 03    member             148827
## 16 04    member             180663
## 17 05    member             282299
## 18 06    member             328282
## 19 07    member             331002
## 20 08    member             335230
## 21 09    member             314230
## 22 10    member             262945
## 23 11    member             182238
## 24 12    member             103898
```

As well as per day.

```
total_dataset %>%
  group_by(day_of_week, member_casual) %>%
  summarize(total_rides = n()) %>%
  arrange(member_casual)
```

```
## `summarise()` has grouped output by 'day_of_week'. You can override usin
g the
## `.groups` argument.
## # A tibble: 14 × 3
## # Groups:   day_of_week [7]
##    day_of_week member_casual total_rides
##          <dbl> <chr>               <int>
##  1           1 casual             266124
##  2           2 casual             188562
##  3           3 casual             178879
##  4           4 casual             186479
##  5           5 casual             210521
##  6           6 casual             225778
##  7           7 casual             326988
##  8           1 member             297733
##  9           2 member             375171
```

```
## 10           3 member            411249
## 11           4 member            412795
## 12           5 member            415890
## 13           6 member            360054
## 14           7 member            338279
```

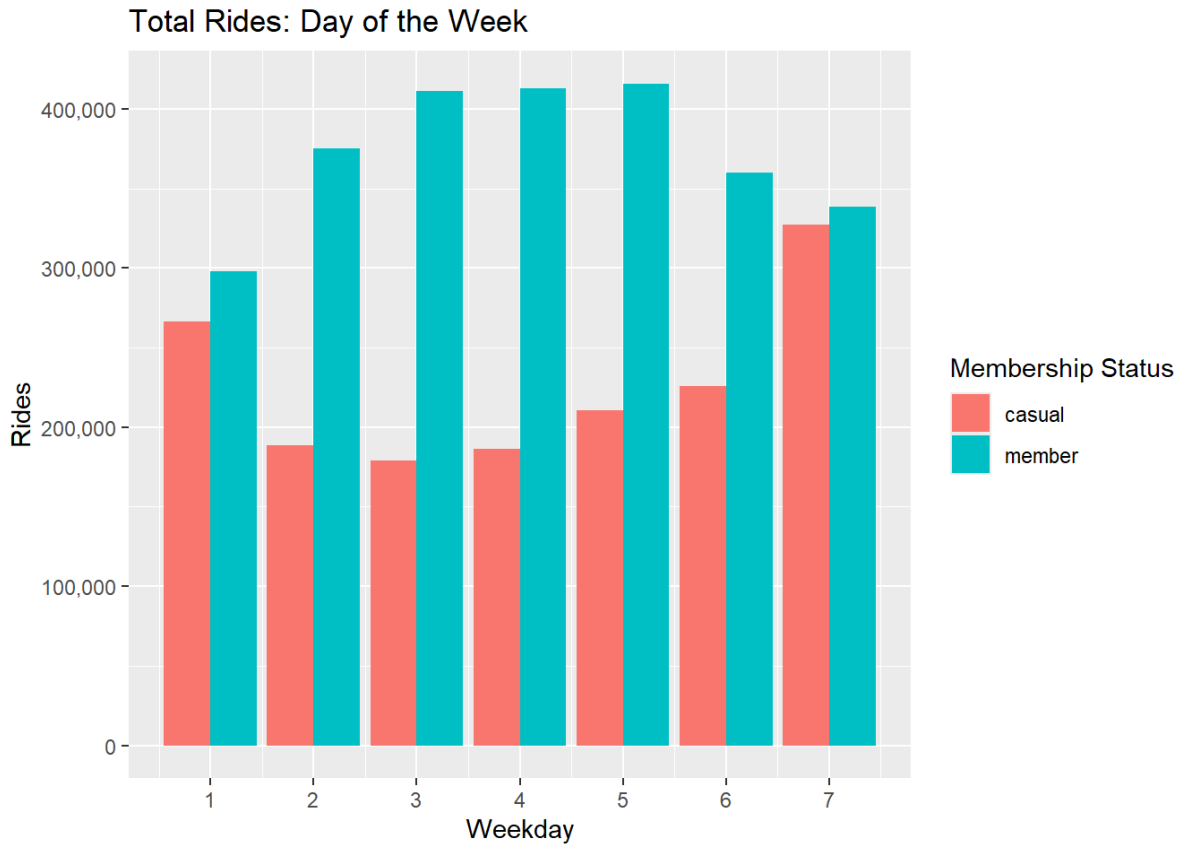
Share

While one can eyeball these numbers to find conclusions, its more helpful to visualize the results so that stakeholders can more quickly understand the results. The first visualization is total rides per each month and then total rides per weekday, filtered by membership status. The scale for weekday is 1 = Sunday to 7 = Saturday.

```
ggplot(total_dataset, aes(x=month, fill = member_casual)) +
  geom_bar(position = "dodge") + scale_y_continuous(labels=comma) +
  ggtitle("Total Rides: Month") +
  xlab("Month") + ylab("Rides") + labs(fill = "Membership Status")
```


```
ggplot(total_dataset, aes(x=day_of_week, fill = member_casual)) +
  geom_bar(position = "dodge") + scale_y_continuous(labels=comma) +
  scale_x_continuous(breaks=seq(1,7,1)) +
```

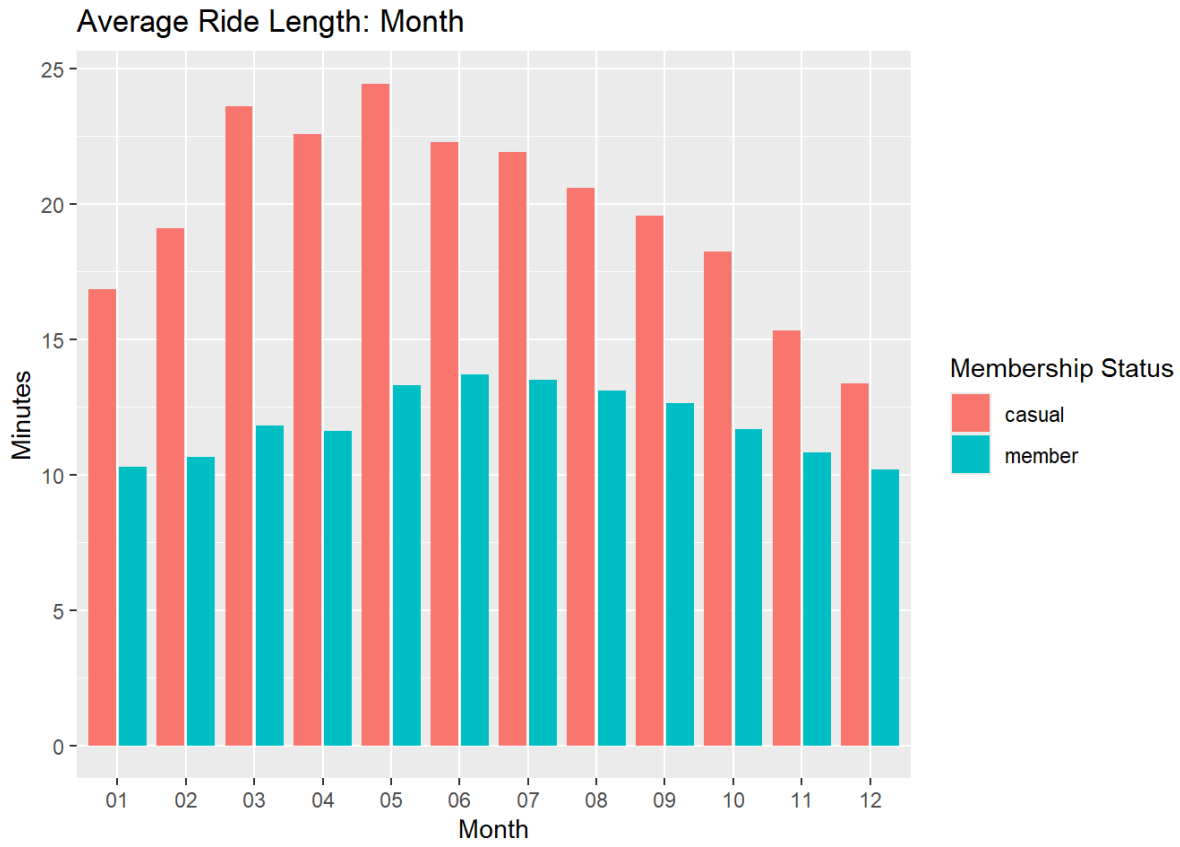
```
  ggtitle("Total Rides: Day of the Week") +
