Socio-economic Factors for Geographic Clustering

Context

The study of socio-economic factors is foundational to understanding and shaping the future of societies and hence of extreme interest to various government and non-government institutions. While GDP is one of the important measures used in one of the popular economic vernacular, it is not the only measure of the growth and the state of an economy. This case study aims to deep dive into one such dataset that contains various socio-economic attributes for countries around the world.

Objective

To identify if there exist various clusters of countries that are more similar to each other than others, in terms of certain socio-economic factors.

Data Dictionary

The data has the following attributes:

- country: Name of the country
- child_mort: Death of children under 5 years of age per 1000 live births
- exports - Exports in % of the GDP per capita
- health - The total spend on health given as % of GDP
- imports - The value of imports given as % of GDP per capita
- income - The net income per person
- inflation - Inflation rate %
- life_expec - Average life expectancy in years
- total_fer - The fertility rate - Average children per woman in the country
- gdpp - GDP per capita

In the dataset, we will not do clustering on the GDP. We will rather try to understand the variation of other factors with GDP across the groups that we get.

Importing the libraries and overview of the dataset

Note: Please make sure you have installed the sklearn_extra library before running the below cell. If you have not installed the library, please run the below code to install the library:

!pip install scikit-learn-extra

Clustering

Context

The study of socio-economic factors is foundational to understanding and shaping the future of societies and hence of extreme interest to various government and non-government institutions. While GDP is one of the most popular measures used in popular vernacular, it is not the only measure of the growth and the state of an economy. This case study aims to deep dive into one such dataset that contains various socio-economic attributes for countries around the world.

Objective

To identify if there exist various clusters of countries that are more similar to each other in terms of certain socio-economic factors

Data Dictionary

The data has the following attributes:

country - Name of the country child_mort - Death of children under 5 years of age per 1000 live births exports - Exports in % age of the GDP per capita health - The total spend on health given as % of GDP imports - The value of imports given as % of GDP per capita income - The net income per person inflation - Inflation rate % life_expec - Average life expectancy in years total_fer - The fertility rate - Average children per woman in the country gdpp - GDP per capita

In [4]:
```
#!pip install scikit-learn-extra
```

In [5]:
```
import pandas as pd

import numpy as np

import matplotlib.pylab as plt

import seaborn as sns
```

```
# To scale the data using z-score
from sklearn.preprocessing import StandardScaler

# Importing clustering algorithms
from sklearn.cluster import KMeans

from sklearn.mixture import GaussianMixture

from sklearn_extra.cluster import KMedoids

from sklearn.cluster import AgglomerativeClustering

from sklearn.cluster import DBSCAN

# Silhouette score
from sklearn.metrics import silhouette_score

import warnings
warnings.filterwarnings("ignore")
```

Loading the data

In [6]:
```
data = pd.read_csv("Country-data.csv")

data.head()
```

Out[6]:

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

Checking the info of the data

In [7]:
```
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
```

Observations:

- There are **167 observations and 10 columns** in the data.
- All columns have 167 non-null values, i.e., there are **no missing values**.
- All the **columns except the country name are numerical**.
- Everything looks great, let's move ahead to check for duplicates.

Check duplicate entries

In [8]:
```
data[data.duplicated()]
```

Out[8]:

country	child_mort	exports	health	imports	income	inflation	life_expec	total_fe

- There are **no duplicate rows** in the data. That's good.

Exploratory Data Analysis

Summary Statistics

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

Out[9]:

	count	mean	std	min	25%	50%	75
child_mort	167.0	38.270060	40.328931	2.6000	8.250	19.30	62.
exports	167.0	41.108976	27.412010	0.1090	23.800	35.00	51.
health	167.0	6.815689	2.746837	1.8100	4.920	6.32	8.
imports	167.0	46.890215	24.209589	0.0659	30.200	43.30	58.
income	167.0	17144.688623	19278.067698	609.0000	3355.000	9960.00	22800.
inflation	167.0	7.781832	10.570704	-4.2100	1.810	5.39	10.
life_expec	167.0	70.555689	8.893172	32.1000	65.300	73.10	76.
total_fer	167.0	2.947964	1.513848	1.1500	1.795	2.41	3.
gdpp	167.0	12964.155689	18328.704809	231.0000	1330.000	4660.00	14050.

Observations:

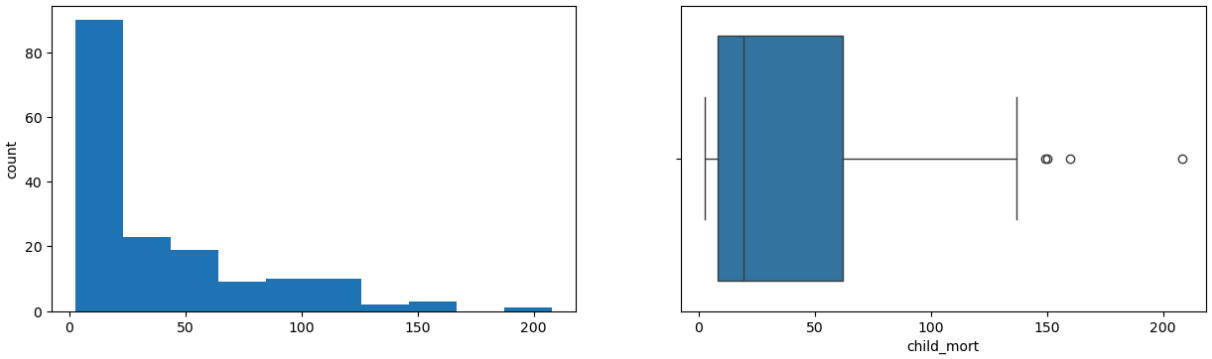
- The child mortality rate has a high range from 2.6 to 208 deaths per 1000 live births. The average child mortality rate is approx 38 deaths per 1000 live births.
- Similarly, the exports and imports have a high range of values. The maximum values for exports and imports are 200% and 174% of GDP respectively. This can happen if a country's export or import industry exceeds its domestic economy.
- The total spend on health is very less in comparison to exports and imports for the majority of countries. The average spending on health is approx 6.8% of GDP.
- The average life expectancy is approx 70 years but the minimum value is just 32 years.
- Other variables like gdpp, inflation, and income also show a high variability which can be expected as they can be very different for different countries.
- Overall, % expenditure on health and average life expectancy seem to have a lesser standard deviation, which reflects less variability across countries. All other variables seem to have a very high spread across countries. These are the variables that might help us identify the clusters if they exist.

Let's check the distribution and outliers for each column in the data

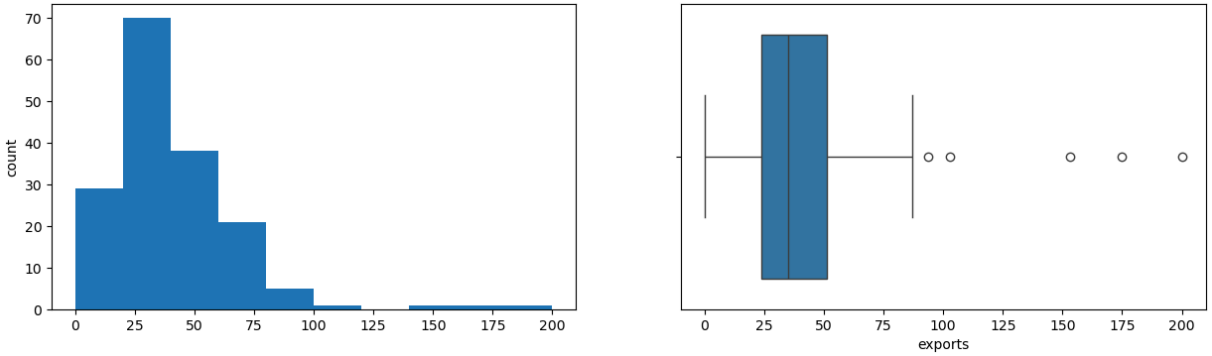
In [10]:
```
for col in data.columns[1:]:
    print(col)

    print('Skew :', round(data[col].skew(), 2))

    plt.figure(figsize = (15, 4))

    plt.subplot(1, 2, 1)

    data[col].hist(bins = 10, grid = False)
```

```
    plt.ylabel('count')

    plt.subplot(1, 2, 2)

    sns.boxplot(x = data[col])

    plt.show()
```

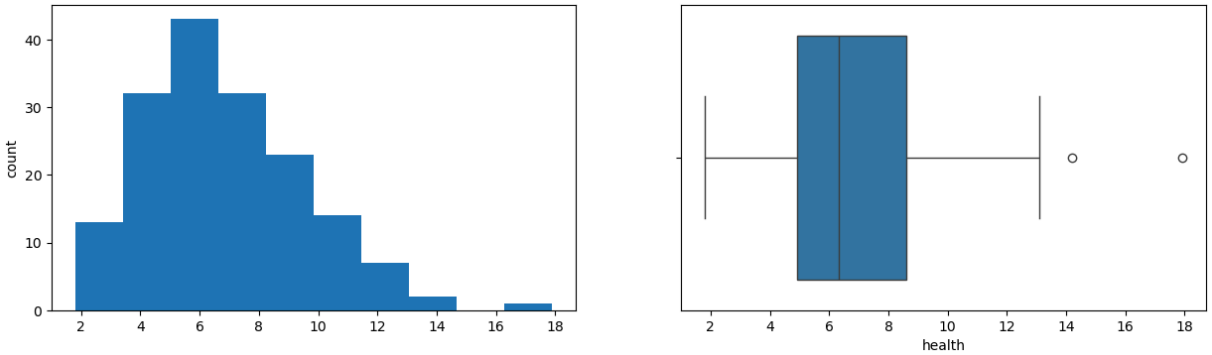
child_mort
Skew : 1.45



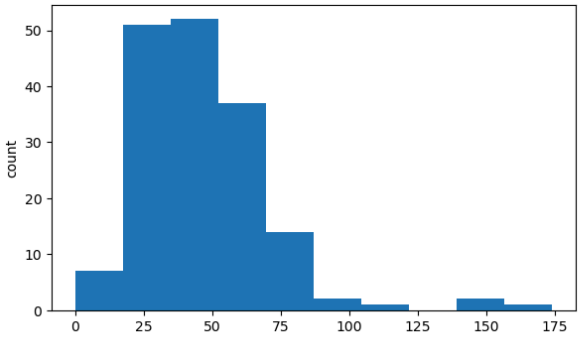
exports
Skew : 2.45

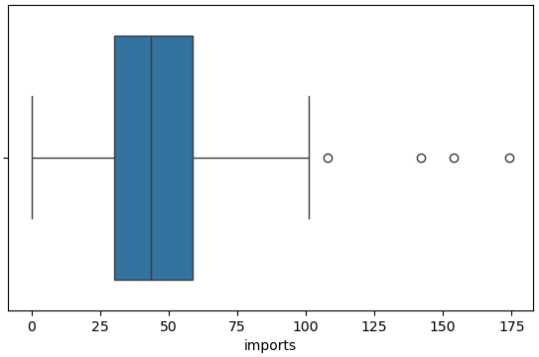


health
Skew : 0.71

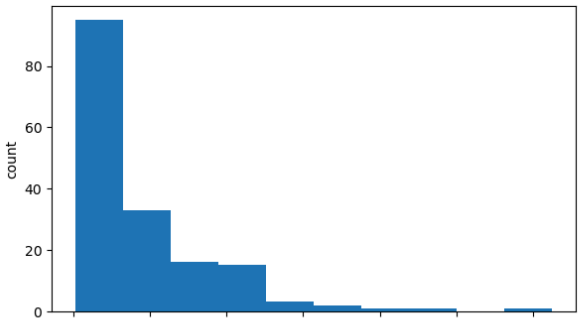


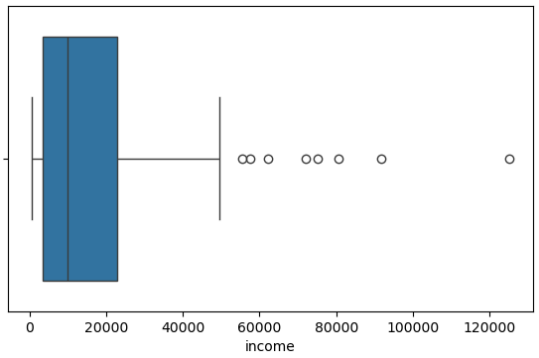
imports
Skew : 1.91



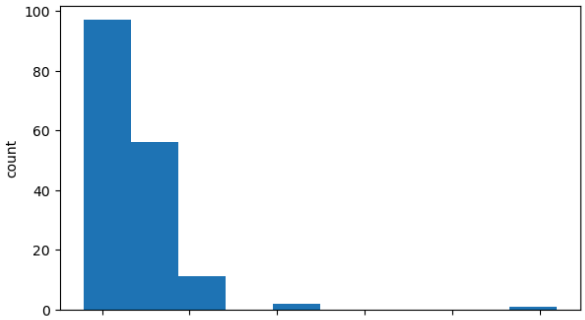


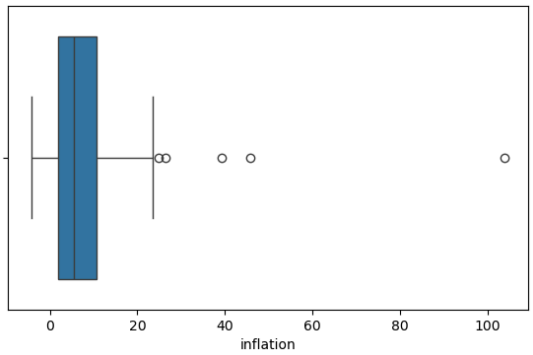
income
Skew : 2.23



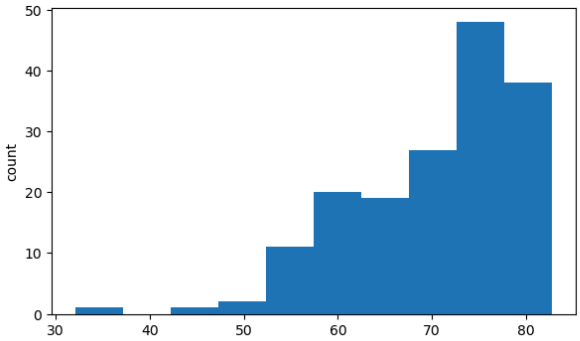


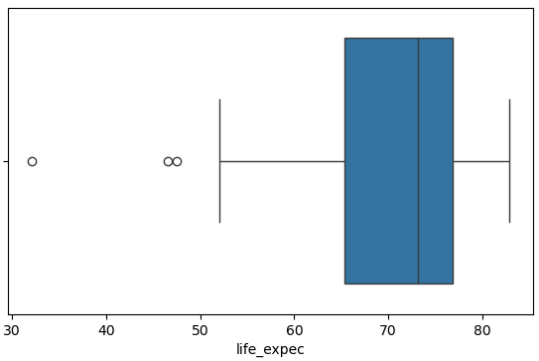
inflation
Skew : 5.15



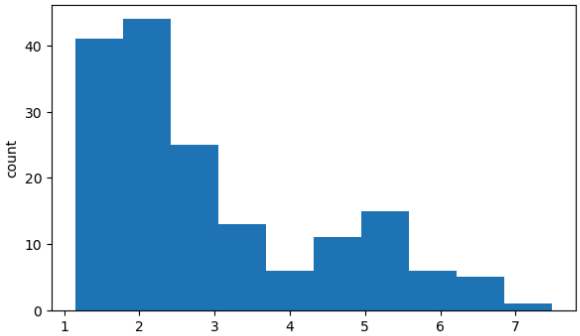


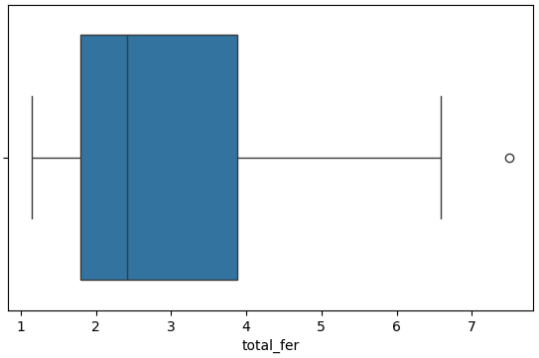
life_expec
Skew : -0.97



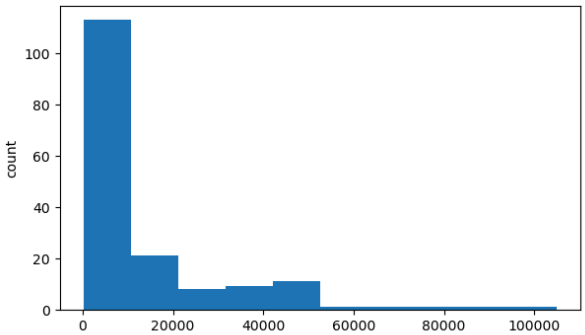


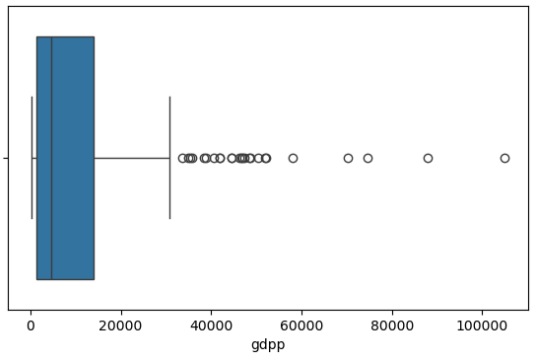
total_fer
Skew : 0.97





gdpp
Skew : 2.22





Observations:

- As observed earlier, most of the variables have skewed distributions.
- The distribution for the % expenditure on health is relatively less skewed with fewer outliers.
- The life expectancy is the only variable which is skewed to the left meaning most of the countries have already been successful in achieving high life expectancy.
- The distribution for all other variables is highly skewed to the right. All these variables have some outliers to the right end.

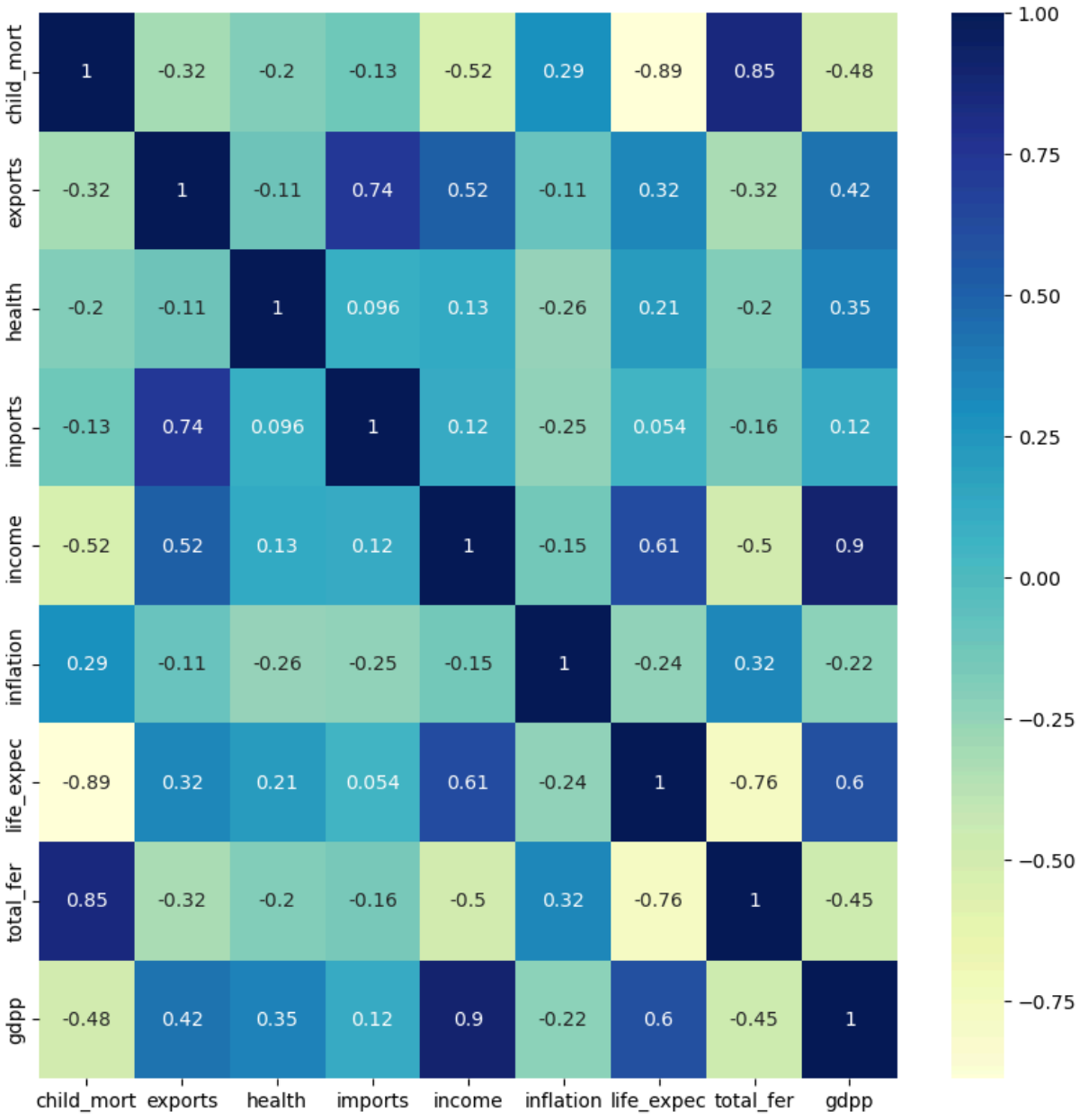
Let's check the correlation among the variables

In [11]:
```
data.select_dtypes(include = "number").columns.to_list()
```

Out[11]:
```
['child_mort',
 'exports',
 'health',
 'imports',
 'income',
 'inflation',
 'life_expec',
 'total_fer',
 'gdpp']
```

In [12]:
```
plt.figure(figsize  = (10, 10))

num_cols = data.select_dtypes(include = "number").columns.to_list()
```

```
sns.heatmap(data[num_cols].corr(), annot = True, cmap = "YlGnBu")

plt.show()
```



Observations:

- There is a strong positive correlation between gdpp and income. This makes sense.
- The life expectancy is positively correlated with gdpp. This indicates that people live longer in richer countries.
- There is a strong negative correlation between life expectancy and child mortality. This is understandable.
- The child mortality is also seen to have a strong positive correlation with the fertility rate.

Scaling the data

- Clustering algorithms are distance-based algorithms, and all distance-based algorithms are affected by the scale of the variables. Therefore, we will scale the data before applying clustering.
- We will drop the variables `'country'` variable because it is unique for each country and would not add value to clustering.
- We will also drop the `'gdpp'` variable for now, because we want to see if we can identify clusters of countries without relying on GDP and see later if these clusters correspond to an average GDP value for the countries in each cluster.

In [13]:
```
data_new = data.drop(columns = ["country", "gdpp"])
```

In [14]:
```
# Scaling the data and storing the output as a new DataFrame

scaler = StandardScaler()

data_scaled = pd.DataFrame(scaler.fit_transform(data_new), columns = data_ne

data_scaled.head()
```

Out[14]:

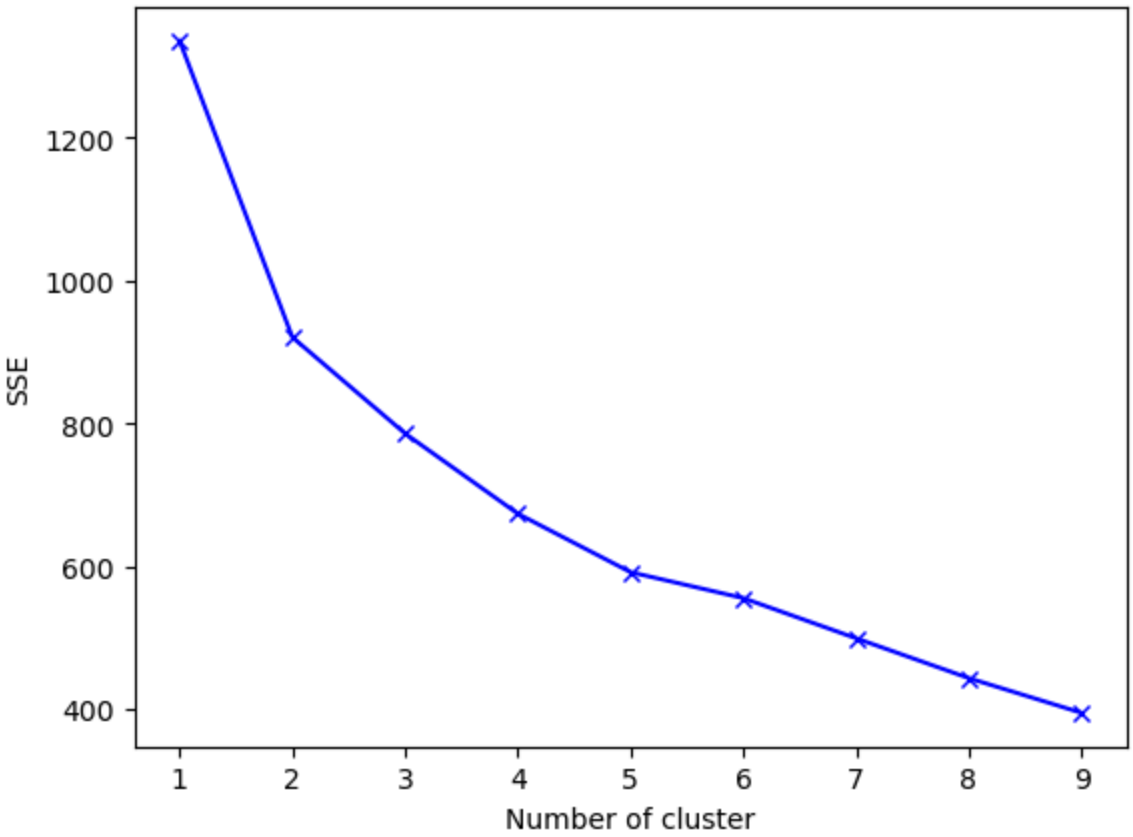
	child_mort	exports	health	imports	income	inflation	life_expec	t
0	1.291532	-1.138280	0.279088	-0.082455	-0.808245	0.157336	-1.619092	1.
1	-0.538949	-0.479658	-0.097016	0.070837	-0.375369	-0.312347	0.647866	-0.
2	-0.272833	-0.099122	-0.966073	-0.641762	-0.220844	0.789274	0.670423	-0.
3	2.007808	0.775381	-1.448071	-0.165315	-0.585043	1.387054	-1.179234	2
4	-0.695634	0.160668	-0.286894	0.497568	0.101732	-0.601749	0.704258	-0.

In [15]:
```
# Creating copy of the data to store labels from each algorithm
data_scaled_copy = data_scaled.copy(deep = True)
```

K-Means Clustering

In [16]:
```
# Empty dictionary to store the SSE for each value of K
sse = {}

# Iterate for a range of Ks and fit the scaled data to the algorithm.
# Use inertia attribute from the clustering object and store the inertia val
for k in range(1, 10):
    kmeans = KMeans(n_clusters = k, random_state = 1).fit(data_scaled)

    sse[k] = kmeans.inertia_

# Elbow plot
plt.figure()

plt.plot(list(sse.keys()), list(sse.values()), 'bx-')
```

```
plt.xlabel("Number of cluster")

plt.ylabel("SSE")

plt.show()
```

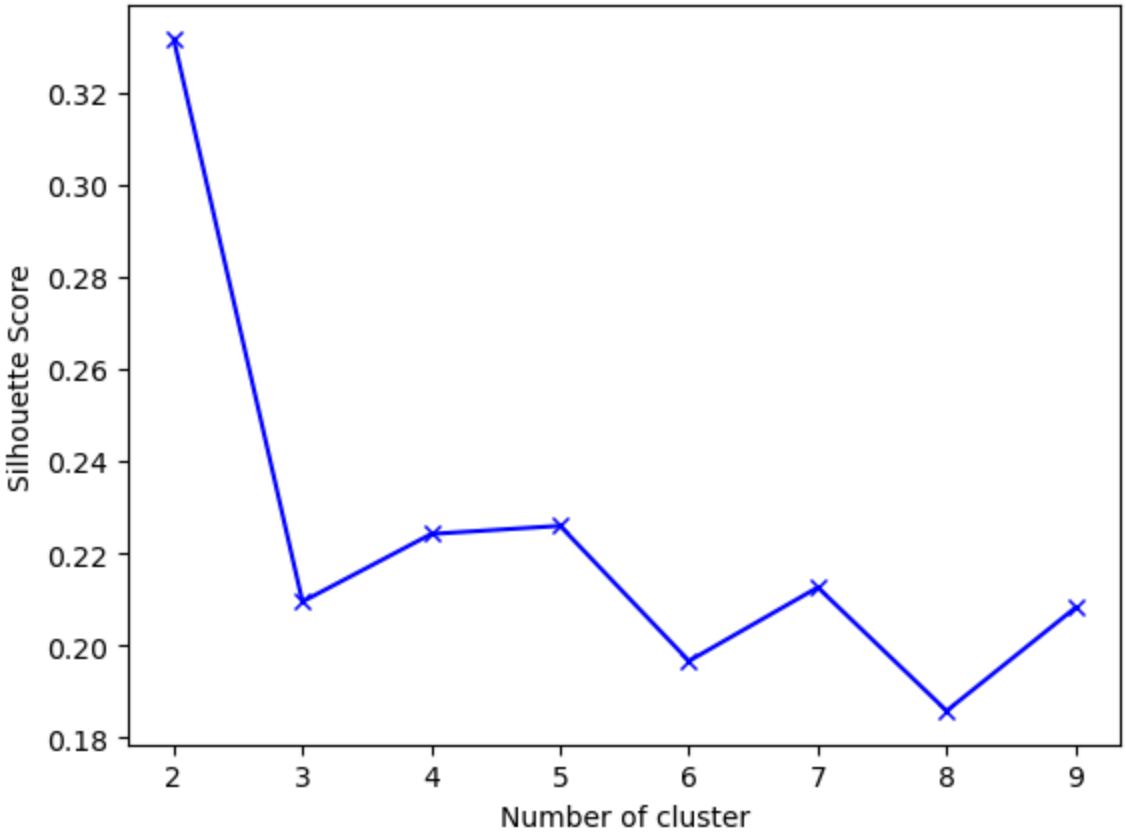


Observations:

- We can see from the plot that there is a consistent dip from 2 to 8 and there doesn't seem to be a clear 'elbow' here. We may choose any number of clusters from 2 to 8.
- So, let's look at another method to get a 'second opinion'. Let's create a plot with Silhouette scores to see how it varies with K.

In [17]:
```
# Empty dictionary to store the Silhouette score for each value of K
sc = {}

# Iterate for a range of Ks and fit the scaled data to the algorithm. Store
for k in range(2, 10):
    kmeans = KMeans(n_clusters = k, random_state = 1).fit(data_scaled)

    labels = kmeans.predict(data_scaled)

    sc[k] = silhouette_score(data_scaled, labels)

# Elbow plot
plt.figure()
```

```
plt.plot(list(sc.keys()), list(sc.values()), 'bx-')

plt.xlabel("Number of cluster")

plt.ylabel("Silhouette Score")

plt.show()
```



Observation:

- We observe from the plot that the silhouette score is the highest for K=2. Let's first understand these 2 clusters.

In [18]:
```
kmeans = KMeans(n_clusters = 2, random_state = 1)

kmeans.fit(data_scaled)

# Adding predicted labels to the original data and the scaled data
data_scaled_copy['KMeans_Labels'] = kmeans.predict(data_scaled)

data['KMeans_Labels'] = kmeans.predict(data_scaled)
```

In [19]:
```
data['KMeans_Labels'].value_counts()
```

Out[19]:
```
KMeans_Labels
0    115
1     52
Name: count, dtype: int64
```

In [ ]:

Observation:

- This looks like a very skewed clustering, with only three observations in one cluster and more than a hundred in another. Let's check out the profiles of these clusters.

In [20]:
```
# Calculating the mean and the median of the original data for each label
num_cols = num_cols + ["KMeans_Labels"]

mean = data[num_cols].groupby('KMeans_Labels').mean()

median = data[num_cols].groupby('KMeans_Labels').median()

df_kmeans = pd.concat([mean, median], axis = 0)

df_kmeans.index = ['group_0 Mean', 'group_1 Mean', 'group_0 Median', 'group_

df_kmeans.T
```

Out[20]:

	group_0 Mean	group_1 Mean	group_0 Median	group_1 Median
child_mort	15.401739	88.844231	11.70	85.65
exports	46.944348	28.203827	39.80	23.30
health	7.062261	6.270385	6.84	5.48
imports	49.026957	42.164729	46.20	39.75
income	23164.000000	3832.750000	16500.00	1960.00
inflation	5.949661	11.833750	3.80	8.95
life_expec	75.377391	59.892308	76.00	60.45
total_fer	2.100522	4.822115	1.95	5.00
gdpp	17997.426087	1832.884615	9070.00	932.00

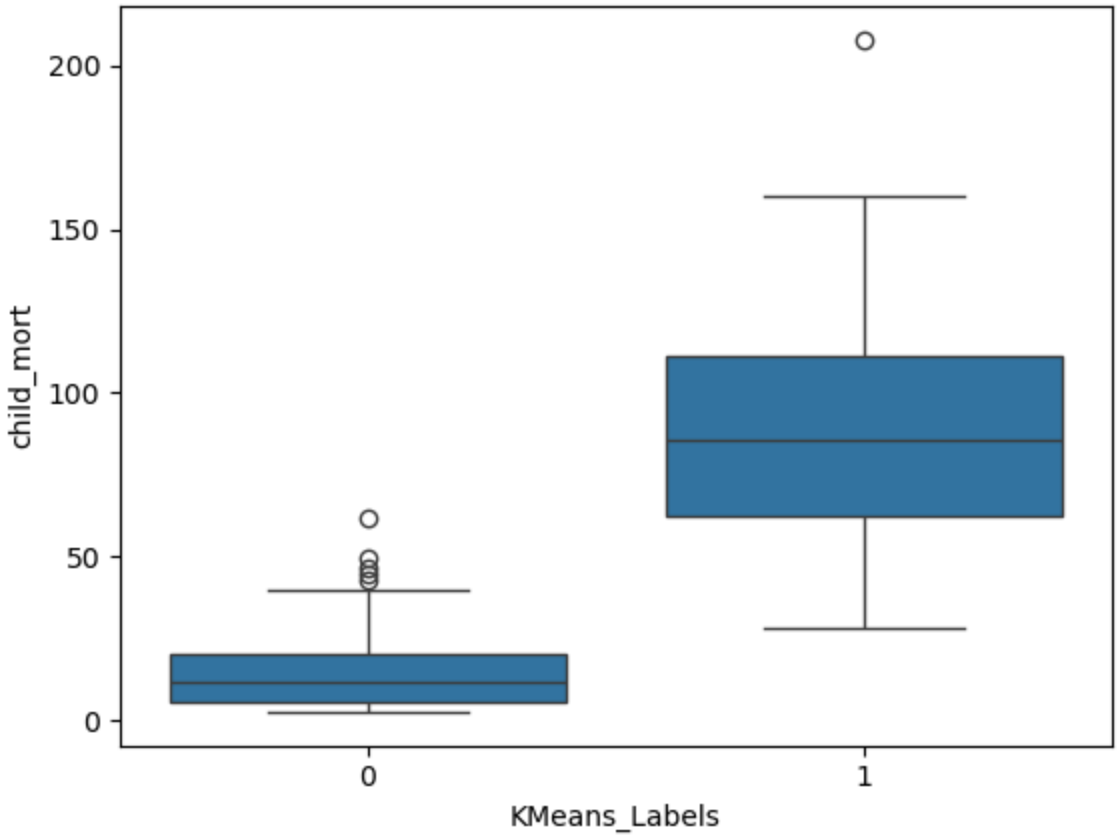
Observations:

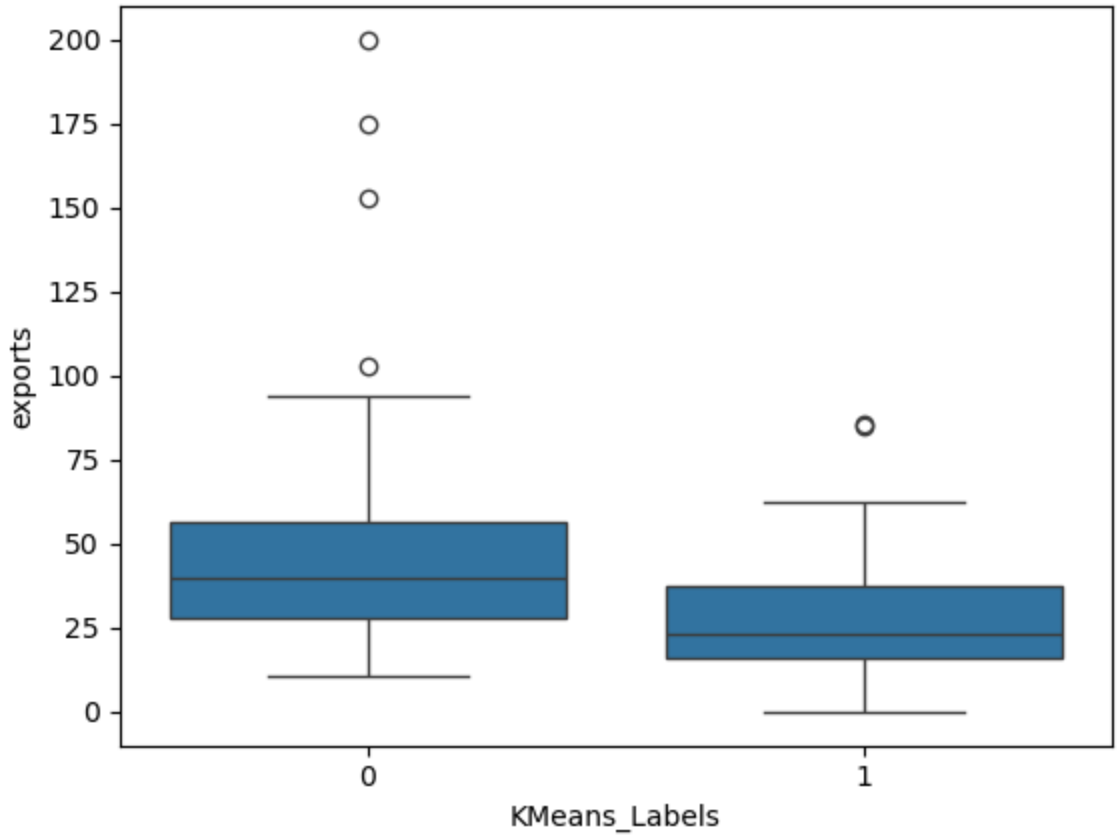
- It looks like Cluster 2 belongs to high income countries which also have high gdpp.
- Cluster 1 seems to be of low income countries, with low mean gdp as well.
- The remaining countries are in Cluster 0 which also happens to be the biggest cluster. Since the number of developing countries is larger than the group of highly developed countries, this intuitively makes sense.

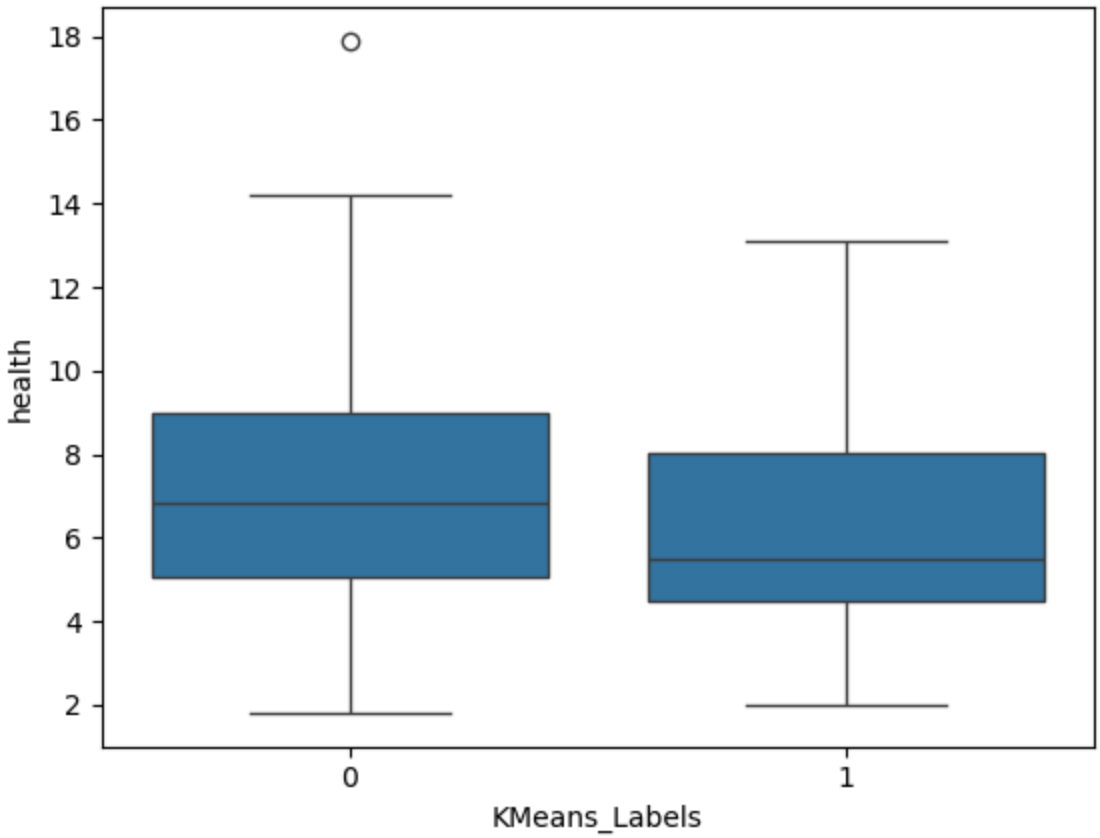
Let us now visualize the summary statistics of these clusters below.

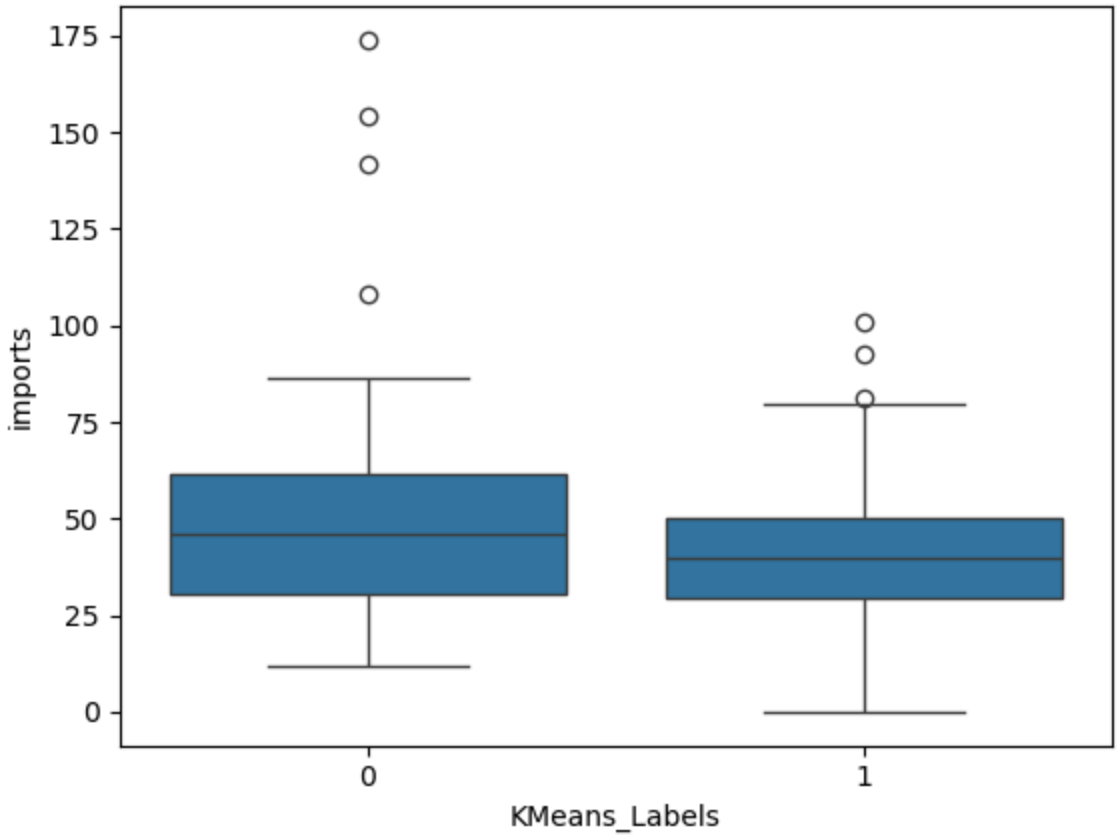
In [21]:
```
cols_visualise = ['child_mort', 'exports', 'health', 'imports', 'income', 'i

for col in cols_visualise:
```

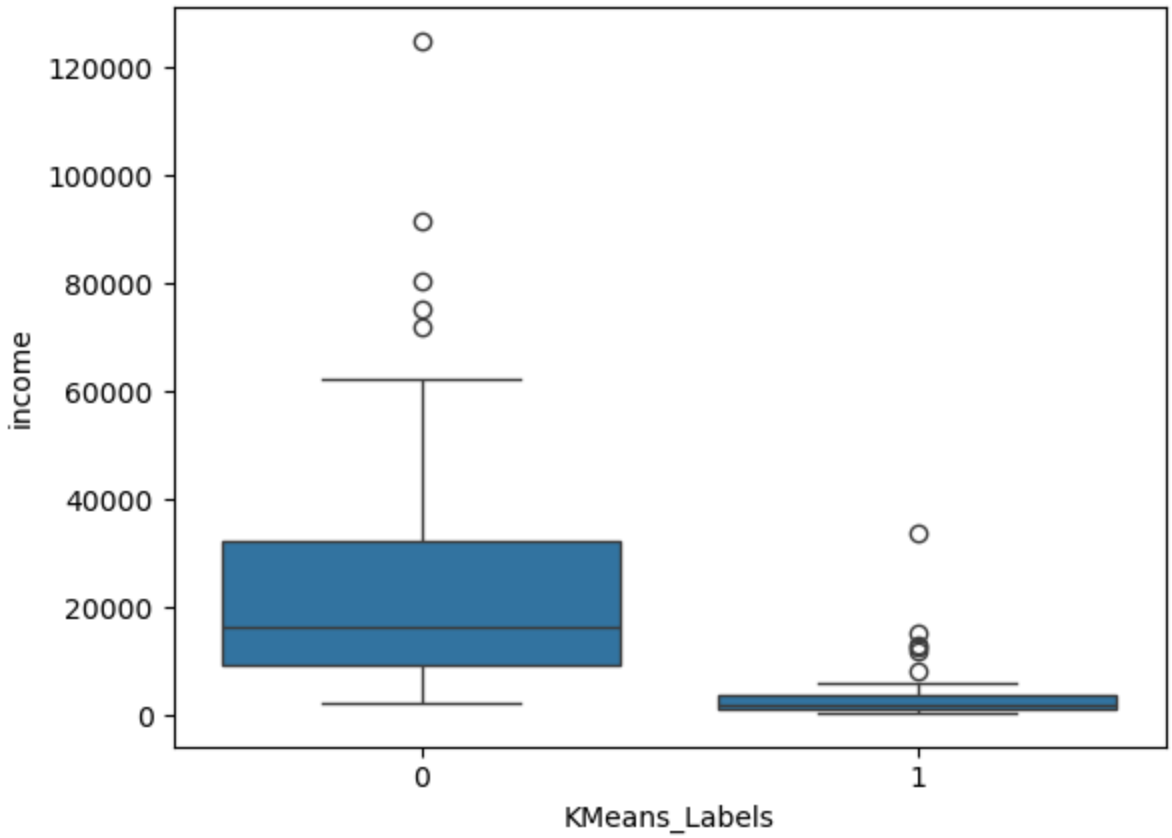
```
    sns.boxplot(x = 'KMeans_Labels', y = col, data = data)
    plt.show()
```

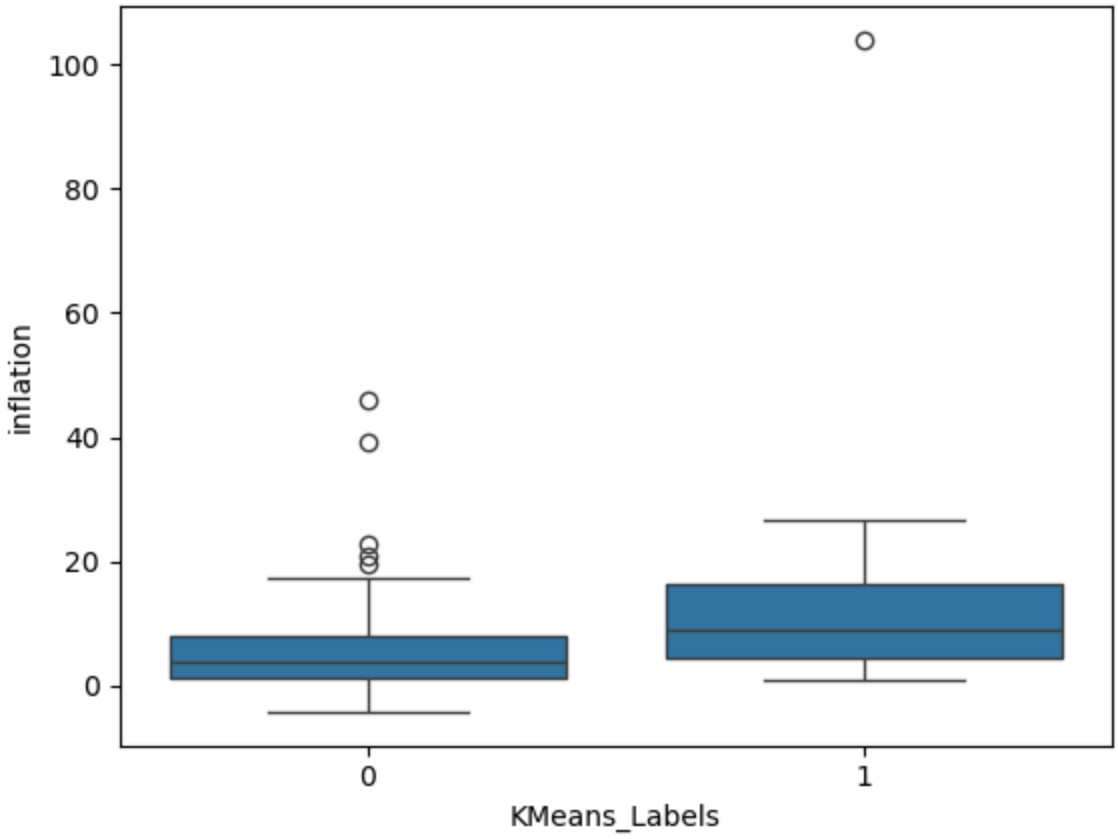


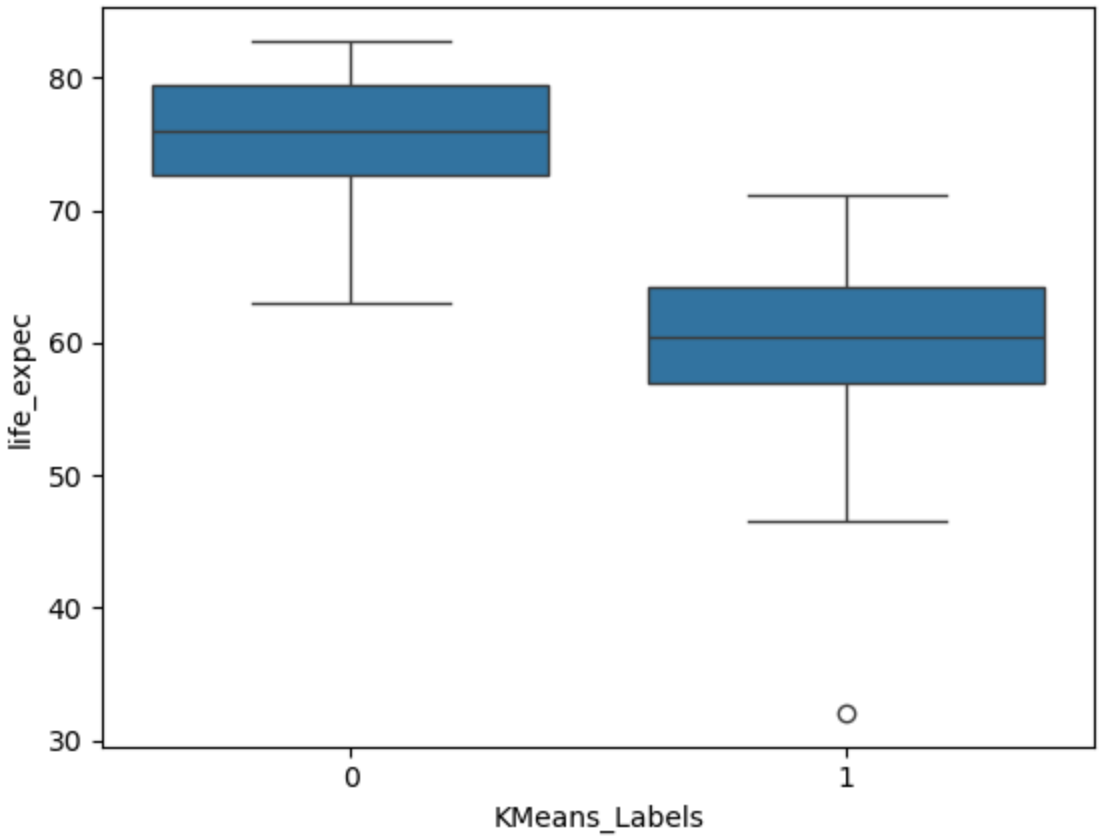


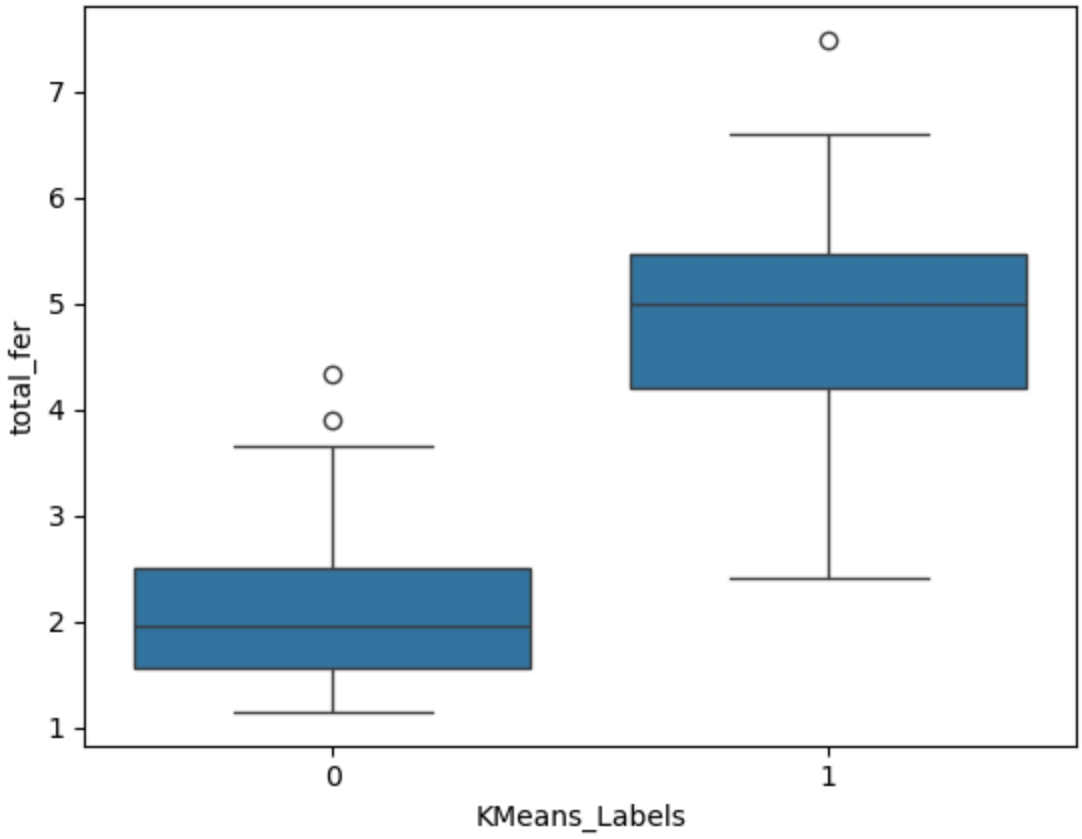

18
16
14
12
10
8
6
4
2
health
0
1
KMeans_Labels

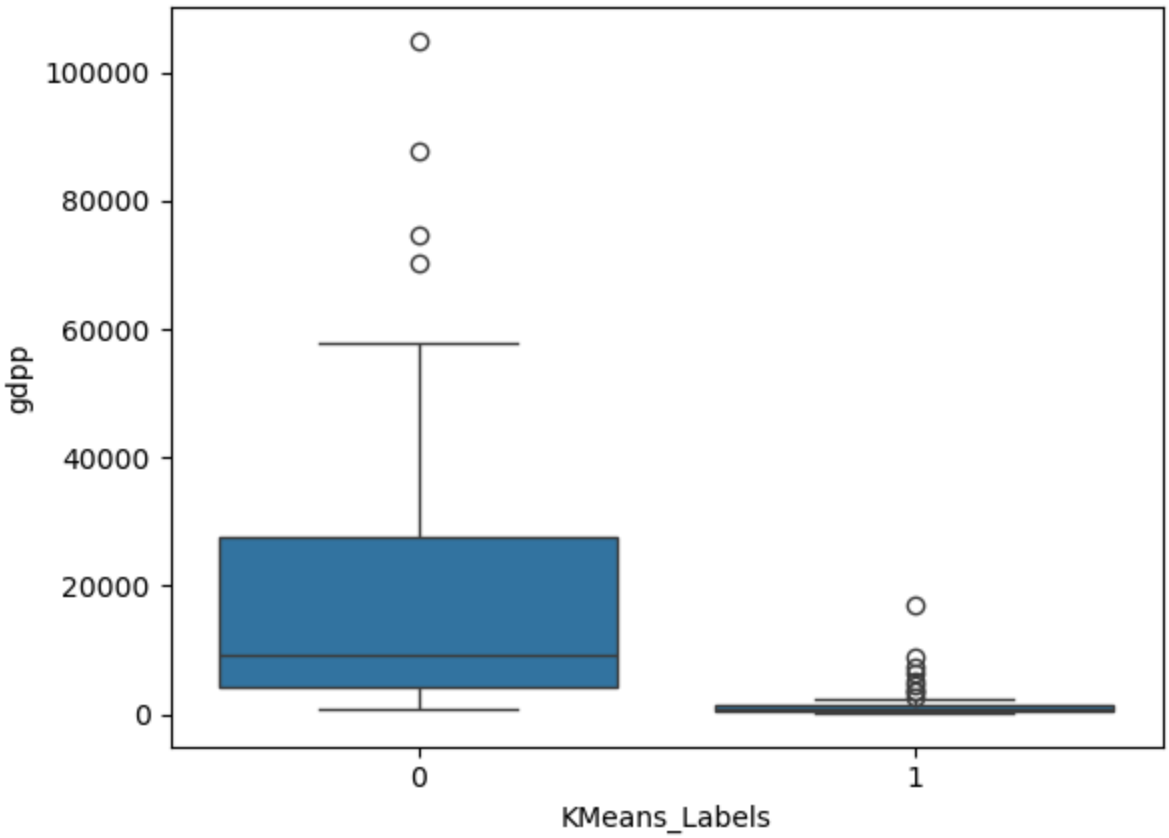

175
150
125
100
75
50
25
0
imports
0
1
KMeans_Labels


120000
100000
80000
60000
40000
20000
0
income
0
1
KMeans_Labels


100
80
60
40
20
0
inflation
0
1
KMeans_Labels


life_expec
80
70
60
50
40
30
0
1
KMeans_Labels


total_fer
7
6
5
4
3
2
1
0
1
KMeans_Labels

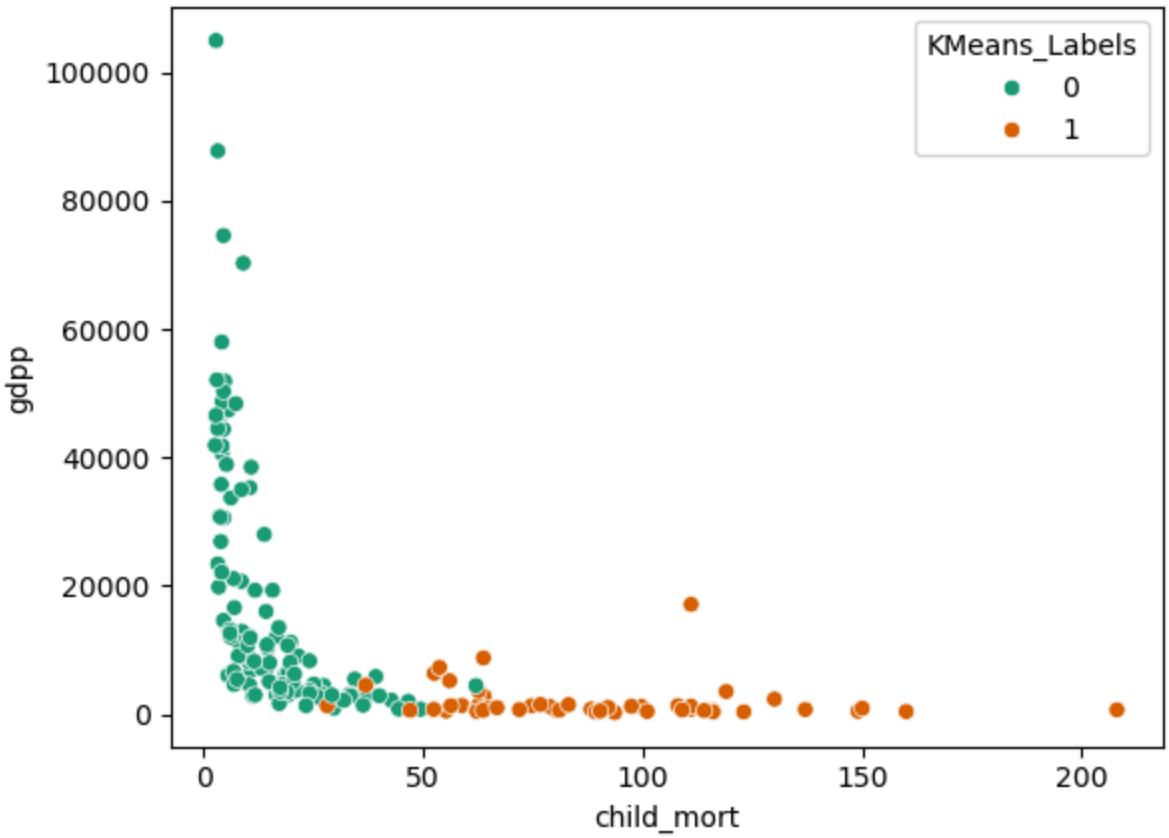

100000
80000
60000
40000
20000
0
gdpp
0
1
KMeans_Labels

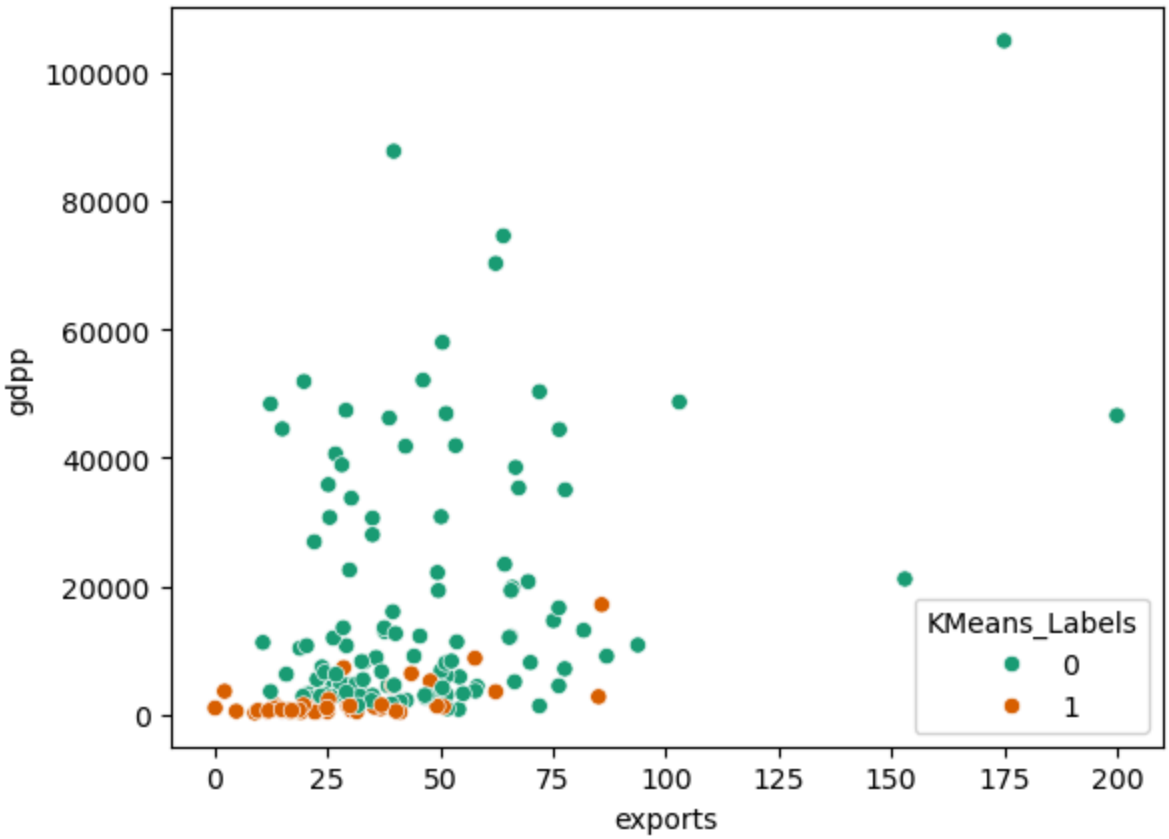
Cluster Profiles:

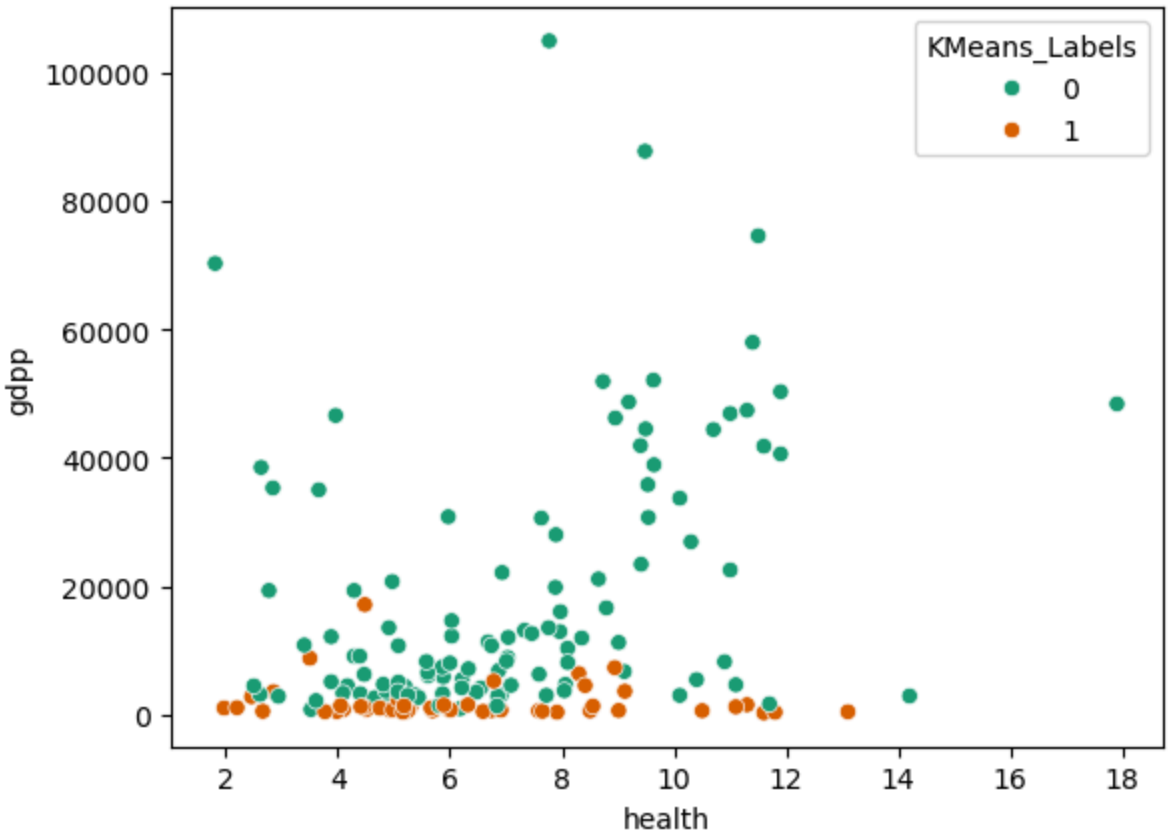
- Cluster 2 has only 3 observations. As observed from the scatter plots and the boxplots, this group **consists of outlier high income countries** with the highest percentages of imports and exports in terms of GDP.
- Cluster 1 seems to have countries with less desirable values for many indicators. These countries seem to have the highest inflation rates, the lowest GDP per capita, the lowest exports as well as imports - all signaling a very poor economic situation. These countries also have the highest child mortalities, the highest fertility rates, and the lowest life expectancies. These characteristics are traits of **underdeveloped or developing countries**. These countries also seem to have a trade deficit, i.e., more imports than exports, and as a consequence, may be more reliant on borrowing and lines of credit to finance their economy.
- Cluster 0 is the largest cluster with traits of countries that fall in the **middle of the development spectrum**. These countries have a comparatively better state of affairs than the countries in cluster 1. However, this cluster has a large range of values, indicating that it is a mix of many different types of countries. Ideally, we do not want a cluster to be like this as the fundamental idea behind clustering is to 'group similar things' and this cluster seems to have a lot of 'dissimilarity' within it.
- Overall, this clustering solution does give us good insights into potential clusters of similar countries but is not very useful as it is impacted by outlier countries resulting in one very small cluster and two very big clusters. We should try other algorithms to see if we can do better.

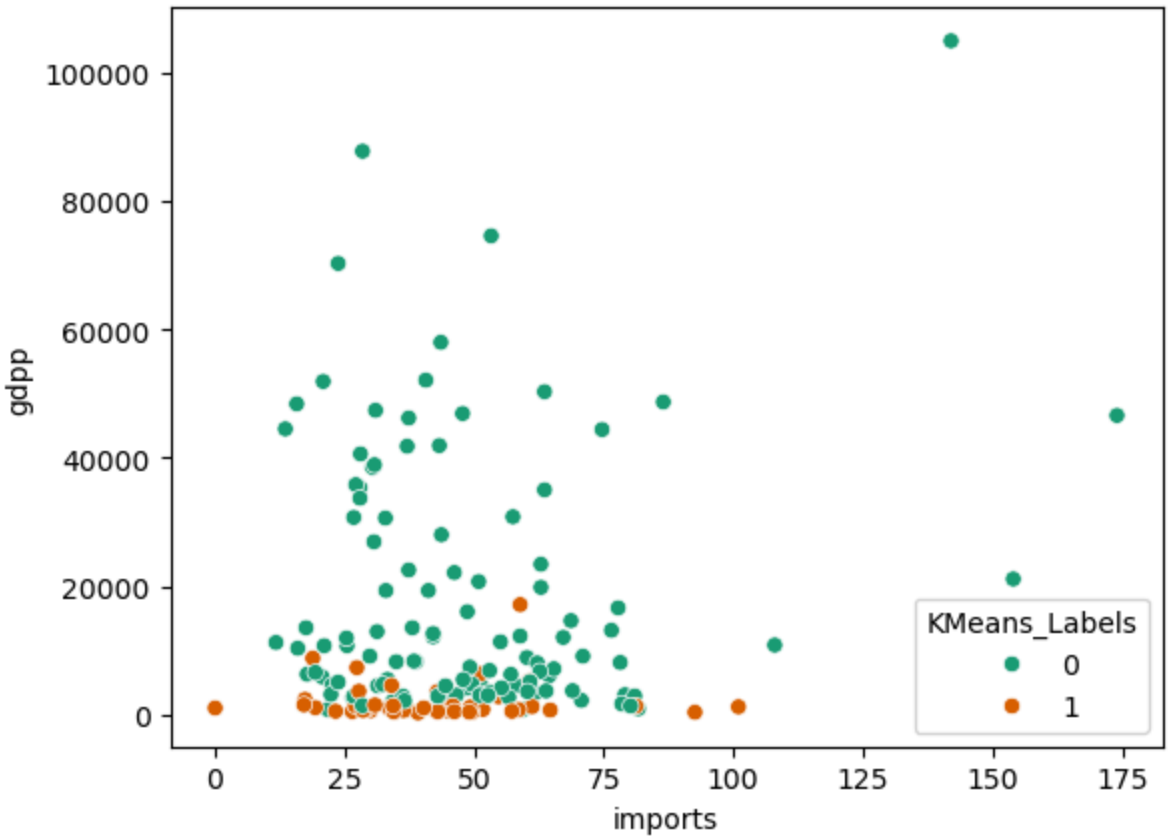
But before that, let's validate if these clusters relate well with the GDP of the country.

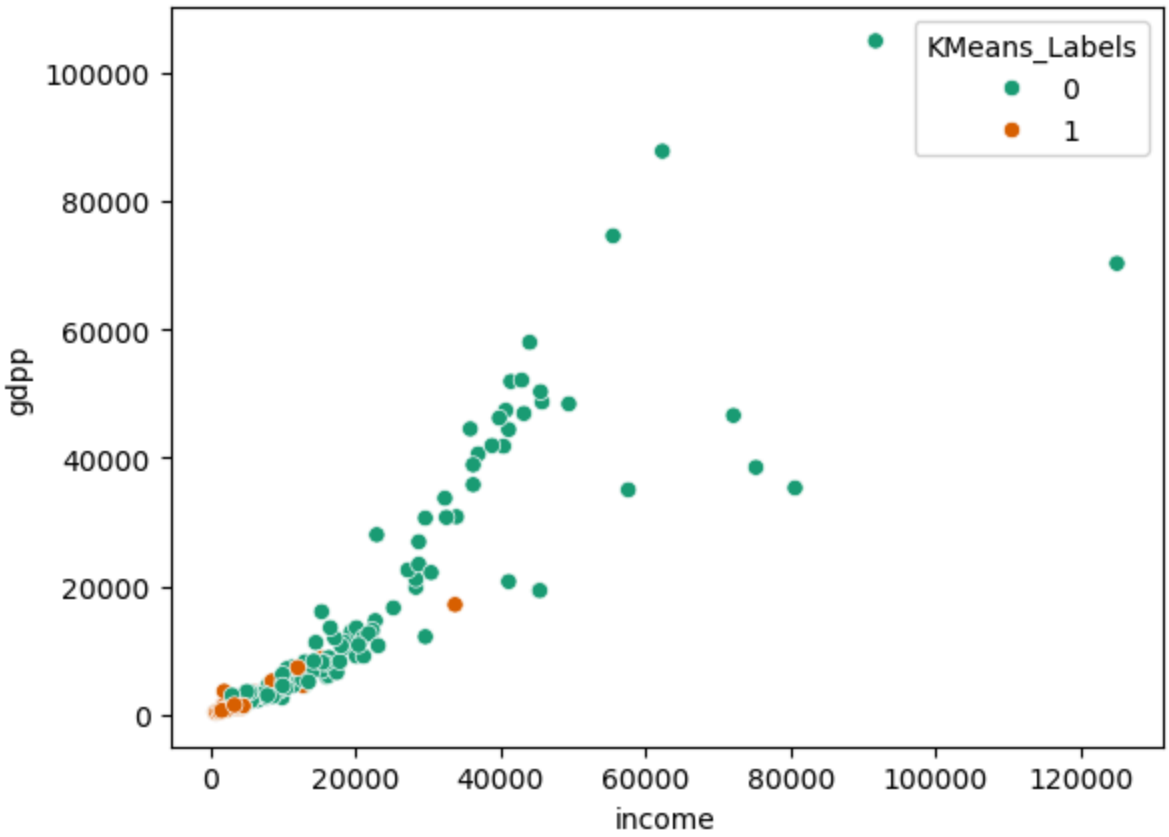
In [22]:
```
cols_visualise = ['child_mort', 'exports', 'health', 'imports', 'income', 'i

for col in cols_visualise:
    sns.scatterplot(x = col, y = 'gdpp', data = data, hue = 'KMeans_Labels',

    plt.show()
```

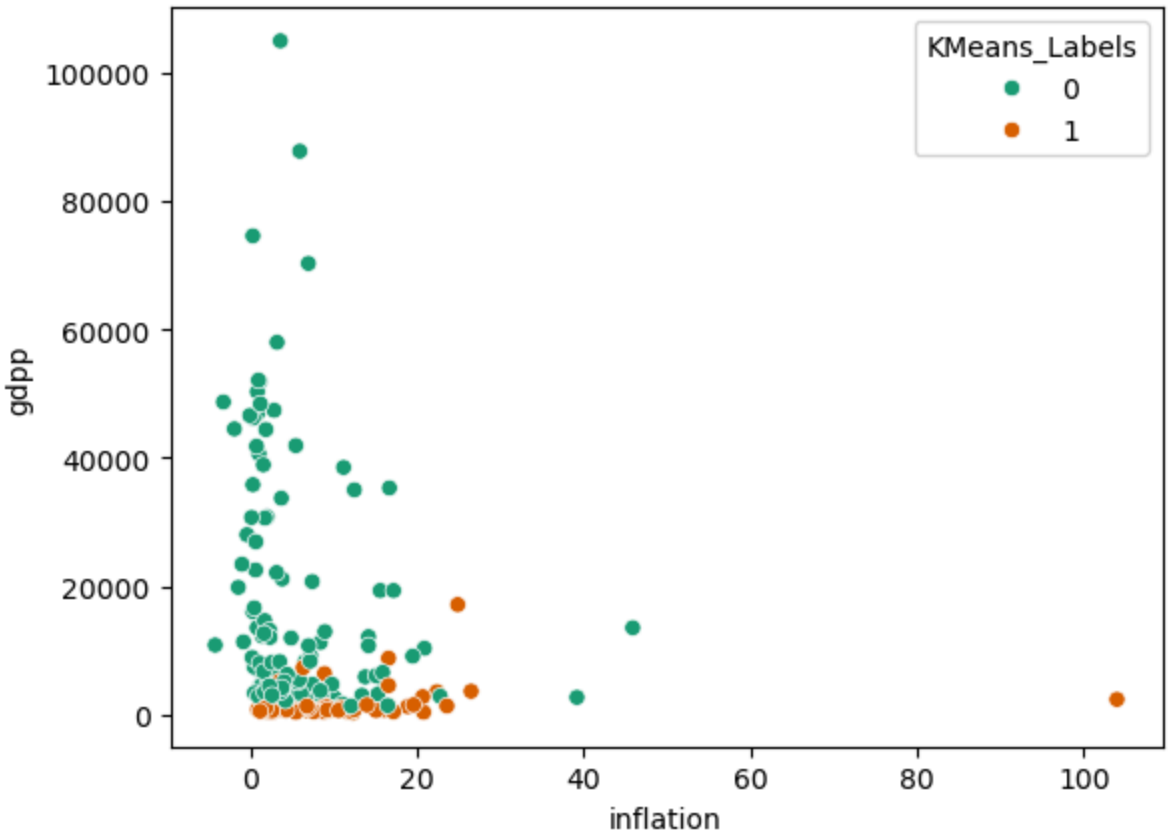

KMeans_Labels
0
1
gdpp
child_mort
0
20000
40000
60000
80000
100000
0
50
100
150
200

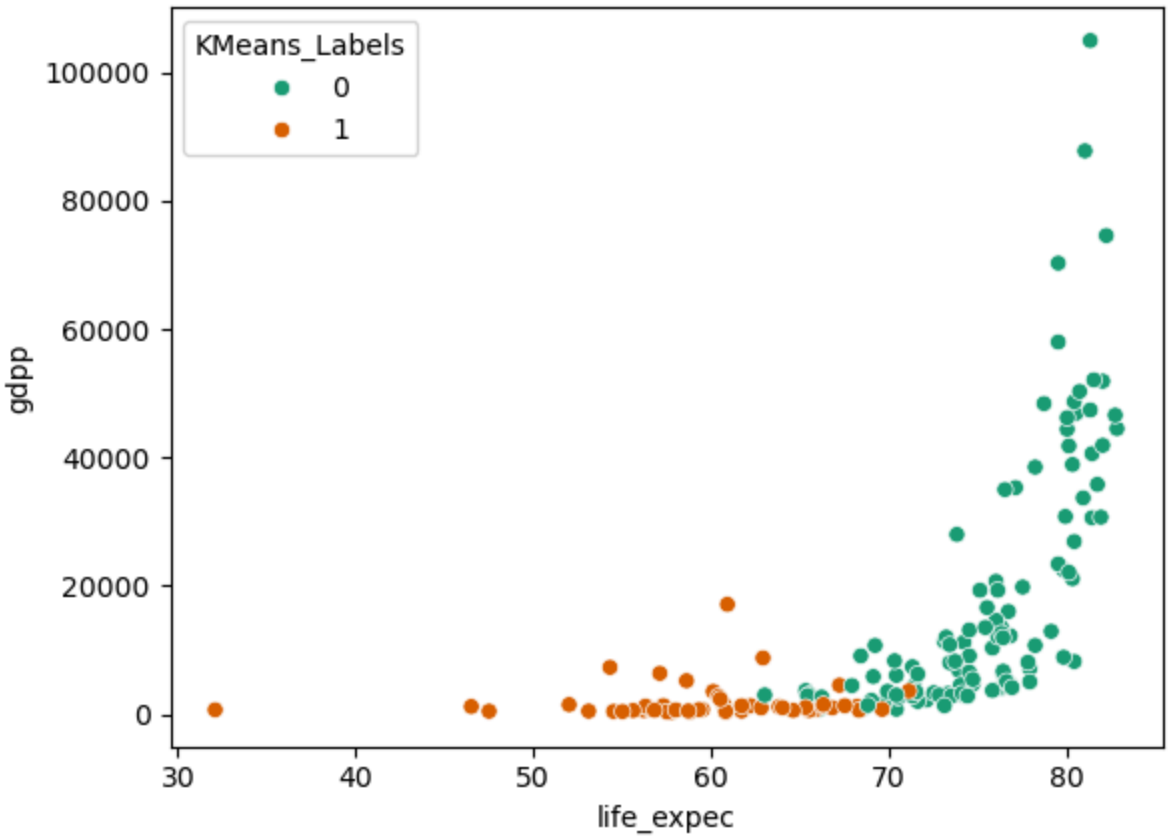

KMeans_Labels
0
1
gdpp
exports
0
20000
40000
60000
80000
100000
0
25
50
75
100
125
150
175
200

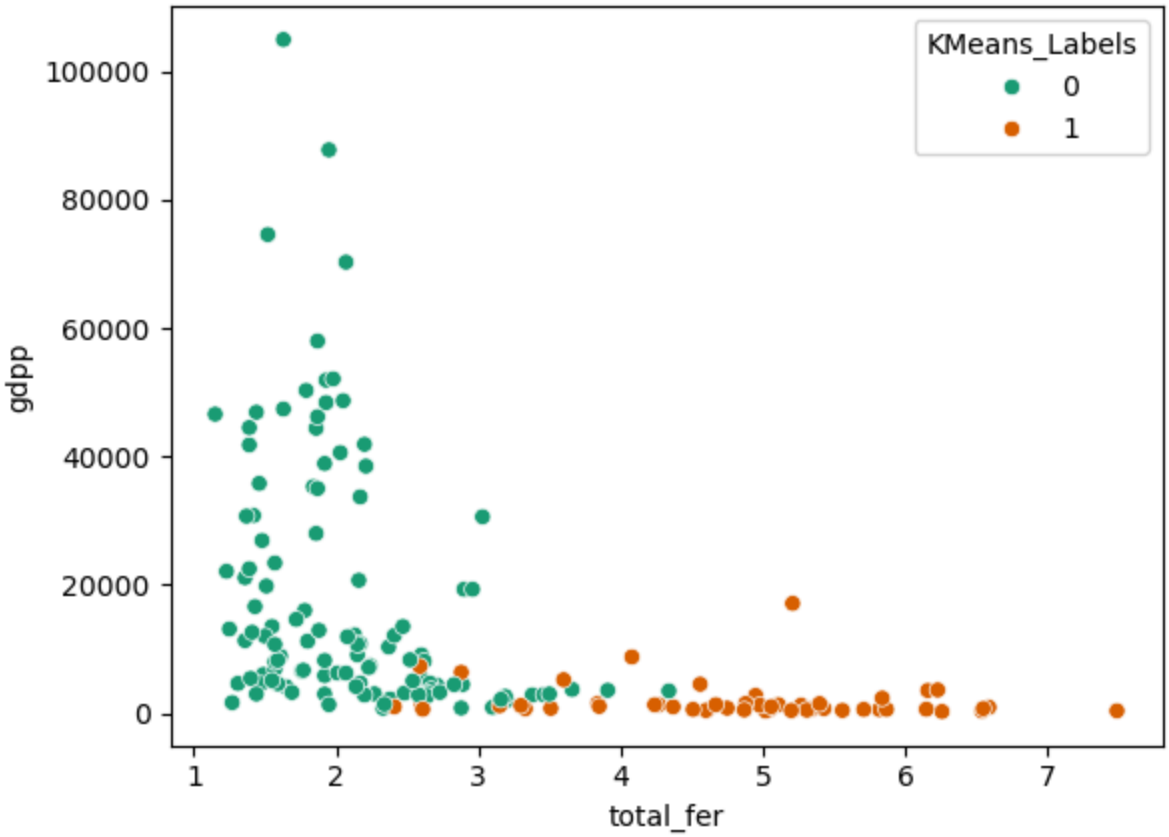

KMeans_Labels
0
1
gdpp
health
100000
80000
60000
40000
20000
0
2
4
6
8
10
12
14
16
18

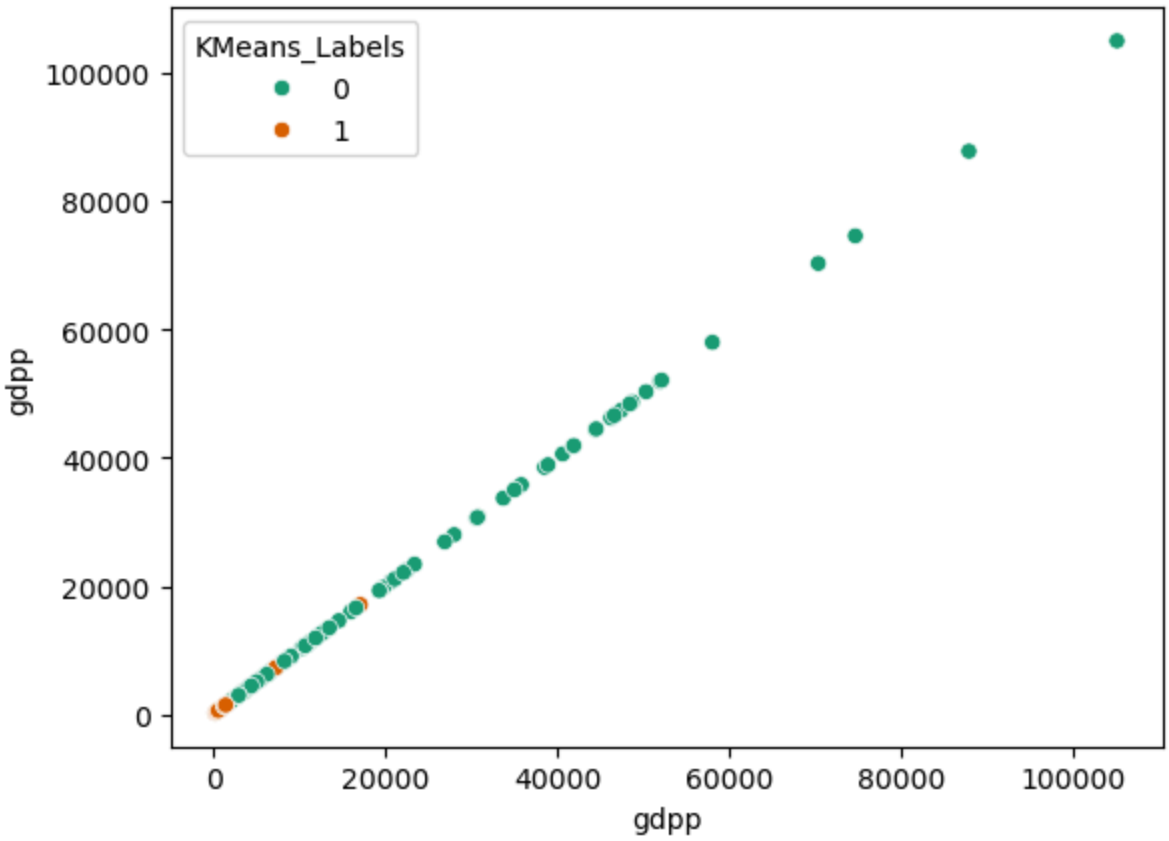

KMeans_Labels
0
1
gdpp
imports
100000
80000
60000
40000
20000
0
0
25
50
75
100
125
150
175


KMeans_Labels
0
1
gdpp
income
0
20000
40000
60000
80000
100000
120000


KMeans_Labels
0
1
gdpp
inflation
0
20000
40000
60000
80000
100000
0
20
40
60
80
100


KMeans_Labels
0
1
gdpp
life_expec
100000
80000
60000
40000
20000
0
30
40
50
60
70
80


KMeans_Labels
0
1
gdpp
total_fer
100000
80000
60000
40000
20000
0
1
2
3
4
5
6
7



Observations:

- The countries with higher fertility rates also seem to have higher populations, corresponding with lower per capita income in these countries.
- The child mortality also seems to be negatively correlated with the GDP of the country. The high child mortality in such countries could be due to several reasons such as high poverty or lower net income per person and a relative lack of health facilities among others.

Let's try another algorithm

K-Medoids Clustering

In [23]:
```
kmedo = KMedoids(n_clusters = 2, random_state = 1)

kmedo.fit(data_scaled)

data_scaled_copy['kmedoLabels'] = kmedo.predict(data_scaled)

data['kmedoLabels'] = kmedo.predict(data_scaled)
```

In [24]:
```
data.kmedoLabels.value_counts()
```

Out[24]:
```
kmedoLabels
1    98
0    69
Name: count, dtype: int64
```

In [25]:
```
data
```

Out[25]:

	country	child_mort	exports	health	imports	income	inflation	life_expec
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8
...	...	...	...	...	...	...	...	...
162	Vanuatu	29.2	46.6	5.25	52.7	2950	2.62	63.0
163	Venezuela	17.1	28.5	4.91	17.6	16500	45.90	75.4
164	Vietnam	23.3	72.0	6.84	80.2	4490	12.10	73.1
165	Yemen	56.3	30.0	5.18	34.4	4480	23.60	67.5
166	Zambia	83.1	37.0	5.89	30.9	3280	14.00	52.0

167 rows × 12 columns

In [26]:
```
# Calculating the mean and the median of the original data for each label
original_features = ['child_mort', 'exports', 'health', 'imports', 'income',
num_cols = original_features + ['kmedoLabels']
mean = data[num_cols].groupby('kmedoLabels').mean()

median = data[num_cols].groupby('kmedoLabels').median()

df_kmedoids = pd.concat([mean, median], axis = 0)

df_kmedoids.index = ['group_0 Mean', 'group_1 Mean', 'group_0 Median', 'grou

df_kmedoids[original_features].T
```

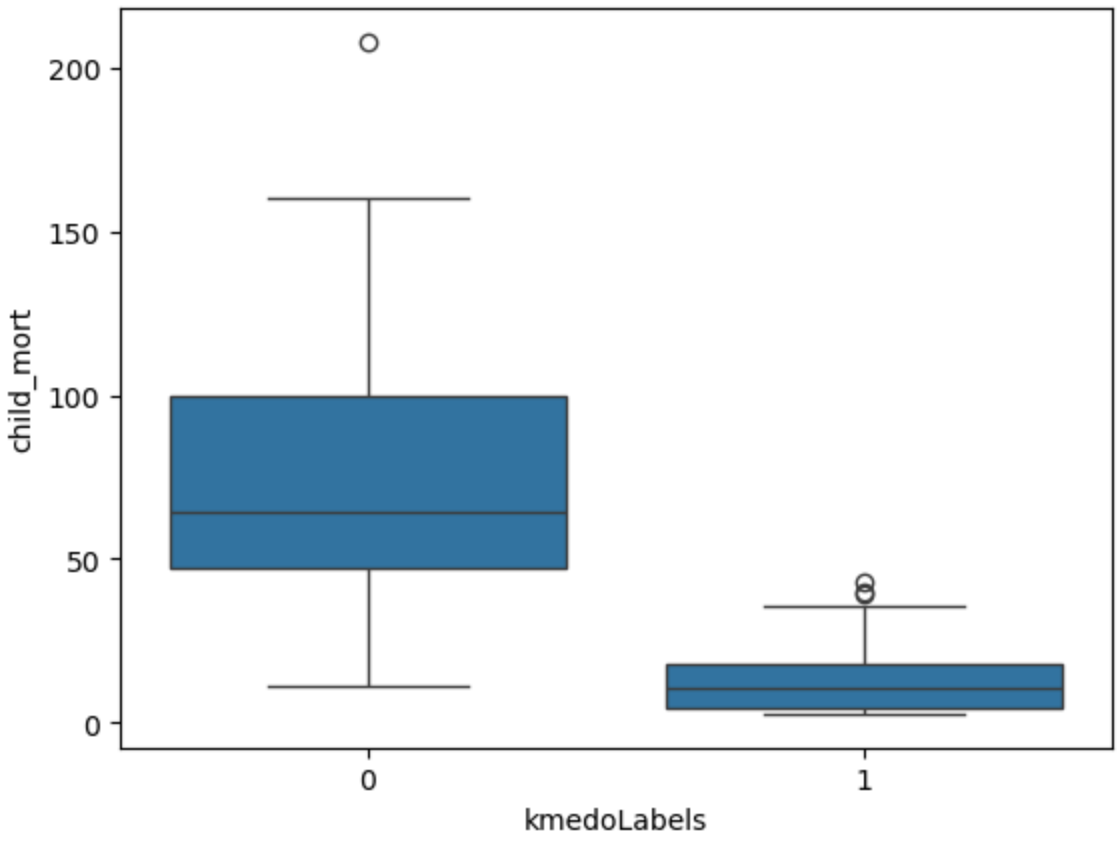
Out[26]:

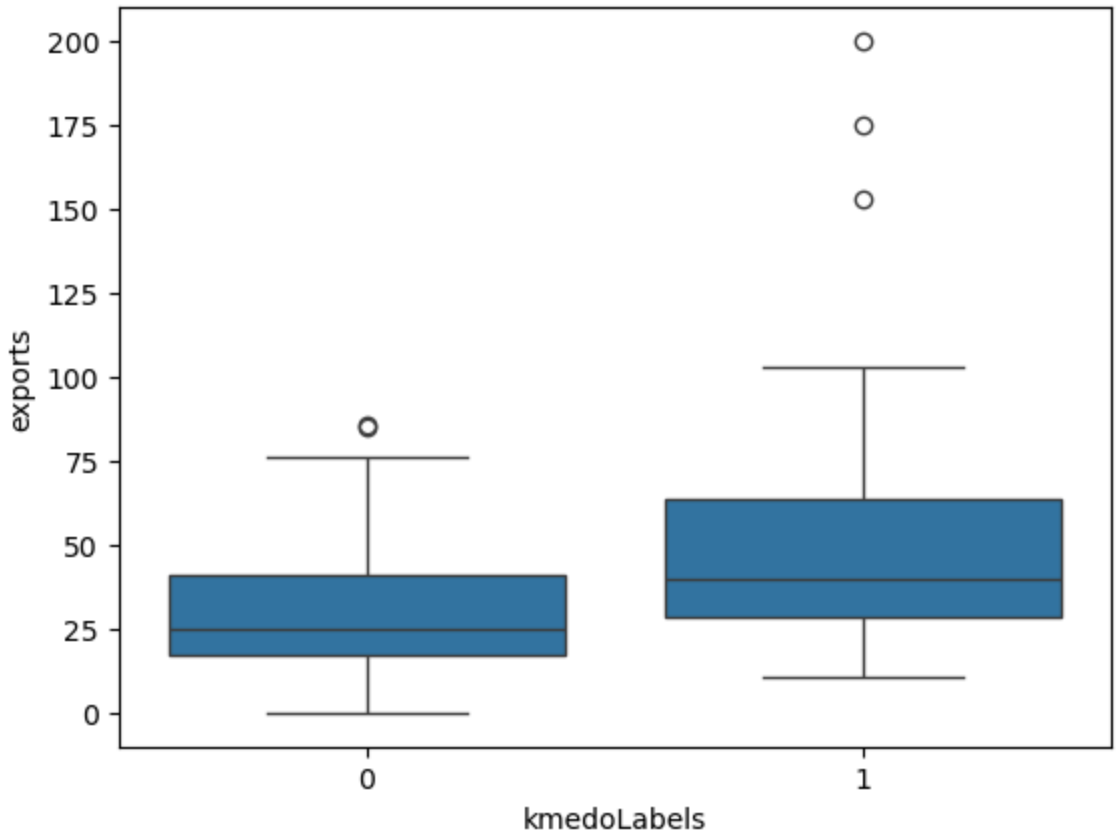
	group_0 Mean	group_1 Mean	group_0 Median	group_1 Median
child_mort	74.804348	12.546939	64.40	10.300
exports	30.384043	48.660204	25.00	39.950
health	5.872754	7.479592	5.25	7.210
imports	42.386462	50.061224	39.20	47.200
income	4674.971014	25924.387755	2660.00	19250.000
inflation	12.086594	4.750929	8.92	3.375
life_expec	62.194203	76.442857	62.20	76.400
total_fer	4.356087	1.956531	4.56	1.875
gdpp	2237.884058	20516.326531	1170.00	11600.000

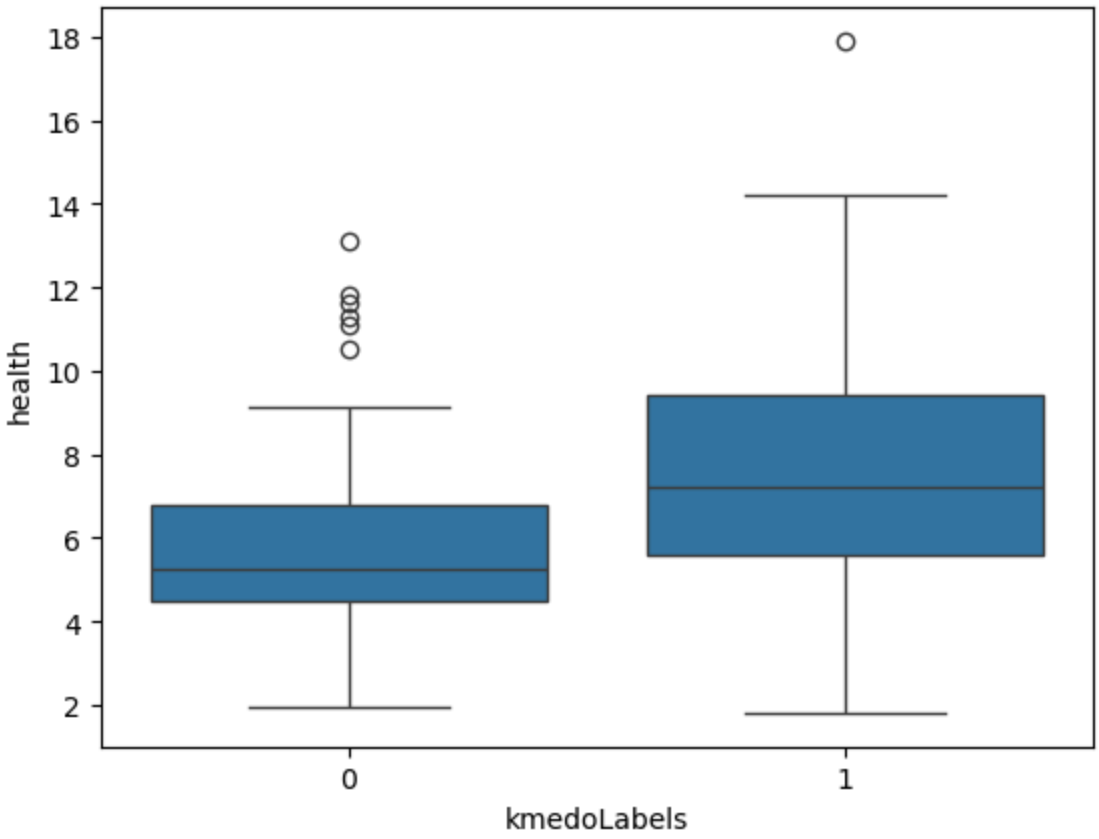
Observations:

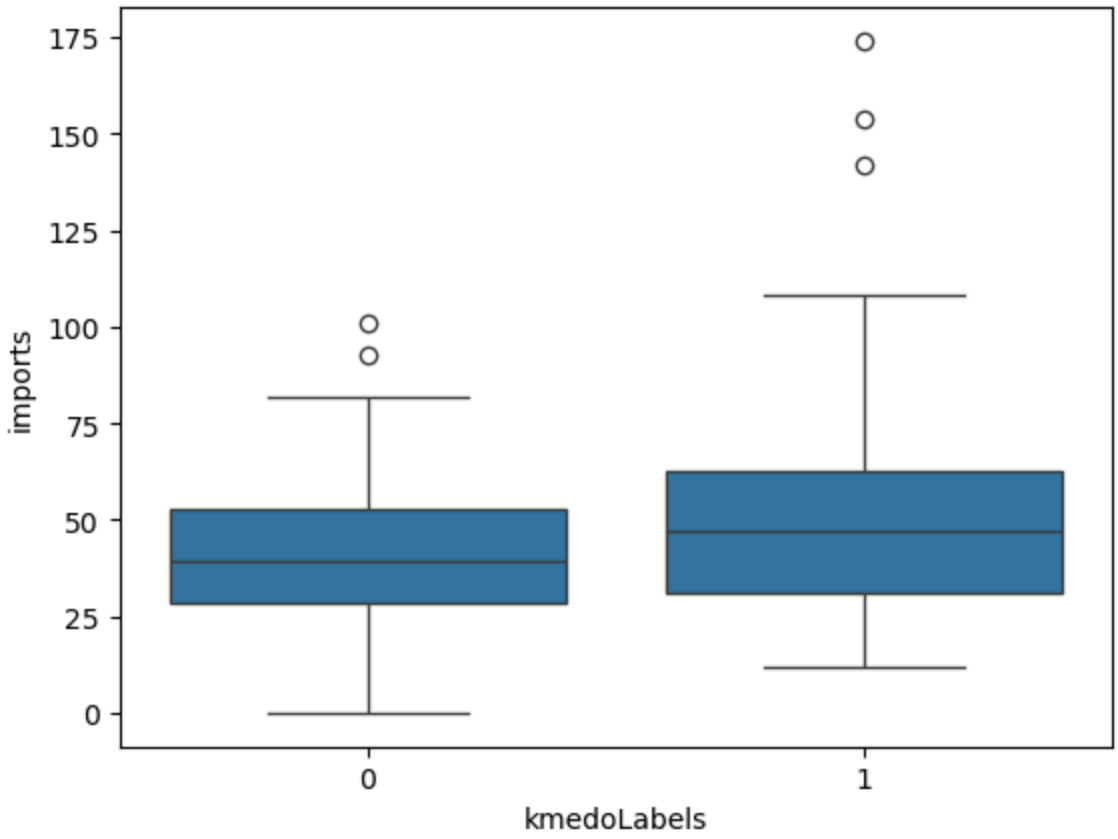
- It looks like Cluster 0 belongs to high income countries, Cluster 2 has poorer countries with low incomes, and the remaining countries are in Cluster 1, which happens to be the biggest cluster as well.

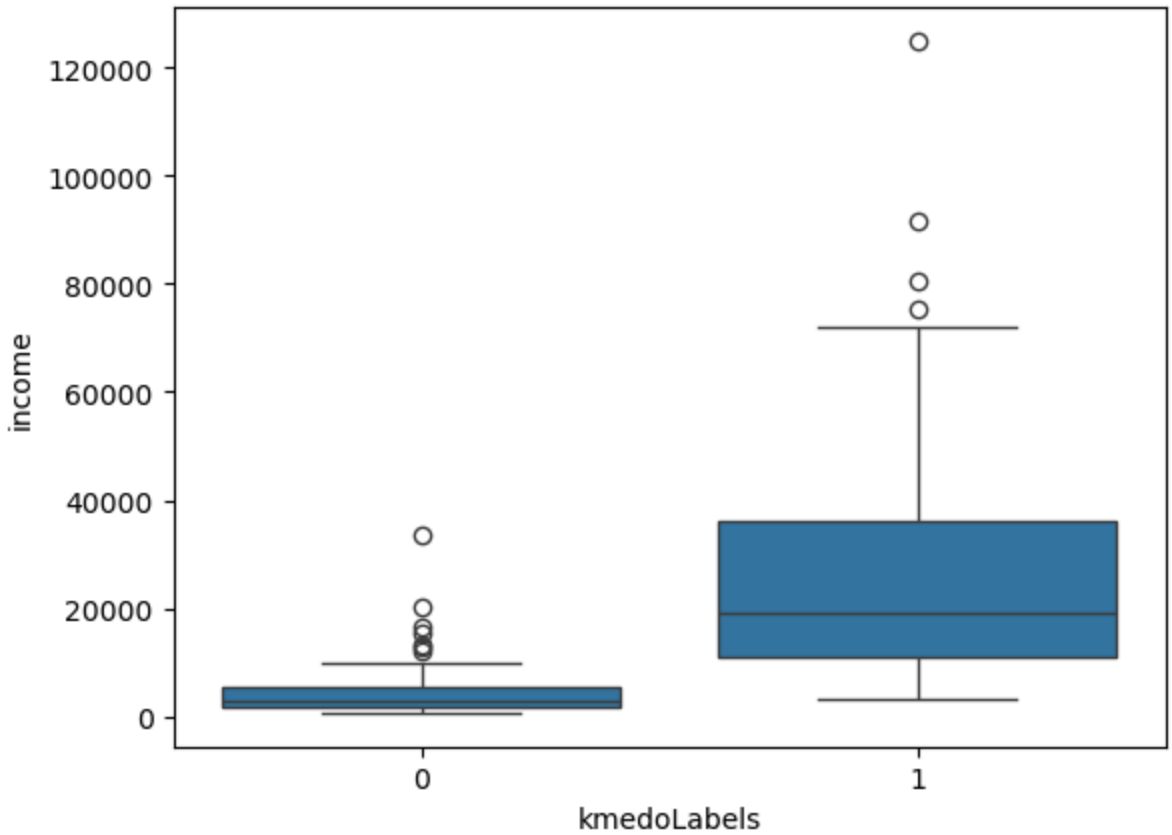
In [27]:
```
for col in cols_visualise:
    sns.boxplot(x = 'kmedoLabels', y = col, data = data)

    plt.show()
```

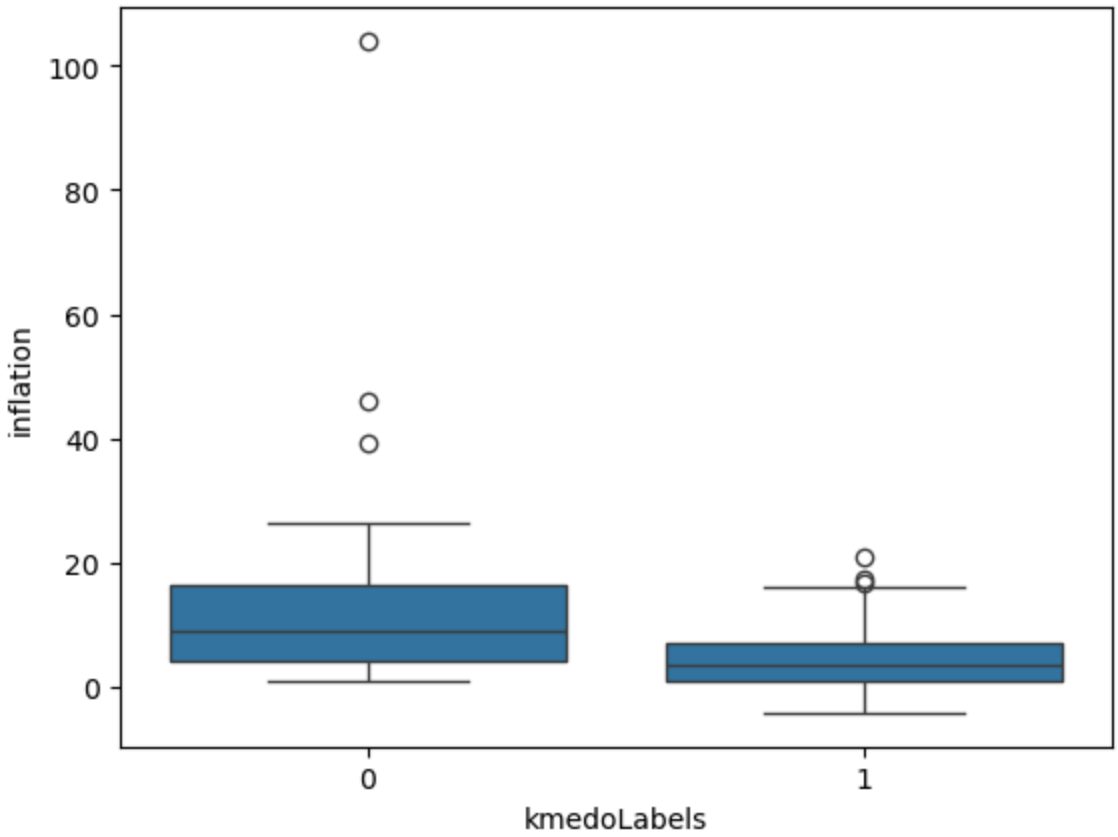


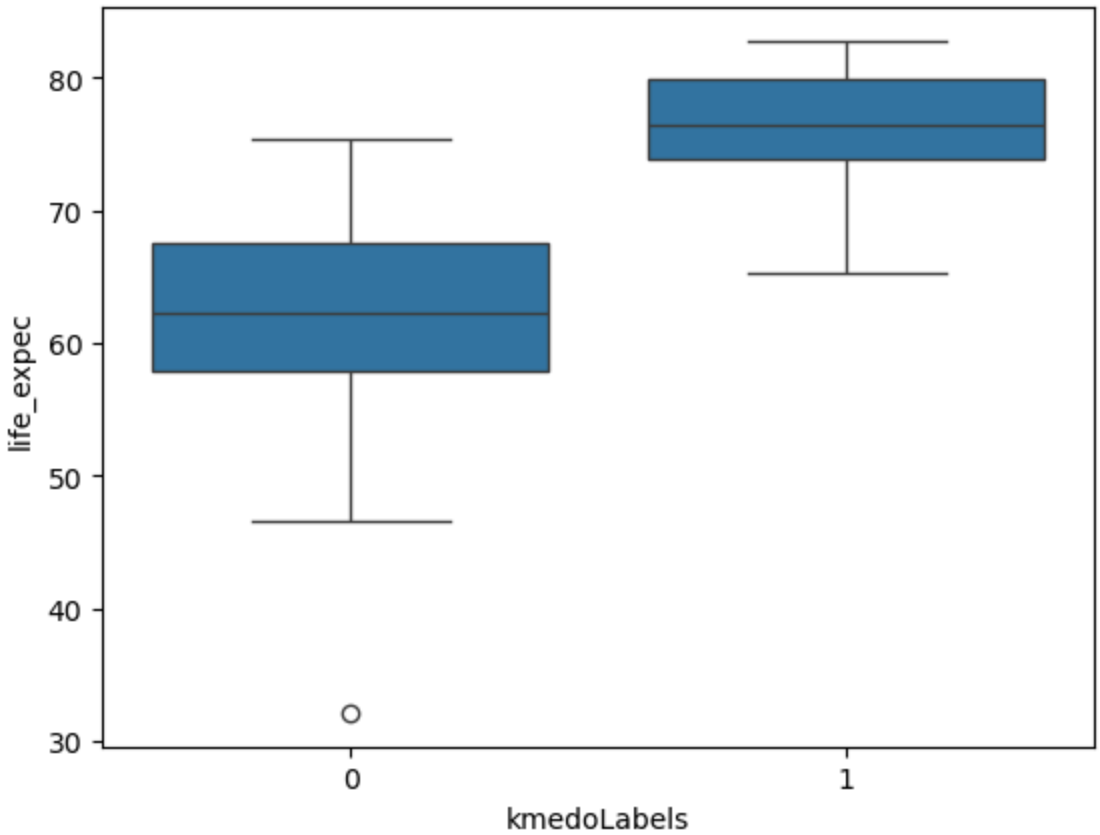

200
175
150
125
100
75
50
25
0
exports
0
1
kmedoLabels

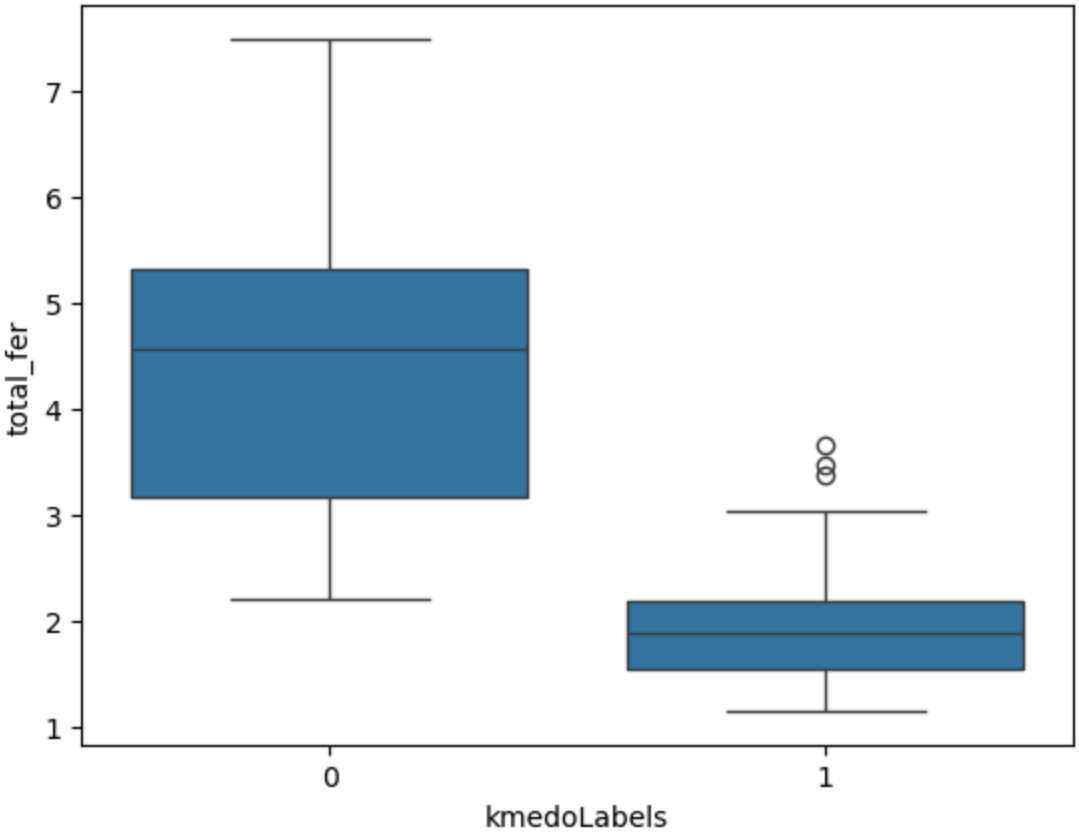

18
16
14
12
10
8
6
4
2
health
0
1
kmedoLabels

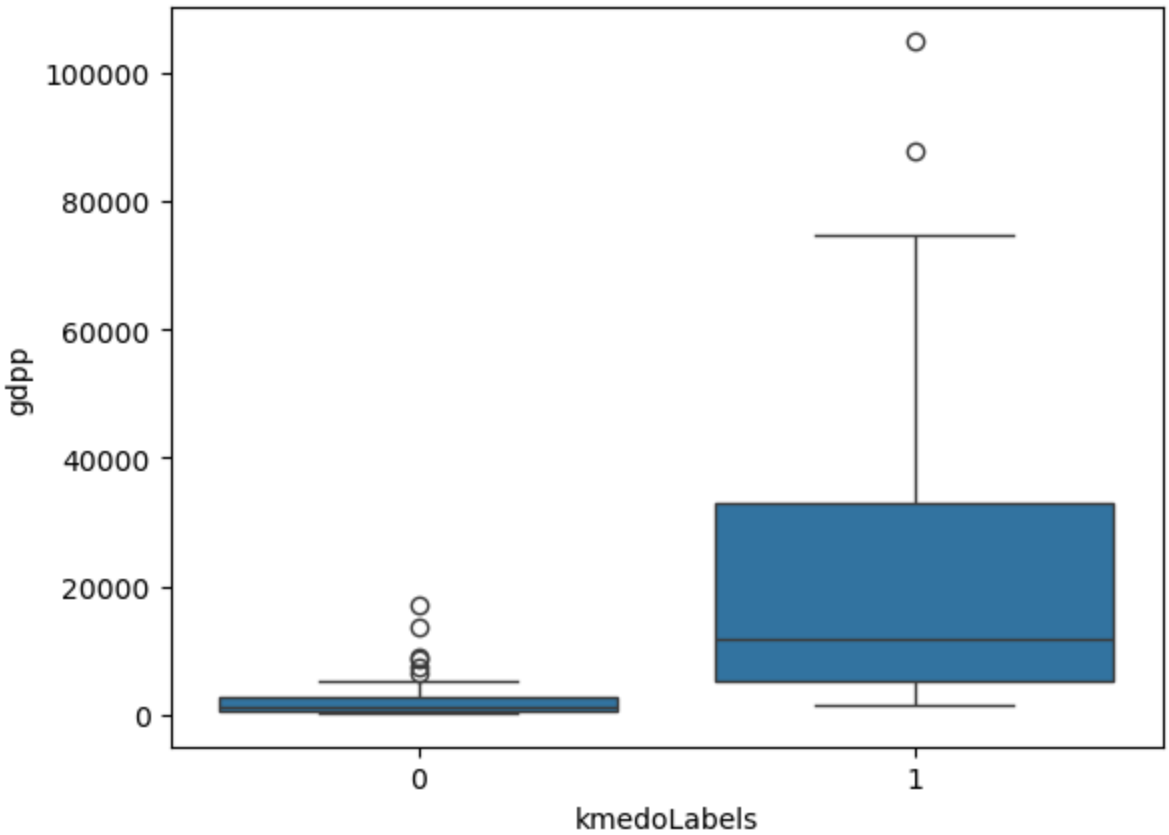

175
150
125
100
75
50
25
0
imports
0
1
kmedoLabels


120000
100000
80000
60000
40000
20000
0
income
0
1
kmedoLabels


inflation
100
80
60
40
20
0
0
1
kmedoLabels


life_expec
80
70
60
50
40
30
0
1
kmedoLabels





Cluster Profiles:

- Cluster 2 countries have the highest average child mortality rate, trade deficit, inflation rate and least average GDP and net income per person. But the large range of values for different variables implies that cluster 2 contains a variety of countries, from **underdeveloped to developing** ones.
- Cluster 1 shows traits of **developing countries** with comparatively higher GDP, net income per person and significantly lower child mortality rate as compared to cluster 2. The cluster consists of some outliers but majorly it consists of countries with low to medium GDP, with a comparatively higher percentage of imports and exports vs GDP.
- Cluster 0 shows traits of **highly developed countries** with a low child mortality rate and a higher net income per person, life expectancy, and GDP. These countries have the highest average expenditure on health as a percentage of GDP.

Observations:

- The number of observations for each cluster from K-Medoids is more evenly distributed in comparison to K-Means clustering.
- This is because the clusters from K_Medoids are less affected by outliers from the data. As we observe, the three outlier countries from K-Means (in terms of imports and exports) are now included in cluster 1 and do not form a separate cluster like in K-Means.
- Unlike in K-Means, the cluster for developed countries is much bigger but still retains the overall characteristics of developed countries, as reflected in the higher values for income per person, life expectancy, and especially in health expenditure as a percentage of GDP.

Now, let's see what we get with Gaussian Mixture Model.

Gaussian Mixture Model

In [28]:
```
gmm = GaussianMixture(n_components = 2, random_state = 1)

gmm.fit(data_scaled)

data_scaled_copy['GmmLabels'] = gmm.predict(data_scaled)

data['GmmLabels'] = gmm.predict(data_scaled)
```

In [29]:
```
data.GmmLabels.value_counts()
```

Out[29]:
```
GmmLabels
1    84
0    83
Name: count, dtype: int64
```

In [30]:
```
# Calculating the mean and the median of the original data for each label
original_features = ['child_mort', 'exports', 'health', 'imports', 'income',
```

```
num_cols = original_features + ['GmmLabels']
mean = data[num_cols].groupby('GmmLabels').mean()

median = data[num_cols].groupby('GmmLabels').median()

df_gmm = pd.concat([mean, median], axis = 0)

df_gmm.index = ['group_0 Mean', 'group_1 Mean','group_0 Median', 'group_1 Me

df_gmm[original_features].T
```

Out[30]:

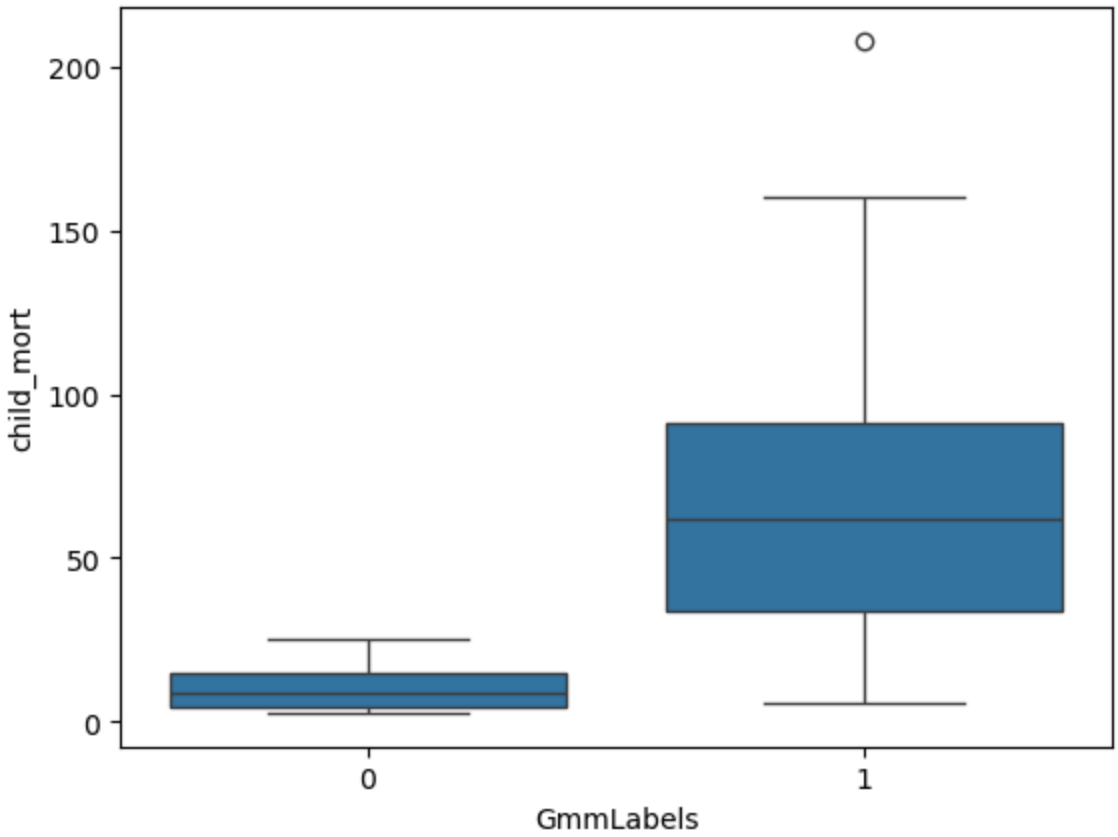
	group_0 Mean	group_1 Mean	group_0 Median	group_1 Median
child_mort	9.951807	66.251190	8.60	62.100
exports	50.393976	31.934512	42.30	28.550
health	7.623253	6.017738	7.72	5.275
imports	49.151807	44.655546	45.30	42.950
income	29058.072289	5373.130952	22700.00	3355.000
inflation	4.593012	10.932690	2.87	7.420
life_expec	77.337349	63.854762	76.80	65.300
total_fer	1.853373	4.029524	1.84	3.880
gdpp	23478.072289	2575.404762	13500.00	1365.000

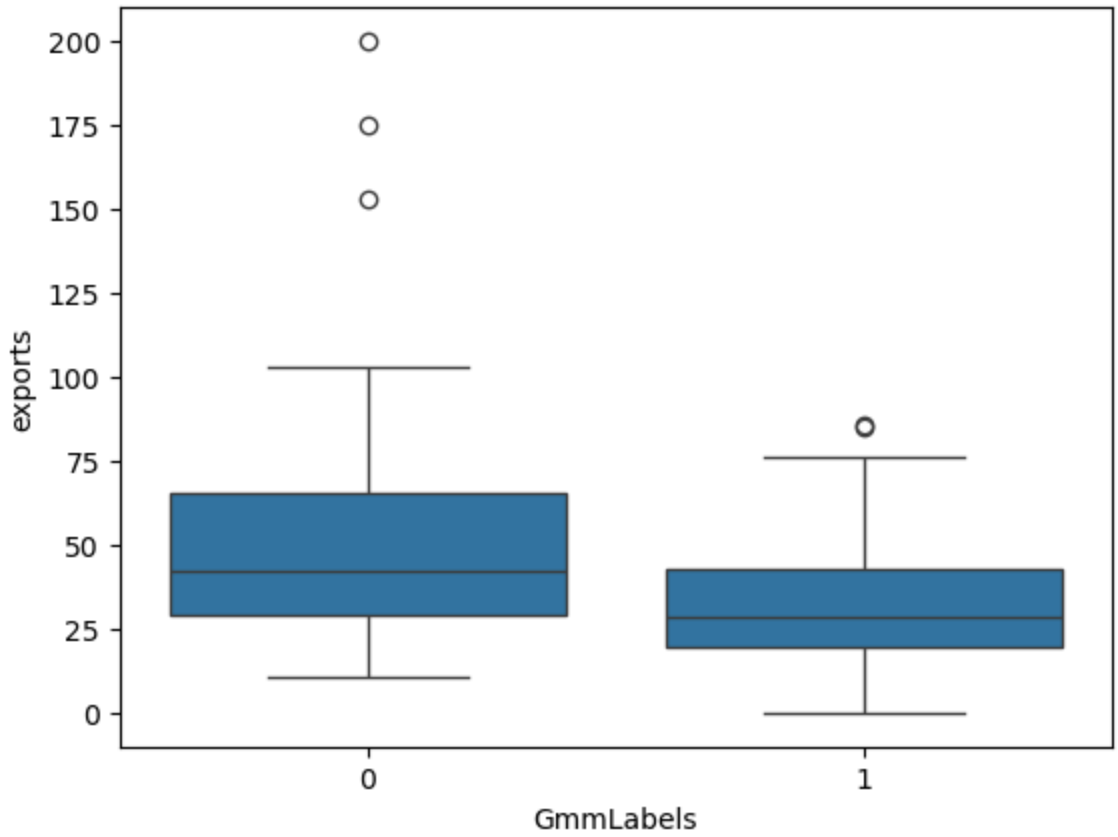
Observation:

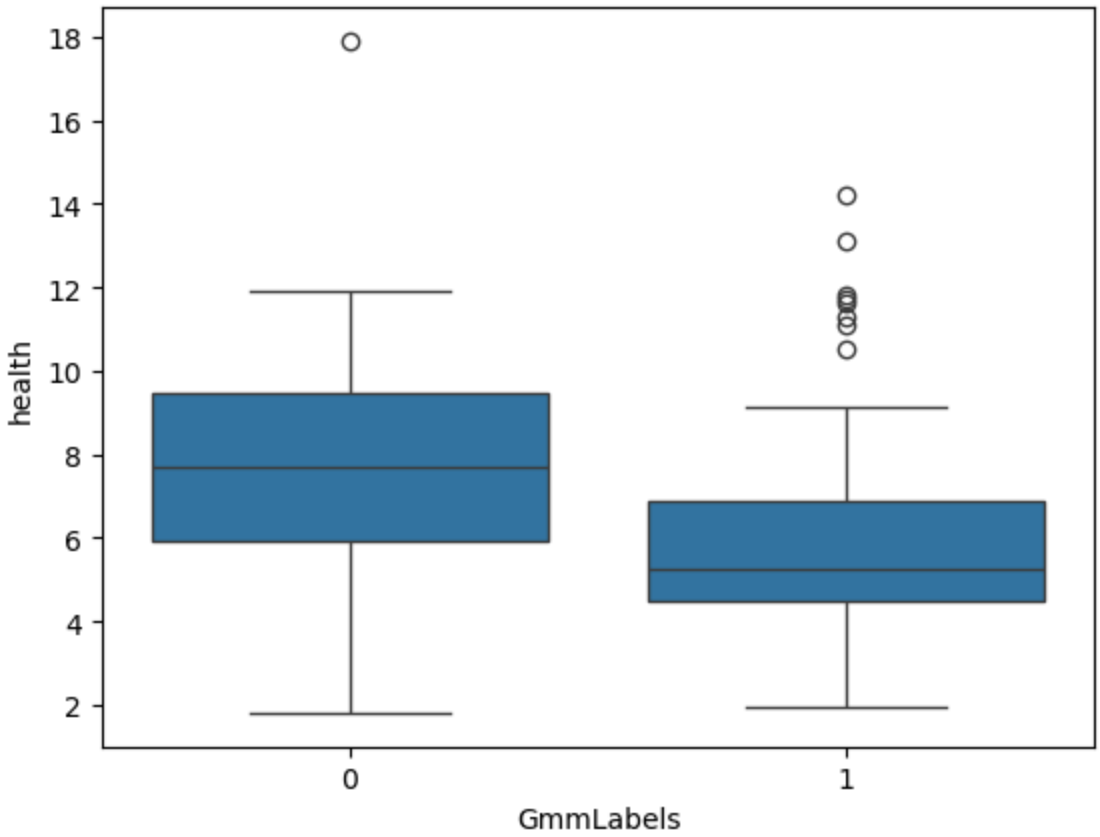
- Cluster 1 belongs to high income countries, Cluster 0 belongs to lower income countries, and the rest of the countries are in Cluster 2.

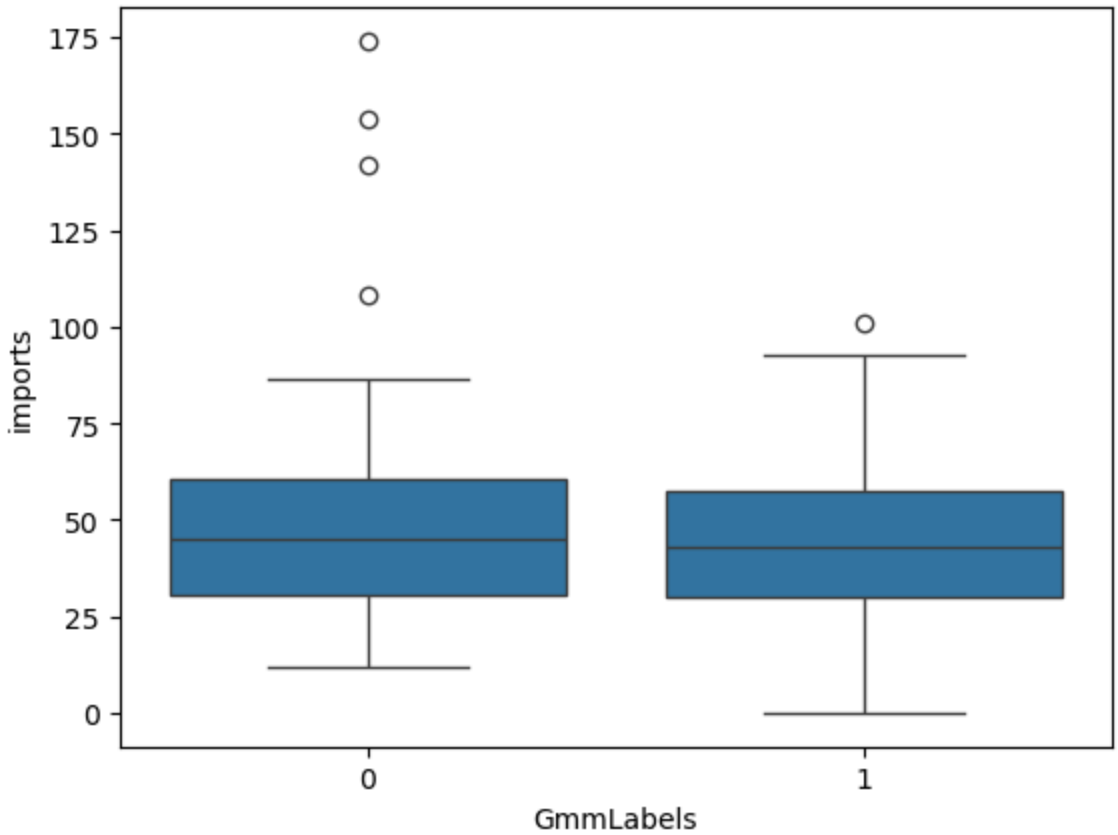
In [31]:

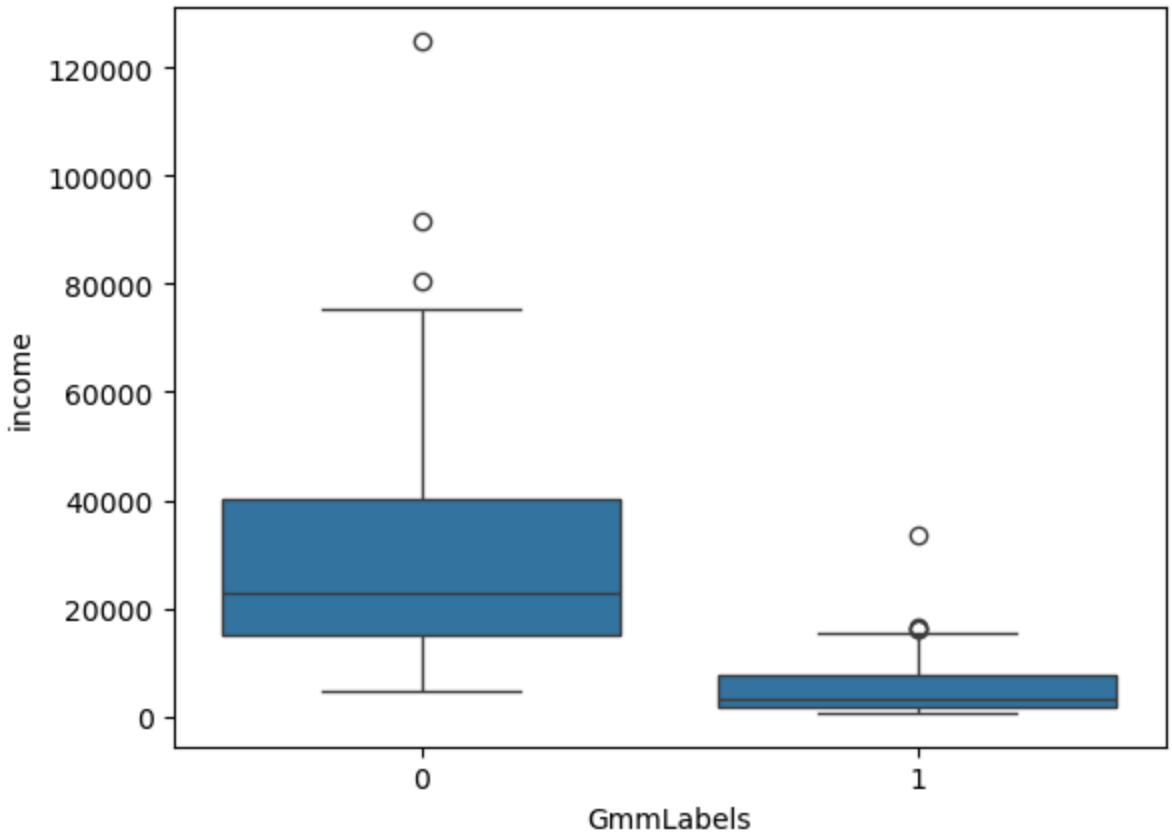
```
cols_visualise = ['child_mort', 'exports', 'health', 'imports', 'income', 'i

for col in cols_visualise:
    sns.boxplot(x = 'GmmLabels', y = col, data = data)

    plt.show()
```

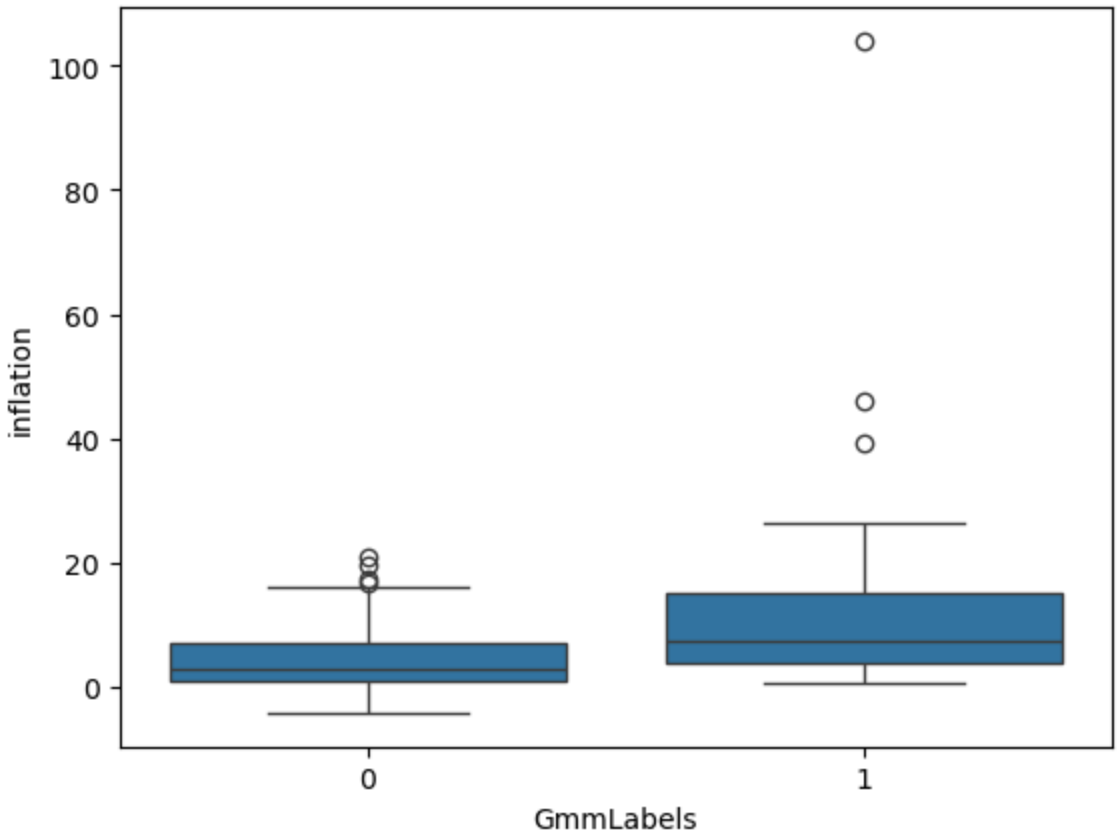

200
150
100
50
0
child_mort
0
1
GmmLabels

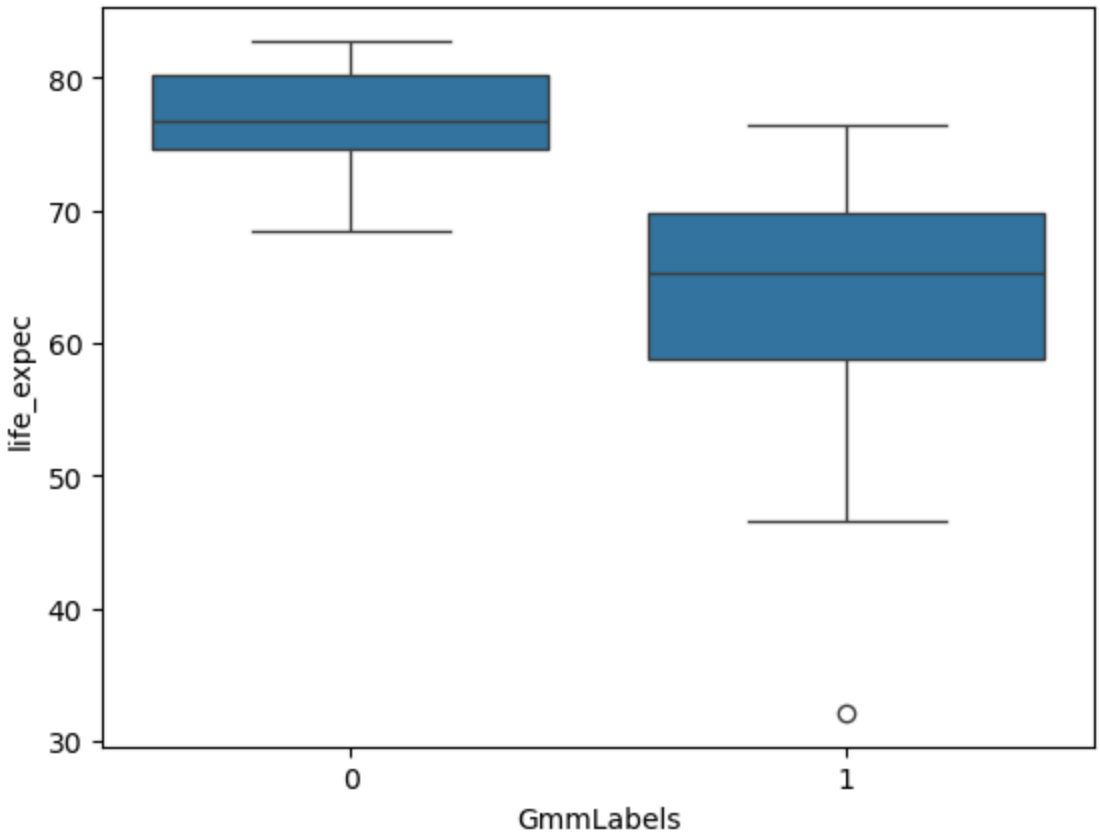

200
175
150
125
100
75
50
25
0
exports
0
1
GmmLabels

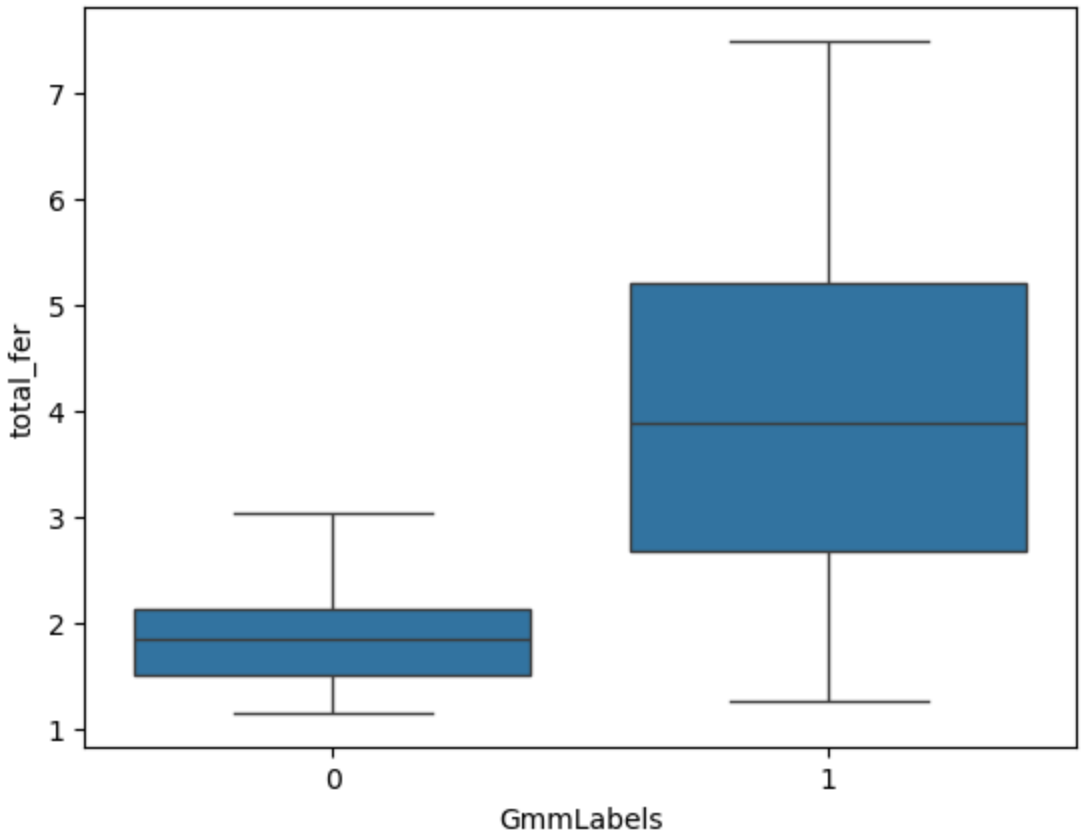

18
16
14
12
10
8
6
4
2
health
0
1
GmmLabels

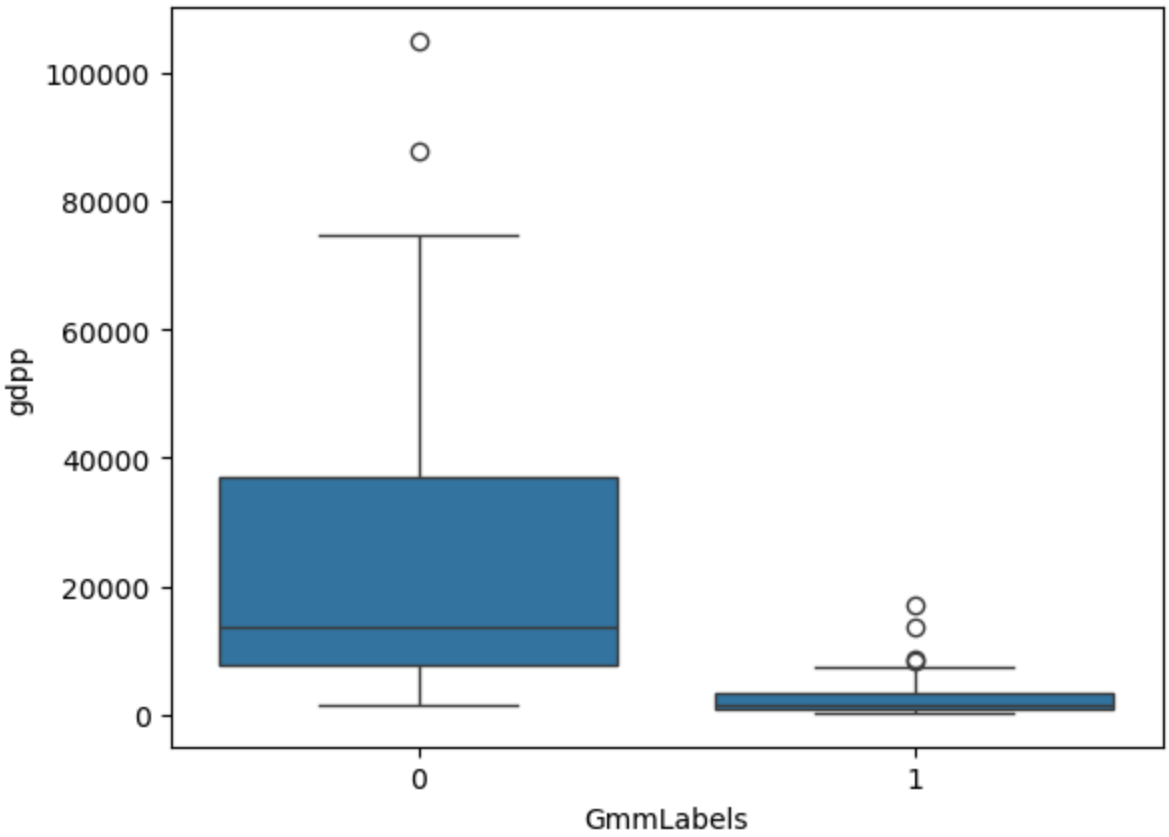

175
150
125
100
75
50
25
0
imports
0
1
GmmLabels


120000
100000
80000
60000
40000
20000
0
income
0
1
GmmLabels


100
80
60
40
20
0
inflation
0
1
GmmLabels


life_expec
80
70
60
50
40
30
0
1
GmmLabels


total_fer
7
6
5
4
3
2
1
0
1
GmmLabels



Cluster Profiles:

- This clustering solution looks very similar to the once created using K-Medoids with one cluster of 'high income' countries, one of 'low income' and one of 'all the others'. But on closer inspection, we can identify some important differences in this solution using GMM.
- Cluster 1 seems to be of 'developed' countries but this time the median values for all the key indicators have all improved in comparison to the same cluster obtained from K-Medoids, with a higher GDP per capita, higher income, higher exports and imports and marginally higher life expectancy. At the same time, it has lower inflation rates, lower child mortality rates, and lower fertility as well. Overall, we can say that this cluster has become more 'pure' in comparison to the one from K-Medoids.
- Cluster 0 seems to be of 'underdeveloped' countries but this time the median values for all the key indicators have improved in comparison to the corresponding K-Medoids cluster. For e.g., it has higher GDP per capita, higher income per person, higher exports and imports, and slightly better health expenditure and life expectancy. That means that this cluster of 'underdeveloped' countries has become less 'pure'.
- Both of the above points can give an idea of what might have happened to the third cluster, i.e., Cluster 2. It was a mix of 'underdeveloped' & 'developing' countries and continues to be so, but it has gained some countries on the rich end of the

spectrum, and some countries on the 'underdeveloped' end have moved to the last cluster.

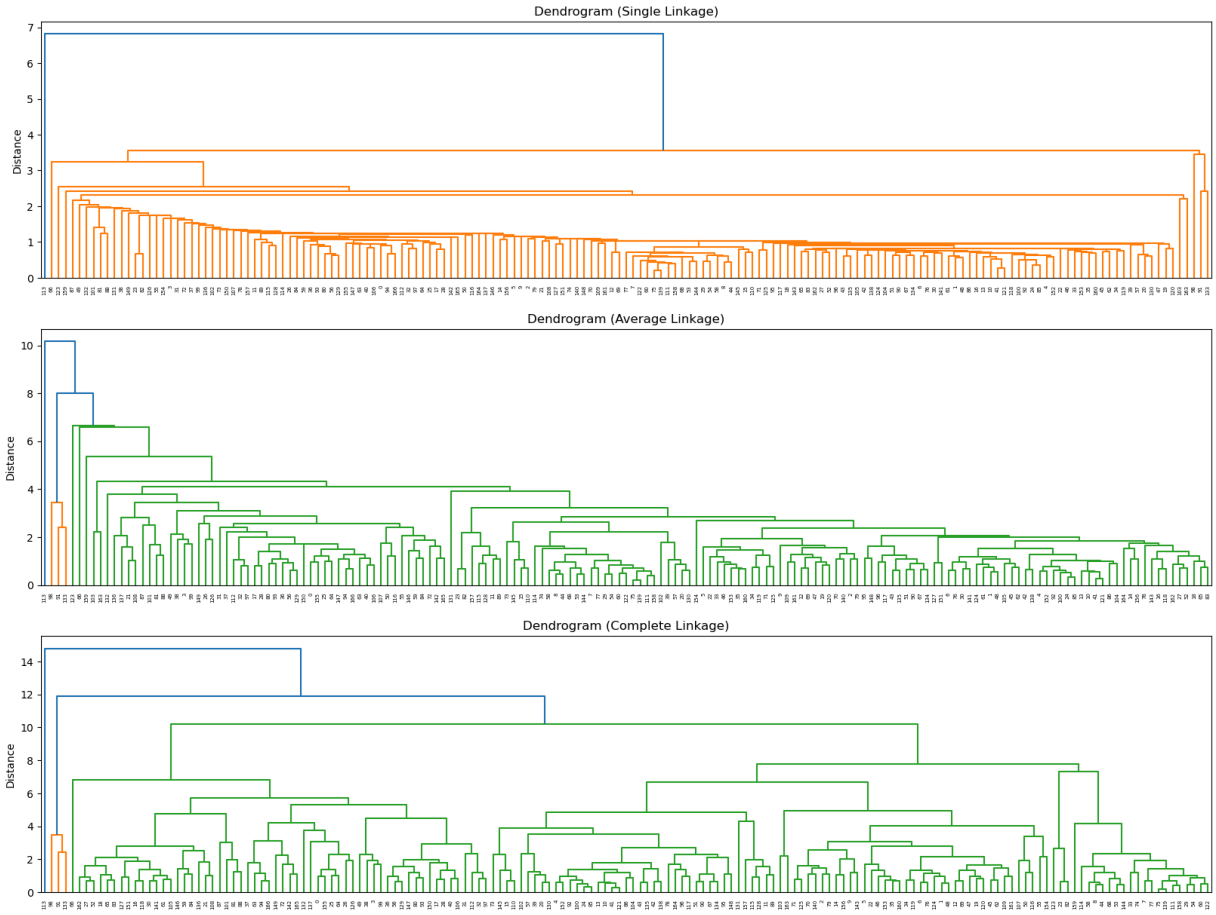
Overall, this is a slightly more evenly distributed clustering solution than K-Medoids.

Hierarchical Clustering

- Let's try to create clusters using Agglomerative Hierarchical clustering.
- Here, we decide the number of clusters using a concept called `Dendrogram` which is a tree-like diagram that records the sequences of merges or splits.

In [32]:
```
from scipy.cluster.hierarchy import dendrogram, linkage
```

In [33]:
```
# The List of all linkage methods to check
methods = ['single',
           'average',
           'complete']

# Create a subplot image
fig, axs = plt.subplots(len(methods), 1, figsize = (20, 15))

# Enumerate through the list of all methods above, get linkage and plot denc
for i, method in enumerate(methods):
    Z = linkage(data_scaled, metric = 'euclidean', method = method)

    dendrogram(Z, ax = axs[i]);

    axs[i].set_title(f'Dendrogram ({method.capitalize()} Linkage)')

    axs[i].set_ylabel('Distance')
```

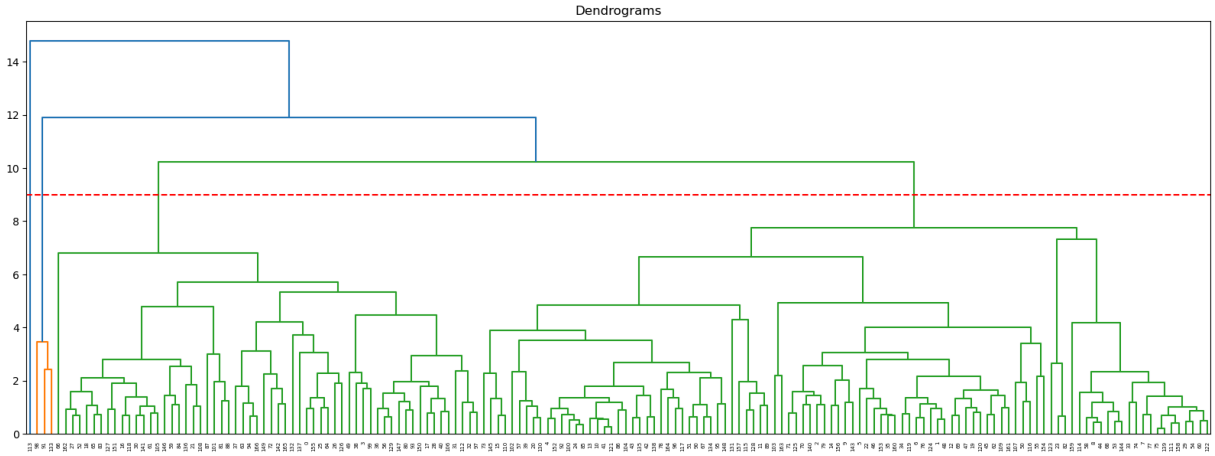


Observations:

- We can see that the complete linkage gives better separated clusters. A cluster is considered better separated if the vertical distance connecting those clusters is higher.
- Now, we can set a threshold distance and draw a horizontal line. The number of clusters will be the number of vertical lines which are being intersected by the line drawn using the threshold.
- The branches of this dendrogram are cut at a level where there is a lot of 'space' to cut them, that is where the jump in levels of two consecutive nodes is large
- Here, we can choose to cut it at ~9 since the space between the two nodes is largest.

In [34]:
```
plt.figure(figsize = (20, 7))

plt.title("Dendrograms")

dend = dendrogram(linkage(data_scaled, method = 'complete'))

plt.axhline(y = 9, color = 'r', linestyle = '--')
```

Out[34]:
```
<matplotlib.lines.Line2D at 0x30feccf90>
```



Observations:

- We can see that the if we create a horizontal line at threshold distance ~ 9, it cuts 4 vertical lines, i.e., we get 4 different clusters.
- Let's fit the algorithms using 4 as the number of clusters.

In [35]:
```
# Clustering with 4 clusters
hierarchical = AgglomerativeClustering(n_clusters = 4, linkage = 'complete')

hierarchical.fit(data_scaled)
```

Out[35]:
```
AgglomerativeClustering(linkage='complete', n_clusters=4)
```

In [36]:
```
data_scaled_copy['HCLabels'] = hierarchical.labels_

data['HCLabels'] = hierarchical.labels_
```

In [37]:
```
data.HCLabels.value_counts()
```

Out[37]:
```
HCLabels
0    103
1     60
2      3
3      1
Name: count, dtype: int64
```

Observations:

- The count of observations in the resulting 4 clusters is unevenly distributed.
- We have two clusters with only 3 countries and 1 country, respectively. Let's check the countries in these clusters.

In [38]:
```
# Checking 3 countries in cluster 2
data[data.HCLabels == 2]
```

Out[38]:

	country	child_mort	exports	health	imports	income	inflation	life_expec
91	Luxembourg	2.8	175.0	7.77	142.0	91700	3.620	81.3
98	Malta	6.8	153.0	8.65	154.0	28300	3.830	80.3
133	Singapore	2.8	200.0	3.96	174.0	72100	-0.046	82.7

Observations:

- Similar to K-Means, we got a separate cluster for 3 small countries with the highest values for imports and exports - Luxembourg, Malta, Singapore.

In [39]:
```
# Checking 1 country in cluster 3
data[data.HCLabels == 3]
```

Out[39]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	tota
113	Nigeria	130.0	25.3	5.07	17.4	5150	104.0	60.5	

Observations:

- Cluster 3 consists of just one country - Nigeria.
- Nigeria has an inflation rate of 104 which is the highest inflation rate in this dataset. This might have made its distance with the other clusters significantly higher not allowing it to merge with any of those data points.

In [40]:
```
# Calculating the mean and the median of the original data for each label
original_features = ['child_mort', 'exports', 'health', 'imports', 'income',
num_cols = original_features + ['HCLabels']
mean = data[num_cols].groupby('HCLabels').mean()

median = data[num_cols].groupby('HCLabels').median()

df_hierachical = pd.concat([mean, median], axis = 0)

df_hierachical.index = ['group_0 Mean', 'group_1 Mean', 'group_2 Mean', 'gro

df_hierachical[original_features].T
```

Out[40]:

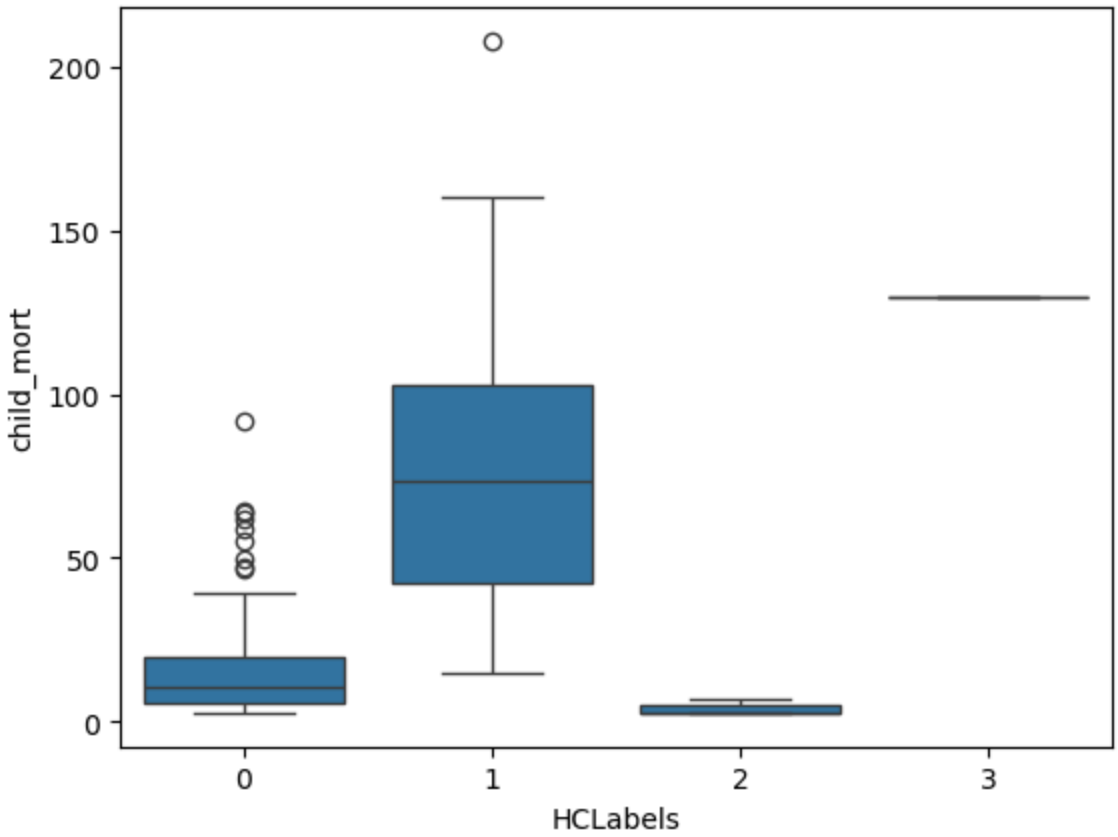
	group_0 Mean	group_1 Mean	group_2 Mean	group_3 Mean	group_0 Median	group_1 Median
child_mort	16.678641	75.513333	4.133333	130.00	10.80	73.300
exports	42.532806	32.183667	176.000000	25.30	38.70	28.900
health	7.013883	6.505667	6.793333	5.07	6.91	5.685
imports	42.438504	49.535000	156.666667	17.40	38.40	47.650
income	23425.533981	4218.050000	64033.333333	5150.00	17800.00	2500.000
inflation	6.723262	8.261100	2.468000	104.00	4.49	5.860
life_expec	75.471845	61.740000	81.433333	60.50	76.10	61.300
total_fer	2.074660	4.477333	1.380000	5.84	1.93	4.710
gdpp	18053.689320	2174.233333	57566.666667	2330.00	10700.00	1185.000

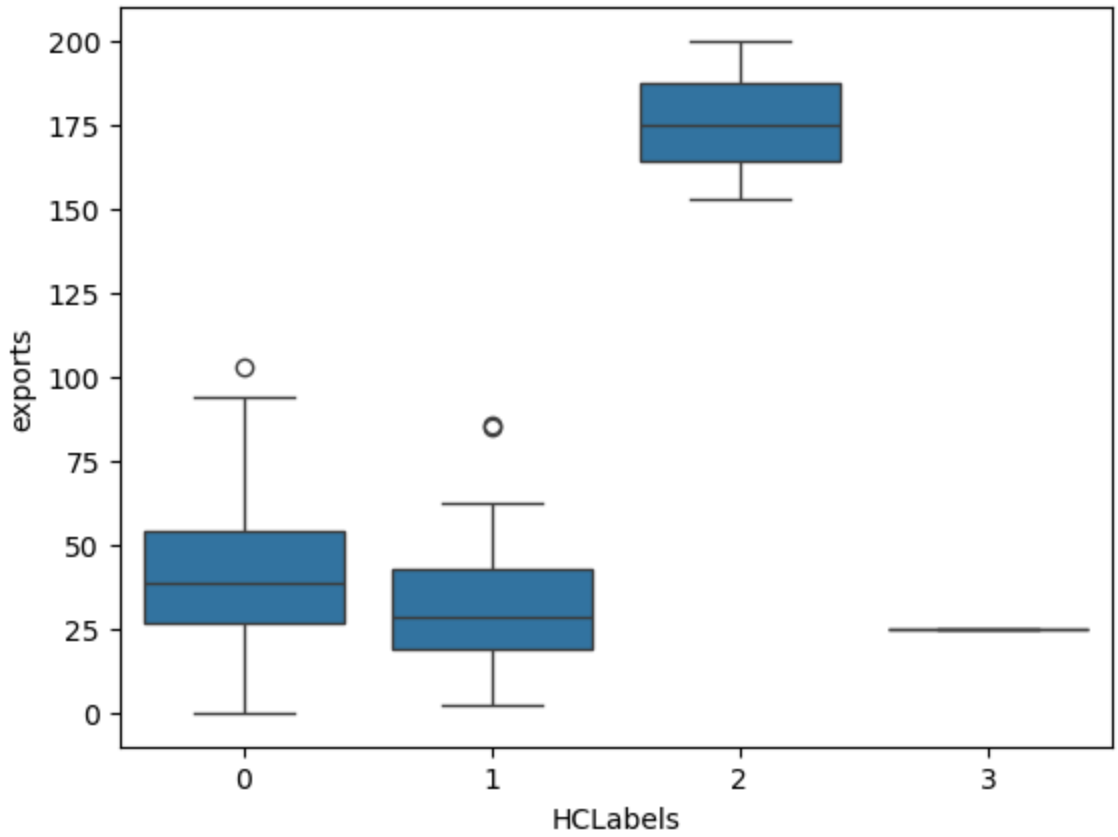
Observations:

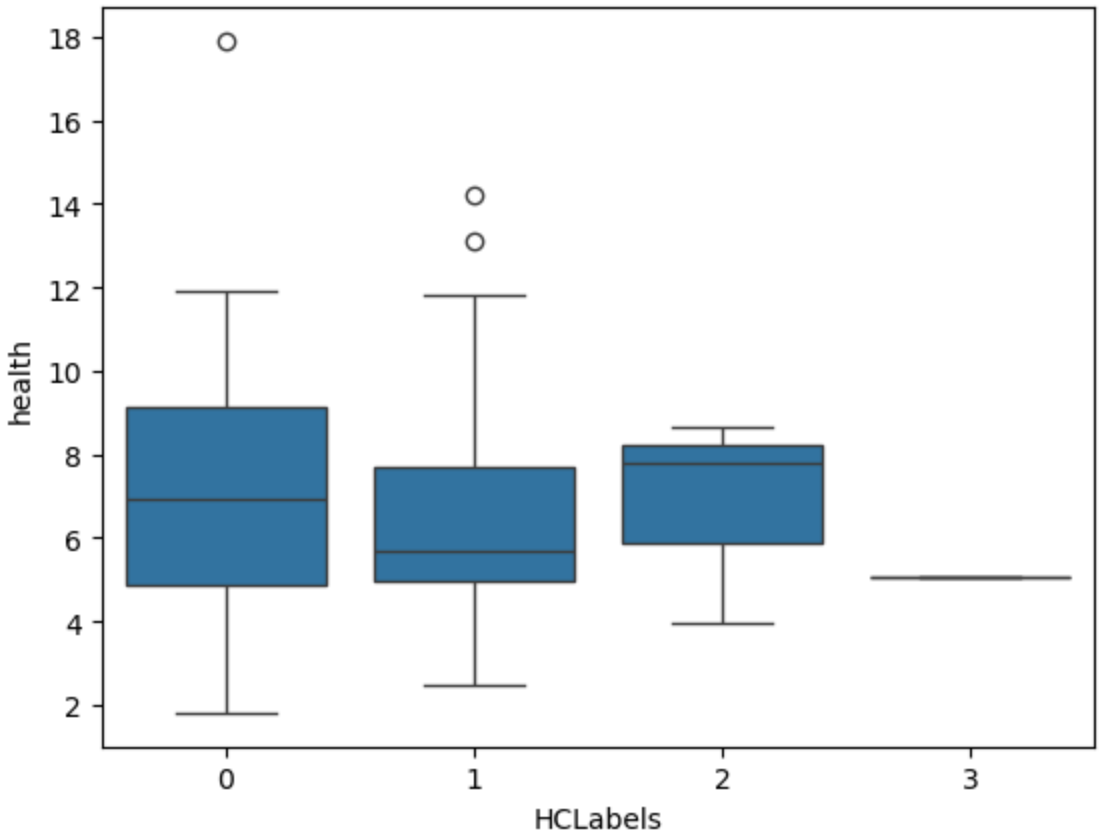
- It looks like Cluster 2 has only 3 countries with high income and high gdpp, Cluster 1 has low income and low gdpp countries, and the rest of the countries are in cluster 0 except for one country which is in cluster 3.

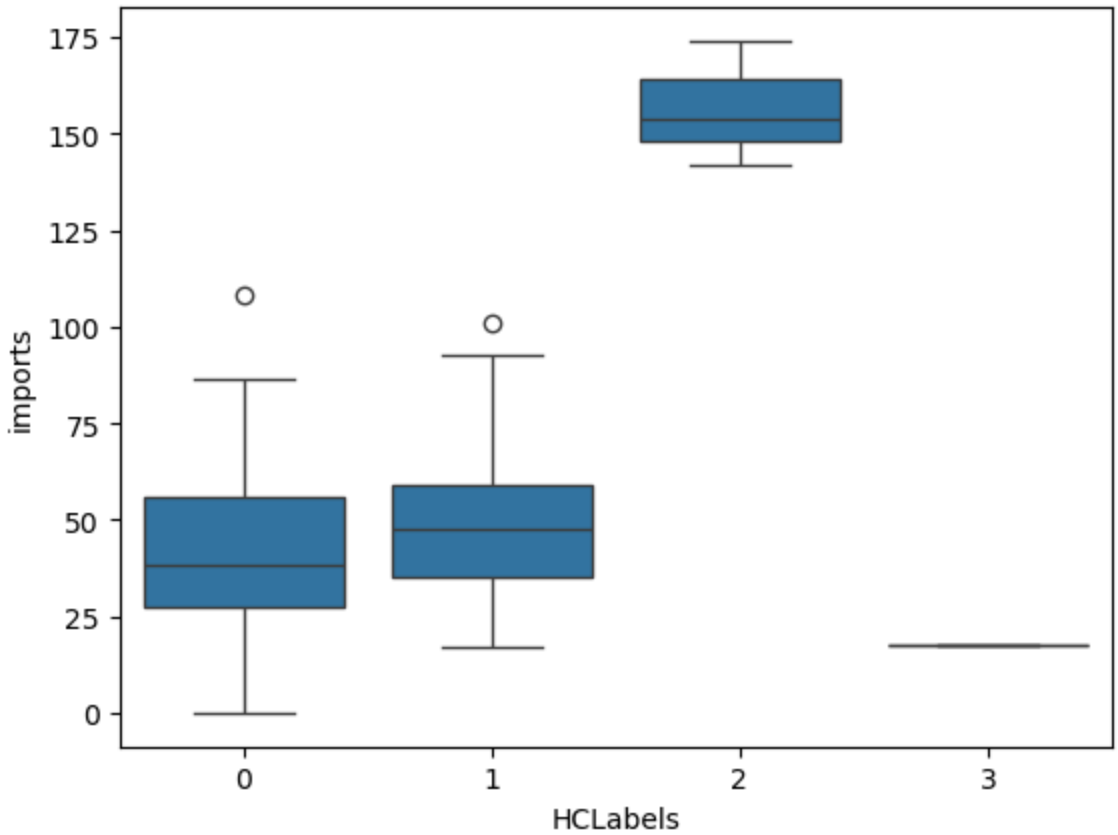
Let's try to visualize the boxplots of different attributes for each cluster to see if we can spot some more granular patterns.

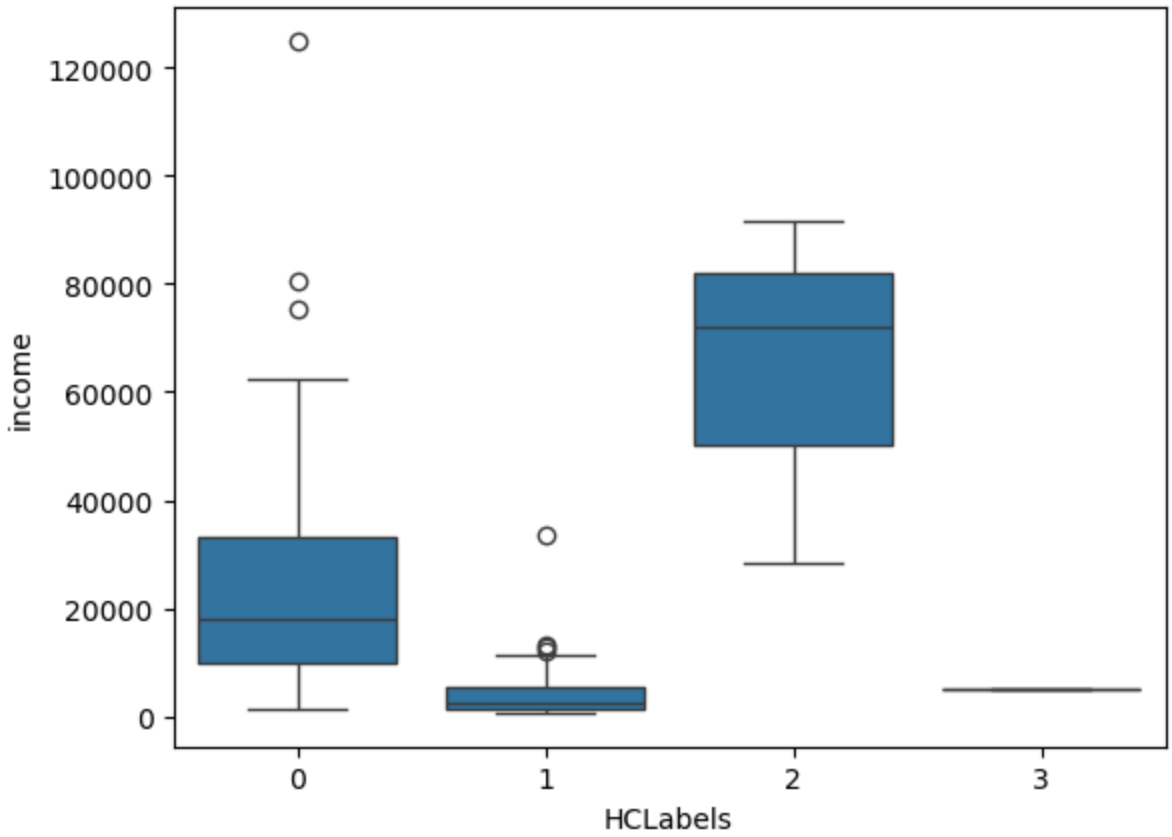
In [41]:
```
cols_visualise = ['child_mort', 'exports', 'health', 'imports', 'income', 'i

for col in cols_visualise:
    sns.boxplot(x = 'HCLabels', y = col, data = data)
    plt.show()
```

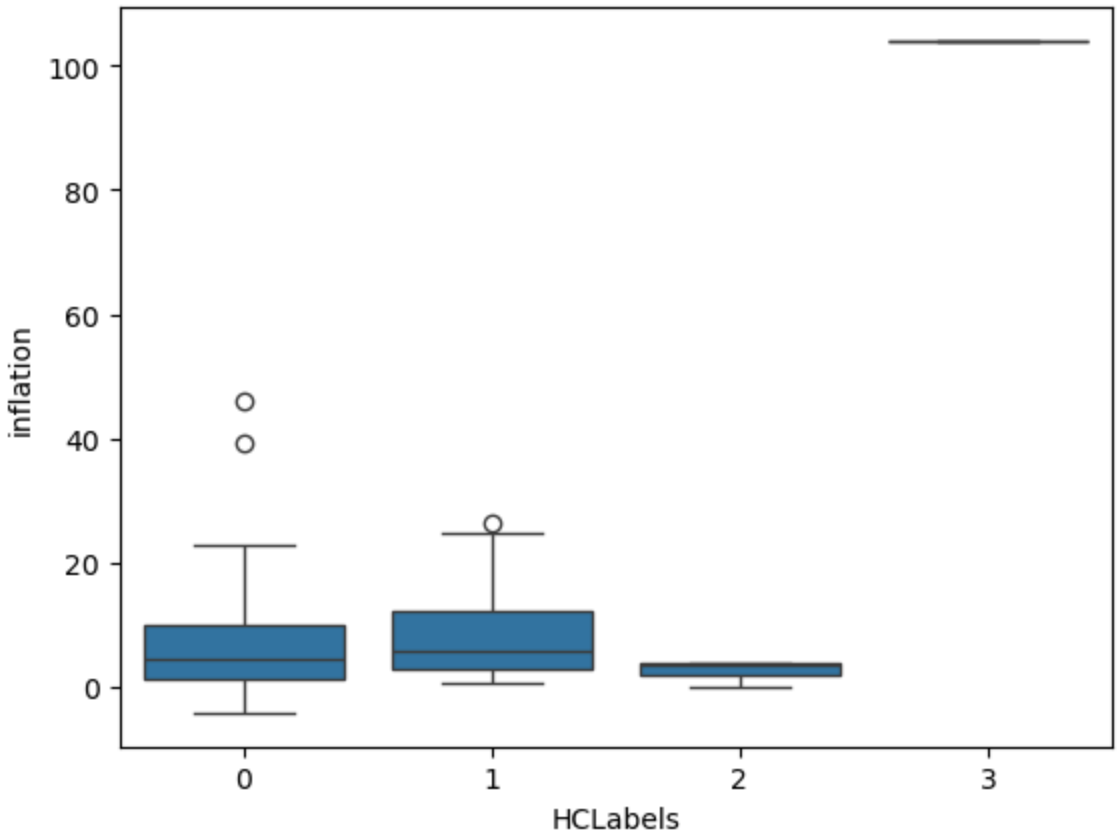

child_mort
200
150
100
50
0
0
1
2
3
HCLabels

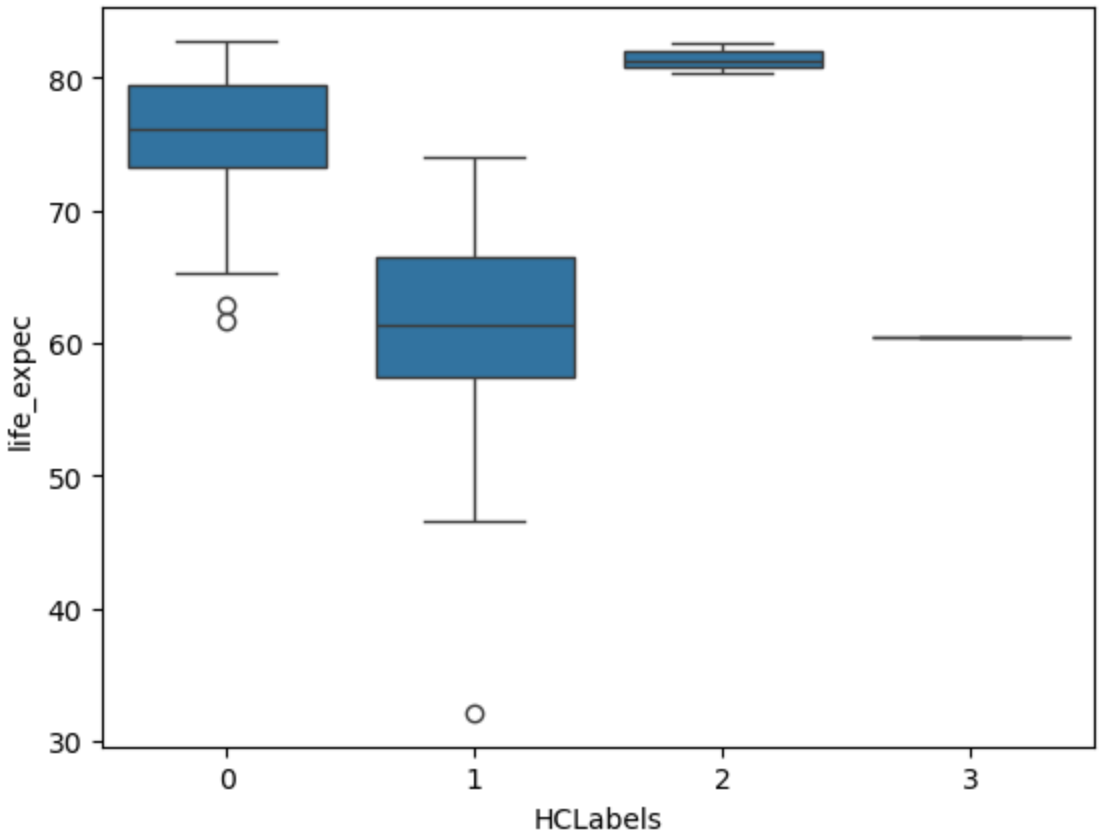

exports
200
175
150
125
100
75
50
25
0
0
1
2
3
HCLabels

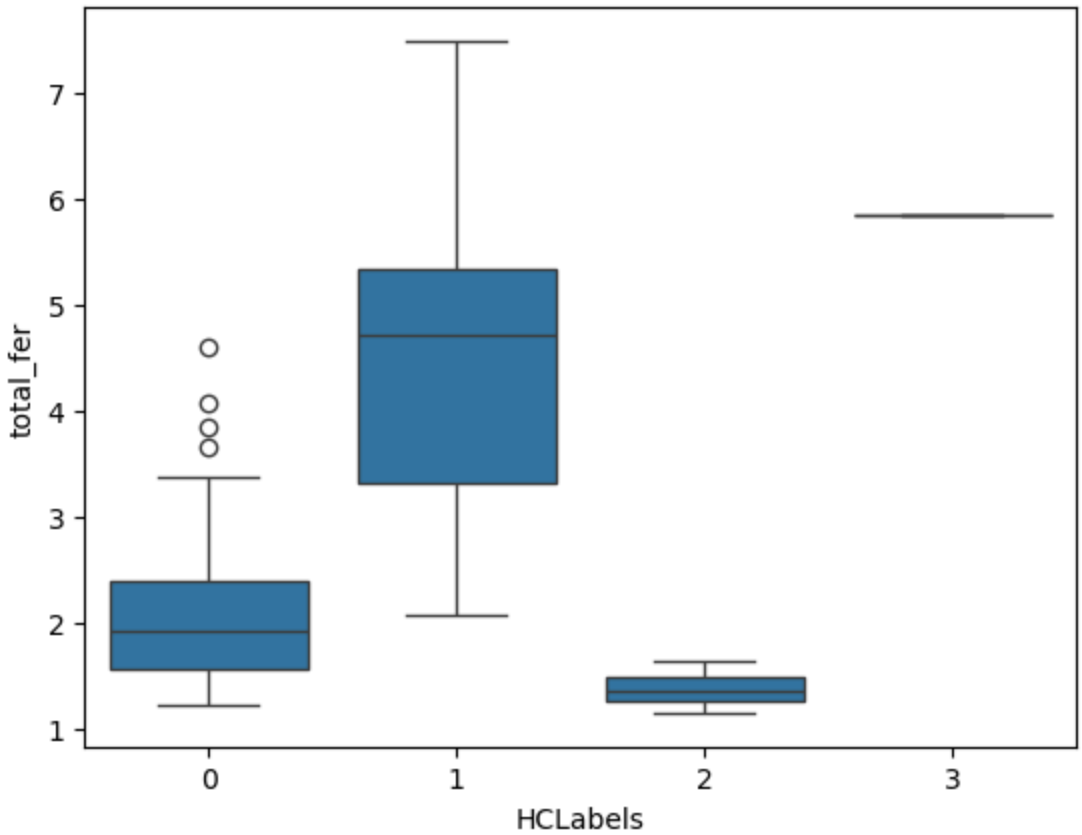

health
18
16
14
12
10
8
6
4
2
0
1
2
3
HCLabels

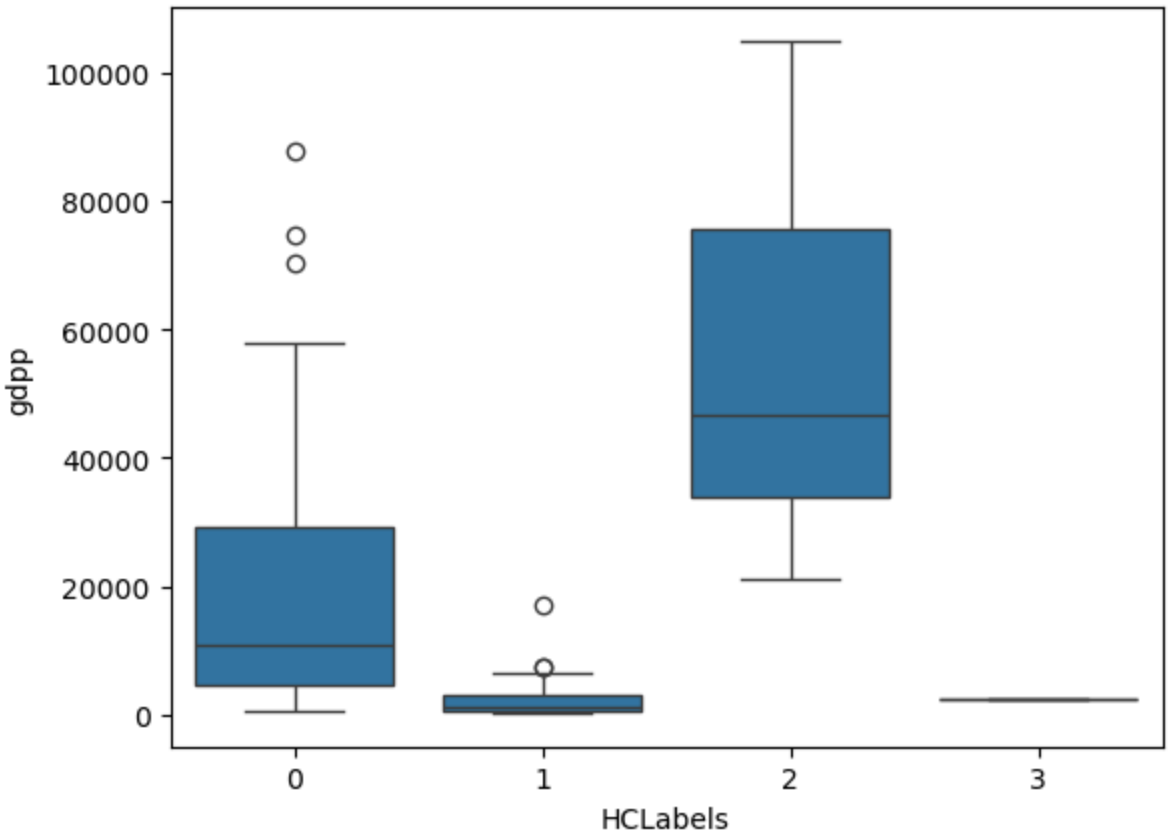

imports
175
150
125
100
75
50
25
0
0
1
2
3
HCLabels


120000
100000
80000
60000
40000
20000
0
income
0
1
2
3
HCLabels


100
80
60
40
20
0
inflation
0
1
2
3
HCLabels


life_expec
80
70
60
50
40
30
0
1
2
3
HCLabels


total_fer
7
6
5
4
3
2
1
0
1
2
3
HCLabels



Observations:

- The results from hierarchical clustering seem to be difficult to distinguish and comment on especially because of one cluster which contains 103 countries

Let's try to use DBSCAN algorithm

DBSCAN

In [42]:
```
dbs = DBSCAN(eps = 1)

data_scaled_copy['DBSLabels'] = dbs.fit_predict(data_scaled)

data['DBSLabels'] = dbs.fit_predict(data_scaled)
```

In [43]:
```
data['DBSLabels'].value_counts()
```

Out[43]:
```
DBSLabels
-1    90
 0    55
 1    17
 2     5
Name: count, dtype: int64
```

In [44]:
```
# Calculating the mean and the median of the original data for each label
original_features = ['child_mort', 'exports', 'health', 'imports', 'income',
num_cols = original_features + ['DBSLabels']
```

```
mean = data[num_cols].groupby('DBSLabels').mean()

median = data[num_cols].groupby('DBSLabels').median()

df_hierachical = pd.concat([mean, median], axis = 0)

df_hierachical.index = ['group_-1 Mean', 'group_0 Mean', 'group_1 Mean', 'gr

df_hierachical[original_features].T
```

Out[44]:

	group_-1 Mean	group_0 Mean	group_1 Mean	group_2 Mean	group_-1 Median	group_0 Median
child_mort	54.907778	17.130909	4.147059	87.340	50.900	15.70
exports	42.922211	41.525455	35.194118	24.000	36.100	37.00
health	6.254556	6.709455	10.294706	6.256	5.275	6.55
imports	48.265177	49.510909	33.982353	37.200	42.400	51.30
income	16254.611111	13433.090909	38382.352941	1785.600	5170.000	11200.00
inflation	11.155856	4.015527	1.309118	10.486	8.605	3.53
life_expec	67.202222	74.203636	81.076471	55.020	67.700	74.50
total_fer	3.578222	2.067455	1.708235	5.504	3.250	1.92
gdpp	10940.611111	8043.018182	43200.000000	718.600	2775.000	6250.00

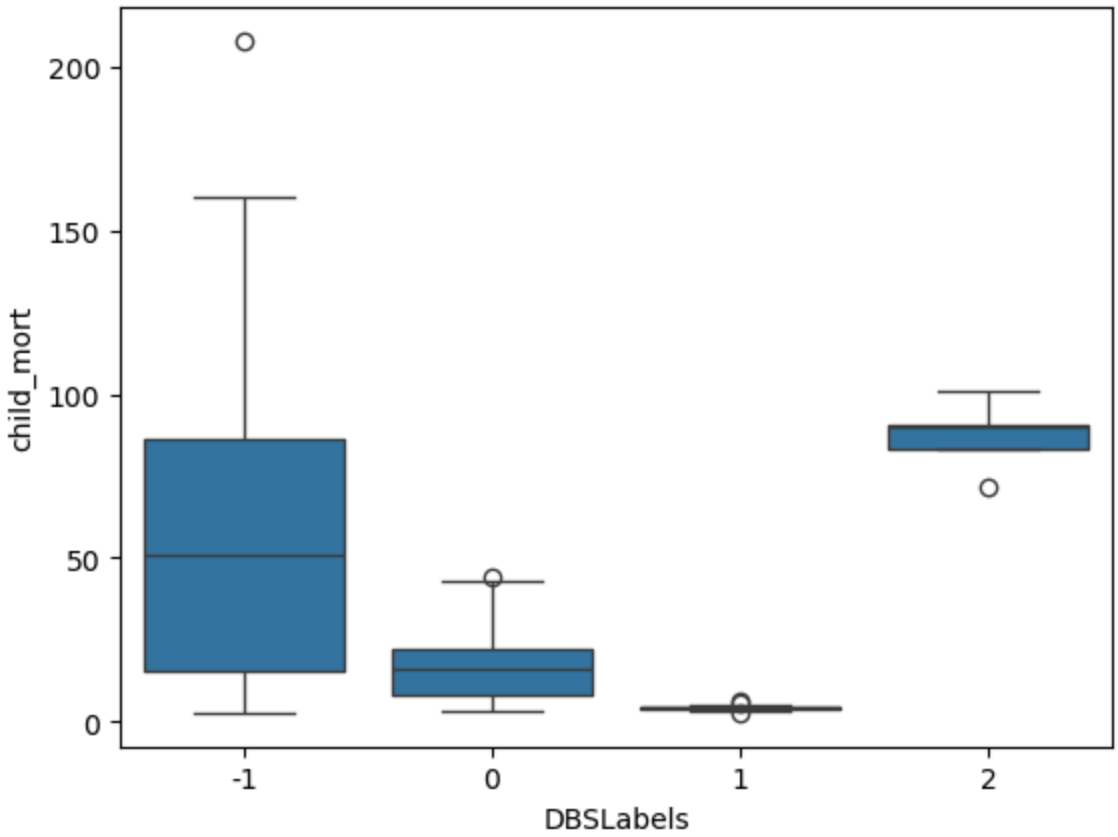
Observations:

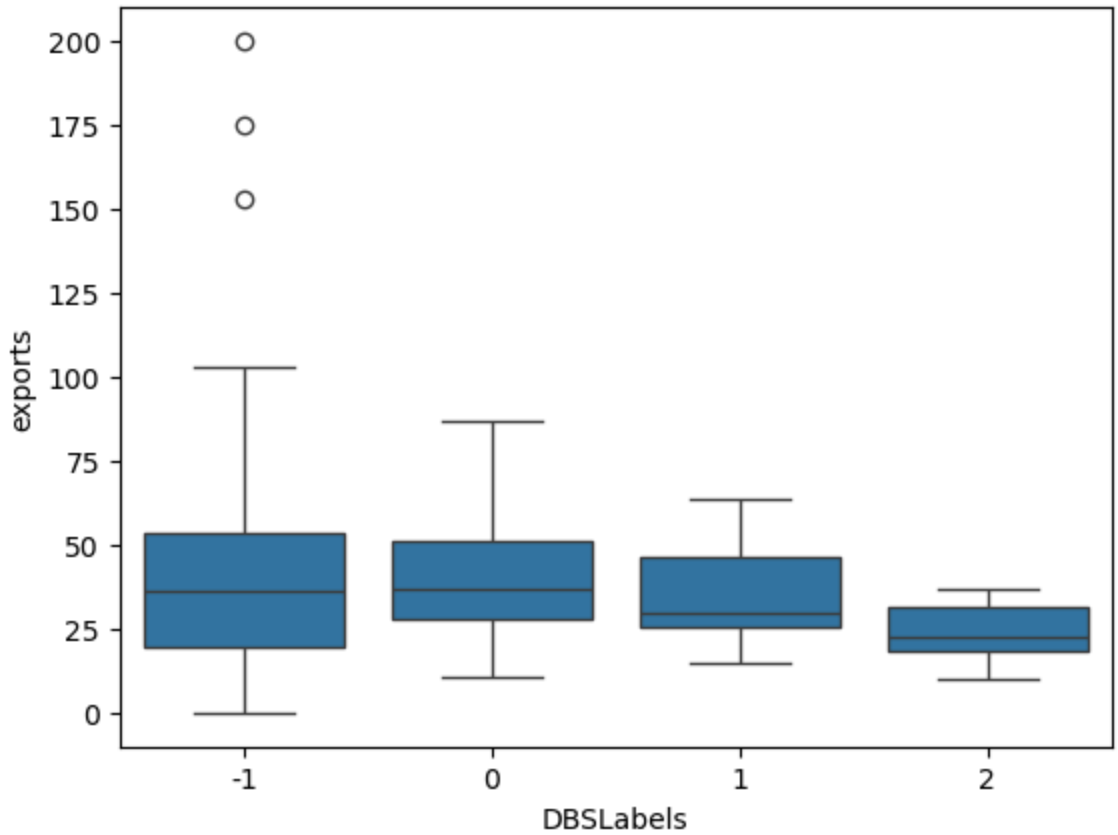
- DBSCAN returns 4 clusters. The countries in 3 of these clusters have similar profiles to the results seen in the other clustering algorithms - high income, low income and moderately developed countries.
- The country profile of the last cluster (cluster -1) seems uncertain. This cluster has a large difference between the mean values and the median values of various attributes implying the presence of outliers in the cluster.

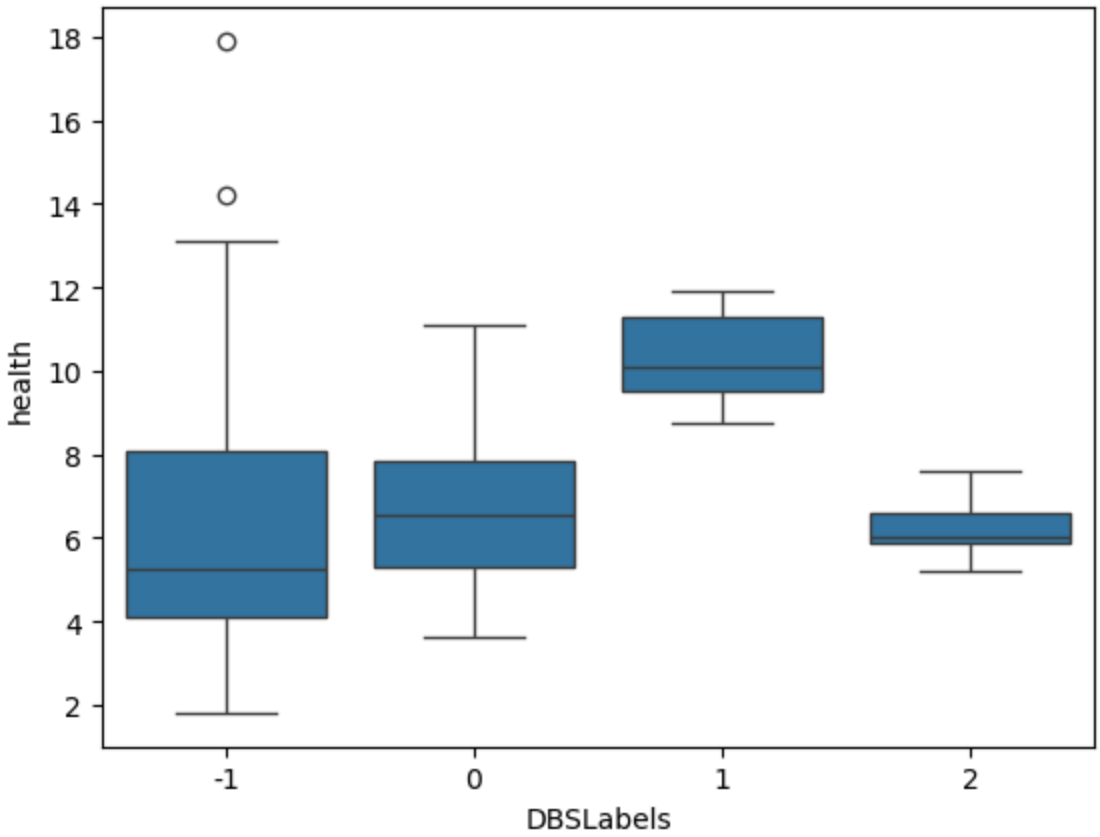
Let's visualize the box plots to comment further on these clusters

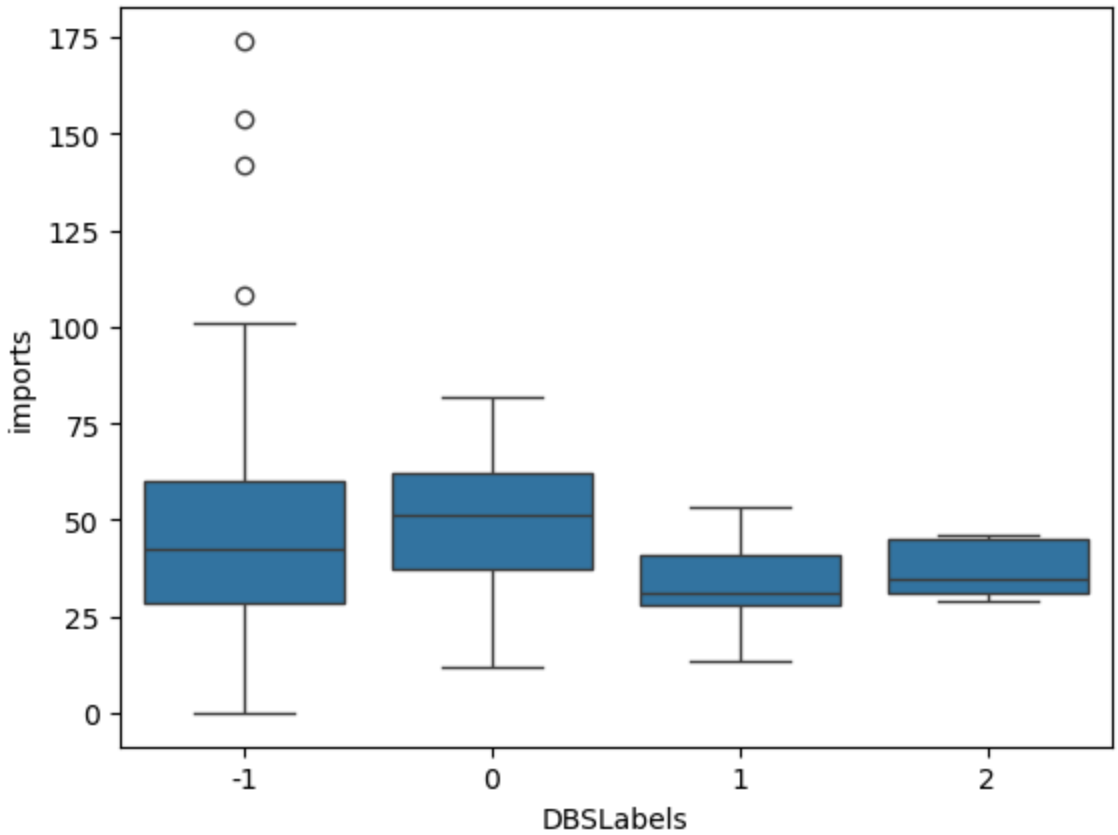
In [45]:

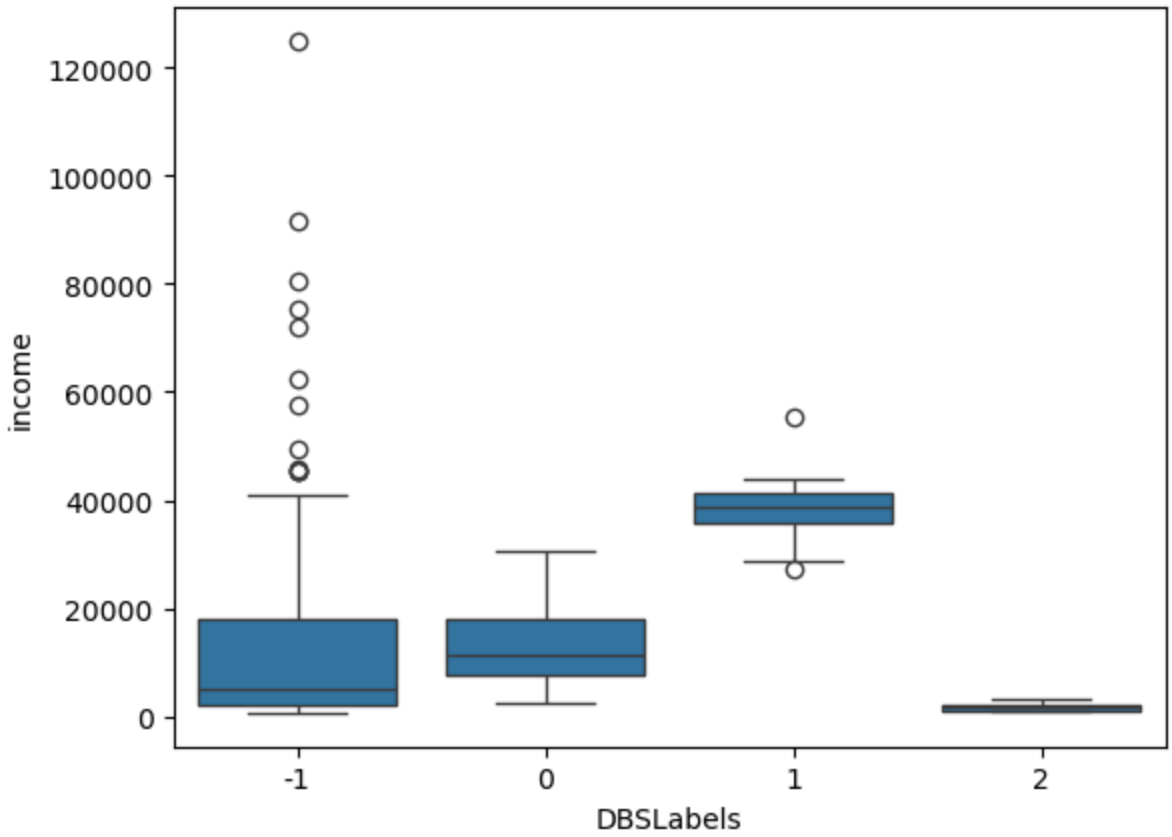
```
for col in cols_visualise:
    sns.boxplot(x = 'DBSLabels', y = col, data = data)

    plt.show()
```

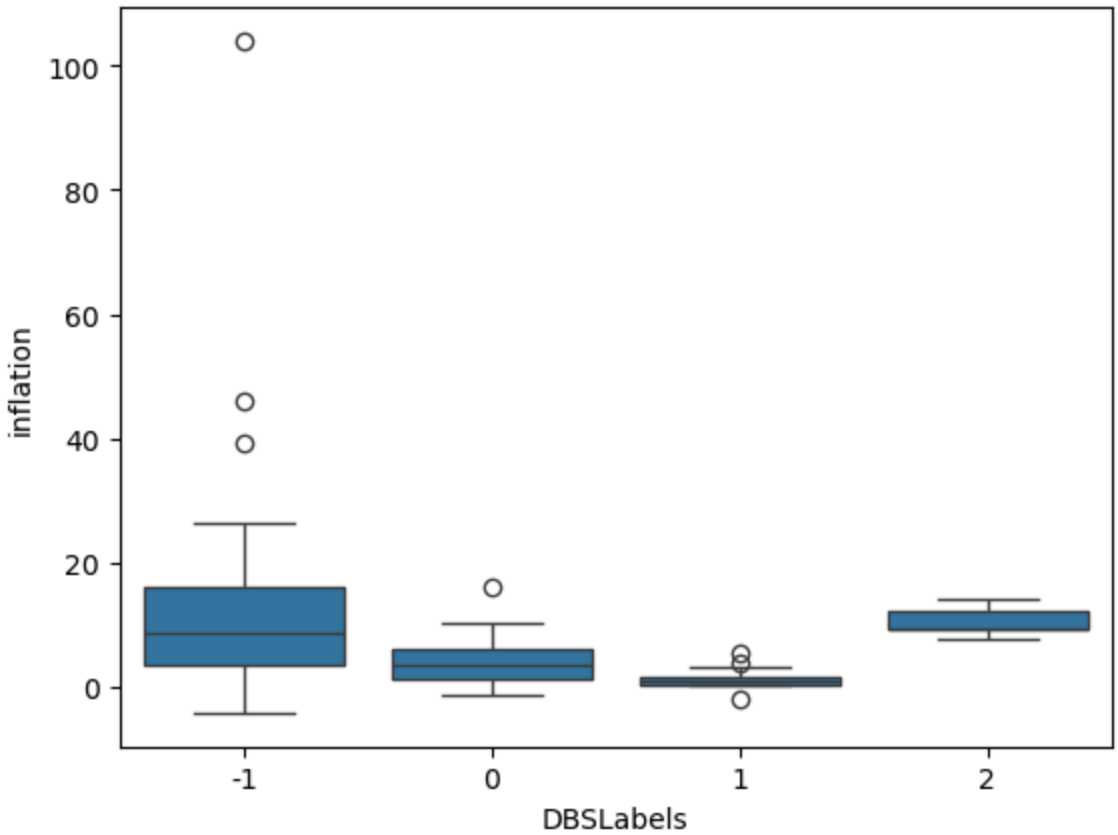

child_mort
200
150
100
50
0
-1
0
1
2
DBSLabels

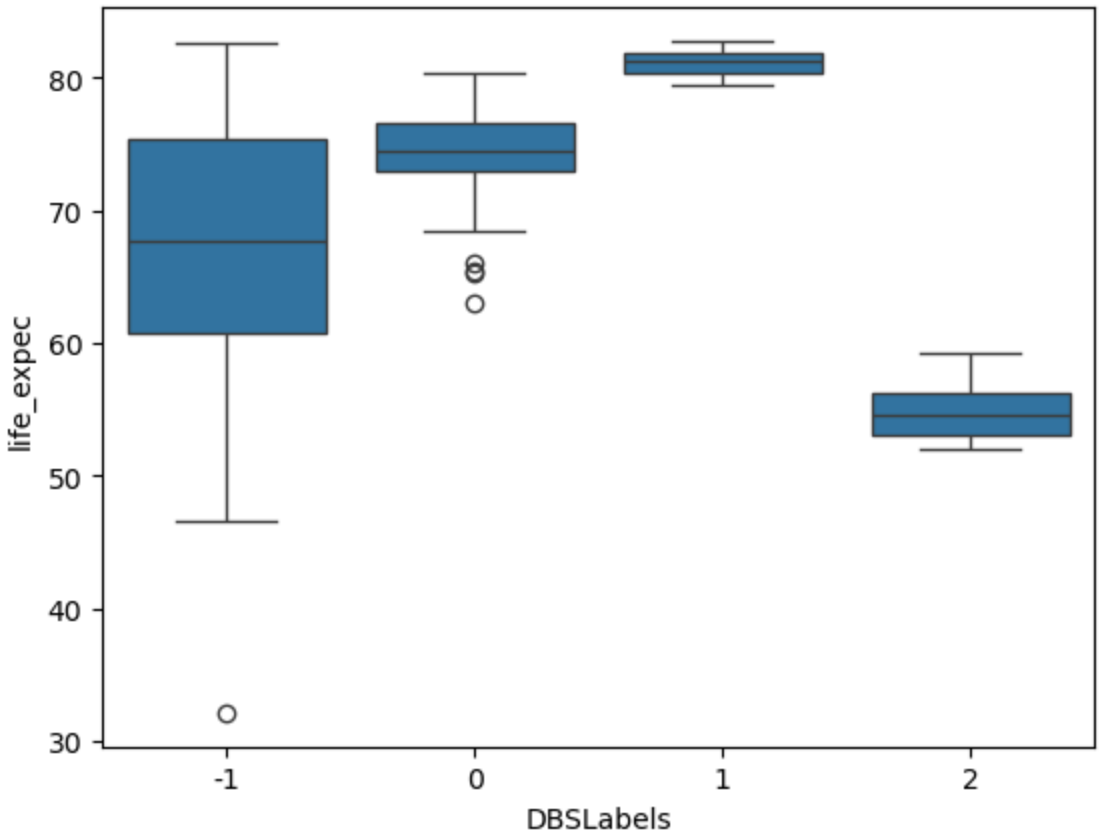

exports
200
175
150
125
100
75
50
25
0
-1
0
1
2
DBSLabels

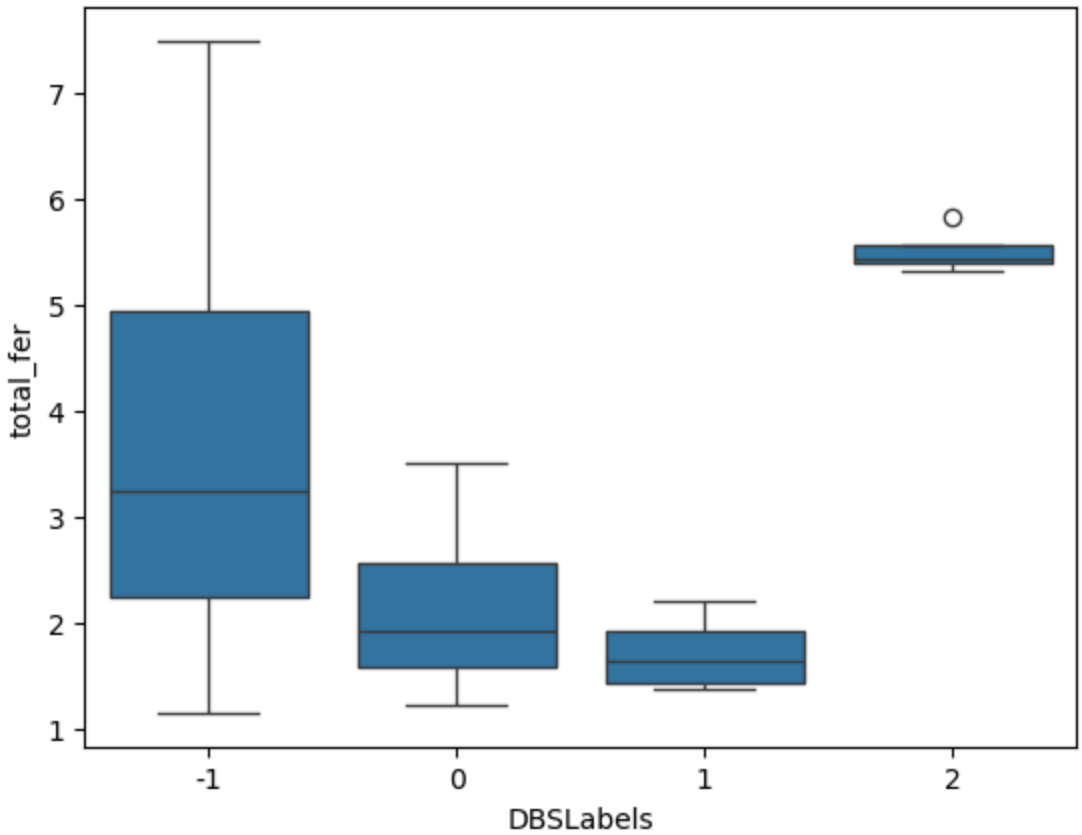

health
18
16
14
12
10
8
6
4
2
-1
0
1
2
DBSLabels

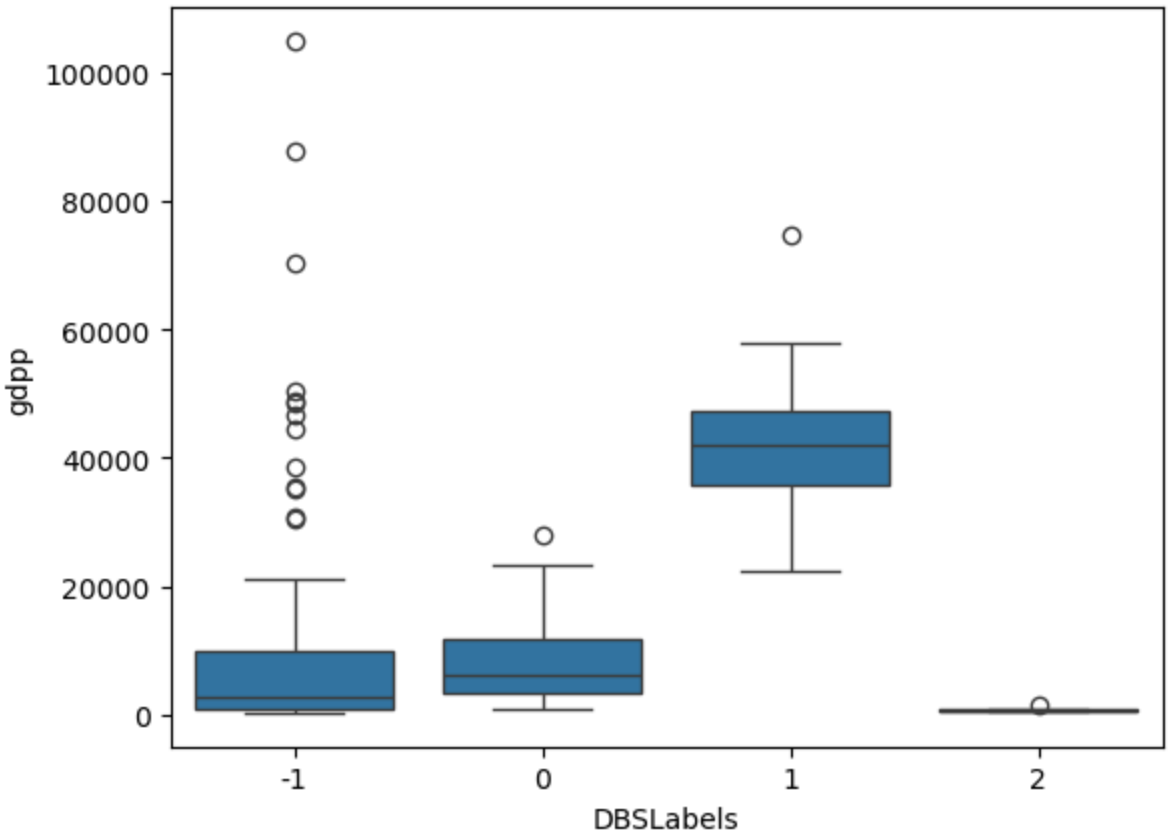

imports
175
150
125
100
75
50
25
0
-1
0
1
2
DBSLabels


120000
100000
80000
60000
40000
20000
0
income
-1
0
1
2
DBSLabels


100
80
60
40
20
0
inflation
-1
0
1
2
DBSLabels


life_expec
80
70
60
50
40
30
-1
0
1
2
DBSLabels


total_fer
7
6
5
4
3
2
1
-1
0
1
2
DBSLabels



Observations

- We can see that while the three clusters (0, 1, and 2) seem to be way more compact across all attributes, cluster -1 consists of extreme outliers on at least one attribute.
- Therefore, it is not adding any value to our cluster analysis. We can explore it further to understand which type of countries it consists of.

Conclusion

The choice of clustering algorithm here will depend on the context and use case. But purely based on foundations of 'what good clustering looks like', one can propose K-Medoids as it has extreme clusters that are more distinct from each other.

In [ ]: