Problem Statement:

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!

Just follow the steps below to analyze the customer data (it's fake, don't worry I didn't give you real credit card numbers or emails).

Imports

** Import pandas, numpy, matplotlib,and seaborn. Then set %matplotlib inline (You'll import sklearn as you need it.)**

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

Get the Data

We'll work with the Ecommerce Customers csv file from the company. It has Customer info, suchas Email, Address, and their color Avatar. Then 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.

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

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

Check the head of customers, and check out its info() and describe() methods.

In [3]:
```
customers.head()
```

Out[3]:

	Email	Address	Avatar	Avg. Session Length	Time on App	Ti W
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651	39.5
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461	37.2
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278	37.1
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	36.7
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	37.5

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

Out[4]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
count	500.000000	500.000000	500.000000	500.000000	500.000000
mean	33.053194	12.052488	37.060445	3.533462	499.314038
std	0.992563	0.994216	1.010489	0.999278	79.314782
min	29.532429	8.508152	33.913847	0.269901	256.670582
25%	32.341822	11.388153	36.349257	2.930450	445.038277
50%	33.082008	11.983231	37.069367	3.533975	498.887875
75%	33.711985	12.753850	37.716432	4.126502	549.313828
max	36.139662	15.126994	40.005182	6.922689	765.518462

In [5]:
```
customers.info()
```

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

Exploratory Data Analysis

Let's explore the data!

For the rest of the exercise we'll only be using the numerical data of the csv file.

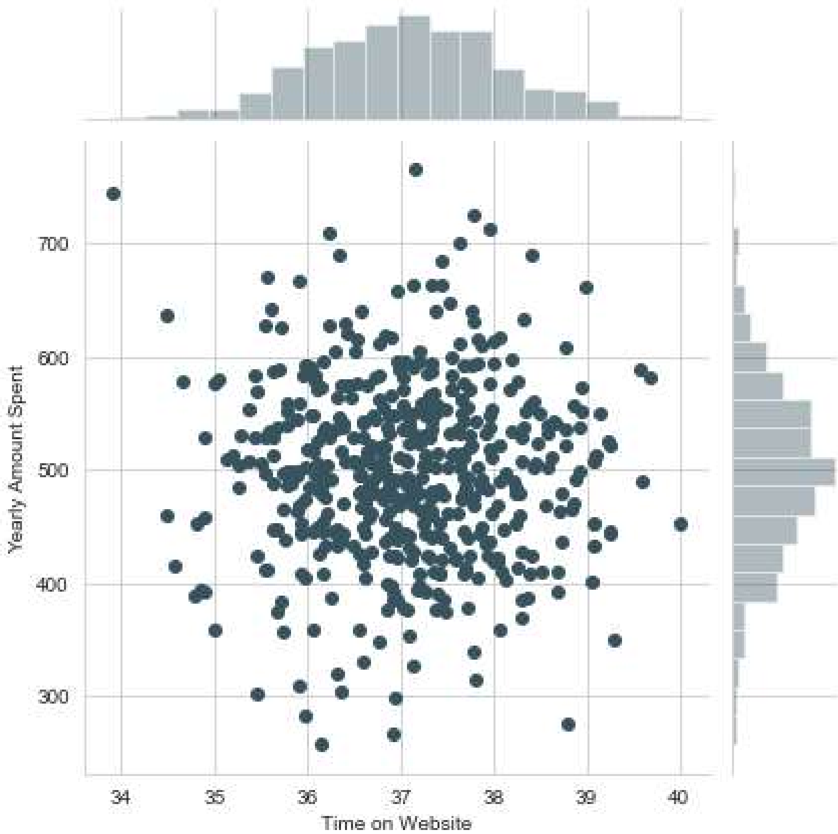
Use seaborn to create a jointplot to compare the Time on Website and Yearly Amount Spent columns. Does the correlation make sense?

In [6]:
```
sns.set_palette("GnBu_d")
sns.set_style('whitegrid')
```

** Do the same but with the Time on App column instead. **

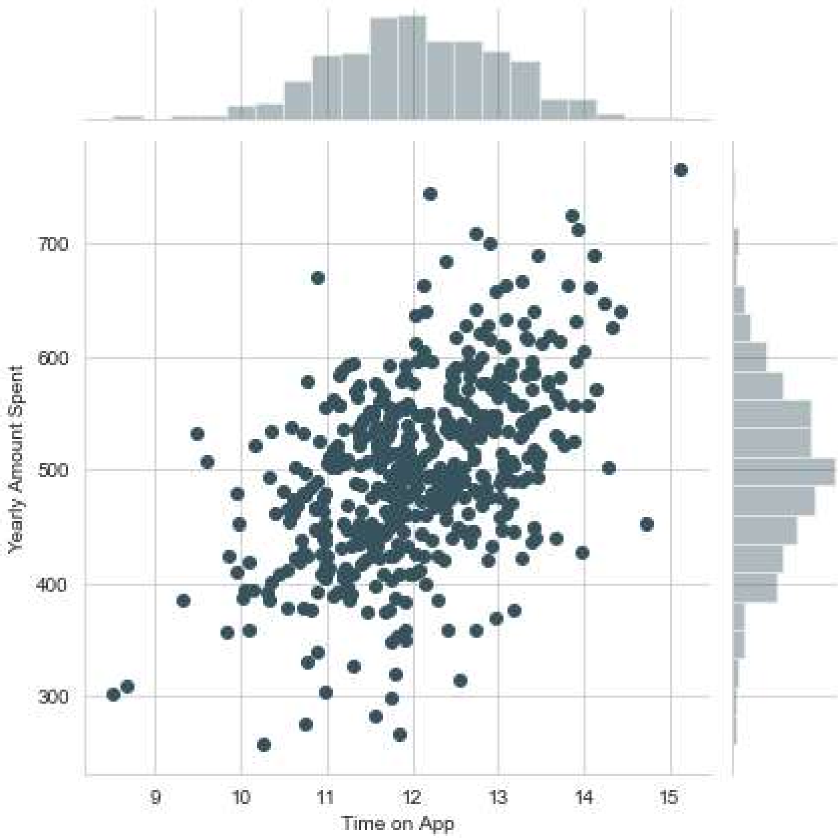
In [7]:
```
# More time on site, more money spent.
sns.jointplot(x='Time on Website',y='Yearly Amount Spent',data=customers)
```

Out[7]:
```
<seaborn.axisgrid.JointGrid at 0x2b6e7775860>
```



In [8]:
```
# as we can see there is liner relationship between "Time on App" and "yearly spe
sns.jointplot(x='Time on App',y='Yearly Amount Spent',data=customers)
```

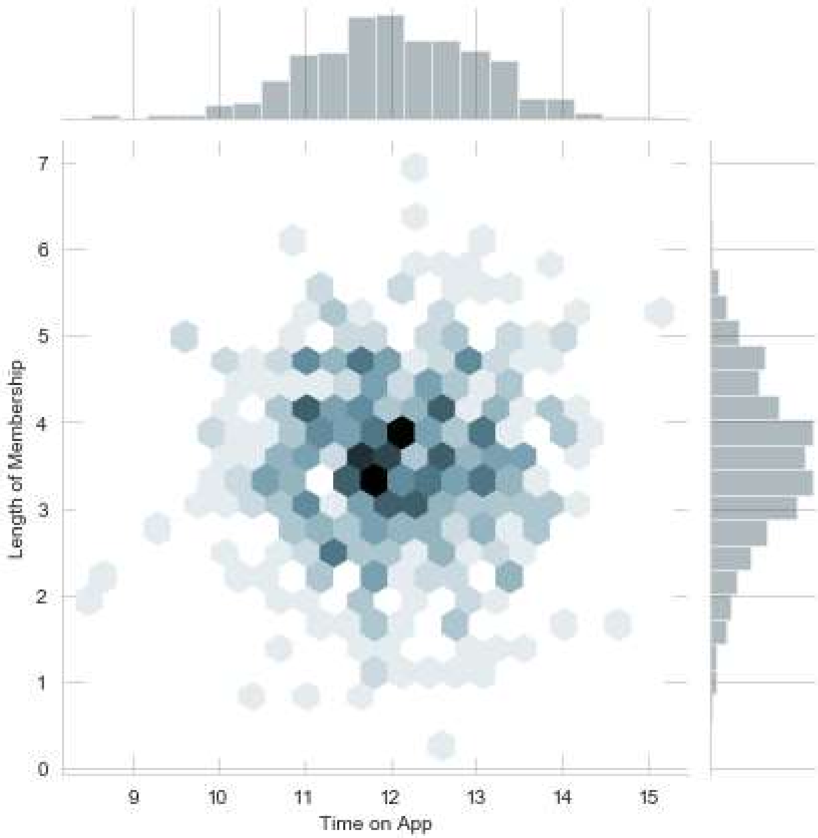
Out[8]: <seaborn.axisgrid.JointGrid at 0x2b6e7b7b630>



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

In [9]:
```
sns.jointplot(x='Time on App',y='Length of Membership',kind='hex',data=customers)
```

Out[9]: <seaborn.axisgrid.JointGrid at 0x2b6e7d23f98>

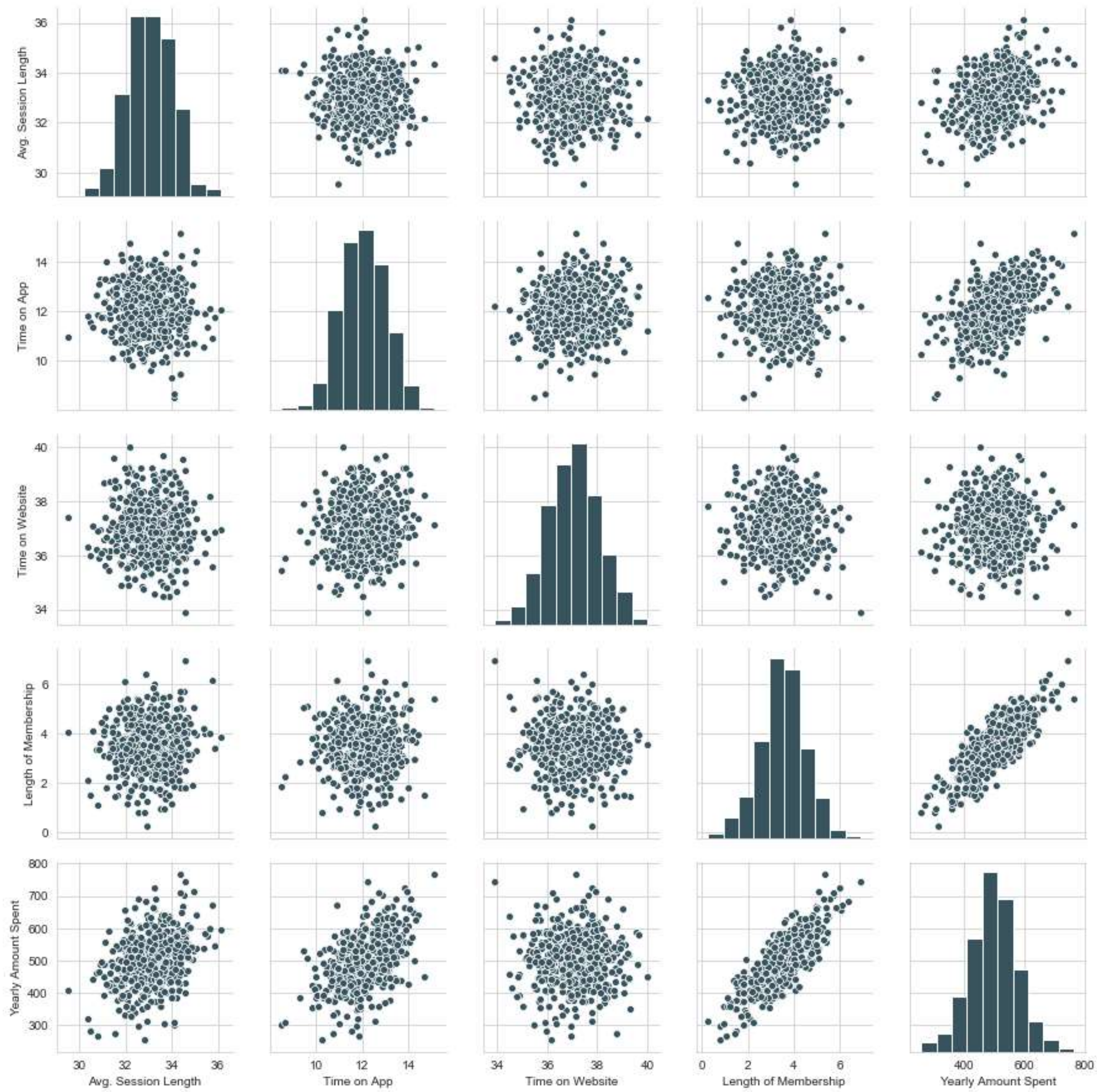


In [10]:
```
# as above hex, we can see black portion where highest time and amount has been s
```

**Let's explore these types of relationships across the entire data set. Use [pairplot (https://stanford.edu/~mwaskom/software/seaborn/tutorial/axis_grids.html#plotting-pairwise-relationships-with-pairgrid-and-pairplot)](https://stanford.edu/~mwaskom/software/seaborn/tutorial/axis_grids.html#plotting-pairwise-relationships-with-pairgrid-and-pairplot) to recreate the plot

In [11]:
```
sns.pairplot(customers)
```

Out[11]: <seaborn.axisgrid.PairGrid at 0x2b6e7e85ef0>



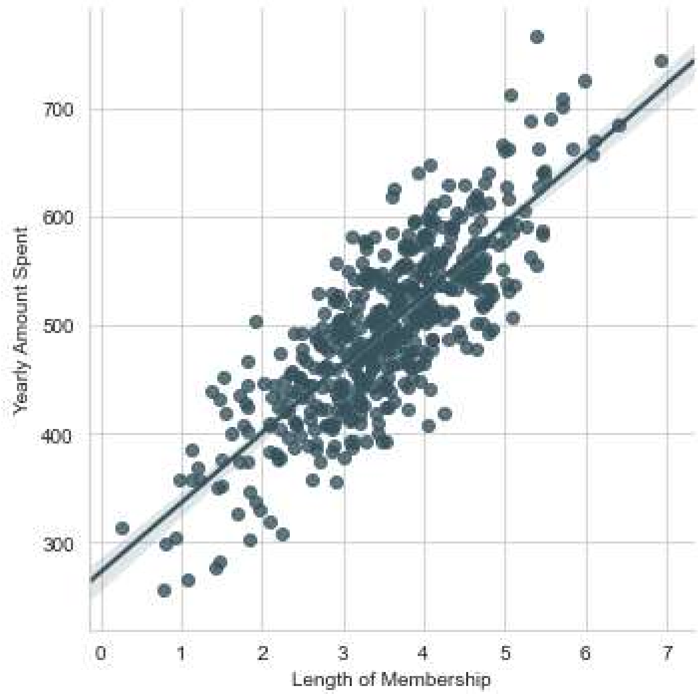
Based off this plot what looks to be the most correlated feature with Yearly Amount Spent?

In [12]:
```
# Based on above plots we can see between 'Length of Membership' and 'Yeraly Amou
#in deep and creat linear plot to see how's it look
```

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

In [13]:
```
sns.lmplot(x='Length of Membership',y='Yearly Amount Spent',data=customers)
```

Out[13]: <seaborn.axisgrid.FacetGrid at 0x2b6e8aaa320>



Training and Testing Data

Now that we've explored the data a bit, let's go ahead and split the data into training and testing sets. ** Set a variable X equal to the numerical features of the customers and a variable y equal to the "Yearly Amount Spent" column. **

In [14]:
```
y = customers['Yearly Amount Spent']
```

In [15]:
```
X = customers[['Avg. Session Length', 'Time on App','Time on Website', 'Length of
```

** 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**

In [16]:
```
from sklearn.model_selection import train_test_split
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s
```

Training the Model

Now its time to train our model on our training data!

** Import LinearRegression from sklearn.linear_model **

In [18]:
```
from sklearn.linear_model import LinearRegression
```

In [19]:
```
##Create an instance of a LinearRegression() model named lm
lm=LinearRegression()
```

** Train/fit lm on the training data.**

In [20]:
```
lm.fit(X_train,y_train)
```

Out[20]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Print out the coefficients of the model

In [67]:
```
lm.coef_
```

Out[67]:
```
array([25.98154972, 38.59015875,  0.19040528, 61.27909654])
```

Predicting Test Data

Now that we have fit our model, let's evaluate its performance by predicting off the test values!

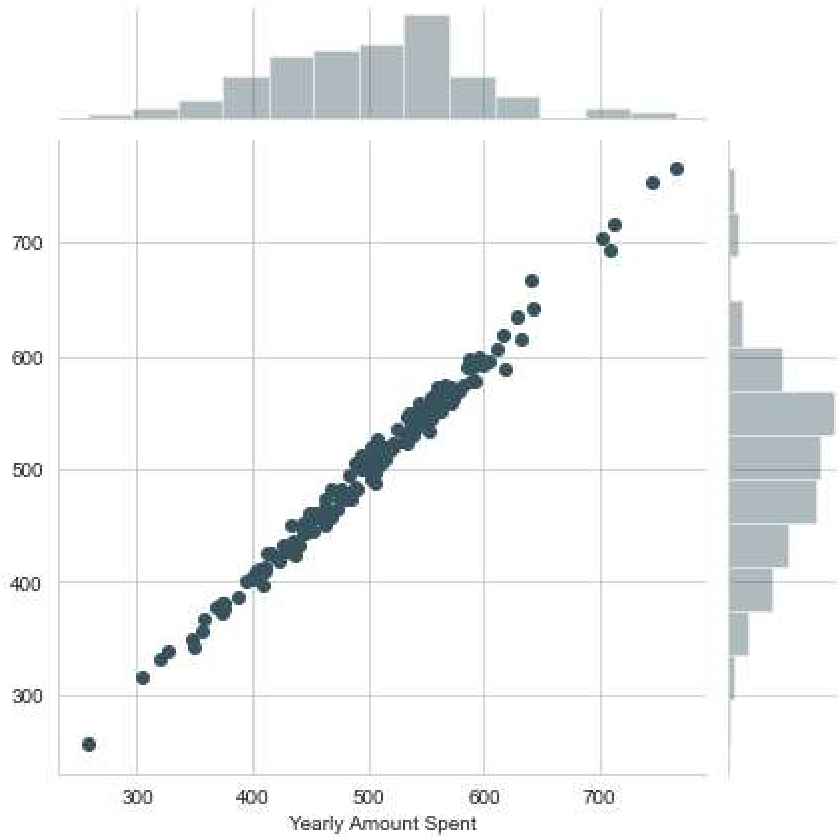
** Use lm.predict() to predict off the X_test set of the data.**

In [21]:
```
prediction= lm.predict(X_test)
```

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

In [22]:
```
sns.jointplot(x=y_test,y=prediction)
```

Out[22]: <seaborn.axisgrid.JointGrid at 0x2b6eba30f28>



Evaluating the Model

Let's evaluate our model performance by calculating the residual sum of squares and the explained variance score (R^2).

** Calculate the Mean Absolute Error, Mean Squared Error, and the Root Mean Squared Error. Refer to the lecture or to Wikipedia for the formulas**

In [26]:
```
from sklearn import metrics
```

In [27]:
```
metrics.mean_absolute_error(prediction,y_test)
```

Out[27]:
```
7.228148653430853
```

In [28]:
```
metrics.mean_squared_error(prediction,y_test)
```

Out[28]:
```
79.81305165097487
```

In [29]:
```
np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

Out[29]:
```
8.933815066978656
```

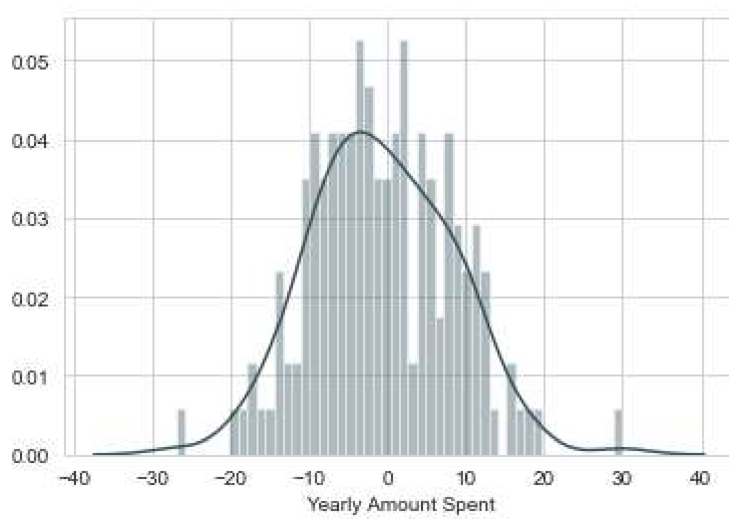
Residuals

You should have gotten a very good model with a good fit. Let's quickly explore the residuals to make sure everything was okay with our data.

Plot a histogram of the residuals and make sure it looks normally distributed. Use either seaborn distplot, or just plt.hist().

In [33]:
```
sns.distplot(y_test-prediction,bins=50)
```

Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b6ef4e0128>
```



Conclusion

We still want to figure out the answer to the original question, do we focus our efforst on mobile app or website development? Or maybe that doesn't even really matter, and Membership Time is what is really important. Let's see if we can interpret the coefficients at all to get an idea.

** Recreate the dataframe below. **

In [38]:
```
coff=pd.DataFrame(lm.coef_,X.columns,columns=['coff'])
coff
```

Out[38]:

	coff
Avg. Session Length	25.981550
Time on App	38.590159
Time on Website	0.190405
Length of Membership	61.279097

** How can you interpret these coefficients? **

Holding all other features fixed, a 1 unit increase in Avg. Session Length is associated with an increase of 25.98 total dollars spent.

Holding all other features fixed, a 1 unit increase in Time on App is associated with an increase of 38.59 total dollars spent.

Holding all other features fixed, a 1 unit increase in Time on Website is associated with an increase of 0.19 total dollars spent.

Holding all other features fixed, a 1 unit increase in Length of Membership is associated with an increase of 61.27 total dollars spent.

Do you think the company should focus more on their mobile app or on their website?

This is tricky, there are two ways to think about this: Develop the Website to catch up to the performance of the mobile app, or develop the app more since that is what is working better. This sort of answer really depends on the other factors going on at the company, you would probably want to explore the relationship between Length of Membership and the App or the Website before coming to a conclusion!