Hierarchical Clustering - Unsupervised Learning on Country Data dataset

Hierarchical Clustering

- Hierarchical clustering is a connectivity-based clustering model that groups the data points together that are close to each other based on the measure of similarity or distance. The assumption is that data points that are close to each other are more similar or related than data points that are farther apart.

Types of Hierarchical Clustering

- Agglomerative Clustering - Agglomerative is a bottom-up approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
- Divisive clustering - It is the reverse of the agglomerative algorithm as it is a top-down approach.

Objective

- To categorise the countries using socio-economic and health factors that determine the overall development of the country and to choose the cluster of countries that are in need of aid.

Dataset Source & Brief

- The dataset has been sourced from Kaggle and it contains 167 rows and 10 columns.
- It has child_mort column representing child mortality rate measured as the number of deaths per 1,000 live births of children under five years of age, exports represents the income generated from selling goods and services to other countries, health column represents the expenditure on healthcare, imports column measures the amount of money spent on purchasing goods and services from other countries, income column represents the per capita income or the average income per person,inflation indicates

the annual inflation rate, life_expec represents the average life expectancy and is measured in years, total_fer represents the total fertility rate and gdpp refers to the GDP per capita of the countries.

Import the Libraries

In [42]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Load and read the dataset

In [43]:

```
df=pd.read_csv(r"C:\Users\manme\Documents\Priya\Stats and ML\Dataset\Country-data.csv")
df.head()
```

Out[43]:

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

Check basic information

In [44]:

```
df.shape    #check shape
```

Out[44]:

```
(167, 10)
```

In [45]:

```
df.columns  #check column names
```

Out[45]:

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

In [46]:

```
df.duplicated().sum()  #check duplicates
```

Out[46]:

```
0
```

In [47]:

```
df.info()    #check 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
```

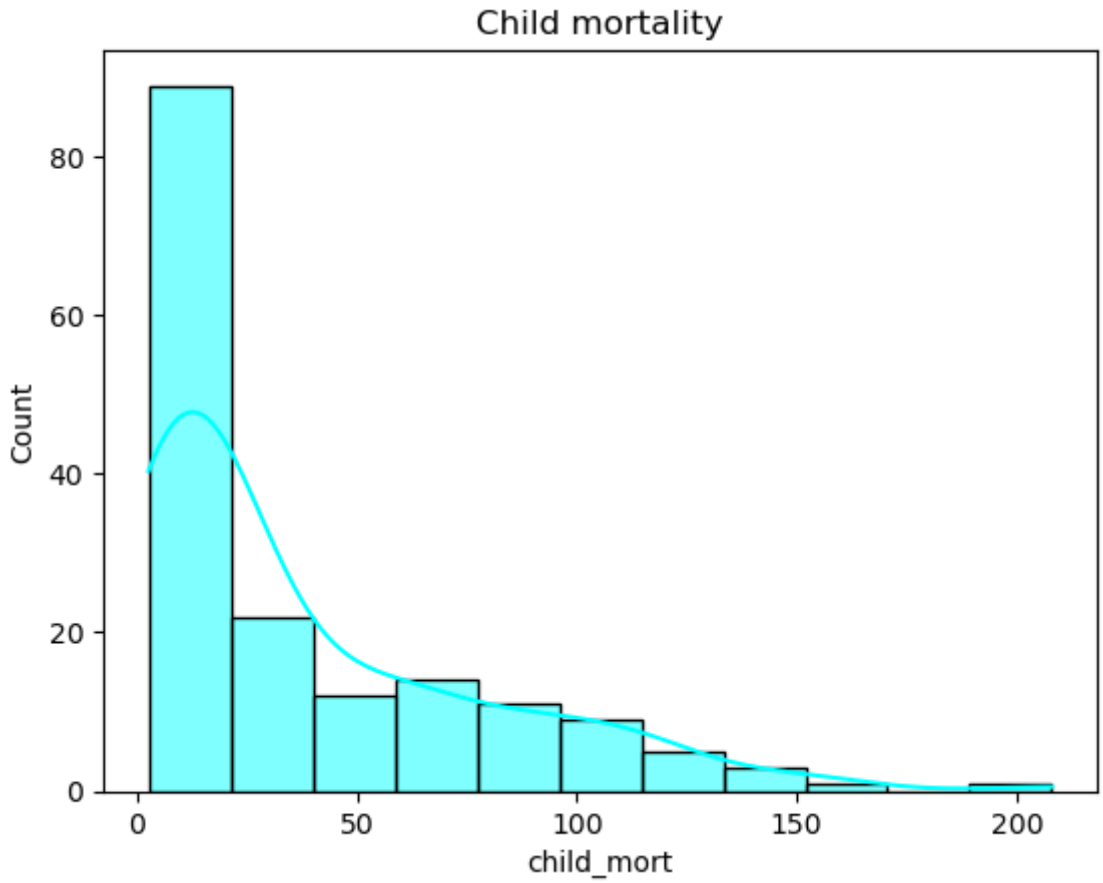
In [48]:

```
# Segregation of Numerical and Categorical Variables/Columns
categorical_col = df.select_dtypes(include = ['object']).columns
numerical_col = df.select_dtypes(exclude = ['object']).columns
```

Data Visualization

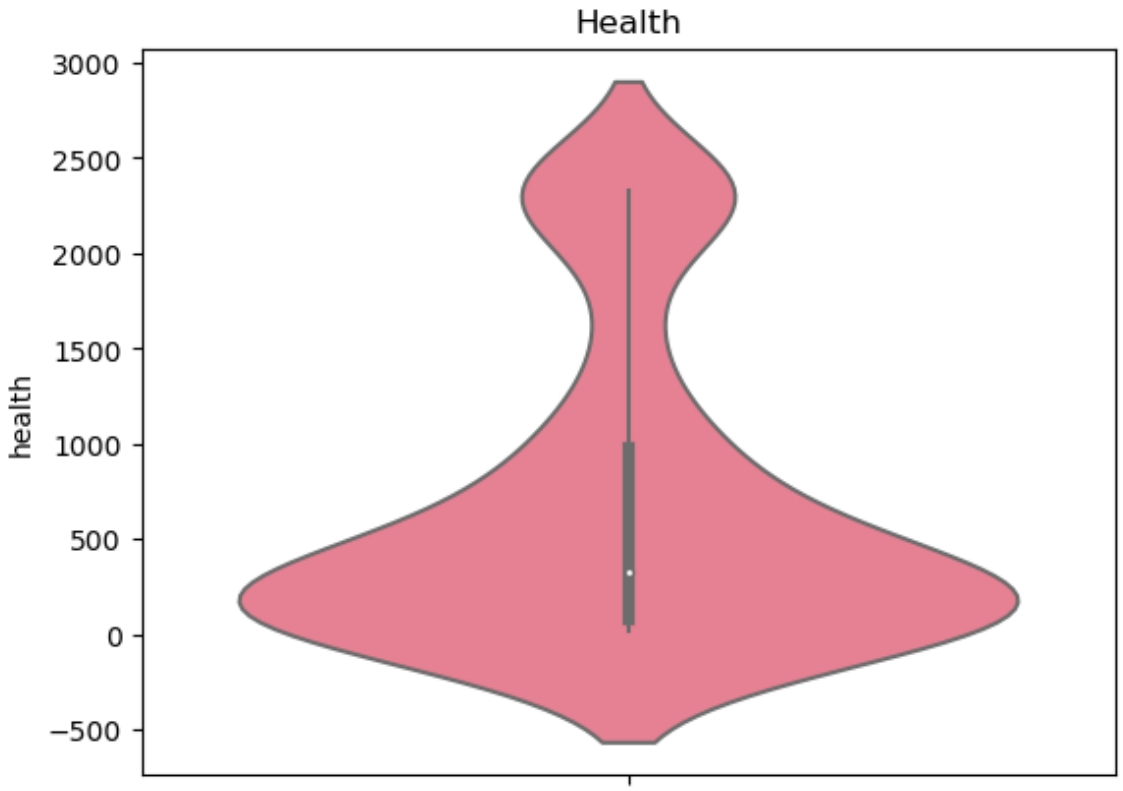
In [49]:

```
sns.histplot( x= "child_mort",data = df, kde=True, color='cyan')
plt.title('Child mortality')
plt.show()
```



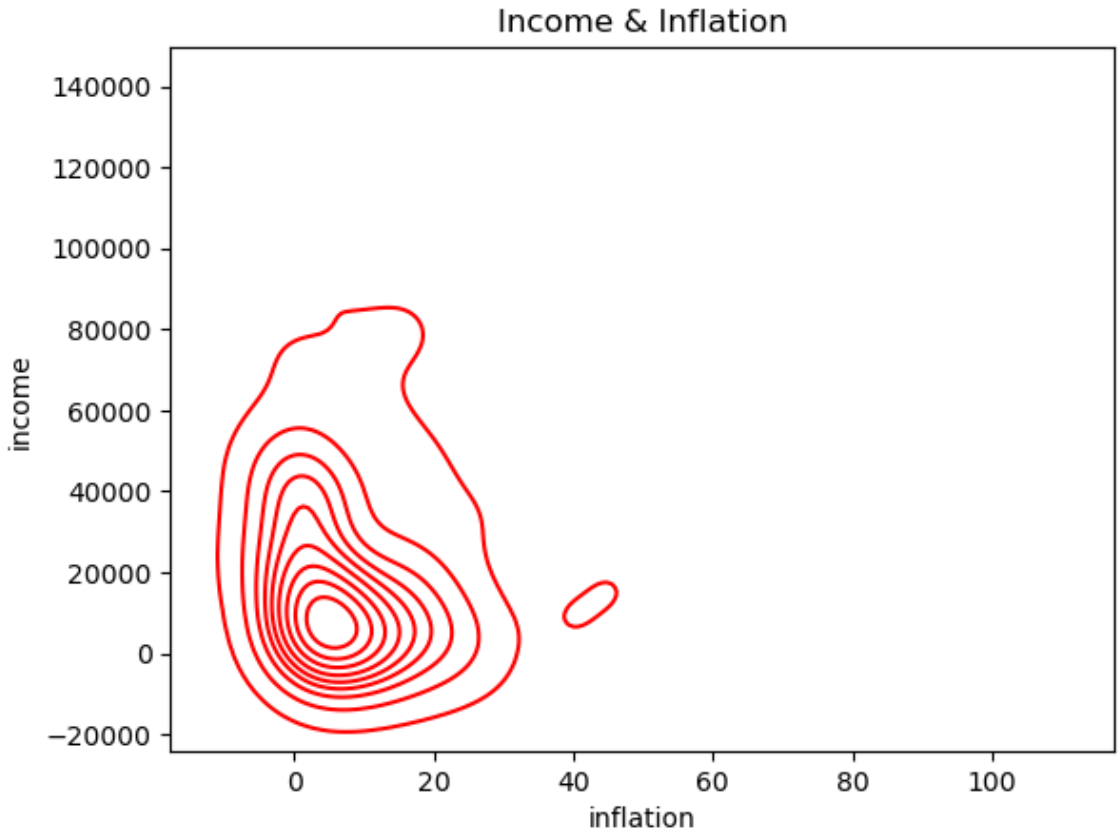
In [86]:

```
sns.violinplot(y='health', data=df,palette ="husl")
plt.title('Health')
plt.show()
```



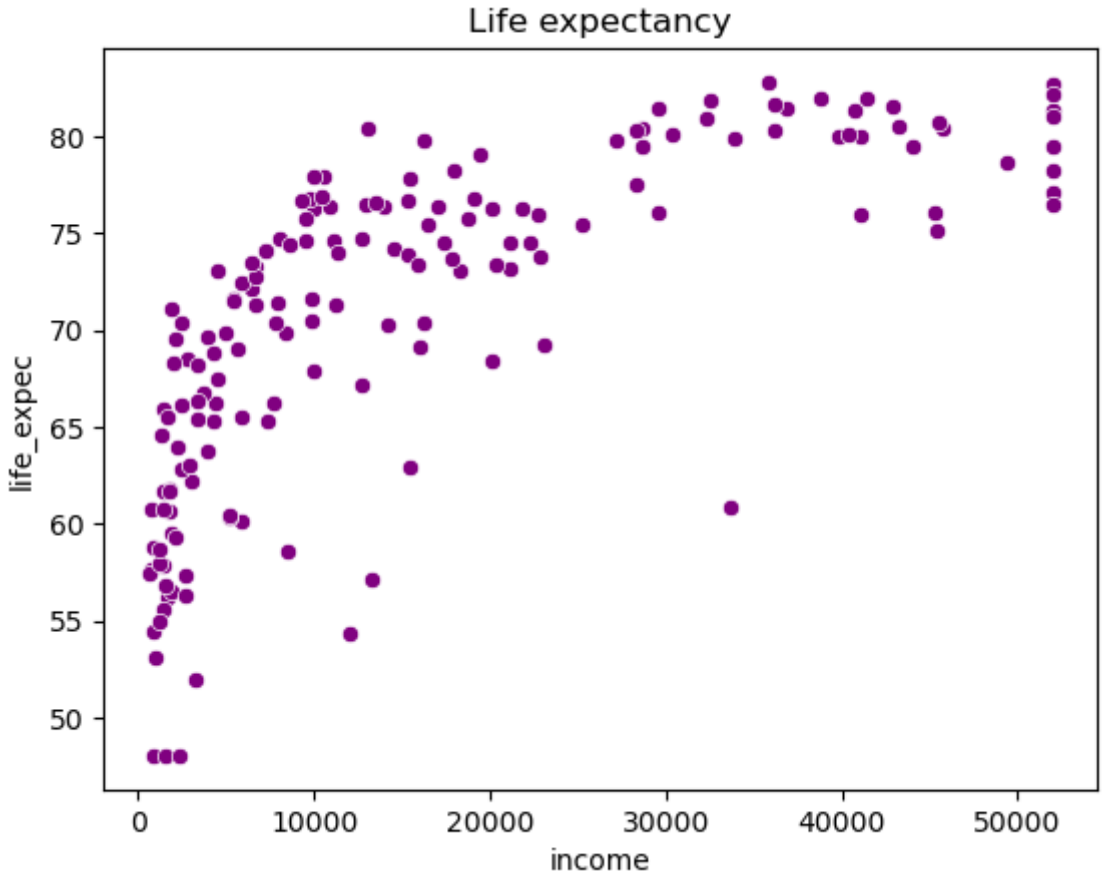
In [51]:

```
sns.kdeplot(x='inflation', y= 'income',data=df,color='red')
plt.title('Income & Inflation')
plt.show()
```



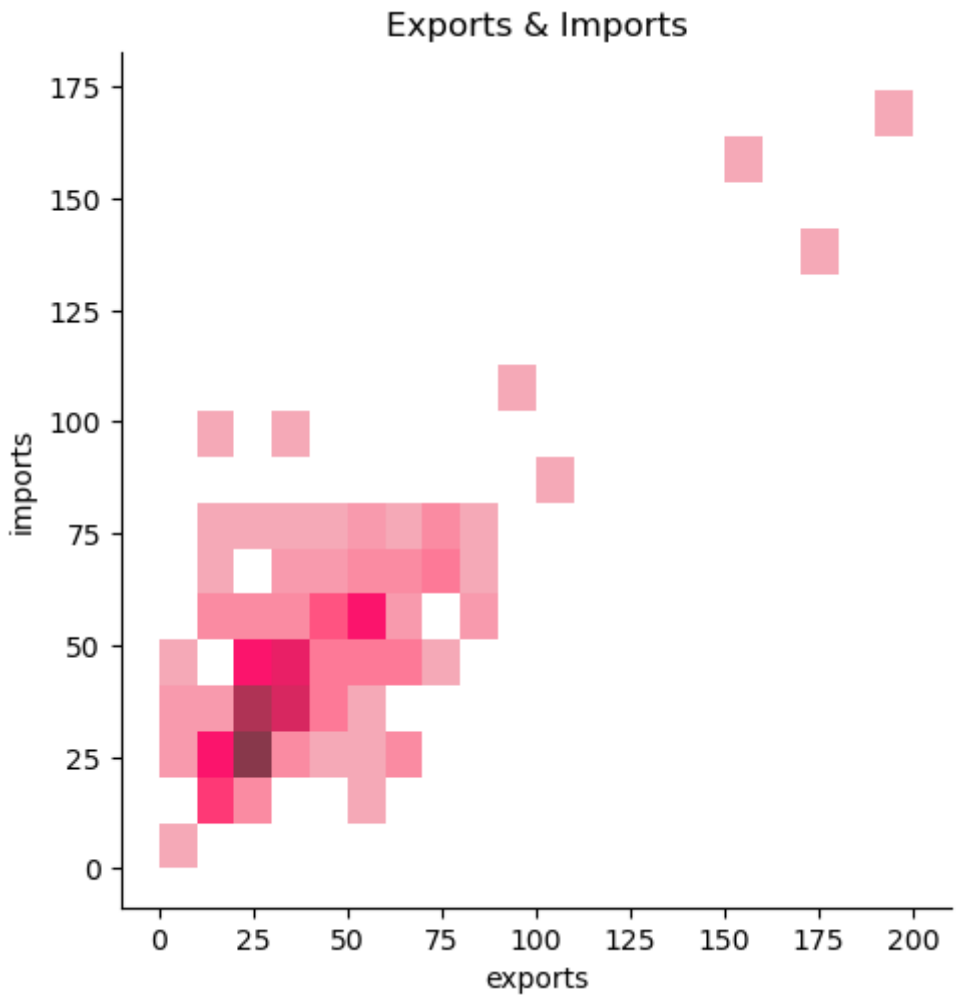
In [87]:

```
sns.scatterplot(data=df, x='income', y='life_expec', color='purple')
plt.title('Life expectancy')
plt.show()
```



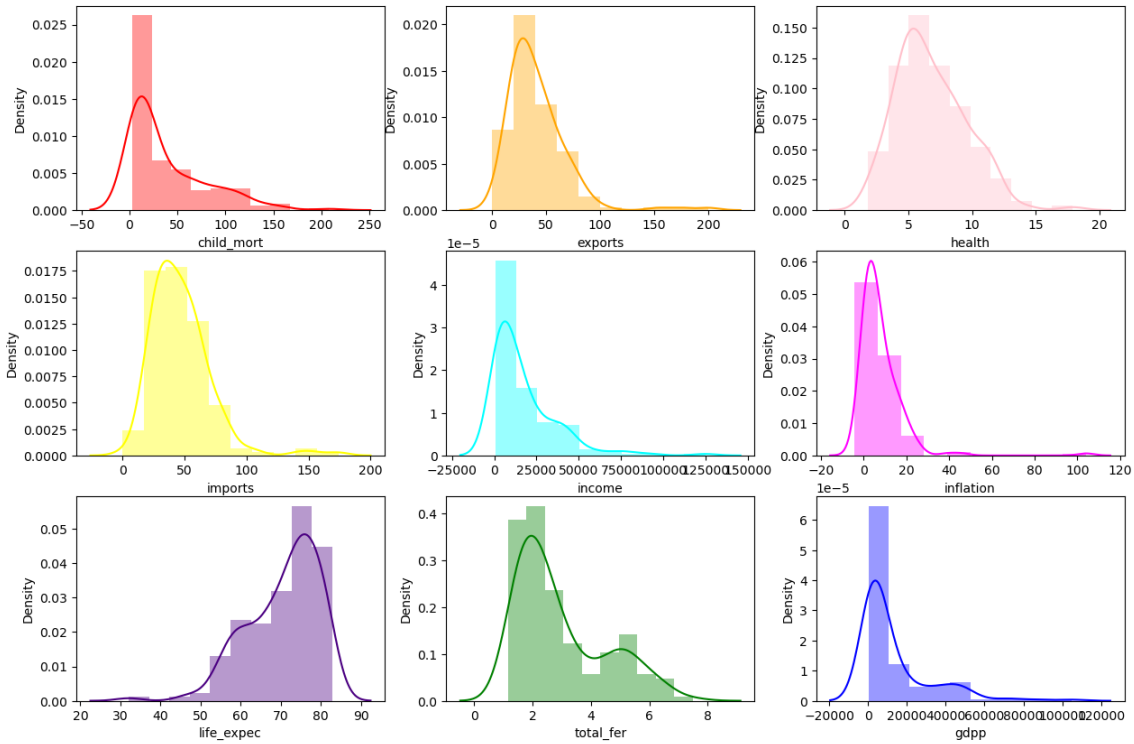
In [53]:

```
sns.displot(x='exports', y='imports', data=df,color='pink')
plt.title('Exports & Imports')
plt.show()
```



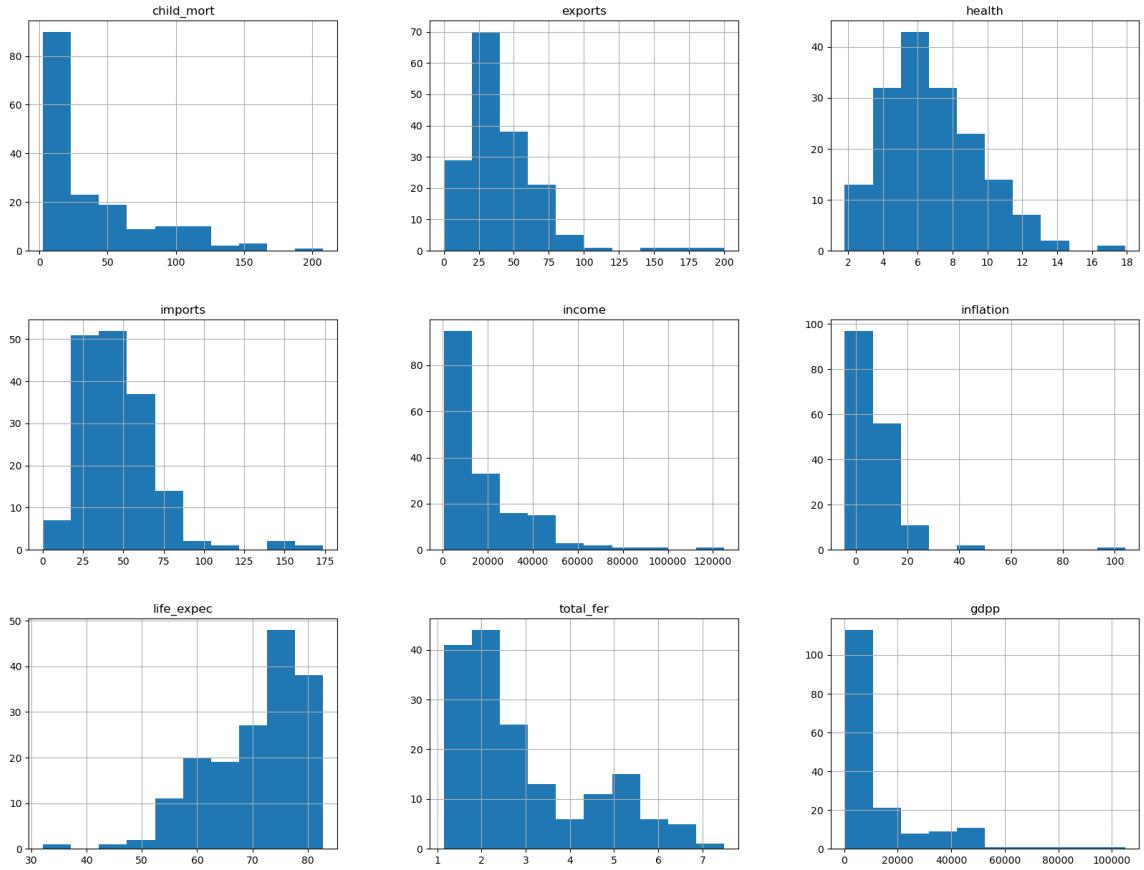
In [54]:

```
f, ax = plt.subplots(3,3,figsize=(15,10))
d1 = sns.distplot(df["child_mort"], bins=10, ax=ax[0][0],color='red')
d2 = sns.distplot(df["exports"],bins=10, ax=ax[0][1],color='orange')
d3 = sns.distplot(df["health"],bins=10, ax=ax[0][2],color="pink")
d4 = sns.distplot(df["imports"],bins=10, ax=ax[1][0],color="yellow")
d5 = sns.distplot(df["income"],bins=10, ax=ax[1][1],color="cyan")
d6 = sns.distplot(df["inflation"],bins=10, ax=ax[1][2],color="fuchsia")
d7 = sns.distplot(df["life_expec"],bins=10, ax=ax[2][0],color="indigo")
d8 = sns.distplot(df["total_fer"],bins=10, ax=ax[2][1],color="green")
d9 = sns.distplot(df["gdpp"],bins=10, ax=ax[2][2],color="blue")
```



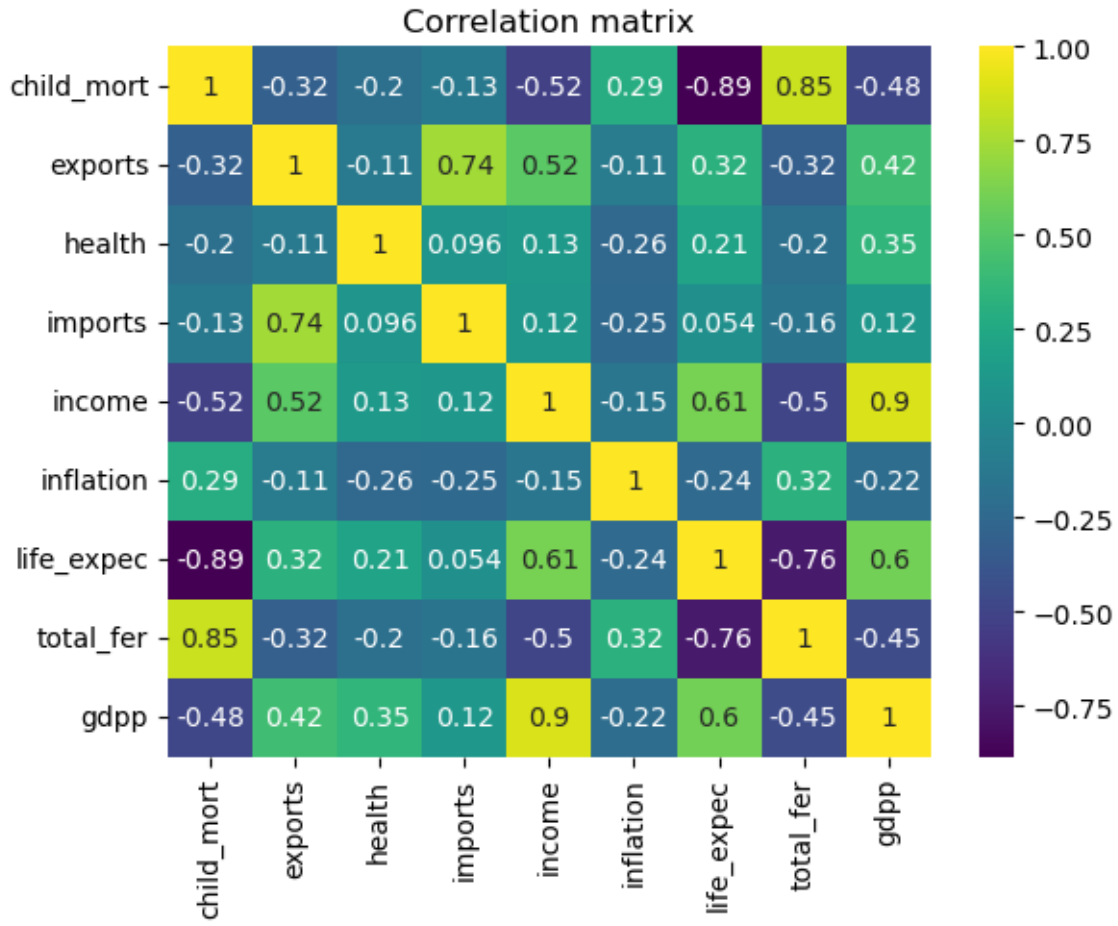
In [58]:

```
df.hist(figsize=(20,15))
plt.show()
```



In [59]:

```
sns.heatmap(df.corr(),annot=True,cmap='viridis')
plt.title('Correlation matrix')
plt.show()
```



Data Pre processing

In [60]:

```
for i in df.columns:
    print("**********************************", i ,
          "*****************************************")
    print()
    print(set(df[i].tolist()))
    print()
```

```
********************************** country ***************************
***************

{'Cameroon', 'Pakistan', 'Luxembourg', 'Rwanda', 'Portugal', 'Mauritiu
s', 'Kazakhstan', 'China', 'Netherlands', 'Serbia', 'Benin', 'Comoros',
'Panama', 'Chad', 'El Salvador', 'Argentina', 'Slovenia', 'Togo', 'Chil
e', 'Bosnia and Herzegovina', 'Uganda', 'Australia', 'Denmark', 'Eritre
a', 'Angola', 'South Africa', 'Romania', 'Bahrain', 'Croatia', 'Central
African Republic', 'Morocco', 'Brazil', 'Estonia', 'Switzerland', 'Guin
ea', 'Solomon Islands', 'Ukraine', 'Vietnam', 'Burkina Faso', 'Namibi
a', 'Zambia', 'Japan', 'Tonga', 'Montenegro', 'Turkey', 'Ecuador', 'Uni
ted Arab Emirates', 'Guyana', 'Suriname', 'Haiti', 'Lao', 'Oman', 'Arme
nia', 'Mauritania', 'Gambia', 'Liberia', 'Cyprus', 'Grenada', 'Israel',
'Lebanon', 'Samoa', 'Moldova', 'France', 'Seychelles', 'Macedonia, FY
R', 'Azerbaijan', 'Malta', 'Peru', 'Cambodia', 'Equatorial Guinea', 'Ba
rbados', 'India', 'Iceland', 'Costa Rica', 'Vanuatu', 'Belize', 'Liby
a', 'Canada', 'Egypt', 'Congo, Rep.', 'Guatemala', 'Norway', 'South Kor
ea', 'Yemen', 'Tunisia', 'Venezuela', 'Kiribati', 'United Kingdom', 'Su
dan', 'Lithuania', 'Turkmenistan', 'Paraguay', 'Maldives', 'Kuwait', 'S
```

In [61]:

```
df.isnull().sum()  #check null values
```

Out[61]:

```
country       0
child_mort    0
exports       0
health        0
imports       0
income        0
inflation     0
life_expec    0
total_fer     0
gdpp          0
dtype: int64
```

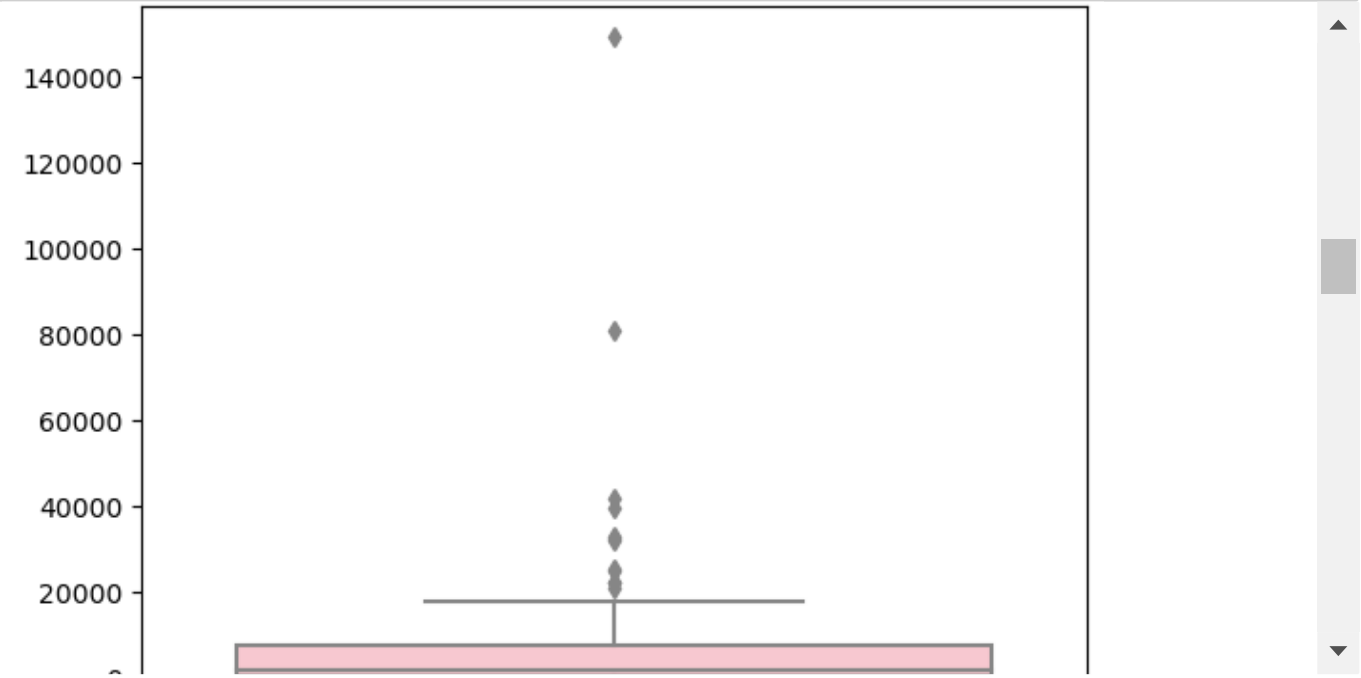
In [62]:

```
# Converting exports,imports and health spending percentages to absolute values
df['exports'] =df['exports'] * df['gdpp']/100
df['imports'] = df['imports'] * df['gdpp']/100
df['health'] = df['health'] * df['gdpp']/100
```

Check for Outliers

In [63]:

```
def boxplots(col):
    sns.boxplot(df[col],color='pink')
    plt.show()
for i in list(df.select_dtypes(exclude=['object']).columns)[0:]:
    boxplots(i)
```



Handling Outliers

In [64]:

```
def outlier(col):
    q3=df[col].quantile(0.75)
    q1=df[col].quantile(0.25)
    iqr=q3-q1
    lower=q1-1.5*iqr
    upper=q3+1.5*iqr
    df[col].clip(lower,upper,inplace=True)
```

In [65]:

```
for i in numerical_col:
    outlier(i)
```

Feature Scaling

In [66]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled = scaler.fit_transform(df.drop('country', axis = 1))
x_scaled
```

Out[66]:

```
array([[ 1.36980194, -0.80056371, -0.79556251, ..., -1.70222504,
         1.91527602, -0.84634107],
       [-0.55046422, -0.61314982, -0.51391972, ...,  0.66332125,
        -0.86277926, -0.54082746],
       [-0.27129509, -0.51557561, -0.6160104 , ...,  0.68685903,
        -0.03669088, -0.50886816],
       ...,
       [-0.37565738, -0.64788476, -0.73612924, ...,  0.28671687,
        -0.66291917, -0.78095407],
       [ 0.48533152, -0.74249447, -0.76323195, ..., -0.37234081,
         1.14914567, -0.78095407],
       [ 1.18455887, -0.71718267, -0.7406285 , ..., -2.19651829,
         1.63547189, -0.7679976 ]])
```

Building Hierarchical Cluster

Dendrogram

- A dendrogram, a tree-like figure produced by hierarchical clustering, depicts the hierarchical relationships between groups. Individual data points are located at the bottom of the dendrogram, while the largest clusters, which include all the data points, are located at the top.
- The dendrogram is created by iteratively merging or splitting clusters based on a measure of similarity or distance between data points. Clusters are divided or merged repeatedly until all data points are contained within a single cluster, or until the predetermined number of clusters is attained

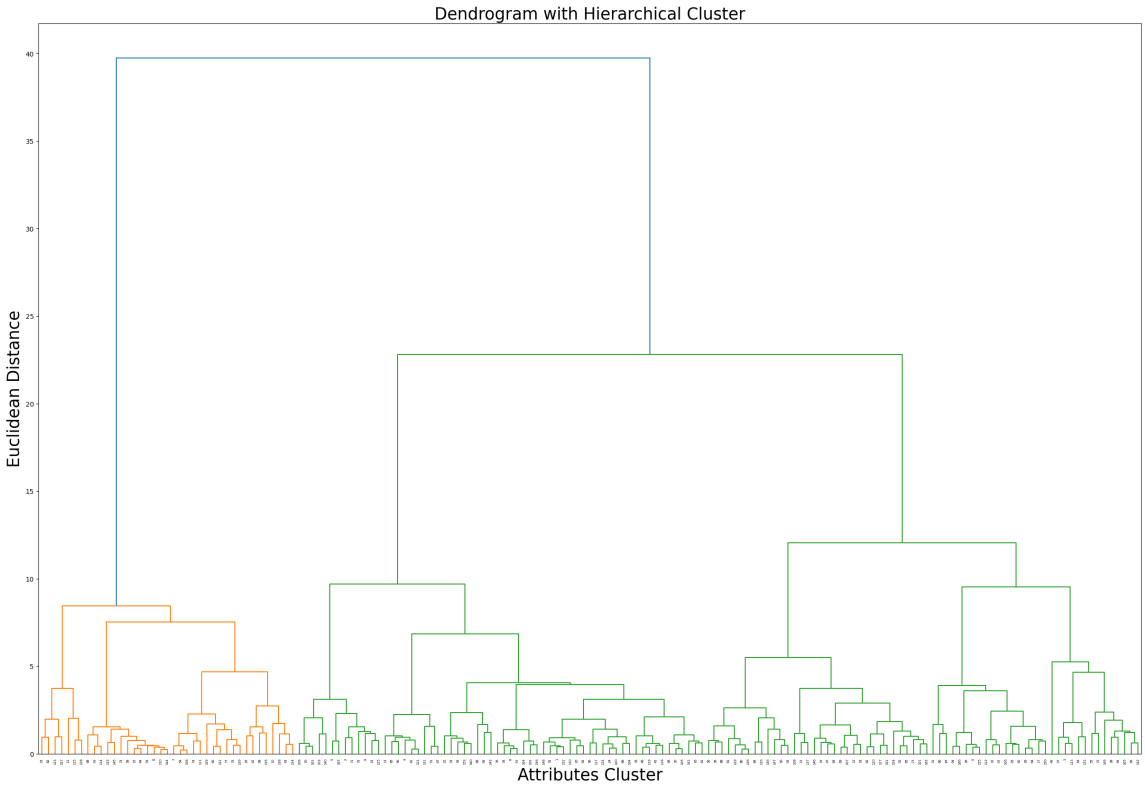
Step 1 - Finding the optimal number of clusters using the Dendrogram

In [67]:

```
import scipy.cluster.hierarchy as sch
```

In [89]:

```
plt.figure(figsize=(30,20))
dendrogram = sch.dendrogram(sch.linkage(x_scaled,method='ward'))
plt.title('Dendrogram with Hierarchical Cluster',fontsize=25)
plt.xlabel('Attributes Cluster',fontsize=25)
plt.ylabel('Euclidean Distance',fontsize=25)
plt.show()
```



Step 2- Training model by Agglomerative clustering approach

Hierarchical Agglomerative Clustering

- It is also known as the bottom-up approach or hierarchical agglomerative clustering (HAC). A structure that is more informative than the unstructured set of clusters returned by flat clustering. This clustering algorithm does not require us to prespecify the number of clusters. Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerate pairs of clusters until all clusters have been merged into a single cluster that contains all data.

In [69]:

```
from sklearn.cluster import AgglomerativeClustering
```

In [70]:

```
cluster=AgglomerativeClustering(n_clusters=3, affinity ='euclidean', linkage='ward')
cluster
```

Out[70]:

```
AgglomerativeClustering(affinity='euclidean', n_clusters=3)
```

Step 3 - Predicting the model

In [71]:

```
# Predict
y_cluster=cluster.fit_predict(x_scaled)
y_cluster
```

Out[71]:

```
array([0, 2, 2, 0, 2, 2, 2, 1, 1, 2, 1, 1, 0, 2, 2, 1, 2, 0, 0, 0, 2, 0,
       2, 1, 2, 0, 0, 0, 0, 1, 2, 0, 0, 2, 2, 2, 0, 0, 0, 2, 0, 2, 1, 1,
       1, 2, 2, 0, 2, 0, 0, 2, 0, 1, 1, 0, 0, 2, 1, 0, 1, 2, 0, 0, 0, 0,
       0, 2, 1, 0, 2, 2, 0, 1, 1, 1, 2, 1, 2, 2, 0, 0, 1, 0, 0, 2, 2, 0,
       0, 2, 2, 1, 2, 0, 0, 2, 2, 0, 1, 0, 2, 0, 2, 2, 2, 2, 0, 0, 0, 2,
       1, 1, 0, 0, 1, 1, 0, 2, 2, 2, 0, 2, 1, 1, 2, 2, 0, 0, 1, 0, 2, 2,
       0, 1, 1, 1, 0, 0, 1, 1, 2, 2, 0, 2, 1, 1, 0, 0, 2, 0, 0, 0, 2, 2,
       0, 0, 2, 1, 1, 1, 2, 2, 0, 2, 2, 0, 0], dtype=int64)
```

In [72]:

```
# Concatenation
dataset=pd.concat([df,pd.DataFrame(y_cluster)],axis=1)
dataset
```

Out[72]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total
0	Afghanistan	90.2	55.30	41.9174	248.297	1610.0	9.44	56.2	5
1	Albania	16.6	1145.20	267.8950	1987.740	9930.0	4.49	76.3	1
2	Algeria	27.3	1712.64	185.9820	1400.440	12900.0	16.10	76.5	2
3	Angola	119.0	2199.19	100.6050	1514.370	5900.0	22.40	60.1	6
4	Antigua and Barbuda	10.3	5551.00	735.6600	7185.800	19100.0	1.44	76.8	2
...	...	...	...	...	...	...	...	...	...
162	Vanuatu	29.2	1384.02	155.9250	1565.190	2950.0	2.62	63.0	3
163	Venezuela	17.1	3847.50	662.8500	2376.000	16500.0	24.16	75.4	2
164	Vietnam	23.3	943.20	89.6040	1050.620	4490.0	12.10	73.1	1
165	Yemen	56.3	393.00	67.8580	450.640	4480.0	23.60	67.5	4
166	Zambia	83.1	540.20	85.9940	451.140	3280.0	14.00	52.0	5

167 rows × 11 columns

In [73]:

```
# Renaming column name for easy understanding
renamed_col={0: 'Clusters'}
dataset.rename(columns=renamed_col,inplace=True)
```

In [74]:

```
dataset.head(2)
```

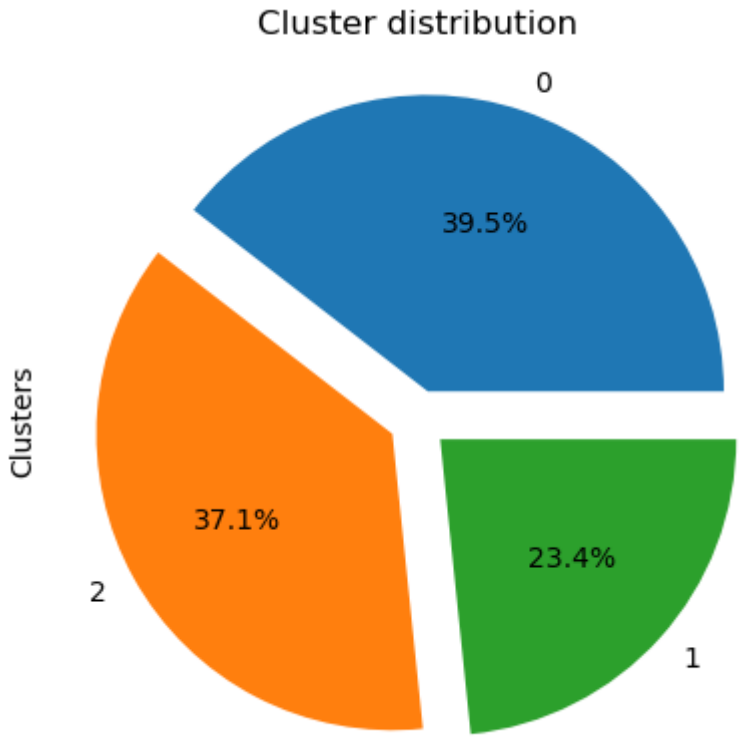
Out[74]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fe
0	Afghanistan	90.2	55.3	41.9174	248.297	1610.0	9.44	56.2	5.8
1	Albania	16.6	1145.2	267.8950	1987.740	9930.0	4.49	76.3	1.6

Step 4 - Visualizing the clusters

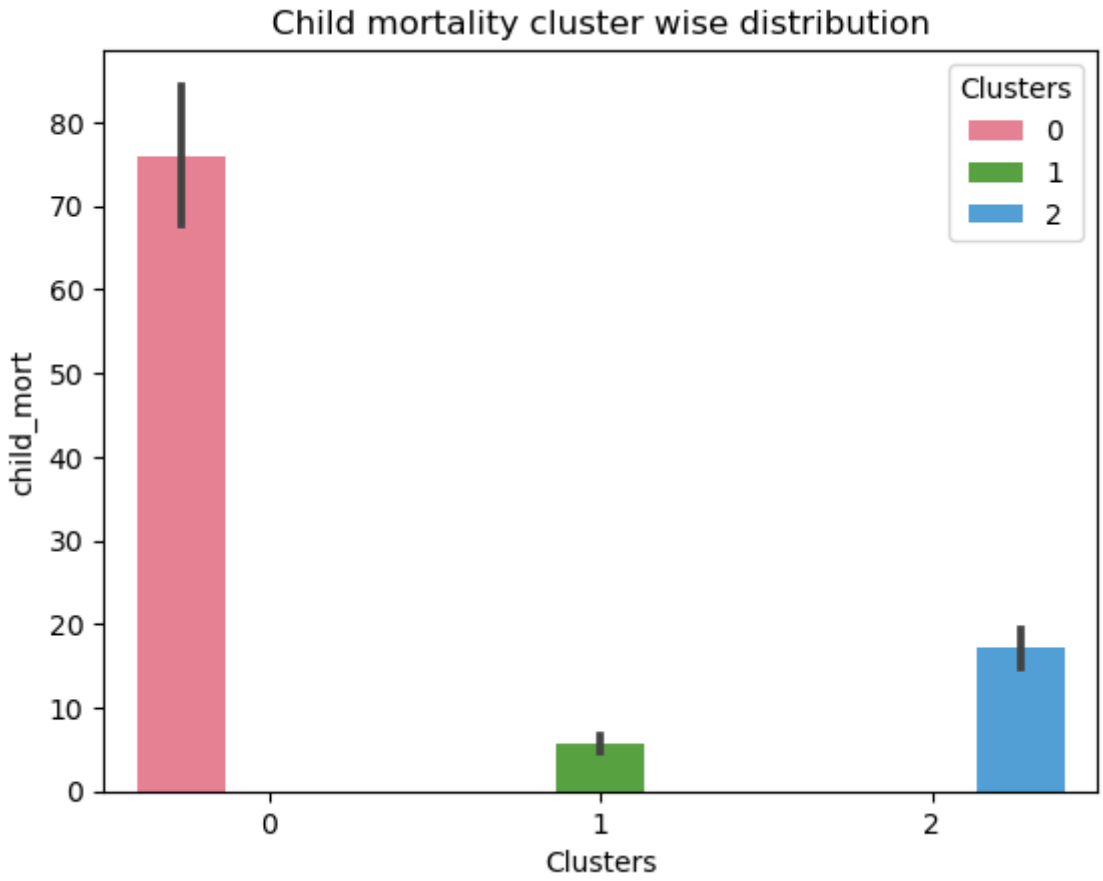
In [75]:

```
dataset['Clusters'].value_counts().plot(kind='pie',explode=
                                        [0.1,0.1,0.1],autopct='%0.1f%%')
plt.title('Cluster distribution')
plt.show()
```



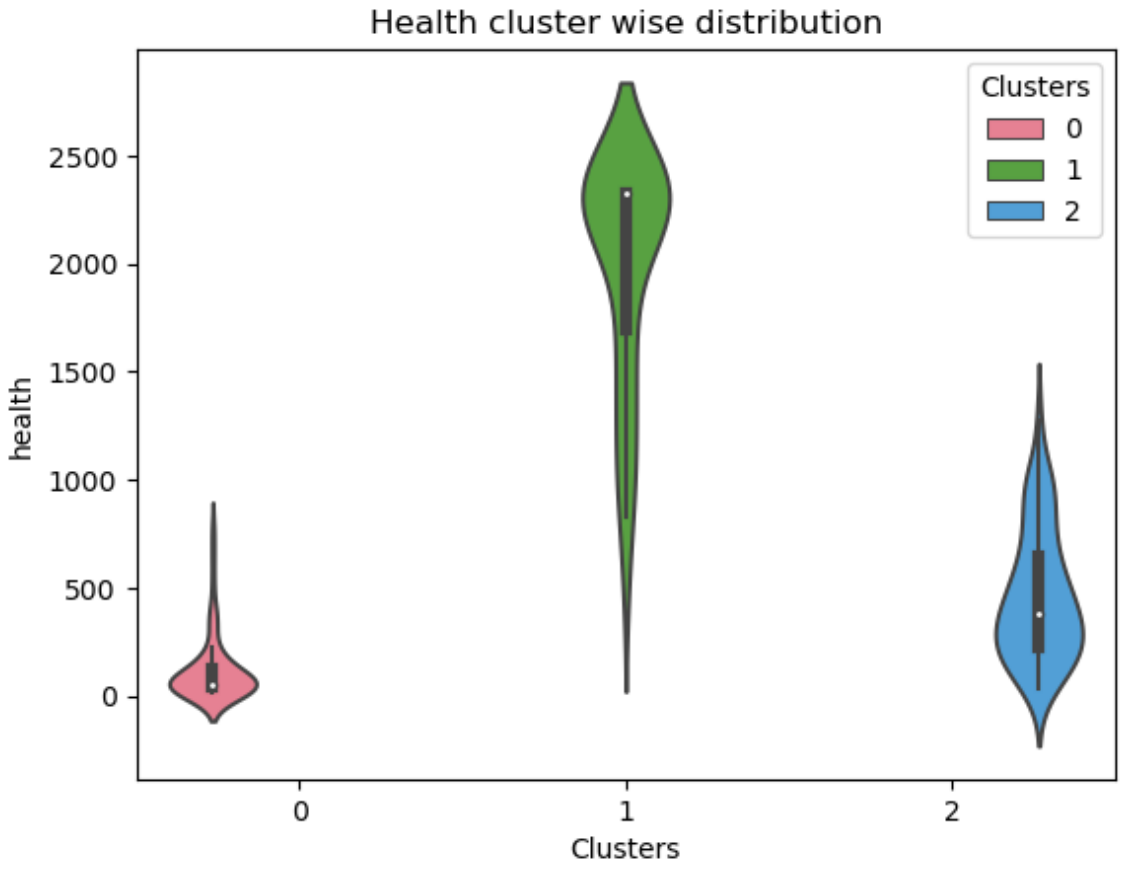
In [76]:

```
sns.barplot(x='Clusters', y='child_mort', hue='Clusters',data=dataset,palette='husl')
plt.title('Child mortality cluster wise distribution')
plt.show()
```



In [77]:

```
sns.violinplot(x='Clusters', y='health', hue='Clusters',data=dataset,palette='husl')
plt.title('Health cluster wise distribution')
plt.show()
```

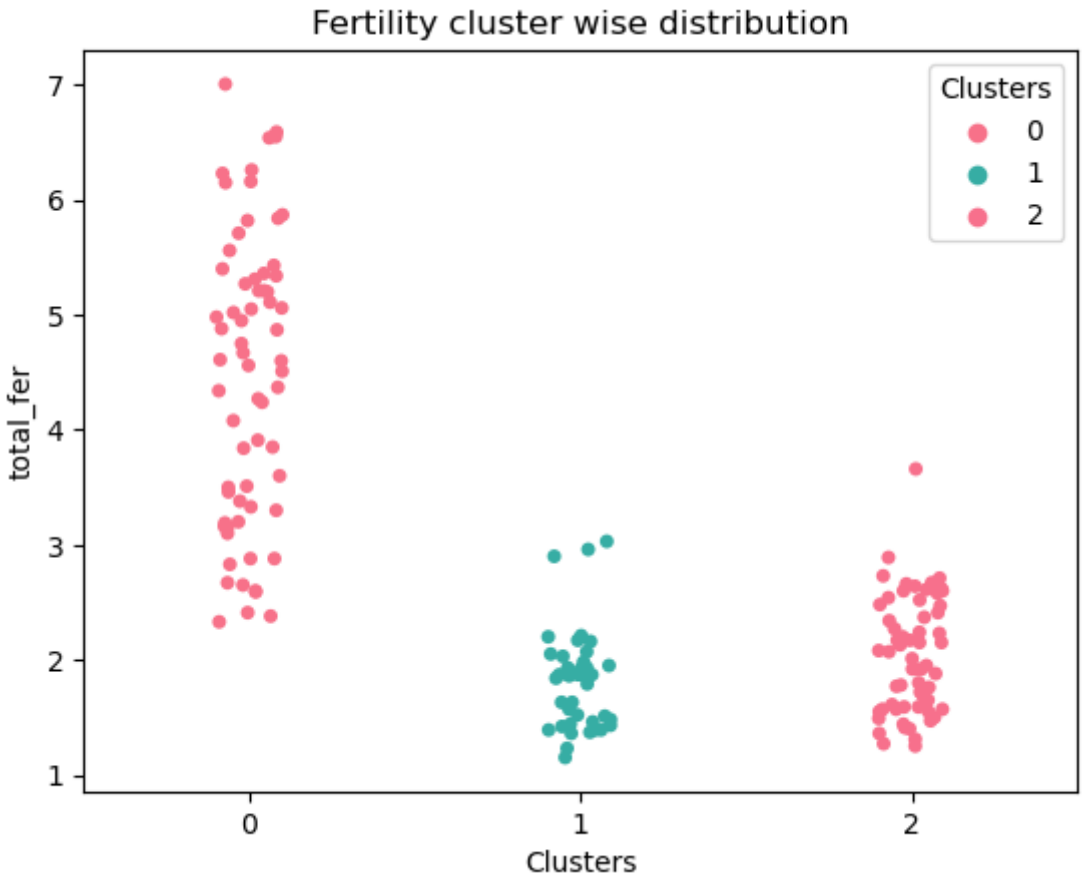


In [78]:

```
sns.swarmplot(x='Clusters', y='life_expec', hue='Clusters',data=dataset,palette='husl')
plt.title('Life expectancy cluster wise distribution')
plt.show()
```

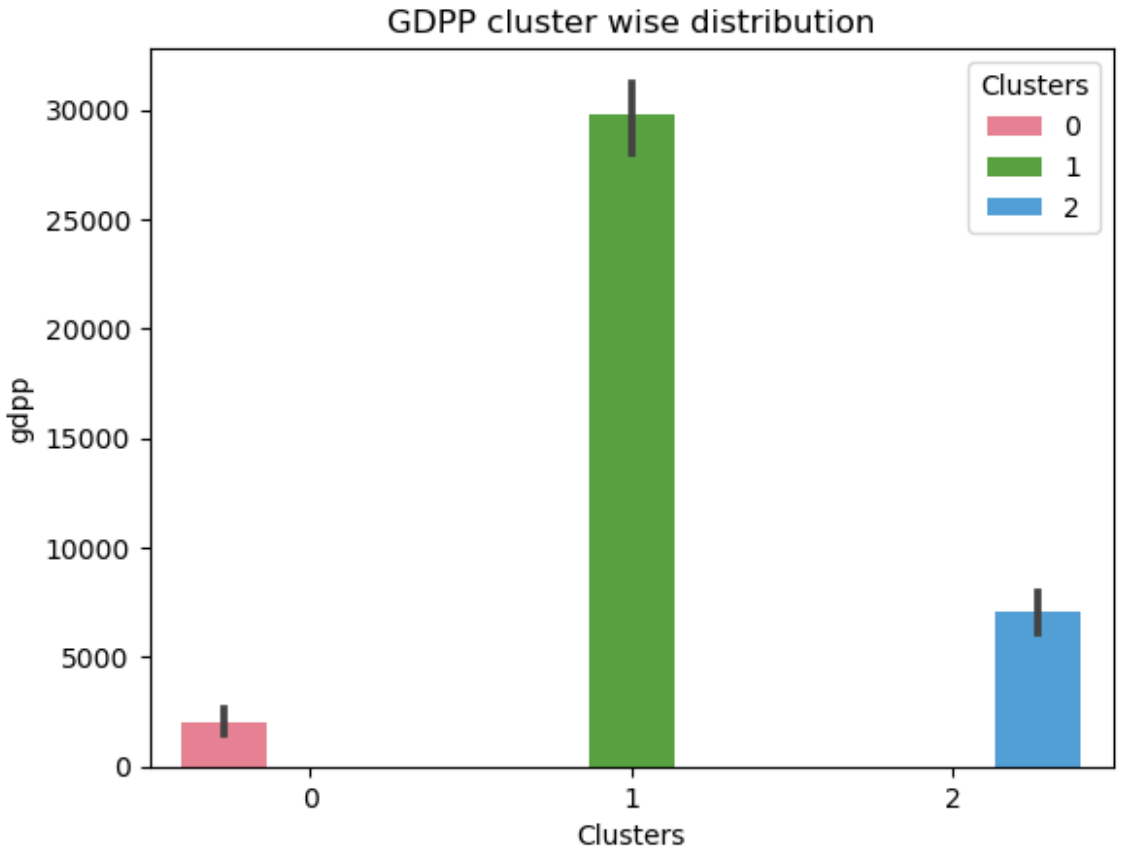
In [79]:

```
sns.stripplot(x='Clusters', y='total_fer', hue='Clusters',data=dataset,palette='husl')
plt.title('Fertility cluster wise distribution')
plt.show()
```



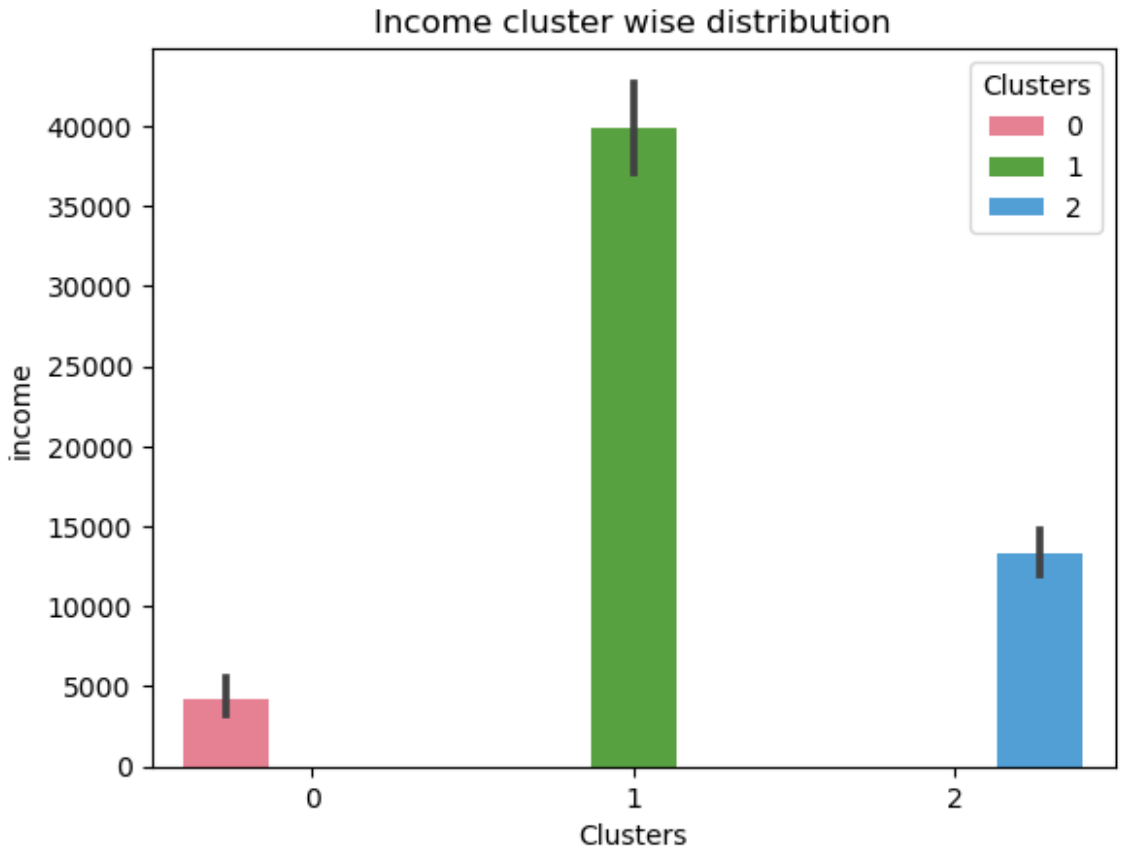
In [80]:

```
sns.barplot(x='Clusters', y='gdpp', hue='Clusters',data=dataset,
            palette='husl')
plt.title('GDPP cluster wise distribution')
plt.show()
```



In [81]:

```
sns.barplot(x='Clusters', y='income', hue='Clusters',data=dataset,
            palette='husl')
plt.title('Income cluster wise distribution')
plt.show()
```

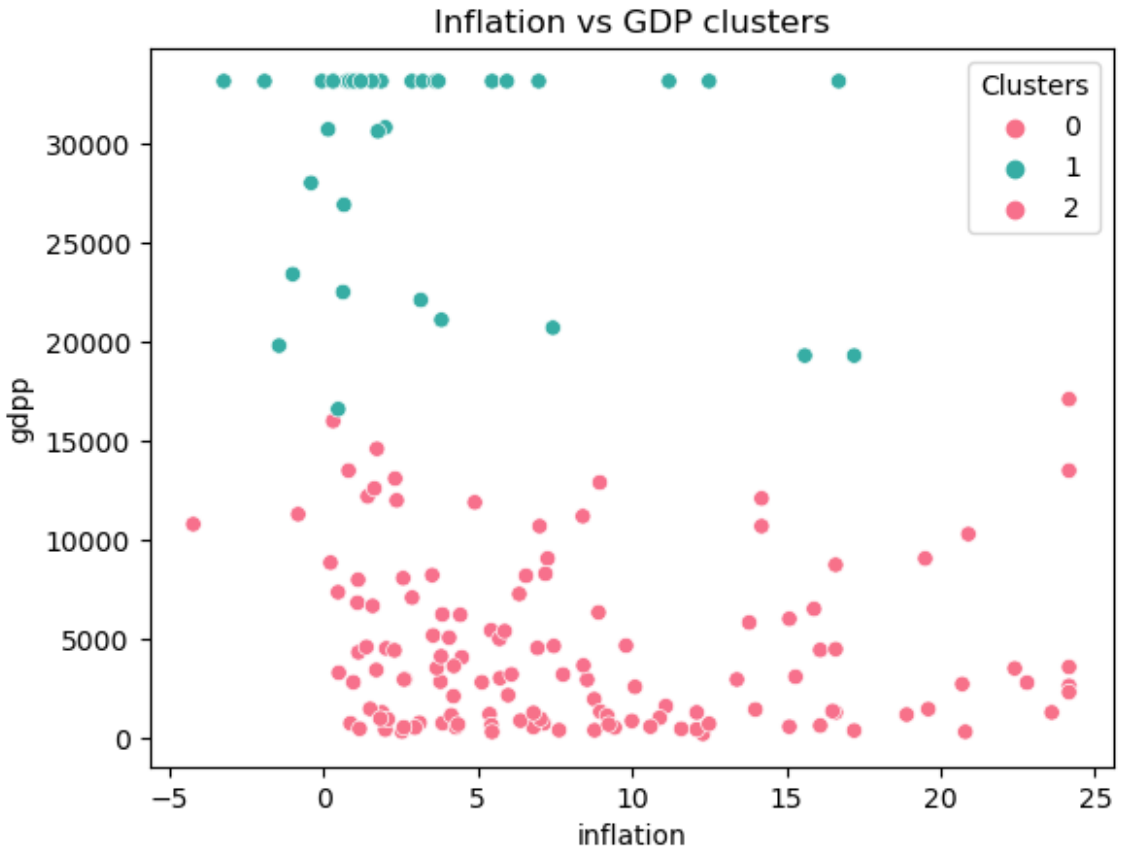


In [82]:

```
sns.scatterplot(x = 'exports', y = 'imports', hue = 'Clusters', data = dataset,
                palette = 'husl')
plt.title('Export vs Imports clusters')
plt.show()
```

In [83]:

```
sns.scatterplot(x = 'inflation',  y = 'gdpp',hue = 'Clusters', data = dataset,
                palette = 'husl')
plt.title('Inflation vs GDP clusters' )
plt.show()
```



Converting final data

In [85]:

```
dataset.to_csv('Final Hierarchical cluster report.csv')
```

Conclusion

- In this project I performed Hierarchical Agglomerative Clustering.
- From the analysis, it can be seen that Cluster 0 has the lowest income & high child mortality, so we must focus more on low income countries.
- After cluster 0 focus should be on Cluster 2.
- Cluster 1 is self sufficient, they dont need any aid.