  xlab("Weekday") + ylab("Rides") + labs(fill = "Membership Status")
```



Then we will look at the average ride length per month and per weekday.

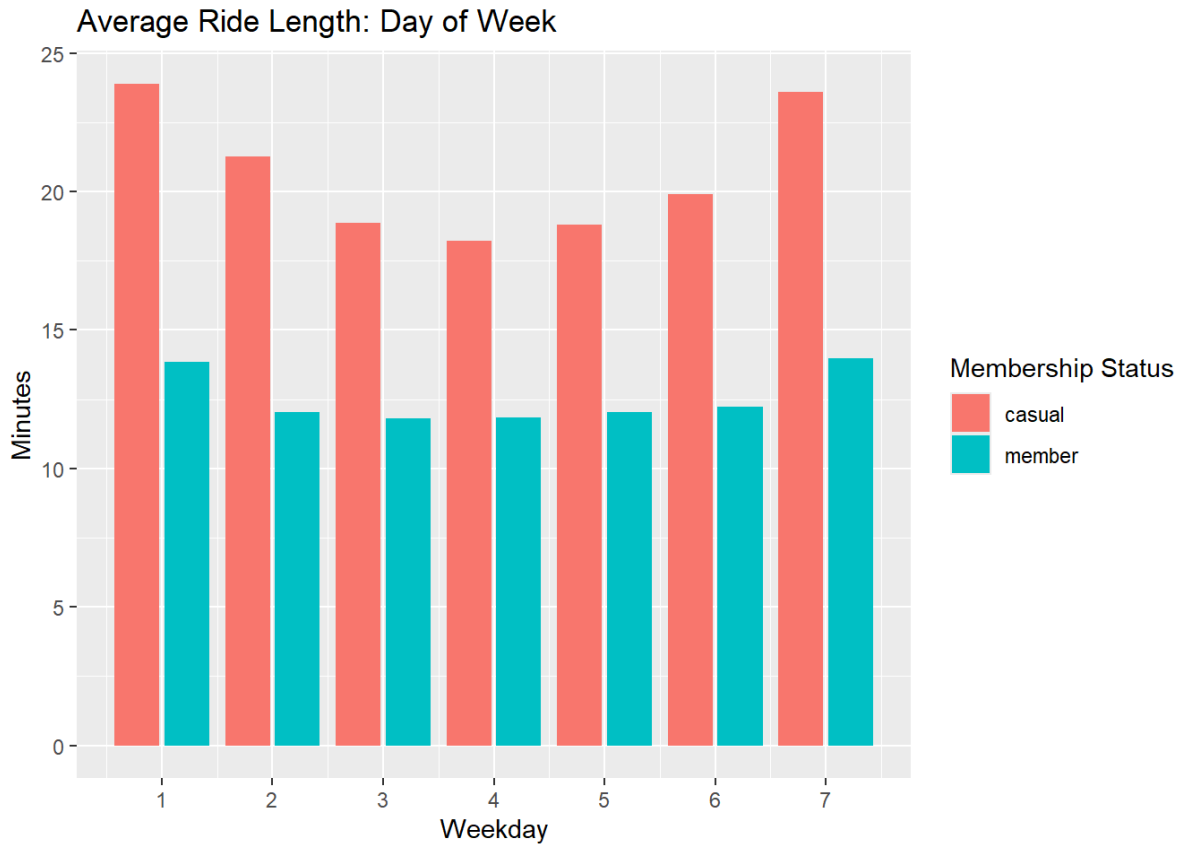
```
total_dataset %>%
  group_by(month, member_casual) %>%
  summarize(mean_ridelength = mean(ride_length)) %>%
  ggplot(aes(x = month, y = mean_ridelength, fill = member_casual)) +
  geom_col(position = "dodge2")  + ggtitle("Average Ride Length: Month") +
  xlab("Month") + ylab("Minutes") + labs(fill = "Membership Status")
```

```
## `summarise()` has grouped output by 'month'. You can override using the
## `.groups` argument.
## Don't know how to automatically pick scale for object of type <difftime>
.
## Defaulting to continuous.
```



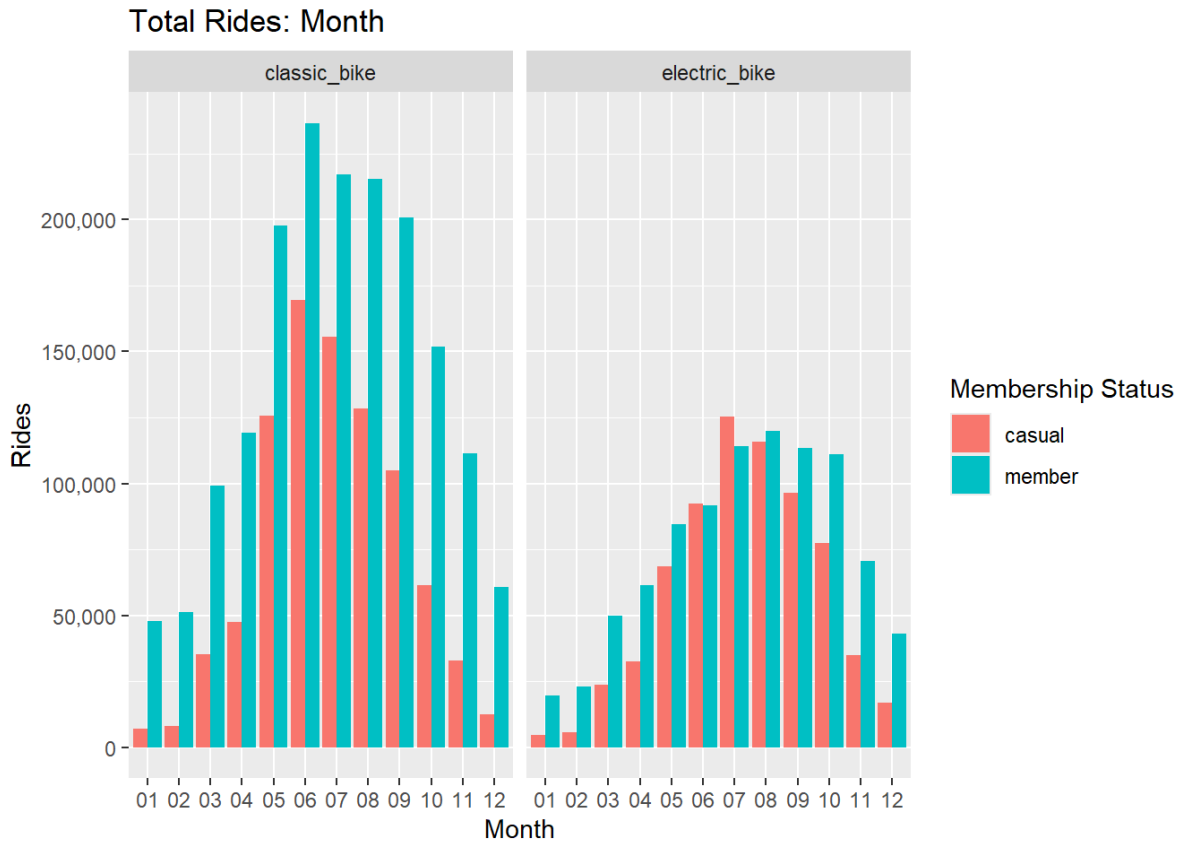

```
total_dataset %>%
  group_by(day_of_week, member_casual) %>%
  summarize(mean_ridelength = mean(ride_length)) %>%
  ggplot(aes(x = day_of_week, y = mean_ridelength, fill = member_casual)) +
  scale_x_continuous(breaks=seq(1,7,1)) + geom_col(position = "dodge2")  +
  ggtitle("Average Ride Length: Day of Week") + xlab("Weekday") +
  ylab("Minutes") + labs(fill = "Membership Status")
```

```
## `summarise()` has grouped output by 'day_of_week'. You can override usin
g the
## `.groups` argument.
## Don't know how to automatically pick scale for object of type <difftime>
.
## Defaulting to continuous.
```

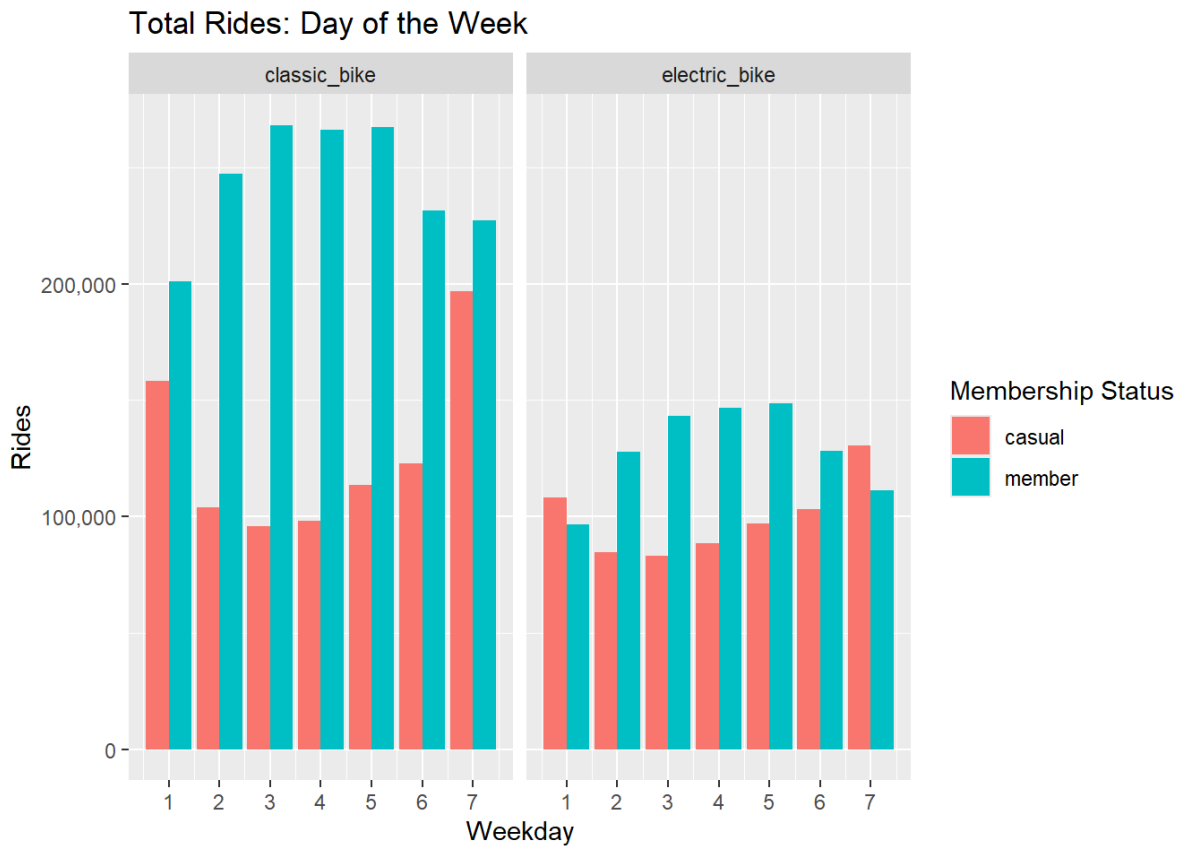


Recognizing that electric bike and classic bike riders may have different riding patterns, I plotted the usage separately for each bike type for both member and casual riders.

