Part I - Bike Data Exploration

by Hastings Sibanda

Introduction

This document explores data from Bike Rentals in the San Francisco Bay Area for a company called Lyft. It details over 180,000 trips made by riders in February 2019.

Importing necessary libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import haversine as hs
```

Gathering Data

In [2]:
```
bike_df = pd.read_csv("201902-fordgobike-tripdata.csv")
bike_df.shape
```

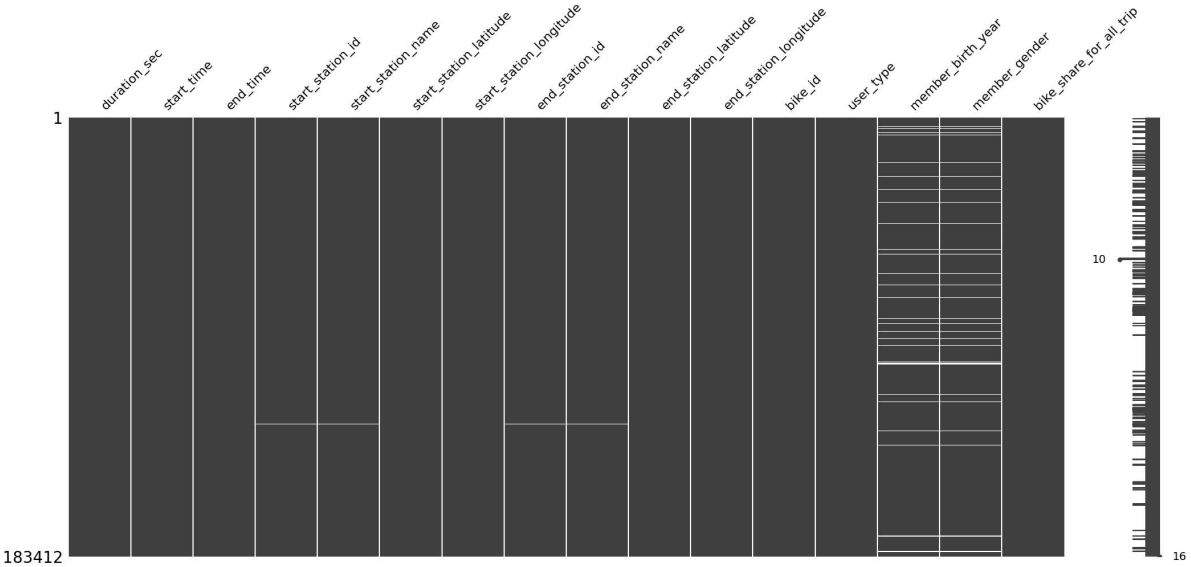
Out[2]:
```
(183412, 16)
```

Assessing Data

In [3]:
```
bike_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 16 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   duration_sec             183412 non-null  int64  
 1   start_time               183412 non-null  object 
 2   end_time                 183412 non-null  object 
 3   start_station_id         183215 non-null  float64
 4   start_station_name       183215 non-null  object 
 5   start_station_latitude   183412 non-null  float64
 6   start_station_longitude  183412 non-null  float64
 7   end_station_id           183215 non-null  float64
 8   end_station_name         183215 non-null  object 
 9   end_station_latitude     183412 non-null  float64
 10  end_station_longitude    183412 non-null  float64
 11  bike_id                  183412 non-null  int64  
 12  user_type                183412 non-null  object 
 13  member_birth_year        175147 non-null  float64
 14  member_gender            175147 non-null  object 
 15  bike_share_for_all_trip  183412 non-null  object 
dtypes: float64(7), int64(2), object(7)
memory usage: 22.4+ MB
```

In [4]:
```
import missingno as msno
msno.matrix(bike_df);
```



Missing data in columns:

- start_station_id
- start_station_name
- end_station_id
- end_station_name
- member_birth_year
- member_gender

from the matrix, its probably the same people with missing member_birth_year who have missing member_gender.

In [5]:
```
bike_df.head()
```

Out[5]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latit
0	52185	2019-02-28 17:32:10.1450	2019-03-01 08:01:55.9750	21.0	Montgomery St BART Station (Market St at 2nd St)	37.789
1	42521	2019-02-28 18:53:21.7890	2019-03-01 06:42:03.0560	23.0	The Embarcadero at Steuart St	37.791
2	61854	2019-02-28 12:13:13.2180	2019-03-01 05:24:08.1460	86.0	Market St at Dolores St	37.769
3	36490	2019-02-28 17:54:26.0100	2019-03-01 04:02:36.8420	375.0	Grove St at Masonic Ave	37.774
4	1585	2019-02-28 23:54:18.5490	2019-03-01 00:20:44.0740	7.0	Frank H Ogawa Plaza	37.804

In [6]:
```
bike_df['start_station_id'].nunique()
```

Out[6]:
```
329
```

In [7]:
```
bike_df['end_station_id'].nunique()
```

Out[7]:
```
329
```

In [8]:
```
bike_df['start_station_latitude'].nunique()
```

Out[8]:
```
334
```

Incorrect datatypes for columns:

- start_time
- end_time

In [9]:
```
bike_df['bike_id'].nunique()
```

Out[9]:
```
4646
```

There are 4646 different bicycles in this dataset.

In [10]:
```
bike_df.describe()
```

Out[10]:

	duration_sec	start_station_id	start_station_latitude	start_station_longitude	end_station_id
count	183412.000000	183215.000000	183412.000000	183412.000000	183215.000000
mean	726.078435	138.590427	37.771223	-122.352664	136.249123
std	1794.389780	111.778864	0.099581	0.117097	111.515131
min	61.000000	3.000000	37.317298	-122.453704	3.000000
25%	325.000000	47.000000	37.770083	-122.412408	44.000000
50%	514.000000	104.000000	37.780760	-122.398285	100.000000
75%	796.000000	239.000000	37.797280	-122.286533	235.000000
max	85444.000000	398.000000	37.880222	-121.874119	398.000000

Cleaning Data

We will make a copy of the dataset first

Define

correct datatypes for start_time and end_time

In [11]:
```
bike_clean = bike_df.copy()
```

Code

In [12]:
```
bike_clean['start_time'] = pd.to_datetime(bike_clean['start_time'])
bike_clean['end_time'] = pd.to_datetime(bike_clean['end_time'])
```

Test

In [13]:
```
print(bike_clean['start_time'].dtype)
print(bike_clean['end_time'].dtype)
```

```
datetime64[ns]
datetime64[ns]
```

Define

Drop missing data for columns: start_station_id start_station_name end_station_id end_station_name member_birth_year member_gender

Code

In [14]:
```
bike_clean.isnull().sum()
```

Out[14]:
```
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 [15]:
```
bike_clean.shape
```

Out[15]:
```
(183412, 16)
```

In [16]:
```
# Missing data is just 4.5% of the whole dataset, so we will drop it for now
bike_clean.dropna(inplace=True)
```

Test

In [17]:
```
bike_clean.isnull().sum()
```

Out[17]:
```
duration_sec               0
start_time                 0
end_time                   0
start_station_id           0
start_station_name         0
start_station_latitude     0
start_station_longitude    0
end_station_id             0
end_station_name           0
end_station_latitude       0
end_station_longitude      0
bike_id                    0
user_type                  0
member_birth_year          0
member_gender              0
bike_share_for_all_trip    0
dtype: int64
```

In [18]:
```
bike_clean.head()
```

Out[18]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitud
0	52185	2019-02-28 17:32:10.145	2019-03-01 08:01:55.975	21.0	Montgomery St BART Station (Market St at 2nd St)	37.7896
2	61854	2019-02-28 12:13:13.218	2019-03-01 05:24:08.146	86.0	Market St at Dolores St	37.7693
3	36490	2019-02-28 17:54:26.010	2019-03-01 04:02:36.842	375.0	Grove St at Masonic Ave	37.7748
4	1585	2019-02-28 23:54:18.549	2019-03-01 00:20:44.074	7.0	Frank H Ogawa Plaza	37.8045
5	1793	2019-02-28 23:49:58.632	2019-03-01 00:19:51.760	93.0	4th St at Mission Bay Blvd S	37.7704

In [19]:
```
bike_clean.reset_index(drop=True, inplace=True)
```

Define

Change datatypes for end_station_id, start_station_id to int

Code

In [20]:
```
bike_clean['start_station_id'] = bike_clean['start_station_id'].astype(int)
bike_clean['end_station_id'] = bike_clean['end_station_id'].astype(int)
```

Test

In [21]:
```
bike_clean['start_station_id'].dtype
```

Out[21]:
```
dtype('int32')
```

In [22]:
```
bike_clean['end_station_id'].dtype
```

Out[22]:
```
dtype('int32')
```

Feature Engineering

Define

Creating a new feature called start_trip_date from start_trip_time

Code

In [23]:
```
bike_clean['start_trip_date'] = pd.to_datetime(bike_clean['start_time']).dt.date
```

In [24]:
```
bike_clean.loc[0, 'start_trip_date']
```

Out[24]:
```
datetime.date(2019, 2, 28)
```

Test

In [25]:
```
bike_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 174952 entries, 0 to 174951
Data columns (total 17 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   duration_sec             174952 non-null  int64
 1   start_time               174952 non-null  datetime64[ns]
 2   end_time                 174952 non-null  datetime64[ns]
 3   start_station_id         174952 non-null  int32
 4   start_station_name       174952 non-null  object
 5   start_station_latitude   174952 non-null  float64
 6   start_station_longitude  174952 non-null  float64
 7   end_station_id           174952 non-null  int32
 8   end_station_name         174952 non-null  object
 9   end_station_latitude     174952 non-null  float64
 10  end_station_longitude    174952 non-null  float64
 11  bike_id                  174952 non-null  int64
 12  user_type                174952 non-null  object
 13  member_birth_year        174952 non-null  float64
 14  member_gender            174952 non-null  object
 15  bike_share_for_all_trip  174952 non-null  object
 16  start_trip_date          174952 non-null  object
dtypes: datetime64[ns](2), float64(5), int32(2), int64(2), object(6)
memory usage: 21.4+ MB
```

Define

Compute Age from member_birth_year column, and drop the column

Code

In [26]:
```
bike_clean['start_trip_date']
```

Out[26]:
```
0         2019-02-28
1         2019-02-28
2         2019-02-28
3         2019-02-28
4         2019-02-28
             ...
174947    2019-02-01
174948    2019-02-01
174949    2019-02-01
174950    2019-02-01
174951    2019-02-01
Name: start_trip_date, Length: 174952, dtype: object
```

In [27]:
```
bike_clean['member_age'] = bike_clean['start_time'].dt.year - bike_clean['member_b
```

In [28]:
```
#we will use the start_time of the trip, to compute member age at the time they to
bike_clean['start_time'].max().year
```

Out[28]:
```
2019
```

In [29]:
```
#change member_age data type to int, then drop member_birth_year column
bike_clean['member_age'] = bike_clean['member_age'].astype(int)
bike_clean.drop(columns=['member_birth_year'], inplace=True)
```

Test

In [30]:
```
bike_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 174952 entries, 0 to 174951
Data columns (total 17 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   duration_sec             174952 non-null  int64
 1   start_time               174952 non-null  datetime64[ns]
 2   end_time                 174952 non-null  datetime64[ns]
 3   start_station_id         174952 non-null  int32
 4   start_station_name       174952 non-null  object
 5   start_station_latitude   174952 non-null  float64
 6   start_station_longitude  174952 non-null  float64
 7   end_station_id           174952 non-null  int32
 8   end_station_name         174952 non-null  object
 9   end_station_latitude     174952 non-null  float64
 10  end_station_longitude    174952 non-null  float64
 11  bike_id                  174952 non-null  int64
 12  user_type                174952 non-null  object
 13  member_gender            174952 non-null  object
 14  bike_share_for_all_trip  174952 non-null  object
 15  start_trip_date          174952 non-null  object
 16  member_age               174952 non-null  int32
dtypes: datetime64[ns](2), float64(4), int32(3), int64(2), object(6)
memory usage: 20.7+ MB
```

Define

Extract distance between start_station and end_station

Code

In [31]:
```
#using haversian distance formula to calculate distance between two geographical po
#and latitude
distances = []
for ind in bike_clean.index:
        loc1 = (bike_clean['start_station_latitude'][ind], bike_clean['start_static
        loc2 = (bike_clean['end_station_latitude'][ind], bike_clean['end_station_lo
        distance = np.round(hs.haversine(loc1, loc2),2)
        distances.append(distance)
```

In [32]:
```
distances[1]
```

Out[32]:
```
2.7
```

In [33]:
```
bike_clean['trip_distance'] = distances
```

Test

In [34]:
```
bike_clean['trip_distance'].describe()
```

Out[34]:
```
count    174952.000000
mean          1.690052
std           1.097011
min           0.000000
25%           0.910000
50%           1.430000
75%           2.220000
max          69.470000
Name: trip_distance, dtype: float64
```

In [35]:
```
bike_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 174952 entries, 0 to 174951
Data columns (total 18 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   duration_sec             174952 non-null  int64
 1   start_time               174952 non-null  datetime64[ns]
 2   end_time                 174952 non-null  datetime64[ns]
 3   start_station_id         174952 non-null  int32
 4   start_station_name       174952 non-null  object
 5   start_station_latitude   174952 non-null  float64
 6   start_station_longitude  174952 non-null  float64
 7   end_station_id           174952 non-null  int32
 8   end_station_name         174952 non-null  object
 9   end_station_latitude     174952 non-null  float64
 10  end_station_longitude    174952 non-null  float64
 11  bike_id                  174952 non-null  int64
 12  user_type                174952 non-null  object
 13  member_gender            174952 non-null  object
 14  bike_share_for_all_trip  174952 non-null  object
 15  start_trip_date          174952 non-null  object
 16  member_age               174952 non-null  int32
 17  trip_distance            174952 non-null  float64
dtypes: datetime64[ns](2), float64(5), int32(3), int64(2), object(6)
memory usage: 22.0+ MB
```

Dataset Structure:

The dataset consists of more than 170,000 unique rows of trip data, detailed by 18 columns.

Interesting features to be explored:

Qualitative features:

- user_type
- member_gender
- bike_share_for_all_trip

Quantitative features:

- member_age
- duration_sec

Geographic features

- start_station_longitude
- start_station_latitude
- end_station_longitude
- end_station_latitude

Time Series

- start_time
- trip_start_date
- end_time

Exploratory Data Analysis

Univariate Analysis

Question

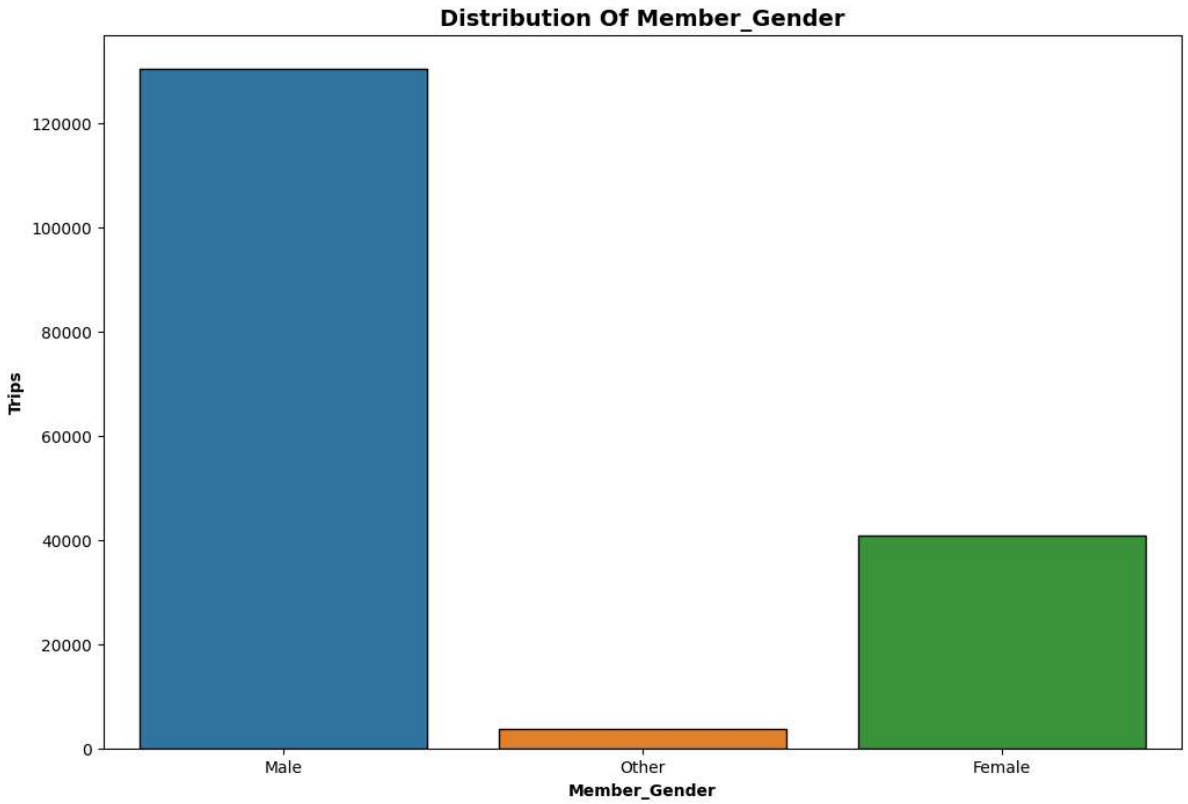
What does the gender distribution in the dataset look like?

Visualization

In [36]:
```
def uni_CountPlot(df, xVar):

    #set the plot parameters
    plt.figure(figsize=(12,8))

    #plot
    sns.countplot(data=df, x=xVar, edgecolor='black');

    #add title and format it
    plt.title(f'''Distribution of {xVar}'''.title(), fontsize=14, weight='bold')

    #add x label and format it
    plt.xlabel(xVar.title(), fontsize=10, weight='bold')
```

```
    #add y label and format it
    plt.ylabel('Trips'.title(), fontsize=10, weight='bold')
```

In [37]:
```
uni_CountPlot(bike_clean, 'member_gender')
# plt_count_trips('member_gender', "Gender distribution among bikers", "Gender")

# sns.countplot(
#     data = bike_clean,
#     x = 'member_gender'
# );
# plt.title("Gender distribution among bikers");
# plt.xlabel("Gender");
# plt.ylabel("Number of Trips");
```



Observation

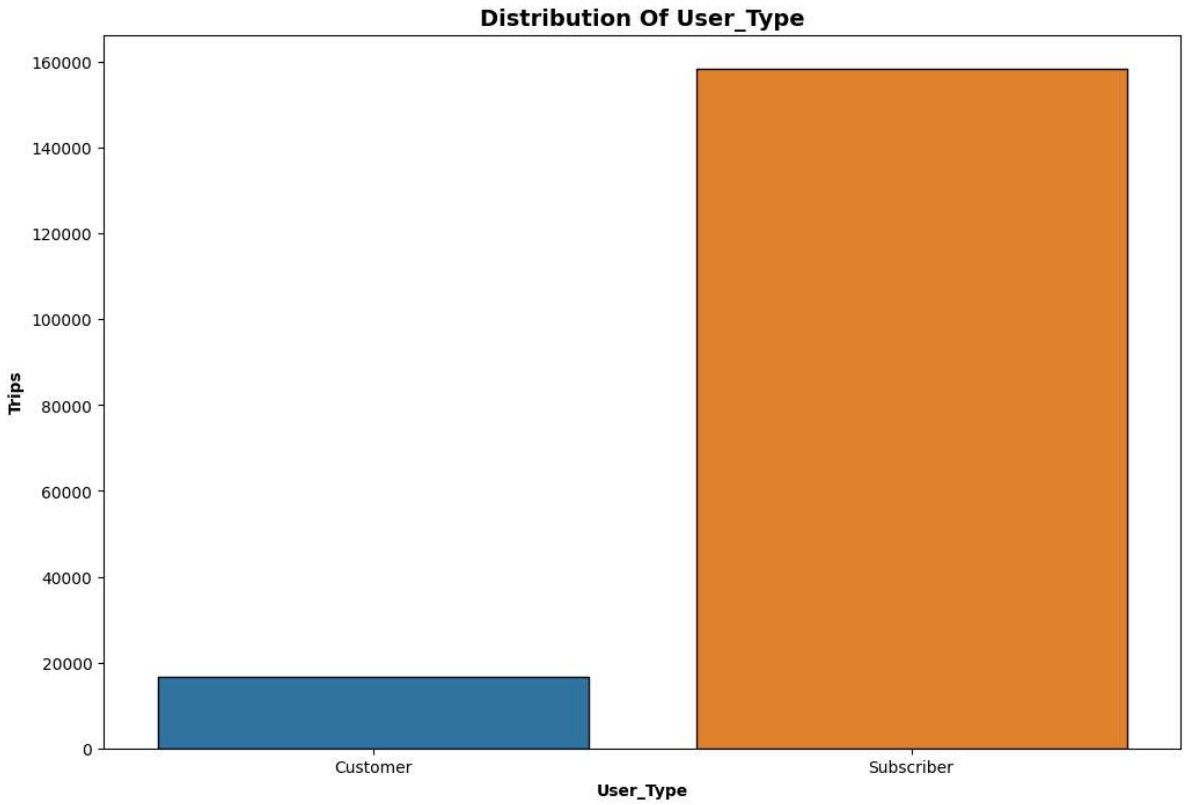
Males dominate the other 2 gender types. Gender type categorized as 'Other' has the least number of bikers.

Question

Let's see the composition in trips by type of user.

Visualization

In [38]:
```
uni_CountPlot(bike_clean, 'user_type')
# sns.countplot(
#     data = bike_clean,
#     x = 'user_type'
# );
# plt.title("User type distribution");
# plt.xlabel("User");
# plt.ylabel("Number of trips");
```



Observation

There are about 8-times more rides by Subscribers than Customers

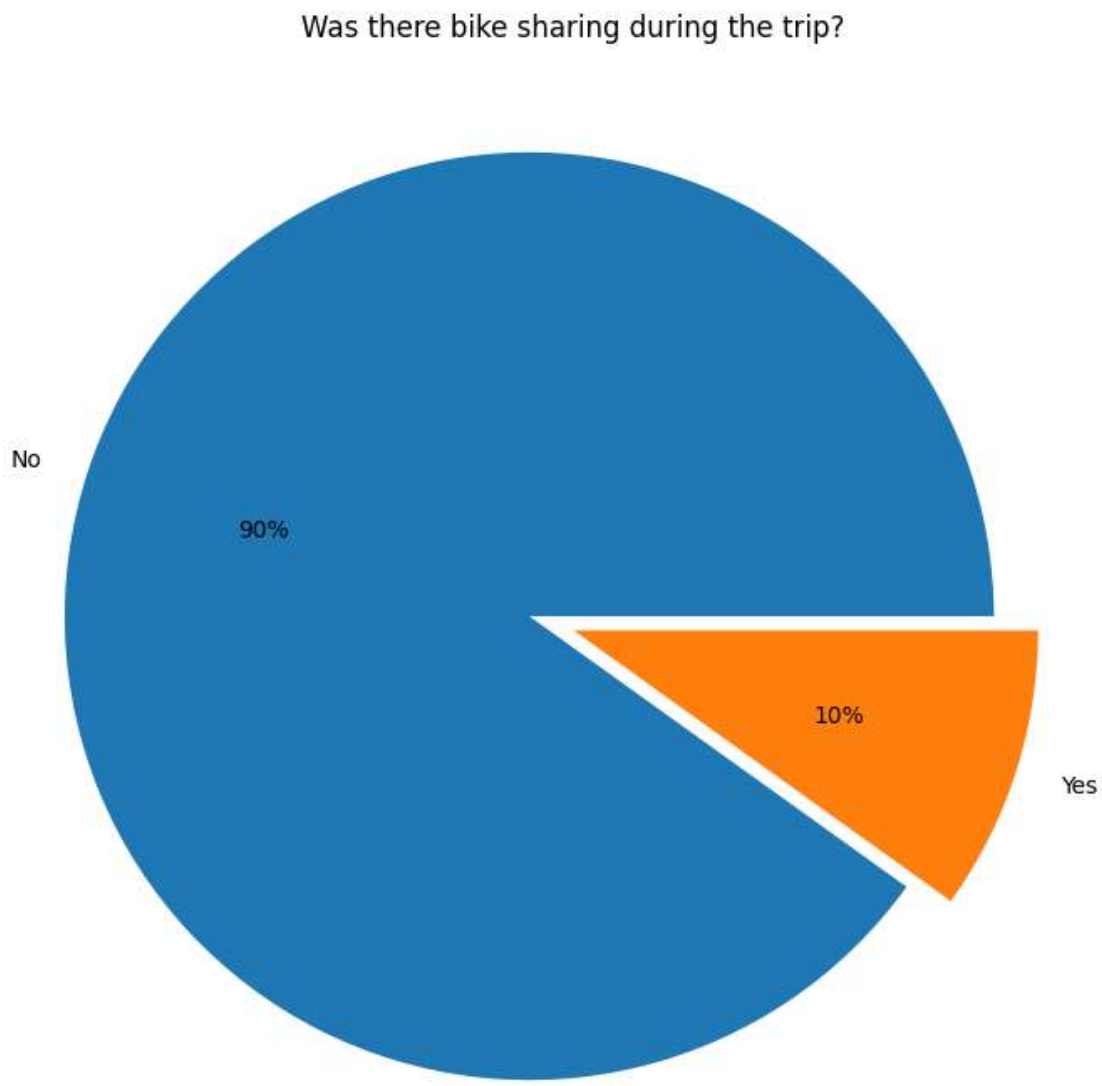
Question

Do the clients share bikes on trips?

Visualization

In [39]:
```
share_trip_counts = bike_clean['bike_share_for_all_trip'].value_counts()
```

In [40]:
```
explode = (0.1, 0)
plt.figure(figsize=(9,9))
plt.pie(
    share_trip_counts,
    labels = share_trip_counts.index,
    autopct='%1.0f%%',
    explode=explode
);
plt.title("Was there bike sharing during the trip?");
```



Observation

Majority of the clients preferred not to share bikes in this particular period.

Question

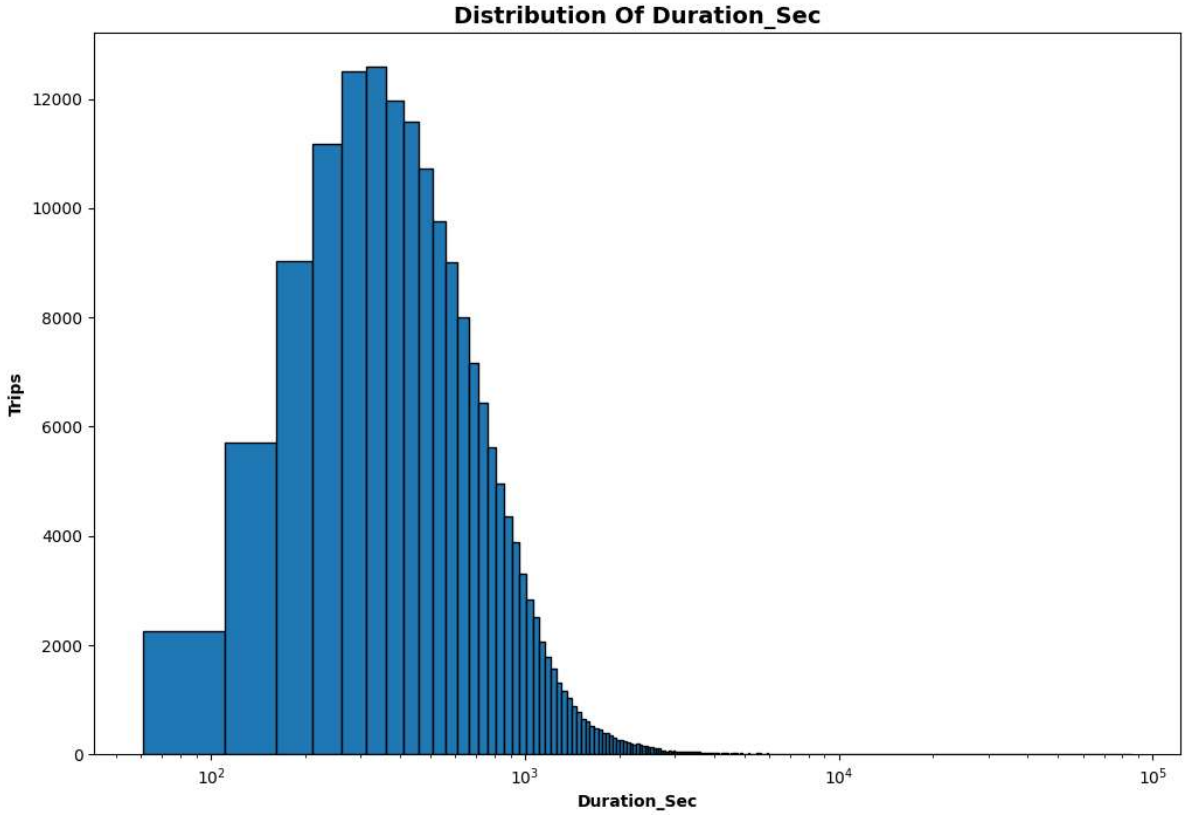
Let us check the distribution for the duration of trips.

Visualization

In [41]:
```
def uni_HistPlot(df, xVar, bins):

    #set the plot parameters
    plt.figure(figsize=(12,8))

    #plot
    plt.hist(data=df, x=xVar, edgecolor='black', bins = bins);

    #add title and format it
    plt.title(f'''Distribution of {xVar}'''.title(), fontsize=14, weight='bold')
```

```
    #add x label and format it
    plt.xlabel(xVar.title(), fontsize=10, weight='bold')

    #add y label and format it
    plt.ylabel('Trips'.title(), fontsize=10, weight='bold')
```

In [42]:
```
bins = np.arange(bike_df['duration_sec'].min(), bike_df['duration_sec'].max() , 50)

uni_HistPlot(bike_clean, 'duration_sec', bins)
plt.xscale('log');
# plt.figure(figsize=(12,8))
# plt.xscale('log')
# plt.hist(
#     data = bike_clean,
#     x = 'duration_sec',
#     bins = bins
# );
# plt.title("Trip duration distribution");
# plt.xlabel("Trip duration (seconds)");
# plt.ylabel("Number of trips");
```



In [43]:
```
bike_clean['duration_sec'].describe()
```

Out[43]:
```
count    174952.000000
mean        704.002744
std        1642.204905
min          61.000000
25%         323.000000
50%         510.000000
75%         789.000000
max       84548.000000
Name: duration_sec, dtype: float64
```

Observation

The distribution is heavily right skewed, with the bulk of trips lasting under 800 seconds.

Question

Lets have a look the ages of the riders.

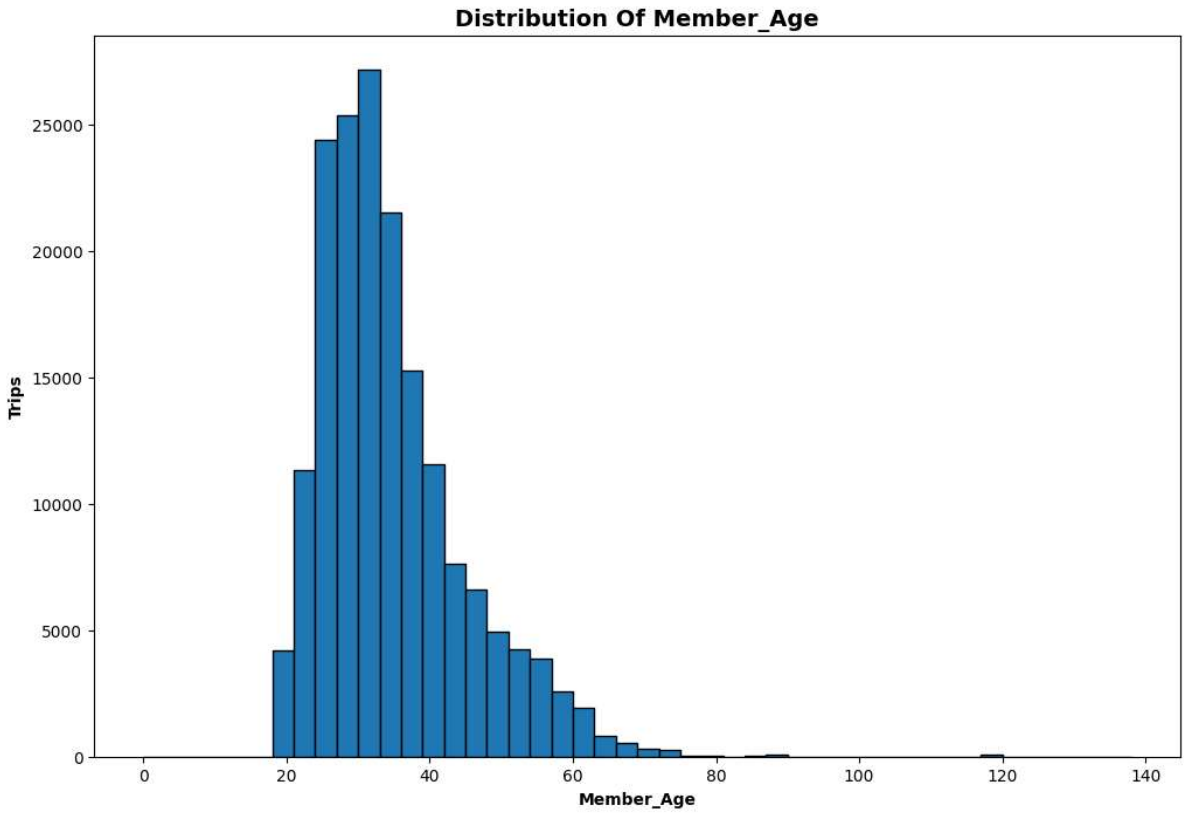
Visualization

In [44]:
```
bike_clean['member_age'].describe()
```

Out[44]:
```
count    174952.000000
mean         34.196865
std          10.118731
min          18.000000
25%          27.000000
50%          32.000000
75%          39.000000
max         141.000000
Name: member_age, dtype: float64
```

Quite bizarre that the maximum bike rider age is 141 years old. Lets explore the distribution.

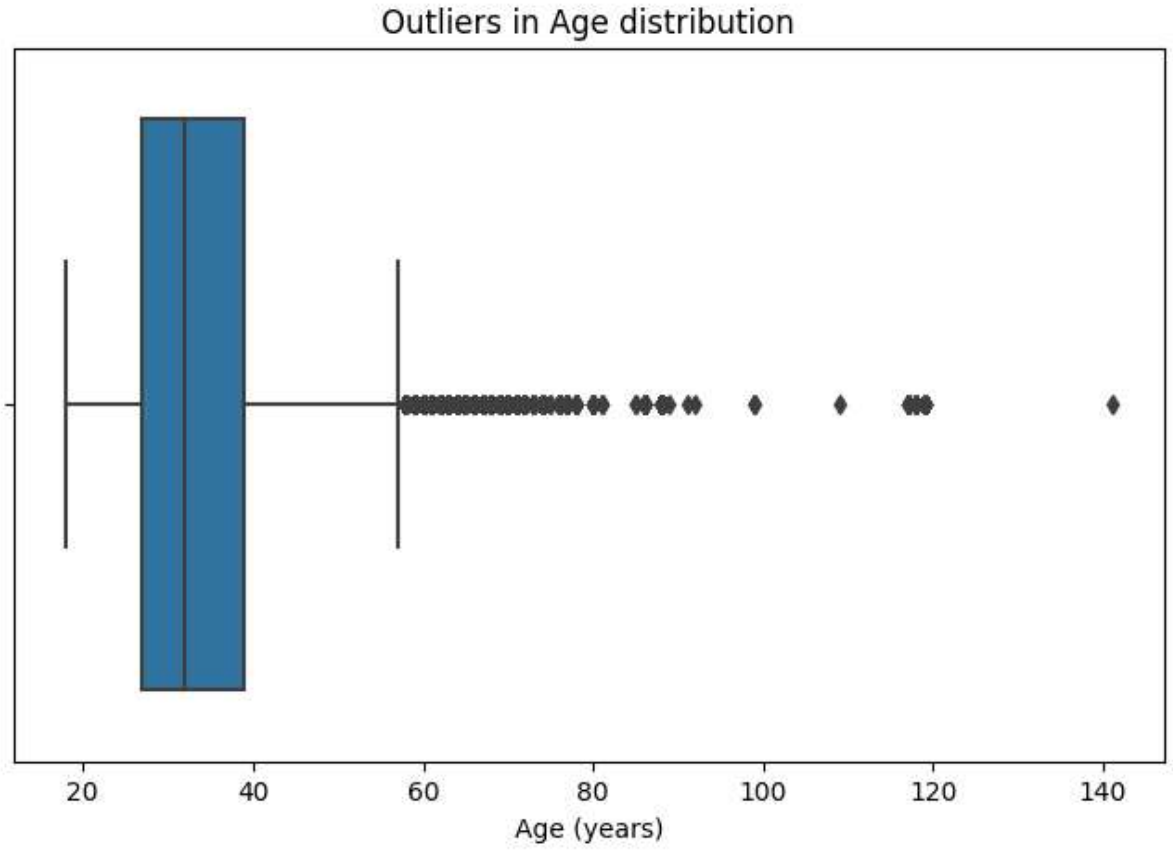
In [45]:
```
bins = np.arange(0, bike_clean['member_age'].max(), 3)
uni_HistPlot(bike_clean, 'member_age', bins)
# plt.hist(
#     data = bike_clean,
#     x = 'member_age',
#     bins = bins
# );
# plt.title("Age distribution");
# plt.xlabel("Age (years)");
# plt.ylabel("Number of riders");
```



Observation

The age distribution is unimodal and right-skewed, with the majority of the rider's ages are in their 20s and 30s. It makes sense because that is the age range when the normal human being is most physically active. However, there are visible outliers, and we will explore those next.

In [46]:
```
plt.figure(figsize=(8,5))
sns.boxplot(
    data = bike_clean,
    x = 'member_age',
);
plt.title("Outliers in Age distribution");
plt.xlabel("Age (years)");
```



Observation

The outliers span across the different gender groups.

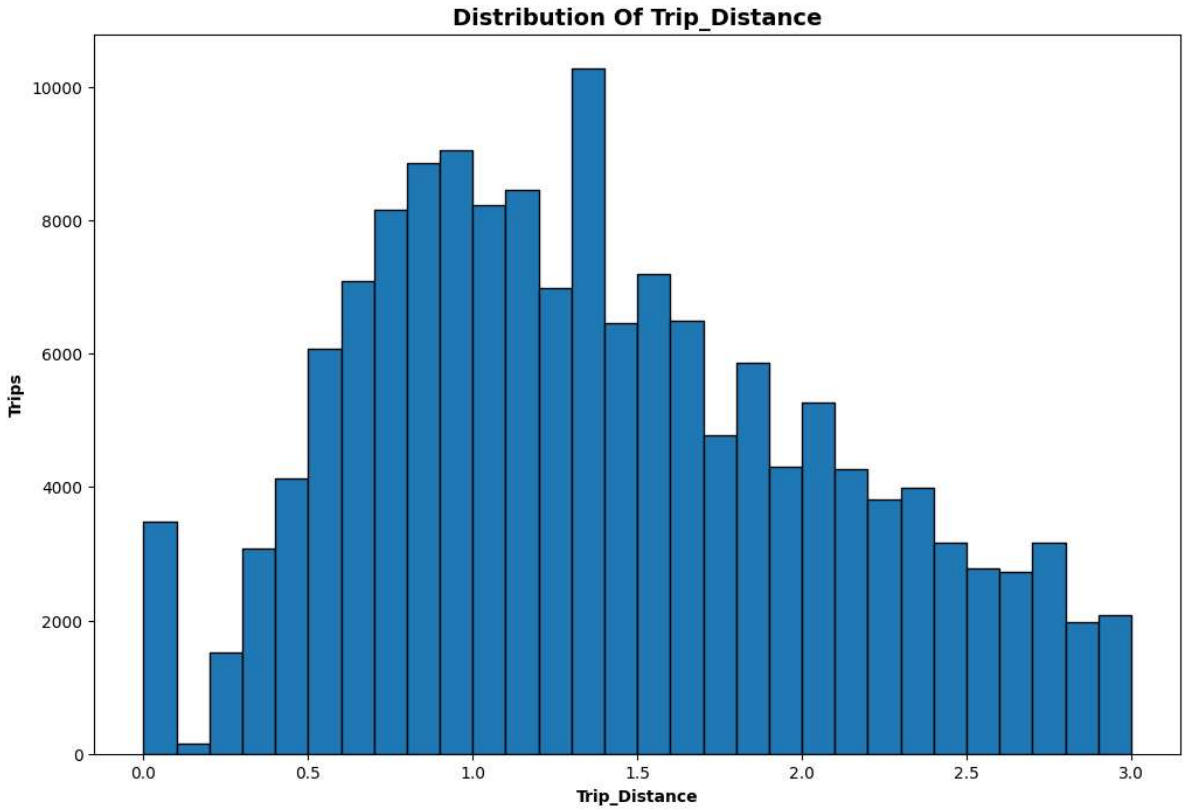
Question

Moving on to see the distances travelled during the trips.

Visualization

In [47]:
```
step_d = 0.1 #step distance
bins = np.arange(0, 3 + step_d, step = step_d)
uni_HistPlot(bike_clean, 'trip_distance', bins)
# plt.hist(
#     data = bike_clean,
#      x = 'trip_distance',
#      bins = bins
```

```
# );
# plt.title("Trip distance distribution");
# plt.xlabel("Distance (kms)");
# plt.ylabel("Number of trips");
```



Observation

It might be that those with total trip distances of zero, ended where their trips started. The distribution is right skewed as well.

Question

We will delve more into the zero distance trips to ascertain if the end station is indeed the start station.

In [48]:
```
df_trip_zero = bike_clean.query('trip_distance == 0')
```

In [49]:
```
df_trip_zero[['start_station_name', 'end_station_name']].sample(10)
```

Out[49]:

	start_station_name	end_station_name
154654	Bryant St at 2nd St	Bryant St at 2nd St
17332	Folsom St at 19th St	Folsom St at 19th St
17901	Myrtle St at Polk St	Myrtle St at Polk St
51961	Howard St at Mary St	Howard St at Mary St
91255	Valencia St at Clinton Park	Valencia St at Clinton Park
126029	Valencia St at 22nd St	Valencia St at 22nd St
82016	Mission Dolores Park	Mission Dolores Park
143050	O'Farrell St at Divisadero St	O'Farrell St at Divisadero St
70089	18th St at Noe St	18th St at Noe St
141712	Precita Park	Precita Park

Observation

For sure those with 0 kms distance travelled, had their trips end where they started.

Question

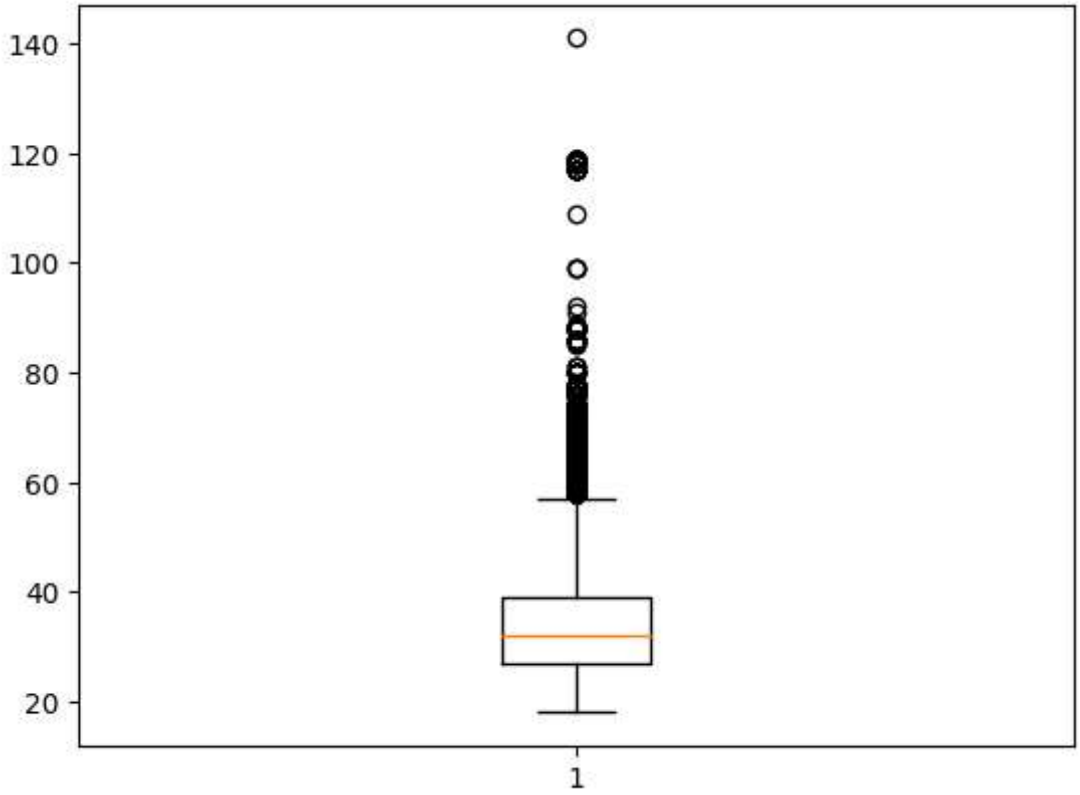
I am still interested to know how the distribution of distances travelled by the outliers in age look like. So I will create a dataset and investigate.

Visualization

In [50]:
```
bike_clean['member_age'].describe()
```

Out[50]:
```
count    174952.000000
mean         34.196865
std          10.118731
min          18.000000
25%          27.000000
50%          32.000000
75%          39.000000
max         141.000000
Name: member_age, dtype: float64
```

In [51]:
```
#creating a boxplot dictionary to extract statistics so we can see the max age befo
bp = plt.boxplot(
    data = bike_clean,
    x = 'member_age'
)
```



In [52]:
```
bp.keys()
```

Out[52]:
```
dict_keys(['whiskers', 'caps', 'boxes', 'medians', 'fliers', 'means'])
```

In [53]:
```
for key in bp:
    print(f'{key}: {[item.get_ydata() for item in bp[key]]}\n')
```

```
whiskers: [array([27., 18.]), array([39., 57.])]

caps: [array([18, 18]), array([57, 57])]

boxes: [array([27., 27., 39., 39., 27.])]

medians: [array([32., 32.])]

fliers: [array([60, 60, 60, ..., 69, 74, 92])]

means: []
```

Observation

The maximum and minimum values of the age distribution, besides outliers are in the *caps* array.

In [54]:
```
#creating a dataframe of ages under the maximum
df_age_outliers = bike_clean.query('member_age > 57')
df_age_outliers.shape
```

Out[54]:
```
(5781, 18)
```

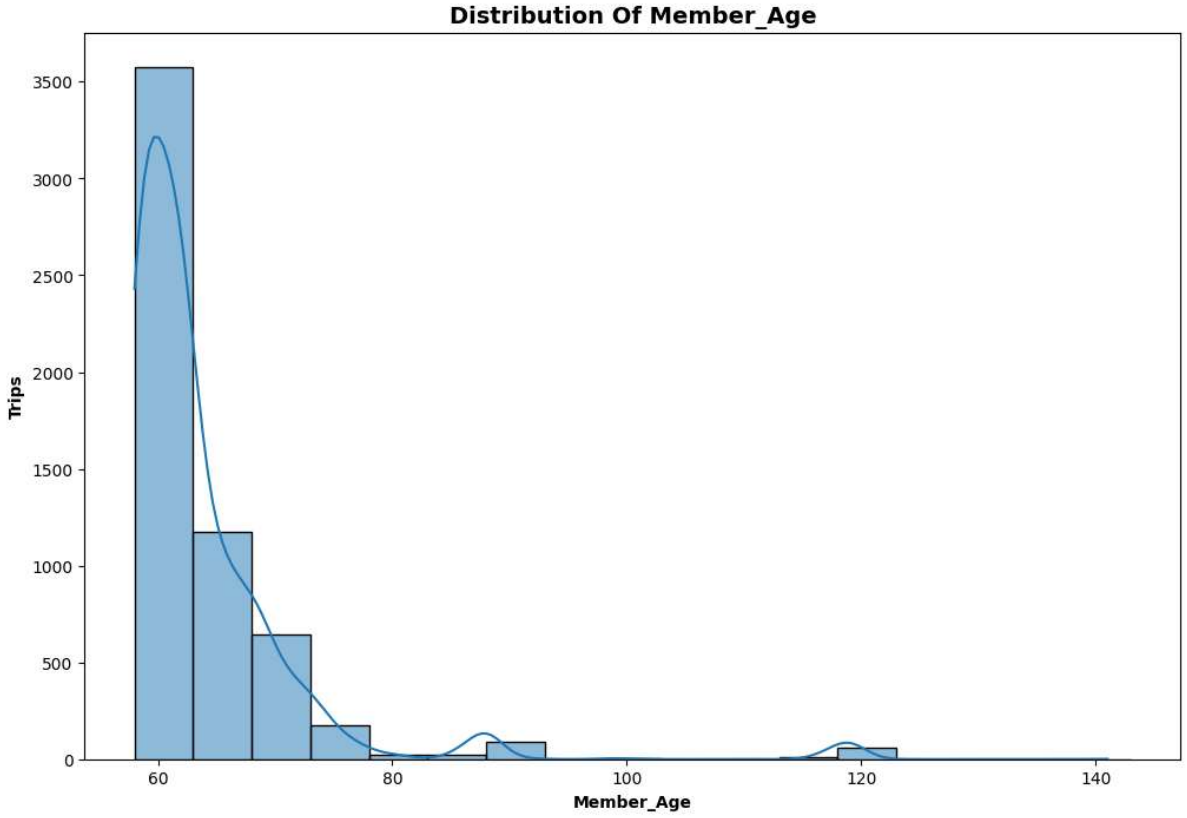
Observation

We have 5781 trips by members above the age of 57 in one month. Quite interesting. Lets see the age distribution.

Visualization

In [55]:
```
def uni_sns_Hist(df, xVar, bw):

    #set the plot parameters
    plt.figure(figsize=(12,8))

    #plot
    sns.histplot(data=df, x=xVar, binwidth = bw, kde=True);

    #add title and format it
    plt.title(f'''Distribution of {xVar}'''.title(), fontsize=14, weight='bold')

    #add x label and format it
    plt.xlabel(xVar.title(), fontsize=10, weight='bold')

    #add y label and format it
    plt.ylabel('Trips'.title(), fontsize=10, weight='bold')
```

In [56]:
```
uni_sns_Hist(df_age_outliers, 'member_age', 5)
# plt.figure(figsize=(12,8))
# sns.histplot(
#     data = df_age_outliers,
#     x = 'member_age'
# );
# plt.title("Age distribution for outliers");
# plt.xlabel("Age (years)");
# plt.ylabel("Number of trips");
```

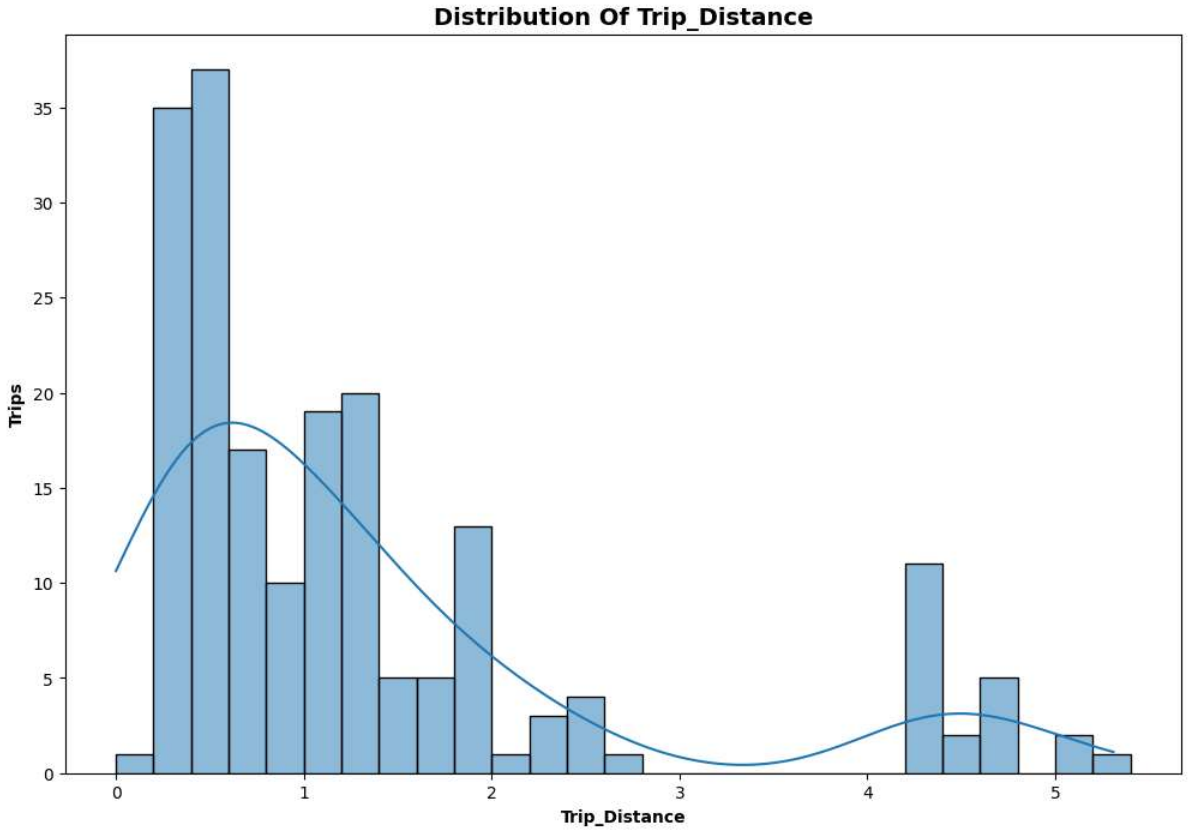


Observation

Most of the outliers are their 50s, 60s and 70s. Some people are still active at this age. However those above 80 are still weird. Lets see what distances they managed to do during trips.

Visualization

In [57]:
```
uni_sns_Hist(df_age_outliers.query('member_age > 80'), 'trip_distance', 0.2)
# plt.figure(figsize=(12,6))
# sns.histplot(
#     data = df_age_outliers.query('member_age > 80'),
#     x = 'trip_distance',
#     kde=True,
#     binwidth=0.2,
# );
# plt.title("Trip distances for Ages above 80");
# plt.xlabel("Age (years)");
# plt.ylabel("Number of trips");
```

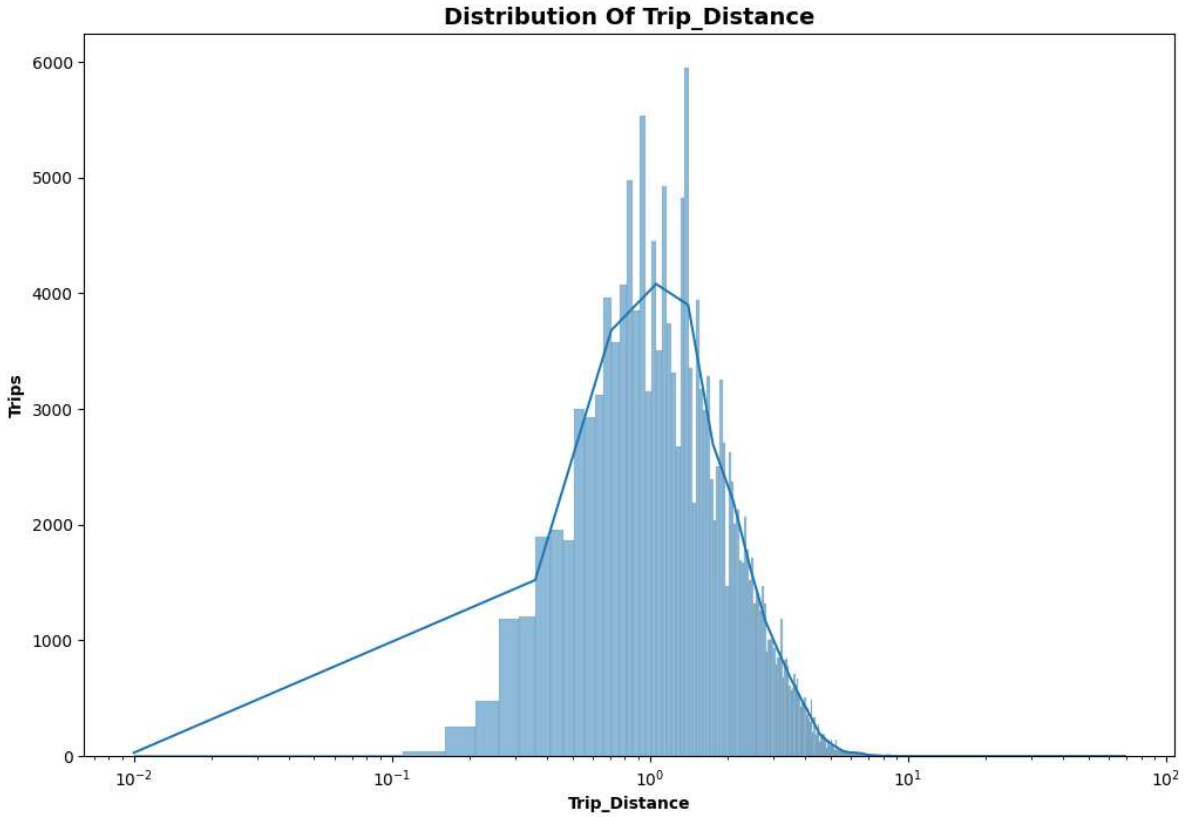


Observation

The majority of the outliers above 80 years old are riding for 600 metres and 400 metres, as we can see that the distribution is right skewed. But how does this compare to their normal-age range counterparts?

In [58]:
```
#slicing the dataframe to exclude outliers, and the zero distance anomaly
uni_sns_Hist(bike_clean.query('member_age <= 57 & trip_distance > 0'), 'trip_distar
plt.xscale('log');
# plt.figure(figsize=(12,8))
# sns.histplot(
#     data = bike_clean.query('member_age <= 57 & trip_distance > 0'),
#     x = 'trip_distance'
```

```
# );
#
```

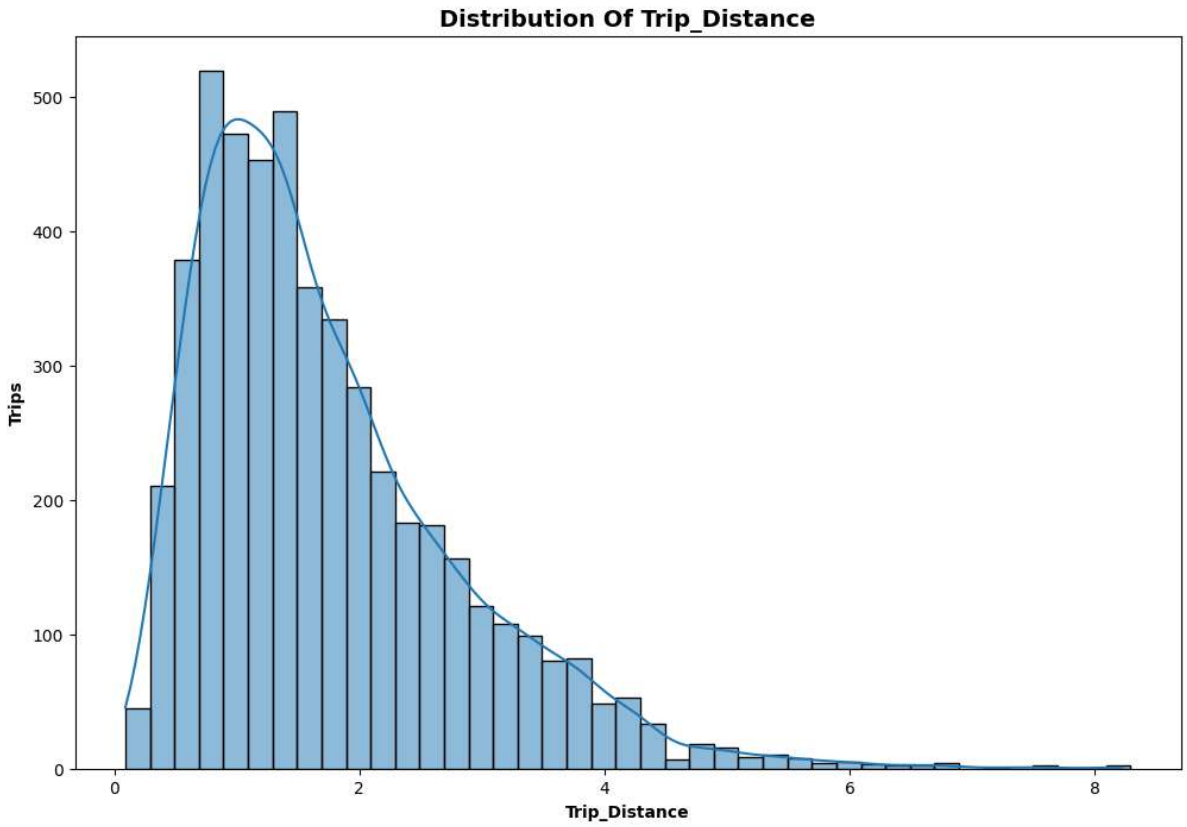


This is not really clear. Lets take a sample of about 5000 trips.

In [59]:
```
df_normal_age_sample = bike_clean.query('member_age <= 57 & trip_distance > 0').sa
```

Visualization

In [60]:
```
uni_sns_Hist(df_normal_age_sample, 'trip_distance', 0.2)

# plt.figure(figsize=(12,6))
# sns.histplot(
#     data = df_normal_age_sample,
#     x = 'trip_distance',
#     binwidth=0.2,
#     kde=True
# );
# plt.title("Trip distance distribution for normal age riders");
# plt.xlabel("Age (years)");
# plt.ylabel("Number of trips");
```



Observation

The majority of the riders ride about 1 km per trip. Not surprising considering this is a very large dataset as it shows a largely normal distribution. The spread goes across to around 4 kms.

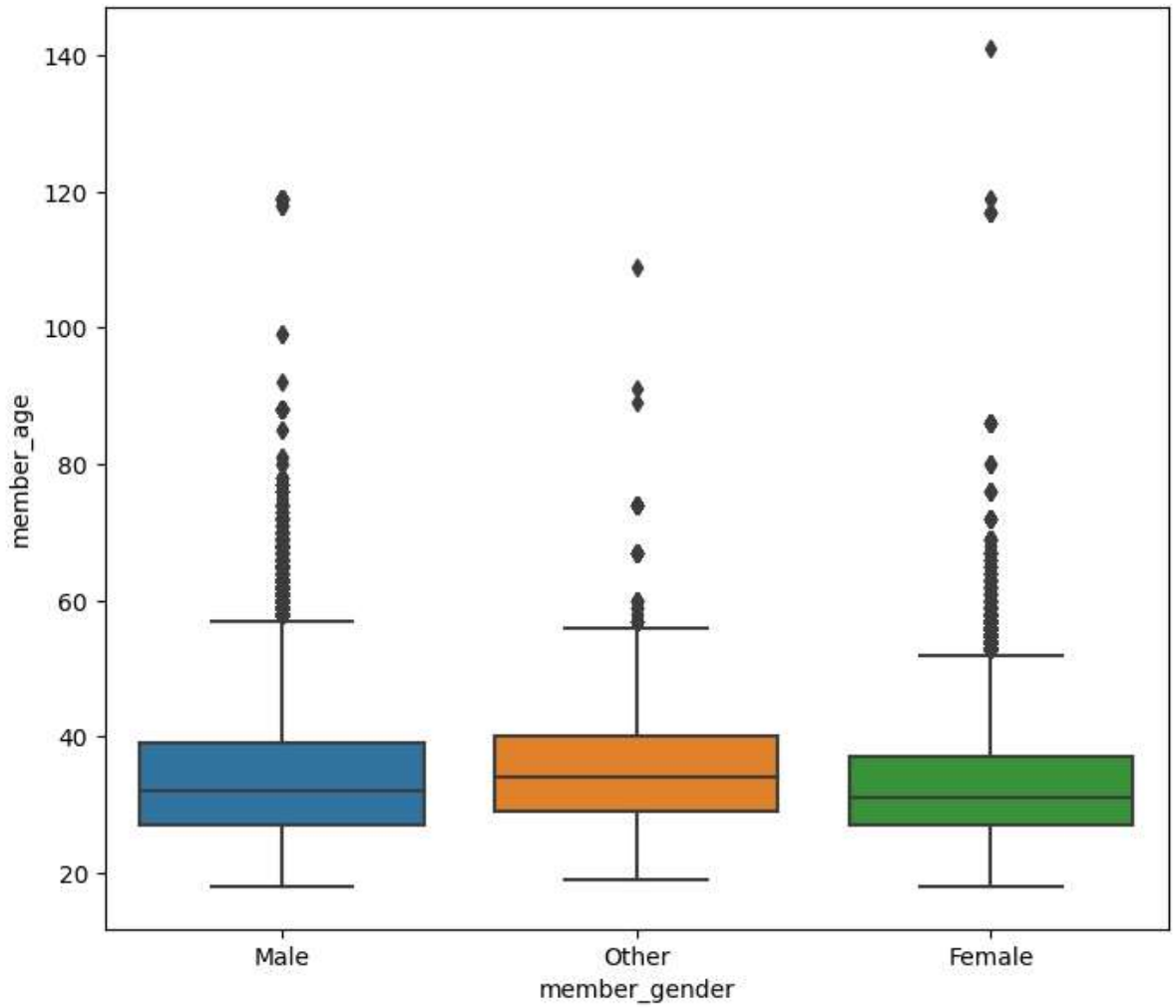
Bivariate Analysis

Question

There is a significant number of outliers in the age column, with the highest age at 141? Lets investigate the gender relations with age, to see if the outliers fall in a certain gender type.

Visualization

In [61]:
```
plt.figure(figsize=(8,7))
sns.boxplot(
    data = bike_clean,
    x = 'member_gender',
    y = 'member_age',
    orient = 'v'
);
```



Observation

The outliers are across all the gender types.

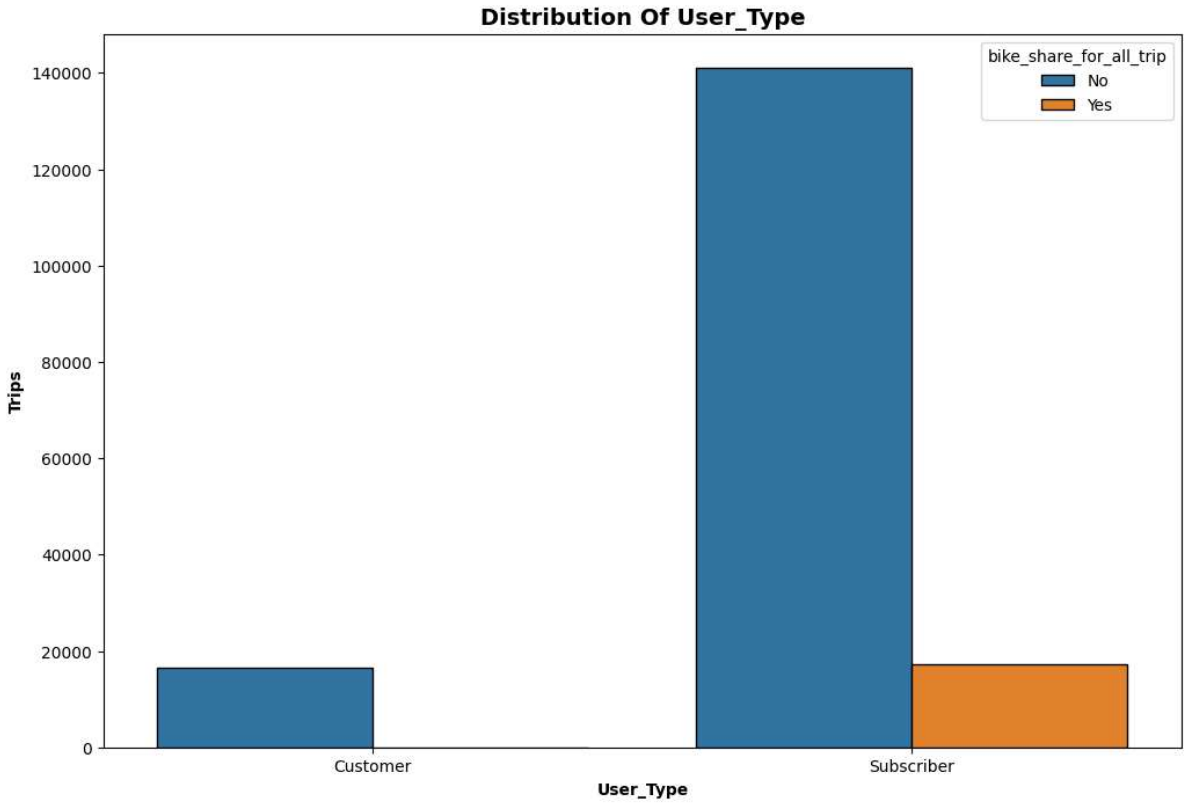
Question

But how does bike sharing relate to user_type?

In [62]:
```
def bi_CountPlot(df, xVar, hue_col):
    '''
    This function produces a seaborn countplot,
    for columns in a provided dataset by using hue
    '''

    #set the plot parameters
    plt.figure(figsize=(12,8))

    #plot
    sns.countplot(data=df, x=xVar, hue=hue_col, edgecolor='black');

    #add title and format it
    plt.title(f'''Distribution of {xVar}'''.title(), fontsize=14, weight='bold')

    #add x label and format it
    plt.xlabel(xVar.title(), fontsize=10, weight='bold')

    #add y label and format it
    plt.ylabel('Trips'.title(), fontsize=10, weight='bold')
```

Visualization

In [63]:
```
bi_CountPlot(bike_clean, 'user_type', 'bike_share_for_all_trip')
# plt.figure(figsize=(8,8))
# sns.countplot(
#     data = bike_clean,
#     x = 'user_type',
#     hue = 'bike_share_for_all_trip'
# );
# plt.title("Trip Sharing by User type");
# plt.xlabel("User type");
# plt.ylabel("Number of trips");
```



Observation

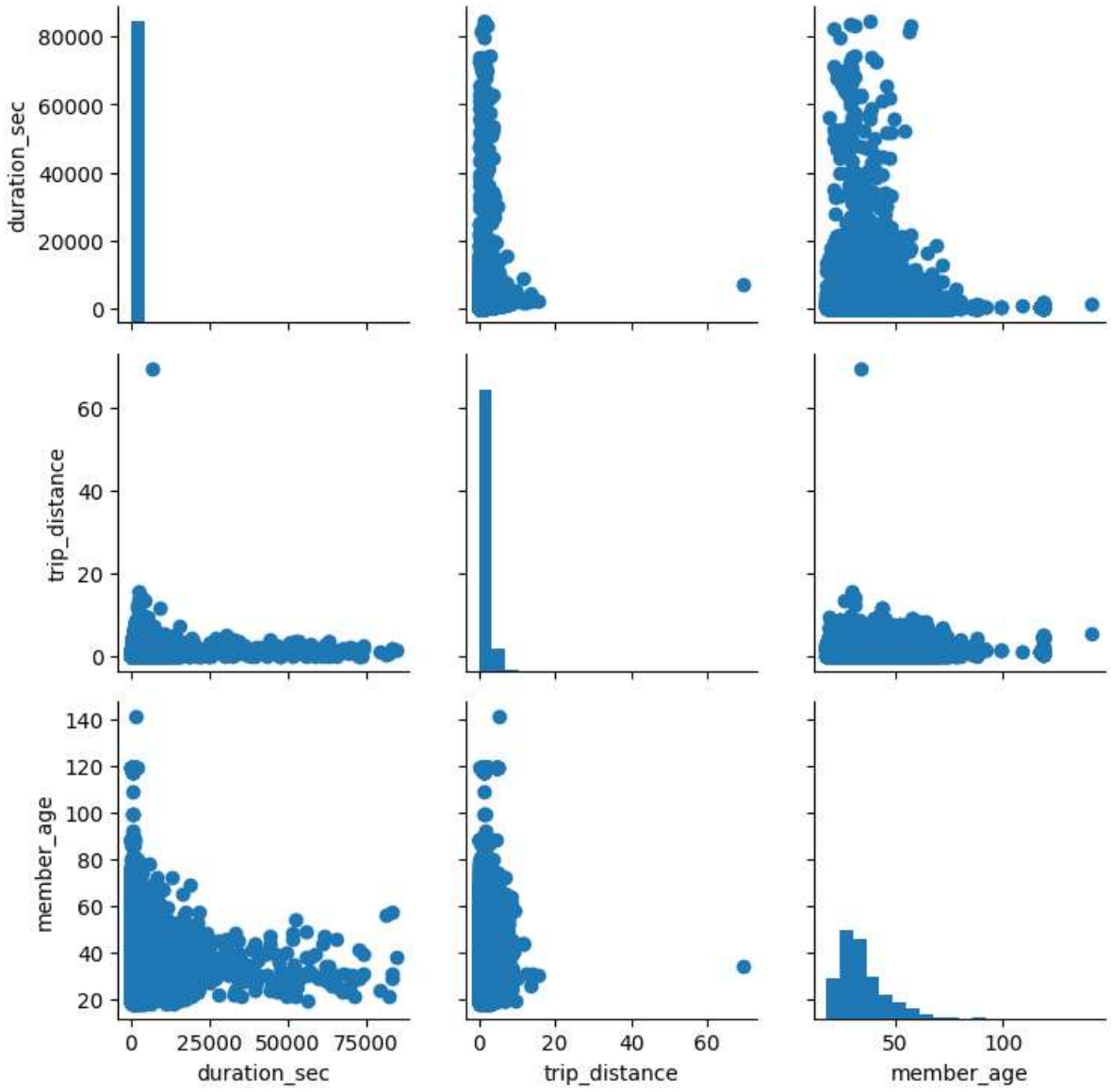
There was no sharing at all for the Customer user type. While for the Subscriber type sharing is very low.

Question

Starting with a pairgrid just to see correlations between various numeric features

Visualization

In [64]:
```
#creating a list of quantitative features
numeric_vars = ['duration_sec', 'trip_distance', 'member_age']

g = sns.PairGrid(data = bike_clean, vars = numeric_vars)
g = g.map_diag(plt.hist, bins = 20);
g.map_offdiag(plt.scatter);
```

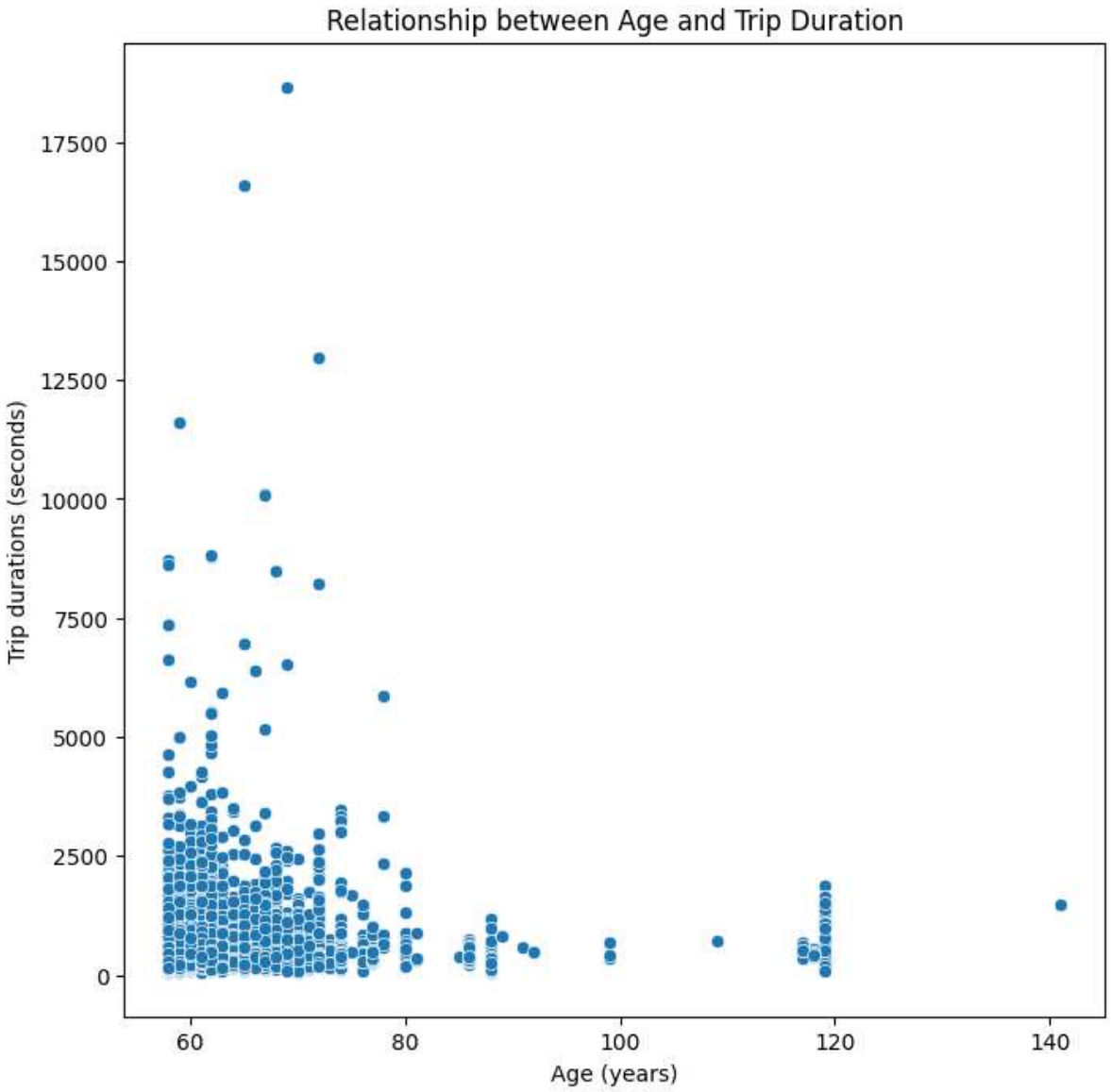


Question

There seems to be an interesting relationship between member_age and duration_sec. Lets look further into that.

Visualization

In [65]:
```
plt.figure(figsize=(8,8))
sns.scatterplot(
    data = df_age_outliers,
    x = 'member_age',
    y = 'duration_sec'
);
plt.title("Relationship between Age and Trip Duration");
plt.xlabel("Age (years)");
plt.ylabel("Trip durations (seconds)");
```

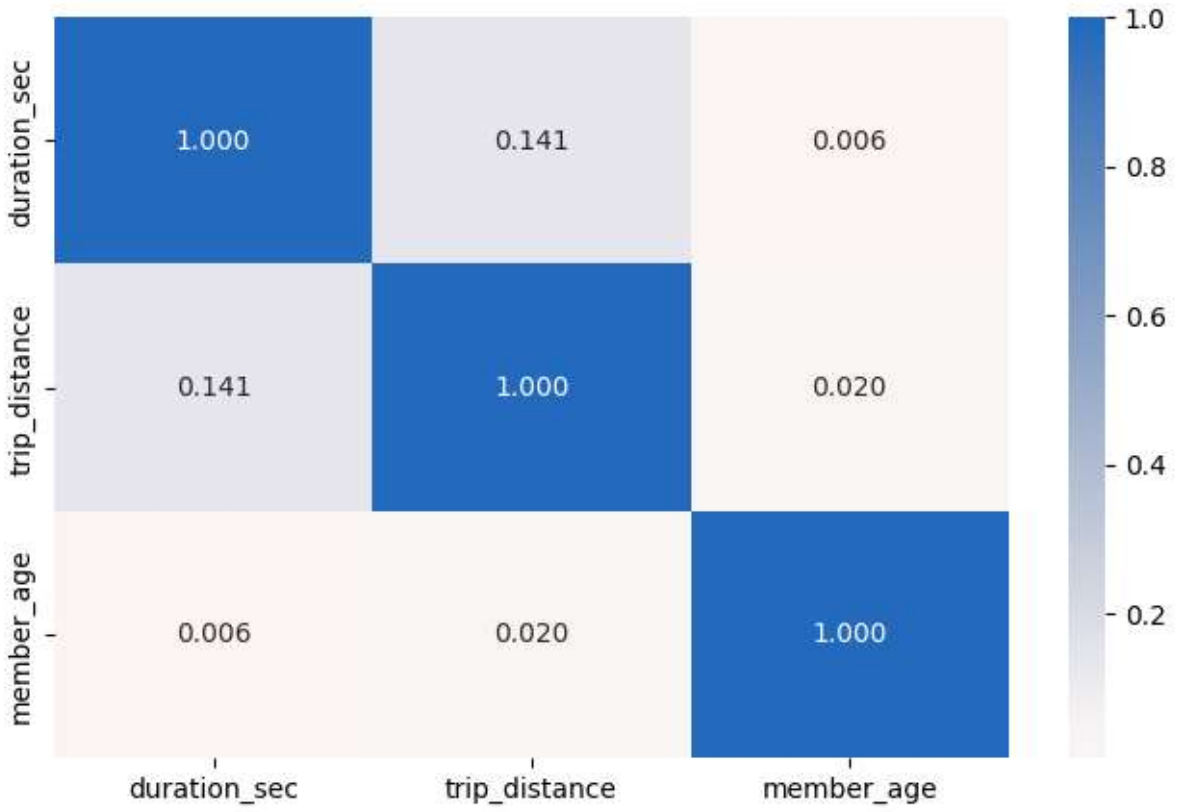


Observation

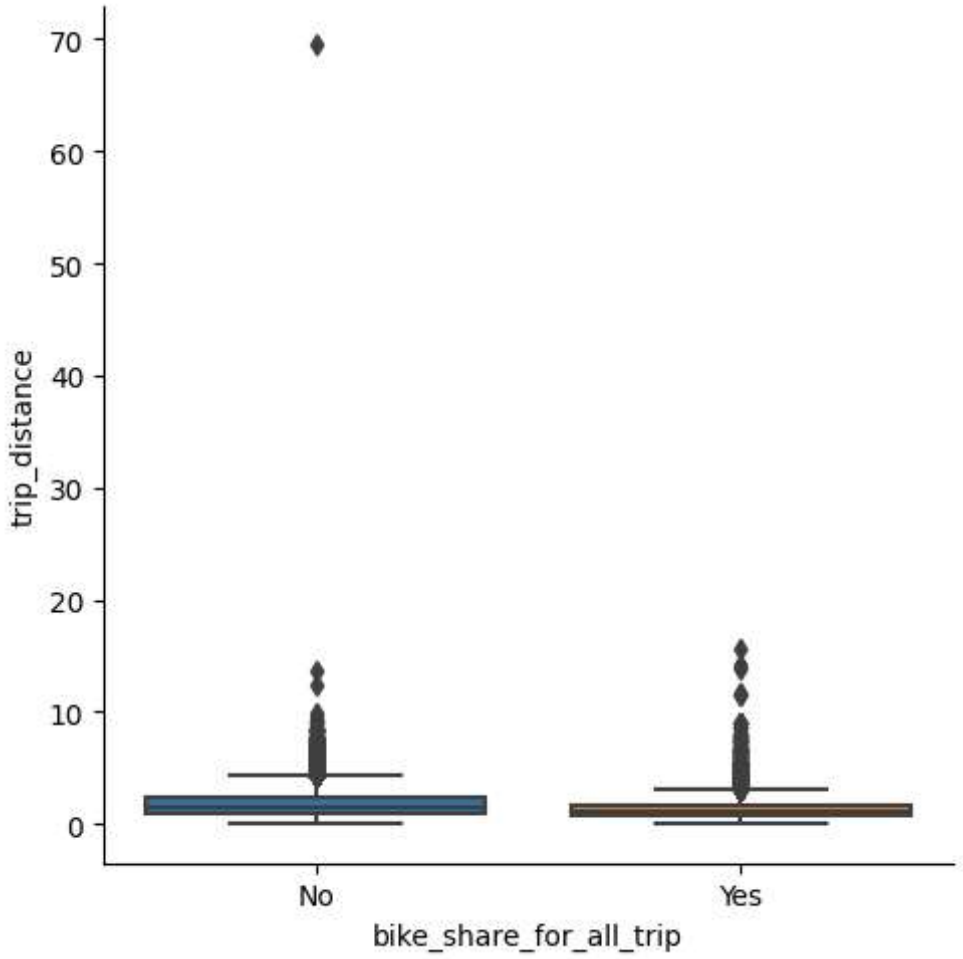
There is a very weak correlation, meaning there might be no relationship. Lets see the correlation numbers.

Visualization

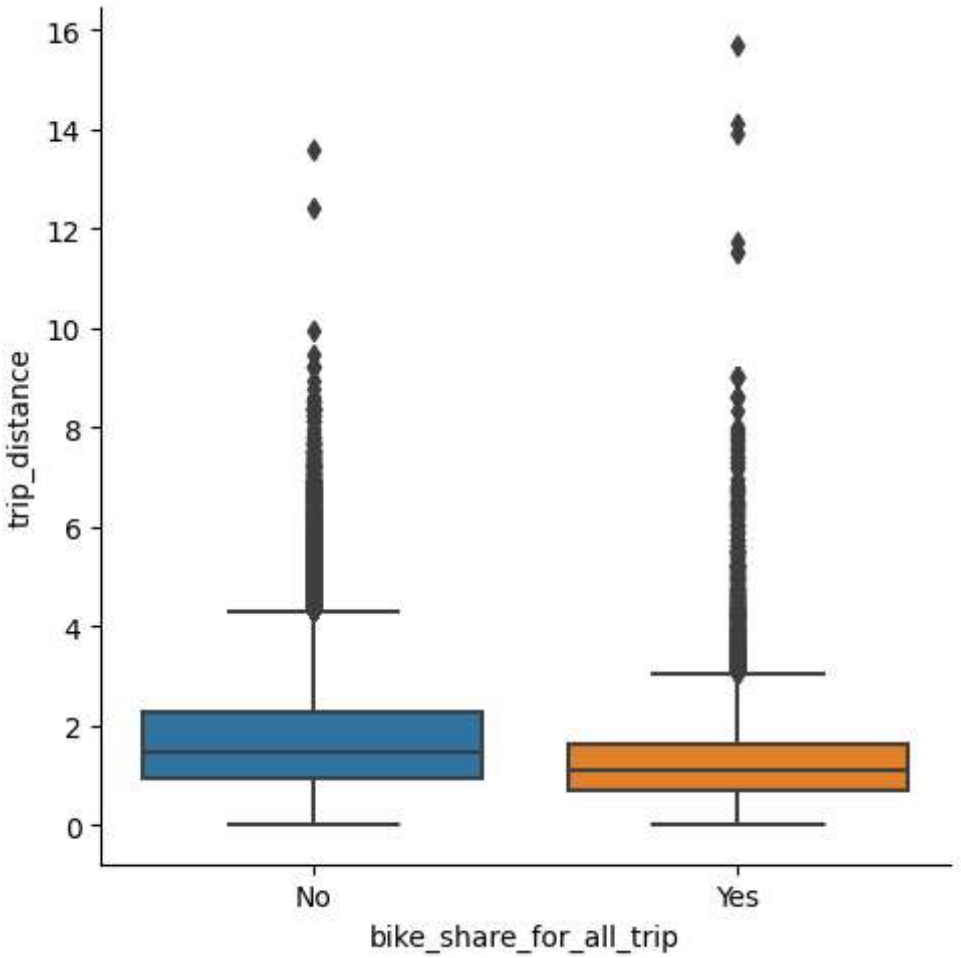
In [66]:
```
# correlation plot
plt.figure(figsize = [8, 5])
sns.heatmap(bike_clean[numeric_vars].corr(), annot = True, fmt = '.3f',
            cmap = 'vlag_r', center = 0)
plt.show();
```



In [67]:
```
sns.catplot(
    data = bike_clean,
    x = 'bike_share_for_all_trip',
    y = 'trip_distance',
    kind = 'box'
);
```



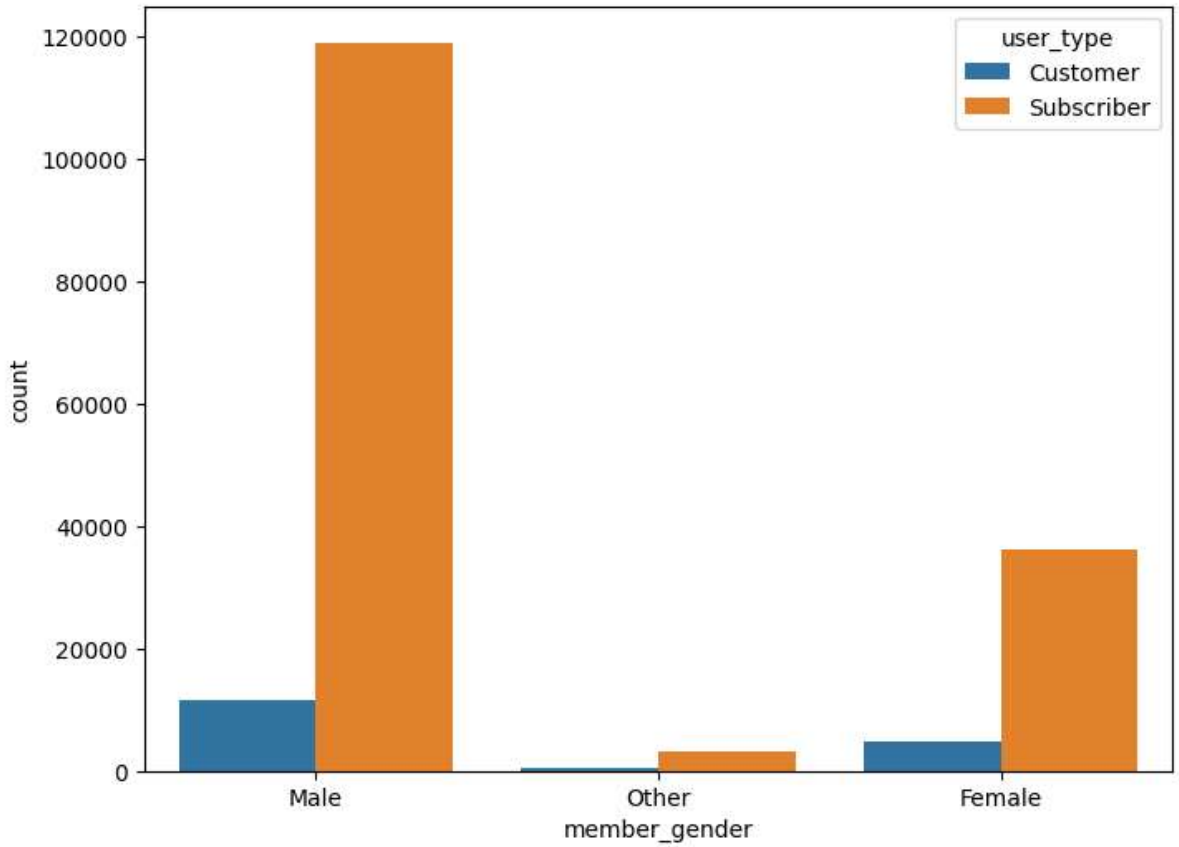
In [68]:
```
#slicing out the 50 kms trip distance
sns.catplot(
    data = bike_clean.query('trip_distance < 50'),
    x = 'bike_share_for_all_trip',
    y = 'trip_distance',
    kind = 'box'
);
```



Observation

They are outliers as well in the distance covered on trips. The median distance covered for those sharing bikes, and those who aren't is below 2 kms.

In [69]:
```
plt.figure(figsize=(8,6))
sns.countplot(
    data = bike_clean,
    x = 'member_gender',
    hue = 'user_type'
);
```

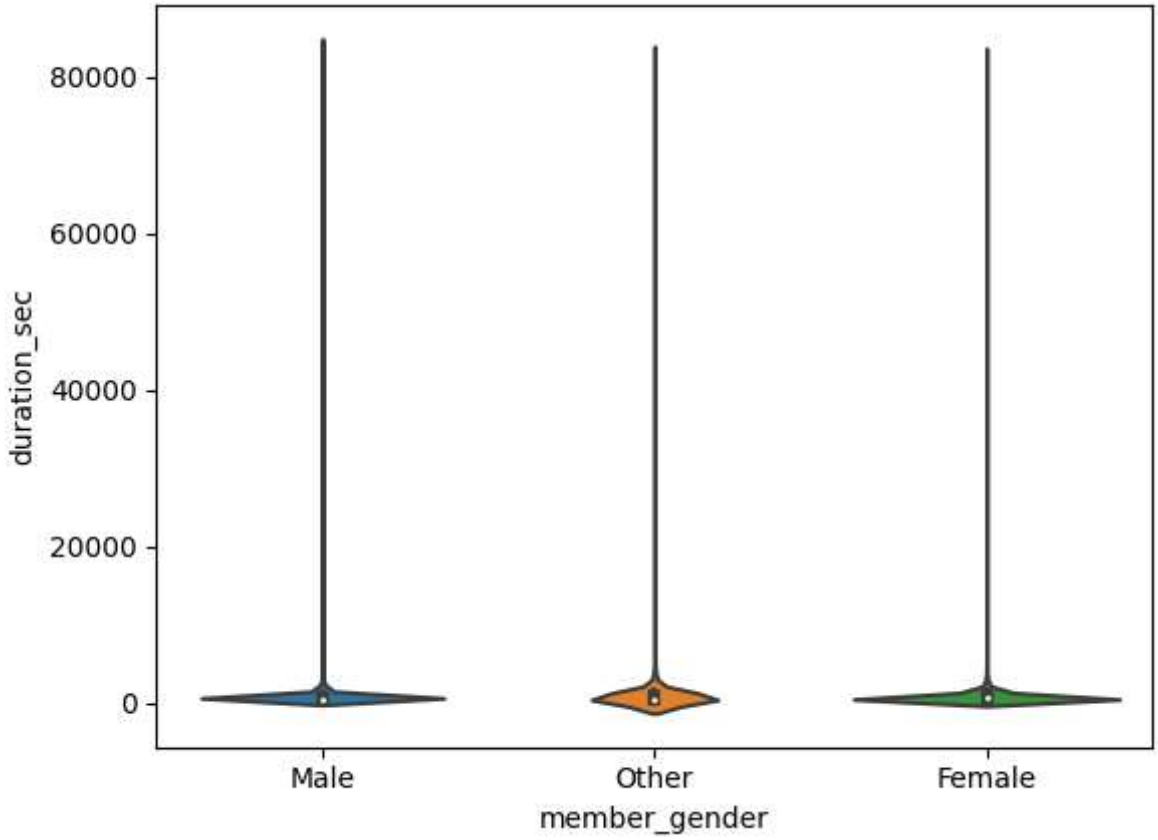


Question

Is there a relationship between gender and duration of rides?

Visualization

In [70]:
```
sns.violinplot(
    data = bike_clean,
    x = 'member_gender',
    y = 'duration_sec'
);
```



Observation

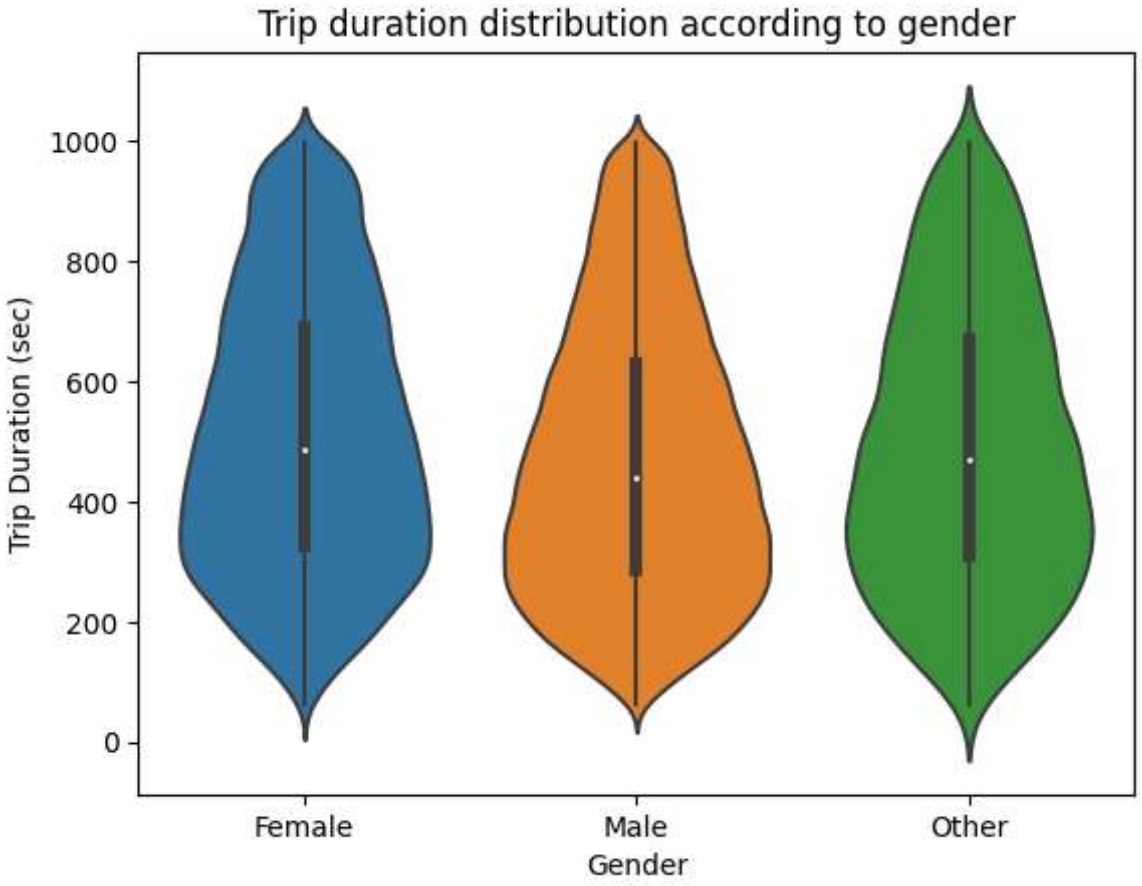
There are certainly some heavy riders compared to the rest of the crowd. Lets exclude these outliers.

In [71]:
```
bike_clean['duration_sec'].describe()
```

Out[71]:
```
count    174952.000000
mean        704.002744
std        1642.204905
min          61.000000
25%         323.000000
50%         510.000000
75%         789.000000
max       84548.000000
Name: duration_sec, dtype: float64
```

Visualization

In [72]:
```
sns.violinplot(
    data = bike_clean[bike_clean['duration_sec'] < 1000],
    x = 'member_gender',
    y = 'duration_sec'
);
plt.title("Trip duration distribution according to gender");
plt.xlabel("Gender");
plt.ylabel("Trip Duration (sec)");
```



Observation

There is not really much of a difference in the distributions above, therefore there is no visible relationship between gender and duration of trips.

Question

But lets see which gender rides more.

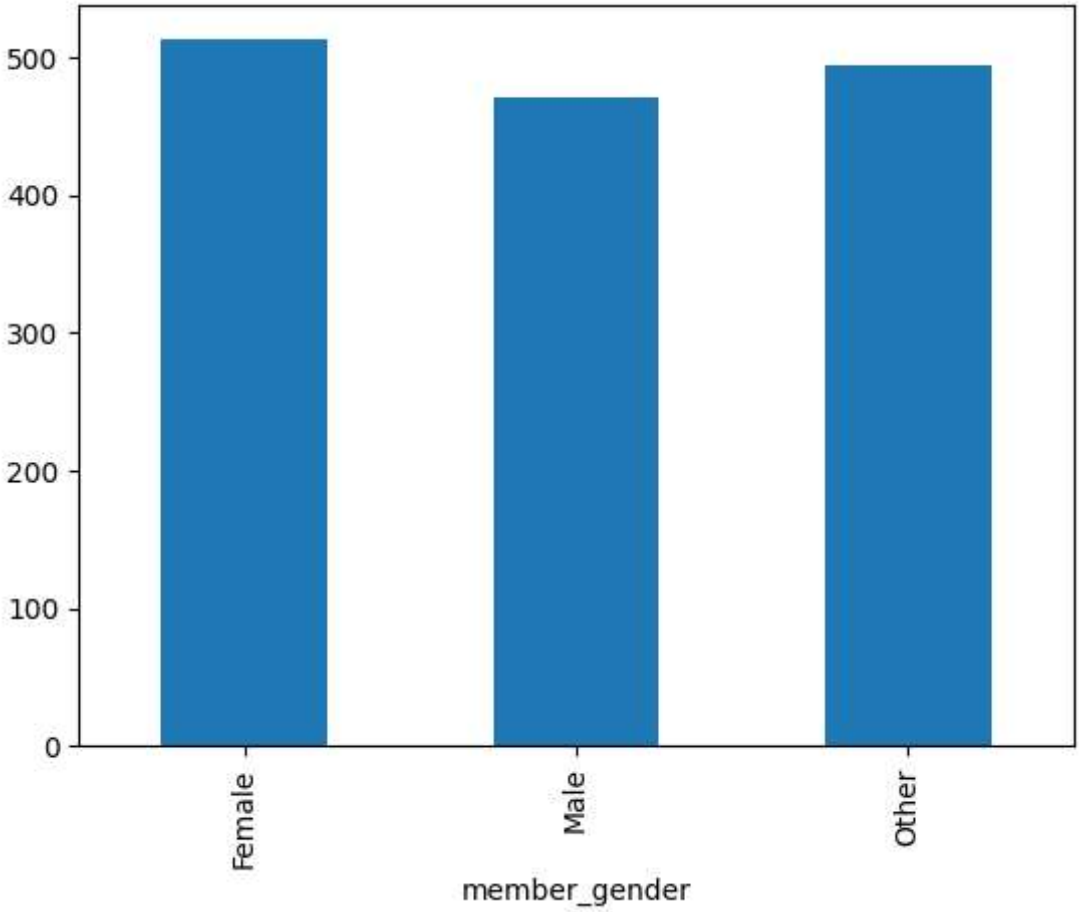
Visualization

In [73]:
```
#lets subset for riders with less than 1000 seconds in time duration
df_1000_less = bike_clean.query('duration_sec < 1000')
```

In [74]:
```
#group by gender
grouped_gender_duration = df_1000_less.groupby('member_gender')['duration_sec'].mea
grouped_gender_duration.sort_values(ascending=False)
```

Out[74]:
```
member_gender
Female    512.656786
Other     494.259259
Male      470.862369
Name: duration_sec, dtype: float64
```

In [75]:
```
grouped_gender_duration.plot(kind='bar');
```



Observation

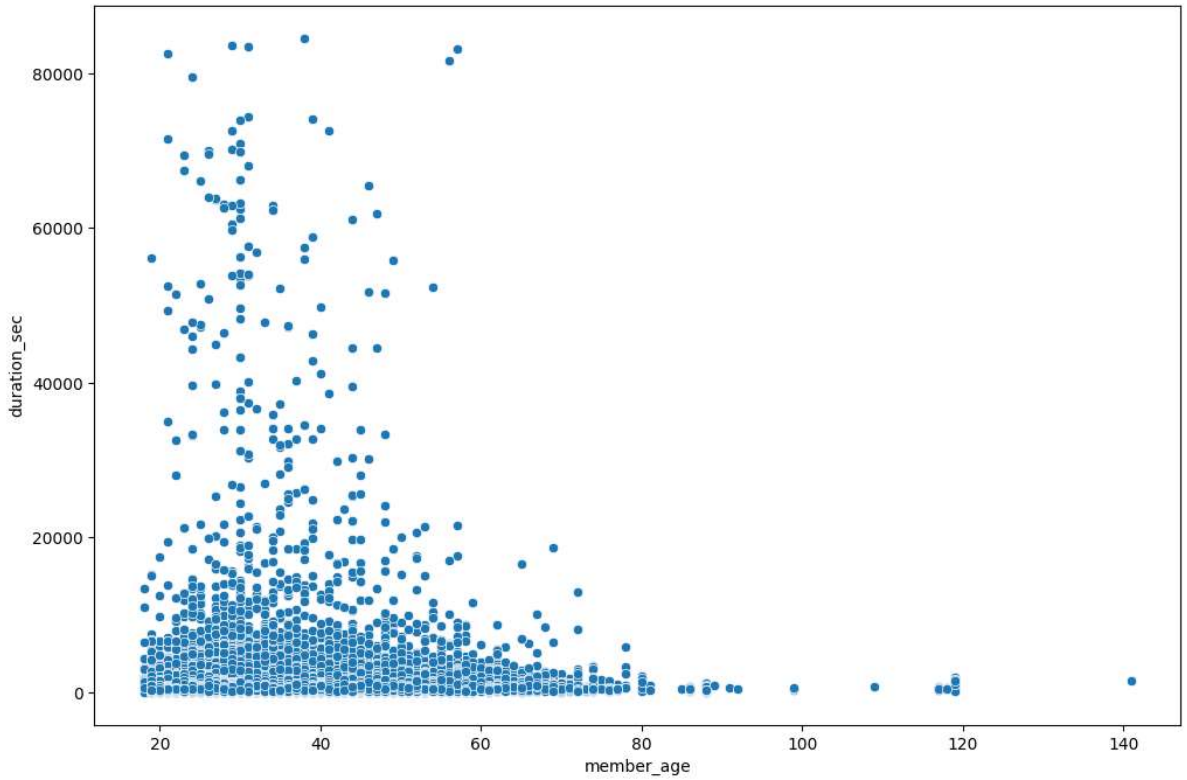
Females tend to take more time on average during their rides than the Male and Other gender types. Males take the least time.

Question

Do bike ride times decrease with age?

Visualization

In [76]:
```
plt.figure(figsize=(12,8))
sns.scatterplot(
    data = bike_clean,
    x = 'member_age',
    y = 'duration_sec'
);
```

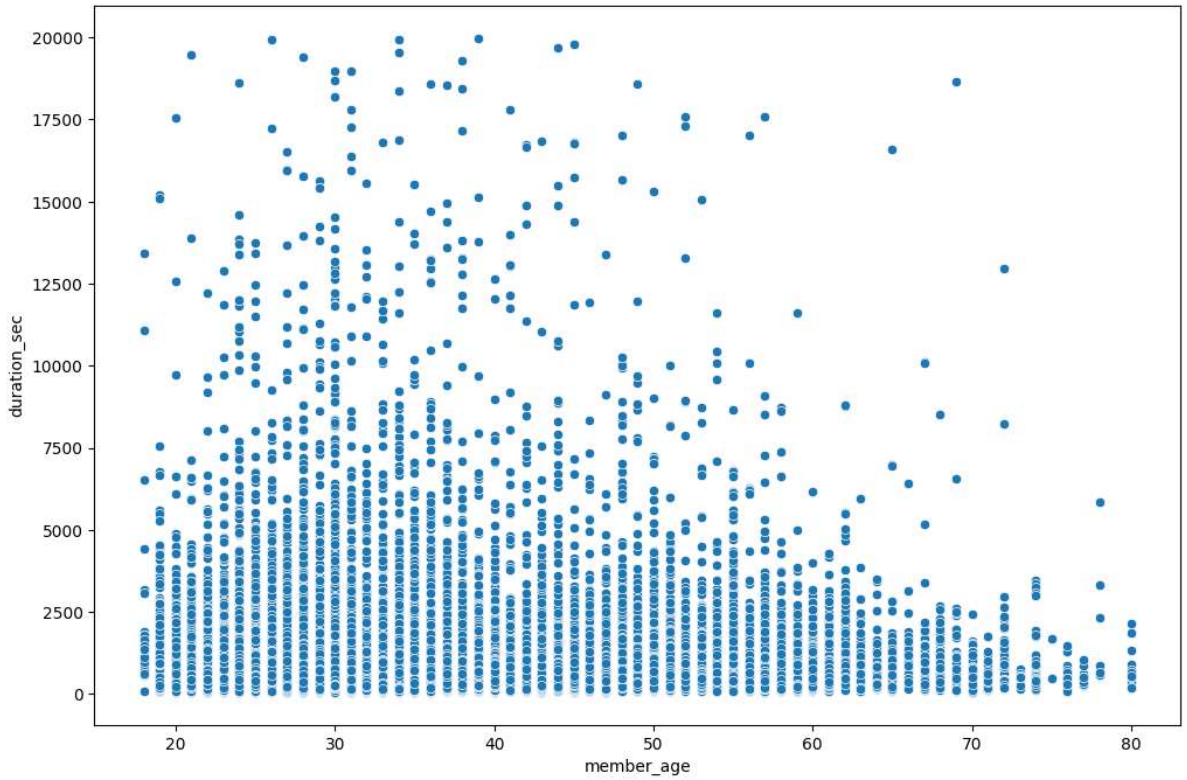


Observation

The majority of the data is for age < 80 and duration less than 20k. Lets investigate that.

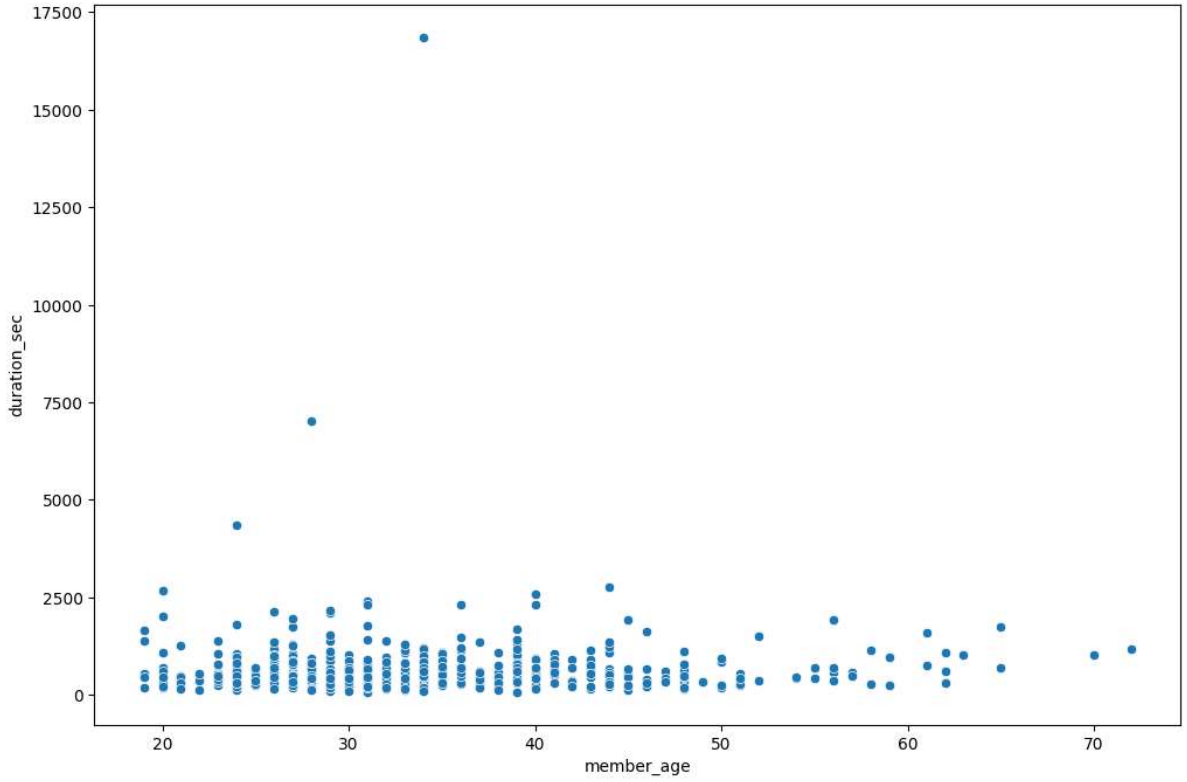
In [77]:
```
#slicing data for ages below 80 and trip durations less than 20,000 seconds.
cut_df = bike_clean.loc[(bike_clean['member_age'] <= 80) & (bike_clean['duration_se
```

In [78]:
```
plt.figure(figsize=(12,8))
sns.scatterplot(
    data = cut_df,
    x = 'member_age',
    y = 'duration_sec'
);
```



In [79]:
```
#lets get a subset of about 500 points of the cut_df
df_subset = cut_df.sample(500)
```

In [80]:
```
plt.figure(figsize=(12,8))
sns.scatterplot(
    data = df_subset,
    x = 'member_age',
    y = 'duration_sec'
);
```



Observation

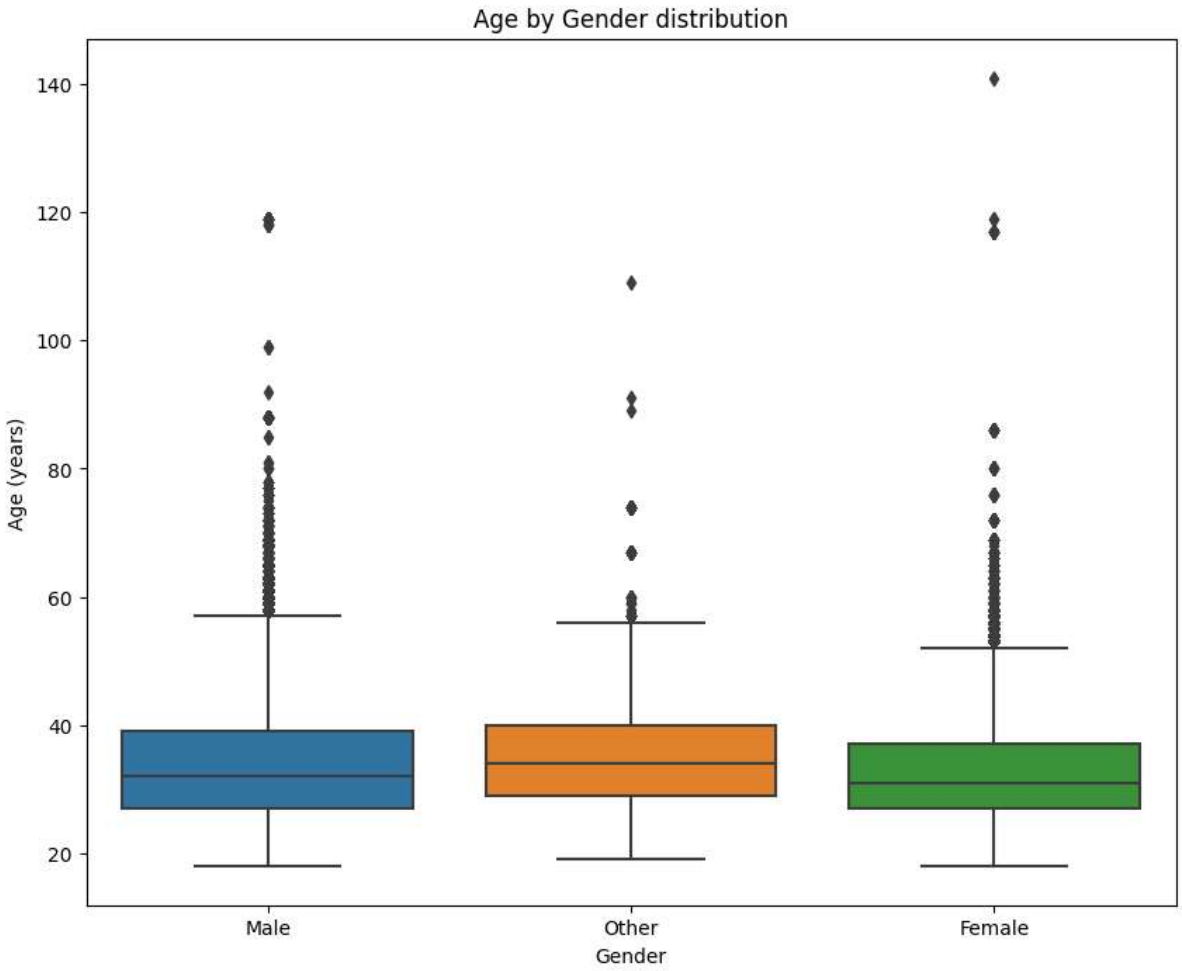
The correlation between Age and Duration of rides seems to be very weak. Therefore there is no discernible evidence that older people take longer times during their rides.

Question

Which gender has the oldest average age distribution amongst the riders?

Visualization

In [81]:
```
plt.figure(figsize=(10, 8))
sns.boxplot(
    data = bike_clean,
    x = 'member_gender',
    y = 'member_age'
);
plt.title("Age by Gender distribution");
plt.xlabel("Gender");
plt.ylabel("Age (years)");
```



Observation

The Other gender type has a slightly older age distribution than Male and Female. However, the oldest is a Female, and the Femal e and Male genders have many age outliers.

Time Series Analysis

We will leverage time series data in this dataset to do some bivariate and multivariate analysis.

In [82]:
```
#introducing number_of_bikes column for easy aggregation
bike_clean['number_of_bikes'] = 1
```

I have introduced a number_of_bikes feature, and set it to 1 per trip, so that we can aggregate on it for interesting insights

In [83]:
```
bike_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 174952 entries, 0 to 174951
Data columns (total 19 columns):
 #   Column                   Non-Null Count   Dtype         
---  ------                   --------------   -----         
 0   duration_sec             174952 non-null  int64         
 1   start_time               174952 non-null  datetime64[ns]
 2   end_time                 174952 non-null  datetime64[ns]
 3   start_station_id         174952 non-null  int32         
 4   start_station_name       174952 non-null  object        
 5   start_station_latitude   174952 non-null  float64       
 6   start_station_longitude  174952 non-null  float64       
 7   end_station_id           174952 non-null  int32         
 8   end_station_name         174952 non-null  object        
 9   end_station_latitude     174952 non-null  float64       
 10  end_station_longitude    174952 non-null  float64       
 11  bike_id                  174952 non-null  int64         
 12  user_type                174952 non-null  object        
 13  member_gender            174952 non-null  object        
 14  bike_share_for_all_trip  174952 non-null  object        
 15  start_trip_date          174952 non-null  object        
 16  member_age               174952 non-null  int32         
 17  trip_distance            174952 non-null  float64       
 18  number_of_bikes          174952 non-null  int64         
dtypes: datetime64[ns](2), float64(5), int32(3), int64(3), object(6)
memory usage: 23.4+ MB
```

In [84]:
```
day_1 = bike_clean['start_trip_date'][0]
day_1.year, day_1.month, day_1.day
```

Out[84]:
```
(2019, 2, 28)
```

In [85]:
```
bike_clean['start_trip_date'][0]
```

Out[85]:
```
datetime.date(2019, 2, 28)
```

In [86]:
```
bike_clean['start_trip_date'].min()
```

Out[86]:
```
datetime.date(2019, 2, 1)
```

In [87]:
```
bike_clean['start_trip_date'].max()
```

Out[87]:
```
datetime.date(2019, 2, 28)
```

In [88]:
```
bike_clean['start_time'].duplicated().sum()
```

Out[88]:
```
11
```

Introducing a new dataframe to observe some aggregated features

In [89]:
```
time_df = bike_clean.groupby([pd.Grouper(key='start_time', freq='D')]).agg(unique_b
         total_bikes = ('number_of_bikes','sum'),
         total_trip_time = ('duration_sec','sum'),
         total_trip_distance = ('trip_distance', 'sum'))
```

In [90]:
```
time_df.info()
```

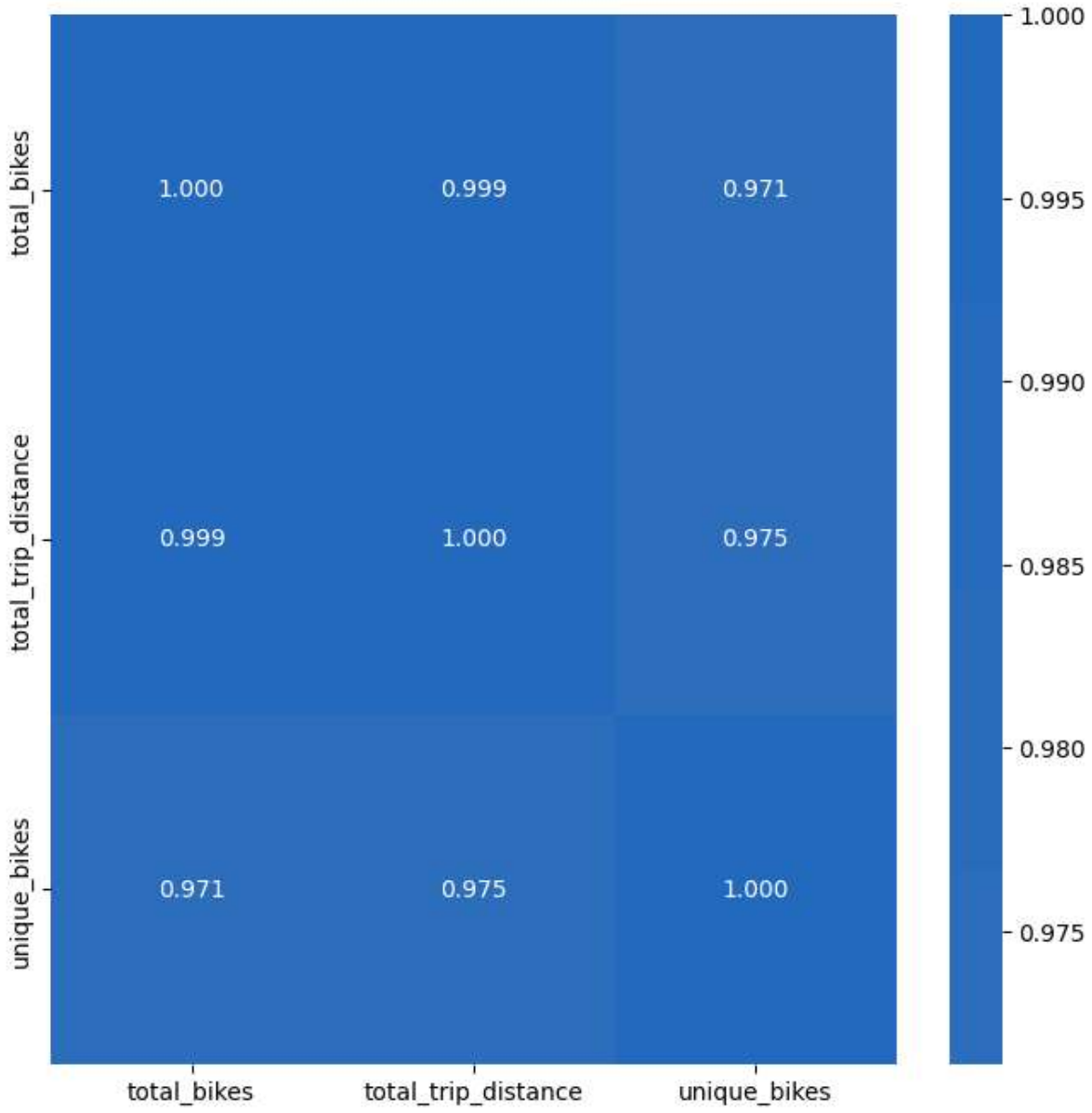
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 28 entries, 2019-02-01 to 2019-02-28
Freq: D
Data columns (total 4 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   unique_bikes         28 non-null     int64  
 1   total_bikes          28 non-null     int64  
 2   total_trip_time      28 non-null     int64  
 3   total_trip_distance  28 non-null     float64
dtypes: float64(1), int64(3)
memory usage: 1.1 KB
```

Question

Lets see how the numeric features in the time_df correlate.

Visualization

In [91]:
```
# correlation plot
n_vars = ['total_bikes', 'total_trip_distance', 'unique_bikes']
plt.figure(figsize = [8, 8])
sns.heatmap(time_df[n_vars].corr(), annot = True, fmt = '.3f',
           cmap = 'vlag_r', center = 0)
plt.show()
```



Observation

The correlation heatmap stamps the observation that as bike rentals increase, the number of unique bikes rented also increases and so does the covered trip distance.

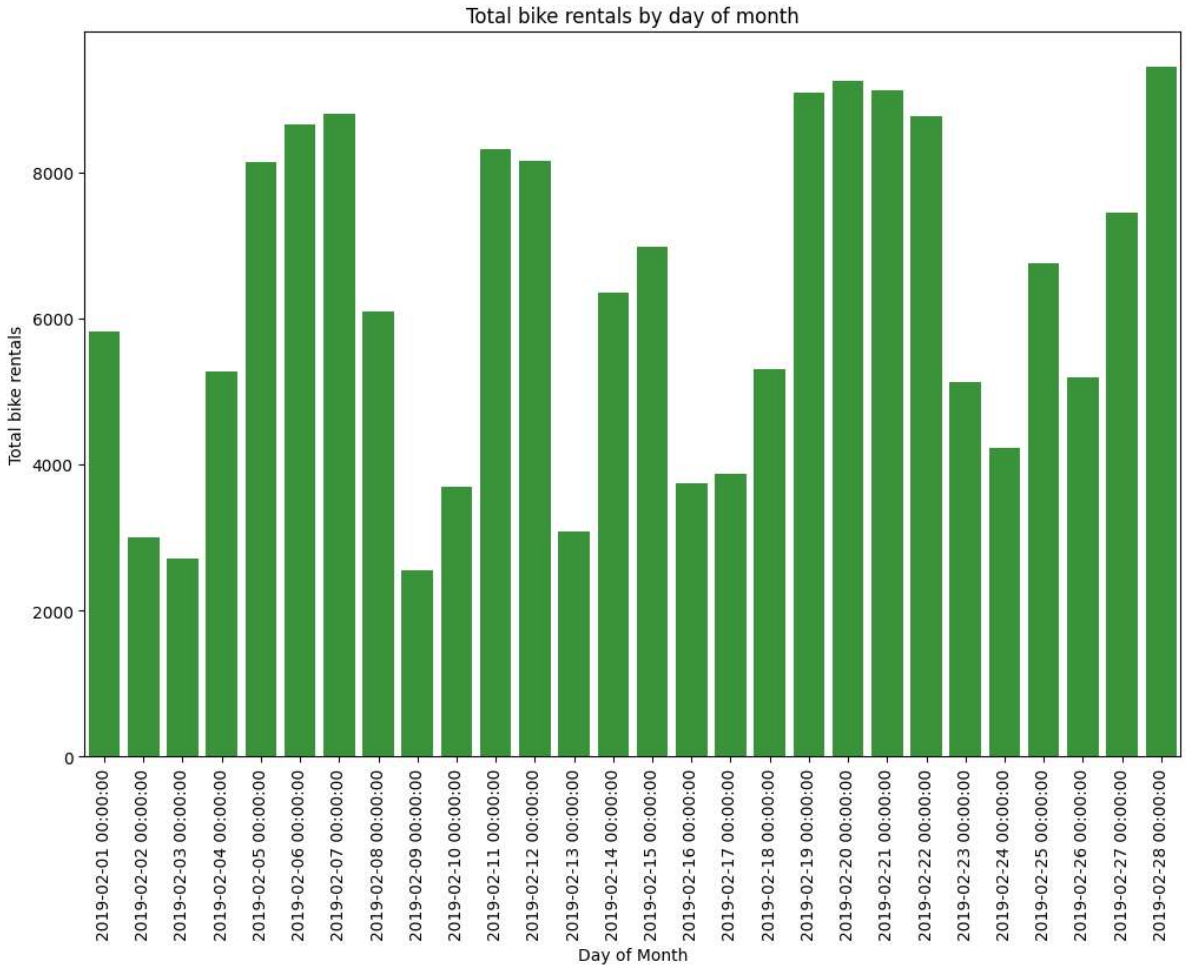
Question

Which period of the month sees more bike rentals?

Visualization

In [92]:
```
plt.figure(figsize=(12,8))
g_color = sns.color_palette()[2]
sns.barplot(
    data = time_df,
    x = time_df.index,
    y = 'total_bikes',
    color = g_color,
);
plt.title("Total bike rentals by day of month")
plt.ylabel("Total bike rentals")
```

```
plt.xlabel("Day of Month")
plt.xticks(rotation=90);
```



Observation

There is a pattern which seems to suggest that some days of the week see more traffic than others. We will investigate this further later on.

Question

Now back to our question, is there more traffic during certain days of the week ?

Visualization

In [93]:
```
# extracting a series of days of the week per row in our dataset
s = pd.date_range(bike_clean['start_time'].min(), bike_clean['start_time'].max(),
weekdays_ser = s.dt.dayofweek

# creating a dictionary to match days in the series above which come as numbers
weekday_dict = {
    0 : 'Monday',
    1 : 'Tuesday',
    2 : 'Wednesday',
    3 : 'Thursday',
    4 : 'Friday',
    5 : 'Saturday',
    6 : 'Sunday'
}
```

```
# matching the series day numbers to actual days of the week
weekdays_list = []
for entry in weekdays_ser:
    if entry in weekday_dict.keys():
        weekdays_list.append(weekday_dict[entry])

#appending the weekdays to the dataframe we created earlier
time_df['weekday'] = weekdays_list


# grouping our time series dataframe by week of the day
week_day_df = time_df.groupby('weekday').agg(
    total_rentals = ('total_bikes', 'sum')
    )
```

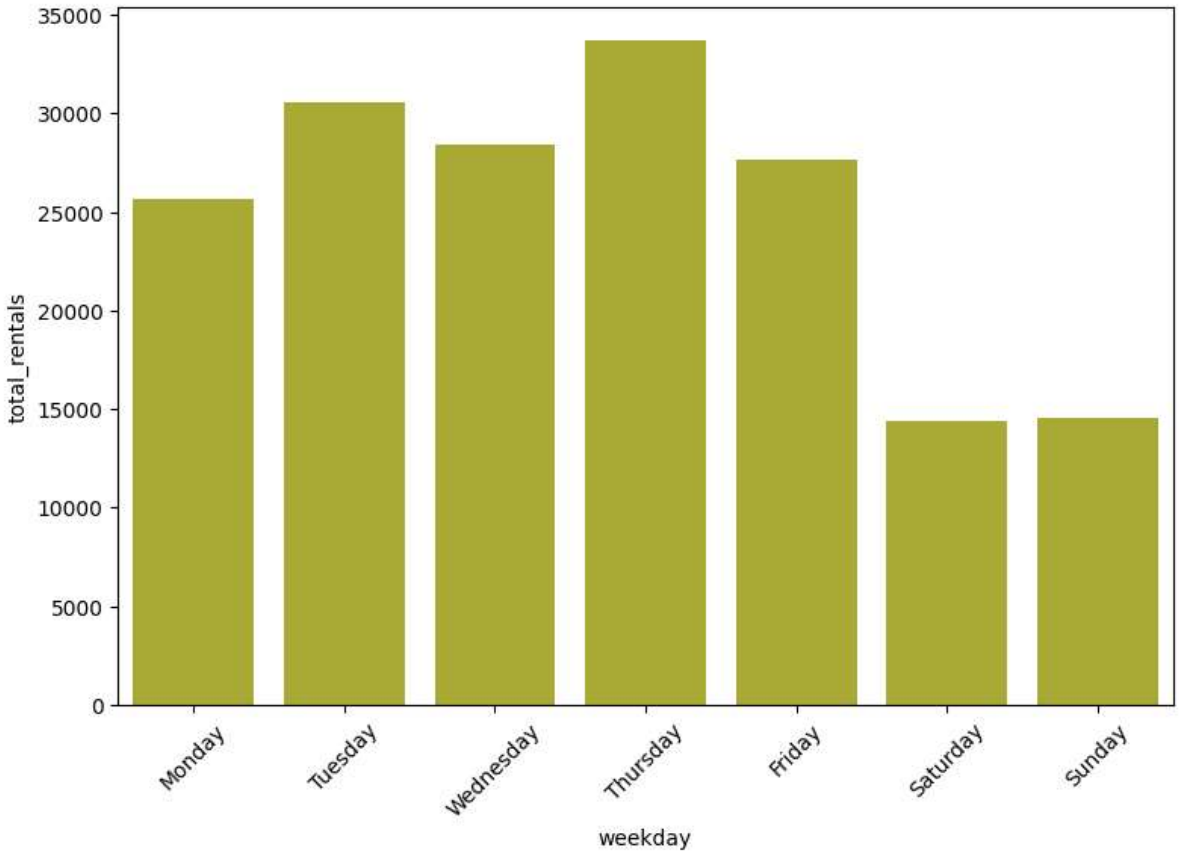
In [94]:
```
week_day_df.reset_index(inplace=True)
week_day_df
```

Out[94]:

	weekday	total_rentals
0	Friday	27663
1	Monday	25641
2	Saturday	14414
3	Sunday	14512
4	Thursday	33712
5	Tuesday	30584
6	Wednesday	28426

In [95]:
```
#converting weekdays into an ordered categorical dtype
weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday

w_classes = pd.api.types.CategoricalDtype(ordered=True, categories=weekday_order)

week_day_df['weekday'] = week_day_df['weekday'].astype(w_classes)
```

In [96]:
```
color = sns.color_palette()[8]
plt.figure(figsize=(9,6))
sns.barplot(
    data = week_day_df,
    x = 'weekday',
    y = 'total_rentals',
    color = color
);
plt.xticks(rotation = 45);
```



Observation

Most trips occur during weekdays, whilst the least trips are taken on weekends (Saturdays and Sundays)

Question

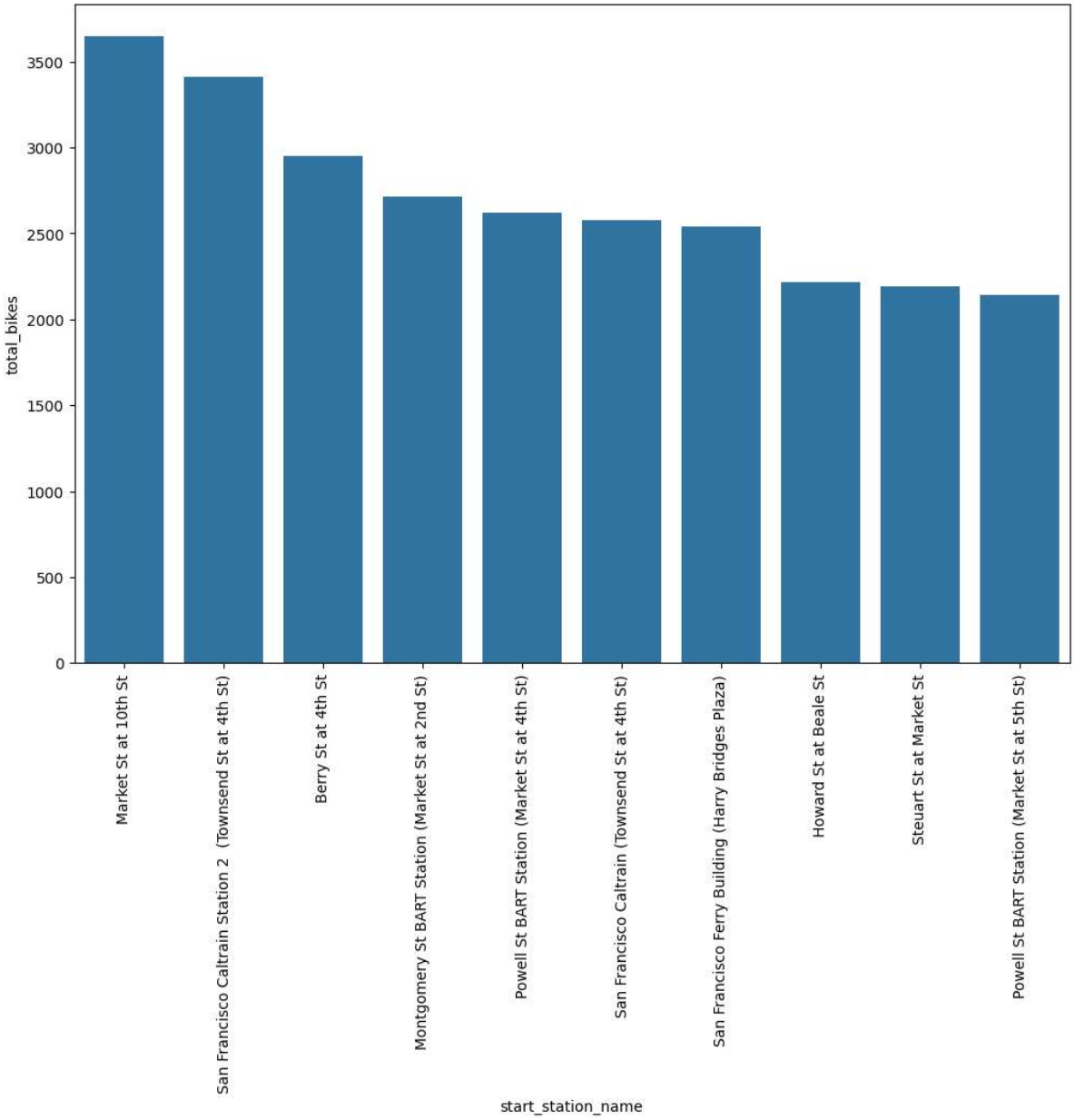
So which stations are most popular with bikers?

Visualization

In [97]:
```
#lets create a group by dataframe for geo data
# grouping data and named aggregation on bike_id, number_of_bikes, and duration_sec
geo_start_df = bike_clean.groupby([pd.Grouper(key='start_station_name'), 'start_sta
         total_bikes=('number_of_bikes','sum'),
         total_trip_time=('duration_sec','sum'))

geo_start_df.reset_index(inplace=True)
```

In [98]:
```
top_10_geo_df = geo_start_df.sort_values(by=['total_bikes'], ascending=False).head
```

In [99]:
```
plt.figure(figsize=(12,8))
color = sns.color_palette()[0]
sns.barplot(
    data = top_10_geo_df,
    x = 'start_station_name',
    y = 'total_bikes',
    color = color
);
plt.xticks(rotation=90);
```



Observation

From the top 10 stations by bike rentals, 5 including the station with the highest rentals, "Market St at 10th St"

Multivariate Analysis

Question

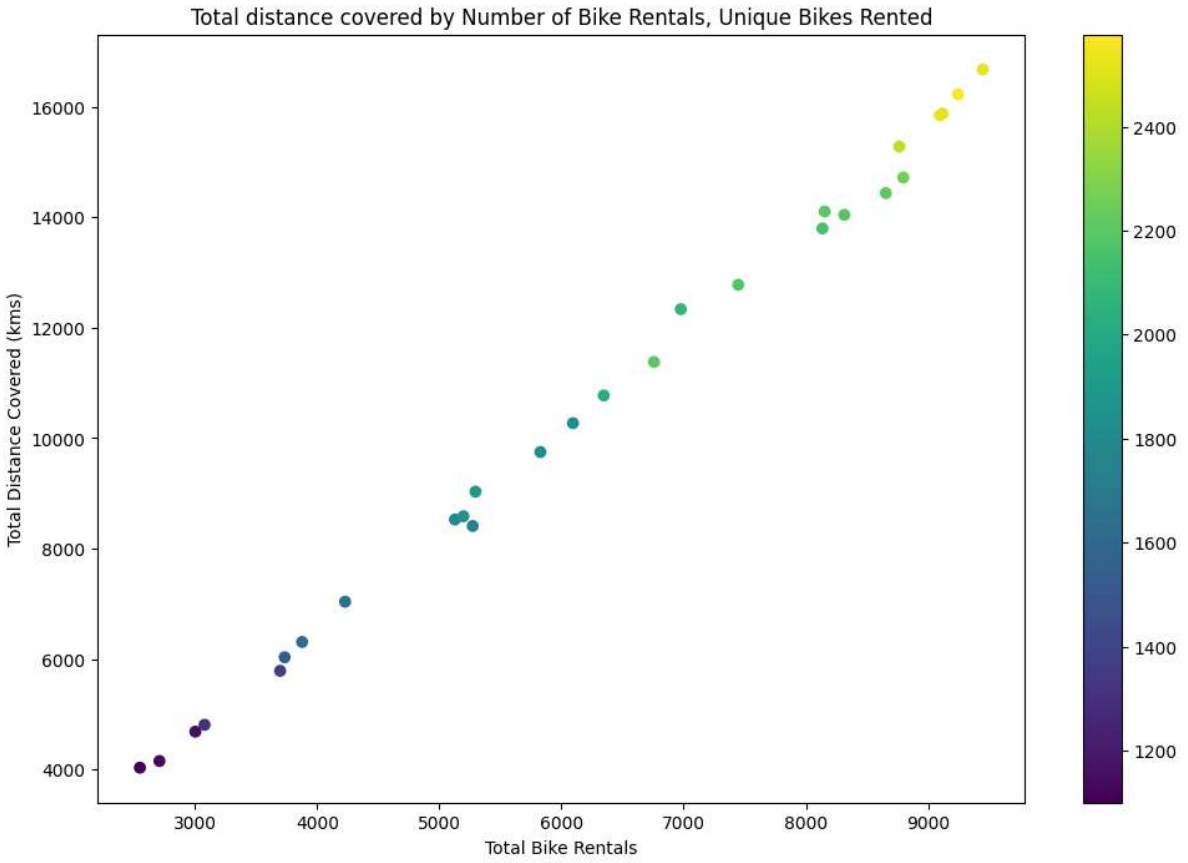
When more bikes are rented, does that result in more trips taken and more distance covered on trips?

Visualization

In [100…

```
plt.figure(figsize=(12,8))
plt.scatter(
    data = time_df,
    x = 'total_bikes',
```

```
    y = 'total_trip_distance',
    c = 'unique_bikes'
);
plt.title('Total distance covered by Number of Bike Rentals, Unique Bikes Rented')
plt.xlabel('Total Bike Rentals')
plt.ylabel('Total Distance Covered (kms)')
plt.colorbar();
```



Observation

As expected, the correlation between the 3 features is very positive .Lets confirm this with the actual correlation numbers.

Question

Do rentals between user types vary by week of the month?

Visualization

In [101…

```
bike_clean.groupby([pd.Grouper(key='start_time', freq='W'), 'user_type']).number_o
```

Out[101]:
```
start_time  user_type
2019-02-03  Customer       1158
            Subscriber    10381
2019-02-10  Customer       3506
            Subscriber    39696
2019-02-17  Customer       3927
            Subscriber    36555
2019-02-24  Customer       5791
            Subscriber    45089
2019-03-03  Customer       2184
            Subscriber    26665
Name: number_of_bikes, dtype: int64
```

In [102…
```
# grouping data and named aggregation on bike_id, number_of_bikes, and duration_sec
weekly_df = bike_clean.groupby([pd.Grouper(key='start_time', freq='W'), 'user_type
         total_bikes=('number_of_bikes','sum'),
         total_trip_time=('duration_sec','sum'))
```

In [103…
```
weekly_df.head()
```

Out[103]:

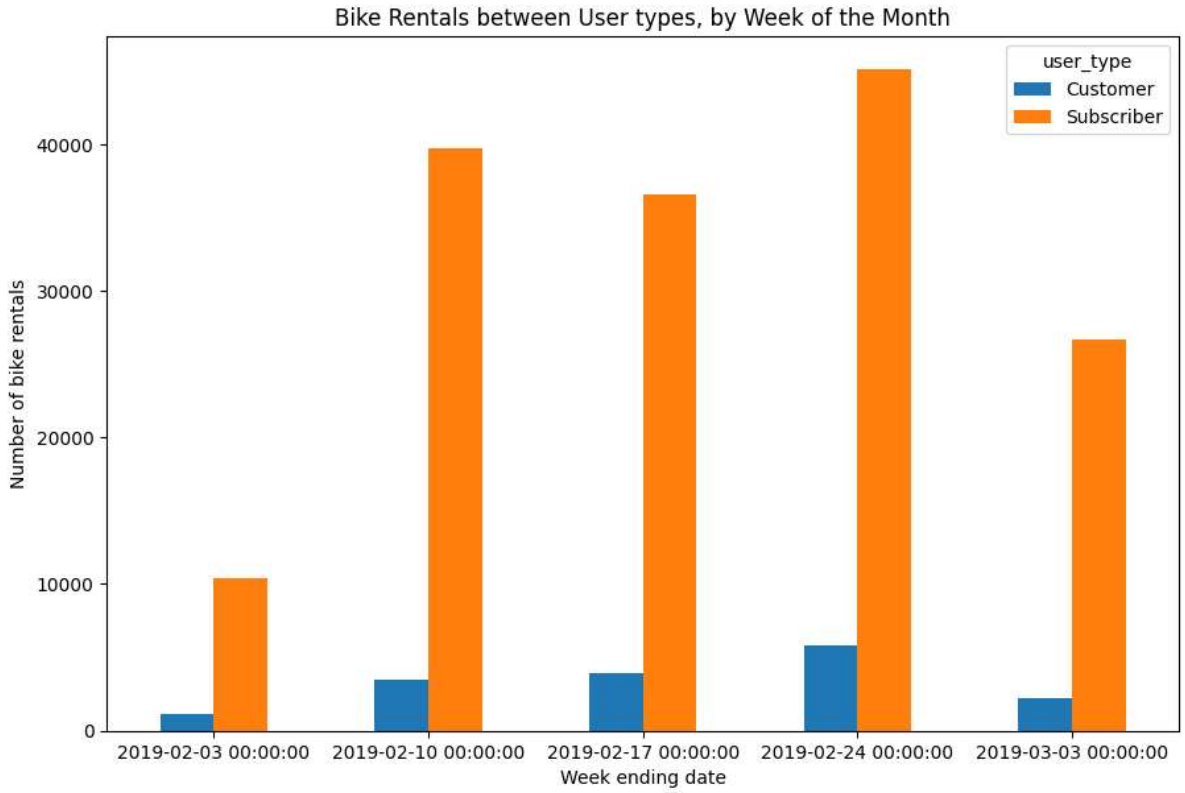
		unique_bikes	total_bikes	total_trip_time
start_time	**user_type**			
2019-02-03	**Customer**	791	1158	1418293
	Subscriber	2234	10381	6593668
2019-02-10	**Customer**	1678	3506	4963089
	Subscriber	3070	39696	24606266
2019-02-17	**Customer**	1944	3927	5105009

In [104…
```
weekly_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 10 entries, (Timestamp('2019-02-03 00:00:00', freq='W-SUN'), 'Custome
r') to (Timestamp('2019-03-03 00:00:00', freq='W-SUN'), 'Subscriber')
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   unique_bikes     10 non-null     int64
 1   total_bikes      10 non-null     int64
 2   total_trip_time  10 non-null     int64
dtypes: int64(3)
memory usage: 433.0+ bytes
```

In [105…
```
ax = weekly_df.unstack(level=1)
```

In [106…
```
ax.plot(y = 'total_bikes', kind='bar', rot=0, figsize=(9, 6))
plt.tight_layout()
plt.title("Bike Rentals between User types, by Week of the Month");
plt.xlabel("Week ending date");
plt.ylabel("Number of bike rentals");
```



Observation

Bike rentals are higher during the middle of the month, than the start/end of the month for both user types.

Question

Which locations are the most popular? We are going to investigate this using longitude and latitude data.

In [107…

```
bike_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 174952 entries, 0 to 174951
Data columns (total 19 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   duration_sec             174952 non-null  int64
 1   start_time               174952 non-null  datetime64[ns]
 2   end_time                 174952 non-null  datetime64[ns]
 3   start_station_id         174952 non-null  int32
 4   start_station_name       174952 non-null  object
 5   start_station_latitude   174952 non-null  float64
 6   start_station_longitude  174952 non-null  float64
 7   end_station_id           174952 non-null  int32
 8   end_station_name         174952 non-null  object
 9   end_station_latitude     174952 non-null  float64
 10  end_station_longitude    174952 non-null  float64
 11  bike_id                  174952 non-null  int64
 12  user_type                174952 non-null  object
 13  member_gender            174952 non-null  object
 14  bike_share_for_all_trip  174952 non-null  object
 15  start_trip_date          174952 non-null  object
 16  member_age               174952 non-null  int32
 17  trip_distance            174952 non-null  float64
 18  number_of_bikes          174952 non-null  int64
dtypes: datetime64[ns](2), float64(5), int32(3), int64(3), object(6)
memory usage: 23.4+ MB
```

In [108…
```
bike_clean['start_station_name'].nunique()
```

Out[108]:
```
329
```

There are 329 unique start stations in our dataset. We will investigate the popular start stations.

In [109…
```
import locale
locale.setlocale(locale.LC_ALL, 'en_US.UTF8')
```

Out[109]:
```
'en_US.UTF8'
```

Visualization

In [110…
```
import plotly.express as px

px.set_mapbox_access_token('pk.eyJ1Ijoic3Rpbmd6dyIsImEiOiJjbDF5dmJkc2EwZzltM2ptazNr
fig = px.scatter_mapbox(
    geo_start_df,
    lat = 'start_station_latitude',
    lon = 'start_station_longitude',
    size = 'total_bikes',
    color = 'unique_bikes',
    color_continuous_scale=px.colors.cyclical.IceFire,
    hover_data = ['start_station_name'],
    title = "San Francisco Bay Area bike rentals by Location"
)
fig.show()
```

San Francisco Bay Area bike rentals by Location

Observation

We see that generally stations along Market Street are very popular, with high bike rental numbers. The most popular station being "Market St at 10th St".

Conclusion

I did a lot of feature engineering in this dataset to uncover some insights. I first dropped all rows that had null values, and introduced new features like Age, Distance and Number of Bikes per trip.

There were some outliers especially in the Age column where the oldest rider had an age of 141. It was interesting to note that there are a considerable number of bikers in their 50s and 60s that are still active riders.

Males dominate in the dataset, taking 3 times more trips than Females. There are also about 8 times more trips taken by Subcribers, than by Customers.

Distance covered during trips also produced some unusual points because I used the haversian method of measuring distance between points. Therefore, trips that started and ended at the same station were recorded as zeroes.

Also notable is the fact that riders generally do not share bikes during their trips.

It was also apparent that most riders prefer to take trips during the week, than on weekends.

Main Street is popular with riders, with the most frequented being Market St at 10th St.

In [111…

```
# Saving clean dataset to file
#bike_clean.to_csv("bike_clean.csv", index=False)
```