```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
# Reading data from an Excel file
df = pd.read_excel('data.xlsx')
```

```
# Displaying the first few rows
print(df.head())
```

Show hidden output

```
#Checking for missing values
print(df.isnull().sum())
```

Show hidden output

```
# Converting 'Rank' to numeric
df['Rank'] = df['Rank'].str.extract('(\d+)').astype(int)
```

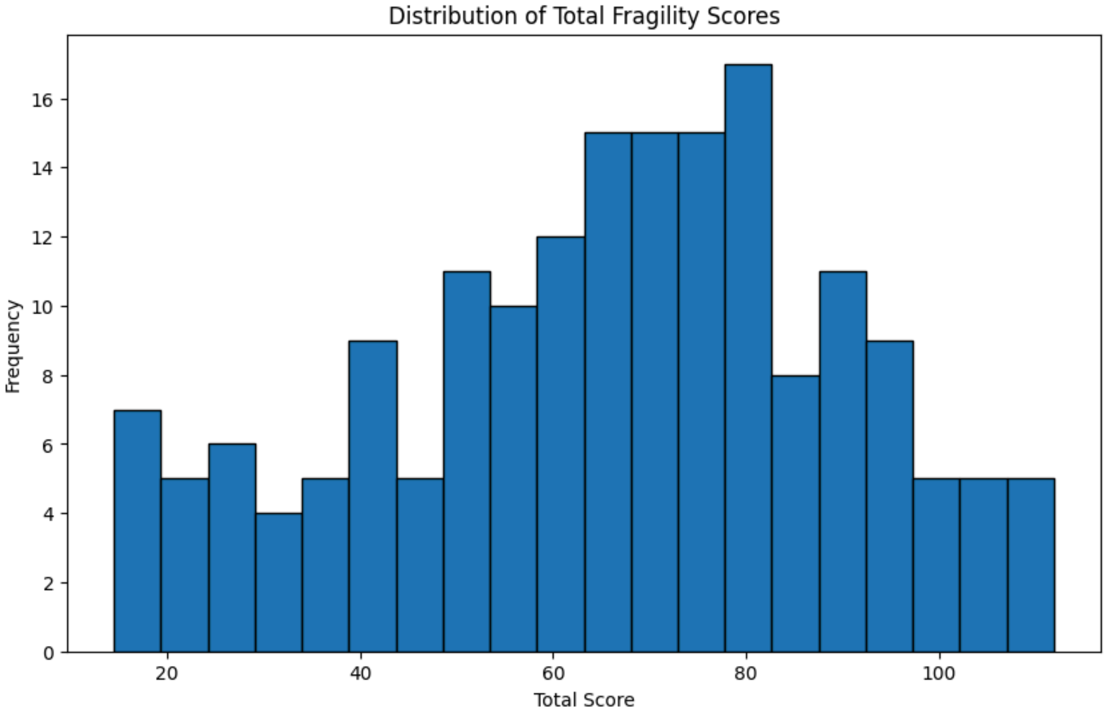
```
# Convert 'Total' to float
df['Total'] = df['Total'].astype(float)
```

```
# Exploratory Data Analysis (EDA)
# Summary statistics
print(df.describe())
```

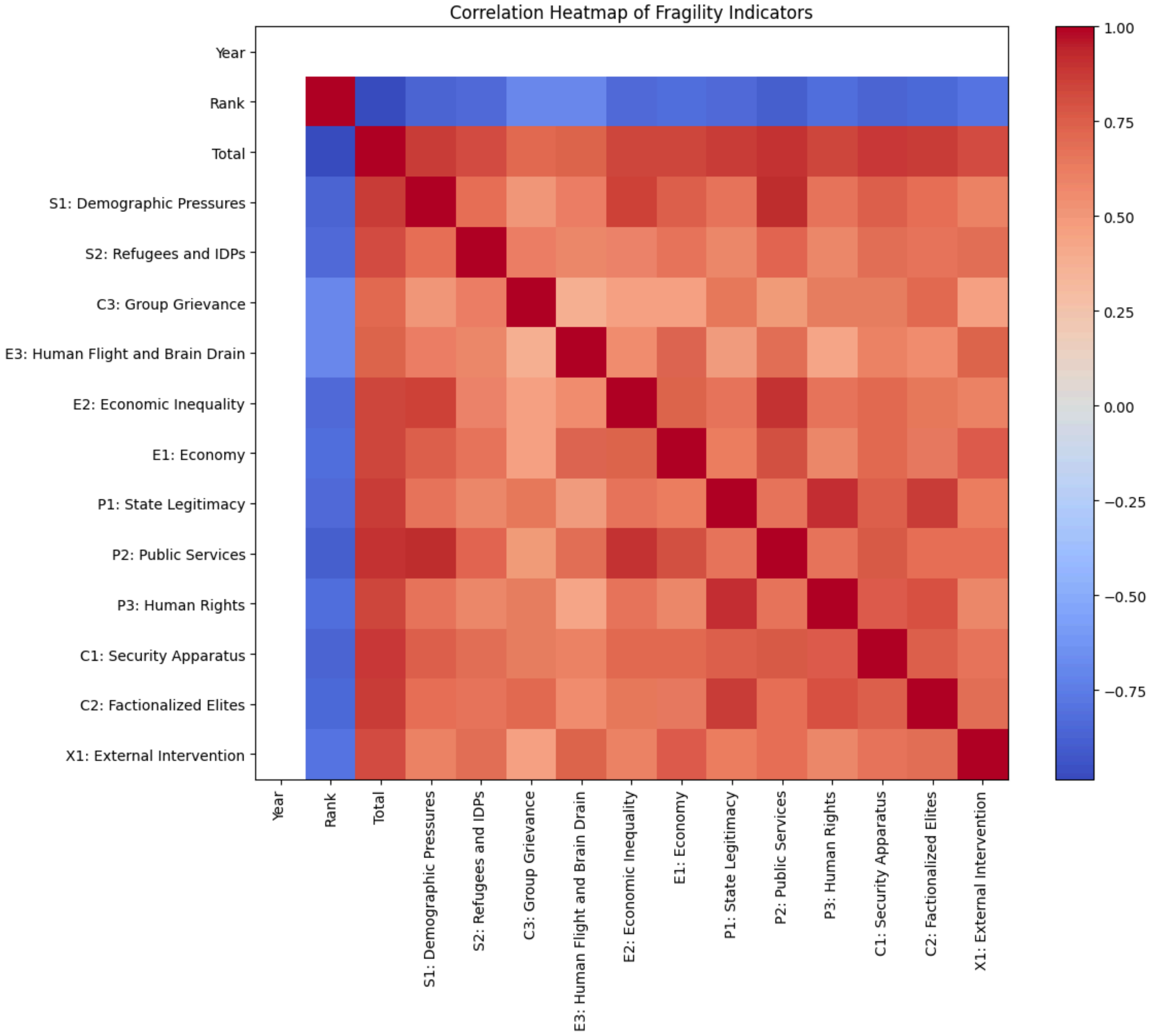
```
         Year        Rank       Total  S1: Demographic Pressures  \
count   179.0  179.000000  179.000000                 179.000000   
mean   2023.0   90.000000   65.832402                   5.955866   
std       0.0   51.816986   23.966251                   2.278726   
min    2023.0    1.000000   14.500000                   1.100000   
25%    2023.0   45.500000   49.000000                   4.100000   
50%    2023.0   90.000000   68.200000                   5.900000   
75%    2023.0  134.500000   82.200000                   8.050000   
max    2023.0  179.000000  111.900000                  10.000000   

       S2: Refugees and IDPs  C3: Group Grievance  \
count             179.000000           179.000000   
mean                4.764246             5.574860   
std                 2.373935             2.367757   
min                 0.500000             0.300000   
25%                 2.800000             3.600000   
50%                 4.500000             5.500000   
75%                 6.450000             7.550000   
max                10.000000             9.700000   

       E3: Human Flight and Brain Drain  E2: Economic Inequality  E1: Economy  \
count                        179.000000               179.000000   179.000000   
mean                           5.184358                 5.323464     5.687151   
std                            2.079591                 2.068546     2.200741   
min                            0.400000                 1.400000     1.000000   
25%                            3.700000                 3.650000     4.100000   
50%                            5.600000                 5.200000     6.000000   
75%                            6.600000                 7.200000     7.150000   
max                           10.000000                 9.600000     9.900000   

       P1: State Legitimacy  P2: Public Services  P3: Human Rights  \
count            179.000000           179.000000        179.000000   
mean               5.741341             5.459218          5.436872   
std                2.901853             2.581299          2.602588   
min                0.300000             0.900000          0.400000   
25%                3.650000             3.450000          3.600000   
50%                6.400000             5.100000          5.700000   
75%                8.100000             7.950000          7.500000   
max               10.000000            10.000000          9.900000   

       C1: Security Apparatus  C2: Factionalized Elites  \
count              179.000000                179.000000   
mean                 5.014525                  6.618436   
std                  2.379810                  2.427869   
min                  0.300000                  1.000000   
25%                  3.350000                  4.950000   
50%                  5.100000                  7.200000   
```

```
75%                 6.700000                  8.550000
max                10.000000                 10.000000

       X1: External Intervention
count                 179.000000
mean                    5.072067
std                     2.577801
min                     0.300000
25%                     3.150000
50%                     5.300000
75%                     7.000000
```

```
# Visualizing distribution of Total scores
plt.figure(figsize=(10, 6))
plt.hist(df['Total'], bins=20, edgecolor='black')
plt.title('Distribution of Total Fragility Scores')
plt.xlabel('Total Score')
plt.ylabel('Frequency')
plt.show()
```



```
# Correlation heatmap for numeric variables
numeric_columns = df.select_dtypes(include=[np.number]).columns
correlation_matrix = df[numeric_columns].corr()


plt.figure(figsize=(12, 10))
plt.imshow(correlation_matrix, cmap='coolwarm', interpolation='nearest')
plt.colorbar()
plt.xticks(range(len(numeric_columns)), numeric_columns, rotation=90)
plt.yticks(range(len(numeric_columns)), numeric_columns)
plt.title('Correlation Heatmap of Fragility Indicators')
plt.tight_layout()
plt.show()
```

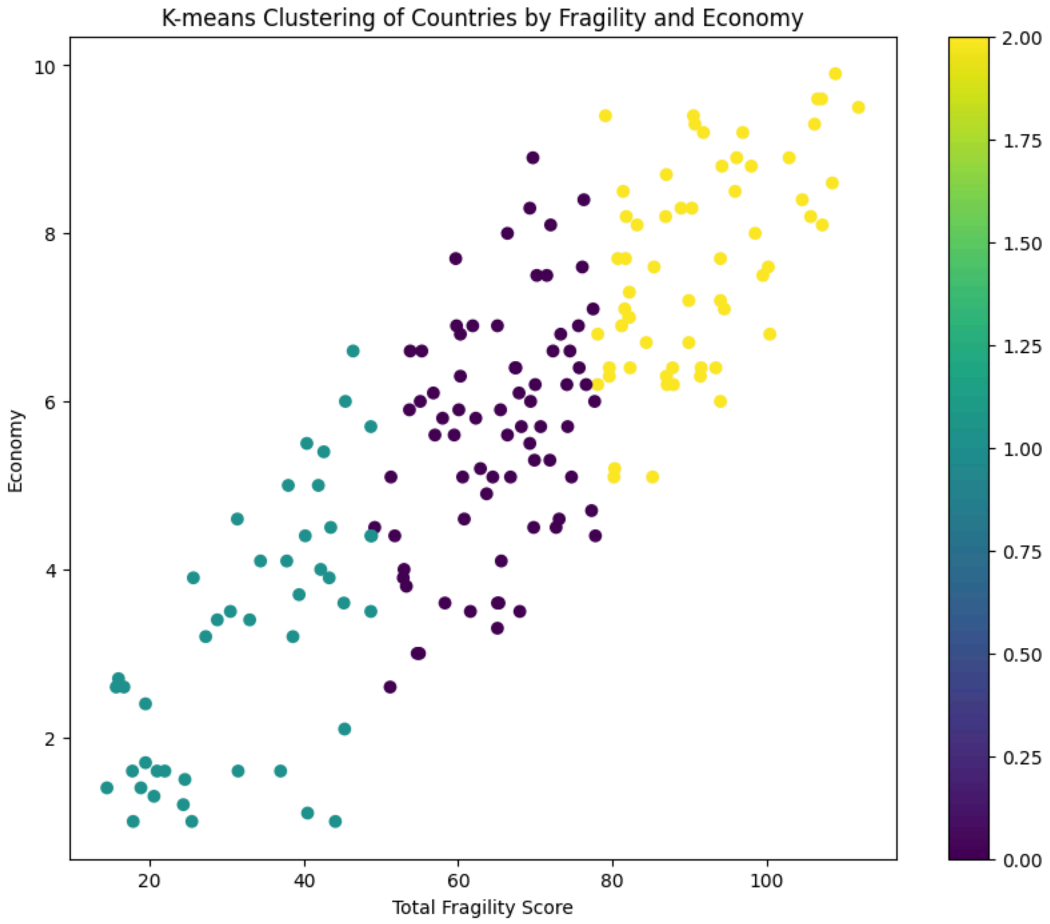


```
# Unsupervised Learning -Using Clustering
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler


# Select numeric features for clustering
features = df[['Total', 'S1: Demographic Pressures', 'S2: Refugees and IDPs', 'C3: Group Grievance',
               'E3: Human Flight and Brain Drain', 'E2: Economic Inequality', 'E1: Economy',
               'P1: State Legitimacy', 'P2: Public Services', 'P3: Human Rights', 'C1: Security Apparatus',
               'C2: Factionalized Elites', 'X1: External Intervention']]


# Scale the features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)


# Determining number of clusters
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(features_scaled)
```

```
KMeans
KMeans(n_clusters=3, random_state=42)
```

```
# Adding cluster labels to the original DataFrame
df['Cluster'] = kmeans.labels_
```

```
# Visualizing clusters
plt.figure(figsize=(10, 8))
scatter = plt.scatter(df['Total'], df['E1: Economy'], c=df['Cluster'], cmap='viridis')
plt.colorbar(scatter)
plt.xlabel('Total Fragility Score')
plt.ylabel('Economy')
plt.title('K-means Clustering of Countries by Fragility and Economy')
plt.show()
```



```
# Supervised Learning
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

```
X = features.drop('Total', axis=1)
y = features['Total']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

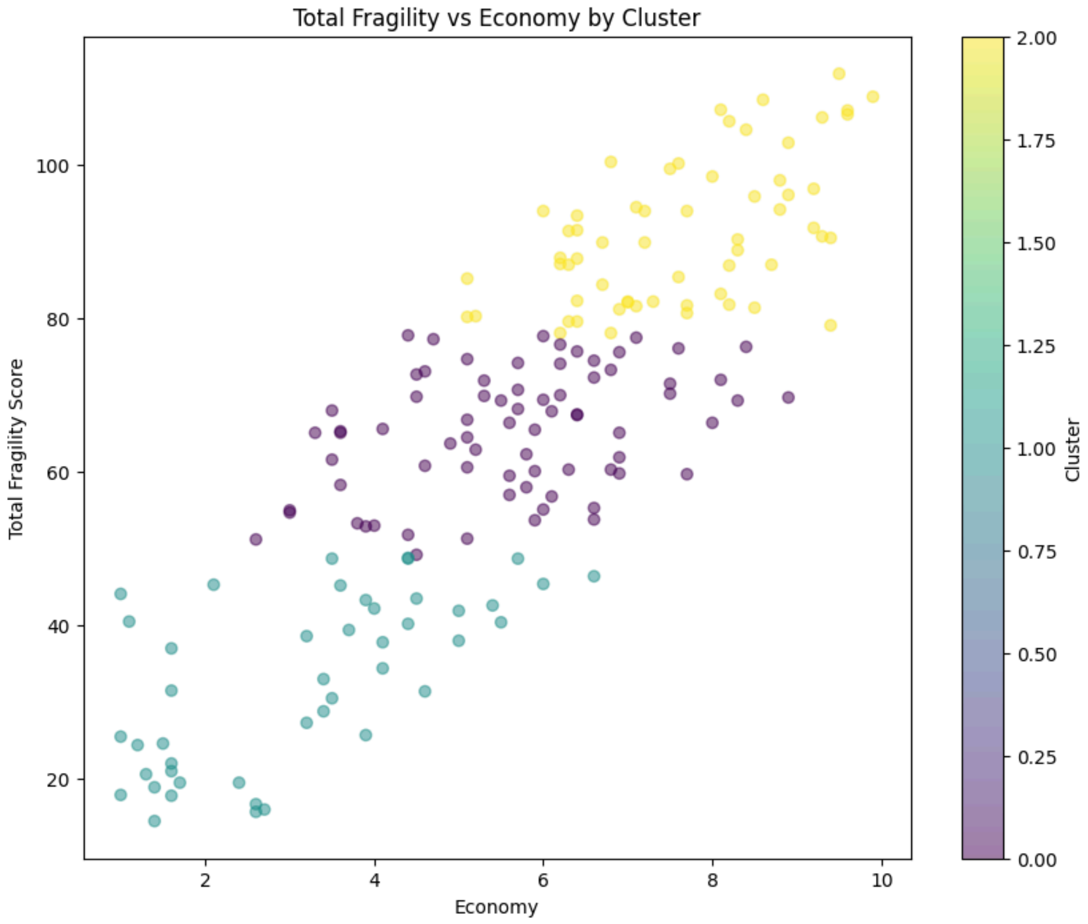
```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
▾ LinearRegression ⓘ ?
```

```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 5.33269971929404e-28
```

```
plt.figure(figsize=(10, 8))
plt.scatter(df['E1: Economy'], df['Total'], c=df['Cluster'], cmap='viridis', alpha=0.5)
plt.xlabel('Economy')
plt.ylabel('Total Fragility Score')
plt.title('Total Fragility vs Economy by Cluster')
plt.colorbar(label='Cluster')
plt.show()
```


Total Fragility vs Economy by Cluster
Total Fragility Score
Economy
Cluster