```
ggplot(total_dataset, aes(x=month, fill = member_casual)) +
  geom_bar(position = "dodge") + scale_y_continuous(labels=comma) +
  ggtitle("Total Rides: Month") +
  xlab("Month") + ylab("Rides") + labs(fill = "Membership Status") +
  facet_wrap(vars(rideable_type))
```



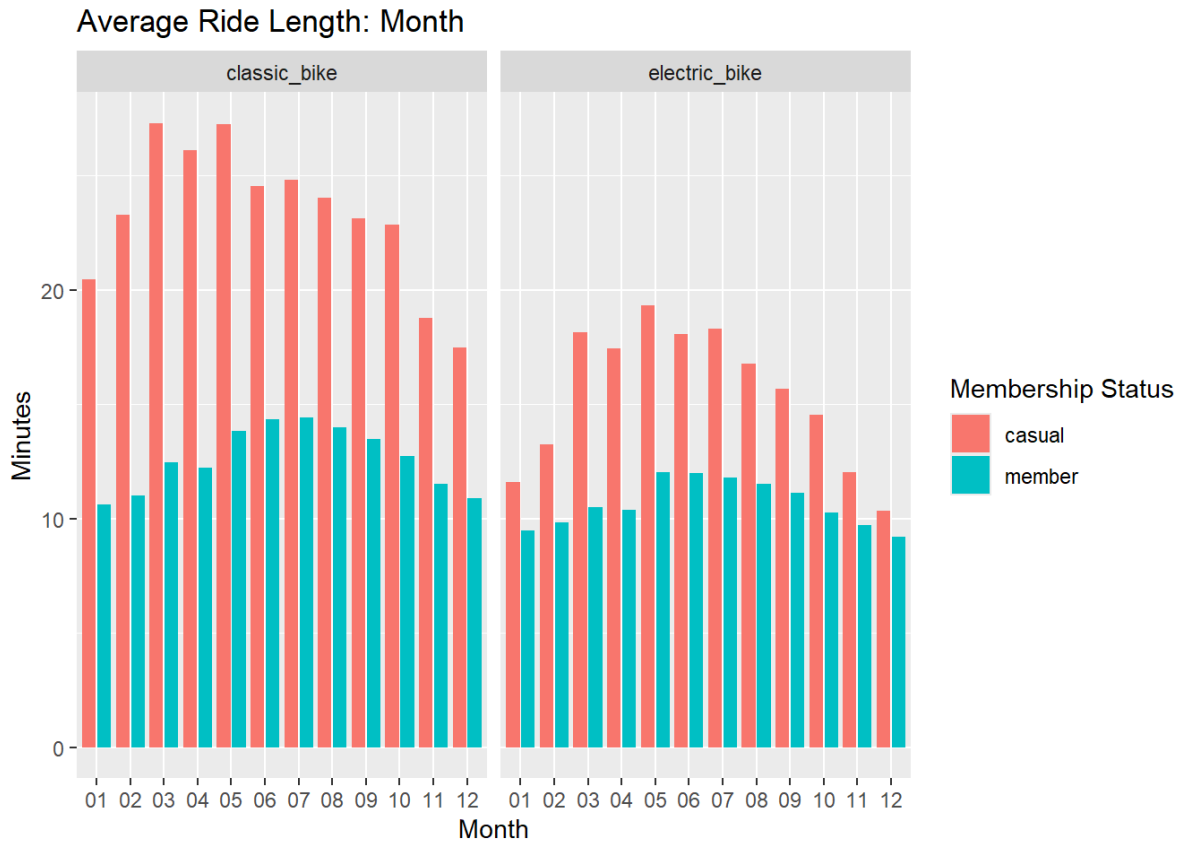

```
ggplot(total_dataset, aes(x=day_of_week, fill = member_casual)) +
  geom_bar(position = "dodge") + scale_y_continuous(labels=comma) +
  ggtitle("Total Rides: Day of the Week") +
  scale_x_continuous(breaks=seq(1,7,1)) +
  xlab("Weekday") + ylab("Rides") + labs(fill = "Membership Status") +
  facet_wrap(vars(rideable_type))
```



Average ride length

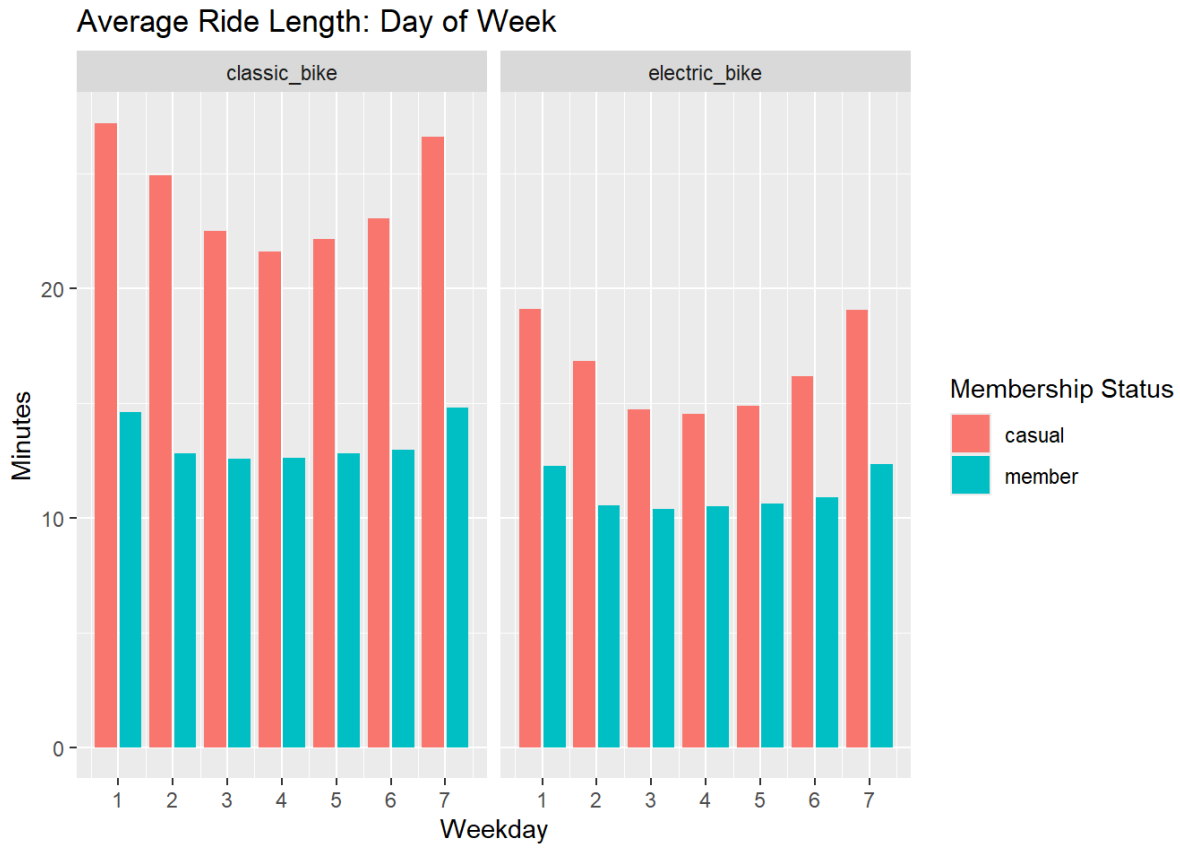
```
total_dataset %>%
  group_by(month, member_casual, rideable_type) %>%
  summarize(mean_ridelength = mean(ride_length)) %>%
  ggplot(aes(x = month, y = mean_ridelength, fill = member_casual)) +
  geom_col(position = "dodge2")  + ggtitle("Average Ride Length: Month") +
  xlab("Month") + ylab("Minutes") + labs(fill = "Membership Status") +
  facet_wrap(vars(rideable_type))
```

```
## `summarise()` has grouped output by 'month', 'member_casual'. You can ov
erride
## using the `.groups` argument.
## Don't know how to automatically pick scale for object of type <difftime>
.
## Defaulting to continuous.
```




```
total_dataset %>%
  group_by(day_of_week, member_casual, rideable_type) %>%
  summarize(mean_ridelength = mean(ride_length)) %>%
  ggplot(aes(x = day_of_week, y = mean_ridelength, fill = member_casual)) +
  scale_x_continuous(breaks=seq(1,7,1)) + geom_col(position = "dodge2")  +
  ggtitle("Average Ride Length: Day of Week") + xlab("Weekday") + ylab("Min
utes") +
  labs(fill = "Membership Status") + facet_wrap(vars(rideable_type))
```

```
## `summarise()` has grouped output by 'day_of_week', 'member_casual'. You
can
## override using the `.groups` argument.
## Don't know how to automatically pick scale for object of type <difftime>
.
## Defaulting to continuous.
```



The data reveals some interesting trends between the casual users and the member users:

- Casual riders have long average ride lengths, whereas member riders take more frequent rides.
- Member riders take more rides during the weekday, whereas casual riders take more rides on the weekend.
- Average ride length increases on weekends for both casual and member riders.
- Between both rider groups, there is a greater disparity in average ride length among classic bike riders than electric bike riders.

These trends all suggest that casual riders and member riders use the program for different purposes, and therefore, it is not precisely clear how easy it is to convert casual riders to member riders.

Act

Cyclistic needs more information about the member riders in order to determine the best strategy for attracting more members. A survey of member riders would be a good exploratory step to determine why they joined the program. Some questions I would use in the survey would include:

- Why do you use the program?
- Why did you join the membership program?
- How frequently do you use the program?
- How did you learn about the program?

After collecting more data, the company can create a more effective marketing campaign.