Part_I_exploration_template

April 26, 2022

1 Part I - Exploring Ford GoBike System Data

1.1 by Mark Lam

1.2 Introduction

This data set includes information about individual rides made in a bike-sharing system covering the greater San Francisco Bay area.

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

        %matplotlib inline

In [3]: df = pd.read_csv('201902-fordgobike-tripdata.csv')

In [4]: 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 [5]: df.head()

```
Out[5]:    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

           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

           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

                                      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

           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

          member_gender bike_share_for_all_trip
        0          Male                      No
        1           NaN                      No
```

```
2          Male                        No
3         Other                        No
4          Male                       Yes
```

```
In [6]: # to get a new column of duration in minutes
        df['duration_min'] = (df['duration_sec']/60).round(2)
```

```
In [7]: # to change the data type of start_time to datetime
        df['start_time'] = pd.to_datetime(df['start_time'])
```

```
In [8]: # to change the data type of end_time to datetime
        df['end_time'] = pd.to_datetime(df['end_time'])
```

```
In [9]: df['start_day'] = df['start_time'].dt.day_name()
```

```
In [10]: df['end_day'] = df['end_time'].dt.day_name()
```

```
In [11]: df['age'] = 2019 - df['member_birth_year']
```

```
In [12]: # to get the distance, I'd like to use the module geopy.
         !pip install geopy
```

```
Collecting geopy
  Downloading https://files.pythonhosted.org/packages/e1/e1/45f25e3d3acf26782888f847de7c958a2807
    100% || 122kB 4.5MB/s ta 0:00:01
Collecting geographiclib<2,>=1.49 (from geopy)
  Downloading https://files.pythonhosted.org/packages/df/60/d1d4c4944f9726228faa80fbe2206c8ddfd9
Installing collected packages: geographiclib, geopy
Successfully installed geographiclib-1.52 geopy-2.2.0
```

```
In [13]: import geopy.distance as dist
```

```
In [14]: # to get the column with tuple of start latitude and longitude
         df['start_coords'] = df.apply(lambda x: (x[5],x[6]), axis = 1)
```

```
In [15]: # to get the column with tuple of end latitude and longitude
         df['end_coords'] = df.apply(lambda x: (x[9],x[10]), axis = 1)
```

```
In [16]: # to get the distance by start and end coordinates
         df['distance'] = df.apply(lambda x: dist.distance(x[-2],x[-1]).miles, axis = 1)
```

```
In [17]: # To get the start time in hour
         df['start_hour'] = df['start_time'].apply(lambda x: x.hour)
```

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

```
Out[18]:    duration_sec              start_time                end_time  \
         0         52185 2019-02-28 17:32:10.145 2019-03-01 08:01:55.975
         1         42521 2019-02-28 18:53:21.789 2019-03-01 06:42:03.056
```

```
2        61854 2019-02-28 12:13:13.218 2019-03-01 05:24:08.146
3        36490 2019-02-28 17:54:26.010 2019-03-01 04:02:36.842
4         1585 2019-02-28 23:54:18.549 2019-03-01 00:20:44.074

   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

   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

                                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

      ...      member_gender  bike_share_for_all_trip duration_min  start_day  \
0     ...               Male                       No       869.75   Thursday
1     ...                NaN                       No       708.68   Thursday
2     ...               Male                       No      1030.90   Thursday
3     ...              Other                       No       608.17   Thursday
4     ...               Male                      Yes        26.42   Thursday

  end_day   age                             start_coords  \
0  Friday  35.0              (37.7896254, -122.400811)
1  Friday   NaN        (37.791464000000005, -122.391034)
2  Friday  47.0        (37.76930529999999, -122.4268256)
3  Friday  30.0  (37.77483629413345, -122.44654566049576)
4  Friday  45.0   (37.8045623549303, -122.27173805236816)

                                  end_coords  distance  start_hour
0        (37.794230999999996, -122.402923)  0.338015          17
1                   (37.77588, -122.39317)  1.081130          18
2  (37.78637526861584, -122.40490436553955)  1.681051          12
3  (37.77331087889723, -122.44429260492323)  0.162113          17
4        (37.7927143, -122.24877959999999)  1.498758          23

[5 rows x 24 columns]
```

```
In [19]: df['weekend'] = df.apply(lambda x: 'weekday' if x['start_day'] in ['Monday', 'Tuesday',
```

```
In [20]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 25 columns):
duration_sec               183412 non-null int64
start_time                 183412 non-null datetime64[ns]
end_time                   183412 non-null datetime64[ns]
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
duration_min               183412 non-null float64
start_day                  183412 non-null object
end_day                    183412 non-null object
age                        175147 non-null float64
start_coords               183412 non-null object
end_coords                 183412 non-null object
distance                   183412 non-null float64
start_hour                 183412 non-null int64
weekend                    183412 non-null object
dtypes: datetime64[ns](2), float64(10), int64(3), object(10)
memory usage: 35.0+ MB
```

1.3.1 What is the structure of your dataset?

There are 183,412 bike trips in the dataset with 16 columns of information about each trip (duration_sec, start_time, end_time, start_station_id,start_station_name, start_station_latitude, start_station_longitude, end_station_id, end_station_name, end_station_latitude, end_station_longitude, bike_id, user_type, member_birth_year, member_gender, bike_share_for_all_trip). Most variables regarding the trips are numeric in nature, but there are also variables about the riders.

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

I'd like to see how long the average trip takes. How user types and gender would affect the riding behaviour. How age would affect the riding behaviour. How weekdays and weekends riding activities would differ. And finally the top 10 starting and ending spots during weekdays and weekends respectively.

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

I expect duration is the indicator of how long each ride takes. Features like user type, member gender, member birth year, and some new columns like duration_min (in minutes), age of users, distance calculated by coordinates, start_hour from start_time, start day and weekday/weekend, would help my investigation.

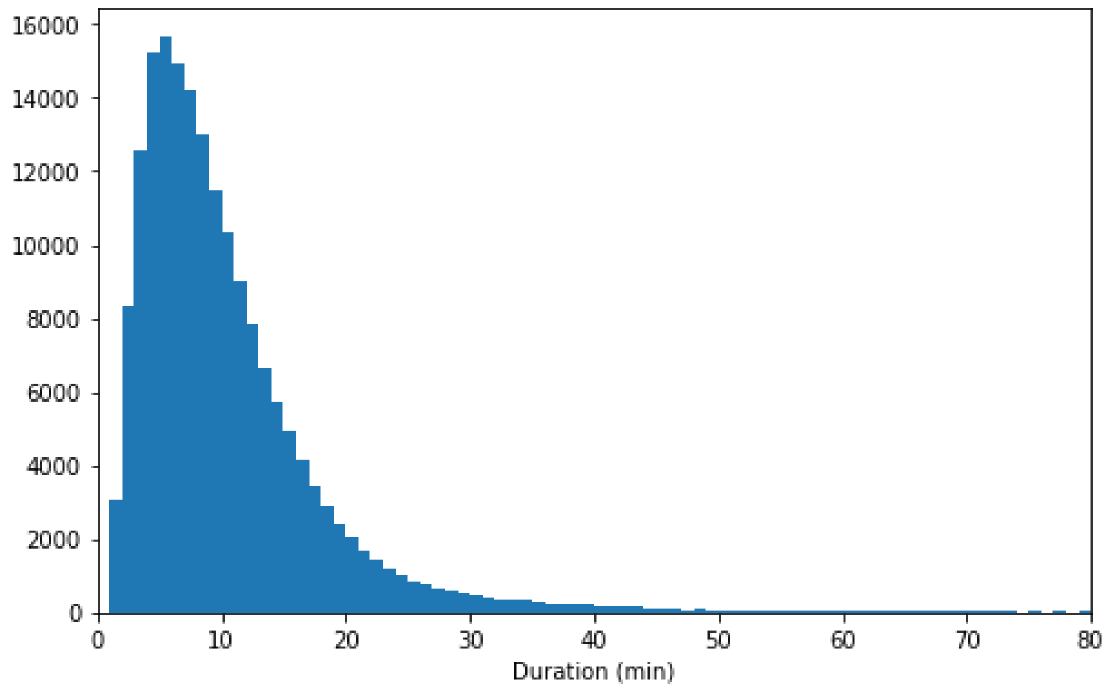
1.4 Univariate Exploration

The first variable I look at is 'duration_min', which is the duration of the trip in minutes.

```
In [21]: df['duration_min'].describe()
```

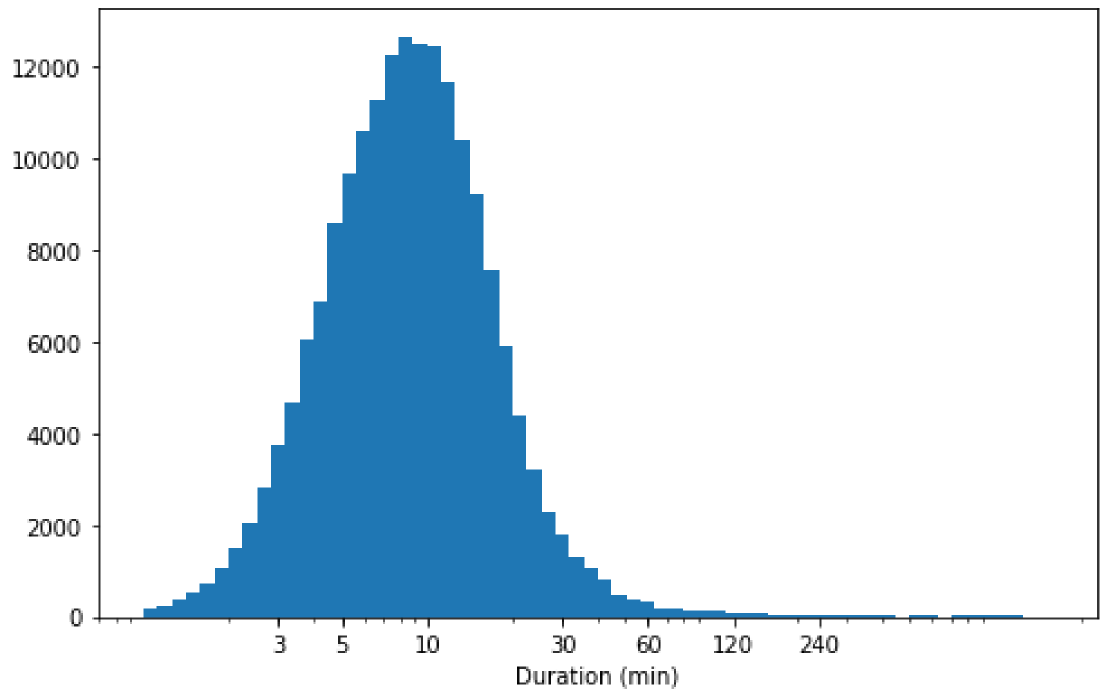
```
Out[21]: count    183412.000000
         mean         12.101301
         std          29.906501
         min           1.020000
         25%           5.420000
         50%           8.570000
         75%          13.270000
         max        1424.070000
         Name: duration_min, dtype: float64
```

```
In [22]: # plotting duration in minutes on a standard scale
         bins = np.arange(0, 1450, 1)

         plt.figure(figsize = [8, 5])
         plt.hist(data = df, x = 'duration_min', bins = bins)
         plt.xlabel('Duration (min)')
         plt.xlim(0, 80)
         plt.show()
```




```
In [23]: np.log10(df['duration_min'].describe())

Out[23]: count    5.263428
         mean     1.082832
         std      1.475766
         min      0.008600
         25%      0.733999
         50%      0.932981
         75%      1.122871
         max      3.153531
         Name: duration_min, dtype: float64
```

```
In [24]: # there's a long tail in the distribution, so let's put it on a log scale instead
         log_binsize = 0.05
         bins = 10 ** np.arange(0, np.log10(df['duration_min'].max())+log_binsize, log_binsize)

         plt.figure(figsize = [8, 5])
         plt.hist(data = df, x = 'duration_min', bins = bins)
         plt.xscale('log')
         plt.xticks([3, 5, 10, 30, 60, 120, 240], [3, 5, 10, 30, 60, 120, 240])
         plt.xlabel('Duration (min)')
         plt.show()
```

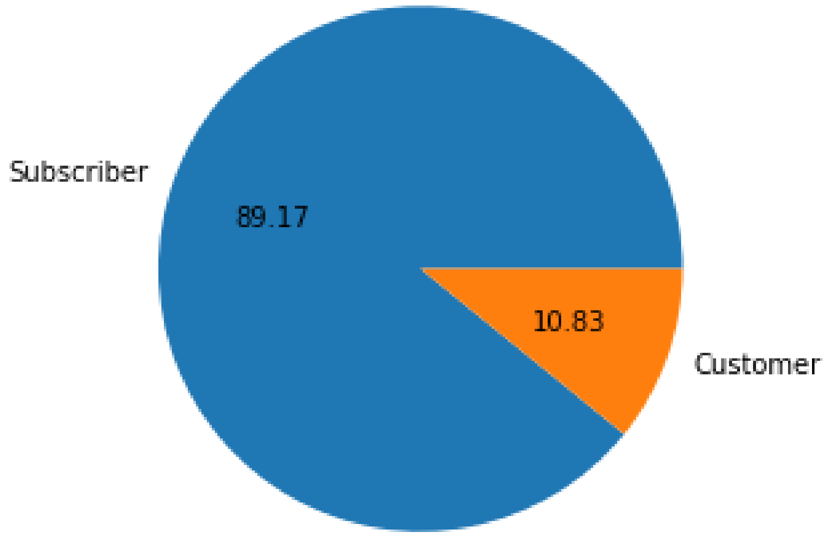


Duration in minutes has a long-tailed distribution, with most of the riders riding for around 10 minutes, and few over 30 minutes. When plotted on a log-scale, the price distribution looks unimodal, with one peak between 5 to 15 minutes.

Next, I'd like to look at user type.

```
In [25]: user_type = df['user_type'].value_counts()
```

```
In [26]: plt.pie(user_type, labels = user_type.index, autopct='%.2f')
         plt.axis('square');
```

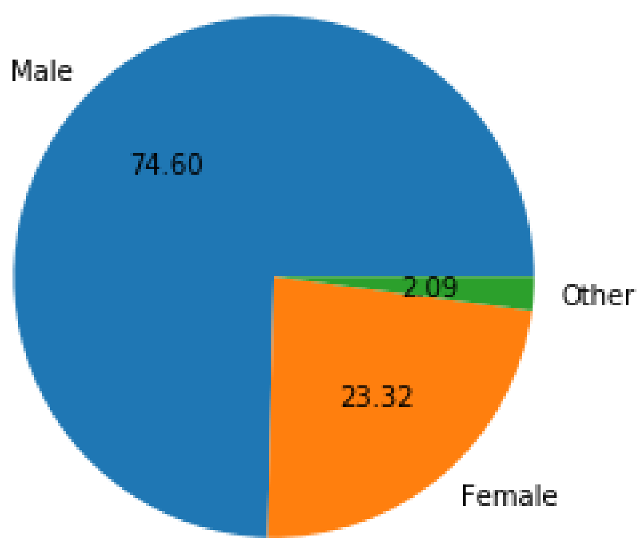


89.17% of users are subsribers, while 10.83% of them are normal customers. It means most of the users bought packages (daily package, monthly package and annual package)

Next I'll move on to variable 'member_gender'.

```
In [27]: gender = df['member_gender'].value_counts()
```

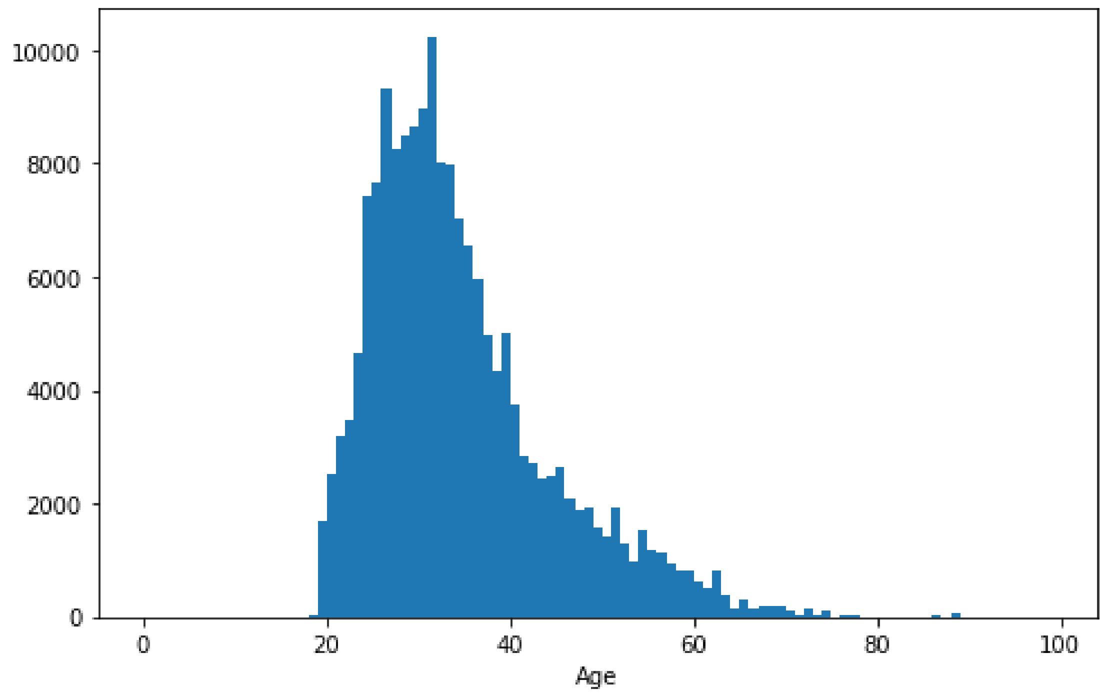
```
In [28]: plt.pie(gender, labels = gender.index, autopct='%.2f')
         plt.axis('square');
```



74.60% of the users are males, while 23.32% of them are females. It means most of the users are males.

Next I'll move on to users' age.

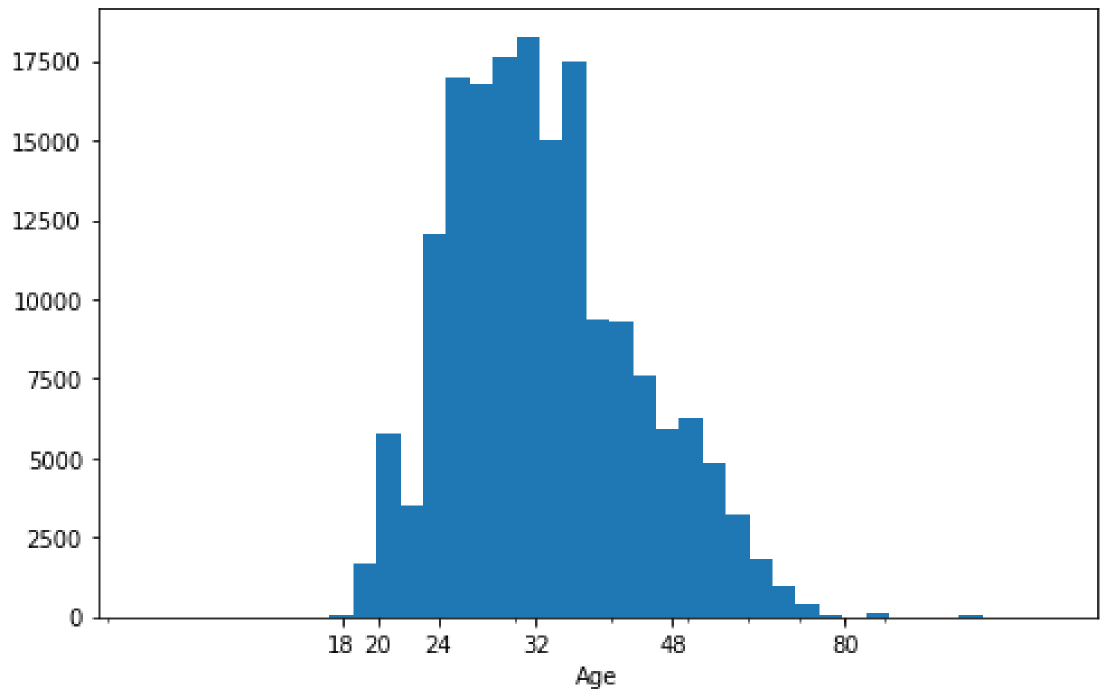
```
In [29]: # There are some data that has no information on users' age.
         no_age = df.query('age == "Nan"')

In [30]: no_age_index = no_age.index

In [31]: # to prepare a dataframe with only useful information in the column 'age'
         df_age = df.drop(no_age_index, axis = 0)

In [32]: df_age['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: age, dtype: float64

In [33]: bins = np.arange(0, 100, 1)

         plt.figure(figsize = [8, 5])
         plt.hist(data = df_age, x = 'age', bins = bins)
         plt.xlabel('Age')
         plt.show()
```



In [34]:
```
np.log10(df_age['age'].describe())
```

Out[34]:
```
count    5.243403
mean     1.533944
std      1.005038
min      1.255273
25%      1.431364
50%      1.505150
75%      1.591065
max      2.149219
Name: age, dtype: float64
```

In [35]:
```
# there's a long tail in the distribution, so let's put it on a log scale instead
log_binsize = 0.03
bins = 10 ** np.arange(1, np.log10(df_age['age'].max())+log_binsize, log_binsize)

plt.figure(figsize = [8, 5])
plt.hist(data = df_age, x = 'age', bins = bins)
plt.xscale('log')
plt.xticks([18, 20, 24, 32, 48, 80], [18, 20, 24, 32, 48, 80])
plt.xlabel('Age')
plt.show()
```



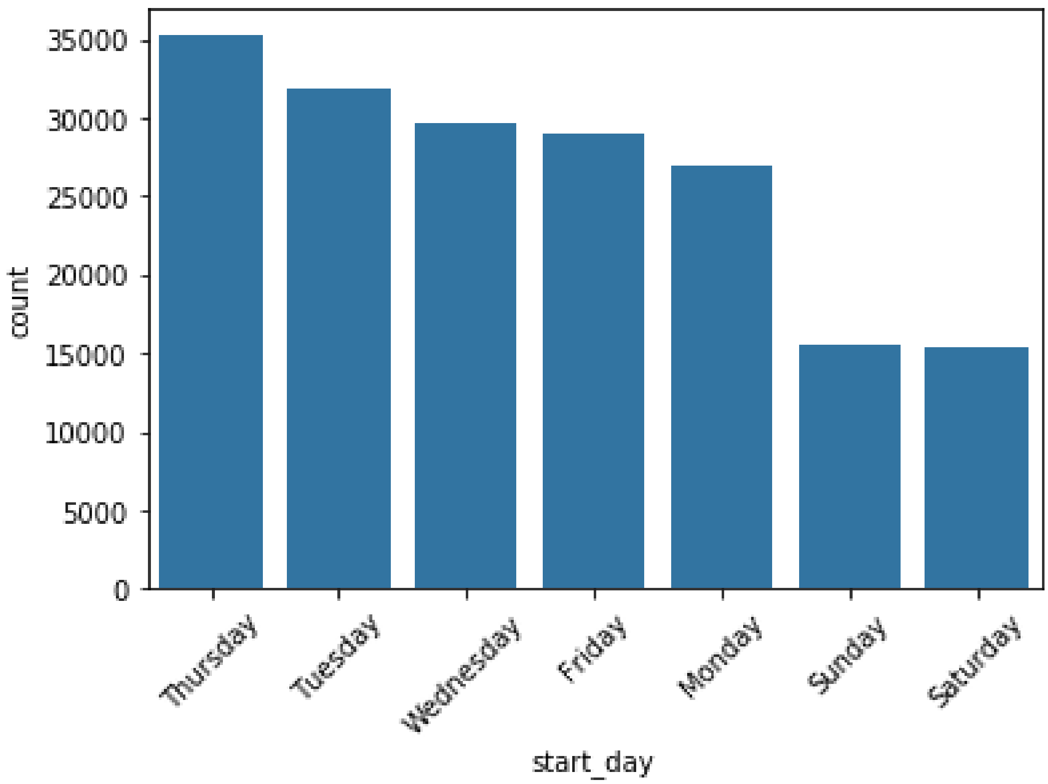
Age has a long-tailed distribution, with most of the riders at the age of 24 - 40. When plotted on a log-scale, the age distribution looks unimodal, with one peak between 24 - 38 years old.

Next up, I will look into the variable start_day.

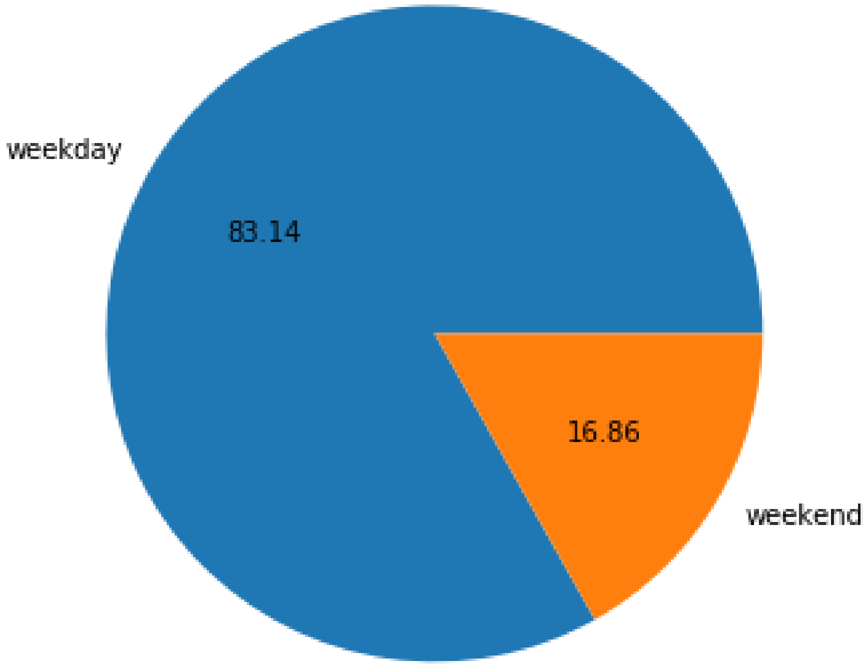
```
In [36]: df['start_day'].value_counts()
```

```
Out[36]: Thursday     35197
         Tuesday      31813
         Wednesday    29641
         Friday       28981
         Monday       26852
         Sunday       15523
         Saturday     15405
         Name: start_day, dtype: int64
```

```
In [37]: color = sb.color_palette()[0]
         order = df['start_day'].value_counts().index
         sb.countplot(data = df, x = 'start_day', order = order, color = color)
         plt.xticks(rotation = 45);
```



In [38]:
```
weekend = df['weekend'].value_counts()

plt.figure(figsize = (8,5))
plt.pie(weekend, labels = weekend.index, autopct='%.2f')
plt.axis('square');
```



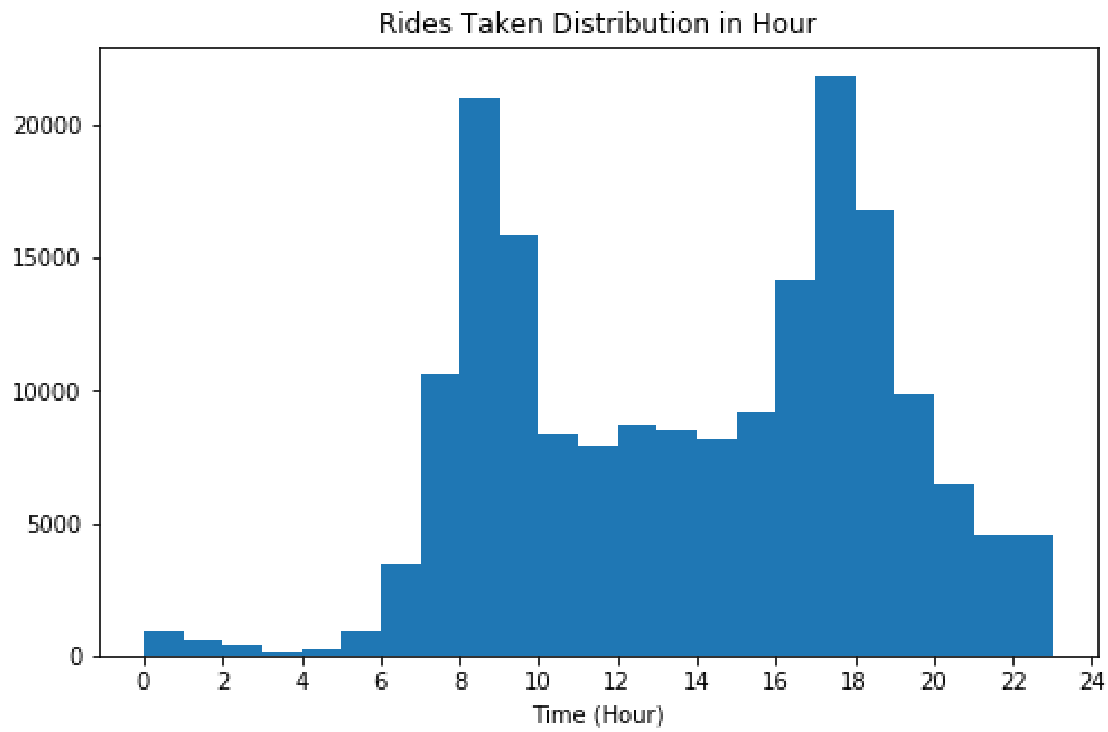
83.14% of users ride on weekdays.
Next I will move on to time of day.

```
In [39]: df['start_hour'].value_counts()
```

```
Out[39]: 17    21864
         8     21056
         18    16827
         9     15903
         16    14169
         7     10614
         19     9881
         15     9174
         12     8724
         13     8551
         10     8364
         14     8152
         11     7884
         20     6482
         21     4561
         6      3485
         22     2916
         23     1646
         0       925
```

```
5        896
1        548
2        381
4        235
3        174
Name: start_hour, dtype: int64
```

In [40]:
```python
bins = np.arange(0, 24, 1)

plt.figure(figsize = [8, 5])
plt.hist(data = df, x = 'start_hour', bins = bins)
plt.xlabel('Time (Hour)')
plt.xticks([0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24], [0, 2, 4, 6, 8, 10, 12, 14,
plt.title('Rides Taken Distribution in Hour')
plt.show()
```



There are two peaks in a day. Most users ride from 8 - 10, and 16 - 20, which are the usual time people get to work and go home.

Next I will move on to distance.

In [41]:
```python
no_distance_ind = df.query('distance == 0').index
```

In [42]:
```python
df_distance = df.drop(no_distance_ind, axis = 0)
```

In [43]:
```python
df['distance'].describe()
```

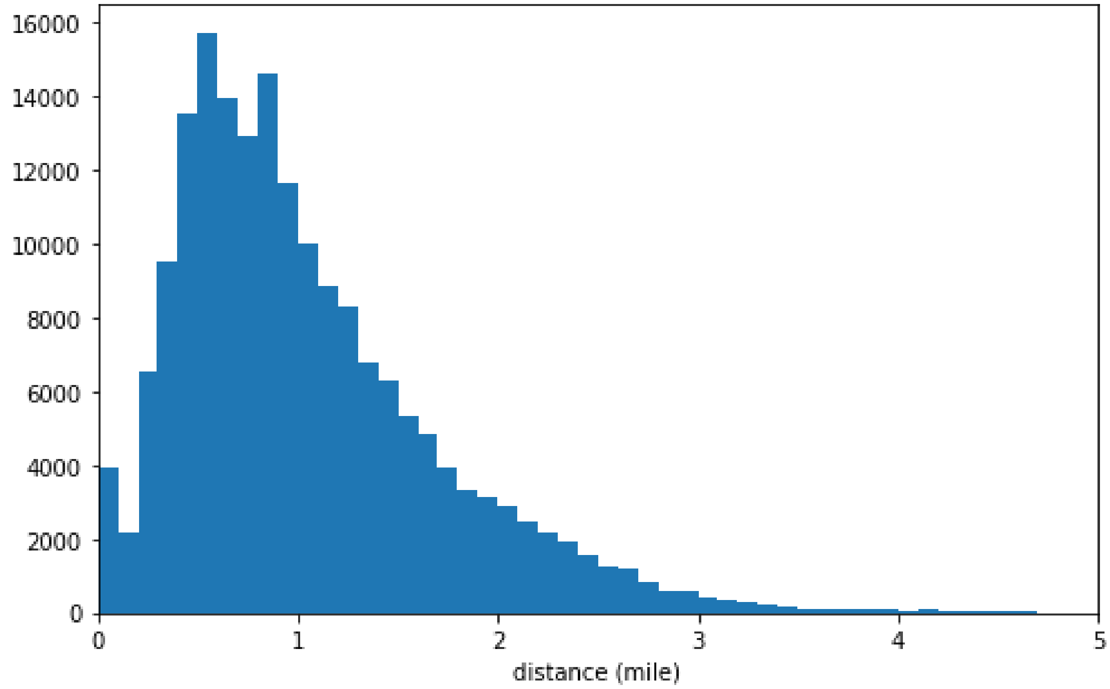
```
Out[43]: count    183412.000000
         mean          1.050462
         std           0.681742
         min           0.000000
         25%           0.565309
         50%           0.888503
         75%           1.383277
         max          43.164157
         Name: distance, dtype: float64
```

```
In [44]: df_distance_log = df.query('distance != 0')
```

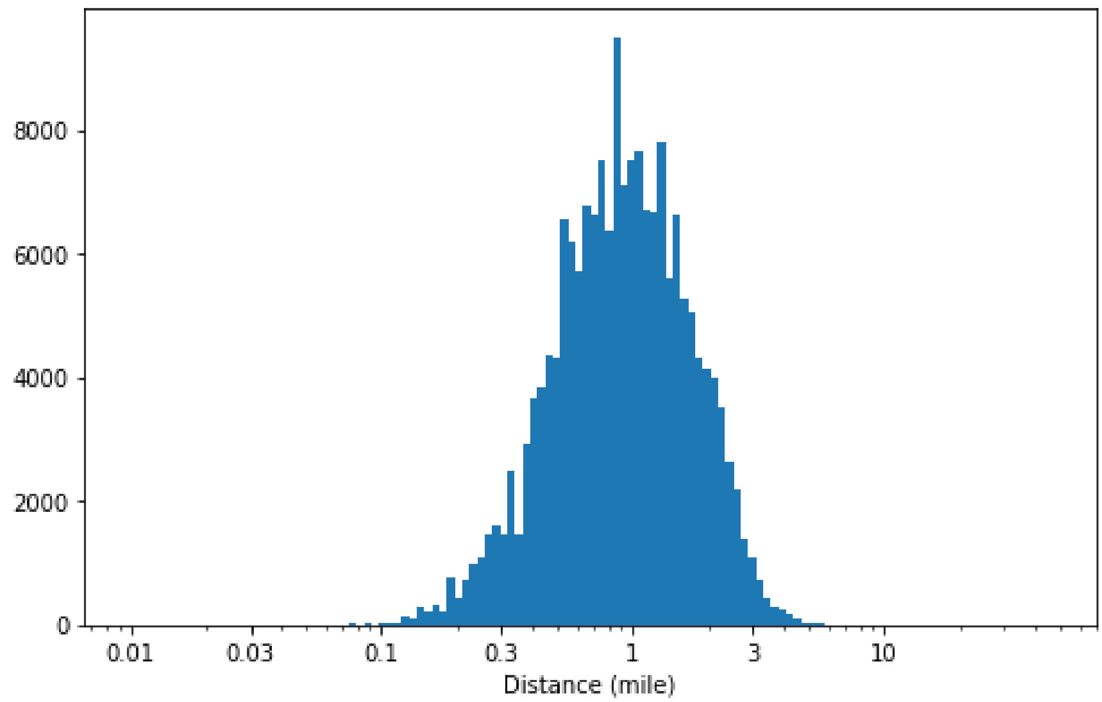
```
In [45]: np.log10(df_distance_log['distance'].describe())
```

```
Out[45]: count    5.254142
         mean     0.030666
         std     -0.173175
         min     -2.082567
         25%     -0.232732
         50%     -0.042743
         75%      0.146168
         max      1.635123
         Name: distance, dtype: float64
```

```
In [46]: bins = np.arange(0, 8, 0.1)

         plt.figure(figsize = [8, 5])
         plt.hist(data = df, x = 'distance', bins = bins)
         plt.xlabel('distance (mile)')
         plt.xlim(0, 5)
         plt.show()
```



In [47]:
```
# there's a long tail in the distribution, so let's put it on a log scale instead

log_binsize = 0.03
bins = 10 ** np.arange(-2, np.log10(df_age['distance'].max())+log_binsize, log_binsize)

plt.figure(figsize = [8, 5])
plt.hist(data = df_distance_log, x = 'distance', bins = bins)
plt.xscale('log')
plt.xticks([0.01, 0.03, 0.1, 0.3, 1, 3, 10], [0.01, 0.03, 0.1, 0.3, 1, 3, 10])
plt.xlabel('Distance (mile)')
plt.show()
```



Distance has a long-tailed distribution. When plotted on a log-scale, the distance distribution looks unimodal, with one peak between 0.5 to 2 miles.

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

The duration (in minutes) has a long range of values, so I have used a log transformation. It then looks unimodal, with one peak between 5 - 15 minutes.

The age also has a long range of values, so I have used a log transformation again. It looks unimodal with one peak between 24 - 38 years old. I have also cleaned the data that contains "NaN".

1.4.2 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?

When plotting distance in miles and duration in minutes, there are some outliers. So I used plt.xlim to limit the range we can see from the plot.

1.5 Bivariate Exploration

```
In [48]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
```

```
Data columns (total 25 columns):
duration_sec               183412 non-null int64
start_time                 183412 non-null datetime64[ns]
end_time                   183412 non-null datetime64[ns]
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
duration_min               183412 non-null float64
start_day                  183412 non-null object
end_day                    183412 non-null object
age                        175147 non-null float64
start_coords               183412 non-null object
end_coords                 183412 non-null object
distance                   183412 non-null float64
start_hour                 183412 non-null int64
weekend                    183412 non-null object
dtypes: datetime64[ns](2), float64(10), int64(3), object(10)
memory usage: 35.0+ MB
```

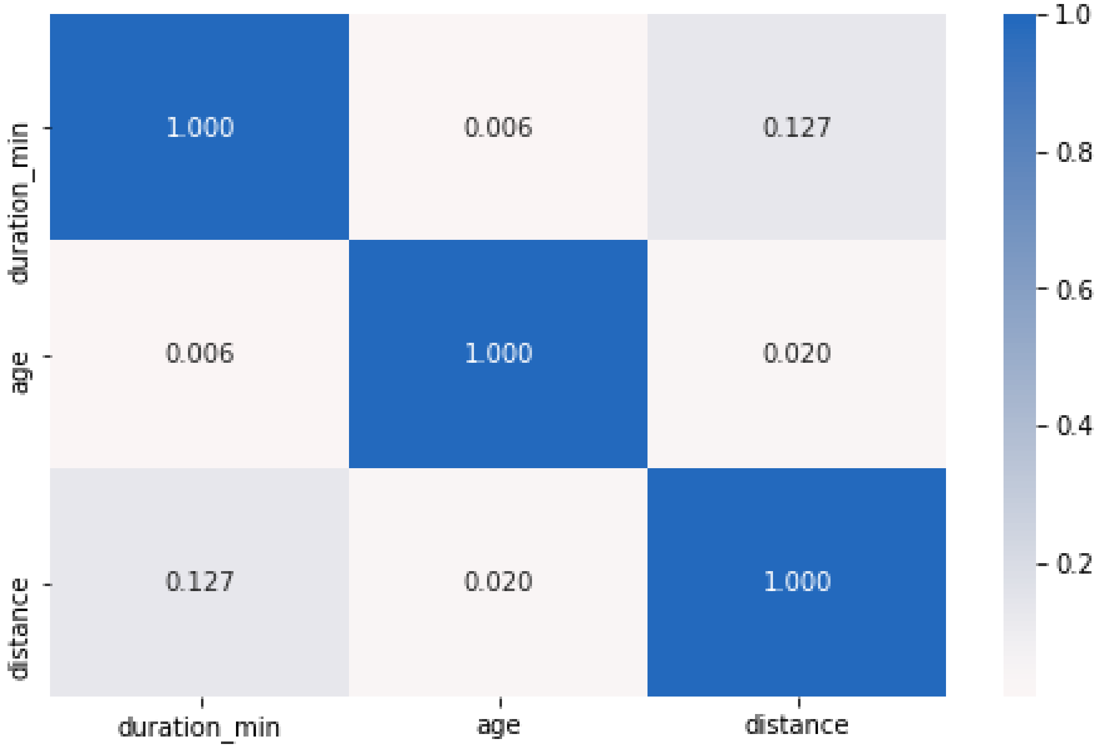
In [49]:
```
df.head()
```

Out[49]:
```
   duration_sec              start_time                end_time  \
0         52185 2019-02-28 17:32:10.145 2019-03-01 08:01:55.975
1         42521 2019-02-28 18:53:21.789 2019-03-01 06:42:03.056
2         61854 2019-02-28 12:13:13.218 2019-03-01 05:24:08.146
3         36490 2019-02-28 17:54:26.010 2019-03-01 04:02:36.842
4          1585 2019-02-28 23:54:18.549 2019-03-01 00:20:44.074

   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

   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

                              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

    ...     bike_share_for_all_trip  duration_min start_day  end_day   age  \
0   ...                          No        869.75  Thursday   Friday  35.0
1   ...                          No        708.68  Thursday   Friday   NaN
2   ...                          No       1030.90  Thursday   Friday  47.0
3   ...                          No        608.17  Thursday   Friday  30.0
4   ...                         Yes         26.42  Thursday   Friday  45.0

                              start_coords  \
0             (37.7896254, -122.400811)
1      (37.791464000000005, -122.391034)
2       (37.76930529999999, -122.4268256)
3  (37.77483629413345, -122.44654566049576)
4   (37.8045623549303, -122.27173805236816)

                                end_coords  distance start_hour  weekend
0      (37.794230999999996, -122.402923)  0.338015         17  weekday
1                   (37.77588, -122.39317)  1.081130         18  weekday
2  (37.78637526861584, -122.40490436553955)  1.681051         12  weekday
3  (37.77331087889723, -122.44429260492323)  0.162113         17  weekday
4       (37.7927143, -122.24877959999999)  1.498758         23  weekday

[5 rows x 25 columns]
```

In [50]:
```
numeric_vars = ['duration_min', 'age', 'distance']
categoric_vars = ['start_day', 'start_hour', 'member_gender', 'user_type']
```

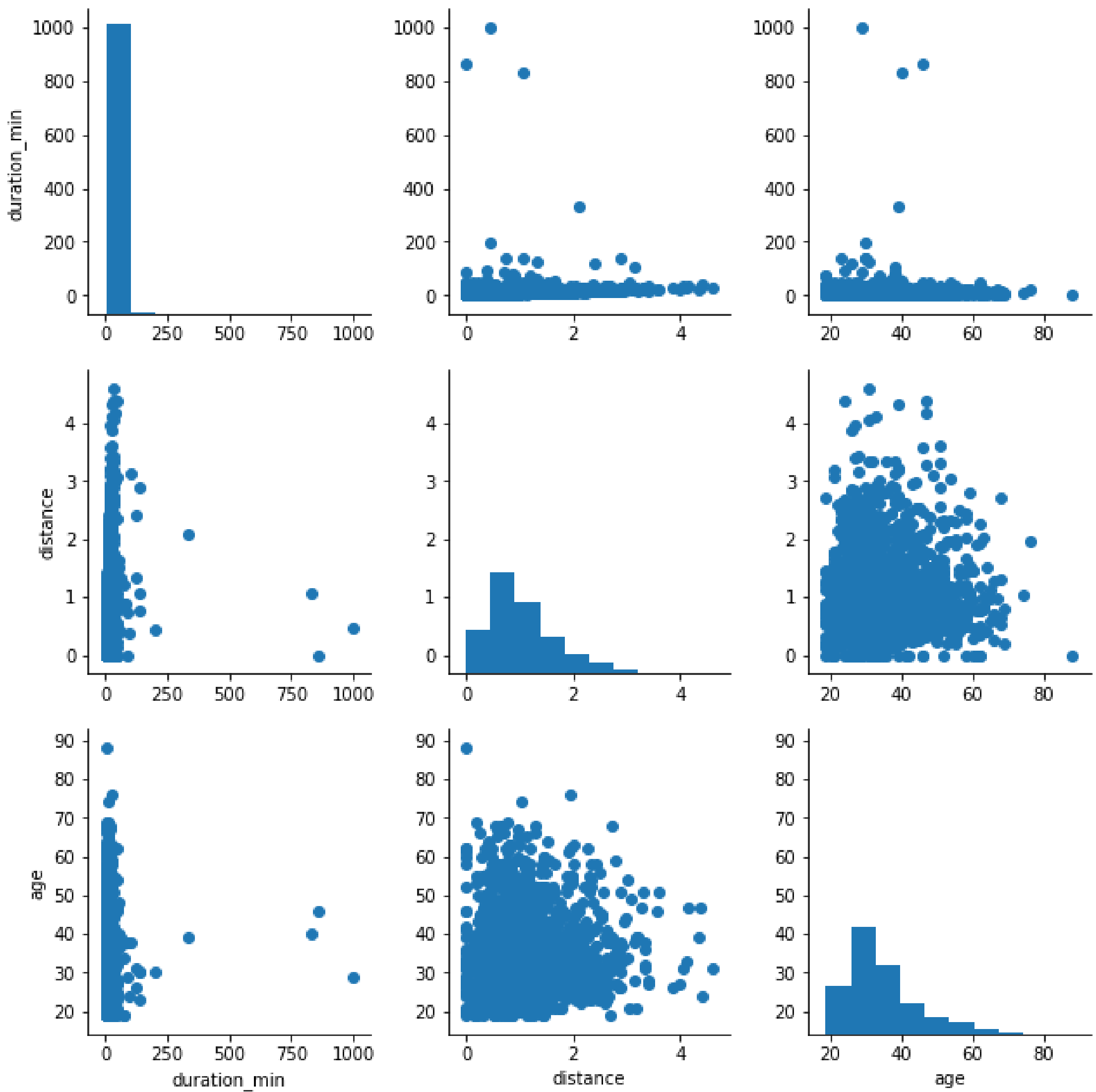
In [51]:
```
# correlation plot
plt.figure(figsize = [8, 5])
sb.heatmap(df[numeric_vars].corr(), annot = True, fmt = '.3f',
           cmap = 'vlag_r', center = 0)
plt.show()
```



In [52]:
```
# plot matrix of all numeric features

df_1 = df.query('age != "Nan"')
```

In [53]:
```
df_samp = df_1.sample(n=2000, replace = False)


g = sb.PairGrid(data = df_samp, vars = ['duration_min', 'distance', 'age'], size = 3)
g = g.map_diag(plt.hist)
g.map_offdiag(plt.scatter)
```

Out[53]: <seaborn.axisgrid.PairGrid at 0x7ff4f6667a58>



Duration, distance and age of users don't have any strong correlation.

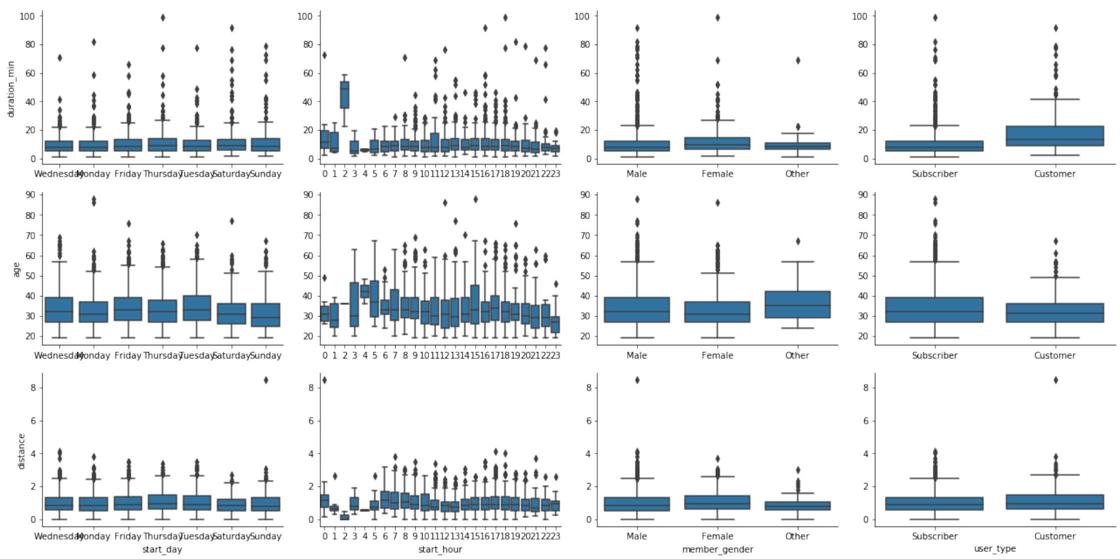
Next I'd like to get a plot matrix of numeric features against categorical features.

Since there are some outliers for duration_min, I'd like to focus on the range of 0 minute to 100 minutes.

```
In [54]: df_1 = df.query('duration_min < 100')

In [55]: # plot matrix of numeric features against categorical features.

         df_samp1 = df_1.sample(n=2000, replace = False)


         def boxgrid(x, y, **kwargs):
             default_color = sb.color_palette()[0]
```

```
        sb.boxplot(x=x, y=y, color=default_color)

    plt.figure(figsize = [10, 10])
    g = sb.PairGrid(data = df_samp1, y_vars = numeric_vars, x_vars = categoric_vars,
                    size = 3, aspect = 1.5, dropna = True)
    g.map(boxgrid)
    plt.show();
```

```
<matplotlib.figure.Figure at 0x7ff4fa66f0f0>
```



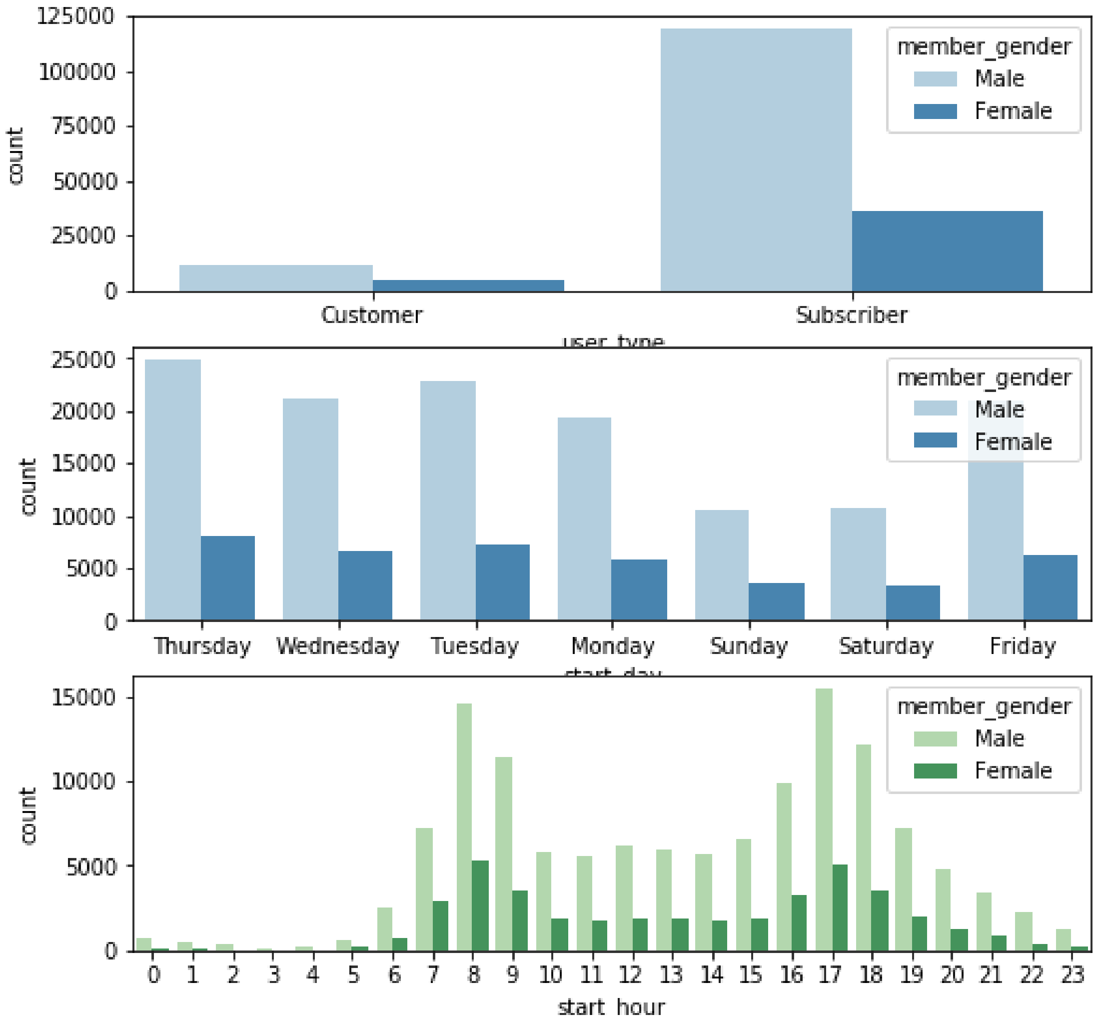
As we are more interested in the difference between 'male' and 'female', I will remove the rows with 'other' gender. Subscribers have a shorter average trip duration and average trip distance than normal customers. Let's say the busy hours in the morning is 6am to 8am. Although the distance of trips from 6am to 8am is relatively longer, the duration is not relatively higher. It suggested that the riders might be riding in a faster pace during the busy hours.

Next, I'd move on to plots of categorical variables.

```
In [54]: df_2 = df.query('member_gender != "Other"')
```

```
In [55]: # since there's only three subplots to create, using the full data should be fine.
         plt.figure(figsize = [8, 8])

         # subplot 1: user_type vs member_gender
         plt.subplot(3, 1, 1)
         sb.countplot(data = df_2, x = 'user_type', hue = 'member_gender', palette = 'Blues')

         # subplot 2: start_day vs member_gender
         ax = plt.subplot(3, 1, 2)
         sb.countplot(data = df_2, x = 'start_day', hue = 'member_gender', palette = 'Blues')
```

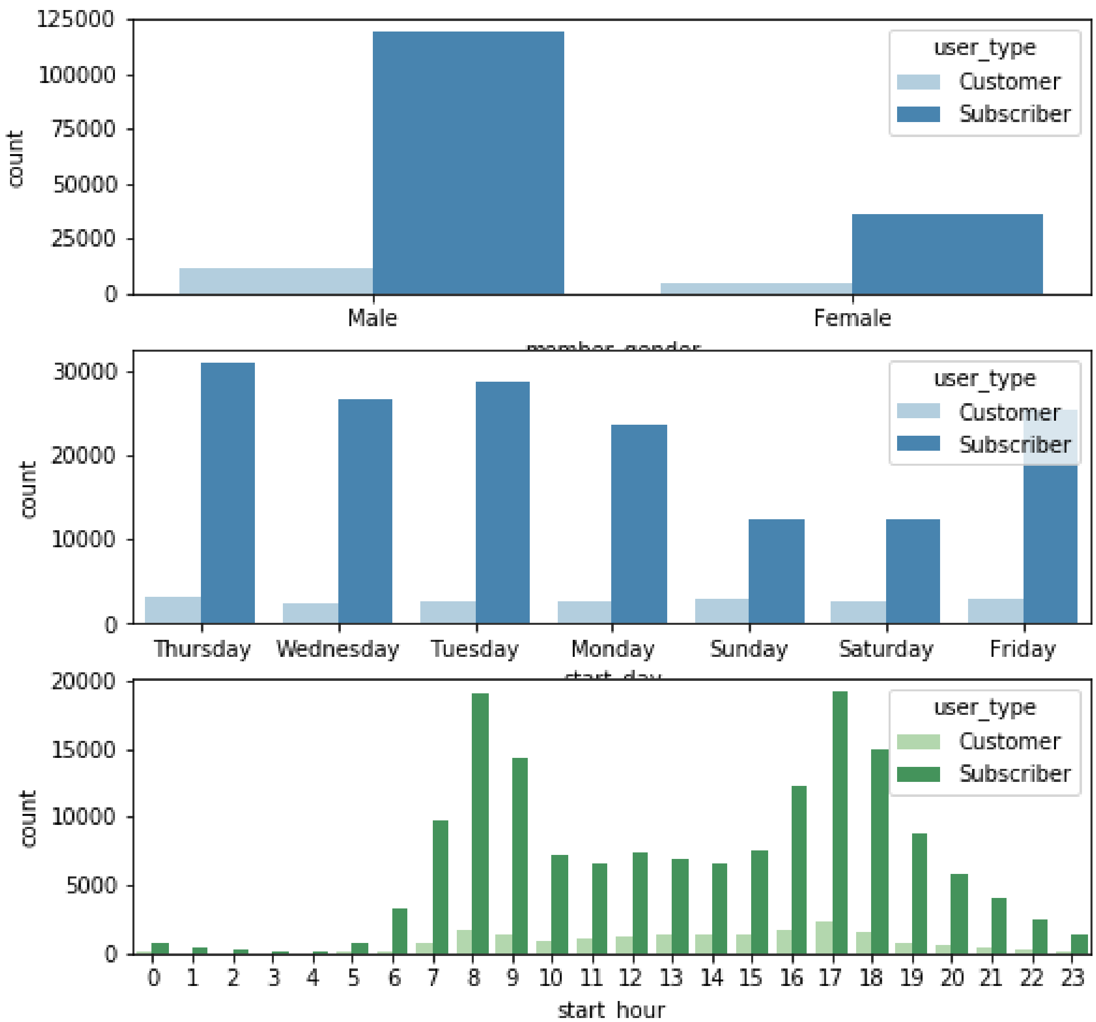
```
# subplot 3: start_hour vs member_gender
ax = plt.subplot(3, 1, 3)
sb.countplot(data = df_2, x = 'start_hour', hue = 'member_gender', palette = 'Greens')

plt.show()
```



Most of the subscribers are male. Their activities are much more higher than females.

In [56]:
```
# since there's only three subplots to create, using the full data should be fine.
plt.figure(figsize = [8, 8])

# subplot 1: member_gender vs user_type, this is the same as the first subplot on the p
plt.subplot(3, 1, 1)
sb.countplot(data = df_2, x = 'member_gender', hue = 'user_type', palette = 'Blues')
```

```
# subplot 2: start_day vs user_type
ax = plt.subplot(3, 1, 2)
sb.countplot(data = df_2, x = 'start_day', hue = 'user_type', palette = 'Blues')

# subplot 3: start_hour vs user_type, use different color palette
ax = plt.subplot(3, 1, 3)
sb.countplot(data = df_2, x = 'start_hour', hue = 'user_type', palette = 'Greens')

plt.show()
```



Most of the users are subscribers. Their activities are much more higher than normal customers.

Next, I'd like to see how duration_min and distance are related to one another for all of the data

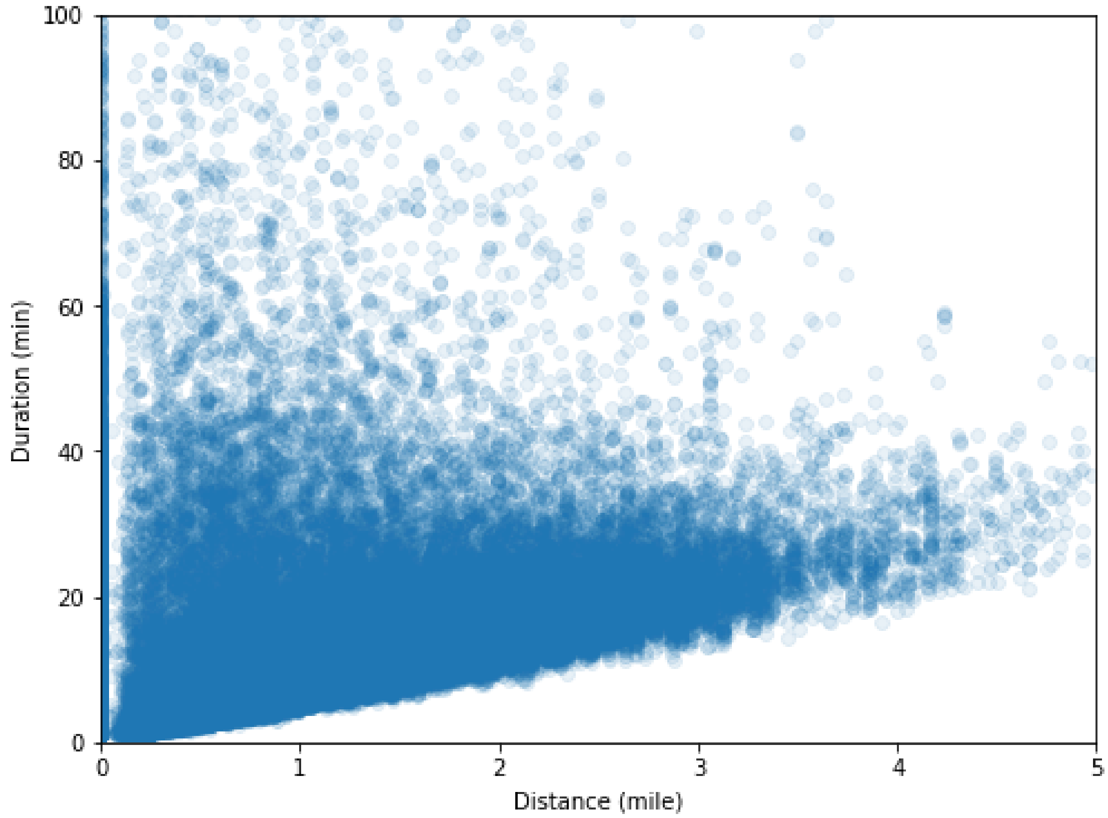
```
In [57]: df['duration_min'].describe()
```

```
Out[57]: count    183412.000000
         mean         12.101301
         std          29.906501
         min           1.020000
         25%           5.420000
         50%           8.570000
         75%          13.270000
         max        1424.070000
         Name: duration_min, dtype: float64
```

```
In [59]: df['distance'].describe()
```

```
Out[59]: count    183412.000000
         mean          1.050462
         std           0.681742
         min           0.000000
         25%           0.565309
         50%           0.888503
         75%           1.383277
         max          43.164157
         Name: distance, dtype: float64
```

```
In [61]: # scatter plot of duration_min vs distance

         plt.figure(figsize = [8, 6])
         plt.scatter(data = df, x = 'distance', y = 'duration_min', alpha = 1/10)
         plt.xlim([0, 5])
         plt.ylim([0, 100])
         plt.xlabel('Distance (mile)')
         plt.ylabel('Duration (min)')
         plt.show()
```



I thought there would be a strong correlation between distance and duration. However, the only conclusion I could draw is that the longer the distance (the further away from start station to end station), the less probable that the duration could be relatively short. It means that it would take a minimum amount of time to return the bike at an end station with certain amount of distance. We would only be confident in determining the minimum duration, but we could not be confident in determining the average duration.

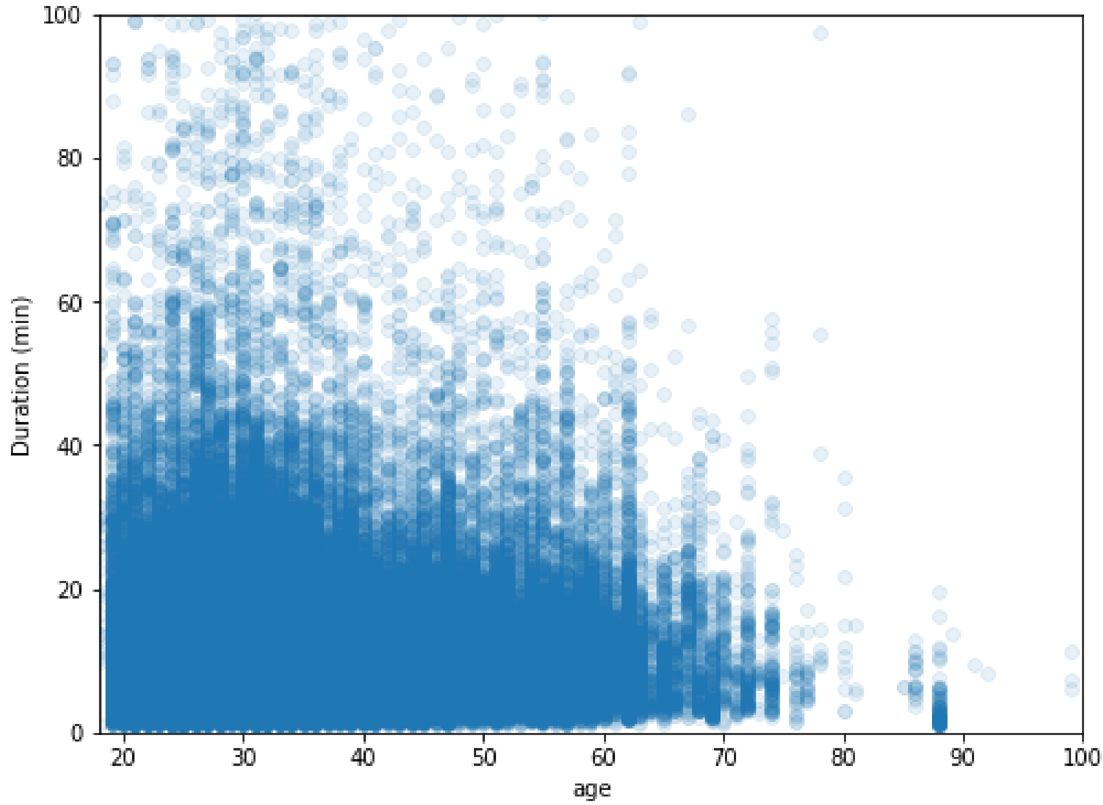
Next, I'd move on to duration_min vs age.

```
In [62]: df['duration_min'].describe()
```

```
Out[62]: count    183412.000000
         mean         12.101301
         std          29.906501
         min           1.020000
         25%           5.420000
         50%           8.570000
         75%          13.270000
         max        1424.070000
         Name: duration_min, dtype: float64
```

```
In [63]: # scatter plot of duration_min vs age.
         plt.figure(figsize = [8, 6])
```

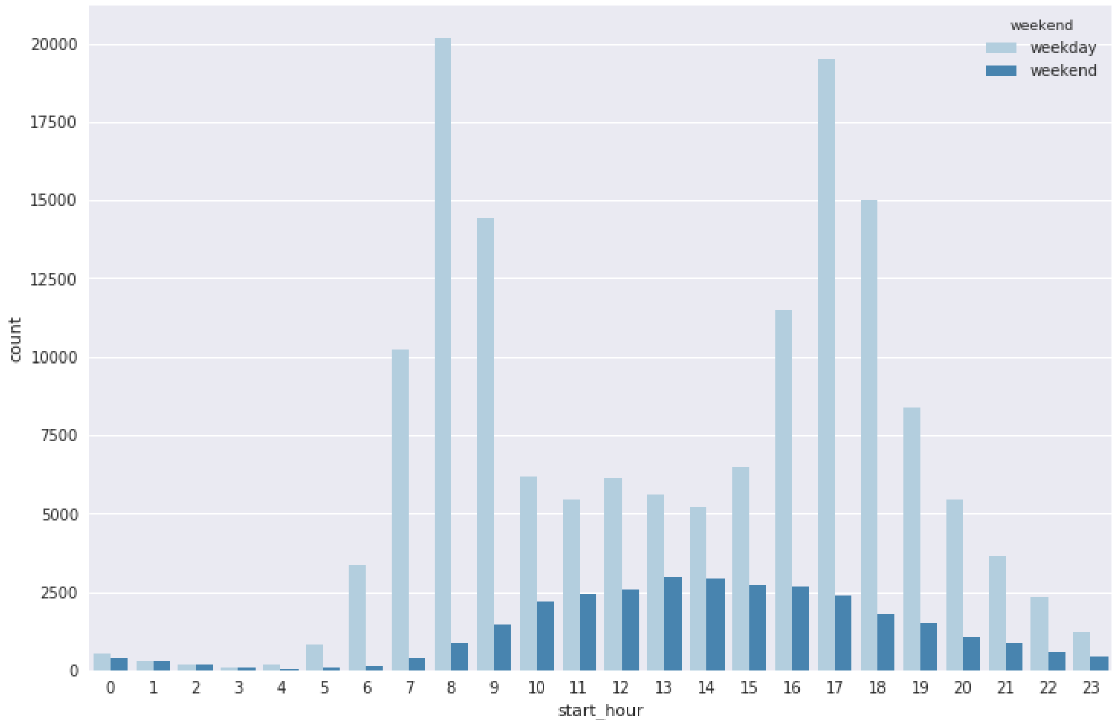
```
plt.scatter(data = df, x = 'age', y = 'duration_min', alpha = 1/10)
plt.xlabel('age')
# plt.yticks([1, 3, 10, 30, 100, 300], [1, 3, 10, 30, 100, 300])
plt.ylabel('Duration (min)')
plt.xlim([18, 100])
plt.ylim([0, 100])
plt.show()
```



The users with older ages, their trip duration would tend to decrease.
Next I'd move on to start hour vs weekday/weekend

In [116]:
```
plt.figure(figsize = [12, 8])

sb.countplot(data = df, x = 'start_hour', hue = 'weekend', palette = 'Blues')
```

Out[116]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcc907e2e80>



For weekday, let's say the busy hours are 6am to 8am and 4pm to 7pm. There are two peaks during these two periods. But for weekend, the busy hours only start from 10am until 6pm. There is only one peak, with much less activity compared with weekdays.

1.5.1 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?

Subscribers have a shorter average trip duration and average trip distance than normal customers.

Most of the subscribers are male. Their activities are much more higher than females.

Most of the users are subscribers. Their activities are much more higher than normal customers.

The longer the distance (the further away from start station to end station), the less probable that the duration could be relatively short. It means that it would take a minimum amount of time to return the bike at an end station with certain amount of distance. We would only be confident in determining the minimum duration, but we could not be confident in determining the average duration.

The users with older ages, their trip duration would tend to decrease.

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

Duration, distance and age of users don't have any strong correlation.

Let's say the busy hours in the morning is 6am to 8am. Although the distance of trips from 6am to 8am is relatively longer, the duration is not relatively higher. It suggested that the riders might be riding in a faster pace during the busy hours.

1.6 Multivariate Exploration

I would like to look at the top 10 spots people would like to start and end a bike trip on weekdays and weekends.

```
In [65]: # to use marker on a map, I'd like to use the Folium module.
         !pip install folium
```

```
Collecting folium
  Downloading https://files.pythonhosted.org/packages/b9/05/bb30dc97efa1b431c88deac7a77af3d62df1
    100% || 102kB 4.8MB/s a 0:00:011
Collecting branca>=0.3.0 (from folium)
  Downloading https://files.pythonhosted.org/packages/6c/e2/16ce27dbfbc48b460e95aa2e900e905d3f10
Requirement already satisfied: jinja2>=2.9 in /opt/conda/lib/python3.6/site-packages (from foliu
Requirement already satisfied: numpy in /opt/conda/lib/python3.6/site-packages (from folium) (1.
Requirement already satisfied: requests in /opt/conda/lib/python3.6/site-packages (from folium)
Requirement already satisfied: MarkupSafe>=0.23 in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /opt/conda/lib/python3.6/site-packages (
Requirement already satisfied: idna<2.7,>=2.5 in /opt/conda/lib/python3.6/site-packages (from re
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /opt/conda/lib/python3.6/site-packages (
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.6/site-packages (fro
Installing collected packages: branca, folium
Successfully installed branca-0.5.0 folium-0.12.1.post1
```

```
In [66]: import folium
```

In the following steps, I'll create some relavant dataframes and index

```
In [67]: # to get a dataframe contains only information we are interested
         stops = df[['start_station_id', 'start_station_latitude', 'start_station_longitude', 'e

In [68]: # to get a dataframe that contains only information about weekdays
         stops_weekday = stops.query('start_day != "Saturday" and start_day != "Sunday"')

In [69]: # to get the top 10 starting spots on weekday
         index_start_weekday = stops_weekday['start_station_id'].value_counts().index[:10]

In [70]: top_start_weekday = stops_weekday[stops_weekday['start_station_id'].isin(index_start_we

In [71]: # to drop the duplicates as we only need the coordinates of the station spots
         top_start_weekday = top_start_weekday.drop_duplicates(subset = ['start_station_id'])

In [ ]:

In [72]: # to get the top 10 ending spots on weekday
         index_end_weekday = stops_weekday['end_station_id'].value_counts().index[:10]

In [73]: top_end_weekday = stops_weekday[stops_weekday['end_station_id'].isin(index_end_weekday)
```

In [74]:
```
# to drop the duplicates as we only need the coordinates of the station spots
top_end_weekday = top_end_weekday.drop_duplicates(subset = ['end_station_id'])
```

In [ ]:

In [93]:
```
# to get a dataframe that contains only information about weekend
stops_weekend = stops.query('start_day == "Saturday" or start_day == "Sunday"')
```

In [94]:
```
# to get the top 10 starting spots on weekend
index_start_weekend = stops_weekend['start_station_id'].value_counts().index[:10]
```

In [95]:
```
top_start_weekend = stops_weekend[stops_weekend['start_station_id'].isin(index_start_we
```

In [96]:
```
# to drop the duplicates as we only need the coordinates of the station spots
top_start_weekend = top_start_weekend.drop_duplicates(subset = ['start_station_id'])
```

In [ ]:

In [97]:
```
# to get the top 10 ending spots on weekend
index_end_weekend = stops_weekend['end_station_id'].value_counts().index[:10]
```

In [98]:
```
top_end_weekend = stops_weekend[stops_weekend['end_station_id'].isin(index_end_weekend)
```

In [99]:
```
# to drop the duplicates as we only need the coordinates of the station spots
top_end_weekend = top_end_weekend.drop_duplicates(subset = ['end_station_id'])
```

In [ ]:

In [85]:
```
# to create a map for weekday top spots
sfmap_weekday = folium.Map(location = [top_start_weekday['start_station_latitude'].mean
```

In [86]:
```
# to mark the top 10 starting spots on weekdays
for i, v in top_start_weekday.iterrows():
    folium.Marker(location = [v['start_station_latitude'], v['start_station_longitude']
```

In [87]:
```
# to mark the top 10 ending spots on weekdays
for i, v in top_end_weekday.iterrows():
    folium.Marker(location = [v['end_station_latitude'], v['end_station_longitude']]).a
```

In [88]:
```
# this is the top 10 starting and top 10 ending bike trip spots on weekdays.
sfmap_weekday
```

Out[88]:
```
<folium.folium.Map at 0x7fcc8e64c550>
```

In [100]:
```
# to create a map for weekend top spots
sfmap_weekend = folium.Map(location = [top_start_weekend['start_station_latitude'].mea
```

In [101]:
```
# to mark the top 10 starting spots on weekends
for i, v in top_start_weekend.iterrows():
    folium.Marker(location = [v['start_station_latitude'], v['start_station_longitude'
```

```
In [102]: # to mark the top 10 ending spots on weekends
          for i, v in top_end_weekend.iterrows():
              folium.Marker(location = [v['end_station_latitude'], v['end_station_longitude']]).
```

```
In [103]: # this is the top 10 starting and top 10 ending bike trip spots on weekends.
          sfmap_weekend
```

```
Out[103]: <folium.folium.Map at 0x7fcc8e5f6e48>
```

There are some overlapping top spots when we consider the top 10 starting and top 10 ending spots on weekdays and weekends respectively. Most of them are near the San Francisco downtown.

1.6.1 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?

During the weekdays, the top 10 spots would be centred around the San Francisco downton. But the top 10 spots would be slightly more spreaded for weekends.

1.6.2 Were there any interesting or surprising interactions between features?

I thought the spots for weekend would be much more spreaded. But indeed there are a lot of overlapping top spots, even for weekends.

1.7 Conclusions

This data set includes information about individual rides made in a bike-sharing system covering the greater San Francisco Bay area. I have further wrangled the data by changing the columns into the suitable data type, such as having the start_time in datetime data type. I have also added new columns like duration_min (in minutes), age of users, distance calculated by coordinates, start_hour from start_time, start day and weekend.

Here is a summary of the findings from the study:

Finding 1: Most of the riders take a ride with 5 to 15 minutes. 15 minutes would already be over 75% percentile (13 minutes).

Finding 2: 89.17% of users are subscribers, while 10.83% of them are normal customers.

Finding 3: 74.60% of the users are males, while 23.32% of them are females.

Finding 4: The average age of the users is 34.2.

Finding 5: 83.14% of users ride on weekdays.

Finding 6: There are two peaks in a day. Most users ride from 8 - 10, and 16 - 20, which are the usual time people get to work and go home.

Finding 7: Most of the rides have an average distance of 1 mile.

Finding 8: Duration, distance and age of users don't have any strong correlation.

Finding 9: Subscribers have a shorter average trip duration and average trip distance than normal customers.

Finding 10: The

Finding 11: Most of the subscribers are male. Their activities are much more higher than females.

Finding 12: Most of the users are subscribers. Their activities are much more higher than normal customers.

Finding 13: The longer the distance (the further away from start station to end station), the less probable that the duration could be relatively short. It means that it would take a minimum amount of time to return the bike at an end station with certain amount of distance. We would only be confident in determining the minimum duration, but we could not be confident in determining the average duration.

Finding 14: The users with older ages, their trip duration would tend to decrease.

Finding 15: I have identified the top 10 starting and ending spots on weekdays and weekends respectively. Most of them are near the San Francisco downtown.