HW2

September 13, 2025

1 HW2 - Matplotlib Exercises

NOTE: ALL THE COMMANDS FOR PLOTTING A FIGURE SHOULD ALL GO IN THE SAME CELL. SEPARATING THEM OUT INTO MULTIPLE CELLS MAY CAUSE NOTHING TO SHOW UP.

Maximum Number of Points: 10 (2 points per question)

1.0.1 Follow the instructions to recreate the plots

[37]:
```
# Use this data for the exercises
import numpy as np
x = np.arange(0,100)
y = x*2
z = x**2
```

Import matplotlib.pyplot as plt

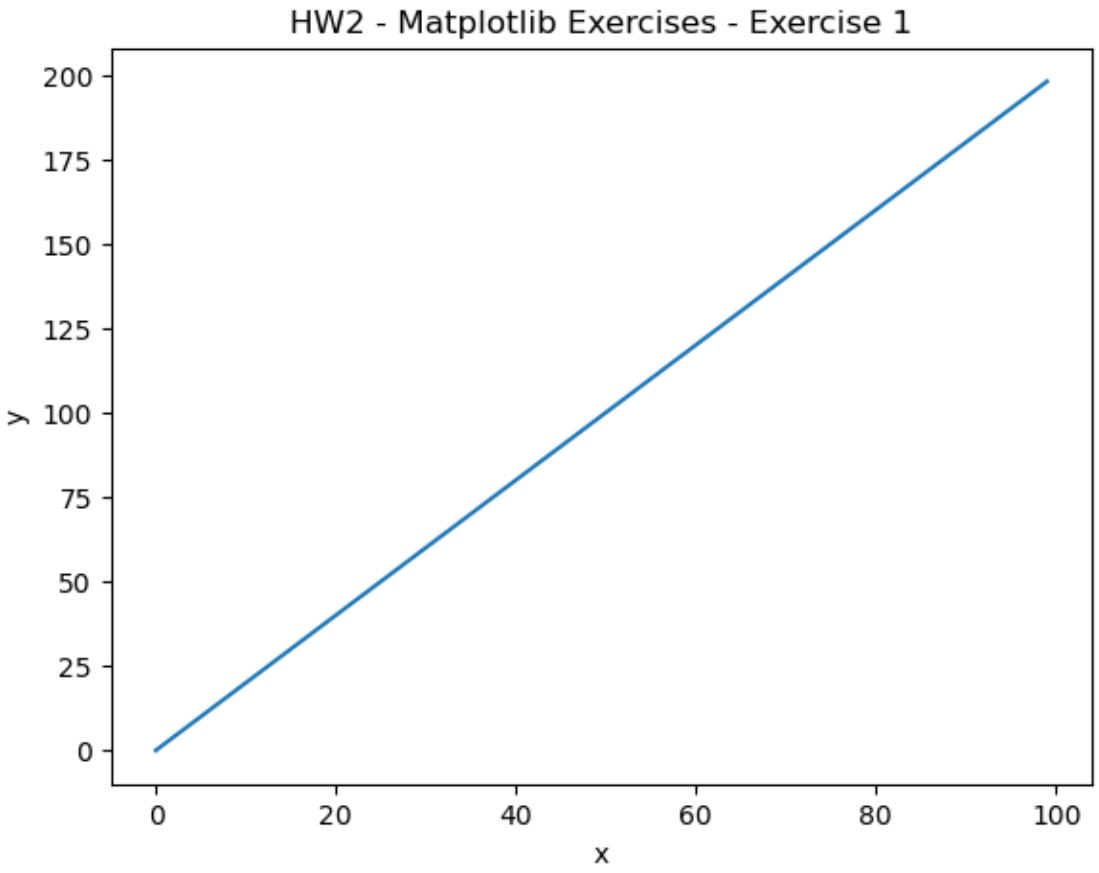
[38]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

[ ]:
```
# Question 1
```

- **Create a figure object called fig using plt.figure() -Use add_axes to add an axis to the figure canvas at [0,0,0.8,0.8]. Call this new axis ax. -Plot (x,y) on that axes and set the labels and titles to match the plot below:**

[39]:
```

```

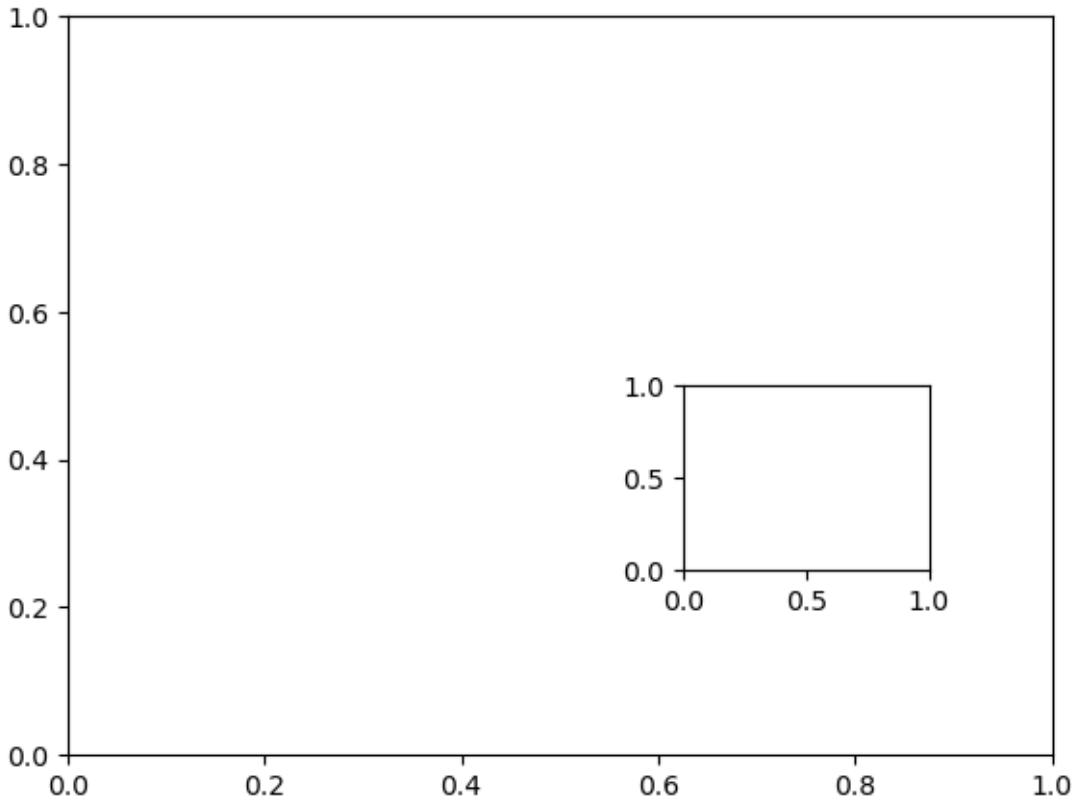
[39]:
```
Text(0.5, 1.0, 'HW2 - Matplotlib Exercises - Exercise 1')
```



[ ]:
```
# Question 2
```

Create a figure object and put two axes on it, ax1 and ax2. Located at [0,0,0.8,0.8] and [0.5,0.2,.2,.2] respectively.

[40]:
```

```

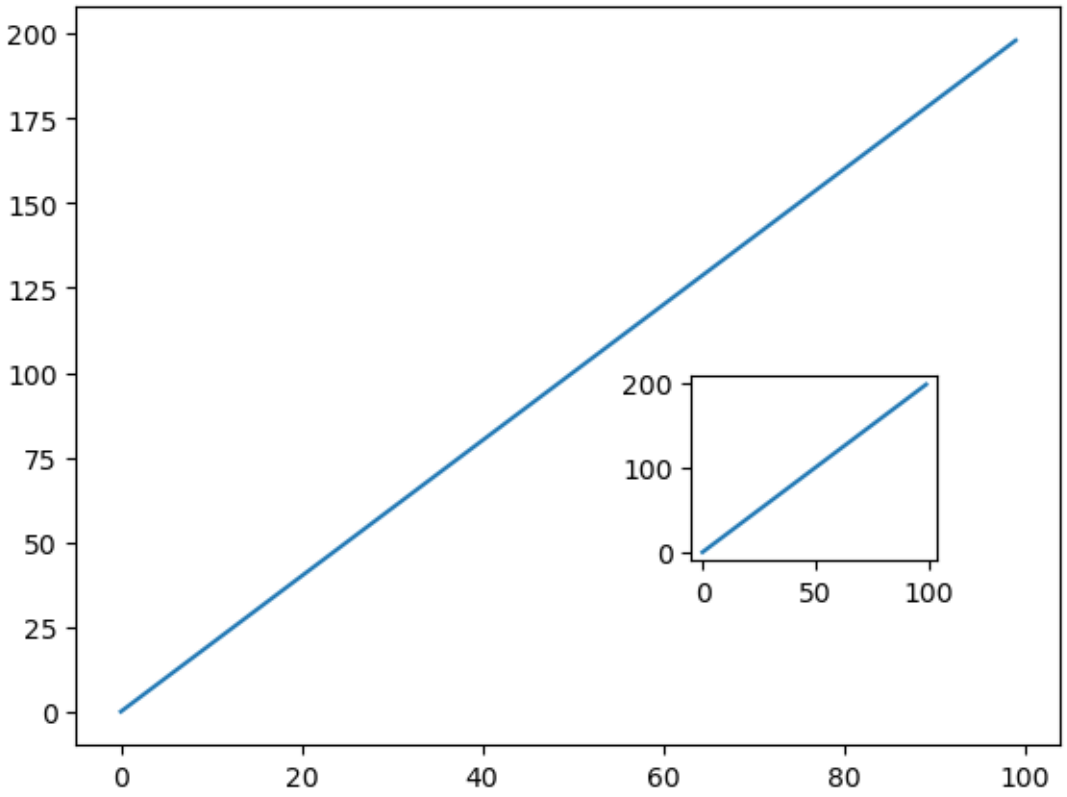


[ ]:
```
# Question 3
```

Now plot (x,y) on both axes. And call your figure object to show it.

[41]:
```

```

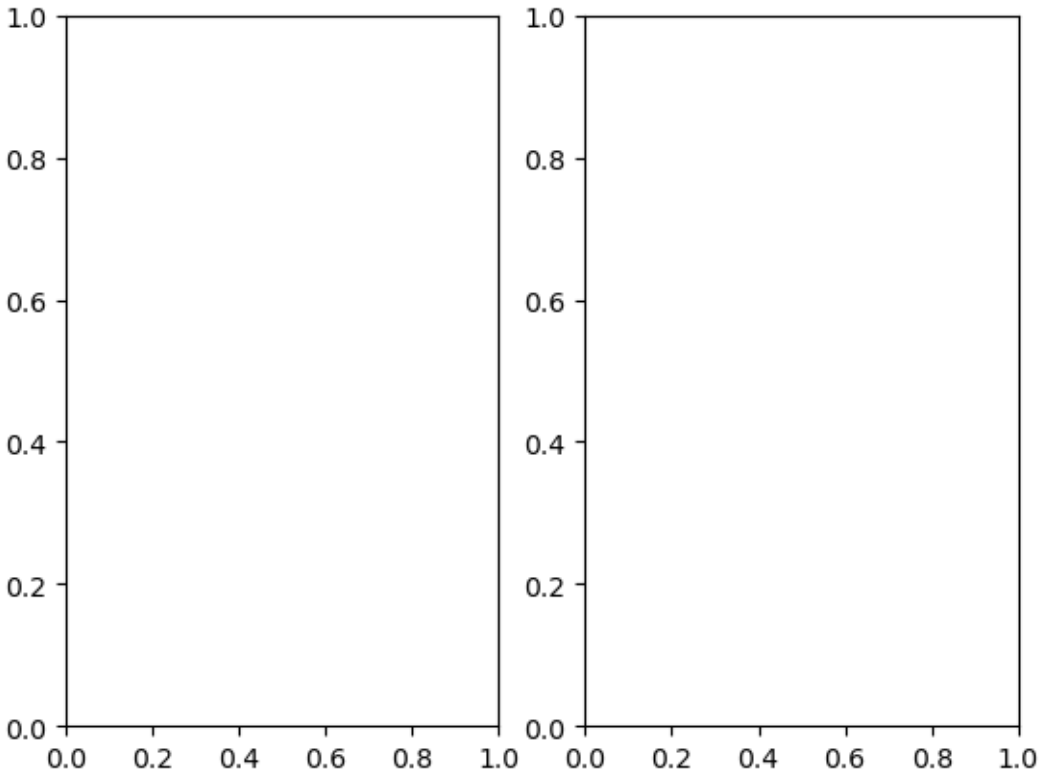
[41]:



[ ]:
```
# Question 4
```

Use plt.subplots(nrows=1, ncols=2) to create the plot below.

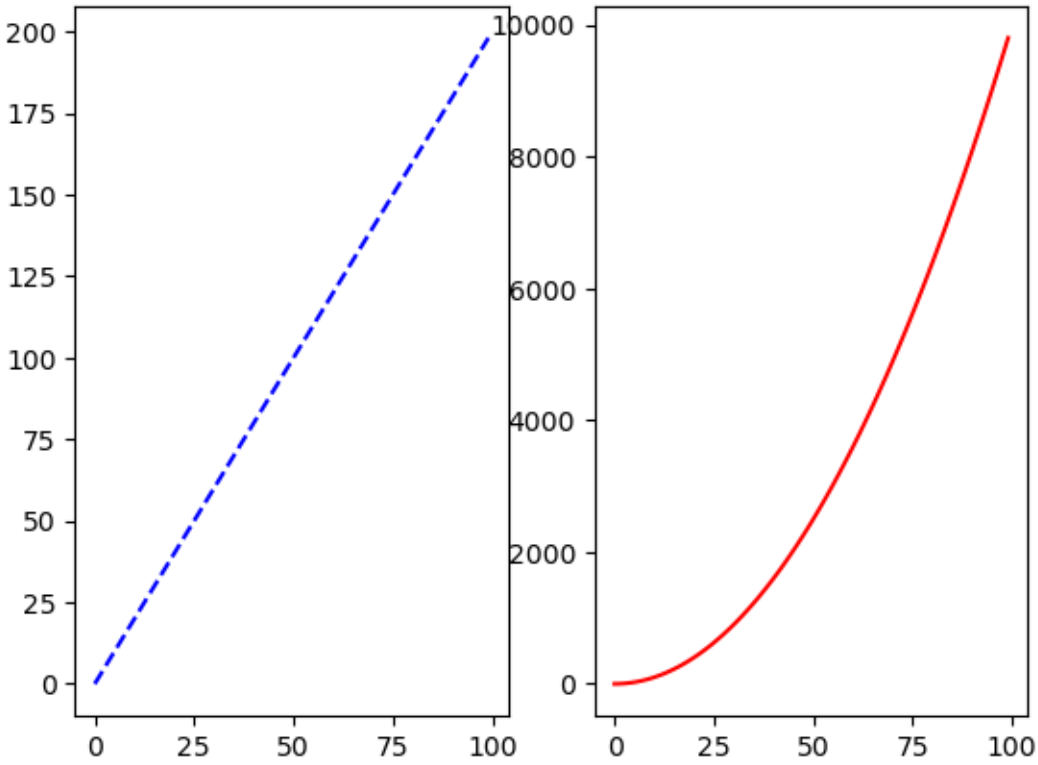
[44]:
```

```



Now plot (x,y) and (x,z) on the axes. Play around with the linewidth and style

[45]:

[45]:

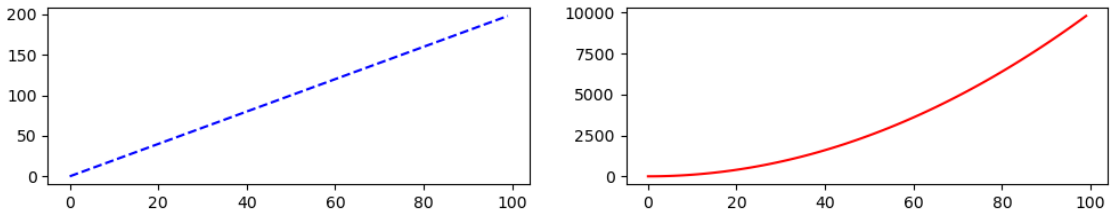


[ ]:
```
# Question 5
```

Resize the plot by adding the 'figsize = (12,2)' argument in plt.subplots()

[46]:
```

```

[46]:
```
[<matplotlib.lines.Line2D at 0x20708efa400>]
```



2 HW2 - Seaborn Exercises - Sol

Maximum Number of Points: 10 (2 points per question)

2.1 The Data

You will be working with a famous titanic data set for these exercises. In later lectures, we will revisit this data, and use it to predict survival rates of passengers. For now, just focus on the visualization of the data with seaborn:

[1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
sns.set_style('whitegrid')
```

[3]:
```
titanic = sns.load_dataset('titanic')
```

[4]:
```
titanic.head()
```

[4]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

3 Exercises

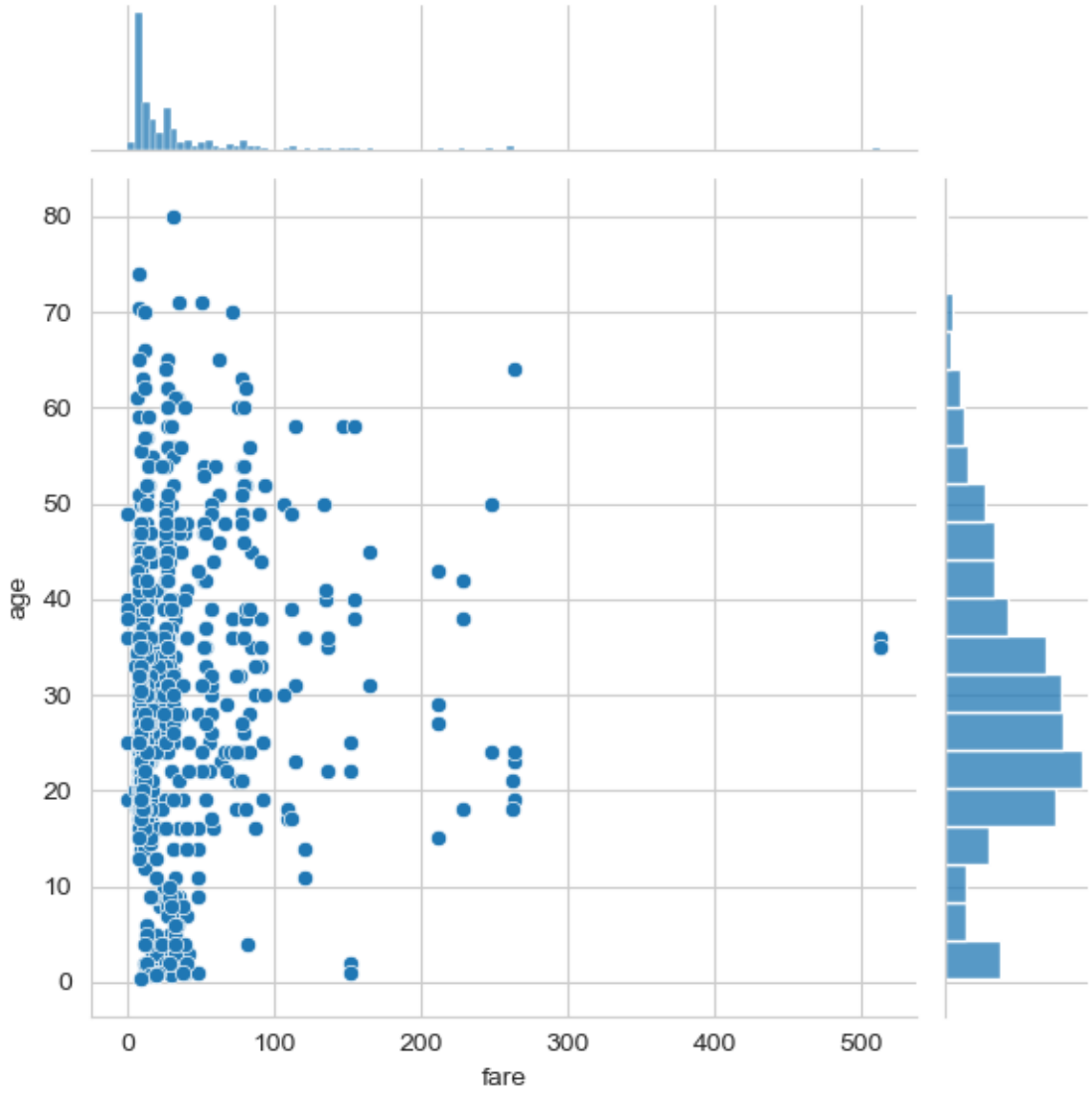
Recreate the plots below using the titanic dataframe. There are very few hints since most of the plots can be done with just one or two lines of code and a hint would basically give away the solution.

Note: In order to not lose the plot image, make sure you don't code in the cell that is directly above the plot, there is an extra cell above that one which won't overwrite that plot.

[5]:
```
# Question 1
# REPLICATE EXERCISE PLOT IMAGE BELOW
# BE CAREFUL NOT TO OVERWRITE CELL BELOW
# THAT WOULD REMOVE THE EXERCISE PLOT IMAGE
```

[6]:
```

```

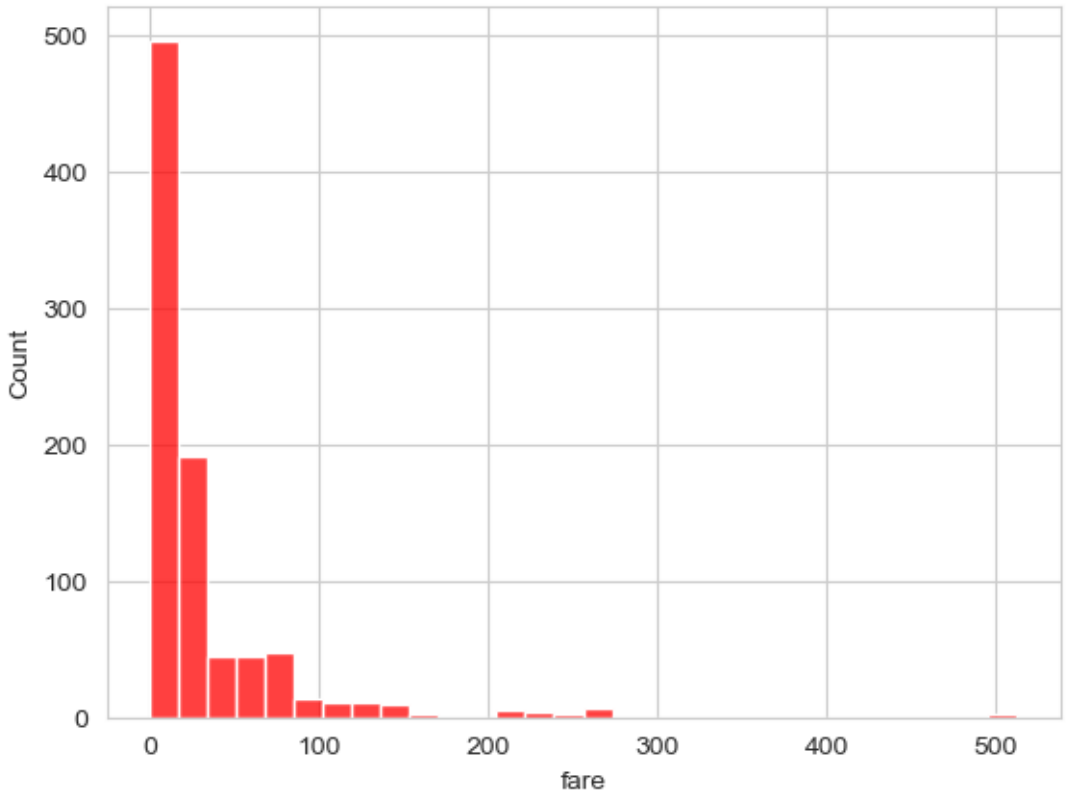
[6]:
```
<seaborn.axisgrid.JointGrid at 0x16242359e50>
```



[8]:
```
# Question 2
# REPLICATE EXERCISE PLOT IMAGE BELOW
# BE CAREFUL NOT TO OVERWRITE CELL BELOW
# THAT WOULD REMOVE THE EXERCISE PLOT IMAGE
```

[26]:
```

```

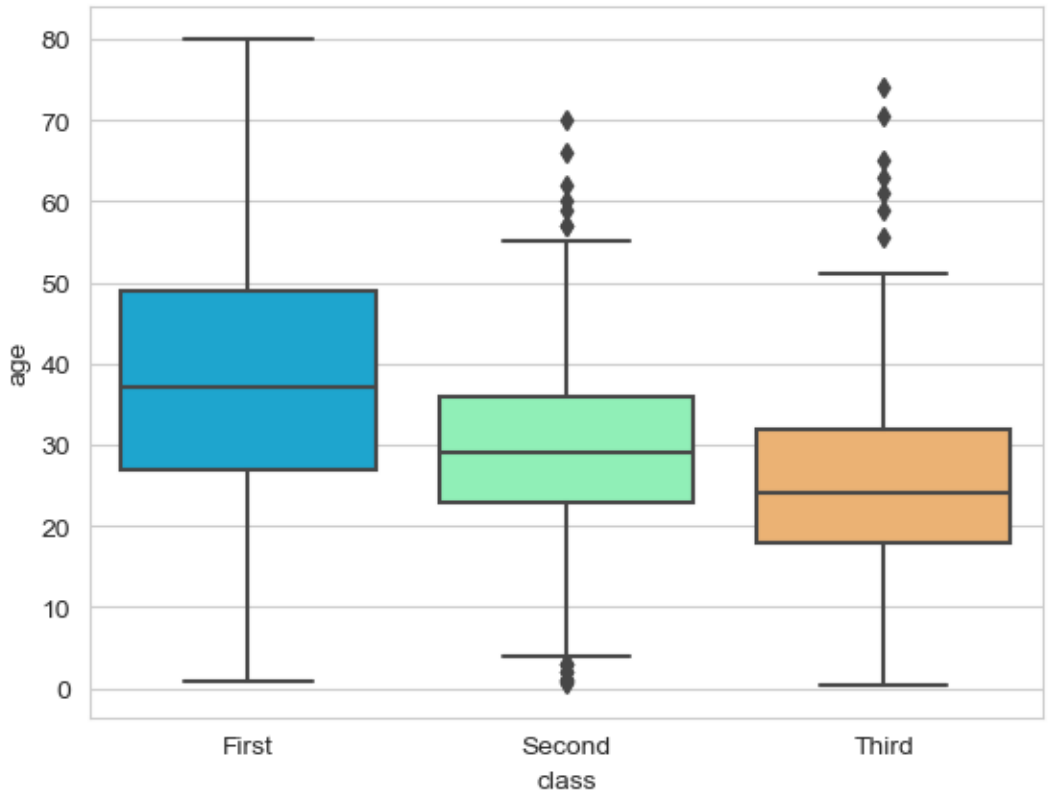
[26]: <AxesSubplot:xlabel='fare', ylabel='Count'>



[11]:
```
# Question 3
# REPLICATE EXERCISE PLOT IMAGE BELOW
# BE CAREFUL NOT TO OVERWRITE CELL BELOW
# THAT WOULD REMOVE THE EXERCISE PLOT IMAGE
```

[12]:
```

```

[12]: <AxesSubplot:xlabel='class', ylabel='age'>

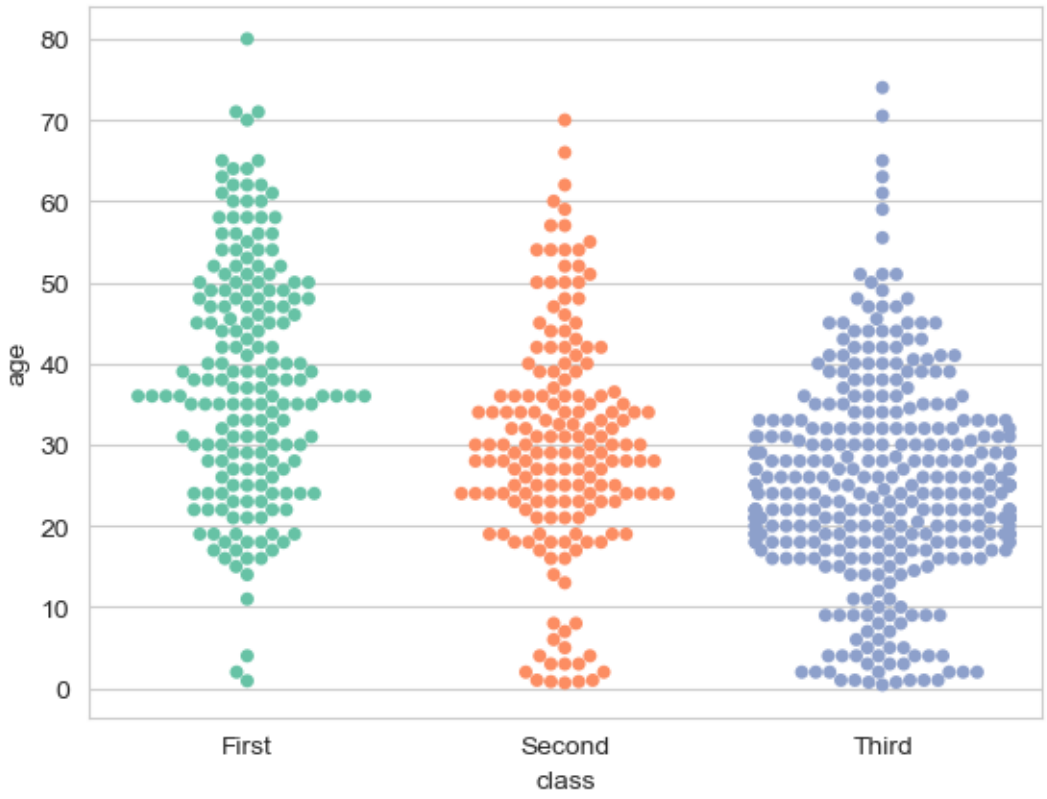


[14]:
```
# Question 4
# REPLICATE EXERCISE PLOT IMAGE BELOW
# BE CAREFUL NOT TO OVERWRITE CELL BELOW
# THAT WOULD REMOVE THE EXERCISE PLOT IMAGE
```

[15]:
```

```

```
C:\Users\tomma\anaconda3\lib\site-packages\seaborn\categorical.py:1296:
UserWarning: 11.0% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

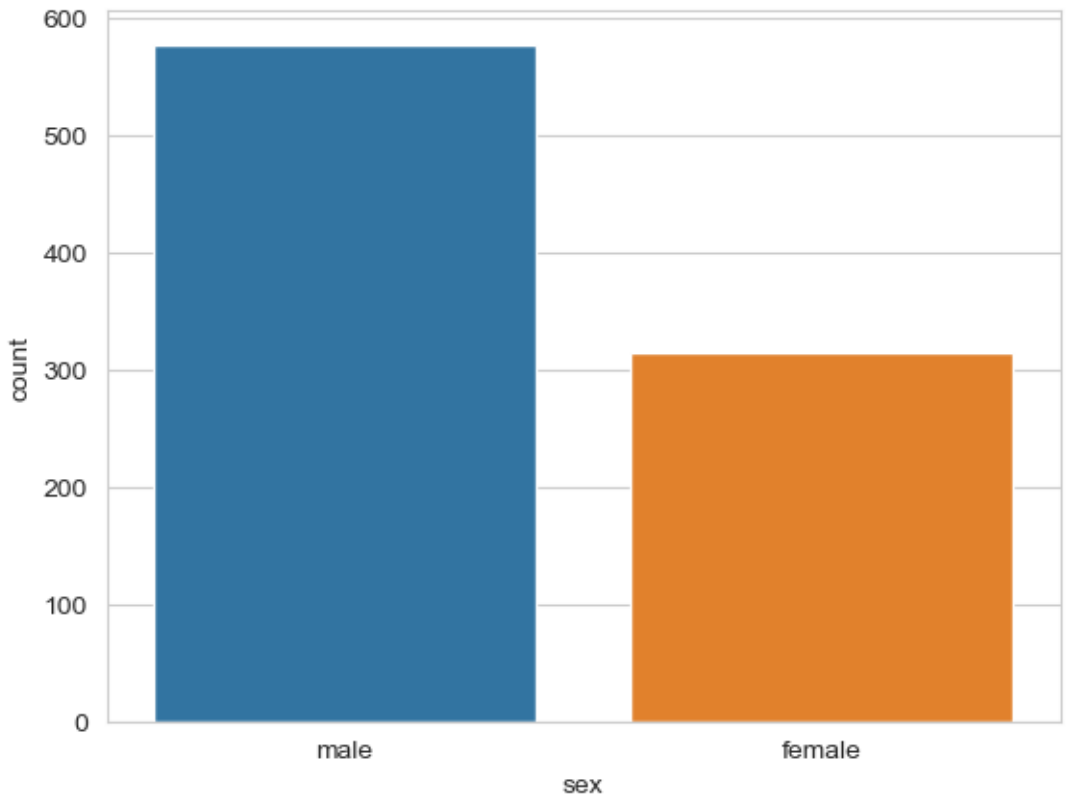
[15]: <AxesSubplot:xlabel='class', ylabel='age'>



[17]:
```
# Question 5
# REPLICATE EXERCISE PLOT IMAGE BELOW
# BE CAREFUL NOT TO OVERWRITE CELL BELOW
# THAT WOULD REMOVE THE EXERCISE PLOT IMAGE
```

[18]:
```

```

[18]: <AxesSubplot:xlabel='sex', ylabel='count'>



4 HW2 - Linear Regression Exercise - Sol

Maximum Number of points: 50

Questions 1-21 (2 points per question) - Question 22 (8 points)

An Ecommerce company sells clothing online but they also have in-store style and clothing advice sessions. Customers come in to the store, have sessions/meetings with a personal stylist, then they can go home and order either on a mobile app or website for the clothes they want.
The company is trying to decide whether to focus their efforts on their mobile app experience or their website.

Complete the steps below to analyze the customer data (it's fake … not real credit card numbers or emails).

[51]:
```
%matplotlib inline
```

1. Import pandas, numpy, matplotlib,and seaborn.

[52]:
```
```

4.0.1 Get the Data

The Ecommerce Customers csv file from the company has Customer info, such as Email, Address, and their color Avatar.
It also has numerical value columns:

- Avg. Session Length: Average session of in-store style advice sessions.
- Time on App: Average time spent on App in minutes
- Time on Website: Average time spent on Website in minutes
- Length of Membership: How many years the customer has been a member.
- Yearly Amount Spent: The average annual customer spending.

2. Read in the Ecommerce Customers.csv file as a DataFrame called customers.

[53]:

3. Check the head of customers, and examine it using info and describe methods.

[54]:

[54]:

```
                          Email  \
0     mstephenson@fernandez.com
1             hduke@hotmail.com
2              pallen@yahoo.com
3       riverarebecca@gmail.com
4  mstephens@davidson-herman.com

                                          Address            Avatar  \
0      835 Frank Tunnel\nWrightmouth, MI 82180-9605            Violet
1    4547 Archer Common\nDiazchester, CA 06566-8576         DarkGreen
2  24645 Valerie Unions Suite 582\nCobbborough, D…            Bisque
3   1414 David Throughway\nPort Jason, OH 22070-1220       SaddleBrown
4  14023 Rodriguez Passage\nPort Jacobville, PR 3…  MediumAquaMarine

   Avg. Session Length  Time on App  Time on Website  Length of Membership  \
0            34.497268    12.655651        39.577668              4.082621
1            31.926272    11.109461        37.268959              2.664034
2            33.000915    11.330278        37.110597              4.104543
3            34.305557    13.717514        36.721283              3.120179
4            33.330673    12.795189        37.536653              4.446308

   Yearly Amount Spent
0           587.951054
1           392.204933
2           487.547505
3           581.852344
4           599.406092
```

[55]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Email                 500 non-null    object 
 1   Address               500 non-null    object 
 2   Avatar                500 non-null    object 
 3   Avg. Session Length   500 non-null    float64
 4   Time on App           500 non-null    float64
 5   Time on Website       500 non-null    float64
 6   Length of Membership  500 non-null    float64
 7   Yearly Amount Spent   500 non-null    float64
dtypes: float64(5), object(3)
memory usage: 31.4+ KB
```

[56]:

[56]:
```
       Avg. Session Length  Time on App  Time on Website  \
count           500.000000   500.000000       500.000000   
mean             33.053194    12.052488        37.060445   
std               0.992563     0.994216         1.010489   
min              29.532429     8.508152        33.913847   
25%              32.341822    11.388153        36.349257   
50%              33.082008    11.983231        37.069367   
75%              33.711985    12.753850        37.716432   
max              36.139662    15.126994        40.005182   

       Length of Membership  Yearly Amount Spent  
count            500.000000           500.000000  
mean               3.533462           499.314038  
std                0.999278            79.314782  
min                0.269901           256.670582  
25%                2.930450           445.038277  
50%                3.533975           498.887875  
75%                4.126502           549.313828  
max                6.922689           765.518462  
```

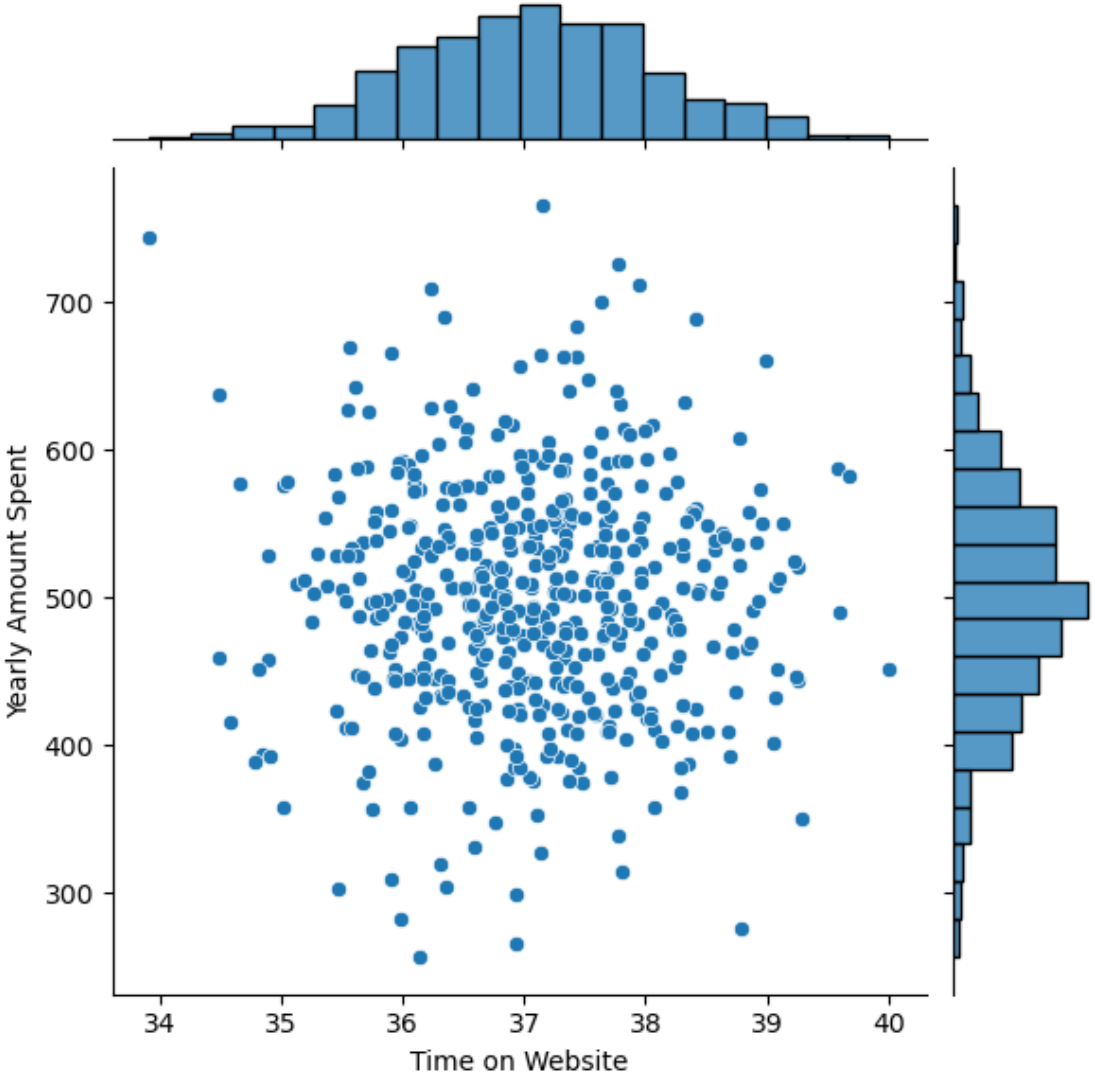
4.1 Exploratory Data Analysis

For the rest of the exercise only the numerical data of the csv file will be used.

4. Use seaborn to create a jointplot to compare the 'Time on Website' and 'Yearly Amount Spent' columns.

[57]:

[57]:
```
<seaborn.axisgrid.JointGrid at 0x21177142340>
```



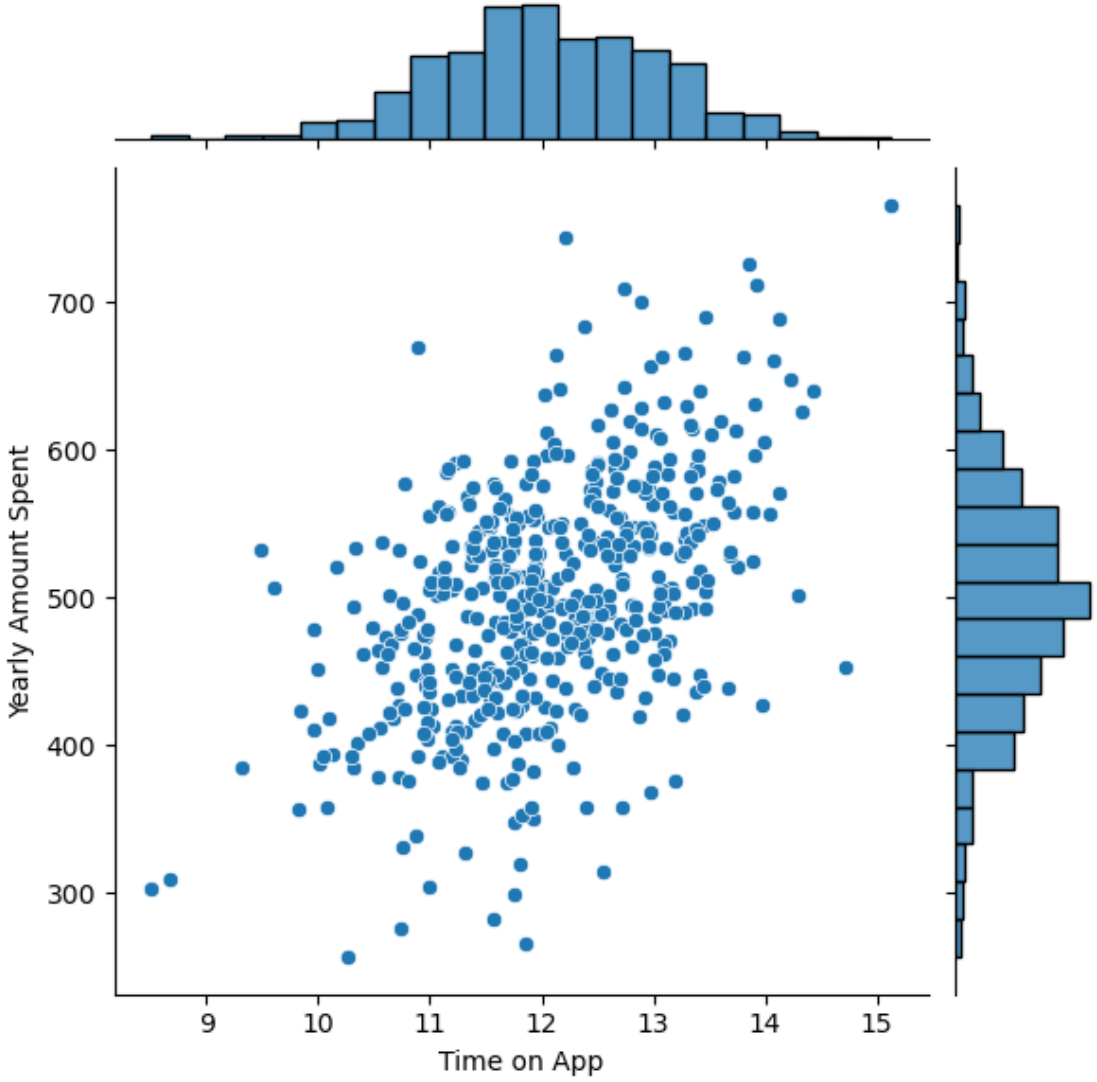
5. Do 'Time on Website' and 'Yearly Amount Spent' look positively correlated, negatively correlated, or not correlated?

[58]:
```
# Write here your answer to Question 5.
```

6. Do the same with the 'Time on App' column.

[59]:
```

```

[59]: <seaborn.axisgrid.JointGrid at 0x21178c07400>



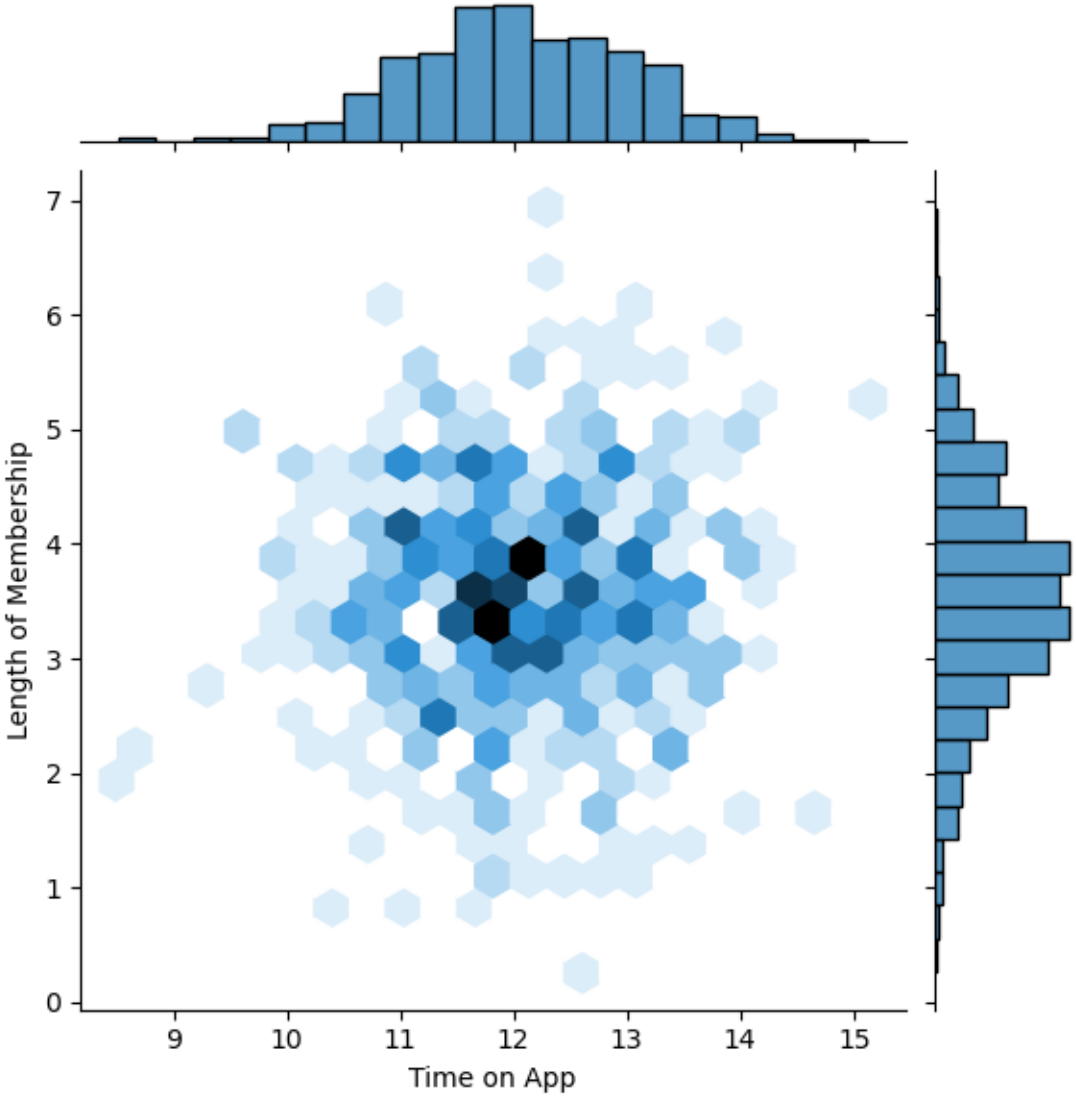
7. Do 'Time on Website' and 'Yearly Amount Spent' look positively correlated, negatively correlated, or not correlated?

[60]:
```
# Write here your answer to Question 7.
```

8. Use jointplot to create a 2D hex bin plot comparing 'Time on App' and 'Length of Membership'.

[61]:
```

```

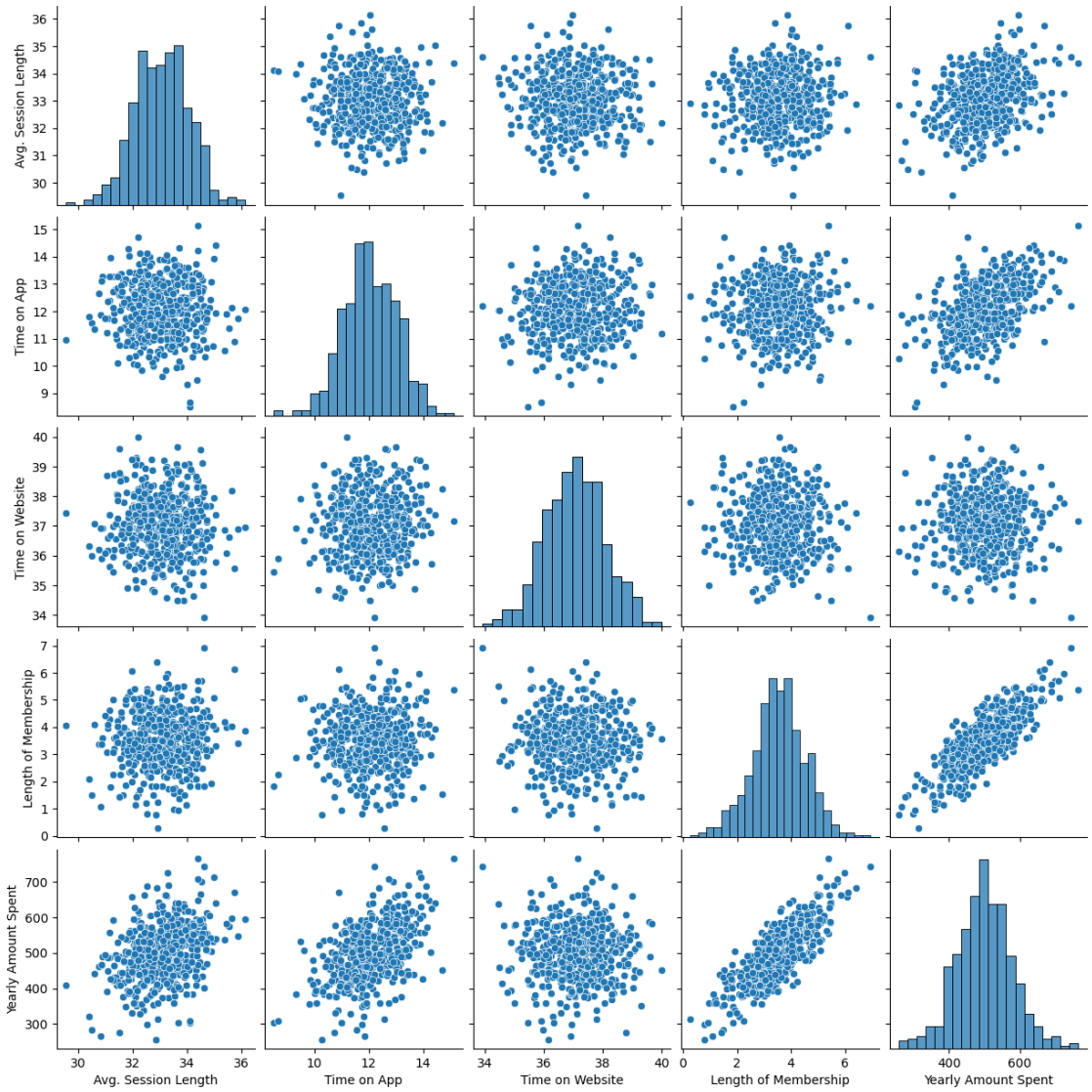
[61]: <seaborn.axisgrid.JointGrid at 0x21178fff5b0>



9. Explore the relationships across the entire data set. Use pairplot to recreate the plot below
(Depending on the speed of your computer, this may take a short while because of all the required calculations)

[62]:

[62]: `<seaborn.axisgrid.PairGrid at 0x21179435730>`



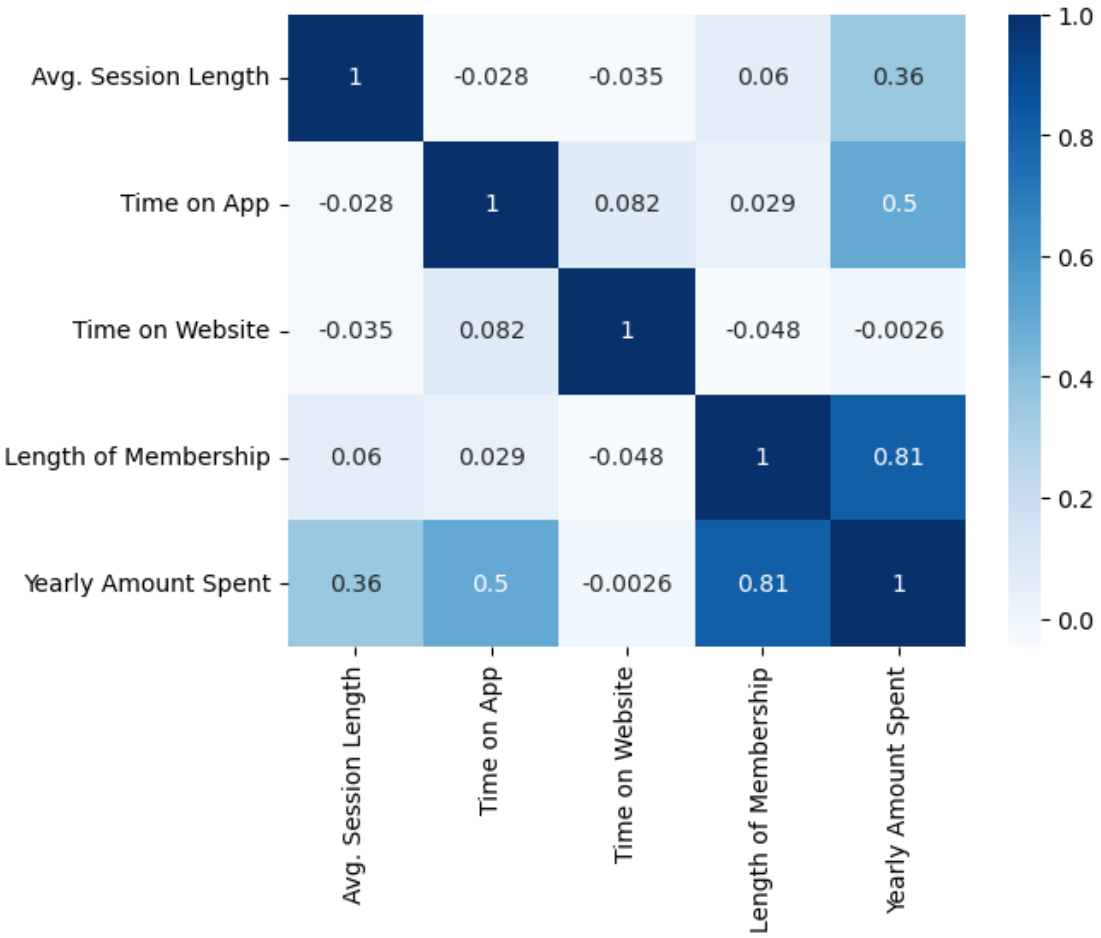
10. Based off this plot what looks to be the most correlated feature with Yearly Amount Spent?

[63]:
```
# Write here your answer to Question 10.
```

11. Use heatmap to recreate the plot below

[64]:
```

```

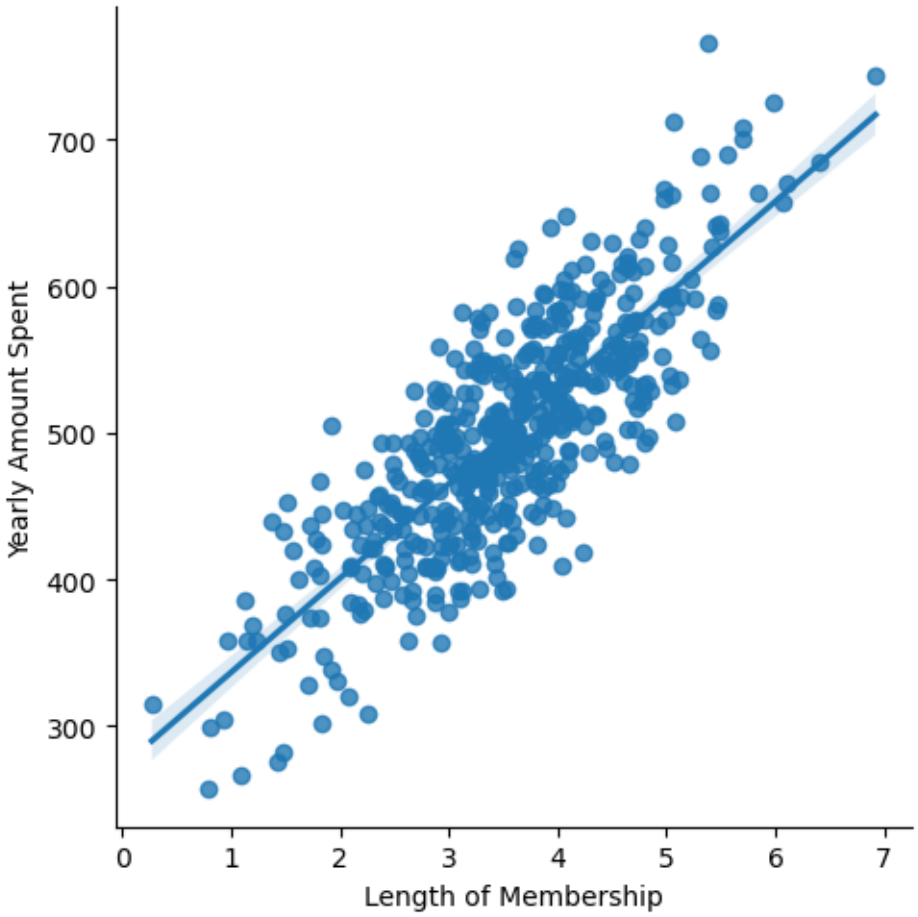
[64]: <AxesSubplot:>



12. Create a linear model plot (using seaborn's lmplot) of 'Yearly Amount Spent' vs. 'Length of Membership'.

[65]:

[65]: <seaborn.axisgrid.FacetGrid at 0x2117af239a0>



4.2 Training and Testing Data

Set a variable X equal to the numerical features of the customers

[66]:
```
X = customers[['Avg. Session Length', 'Time on App', 'Time on Website', 'Length␣
↪of Membership']]
X.head()
```

[66]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership
0	34.497268	12.655651	39.577668	4.082621
1	31.926272	11.109461	37.268959	2.664034
2	33.000915	11.330278	37.110597	4.104543
3	34.305557	13.717514	36.721283	3.120179
4	33.330673	12.795189	37.536653	4.446308

13. Set a variable y equal to the "Yearly Amount Spent" column.

[67]:
```

```

[67]:
```
0    587.951054
1    392.204933
2    487.547505
3    581.852344
4    599.406092
Name: Yearly Amount Spent, dtype: float64
```

14. Use model_selection.train_test_split from sklearn to split the data into training and testing sets. Set test_size=0.3 and random_state=101

[68]:
```
from sklearn.model_selection import train_test_split
```

[69]:
```

```

4.3 Train the Model on the training data

Import LinearRegression from sklearn.linear_model

[70]:
```
from sklearn.linear_model import LinearRegression
```

15. Create an instance of a LinearRegression model named lm.

[71]:
```

```

16. Train/fit lm on the training data.

[72]:
```

```

[72]:
```
LinearRegression()
```

17. Print out the coefficients of the model

[73]:
```

```

[73]:
```
array([25.98154972, 38.59015875,  0.19040528, 61.27909654])
```

4.4 Evaluate model performance by predicting off the test values

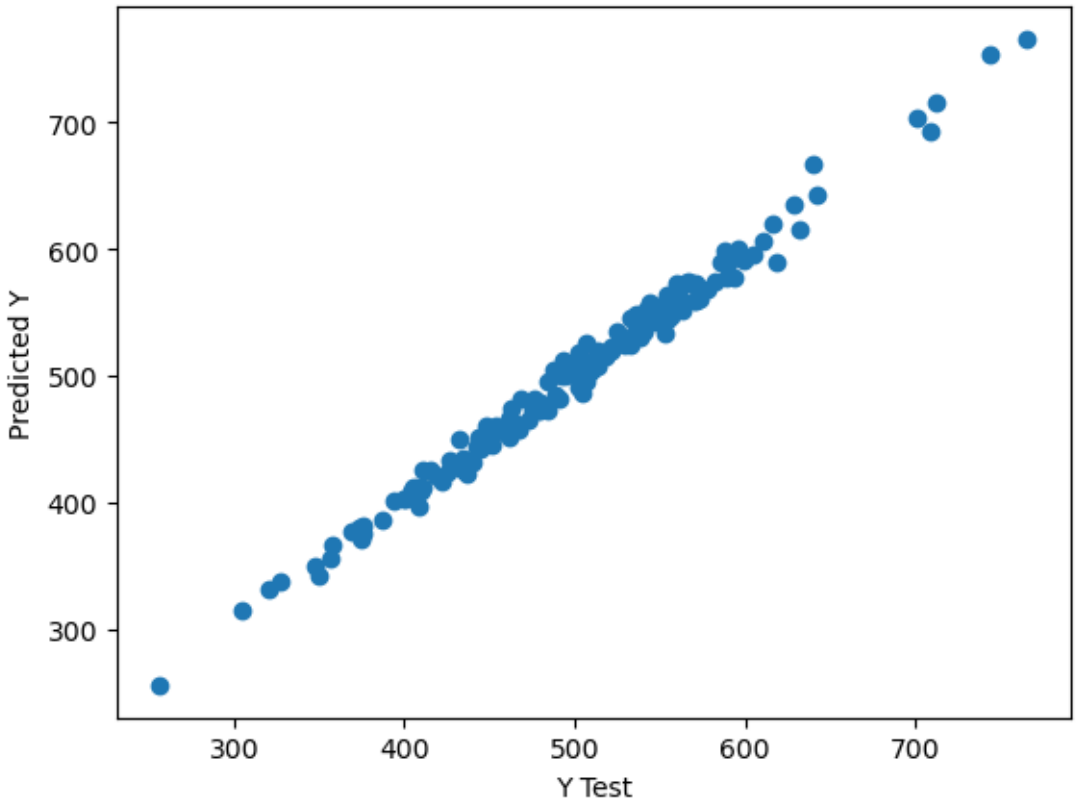
18. Use lm.predict() to predict off the X_test set of the data.

[74]:
```

```

19. Create a scatterplot of the real test values versus the predicted values.

[75]:
```

```

[75]:
```
Text(0, 0.5, 'Predicted Y')
```



Import the metrics module form sklearn

[76]:
```
from sklearn import metrics
```

19. Calculate the Mean Absolute Error, Mean Squared Error, and the Root Mean Squared Error.

[77]:
```

```

```
MAE: 7.228148653430832
MSE: 79.81305165097444
RMSE: 8.933815066978633
```

20. Calculate R-squared (percentage of explained variance)

[78]:
```

```

```
R-squared is: 0.9890046246741234
```

21. Do you think the company should focus more on their mobile app or on their website? Write your thoughts

[ ]:
```
# Write here your answer to Question 21.
```

22. Open Question: Imagine client asked: are there other insights from this data. Do addtional analysis, pick two insights and prove your claims. (8 points)

[ ]:
```
# Write here your answer to Question 22.
```

5 Logistic Regression Exercise

Maximum Number of points: 30

Questions 1-15 (2 points per question)

You will be working with a fake advertising data set, indicating whether or not a particular internet user clicked on an advertisement on a company website.
The task is to create a model to predict whether or not a user will click on an ad based off the features of that user.

This data set contains the following features:

- 'Daily Time Spent on Site': consumer time on site in minutes
- 'Age': cutomer age in years
- 'Area Income': Avg. Income of geographical area of consumer
- 'Daily Internet Usage': Avg. minutes a day consumer is on the internet
- 'Ad Topic Line': Headline of the advertisement
- 'City': City of consumer
- 'Male': Whether or not consumer was male
- 'Country': Country of consumer
- 'Timestamp': Time at which consumer clicked on Ad or closed window
- 'Clicked on Ad': 0 or 1 indicated clicking on Ad

5.1 Import Libraries

Import the libraries you will need

[120]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set_style('whitegrid')
import warnings
warnings.filterwarnings('ignore')
```

5.2 Get the Data

1. Read in the advertising.csv file and set it to a data frame called ad_data.

[121]:
```

```

2. Check the head of ad_data

[122]:
```

```

[122]:
```
   Daily Time Spent on Site  Age  Area Income  Daily Internet Usage  \
0                     68.95   35     61833.90                256.09
1                     80.23   31     68441.85                193.77
2                     69.47   26     59785.94                236.50
3                     74.15   29     54806.18                245.89
4                     68.37   35     73889.99                225.58

                          Ad Topic Line            City  Male     Country  \
0    Cloned 5thgeneration orchestration     Wrightburgh     0     Tunisia
1    Monitored national standardization       West Jodi     1       Nauru
2      Organic bottom-line service-desk        Davidton     0  San Marino
3  Triple-buffered reciprocal time-frame  West Terrifurt     1       Italy
4          Robust logistical utilization    South Manuel     0     Iceland

             Timestamp  Clicked on Ad
0  2016-03-27 00:53:11              0
1  2016-04-04 01:39:02              0
2  2016-03-13 20:35:42              0
3  2016-01-10 02:31:19              0
4  2016-06-03 03:36:18              0
```

3. Use info and describe on ad_data

[123]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Daily Time Spent on Site  1000 non-null   float64
 1   Age                       1000 non-null   int64
 2   Area Income               1000 non-null   float64
 3   Daily Internet Usage      1000 non-null   float64
 4   Ad Topic Line             1000 non-null   object
 5   City                      1000 non-null   object
 6   Male                      1000 non-null   int64
 7   Country                   1000 non-null   object
 8   Timestamp                 1000 non-null   object
 9   Clicked on Ad             1000 non-null   int64
dtypes: float64(3), int64(3), object(4)
memory usage: 78.2+ KB
```

[124]:

[124]:
```
       Daily Time Spent on Site          Age   Area Income  \
count               1000.000000  1000.000000   1000.000000
mean                  65.000200    36.009000  55000.000080
```

```
std                   15.853615     8.785562  13414.634022
min                   32.600000    19.000000  13996.500000
25%                   51.360000    29.000000  47031.802500
50%                   68.215000    35.000000  57012.300000
75%                   78.547500    42.000000  65470.635000
max                   91.430000    61.000000  79484.800000

       Daily Internet Usage         Male  Clicked on Ad
count           1000.000000  1000.000000     1000.00000
mean             180.000100     0.481000        0.50000
std               43.902339     0.499889        0.50025
min              104.780000     0.000000        0.00000
25%              138.830000     0.000000        0.00000
50%              183.130000     0.000000        0.50000
75%              218.792500     1.000000        1.00000
max              269.960000     1.000000        1.00000
```

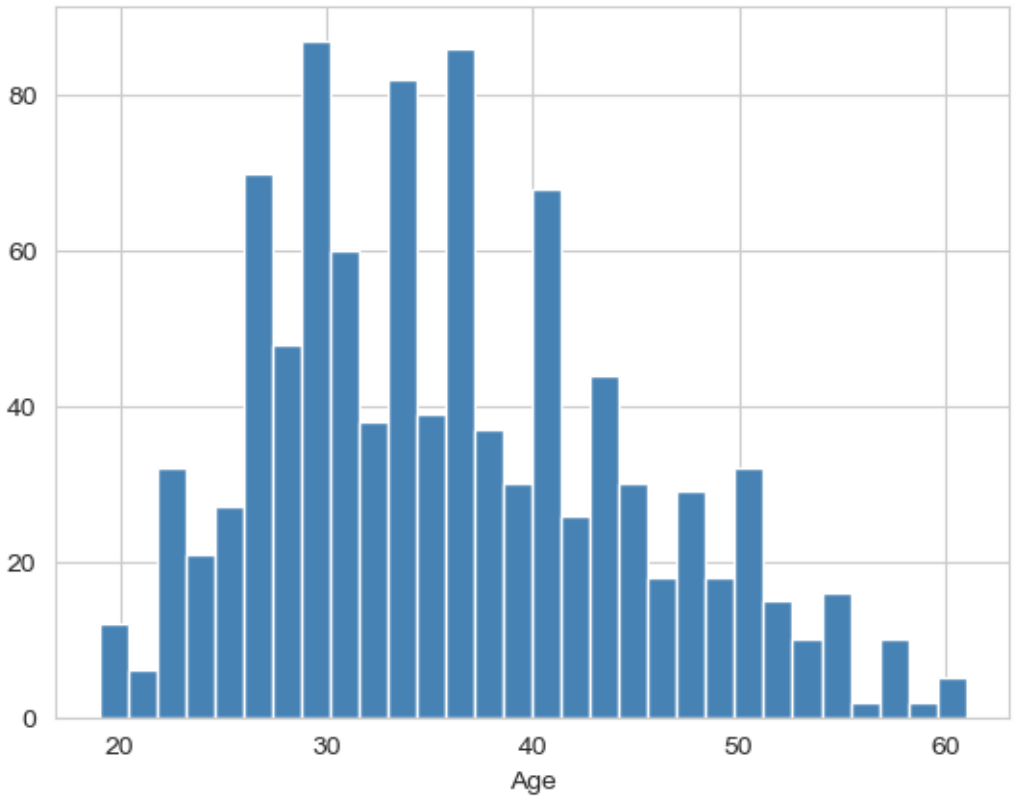
5.3 Exploratory Data Analysis

Use seaborn to explore the data.

4. Create a histogram of the Age

[125]:

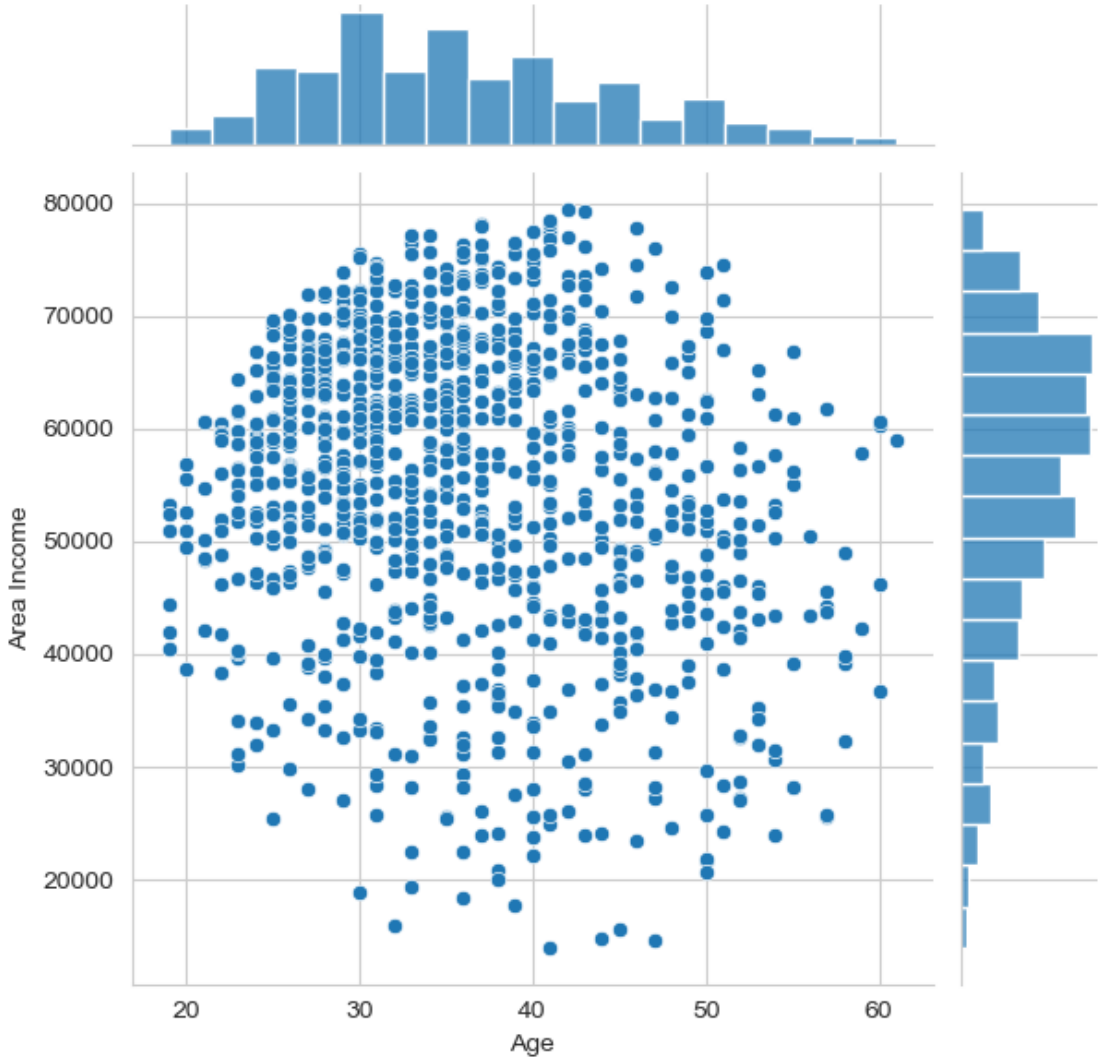
[125]: <AxesSubplot:xlabel='Age'>



5. Create a jointplot showing Area Income versus Age.

[126]:

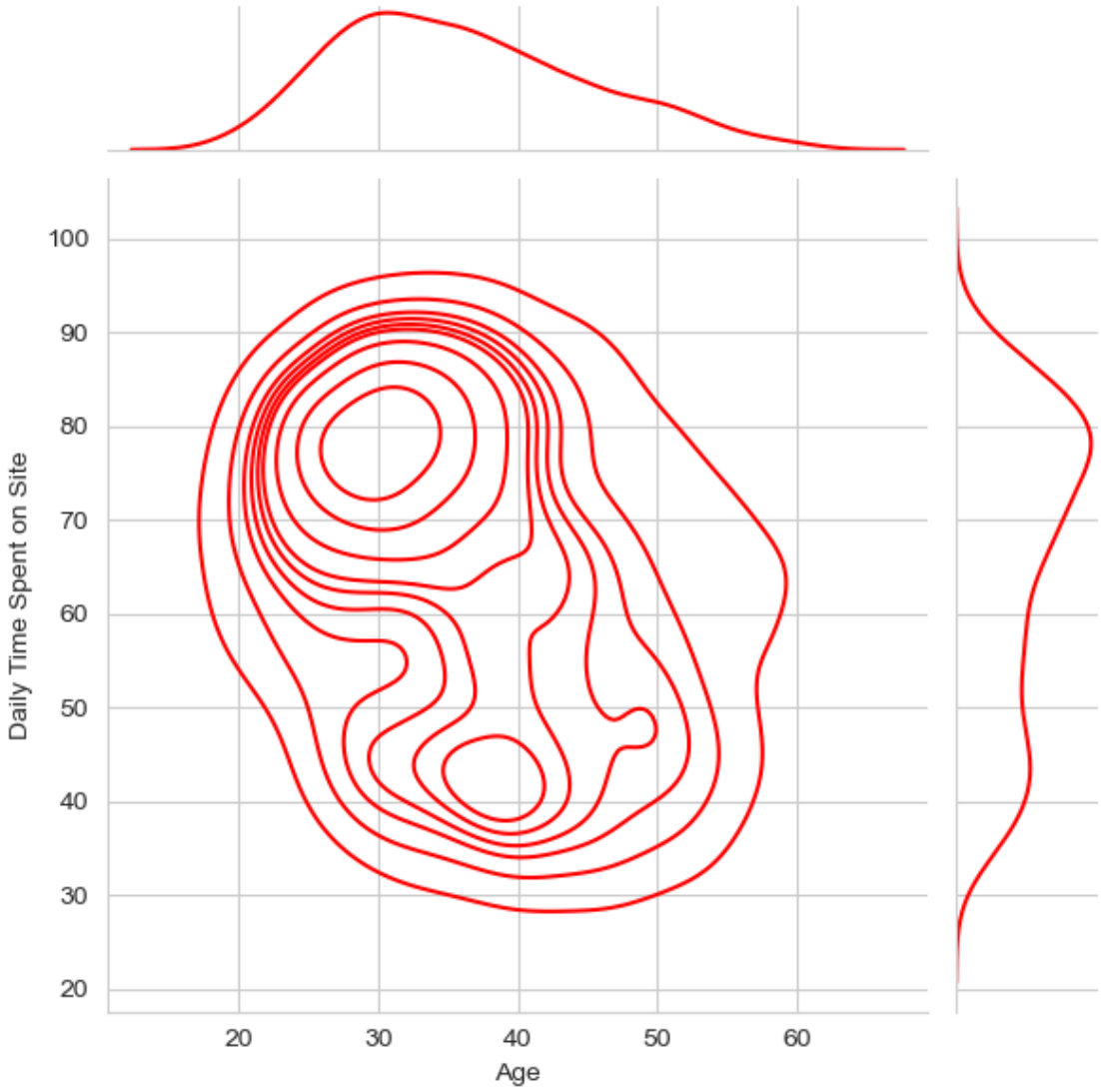
[126]: <seaborn.axisgrid.JointGrid at 0x1de2e61e430>



6. Create a jointplot showing the kde distributions of Daily Time spent on site vs. Age.

[127]:

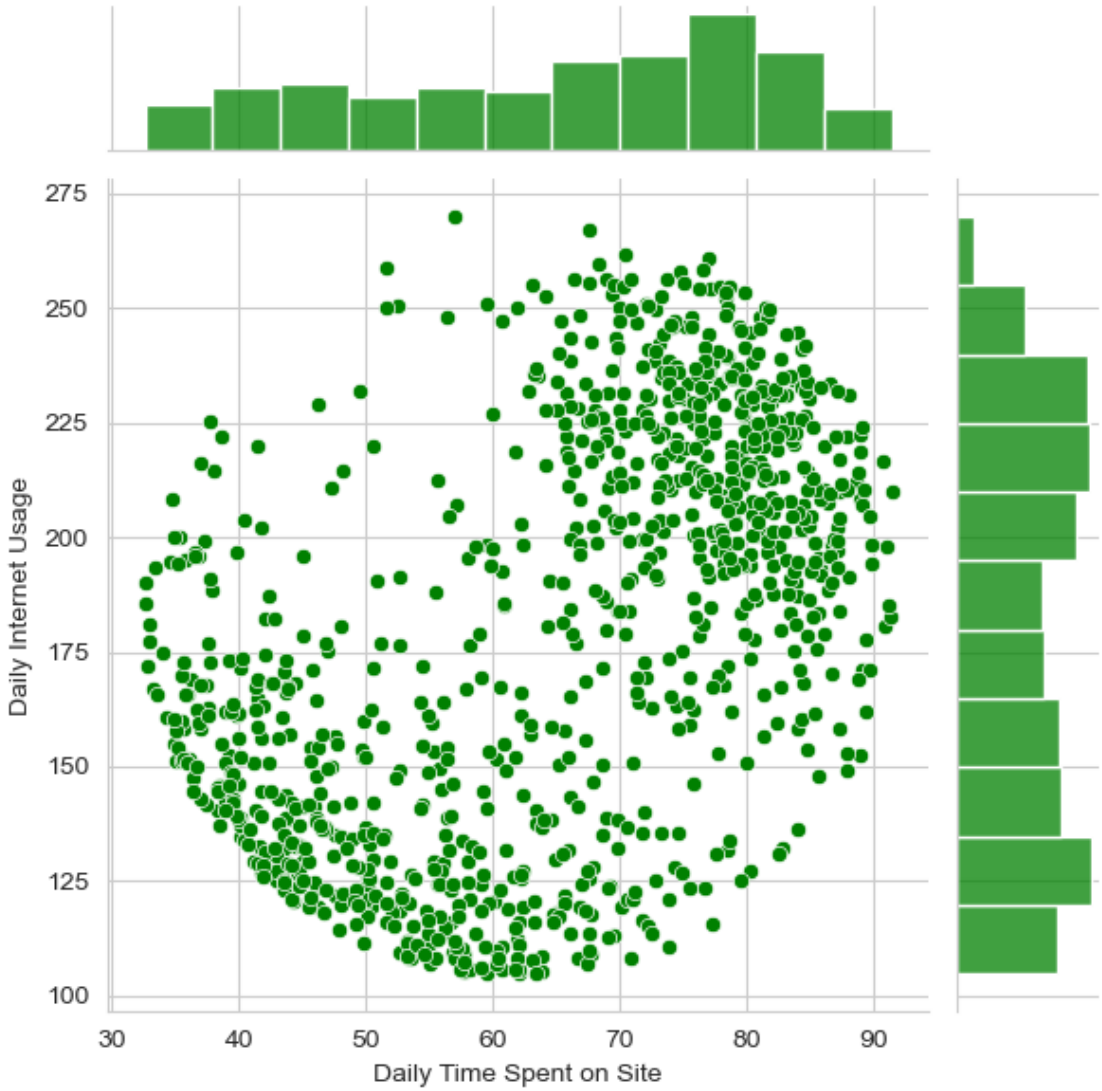
[127]: <seaborn.axisgrid.JointGrid at 0x1de31473f40>



7. Create a jointplot showing scatter plot of 'Daily Time Spent on Site' vs. 'Daily Internet Usage'

[128]:

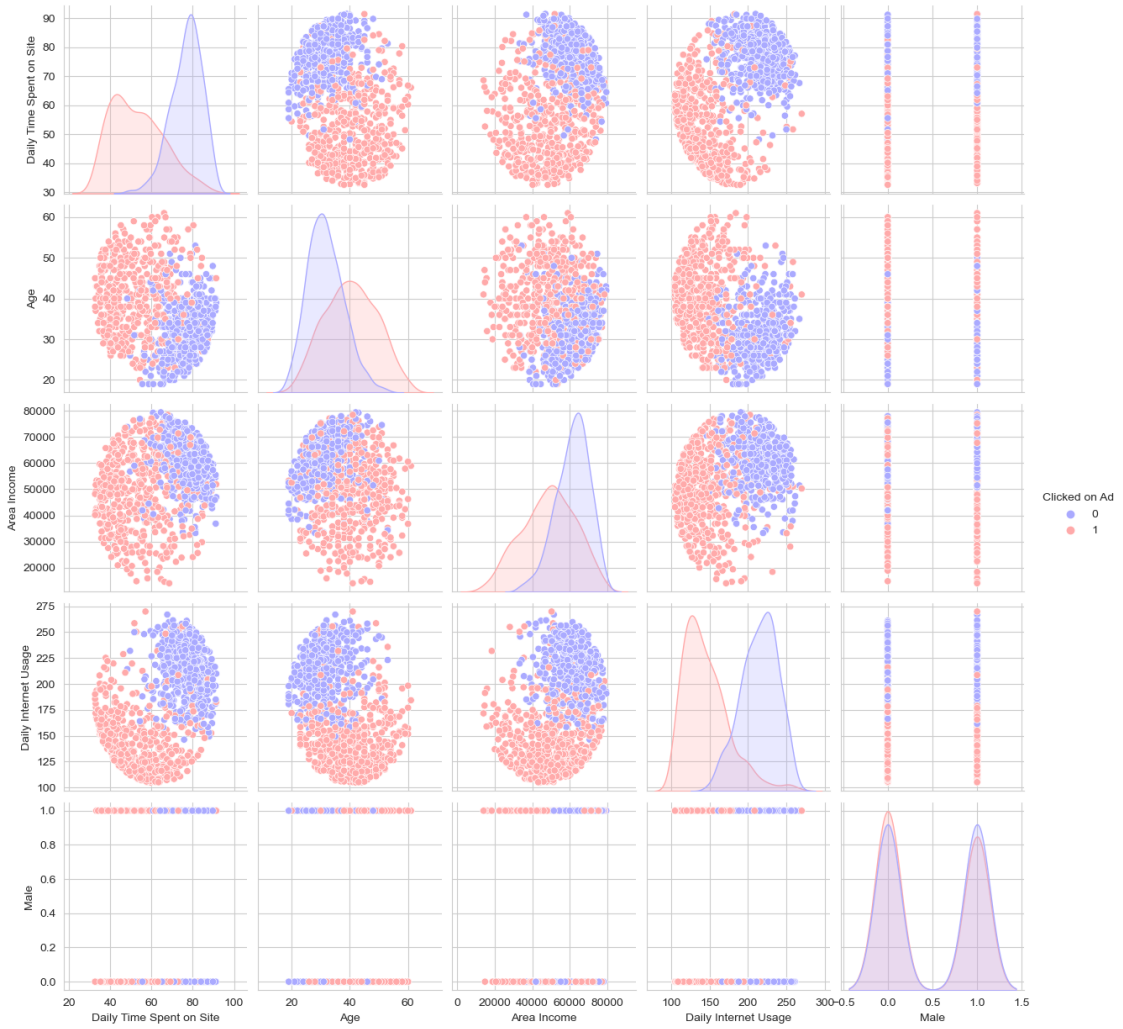
[128]: <seaborn.axisgrid.JointGrid at 0x1de318b2be0>



8. Create a pairplot with the hue defined by the 'Clicked on Ad' column feature.

[129]:

[129]: <seaborn.axisgrid.PairGrid at 0x1de323ffaf0>



6 Logistic Regression

Create, train, and test a model using different features. We want to experiment to find the features that lead to the best model.

9. Split the data into training set (with 'Daily Time Spent on Site' and 'Daily Internet Usage' as the only features) and testing set using 'train_test_split' with 'random_state' set to 101.

[130]:
```
from sklearn.model_selection import train_test_split
```

[131]:
```

```

[132]:
```

```

[133]:

10. Train and fit a logistic regression model on the training set.

[134]:
```
from sklearn.linear_model import LogisticRegression
```

[135]:

[136]:

[136]:
```
LogisticRegression()
```

6.1 Predictions and Evaluations

11. Predict values for the testing data.

[137]:

12. Create a confusion_matrix for the model.

[138]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

[139]:

```
[[153   4]
 [ 10 133]]
```

13. Create a classification report for the model.

[140]:

	precision	recall	f1-score	support
0	0.94	0.97	0.96	157
1	0.97	0.93	0.95	143
accuracy			0.95	300
macro avg	0.95	0.95	0.95	300
weighted avg	0.95	0.95	0.95	300

14. Retrain your model on a new training set with 'Age' and 'Daily Internet Usage' as the only features and re-do the confusion matrix.

[141]:

[142]:

[143]:

[144]:

[144]:
```
LogisticRegression()
```

[145]:
```

```

[146]:
```

```

```
[[146  11]
 [ 17 126]]
```

15. Are the results in Question 14 better or worse than before? Why?

[147]:
```
# Write your answer here.
```