CMP4336 Intr. to Data Mining
Term Project Report
2020 Summer

Help International Dataset K-Means and Hierarchical Clustering Analysis

Onur Güzel

September 8, 2020

Abstract

This project aims to compare k-means clustering and hierarchical clustering methods with different parameters on a single dataset. Help International Dataset that includes expected life year, child mortality and so on, US Crime Reports until 1977 and updated at 2020. Comprehensive approach is supported by scatter, dendrograpic as well as other various methods of visualization.

Introduction:

Throughout the history of Earth people were continuously produced data in various fields. Nowadays data that is collected throughout the years is accumulated enough to make predictions out of them and extract information from a whole array of data called datasets. Regarding this improvement on the data science field, this report will clarify K-Means and Hierarchical Clustering on two different datasets. Datasets are provided by Help International NGO. Main purpose of this project is to make appropriate clustering according to the features provided by the dataset like export of goods and services per capita, net income per person. inflation etc.

Dataset Description:

There are two different datasets used for comparison of K-Means clustering and Hierarchical Clustering. Dataset used in the experiment is the Help International NGO Dataset that categorizes the countries using some socio-economic and health factors that determine the overall development of the country.

Methods:

During the experimentation various methods were used depending on the requirements. Hopkins Statistics used for tendency measurement of the dataset. According to results given in the appendix, it is observed that the dataset used in this experiment is suitable for clustering. After the analysis, data is scaled for better machine learning performance. Then the elbow curve method applied to in order to detect how many clusters the dataset has. After the observerations, it’s understood that three clusters will be enough. Another method used for clustering count is silhouette scoring. As it can be seen at the appendix silhouette scoring says that dataset can have three, four or five clusters. Therefore, we will compare Help International Dataset with three, four and five clusters and this comparison will be the main target of this paper. These numbers of clusters gathered from different methods will be tested by the K-Means clustering method by iterating over them. Another method used in classification is Hierarchical clustering method. Cluster profiling is used after hierarchical clustering, country segmentation was implemented in order to get a better business understanding.

Experimental Results:

Experimental results shows that when the dataset is clustered with 3 classes there exist 91 countries belonging to class 0, 28 countries belonging to class 1 and 48 countries belonging to class 2. When the cluster number is increased from 3 to 4, results are 0th class 48, 1st class 88, 2nd class 2 and 3rd class 29 countries. Last but not least when k equals to 5, 0th class 87, 1st class 2, 2nd class 47, 3rd class 30 and 4th class has 1 country.

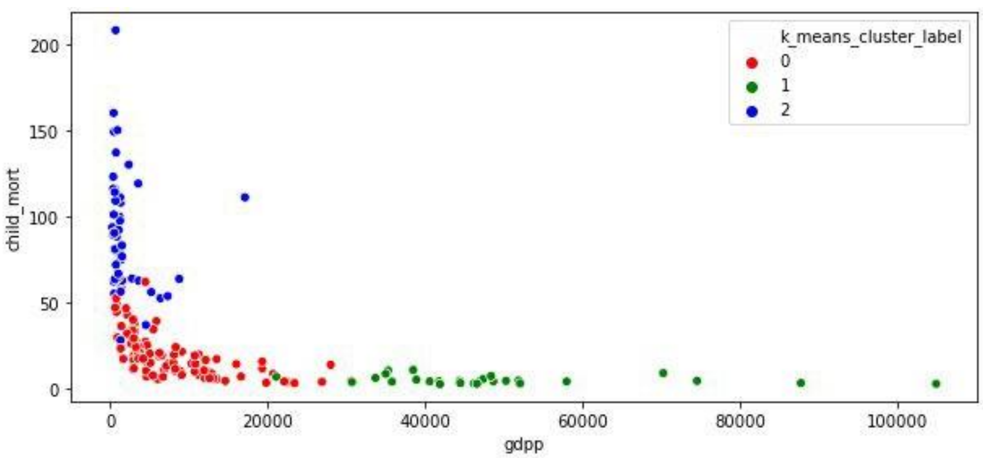


Figure 1. Three Clusters for “child_mort” and “gdpp”

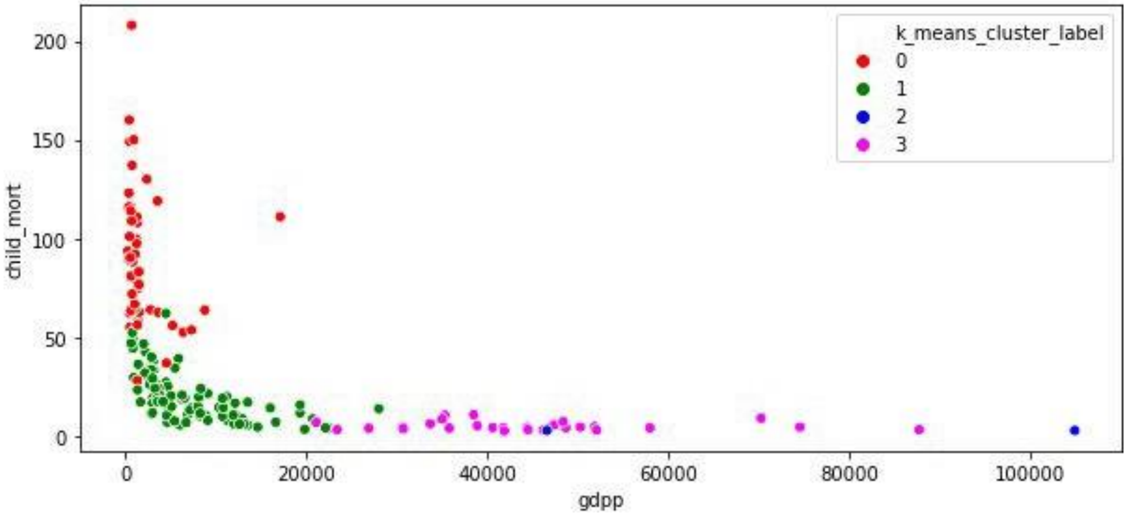


Figure 2. Four Clusters for “child_mort” and “gdpp”

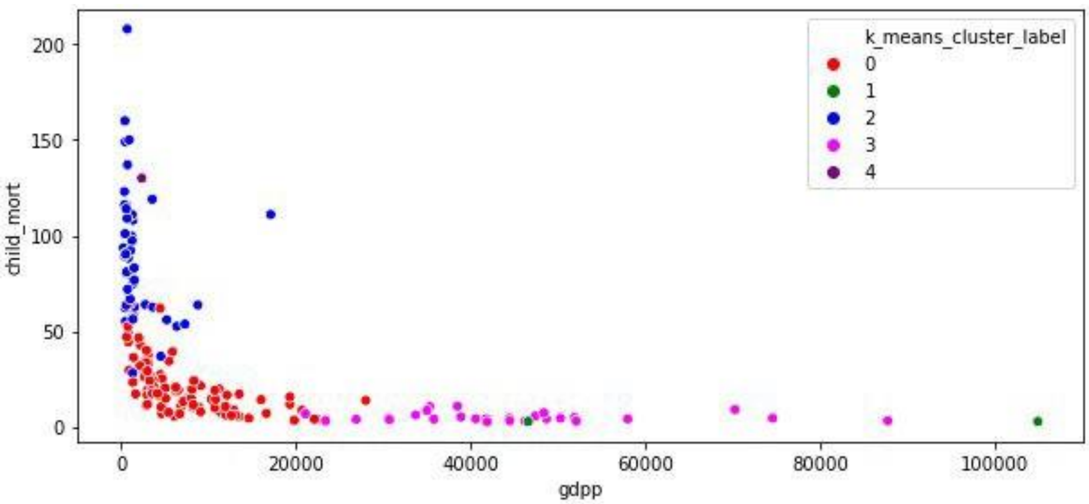


Figure 3. Five Clusters for “child_mor”t and “gdpp”

For more visualizations please refer to the appendix section detailed graphs is attached alongside with the source code.

According to graphs and results k value is chosen as three for better clustering. When K value is chosen 3 there appear 3 different classes. 0: developing countries, 1: developed countries, 2:under-developed countries. Similar results observed at scatter plot also can be seen at boxplot (see appendix). As a result, after deeper clustering and labeling methods, top 20 Countries that require aid on priority using the K-means algorithm are shown below.

Out[61]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	k_means_cluster_label
26	Burundi	93.6	8.92	11.60	39.2	764	12.30	57.7	6.26	231	Under-Developed Countries
88	Liberia	89.3	19.10	11.80	92.6	700	5.47	60.8	5.02	327	Under-Developed Countries
37	Congo, Dem. Rep.	116.0	41.10	7.91	49.6	609	20.80	57.5	6.54	334	Under-Developed Countries
112	Niger	123.0	22.20	5.16	49.1	814	2.55	58.8	7.49	348	Under-Developed Countries
132	Sierra Leone	160.0	16.80	13.10	34.5	1220	17.20	55.0	5.20	399	Under-Developed Countries
93	Madagascar	62.2	25.00	3.77	43.0	1390	8.79	60.8	4.60	413	Under-Developed Countries
106	Mozambique	101.0	31.50	5.21	46.2	918	7.64	54.5	5.56	419	Under-Developed Countries
31	Central African Republic	149.0	11.80	3.98	26.5	888	2.01	47.5	5.21	446	Under-Developed Countries
94	Malawi	90.5	22.80	6.59	34.9	1030	12.10	53.1	5.31	459	Under-Developed Countries
50	Eritrea	55.2	4.79	2.66	23.3	1420	11.60	61.7	4.61	482	Under-Developed Countries
150	Togo	90.3	40.20	7.65	57.3	1210	1.18	58.7	4.87	488	Under-Developed Countries
64	Guinea-Bissau	114.0	14.90	8.50	35.2	1390	2.97	55.6	5.05	547	Under-Developed Countries
0	Afghanistan	90.2	10.00	7.58	44.9	1610	9.44	56.2	5.82	553	Under-Developed Countries
56	Gambia	80.3	23.80	5.69	42.7	1660	4.30	65.5	5.71	562	Under-Developed Countries
126	Rwanda	63.6	12.00	10.50	30.0	1350	2.61	64.6	4.51	563	Under-Developed Countries
25	Burkina Faso	116.0	19.20	6.74	29.6	1430	6.81	57.9	5.87	575	Under-Developed Countries
155	Uganda	81.0	17.10	9.01	28.6	1540	10.60	56.8	6.15	595	Under-Developed Countries
63	Guinea	109.0	30.30	4.93	43.2	1190	16.10	58.0	5.34	648	Under-Developed Countries
66	Haiti	208.0	15.30	6.91	64.7	1500	5.45	32.1	3.33	662	Under-Developed Countries
147	Tanzania	71.9	18.70	6.01	29.1	2090	9.25	59.3	5.43	702	Under-Developed Countries

Figure 4. Top 20 Countries on help priority

In hierarchical clustering it can be observed that we have three main clusters among all the countries. Lines in blue, orange and green represent different classes in the cluster.

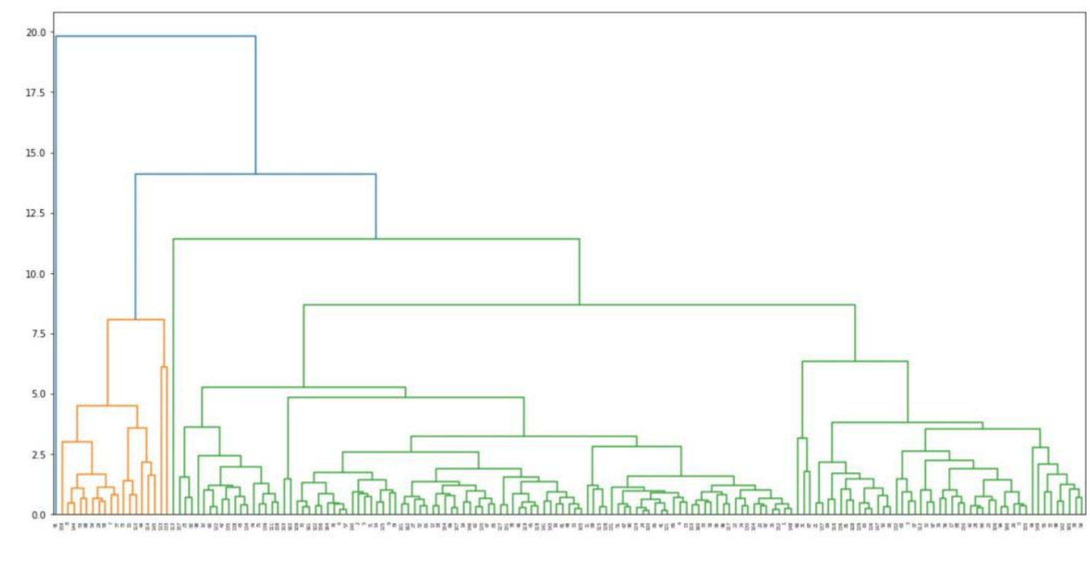


Figure 5. Hierarchical Clusters

Out[61]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster_labels
26	Burundi	93.6	20.6052	26.7960	90.552	764	12.30	57.7	6.26	231	Under-Developed Countries
88	Liberia	89.3	62.4570	38.5860	302.802	700	5.47	60.8	5.02	327	Under-Developed Countries
37	Congo, Dem. Rep.	116.0	137.2740	26.4194	165.664	609	20.80	57.5	6.54	334	Under-Developed Countries
112	Niger	123.0	77.2560	17.9568	170.868	814	2.55	58.8	7.49	348	Under-Developed Countries
132	Sierra Leone	160.0	67.0320	52.2690	137.655	1220	17.20	55.0	5.20	399	Under-Developed Countries
93	Madagascar	62.2	103.2500	15.5701	177.590	1390	8.79	60.8	4.60	413	Under-Developed Countries
106	Mozambique	101.0	131.9850	21.8299	193.578	918	7.64	54.5	5.56	419	Under-Developed Countries
31	Central African Republic	149.0	52.6280	17.7508	118.190	888	2.01	47.5	5.21	446	Under-Developed Countries
94	Malawi	90.5	104.6520	30.2481	160.191	1030	12.10	53.1	5.31	459	Under-Developed Countries
50	Eritrea	55.2	23.0878	12.8212	112.306	1420	11.60	61.7	4.61	482	Under-Developed Countries

Figure 6. Top 10 Country on Help Priority according to hierarchical clustering

Conclusion:

After the experimentations, we can conclude that both K-means and hierarchical clustering method is suitable for demographic analysis of the countries included in the data-set. During the analysis it was possible to determine each country's needs by different categories while classifying the countries in need of aid more than the other ones. In terms of algorithm efficiency both of them performed similar results but only difference was the significance in the computational efficiency. Hierarchical clustering takes significantly more time while computing the results compared to the K-means clustering. Time complexity of Hierarchical Clustering is $O(n^2)$ where K-means time complexity is $O(n)$.
In terms aid need countries both of the techniques performed identically. The list of countries needing aid is shown below.

- Burundi
- Liberia
- Congo, Dem. Rep.
- Niger
- Sierra Leone
- Madagascar
- Mozambique
- Central African Republic
- Malawi
- Eritrea

APPENDIX

Import required libraries

In [1]:
```python
try:
    import os # accessing directory structure
    import warnings # to supress warnings

    import numpy as np # linear algebra
    import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
    import matplotlib.pyplot as plt
    import seaborn as sns
    import sklearn
    import sklearn

    from sklearn.metrics.pairwise import pairwise_distances

    #Calculating the Hopkins statistic
    from sklearn.neighbors import NearestNeighbors
    from sklearn.preprocessing import StandardScaler
    from sklearn.cluster import KMeans
    from sklearn.metrics import silhouette_score
    from random import sample
    from numpy.random import uniform
    from math import isnan

    # To perform Hierarchical clustering
    from scipy.cluster.hierarchy import linkage
    from scipy.cluster.hierarchy import dendrogram
    from scipy.cluster.hierarchy import cut_tree

except ImportError as err:
    print("Some required files could not be found for program...",
          "\nPlease contact with the manufacturer!")

warnings.filterwarnings('ignore')
```

Data Loading

List datasets

In [2]:
```python
for dirname, _, filenames in os.walk('..\Dataset'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
..\Dataset\country-data.csv
..\Dataset\data-dictionary.csv
```

Get data

In [3]:
```python
data = pd.read_csv("../Dataset/country-data.csv", sep = ",")
data.head()
```

Out[3]:

	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

Print some information about data

In [4]:
```python
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
```

Get dictionary

In [5]:
```
dictionary = pd.read_csv("../Dataset/data-dictionary.csv", sep = ",")
dictionary.head(len(dictionary))
```

Out[5]:

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

Print some information about dictionary

In [6]:
```
dictionary.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Column Name  10 non-null     object
 1   Description  10 non-null     object
dtypes: object(2)
memory usage: 288.0+ bytes
```

Print shapes of both data and dictionary

In [7]:
```
#database dimension
print("Database dimension   :", data.shape)
print("Database size        :", data.size)
print("Dictionary dimension :", dictionary.shape)
print("Dictionary size      :", dictionary.size)
```

```
Database dimension   : (167, 10)
Database size        : 1670
Dictionary dimension : (10, 2)
Dictionary size      : 20
```

Check duplicates and missing values

In [8]:
```
if(data.duplicated().sum() == 0):
    print("No duplicated rows!")
else:
    print("Error! Duplicated row(s)!")

if(data.isnull().sum().sum() == 0):
    print("No missing value!")
else:
    print("Error! Missing Value(s)!")
    print("Number of missing value : ", data.isnull().sum().sum())
```

```
No duplicated rows!
No missing value!
```

Exploratory Data Analysis (EDA)

Deducing imports,exports and health spending from percentage values to actual values of their GDP per capita

In [9]:
```
# Converting exports,imports and health spending percentages to absolute values.

data['exports'] = data['exports'] * data['gdpp']/100
data['imports'] = data['imports'] * data['gdpp']/100
data['health']  = data['health']  * data['gdpp']/100

data.head(10)
```

Out[9]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	55.30	41.9174	248.297	1610	9.440	56.2	5.82	553
1	Albania	16.6	1145.20	267.8950	1987.740	9930	4.490	76.3	1.65	4090
2	Algeria	27.3	1712.64	185.9820	1400.440	12900	16.100	76.5	2.89	4460
3	Angola	119.0	2199.19	100.6050	1514.370	5900	22.400	60.1	6.16	3530
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100	1.440	76.8	2.13	12200
5	Argentina	14.5	1946.70	834.3000	1648.000	18700	20.900	75.8	2.37	10300
6	Armenia	18.1	669.76	141.6800	1458.660	6700	7.770	73.3	1.69	3220
7	Australia	4.8	10276.20	4530.8700	10847.100	41400	1.160	82.0	1.93	51900
8	Austria	4.3	24059.70	5159.0000	22418.200	43200	0.873	80.5	1.44	46900
9	Azerbaijan	39.2	3171.12	343.3920	1208.880	16000	13.800	69.1	1.92	5840

Choose the countries that are in the direst need of aid

Data Preparation

In [10]:
```
countries = data.columns[0]
list_of_features = data.columns.drop('country')
list_of_ascending_dictionary = {
  "child_mort": False,
  "exports": True,
  "health": True,
  "imports": True,
  "income": True,
  "inflation": False,
  "life_expec": True,
  "total_fer": False,
  "gdpp": True
}

print("X axis feature : ", countries)
print("Y axis features : ", list_of_features)
print("Dictionary : \n", list_of_ascending_dictionary)
```

```
X axis feature :  country
Y axis features :  Index(['child_mort', 'exports', 'health', 'imports', 'income', 'inflation',
       'life_expec', 'total_fer', 'gdpp'],
      dtype='object')
Dictionary :
 {'child_mort': False, 'exports': True, 'health': True, 'imports': True, 'income': True, 'inflation': False, 'life_expec': True, 'total_fer': False,
'gdpp': True}
```

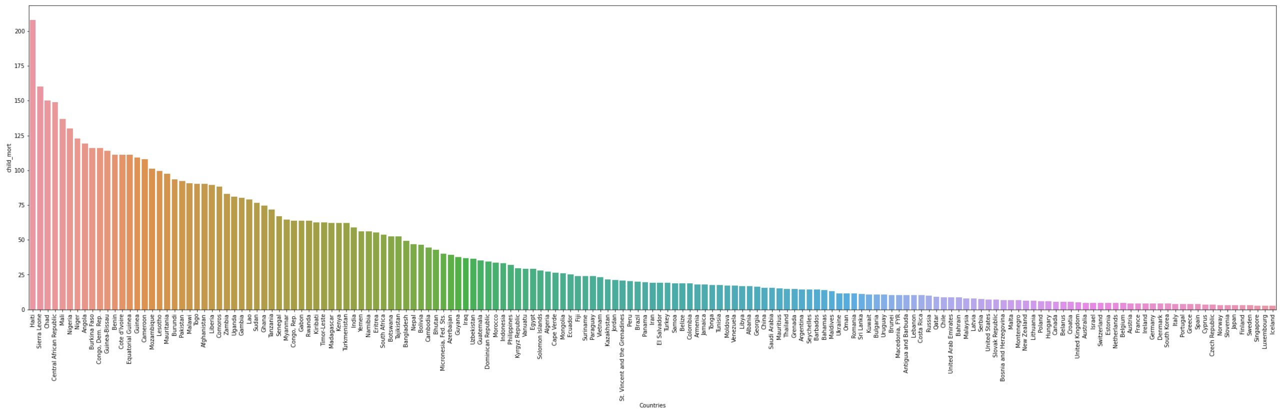
Graphics respectively;

- Child Mortality Rate : Death of children under 5 years of age per 1000 live births
- Exports: Exports of goods and services. Given as % of the Total GDP
- Health :Total health spending as % of Total GDP.
- Imports: Imports of goods and services. Given as % of the Total GDP
- Per capita Income : Net income per person
- Inflation: The measurement of the annual growth rate of the Total GDP
- Life Expectancy: The average number of years a new born child would live if the current mortality patterns are to remain same
- Fertility Rate : The number of children that would be born to each woman if the current age-fertility rates remain the same
- The GDP per capita : Calculated as the Total GDP divided by the total population.

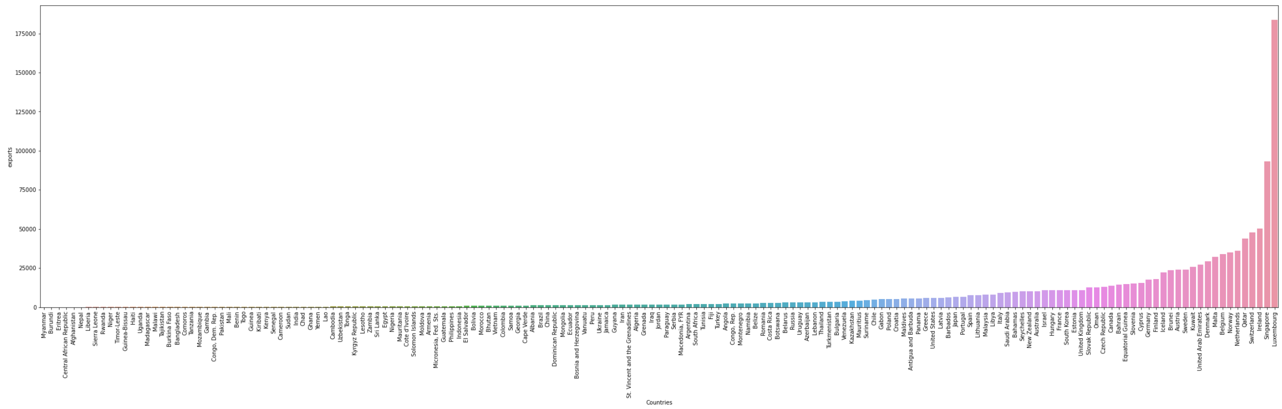
In [11]:

```
for feature in list_of_features:

    plt.figure(figsize = (40, 10))

    health = data[[countries, feature]].sort_values(feature, ascending = list_of_ascending_dictionary[feature])

    axis = sns.barplot(x = countries, y = feature, data = health)
    axis.set(xlabel = 'Countries', ylabel = feature)

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

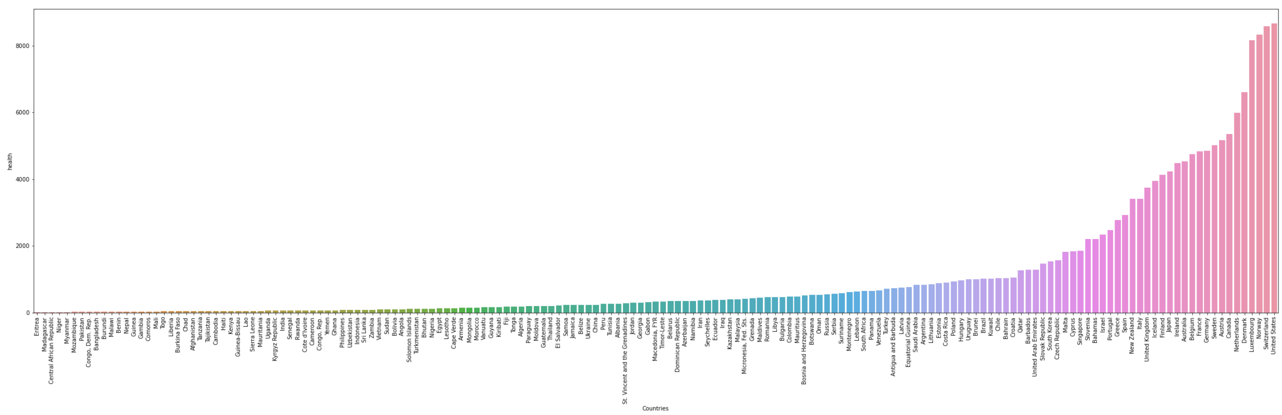
Child Mortality Rate : Death of children under 5 years of age per 1000 live births



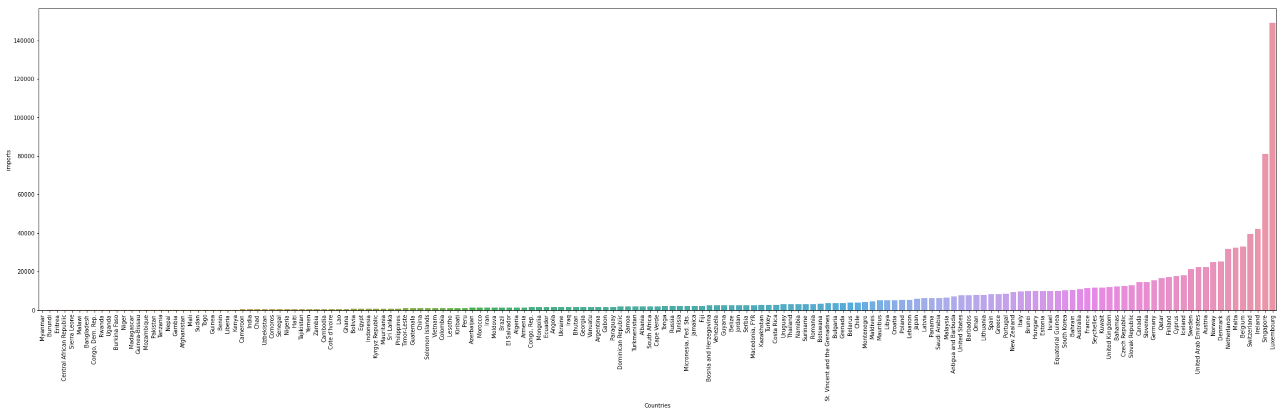
Exports: Exports of goods and services. Given as % of the Total GDP



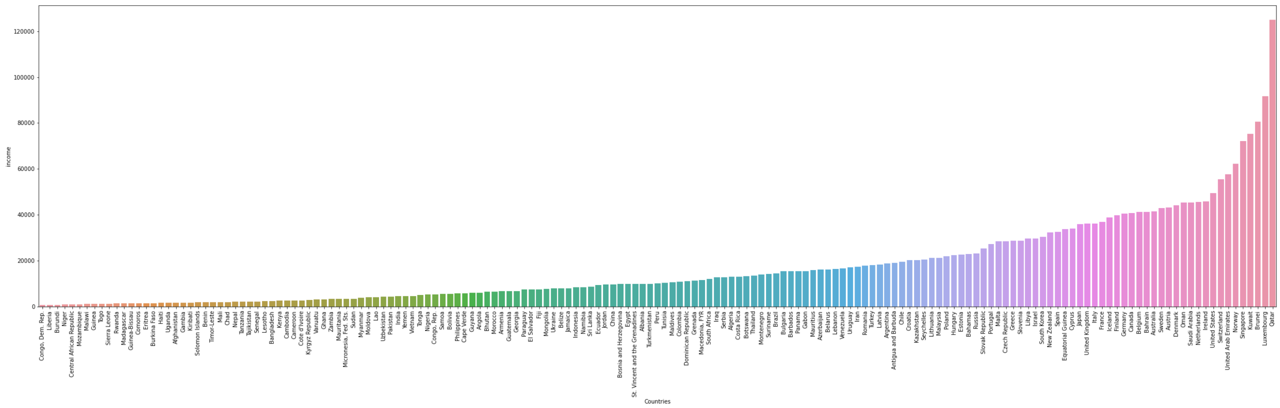
Health :Total health spending as % of Total GDP.



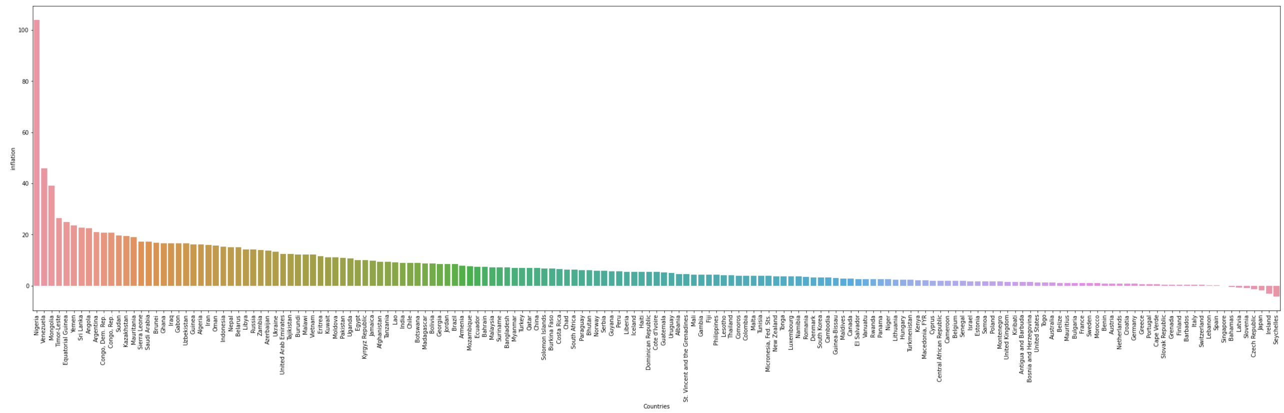
Imports: Imports of goods and services. Given as % of the Total GDP



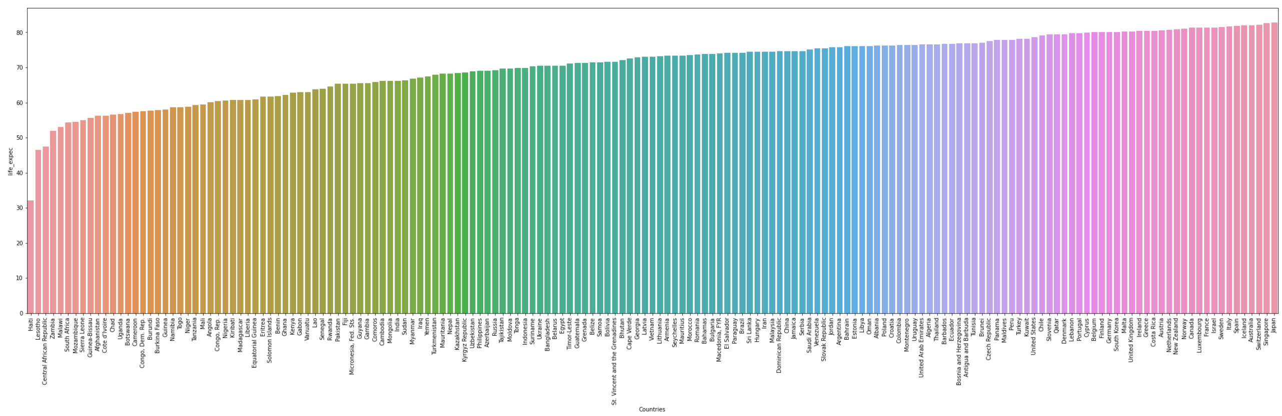
Per capita Income : Net income per person



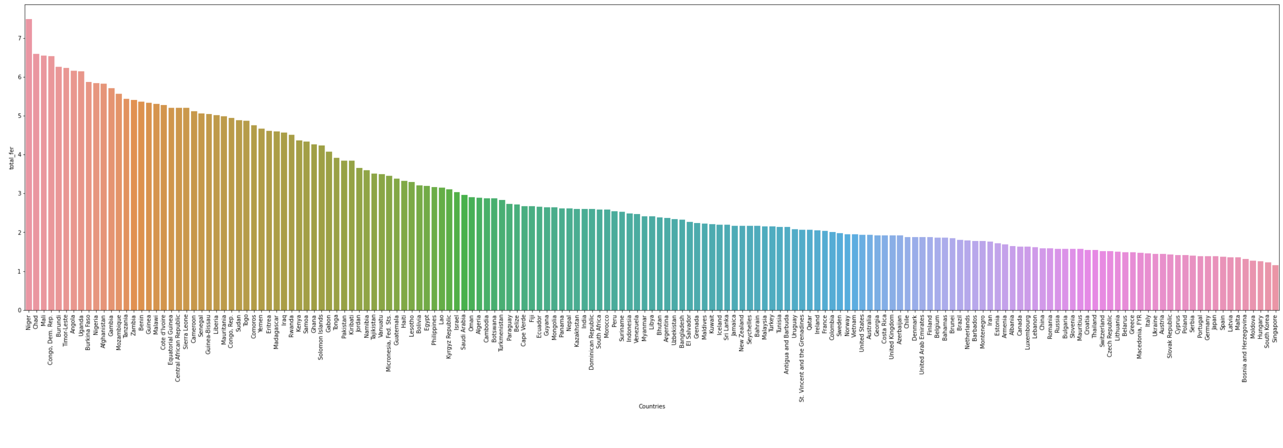
Inflation: The measurement of the annual growth rate of the Total GDP



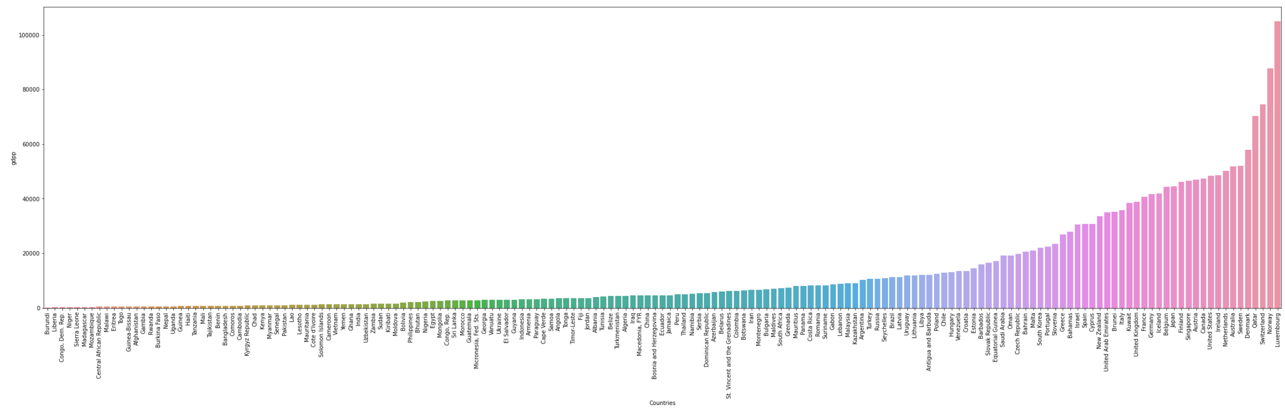
Life Expectancy: The average number of years a new born child would live if the current mortality patterns are to remain same



Fertility Rate : The number of children that would be born to each woman if the current age-fertility rates remain the same



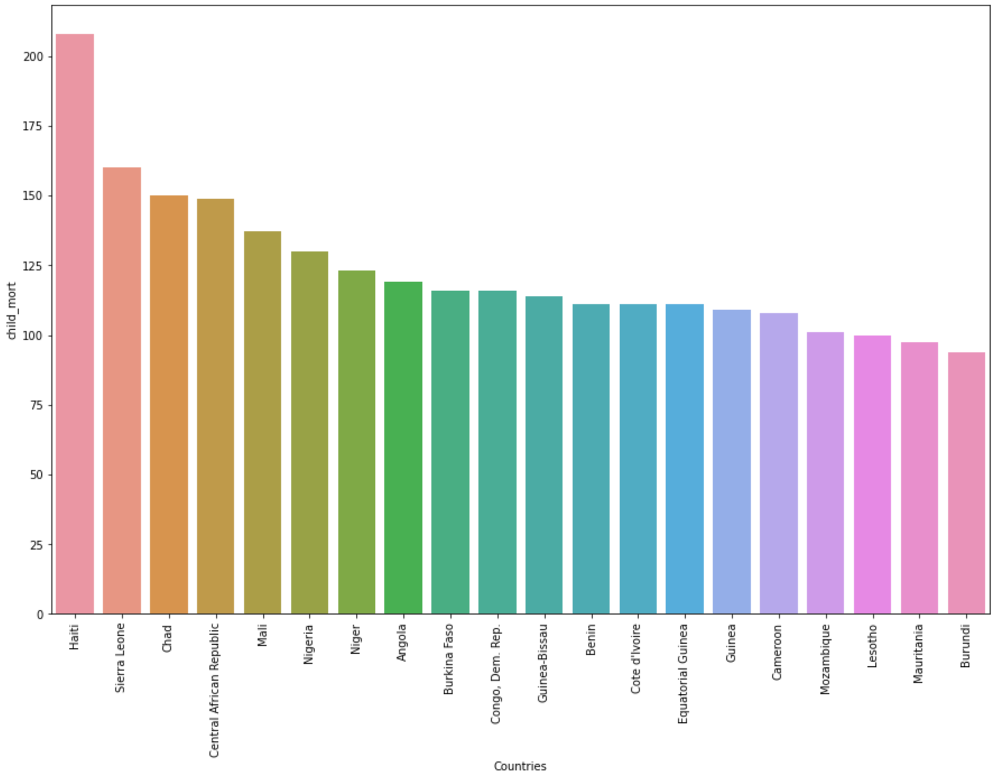
The GDP per capita : Calculated as the Total GDP divided by the total population.



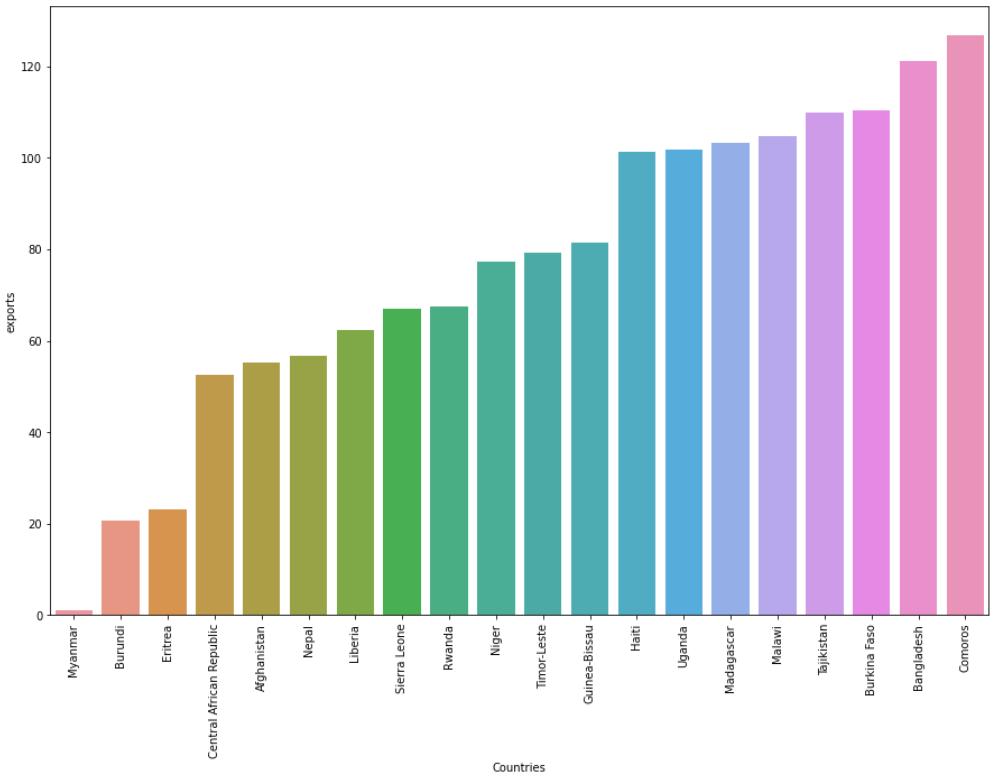
Graphics respectively;

- Child Mortality Rate : Death of children under 5 years of age per 1000 live births
- Exports: Exports of goods and services. Given as %age of the Total GDP
- Health :Total health spending as %age of Total GDP.
- Imports: Imports of goods and services. Given as %age of the Total GDP
- Per capita Income : Net income per person
- Inflation: The measurement of the annual growth rate of the Total GDP
- Life Expectancy: The average number of years a new born child would live if the current mortality patterns are to remain same
- Fertility Rate : The number of children that would be born to each woman if the current age-fertility rates remain the same
- The GDP per capita : Calculated as the Total GDP divided by the total population.

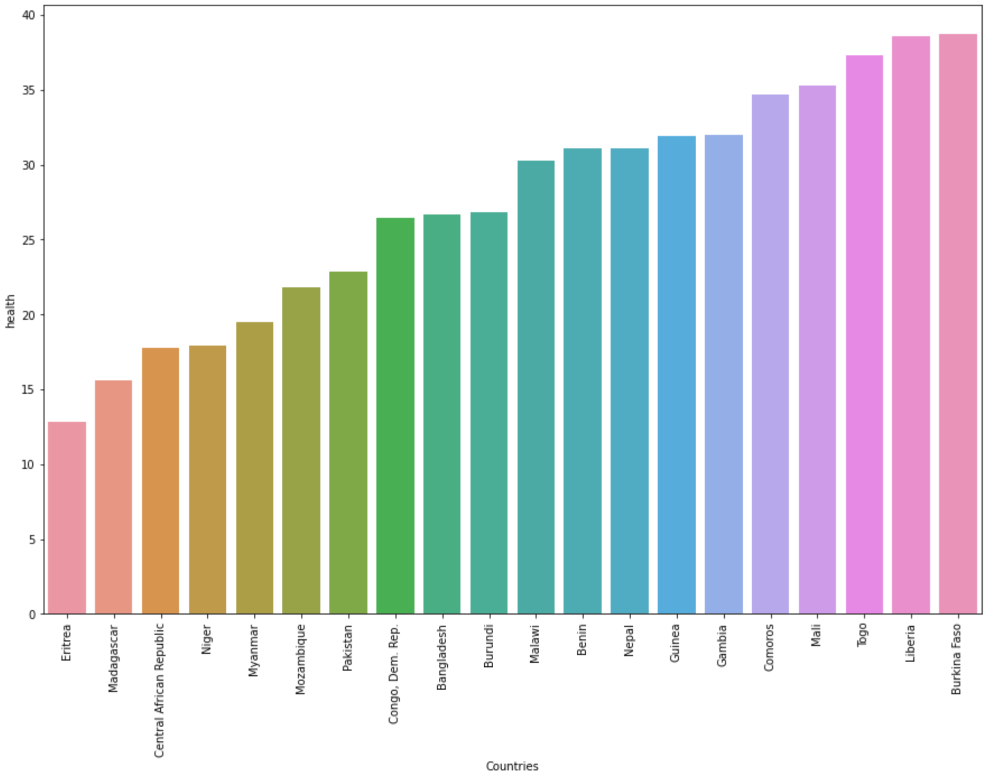
In [12]:
```
for feature in list_of_features:

    plt.figure(figsize = (15, 10))

    health = data[[countries, feature]].sort_values(feature, ascending = list_of_ascending_dictionary[feature]).head(20)

    axis = sns.barplot(x = countries, y = feature, data = health)
    axis.set(xlabel = 'Countries', ylabel = feature)

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

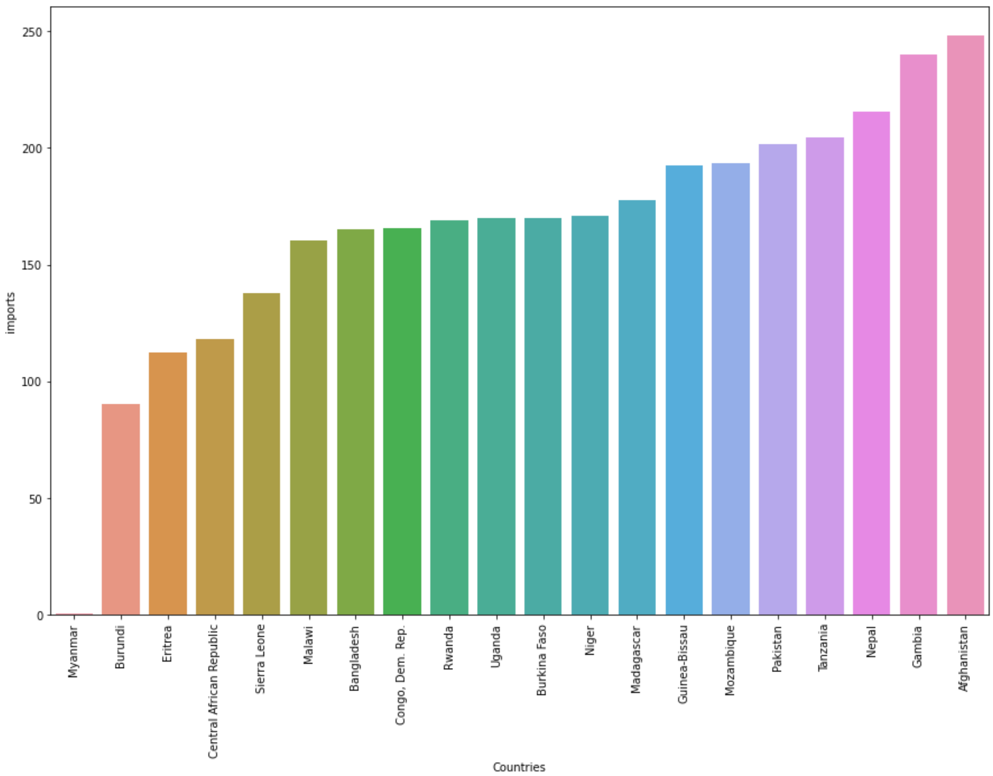


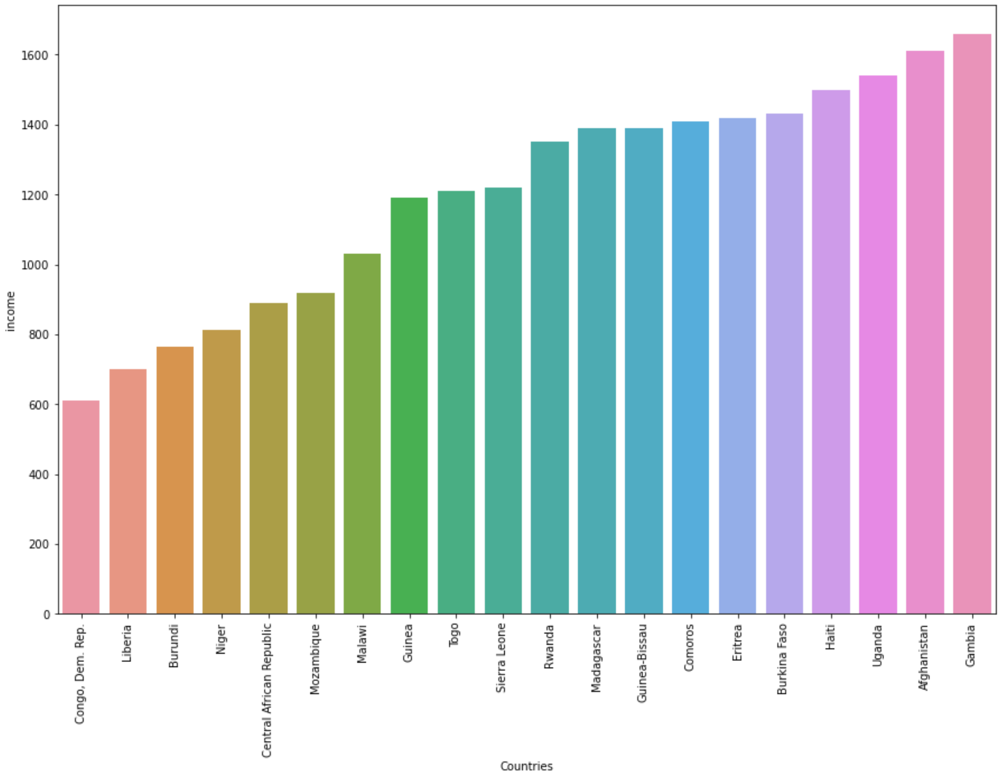
200
175
150
125
100
75
50
25
0
child_mort
Haiti
Sierra Leone
Chad
Central African Republic
Mali
Nigeria
Niger
Angola
Burkina Faso
Congo, Dem. Rep.
Guinea-Bissau
Benin
Cote d'Ivoire
Equatorial Guinea
Guinea
Cameroon
Mozambique
Lesotho
Mauritania
Burundi
Countries



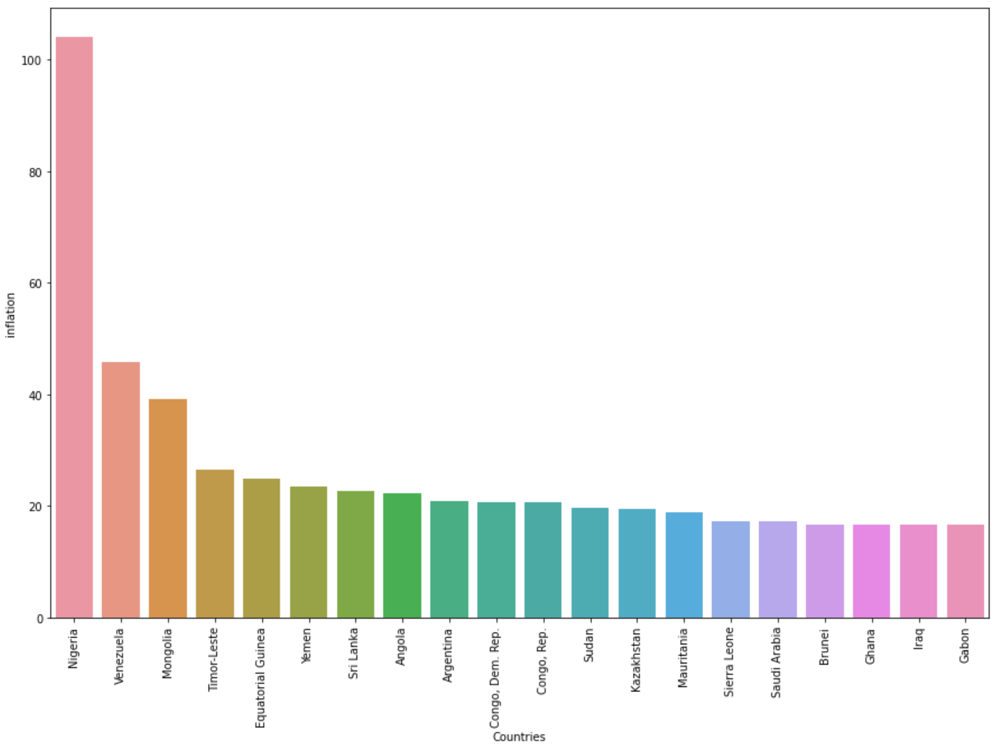
120
100
80
60
40
20
0
exports
Myanmar
Burundi
Eritrea
Central African Republic
Afghanistan
Nepal
Liberia
Sierra Leone
Rwanda
Niger
Timor-Leste
Guinea-Bissau
Haiti
Uganda
Madagascar
Malawi
Tajikistan
Burkina Faso
Bangladesh
Comoros
Countries


health
Countries
Eritrea
Madagascar
Central African Republic
Niger
Myanmar
Mozambique
Pakistan
Congo, Dem. Rep.
Bangladesh
Burundi
Malawi
Benin
Nepal
Guinea
Gambia
Comoros
Mali
Togo
Liberia
Burkina Faso

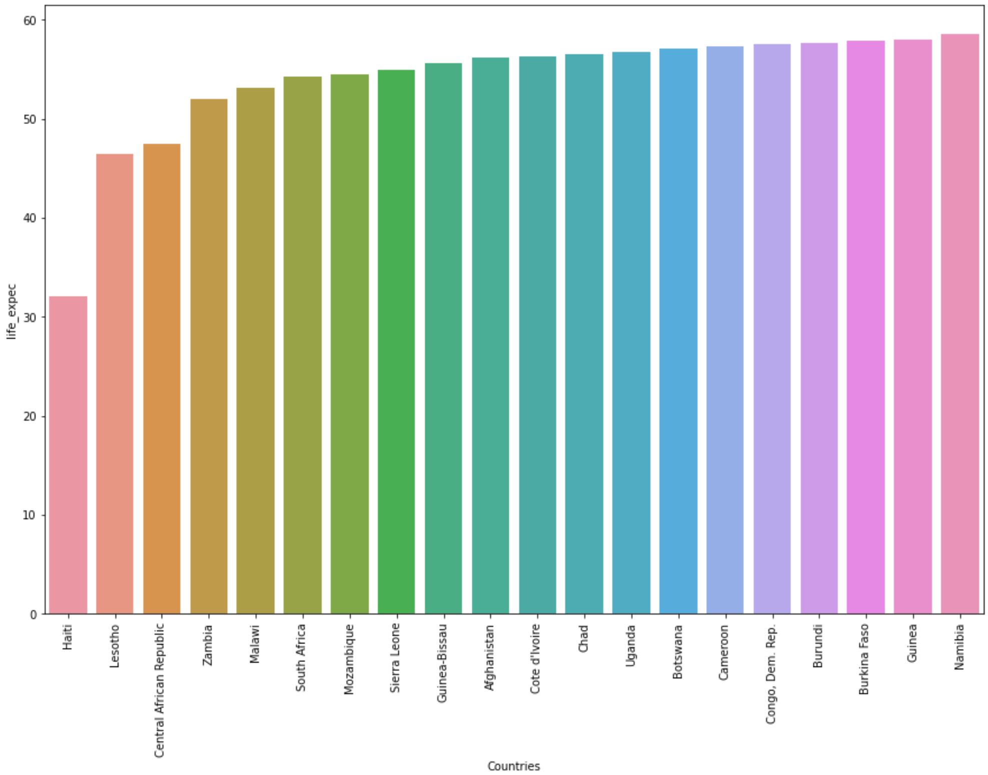

imports
Countries
Myanmar
Burundi
Eritrea
Central African Republic
Sierra Leone
Malawi
Bangladesh
Congo, Dem. Rep.
Rwanda
Uganda
Burkina Faso
Niger
Madagascar
Guinea-Bissau
Mozambique
Pakistan
Tanzania
Nepal
Gambia
Afghanistan



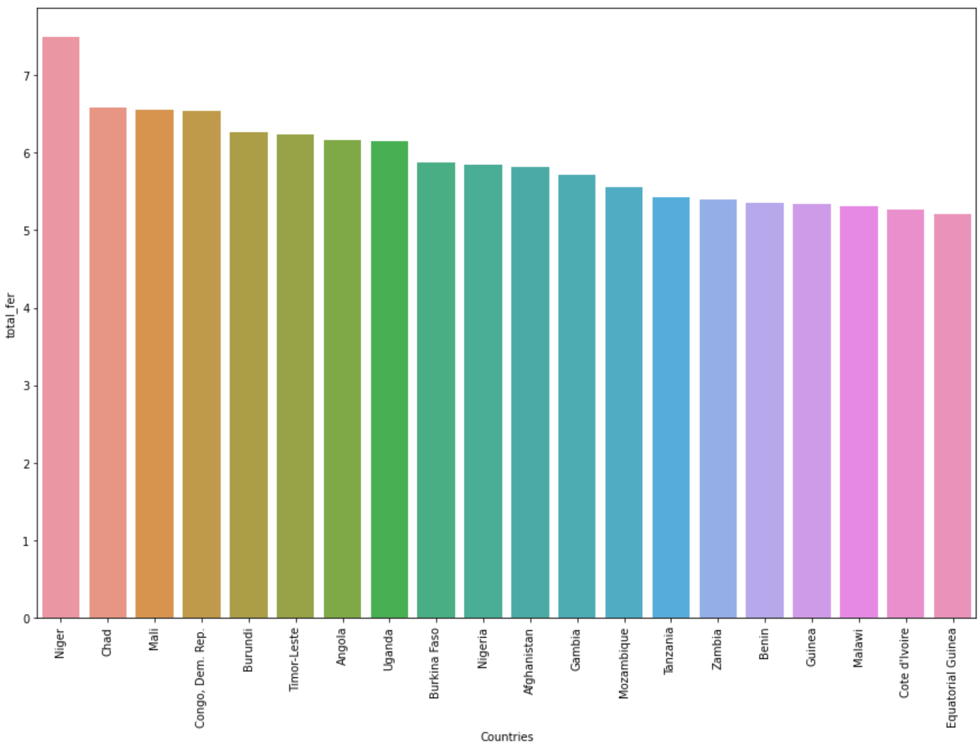
income
0
200
400
600
800
1000
1200
1400
1600
Congo, Dem. Rep.
Liberia
Burundi
Niger
Central African Republic
Mozambique
Malawi
Guinea
Togo
Sierra Leone
Rwanda
Madagascar
Guinea-Bissau
Comoros
Eritrea
Burkina Faso
Haiti
Uganda
Afghanistan
Gambia
Countries



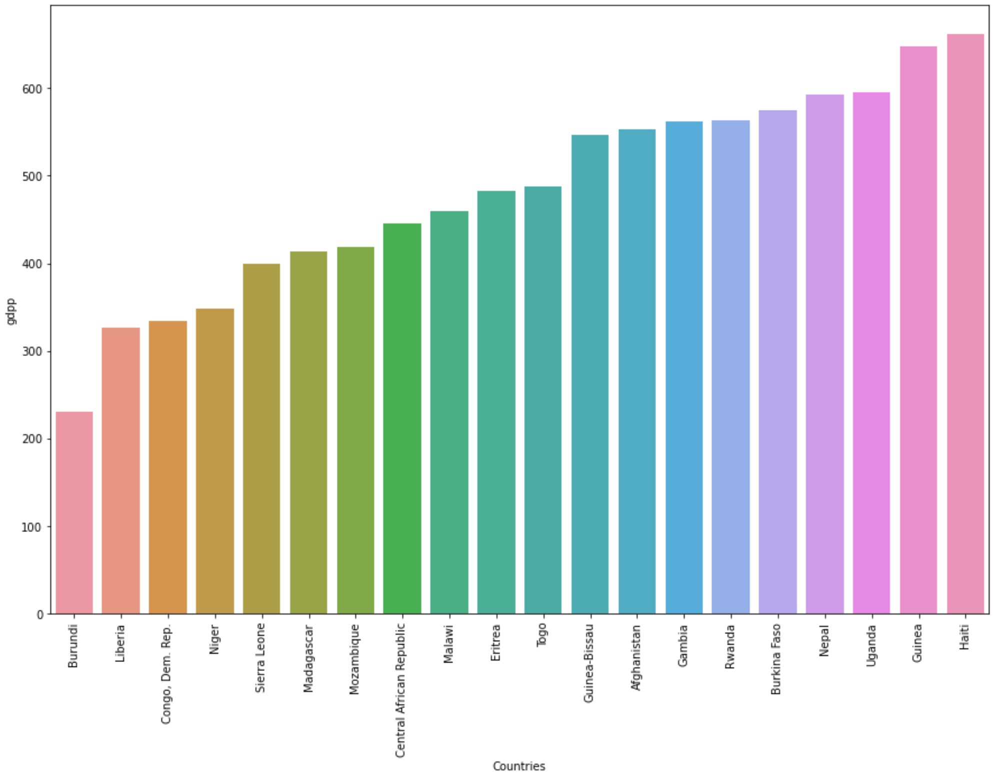
inflation
0
20
40
60
80
100
Nigeria
Venezuela
Mongolia
Timor-Leste
Equatorial Guinea
Yemen
Sri Lanka
Angola
Argentina
Congo, Dem. Rep.
Congo, Rep.
Sudan
Kazakhstan
Mauritania
Sierra Leone
Saudi Arabia
Brunei
Ghana
Iraq
Gabon
Countries



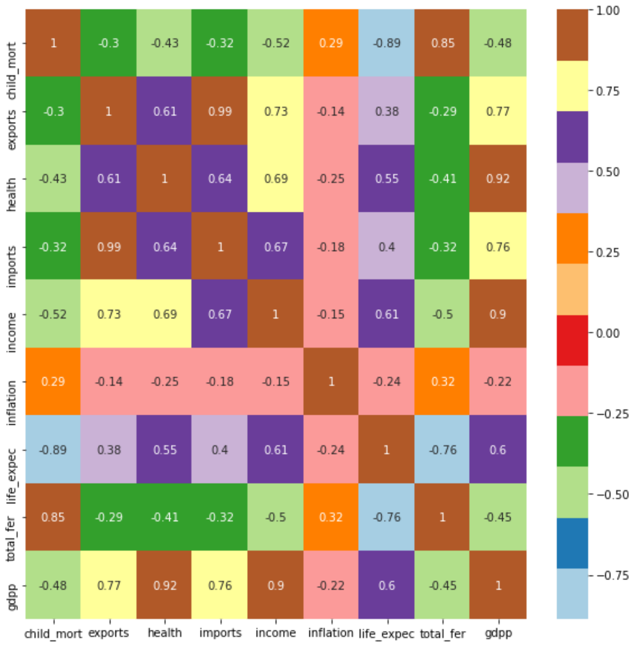
life_expec
Countries
Haiti
Lesotho
Central African Republic
Zambia
Malawi
South Africa
Mozambique
Sierra Leone
Guinea-Bissau
Afghanistan
Cote d'Ivoire
Chad
Uganda
Botswana
Cameroon
Congo, Dem. Rep.
Burundi
Burkina Faso
Guinea
Namibia



total_fer
Countries
Niger
Chad
Mali
Congo, Dem. Rep.
Burundi
Timor-Leste
Angola
Uganda
Burkina Faso
Nigeria
Afghanistan
Gambia
Mozambique
Tanzania
Zambia
Benin
Guinea
Malawi
Cote d'Ivoire
Equatorial Guinea



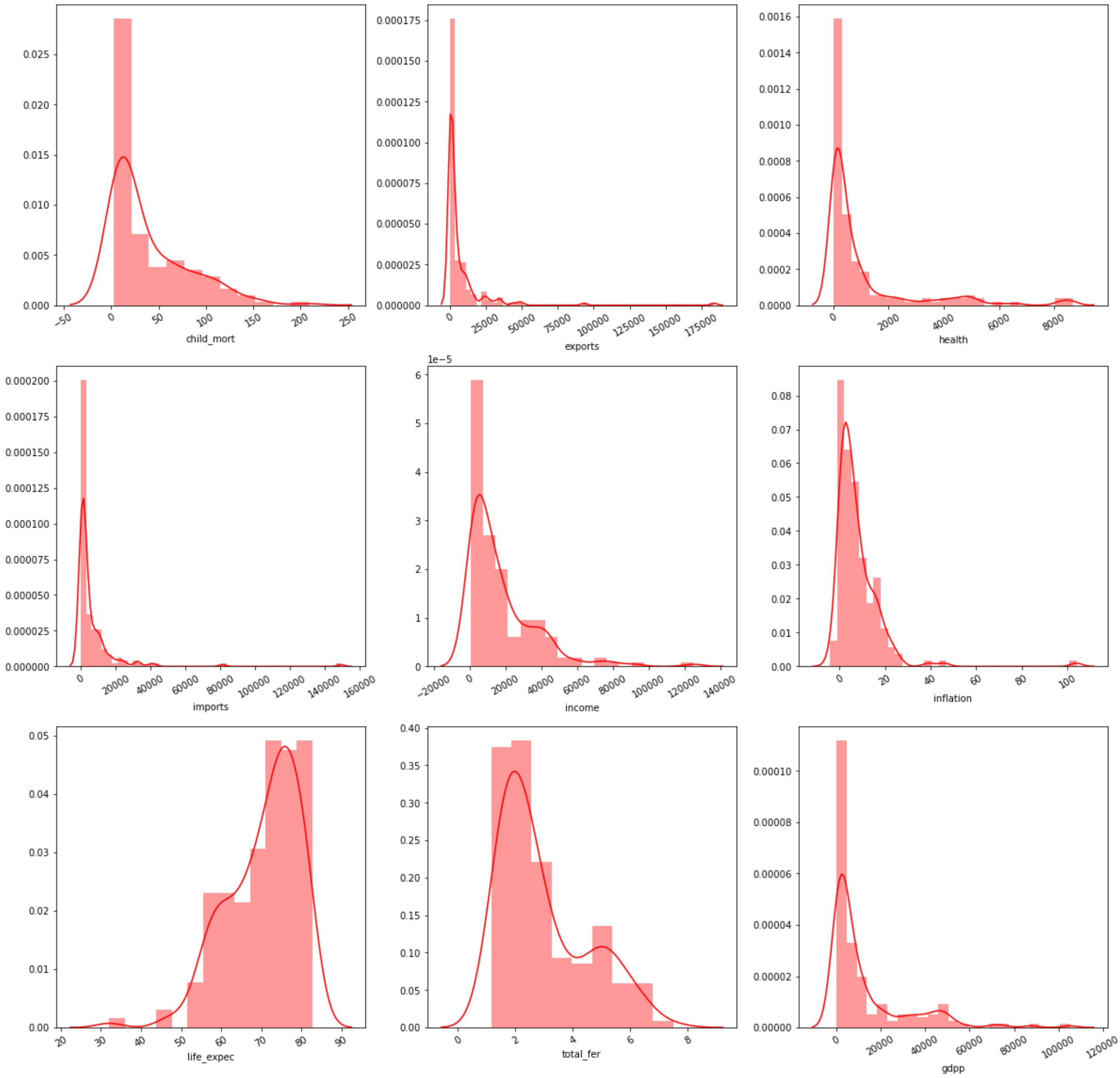
In [13]:
```
# Let's check the correlation coefficients to see which variables are highly correlated
plt.figure(figsize = (10, 10))
sns.heatmap(data.corr(), annot = True, cmap="Paired")
plt.show()
```



	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
child_mort	1	-0.3	-0.43	-0.32	-0.52	0.29	-0.89	0.85	-0.48
exports	-0.3	1	0.61	0.99	0.73	-0.14	0.38	-0.29	0.77
health	-0.43	0.61	1	0.64	0.69	-0.25	0.55	-0.41	0.92
imports	-0.32	0.99	0.64	1	0.67	-0.18	0.4	-0.32	0.76
income	-0.52	0.73	0.69	0.67	1	-0.15	0.61	-0.5	0.9
inflation	0.29	-0.14	-0.25	-0.18	-0.15	1	-0.24	0.32	-0.22
life_expec	-0.89	0.38	0.55	0.4	0.61	-0.24	1	-0.76	0.6
total_fer	0.85	-0.29	-0.41	-0.32	-0.5	0.32	-0.76	1	-0.45
gdpp	-0.48	0.77	0.92	0.76	0.9	-0.22	0.6	-0.45	1

- imports and gdpp are highly correlated with correlation of 0.76
- child_mortality and total_fertility are highly correlated with correlation of 0.85
- income and exports are highly correlated with correlation of 0.73

In [14]:
```python
plt.figure(figsize=(20, 20))

# Total we have 9 features, we show as 3x3 subplot
for i in enumerate(list_of_features):
    axis = plt.subplot(3, 3, i[0]+1)
    sns.distplot(data[i[1]], color = 'red')
    plt.xticks(rotation = 30)
```



In [15]:
```python
# Converting exports,imports and health spending percentages to absolute values.
data_numerical = data.copy()
data_numerical = data.drop(columns=['country'])

data_numerical.head() # Lets check data after conversion
```

Out[15]:

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

Hopkins Statistics Test

The Hopkins statistic is a way of measuring the cluster tendency of a data set.

A value close to 1 tends to indicate the data is highly clustered, random data will tend to result in values around 0.5, and uniformly distributed data will tend to result in values close to 0.

In [16]:
```
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).reshape(1, -1), 2, return_distance=True)
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors(X.iloc[rand_X[j]].values.reshape(1, -1), 2, return_distance=True)
        wjd.append(w_dist[0][1])

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

    return H
```

In [17]:
```
# Hopkins score
Hopkins_score = round(hopkins(data_numerical), 3)
print("{} is a good Hopkins score for Clustering.".format(Hopkins_score))
```

```
0.875 is a good Hopkins score for Clustering.
```

Data scaling

In [18]:
```
data_numerical.head(10)
```

Out[18]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	90.2	55.30	41.9174	248.297	1610	9.440	56.2	5.82	553
1	16.6	1145.20	267.8950	1987.740	9930	4.490	76.3	1.65	4090
2	27.3	1712.64	185.9820	1400.440	12900	16.100	76.5	2.89	4460
3	119.0	2199.19	100.6050	1514.370	5900	22.400	60.1	6.16	3530
4	10.3	5551.00	735.6600	7185.800	19100	1.440	76.8	2.13	12200
5	14.5	1946.70	834.3000	1648.000	18700	20.900	75.8	2.37	10300
6	18.1	669.76	141.6800	1458.660	6700	7.770	73.3	1.69	3220
7	4.8	10276.20	4530.8700	10847.100	41400	1.160	82.0	1.93	51900
8	4.3	24059.70	5159.0000	22418.200	43200	0.873	80.5	1.44	46900
9	39.2	3171.12	343.3920	1208.880	16000	13.800	69.1	1.92	5840

In [19]:
```
# Standarisation technique for scaling
scaler = StandardScaler()
data_rescaled = scaler.fit_transform(data_numerical)

data_rescaled = pd.DataFrame(data_rescaled, columns = list_of_features)
data_rescaled
```

Out[19]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	1.291532	-0.411011	-0.565040	-0.432276	-0.808245	0.157336	-1.619092	1.902882	-0.679180
1	-0.538949	-0.350191	-0.439218	-0.313677	-0.375369	-0.312347	0.647866	-0.859973	-0.485623
2	-0.272833	-0.318526	-0.484826	-0.353720	-0.220844	0.789274	0.670423	-0.038404	-0.465376
3	2.007808	-0.291375	-0.532363	-0.345953	-0.585043	1.387054	-1.179234	2.128151	-0.516268
4	-0.695634	-0.104331	-0.178771	0.040735	0.101732	-0.601749	0.704258	-0.541946	-0.041817
...	...	...	...	...	...	...	...	...	...
162	-0.225578	-0.336864	-0.501562	-0.342488	-0.738527	-0.489784	-0.852161	0.365754	-0.546913
163	-0.526514	-0.199393	-0.219310	-0.287205	-0.033542	3.616865	0.546361	-0.316678	0.029323
164	-0.372315	-0.361463	-0.538488	-0.377572	-0.658404	0.409732	0.286958	-0.661206	-0.637754
165	0.448417	-0.392166	-0.550596	-0.418479	-0.658924	1.500916	-0.344633	1.140944	-0.637754
166	1.114951	-0.383952	-0.540498	-0.418445	-0.721358	0.590015	-2.092785	1.624609	-0.629546

167 rows × 9 columns

K- means Clustering

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms.

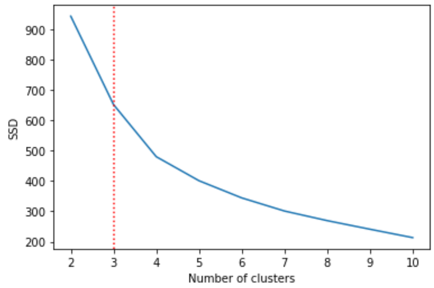
- First we initialize k points, called means, randomly.
- We categorize each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that mean so far.
- We repeat the process for a given number of iterations and at the end, we have our clusters.

Finding the Optimal Number of Clusters

Elbow Curve to get the right number of Clusters

A fundamental step for any unsupervised algorithm is to determine the optimal number of clusters into which the data may be clustered. The Elbow Method is one of the most popular methods to determine this optimal value of k.

In [20]:
```
# Elbow curve method / SSD
ssd = []
for k in range(2, 11):
    kmean = KMeans(n_clusters = k).fit(data_rescaled)
    ssd.append([k, kmean.inertia_])

temp = pd.DataFrame(ssd)
ax = plt.axes()
ax.plot(temp[0], temp[1]) # plot the SSDs for each n_clusters
ax.axvline(3, ls='dotted',color='red') # elbow formed as 3
plt.xlabel('Number of clusters')
plt.ylabel('SSD')
plt.show()
```

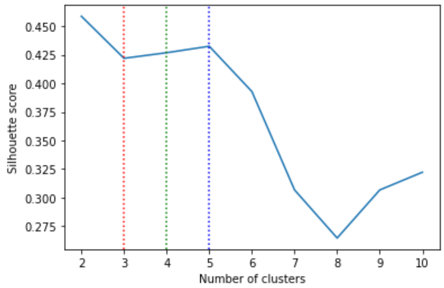


Looking at the above elbow curve it looks good to proceed with 3 clusters.

Silhouette Analysis

In [21]:
```
# Silhouette score
silhouette_scores_list = []
for k in range(2, 11):

    kmeans = KMeans(n_clusters = k, max_iter=50,random_state= 100)
    kmeans.fit(data_rescaled)

    silhouette_avg = silhouette_score(data_rescaled, kmeans.labels_) # silhouette score
    silhouette_scores_list.append([k, silhouette_avg])
    print("For k_clusters={0}, the silhouette score is {1:2f}".format(k, silhouette_avg))

temp = pd.DataFrame(silhouette_scores_list)

ax = plt.axes()
ax.plot(temp[0], temp[1])
ax.axvline(3, ls='dotted',color='red')
ax.axvline(4, ls='dotted',color='green')
ax.axvline(5, ls='dotted',color='blue')

plt.xlabel('Number of clusters')
plt.ylabel('Silhouette score')
plt.show()
```

```
For k_clusters=2, the silhouette score is 0.458633
For k_clusters=3, the silhouette score is 0.421862
For k_clusters=4, the silhouette score is 0.426734
For k_clusters=5, the silhouette score is 0.432400
For k_clusters=6, the silhouette score is 0.392794
For k_clusters=7, the silhouette score is 0.306822
For k_clusters=8, the silhouette score is 0.264748
For k_clusters=9, the silhouette score is 0.306805
For k_clusters=10, the silhouette score is 0.322283
```



From the above validations(Elbow Curve & silhouette analysis), we could see that 3, 4 or 5 clusters are optimal number of clusters to be used. We will try three different iterations in K-Means clustering using 3, 4 and 5 Clusters and analyse the results. We will not consider 2 clusters as it will not provide us our desierd results of country segregation.

Iterating with k= 3, 4 and 5

```
In [22]: # Function for all steps of Kmean Clustering; Call with K=3,4,5
def model_k_means(k):
    kmeans = KMeans(n_clusters = k, max_iter = 500, random_state = 300)
    kmeans.fit(data_rescaled)

    data_kmeans = data.copy() # copy the actual data into a new dataframe to explain the cluster profiling
    label  = pd.DataFrame(kmeans.labels_, columns= ['k_means_cluster_label'])
    data_kmeans = pd.concat([data_kmeans, label], axis =1) # assign the countries with the cluster labels.

    print("Number of countries in each cluster(k=%s):" %k)
    print(data_kmeans.k_means_cluster_label.value_counts(ascending = False))# shows how many countries are in each cluster
    return(data_kmeans) # returns clustered labelled dataset for further analysis
```

```
In [23]: # Created Models are available globally to access inside cluster profiling functions
k_3_model = model_k_means(3) # K means model with 3 clusters
k_4_model = model_k_means(4) # K means model with 4 clusters
k_5_model = model_k_means(5) # K means model with 5 clusters
```

```
Number of countries in each cluster(k=3):
0    91
2    48
1    28
Name: k_means_cluster_label, dtype: int64
Number of countries in each cluster(k=4):
0    87
2    48
3    30
1     2
Name: k_means_cluster_label, dtype: int64
Number of countries in each cluster(k=5):
0    88
2    47
1    30
4     1
3     1
Name: k_means_cluster_label, dtype: int64
```

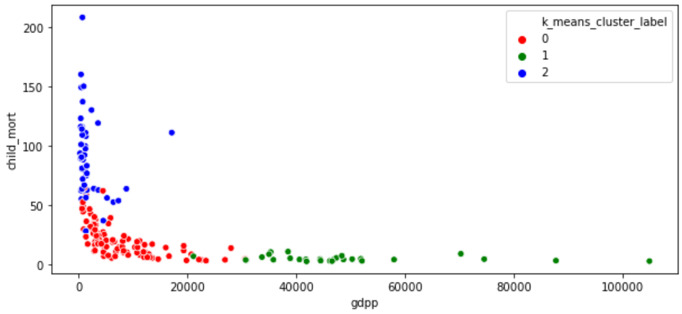
As the IMF and the UN, world is divided into 3 major classification for countries on scale of development.

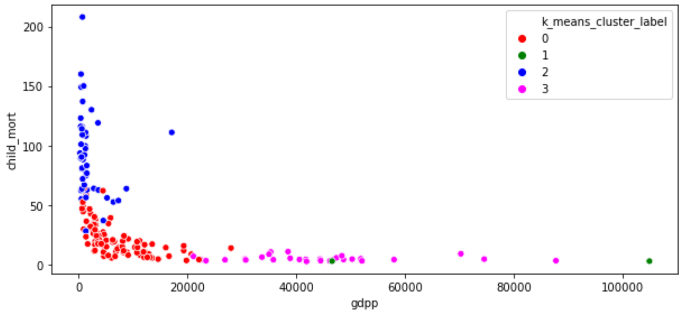
- Developed countries
- Developing countries
- Least developed countries

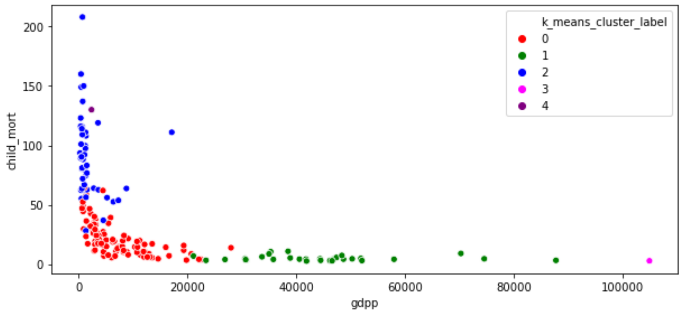
Cluster Profiling

```
In [24]: # Function for Profiling Clusters to plot scatter plots
def clusters_scatter_plots(column1, column2):
    plt.figure(figsize=(10,15))
    plt.subplot(3,1,1)
    sns.scatterplot(x = column1, y = column2, hue = 'k_means_cluster_label', data = k_3_model, palette=['red','green','blue'])
    plt.subplot(3,1,2)
    sns.scatterplot(x = column1, y = column2, hue = 'k_means_cluster_label', data = k_4_model, palette=['red','green','blue','magenta'])
    plt.subplot(3,1,3)
    sns.scatterplot(x = column1, y = column2, hue = 'k_means_cluster_label', data = k_5_model, palette=['red','green','blue','magenta','purple'])
```

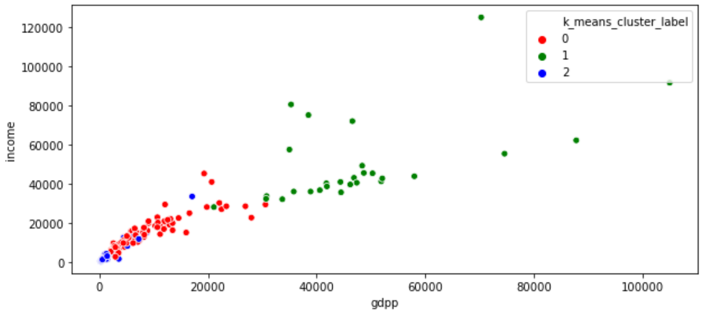
In [25]:
```
clusters_scatter_plots('gdpp','child_mort')
```

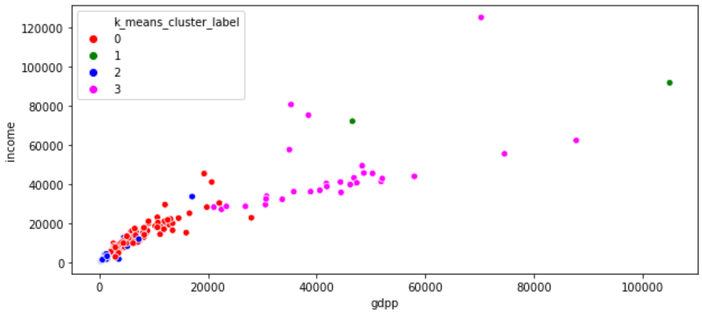


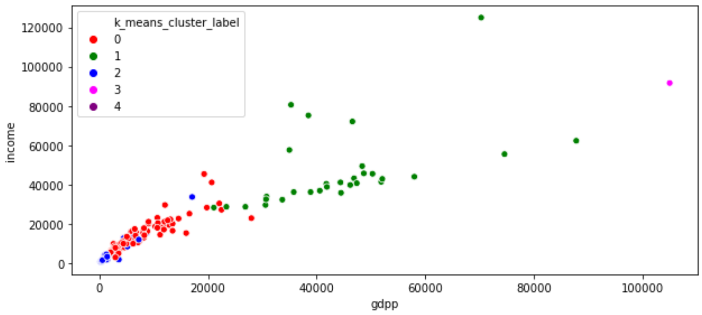




In [26]:
```
clusters_scatter_plots('gdpp','income')
```







Final Model: K-means clustering with K = 3

In [27]:
```
kmeans = KMeans(n_clusters = 3, random_state = 50)
kmeans.fit(data_rescaled)
```

Out[27]:
```
KMeans(n_clusters=3, random_state=50)
```

Creating Cluster labels using K-means

Since scaled data will be a bit confusing while explaining to business people, we will copy the actual data into a new dataframe to explain the cluster labels. We will use this `data_kmean` for cluster profiling. Lets create a column called `k_means_cluster_label` and concatenate to the `country_df_kmean_3` to assign the countries with the cluster labels.

In [28]:
```
data_kmeans = data.copy() # copy df into new df, as the same df will be used for hierarchical clustering too.
label = pd.DataFrame(kmeans.labels_, columns = ['k_means_cluster_label'])
label.head(10)
```

Out[28]:

	k_means_cluster_label
0	2
1	0
2	0
3	2
4	0
5	0
6	0
7	1
8	1
9	0

In [29]:
```
data_kmeans = pd.concat([data_kmeans, label], axis = 1)
data_kmeans.head(10)
```

Out[29]:

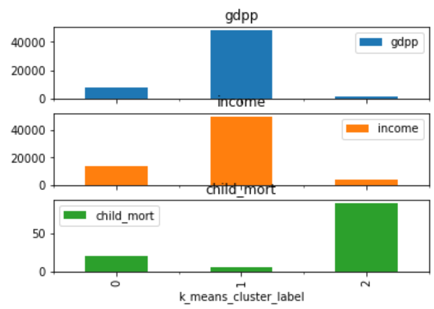
	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	k_means_cluster_label
0	Afghanistan	90.2	55.30	41.9174	248.297	1610	9.440	56.2	5.82	553	2
1	Albania	16.6	1145.20	267.8950	1987.740	9930	4.490	76.3	1.65	4090	0
2	Algeria	27.3	1712.64	185.9820	1400.440	12900	16.100	76.5	2.89	4460	0
3	Angola	119.0	2199.19	100.6050	1514.370	5900	22.400	60.1	6.16	3530	2
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100	1.440	76.8	2.13	12200	0
5	Argentina	14.5	1946.70	834.3000	1648.000	18700	20.900	75.8	2.37	10300	0
6	Armenia	18.1	669.76	141.6800	1458.660	6700	7.770	73.3	1.69	3220	0
7	Australia	4.8	10276.20	4530.8700	10847.100	41400	1.160	82.0	1.93	51900	1
8	Austria	4.3	24059.70	5159.0000	22418.200	43200	0.873	80.5	1.44	46900	1
9	Azerbaijan	39.2	3171.12	343.3920	1208.880	16000	13.800	69.1	1.92	5840	0

value_counts shows how many countries are clustered under each cluster label

In [30]:
```
data_kmeans.k_means_cluster_label.value_counts()
```

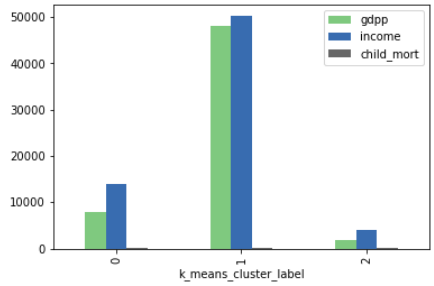
Out[30]:
```
0    91
2    48
1    28
Name: k_means_cluster_label, dtype: int64
```

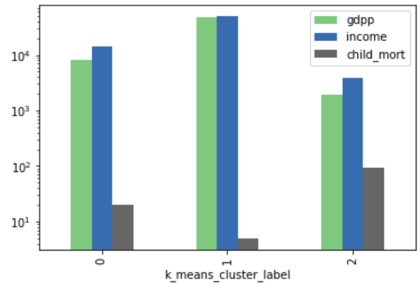
In [31]:
```
# Profiling GDP, INCOME AND CHID_MORT in separate plots
data_grouped = data_kmeans[['gdpp', 'income', 'child_mort','k_means_cluster_label']].groupby('k_means_cluster_label').mean()
axes = data_grouped.plot.bar(subplots = True)
plt.show()
```



In [32]:
```
# GDPP, INCOME AND CHID_MORT
data_grouped.plot(kind = 'bar', colormap = 'Accent')
data_grouped.plot(kind = 'bar', logy = True, colormap = 'Accent')
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x2496b4c1fa0>





From the above three plots, We can see that the clusters are grouped as

- 0 : Medium GDPP, medium Income and mild child mortality rate.
- 1 : High GDPP, High income and very low child mortality rate.
- 2 : Low GDPP, Low income and very high mortality rate.

4.5 Countries Segmentation

We can rename the labels for better business understanding as cluster label 0, 1 and 2 does not make sense to interpret. Then we will perform the cluster profiling with new labels.

Lets rename the cluster labesl as

- 0 : Developing Countries
- 1 : Developed Countries
- 2 : Under-developed Countries

This would help the NGO to recognise and differentiate the clusters of developed countries from the clusters of under-developed countries and focus on **Cluster 2: Under-developed Countries**

In [33]:
```
# Medium income, Medium GDP and Slightly high Child_mort
# Filter the data for that clsuter

data_kmeans.loc[data_kmeans['k_means_cluster_label'] == 0,'k_means_cluster_label'] ='Developing Countries'
data_kmeans[data_kmeans['k_means_cluster_label'] == 'Developing Countries']
```

Out[33]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	k_means_cluster_label
1	Albania	16.6	1145.20	267.895	1987.74	9930	4.49	76.3	1.65	4090	Developing Countries
2	Algeria	27.3	1712.64	185.982	1400.44	12900	16.10	76.5	2.89	4460	Developing Countries
4	Antigua and Barbuda	10.3	5551.00	735.660	7185.80	19100	1.44	76.8	2.13	12200	Developing Countries
5	Argentina	14.5	1946.70	834.300	1648.00	18700	20.90	75.8	2.37	10300	Developing Countries
6	Armenia	18.1	669.76	141.680	1458.66	6700	7.77	73.3	1.69	3220	Developing Countries
...	...	...	...	...	...	...	...	...	...	...	...
160	Uruguay	10.6	3129.70	993.650	3022.60	17100	4.91	76.4	2.08	11900	Developing Countries
161	Uzbekistan	36.3	437.46	80.178	393.30	4240	16.50	68.8	2.34	1380	Developing Countries
162	Vanuatu	29.2	1384.02	155.925	1565.19	2950	2.62	63.0	3.50	2970	Developing Countries
163	Venezuela	17.1	3847.50	662.850	2376.00	16500	45.90	75.4	2.47	13500	Developing Countries
164	Vietnam	23.3	943.20	89.604	1050.62	4490	12.10	73.1	1.95	1310	Developing Countries

91 rows × 11 columns

In [34]:
```
data_kmeans[data_kmeans['k_means_cluster_label'] == 'Developing Countries'].describe()
```

Out[34]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
count	91.000000	91.000000	91.000000	91.000000	91.000000	91.000000	91.000000	91.000000	91.000000
mean	20.357143	3604.149434	547.279455	3710.446386	13968.021978	7.070549	73.460440	2.235055	7979.912088
std	14.067394	3752.816160	575.140958	3450.055951	9326.576390	7.777400	4.024505	0.677301	6773.212754
min	3.200000	1.076920	19.463600	0.651092	1990.000000	-4.210000	63.000000	1.230000	592.000000
25%	10.300000	1000.575000	178.687500	1357.470000	7010.000000	1.755000	70.400000	1.600000	3015.000000
50%	17.200000	1806.920000	365.680000	2364.930000	11200.000000	5.140000	74.100000	2.170000	5450.000000
75%	26.300000	5064.300000	728.420000	5118.800000	18900.000000	9.395000	76.350000	2.645000	11250.000000
max	64.400000	15046.200000	2770.700000	14718.600000	45400.000000	45.900000	81.400000	4.340000	30600.000000

In [35]:
```
# Developed Countries: High income, High GDP and Low Child_mort
# Filter the data for that clsuter
data_kmeans.loc[data_kmeans['k_means_cluster_label'] == 1,'k_means_cluster_label'] ='Developed Countries'
data_kmeans[data_kmeans['k_means_cluster_label'] == 'Developed Countries']
```

Out[35]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	k_means_cluster_label
7	Australia	4.8	10276.2	4530.87	10847.1	41400	1.160	82.0	1.93	51900	Developed Countries
8	Austria	4.3	24059.7	5159.00	22418.2	43200	0.873	80.5	1.44	46900	Developed Countries
15	Belgium	4.5	33921.6	4750.80	33166.8	41100	1.880	80.0	1.86	44400	Developed Countries
23	Brunei	10.5	23792.2	1002.52	9884.0	80600	16.700	77.1	1.84	35300	Developed Countries
29	Canada	5.6	13793.4	5356.20	14694.0	40700	2.870	81.3	1.63	47400	Developed Countries
42	Cyprus	3.6	15461.6	1838.76	17710.0	33900	2.010	79.9	1.42	30800	Developed Countries
44	Denmark	4.1	29290.0	6612.00	25288.0	44000	3.220	79.5	1.87	58000	Developed Countries
53	Finland	3.0	17879.4	4134.90	17278.8	39800	0.351	80.0	1.87	46200	Developed Countries
54	France	4.2	10880.8	4831.40	11408.6	36900	1.050	81.4	2.03	40600	Developed Countries
58	Germany	4.2	17681.4	4848.80	15507.8	40400	0.758	80.1	1.39	41800	Developed Countries
68	Iceland	2.6	22374.6	3938.60	18142.7	38800	5.470	82.0	2.20	41900	Developed Countries
73	Ireland	4.2	50161.0	4475.53	42125.5	45700	-3.220	80.4	2.05	48700	Developed Countries
75	Italy	4.0	9021.6	3411.74	9737.6	36200	0.319	81.7	1.46	35800	Developed Countries
77	Japan	3.2	6675.0	4223.05	6052.0	35800	-1.900	82.8	1.39	44500	Developed Countries
82	Kuwait	10.8	25679.5	1012.55	11704.0	75200	11.200	78.2	2.21	38500	Developed Countries
91	Luxembourg	2.8	183750.0	8158.50	149100.0	91700	3.620	81.3	1.63	105000	Developed Countries
98	Malta	6.8	32283.0	1825.15	32494.0	28300	3.830	80.3	1.36	21100	Developed Countries
110	Netherlands	4.5	36216.0	5985.70	31990.8	45500	0.848	80.7	1.79	50300	Developed Countries
111	New Zealand	6.2	10211.1	3403.70	9436.0	32300	3.730	80.9	2.17	33700	Developed Countries
114	Norway	3.2	34856.6	8323.44	25023.0	62300	5.950	81.0	1.95	87800	Developed Countries
123	Qatar	9.0	43796.9	1272.43	16731.4	125000	6.980	79.5	2.07	70300	Developed Countries
133	Singapore	2.8	93200.0	1845.36	81084.0	72100	-0.046	82.7	1.15	46600	Developed Countries
139	Spain	3.8	7828.5	2928.78	8227.6	32500	0.160	81.9	1.37	30700	Developed Countries
144	Sweden	3.0	24070.2	5017.23	21204.7	42900	0.991	81.5	1.98	52100	Developed Countries
145	Switzerland	4.5	47744.0	8579.00	39761.8	55500	0.317	82.2	1.52	74600	Developed Countries
157	United Arab Emirates	8.6	27195.0	1281.00	22260.0	57600	12.500	76.5	1.87	35000	Developed Countries
158	United Kingdom	5.2	10969.8	3749.96	11981.2	36200	1.570	80.3	1.92	38900	Developed Countries
159	United States	7.3	6001.6	8663.60	7647.2	49400	1.220	78.7	1.93	48400	Developed Countries

In [36]:
```
data_kmeans[data_kmeans['k_means_cluster_label'] == 'Developed Countries'].describe()
```

Out[36]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
count	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000
mean	5.046429	31038.239286	4327.163214	25818.100000	50178.571429	3.014679	80.514286	1.760714	48114.285714
std	2.289183	35026.139893	2292.822837	28570.881329	21515.650757	4.353749	1.535437	0.297905	17756.891501
min	2.600000	6001.600000	1002.520000	6052.000000	28300.000000	-3.220000	76.500000	1.150000	21100.000000
25%	3.500000	10947.550000	2657.925000	11268.225000	36725.000000	0.656250	79.975000	1.455000	37825.000000
50%	4.250000	23925.950000	4349.290000	17494.400000	42150.000000	1.395000	80.600000	1.865000	45350.000000
75%	5.750000	34155.350000	5208.300000	26963.700000	56025.000000	3.755000	81.550000	1.957500	50700.000000
max	10.800000	183750.000000	8663.600000	149100.000000	125000.000000	16.700000	82.800000	2.210000	105000.000000

In [37]:

```
# Under-Developed Countries:Low income, Low GDP and High Child_mort
# Filter the data for that clsuter

data_kmeans.loc[data_kmeans['k_means_cluster_label'] == 2,'k_means_cluster_label'] ='Under-Developed Countries'
data_kmeans[data_kmeans['k_means_cluster_label'] == 'Under-Developed Countries']
```

Out[37]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	k_means_cluster_label
0	Afghanistan	90.2	55.3000	41.9174	248.297	1610	9.440	56.2	5.82	553	Under-Developed Countries
3	Angola	119.0	2199.1900	100.6050	1514.370	5900	22.400	60.1	6.16	3530	Under-Developed Countries
17	Benin	111.0	180.4040	31.0780	281.976	1820	0.885	61.8	5.36	758	Under-Developed Countries
21	Botswana	52.5	2768.6000	527.0500	3257.550	13300	8.920	57.1	2.88	6350	Under-Developed Countries
25	Burkina Faso	116.0	110.4000	38.7550	170.200	1430	6.810	57.9	5.87	575	Under-Developed Countries
26	Burundi	93.6	20.6052	26.7960	90.552	764	12.300	57.7	6.26	231	Under-Developed Countries
28	Cameroon	108.0	290.8200	67.2030	353.700	2660	1.910	57.3	5.11	1310	Under-Developed Countries
31	Central African Republic	149.0	52.6280	17.7508	118.190	888	2.010	47.5	5.21	446	Under-Developed Countries
32	Chad	150.0	330.0960	40.6341	390.195	1930	6.390	56.5	6.59	897	Under-Developed Countries
36	Comoros	88.2	126.8850	34.6819	397.573	1410	3.870	65.9	4.75	769	Under-Developed Countries
37	Congo, Dem. Rep.	116.0	137.2740	26.4194	165.664	609	20.800	57.5	6.54	334	Under-Developed Countries
38	Congo, Rep.	63.9	2331.7400	67.4040	1498.780	5190	20.700	60.4	4.95	2740	Under-Developed Countries
40	Cote d'Ivoire	111.0	617.3200	64.6600	528.260	2690	5.390	56.3	5.27	1220	Under-Developed Countries
49	Equatorial Guinea	111.0	14671.8000	766.0800	10071.900	33700	24.900	60.9	5.21	17100	Under-Developed Countries
50	Eritrea	55.2	23.0878	12.8212	112.306	1420	11.600	61.7	4.61	482	Under-Developed Countries
55	Gabon	63.7	5048.7500	306.2500	1653.750	15400	16.600	62.9	4.08	8750	Under-Developed Countries
56	Gambia	80.3	133.7560	31.9778	239.974	1660	4.300	65.5	5.71	562	Under-Developed Countries
59	Ghana	74.7	386.4500	68.3820	601.290	3060	16.600	62.2	4.27	1310	Under-Developed Countries
63	Guinea	109.0	196.3440	31.9464	279.936	1190	16.100	58.0	5.34	648	Under-Developed Countries
64	Guinea-Bissau	114.0	81.5030	46.4950	192.544	1390	2.970	55.6	5.05	547	Under-Developed Countries
66	Haiti	208.0	101.2860	45.7442	428.314	1500	5.450	32.1	3.33	662	Under-Developed Countries
72	Iraq	36.9	1773.0000	378.4500	1534.500	12700	16.600	67.2	4.56	4500	Under-Developed Countries
80	Kenya	62.2	200.1690	45.9325	324.912	2480	2.090	62.8	4.37	967	Under-Developed Countries
81	Kiribati	62.7	198.1700	168.3700	1190.510	1730	1.520	60.7	3.84	1490	Under-Developed Countries
84	Lao	78.9	403.5600	50.9580	562.020	3980	9.200	63.8	3.15	1140	Under-Developed Countries
87	Lesotho	99.7	460.9800	129.8700	1181.700	2380	4.150	46.5	3.30	1170	Under-Developed Countries
88	Liberia	89.3	62.4570	38.5860	302.802	700	5.470	60.8	5.02	327	Under-Developed Countries
93	Madagascar	62.2	103.2500	15.5701	177.590	1390	8.790	60.8	4.60	413	Under-Developed Countries
94	Malawi	90.5	104.6520	30.2481	160.191	1030	12.100	53.1	5.31	459	Under-Developed Countries
97	Mali	137.0	161.4240	35.2584	248.508	1870	4.370	59.5	6.55	708	Under-Developed Countries
99	Mauritania	97.4	608.4000	52.9200	734.400	3320	18.900	68.2	4.98	1200	Under-Developed Countries
106	Mozambique	101.0	131.9850	21.8299	193.578	918	7.640	54.5	5.56	419	Under-Developed Countries
108	Namibia	56.0	2480.8200	351.8820	3150.330	8460	3.560	58.6	3.60	5190	Under-Developed Countries
112	Niger	123.0	77.2560	17.9568	170.868	814	2.550	58.8	7.49	348	Under-Developed Countries
113	Nigeria	130.0	589.4900	118.1310	405.420	5150	104.000	60.5	5.84	2330	Under-Developed Countries
116	Pakistan	92.1	140.4000	22.8800	201.760	4280	10.900	65.3	3.85	1040	Under-Developed Countries
126	Rwanda	63.6	67.5600	59.1150	168.900	1350	2.610	64.6	4.51	563	Under-Developed Countries
129	Senegal	66.8	249.0000	56.6000	403.000	2180	1.850	64.0	5.06	1000	Under-Developed Countries
132	Sierra Leone	160.0	67.0320	52.2690	137.655	1220	17.200	55.0	5.20	399	Under-Developed Countries
136	Solomon Islands	28.1	635.9700	110.2950	1047.480	1780	6.810	61.7	4.24	1290	Under-Developed Countries
137	South Africa	53.7	2082.0800	650.8320	1994.720	12000	6.350	54.3	2.59	7280	Under-Developed Countries
142	Sudan	76.7	291.5600	93.5360	254.560	3370	19.600	66.3	4.88	1480	Under-Developed Countries
147	Tanzania	71.9	131.2740	42.1902	204.282	2090	9.250	59.3	5.43	702	Under-Developed Countries
149	Timor-Leste	62.6	79.2000	328.3200	1000.800	1850	26.500	71.1	6.23	3600	Under-Developed Countries
150	Togo	90.3	196.1760	37.3320	279.624	1210	1.180	58.7	4.87	488	Under-Developed Countries
155	Uganda	81.0	101.7450	53.6095	170.170	1540	10.600	56.8	6.15	595	Under-Developed Countries
165	Yemen	56.3	393.0000	67.8580	450.640	4480	23.600	67.5	4.67	1310	Under-Developed Countries
166	Zambia	83.1	540.2000	85.9940	451.140	3280	14.000	52.0	5.40	1460	Under-Developed Countries

In [38]:

```
data_kmeans[data_kmeans['k_means_cluster_label'] == 'Under-Developed Countries'].describe()
```

Out[38]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
count	48.000000	48.000000	48.000000	48.000000	48.000000	48.000000	48.000000	48.000000	48.000000
mean	91.610417	879.063521	114.821765	827.028771	3897.354167	11.911146	59.239583	4.992083	1909.208333
std	34.319855	2252.474004	165.518331	1540.981910	5590.168621	15.362485	6.384914	1.036192	2925.911009
min	28.100000	20.605200	12.821200	90.552000	609.000000	0.885000	32.100000	2.590000	231.000000
25%	63.675000	102.873750	34.005875	193.319500	1390.000000	4.080000	56.725000	4.475000	551.500000
50%	89.750000	196.260000	51.613500	339.306000	1860.000000	8.855000	59.800000	5.055000	932.000000
75%	111.000000	552.522500	95.303250	801.000000	3522.500000	16.600000	62.825000	5.597500	1465.000000
max	208.000000	14671.800000	766.080000	10071.900000	33700.000000	104.000000	71.100000	7.490000	17100.000000

Summary statistics show that the variation within the group is very less and mean and median are so close. So this clustering is good.

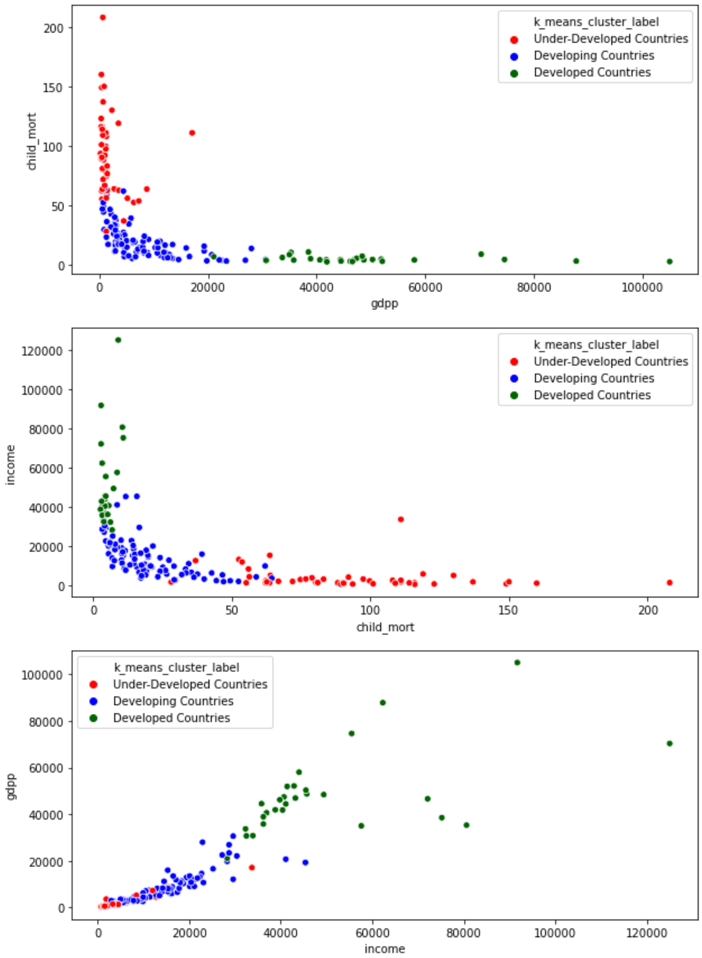
Also the stats of this cluster and the previous clusters has wide difference.

Hence cohesion and separation are well preserved in this clustering

Cluster profiling with new labels

In [39]:
```
profiling_cols = ['gdpp','child_mort','income'] # create a list to store profiling variables
```

In [40]:
```
# Plot the cluster
plt.figure(figsize=(10,15))
i=0
for i in range(len(profiling_cols)):
    plt.subplot(3,1,i+1)
    sns.scatterplot(x = profiling_cols[i], y = profiling_cols[(i+1)%len(profiling_cols)], hue = 'k_means_cluster_label', data = data_kmeans, palette
=['red','blue','darkgreen'])
```



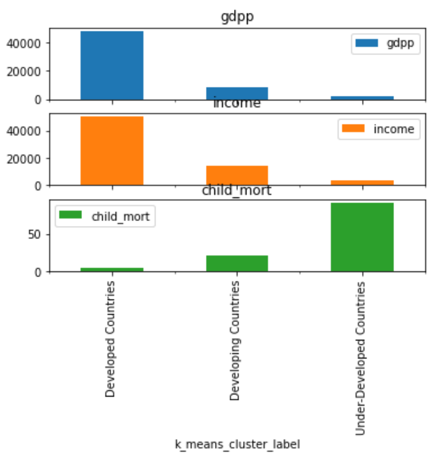
The similar observation that we got from scatter plots can be seen in boxplots too. The clusters are grouped as

- Developing countries have Medium GDPP, medium Income and mild child mortality rate.
- Developed countries have High GDPP, High income and very low child mortality rate.
- Under-Developed countries have Low GDPP, Low income and very high mortality rate and should be our primary focus.

We can see that GDPP and income of the under-developed countries are so low that they are not seen properly in the same scale that of the developed countries.

In [41]:
```
# Profiling GDP, INCOME AND CHID_MORT in sub-plots
plt.figure(figsize=(18,8))
data_grouped = data_kmeans[['gdpp', 'income', 'child_mort','k_means_cluster_label']].groupby('k_means_cluster_label').mean()
axes = data_grouped.plot.bar(subplots=True)
plt.show()
```

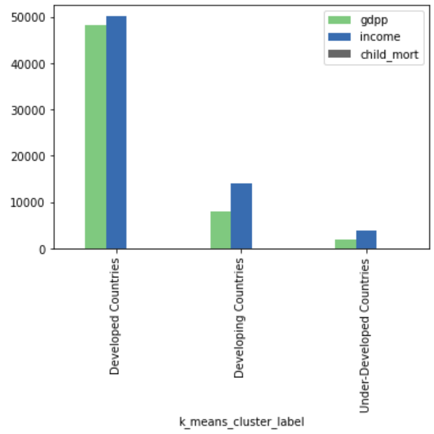
```
<Figure size 1296x576 with 0 Axes>
```

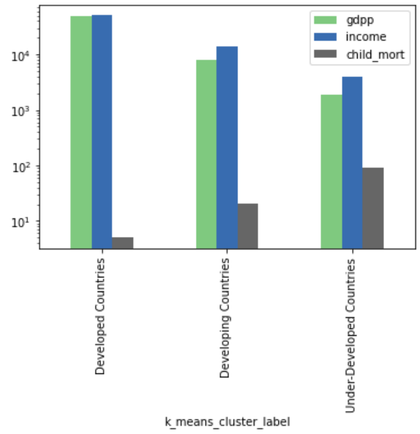


We can see the mean of the gdpp and income of the under-developed countries are so low when compared to developing or developed countries and we need to look further into this cluster to get the countries which are in most need of aid.

In [42]:
```
# Profiling GDP, INCOME AND CHID_MORT together from the above grouped_df
data_grouped.plot(kind = 'bar', colormap = 'Accent')
data_grouped.plot(kind = 'bar', logy = True, colormap = 'Accent')
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x2496b56dbb0>





The mean of each cluster show the similar observation and the grouping is done perfectly such that we can focus on cluster **Under-Developed Countries** as it has **Low GDPP, Low income and very high mortality rate.**

Identification of Top 20 countries that require aid on priority using K-means algorithm:

In [43]:
```
data_kmeans_top_20 = data_kmeans[data_kmeans['k_means_cluster_label'] =='Under-Developed Countries'].sort_values(['gdpp', 'child_mort', 'income'], a
scending = [True, False, True]).head(20)
data_kmeans_top_20
```

Out[43]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	k_means_cluster_label
26	Burundi	93.6	20.6052	26.7960	90.552	764	12.30	57.7	6.26	231	Under-Developed Countries
88	Liberia	89.3	62.4570	38.5860	302.802	700	5.47	60.8	5.02	327	Under-Developed Countries
37	Congo, Dem. Rep.	116.0	137.2740	26.4194	165.664	609	20.80	57.5	6.54	334	Under-Developed Countries
112	Niger	123.0	77.2560	17.9568	170.868	814	2.55	58.8	7.49	348	Under-Developed Countries
132	Sierra Leone	160.0	67.0320	52.2690	137.655	1220	17.20	55.0	5.20	399	Under-Developed Countries
93	Madagascar	62.2	103.2500	15.5701	177.590	1390	8.79	60.8	4.60	413	Under-Developed Countries
106	Mozambique	101.0	131.9850	21.8299	193.578	918	7.64	54.5	5.56	419	Under-Developed Countries
31	Central African Republic	149.0	52.6280	17.7508	118.190	888	2.01	47.5	5.21	446	Under-Developed Countries
94	Malawi	90.5	104.6520	30.2481	160.191	1030	12.10	53.1	5.31	459	Under-Developed Countries
50	Eritrea	55.2	23.0878	12.8212	112.306	1420	11.60	61.7	4.61	482	Under-Developed Countries
150	Togo	90.3	196.1760	37.3320	279.624	1210	1.18	58.7	4.87	488	Under-Developed Countries
64	Guinea-Bissau	114.0	81.5030	46.4950	192.544	1390	2.97	55.6	5.05	547	Under-Developed Countries
0	Afghanistan	90.2	55.3000	41.9174	248.297	1610	9.44	56.2	5.82	553	Under-Developed Countries
56	Gambia	80.3	133.7560	31.9778	239.974	1660	4.30	65.5	5.71	562	Under-Developed Countries
126	Rwanda	63.6	67.5600	59.1150	168.900	1350	2.61	64.6	4.51	563	Under-Developed Countries
25	Burkina Faso	116.0	110.4000	38.7550	170.200	1430	6.81	57.9	5.87	575	Under-Developed Countries
155	Uganda	81.0	101.7450	53.6095	170.170	1540	10.60	56.8	6.15	595	Under-Developed Countries
63	Guinea	109.0	196.3440	31.9464	279.936	1190	16.10	58.0	5.34	648	Under-Developed Countries
66	Haiti	208.0	101.2860	45.7442	428.314	1500	5.45	32.1	3.33	662	Under-Developed Countries
147	Tanzania	71.9	131.2740	42.1902	204.282	2090	9.25	59.3	5.43	702	Under-Developed Countries

```
In [44]: data_kmeans_top_20.country
```

```
Out[44]: 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
25                Burkina Faso
155                     Uganda
63                      Guinea
66                       Haiti
147                   Tanzania
Name: country, dtype: object
```

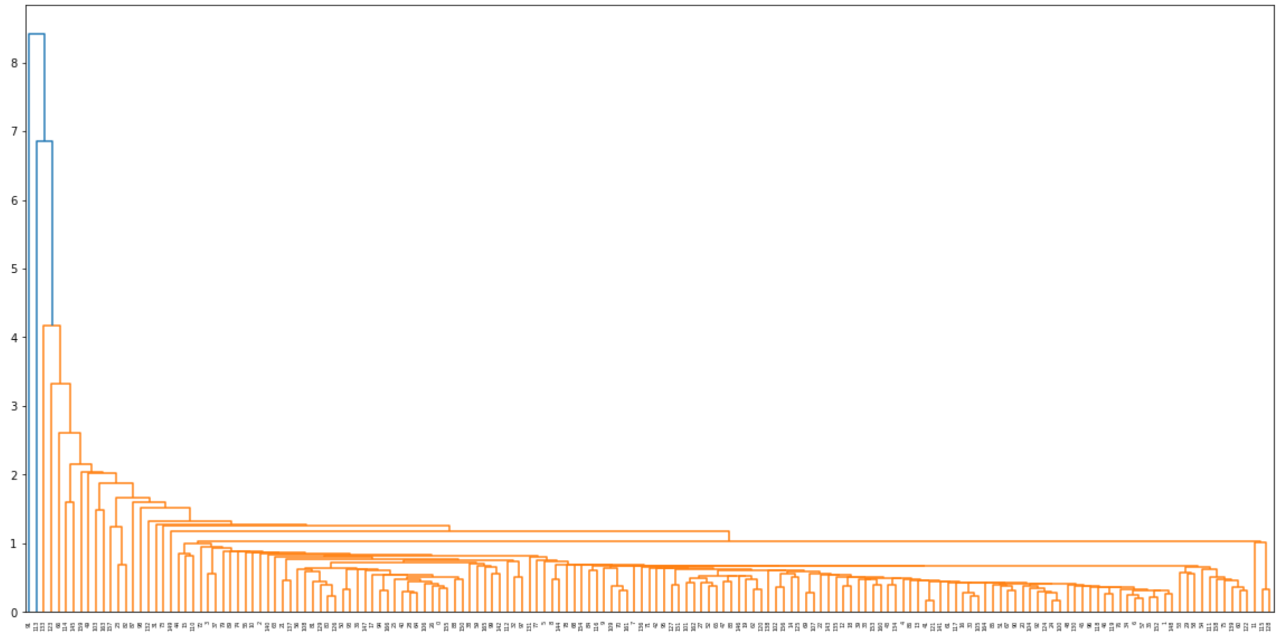
Hierarchical Clustering

Hierarchical clustering involves creating clusters that have a predetermined ordering from top to bottom. For example, all files and folders on the hard disk are organized in a hierarchy. There are two types of hierarchical clustering,

- Divisive
- Agglomerative.

Single Linkage

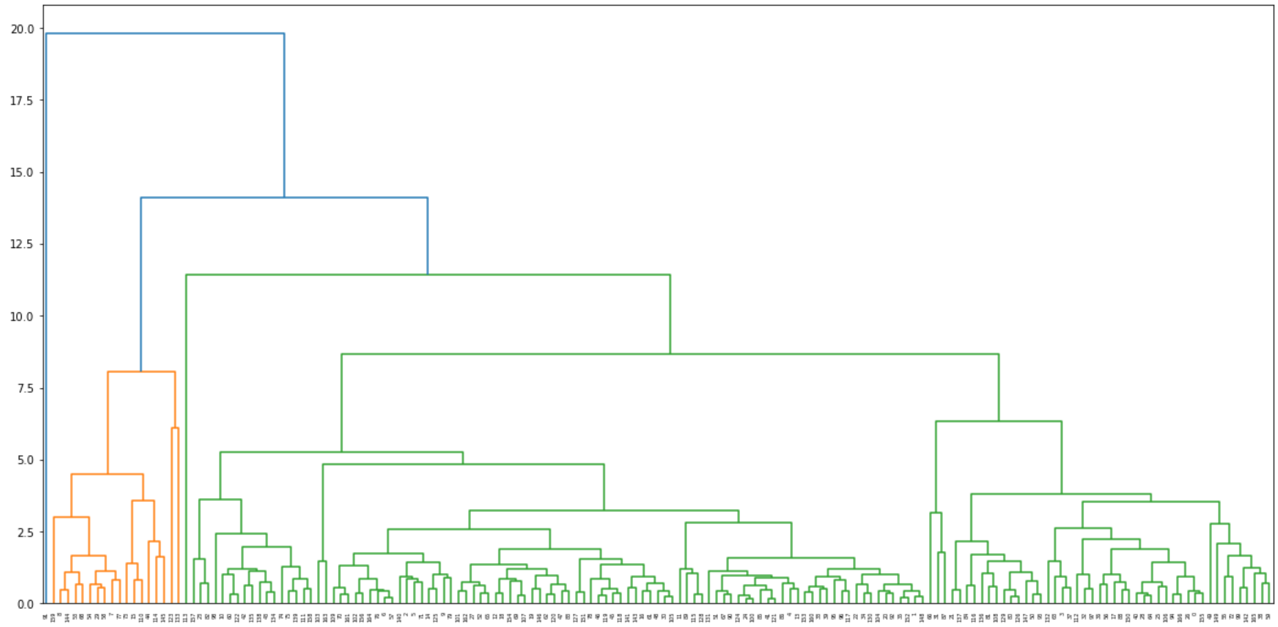
```
In [45]: # Single Linkage
plt.figure(figsize = (20,10))
mergings_single = linkage(data_rescaled, method = 'single', metric = 'euclidean')
dendrogram(mergings_single)
plt.show()
```



Single linkage's dendogram is not readable or interpretable. Hence we cannot use this for our problem.

Complete Linkage

In [46]:
```
# Complete Linkage
plt.figure(figsize = (20,10))
mergings_complete = linkage(data_rescaled, method = 'complete', metric = 'euclidean')
dendrogram(mergings_complete)
plt.show()
```



Complete linkage's dendogram is readable and better to interpret when compared to single linkage's dendogram.

We can see merging of clusters represented in different colors.

This indicates 3 clusters is a good choice as there will be good dissimilarity between clusters and good similarity within clusters.

In [47]:
```
data_hierarchical = data_numerical.copy()
data_hierarchical.head()
```

Out[47]:

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

As the IMF and the UN, world is divided into 3 major classification for countries on scale of development.

- Developed countries
- Developing countries
- Least developed countries

In [48]:
```
# 3 clusters
cluster_labels = cut_tree(mergings_complete, n_clusters = 3).reshape(-1,)
cluster_labels
```

Out[48]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 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,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 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,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0])
```

In [49]:
```
# Assign cluster labels
data_hierarchical['cluster_labels'] = cluster_labels
data_hierarchical.head()
```

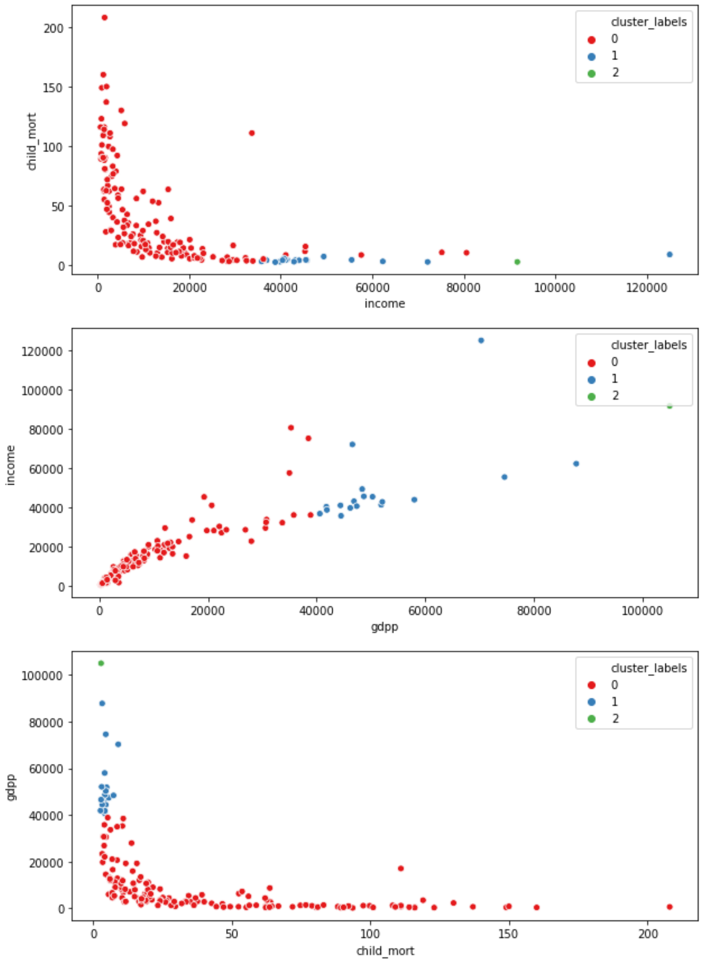
Out[49]:

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

In [50]:
```
# Number of countries in each cluster
data_hierarchical.cluster_labels.value_counts(ascending = False)
```

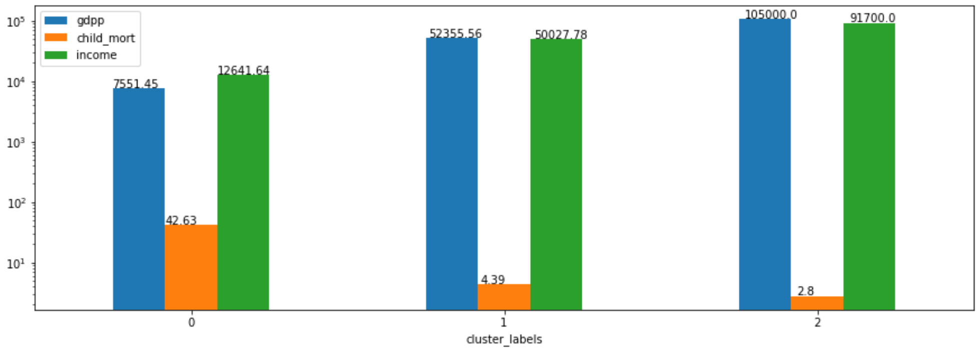
Out[50]:
```
0    148
1     18
2      1
Name: cluster_labels, dtype: int64
```

In [51]:
```python
# Scatter plot on Original attributes to visualize the spread of the data
plt.figure(figsize = (10, 15))
plt.subplot(3,1,1)
sns.scatterplot(x = 'income', y = 'child_mort',hue='cluster_labels',data = data_hierarchical,legend='full',palette="Set1")
plt.subplot(3,1,2)
sns.scatterplot(x = 'gdpp', y = 'income',hue='cluster_labels', data = data_hierarchical,legend='full',palette="Set1")
plt.subplot(3,1,3)
sns.scatterplot(x = 'child_mort', y = 'gdpp',hue='cluster_labels', data = data_hierarchical,legend='full',palette="Set1")
plt.show()
```



Cluster Profiling

In [52]:
```python
axis = data_hierarchical[['gdpp','child_mort','income','cluster_labels']].groupby('cluster_labels').mean().plot(kind = 'bar',figsize = (15,5))

for p in axis.patches:
    axis.annotate(str(round(p.get_height(),2)), (p.get_x() * 1.01 , p.get_height() * 1.01))
plt.yscale('log')
plt.xticks(rotation = 0)
plt.show();
```

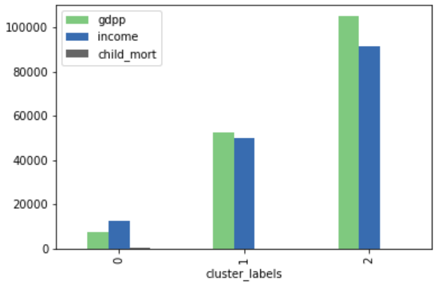


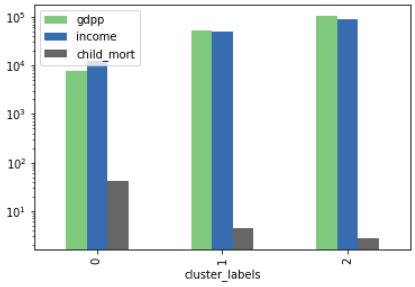
In [53]:
```python
data_hierarchical[data_hierarchical['cluster_labels'] == 1].sort_values(by = ['child_mort','income','gdpp',], ascending = [False, True, True]).head
()
# They are Developed countries as per UN & IMF
```

Out[53]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster_labels
123	9.0	43796.9	1272.43	16731.4	125000	6.98	79.5	2.07	70300	1
159	7.3	6001.6	8663.60	7647.2	49400	1.22	78.7	1.93	48400	1
29	5.6	13793.4	5356.20	14694.0	40700	2.87	81.3	1.63	47400	1
7	4.8	10276.2	4530.87	10847.1	41400	1.16	82.0	1.93	51900	1
15	4.5	33921.6	4750.80	33166.8	41100	1.88	80.0	1.86	44400	1

In [54]:
```python
# Profiling GDP, INCOME AND CHID_MORT in separete plots
data_grouped = data_hierarchical[['gdpp', 'income', 'child_mort','cluster_labels']].groupby('cluster_labels').mean()
data_grouped.plot(kind='bar', colormap='Accent')
data_grouped.plot(kind='bar',logy=True, colormap='Accent')

plt.show()
```





Countries Segmentation

Similar to our approach in K-mean algorithm, We can rename the labels for better business understanding as cluster label 0, 1 and 2 does not make sense to interpret. Then we will perform the cluster profiling with new labels.

Lets rename the cluster labels as

- 0 : Under-developed Countries
- 1 : Developing Countries
- 2 : Developed Countries

In [55]:
```python
# Low income, Low GDP and High Child_mort
# Filter the data for that clsuter
data_hierarchical.insert(0, 'country', data.country)

data_hierarchical.loc[data_hierarchical['cluster_labels'] == 0, 'cluster_labels'] = 'Under-Developed Countries'
data_hierarchical[data_hierarchical['cluster_labels'] == 'Under-Developed Countries']
```

Out[55]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster_labels
0	Afghanistan	90.2	55.30	41.9174	248.297	1610	9.44	56.2	5.82	553	Under-Developed Countries
1	Albania	16.6	1145.20	267.8950	1987.740	9930	4.49	76.3	1.65	4090	Under-Developed Countries
2	Algeria	27.3	1712.64	185.9820	1400.440	12900	16.10	76.5	2.89	4460	Under-Developed Countries
3	Angola	119.0	2199.19	100.6050	1514.370	5900	22.40	60.1	6.16	3530	Under-Developed Countries
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100	1.44	76.8	2.13	12200	Under-Developed Countries
...	...	...	...	...	...	...	...	...	...	...	...
162	Vanuatu	29.2	1384.02	155.9250	1565.190	2950	2.62	63.0	3.50	2970	Under-Developed Countries
163	Venezuela	17.1	3847.50	662.8500	2376.000	16500	45.90	75.4	2.47	13500	Under-Developed Countries
164	Vietnam	23.3	943.20	89.6040	1050.620	4490	12.10	73.1	1.95	1310	Under-Developed Countries
165	Yemen	56.3	393.00	67.8580	450.640	4480	23.60	67.5	4.67	1310	Under-Developed Countries
166	Zambia	83.1	540.20	85.9940	451.140	3280	14.00	52.0	5.40	1460	Under-Developed Countries

148 rows × 11 columns

In [56]:
```python
# Medium income, Medium GDP and Mild Child_mort
# Filter the data for that clsuter
data_hierarchical.loc[data_hierarchical['cluster_labels'] == 1,'cluster_labels'] = 'Developing Countries'
data_hierarchical[data_hierarchical['cluster_labels'] == 'Developing Countries']
```

Out[56]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster_labels
7	Australia	4.8	10276.2	4530.87	10847.1	41400	1.160	82.0	1.93	51900	Developing Countries
8	Austria	4.3	24059.7	5159.00	22418.2	43200	0.873	80.5	1.44	46900	Developing Countries
15	Belgium	4.5	33921.6	4750.80	33166.8	41100	1.880	80.0	1.86	44400	Developing Countries
29	Canada	5.6	13793.4	5356.20	14694.0	40700	2.870	81.3	1.63	47400	Developing Countries
44	Denmark	4.1	29290.0	6612.00	25288.0	44000	3.220	79.5	1.87	58000	Developing Countries
53	Finland	3.0	17879.4	4134.90	17278.8	39800	0.351	80.0	1.87	46200	Developing Countries
54	France	4.2	10880.8	4831.40	11408.6	36900	1.050	81.4	2.03	40600	Developing Countries
58	Germany	4.2	17681.4	4848.80	15507.8	40400	0.758	80.1	1.39	41800	Developing Countries
68	Iceland	2.6	22374.6	3938.60	18142.7	38800	5.470	82.0	2.20	41900	Developing Countries
73	Ireland	4.2	50161.0	4475.53	42125.5	45700	-3.220	80.4	2.05	48700	Developing Countries
77	Japan	3.2	6675.0	4223.05	6052.0	35800	-1.900	82.8	1.39	44500	Developing Countries
110	Netherlands	4.5	36216.0	5985.70	31990.8	45500	0.848	80.7	1.79	50300	Developing Countries
114	Norway	3.2	34856.6	8323.44	25023.0	62300	5.950	81.0	1.95	87800	Developing Countries
123	Qatar	9.0	43796.9	1272.43	16731.4	125000	6.980	79.5	2.07	70300	Developing Countries
133	Singapore	2.8	93200.0	1845.36	81084.0	72100	-0.046	82.7	1.15	46600	Developing Countries
144	Sweden	3.0	24070.2	5017.23	21204.7	42900	0.991	81.5	1.98	52100	Developing Countries
145	Switzerland	4.5	47744.0	8579.00	39761.8	55500	0.317	82.2	1.52	74600	Developing Countries
159	United States	7.3	6001.6	8663.60	7647.2	49400	1.220	78.7	1.93	48400	Developing Countries

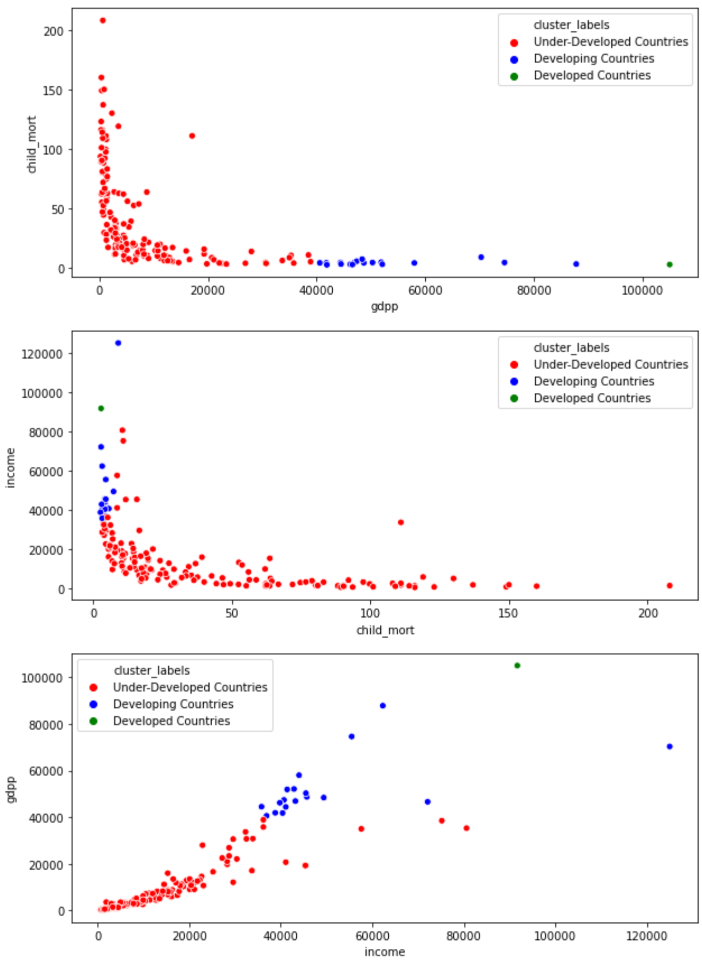
In [57]:
```python
# High income, High GDP and Low Child_mort
# Filter the data for that clsuter
data_hierarchical.loc[data_hierarchical['cluster_labels'] == 2,'cluster_labels'] ='Developed Countries'
data_hierarchical[data_hierarchical['cluster_labels'] == 'Developed Countries']
```

Out[57]:

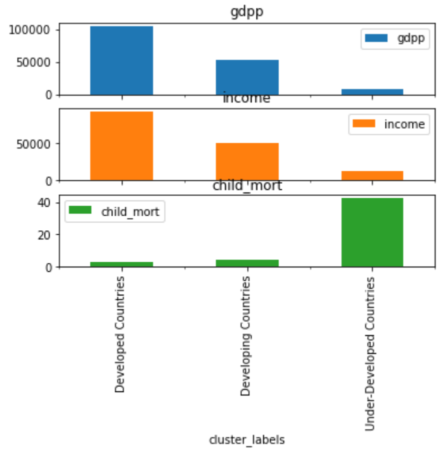
	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster_labels
91	Luxembourg	2.8	183750.0	8158.5	149100.0	91700	3.62	81.3	1.63	105000	Developed Countries

Cluster profiling with new labels

In [58]:
```python
# Plot the cluster
plt.figure(figsize=(10,15))
i=0
for i in range(len(profiling_cols)):
    plt.subplot(3,1,i+1)
    sns.scatterplot(x = profiling_cols[i], y = profiling_cols[(i+1)%len(profiling_cols)], hue = 'cluster_labels', data = data_hierarchical, palette=
['red','blue','green'])
```

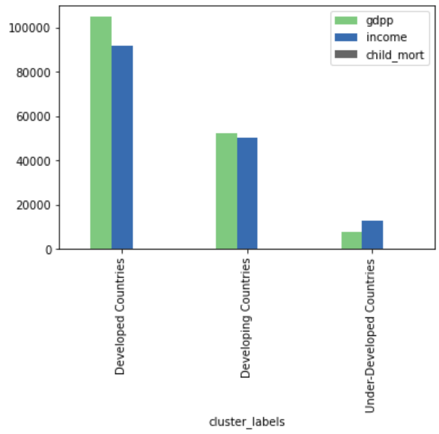


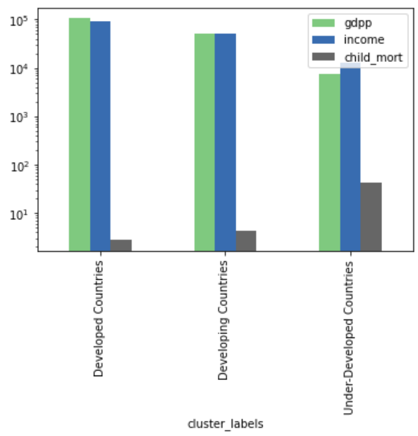
In [59]:
```
# Profiling GDP, INCOME AND CHID_MORT in sub-plots
data_grouped = data_hierarchical[['gdpp', 'income', 'child_mort','cluster_labels']].groupby('cluster_labels').mean()
data_grouped.plot(kind='bar', subplots=True)
plt.show()
```



In [60]:
```
# Profiling GDP, INCOME AND CHID_MORT together
data_grouped.plot(kind = 'bar', colormap = 'Accent')
data_grouped.plot(kind = 'bar',logy = True, colormap = 'Accent')
```

Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x2496a95d430>





Countries that require aid on priority

In [61]:
```
require_aid_countries = data_hierarchical[data_hierarchical['cluster_labels'] =='Under-Developed Countries'].sort_values(by = ['gdpp','child_mort',
'income'], ascending = [True, False, True]).head(10)
require_aid_countries
```

Out[61]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster_labels
26	Burundi	93.6	20.6052	26.7960	90.552	764	12.30	57.7	6.26	231	Under-Developed Countries
88	Liberia	89.3	62.4570	38.5860	302.802	700	5.47	60.8	5.02	327	Under-Developed Countries
37	Congo, Dem. Rep.	116.0	137.2740	26.4194	165.664	609	20.80	57.5	6.54	334	Under-Developed Countries
112	Niger	123.0	77.2560	17.9568	170.868	814	2.55	58.8	7.49	348	Under-Developed Countries
132	Sierra Leone	160.0	67.0320	52.2690	137.655	1220	17.20	55.0	5.20	399	Under-Developed Countries
93	Madagascar	62.2	103.2500	15.5701	177.590	1390	8.79	60.8	4.60	413	Under-Developed Countries
106	Mozambique	101.0	131.9850	21.8299	193.578	918	7.64	54.5	5.56	419	Under-Developed Countries
31	Central African Republic	149.0	52.6280	17.7508	118.190	888	2.01	47.5	5.21	446	Under-Developed Countries
94	Malawi	90.5	104.6520	30.2481	160.191	1030	12.10	53.1	5.31	459	Under-Developed Countries
50	Eritrea	55.2	23.0878	12.8212	112.306	1420	11.60	61.7	4.61	482	Under-Developed Countries

In [62]:
```
require_aid_countries_priority_1 = require_aid_countries.head(5)
require_aid_countries_priority_1['aid priority'] = "Aid Requirement Priority 1"
require_aid_countries_priority_1
```

Out[62]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster_labels	aid priority
26	Burundi	93.6	20.6052	26.7960	90.552	764	12.30	57.7	6.26	231	Under-Developed Countries	Aid Requirement Priority 1
88	Liberia	89.3	62.4570	38.5860	302.802	700	5.47	60.8	5.02	327	Under-Developed Countries	Aid Requirement Priority 1
37	Congo, Dem. Rep.	116.0	137.2740	26.4194	165.664	609	20.80	57.5	6.54	334	Under-Developed Countries	Aid Requirement Priority 1
112	Niger	123.0	77.2560	17.9568	170.868	814	2.55	58.8	7.49	348	Under-Developed Countries	Aid Requirement Priority 1
132	Sierra Leone	160.0	67.0320	52.2690	137.655	1220	17.20	55.0	5.20	399	Under-Developed Countries	Aid Requirement Priority 1

In [63]:
```
require_aid_countries_priority_2 = require_aid_countries.tail(5)
require_aid_countries_priority_2['aid priority'] = "Aid Requirement Priority 2"
require_aid_countries_priority_2
```

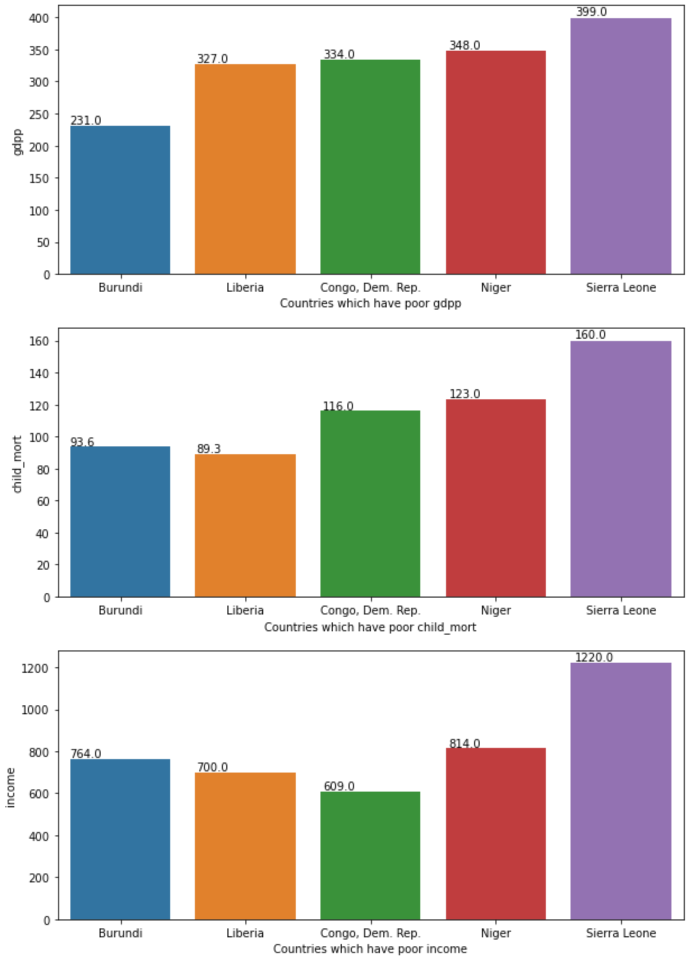
Out[63]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	cluster_labels	aid priority
93	Madagascar	62.2	103.2500	15.5701	177.590	1390	8.79	60.8	4.60	413	Under-Developed Countries	Aid Requirement Priority 2
106	Mozambique	101.0	131.9850	21.8299	193.578	918	7.64	54.5	5.56	419	Under-Developed Countries	Aid Requirement Priority 2
31	Central African Republic	149.0	52.6280	17.7508	118.190	888	2.01	47.5	5.21	446	Under-Developed Countries	Aid Requirement Priority 2
94	Malawi	90.5	104.6520	30.2481	160.191	1030	12.10	53.1	5.31	459	Under-Developed Countries	Aid Requirement Priority 2
50	Eritrea	55.2	23.0878	12.8212	112.306	1420	11.60	61.7	4.61	482	Under-Developed Countries	Aid Requirement Priority 2

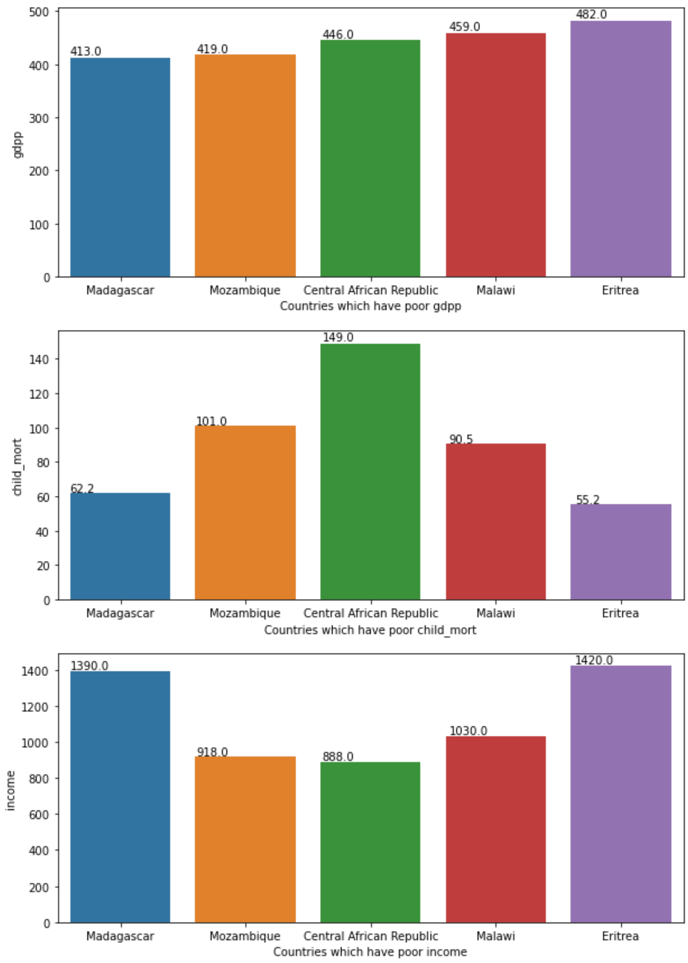
Countries that are in need of help

In [64]:
```
def results_plots(df):
    plt.figure(figsize=[10,15])
    for i,column_name in enumerate(profiling_cols):
        plt.subplot(3,1,i+1)
        ax = sns.barplot(x='country', y=column_name, data= df)
        for each_bar in ax.patches:
            ax.annotate(str(each_bar.get_height()), (each_bar.get_x() * 1.01 , each_bar.get_height() * 1.01))
        plt.ylabel(column_name)
        plt.xlabel('Countries which have poor %s' %column_name)
```

In [65]:
```
results_plots(require_aid_countries_priority_1)
```



In [66]:
```
results_plots(require_aid_countries_priority_2)
```



Countries that are in need of aid:

Priority;

- Burundi
- Liberia
- Congo, Dem. Rep.
- Niger
- Sierra Leone
- Madagascar
- Mozambique
- Central African Republic
- Malawi
- Eritrea