In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
# For Suppressing warnings
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
# For Hopkins Statistics
from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
from math import isnan
```

In [4]:
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
```

In [5]:
```
# For K Means
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

In [6]:
```
# For Hierarchical Clustering
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree

plt.style.use("ggplot")
```

In [8]:
```
dataset=pd.read_csv(R'C:\Users\admin\Desktop\projectDataset\Country-data.csv')
```

In [9]:
```
dataset
```

Out[9]:

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

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
162	Vanuatu	29.2	46.6	5.25	52.7	2950	2.62	63.0	3.50	2970
163	Venezuela	17.1	28.5	4.91	17.6	16500	45.90	75.4	2.47	13500
164	Vietnam	23.3	72.0	6.84	80.2	4490	12.10	73.1	1.95	1310
165	Yemen	56.3	30.0	5.18	34.4	4480	23.60	67.5	4.67	1310
166	Zambia	83.1	37.0	5.89	30.9	3280	14.00	52.0	5.40	1460

167 rows × 10 columns

In [11]:

```
dataset.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 [13]:

```
#Comment
#There is only 1 categorical column here, i.e. 'country'
#Rest all columns are numeric
#There is no missing values in the data, hence missing value handling is not required.
#All columns have correct datatypes, hence type casting is not required.
#'exports', 'health', 'imports' are given in percentage of gdpp. This features would be

dataset[['exports','health','imports']] = dataset[['exports','health','imports']].apply
dataset.head()
```

Out[13]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	55.30	41.9174	248.297	1610	9.44	56.2	5.82	553
1	Albania	16.6	1145.20	267.8950	1987.740	9930	4.49	76.3	1.65	4090
2	Algeria	27.3	1712.64	185.9820	1400.440	12900	16.10	76.5	2.89	4460
3	Angola	119.0	2199.19	100.6050	1514.370	5900	22.40	60.1	6.16	3530
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100	1.44	76.8	2.13	12200

In [14]:

```
dataset.describe()
```

Out[14]:

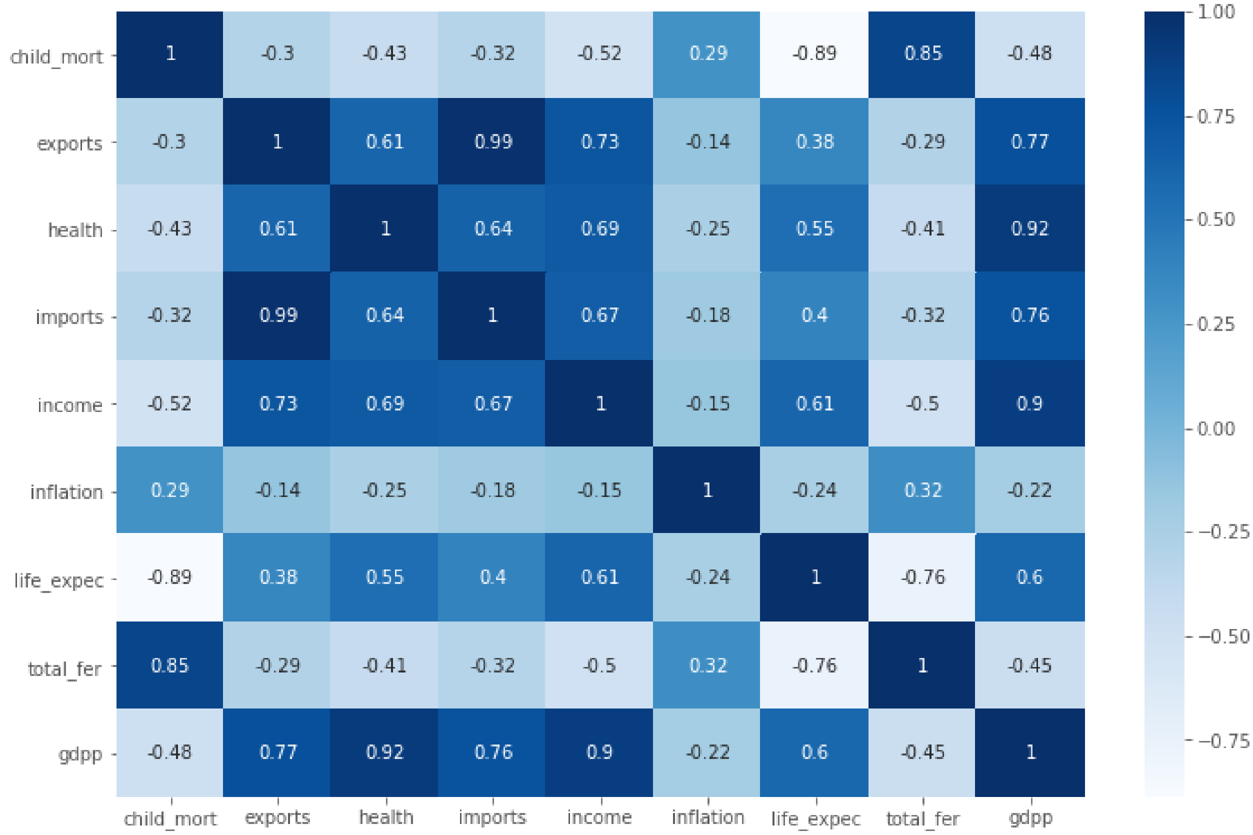
	child_mort	exports	health	imports	income	inflation	life_expec
count	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000
mean	38.270060	7420.618847	1056.733204	6588.352108	17144.688623	7.781832	70.555689
std	40.328931	17973.885795	1801.408906	14710.810418	19278.067698	10.570704	8.893172
min	2.600000	1.076920	12.821200	0.651092	609.000000	-4.210000	32.100000
25%	8.250000	447.140000	78.535500	640.215000	3355.000000	1.810000	65.300000
50%	19.300000	1777.440000	321.886000	2045.580000	9960.000000	5.390000	73.100000
75%	62.100000	7278.000000	976.940000	7719.600000	22800.000000	10.750000	76.800000
max	208.000000	183750.000000	8663.600000	149100.000000	125000.000000	104.000000	82.800000

In [16]:

```
#data visualization
# Bivariate Analysis

sns.pairplot(dataset)
plt.show()
```

In [17]:
```
#Comment
#gdpp is linearly related with exports, health, imports, income. (positively correlated
#child_mort is negatively correlated with life_expec (greater the child mortality, less
```

In [18]:
```
plt.figure(figsize = (12,8))
sns.heatmap(dataset.corr(),annot = True, cmap='Blues')
plt.show()
```

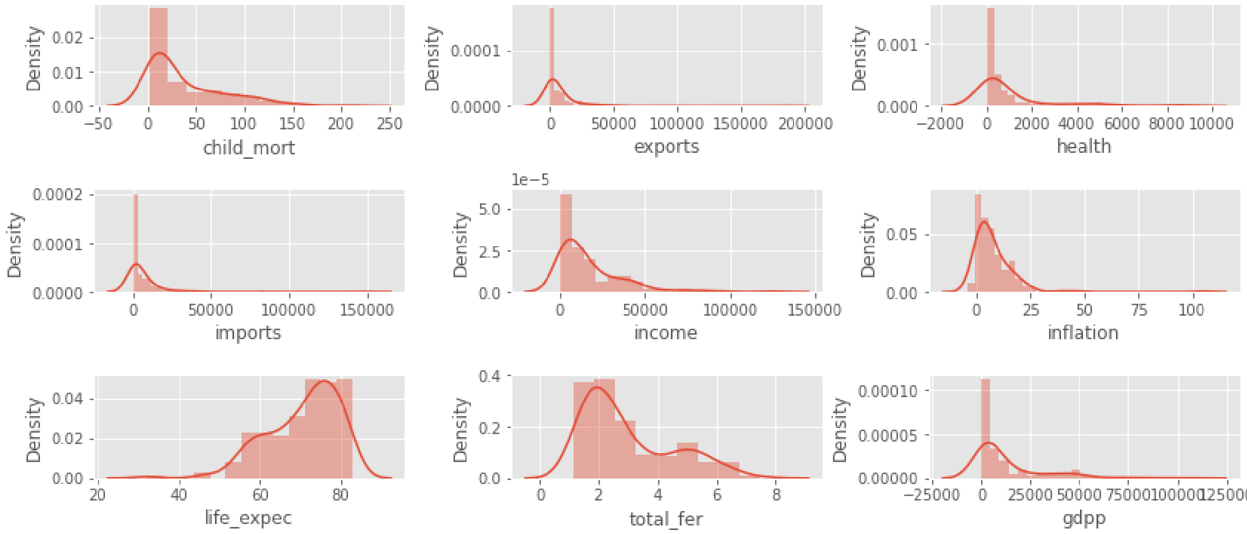


In [19]:

```
#Comment:

#Following feature pairs are highly correlated (positively or negatively)
#imports and exports (correlation factor = 0.99)
#health and gdpp (correlation factor = 0.92)
#income and gdpp (correlation factor = 0.9)
#life_expce and child_mort (correlation factor = -0.89)
#total_fer and child_mort (correlation factor = 0.85)
```

In [20]:

```
# univariate analysis

fig=plt.subplots(figsize=(12, 10))

for i, feature in enumerate(dataset.drop('country', axis=1).columns):
    plt.subplot(6, 3, i+1)
    plt.subplots_adjust(hspace = 2.0)
    sns.distplot(dataset[feature])
    plt.tight_layout()
```

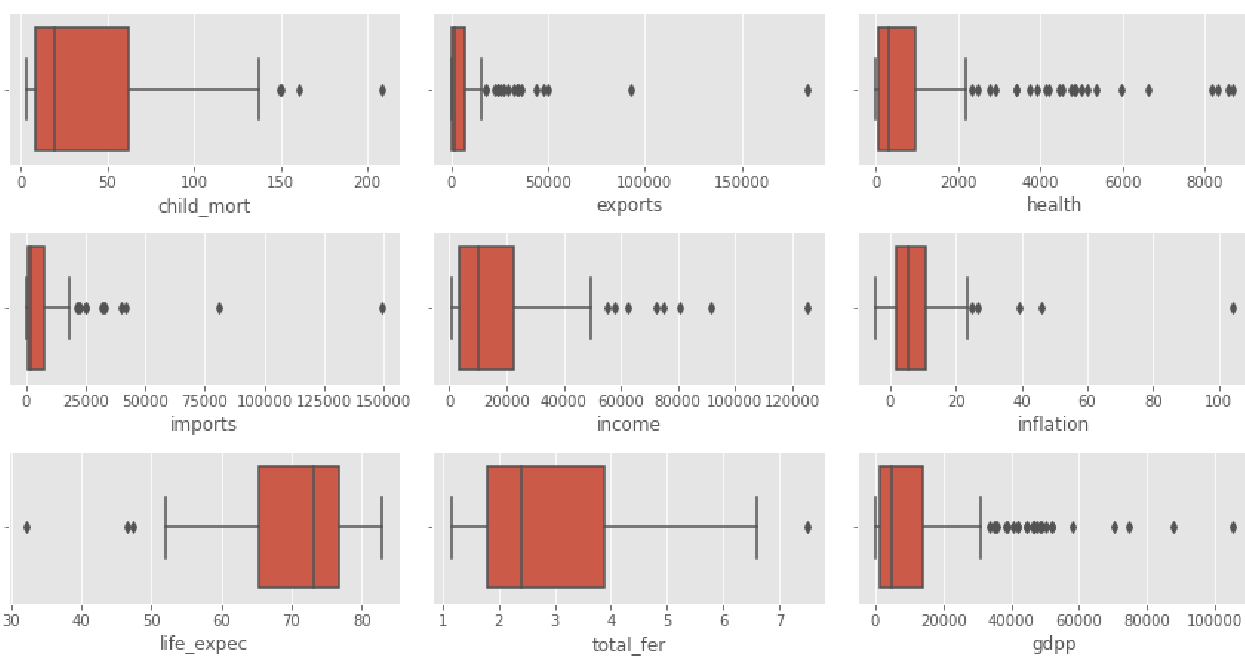


In [21]:

```
#Comment:
#Except Life expectancy (life_expec) all the features are right-skewed.
```

In [22]:

```
#Handling Outliers
fig=plt.subplots(figsize=(12, 12))

for i, feature in enumerate(dataset.drop('country', axis=1).columns):
    plt.subplot(6, 3, i+1)
    plt.subplots_adjust(hspace = 2.0)
    sns.boxplot(dataset[feature])
    plt.tight_layout()
```



In [23]:

```
#Comment
#Outliers for features like 'child_mort', 'inflation', 'life_expec', 'total_fer' are at
#Outliers for exports, imports, health, income, gdpp features are mostly developed coun
#Since there are so many outliers in the dataset, removing the outliers would mean insu
#where, Q1 = 25th percentile, Q3 = 75th percentile and IQR = (Q3 - Q1)
#The new dataframe after outlier treatment will be 'country_df_updated'
```

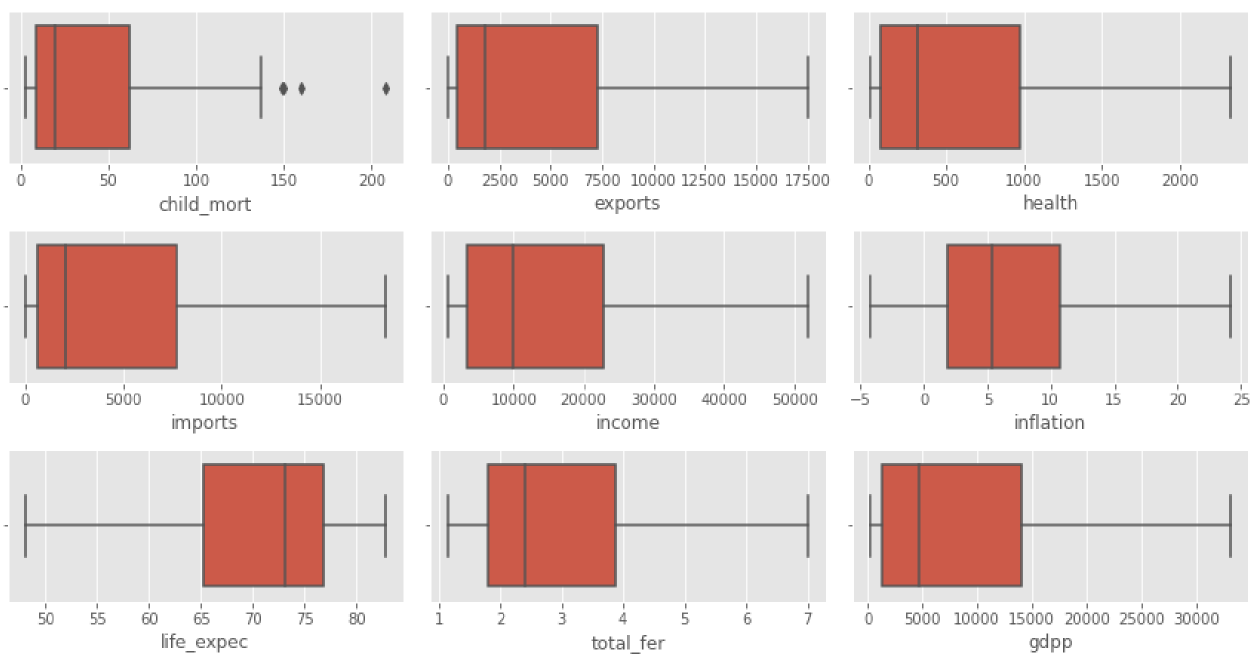
```
#For child_mort, outliers are in higher values, not capping this feature as this featur
#gdpp and income do not have outliers in lower range- they only have outliers in the hi
#Rest other features will be capped and floored as the outliers represent the developed
```

In [24]:

```
dataset_updated = dataset.iloc[:,:]

def outliers_for_features(dataset, col):
    Q1 = dataset.loc[:,col].quantile(0.25)
    Q3 = dataset.loc[:,col].quantile(0.75)

    upper_limit = Q3 + 1.5*(Q3-Q1)
    lower_limit = Q1 - 1.5*(Q3-Q1)

    return dataset_updated[col].apply(lambda x : upper_limit if x > upper_limit else lo

for col in ['life_expec','inflation', 'total_fer', 'exports', 'imports', 'health', 'inc
    dataset_updated[col] = outliers_for_features(dataset, col)
```

In [25]:

```
# Checking the distribution after flooring and capping

fig=plt.subplots(figsize=(12, 12))

for i, feature in enumerate(dataset_updated.drop('country', axis=1).columns):
    plt.subplot(6, 3, i+1)
    plt.subplots_adjust(hspace = 2.0)
    sns.boxplot(dataset_updated[feature])
    plt.tight_layout()
```



In [26]:

```
dataset_updated
```

Out[26]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gd
0	Afghanistan	90.2	55.30	41.9174	248.297	1610.0	9.44	56.2	5.82	55

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gd
1	Albania	16.6	1145.20	267.8950	1987.740	9930.0	4.49	76.3	1.65	409(
2	Algeria	27.3	1712.64	185.9820	1400.440	12900.0	16.10	76.5	2.89	446(
3	Angola	119.0	2199.19	100.6050	1514.370	5900.0	22.40	60.1	6.16	353(
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100.0	1.44	76.8	2.13	1220(
...	...	...	...	...	...	...	...	...	...	
162	Vanuatu	29.2	1384.02	155.9250	1565.190	2950.0	2.62	63.0	3.50	297(
163	Venezuela	17.1	3847.50	662.8500	2376.000	16500.0	24.16	75.4	2.47	1350(
164	Vietnam	23.3	943.20	89.6040	1050.620	4490.0	12.10	73.1	1.95	131(
165	Yemen	56.3	393.00	67.8580	450.640	4480.0	23.60	67.5	4.67	131(
166	Zambia	83.1	540.20	85.9940	451.140	3280.0	14.00	52.0	5.40	146(

167 rows × 10 columns

In [27]:

```
#Checking Suitability of dataset for clustering, Hopkins test

def hopkins(X):
    d = X.shape[1]
    #d = len(vars) # columns
    n = len(X) # rows
    m = int(0.1 * n)
    nbrs = NearestNeighbors(n_neighbors=1).fit(X.values)

    rand_X = sample(range(0, n, 1), m)

    ujd = []
    wjd = []
    for j in range(0, m):
        u_dist, _ = nbrs.kneighbors(uniform(np.amin(X,axis=0),np.amax(X,axis=0),d).resh
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors(X.iloc[rand_X[j]].values.reshape(1, -1), 2, return_
        wjd.append(w_dist[0][1])

    H = sum(ujd) / (sum(ujd) + sum(wjd))
    if isnan(H):
        print(ujd, wjd)
        H = 0

    return H
```

In [28]:

```
hopkins(dataset.drop('country', axis = 1))
```

Out[28]:

```
0.8996060478158507
```

In [29]:

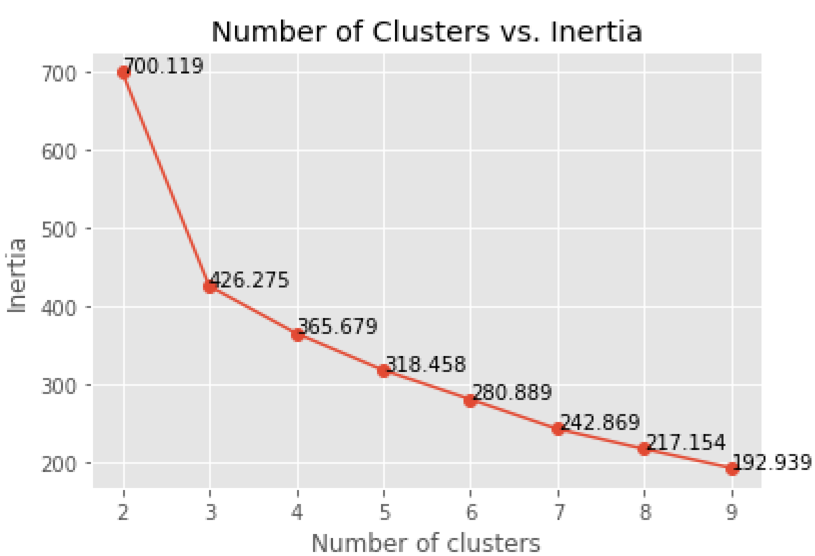
```
hopkins(dataset_updated.drop('country', axis = 1))
```

Out[29]: 0.8930517061181519

In [30]:
```
#Comment:

#High value of Hopkins Statistics implements that dataset has high tendency to cluster
```

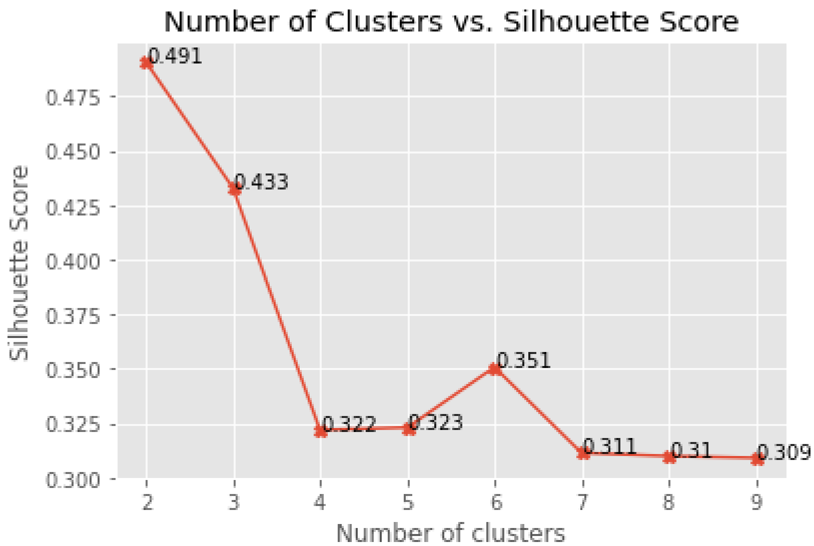
In [31]:
```
#Scaling the data

standard_scaler = StandardScaler()
dataset_scaled = standard_scaler.fit_transform(dataset_updated.iloc[:, 1:])
```

In [32]:
```
#Clustering
#K means
#Choosing k-value for K means algorithm

ssd = []
num_of_clusters = list(range(2,10))

for n in num_of_clusters:
    km = KMeans(n_clusters = n, max_iter = 50, random_state=101).fit(dataset_scaled)
    ssd.append(km.inertia_)

plt.plot(num_of_clusters, ssd, marker='o')

for xy in zip(num_of_clusters, ssd):
    plt.annotate(s = round(xy[1],3), xy = xy,  textcoords='data')

plt.xlabel("Number of clusters")
plt.ylabel("Inertia") # Inertia is within cluster sum of squares
plt.title("Number of Clusters vs. Inertia")
plt.show()
```



In [33]:
```
silhouette_value = []
for n in range(2,10):
    km = KMeans(n_clusters = n, random_state=101).fit(dataset_scaled)
    silhouette_value.append(silhouette_score(dataset_scaled, km.labels_))
```

```
plt.plot(num_of_clusters, silhouette_value, marker='X', label=silhouette_value)
plt.xlabel("Number of clusters")
plt.ylabel("Silhouette Score")
plt.title("Number of Clusters vs. Silhouette Score")

for xy in zip(num_of_clusters, silhouette_value):
    plt.annotate(s = round(xy[1],3), xy = xy,  textcoords='data')

plt.show()
```



In [34]:

```
#Comment: K=2 has highest Silhouette score but it will make more sense to take more tha
#Hence k=3 is closen.
```

In [36]:

```
# Building K Means model with 3 clusters
km = KMeans(n_clusters=3, max_iter=100 , random_state = 101)
km.fit(dataset_scaled)
```

Out[36]:

```
KMeans(max_iter=100, n_clusters=3, random_state=101)
```

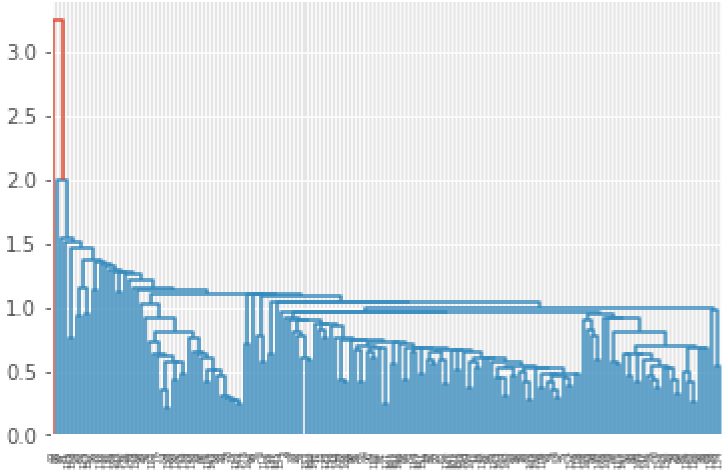
In [37]:

```
print(km.labels_)
print(km.labels_.shape)
```

```
[2 1 1 2 1 1 1 0 0 1 0 0 1 1 1 0 1 2 1 1 1 1 1 0 1 2 2 1 2 0 1 2 2 1 1 1 2
 2 2 1 2 1 0 0 0 1 1 1 1 2 2 0 1 0 0 2 2 1 0 2 0 1 1 2 2 1 2 0 0 1 1 1 2 0
 0 0 1 0 1 1 2 2 0 1 2 1 1 2 2 1 1 0 1 2 2 1 1 2 0 2 1 1 1 1 1 1 2 1 2 1 0
 0 2 2 0 0 2 1 1 1 1 1 0 0 1 1 2 1 0 2 1 1 2 0 0 0 2 1 0 0 1 1 2 1 0 0 1 2
 1 2 2 1 1 1 1 2 1 0 0 0 1 1 1 1 1 2 2]
(167,)
```

In [38]:

```
## Hierarchical Clustering
mergings_single = linkage(dataset_scaled, method="single", metric='euclidean')
dendrogram(mergings_single)
plt.show()
```

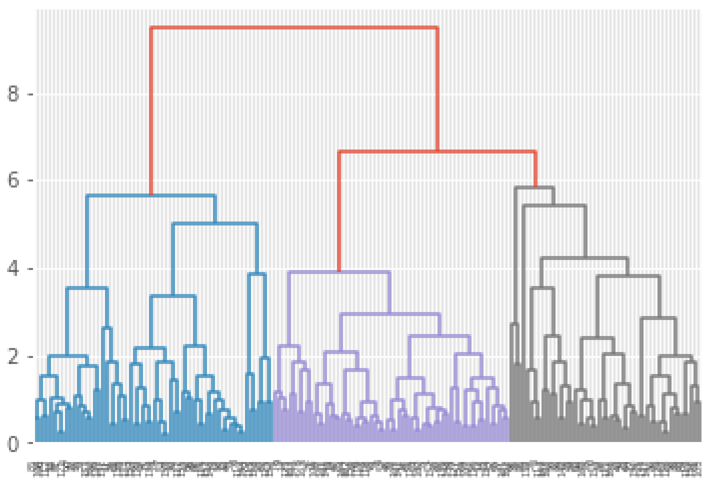


In [39]:

```
#Comment:

#With single linkage clusters are not interpretable.
#Using complete linkage.
```

In [40]:

```
mergings_complete = linkage(dataset_scaled, method="complete", metric='euclidean')
dendrogram(mergings_complete)
plt.show()
```



In [41]:

```
##Comment: The dendogram generate from Hierarchical Clustering with complete linkage sh
#Hence 3 clusters are chosen.
```

In [42]:

```
# Taking 3 clusters for hierarchical clustering
cluster_labels = cut_tree(mergings_complete, n_clusters=3).reshape(-1, )
print(cluster_labels)
print(cluster_labels.shape)
```

```
[0 1 1 0 2 1 1 2 2 1 2 2 1 2 1 2 1 0 1 1 1 0 2 2 1 0 0 1 0 2 1 0 0 2 1 1 0
 0 0 2 0 2 2 2 2 1 1 1 1 0 0 2 1 2 2 0 0 1 2 0 2 1 1 0 0 1 0 2 2 1 1 1 0 2
 2 2 1 2 1 1 0 0 2 1 0 2 2 0 0 2 2 2 1 0 0 2 2 0 2 0 1 1 1 1 2 1 0 1 0 1 2
 2 0 0 2 2 0 2 1 1 1 2 2 2 1 1 0 1 2 0 1 2 0 2 2 2 0 0 2 2 1 1 0 2 2 2 1 0
 1 0 0 1 1 2 1 0 1 2 2 2 2 1 1 1 1 0 0]
(167,)
```

In [43]:
```
dataset_clustered = dataset_updated.iloc[:,:]
dataset_clustered = pd.concat([dataset_clustered, pd.DataFrame(km.labels_, columns=['cl
dataset_clustered = pd.concat([dataset_clustered, pd.DataFrame(cluster_labels, columns=
dataset_clustered.head()
```

Out[43]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	55.30	41.9174	248.297	1610.0	9.44	56.2	5.82	553.0
1	Albania	16.6	1145.20	267.8950	1987.740	9930.0	4.49	76.3	1.65	4090.0
2	Algeria	27.3	1712.64	185.9820	1400.440	12900.0	16.10	76.5	2.89	4460.0
3	Angola	119.0	2199.19	100.6050	1514.370	5900.0	22.40	60.1	6.16	3530.0
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100.0	1.44	76.8	2.13	12200.0

In [44]:
```
#Comment:
#The new dataframe country_df_clustered has original data along with cluster labels
#given by both K Means and Hierarchical clustering model
```

In [46]:
```
print(dataset_clustered['cluster_id_km'].value_counts())
print(dataset_clustered['cluster_id_hc'].value_counts())
```

```
1    80
2    46
0    41
Name: cluster_id_km, dtype: int64
2    60
1    59
0    48
Name: cluster_id_hc, dtype: int64
```

In [47]:
```
print("Cluster 0 of Hierarchical Clustering model")
print(dataset_clustered[dataset_clustered['cluster_id_hc'] == 0].country.unique())

print("Cluster 1 of Hierarchical Clustering model")
print(dataset_clustered[dataset_clustered['cluster_id_hc'] == 1].country.unique())

print("Cluster 2 of Hierarchical Clustering model")
print(dataset_clustered[dataset_clustered['cluster_id_hc'] == 2].country.unique())
```

```
Cluster 0 of Hierarchical Clustering model
['Afghanistan' 'Angola' 'Benin' 'Botswana' 'Burkina Faso' 'Burundi'
 'Cameroon' 'Central African Republic' 'Chad' 'Comoros' 'Congo, Dem. Rep.'
 'Congo, Rep.' "Cote d'Ivoire" 'Equatorial Guinea' 'Eritrea' 'Gabon'
 'Gambia' 'Ghana' 'Guinea' 'Guinea-Bissau' 'Haiti' 'Iraq' 'Kenya'
 'Kiribati' 'Lao' 'Lesotho' 'Liberia' 'Madagascar' 'Malawi' 'Mali'
 'Mauritania' 'Mozambique' 'Namibia' 'Niger' 'Nigeria' 'Pakistan' 'Rwanda'
 'Senegal' 'Sierra Leone' 'Solomon Islands' 'South Africa' 'Sudan'
 'Tanzania' 'Timor-Leste' 'Togo' 'Uganda' 'Yemen' 'Zambia']
Cluster 1 of Hierarchical Clustering model
['Albania' 'Algeria' 'Argentina' 'Armenia' 'Azerbaijan' 'Bangladesh'
 'Belarus' 'Belize' 'Bhutan' 'Bolivia' 'Bosnia and Herzegovina' 'Bulgaria'
 'Cambodia' 'Cape Verde' 'China' 'Colombia' 'Dominican Republic' 'Ecuador'
 'Egypt' 'El Salvador' 'Fiji' 'Georgia' 'Grenada' 'Guatemala' 'Guyana'
```

```
 'India' 'Indonesia' 'Iran' 'Jamaica' 'Jordan' 'Kazakhstan'
 'Kyrgyz Republic' 'Macedonia, FYR' 'Mauritius' 'Micronesia, Fed. Sts.'
 'Moldova' 'Mongolia' 'Morocco' 'Myanmar' 'Nepal' 'Paraguay' 'Peru'
 'Philippines' 'Romania' 'Russia' 'Samoa' 'Serbia' 'Sri Lanka'
 'St. Vincent and the Grenadines' 'Tajikistan' 'Thailand' 'Tonga'
 'Tunisia' 'Turkmenistan' 'Ukraine' 'Uzbekistan' 'Vanuatu' 'Venezuela'
 'Vietnam']
Cluster 2 of Hierarchical Clustering model
['Antigua and Barbuda' 'Australia' 'Austria' 'Bahamas' 'Bahrain'
 'Barbados' 'Belgium' 'Brazil' 'Brunei' 'Canada' 'Chile' 'Costa Rica'
 'Croatia' 'Cyprus' 'Czech Republic' 'Denmark' 'Estonia' 'Finland'
 'France' 'Germany' 'Greece' 'Hungary' 'Iceland' 'Ireland' 'Israel'
 'Italy' 'Japan' 'Kuwait' 'Latvia' 'Lebanon' 'Libya' 'Lithuania'
 'Luxembourg' 'Malaysia' 'Maldives' 'Malta' 'Montenegro' 'Netherlands'
 'New Zealand' 'Norway' 'Oman' 'Panama' 'Poland' 'Portugal' 'Qatar'
 'Saudi Arabia' 'Seychelles' 'Singapore' 'Slovak Republic' 'Slovenia'
 'South Korea' 'Spain' 'Suriname' 'Sweden' 'Switzerland' 'Turkey'
 'United Arab Emirates' 'United Kingdom' 'United States' 'Uruguay']
```

In [48]:

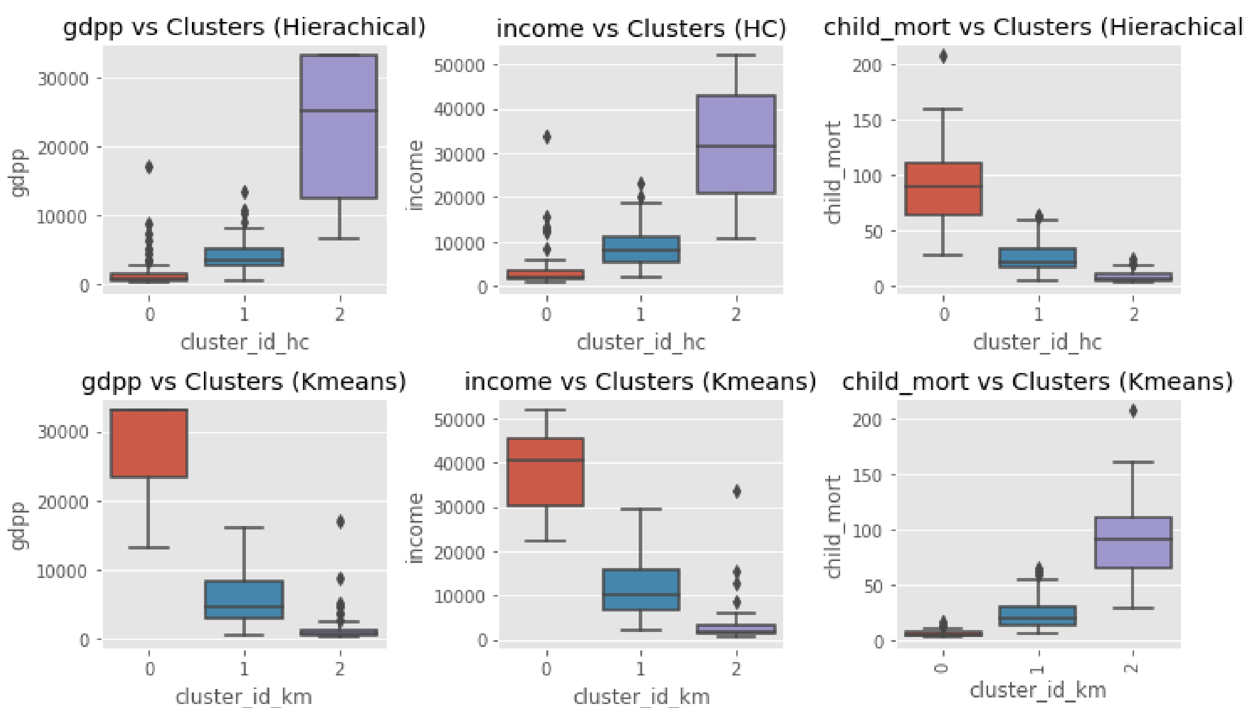
```
print("Cluster 0 of KMeans model")
print(dataset_clustered[dataset_clustered['cluster_id_km'] == 0].country.unique())

print("Cluster 1 of KMeans model")
print(dataset_clustered[dataset_clustered['cluster_id_km'] == 1].country.unique())

print("Cluster 2 of KMeans model")
print(dataset_clustered[dataset_clustered['cluster_id_km'] == 2].country.unique())
```

```
Cluster 0 of KMeans model
['Australia' 'Austria' 'Bahamas' 'Bahrain' 'Belgium' 'Brunei' 'Canada'
 'Cyprus' 'Czech Republic' 'Denmark' 'Estonia' 'Finland' 'France'
 'Germany' 'Greece' 'Hungary' 'Iceland' 'Ireland' 'Israel' 'Italy' 'Japan'
 'Kuwait' 'Luxembourg' 'Malta' 'Netherlands' 'New Zealand' 'Norway' 'Oman'
 'Portugal' 'Qatar' 'Saudi Arabia' 'Singapore' 'Slovak Republic'
 'Slovenia' 'South Korea' 'Spain' 'Sweden' 'Switzerland'
 'United Arab Emirates' 'United Kingdom' 'United States']
Cluster 1 of KMeans model
['Albania' 'Algeria' 'Antigua and Barbuda' 'Argentina' 'Armenia'
 'Azerbaijan' 'Bangladesh' 'Barbados' 'Belarus' 'Belize' 'Bhutan'
 'Bolivia' 'Bosnia and Herzegovina' 'Botswana' 'Brazil' 'Bulgaria'
 'Cambodia' 'Cape Verde' 'Chile' 'China' 'Colombia' 'Costa Rica' 'Croatia'
 'Dominican Republic' 'Ecuador' 'Egypt' 'El Salvador' 'Fiji' 'Georgia'
 'Grenada' 'Guatemala' 'Guyana' 'India' 'Indonesia' 'Iran' 'Jamaica'
 'Jordan' 'Kazakhstan' 'Kyrgyz Republic' 'Latvia' 'Lebanon' 'Libya'
 'Lithuania' 'Macedonia, FYR' 'Malaysia' 'Maldives' 'Mauritius'
 'Micronesia, Fed. Sts.' 'Moldova' 'Mongolia' 'Montenegro' 'Morocco'
 'Myanmar' 'Nepal' 'Panama' 'Paraguay' 'Peru' 'Philippines' 'Poland'
 'Romania' 'Russia' 'Samoa' 'Serbia' 'Seychelles' 'South Africa'
 'Sri Lanka' 'St. Vincent and the Grenadines' 'Suriname' 'Tajikistan'
 'Thailand' 'Tonga' 'Tunisia' 'Turkey' 'Turkmenistan' 'Ukraine' 'Uruguay'
 'Uzbekistan' 'Vanuatu' 'Venezuela' 'Vietnam']
Cluster 2 of KMeans model
['Afghanistan' 'Angola' 'Benin' 'Burkina Faso' 'Burundi' 'Cameroon'
 'Central African Republic' 'Chad' 'Comoros' 'Congo, Dem. Rep.'
 'Congo, Rep.' "Cote d'Ivoire" 'Equatorial Guinea' 'Eritrea' 'Gabon'
 'Gambia' 'Ghana' 'Guinea' 'Guinea-Bissau' 'Haiti' 'Iraq' 'Kenya'
 'Kiribati' 'Lao' 'Lesotho' 'Liberia' 'Madagascar' 'Malawi' 'Mali'
 'Mauritania' 'Mozambique' 'Namibia' 'Niger' 'Nigeria' 'Pakistan' 'Rwanda'
 'Senegal' 'Sierra Leone' 'Solomon Islands' 'Sudan' 'Tanzania'
 'Timor-Leste' 'Togo' 'Uganda' 'Yemen' 'Zambia']
```

In [49]:

```
###Cluster Profiling
fig = plt.figure(figsize=(10,6))
```

```
ax1 = fig.add_subplot(2, 3, 1, title="gdpp vs Clusters (Hierachical)")
ax4 = fig.add_subplot(2, 3, 4, title="gdpp vs Clusters (Kmeans)")

ax2 = fig.add_subplot(2, 3, 2, title="income vs Clusters (HC)")
ax5 = fig.add_subplot(2, 3, 5, title="income vs Clusters (Kmeans)")

ax3 = fig.add_subplot(2, 3, 3, title="child_mort vs Clusters (Hierachical)")
ax6 = fig.add_subplot(2, 3, 6, title="child_mort vs Clusters (Kmeans)")

sns.boxplot(dataset_clustered['cluster_id_hc'],dataset_clustered['gdpp'],ax=ax1)
sns.boxplot(dataset_clustered['cluster_id_km'],dataset_clustered['gdpp'],ax=ax4)

sns.boxplot(dataset_clustered['cluster_id_hc'],dataset_clustered['income'],ax=ax2)
sns.boxplot(dataset_clustered['cluster_id_km'],dataset_clustered['income'],ax=ax5)

sns.boxplot(dataset_clustered['cluster_id_hc'],dataset_clustered['child_mort'],ax=ax3)
sns.boxplot(dataset_clustered['cluster_id_km'],dataset_clustered['child_mort'],ax=ax6)


plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



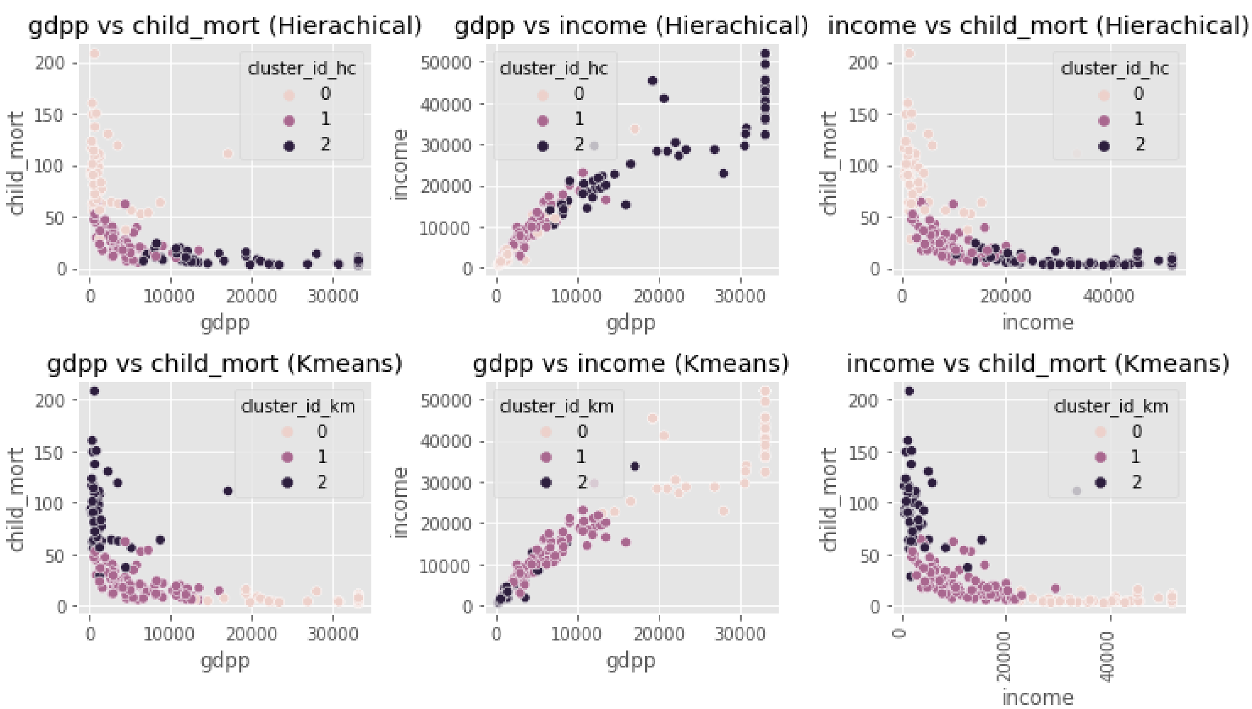
In [50]:

```
##Comment:

#Cluster 0 of Hierarchical model is similar with cluster 1 of Kmeans model
#   (socio-economically backward countries/under developed countries)
#Cluster 1 of Hierarchical model is similar with cluster 2 of Kmeans model
#Cluster 2 of Hierarchical model is similar with cluster 0 of Kmeans model (socio-econo

###Question Segment

#Analyse the clusters and identify the ones which are in dire need of aid. You can anal
     #how these three variables - [gdpp, child_mort and income] vary for each cluster o
     #and differentiate the clusters of developed countries from the clusters of under-
```

```
#Also, you need to perform visualisations on the clusters that have been formed. You ca
    #two of the three variables mentioned above on the X-Y axes and plotting a scatter
    # and differentiating the clusters. Make sure you create visualisations for all th
    #choose other types of plots like boxplots, etc.
```

In [53]:

```
fig = plt.figure(figsize=(10,6))

ax1 = fig.add_subplot(2, 3, 1, title="gdpp vs child_mort (Hierachical)")
ax4 = fig.add_subplot(2, 3, 4, title="gdpp vs child_mort (Kmeans)")

ax2 = fig.add_subplot(2, 3, 2, title="gdpp vs income (Hierachical)")
ax5 = fig.add_subplot(2, 3, 5, title="gdpp vs income (Kmeans)")

ax3 = fig.add_subplot(2, 3, 3, title="income vs child_mort (Hierachical)")
ax6 = fig.add_subplot(2, 3, 6, title="income vs child_mort (Kmeans)")

sns.scatterplot(dataset_clustered['gdpp'],dataset_clustered['child_mort'],hue=dataset_c
sns.scatterplot(dataset_clustered['gdpp'],dataset_clustered['child_mort'],hue=dataset_c

sns.scatterplot(dataset_clustered['gdpp'],dataset_clustered['income'],hue=dataset_clust
sns.scatterplot(dataset_clustered['gdpp'],dataset_clustered['income'],hue=dataset_clust

sns.scatterplot(dataset_clustered['income'],dataset_clustered['child_mort'],hue=dataset_
sns.scatterplot(dataset_clustered['income'],dataset_clustered['child_mort'],hue=dataset_

plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



In [54]:

```
##Comment:

#Top row represents results of Hierarchical Clustering and bottom row represents KMeans
   #Each column represents same pair of features.
```

```
#Countries with low gdpp (GDP per capita) have high child mortality
#Countries with low income (Net income per person) have high child mortality
#gdpp and income have strong linear relationship

#From the 3 features (gdpp, income and child_mort), the countries that need the financi
   #to cluster 0 of hierarchical model and cluster 1 of K means model.
```

In [55]:

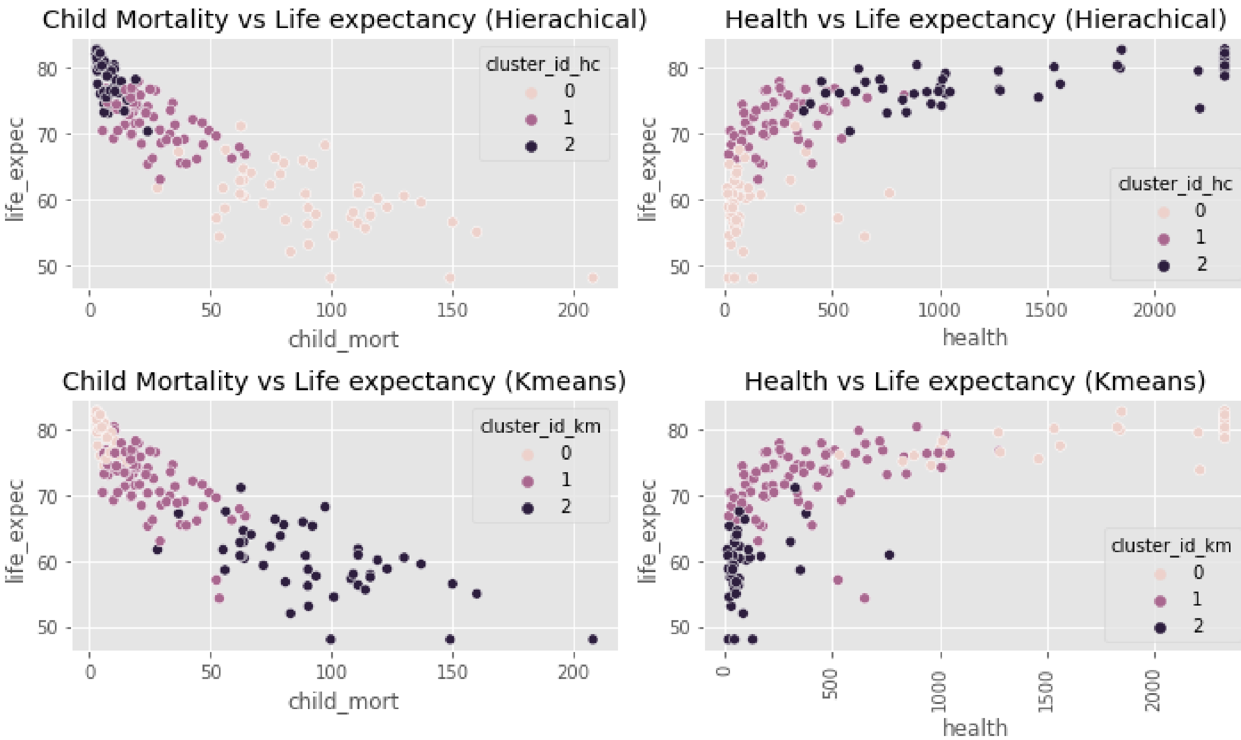
```
dataset_clustered.head(5)
```

Out[55]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	55.30	41.9174	248.297	1610.0	9.44	56.2	5.82	553.0
1	Albania	16.6	1145.20	267.8950	1987.740	9930.0	4.49	76.3	1.65	4090.0
2	Algeria	27.3	1712.64	185.9820	1400.440	12900.0	16.10	76.5	2.89	4460.0
3	Angola	119.0	2199.19	100.6050	1514.370	5900.0	22.40	60.1	6.16	3530.0
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100.0	1.44	76.8	2.13	12200.0

In [56]:

```
fig = plt.figure(figsize=(10,6))

ax1 = fig.add_subplot(2, 2, 1, title="Child Mortality vs Life expectancy (Hierachical)"
ax3 = fig.add_subplot(2, 2, 3, title="Child Mortality vs Life expectancy (Kmeans)")
ax2 = fig.add_subplot(2, 2, 2, title="Health vs Life expectancy (Hierachical)")
ax4 = fig.add_subplot(2, 2, 4, title="Health vs Life expectancy (Kmeans)")

sns.scatterplot(dataset_clustered['child_mort'],dataset_clustered['life_expec'],hue=dat
sns.scatterplot(dataset_clustered['child_mort'],dataset_clustered['life_expec'],hue=dat
sns.scatterplot(dataset_clustered['health'],dataset_clustered['life_expec'],hue=dataset
sns.scatterplot(dataset_clustered['health'],dataset_clustered['life_expec'],hue=dataset

plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



In [58]:

```
# Under-developed countries obtained from both the model

dataset_clustered[(dataset_clustered['cluster_id_hc']==0) | (dataset_clustered['cluster
```

Out[58]:

	country	cluster_id_hc	cluster_id_km
26	Burundi	0	2
88	Liberia	0	2
37	Congo, Dem. Rep.	0	2
112	Niger	0	2
132	Sierra Leone	0	2
...	...	...	...
33	Chile	2	1
163	Venezuela	1	1
41	Croatia	2	1
13	Barbados	2	1
49	Equatorial Guinea	0	2

126 rows × 3 columns

In [59]:

```
# Final list of under-developed countries, in order of socio-economic condition from wo
dataset_clustered[(dataset_clustered['cluster_id_hc']==0)].sort_values(by=['gdpp', 'inc
```

Out[59]:

	country

	country
26	Burundi
88	Liberia
37	Congo, Dem. Rep.
112	Niger
132	Sierra Leone
93	Madagascar
106	Mozambique
31	Central African Republic
94	Malawi
50	Eritrea
150	Togo
64	Guinea-Bissau
0	Afghanistan
56	Gambia
126	Rwanda

Conclusion:

gdpp, income and child_mort are 3 main driving factors for clustering. Low gdpp and income imply high rate of child mortality. Life expectancy in the under-developed countries is low because of high child mortality rate.

Hierarchical Clustering model is chosen as final model as Kmeans can produce different results depending on the initial positions of the centroids of the cluster.

Also KMeans needs prespecified number of clusters.

In Hierarchical model, the dendogram has better interpretability than KMeans and also does not need the number of clusters to be specified before.

Hence, Hierarchical Clustering model is chosen for the final list of countries.

In [ ]:

```
```