CLUSTERING COUNTRY PROBLEM

In [41]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.inspection import permutation_importance
from sklearn.cluster import AgglomerativeClustering
from sklearn.mixture import GaussianMixture
```

In [2]:
```
plt.style.use("seaborn-darkgrid")
```

```
C:\Users\gadoc\AppData\Local\Temp\ipykernel_32172\1120890811.py:1: MatplotlibDeprecationWarning: The seaborn styles shipped by M
atplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will remain ava
ilable as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn API instead.
  plt.style.use("seaborn-darkgrid")
```

In [3]:
```
file_path = 'country_data.csv'
data = pd.read_csv(file_path)
```

In [4]:
```
print("First few rows of the dataset:")
display(data.head())
```

```
First few rows of the dataset:
```

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200

In [5]:
```
print("\nDataset statistics:")
display(data.describe())
```

```
Dataset statistics:
```

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
count	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000
mean	38.270060	41.108976	6.815689	46.890215	17144.688623	7.781832	70.555689	2.947964	12964.155689
std	40.328931	27.412010	2.746837	24.209589	19278.067698	10.570704	8.893172	1.513848	18328.704809
min	2.600000	0.109000	1.810000	0.065900	609.000000	-4.210000	32.100000	1.150000	231.000000
25%	8.250000	23.800000	4.920000	30.200000	3355.000000	1.810000	65.300000	1.795000	1330.000000
50%	19.300000	35.000000	6.320000	43.300000	9960.000000	5.390000	73.100000	2.410000	4660.000000
75%	62.100000	51.350000	8.600000	58.750000	22800.000000	10.750000	76.800000	3.880000	14050.000000
max	208.000000	200.000000	17.900000	174.000000	125000.000000	104.000000	82.800000	7.490000	105000.000000

In [6]:
```
print("\nMissing values in each column:")
print(data.isnull().sum())
```

```
Missing values in each column:
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
```

In [7]:
```
data_description = data.describe(include='all')
data_description
```

Out[7]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
count	167	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000
unique	167	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	Afghanistan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	38.270060	41.108976	6.815689	46.890215	17144.688623	7.781832	70.555689	2.947964	12964.155689
std	NaN	40.328931	27.412010	2.746837	24.209589	19278.067698	10.570704	8.893172	1.513848	18328.704809
min	NaN	2.600000	0.109000	1.810000	0.065900	609.000000	-4.210000	32.100000	1.150000	231.000000
25%	NaN	8.250000	23.800000	4.920000	30.200000	3355.000000	1.810000	65.300000	1.795000	1330.000000
50%	NaN	19.300000	35.000000	6.320000	43.300000	9960.000000	5.390000	73.100000	2.410000	4660.000000
75%	NaN	62.100000	51.350000	8.600000	58.750000	22800.000000	10.750000	76.800000	3.880000	14050.000000
max	NaN	208.000000	200.000000	17.900000	174.000000	125000.000000	104.000000	82.800000	7.490000	105000.000000

Check for outliers by identifying values that are far from mean values in specific columns

In [8]:
```
outliers_info = data[(data['inflation'] > 50) | (data['gdpp'] > 100000) | (data['income'] > 100000)]
```

In [9]:
```
scaler = StandardScaler()
data[['income', 'gdpp', 'inflation']] = scaler.fit_transform(data[['income', 'gdpp', 'inflation']])
```

In [10]:
```
normalized_summary = data.describe()

outliers_info
```

Out[10]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
91	Luxembourg	2.8	175.0	7.77	142.0	91700	3.62	81.3	1.63	105000
113	Nigeria	130.0	25.3	5.07	17.4	5150	104.00	60.5	5.84	2330
123	Qatar	9.0	62.3	1.81	23.8	125000	6.98	79.5	2.07	70300

In [11]:
```
normalized_summary
```

Out[11]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
count	167.000000	167.000000	167.000000	167.000000	1.670000e+02	1.670000e+02	167.000000	167.000000	1.670000e+02
mean	38.270060	41.108976	6.815689	46.890215	-7.977650e-17	-1.063687e-17	70.555689	2.947964	5.850277e-17
std	40.328931	27.412010	2.746837	24.209589	1.003008e+00	1.003008e+00	8.893172	1.513848	1.003008e+00
min	2.600000	0.109000	1.810000	0.065900	-8.603259e-01	-1.137852e+00	32.100000	1.150000	-6.968005e-01
25%	8.250000	23.800000	4.920000	30.200000	-7.174558e-01	-5.666409e-01	65.300000	1.795000	-6.366596e-01
50%	19.300000	35.000000	6.320000	43.300000	-3.738080e-01	-2.269504e-01	73.100000	2.410000	-4.544309e-01
75%	62.100000	51.350000	8.600000	58.750000	2.942370e-01	2.816364e-01	76.800000	3.880000	5.942100e-02
max	208.000000	200.000000	17.900000	174.000000	5.611542e+00	9.129718e+00	82.800000	7.490000	5.036507e+00

In [12]:
```
features = data.select_dtypes(include=[np.number])
```

In [13]:
```
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

