Cyclistic Bike Share Analysis

Grace Elsa George

2024-02-29

CYCLISTIC BIKE SHARE: CASE STUDY

AN EXPLORATORY DATA ANALYSIS USING EXCEL AND R

This is a capstone project which is a part of Google Data Analytics Professional Certificate Course. In this project, information about the rides made by the customers of Cyclistic bike share company for the year 2023 is analysed.

Scenario and Problem statement

As a junior data analyst working on the marketing analyst team at Cyclistic, a bike-share company in Chicago, the team needs information on maximizing the number of annual memberships for the future success of the organization. Therefore, marketing analyst team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, the team will design a new marketing strategy to convert casual riders into annual members. But first, Cyclistic executives must approve the recommendations, so they must be backed up with compelling data insights and professional data visualizations.

Approach used

This project uses the following data analysis approach:

1. Ask : Understanding the scenario and problem by identifying the business task and key stakeholders.
2. Prepare : Gathering the required data and check for its credibility and organize as per the requirement.
3. Process : Checking the data for any errors or inconsistencies and cleaning them to make the data more reliable. Transform the data to make it effective when analysis is done.
4. Analysis : Aggregate, organize and format your data so it's useful and accessible. Perform calculations and identify trends and relationships within data.
5. Share : Create effective data visualizations to present your findings. Ensure your work is accessible.
6. Act : Prepare the deliverable including the top recommendations based on the analysis.

Tools used

- MS Excel to clean and transform the data.
- RStudio IDE to do the analysis and visualization.

Data Source

Cyclistic's historical trip data to analyse and identify trends can be downloaded for previous 12 months from here **link**. (Note: The datasets have a different name because Cyclistic is a fictional company. For the purposes of this case study,

the datasets are appropriate and will enable you to answer the business questions. The data has been made available by Motivate International Inc. under this license **link**.

- ASK

In order to understand the business task **SMART** framework is used i.e, **S**pecific, **M**easurable, **A**ction-oriented, **R**elevant, **T**ime-bound.

For the future success of the organization, The Director of Marketing team is planning to increase the number of annual members of the Cyclistic bike share program. The business task is to understand how casual riders and annual members use Cyclistic bikes differently so that new marketing strategy can be created to convert casual riders into annual members. In-order to approve the new strategy and make decisions, the team needs a thorough analysis to get important data insights and recommendations.

The key stakeholders are The Marketing Director, Cyclistic Executive Team and Cyclistic marketing Analytics Team.

- PREPARE

The Cyclistic dataset for the year 2023 is downloaded and it includes 12 separate csv files with ride details for each month. There are 13 columns, including the ride IDs, bike type, start and end timestamp, start and end station name, location and rider type.

Using the ROCCC framework, the data is checked for its credibility as follows:

- R (Reliable): The dataset contains inconsistent station names. There are some riding time inaccuracies and missing station names as well. But as they only make up a small percentage of the information, they have little effect on the accuracy of the data as a whole.
- O (Original): The data is available in its original site i.e. the data source is primary.
- C (Comprehensive): The dataset contains more than four million rows of full data which makes it comprehensive.
- C (Current): The dataset is from the previous year i.e. 2023. So, the data is up-to-date.
- C (Cited): The data is cited on a reliable primary source.

The dataset is saved as excel file with .csv extension. In-order to make it compatible to perform multiple functions, they are saved as xlxs files and made a copy to maintain the original file as it is for future reference.

- PROCESS

The dataset is saved as excel file with .csv extension. In-order to make it compatible to perform multiple functions, they are saved as xlxs files and made a copy to maintain the original file as it is for future reference.

The xlxs files are checked for the errors or missing values. The dataset included inconsistent ride_id column and blank cells in station_name column which are removed as they only make up a small percentage of the entire data. To remove the specific data filter option is used.

To make the analysis phase easier, transformation is done on the dataset. It included creation of new columns named start_date, start_time, end_date, end_time, ride_length and weekday as follows:

- Start_Date, Start_time, End_Date, End_Time: Seperated the date and time from the Started_at and Ended_at columns by using Power query with “Add column” option.The columns are named as Start_Date, Start_Time, End_Date, End_Time
- ride_length: Calculated the length of each ride by subtracting the column started_at from the column ended_at (for example, =D2-C2) and format as HH:MM:SS using Format > Cells > Time > 37:30:55.
- Weekday: calculate the day of the week that each ride started using the WEEKDAY command (for example, =WEEKDAY(C2,1)) in each file. Format as General or as a number with no decimals, noting that 1 = Sunday and 7 = Saturday.

- ANALYSIS

The dataset is cleaned, transformed and ready for analysis. Descriptive analysis is performed on the data using R programming in RStudio IDE.

First, a new project is created required library packages are installed and loaded into the console.

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

```
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\grace\AppData\Local\Temp\Rtmp6xvdms\downloaded_packages
```

```
install.packages("skimr")
```

```
## package 'skimr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\grace\AppData\Local\Temp\Rtmp6xvdms\downloaded_packages
```

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

```
## package 'janitor' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\grace\AppData\Local\Temp\Rtmp6xvdms\downloaded_packages
```

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

```
## package 'lubridate' successfully unpacked and MD5 sums checked
## Warning: cannot remove prior installation of package 'lubridate'
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Users\grace\AppData\Local\R\win-library\4.3\00LOCK\lubridate\libs\x64\lubridate.dll
## to
## C:\Users\grace\AppData\Local\R\win-library\4.3\lubridate\libs\x64\lubridate.dll:
## Permission denied
## Warning: restored 'lubridate'
```

```
##
## The downloaded binary packages are in
##  C:\Users\grace\AppData\Local\Temp\Rtmp6xvdms\downloaded_packages
```

```
install.packages("dplyr")
```

```
## package 'dplyr' successfully unpacked and MD5 sums checked
## Warning: cannot remove prior installation of package 'dplyr'
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Users\grace\AppData\Local\R\win-library\4.3\00LOCK\dplyr\libs\x64\dplyr.dll
## to C:\Users\grace\AppData\Local\R\win-library\4.3\dplyr\libs\x64\dplyr.dll:
## Permission denied
## Warning: restored 'dplyr'
##
## The downloaded binary packages are in
##  C:\Users\grace\AppData\Local\Temp\Rtmp6xvdms\downloaded_packages
```

```
library(tidyverse)
library(skimr)
library(janitor)
library(lubridate)
library(dplyr)
```

Now, current working directory is set to the file path of the local device where the datasets are stored. Then, the data of 12 months are read and combined to a single data frame named “cyclistic_complete_data” for further analysis.

```
setwd("C:/Data Analysis/Google Data Analytics Course/Case studies/Case study 1/Cyclistic-datasets/3.Cleaned dataset final")
cyclistic_complete_data <- list.files(pattern = "*.csv",full.names = T) %>% lapply(read_csv) %>% bind_rows()
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
## One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
```

```
##   problems(dat)
```

Check the merged data frame “cyclistic_complete_data” for any inconsistencies.

```
colnames(cyclistic_complete_data)  #List of column names
```

```
##  [1] "ride_id"            "rideable_type"      "Start_Date"
##  [4] "Start_Time"         "End_Date"           "End_Time"
##  [7] "start_station_name" "start_station_id"   "end_station_name"
## [10] "end_station_id"     "start_lat"          "start_lng"
## [13] "end_lat"            "end_lng"            "member_casual"
## [16] "ride_length"
```

```
head(cyclistic_complete_data)  #See the first 6 rows of data frame.
```

ride_id <chr>	rideable_type <chr>	Start_Date <chr>	Start_Time <time>	End_Date <chr>	End_Time <time>
5B6500E1E58655C0	classic_bike	10-04-2023	17:34:00	10-04-2023	18:02:00
AA65D25D69AF771F	classic_bike	12-04-2023	12:29:00	12-04-2023	12:54:00
079FB2C196414482	electric_bike	13-04-2023	17:39:00	13-04-2023	17:40:00
599623864C871207	classic_bike	29-04-2023	20:57:00	29-04-2023	20:57:00
63ECC8A13D11A76A	classic_bike	20-04-2023	17:03:00	20-04-2023	17:24:00
A396F54F4C1927D8	electric_bike	14-04-2023	22:29:00	14-04-2023	22:29:00

6 rows | 1-6 of 16 columns

```
tail(cyclistic_complete_data)  #See the last rows of data frame.
```

ride_id <chr>	rideable_type <chr>	Start_Date <chr>	Start_Time <time>	End_Date <chr>	End_Time <time>
44149E61BFC71FA8	electric_bike	20-09-2023	07:15:00	20-09-2023	07:29:00
E614A9BE598C53CD	electric_bike	25-09-2023	07:14:00	25-09-2023	07:22:00
ED01CDDC72B5EBE8	classic_bike	26-09-2023	15:21:00	26-09-2023	15:30:00
DB6FCF2E96662AC3	electric_bike	27-09-2023	15:35:00	27-09-2023	15:42:00
0C3ACCE37F56C4A9	electric_bike	03-09-2023	01:56:00	03-09-2023	02:06:00
972A982FE3AFD22A	classic_bike	26-09-2023	13:26:00	26-09-2023	13:32:00

6 rows | 1-6 of 16 columns

```
str(cyclistic_complete_data)   #Displays the structure of data frame.
```

```
## spc_tbl_ [4,331,646 × 16] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:4331646] "5B6500E1E58655C0" "AA65D25D69AF771F" "079FB2C19641448
2" "599623864C871207" ...
##  $ rideable_type     : chr [1:4331646] "classic_bike" "classic_bike" "electric_bike" "classic
_bike" ...
##  $ Start_Date        : chr [1:4331646] "10-04-2023" "12-04-2023" "13-04-2023" "29-04-2023" ..
.
```

```
##  $ Start_Time        : 'hms' num [1:4331646] 17:34:00 12:29:00 17:39:00 20:57:00 ...
##   ..- attr(*, "units")= chr "secs"
##  $ End_Date          : chr [1:4331646] "10-04-2023" "12-04-2023" "13-04-2023" "29-04-2023" ..
.
##  $ End_Time          : 'hms' num [1:4331646] 18:02:00 12:54:00 17:40:00 20:57:00 ...
##   ..- attr(*, "units")= chr "secs"
##  $ start_station_name: chr [1:4331646] "Avenue O & 134th St" "Cottage Grove Ave & 51st St" "M
organ Ave & 14th Pl" "Cottage Grove Ave & 51st St" ...
##  $ start_station_id  : chr [1:4331646] "20214" "TA1309000067" "TA1306000002" "TA1309000067" .
..
##  $ end_station_name  : chr [1:4331646] "Avenue O & 134th St" "Cottage Grove Ave & 51st St" "M
organ Ave & 14th Pl" "Cottage Grove Ave & 51st St" ...
##  $ end_station_id    : chr [1:4331646] "20214" "TA1309000067" "TA1306000002" "TA1309000067" .
..
##  $ start_lat         : num [1:4331646] 41.7 41.8 41.9 41.8 41.9 ...
##  $ start_lng         : num [1:4331646] -87.5 -87.6 -87.7 -87.6 -87.7 ...
##  $ end_lat           : num [1:4331646] 41.7 41.8 41.9 41.8 41.9 ...
##  $ end_lng           : num [1:4331646] -87.5 -87.6 -87.7 -87.6 -87.7 ...
##  $ member_casual     : chr [1:4331646] "member" "member" "member" "member" ...
##  $ ride_length       : 'hms' num [1:4331646] 00:28:00 00:25:00 00:01:00 00:00:00 ...
##   ..- attr(*, "units")= chr "secs"
##  - attr(*, "spec")=
##   .. cols(
##   ..   ride_id = col_character(),
##   ..   rideable_type = col_character(),
##   ..   Start_Date = col_character(),
##   ..   Start_Time = col_time(format = ""),
##   ..   End_Date = col_character(),
##   ..   End_Time = col_time(format = ""),
##   ..   start_station_name = col_character(),
##   ..   start_station_id = col_character(),
##   ..   end_station_name = col_character(),
##   ..   end_station_id = col_character(),
##   ..   start_lat = col_double(),
##   ..   start_lng = col_double(),
##   ..   end_lat = col_double(),
##   ..   end_lng = col_double(),
##   ..   member_casual = col_character(),
##   ..   ride_length = col_time(format = "")
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
cyclistic_complete_data_sorted <- arrange(cyclistic_complete_data,Start_Date)   #Sorts the data
in ascending order based on Start_Date column
print(cyclistic_complete_data_sorted)
```

```
## # A tibble: 4,331,646 × 16
```

```
##    ride_id          rideable_type Start_Date Start_Time End_Date   End_Time
##    <chr>            <chr>         <chr>      <time>     <chr>      <time>
##  1 06EF8DD39CF7170F electric_bike 01-01-2023 18:04      01-01-2023 18:09
##  2 278C5B9B24F9676B electric_bike 01-01-2023 15:34      01-01-2023 15:43
##  3 C736F89E76D0E94C electric_bike 01-01-2023 15:18      01-01-2023 15:28
##  4 E45F1695932FFAD9 electric_bike 01-01-2023 21:27      01-01-2023 21:36
##  5 1CE75766A8D48796 electric_bike 01-01-2023 17:11      01-01-2023 17:13
##  6 42F33089E1A8B14F docked_bike   01-01-2023 12:00      01-01-2023 14:53
##  7 FD00061B604B8AAD electric_bike 01-01-2023 16:25      01-01-2023 16:44
##  8 5A9F13080F18C073 electric_bike 01-01-2023 16:21      01-01-2023 16:45
##  9 66E95481EAE4FAB7 classic_bike  01-01-2023 17:21      01-01-2023 17:28
## 10 0C6BD4EBC8C2459D classic_bike  01-01-2023 02:16      01-01-2023 02:23
## # ℹ 4,331,636 more rows
## # ℹ 10 more variables: start_station_name <chr>, start_station_id <chr>,
## #   end_station_name <chr>, end_station_id <chr>, start_lat <dbl>,
## #   start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>,
## #   ride_length <time>
```

There are rows with ride_length “00:00” which are incorrect entries in the data set and cannot be considered for analysis. So, delete those rows using simple filter function.

```
cyclistic_complete_data_without_0 <- cyclistic_complete_data_sorted[cyclistic_complete_
data_sorted$ride_length != "0", ]

cyclistic_complete_data_without_0
```

ride_id <chr>	rideable_type <chr>	Start_Date <chr>	Start_Time <time>	End_Date <chr>	End_Time <time>
06EF8DD39CF7170F	electric_bike	01-01-2023	18:04:00	01-01-2023	18:09:00
278C5B9B24F9676B	electric_bike	01-01-2023	15:34:00	01-01-2023	15:43:00
C736F89E76D0E94C	electric_bike	01-01-2023	15:18:00	01-01-2023	15:28:00
E45F1695932FFAD9	electric_bike	01-01-2023	21:27:00	01-01-2023	21:36:00
1CE75766A8D48796	electric_bike	01-01-2023	17:11:00	01-01-2023	17:13:00
42F33089E1A8B14F	docked_bike	01-01-2023	12:00:00	01-01-2023	14:53:00
FD00061B604B8AAD	electric_bike	01-01-2023	16:25:00	01-01-2023	16:44:00
5A9F13080F18C073	electric_bike	01-01-2023	16:21:00	01-01-2023	16:45:00
66E95481EAE4FAB7	classic_bike	01-01-2023	17:21:00	01-01-2023	17:28:00
0C6BD4EBC8C2459D	classic_bike	01-01-2023	02:16:00	01-01-2023	02:23:00

Next
123456
…
1000
Previous
1-10 of 10,000 rows | 1-6 of 16 columns

In order to make the data more reliable, data transformation can be done by extracting day, month, quarter from Start_Date column into different columns named Day, Month and Quarter.

```
cyclistic_complete_data_quarter <- cyclistic_complete_data_without_0 %>%
    add_column(Quarter = quarter(cyclistic_complete_data_without_0$Start_Date,type = "quarter", fiscal_start = 1),
               Month = months(as.Date(cyclistic_complete_data_without_0$Start_Date)),
               Day = day(as.Date(cyclistic_complete_data_without_0$Start_Date,"%d-%m-%y")))
head(cyclistic_complete_data_quarter[c("Start_Date","Quarter","Month","Day")])
```

Start_Date <chr>	Quarter <int>	Month <chr>	Day <int>
01-01-2023	1	January	1
01-01-2023	1	January	1
01-01-2023	1	January	1
01-01-2023	1	January	1
01-01-2023	1	January	1
01-01-2023	1	January	1

6 rows

```
cyclistic_final_data <- cyclistic_complete_data_quarter
cyclistic_final_data
```

ride_id <chr>	rideable_type <chr>	Start_Date <chr>	Start_Time <time>	End_Date <chr>
06EF8DD39CF7170F	electric_bike	01-01-2023	18:04:00	01-01-2023
278C5B9B24F9676B	electric_bike	01-01-2023	15:34:00	01-01-2023
C736F89E76D0E94C	electric_bike	01-01-2023	15:18:00	01-01-2023
E45F1695932FFAD9	electric_bike	01-01-2023	21:27:00	01-01-2023
1CE75766A8D48796	electric_bike	01-01-2023	17:11:00	01-01-2023
42F33089E1A8B14F	docked_bike	01-01-2023	12:00:00	01-01-2023
FD00061B604B8AAD	electric_bike	01-01-2023	16:25:00	01-01-2023
5A9F13080F18C073	electric_bike	01-01-2023	16:21:00	01-01-2023
66E95481EAE4FAB7	classic_bike	01-01-2023	17:21:00	01-01-2023
0C6BD4EBC8C2459D	classic_bike	01-01-2023	02:16:00	01-01-2023

Next
123456
...
1000
Previous
1-10 of 10,000 rows | 1-6 of 19 columns

When the above data is checked for inconsistency, it is found that there are some records whose value is NA for the entire row. So, final data set for the analysis is prepared by selecting only the rows with complete values. Also, to make the data easier to understand, added a new column named Day_of_week with names of the day extracted from start_Date column.

```
cyclistic_final_data_v2 <- cyclistic_final_data[complete.cases(cyclistic_final_data),]
cyclistic_final_data_v2$Day_of_week <- weekdays(as.Date(cyclistic_final_data_v2$Start_Date))
cyclistic_final_data_v2
```

ride_id <chr>	rideable_type <chr>	Start_Date <chr>	Start_Time <time>	End_Date <chr>	End_Time <time>
06EF8DD39CF7170F	electric_bike	01-01-2023	18:04:00	01-01-2023	18:09:00
278C5B9B24F9676B	electric_bike	01-01-2023	15:34:00	01-01-2023	15:43:00
C736F89E76D0E94C	electric_bike	01-01-2023	15:18:00	01-01-2023	15:28:00
E45F1695932FFAD9	electric_bike	01-01-2023	21:27:00	01-01-2023	21:36:00
1CE75766A8D48796	electric_bike	01-01-2023	17:11:00	01-01-2023	17:13:00
42F33089E1A8B14F	docked_bike	01-01-2023	12:00:00	01-01-2023	14:53:00
FD00061B604B8AAD	electric_bike	01-01-2023	16:25:00	01-01-2023	16:44:00
5A9F13080F18C073	electric_bike	01-01-2023	16:21:00	01-01-2023	16:45:00
66E95481EAE4FAB7	classic_bike	01-01-2023	17:21:00	01-01-2023	17:28:00
0C6BD4EBC8C2459D	classic_bike	01-01-2023	02:16:00	01-01-2023	02:23:00

Next
123456
…
1000
Previous
1-10 of 10,000 rows | 1-6 of 20 columns

Now, lets get into the descriptive analysis…

Calculate the overall summary of ride length and group them on the basis of customer type.

```
average_ride_length <- mean(cyclistic_final_data_v2$ride_length) #average ride length ( total ride length / rides)

max_ride_length <- max(cyclistic_final_data_v2$ride_length) #longest ride

min_ride_length <- min(cyclistic_final_data_v2$ride_length) #shortest ride

# summarize the above calculated values and group them according to customer type i.e, member/casual column.
cyclistic_summary_groupby_customer <- cyclistic_final_data_v2 %>%
 group_by(member_casual) %>%
  summarise(average_ride_length=mean(ride_length),
```

```
            min_ride_length=min(ride_length),
            max_ride_length=max(ride_length))
cyclistic_summary_groupby_customer
```

member_casual <chr>	average_ride_length <drtn>	min_ride_length <drtn>	max_ride_length <drtn>
casual	1382.5976 secs	60 secs	86220 secs
member	736.4851 secs	60 secs	86100 secs

2 rows

```
#Summary by customer,month grouping
cyclistic_summary_groupby_customer_month <- cyclistic_final_data_v2 %>%
 group_by(member_casual,Month) %>%
  arrange(member_casual,Month) %>%
  summarise(average_ride_length=mean(ride_length),
            min_ride_length=min(ride_length),
            max_ride_length=max(ride_length))
cyclistic_summary_groupby_customer_month
```

member_casual <chr>	Month <chr>	average_ride_length <drtn>	min_ride_length <drtn>	max_ride_length <drtn>
casual	April	1366.4914 secs	60 secs	85800 secs
casual	August	1466.1885 secs	60 secs	86220 secs
casual	December	1000.1083 secs	60 secs	84900 secs
casual	February	1069.5689 secs	60 secs	85440 secs
casual	January	900.3566 secs	60 secs	81120 secs
casual	July	1522.1767 secs	60 secs	86040 secs
casual	June	1448.3479 secs	60 secs	86040 secs
casual	March	1013.2458 secs	60 secs	85680 secs
casual	May	1477.0063 secs	60 secs	86160 secs
casual	November	1074.2027 secs	60 secs	85560 secs

Next
123
Previous
1-10 of 24 rows

```
library(dplyr)
#Sorting the above result based on month.
cyclistic_final_data_v2$Month <- ordered(cyclistic_final_data_v2$Month,
                                levels=c("January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"))
```

```
cyclistic_final_data_v2 %>%
  group_by(member_casual, Month) %>%
  summarise(number_of_rides = n(), average_ride_length = mean(ride_length), min_ride_length=min(ride_length),
            max_ride_length=max(ride_length), .groups="drop") %>%
  arrange(member_casual, Month)
```

member_casual <chr>	Month <ord>	number_of_rides <int>	average_ride_length <drtn>	min_ride_length <drtn>	max_ride_length <drtn>
casual	January	29276	900.3566 secs	60 secs	81120 secs
casual	February	32407	1069.5689 secs	60 secs	85440 secs
casual	March	46158	1013.2458 secs	60 secs	85680 secs
casual	April	109157	1366.4914 secs	60 secs	85800 secs
casual	May	175086	1477.0063 secs	60 secs	86160 secs
casual	June	217585	1448.3479 secs	60 secs	86040 secs
casual	July	242906	1522.1767 secs	60 secs	86040 secs
casual	August	231798	1466.1885 secs	60 secs	86220 secs
casual	September	195297	1417.9050 secs	60 secs	85500 secs
casual	October	129183	1289.9566 secs	60 secs	86220 secs

Next
123
Previous
1-10 of 24 rows

```
#Summarize and sort the results based on day.
cyclistic_final_data_v2$Day_of_week <- ordered(cyclistic_final_data_v2$Day_of_week,
                                    levels=c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))


cyclistic_final_data_v2 %>%
  group_by(member_casual, Day_of_week) %>%  #groups by member_casual
  summarise(number_of_rides = n(), average_ride_length = mean(ride_length),.groups="drop") %>%
  arrange(member_casual, Day_of_week)
```

member_casual <chr>	Day_of_week <ord>	number_of_rides <int>	average_ride_length <drtn>
casual	Sunday	223951	1393.3875 secs
casual	Monday	220321	1395.6121 secs
casual	Tuesday	209464	1344.9480 secs
casual	Wednesday	205029	1348.6221 secs

member_casual <chr>	Day_of_week <ord>	number_of_rides <int>	average_ride_length <drtn>
casual	Thursday	217606	1397.4045 secs
casual	Friday	227660	1406.8079 secs
casual	Saturday	212734	1386.5208 secs
member	Sunday	394466	740.8961 secs
member	Monday	396305	734.7792 secs
member	Tuesday	392466	723.5304 secs

Next
12
Previous
1-10 of 14 rows

```
#Summarize the calculations based on rideable types.
cyclistic_final_data_v2 %>%
  group_by(member_casual, rideable_type) %>%  #groups by member_casual
  summarise(number_of_rides = n(), average_ride_length = mean(ride_length),.groups="drop") %>%
  arrange(member_casual, rideable_type)
```

member_casual <chr>	rideable_type <chr>	number_of_rides <int>	average_ride_length <drtn>
casual	classic_bike	866276	1550.3500 secs
casual	docked_bike	75690	3189.7883 secs
casual	electric_bike	574799	891.8066 secs
member	classic_bike	1800228	785.9426 secs
member	electric_bike	963293	644.0576 secs

