Bike-Sharing Analysis Report

Rishabh Jaiswal

2024-05-23

Introduction

Background

As a junior data analyst working in the marketing analyst team at Cyclistic, a bike-share company in Chicago. The director of marketing believes the company's future success depends on maximizing the number of annual memberships. Therefore, my team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, we will design a new marketing strategy to convert casual riders into annual members.

Objective of the Analysis

Analyze the behavioral patterns and usage metrics of casual riders compared to annual paid members in our bike share program to identify key differences. Use these insights to develop targeted strategies and recommendations aimed at converting casual riders into annual members

Data description

This analysis is based on the Divvy case study "'Sophisticated, Clear, and Polished': Divvy and Data Visualization" written by Kevin Hartman (found here: https://artscience.blog/home/divvy-dataviz-case-study).

Data preparation

Installing and loading necessary libraries

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

```
## Installing package into 'C:/Users/Saurabh/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Saurabh\AppData\Local\Temp\RtmpsD0YD4\downloaded_packages
```

```
install.packages("readxl")
```

```
## Installing package into 'C:/Users/Saurabh/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'readxl' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Saurabh\AppData\Local\Temp\RtmpsD0YD4\downloaded_packages
```

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

```
## Installing package into 'C:/Users/Saurabh/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'lubridate' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Saurabh\AppData\Local\Temp\RtmpsD0YD4\downloaded_packages
```

```
install.packages("janitor")
```

```
## Installing package into 'C:/Users/Saurabh/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'janitor' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Saurabh\AppData\Local\Temp\RtmpsD0YD4\downloaded_packages
```

```
install.packages("hms")
```

```
## Installing package into 'C:/Users/Saurabh/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'hms' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Saurabh\AppData\Local\Temp\RtmpsD0YD4\downloaded_packages
```

```
library("tidyverse")
```

```
## ── 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_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library("readxl")
library("lubridate")
library("hms")

##
## Attaching package: 'hms'
##
## The following object is masked from 'package:lubridate':
##
##     hms

library("janitor")

##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

Loading datasets

```
bike_share_01 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202301-divvy-
tripdata.xlsx")
bike_share_02 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202302-divvy-
tripdata.xlsx")
bike_share_03 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202303-divvy-
tripdata.xlsx")
bike_share_04 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202304-divvy-
tripdata.xlsx")
bike_share_05 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202305-divvy-
tripdata.xlsx")
bike_share_06 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202306-divvy-
tripdata.xlsx")
bike_share_07 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202307-divvy-
tripdata.xlsx")
bike_share_08 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202308-divvy-
tripdata.xlsx")
bike_share_09 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202309-divvy-
tripdata.xlsx")
bike_share_10 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202310-divvy-
tripdata.xlsx")
```

```
bike_share_11 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202311-divvy-tripdata.xlsx")
bike_share_12 <-
read_excel("C:/Users/Saurabh/Documents/bike_share_success/xlsx/202312-divvy-tripdata.xlsx")
```

Merging all the data in one dataframe

```
bike_share_2023 <- bind_rows(bike_share_01, bike_share_02, bike_share_03, bike_share_04, bike_share_05, bike_share_06, bike_share_07, bike_share_08, bike_share_09, bike_share_10, bike_share_11, bike_share_12)
```

Data cleaning and preprocessing

Add columns that list the date, month, day, and year of each ride, this will allow us to aggregate ride data for each month, day, or year

```
bike_share_2023$date <- as.Date(bike_share_2023$started_at) #The default format is yyyy-mm-dd
bike_share_2023$month <- format(as.Date(bike_share_2023$date), "%m")
bike_share_2023$day <- format(as.Date(bike_share_2023$date), "%d")
bike_share_2023$year <- format(as.Date(bike_share_2023$date), "%Y")
bike_share_2023 <- bike_share_2023 %>% select(-date)
```

Ensuring that every column values have relevant data-type

```
bike_share_2023$month <- as.numeric(bike_share_2023$month)
bike_share_2023$day <- as.numeric(bike_share_2023$day)
bike_share_2023$year <- as.numeric(bike_share_2023$year)
```

Adding a new column with day_name corresponding to day_of_week and month_name corresponding to month

```
bike_share_2023 <- bike_share_2023 %>% mutate(day_name = factor(day_of_week, levels = 1:7, labels = c("sun", "mon", "tue", "wed","thu", "fri", "sat")))
bike_share_2023 <- bike_share_2023 %>% mutate(month_name = factor(month, levels = 1:12, labels = c("jan", "feb", "mar", "apr", "may", "jun", "jul", "aug", "sep", "oct", "nov", "dec")))
```

Removing the date values from the column ride_length

```
bike_share_2023$ride_length <- as_hms(bike_share_2023$ride_length)
```

Remove rows with missing ride_id, member_casual, ride_length, start_station_name, end_station_name

```
bike_share_2023 <- bike_share_2023 %>% drop_na(ride_id, member_casual, ride_length, start_station_name, end_station_name)
```

Remove duplicate rows if any

```
bike_share_2023 <- bike_share_2023 %>% distinct()
```

Ensure ended_at is not before than started_at

```
bike_share_2023 <- bike_share_2023 %>% filter(is.na(ended_at) | ended_at >=
started_at)
```

Ensuring that column names are unique and consistent

```
clean_names(bike_share_2023)
```

```
## # A tibble: 4,332,034 × 14
##    ride_id          rideable_type started_at          ended_at
##    <chr>            <chr>         <dttm>              <dttm>
##  1 F96D5A74A3E41399 electric_bike 2023-01-21 20:05:00 2023-01-21 20:16:00
##  2 13CB7EB698CEDB88 classic_bike  2023-01-10 15:37:00 2023-01-10 15:46:00
##  3 BD88A2E670661CE5 electric_bike 2023-01-02 07:51:00 2023-01-02 08:05:00
##  4 C90792D034FED968 classic_bike  2023-01-22 10:52:00 2023-01-22 11:01:00
##  5 3397017529188E8A classic_bike  2023-01-12 13:58:00 2023-01-12 14:13:00
##  6 58E68156DAE3E311 electric_bike 2023-01-31 07:18:00 2023-01-31 07:21:00
##  7 2F7194B6012A98D4 electric_bike 2023-01-15 21:18:00 2023-01-15 21:32:00
##  8 DB1CF84154D6A049 classic_bike  2023-01-25 10:49:00 2023-01-25 10:58:00
##  9 34EAB943F88C4C5D electric_bike 2023-01-25 20:49:00 2023-01-25 21:02:00
## 10 BC8AB1AA51DA9115 classic_bike  2023-01-06 16:37:00 2023-01-06 16:49:00
## # ℹ 4,332,024 more rows
## # ℹ 10 more variables: start_station_name <chr>, end_station_name <chr>,
## #   member_casual <chr>, ride_length <time>, day_of_week <dbl>, month
<dbl>,
## #   day <dbl>, year <dbl>, day_name <fct>, month_name <fct>
```

Inspect the dataset

```
head(bike_share_2023)
```

```
## # A tibble: 6 × 14
##   ride_id          rideable_type started_at          ended_at
##   <chr>            <chr>         <dttm>              <dttm>
## 1 F96D5A74A3E41399 electric_bike 2023-01-21 20:05:00 2023-01-21 20:16:00
## 2 13CB7EB698CEDB88 classic_bike  2023-01-10 15:37:00 2023-01-10 15:46:00
## 3 BD88A2E670661CE5 electric_bike 2023-01-02 07:51:00 2023-01-02 08:05:00
## 4 C90792D034FED968 classic_bike  2023-01-22 10:52:00 2023-01-22 11:01:00
## 5 3397017529188E8A classic_bike  2023-01-12 13:58:00 2023-01-12 14:13:00
## 6 58E68156DAE3E311 electric_bike 2023-01-31 07:18:00 2023-01-31 07:21:00
## # ℹ 10 more variables: start_station_name <chr>, end_station_name <chr>,
## #   member_casual <chr>, ride_length <time>, day_of_week <dbl>, month
<dbl>,
## #   day <dbl>, year <dbl>, day_name <fct>, month_name <fct>
```

```
colnames(bike_share_2023)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "end_station_name"
##  [7] "member_casual"      "ride_length"        "day_of_week"
```

```
## [10] "month"              "day"                "year"
## [13] "day_name"           "month_name"

glimpse(bike_share_2023)

## Rows: 4,332,034
## Columns: 14
## $ ride_id            <chr> "F96D5A74A3E41399", "13CB7EB698CEDB88",
"BD88A2E670…
## $ rideable_type      <chr> "electric_bike", "classic_bike",
"electric_bike", "…
## $ started_at         <dttm> 2023-01-21 20:05:00, 2023-01-10 15:37:00,
2023-01-…
## $ ended_at           <dttm> 2023-01-21 20:16:00, 2023-01-10 15:46:00,
2023-01-…
## $ start_station_name <chr> "Lincoln Ave & Fullerton Ave", "Kimbark Ave &
53rd …
## $ end_station_name   <chr> "Hampden Ct & Diversey Ave", "Greenwood Ave &
47th …
## $ member_casual      <chr> "member", "member", "casual", "member",
"member", "…
## $ ride_length        <time> 00:11:00, 00:09:00, 00:14:00, 00:09:00,
00:15:00, …
## $ day_of_week        <dbl> 7, 3, 2, 1, 5, 3, 1, 4, 4, 6, 5, 3, 2, 3, 5, 2,
7, …
## $ month              <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, …
## $ day                <dbl> 21, 10, 2, 22, 12, 31, 15, 25, 25, 6, 5, 3, 9,
3, 1…
## $ year               <dbl> 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023,
202…
## $ day_name           <fct> sat, tue, mon, sun, thu, tue, sun, wed, wed,
fri, t…
## $ month_name         <fct> jan, jan, jan, jan, jan, jan, jan, jan, jan,
jan, j…
```

Calculations to start analysis of data

Calculating average ride duration and number of rides by usertypes

```
avg_ride_length_per_usertype <- bike_share_2023 %>% group_by(member_casual)
%>% summarise(avg_ride_length_per_usertype =
seconds_to_period(mean(ride_length)), no_ride = n())
avg_ride_length_per_usertype

## # A tibble: 2 × 3
##   member_casual avg_ride_length_per_usertype no_ride
##   <chr>         <Period>                       <int>
## 1 casual        22M 49.2003618992655S        1531918
## 2 member        12M 6.89585717163106S        2800116
```

Finding the variation of the metric- avg_ride_length_per_usertype through different months and filtering and sorting for casual riders

```
avg_rl_ut_mon <- bike_share_2023 %>% group_by(month_name, member_casual) %>%
summarise(avg_rl_ut_mon = seconds_to_period(mean(ride_length)), no_ride =
n())

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

avg_rl_ut_mon

## # A tibble: 24 × 4
## # Groups:   month_name [12]
##    month_name member_casual avg_rl_ut_mon         no_ride
##    <fct>      <chr>         <Period>                <int>
##  1 jan        casual        14M 49.8902805442085S   29621
##  2 jan        member        9M 57.3519968313627S   118663
##  3 feb        casual        17M 38.6499817318231S   32844
##  4 feb        member        10M 23.5999726406867S  116962
##  5 mar        casual        16M 39.5375277825269S   46792
##  6 mar        member        10M 9.17015391624091S  153655
##  7 apr        casual        22M 29.4192042555501S  110538
##  8 apr        member        11M 33.3464071253727S  213659
##  9 may        casual        24M 20.7188246657515S  177039
## 10 may        member        12M 41.573860539226S   286188
## # ℹ 14 more rows

avg_rl_cas_mon <- avg_rl_ut_mon %>% filter(member_casual == "casual")
avg_rl_cas_mon <- avg_rl_cas_mon %>% arrange(-avg_rl_ut_mon)
avg_rl_cas_mon

## # A tibble: 12 × 4
## # Groups:   month_name [12]
##    month_name member_casual avg_rl_ut_mon         no_ride
##    <fct>      <chr>         <Period>                <int>
##  1 jul        casual        25M 7.55700830212049S  245359
##  2 may        casual        24M 20.7188246657515S  177039
##  3 aug        casual        24M 13.6677298263317S  233894
##  4 jun        casual        23M 53.7975651034558S  219804
##  5 sep        casual        23M 25.9184719972786S  196963
##  6 apr        casual        22M 29.4192042555501S  110538
##  7 oct        casual        21M 18.8988488104374S  130300
##  8 nov        casual        17M 45.9835178556564S   72078
##  9 feb        casual        17M 38.6499817318231S   32844
## 10 mar        casual        16M 39.5375277825269S   46792
## 11 dec        casual        16M 32.1114321539552S   36686
## 12 jan        casual        14M 49.8902805442085S   29621
```

```
avg_nr_cas_mon <- avg_rl_cas_mon %>% arrange(-no_ride)
avg_nr_cas_mon
```

```
## # A tibble: 12 × 4
## # Groups:   month_name [12]
##    month_name member_casual avg_rl_ut_mon         no_ride
##    <fct>      <chr>         <Period>                <int>
##  1 jul        casual        25M 7.55700830212049S  245359
##  2 aug        casual        24M 13.6677298263317S  233894
##  3 jun        casual        23M 53.7975651034558S  219804
##  4 sep        casual        23M 25.9184719972786S  196963
##  5 may        casual        24M 20.7188246657515S  177039
##  6 oct        casual        21M 18.8988488104374S  130300
##  7 apr        casual        22M 29.4192042555501S  110538
##  8 nov        casual        17M 45.9835178556564S   72078
##  9 mar        casual        16M 39.5375277825269S   46792
## 10 dec        casual        16M 32.1114321539552S   36686
## 11 feb        casual        17M 38.6499817318231S   32844
## 12 jan        casual        14M 49.8902805442085S   29621
```

Calculating average ride length of users by rideable_type i.e electric or classic

```
avg_rl_ut_rt <- bike_share_2023 %>% group_by(rideable_type, member_casual)
%>% summarise(avg_rl_ut_rt = seconds_to_period(mean(ride_length)))
```

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

```
avg_rl_ut_rt
```

```
## # A tibble: 5 × 3
## # Groups:   rideable_type [3]
##   rideable_type member_casual avg_rl_ut_rt
##   <chr>         <chr>         <Period>
## 1 classic_bike  casual        25M 38.4544686076106S
## 2 classic_bike  member        12M 58.37194922704S
## 3 docked_bike   casual        52M 50.7371458551938S
## 4 electric_bike casual        14M 39.8420958684043S
## 5 electric_bike member        10M 31.6353896679012S
```

```
avg_rl_ut_rt_mon <- bike_share_2023 %>% group_by(month_name, rideable_type,
member_casual) %>% summarise(avg_rl_ut_rt_mon =
seconds_to_period(mean(ride_length)))
```

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

```
avg_rl_ut_rt_mon <- avg_rl_ut_rt_mon %>% filter(member_casual == "casual")
avg_rl_ut_rt_mon
```

```
## # A tibble: 32 × 4
## # Groups:   month_name, rideable_type [32]
##    month_name rideable_type member_casual avg_rl_ut_rt_mon
##    <fct>      <chr>         <chr>         <Period>
##  1 jan        classic_bike  casual        17M 13.8961038961038S
##  2 jan        docked_bike   casual        37M 35.564803804994S
##  3 jan        electric_bike casual        9M 44.9691029192414S
##  4 feb        classic_bike  casual        20M 16.6313614053215S
##  5 feb        docked_bike   casual        42M 56.0390516039051S
##  6 feb        electric_bike casual        11M 23.2086264711684S
##  7 mar        classic_bike  casual        20M 12.8637770897833S
##  8 mar        docked_bike   casual        41M 25.3940217391305S
##  9 mar        electric_bike casual        10M 51.7925453653752S
## 10 apr        classic_bike  casual        26M 13.5373060822458S
## # ℹ 22 more rows
```

Now calculating average ride length and number of rides of users by day of week

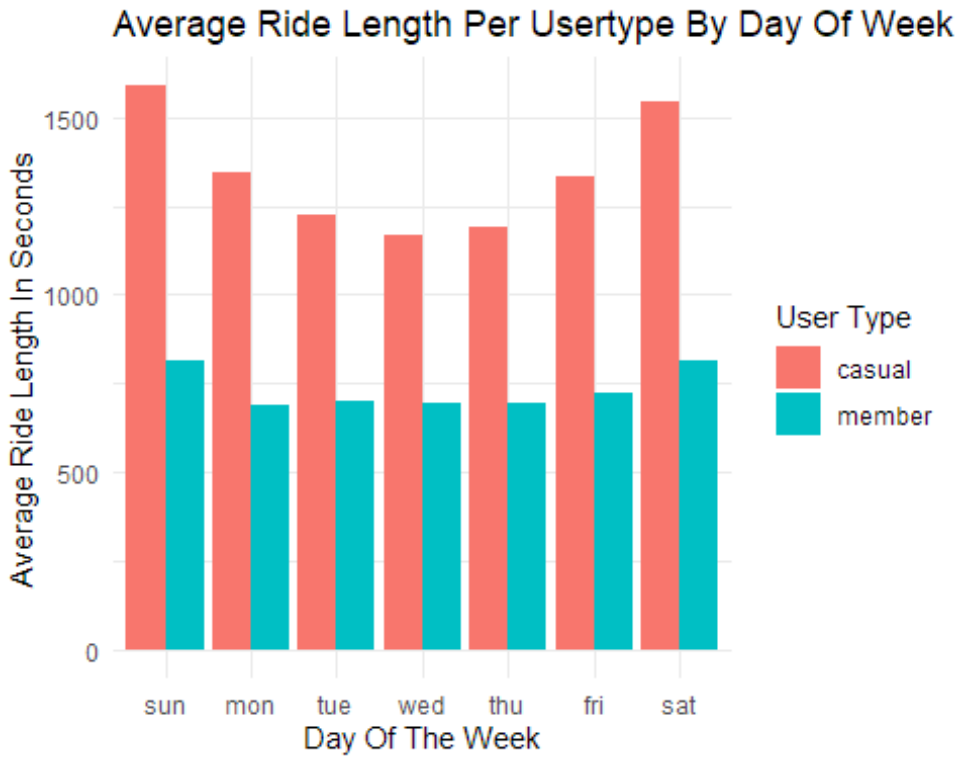
```
avg_rl_ut_dow <- bike_share_2023 %>% group_by(day_name, member_casual) %>%
summarise(avg_rl_ut_dow = seconds_to_period(mean(ride_length)), num_of_rides
= n())

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

avg_rl_ut_dow

## # A tibble: 14 × 4
## # Groups:   day_name [7]
##    day_name member_casual avg_rl_ut_dow         num_of_rides
##    <fct>    <chr>         <Period>                     <int>
##  1 sun      casual        26M 28.9277639966087S       254776
##  2 sun      member        13M 35.858243011053S        307879
##  3 mon      casual        22M 25.0861582484481S       175433
##  4 mon      member        11M 31.6054694772554S       386728
##  5 tue      casual        20M 22.1934602653566S       181567
##  6 tue      member        11M 37.8594915390167S       448884
##  7 wed      casual        19M 30.189726169541S        183106
##  8 wed      member        11M 33.6274804304419S       452744
##  9 thu      casual        19M 52.412392689841S        198956
## 10 thu      member        11M 35.6863542078557S       452689
## 11 fri      casual        22M 13.4668784666153S       227888
## 12 fri      member        12M 1.299375079581985       400531
## 13 sat      casual        25M 45.5332826120596S       310192
## 14 sat      member        13M 34.5129911795153S       350661
```

Visualizing different metrics calculated to find some insights

Changing the column avg_rl_ut_dow from time period to numeric, so that it can be plotted in ggplot.

Visualizing average ride duration per usertype across days of week

```
arlpubdow <- avg_rl_ut_dow %>% mutate(avg_ride_length_sec =
as.numeric(avg_rl_ut_dow))
ggplot(data = arlpubdow, aes(x = day_name, y = avg_ride_length_sec, fill =
member_casual)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "Average Ride Length Per Usertype By Day Of Week", x = "Day Of
The Week", y = "Average Ride Length In Seconds", fill = "User Type") +
  theme_minimal()
```



Possible reasons for longer ride durations of Casual riders

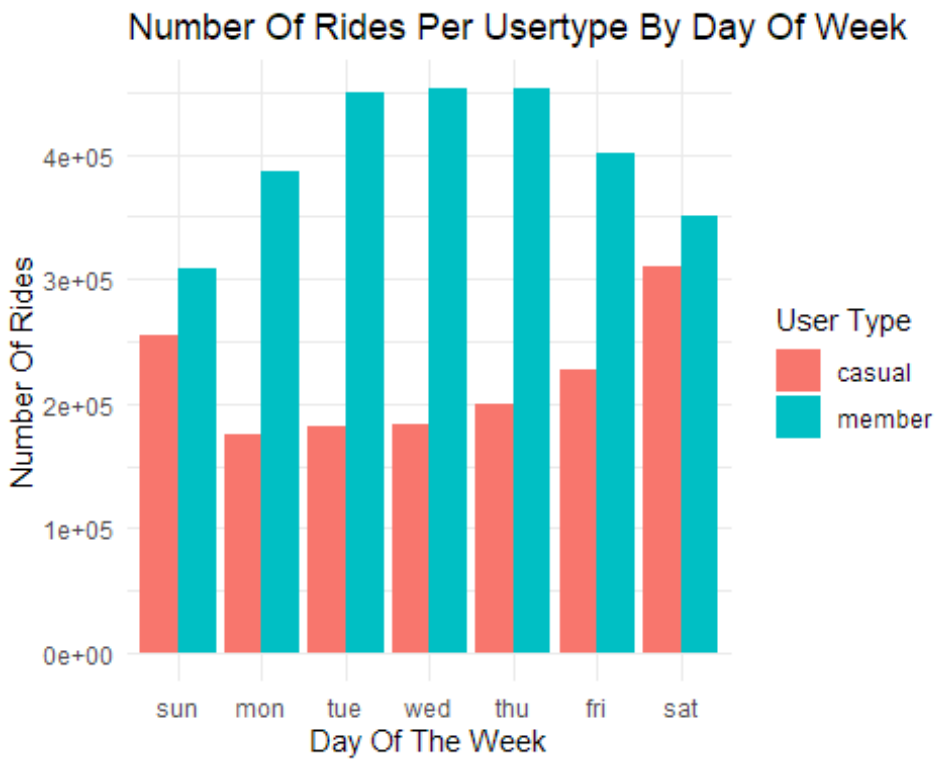
- Casual riders might be using bike sharing service primarily for leisure activities, sightseeing or tourism, which typically involve longer ride duration.
- Weekends generally offer more free time for leisure activities, therefore rise in ride duration on weekends especially for casual riders.

Targeted marketing strategies

- Emphasize leisure and recreational benefits in marketing campaigns.
- Highlight scenic routes, local attractions, and weekend events.
- Offer weekend specific promotions to encourage casual riders to use the service more frequently and potentially convert them to members.

Visualizing number of rides per usertype across days of week

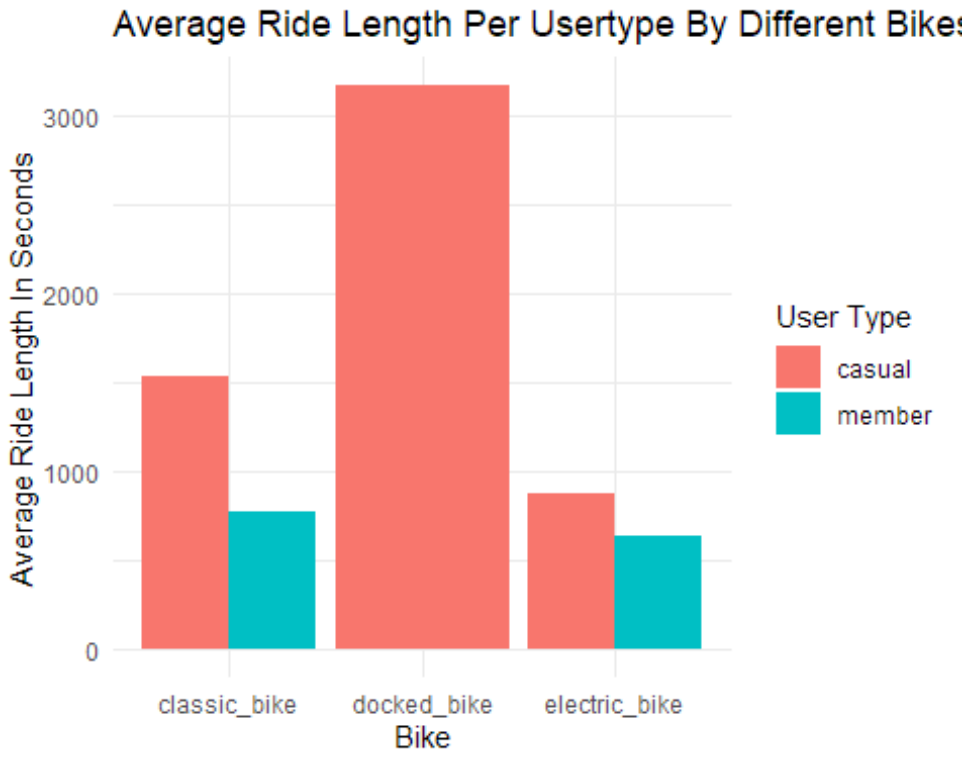
```
ggplot(data = avg_rl_ut_dow, aes(x = day_name, y = num_of_rides, fill = member_casual)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "Number Of Rides Per Usertype By Day Of Week", x = "Day Of The Week", y = "Number Of Rides", fill = "User Type") +
  theme_minimal()
```



- Develop membership plans tailored to casual riders who primarily use the service on weekends.
- Increase the distribution of bikes in parks, tourist spots, and recreational areas on weekends to meet the higher demand for casual riders.

Visualizing average ride length for different users for different bikes

```
arlpubrt <- avg_rl_ut_rt %>% mutate(avg_rt_sec = as.numeric(avg_rl_ut_rt))
ggplot(data = arlpubrt, aes(x = rideable_type, y = avg_rt_sec, fill = member_casual)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "Average Ride Length Per Usertype By Different Bikes", x = "Bike", y = "Average Ride Length In Seconds", fill = "User Type") +
  theme_minimal()
```



Possible reasons of popularity of docked bikes among casual riders

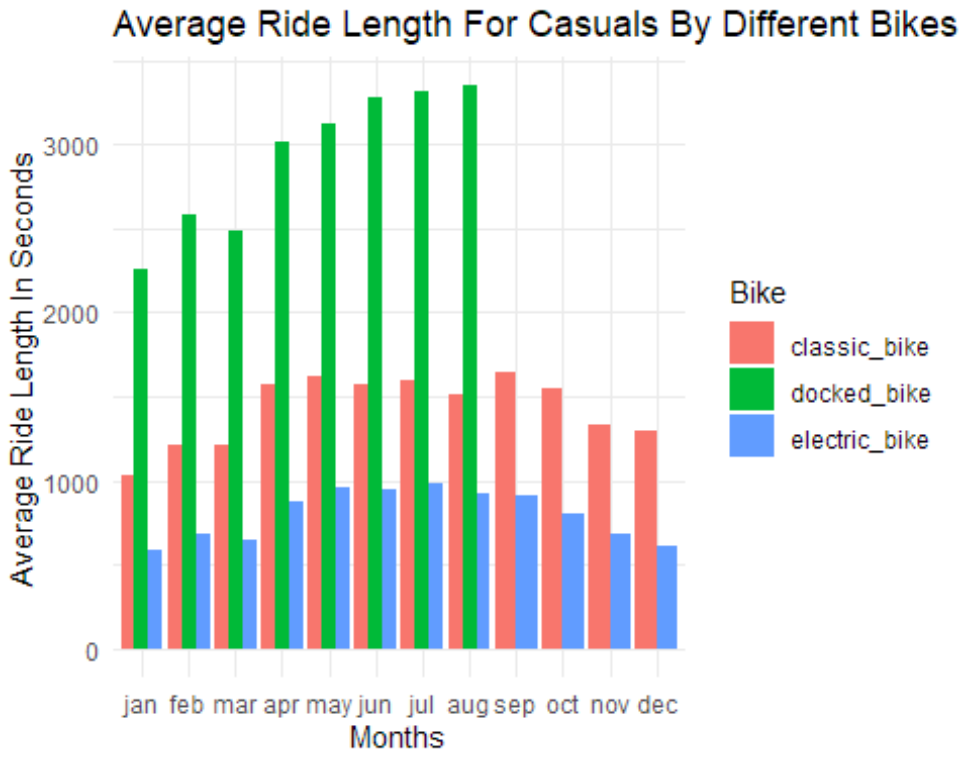
- Due to its availability, convenience, or specific routes where these bikes are docked.

Marketing and service strategies for casual riders

- Highlight the availability and benefits of docked bikes in marketing campaigns targeted at casual riders.
- Offer packages or passes specifically for docked bikes, which could include longer ride times or discounts for casual riders.
- Ensure that docked bikes are well maintained and available in popular leisure areas, especially on weekends.

Visualizing average ride length for different users for different bikes across months

```
arlpubrtm <- avg_rl_ut_rt_mon %>% mutate(avg_rt_sec =
as.numeric(avg_rl_ut_rt_mon))
ggplot(data = arlpubrtm, aes(x = month_name, y = avg_rt_sec, fill =
rideable_type)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "Average Ride Length For Casuals By Different Bikes Across
Months", x = "Months", y = "Average Ride Length In Seconds", fill = "Bike") +
  theme_minimal()
```

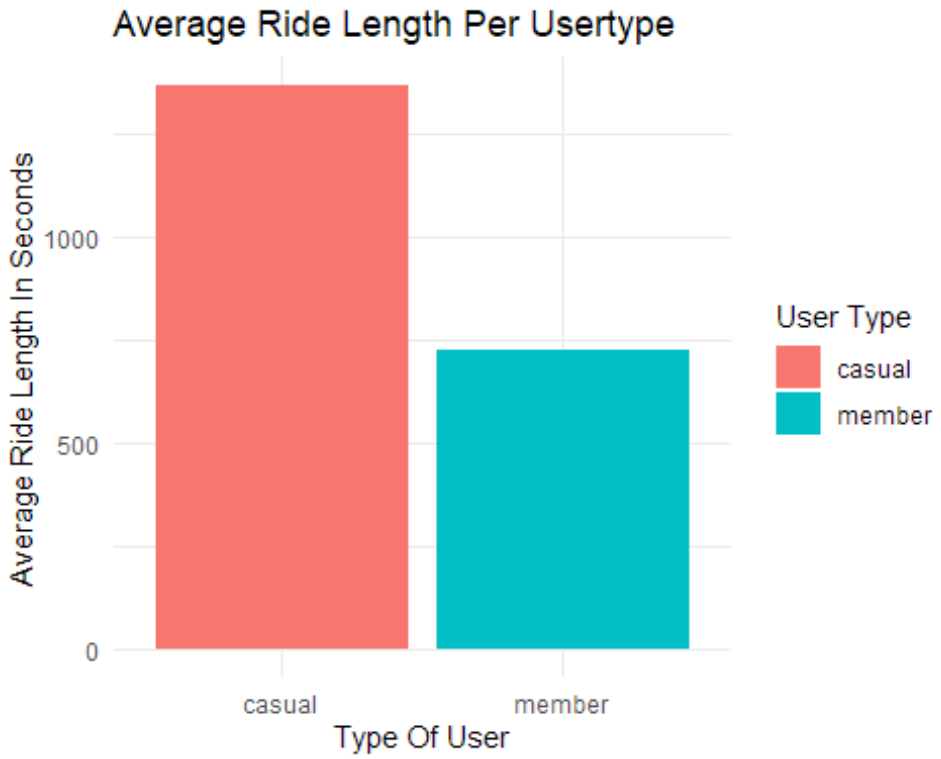


Possible reasons for disappearance of docked bike from september onwards

- Further investigation is required, if there were any operational changes made by the company around september, this could include removal or reduction of docked bikes from the fleet, changes in the areas where they were deployed, or modifications in the service offerings.
- Consulting internal reports from company regarding changes in bike availability.
- Ensure that data collection methods remained consistent throughout the year, any changes in how data was recorded or processed could affect the presence of certain bike types in the data.
- Examine if casual riders' preferences or behavior changed around september, leading to drop in the use of docked bikes. Although this is less likely, it's worth considering.

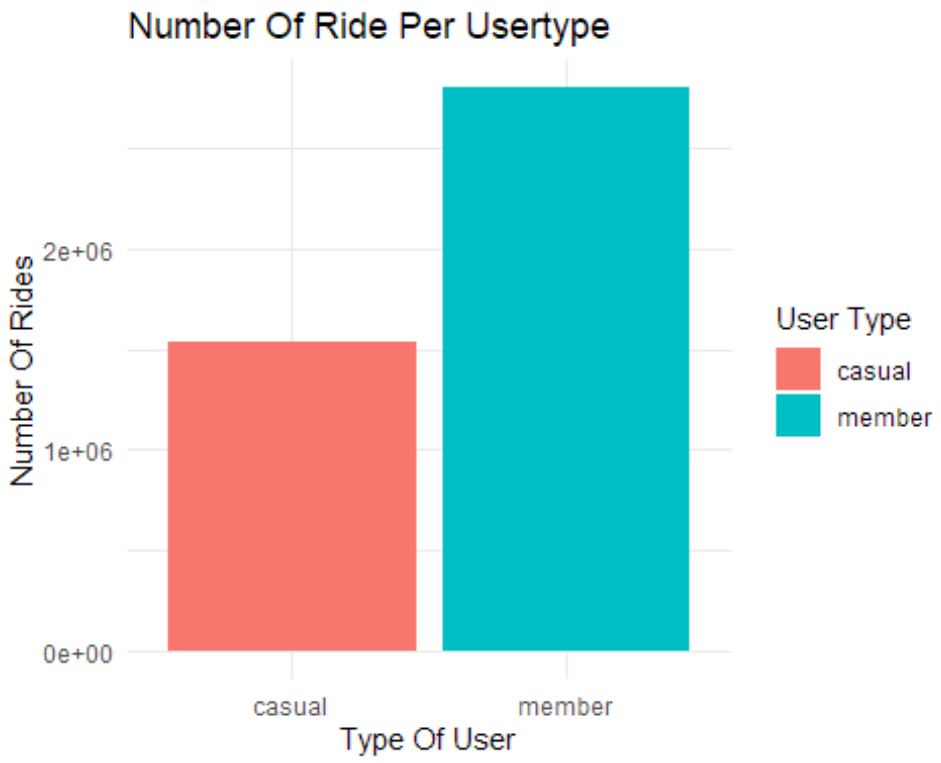
Visualizing average ride length per usertype

```
arlput <- avg_ride_length_per_usertype %>% mutate(avg_ut_sec =
as.numeric(avg_ride_length_per_usertype))
ggplot(data = arlput, aes(x = member_casual, y = avg_ut_sec, fill =
member_casual)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Ride Length Per Usertype", x = "Type Of User", y =
"Average Ride Length In Seconds", fill = "User Type") +
  theme_minimal()
```



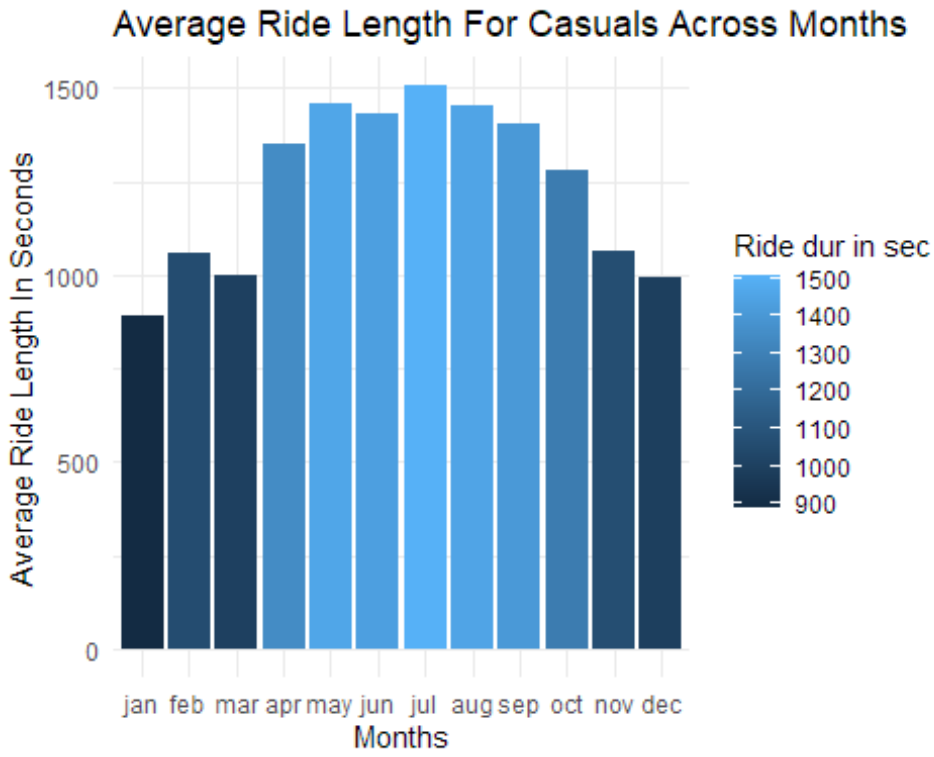
Visualizing number of rides per usertype

```
ggplot(data = arlput, aes(x = member_casual, y = no_ride, fill =
member_casual)) +
  geom_bar(stat = "identity") +
  labs(title = "Number Of Ride Per Usertype", x = "Type Of User", y = "Number
Of Rides", fill = "User Type") +
  theme_minimal()
```



Visualizing average ride duration for casual riders across months

```
arlputm <- avg_rl_cas_mon %>% mutate(avg_cas_mon_sec =
as.numeric(avg_rl_ut_mon))
ggplot(data = arlputm, aes(x = month_name, y = avg_cas_mon_sec, fill =
avg_cas_mon_sec)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Ride Length For Casuals Across Months", x = "Months",
y = "Average Ride Length In Seconds", fill = "Ride dur in sec") +
  theme_minimal()
```



Seasonal influence on usage(from april to october)

- The period from april to october generally corresponds to warmer weather and longer daylight hours, making it more conducive to outdoor activities and longer rides.
- These months might coincide with peak tourism and recreational activities, leading to mor leisurely and longer rides by casual riders.

Operational adjustments

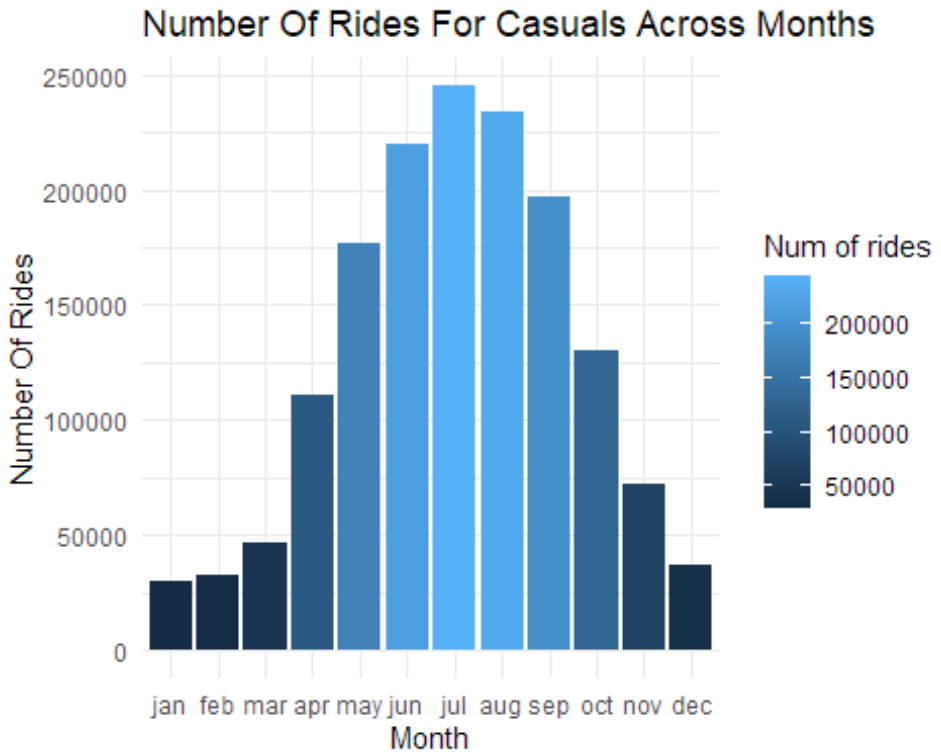
- Ensure that there are enough bikes available, especially docked bikes, during the peak months to meet the higher demand for longer rides.
- Schedule maintainance activities in the off-peak months(november to march) to ensure the fleet is in optimal condition during the peak season.

Recommendations

- Offer promotions and discounts during the peak months(april to october) to attract more casual riders.
- Introduce seasonal passes or packages specifically for peak months, encouraging casual riders to purchase longer term access during the high demand period.
- Provide enhance services such as guided tours, themed rides, or partnerships with local attractions during the peak months to capitalize on the increased interest.

Visualizing number of rides or frequency of rides for casual riders across months

```
ggplot(data = arlputm, aes(x = month_name, y = no_ride, fill = no_ride)) +
  geom_bar(stat = "identity") +
  labs(title = "Number Of Rides For Casuals Across Months", x = "Month", y =
"Number Of Rides", fill = "Num of rides") +
  theme_minimal()
```



- Consider scaling services e.g., additional bike stations, more docking points during the high demand season and potentially reducing them during the off-peak months to optimize resource allocation.