In [1]:

```
#importing the libraries
import numpy as np
import scipy as sp
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import statsmodels
import sklearn
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

import plotly.express as px
import plotly.graph_objects as go
from chart_studio.plotly import plot, iplot
from plotly.offline import iplot
import plotly.graph_objects as go
from fancyimpute import KNN, IterativeImputer
```

In [2]:

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

In [3]:

```
 sns.set_style('whitegrid')
```

In [4]:

```
# Load the dataset
df=pd.read_csv('/Users/pamel/Downloads/Happiness.csv')
df.head()
```

Out[4]:

	Country name	Regional indicator	Ladder score	Standard error of ladder score	upperwhisker	lowerwhisker	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
0	Finland	Western Europe	7.842	0.032	7.904	7.780	10.775	0.954	72.0	0.949
1	Denmark	Western Europe	7.620	0.035	7.687	7.552	10.933	0.954	72.7	0.946
2	Switzerland	Western Europe	7.571	0.036	7.643	7.500	11.117	0.942	74.4	0.919
3	Iceland	Western Europe	7.554	0.059	7.670	7.438	10.878	0.983	73.0	0.955
4	Netherlands	Western Europe	7.464	0.027	7.518	7.410	10.932	0.942	72.4	0.913

In [5]:

```
#Checking columns

df.columns
```

Out[5]:

```
Index(['Country name', 'Regional indicator', 'Ladder score',
       'Standard error of ladder score', 'upperwhisker', 'lowerwhisker',
```

```
       'Logged GDP per capita', 'Social support', 'Healthy life expectancy',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption', 'Ladder score in Dystopia',
       'Explained by: Log GDP per capita', 'Explained by: Social support',
       'Explained by: Healthy life expectancy',
       'Explained by: Freedom to make life choices',
       'Explained by: Generosity', 'Explained by: Perceptions of corruption',
       'Dystopia + residual'],
      dtype='object')
```

In [6]:

```
#Dropping unnecessary columns

df.drop(['Standard error of ladder score', 'upperwhisker', 'lowerwhisker',
        'Explained by: Log GDP per capita', 'Ladder score in Dystopia','Explained by: Socia
        'Explained by: Healthy life expectancy',
        'Explained by: Freedom to make life choices',
        'Explained by: Generosity', 'Explained by: Perceptions of corruption', 'Dystopia +
```

In [7]:

```
#Checking data types and null values

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Country name                  149 non-null    object
 1   Regional indicator            149 non-null    object
 2   Ladder score                  149 non-null    float64
 3   Logged GDP per capita         149 non-null    float64
 4   Social support                149 non-null    float64
 5   Healthy life expectancy       149 non-null    float64
 6   Freedom to make life choices  149 non-null    float64
 7   Generosity                    149 non-null    float64
 8   Perceptions of corruption     149 non-null    float64
dtypes: float64(7), object(2)
memory usage: 10.6+ KB
```

In [8]:

```
df.head()
```

Out[8]:

	Country name	Regional indicator	Ladder score	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	Finland	Western Europe	7.842	10.775	0.954	72.0	0.949	-0.098	0.186
1	Denmark	Western Europe	7.620	10.933	0.954	72.7	0.946	0.030	0.179
2	Switzerland	Western Europe	7.571	11.117	0.942	74.4	0.919	0.025	0.292
3	Iceland	Western Europe	7.554	10.878	0.983	73.0	0.955	0.160	0.673
4	Netherlands	Western Europe	7.464	10.932	0.942	72.4	0.913	0.175	0.338

In [9]:

```
df.shape
```

Out[9]: (149, 9)

In [10]:

```
#Statistics summary

df.describe()
```

Out[10]:

	Ladder score	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
count	149.000000	149.000000	149.000000	149.000000	149.000000	149.000000	149.000000
mean	5.532839	9.432208	0.814745	64.992799	0.791597	-0.015134	0.727450
std	1.073924	1.158601	0.114889	6.762043	0.113332	0.150657	0.179226
min	2.523000	6.635000	0.463000	48.478000	0.382000	-0.288000	0.082000
25%	4.852000	8.541000	0.750000	59.802000	0.718000	-0.126000	0.667000
50%	5.534000	9.569000	0.832000	66.603000	0.804000	-0.036000	0.781000
75%	6.255000	10.421000	0.905000	69.600000	0.877000	0.079000	0.845000
max	7.842000	11.647000	0.983000	76.953000	0.970000	0.542000	0.939000

In [11]:

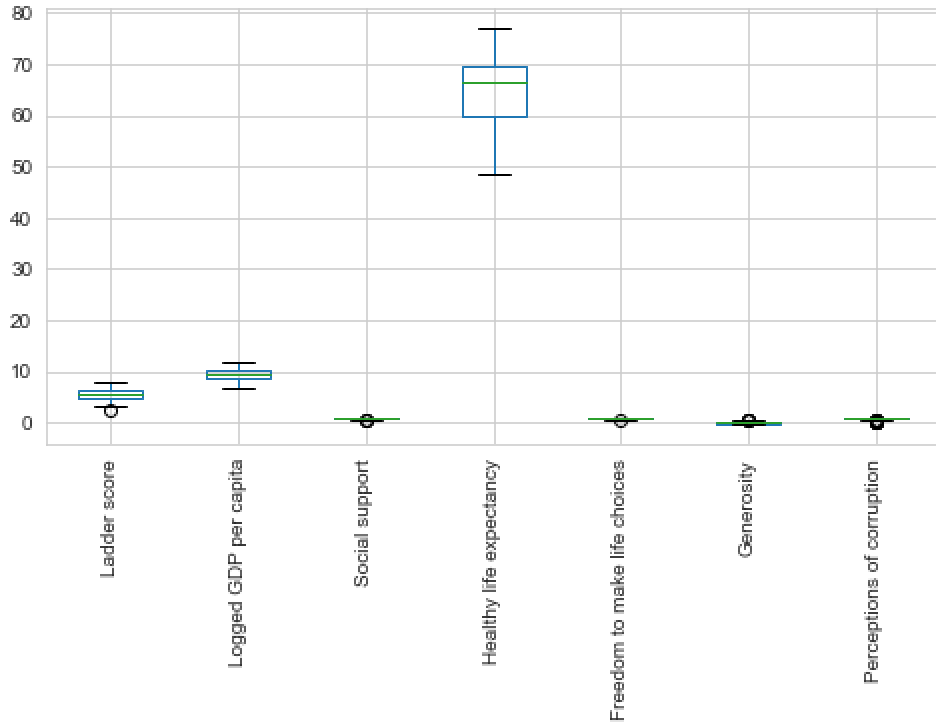
```
df.isnull().sum()
```

Out[11]:

```
Country name                    0
Regional indicator              0
Ladder score                    0
Logged GDP per capita           0
Social support                  0
Healthy life expectancy         0
Freedom to make life choices    0
Generosity                      0
Perceptions of corruption       0
dtype: int64
```

In [12]:

```
#Checking for outliers/data distribution

plt.figure(figsize=(8,4))
df.boxplot()
plt.xticks(rotation=90)

plt.show()
```

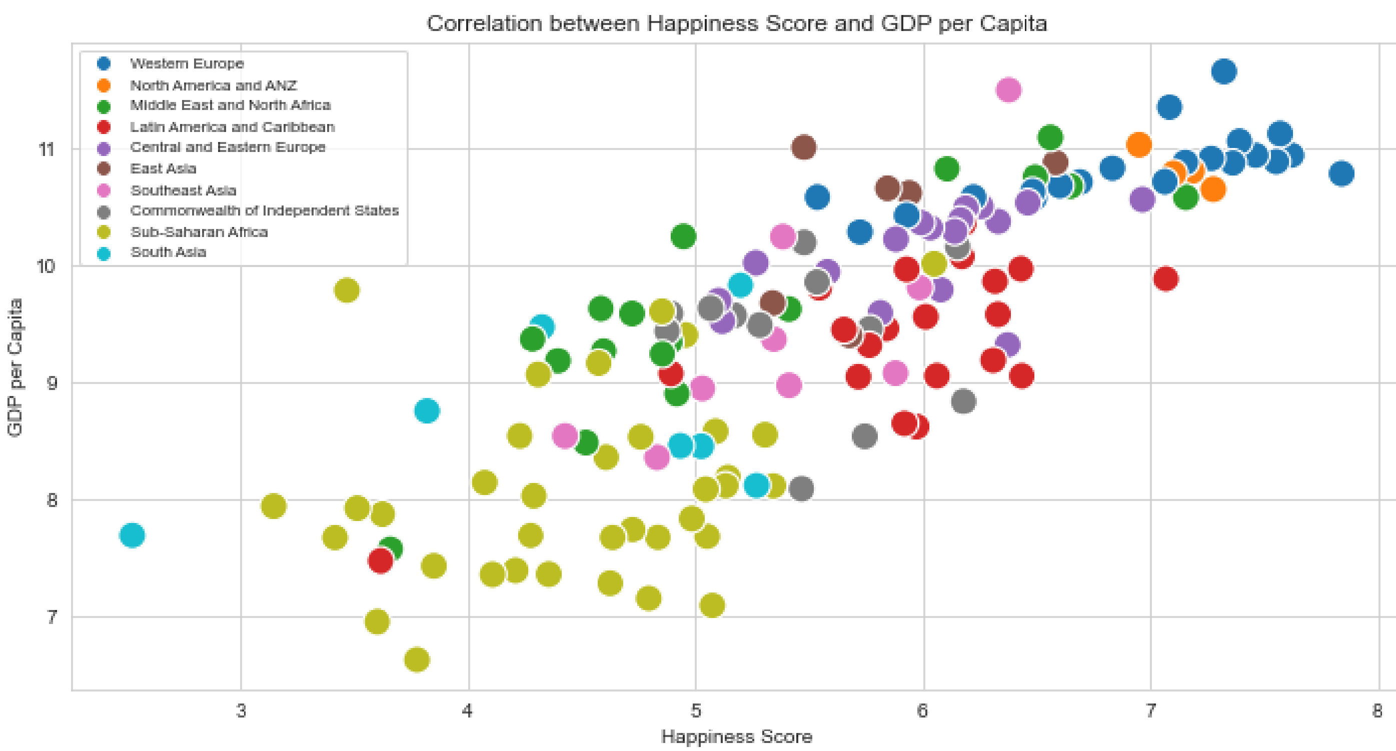


In [13]:

```
#EDA - Happiness Score vs GDP per Capita

plt.rcParams['figure.figsize'] = (12, 6)
plt.title('Correlation between Happiness Score and GDP per Capita')
sns.scatterplot(x = df['Ladder score'], y = df['Logged GDP per capita'], hue = df['Regiona

plt.legend(loc = 'upper left', fontsize = '8')
plt.xlabel('Happiness Score')
plt.ylabel('GDP per Capita')
```

Out[13]:

```
Text(0, 0.5, 'GDP per Capita')
```



In [14]:

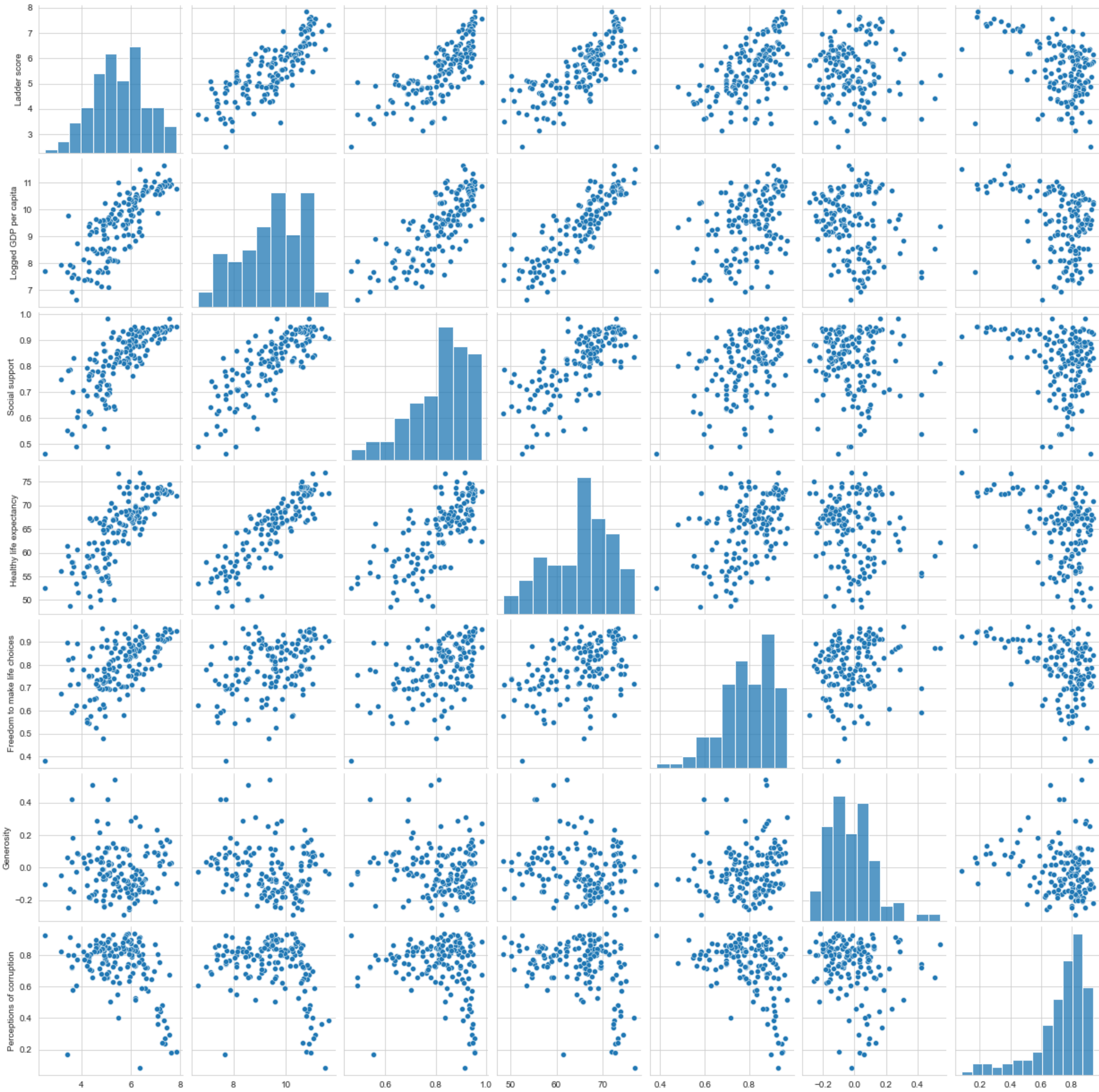
```
df.columns
```

```
Index(['Country name', 'Regional indicator', 'Ladder score',
```

Out[14]:
```
       'Logged GDP per capita', 'Social support', 'Healthy life expectancy',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption'],
      dtype='object')
```

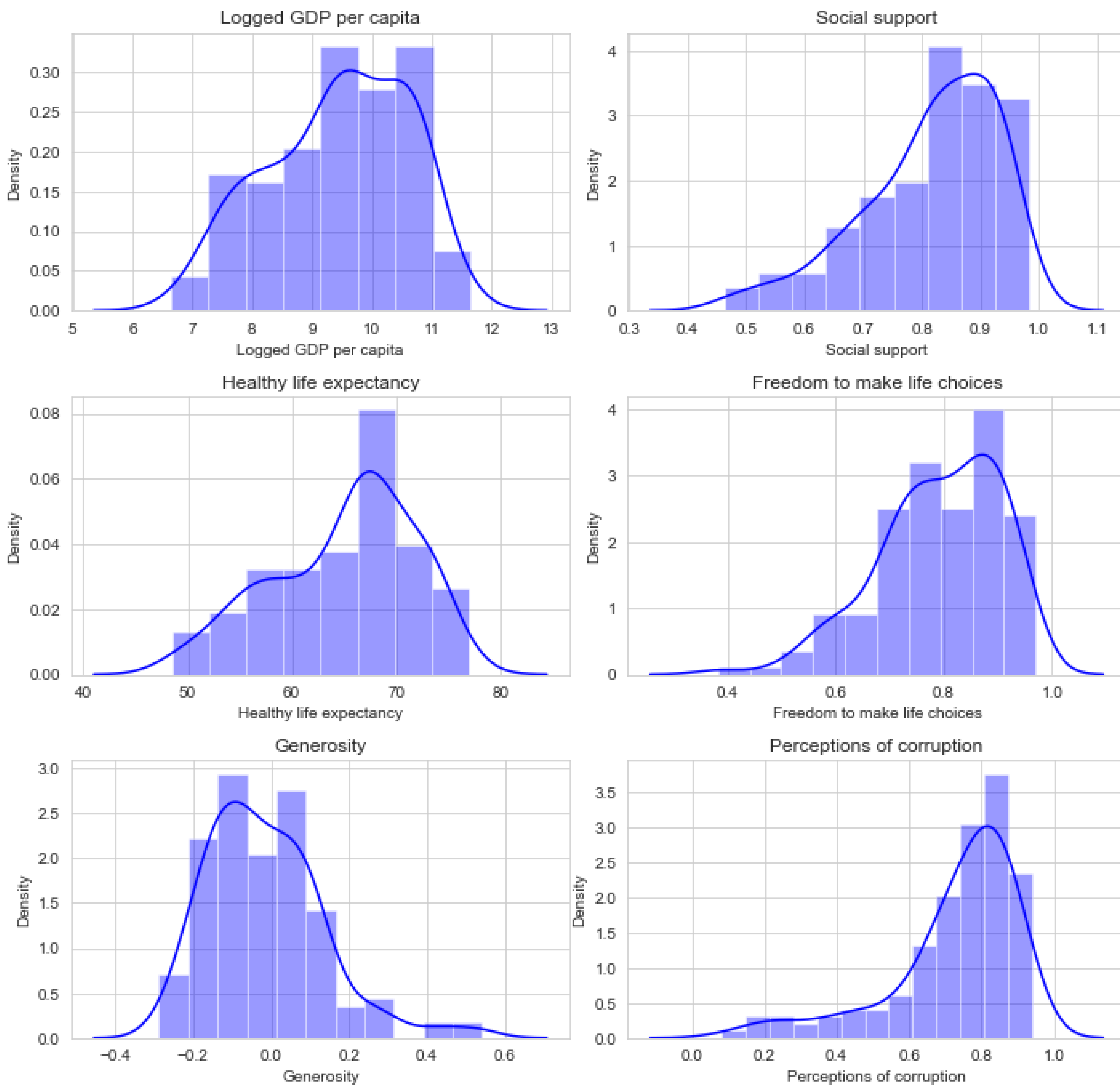
In [15]:
```
#Checking the distribution/correlation

fig = plt.figure(figsize=(10, 8))
sns.pairplot(df[['Ladder score','Logged GDP per capita', 'Social support',
       'Healthy life expectancy', 'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption']])
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x20332121610>

<Figure size 720x576 with 0 Axes>
```



In [16]:
```
columns = ['Logged GDP per capita', 'Social support',
       'Healthy life expectancy', 'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption']
plt.figure(figsize = (10, 25))
```

```python
for i in range(len(columns)):
  plt.subplot(8, 2, i+1)
  sns.distplot(df[columns[i]], color = 'b');
  plt.title(columns[i])

plt.tight_layout()
```



In [17]:

```python
df.columns
```

Out[17]:

```
Index(['Country name', 'Regional indicator', 'Ladder score',
       'Logged GDP per capita', 'Social support', 'Healthy life expectancy',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption'],
      dtype='object')
```

In [18]:

```python
# Separating out the features
x_features = df.loc[:,['Logged GDP per capita', 'Social support',
       'Healthy life expectancy', 'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption']]
```

```
Country = df.loc[:,['Country name']]
```

In [19]:

```
#outliers treatment

def outlier_limits(col):
    Q3, Q1 = np.nanpercentile(col, [75,25])
    IQR = Q3 - Q1
    UL = Q3 + 1.5*IQR
    LL = Q1 - 1.5*IQR
    return UL, LL
```

In [20]:

```
for column in x_features:
    if x_features[column].dtype != 'object':
        UL, LL = outlier_limits(x_features[column])
        x_features[column] = np.where((x_features[column] > UL) | (x_features[column] < LL
```

In [21]:

```
x_features.isnull().sum()
```

Out[21]:

```
Logged GDP per capita            0
Social support                   3
Healthy life expectancy          0
Freedom to make life choices     1
Generosity                       4
Perceptions of corruption       11
dtype: int64
```

In [22]:

```
df_mean = x_features.copy()
```

In [23]:

```
df_int = x_features.copy()
```

In [24]:

```
df_mean.isnull().sum()
```

Out[24]:

```
Logged GDP per capita            0
Social support                   3
Healthy life expectancy          0
Freedom to make life choices     1
Generosity                       4
Perceptions of corruption       11
dtype: int64
```

In [25]:

```
#Replacing outliers w/ mean

df_mean[['Social support','Freedom to make life choices','Generosity', 'Perceptions of cor
```

In [26]:

```
df_mean.isnull().sum()
```

Out[26]:

```
Logged GDP per capita           0
Social support                  0
Healthy life expectancy         0
Freedom to make life choices    0
Generosity                      0
Perceptions of corruption       0
dtype: int64
```

In [27]:

```
df_int.isnull().sum()
```

Out[27]:

```
Logged GDP per capita            0
Social support                   3
Healthy life expectancy          0
Freedom to make life choices     1
Generosity                       4
Perceptions of corruption       11
dtype: int64
```

In [28]:

```
#Replacing outliers w/ interpolation

df_int[['Social support','Freedom to make life choices','Generosity', 'Perceptions of corr
```

In [29]:

```
df_int.isnull().sum()
```

Out[29]:

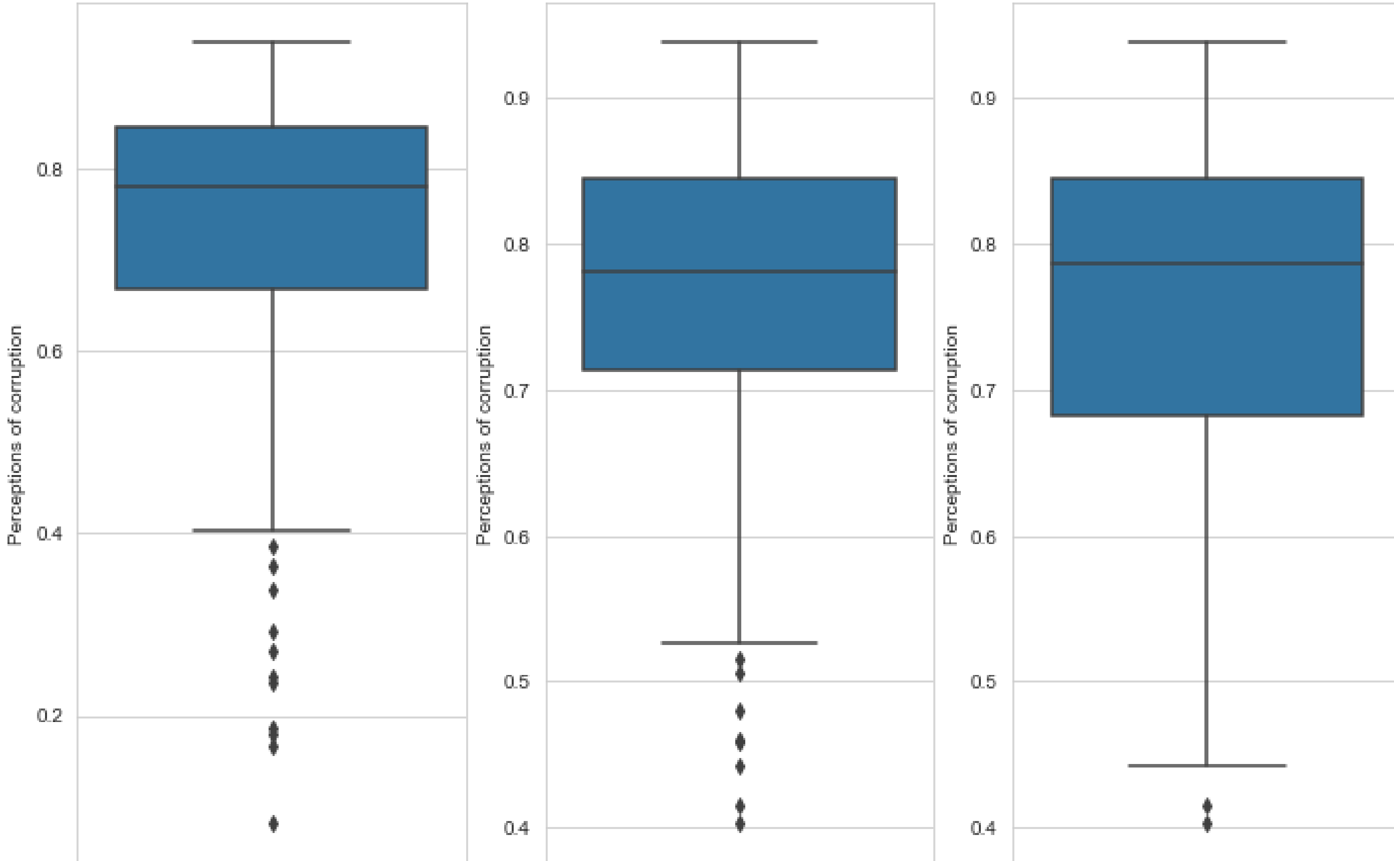
```
Logged GDP per capita           0
Social support                  0
Healthy life expectancy         0
Freedom to make life choices    0
Generosity                      0
Perceptions of corruption       0
dtype: int64
```

In [30]:

```
data =pd.read_csv('/Users/pamel/Downloads/Happiness.csv')
```

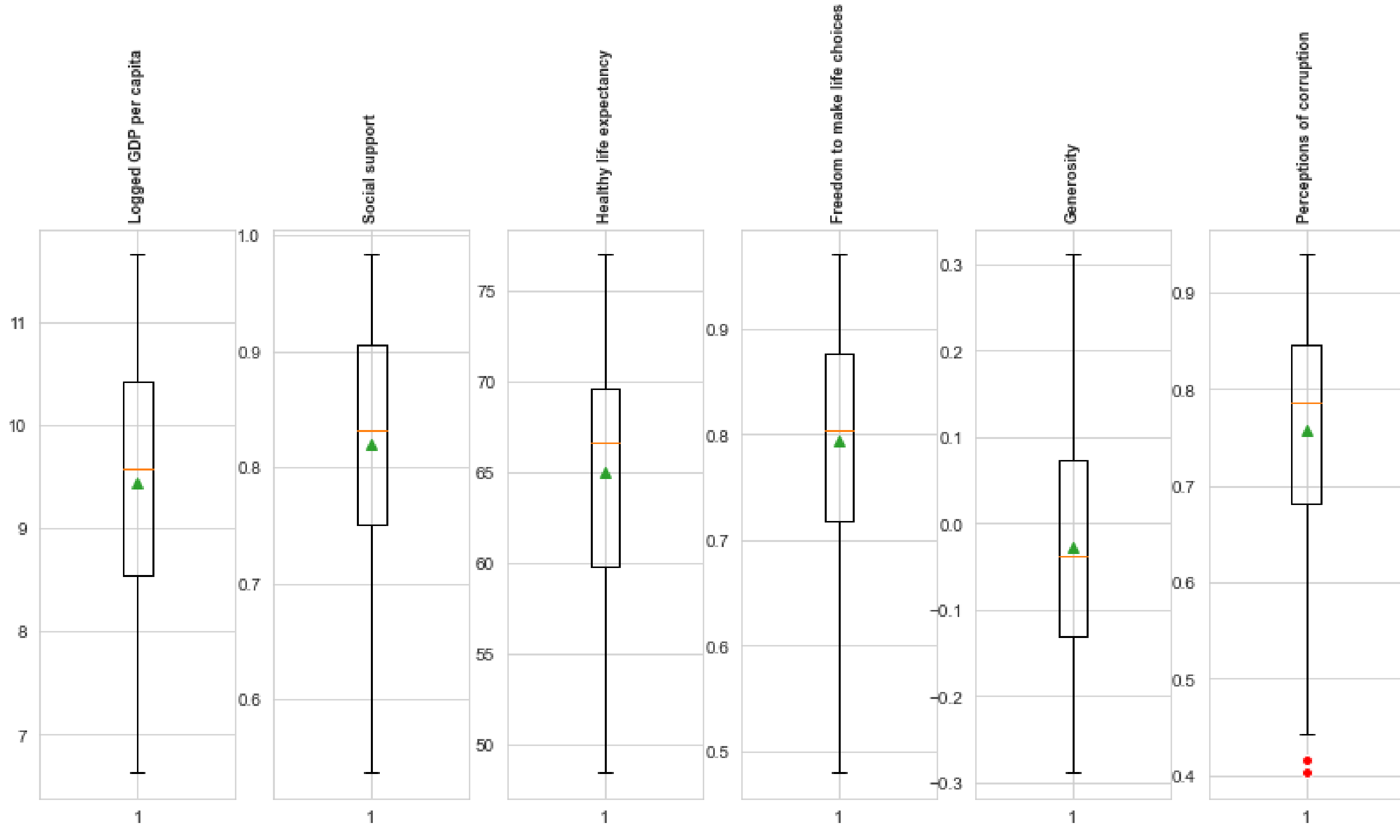
In [31]:

```
# Comparing best outlier treatment (Mean or interpolation method)

plt.figure(figsize = (12, 8))
plt.subplot(1,3,1)
sns.boxplot(y = data.iloc[:,11])

plt.subplot(1,3,2)
sns.boxplot(y = df_mean.iloc[:,5])

plt.subplot(1,3,3)
sns.boxplot(y = df_int.iloc[:,5])
plt.show()
```



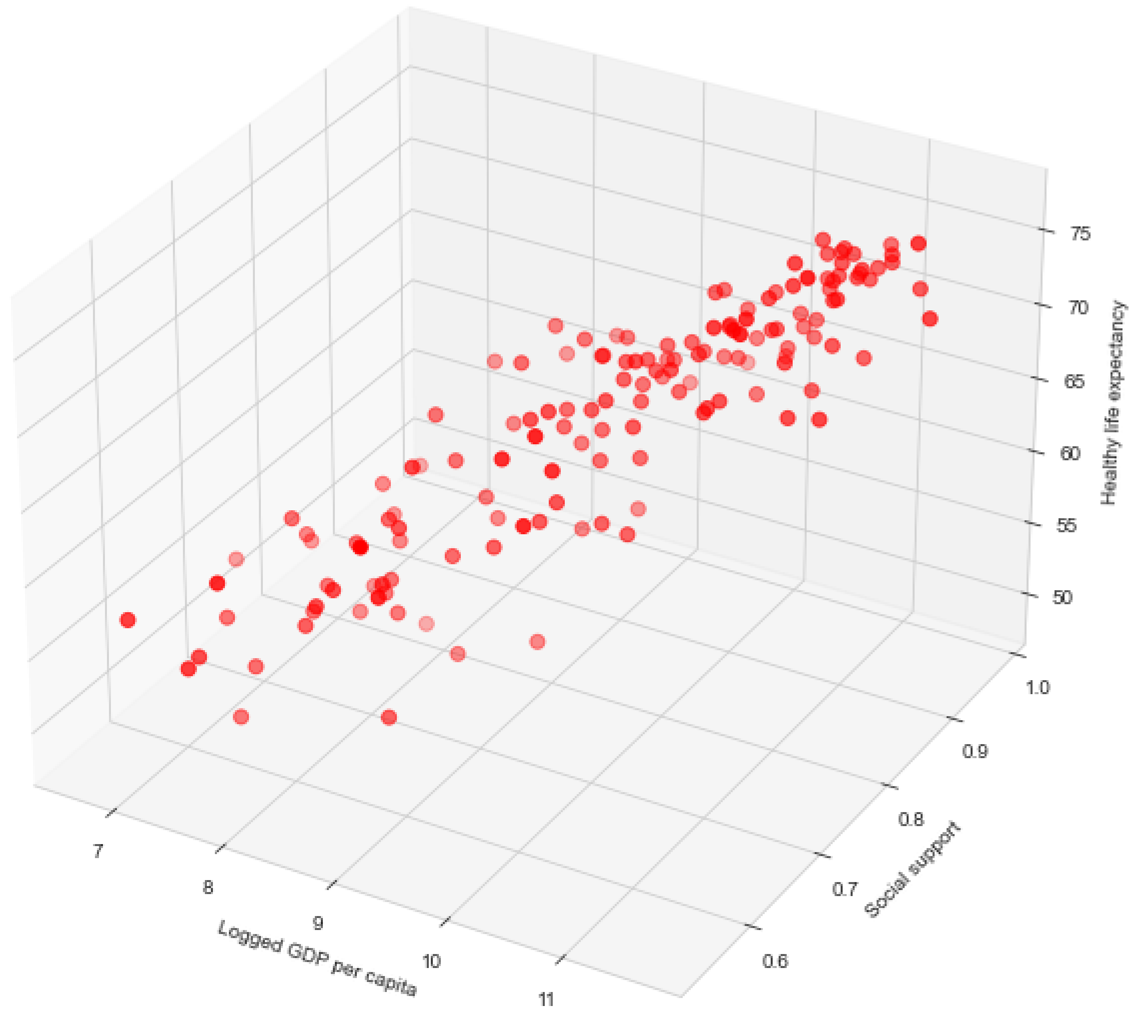
In [32]:

```
#Data after outliers treatment (Interpolation method)

red_circle = dict(markerfacecolor='red', marker='o', markeredgecolor='white')

fig, axs = plt.subplots(1, len(df_int.iloc[:6:]), figsize=(14,6))


for i, ax in enumerate(axs.flat):
    ax.boxplot(df_int.iloc[:,i], flierprops=red_circle, showmeans=True)
    ax.set_title(df_int.columns[i], fontsize=9, fontweight='bold', rotation=90)
    ax.tick_params(axis='y', labelsize=10)
```



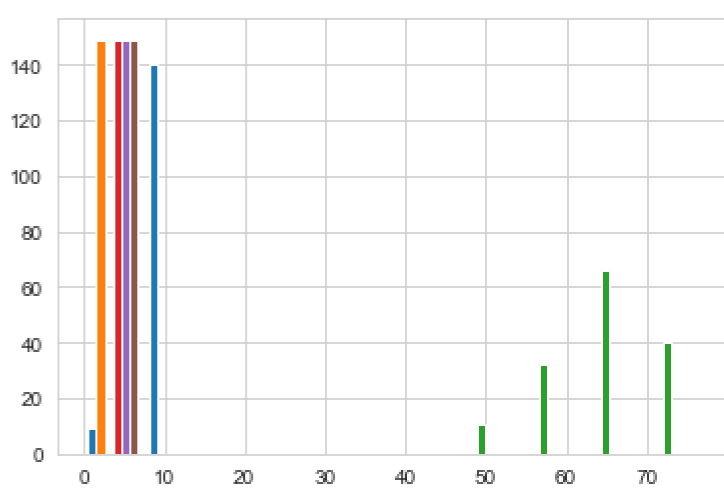
In [33]:

```
#3D visualisation

%matplotlib inline
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# defining a figure and a 3D axis
fig = plt.figure(figsize=(8,10))
ax = Axes3D(fig)

# defining the x, y, & z of scatter plot
x = list(df_int.iloc[:,0])
y = list(df_int.iloc[:,1])
z = list(df_int.iloc[:,2])

# defining the axis labels
column_names = df_int.columns
ax.set_xlabel(column_names[0])
ax.set_ylabel(column_names[1])
ax.set_zlabel(column_names[2])

# defining the markers, and the color
ax.scatter(x, y, z, c='red', marker='o', s = 50)

plt.show()
```



In [34]:

```
# histogram plot
from matplotlib import pyplot

pyplot.hist(df_int)
pyplot.show()
```

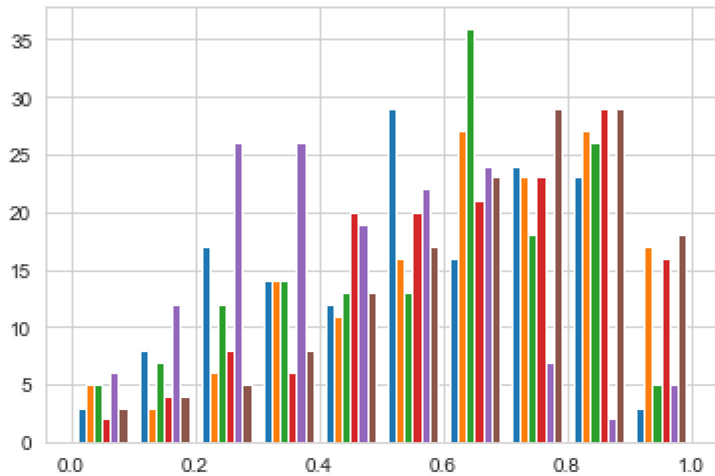


In [35]:

```
#MinMaxScaling

from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import MinMaxScaler
```

```
min_max_scaler = MinMaxScaler()

df_MM = min_max_scaler.fit_transform(df_int)
```

In [36]:
```
pyplot.hist(df_MM)
pyplot.show()
```



In [37]:
```
df_MM = pd.DataFrame(df_MM, columns = df_int.columns)
df_MM
```

Out[37]:

	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	0.826018	0.934978	0.826058	0.957143	0.317195	0.503731
1	0.857542	0.934978	0.850641	0.951020	0.530885	0.503731
2	0.894254	0.908072	0.910342	0.895918	0.522538	0.503731
3	0.846568	1.000000	0.861176	0.969388	0.747913	0.503731
4	0.857342	0.908072	0.840105	0.883673	0.772955	0.444030
...	...	...	...	...	...	...
144	0.257582	0.560538	0.007796	0.479592	0.262104	0.955224
145	0.627893	0.553812	0.378964	0.702041	0.070117	0.742537
146	0.207702	0.033632	0.453802	0.851020	0.582638	0.761194
147	0.260974	0.477578	0.271220	0.402041	0.402337	0.779851
148	0.211492	0.477578	0.141001	0.402041	0.310518	0.972015

149 rows × 6 columns

In [38]:
```
#Applying PCA

rand_state = 1000
from pca import pca
```

In [39]:
```
pca()
model = pca(n_components=3, normalize=False)
pca_results = model.fit_transform(df_MM)
```

```
[pca] >Processing dataframe..
[pca] >The PCA reduction is performed on the [6] columns of the input dataframe.
[pca] >Fitting using PCA..
[pca] >Computing loadings and PCs..
[pca] >Computing explained variance..
[pca] >Outlier detection using Hotelling T2 test with alpha=[0.05] and n_components=[3]
[pca] >Outlier detection using SPE/DmodX with n_std=[2]
```

In [40]:

```
pca_results
```

Out[40]:

```
{'loadings':      Logged GDP per capita  Social support  Healthy life expectancy  \
 PC1              -0.530443       -0.506895                -0.538076
 PC2              -0.174324       -0.160103                -0.111944
 PC3              -0.031800        0.206282                -0.041489

      Freedom to make life choices  Generosity  Perceptions of corruption
 PC1                     -0.343899    0.033481                   0.229726
 PC2                      0.372380    0.691195                  -0.561274
 PC3                      0.373181    0.462028                   0.775875  ,
 'PC':            PC1       PC2       PC3
 0   -0.576043  0.001498 -0.013779
 1   -0.596733  0.138672  0.080644
 2   -0.616022  0.103608  0.047030
 3   -0.628591  0.285844  0.201096
 4   -0.559769  0.319942  0.115927
 ..        ...       ...       ...
 144  0.621672 -0.217182  0.107641
 145  0.097149 -0.252698 -0.091630
 146  0.513656  0.294714  0.118194
 147  0.511260 -0.067497 -0.020726
 148  0.648646 -0.215616  0.092922

 [149 rows x 3 columns],
 'explained_var': array([0.52143042, 0.71782246, 0.84008066]),
 'model': PCA(n_components=3),
 'scaler': None,
 'pcp': 0.8400806569968364,
 'topfeat':     PC                       feature   loading  type
 0  PC1       Healthy life expectancy -0.538076  best
 1  PC2                    Generosity  0.691195  best
 2  PC3     Perceptions of corruption  0.775875  best
 3  PC1         Logged GDP per capita -0.530443  weak
 4  PC1                Social support -0.506895  weak
 5  PC3  Freedom to make life choices  0.373181  weak,
 'outliers':       y_proba    y_score  y_bool  y_bool_spe  y_score_spe
 0    0.416495   6.060067   False       False     0.576045
 1    0.261334   7.694699   False       False     0.612634
 2    0.273679   7.541188   False       False     0.624674
 3    0.106812  10.453385   False       False     0.690531
 4    0.172566   9.018035   False       False     0.644752
 ..        ...        ...     ...         ...          ...
 144  0.176821   8.942797   False       False     0.658517
 145  0.804013   3.038396   False       False     0.270729
 146  0.232849   8.072373   False       False     0.592198
 147  0.480671   5.506505   False       False     0.515696
 148  0.158086   9.286682   False       False     0.683544

 [149 rows x 5 columns],
 'outliers_params': {'paramT2': (-3.9739526608506724e-17, 0.08667571976041136),
  'paramSPE': (array([1.56474386e-17, 4.00499917e-17]),
   array([[ 1.62487010e-01, -2.88808017e-17],
          [-2.88808017e-17,  6.11992581e-02]]))}}
```

In [41]:

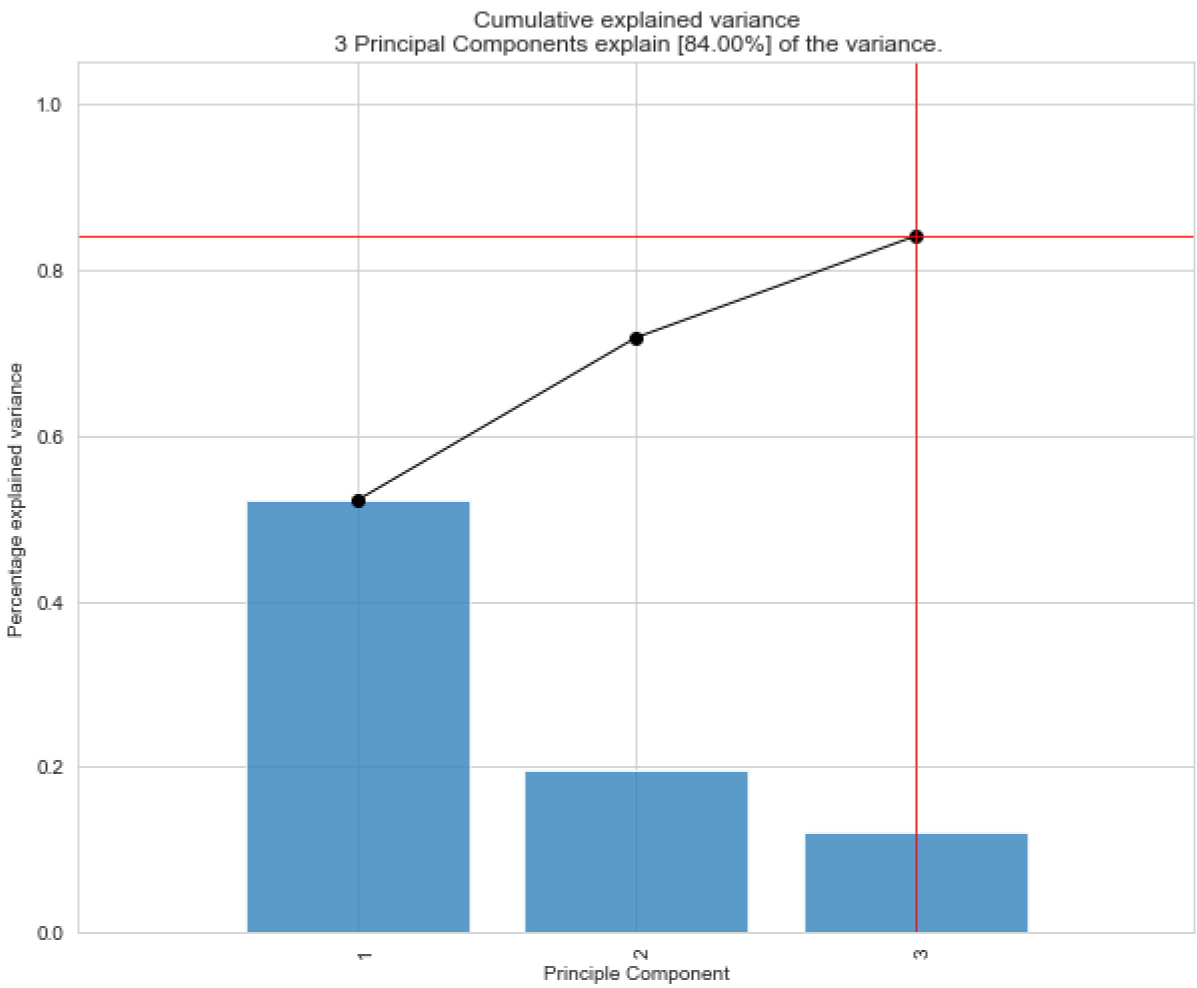
```
#Checking best features

Top_pca = pca_results['topfeat']
Top_pca
```

Out[41]:

	PC	feature	loading	type
0	PC1	Healthy life expectancy	-0.538076	best
1	PC2	Generosity	0.691195	best
2	PC3	Perceptions of corruption	0.775875	best
3	PC1	Logged GDP per capita	-0.530443	weak
4	PC1	Social support	-0.506895	weak
5	PC3	Freedom to make life choices	0.373181	weak

In [42]:

```
#Principle components Graph

model.plot(figsize=(10,8))
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```

In [43]:

```
#Explained variance

pca_results['explained_var']
```

Out[43]:

```
array([0.52143042, 0.71782246, 0.84008066])
```

In [44]:
```
#saving results in a dataframe
pca_df = pd.DataFrame(pca_results['PC'])
```

In [45]:
```
pca_df
```

Out[45]:

	PC1	PC2	PC3
0	-0.576043	0.001498	-0.013779
1	-0.596733	0.138672	0.080644
2	-0.616022	0.103608	0.047030
3	-0.628591	0.285844	0.201096
4	-0.559769	0.319942	0.115927
...	...	...	...
144	0.621672	-0.217182	0.107641
145	0.097149	-0.252698	-0.091630
146	0.513656	0.294714	0.118194
147	0.511260	-0.067497	-0.020726
148	0.648646	-0.215616	0.092922

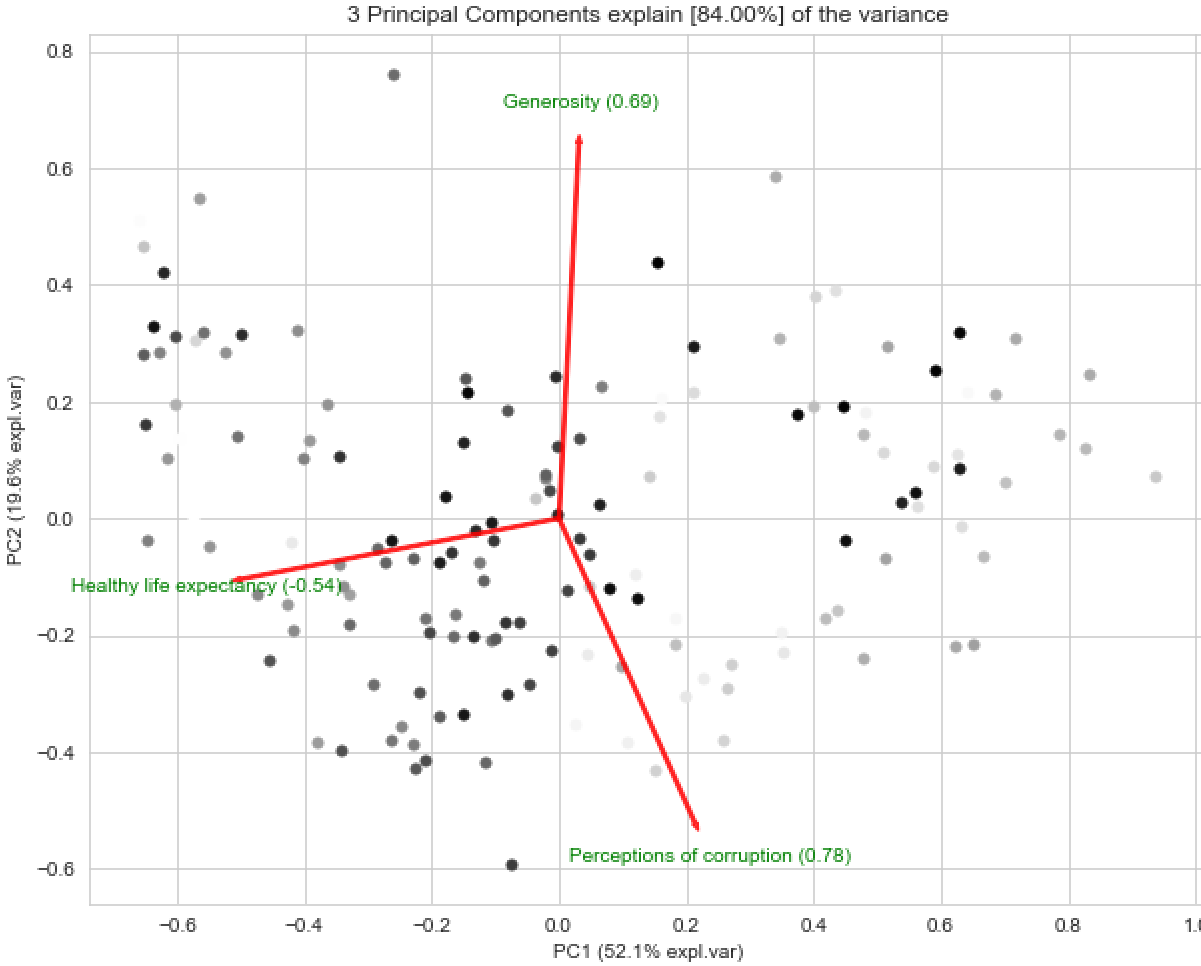
149 rows × 3 columns

In [46]:
```
pca_labels = pca_results['loadings']
pca_labels
```

Out[46]:

	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
PC1	-0.530443	-0.506895	-0.538076	-0.343899	0.033481	0.229726
PC2	-0.174324	-0.160103	-0.111944	0.372380	0.691195	-0.561274
PC3	-0.031800	0.206282	-0.041489	0.373181	0.462028	0.775875

In [47]:
```
#PCA features visualisation

model.biplot(n_feat=3,  legend=False, figsize=(10,8), label=False, cmap='binary')
plt.show()
```



In [48]:

```
from sklearn import metrics

#Kmeans clustering (checking for number of k')

# definying a dictionary
results_dict = {}

# definying how many clusters.
num_of_clusters = 10

# runing through each instance of K
for k in range(2, num_of_clusters):

    print("-"*100)

    # definying a dictionary to hold the results.
    results_dict[k] = {}

    # fiting the training data
    kmeans = KMeans(n_clusters=k, random_state=0).fit(pca_df)

    # definying the silhouette score
    sil_score = metrics.silhouette_score(pca_df, kmeans.labels_, metric='euclidean')

    # storying the different metrics
    results_dict[k]['silhouette_score'] = sil_score
    results_dict[k]['inertia'] = kmeans.inertia_
    results_dict[k]['score'] = kmeans.score
    results_dict[k]['model'] = kmeans

    # printing the results
    print("Number of Clusters: {}".format(k))
    print('silhouette_score', sil_score)
```

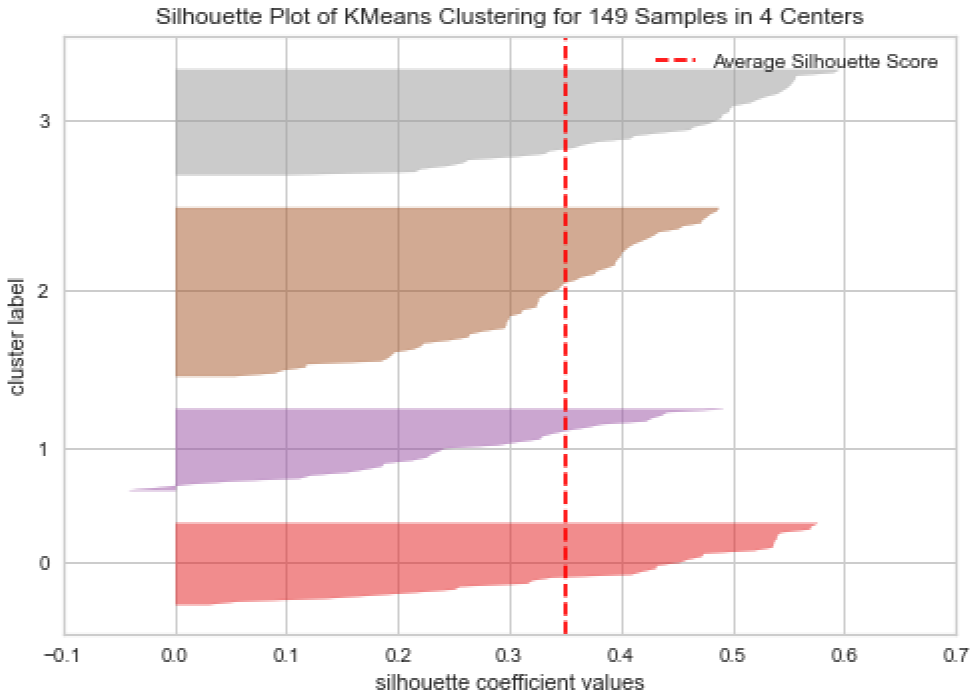
```
    print('inertia', kmeans.inertia_)
```

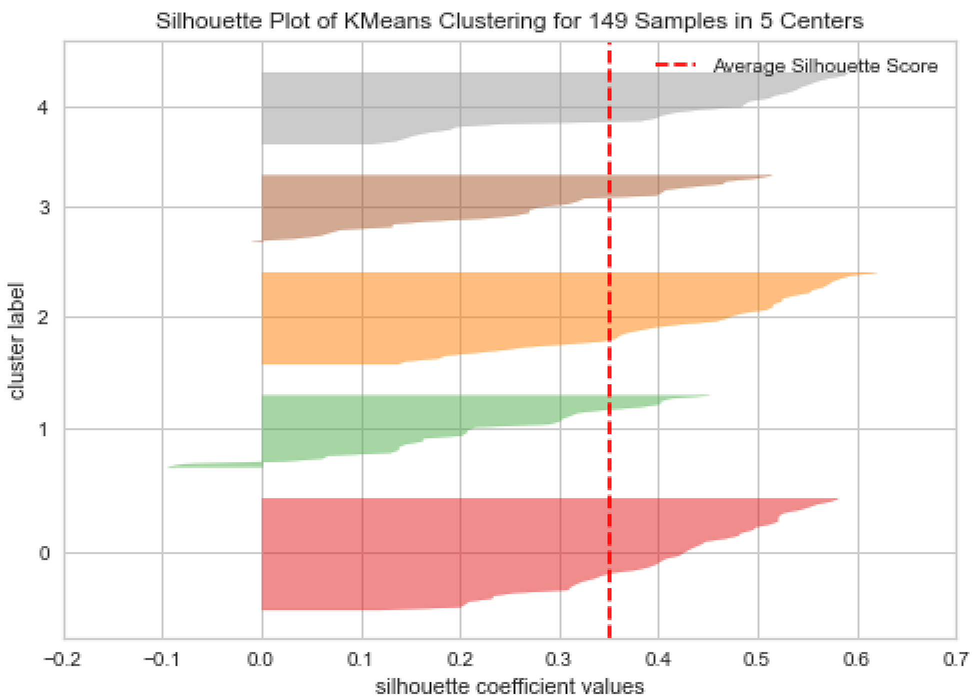
```
------------------------------------------------------------------------------------------
----------
Number of Clusters: 2
silhouette_score 0.39166264988335747
inertia 21.530629512364875
------------------------------------------------------------------------------------------
----------
Number of Clusters: 3
silhouette_score 0.37603260921083437
inertia 14.739353799954817
------------------------------------------------------------------------------------------
----------
Number of Clusters: 4
silhouette_score 0.350254982549967
inertia 11.986304310498515
------------------------------------------------------------------------------------------
----------
Number of Clusters: 5
silhouette_score 0.3516718801376708
inertia 9.818862222300993
------------------------------------------------------------------------------------------
----------
Number of Clusters: 6
silhouette_score 0.3367962572788354
inertia 8.66289514271759
------------------------------------------------------------------------------------------
----------
Number of Clusters: 7
silhouette_score 0.30798899421641435
inertia 7.599402014225908
------------------------------------------------------------------------------------------
----------
Number of Clusters: 8
silhouette_score 0.3055943665139348
inertia 6.9838240413671135
------------------------------------------------------------------------------------------
----------
Number of Clusters: 9
silhouette_score 0.3117324092440835
inertia 6.32641723980883
```

In [49]:

```
from yellowbrick.cluster import SilhouetteVisualizer

clusters = [4,5]

for cluster in clusters:

    print('-'*100)

    # defining the model for K
    kmeans = KMeans(n_clusters = cluster, random_state=0)

    # passing the model through the visualizer
    visualizer = SilhouetteVisualizer(kmeans)

    # fiting the data
    visualizer.fit(pca_df)


    visualizer.poof()
```

```
------------------------------------------------------------------------------------------
```





In [50]:

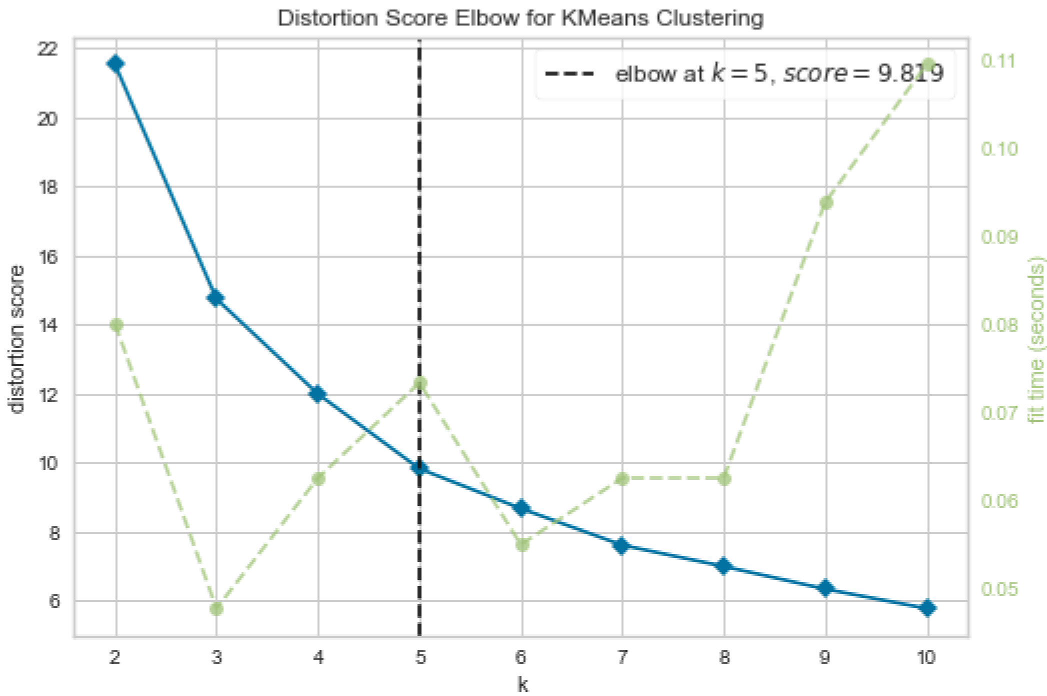
```
from yellowbrick.cluster import KElbowVisualizer
```

In [51]:

```
clusters = [10]

for cluster in clusters:

    print('-'*100)


    kmeans = KMeans(n_clusters = cluster, random_state=0)


    visualizer = KElbowVisualizer(kmeans)
```

```
    visualizer.fit(pca_df)


    visualizer.poof()
```

```
----------------------------------------------------------------------------------------------------
```



In [52]:

```
pca_array = pca_df.values
```

In [53]:

```
pca_array.shape
```

Out[53]:

```
(149, 3)
```

In [54]:

```
df_val = df_MM.values
```

In [55]:

```
df_MM.columns
```

Out[55]:

```
Index(['Logged GDP per capita', 'Social support', 'Healthy life expectancy',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption'],
      dtype='object')
```

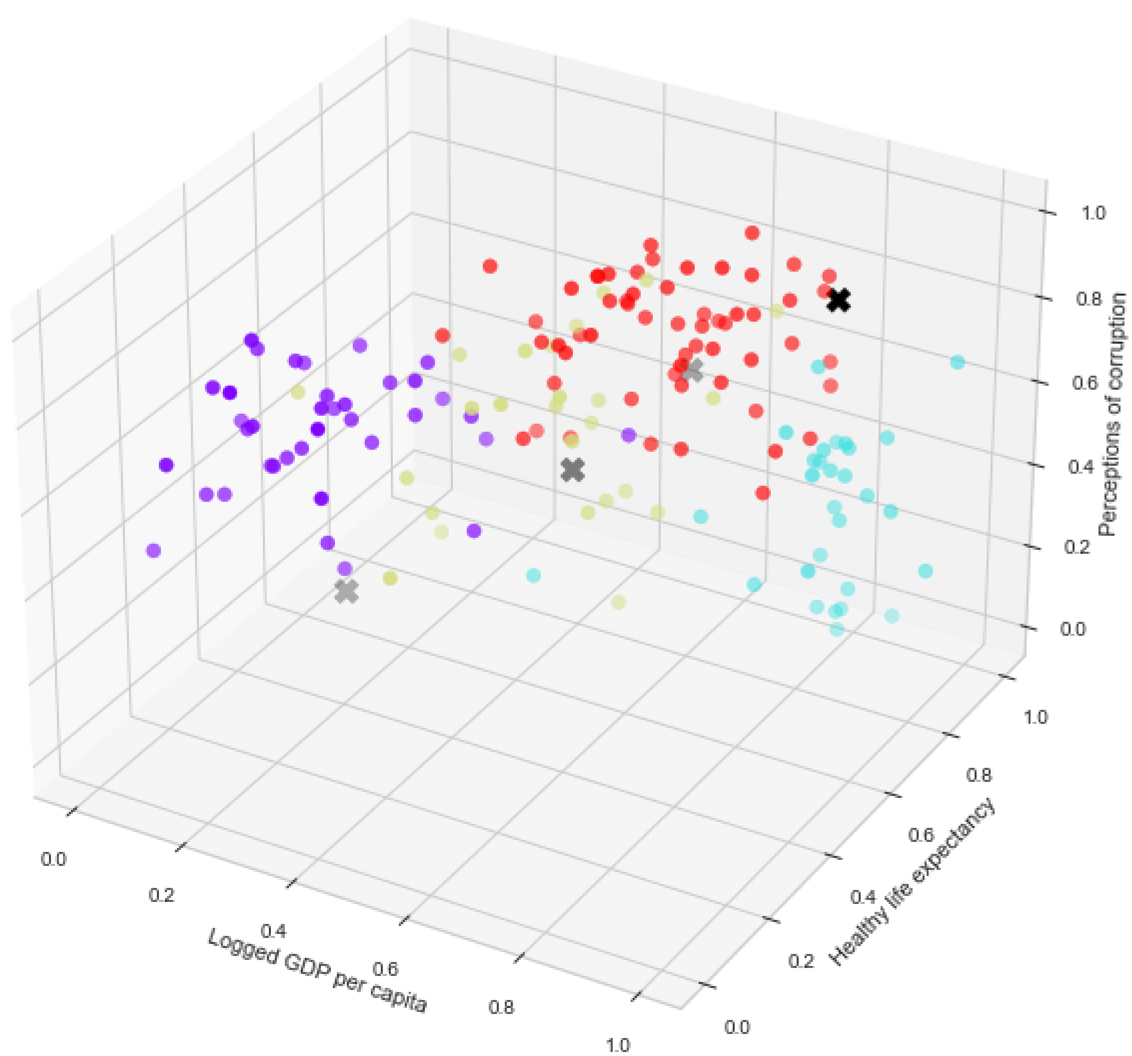
In [56]:

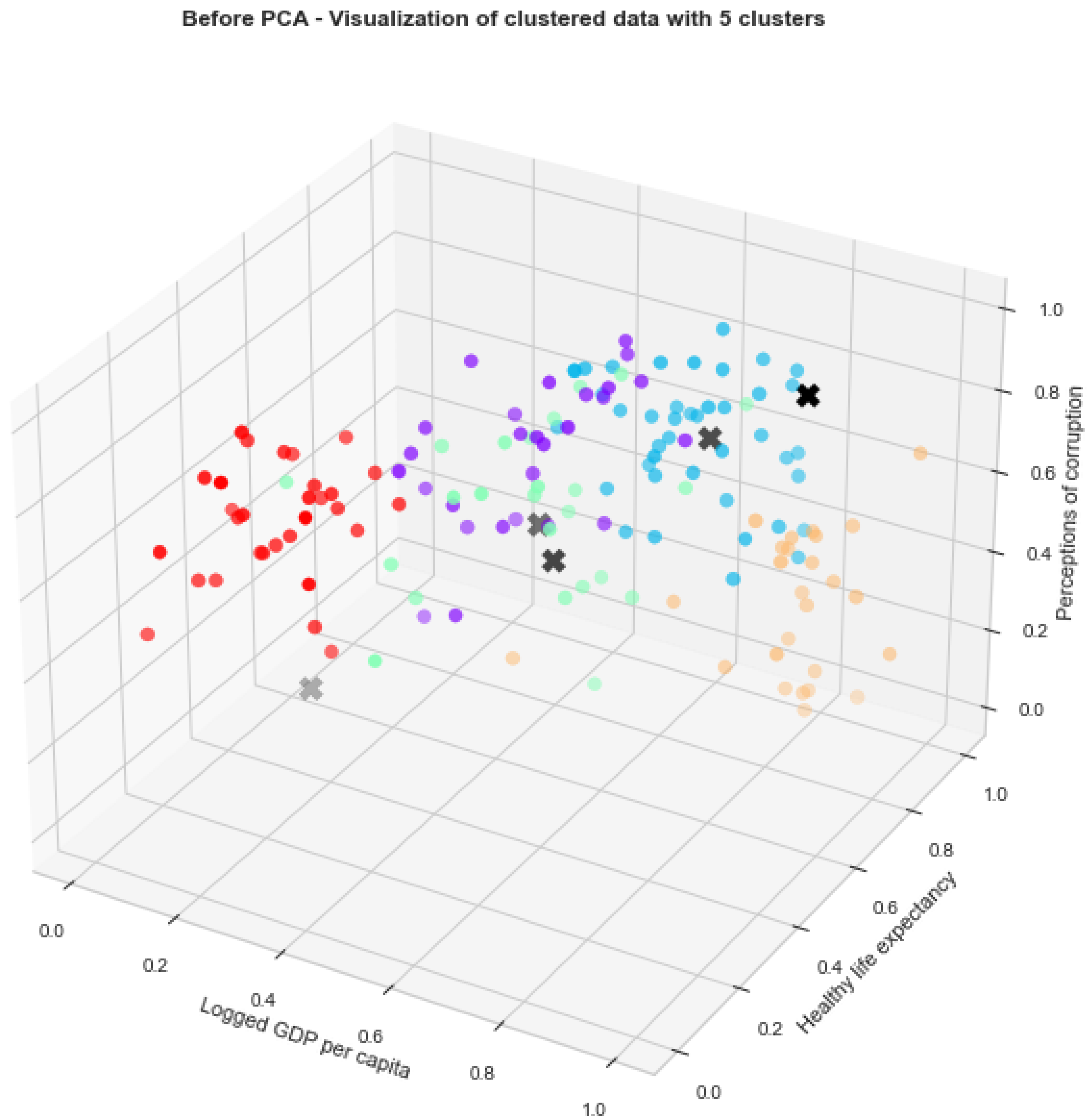
```
#Kmeans w/ data scaled
clusters = [4,5]

for cluster in clusters:

    print('-'*100)

    kmeans = KMeans(n_clusters= cluster, random_state=0).fit(df_val)

    # defining the cluster centers
    cluster_centers = kmeans.cluster_centers_
    C1 = cluster_centers[:, 0]
    C2 = cluster_centers[:, 1]
    C3 = cluster_centers[:, 2]
```

```
# creating a new plot
fig = plt.figure(figsize=(8,10))
ax = Axes3D(fig)

# taking the scaled data
x = df_val[: ,0]
y = df_val[: ,2]
z = df_val[: ,5]

# defining the axes labels
column_names = df_int.columns
ax.set_xlabel(column_names[0])
ax.set_ylabel(column_names[2])
ax.set_zlabel(column_names[5])

# creating a new plot
ax.scatter(x, y, z, c = kmeans.labels_.astype(float), cmap='rainbow', s = 50)
ax.scatter(C1, C2, C3, marker="X",s=150, color='black')

plt.title('Before PCA - Visualization of clustered data with {} clusters'.format(clust

plt.show()
```

```
----------------------------------------------------------------------------------------------------------
----------
```

Before PCA - Visualization of clustered data with 4 clusters

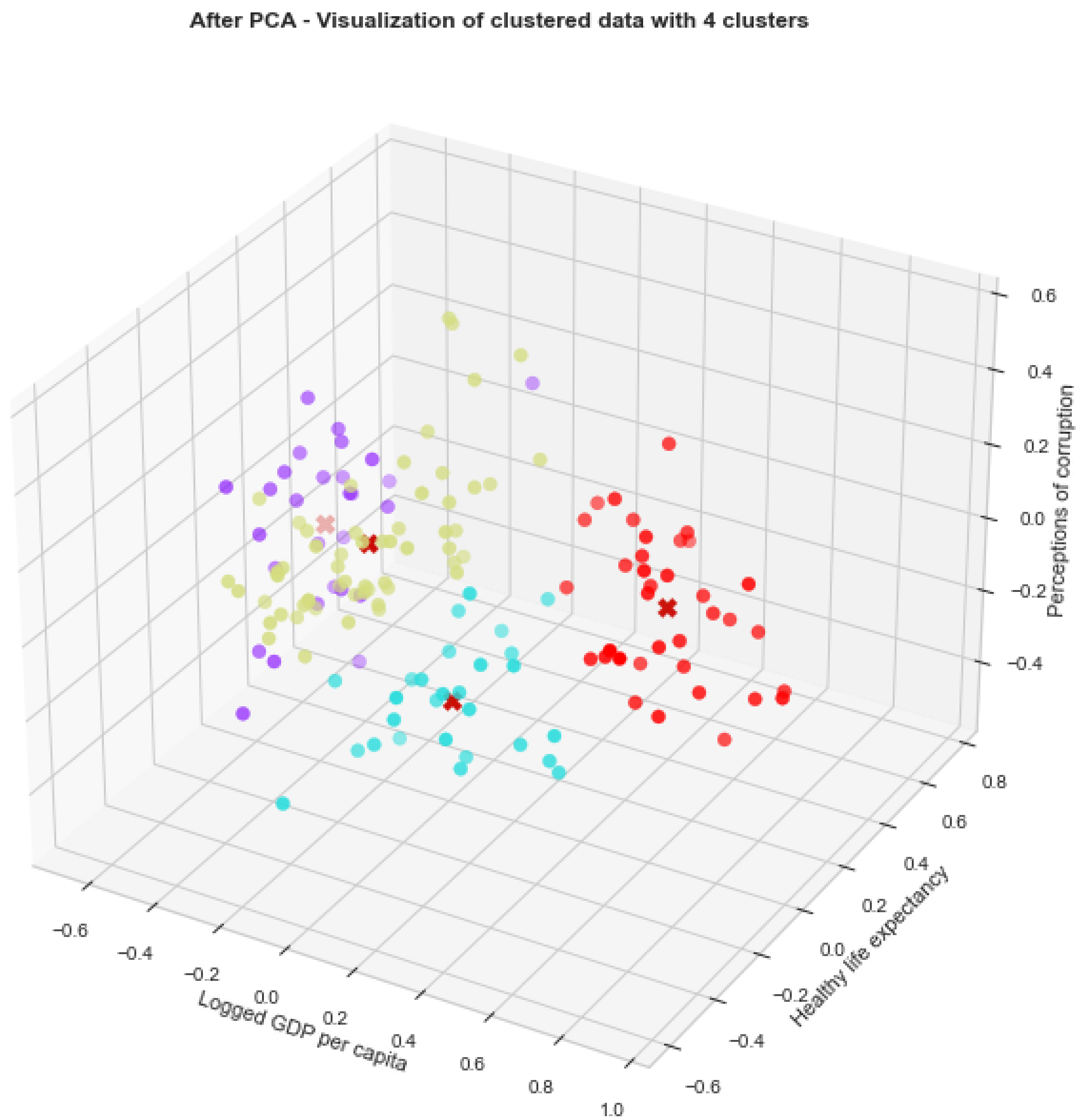


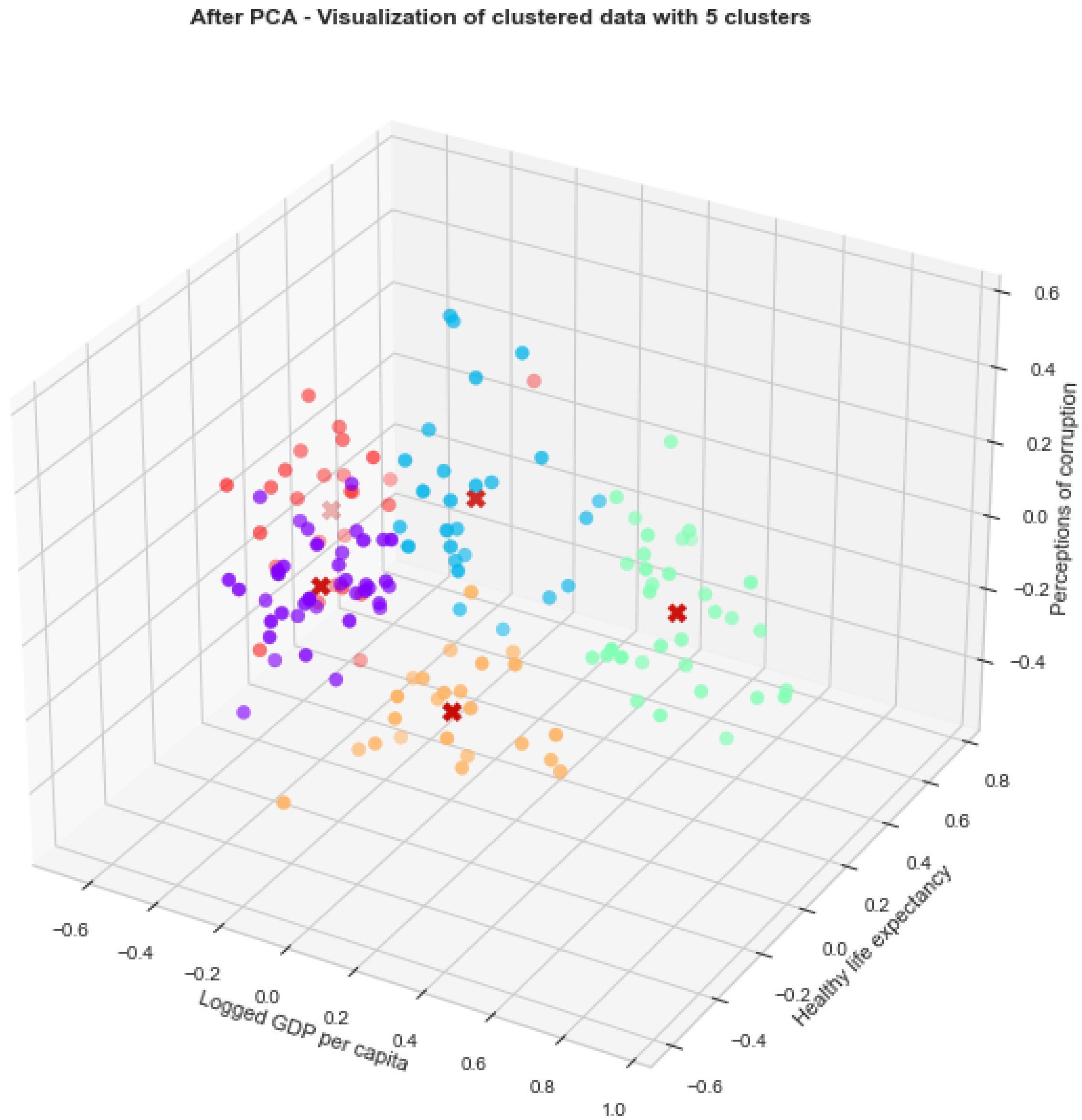


In [57]:

```
clusters = [4,5]

for cluster in clusters:

    print('-'*100)

    kmeans = KMeans(n_clusters= cluster, random_state=0).fit(pca_df)


    cluster_centers = kmeans.cluster_centers_
    C1 = cluster_centers[:, 0]
    C2 = cluster_centers[:, 1]
    C3 = cluster_centers[:, 2]


    fig = plt.figure(figsize=(8,10))
    ax = Axes3D(fig)


    x = pca_df['PC1']
    y = pca_df['PC2']
    z = pca_df['PC3']
```

```
column_names = df_int.columns
ax.set_xlabel(column_names[0])
ax.set_ylabel(column_names[2])
ax.set_zlabel(column_names[5])

ax.scatter(x, y, z, c = kmeans.labels_.astype(float), cmap='rainbow', s=50)
ax.scatter(C1, C2, C3, marker="X",s=100, color='r')

plt.title('After PCA - Visualization of clustered data with {} clusters'.format(cluste

plt.show()
```



In [58]:

```
#predicting the model
kmeans = results_dict[5]['model']

y_kmeans = kmeans.predict(pca_df)
```

In [59]:

```
kmeans.cluster_centers_.shape
```

Out[59]:

```
(5, 3)
```

In [60]:

```
kmeans.cluster_centers_
```

Out[60]:

```
array([[-0.23263423, -0.21273582,  0.04457142],
       [ 0.01996101,  0.08959027,  0.18498086],
       [ 0.57835192,  0.14225062, -0.00402674],
       [ 0.15822935, -0.21580598, -0.18634406],
       [-0.5421821 ,  0.25635225, -0.07643151]])
```

In [61]:

```
pca_df.shape
```

Out[61]:

```
(149, 3)
```

In [62]:
```
cluster_centers = pd.DataFrame(data = kmeans.cluster_centers_, columns = [pca_df])
cluster_centers
```

Out[62]:

	PC1	PC2	PC3
0	-0.232634	-0.212736	0.044571
1	0.019961	0.089590	0.184981
2	0.578352	0.142251	-0.004027
3	0.158229	-0.215806	-0.186344
4	-0.542182	0.256352	-0.076432

In [63]:
```
labels = kmeans.labels_
```

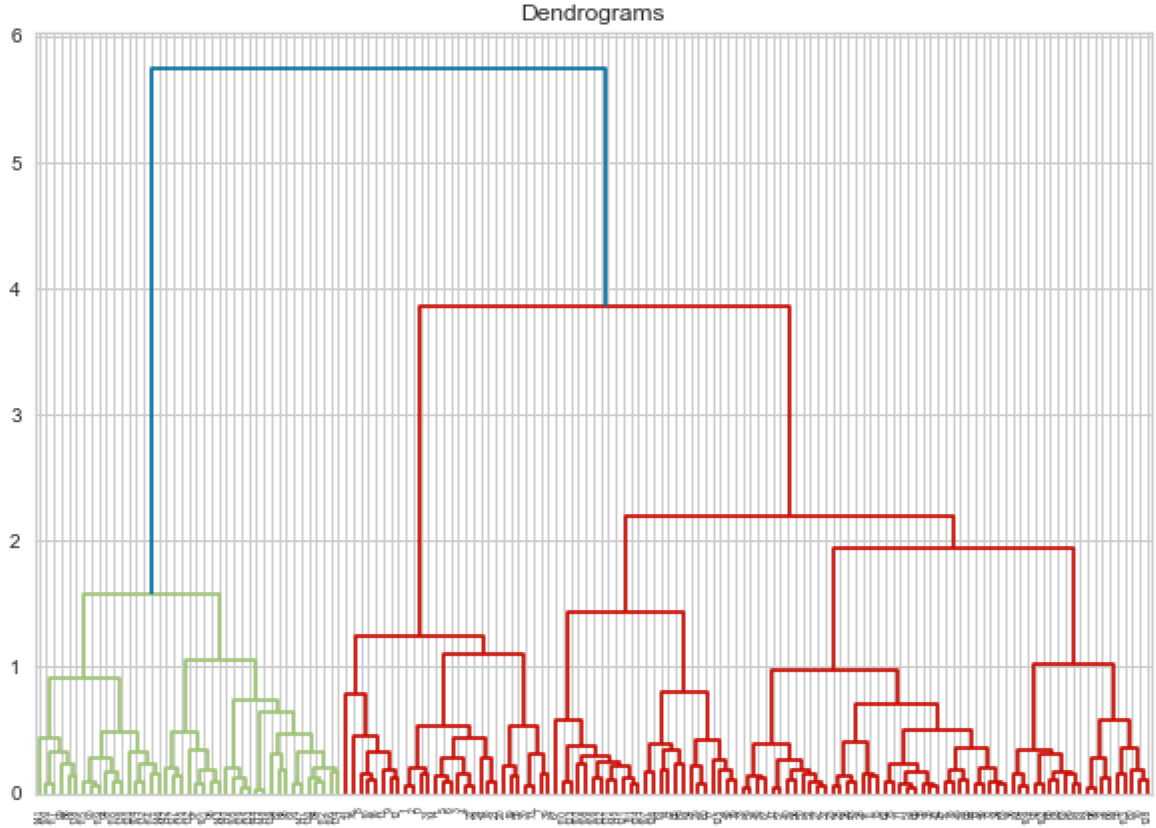
In [64]:
```
labels
```

Out[64]:
```
array([4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 0, 4, 4, 4, 0, 4, 0, 4, 0, 4, 4,
       4, 0, 4, 0, 0, 0, 0, 1, 4, 4, 1, 0, 0, 0, 0, 0, 0, 4, 0, 4, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       1, 3, 1, 1, 1, 0, 0, 4, 3, 0, 4, 1, 1, 3, 1, 1, 2, 0, 2, 3, 1, 0,
       0, 3, 2, 2, 1, 1, 2, 2, 1, 2, 2, 1, 1, 2, 3, 3, 2, 3, 3, 3, 3, 1,
       3, 3, 2, 1, 2, 2, 2, 1, 2, 2, 2, 3, 3, 3, 3, 3, 3, 2, 1, 3, 2, 3,
       2, 3, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 3, 2, 2, 2])
```

In [65]:
```
#CLUSTERING By Hierarchy: Hierarchical Agglomerative Clustering
```

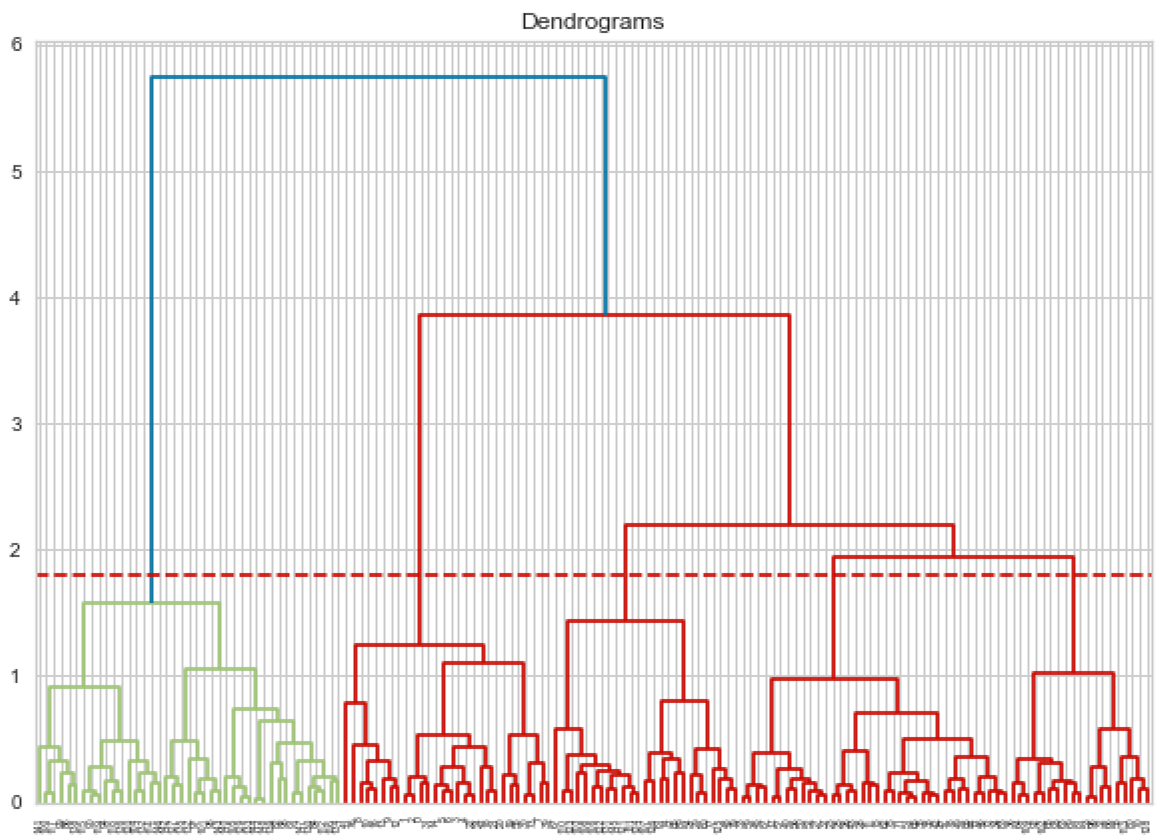
In [66]:
```
pca_h = pca_df.copy()
```

In [67]:
```
import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
hir_df = shc.dendrogram(shc.linkage(pca_h, method='ward'))
```



In [68]:
```
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
hir_df = shc.dendrogram(shc.linkage(pca_h, method='ward'))
plt.axhline(y=1.8, color='r', linestyle='--')
```

Out[68]: <matplotlib.lines.Line2D at 0x20335ed5280>



In [69]:
```
from sklearn.cluster import AgglomerativeClustering
```

```
hir_cluster = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
hir_cluster.fit_predict(pca_h)
```

Out[69]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 4, 2, 4, 2, 2, 2, 2,
       2, 4, 2, 4, 4, 4, 4, 1, 2, 2, 3, 4, 4, 4, 3, 4, 4, 2, 4, 2, 4, 4,
       4, 4, 4, 4, 1, 4, 4, 4, 4, 3, 1, 2, 4, 4, 3, 4, 1, 1, 4, 3, 3, 4,
       3, 1, 3, 3, 3, 4, 4, 2, 1, 4, 2, 1, 4, 1, 3, 3, 0, 4, 0, 1, 0, 4,
       4, 1, 0, 0, 3, 3, 0, 0, 3, 0, 0, 0, 0, 0, 3, 1, 0, 0, 1, 1, 1, 3,
       1, 1, 0, 3, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 3, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0], dtype=int64)
```

In [70]:
```
pca_h
```

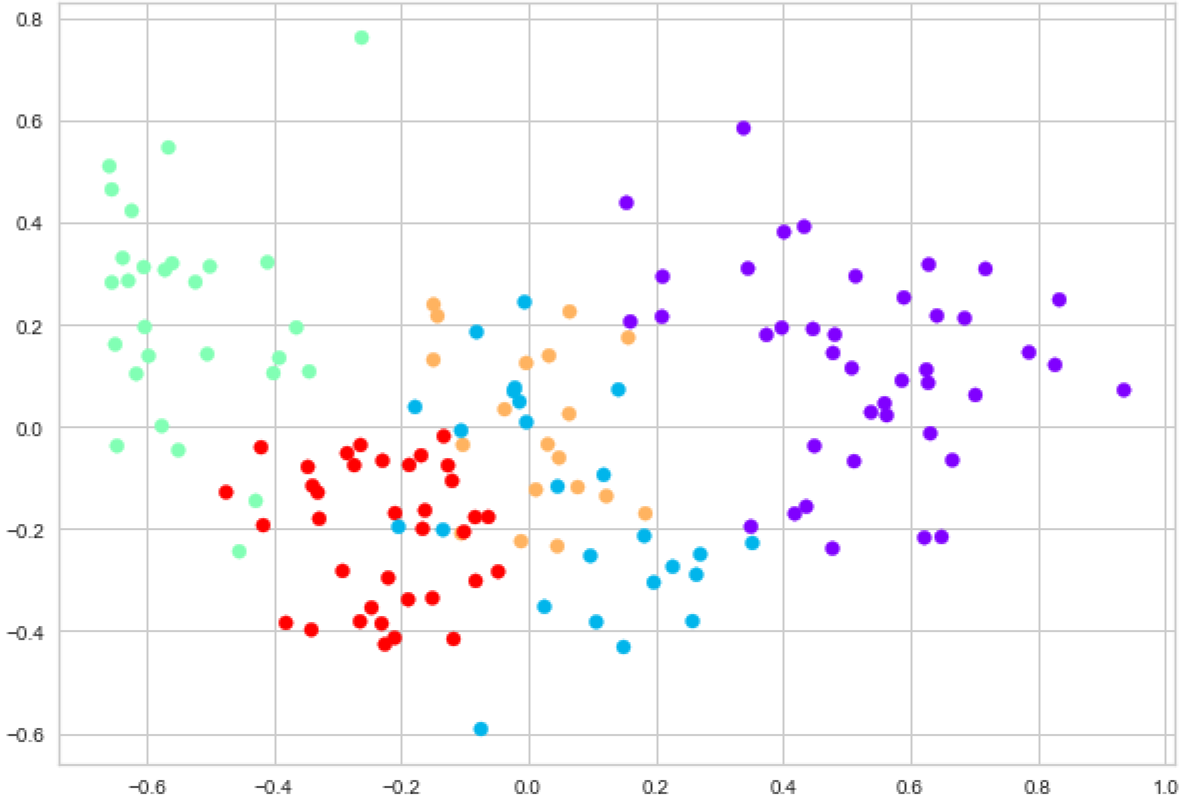
Out[70]:

	PC1	PC2	PC3
0	-0.576043	0.001498	-0.013779
1	-0.596733	0.138672	0.080644
2	-0.616022	0.103608	0.047030
3	-0.628591	0.285844	0.201096
4	-0.559769	0.319942	0.115927
...	...	...	...
144	0.621672	-0.217182	0.107641
145	0.097149	-0.252698	-0.091630
146	0.513656	0.294714	0.118194
147	0.511260	-0.067497	-0.020726
148	0.648646	-0.215616	0.092922

149 rows × 3 columns

In [71]:
```
plt.figure(figsize=(10, 7))
plt.scatter(pca_h['PC1'], pca_h['PC2'], c=hir_cluster.labels_, cmap='rainbow')
```

Out[71]:
```
<matplotlib.collections.PathCollection at 0x2033528ee20>
```



In [72]:

```
#VISUALISATION - Interpreting results
```

In [78]:

```
final_df = pd.concat([pca_df, pd.DataFrame({'Cluster':labels})], axis =1)
pca_df = pd.concat([pca_df, Country[["Country name"]]], axis=1)
final_df
```

Out[78]:

	PC1	PC2	PC3	Country name	Cluster
0	-0.576043	0.001498	-0.013779	Finland	4
1	-0.596733	0.138672	0.080644	Denmark	4
2	-0.616022	0.103608	0.047030	Switzerland	4
3	-0.628591	0.285844	0.201096	Iceland	4
4	-0.559769	0.319942	0.115927	Netherlands	4
...	...	...	...	...	...
144	0.621672	-0.217182	0.107641	Lesotho	2
145	0.097149	-0.252698	-0.091630	Botswana	3
146	0.513656	0.294714	0.118194	Rwanda	2
147	0.511260	-0.067497	-0.020726	Zimbabwe	2
148	0.648646	-0.215616	0.092922	Afghanistan	2

149 rows × 5 columns

In [79]:

```
Top_pca
```

Out[79]:

PC	feature	loading	type

	PC	feature	loading	type
0	PC1	Healthy life expectancy	-0.538076	best
1	PC2	Generosity	0.691195	best
2	PC3	Perceptions of corruption	0.775875	best
3	PC1	Logged GDP per capita	-0.530443	weak
4	PC1	Social support	-0.506895	weak
5	PC3	Freedom to make life choices	0.373181	weak

In [80]:

```
happy_f = final_df.rename(columns={'PC1':'Healthy life expectancy',  'PC2':'Generosity', '
```

In [81]:

```
happy_f
```

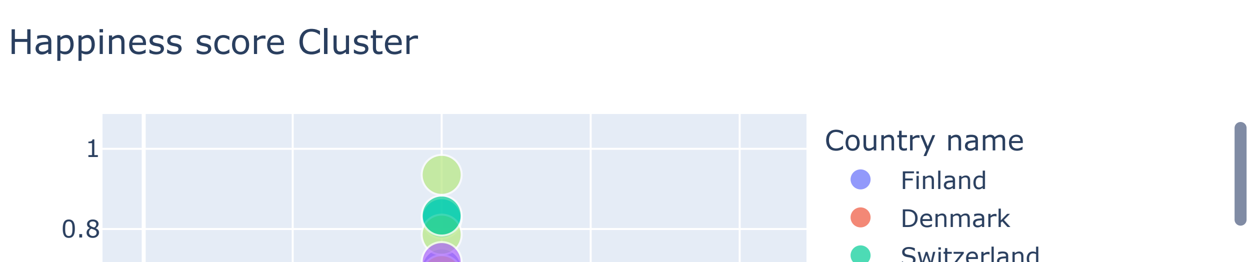
Out[81]:

	Healthy life expectancy	Generosity	Perceptions of corruption	Country name	Cluster
0	-0.576043	0.001498	-0.013779	Finland	4
1	-0.596733	0.138672	0.080644	Denmark	4
2	-0.616022	0.103608	0.047030	Switzerland	4
3	-0.628591	0.285844	0.201096	Iceland	4
4	-0.559769	0.319942	0.115927	Netherlands	4
...	...	...	...	...	...
144	0.621672	-0.217182	0.107641	Lesotho	2
145	0.097149	-0.252698	-0.091630	Botswana	3
146	0.513656	0.294714	0.118194	Rwanda	2
147	0.511260	-0.067497	-0.020726	Zimbabwe	2
148	0.648646	-0.215616	0.092922	Afghanistan	2

149 rows × 5 columns

In [82]:

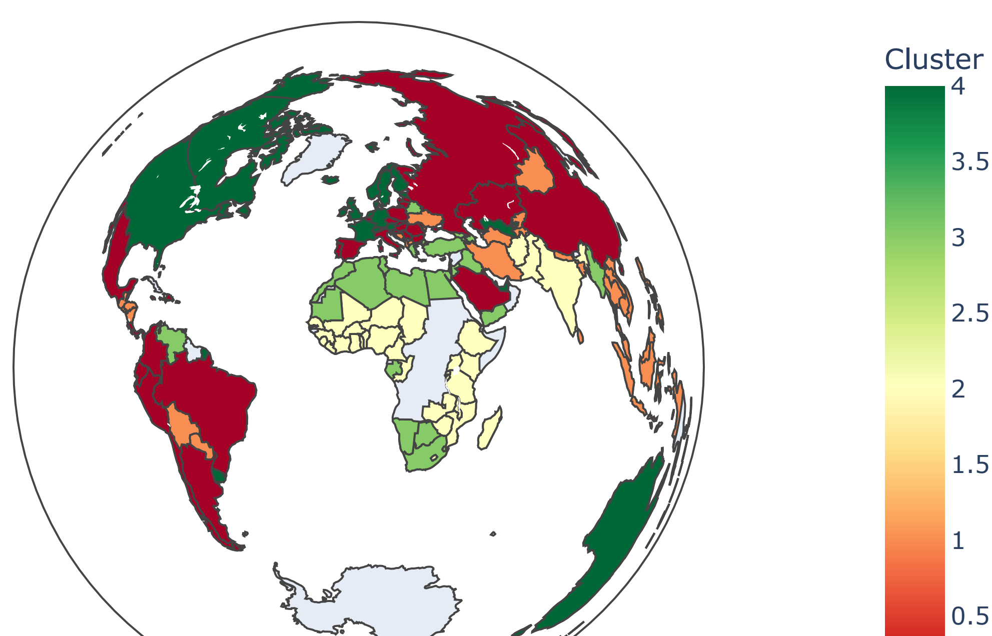
```
fig = px.scatter(happy_f, x = 'Cluster', y = 'Healthy life expectancy',
                 size = 'Cluster', color = 'Country name', hover_name = 'Country name',
                 trendline = 'ols')

fig.update_layout(
title_text = 'Happiness score Cluster'

)

fig.show()
```

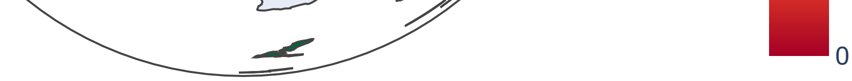


In [83]:

```
data = dict(type = 'choropleth',
            locations = happy_f['Country name'],
            locationmode = 'country names',
            colorscale= 'RdYlGn',
            z = happy_f['Cluster'],
            text = happy_f['Country name'],
            colorbar = {'title': 'Cluster'})

layout = dict(title = 'Geographical Visualisation of Clusters',
             geo = dict(showframe = True, projection = {'type': 'azimuthal equal area'}))

choromap3 = go.Figure(data = [data], layout=layout)
iplot(choromap3)
```

Geographical Visualisation of Clusters





In [84]:

```
Group_df = happy_f.groupby(['Cluster', 'Country name']).sum()
pd.set_option('display.max_rows', Group_df.shape[0]+1)
print(Group_df)
```

```
                                     Healthy life expectancy  Generosity  \
Cluster Country name
0       Argentina                                  -0.220434   -0.295804
        Belgium                                    -0.428472   -0.145068
        Brazil                                     -0.126553   -0.075714
        Bulgaria                                   -0.150979   -0.335448
        Chile                                      -0.166325   -0.199710
        China                                      -0.187361   -0.074829
        Colombia                                   -0.101420   -0.205738
        Costa Rica                                 -0.331131   -0.128520
        Croatia                                    -0.210451   -0.413416
        Cyprus                                     -0.209624   -0.169118
        Czech Republic                             -0.381072   -0.383964
        Dominican Republic                         -0.168746   -0.056173
        Ecuador                                    -0.063181   -0.176661
        Hungary                                    -0.225581   -0.425873
        Israel                                     -0.420159   -0.040221
        Italy                                      -0.246696   -0.354222
        Jamaica                                    -0.105663   -0.208692
        Japan                                      -0.454271   -0.244106
        Kazakhstan                                 -0.284944   -0.051808
        Kuwait                                     -0.273745   -0.075277
        Latvia                                     -0.188997   -0.338216
        Lithuania                                  -0.264648   -0.381115
        Maldives                                   -0.263914   -0.035858
        Mauritius                                  -0.229319   -0.066417
        Mexico                                     -0.162677   -0.163709
        Moldova                                    -0.011725   -0.224107
        Montenegro                                 -0.083893   -0.176874
        Panama                                     -0.292103   -0.282280
        Peru                                       -0.047704   -0.283571
        Poland                                     -0.328754   -0.180007
        Portugal                                   -0.341113   -0.397505
        Romania                                    -0.117842   -0.415523
        Russia                                     -0.083058   -0.301787
        Saudi Arabia                               -0.346675   -0.078641
        Serbia                                     -0.120136   -0.105894
        Slovakia                                   -0.230738   -0.385212
        Slovenia                                   -0.475016   -0.128166
        South Korea                                -0.204269   -0.195763
        Spain                                      -0.416716   -0.192774
        Taiwan Province of China                   -0.339266   -0.115650
1       Albania                                     0.122198   -0.135714
        Bangladesh                                  0.159864    0.205856
        Bolivia                                     0.047904   -0.060665
        Bosnia and Herzegovina                      0.011490   -0.123151
        Cambodia                                    0.156537    0.174555
        El Salvador                                -0.021335    0.076376
        Guatemala                                  -0.023227    0.069439
        Honduras                                    0.032131    0.139152
        Indonesia                                   0.063770    0.025325
        Iran                                        0.209591    0.215479
```

	Kosovo	0.064588	0.225442
	Kyrgyzstan	-0.003695	0.124755
	Laos	0.153833	0.438524
	Malaysia	-0.149440	0.131069
	Mongolia	0.030126	-0.034351
	Nepal	0.210617	0.294012
	Nicaragua	-0.081569	0.185545
	North Macedonia	0.077130	-0.118204
	Paraguay	-0.103151	-0.035407
	Philippines	-0.014248	0.049000
	Sri Lanka	-0.037633	0.034167
	Tajikistan	-0.006257	0.244148
	Thailand	-0.149181	0.239401
	Turkmenistan	-0.143064	0.216900
	Ukraine	0.045160	-0.233768
	Vietnam	-0.133153	-0.018540
2	Afghanistan	0.648646	-0.215616
	Benin	0.446740	0.191404
	Burkina Faso	0.624897	0.111750
	Burundi	0.684918	0.212548
	Cameroon	0.559195	0.045866
	Chad	0.935060	0.071757
	Comoros	0.701787	0.062149
	Congo (Brazzaville)	0.537859	0.028464
	Ethiopia	0.397667	0.193815
	Gambia	0.589448	0.252936
	Ghana	0.373648	0.179846
	Guinea	0.641399	0.217340
	Haiti	0.833617	0.248686
	India	0.344505	0.309907
	Ivory Coast	0.627651	0.086019
	Kenya	0.401191	0.381253
	Lesotho	0.621672	-0.217182
	Liberia	0.586210	0.090527
	Madagascar	0.665597	-0.065382
	Malawi	0.717608	0.309053
	Mali	0.630990	-0.012995
	Mozambique	0.432852	0.391712
	Niger	0.628557	0.317614
	Nigeria	0.562447	0.022388
	Pakistan	0.481022	0.180429
	Rwanda	0.513656	0.294714
	Senegal	0.449005	-0.037605
	Sierra Leone	0.786028	0.145611
	Tanzania	0.337721	0.584455
	Togo	0.826904	0.121137
	Uganda	0.507629	0.114747
	Zambia	0.478224	0.144420
	Zimbabwe	0.511260	-0.067497
3	Algeria	0.226408	-0.273902
	Armenia	-0.105866	-0.007556
	Azerbaijan	-0.178231	0.038784
	Belarus	-0.134308	-0.201725
	Botswana	0.097149	-0.252698
	Egypt	0.181788	-0.213651
	Gabon	0.196647	-0.304760
	Georgia	0.118007	-0.094103
	Greece	-0.074758	-0.591805
	Iraq	0.351529	-0.227587
	Jordan	0.045580	-0.116871
	Lebanon	0.149216	-0.431210
	Libya	-0.003411	0.009028
	Mauritania	0.418122	-0.170252
	Morocco	0.349290	-0.195453
	Myanmar	0.141149	0.072764
	Namibia	0.269734	-0.249629

	Palestinian Territories	0.263577	-0.289504
	South Africa	0.183503	-0.169729
	Swaziland	0.436130	-0.156309
	Tunisia	0.257567	-0.380550
	Turkey	0.024961	-0.352267
	Venezuela	0.106384	-0.382099
	Yemen	0.477338	-0.238257
4	Australia	-0.658334	0.510174
	Austria	-0.637432	0.330454
	Bahrain	-0.364631	0.194292
	Canada	-0.654603	0.464547
	Denmark	-0.596733	0.138672
	Estonia	-0.504633	0.142225
	Finland	-0.576043	0.001498
	France	-0.550006	-0.045834
	Germany	-0.571162	0.307015
	Hong Kong S.A.R. of China	-0.500498	0.313899
	Iceland	-0.628591	0.285844
	Ireland	-0.602889	0.195332
	Luxembourg	-0.649194	0.161291
	Malta	-0.523213	0.283260
	Netherlands	-0.559769	0.319942
	New Zealand	-0.623280	0.422793
	North Cyprus	-0.344465	0.108174
	Norway	-0.654505	0.282494
	Singapore	-0.646359	-0.037474
	Sweden	-0.604330	0.312533
	Switzerland	-0.616022	0.103608
	United Arab Emirates	-0.410100	0.322066
	United Kingdom	-0.565617	0.546718
	United States	-0.391566	0.134890
	Uruguay	-0.400705	0.105007
	Uzbekistan	-0.262054	0.761740

		Perceptions of corruption
Cluster	Country name	
0	Argentina	0.045957
	Belgium	-0.244505
	Brazil	-0.001073
	Bulgaria	0.241520
	Chile	0.071564
	China	-0.022018
	Colombia	0.079633
	Costa Rica	0.127478
	Croatia	0.198741
	Cyprus	0.084307
	Czech Republic	0.114227
	Dominican Republic	-0.081156
	Ecuador	0.083179
	Hungary	0.067829
	Israel	0.080189
	Italy	0.045540
	Jamaica	0.197658
	Japan	-0.340232
	Kazakhstan	0.035245
	Kuwait	-0.033838
	Latvia	-0.059951
	Lithuania	-0.008409
	Maldives	0.217094
	Mauritius	0.115546
	Mexico	0.018436
	Moldova	0.231496
	Montenegro	0.018954
	Panama	0.119317
	Peru	0.119258
	Poland	-0.078023

	Portugal	0.106650
	Romania	0.150735
	Russia	0.040002
	Saudi Arabia	-0.113101
	Serbia	0.141380
	Slovakia	0.169635
	Slovenia	0.176088
	South Korea	-0.208676
	Spain	-0.052439
	Taiwan Province of China	-0.071377
1	Albania	0.137275
	Bangladesh	-0.107120
	Bolivia	0.142519
	Bosnia and Herzegovina	0.311719
	Cambodia	0.284210
	El Salvador	-0.117507
	Guatemala	0.103371
	Honduras	0.205752
	Indonesia	0.263766
	Iran	-0.067385
	Kosovo	0.511971
	Kyrgyzstan	0.480158
	Laos	0.031328
	Malaysia	0.304570
	Mongolia	0.246417
	Nepal	0.074943
	Nicaragua	-0.043475
	North Macedonia	0.223612
	Paraguay	0.319439
	Philippines	0.030499
	Sri Lanka	0.271915
	Tajikistan	-0.259393
	Thailand	0.537360
	Turkmenistan	0.563427
	Ukraine	0.233609
	Vietnam	0.126522
2	Afghanistan	0.092922
	Benin	-0.199084
	Burkina Faso	-0.114784
	Burundi	-0.354249
	Cameroon	0.099643
	Chad	-0.093414
	Comoros	-0.136954
	Congo (Brazzaville)	-0.190001
	Ethiopia	0.053021
	Gambia	-0.010235
	Ghana	0.231793
	Guinea	-0.024514
	Haiti	-0.201410
	India	0.101046
	Ivory Coast	-0.031171
	Kenya	0.282482
	Lesotho	0.107641
	Liberia	0.130092
	Madagascar	-0.140238
	Malawi	-0.104703
	Mali	0.007079
	Mozambique	0.024777
	Niger	-0.099153
	Nigeria	0.172789
	Pakistan	0.026508
	Rwanda	0.118194
	Senegal	-0.060054
	Sierra Leone	0.130278
	Tanzania	-0.102103
	Togo	-0.153046

```
           Uganda                                    0.199388
           Zambia                                    0.125304
           Zimbabwe                                 -0.020726
3          Algeria                                  -0.285176
           Armenia                                  -0.282214
           Azerbaijan                               -0.492430
           Belarus                                  -0.377458
           Botswana                                 -0.091630
           Egypt                                    -0.125138
           Gabon                                    -0.074097
           Georgia                                  -0.386368
           Greece                                   -0.281025
           Iraq                                      0.000554
           Jordan                                   -0.237528
           Lebanon                                  -0.026686
           Libya                                    -0.186834
           Mauritania                               -0.269243
           Morocco                                  -0.230133
           Myanmar                                  -0.188137
           Namibia                                  -0.006927
           Palestinian Territories                  -0.105580
           South Africa                              0.115725
           Swaziland                                -0.303635
           Tunisia                                  -0.136286
           Turkey                                   -0.231045
           Venezuela                                -0.124490
           Yemen                                    -0.146476
4          Australia                                -0.185958
           Austria                                  -0.226919
           Bahrain                                   0.144881
           Canada                                   -0.284476
           Denmark                                   0.080644
           Estonia                                  -0.261237
           Finland                                  -0.013779
           France                                   -0.303958
           Germany                                  -0.319324
           Hong Kong S.A.R. of China                -0.517059
           Iceland                                   0.201096
           Ireland                                  -0.027825
           Luxembourg                               -0.209367
           Malta                                     0.108404
           Netherlands                               0.115927
           New Zealand                              -0.085644
           North Cyprus                             -0.177732
           Norway                                    0.045623
           Singapore                                 0.117920
           Sweden                                   -0.024717
           Switzerland                               0.047030
           United Arab Emirates                     -0.061693
           United Kingdom                           -0.146330
           United States                             0.076529
           Uruguay                                  -0.173715
           Uzbekistan                                0.094459
```

In [85]:

```
Group_df
```

Out[85]:

Cluster	Country name	Healthy life expectancy	Generosity	Perceptions of corruption
0	**Argentina**	-0.220434	-0.295804	0.045957
	Belgium	-0.428472	-0.145068	-0.244505
	Brazil	-0.126553	-0.075714	-0.001073

Cluster	Country name	Healthy life expectancy	Generosity	Perceptions of corruption
	Bulgaria	-0.150979	-0.335448	0.241520
	Chile	-0.166325	-0.199710	0.071564
	China	-0.187361	-0.074829	-0.022018
	Colombia	-0.101420	-0.205738	0.079633
	Costa Rica	-0.331131	-0.128520	0.127478
	Croatia	-0.210451	-0.413416	0.198741
	Cyprus	-0.209624	-0.169118	0.084307
	Czech Republic	-0.381072	-0.383964	0.114227
	Dominican Republic	-0.168746	-0.056173	-0.081156
	Ecuador	-0.063181	-0.176661	0.083179
	Hungary	-0.225581	-0.425873	0.067829
	Israel	-0.420159	-0.040221	0.080189
	Italy	-0.246696	-0.354222	0.045540
	Jamaica	-0.105663	-0.208692	0.197658
	Japan	-0.454271	-0.244106	-0.340232
	Kazakhstan	-0.284944	-0.051808	0.035245
	Kuwait	-0.273745	-0.075277	-0.033838
	Latvia	-0.188997	-0.338216	-0.059951
	Lithuania	-0.264648	-0.381115	-0.008409
	Maldives	-0.263914	-0.035858	0.217094
	Mauritius	-0.229319	-0.066417	0.115546
	Mexico	-0.162677	-0.163709	0.018436
	Moldova	-0.011725	-0.224107	0.231496
	Montenegro	-0.083893	-0.176874	0.018954
	Panama	-0.292103	-0.282280	0.119317
	Peru	-0.047704	-0.283571	0.119258
	Poland	-0.328754	-0.180007	-0.078023
	Portugal	-0.341113	-0.397505	0.106650
	Romania	-0.117842	-0.415523	0.150735
	Russia	-0.083058	-0.301787	0.040002
	Saudi Arabia	-0.346675	-0.078641	-0.113101
	Serbia	-0.120136	-0.105894	0.141380
	Slovakia	-0.230738	-0.385212	0.169635
	Slovenia	-0.475016	-0.128166	0.176088
	South Korea	-0.204269	-0.195763	-0.208676

Cluster	Country name	Healthy life expectancy	Generosity	Perceptions of corruption
	Spain	-0.416716	-0.192774	-0.052439
	Taiwan Province of China	-0.339266	-0.115650	-0.071377
1	Albania	0.122198	-0.135714	0.137275
	Bangladesh	0.159864	0.205856	-0.107120
	Bolivia	0.047904	-0.060665	0.142519
	Bosnia and Herzegovina	0.011490	-0.123151	0.311719
	Cambodia	0.156537	0.174555	0.284210
	El Salvador	-0.021335	0.076376	-0.117507
	Guatemala	-0.023227	0.069439	0.103371
	Honduras	0.032131	0.139152	0.205752
	Indonesia	0.063770	0.025325	0.263766
	Iran	0.209591	0.215479	-0.067385
	Kosovo	0.064588	0.225442	0.511971
	Kyrgyzstan	-0.003695	0.124755	0.480158
	Laos	0.153833	0.438524	0.031328
	Malaysia	-0.149440	0.131069	0.304570
	Mongolia	0.030126	-0.034351	0.246417
	Nepal	0.210617	0.294012	0.074943
	Nicaragua	-0.081569	0.185545	-0.043475
	North Macedonia	0.077130	-0.118204	0.223612
	Paraguay	-0.103151	-0.035407	0.319439
	Philippines	-0.014248	0.049000	0.030499
	Sri Lanka	-0.037633	0.034167	0.271915
	Tajikistan	-0.006257	0.244148	-0.259393
	Thailand	-0.149181	0.239401	0.537360
	Turkmenistan	-0.143064	0.216900	0.563427
	Ukraine	0.045160	-0.233768	0.233609
	Vietnam	-0.133153	-0.018540	0.126522
2	Afghanistan	0.648646	-0.215616	0.092922
	Benin	0.446740	0.191404	-0.199084
	Burkina Faso	0.624897	0.111750	-0.114784
	Burundi	0.684918	0.212548	-0.354249
	Cameroon	0.559195	0.045866	0.099643
	Chad	0.935060	0.071757	-0.093414
	Comoros	0.701787	0.062149	-0.136954

Cluster	Country name	Healthy life expectancy	Generosity	Perceptions of corruption
	Congo (Brazzaville)	0.537859	0.028464	-0.190001
	Ethiopia	0.397667	0.193815	0.053021
	Gambia	0.589448	0.252936	-0.010235
	Ghana	0.373648	0.179846	0.231793
	Guinea	0.641399	0.217340	-0.024514
	Haiti	0.833617	0.248686	-0.201410
	India	0.344505	0.309907	0.101046
	Ivory Coast	0.627651	0.086019	-0.031171
	Kenya	0.401191	0.381253	0.282482
	Lesotho	0.621672	-0.217182	0.107641
	Liberia	0.586210	0.090527	0.130092
	Madagascar	0.665597	-0.065382	-0.140238
	Malawi	0.717608	0.309053	-0.104703
	Mali	0.630990	-0.012995	0.007079
	Mozambique	0.432852	0.391712	0.024777
	Niger	0.628557	0.317614	-0.099153
	Nigeria	0.562447	0.022388	0.172789
	Pakistan	0.481022	0.180429	0.026508
	Rwanda	0.513656	0.294714	0.118194
	Senegal	0.449005	-0.037605	-0.060054
	Sierra Leone	0.786028	0.145611	0.130278
	Tanzania	0.337721	0.584455	-0.102103
	Togo	0.826904	0.121137	-0.153046
	Uganda	0.507629	0.114747	0.199388
	Zambia	0.478224	0.144420	0.125304
	Zimbabwe	0.511260	-0.067497	-0.020726
3	**Algeria**	0.226408	-0.273902	-0.285176
	Armenia	-0.105866	-0.007556	-0.282214
	Azerbaijan	-0.178231	0.038784	-0.492430
	Belarus	-0.134308	-0.201725	-0.377458
	Botswana	0.097149	-0.252698	-0.091630
	Egypt	0.181788	-0.213651	-0.125138
	Gabon	0.196647	-0.304760	-0.074097
	Georgia	0.118007	-0.094103	-0.386368
	Greece	-0.074758	-0.591805	-0.281025

Cluster	Country name	Healthy life expectancy	Generosity	Perceptions of corruption
	Iraq	0.351529	-0.227587	0.000554
	Jordan	0.045580	-0.116871	-0.237528
	Lebanon	0.149216	-0.431210	-0.026686
	Libya	-0.003411	0.009028	-0.186834
	Mauritania	0.418122	-0.170252	-0.269243
	Morocco	0.349290	-0.195453	-0.230133
	Myanmar	0.141149	0.072764	-0.188137
	Namibia	0.269734	-0.249629	-0.006927
	Palestinian Territories	0.263577	-0.289504	-0.105580
	South Africa	0.183503	-0.169729	0.115725
	Swaziland	0.436130	-0.156309	-0.303635
	Tunisia	0.257567	-0.380550	-0.136286
	Turkey	0.024961	-0.352267	-0.231045
	Venezuela	0.106384	-0.382099	-0.124490
	Yemen	0.477338	-0.238257	-0.146476
4	**Australia**	-0.658334	0.510174	-0.185958
	Austria	-0.637432	0.330454	-0.226919
	Bahrain	-0.364631	0.194292	0.144881
	Canada	-0.654603	0.464547	-0.284476
	Denmark	-0.596733	0.138672	0.080644
	Estonia	-0.504633	0.142225	-0.261237
	Finland	-0.576043	0.001498	-0.013779
	France	-0.550006	-0.045834	-0.303958
	Germany	-0.571162	0.307015	-0.319324
	Hong Kong S.A.R. of China	-0.500498	0.313899	-0.517059
	Iceland	-0.628591	0.285844	0.201096
	Ireland	-0.602889	0.195332	-0.027825
	Luxembourg	-0.649194	0.161291	-0.209367
	Malta	-0.523213	0.283260	0.108404
	Netherlands	-0.559769	0.319942	0.115927
	New Zealand	-0.623280	0.422793	-0.085644
	North Cyprus	-0.344465	0.108174	-0.177732
	Norway	-0.654505	0.282494	0.045623
	Singapore	-0.646359	-0.037474	0.117920
	Sweden	-0.604330	0.312533	-0.024717

		Healthy life expectancy	Generosity	Perceptions of corruption
Cluster	**Country name**			
	Switzerland	-0.616022	0.103608	0.047030
	United Arab Emirates	-0.410100	0.322066	-0.061693
	United Kingdom	-0.565617	0.546718	-0.146330
	United States	-0.391566	0.134890	0.076529
	Uruguay	-0.400705	0.105007	-0.173715
	Uzbekistan	-0.262054	0.761740	0.094459

```
In [ ]:
```