In [58]:

```
import numpy as np
import pandas as pd
```

In [59]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [60]:

```
customers = pd.read_csv(r'C:\Users\DELL\Downloads\3PythonCourse\Refactored_Py_DS_ML_Bootcamp-maste
```

In [61]:

```
customers.head()
```

Out[61]:

	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	3
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461	3
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278	
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	3
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	3

In [62]:

```
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
```

In [63]:

```
customers.describe()
```

Out[63]:

	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

EDA

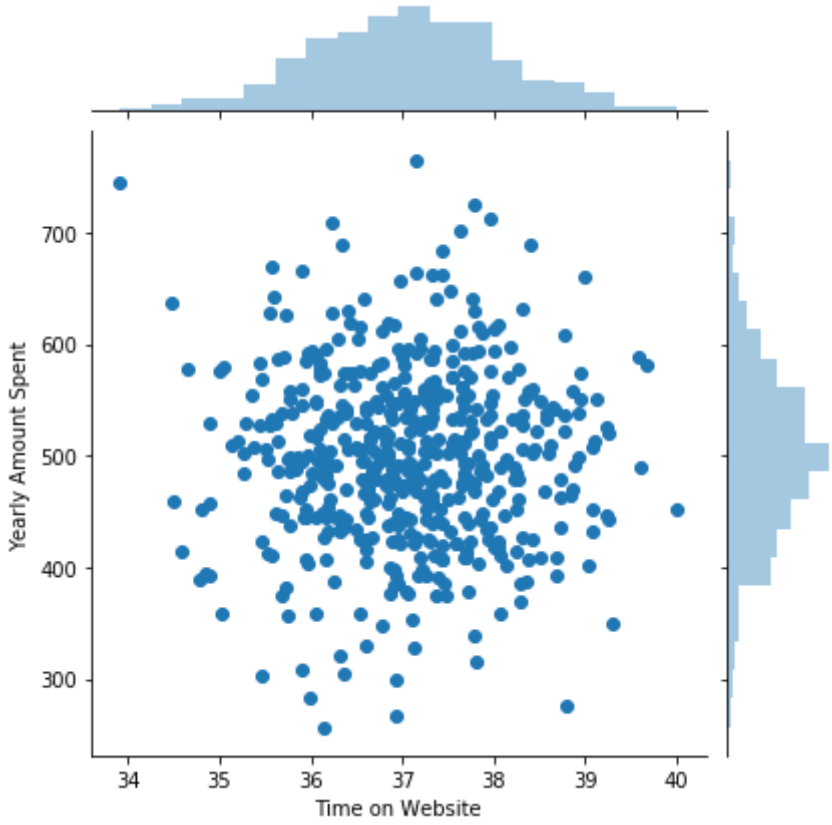
In [64]:

```
sns.jointplot(x='Time on Website', y = 'Yearly Amount Spent', data = customers)
```

C:\Users\DELL\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval

Out[64]:

<seaborn.axisgrid.JointGrid at 0x217b9c9c438>

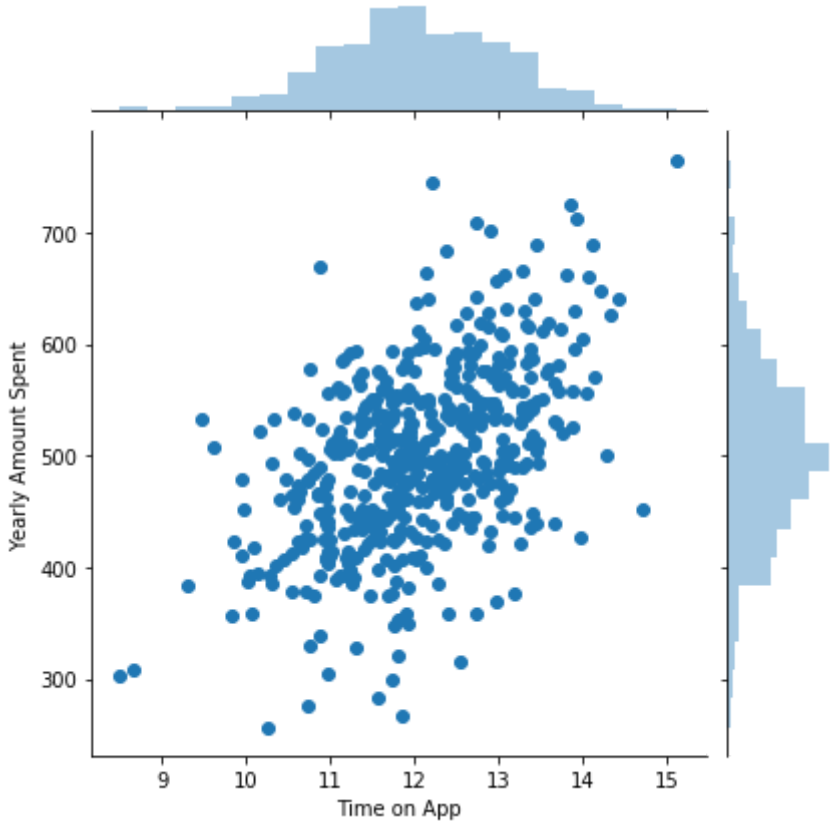


In [65]:

```
sns.jointplot(x='Time on App', y = 'Yearly Amount Spent', data = customers)
```

Out[65]:

<seaborn.axisgrid.JointGrid at 0x217b97bea58>

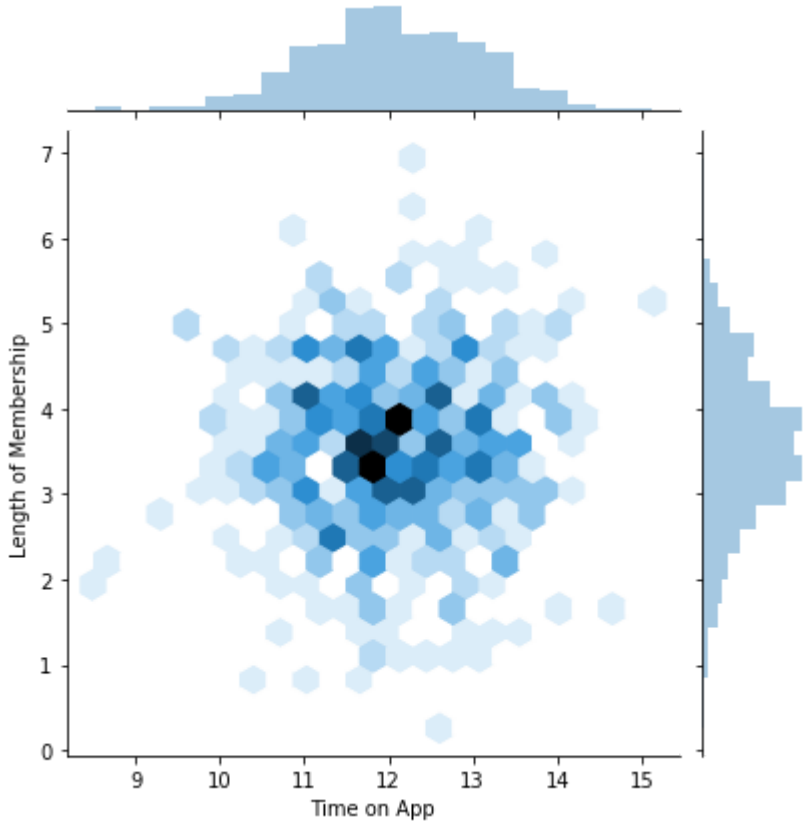


In [66]:

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

Out[66]:

<seaborn.axisgrid.JointGrid at 0x217b996e7b8>

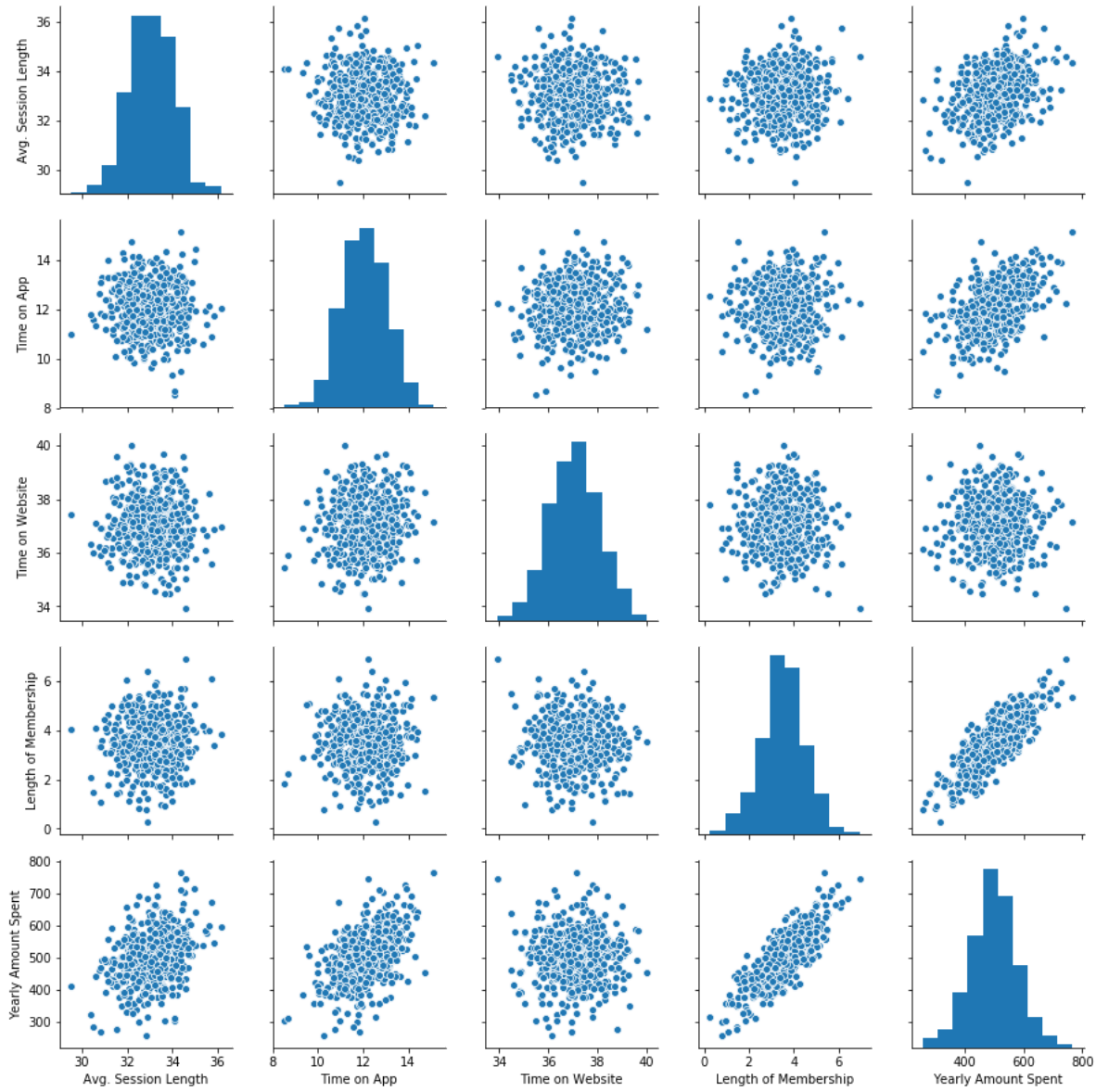


In [67]:

```
sns.pairplot(customers)
```

Out[67]:

<seaborn.axisgrid.PairGrid at 0x217b9fa5208>



from the above result we can see that the feature 'Length of Membership' is most correlated with the variable 'Yearly Amount Spent'

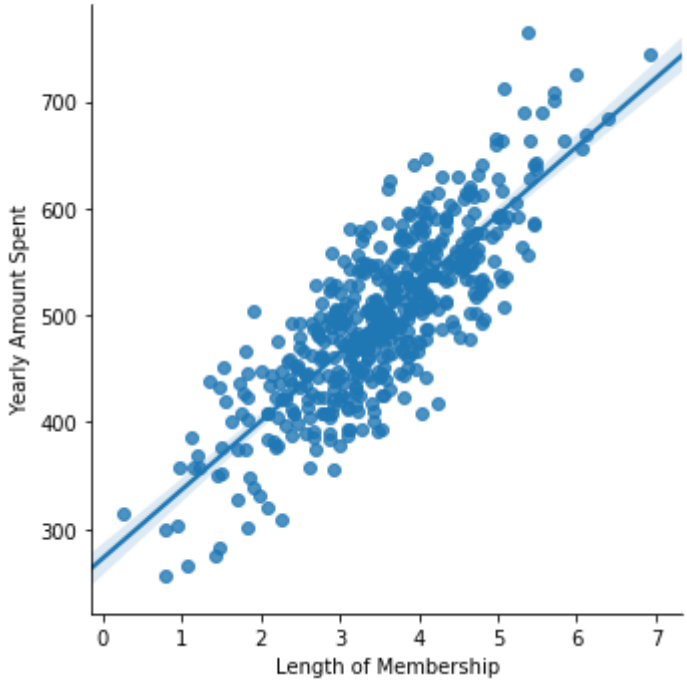
creating a linear model plot of Yearly Amount Spent vs the Length of Membership to visualize the relation in more detail

In [14]:

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

Out[14]:

<seaborn.axisgrid.FacetGrid at 0x217b7b8fd30>

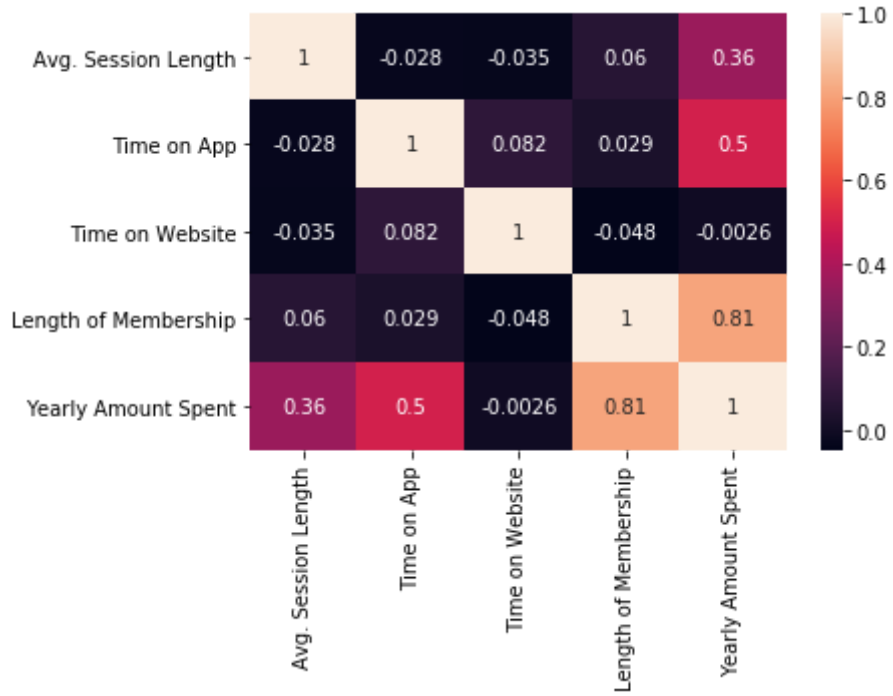


In [68]:

```
sns.heatmap(customers.corr(), annot= True)
```

Out[68]:

<matplotlib.axes._subplots.AxesSubplot at 0x217bb947860>



Splitting the dataset into training and testing data

In [69]:

```
customers.columns
```

Out[69]:

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

In [70]:

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

In [71]:

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

In [72]:

```
from sklearn.model_selection import train_test_split
```

In [73]:

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

Training the Model(on training data)

In [74]:

```
from sklearn.linear_model import LinearRegression
```

In [75]:

```
lm = LinearRegression()
```

In [76]:

```
lm.fit(X_train, y_train)
```

Out[76]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

After rhe training data is trained, the model can be used to make predictions

In [77]:

```
lm.coef_
```

Out[77]:

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

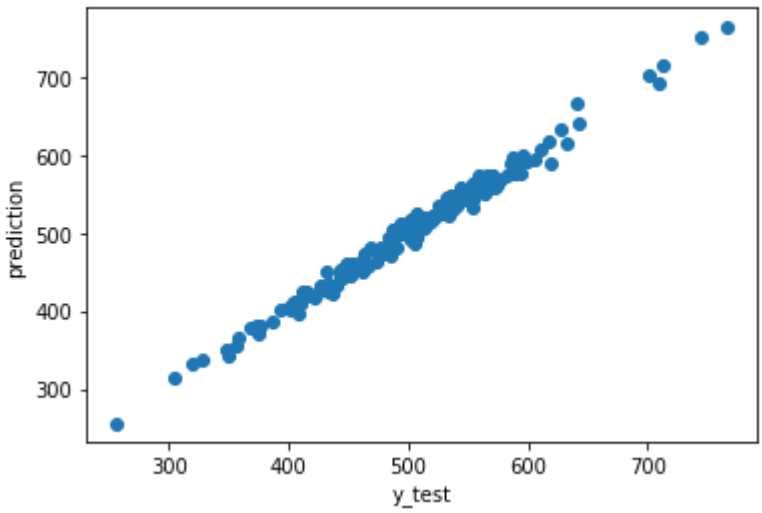
Predicting Test Data

In [78]:

```
prediction = lm.predict(X_test)
```

In [79]:

```
plt.scatter(y_test, prediction)
plt.xlabel('y_test')
plt.ylabel('prediction')
plt.show()
```



In [80]:

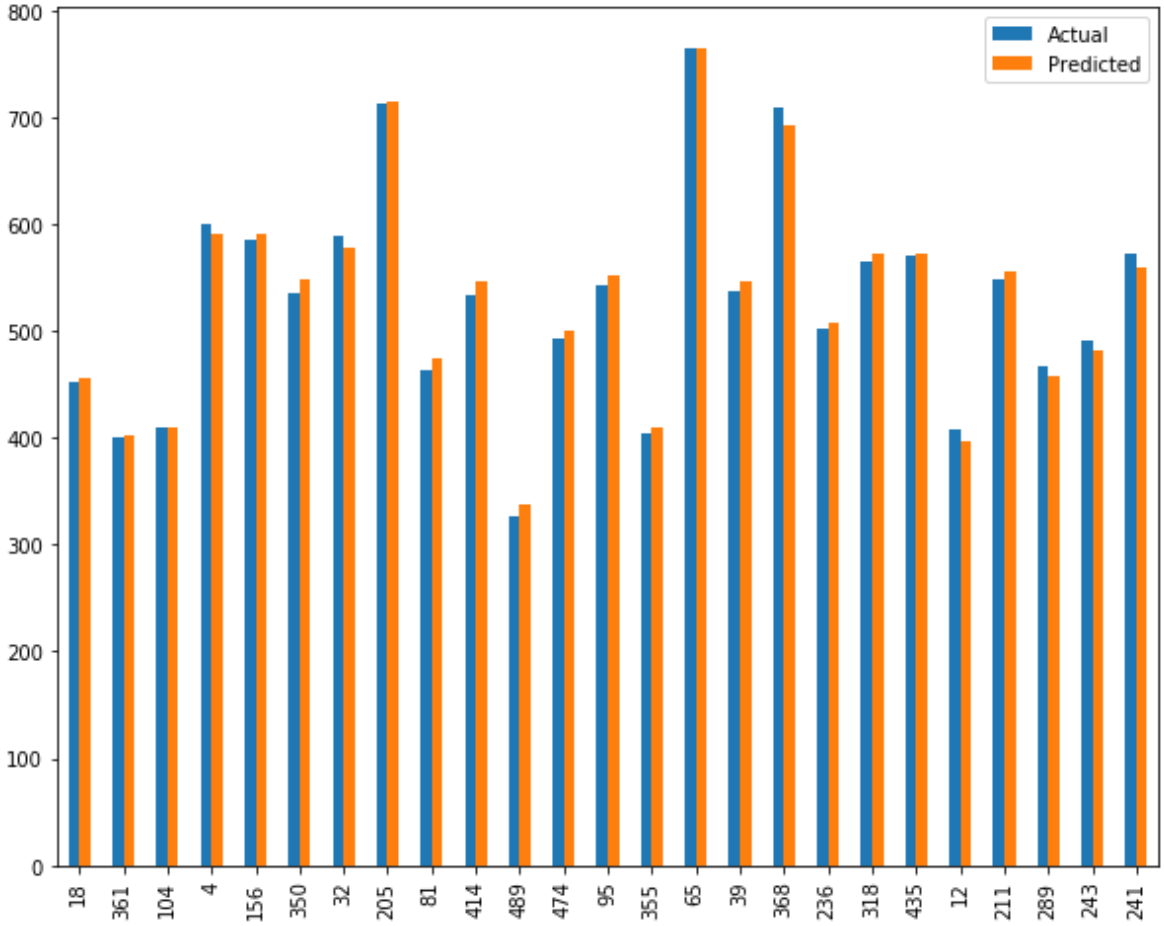
```
df = pd.DataFrame({'Actual': y_test, 'Predicted': prediction})
df1 = df.head(25)
df1
```

Out[80]:

	Actual	Predicted
18	452.315675	456.441861
361	401.033135	402.720053
104	410.069611	409.253154
4	599.406092	591.431034
156	586.155870	590.014373
350	535.480775	548.823966
32	588.712606	577.597380
205	712.396327	715.444281
81	462.897636	473.789345
414	532.724805	545.921136
489	327.377953	337.858031
474	492.556834	500.385067
95	543.340166	552.934780
355	403.766902	409.603896
65	765.518462	765.525908
39	537.846195	545.839737
368	708.935185	693.259691
236	501.928265	507.324162
318	564.790969	573.105332
435	571.216005	573.207663
12	408.640351	397.449897
211	548.518529	555.098511
289	467.427849	458.198681
243	490.600443	482.668999
241	571.471034	559.265596

In [81]:

```
df1.plot(kind='bar',figsize=(10,8))
plt.show()
```



Evaluating the Model

In [84]:

```
from sklearn import metrics
print('Mean Absolute Error', metrics.mean_absolute_error(y_test, prediction))
print('Mean Squared Error', metrics.mean_squared_error(y_test,prediction))
print('Root Mean Squared Error', np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Mean Absolute Error 7.228148653430838
Mean Squared Error 79.81305165097461
Root Mean Squared Error 8.933815066978642
```

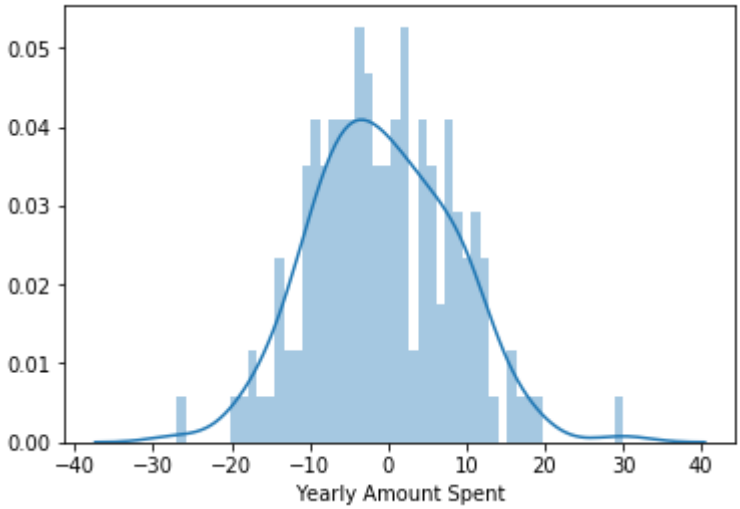
In [ ]:

```
```

In [93]:

```
sns.distplot((y_test-prediction), bins = 50)
plt.show()
```

C:\Users\DELL\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval



Conclusions

In [95]:

```
coefficients = pd.DataFrame(lm.coef_,X.columns)
coefficients.columns = ['Coefficients']
coefficients
```

Out[95]:

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

~

Interpreting the 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.

In [ ]: