Part_I_exploration

November 8, 2022

1 Part I - Ford Go Bike Trip Data

1.1 by Mustafe Mohamed Abdulahi

1.2 Introduction

This data set contains a single csv file and consists of information about individual bike-sharing system covering the greater San Francisco Bay area. The data features include tripduration (secs), start_time, end_time, user information i.e (user_type, age), and some other variable.

1.3 Preliminary Wrangling

[1]:
```python
# import all packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import datetime as dt
from datetime import datetime
plt.style.use('ggplot')
%matplotlib inline
```

[2]:
```python
# load the dataset into a pandas dataframe
df = pd.read_csv("fordgobiketripdata.csv")
```

[3]:
```python
# show the top 5 records
df.head(5)
```

[3]:
```
   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
```

[4]:
```python
# Identify missing Values
missing_data = df.isnull()
missing_data.head()
```

[4]:
```
   duration_sec  start_time  end_time  start_station_id  start_station_name  \
0         False       False     False             False               False
1         False       False     False             False               False
2         False       False     False             False               False
3         False       False     False             False               False
4         False       False     False             False               False

   start_station_latitude  start_station_longitude  end_station_id  \
0                   False                    False           False
1                   False                    False           False
2                   False                    False           False
```

```
3                   False                        False            False
4                   False                        False            False

   end_station_name  end_station_latitude  end_station_longitude  bike_id  \
0             False                 False                  False    False
1             False                 False                  False    False
2             False                 False                  False    False
3             False                 False                  False    False
4             False                 False                  False    False

   user_type  member_birth_year  member_gender  bike_share_for_all_trip
0      False              False          False                    False
1      False               True           True                    False
2      False              False          False                    False
3      False              False          False                    False
4      False              False          False                    False
```

[5]:

```
# count the missing Values
for c in missing_data.columns.values.tolist():
    print(c)
    print(missing_data[c].value_counts())
    print("")
```

```
duration_sec
False    183412
Name: duration_sec, dtype: int64

start_time
False    183412
Name: start_time, dtype: int64

end_time
False    183412
Name: end_time, dtype: int64

start_station_id
False    183215
True        197
Name: start_station_id, dtype: int64

start_station_name
False    183215
True        197
Name: start_station_name, dtype: int64

start_station_latitude
False    183412
```

```
Name: start_station_latitude, dtype: int64

start_station_longitude
False    183412
Name: start_station_longitude, dtype: int64

end_station_id
False    183215
True        197
Name: end_station_id, dtype: int64

end_station_name
False    183215
True        197
Name: end_station_name, dtype: int64

end_station_latitude
False    183412
Name: end_station_latitude, dtype: int64

end_station_longitude
False    183412
Name: end_station_longitude, dtype: int64

bike_id
False    183412
Name: bike_id, dtype: int64

user_type
False    183412
Name: user_type, dtype: int64

member_birth_year
False    175147
True       8265
Name: member_birth_year, dtype: int64

member_gender
False    175147
True       8265
Name: member_gender, dtype: int64

bike_share_for_all_trip
False    183412
Name: bike_share_for_all_trip, dtype: int64
```

1.3.1 Observation on missing data

start_station_id: 197 Missing data

start_station_name: 197 Missing data

end_station_id: 197 Missing data

end_station_name: 197 Missing data

member_birth_year: 8265 Missing data

member_gender: 8265 Missing data

1.3.2 Deal with missing data

How we do deal with missing data? Well, we have two way to deal with it. 1. drop data - drop the whole row or column: Choose this option if most entries in the column are empty or you don't need the columns. 2. Replace by the mean, frequence or other function.

drop these columns `start_station_id`, `start_station_name`, `end_station_id`, `end_station_name`since we don't them in our analysis.

[6]:
```
df.drop(["start_station_id", "start_station_name", "end_station_id",␣
↪"end_station_name"], axis = 1, inplace = True)
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 12 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_latitude   183412 non-null  float64
 4   start_station_longitude  183412 non-null  float64
 5   end_station_latitude     183412 non-null  float64
 6   end_station_longitude    183412 non-null  float64
 7   bike_id                  183412 non-null  int64
 8   user_type                183412 non-null  object
 9   member_birth_year        175147 non-null  float64
 10  member_gender            175147 non-null  object
 11  bike_share_for_all_trip  183412 non-null  object
dtypes: float64(5), int64(2), object(5)
memory usage: 16.8+ MB
```

[8]:
```
# Check for duplicates
df.duplicated().sum()
```

[8]: 0

[9]:
```
# How many unique Id do we have in the dataset
df.bike_id.nunique()
```

[9]: 4646

[10]:
```
# Check for stats
df.describe()
```

[10]:
```
        duration_sec  start_station_latitude  start_station_longitude  \
count  183412.000000           183412.000000            183412.000000
mean      726.078435               37.771223              -122.352664
std      1794.389780                0.099581                 0.117097
min        61.000000               37.317298              -122.453704
25%       325.000000               37.770083              -122.412408
50%       514.000000               37.780760              -122.398285
75%       796.000000               37.797280              -122.286533
max     85444.000000               37.880222              -121.874119

       end_station_latitude  end_station_longitude        bike_id  \
count         183412.000000          183412.000000  183412.000000
mean              37.771427            -122.352250    4472.906375
std                0.099490               0.116673    1664.383394
min               37.317298            -122.453704      11.000000
25%               37.770407            -122.411726    3777.000000
50%               37.781010            -122.398279    4958.000000
75%               37.797320            -122.288045    5502.000000
max               37.880222            -121.874119    6645.000000

       member_birth_year
count      175147.000000
mean         1984.806437
std            10.116689
min          1878.000000
25%          1980.000000
50%          1987.000000
75%          1992.000000
max          2001.000000
```

1.4 Data Quality

1. Remove unwanted columns
2. Correct erroneous data types for (start_time, end_time, bike_id, and user_type)

1.5 Cleaning

1.5.1 Define:

Drop unwanted columns:`start_station_latitude`,`start_station_longitude`, `end_station_latitude`, `end_station_longitude`

1.5.2 Code

[11]:
```
#  drop unwanted columns We drop 'NaN' values
df.drop(['start_station_latitude','start_station_longitude',␣
↪'end_station_latitude', 'end_station_longitude'], axis=1, inplace=True)
```

1.5.3 Test

[12]:
```
# Verify if columns are dropped
for i,v in enumerate(df.columns):
    print(i,v)
```

```
0 duration_sec
1 start_time
2 end_time
3 bike_id
4 user_type
5 member_birth_year
6 member_gender
7 bike_share_for_all_trip
```

1.5.4 Define:

Correct erroneous data types of (`start_time`, `end_time`) and change to datetime which is the proper datatype format

1.5.5 Code

[13]:
```
# Change datatype of start_time, end_time` to datetime.
df.start_time = pd.to_datetime(df.start_time)
df.end_time = pd.to_datetime(df.end_time)
```

1.5.6 Test

[14]:
```
# Verify if columns are dropped
print(df.start_time.dtype)
print(df.end_time.dtype)
```

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

[15]:
```
# create a copy of the cleaned data
df_cleaned = df.copy()
df_cleaned.head()
```

[15]:
```
   duration_sec              start_time                end_time  bike_id  \
0         52185 2019-02-28 17:32:10.145 2019-03-01 08:01:55.975     4902
1         42521 2019-02-28 18:53:21.789 2019-03-01 06:42:03.056     2535
2         61854 2019-02-28 12:13:13.218 2019-03-01 05:24:08.146     5905
3         36490 2019-02-28 17:54:26.010 2019-03-01 04:02:36.842     6638
4          1585 2019-02-28 23:54:18.549 2019-03-01 00:20:44.074     4898

    user_type  member_birth_year member_gender bike_share_for_all_trip
0    Customer             1984.0          Male                      No
1    Customer                NaN           NaN                      No
2    Customer             1972.0          Male                      No
3  Subscriber             1989.0         Other                      No
4  Subscriber             1974.0          Male                     Yes
```

[16]:
```
df_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 8 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   duration_sec             183412 non-null  int64
 1   start_time               183412 non-null  datetime64[ns]
 2   end_time                 183412 non-null  datetime64[ns]
 3   bike_id                  183412 non-null  int64
 4   user_type                183412 non-null  object
 5   member_birth_year        175147 non-null  float64
 6   member_gender            175147 non-null  object
 7   bike_share_for_all_trip  183412 non-null  object
dtypes: datetime64[ns](2), float64(1), int64(2), object(3)
memory usage: 11.2+ MB
```

1.5.7 What is the structure of your dataset?

After cleaning the dataset we have 183412 rows/recordes and 8 columns including:

Duration_Sec: How long is the trip in seconds

Start and End time: when the trip has started and whenit end.

And user information i.e User type, birth year and gender etc

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

I am interested in looking at trip duration with respect to time and user type information. My objective in this investigation is to find out `when most trip occur or take place, what hours` of the day, days of the week or month of the year? How long does the avarage trip take? and which user types made on these trips. ### What features in the dataset do you think will help support your investigation into your feature(s) of interest?

Transforming the `start_time` and `Duration_sec` columns will be helpful in my investigation i.e trans forming the datetime column days, week, month etc, user_type, will help me find and determine the differences between subscribers and customers for the bike usage and their relationships.

1.6 Exploration

Let's transform our data and exteract new columns (start_day, end_day, start_hour, end_hour, start_month, and end month from the start_time and end_time in our origal dataset.

[17]:
```
### Extract day, month and year columns from the start_time and end datetime
df_cleaned['start_day'] = df_cleaned['start_time'].dt.day_name()
df_cleaned['end_day'] = df_cleaned['end_time'].dt.day_name()


df_cleaned['start_hour'] = df_cleaned['start_time'].dt.hour
df_cleaned['end_hour'] = df_cleaned['end_time'].dt.hour


df_cleaned['start_month'] = df_cleaned['start_time'].dt.month_name()
df_cleaned['end_month'] = df_cleaned['end_time'].dt.month_name()

df_cleaned['start_hour'] = df_cleaned.start_hour.astype(int)
df_cleaned['end_hour'] = df_cleaned.end_hour.astype(int)

df_cleaned['dur_per_minute'] =df_cleaned['duration_sec']/60
```

[18]:
```
df_cleaned.head(3)
```

[18]:
```
   duration_sec              start_time                end_time  bike_id  \
0         52185 2019-02-28 17:32:10.145 2019-03-01 08:01:55.975     4902
1         42521 2019-02-28 18:53:21.789 2019-03-01 06:42:03.056     2535
2         61854 2019-02-28 12:13:13.218 2019-03-01 05:24:08.146     5905

  user_type  member_birth_year member_gender bike_share_for_all_trip  \
0  Customer             1984.0          Male                      No
1  Customer                NaN           NaN                      No
2  Customer             1972.0          Male                      No

  start_day end_day  start_hour  end_hour start_month end_month  \
0  Thursday  Friday          17         8    February     March
1  Thursday  Friday          18         6    February     March
2  Thursday  Friday          12         5    February     March

   dur_per_minute
```

```
0      869.750000
1      708.683333
2     1030.900000
```

[19]:
```python
# Let us create ane column from member birth year to determine our users age
df_cleaned["age"] = datetime.now().year - df_cleaned.member_birth_year
```

[20]:
```python
df_cleaned.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  datetime64[ns]
 2   end_time                 183412 non-null  datetime64[ns]
 3   bike_id                  183412 non-null  int64
 4   user_type                183412 non-null  object
 5   member_birth_year        175147 non-null  float64
 6   member_gender            175147 non-null  object
 7   bike_share_for_all_trip  183412 non-null  object
 8   start_day                183412 non-null  object
 9   end_day                  183412 non-null  object
 10  start_hour               183412 non-null  int64
 11  end_hour                 183412 non-null  int64
 12  start_month              183412 non-null  object
 13  end_month                183412 non-null  object
 14  dur_per_minute           183412 non-null  float64
 15  age                      175147 non-null  float64
dtypes: datetime64[ns](2), float64(3), int64(4), object(7)
memory usage: 22.4+ MB
```

1.6.1 Let's define age category and create a new column with our age category.

Let make ages between:

15-25 = Youth

24-64 = Adult and

65+ = Senior

[21]:
```python
# Add a new column catagoray next age group and call it "age_group"
category = pd.cut(df_cleaned.age, bins=[15, 25, 65, 140], labels=["Youth",␣
 ↪"Adult", "Senior"])
df_cleaned.insert(15, "age_group", category)
```

1.7 Univariate Exploration

[22]:
```
# Lets check our column names
for i,v in enumerate(df_cleaned.columns):
    print(i,v)
```

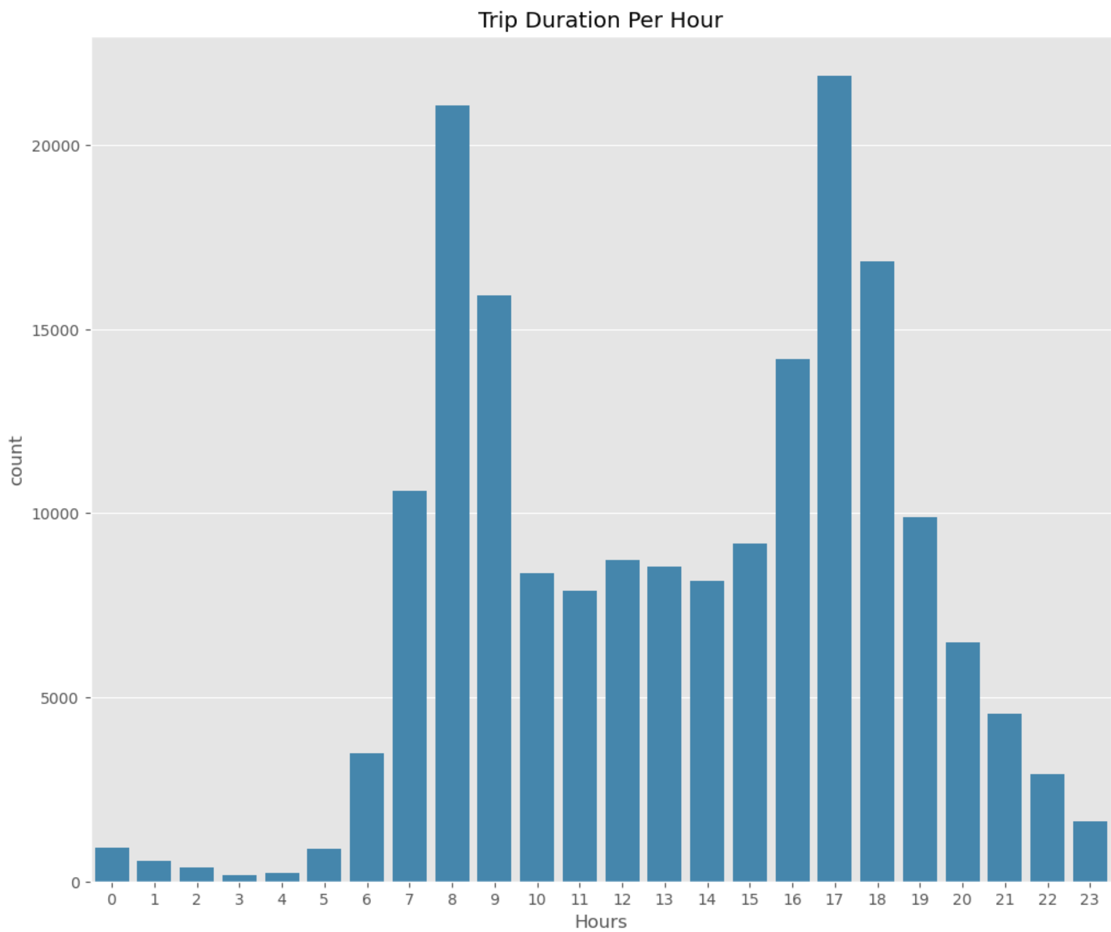
```
0 duration_sec
1 start_time
2 end_time
3 bike_id
4 user_type
5 member_birth_year
6 member_gender
7 bike_share_for_all_trip
8 start_day
9 end_day
10 start_hour
11 end_hour
12 start_month
13 end_month
14 dur_per_minute
15 age_group
16 age
```

Let us start with the usage of the bikes and find out when the most trips are taken with repect time_start i.e hour weekday, and month.

[23]:
```
base_color = sb.color_palette()[1]
```

Ride Frequency by hours

[24]:
```
# univirate analysis
# let take a look at the trip duration per hour frequence
plt.figure(figsize=(12, 10))
sb.countplot(data= df_cleaned,
            x=df_cleaned['start_hour'].sort_values(ascending=True),
            color=base_color)
plt.title("Trip Duration Per Hour")
plt.xlabel('Hours');
```

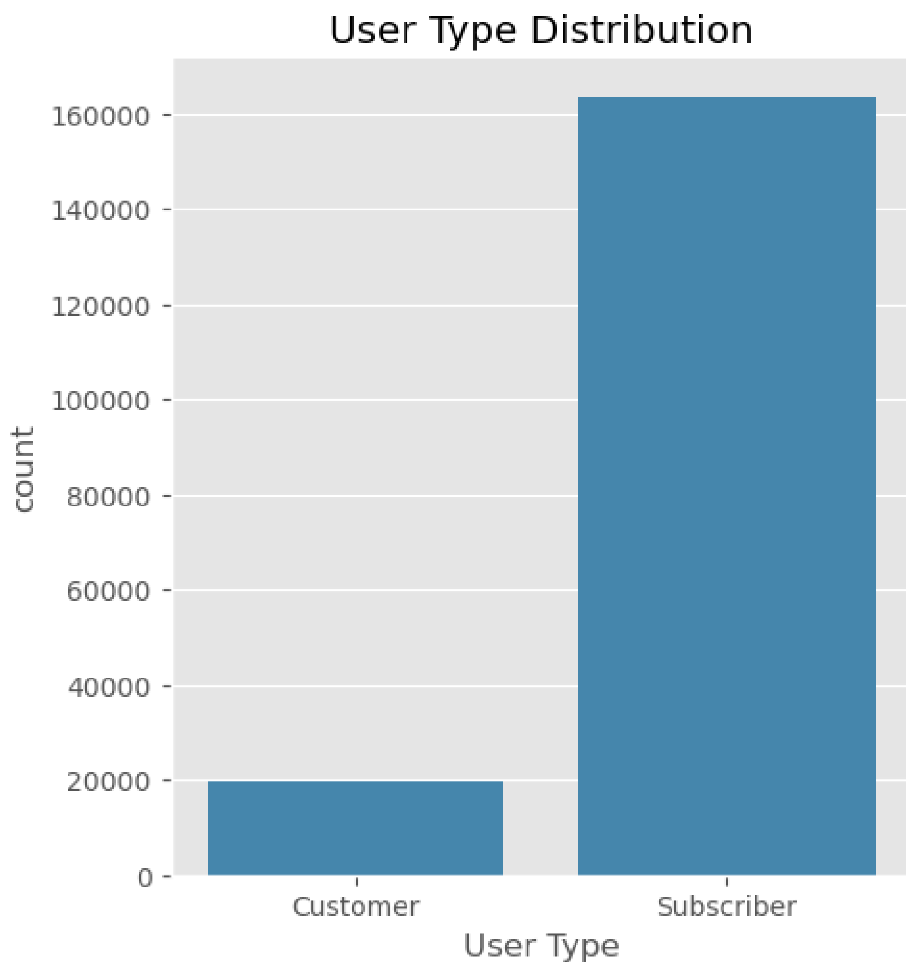


The 8th, 9th, 17th and 18th hours have the highest trip records. This is expected as it can be linked to morning rush and closing hour from work.

Which user types made the most trips?

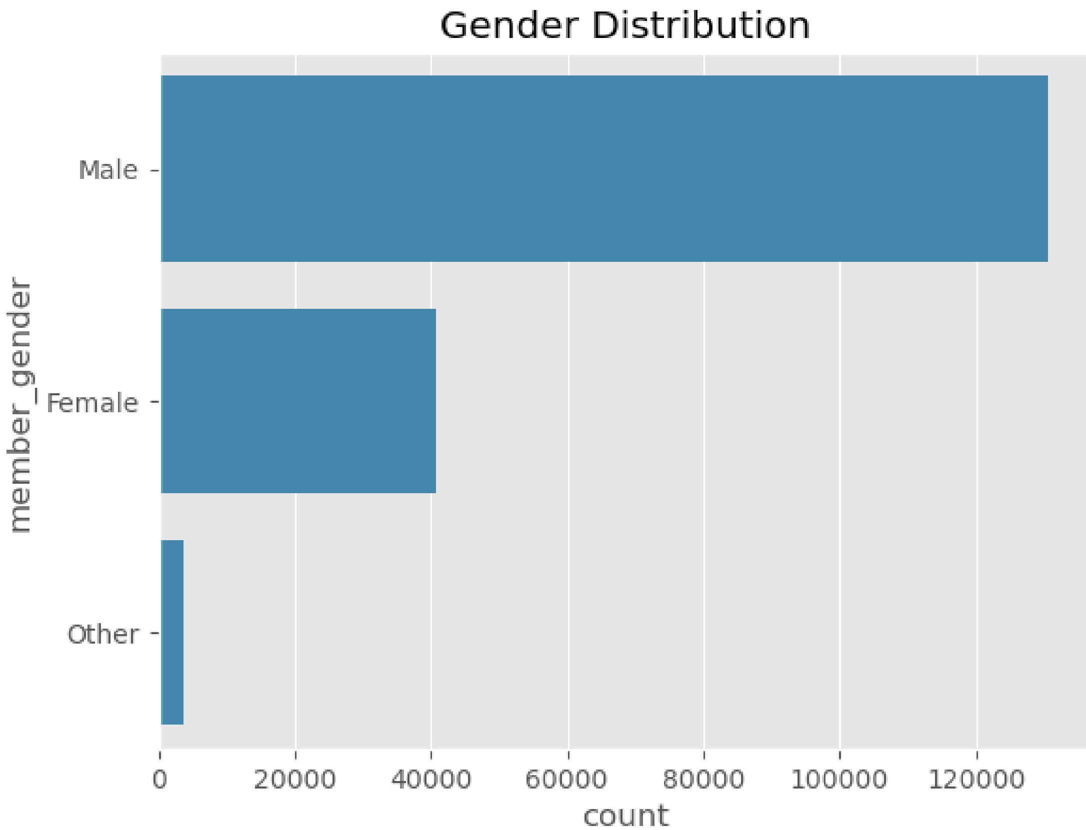
[25]:
```
# let's see Which user types made the most trip?
plt.figure(figsize=(12, 12))
sb.catplot(data = df_cleaned, x = 'user_type', kind="count", color = base_color)
plt.title('User Type Distribution')
plt.xlabel("User Type");
```

```
<Figure size 1200x1200 with 0 Axes>
```



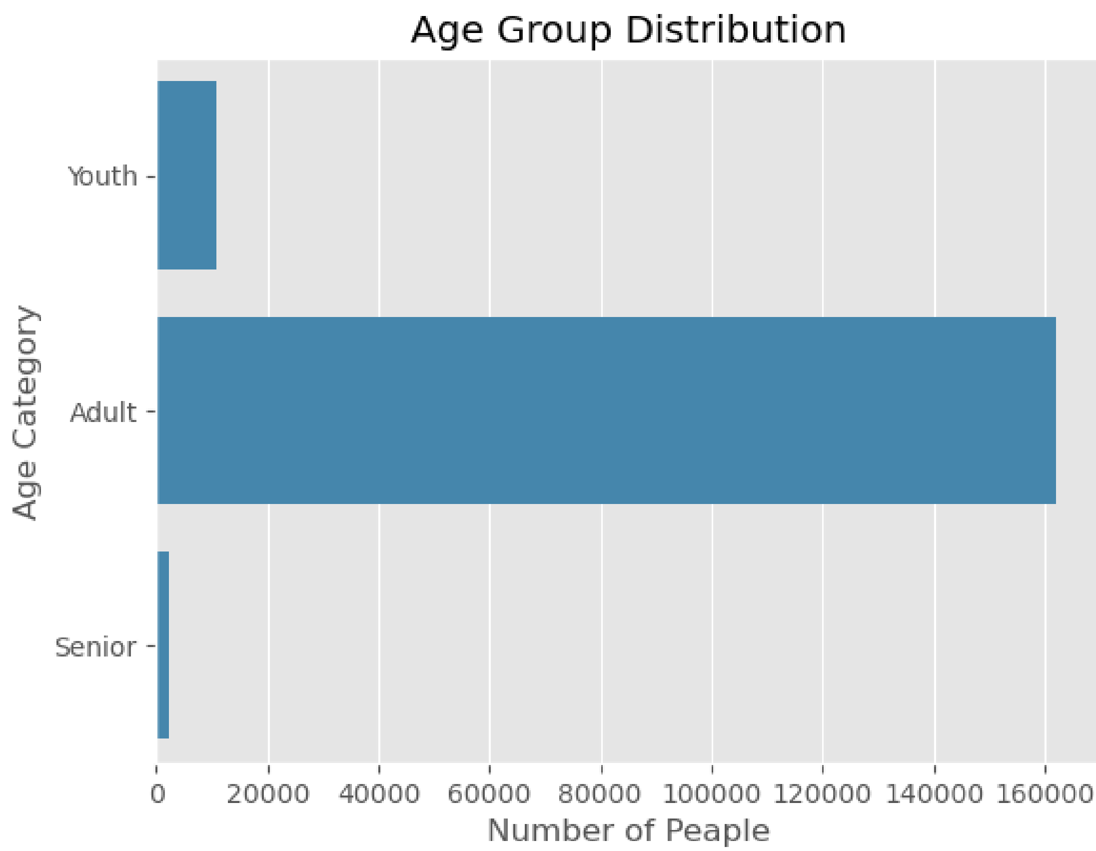
Observation From this visual graph we see that Subscribers have made the most trips in data.

[26]:
```
#which gender is the most predominat in our data
sex_order =df.member_gender.value_counts().index
sb.countplot(y='member_gender', data= df_cleaned, color=base_color,␣
 ↪order=sex_order)
plt.title('Gender Distribution');
```



Observations In our data we have more male than anyother Gender.

[27]:
```
# Let's plot the histogram of members age to see what the distribution of age␣
↪looks like
ax = sb.countplot(data = df_cleaned, y = 'age_group', color = base_color)
# lets Show the total count
plt.title("Age Group Distribution")
plt.xlabel("Number of Peaple")
plt.ylabel("Age Category");
```



 # In which day of the week are most bike rides occured with respect to duration in seconds

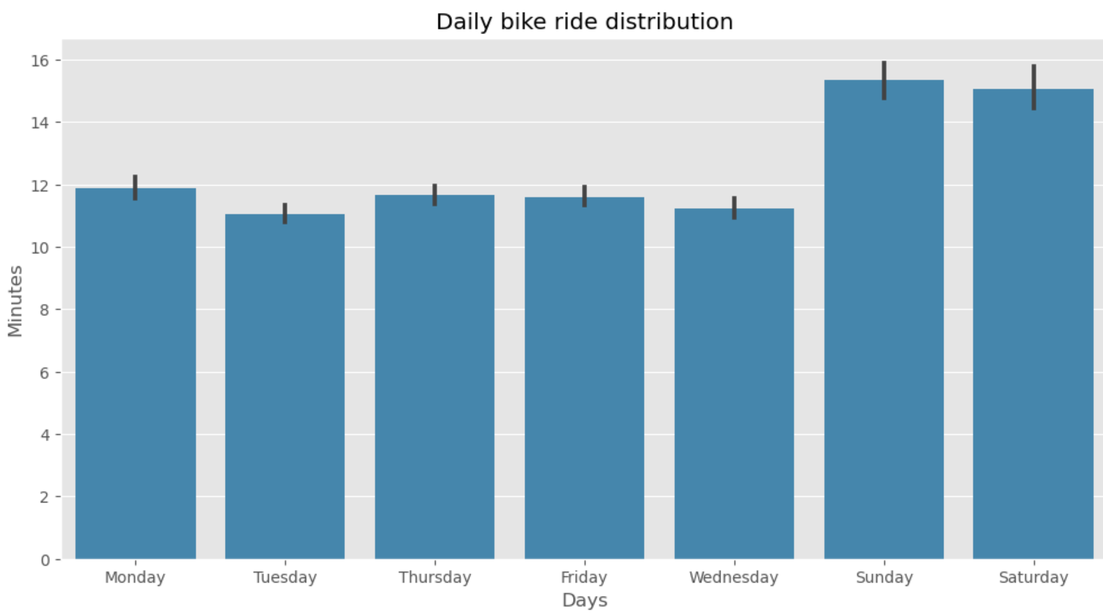
[28]:
```
# in which day of the week are most bike rides occured with respect to duration
↪in seconds
count=df_cleaned.groupby('start_day')["duration_sec"].mean().reset_index()
count.sort_values(by=['duration_sec'],ascending=True )
```

[28]:

	start_day	duration_sec
5	Tuesday	663.305567
6	Wednesday	673.671165
0	Friday	695.795073
4	Thursday	699.040998
1	Monday	713.159616
2	Saturday	902.661993
3	Sunday	919.746054

Which days of the week most trip were taken?

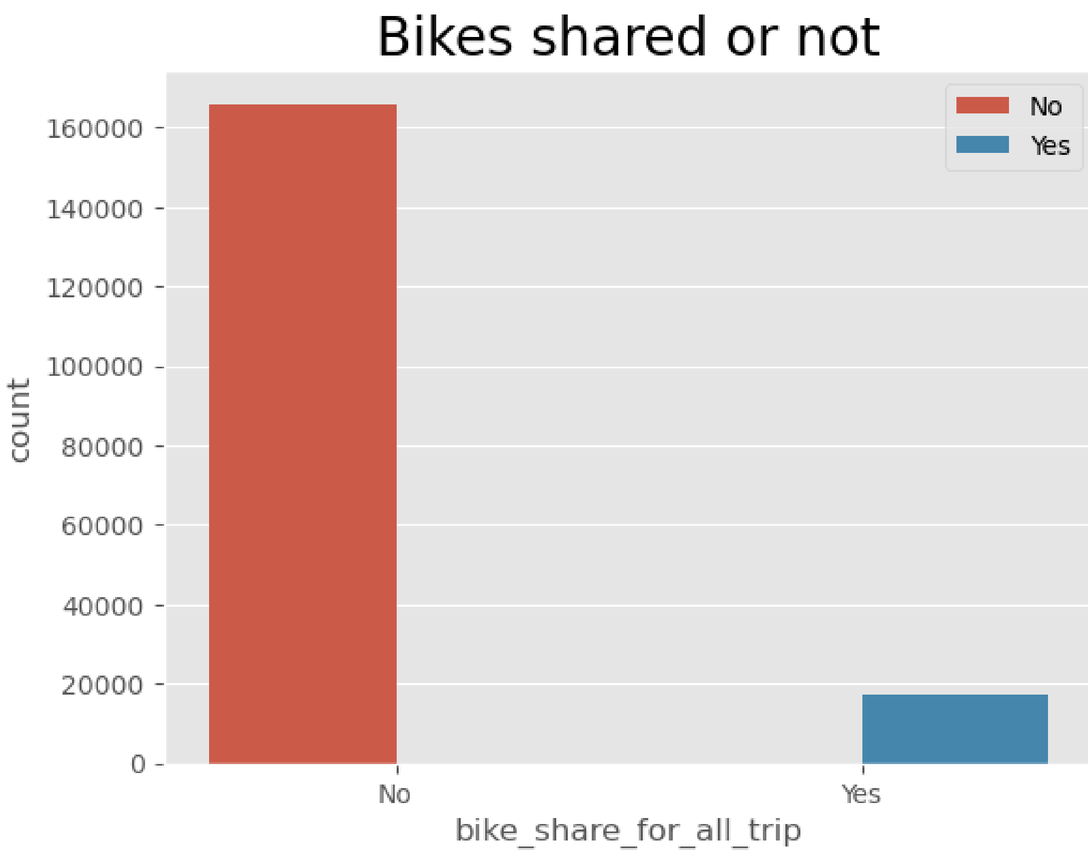
[29]:
```
# let take a look at the trip duration per day frequence
def horizontal_bar():
    plt.figure(figsize=(12, 6))
    myplot=sb.barplot(x='start_day',y= 'dur_per_minute',
                     data=df_cleaned.
 ↪sort_values(by=['dur_per_minute'],ascending=True ),
                     color=base_color)
    plt.title("Daily bike ride distribution")
    plt.xlabel("Days")
    plt.ylabel("Minutes")
horizontal_bar()
```



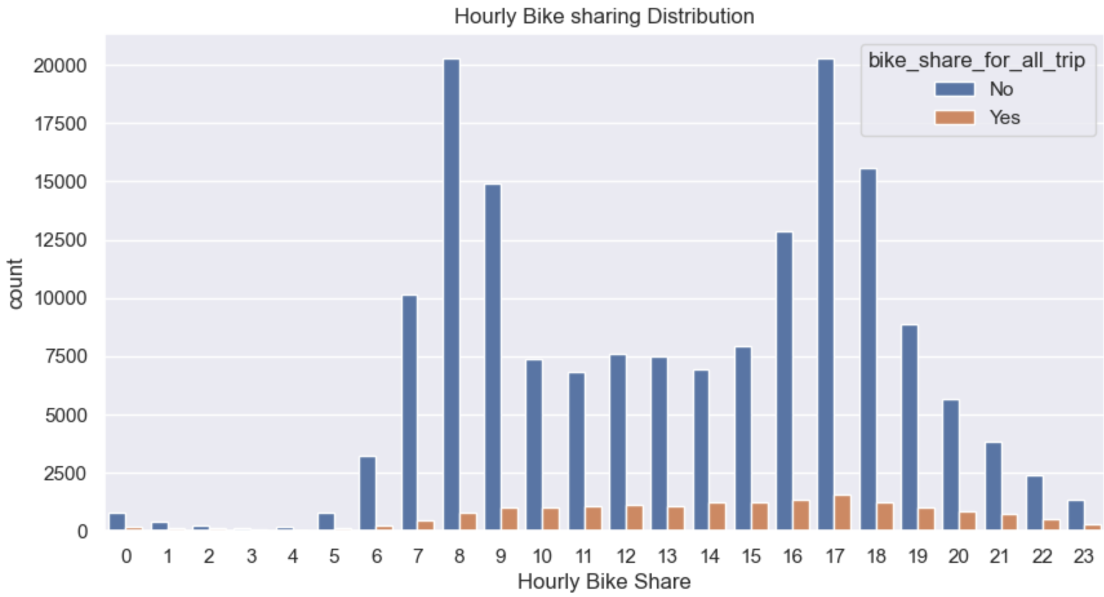
Observations Most of the trips were taken (start and end days) on weekends (sat, Sun), It looks like it is pretty consistance during the weekdays

Count the number of bikes shared for all trips vs Not shared?

[30]:
```
## Let's Check the count the number of bike shared for all trips vs Not shared.
bake_share = sb.countplot(x = df_cleaned.bike_share_for_all_trip, hue =␣
 ↪'bike_share_for_all_trip', data = df_cleaned)
plt.legend()
plt.title("Bikes shared or not", fontdict = {'fontsize': 20});
```



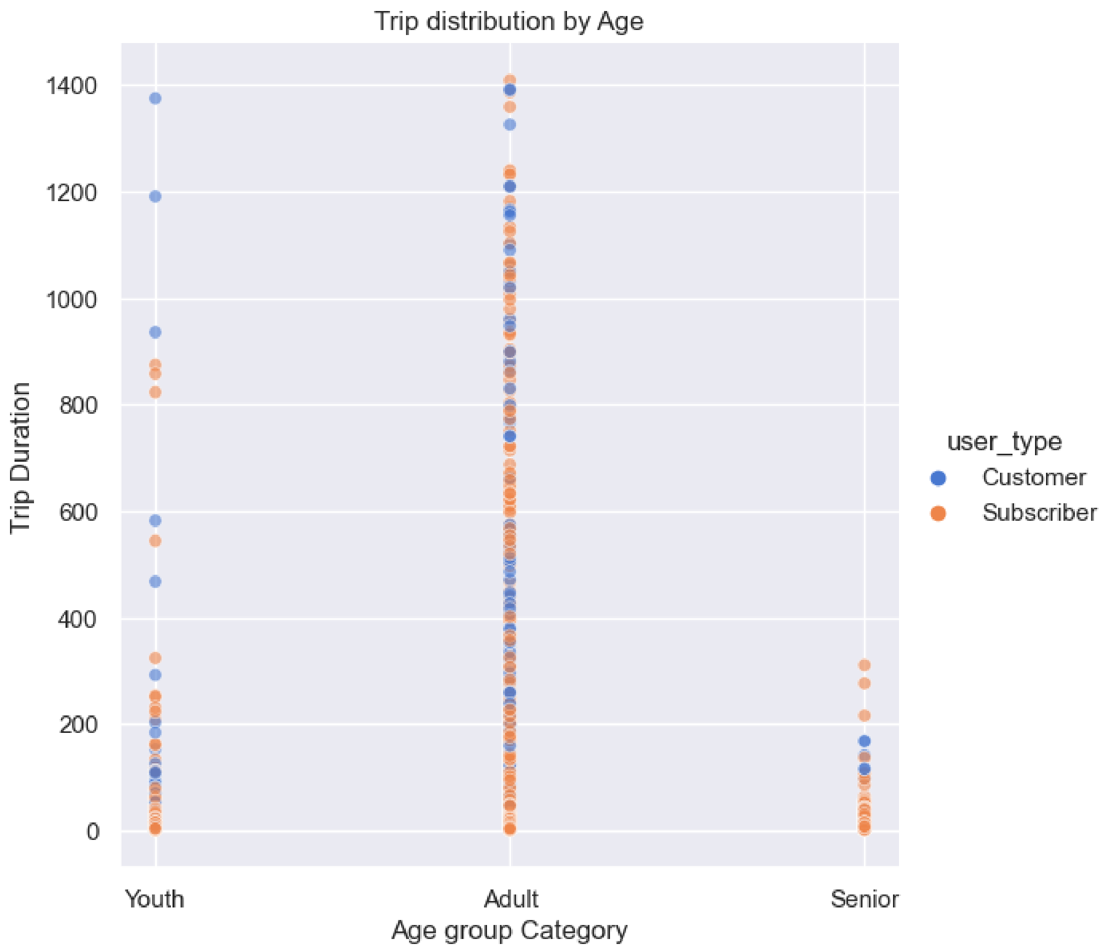
[31]:
```
# Let's check the countplot distribution of all shared rides by hourly start␣
↪and hourly end and compare the regular rides?.
plt.figure(figsize = (10,5))
sb.set(style = "darkgrid")
sb.countplot(x =df_cleaned["start_hour"].sort_values(ascending=True), hue =␣
↪'bike_share_for_all_trip', data = df_cleaned)
plt.title('Hourly Bike sharing Distribution')
plt.xlabel('Hourly Bike Share');
```



Observation as expected the vitual graph shows that the shared bikes trips ("green) are less thoughtout the hours while regular ride bikes (blue) are more when comparing to the shared bikes.

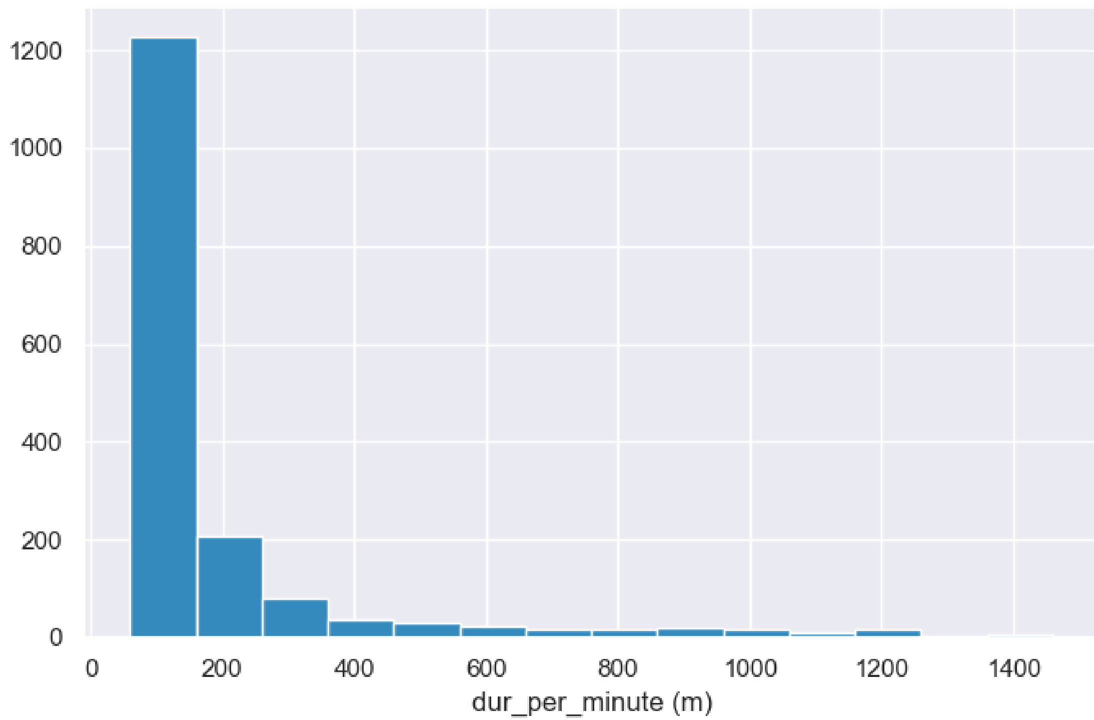
[32]:
```
# let'a plot reple plot showing distribution between age and trip duration
plt.figure(figsize=[12,8])
sb.relplot(x="age_group", y="dur_per_minute", hue="user_type",
            sizes=(40, 400), alpha=.6, palette="muted",
            height=6, data=df_cleaned)
plt.title('Trip distribution by Age')
plt.xlabel('Age group Category')
plt.ylabel('Trip Duration');
```

```
<Figure size 1200x800 with 0 Axes>
```



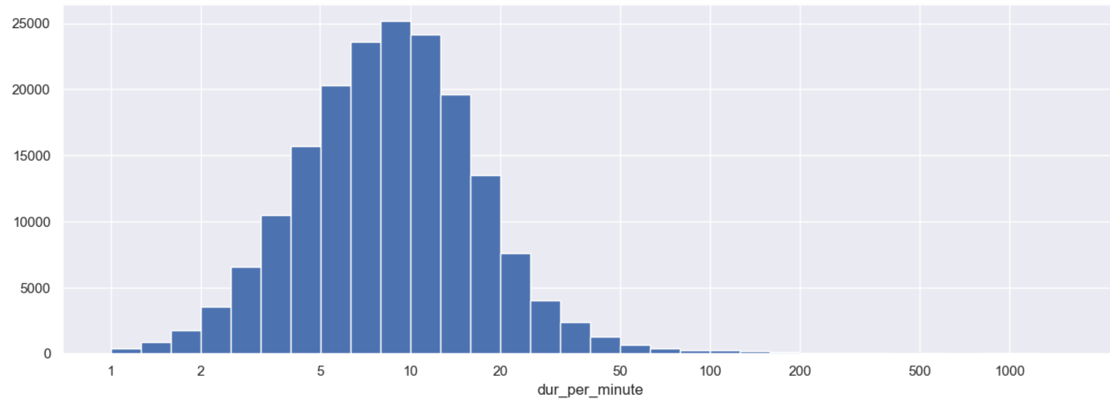
[33]:

```
# Find trip duration distribution
def histogram():
    bins = np.arange(60, df_cleaned['dur_per_minute'].max()+100, 100)
    plt.figure(figsize=[8, 5])
    plt.hist(df_cleaned['dur_per_minute'], bins = bins, color= base_color);
    plt.xlabel('dur_per_minute (m)');

histogram()
```



There's a long tail in the distribution, so let's put it on a log scale instead

[34]:
```
def histogram():
    plt.figure(figsize=[15,5])
    bin_edges = 10 ** np.arange(0.0, np.log10(df_cleaned.dur_per_minute.
 ↪max())+0.1, 0.1)
    plt.hist(data = df_cleaned, x = 'dur_per_minute', bins = bin_edges)
    plt.xscale('log')
    tick_locs = [1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]
    plt.xticks(tick_locs, tick_locs)
    plt.xlabel('dur_per_minute');

histogram()
```




```
[35]: # What is average trip duratons
df_cleaned['dur_per_minute'].mean()
```

```
[35]: 12.10130725724969
```

We can see from the histogram that most rides took about (8-12) minutes. And very few rides lasted more than one hour (60 minutes). We also find trip duration avearge is 12 minutes.

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

'Time' The Avarage trip duration in the dataset is 12 minutes. most trips were made by adults age between 25 to 64.

Based on hours:The 8th, 9th, 17th and 18th hours have the highest trip records. This is expected as it can be linked to morning rush and closing hour from work. Weekdays: Most of the trips were taken (start and end days) on weekends, It looks like it pretty consistance during the weekdays.

`user types` Subscribers have made the most trips in data

1.7.2 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?

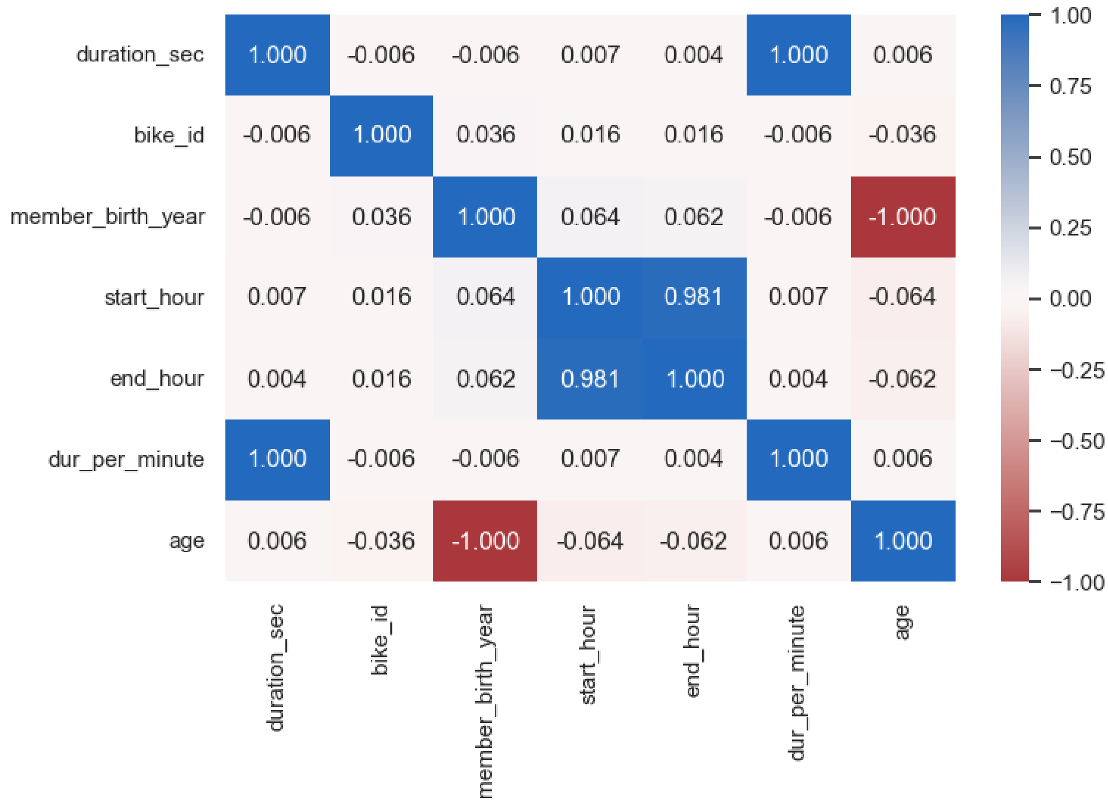
A logarithmic scale transformation was applied on duration_minutes hist plot, because there's a long tail in the distribution.

1.8 Bivariate Exploration

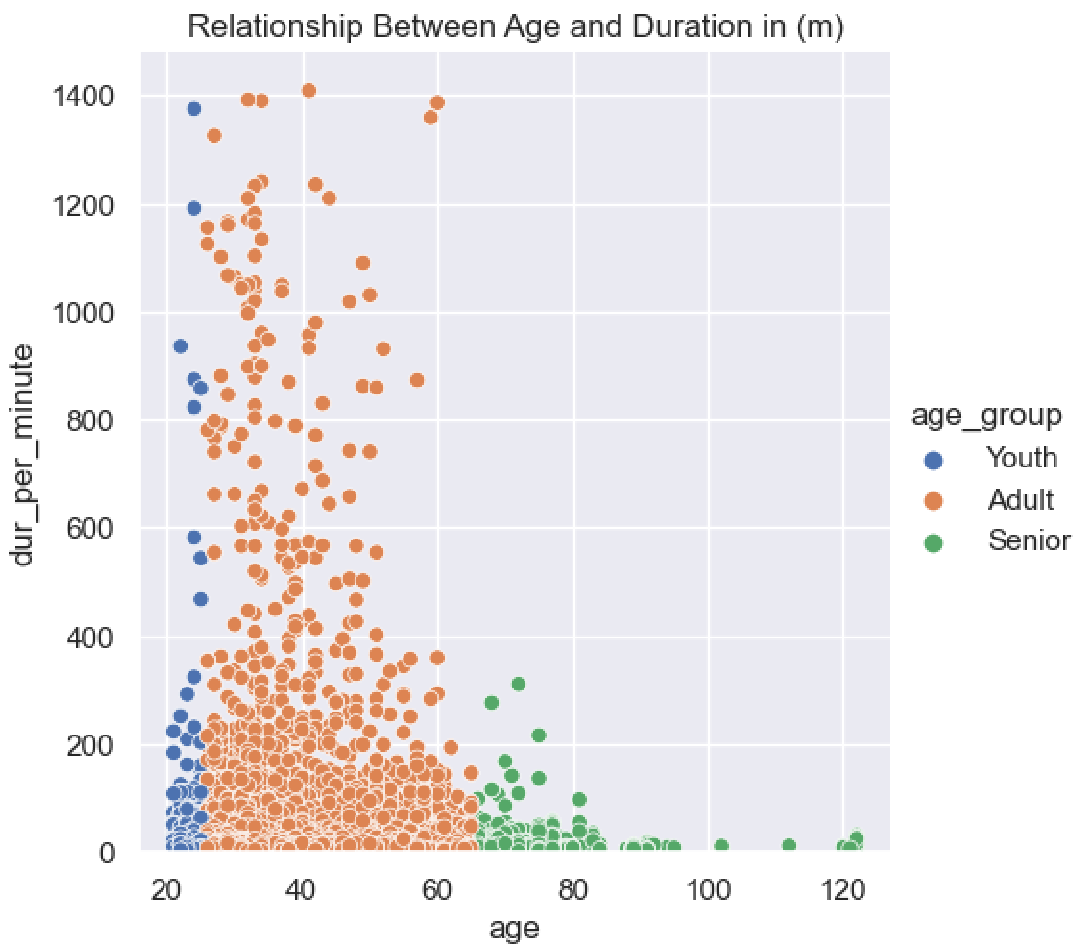
In this section, investigate relationships between pairs of variables in your data. Make sure the variables that you cover here have been introduced in some fashion in the previous section (univariate exploration).

```
[36]: # let's find the correlations between our numerial variables
corr = df_cleaned.corr()
```

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



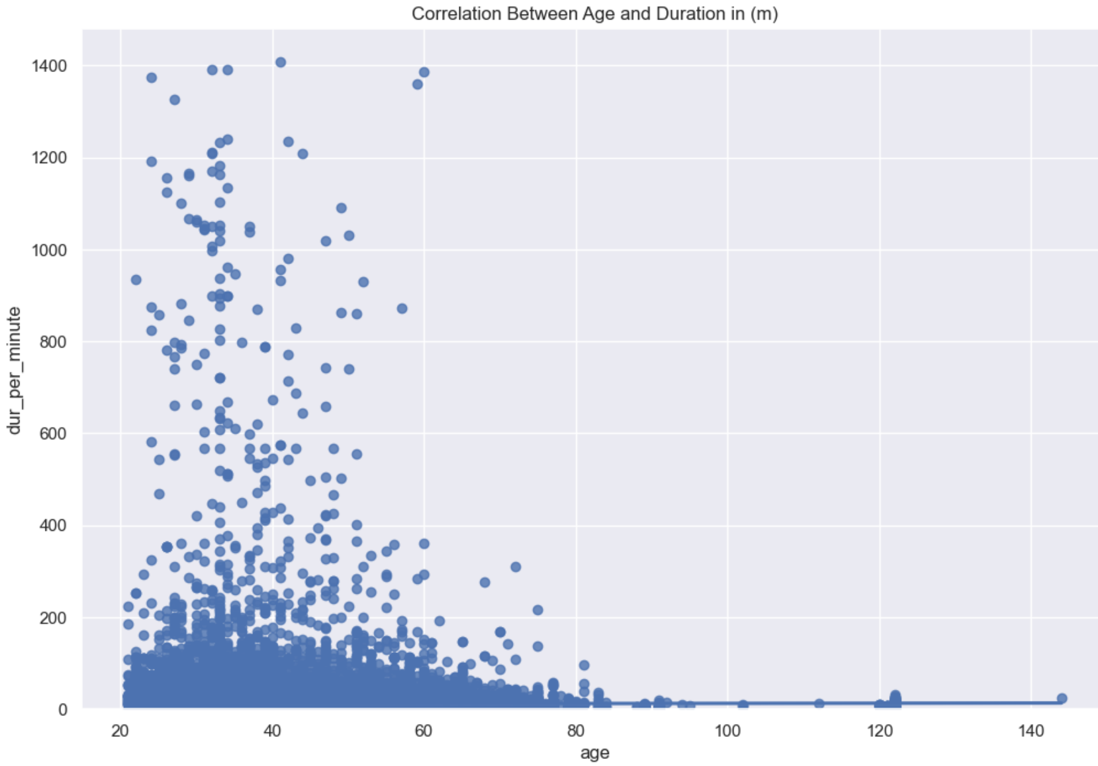
[37]:
```
# lets find the realation between age and duration_sec alone
sb.relplot(x="age", y="dur_per_minute", hue="age_group", data=df_cleaned)
plt.ylim(0)
plt.title("Relationship Between Age and Duration in (m)");
```



1.8.1 Observation

As the Age increases the trip duration decreases

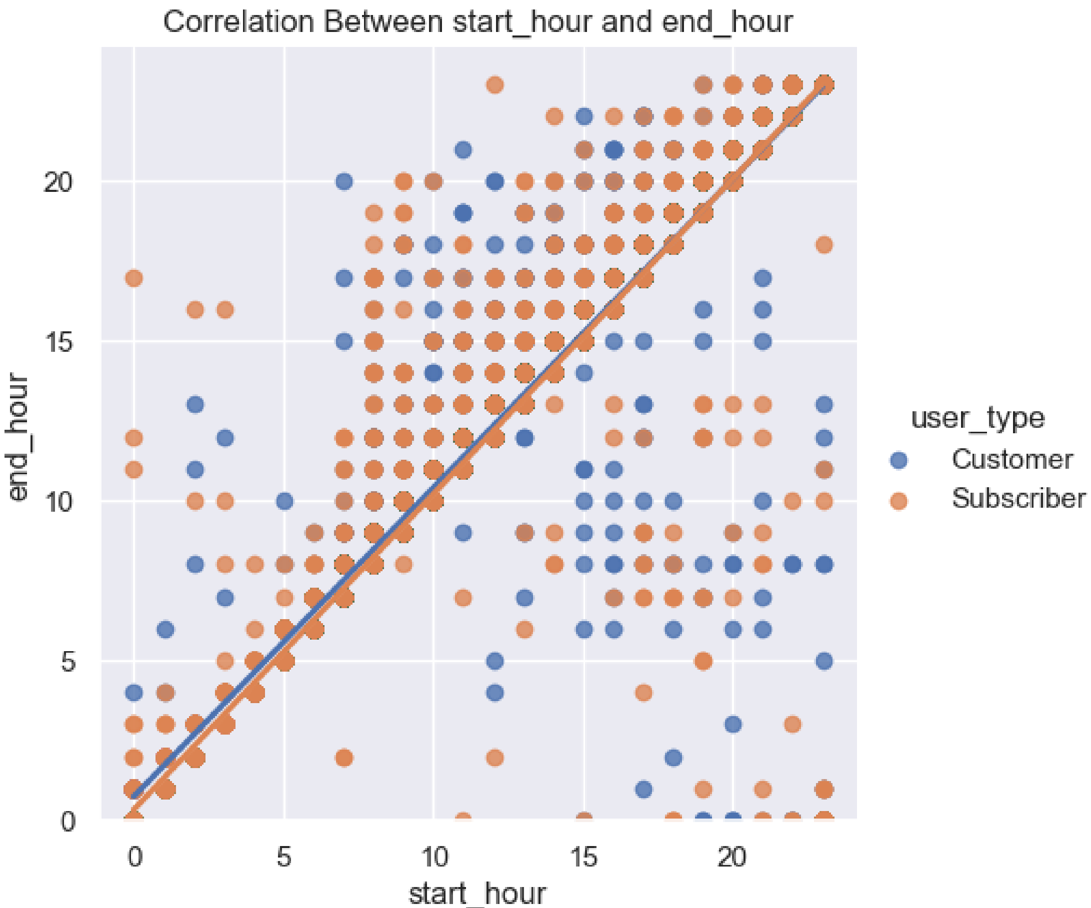
[38]:
```
# lets look the relationship more closely by coloring age group category to the␣
 ↪relationship
plt.figure(figsize=(12, 8))
sb.regplot(x="age", y="dur_per_minute", data=df_cleaned)
plt.ylim(0)
plt.title("Correlation Between Age and Duration in (m)");
```



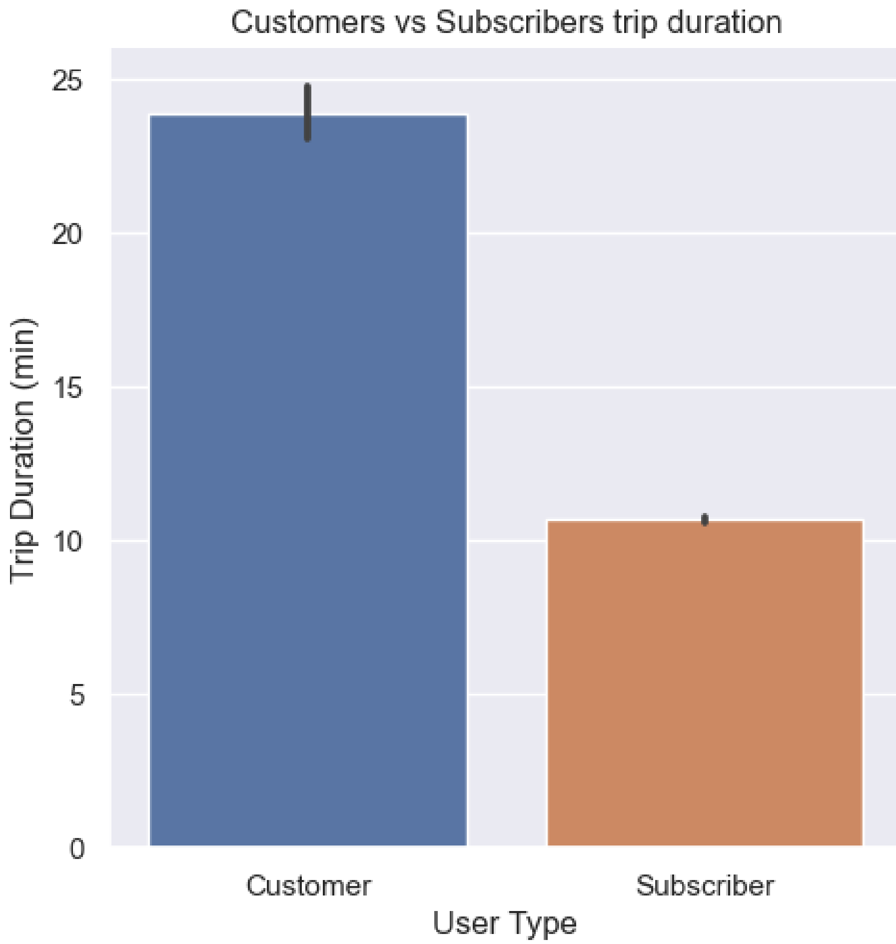
Observation Age doesn't seem to have a good relatioship with duration since the regression is so close to the horizantal. from this graph we see that as the age increases the duration decreases.

[39]:
```
plt.figure(figsize=(8, 5))
sb.lmplot(x="start_hour", y="end_hour", data=df_cleaned, hue="user_type")
plt.ylim(0)
plt.title("Correlation Between start_hour and end_hour");
```

```
<Figure size 800x500 with 0 Axes>
```



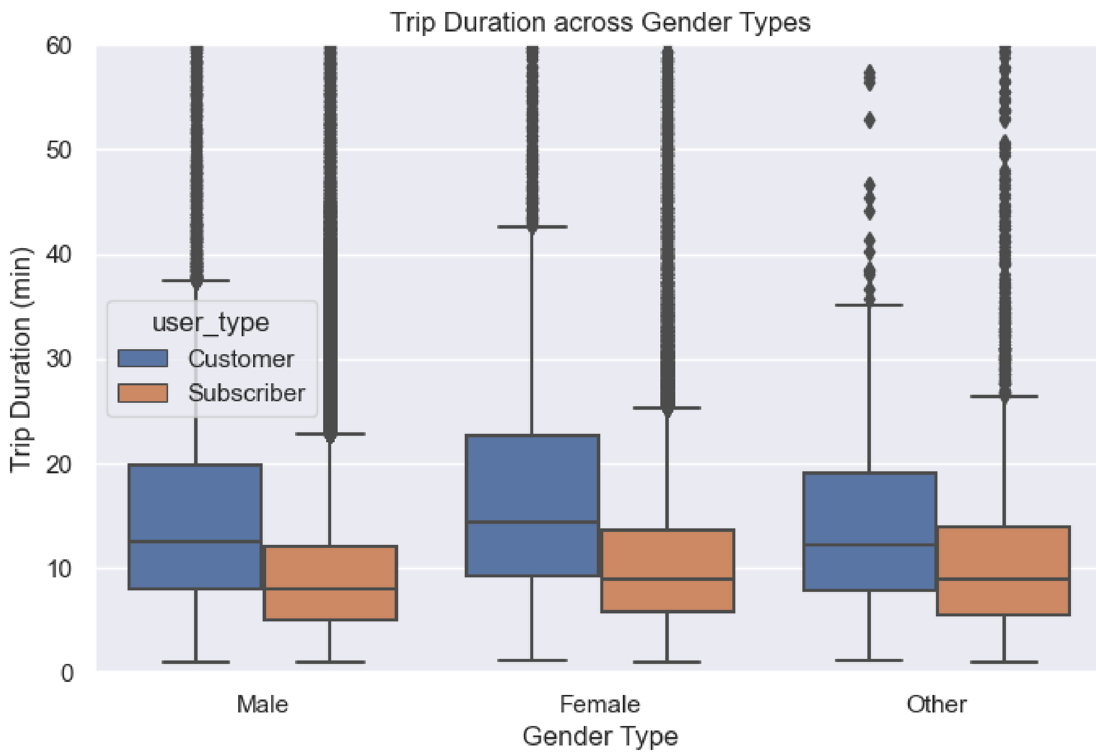
[40]:
```
# Chech trip duration  between Customers and Subscribers
sb.catplot(data=df_cleaned, kind="bar", x="user_type", y="dur_per_minute")
plt.title('Customers vs Subscribers trip duration')
plt.xlabel('User Type')
plt.ylabel('Trip Duration (min)');
```



Observation The distribution shows us that customers take a longer trip than subscribers.

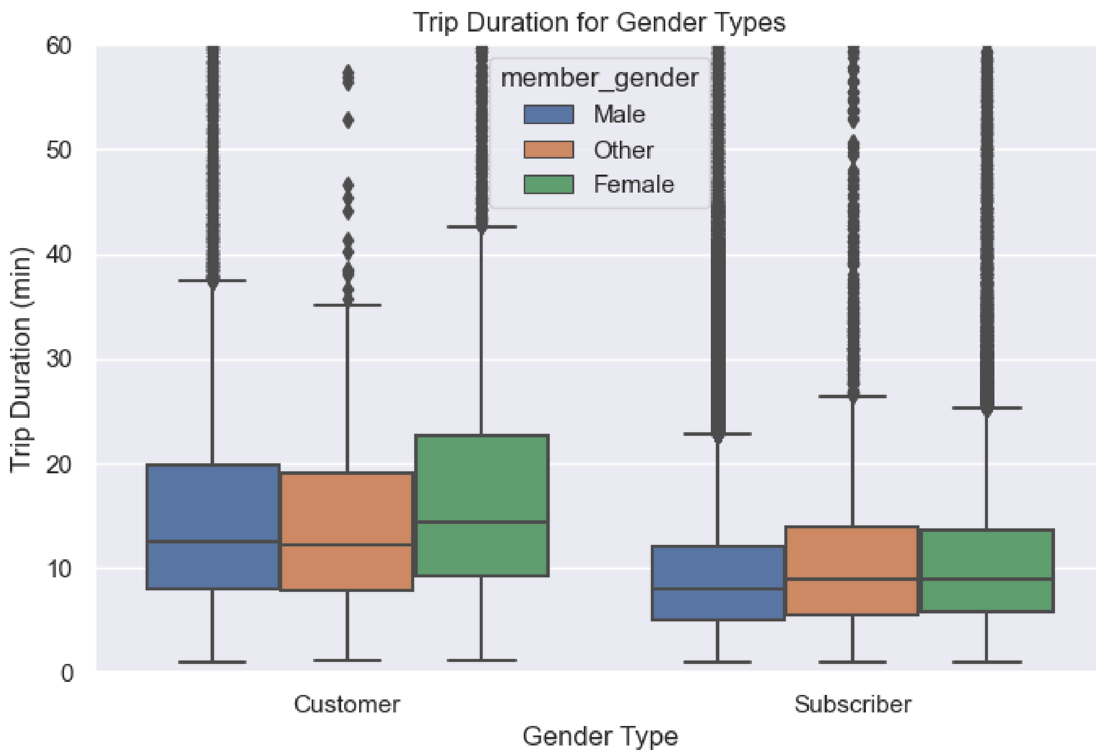
Let's look at the Categorical variable relationships in terms of user-type.

[41]:
```
# let us look at gender type
plt.figure(figsize = [8, 5])
sb.boxplot(x='member_gender', y='dur_per_minute', data = df_cleaned,␣
↪hue="user_type",  order=['Male', 'Female', 'Other'])
plt.ylim(0, 60)
plt.title('Trip Duration across Gender Types')
plt.xlabel('Gender Type')
plt.ylabel('Trip Duration (min)');
```



Observation Customer type users take longer trips through all the gender groups.

[42]:
```
# Let look at Duration across all the genders
plt.figure(figsize = [8, 5])
sb.boxplot(x='user_type', y='dur_per_minute', data = df_cleaned,␣
  ↪hue="member_gender")
plt.ylim(0, 60)
plt.title('Trip Duration for Gender Types')
plt.xlabel('Gender Type')
plt.ylabel('Trip Duration (min)');
```



Observations looking at customer boxlot, females long trips followeb by male, on other hand female and other gender are very close with the male duration is less compared female and other gender. from this visual we can deduce females take longer trips than any other gender.

1.8.2 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 this part of project we looked and examined the realtionships between selected numerical and categorical variables of interest.

first we have examined the relation between "age" and "duration per minute" and we observed that as the user age increases the trip duration decreases.

We also looked at the correlation between usertype and duration and found the customer user types take more trips than subscribe type users.

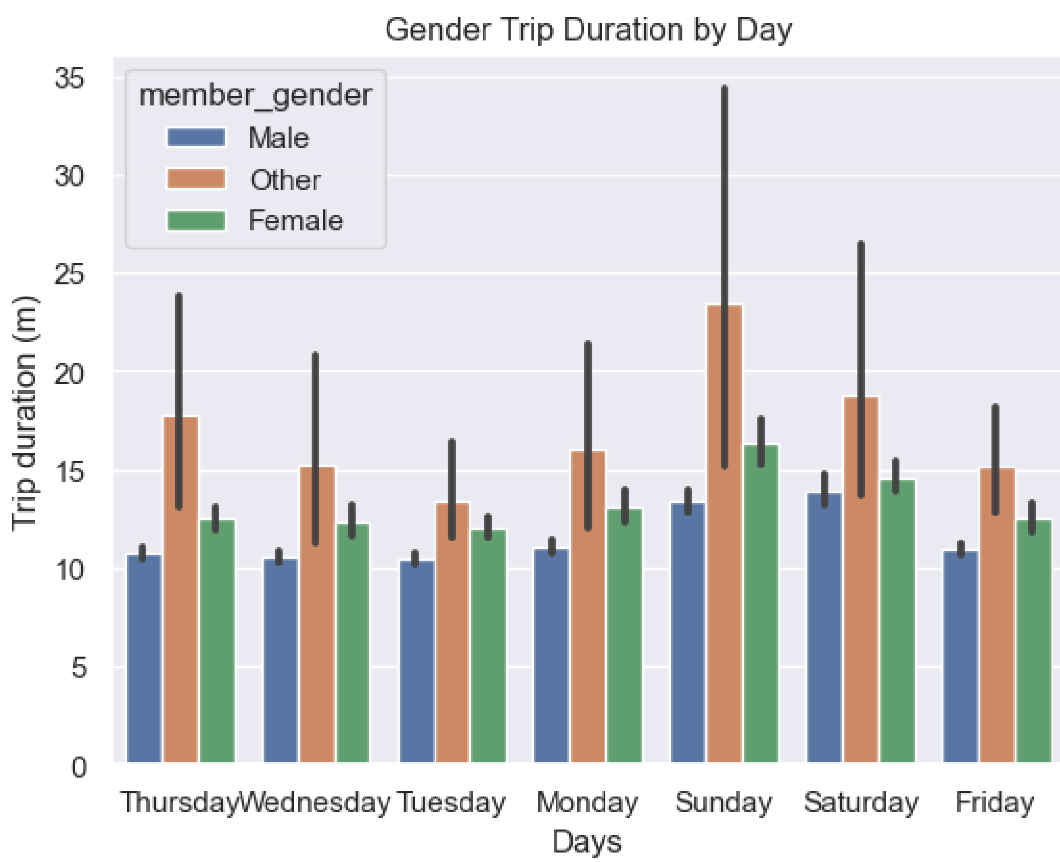
1.8.3 Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

Looking at the relationship between gender type and duration per minute. The garph showed that females and Other genders take longer trip durations then Male which I was surprised.

1.9 Multivariate Exploration

[43]:
```
# Compare daily trip duration by gender
sb.barplot(x="start_day", y="dur_per_minute",
              hue="member_gender",
              data=df_cleaned)
plt.title('Gender Trip Duration by Day')
plt.ylabel('Trip duration (m)')
plt.xlabel('Days')
```

[43]: Text(0.5, 0, 'Days')

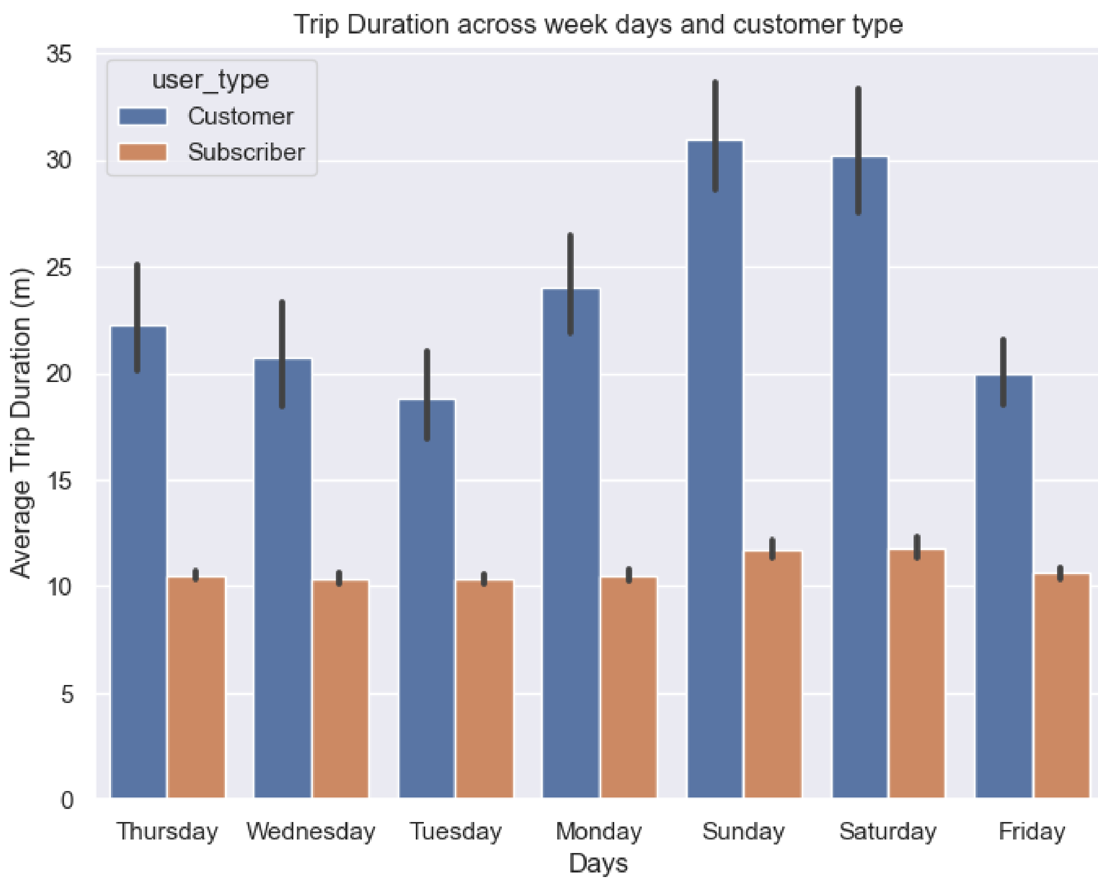


Observation Unsureprisinly as we have seen our previeous analaysis, males still have the shortest bike trip duration per day compared to female and other genders.

[44]:
```
# Compare Week day trip duration with respect user_type
fig = plt.figure(figsize = [8,6])
ax = sb.barplot(data = df_cleaned, x ='start_day', y = 'dur_per_minute', hue =␣
 ↪'user_type')
plt.title('Trip Duration across week days and customer type')
```

```
plt.xlabel('Days')
plt.ylabel('Average Trip Duration (m)')
plt.xlabel('Days')
```

[44]: Text(0.5, 0, 'Days')

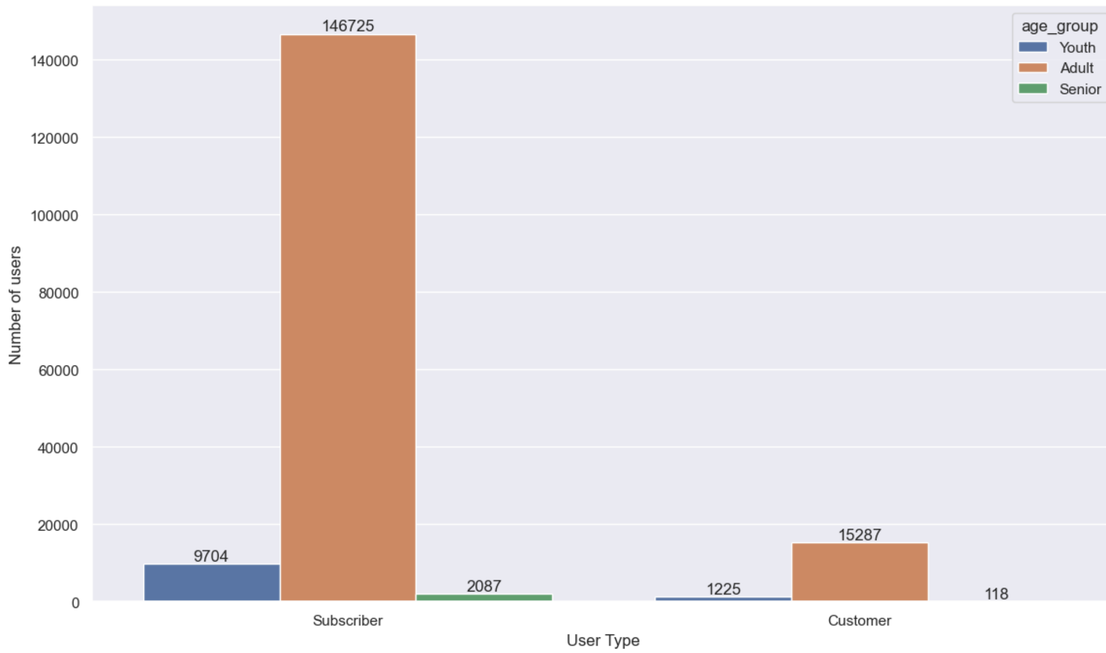


Observation The customer user types take more trips than subscribe type users during week day.

1.9.1 Display the number of users, type and and their age catagory.

[45]:
```
#display the numbers of users, their user_typ and and their age category

plt.figure(figsize = [30, 8])
plt.subplot(1, 2, 1)
ax = sb.countplot(data=df_cleaned, x="user_type", hue="age_group",
                            order=df_cleaned.user_type.value_counts().index)
for container in ax.containers:
    ax.bar_label(container)
```

```
plt.xlabel('User Type')
plt.ylabel('Number of users');
```



Observations In our data we, have 9704 youth ages between 15 through 24. 14625 Adults subscribers between age 24 and 64, and 2087 seniors age 65+

For Customer user types, we have 15287 Adults 1225 youth, and 118 seniors.

[46]:
```
df_cleaned.to_csv('fordgobiketrip_cleaned_data.csv', index=False)
new_df = pd.read_csv("fordgobiketrip_cleaned_data.csv")
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 17 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   bike_id                  183412 non-null  int64
 4   user_type                183412 non-null  object
 5   member_birth_year        175147 non-null  float64
 6   member_gender            175147 non-null  object
 7   bike_share_for_all_trip  183412 non-null  object
 8   start_day                183412 non-null  object
```

```
 9   end_day                    183412 non-null  object
 10  start_hour                 183412 non-null  int64
 11  end_hour                   183412 non-null  int64
 12  start_month                183412 non-null  object
 13  end_month                  183412 non-null  object
 14  dur_per_minute             183412 non-null  float64
 15  age_group                  175146 non-null  object
 16  age                        175147 non-null  float64
dtypes: float64(3), int64(4), object(10)
memory usage: 23.8+ MB
```

Find the number of rides in each hour of the day for each user type and age group?

1.9.2 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?

Looking at the same variables, I again examined the relationship between Weekdays, gender and the trip durations and as we have seen in our previous analysis from Bivariate section Males have the shortest bike trip on weekdays.

Analazing the group age and user type distribution I found adults which is defined a ages between 25-64 in our dataset made the most trips

1.9.3 Were there any interesting or surprising interactions between features?

There wasn’st any interactions that got my attention.

1.10 Conclusions

1.10.1 Key Insights from my posted questions:

My goal in this project was to answer the following simple questions:

####

Q1: Which hours of the day most trip were taken?

Answer: ‘8th, 9th, 17th, and 18th is when most trips happen during the day

####

Q2: Which user types made the most trips?

Answer: Subscribers have mode most trips in our dataset

####

Q3: which day of the week were most bike rides occured with respect to duration in seconds?

Answer: Most of the trips were taken on weekends (Sat and sun), Duration is pretty consistance during the weekdays (Mon - Friday)

####

Q4: Which user types take the longest trip with respect duration per minutes.

Answer:Customers on average take a longer trip than subscribers.

1.10.2 Sources

https://seaborn.pydata.org/generated/seaborn.regplot.html https://stackoverflow.com/questions/55104819/displa
count-on-top-of-seaborn-barplot https://deepnote.com/@dain-russell/bike-exploration-328b5ba1-25e4-4a35-aaad-e70146c9e182 https://seaborn.pydata.org/generated/seaborn.boxplot.html https://seaborn.pydata.org/generated/seaborn.countplot.html https://stackoverflow.com/questions/26597116/sea
plots-not-showing-up

```
[ ]:
```