In [4]:

```python
import pandas as pd
import numpy as np
import matplotlib as plt
import seaborn as sns
```

In [5]:

```python
customers= pd.read_csv('Ecommerce Customers.csv')
customers
```

Out[5]:

	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
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461
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
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189
...	...	...	...	...	...
495	lewisjessica@craig-evans.com	4483 Jones Motorway Suite 872\nLake Jamiefurt,...	Tan	33.237660	13.566160
496	katrina56@gmail.com	172 Owen Divide Suite 497\nWest Richard, CA 19320	PaleVioletRed	34.702529	11.695736
497	dale88@hotmail.com	0787 Andrews Ranch Apt. 633\nSouth Chadburgh, ...	Cornsilk	32.646777	11.499409
498	cwilson@hotmail.com	680 Jennifer Lodge Apt. 808\nBrendachester, TX...	Teal	33.322501	12.391423
499	hannahwilson@davidson.com	49791 Rachel Heights Apt. 898\nEast Drewboroug...	DarkMagenta	33.715981	12.418808

500 rows × 8 columns

In [6]:

```
df= customers.drop(['Email', 'Address', 'Avatar'], axis =1)
#When we are dropping the column, we put axis=1
df
```

Out[6]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	34.497268	12.655651	39.577668	4.082621	587.951054
1	31.926272	11.109461	37.268959	2.664034	392.204933
2	33.000915	11.330278	37.110597	4.104543	487.547505
3	34.305557	13.717514	36.721283	3.120179	581.852344
4	33.330673	12.795189	37.536653	4.446308	599.406092
...	...	...	...	...	...
495	33.237660	13.566160	36.417985	3.746573	573.847438
496	34.702529	11.695736	37.190268	3.576526	529.049004
497	32.646777	11.499409	38.332576	4.958264	551.620145
498	33.322501	12.391423	36.840086	2.336485	456.469510
499	33.715981	12.418808	35.771016	2.735160	497.778642

500 rows × 5 columns

EDA

Lets look at our dataset

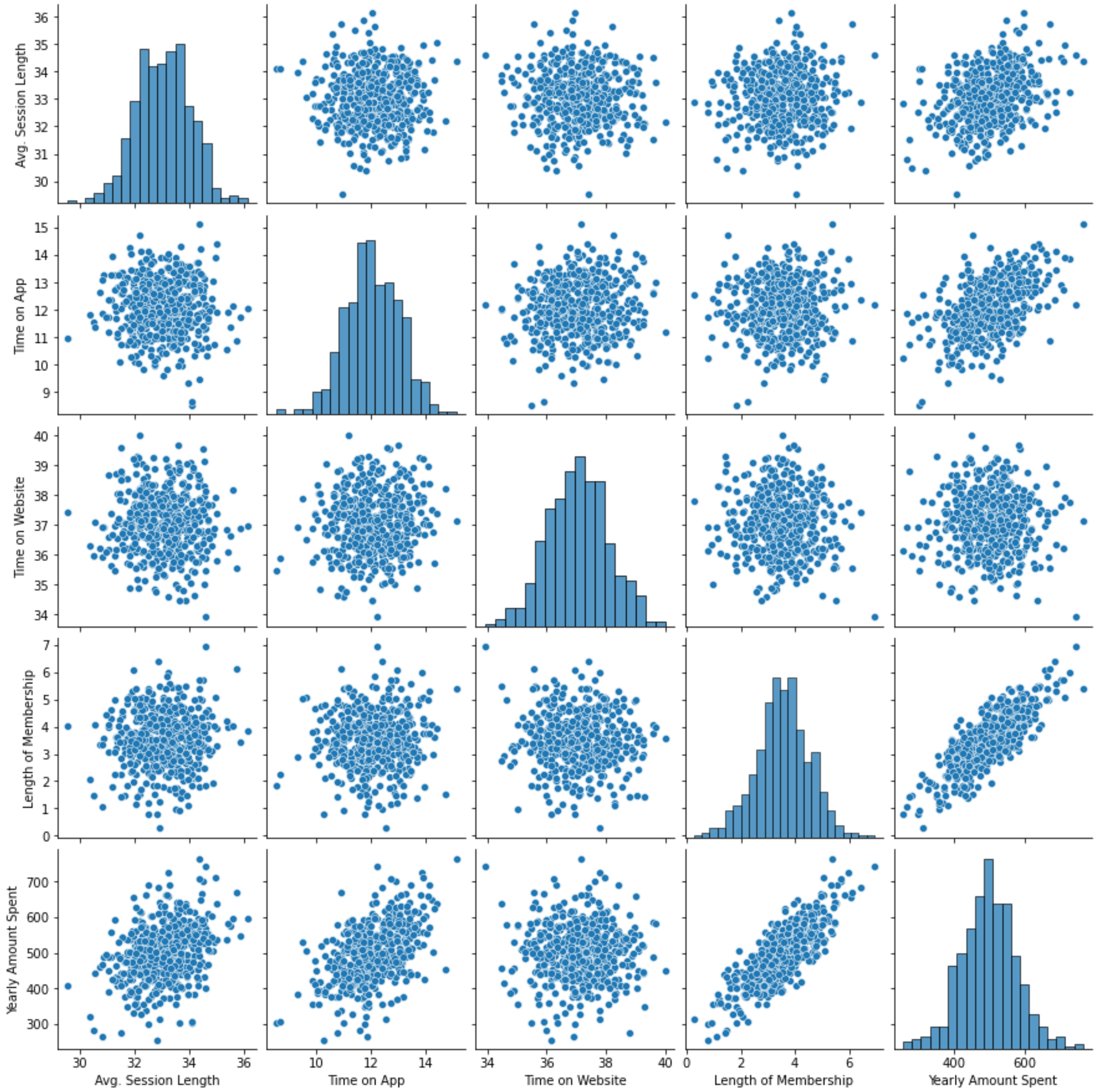
1. Does the yearly amount correlate with any of the other variables?

In [7]:

```
sns.pairplot(df)
#a pairplot plots everything against everything
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1ce3dc9e610>
```

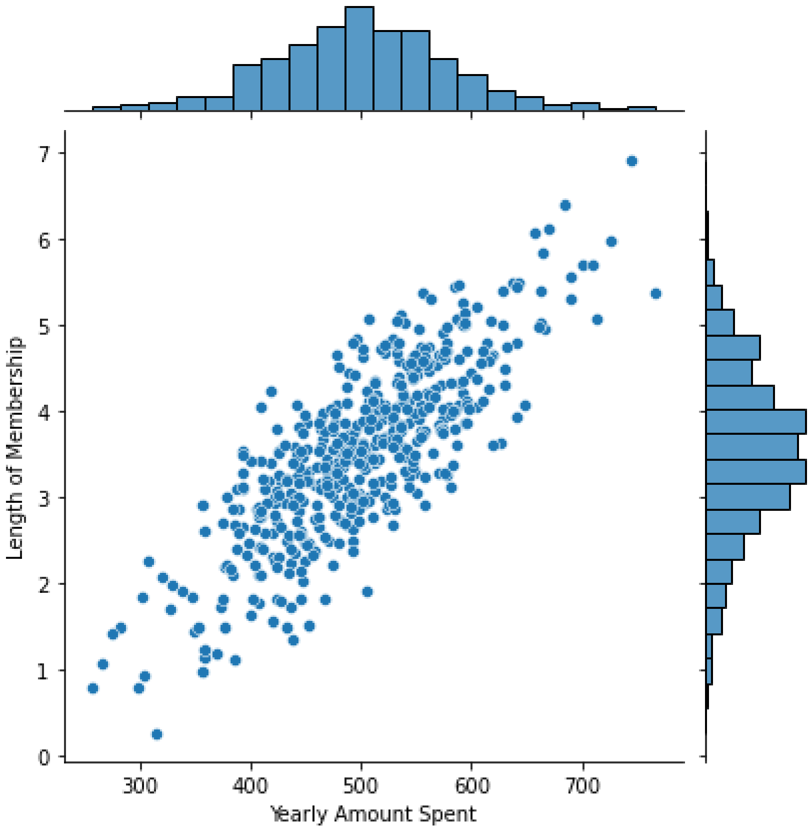


In [8]:

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

Out[8]:

```
<seaborn.axisgrid.JointGrid at 0x1ce42f9deb0>
```

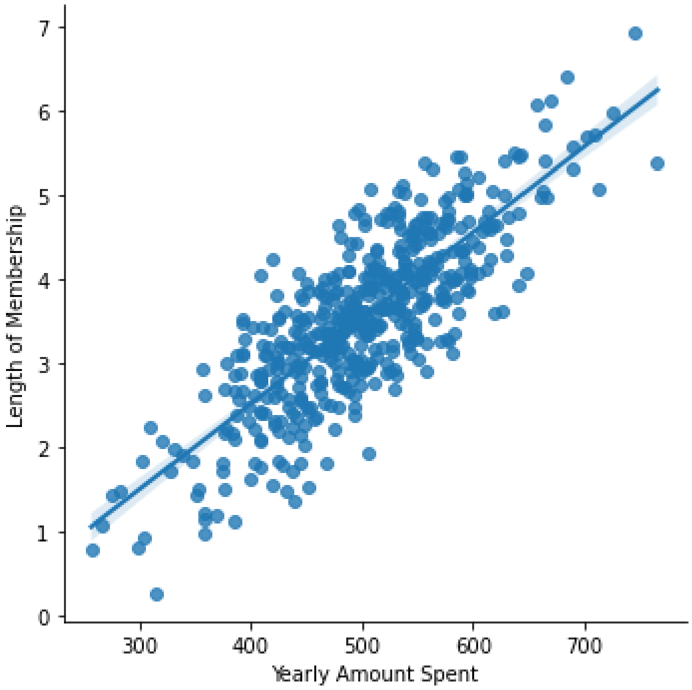


In [9]:

```
#create a linear model plot
sns.lmplot(x='Yearly Amount Spent', y= 'Length of Membership', data =df)
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1ce42d199d0>
```



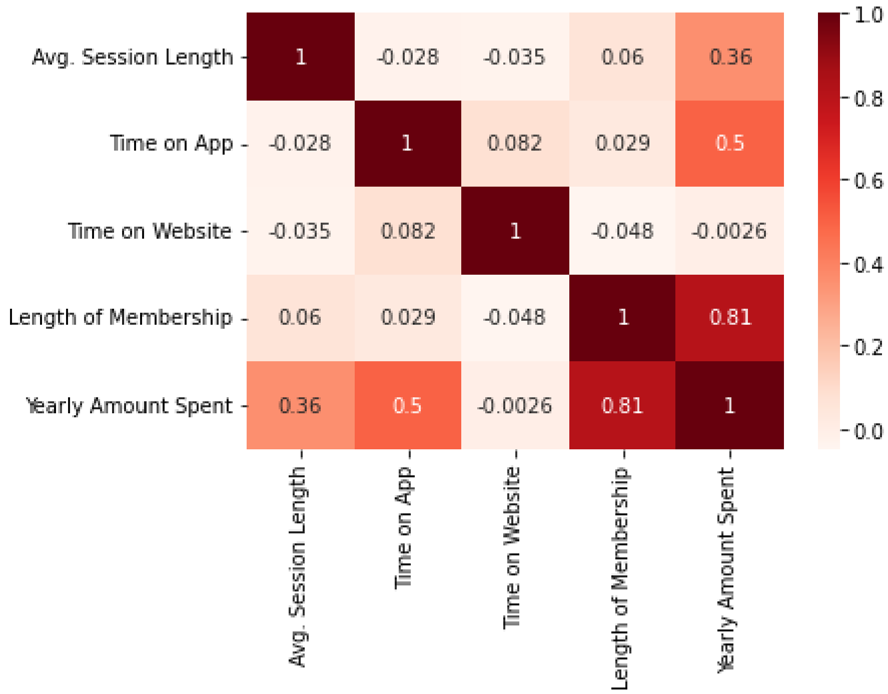
Corellation plot

In [10]:

```python
sns.heatmap(df.corr(), annot= True, cmap='Reds')
```

Out[10]:

```
<AxesSubplot:>
```

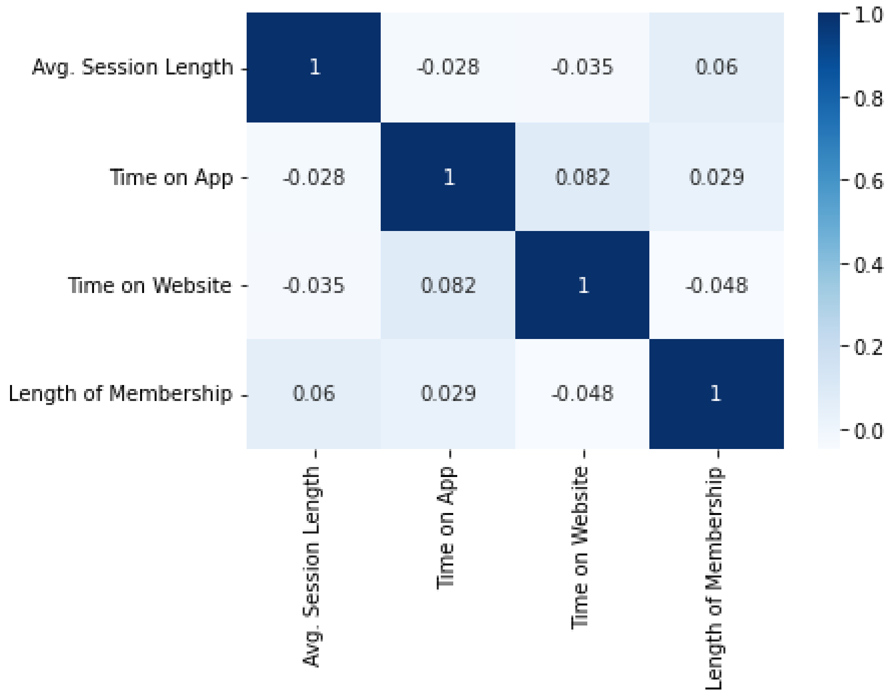


In [11]:

```python
#when the numbers are too high, think about droping a particular variable that doesnt fit i
sns.heatmap(df.drop('Yearly Amount Spent', axis=1).corr(), annot= True, cmap='Blues')
```

Out[11]:

```
<AxesSubplot:>
```



####Training and Testing data

we need to seperate the features and lable we need to split our data into training and testing data

In [13]:

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

In [14]:

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

In [15]:

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

In [16]:

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

In [17]:

```
X_train
```

Out[17]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership
202	31.525752	11.340036	37.039514	3.811248
428	31.862741	14.039867	37.022269	3.738225
392	33.258238	11.514949	37.128039	4.662845
86	33.877779	12.517666	37.151921	2.669942
443	33.025020	12.504220	37.645839	4.051382
...	...	...	...	...
63	32.789773	11.670066	37.408748	3.414688
326	33.217188	10.999684	38.442767	4.243813
337	31.827979	12.461147	37.428997	2.974737
11	33.879361	11.584783	37.087926	3.713209
351	32.189845	11.386776	38.197483	4.808320

350 rows × 4 columns

In [18]:

```
X_test
```

Out[18]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership
18	32.187812	14.715388	38.244115	1.516576
361	32.077590	10.347877	39.045156	3.434560
104	31.389585	10.994224	38.074452	3.428860
4	33.330673	12.795189	37.536653	4.446308
156	32.294642	12.443048	37.327848	5.084861
...	...	...	...	...
147	32.255901	10.480507	37.338670	4.514122
346	32.765665	12.506548	35.823467	3.126509
423	33.128693	10.398458	36.683393	3.859818
17	32.338899	12.013195	38.385137	2.420806
259	32.096109	10.804891	37.372762	2.699562

150 rows × 4 columns

Training the model NB: we only use train data to train and test data to test, don't test on your training data!

In [19]:

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

In [20]:

```
#create an instance of a linear regression model

model = LinearRegression()
```

In [21]:

```
#train or fit the data
model.fit(X_train, y_train)
```

Out[21]:

```
LinearRegression()
```

In [22]:

```
X.columns
```

Out[22]:

```
Index(['Avg. Session Length', 'Time on App', 'Time on Website',
       'Length of Membership'],
      dtype='object')
```

In [23]:

```
model.coef_
```

Out[23]:

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

In [24]:

```
model.intercept_
```

Out[24]:

```
-1047.932782250239
```

In [25]:

```
#the equation of the regresion is
# Yearly amount spent = 26*Avg. Session Length+38.6*'Time on App'+ 0.19*Time on Website+61.
```

Predicting on test data we want to evaluate the performance of the model on our test data

In [26]:

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

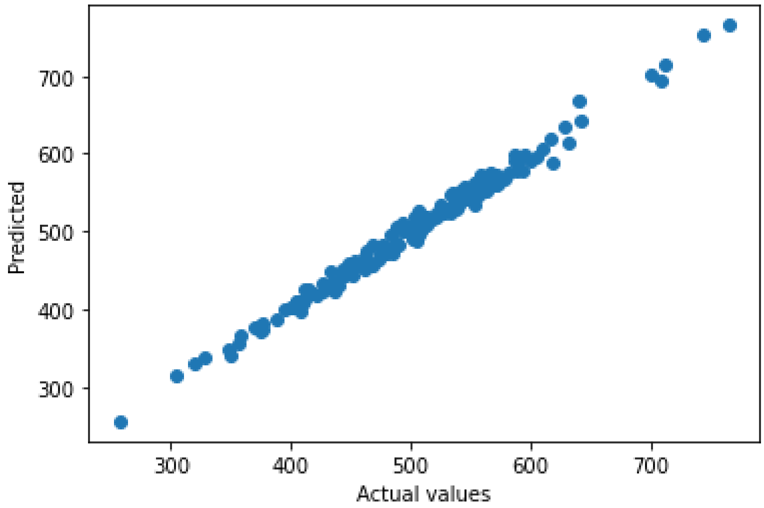
let's compare our prediction with the y_test on a scatter plot

In [27]:

```
plt.pyplot.scatter(y_test, prediction)
plt.pyplot.ylabel('Predicted')
plt.pyplot.xlabel('Actual values')
```

Out[27]:

```
Text(0.5, 0, 'Actual values')
```



Mini exercise

1. What evaluation metrics do we use for linear regression model?
2. Briefly discuss these metrics within your group
3. Evaluate the model here using the metrics you identified
4. Based on your evaluation, is this a good model? why?
5. Explain breifly what the values you obtain for each metric mean in this partiular case

1.What evaluation metrics do we use for linear regression model?

Mean squared Error, Mean Absolute Error, R-Squared or Coefficient of determination, Root Mean Squared Error, Root Mean Squared Log Error

2. Briefly discuss these metrics within your group

We watched some videos on youtube and learnt about the different metrics

3. Evaluate the model

In [28]:

```
#Mean Absolute Error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
mean_absolute_error(y_test,prediction)
```

Out[28]:

```
7.228148653430832
```

In [29]:

```
#Mean Squared Error
MSE = mean_squared_error(y_test, prediction)
MSE
```

Out[29]:

```
79.81305165097456
```

In [30]:

```
#R-Squared
from sklearn.metrics import r2_score
r2_score(y_test,prediction)
```

Out[30]:

```
0.9890046246741234
```

In [34]:

```
#Root Mean Squared Error
import math
math.sqrt(MSE)
```

Out[34]:

```
8.93381506697864
```

In [35]:

```
#Root Mean Squared Log Error
from sklearn.metrics import mean_squared_log_error
MSLE = mean_squared_log_error(y_test, prediction)
MSLE
math.sqrt(MSLE)
```

Out[35]:

```
0.017753763103974033
```

```
Mean Absolute Error = 7.228148653430832. A mean absolute error of 0 means the model is
perfect. the father away from 0 the less perfect the model.

Mean Square Error= 79.81305165097456 This is better for datasets where there are outliners
```

```
R-Squared = 0.9890046246741234 The model is good

Root Mean Squared Error = 8.93381506697864

Root Mean Squared Log Error= 0.017753763103974033


The model is good.
```