# Regressão Linear - Projeto

Parabéns! Você obteve algum contrato de trabalho com uma empresa de comércio eletrônico com sede na cidade de Nova York que vende roupas online, mas também tem sessões de consultoria em estilo e vestuário na loja. Os clientes entram na loja, têm sessões / reuniões com um estilista pessoal, então podem ir para casa e encomendarem em um aplicativo móvel ou site para a roupa que desejam.

A empresa está tentando decidir se deve concentrar seus esforços em sua experiência em aplicativos móveis ou em seu site. Eles contrataram você no contrato para ajudá-los a descobrir isso! Vamos começar!

Basta seguir as etapas abaixo para analisar os dados do cliente (é falso, não se preocupe, eu não lhe dei números reais de cartões de crédito ou e-mails).

## Imports

**Importe pandas, numpy, matplotlib,e seaborn. Em seguida, configure% matplotlib inline (Você importará sklearn conforme você precisar).**

In [4]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

# Obter dados

Trabalharemos com o arquivo csv do Ecommerce Customers da empresa. Possui informações do cliente, como Email, Endereço e sua cor Avatar. Em seguida, ele também possui colunas de valores numéricos:.

- Avg. Session Length: Tempo médio das sessões de consultoria de estilo na loja.
- Time on App: tempo médio gasto no app em minutos.
- Time on Website: tempo médio gasto no site em minutos.
- Lenght of Membership: Há quantos anos o cliente é membro.

**Leia no arquivo csv do Ecommerce Customers como um DataFrame chamado clientes.**

```
In [5]: clientes = pd.read_csv("Ecommerce Customers")
```

**Verifique o cabeçalho dos clientes e confira os seus métodos info () e describe().**

```
In [6]: clientes.head()
```

Out[6]:

| | 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 [7]:
```
clientes.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: 31.4+ KB
```

In [8]:
```
clientes.describe()
```

Out[8]:

| | 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 [ ]:

In [ ]:

In [ ]:

# Análise de dados exploratória

**Vamos explorar os dados!**

Pelo resto do exercício, só estaremos usando os dados numéricos do arquivo csv.

---

**Use seaborn para criar um jointplot para comparar as colunas Time On Website e Volume anual. A correlação faz sentido?**

In [10]:
```
clientes.columns
```

Out[10]:
```
Index(['Email', 'Address', 'Avatar', 'Avg. Session Length', 'Time on App',
       'Time on Website', 'Length of Membership', 'Yearly Amount Spent'],
      dtype='object')
```

In [12]:
```
clientes_dados_numericos = clientes[['Avg. Session Length', 'Time on App',
       'Time on Website', 'Length of Membership', 'Yearly Amount Spent']]
```

In [14]:
```
clientes_dados_numericos
```

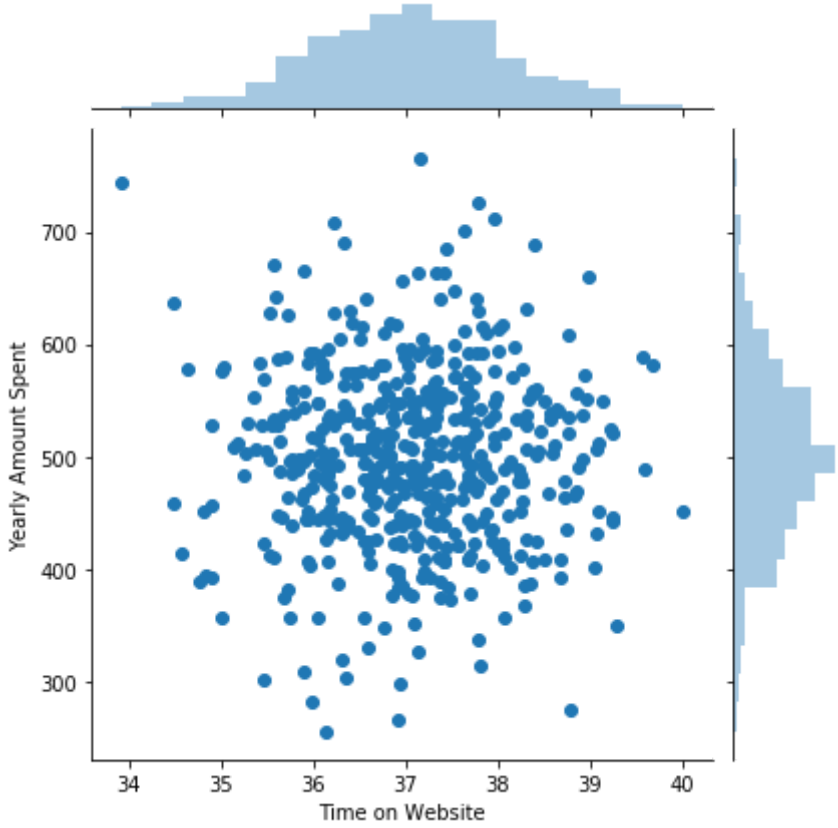
Out[14]:

| | 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

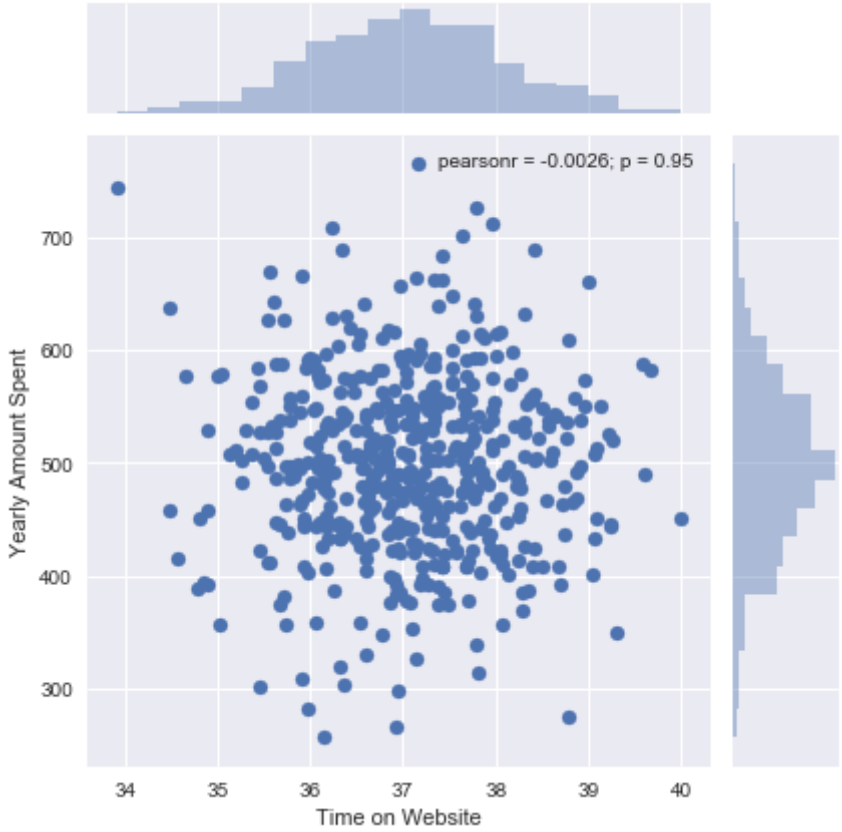
In [15]:
```
sns.jointplot(x = clientes_dados_numericos['Time on Website'],
              y = clientes_dados_numericos['Yearly Amount Spent'])
```

Out[15]: <seaborn.axisgrid.JointGrid at 0x1aebc6f8188>



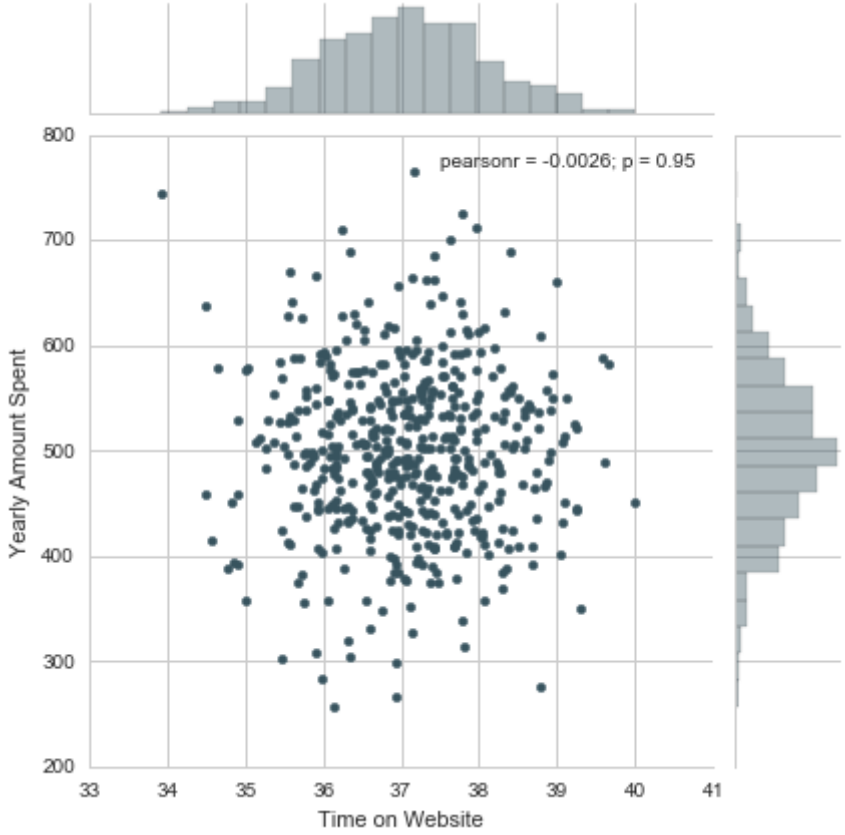
In [6]:

Out[6]: <seaborn.axisgrid.JointGrid at 0x2c7e6b05ba8>



In [281]:

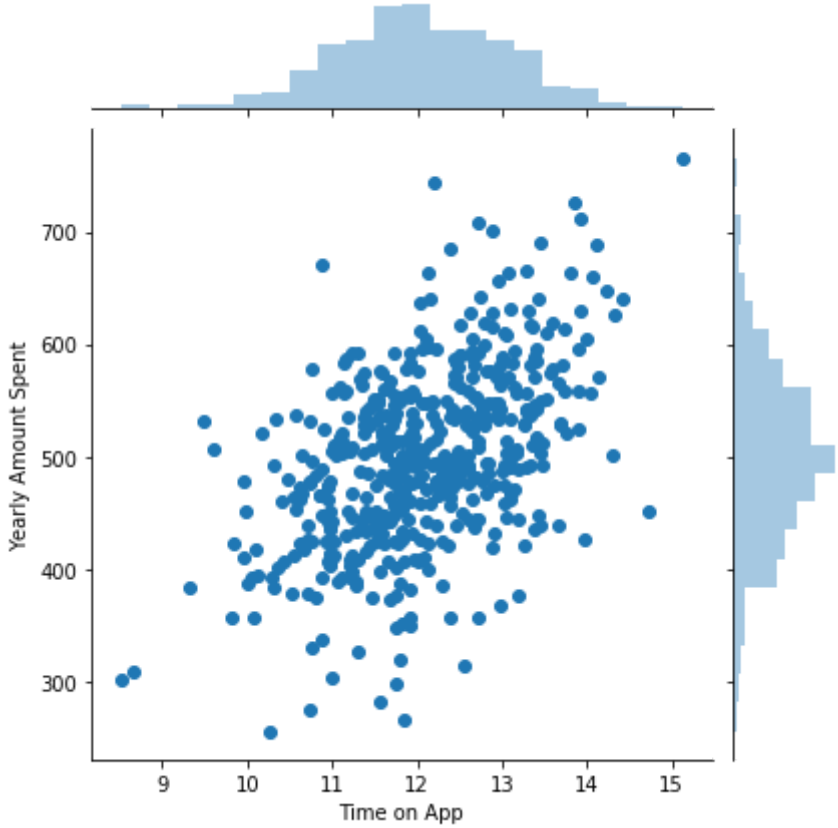
Out[281]: <seaborn.axisgrid.JointGrid at 0x120bfcc88>




**Faça o mesmo, mas com a coluna tempo no aplicativo (Time on App), em vez disso.**

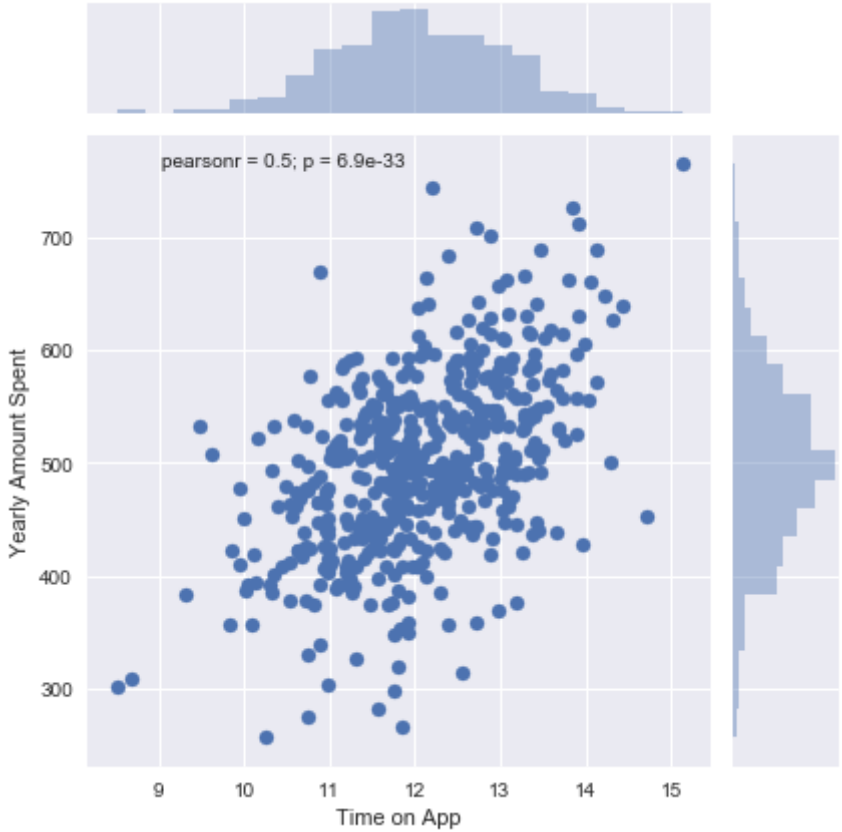
In [17]:
```
sns.jointplot(x = clientes_dados_numericos['Time on App'],
              y = clientes_dados_numericos['Yearly Amount Spent'])
```

Out[17]: <seaborn.axisgrid.JointGrid at 0x1aebca59288>



In [7]:

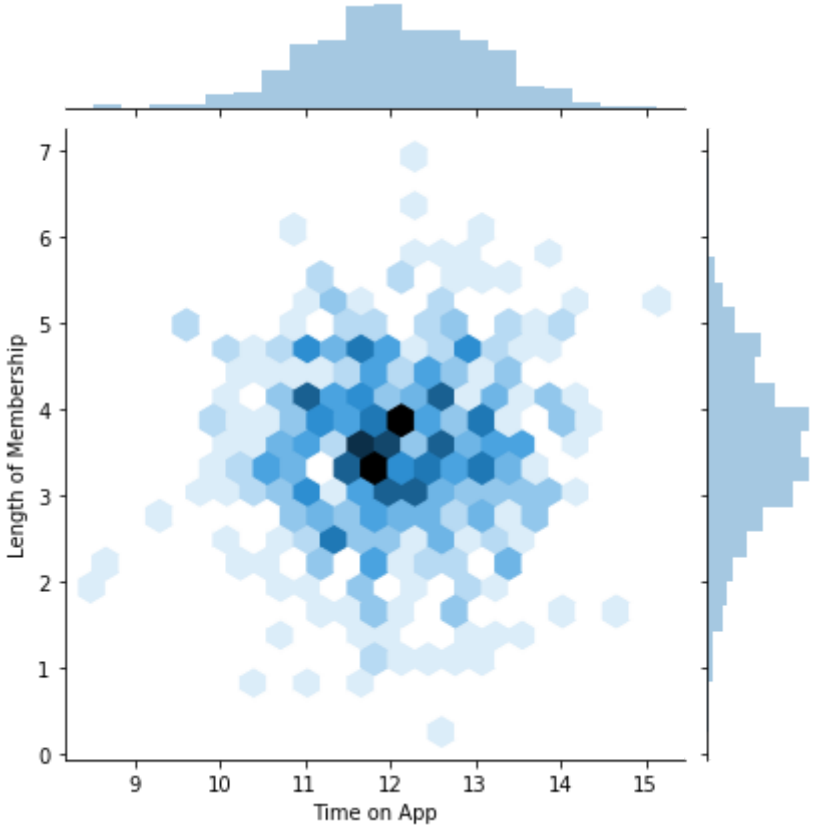
Out[7]: <seaborn.axisgrid.JointGrid at 0x2c7e94d78d0>




**Use jointplot criar um lote de caixa hexagonal 2D que compara tempo no aplicativo (Time on App) e o tempo da associação (Length of Membership).**

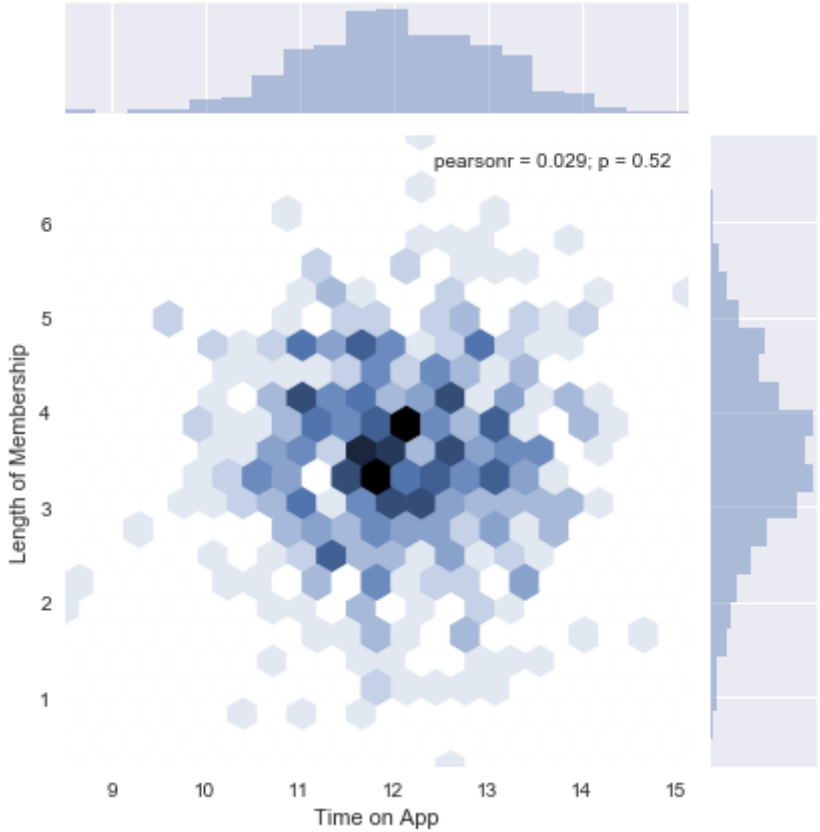
In [18]:
```
sns.jointplot(x = clientes_dados_numericos['Time on App'],
              y = clientes_dados_numericos['Length of Membership'],kind = 'hex')
```

Out[18]: <seaborn.axisgrid.JointGrid at 0x1aebcbcc1c8>



In [8]:

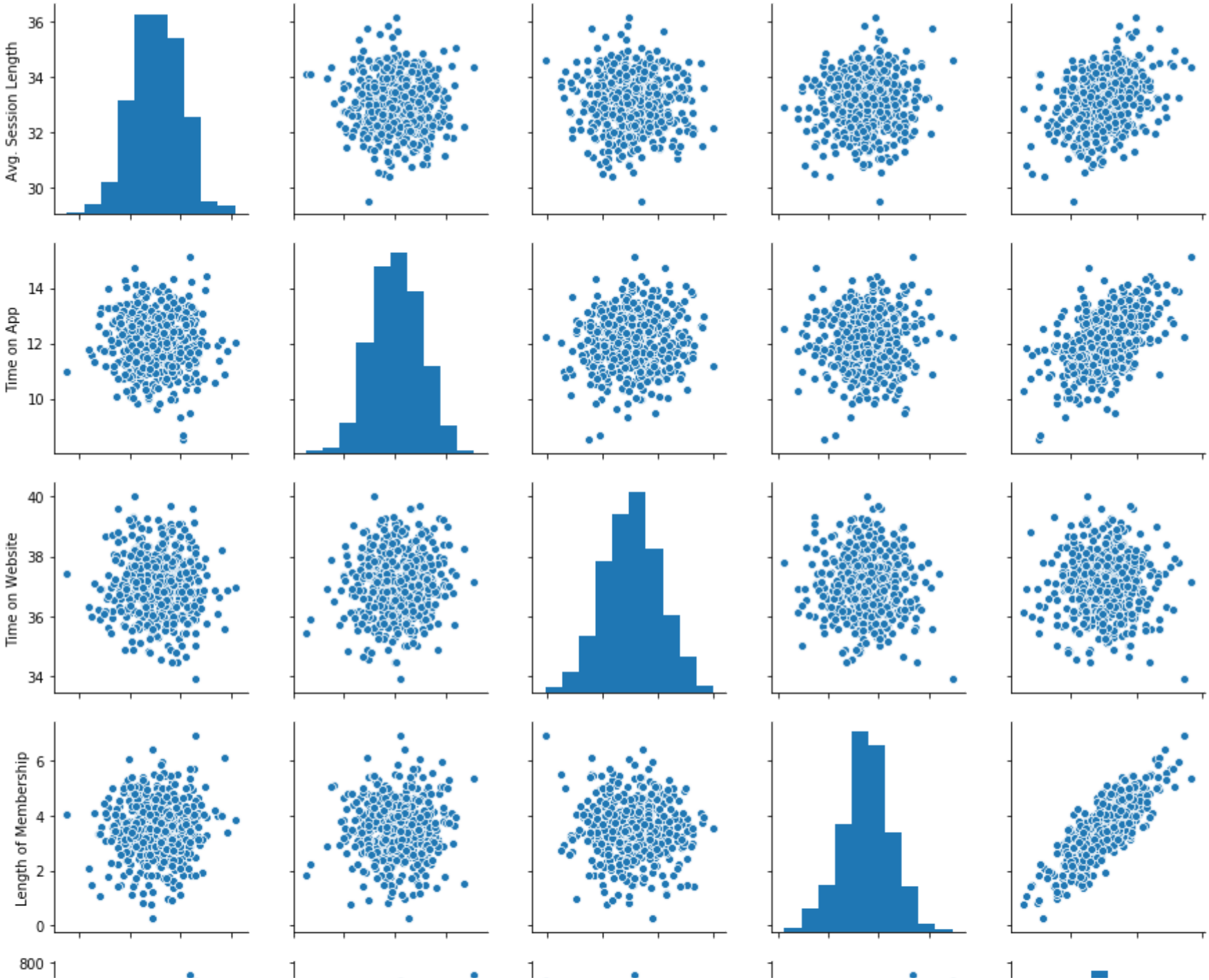
Out[8]: <seaborn.axisgrid.JointGrid at 0x2c7ec72b240>

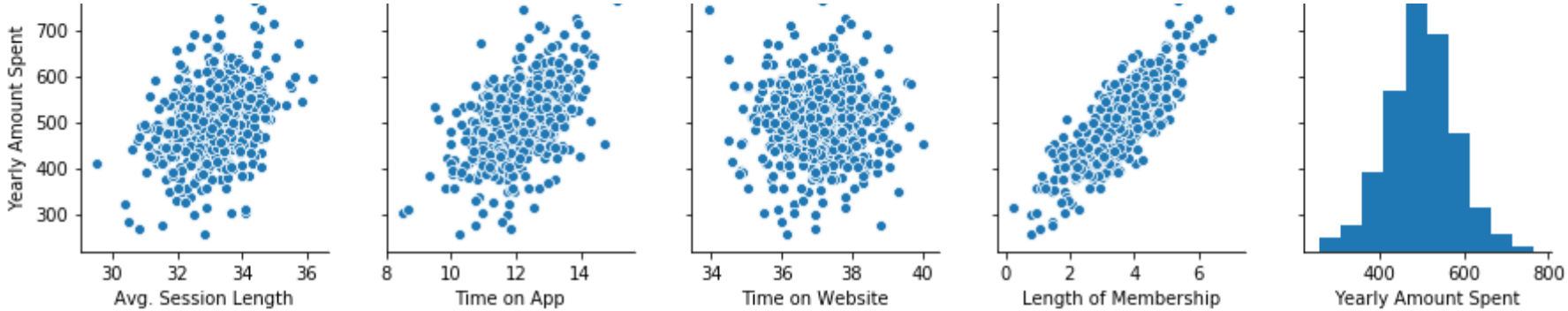



**Vamos explorar esses tipos de relações em todo o conjunto de dados. Use [parplot](https://stanford.edu/~mwaskom/software/seaborn/tutorial/axis_grids.html#plotting-pairwise-relationships-with-pairgrid-and-pairplot) para recriar o gráfico abaixo. (Não se preocupe com as cores)**

In [19]:
```
sns.pairplot(clientes_dados_numericos)
```

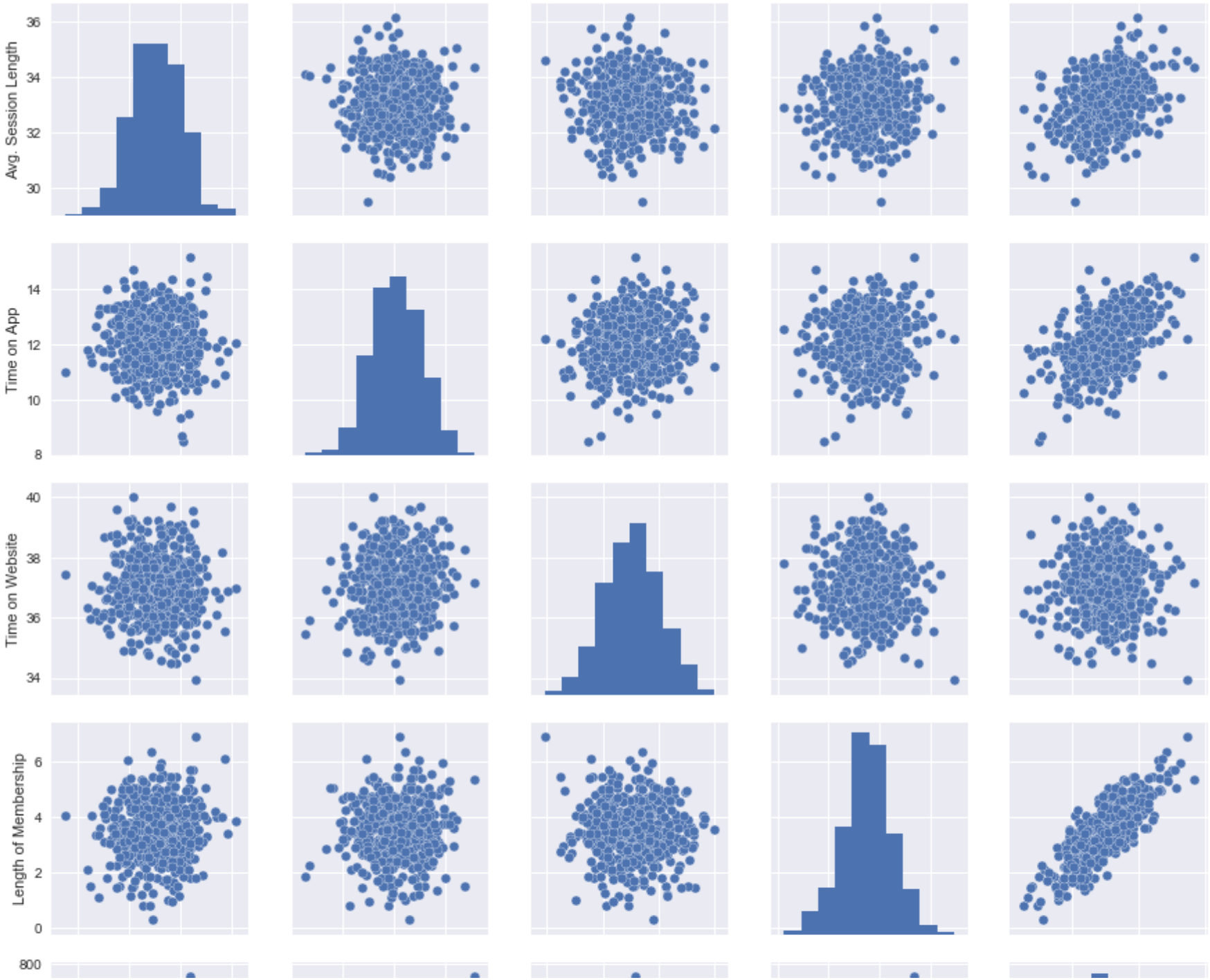
Out[19]: <seaborn.axisgrid.PairGrid at 0x1aebcdb3648>

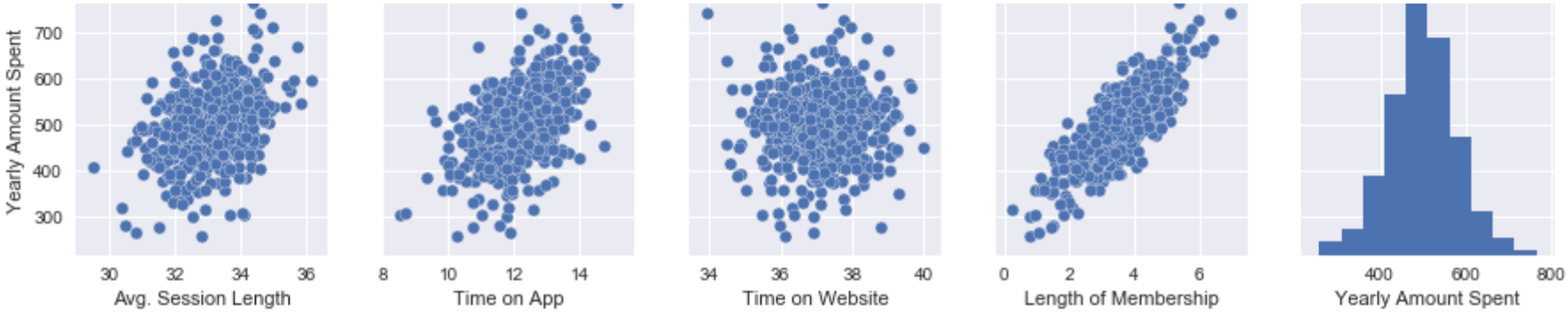

Avg. Session Length
Time on App
Time on Website
Length of Membership
36
34
32
30
14
12
10
40
38
36
34
6
4
2
0
800


Yearly Amount Spent
700
600
500
400
300
30
32
34
36
Avg. Session Length
8
10
12
14
Time on App
34
36
38
40
Time on Website
0
2
4
6
Length of Membership
400
600
800
Yearly Amount Spent

In [9]:

Out[9]: <seaborn.axisgrid.PairGrid at 0x2c7e947dda0>


Avg. Session Length
Time on App
Time on Website
Length of Membership
36
34
32
30
14
12
10
8
40
38
36
34
6
4
2
0
800




**Baseado neste plot o que parece ser a característica mais correlacionada com o valor anual gasto (Yearly Amount Spent)?**

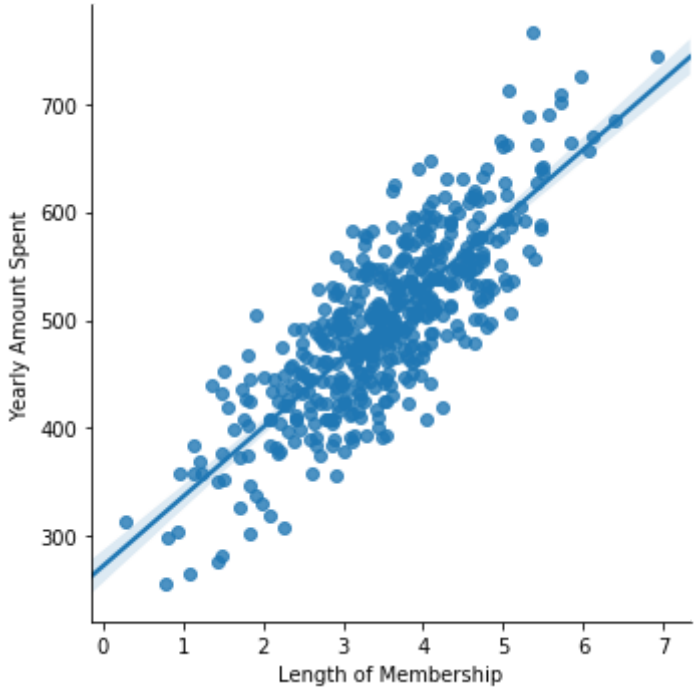
Considerando as informações obtidas pelo pairplot, aparentemente a maioria das variáveis apresenta comporta linear com relação a variável "Yearly Amount Spent". A única que esse comportamento não ficou muito claro foi o da variável "Time on Website".

In [285]:

**Crie um plot de um modelo linear (usando o lmplot de Seaborn) da quantia anual gasta (Yearly Amount Spent) vs. tempo de associação (Length of Membership).**

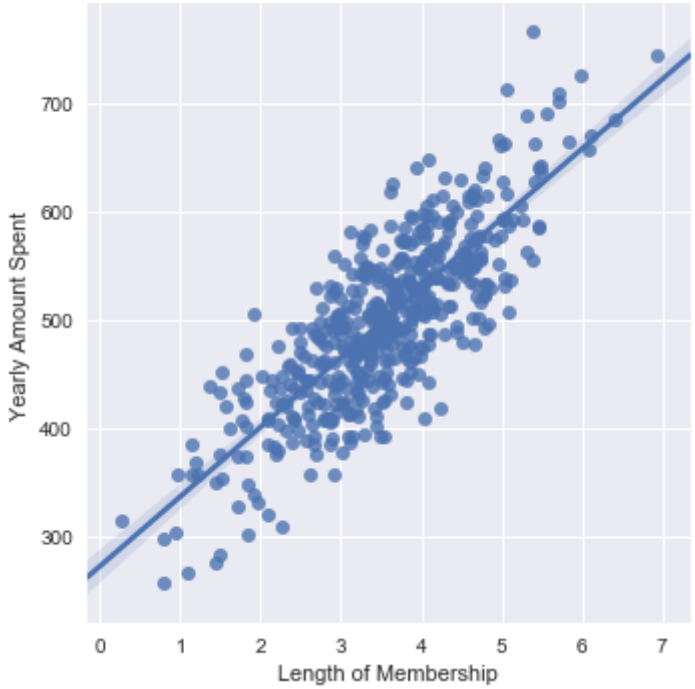
In [22]:
```
sns.lmplot(x = 'Length of Membership', y = 'Yearly Amount Spent'
           ,data = clientes_dados_numericos)
```

Out[22]: <seaborn.axisgrid.FacetGrid at 0x1aebdefefc8>



In [11]:

Out[11]: <seaborn.axisgrid.FacetGrid at 0x2c7edce7e80>




# Treinando e testando os dados

Agora que exploramos um pouco os dados, vamos avançar e dividir os dados em conjuntos de treinamento e teste. **Defina uma variável X igual a todas as características numéricas dos clientes e uma variável y igual à coluna Valor anual gasto (Yearly Amount Spent).**

In [23]:
```
X = clientes_dados_numericos[['Avg. Session Length', 'Time on App',
       'Time on Website', 'Length of Membership']]
```

In [25]:
```
y = clientes_dados_numericos[['Yearly Amount Spent']]
```

In [ ]:

In [13]:

**Use model_selection.train_test_split da sklearn para dividir os dados em conjuntos de treinamento e teste. Defina test_size = 0.3 e random_state = 101**

In [30]:
```
from sklearn.model_selection import train_test_split_split
```

In [35]:
```
X_train, X_test, y_train,y_test = train_test_split(X,y, test_size=0.3,
                                                   random_state=101)
```

In [ ]:

In [17]:

# Treinando o modelo

Agora é hora de treinar nosso modelo em nossos dados de treinamento!

**Importe LinearRegression do sklearn.linear_model**

In [39]:
```
from sklearn.linear_model import LinearRegression
```

In [18]:

**Crie uma instância de um modelo LinearRegression () chamado lm.**

In [40]:
```
lm = LinearRegression()
```

In [19]:

**Treine lm nos dados de treinamento.**

In [41]:
```
lm.fit(X_train, y_train)
```

Out[41]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [ ]:

In [ ]:

**Print os coeficientes do modelo**

In [42]:
```
print(lm.coef_)
```

```
[[25.98154972 38.59015875  0.19040528 61.27909654]]
```

In [ ]:

In [ ]:

# Previsão de dados de teste

Agora que nos ajustamos ao nosso modelo, vamos avaliar o seu desempenho ao prever os valores de teste!

**Use lm.predict () para prever o conjunto X_test dos dados.**

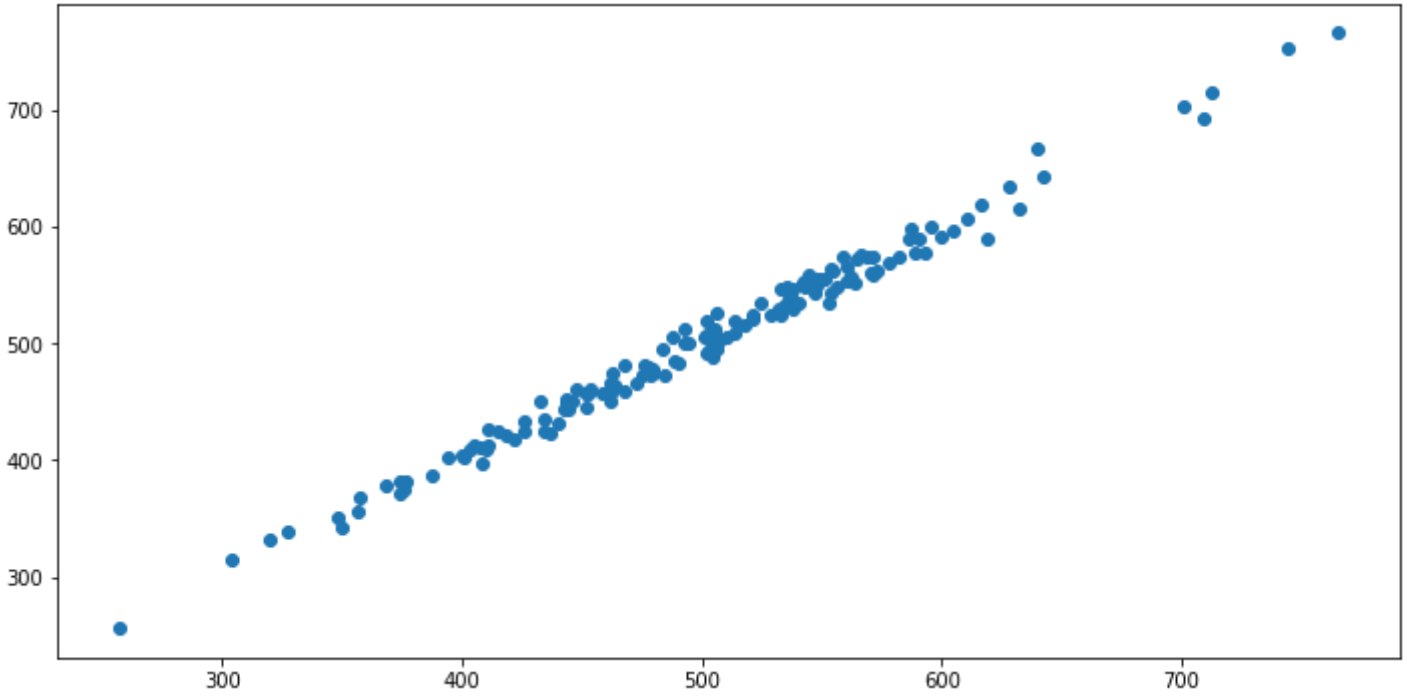
In [43]:
```
predict = lm.predict(X_test)
```

In [23]:

**Crie um diagrama de dispersão (scatterplot) dos valores reais de teste em relação aos valores preditos.**

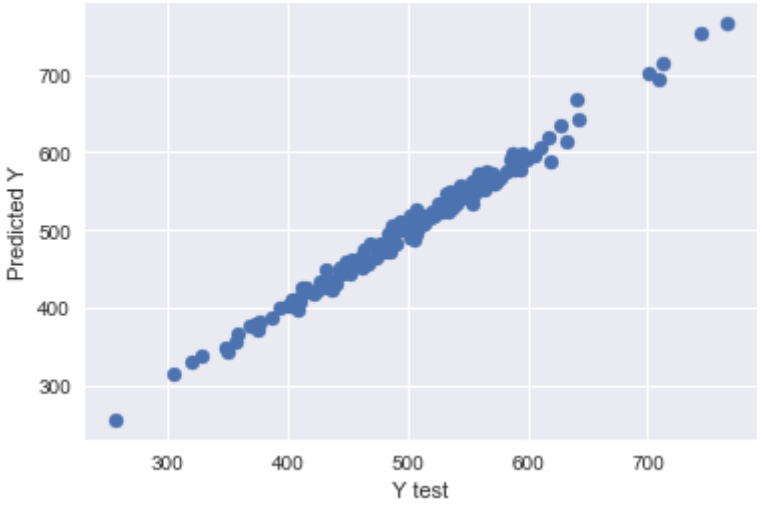
In [44]:
```
plt.figure(figsize=(12,6))
plt.scatter(y_test,predict)
```

Out[44]: <matplotlib.collections.PathCollection at 0x1aec14b2388>



In [24]:

Out[24]: <matplotlib.text.Text at 0x2c7ef745240>




# Avaliando o Modelo

Vamos avaliar o desempenho do nosso modelo calculando a soma residual dos quadrados e o escore de variância explicado (R ^ 2).

**Calcule o erro absoluto médio, o erro quadrado médio e o erro quadrado médio da raiz. Consulte a palestra ou a Wikipédia para as fórmulas**

In [45]:
```
from sklearn import metrics
```

In [46]:
```
print('MAE: ', metrics.mean_absolute_error(y_test,predict))
print('MSE: ', metrics.mean_squared_error(y_test,predict))
print('RMSE: ', np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
MAE:  7.228148653430815
MSE:  79.81305165097429
RMSE:  8.933815066978624
```

In [26]:

```
MAE: 7.22814865343
MSE: 79.813051651
RMSE: 8.93381506698
```

# Resíduos

Você deve ter obtido um modelo muito bom com um bom ajuste. Vamos explorar rapidamente os resíduos para garantir que tudo esteja bem com os nossos dados.

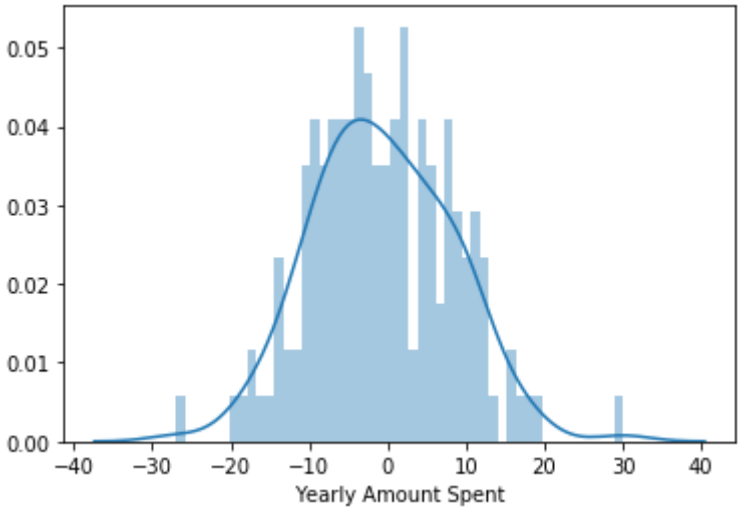
**Trace um histograma dos resíduos e certifique-se de que ele parece normalmente distribuído. Use o seaborn distplot, ou apenas o plt.hist ().**

In [51]:

```
ax = sns.distplot((y_test-predict), bins = 50)
ax.set_xlabel('Yearly Amount Spent')
```

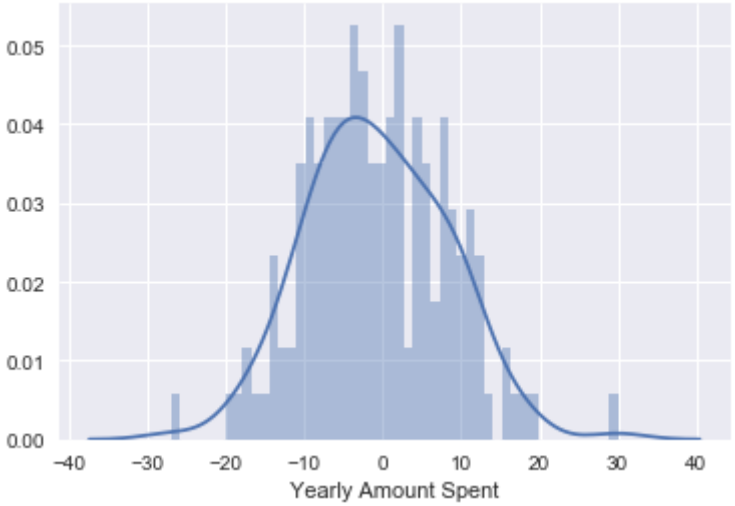
Out[51]:

```
Text(0.5, 0, 'Yearly Amount Spent')
```



In [28]:

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x2c7edd42630>




# Conclusão

Ainda desejamos descobrir a resposta à pergunta original, concentramos-nos no desenvolvimento de aplicativos móveis ou de sites? Ou talvez isso realmente não importe, e o tempo como membro é o que é realmente importante? Vamos ver se podemos interpretar os coeficientes para ter uma idéia.

**Recrie o quadro de dados abaixo.**

In [53]:
```
coefs = pd.DataFrame(lm.coef_[0],np.array(X.columns).transpose(),columns = ['Coefs'])
```

In [54]:
```
coefs
```

Out[54]:

| | Coefs |
|---|---|
| Avg. Session Length | 25.981550 |
| Time on App | 38.590159 |
| Time on Website | 0.190405 |
| Length of Membership | 61.279097 |

Bom, através dos coeficientes, posso observar quais variáveis afetam mais o "Yearly Amount Spent". Neste caso, a variável "Time on App" afeta muito mais do que a variável "Time on Website". Com isso recomendaria fortemente o investimento no APP.

**Como você pode interpretar esses coeficientes?**

**Você acha que a empresa deve se concentrar mais em seu aplicativo móvel ou em seu site?**