Part_I_exploration

December 21, 2024

1 Part I - (Dataset Exploration For The Ford GoBike Trip Data)

1.1 by Momen Ghulmi

1.2 Introduction

In this Analysis, we will explore the Ford GoBike Dataset. Which is a set of records documenting individual rides in a bike sharing system covering the San Francisco Bay area. during the analysis process, we will explore the distribution of several variables, the relation between certain parameters and the effect of multiple variables has on each other.

1.3 Preliminary Wrangling

[1]:
```
# 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 sns
import math
import calendar
```

[2]:
```
# loading in the dataset into a pandas dataframe
df = pd.read_csv('data/raw/201902-fordgobike-tripdata_raw.csv')
```

[3]:
```
# checking the shape of the data
df.shape
```

[3]:
```
(183412, 16)
```

[4]:
```
# printing data sample for visual assessment
df.head(10)
```

[4]:
```
   duration_sec                start_time                  end_time  \
0         52185  2019-02-28 17:32:10.1450  2019-03-01 08:01:55.9750
1         42521  2019-02-28 18:53:21.7890  2019-03-01 06:42:03.0560
2         61854  2019-02-28 12:13:13.2180  2019-03-01 05:24:08.1460
3         36490  2019-02-28 17:54:26.0100  2019-03-01 04:02:36.8420
4          1585  2019-02-28 23:54:18.5490  2019-03-01 00:20:44.0740
5          1793  2019-02-28 23:49:58.6320  2019-03-01 00:19:51.7600
```

```
6         1147  2019-02-28 23:55:35.1040  2019-03-01 00:14:42.5880
7         1615  2019-02-28 23:41:06.7660  2019-03-01 00:08:02.7560
8         1570  2019-02-28 23:41:48.7900  2019-03-01 00:07:59.7150
9         1049  2019-02-28 23:49:47.6990  2019-03-01 00:07:17.0250

   start_station_id                                start_station_name  \
0              21.0  Montgomery St BART Station (Market St at 2nd St)
1              23.0                     The Embarcadero at Steuart St
2              86.0                           Market St at Dolores St
3             375.0                           Grove St at Masonic Ave
4               7.0                               Frank H Ogawa Plaza
5              93.0                      4th St at Mission Bay Blvd S
6             300.0                              Palm St at Willow St
7              10.0                         Washington St at Kearny St
8              10.0                         Washington St at Kearny St
9              19.0                              Post St at Kearny St

   start_station_latitude  start_station_longitude  end_station_id  \
0               37.789625              -122.400811            13.0
1               37.791464              -122.391034            81.0
2               37.769305              -122.426826             3.0
3               37.774836              -122.446546            70.0
4               37.804562              -122.271738           222.0
5               37.770407              -122.391198           323.0
6               37.317298              -121.884995           312.0
7               37.795393              -122.404770           127.0
8               37.795393              -122.404770           127.0
9               37.788975              -122.403452           121.0

                              end_station_name  end_station_latitude  \
0               Commercial St at Montgomery St             37.794231
1                           Berry St at 4th St             37.775880
2  Powell St BART Station (Market St at 4th St)             37.786375
3                       Central Ave at Fell St             37.773311
4                        10th Ave at E 15th St             37.792714
5                           Broadway at Kearny             37.798014
6                      San Jose Diridon Station             37.329732
7                       Valencia St at 21st St             37.756708
8                       Valencia St at 21st St             37.756708
9                           Mission Playground             37.759210

   end_station_longitude  bike_id   user_type  member_birth_year  \
0            -122.402923     4902    Customer             1984.0
1            -122.393170     2535    Customer                NaN
2            -122.404904     5905    Customer             1972.0
3            -122.444293     6638  Subscriber             1989.0
4            -122.248780     4898  Subscriber             1974.0
```

```
5           -122.405950     5200  Subscriber              1959.0
6           -121.901782     3803  Subscriber              1983.0
7           -122.421025     6329  Subscriber              1989.0
8           -122.421025     6548  Subscriber              1988.0
9           -122.421339     6488  Subscriber              1992.0

  member_gender bike_share_for_all_trip
0          Male                      No
1           NaN                      No
2          Male                      No
3         Other                      No
4          Male                     Yes
5          Male                      No
6        Female                      No
7          Male                      No
8         Other                      No
9          Male                      No
```

1.3.1 The structure of the dataset.

The dataset consist of 183412 records containing 16 columns. the columns and their description info are in the table below

Column	Description
duration_sec	the duration of the trip in seconds
start_time	the start time of the trip
end_time	The end time of the trip
start_station_id	the start station's ID
start_station_name	the start station's Name
start_station_latitude	the start station's Latitude
start_station_longitude	the start station's Longitude
end_station_id	the end station's ID
end_station_name	the end station's Name
end_station_latitude	the end station's Latitude
end_station_longitude	the end station's Longitude
bike_id	the bike's ID
user_type	the Type of user (Subscriber/Customer)
member_birth_year	the member's Birth year
member_gender	the member's Gender
bike_share_for_all_trip	Whether the bike was shared for all trips

1.3.2 The main features of interest in the dataset.

the main features of interest in this are duration_sec and bike_share_for_all_trip. we will focus on analyzing them and their relation ships with other variables.

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

the variables that are most likely to support the investigation are: member_birth_year, distance (will be calculated using longitude and latitude), user_type and member_gender.

1.3.4 Data Cleaning

Before we can start exploring the Dataset, we need to clean the Data to make sure it's clean and Tidy

[5]:
```
# creating a copy of the dataframe to work on inorder to keep the original data
df_clean = df.copy()
```

Completeness the first step is to check the data set for null values

[6]:
```
# checking for missing values
df_clean.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
```

[7]:
```
#checking number of null values in each column
df_clean.isnull().sum()
```

[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
```

By looking at the columns info we can see that we have 197 records missing station names and 8265 records missing member info. As their numbers are negligible compared to the dataset size, we can just drop those records.

[8]:
```python
# dropping null records
df_clean.dropna(inplace=True)
```

[9]:
```python
#verifying if the null records were dropped
df_clean.isna().sum()
```

[9]:
```
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
```

Duplicates the next step is to look for duplicates in the dataset

[10]:
```python
# checking for duplicates
df_clean.duplicated().sum()
```

[10]: np.int64(0)

There are no Duplicates in the dataset

Validity next we need to check the validity of the data

[11]:
```
# checking validity of numerical columns
df_clean.describe()
```

[11]:
```
         duration_sec  start_station_id  start_station_latitude  \
count   174952.000000     174952.000000           174952.000000
mean       704.002744        139.002126               37.771220
std       1642.204905        111.648819                0.100391
min         61.000000          3.000000               37.317298
25%        323.000000         47.000000               37.770407
50%        510.000000        104.000000               37.780760
75%        789.000000        239.000000               37.797320
max      84548.000000        398.000000               37.880222

       start_station_longitude  end_station_id  end_station_latitude  \
count            174952.000000   174952.000000         174952.000000
mean               -122.351760      136.604486             37.771414
std                   0.117732      111.335635              0.100295
min                -122.453704        3.000000             37.317298
25%                -122.411901       44.000000             37.770407
50%                -122.398279      101.000000             37.781010
75%                -122.283093      238.000000             37.797673
max                -121.874119      398.000000             37.880222

       end_station_longitude        bike_id  member_birth_year
count          174952.000000  174952.000000      174952.000000
mean             -122.351335    4482.587555        1984.803135
std                 0.117294    1659.195937          10.118731
min              -122.453704      11.000000        1878.000000
25%              -122.411647    3799.000000        1980.000000
50%              -122.397437    4960.000000        1987.000000
75%              -122.286533    5505.000000        1992.000000
max              -121.874119    6645.000000        2001.000000
```

looking at the member birth year we can see that we have members born in 1878, which is most likely an entry issue. so to fix this issue we will remove all data with ages larger than one hundred as they are less likely to be valid. As the data was in 2019 that would make the minimum year 1920.

[12]:
```
# removing member birth years less that 1920
df_clean = df_clean[df_clean['member_birth_year'] > 1920]
```

Tidiness we have both start_time and end_time containing the date and time data, which need to be separated.

[13]:
```
#seperating the start time column into date and time columns
df_clean['start_time'] = pd.to_datetime(df_clean['start_time'])
df_clean['start_date'] = df_clean['start_time'].dt.date
df_clean['start_time'] = df_clean['start_time'].dt.time
```

[14]:
```
#seperating the end time column into date and time columns
df_clean['end_time'] = pd.to_datetime(df_clean['end_time'])
df_clean['end_date'] = df_clean['end_time'].dt.date
df_clean['end_time'] = df_clean['end_time'].dt.time
```

[15]:
```
#checking the data visually
df_clean.head(10)
```

[15]:
```
    duration_sec       start_time         end_time  start_station_id  \
0          52185  17:32:10.145000  08:01:55.975000              21.0
2          61854  12:13:13.218000  05:24:08.146000              86.0
3          36490  17:54:26.010000  04:02:36.842000             375.0
4           1585  23:54:18.549000  00:20:44.074000               7.0
5           1793  23:49:58.632000  00:19:51.760000              93.0
6           1147  23:55:35.104000  00:14:42.588000             300.0
7           1615  23:41:06.766000  00:08:02.756000              10.0
8           1570  23:41:48.790000  00:07:59.715000              10.0
9           1049  23:49:47.699000  00:07:17.025000              19.0
10           458  23:57:57.211000  00:05:35.435000             370.0

                                   start_station_name  start_station_latitude  \
0   Montgomery St BART Station (Market St at 2nd St)               37.789625
2                           Market St at Dolores St               37.769305
3                           Grove St at Masonic Ave               37.774836
4                               Frank H Ogawa Plaza               37.804562
5                         4th St at Mission Bay Blvd S               37.770407
6                               Palm St at Willow St               37.317298
7                          Washington St at Kearny St               37.795393
8                          Washington St at Kearny St               37.795393
9                               Post St at Kearny St               37.788975
10                              Jones St at Post St               37.787327

    start_station_longitude  end_station_id  \
0               -122.400811            13.0
2               -122.426826             3.0
3               -122.446546            70.0
4               -122.271738           222.0
5               -122.391198           323.0
6               -121.884995           312.0
7               -122.404770           127.0
```

```
8            -122.404770           127.0
9            -122.403452           121.0
10           -122.413278            43.0

                                 end_station_name  end_station_latitude  \
0                  Commercial St at Montgomery St             37.794231
2       Powell St BART Station (Market St at 4th St)             37.786375
3                            Central Ave at Fell St             37.773311
4                             10th Ave at E 15th St             37.792714
5                                Broadway at Kearny             37.798014
6                          San Jose Diridon Station             37.329732
7                            Valencia St at 21st St             37.756708
8                            Valencia St at 21st St             37.756708
9                                Mission Playground             37.759210
10  San Francisco Public Library (Grove St at Hyde…            37.778768

    end_station_longitude  bike_id   user_type  member_birth_year  \
0             -122.402923     4902    Customer             1984.0
2             -122.404904     5905    Customer             1972.0
3             -122.444293     6638  Subscriber             1989.0
4             -122.248780     4898  Subscriber             1974.0
5             -122.405950     5200  Subscriber             1959.0
6             -121.901782     3803  Subscriber             1983.0
7             -122.421025     6329  Subscriber             1989.0
8             -122.421025     6548  Subscriber             1988.0
9             -122.421339     6488  Subscriber             1992.0
10            -122.415929     5318  Subscriber             1996.0

   member_gender bike_share_for_all_trip  start_date    end_date
0           Male                      No  2019-02-28  2019-03-01
2           Male                      No  2019-02-28  2019-03-01
3          Other                      No  2019-02-28  2019-03-01
4           Male                     Yes  2019-02-28  2019-03-01
5           Male                      No  2019-02-28  2019-03-01
6         Female                      No  2019-02-28  2019-03-01
7           Male                      No  2019-02-28  2019-03-01
8          Other                      No  2019-02-28  2019-03-01
9           Male                      No  2019-02-28  2019-03-01
10        Female                     Yes  2019-02-28  2019-03-01
```

In the above code we separated the columns into start_date and end_date containing the date data, and start_time, end_time containing the time data

Columns Data Types in the final cleaning phase we need to make sure every column has the correct data type

[16]:
```
#checking the data types of the columns
df_clean.dtypes
```

[16]:
```
duration_sec                  int64
start_time                   object
end_time                     object
start_station_id            float64
start_station_name           object
start_station_latitude      float64
start_station_longitude     float64
end_station_id              float64
end_station_name             object
end_station_latitude        float64
end_station_longitude       float64
bike_id                       int64
user_type                    object
member_birth_year           float64
member_gender                object
bike_share_for_all_trip      object
start_date                   object
end_date                     object
dtype: object
```

In order to fix the data types we need to fix the following: - (start_station/end_station)_id: to int - user_type: to category - member_birth_year: to int - member_gender: to category - bike_share_for_all_trip: to category

[17]:
```
#fixing the data types of the columns
df_clean['start_station_id'] = df_clean['start_station_id'].astype(int)
df_clean['end_station_id'] = df_clean['end_station_id'].astype(int)
df_clean['bike_id'] = df_clean['bike_id'].astype(int)
df_clean['member_birth_year'] = df_clean['member_birth_year'].astype(int)
df_clean['user_type'] = df_clean['user_type'].astype('category')
df_clean['member_gender'] = df_clean['member_gender'].astype('category')
df_clean['bike_share_for_all_trip'] = df_clean['bike_share_for_all_trip'].
 ↪astype('category')
```

[18]:
```
df_clean.dtypes
```

[18]:
```
duration_sec                   int64
start_time                    object
end_time                      object
start_station_id               int64
start_station_name            object
start_station_latitude       float64
start_station_longitude      float64
end_station_id                 int64
end_station_name              object
```

```
end_station_latitude         float64
end_station_longitude        float64
bike_id                        int64
user_type                   category
member_birth_year              int64
member_gender               category
bike_share_for_all_trip     category
start_date                    object
end_date                      object
dtype: object
```

with the data types fixed the data is cleaned.

Creating the additional parameters

distance one of the variables we could compute is the distant traveled in Trip. this can be calculated using the longitude and latitude through the haversine formula. the formula and its code has been referenced from https://www.movable-type.co.uk/scripts/latlong.html. Please view the page to learn more

[19]:
```python
# calculating the distance between the start and end stations using the␣
↪haversine formula. this returns the distance in miters
def calc_distance(lat1, lon1, lat2, lon2):
    R = 6371e3  #meters
    phi1 = math.radians(lat1)
    phi2 = math.radians(lat2)
    delta_phi = math.radians(lat2 - lat1)
    delta_lambda = math.radians(lon2 - lon1)
    a = math.sin(delta_phi / 2)**2 + math.cos(phi1) * math.cos(phi2) * math.
↪sin(delta_lambda / 2)**2
    a = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    return R * a
```

[20]:
```python
#creating the distance column
df_clean['distance'] = df_clean.apply(lambda x:␣
↪calc_distance(x['start_station_latitude'], x['start_station_longitude'],␣
↪x['end_station_latitude'], x['end_station_longitude']), axis=1)
df_clean['distance'] = df_clean['distance'].astype(int)
df_clean['distance'].head(10)
```

[20]:
```
0       544
2      2704
3       260
4      2409
5      3332
6      2028
7      4532
```

```
8     4532
9     3664
10     979
Name: distance, dtype: int64
```

[21]:
```python
#description statistics of the distance column
df_clean['distance'].describe()
```

[21]:
```
count    174877.000000
mean       1689.399172
std        1096.668979
min           0.000000
25%         910.000000
50%        1429.000000
75%        2223.000000
max       69469.000000
Name: distance, dtype: float64
```

with this we created a distance variable that contains the traveled distance in meters

Age to create the age we will subtract the birth year from 2019 as it was the year the trips

[22]:
```python
# creating the age column
df_clean['age'] = 2019 - df_clean['member_birth_year']
```

day of week to create a day of week parameter we will take the day of week from the start date

[23]:
```python
#getting the day of week
df_clean['start_day'] = df_clean['start_date'].apply(lambda x: calendar.
 ↪day_name[x.weekday()])
```

[24]:
```python
#creating a category type for the day of the week for sorting
days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',␣
 ↪'Sunday']
day_type = pd.CategoricalDtype(categories=days, ordered=True)
df_clean['start_day'] = df_clean['start_day'].astype(day_type)
```

finally we need to reset the index, then saving the data into clean data file

[25]:
```python
#resetting the index
df_clean.reset_index(drop=True, inplace=True)
```

[26]:
```python
#saving the cleaned data to a new csv file
df_clean.to_csv('data/clean/201902-fordgobike-tripdata_clean.csv', index=False)
```

1.4 Univariate Exploration

In this section, we will investigate distributions of individual variables.

1.4.1 Taking a Random Sample

as the data is too large for plotting we will need to take a sample of the data to plot with

[27]:
```
# taking a random sample of the data to work with
random_seed = 42
df_clean_sample = df_clean.sample(1000, random_state=random_seed)
```

1.4.2 what is the distribution of duration and what is the most common duration of rental ?

to look at the distribution we will need first to look at the descriptive statistics of the duration_sec column

[28]:
```
df_clean_sample['duration_sec'].describe()
```

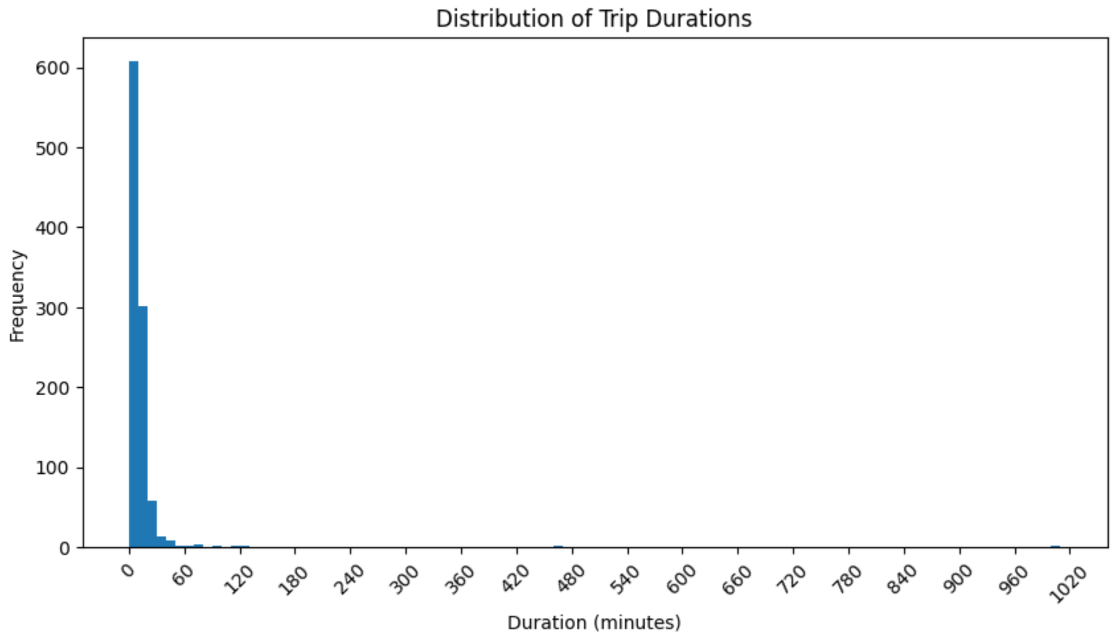
[28]:
```
count     1000.000000
mean       726.954000
std       2168.917953
min         70.000000
25%        306.750000
50%        496.000000
75%        767.500000
max      60441.000000
Name: duration_sec, dtype: float64
```

As the data contains some valid outliers with large values, we will need to convert the data into minutes in order to look at the data better

[29]:
```
def duration_hist(data, bins_step, ticks_step, x_label, y_label, title,␣
 ↪rotation):
    plt.figure(figsize=(10, 5));
    bins = np.arange(0, data.max() + bins_step, bins_step)
    plt.hist(data, bins=bins)
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.title(title)
    plt.xticks(np.arange(0, data.max() + ticks_step, ticks_step),␣
 ↪rotation=rotation)
```

[30]:
```
#converting the duration column to minutes
duration_min = df_clean_sample['duration_sec'] / 60
```

[31]:
```
#plotting the distribution of the duration in minutes
#setting the xticks to be in 60 minutes intervals
duration_hist(duration_min, 10, 60, 'Duration (minutes)', 'Frequency',␣
 ↪'Distribution of Trip Durations', 45)
```



[32]:
```
#confirming the count of trips with duration greater than 60 minutes in the␣
↪whole dataset
df_clean[df_clean['duration_sec'] > 3600].shape[0]
```

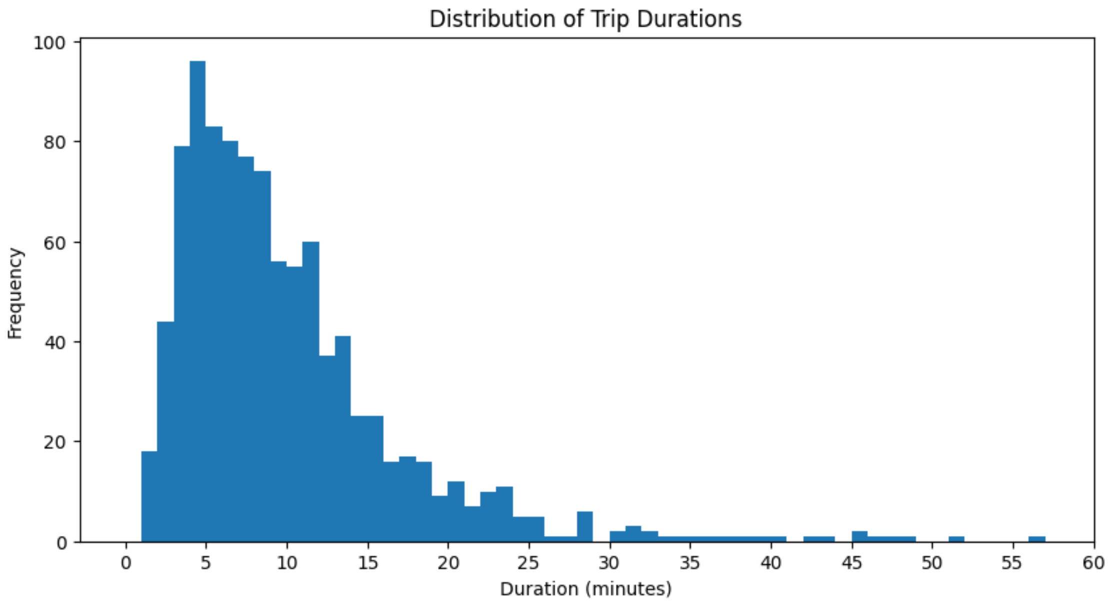
[32]: 1386

[33]:
```
# confirming the count of trips with duration greater than 60 minutes in the␣
↪sample dataset
df_clean_sample[df_clean_sample['duration_sec'] > 3600].shape[0]
```

[33]: 10

we can see that the distribution is very skewed to the left with most of the data being less that 30 minutes. Moreover we confirmed that from all the data we have 1386 records in the whole data and 8 records in the sample data with a duration larger that one hour, which are negligible numbers. So to get a clearer picture of the data we will plot the data without the outliers.

[34]:
```
duration_hist(duration_min[duration_min <= 60], 1, 5, 'Duration (minutes)',␣
↪'Frequency', 'Distribution of Trip Durations', 0)
```

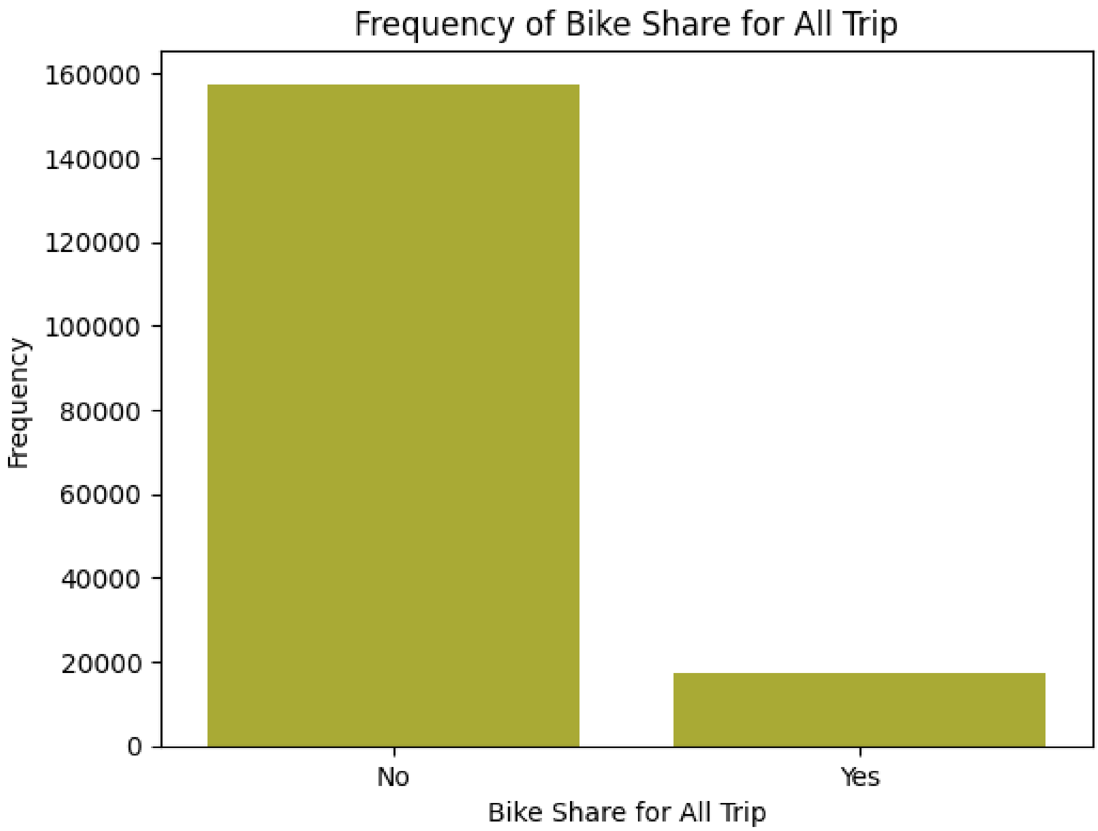


Conclusion As seen in the histogram above, the distribution of the duration_sec is heavily skewed to the left with the most common duration times being between 4-8

1.4.3 What is the ratio of the bike share for all trip ?

in order to look at the look at the ration of the all trip bike shares, we will plot a bar plot of the variable

[35]:
```
sns.countplot(data=df_clean, x='bike_share_for_all_trip', color='tab:olive')
plt.xlabel('Bike Share for All Trip')
plt.ylabel('Frequency')
plt.title('Frequency of Bike Share for All Trip');
```



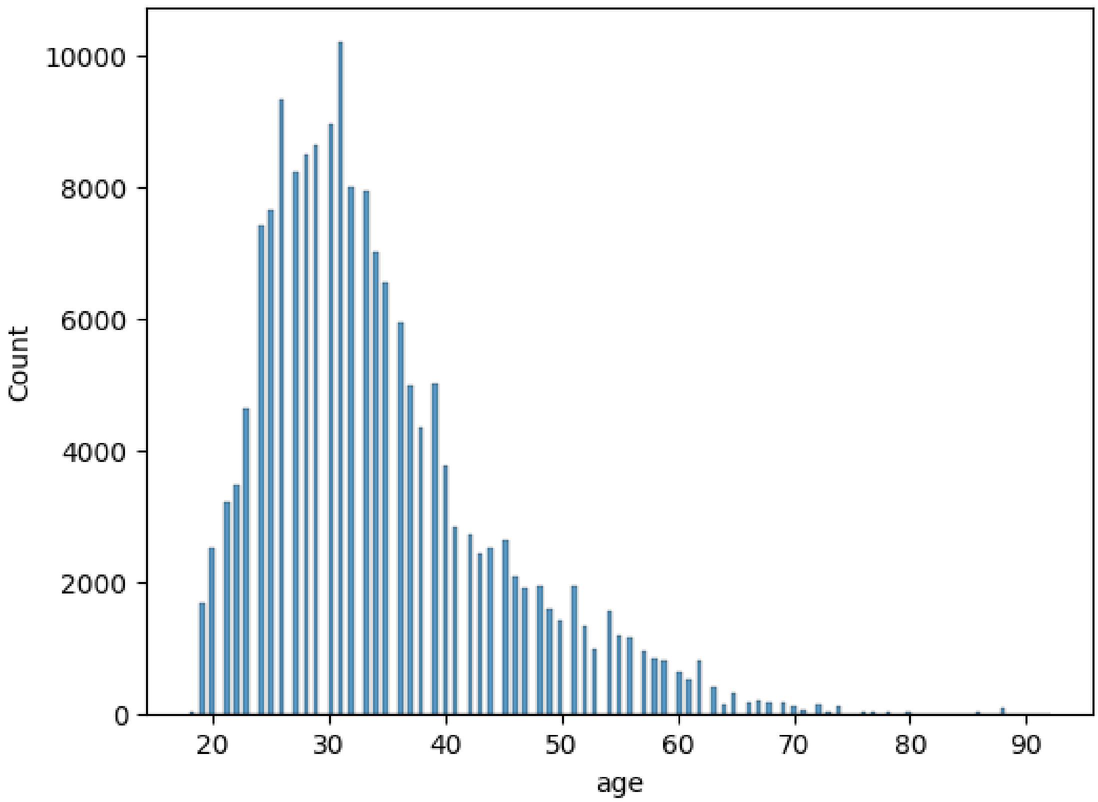
conclusion The bar plot above shows that the majority of bike trips are not shared for all trips. with a small minority sharing for all trip

1.4.4 What is the distribution of the age of members ?

To look at the distribution of age we will create a histogram for it.

[36]:
```
#plotting the histogram of the age column
sns.histplot(data=df_clean, x='age', color='tab:blue')
```

[36]: <Axes: xlabel='age', ylabel='Count'>



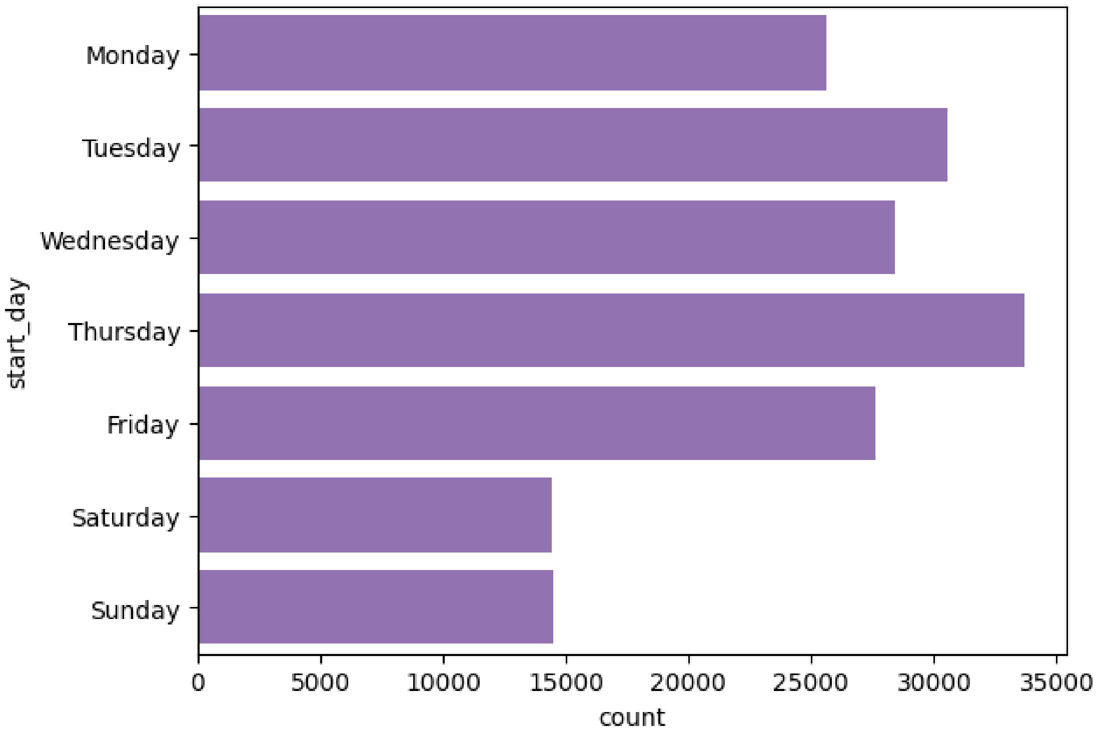
Conclusion: We can see that the distribution is skewed to the left, with most of the members being under 40

1.4.5 What days is the rental used most ?

In order to answer this question we will need to plot a bar chart for the days of week.

[37]:
```
#plotting a horizontal bar chart of the day of the week
sns.countplot(data=df_clean, y='start_day', color='tab:purple')
```

[37]:
```
<Axes: xlabel='count', ylabel='start_day'>
```



Conclusion: From the plot above, We can determine that of the days of the week Thursday has the most traffic, followed by Tuesday. With Saturday and Sunday Having the least traffic by a good margin.

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

In the univariate analysis we plotted 4 parameter: duration, bike_share_all_trip, day of week and age. We First had to covert the duration top minutes because of the large and take a sample because of the large and wide distribution of Second, and after plotting the histogram we found the distribution leaning heavily to below 15 minutes. The bike share bar chart informed us that most of the users do not share rides for all trip.Moreover, we found that that most bike use is on work days with the weekend having the least traffic. Finally we found the age histogram skewed to the left with most members below 40. ### 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? there were no unusual distributions as the duration and age are logically expected to be this way. and I needed to change the duration data to minutes for the histogram to be readable.

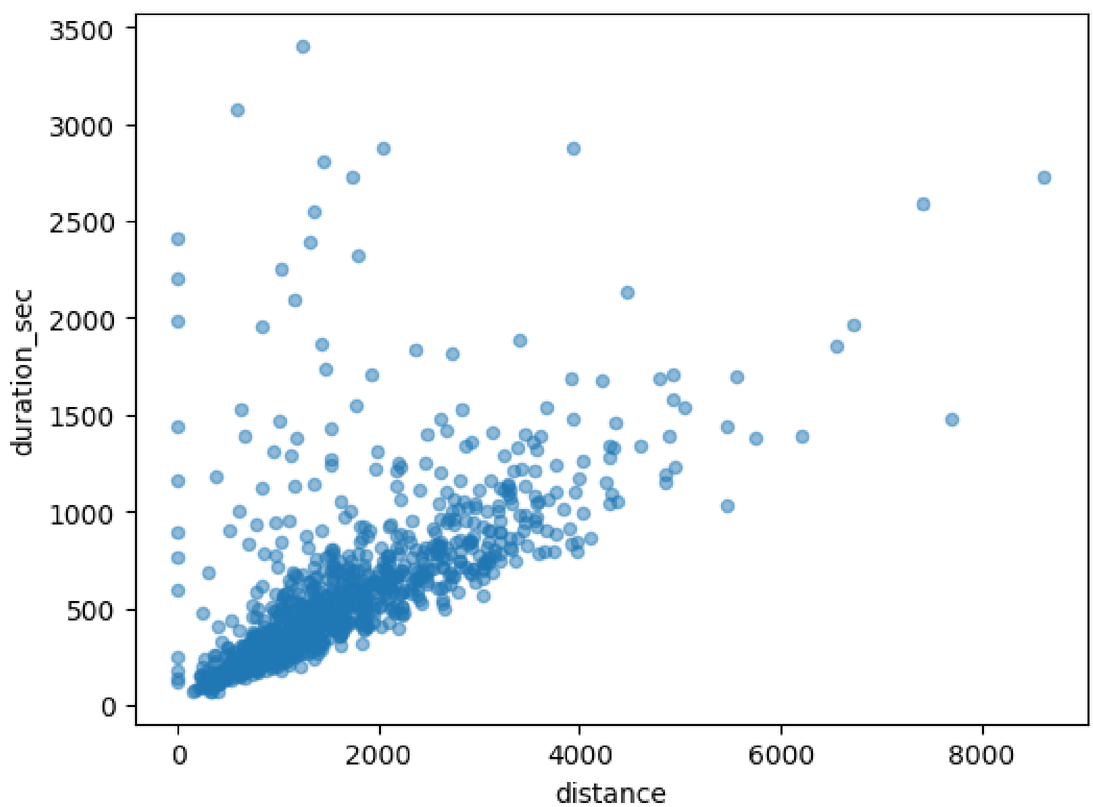
1.5 Bivariate Exploration

1.5.1 Are the Trips facing delays ?

In order to look figure if the Trips are facing delays, we need will plot the relation between distance traveled and duration_sec in a scatter plot. a high corelation in the plot indactes that trips are going slow. And a low correlation in the relation would mean that some trips are facing delays and taking more time that necessary to the point where they approch the time of longer trips. We will filter out outliers for the histogram to be easier to read.

[38]:
```
df_clean_sample[df_clean_sample["duration_sec"] <= 3600].plot(kind="scatter",␣
↪x='distance', y='duration_sec', alpha=0.5)
```

[38]: `<Axes: xlabel='distance', ylabel='duration_sec'>`



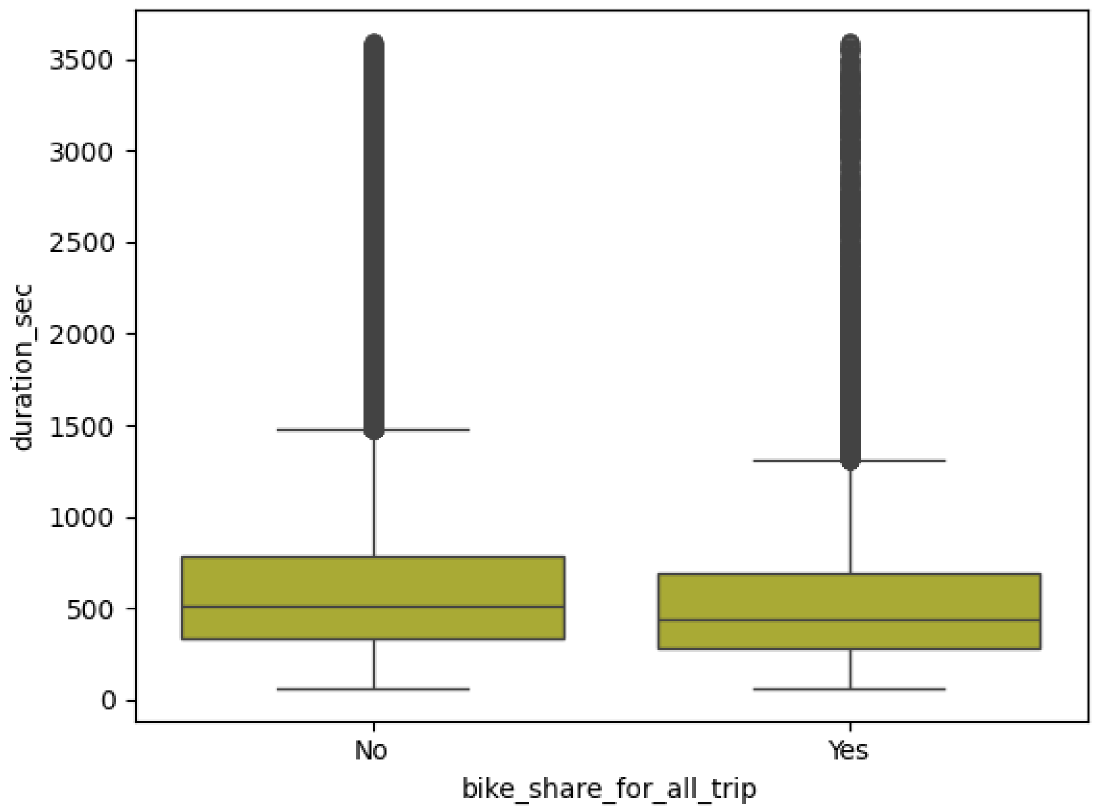
Conclusion: As seen above the relation is mostly a high correlation, with some Trips most likely facing some delay and multiple trips that have a high duration time with no distance traveled which indicates roundabout trips returning to the starting point in the end of the trip.

1.5.2 Is there a difference of time between sharing for all trip and not?

For this Question, we will plot a box plot between bike_share_for_all_trip and duration_sec variables to see if the distribution differs between them. We will filter the outliers from the duration_sec for the plot to be readable.

[39]:
```
sns.boxplot(data=df_clean[df_clean['duration_sec'] < 3600],
    x="bike_share_for_all_trip", y="duration_sec", color='tab:olive')
```

[39]:
```
<Axes: xlabel='bike_share_for_all_trip', ylabel='duration_sec'>
```



Conclusion: We can see that the duration is slightly less in average for the members sharing all trip.

1.5.3 Are different member types more likely to share all trip ?

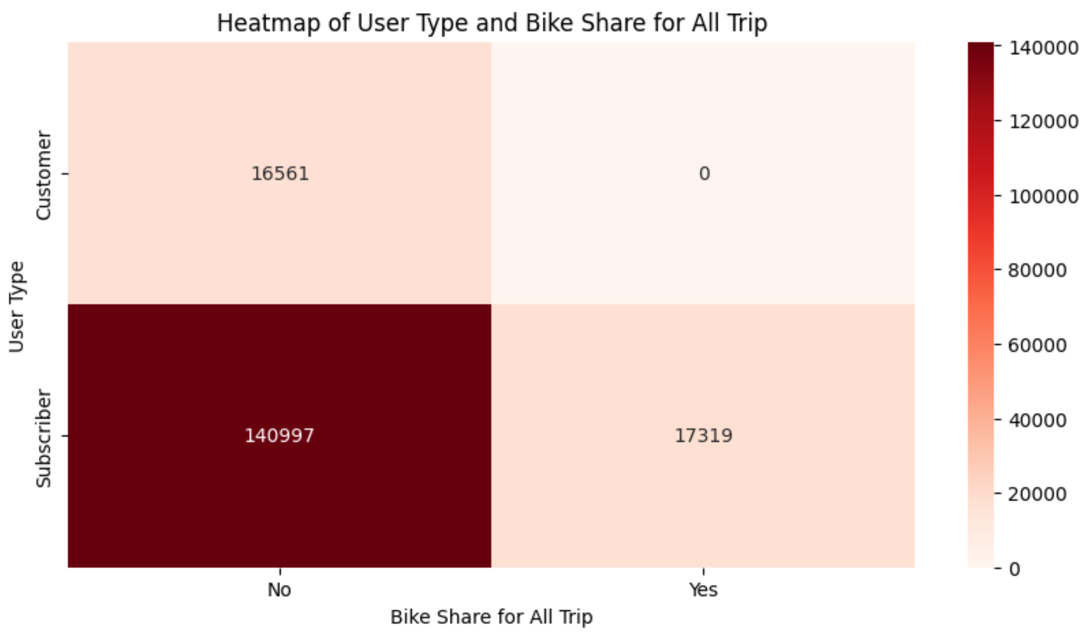
In order to answer this question we will plot a heat map between the user_type and bike_share_for_all_trip variables

Before we can plot we need to prepare the data

[40]:
```
#preparing the data for the heatmap
df_heatmap = df_clean.groupby(['user_type', 'bike_share_for_all_trip'],
    observed=False).size().reset_index(name='count')
df_heatmap = df_heatmap.pivot(index='user_type',
    columns='bike_share_for_all_trip', values='count')
```

[41]:
```
#plotting the heatmap

plt.figure(figsize=(10, 5))
sns.heatmap(df_heatmap, annot=True, fmt='d', cmap='Reds', cbar_kws={'label':␣
↪'Count'})
plt.xlabel('Bike Share for All Trip')
plt.ylabel('User Type')
plt.title('Heatmap of User Type and Bike Share for All Trip');
```



Conclusion: From the heatmap above, We can see that no customers share all trip, with a small percentage of Subscribers do share all trip.

1.5.4 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?

In the Bivariate analysis above we explored three relationships. First, through the scatter plot we observed a positive correlation relation between duration and distance with a small number of trip facing delays and some roundabout trips. Moreover, in the Box plot we observed a slight difference in duration between all trip bike sharing with member sharing all trip having a slightly lower averages. Finally through the heat map we found that bike sharing for all trip being exclusive to Subscribers only.

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

NO

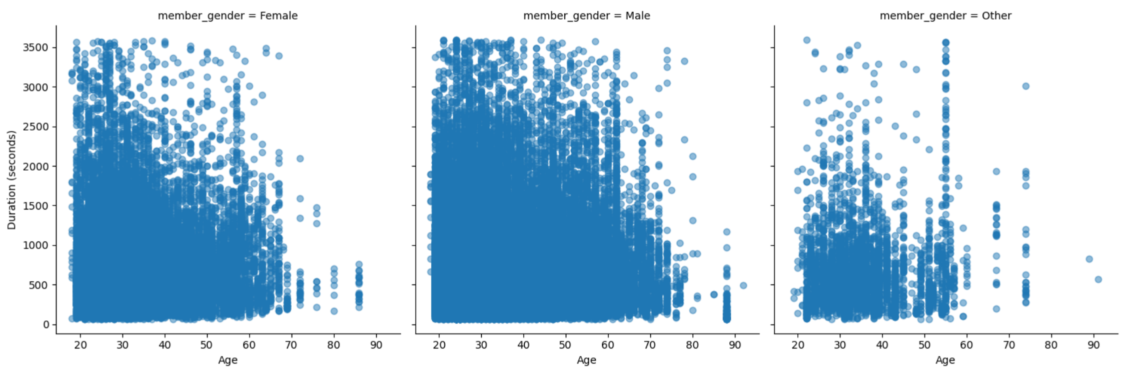
1.6 Multivariate Exploration

1.6.1 does duration vary between age and gender ?

For this question we will plot a gender focused FacetGrid with scatter plots between age and duration. we will filter out duration outliers for better observation.

[42]:
```
#plotting the facet grid
df_duration = df_clean[df_clean['duration_sec'] <= 3600]
g = sns.FacetGrid(data=df_duration, col='member_gender', col_wrap=3, height=5,␣
↪aspect=1)
g.map(plt.scatter, 'age', 'duration_sec', alpha=0.5)
g.set_xlabels('Age')
g.set_ylabels('Duration (seconds)')
```

[42]: <seaborn.axisgrid.FacetGrid at 0x19df12a9940>

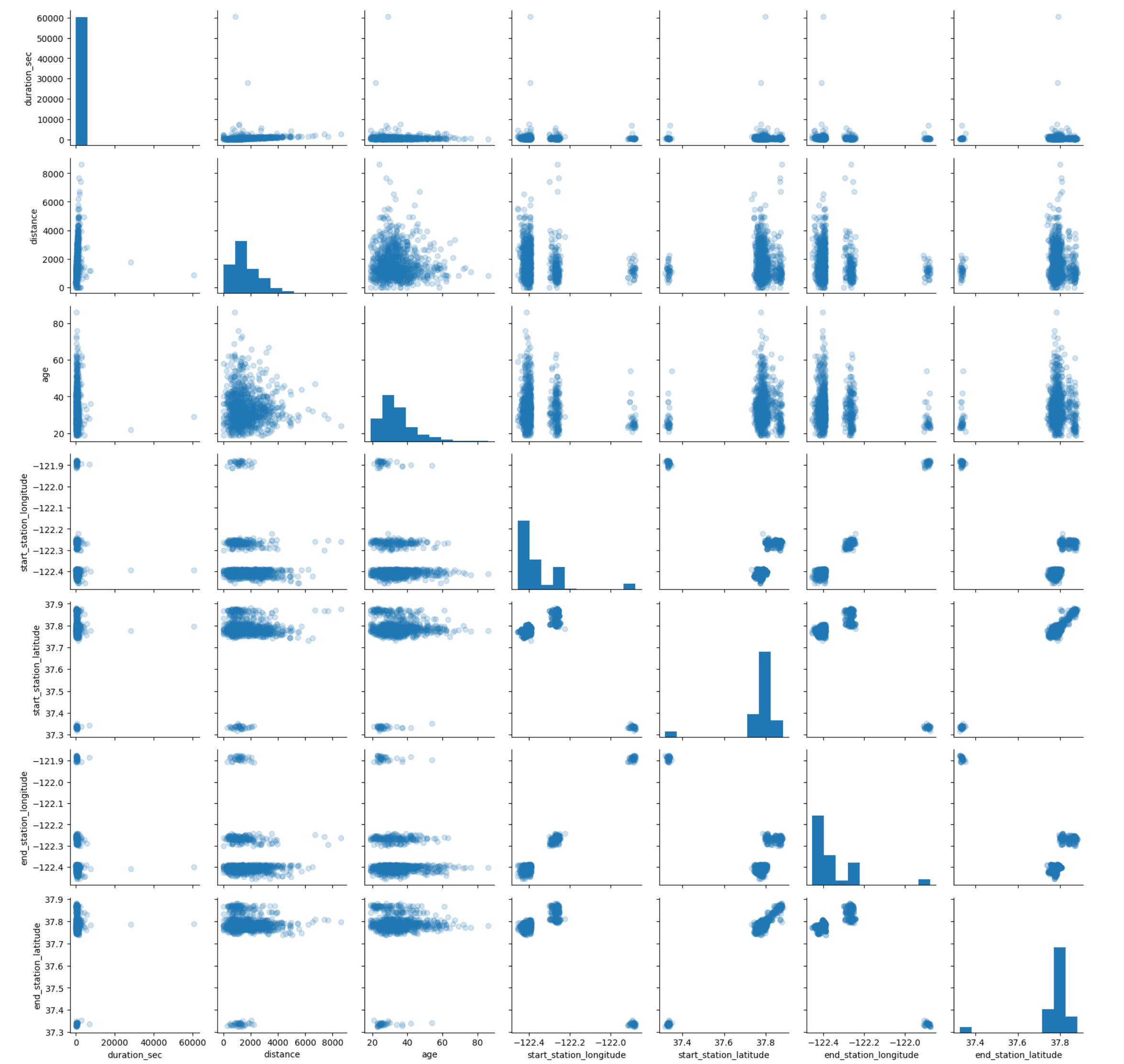


Conclusion: As seen the FacetGrid above, age and gender seem to have no effect on duration of the trip taken.

1.6.2 Are there any interesting relations we missed ?

In order to investigate this we will plot a scatter matrix of the numerical variables to see what other relations we may have missed.

[43]:
```
#preparing data for the pair plot
vars = ['duration_sec', 'distance', 'age', 'start_station_longitude',␣
↪'start_station_latitude', 'end_station_longitude', 'end_station_latitude']
#plotting the pair plot
g = sns.PairGrid(data=df_clean_sample, vars=vars)
g = g.map_diag(plt.hist)
g = g.map_offdiag(plt.scatter, alpha=0.2)
```



1.6.3 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?

In this multivariate analysis, we looked first at the effect of age and gender on the duration, then we looked at a scatter matrix to see if we missed any relations. However, no interesting relations have been observed through the plots.

1.6.4 Were there any interesting or surprising interactions between features?

No

1.7 Conclusions

Through This analysis, we attempted to answer a main question: How is the rental being used. To that end we Cleaned a prepared the data, then we created several visualizations in an effort to answer this question and related observations. The findings from this investigation are as follows:

- Most trips durations are less than 15 minutes, with trips over 25 minutes being rare.
- A small minority share bike all trip.
- The majority of the users are under 40 years old
- The Weekend see little use compared to work day, with Thursday having most traffic followed by Tuesday.
- The Trips seem to be going mostly smoothly, with a few seeming to be facing unexpected delays. We also have a few roundabout trips returning to the start station.
- most users do not share all trip.
- all the users who share all trip are Subscribers, with no Regular Customers sharing all trip.
- the distribution of duration does not vary with age or gender.