Analysis Report

Aryan Barnwal

2022-06-17

Overview

How Does a Bike Sharing Company Navigate Speedy Success?

Introduction

Welcome to the Cyclistic bike-share analysis project! In this project, I performed many real-world tasks of a junior data analyst.

Scenario

I am 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, the team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights,our team will design a new marketing strategy to convert casual riders into annual members.

Characters and teams:

- **Cyclistic**: A bike-share program that features more than 5,800 bicycles and 600 docking stations. Cyclistic sets itself apart by also offering reclining bikes, hand tricycles, and cargo bikes, making bike-share more inclusive to people with disabilities and riders who can't use a standard two-wheeled bike. The majority of riders opt for traditional bikes;about 8% of riders use the assistive options. Cyclistic users are more likely to ride for leisure, but about 30% use them to commute to work each day.
- **Lily Moreno**: The director of marketing and the manager. Moreno is responsible for the development of campaigns and initiatives to promote the bike-share program. These may include email, social media, and other channels.

1.Ask

The questions that the stakeholders want answer are:
1. How do annual members and casual riders use Cyclistic bikes differently?
2. Why would casual riders buy Cyclistic annual memberships?
3. How can Cyclistic use digital media to influence casual riders to become members?

From these tasks the primary objective is to identify the differences between the casual riders and annual riders.
Then use those insights to launch marketing campaigns to convert casual riders into annual riders thus maximizing profitability.

2.Prepare

For this project i will be using the ride share data from their website. The data has been made available by Motivate International Inc. under this license. There were 12 csv files containing data for the respective months. I downloaded the dataset and stored it in a folder in appropriate hierarchy.

At this point the data was ready to be processed. The point to note is that data was made available directly by the Cylistic company. The data is **Secondary Data.**

Lets start by loading the packages we will be using for the project

```
# Loading the required packages
library(tidyverse) # For Data Analysis
```

```
## -- Attaching packages --------------------------------------- tidyverse
1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.1.3

## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(dplyr) # For Data Manipulation
library(skimr) # For summarizing the data
library(janitor) # For Data Cleaning

##
## Attaching package: 'janitor'

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

library(here) # Makes referencing files easier

## here() starts at C:/Users/aryan/OneDrive/Documents

library(geosphere) # Working with Geo-Spatial Data
library(ggmap) # For plotting points on maps

## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

## Please cite ggmap if you use it! See citation("ggmap") for details.

library(lubridate) # For working with date-times.

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

library(plotly)

## Warning: package 'plotly' was built under R version 4.1.3

##
## Attaching package: 'plotly'
## The following object is masked from 'package:ggmap':
##
##     wind

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout

library(ggpubr)

## Warning: package 'ggpubr' was built under R version 4.1.3
```

Now lets load the dataset.

```
# Changing the working directory to where our data is stored
setwd("D:\\Capstone Dataset\\Cyclistic Dataset Copy")
jan_2021 <- read_csv("202101-divvy-tripdata.csv")

## Rows: 96834 Columns: 13
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.

feb_2021 <- read_csv("202102-divvy-tripdata.csv")

## Rows: 49622 Columns: 13
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```r
mar_2021 <- read_csv("202103-divvy-tripdata.csv")
```

```
## Rows: 228496 Columns: 13
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```r
apr_2021 <- read_csv("202104-divvy-tripdata.csv")
```

```
## Rows: 337230 Columns: 13
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```r
may_2021 <- read_csv("202105-divvy-tripdata.csv")
```

```
## Rows: 531633 Columns: 13
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```r
jun_2021 <- read_csv("202106-divvy-tripdata.csv")
```

```
## Rows: 729595 Columns: 13
## -- Column specification --------------------------------------------------
```

```
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
jul_2021 <- read_csv("202107-divvy-tripdata.csv")
```

```
## Rows: 822410 Columns: 13
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
aug_2021 <- read_csv("202108-divvy-tripdata.csv")
```

```
## Rows: 804352 Columns: 13
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
sep_2021 <- read_csv("202109-divvy-tripdata.csv")
```

```
## Rows: 756147 Columns: 13
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
oct_2021 <- read_csv("202110-divvy-tripdata.csv")
```

```
## Rows: 631226 Columns: 13
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
nov_2021 <- read_csv("202111-divvy-tripdata.csv")
```

```
## Rows: 359978 Columns: 13
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
dec_2021 <- read_csv("202112-divvy-tripdata.csv")
```

```
## Rows: 247540 Columns: 13
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id,
end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

Everything looks great the columns have the correct data types

3.Process

The tools used for this project are **R-Studio(2022.02.3 Build 492)** *runs on* **R(Version 4.1.1)** *and* **Tableau.** *R was chosen because it can handle and process larger datasets much quicker as compared to spreadsheets. It can also visualize and create well-formatted reports, complete with code chunks and pictures.*

Tableau *in this case was used for creating some visualizations since it is a great tool to create detailed visualizations.*

Lets check if all the dataframes have same column names

```
# Code Chunk: 3.1: Checking if all the dataframes have same column names
if (colnames(jan_2021) == colnames(feb_2021) && colnames(feb_2021) ==
colnames(mar_2021) && colnames(mar_2021) == colnames(apr_2021) &&
colnames(apr_2021) == colnames(may_2021) && colnames(may_2021) ==
colnames(jun_2021) && colnames(jun_2021) == colnames(jul_2021) &&
colnames(jul_2021) == colnames(aug_2021) && colnames(aug_2021) ==
colnames(sep_2021) && colnames(sep_2021) == colnames(oct_2021) &&
colnames(oct_2021) == colnames(nov_2021) && colnames(nov_2021) ==
colnames(dec_2021)) {
  print("All column names are same")
}

## [1] "All column names are same"
```

Now lets combine all the dataframes into a single dataframe by using row bind

```
# Code Chunk 3.2: Binding all the dataframes into a single dataframe.
rides_2021 <-
rbind(jan_2021,feb_2021,mar_2021,apr_2021,may_2021,jun_2021,jul_2021,aug_2021
,sep_2021,oct_2021,nov_2021,dec_2021)
```

Now lets print the first few rows our data by using glimpse() function.

```
# Code Chunk 3.3
glimpse(rides_2021)

## Rows: 5,595,063
## Columns: 13
## $ ride_id            <chr> "E19E6F1B8D4C42ED", "DC88F20C2C55F27F",
"EC45C94683~
## $ rideable_type      <chr> "electric_bike", "electric_bike",
"electric_bike", ~
## $ started_at         <dttm> 2021-01-23 16:14:19, 2021-01-27 18:43:08,
2021-01-~
## $ ended_at           <dttm> 2021-01-23 16:24:44, 2021-01-27 18:47:12,
2021-01-~
## $ start_station_name <chr> "California Ave & Cortez St", "California Ave &
Cor~
## $ start_station_id   <chr> "17660", "17660", "17660", "17660", "17660",
"17660~
## $ end_station_name   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, "Wood St & Augu~
## $ end_station_id     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, "657", "13258",~
## $ start_lat          <dbl> 41.90034, 41.90033, 41.90031, 41.90040, 41.90033, 4~
## $ start_lng          <dbl> -87.69674, -87.69671, -87.69664, -87.69666, -87.696~
## $ end_lat            <dbl> 41.89000, 41.90000, 41.90000, 41.92000, 41.90000, 4~
## $ end_lng            <dbl> -87.72000, -87.69000, -87.70000, -87.69000, -87.700~
## $ member_casual      <chr> "member", "member", "member", "member", "casual", "~
```

One issue here is that we can see that there are NA values in some columns, we will look into it in the later part of our analysis.

Except that everything looks good, all the columns have the correct data type except for the columns **"rideable_type"** *and* **"member_casual"**. *These columns contain categorical data and can be converted to factor type to save some space.*

```
# Code Chunk 3.4: Converting two columns data types to factor type instead of character
rides_2021 <- rides_2021 %>%
  mutate(rideable_type = as.factor(rideable_type)) %>%
  mutate(member_casual = as.factor(member_casual))
```

3.1 Data Integrity Checkups

3.1.1 Check for Duplicate Rows

Checking for duplicate rows should be one of the first tasks in data processing.
However the nature of the dataset allows for duplicate values in all but the **"ride_id"** *column.*
Each "ride_id" represents a unique trip.
In this case, the red flag appears when there are duplicate **"ride_id"** *the following code chunk checked for that.*

```
# Code Chunk 3.5:   Checking for duplicate ride_ids
rides_2021 %>%
  group_by(ride_id) %>%
  filter(n() > 1)

## # A tibble: 0 x 13
## # Groups:   ride_id [0]
## # ... with 13 variables: ride_id <chr>, rideable_type <fct>, started_at <dttm>,
## #   ended_at <dttm>, 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 <fct>
```

Luckily there are no duplicate **"ride_id"**.

3.1.2 Handling NAs

Handling NAs is one of the most inevitable tasks while working with data they can be generated due to many issues such as missing data, data entry errors, etc.

As seen earlier while running the code chunk 3.3 there are some missing values in the columns **"end_station_name", "end_station_id".**

Now lets use the skim function to find which columns have missing values.

```
# Code Chunk 3.6:
skim(rides_2021)
```

Data summary

Name	rides_2021
Number of rows	5595063
Number of columns	13
_______________________	
Column type frequency:	
character	5
factor	2
numeric	4
POSIXct	2
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ride_id	0	1.00	16	16	0	5595063	0
start_station_name	690809	0.88	3	53	0	847	0
start_station_id	690806	0.88	3	36	0	834	0
end_station_name	739170	0.87	10	53	0	844	0
end_station_id	739170	0.87	3	36	0	832	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
rideable_type	0	1	FALSE	3	cla: 3251028, ele:

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
					2031692, doc: 312343
member_casual	0	1	FALSE	2	mem: 3066058, cas: 2529005

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
start_lat	0	1	41.90	0.05	41.64	41.88	41.90	41.93	42.07	▁▁▇▇▁
start_lng	0	1	-87.65	0.03	-87.84	-87.66	-87.64	-87.63	-87.52	▁▁▆▇▁
end_lat	4771	1	41.90	0.05	41.39	41.88	41.90	41.93	42.17	▁▁▁▇▁
end_lng	4771	1	-87.65	0.03	-88.97	-87.66	-87.64	-87.63	-87.49	▁▁▁▁▇

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
started_at	0	1	2021-01-01 00:02:05	2021-12-31 23:59:48	2021-08-01 01:52:11	4677998
ended_at	0	1	2021-01-01 00:08:39	2022-01-03 17:32:18	2021-08-01 02:21:55	4671372

From the above output we can see that the columns **"start_station_name", "start_station_id", "end_station_name","end_station_id", "end_lat" and "end_lng"** have missing values.

Missing end_station_id or end_station_name means that some of the riders do not return the bikes at stations.
Missing start_station_name or start_station_id means that the rides are not starting from stations.

Another interesting observation is that there are missing values in the **"end_lat"** and **"end_lng"** columns which might mean that the locating device in the bike ran out of power midway during the trip. This can be informed to the technical team. These constitute around **0.07%** of the total observations. These values will be removed.

```
# Code Chunk 3.7: Removing records with missing location information
rides_2021 <- rides_2021 %>%
  filter(!(is.na(end_lat))) %>%
  filter(!(is.na(end_lng)))
```

Although I won't be removing rides containing missing end or start station names beacuse: .
They constitute of around **13.2 %** of total observations, removing them will result in a loss of a lot of observations.
. Geo-Spatial information can be used to track those rides.
These observations containing NAs could reveal more insights about customer behaviors and requirements. I will be doing that in the analysis step of the analysis.

3.2 Creating Derived columns for Analysis.

3.2.1 Creating a derived column Ride_duration which shows the ride duration for a particular ride.

```
# Code Chunk 3.7: Adding a column ride duration
rides_2021 <- rides_2021 %>%
  mutate(ride_duration = difftime(ended_at,started_at,units = "mins")) %>%
# Converting ride_duration to double data type
  mutate(ride_duration = as.double(ride_duration))
```

Now lets perform some **Descriptive Statitics** on the **"ride_duration"** column.

```
# Code Chunk 3.8
summary(rides_2021$ride_duration)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   -58.03     6.73    11.98    20.98    21.73 55944.15
```

Looks like there are some negative values in the ride duration column which is not possible. Lets have a look at those rides.

```
# Code Chunk 3.9
rides_2021 %>%
  filter(ride_duration < 0)

## # A tibble: 146 x 14
##    ride_id          rideable_type started_at          ended_at
##    <chr>            <fct>         <dttm>              <dttm>
##  1 B1235D38EB2F8A9E electric_bike 2021-01-06 18:33:12 2021-01-06 18:31:07
##  2 F79335E3A77A57B5 electric_bike 2021-03-29 15:41:21 2021-03-29 15:41:20
##  3 5D2797A8FFA71B49 classic_bike  2021-03-13 18:02:58 2021-03-13 18:02:57
##  4 BC53ECCBC76278FD classic_bike  2021-04-07 16:11:33 2021-04-07 16:11:26
##  5 209C097828F9CD43 electric_bike 2021-04-27 17:13:44 2021-04-27 17:11:32
##  6 6E81034B446FC2FD electric_bike 2021-04-23 09:43:39 2021-04-23 09:43:29
##  7 318DD838369AEA61 classic_bike  2021-04-30 10:56:32 2021-04-30 10:56:30
##  8 8ADD13BD8F6A7567 classic_bike  2021-04-17 12:43:36 2021-04-17 12:43:27
##  9 3EC1B5A4D4B9AB99 classic_bike  2021-05-05 16:10:04 2021-05-05 16:09:51
## 10 518535DDFA372694 electric_bike 2021-05-20 12:31:53 2021-05-20 12:31:52
## # ... with 136 more rows, and 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 <fct>, ride_duration <dbl>
```

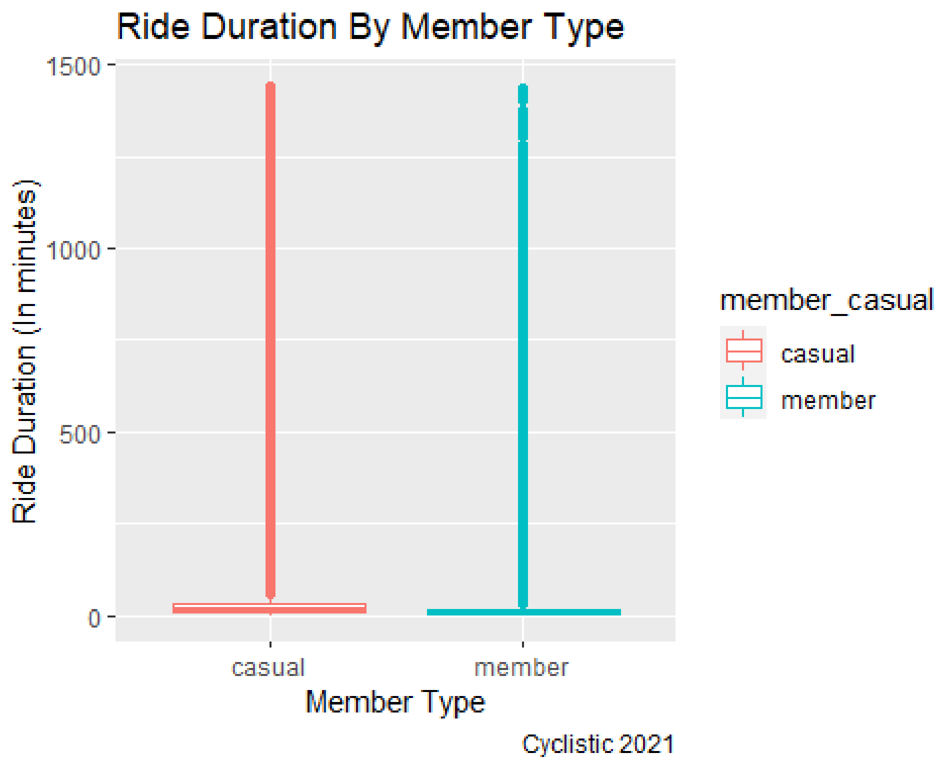
This might have happened due to some error but we will be removing those rows with negative **"ride_duration".**

```
# Code Chunk 3.10: Removing rows with negative ride duration
rides_2021 <- rides_2021 %>%
  filter(ride_duration > 0)
```

Also the maximum value for the ride duration is 559944 minutes which is around 39 days. If we look at the current pricing of Cyclistic a single day pass costs around $15 such long trip would cost him $585 which is higher than the cost of a new bike.

These large values can skew the dataset and tends to deviate mean. Let's plot a box plot for the ride_duration column for the two member types.

```
## Warning: Removed 1333 rows containing non-finite values (stat_boxplot).
```



Plot 3.1 Box Plot of Ride Duration by Member Type.

We can see that there are a lot of outliers in the **"ride_duration"** column specially for casual riders. There are different methods to remove outliers such as the **IQR(Inter quaaertile range)** method.

With the current context using that method would be senseless since it is possible that some rides can be of longer duration. But I will be assuming that for rides with ride duration greater than 24 hours the bike is stolen.

Also according to Cyclistic any rides with ride duration less than 1 minute is either rider re-docking the bike to make sure its safely docke or the Cyclistic team testing the bikes.

So lets remove rides with ride duration less than 1 minute and greater than 24 hours.

```
# Code Chunk 11: Removing rides with ride duration less than 1 minute and
greater than 24 hours.
rides_2021_v1 <- rides_2021 %>%
  filter(ride_duration >= 1) %>%
  filter(ride_duration <= 1440)
```

Now lets again have a look at the summary of the **"ride_duration"** *column.*

```
# Code Chunk 3.12: Printing summary of the ride duration column
summary(rides_2021_v1$ride_duration)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    6.95   12.18   19.68   21.97 1439.85
```

Everything looks great now.

The current data is less **granular** that is it by **"Date and time"** only, lets add columns for **day, Month, day type and hour of the day** to increase its **granularity.**

Adding granularity to our dataset help us identify trends much easier with respect to different levels of granularity.

3.2.2 Ceating Derived Columns for Hour of the day, Day and Day Type.

```
# Code Chunk 3.13: Creating columns Hour, Day and Day_Type
rides_2021_v1 <- rides_2021_v1 %>%
  mutate(hour = hour(started_at)) %>%
  mutate(day = wday(started_at, label = TRUE)) %>%
  mutate(day_type = if_else(day == "Sun" | day == "Sat", "Weekend",
"Weekday"))
```

Now lets calculate "ride_length" for every trip using the distGeo() Function. Its documentation can be found here

3.2.3 Creating Derived Column Ride Length

```
# Code Chunk 3.14: Calculating Ride Length for Each Trip
rides_2021_v1 <- rides_2021_v1 %>%
  mutate(ride_length = distGeo(matrix(c(start_lng, start_lat), ncol = 2),
matrix(c(end_lng, end_lat), ncol =2))) %>%
  mutate(ride_length = ride_length/1000)
```

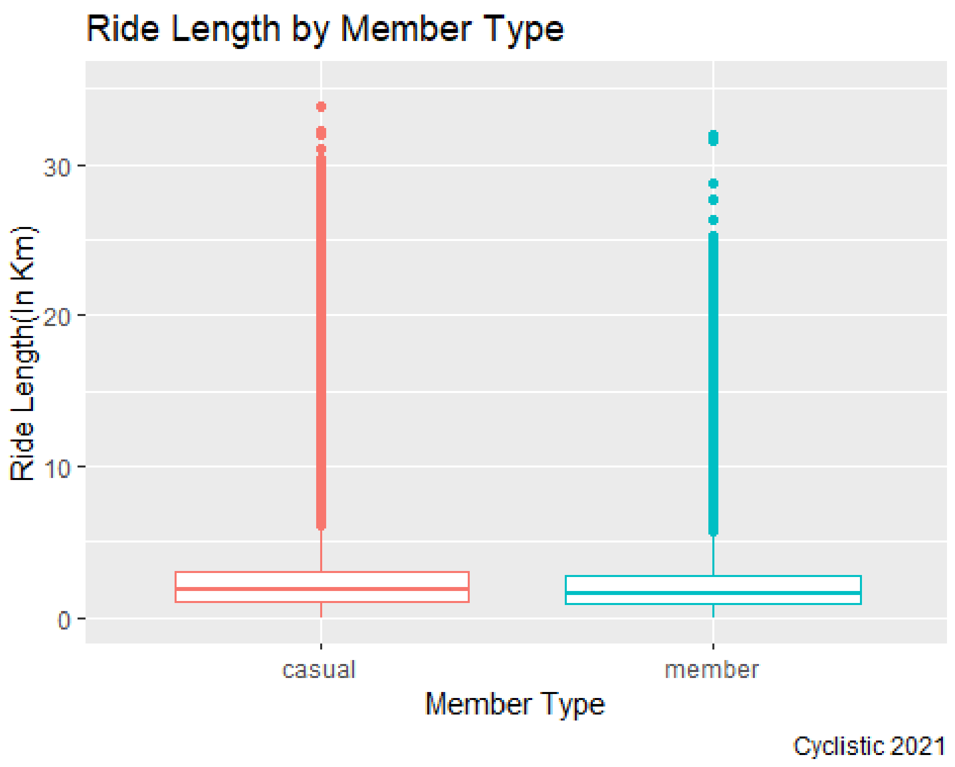
Now lets perform **Descriptive Statistics** on the **"ride_length"** column.

```
# Code Chunk 3.15: Summarising the ride_length column
summary(rides_2021_v1$ride_length)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
##   0.0000   0.9416   1.6594   2.2199   2.9110 114.5674
```

Lets plot a boxplot of **"ride length"** for casual and annual riders.

```
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```



Plot 3.2 Box Plot of Ride Distance by Member Type.

Looks like casual and annual riders have approximately identical **"ride duration"**

At this point, the resulting data was comprehensive with numerical, positional, categorical and chronological data.

The next step of data analysis can finally be commenced.

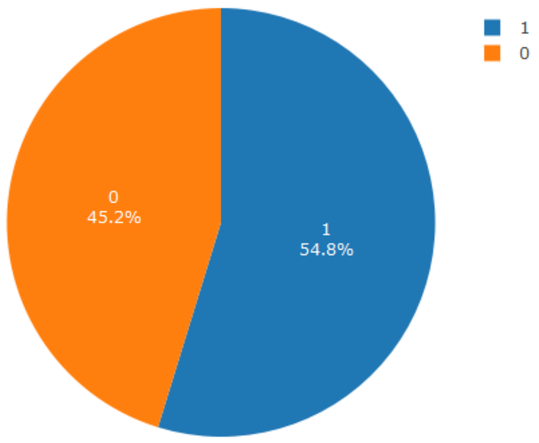
4 Analyze

In this step of the process I will try to look for trends & patterns in the data to gather insights and share it to the stakeholders.

4.1 Analyzing Data by Number of Rides.

Lets plot a pie chart to show the number of rides taken by the different category of riders.

```
## Warning: 'pie' objects don't have these attributes: 'categories'
## Valid attributes include:
## '_deprecated', 'automargin', 'customdata', 'customdatasrc', 'direction',
'dlabel', 'domain', 'hole', 'hoverinfo', 'hoverinfosrc', 'hoverlabel',
'hovertemplate', 'hovertemplatesrc', 'hovertext', 'hovertextsrc', 'ids',
'idssrc', 'insidetextfont', 'insidetextorientation', 'label0', 'labels',
'labelssrc', 'legendgroup', 'legendgrouptitle', 'legendrank', 'marker',
'meta', 'metasrc', 'name', 'opacity', 'outsidetextfont', 'pull', 'pullsrc',
'rotation', 'scalegroup', 'showlegend', 'sort', 'stream', 'text', 'textfont',
'textinfo', 'textposition', 'textpositionsrc', 'textsrc', 'texttemplate',
'texttemplatesrc', 'title', 'transforms', 'type', 'uid', 'uirevision',
'values', 'valuessrc', 'visible', 'key', 'set', 'frame', 'transforms',
'_isNestedKey', '_isSimpleKey', '_isGraticule', '_bbox'
```



Plot 4.1 A Pie chart showing the number of riders taken by different group of riders

Annual riders ride more than the casual riders which is a good thing.

4.1.1 Number of Rides in different Locations of the city.

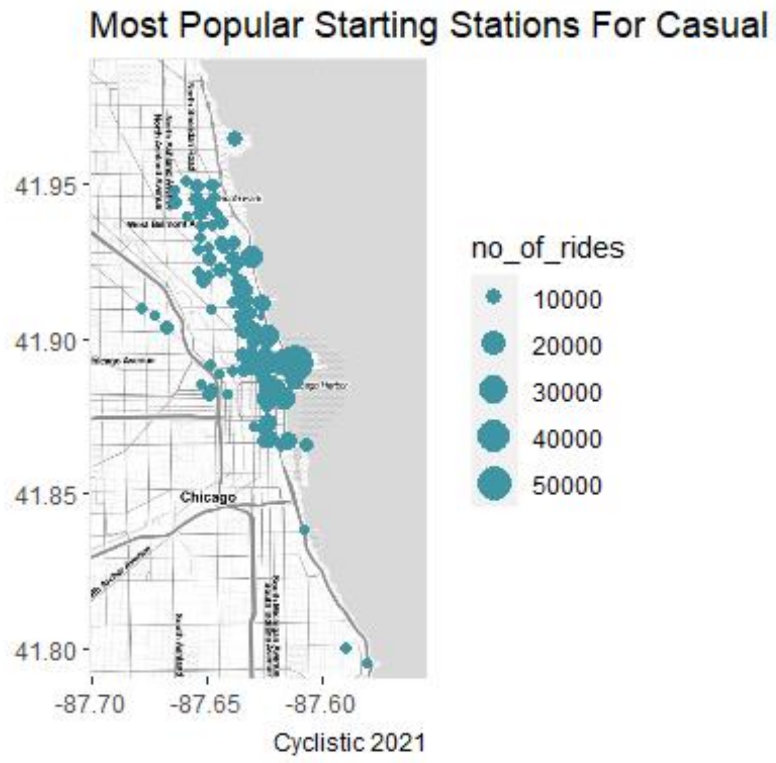
Note: One limitation of the dataset is that we have information only about the starting and ending coordinates of the rides but not about the path taken to reach that point.

Now, lets use the geospatial data to gather some insights. First start by loading the map of Chicago so that it can be used at later stage.

Plotting the top 100 most popular stations among different categories of riders on the map of Chicago.

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

## Warning: Removed 1 rows containing missing values (geom_point).
```



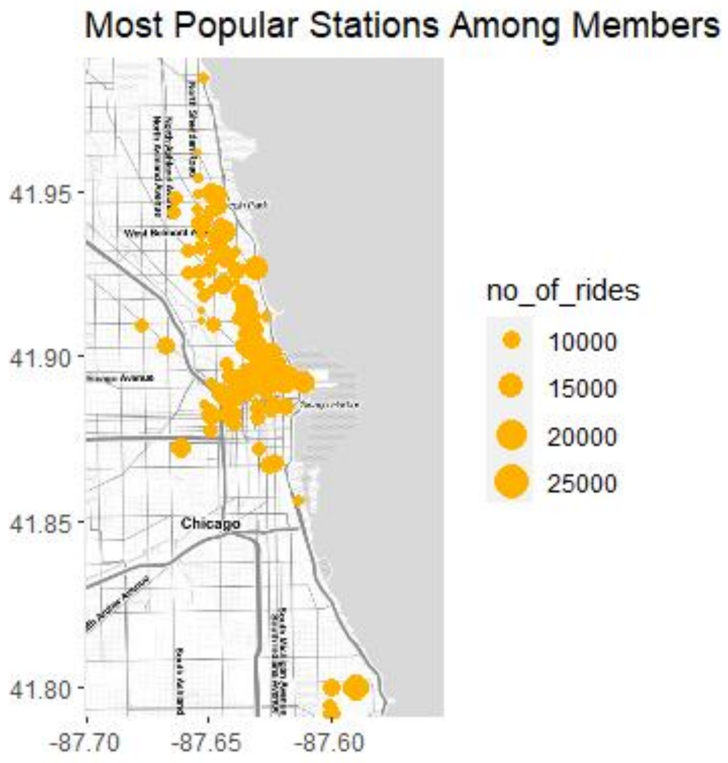
Plot 4.2 Most Popular Start Stations among Casual Riders

Casual riders tend to use the stations which are closer to the Chicago Harbor. These are the places where most of the leisure activities happen.

Lets now do the same for Annual riders

```
## `summarise()` has grouped output by 'start_lng', 'start_lat',
## 'start_station_name'. You can override using the `.groups` argument.

## Warning: Removed 3 rows containing missing values (geom_point).
```



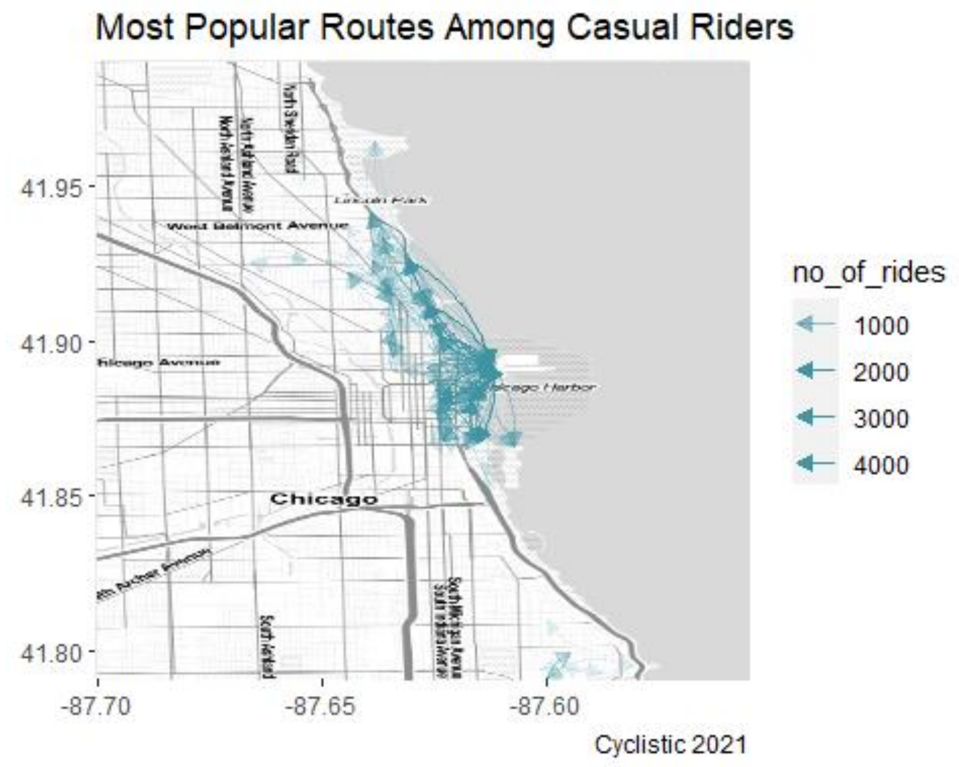
Plot 4.3 Most Popular Starting Stations among Annual riders.

For Annual riders the most used starting stations are more concentrated on the insides of the city, these are the places where most of the corporate offices are located.

Now lets plot the most popular routes among **Casual.**

```
## `summarise()` has grouped output by 'start_lng', 'start_lat', 'end_lng',
'end_lat'. You can override using the `.groups` argument.
## Coordinate system already present. Adding new coordinate system, which
will replace the existing one.

## Warning: Removed 14 rows containing missing values (geom_curve).
```



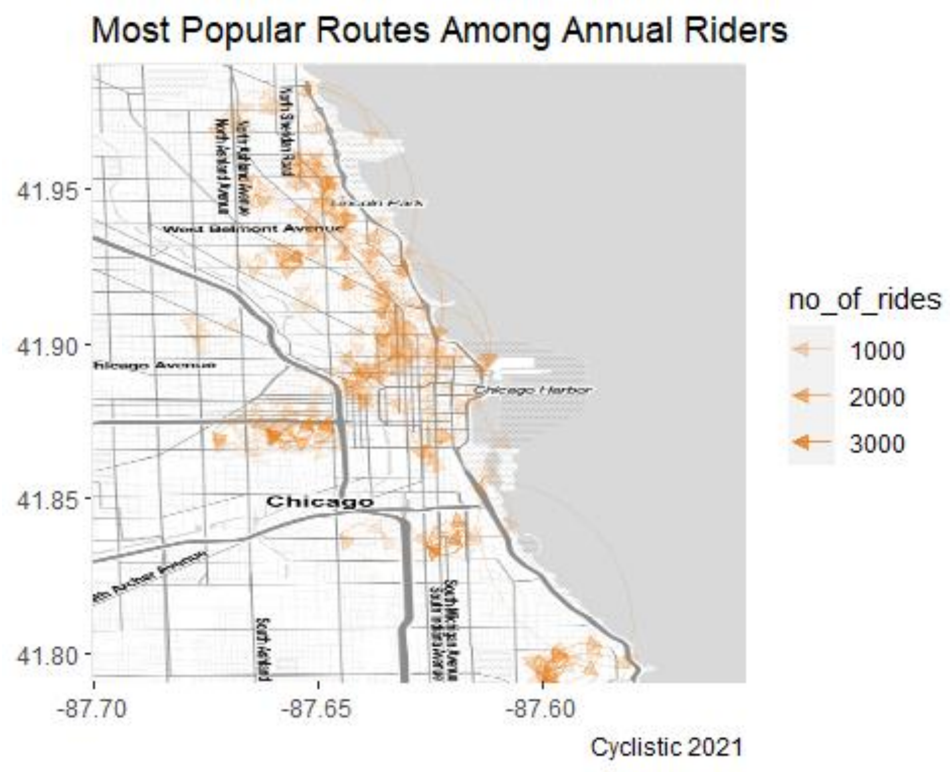
Plot 4.4 Map Showing Most Popular Routes Among Casual Riders

Casual Riders tend to take routes along the Chicago coast, these are the places where parks and chill places are located.

With the result from the above maps we can say that casual riders tend to use our bike for leisure activities.

Now lets do the same for **Annual** riders.

```
## `summarise()` has grouped output by 'start_lng', 'start_lat', 'end_lng',
'end_lat'. You can override using the `.groups` argument.
## Coordinate system already present. Adding new coordinate system, which
will replace the existing one.
```



Plot 4.5 Map Showing Most Popular Routes among Annual Riders

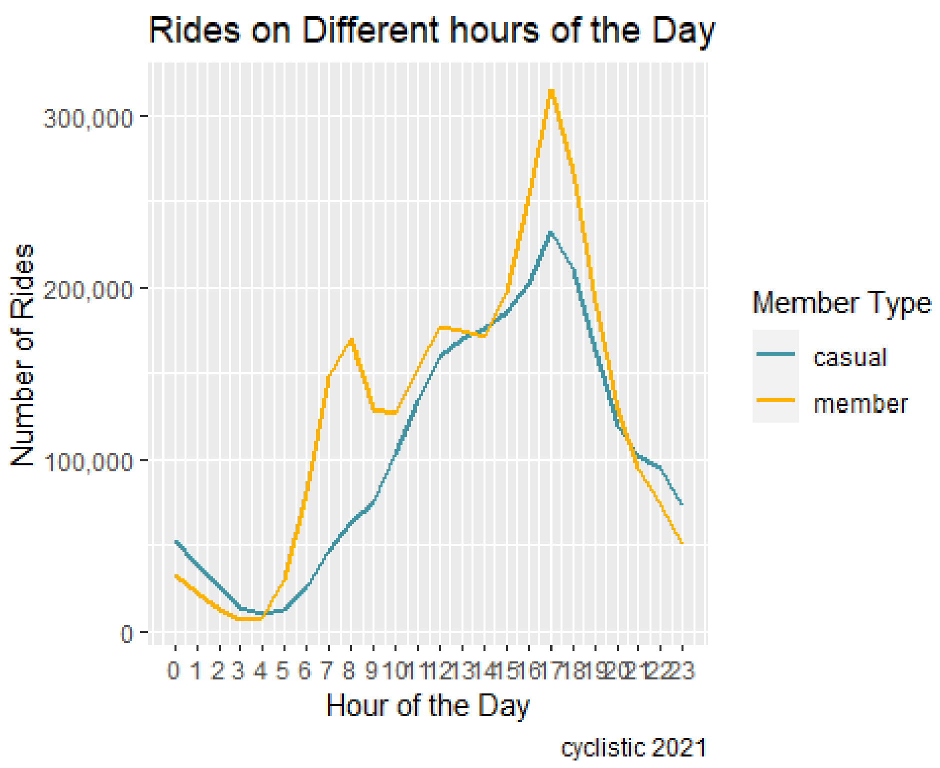
Annual riders tend to take routes which are more in the inside of the city. These are the places where most corporate buildings are located.

4.2 Number of Rides by Different Time Frames.

4.2.1 Number of Rides by Hour of the Day

```
# Code Chunk 4.7: Number of rides on Different times of the day.
rides_2021_v1 %>%
  group_by(hour, member_casual) %>%
  summarize(no_of_rides = n()) %>%
  ggplot(aes(x = hour, y = no_of_rides)) +
  geom_line(aes(color = member_casual), size = 1) +
  scale_y_continuous(name = "Number of Rides", label = scales::comma) +
  scale_x_continuous(name = "Hour of the Day", labels = 0:23, breaks = 0:23)
+
  labs(title = "Rides on Different hours of the Day", caption = "cyclistic
2021") +
  scale_color_manual(values = c(casual = "#4095A5", member = "#FFB100")) +
  guides(col=guide_legend("Member Type")) +
  theme_gray()

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



Plot 4.6 Plot showing Number of Rides at Different Times of the Day

We can say that most of the ***Annual Riders*** use our bikes to commute to work because of the following insights:

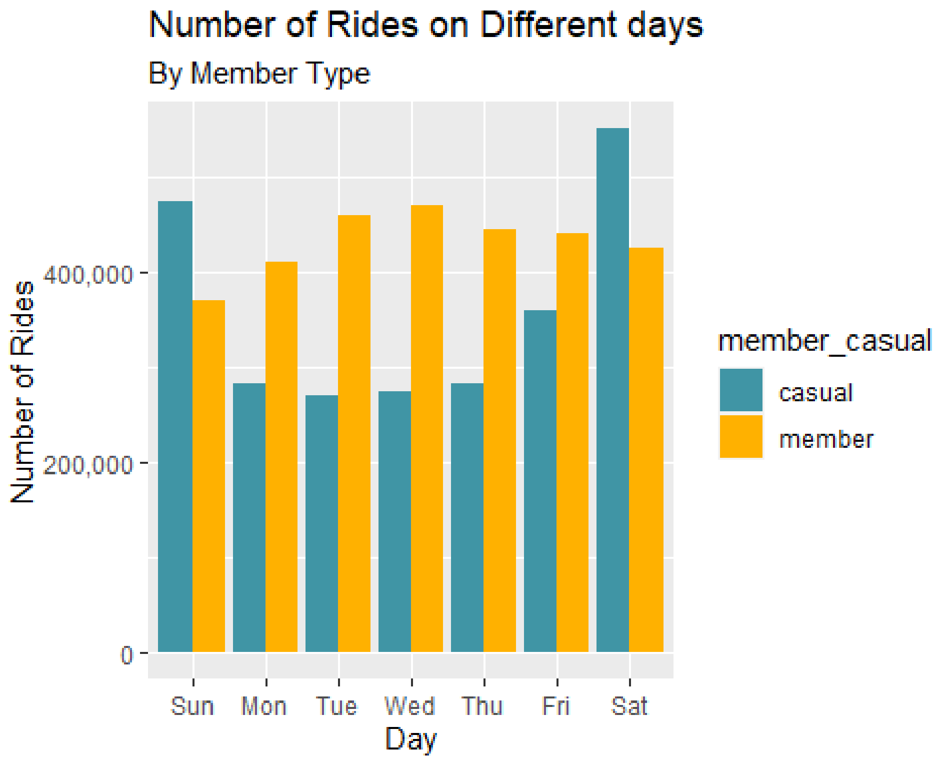
1. The number of rides starts to increase in the morning and peaks at around 08:00 A.M, this is the time when people generally leave for work.
2. The number of rides then decrease from there this is the time when people are working at their offices. + Then the number of rides starts to increase from 11:00 A.M and peaks at around 12:00 P.M, this is the time when corporate have their lunch break.
3. Then the number of rides decrease till 03:00 P.M after which the rides starts to increase and peaks at around 05:00 P.M this is the time when people are free from their work.

4.2.2 Number of Rides on Different Days of the Week.

Now let's have a look at the number of rides on different days of the week.

```
# Code Chunk 4.7
# Calculating ridership on different days of the week.
rides_2021_v1 %>%
  group_by(day, member_casual) %>%
  summarize(no_of_rides = n()) %>%
  ggplot(aes(x = day, y = no_of_rides)) +
  geom_col(aes(fill = member_casual), position = "dodge") +
  scale_y_continuous(name = "Number of Rides", label = scales::comma) +
  labs(title = "Number of Rides on Different days", subtitle = "By Member
Type", x = "Day") +
  scale_fill_manual(values = c("#4095A5", "#FFB100")) +
  guides(col=guide_legend("Member Type")) +
  theme_gray()

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



Plot 4.7 Plot showing Number of Rides on Different Days of the Week.

Members have almost the same ridership throughout the week .

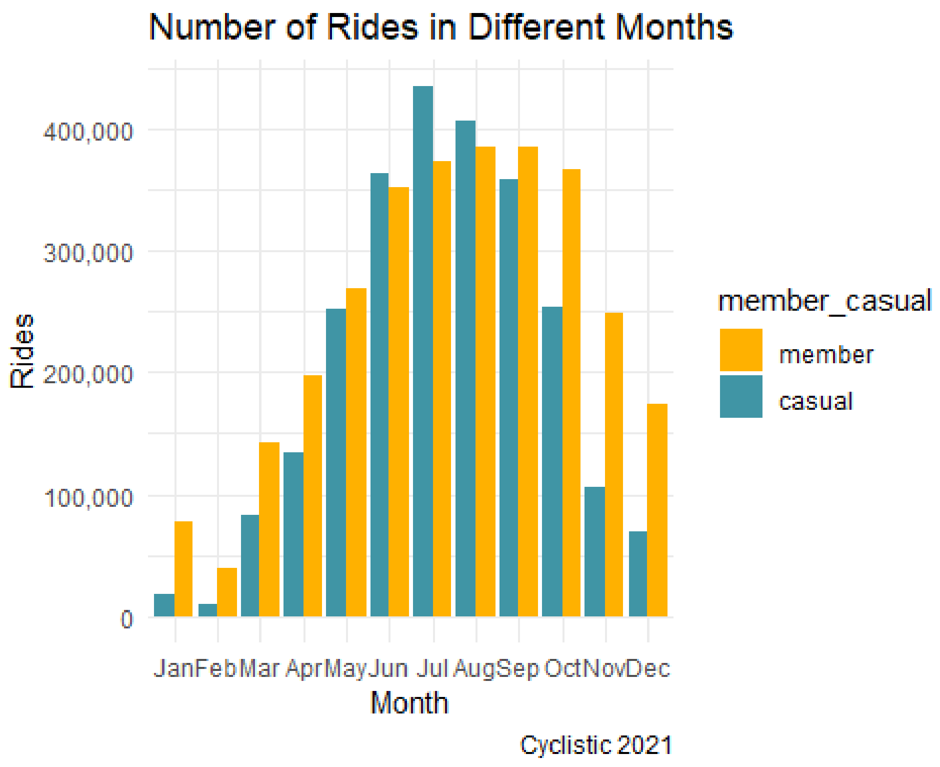
While Casual riders prefer to ride more in the weekends. On weekdays the number of rides by Casual riders is almost 40% of the rides taken by Annual riders. But on weekends Casual riders surpass the Annual riders in terms of number of rides.

Now lets try to visualize ridership by different months of the year.

4.2.3 Number of Rides on different Months of the Year.

```
# Code Chunk 4.8: Plot showing number of Rides on different parts of the
year.
rides_2021_v1 %>%
  group_by(month = month(started_at, label = TRUE), member_casual) %>%
  summarize(no_of_rides = n()) %>%
  ggplot(aes(x = month, y = no_of_rides)) +
  geom_col(aes(fill = member_casual), position = "dodge") +
  scale_y_continuous(name = "Rides", label = scales::comma) +
  labs(title = "Number of Rides in Different Months",  caption = "Cyclistic
2021", x = "Month", y = "Number of Rides") +
  scale_fill_manual(values = c(member = "#ffb100", casual = "#4095a5")) +
  theme_minimal() +
  guides(col=guide_legend("Member Type"))

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



Plot 4.8 Plot Showing Number of rides on Different Months of the Year.

We can clearly see a seasonal trend here. People like to ride our bicycles mostly in the summer season.

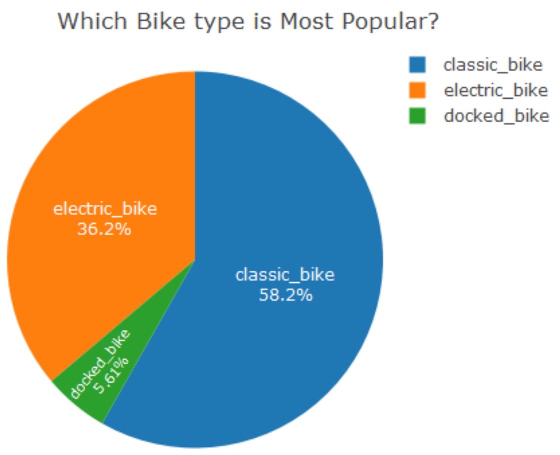
We have the lowest ridership in the months of December, January, February and March that is in the winter season.

These insights can be used to launch marketing campaigns since, we have most ridership in the summer season. Marketing team can launch marketing campaigns in this part of the year and try to maximize the profit.

Note: Since we don't have historical data, we can't compare data to previous years. Also we can't get an understanding of how *COVID* has affected our business.

4.3 Number of Rides by Bike Type.

```
# Code Chunk 4.9 Pie Chart of Number of Rides by Bike Type.
rides_2021_v1 %>%
  group_by(rideable_type) %>%
  summarize(number_of_rides = n()) %>%
  plot_ly(labels = ~rideable_type, values = ~number_of_rides, type =
"pie",textposition = 'inside',
        textinfo = 'label+percent',
        insidetextfont = list(color = '#FFFFFF')) %>%
  layout(title = "Which Bike type is Most Popular?")
```



Plot 4.9 Pie Chart Showing the popularity of the Three Bike Types.

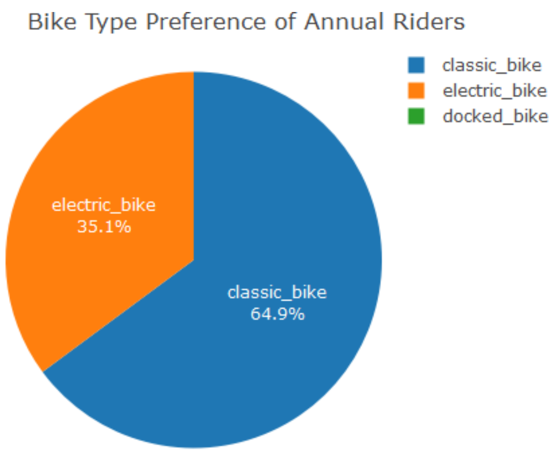
Classic bikes are the most used followed by the **Electric Bikes**. **Docked bikes** are least popular contributing to only around **5.68%** of the total rides.

4.3.1 Which kind of members prefer which bikes?

Now lets try to understand about the preferences of the two categories of members when it come to bike type.

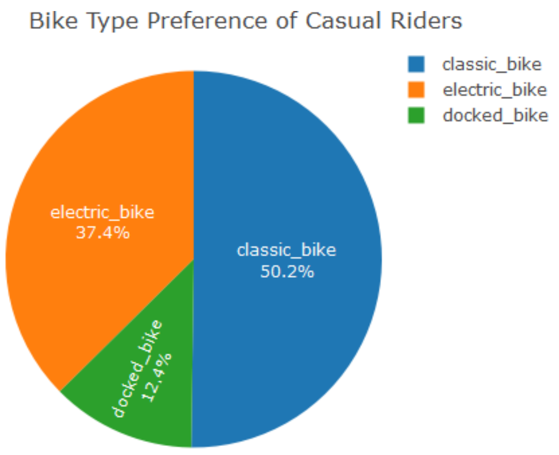
```
# Code Chunk 4.10: Bike Type Preference of different category of Riders
rides_2021_v1 %>%
  filter(member_casual == "member") %>%
  group_by(member_casual, rideable_type) %>%
  summarize(Number_of_rides = n()) %>%
  plot_ly(labels = ~rideable_type, values = ~Number_of_rides, type =
"pie",textposition = 'inside',
        textinfo = 'label+percent',
        insidetextfont = list(color = '#FFFFFF')) %>%
  layout(title = "Bike Type Preference of Annual Riders")

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



```
rides_2021_v1 %>%
  filter(member_casual == "casual") %>%
  group_by(member_casual, rideable_type) %>%
  summarize(Number_of_rides = n()) %>%
  plot_ly(labels = ~rideable_type, values = ~Number_of_rides, type =
"pie",textposition = 'inside',
        textinfo = 'label+percent',
        insidetextfont = list(color = '#FFFFFF')) %>%
  layout(title = "Bike Type Preference of Casual Riders")

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



Plot 4.10 Pie Chart Showing Bike Type Preference of the two Rider Types.

Annual Riders prefer **Classic bikes** followed by the **Electric bikes.** One thing to note is that **Annual Riders** don't use docked bikes at all.

Casual Riders also prefer **Classic bikes** followed by the **Electric bikes** but the difference is not that huge as seen in the case of **Annual Riders.**

To sum up if the organization is planning to increase their fleet of bikes then they should focus on adding more classic bikes to it.

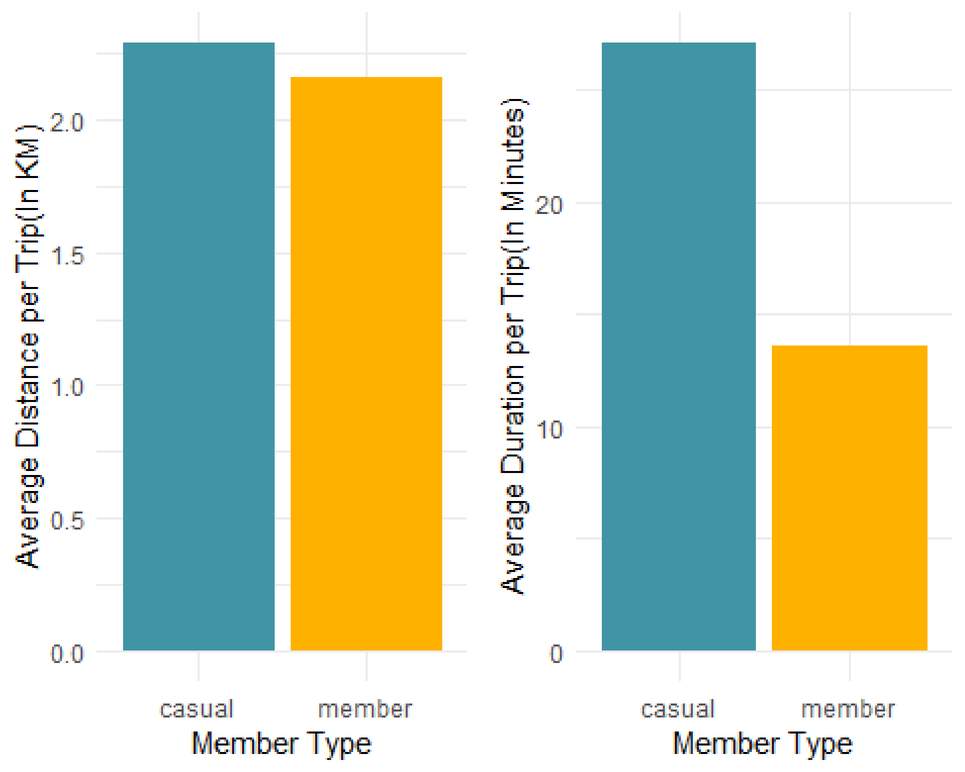
4.4 Trends in Ride Duration and Ride Length

Now lets have a look at the average ride duration and ride length for casual and annual riders.

```
distance <- rides_2021_v1 %>%
  group_by(member_casual) %>%
  summarize(average_ride_length = mean(ride_length)) %>%
  ggplot(aes(x = member_casual, y = average_ride_length, fill =
member_casual)) +
  geom_col() +
  labs(x = "Member Type", y = "Average Distance per Trip(In KM)") +
  scale_fill_manual(values = c(member = "#ffb100", casual = "#4095a5")) +
  theme_minimal() +
  theme(legend.position = "none")

duration <- rides_2021_v1 %>%
  group_by(member_casual) %>%
  summarize(average_ride_duration = mean(ride_duration)) %>%
  ggplot(aes(x = member_casual, y = average_ride_duration, fill =
member_casual)) +
  geom_col() +
  labs(x = "Member Type", y = "Average Duration per Trip(In Minutes)") +
  scale_fill_manual(values = c(member = "#ffb100", casual = "#4095a5")) +
  theme_minimal() +
  theme(legend.position = "none")

ggarrange(distance, duration, ncol = 2, nrow =1)
```



Plot 4.11 Bar Plot showing the Average Ride Duration and Average Ride Distance.**

The average Distance covered by both **Casual** and **Annual** riders are comparable but **Casual** riders takes **almost double the time** to cover the same distance which may be due to many reasons such as they have more time or they do more recreational activities.

5. Share

The deliverable for this project is a **Powerpoint Presentation** file with visualizations and short but crisp explanations about insights obtained from them. The presentation can be accessed here.

6. Act

Insights Summary

Casual riders use our bikes mostly for recreational purposes. They ride mainly along the routes closer to the Chicago Harbor. Since, casual riders have no time constraints they take 40% more time as compared to **Annual** riders to complete the trip.

Casual riders use our services the most on the weekends. **Annual** riders ridership remains almost steady throughout the week. **Casual** and **Annual** both types of riders perfer to use our services in the Summer months.

Recommendations

With the current data and insights we know that how Casual and Annual riders use our services but we don't know about why they do so in that way. So going with an conversion story right now is going to be risky.

Suppose there is a Casual user Clara she visited Chicago as a tourist and used our bike to commute to different parts of the city. Now since she doesn't live in Chicago why would she even buy our Annual pass. So to launch marketting campaigns that can convert Clara we need to know some more information about her such as Demographic information, her age, her gender.

There are two paths that I am going to recommend to stakeholders:

1. Reiterating

1.1 Changing The Business Objective

Since, the current data we have is not sufficient to answer the business objective. We can wait and tell the stakeholders about how the current data is not sufficient to answer the business objective. One way to move forward is changing the business objective.

1.2 Collecting more information about user demographic(While ensuring user privacy)

Sample and collect more data about user demographics such as their locations, age, gender and their income.

2. Forging Ahead

One way to advance in this project is to move forward with whatever information we have.

My top three recommendations for this path:

With the current scenario launching a whole marketing campaign can cost us money and resources, but still we can make some changes with the insights we have.

2.1 Introducing a weekend only pass.

Currently we have only two kind of passes, **a daily or hourly pass** and **Annual Pass** there is a huge gap between our offerings. **If we can introduce something in between such as a Weekly pass that would be a win-win situation for both of us** since, we already know from our insights that casual riders nearly doubles in the weekends.

2.2 Introducing a 6-month pass.

Since, Casual riders mostly use our services in the summer season, introducing a six months pass would be the best way to attract the Casual riders into buying our passes.

2.3 Giving some special priviledges to Annual riders.

Currently **Casual** and **Annual** riders enjoy the same freedom and services but **if we can give some special privilege to Annual riders such as making sure during peak hours Annual riders can book their bikes at a particular station from their app so that they don’t have to worry about bike availability this can motivate regular casual users to buy our passes.**