Part_I_exploration_template

April 22, 2022

1 Part I - (Dataset Exploration Title)

1.1 by (your name here)

1.2 Introduction

Introduce the dataset

Rubric Tip: Your code should not generate any errors, and should use functions, loops where possible to reduce repetitive code. Prefer to use functions to reuse code statements.

Rubric Tip: Document your approach and findings in markdown cells. Use comments and docstrings in code cells to document the code functionality.

Rubric Tip: Markup cells should have headers and text that organize your thoughts, findings, and what you plan on investigating next.

1.3 Preliminary Wrangling

```
In [2]: # import all packages and set plots to be embedded inline
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sb
        import calendar


        %matplotlib inline
```

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

```
In [3]: read_file = pd.read_csv (r'201902-fordgobike-tripdata.txt')
        read_file.to_csv (r'201902-fordgobike-tripdata.csv', index=None)
```

```
In [4]: df_bike_data = pd.read_csv('201902-fordgobike-tripdata.csv')
        df_bike_data.shape
```

Out[4]: (183412, 16)

In [5]: df_bike_data.tail(10)

```
Out[5]:         duration_sec               start_time                 end_time  \
        183402           122  2019-02-01 00:17:32.2580  2019-02-01 00:19:34.9380
        183403           249  2019-02-01 00:15:12.0670  2019-02-01 00:19:21.6990
        183404           256  2019-02-01 00:12:50.5540  2019-02-01 00:17:07.3620
        183405           111  2019-02-01 00:14:49.8740  2019-02-01 00:16:41.3010
        183406           706  2019-02-01 00:04:40.6160  2019-02-01 00:16:27.0800
        183407           480  2019-02-01 00:04:49.7240  2019-02-01 00:12:50.0340
        183408           313  2019-02-01 00:05:34.7440  2019-02-01 00:10:48.5020
        183409           141  2019-02-01 00:06:05.5490  2019-02-01 00:08:27.2200
        183410           139  2019-02-01 00:05:34.3600  2019-02-01 00:07:54.2870
        183411           271  2019-02-01 00:00:20.6360  2019-02-01 00:04:52.0580

                start_station_id                                start_station_name  \
        183402             119.0                                 18th St at Noe St
        183403             256.0                            Hearst Ave at Euclid Ave
        183404             241.0                                 Ashby BART Station
        183405             324.0                   Union Square (Powell St at Post St)
        183406             138.0                             Jersey St at Church St
        183407              27.0                             Beale St at Harrison St
        183408              21.0   Montgomery St BART Station (Market St at 2nd St)
        183409             278.0                            The Alameda at Bush St
        183410             220.0                           San Pablo Ave at MLK Jr Way
        183411              24.0                              Spear St at Folsom St

                start_station_latitude  start_station_longitude  end_station_id  \
        183402               37.761047              -122.432642           120.0
        183403               37.875112              -122.260553           247.0
        183404               37.852477              -122.270213           248.0
        183405               37.788300              -122.408531            19.0
        183406               37.750900              -122.427411            78.0
        183407               37.788059              -122.391865           324.0
        183408               37.789625              -122.400811            66.0
        183409               37.331932              -121.904888           277.0
        183410               37.811351              -122.273422           216.0
        183411               37.789677              -122.390428            37.0

                                   end_station_name  end_station_latitude  \
        183402                  Mission Dolores Park             37.761420
        183403                Fulton St at Bancroft Way             37.867789
        183404              Telegraph Ave at Ashby Ave             37.855956
        183405                    Post St at Kearny St             37.788975
        183406                     Folsom St at 9th St             37.773717
        183407   Union Square (Powell St at Post St)             37.788300
        183408                  3rd St at Townsend St             37.778742
```

```
183409           Morrison Ave at Julian St            37.333658
183410            San Pablo Ave at 27th St            37.817827
183411                 2nd St at Folsom St            37.785000

        end_station_longitude  bike_id   user_type  member_birth_year  \
183402            -122.426435     4326  Subscriber                NaN
183403            -122.265896     4642  Subscriber             2000.0
183404            -122.259795     4845  Subscriber             1980.0
183405            -122.403452     4832  Subscriber             1984.0
183406            -122.411647     5017  Subscriber             1988.0
183407            -122.408531     4832  Subscriber             1996.0
183408            -122.392741     4960  Subscriber             1984.0
183409            -121.908586     3824  Subscriber             1990.0
183410            -122.275698     5095  Subscriber             1988.0
183411            -122.395936     1057  Subscriber             1989.0

       member_gender bike_share_for_all_trip
183402           NaN                      No
183403          Male                      No
183404          Male                     Yes
183405          Male                      No
183406          Male                      No
183407          Male                      No
183408          Male                      No
183409          Male                     Yes
183410          Male                      No
183411          Male                      No
```

In [6]:
```
df_bike_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 16 columns):
duration_sec               183412 non-null int64
start_time                 183412 non-null object
end_time                   183412 non-null object
start_station_id           183215 non-null float64
start_station_name         183215 non-null object
start_station_latitude     183412 non-null float64
start_station_longitude    183412 non-null float64
end_station_id             183215 non-null float64
end_station_name           183215 non-null object
end_station_latitude       183412 non-null float64
end_station_longitude      183412 non-null float64
bike_id                    183412 non-null int64
user_type                  183412 non-null object
member_birth_year          175147 non-null float64
member_gender              175147 non-null object
```

```
bike_share_for_all_trip    183412 non-null object
dtypes: float64(7), int64(2), object(7)
memory usage: 22.4+ MB
```

```
In [7]: df_bike_data.isnull().sum()
```

```
Out[7]: duration_sec                  0
        start_time                    0
        end_time                      0
        start_station_id            197
        start_station_name          197
        start_station_latitude        0
        start_station_longitude       0
        end_station_id              197
        end_station_name            197
        end_station_latitude          0
        end_station_longitude         0
        bike_id                       0
        user_type                     0
        member_birth_year          8265
        member_gender              8265
        bike_share_for_all_trip       0
        dtype: int64
```

```
In [8]: df_bike_data.duplicated().sum()
```

```
Out[8]: 0
```

```
In [9]: #Find unique start stations:
        df_bike_data.start_station_name.nunique()
```

```
Out[9]: 329
```

```
In [10]: #Find unique end stations:
         df_bike_data.end_station_name.nunique()
```

```
Out[10]: 329
```

```
In [11]: #Find unique bike_id:
         df_bike_data.bike_id.nunique()
```

```
Out[11]: 4646
```

```
In [12]: #Find unique user_type:
         df_bike_data.user_type.nunique()
         df_bike_data.user_type.unique()
```

```
Out[12]: array(['Customer', 'Subscriber'], dtype=object)
```

In [13]:
```
df_bike_data.describe()
```

Out[13]:
```
         duration_sec  start_station_id  start_station_latitude  \
count   183412.000000     183215.000000           183412.000000   
mean       726.078435        138.590427               37.771223   
std       1794.389780        111.778864                0.099581   
min         61.000000          3.000000               37.317298   
25%        325.000000         47.000000               37.770083   
50%        514.000000        104.000000               37.780760   
75%        796.000000        239.000000               37.797280   
max      85444.000000        398.000000               37.880222   

       start_station_longitude  end_station_id  end_station_latitude  \
count            183412.000000   183215.000000         183412.000000   
mean               -122.352664      136.249123             37.771427   
std                   0.117097      111.515131              0.099490   
min                -122.453704        3.000000             37.317298   
25%                -122.412408       44.000000             37.770407   
50%                -122.398285      100.000000             37.781010   
75%                -122.286533      235.000000             37.797320   
max                -121.874119      398.000000             37.880222   

       end_station_longitude        bike_id  member_birth_year  
count          183412.000000  183412.000000      175147.000000  
mean             -122.352250    4472.906375        1984.806437  
std                 0.116673    1664.383394          10.116689  
min              -122.453704      11.000000        1878.000000  
25%              -122.411726    3777.000000        1980.000000  
50%              -122.398279    4958.000000        1987.000000  
75%              -122.288045    5502.000000        1992.000000  
max              -121.874119    6645.000000        2001.000000  
```

In [14]:
```
df_bike_data.duration_sec.sort_values(ascending=False)
```

Out[14]:
```
101361    85444
85465     84548
153705    83772
127999    83519
112435    83407
5203      83195
95750     82512
173365    82385
8631      81549
176987    80891
107581    79548
94581     75262
90195     74408
86454     74097
```

```
123383    73930
73587     72824
129176    72627
116671    72590
129177    72576
136599    72361
145977    71470
128648    71326
29922     70925
14381     70211
95747     70072
31295     70050
32098     69980
54376     69803
33431     69620
120711    69335
          ...
105162       62
84454        62
83770        62
129402       62
9221         62
86969        62
33320        62
97032        62
91046        62
161611       62
171049       62
73372        62
85488        61
27017        61
154115       61
179646       61
19581        61
121470       61
51120        61
18578        61
58992        61
80047        61
64088        61
82564        61
134955       61
44301        61
44787        61
103565       61
157305       61
169882       61
Name: duration_sec, Length: 183412, dtype: int64
```

```
In [15]: print(df_bike_data.member_gender.nunique())
         print(df_bike_data.member_gender.unique())
```

```
3
['Male' nan 'Other' 'Female']
```

```
In [16]: print(df_bike_data.bike_share_for_all_trip.nunique())

         print(df_bike_data.bike_share_for_all_trip.unique())
```

```
2
['No' 'Yes']
```

2 Observations :

1.Convert start_time and end_time to datetime.
2.Convert start_station_id and end_station_id from float to string.
3.Convert bike_id to from integer to string.
4.Convert user_type and member_gender to category.
5.Convert the member_birth_year to member_age so we can analyze the age statistics of the bike users.
5.I do see missing values in start_station_name,end_station_name and member_birth_year,member_gender, we do not need this for
our analysis so I am going to let it pass.

3 Cleaning Data:

- First copy the data.
- Then work on the Data type issues:(Tidiness)
- start_time and end_time to datetime
- start_station_id, end_staion_id has to be changed to string.
- bike_id should be changed to str.
- user_type,member_gender,bike_share_for_all_trip change to category

```
In [17]: #Copy the Data :
         df=df_bike_data.copy()
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 16 columns):
duration_sec               183412 non-null int64
start_time                 183412 non-null object
end_time                   183412 non-null object
start_station_id           183215 non-null float64
start_station_name         183215 non-null object
```

```
start_station_latitude     183412 non-null float64
start_station_longitude    183412 non-null float64
end_station_id             183215 non-null float64
end_station_name           183215 non-null object
end_station_latitude       183412 non-null float64
end_station_longitude      183412 non-null float64
bike_id                    183412 non-null int64
user_type                  183412 non-null object
member_birth_year          175147 non-null float64
member_gender              175147 non-null object
bike_share_for_all_trip    183412 non-null object
dtypes: float64(7), int64(2), object(7)
memory usage: 22.4+ MB
```

In [18]:
```
## We are using pd.to_datetime to convert datetime :
df.start_time = pd.to_datetime(df.start_time)
df.end_time = pd.to_datetime(df.end_time)
```

In [19]:
```
# Convert datatype to str using astype(str)
df.start_station_id = df.bike_id.astype(str)
df.end_station_id = df.bike_id.astype(str)
df.bike_id = df.bike_id.astype(str)
```

In [20]:
```
# Converting categorical data like user_type,member_gender and bike_share_for_all_trip
df['user_type'] = df['user_type'].astype('category')
df['member_gender'] = df['member_gender'].astype('category')
df['bike_share_for_all_trip'] = df['bike_share_for_all_trip'].astype('category')
```

In [21]:
```
# lets check the data now:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 16 columns):
duration_sec               183412 non-null int64
start_time                 183412 non-null datetime64[ns]
end_time                   183412 non-null datetime64[ns]
start_station_id           183412 non-null object
start_station_name         183215 non-null object
start_station_latitude     183412 non-null float64
start_station_longitude    183412 non-null float64
end_station_id             183412 non-null object
end_station_name           183215 non-null object
end_station_latitude       183412 non-null float64
end_station_longitude      183412 non-null float64
bike_id                    183412 non-null object
user_type                  183412 non-null category
member_birth_year          175147 non-null float64
```

```
member_gender              175147 non-null category
bike_share_for_all_trip    183412 non-null category
dtypes: category(3), datetime64[ns](2), float64(5), int64(1), object(5)
memory usage: 18.7+ MB
```

In [22]:
```
df['start_time'].value_counts()
```

Out[22]:
```
2019-02-01 13:40:09.492    2
2019-02-11 17:05:07.840    2
2019-02-19 17:52:44.175    2
2019-02-01 18:24:34.874    2
2019-02-22 20:11:42.256    2
2019-02-25 08:52:07.582    2
2019-02-07 09:06:07.056    2
2019-02-15 07:47:00.197    2
2019-02-07 17:56:08.897    2
2019-02-06 21:35:57.574    2
2019-02-15 08:43:18.422    2
2019-02-14 16:48:08.897    1
2019-02-12 00:37:22.208    1
2019-02-07 19:05:00.464    1
2019-02-07 17:49:10.708    1
2019-02-05 08:04:22.768    1
2019-02-22 18:44:48.135    1
2019-02-23 14:41:59.618    1
2019-02-24 17:34:23.058    1
2019-02-16 07:45:55.848    1
2019-02-27 17:38:49.466    1
2019-02-14 08:22:41.664    1
2019-02-15 17:29:47.892    1
2019-02-04 19:04:57.803    1
2019-02-13 13:05:53.508    1
2019-02-10 16:18:50.219    1
2019-02-08 13:13:04.955    1
2019-02-28 08:17:47.738    1
2019-02-15 08:15:46.769    1
2019-02-11 14:32:39.707    1
                          ..
2019-02-19 18:51:36.112    1
2019-02-24 13:50:59.528    1
2019-02-26 16:42:00.925    1
2019-02-08 11:59:53.802    1
2019-02-28 18:37:53.980    1
2019-02-15 08:06:04.668    1
2019-02-12 16:43:27.189    1
2019-02-22 21:07:22.133    1
2019-02-28 20:58:51.575    1
```

```
2019-02-22 16:51:18.573    1
2019-02-26 18:00:14.261    1
2019-02-23 15:48:40.739    1
2019-02-27 17:05:03.492    1
2019-02-06 18:58:06.780    1
2019-02-06 17:15:16.704    1
2019-02-13 17:43:25.648    1
2019-02-01 17:15:10.299    1
2019-02-06 11:24:39.263    1
2019-02-01 13:58:40.112    1
2019-02-08 23:48:34.668    1
2019-02-23 19:34:42.560    1
2019-02-10 14:34:28.356    1
2019-02-12 18:11:13.018    1
2019-02-07 17:55:09.080    1
2019-02-18 22:36:48.726    1
2019-02-22 08:46:15.436    1
2019-02-19 17:29:08.242    1
2019-02-11 10:24:37.610    1
2019-02-20 08:02:52.933    1
2019-02-15 07:01:31.319    1
Name: start_time, Length: 183401, dtype: int64
```

3.0.1 What is the structure of your dataset?

The dataset has 183401 records with 16 columns for the month of Febuary 2019.

The original data has the following details trip duration in seconds, start_time and end_time of the trip, start_station and end_station,user_type, member_gender, member_birth_year and so on.

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

I am most intrested to check when the bikes are most in demand in a particular day,time and also how age and gender factor affects the demands for bike rides.

- Most frequent times of travel.
- The hour of the day that has highest number of bike rides.
- Most popular start and end stations.
- Statistics of Bike users based on age and gender.
- Mean travel time.

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

Most features in the dataset are useful particularly start_time, end_time,start_station,end_station,gender,type of users,age and durations. These features will help me get more information on bike rides and their users.

3.1 Univariate Exploration

In this section, investigate distributions of individual variables. If you see unusual points or outliers, take a deeper look to clean things up and prepare yourself to look at relationships between variables.

Rubric Tip: The project (Parts I alone) should have at least 15 visualizations distributed over univariate, bivariate, and multivariate plots to explore many relationships in the data set. Use reasoning to justify the flow of the exploration.

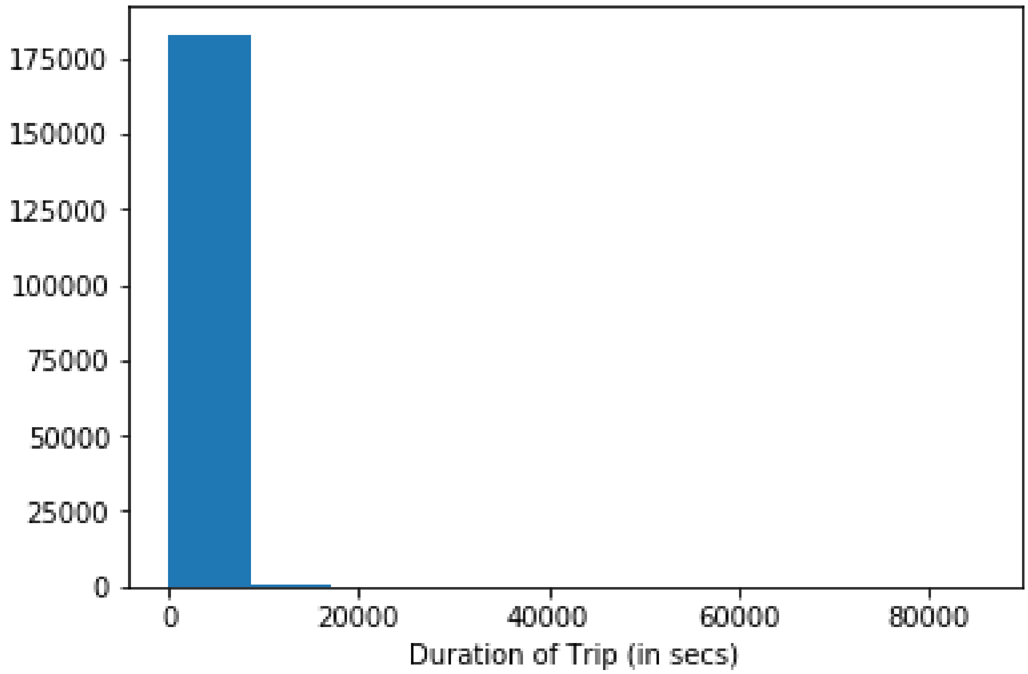
Rubric Tip: Use the "Question-Visualization-Observations" framework throughout the exploration. This framework involves **asking a question from the data, creating a visualization to find answers, and then recording observations after each visualisation.**

In [23]:
```
#Lets look into our first feature duration:
df['duration_sec'].describe()
```

Out[23]:
```
count    183412.000000
mean        726.078435
std        1794.389780
min          61.000000
25%         325.000000
50%         514.000000
75%         796.000000
max       85444.000000
Name: duration_sec, dtype: float64
```

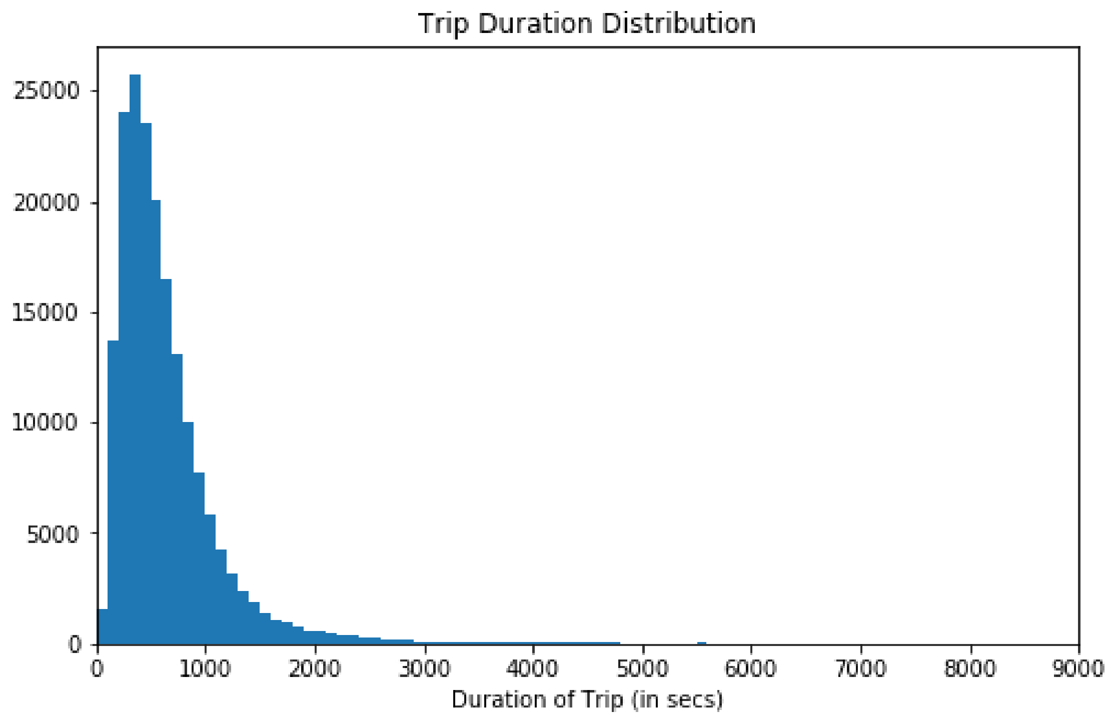
In [24]:
```
#lets plot a histogram to check it skewness:
plt.hist(data = df, x= 'duration_sec');
plt.xlabel('Duration of Trip (in secs)')
```

Out[24]:
```
Text(0.5,0,'Duration of Trip (in secs)')
```



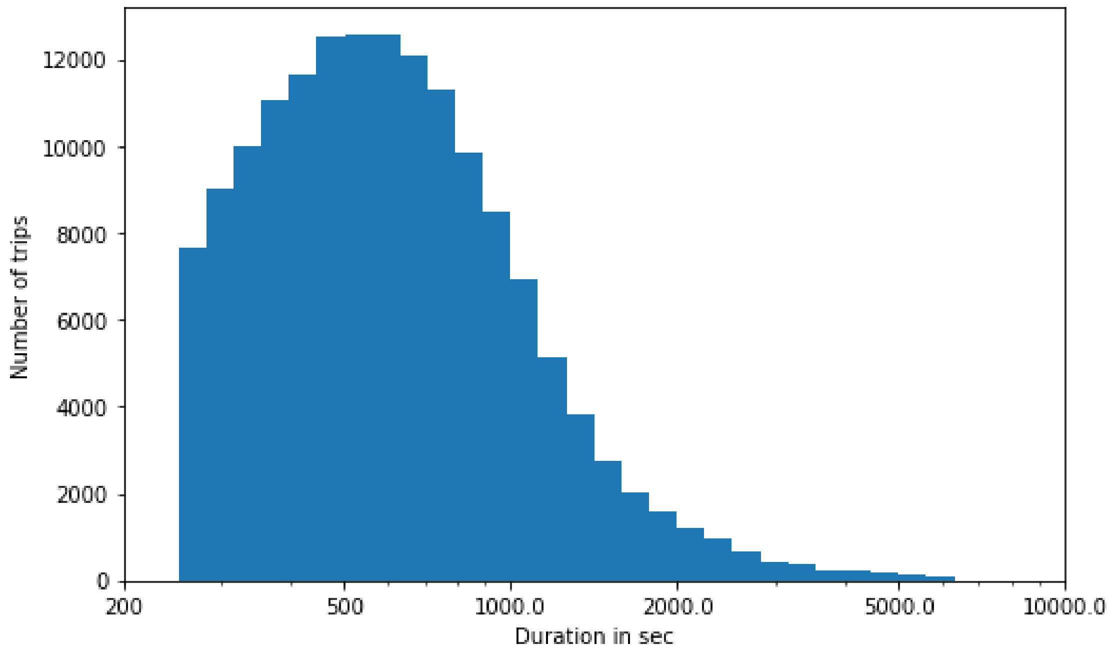
Here we dont see a very clear graph, from which we can analyze.

```
In [25]: #Lets try with a standard plot:
         binsize = 100
         bins = np.arange(0, df['duration_sec'].max()+binsize, binsize)

         plt.figure(figsize=[8, 5])
         plt.hist(data = df, x = 'duration_sec', bins = bins)
         plt.xlim(0, 9000)
         plt.xlabel('Duration of Trip (in secs)')
         plt.title('Trip Duration Distribution')
         plt.show();
```



The plot clearly shows its right skewed.

```
In [26]: # Using log scale to plot as we see a long tail the distribution:
         #First lets check the sum of putliers beyond 6000seconds:
         outliers = ((df['duration_sec'] > 6000))
         outliers.sum()

Out[26]: 909

In [27]: #Lets plot a log scale with values less than 6000 and remove the outliers greater than
         log_filter = df.query('duration_sec < 6000')
         log_binsize = 0.05

         log_bins = 10 ** np.arange(2.4, np.log10(log_filter['duration_sec'].max())+log_binsize,

         #lets plot the histogram with title,ticks and labels:
         plt.figure(figsize=[8,5])
         plt.hist(data = log_filter, x = 'duration_sec', bins = log_bins)
         plt.xscale('log')
         tick_locs=[200,500,1e3,2e3,5e3,1e4]
         plt.xticks(tick_locs,tick_locs)#Set ticks as 1,3,5
         plt.xlabel('Duration in sec')
         plt.ylabel('Number of trips')
         plt.show();
```

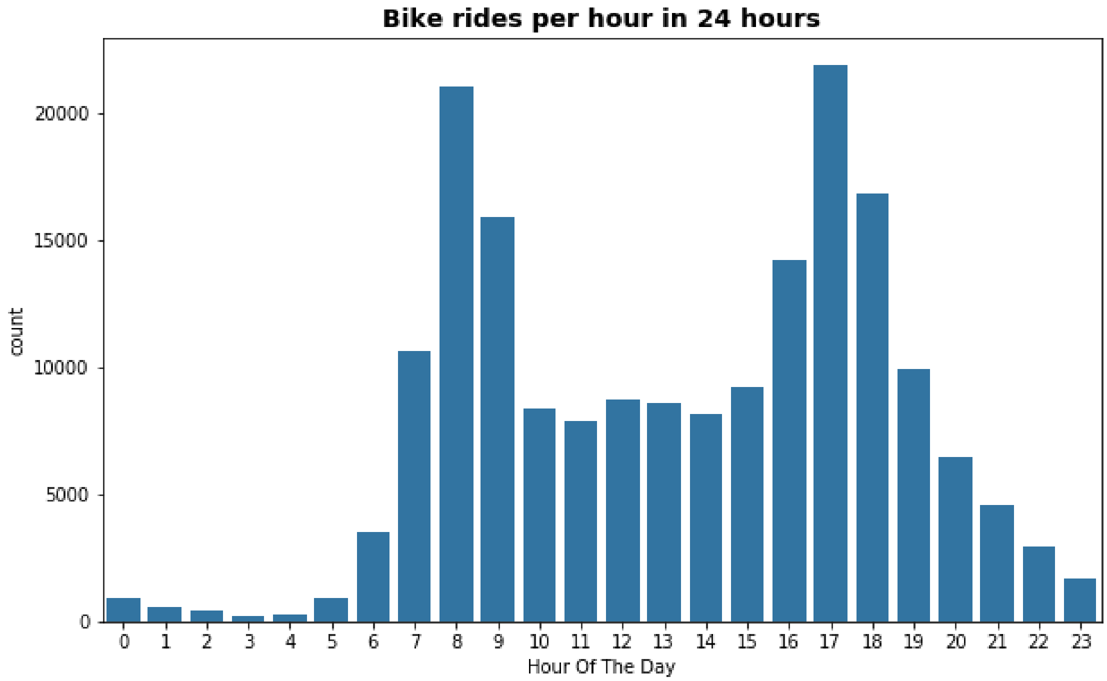


Observation for the above Plot:

After using the log scale to plot the duartion seconds we are able to see a much clearer graph. From the plot we observe that maximum rides around 550 seconds,and almost minimum after 6000 seconds..In this plot we see that the numbers of rides are on the short duration end and very few trips on the long duration. There are about 909 outliers as per the data(as plotted when duration_seconds for greater than 6000).The outliers are reason for long right skewed.

> **Rubric Tip**: Visualizations should depict the data appropriately so that the plots are easily interpretable. You should choose an appropriate plot type, data encodings, and formatting as needed. The formatting may include setting/adding the title, labels, legend, and comments. Also, do not overplot or incorrectly plot ordinal data.

In [28]:
```
#2.Lets plot the start_time per hour basis for 1 day:
#First lets extract the hour from start_time:
df['start_time_hour']= pd.DatetimeIndex(df['start_time']).hour
# Now lets plot the graph :
plt.figure(figsize = (10,6))
base_color = sb.color_palette()[0]
sb.countplot(data = df, x = 'start_time_hour', color = base_color)
plt.title('Bike rides per hour in 24 hours',fontsize = 14, fontweight = 'bold')
plt.xlabel('Hour Of The Day');
```



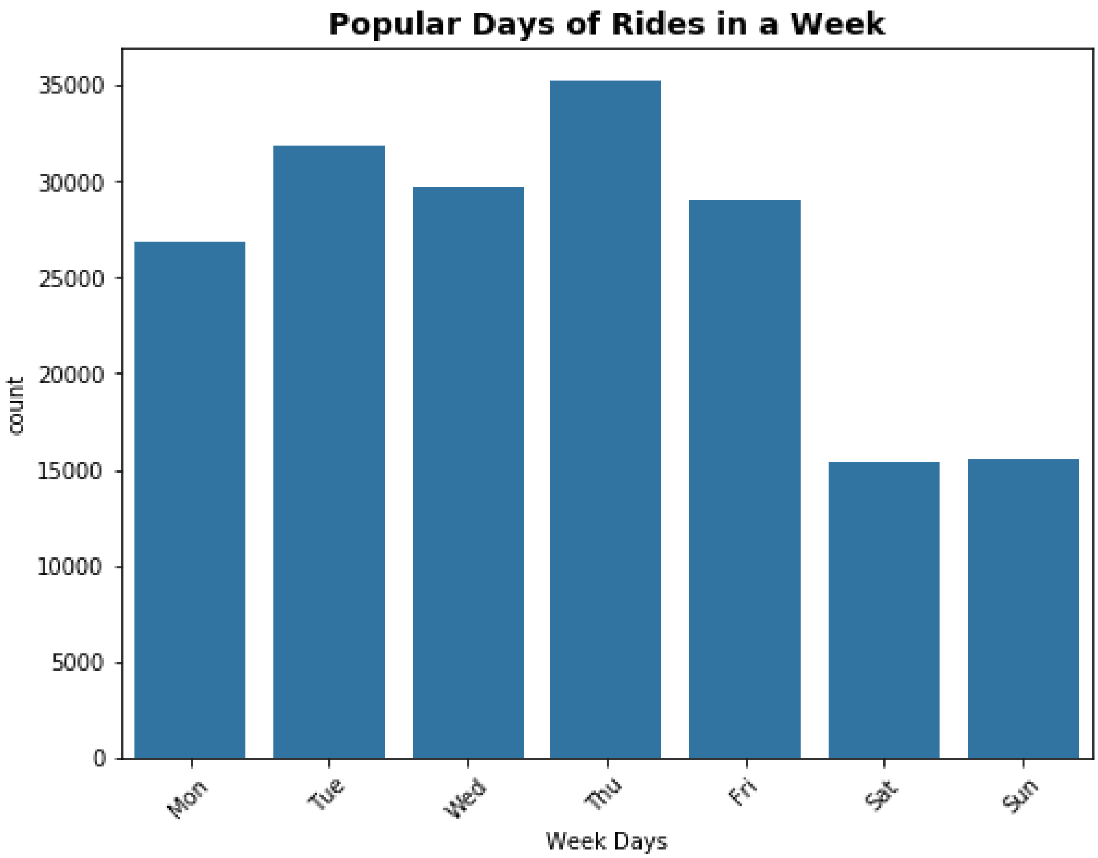
Observation Of the above plot: From the plot we observe that there are two peaks, therefore its a bimodal graph.

Its clear from the above that there are two peak times one in the morning at 8:00am and one in the evening at 17:00 hours.

The times clearly show that the peaks are during office commute hours. This is follwed by 9:00am in the morning and 18:00 hours in the evening. Most bike are hired from 6:00am in the morning to almost 22:00 hours in the night.

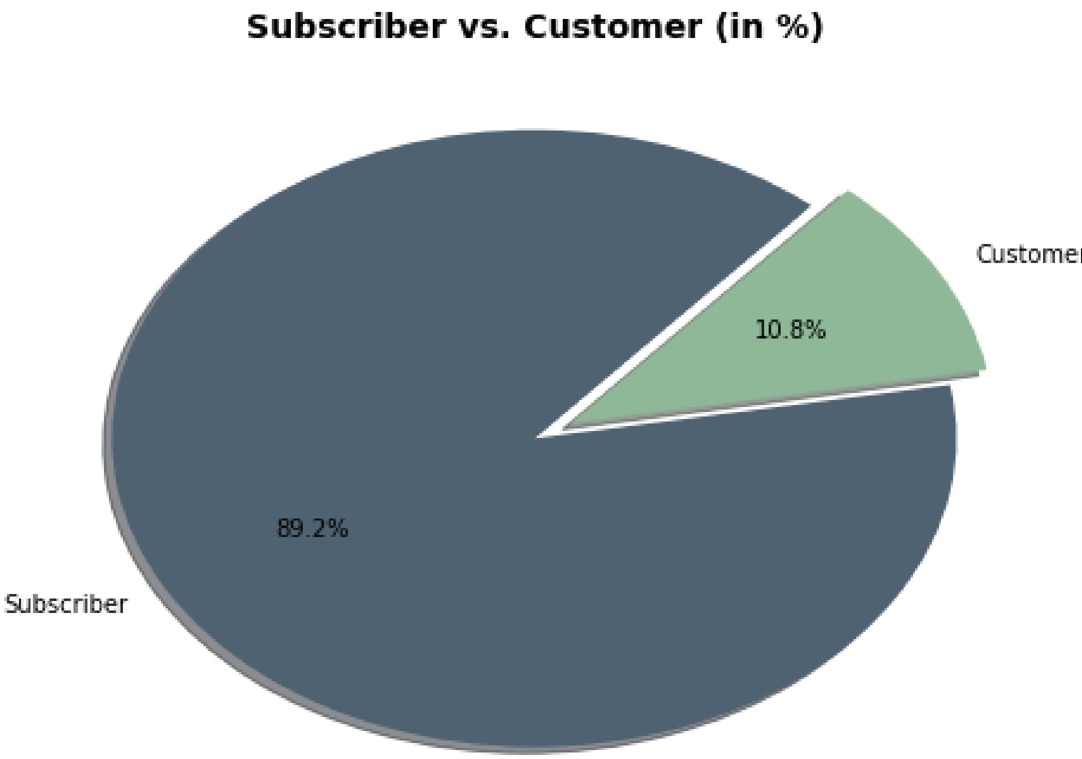
Most active times are during 7:00am to 19:00 hours in the evening.

```
In [29]: #Lets plot the next category popular days of the week:
         #First we need to extract the day of the week from start_time:
         df['day_of_week']= pd.DatetimeIndex(df['start_time']).dayofweek
         df.head()

         #Here we get numbers for day of the week so we need to change dayofweek number to day n
         df['day_of_week'] = df['day_of_week'].apply(lambda x: calendar.day_abbr[x])

         #Now lets plot the graph :
         weekdays = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
         plt.figure(figsize=(8,6))
         base_color = sb.color_palette()[0]
         sb.countplot(data=df, x='day_of_week', color=base_color, order=weekdays)
         plt.title('Popular Days of Rides in a Week', fontsize=14, fontweight='bold')
         plt.xticks(rotation = 45)
         plt.xlabel('Week Days');
```



Observation : The above plot clearly shows that Thursdays are the days bikes are mostly hired followed by Tuesdays Wednesdays and Fridays. Weekends being low compared to other days. Clearly bikes are being hired to reach work places,schools,colleges etc mostly during weekdays so the demand is higher during weekdays.

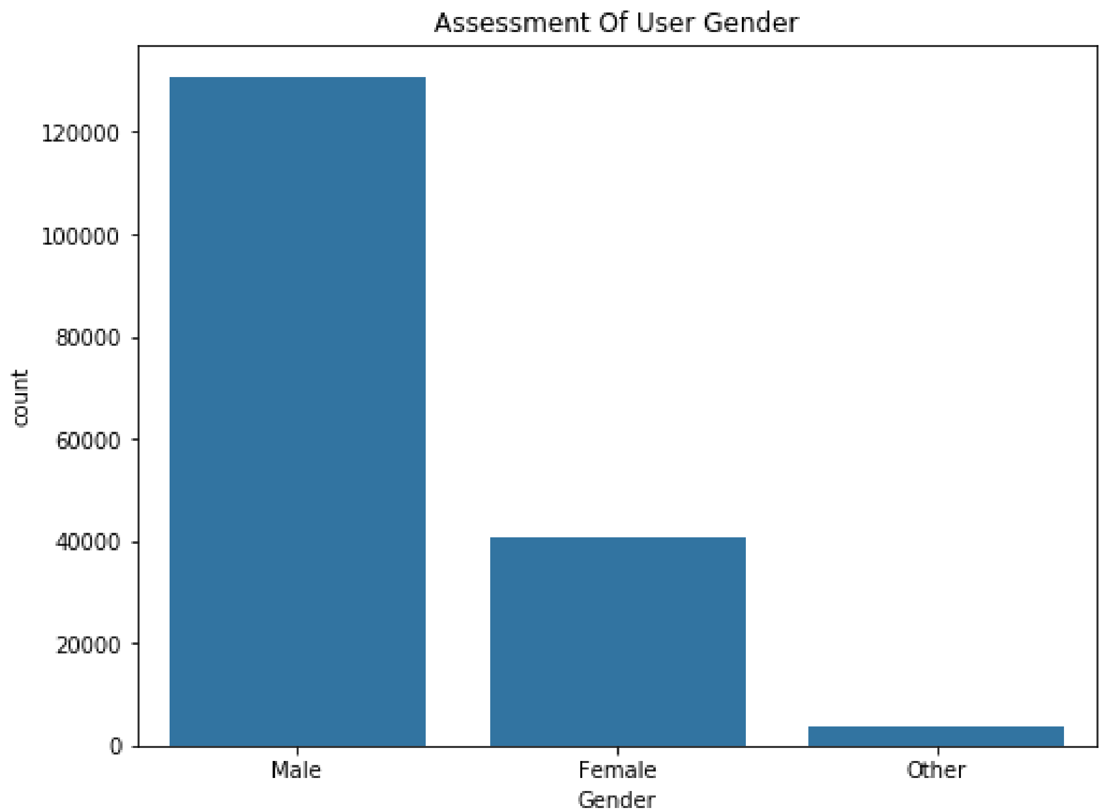
```
In [30]: #Lets look into the user_type now :
         #Now lets plot this on a pie graph:
         plt.figure(figsize=[8,6])
         colors = ['#4F6272', '#8EB897']
         explode = (0, 0.1)
         sorted_counts = df['user_type'].value_counts()
         plt.pie(sorted_counts, explode=explode, labels = sorted_counts.index,
                 autopct='%1.1f%%',shadow=True, startangle = 10,counterclock = False,colors = co
         plt.title('Subscriber vs. Customer (in %)', fontsize=14, fontweight='bold');
```



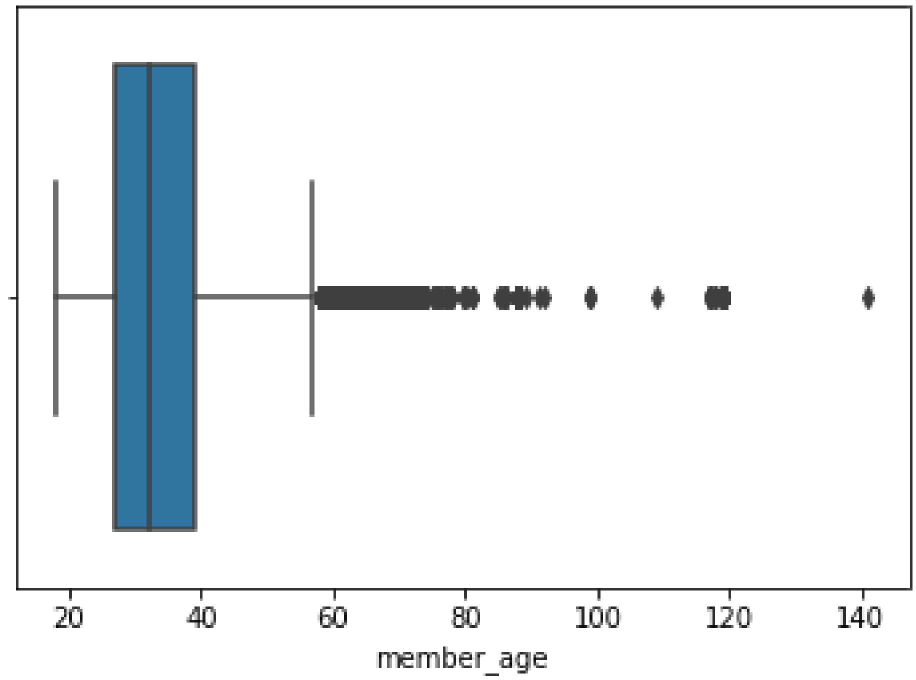
Observation: Clearly majority users are Subscribers or members with 89.2%, the rest being casual customers contributing 10.8%.

```
In [31]: #Lets plot and check member_gender:
         gender = df['member_gender'].value_counts().index
         print(gender)
         #lets plot the graph :
         plt.figure(figsize=[8,6])
         base_color= sb.color_palette()[0]
         sb.countplot(data=df,x='member_gender',color = base_color,order = gender)
         plt.title('Assessment Of User Gender')
         plt.xlabel('Gender')
         plt.show();
```

```
CategoricalIndex(['Male', 'Female', 'Other'], categories=['Female', 'Male', 'Other'], ordered=Fa
```




```
In [32]: #Lets calculate member_age and store data in user_age column:
         df['member_age'] = 2019 - df['member_birth_year']
         df = df[np.isfinite(df['member_age'])]
         df['member_age'].describe()

Out[32]: count    175147.000000
         mean         34.193563
         std          10.116689
         min          18.000000
         25%          27.000000
         50%          32.000000
         75%          39.000000
         max         141.000000
         Name: member_age, dtype: float64

In [33]: #lets plot and check :
         sb.boxplot(data=df, x='member_age');
```



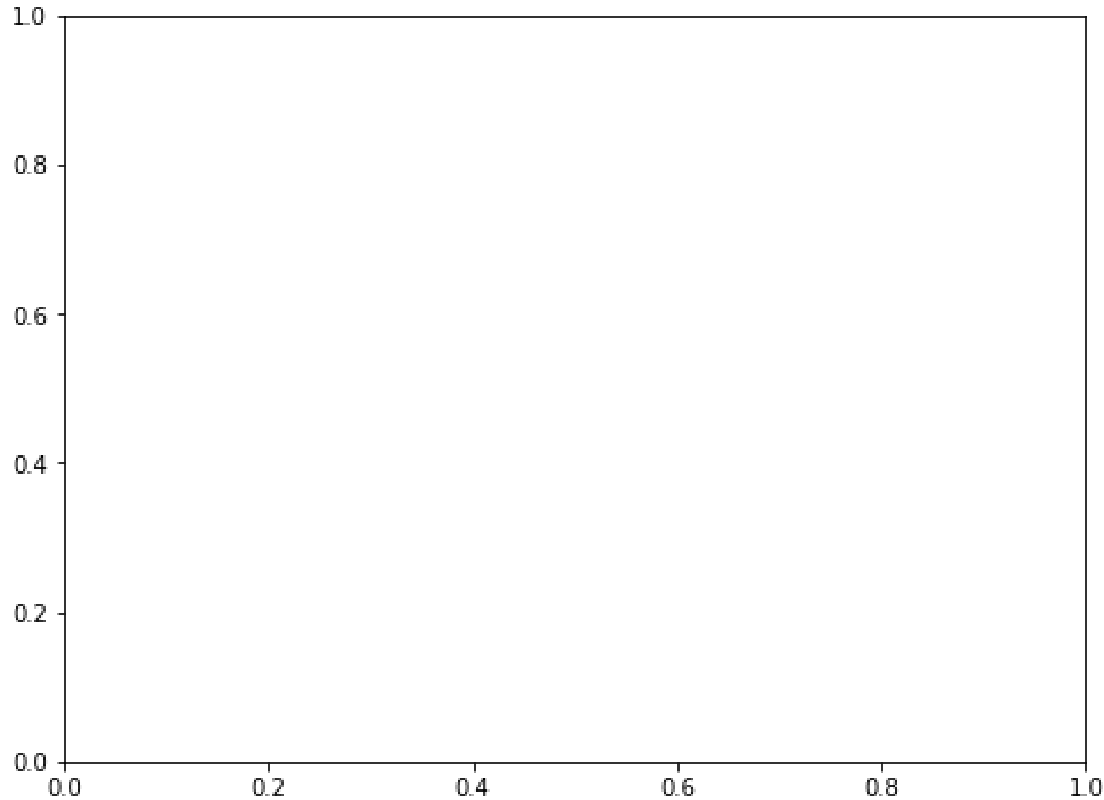
Observation :

```
From the above plot we can clearly see that there are outliers. Here the maximum age shows 140 w
Also we see that after 100 also we many more outliers, lets focus on member age group 100 and be
```

```
In [34]: #Lets drop the members ages that is less than or equal to 80 years of age:
         df['member_age'] = df.query('member_age <=100')

In [35]: #df['member_age']=np.isfinite(df['member_age']).values

In [36]: #lets plot:
         binsize=3
         bin_edges=np.arange(20, df.member_age.max()+binsize, binsize)
         plt.figure(figsize=[8,6])
         plt.hist(data=df, x='member_age', bins=bin_edges)
         plt.xlabel('Age (years)')
         plt.xlim([15,100])
         plt.show()
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-36-8047ed5df807> in <module>()
  3 bin_edges=np.arange(20, df.member_age.max()+binsize, binsize)
```

```
      4 plt.figure(figsize=[8,6])
----> 5 plt.hist(data=df, x='member_age', bins=bin_edges)
      6 plt.xlabel('Age (years)')
      7 plt.xlim([15,100])


    /opt/conda/lib/python3.6/site-packages/matplotlib/pyplot.py in hist(x, bins, range, dens
   3002                       histtype=histtype, align=align, orientation=orientation,
   3003                       rwidth=rwidth, log=log, color=color, label=label,
-> 3004                       stacked=stacked, normed=normed, data=data, **kwargs)
   3005     finally:
   3006         ax._hold = washold


    /opt/conda/lib/python3.6/site-packages/matplotlib/__init__.py in inner(ax, *args, **kwar
   1708                     warnings.warn(msg % (label_namer, func.__name__),
   1709                                   RuntimeWarning, stacklevel=2)
-> 1710             return func(ax, *args, **kwargs)
   1711         pre_doc = inner.__doc__
   1712         if pre_doc is None:


    /opt/conda/lib/python3.6/site-packages/matplotlib/axes/_axes.py in hist(***failed resolv
   6205             # this will automatically overwrite bins,
   6206             # so that each histogram uses the same bins
-> 6207             m, bins = np.histogram(x[i], bins, weights=w[i], **hist_kwargs)
   6208             m = m.astype(float)  # causes problems later if it's an int
   6209             if mlast is None:


    /opt/conda/lib/python3.6/site-packages/numpy/lib/function_base.py in histogram(a, bins,
    667     if not np.all(np.isfinite([mn, mx])):
    668         raise ValueError(
--> 669             'range parameter must be finite.')
    670     if mn == mx:
    671         mn -= 0.5


    ValueError: range parameter must be finite.
```



3.2 Bivariate Exploration

In this section, investigate relationships between pairs of variables in your data. Make sure the variables that you cover here have been introduced in some fashion in the previous section (univariate exploration).

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

Main variables of intrest are duration(in seconds),start_time-from which I have extracted hours in a day and day of the week,user_type,member-gender,member_birth_year(from which I have calculated member_age). I noticed that duration in seconds was not very clear so I used a log function to get a clearer view of the data.I have used start_time column to extract the hours and days of the week.After calculating the member_age I realized there are lot of outliers, so I set a limit of maximum age limit of 100 years inorder for me to get a clear and clean graph.

3.2.2 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?

The two variables with many outliers with large range of values was duration in seconds for which I use log function to transform the data.Also after calculating the member age i found a lot of outliers with large values so I had to limit with a maximum value of 100 years, which is reasonable.

```
In [ ]:
```

3.2.3 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?

Your answer here!

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

Your answer here!

3.3 Multivariate Exploration

Create plots of three or more variables to investigate your data even further. Make sure that your investigations are justified, and follow from your work in the previous sections.

```
In [ ]:
```

3.3.1 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?

Your answer here!

3.3.2 Were there any interesting or surprising interactions between features?

Your answer here!

3.4 Conclusions

You can write a summary of the main findings and reflect on the steps taken during the data exploration.

Remove all Tips mentioned above, before you convert this notebook to PDF/HTML

At the end of your report, make sure that you export the notebook as an html file from the `File > Download as... > HTML or PDF` menu. Make sure you keep track of where the exported file goes, so you can put it in the same folder as this notebook for project submission. Also, make sure you remove all of the quote-formatted guide notes like this one before you finish your report!

```
In [ ]:
```