Cyclistic_BikeShare

Francisco E. Navarro

2024-08-13

Contents

Analyzing how do annual members and casual riders use Cyclistic bikes differently

1. Introduction

Welcome to the Cyclistic bike-share analysis case study! In this report, we explore how Cyclistic, a bike-share company in Chicago, can maximize the number of annual memberships by understanding the differences in usage between casual riders and annual members. This analysis aims to provide actionable insights to design an effective marketing strategy to convert casual riders into annual members.

2. Data Overview

Data Source: The data-set used for this analysis comprises monthly data files from July 2023 to June 2024.

Tools and Packages Used:

R: For data cleaning and analysis

Tableau: For visualization and dashboard creation

R Packages Installed:

options(repos = c(CRAN = "https://cran.rstudio.com/"))

install.packages(c('tidyverse', 'skimr', 'janitor', 'here')')

Note 2: Loading necessary packages

```
# Load libraries
library(ggplot2)    # for data visualization
library(tidyr)      # to tidy data
library(dplyr)      # for data manipulation
library(skimr)      # for summarizing data
library(janitor)    # for cleaning data
library(here)       # to manage file paths
library(lubridate)  # for working with dates and times
```

3. Data Preparation

3.1. Loading Data

Data was loaded from CSV files for each month:

Note 3: Loading monthly data from 07-2023 to 06-2024

```
# Loading the data from different files
c202307 <- read.csv('202307.csv')
c202308 <- read.csv('202308.csv')
c202309 <- read.csv('202309.csv')
c202310 <- read.csv('202310.csv')
c202311 <- read.csv('202311.csv')
c202312 <- read.csv('202312.csv')
c202401 <- read.csv('202401.csv')
c202402 <- read.csv('202402.csv')
c202403 <- read.csv('202403.csv')
c202404 <- read.csv('202404.csv')
c202405 <- read.csv('202405.csv')
c202406 <- read.csv('202406.csv')
```

3.2. Merging Data

All monthly datasets were combined into a single data frame:

```
# Merging data into one frame
cyclistic_bike_share <- bind_rows(c202307, c202308, c202309, c202310, c202311,
                                  c202312, c202401, c202402, c202403, c202404,
                                  c202405, c202406)
```

4. Data Cleaning

4.1. Initial Inspection

The initial data structure was checked for any issues:

```r
# Check the data set with one or more of the following functions
#head(cyclistic_bike_share)
skim_without_charts(cyclistic_bike_share)
```

Table 1: Data summary

Name	cyclistic_bike_share
Number of rows	5734381
Number of columns	13
_____	
Column type frequency:	
character	9
numeric	4
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ride_id	0	1	16	16	0	5734170	0
rideable_type	0	1	11	13	0	3	0
started_at	0	1	19	23	0	4969058	0
ended_at	0	1	19	23	0	4978308	0
start_station_name	0	1	0	64	933003	1678	0
start_station_id	0	1	0	14	933003	1641	0
end_station_name	0	1	0	64	980556	1695	0
end_station_id	0	1	0	36	980556	1653	0
member_casual	0	1	6	6	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
start_lat	0	1	41.90	0.05	41.63	41.88	41.90	41.93	42.07
start_lng	0	1	-87.65	0.03	-87.94	-87.66	-87.64	-87.63	-87.46
end_lat	7919	1	41.90	0.05	0.00	41.88	41.90	41.93	42.19
end_lng	7919	1	-87.65	0.05	-88.12	-87.66	-87.64	-87.63	0.00

```r
#View(cyclistic_bike_share)
#summary(cyclistic_bike_share)
```

4.2. Identified Issues

After confirming the structure of the data, several issues were detected:

- **Date and Time Format:** The started_at and ended_at columns are in character format and need to be converted to date-time objects for accurate calculations.
- **Column Naming:** The column names started_at and ended_at are not sufficiently descriptive. They should be renamed to improve clarity.

- **Trip Duration Calculation:** A new column will be created to calculate the duration of each trip. Negative values will be removed to ensure accuracy.
- **Defining Days and Months:** Columns for days of the week and months will be added for better temporal analysis.
- **Missing Station Information:** Some station names and IDs are missing. If either the name or ID is provided, the missing counterpart can be tracked. However, if both are missing, the data will be filtered out because the coordinates do not match the station in several cases, rendering the data unreliable.
- **Handling Incomplete Data:** Incomplete data will be removed to avoid errors in analysis.
- **Removing Duplicates:** Duplicate records will be removed using the distinct() function to ensure each entry is unique.

To ensure that column names are unique and consistent, the clean_names() function from the janitor package will be used.

A new data frame, all_trips, will be created by filtering and cleaning the data, renaming columns, and adding new columns for trip duration in minutes, days of the week and months.

4.3. Cleaning Process

Data was cleaned and transformed:

```
all_trips <- cyclistic_bike_share %>%

  # Convert date-time columns to POSIXct
  mutate(
    started_at = ymd_hms(started_at),
    ended_at = ymd_hms(ended_at)
  ) %>%

  # Rename columns
  rename(
    initial_time = started_at,
    final_time = ended_at,
    client = member_casual
  ) %>%

  #determine the date of the week
  mutate(
    date1 = as.Date(initial_time),
    day_of_week = weekdays(date1, abbreviate = FALSE),
    # Convert day_of_week to a factor with natural order
    day_of_week = factor(day_of_week, levels = c("Monday", "Tuesday", "Wednesday",
                                                 "Thursday", "Friday", "Saturday",
                                                 "Sunday"))
    ) %>%

  #determine the month
  mutate(
    date2 = as.Date(initial_time),
    month = format(date2, "%B"),
```

```
    # Convert month to a factor with natural order
    month = factor(month, levels = c("January", "February", "March", "April", "May", "June",
                                     "July", "August", "September",
                                     "October", "November", "December"))
  ) %>%

  # Calculate trip time in minutes
  mutate(
    trip_time_minutes = as.numeric(difftime(final_time,
                                            initial_time,
                                            units = 'mins'))
  ) %>%

  # Filter out negative trip times
  filter(trip_time_minutes >= 0) %>%


  # Filter rows with blank fields for station's names or id
  filter(
    (start_station_name != "" |   # filter the data to keep rows where at least
       start_station_id != "") &   # the start station name or id exists and where
      (end_station_name != "" |   # the end station name or id exists.
         end_station_id != "")
  ) %>%

  # Drop rows with NA values
  drop_na() %>%

  # Drop duplicates
  distinct() %>%

  # Select all columns except hours, minutes, seconds and trip_time_seconds
  select(-date1, -date2) %>%

  # Clean column names
  clean_names()
```

Verifying the Cleaned Data:

Now that our data-set has been cleaned, we need to verify that everything is correct.

```
# Check the updated data set with one or more of the following functions
#head(all_trips)
skim_without_charts(all_trips)
```

Table 4: Data summary

Name	all_trips
Number of rows	4274279
Number of columns	16
Column type frequency:	
character	7

factor	2
numeric	5
POSIXct	2
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ride_id	0	1	16	16	0	4274158	0
rideable_type	0	1	11	13	0	3	0
start_station_name	0	1	10	64	0	1627	0
start_station_id	0	1	3	14	0	1600	0
end_station_name	0	1	10	64	0	1647	0
end_station_id	0	1	3	36	0	1612	0
client	0	1	6	6	0	2	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
day_of_week	0	1	FALSE	7	Sat: 661450, Wed: 626180, Thu: 623743, Tue: 611125
month	0	1	FALSE	12	Aug: 584912, Jul: 573955, Sep: 506632, Jun: 494205

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
start_lat	0	1	41.90	0.04	41.65	41.88	41.90	41.93	42.06
start_lng	0	1	-87.64	0.03	-87.84	-87.66	-87.64	-87.63	-87.53
end_lat	0	1	41.90	0.05	0.00	41.88	41.90	41.93	42.06
end_lng	0	1	-87.64	0.05	-87.84	-87.66	-87.64	-87.63	0.00
trip_time_minutes	0	1	16.51	36.52	0.00	5.78	10.07	18.03	6891.22

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
initial_time	0	1	2023-07-01 00:00:00	2024-06-30 23:54:52	2023-11-07 06:25:37	3817703
final_time	0	1	2023-07-01 00:03:30	2024-06-30 23:59:57	2023-11-07 06:34:42	3826548

```
#View(all_trips)
#summary(all_trips)
```

5. Descriptive Analysis

5.1. Trip Duration Analysis

Summary statistics of trip durations:

```
# Descriptive analysis on trip_time_minutes (all data in minutes)
summary(all_trips$trip_time_minute)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    0.000    5.783   10.067   16.511   18.026 6891.217
```

5.2. Comparison by Client Type

Comparison of trip duration between members and casual users:

```
# Compare members and casual users
aggregate(all_trips$trip_time_minutes ~ all_trips$client, FUN = mean)
```

```
##   all_trips$client all_trips$trip_time_minutes
## 1           casual                    23.90254
## 2           member                    12.49894
```

```
aggregate(all_trips$trip_time_minutes ~ all_trips$client, FUN = median)
```

```
##   all_trips$client all_trips$trip_time_minutes
## 1           casual                   13.262017
## 2           member                    8.816667
```

```
aggregate(all_trips$trip_time_minutes ~ all_trips$client, FUN = max)
```

```
##   all_trips$client all_trips$trip_time_minutes
## 1           casual                    6891.217
## 2           member                    1497.650
```

```
aggregate(all_trips$trip_time_minutes ~ all_trips$client, FUN = min)
```

```
##   all_trips$client all_trips$trip_time_minutes
## 1           casual                           0
## 2           member                           0
```

5.3. Ridership Analysis by Day of Week

Average trip duration and number of rides by client type and weekday:

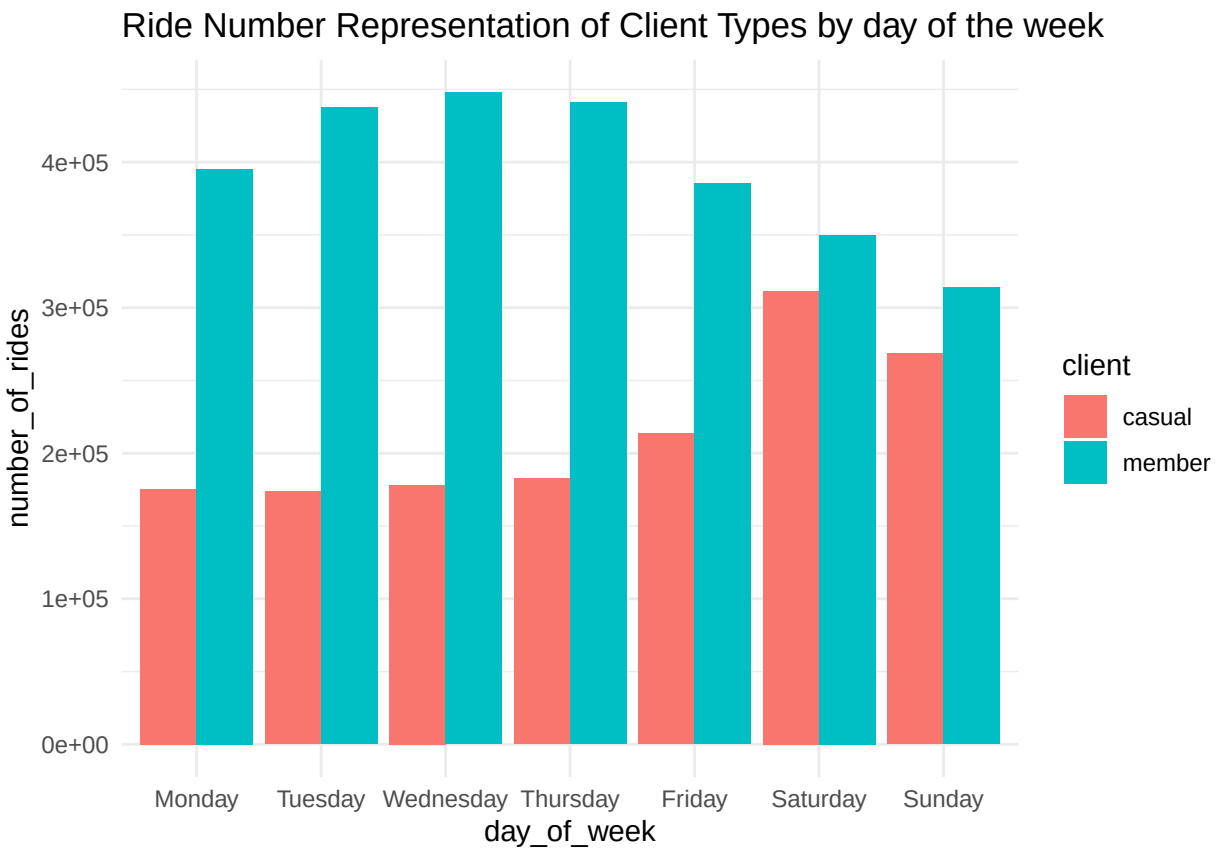
```
all_trips %>%
   group_by(client, day_of_week) %>%  #groups by usertype and weekday
  summarise(number_of_rides = n()                              #calculates the number of rides and average
            ,average_duration = mean(trip_time_minutes)) %>%        # calculates the average duration
  arrange(client, day_of_week)                                 # sorts
```

```
## # A tibble: 14 x 4
## # Groups:   client [2]
##    client day_of_week number_of_rides average_duration
##    <chr>  <fct>                 <int>            <dbl>
##  1 casual Monday               175575             23.4
##  2 casual Tuesday              173715             21.2
##  3 casual Wednesday            178204             20.9
##  4 casual Thursday             182752             20.4
##  5 casual Friday               213538             23.0
##  6 casual Saturday             311492             26.8
##  7 casual Sunday               268381             27.6
##  8 member Monday               395103             12.0
##  9 member Tuesday              437410             12.1
## 10 member Wednesday            447976             12.1
## 11 member Thursday             440991             11.8
## 12 member Friday               385408             12.2
## 13 member Saturday             349958             13.9
## 14 member Sunday               313776             14.1
```

5.4. Visualization

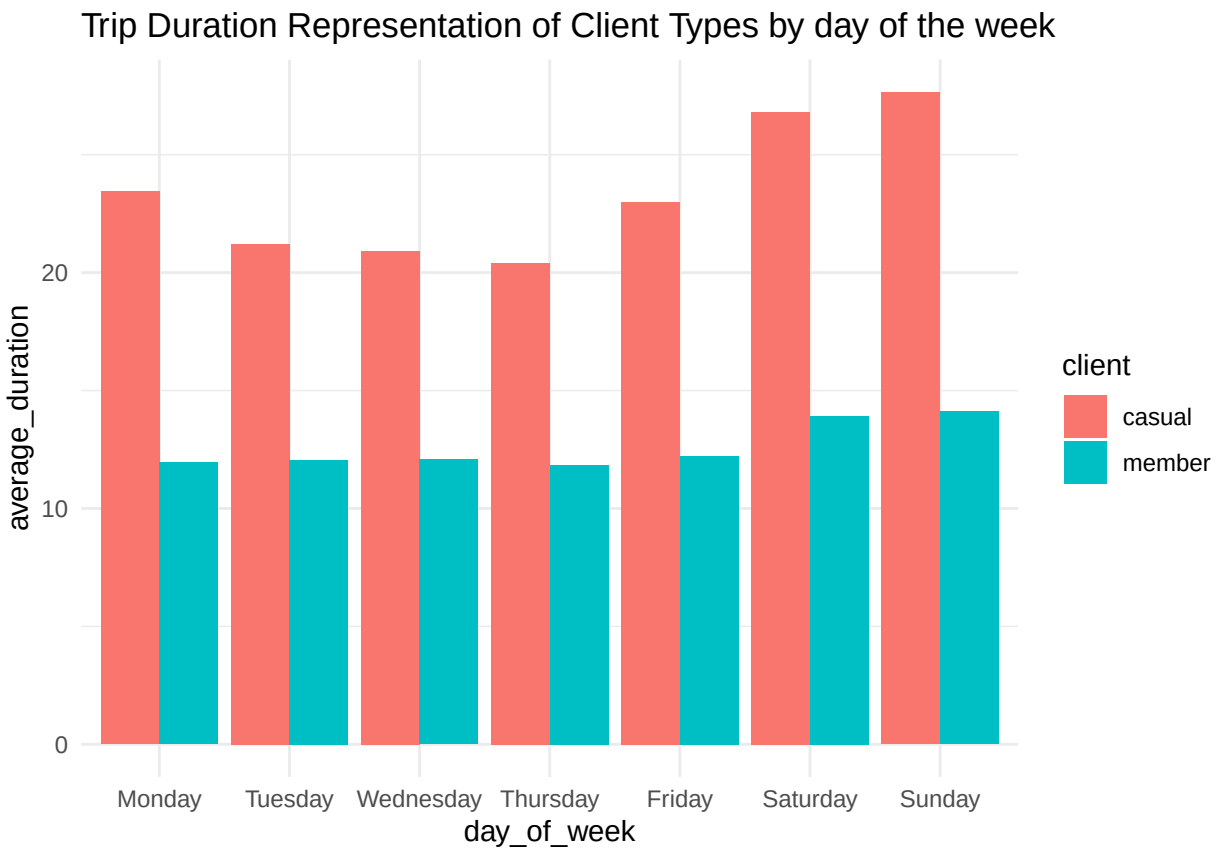
- **Number of Rides by Day of Week:**

```
all_trips %>%
  group_by(client, day_of_week) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(trip_time_minutes)) %>%
  arrange(client, day_of_week)  %>%
  ggplot(aes(x = day_of_week, y = number_of_rides, fill = client)) +
  geom_col(position = "dodge")+
  theme_minimal() +
  labs(title = "Ride Number Representation of Client Types by day of the week")
```



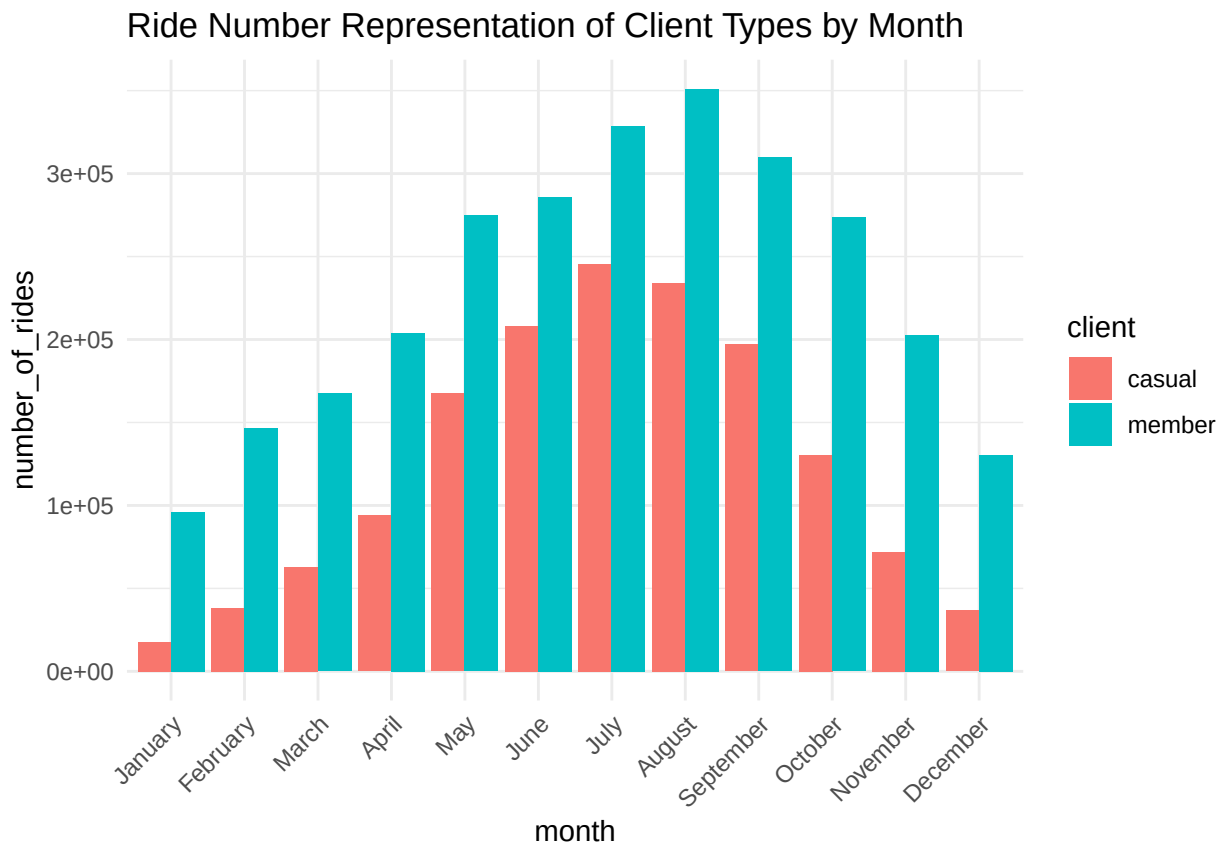
- **Average Duration by Day of Week:**

```
all_trips %>%
  group_by(client, day_of_week) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(trip_time_minutes)) %>%
  arrange(client, day_of_week)  %>%
  ggplot(aes(x = day_of_week, y = average_duration, fill = client)) +
  geom_col(position = "dodge") +
  theme_minimal() +
  labs(title = "Trip Duration Representation of Client Types by day of the week")
```



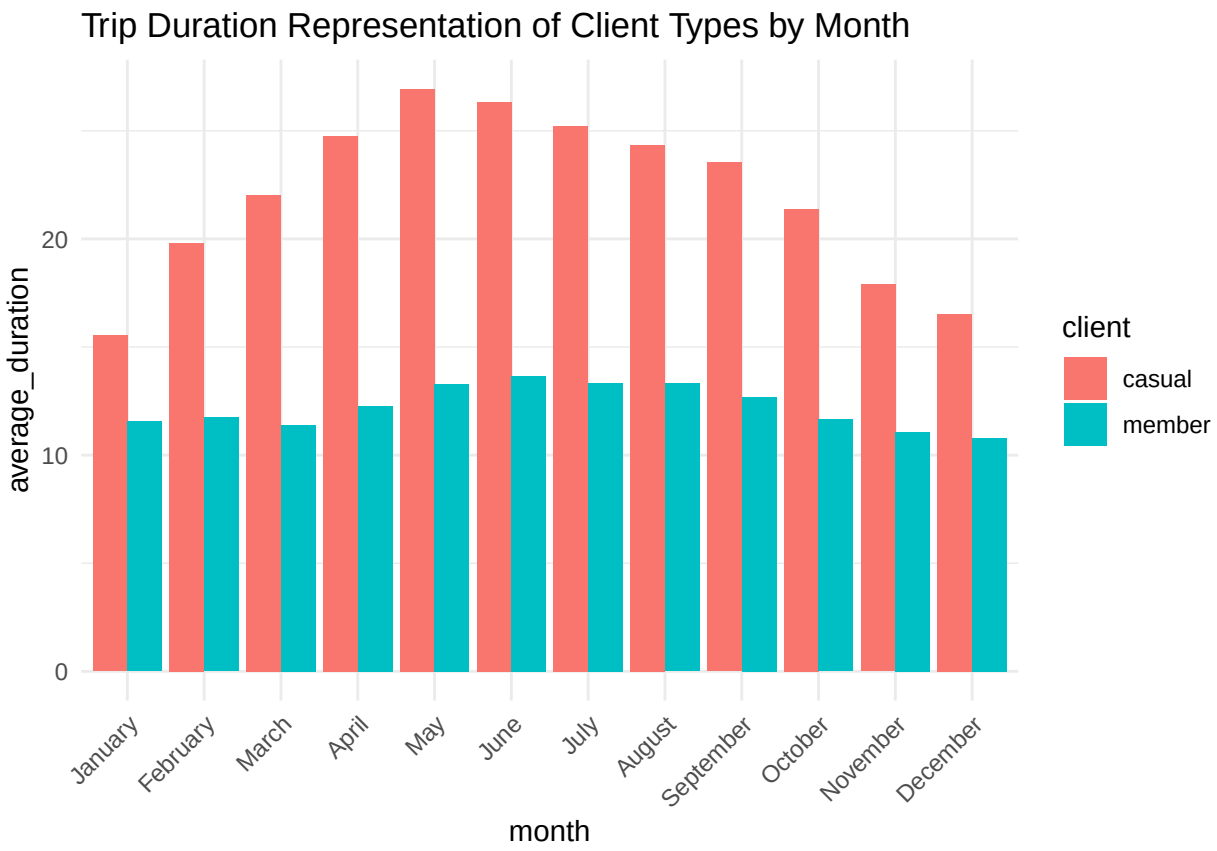
- **Number of Rides by Month:**

```
all_trips %>%
  group_by(client, month) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(trip_time_minutes)) %>%
  arrange(client, month)  %>%
  ggplot(aes(x = month, y = number_of_rides, fill = client)) +
  geom_col(position = "dodge") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+
  labs(title = "Ride Number Representation of Client Types by Month")
```



- **Average Duration by Month:**

```
all_trips %>%
  group_by(client, month) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(trip_time_minutes)) %>%
  arrange(client, month)  %>%
  ggplot(aes(x = month, y = average_duration, fill = client)) +
  geom_col(position = "dodge") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+
  labs(title = "Trip Duration Representation of Client Types by Month")
```



6. Processing and Summarizing Data

6.1. Weekly and Monthly Summaries

Summarizing data for weekly and monthly insights:

Summarizing data for weekly insights

```
week_summary <- all_trips %>%
  # Convert initial_time to Date type
  mutate(date1 = as.Date(initial_time),
         day_of_week = weekdays(date1, abbreviate = FALSE),
         # Convert day_of_week to a factor with natural order
         day_of_week = factor(day_of_week, levels = c("Monday", "Tuesday", "Wednesday",
                                                      "Thursday", "Friday", "Saturday",
                                                      "Sunday"))
  ) %>%
  # Group by day_of_week and client_type
  group_by(day_of_week, client) %>%
  summarise(total_cases = n(), .groups = 'drop') %>%
  # Group by month and year to get total cases per month
  group_by(day_of_week) %>%
  mutate(total_cases_day = sum(total_cases)) %>%
  # Calculate percentage for each client type
  mutate(percentage = (total_cases / total_cases_day) * 100) %>%
  ungroup()
```

Summarizing data for monthly insights

```
# Monthly summary
monthly_summary <- all_trips %>%
  # Convert initial_time to Date type
  mutate(date2 = as.Date(initial_time),
         month2 = format(date2, "%B"),
         year2 = year(date2),
         # Convert month2 to a factor with natural order
         month2 = factor(month2, levels = c("January", "February", "March", "April",
                                            "May", "June", "July", "August", "September",
                                           "October", "November", "December"))
         ) %>%
  # Group by month, year, and client_type
  group_by(year2, month2, client) %>%
  summarise(total_cases = n(), .groups = 'drop') %>%
  # Group by month and year to get total cases per month
  group_by(year2, month2) %>%
  mutate(total_cases_month = sum(total_cases)) %>%
  # Calculate percentage for each client type
  mutate(percentage = (total_cases / total_cases_month) * 100) %>%
  ungroup()
```

Summarizing data for weekly and monthly insights

```
# Combined weekly and monthly summary

monthly_week_summary <- all_trips %>%
  # Convert initial_time to Date type
  mutate(date2 = as.Date(initial_time, format = "%Y-%m-%d"),
         day_of_week = weekdays(date2, abbreviate = FALSE),
         # Convert day_of_week to a factor with natural order
         day_of_week = factor(day_of_week, levels = c("Monday", "Tuesday",
                                                      "Wednesday","Thursday",
                                                      "Friday", "Saturday",
                                                      "Sunday")),
         month2 = format(date2, "%B"),
         year2 = year(date2),
         # Convert month2 to a factor with natural order
         month2 = factor(month2, levels = c("January", "February", "March", "April",
                                            "May", "June", "July", "August", "September",
                                            "October", "November", "December"))
  ) %>%
  # Group by year, month, client, and day_of_week
  group_by(year2, month2, client, day_of_week) %>%
  summarise(total_cases = n(), .groups = 'drop') %>%
  # Group by year and month to get total cases per month
  group_by(year2, month2) %>%
  mutate(total_cases_month = sum(total_cases)) %>%
  # Calculate percentage for each client type
  mutate(percentage = (total_cases / total_cases_month) * 100) %>%
  ungroup()
```

Check the Summaries Before Continuing Weekly Summary To review the weekly summary, use the following code:

```
View(week_summary)
aggregate(week_summary$percentage ~ week_summary$client +
            week_summary$ day_of_week, FUN = mean)
```

```
##    week_summary$client week_summary$day_of_week week_summary$percentage
## 1               casual                   Monday                30.76604
## 2               member                   Monday                69.23396
## 3               casual                  Tuesday                28.42544
## 4               member                  Tuesday                71.57456
## 5               casual                Wednesday                28.45891
## 6               member                Wednesday                71.54109
## 7               casual                 Thursday                29.29925
## 8               member                 Thursday                70.70075
## 9               casual                   Friday                35.65230
## 10              member                   Friday                64.34770
## 11              casual                 Saturday                47.09230
## 12              member                 Saturday                52.90770
## 13              casual                   Sunday                46.10114
## 14              member                   Sunday                53.89886
```

Monthly Summary To review the monthly summary, use the following code:

```
View(monthly_summary)
aggregate(monthly_summary$percentage ~ monthly_summary$client +
            monthly_summary$month2, FUN = mean)
```

```
##    monthly_summary$client monthly_summary$month2 monthly_summary$percentage
## 1                  casual                January                  15.56407
## 2                  member                January                  84.43593
## 3                  casual               February                  20.66203
## 4                  member               February                  79.33797
## 5                  casual                  March                  27.27956
## 6                  member                  March                  72.72044
## 7                  casual                  April                  31.54641
## 8                  member                  April                  68.45359
## 9                  casual                    May                  37.88281
## 10                 member                    May                  62.11719
## 11                 casual                   June                  42.14810
## 12                 member                   June                  57.85190
## 13                 casual                   July                  42.73715
## 14                 member                   July                  57.26285
## 15                 casual                 August                  39.98054
## 16                 member                 August                  60.01946
## 17                 casual              September                  38.87674
## 18                 member              September                  61.12326
## 19                 casual                October                  32.26970
## 20                 member                October                  67.73030
## 21                 casual               November                  26.23214
## 22                 member               November                  73.76786
## 23                 casual               December                  21.94940
## 24                 member               December                  78.05060
```

Create Weekly, Monthly, and Weekly-Monthly Trip Counts for Further Analysis

For further analysis, create trip counts for weekly, monthly, and weekly-monthly periods using the following code:

```
weeklycounts <- aggregate(all_trips$trip_time_minutes ~
                            all_trips$client +
                            all_trips$day_of_week,
                          FUN = mean)
monthlycounts <- aggregate(all_trips$trip_time_minutes ~
                             all_trips$client +
                             all_trips$month,
                           FUN = mean)
monthly_week_counts <- aggregate(all_trips$trip_time_minutes ~
                                   all_trips$client +
                                   all_trips$month +
                                   all_trips$day_of_week,
                                 FUN = mean)
```

7. Export Summary Files for Further Analysis

Create a csv file that we will visualize in Excel, Tableau, or my presentation software

```
#Define the file paths 1, 2, 3...
#file_path <- "filepath/filename.csv"

# Write the data frames you need to a CSV file
#write.csv(monthly_week_summary, file = file_path, row.names = FALSE)
#write.csv(cyclistic_bike_share, file = file_path2, row.names = FALSE)
#etc
```

8. Recommendations

Based on the analysis, we recommend the following actions:

Targeted Promotions: Increase marketing efforts on weekends and the most popular days for casual riders to encourage them to sign up for annual memberships. **Time-Based Incentives:** Offer incentives for rides during off-peak hours to attract more members. **Monthly Campaigns:** Launch monthly campaigns highlighting the benefits of annual memberships, especially during months with lower membership growth.

9. Conclusion

This analysis provides valuable insights into the usage patterns of Cyclistic's bike-share system. By understanding these patterns, Cyclistic can design targeted marketing strategies to increase annual memberships and enhance overall user satisfaction.

10. Data Usage Note

Data Licence Link (here)

Data Source: The data used in this dashboard is sourced from the Divvy bike-sharing service, operated by Lyft Bikes and Scooters, LLC in partnership with the City of Chicago. **Data-sets** here

Purpose: This report was created for educational and portfolio demonstration purposes only. It is not intended for commercial use.

Data Usage: The data has been integrated into this analysis to showcase data visualization and analytical skills. The data itself is not sold or distributed as a standalone product.

Affiliation: This work is independent and not affiliated with, endorsed by, or sponsored by Bikeshare or the City of Chicago.

Trademark Notice: No logos or trademarks of Divvy or Bikeshare are used in this dashboard. All trademarks and logos are the property of their respective owners.

Data Access: The data was accessed through authorized channels, including the provided API and data download options.