Task 2

Clustering

Importing Necessary Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy import stats
from sklearn.preprocessing import StandardScaler
import seaborn as sns
```

Loading and Exploring the data

In [2]:
```
data_path = "/content/drive/MyDrive/Regression,clustering,ANNproject/country_data.csv"

df = pd.read_csv(data_path)

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

# Getting basic statistics about the dataset
print(df.describe())

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

# Checking data types and column names
print(df.info())
```

```
              country  child_mort  exports  health  imports  income  \
0         Afghanistan        90.2     10.0    7.58     44.9    1610
1             Albania        16.6     28.0    6.55     48.6    9930
2             Algeria        27.3     38.4    4.17     31.4   12900
3              Angola       119.0     62.3    2.85     42.9    5900
4  Antigua and Barbuda        10.3     45.5    6.03     58.9   19100

   inflation  life_expec  total_fer   gdpp
0       9.44        56.2       5.82    553
1       4.49        76.3       1.65   4090
2      16.10        76.5       2.89   4460
3      22.40        60.1       6.16   3530
4       1.44        76.8       2.13  12200
       child_mort     exports      health     imports         income  \
count  167.000000  167.000000  167.000000  167.000000     167.000000
mean    38.270060   41.108976    6.815689   46.890215   17144.688623
std     40.328931   27.412010    2.746837   24.209589   19278.067698
min      2.600000    0.109000    1.810000    0.065900     609.000000
25%      8.250000   23.800000    4.920000   30.200000    3355.000000
50%     19.300000   35.000000    6.320000   43.300000    9960.000000
75%     62.100000   51.350000    8.600000   58.750000   22800.000000
max    208.000000  200.000000   17.900000  174.000000  125000.000000

        inflation  life_expec   total_fer           gdpp
count  167.000000  167.000000  167.000000     167.000000
mean     7.781832   70.555689    2.947964   12964.155689
std     10.570704    8.893172    1.513848   18328.704809
min     -4.210000   32.100000    1.150000     231.000000
25%      1.810000   65.300000    1.795000    1330.000000
50%      5.390000   73.100000    2.410000    4660.000000
75%     10.750000   76.800000    3.880000   14050.000000
max    104.000000   82.800000    7.490000  105000.000000
country       0
child_mort    0
exports       0
health        0
imports       0
income        0
inflation     0
life_expec    0
total_fer     0
gdpp          0
dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   country     167 non-null    object 
 1   child_mort  167 non-null    float64
 2   exports     167 non-null    float64
 3   health      167 non-null    float64
 4   imports     167 non-null    float64
 5   income      167 non-null    int64  
 6   inflation   167 non-null    float64
 7   life_expec  167 non-null    float64
 8   total_fer   167 non-null    float64
 9   gdpp        167 non-null    int64  
dtypes: float64(7), int64(2), object(1)
memory usage: 13.2+ KB
None
```

Data Preprocessing

In [3]:
```
# Removing Outliers
numeric_cols = df.select_dtypes(include=[np.number])

# Calculating z-scores for numeric columns
z_scores = stats.zscore(numeric_cols)

row_mask = (np.abs(z_scores) < 3).all(axis=1)

df_filtered = df[row_mask]

# Displaying the first few rows of the preprocessed DataFrame
print(df_filtered.head())
```

```
              country  child_mort  exports  health  imports  income  \
0         Afghanistan        90.2     10.0    7.58     44.9    1610
1             Albania        16.6     28.0    6.55     48.6    9930
2             Algeria        27.3     38.4    4.17     31.4   12900
3              Angola       119.0     62.3    2.85     42.9    5900
4  Antigua and Barbuda        10.3     45.5    6.03     58.9   19100

   inflation  life_expec  total_fer   gdpp
0       9.44        56.2       5.82    553
1       4.49        76.3       1.65   4090
2      16.10        76.5       2.89   4460
3      22.40        60.1       6.16   3530
4       1.44        76.8       2.13  12200
```

Feature Selection

In [4]:
```
selected_features = ['child_mort', 'exports', 'health', 'imports', 'income', 'inflation', 'life_expec', 't

# Select the relevant features from the DataFrame
df_selected = df_filtered[selected_features]

# Display the first few rows of the DataFrame with selected features
print(df_selected.head())
```

```
   child_mort  exports  health  imports  income  inflation  life_expec  \
0        90.2     10.0    7.58     44.9    1610       9.44        56.2
1        16.6     28.0    6.55     48.6    9930       4.49        76.3
2        27.3     38.4    4.17     31.4   12900      16.10        76.5
3       119.0     62.3    2.85     42.9    5900      22.40        60.1
4        10.3     45.5    6.03     58.9   19100       1.44        76.8

   total_fer   gdpp
0       5.82    553
1       1.65   4090
2       2.89   4460
3       6.16   3530
4       2.13  12200
```

Standardizing the selected Features

In [5]:
```
# Using standard scalar
scaler = StandardScaler()

X_std = scaler.fit_transform(df_selected)

df_standardized = pd.DataFrame(data=X_std, columns=selected_features)

print(df_standardized.head())
```

```
   child_mort   exports    health   imports    income  inflation  life_expec  \
0    1.461836 -1.413304  0.318094 -0.043800 -0.954569   0.348785   -1.738235
1   -0.569112 -0.526002 -0.088760  0.150114 -0.331921  -0.365865    0.712299
2   -0.273852 -0.013338 -1.028868 -0.751321 -0.109654   1.310315    0.736682
3    2.256555  1.164802 -1.550273 -0.148618 -0.633516   2.219869   -1.262759
4   -0.742957  0.336653 -0.294162  0.689927  0.354339  -0.806205    0.773257

   total_fer      gdpp
0   1.944385 -0.722055
1  -0.886986 -0.467590
2  -0.045044 -0.440971
3   2.175240 -0.507878
4  -0.561073  0.115874
```

Using Clustering Algorithm/ Fitting the KMeans model

In [6]:
```
num_clusters = 3

kmeans = KMeans(n_clusters=num_clusters, random_state=42)

kmeans.fit(df_standardized)

# Obtaining the cluster labels for each data point
cluster_labels = kmeans.labels_

#Fitting the kmeans model
cluster_labels = kmeans.fit_predict(df_standardized)

# Adding cluster labels to the DataFrame
df_standardized['Cluster'] = cluster_labels

print(df_standardized.head())
```

```
   child_mort   exports    health   imports    income  inflation  life_expec  \
0    1.461836 -1.413304  0.318094 -0.043800 -0.954569   0.348785   -1.738235
1   -0.569112 -0.526002 -0.088760  0.150114 -0.331921  -0.365865    0.712299
2   -0.273852 -0.013338 -1.028868 -0.751321 -0.109654   1.310315    0.736682
3    2.256555  1.164802 -1.550273 -0.148618 -0.633516   2.219869   -1.262759
4   -0.742957  0.336653 -0.294162  0.689927  0.354339  -0.806205    0.773257

   total_fer      gdpp  Cluster
0   1.944385 -0.722055        2
1  -0.886986 -0.467590        1
2  -0.045044 -0.440971        1
3   2.175240 -0.507878        2
4  -0.561073  0.115874        1
```

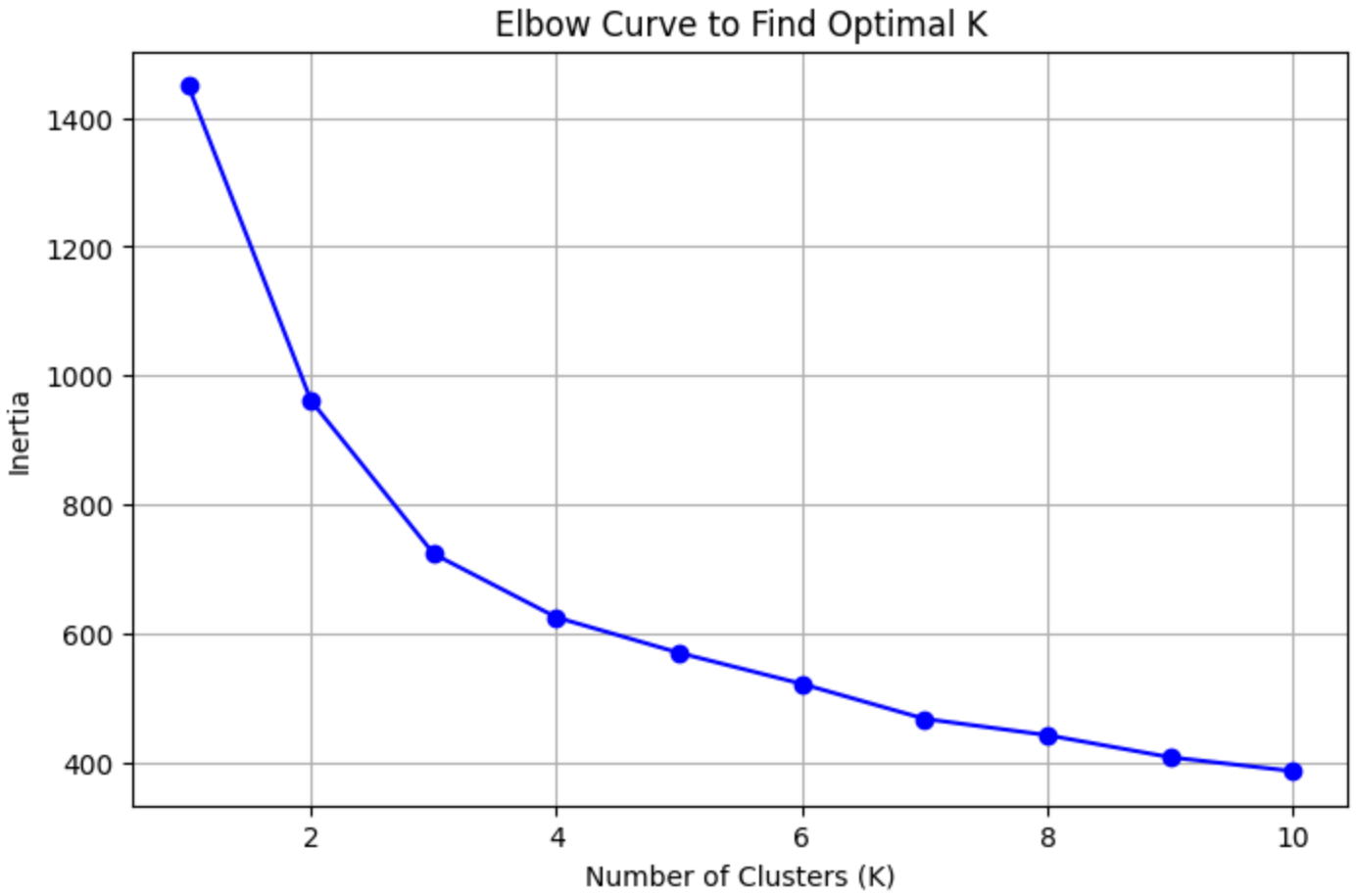
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
```

Determining the optimal number of clusters

In [7]:

```
inertia = []

# Defining a range of candidate cluster numbers (K values)
k_values = range(1, 11)

# Calculating the inertia for each K value
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df_standardized)
    inertia.append(kmeans.inertia_)

# Plotting the elbow curve to visualize the inertia values for different K values
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, marker='o', linestyle='-', color='b')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Curve to Find Optimal K')
plt.grid(True)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the wa
rning
  warnings.warn(
```

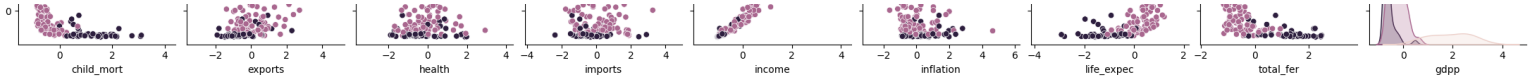

Elbow Curve to Find Optimal K
Inertia
1400
1200
1000
800
600
400
2
4
6
8
10
Number of Clusters (K)

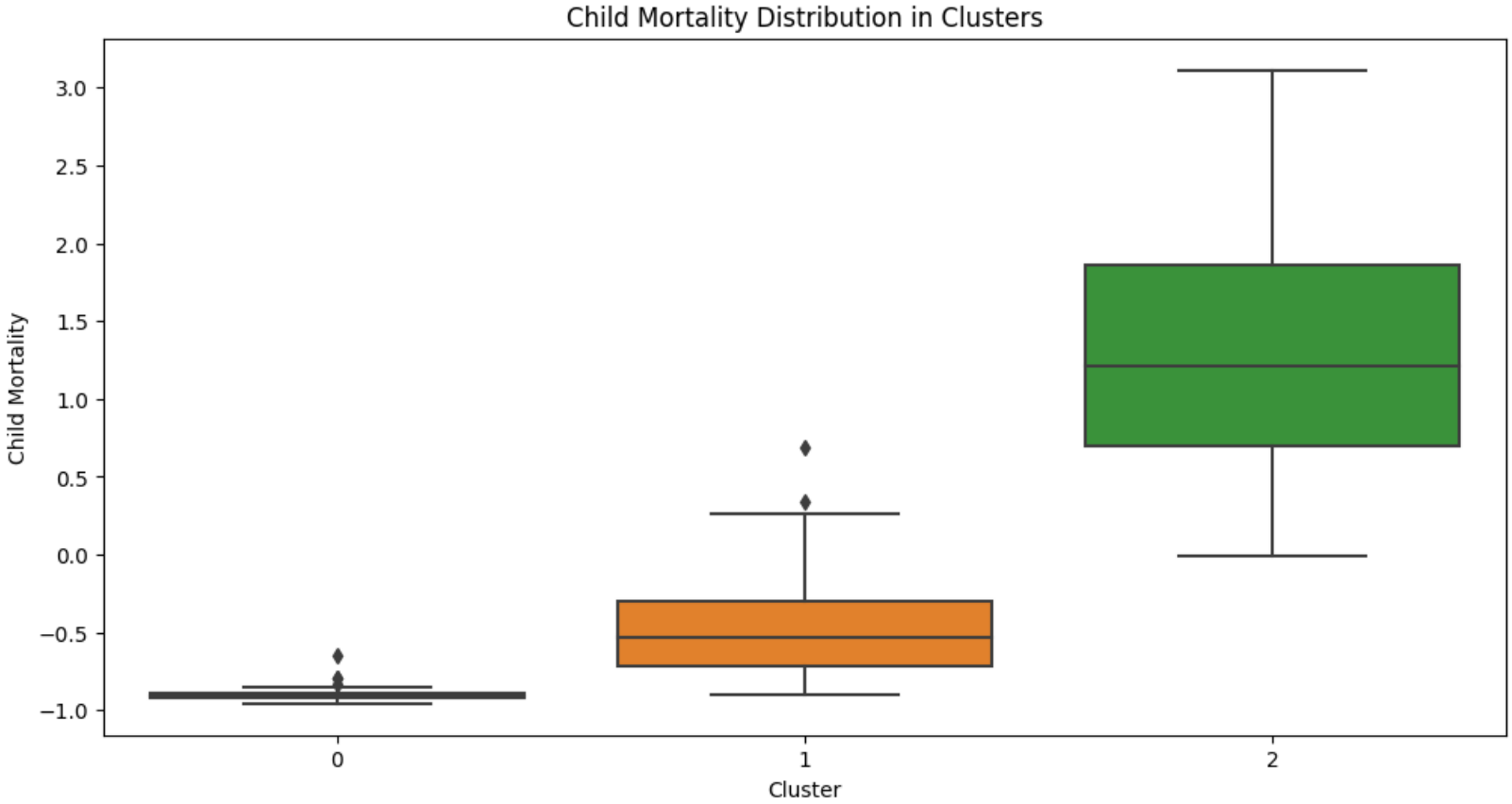
Visualize the Clustering

In [8]:
```
# Calculate descriptive statistics for each cluster
cluster_stats = df_standardized.groupby('Cluster').mean()

print(cluster_stats)

# Pairplot to visualize relationships between features for each cluster
sns.pairplot(df_standardized, hue='Cluster', diag_kind='kde')
plt.show()

# Boxplot to visualize feature distributions for each cluster
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_standardized, x='Cluster', y='child_mort')
plt.xlabel('Cluster')
plt.ylabel('Child Mortality')
plt.title('Child Mortality Distribution in Clusters')
plt.show()
```

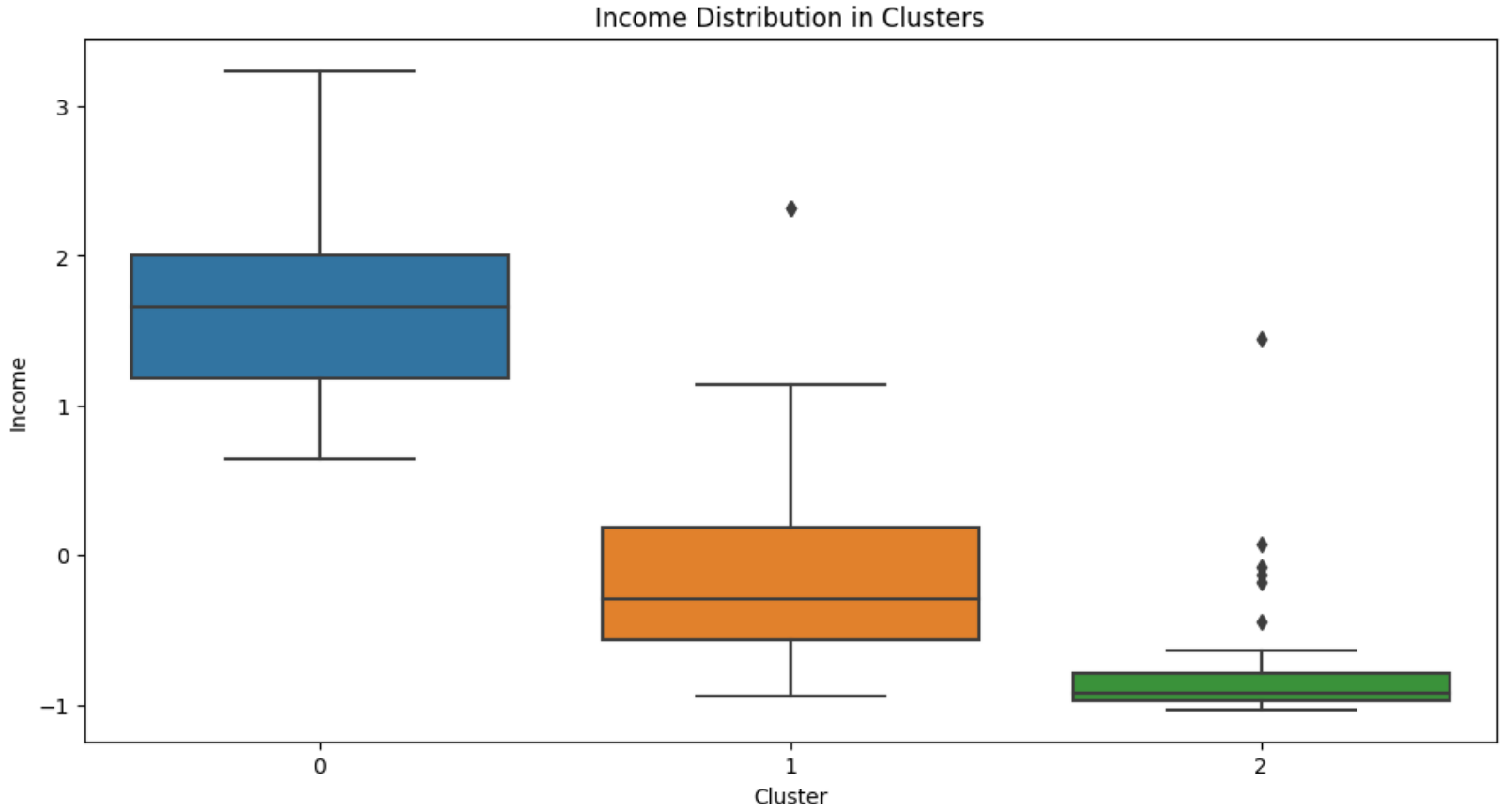
```
         child_mort   exports    health   imports    income  inflation  \
Cluster
0         -0.893051  0.398448  0.969709 -0.061768  1.669876  -0.776912
1         -0.465244  0.160204 -0.190397  0.159185 -0.132834   0.002712
2          1.304137 -0.503243 -0.261721 -0.227381 -0.774372   0.458340

         life_expec  total_fer      gdpp
Cluster
0          1.152071  -0.811813  1.885177
1          0.313567  -0.454612 -0.300944
2         -1.206729   1.238095 -0.623645
```


child_mort
exports
health
imports
income
inflation
life_expec
total_fer
gdpp
Cluster
0
1
2

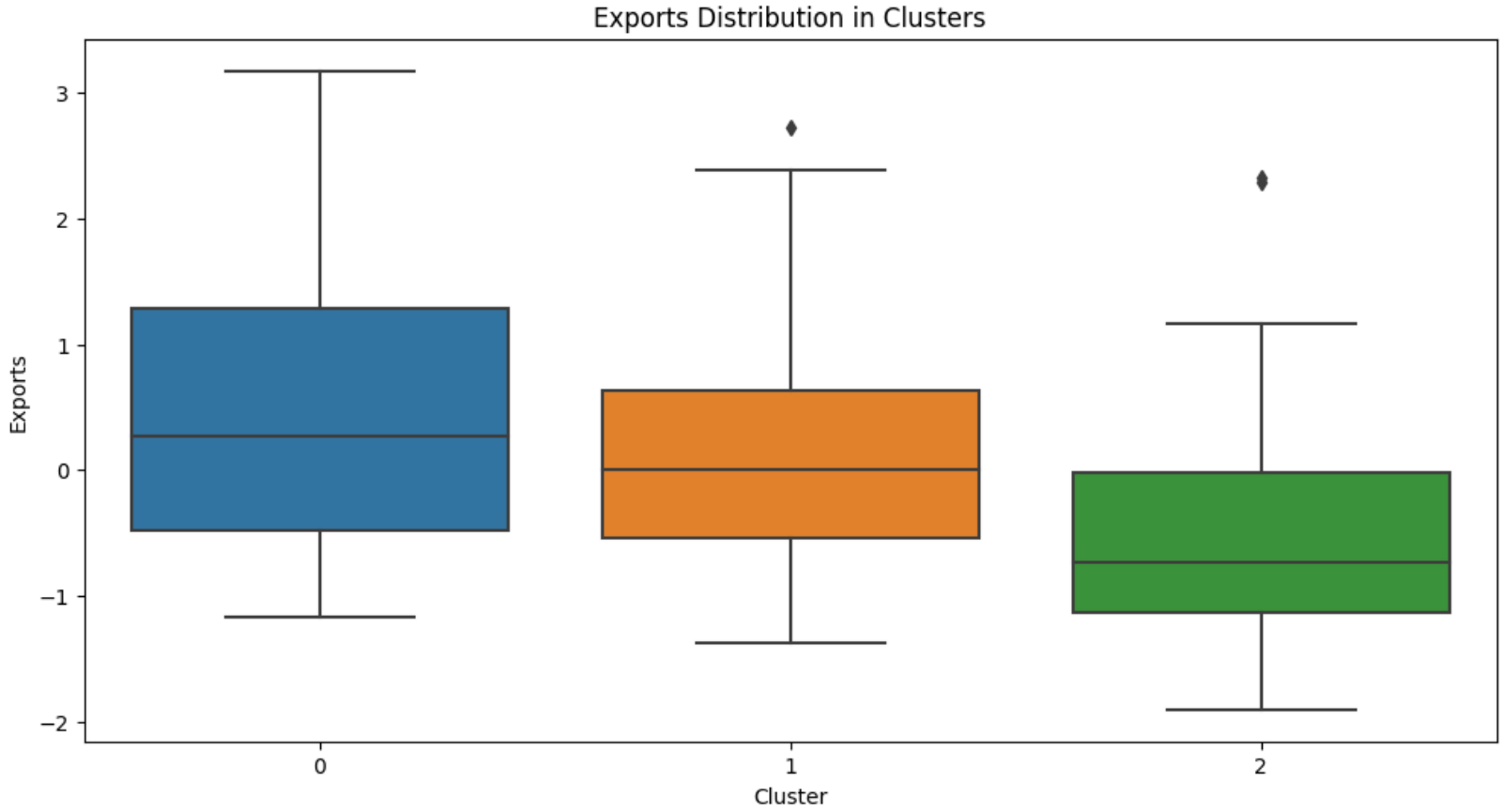

child_mort
exports
health
imports
income
inflation
life_expec
total_fer
gdpp


Child Mortality Distribution in Clusters
Child Mortality
Cluster
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
1
2

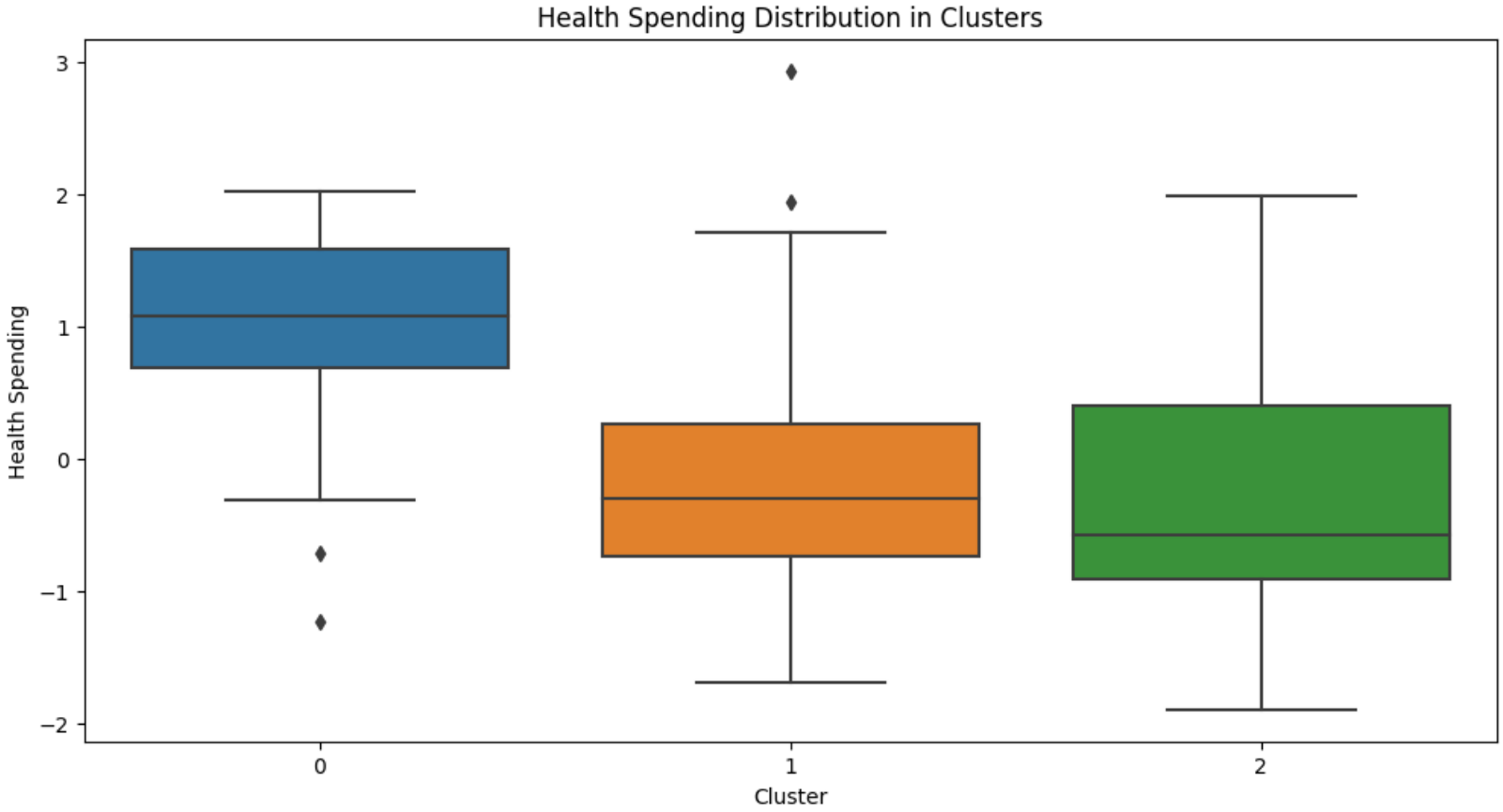
In [9]:
```
# Boxplot to visualize 'income' feature distribution for each cluster
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_standardized, x='Cluster', y='income')
plt.xlabel('Cluster')
plt.ylabel('Income')
plt.title('Income Distribution in Clusters')
plt.show()
```



In [10]:
```
# Boxplot to visualize 'exports' feature distribution for each cluster
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_standardized, x='Cluster', y='exports')
plt.xlabel('Cluster')
plt.ylabel('Exports')
plt.title('Exports Distribution in Clusters')
plt.show()
```



In [11]:
```
# Boxplot to visualize 'health' feature distribution for each cluster
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_standardized, x='Cluster', y='health')
plt.xlabel('Cluster')
plt.ylabel('Health Spending')
plt.title('Health Spending Distribution in Clusters')
plt.show()
```



In [12]:
```
# Boxplot to visualize 'inflation' feature distribution for each cluster
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_standardized, x='Cluster', y='inflation')
plt.xlabel('Cluster')
plt.ylabel('Inflation')
plt.title('Inflation Distribution in Clusters')
plt.show()
```