Cyclistic Bike-Share Analysis

Introduction

Cyclistic is a bike-share company offering a range of bicycles across the city. This analysis aims to understand user behavior and trends, focusing on casual riders and annual members to convert casual riders to member riders and help guide the marketing strategy.

PHASE 1: ASK

Business Task:

How can cyclistic maximize the number of annual memberships?

Consider Key stack-holders:

Lily Moreno: director of marketing, Cyclistic marketing analytics team,Cyclistic executive team.

PHASE 2: PREPARE

Data Collection

- I downloaded the raw data for the year 2022, divided into monthly `.csv` files.
- Each file contained trip-level information, including ride details, user types, and timestamps.

Data Cleaning in Excel

The data cleaning process was performed in Microsoft Excel. The following steps were taken:

1. **Removing Unnecessary Columns**:
   - Dropped columns such as `start_station_name`, `start_station_id`, `end_station_name`,`end_station_id`,`start_l` as they were not required for this analysis because the data in these columns are not complete.
2. **Handling Missing Values**:
   - Removed rows containing `NA` values in critical columns like `ride_id`, `rideable_type`, or `member_casual`.
3. **Removed Duplicate Entries**:
   - Duplicate entries in some of the files were identified and subsequently removed to ensure data accuracy and integrity.

This cleaned files serves as the foundation for my analysis and will be imported into R for further analysis for trends in user behavior.

PHASE 3: PROCESS

- Initially, we will install the necessary packages, followed by uploading our xlsx files and binding them together in R for further analysis.

```
# Load the tidyverse package
library(tidyverse)
```

```
## -- 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   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- 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 errors
```

```
library(readxl)

# Loading xlsx file
Jan_2022 <- read_excel("Jan_2022.xlsx")
Feb_2022 <- read_excel("Feb_2022.xlsx")
Mar_2022 <- read_excel("Mar_2022.xlsx")
Apr_2022 <- read_excel("Apr_2022.xlsx")
May_2022 <- read_excel("May_2022.xlsx")
Jun_2022 <- read_excel("Jun_2022.xlsx")
July_2022 <- read_excel("July_2022.xlsx")
Aug_2022 <- read_excel("Aug_2022.xlsx")
Sep_2022 <- read_excel("Sep_2022.xlsx")
Oct_2022 <- read_excel("Oct_2022.xlsx")
Nov_2022 <- read_excel("Nov_2022.xlsx")
Dec_2022 <- read_excel("Dec_2022.xlsx")

# The data from all 12 CSV files have been imported and are now being combined into a single dataset.
all_trips <- bind_rows(Jan_2022,Feb_2022,Mar_2022,Apr_2022,May_2022,Jun_2022,July_2022,Aug_2022,Sep_202
```

```
# Let's take a peek into the data now.
summary(all_trips)
```

```
##    ride_id          rideable_type        started_at
##  Length:5667712     Length:5667712     Min.   :2022-01-01 00:00:05
##  Class :character   Class :character   1st Qu.:2022-05-28 19:21:09
##  Mode  :character   Mode  :character   Median :2022-07-22 15:04:03
##                                        Mean   :2022-07-20 07:21:32
##                                        3rd Qu.:2022-09-16 07:21:31
##                                        Max.   :2022-12-31 23:59:26
##     ended_at                    member_casual
##  Min.   :2022-01-01 00:01:48   Length:5667712
##  1st Qu.:2022-05-28 19:43:10   Class :character
##  Median :2022-07-22 15:24:51   Mode  :character
##  Mean   :2022-07-20 07:40:59
##  3rd Qu.:2022-09-16 07:39:03
##  Max.   :2023-01-02 04:56:45
```

```
head(all_trips)
```

```
## # A tibble: 6 x 5
##   ride_id    rideable_type started_at          ended_at            member_casual
##   <chr>      <chr>         <dttm>              <dttm>              <chr>
## 1 FFFE5FA26~ electric_bike 2022-01-13 13:22:00 2022-01-13 13:28:33 member
## 2 FFFE4D9D5~ classic_bike  2022-01-19 17:57:02 2022-01-19 18:12:12 member
## 3 FFFD03555~ classic_bike  2022-01-03 10:40:41 2022-01-03 10:41:24 member
## 4 FFFB93713~ electric_bike 2022-01-05 15:46:44 2022-01-05 16:17:14 member
## 5 FFFAB2A19~ electric_bike 2022-01-11 15:45:33 2022-01-11 16:01:34 member
```

```
## 6 FFF9DA5B0~ electric_bike 2022-01-12 07:19:43 2022-01-12 07:40:25 casual
```

```
# First let's make a function to convert seconds into time for `ride_length column`.
seconds_to_hms <- function(seconds) {
  if (!is.numeric(seconds)) {
    stop("Input must be numeric")
  }

  hours <- floor(seconds / 3600)
  minutes <- floor((seconds %% 3600) / 60)
  secs <- round(seconds %% 60)

  # Format the time
  sprintf("%02d:%02d:%02d", hours, minutes, secs)
}
```

```
# We are going to add columns `ride_length_seconds`, `ride_length` and `weekday`.
all_trips <- all_trips %>%
  mutate(ride_length_seconds = as.numeric(difftime(ended_at, started_at, units = "sec")))

all_trips <- all_trips %>%
  mutate(ride_length = sprintf("%02d:%02d:%02d",
                                    ride_length_seconds %/% 3600,
                                    (ride_length_seconds %% 3600) %/% 60,
                                    ride_length_seconds %% 60))

all_trips$day_of_week <- wday(all_trips$started_at)
head(all_trips, 10)
```

```
## # A tibble: 10 x 8
##    ride_id   rideable_type started_at          ended_at            member_casual
##    <chr>     <chr>         <dttm>              <dttm>              <chr>
##  1 FFFE5FA2~ electric_bike 2022-01-13 13:22:00 2022-01-13 13:28:33 member
##  2 FFFE4D9D~ classic_bike  2022-01-19 17:57:02 2022-01-19 18:12:12 member
##  3 FFFD0355~ classic_bike  2022-01-03 10:40:41 2022-01-03 10:41:24 member
##  4 FFFB9371~ electric_bike 2022-01-05 15:46:44 2022-01-05 16:17:14 member
##  5 FFFAB2A1~ electric_bike 2022-01-11 15:45:33 2022-01-11 16:01:34 member
##  6 FFF9DA5B~ electric_bike 2022-01-12 07:19:43 2022-01-12 07:40:25 casual
##  7 FFF95772~ electric_bike 2022-01-26 16:01:21 2022-01-26 16:21:56 member
##  8 FFF9363B~ classic_bike  2022-01-13 17:13:09 2022-01-13 17:17:21 member
##  9 FFF7B333~ electric_bike 2022-01-31 16:27:42 2022-01-31 16:38:25 member
## 10 FFF79DF8~ classic_bike  2022-01-25 08:22:14 2022-01-25 08:51:44 member
## # i 3 more variables: ride_length_seconds <dbl>, ride_length <chr>,
## #   day_of_week <dbl>
```

PHASE 4: ANALYZE

- In this phase we are going to do descriptive analysis and see what insight we gain from the data.

```
# Let's perform a descriptive analysis. Since the data is initially in seconds, we will convert it to a
Descriptive_analysis <- all_trips %>%
  summarize(
    mean_ride_length = mean(ride_length_seconds, na.rm = TRUE),
    median_ride_length = median(ride_length_seconds, na.rm = TRUE),
    max_ride_length = max(ride_length_seconds, na.rm = TRUE),
    min_ride_length = min(ride_length_seconds, na.rm = TRUE)
```

```
  ) %>%
  mutate(mean_ride_length_time = seconds_to_hms(mean_ride_length) %>%
           format(format = "%H:%M:%S"),
         median_ride_length_time = seconds_to_hms(median_ride_length) %>%
           format(format = "%H:%M:%S"),
         max_ride_length_time = seconds_to_hms(max_ride_length) %>%
           format(format = "%H:%M:%S"),
         min_ride_length_time = seconds_to_hms(min_ride_length) %>%
           format(format = "%H:%M:%S")
  )
head(Descriptive_analysis)
```

```
## # A tibble: 1 x 8
##   mean_ride_length median_ride_length max_ride_length min_ride_length
##              <dbl>              <dbl>           <dbl>           <dbl>
## 1            1167.                617         2483235         -621201
## # i 4 more variables: mean_ride_length_time <chr>,
## #   median_ride_length_time <chr>, max_ride_length_time <chr>,
## #   min_ride_length_time <chr>
```

- It appears that the data is displaying unexpected minimum and maximum values. Let's see maximum rides.

```
maximum_rides <- all_trips %>% arrange(desc(ride_length_seconds))
head(maximum_rides,10)
```

```
## # A tibble: 10 x 8
##    ride_id   rideable_type started_at          ended_at            member_casual
##    <chr>     <chr>         <dttm>              <dttm>              <chr>
##  1 7D4CB0DD~ docked_bike   2022-10-01 15:04:38 2022-10-30 08:51:53 casual
##  2 DCFE0DB8~ docked_bike   2022-05-08 00:28:53 2022-06-02 04:46:41 casual
##  3 94DD1FB2~ docked_bike   2022-06-15 07:56:59 2022-07-10 04:57:37 casual
##  4 23697816~ docked_bike   2022-03-05 19:08:58 2022-03-29 15:43:02 casual
##  5 70835A30~ docked_bike   2022-07-09 01:02:46 2022-08-01 19:11:35 casual
##  6 3BFD0599~ docked_bike   2022-07-09 01:03:19 2022-08-01 19:11:26 casual
##  7 A256444C~ docked_bike   2022-07-09 01:02:45 2022-08-01 18:51:57 casual
##  8 59F28D6F~ docked_bike   2022-06-10 16:13:34 2022-07-03 04:16:32 casual
##  9 DC510E6F~ docked_bike   2022-07-04 18:37:11 2022-07-27 00:32:38 casual
## 10 4BF7A8BC~ docked_bike   2022-10-09 11:24:11 2022-10-31 04:33:40 casual
## # i 3 more variables: ride_length_seconds <dbl>, ride_length <chr>,
## #   day_of_week <dbl>
```

- The docked bikes are showing unexpected results, taking hours, possibly due to issues such as data entry errors, incorrect timestamps, or outliers in ride duration. This could happen if the start and end times are recorded inaccurately, leading to unusually long ride durations. those are total 5361 entries and also these entries csv file is attached on github name Extra_hours.

```
negative_rides <- all_trips %>% filter(ride_length_seconds < 0)
head(negative_rides, 10)
```

```
## # A tibble: 10 x 8
##    ride_id   rideable_type started_at          ended_at            member_casual
##    <chr>     <chr>         <dttm>              <dttm>              <chr>
##  1 2D97E3C9~ classic_bike  2022-03-05 11:00:57 2022-03-05 10:55:01 casual
##  2 7407049C~ electric_bike 2022-03-05 11:38:04 2022-03-05 11:37:57 casual
##  3 0793C920~ electric_bike 2022-05-30 11:06:29 2022-05-30 11:06:17 casual
```

```
##  4 B897BE02~ electric_bike 2022-06-07 19:15:39 2022-06-07 17:05:37 casual
##  5 072E947E~ electric_bike 2022-06-07 19:14:46 2022-06-07 17:07:45 casual
##  6 6A871510~ electric_bike 2022-06-23 19:22:57 2022-06-23 19:21:46 member
##  7 BF114472~ electric_bike 2022-06-07 19:14:47 2022-06-07 17:05:42 member
##  8 B3BE8FE6~ electric_bike 2022-06-07 16:18:37 2022-06-07 16:07:28 member
##  9 EF3CCF2B~ electric_bike 2022-06-07 18:47:01 2022-06-07 17:05:41 casual
## 10 BBD84670~ electric_bike 2022-06-07 19:11:33 2022-06-07 17:05:24 casual
## # i 3 more variables: ride_length_seconds <dbl>, ride_length <chr>,
## #   day_of_week <dbl>
```

- We have identified 100 entries with negative time values, which we have decided to remove as they may distort the results. Additionally, we have filtered out any entries where the ride duration exceeds 24 hours, as these are considered outliers for our analysis.

```
# Here we will remove those unusual long rides duration and negative ride length
all_trips$ride_length_seconds <- abs(all_trips$ride_length_seconds)

all_trips$Extra_Hours <- ifelse(all_trips$member_casual > 86400, all_trips$ride_length_seconds - 86400,
nrow(all_trips[all_trips$ride_length_seconds >= 86400,])
```

```
## [1] 5361
```

```
Extra_hours <- all_trips[all_trips$ride_length_seconds >= 86400,]
all_trips <- all_trips[all_trips$ride_length_seconds < 86400,]
```

```
# Now let's see summary statistics after removing unexpected max right length and negative ride length.
summary_statistics <- all_trips %>%
  group_by(member_casual) %>%
  summarize(
    mean_ride_length = mean(ride_length_seconds),
    median_ride_length = median(ride_length_seconds),
    max_ride_length = max(ride_length_seconds),
    min_ride_length = min(ride_length_seconds)
  ) %>%
  mutate(
    mean_ride_length_time = seconds_to_hms(mean_ride_length) %>%
      format(format = "%H:%M:%S"),
    median_ride_length_time = seconds_to_hms(median_ride_length) %>%
      format(format = "%H:%M:%S"),
    max_ride_length_time = seconds_to_hms(max_ride_length) %>%
      format(format = "%H:%M:%S"),
    min_ride_length_time = seconds_to_hms(min_ride_length) %>%
      format(format = "%H:%M:%S")
  )
print(summary_statistics)
```

```
## # A tibble: 2 x 9
##   member_casual mean_ride_length median_ride_length max_ride_length
##   <chr>                   <dbl>              <dbl>           <dbl>
## 1 casual                  1311.                778           86396
## 2 member                   744.                530           86390
## # i 5 more variables: min_ride_length <dbl>, mean_ride_length_time <chr>,
## #   median_ride_length_time <chr>, max_ride_length_time <chr>,
## #   min_ride_length_time <chr>
```

- Now we are going to aggregate the data.

```r
# Let's aggregate `ride_length_seconds`, `member_casual`, and `day_of_week`.
average_ride_time <- aggregate(all_trips$ride_length_seconds ~ all_trips$member_casual + all_trips$day_

average_ride_time <- average_ride_time %>%
  mutate('all_trips$ride_length_time' = seconds_to_hms(`all_trips$ride_length_seconds`))
print(average_ride_time)
```

```
##    all_trips$member_casual all_trips$day_of_week all_trips$ride_length_seconds
## 1                   casual                      1                      1499.2794
## 2                   member                      1                       820.5358
## 3                   casual                      2                      1340.2940
## 4                   member                      2                       718.4944
## 5                   casual                      3                      1174.8069
## 6                   member                      3                       707.8353
## 7                   casual                      4                      1130.8497
## 8                   member                      4                       709.4227
## 9                   casual                      5                      1169.9969
## 10                  member                      5                       719.4908
## 11                  casual                      6                      1229.6088
## 12                  member                      6                       732.7440
## 13                  casual                      7                      1470.2194
## 14                  member                      7                       827.0692
##    all_trips$ride_length_time
## 1                    00:24:59
## 2                    00:13:41
## 3                    00:22:20
## 4                    00:11:58
## 5                    00:19:35
## 6                    00:11:48
## 7                    00:18:51
## 8                    00:11:49
## 9                    00:19:30
## 10                   00:11:59
## 11                   00:20:30
## 12                   00:12:13
## 13                   00:24:30
## 14                   00:13:47
```

```r
# Let's calculate average ride length for the users.
average_ride_length <- all_trips %>%
  group_by(member_casual) %>%
  summarize(average_ride_length = mean(ride_length_seconds))

average_ride_length <- average_ride_length %>%
  mutate(average_ride_length_time = seconds_to_hms(average_ride_length) %>%
           format(format = "%H:%M:%S"))
print(average_ride_length)
```

```
## # A tibble: 2 x 3
##   member_casual average_ride_length average_ride_length_time
##   <chr>                      <dbl> <chr>
## 1 casual                     1311. 00:21:51
## 2 member                      744. 00:12:24
```

```r
# Now let's calculate the total rides taken by users.
total_rides <- all_trips %>%
  group_by(day_of_week, member_casual) %>%
  summarize(count_of_rides = n())
```

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

```r
print(total_rides)
```

```
## # A tibble: 14 x 3
## # Groups:   day_of_week [7]
##    day_of_week member_casual count_of_rides
##          <dbl> <chr>                  <int>
##  1           1 casual                 388128
##  2           1 member                 387129
##  3           2 casual                 277137
##  4           2 member                 473245
##  5           3 casual                 263252
##  6           3 member                 518508
##  7           4 casual                 273874
##  8           4 member                 523769
##  9           5 casual                 308760
## 10           5 member                 532153
## 11           6 casual                 334004
## 12           6 member                 466985
## 13           7 casual                 472232
## 14           7 member                 443175
```

```r
# Let's see how users use bikes differently.
bike_usage <- all_trips %>%
  group_by(member_casual, rideable_type, day_of_week) %>%
  summarise(
    total_rides = n(),
    .groups = 'drop'
  )
print(bike_usage)
```

```
## # A tibble: 35 x 4
##    member_casual rideable_type day_of_week total_rides
##    <chr>         <chr>                <dbl>       <int>
##  1 casual        classic_bike             1      158107
##  2 casual        classic_bike             2      103973
##  3 casual        classic_bike             3       95820
##  4 casual        classic_bike             4       98063
##  5 casual        classic_bike             5      113500
##  6 casual        classic_bike             6      122724
##  7 casual        classic_bike             7      196665
##  8 casual        docked_bike              1       35295
##  9 casual        docked_bike              2       22281
## 10 casual        docked_bike              3       17567
## # i 25 more rows
```

- we have done with the analysis let's download the data because we are going to do visualization on Tableau.

```r
# Downloading the files.
write.csv(average_ride_length, "average_ride_length.csv", row.names = FALSE)
write.csv(average_ride_time, "average_ride_time.csv", row.names = FALSE)
write.csv(bike_usage, "bike_usage.csv", row.names = FALSE)
write.csv(Descriptive_analysis, "Descriptive_analysis.csv", row.names = FALSE)
write.csv(Extra_hours,"Extra_hours.csv", row.names = FALSE)
write.csv(summary_statistics, "summary_statistics.csv", row.names = FALSE)
write.csv(total_rides, "total_rides.csv", row.names = FALSE)
write.csv(all_trips, "all_rides.csv", row.names = FALSE)
```

PHASE 5: SHARE

- I chose Tableau because it provides a clearer and more interactive view of my data. ggplot2, while powerful for static visualizations, often requires significant customization to untangle complex datasets, which can be time-consuming. Tableau's user-friendly interface, real-time capabilities, and ability to handle large datasets seamlessly make it the ideal tool for my analysis and presentation needs.
- To view the visualizations of the data, please refer to the PNG files available on the GitHub repository.

Results and Findings:

- **Usage Patterns:** Casual riders tend to use the bikes more on weekends, while members have a more consistent usage pattern throughout the week. Additionally, casual riders prefer electric bikes, while members show a preference for classic bikes.
- **Ride Duration:** Casual riders typically have longer ride durations than members.
- **Analysis Summary:**

1. **Classic Bikes:** Classic bikes are used by both casual and member users. However, member users show a higher and more consistent usage pattern across the week compared to casual users. **Member users:** The highest number of rides is observed on day 3 (Wednesday), with 268,250. Usage remains relatively stable, with minor fluctuations, ending the week with 227,226 rides on day 7 (Sunday). **Casual users:** Usage starts strong on day 1 (Monday) with 158,107 rides but experiences a significant drop in the middle of the week, reaching a low of 95,820 rides on day 3 (Wednesday). There's a recovery towards the end of the week, with 196,665 rides on day 7 (Sunday). **Interpretation:** Members prefer classic bikes for commuting, showing a steady pattern throughout the week, while casual users, likely influenced by external factors like weather or events, show more variability.
2. **Docked Bikes** Docked bikes are almost exclusively used by casual users, with very limited or no usage by members. **Member users:** No significant usage was observed. **Casual users:** The number of rides starts at 35,295 on day 1 (Monday) and decreases steadily, hitting a low of 17,155 on day 4 (Thursday). Usage then increases towards the weekend, peaking at 40,505 rides on day 7 (Sunday). **Interpretation:** Docked bikes are primarily favored by casual users, especially during the weekend, possibly for leisure or recreational activities. The lack of member usage suggests that this bike type may not meet the needs or preferences of regular commuters.
3. **Electric Bikes** Electric bikes are popular among both casual and member users, with member users again showing higher engagement throughout the week. **Member users:** Electric bike usage starts strong with 225,937 rides on day 2 (Tuesday). The number of rides peaks at 264,806 on day 5 (Friday) and then slightly decreases to 215,949 on day 7 (Sunday). **Casual users:** Usage starts at 186,132 rides on day 1 (Monday) but shows a dip in the middle of the week, reaching 149,865 rides on day 3 (Wednesday). There's a steady increase towards the end of the week, with 235,062 rides on day 7 (Sunday). **Interpretation:** Electric bikes are well-utilized by both user groups, with members consistently favoring them for their likely convenience and speed. Casual users also show a strong preference, particularly towards the weekend, indicating a possible combination of commuting and leisure use.

PHASE 6: ACT

CONCLUSION:

The analysis of bike usage across different types and user groups reveals distinct patterns: - Classic Bikes are favored by members for consistent use throughout the week, with casual users showing more fluctuation. - Docked Bikes are predominantly used by casual users, particularly during the weekend, while members show little to no interest. - Electric Bikes are popular among both casual and member users, with members showing a higher and more consistent usage.

These insights can inform operational decisions, such as bike availability, marketing strategies, and membership promotions, ensuring that the needs of both casual and member users are met effectively.

RECOMMENDATIONS:

- Cyclistic should focus on converting casual riders to annual members by promoting weekday benefits.
- Increase availability of classic bikes during peak casual rider times.
- Launch targeted marketing campaigns for casual riders during weekends.