5 rows

```
#Sorting the above result including quarter of the year.
cyclistic_final_data_v2 %>%
  group_by(member_casual, rideable_type, Quarter) %>%  #groups by member_casual
  summarise(number_of_rides = n(), average_ride_length = mean(ride_length),.groups="drop") %>%
  arrange(member_casual, rideable_type, Quarter)
```

member_casual <chr>	rideable_type <chr>	Quarter <int>	number_of_rides <int>	average_ride_length <drtn>
casual	classic_bike	1	48355	1170.0019 secs
casual	classic_bike	2	255040	1601.0437 secs
casual	classic_bike	3	418658	1592.9569 secs

member_casual <chr>	rideable_type <chr>	Quarter <int>	number_of_rides <int>	average_ride_length <drtn>
casual	classic_bike	4	144223	1464.5462 secs
casual	docked_bike	1	6745	2468.6820 secs
casual	docked_bike	2	35819	3179.8693 secs
casual	docked_bike	3	33126	3347.3429 secs
casual	electric_bike	1	52741	655.3357 secs
casual	electric_bike	2	210969	951.2017 secs
casual	electric_bike	3	218217	956.5240 secs

Next
12
Previous
1-10 of 19 rows

```
#Summary based on month order.
cyclistic_final_data_v2 %>%
  group_by(member_casual, Month, rideable_type) %>%  #groups by member_casual
  summarise(number_of_rides = n(), average_ride_length = mean(ride_length),.groups="drop") %>%
  arrange(member_casual, Month, rideable_type)
```

member_casual <chr>	Month <ord>	rideable_type <chr>	number_of_rides <int>	average_ride_length <drtn>
casual	January	classic_bike	13772	1040.5025 secs
casual	January	docked_bike	1675	2264.9910 secs
casual	January	electric_bike	13829	595.5008 secs
casual	February	classic_bike	15347	1221.1077 secs
casual	February	docked_bike	2139	2590.4909 secs
casual	February	electric_bike	14921	695.6719 secs
casual	March	classic_bike	19236	1221.9432 secs
casual	March	docked_bike	2931	2496.1924 secs
casual	March	electric_bike	23991	664.7393 secs
casual	April	classic_bike	48326	1586.7570 secs

Next
123456
Previous
1-10 of 56 rows

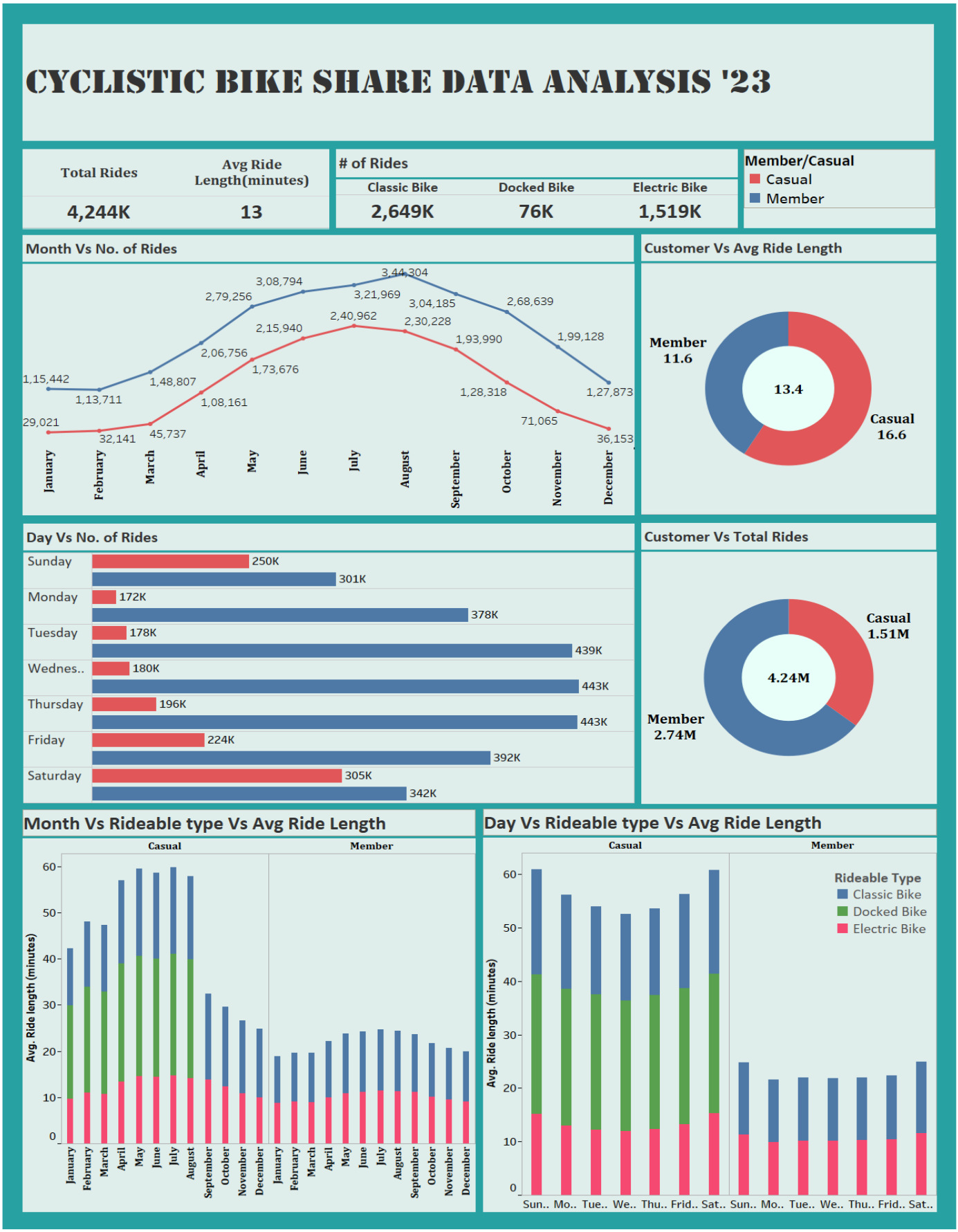
- SHARE

Visualizations are useful to share the findings and key insights of the analysis to the stake holders. Tableau is used to create the visualizations and dashboard.

Initially, the cleaned dataset is loaded into Tableau in a new workbook. Then variety of charts are created.

- KPIs are created using cards.
- Customer relationship with rides and ride length are depicted using donut charts.
- Relationship between bike types, month, day and rides are depicted using line and bar charts respectively.

The dashboard is given below:



Key Insights

- During the year 2023, a total of 4.24M rides are taken by the customers with an average ride length of 13 minutes.
- Around 75% of rides (around 2.7M) were taken by annual members, but the average ride length (11 minutes) was less than that of casual riders (16 minutes).
- There is an interesting pattern for the rides taken by both type of riders. The annual members tend to ride more on weekdays in contrast to the casual riders who rides on weekends more.
- Total number of rides are high during the spring and summer seasons of the year with a peak in months July - August.
- Classic bikes are mostly used throughout the year irrespective of the customer type which is followed by electric bikes and then docked bikes.
- Docked bikes are only used by the casual riders but it is used for the long trips instead of classic or electric bikes. At the same time, the docked bikes are used throughout a week with an average ride length of 25 minutes which is longer than the trips with other two bikes.
- Both riders use classic and electric bikes for their trips in a similar manner, but docked bikes are used in the first 3 quarters of the year.

- ACT

Conclusion

Analysing the above insights, it is evident that annual member and casual riders are mainly differed in the purpose of their rides. Annual members use the system for commute to work or studies with a steady average ride length and number of rides. On the other hand, casual riders use bikes for long trips during weekends and they mostly use docked bikes.

Recommendation

Annual members contribute largely to the ride but their average ride duration is less than that of casual riders. According to the analysis done, in order to convert the casual riders to annual members, new marketing strategy should be designed so that they can benefit more from the membership rides than the casual ones. For this, new travel plans, offers and discounts can be introduced for the annual members who uses docked bikes and long trips. Design more plans for specific group of customers such as students, senior citizens, etc. To reach out these plans to the public, more digital advertisements can be placed on public areas and bike stations.

More analysis should be done based on customer data such as age, travel destinations, etc to know more about their ride behaviour. Surveys and feedback from the customers about their ride options and plans can be also taken and analysed to plan new marketing strategies.