Linear Regression - Project

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
customers= pd.read_csv('Ecommerce Customers')
```

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

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

Out[4]:

	Email	Address	Avatar	Avg. Session Length	Time on App	Tir We
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 [5]:
```
customers.describe()
```

Out[5]:

	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 [6]:
```
customers.info()
```

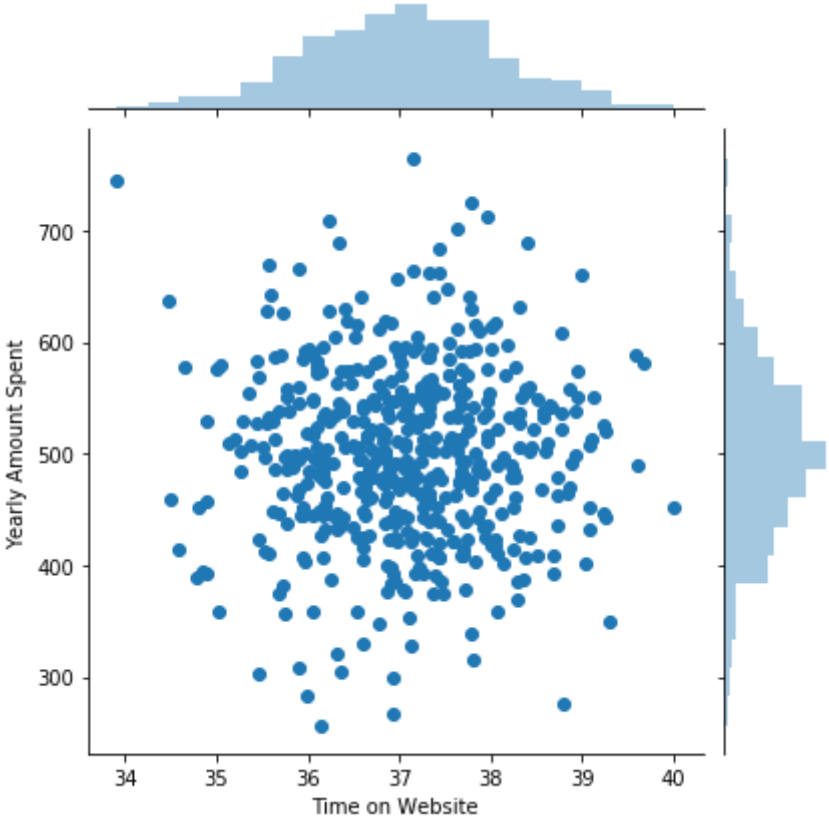
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
Email                   500 non-null object
Address                 500 non-null object
Avatar                  500 non-null object
Avg. Session Length     500 non-null float64
Time on App             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: 31.3+ KB
```

Exploratory Data Analysis

Let's explore the data!

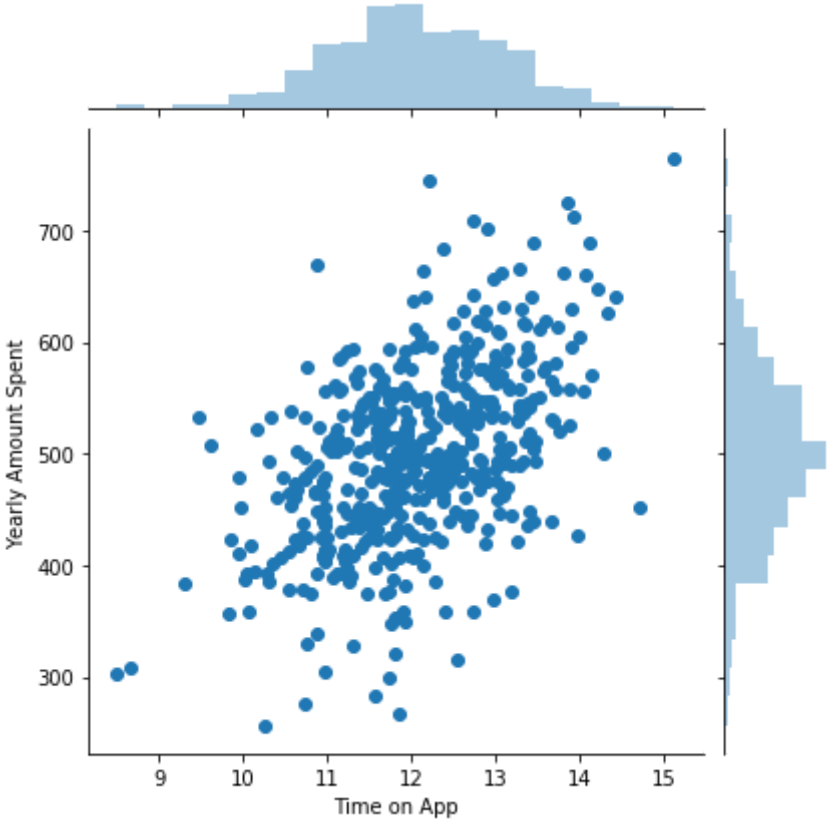
In [7]:
```
sb.jointplot(x='Time on Website',y='Yearly Amount Spent',data=customers)
```

Out[7]:
```
<seaborn.axisgrid.JointGrid at 0x122128f28>
```



In [8]:
```
sb.jointplot(x='Time on App',y='Yearly Amount Spent',data=customers)
```

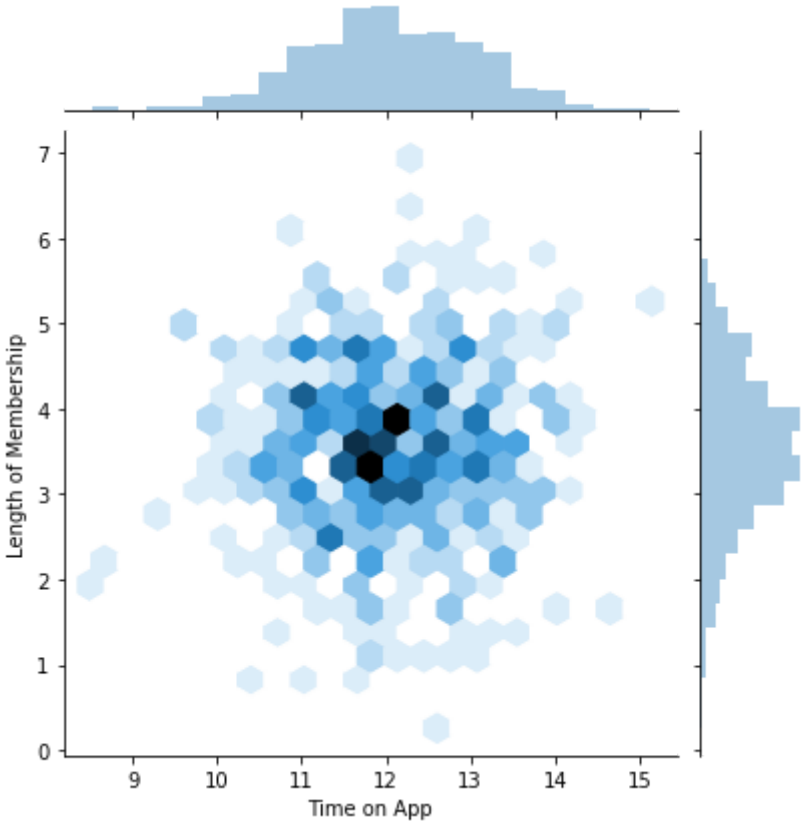
Out[8]:
```
<seaborn.axisgrid.JointGrid at 0x1240a5208>
```



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

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

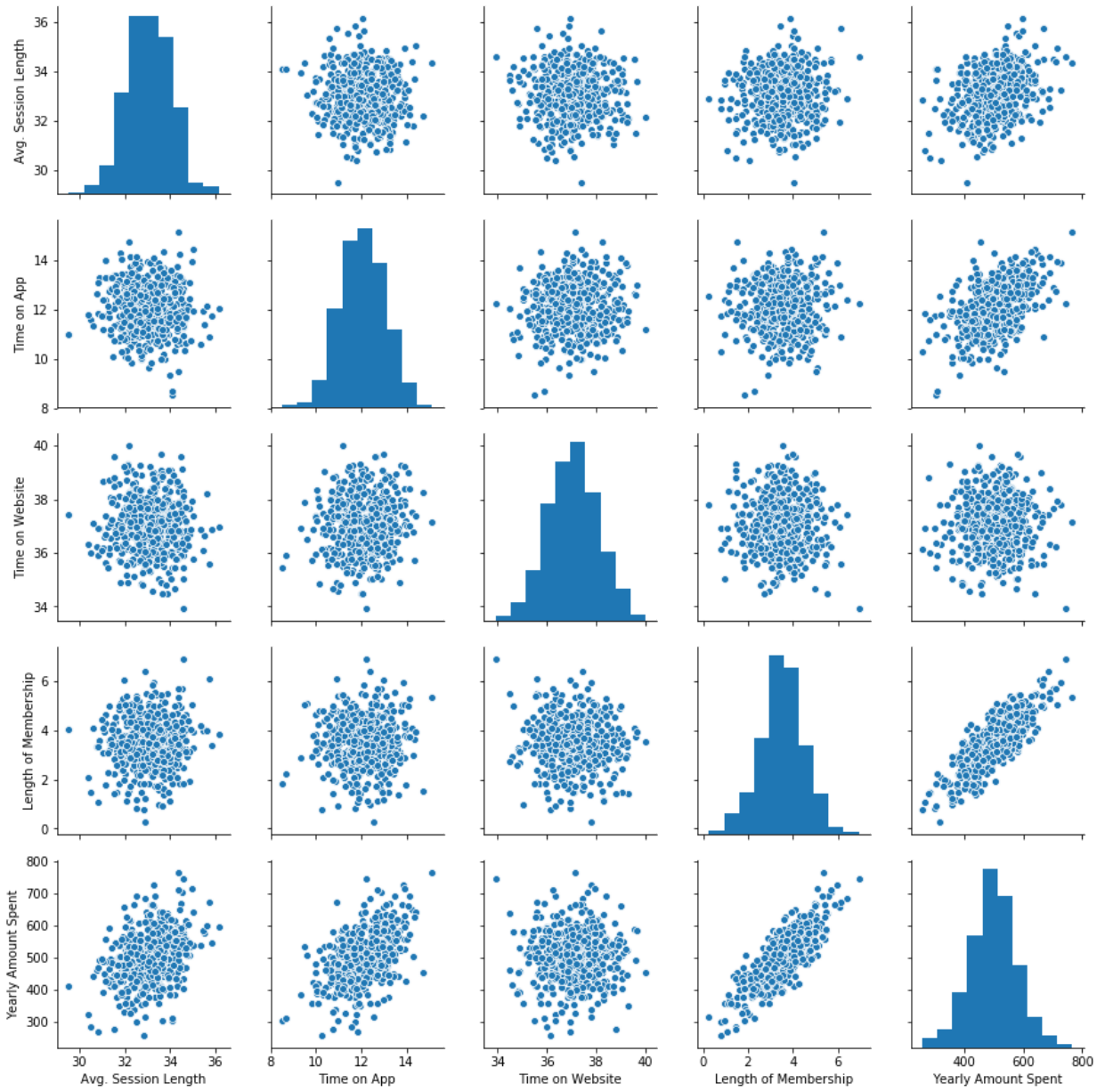
Out[9]:
```
<seaborn.axisgrid.JointGrid at 0x12622df60>
```



Let's explore these types of relationships across the entire data set.

In [10]:
```
sb.pairplot(customers)
```

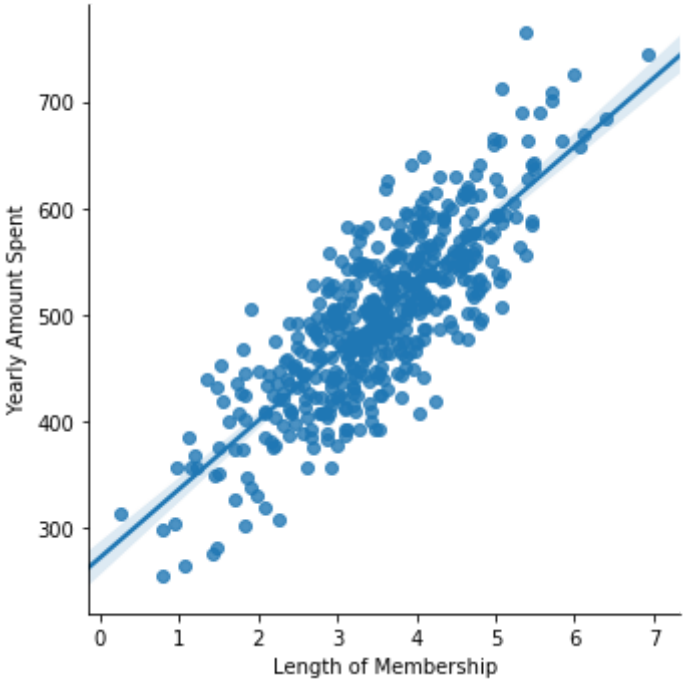
Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x1267918d0>
```



Based off this plot what looks to be the most correlated feature with Yearly Amount Spent is Length of membership

In [11]:
```
sb.lmplot(x='Length of Membership',y='Yearly Amount Spent',data=customers)
```

Out[11]:
```
<seaborn.axisgrid.FacetGrid at 0x12754d358>
```



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 [12]:
```
customers.columns
```

Out[12]:
```
Index(['Email', 'Address', 'Avatar', 'Avg. Session Length', 'Time on Ap
p',
       'Time on Website', 'Length of Membership', 'Yearly Amount Spent'],
      dtype='object')
```

In [13]:
```
x= customers[['Avg. Session Length', 'Time on App',
       'Time on Website', 'Length of Membership']]
```

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

** Using model_selection.train_test_split from sklearn to split the data into training and testing sets.**

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, ra
```

Training the Model

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

In [17]:
```
from sklearn.linear_model import LinearRegression
```

Create an instance of a LinearRegression() model named lm.

In [18]:
```
lm= LinearRegression()
```

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

In [19]:
```
lm.fit(X_train,y_train)
```

Out[19]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

Print out the coefficients of the model

In [20]:
```
lm.coef_
```

Out[20]:
```
array([25.98154972, 38.59015875,  0.19040528, 61.27909654])
```

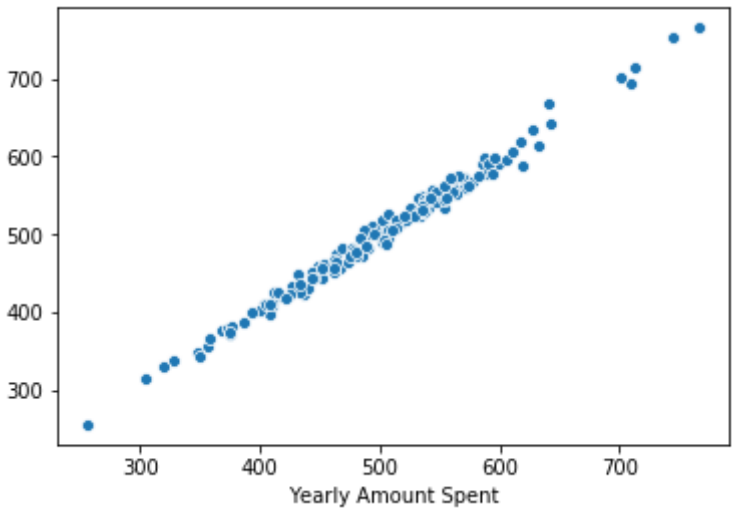
Predicting Test Data

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

In [22]:
```
predicted=lm.predict(X_test)
```

In [23]:
```
sb.scatterplot(y_test,predicted)
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1294d5470>
```



Evaluating the Model

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

Mean Absolute Error, Mean Squared Error, and the Root Mean Squared Error

In [24]:
```
from sklearn import metrics
```

In [25]:
```
print("MAE:",metrics.mean_absolute_error(y_test,predicted))
print("MSE:",metrics.mean_squared_error(y_test,predicted))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test,predicted)))
```

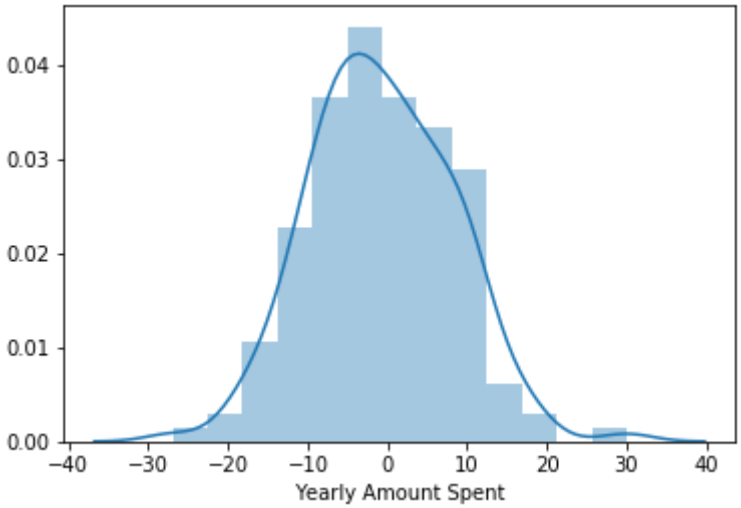
```
MAE: 7.2281486534308295
MSE: 79.8130516509743
RMSE: 8.933815066978626
```

Residuals

Plot a histogram of the residuals

In [26]:
```
sb.distplot((y_test-predicted))
```

Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12ae95908>
```



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.

In [28]:
```
x.columns
```

Out[28]:
```
Index(['Avg. Session Length', 'Time on App', 'Time on Website',
       'Length of Membership'],
      dtype='object')
```

In [33]:
```
pd.DataFrame(lm.coef_,x.columns, columns=['Coeffeicient'])
```

Out[33]:

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

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

Membership time has more influence on a users yearly spending but also the focus towards improving mobile app should also be increased. Putting efforts on website is not going to help but focus towards membership time of a user and the mobile application should be more.