```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np
```

```
df = pd.read_excel('/content/ENB2012_data.xlsx')
print("Current column names:", df.columns.tolist())
df.head()
```

```
Current column names: ['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'Y1', 'Y2']
```

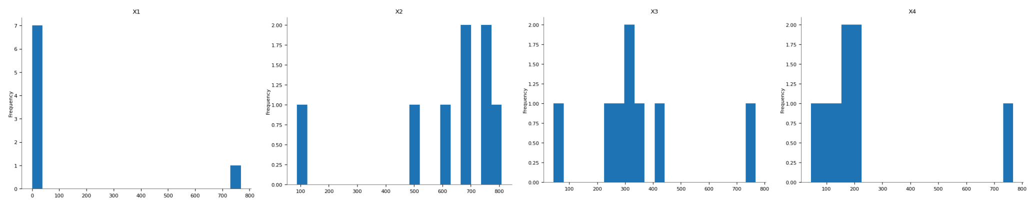
	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	15.55	21.33
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	15.55	21.33
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	15.55	21.33
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	15.55	21.33
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	20.84	28.28

Next steps: Generate code with df New interactive sheet

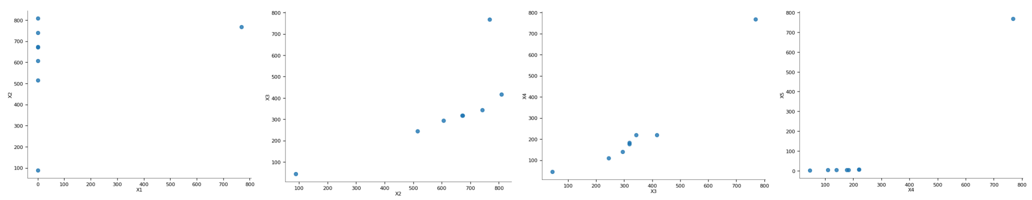
```
df.describe()
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.00000	768.000000	768.0000
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375	2.81250	22.307195	24.5877
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221	1.55096	10.090204	9.5133
min	0.620000	514.500000	245.000000	110.250000	3.50000	2.000000	0.000000	0.00000	6.010000	10.9000
25%	0.682500	606.375000	294.000000	140.875000	3.50000	2.750000	0.100000	1.75000	12.992500	15.6200
50%	0.750000	673.750000	318.500000	183.750000	5.25000	3.500000	0.250000	3.00000	18.950000	22.0800
75%	0.830000	741.125000	343.000000	220.500000	7.00000	4.250000	0.400000	4.00000	31.667500	33.1325
max	0.980000	808.500000	416.500000	220.500000	7.00000	5.000000	0.400000	5.00000	43.100000	48.0300

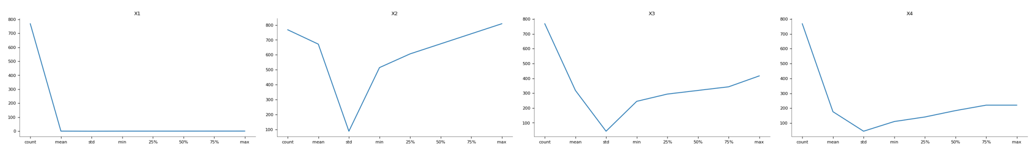
Distributions



2-d distributions



Values

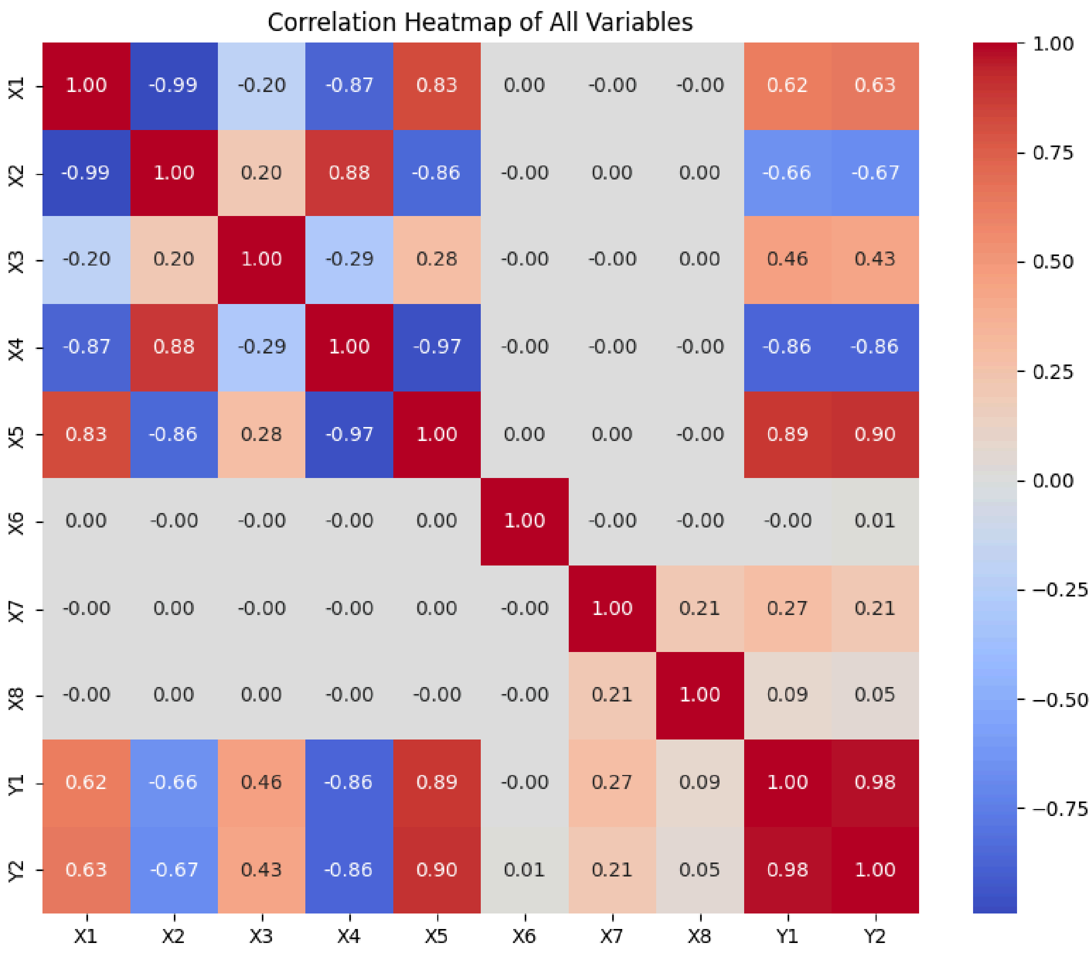


```
df.isnull().sum()
```

	0
X1	0
X2	0
X3	0
X4	0
X5	0
X6	0
X7	0
X8	0
Y1	0
Y2	0

dtype: int64

```
import seaborn as sns

plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap of All Variables')
plt.show()
```




```
#simple linear regression from independent variable X1 and dependent variable Y1
X = df[['X1']]
y = df['Y1']
```

```
#split into training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

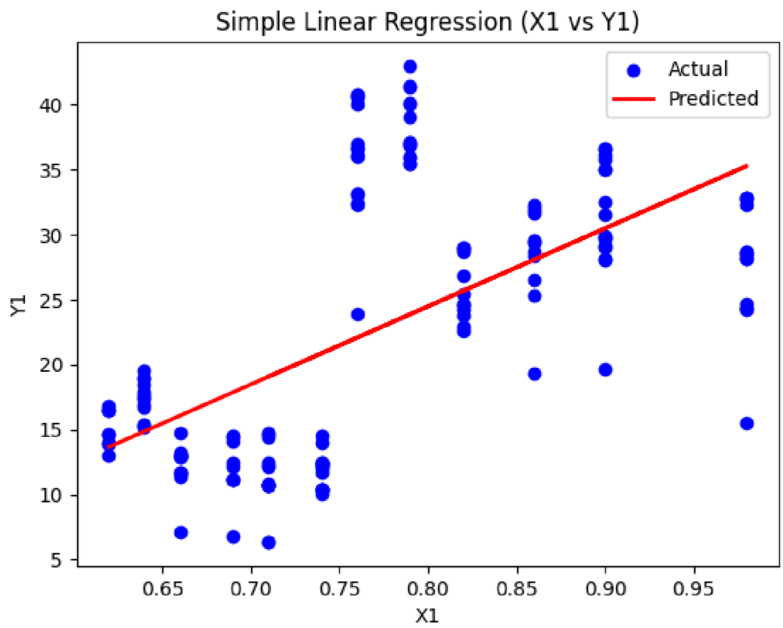
```
y_pred = model.predict(X_test)
```

```
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
```

```
print(f"R-squared: {r2:.2f}")
print(f"Mean Squared Error: {mse:.2f}")
```

```
R-squared: 0.35
Mean Squared Error: 67.72
```

```
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Predicted')
plt.xlabel('X1')
plt.ylabel('Y1')
plt.title('Simple Linear Regression (X1 vs Y1)')
plt.legend()
plt.show()
```

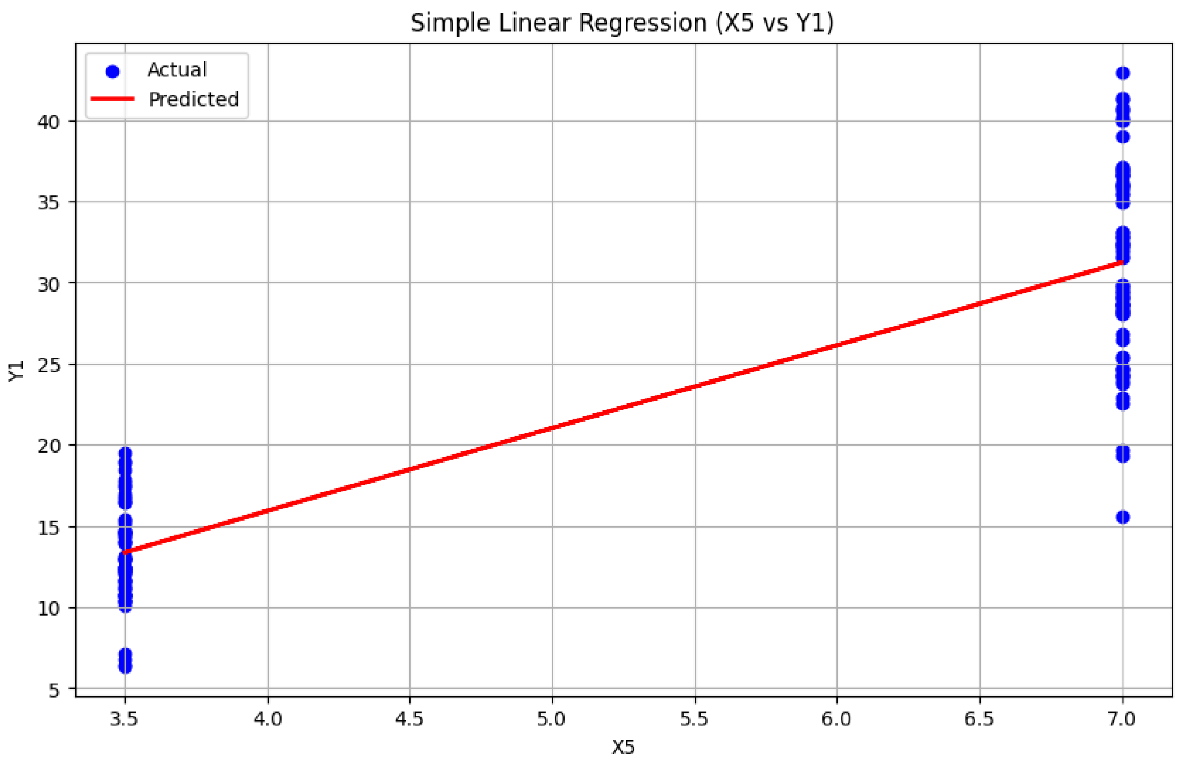



```
# Simple linear regression from independent variable X5 and dependent variable Y
X_new = df[['X5']]
y_new = df['Y1']

X_train_new, X_test_new, y_train_new, y_test_new = train_test_split(X_new, y_new

model_new = LinearRegression()
model_new.fit(X_train_new, y_train_new)

y_pred_new = model_new.predict(X_test_new)

# Evaluate the model
r2_new = r2_score(y_test_new, y_pred_new)
mse_new = mean_squared_error(y_test_new, y_pred_new)
```

```
print(f"R-squared (X5 vs Y1): {r2_new:.2f}")
print(f"Mean Squared Error (X5 vs Y1): {mse_new:.2f}")
```

```
R-squared (X5 vs Y1): 0.79
Mean Squared Error (X5 vs Y1): 21.68
```

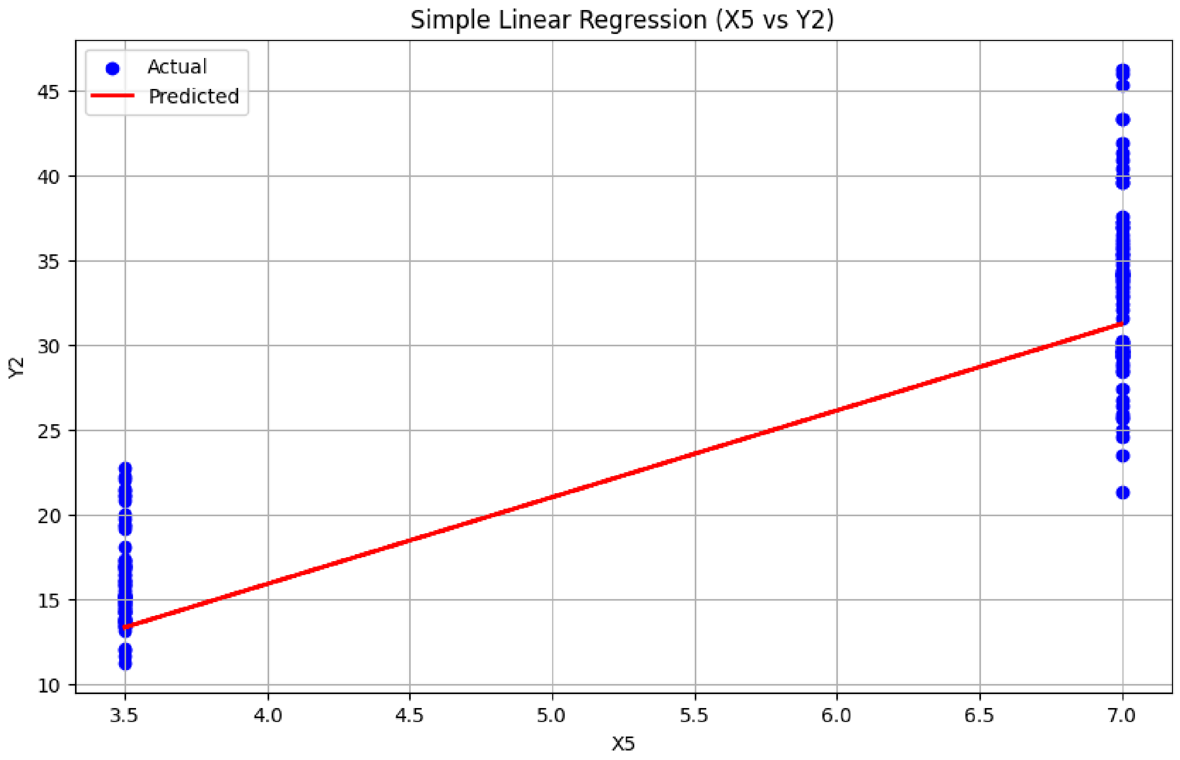
```
plt.figure(figsize=(10, 6))
plt.scatter(X_test_new, y_test_new, color='blue', label='Actual')
plt.plot(X_test_new, y_pred_new, color='red', linewidth=2, label='Predicted')
plt.xlabel('X5')
plt.ylabel('Y1')
plt.title('Simple Linear Regression (X5 vs Y1)')
plt.legend()
plt.grid(True)
plt.show()
```




```
# Simple linear regression from independent variable X5 and dependent variable Y2
X_x5 = df[['X5']]
y_y2 = df['Y2']

X_train_x5, X_test_x5, y_train_y2, y_test_y2 = train_test_split(X_x5, y_y2, test_

model_x5y2 = LinearRegression()
model_x5y2.fit(X_train_x5, y_train_y2)

y_pred_y2 = model_new.predict(X_test_x5)

# Evaluate the model
r2_x5y2 = r2_score(y_test_y2, y_pred_y2)
mse_x5y2 = mean_squared_error(y_test_y2, y_pred_y2)

print(f"R-squared (X5 vs Y2): {r2_x5y2:.2f}")
print(f"Mean Squared Error (X5 vs Y2): {mse_x5y2:.2f}")
```

```
R-squared (X5 vs Y2): 0.74
Mean Squared Error (X5 vs Y2): 23.71
```

```
plt.figure(figsize=(10, 6))
plt.scatter(X_test_x5, y_test_y2, color='blue', label='Actual')
plt.plot(X_test_x5, y_pred_y2, color='red', linewidth=2, label='Predicted')
plt.xlabel('X5')
plt.ylabel('Y2')
plt.title('Simple Linear Regression (X5 vs Y2)')
plt.legend()
plt.grid(True)
plt.show()
```

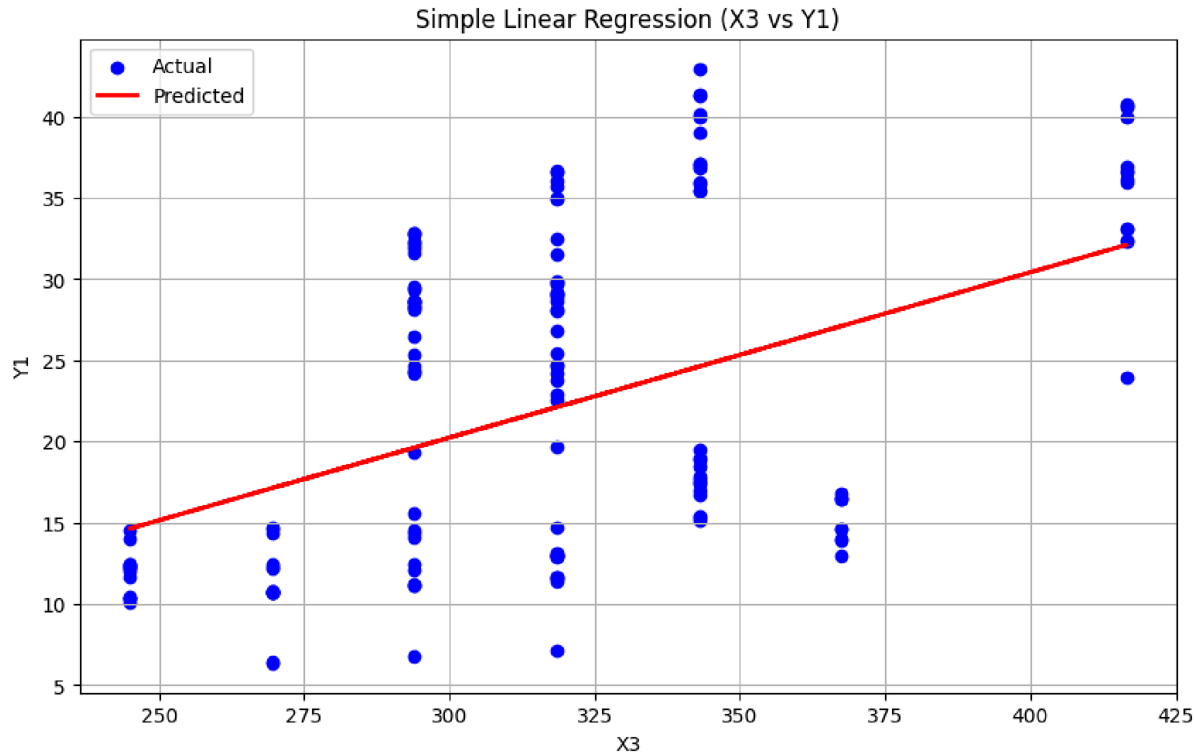



```
# Simple linear regression from independent variable X3 and dependent variable Y
X_x3 = df[['X3']]
y_y1 = df['Y1']

X_train_x3, X_test_x3, y_train_y1, y_test_y1 = train_test_split(X_x3, y_y1, test_

model_new = LinearRegression()
model_new.fit(X_train_x3, y_train_y1)

y_pred_y1 = model_new.predict(X_test_x3)

# Evaluate the model
r2_x3y1 = r2_score(y_test_y1, y_pred_y1)
mse_x3y1 = mean_squared_error(y_test_new, y_pred_y1)

print(f"R-squared (X3 vs Y1): {r2_new:.2f}")
print(f"Mean Squared Error (X3 vs Y1): {mse_new:.2f}")
```

```
R-squared (X3 vs Y1): 0.79
Mean Squared Error (X3 vs Y1): 21.68
```

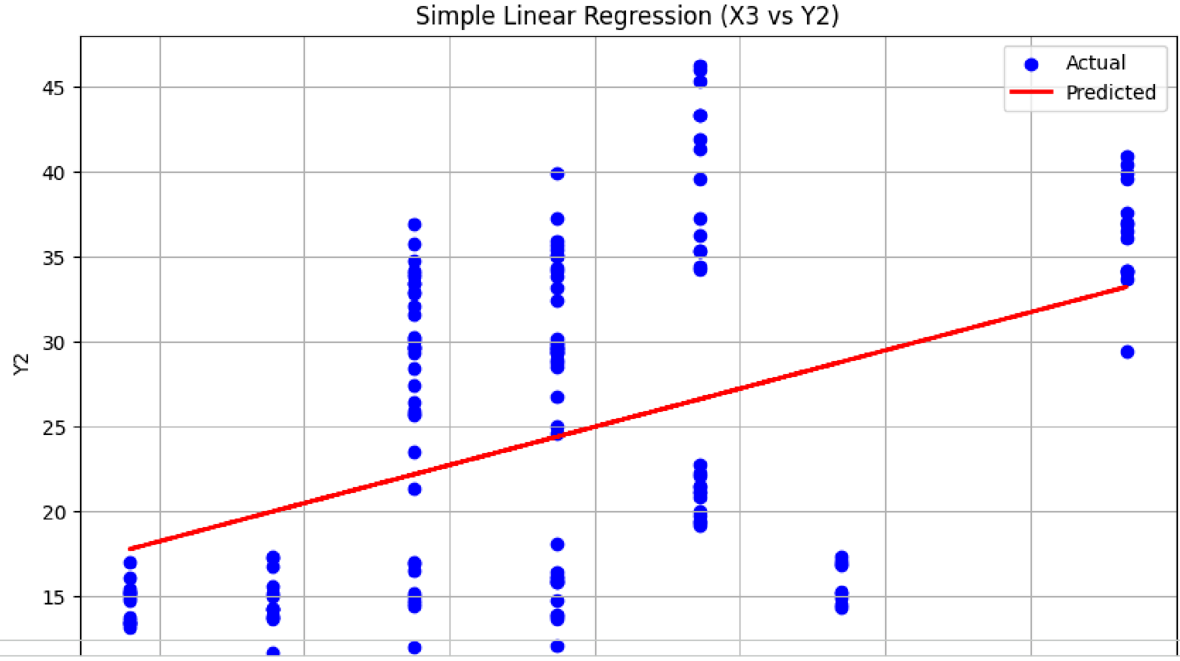
```
plt.figure(figsize=(10, 6))
plt.scatter(X_test_x3, y_test_y1, color='blue', label='Actual')
plt.plot(X_test_x3, y_pred_y1, color='red', linewidth=2, label='Predicted')
plt.xlabel('X3')
plt.ylabel('Y1')
```

```
plt.title('Simple Linear Regression (X3 vs Y1)')
plt.legend()
plt.grid(True)
plt.show()
```




```
# Simple linear regression from independent variable X3 and dependent variable Y2
X_x3 = df[['X3']]
y_y2 = df['Y2']

X_train_x3, X_test_x3, y_train_y2, y_test_y2 = train_test_split(X_x3, y_y2, test_

model_x3y2 = LinearRegression()
model_x3y2.fit(X_train_x3, y_train_y2)

y_pred_y2 = model_x3y2.predict(X_test_x3)

# Evaluate the model
r2_x3y2 = r2_score(y_test_y2, y_pred_y2)
mse_x3y2 = mean_squared_error(y_test_y2, y_pred_y2)

print(f"R-squared (X3 vs Y2): {r2_x3y2:.2f}")
print(f"Mean Squared Error (X3 vs Y2): {mse_x3y2:.2f}")

plt.figure(figsize=(10, 6))
plt.scatter(X_test_x3, y_test_y2, color='blue', label='Actual')
plt.plot(X_test_x3, y_pred_y2, color='red', linewidth=2, label='Predicted')
plt.xlabel('X3')
plt.ylabel('Y2')
plt.title('Simple Linear Regression (X3 vs Y2)')
plt.legend()
plt.grid(True)
plt.show()
```

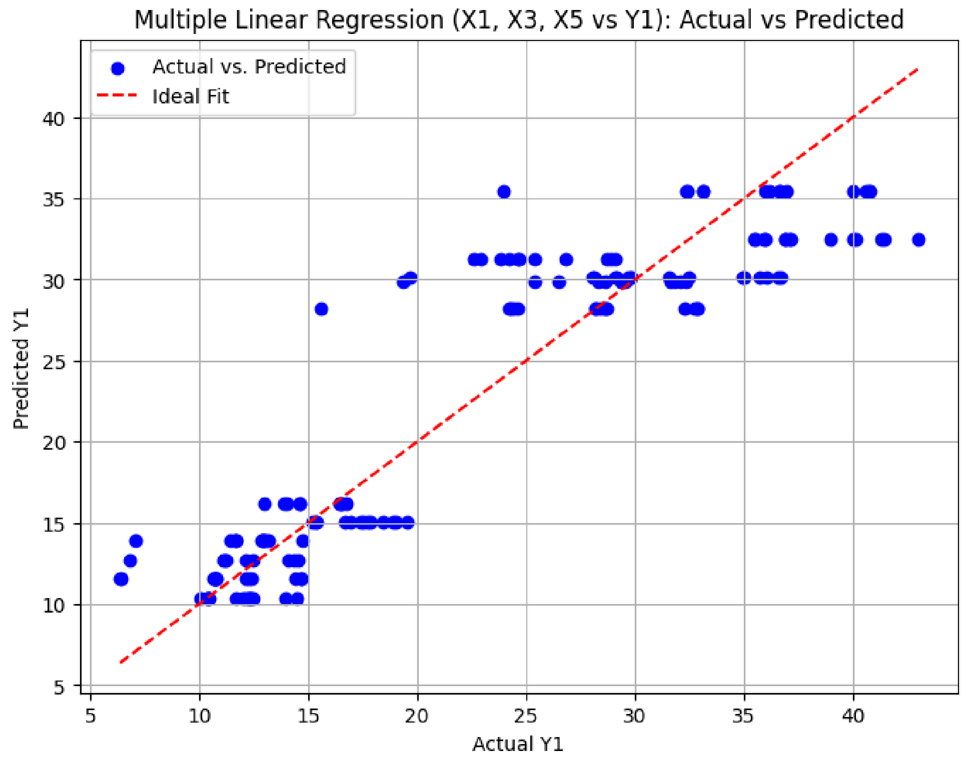
```
R-squared (X3 vs Y2): 0.23
Mean Squared Error (X3 vs Y2): 71.72
```




```
#Trying multiple linear model due to presence of multiple independent variable
# Multiple linear regression with X1,X3, X5 as features and Y1 as target
X_multi = df[['X1', 'X3', 'X5']]
y_multi = df['Y1']

# Split the data into training and testing
X_train_multi, X_test_multi, y_train_multi, y_test_multi = train_test_split(X_mu

# Create a Linear Regression model
model_multi = LinearRegression()

# Train the model
model_multi.fit(X_train_multi, y_train_multi)

# Make predictions on the test set
y_pred_multi = model_multi.predict(X_test_multi)

# Evaluate the model
r2_multi = r2_score(y_test_multi, y_pred_multi)
mse_multi = mean_squared_error(y_test_multi, y_pred_multi)

print(f"R-squared (Multiple Regression X1, X3, X5 vs Y1): {r2_multi:.2f}")
print(f"Mean Squared Error (Multiple Regression X1, X3, X5 vs Y1): {mse_multi:.2
```

```
R-squared (Multiple Regression X1, X3, X5 vs Y1): 0.85
Mean Squared Error (Multiple Regression X1, X3, X5 vs Y1): 15.60
```

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test_multi, y_pred_multi, color='blue', label='Actual vs. Predicte
plt.plot([min(y_test_multi), max(y_test_multi)], [min(y_test_multi), max(y_test_
plt.xlabel('Actual Y1')
plt.ylabel('Predicted Y1')
plt.title('Multiple Linear Regression (X1, X3, X5 vs Y1): Actual vs Predicted')
plt.legend()
plt.grid(True)
plt.show()
```


Multiple Linear Regression (X1, X3, X5 vs Y1): Actual vs Predicted
Actual vs. Predicted
Ideal Fit
Predicted Y1
Actual Y1
5
10
15
20
25
30
35
40