Background Congratulations! You just got some contract work with an Ecommerce company based in New York City that 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. They've hired you on contract to help them figure it out! Let's get started!

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib as plt
import seaborn as sns
%matplotlib inline
```

Get the data

We'll work with the customer's csv file from the company. It has details of the customers such as email, address etc The description of the numerical column are:

- Avg session length: average session of the in store style advice
- Time on app: Average time spent on the App in minutes.
- Time on website: Average time spent on the store website in minutes.
- Length of membership: How many years the customer has been a member

Read in the csv file

In [2]:
```
customers = pd.read_csv("Ecommerce Customers.csv")
```

In [3]:
```
customers
```

Out[3]:

	Email	Address	Avatar	Avg. Session Length	Time on App	
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651	39
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461	37
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278	3
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	36
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	37
...	...	...	...	...	...	
495	lewisjessica@craig-evans.com	4483 Jones Motorway Suite 872\nLake Jamiefurt,...	Tan	33.237660	13.566160	36
496	katrina56@gmail.com	172 Owen Divide Suite 497\nWest Richard, CA 19320	PaleVioletRed	34.702529	11.695736	37
497	dale88@hotmail.com	0787 Andrews Ranch Apt. 633\nSouth Chadburgh, ...	Cornsilk	32.646777	11.499409	38
498	cwilson@hotmail.com	680 Jennifer Lodge Apt. 808\nBrendachester, TX...	Teal	33.322501	12.391423	36
499	hannahwilson@davidson.com	49791 Rachel Heights Apt. 898\nEast Drewboroug...	DarkMagenta	33.715981	12.418808	35

500 rows × 8 columns

In [4]:
```
customers.info()
```

```
<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
```

We'll drop the object columns

In [5]:
```
df = customers.drop(["Email", "Address", "Avatar"], axis = 1)
```

In [6]:
```
df
```

Out[6]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	34.497268	12.655651	39.577668	4.082621	587.951054
1	31.926272	11.109461	37.268959	2.664034	392.204933
2	33.000915	11.330278	37.110597	4.104543	487.547505
3	34.305557	13.717514	36.721283	3.120179	581.852344
4	33.330673	12.795189	37.536653	4.446308	599.406092
...	...	...	...	...	...
495	33.237660	13.566160	36.417985	3.746573	573.847438
496	34.702529	11.695736	37.190268	3.576526	529.049004
497	32.646777	11.499409	38.332576	4.958264	551.620145
498	33.322501	12.391423	36.840086	2.336485	456.469510
499	33.715981	12.418808	35.771016	2.735160	497.778642

500 rows × 5 columns

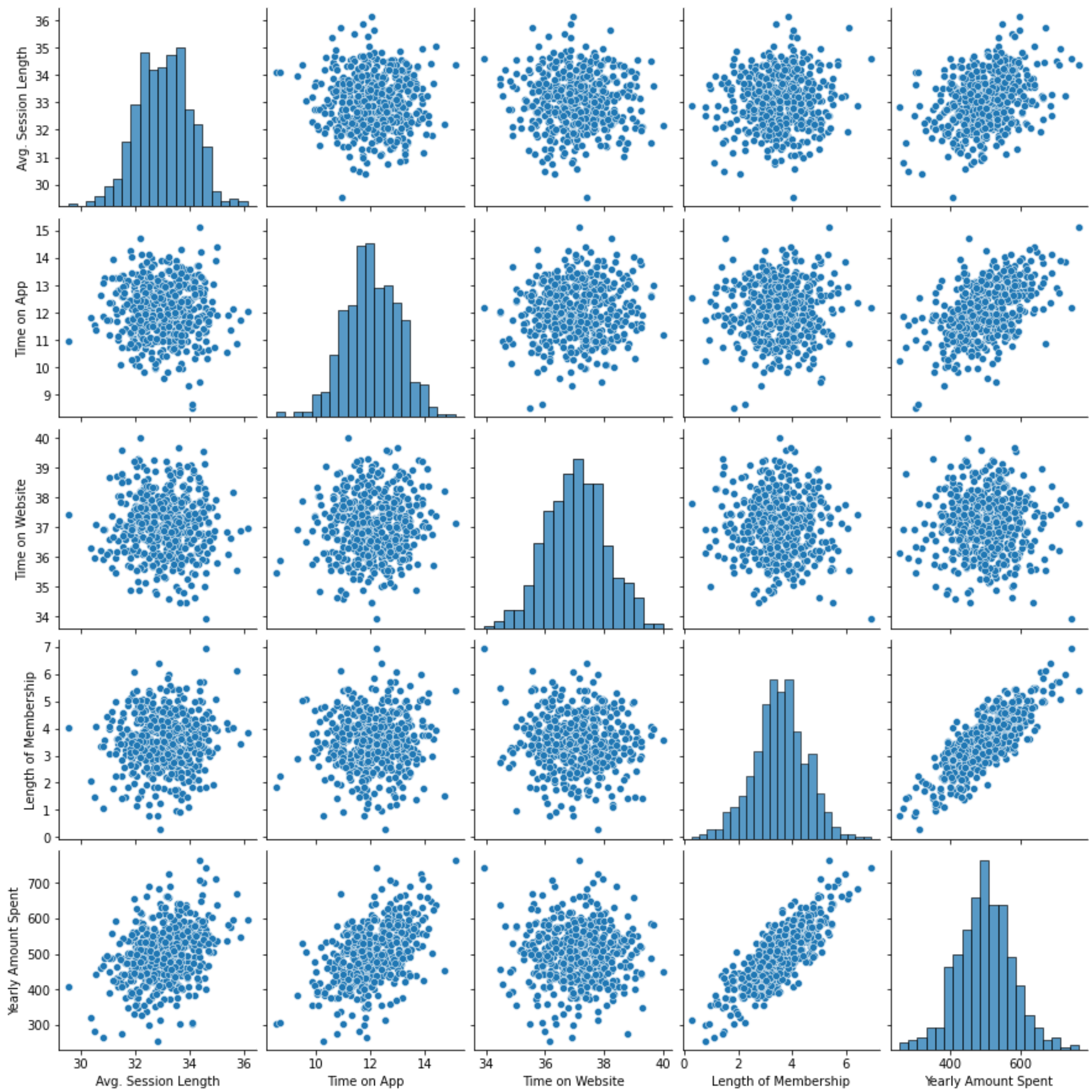
EDA

let's look at our dataset, and ask a few questions.

1. Does the yearly amount spent correlate with any other variables?

In [7]:
```
sns.pairplot(df)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x1ddbe513640>



