Part_I_exploration

November 22, 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]:
```
# import all packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
import seaborn as sb
import datetime as dt
from datetime import datetime
plt.style.use('ggplot')
%matplotlib inline
```

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

[3]:
```
# 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]:
```
df.shape
```

[4]:
```
(183412, 16)
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 16 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   duration_sec             183412 non-null  int64
 1   start_time               183412 non-null  object
 2   end_time                 183412 non-null  object
```

```
 3   start_station_id         183215 non-null  float64
 4   start_station_name       183215 non-null  object
 5   start_station_latitude   183412 non-null  float64
 6   start_station_longitude  183412 non-null  float64
 7   end_station_id           183215 non-null  float64
 8   end_station_name         183215 non-null  object
 9   end_station_latitude     183412 non-null  float64
 10  end_station_longitude    183412 non-null  float64
 11  bike_id                  183412 non-null  int64
 12  user_type                183412 non-null  object
 13  member_birth_year        175147 non-null  float64
 14  member_gender            175147 non-null  object
 15  bike_share_for_all_trip  183412 non-null  object
dtypes: float64(7), int64(2), object(7)
memory usage: 22.4+ MB
```

[6]:
```
# Identify missing Values
missing_data = df.isnull().sum()
missing_data
```

[6]:
```
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
```

[7]:
```
# Check for duplicates
df.duplicated().sum()
```

[7]: 0

[8]:
```
# Check for stats
df.describe()
```

[8]:

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

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

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

[9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 16 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   duration_sec             183412 non-null  int64
 1   start_time               183412 non-null  object
 2   end_time                 183412 non-null  object
 3   start_station_id         183215 non-null  float64
 4   start_station_name       183215 non-null  object
 5   start_station_latitude   183412 non-null  float64
 6   start_station_longitude  183412 non-null  float64
 7   end_station_id           183215 non-null  float64
 8   end_station_name         183215 non-null  object
 9   end_station_latitude     183412 non-null  float64
```

```
 10  end_station_longitude    183412 non-null  float64
 11  bike_id                  183412 non-null  int64
 12  user_type                183412 non-null  object
 13  member_birth_year        175147 non-null  float64
 14  member_gender            175147 non-null  object
 15  bike_share_for_all_trip  183412 non-null  object
dtypes: float64(7), int64(2), object(7)
memory usage: 22.4+ MB
```

1.3.1 What is the structure of your dataset?

Dataset structure, we have 183412 rows/recordes and 17 columns including:

Information on trip duration (`Duration_Sec`), starting and ending time/location (`Start and End time`,`start_station_name` , and `user information` i.e User type, birth year and gender etc

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

I am interested in analizing the trip duration with respect to time and user type information.

My objective in this investigation is to find out `when and where most trip occur/take place, what hours of the day, days of the week? How long does the avarage trip take? which user types made the trips`and `how are the dataset variables related to each other?`.

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

I believe`start_time`, `Duration_sec`, `user_type`, `start_station_name` and `end_station_name` columns will be helpful in my investigation. I will be extracting hours, and days week, from the start column to analyze and visualize bike usage over time,

Station names will be help me find out the most and least used stations in terms popularity. and User_type column will help me find detertmine the differences between subscribers and customers for the bike usage.

1.3.4 Data Quality

We have some data quality issues that we need to clean: - Remove unwanted columns - convert the proper data types for (start_time, end_time, bike_id, and user_type)

2 Cleaning Data

2.0.1 Define:

Drop unwanted columns: `start_station_id,end_station_id,start_station_latitude,start_station_long` `end_station_latitude, end_station_longitude`

3 Code

[10]:
```
#  drop unwanted columns
df.
 ↪drop(['start_station_id','end_station_id','start_station_latitude','start_station_longitude
 ↪'end_station_latitude', 'end_station_longitude', ], axis=1, inplace=True)
```

4 Test

[11]:
```
# 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 start_station_name
4 end_station_name
5 bike_id
6 user_type
7 member_birth_year
8 member_gender
9 bike_share_for_all_trip
```

4.0.1 Define:

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

4.0.2 Code

[12]:
```
# 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)
```

4.0.3 Test

[13]:
```
# Verify if columns are dropped
print(df.start_time.dtype)
print(df.end_time.dtype)
```

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

4.1 Exploration

Let's transform our data and exteract new columns by performing the following actions:

- convert duration_sec into duration_min,
- extract hour, day, month from start_time
- extract age from member_birth_year,
- add age_group catagory based on users age i.e (teenage (13-19), Young_Adult(20-30), Adult (31-49), Senior(50+)

[14]:
```
# Extract hour and day of the week columns from the start_time and age
# from member birth year
def extr_new_columns():
    #extract hour, day, month from start_time
    df['day'] = df['start_time'].dt.day_name()
    df['hour'] = df['start_time'].dt.hour
    # convert duration_sec into duration_min,
    df['dur_per_minute'] = df['duration_sec']//60

    #extract age from member_birth_year and convert into,
    df["age"] = (datetime.now().year - df.member_birth_year)

extr_new_columns()
```

[15]:
```
df["start_time"].head()
```

[15]:
```
0   2019-02-28 17:32:10.145
1   2019-02-28 18:53:21.789
2   2019-02-28 12:13:13.218
3   2019-02-28 17:54:26.010
4   2019-02-28 23:54:18.549
Name: start_time, dtype: datetime64[ns]
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 14 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   start_station_name       183215 non-null  object
 4   end_station_name         183215 non-null  object
 5   bike_id                  183412 non-null  int64
 6   user_type                183412 non-null  object
 7   member_birth_year        175147 non-null  float64
 8   member_gender            175147 non-null  object
 9   bike_share_for_all_trip  183412 non-null  object
 10  day                      183412 non-null  object
 11  hour                     183412 non-null  int64
```

```
 12  dur_per_minute           183412 non-null  int64
 13  age                      175147 non-null  float64
dtypes: datetime64[ns](2), float64(2), int64(4), object(6)
memory usage: 19.6+ MB
```

[17]:
```
# Lets check our new added column names
for i,v in enumerate(df.columns):
    print(i,v)
```

```
0 duration_sec
1 start_time
2 end_time
3 start_station_name
4 end_station_name
5 bike_id
6 user_type
7 member_birth_year
8 member_gender
9 bike_share_for_all_trip
10 day
11 hour
12 dur_per_minute
13 age
```

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

Let make ages between:

12-20 = Teenage

21-30 = Young Adult

31- 49 = Adult

50+ Seniors

[18]:
```
# add age_group catagory based on users age i.e ( teenage (13-19),␣
↪Young_Adult(20-30), Adult (31-49), Senior(50+)
category = pd.cut(df.age, bins=[12, 21, 31, 50, 140], labels=["Teenage", "Yound␣
↪Adult", "Adult", "Senior"])
df.insert(14, "age_group", category)
```

[19]:
```
df.describe()
```

[19]:
```
        duration_sec         bike_id  member_birth_year            hour  \
count  183412.000000  183412.000000      175147.000000  183412.000000
mean      726.078435    4472.906375        1984.806437      13.458421
std      1794.389780    1664.383394          10.116689       4.724978
```

```
min        61.000000      11.000000          1878.000000        0.000000
25%       325.000000    3777.000000          1980.000000        9.000000
50%       514.000000    4958.000000          1987.000000       14.000000
75%       796.000000    5502.000000          1992.000000       17.000000
max     85444.000000    6645.000000          2001.000000       23.000000

       dur_per_minute            age
count   183412.000000  175147.000000
mean        11.609393      37.193563
std         29.908067      10.116689
min          1.000000      21.000000
25%          5.000000      30.000000
50%          8.000000      35.000000
75%         13.000000      42.000000
max       1424.000000     144.000000
```

4.2 Univariate Exploration

[20]:
```
# Lets check our column names
for i,v in enumerate(df.columns):
    print(i,v)
```

```
0 duration_sec
1 start_time
2 end_time
3 start_station_name
4 end_station_name
5 bike_id
6 user_type
7 member_birth_year
8 member_gender
9 bike_share_for_all_trip
10 day
11 hour
12 dur_per_minute
13 age
14 age_group
```

Let us start with the usage of the bikes and find out when the most trips are taken with repect time_start i.e hours and day of the week.

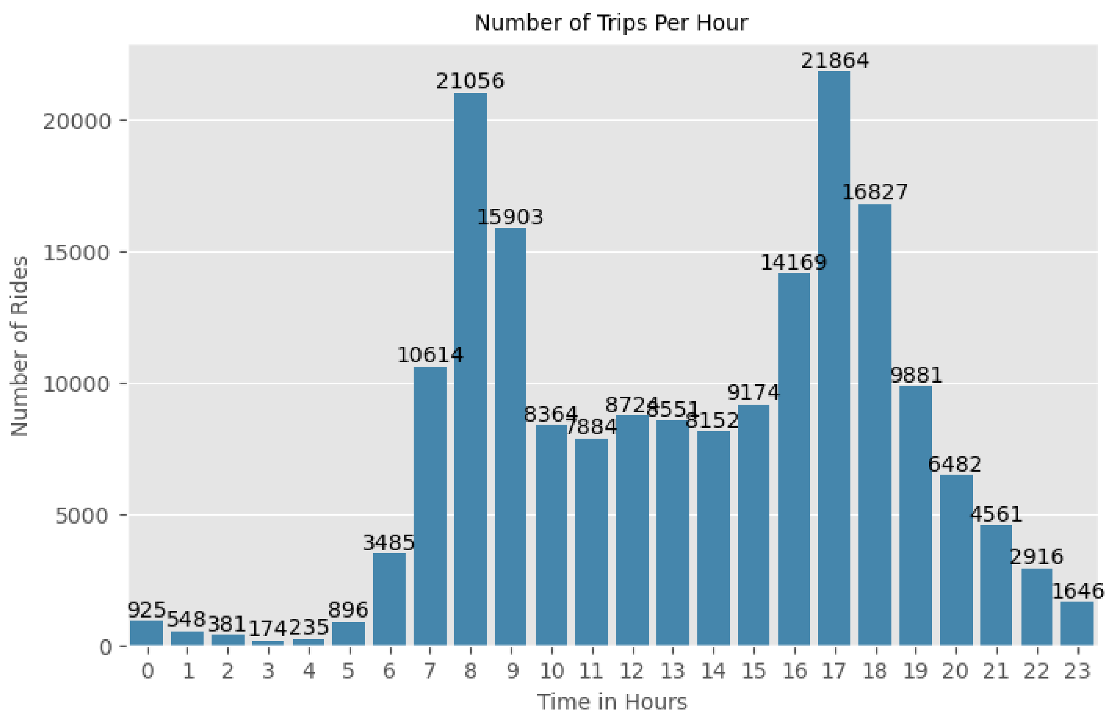
[21]:
```
base_color = sb.color_palette()[1]
```

Ride Frequency by hours

[22]:
```
# univirate analysis
# let take a look at the trip duration per hour frequence
plt.figure(figsize=(8, 5))
```

```
myplot = sb.countplot(data= df,
              x=df['hour'].sort_values(ascending=True),
color= base_color)

myplot.bar_label(myplot.containers[0])
plt.title("Number of Trips Per Hour", size = 10)
plt.xlabel('Time in Hours', size = 10)
plt.ylabel("Number of Rides", size = 10);
```

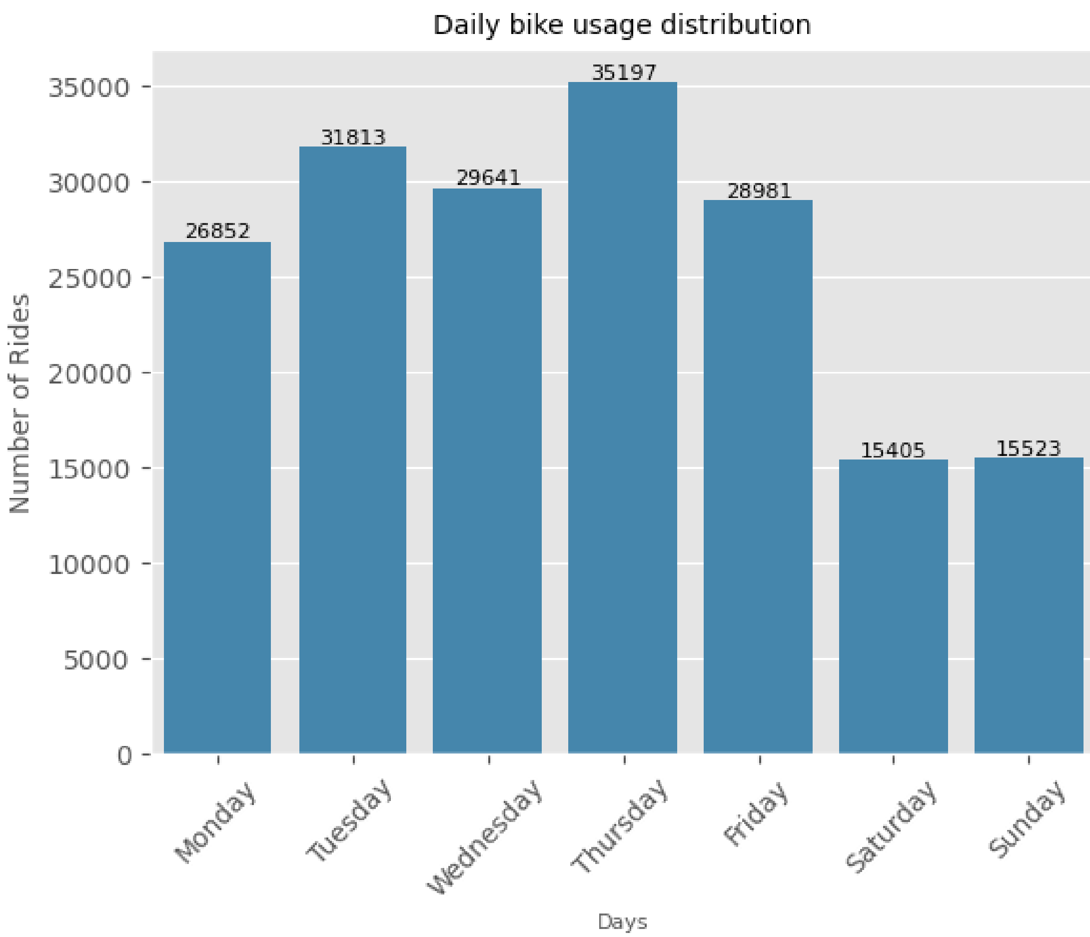


Observation The 8th, 9th, 17th and 18th hours have the highest trip records. This is expected as it can be linked to morning rush and closing hour from work.

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

[23]:
```
# let take a look at the average trip duration per day frequence
def horizontal_bar():
    order_days =␣
 ↪["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"]
    myplot=sb.countplot(x='day',
                        data=df, color=base_color, order= order_days)
    myplot.bar_label(myplot.containers[0], size = 8)
    plt.xticks(rotation=46)
    plt.title("Daily bike usage distribution", size = 10)
```

```
    plt.xlabel("Days", size = 8)
    plt.ylabel("Number of Rides", size = 10)
horizontal_bar()
```



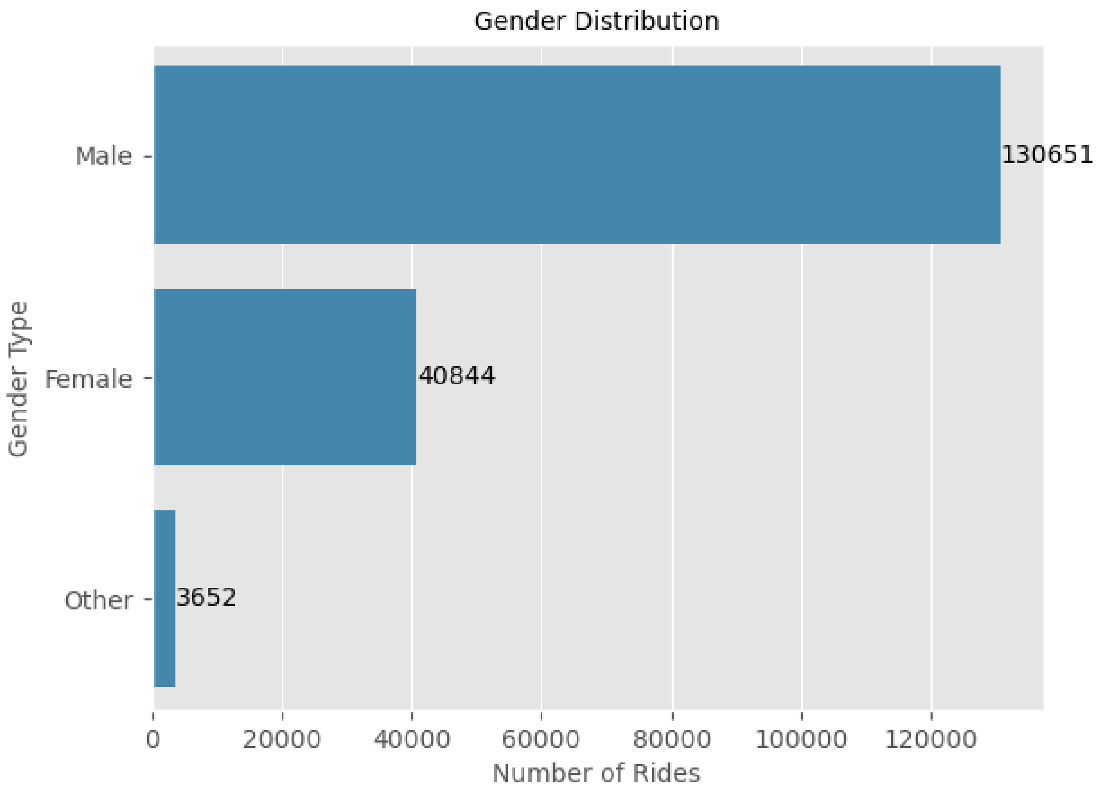
Observations Most of the trips were taken Thrusday, followed by Tuesday. Weekend (sat, Sun) have least trips compared to all the weekdays.

5 which gender is the most predominat in our data?

[24]:
```
#which gender is the most predominat in our data
sex_order =df.member_gender.value_counts().index
myplot = sb.countplot(y='member_gender', data= df, color=base_color,␣
↪order=sex_order)
myplot.bar_label(myplot.containers[0], size=10)
plt.title("Gender Distribution", size = 10)
plt.xlabel("Number of Rides", size = 10)
```

```
plt.ylabel("Gender Type", size = 10)
```

[24]: Text(0, 0.5, 'Gender Type')



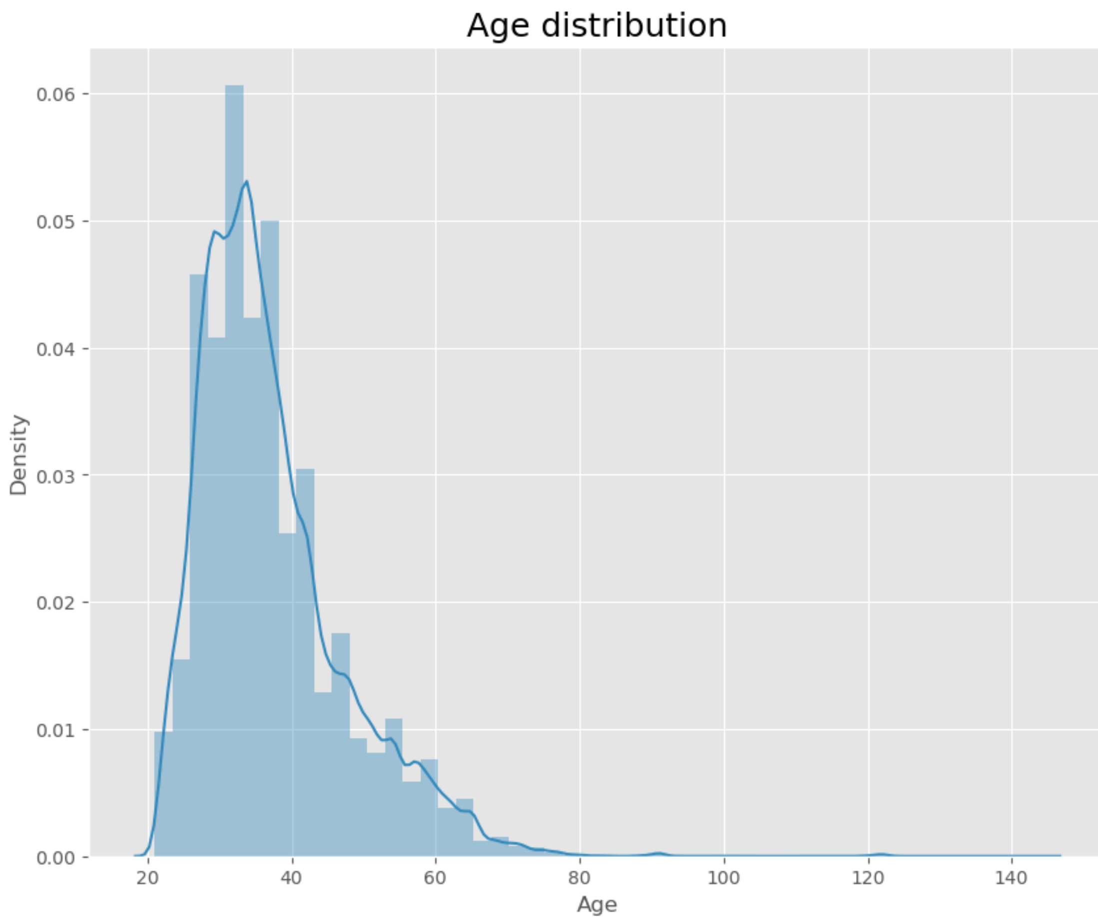
Observations Most trips were made by males

5.1 lets investigate age distribution and see what it looks like

[25]:
```
# Investigating the distribution of age
rcParams['figure.figsize'] = 10,8
x = df["age"].values
sb.distplot(x, color= base_color)
plt.title("Age distribution", size =18)
plt.xlabel("Age")
```

```
/Users/mabdulahi/opt/anaconda3/lib/python3.9/site-
packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

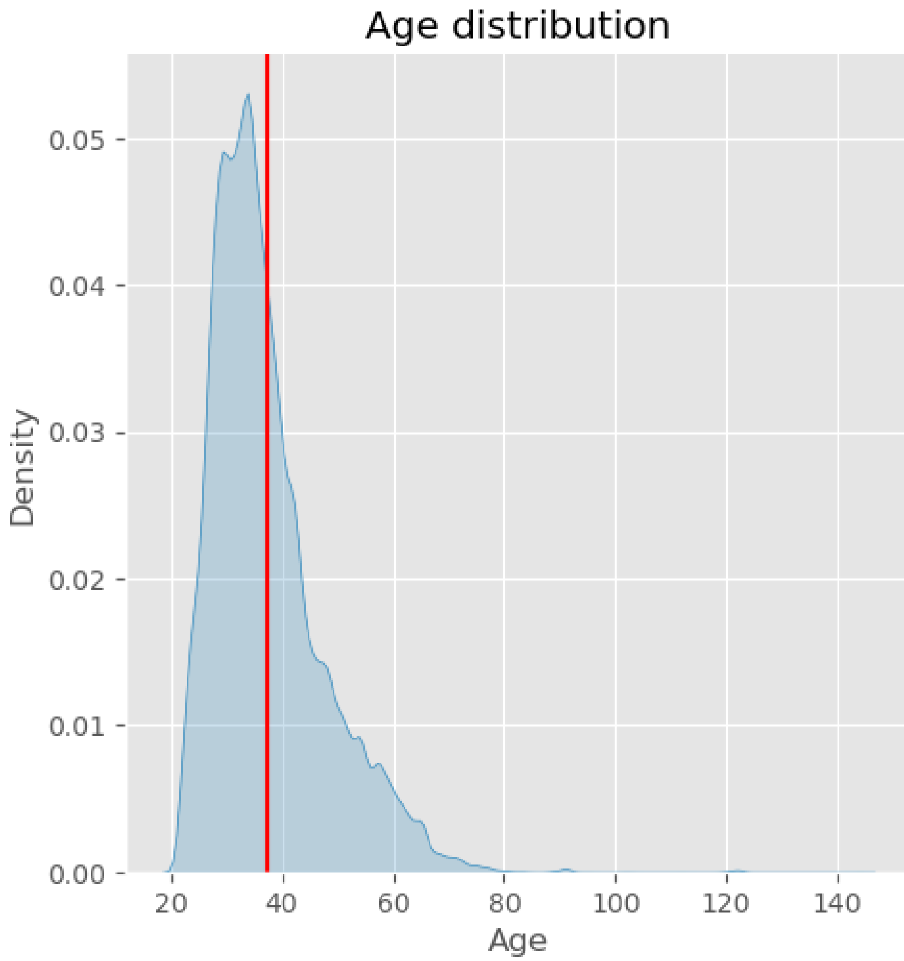
[25]: Text(0.5, 0, 'Age')



[26]:
```
# Let us know check out the age distribution by adding the mean.
rcParams['figure.figsize'] = 10,8
x = df['age'].values
sb.displot(df, x="age", kind="kde", fill= True, color= base_color)

# Calculating the mean
mean = df['age'].mean()

#ploting the mean
plt.axvline(mean, 0,2, color = 'red')
plt.title("Age distribution")
plt.xlabel("Age");
```

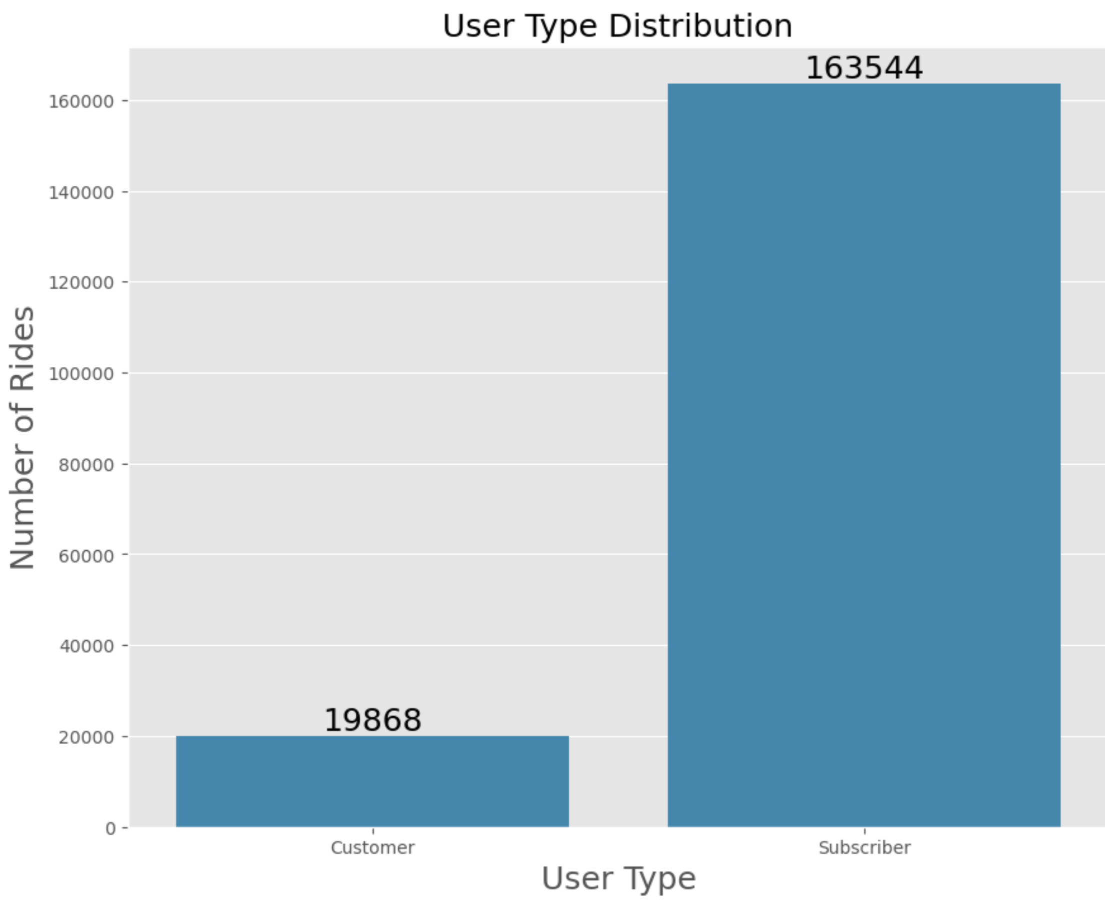


Observation: In thes graph we can observe that the user age is right skewed distribution and the average user age is about 37 years old give or take.

Which user types made the most trips?

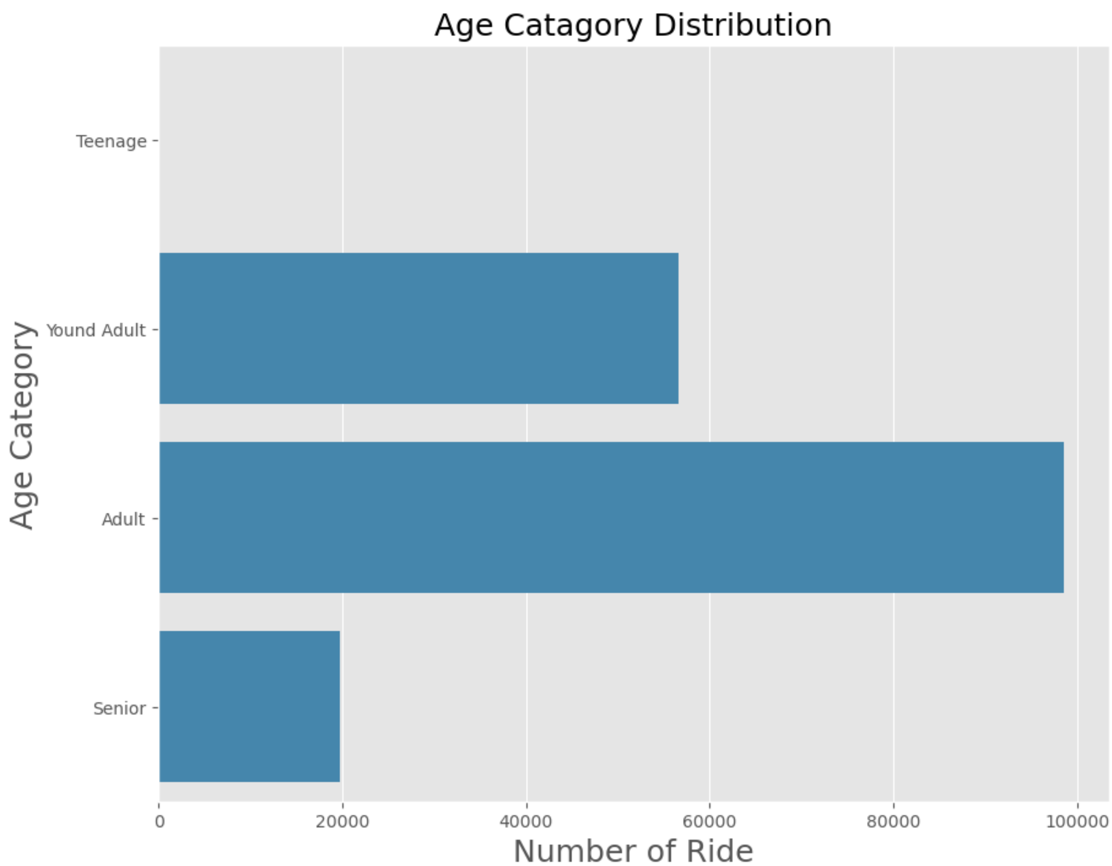
[27]:
```
myplot = sb.countplot(data = df, x = 'user_type', color = base_color)
myplot.bar_label(myplot.containers[0], size=18)
plt.title("User Type Distribution", size = 18)
plt.xlabel("User Type", size = 18)
plt.ylabel("Number of Rides", size = 18)
```

[27]:
```
Text(0, 0.5, 'Number of Rides')
```



Observation From this visual graph we see that Subscribers have made the 7x trips than customers in our data.

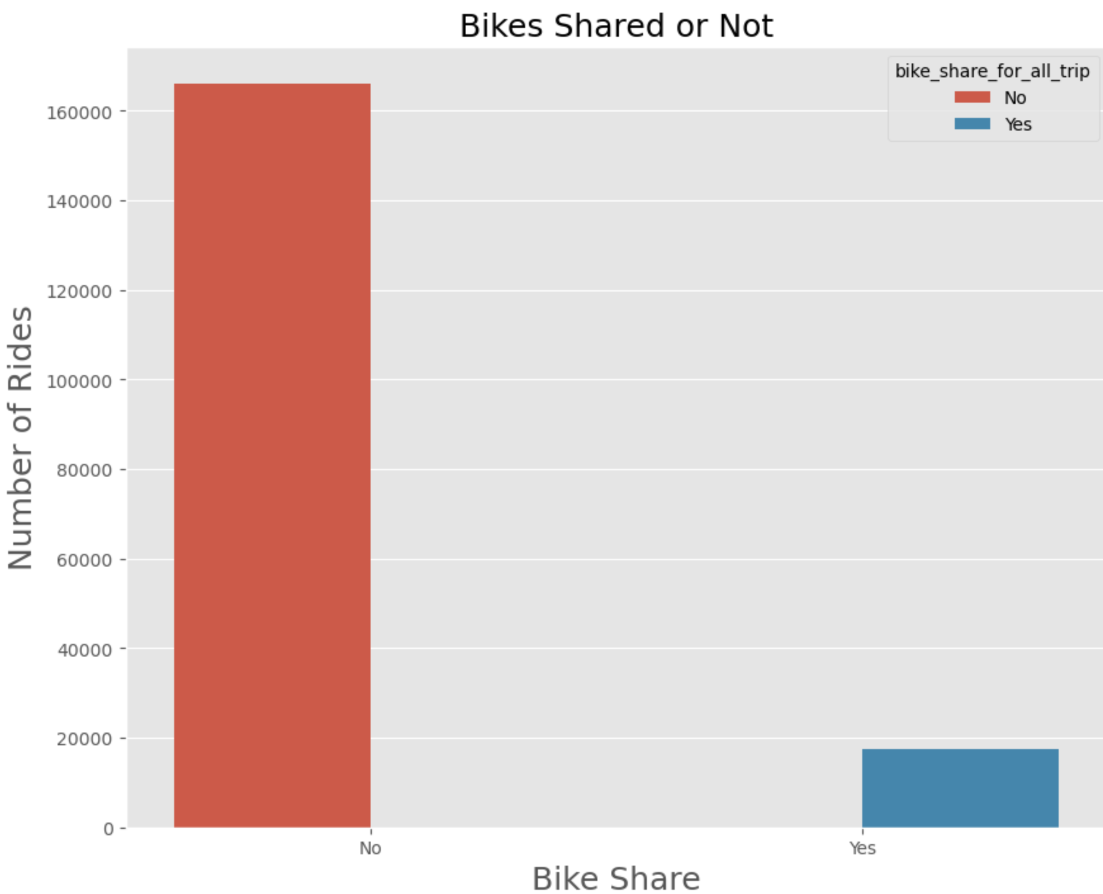
[28]:
```
# Let's plot the histogram of members age to see what the distribution of age␣
↪looks like
ax = sb.countplot(data = df, y = 'age_group', color = base_color)
plt.title("Age Catagory Distribution", size=18)
plt.xlabel("Number of Ride",size=18)
plt.ylabel("Age Category", size=18);
```



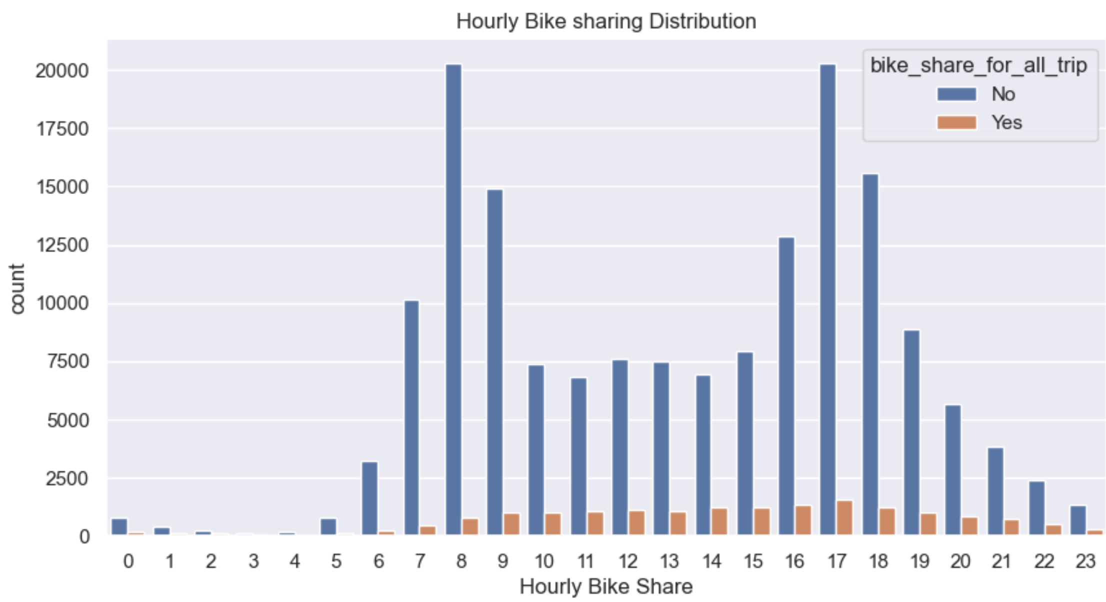
Count the number of bikes shared for all trips vs Not shared?

[29]:
```
## Let's Check the count the number of bike shared for all trips vs Not shared.
myplot = sb.countplot(x = df.bike_share_for_all_trip,
                          hue = 'bike_share_for_all_trip',
                          data = df)
plt.title("Bikes Shared or Not", size=18)
plt.xlabel("Bike Share",size=18)
plt.ylabel("Number of Rides", size=18)
```

[29]:
```
Text(0, 0.5, 'Number of Rides')
```



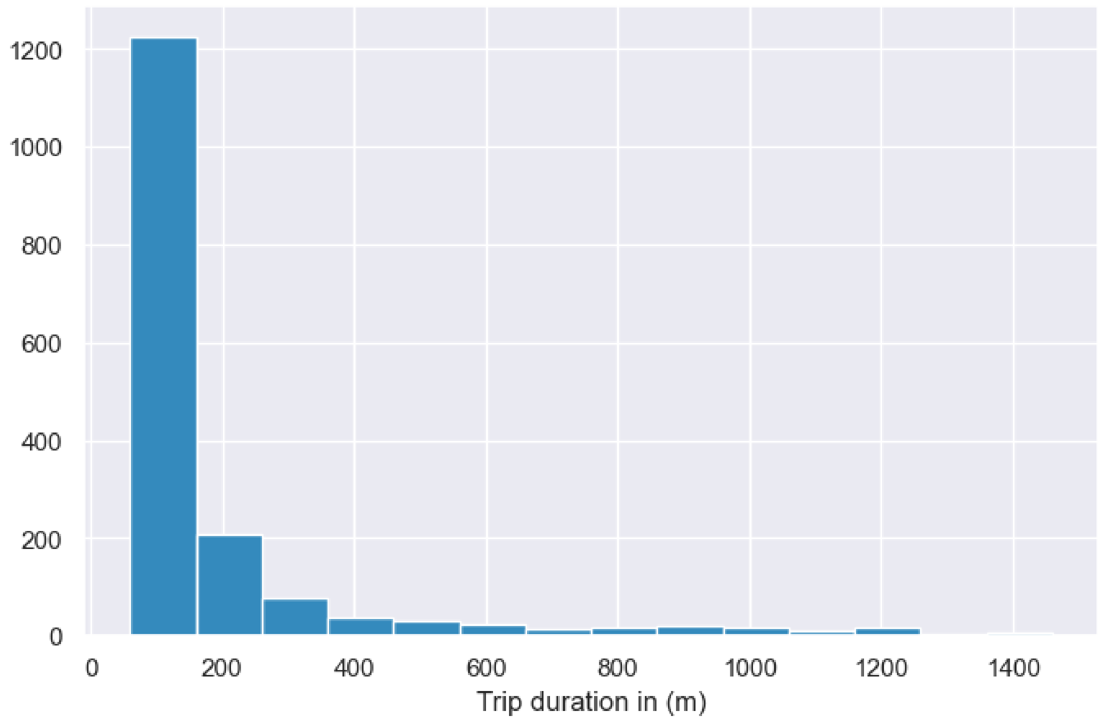
[30]:
```
# Let's check the countplot distribution of all shared rides by hourly start␣
↪and hourly end and compare the regular rides?.
plt.figure(figsize = (10,5))
sb.set(style = "darkgrid")
sb.countplot(x =df["hour"].sort_values(ascending=True), hue =␣
↪'bike_share_for_all_trip', data = df)
plt.title('Hourly Bike sharing Distribution')
plt.xlabel('Hourly Bike Share');
```



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

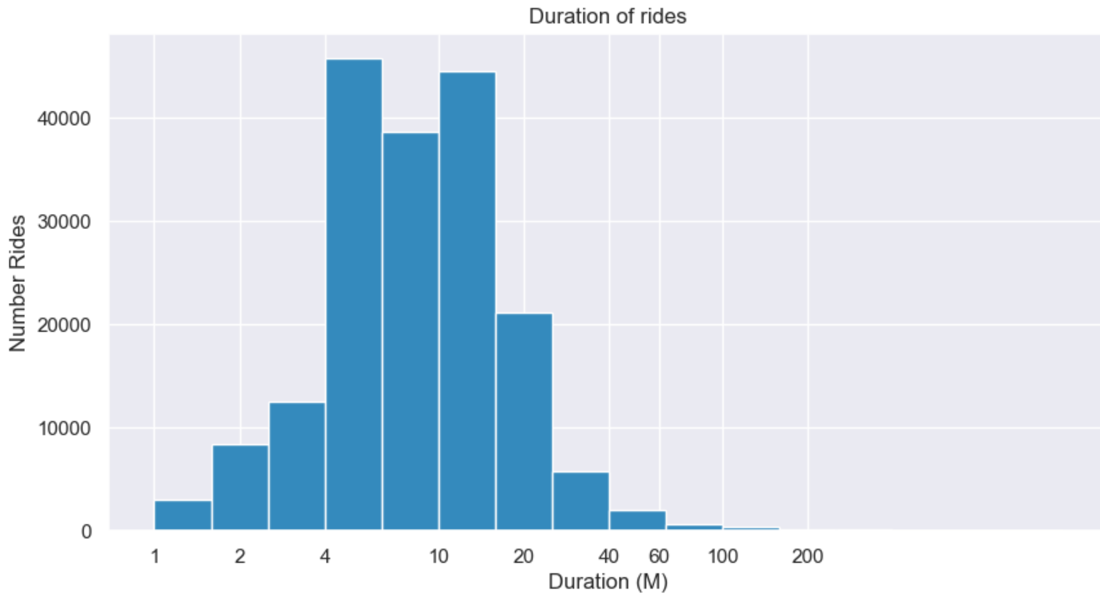
5.1.1 Distribution of Ride Duration

[31]:
```
# investigation Ride Duration
def histogram():
    plt.figure(figsize=[8, 5])
    bins = np.arange(60, df['dur_per_minute'].max()+100, 100)
    plt.hist(df['dur_per_minute'], bins = bins, color= base_color);
    plt.xlabel('Trip duration in (m)');
histogram()
```



There's a long tail in the distributio and the duration skewed so, I am going to put it on a log scale and use and use smaller binsize to get a mor detailed distribution.

[32]:
```
# investigation Ride Duration
def histogram():
    plt.figure(figsize=[10,5])
    ticks = [1, 2, 4, 10, 20, 40, 60, 100, 200 ]
    bin_edges = 10 ** np.arange(0.0, np.log10(df.dur_per_minute.max())+0.2, 0.2)
    plt.hist(data = df, x = 'dur_per_minute', bins = bin_edges,color =␣
 ↪base_color)
    plt.xscale('log')
    plt.xticks(ticks, ticks)
    plt.xlabel('Duration (M)')
    plt.ylabel("Number Rides")
    plt.title("Duration of rides");
histogram()
```



[33]:
```
df["dur_per_minute"].mean()
```

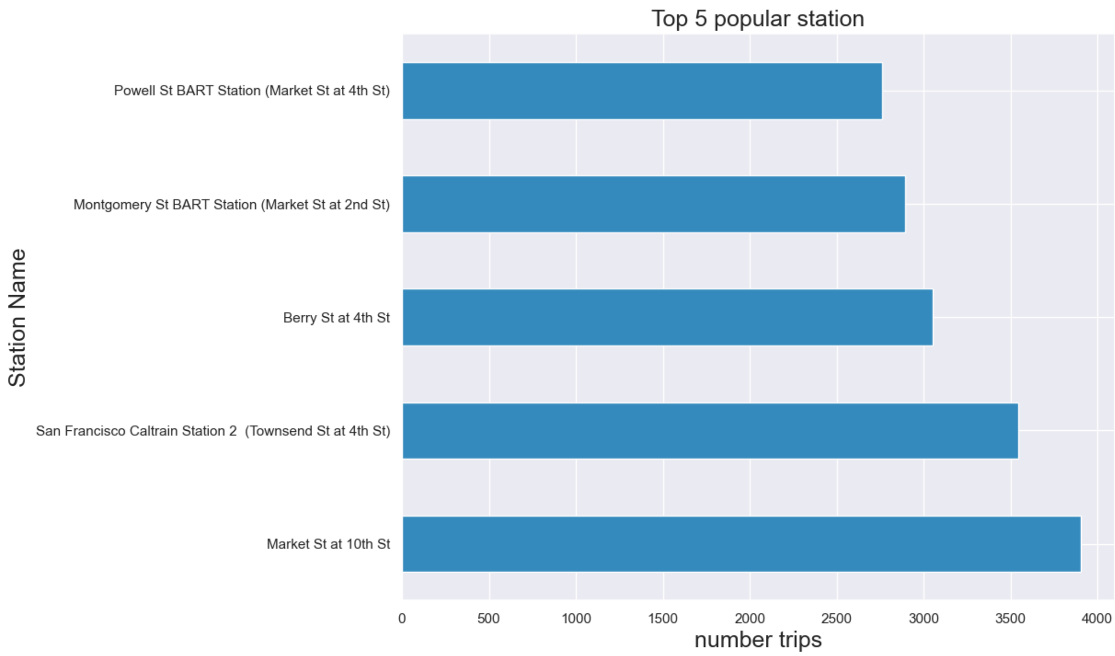
[33]: 11.60939306043225

Observation 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 also confirmed the average trip duration is about 12 minutes.

5.1.2 Investigate the most and least popular bike stations

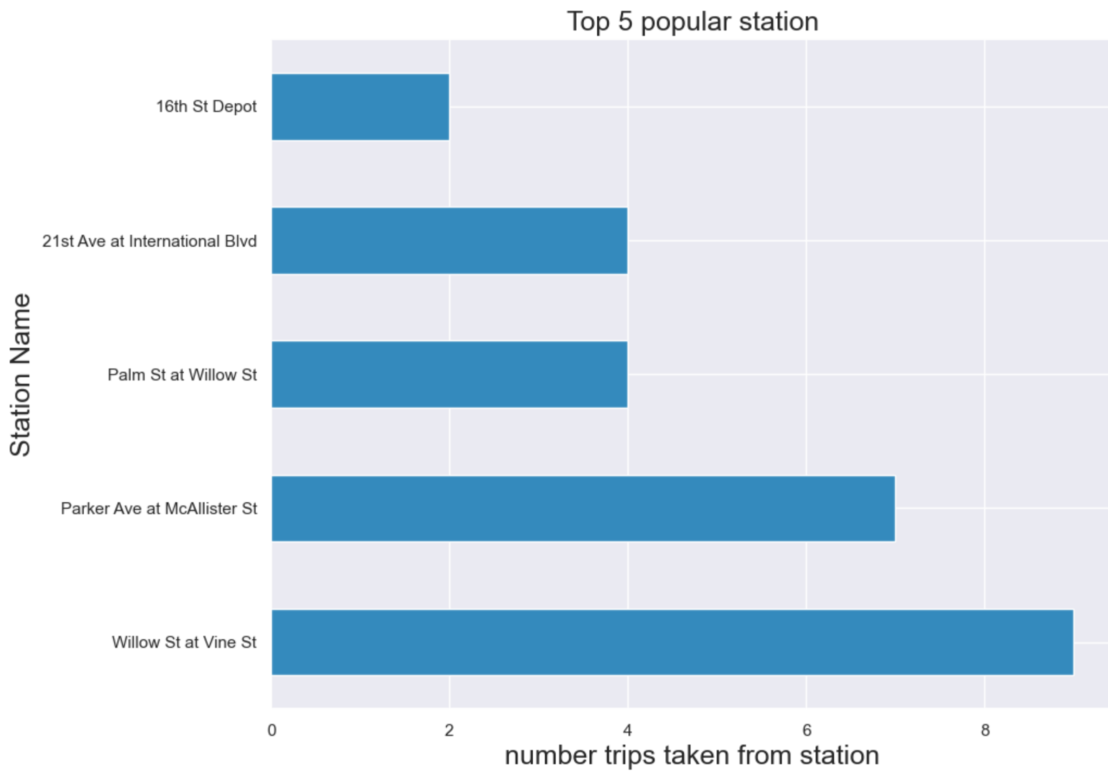
[34]:
```
# check top 5 most popular stations
top5_stations = df["start_station_name"].value_counts()
top5 = top5_stations.head(5).plot.barh(color=base_color)
plt.title("Top 5 popular station", size =18)
plt.xlabel("number trips", size=18)
plt.ylabel("Station Name", size=18)
```

[34]: Text(0, 0.5, 'Station Name')



[35]:
```python
# check least 5 worest bike stations
least5_stations = df["start_station_name"].value_counts()
least5 = top5_stations.tail(5).plot.barh(color=base_color)
plt.title("Top 5 popular station", size =18)
plt.xlabel("number trips taken from station", size=18)
plt.ylabel("Station Name", size=18)
```

[35]:
```
Text(0, 0.5, 'Station Name')
```



Observation `Market St a 10th st` is the most popular station while `Willow St At Vin St` is least worest bike station as the figures show.

5.1.3 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 about 12 minutes. most trips were made by adults age between 31 to 49.

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

`stations` Market St a 10th st is the most popular station while 'Willow St At Vin St is least popular.

5.1.4 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?

I saw a long tail on of trip duration, so I applied A logarithmic scale transformation on the the trip duration to get a more detailed look at data.

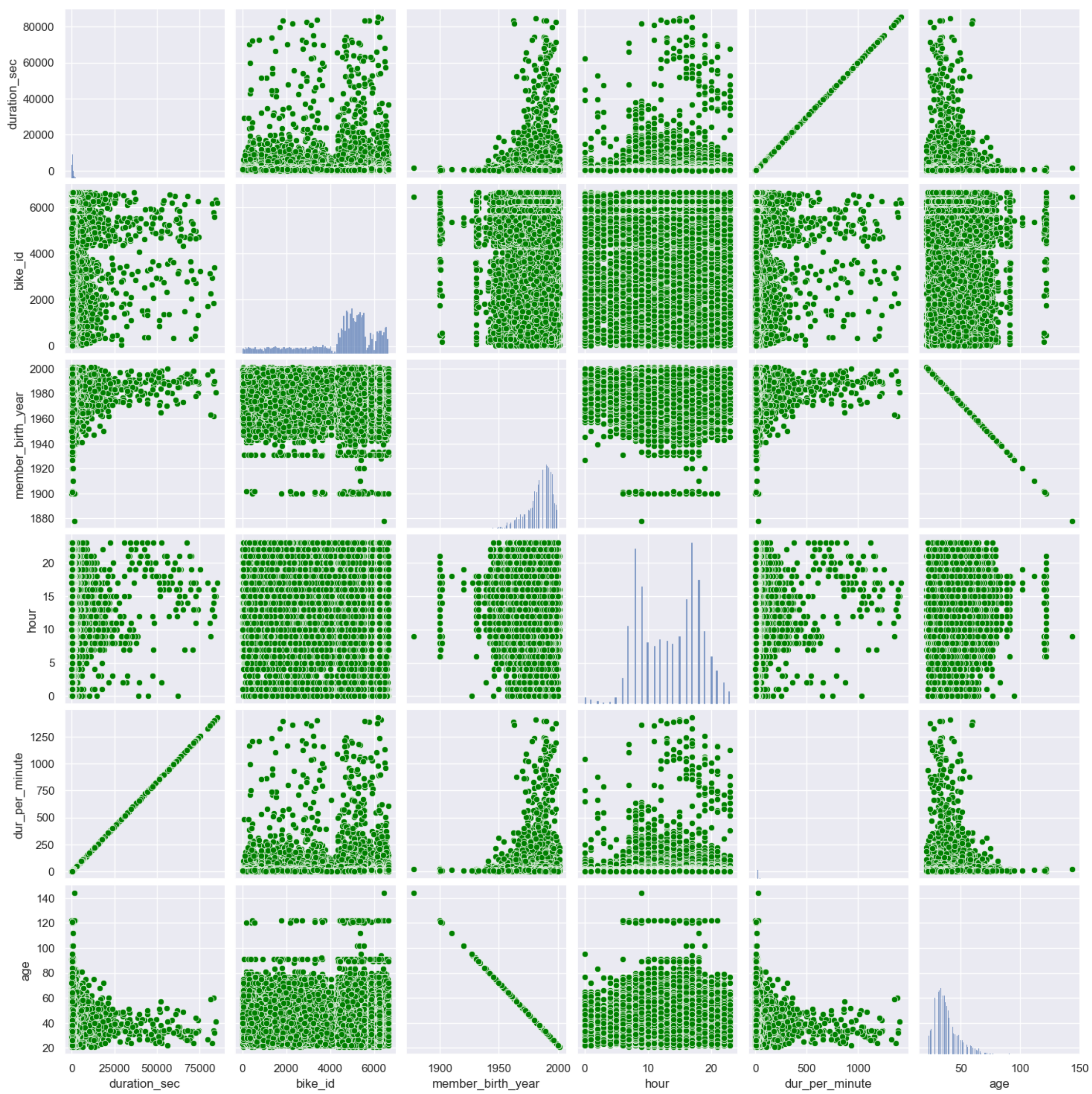
5.2 Bivariate Exploration

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

6 To start of let us take a look at the relationships between variables

[36]:
```
# Pairplot
sb.pairplot(df, plot_kws={'color':'green'})
```

[36]: <seaborn.axisgrid.PairGrid at 0x7f77ca382970>

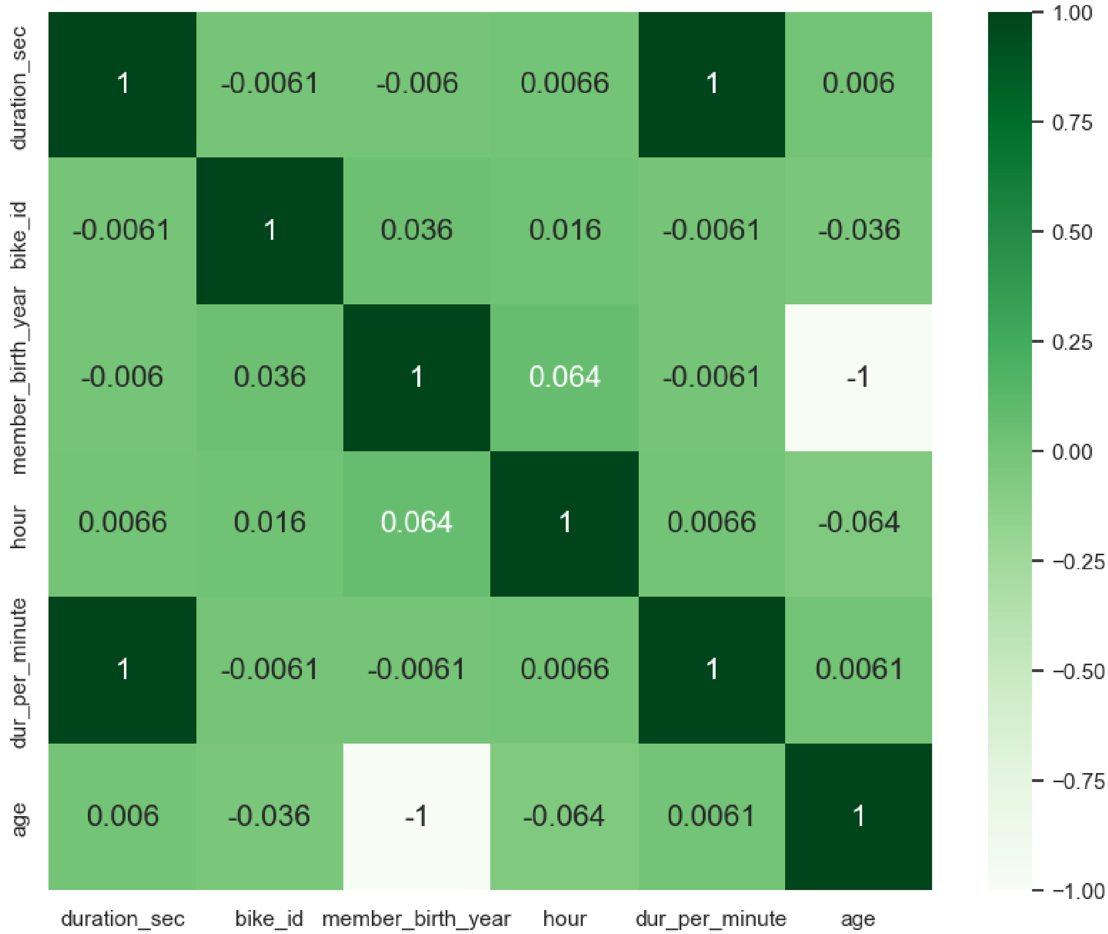


[37]:
```
df.columns
```

[37]:
```
Index(['duration_sec', 'start_time', 'end_time', 'start_station_name',
       'end_station_name', 'bike_id', 'user_type', 'member_birth_year',
       'member_gender', 'bike_share_for_all_trip', 'day', 'hour',
       'dur_per_minute', 'age', 'age_group'],
      dtype='object')
```

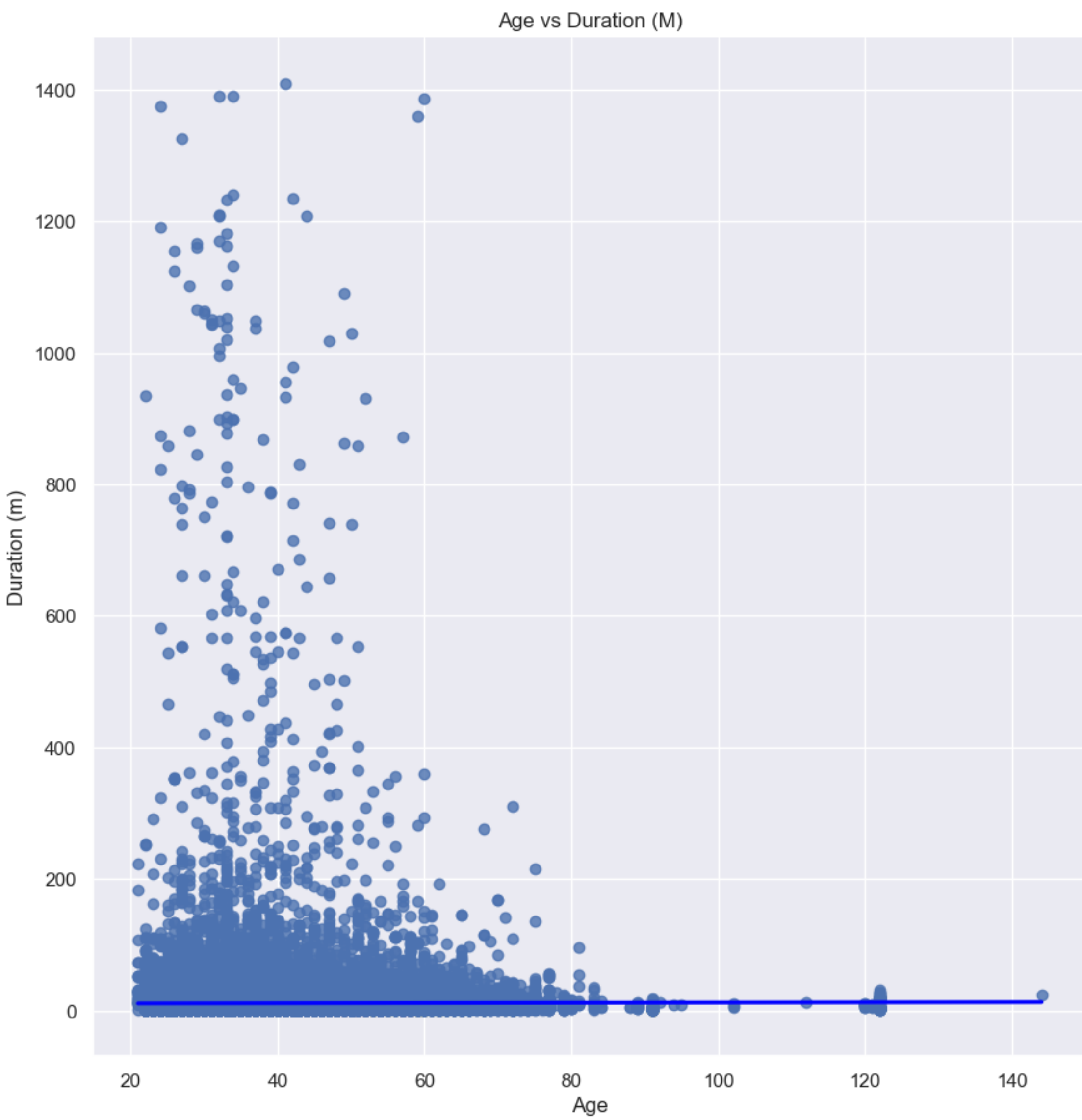
[38]:
```
sb.heatmap(df.corr(method ='pearson'), annot=True,
                 annot_kws={'size': 15},
          cmap="Greens")
```

[38]: <AxesSubplot:>



lets find the realation between age and duration in minutes and see if age has any effect on the duration

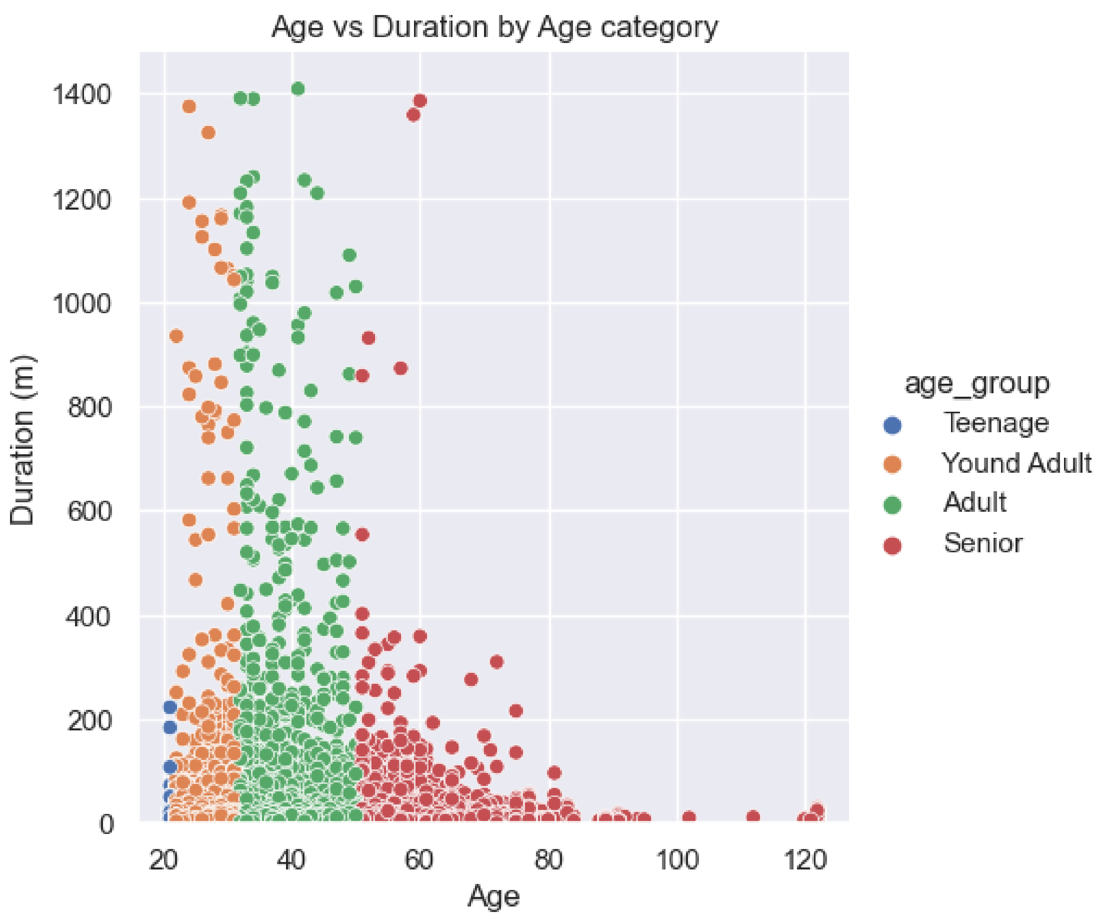
[39]:
```
ax=sb.lmplot(x="age", y="dur_per_minute", data=df, height=9,line_kws={'color':
    'blue'})
ax.set_xlabels("Age")
ax.set_ylabels("Duration (m)")
plt.title("Age vs Duration (M)");
```



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.

[40]:
```
# Relationship between age and duration by age category
sb.relplot(x="age", y="dur_per_minute", hue="age_group", data=df)
plt.ylim(0)
```

```
plt.xlabel("Age")
plt.ylabel("Duration (m)")
plt.title("Age vs Duration (M)");
plt.title("Age vs Duration by Age category");
```



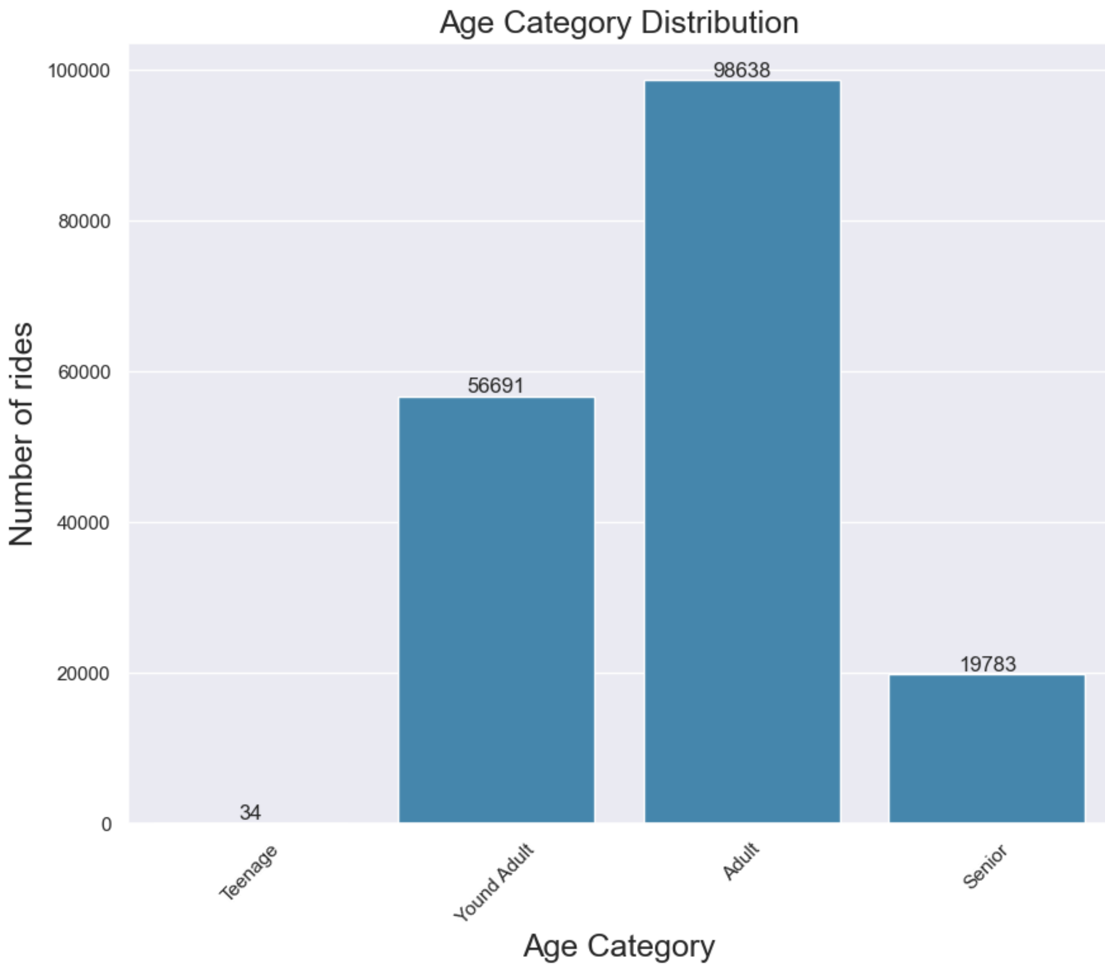
Observation the figure show that as the Age increases the trip duration decreases which we can say that age and duration have inverse relationship.

[41]:
```
df1 =df.groupby("age_group")["age"].count().reset_index()
df1
```

[41]:

	age_group	age
0	Teenage	34
1	Yound Adult	56691
2	Adult	98638
3	Senior	19783

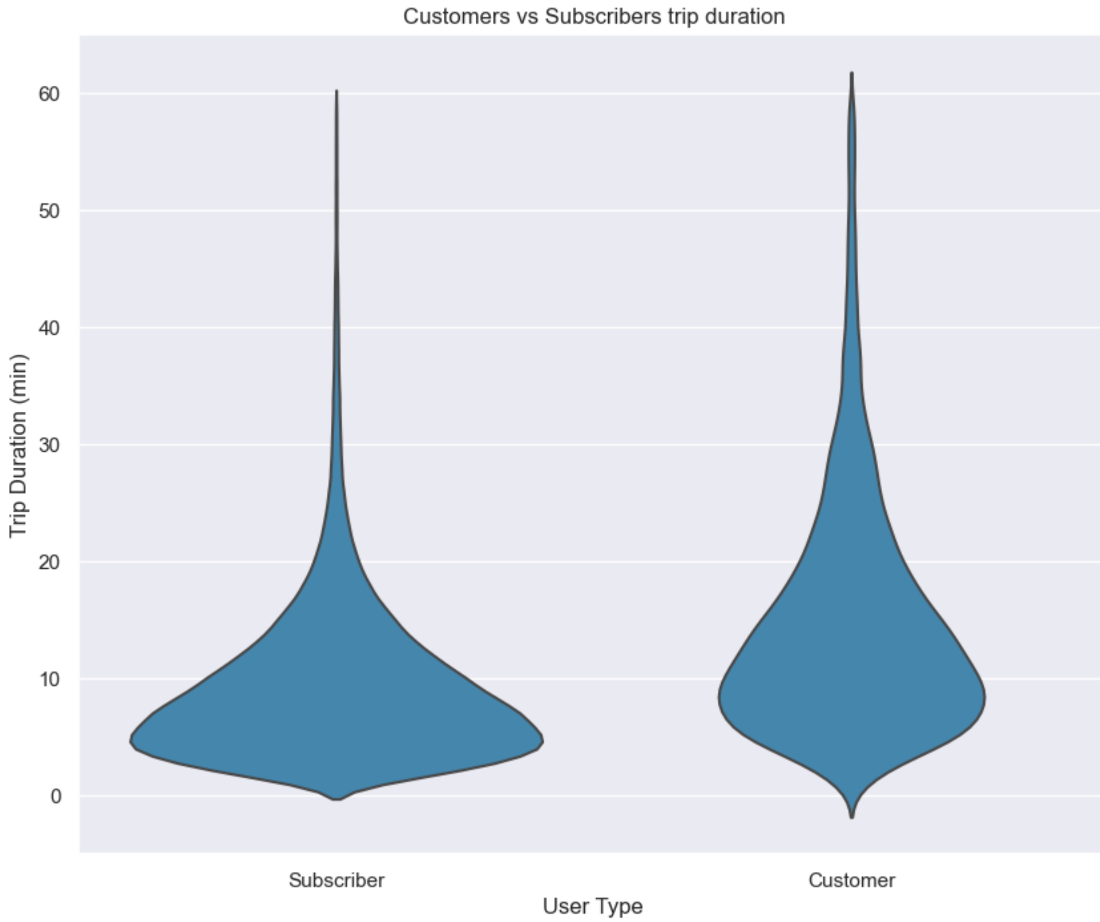
[42]:
```
# coun the number of rides by age category
myplot=sb.barplot(x='age_group',y= 'age',data=df1,color=base_color)
myplot.bar_label(myplot.containers[0])
plt.title('Age Category Distribution', size=18)
plt.xlabel("Age Category",size = 18)
plt.ylabel("Number of rides", size= 18)
plt.xticks(rotation=46);
```



Observation Adults ages between 31-49 made the the majority ride trips.

[43]:
```
# Chech trip duration  between Customers and Subscribers
# we will only consider trips less than an hour to get more detailed data.
df1 = df.query("dur_per_minute < 60")
sb.violinplot(data=df1, x="user_type", y="dur_per_minute", color=base_color,␣
↪inner=None)
#ax.bar_label(ax.containers[0])
plt.title('Customers vs Subscribers trip duration')
```

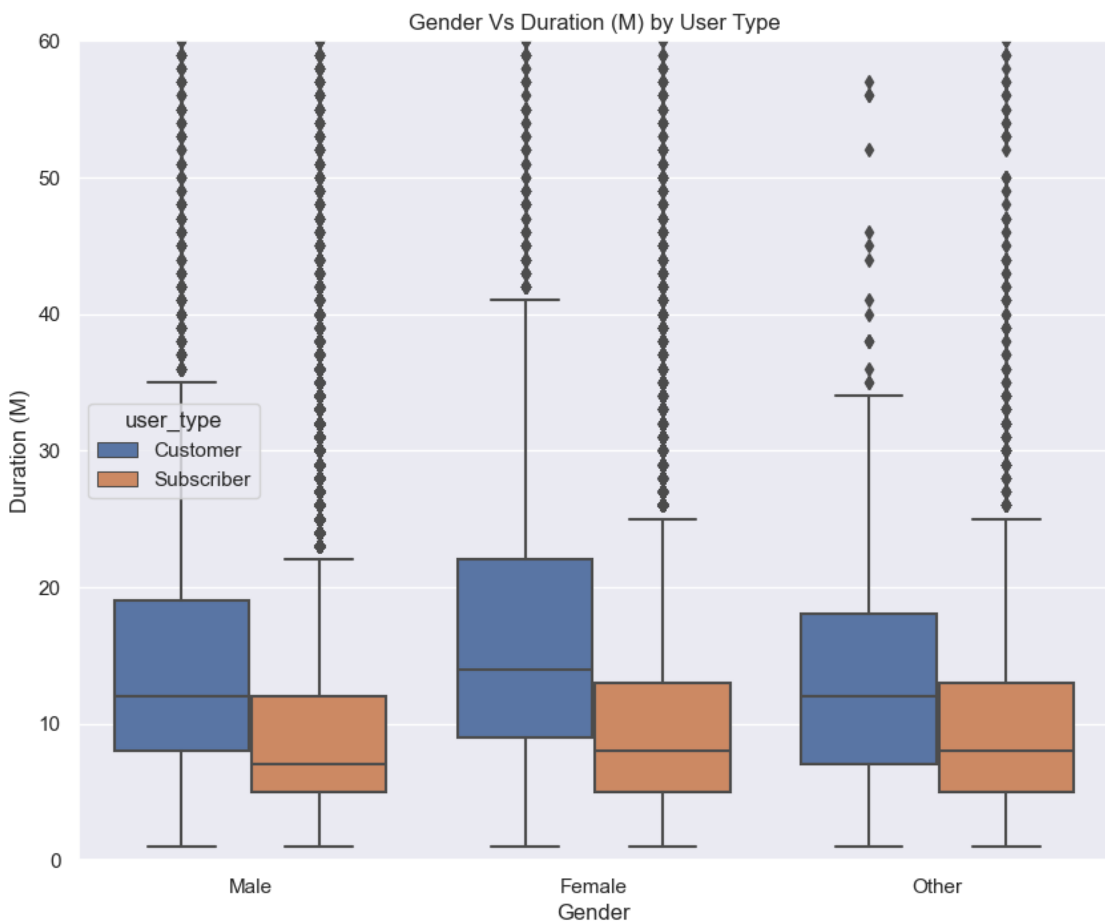
```
plt.xlabel('User Type')
plt.ylabel('Trip Duration (min)');
```



Observation customers user types take 2x longer trips than subscribers.

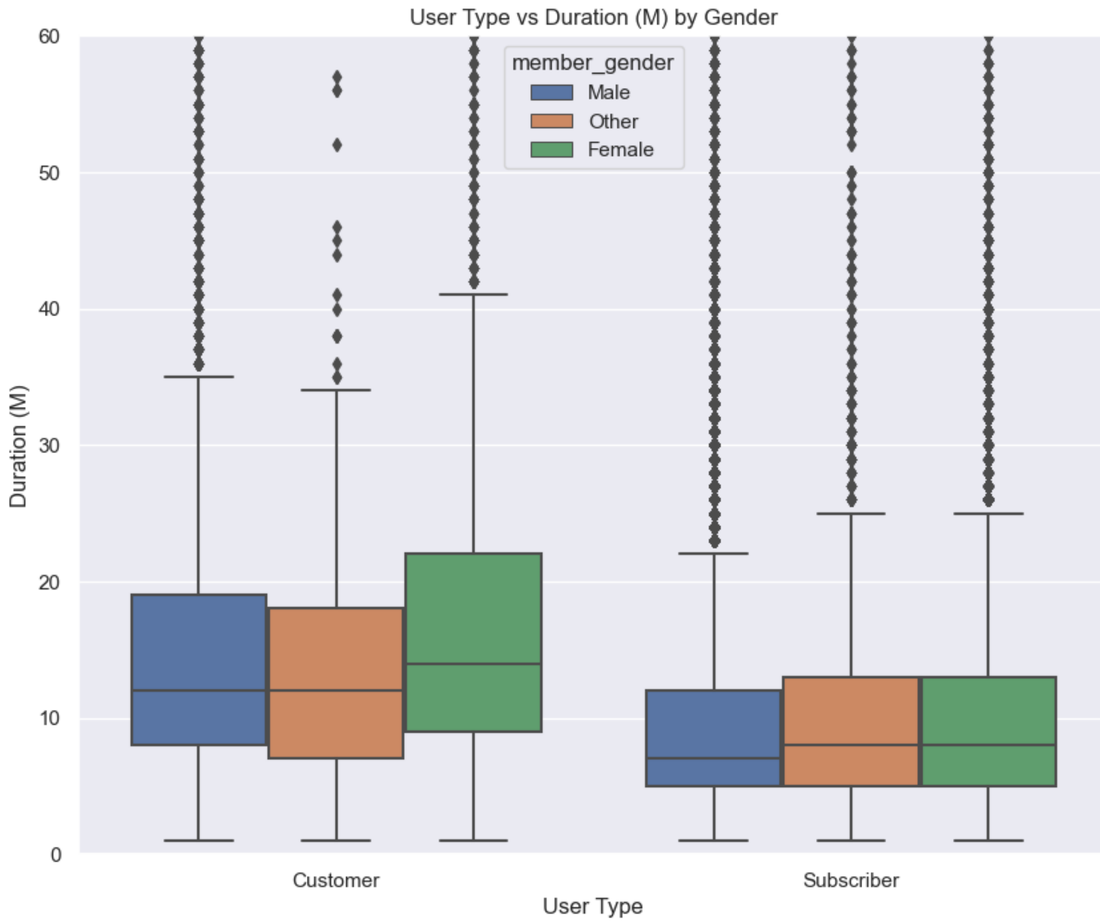
Let's look at the the gender and duration relationships in terms of user-type.

[44]:
```
# Investigating the distribution gender and duration by user type
sb.boxplot(x='member_gender', y='dur_per_minute', data = df, hue="user_type", ␣
↪order=['Male', 'Female', 'Other'])
plt.ylim(0, 60)
plt.title('Gender Vs Duration (M) by User Type')
plt.xlabel('Gender')
plt.ylabel('Duration (M)');
```



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

[45]:
```
# Investigating the distribution of user type and duration by gender
sb.boxplot(x='user_type', y='dur_per_minute', data = df, hue="member_gender")
plt.ylim(0, 60)
plt.title('User Type vs Duration (M) by Gender')
plt.xlabel('User Type')
plt.ylabel('Duration (M)');
```



Observations Looking at customer boxlot, females take long trips followed by male.

On other hand, the subscriber boxplot depicts that female and other genders are leveled while the male duration is smalled compared to female and other genders. Therefore, we can say, from this figure that females take longer trips than any other gender.

6.0.1 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?

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

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.

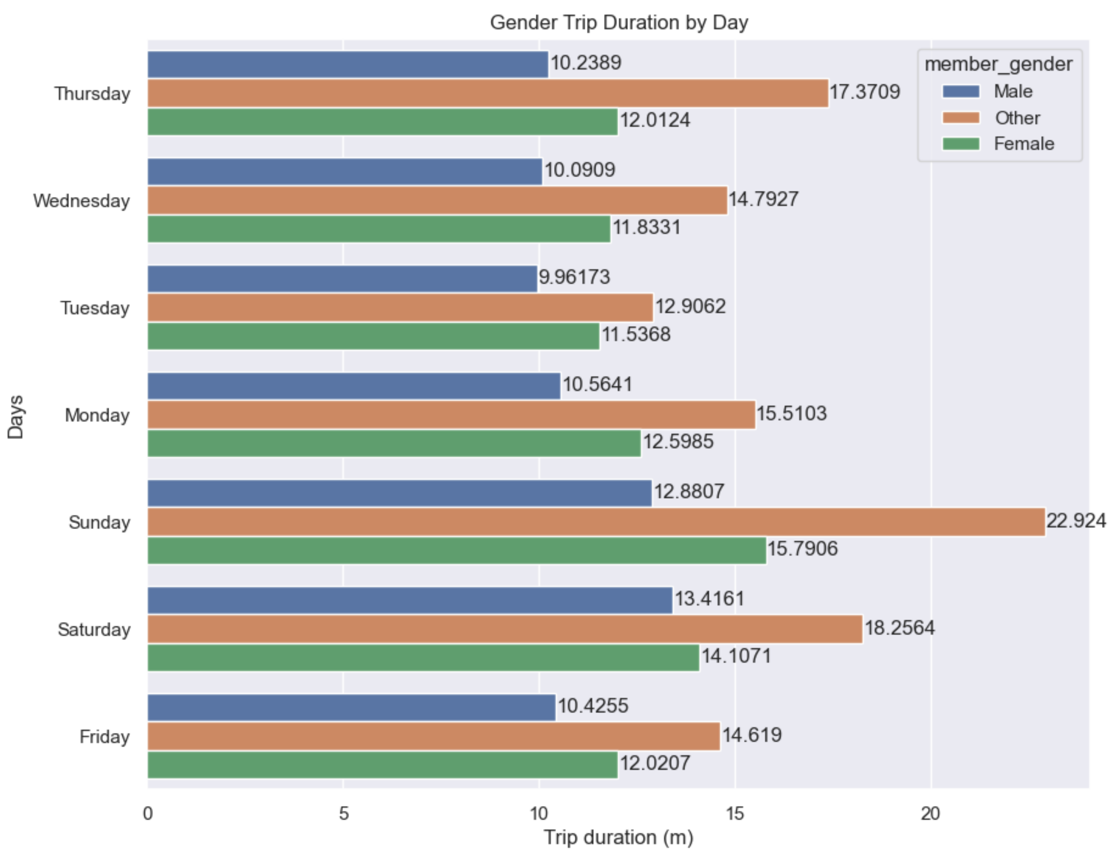
6.0.2 Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

Looking at the relationship between gender type and duration per minute. The garph showed that females and Other genders take longer trip durations than Male which I was surprised because in the previous section we saw that most trips were made by men.

6.1 Multivariate Exploration

[46]:
```
# Compare daily trip average duration by gender
rcParams['figure.figsize'] = 10,8
ax = sb.barplot(y="day", x="dur_per_minute",
                hue="member_gender",
                data=df, ci=False)
for container in ax.containers:
    ax.bar_label(container)
plt.title('Gender Trip Duration by Day')
plt.xlabel('Trip duration (m)')
plt.ylabel('Days')
```

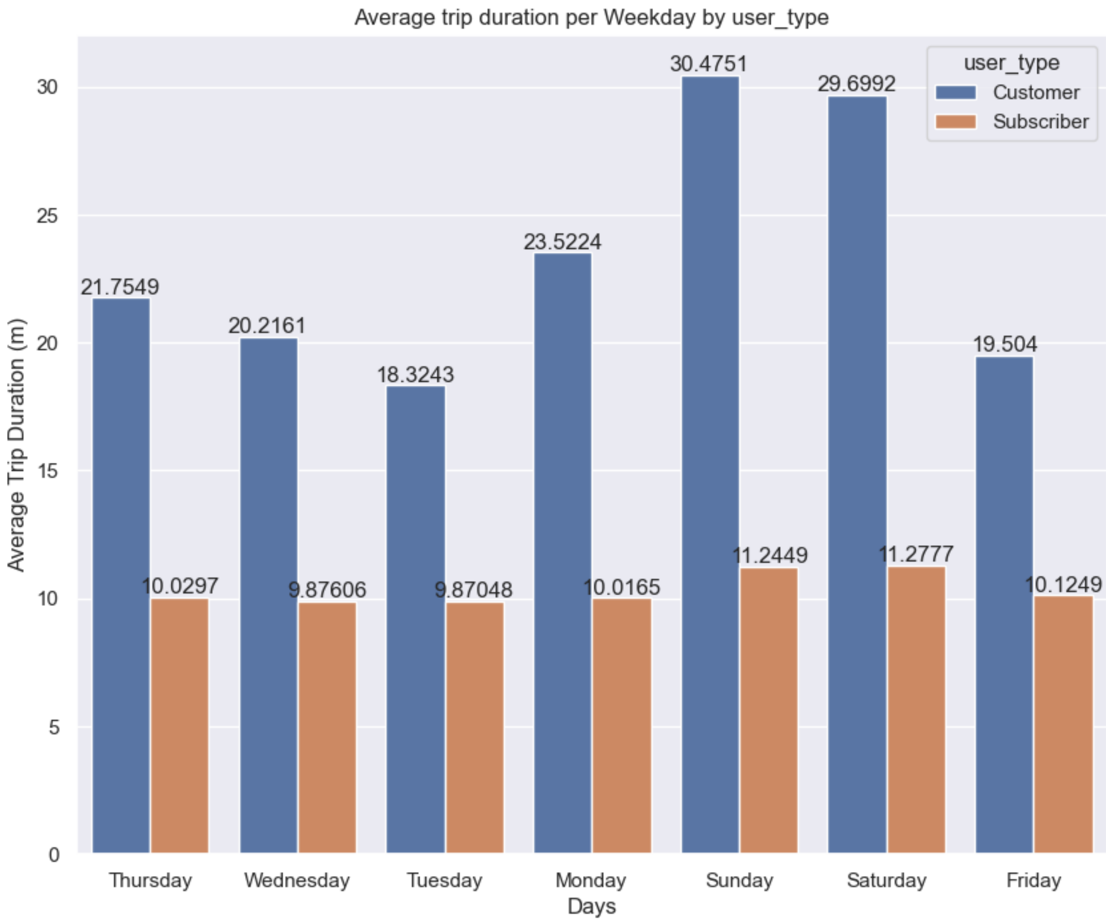
[46]:
```
Text(0, 0.5, 'Days')
```



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.

[47]:
```
# Compare Average trip duration per Weekday by user_type
ax = sb.barplot(data = df, x ='day', y = 'dur_per_minute', hue =␣
 ↪'user_type',ci= False)
for container in ax.containers:
    ax.bar_label(container)
plt.title('Average trip duration per Weekday by user_type')
plt.xlabel('Days')
plt.ylabel('Average Trip Duration (m)')
plt.xlabel('Days')
```

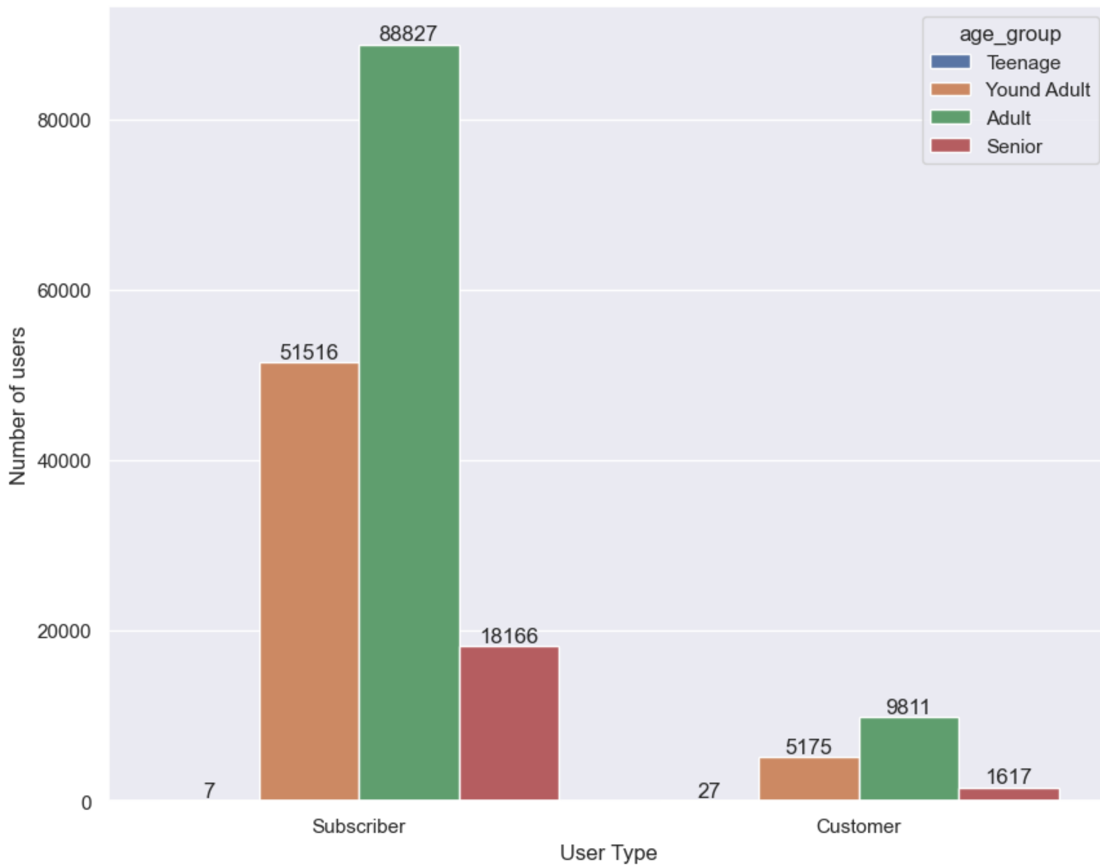
[47]:
```
Text(0.5, 0, 'Days')
```



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

6.1.1 Display the total number of users_types and their age catagory.

[48]:
```
#display the numbers of users, their user_type and and their age category
ax = sb.countplot(data=df, x="user_type", hue="age_group",
                            order=df.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 7 teenagers (ages12-20), 51516 youth adults ages between 21 through 30. 88827 Adult subscribers between age 31 and 49, and 18166 seniors age 50+

For Customer user types, we have 5175 young adults 9811 youth, 1617 seniors, and 27 teenagers

[49]:
```
df.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 15 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_name       183215 non-null  object
 4   end_station_name         183215 non-null  object
 5   bike_id                  183412 non-null  int64
 6   user_type                183412 non-null  object
 7   member_birth_year        175147 non-null  float64
 8   member_gender            175147 non-null  object
 9   bike_share_for_all_trip  183412 non-null  object
 10  day                      183412 non-null  object
 11  hour                     183412 non-null  int64
 12  dur_per_minute           183412 non-null  int64
 13  age                      175147 non-null  float64
 14  age_group                175146 non-null  object
dtypes: float64(2), int64(4), object(9)
memory usage: 21.0+ MB
```

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

6.1.2 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 31-49 in our dataset made the most trips

6.1.3 Were there any interesting or surprising interactions between features?

There wasn'st any interactions that got my attention.

6.2 Conclusions

6.2.1 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 Thrusday, followed by Tuesday. Weekend (sat, Sun) have least trips compared to all the weekdays.

Q4: Which user types take the longest trip with respect duration per minutes. Answer:Customers on average take a longer trip than subscribers.

6.2.2 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 https://stackoverflow.com/questions/67723105/how-to-convert-time-from-24-hour-format-to-12-hour-format-am-pm-with-pandas-p https://dataindependent.com/pandas/pandas-to-datetime-string-to-date-pd-to_datetime/ https://stackoverflow.com/questions/49153253/pandas-rounding-when-converting-float-to-integer

```
[ ]:
```