In [14]:
```
scaled_features
```

Out[14]:
```
array([[ 1.29153238, -1.13827979,  0.27908825, ..., -1.61909203,
         1.90288227, -0.67917961],
       [-0.5389489 , -0.47965843, -0.09701618, ...,  0.64786643,
        -0.85997281, -0.48562324],
       [-0.27283273, -0.09912164, -0.96607302, ...,  0.67042323,
        -0.0384044 , -0.46537561],
       ...,
       [-0.37231541,  1.13030491,  0.0088773 , ...,  0.28695762,
        -0.66120626, -0.63775406],
       [ 0.44841668, -0.40647827, -0.59727159, ..., -0.34463279,
         1.14094382, -0.63775406],
       [ 1.11495062, -0.15034774, -0.33801514, ..., -2.09278484,
         1.6246091 , -0.62954556]])
```

Determine Optimal Number of Clusters Using the Elbow Method

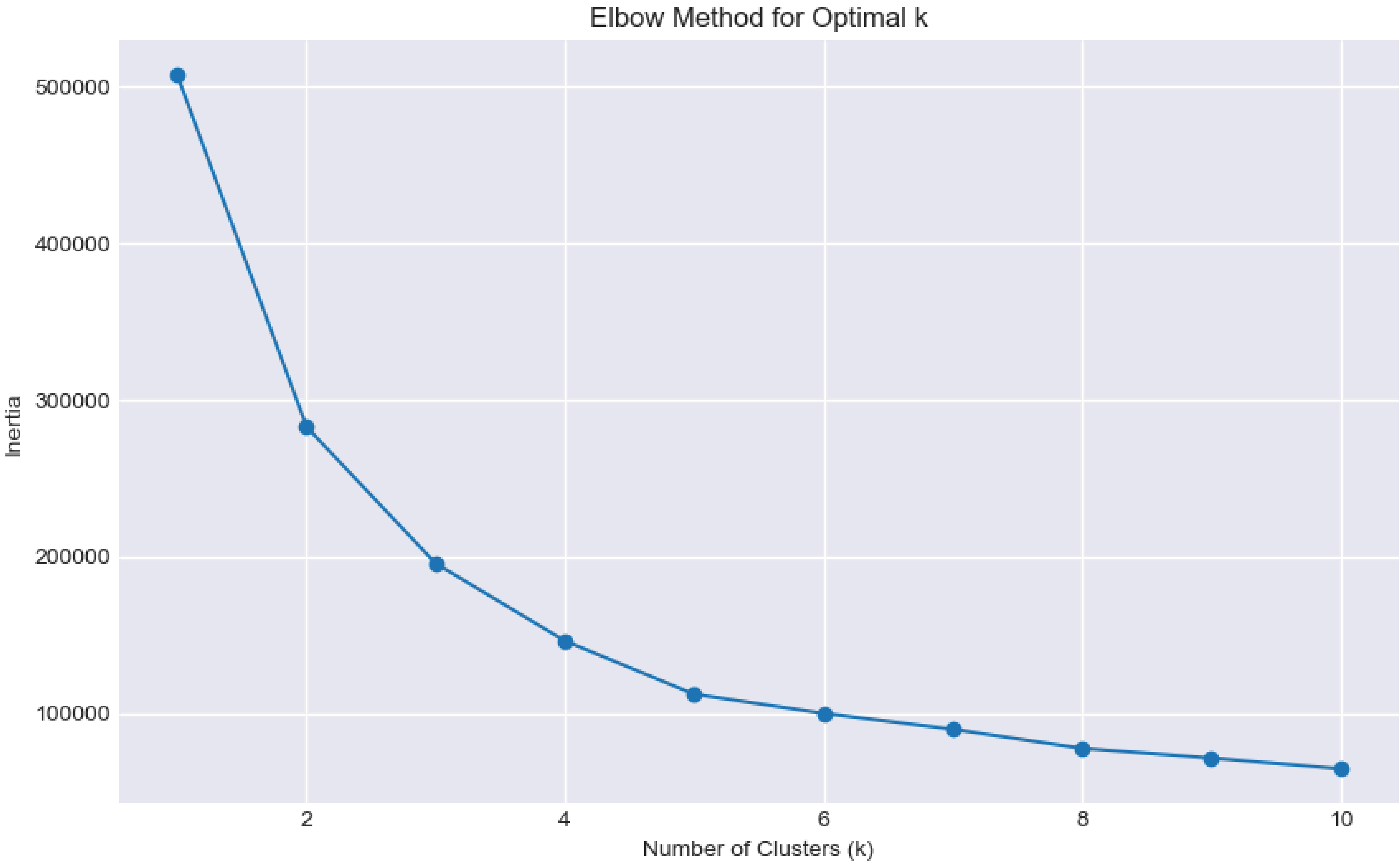
In [15]:
```python
k_values = range(1, 11)
inertia_values = []
```

In [17]:
```python
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data[['income', 'gdpp', 'inflation', 'child_mort', 'exports', 'health', 'imports', 'life_expec', 'total_fer']])
    inertia_values.append(kmeans.inertia_)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
```

```
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
```

In [18]:
```
plt.figure(figsize=(10, 6))
plt.plot(k_values, inertia_values, marker='o')
plt.title("Elbow Method for Optimal k")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia")
plt.grid(True)
plt.show()
```



Apply K-Means Clustering

In [105…

```
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=0)
clusters = kmeans.fit_predict(data[['income', 'gdpp', 'inflation', 'child_mort', 'exports', 'health', 'imports', 'life_expec']])
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
```

In [106…

```
data['Cluster'] = clusters
```

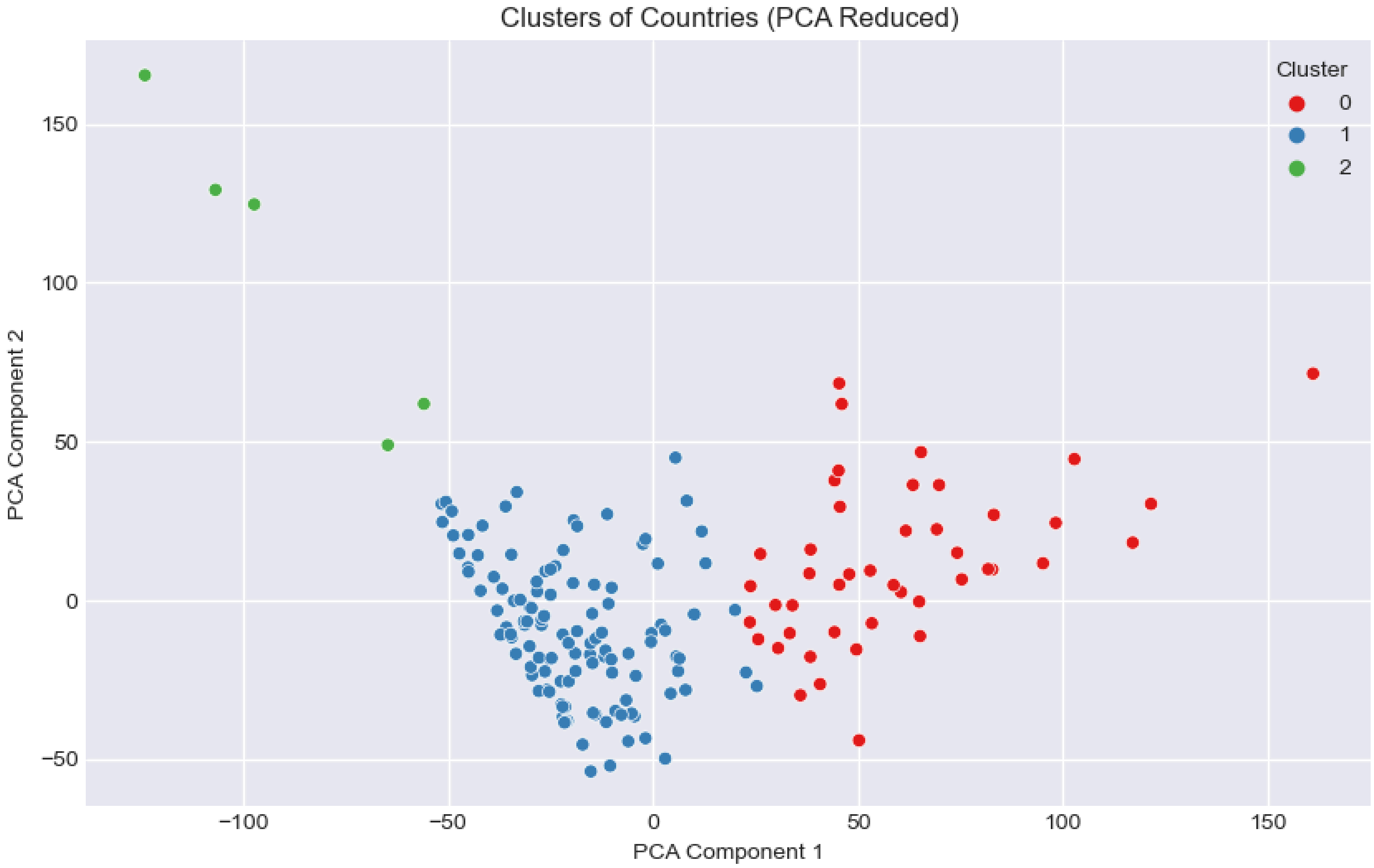
Visualize clusters using PCA for 2D plotting

In [21]:

```
pca = PCA(n_components=2)
pca_features = pca.fit_transform(data[['income', 'gdpp', 'inflation', 'child_mort', 'exports', 'health', 'imports', 'life_expec',
```

In [22]:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=pca_features[:, 0], y=pca_features[:, 1], hue=data['Cluster'], palette='Set1')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title('Clusters of Countries (PCA Reduced)')
plt.legend(title='Cluster')
plt.show()
```



In [107…
```
silhouette_avg = silhouette_score(data[['income', 'gdpp', 'inflation', 'child_mort', 'exports', 'health', 'imports', 'life_expec'
print(f'Silhouette Score for {optimal_k} clusters: {silhouette_avg:.2f}')
```

```
Silhouette Score for 3 clusters: 0.50
```

Feature importance analysis for clustering

In [24]:
```
cont_data = data
```

In [25]:
```
cont_data = cont_data.select_dtypes(include=[float, int])
```

In [26]:
```
cont_data.head()
```

Out[26]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	Cluster
0	90.2	10.0	7.58	44.9	-0.808245	0.157336	56.2	5.82	-0.679180	0
1	16.6	28.0	6.55	48.6	-0.375369	-0.312347	76.3	1.65	-0.485623	1
2	27.3	38.4	4.17	31.4	-0.220844	0.789274	76.5	2.89	-0.465376	1
3	119.0	62.3	2.85	42.9	-0.585043	1.387054	60.1	6.16	-0.516268	0
4	10.3	45.5	6.03	58.9	0.101732	-0.601749	76.8	2.13	-0.041817	1

In [48]:
```
cluster_summary = cont_data.groupby('Cluster').mean()
print("\nAverage values for each cluster:")
display(cluster_summary)
```

```
Average values for each cluster:
```

Cluster	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	95.180000	26.153756	6.229556	40.345909	-0.714221	0.381614	59.366667	4.986667	-0.619562
1	17.752137	42.422906	7.050598	45.731624	0.198002	-0.115199	74.471795	2.218376	0.160006
2	6.200000	144.960000	6.594000	132.900000	1.794737	-0.738877	79.620000	1.672000	1.831909

In [49]:
```
for cluster in range(optimal_k):
    print(f"\nCluster {cluster}:")
    print("Interpret characteristics and implications based on feature averages.")
```

```
Cluster 0:
Interpret characteristics and implications based on feature averages.

Cluster 1:
Interpret characteristics and implications based on feature averages.

Cluster 2:
Interpret characteristics and implications based on feature averages.
```

In [27]:
```
scaler = StandardScaler()
data_scaled = scaler.fit_transform(cont_data)
```

In [28]:
```
kmeans = KMeans(n_clusters=3, random_state=42)  # Using 3 clusters as chosen earlier
data['Cluster'] = kmeans.fit_predict(data_scaled)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
```

In [29]:
```
X = cont_data.drop('Cluster', axis=1)
y = cont_data['Cluster']
```

In [30]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [31]:
```
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)
```

Out[31]:
```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [32]:
```
perm_importance = permutation_importance(clf, X_test, y_test, random_state=42)
```

In [33]:
```
feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': perm_importance.importances_mean
}).sort_values(by='Importance', ascending=False)
```

In [34]:
```
print("Feature Importance in Determining Clusters:")
print(feature_importance_df)
```

```
Feature Importance in Determining Clusters:
      Feature  Importance
0  child_mort    0.156863
3     imports    0.019608
1     exports    0.015686
6  life_expec    0.003922
2      health    0.000000
4      income    0.000000
5   inflation    0.000000
8        gdpp    0.000000
7   total_fer   -0.015686
```

REVISED FITTING OF MODEL BASED ON FEATURE IMPORTANCE RESULTS

In [101…

```
important_features = ['child_mort', 'imports', 'exports', 'life_expec']
data_selected = data[important_features]
```

In [102…

```
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_selected)
```

In [103…

```
kmeans = KMeans(n_clusters=3, random_state=0)
data['Cluster'] = kmeans.fit_predict(data_scaled)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_
NUM_THREADS=1.
  warnings.warn(
```

In [108…

```
silhouette_avg = silhouette_score(data_scaled, data['Cluster'])
print(f"Silhouette Score with selected features: {silhouette_avg:.2f}")
```

```
Silhouette Score with selected features: 0.44
```

In [ ]:

In [ ]:

In [ ]: