LINEAR REGRESSION MODEL

To understand the factors that influence yearly spending and build a predictive model for accurate forecasts.*italicized text*

```
#IMPORTING NECESSARY LIBRARIES
#DATA MANIPULATION AND ANALYSIS
import pandas as pd
#VISUALIZING DATA
import matplotlib.pyplot as plt
#ADVANCED STATISTICAL VISUALIZATION
import seaborn as sns
```

```
from google.colab import files
uploaded= files.upload()
```

Choose Files Ecommerce Customers

- **Ecommerce Customers**(n/a) - 87360 bytes, last modified: 11/23/2024 - 100% done

```
Saving Ecommerce Customers to Ecommerce Customers
```

+ Code + Text

```
#LOADING DATASET
import io
df = pd.read_csv(io.BytesIO(uploaded['Ecommerce Customers']))
df.head(5)
```

	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

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

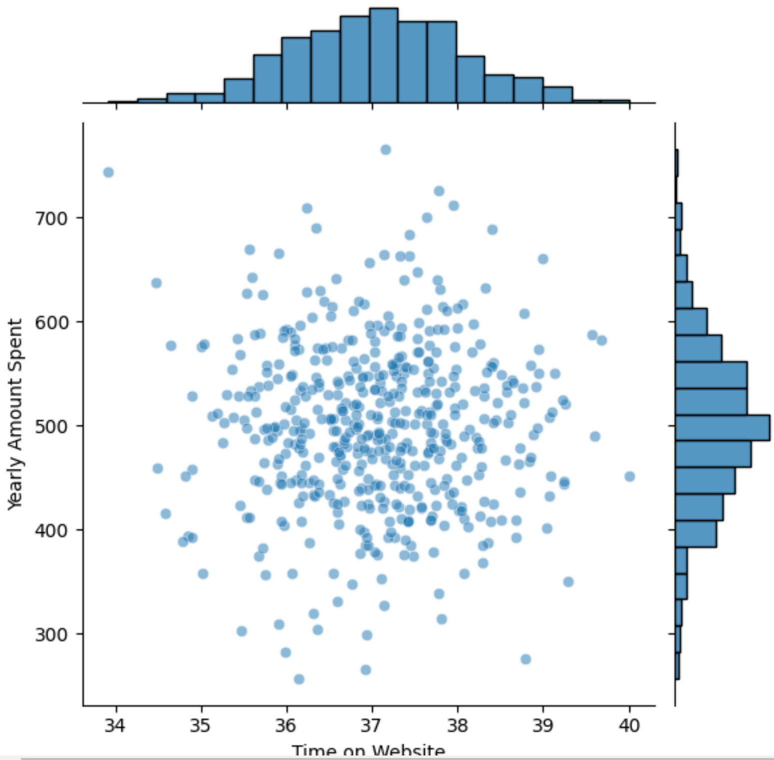
```
#BASIC INFORMATION ABOUT THE DATASET
df.info()
df.describe()
```

```
<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
```

	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

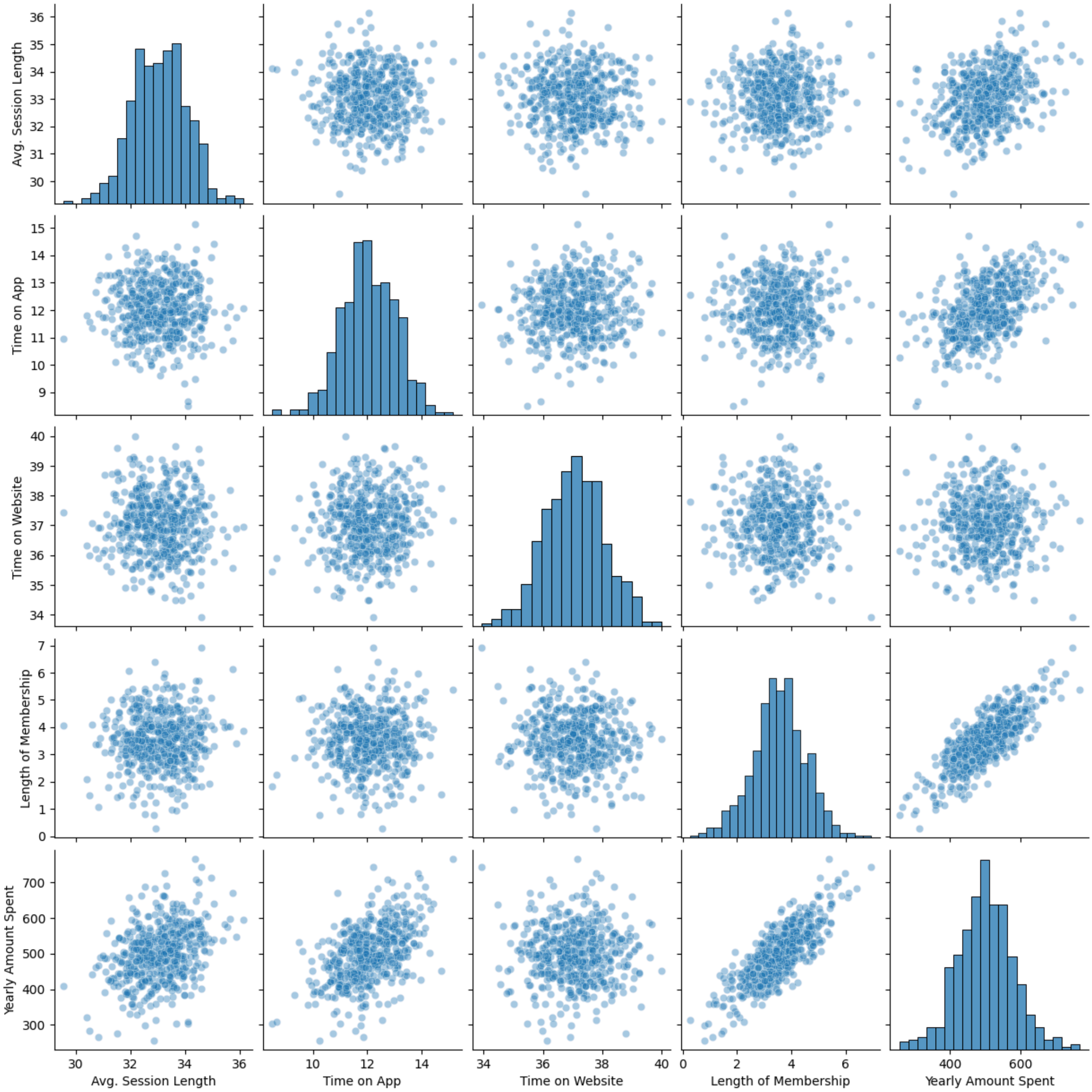
```
#EXPLORATORY DATA ANYLYSIS
sns.jointplot(x="Time on Website", y="Yearly Amount Spent", data=df, alpha=0.5)
```

```
<seaborn.axisgrid.JointGrid at 0x7dc54dc4bb80>
```




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

```
<seaborn.axisgrid.PairGrid at 0x7dc54e9d9cc0>
```




```
#SPLITTING DATASET INTO TRAINING AND TESTING SET
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)
```

```
X_train
```

	Avg. Session Length	Time on App	Time on Website	Length of Membership
5	33.871038	12.026925	34.476878	5.493507
116	33.925795	12.011022	36.701052	2.753424
45	34.555768	12.170525	39.131097	3.663105
16	32.125387	11.733862	34.894093	3.136133
462	33.503810	11.233415	37.211153	2.320550
...	...	...	...	...
106	32.291756	12.190474	36.152462	3.781823
270	34.006489	12.956277	38.655095	3.275734
348	31.812483	10.886921	34.897828	3.128639
435	32.259973	14.132893	37.023479	3.762070
102	32.425697	11.448902	37.580190	2.586968

350 rows × 4 columns

Next steps: Generate code with `X_train` View recommended plots New interactive sheet

```
X_test
```

	Avg. Session Length	Time on App	Time on Website	Length of Membership
361	32.077590	10.347877	39.045156	3.434560
73	32.808698	12.817113	37.031539	3.851579
374	31.447446	10.101632	38.043453	4.238296
155	32.449522	13.457725	37.238806	2.941411
104	31.389585	10.994224	38.074452	3.428860
...	...	...	...	...
266	34.555283	11.777772	37.979827	3.784273
23	32.903251	11.657576	36.772604	3.919302
222	34.334865	11.109456	38.585855	3.892891
261	32.550527	13.041245	36.655208	3.456234
426	31.425227	13.271475	37.239847	4.022103

150 rows × 4 columns

Next steps: Generate code with `X_test` View recommended plots New interactive sheet

```
y_train
```

	Yearly Amount Spent
5	637.102448
116	479.231093
45	549.860590
16	457.847696
462	397.420584
...	...
106	494.551861
270	540.995739
348	392.810345
435	571.216005
102	420.737673

350 rows × 1 columns

```
y_test
```

	Yearly Amount Spent
361	401.033135
73	534.777188
374	418.602742
155	503.978379
104	410.069611
...	...
266	554.003093
23	519.340989
222	502.409785
261	514.009818
426	530.766719

150 rows × 1 columns

```
#TRAINING THE MODEL
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
#DETERMINING THE COEFFICIENTS
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
```

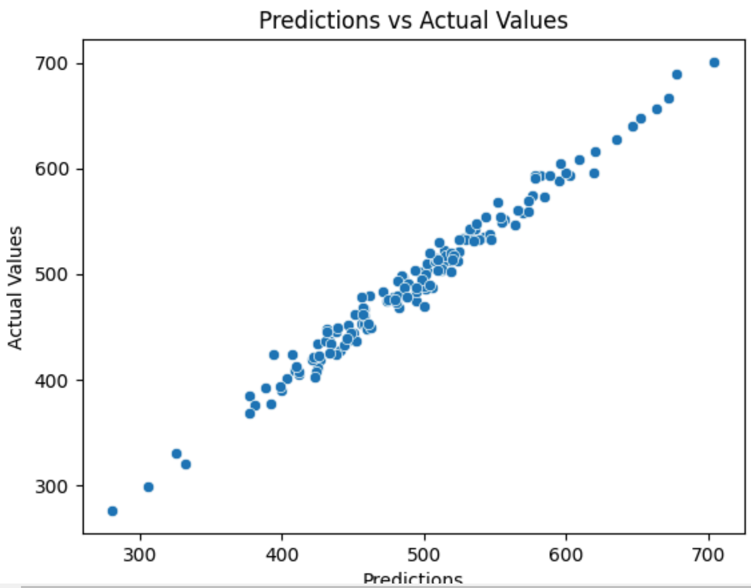
FEATURES WITH HIGHER COEFFICIENT VALUES ARE OF GREATER IMPORTANCE

```
#PREDICTIONS
predictions = lm.predict(X_test)
print(predictions)
```

```
[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]
```

```
#EVALUATION OF PREDICTIONS
dfyt = pd.DataFrame({'Predictions': predictions, 'Actual': y_test})

#SCATTERPLOT
sns.scatterplot(x='Predictions', y='Actual', data=dfyt)
plt.xlabel("Predictions")
plt.ylabel("Actual Values")
plt.title("Predictions vs Actual Values") # Added a title
plt.show()
```



PREDICTIONS ARE SOMEWHAT ACCURATE

```
#ANALYTICAL EVALUATION OF THE ERRORS
from sklearn.metrics import mean_squared_error, mean_absolute_error
print("Mean Absoulute Error: ", mean_absolute_error(y_test,predictions))
print("Mean Squared Error: ", mean_squared_error(y_test,predictions))
```

```
Mean Absoulute Error:  8.426091641432116
Mean Squared Error:  103.91554136503333
```

```
#RESIDUALS
residuals = y_test - predictions
print(residuals)
```

```
361    -2.636795
73     -7.800375
374    -8.463174
155     1.953775
104    -0.051825
         ...
266    10.327176
23     15.027984
222   -16.778237
261    -6.021734
426    -4.371832
Name: Yearly Amount Spent, Length: 150, dtype: float64
```