

Import Library and Data

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

 [Hiện kết quả đã ẩn](#)

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 import pandas as pd
5 import io
6 from scipy.stats import t
7 import scipy.stats as stats
8 from sklearn.linear_model import LinearRegression
9
```

Read file excel and display data for Example

```
1 # Load the data
2 df = pd.read_excel('/content/drive/MyDrive/Colab Notebooks/Adidas US Sales Datasets.xlsx')
3
4 # Drop the unnecessary column
5 df = df.drop('Unnamed: 0', axis=1)
6
7 # Select relevant columns
8 df_use = df[['Price per Unit', 'Units Sold', 'Product', 'Region', 'State']]
9
10 df_use=df_use.loc[(df_use['Product'] == "Men's Street Footwear") & (df_use['State']=='Florida')]
11
12 print(df_use.info())
13 print(df_use.describe())
```

 [Hiện kết quả đã ẩn](#)

```
1 df.head(10)
```



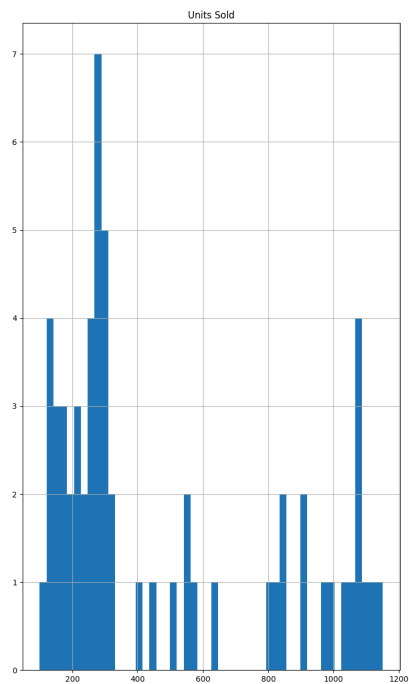
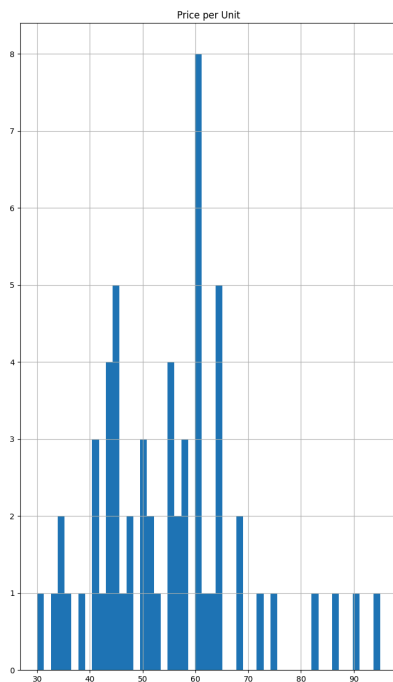
	Retailer	Retailer ID	Invoice Date	Region	State	City	Product	Price per Unit	Units Sold	Total Sales
0	Foot Locker	1185732	2020-01-01	Northeast	New York	New York	Men's Street Footwear	50.0	1200	600000.0
1	Foot Locker	1185732	2020-01-02	Northeast	New York	New York	Men's Athletic Footwear	50.0	1000	500000.0
2	Foot Locker	1185732	2020-01-03	Northeast	New York	New York	Women's Street Footwear	40.0	1000	400000.0
3	Foot Locker	1185732	2020-01-04	Northeast	New York	New York	Women's Athletic Footwear	45.0	850	382500.0
4	Foot Locker	1185732	2020-01-05	Northeast	New York	New York	Men's Apparel	60.0	900	540000.0
5	Foot Locker	1185732	2020-01-06	Northeast	New York	New York	Women's Apparel	50.0	1000	500000.0
6	Foot Locker	1185732	2020-01-07	Northeast	New York	New York	Men's Street Footwear	50.0	1250	625000.0
	Foot		2020-		New	New	Men's			

Display

```
1 display(df_use)
```

 **Hiện kết quả đã ẩn**

```
1 %matplotlib inline
2 import matplotlib.pyplot as plt
3 df_use.hist(bins=50, figsize=(20,15))
4 plt.show()
```



I.Linear Regression

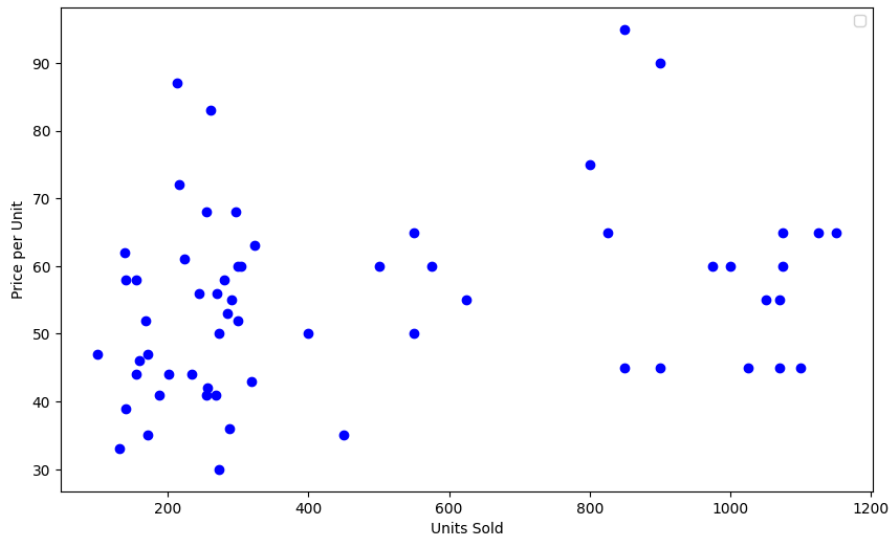
Scatter diagram

```

1 X=df_use['Units Sold']
2 y=df_use['Price per Unit']
3 X = pd.Series(X) # Flatten to convert to 1D Series
4
5 # Reshape X to be a 2D array (required by scikit-learn)
6 X = np.array(X).reshape(-1, 1)
7 y = pd.Series(y) # Flatten to convert to 1D Series
8
9 # Reshape y to be a 2D array (required by scikit-learn)
10 y = np.array(y).reshape(-1, 1)
11 plt.figure(figsize=(10, 6))
12
13 # Scatter plot
14 plt.scatter(X, y, color='blue')
15 plt.ylabel('Price per Unit')
16 plt.xlabel('Units Sold')
17 plt.legend()

```

WARNING:matplotlib.legend.No artists with labels found to put in legend. Note that arti
<matplotlib.legend.Legend at 0x7d4b90d4f4c0>



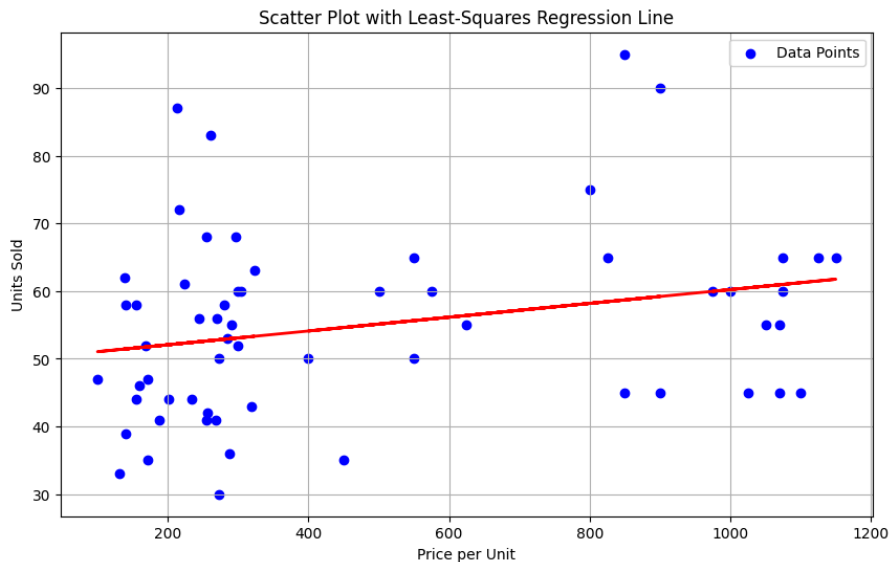
Least-Square Regression Line

```

1 model = LinearRegression()
2
3 # Fit the model to the data
4 model.fit(X, y)
5
6 # Get the coefficients (slope and intercept)
7 slope = model.coef_[0][0]
8 intercept = model.intercept_[0]
9
10 print(f"Slope: {slope:.2f}")
11 print(f"Intercept: {intercept:.2f}")
12 # Plot the data points
13 plt.figure(figsize=(10, 6))
14 plt.scatter(X, y, color='blue', label='Data Points')
15
16 # Plot the regression line
17 plt.plot(X, model.predict(X), color='red', linewidth=2)
18
19 # Add labels and title
20 plt.xlabel('Price per Unit')
21 plt.ylabel('Units Sold')
22 plt.title('Scatter Plot with Least-Squares Regression Line')
23 plt.legend()
24 plt.grid(True)
25 plt.show()
26

```

→ Slope: 0.01
Intercept: 50.04



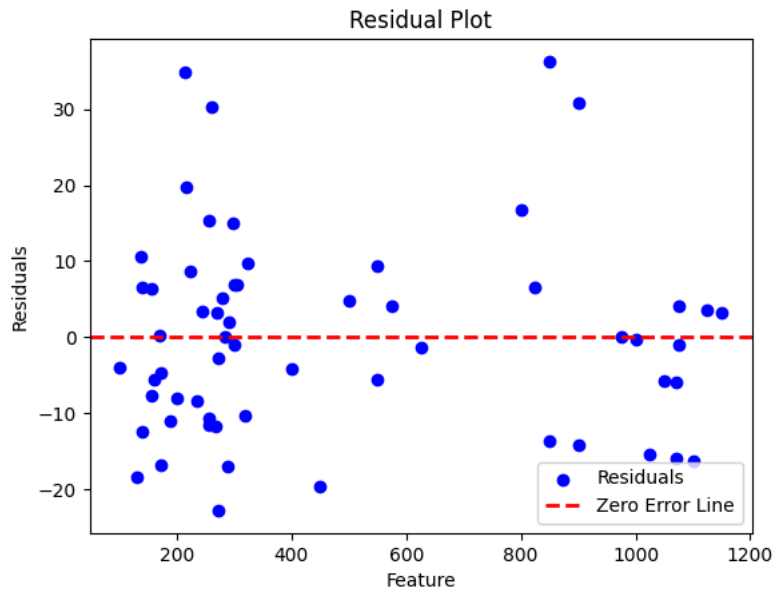
Residual Plot

```

1 model = LinearRegression()
2 model.fit(X, y)
3 y_pred = model.predict(X)
4 residuals = y - y_pred
5 plt.scatter(X, residuals, color='blue', label='Residuals')
6 plt.axhline(y=0, color='red', linestyle='--', linewidth=2, label='Zero Error Line')
7
8 # Labels and title
9 plt.xlabel('Feature')
10 plt.ylabel('Residuals')
11 plt.title('Residual Plot')
12 plt.legend()

```

 <matplotlib.legend.Legend at 0x7d4b904fb850>




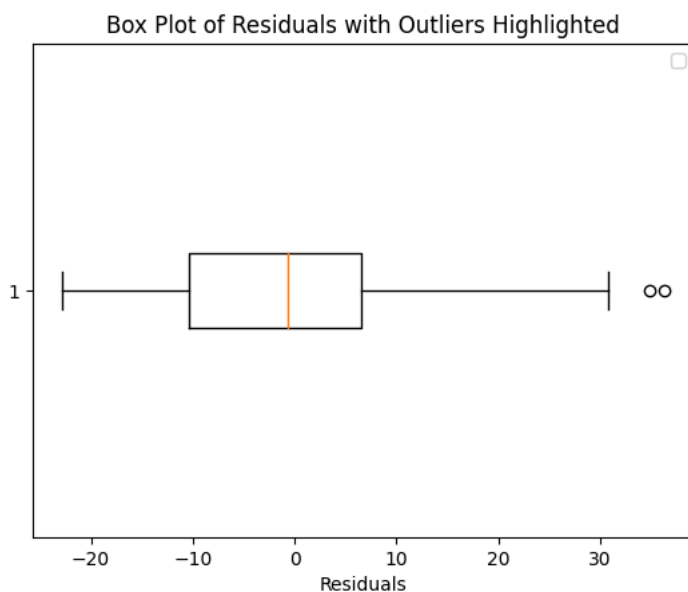
Outlier Dealing

```

1 Q1 = np.percentile(residuals, 25)
2 Q3 = np.percentile(residuals, 75)
3 IQR = Q3 - Q1
4 lower_bound = Q1 - 1.5 * IQR
5 upper_bound = Q3 + 1.5 * IQR
6 outliers = (residuals < lower_bound) | (residuals > upper_bound)
7 print(f"Outliers detected: {np.sum(outliers)}")
8 print(f"Outlier indices: {np.where(outliers)[0]}")
9 plt.boxplot(residuals, vert=False)
10 plt.xlabel('Residuals')
11 plt.title('Box Plot of Residuals with Outliers Highlighted')
12 plt.legend()
13 plt.figure(figsize=(10, 6))
14 plt.show()

```

 WARNING:matplotlib.legend.No artists with labels found to put in legend. Note that arti
 Outliers detected: 2
 Outlier indices: [17 41]

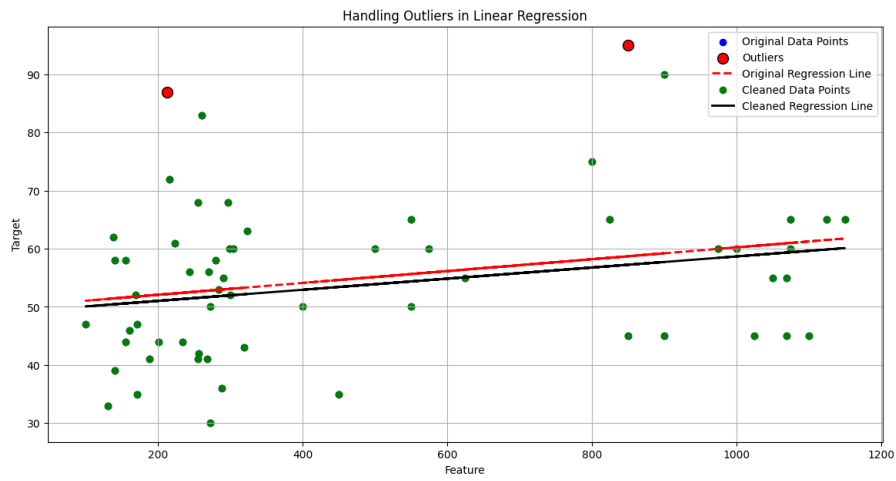


<Figure size 1000x600 with 0 Axes>

```

1 # Remove outliers
2 X_cleaned = X[~outliers.flatten()]
3 y_cleaned = y[~outliers.flatten()]
4
5 # Reshape cleaned data
6
7 # Step 5: Re-fit Linear Regression Model on Cleaned Data
8 model_cleaned = LinearRegression()
9 model_cleaned.fit(X_cleaned, y_cleaned)
10 y_cleaned_pred = model_cleaned.predict(X_cleaned)
11
12 # Plot the original data, cleaned data, and regression lines
13 plt.figure(figsize=(14, 7))
14
15 # Scatter plot of the original data points
16 plt.scatter(X, y, color='blue', label='Original Data Points')
17
18 # Highlight the outliers
19 plt.scatter(X[outliers], y[outliers], color='red', edgecolor='k', s=100, label='Outliers')
20
21 # Plot the original regression line
22 plt.plot(X, y_pred, color='red', linewidth=2, linestyle='--', label='Original Regression Line')
23
24 # Scatter plot of the cleaned data points
25 plt.scatter(X_cleaned, y_cleaned, color='green', label='Cleaned Data Points')
26
27 # Plot the cleaned regression line
28 plt.plot(X_cleaned, y_cleaned_pred, color='black', linewidth=2, label='Cleaned Regression Line')
29
30 # Adding labels and title
31 plt.xlabel('Feature')
32 plt.ylabel('Target')
33 plt.title('Handling Outliers in Linear Regression')
34 plt.legend()
35 plt.grid(True)
36 plt.show()
37
38 # Print the slopes and intercepts of both models
39 original_slope = model.coef_[0][0]
40 original_intercept = model.intercept_[0]
41 cleaned_slope = model_cleaned.coef_[0][0]
42 cleaned_intercept = model_cleaned.intercept_[0]
43 print(f"Original Slope: {original_slope:.2f}, Original Intercept: {original_intercept:.2f}")
44 print(f"Cleaned Slope: {cleaned_slope:.2f}, Cleaned Intercept: {cleaned_intercept:.2f}")
45
46 # Plot boxplot of residuals for the cleaned data
47 cleaned_residuals = y_cleaned - y_cleaned_pred
48 plt.figure(figsize=(10, 5))
49 plt.boxplot(cleaned_residuals, vert=False)
50 plt.title('Box Plot of Residuals (Cleaned Data)')
51 plt.show()

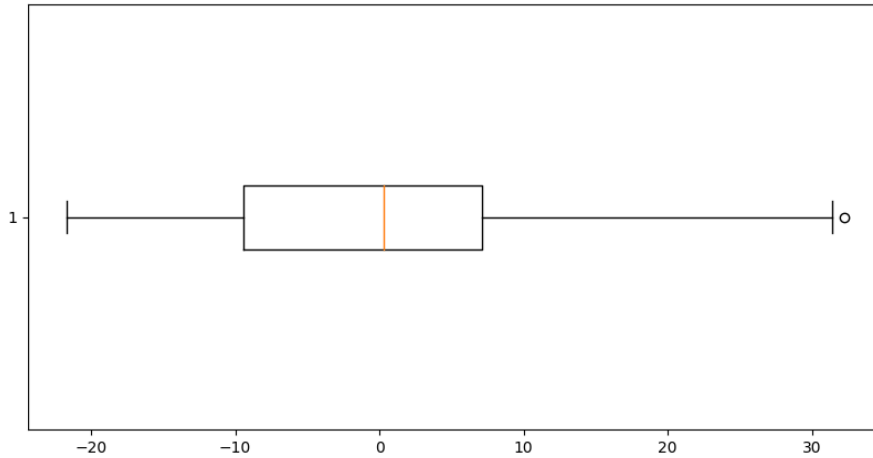
```



Original Slope: 0.01, Original Intercept: 50.04

Cleaned Slope: 0.01, Cleaned Intercept: 49.11

Box Plot of Residuals (Cleaned Data)



II Descriptive Statistics

```
1 df_use['Price per Unit'].describe()
```



```
count    60.000000
mean     54.916667
std      13.721555
min      30.000000
25%      45.000000
50%      55.000000
75%      61.250000
max      95.000000
Name: Price per Unit, dtype: float64
```

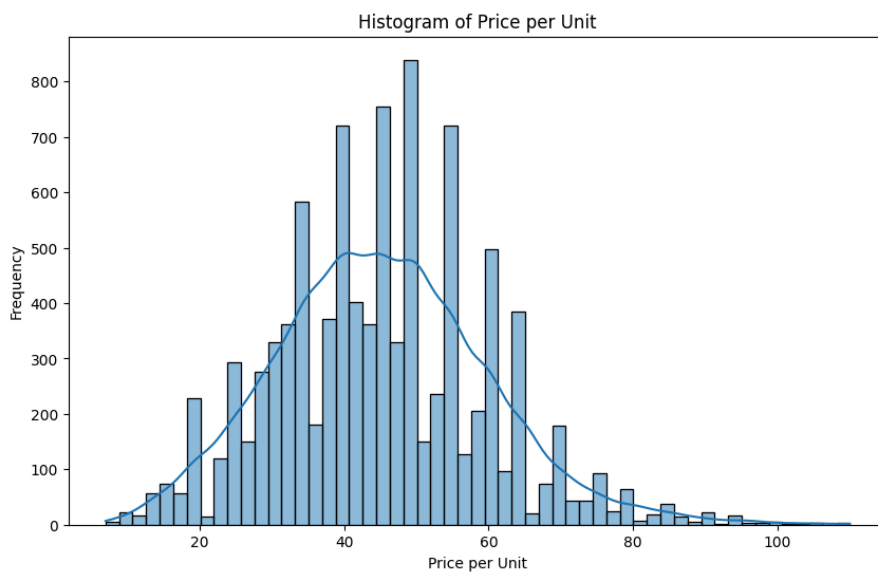
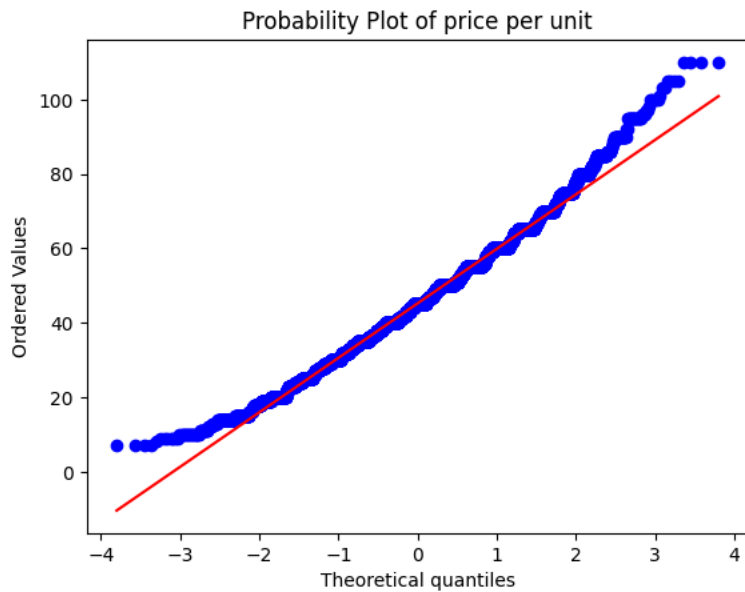
```
1 df_use['Units Sold'].describe()
```

```
count      60.000000
mean       478.866667
std        346.777097
min        100.000000
25%        221.250000
50%        293.500000
75%        831.250000
max        1150.000000
Name: Units Sold, dtype: float64
```

III Normality

Price per unit

```
1  # Draw probability plot
2  stats.probplot(df['Price per Unit'], dist="norm", plot=plt)
3  plt.title('Probability Plot of price per unit')
4  plt.show()
5
6  # Draw histogram plot
7  plt.figure(figsize=(10, 6))
8  sns.histplot(df['Price per Unit'], kde=True)
9  plt.title('Histogram of Price per Unit')
10 plt.xlabel('Price per Unit')
11 plt.ylabel('Frequency')
12 plt.show()
13
```

Transform data to normal distribution

```
1 # Using Log
2 transformd_data = np.log(df['Price per Unit'])
3 sns.histplot(transformd_data, kde = True, color = 'blue')
4 plt.title('Histogram of Price per Unit')
5 plt.xlabel('Price per Unit')
6 plt.ylabel('Frequency')
7 plt.show()
```



Hiện kết quả đã ẩn

Units Sold

```

1 # Draw probability plot
2 stats.probplot(df['Units Sold'], dist="norm", plot=plt)
3 plt.title('Probability Plot')
4 plt.show()
5
6 # Draw histogram plot
7 plt.figure(figsize=(10, 6))
8 sns.histplot(df['Units Sold'], kde=True)
9 plt.title('Histogram of Units Sold')
10 plt.xlabel('Units Sold')
11 plt.ylabel('Frequency')
12 plt.show()

```

 [Hiện kết quả đã ẩn](#)

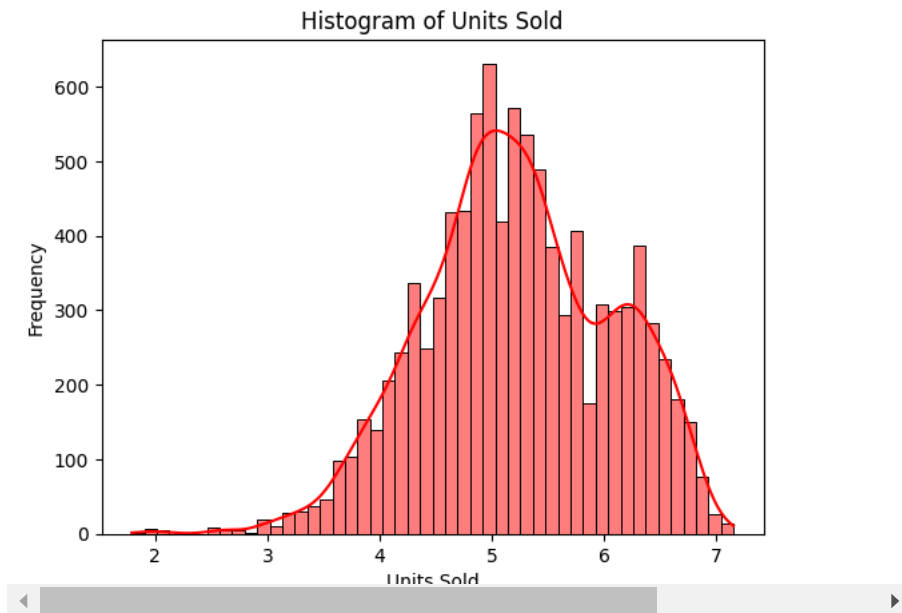
Transform data to normal distribution

```

1 # Using Log
2 transformd_data2 = np.log(df['Units Sold'])
3 sns.histplot(transformd_data2, kde = True, color = 'red')
4 plt.title('Histogram of Units Sold')
5 plt.xlabel('Units Sold')
6 plt.ylabel('Frequency')
7 plt.show()

```

 /usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:396: RuntimeWarning: di
result = getattr(ufunc, method)(*inputs, **kwargs)



III Confidence Interval

```
1 #Construct confidence Interval
2 def confidence_interval(data, confidence=0.95):
3     # Convert data to a numpy arr
4     data = np.array(data)
5
6     #calculate Sample mean and SD error
1 data_Price = df_use['Price per Unit']
2
3 #calculate the confidence interval
```