Assume you are appointed as a Data scientist in any international humanitarian NGO, after the recent funding programmes, have been able to raise around 120 million. Now thof the NGO callto choose how to use this money strategically and effectively. The significant issues that comes while making this conclusion are mostly related to choosing the countries that are in the direst need of aid. Your job is to classify 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. Apply Principal component analysis, K-Means Clustering & Hierarchical Clustering.

In [3]:
```
# Required Lib
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [4]:
```
c_data = pd.read_csv('country.csv')
c_data.head()
```

Out[4]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	55
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	40
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	44
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	35
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	122

In [5]:
```
c_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]:
```
# Checking for null values
c_data.isnull().sum()
```

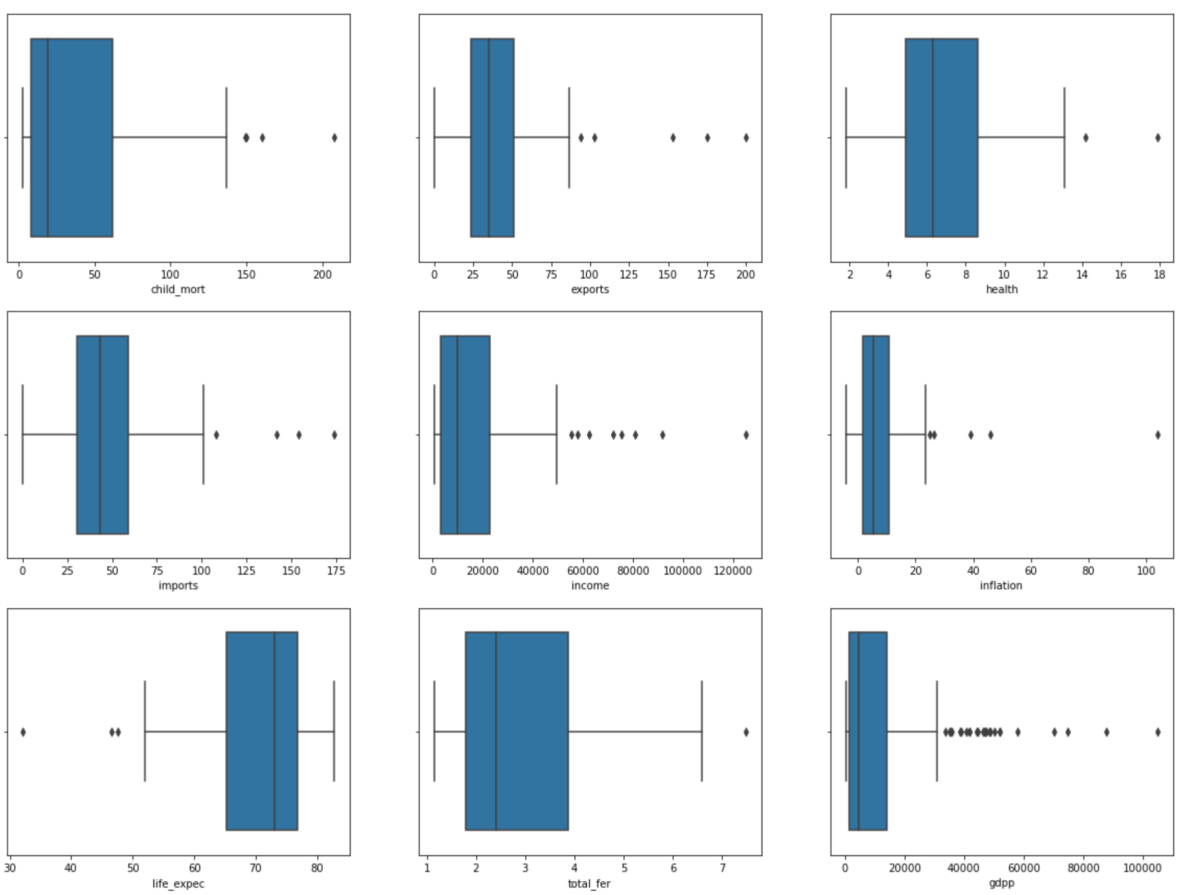
Out[6]:
```
country       0
child_mort    0
exports       0
health        0
imports       0
income        0
inflation     0
life_expec    0
total_fer     0
gdpp          0
dtype: int64
```

In [7]:
```
c_data.columns
```

Out[7]:
```
Index(['country', 'child_mort', 'exports', 'health', 'imports', 'income',
       'inflation', 'life_expec', 'total_fer', 'gdpp'],
      dtype='object')
```

In [8]:
```
# Checking for outliers
plt.figure(figsize=(20,20))
plt.subplot(4,3,1)
sns.boxplot(x = 'child_mort', data = c_data)
plt.subplot(4,3,2)
sns.boxplot(x = 'exports', data = c_data)
plt.subplot(4,3,3)
sns.boxplot(x = 'health', data = c_data)
plt.subplot(4,3,4)
sns.boxplot(x = 'imports', data = c_data)
plt.subplot(4,3,5)
sns.boxplot(x = 'income', data = c_data)
plt.subplot(4,3,6)
sns.boxplot(x = 'inflation', data = c_data)
plt.subplot(4,3,7)
sns.boxplot(x = 'life_expec', data = c_data)
plt.subplot(4,3,8)
sns.boxplot(x = 'total_fer', data = c_data)
plt.subplot(4,3,9)
sns.boxplot(x = 'gdpp', data = c_data)
```

Out[8]:
```
<AxesSubplot:xlabel='gdpp'>
```



In [9]:
```
# Checking the outliers using Z-Score
from scipy import stats
z = np.abs(stats.zscore(c_data[['child_mort', 'exports', 'health',
'imports', 'income',
 'inflation', 'life_expec', 'total_fer', 'gdpp']]))
print(z)
print('*******************************************************************')
print(np.where(z > 3))
```

```
     child_mort   exports    health   imports    income  inflation  \
0      1.291532  1.138280  0.279088  0.082455  0.808245   0.157336
1      0.538949  0.479658  0.097016  0.070837  0.375369   0.312347
2      0.272833  0.099122  0.966073  0.641762  0.220844   0.789274
3      2.007808  0.775381  1.448071  0.165315  0.585043   1.387054
4      0.695634  0.160668  0.286894  0.497568  0.101732   0.601749
..          ...       ...       ...       ...       ...        ...
162    0.225578  0.200917  0.571711  0.240700  0.738527   0.489784
163    0.526514  0.461363  0.695862  1.213499  0.033542   3.616865
164    0.372315  1.130305  0.008877  1.380030  0.658404   0.409732
165    0.448417  0.406478  0.597272  0.517472  0.658924   1.500916
166    1.114951  0.150348  0.338015  0.662477  0.721358   0.590015

     life_expec  total_fer      gdpp
0      1.619092   1.902882  0.679180
1      0.647866   0.859973  0.485623
2      0.670423   0.038404  0.465376
3      1.179234   2.128151  0.516268
4      0.704258   0.541946  0.041817
..          ...        ...       ...
162    0.852161   0.365754  0.546913
163    0.546361   0.316678  0.029323
164    0.286958   0.661206  0.637754
165    0.344633   1.140944  0.637754
166    2.092785   1.624609  0.629546

[167 rows x 9 columns]
*******************************************************************
(array([ 23,  66,  66,  82,  91,  91,  91,  91,  98,  98, 112, 113, 114,
       123, 123, 132, 133, 133, 145, 159, 163], dtype=int64), array([4, 0, 6,
4, 1, 3, 4, 8, 1, 3, 7, 5, 8, 4, 8, 0, 1, 3, 8, 2, 5],
      dtype=int64))
```

In [10]:
```
# Removing the outlier
c_data_outlier_removed = c_data[(z<3).all(axis=1)]
c_data_outlier_removed.head(10)
```

Out[10]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.440	56.2	5.82	55
1	Albania	16.6	28.0	6.55	48.6	9930	4.490	76.3	1.65	409
2	Algeria	27.3	38.4	4.17	31.4	12900	16.100	76.5	2.89	446
3	Angola	119.0	62.3	2.85	42.9	5900	22.400	60.1	6.16	353
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.440	76.8	2.13	1220
5	Argentina	14.5	18.9	8.10	16.0	18700	20.900	75.8	2.37	1030
6	Armenia	18.1	20.8	4.40	45.3	6700	7.770	73.3	1.69	322
7	Australia	4.8	19.8	8.73	20.9	41400	1.160	82.0	1.93	5190
8	Austria	4.3	51.3	11.00	47.8	43200	0.873	80.5	1.44	4690
9	Azerbaijan	39.2	54.3	5.88	20.7	16000	13.800	69.1	1.92	584

In [11]:
```
print('Shape of dataframe before outlier removal: ' +str(c_data.shape))
print('Shape of dataframe after outlier removal: ' +str(c_data_outlier_removed
x = c_data_outlier_removed.drop('country',axis=1)
y = c_data_outlier_removed['country']
```

```
Shape of dataframe before outlier removal: (167, 10)
Shape of dataframe after outlier removal: (153, 10)
```

In [12]:
```
x.shape
```

Out[12]: (153, 9)

In [13]:
```
y.shape
```

Out[13]: (153,)

In [14]:
```
# Principle Component Analaysis
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_scaled = sc.fit_transform(x)
x_scaled
```

Out[14]:
```
array([[ 1.46183636, -1.41330427,  0.31809414, ..., -1.73823548,
         1.94438462, -0.72205486],
       [-0.56911214, -0.52600184, -0.08875965, ...,  0.71229884,
        -0.88698624, -0.46758977],
       [-0.27385196, -0.01333821, -1.02886841, ...,  0.73668227,
        -0.04504383, -0.44097058],
       ...,
       [-0.3842296 ,  1.64295967,  0.02579142, ...,  0.32216403,
        -0.6832905 , -0.66759343],
       [ 0.5263859 , -0.42741268, -0.62991469, ..., -0.36057191,
         1.16355093, -0.66759343],
       [ 1.26591606, -0.08235062, -0.34946208, ..., -2.25028743,
         1.65921057, -0.65680187]])
```

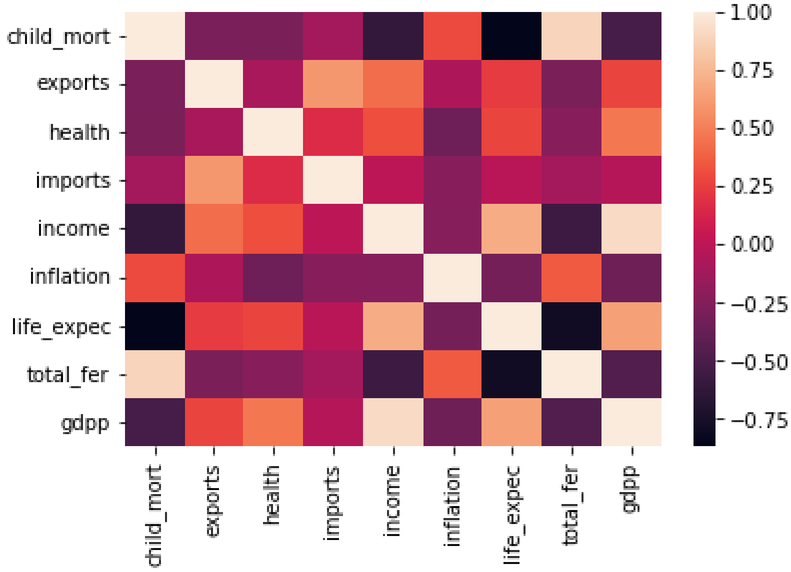
In [15]:
```
x_scaled_dataframe = pd.DataFrame(x_scaled,columns=x.columns)
x_scaled_dataframe.head()
```

Out[15]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	g
0	1.461836	-1.413304	0.318094	-0.043800	-0.954569	0.348785	-1.738235	1.944385	-0.722
1	-0.569112	-0.526002	-0.088760	0.150114	-0.331921	-0.365865	0.712299	-0.886986	-0.467
2	-0.273852	-0.013338	-1.028868	-0.751321	-0.109654	1.310315	0.736682	-0.045044	-0.440
3	2.256555	1.164802	-1.550273	-0.148618	-0.633516	2.219869	-1.262759	2.175240	-0.507
4	-0.742957	0.336653	-0.294162	0.689927	0.354339	-0.806205	0.773257	-0.561073	0.115

In [16]:
```
sns.heatmap(x_scaled_dataframe.corr())
```

Out[16]:
```
<AxesSubplot:>
```



In [17]:
```
from sklearn.decomposition import PCA
pca = PCA(random_state=42)
pca.fit(x_scaled)
PCA(random_state=42)
pca.components_[0]
```

Out[17]:
```
array([-0.42321972,  0.2036042 ,  0.21754201,  0.08290998,  0.41369318,
       -0.22650995,  0.42715413, -0.40550525,  0.39482635])
```

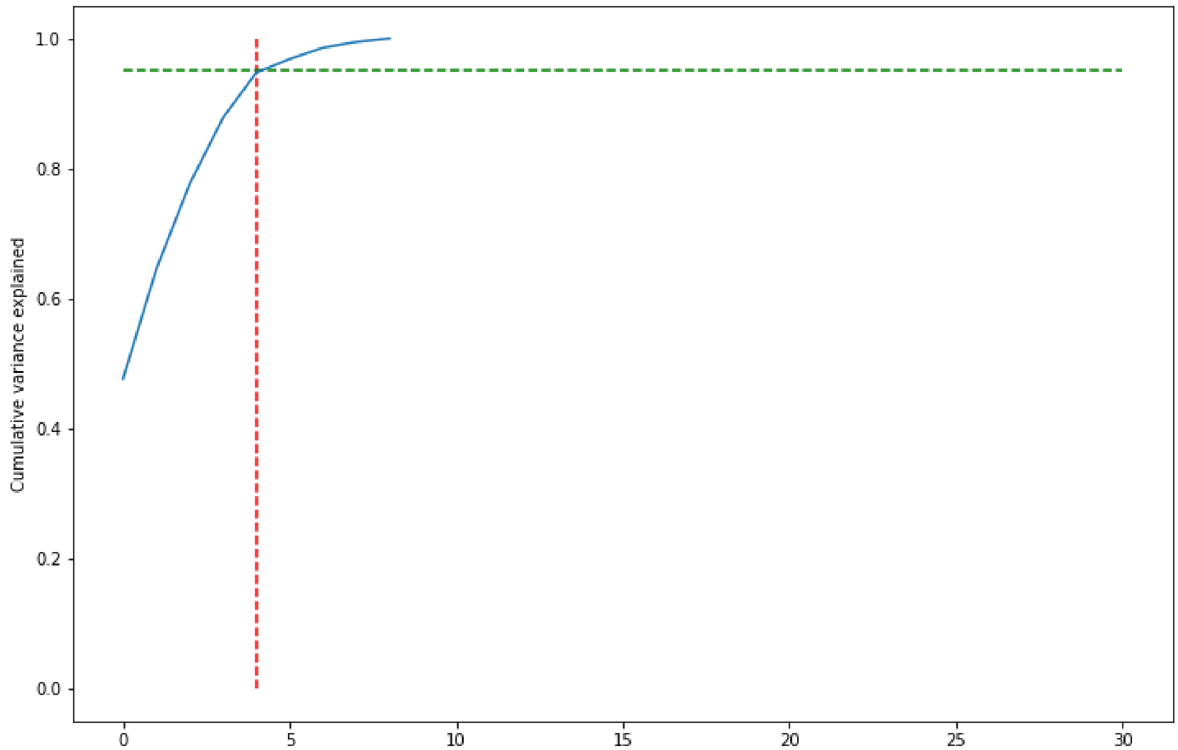
In [18]:
```
pca.explained_variance_ratio_
```

Out[18]:
```
array([0.47638387, 0.16902847, 0.13080614, 0.10179586, 0.06939066,
       0.02084938, 0.01747184, 0.00883956, 0.00543422])
```

In [19]:
```
var_cumsm = np.cumsum(pca.explained_variance_ratio_)
var_cumsm
```

Out[19]:
```
array([0.47638387, 0.64541234, 0.77621848, 0.87801434, 0.94740499,
       0.96825437, 0.98572622, 0.99456578, 1.        ])
```

In [20]:
```
fig = plt.figure(figsize=[12,8])
plt.vlines(x=4, ymax=1, ymin=0, colors="r", linestyles="--")
plt.hlines(y=0.95, xmax=30, xmin=0, colors="g", linestyles="--")
plt.plot(var_cumsm)
plt.ylabel("Cumulative variance explained")
plt.show()
```



In [21]:
```
# Performing PCA with 4 features
from sklearn.decomposition import IncrementalPCA
pca_end = IncrementalPCA(n_components=4)
pca_end = pca_end.fit_transform(x_scaled)
print(x.shape)
print(pca_end.shape)
```

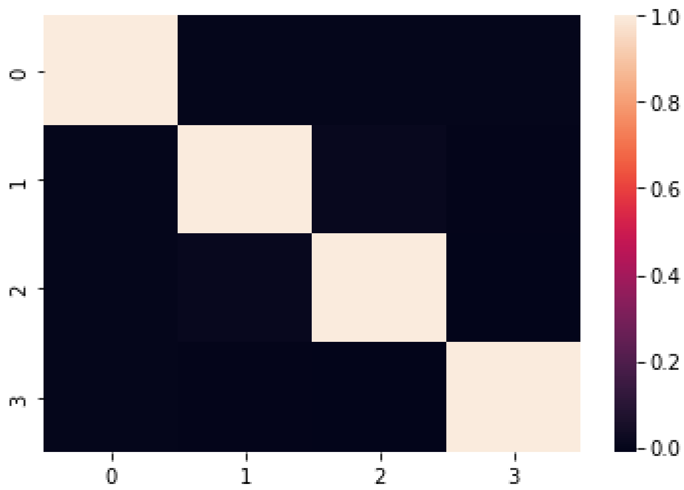
```
(153, 9)
(153, 4)
```

In [22]:
```
corr = np.corrcoef(pca_end.T)
corr.shape
```

Out[22]:
```
(4, 4)
```

In [23]:
```
sns.heatmap(corr)
```

Out[23]: <AxesSubplot:>

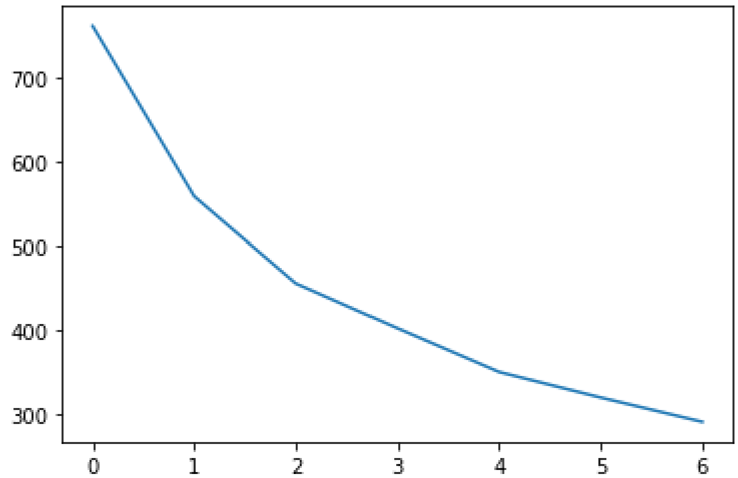


In [24]:
```
# Kmeans Clustering
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
kmeans = KMeans(n_clusters=5,max_iter=1000)
kmeans.fit(pca_end)
```

Out[24]: KMeans(max_iter=1000, n_clusters=5)

In [25]:
```
wcss = []
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
for num_clusters in range_n_clusters:
 kmeans = KMeans(n_clusters=num_clusters, max_iter=1000)
 kmeans.fit(pca_end)

 wcss.append(kmeans.inertia_)

plt.plot(wcss)
```

Out[25]:
```
[<matplotlib.lines.Line2D at 0x1ae59ed2dc0>]
```



In [ ]:

In [26]:
```
# Shiloute Analysis
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
for num_clusters in range_n_clusters:

 # intialise kmeans
 kmeans = KMeans(n_clusters=num_clusters, max_iter=1000)
 kmeans.fit(pca_end)

 cluster_labels = kmeans.labels_

 # silhouette score
 silhouette_avg = silhouette_score(pca_end, cluster_labels)
 print("For n_clusters={0}, the silhouette score is {1}".format(num_clusters,
```

```
For n_clusters=2, the silhouette score is 0.3185783255391956
For n_clusters=3, the silhouette score is 0.3214159977534033
For n_clusters=4, the silhouette score is 0.3069208111507679
For n_clusters=5, the silhouette score is 0.29395007295356534
For n_clusters=6, the silhouette score is 0.30772683635772596
For n_clusters=7, the silhouette score is 0.2955666989197241
For n_clusters=8, the silhouette score is 0.2883747810973742
```

In [27]:
```
kmeans = KMeans(n_clusters=4,max_iter=1000,random_state=42)
kmeans.fit(pca_end)
KMeans(max_iter=1000, n_clusters=4, random_state=42)
kmeans.labels_
```

Out[27]:
```
array([1, 3, 0, 1, 3, 0, 0, 2, 2, 0, 2, 2, 0, 3, 3, 2, 3, 1, 3, 0, 3, 1,
       0, 3, 1, 1, 3, 1, 2, 3, 1, 1, 0, 0, 0, 1, 1, 1, 3, 1, 3, 2, 3, 2,
       0, 0, 0, 3, 1, 1, 3, 3, 2, 2, 0, 1, 3, 2, 1, 2, 3, 0, 1, 1, 3, 3,
       2, 0, 0, 0, 0, 2, 2, 2, 0, 2, 3, 0, 1, 1, 3, 1, 3, 3, 1, 1, 0, 3,
       3, 1, 1, 3, 3, 1, 1, 3, 3, 3, 0, 3, 3, 1, 0, 1, 0, 2, 2, 0, 1, 3,
       3, 0, 0, 3, 2, 3, 0, 1, 3, 0, 1, 3, 3, 3, 2, 3, 1, 2, 2, 0, 3, 1,
       3, 2, 1, 1, 3, 1, 1, 3, 3, 0, 3, 1, 3, 2, 2, 0, 0, 3, 3, 1, 1])
```

In [28]:
```
c_data_outlier_removed['K-Means_Cluster_ID'] = kmeans.labels_
```

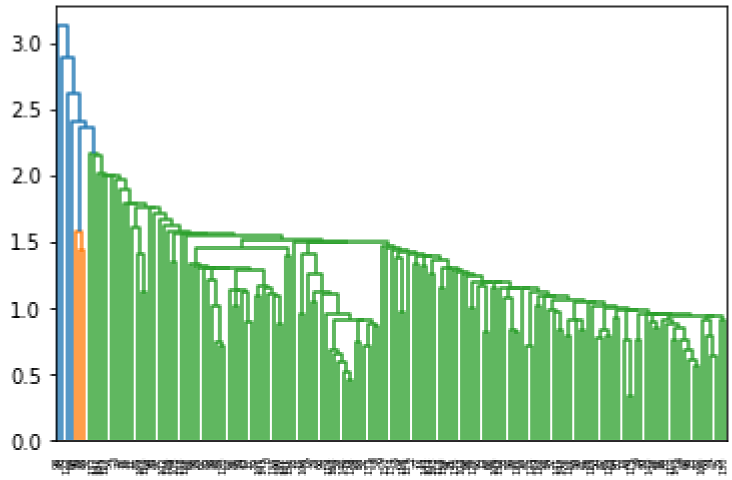
```
C:\Users\91830\AppData\Local\Temp\ipykernel_7892\2507830492.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  c_data_outlier_removed['K-Means_Cluster_ID'] = kmeans.labels_
```

In [29]:
```
# Hierarchical Clustering
x_scaled_dataframe.head()
```

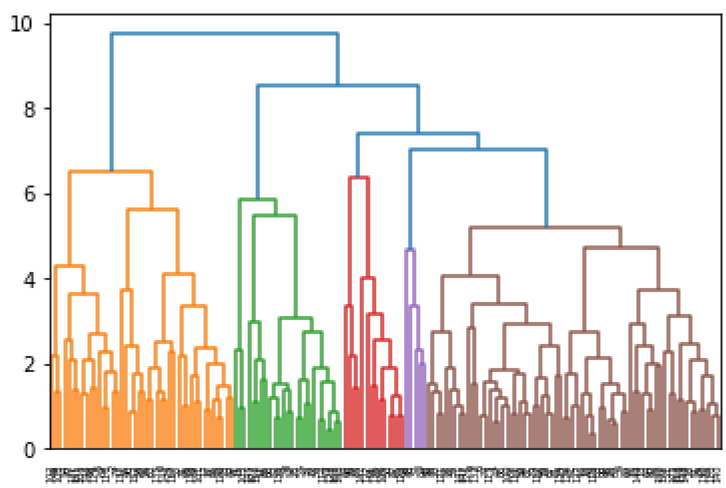
Out[29]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	g
0	1.461836	-1.413304	0.318094	-0.043800	-0.954569	0.348785	-1.738235	1.944385	-0.722
1	-0.569112	-0.526002	-0.088760	0.150114	-0.331921	-0.365865	0.712299	-0.886986	-0.467
2	-0.273852	-0.013338	-1.028868	-0.751321	-0.109654	1.310315	0.736682	-0.045044	-0.440
3	2.256555	1.164802	-1.550273	-0.148618	-0.633516	2.219869	-1.262759	2.175240	-0.507
4	-0.742957	0.336653	-0.294162	0.689927	0.354339	-0.806205	0.773257	-0.561073	0.115

In [30]:
```
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree
sl_mergings = linkage(x_scaled_dataframe, method="single",
metric='euclidean')
dendrogram(sl_mergings)
plt.show()
```



In [31]:
```
cl_mergings = linkage(x_scaled_dataframe, method="complete",
metric='euclidean')
dendrogram(cl_mergings)
plt.show()
```



In [32]:
```
sl_cluster_labels = cut_tree(sl_mergings, n_clusters=4).reshape(-1, )
sl_cluster_labels
```

Out[32]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [33]:
```
# single linkage doesnot perform well
cl_cluster_labels = cut_tree(cl_mergings, n_clusters=4).reshape(-1, )
cl_cluster_labels
```

Out[33]:
```
array([0, 1, 0, 1, 1, 0, 1, 2, 2, 1, 1, 2, 1, 1, 1, 2, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 0, 1, 0, 2, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 2,
       1, 1, 1, 1, 1, 0, 3, 1, 2, 2, 0, 0, 1, 2, 0, 2, 1, 1, 0, 0, 1, 3,
       2, 1, 0, 0, 0, 2, 2, 2, 1, 2, 1, 0, 0, 3, 1, 1, 1, 1, 3, 3, 2, 3,
       1, 0, 0, 3, 3, 0, 1, 1, 3, 1, 1, 1, 1, 0, 0, 1, 0, 2, 2, 2, 0, 3,
       1, 1, 1, 1, 2, 1, 0, 0, 1, 2, 0, 1, 3, 3, 1, 1, 1, 1, 2, 0, 1, 0,
       1, 2, 1, 0, 3, 0, 0, 1, 1, 1, 3, 0, 1, 2, 2, 1, 0, 1, 1, 0, 0])
```

In [34]:
```
c_data_outlier_removed["Hierarchical_Cluster_labels"] =cl_cluster_labels
```

```
C:\Users\91830\AppData\Local\Temp\ipykernel_7892\1597811289.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  c_data_outlier_removed["Hierarchical_Cluster_labels"] =cl_cluster_labels
```

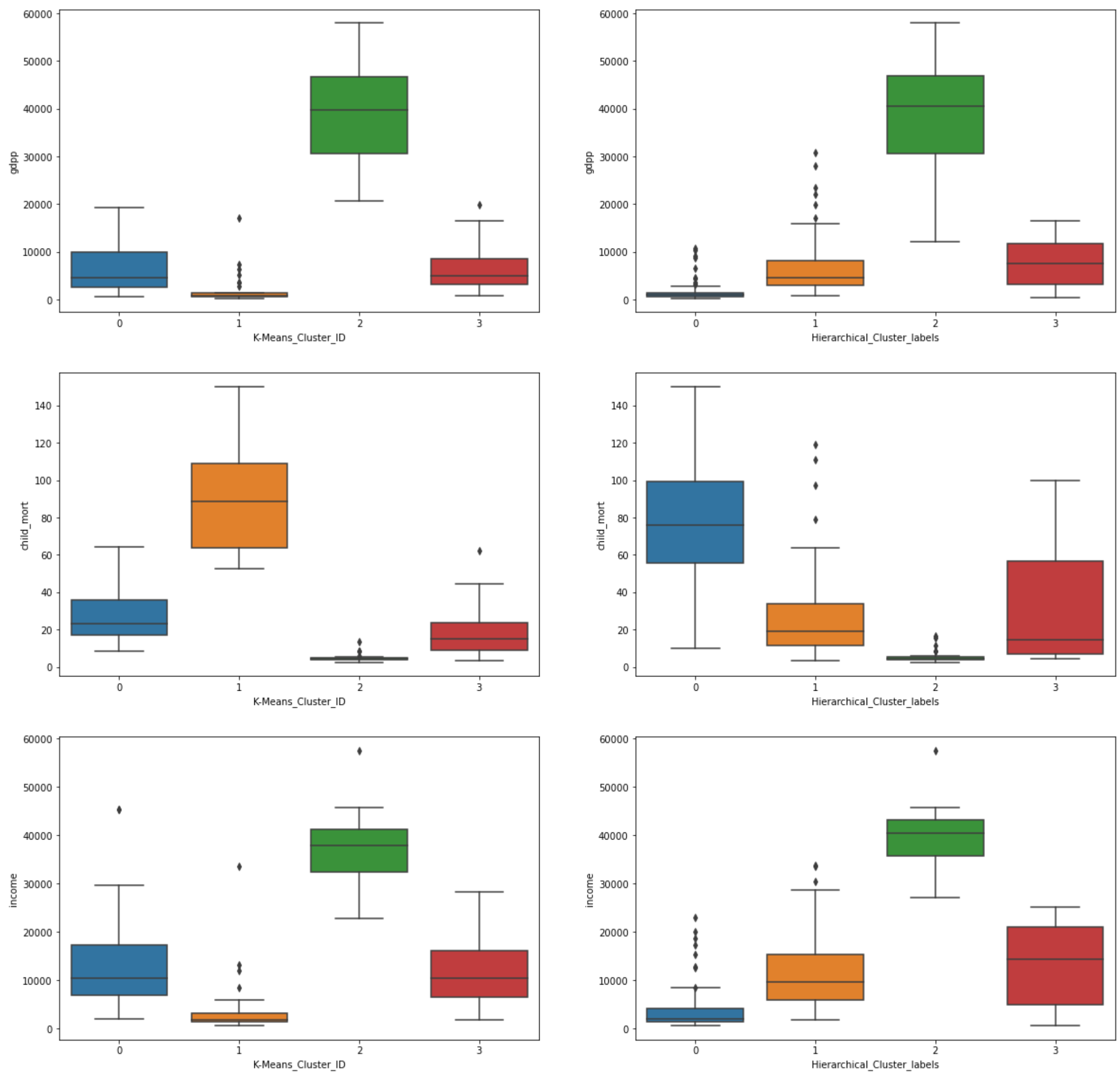
In [35]:
```
c_data_outlier_removed.head()
```

Out[35]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	55
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	409
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	446
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	353
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	1220

In [36]:
```
plt.figure(figsize=(20,20))
plt.subplot(3,2,1)
sns.boxplot(x='K-Means_Cluster_ID', y='gdpp',
data=c_data_outlier_removed)
plt.subplot(3,2,2)
sns.boxplot(x='Hierarchical_Cluster_labels', y='gdpp',
data=c_data_outlier_removed)
plt.subplot(3,2,3)
sns.boxplot(x='K-Means_Cluster_ID', y='child_mort',
data=c_data_outlier_removed)
plt.subplot(3,2,4)
sns.boxplot(x='Hierarchical_Cluster_labels', y='child_mort',
data=c_data_outlier_removed)
plt.subplot(3,2,5)
sns.boxplot(x='K-Means_Cluster_ID', y='income',
data=c_data_outlier_removed)
plt.subplot(3,2,6)
sns.boxplot(x='Hierarchical_Cluster_labels', y='income',
data=c_data_outlier_removed)
```

Out[36]:
```
<AxesSubplot:xlabel='Hierarchical_Cluster_labels', ylabel='income'>
```



In [37]:
```
X_pca_final_df =pd.DataFrame(pca_end,columns=['PC1','PC2','PC3','PC4'])
X_pca_final_df.head()
```

Out[37]:

	PC1	PC2	PC3	PC4
0	-3.129112	-0.530438	1.326366	0.592673
1	0.552498	-0.242770	-0.157737	-1.362826
2	-0.357008	-0.461483	-1.876976	-0.109599
3	-3.456355	1.213750	-1.381585	2.217845
4	1.308078	0.615244	-0.031004	-0.713291

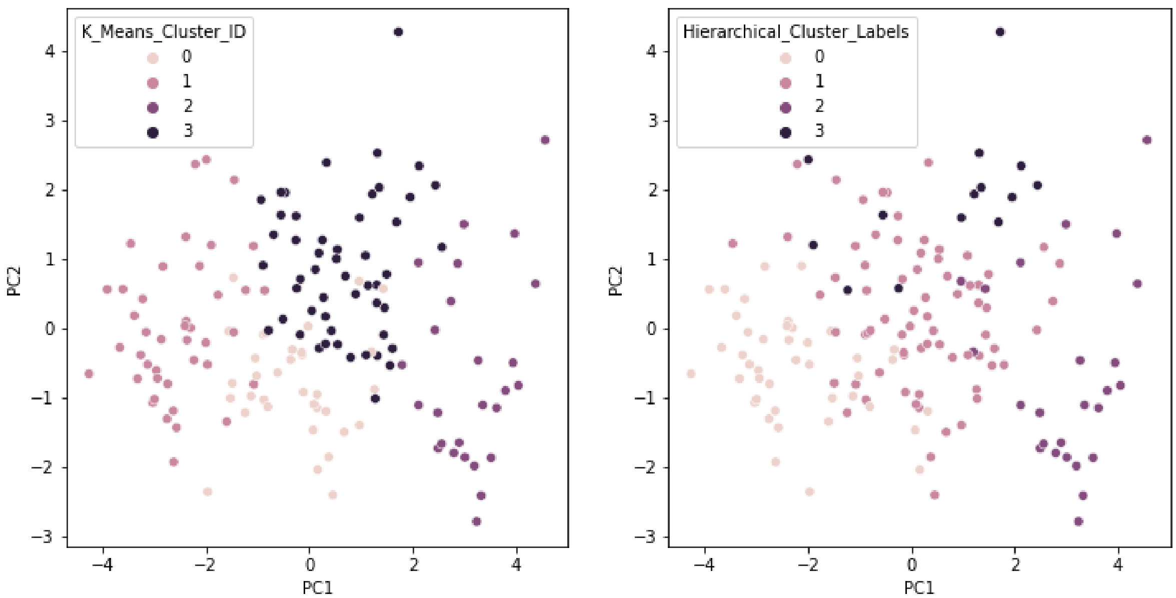
In [38]:
```
X_pca_final_df['K_Means_Cluster_ID'] = kmeans.labels_
X_pca_final_df['Hierarchical_Cluster_Labels'] = cl_cluster_labels
X_pca_final_df.head()
```

Out[38]:

	PC1	PC2	PC3	PC4	K_Means_Cluster_ID	Hierarchical_Cluster_Labels
0	-3.129112	-0.530438	1.326366	0.592673	1	0
1	0.552498	-0.242770	-0.157737	-1.362826	3	1
2	-0.357008	-0.461483	-1.876976	-0.109599	0	0
3	-3.456355	1.213750	-1.381585	2.217845	1	1
4	1.308078	0.615244	-0.031004	-0.713291	3	1

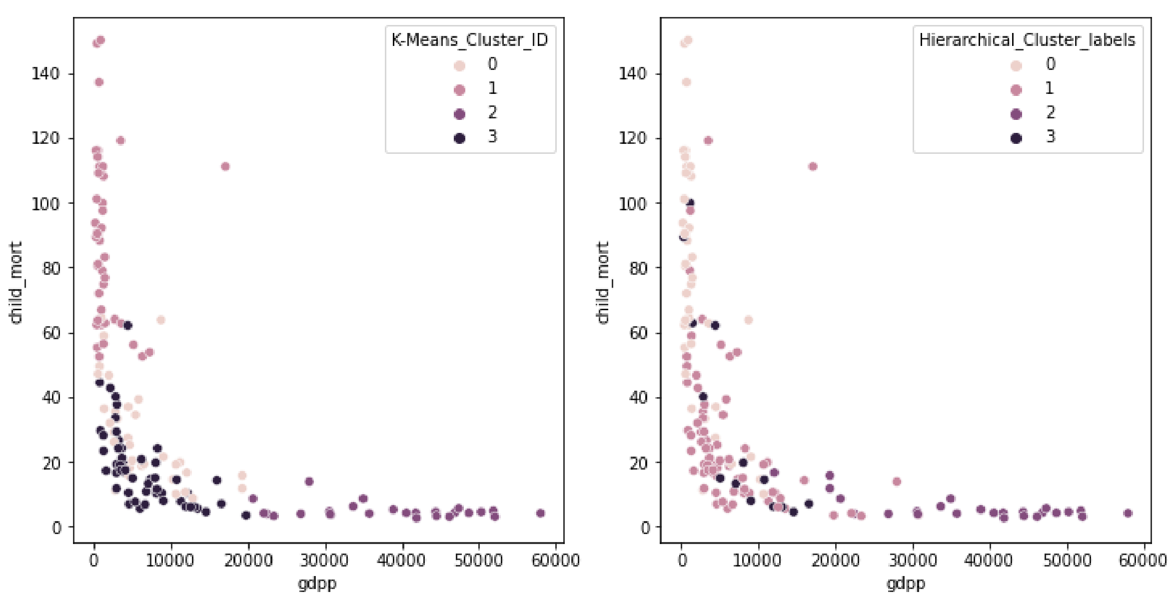
In [39]:
```
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
sns.scatterplot(x='PC1',y='PC2',data=X_pca_final_df,hue='K_Means_Cluster_ID')
plt.subplot(1,2,2)
sns.scatterplot(x='PC1',y='PC2',data=X_pca_final_df,hue='Hierarchical_Cluster_
```

Out[39]: <AxesSubplot:xlabel='PC1', ylabel='PC2'>



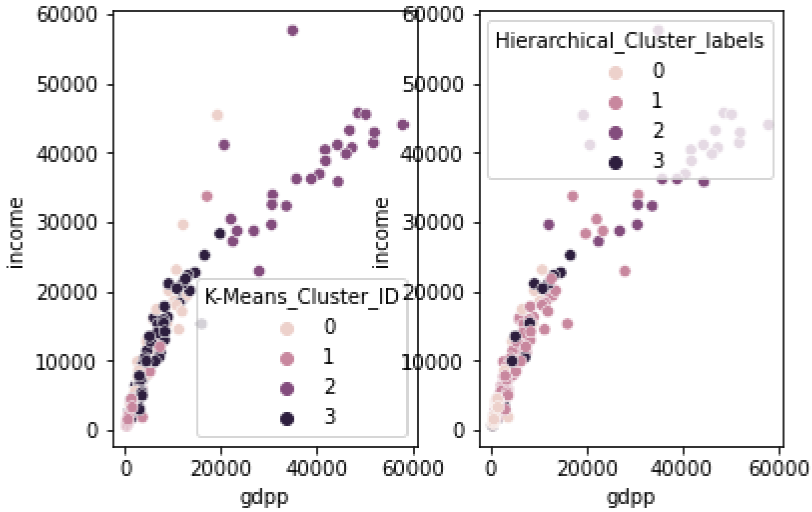
In [40]:
```
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
sns.scatterplot(x='gdpp',y='child_mort',data=c_data_outlier_removed,hue='K-Mea
plt.subplot(1,2,2)
sns.scatterplot(x='gdpp',y='child_mort',data=c_data_outlier_removed,hue='Hiera
```

Out[40]: <AxesSubplot:xlabel='gdpp', ylabel='child_mort'>



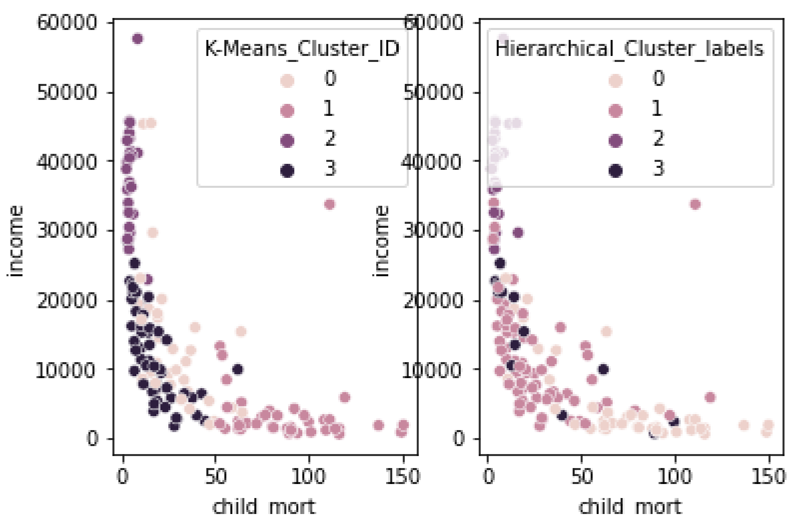
In [41]:
```
#Low gdpp corrsponds to low household income and hence higher child mortality
plt.subplot(1,2,1)
sns.scatterplot(x='gdpp',y='income',data=c_data_outlier_removed,hue='K-Means_C
plt.subplot(1,2,2)
sns.scatterplot(x='gdpp',y='income',data=c_data_outlier_removed,hue='Hierarchi
```

