Import the necessary Libraries

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

import the data

In [14]:
```
customers = pd.read_excel("C:/Users/admin/Desktop/E_commerce_linear_regression/Ecommerce Customers.xlsx")
```

In [19]:
```
customers.head()
```

Out[19]:

	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	NaN	NaN	NaN	NaN	NaN	NaN
1	Wrightmouth	" MI 82180-9605"""	Violet	34.497268	12.655651	39.577668	4.082621	587.951054
2	hduke@hotmail.com	4547 Archer Common	NaN	NaN	NaN	NaN	NaN	NaN
3	Diazchester	" CA 06566-8576"""	DarkGreen	31.926272	11.109461	37.268959	2.664034	392.204933
4	pallen@yahoo.com	24645 Valerie Unions Suite 582	NaN	NaN	NaN	NaN	NaN	NaN

In [20]:
```
# Summary statistics
print(customers.describe())
```

```
       Avg. Session Length  Time on App  Time on Website  \
count           500.000000   500.000000       500.000000
mean             31.130462    14.350284        33.956926
std               6.128380     7.327838         9.787361
min               8.668350     8.508152         0.789520
25%              32.086022    11.469670        36.088966
50%              32.904773    12.109744        36.943004
75%              33.636921    12.992473        37.648456
max              36.139662    39.220713        40.005182

       Length of Membership  Yearly Amount Spent
count            500.000000           454.000000
mean              48.607305           499.919858
std              144.216919            78.325793
min                0.269901           266.086341
25%                3.014640           445.917049
50%                3.653135           498.887875
75%                4.534414           549.100737
max              744.221867           765.518462
```

In [21]:
```
# Info about data types and nulls
print(customers.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Email                 1000 non-null   object 
 1   Address               1000 non-null   object 
 2   Avatar                516 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   454 non-null    float64
dtypes: float64(5), object(3)
memory usage: 62.6+ KB
None
```

Check for Missing Values

In [22]:
```
# Check for nulls
print(customers.isnull().sum())
```

```
Email                   0
Address                 0
Avatar                484
Avg. Session Length     500
Time on App             500
Time on Website         500
Length of Membership    500
Yearly Amount Spent     546
dtype: int64
```

In [23]:
```
 # drop rows with missing values
 customers = customers.dropna()

# drop irrelevant columns
customers = customers.drop(['Email', 'Address', 'Avatar'], axis=1)

customers.head()
```

Out[23]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
1	34.497268	12.655651	39.577668	4.082621	587.951054
3	31.926272	11.109461	37.268959	2.664034	392.204933
5	33.000915	11.330278	37.110597	4.104543	487.547505
7	34.305557	13.717514	36.721283	3.120179	581.852344
9	33.330673	12.795189	37.536653	4.446308	599.406092

checking data type

In [24]:
```
print(customers.dtypes)
```

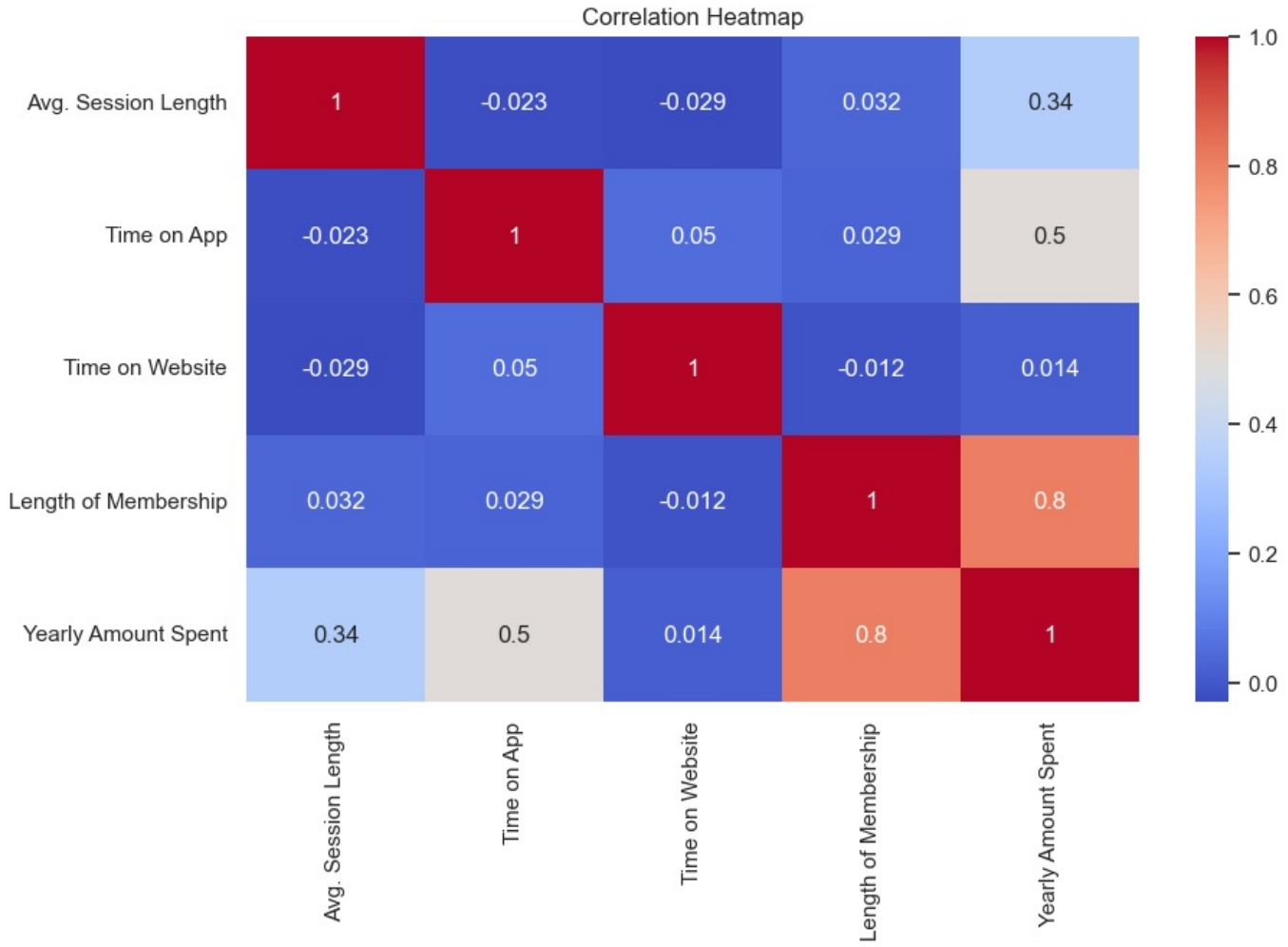
```
Avg. Session Length     float64
Time on App             float64
Time on Website         float64
Length of Membership    float64
Yearly Amount Spent     float64
dtype: object
```

Exploratory Data Analysis

In [25]:
```
# Set plot style
sns.set(style="whitegrid")
sns.set_palette("GnBu_d")
```

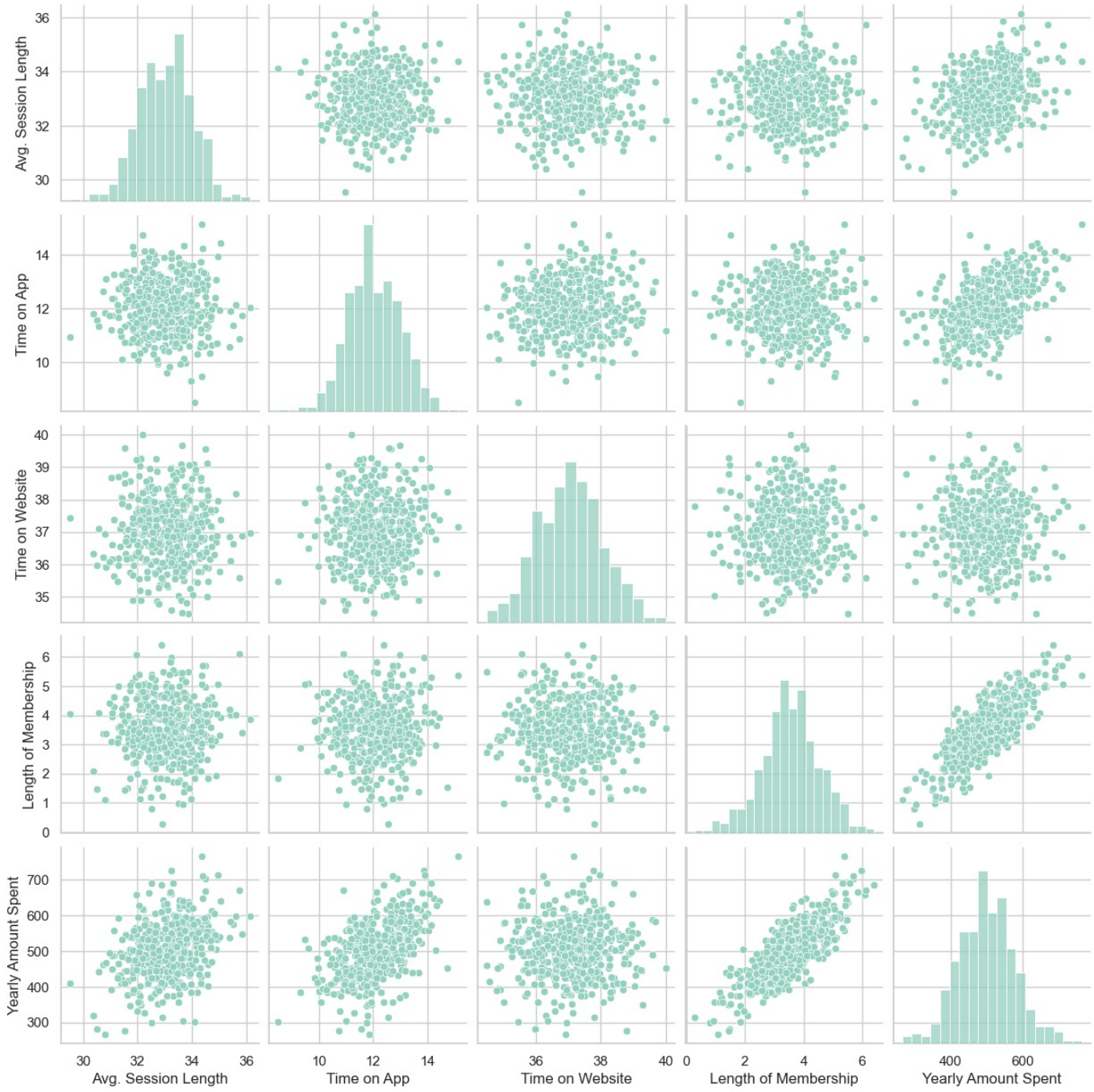
Correlation Heatmap

In [26]:
```
plt.figure(figsize=(10, 6))
sns.heatmap(customers.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



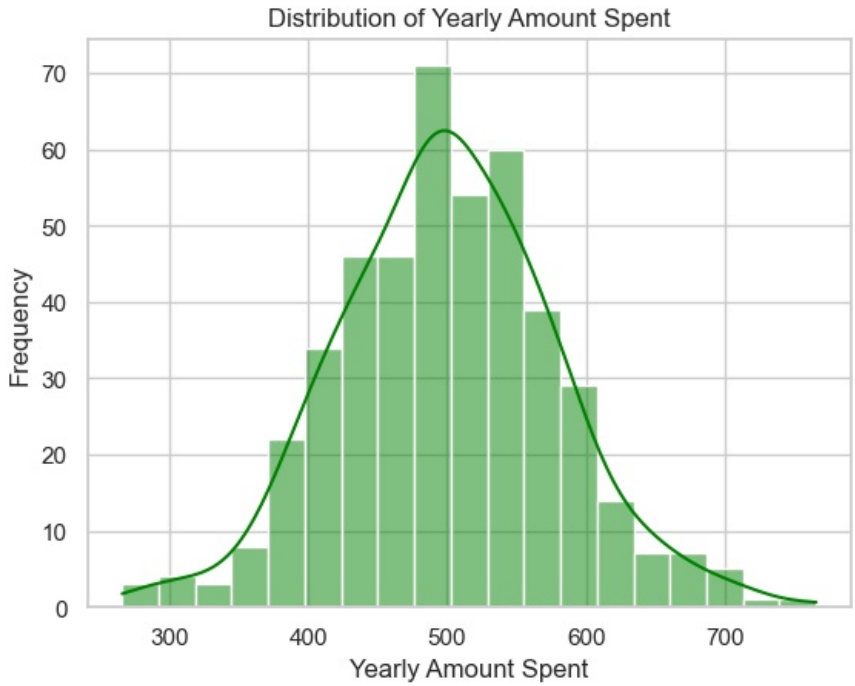
Pairplot for Feature Relationships

In [28]:
```
sns.pairplot(customers)
plt.show()
```



Distribution of Yearly Amount Spent

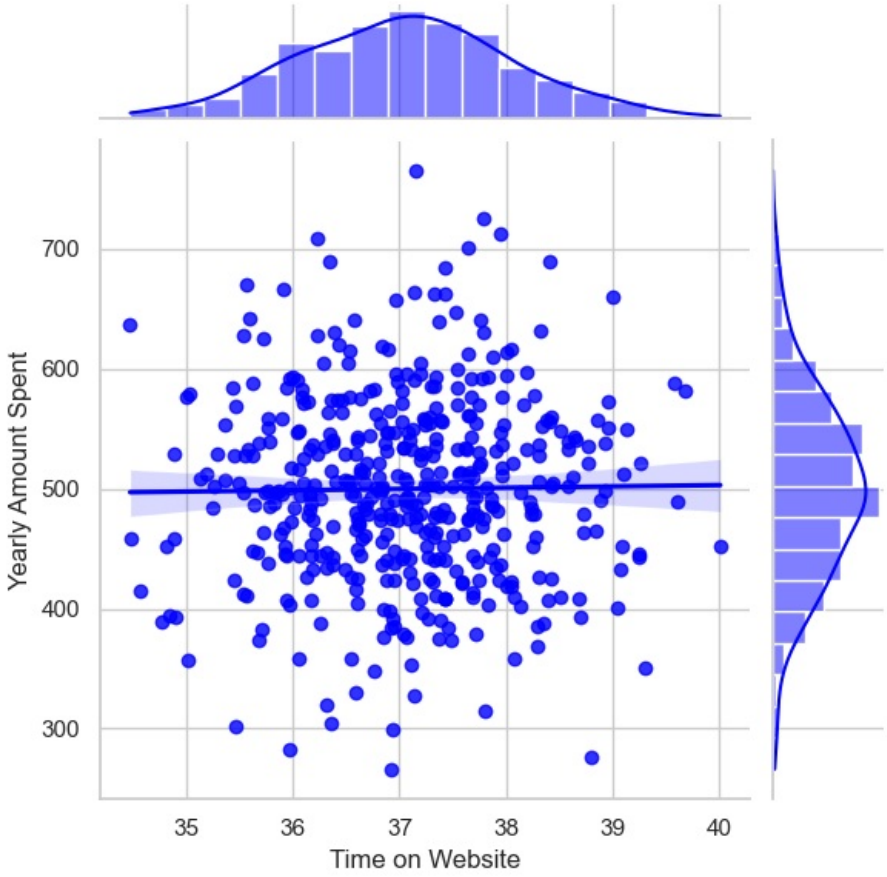
In [27]:
```
sns.histplot(customers['Yearly Amount Spent'], kde=True, color='green')
plt.title("Distribution of Yearly Amount Spent")
plt.xlabel("Yearly Amount Spent")
plt.ylabel("Frequency")
plt.show()
```



Relationships with Target Variable

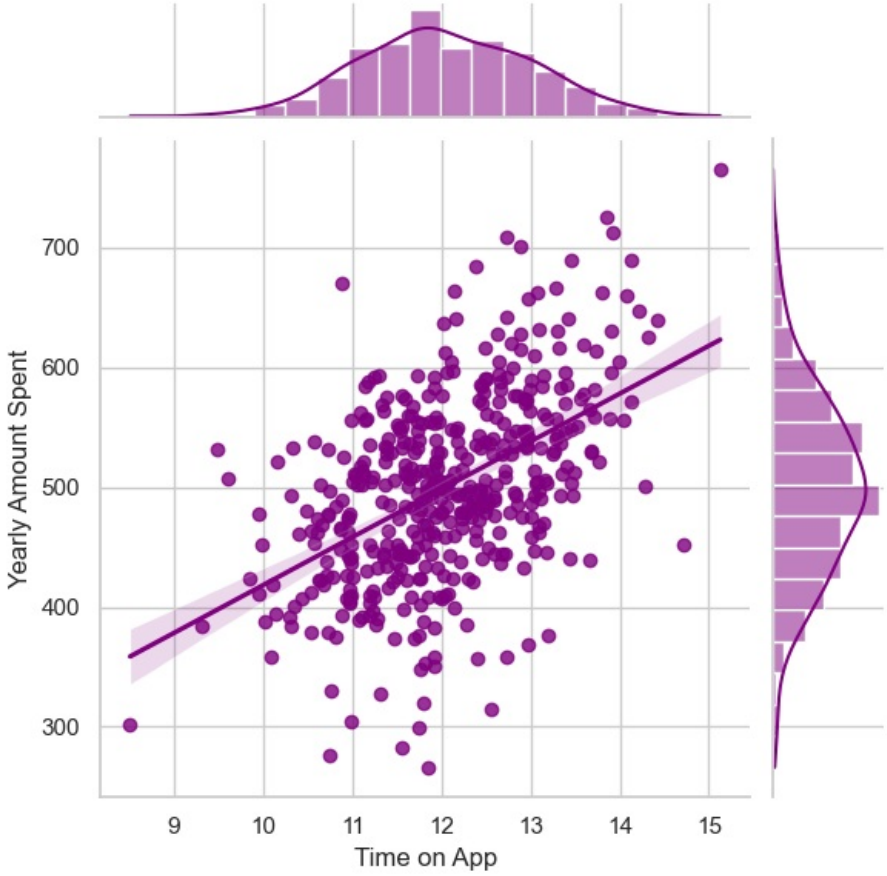
Time on Website vs Yearly Amount Spent

In [31]:
```
sns.jointplot(x='Time on Website', y='Yearly Amount Spent', data=customers, kind='reg', color='blue')
plt.show()
```



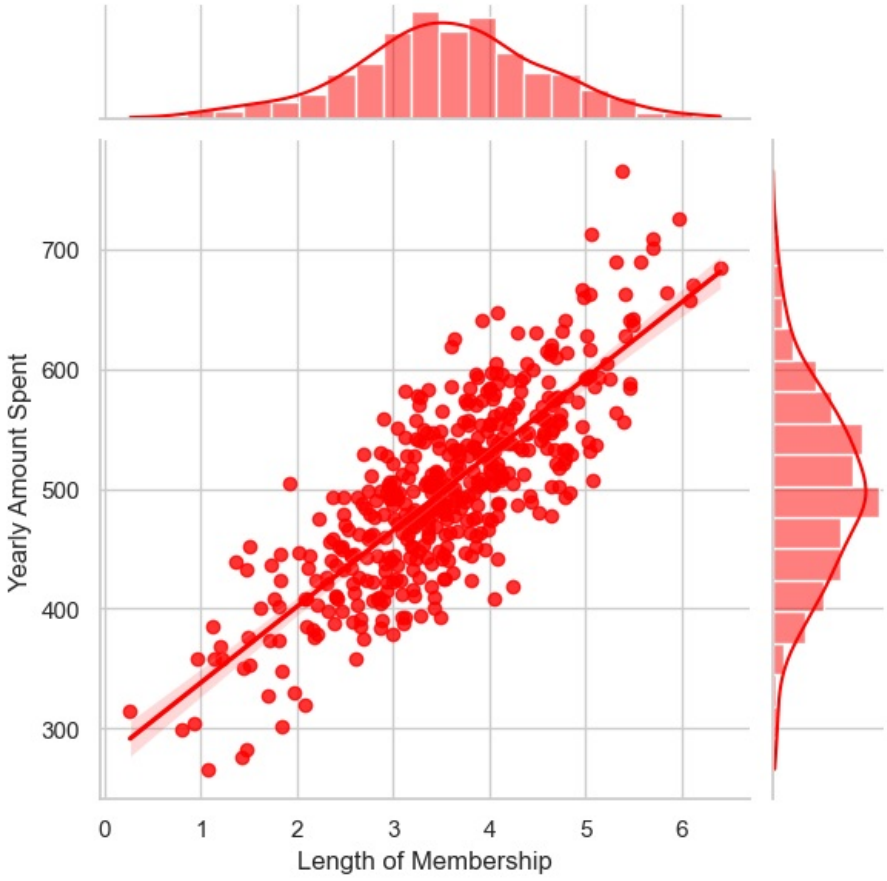
Time on App vs Yearly Amount Spent

In [32]:
```
sns.jointplot(x='Time on App', y='Yearly Amount Spent', data=customers, kind='reg', color='purple')
plt.show()
```



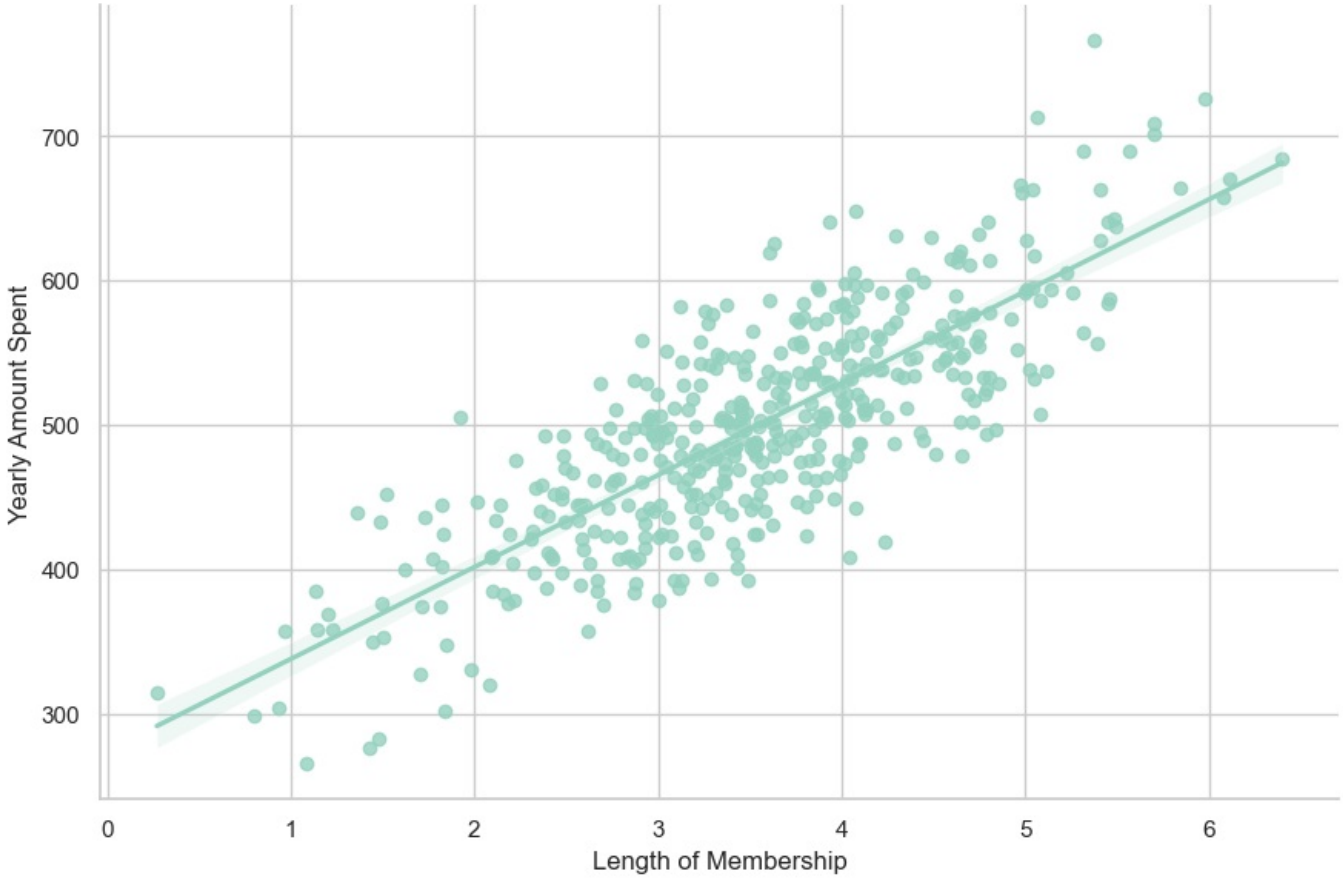
Length of Membership vs Yearly Amount Spent

In [33]:
```
sns.jointplot(x='Length of Membership', y='Yearly Amount Spent', data=customers, kind='reg', color='red')
plt.show()
```



Multivariate Analysis

In [34]:
```
sns.lmplot(x='Length of Membership', y='Yearly Amount Spent', data=customers,
           hue=None, palette='Set1', height=6, aspect=1.5)
plt.show()
```



Summary Statistics by Quartile

In [35]:
```
customers['Membership_Quartile'] = pd.qcut(customers['Length of Membership'], 4)
group_stats = customers.groupby('Membership_Quartile')['Yearly Amount Spent'].describe()
print(group_stats)
```

```
                     count        mean        std         min         25%  \
Membership_Quartile
(0.269, 2.954]       114.0  420.607392  57.324718  266.086341  385.747933
(2.954, 3.534]       113.0  482.109328  46.726605  378.330907  448.340425
(3.534, 4.128]       113.0  519.933013  49.510440  408.640351  486.083425
(4.128, 6.401]       114.0  577.049020  60.405409  418.602742  535.684182

                           50%         75%         max
Membership_Quartile
(0.269, 2.954]       423.682743  457.900061  558.427257
(2.954, 3.534]       483.159721  508.771907  582.491924
(3.534, 4.128]       516.831557  554.003093  647.619456
(4.128, 6.401]       568.096293  613.295800  765.518462
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11300\1662496834.py:2: FutureWarning: The default of observed=False
is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current b
ehavior or observed=True to adopt the future default and silence this warning.
  group_stats = customers.groupby('Membership_Quartile')['Yearly Amount Spent'].describe()
```

Training and Testing Data

In [36]:
```
y = customers['Yearly Amount Spent']
X = customers[['Avg. Session Length', 'Time on App','Time on Website', 'Length of Membership']]
```

split the dataset into train and test

In [37]:
```
from sklearn.model_selection import train_test_split
```

In [38]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

Training the Model

In [41]:
```
# Import LinearRegression from sklearn.linear_model
from sklearn.linear_model import LinearRegression
```

In [42]:
```
# Create an instance of a LinearRegression() model named lm
lm = LinearRegression()
```

In [43]:
```
# Train/fit lm on the training data.
lm.fit(X_train,y_train)
```

Out[43]:

LinearRegression

LinearRegression()

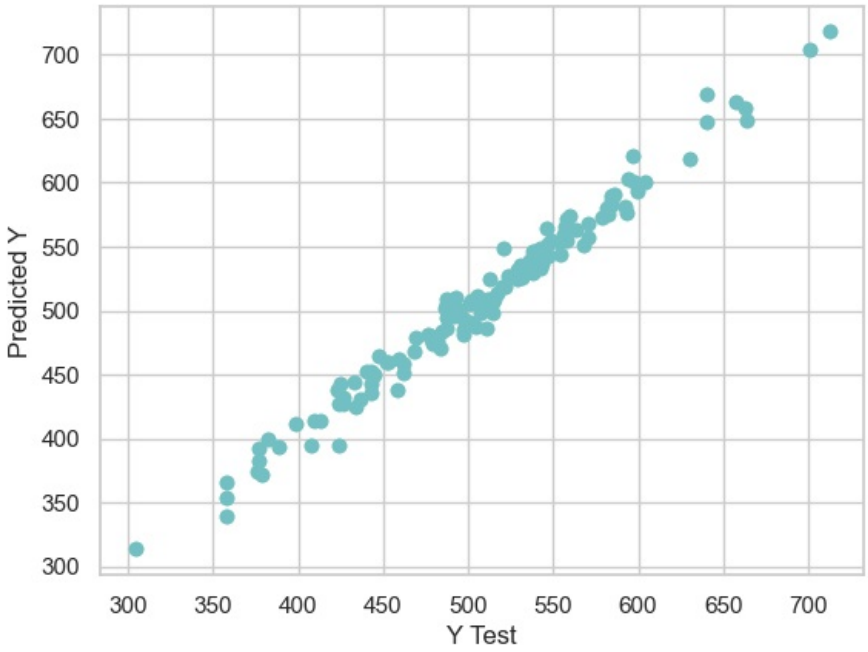
In [44]:
```
# The coefficients
print('Coefficients: \n', lm.coef_)
```

```
Coefficients:
 [25.8679068  39.5435202   0.62161736 61.48119726]
```

Predicting Test Data

In [45]:
```
predictions = lm.predict( X_test)
```

In [47]:
```
# Create a scatterplot of the real test values versus the predicted values.
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.show()
```



Evaluating the Model

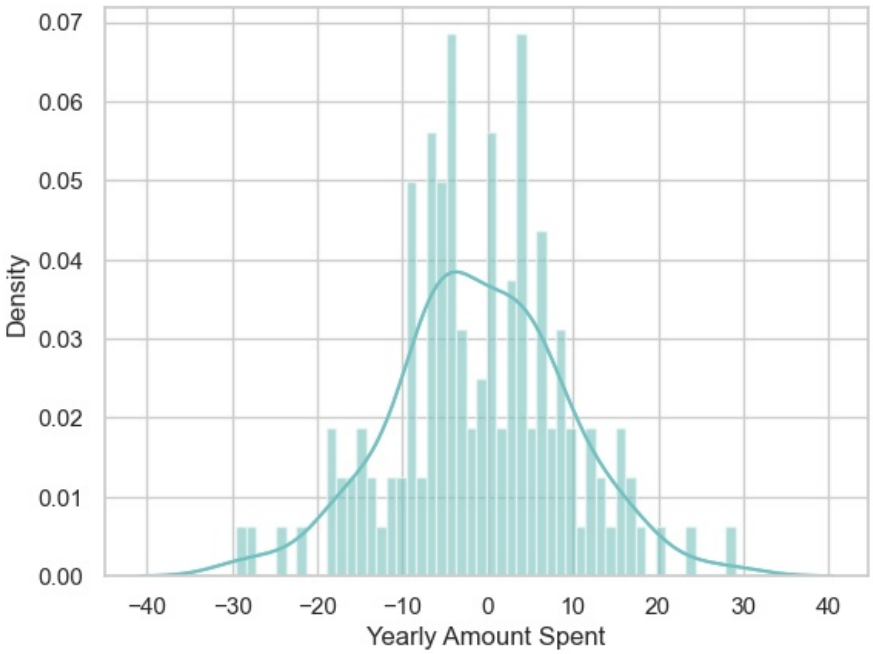
In [48]:
```
# calculate the metrics
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 8.042176495294282
MSE: 103.99375361008293
RMSE: 10.197732768124634
```

Residuals

In [50]:
```
sns.distplot((y_test-predictions),bins=50);
plt.show()
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11300\3045533399.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot((y_test-predictions),bins=50);
```



Random Forest

In [56]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np
```

In [57]:
```
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

Out[57]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

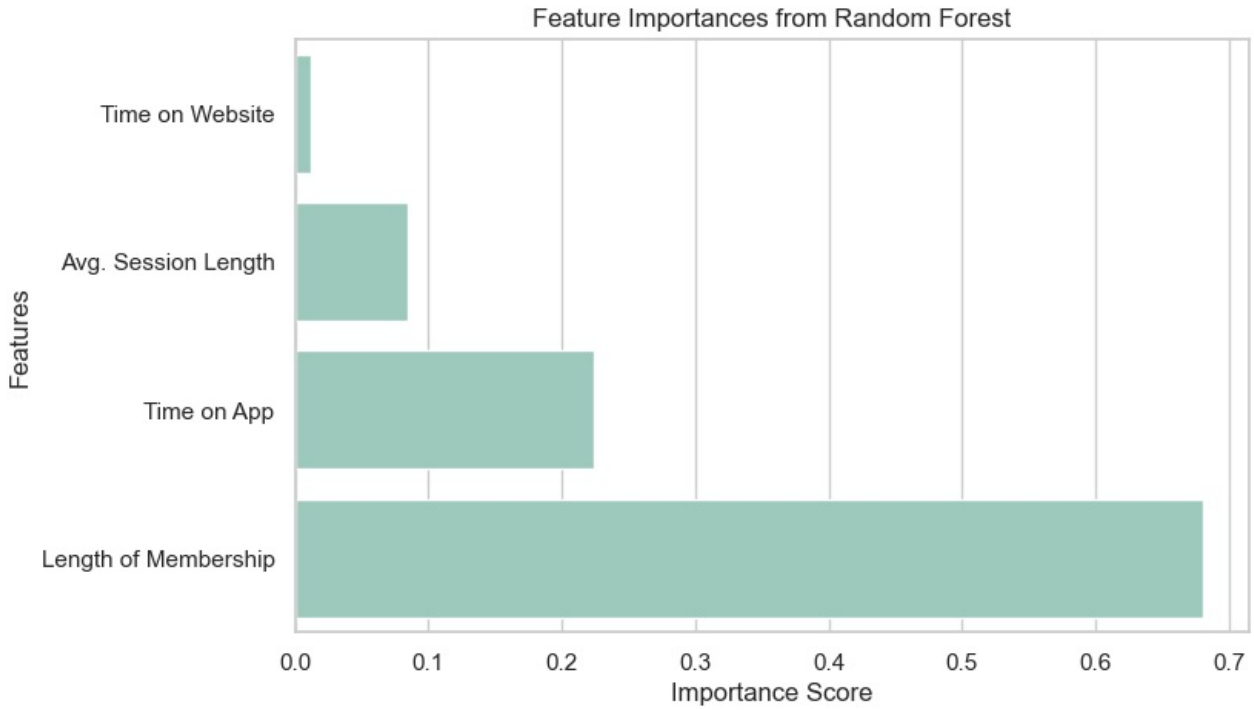
Predictions and Evaluation

In [58]:
```
rf_predictions = rf.predict(X_test)

# Evaluation Metrics
rf_mae = mean_absolute_error(y_test, rf_predictions)
rf_mse = mean_squared_error(y_test, rf_predictions)
rf_rmse = np.sqrt(rf_mse)

print(f"Random Forest MAE: {rf_mae}")
print(f"Random Forest MSE: {rf_mse}")
print(f"Random Forest RMSE: {rf_rmse}")
```

```
Random Forest MAE: 14.030867244335733
Random Forest MSE: 342.2750813069837
Random Forest RMSE: 18.500677860742933
```

Feature Importance

In [59]:
```
feature_importances = pd.Series(rf.feature_importances_, index=X.columns)
plt.figure(figsize=(8, 5))
sns.barplot(x=feature_importances.sort_values(), y=feature_importances.sort_values().index)
plt.title("Feature Importances from Random Forest")
plt.xlabel("Importance Score")
plt.ylabel("Features")
plt.show()
```



Conclusion

The original question is, do we focus our effort on mobile app or website development? Or maybe that doesn't even really matter, and Membership Time is what is really important. From the two models, Linear Regression outperformed Random Forest on all metrics: Lower errors across MAE, MSE, and RMSE. This suggests the relationship between your features and target variable is fairly linear, and a complex model like Random Forest may be overfittin Let's interpret the coefficients at all to get an idea.

In [60]:
```
coeffecients = pd.DataFrame(lm.coef_,X.columns)
coeffecients.columns = ['Coeffecient']
coeffecients
```

Out[60]:

	Coeffecient
Avg. Session Length	25.867907
Time on App	39.543520
Time on Website	0.621617
Length of Membership	61.481197

Interpretating the coefficients:

Interpreting the coefficients:

Holding all other features fixed, a 1 unit increase in **Avg. Session Length** is associated with an **increase of 25.86 total dollars spent**.

- Holding all other features fixed, a 1 unit increase in **Time on App** is associated with an **increase of 39.54 total dollars spent**.
- Holding all other features fixed, a 1 unit increase in **Time on Website** is associated with an **increase of 0.62 total dollars spent**.
- Holding all other features fixed, a 1 unit increase in **Length of Membership** is associated with an **increase of 61.48 total dollars spent**.

should the company focus more on their mobile app or on their website?

Time on App 39.54 dollars and over:
Customers who spend more time on the mobile app tend to spend significantly more. For each extra unit of time (likely in minutes), there's a 39.54 dollars increase in annual spending — the second-highest effect after Length of Membership.

Time on Website +0.62 dollars: In contrast, more time on the website barely affects spending — a $0.62 increase per unit time is almost negligible.

Avg. Session Length +25.86 dollars and Length of Membership (+$61.48): These are also strong predictors, but not tied specifically to channel preference (app vs. website).

Recommendation to the Company

Focus on improving and investing in the mobile app.

Why?

It has a much stronger positive correlation with spending compared to the website.

Small increases in app engagement lead to big spending gains.

Website time has a very weak impact on purchases, so further optimizing it may not yield much return.

In [ ]: