```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv("ecommerce.txt")
```

```
df.head()
```

| | 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 |

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
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
```

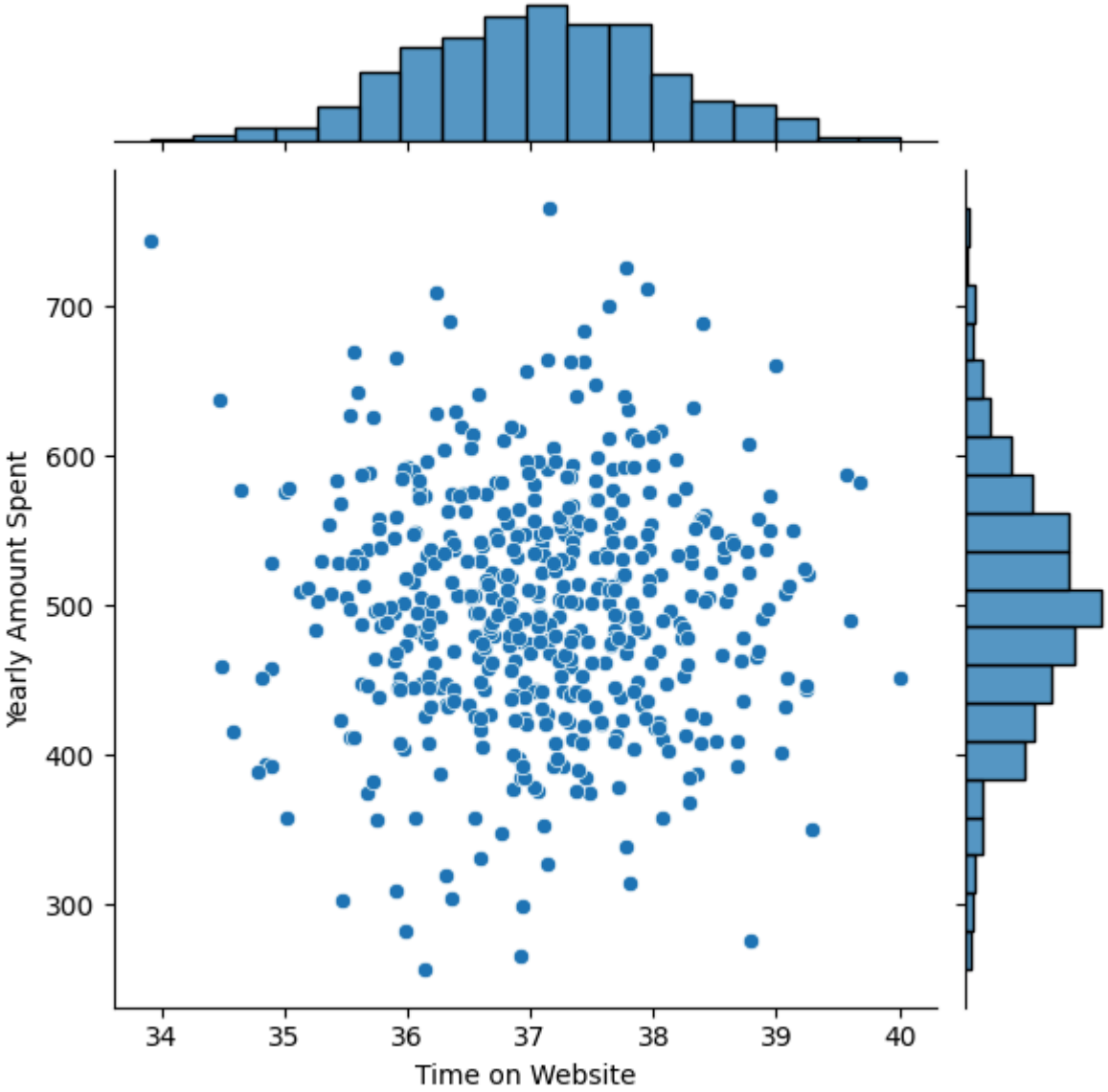
```
df.describe()
```

| | 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

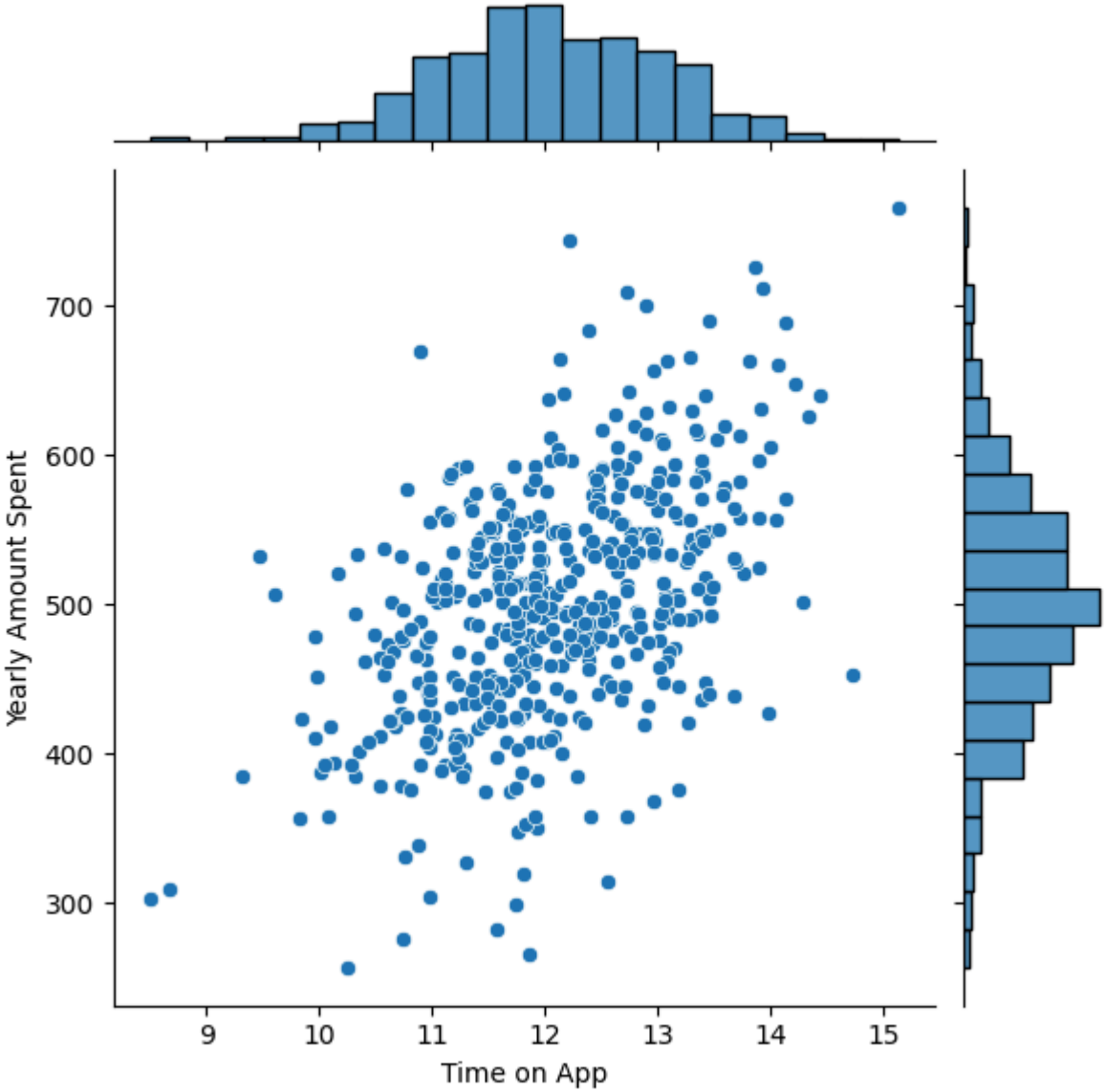
```
sns.jointplot(x='Time on Website',y='Yearly Amount Spent',data=df)
```

<seaborn.axisgrid.JointGrid at 0x7d28bdd30dc0>



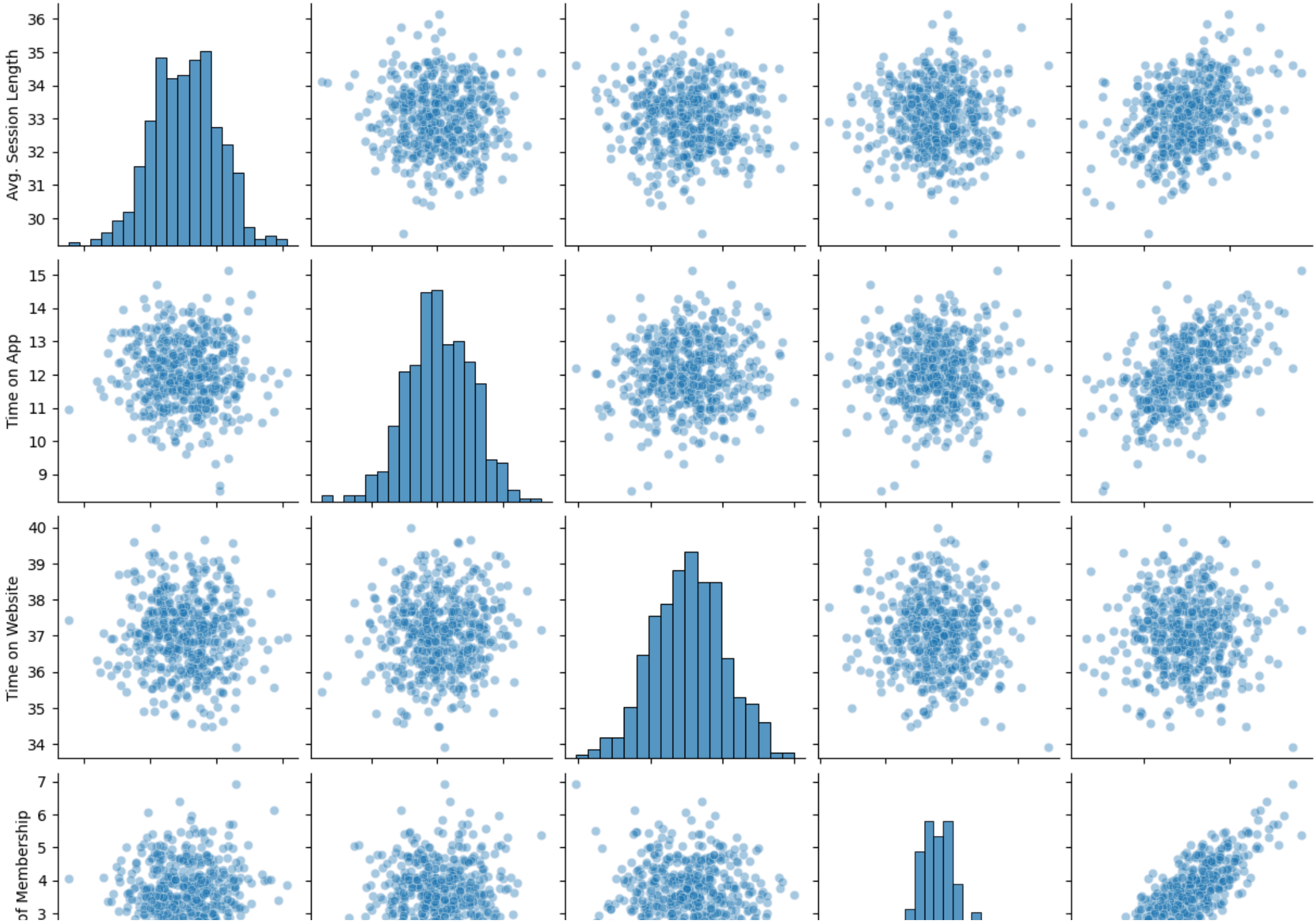

```
sns.jointplot(x='Time on App',y='Yearly Amount Spent',data=df)
```

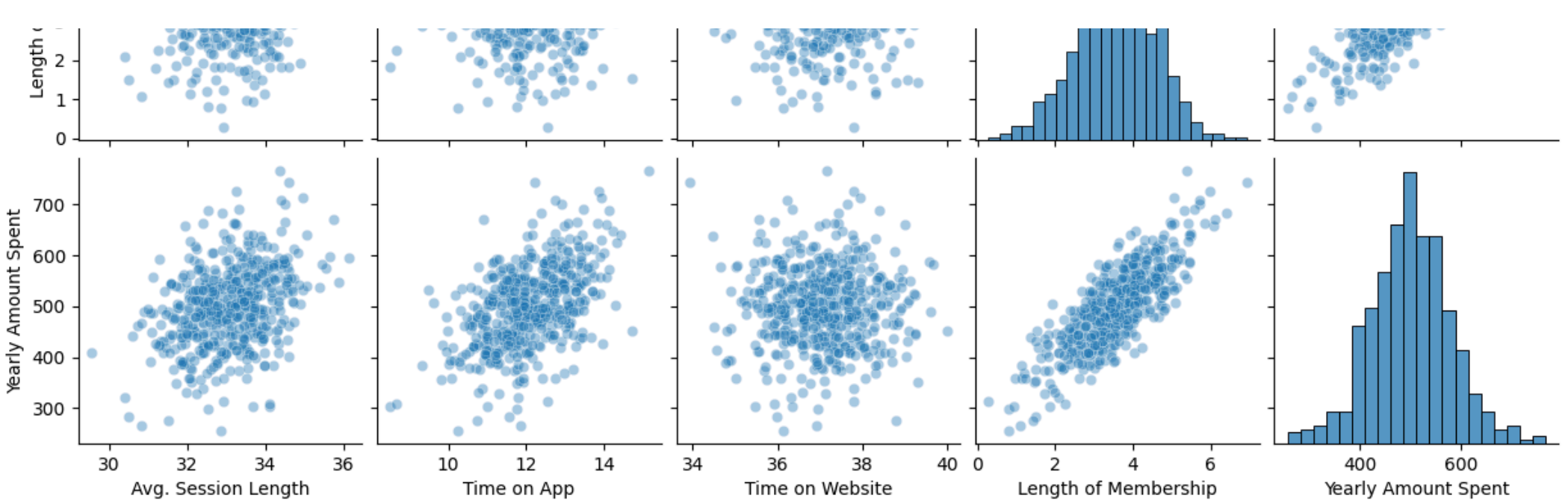
<seaborn.axisgrid.JointGrid at 0x7d28bdef0040>




```
sns.pairplot(df,kind='scatter',plot_kws={'alpha':0.4})
```

<seaborn.axisgrid.PairGrid at 0x7d28bdd313f0>

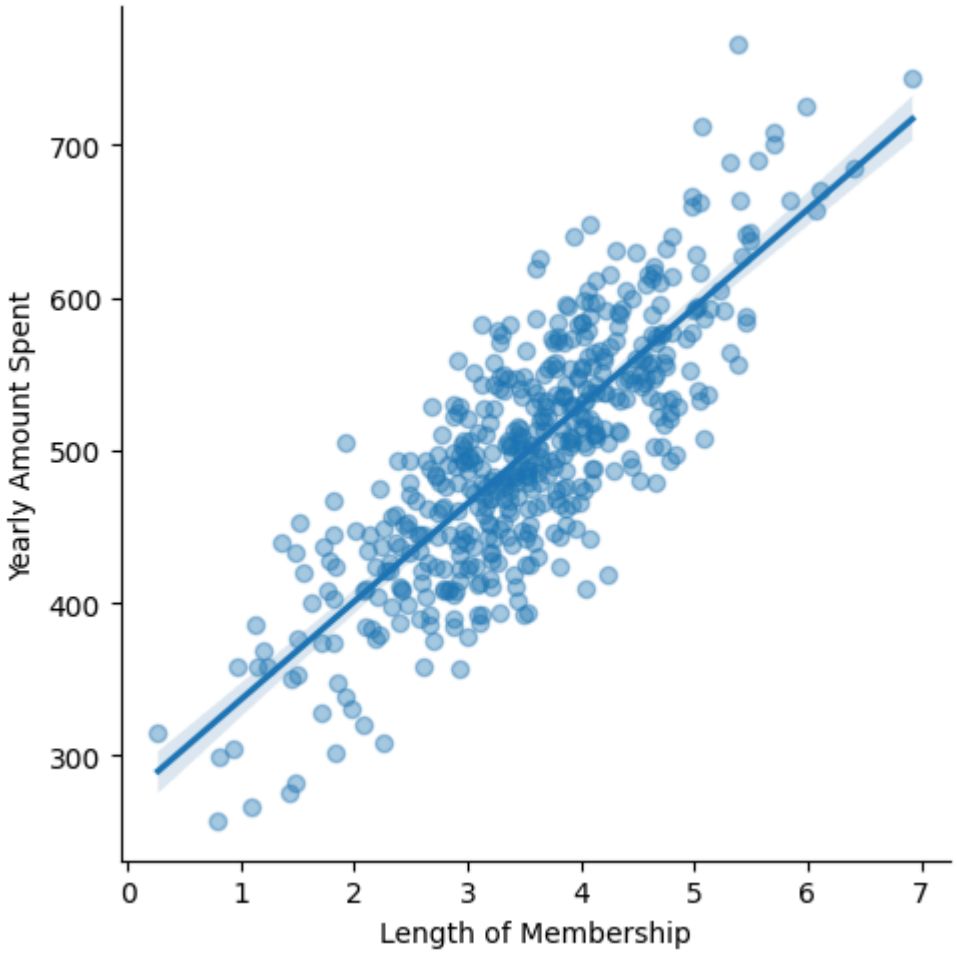





quick linear regression

```
sns.lmplot(x='Length of Membership',y='Yearly Amount Spent',data=df,scatter_kws={'alpha':0.4})
```

<seaborn.axisgrid.FacetGrid at 0x7d28ae1fe9e0>




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

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

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42)
```

Training the model

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

```
lm=LinearRegression()
```

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

```
LinearRegression()
```

```
lm.coef_
```

```
array([25.72425621, 38.59713548,  0.45914788, 61.67473243])
```

```
cdf=pd.DataFrame(lm.coef_,X.columns,columns=['Coef'])
print(cdf)
```

```
                        Coef
Avg. Session Length   25.724256
Time on App           38.597135
Time on Website        0.459148
Length of Membership  61.674732
```

## Predictions

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

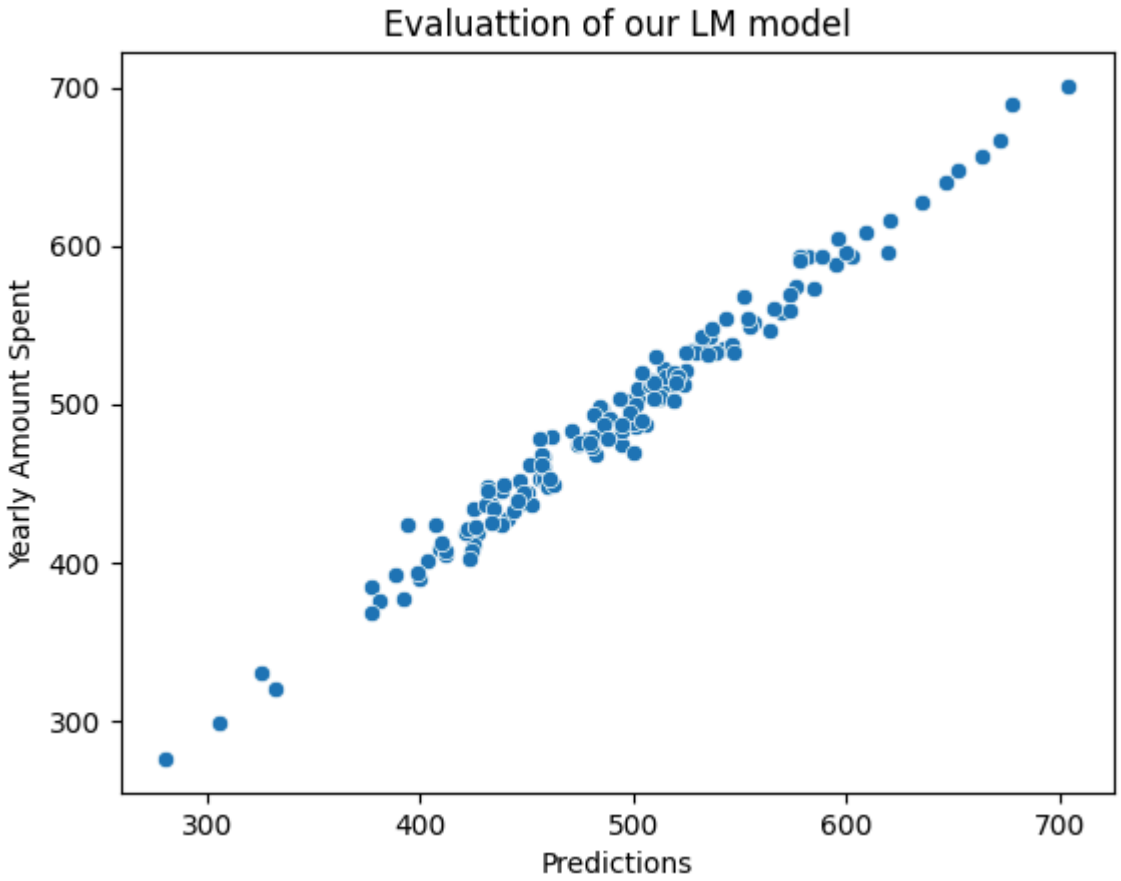
```
predictions
```

```
array([403.66993069, 542.57756289, 427.06591658, 502.02460425,
       410.12143559, 569.93442508, 531.93431341, 506.29650969,
       408.71870658, 473.97737105, 441.46912726, 425.33703059,
       425.1297229 , 527.61676714, 431.45684016, 424.0769184 ,
       575.76543296, 484.89856554, 458.35936863, 481.96502182,
```

```
       502.32441491, 513.63783554, 507.58877002, 646.57464283,
       450.24372141, 496.27043415, 556.40457807, 554.95630839,
       399.64237199, 325.84623136, 532.89783259, 478.12238702,
       501.05701845, 305.97335848, 505.77244448, 483.79591969,
       518.8331528 , 438.18241857, 456.71094234, 471.04609461,
       494.44008972, 445.31155755, 508.78802753, 501.04594193,
       488.83499673, 535.38079541, 595.20129802, 514.04714872,
       280.76758312, 433.10112367, 421.70823427, 481.23640152,
       584.71372272, 608.7748096 , 563.98513427, 494.72804869,
       394.52133407, 456.4197529 , 573.08767515, 499.6984241 ,
       512.83277025, 392.12434043, 480.05057697, 481.54520299,
       475.1117359 , 546.2717533 , 430.85039085, 602.16082001,
       422.3695128 , 493.57280186, 528.74970313, 581.49002635,
       620.19139276, 512.56880298, 411.76623862, 498.47637494,
       461.51337557, 446.41371051, 448.07229961, 535.44710412,
       599.45225302, 619.33717662, 494.15919062, 671.99976398,
       532.46469814, 438.90606319, 515.04975242, 546.7821954 ,
       331.94282076, 510.51987447, 536.57891032, 500.19533618,
       376.92345776, 573.73961388, 479.68031607, 588.61435483,
       485.69922203, 456.40200844, 399.25197845, 451.5098931 ,
       519.40693826, 434.71194217, 596.13049586, 487.91791966,
       407.46691799, 524.16812757, 504.12982787, 452.11540623,
       524.21791295, 457.59311643, 444.19371592, 457.80432916,
       448.76590761, 438.31789012, 677.04967982, 566.09639245,
       651.93616661, 381.08127926, 577.5577254 , 578.35797052,
       518.61431291, 538.94532336, 377.4301223 , 663.30814872,
       523.83158824, 456.86065622, 446.07594402, 388.55038282,
       521.03242183, 431.94999241, 460.08016327, 426.31959507,
       433.30417088, 634.89577554, 462.41086078, 460.71673829,
       512.49535288, 703.83033889, 411.84238624, 551.54681408,
       553.33669558, 409.68202123, 423.34491341, 509.66438623,
       509.88865178, 543.67591782, 504.31300469, 519.18802223,
       520.03155195, 535.13855037])
```

```
sns.scatterplot(x=predictions, y=y_test)
plt.xlabel('Predictions')
plt.title('Evaluattion of our LM model')
```

Text(0.5, 1.0, 'Evaluattion of our LM model')




```
from sklearn.metrics import mean_squared_error,mean_absolute_error
import math
```

```
print('Mean Absolute Error : ',mean_absolute_error(y_test,predictions))
print('Mean Squared Error : ',mean_squared_error(y_test,predictions))
print('Root Mean Squared Error : ',math.sqrt(mean_squared_error(y_test,predictions)))
```

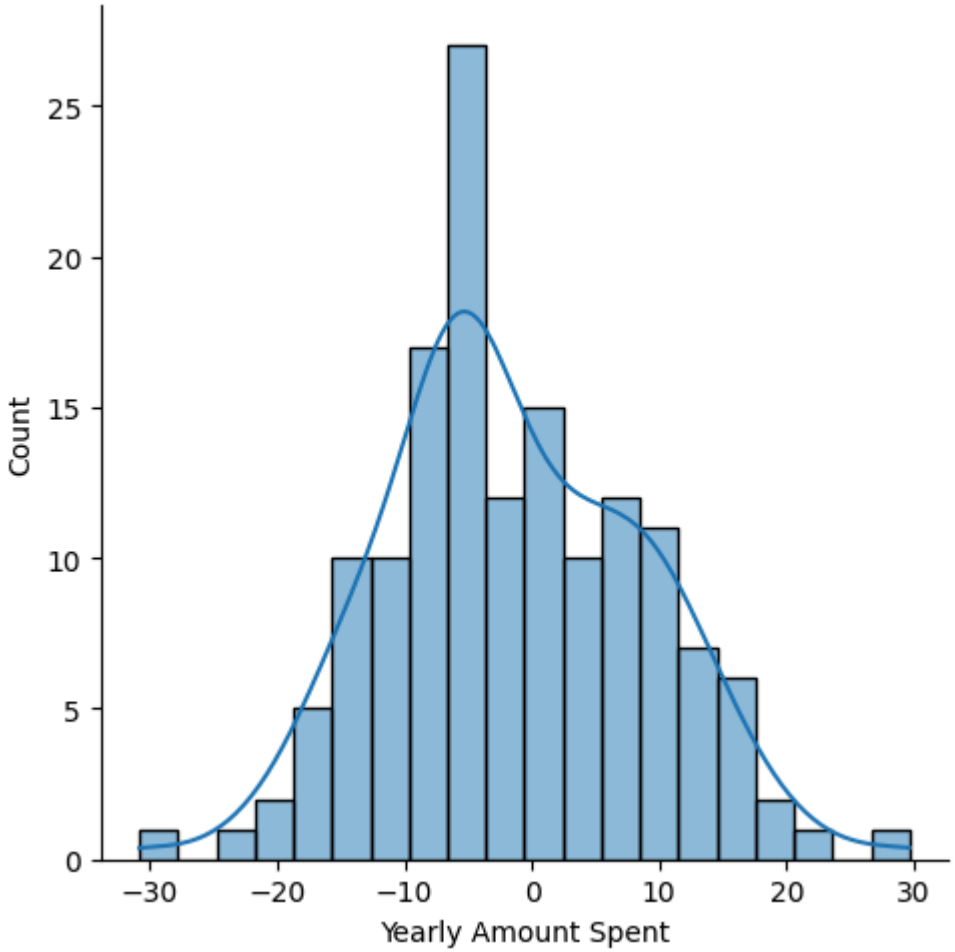
```
Mean Absolute Error :  8.426091641432116
Mean Squared Error :  103.91554136503333
Root Mean Squared Error :  10.193897260863155
```

Residuals

```
residuals=y_test - predictions
```

```
sns.displot(residuals,bins=20,kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d28ae68c520>
```




```
import pylab
import scipy.stats as stats
```

```
stats.probplot(residuals,dist="norm",plot=pylab
```