Aim for how to maximize the Yearly Amount spent (App / Website) by the customers by using Linear regression

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

In [4]:
```
df=pd.read_csv("Ecommerce Customers.csv")
df.head()
```

Out[4]:

	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 [5]:
```
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
```

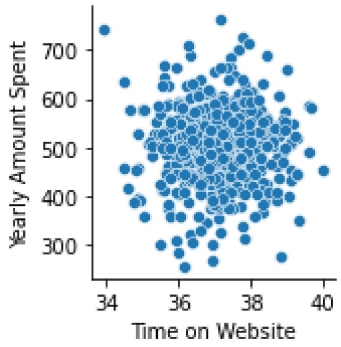
In [7]:
```
df.describe()
```

Out[7]:

	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

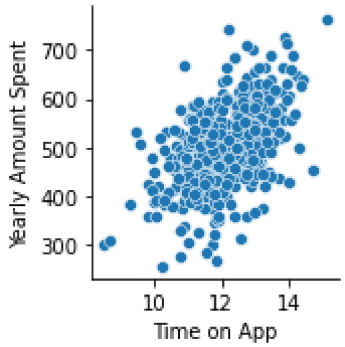
In [9]:
```
sns.pairplot(df, x_vars="Time on Website", y_vars="Yearly Amount Spent", kind="scatter")
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1edb5a40c40>
```



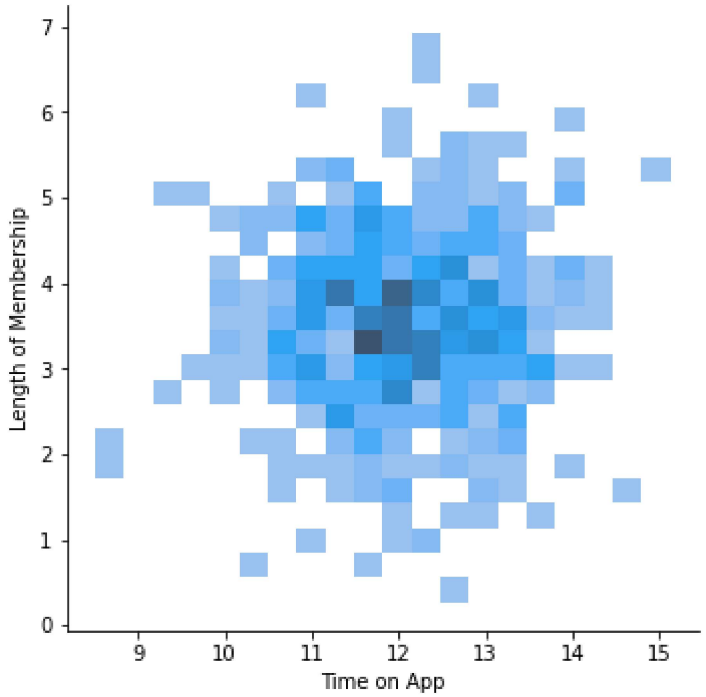
In [10]:
```
sns.pairplot(df, x_vars="Time on App", y_vars="Yearly Amount Spent", kind="scatter")
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x1edb61811f0>
```



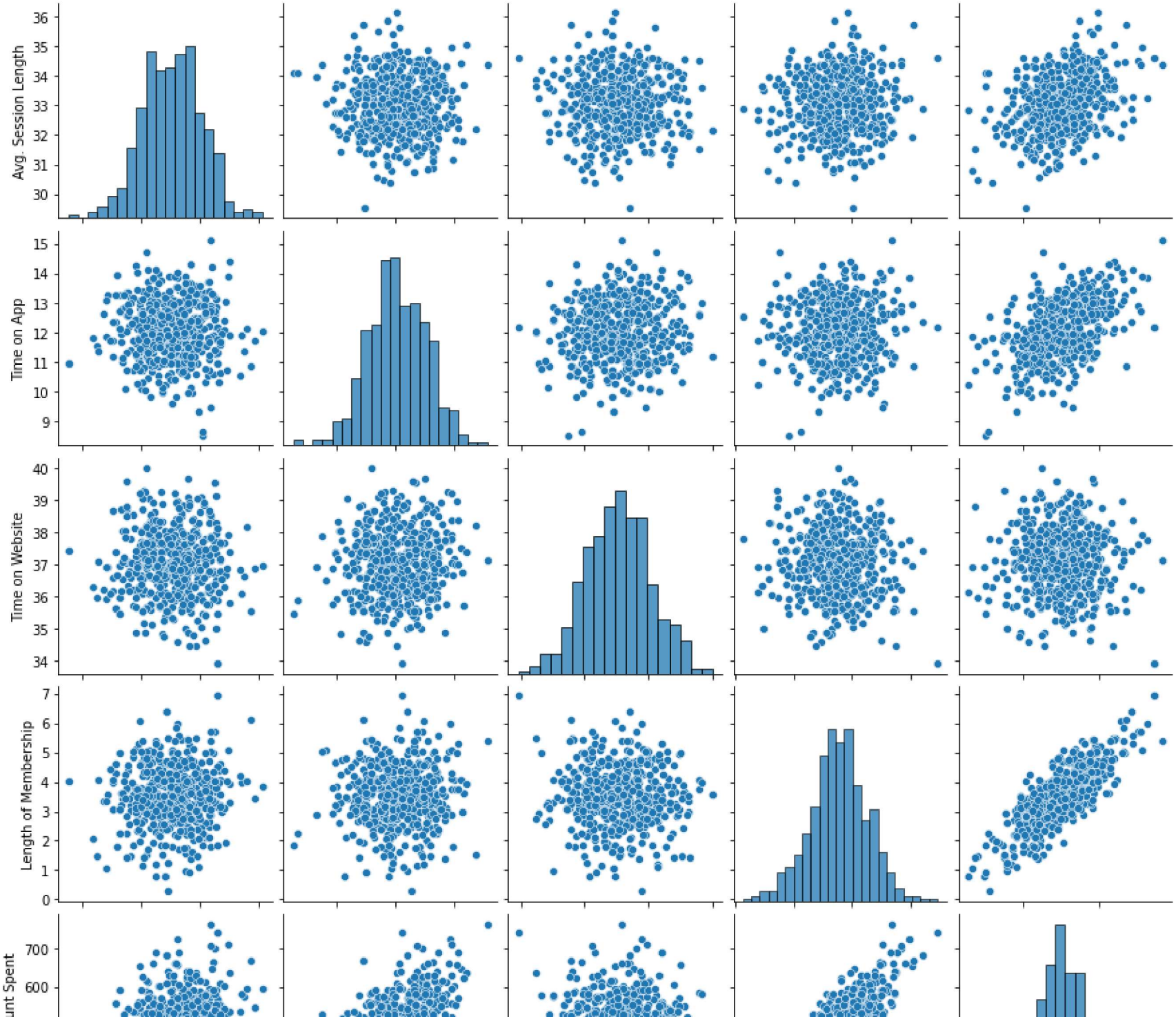
In [20]:
```
sns.pairplot(df, x_vars="Time on App", y_vars="Length of Membership", kind="hist", height=5)
```

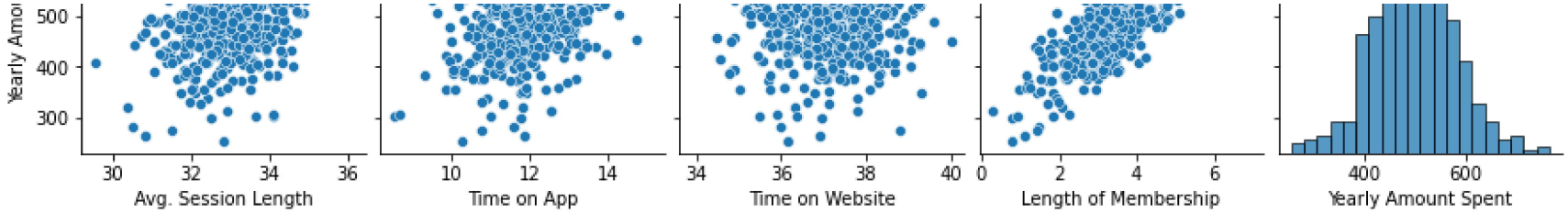
Out[20]:
```
<seaborn.axisgrid.PairGrid at 0x1edb76c1220>
```



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

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x1edb63141c0>
```


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



From the pairplot above we can observe that the length of membership is most closely related to the yearly amount spent by the consumers.

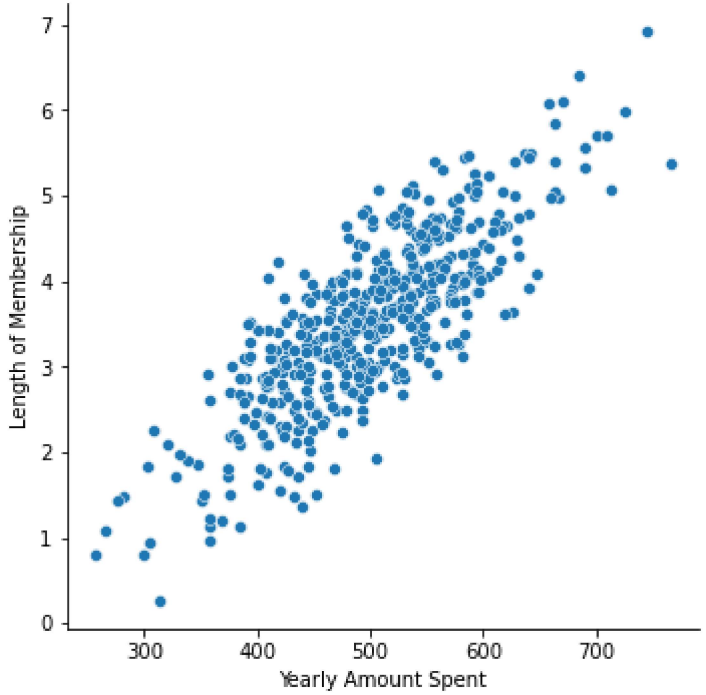
In [16]:
```
df.corr().style.background_gradient()
```

Out[16]:

	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 [21]:
```
sns.pairplot(df, x_vars="Yearly Amount Spent", y_vars="Length of Membership", kind="scatter", height=5)
```

Out[21]:
```
<seaborn.axisgrid.PairGrid at 0x1edb7617370>
```


Length of Membership
Yearly Amount Spent
0
1
2
3
4
5
6
7
300
400
500
600
700

In [22]:
```
df.head()
```

Out[22]:

	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 [26]:
```
x = df[['Avg. Session Length','Time on App','Time on Website','Length of Membership']]
y = df['Yearly Amount Spent']
```

In [28]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.3, random_state=101)
```

In [29]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(xtrain, ytrain)
ypred = linreg.predict(xtest)
```

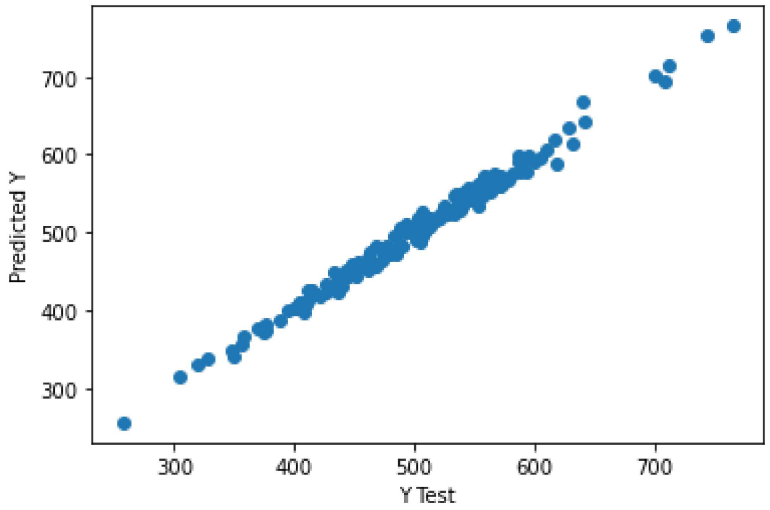
In [30]:
```
linreg.coef_
```

Out[30]:
```
array([25.98154972, 38.59015875,  0.19040528, 61.27909654])
```

In [32]:
```
np.array(ypred)
```

Out[32]:
```
array([456.44186104, 402.72005312, 409.2531539 , 591.4310343 ,
       590.01437275, 548.82396607, 577.59737969, 715.44428115,
       473.7893446 , 545.9211364 , 337.8580314 , 500.38506697,
       552.93478041, 409.6038964 , 765.52590754, 545.83973731,
       693.25969124, 507.32416226, 573.10533175, 573.2076631 ,
       397.44989709, 555.0985107 , 458.19868141, 482.66899911,
       559.2655959 , 413.00946082, 532.25727408, 377.65464817,
       535.0209653 , 447.80070905, 595.54339577, 667.14347072,
       511.96042791, 573.30433971, 505.02260887, 565.30254655,
       460.38785393, 449.74727868, 422.87193429, 456.55615271,
       598.10493696, 449.64517443, 615.34948995, 511.88078685,
       504.37568058, 515.95249276, 568.64597718, 551.61444684,
       356.5552241 , 464.9759817 , 481.66007708, 534.2220025 ,
       256.28674001, 505.30810714, 520.01844434, 315.0298707 ,
       501.98080155, 387.03842642, 472.97419543, 432.8704675 ,
       539.79082198, 590.03070739, 752.86997652, 558.27858232,
       523.71988382, 431.77690078, 425.38411902, 518.75571466,
       641.9667215 , 481.84855126, 549.69830187, 380.93738919,
       555.18178277, 403.43054276, 472.52458887, 501.82927633,
       473.5561656 , 456.76720365, 554.74980563, 702.96835044,
       534.68884588, 619.18843136, 500.11974127, 559.43899225,
       574.8730604 , 505.09183544, 529.9537559 , 479.20749452,
       424.78407899, 452.20986599, 525.74178343, 556.60674724,
       425.7142882 , 588.8473985 , 490.77053065, 562.56866231,
       495.75782933, 445.17937217, 456.64011682, 537.98437395,
       367.06451757, 421.12767301, 551.59651363, 528.26019754,
       493.47639211, 495.28105313, 519.81827269, 461.15666582,
       528.8711677 , 442.89818166, 543.20201646, 350.07871481,
       401.49148567, 606.87291134, 577.04816561, 524.50431281,
       554.11225704, 507.93347015, 505.35674292, 371.65146821,
       342.37232987, 634.43998975, 523.46931378, 532.7831345 ,
       574.59948331, 435.57455636, 599.92586678, 487.24017405,
       457.66383406, 425.25959495, 331.81731213, 443.70458331,
       563.47279005, 466.14764208, 463.51837671, 381.29445432,
       411.88795623, 473.48087683, 573.31745784, 417.55430913,
       543.50149858, 547.81091537, 547.62977348, 450.99057409,
       561.50896321, 478.30076589, 484.41029555, 457.59099941,
       411.52657592, 375.47900638])
```

In [35]:
```
plt.scatter(ytest,ypred)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.show()
```

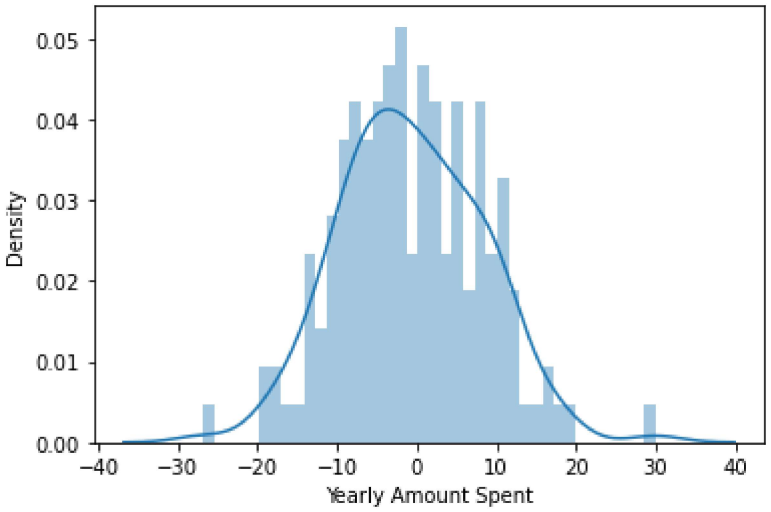


In [37]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
mae = mean_absolute_error(ytest, ypred)
mse = mean_squared_error(ytest, ypred)
rmse = np.sqrt(mse)


print(f"MAE : {mae}\nMSE : {mse}\nRMSE : {rmse}")
```

```
MAE : 7.228148653430832
MSE : 79.81305165097444
RMSE : 8.933815066978633
```

In [49]:
```
residual = ytest - ypred
sns.distplot((residual),bins=40);
```



In [52]:
```
pd.DataFrame(linreg.coef_, index=x.columns, columns=["Coefficients"])
```

Out[52]:

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

For every unit growth in "Avg. Session Length" , we estimate that "yearly amount spent" will grow by 26. Similarly increasing "Time on App" by one unit increases yearly amount spent by 38.5 whereas, increasing Time on Website by one unit increases yearly amount spent by only 0.6 approximately. For every unit growth in "Length of Membership" , we estimate that "yearly amount spent" will grow by 61.

Using the predictions above we can develop ways in order to increase yearly amount spent by the customers. We can improve the app experience so that the users spend more time on the app or we can also focus on the website and develop it so that it becomes as efficient as the app or we can focus on customer relationship so that people remain customers for long periods of time.