Visualizing FordGoBike Data(2017)

by Anubhav Nehru

Preliminary Wrangling

Bay Wheels (previously known as Ford GoBike) is a regional public bike sharing system in the San Francisco Bay Area, California. Bay Wheels is the first regional and large-scale bicycle sharing system deployed in California and on the West Coast of the United States with nearly 500,000 rides since the launch in 2017 and had about 10,000 annual subscribers as of January 2018. The dataset used for this exploratory analysis consists of monthly individual trip data from June 2017 to December 2018 in CSV format covering the greater San Francisco Bay area. I was unable find gender columns , age column and some few more columns in the lyft website , so I have refered to the data that is also available here(https://drive.google.com/file/d/1FhzMjCIpv-kgZqauYcwTSPEkTMUH7jjm/view?usp=sharing (https://drive.google.com/file/d/1FhzMjCIpv-kgZqauYcwTSPEkTMUH7jjm/view?usp=sharing)).

In [1]:

```
# import all packages and set plots to be embedded inline
import os
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

os.chdir("C:/Users/anubh/OneDrive/Documents/Project/Udacity/Untitled Folder")
```

> Load in your dataset and describe its properties through the questions below. Try and motivate your exploration goals through this section.

In [2]:

```
df = pd.read_csv("data.csv")
```

```
C:\Users\anubh\Anaconda3\lib\site-packages\IPython\core\interactiveshell.p
y:3020: DtypeWarning: Columns (3) have mixed types. Specify dtype option o
n import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:

```
df.drop(columns = ["Unnamed: 0" ,"Unnamed: 0.1"] , inplace = True)
```

In [4]:

```
df.shape
```

Out[4]:

```
(1538086, 16)
```

In [5]:

```
df.head()
```

Out[5]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	end_sta
0	1035	No	598	114.0	37.764478	
1	1673	No	943	324.0	37.788300	
2	3498	No	18587	15.0	37.795392	
3	3129	No	18558	15.0	37.795392	
4	1839	Yes	885	297.0	37.322980	

In [6]:

```
df.sample(10)
```

Out[6]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	e
570450	712	NaN	1215	160.0	37.805318	
1532135	85	No	361	179.0	37.816073	
224758	1617	No	678	42.0	37.778650	
1528208	3724	Yes	460	157.0	37.846784	
1028654	2629	No	661	66.0	37.778742	
347300	1476	NaN	585	24.0	37.789677	
1020624	3729	No	621	62.0	37.777791	
845301	4218	No	1074	323.0	37.798014	
842398	1792	No	495	130.0	37.757369	
1489056	2362	No	737	22.0	37.789756	

In [7]:

```
df.tail()
```

Out[7]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	e
1538081	1508	Yes	887	215.0	37.822547	
1538082	629	No	387	79.0	37.773492	
1538083	2070	No	480	21.0	37.789625	
1538084	2556	No	503	93.0	37.770407	
1538085	2144	No	192	215.0	37.822547	

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538086 entries, 0 to 1538085
Data columns (total 16 columns):
bike_id                    1538086 non-null int64
bike_share_for_all_trip    1018386 non-null object
duration_sec               1538086 non-null int64
end_station_id             1532841 non-null float64
end_station_latitude       1538086 non-null float64
end_station_longitude      1538086 non-null float64
end_station_name           1532841 non-null object
end_time                   1538086 non-null object
member_birth_year          1400419 non-null float64
member_gender              1400760 non-null object
start_station_id           1532841 non-null float64
start_station_latitude     1538086 non-null float64
start_station_longitude    1538086 non-null float64
start_station_name         1532841 non-null object
start_time                 1538086 non-null object
user_type                  1538086 non-null object
dtypes: float64(7), int64(2), object(7)
memory usage: 187.8+ MB
```

- start_time and end_time to be converted into datetime format
- start_station_id ,end_station_id ,bike_id to be converted into string format

In [9]:

```
df.describe()
```

Out[9]:

	bike_id	duration_sec	end_station_id	end_station_latitude	end_station_longitude
count	1.538086e+06	1.538086e+06	1.532841e+06	1.538086e+06	1.538086e+06
mean	2.020603e+03	9.573795e+02	1.056254e+02	3.776912e+01	-1.223547e+02
std	1.152290e+03	2.891834e+03	9.259640e+01	9.756527e-02	1.517343e-01
min	1.000000e+01	6.100000e+01	3.000000e+00	3.728000e+01	-1.224443e+02
25%	1.045000e+03	3.610000e+02	2.700000e+01	3.777241e+01	-1.224103e+02
50%	2.072000e+03	5.690000e+02	7.700000e+01	3.778240e+01	-1.223974e+02
75%	2.952000e+03	8.970000e+02	1.710000e+02	3.779539e+01	-1.223881e+02
max	4.307000e+03	8.636900e+04	3.570000e+02	4.551000e+01	-7.357000e+01

In [10]:

```
df.isnull().sum()
```

Out[10]:

```
bike_id                      0
bike_share_for_all_trip    519700
duration_sec                 0
end_station_id            5245
end_station_latitude         0
end_station_longitude        0
end_station_name          5245
end_time                     0
member_birth_year         137667
member_gender             137326
start_station_id          5245
start_station_latitude       0
start_station_longitude      0
start_station_name        5245
start_time                   0
user_type                    0
dtype: int64
```

In [11]:

```
df.duplicated().sum()
```

Out[11]:

```
0
```

What is the structure of your dataset?

There are 1538086 Ford GoBike trips in the dataset with 16 columns. Out of 16 columns 9 should be numerical, 2 should be datetime, 4 should be object type and 1 should be boolean type.

What is/are the main feature(s) of interest in your dataset?

I think the main features of my interest in this dataset will be the trip duration time and the gender of the users.

What features in the dataset do you think will help support your investigation into your feature(s) of interest?

I think the features that will support my investigation in the data set will be the type of user , age of the user and start time of the trip.

Cleaning

I have found some quality issues in the data ,that need to be taken care of and below is the list of the quality issues that I will clean from the data:

- The data has missing values present in it
- Bike share for all trips columns has maximum number of null values and the null values should be replaced by NA(Not applicable)
- Member birth is in the float format, but should be converted to integer format
- Member birthdates have lots of missing values
- Start/end times are not in the timestamp format
- start_station_id and end_station_id should be converted into string format

In [12]:

```
#creating a copy of the file as df_clean
df_clean = df.copy()
```

Define: Converting several of the columns to the appropriate data type:

- Starting and ending times to the timestamp format
- start station ID, end station ID and bikeid to object format

Code

In [13]:

```
#converting start_time and end_time into datetime format
df_clean.start_time = pd.to_datetime(df_clean.start_time)
df_clean.end_time = pd.to_datetime(df_clean.end_time)
```

In [14]:

```
#converting start_station_id , end_station_id and bike_id to string
df_clean.start_station_id = df_clean.start_station_id.astype(str)
df_clean.end_station_id = df_clean.end_station_id.astype(str)
df_clean.bike_id = df_clean.bike_id.astype(str)
```

Test

In [15]:

```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538086 entries, 0 to 1538085
Data columns (total 16 columns):
bike_id                  1538086 non-null object
bike_share_for_all_trip  1018386 non-null object
duration_sec             1538086 non-null int64
end_station_id           1538086 non-null object
end_station_latitude     1538086 non-null float64
end_station_longitude    1538086 non-null float64
end_station_name         1532841 non-null object
end_time                 1538086 non-null datetime64[ns]
member_birth_year        1400419 non-null float64
member_gender            1400760 non-null object
start_station_id         1538086 non-null object
start_station_latitude   1538086 non-null float64
start_station_longitude  1538086 non-null float64
start_station_name       1532841 non-null object
start_time               1538086 non-null datetime64[ns]
user_type                1538086 non-null object
dtypes: datetime64[ns](2), float64(5), int64(1), object(8)
memory usage: 187.8+ MB
```

In [16]:

```
df_clean.isnull().sum()
```

Out[16]:

```
bike_id                       0
bike_share_for_all_trip    519700
duration_sec                    0
end_station_id                  0
end_station_latitude            0
end_station_longitude           0
end_station_name             5245
end_time                        0
member_birth_year          137667
member_gender              137326
start_station_id                0
start_station_latitude          0
start_station_longitude         0
start_station_name           5245
start_time                      0
user_type                       0
dtype: int64
```

Bike_Share

In [17]:

```
df_clean.bike_share_for_all_trip.describe()
```

Out[17]:

```
count     1018386
unique          2
top            No
freq       934032
Name: bike_share_for_all_trip, dtype: object
```

In [18]:

```
df_clean.bike_share_for_all_trip.value_counts(dropna = False)
```

Out[18]:

```
No     934032
NaN    519700
Yes     84354
Name: bike_share_for_all_trip, dtype: int64
```

In [19]:

```
df_clean[df_clean["bike_share_for_all_trip"].isnull()].sample(20)
```

Out[19]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	er
499534	2069	NaN	1044	24.0	37.789677	
609560	1953	NaN	639	67.0	37.776639	
667754	1112	NaN	280	212.0	37.824931	
747907	2478	NaN	185	56.0	37.773414	
654502	1374	NaN	1718	324.0	37.788300	
672639	986	NaN	609	40.0	37.779208	
564774	522	NaN	384	182.0	37.809013	
799634	278	NaN	257	90.0	37.771058	
416963	257	NaN	1446	20.0	37.791300	
416222	598	NaN	531	45.0	37.781752	
662638	1738	NaN	765	24.0	37.789677	
649078	230	NaN	1245	58.0	37.776619	
386218	2751	NaN	1088	90.0	37.771058	
802530	1087	NaN	475	19.0	37.788975	
368812	3110	NaN	682	4.0	37.785881	
649313	2241	NaN	410	45.0	37.781752	
580233	2162	NaN	984	102.0	37.766883	
342062	3328	NaN	933	36.0	37.783830	
439039	2519	NaN	406	163.0	37.797320	
442884	105	NaN	114	22.0	37.789756	

In [20]:

```
#Filling NaN values with NA (Not applicable), since it maybe the case that the bike
#sharing service was started from 2018 and most of the mising values are in 2017
df_clean.bike_share_for_all_trip.fillna("NA" ,inplace = True)
```

Test

In [21]:

```
df_clean.isnull().sum()
```

Out[21]:

```
bike_id                        0
bike_share_for_all_trip        0
duration_sec                   0
end_station_id                 0
end_station_latitude           0
end_station_longitude          0
end_station_name            5245
end_time                       0
member_birth_year         137667
member_gender             137326
start_station_id               0
start_station_latitude         0
start_station_longitude        0
start_station_name          5245
start_time                     0
user_type                      0
dtype: int64
```

In [22]:

```
df_clean[df_clean["member_gender"].isnull()].sample(20)
```

Out[22]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	e
703845	1167	NA	1340	144.0	37.747300	
569618	1011	NA	605	144.0	37.747300	
469338	1253	NA	4752	16.0	37.794130	
562410	3036	NA	5080	6.0	37.804770	
1272247	2853	No	3487	6.0	37.804770	
956345	2496	No	374	50.0	37.780526	
376091	1604	NA	484	67.0	37.776639	
609097	674	NA	1381	44.0	37.781074	
1199127	251	No	573	249.0	37.858473	
1260301	2186	No	554	15.0	37.795392	
1259970	3138	No	9428	8.0	37.799953	
354380	195	NA	224	56.0	37.773414	
563899	1991	NA	3064	120.0	37.761420	
496132	102	NA	472	308.0	37.336802	
228210	3353	No	18665	162.0	37.800516	
673841	1638	NA	7430	114.0	37.764478	
969632	2089	No	745	284.0	37.784872	
718273	692	NA	1515	15.0	37.795392	
397940	1530	NA	354	121.0	37.759210	
1210016	2490	No	831	50.0	37.780526	

In [23]:

```
df_clean.member_gender.value_counts()
```

Out[23]:

```
Male      1043512
Female     336222
Other       21026
Name: member_gender, dtype: int64
```

In [24]:

```
df_clean.member_birth_year.describe()
```

Out[24]:

```
count    1.400419e+06
mean     1.981805e+03
std      1.056463e+01
min      1.881000e+03
25%      1.976000e+03
50%      1.984000e+03
75%      1.989000e+03
max      2.000000e+03
Name: member_birth_year, dtype: float64
```

In [25]:

```
#checking the % of missing value in the data
137667/len(df_clean)*100
```

Out[25]:

```
8.950539826771715
```

In [26]:

```
df_clean.dropna(inplace = True)
```

Test

In [27]:

```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1395361 entries, 0 to 1538085
Data columns (total 16 columns):
bike_id                    1395361 non-null object
bike_share_for_all_trip    1395361 non-null object
duration_sec               1395361 non-null int64
end_station_id             1395361 non-null object
end_station_latitude       1395361 non-null float64
end_station_longitude      1395361 non-null float64
end_station_name           1395361 non-null object
end_time                   1395361 non-null datetime64[ns]
member_birth_year          1395361 non-null float64
member_gender              1395361 non-null object
start_station_id           1395361 non-null object
start_station_latitude     1395361 non-null float64
start_station_longitude    1395361 non-null float64
start_station_name         1395361 non-null object
start_time                 1395361 non-null datetime64[ns]
user_type                  1395361 non-null object
dtypes: datetime64[ns](2), float64(5), int64(1), object(8)
memory usage: 181.0+ MB
```

Define : Creating a new column named member_age which will contain the age of the users

Code

In [28]:

```
# Subtracting the member_birth_year from the current year to get the ages of the member
s
df_clean['member_age'] = 2020 - df_clean['member_birth_year']
```

Test

In [29]:

```
df_clean.head()
```

Out[29]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	end_sta
0	1035	No	598	114.0	37.764478	
1	1673	No	943	324.0	37.788300	
2	3498	No	18587	15.0	37.795392	
3	3129	No	18558	15.0	37.795392	
4	1839	Yes	885	297.0	37.322980	

Define : Checking the different percentiles of age group in the member_age column

Code

In [30]:

```
df_clean.member_age.describe()
```

Out[30]:

```
count    1.395361e+06
mean     3.821268e+01
std      1.056918e+01
min      2.000000e+01
25%      3.100000e+01
50%      3.600000e+01
75%      4.400000e+01
max      1.390000e+02
Name: member_age, dtype: float64
```

In [31]:

```
df_clean.member_age.median()
```

Out[31]:

```
36.0
```

In [32]:

```
df_clean.member_age.describe(percentiles = [.25 , .75 ,.99])
```

Out[32]:

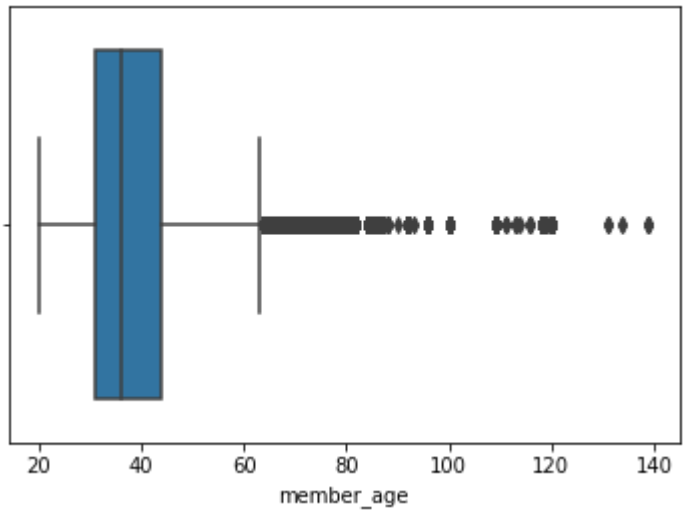
```
count    1.395361e+06
mean     3.821268e+01
std      1.056918e+01
min      2.000000e+01
25%      3.100000e+01
50%      3.600000e+01
75%      4.400000e+01
99%      6.800000e+01
max      1.390000e+02
Name: member_age, dtype: float64
```

In [33]:

```
sns.boxplot(x = df_clean["member_age"])
```

Out[33]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2406db98908>
```



In [34]:

```
#calculating the iqr
q1  = df_clean.member_age.quantile(0.25)
q3  = df_clean.member_age.quantile(0.75)
iqr = q3 - q1

lr = q1 - (1.5 * iqr)
ur = q3 + (1.5 * iqr)
print("The lower range is {} and the upper range is {}".format(lr,ur))
```

```
The lower range is 11.5 and the upper range is 63.5
```

In [35]:

```
#taking ages less than 64 in our dataset
df_clean = df_clean[df_clean["member_age"] < 64]
```

Test

In [36]:

```
df_clean.sample(10)
```

Out[36]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	e
142144	3725	No	1410	160.0	37.805318	
713554	2658	NA	932	85.0	37.770083	
1377458	1296	No	284	176.0	37.828410	
345369	2444	NA	338	30.0	37.776598	
598794	2864	NA	426	66.0	37.778742	
1443806	2388	No	694	30.0	37.776598	
1463275	2477	No	3430	64.0	37.776754	
177665	2457	No	579	5.0	37.783899	
492056	424	NA	566	22.0	37.789756	
477764	3121	NA	453	108.0	37.764710	

Define : Extracting the Month, weekday and year-month from the dataframe and adding the values in a new columns

Code

In [37]:

```
df_clean['start_month_year'] = pd.to_datetime(df_clean['start_time']).dt.to_period('M')
```

Test

In [38]:

```
df_clean.head()
```

Out[38]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	end_sta
0	1035	No	598	114.0	37.764478	
1	1673	No	943	324.0	37.788300	
2	3498	No	18587	15.0	37.795392	
3	3129	No	18558	15.0	37.795392	
4	1839	Yes	885	297.0	37.322980	

Define : Creating 2 columns for the month and month number for the start time of the trip

Code

In [39]:

```
# Start time month (January - December)
df_clean['start_time_month'] = df_clean['start_time'].dt.strftime('%B')
```

In [40]:

```
# Start time weekday
df_clean['start_time_weekday'] = df_clean['start_time'].dt.strftime('%a')
```

Test

In [41]:

```
#checking
df_clean.sample(20)
```

Out[41]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	e
672043	2729	NA	401	175.0	37.835946	
74972	1957	No	220	305.0	37.342725	
1401901	3390	No	280	79.0	37.773492	
599581	126	NA	280	16.0	37.794130	
1497253	389	No	1299	13.0	37.794231	
1062312	2683	Yes	375	231.0	37.808750	
1438313	766	No	13771	307.0	37.332692	
493549	3175	NA	438	60.0	37.774520	
729816	213	NA	319	99.0	37.767037	
625815	2408	NA	1467	285.0	37.783521	
366946	2908	NA	780	89.0	37.769218	
785242	1468	NA	1048	304.0	37.348759	
710813	2574	NA	1387	109.0	37.763316	
1284544	3464	No	796	212.0	37.824931	
1307135	2194	No	1229	203.0	37.795195	
557386	1749	NA	443	3.0	37.786375	
1091160	3627	No	421	176.0	37.828410	
1007614	3418	No	591	30.0	37.776598	
1379433	2852	No	1624	139.0	37.751017	
405806	325	NA	428	195.0	37.812314	

Define : Creating a column for start and end time hour

Code

In [42]:

```
# Start and end time hour
df_clean['start_time_hour'] = df_clean['start_time'].dt.hour
df_clean['end_time_hour'] = df_clean['end_time'].dt.hour
```

Test

In [43]:

```
df_clean.sample(20)
```

Out[43]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	e
1256266	2426	Yes	573	286.0	37.336466	
1464748	1321	No	1448	126.0	37.761634	
883101	4008	No	435	101.0	37.766008	
1027298	2527	No	496	181.0	37.811377	
1319152	1941	No	756	81.0	37.775880	
1283415	2290	No	467	114.0	37.764478	
1082775	3520	No	607	7.0	37.804562	
1170926	292	No	455	66.0	37.778742	
1296350	2005	Yes	656	85.0	37.770083	
356485	3045	NA	1051	114.0	37.764478	
529215	345	NA	384	176.0	37.828410	
586404	2976	NA	1012	63.0	37.775910	
1106618	1303	No	491	30.0	37.776598	
93008	698	No	277	182.0	37.809013	
498433	126	NA	504	62.0	37.777791	
286121	2763	No	670	67.0	37.776639	
1071610	1930	No	866	21.0	37.789625	
780733	1831	NA	650	304.0	37.348759	
1135237	670	No	411	144.0	37.747300	
300670	4031	Yes	419	133.0	37.755213	

20 rows × 22 columns

Define : Splitting the member ages into various age groups

Code

In [44]:

```
df_clean['age_bins'] = df_clean['member_age'].apply(lambda x: '15 - 25' if 15<x<=25
                                                    else '26 - 35' if 25<x<=35
                                                    else '36 - 45' if 35<x<=45
                                                    else '46 - 55' if 45<x<=55
                                                    else '56 - 66' if 55<x<=66
                                                    else '66 - 75' if 66<x<=75
                                                     else x)
```

Test

In [45]:

```
df_clean.age_bins.value_counts()
```

Out[45]:

```
26 - 35    594719
36 - 45    416647
46 - 55    194996
15 - 25     81820
56 - 66     76370
Name: age_bins, dtype: int64
```

Define : Converting duration_sec into minutes in column duration_min

Code

In [46]:

```
# Duration in seconds to duration in minutes
df_clean['duration_min'] = df_clean['duration_sec']/60
df_clean['duration_min'] = df_clean['duration_min'].astype(int)
```

In [47]:

```
#calculating the iqr
q1  = df_clean.duration_min.quantile(0.25)
q3  = df_clean.duration_min.quantile(0.75)
iqr = q3 - q1

lr = q1 - (1.5 * iqr)
ur = q3 + (1.5 * iqr)
print("The lower range is {} and the upper range is {}".format(lr,ur))
```

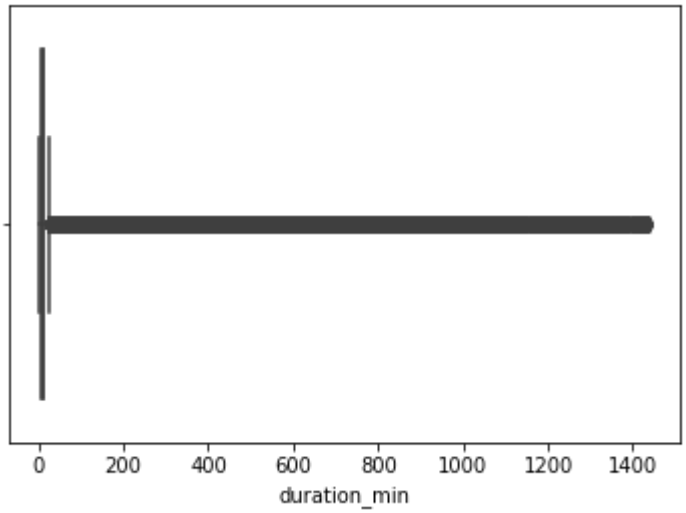
```
The lower range is -7.0 and the upper range is 25.0
```

In [48]:

```
sns.boxplot(df_clean["duration_min"])
```

Out[48]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2406da7c630>
```



In [49]:

```
df_clean["duration_min"].describe(percentiles = [.25 , .50 , .75 ,.99])
```

Out[49]:

```
count    1.364552e+06
mean     1.277215e+01
std      3.703711e+01
min      1.000000e+00
25%      5.000000e+00
50%      9.000000e+00
75%      1.300000e+01
99%      6.800000e+01
max      1.437000e+03
Name: duration_min, dtype: float64
```

In [50]:

```
df_clean[df_clean["duration_min"]  > 68]
```

Out[50]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	e
2	3498	No	18587	15.0	37.795392	
3	3129	No	18558	15.0	37.795392	
183	1268	No	4196	162.0	37.800516	
898	2016	No	36573	47.0	37.780955	
1621	3271	No	7013	248.0	37.855956	
1877	1461	No	5952	56.0	37.773414	
1948	3560	No	20498	61.0	37.776513	
2036	2079	No	5538	24.0	37.789677	
2337	3698	No	8397	75.0	37.773793	
2648	2589	No	72175	75.0	37.773793	
2662	3742	No	5428	81.0	37.775880	
3475	2261	No	41993	127.0	37.756708	
4029	3349	No	38316	204.0	37.840186	
4385	3706	Yes	4241	310.0	37.335885	
4958	3505	No	20809	258.0	37.872355	
5028	2648	No	5104	3.0	37.786375	
5424	3200	Yes	10765	254.0	37.880222	
5640	11	Yes	4925	183.0	37.808702	
5655	1410	No	30584	114.0	37.764478	
5679	2705	No	34412	246.0	37.869060	
5828	3049	Yes	5013	251.0	37.870555	
5991	750	No	18248	6.0	37.804770	
6029	326	No	5172	85.0	37.770083	

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	e
6030	2730	No	4756	323.0	37.798014	
6112	2026	Yes	4245	243.0	37.869360	
6183	11	Yes	5911	164.0	37.814988	
6299	2529	Yes	10981	43.0	37.778768	
6971	822	Yes	5119	256.0	37.875112	
7042	3626	No	7331	213.0	37.823847	
7272	150	No	57357	189.0	37.839649	
...	...	...	...	...	...	
1536311	714	No	13429	132.0	37.751819	
1536315	1427	No	4164	70.0	37.773311	
1536357	3150	No	9784	47.0	37.780955	
1536369	2316	No	5018	212.0	37.824931	
1536375	1497	No	7990	70.0	37.773311	
1536744	1939	No	10419	154.0	37.841924	
1536956	3331	No	4208	157.0	37.846784	
1537023	3121	No	7678	323.0	37.798014	
1537048	3413	No	7486	259.0	37.866249	
1537091	1094	No	15187	230.0	37.810743	
1537146	2881	No	6918	56.0	37.773414	
1537175	3114	No	5782	70.0	37.773311	
1537176	2658	No	16436	41.0	37.781270	
1537177	2729	No	15021	41.0	37.781270	
1537192	307	No	5637	239.0	37.868813	
1537303	3696	No	5109	250.0	37.874014	
1537332	609	No	4580	70.0	37.773311	

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	e
1537408	1167	No	6125	98.0	37.765052	
1537415	2556	Yes	4558	74.0	37.776435	
1537431	530	No	4778	259.0	37.866249	
1537438	3086	No	5703	272.0	37.850578	
1537592	2050	Yes	5093	8.0	37.799953	
1537640	2038	Yes	6510	299.0	37.323678	
1537769	1991	No	7519	6.0	37.804770	
1537770	2256	No	7530	6.0	37.804770	
1537771	3660	No	7465	6.0	37.804770	
1537774	1977	No	7412	6.0	37.804770	
1537848	1326	No	6917	182.0	37.809013	
1537863	1143	No	4396	152.0	37.835632	
1537932	2873	No	25719	159.0	37.816060	

13532 rows × 24 columns

In [51]:

```
#removing rows that have duration_min > 150 minutes
df_clean = df_clean[df_clean["duration_min"] < 150]
```

Test

In [52]:

```
#checking
df_clean.sample(10)
```

Out[52]:

	bike_id	bike_share_for_all_trip	duration_sec	end_station_id	end_station_latitude	e
6332	2085	No	406	15.0	37.795392	
932397	3851	No	209	72.0	37.772406	
1061955	599	No	756	31.0	37.783813	
1464921	602	No	1061	130.0	37.757369	
53739	1359	No	422	197.0	37.808848	
810839	274	NA	518	15.0	37.795392	
953792	530	No	205	182.0	37.809013	
877907	3153	No	869	37.0	37.785000	
752727	2643	NA	342	15.0	37.795392	
460666	2052	NA	1044	30.0	37.776598	

10 rows × 24 columns

In [53]:

```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1358803 entries, 0 to 1538085
Data columns (total 24 columns):
bike_id                     1358803 non-null object
bike_share_for_all_trip     1358803 non-null object
duration_sec                1358803 non-null int64
end_station_id              1358803 non-null object
end_station_latitude        1358803 non-null float64
end_station_longitude       1358803 non-null float64
end_station_name            1358803 non-null object
end_time                    1358803 non-null datetime64[ns]
member_birth_year           1358803 non-null float64
member_gender               1358803 non-null object
start_station_id            1358803 non-null object
start_station_latitude      1358803 non-null float64
start_station_longitude     1358803 non-null float64
start_station_name          1358803 non-null object
start_time                  1358803 non-null datetime64[ns]
user_type                   1358803 non-null object
member_age                  1358803 non-null float64
start_month_year            1358803 non-null object
start_time_month            1358803 non-null object
start_time_weekday          1358803 non-null object
start_time_hour             1358803 non-null int64
end_time_hour               1358803 non-null int64
age_bins                    1358803 non-null object
duration_min                1358803 non-null int32
dtypes: datetime64[ns](2), float64(6), int32(1), int64(3), object(12)
memory usage: 254.0+ MB
```

Saving the cleaned data

In [54]:

```
df_clean.to_csv('fordgo_master_clean.csv', index = False)
```

Univariate Exploration

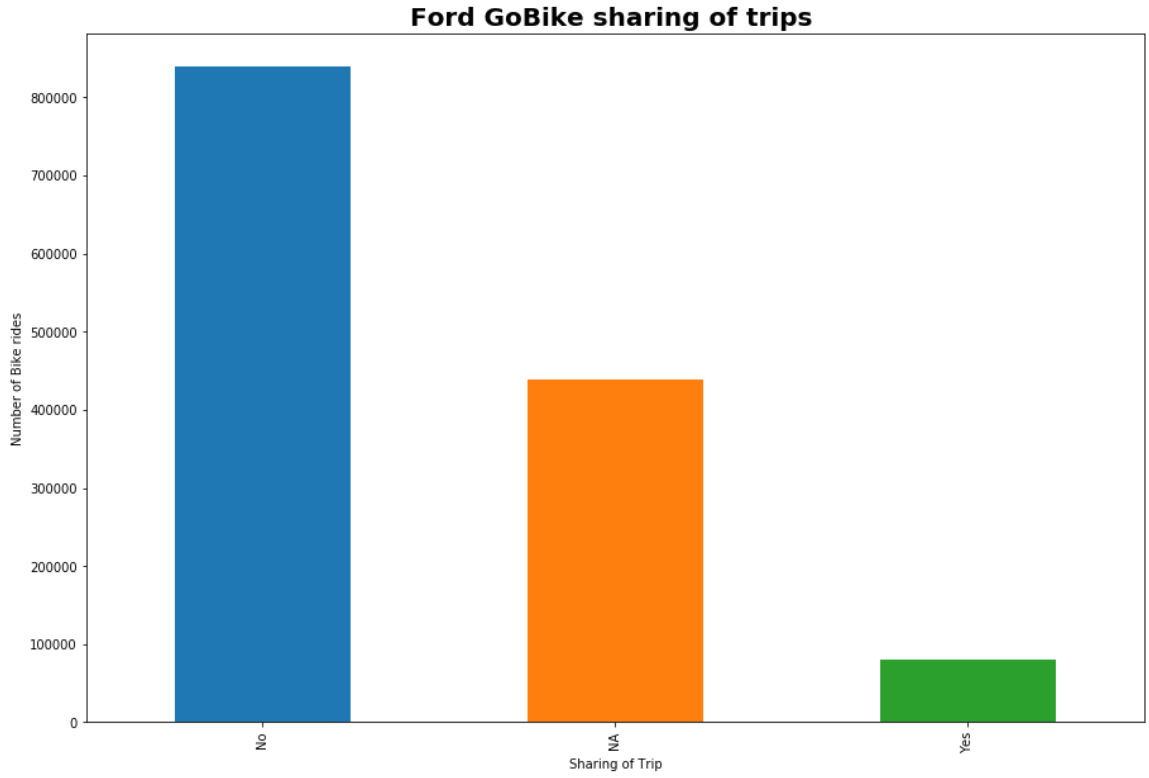
> In this section, I will investigate the distributions of individual variables and find out various insights from them

In [56]:

```
plt.figure(figsize= (15,10))
df_clean.bike_share_for_all_trip.value_counts().plot(kind = "bar")
plt.xlabel("Sharing of Trip")
plt.ylabel("Number of Bike rides")
plt.title("Ford GoBike sharing of trips " , fontsize = 20 , fontweight = "bold")
```

Out[56]:

```
Text(0.5, 1.0, 'Ford GoBike sharing of trips ')
```



Observation 1 : Most number of bike rides are done without sharing the trip , followed by NA(Not Applicable) ,which maybe because the bike sharing data was not collected in the year 2017

In [57]:

```
df_clean.duration_min.describe()
```

Out[57]:

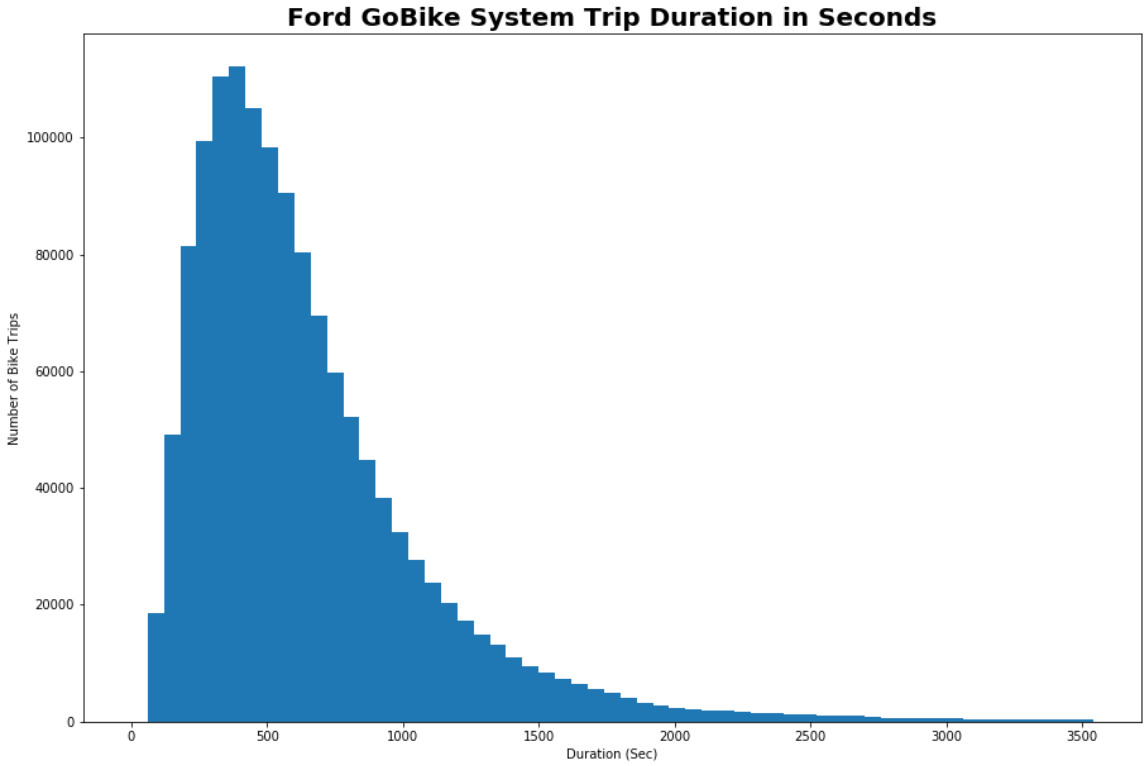
```
count    1.358803e+06
mean     1.094620e+01
std      1.032480e+01
min      1.000000e+00
25%      5.000000e+00
50%      9.000000e+00
75%      1.300000e+01
max      1.490000e+02
Name: duration_min, dtype: float64
```

In [58]:

```
# Duration of the bike rides - in seconds
bin_edges = np.arange(0, 3600, 60)

plt.figure(figsize= (15,10))
plt.hist(data = df_clean, x = 'duration_sec', bins = bin_edges);

plt.title("Ford GoBike System Trip Duration in Seconds", fontsize=20, fontweight='bold'
)
plt.xlabel('Duration (Sec)')
plt.ylabel('Number of Bike Trips');
```



Observation 2 : Most of the trip fall around 500 seconds and the data is skewed to the right with most of the data ranging between 250-750 seconds and ranging till 3500 seconds

In [59]:

```
df_clean.duration_min.describe()
```

Out[59]:

```
count    1.358803e+06
mean     1.094620e+01
std      1.032480e+01
min      1.000000e+00
25%      5.000000e+00
50%      9.000000e+00
75%      1.300000e+01
max      1.490000e+02
Name: duration_min, dtype: float64
```

In [60]:

```
df_clean.duration_min.quantile(.99)
```

Out[60]:

```
50.0
```

In [61]:

```
df_clean.duration_min.describe()
```

Out[61]:

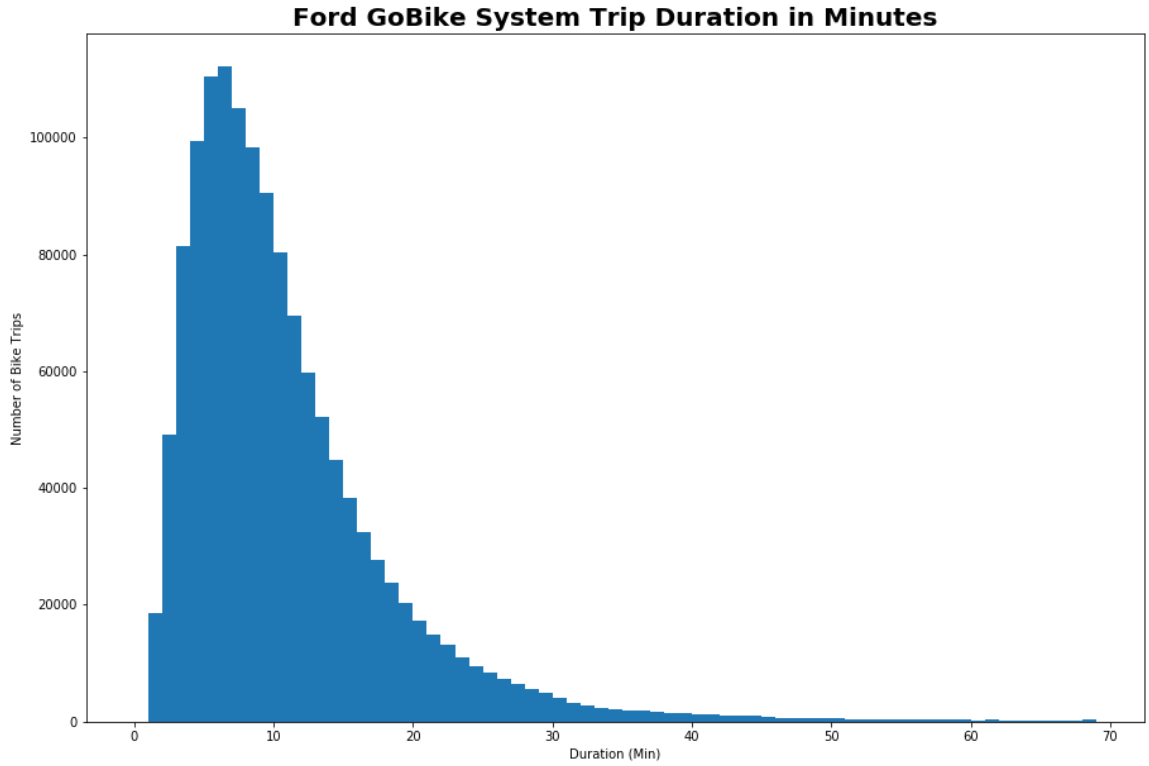
```
count    1.358803e+06
mean     1.094620e+01
std      1.032480e+01
min      1.000000e+00
25%      5.000000e+00
50%      9.000000e+00
75%      1.300000e+01
max      1.490000e+02
Name: duration_min, dtype: float64
```

In [62]:

```
# Duration of the bike rides - in minutes
bin_edges = np.arange(0, 70 ,1)

plt.figure(figsize= (15,10))
plt.hist(data = df_clean, x = 'duration_min', bins = bin_edges);

plt.title("Ford GoBike System Trip Duration in Minutes", fontsize = 20, fontweight='bol
d')
plt.xlabel('Duration (Min)')
plt.ylabel('Number of Bike Trips');
```



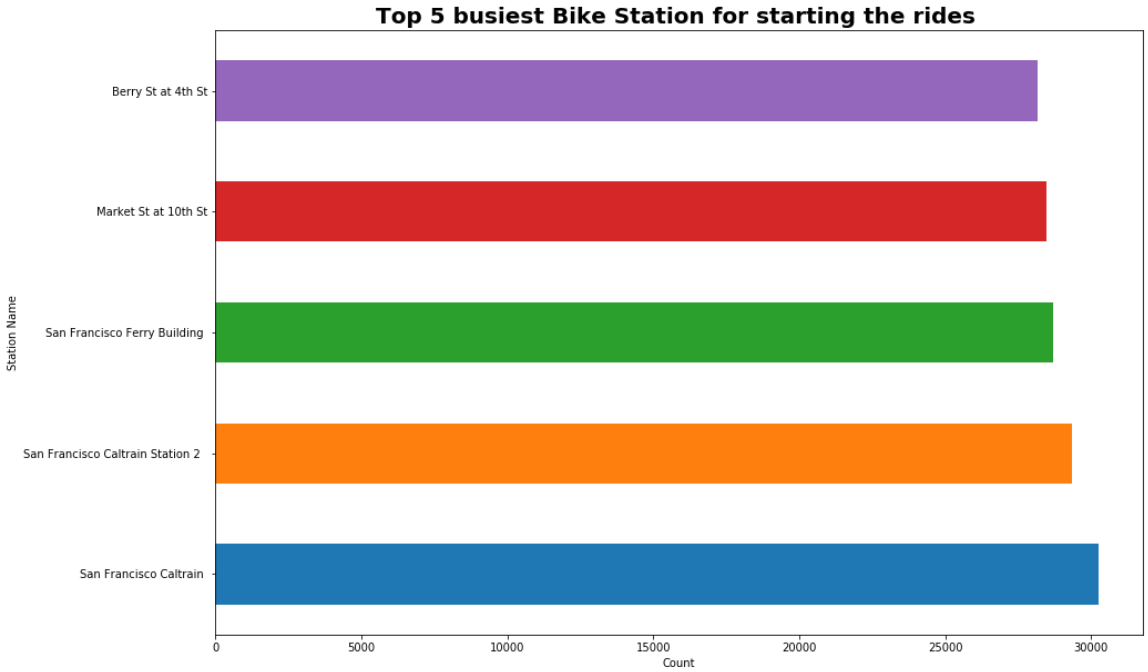
Observation 3 :Most of the trip fall around 10 minutes and the data is skewed to the right with most of the data ranging between 5-20 minutes and ranging till 68 minutes ,which is highly unlikely

In [63]:

```
a = df_clean.start_station_name.value_counts()[:5]
li = a.index.tolist()
new = []
for i in li :
    new.append(i.split("(")[0])

a.index = new
a.plot(kind = "barh" ,figsize = (15,10))

plt.title("Top 5 busiest Bike Station for starting the rides ", fontsize=20, fontweight
='bold')
plt.xlabel('Count')
plt.ylabel('Station Name');
```



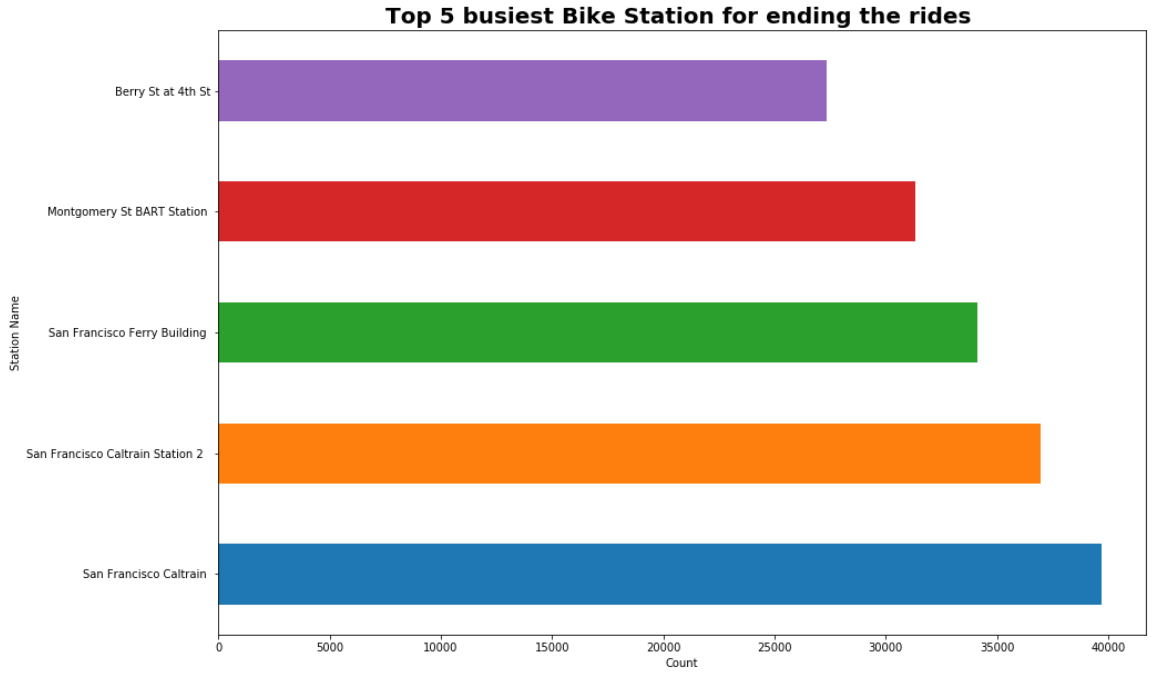
Observation 4 : San Francisco Caltrain is the most busiest station among all stations from where the bike trips start

In [64]:

```
b = df_clean.end_station_name.value_counts()[:5]
li1 = b.index.tolist()
new1 = []
for i in li1 :
    new1.append(i.split("(")[0])

b.index = new1
b.plot(kind = "barh" ,figsize = (15,10))

plt.title("Top 5 busiest Bike Station for ending the rides ", fontsize=20, fontweight=
'bold')
plt.xlabel('Count')
plt.ylabel('Station Name');
```



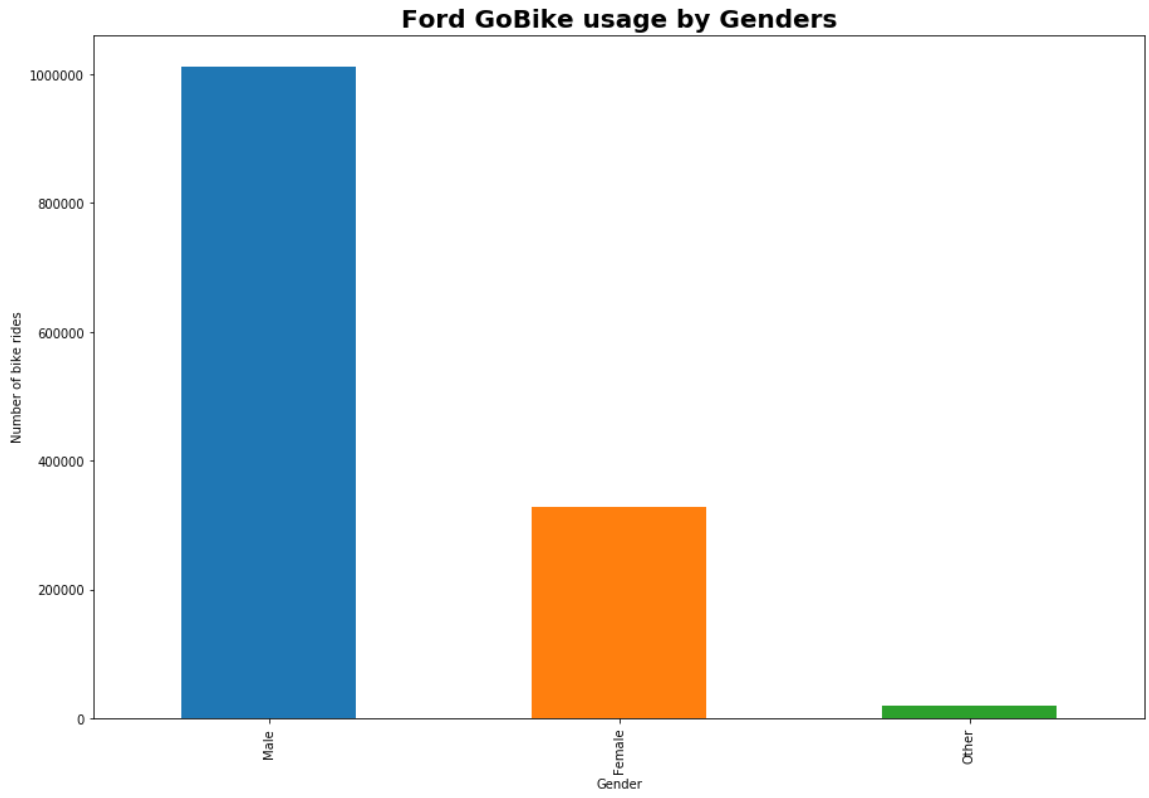
Observation 5 : San Francisco Caltrain is the most busiest station among all stations , where the bike trips end

In [65]:

```
plt.figure(figsize = (15,10))
df_clean.member_gender.value_counts().plot(kind = "bar")
plt.xlabel("Gender")
plt.ylabel("Number of bike rides")
plt.title("Ford GoBike usage by Genders" , fontsize = 20 , fontweight = "bold")
```

Out[65]:

```
Text(0.5, 1.0, 'Ford GoBike usage by Genders')
```



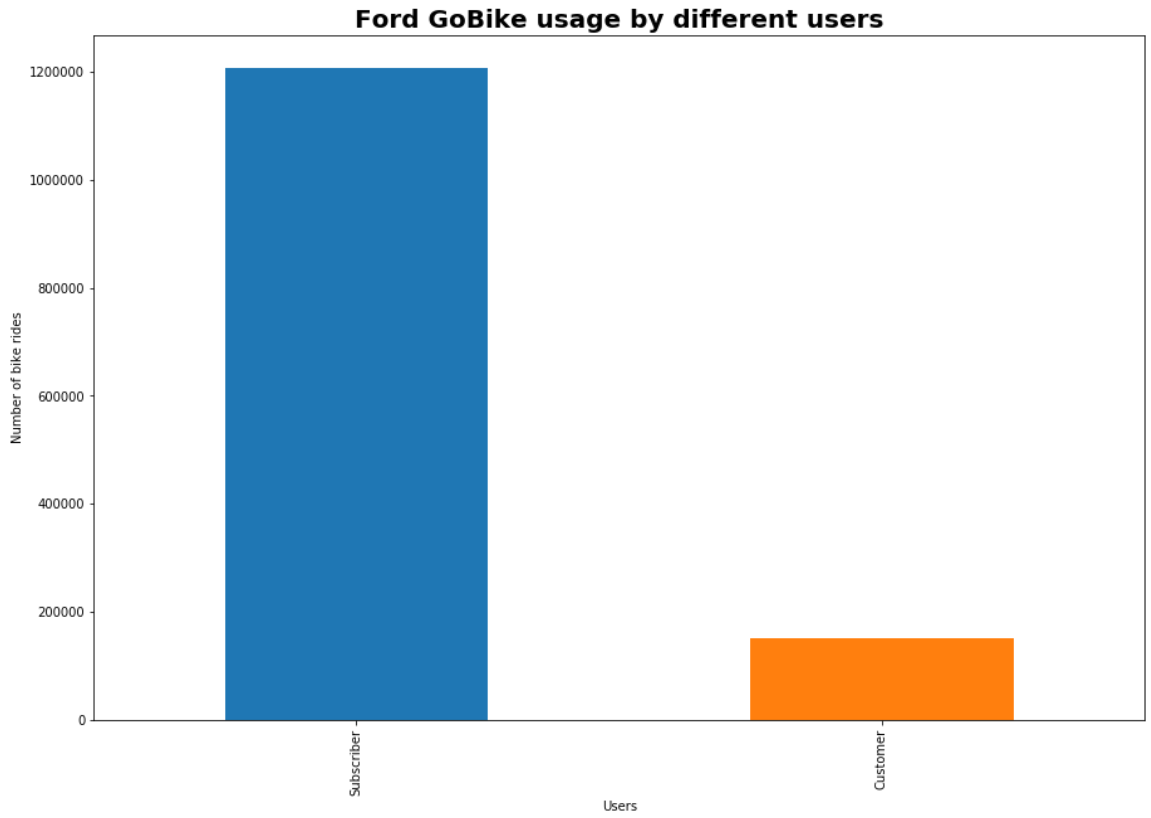
Observation 6 : Most of the users of the bikes service are Males followed by females.

In [66]:

```
plt.figure(figsize = (15,10))
df_clean.user_type.value_counts().plot(kind = "bar")
plt.xlabel("Users")
plt.ylabel("Number of bike rides")
plt.title("Ford GoBike usage by different users" , fontsize = 20 ,fontweight = "bold")
```

Out[66]:

```
Text(0.5, 1.0, 'Ford GoBike usage by different users')
```



Observation 7 : Most of the users of the bike service are the Subscribers.Customers use the bike service very less as compared to the Subscribers

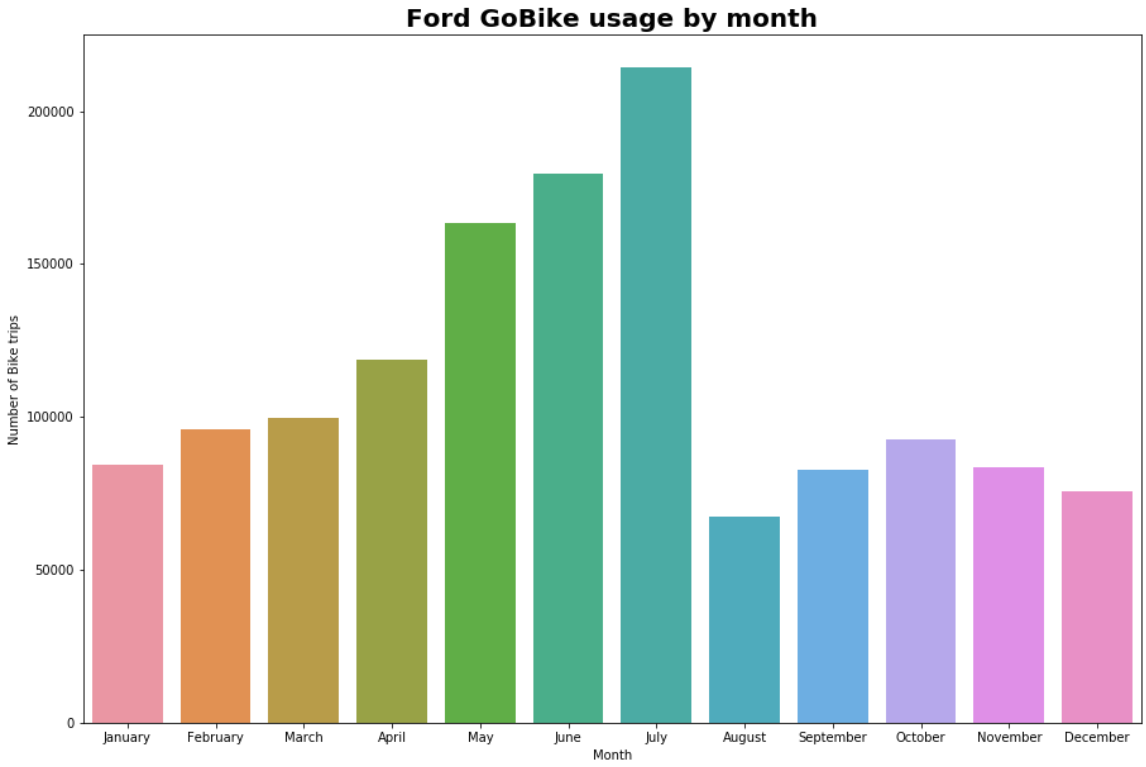
In [67]:

```
#usage by month
month = ["January" , "February" , "March" , "April" , "May" , "June" , "July" , "Augus
t" , "September" ,"October" ,"November" ,"December"]
plt.figure(figsize= (15,10))
sns.countplot(x = "start_time_month" , data = df_clean  ,order=month)

plt.xlabel("Month")
plt.ylabel("Number of Bike trips")
plt.title('Ford GoBike usage by month' , fontweight = "bold" , fontsize = 20)
```

Out[67]:

```
Text(0.5, 1.0, 'Ford GoBike usage by month')
```



Observation 8 : July is the peak month in which the bike service is used the most.Most of the trips are done from the month of April to July, which are the early summer months.

In [68]:

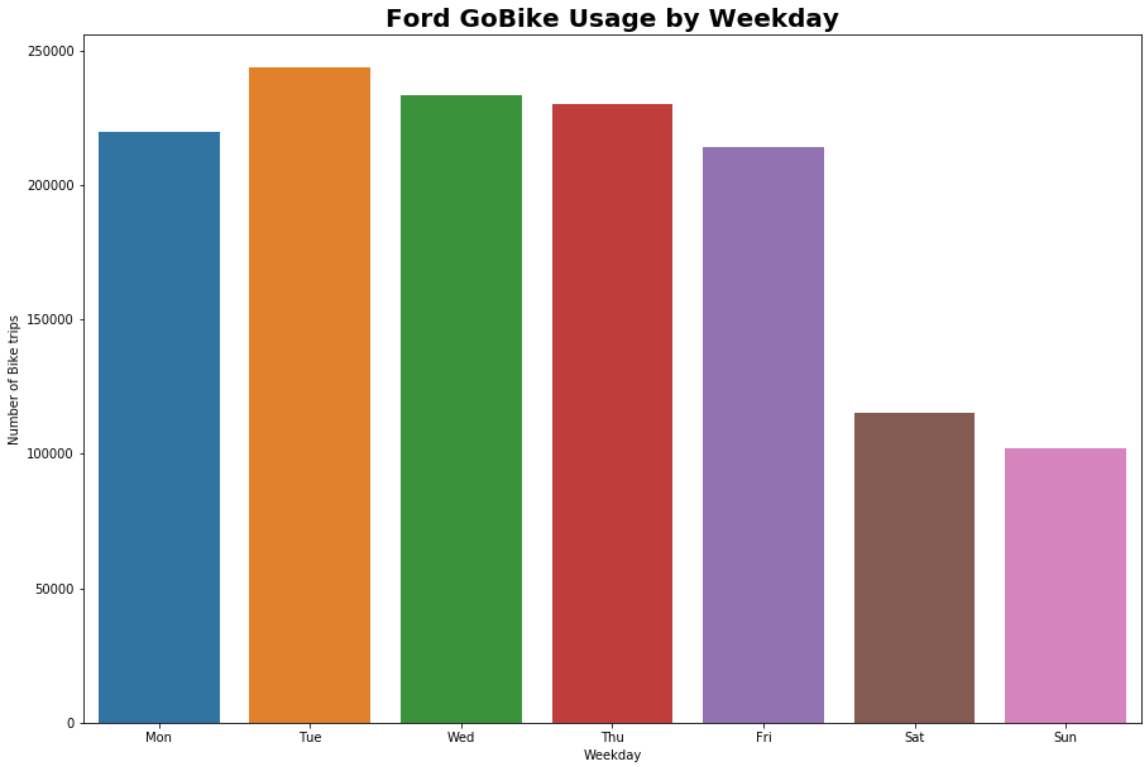
```
#usage by weekdays
weekdays = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']

plt.figure(figsize= (15,10))
sns.countplot(x = df_clean["start_time_weekday"] , data = df_clean ,order = weekdays)

plt.xlabel("Weekday")
plt.ylabel("Number of Bike trips")
plt.title('Ford GoBike Usage by Weekday' , fontweight = "bold" , fontsize = 20)
```

Out[68]:

```
Text(0.5, 1.0, 'Ford GoBike Usage by Weekday')
```



Observation 9 : Weekdays are the most busiest days compared to weekends.Bike service has the highest usage on Tuesday.Wednesday and Thursday have almost same usage of bikes ,followed by Thursday and Monday and then the usage dips on weekend

In [69]:

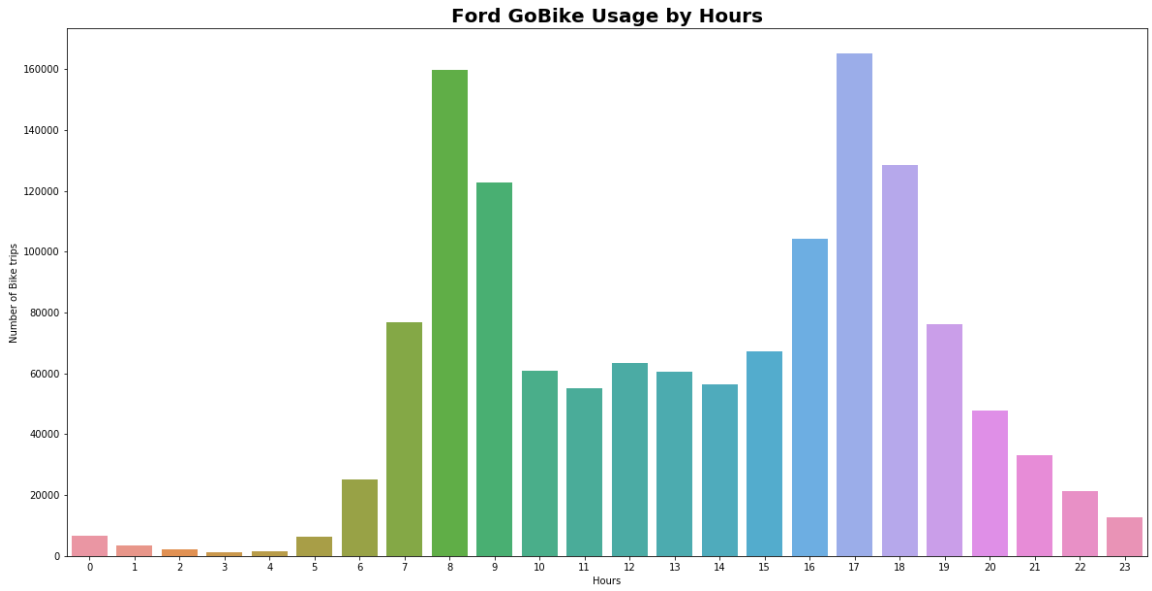
```
#usage by hours

plt.figure(figsize= (20,10))
sns.countplot(x = df_clean["start_time_hour"] , data = df_clean)

plt.xlabel("Hours")
plt.ylabel("Number of Bike trips")
plt.title('Ford GoBike Usage by Hours' , fontweight = "bold" , fontsize = 20)
```

Out[69]:

```
Text(0.5, 1.0, 'Ford GoBike Usage by Hours')
```



Observation 10 : The morning hours(8-9am) and afternoon hours(5-6pm) are the busiest hours throught the day.It maybe because most of the trips are used to commute to office/school

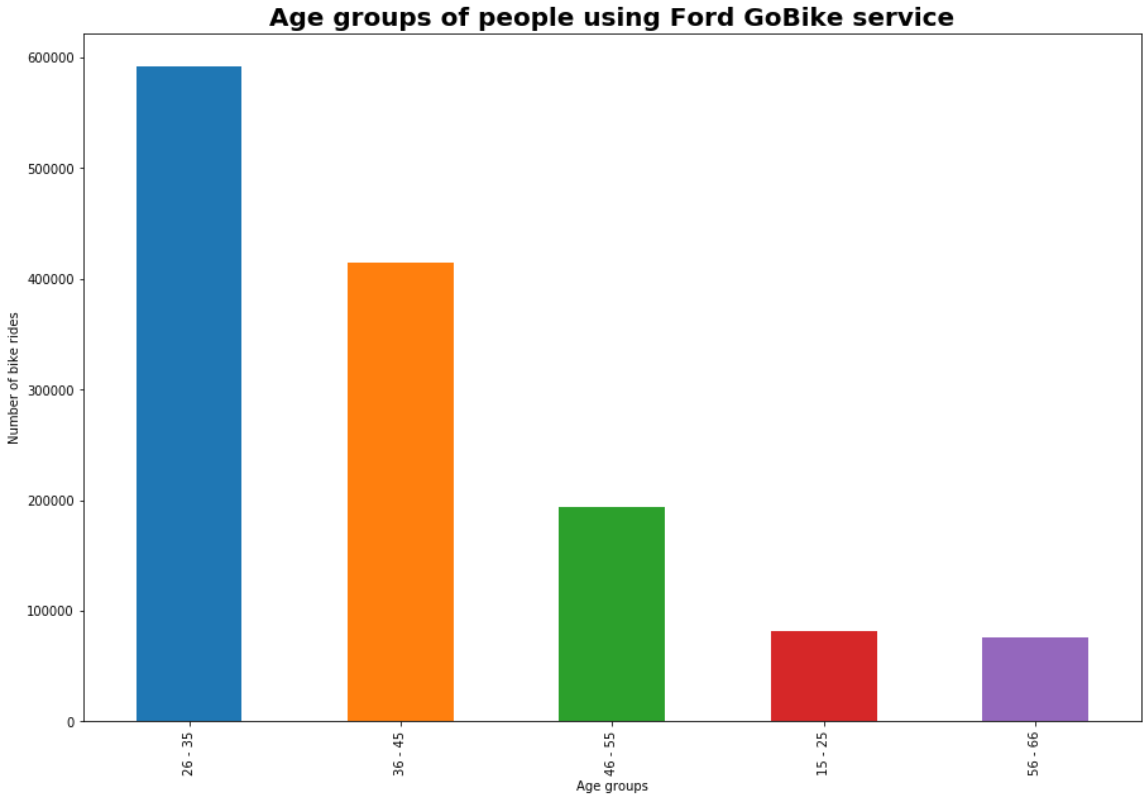
In [70]:

```
plt.figure(figsize = (15,10))
df_clean.age_bins.value_counts().plot(kind = "bar")

plt.xlabel("Age groups")
plt.ylabel("Number of bike rides")
plt.title("Age groups of people using Ford GoBike service" , fontsize = 20 , fontweight
= "bold")
```

Out[70]:

```
Text(0.5, 1.0, 'Age groups of people using Ford GoBike service')
```



Observation 11 : Most of the users of the bike service fall under the age gap of 26-35 years ,followed by 36-45 years.It maybe because the bike service are mostly used by the American working class people

Discuss the distribution(s) of your variable(s) of interest. Were there any unusual points? Did you need to perform any transformations?

> My variables of interest are Trip duration and gender.Trip duration has lots of outtliers ,so I have considered data for which trip duration is less than 68 minutes.Gender column has 3 variables namely Male , Female and Others.Males are the highest number of users followed by females.Trip duration data is skewed to the right.

Of the features you investigated, were there any unusual distributions? Did you perform any operations on the data to tidy, adjust, or change the form of the data? If so, why did you do this?

> I investigated various variables inluding Type of user , Age and Start time of the trip.There are 2 types of users namely Subscribers and Customers.As infered from the data the Subscribers are daily commuters and use the service mostlty on weekdays ,while Customers are more of casual users and use the service mostly on weekends.There were outliers in the age of the data ,the maximum age was 139 years ,which is bizarre and 99% of the data is less than age 68 years.I have taken data where age is less than 64 years using the IQR technique.Start time of the trip ranges from 12am to 11pm ,while most of the trips take place at 8-9am and 5-6pm

Bivariate Exploration

> In this section, I investigate relationships between pairs of variables in the data.

In [71]:

```
#checking for correlation between the quantitaive variables
df_clean[["duration_sec" , "duration_min" ,"member_age" , "start_time_hour" ]].corr()
```

Out[71]:

	duration_sec	duration_min	member_age	start_time_hour
duration_sec	1.000000	0.999610	-0.006863	0.019108
duration_min	0.999610	1.000000	-0.006840	0.019132
member_age	-0.006863	-0.006840	1.000000	-0.061749
start_time_hour	0.019108	0.019132	-0.061749	1.000000

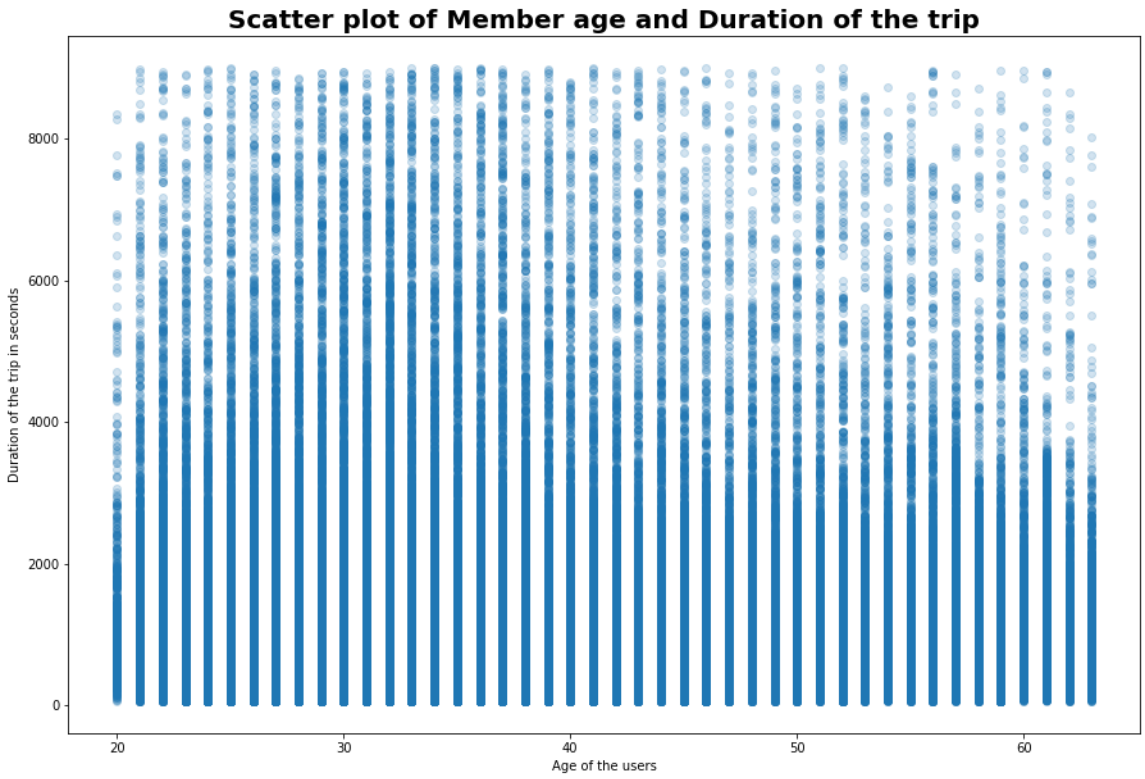
In [72]:

```
plt.figure(figsize = (15 , 10))

plt.scatter(x = df_clean["member_age"] , y = df_clean["duration_sec"] , alpha = 1/5)
plt.xlabel("Age of the users")
plt.ylabel("Duration of the trip in seconds")
plt.title("Scatter plot of Member age and Duration of the trip" , fontweight = "bold" ,
fontsize = 20)
```

Out[72]:

```
Text(0.5, 1.0, 'Scatter plot of Member age and Duration of the trip')
```



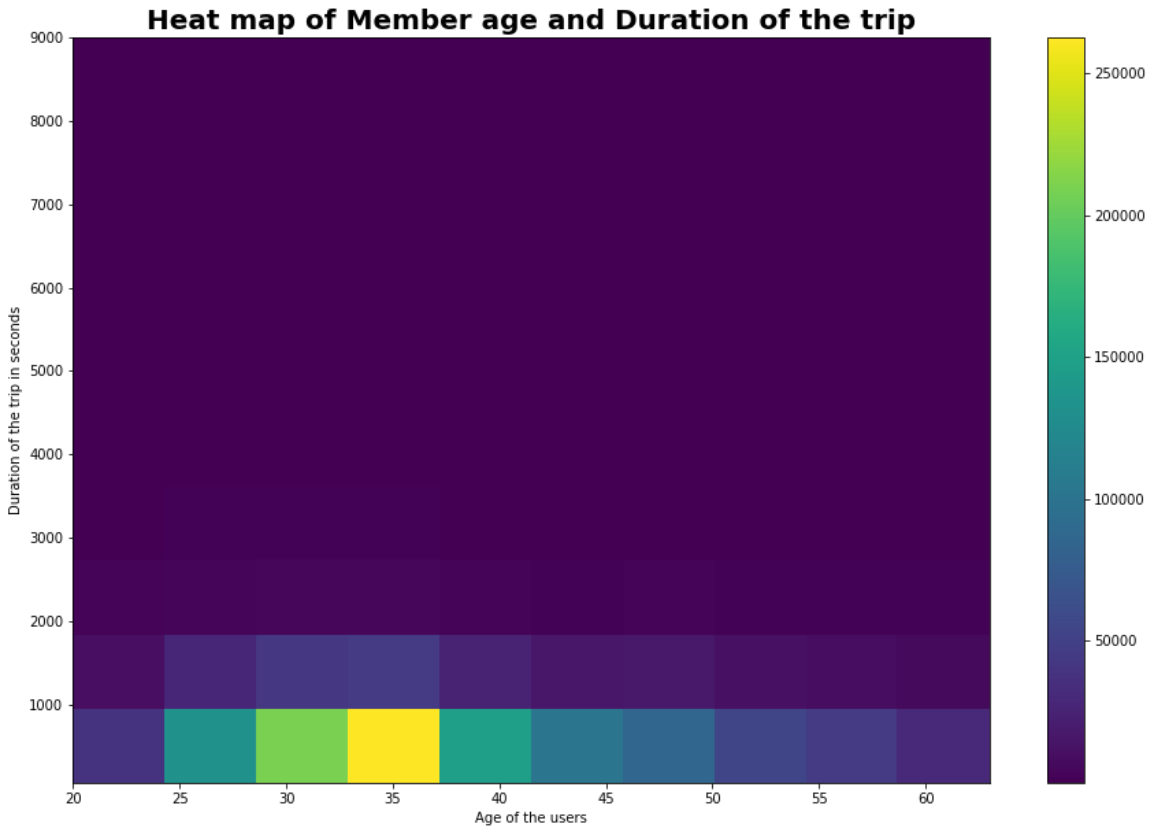
In [73]:

```
plt.figure(figsize = (15 , 10))
plt.hist2d(x = df_clean["member_age"] , y = df_clean["duration_sec"])
plt.colorbar()

plt.xlabel("Age of the users")
plt.ylabel("Duration of the trip in seconds")
plt.title("Heat map of Member age and Duration of the trip" , fontweight = "bold" , fon
tsize = 20)
```

Out[73]:

```
Text(0.5, 1.0, 'Heat map of Member age and Duration of the trip')
```



Observation 12 : The longest duration of the trips are done by the users falling in 30-40 years of age falling below 2000 seconds

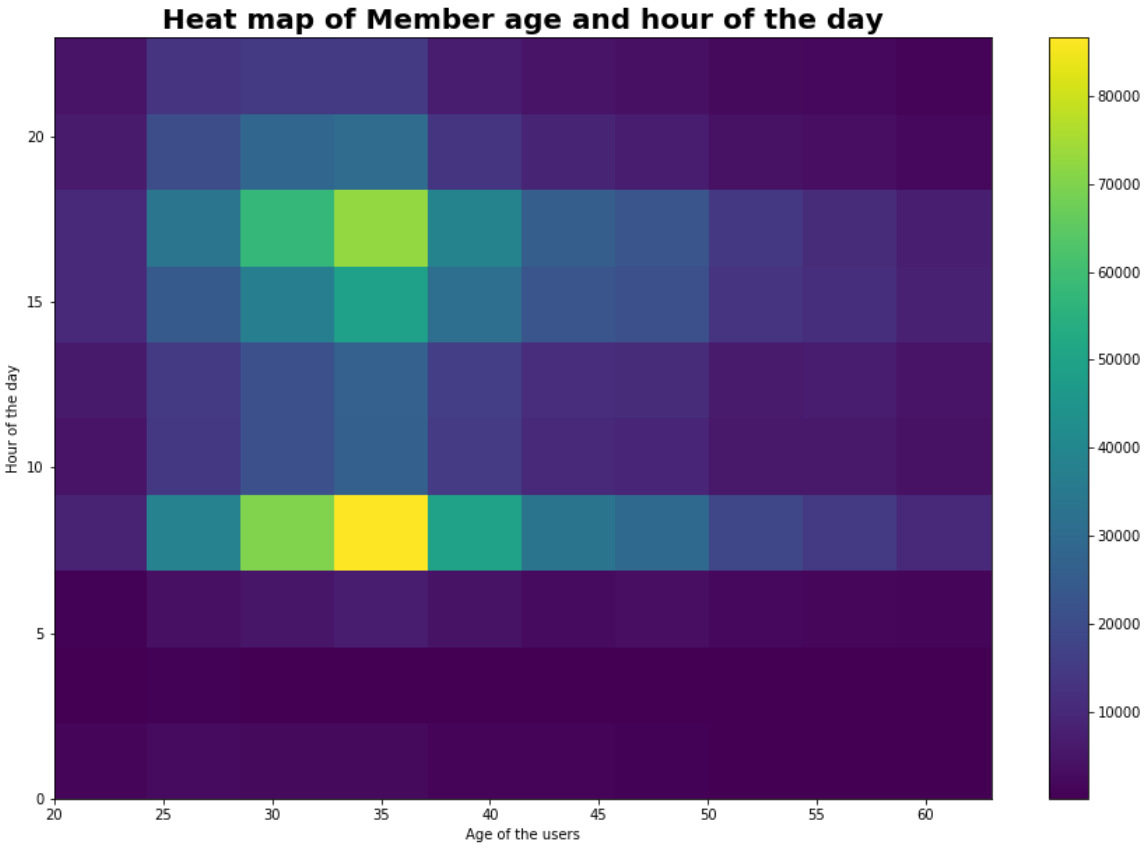
In [74]:

```
plt.figure(figsize = (15,10))
plt.hist2d(x = df_clean["member_age"] , y = df_clean["start_time_hour"])
plt.colorbar()

plt.xlabel("Age of the users")
plt.ylabel("Hour of the day")
plt.title("Heat map of Member age and hour of the day" , fontweight = "bold" , fontsize
= 20)
```

Out[74]:

```
Text(0.5, 1.0, 'Heat map of Member age and hour of the day')
```



Observation 13 : Users falling in the age group of 25 to 40 years have the largest number of trip between the peak hours of 8-9 am to 5-6pm

In [75]:

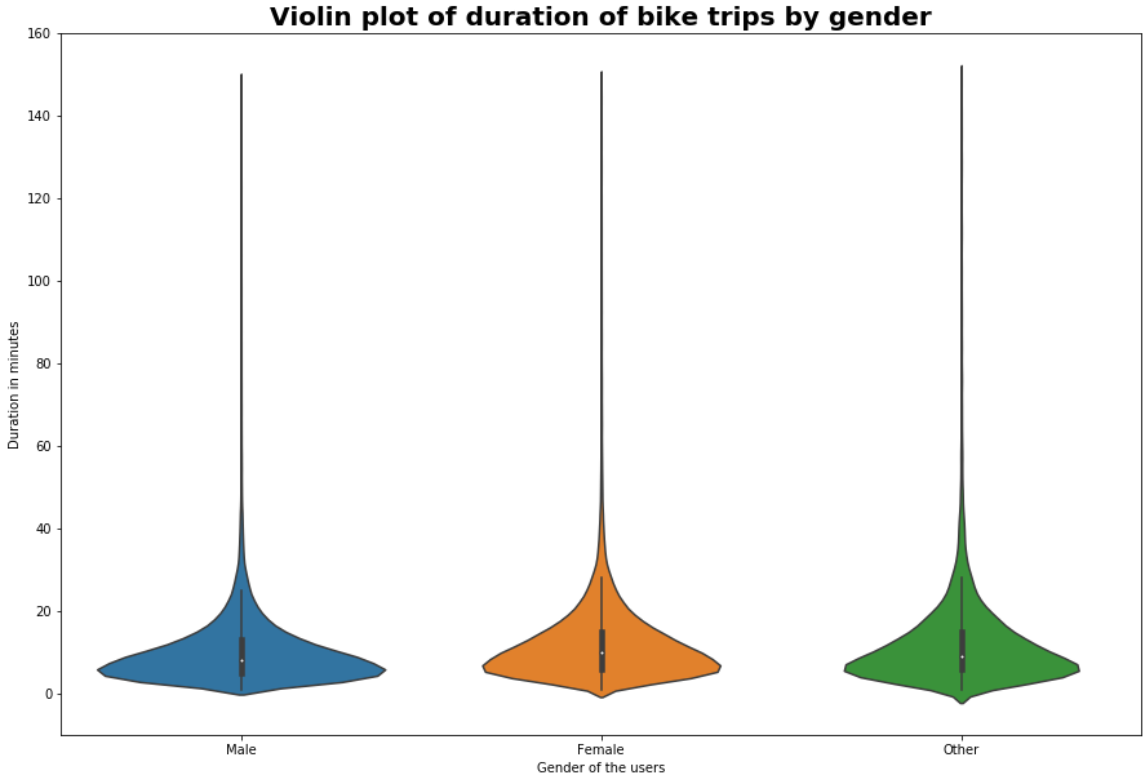
```
plt.figure(figsize = (15,10))
sns.violinplot(df_clean["member_gender"] , df_clean["duration_min"])

plt.xlabel("Gender of the users")
plt.ylabel("Duration in minutes")
plt.title("Violin plot of duration of bike trips by gender" , fontweight = "bold" , fon
tsize = 20)
```

```
C:\Users\anubh\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: Futu
reWarning: Using a non-tuple sequence for multidimensional indexing is dep
recated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this w
ill be interpreted as an array index, `arr[np.array(seq)]`, which will res
ult either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[75]:

```
Text(0.5, 1.0, 'Violin plot of duration of bike trips by gender')
```



In [76]:

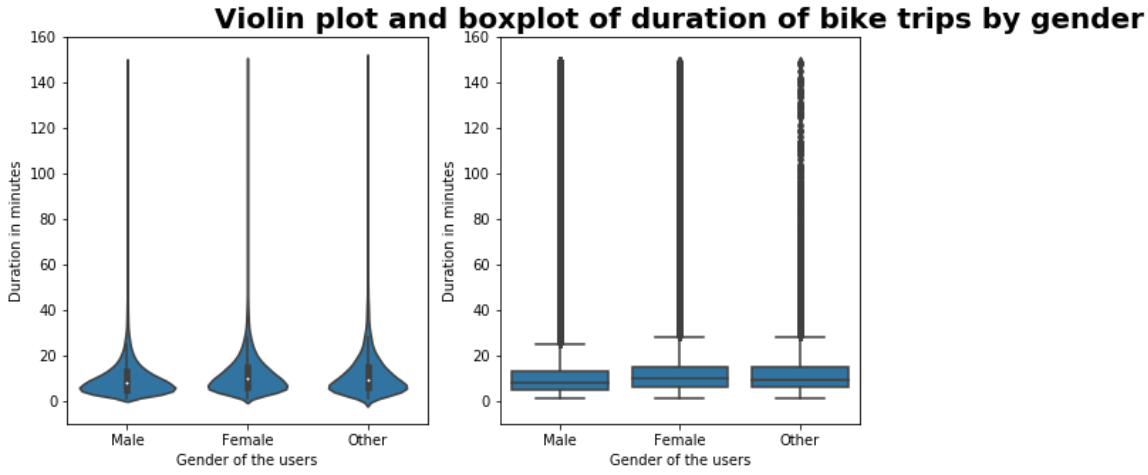
```
plt.figure(figsize = [10, 5])
base_color = sns.color_palette()[0]

# left plot: violin plot
plt.subplot(1, 2, 1)
ax1 = sns.violinplot(df_clean["member_gender"] , df_clean["duration_min"], color = base
_color)
plt.xlabel("Gender of the users")
plt.ylabel("Duration in minutes")
# right plot: box plot
plt.subplot(1, 2, 2)
sns.boxplot(df_clean["member_gender"] , df_clean["duration_min"], color = base_color)
plt.ylim(ax1.get_ylim()) # set y-axis limits to be same as left plot
plt.xlabel("Gender of the users")
plt.ylabel("Duration in minutes")

plt.title("Violin plot and boxplot of duration of bike trips by gender" , fontweight =
"bold" , fontsize = 20)
```

Out[76]:

```
Text(0.5, 1.0, 'Violin plot and boxplot of duration of bike trips by gende
r')
```



Observation 14 : Females are taking the longest trips(in minutes) among all the genders.Females trip duration falls between 15-17 minutes and males tend to take trips falling between 13-15 minutes. Most of the bike trips fall below 35 minutes of time, hence creating a scenario that the bike trips are usually used for shorter trips

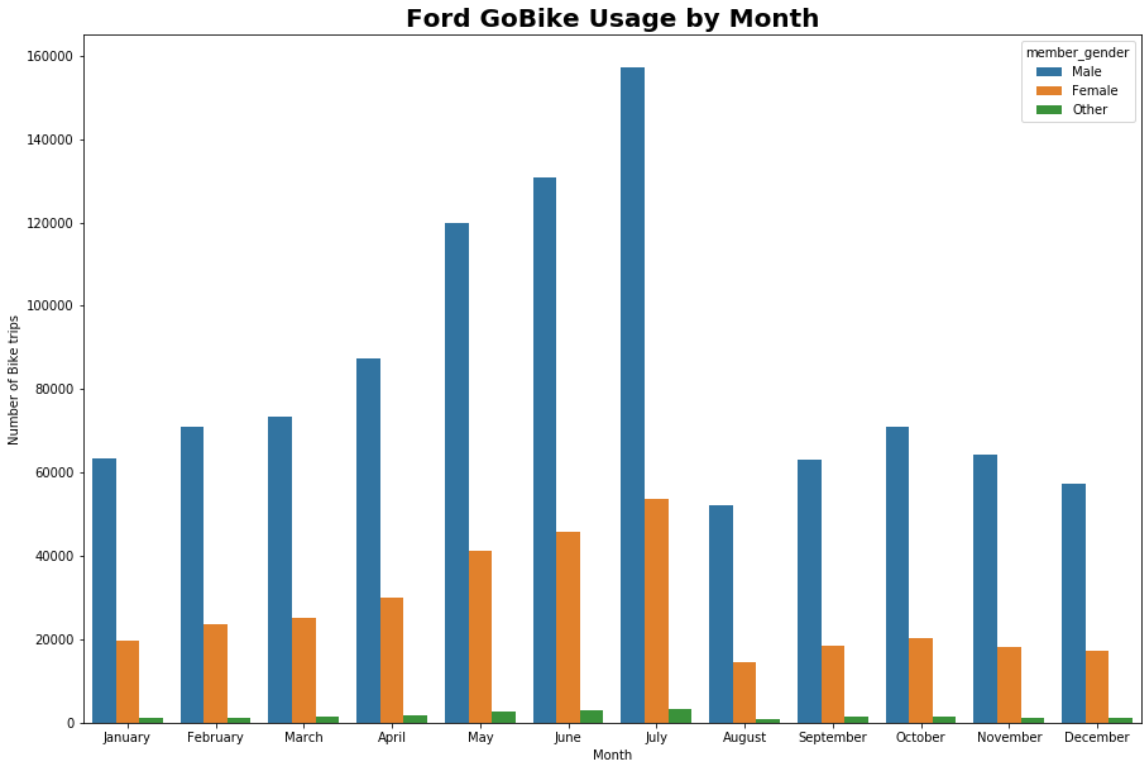
In [77]:

```
#usage by month
month = ["January" , "February" , "March" , "April" , "May" , "June" , "July" , "Augus
t" , "September" ,"October" ,"November" ,"December"]
plt.figure(figsize= (15,10))
sns.countplot(x = "start_time_month" , hue = "member_gender" , data = df_clean  ,order=
month)

plt.xlabel("Month")
plt.ylabel("Number of Bike trips")
plt.title('Ford GoBike Usage by Month' , fontweight = "bold" , fontsize = 20)
```

Out[77]:

```
Text(0.5, 1.0, 'Ford GoBike Usage by Month')
```



Observation 15: Male users are the largest users of bike trips and are using the bike trips mostly in the early summer months and the usage peaking in the month of July.Female users are also having the same trend like the Male users.While for other gender people , the rate of usage is almost same

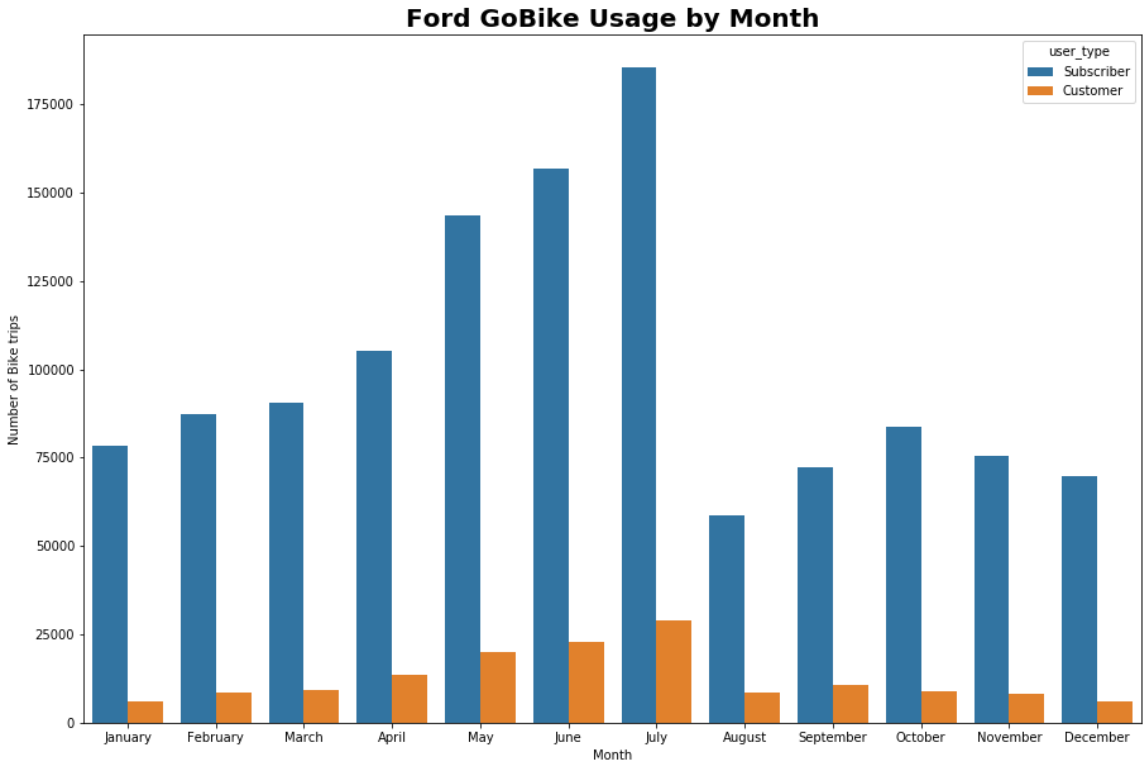
In [78]:

```
#usage by month
month = ["January" , "February" , "March" , "April" , "May" , "June" , "July" , "Augus
t" , "September" ,"October" ,"November" ,"December"]
plt.figure(figsize= (15,10))
sns.countplot(x = "start_time_month" , hue = "user_type" , data = df_clean  ,order=mont
h)

plt.xlabel("Month")
plt.ylabel("Number of Bike trips")
plt.title('Ford GoBike Usage by Month' , fontweight = "bold" , fontsize = 20)
```

Out[78]:

```
Text(0.5, 1.0, 'Ford GoBike Usage by Month')
```



Observation 16 : Subscribers are the largest users of the bike service and the usage of the bike service peaks in the month of July .Customers usage also rise in the month of July and follows the same trend like the customer but with very less usage as compared to Subscribers

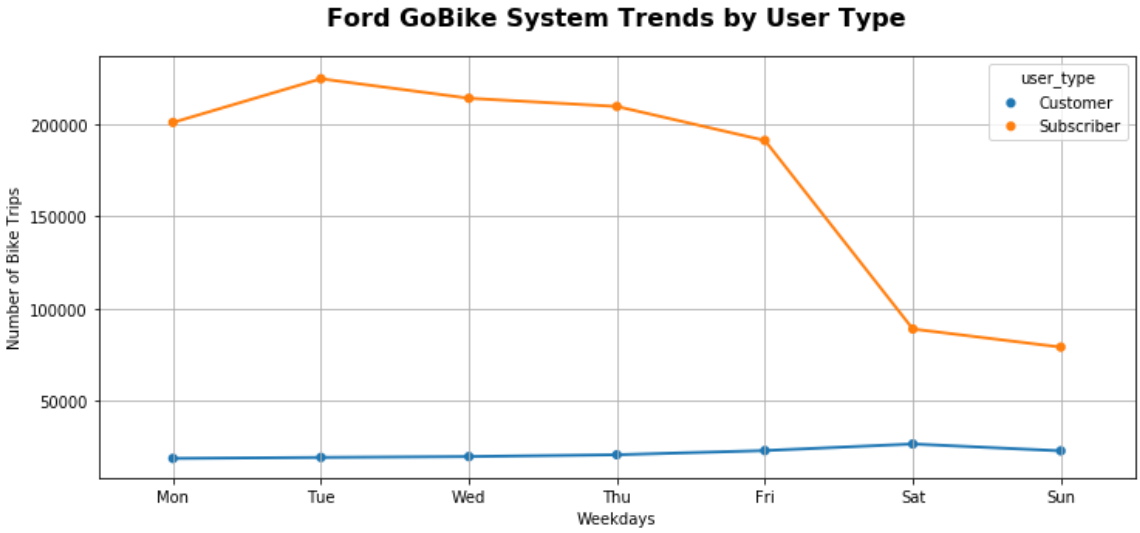
In [79]:

```
# Customer Usage by Weekday vs. Subscriber Usage by Weekday
plt.figure(figsize=(12, 5))

df_cleaned_user_week = df_clean.groupby(['start_time_weekday', 'user_type']).size().res
et_index()
weekday = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']

sns.pointplot(data=df_cleaned_user_week, x='start_time_weekday', y=0, hue = 'user_type'
, scale=.7, order = weekday);

plt.title('Ford GoBike System Trends by User Type', y=1.05, fontsize=16, fontweight='bo
ld')
plt.xlabel('Weekdays')
plt.ylabel('Number of Bike Trips');
plt.grid()
```



Observation 17 : Customers have very less usage of bike service as compared to Subscribers.There is a slight increase in the usage by Customers on Saturday and again dipping post sunday ,which may be because the customers are tourists and visit SF on weekends.Subscribers have the highest usage of service during the weekdays and the usage dips in weekends.

In [80]:

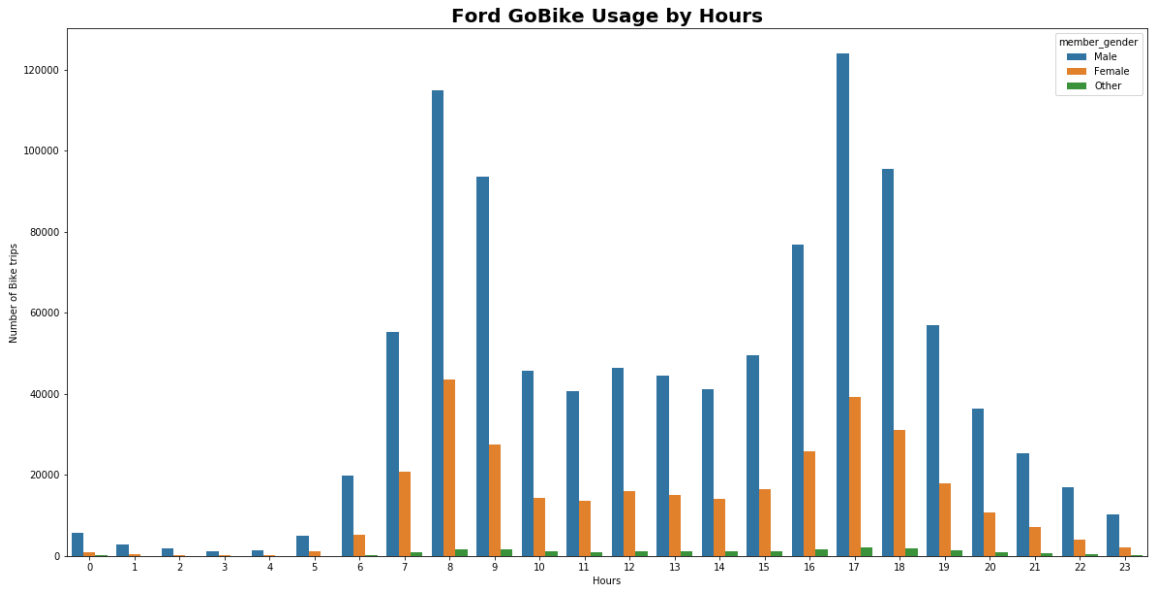
```
# Customer Usage by Duration vs. Subscriber Usage by Hour

plt.figure(figsize= (20,10))
sns.countplot(x = "start_time_hour" , hue = "member_gender" , data = df_clean)

plt.xlabel("Hours")
plt.ylabel("Number of Bike trips")
plt.title('Ford GoBike Usage by Hours' , fontweight = "bold" , fontsize = 20)
```

Out[80]:

```
Text(0.5, 1.0, 'Ford GoBike Usage by Hours')
```



Observation 18 : Male users have the highest usage of the bike trips .Both Male and female use the bike service mostly during the morning and evening hours of the day

In [81]:

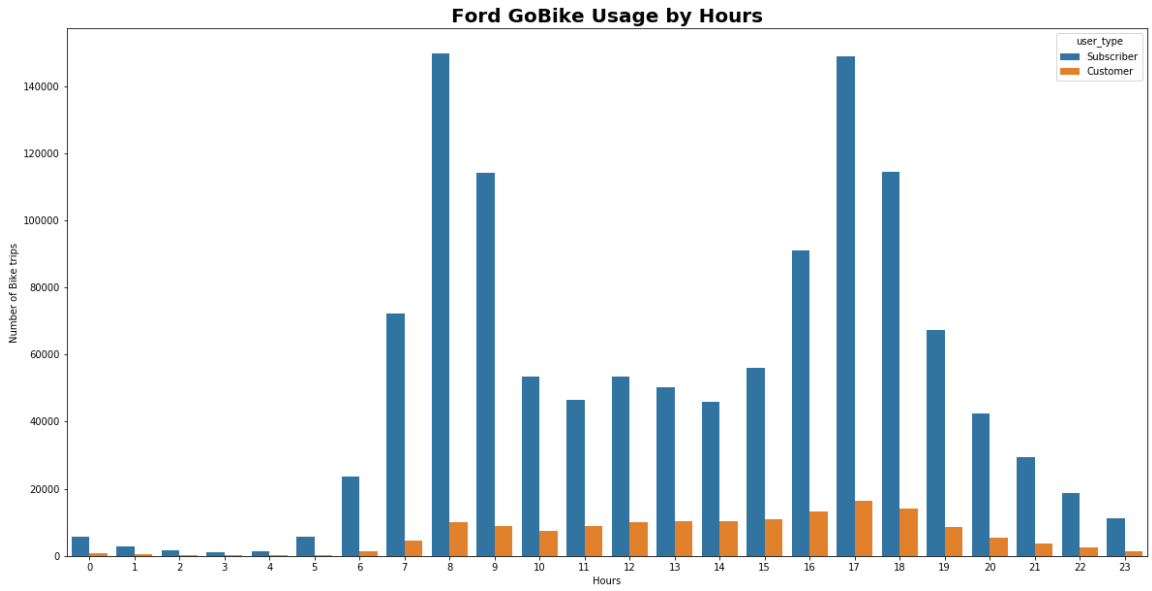
```
#usage by hours

plt.figure(figsize= (20,10))
sns.countplot(x = df_clean["start_time_hour"]  , hue = "user_type", data = df_clean)

plt.xlabel("Hours")
plt.ylabel("Number of Bike trips")
plt.title('Ford GoBike Usage by Hours' , fontweight = "bold" , fontsize = 20)
```

Out[81]:

```
Text(0.5, 1.0, 'Ford GoBike Usage by Hours')
```



Observation 19 : Subscribers are the majority users of the bike rides.Subscribers use the bike ride service mostly during the morning hours and evening hours , while the usage of bike rides by the Customers is high throughout the day ,which maybe the case that customers are tourists or non-working class people and use the bike service throughout the day and Subscribers are users who use the bike service to commute to their work/school

Talk about some of the relationships you observed in this part of the investigation. How did the feature(s) of interest vary with other features in the dataset?

I tried to explore the relationship between the data with respect to user types and user gender.The customers are casual users,mostly tourists who use the bike service to explore the city during weekends or holidays.Subscribers are daily commuters who use the bike service mostly on weekend to travel to office/school.The bike service is used mostly between 8-9am and 5-6pm.

Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

Yes ,there is a difference in the trip duration between customers and subscribers. Customers trips are usually longer than for subscribers, most probably becuase the Customers use the service on weekends to roam around the city and maybe the customers are tourists that visit the city and use the bike service to explore the city.While the subscribers use the bike service for shorter duration,mostly for office/school commute or commute between stations

Multivariate Exploration

I created plots of three or more variables to investigate the Ford GoBike for more interesting insights

In [82]:

```
age_order = ["15 - 25" , "26 - 35" , "36 - 45" , "46 - 55" , "56 - 66"]
```

In [83]:

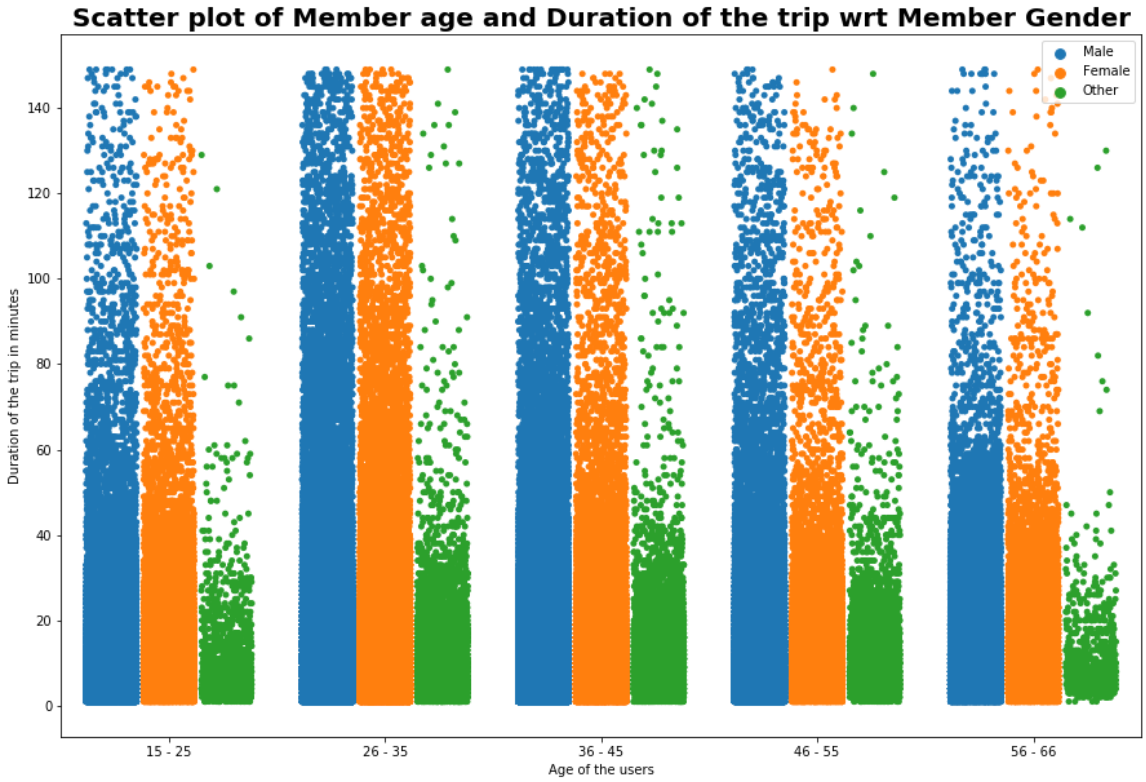
```
plt.figure(figsize = (15 , 10))

sns.stripplot(x = "age_bins" , y = "duration_min" , hue = "member_gender" ,
              data = df_clean , jitter = 0.35 , dodge = True ,order=age_order)

plt.xlabel("Age of the users")
plt.ylabel("Duration of the trip in minutes")
plt.legend()
plt.title("Scatter plot of Member age and Duration of the trip wrt Member Gender" , fon
tweight = "bold" , fontsize = 20)
```

Out[83]:

```
Text(0.5, 1.0, 'Scatter plot of Member age and Duration of the trip wrt Me
mber Gender')
```



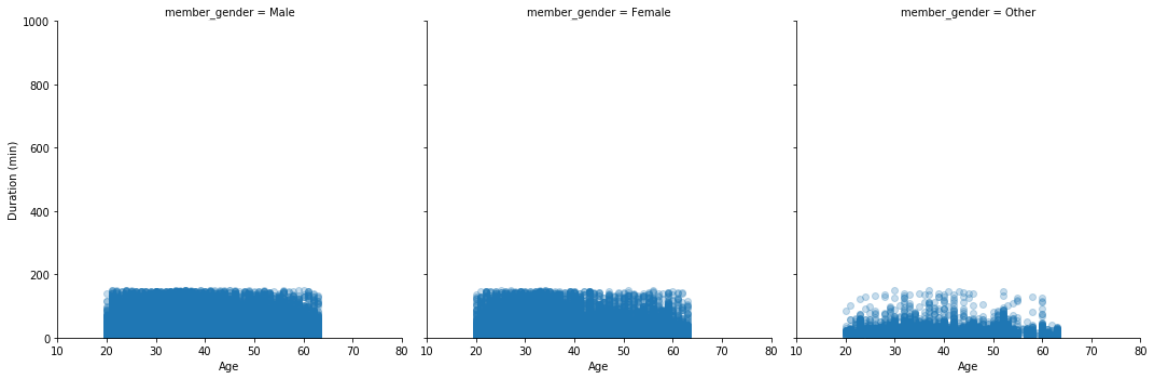
Observation 20 : Most of the users are Male ,followed by Females and Other users are the least.Users in the age gap 15-25 years ,tend to take bike trips of duration below 100 minutes .Users between age gap 26-35 years ,tend to take longer duration of trips below 140 minutes. Users in age group 36-45 mostly take trip duration below 120 minutes.Users in age group 46-55 take rides mostly below 100 minutes.People between 55-66 years take the most shortest rides, followed by people between 15-25 years.

In [84]:

```
plt.figure(figsize = (25,10))
genders = sns.FacetGrid(data = df_clean, col = 'member_gender', col_wrap = 3, height =
5 ,
                 xlim = [10, 80], ylim = [0, 1000])
genders.map(plt.scatter, 'member_age', 'duration_min', alpha=0.25)
genders.set_xlabels('Age')
genders.set_ylabels('Duration (min)')

plt.show()
```

```
<Figure size 1800x720 with 0 Axes>
```



Observation 21 : Male users range mostly between 20-60 years of age and the all the trip durations by male users are below 160 minutes.Male users till 45 years tend to use the service for longer duration compared to Male users above 45 years.Female users also lay between 20-60 years of age and Female users belwo 40 years have longer trip duration compared to users above 40 years.Other users tend to have very short duration trip mostly around 20-30 minutes ,with very few trip above 60 minutes

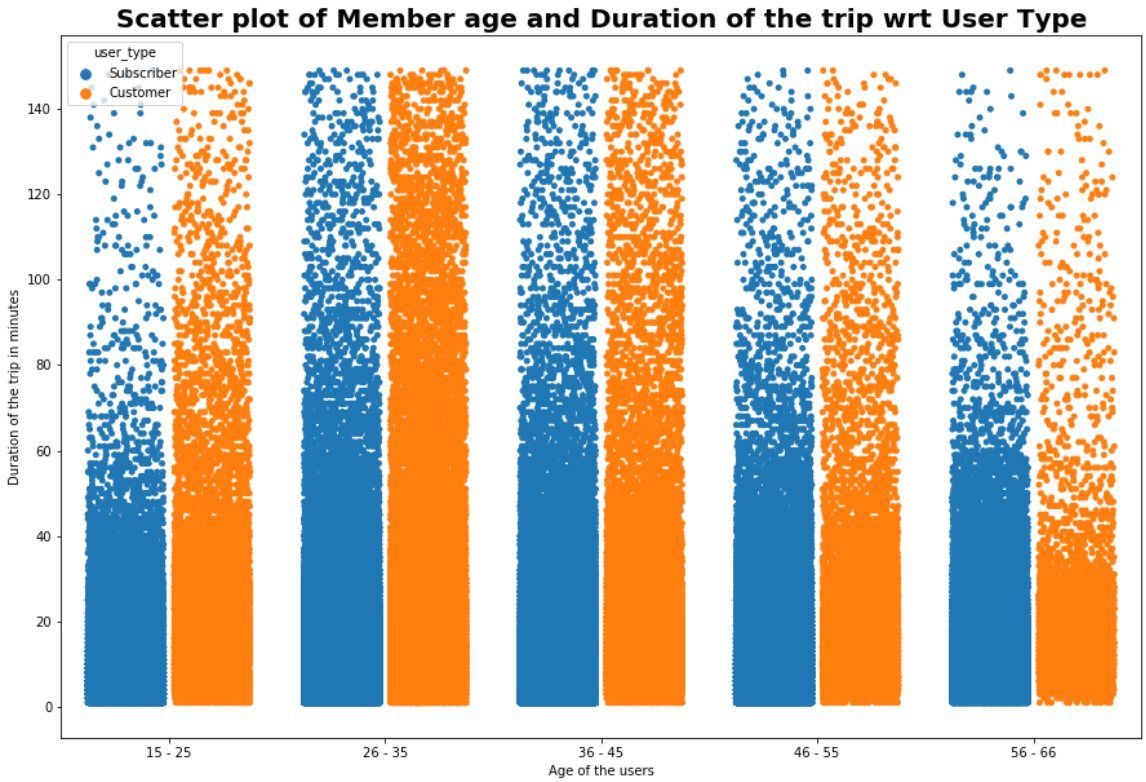
In [85]:

```
plt.figure(figsize = (15 , 10))

sns.stripplot(x = "age_bins" , y = "duration_min" , data = df_clean , hue = "user_type"
,
                  jitter = 0.35 , dodge = True , order = age_order)
plt.xlabel("Age of the users")
plt.ylabel("Duration of the trip in minutes")
plt.title("Scatter plot of Member age and Duration of the trip wrt User Type" , fontwei
ght = "bold" , fontsize = 20)
```

Out[85]:

```
Text(0.5, 1.0, 'Scatter plot of Member age and Duration of the trip wrt Us
er Type')
```



Observation 22 : Users below 25 years of age are mostly Customers and tend to take longer duration of rides compared to the Subscribers.Subscribers falling between 26-35 years are taking shorter duration rides compared to Customers who are taking longer duration rides .While users falling between age gap 36-45 tend to take shorter duration rides than users between age 25-36 years .Post the age gap of 46 years , users tend to take shorter duration of trips while the number of subscriber are more than customers.Overall customers tend to take longer duration of trips compared to subscribers

In [86]:

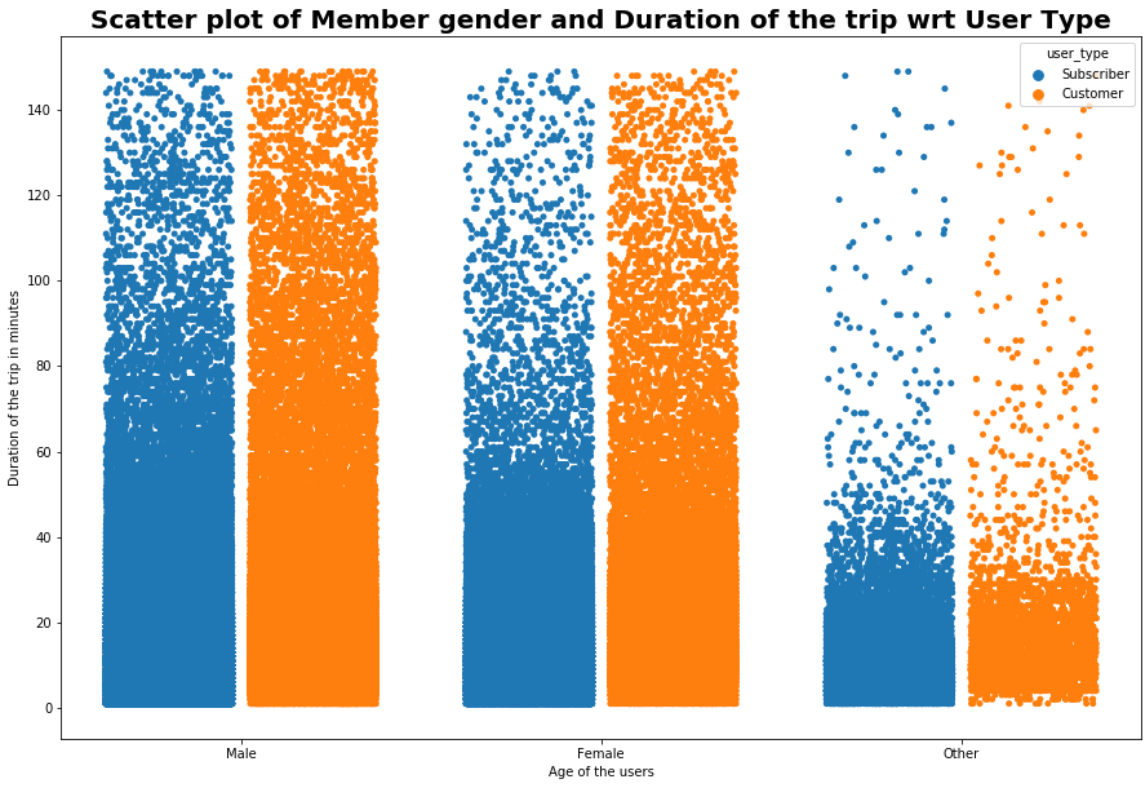
```
plt.figure(figsize = (15 , 10))

sns.stripplot(x = "member_gender" , y = "duration_min" , data = df_clean ,hue = "user_type",
                jitter = 0.35 , dodge = True )

plt.xlabel("Age of the users")
plt.ylabel("Duration of the trip in minutes")
plt.title("Scatter plot of Member gender and Duration of the trip wrt User Type" , fontweight = "bold" , fontsize = 20)
```

Out[86]:

```
Text(0.5, 1.0, 'Scatter plot of Member gender and Duration of the trip wrt User Type')
```



Observation 23 : Male subscribers tend to take only short duration rides while Male customers tend to take longer duration rides.Female subscribers also tend to take more shorter duration rides and Female customers tend to longer duration rides. Other gender subscribers tend to take longer rides compared to customers of other gender.The reason why most of the subscribers take shorter duration rides can be that most of them use the bike service to ferry around shorter distance between stations or schools.While Customers who are more of a casual user who use the bike service to explore the city or shop around the city,hence taking longer rides

Talk about some of the relationships you observed in this part of the investigation. Were there features that strengthened each other in terms of looking at your feature(s) of interest?

This exploration helped me to find out that most of the users fall in the age group of 25-36 and most of the users of the bike service are males and their number has kept of increasing at a steady rate over the years, while female users tend to use the service for longer duration of time.

Were there any interesting or surprising interactions between features?

The usage of bike service by the Other gender is almost flat over the years.The male and female users follow almost the same trend over the years

- https://stackoverflow.com/questions/20906474/import-multiple-csv-files-into-pandas-and-concatenate-into-one-dataframe (https://stackoverflow.com/questions/20906474/import-multiple-csv-files-into-pandas-and-concatenate-into-one-dataframe)
- https://stackoverflow.com/questions/13148429/how-to-change-the-order-of-dataframe-columns (https://stackoverflow.com/questions/13148429/how-to-change-the-order-of-dataframe-columns)