Out[41]: <AxesSubplot:xlabel='gdpp', ylabel='income'>



In [42]:
```
# We can observe a linear relationship between gdpp and income
plt.subplot(1,2,1)
sns.scatterplot(x='child_mort',y='income',data=c_data_outlier_removed,hue='K-M
plt.subplot(1,2,2)
sns.scatterplot(x='child_mort',y='income',data=c_data_outlier_removed,hue='Hie
```

Out[42]: <AxesSubplot:xlabel='child_mort', ylabel='income'>



In [43]:
```
K_Means_countries = c_data_outlier_removed[c_data_outlier_removed['K-Means_Clu
K_Means_countries
```

Out[43]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	
0	Afghanistan	90.2	10.00	7.58	44.9	1610	9.440	56.2	5.82	
3	Angola	119.0	62.30	2.85	42.9	5900	22.400	60.1	6.16	
17	Benin	111.0	23.80	4.10	37.2	1820	0.885	61.8	5.36	
21	Botswana	52.5	43.60	8.30	51.3	13300	8.920	57.1	2.88	
25	Burkina Faso	116.0	19.20	6.74	29.6	1430	6.810	57.9	5.87	
26	Burundi	93.6	8.92	11.60	39.2	764	12.300	57.7	6.26	
28	Cameroon	108.0	22.20	5.13	27.0	2660	1.910	57.3	5.11	
31	Central African Republic	149.0	11.80	3.98	26.5	888	2.010	47.5	5.21	
32	Chad	150.0	36.80	4.53	43.5	1930	6.390	56.5	6.59	
36	Comoros	88.2	16.50	4.51	51.7	1410	3.870	65.9	4.75	
37	Congo, Dem. Rep.	116.0	41.10	7.91	49.6	609	20.800	57.5	6.54	
38	Congo, Rep.	63.9	85.10	2.46	54.7	5190	20.700	60.4	4.95	
40	Cote d'Ivoire	111.0	50.60	5.30	43.3	2690	5.390	56.3	5.27	
49	Equatorial Guinea	111.0	85.80	4.48	58.9	33700	24.900	60.9	5.21	1
50	Eritrea	55.2	4.79	2.66	23.3	1420	11.600	61.7	4.61	
56	Gambia	80.3	23.80	5.69	42.7	1660	4.300	65.5	5.71	
59	Ghana	74.7	29.50	5.22	45.9	3060	16.600	62.2	4.27	
63	Guinea	109.0	30.30	4.93	43.2	1190	16.100	58.0	5.34	
64	Guinea-Bissau	114.0	14.90	8.50	35.2	1390	2.970	55.6	5.05	
80	Kenya	62.2	20.70	4.75	33.6	2480	2.090	62.8	4.37	
81	Kiribati	62.7	13.30	11.30	79.9	1730	1.520	60.7	3.84	
84	Lao	78.9	35.40	4.47	49.3	3980	9.200	63.8	3.15	
87	Lesotho	99.7	39.40	11.10	101.0	2380	4.150	46.5	3.30	
88	Liberia	89.3	19.10	11.80	92.6	700	5.470	60.8	5.02	
93	Madagascar	62.2	25.00	3.77	43.0	1390	8.790	60.8	4.60	
94	Malawi	90.5	22.80	6.59	34.9	1030	12.100	53.1	5.31	
97	Mali	137.0	22.80	4.98	35.1	1870	4.370	59.5	6.55	
99	Mauritania	97.4	50.70	4.41	61.2	3320	18.900	68.2	4.98	
106	Mozambique	101.0	31.50	5.21	46.2	918	7.640	54.5	5.56	
108	Namibia	56.0	47.80	6.78	60.7	8460	3.560	58.6	3.60	
116	Pakistan	92.1	13.50	2.20	19.4	4280	10.900	65.3	3.85	
126	Rwanda	63.6	12.00	10.50	30.0	1350	2.610	64.6	4.51	

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer
129	Senegal	66.8	24.90	5.66	40.3	2180	1.850	64.0	5.06
137	South Africa	53.7	28.60	8.94	27.4	12000	6.350	54.3	2.59
142	Sudan	76.7	19.70	6.32	17.2	3370	19.600	66.3	4.88
146	Tajikistan	52.4	14.90	5.98	58.6	2110	12.500	69.6	3.51
147	Tanzania	71.9	18.70	6.01	29.1	2090	9.250	59.3	5.43
149	Timor-Leste	62.6	2.20	9.12	27.8	1850	26.500	71.1	6.23
150	Togo	90.3	40.20	7.65	57.3	1210	1.180	58.7	4.87
155	Uganda	81.0	17.10	9.01	28.6	1540	10.600	56.8	6.15
165	Yemen	56.3	30.00	5.18	34.4	4480	23.600	67.5	4.67
166	Zambia	83.1	37.00	5.89	30.9	3280	14.000	52.0	5.40

In [44]:
```
Hirarchical_countries =c_data_outlier_removed[c_data_outlier_removed['Hierarch
Hirarchical_countries
```

Out[44]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	g
0	Afghanistan	90.2	10.000	7.58	44.9000	1610	9.440	56.2	5.82	
2	Algeria	27.3	38.400	4.17	31.4000	12900	16.100	76.5	2.89	
5	Argentina	14.5	18.900	8.10	16.0000	18700	20.900	75.8	2.37	1
17	Benin	111.0	23.800	4.10	37.2000	1820	0.885	61.8	5.36	
25	Burkina Faso	116.0	19.200	6.74	29.6000	1430	6.810	57.9	5.87	
26	Burundi	93.6	8.920	11.60	39.2000	764	12.300	57.7	6.26	
28	Cameroon	108.0	22.200	5.13	27.0000	2660	1.910	57.3	5.11	
31	Central African Republic	149.0	11.800	3.98	26.5000	888	2.010	47.5	5.21	
32	Chad	150.0	36.800	4.53	43.5000	1930	6.390	56.5	6.59	
36	Comoros	88.2	16.500	4.51	51.7000	1410	3.870	65.9	4.75	
37	Congo, Dem. Rep.	116.0	41.100	7.91	49.6000	609	20.800	57.5	6.54	
40	Cote d'Ivoire	111.0	50.600	5.30	43.3000	2690	5.390	56.3	5.27	
50	Eritrea	55.2	4.790	2.66	23.3000	1420	11.600	61.7	4.61	
55	Gabon	63.7	57.700	3.50	18.9000	15400	16.600	62.9	4.08	
56	Gambia	80.3	23.800	5.69	42.7000	1660	4.300	65.5	5.71	
59	Ghana	74.7	29.500	5.22	45.9000	3060	16.600	62.2	4.27	
63	Guinea	109.0	30.300	4.93	43.2000	1190	16.100	58.0	5.34	
64	Guinea-Bissau	114.0	14.900	8.50	35.2000	1390	2.970	55.6	5.05	
70	Indonesia	33.3	24.300	2.61	22.4000	8430	15.300	69.9	2.48	
71	Iran	19.3	24.400	5.60	19.4000	17400	15.900	74.5	1.76	
72	Iraq	36.9	39.400	8.41	34.1000	12700	16.600	67.2	4.56	
79	Kazakhstan	21.5	44.200	4.29	29.9000	20100	19.500	68.4	2.60	
80	Kenya	62.2	20.700	4.75	33.6000	2480	2.090	62.8	4.37	
93	Madagascar	62.2	25.000	3.77	43.0000	1390	8.790	60.8	4.60	
94	Malawi	90.5	22.800	6.59	34.9000	1030	12.100	53.1	5.31	
97	Mali	137.0	22.800	4.98	35.1000	1870	4.370	59.5	6.55	
106	Mozambique	101.0	31.500	5.21	46.2000	918	7.640	54.5	5.56	
107	Myanmar	64.4	0.109	1.97	0.0659	3720	7.040	66.8	2.41	
109	Nepal	47.0	9.580	5.25	36.4000	1990	15.100	68.3	2.61	
116	Pakistan	92.1	13.500	2.20	19.4000	4280	10.900	65.3	3.85	
125	Russia	10.0	29.200	5.08	21.1000	23100	14.200	69.2	1.57	1
126	Rwanda	63.6	12.000	10.50	30.0000	1350	2.610	64.6	4.51	

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer
129	Senegal	66.8	24.900	5.66	40.3000	2180	1.850	64.0	5.06
140	Sri Lanka	11.2	19.600	2.94	26.8000	8560	22.800	74.4	2.20
142	Sudan	76.7	19.700	6.32	17.2000	3370	19.600	66.3	4.88
147	Tanzania	71.9	18.700	6.01	29.1000	2090	9.250	59.3	5.43
149	Timor-Leste	62.6	2.200	9.12	27.8000	1850	26.500	71.1	6.23
150	Togo	90.3	40.200	7.65	57.3000	1210	1.180	58.7	4.87
155	Uganda	81.0	17.100	9.01	28.6000	1540	10.600	56.8	6.15
161	Uzbekistan	36.3	31.700	5.81	28.5000	4240	16.500	68.8	2.34
165	Yemen	56.3	30.000	5.18	34.4000	4480	23.600	67.5	4.67
166	Zambia	83.1	37.000	5.89	30.9000	3280	14.000	52.0	5.40

In [45]:
```
common_countries =pd.merge(K_Means_countries,Hirarchical_countries,how='inner'
 'inflation', 'life_expec', 'total_fer', 'gdpp', 'K-Means_Cluster_ID',
 'Hierarchical_Cluster_labels'])
```

In [46]:
```
common_countries.columns
```

Out[46]:
```
Index(['country', 'child_mort', 'exports', 'health', 'imports', 'income',
       'inflation', 'life_expec', 'total_fer', 'gdpp', 'K-Means_Cluster_ID',
       'Hierarchical_Cluster_labels'],
      dtype='object')
```

In [47]:
```
common_countries[['country', 'child_mort', 'income','gdpp']]
```

Out[47]:

	country	child_mort	income	gdpp
0	Afghanistan	90.2	1610	553
1	Benin	111.0	1820	758
2	Burkina Faso	116.0	1430	575
3	Burundi	93.6	764	231
4	Cameroon	108.0	2660	1310
5	Central African Republic	149.0	888	446
6	Chad	150.0	1930	897
7	Comoros	88.2	1410	769
8	Congo, Dem. Rep.	116.0	609	334
9	Cote d'Ivoire	111.0	2690	1220
10	Eritrea	55.2	1420	482
11	Gambia	80.3	1660	562
12	Ghana	74.7	3060	1310
13	Guinea	109.0	1190	648
14	Guinea-Bissau	114.0	1390	547
15	Kenya	62.2	2480	967
16	Madagascar	62.2	1390	413
17	Malawi	90.5	1030	459
18	Mali	137.0	1870	708
19	Mozambique	101.0	918	419
20	Pakistan	92.1	4280	1040
21	Rwanda	63.6	1350	563
22	Senegal	66.8	2180	1000
23	Sudan	76.7	3370	1480
24	Tanzania	71.9	2090	702
25	Timor-Leste	62.6	1850	3600
26	Togo	90.3	1210	488
27	Uganda	81.0	1540	595
28	Yemen	56.3	4480	1310
29	Zambia	83.1	3280	1460

In [50]:
```
## dataframe with dereasing child mortality rate and increasing income
common_countries_final = common_countries[['country',
'child_mort','income','gdpp']].sort_values(['child_mort','income'],ascending=[
common_countries_final
```

Out[50]:

	country	child_mort	income	gdpp
6	Chad	150.0	1930	897
5	Central African Republic	149.0	888	446
18	Mali	137.0	1870	708
8	Congo, Dem. Rep.	116.0	609	334
2	Burkina Faso	116.0	1430	575
14	Guinea-Bissau	114.0	1390	547
1	Benin	111.0	1820	758
9	Cote d'Ivoire	111.0	2690	1220
13	Guinea	109.0	1190	648
4	Cameroon	108.0	2660	1310
19	Mozambique	101.0	918	419
3	Burundi	93.6	764	231
20	Pakistan	92.1	4280	1040
17	Malawi	90.5	1030	459
26	Togo	90.3	1210	488
0	Afghanistan	90.2	1610	553
7	Comoros	88.2	1410	769
29	Zambia	83.1	3280	1460
27	Uganda	81.0	1540	595
11	Gambia	80.3	1660	562
23	Sudan	76.7	3370	1480
12	Ghana	74.7	3060	1310
24	Tanzania	71.9	2090	702
22	Senegal	66.8	2180	1000
21	Rwanda	63.6	1350	563
25	Timor-Leste	62.6	1850	3600
16	Madagascar	62.2	1390	413
15	Kenya	62.2	2480	967
28	Yemen	56.3	4480	1310
10	Eritrea	55.2	1420	482

In [51]:
```
#Countries with direst need for aid can be selected from the above dataframe
final_countries =common_countries_final[(common_countries_final['child_mort']
(common_countries_final['income'] < 1200)]
final_countries = final_countries.reset_index(drop=True)
final_countries
```

Out[51]:

	country	child_mort	income	gdpp
0	Central African Republic	149.0	888	446
1	Congo, Dem. Rep.	116.0	609	334
2	Guinea	109.0	1190	648
3	Mozambique	101.0	918	419
4	Burundi	93.6	764	231
5	Malawi	90.5	1030	459

In [ ]:
```
#Countries that are in direst need for aid 1.Central African Republic 2.Congo,
#3.Mozambique 4.Burundi 5.Malawi 6.Guinea
```

In [ ]:

In [ ]:

In [ ]: