Clustering the Countries by using Unsupervised Learning for HELP International

Objective:

To categorise the countries using socio-economic and health factors that determine the overall development of the country.

About organization:

HELP International is an international humanitarian NGO that is committed to fighting poverty and providing the people of backward countries with basic amenities and relief during the time of disasters and natural calamities.

Problem Statement:

HELP International have been able to raise around $ 10 million. Now the CEO of the NGO needs to decide how to use this money strategically and effectively. So, CEO has to make decision to choose the countries that are in the direst need of aid. Hence, your Job as a Data scientist is to categorise the countries using some socio-economic and health factors that determine the overall development of the country. Then you need to suggest the countries which the CEO needs to focus on the most.

In [1]:
```
import numpy as np
import pandas as pd
import datetime
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import colors
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt, numpy as np
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import AgglomerativeClustering
from matplotlib.colors import ListedColormap
from sklearn import metrics
from sklearn.metrics import silhouette_score
from matplotlib.cm import get_cmap
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import adjusted_rand_score
import warnings
# Ignore warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
data = pd.read_csv("Country-data.csv")
Data_Info = pd.read_csv("data-dictionary.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fe
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.8
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.6
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.8
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.1
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.1

In [4]:
```
data.shape
```

Out[4]: (167, 10)

In [5]:
```
data.info()
```

```
<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
```

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
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["country"].nunique()
```

Out[7]: 167

In [8]:
```
# Check if 'country' column has unique values
unique_country = data['country'].is_unique

print(f"Are all values in 'country' column unique? {unique_country}")
```

```
Are all values in 'country' column unique? True
```

In [9]:
```
data.describe()
```

Out[9]:

	child_mort	exports	health	imports	income	inflation	life_exp
count	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.0000
mean	38.270060	41.108976	6.815689	46.890215	17144.688623	7.781832	70.5556
std	40.328931	27.412010	2.746837	24.209589	19278.067698	10.570704	8.8931
min	2.600000	0.109000	1.810000	0.065900	609.000000	-4.210000	32.1000
25%	8.250000	23.800000	4.920000	30.200000	3355.000000	1.810000	65.3000
50%	19.300000	35.000000	6.320000	43.300000	9960.000000	5.390000	73.1000
75%	62.100000	51.350000	8.600000	58.750000	22800.000000	10.750000	76.8000
max	208.000000	200.000000	17.900000	174.000000	125000.000000	104.000000	82.8000

In [10]:
```
Data_Info
```

Out[10]:

	Column Name	Description
0	country	Name of the country
1	child_mort	Death of children under 5 years of age per 100...
2	exports	Exports of goods and services per capita. Give...
3	health	Total health spending per capita. Given as %ag...
4	imports	Imports of goods and services per capita. Give...
5	Income	Net income per person
6	Inflation	The measurement of the annual growth rate of t...
7	life_expec	The average number of years a new born child w...
8	total_fer	The number of children that would be born to e...
9	gdpp	The GDP per capita. Calculated as the Total GD...

In [11]:

```
df = data.copy()
later_df = data.copy()
```

In [12]:

```
df.head()
```

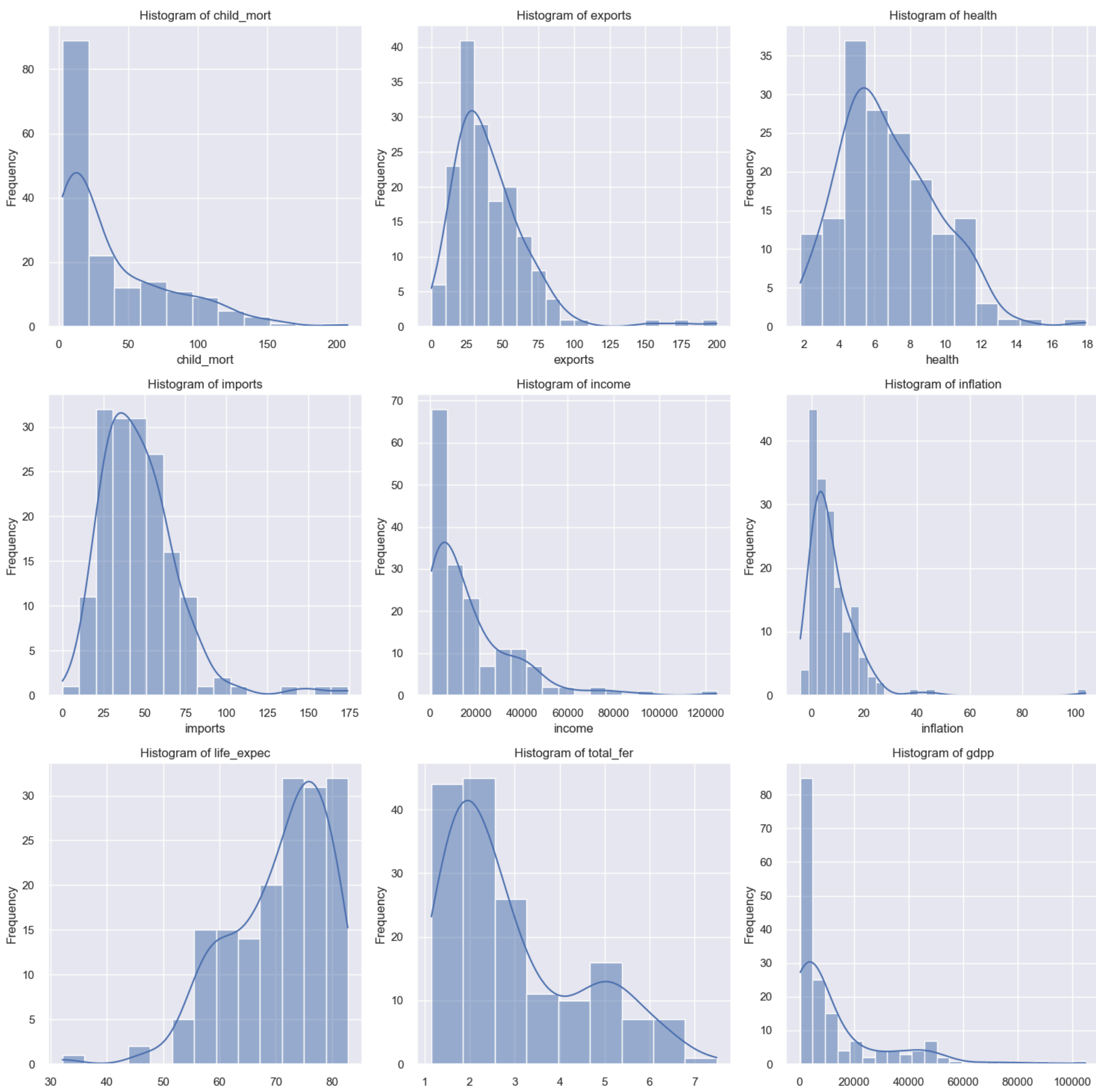
Out[12]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fe
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.8
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.6
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.8
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.1
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.1

In [13]:

```
# Set the style
sns.set(style='darkgrid')

# Step 2: Select numeric columns
numeric_columns = df.select_dtypes(include=['number']).columns

# Calculate the number of rows and columns for subplots
num_rows = len(numeric_columns) // 3 + (1 if len(numeric_columns) % 3 > 0 else 0)
num_cols = min(3, len(numeric_columns))

# Create subplots
fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 5 * num_rows))

# Flatten the axes array for easier indexing
```

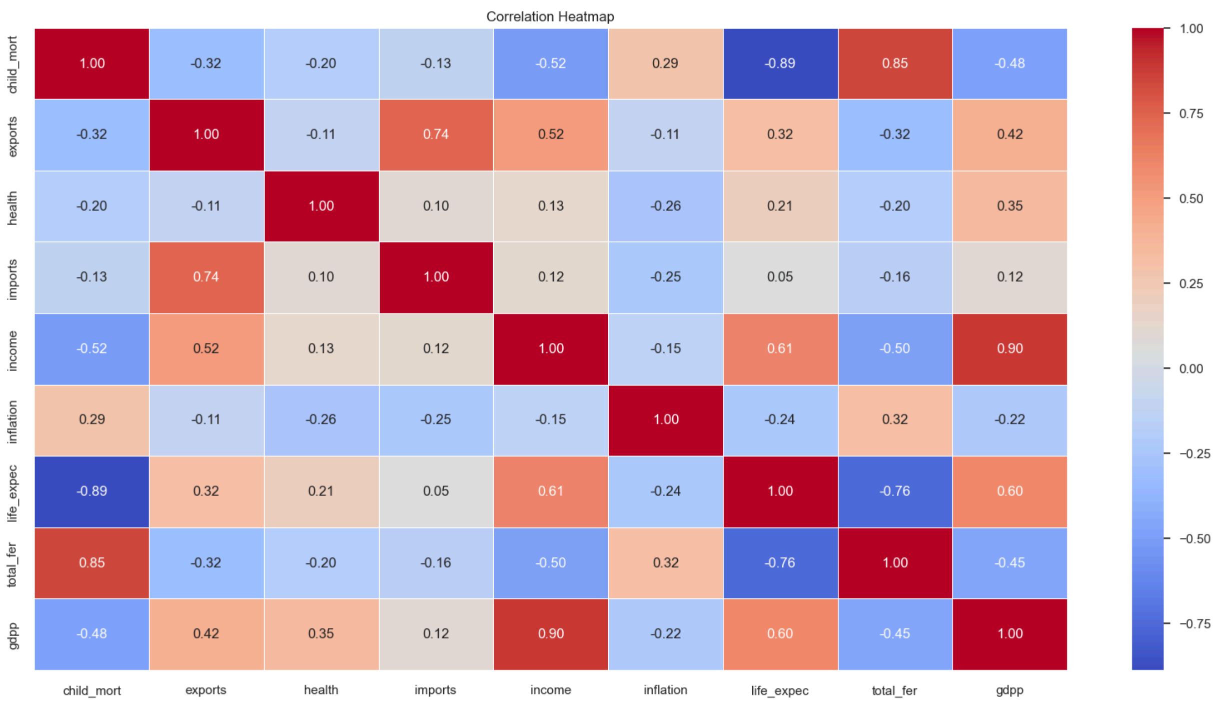
```
if num_rows * num_cols > 1:
    axes = axes.flatten()
else:
    axes = [axes]

# Plot histograms for each numeric column
for i, column in enumerate(numeric_columns):
    sns.histplot(df[column], ax=axes[i], kde=True)
    axes[i].set_title('Histogram of {}'.format(column))
    axes[i].set_xlabel(column)
    axes[i].set_ylabel('Frequency')

# Hide empty subplots
for j in range(i + 1, len(axes)):
    axes[j].axis('off')

plt.tight_layout()
plt.show()
```



In [14]:
```python
#  Drop the original "country" column from df
df = df.drop('country', axis=1)
```

Correlation check Using HeatMap

In [15]:
```python
# Calculate the correlation matrix
correlation_matrix =df.corr()

# Plot the heatmap
plt.figure(figsize=(20, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=
plt.title('Correlation Heatmap')
plt.show()


#Save the image below
plt.savefig("Correlation Heatmap.png",dpi = 300,bbox_inches ="tight")
```




```
<Figure size 800x550 with 0 Axes>
```

Standardization of the Dataset

In [16]:
```python
# Standardize the numerical columns
scaler = StandardScaler()
scaled_df = scaler.fit_transform(df)

# Convert scaled array back to DataFrame
scaled_df = pd.DataFrame(scaled_df, columns=df.columns)
```

PCA

In [17]:

```
#Initiating PCA to reduce dimensions aka features to 10
pca = PCA(n_components=5)
pca.fit(scaled_df)
PCA_use = pd.DataFrame(pca.transform(scaled_df), columns=(["PCA1","PCA2", "PCA3", "
PCA_use.head()
```

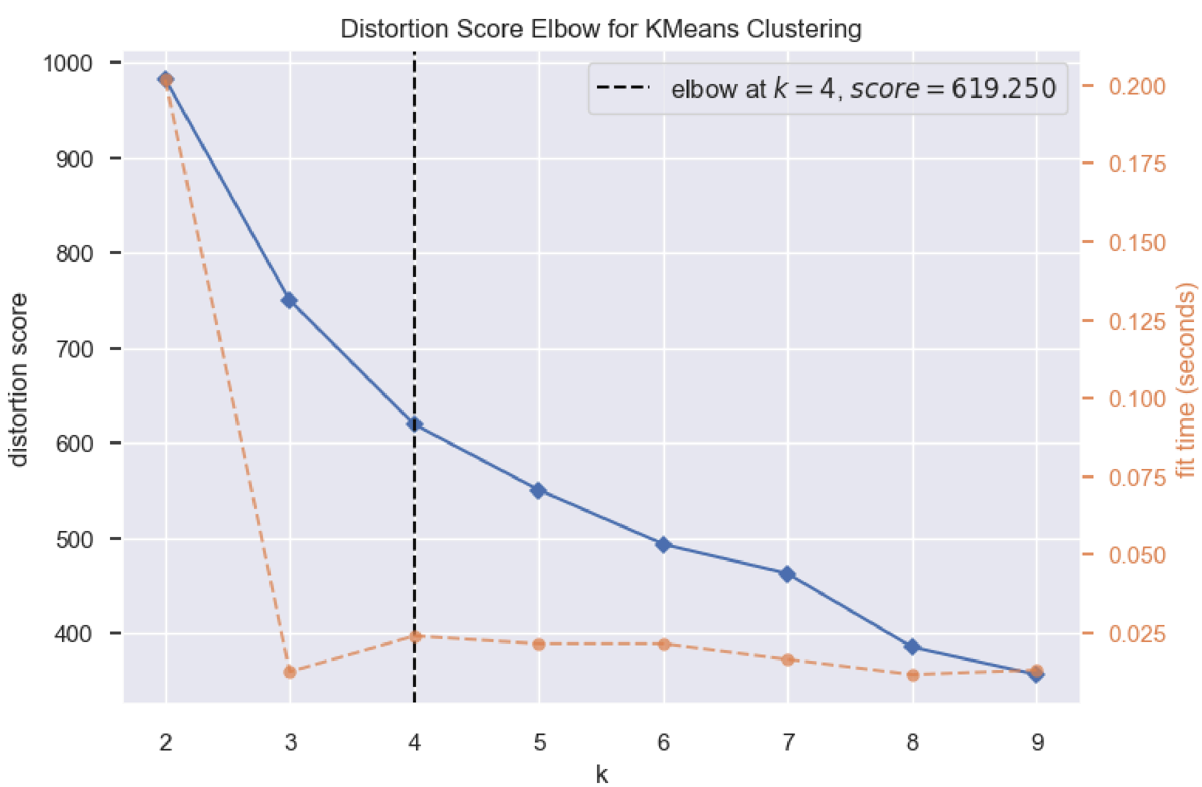
Out[17]:

	PCA1	PCA2	PCA3	PCA4	PCA5
0	-2.913025	0.095621	-0.718118	1.005255	-0.158310
1	0.429911	-0.588156	-0.333486	-1.161059	0.174677
2	-0.285225	-0.455174	1.221505	-0.868115	0.156475
3	-2.932423	1.695555	1.525044	0.839625	-0.273209
4	1.033576	0.136659	-0.225721	-0.847063	-0.193007

ELBOW Method

In [18]:

```
model = KMeans(random_state=42)
visualizer = KElbowVisualizer(model, k=(2, 10))  # Search for optimal k from 2 to 1
visualizer.fit(PCA_use)

#Save the image below
plt.savefig("Elbow Graph.png",dpi = 300,bbox_inches ="tight")

visualizer.show()

print("Optimal number of clusters:", visualizer.elbow_value_)
```



```
Optimal number of clusters: 4
```

Performing K- Means on Original Data

In [19]:
```
#Perform KMeans Clustering
kmeans = KMeans(n_clusters=visualizer.elbow_value_, random_state=42)  # Assuming yo
kmeans.fit(scaled_df)
labels_original = kmeans.labels_
```

In [20]:
```
# Evaluate Clustering
silhouette_avg_original = silhouette_score(scaled_df.iloc[:, :-1], labels_original)
print(f'Silhouette Score: {silhouette_avg_original}')
```

```
Silhouette Score: 0.25455983233016954
```

Performing K-Means clustering on PCA data

In [21]:
```
#Initiating PCA to reduce dimensions aka features to 10
pca = PCA(n_components=5)
pca.fit(scaled_df)
PCA_use = pd.DataFrame(pca.transform(scaled_df), columns=(["PCA1","PCA2", "PCA3", "
PCA_use.head()
```

Out[21]:

	PCA1	PCA2	PCA3	PCA4	PCA5
0	-2.913025	0.095621	-0.718118	1.005255	-0.158310
1	0.429911	-0.588156	-0.333486	-1.161059	0.174677
2	-0.285225	-0.455174	1.221505	-0.868115	0.156475
3	-2.932423	1.695555	1.525044	0.839625	-0.273209
4	1.033576	0.136659	-0.225721	-0.847063	-0.193007

In [22]:
```
#Perform KMeans Clustering
kmeans = KMeans(n_clusters=visualizer.elbow_value_, random_state=42)  # Assuming yo
kmeans.fit(PCA_use)
labels = kmeans.labels_
```

In [23]:
```
# Add the cluster labels to the PCA DataFrame
PCA_use['Cluster'] = labels
PCA_use.head()
```

Out[23]:

	PCA1	PCA2	PCA3	PCA4	PCA5	Cluster
0	-2.913025	0.095621	-0.718118	1.005255	-0.158310	2
1	0.429911	-0.588156	-0.333486	-1.161059	0.174677	0
2	-0.285225	-0.455174	1.221505	-0.868115	0.156475	0
3	-2.932423	1.695555	1.525044	0.839625	-0.273209	2
4	1.033576	0.136659	-0.225721	-0.847063	-0.193007	0

In [24]:

```
# Evaluate Clustering
silhouette_avg = silhouette_score(PCA_use.iloc[:, :-1], labels)
print(f'Silhouette Score: {silhouette_avg}')
```

```
Silhouette Score: 0.3286884389759583
```

Adjusted Rand Index (ARI) Calculation and Analysis

Explanation:

The Adjusted Rand Index (ARI) is a measure of similarity between two data clusterings. It adjusts for the chance grouping of elements and ranges from -1 to 1:

- 1: Perfect agreement between the two clusterings.
- 0: The clustering is random, no agreement.
- Negative values: Indicates less agreement than expected by chance.

Analysis of the Result:

In this context, the ARI is used to compare the original clustering of countries with the clustering obtained after applying PCA (Principal Component Analysis) transformation. The PCA transformation is used to reduce the dimensionality of the data while retaining most of the variability.

In [25]:

```
# Calculate Adjusted Rand Index (ARI)
ari_score = adjusted_rand_score(labels_original, labels)
print("Adjusted Rand Index between Original and PCA-transformed Data:", ari_score)
```

```
Adjusted Rand Index between Original and PCA-transformed Data: 0.9249402789621708
```

Interpretation:

- High ARI Score (0.9249): The ARI score of approximately 0.925 indicates a very high level of agreement between the original clustering and the PCA-transformed clustering.

Implications:

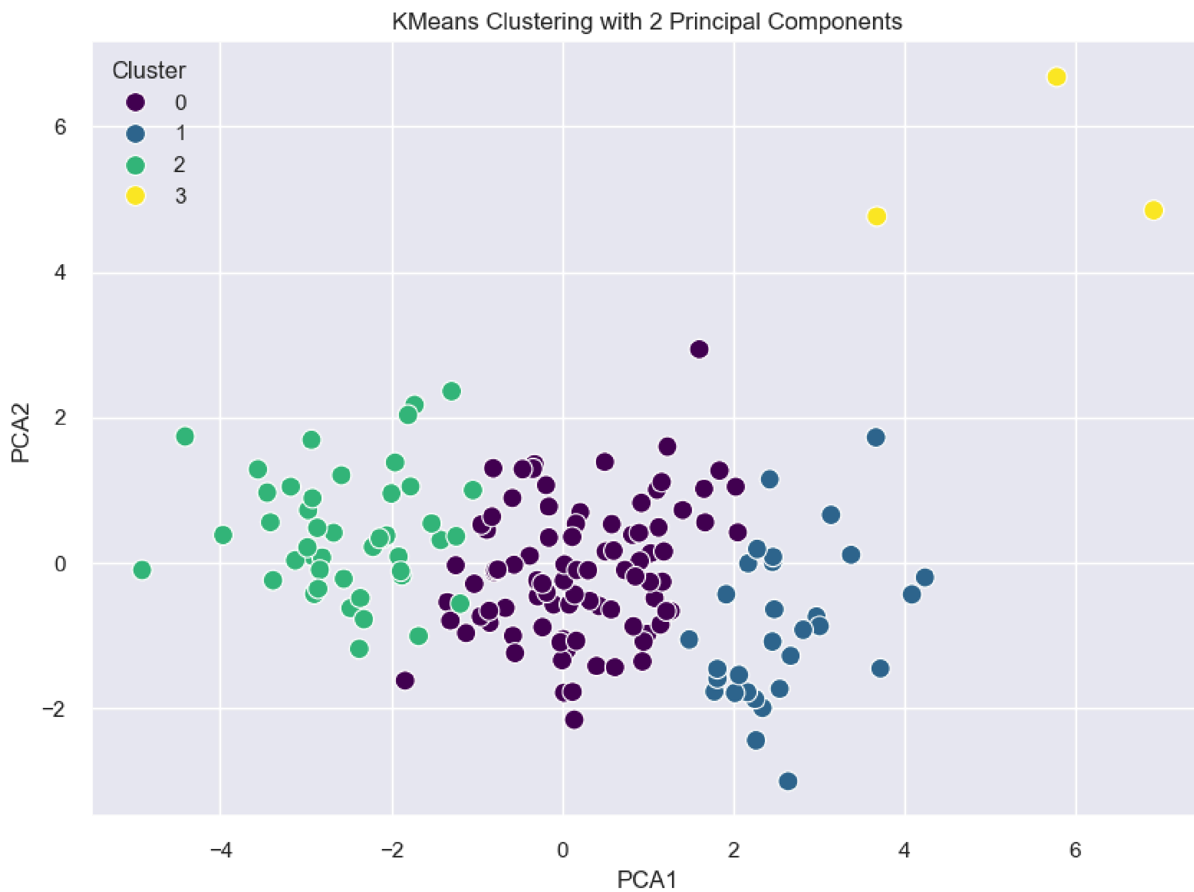
- Effectiveness of PCA: This high ARI score suggests that the PCA transformation has preserved the essential structure of the data. The reduced-dimensionality data still captures the same clusters as the original high-dimensional data. Robustness of

Clustering: The clustering algorithm used is robust, and the clusters identified are not significantly affected by the dimensionality reduction process.

Conclusion:

The high Adjusted Rand Index score of 0.9249 demonstrates that the clustering of countries is consistent before and after PCA transformation, affirming the reliability of the clustering results and the effectiveness of PCA in maintaining the integrity of the data structure.

In [26]:
```
#Visualize Clusters
plt.figure(figsize=(10, 7))
sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', data=PCA_use, palette='viridis',
plt.title('KMeans Clustering with 2 Principal Components')
plt.show()

#Save the image below
plt.savefig("Clustering with 2 Principal Components.png",dpi = 300,bbox_inches ="ti
```




```
<Figure size 800x550 with 0 Axes>
```

In [27]:
```
# Add cluster labels to the original DataFrame or a copy of it
later_df['cluster'] = labels
later_df.head()
```

Out[27]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fe
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.8
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.6
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.8
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.1
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.1

In [28]:

```
cluster_counts =later_df["cluster"].value_counts()
cluster_counts
```

Out[28]:

```
cluster
0    89
2    45
1    30
3     3
Name: count, dtype: int64
```

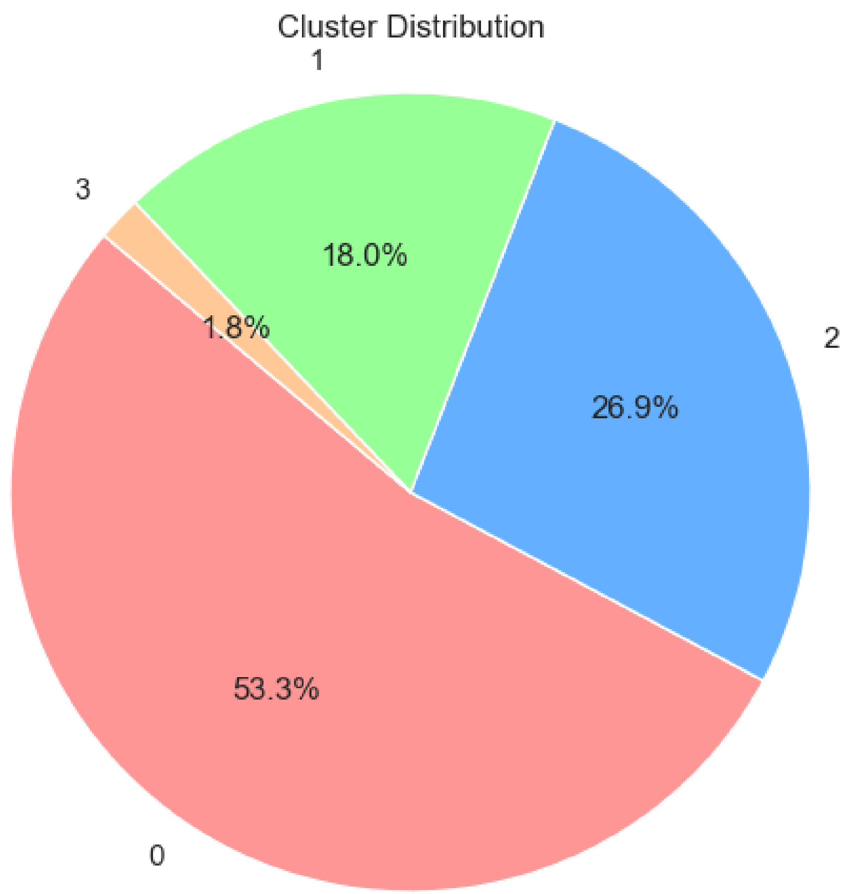
In [29]:

```
# Calculate the mean Income, and Amount each cluster spent
cluster_summary_PCA = later_df.groupby('cluster').agg({
    'income': 'mean',
    'gdpp': 'mean',
    'child_mort': 'mean',
    'inflation': 'mean',
    'health': 'mean',
    'life_expec': 'mean',
    'total_fer': 'mean',
    'exports': 'mean',
    'imports': 'mean'



}).reset_index()

# Print the cluster summary
cluster_summary_PCA
```

Out[29]:

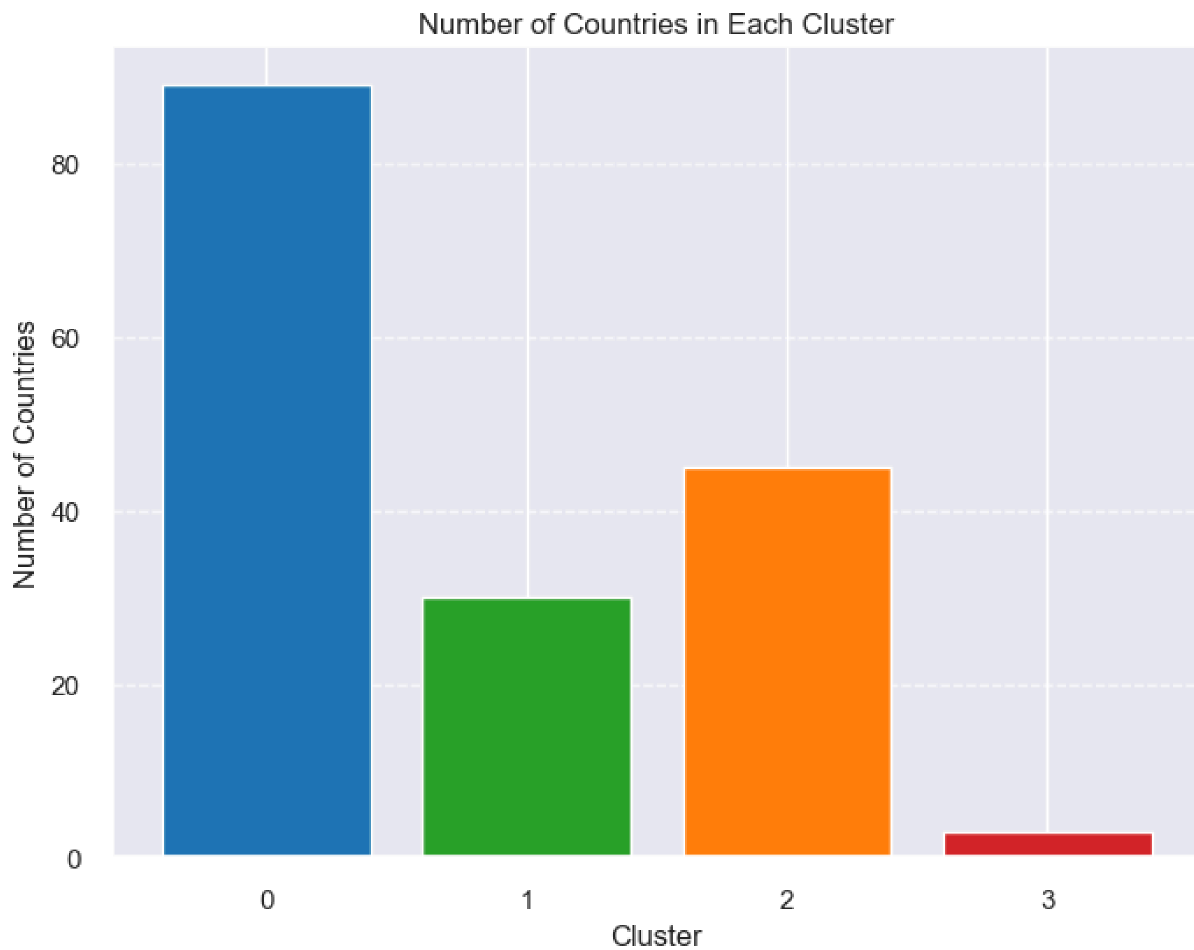
	cluster	income	gdpp	child_mort	inflation	health	life_expec	total_fe
0	0	12969.325843	6885.528090	21.913483	7.533607	6.283483	72.693258	2.31876
1	1	45250.000000	43333.333333	4.953333	2.742200	9.168667	80.376667	1.79533
2	2	3539.844444	1766.711111	95.106667	11.986778	6.301111	59.055556	5.06533
3	3	64033.333333	57566.666667	4.133333	2.468000	6.793333	81.433333	1.38000

In [30]:
```
# Plotting the pie chart
plt.figure(figsize=(8, 6))
plt.pie(cluster_counts, labels=cluster_counts.index, autopct='%1.1f%%', startangle=
plt.title('Cluster Distribution')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()

#Save the image below
plt.savefig("Pie-chart PCA Cluster Distribution.png",dpi = 300,bbox_inches ="tight"
```




```
<Figure size 800x550 with 0 Axes>
```

In [31]:
```
# Plotting the bar chart
plt.figure(figsize=(8, 6))
plt.bar(cluster_counts.index, cluster_counts.values, color=['#1f77b4', '#ff7f0e', '
plt.xlabel('Cluster')
plt.ylabel('Number of Countries')
plt.title('Number of Countries in Each Cluster')
plt.xticks(cluster_counts.index)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

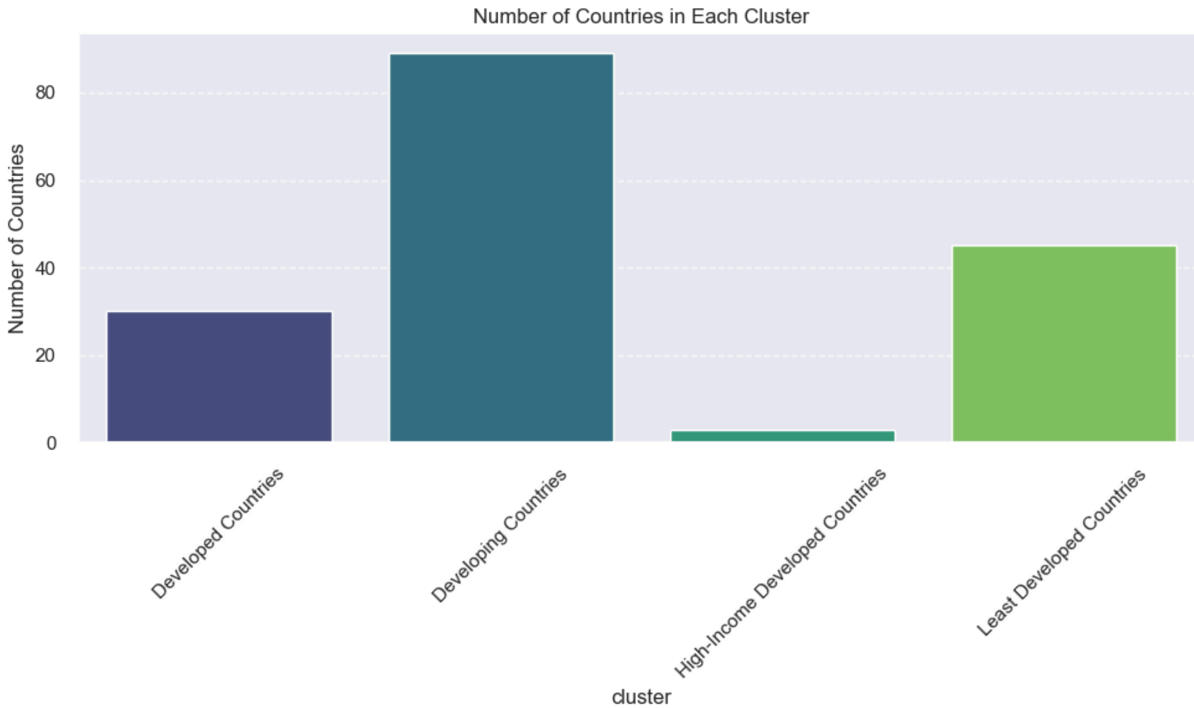


In [32]:
```
# Assigning cluster names based on summary
cluster_names = {
    0: 'Developing Countries',
    1: 'Developed Countries',
    2: 'Least Developed Countries',
    3: 'High-Income Developed Countries'
}
```

In [33]:
```
# Map cluster names to the 'cluster' column
later_df['cluster_name'] = later_df['cluster'].map(cluster_names)
```

In [34]:
```
# Count number of countries in each cluster
cluster_counts = later_df['cluster_name'].value_counts().sort_index()

# Plotting the bar chart using seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x=cluster_counts.index, y=cluster_counts.values, palette='viridis')
plt.xlabel('cluster')
plt.ylabel('Number of Countries')
plt.title('Number of Countries in Each Cluster')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
#Save the image below
plt.savefig("Bar-chart  Cluster Distribution.png",dpi = 300,bbox_inches ="tight")
```




```
<Figure size 800x550 with 0 Axes>
```

Analysis Summary with Assigned Cluster Names:

Cluster 0: Developing Countries

Income: $12,969GDPP$:6,886 Child Mortality: 21.91 per 1000 births Inflation: 7.53% Health: 6.28% of GDP Life Expectancy: 72.69 years Total Fertility: 2.32 children per woman Exports: 41.30% of GDP Imports: 47.92% of GDP

Cluster 1: Developed Countries

Income: $45,250GDPP$:43,333 Child Mortality: 4.95 per 1000 births Inflation: 2.74% Health: 9.17% of GDP Life Expectancy: 80.38 years Total Fertility: 1.80 children per woman Exports: 45.83% of GDP Imports: 39.74% of GDP

Cluster 2: Least Developed Countries

Income: $3,540GDPP$:1,767 Child Mortality: 95.11 per 1000 births Inflation: 11.99% Health: 6.30% of GDP Life Expectancy: 59.06 years Total Fertility: 5.07 children per woman Exports: 28.60% of GDP Imports: 42.31% of GDP

Cluster 3: High-Income Developed Countries

Income: $64,033 GDPP$:57,567 Child Mortality: 4.13 per 1000 births Inflation: 2.47% Health: 6.79% of GDP Life Expectancy: 81.43 years Total Fertility: 1.38 children per woman Exports: 176.00% of GDP Imports: 156.67% of GDP

These names reflect the overall performance and characteristics of each cluster, providing a clearer understanding of the economic and social conditions in each group of countries.

In [35]:
```
# Create a pivot table with country as index and cluster values as columns
df_pivot = later_df.pivot_table(index='country', columns='cluster', aggfunc='size',

# Reset index to turn the index into a column
df_pivot.reset_index(inplace=True)

# Rename columns to include 'cluster_' prefix
df_pivot.columns = ['country'] + [f'cluster_{col}' for col in df_pivot.columns[1:]]

# Assign DataFrames for each cluster to separate variables
Developing = df_pivot[df_pivot['cluster_0'] > 0][['country', 'cluster_0']].reset_in
Developed = df_pivot[df_pivot['cluster_1'] > 0][['country', 'cluster_1']].reset_ind
Underdeveloped = df_pivot[df_pivot['cluster_2'] > 0][['country', 'cluster_2']].rese
High_Income = df_pivot[df_pivot['cluster_3'] > 0][['country', 'cluster_3']].reset_i
```

In [36]:
```
# Print list of countries in each cluster
print("Developing Countries:")
print(Developing)
```

```
Developing Countries:
                country  cluster_0
0               Albania          1
1               Algeria          1
2   Antigua and Barbuda          1
3             Argentina          1
4               Armenia          1
..                  ...        ...
84              Uruguay          1
85           Uzbekistan          1
86              Vanuatu          1
87            Venezuela          1
88              Vietnam          1

[89 rows x 2 columns]
```

In [37]:
```
print("\nDeveloped Countries:")
print(Developed)
```

```
Developed Countries:
                 country  cluster_1
0              Australia          1
1                Austria          1
2                Belgium          1
3                 Brunei          1
4                 Canada          1
5                 Cyprus          1
6                Denmark          1
7                Finland          1
8                 France          1
9                Germany          1
10                Greece          1
11               Iceland          1
12               Ireland          1
13                Israel          1
14                 Italy          1
15                 Japan          1
16                Kuwait          1
17           Netherlands          1
18           New Zealand          1
19                Norway          1
20              Portugal          1
21                 Qatar          1
22              Slovenia          1
23           South Korea          1
24                 Spain          1
25                Sweden          1
26           Switzerland          1
27  United Arab Emirates          1
28        United Kingdom          1
29         United States          1
```

In [38]:

```
print("\nDeveloped Countries:")
print(Underdeveloped)
```

```
Developed Countries:
                     country  cluster_2
0                Afghanistan          1
1                     Angola          1
2                      Benin          1
3               Burkina Faso          1
4                    Burundi          1
5                   Cameroon          1
6   Central African Republic          1
7                       Chad          1
8                    Comoros          1
9           Congo, Dem. Rep.          1
10               Congo, Rep.          1
11             Cote d'Ivoire          1
12         Equatorial Guinea          1
13                   Eritrea          1
14                     Gabon          1
15                    Gambia          1
16                     Ghana          1
17                    Guinea          1
18             Guinea-Bissau          1
19                     Haiti          1
20                     Kenya          1
21                  Kiribati          1
22                       Lao          1
23                   Lesotho          1
24                   Liberia          1
25                Madagascar          1
26                    Malawi          1
27                      Mali          1
28                Mauritania          1
29                Mozambique          1
30                   Namibia          1
31                     Niger          1
32                   Nigeria          1
33                  Pakistan          1
34                    Rwanda          1
35                   Senegal          1
36              Sierra Leone          1
37              South Africa          1
38                     Sudan          1
39                  Tanzania          1
40               Timor-Leste          1
41                      Togo          1
42                    Uganda          1
43                     Yemen          1
44                    Zambia          1
```

In [39]:
```
print("\nDeveloped Countries:")
print(High_Income)
```

```
Developed Countries:
      country  cluster_3
0  Luxembourg          1
1       Malta          1
2   Singapore          1
```

```
In [40]: Developing.to_csv('PCA_Developing.csv', index=False)
Developed.to_csv('PCA_Developed.csv', index=False)
High_Income.to_csv('PCA_High_Income.csv', index=False)
Underdeveloped.to_csv('PCA_Underdeveloped.csv', index=False)
```

Recommendations:

Primary Focus: Least Developed Countries (Cluster 2)

Countries in Cluster 2 (Least Developed Countries) should be the primary focus for HELP International due to the following reasons:

Lowest Income and GDPP: These countries have the lowest income and GDP per capita, indicating severe economic challenges. Highest Child Mortality Rates: Extremely high child mortality rates suggest significant health and nutrition issues. High Inflation: Indicates economic instability. Low Life Expectancy: Reflects poor overall health and living conditions. High Total Fertility Rates: Points to potential population growth challenges.

Suggested Actions:

Basic Health Services: Improve access to basic health services to reduce child mortality and improve overall health outcomes. Economic Stability: Programs to stabilize inflation and boost economic development. Education and Family Planning: Initiatives to provide education and promote family planning to control high fertility rates. Infrastructure Development: Investments in infrastructure to improve living conditions and economic opportunities. Secondary Focus: Developing Countries (Cluster 0) Countries in Cluster 0 (Developing Countries) also require significant attention, although their situation is not as dire as those in Cluster 2.

Suggested Actions:

Health and Nutrition Programs: Targeted health and nutrition programs to continue improving child mortality rates and life expectancy. Economic Development: Support economic initiatives that help sustain moderate income levels and improve GDP per capita. Inflation Control: Policies and programs to control moderate inflation rates and ensure economic stability.

Conclusion:

By focusing primarily on the Least Developed Countries (Cluster 2) and secondarily on Developing Countries (Cluster 0), HELP International can strategically allocate their $10 million to maximize impact and effectively aid those in the most dire need. This approach will help improve socio-economic and health conditions in the countries that need it the most.