Google Cyclists Capstone

About this project

Introduction

In this case study, I'll be performing many real-world tasks that a data analyst usually does in their day-to-day job. I'll be working with a fictional company named Cyclist and answer key business questions, in order to do that I'll be following the six-step data analysis process: ask, prepare, process, analyse, share and act.

Scenario

I'm a junior data analyst working in the marketing analyst team at Cyclists, a bike-share company in Chicago. The director of marketing believes the company's future success depends on maximizing the number of annual memberships. That being the case I need to understand how casual riders and annual members use Cyclists bikes differently. From these insights, I will design a new marketing strategy to convert casual riders into annual members. But first, Cyclists executives must approve my recommendations, so they must be backed up with compelling data insights and professional data visualizations, that I'll do my best to provide below.

What do we know?

Cyclists has a fleet of 5,824 bicycles that are retracked and locked into a network of 692 stations across Chicago. The bikes can be unlocked from one station and returned to any other station in the system anytime. Until now, Cyclists marketing strategy relied on building general awareness and appealing to broad consumer segments. One approach that helped make these things possible was the flexibility of its pricing plans: single-ride passes, full-day passes, and annual memberships. Customers who purchase single-ride or full-day passes are referred to as casual riders. Customers who purchase annual memberships are Cyclists members.

Cyclists finance analysts have concluded that annual members are much more profitable than casual riders. Although the pricing flexibility helps Cyclists attract more customers, the director of marketing Lily Moreno believes that maximizing the number of annual members will be key to future growth. Rather than creating a marketing campaign that targets all-new customers, Moreno believes there is a very good chance to convert casual riders into members. She notes that casual riders are already aware of the Cyclists program and have chosen Cyclists for their mobility needs.

That's all we know and that brings us to the **ask** phase.

Ask

Our goal is to design marketing strategies aimed at converting casual riders into annual members, the next three questions will be a great guide to get there:

- **How do annual members and casual riders use Cyclists bikes differently?**
- **What would make casual riders buy a membership?**
- **How can Cyclists use digital media to influence casual riders to become members?**

To answer these questions, our marketing director is interested in analysing the Cyclists historical bike trip data to identify trends.

Prepare

The data for this project consisted of 12 months of cyclist data from January to December 2023. Each month's data was downloaded as a separate zipped CSV file. After unzipping the files, I converted them to .xls format for easier manipulation in Excel.

Process

Upon initial inspection of the data, I noticed that four columns had a significant number of missing values.

These columns were start_station_name, start_station_id, end_station_name, end_station_id , also there were another four columns start_lat, start_lng, end_lat, and end_lng. As these columns were not relevant to my analysis, I decided to remove them. The remaining columns in the dataset were:

1. rider_id
2. rideable_type
3. started_at
4. ended_at
5. member_casual

Transform

Microsoft Excel

Several transformations were performed on the data using Excel functions to facilitate further analysis. These transformations included calculating ride length, extracting date and day, month, and time of day, and categorizing the data into seasons and weekdays/weekends.

1. **Ride Length**:

    - Function: =(D2-C2)

- Calculation: This formula subtracts the start time (C2) from the end time (D2). The result is in days, so it's multiplied by 1440 to convert it to minutes).
- Column: F (ride_length)
- Explanation: This gives the length of each ride in minutes.

2. **Date and Day**:

   - Function: =weekday(G2)
   - Calculation: This function extracts the day of the week from the date (G2), with Sunday as 1 and Saturday as 7.
   - Column: H (day_number)
   - Explanation: This gives the day of the week for each ride.

3. **Day Name**:

   - Function: =text(H2, "dddd")
   - Calculation: This function converts the day number (H2) to the day name.
   - Column: I (day_name)
   - Explanation: This gives the name of the day for each ride.

4. **Month**:

   - Function: =month(C2)
   - Calculation: This function extracts the month from the start time (C2).
   - Column: J (month_number)
   - Explanation: This gives the month of each ride.

5. **Month Name**:

   - Function: =text(J2, "mmmm")
   - Calculation: This function converts the month number (J2) to the month name.
   - Column: K (month_name)
   - Explanation: This gives the name of the month for each ride.

6. **Season**:

   - Function: =IFS(AND(J2>=3, J2<=5), "Spring", AND(J2>=6, J2<=8), "Summer", AND(J2>=9, J2<=11), "Fall", J2=12, "Winter", AND(J2>=1, J2<=2), "Winter")
   - Calculation: This function categorizes the month number (J2) into seasons.
   - Column: L (season)
   - Explanation: This gives the season of each ride.

7. **Time of Day**:

   - Function: =hour(C2)
   - Calculation: This function extracts the hour from the start time (C2).
   - Column: M (hour_of_day)

- Explanation: This gives the hour of the day for each ride.

8. **Time of Day Category**:

    - Function: =IFS(AND(M3>=0,M3<=5),"Night",AND(M3>=6,M3<=11),"Morning",AND(M3>=12,M3<=16),"Afternoon",AND(M3>=17,M3<=21),"Evening",OR(M3=22,M3=23),"Night")
    - Calculation: This function categorizes the hour of the day (M2) into time of day categories.
    - Column: N (time_of_day)
    - Explanation: This gives the time of day category for each ride.

9. **Weekday/Weekend**:

    - Function: = IFS(AND(H2>=2,H2<=6),"Weekday", OR(H2=1,H2=7),"Weekend")
    - Calculation: This function categorizes the day of the week (H2) into weekday or weekend.
    - Column: O (weekend/weekday)
    - Explanation: This indicates whether each ride took place on a weekday or a weekend.

Microsoft Power BI

The calculations used in the custom columns are as follows:

- **ride_length**: Calculates the duration of the ride in minutes by subtracting the "started_at" column from the "ended_at" column and multiplying the result by 1440.
- **Time**: Extracts the time from the "started_at" column.
- **Month Name**: Extracts the month name from the "started_at" column.
- **Day Name**: Extracts the day name from the "started_at" column.
- **Month**: Extracts the month number from the "started_at" column.
- **Custom**: Categorizes the months into seasons based on the month number.
- **Hour**: Extracts the hour from the "Time" column.
- **Custom**: Categorizes the hours into time of day.
- **Custom**: Categorizes the days into holidays or weekdays/weekends based on the "Holidays" and "Day Name" columns.

Holidays

- New Year's Day**: Monday, January 2, 2023**[1]
- Dr. Martin Luther King Jr.'s Birthday**: Monday, January 16, 2023**[1]
- Lincoln's Birthday**: Monday, February 13, 2023**[1]
- Washington's Birthday**: Monday, February 20, 2023**[1]
- Pulaski Day**: Monday, March 6, 2023**[1]
- Memorial Day**: Monday, May 29, 2023**[1]
- Juneteenth Day**: Monday, June 19, 2023**[1]
- Independence Day**: Tuesday, July 4, 2023**[1]

- Labor Day**: Monday, September 4, 2023**[1]
- Columbus Day**: Monday, October 9, 2023**[1]
- Veterans Day**: Friday, November 10, 2023**[1]
- Thanksgiving Day**: Thursday, November 23, 2023**[1]
- Christmas Day**: Monday, December 25, 2023**[1]

After that I went to the **analyse** phase.

Analyse

R

1. `jan01_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202301-divvy-tripdata.csv")`
2. `feb02_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202302-divvy-tripdata.csv")`
3. `mar03_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202303-divvy-tripdata.csv")`
4. `apr04_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202304-divvy-tripdata.csv")`
5. `may05_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202305-divvy-tripdata.csv")`
6. `jun06_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202306-divvy-tripdata.csv")`
7. `jul07_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202307-divvy-tripdata.csv")`
8. `aug08_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202308-divvy-tripdata.csv")`
9. `sep09_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202309-divvy-tripdata.csv")`
10. `oct10_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202310-divvy-tripdata.csv")`
11. `nov11_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202311-divvy-tripdata.csv")`
12. `dec12_df <- read_csv("D:/IMP/Google Data Analytics/Capstone/CSV/202312-divvy-tripdata.csv")`

    : Above 12 lines of code reads 12 CSV files from the path to the R script

13. `cyc_df <- rbind(jan01_df, feb02_df, mar03_df, apr04_df, may05_df, jun06_df, jul07_df, aug08_df, sep09_df, oct10_df, nov11_df, dec12_df)`

    : Combines all the data frames created in the previous steps into one data frame called `cyc_df`.

14. `remove(jan01_df, feb02_df, mar03_df, apr04_df, may05_df, jun06_df, jul07_df, aug08_df, sep09_df, oct10_df, nov11_df, dec12_df)`

    : Removes the data frames created in the previous steps to free up memory.

15. `cyc_df$ride_length <- difftime(cyc_df$ended_at, cyc_df$started_at, units = "mins")`

    : Calculates the ride length by subtracting the `ended_at` time from the `started_at` time and converts it to minutes.

16. `cyclists_data <- cyc_df`

    : Creates a new data frame called `cyclists_data` to contain new columns.

17. `cyclists_data$date <- as.Date(cyclists_data$started_at)`

    : Converts the `started_at` column to a date format and stores it in a new column called `date`.

18. `cyclists_data$day_of_week <- wday(cyc_df$started_at)`

    : Calculates the day of the week and stores it in a new column called `day_of_week`.

19. `cyclists_data$day_of_week <- format(as.Date(cyclists_data$date), "%A")`

    : Formats the `date` column to display the day of the week and stores it in a new column called `day_of_week`.

20. `cyclists_data$month <- format(as.Date(cyclists_data$date), "%m")`

    : Formats the `date` column to display the month and stores it in a new column called `month`.

21. `cyclists_data$day <- format(as.Date(cyclists_data$date), "%d")`

    : Formats the `date` column to display the day and stores it in a new column called `day`.

22. `cyclists_data$year <- format(as.Date(cyclists_data$date), "%Y")`

    : Formats the `date` column to display the year and stores it in a new column called `year`.

23. `cyclists_data$time <- format(as.Date(cyclists_data$date), "%H:%M:%S")`

    : Formats the `date` column to display the time as HH:MM:SS and stores it in a new column called `time`.

24. `cyclists_data$time <- as_hms((cyc_df$started_at))`

    : Converts the `started_at` column to a time format and stores it in a new column called `time`.

25. `cyclists_data$hour <- hour(cyclists_data$time)`

    : Extracts the hour from the `time` column and stores it in a new column called `hour`.

26. `cyclists_data <-cyclists_data %>% mutate(season = case_when(month == "03" ~ "Spring", month == "04" ~ "Spring", month == "05" ~ "Spring", month == "06" ~ "Summer", month == "07" ~ "Summer", month == "08" ~ "Summer", month == "09" ~ "Fall", month == "10" ~ "Fall", month == "11" ~ "Fall", month == "12" ~ "Winter", month == "01" ~ "Winter", month == "02" ~ "Winter"))`

    : Creates a new column called `season` and assigns a season based on the month.

27. `cyclists_data <-cyclists_data %>% mutate(time_of_day = case_when(hour == "0" ~ "Night", hour == "1" ~ "Night", hour == "2" ~ "Night", hour == "3" ~ "Night", hour == "4" ~ "Night", hour == "5" ~ "Night", hour == "6" ~ "Morning", hour == "7" ~ "Morning", hour == "8" ~ "Morning", hour == "9" ~ "Morning", hour == "10" ~ "Morning", hour == "11" ~ "Morning", hour == "12" ~ "Afternoon", hour == "13" ~ "Afternoon", hour == "14" ~ "Afternoon", hour == "15" ~ "Afternoon", hour == "16" ~ "Afternoon", hour == "17" ~ "Afternoon", hour == "18" ~ "Evening", hour == "19" ~ "Evening", hour == "20" ~ "Evening", hour == "21" ~ "Night", hour == "22" ~ "Night", hour == "23" ~ "Night"))`

    : Creates a new column called `time_of_day` and assigns a time of day based on the hour.

28. `cyclists_data <-cyclists_data %>% mutate(tiype_of_day = case_when(date == "2023-01-02" ~ "Public Holiday", date == "2023-01-16" ~ "Public Holiday", date == "2023-02-13" ~ "Public Holiday", date == "2023-02-20" ~ "Public Holiday", date == "2023-03-06" ~ "Public Holiday", date == "2023-05-29" ~ "Public Holiday", date == "2023-06-19" ~ "Public Holiday", date == "2023-07-04" ~ "Public Holiday", date == "2023-09-04 ~ "Public Holiday", date == "2023-10-09" ~ "Public Holiday", date == "2023-11-10" ~ "Public Holiday", date == "2023-11-23" ~ "Public Holiday", date == "2023-12-25" ~ "Public Holiday", day_of_week == "Monday" ~ "Weekday", day_of_week == "Tuesday" ~ "Weekday", day_of_week == "Wednesday" ~ "Weekday", day_of_week == "Thursday" ~ "Weekday", day_of_week == "Friday" ~ "Weekday", day_of_week == "Saturday" ~ "Weekend", day_of_week == "Sunday" ~ "Weekend"))`

    : Creates a new column called `type_of_day` and assigns a type of day based on the date or day of the week.

29. `cyclists_data <- cyclists_data %>% select(-c(start_station_id, start_station_name, end_station_id, end_station_name, start_lat,start_lng,end_lat,end_lng))`

    : Removes columns not Removes column not needed
    `start_station_id`, `start_station_name`, `end_station_id`, `end_station_name`, `start_lat`, `start_lng`, `end_lat`, `end_lng`.

30. `cyclists_data <- cyclists_data[!(cyclists_data$ride_length <=0),]`

: Removes rows where ride_length is 0 or negative.

31. `cyclists_data <- distinct(cyclists_data)`

    : Removes duplicate rows.

32. `cyclists_data <- na.omit(cyclists_data)`

    : Removes rows with NA values.

33. `View(cyclists_data)`

    : Displays the final data in a table view.

34. `fwrite(cyclists_cap,"cyclists_data.csv")`

    : Saves the final data as a .csv file.

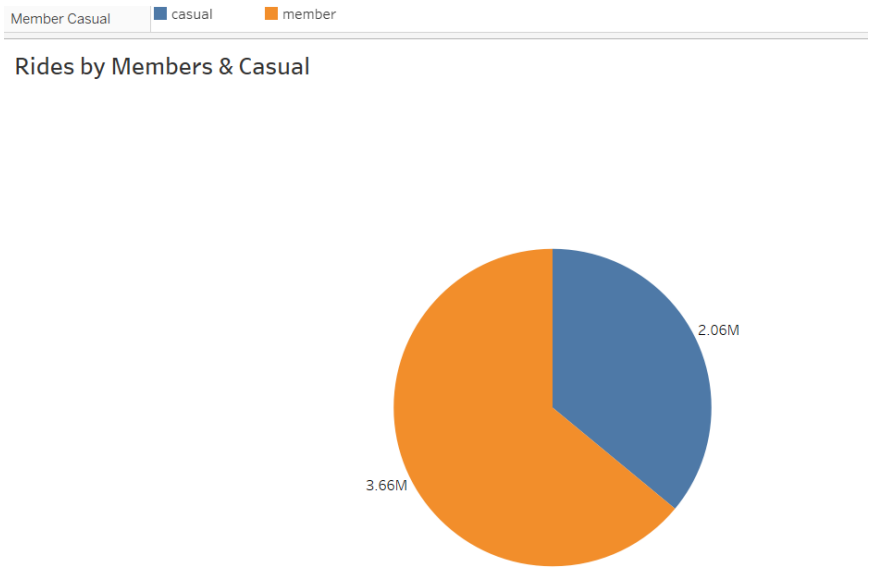
The code uses the `dplyr` package, which is part of the `tidyverse` collection of packages for data manipulation, exploration, and visualization in R. The `dplyr` package provides a set of functions for data manipulation, including `mutate()`, `select()`, and `distinct()`.
The `%>%` operator is used to chain together multiple operations. The `case_when()` function is used to assign values to a new column based on conditions. The `na.omit()` function is used to remove rows with missing values. The `fwrite()` function is used to save the final data as a .csv file. Overall, the code is used to clean and manipulate data to prepare it for analysis or visualization.

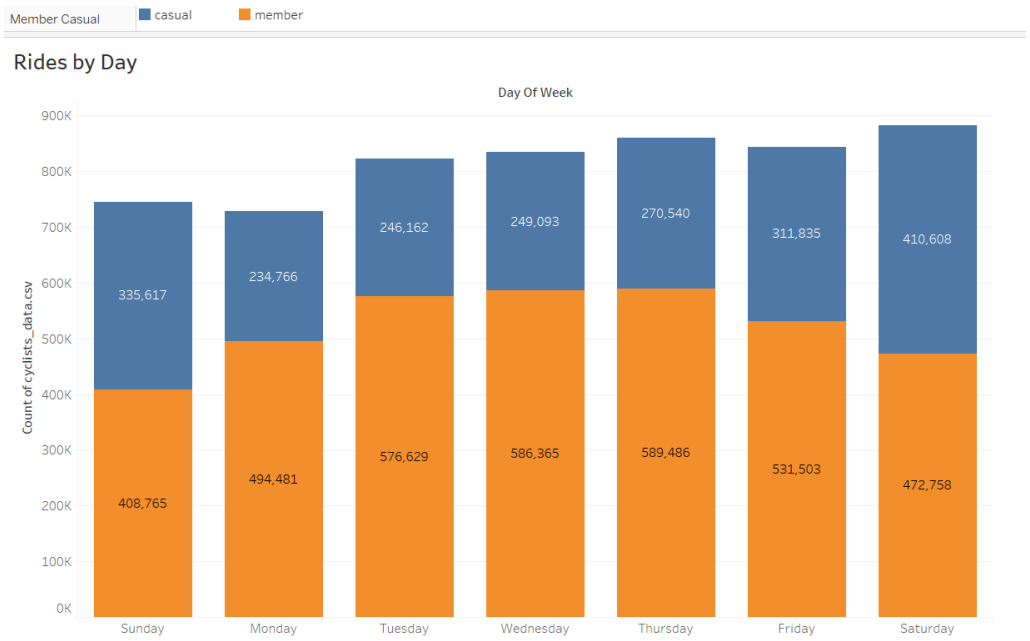
Share

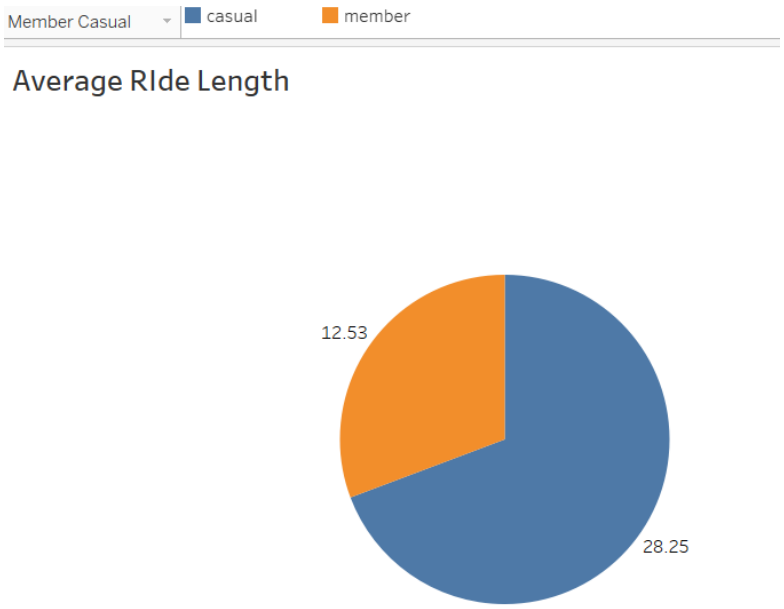
Tableau

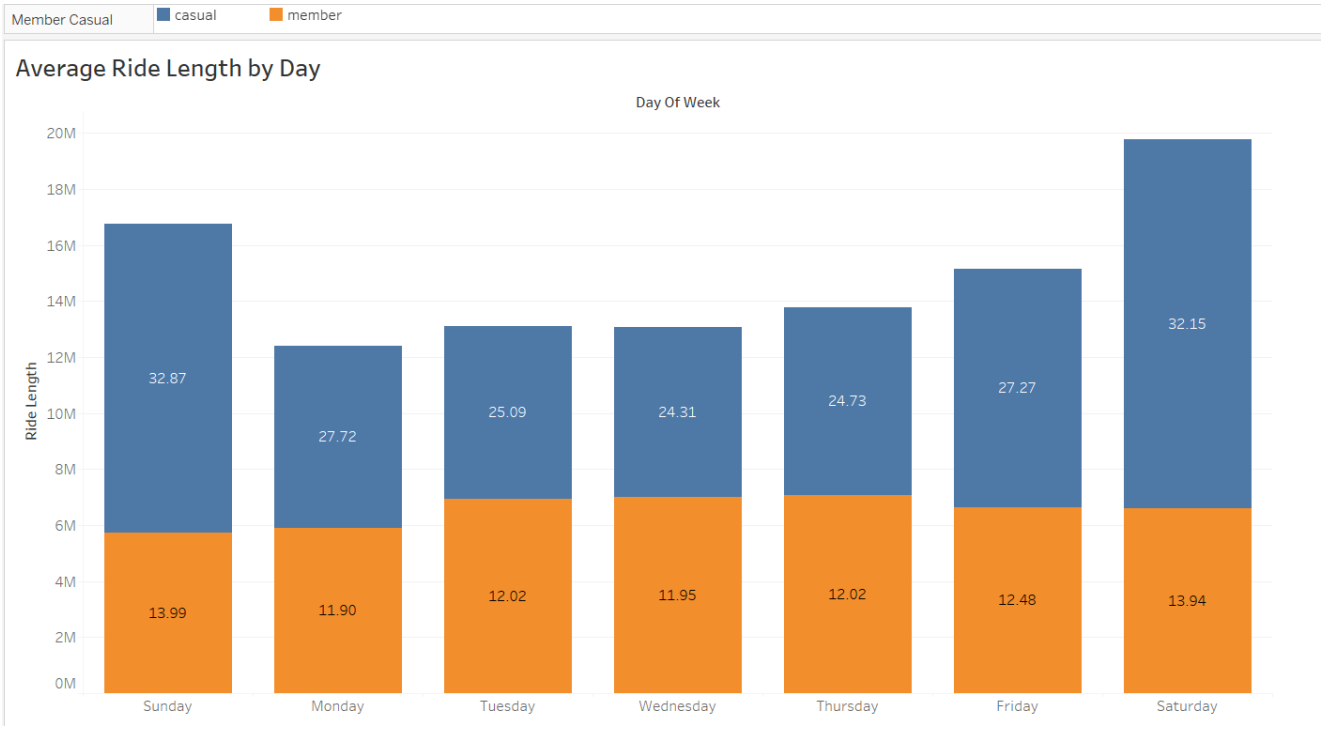
In order to visualize the findings, I opted for using tableau.

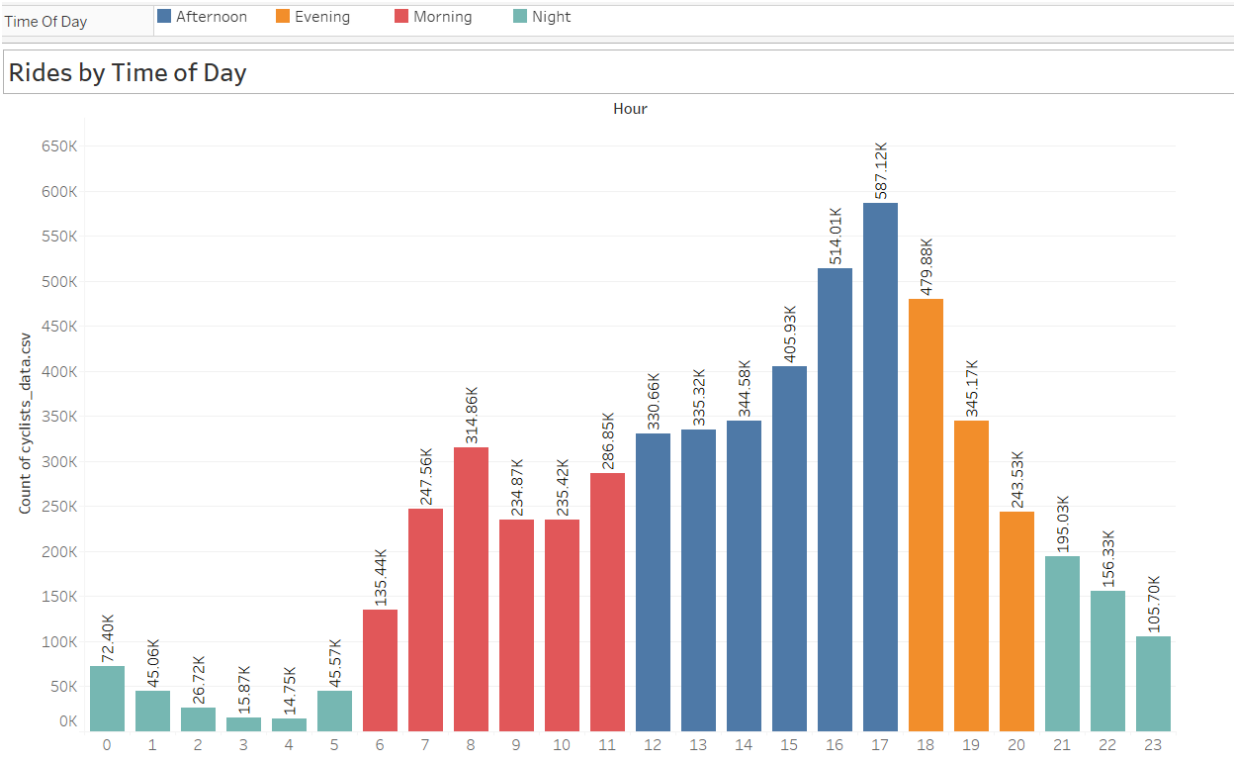
The data processing or the transformations have been done using R and the saved dataset i.e., cyclists_data is loaded to tableau for the visualization
Tableau graphs were created for:

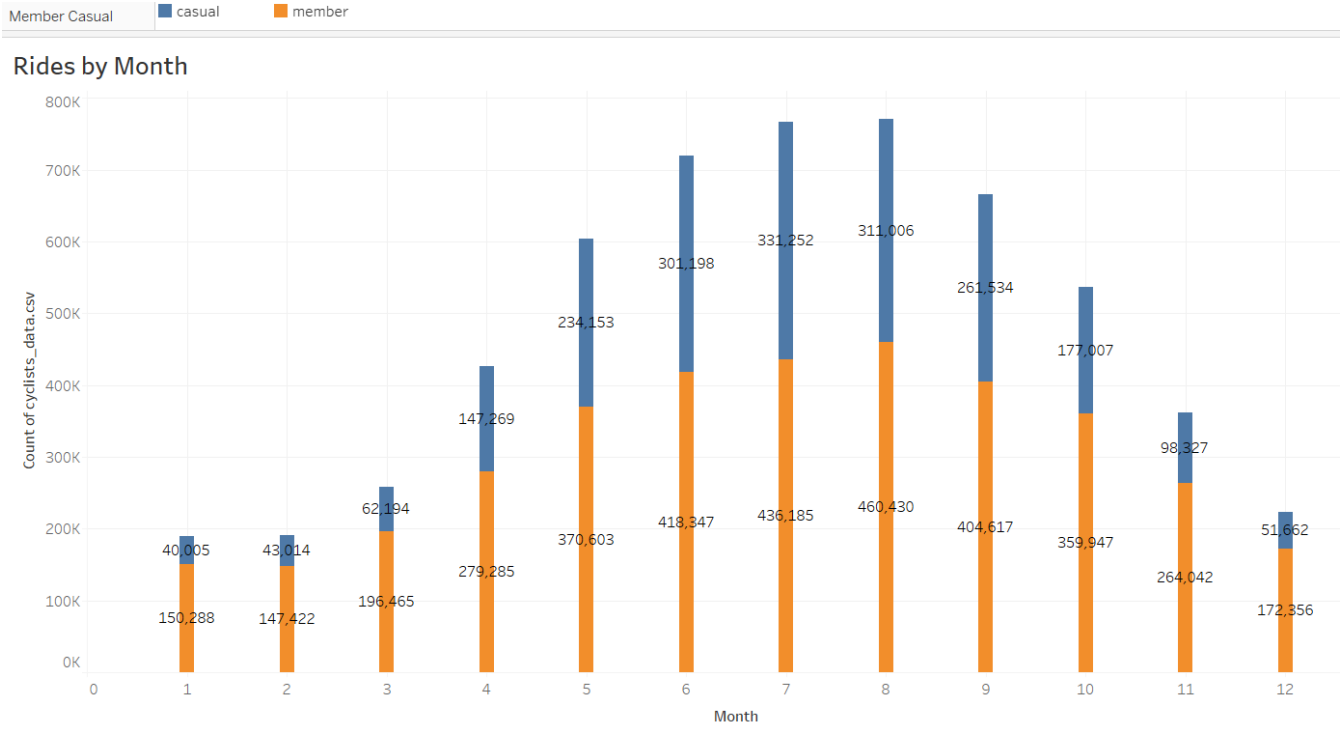

Member Casual
casual
member
Rides by Members & Casual
2.06M
3.66M

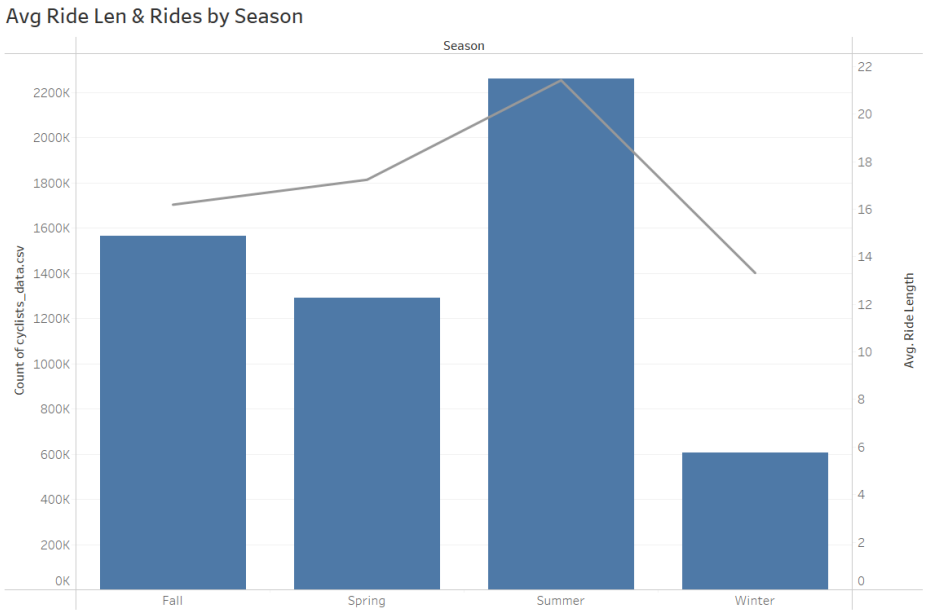

Member Casual
casual
member
Rides by Day
Day Of Week
Count of cyclists_data.csv
900K
800K
700K
600K
500K
400K
300K
200K
100K
0K
335,617
408,765
234,766
494,481
246,162
576,629
249,093
586,365
270,540
589,486
311,835
531,503
410,608
472,758
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

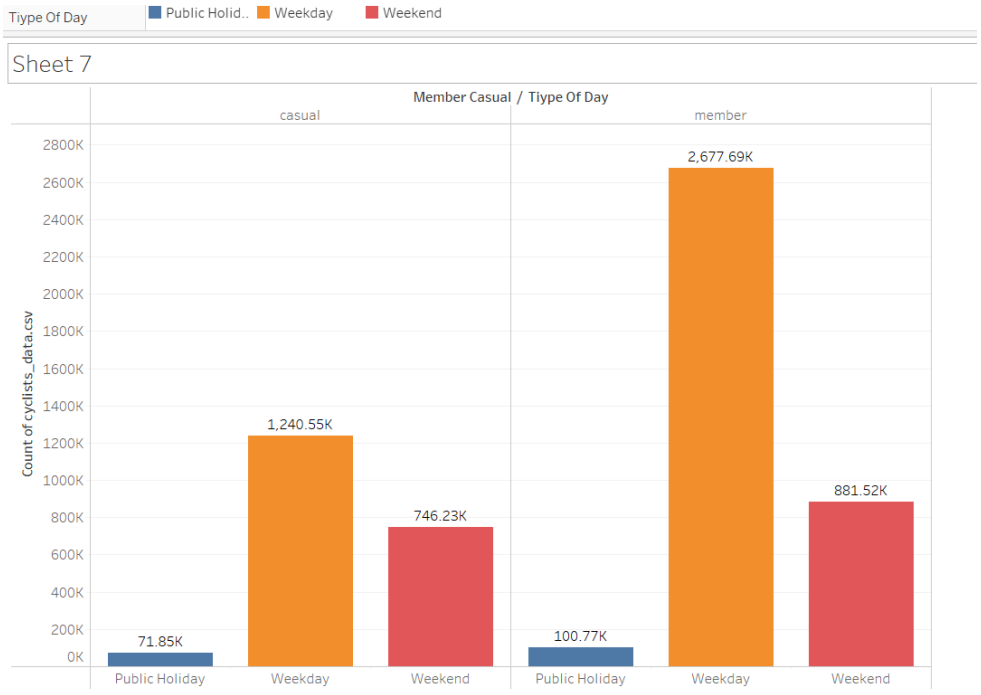

Member Casual
casual
member
Average RIde Length
12.53
28.25


Member Casual
casual
member
Average Ride Length by Day
Day Of Week
Ride Length
20M
18M
16M
14M
12M
10M
8M
6M
4M
2M
0M
32.87
13.99
27.72
11.90
25.09
12.02
24.31
11.95
24.73
12.02
27.27
12.48
32.15
13.94
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

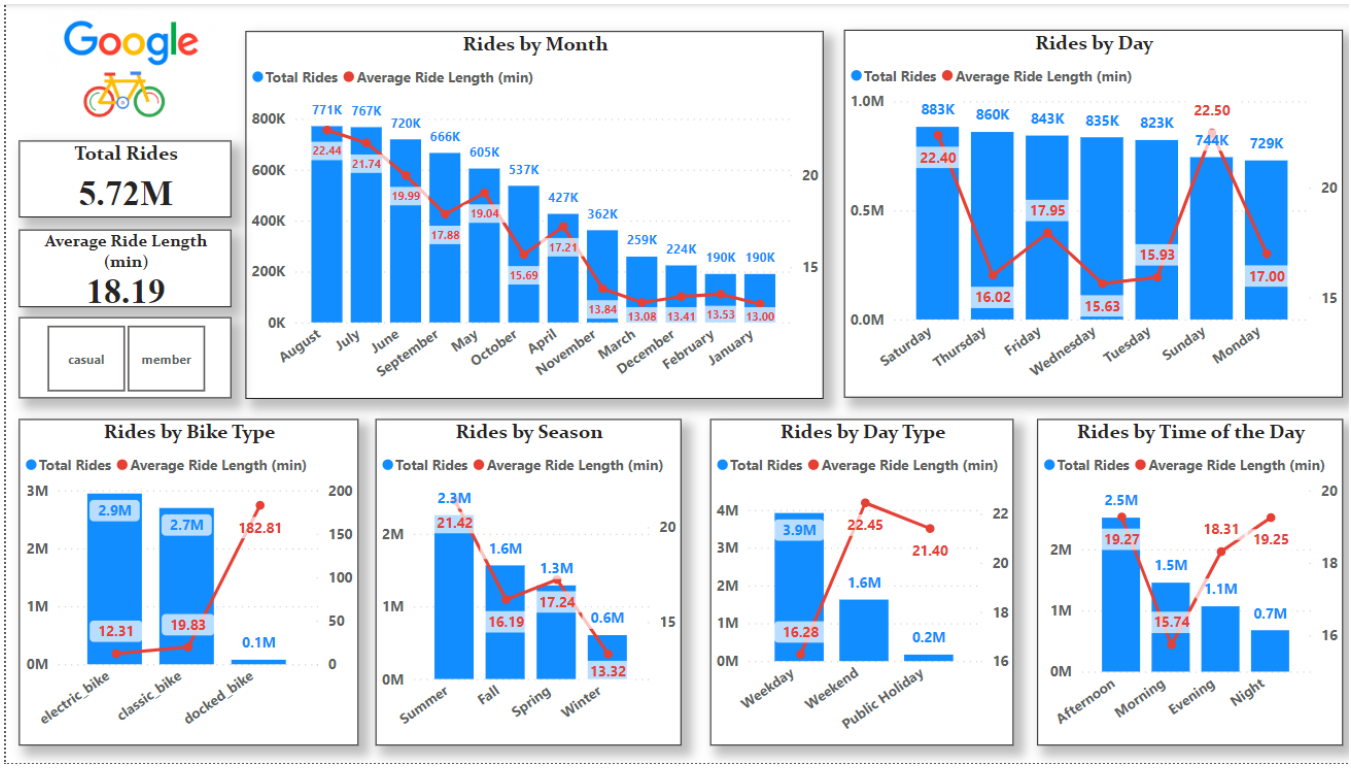

Time Of Day
Afternoon
Evening
Morning
Night
Rides by Time of Day
Hour
Count of cyclists_data.csv
650K
600K
550K
500K
450K
400K
350K
300K
250K
200K
150K
100K
50K
0K
72.40K
45.06K
26.72K
15.87K
14.75K
45.57K
135.44K
247.56K
314.86K
234.87K
235.42K
286.85K
330.66K
335.32K
344.58K
405.93K
514.01K
587.12K
479.88K
345.17K
243.53K
195.03K
156.33K
105.70K
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23


Member Casual
casual
member
Rides by Month
Count of cyclists_data.csv
800K
700K
600K
500K
400K
300K
200K
100K
0K
40,005
150,288
43,014
147,422
62,194
196,465
147,269
279,285
234,153
370,603
301,198
418,347
331,252
436,185
311,006
460,430
261,534
404,617
177,007
359,947
98,327
264,042
51,662
172,356
0
1
2
3
4
5
6
7
8
9
10
11
12
Month


Avg Ride Len & Rides by Season
Season
Count of cyclists_data.csv
Avg. Ride Length
2200K
2000K
1800K
1600K
1400K
1200K
1000K
800K
600K
400K
200K
0K
22
20
18
16
14
12
10
8
6
4
2
0
Fall
Spring
Summer
Winter


Tiype Of Day
Public Holid..
Weekday
Weekend
Sheet 7
Member Casual / Tiype Of Day
casual
member
Count of cyclists_data.csv
2800K
2600K
2400K
2200K
2000K
1800K
1600K
1400K
1200K
1000K
800K
600K
400K
200K
0K
71.85K
1,240.55K
746.23K
100.77K
2,677.69K
881.52K
Public Holiday
Weekday
Weekend
Public Holiday
Weekday
Weekend

Power BI



Act

Below you will find a summary of my key findings. Based on the findings I will answer the questions that was initially made in the **Ask** phase.

Key Findings

- **Members** had the bigger share of rides, amounting to a total of **63%** of all rides.
- The average ride length for **annual members** (*12m53s*) was less than half of the average ride length of **casual riders** (*28m25s*)
- However, when talking about average ride length both **members** and **casual riders** tend to take longer rides on weekends.
- Both **members** and **casual riders** use Cyclists more in the Afternoon, that time of the day amounts to **43.59%** of all rides. The busiest hour turned out to be **16:00/4 PM** for both **members** and **casual riders**, with **10%** of all rides and the average ride lengths are more in the evenings and at the nights.
- The busiest month for **casual riders** was July, as for **members** the busiest was actually August. The **3rd Quarter** was the busiest, counting for **40.77%** of all rides which was expected being that it includes most of the summer season.

Suggestions

- **How do annual members and casual riders use Cyclists bikes differently?**

  Based on what was found casual riders tend to use Cyclists to make longer rides, the average ride length of casual riders (28 min) more than doubles the average ride length of annual members (13 min), which screams in the eyes of casual riders using Cyclists (buying a daily pass or single ride) “only” makes sense if they are going to take longer rides. This is supported by the fact that casual riders have a smaller percentage of the total rides (~37%), concluding that annual members use Cyclists with more freedom since they don’t have to be concerned with maximizing their rides, they can always take a short ride, stop and take another with no downside.

- **What would make casual riders buy a membership?**

  It is observed that the average lengths are more on the weekends and public holidays for both casual with lesser rides suggests that they use bikes only as a medium of transport maybe roam around to have fun and members use it more often than not even on the weekdays shows that they use bikes to travel for their work or something like that, so annual memberships save a lot for people who use it for their office. To attract casual riders maybe give some discounts on the annual memberships also try tom increase the price for single and daily passes.

- **How can Cyclists use digital media to influence casual riders to become members?**

  Investment in ads is needed. Advertising in platforms like Spotify, making an ad promoting the discounts suggested above in a platform like Spotify would gather a lot of annual membership. It is also suggested investing in ads on **podcasts**, **YouTube content creators** and **twitter pages** of which most of the fanbase is composed of bikers, YouTube channels about biking and exercise would be a good example. A Cyclists app with a personal profile that takes in consideration how each rider uses the company services and makes recommendations on how to maximize the potential of Cyclists is created. It should include the promotions available and the benefits of becoming an annual member.