In [ ]:

```python
from IPython.display import Image
Image("img/picture.jpeg")
```

Out[ ]:

1. Introduction

About the Company In 2016, Cyclistic launched a successful bike-share offering. Since then, the program has grown to a fleet of 5,824 bicycles that are geotracked 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, Cyclistic’s 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 Cyclistic member.

Business Understanding Cyclistic’s finance analysts have concluded that annual members are much more profitable than casual riders. Although the pricing flexibility helps Cyclistic attract more customers, Moreno (Manager) believes that maximising the number of annual members will be key to future growth.

“Design marketing strategies aimed at converting casual riders into annual members “

Task Ask (Business Task) Three questions will guide the future marketing program based on goals as from data analyst:

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?

Background

The company provides the last 12 months (April 2020 — March 2021) of historical trip data for us to analyze and identify trends (The data has been made available by Motivate International Inc. under their license

Dataset Data : Cyclist's historical trip data 2021 (Apr 2021 - Mar 2022) Data Format : CSV within ZIP folder Data Licence : https://ride.divvybikes.com/data-license-agreement Data Describe : 5,723,532 obs. and 13 variables :

2. Combine Data

Importing Python libraries to complete project

In [ ]:

```
import pandas as pd # to help manipulation and visualization data
import matplotlib as mpl # to help the set up of  default color
import matplotlib.pyplot as plt # to help visualization
import matplotlib.ticker as ticker # to help the set up of the axis number
import numpy as np # to help matematical function data
```

Reading Data Frame

In [ ]:

```
data_1 = pd.read_csv('Apr_21.csv')
data_1.head()
```

Out[ ]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id
0	6C992BD37A98A63F	classic_bike	2021-04-12 18:25:36	2021-04-12 18:56:55	State St & Pearson St	TA1307000061
1	1E0145613A209000	docked_bike	2021-04-27 17:27:11	2021-04-27 18:31:29	Dorchester Ave & 49th St	KA1503000069

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id
2	E498E15508A80BAD	docked_bike	2021-04-03 12:42:45	2021-04-07 11:40:24	Loomis Blvd & 84th St	20121
3	1887262AD101C604	classic_bike	2021-04-17 09:17:42	2021-04-17 09:42:48	Honore St & Division St	TA1305000034
4	C123548CAB2A32A5	docked_bike	2021-04-03 12:42:25	2021-04-03 14:13:42	Loomis Blvd & 84th St	20121

Grouping files together cohesively into variable 'file_names'

In [ ]:

```
file_names = ['Apr_21.csv', 'May_21.csv','Jun_21.csv', 'Jul_21.csv', 'Aug_21.csv
```

Creating a map for each items that are read to correspond with one another

In [ ]:

```
all_data = list(map(pd.read_csv, file_names))
```

Confirming that all the columns match

In [ ]:

```
len(np.unique([ all_data[i].columns for i in range(12)])) == 13
```

Out[ ]:

```
True
```

In [ ]:

```
data_combine = pd.concat(all_data, ignore_index = True)
```

In [ ]:

```
sum([all_data[i].shape[0] for i in range (12)])
```

Out[ ]:

```
5723532
```

In [ ]:

```
data_combine.shape[0]
```

Out[ ]:

```
5723532
```

In [ ]:

```
data_combine.head()
```

Out[ ]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id
0	6C992BD37A98A63F	classic_bike	2021-04-12 18:25:36	2021-04-12 18:56:55	State St & Pearson St	TA1307000061
1	1E0145613A209000	docked_bike	2021-04-27 17:27:11	2021-04-27 18:31:29	Dorchester Ave & 49th St	KA1503000069

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id
2	E498E15508A80BAD	docked_bike	2021-04-03 12:42:45	2021-04-07 11:40:24	Loomis Blvd & 84th St	20121
3	1887262AD101C604	classic_bike	2021-04-17 09:17:42	2021-04-17 09:42:48	Honore St & Division St	TA1305000034
4	C123548CAB2A32A5	docked_bike	2021-04-03 12:42:25	2021-04-03 14:13:42	Loomis Blvd & 84th St	20121

3. Clean Data

In [ ]:

```
data_combine.isna().sum()
```

Out[ ]:

```
ride_id                    0
rideable_type              0
started_at                 0
ended_at                   0
start_station_name    745376
start_station_id      745373
end_station_name      796247
end_station_id        796247
start_lat                  0
start_lng                  0
end_lat                 4716
end_lng                 4716
member_casual              0
dtype: int64
```

Insuring that there is no duplicate data

In [ ]:

```
data_combine.duplicated(subset = "ride_id").sum()
```

Out[ ]:

```
0
```

Combining data from "started_at" column then converting it to datetime object

In [ ]:

```
data_combine["started_at"] = pd.to_datetime(data_combine["started_at"])
```

Combining data from "ended_at" column then converting it to datetime object

In [ ]:

```
data_combine["ended_at"] = pd.to_datetime(data_combine["ended_at"])
```

Creating category for list

In [ ]:

```
data_combine = data_combine.astype({"rideable_type":"category","member_casual":"
```

In [ ]:

```
data_combine["ride_length"] = (data_combine["ended_at"] - data_combine["started_
data_combine["month"] = data_combine["started_at"].dt.month_name().str.slice(sto
```

```
data_combine["date"] = data_combine["started_at"].dt.day
data_combine["day"] = data_combine["started_at"].dt.day_name()
data_combine["hour"] = data_combine["started_at"].dt.hour
```

In [ ]:

```
days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sun

data_combine["day"] = pd.Categorical(data_combine["day"], categories = days)
```

In [ ]:

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", 

data_combine["month"] = pd.Categorical(data_combine["month"], categories = month
```

Removing items with ride length less than 0

In [ ]:

```
data_cleaning_v2 = data_combine[data_combine["ride_length"] > 0]
```

Dropping longitude and latitude coordinates as data is irrelevant

In [ ]:

```
data_cleaning_v2 = data_cleaning_v2.drop(["start_lat","start_lng","end_lat","end
```

In [ ]:

```
data_cleaning_v2.dtypes
```

Out[ ]:

```
ride_id                        object
rideable_type                category
started_at             datetime64[ns]
ended_at               datetime64[ns]
start_station_name             object
start_station_id               object
end_station_name               object
end_station_id                 object
member_casual                category
ride_length                   float64
month                        category
date                            int64
day                          category
hour                            int64
dtype: object
```

4. Analyze Data

In [ ]:

```
data_cleaning_v2.head()
```

Out[ ]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id
0	6C992BD37A98A63F	classic_bike	2021-04-12 18:25:36	2021-04-12 18:56:55	State St & Pearson St	TA1307000061
1	1E0145613A209000	docked_bike	2021-04-27 17:27:11	2021-04-27 18:31:29	Dorchester Ave & 49th St	KA1503000069

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id
2	E498E15508A80BAD	docked_bike	2021-04-03 12:42:45	2021-04-07 11:40:24	Loomis Blvd & 84th St	20121
3	1887262AD101C604	classic_bike	2021-04-17 09:17:42	2021-04-17 09:42:48	Honore St & Division St	TA1305000034
4	C123548CAB2A32A5	docked_bike	2021-04-03 12:42:25	2021-04-03 14:13:42	Loomis Blvd & 84th St	20121

In [ ]:

```
data_cleaning_v2.describe()
```

Out[ ]:

	ride_length	date	hour
count	5.722873e+06	5.722873e+06	5.722873e+06
mean	2.154530e+01	1.538734e+01	1.422895e+01
std	1.770818e+02	8.749395e+00	5.063354e+00
min	1.666667e-02	1.000000e+00	0.000000e+00
25%	6.583333e+00	8.000000e+00	1.100000e+01
50%	1.171667e+01	1.500000e+01	1.500000e+01
75%	2.133333e+01	2.300000e+01	1.800000e+01
max	5.594415e+04	3.100000e+01	2.300000e+01

In [ ]:

```
#set the number to non scientific
pd.set_option('display.float_format', lambda x: '%.0f' % x)
```

4.1 Descriptive Analysis of Data

In [ ]:

```
data_cleaning_v2["ride_length"].agg([len,np.sum,np.mean,np.median,np.max,np.min]
```

Out[ ]:

```
len          5722873
sum        123301024
mean              22
median            12
amax           55944
amin               0
Name: ride_length, dtype: float64
```

In [ ]:

```
data_cleaning_v2.groupby("member_casual")["ride_length"].agg([len,np.sum,np.mean
```

Out[ ]:

	len	sum	mean	median	amax	amin
member_casual						
casual	2546194	80826493	32	16	55944	0

member_casual	len	sum	mean	median	amax	amin
member	3176679	42474530	13	9	1560	0

In [ ]:

```
data_cleaning_v2.groupby(["member_casual","rideable_type"]).ride_length.agg([len
```

Out[ ]:

member_casual	rideable_type	len	sum	mean	median	amax	amin
casual	classic_bike	1257512	36360641	29	16	1560	0
	docked_bike	303980	25182549	83	29	55944	0
	electric_bike	984702	19283303	20	13	487	0
member	classic_bike	1992903	27806015	14	10	1560	0
	docked_bike	NaN	0	NaN	NaN	NaN	NaN
	electric_bike	1183776	14668516	12	9	481	0

In [ ]:

```
data_cleaning_v2.groupby(["member_casual","day"])["ride_length"].agg([len,np.sum
```

Out[ ]:

member_casual	day	len	sum	mean	median	amax	amin
casual	Monday	292960	9224151	31	16	31031	0
	Tuesday	276338	7582284	27	14	38923	0
	Wednesday	286364	7952176	28	14	38963	0
	Thursday	293604	8186835	28	14	49107	0
	Friday	364237	10966051	30	15	55692	0
	Saturday	549945	18854912	34	18	55944	0
	Sunday	482746	18060083	37	19	53922	0
member	Monday	439405	5698759	13	9	1500	0
	Tuesday	490059	6136916	13	9	1500	0
	Wednesday	499862	6293469	13	9	1500	0
	Thursday	475298	5975356	13	9	1500	0
	Friday	453072	5953654	13	9	1500	0
	Saturday	431302	6467037	15	11	1560	0
	Sunday	387681	5949340	15	11	1500	0

5. Share / Data Visualization

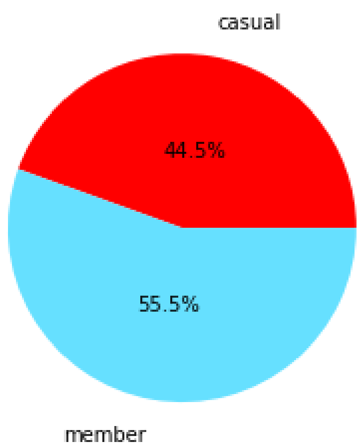
In [ ]:

```python
#set up default color
mpl.rcParams['axes.prop_cycle'] = mpl.cycler(color=["#ff0000", "#66e0ff"])
```

5.1 Membership on piegraph

In [ ]:

```python
plt.pie(data_cleaning_v2["member_casual"].value_counts(ascending = True), autopc
plt.show()
```



In [ ]:

```python
data_cm= data_cleaning_v2.groupby("member_casual")["ride_length"].agg([len,np.su
```

5.2 Membership on bargraph

In [ ]:

```python
fig, ax = plt.subplots(figsize = (8,0.5))
ax.barh(" ", data_cm.loc["casual","len"])
ax.barh(" ", data_cm.loc["member","len"], left = data_cm.loc["casual","len"])
ax.legend(data_cm.index, bbox_to_anchor=(1.2, 1.02) )
ax.bar_label(ax.containers[0], label_type = 'center', color = 'w', fmt='%.f')
ax.bar_label(ax.containers[1], label_type = 'center', color = '0', fmt='%.f')

plt.axis('off')
plt.show()
```

5.3 Length in bargraph

In [ ]:

```python
fig, ax = plt.subplots(figsize = (8,0.5))
ax.barh(" ", data_cm.loc["casual","sum"])
ax.barh(" ", data_cm.loc["member","sum"], left = data_cm.loc["casual","sum"])
ax.legend(data_cm.index, bbox_to_anchor=(1.2, 1.02) )
ax.bar_label(ax.containers[0], label_type = 'center', color = 'w',fmt='%.f Mins'
ax.bar_label(ax.containers[1], label_type = 'center', color = '0', fmt='%.f Mins

plt.axis('off')
plt.show()
```

5.4 Average duration in rides in bargraph

In [ ]:

```
fig, ax = plt.subplots(figsize = (8,0.5))
ax.barh(" ", data_cm.loc["casual","mean"])
ax.barh(" ", data_cm.loc["member","mean"], left = data_cm.loc["casual","mean"])
ax.legend(data_cm.index, bbox_to_anchor=(1.2, 1.02) )
ax.bar_label(ax.containers[0], label_type = 'center', color = 'w',fmt='%.f Mins'
ax.bar_label(ax.containers[1], label_type = 'center', color = '0', fmt='%.f Mins

plt.axis('off')
plt.show()
```

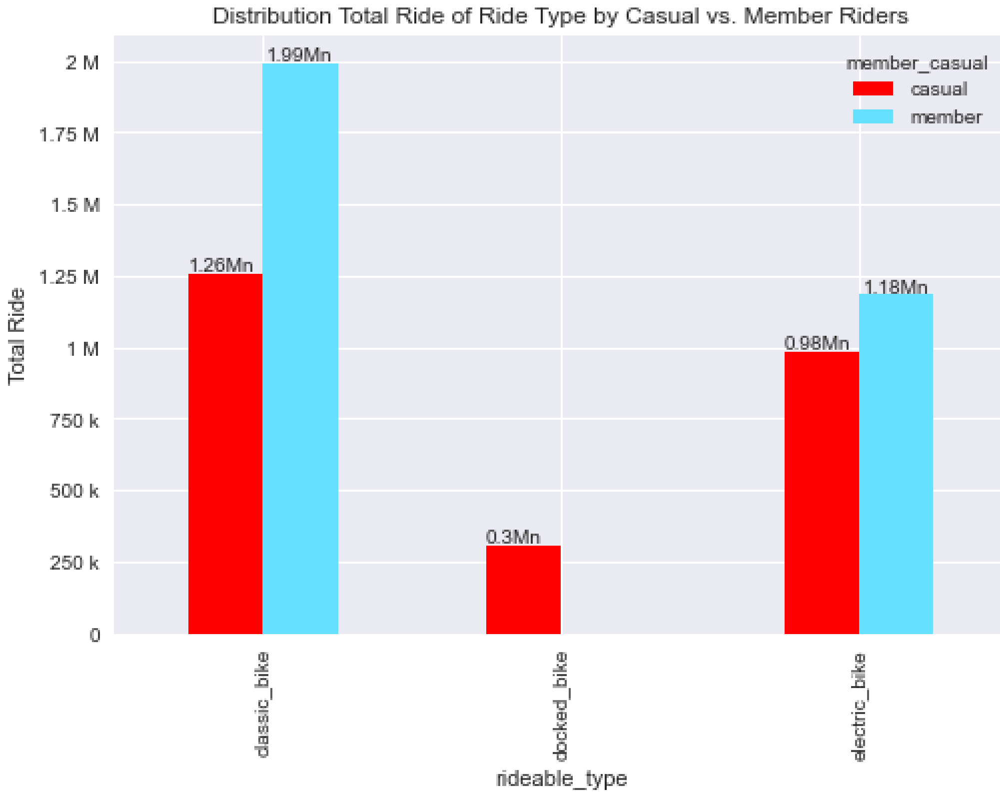
In [ ]:

```
#set style and defaul color
plt.style.use('seaborn')
mpl.rcParams['axes.prop_cycle'] = mpl.cycler(color=["#ff0000", "#66e0ff"])
```

In [ ]:

```
data_ra = data_cleaning_v2.pivot_table(index = 'rideable_type', values = 'ride_l

ax = data_ra["len"].plot.bar()

for i, number in enumerate(data_ra['len']['casual']):
    plt.text(x=i-0.25, y= number + 10000, s=str(round((number/1000000),2)) + "Mn
for i, number in enumerate(data_ra['len']['member']):
    plt.text(x=i+.01, y= number + 10000, s=str(round((number/1000000),2)) + "Mn"

ax.yaxis.set_major_formatter(ticker.EngFormatter())
ax.set(ylabel = "Total Ride", title = "Distribution Total Ride of Ride Type by C

plt.show()
```

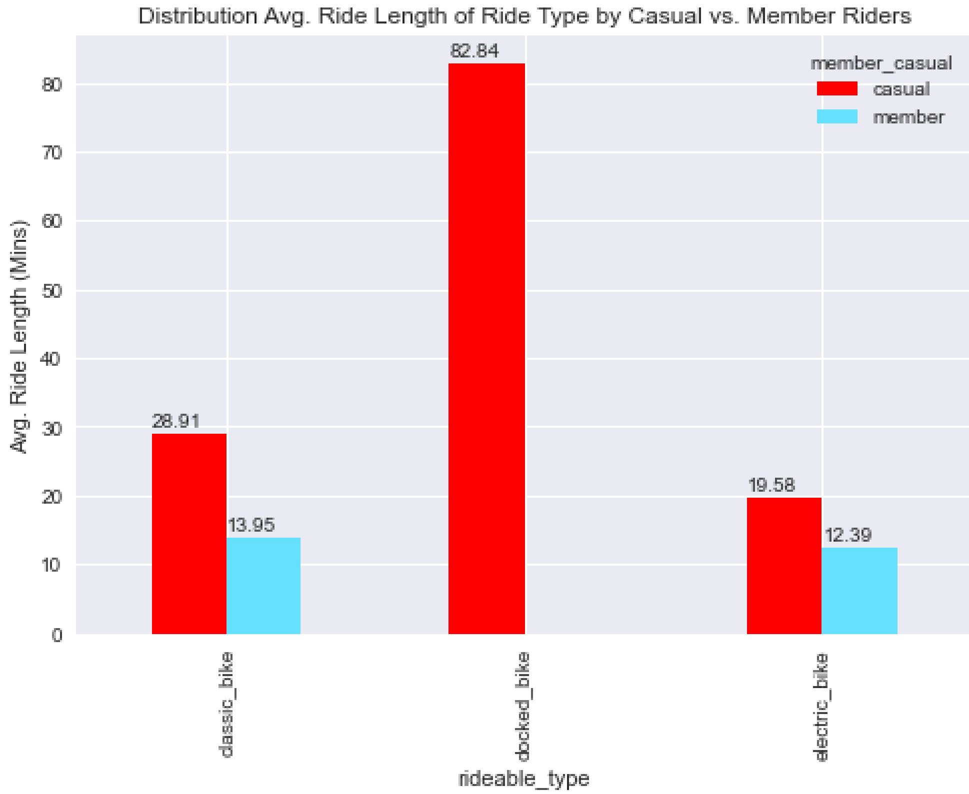
```
posx and posy should be finite values
posx and posy should be finite values
```



In [ ]:

```
ax = data_ra["mean"].plot.bar()

for i, number in enumerate(data_ra['mean']['casual']):
    plt.text(x=i-0.25, y= number + 1, s=round(number,2))
for i, number in enumerate(data_ra['mean']['member']):
    plt.text(x=i, y= number + 1, s=round(number,2))

ax.yaxis.set_major_formatter(ticker.EngFormatter())
ax.set(ylabel = "Avg. Ride Length (Mins)", title = "Distribution Avg. Ride Lengt

plt.show()
```

```
posx and posy should be finite values
posx and posy should be finite values
```



6. Time

In [ ]:

```
month_bar = data_cleaning_v2.groupby(["member_casual","month"])["ride_length"].c
```

In [ ]:

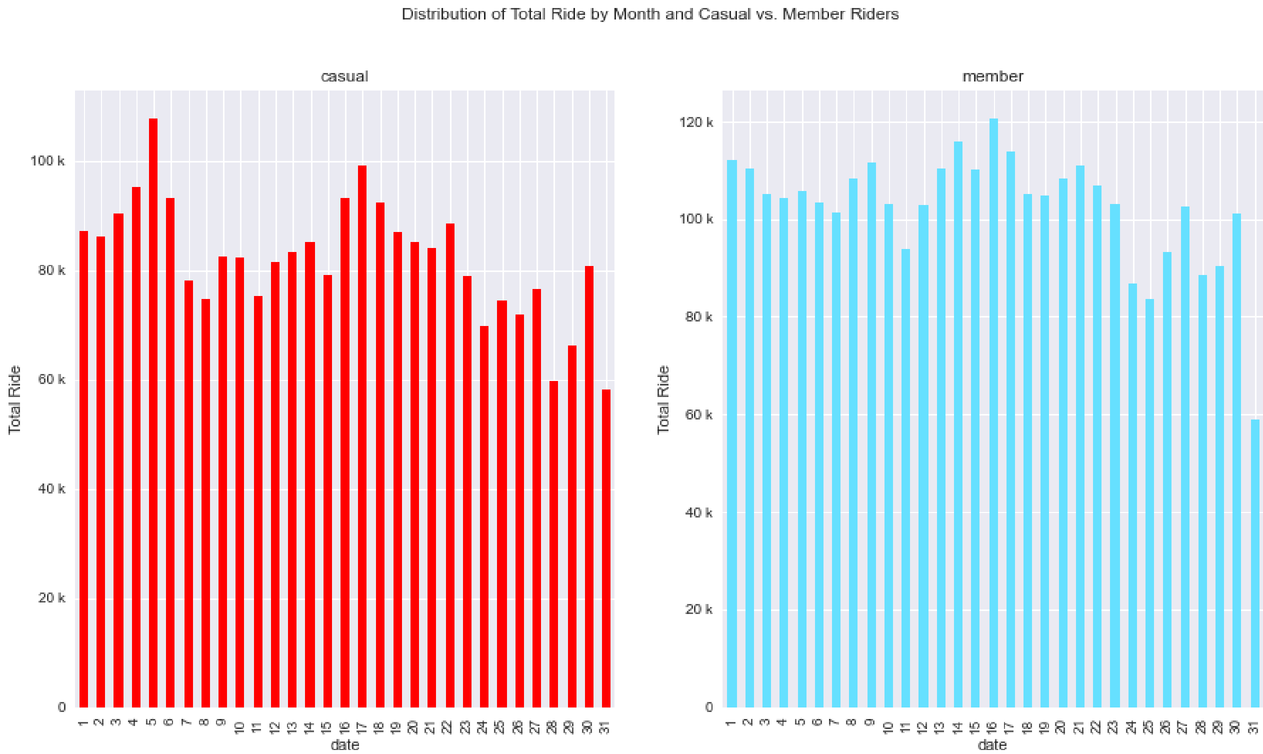
```
fig,ax = plt.subplots(1,2, figsize = (15,8))

ax1 = month_bar["casual"].plot.bar(ax =ax[0], title = 'casual', ylabel = 'Total
ax1.yaxis.set_major_formatter(ticker.EngFormatter())


ax2 = month_bar["member"].plot.bar(ax =ax[1], title = 'member', color= '#66e0ff'
ax2.yaxis.set_major_formatter(ticker.EngFormatter())

fig.suptitle('Distribution of Total Ride by Month and Casual vs. Member Riders')


plt.show()
```

In [ ]:

```
date_bar = data_cleaning_v2.groupby(["member_casual","date"])["ride_length"].cou
```

In [ ]:

```
fig,ax = plt.subplots(1,2, figsize = (15,8))

ax1 = date_bar["casual"].plot.bar(ax =ax[0], title = 'casual', ylabel = 'Total R
ax1.yaxis.set_major_formatter(ticker.EngFormatter())


ax2 = date_bar["member"].plot.bar(ax =ax[1], title = 'member' , color= '#66e0ff'
ax2.yaxis.set_major_formatter(ticker.EngFormatter())

fig.suptitle('Distribution of Total Ride by Month and Casual vs. Member Riders')


plt.show()
```



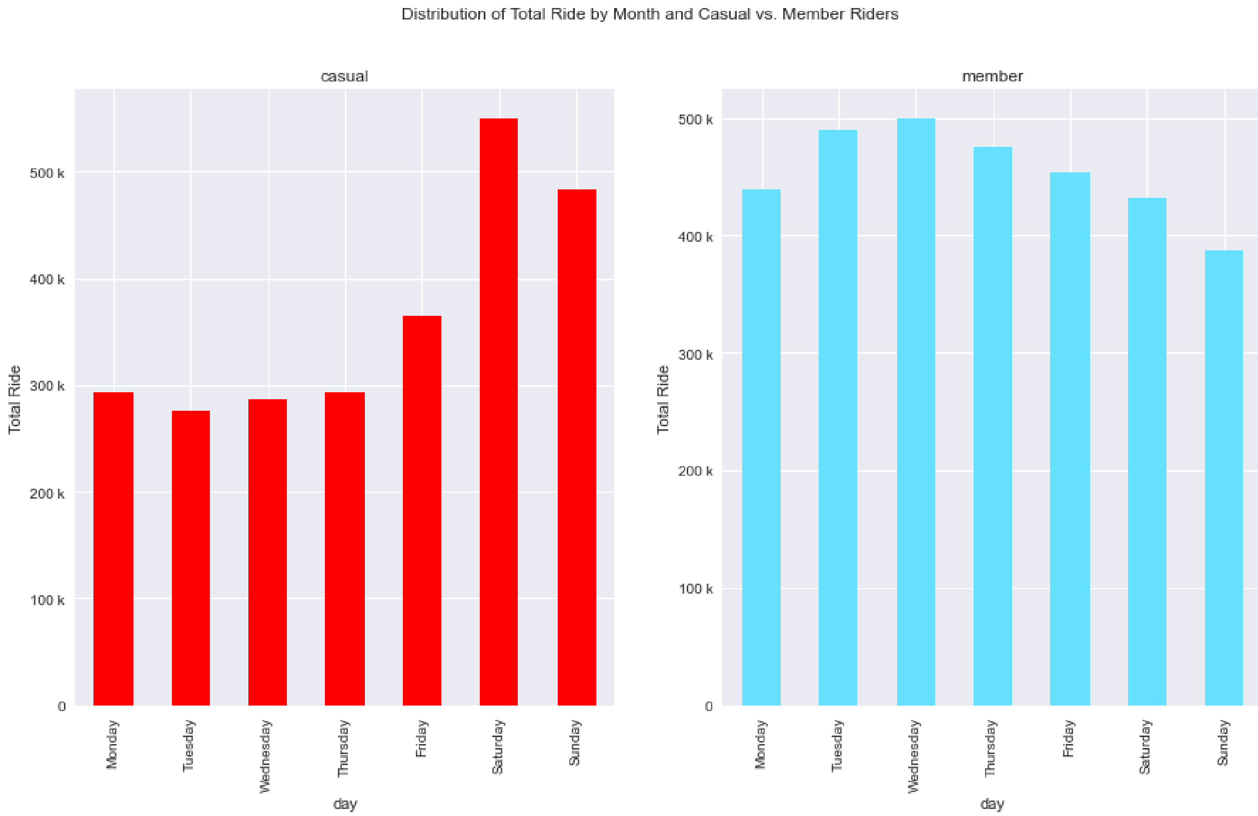
6.1 Day

In [ ]:

```
day_bar = data_cleaning_v2.groupby(["member_casual","day"])["ride_length"].agg([
```

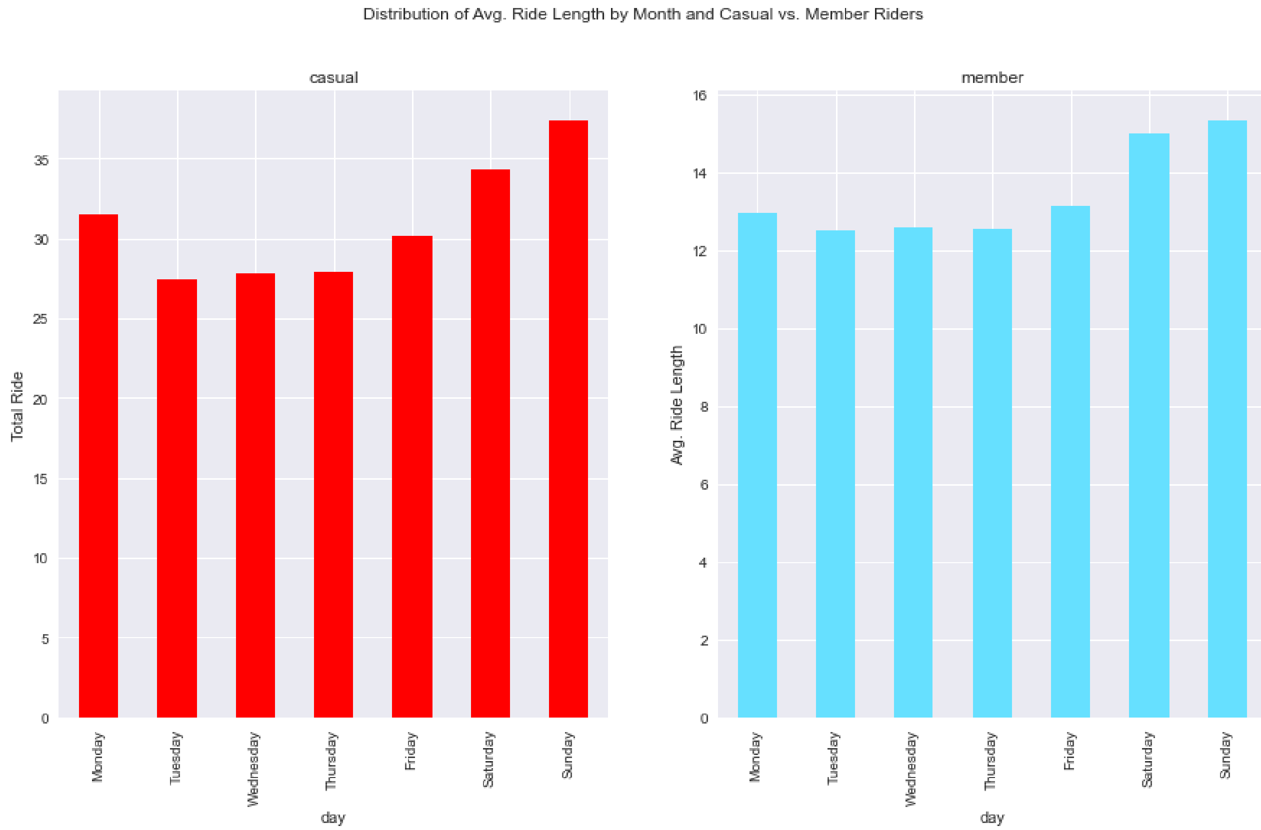
In [ ]:

```
fig,ax = plt.subplots(1,2, figsize = (15,8))

ax1 = day_bar.loc["casual", "len"].plot.bar(ax =ax[0], title = 'casual', ylabel
ax1.yaxis.set_major_formatter(ticker.EngFormatter())

ax2 = day_bar.loc["member","len"].plot.bar(ax =ax[1], title = 'member' , color=
ax2.yaxis.set_major_formatter(ticker.EngFormatter())

fig.suptitle('Distribution of Total Ride by Month and Casual vs. Member Riders')

plt.show()
```



In [ ]:

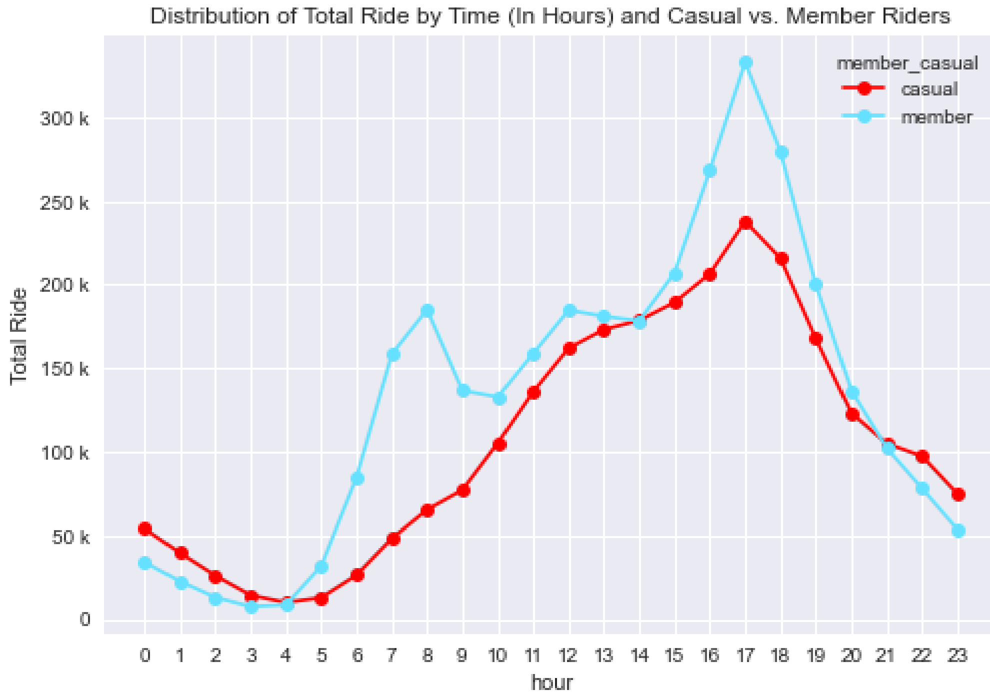
```
fig,ax = plt.subplots(1,2, figsize = (15,8))

ax1 = day_bar.loc["casual", "mean"].plot.bar(ax =ax[0], title = 'casual', ylabel
ax1.yaxis.set_major_formatter(ticker.EngFormatter())

ax2 = day_bar.loc["member","mean"].plot.bar(ax =ax[1], title = 'member' , color=
ax2.yaxis.set_major_formatter(ticker.EngFormatter())

fig.suptitle('Distribution of Avg. Ride Length by Month and Casual vs. Member Ri

plt.show()
```



6.2 Hour

In [ ]:
```
hour_line = data_cleaning_v2.pivot_table(index = "hour", columns = "member_casua
```

In [ ]:
```
ax = hour_line.plot(marker = 'o', ylabel = "Total Ride", title = "Distribution o
ax.yaxis.set_major_formatter(ticker.EngFormatter())
ax.set_xticks(hour_line.index)

plt.show()
```



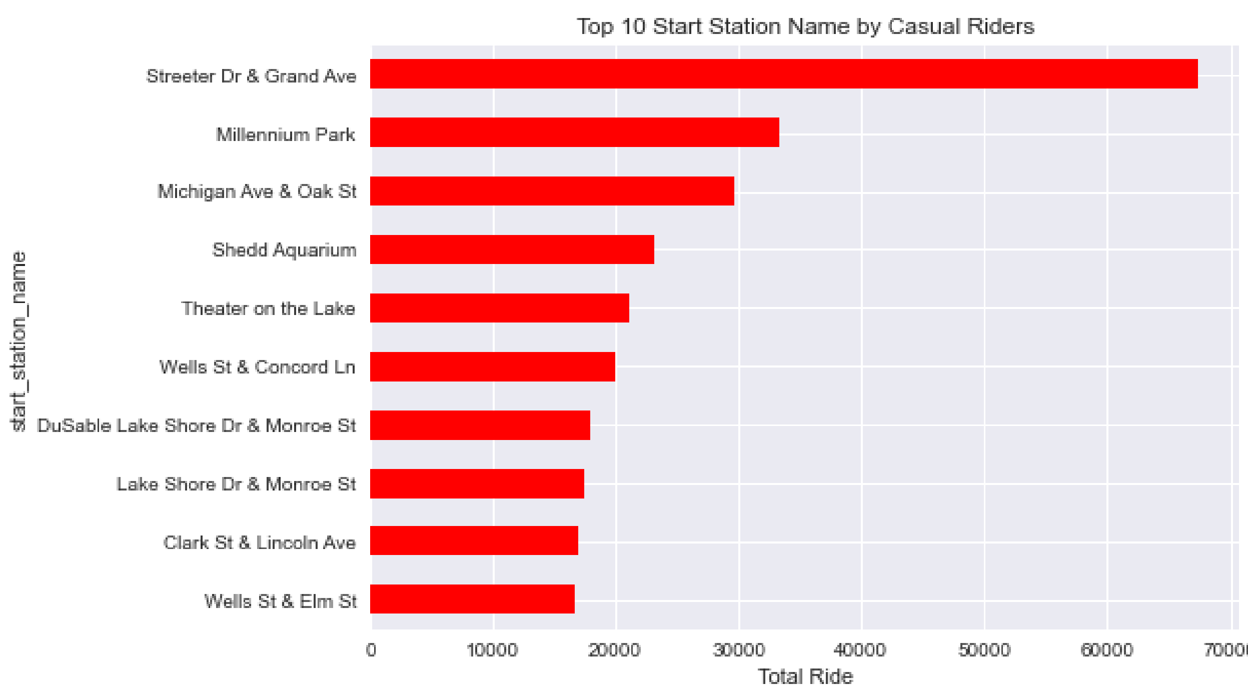
6.3a Casual Riders by Starting Station

In [ ]:

```
#Will not automatically enter null data
ss_bar = data_cleaning_v2.groupby(["member_casual","start_station_name"])["ride_
```

In [ ]:

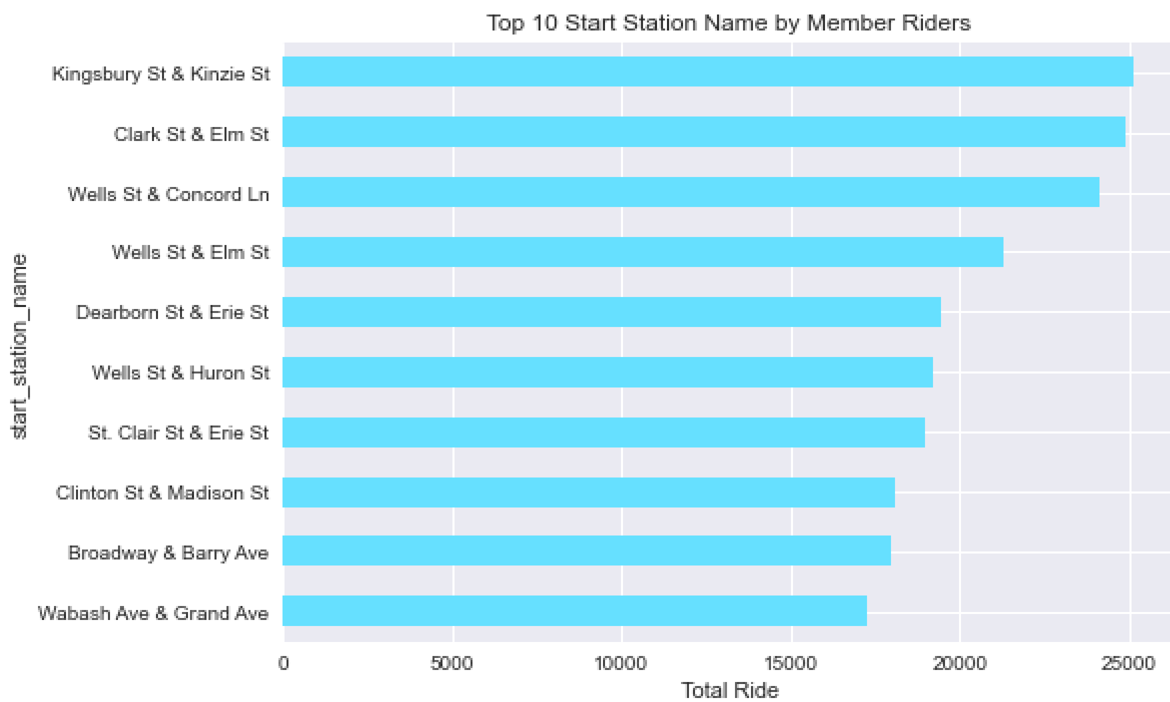
```
ax = ss_bar.loc["casual"][:10].plot.barh()
ax.invert_yaxis()
ax.set(xlabel = 'Total Ride', title = "Top 10 Start Station Name by Casual Rider


plt.show()
```



6.3b Member Riders By Starting Station

In [ ]:

```
ax = ss_bar.loc["member"][:10].plot.barh(color= '#66e0ff')
ax.invert_yaxis()
ax.set(xlabel = 'Total Ride', title = "Top 10 Start Station Name by Member Rider


plt.show()
```



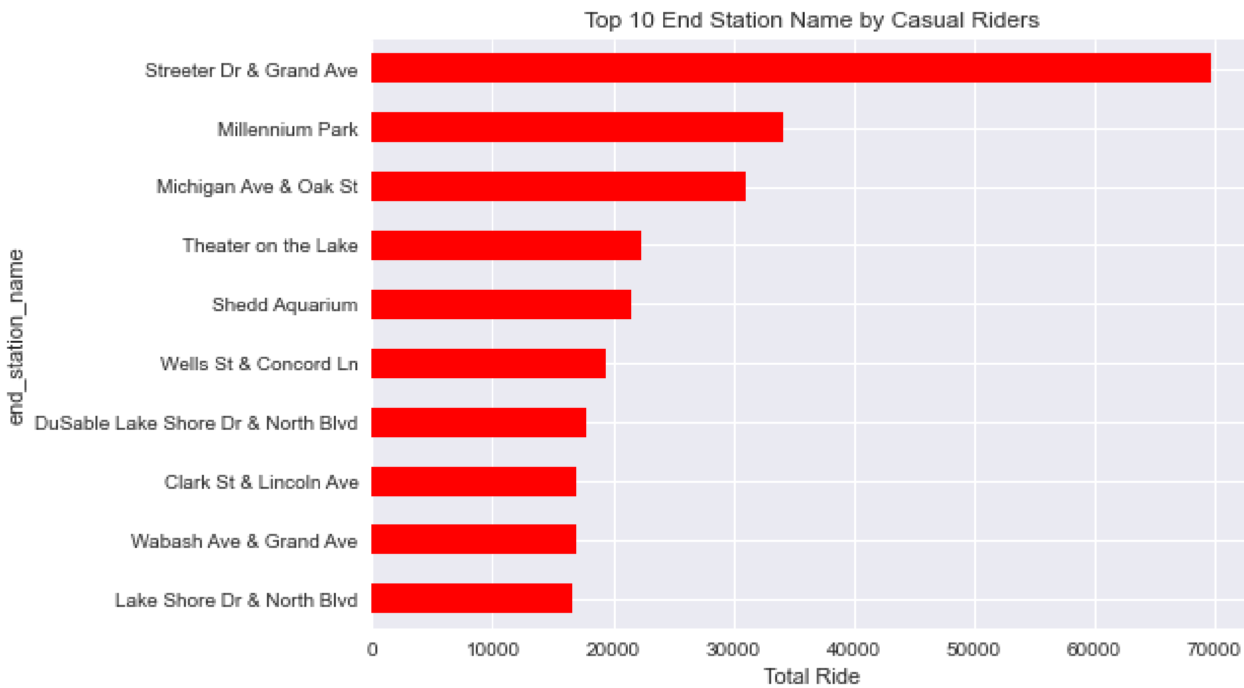
In [ ]:

```
#will not automatically enter null data
es_bar = data_cleaning_v2.groupby(["member_casual","end_station_name"])["ride_le
```

6.3c Casual Riders by Ending Station

In [ ]:

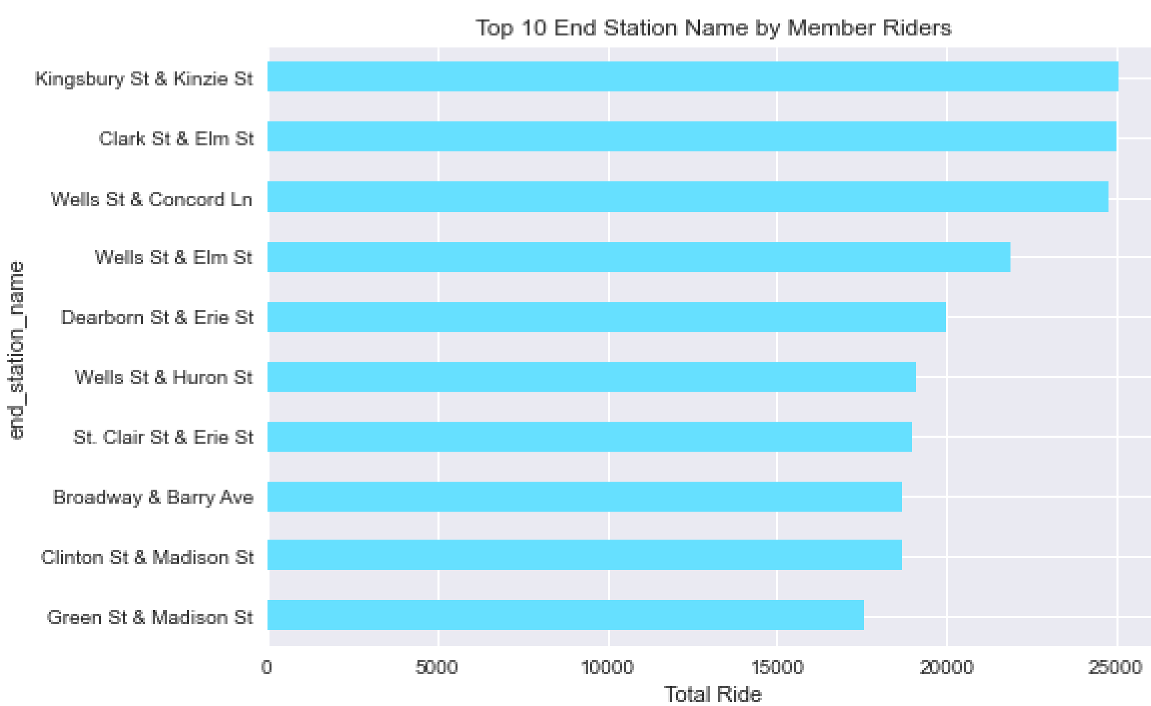
```
ax = es_bar.loc["casual"][:10].plot.barh()
ax.invert_yaxis()
ax.set(xlabel = 'Total Ride', title = "Top 10 End Station Name by Casual Riders"


plt.show()
```



6.3d Member Riders By Ending Station

In [ ]:

```
ax = es_bar.loc["member"][:10].plot.barh(color= '#66e0ff')
ax.invert_yaxis()
ax.set(xlabel = 'Total Ride', title = "Top 10 End Station Name by Member Riders"


plt.show()
```



7. Conclusion

Conclusion Based on the main purpose which is "The team will design a new marketing strategy to convert casual riders into annual members. Cyclistic executives must approve recommendations."

They must be backed up with compelling data insights and professional data visualizations. And one of it has to help to answer this question : How do annual members and casual riders use Cyclistic bikes differently?

Based on our analysis above, I may summarize by a first few past process on dataset :

I am not remove null data because some important columns have no null except for point 4 below Exclude the data which has smaller ride length or the same as zero Add month, date, day, time and ride_length columns and remove the columns that relate to latitude and longitude Analysis in the station which automatically filters out null data with groupby

From the process, I can conclude the analysis as detailed below:

General Analysis. By the number of rides, member is higher than casual, but in terms of numbers by ride length, the casual is higher as seen in the total and average ride length of the casual is two times higher than the member. Riders also prefer to use classic bikes, then electric bikes, and docked bikes.

Analysis by time, based on months, both have increments in summer. The casual members reach the peak in July and August and decrease in Winter, which is in February. In terms of the actual date, in general, there is no pattern, it's just that in casual it looks like at the end of the month there is a slight decrease. Days of week, both the number of rides and average ride length for the casual, which is there is increation in weekends or on Saturdays and Sundays. Inversely proportional with the members, which is generally the same but there will be decrease in weekends. Based on viewed by time in hours, the numbers of ride keep increasing starts from 5.00 AM until on its peak at 5.00 PM and after that get decreased. But for members it isn't as smooth as casual. Increment happens, and then decreament also happens in certain hours. The increment happens because around 5.00 PM is the busy hours. Some people are coming back from work, and some are starting their activities.

Station name, the next is viewed by the area mapping, both member and casual has the different station with the highest number. For casual, it is the highest start ride,and the end station is on Streeter Dr & Grand Ave. For the member, the highest start ride and the end station is in Kingsbury St & Kenzie St.

By those three analysis, there is possibility the casual riders were riding for holiday, so it mostly chosen by tourists, customer who takes time to sport in every weekend, or in certain times for the certain destination, which is not daily visit like the members, whose are possibly the customers like daily workers segmentation or people who have destination to certain area every day.

Advice By those analysis conclusions, below is the advice that can be put in action :

The next step is to do analysis by customer to get insight, such as :

To determine customers who have routine patterns, like weekly routines in using bikes and offer them to join as members To see customer statistics to take a view of the total ride, and total time both monthly and yearly and compare the benefit if they join as members to be next offered to join as member Promotion and program offer period, to decide the best promotion period, like after winter or in weekends and make programs for certain times like :

Make a summer promotion; lower price only available in summer. Make a prepay program for users to have an alotted amount of trips per whatever scale is needed. Area and program. Mapping an area like neighbourhood of start and end location as the promotion facility and make a collaboration program like :

Do a lot of promotion in that area like in airports, train stations, schools, tourist attractions, etc. Make a collaboration program for tourists in vacation and attraction spots, schools, etc.