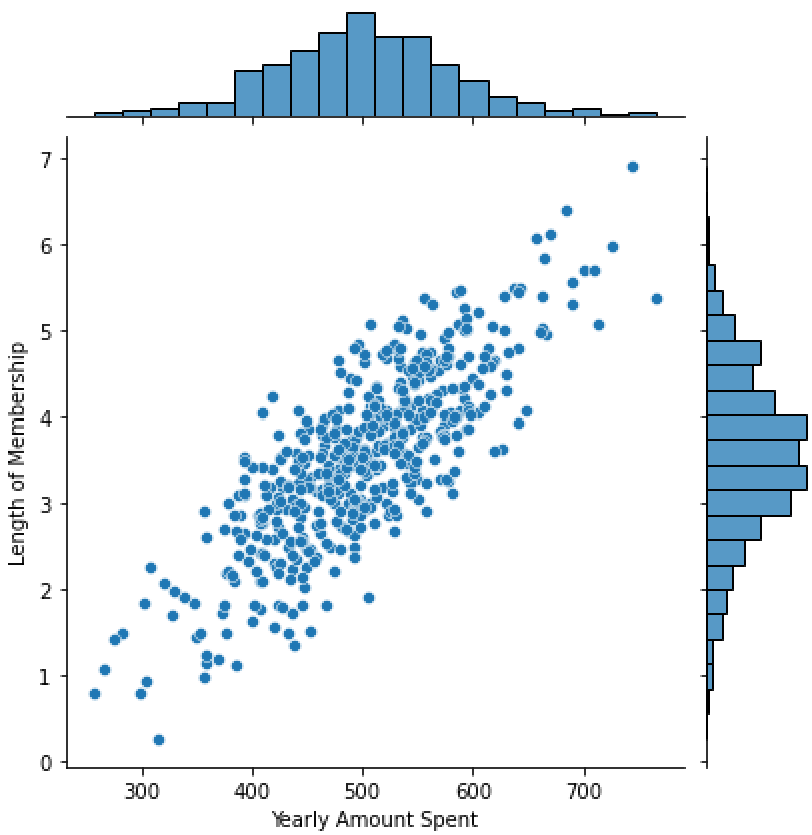
In [8]:
```
df.corr()
```

Out[8]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
Avg. Session Length	1.000000	-0.027826	-0.034987	0.060247	0.355088
Time on App	-0.027826	1.000000	0.082388	0.029143	0.499328
Time on Website	-0.034987	0.082388	1.000000	-0.047582	-0.002641
Length of Membership	0.060247	0.029143	-0.047582	1.000000	0.809084
Yearly Amount Spent	0.355088	0.499328	-0.002641	0.809084	1.000000

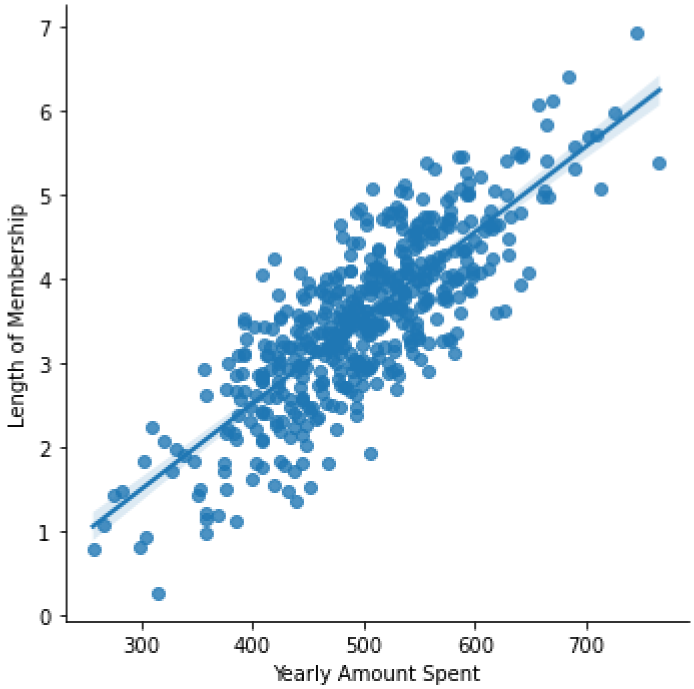
In [9]:
```
sns.jointplot(x = "Yearly Amount Spent", y = "Length of Membership", data = df)
```

Out[9]: <seaborn.axisgrid.JointGrid at 0x1ddc4579400>



In [10]:
```
#Create a linear model plot
sns.lmplot(x = "Yearly Amount Spent", y = "Length of Membership", data = df)
```

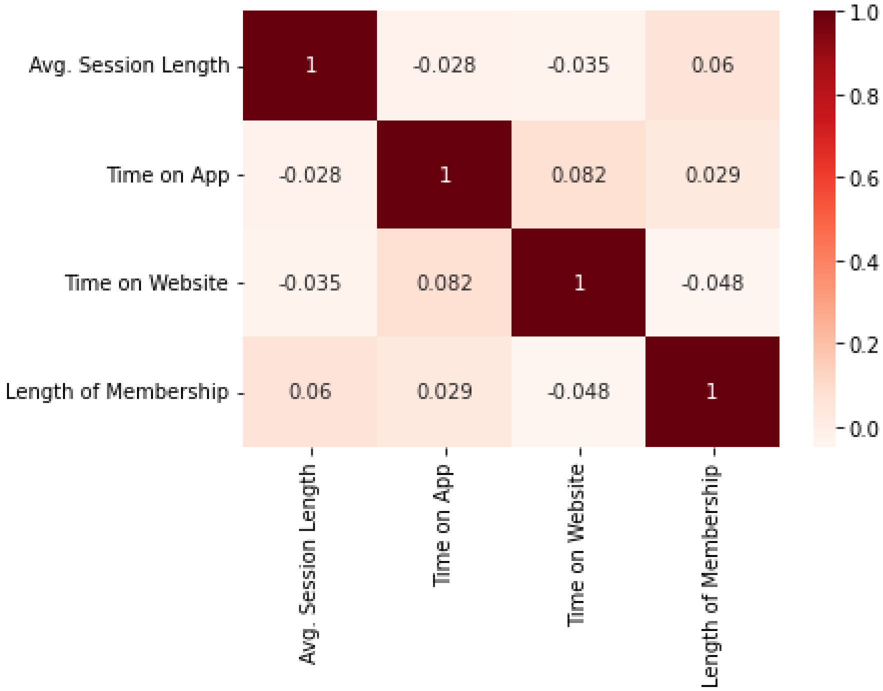
Out[10]: <seaborn.axisgrid.FacetGrid at 0x1ddc4959040>



Normally, you should spend more time on EDA but the objective here is to walk through building a linear ,model

In [11]:
```
sns.heatmap(df.drop(["Yearly Amount Spent"], axis = 1).corr(), annot = True, cmap
```

Out[11]: <AxesSubplot:>



Tarining and testing

We neeed to split our data to training and testing sets

Also, we need to separate the features and the label

In [12]:
```
 X = df[['Avg. Session Length', 'Time on App', 'Time on Website', 'Length of Memb

y = df['Yearly Amount Spent']
```

Sklearn has train test split method, we can specify our test size using this`m

In [13]:
```
from sklearn.model_selection import train_test_split
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, randon
#random - keeping your data steady
```

In [15]:
```
X_train
```

Out[15]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership
202	31.525752	11.340036	37.039514	3.811248
428	31.862741	14.039867	37.022269	3.738225
392	33.258238	11.514949	37.128039	4.662845
86	33.877779	12.517666	37.151921	2.669942
443	33.025020	12.504220	37.645839	4.051382
...	...	...	...	...
63	32.789773	11.670066	37.408748	3.414688
326	33.217188	10.999684	38.442767	4.243813
337	31.827979	12.461147	37.428997	2.974737
11	33.879361	11.584783	37.087926	3.713209
351	32.189845	11.386776	38.197483	4.808320

350 rows × 4 columns

In [16]:
```
X_test
```

Out[16]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership
18	32.187812	14.715388	38.244115	1.516576
361	32.077590	10.347877	39.045156	3.434560
104	31.389585	10.994224	38.074452	3.428860
4	33.330673	12.795189	37.536653	4.446308
156	32.294642	12.443048	37.327848	5.084861
...	...	...	...	...
147	32.255901	10.480507	37.338670	4.514122
346	32.765665	12.506548	35.823467	3.126509
423	33.128693	10.398458	36.683393	3.859818
17	32.338899	12.013195	38.385137	2.420806
259	32.096109	10.804891	37.372762	2.699562

150 rows × 4 columns

In [26]:
```
y_train
```

Out[26]:
```
202    443.965627
428    556.298141
392    549.131573
86     487.379306
443    561.516532
          ...    
63     483.159721
326    505.230068
337    440.002748
11     522.337405
351    533.396554
Name: Yearly Amount Spent, Length: 350, dtype: float64
```

In [27]:
```
y_test
```

Out[27]:
```
18     452.315675
361    401.033135
104    410.069611
4      599.406092
156    586.155870
          ...    
147    479.731938
346    488.387526
423    461.112248
17     407.704548
259    375.398455
Name: Yearly Amount Spent, Length: 150, dtype: float64
```

NB

We only use train data to train and test data to test, don't test on your training data!

In [28]:
```
from sklearn.linear_model import LinearRegression
```

In [29]:
```
#Create an instance of linear regression.
model = LinearRegression()
```

In [30]:
```
model.fit(X_train, y_train)
```

Out[30]:
```
LinearRegression()
```

In [31]:
```
X.columns
```

Out[31]:
```
Index(['Avg. Session Length', 'Time on App', 'Time on Website',
       'Length of Membership'],
      dtype='object')
```

In [32]:
```
model.coef_
```

Out[32]:
```
array([25.98154972, 38.59015875,  0.19040528, 61.27909654])
```

In [33]:
```
model.intercept_
```

Out[33]:
```
-1047.932782250239
```

In [34]:
```
#The equation of the regression is
# Yearly amount spent = 26* Avg.session length + 38.6*Time on App + 0.19 * Time o
```

Predicting on test data

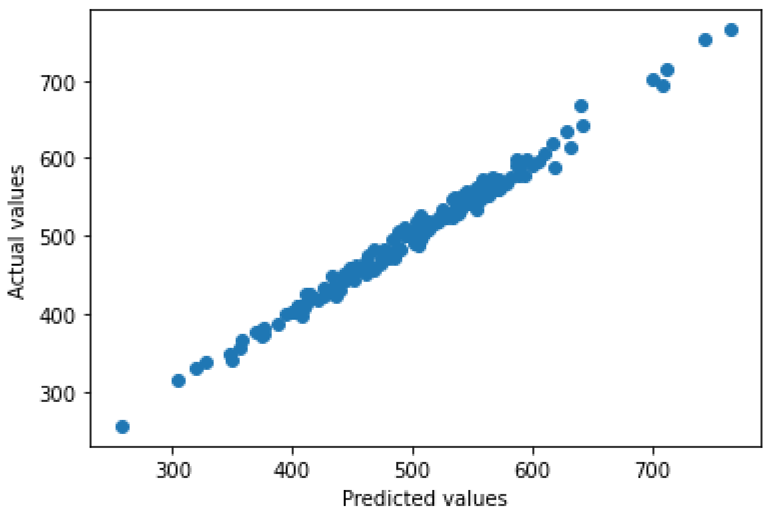
We want to evaluate the performance of the model on our test data

In [35]:
```
prediction = model.predict(X_test)
```

Let's compare our prediction with the y_test on a scatterplot

In [36]:
```
plt.pyplot.scatter(y_test, prediction)
plt.pyplot.xlabel("Predicted values")
plt.pyplot.ylabel("Actual values")
```

Out[36]:
```
Text(0, 0.5, 'Actual values')
```



Mini Exercise

1. What evaluation metrics do we use for linear regression?
2. Briefly discuss these metrics within your group?
3. Evaluate the model here using the metrics you identified.
4. Based on your evaluation, is this a good model? why?
5. Explain briefly what the values you obtain for each metric mean in this particular case.

1&2:

What evaluation metrics do we use for linear regression? Briefly discuss these metrics within your group?

Mean Squared Error

- Used to find the average squared difference between the actual and the predicted value. It measures how close a regression line is to a set of data points. MSE of 0 means all the predicted values match the expected values exactly. The lower the value the better.

  Mean Absolute Error
- Used to find the mean absolute distance when making predictions to know how close the predictions are to the actual model on average.Low MAE values show the model is correctly predicted. Large MAE show the model is poor at predicting.

  R-Squared
- Determines the accuracy of our model in terms of distance. Values close to 100% indicate high accuracy.

3, 4, 5. Evaluate the model here using the metrics you identified.

In [58]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
```

In [59]:
```
mse = mean_squared_error(y_test,prediction)
mse
```

Out[59]:
```
79.81305165097451
```

In [41]:
```
mae =mean_absolute_error(y_test,prediction)
mae
```

Out[41]:
```
7.228148653430837
```

In [43]:
```
r2 = r2_score(y_test,prediction)
r2
```

Out[43]:
```
0.9890046246741234
```

In [57]:
```
print("The model predicted the values with", (r2 *100) ,"% accuracy")
```

```
The model predicted the values with 98.90046246741234 % accuracy
```

According to the R-Squared the model is a good fit, however, the MAE and MSE show that the model is poor at predicting.