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

In [2]:
```
df= pd.read_csv('Ecommerce Customers')
```

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

Out[3]:

	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 [4]:
```
df.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
```

In [5]:
```
df.shape
```

Out[5]:
```
(500, 8)
```

In [6]:
```
df.describe()
```

Out[6]:

	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 [7]:
```
df.isnull().sum()
```

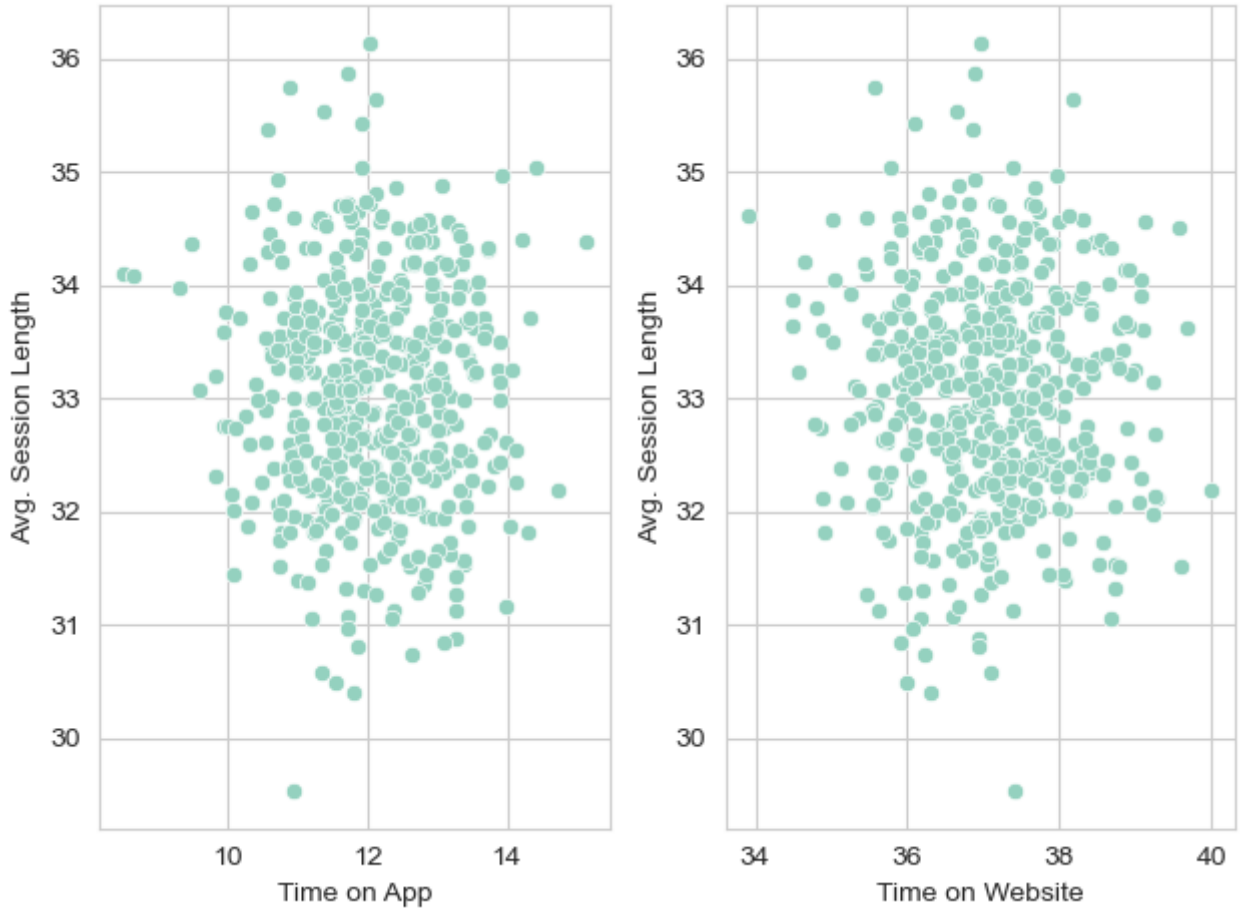
Out[7]:
```
Email                 0
Address               0
Avatar                0
Avg. Session Length   0
Time on App           0
Time on Website       0
Length of Membership  0
Yearly Amount Spent   0
dtype: int64
```

There is no null values in our data

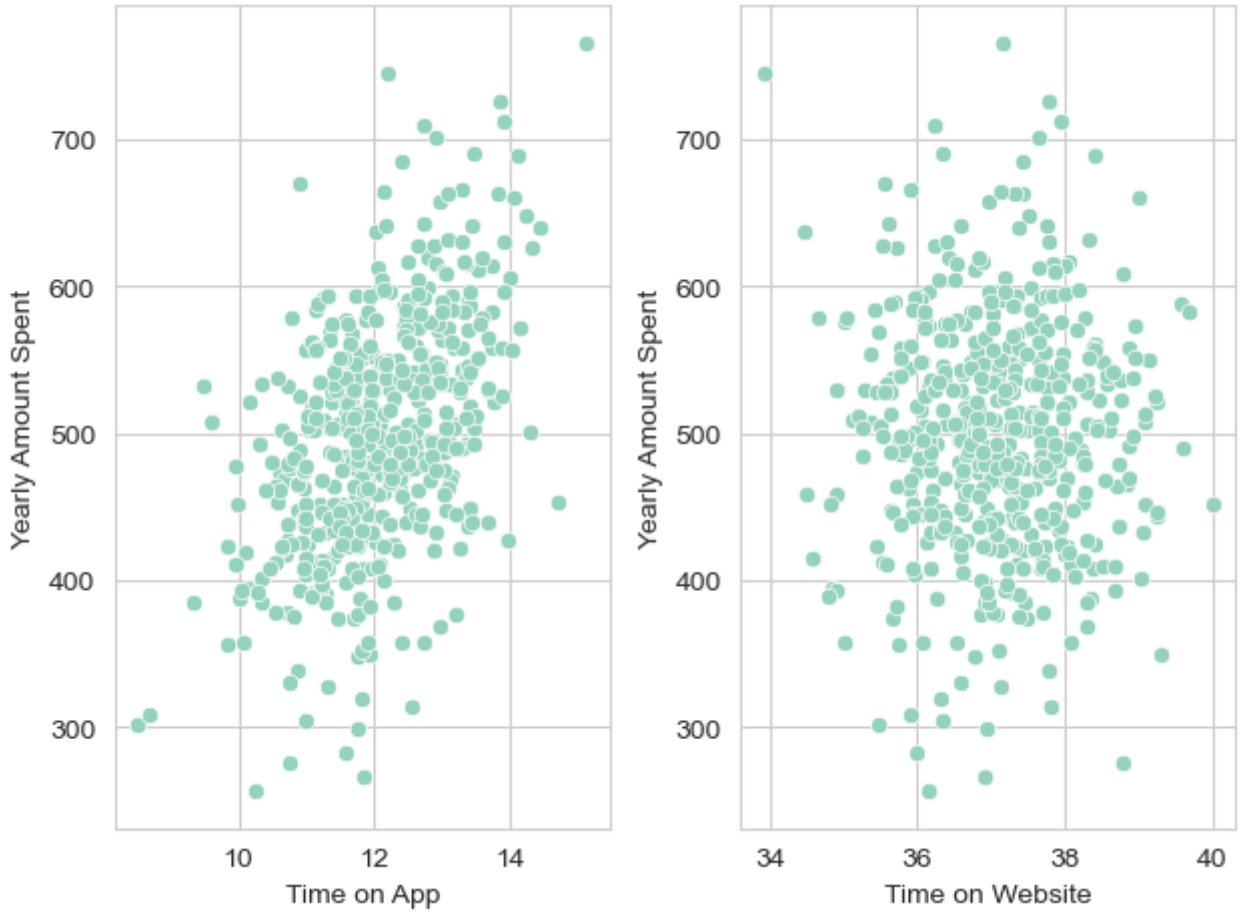
EDA on the data

In [8]:
```
sns.set_palette("GnBu_d")
sns.set_style('whitegrid')
```

In [9]:
```
fig,axes=plt.subplots(nrows=1,ncols=2)
sns.scatterplot(x='Time on App',y='Avg. Session Length',data=df, ax=axes[0])
sns.scatterplot(x='Time on Website',y='Avg. Session Length',data=df, ax=axes[1])
plt.tight_layout()
```



In [10]:
```
fig,axes=plt.subplots(nrows=1,ncols=2)
sns.scatterplot(x='Time on App', y='Yearly Amount Spent', data=df, ax=axes[0])
sns.scatterplot(x='Time on Website',y='Yearly Amount Spent',data=df, ax=axes[1])
#axes[0].set_xlim(0, 40)
plt.tight_layout()
```

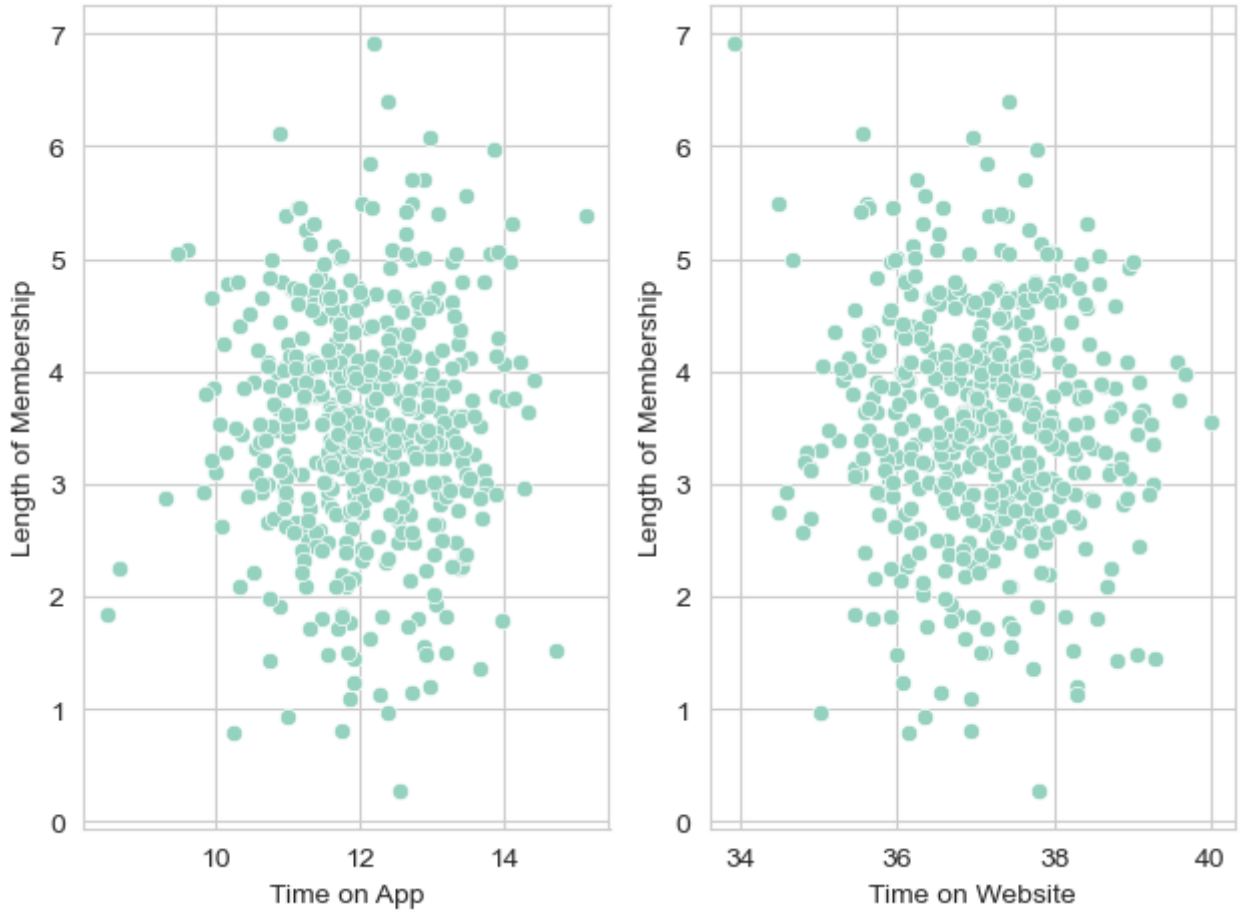

Yearly Amount Spent
700
600
500
400
300
10
12
14
Time on App
Yearly Amount Spent
700
600
500
400
300
34
36
38
40
Time on Website

In [11]:
```
df.head()
```

Out[11]:

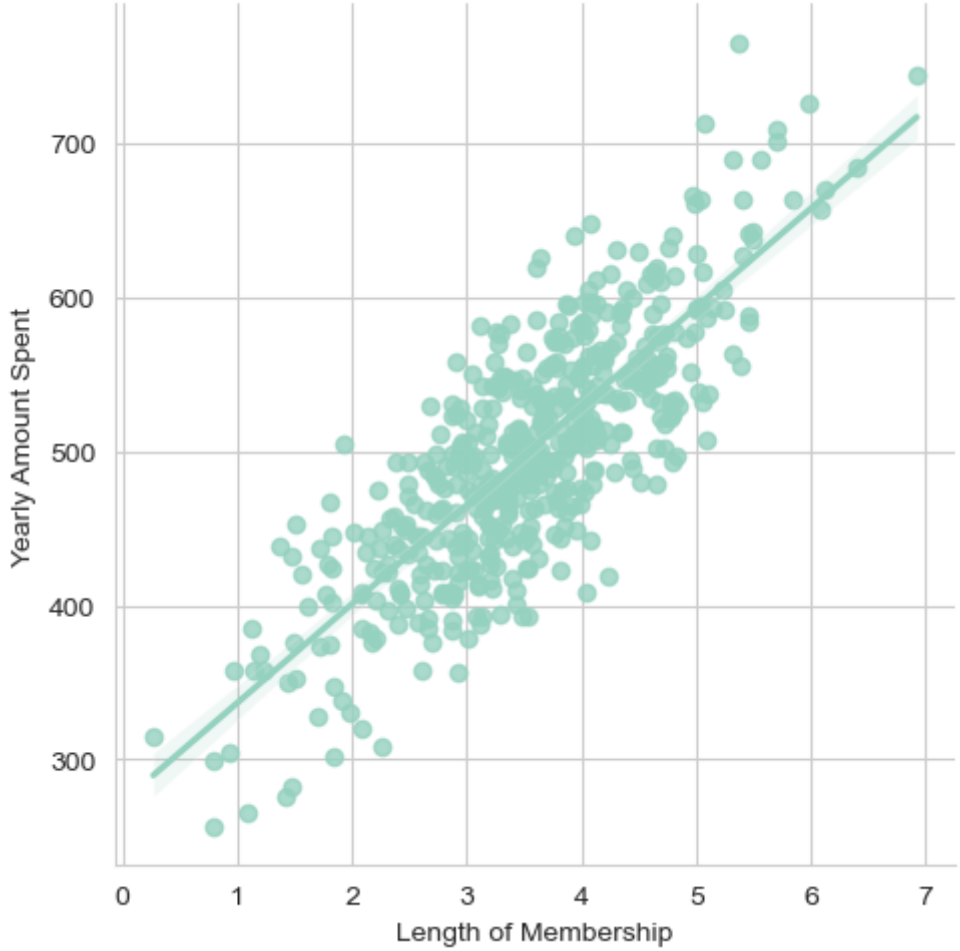
	Email	Address	Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651	39.577668	4.082621	587.951054
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	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 [12]:
```
fig,axes=plt.subplots(nrows=1,ncols=2)

sns.scatterplot(x='Time on App',y='Length of Membership',data=df,ax=axes[0])

sns.scatterplot(x='Time on Website',y='Length of Membership',data=df,ax=axes[1])
plt.tight_layout()
```



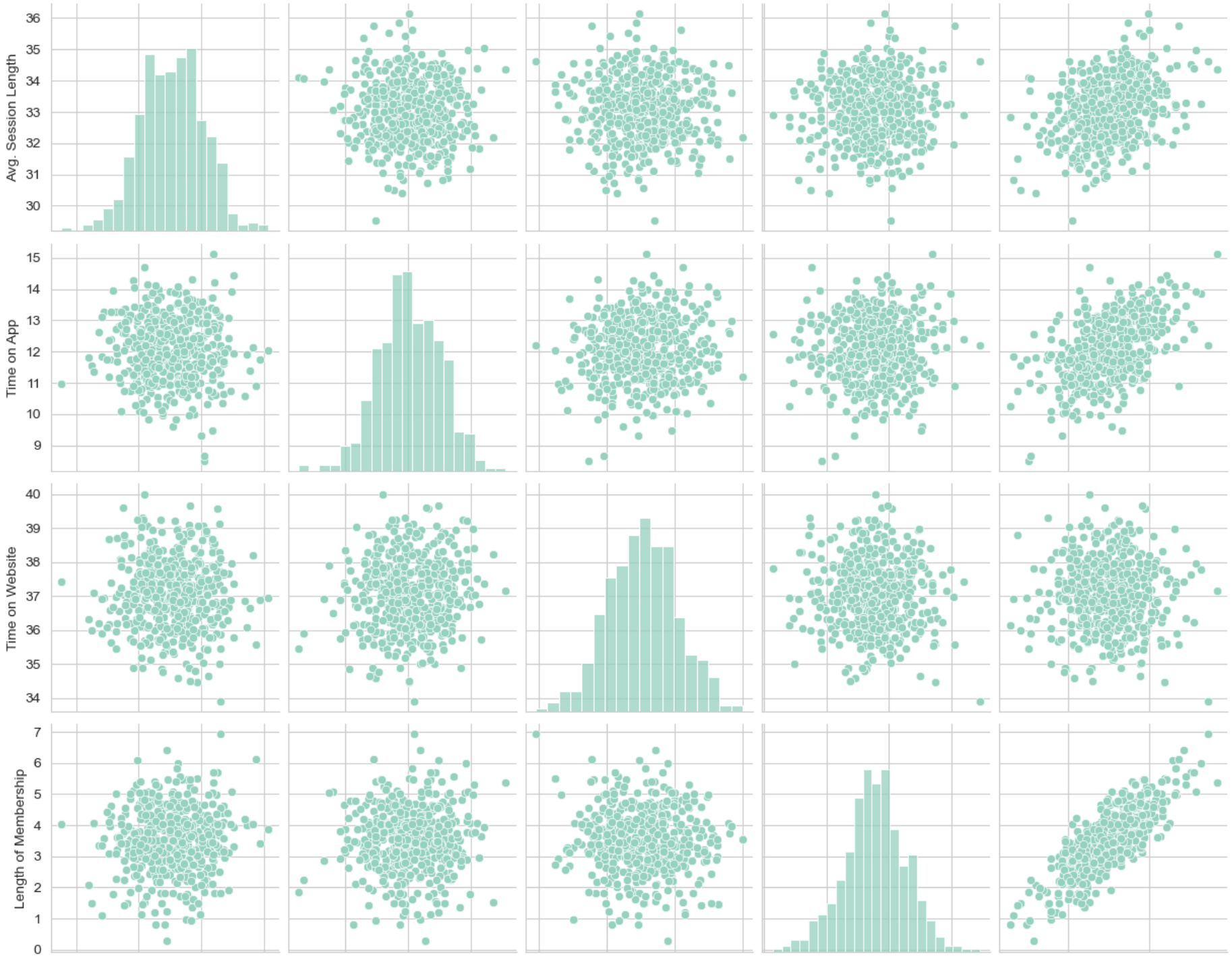
In [13]:
```
sns.lmplot(x='Length of Membership',y='Yearly Amount Spent',data=df)
```

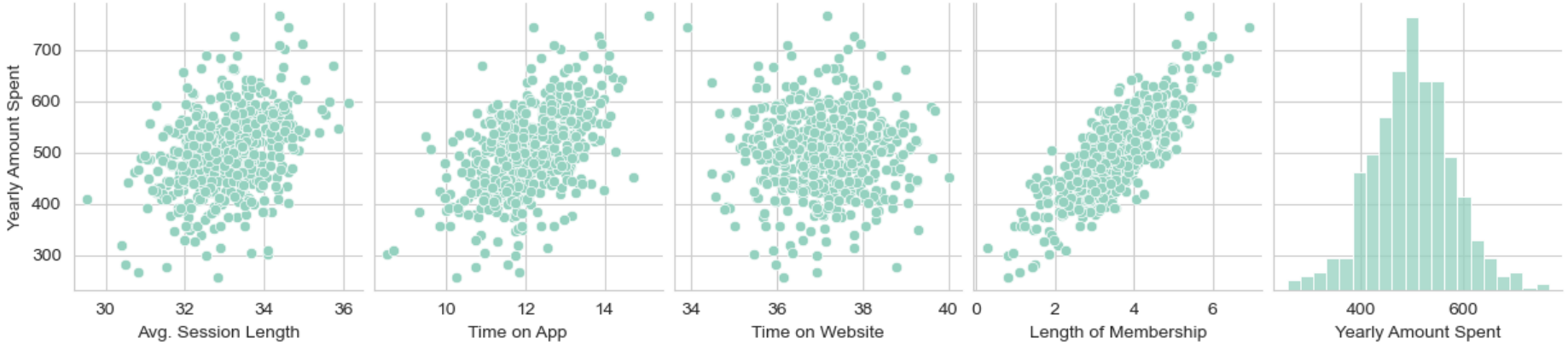
Out[13]:
```
<seaborn.axisgrid.FacetGrid at 0x2eb4688bf10>
```



In [14]:
```
sns.pairplot(df)
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x2eb468a5fd0>
```


Avg. Session Length
Time on App
Time on Website
Length of Membership



In [15]:
```
df.corr()
```

```
C:\Users\ravix\AppData\Local\Temp\ipykernel_15604\1134722465.py:1: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_o
nly to silence this warning.
  df.corr()
```

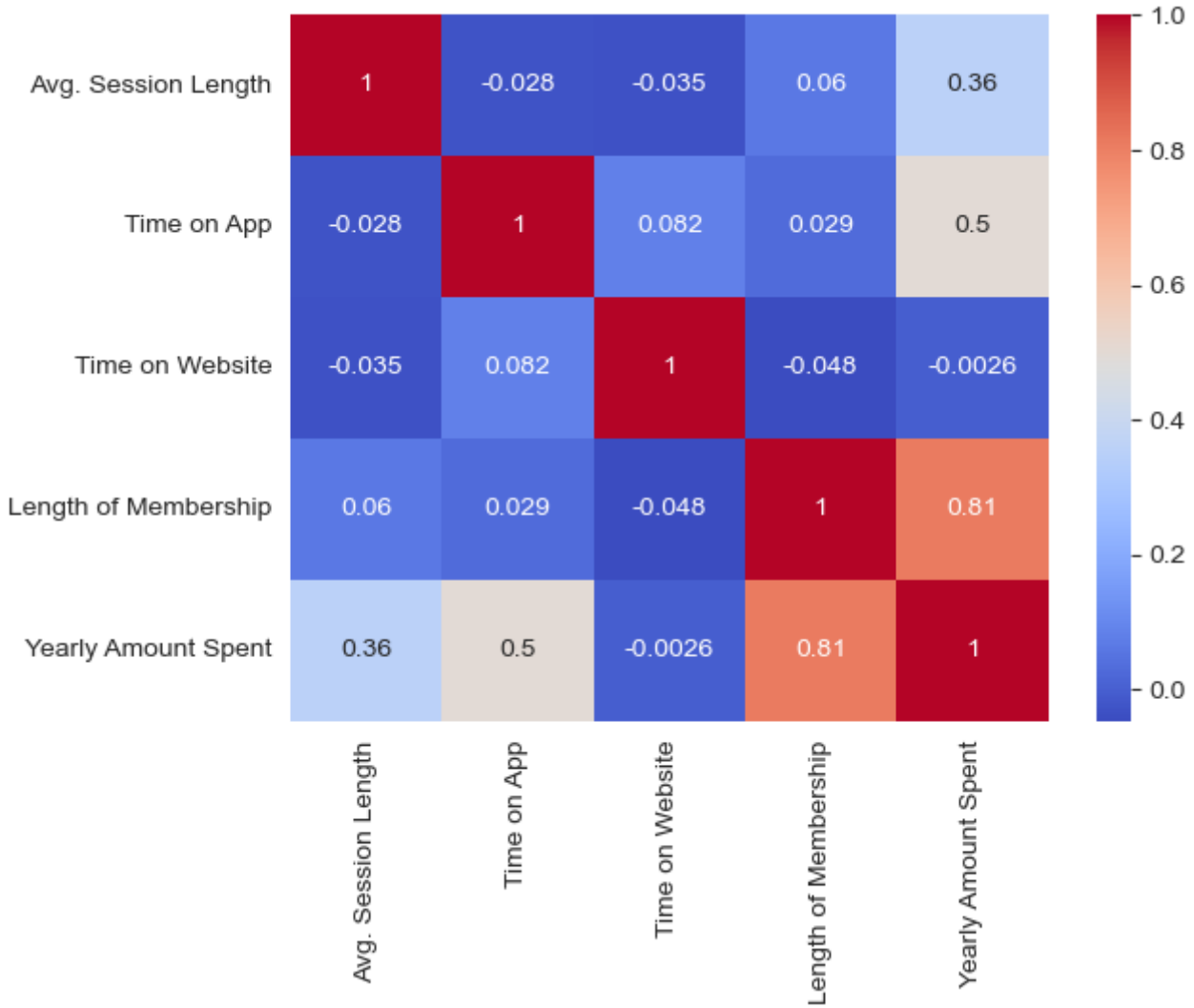
Out[15]:

	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 [16]:
```
sns.heatmap(df.corr(),annot=True, cmap='coolwarm')
```

```
C:\Users\ravix\AppData\Local\Temp\ipykernel_15604\3692148766.py:1: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_o
nly to silence this warning.
  sns.heatmap(df.corr(),annot=True, cmap='coolwarm')
```

Out[16]:
```
<Axes: >
```



Preparing data for our Model

In [17]:
```
df.columns
```

Out[17]:
```
Index(['Email', 'Address', 'Avatar', 'Avg. Session Length', 'Time on App',
       'Time on Website', 'Length of Membership', 'Yearly Amount Spent'],
      dtype='object')
```

In [18]:
```
X= df[['Avg. Session Length', 'Time on App',
       'Time on Website', 'Length of Membership']]
X.head()
```

Out[18]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership
0	34.497268	12.655651	39.577668	4.082621
1	31.926272	11.109461	37.268959	2.664034
2	33.000915	11.330278	37.110597	4.104543
3	34.305557	13.717514	36.721283	3.120179
4	33.330673	12.795189	37.536653	4.446308

In [19]:
```
Y =df['Yearly Amount Spent']
Y.head()
```

Out[19]:
```
0    587.951054
1    392.204933
2    487.547505
3    581.852344
4    599.406092
Name: Yearly Amount Spent, dtype: float64
```

Imprting libraries

In [20]:
```
from sklearn.model_selection import train_test_split
```

In [21]:
```
X_train, X_test, Y_train, Y_test= train_test_split(X, Y, test_size=0.3, random_state=101)
```

Training the Model

In [22]:
```
from sklearn.linear_model import LinearRegression
```

In [23]:
```
lm= LinearRegression()
```

Train the model

In [24]:
```
lm.fit(X_train,Y_train)
```

Out[24]:
```
LinearRegression()
```

Checking the intercept

basically The linear equation used in simple linear regression is of the form

y=mx+b

Where:

y is the dependent variable (target).

x is the independent variable (feature).

m is the coefficient (slope) that the regression model learns.

b is the intercept term.

In [25]:
```
print(lm.intercept_)
```

```
-1047.932782250239
```

Check the co-efficient

In [26]:
```
lm.coef_
```

Out[26]:
```
array([25.98154972, 38.59015875,  0.19040528, 61.27909654])
```

Creating a dataframe for the Co-efficient

In [27]:
```
cdf = pd.DataFrame(lm.coef_,index=X.columns,columns=['coeff'])
```

In [28]:
```
cdf
```

Out[28]:

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

In [ ]:
```

```

Predictions

In [29]:
```
predictions= lm.predict(X_test)
```

In [30]:
```
pred = pd.DataFrame(predictions, columns=[' Predicted Amount '])
```

In [31]:
```
pred.head()
```

Out[31]:

	Predicted Amount
0	456.441861
1	402.720053
2	409.253154
3	591.431034
4	590.014373

Comparing the prediction with Y_test

In [32]:
```
Y_test_reset = Y_test.reset_index(drop=True)
```

In [33]:
```
Y_test_reset.head()
```

Out[33]:
```
0    452.315675
1    401.033135
2    410.069611
3    599.406092
4    586.155870
Name: Yearly Amount Spent, dtype: float64
```

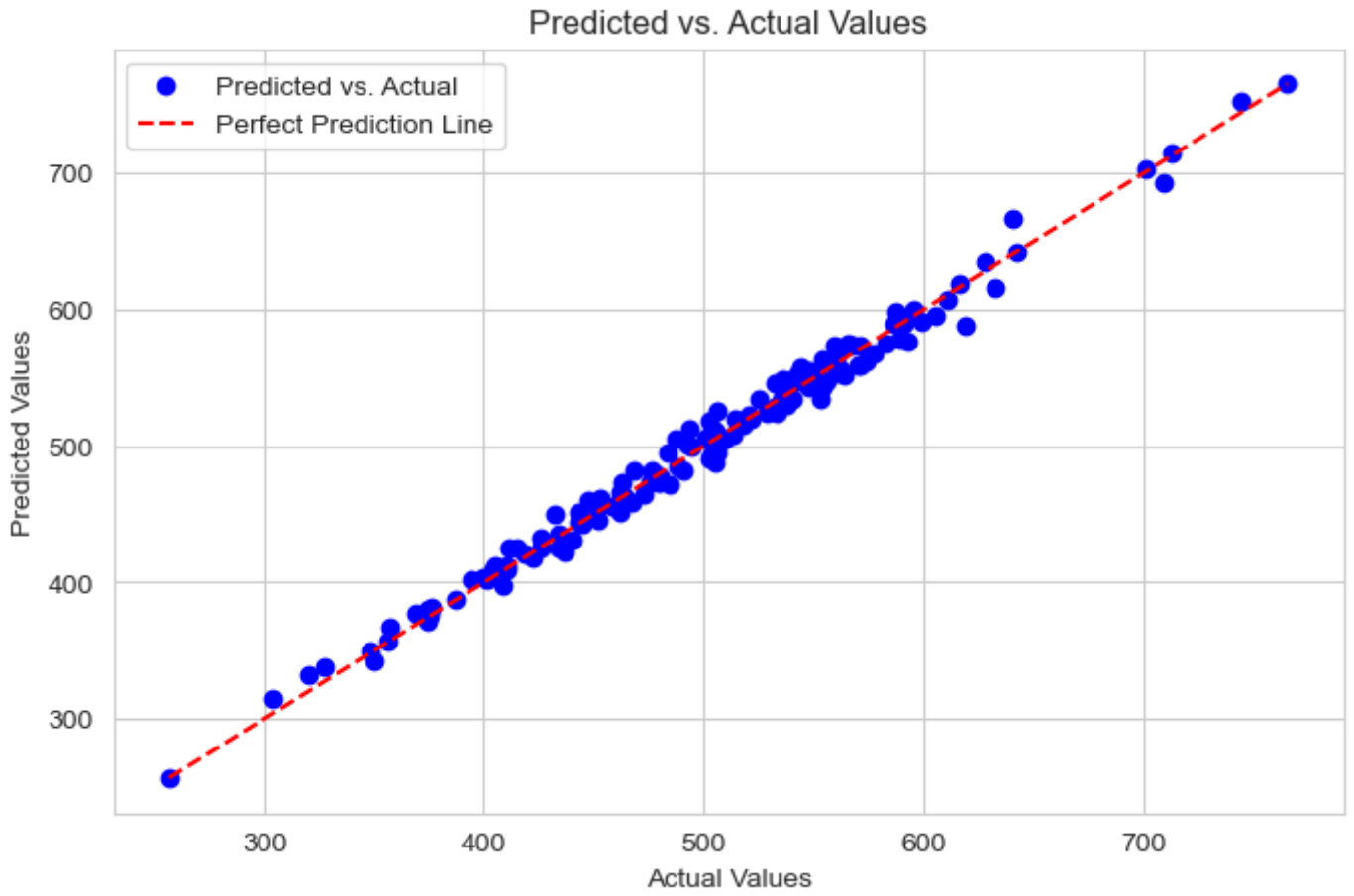
In [34]:
```
result = pd.concat([Y_test_reset, pred], axis=1)
print(result)
```

```
     Yearly Amount Spent   Predicted Amount
0              452.315675          456.441861
1              401.033135          402.720053
2              410.069611          409.253154
3              599.406092          591.431034
4              586.155870          590.014373
..                    ...                 ...
145            479.731938          478.300766
146            488.387526          484.410296
147            461.112248          457.590999
148            407.704548          411.526576
149            375.398455          375.479006

[150 rows x 2 columns]
```

In [35]:
```
result = pd.concat([Y_test_reset, pred], axis=1)
print(result.head())
```

```
   Yearly Amount Spent   Predicted Amount
0           452.315675          456.441861
1           401.033135          402.720053
2           410.069611          409.253154
3           599.406092          591.431034
4           586.155870          590.014373
```

In [36]:
```
plt.figure(figsize=(8, 5))  # Adjust the figure size as needed

plt.scatter(Y_test, pred, color='blue', label='Predicted vs. Actual')
plt.plot([min(Y_test), max(Y_test)], [min(Y_test), max(Y_test)], linestyle='--', color='red', label='Perfect Prediction Line')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Predicted vs. Actual Values')
plt.legend()
plt.show()
```



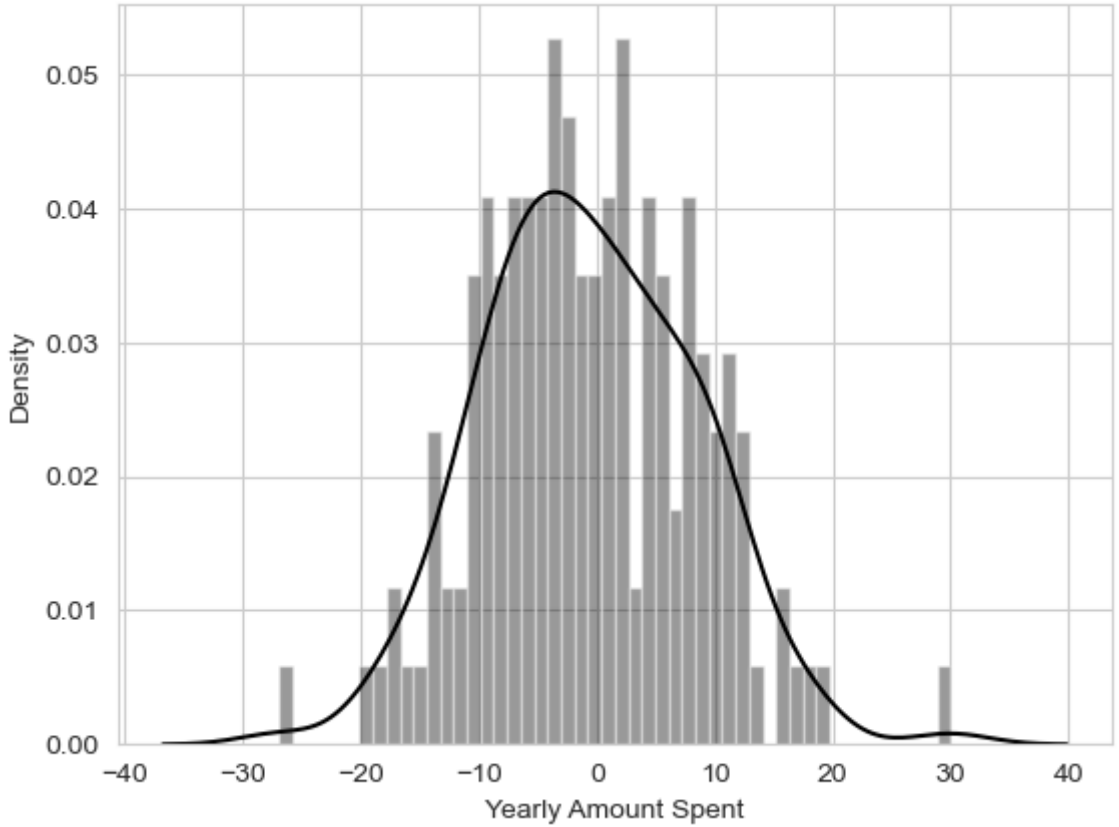
In [37]:
```
# calculating the metrics
from sklearn import metrics

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: 7.228148653430832
MSE: 79.81305165097456
RMSE: 8.93381506697864
```

Plotting Residual

In [38]:
```
sns.distplot((Y_test-predictions),bins=50,color='black');
```

```
C:\Users\ravix\AppData\Local\Temp\ipykernel_15604\3984256040.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot((Y_test-predictions),bins=50,color='black');
```



Drawing conclusions from Co-efficient

In [39]:
```
cdf
```

Out[39]:

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

Conclusion

For 1 unit increase in Avg. Session Length, it is associated with an increase of 25.98 total dollars spent.

For 1 unit increase in Time on App, it is associated with an increase of 25.98 total dollars spent.

For 1 unit increase in Time on Website, it is associated with an increase of 25.98 total dollars spent.

For 1 unit increase in Length of Membership, it is associated with an increase of 25.98 total dollars spent.

In [ ]:

In [ ]: