“Cyclistic” Case Study – Google Data Analytics Capstone

This is the report that deals with the findings on the data analytics done on the “Cyclistic” data set as part of Google Data Analytics course Capstone. The report is broken down into 6 parts which discusses different phases of the data analytics involved in coming up with the findings of the case study.

The data used was the cyclistic data case study problem statement. I used data from Nov, 2020 to Oct, 2021 (12 months) to so my data analysis.

Ask

We would want to answer these three questions through our data analysis

- How do annual members and casual riders use Cyclistic bikes differently?
- Why would casual riders buy Cyclistic annual memberships?
- How can Cyclistic use digital media to influence casual riders to become members?

Prepare

As previously mentioned, data from Nov, 2020 to Oct, 2021 we taken and data analysis was performed on that. A large data frame is created with all the data of 12 months.

```
# Read last 12 months CSV into dataframe and merge into new data frame
df1 = pd.read_csv("202011-divvy-tripdata.csv")
df2 = pd.read_csv("202012-divvy-tripdata.csv")
df3 = pd.read_csv("202101-divvy-tripdata.csv")
df4 = pd.read_csv("202102-divvy-tripdata.csv")
df5 = pd.read_csv("202103-divvy-tripdata.csv")
df6 = pd.read_csv("202104-divvy-tripdata.csv")
df7 = pd.read_csv("202105-divvy-tripdata.csv")
df8 = pd.read_csv("202106-divvy-tripdata.csv")
df9 = pd.read_csv("202107-divvy-tripdata.csv")
df10 = pd.read_csv("202108-divvy-tripdata.csv")
df11 = pd.read_csv("202109-divvy-tripdata.csv")
df12 = pd.read_csv("202110-divvy-tripdata.csv")
df  = pd.concat([df1,df2,df3,df4,df5,df6,df7,df8,df9,df10,df11,df12])
```

Process

In this stage we try to understand the data and clean/remove erroneous and missing data.

We first check the size of the data

```
# Get the length and width of the data frame
df.shape
```

```
(5378834, 13)
```

Now, we see first few rows of the data

```
#View the first 5 rows of the dataframe
df.head(5)
```

ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	start_lat	start_lng	end_lat	end_lng
F9B921	electric_bike	2020-11-01 13:36:00	2020-11-01 13:45:40	Dearborn St & Erie St	110	St. Clair St & Erie St	211	41.894177	-87.629127	41.894434	-87.623379
E4F82D	electric_bike	2020-11-01 10:03:26	2020-11-01 10:14:45	Franklin St & Illinois St	672	Noble St & Milwaukee Ave	29	41.890959	-87.635343	41.900675	-87.662480
82BDC5	electric_bike	2020-11-01 00:34:05	2020-11-01 01:03:06	Lake Shore Dr & Monroe St	76	Federal St & Polk St	41	41.880983	-87.616754	41.872054	-87.629550
080B9E	electric_bike	2020-11-01 00:45:16	2020-11-01 00:54:31	Leavitt St & Chicago Ave	659	Stave St & Armitage Ave	185	41.895499	-87.682013	41.917744	-87.691392
168C60	electric_bike	2020-11-01 15:43:25	2020-11-01 16:16:52	Buckingham Fountain	2	Buckingham Fountain	2	41.876497	-87.620358	41.876448	-87.620338

Now we try to remove duplicates, but we see the numbers of rows are still same, which means that data has no duplicates.

```
#Clean data by removing duplicates
df.drop_duplicates()
df.shape
```

```
(5378834, 13)
```

Next, we add two new columns ride length and day of the week, which shows the length of the ride for each ride and the day of the week on which that ride was taken.

```
# Find the ride length of ride and the start day of the week
df['ride_length'] =  pd.to_datetime(df['ended_at'],format="%Y-%m-%d %H:%M:%S") - pd.to_datetime(df['started_at'],format="%Y-
df['day_of_week'] = pd.to_datetime(df['started_at'],format="%Y-%m-%d %H:%M:%S").dt.day_name()
df.head(5)
```

start_station_name	start_station_id	end_station_name	end_station_id	start_lat	start_lng	end_lat	end_lng	member_casual	ride_length	day_of_week
Dearborn St & Erie St	110	St. Clair St & Erie St	211	41.894177	-87.629127	41.894434	-87.623379	casual	00:09:40	Sunday
Franklin St & Illinois St	672	Noble St & Milwaukee Ave	29	41.890959	-87.635343	41.900675	-87.662480	casual	00:11:19	Sunday
Lake Shore Dr & Monroe St	76	Federal St & Polk St	41	41.880983	-87.616754	41.872054	-87.629550	casual	00:29:01	Sunday
Leavitt St & Chicago Ave	659	Stave St & Armitage Ave	185	41.895499	-87.682013	41.917744	-87.691392	casual	00:09:15	Sunday
Buckingham Fountain	2	Buckingham Fountain	2	41.876497	-87.620358	41.876448	-87.620338	casual	00:33:27	Sunday

Now when we find the summary, we see that ride lengths are negative.

```
#Check the summary of the dataframe
df.describe()
```

	start_lat	start_lng	end_lat	end_lng	ride_length
count	5.378834e+06	5.378834e+06	5.374003e+06	5.374003e+06	5378834
mean	4.190179e+01	-8.764576e+01	4.190207e+01	-8.764598e+01	0 days 00:20:29.374965
std	4.548065e-02	2.801962e-02	4.558547e-02	2.819680e-02	0 days 04:59:59.531975
min	4.164000e+01	-8.784000e+01	4.151000e+01	-8.807000e+01	-21 days +19:50:02
25%	4.188189e+01	-8.766000e+01	4.188209e+01	-8.766000e+01	0 days 00:06:58
50%	4.189964e+01	-8.764178e+01	4.190000e+01	-8.764253e+01	0 days 00:12:23
75%	4.192914e+01	-8.762772e+01	4.192955e+01	-8.762775e+01	0 days 00:22:26
max	4.208000e+01	-8.752000e+01	4.216812e+01	-8.744000e+01	38 days 20:24:09

So, we remove, those records,

```
#Remove rides with negative or zero length
df_pos_ride = df[df['ride_length'].astype('timedelta64[m]') > 0]
df_pos_ride.shape
```

```
(5296787, 15)
```

```
#Summary of new dataframe
df_pos_ride.describe()
```

	start_lat	start_lng	end_lat	end_lng	ride_length
count	5.296787e+06	5.296787e+06	5.291998e+06	5.291998e+06	5296787
mean	4.190183e+01	-8.764574e+01	4.190211e+01	-8.764596e+01	0 days 00:22:50.475192
std	4.541528e-02	2.795828e-02	4.552143e-02	2.813829e-02	0 days 03:01:35.172420
min	4.164000e+01	-8.784000e+01	4.151000e+01	-8.807000e+01	0 days 00:01:00
25%	4.188189e+01	-8.766000e+01	4.188209e+01	-8.766000e+01	0 days 00:07:11
50%	4.189964e+01	-8.764178e+01	4.190000e+01	-8.764259e+01	0 days 00:12:35
75%	4.192914e+01	-8.762772e+01	4.192955e+01	-8.762775e+01	0 days 00:22:40
max	4.208000e+01	-8.752000e+01	4.216812e+01	-8.744000e+01	38 days 20:24:09

Analyse

In the analysis stage we try to summarise different aspect of data to make observations.

We find our dataset contribution, we see that our data has more data for member than for casual rider, so it might be good idea to get more casual ride data.

```
#Find the count of each type of ride
df_pos_ride.groupby(['member_casual']).size()
```

```
member_casual
casual     2437162
member     2859625
dtype: int64
```

Now we check ride count for each day and see that “Saturday” is the busiest day.

```
#Find the count of ride for each day of week
df_pos_ride.groupby(['day_of_week']).size()
```

```
day_of_week
Friday       768420
Monday       659515
Saturday     958009
Sunday       831404
Thursday     692664
Tuesday      685567
Wednesday    701208
dtype: int64
```

We also check the count to find the type of bike favourite among the riders. It is not clear that classic bike is the most favourite bike or if the count is more since classic bike has more availability.

```
#Find the count of ride for each type of bike
df_pos_ride.groupby(['rideable_type']).size()
```

```
rideable_type
classic_bike     3027336
docked_bike       459314
electric_bike    1810137
dtype: int64
```

Next, we find the top 10 stations from which riders start and end the trip.

```
#Find the count of ride for top 10 start station
df_pos_ride.groupby(['start_station_name']).size().nlargest(10)
```

```
start_station_name
Streeter Dr & Grand Ave     79922
Michigan Ave & Oak St       43943
Wells St & Concord Ln       41993
Millennium Park             40972
Clark St & Elm St           40194
Theater on the Lake         37491
Wells St & Elm St           36299
Clark St & Lincoln Ave      32757
Clark St & Armitage Ave     32414
Wabash Ave & Grand Ave      31735
dtype: int64
```

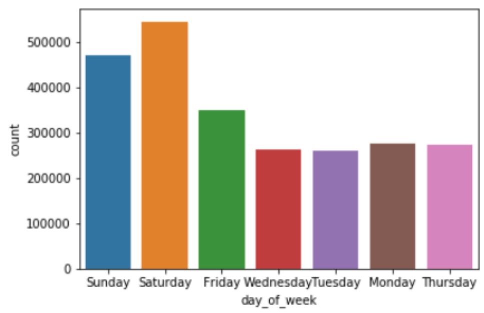
```
#Find the count of ride for 10 end station
df_pos_ride.groupby(['end_station_name']).size().nlargest(10)
```

```
end_station_name
Streeter Dr & Grand Ave     80327
Michigan Ave & Oak St       44431
Wells St & Concord Ln       42357
Millennium Park             41687
Clark St & Elm St           39688
Theater on the Lake         37773
Wells St & Elm St           36110
Clark St & Lincoln Ave      32720
Wabash Ave & Grand Ave      32293
Clark St & Armitage Ave     31689
dtype: int64
```

Next, we summarise data based on rider type. We plot two graphs which show the ride count for each type of rider for each day of the week.

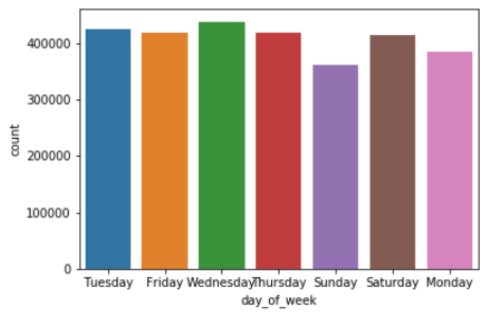
```
#Casual Rider
sns.countplot(x='day_of_week', data=df_casual)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24e112b4448>
```




```
#Member Rider
sns.countplot(x='day_of_week', data=df_member)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24e112f9c08>
```



Now, we summarise ride length data for each type of rider.

```python
df_casual.describe()
```

	start_lat	start_lng	end_lat	end_lng	ride_length
count	2.437162e+06	2.437162e+06	2.433832e+06	2.433832e+06	2437162
mean	4.190159e+01	-8.764466e+01	4.190195e+01	-8.764492e+01	0 days 00:32:59.593203
std	4.552359e-02	2.898526e-02	4.561395e-02	2.930510e-02	0 days 04:25:38.372234
min	4.164000e+01	-8.784000e+01	4.151000e+01	-8.807000e+01	0 days 00:01:00
25%	4.188196e+01	-8.765897e+01	4.188224e+01	-8.765917e+01	0 days 00:09:32
50%	4.190000e+01	-8.763917e+01	4.190000e+01	-8.763919e+01	0 days 00:16:37
75%	4.192877e+01	-8.762581e+01	4.192889e+01	-8.762591e+01	0 days 00:30:16
max	4.208000e+01	-8.752000e+01	4.216812e+01	-8.744000e+01	38 days 20:24:09

```python
df_member.describe()
```

	start_lat	start_lng	end_lat	end_lng	ride_length
count	2.859625e+06	2.859625e+06	2.858166e+06	2.858166e+06	2859625
mean	4.190203e+01	-8.764667e+01	4.190225e+01	-8.764685e+01	0 days 00:14:11.344443
std	4.532181e-02	2.701797e-02	4.544205e-02	2.707370e-02	0 days 00:27:48.694959
min	4.164850e+01	-8.784000e+01	4.160000e+01	-8.796000e+01	0 days 00:01:00
25%	4.188189e+01	-8.766027e+01	4.188189e+01	-8.766029e+01	0 days 00:05:56
50%	4.189918e+01	-8.764407e+01	4.189993e+01	-8.764410e+01	0 days 00:10:05
75%	4.192955e+01	-8.763000e+01	4.192974e+01	-8.763000e+01	0 days 00:17:16
max	4.207000e+01	-8.752000e+01	4.215000e+01	-8.751000e+01	1 days 01:59:56

Now we find the top stations where the riders start and end trip.

```python
df_member.groupby(['start_station_name']).size().nlargest(10)
```

```
start_station_name
Clark St & Elm St            23934
Wells St & Concord Ln        22556
Kingsbury St & Kinzie St     21614
Wells St & Elm St            19977
Dearborn St & Erie St        18814
Wells St & Huron St          18371
St. Clair St & Erie St       18029
Broadway & Barry Ave         17612
Clark St & Armitage Ave      16395
Desplaines St & Kinzie St    16059
dtype: int64
```

```python
df_casual.groupby(['start_station_name']).size().nlargest(10)
```

```
start_station_name
Streeter Dr & Grand Ave       64233
Millennium Park               32937
Michigan Ave & Oak St         29637
Shedd Aquarium                22242
Theater on the Lake           21584
Lake Shore Dr & Monroe St     21167
Wells St & Concord Ln         19437
Clark St & Lincoln Ave        16821
Indiana Ave & Roosevelt Rd    16653
Wells St & Elm St             16322
dtype: int64
```

```
1 df_member.groupby(['end_station_name']).size().nlargest(10)
```

```
end_station_name
Clark St & Elm St           24246
Wells St & Concord Ln       23277
Kingsbury St & Kinzie St    21884
Wells St & Elm St           20677
Dearborn St & Erie St       19481
Broadway & Barry Ave        18235
St. Clair St & Erie St      18114
Wells St & Huron St         17842
Green St & Madison St       16002
Clark St & Armitage Ave     15932
dtype: int64
```

```
1 df_casual.groupby(['end_station_name']).size().nlargest(10)
```

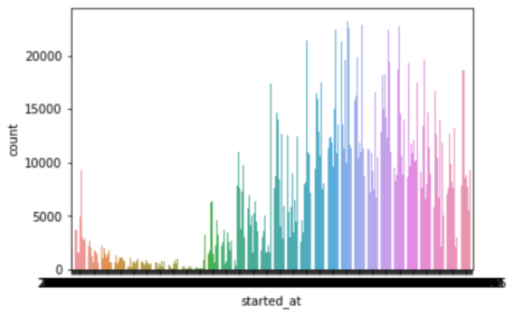
```
end_station_name
Streeter Dr & Grand Ave       66244
Millennium Park               33846
Michigan Ave & Oak St         30990
Theater on the Lake           23113
Shedd Aquarium                20589
Lake Shore Dr & Monroe St     19745
Wells St & Concord Ln         19080
Lake Shore Dr & North Blvd    18124
Clark St & Lincoln Ave        17077
Wabash Ave & Grand Ave        16783
dtype: int64
```

Next when we check the usage of bike across the year for each rider type and see that the usage for casual riders is more seasonal.

```
1 sns.countplot(pd.to_datetime(df_casual['started_at'],format="%Y-%m-%d %H:%M:%S").dt.date)
```

```
C:\Users\Aryan\Anaconda3\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  FutureWarning
```

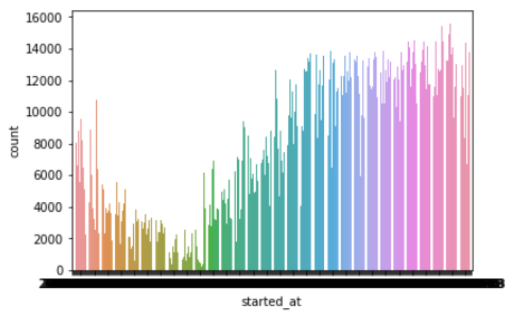
```
<matplotlib.axes._subplots.AxesSubplot at 0x24e695ec248>
```




```
1 sns.countplot(pd.to_datetime(df_member['started_at'],format="%Y-%m-%d %H:%M:%S").dt.date)
```

```
C:\Users\Aryan\Anaconda3\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24e0ec12548>
```



Share

From the above analysis though we can get initial indication towards the question asked above. But to come up with concrete recommendation, more data about the rider would be required.

Following are the observations based on above analysis on understanding how casual riders are different from member riders:

- Member riders ride bike usage is almost same through out the week except on Sunday, while the bike usage for casual rider the usage is more on weekends.
- Casual bikers bike usage is most on Saturday.
- The average duration of ride file casual rider is ~32 minutes and for member rider is ~14 minutes.
- We can see a seasonal usage of bike. This is more evident in case of casual riders than member rides, who avoid riding bike only when the weather is very cold.

Act

Below are my recommendations on how to influence casual riders to become members?

- Since the casual rider's bike usage is significantly higher on weekends, introduce a weekend membership plan. Later show that annual membership would be cheaper than taking weekend membership.
- We know the top stations from which a casual biker starts or ends a trip. We can put advertisement of annual membership on those location.
- We can give a discount on annual membership during the summer months, so that we have higher probability to getting more riders signed into the membership as it's the peak season.
- Keep the daily ride fare low for short trips and increase it significantly once the it crosses 30 minutes mark to make the rider buy week membership and slowly entice them to buy annual membership.
- Since most of the member riders seem to be office goers as observed from the data, we can target the casual riders who go to office by other medium, if such data about causal rider is available.