Imports

In [40]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

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

In [41]:
```python
customers = pd.read_csv('Ecommerce Customers')
```

In [42]:
```python
customers.head()
```

Out[42]:

	Email	Address	Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651	39.577668	4.082621	587.951054
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461	37.268959	2.664034	392.204933
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278	37.110597	4.104543	487.547505
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	36.721283	3.120179	581.852344
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	37.536653	4.446308	599.406092

In [43]:
```python
customers.describe()
```

Out[43]:

	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 [44]:
```python
customers.info()
```

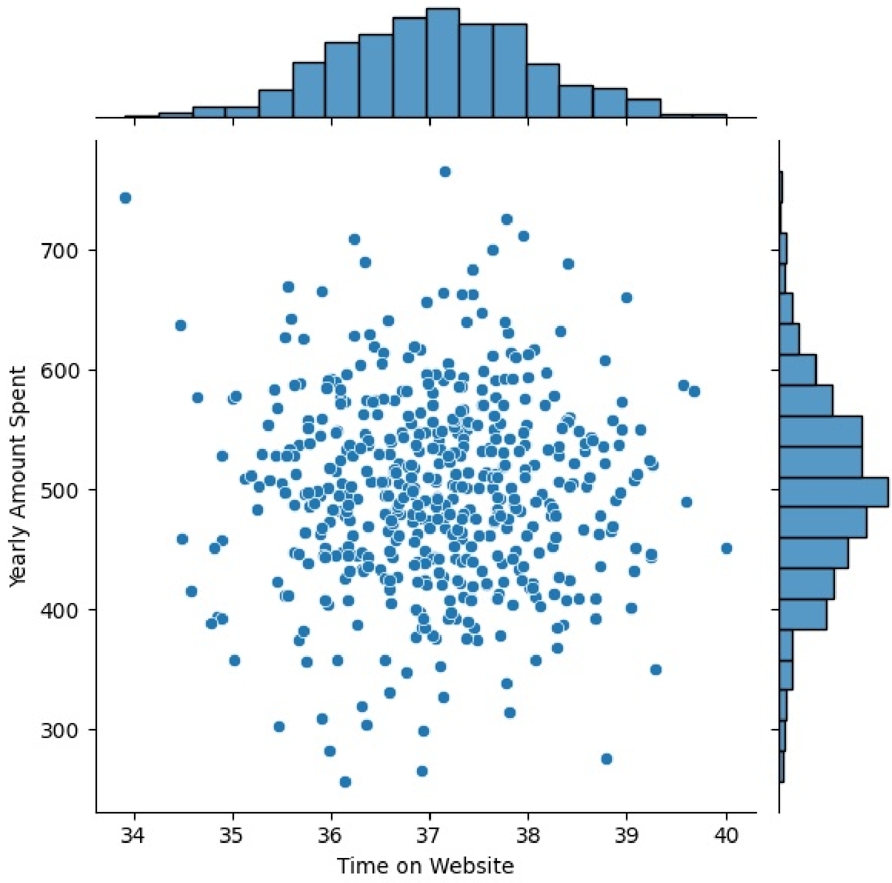
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Email                 500 non-null    object 
 1   Address               500 non-null    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
```

Data Analysis

In [45]:
```python
import seaborn as sns
```

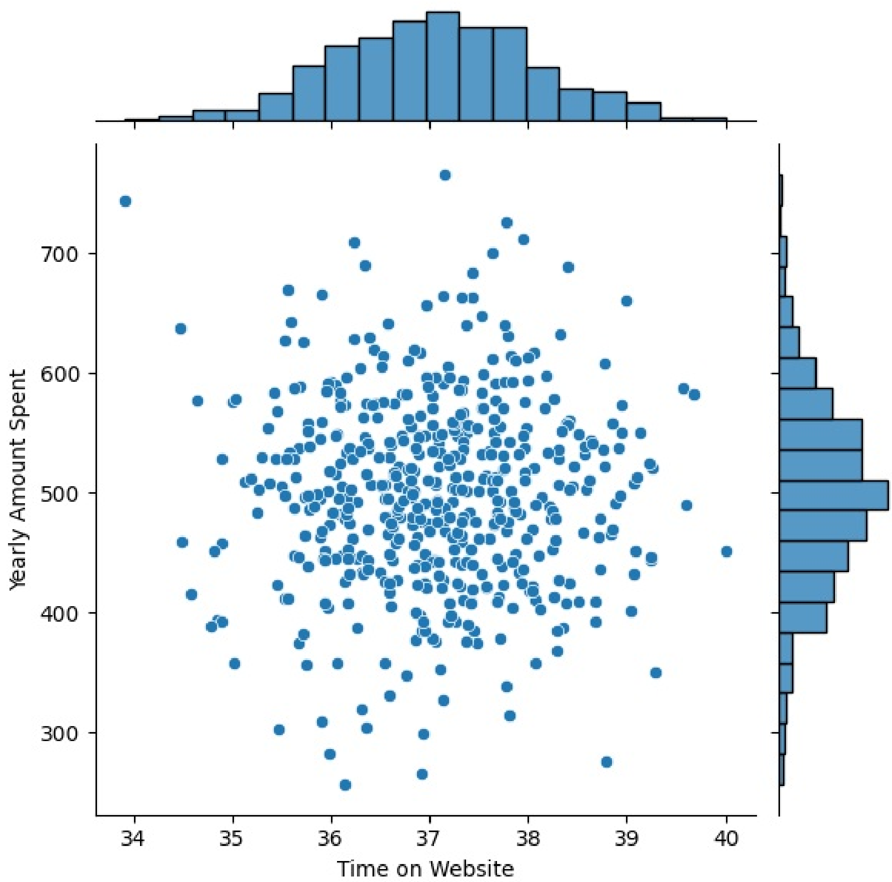
In [50]:
```python
sns.jointplot(x='Time on Website', y='Yearly Amount Spent', data=customers)
```

```
plt.show()
```



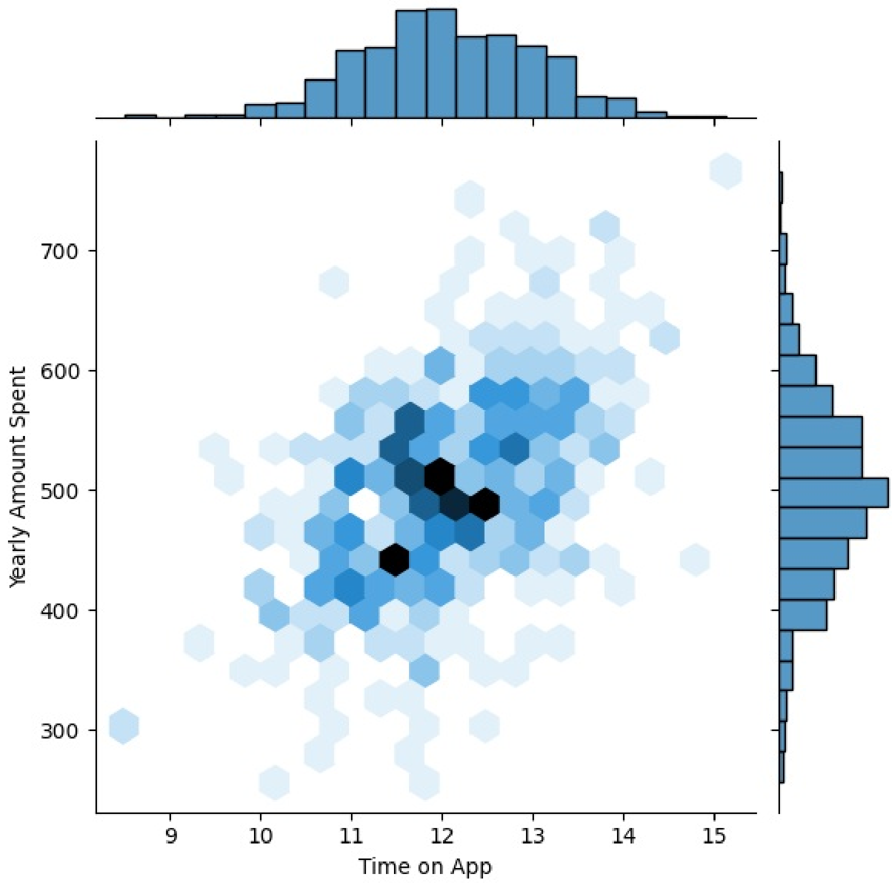
** Do the same but with the Time on App column instead. **

In [51]:
```
sns.jointplot(x='Time on Website', y='Yearly Amount Spent', data=customers)
plt.show()
```



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

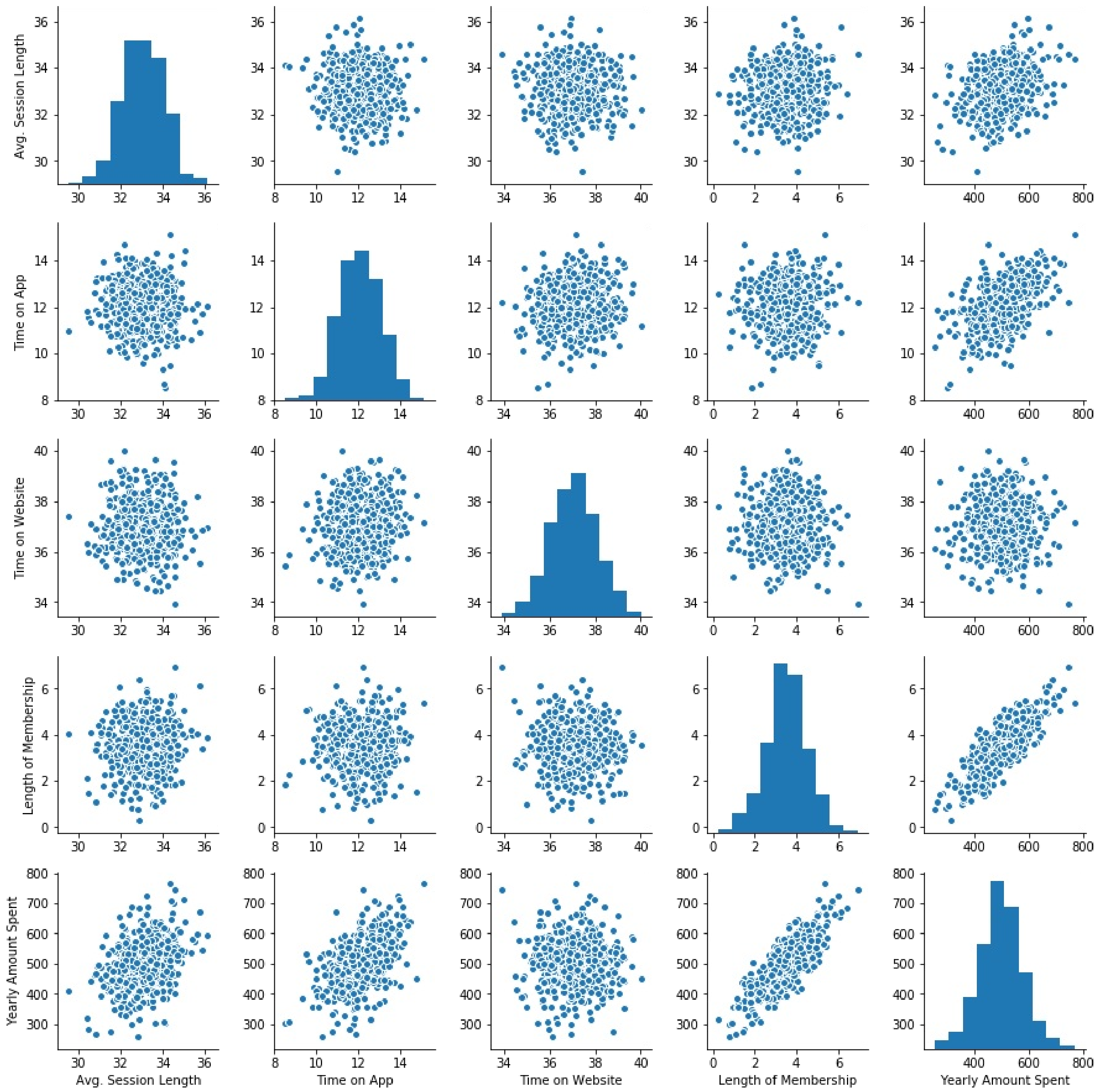
In [52]:
```
sns.jointplot(x='Time on App',y='Yearly Amount Spent',data=customers, kind='hex')
plt.show()
```



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

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

Out[ ]: <seaborn.axisgrid.PairGrid at 0x1a1e8218d0>



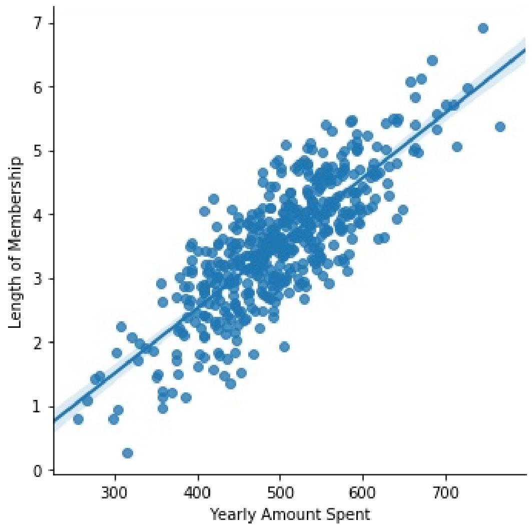
Based off this plot what looks to be the most correlated feature with Yearly Amount Spent?

In [ ]:
```
#Length of Membership
```

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

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

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1a1fa21e80>
```



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

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

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

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

Training the Model

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

** Import LinearRegression from sklearn.linear_model **

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

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

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

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

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

Out[ ]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Print out the coefficients of the model

In [ ]:
```
print('Coefficients: \n', lm.coef_)
```

```
Coefficients:
 [ 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!

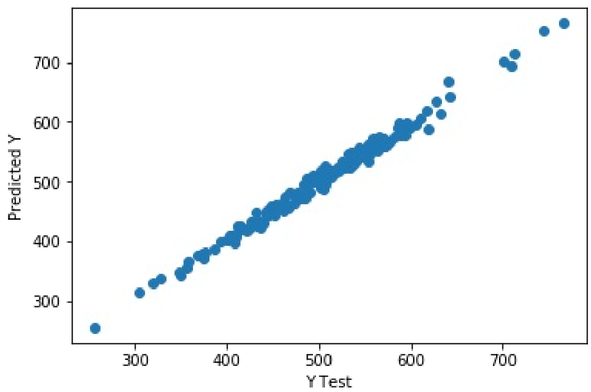
** Use lm.predict() to predict off the X_test set of the data.**

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

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

In [ ]:
```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

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



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.

In [ ]:
```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

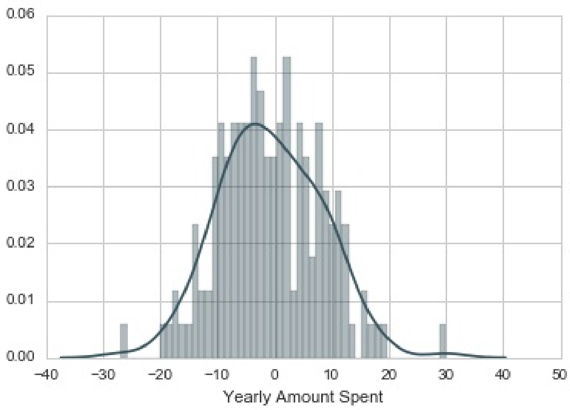
```
MAE: 7.22814865343
MSE: 79.813051651
RMSE: 8.93381506698
```

Residuals

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 [ ]:
```
sns.distplot((y_test-predictions),bins=50);
```



Conclusion

We still want to figure out the answer to the original question, do we focus our efforts 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 [ ]:

```
coeffecients = pd.DataFrame(lm.coef_,X.columns)
coeffecients.columns = ['Coeffecient']
coeffecients
```

Out[ ]:

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

Out of Mobile App and Website, the company should focus more on:

Mobile App

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js