In [5]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns
from sklearn import metrics

print("🏠 House Price Prediction Using Linear Regression Model from scikit-learr
print("☕ Grab a cup of coffee and watch the magic happen!")

# Load the dataset
df = pd.read_csv('EcommerceCustomers.csv')
pd.set_option('display.max_columns', None)
print(df.head())
x = df[['Avg. Session Length', 'Time on App','Time on Website', 'Length of Membe
y = df['Yearly Amount Spent']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random
lrm = LinearRegression()
lrm.fit(x_train, y_train)

# Evaluate the model
print("\n🔍 Model evaluation:")
print(f"🟢 Intercept (constant term): {lrm.intercept_:.2f}")
print("🟢 Coefficients for each feature:")

# Create a DataFrame for coefficients
coeff_df = pd.DataFrame(lrm.coef_, x.columns, columns=['Coefficient'])
print(coeff_df)
print("\n🔎 Interpretation of coefficients:")
for feature, coef in zip(x.columns, lrm.coef_):
    print(f"🔷 A one unit increase in '{feature}' is associated with an increase

predictions = lrm.predict(x_test)
print("\n🔎 House price predictions:")
print(predictions)
```

```
🏠 House Price Prediction Using Linear Regression Model from scikit-learn...
☕ Grab a cup of coffee and watch the magic happen!
                              Email  \
0      mstephenson@fernandez.com
1              hduke@hotmail.com
2               pallen@yahoo.com
3        riverarebecca@gmail.com
4  mstephens@davidson-herman.com

                                              Address            Avatar  \
0       835 Frank Tunnel\nWrightmouth, MI 82180-9605            Violet
1     4547 Archer Common\nDiazchester, CA 06566-8576         DarkGreen
2  24645 Valerie Unions Suite 582\nCobbborough, D...            Bisque
3   1414 David Throughway\nPort Jason, OH 22070-1220       SaddleBrown
4  14023 Rodriguez Passage\nPort Jacobville, PR 3...  MediumAquaMarine

   Avg. Session Length  Time on App  Time on Website  Length of Membership  \
0            34.497268    12.655651        39.577668              4.082621
1            31.926272    11.109461        37.268959              2.664034
2            33.000915    11.330278        37.110597              4.104543
3            34.305557    13.717514        36.721283              3.120179
4            33.330673    12.795189        37.536653              4.446308

   Yearly Amount Spent
0           587.951054
1           392.204933
2           487.547505
3           581.852344
4           599.406092

🔍 Model evaluation:
🟢 Intercept (constant term): -1047.98
🟢 Coefficients for each feature:
                      Coefficient
Avg. Session Length     25.912259
Time on App             38.508126
Time on Website          0.288716
Length of Membership    61.161610

🔎 Interpretation of coefficients:
🔷 A one unit increase in 'Avg. Session Length' is associated with an increase of
$25.91 in the house price.
🔷 A one unit increase in 'Time on App' is associated with an increase of $38.51
in the house price.
🔷 A one unit increase in 'Time on Website' is associated with an increase of $0.
29 in the house price.
🔷 A one unit increase in 'Length of Membership' is associated with an increase o
f $61.16 in the house price.

🔎 House price predictions:
[456.54286407 403.04038845 409.4733783  591.19661352 589.78507712
 548.65253346 577.22540667 714.97215336 473.60765513 545.80428474
 338.10305743 500.26666332 552.89535503 409.57836003 764.88121506
 545.59508474 692.68128903 507.24854118 572.83232081 572.96763324
 397.66463172 554.77893276 458.29897396 482.60971744 558.96461845
 413.11613954 532.08956315 377.91990213 534.80934537 448.04343602
 595.27120684 666.70145306 511.81420024 572.88687124 504.99052047
 565.20573493 460.22874533 449.68777644 422.84378902 456.61172298
 597.67665595 449.88789011 615.14845865 511.67758056 504.18999612
 515.72146994 568.16951591 551.48513998 356.6378314  464.8620116
```

```
 481.60170612 534.06829437 256.58748272 505.21497329 519.95716322
 315.21654441 502.09614923 387.31041674 473.06173507 432.80626366
 539.7086216  589.64106977 751.94849817 558.06069456 523.59234931
 431.86945324 425.51361371 518.75816581 641.45267668 481.82884822
 549.43958683 381.11034334 555.11914644 403.48664165 472.59054176
 501.74908395 473.46525834 456.52751619 554.4243961  702.50615043
 534.72434807 618.85560624 499.94890064 559.24813999 574.63905469
 505.12302239 529.6719221  479.20777206 424.80282544 452.15936464
 525.51286607 556.47232592 425.61830232 588.52893723 490.82444018
 562.38936774 495.61462789 445.50430092 456.58675234 537.97742609
 367.01103033 421.30159286 551.35810069 527.96988425 493.41227515
 495.00368427 519.64738812 461.03646636 528.67738089 442.80460392
 543.01475135 350.27066003 401.38884281 606.52218234 576.70349109
 524.12472044 553.92991615 507.89702237 505.40680844 371.67040153
 342.81954951 633.98898262 523.5392842  532.54259043 574.29573967
 435.50713235 599.56950382 487.08846216 457.54189256 425.06792835
 332.0253581  443.73046773 563.27115892 466.20350336 463.54617144
 381.44663484 411.93032149 473.50490766 573.08604675 417.67287092
 543.38703479 547.58495414 547.56984661 450.94665355 561.1959677
 478.3032658  484.22533552 457.55220485 411.74640875 375.68250566
 449.40702218 557.63342169 553.14582142 485.92999788 547.94903123
 543.1115209  451.86394232 532.29902785 548.49324648 445.98203326
 429.88352809 486.61675893 609.72814219 569.93904011 441.12529378]
```

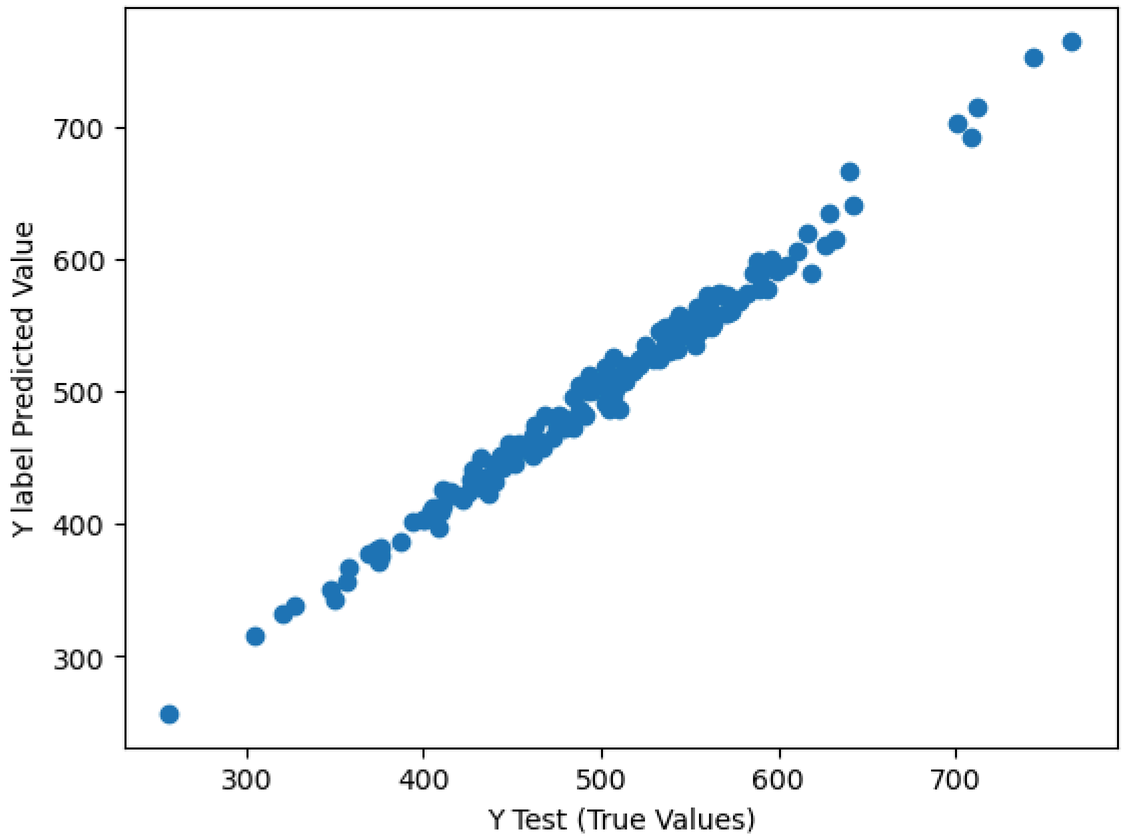
In [7]:

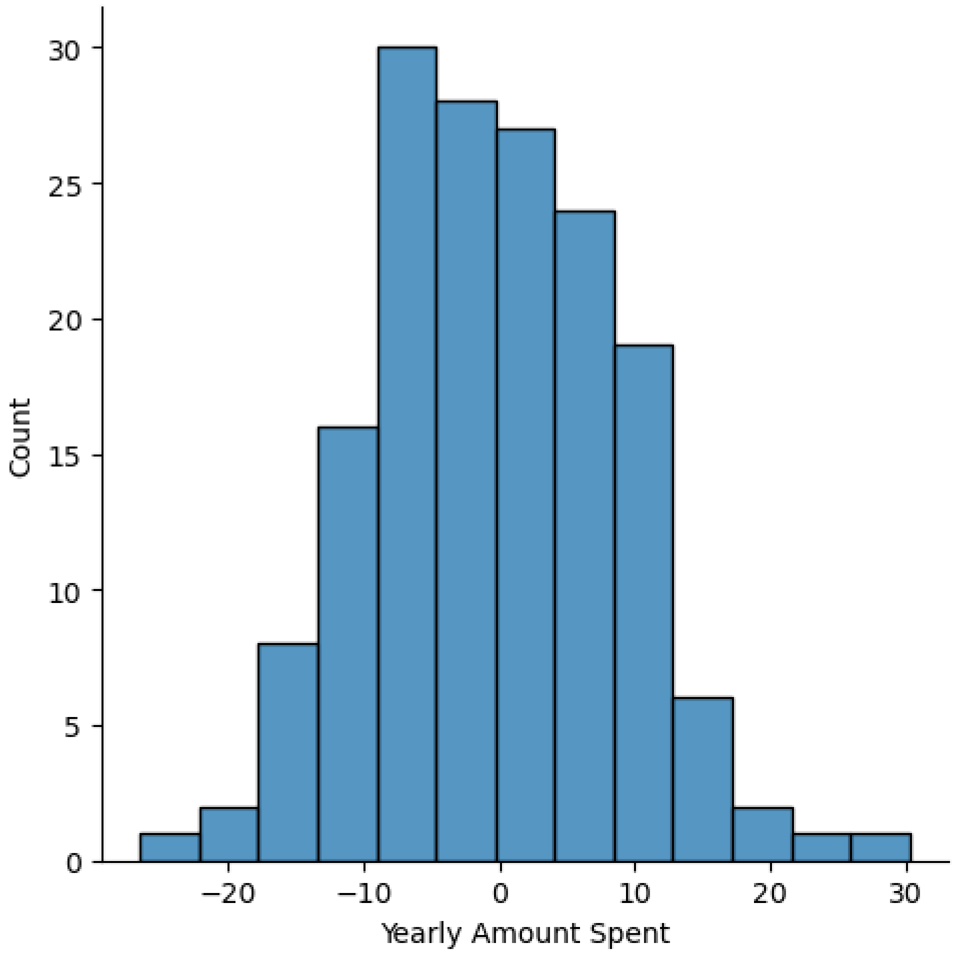
```
# To know how far are we from the actual price compared to predictions using the
# we us can scatter plot from matplotlib with y_test which never used to train a
plt.xlabel('Y Test (True Values)')
plt.ylabel('Y label Predicted Value')
print(plt.scatter(y_test, predictions))

# Print the Histogram of residuals; I see it is normally distributed which is go
print(sns.displot((y_test - predictions)))

# The three common Regression evaluation metrics are Mean Absolute Error(MAE), M
# The lower we minimize the error the best the model
print('MAE')
print(
    metrics.mean_absolute_error(y_test, predictions)
)
print('MSE')
print(metrics.mean_squared_error(y_test, predictions))
print('Root MSE')
print(np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
<matplotlib.collections.PathCollection object at 0x00000279905C5160>
<seaborn.axisgrid.FacetGrid object at 0x00000279905C6CC0>
MAE
7.294546588331314
MSE
81.90726984520933
Root MSE
9.050263523522911
```

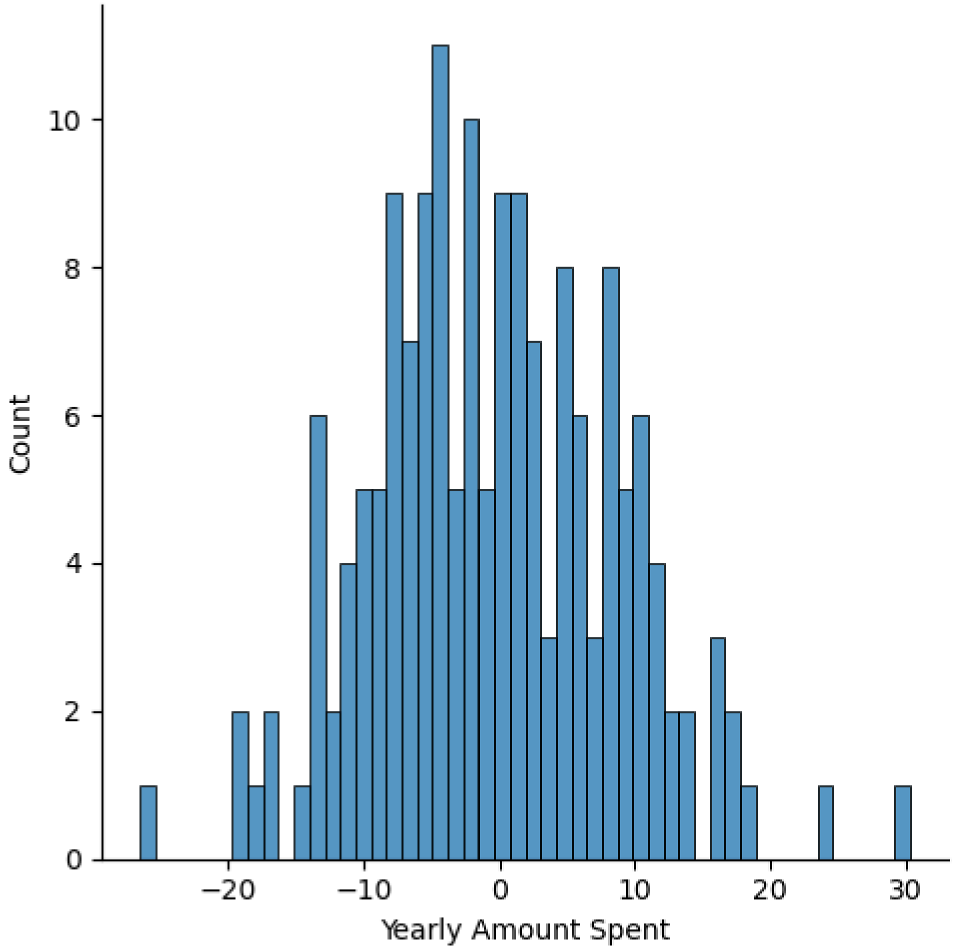




In [15]:
```
print('Residuals')
sns.displot(y_test-predictions, bins=50)
```

```
Residuals
```

Out[15]: <seaborn.axisgrid.FacetGrid at 0x2799162f5c0>



In [ ]: