Maximizing Annual Memberships in Cyclistic Bike-Share Program: A Comparative Analysis of Casual Riders and Annual Members

Joe Chapa

2023-03-30

Introduction

The goal of this case study is to offer a thorough analysis of Cyclistic, a fictional bike-share provider in Chicago. To achieve this goal, we will use the six stages of the data analysis process — "ask, prepare, process, analyze, share, and act".

Ask

The Cyclistic marketing analytics team aims to identify differences in usage patterns between annual members and casual riders. These insights will allow us to recommend marketing strategies to convert causal users to annual members. These recommendations will then be presented to marketing director and executive team for review. This project is crucial to Cyclistic's growth and competitiveness in the bike-share market.

Prepare

We obtained the past 12 months (April 2022 - March 2023) of Cyclistic trip data from our AWS S3 bucket"divvy-tripdata". As our data is in-house, it a reliable. Our data consists of 12 CSV files and was downloaded to the directory "~/Google Capstone Project/csv". It was imported into excel and checked for completeness and accuracy. As a reminder, our data is subject to this licensed agreement with Lyft Bikes and Scooters, LLC, ensuring that licensing, privacy, security, and accessibility concerns were addressed.

One issue this data posed was its size. Based on our analysis, we recommend using a machine with at least 16 GB of RAM to efficiently replicate our results. A detailed data dictionary is provided below.

Data Dictionary

- **ride_id**: Unique id assigned to a single ride
- **rideable_type**: Type of bike used for the ride
  - classic_bike
  - docked_bike
  - electric_bike
- **started_at**: Date and time the ride was started
- **ended_at**: Date and time the ride was terminated
- **start_station_name**: Name of the station where the ride started
- **start_station_id**:Unique id of the station where the ride started
- **end_station_name**: Name of the station where the ride ended
- **end_station_id**: Unique id of the station where the ride ended
- **start_lat**: Latitude value of where the starting station is located
- **start_lng**: Longitude value of where the starting station is located
- **end_lat**: Latitude value of where the ending station is located

- **end_lng**: Longitude value of where the ending station is located
- **member_casual**:
  - casual: Customer of this ride purchased a 'single-ride' or 'full-day' pass
  - member: Customer of this ride has purchased an annual membership

File descriptions

```
##  [1] "202203-divvy-tripdata.csv"       "202204-divvy-tripdata.csv"
##  [3] "202205-divvy-tripdata.csv"       "202206-divvy-tripdata.csv"
##  [5] "202207-divvy-tripdata.csv"       "202208-divvy-tripdata.csv"
##  [7] "202209-divvy-publictripdata.csv" "202210-divvy-tripdata.csv"
##  [9] "202211-divvy-tripdata.csv"       "202212-divvy-tripdata.csv"
## [11] "202301-divvy-tripdata.csv"       "202302-divvy-tripdata.csv"
```

Importing the Data

After downloading the data, we observed that one of the files had a different naming scheme from the others. However, all files started with the pattern 'YYYYMM-divvy-…'. To streamline our code and avoid repetition, we decided to use a for-loop to import and rename the data, utilizing the naming pattern

Load CSVs

```r
months <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
for (file_name in file_list) {
  # extract month and year from file_name
  month <- substr(file_name, 5, 6)
  year <- substr(file_name, 3, 4)

  # create variables
  var_name <- paste(months[as.numeric(month)], year, "_tripdata", sep = "")
  assign(var_name, read.csv(paste(path, file_name, sep = "")))

  #OPTIONAL: print out a statement to inform the user the file has been imported
  print(paste(var_name, "has been imported"))
}
```

```
## [1] "Mar22_tripdata has been imported"
## [1] "Apr22_tripdata has been imported"
## [1] "May22_tripdata has been imported"
## [1] "Jun22_tripdata has been imported"
## [1] "Jul22_tripdata has been imported"
## [1] "Aug22_tripdata has been imported"
## [1] "Sep22_tripdata has been imported"
## [1] "Oct22_tripdata has been imported"
## [1] "Nov22_tripdata has been imported"
## [1] "Dec22_tripdata has been imported"
## [1] "Jan23_tripdata has been imported"
## [1] "Feb23_tripdata has been imported"
```

Verify Data Integrity

Before merging our 12 CSV files into a single dataset, we ensured all files shared consistent column names and data types. To maintain the readability of this report, we have included output for 2 out of 12 CSVs. Nevertheless, it is important to note that this test was applied to all 12 CSV files.

Verify matching data types

```r
#list all objects in the environment
obj_list <- ls()

#filter the data frames from the object list
df_list <- obj_list[sapply(obj_list, function(x) is.data.frame(get(x)))]
df_list
```

```
##  [1] "Apr22_tripdata" "Aug22_tripdata" "Dec22_tripdata" "Feb23_tripdata"
##  [5] "Jan23_tripdata" "Jul22_tripdata" "Jun22_tripdata" "Mar22_tripdata"
##  [9] "May22_tripdata" "Nov22_tripdata" "Oct22_tripdata" "Sep22_tripdata"
```

```r
for (df_name in df_list[2]){
  cat(paste0("Information for ", df_name, ":\n"))
  glimpse(df_name)
}
```

```
## Information for Aug22_tripdata:
##  chr "Aug22_tripdata"
```

Combine data

```r
#Combine data
all_trips <- bind_rows(mget(df_list))
glimpse(all_trips)
```

```
## Rows: 5,829,084
## Columns: 13
## $ ride_id            <chr> "3564070EEFD12711", "0B820C7FCF22F489", "89EEEE3229~
## $ rideable_type      <chr> "electric_bike", "classic_bike", "classic_bike", "c~
## $ started_at         <chr> "2022-04-06 17:42:48", "2022-04-24 19:23:07", "2022~
## $ ended_at           <chr> "2022-04-06 17:54:36", "2022-04-24 19:43:17", "2022~
## $ start_station_name <chr> "Paulina St & Howard St", "Wentworth Ave & Cermak R~
## $ start_station_id   <chr> "515", "13075", "TA1307000121", "13075", "TA1307000~
## $ end_station_name   <chr> "University Library (NU)", "Green St & Madison St",~
## $ end_station_id     <chr> "605", "TA1307000120", "TA1307000120", "KA170600500~
## $ start_lat          <dbl> 42.01913, 41.85308, 41.87184, 41.85308, 41.87181, 4~
## $ start_lng          <dbl> -87.67353, -87.63193, -87.64664, -87.63193, -87.646~
## $ end_lat            <dbl> 42.05294, 41.88189, 41.88189, 41.86749, 41.88224, 4~
## $ end_lng            <dbl> -87.67345, -87.64879, -87.64879, -87.63219, -87.641~
## $ member_casual      <chr> "member", "member", "member", "casual", "member", "~
```

```r
rm(list = df_list)
```

Our dataset contains over almost 6 million entries and 13 features to work with. First, we verified our data was distinct.

Check for duplicate entries

```r
duplicated_entries <- sum(duplicated(all_trips))
duplicated_entries
```

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

We confirmed no duplicates were present. Next we checked each column for missing values.

Check for missing values

```r
na_values <- colSums(is.na(all_trips))
na_values
```

```
##           ride_id      rideable_type         started_at           ended_at
##                 0                  0                  0                  0
## start_station_name   start_station_id   end_station_name     end_station_id
##                 0                  0                  0                  0
##         start_lat          start_lng            end_lat            end_lng
##                 0                  0               5938               5938
##      member_casual
##                 0
```

We identified that the `end_lat` and `end_lng` columns contained missing values. A note has been made and will be address in the *Process* phase. We then proceeded to check if our dataset contained empty strings.

Check for empty strings

```
empty_values <- all_trips %>%
  summarise_all(~sum(. %in% c("", " ")))
empty_values
```

```
##   ride_id rideable_type started_at ended_at start_station_name start_station_id
## 1       0             0          0        0             850418           850550
##   end_station_name end_station_id start_lat start_lng end_lat end_lng
## 1           909038         909179         0         0       0       0
##   member_casual
## 1             0
```

We discovered empty strings in our `start_station_name`, `start_station_id`, `end_station_name`, and `end_station_id` columns. Note `end_lat` and `end_lng` appear as `NA` due to containing numeric values. In addition to checking for missing and empty values, its important we esnsured our data types are appropriate for our analysis.

Confirm valid data-typing

```
str(all_trips)
```

```
## 'data.frame':    5829084 obs. of  13 variables:
##  $ ride_id           : chr  "3564070EEFD12711" "0B820C7FCF22F489" "89EEEE32293F07FF" "84D4751AEB3188
##  $ rideable_type     : chr  "electric_bike" "classic_bike" "classic_bike" "classic_bike" ...
##  $ started_at        : chr  "2022-04-06 17:42:48" "2022-04-24 19:23:07" "2022-04-20 19:29:08" "2022-0
##  $ ended_at          : chr  "2022-04-06 17:54:36" "2022-04-24 19:43:17" "2022-04-20 19:35:16" "2022-0
##  $ start_station_name: chr  "Paulina St & Howard St" "Wentworth Ave & Cermak Rd" "Halsted St & Polk S
##  $ start_station_id  : chr  "515" "13075" "TA1307000121" "13075" ...
##  $ end_station_name  : chr  "University Library (NU)" "Green St & Madison St" "Green St & Madison St
##  $ end_station_id    : chr  "605" "TA1307000120" "TA1307000120" "KA1706005007" ...
##  $ start_lat         : num  42 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num  -87.7 -87.6 -87.6 -87.6 -87.6 ...
##  $ end_lat           : num  42.1 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.7 -87.6 -87.6 -87.6 -87.6 ...
##  $ member_casual     : chr  "member" "member" "member" "casual" ...
```

Upon reviewing the output, it was discovered that the data type for `member_casual` was character instead of factor. Furthermore, `started_at` and `ended_at` were observed to be in character format instead of date/time. These issues will be addressed during the *Process* phase.

Summary

Based on our initial inspection of the data, we verified our data does not any duplicate entries. We did observe that several columns contain missing values. Furthermore, we noted data type issues for the following columns-`started_at`, `ended_at`, and `member_casual`.

Process

Our next step is to clean and transform Cyclistic trip data with the goal of preparing it for analysis. We'll tackle issues that were identified in the previous step, like inconsistent data types and missing values, to improve data quality. This process includes three main phases: data cleaning, data type conversion, and data transformation. We'll dive into the specifics of each sub-step to refine our data for analysis.

Cleaning the data

During the *Prepare* step, we discovered missing values ('NA') in several columns `end_lat`, `end_lng`, `start_station_id`, `start_station_name`, `end_station_id`, and `end_station_name`. The cleaning process was broken down into two steps:

1. Handling NA values
2. Handling missing string values

Handling NA Values

To address the issue of missing `end_lat` and `end_lng` values, we checked to see if we could infer these values using `end_station_id` or `end_station_name`.

Check for end_lat/end_lng values

```
all_trips %>%
  filter(is.na(end_lat) | is.na(end_lng)) %>%
  filter(end_station_id != "" | end_station_name != "")
```

```
##  [1] ride_id            rideable_type      started_at         ended_at
##  [5] start_station_name start_station_id   end_station_name   end_station_id
##  [9] start_lat          start_lng          end_lat            end_lng
## [13] member_casual
## <0 rows> (or 0-length row.names)
```

Unfortunately, since the `end_station_id` and `end_station_name` values are also missing, we cannot impute the `end_lat` and `end_lng` values for those entries. Thus, we have to remove them from our data set.

Omit NA values

```
all_trips_v2 <- na.omit(all_trips)
```

Handling Empty String Values

Recall that `start_station_id/ start_station_name` and `end_station_id/end_station_name` contain empty strings. To resolve this, we first created a dataset called `station_data` that contained the `latitude`, `longitude`, `station_id`, and `station_name` for all stations.

Create station dataset

```
#Create station data set
station_data <- all_trips_v2 %>%
  select(start_station_id, start_station_name, start_lat, start_lng) %>%
  bind_rows(all_trips_v2 %>% #row_bind end_station information to be thorough
            select(start_station_id = end_station_id,
                   start_station_name = end_station_name,
                   start_lat = end_lat,
                   start_lng = end_lng)) %>%
  arrange(start_lat, start_lng, desc(start_station_name)) %>% #desc(start_station_name) to prioritize n
  group_by(start_lat, start_lng) %>%
```

```
  summarize(start_station_id = first(start_station_id),
            start_station_name = first(start_station_name), .groups = "drop") %>%
  ungroup()
```

```
##   start_station_id      start_station_name start_lat start_lng
## 1            20215 Hegewisch Metra Station  41.64850 -87.54609
## 2            20215 Hegewisch Metra Station  41.64850 -87.54609
## 3            20215 Hegewisch Metra Station  41.64859 -87.54622
## 4            20215 Hegewisch Metra Station  41.64850 -87.54609
## 5            20215 Hegewisch Metra Station  41.64850 -87.54609
```

We now have a data set with every unique latitude and longitude value for every station. We then used two `left_join` functions to fill in our missing values. The first join filled `start_station_id` and `start_station_name`. The second join filled `end_station_id` and `end_station_name`.

First join - Fill Start Station ID/Name

```
all_trips_v3 <- all_trips_v2 %>% #fill start_station_id, start_station_name
  left_join(station_data, by=c("start_lat", "start_lng"), suffix = c("","_dup")) %>%
  mutate(start_station_id = ifelse(start_station_id_dup == "", "", start_station_id_dup),
         start_station_name = ifelse(start_station_name_dup == "", "", start_station_name_dup)) %>%
  select(-c(start_station_id_dup, start_station_name_dup))
```

Second join - Fill End Station ID/Name

```
#rename so we can repeat with end_stations
station_data <- station_data %>%
  rename(end_lat = start_lat, end_lng = start_lng, end_station_id = start_station_id, end_station_name =

all_trips_v3 <- all_trips_v3 %>% #fill end_station_id, end_station_name - 5,829,084
  left_join(station_data, by=c("end_lat", "end_lng"), suffix = c("","_dup")) %>%
  mutate(end_station_id = ifelse(end_station_id_dup == "", "", end_station_id_dup),
         end_station_name = ifelse(end_station_name_dup == "", "", end_station_name_dup)) %>%
  select(-c(end_station_id_dup, end_station_name_dup))
```

Values Filled

With this technique, we were able to recover some of our lost data, preventing it from being permanently lost. Now we can calculate how much of the remaining data is still missing.

Calculate missing values

```
all_trips_v3_missing <- all_trips_v3 %>%
  filter(start_station_name == "" | start_station_id =="" | end_station_name == "" | end_station_id ==

#calculate how much is still missing
missing_values = round(nrow(all_trips_v3_missing)/nrow(all_trips_v3)*100, 1)
missing_values
```

```
## [1] 14.6
```

While it is true that 14.6% of missing data is a notable amount, our data set contains nearly 6 million observations. We have a considerable amount of data for our analysis. Therefore, we will exclude the empty strings from our analysis.

Remove empty strings

```
all_trips_v4 <- all_trips_v3 %>%
  filter(start_station_name != "", start_station_id !="", end_station_name != "", end_station_id != "")
```

Summary

During the cleaning phase of the analysis, we addressed missing data issues in the dataset. Specifically, we dealt with missing values by either removing them or filling them in with appropriate values. These steps have helped us to improve the quality and integrity of our data and lay a solid foundation for further analysis.

Data Type Converstion

Recall that we had typing issues earlier in our data cleaning process. These typing issues could lead to inconsistencies in our data analysis and hinder our ability to draw accurate conclusions. To address this issue, we can use data type conversion to ensure that each variable in our dataset has the correct data type.

Correct data-typings

```
all_trips_v4 <- all_trips_v4 %>%
  mutate(
    rideable_type = factor(rideable_type),
    member_casual = factor(member_casual),
    started_at = as.POSIXct(started_at, format = "%Y-%m-%d %H:%M:%S"),
         ended_at = as.POSIXct(ended_at, format = "%Y-%m-%d %H:%M:%S"))
summary(all_trips_v4[c("rideable_type", "member_casual", "started_at", "ended_at")])
```

```
##        rideable_type     member_casual      started_at
##  classic_bike :2663434   casual:1998724   Min.   :2022-03-01 00:00:19
##  docked_bike  : 176404   member:2972787   1st Qu.:2022-06-07 08:18:32
##  electric_bike:2131673                    Median :2022-07-31 22:03:25
##                                           Mean   :2022-08-07 16:23:59
##                                           3rd Qu.:2022-09-30 11:30:16
##                                           Max.   :2023-02-28 23:59:31
##     ended_at
##  Min.   :2022-03-01 00:04:30
##  1st Qu.:2022-06-07 08:30:23
##  Median :2022-07-31 22:21:18
##  Mean   :2022-08-07 16:40:30
##  3rd Qu.:2022-09-30 11:44:56
##  Max.   :2023-03-01 09:48:38
```

Now that we have corrected our typings, we can move onto transforming the data to gain deeper insights.

Transform the data

In order to gain more meaningful insights from our data, we can perform some transformations on it. First, we can utilize the date/time information contained within the `started_at` and `ended_at` columns to calculate the length of each ride in minutes. Furthermore, we can use the `started_at` column to create a new categorical column, `started_at_hour`, which classifies each ride based on the hour it started. Additionally, we can extract the month, day, year, and day of the week from `started_at.` Lastly, we can use `start_lat`, `start_lng`, `end_lat`, `end_lng` as well as the `geosphere` package to calculate the distance between the two stations.

Calculate ride_length

```
all_trips_v5 <- all_trips_v4 %>%
  mutate(ride_length_minutes = as.numeric(difftime(ended_at, started_at, units = "secs"))/60)
```

Infer Date Information

```
all_trips_v5 <- all_trips_v5 %>%
  mutate(date = as.Date(started_at),
```

```r
        month = as.numeric(format(date, "%m")),
        day = as.numeric(format(date, "%d")),
        year = as.numeric(format(date, "%Y")),
        day_of_week = factor(wday(started_at, label = TRUE), levels = c("Mon", "Tue", "Wed", "Thu", "F
```

Append Hour of Day

```r
# Define time intervals
all_trips_v5 <- all_trips_v5 %>%
  mutate(started_at_hour = hour(started_at))
```

Calculate Distance Between Stations

```r
m_to_miles_converstion_rate = 0.000621371

all_trips_v5 <- all_trips_v5 %>%
  mutate(ride_distance_miles = distHaversine(
    p1 = cbind(start_lng, start_lat),
    p2 = cbind(end_lng, end_lat)
  ) * m_to_miles_converstion_rate)


all_trips_v5 %>%
  select(c(ride_length_minutes, date, month, day, year, day_of_week, started_at_hour, ride_distance_mil
  sample_n(5)
```

```
##   ride_length_minutes       date month day year day_of_week started_at_hour
## 1            4.883333 2022-04-28     4  28 2022         Thu              15
## 2           46.816667 2022-09-10     9  10 2022         Sat              15
## 3           16.416667 2023-02-13     2  13 2023         Mon               5
## 4            1.633333 2022-06-24     6  24 2022         Fri               9
## 5           12.100000 2022-04-12     4  12 2022         Mon              21
##   ride_distance_miles
## 1           0.9512884
## 2           5.2076062
## 3           2.1382582
## 4           0.1999746
## 5           2.4855300
```

Now that we have cleaned and transformed our data, we need to perform a check to ensure it is ready for the Analyze step.

Verify readiness

```r
summary(all_trips_v5)
```

```
##    ride_id                rideable_type       started_at
##  Length:4971511     classic_bike :2663434   Min.   :2022-03-01 00:00:19
##  Class :character   docked_bike  : 176404   1st Qu.:2022-06-07 08:18:32
##  Mode  :character   electric_bike:2131673   Median :2022-07-31 22:03:25
##                                             Mean   :2022-08-07 16:23:59
##                                             3rd Qu.:2022-09-30 11:30:16
##                                             Max.   :2023-02-28 23:59:31
##
##     ended_at                     start_station_name start_station_id
##  Min.   :2022-03-01 00:04:30   Length:4971511     Length:4971511
##  1st Qu.:2022-06-07 08:30:23   Class :character   Class :character
```

```
##  Median :2022-07-31 22:21:18   Mode  :character   Mode  :character
##  Mean   :2022-08-07 16:40:30
##  3rd Qu.:2022-09-30 11:44:56
##  Max.   :2023-03-01 09:48:38
##
##  end_station_name   end_station_id       start_lat       start_lng
##  Length:4971511     Length:4971511     Min.   :41.65   Min.   :-87.83
##  Class :character   Class :character   1st Qu.:41.88   1st Qu.:-87.66
##  Mode  :character   Mode  :character   Median :41.90   Median :-87.64
##                                        Mean   :41.90   Mean   :-87.65
##                                        3rd Qu.:41.93   3rd Qu.:-87.63
##                                        Max.   :42.06   Max.   :-87.53
##
##     end_lat         end_lng        member_casual    ride_length_minutes
##  Min.   : 0.00   Min.   :-87.83   casual:1998724   Min.   : -168.70
##  1st Qu.:41.88   1st Qu.:-87.66   member:2972787   1st Qu.:     5.85
##  Median :41.90   Median :-87.64                    Median :    10.32
##  Mean   :41.90   Mean   :-87.65                    Mean   :    16.52
##  3rd Qu.:41.93   3rd Qu.:-87.63                    3rd Qu.:    18.50
##  Max.   :42.06   Max.   :  0.00                    Max.   :34294.07
##
##       date                month             day             year
##  Min.   :2022-03-01   Min.   : 1.000   Min.   : 1.00   Min.   :2022
##  1st Qu.:2022-06-07   1st Qu.: 5.000   1st Qu.: 8.00   1st Qu.:2022
##  Median :2022-08-01   Median : 7.000   Median :16.00   Median :2022
##  Mean   :2022-08-07   Mean   : 6.929   Mean   :15.71   Mean   :2022
##  3rd Qu.:2022-09-30   3rd Qu.: 9.000   3rd Qu.:23.00   3rd Qu.:2022
##  Max.   :2023-03-01   Max.   :12.000   Max.   :31.00   Max.   :2023
##
##  day_of_week  started_at_hour ride_distance_miles
##  Mon:663589   Min.   : 0.0    Min.   :   0.000
##  Tue:702226   1st Qu.:11.0    1st Qu.:   0.540
##  Wed:701985   Median :15.0    Median :   0.968
##  Thu:731717   Mean   :14.2    Mean   :   1.310
##  Fri:694568   3rd Qu.:18.0    3rd Qu.:   1.706
##  Sat:793910   Max.   :23.0    Max.   :6105.009
##  Sun:683516
```

Upon reviewing the summary, we noticed negative values and an abnormally high maximum value for `ride_length_minutes`. Additionally, we observed that there was an occurrence of "0" for `end_lat` and `end_lng`, which is not consistent with the expected values for the city of Chicago. Moreover, `ride_distance_miles` also had a remarkably high maximum value. Lastly, we observe that rideable_type has an unusually low value of 'docked_bike'. To better understand these discrepancies, further investigation is necessary.

Investigation

1) rideable_type After further research into this matter we discovered that 'docked_bike' is an outdated value for 'classic_bike'. We will make this adjustment and move forward.

```
all_trips_v5 <- all_trips_v5 %>%
  mutate(rideable_type = fct_recode(rideable_type, "classic_bike" = "docked_bike"))

summary(all_trips_v5["rideable_type"])
```

```
##        rideable_type
##   classic_bike :2839838
##   electric_bike:2131673
```

2) end_lat and end_lng

```
##     end_station_id        end_station_name  end_lat   end_lng
## 1    chargingstx07 Green St & Madison Ave* 41.88183 -87.64883
## 2    chargingstx07 Green St & Madison Ave* 41.88188 -87.64895
## 3    chargingstx07 Green St & Madison Ave*  0.00000   0.00000
## 4    chargingstx07 Green St & Madison Ave*  0.00000   0.00000
## 5    chargingstx07 Green St & Madison Ave*  0.00000   0.00000
## 6    chargingstx07 Green St & Madison Ave*  0.00000   0.00000
## 7    chargingstx07 Green St & Madison Ave* 41.88188 -87.64895
## 8    chargingstx07 Green St & Madison Ave*  0.00000   0.00000
## 9    chargingstx07 Green St & Madison Ave*  0.00000   0.00000
## 10   chargingstx07 Green St & Madison Ave*  0.00000   0.00000
## 11   chargingstx07 Green St & Madison Ave*  0.00000   0.00000
```

Upon inspection of the data, we discovered that the `0` values for `end_lat` and `end_lng` were specifically associated with `end_station_id` equaling "chargingstx07." Although we are uncertain how this occurred, we noticed that `chargingstx07` did possess `end_lat`/`end_lng` values. We computed the average of these values and used that average to replace the `0` values. Consequently, we also had to re-compute `ride_length_minutes` for the afflicted entries.

```r
#create values to fill
chargingstx07 <-  all_trips_v5 %>%
  filter(end_station_id == "chargingstx07" & end_lat != 0 & end_lng != 0 )

chargingstx07_lat_mean = mean(chargingstx07$end_lat)
chargingstx07_lng_mean = mean(chargingstx07$end_lng)

#fill the 0 values
all_trips_v6 <- all_trips_v5 %>%
    mutate(end_lat = if_else(end_lat == 0, chargingstx07_lat_mean, end_lat),
           end_lng = if_else(end_lng == 0, chargingstx07_lng_mean, end_lng))

#check if end_lat/end_lng changed from all_trips_v5 to minimize re-calculations
all_trips_v6 <- all_trips_v6 %>%
  mutate(ride_distance_miles =
           if_else(end_lat != all_trips_v5$end_lat | end_lng != all_trips_v5$end_lng,
                   distHaversine(p1 = cbind(start_lng, start_lat),p2 = cbind(end_lng, end_lat)) * m_to_r
                                ride_distance_miles))

summary(all_trips_v6[c("end_lat", "end_lng")])
```

```
##     end_lat         end_lng
##  Min.   :41.65   Min.   :-87.83
##  1st Qu.:41.88   1st Qu.:-87.66
##  Median :41.90   Median :-87.64
##  Mean   :41.90   Mean   :-87.65
##  3rd Qu.:41.93   3rd Qu.:-87.63
##  Max.   :42.06   Max.   :-87.53
```

This resolved the issue. We continued to investigate the issue with `ride_length_minutes`.

3) ride_length_minutes

A) Negative Values After examining `ride_length_minutes`, we noticed that the phrase "Hubbard Bike-checking (LBS-WH-TEST)" appears in both the `start_station_id` and `end_station_id` columns for certain entries. Further research was done to discover that LBS and WH are shorthand for "Local Bike Store" and "Warehouse". We believe that these entries correspond to bikes that were taken to a warehouse for inspection or testing, which caused inconsistencies in `started_at` and `ended_at` resulting in negative values for `ride_lengths_seconds`. Therefore, to ensure the accuracy of our analysis, these entries were excluded.

```
inspection_site = "Hubbard Bike-checking (LBS-WH-TEST)"
all_trips_v6 <- all_trips_v6 %>%
  filter(start_station_id != inspection_site, end_station_id != inspection_site, ride_length_minutes > 0

all_trips_v6 %>%
  filter(is.na(ride_length_minutes))
```

```
##  [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       ride_length_minutes date
## [16] month               day                 year
## [19] day_of_week         started_at_hour     ride_distance_miles
## <0 rows> (or 0-length row.names)
```

```
all_trips_v6 %>%
  filter(ride_length_minutes<0)
```

```
##  [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       ride_length_minutes date
## [16] month               day                 year
## [19] day_of_week         started_at_hour     ride_distance_miles
## <0 rows> (or 0-length row.names)
```

B) Outlier Values After examining the ride_length_minutes variable, we observed a significant outlier in the maximum value. To investigate further, we examined a sample of the outliers in terms of days.

Sample of Outliers (in days)

```
##  [1] 22.25  7.51  7.45  6.92  6.08  5.72  4.63  4.05  3.37  3.04  2.67  2.65
## [13]  2.50  2.40  2.07  1.98  1.89  1.89  1.88  1.84  1.75  1.59  1.50  1.38
## [25]  1.35  1.25  1.17  1.10  1.08  1.07  1.07  1.07  1.06  1.04  1.04  1.04
## [37]  1.04  1.04  1.04  1.04  1.04  1.04  1.04  1.04  1.04  1.04  1.04  1.04
## [49]  1.04  1.04
```

With values ranging from one day to just over a week we find one observation with a ride length of over 22 days. As this seems unreasonable was removed.

```
summary(all_trips_v6[c("ride_length_minutes", "ride_distance_miles")])
```

```
##  ride_length_minutes ride_distance_miles
##  Min.   :    0.017   Min.   : 0.0000
##  1st Qu.:    5.850   1st Qu.: 0.5402
##  Median :   10.317   Median : 0.9678
```

```
##  Mean   :   16.497   Mean   : 1.2998
##  3rd Qu.:   18.500   3rd Qu.: 1.7055
##  Max.   :10807.217   Max.   :19.0921
```

By removing the outlier for ride_length_minutes, we were also able to resolve the issue with the ride_distance_miles outlier. Therefore, we have completed the *Process* phase of our analysis.

Summary

We successfully addressed various data quality issues. We also enriched our data by creating new features based off existing ones. These steps allowed us to create a more robust and accurate dataset. We shall proceed to the exploratory analysis phase and begin to extract meaningful insights.

Analyze

The goal of this phase was to explore the data and extract valuable insights that aided us in achieving our business objective. As a reminder, our business task at hand is to figure out how annual members and casual riders bikes differently with the aim of converting casual riders into annual members.

To begin, we will compare the number of rides taken by members and casual users.

```
member_casual_table <- table(all_trips_v6$member_casual)
member_casual_prop <- prop.table(member_casual_table)*100
member_casual_prop
```

```
##
##   casual   member
## 40.19882 59.80118
```

In terms of the total number of rides, about 40.2% were by casual customers whereas 59.8% of rides were by member customers. Let's dive deeper into the data and compute some summary statistics.

Ride Duration Summary

```
#Let's take a look at these values by comparing members vs casual users
all_trips_v6 %>%
  filter(ride_length_minutes>0) %>%
  group_by(member_casual) %>%
  summarise(
    mean_ride_length_minutes = mean(ride_length_minutes),
    median_ride_length_minutes = median(ride_length_minutes),
    min_ride_length_seconds = min(ride_length_minutes)*60,
    max_ride_length_hours = max(ride_length_minutes)/60
  )
```

```
## # A tibble: 2 x 5
##   member_casual mean_ride_length_minutes median_ride_length_~ min_ride_length_s~
##   <fct>                            <dbl>                <dbl>              <dbl>
## 1 casual                            22.7                 13.2                  1
## 2 member                            12.3                  8.83                 1
## # ... with 1 more variable: max_ride_length_hours <dbl>
```

We have observed that casual users tend to have rides that last twice as long on average compared to members. Furthermore, while the longest ride by a member was 25 hours, we have identified a casual user with a ride lasting over 7 days. Next, we will examine the breakdown of ride duration by day.

```
all_trips_v6 %>%
  group_by(member_casual, day_of_week) %>%
```

```r
  summarise(mean_ride_length_minutes = mean(ride_length_minutes)) %>%
  arrange(member_casual, day_of_week)
```

```
## # A tibble: 14 x 3
## # Groups:   member_casual [2]
##    member_casual day_of_week mean_ride_length_minutes
##    <fct>         <ord>                          <dbl>
##  1 casual        Mon                             23.2
##  2 casual        Tue                             20.2
##  3 casual        Wed                             19.5
##  4 casual        Thu                             20.2
##  5 casual        Fri                             21.2
##  6 casual        Sat                             25.5
##  7 casual        Sun                             26.0
##  8 member        Mon                             11.9
##  9 member        Tue                             11.7
## 10 member        Wed                             11.8
## 11 member        Thu                             12.0
## 12 member        Fri                             12.1
## 13 member        Sat                             13.8
## 14 member        Sun                             13.7
```

We see that for both members and casual users Saturday and Sunday have the longest rides on average. Let's see the frequency of rides over this period.

```r
#Let's analyze ridership data by type and weekday
all_trips_v6 %>%
  group_by(member_casual, day_of_week) %>%
  summarise(
    mean_ride_length_minutes = mean(ride_length_minutes),
    number_of_rides = n()) %>%
  arrange(member_casual, day_of_week)
```

```
## # A tibble: 14 x 4
## # Groups:   member_casual [2]
##    member_casual day_of_week mean_ride_length_minutes number_of_rides
##    <fct>         <ord>                          <dbl>           <int>
##  1 casual        Mon                             23.2          240313
##  2 casual        Tue                             20.2          228754
##  3 casual        Wed                             19.5          234414
##  4 casual        Thu                             20.2          262629
##  5 casual        Fri                             21.2          283686
##  6 casual        Sat                             25.5          408089
##  7 casual        Sun                             26.0          339673
##  8 member        Mon                             11.9          422942
##  9 member        Tue                             11.7          473103
## 10 member        Wed                             11.8          467237
## 11 member        Thu                             12.0          468797
## 12 member        Fri                             12.1          410579
## 13 member        Sat                             13.8          385469
## 14 member        Sun                             13.7          343511
```

We observed that there are some differences in the behavior of casual and member users on weekends. Specifically for casual users, on Saturday and Sunday the number of rides tends to be higher and the average ride time is longer. On the other hand, for members, the number of rides decreases over the weekend, but the average ride duration increases. This pattern could suggest that members use the ride service less frequently on weekends, but when they do use it, they tend to take more leisurely rides. Additionally, our analysis shows that members use our service more during the weekdays for shorter periods of time.

Distance Summary

```
#Let's take a look at these values by comparing members vs casual users
all_trips_v6 %>%
  filter(ride_distance_miles>0) %>%
  group_by(member_casual) %>%
  summarise(
    mean_ride_distance_miles = mean(ride_distance_miles),
    median_ride_distance_miles = median(ride_distance_miles),
    min_ride_distance_miles = min(ride_distance_miles),
    max_ride_distance_miles = max(ride_distance_miles)
  )
```

```
## # A tibble: 2 x 5
##   member_casual mean_ride_distance_miles median_ride_distanc~ min_ride_distance~
##   <fct>                            <dbl>                <dbl>              <dbl>
## 1 casual                            1.44                1.10           0.0000115
## 2 member                            1.33                0.965          0.0000125
## # ... with 1 more variable: max_ride_distance_miles <dbl>
```

Although there is a difference in the time spent on rides, it appears that member and casual users exhibit similar behavior in terms of distance traveled.

Bike Type Summary

```
all_trips_v6 %>%
  group_by(member_casual, rideable_type) %>%
  summarise(
    number_of_rides = n(),
    mean_ride_length_minutes = mean(ride_length_minutes),
    mean_ride_distance_miles = mean(ride_distance_miles)
  )
```

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

```
## # A tibble: 4 x 5
## # Groups:   member_casual [2]
##   member_casual rideable_type number_of_rides mean_ride_length~ mean_ride_dista~
##   <fct>         <fct>                   <int>             <dbl>            <dbl>
## 1 casual        classic_bike          1079469              28.4             1.30
## 2 casual        electric_bike          918089              16.0             1.37
## 3 member        classic_bike          1760137              13.2             1.20
## 4 member        electric_bike         1211501              11.1             1.40
```

Based on the data presented in the table, casual users display a mild inclination towards classic bikes, whereas member users exhibit a stronger preference for classic bikes over electric bikes. It is noteworthy that both user groups tend to spend more time on classic bikes than electric bikes, although this could be attributed to the fact that electric bikes can attain higher speeds with less effort. Additionally, it appears that both casual and member users tend to travel farther on average when using electric bikes compared to classic bikes.

Summary

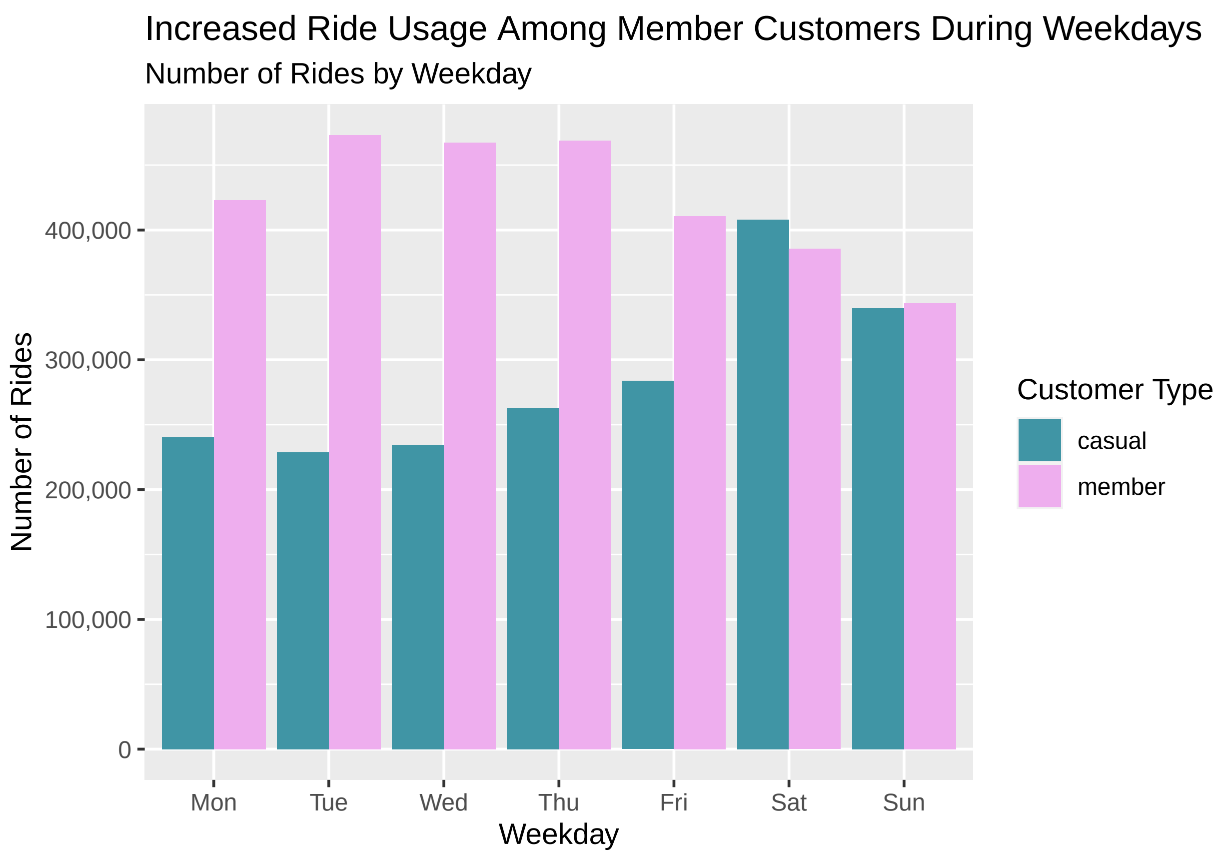
During the analysis portion of the report, we compared the number of rides taken by members and casual users and computed some summary statistics. We observed that casual users tend to have longer rides on average compared to members, and there are some differences in the behavior of casual and member users on weekends. We also noted that members use the service more during weekdays for shorter periods of time, and both user groups tend to spend more time on classic bikes than electric bikes. Additionally, we found that both casual and member users tend to travel farther on average when using electric bikes compared to classic bikes. In the following section, we have included visualizations to help illustrate some of our findings.

Share

Now that we have completed our analysis and gained insights from the data, we would like to present our findings to you. This section is dedicated to providing visualizations that will aid our explanation of our analysis and highlight the trends and patterns in the data. We hope that the following charts and graphs will help to illustrate our key findings and provide a clearer picture of the data.

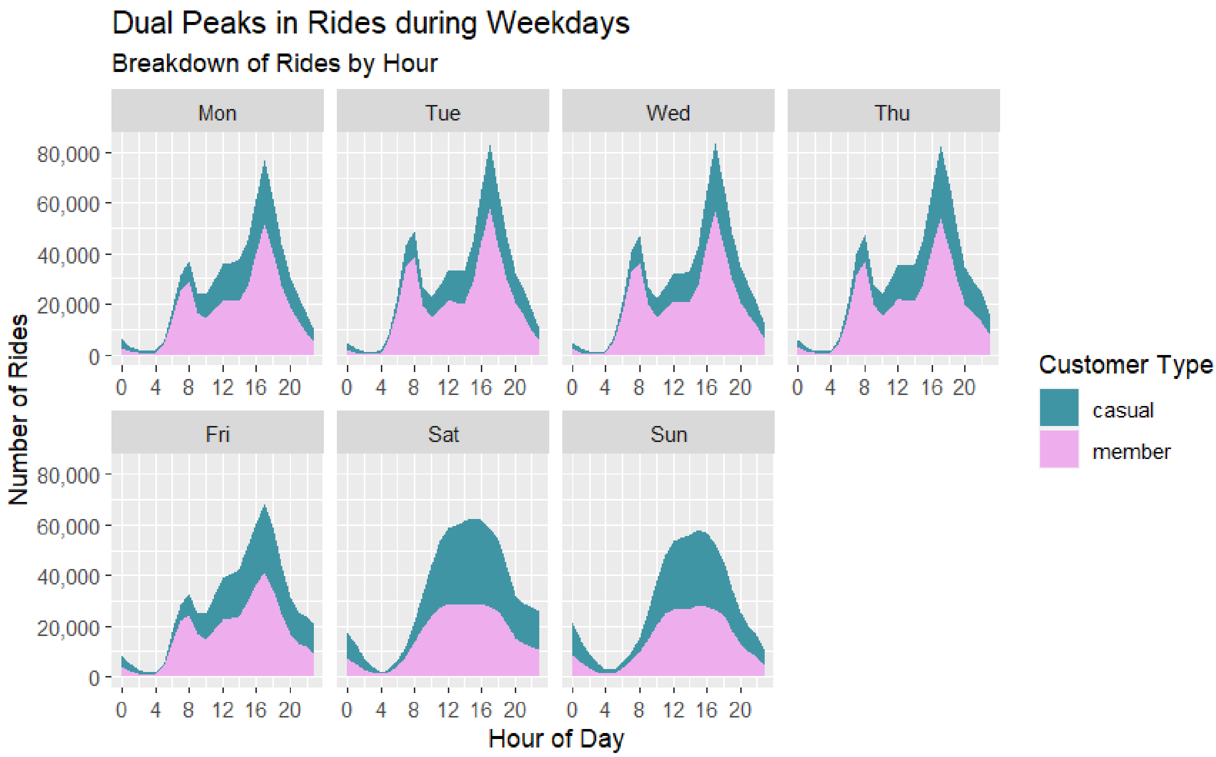
```
#Set graph colors
theme_colors = c("#4095A5", "plum2")
```

```
usage_by_weekday <- all_trips_v6 %>%
  group_by(member_casual, day_of_week) %>%
  summarise(number_of_rides = n()) %>%
  arrange(member_casual, day_of_week)

ggplot(usage_by_weekday, aes(x = day_of_week, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge") + #'dodge'  positions the columns adjacent to each other rather than sta
  labs(title = "Increased Ride Usage Among Member Customers During Weekdays",
       subtitle = "Number of Rides by Weekday",
       x = "Weekday", y = "Number of Rides",
       fill = "Customer Type"
       ) +
  scale_y_continuous(labels = scales::comma_format()) +
  scale_fill_manual(values = theme_colors)
```



Here we present a detailed analysis of our customers' frequency of usage based on the day of the week. Our findings reveal that members use our service considerably more on weekdays, whereas the service usage is evenly distributed between member and casual users on weekends. Moving forward, we will delve deeper into the hourly usage pattern for each day of the week.

```
all_trips_v6 %>%
  group_by(member_casual, day_of_week, started_at_hour) %>%
  summarise(number_of_rides = n()) %>%
  ggplot(aes(x = started_at_hour, y = number_of_rides, fill = member_casual)) +
  geom_area(position = "stack") +
  facet_wrap(~day_of_week, ncol = 4, scales = "free_x") +
  labs(title = "Dual Peaks in Rides during Weekdays",
       subtitle = "Breakdown of Rides by Hour",
       x = "Hour of Day",
       y = "Number of Rides",
       fill = "Customer Type") +
  scale_x_continuous(breaks = seq(0,23, by=4)) +
  scale_y_continuous(labels = scales::comma_format()) +
  scale_fill_manual(values = theme_colors)
```



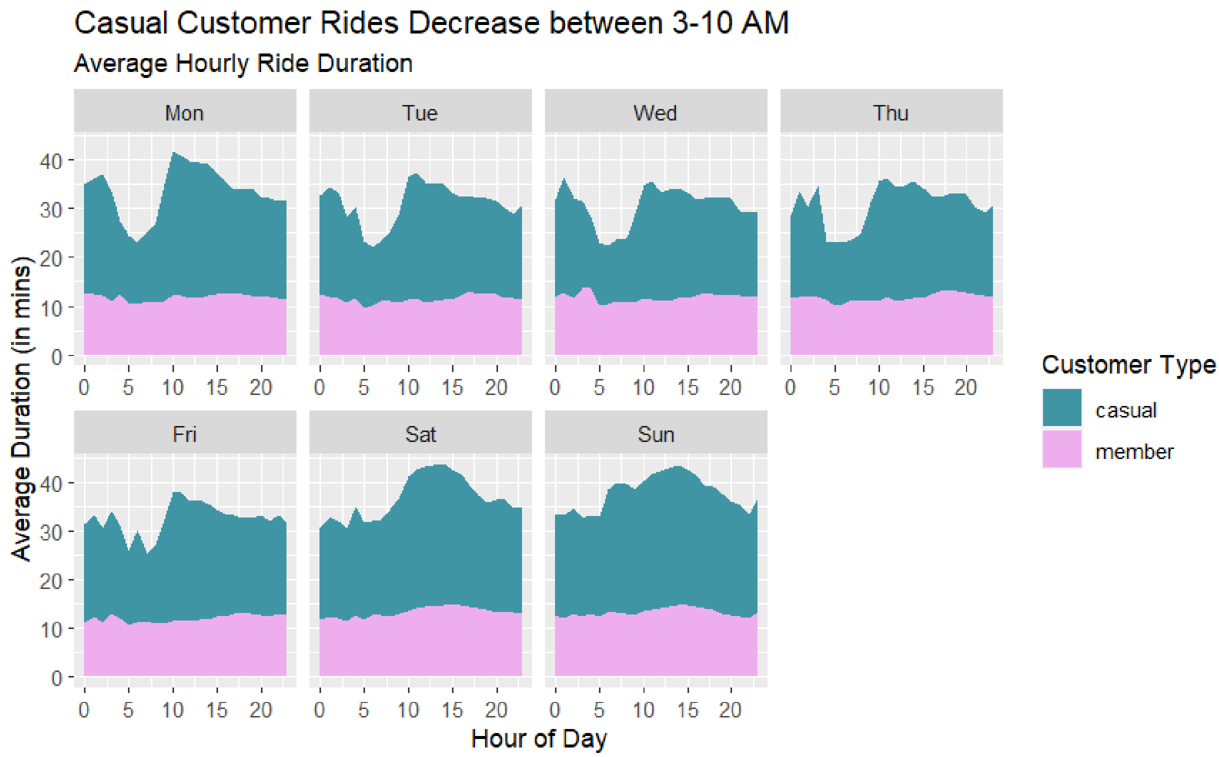
)

In the analysis of the hourly usage patterns by day of the week, it was observed that both user groups exhibit a dual-peaked trend during the weekdays with peak usage times between 6-8 AM and 16-18 (4-6 PM). On the weekends, the hourly usage patterns are observed to be more evenly distributed throughout the day. Next, let's explore the difference in ride durations.

```
all_trips_v6 %>%
  group_by(member_casual, day_of_week) %>%
  summarise(number_of_rides = dplyr::n(),
            mean_ride_length_minutes = mean(ride_length_minutes)) %>%
  arrange(member_casual, day_of_week) %>%
  ggplot(aes(x = day_of_week, y = mean_ride_length_minutes, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(title = "Casual Customers Ride Nearly Twice as Long as Annual Members",
       subtitle = "Average Ride Duration by Weekday",
       x = "Weekday", y = "Number of Rides",
       fill = "Customer Type"
       ) +
  scale_y_continuous(labels = scales::comma_format())+
  scale_fill_manual(values = theme_colors)
```

We can see that casual riders have an average ride duration that is nearly twice as long as that of members. Now, let's take a closer look and examine how the ride durations vary by hour.

```
all_trips_v6 %>%
  group_by(member_casual, started_at_hour, day_of_week) %>%
  summarise(average_duration = mean(ride_length_minutes)) %>%
  ggplot(aes(x = started_at_hour, y = average_duration, fill = member_casual)) +
  geom_area(position = "stack") +
  facet_wrap(~day_of_week, ncol = 4, scales = "free_x") +
  labs(title = "Casual Customer Rides Decrease between 3-10 AM",
       subtitle = "Average Hourly Ride Duration",
       x = "Hour of Day",
       y = "Average Duration (in mins)",
       fill = "Customer Type") +
  scale_fill_manual(values = theme_colors)
```



)

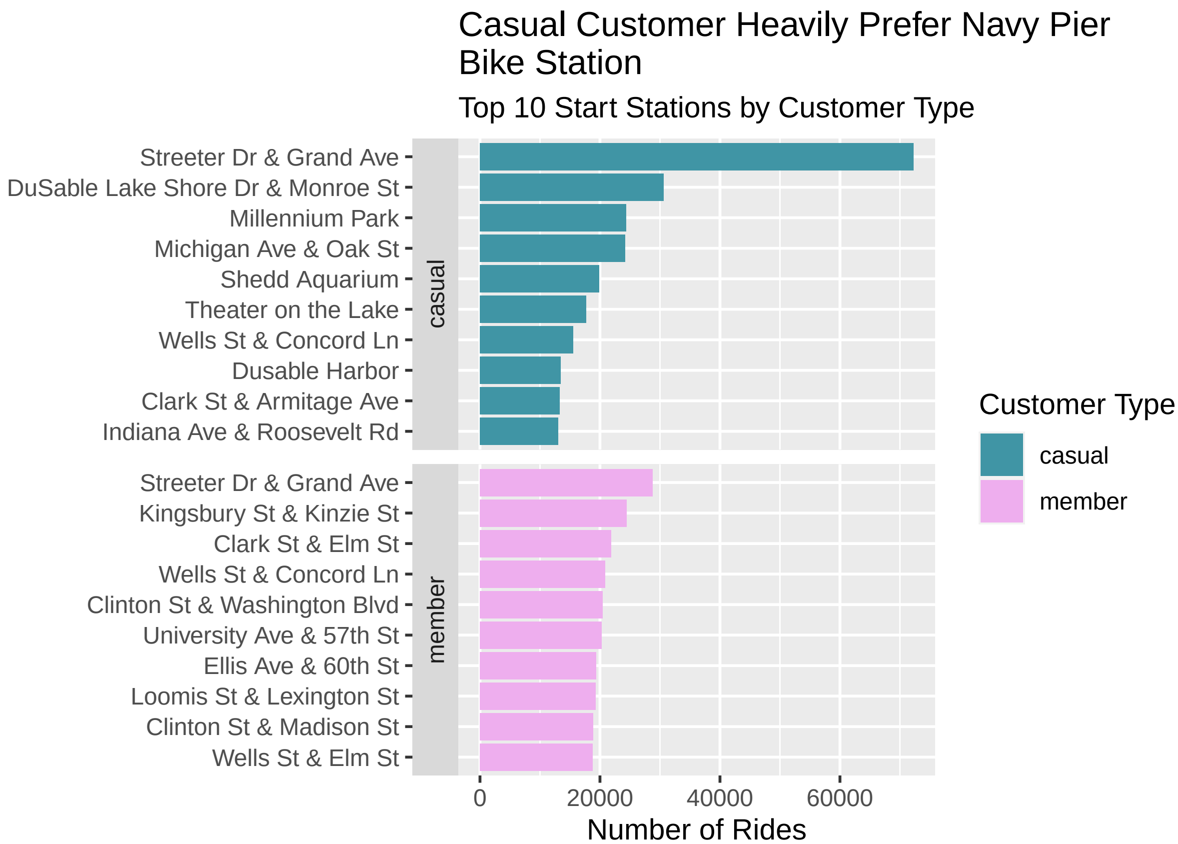
Based on the information given, we can pull the following insights:

1. Casual riders tend to have shorter average ride lengths between 3-10 AM.
2. On weekends, casual riders tend to have slightly longer rides compared to weekdays.

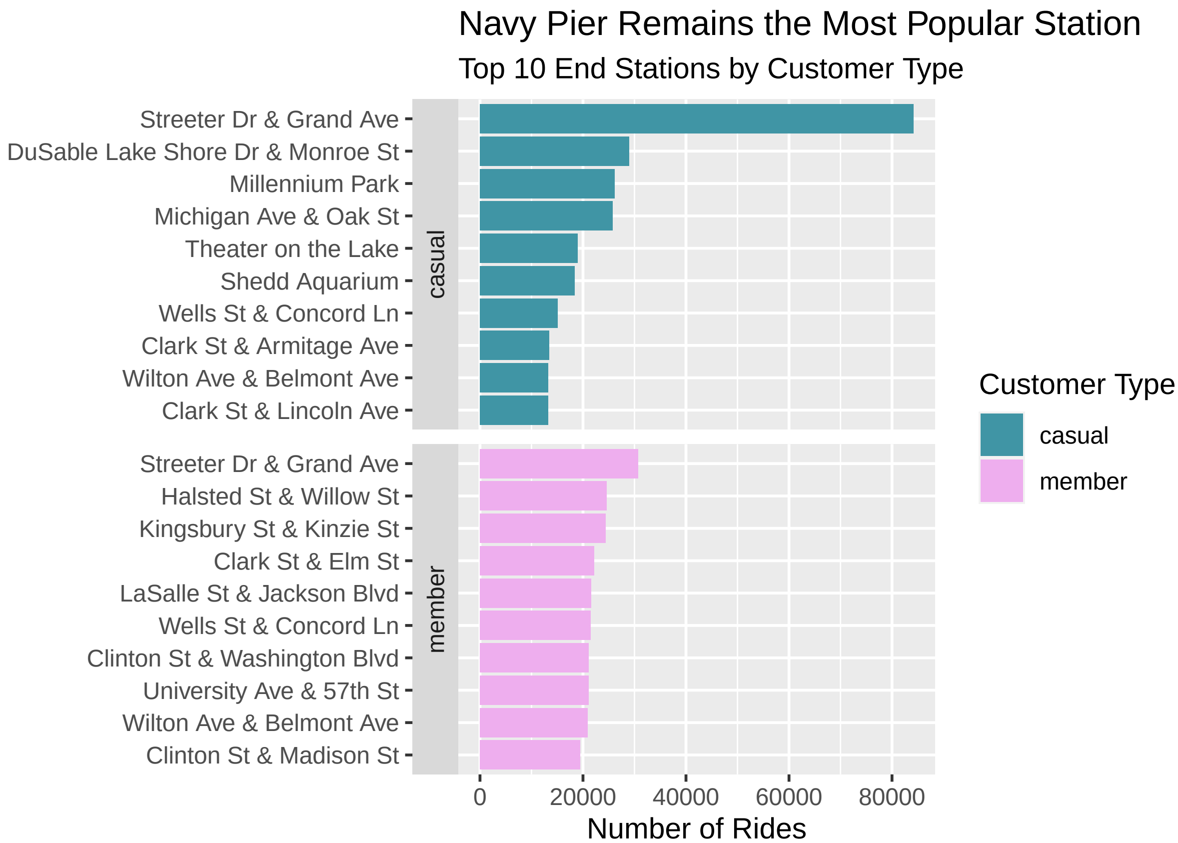
We believe this decrease in ride length could indicate that some casual users are already using our services to commute, as the shorter rides during early morning hours may suggest a pattern of commuting behavior. Additionally, the difference in ride length between casual riders on weekends and weekdays may suggest that bike-sharing is more popular as a leisure activity on weekends or could be a factor of tourism. To investigate that hypothesis further, we will now analyze the start and end stations that are most frequently used by our customers.

```r
# Top 10 Start Stations by Member_Casual
start_stations <- all_trips_v6 %>%
  group_by(member_casual, start_station_name) %>%
  summarise(number_of_rides = n()) %>%
  arrange(member_casual, desc(number_of_rides)) %>%
  top_n(10) %>%
  mutate(start_station_name = factor(start_station_name),
         start_station_name = reorder_within(start_station_name, number_of_rides, member_casual))

ggplot(start_stations, aes(x = number_of_rides, y =  start_station_name, fill = member_casual)) +
  geom_col() +
  facet_grid(member_casual ~ ., scales = "free_y", switch = "y") +
  scale_y_reordered()+
  labs(title = "Casual Customer Heavily Prefer Navy Pier \nBike Station",
       subtitle = "Top 10 Start Stations by Customer Type",
       x = "Number of Rides",
       y = "",
       fill = "Customer Type") +
```

```
  scale_fill_manual(values = theme_colors)
```




```
# Top 10 End Stations by Member_Casual
end_stations <- all_trips_v6 %>%
  group_by(member_casual, end_station_name) %>%
  summarise(number_of_rides = n()) %>%
  arrange(member_casual, desc(number_of_rides)) %>%
  top_n(10) %>%
  mutate(end_station_name = factor(end_station_name),
         end_station_name = reorder_within(end_station_name, number_of_rides, member_casual))

ggplot(end_stations, aes(x = number_of_rides, y =  end_station_name, fill = member_casual)) +
  geom_col() +
  facet_grid(member_casual ~ ., scales = "free_y", switch = "y") +
  scale_y_reordered()+
  labs(title = "Navy Pier Remains the Most Popular Station",
       subtitle = "Top 10 End Stations by Customer Type",
       x = "Number of Rides",
       y = "",
       fill = "Customer Type") +
  scale_fill_manual(values = theme_colors)
```



Based on the graphs we have examined, it appears that the station located at `Streeter Dr & Grand Ave`, situated on Navy Pier, a renowned tourist attraction, is the most frequently used one.

Act

1) Offer promotions during peak usage hours, such as discounted rates for members. Based on our findings, we observed that casual and members customers both have peaks in usage during weekdays around the morning and evening. By targeting these high-traffic times and providing incentives for membership, we can potentially increase our number of members.

2) Implement a targeted digital marketing campaign to convert casual customers who use the service for decreased ride times between 3-10 AM. The campaign would focus on customers who have shorter ride times between 3-10 AM, as this could indicate a pattern of commuting behavior. By identifying these customers through further clustering analysis, we could target them with incentives to become annual members. This would require collecting customer data such as customer IDs to accurately track riding habits and ensure that the marketing campaign is reaching the right audience.

3) Consider planning specialized services or events for members during weekends, such as guided tours or group rides, to encourage more member customers to use our service for leisure. Our analysis found that member customers had a decrease in the number of rides but an increase in ride times on weekends. By adding member-only events, we could encourage member customers to ride more on weekends as well as entice casual riders to become members to access these events. Furthermore, since our analysis found that Navy Pier is the most popular station among casual customers, creating events that start and end at Navy Pier could increase our appeal to tourists and draw in more casual riders.

Conclusion

In conclusion, our analysis of bike-share usage in Chicago reveals interesting insights into the behavior of our customers. Members tend to use our service primarily for commuting purposes during the weekdays, while casual riders use the service for leisure or tourism. On weekends, we observe increased usage by both member and casual riders, suggesting that bike-sharing is a popular leisure activity during this time.

Additionally, we found that the most popular start and end station is located on Navy Pier, a well-known tourist destination. This information could be useful for future business planning, decision making, and targeted marketing efforts.

To maintain the integrity of our analysis and enable further exploration, we are saving the cleaned data used in this report as a CSV file. This will allow us to easily import the data into other tools for further analysis and insights.

```
to_csv <- all_trips_v6

if (!dir.exists("processed_csv")) {
  dir.create("processed_csv")
}
write_csv(to_csv , file = paste0(getwd(), "/processed_csv/trip_data_cleaned.csv"))
```