```python
# Step 1: Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Step 2: Load dataset
df = pd.read_csv("Fish.csv")

# Display first few rows
df.head()
```

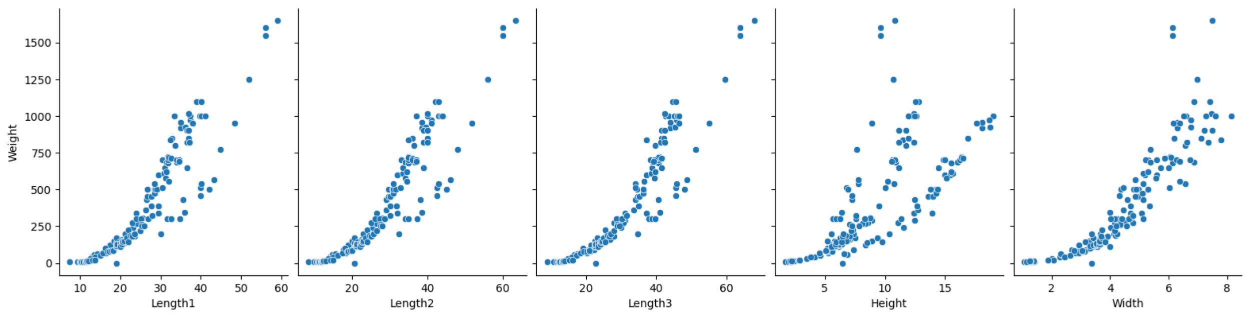

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 159,\n  \"fields\": [\
n    {\n      \"column\": \"Species\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 7,\n
\"samples\": [\n          \"Bream\",\n          \"Roach\",\n
\"Pike\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Weight\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 357.9783165508931,\n        \"min\": 0.0,\n        \"max\":
1650.0,\n        \"num_unique_values\": 101,\n        \"samples\": [\n
770.0,\n          51.5,\n          197.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Length1\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
9.996441210553128,\n        \"min\": 7.5,\n        \"max\": 59.0,\n
\"num_unique_values\": 116,\n        \"samples\": [\n          36.9,\n
26.5,\n          22.1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Length2\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 10.716328098884247,\n        \"min\": 8.4,\n
\"max\": 63.4,\n        \"num_unique_values\": 93,\n
\"samples\": [\n          14.7,\n          18.8,\n          19.6\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Length3\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
11.610245832690964,\n        \"min\": 8.8,\n        \"max\": 68.0,\n
\"num_unique_values\": 124,\n        \"samples\": [\n          39.2,\n
27.2,\n          23.1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Height\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 4.286207619968867,\n        \"min\": 1.7284,\n
\"max\": 18.957,\n        \"num_unique_values\": 154,\n
\"samples\": [\n          15.438,\n          7.293,\n          2.8728\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"Width\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1.6858038699921671,\n        \"min\": 1.0476,\n
\"max\": 8.142,\n        \"num_unique_values\": 152,\n
\"samples\": [\n          3.1571,\n          1.3936,\n
3.6835\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```python
# Step 3: Basic exploration
print(df.info())
print(df.describe())

# Check for missing values
print(df.isnull().sum())

# Visualize relationships
sns.pairplot(df, x_vars=['Length1', 'Length2', 'Length3', 'Height',
'Width'], y_vars='Weight', height=4, aspect=0.8)
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159 entries, 0 to 158
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Species  159 non-null    object 
 1   Weight   159 non-null    float64
 2   Length1  159 non-null    float64
 3   Length2  159 non-null    float64
 4   Length3  159 non-null    float64
 5   Height   159 non-null    float64
 6   Width    159 non-null    float64
dtypes: float64(6), object(1)
memory usage: 8.8+ KB
None
           Weight     Length1     Length2     Length3      Height
Width
count  159.000000  159.000000  159.000000  159.000000  159.000000
159.000000
mean   398.326415   26.247170   28.415723   31.227044    8.970994
4.417486
std    357.978317    9.996441   10.716328   11.610246    4.286208
1.685804
min      0.000000    7.500000    8.400000    8.800000    1.728400
1.047600
25%    120.000000   19.050000   21.000000   23.150000    5.944800
3.385650
50%    273.000000   25.200000   27.300000   29.400000    7.786000
4.248500
75%    650.000000   32.700000   35.500000   39.650000   12.365900
```

```
5.584500
max    1650.000000   59.000000   63.400000   68.000000   18.957000
8.142000
Species    0
Weight     0
Length1    0
Length2    0
Length3    0
Height     0
Width      0
dtype: int64
```




```
# Step 4: Select features and label
X = df[['Length1']]   # independent variable
y = df['Weight']      # dependent variable

# Step 5: Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

print("Training samples:", len(X_train))
print("Testing samples:", len(X_test))

Training samples: 127
Testing samples: 32

# Step 6: Train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Print model parameters
print("Intercept:", model.intercept_)
print("Coefficient:", model.coef_)

Intercept: -464.1341160085111
Coefficient: [32.44308998]

# Step 7: Predict on test data
y_pred = model.predict(X_test)
```

```
# Compare actual vs predicted
comparison = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
comparison.head()
```

{"summary":"{\n \"name\": \"comparison\",\n \"rows\": 32,\n \"fields\": [\n   {\n   \"column\": \"Actual\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 383.18096368089505,\n   \"min\": 6.7,\n   \"max\": 1250.0,\n   \"num_unique_values\": 26,\n   \"samples\": [\n   188.0,\n   10.0,\n   78.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Predicted\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 333.07917486475736,\n   \"min\": -162.41337916742657,\n   \"max\": 1222.9065631029289,\n   \"num_unique_values\": 29,\n   \"samples\": [\n   833.5894833079814,\n   346.94313356429666,\n   132.81873967707543\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"comparison"}

```
<google.colab._quickchart_helpers.SectionTitle at 0x7bb9151fc4d0>

from matplotlib import pyplot as plt
_df_0['Actual'].plot(kind='hist', bins=20, title='Actual')
plt.gca().spines[['top', 'right',]].set_visible(False)

from matplotlib import pyplot as plt
_df_1['Predicted'].plot(kind='hist', bins=20, title='Predicted')
plt.gca().spines[['top', 'right',]].set_visible(False)

<google.colab._quickchart_helpers.SectionTitle at 0x7bb9151fc650>

from matplotlib import pyplot as plt
_df_2.plot(kind='scatter', x='Actual', y='Predicted', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)

<google.colab._quickchart_helpers.SectionTitle at 0x7bb9151fe210>

from matplotlib import pyplot as plt
_df_3['Actual'].plot(kind='line', figsize=(8, 4), title='Actual')
plt.gca().spines[['top', 'right']].set_visible(False)

from matplotlib import pyplot as plt
_df_4['Predicted'].plot(kind='line', figsize=(8, 4),
title='Predicted')
plt.gca().spines[['top', 'right']].set_visible(False)

<google.colab._quickchart_helpers.SectionTitle at 0x7bb915707230>

from matplotlib import pyplot as plt
_df_5['index'].plot(kind='hist', bins=20, title='index')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_6['Actual'].plot(kind='hist', bins=20, title='Actual')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_7['Predicted'].plot(kind='hist', bins=20, title='Predicted')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7bb91254f6e0>
```

```
from matplotlib import pyplot as plt
_df_8.plot(kind='scatter', x='index', y='Actual', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_9.plot(kind='scatter', x='Actual', y='Predicted', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7bb9124d55e0>
```

```
from matplotlib import pyplot as plt
_df_10['index'].plot(kind='line', figsize=(8, 4), title='index')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_11['Actual'].plot(kind='line', figsize=(8, 4), title='Actual')
plt.gca().spines[['top', 'right']].set_visible(False)
```

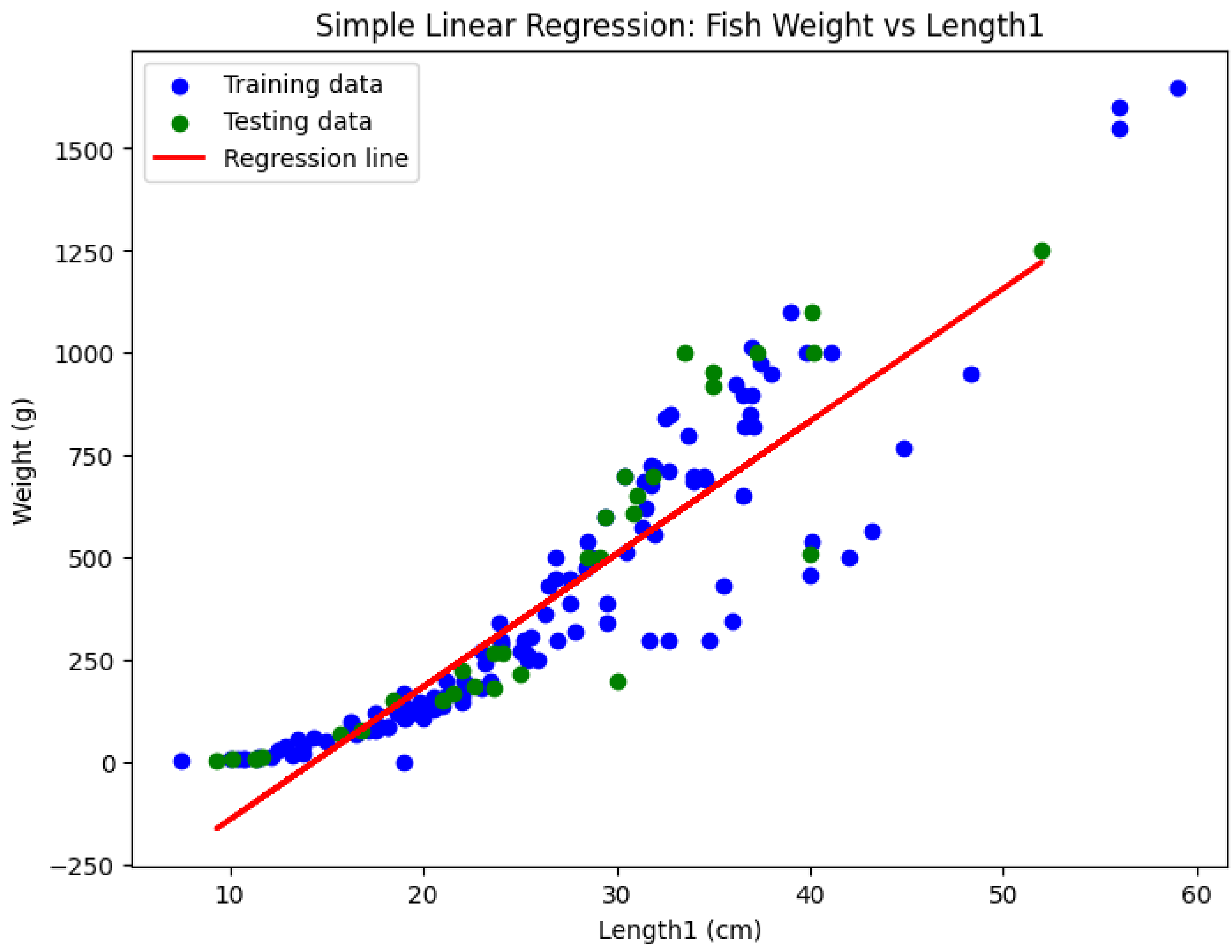
```
from matplotlib import pyplot as plt
_df_12['Predicted'].plot(kind='line', figsize=(8, 4),
title='Predicted')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
# Step 8: Evaluate model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R² Score:", r2)
```

```
Mean Squared Error: 26796.684740821387
R² Score: 0.8116084146869396
```

```
# Step 9: Visualization

plt.figure(figsize=(8,6))
plt.scatter(X_train, y_train, color='blue', label='Training data')
plt.scatter(X_test, y_test, color='green', label='Testing data')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression
line')
plt.xlabel("Length1 (cm)")
plt.ylabel("Weight (g)")
```

```
plt.title("Simple Linear Regression: Fish Weight vs Length1")
plt.legend()
plt.show()
```

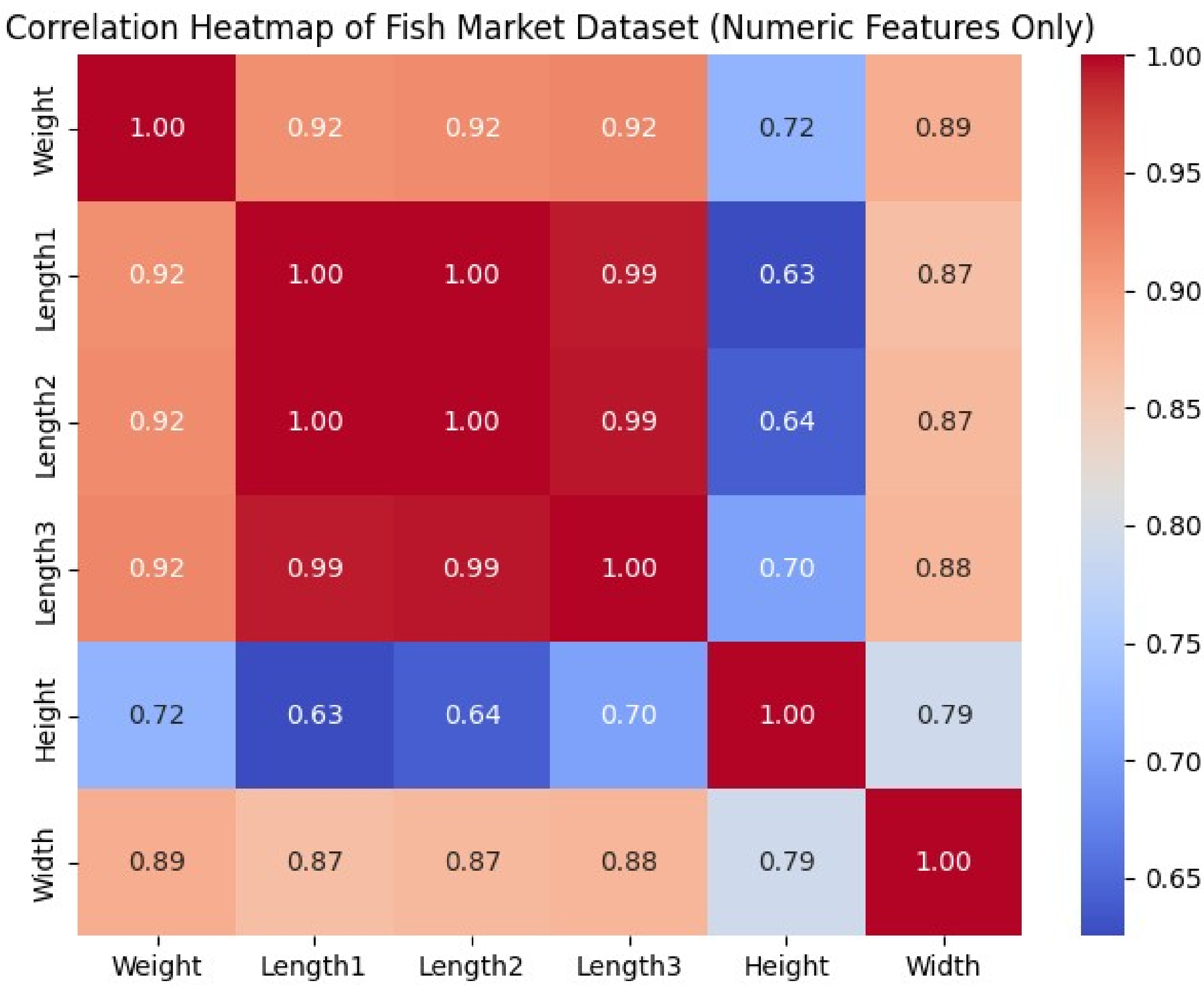



```
# Step 10: Predict a new sample
new_length = np.array([[30]])  # fish length = 30 cm
predicted_weight = model.predict(new_length)
print(f"Predicted Fish Weight for Length 30 cm:
{predicted_weight[0]:.2f} grams")
```

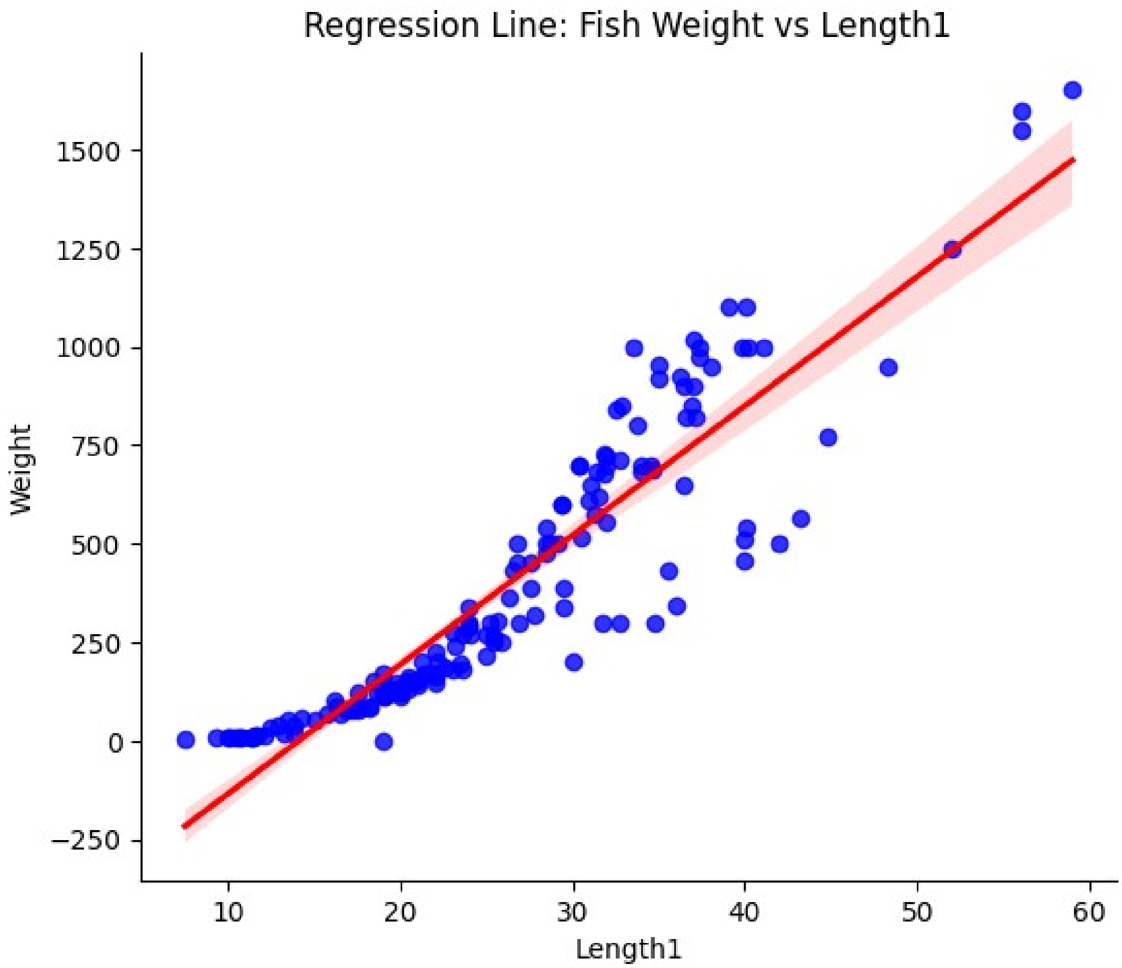
```
Predicted Fish Weight for Length 30 cm: 509.16 grams

/usr/local/lib/python3.12/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but LinearRegression was fitted with feature names
  warnings.warn(
```

```
# Correlation Heatmap (Fixed)
plt.figure(figsize=(8,6))
```

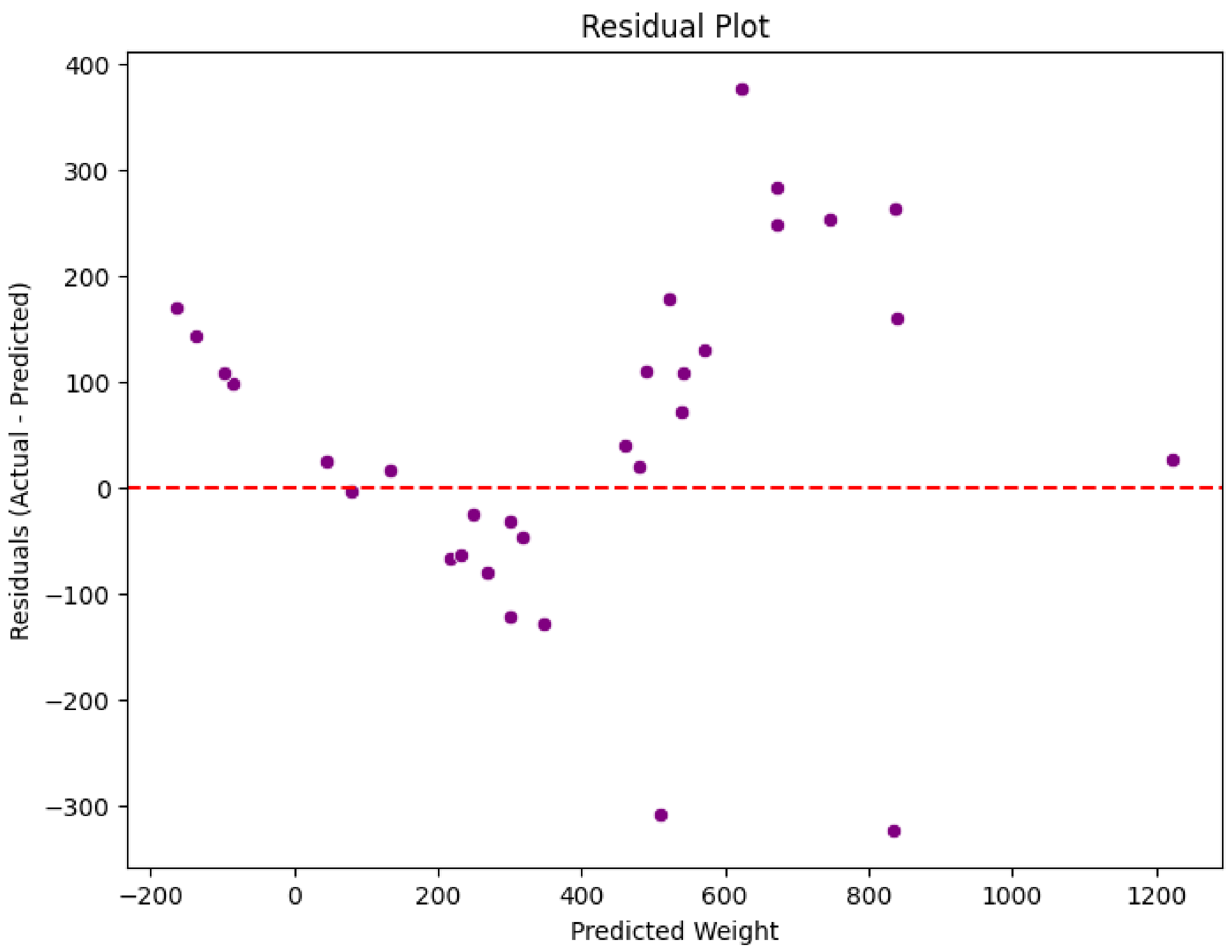
```
# Select only numeric columns
numeric_df = df.select_dtypes(include=[np.number])

sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap of Fish Market Dataset (Numeric
Features Only)")
plt.show()
```



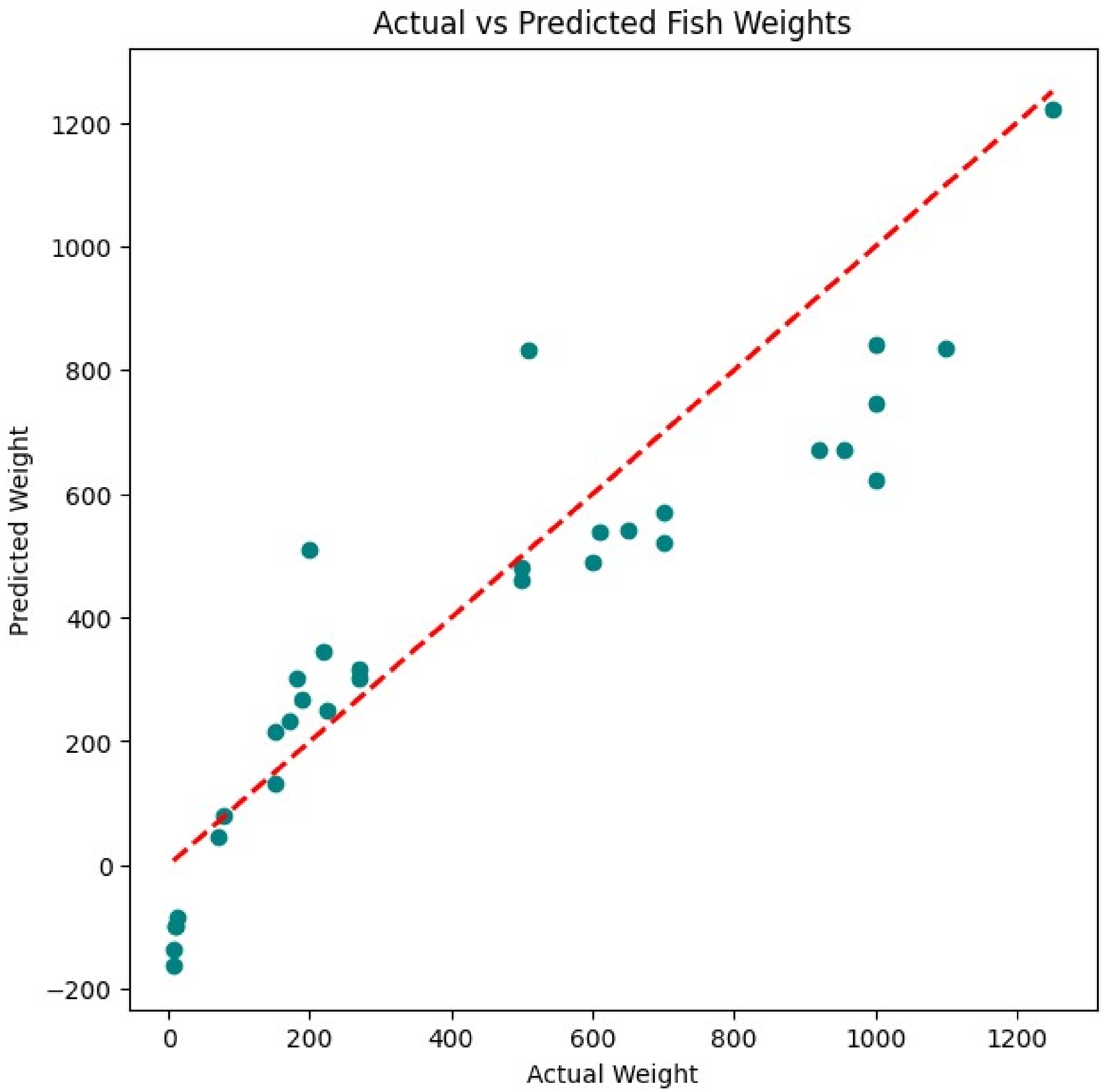

```
# Regression Plot
sns.lmplot(x='Length1', y='Weight', data=df, height=5, aspect=1.2,
scatter_kws={'color': 'blue'}, line_kws={'color': 'red'})
plt.title("Regression Line: Fish Weight vs Length1")
plt.show()
```



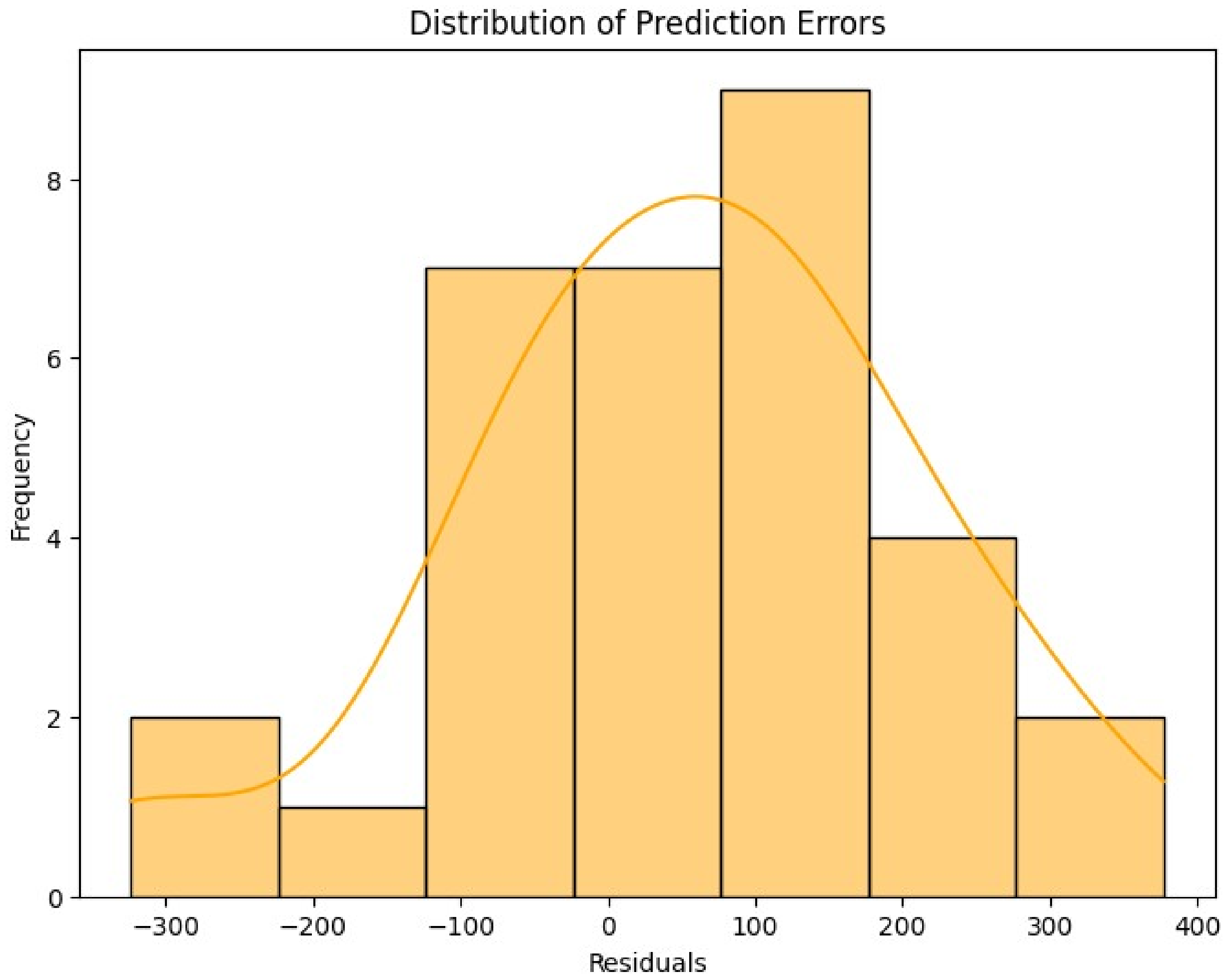

```
# Residual Plot
residuals = y_test - y_pred

plt.figure(figsize=(8,6))
sns.scatterplot(x=y_pred, y=residuals, color='purple')
plt.axhline(y=0, color='red', linestyle='--')
plt.title("Residual Plot")
plt.xlabel("Predicted Weight")
plt.ylabel("Residuals (Actual - Predicted)")
plt.show()
```



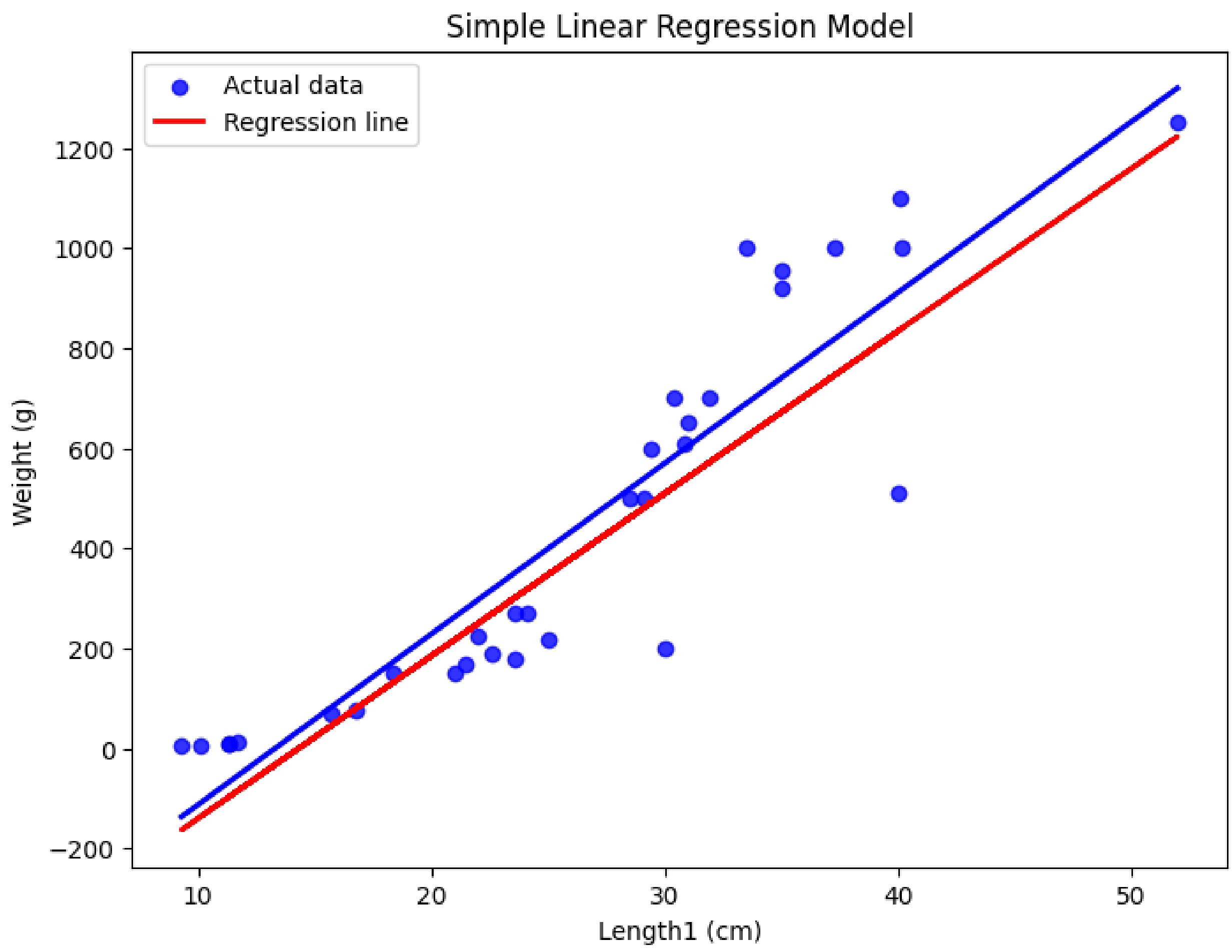

```
# Actual vs Predicted
plt.figure(figsize=(7,7))
plt.scatter(y_test, y_pred, color='teal')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', lw=2)
plt.xlabel("Actual Weight")
plt.ylabel("Predicted Weight")
plt.title("Actual vs Predicted Fish Weights")
plt.show()
```




```python
# Error Distribution
plt.figure(figsize=(8,6))
sns.histplot(residuals, kde=True, color='orange')
plt.title("Distribution of Prediction Errors")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.show()
```




```
plt.figure(figsize=(8,6))
sns.regplot(x=X_test.squeeze(), y=y_test, ci=None, color='blue',
label='Actual data')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression
line')
plt.title("Simple Linear Regression Model")
plt.xlabel("Length1 (cm)")
plt.ylabel("Weight (g)")
plt.legend()
plt.show()
```



Simple Linear Regression Model
Actual data
Regression line
Weight (g)
Length1 (cm)
−200
0
200
400
600
800
1000
1200
10
20
30
40
50