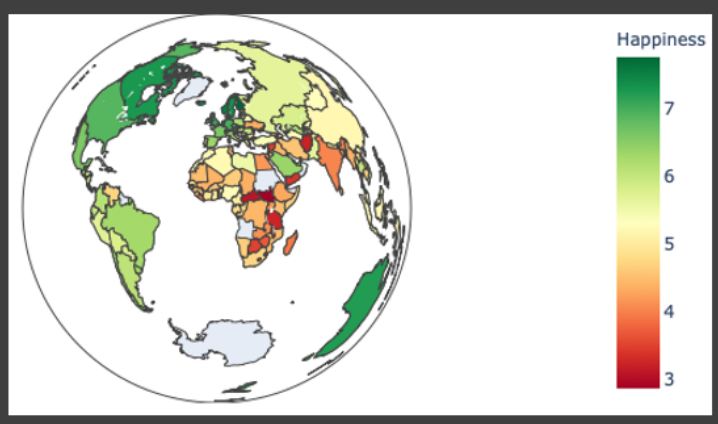
TASK #1 UNDERSTAND THE PROBLEM STATEMENT AND BUSINESS CASE

CLUSTERING: WORLD HAPPINESS REPORT

- In this case study, we will train an unsupervised machine learning algorithm to cluster countries based on features such as economic production, social support, life expectancy, freedom, absence of corruption, and generosity.
- The World Happiness Report determines the state of global happiness.
- The happiness scores and rankings data has been collected by asking individuals to rank their life from 0 (worst possible life) to 10 (best possible life).



Data Source: https://www.kaggle.com/unsdsn/world-happiness

INSTRUCTOR

- Adjunct professor & online instructor
- Passionate about artificial intelligence, machine learning, and electric vehicles
- Taught 200,000+ students globally
- MBA (2018), Ph.D. (2014), M.A.Sc (2011)

Ryan Ahmed, Ph.D.

TASK #2: IMPORT DATASETS AND LIBRARIES

In [1]:
```
# Get the required dependencies
! pip install bubbly
! pip install iplot
! pip install chart_studio
```

```
Processing c:\users\administrator\appdata\local\pip\cache\wheels\85\c7\3d\38784ece9ac882d3afdb852ea8cf867df9
42fe6c71da5c4360\bubbly-1.0.2-py3-none-any.whl
Requirement already satisfied: pandas in c:\users\administrator\anaconda3\lib\site-packages (from bubbly)
(1.0.1)
Collecting plotly
  Using cached plotly-4.14.1-py2.py3-none-any.whl (13.2 MB)
Requirement already satisfied: numpy>=1.13.3 in c:\users\administrator\anaconda3\lib\site-packages (from pan
das->bubbly) (1.18.1)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\users\administrator\anaconda3\lib\site-packages
(from pandas->bubbly) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\administrator\anaconda3\lib\site-packages (from pand
as->bubbly) (2019.3)
Processing c:\users\administrator\appdata\local\pip\cache\wheels\f9\8d\8d\f6af3f7f9eea3553bc2fe6d53e4b287dad
18b06a861ac56ddf\retrying-1.3.3-py3-none-any.whl
Requirement already satisfied: six in c:\users\administrator\anaconda3\lib\site-packages (from plotly->bubbl
y) (1.14.0)
Installing collected packages: retrying, plotly, bubbly
Successfully installed bubbly-1.0.2 plotly-4.14.1 retrying-1.3.3
Collecting iplot
  Using cached iplot-0.0.0-py3-none-any.whl (1.3 kB)
Installing collected packages: iplot
Successfully installed iplot-0.0.0
Collecting chart_studio
  Using cached chart_studio-1.1.0-py3-none-any.whl (64 kB)
Requirement already satisfied: plotly in c:\users\administrator\anaconda3\lib\site-packages (from chart_stud
io) (4.14.1)
Requirement already satisfied: six in c:\users\administrator\anaconda3\lib\site-packages (from chart_studio)
(1.14.0)
Requirement already satisfied: requests in c:\users\administrator\anaconda3\lib\site-packages (from chart_st
udio) (2.22.0)
Requirement already satisfied: retrying>=1.3.3 in c:\users\administrator\anaconda3\lib\site-packages (from c
hart_studio) (1.3.3)
Requirement already satisfied: idna<2.9,>=2.5 in c:\users\administrator\anaconda3\lib\site-packages (from re
quests->chart_studio) (2.8)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\administrator\anaconda3\lib\site-packages (fro
m requests->chart_studio) (2019.11.28)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\administrator\anaconda3\lib\site-packages
(from requests->chart_studio) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in c:\users\administrator\anaconda3\l
ib\site-packages (from requests->chart_studio) (1.25.8)
Installing collected packages: chart-studio
Successfully installed chart-studio-1.1.0
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler, normalize
from sklearn.cluster import KMeans
import plotly.express as px
import plotly.graph_objects as go
from chart_studio.plotly import plot, iplot
from plotly.offline import iplot
```

In [3]:
```
from jupyterthemes import jtplot
jtplot.style(theme='monokai', context='notebook', ticks=True, grid=False)
# setting the style of the notebook to be monokai theme
# this line of code is important to ensure that we are able to see the x and y axes clearly
# If you don't run this code line, you will notice that the xlabel and ylabel on any plot is black on black ar
```

In [4]:
```
# Import csv file into pandas dataframe
happy_df = pd.read_csv('happiness_report.csv')
```

In [5]:
```
# print the first 5 rows of the dataframe
happy_df.head()
```

Out[5]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298

MINI CHALLENGE #1:

- Find out how many samples exist in the DataFrame using two different methods.

- Select your own country from the dataframe and explore scores. Perform sanity check.

In [9]:
```
len(happy_df)
```

Out[9]: 156

In [6]:
```
happy_df
```

Out[6]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298
...	...	...	...	...	...	...	...	...	...
151	152	Rwanda	3.334	0.359	0.711	0.614	0.555	0.217	0.411
152	153	Tanzania	3.231	0.476	0.885	0.499	0.417	0.276	0.147
153	154	Afghanistan	3.203	0.350	0.517	0.361	0.000	0.158	0.025
154	155	Central African Republic	3.083	0.026	0.000	0.105	0.225	0.235	0.035
155	156	South Sudan	2.853	0.306	0.575	0.295	0.010	0.202	0.091

156 rows × 9 columns

In [7]:
```
happy_df[happy_df['Country or region']=='Canada']
```

Out[7]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
8	9	Canada	7.278	1.365	1.505	1.039	0.584	0.285	0.308

TASK #3: PERFORM EXPLORATORY DATA ANALYSIS

In [10]:
```
# Check the number of non-null values in the dataframe
happy_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156 entries, 0 to 155
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Overall rank                  156 non-null    int64  
 1   Country or region             156 non-null    object 
 2   Score                         156 non-null    float64
 3   GDP per capita                156 non-null    float64
 4   Social support                156 non-null    float64
 5   Healthy life expectancy       156 non-null    float64
 6   Freedom to make life choices  156 non-null    float64
 7   Generosity                    156 non-null    float64
 8   Perceptions of corruption     156 non-null    float64
dtypes: float64(7), int64(1), object(1)
memory usage: 11.1+ KB
```

In [11]:
```
# Check Null values
happy_df.isnull().sum()
```

Out[11]:
```
Overall rank                  0
Country or region             0
Score                         0
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]:
```
# Obtain the Statistical summary of the dataframe
happy_df.describe()
```

Out[12]:

	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
count	156.000000	156.000000	156.000000	156.000000	156.000000	156.000000	156.000000	156.000000
mean	78.500000	5.407096	0.905147	1.208814	0.725244	0.392571	0.184846	0.110603
std	45.177428	1.113120	0.398389	0.299191	0.242124	0.143289	0.095254	0.094538
min	1.000000	2.853000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	39.750000	4.544500	0.602750	1.055750	0.547750	0.308000	0.108750	0.047000
50%	78.500000	5.379500	0.960000	1.271500	0.789000	0.417000	0.177500	0.085500
75%	117.250000	6.184500	1.232500	1.452500	0.881750	0.507250	0.248250	0.141250
max	156.000000	7.769000	1.684000	1.624000	1.141000	0.631000	0.566000	0.453000

In [13]:
```
# check the number of duplicated entries in the dataframe
happy_df.duplicated().sum() # since there are no duplicates, no further action is required
```

Out[13]:
```
0
```

MINI CHALLENGE #2:

- What is the country that has the maximum happiness score? What is the perception of corruption in this country?

In [14]:
```
happy_df[happy_df['Score'] == 7.769000]
```

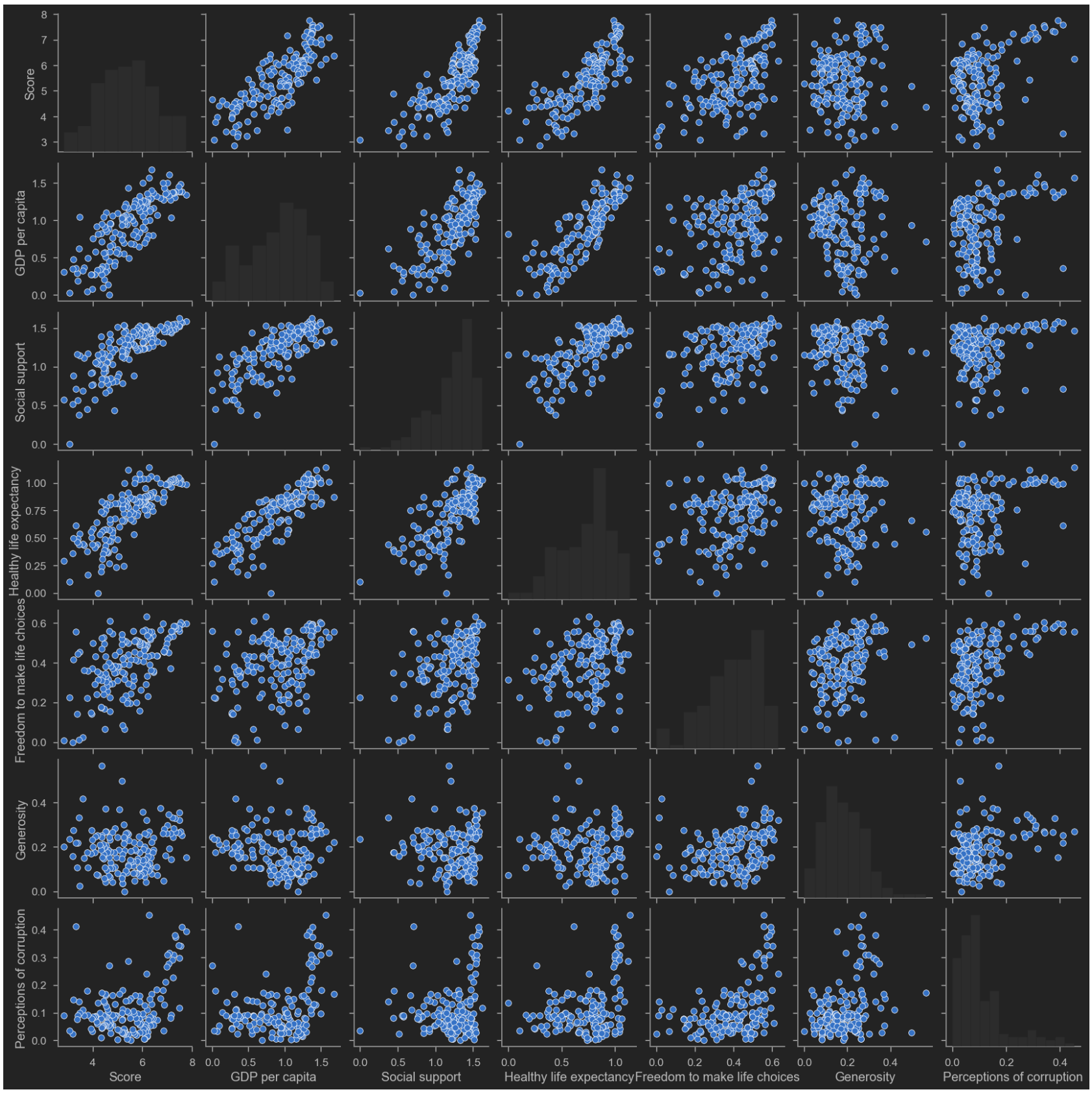
Out[14]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.34	1.587	0.986	0.596	0.153	0.393

TASK #4: PERFORM DATA VISUALIZATION - PART #1

In [15]:
```
# Plot the pairplot
fig = plt.figure(figsize = (20,20))
sns.pairplot(happy_df[['Score','GDP per capita', 'Social support', 'Healthy life expectancy',
    'Freedom to make life choices', 'Generosity', 'Perceptions of corruption']])

# Positive correlation between GDP and score
# Positive correlation between Social Support and score
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x28f7b5bd520>

<Figure size 2000x2000 with 0 Axes>
```

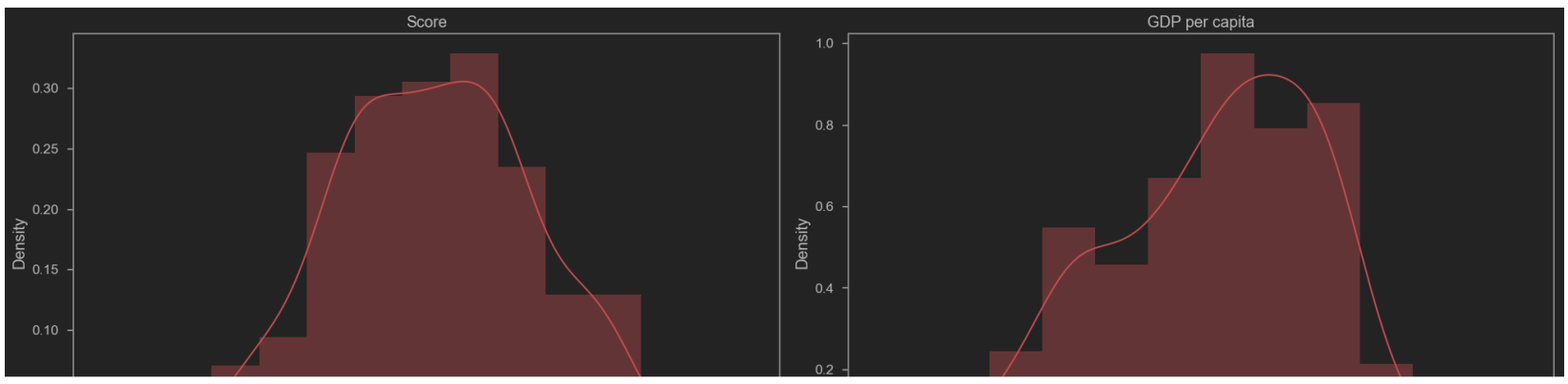


In [16]:
```
# distplot combines the matplotlib.hist function with seaborn kdeplot()
columns = ['Score','GDP per capita', 'Social support', 'Healthy life expectancy',
    'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption']
plt.figure(figsize = (20, 50))
for i in range(len(columns)):
  plt.subplot(8, 2, i+1)
  sns.distplot(happy_df[columns[i]], color = 'r');
  plt.title(columns[i])

plt.tight_layout()
```

for histograms).

C:\Users\SOMNATH\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

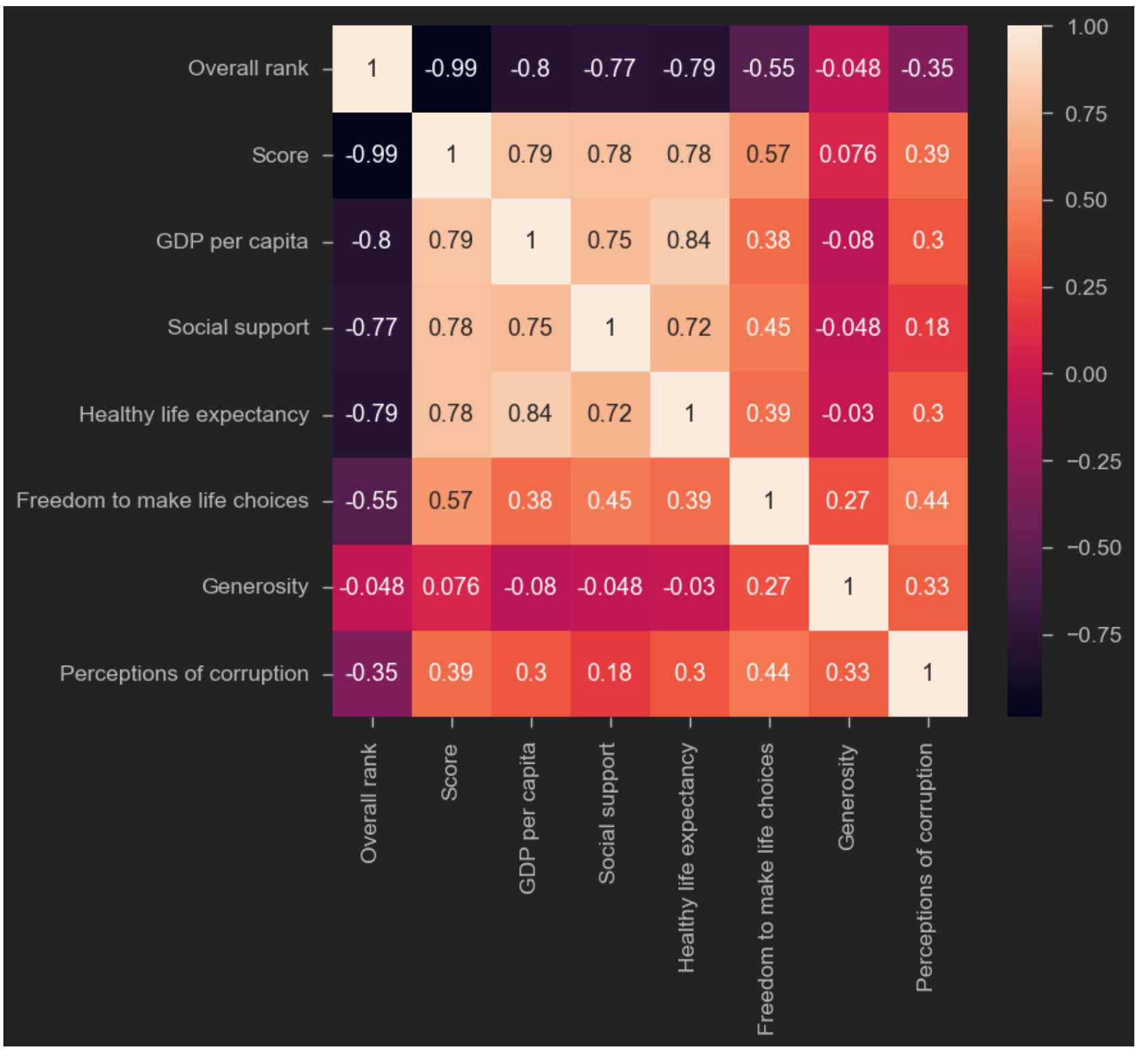


MINI CHALLENGE #3:

- Plot the correlation matrix and comment on the results.

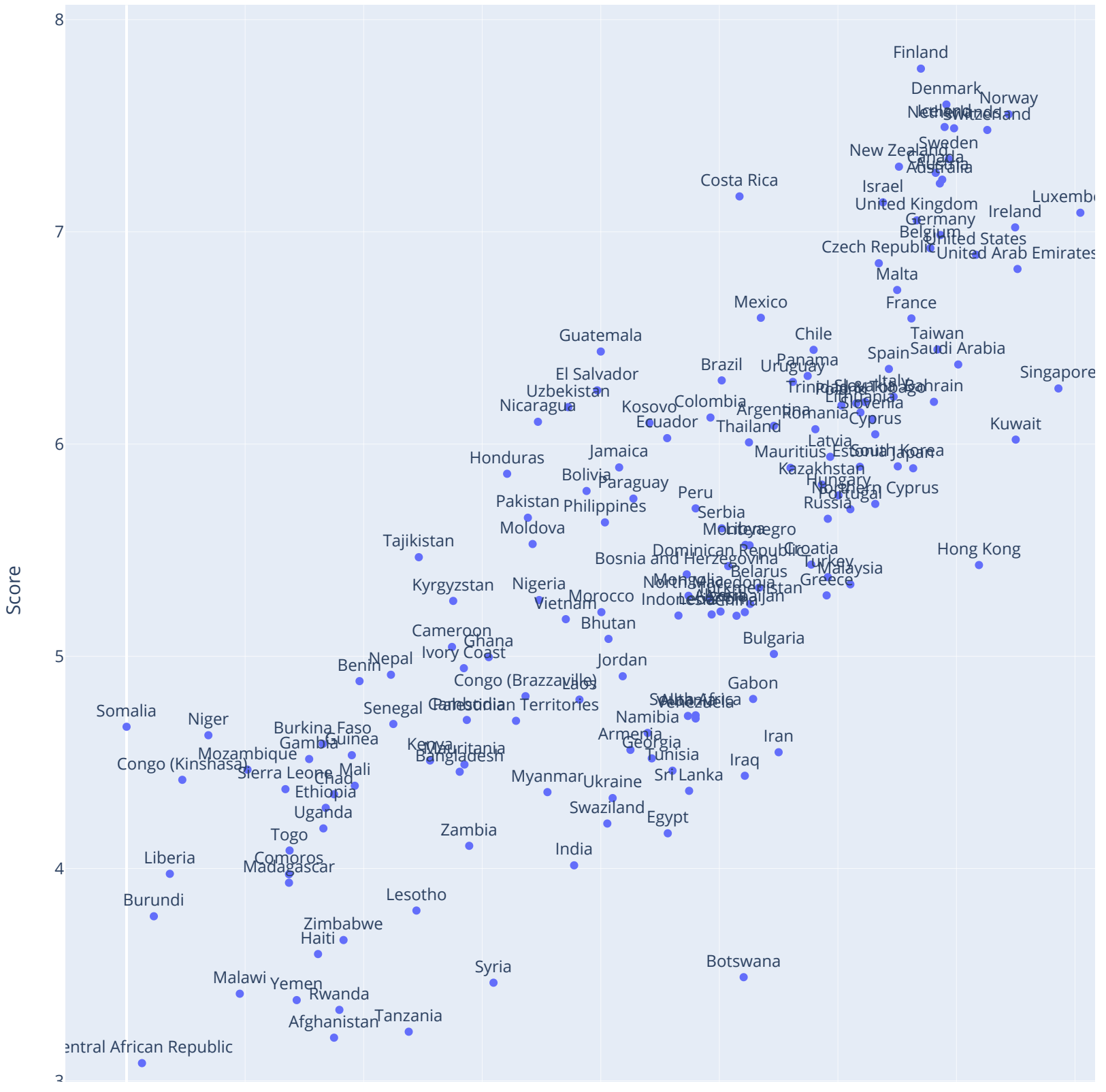
In [17]:
```
# Get the correlation matrix
corr_matrix = happy_df.corr()
corr_matrix
sns.heatmap(corr_matrix, annot = True)
```

Out[17]: <AxesSubplot:>

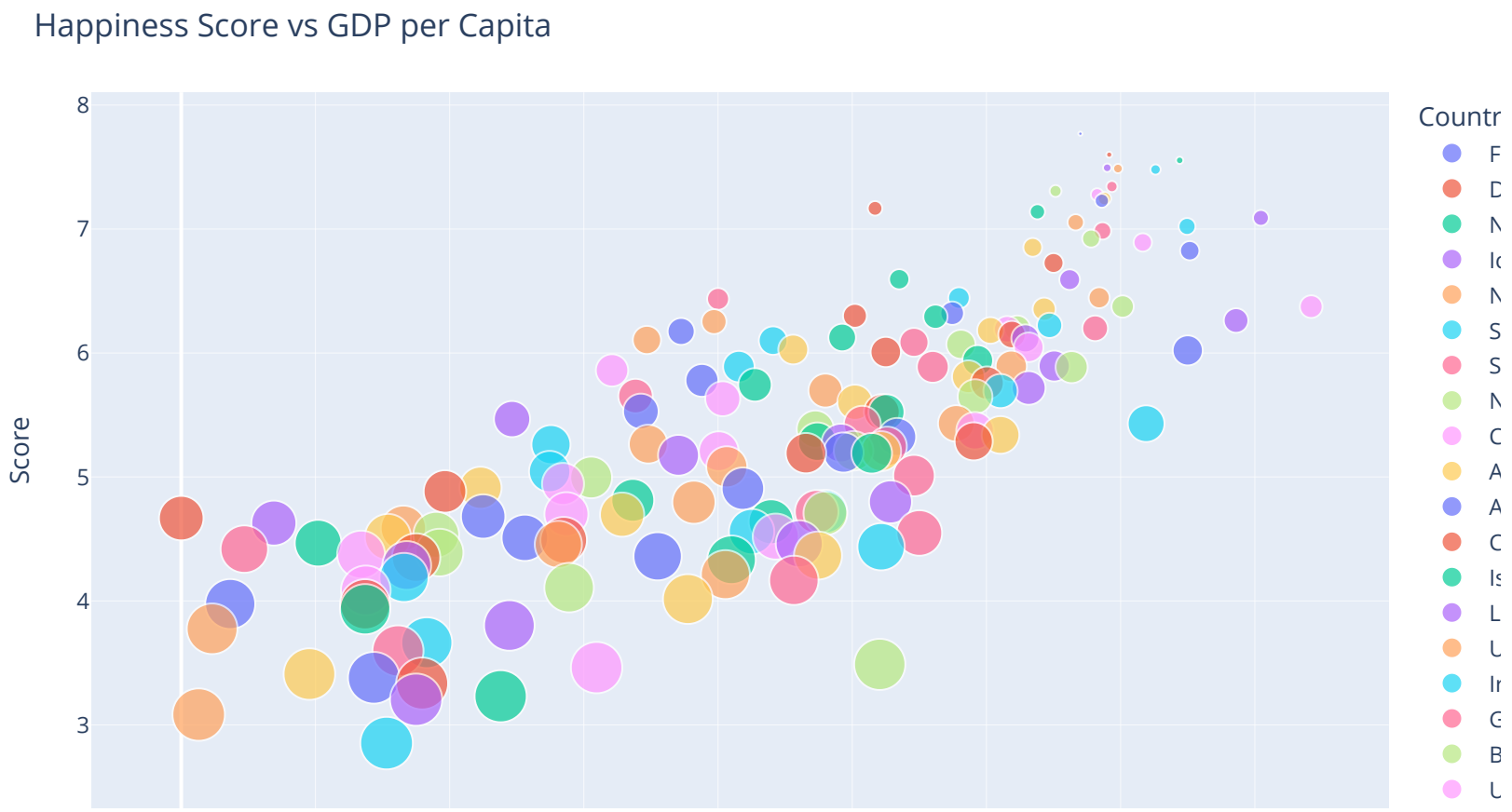


TASK #5: PERFORM DATA VISUALIZATION - PART #2

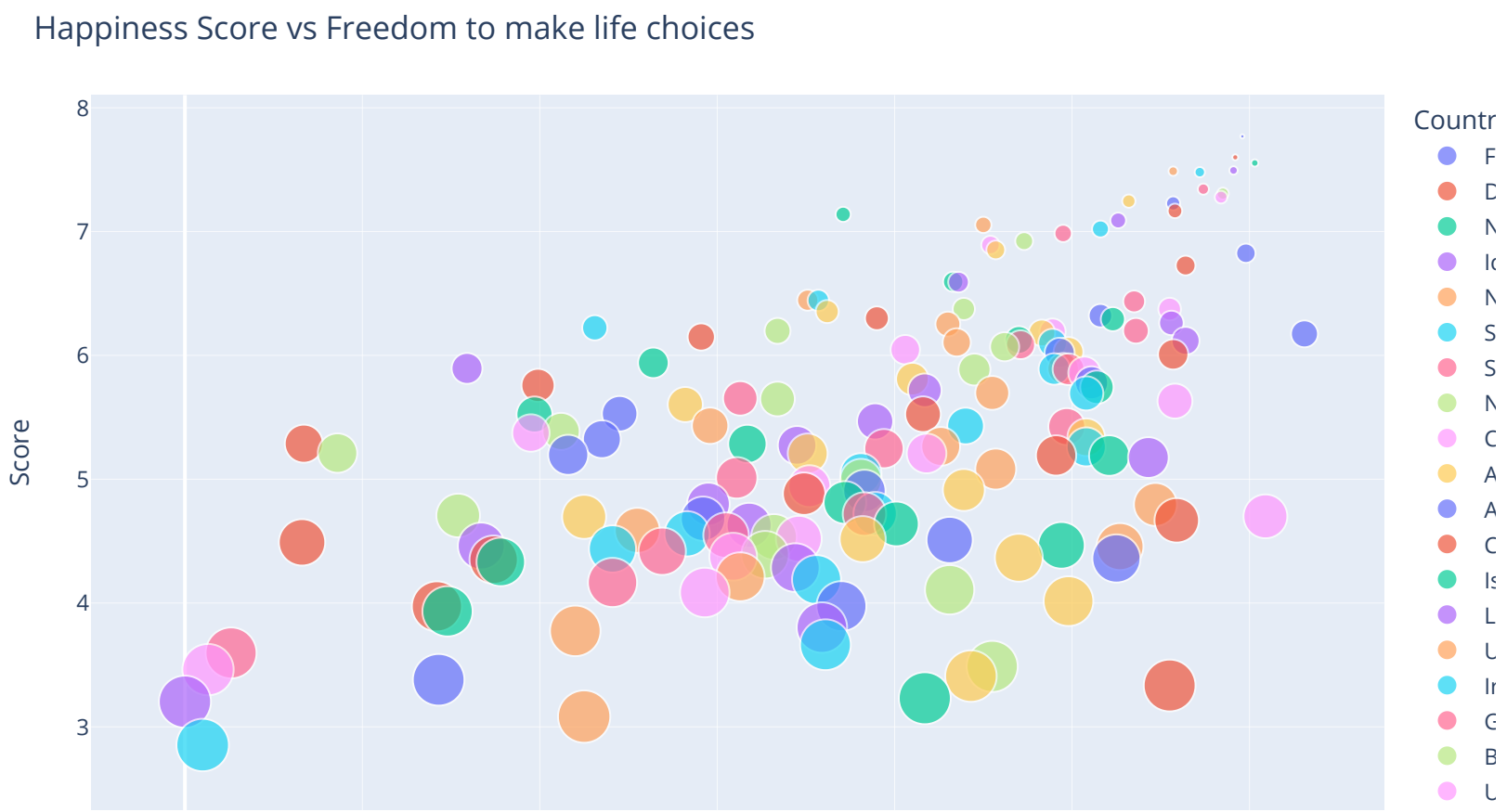
In [18]:
```
# Plot the relationship between score, GDP and region
fig = px.scatter(happy_df, x = 'GDP per capita', y = 'Score', text = 'Country or region')
fig.update_traces(textposition = 'top center')
fig.update_layout(height = 1000)
fig.show()
```



In [19]:
```
# Plot the relationship between score and GDP (while adding color and size)

fig = px.scatter(happy_df, x = "GDP per capita", y = "Score", size = 'Overall rank', color = "Country or regio

fig.update_layout(title_text = 'Happiness Score vs GDP per Capita')
fig.show()
```



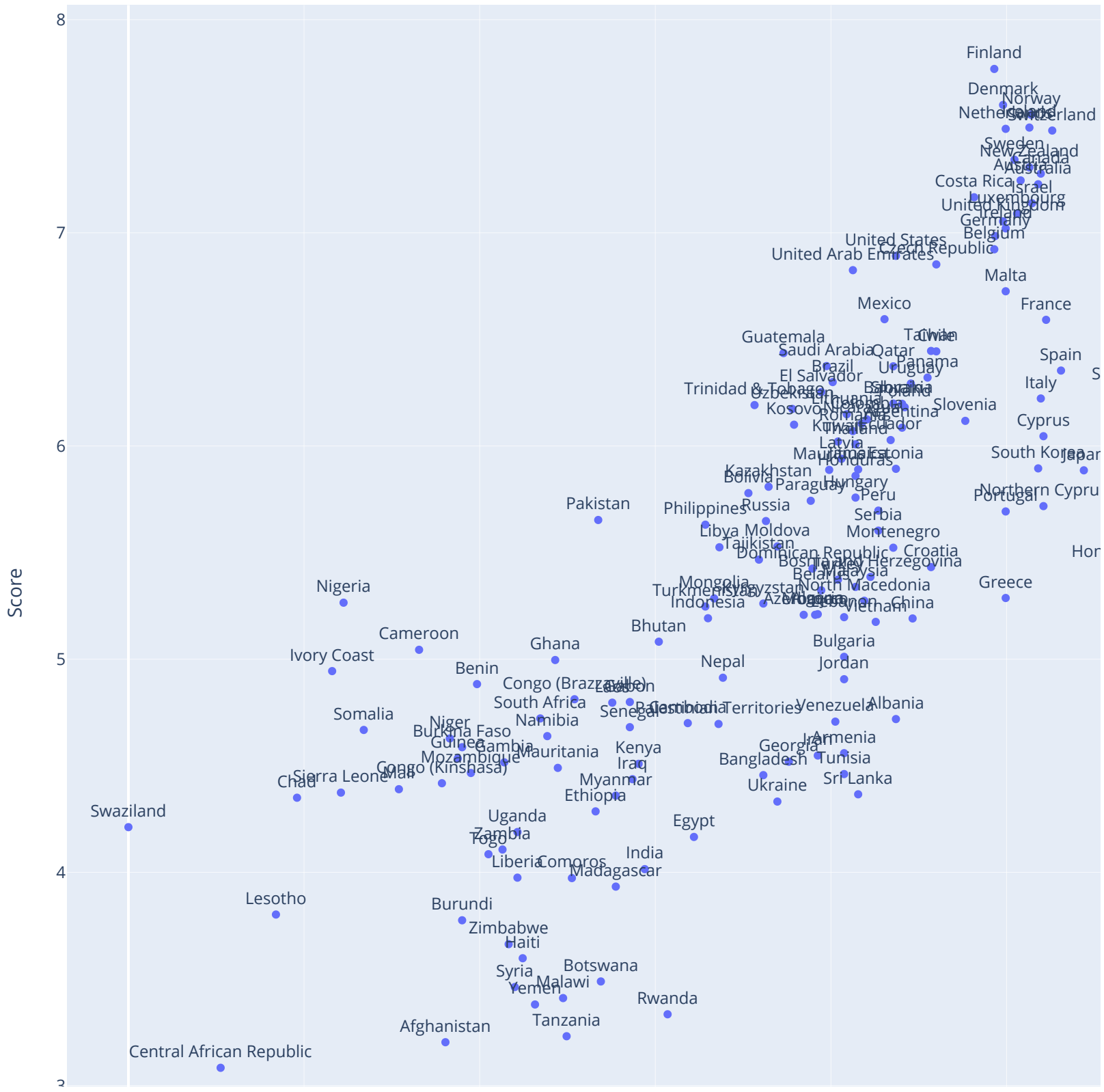
In [20]:
```
# Plot the relationship between score and freedom to make life choices

fig = px.scatter(happy_df, x = 'Freedom to make life choices', y = "Score", size = 'Overall rank', color = "Co
          trendline = "ols")

fig.update_layout(title_text = 'Happiness Score vs Freedom to make life choices')
fig.show()
```



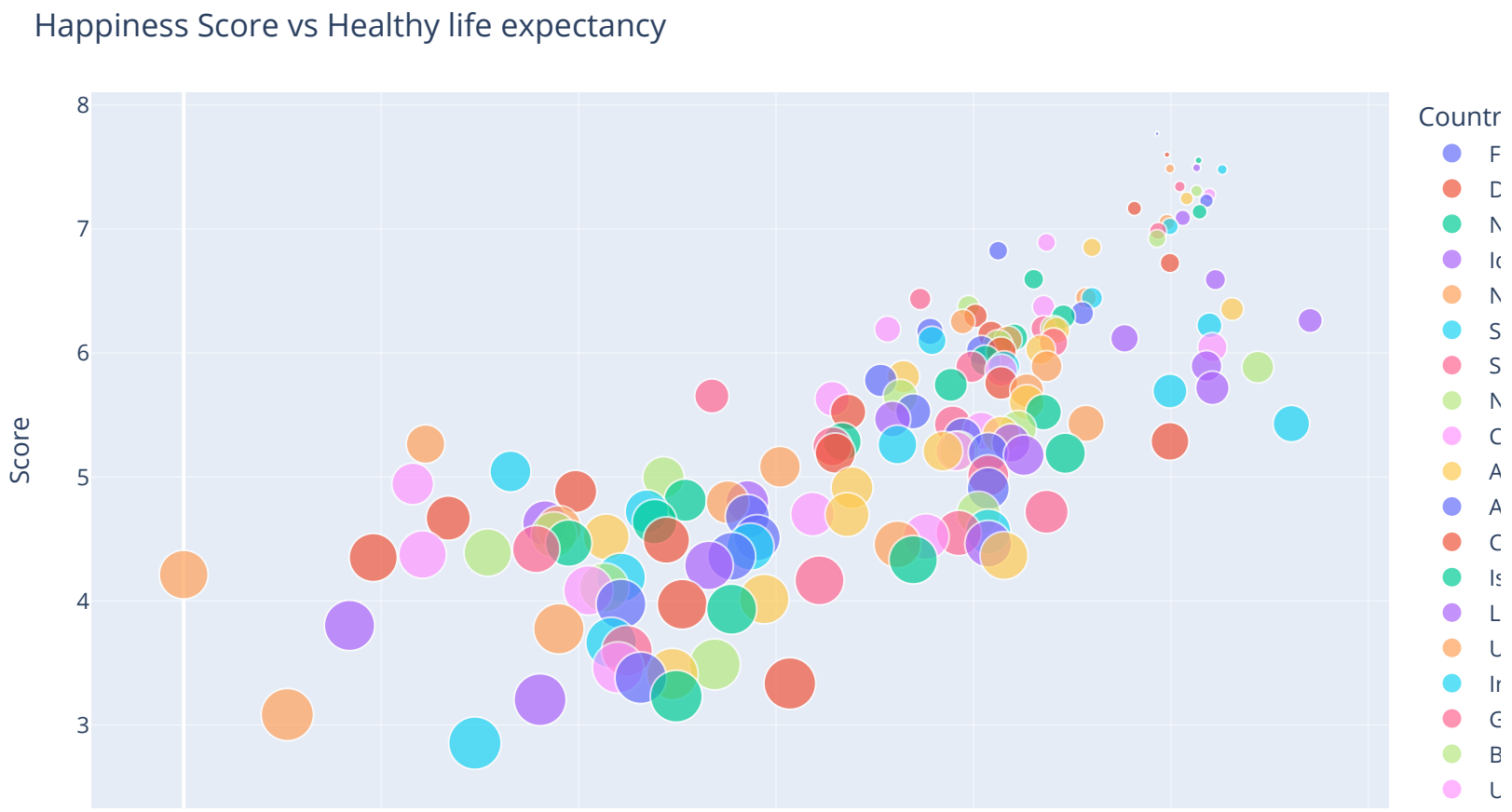
MINI CHALLENGE #4:

- Plot a similar type of plots for 'Healthy life expectancy' and 'Score'

In [21]:
```
# Plot the relationship between score and healthy life expectancy
fig = px.scatter(happy_df, x = 'Healthy life expectancy', y = "Score", text = 'Country or region')
fig.update_traces(textposition = 'top center')
fig.update_layout(height = 1000)
fig.show()
```



In [22]:
```
# Plot the relationship between score and healthy life expectancy
fig = px.scatter(happy_df, x = 'Healthy life expectancy', y = "Score",
           size = 'Overall rank', color = "Country or region", hover_name = "Country or region",
          trendline = "ols")

fig.update_layout(
    title_text = 'Happiness Score vs Healthy life expectancy'
)
fig.show()
```



TASK #6: PREPARE THE DATA TO FEED THE CLUSTERING MODEL

In [23]:
```
# We are going to create clusters without the use of happiness score and rank to see which countries fall unde
```

In [24]:
```
# Select the data without rank and happiness score
df_seg = happy_df.drop(columns = ['Overall rank', 'Country or region', 'Score'])
df_seg
```

Out[24]:

	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1.340	1.587	0.986	0.596	0.153	0.393
1	1.383	1.573	0.996	0.592	0.252	0.410
2	1.488	1.582	1.028	0.603	0.271	0.341
3	1.380	1.624	1.026	0.591	0.354	0.118
4	1.396	1.522	0.999	0.557	0.322	0.298
...	...	...	...	...	...	...
151	0.359	0.711	0.614	0.555	0.217	0.411
152	0.476	0.885	0.499	0.417	0.276	0.147
153	0.350	0.517	0.361	0.000	0.158	0.025
154	0.026	0.000	0.105	0.225	0.235	0.035
155	0.306	0.575	0.295	0.010	0.202	0.091

156 rows × 6 columns

In [25]:
```
# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_seg)
```

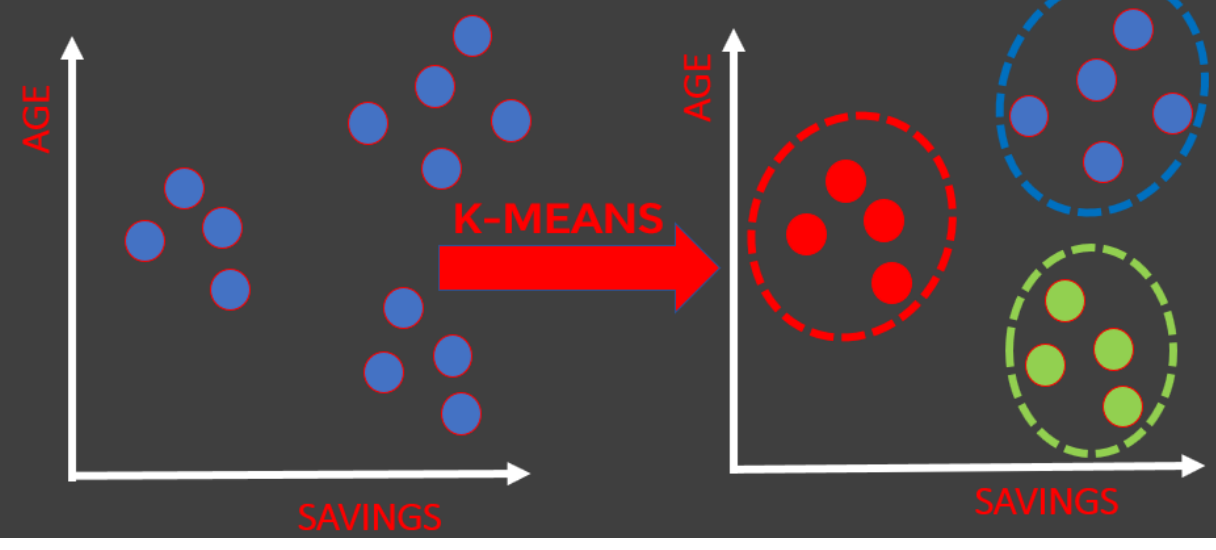
In [26]:
```
scaled_data.shape
```

Out[26]:
```
(156, 6)
```

TASK #7: UNDERSTAND THE THEORY AND INTUITION BEHIND K-MEANS CLUSTERING ALGORITHM

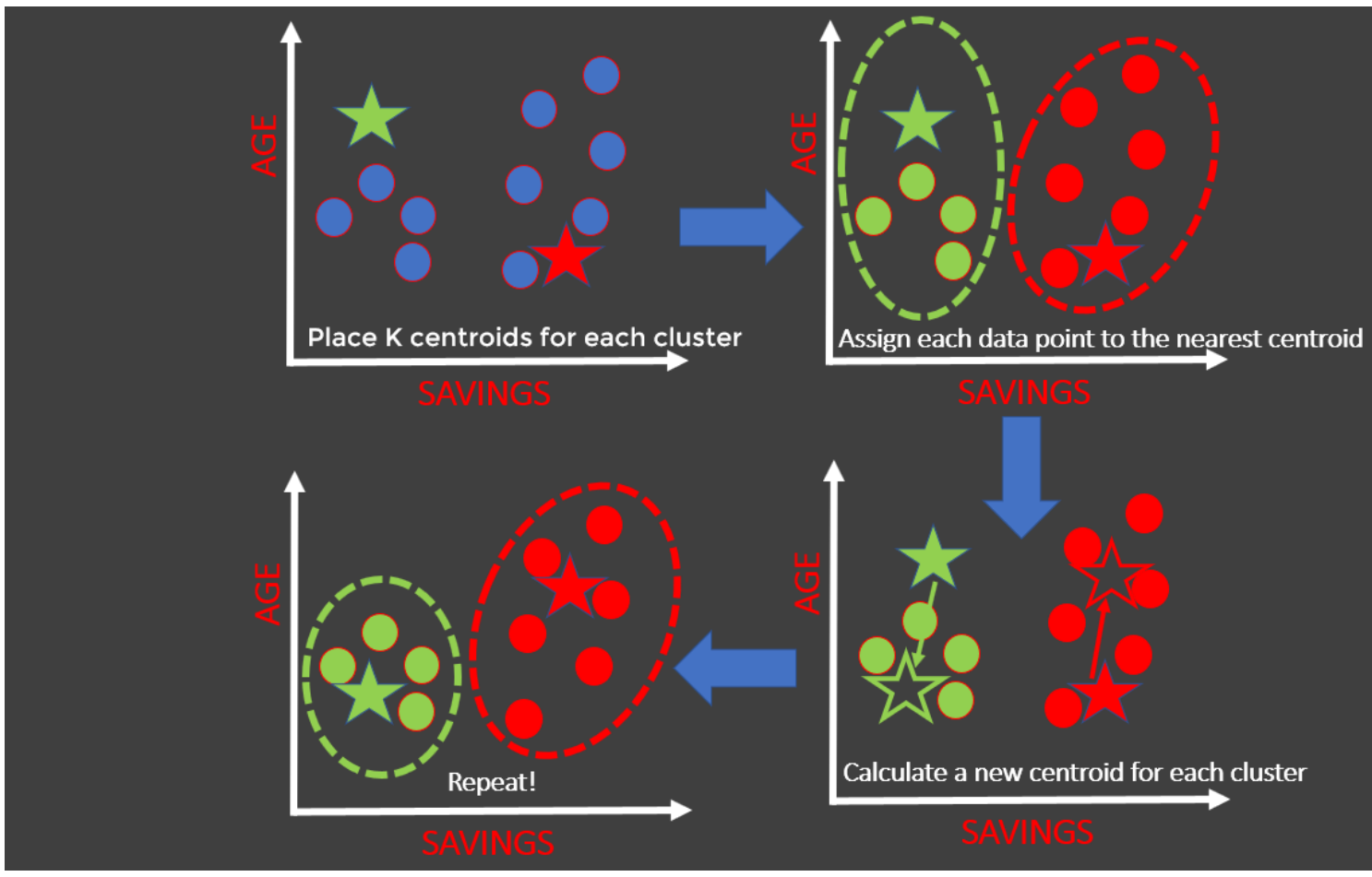
K-MEANS INTUITON

- K-means is an unsupervised learning algorithm (clustering).
- K-means works by grouping some data points together (clustering) in an unsupervised fashion.
- The algorithm groups observations with similar attribute values together by measuring the Euclidian distance between points.



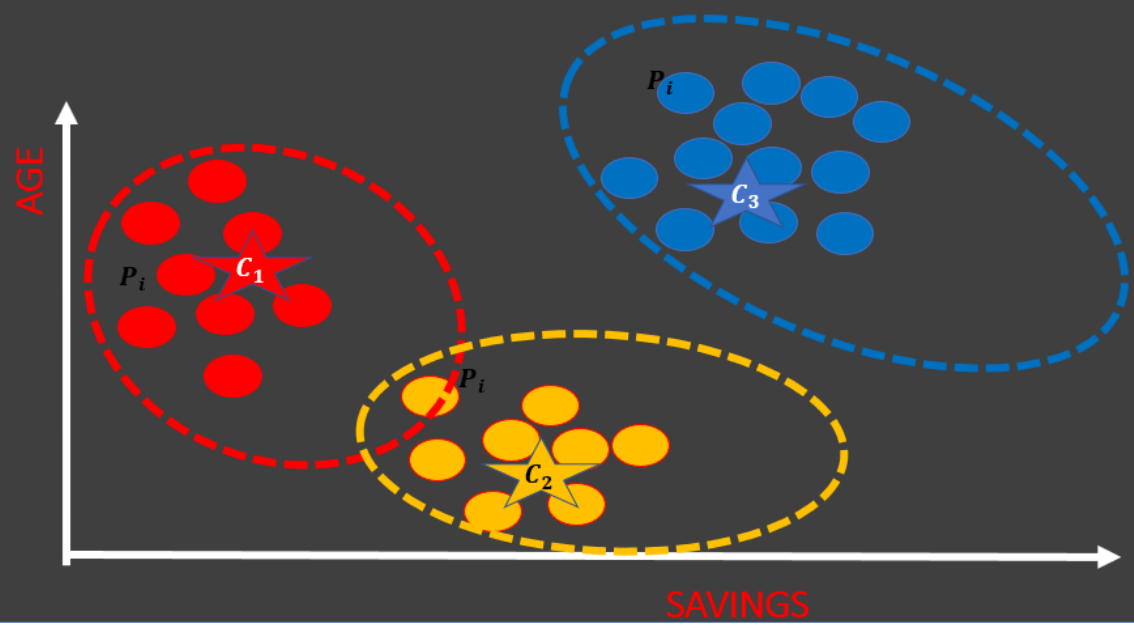
K-MEANS ALGORITHM STEPS

1. Choose number of clusters “K”
2. Select random K points that are going to be the centroids for each cluster
3. Assign each data point to the nearest centroid, doing so will enable us to create “K” number of clusters
4. Calculate a new centroid for each cluster
5. Reassign each data point to the new closest centroid
6. Go to step 4 and repeat.



HOW TO SELECT THE OPTIMAL NUMBER OF CLUSTERS (K)? “ELBOW METHOD”

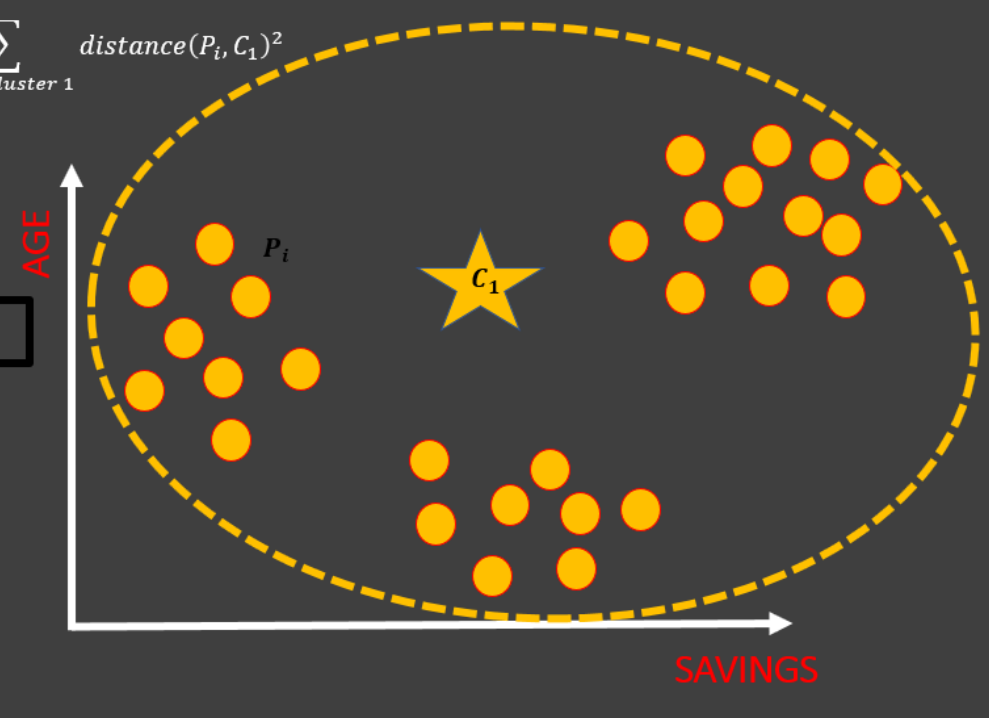
$$Within\ Cluster\ Sum\ of\ Squares\ (WCSS) = \sum_{P_i\ in\ Cluster\ 1} distance(P_i, C_1)^2 + \sum_{P_i\ in\ Cluster\ 2} distance(P_i, C_2)^2 + \sum_{P_i\ in\ Cluster\ 3} distance(P_i, C_3)^2$$



HOW TO SELECT THE OPTIMAL NUMBER OF CLUSTERS (K)? “ELBOW METHOD”

$$Within\ Cluster\ Sum\ of\ Squares\ (WCSS) = \sum_{P_i\ in\ Cluster\ 1} distance(P_i, C_1)^2$$

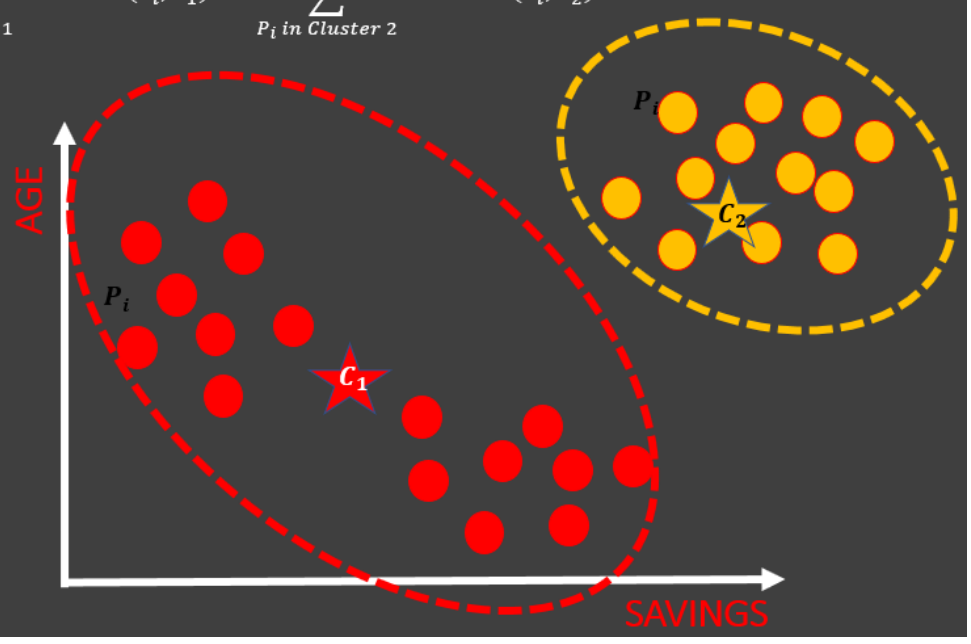
NUMBER OF CLUSTERS (K) = 1

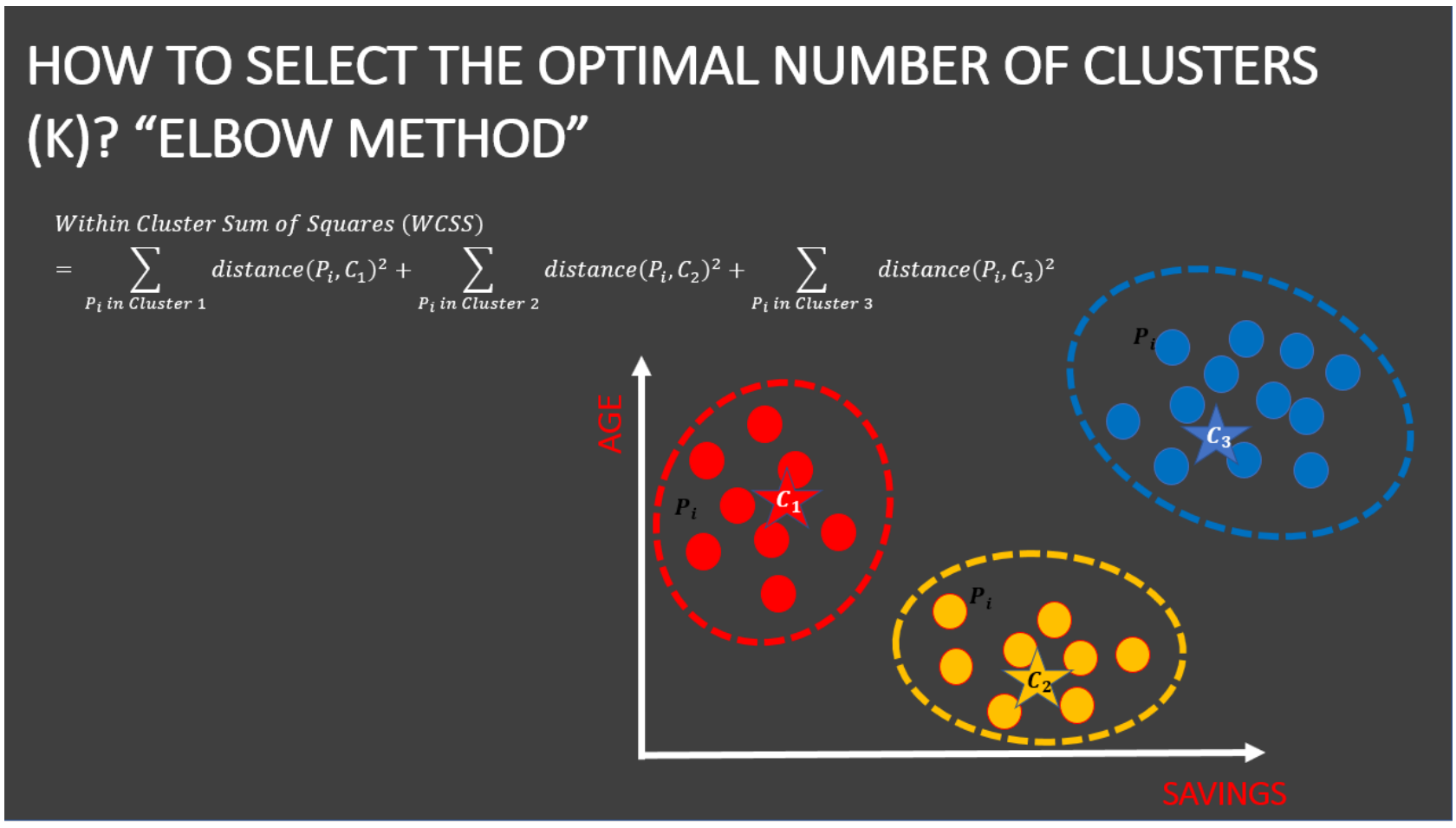


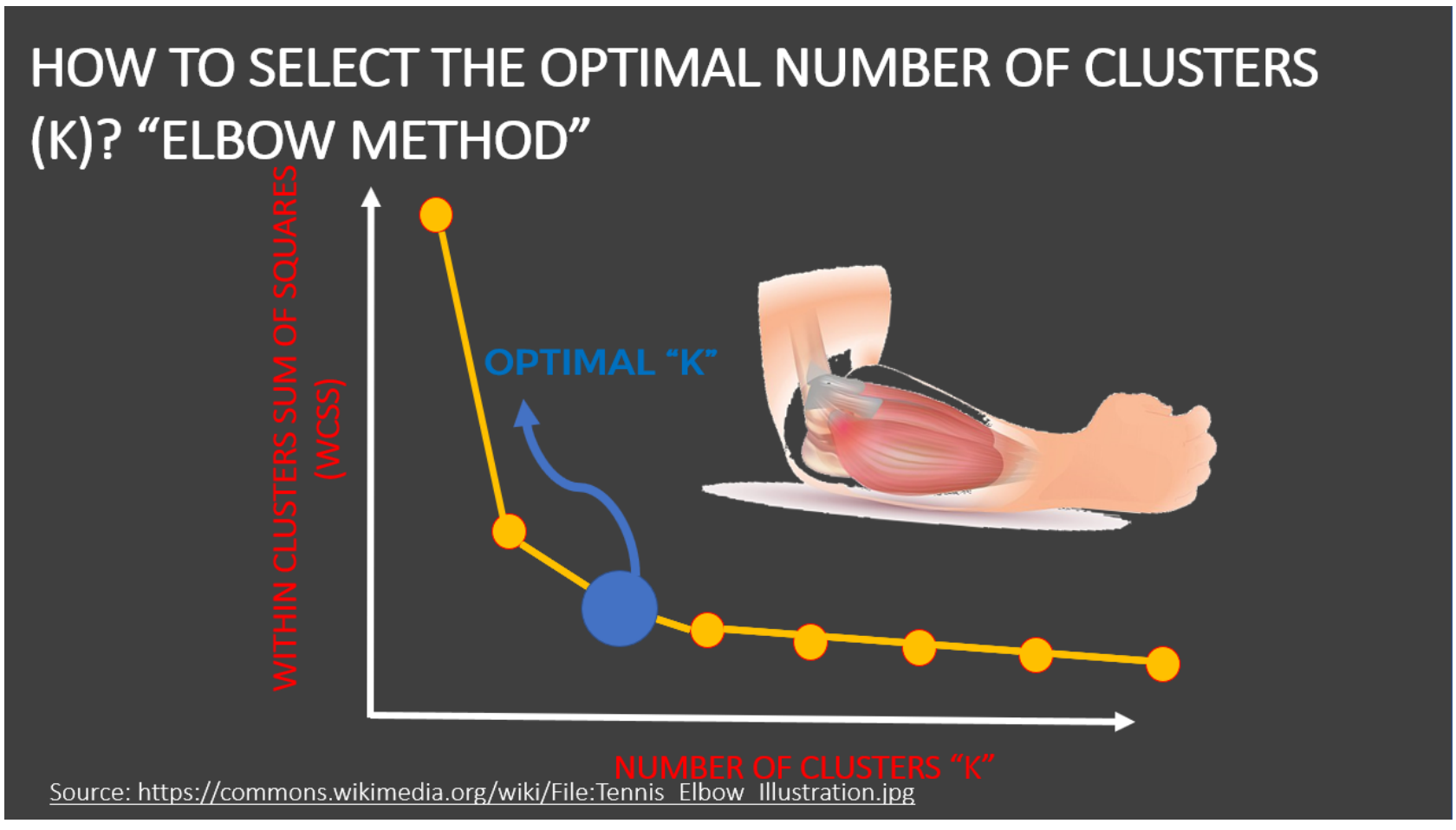
HOW TO SELECT THE OPTIMAL NUMBER OF CLUSTERS (K)? “ELBOW METHOD”

$$Within\ Cluster\ Sum\ of\ Squares\ (WCSS) = \sum_{P_i\ in\ Cluster\ 1} distance(P_i, C_1)^2 + \sum_{P_i\ in\ Cluster\ 2} distance(P_i, C_2)^2$$

NUMBER OF CLUSTERS (K) = 2



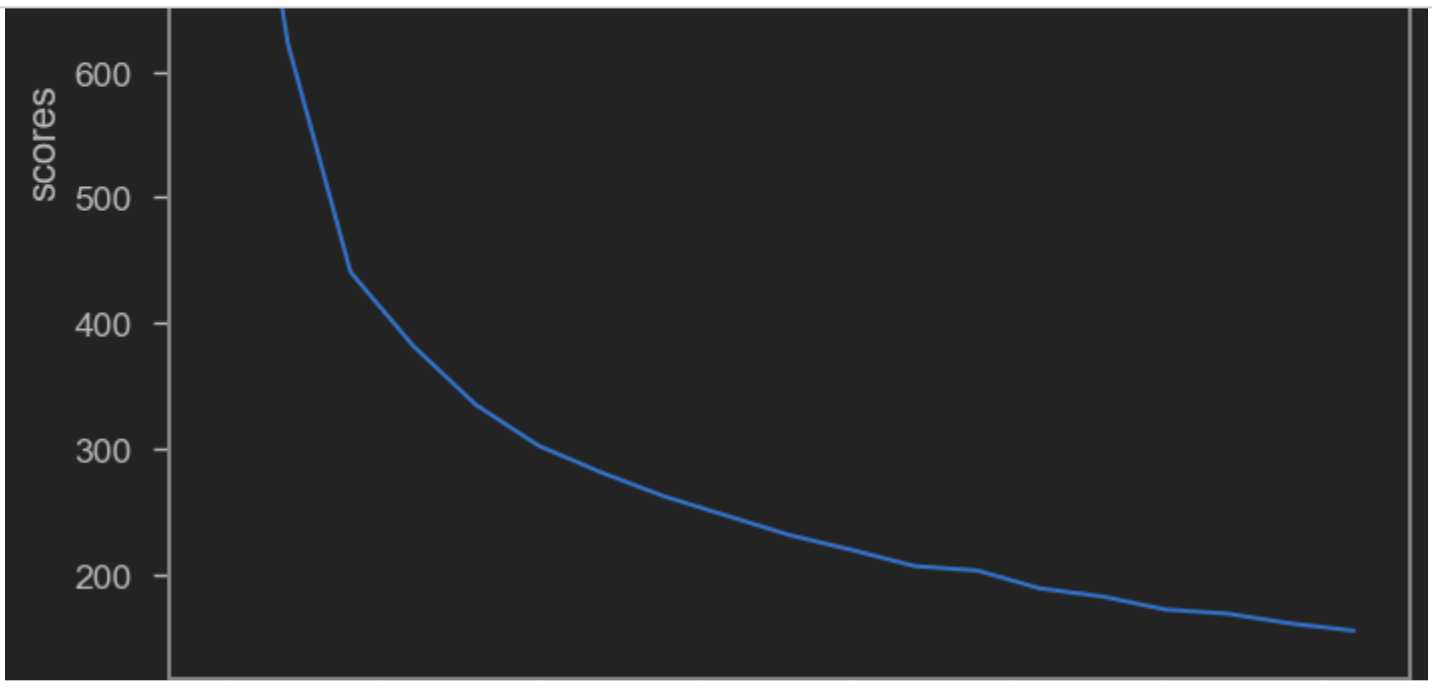




TASK #8: FIND THE OPTIMAL NUMBER OF CLUSTERS USING ELBOW METHOD

- The elbow method is a heuristic method of interpretation and validation of consistency within cluster analysis designed to help find the appropriate number of clusters in a dataset.
- If the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best.
- Source:
  - https://en.wikipedia.org/wiki/Elbow_method_(clustering) (https://en.wikipedia.org/wiki/Elbow_method_(clustering))
  - https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/ (https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/)

In [27]:
```
scores = []

range_values = range(1,20)

for i in range_values:
  kmeans = KMeans(n_clusters= i)
  kmeans.fit(scaled_data)
  scores.append(kmeans.inertia_)

plt.plot(scores, 'bx-')
plt.title('Finding right number of clusters')
plt.xlabel('Clusters')
plt.ylabel('scores')
plt.show()
```



In [28]:
```
# From this we can observe that 3rd cluster seems to be forming the elbow of the curve.
# Let's choose the number of clusters to be 3.
```

TASK #9: APPLY K-MEANS METHOD

In [29]:
```
kmeans = KMeans(3)
kmeans.fit(scaled_data)
labels = kmeans.labels_
```

In [30]:
```
kmeans.cluster_centers_.shape
```

Out[30]:
```
(3, 6)
```

In [31]:
```
cluster_centers = pd.DataFrame(data = kmeans.cluster_centers_, columns = [df_seg.columns])
cluster_centers
```

Out[31]:

	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1.044048	0.840457	0.886718	1.043959	1.201517	1.468652
1	-1.264229	-1.132144	-1.240341	-0.471563	0.260570	-0.114358
2	0.352697	0.348120	0.393020	-0.091938	-0.556448	-0.437339

In [32]:
```
# In order to understand what these numbers mean, let's perform inverse transformation
cluster_centers = scaler.inverse_transform(cluster_centers)
cluster_centers = pd.DataFrame(data = cluster_centers, columns = [df_seg.columns])
cluster_centers
```

Out[32]:

	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1.319750	1.459464	0.939250	0.541679	0.298929	0.249000
1	0.403109	0.871174	0.425891	0.325217	0.209587	0.099826
2	1.045207	1.312634	0.820098	0.379439	0.132012	0.069390

- Cluster 0: countries that have GDP in the range of 0.6 to 1.4 and have high social support. These countries have medium life expectancy and have high freedom to make life choices. These counties have low generosity and low perception of corruption.
- Cluster 1: countries that have very high GDP, high social support and high life expectancy. These counties have high freedom to make life choices, medium generosity and medium perception of corruption.
- Cluster 2: countries that have low GDP average life expectancy and average social support. These counties have low freedom to make life choices, high generosity and medium perception of corruption.

In [33]:
```
labels.shape # Labels associated to each data point
```

Out[33]:
```
(156,)
```

In [34]:
```
labels.max()
```

Out[34]: 2

In [35]:
```
labels.min()
```

Out[35]: 0

In [36]:
```
y_kmeans = kmeans.fit_predict(scaled_data)
y_kmeans
```

Out[36]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1,
       2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       0, 2, 2, 2, 2, 2, 2, 0, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2,
       2, 2, 2, 1, 2, 2, 1, 0, 2, 0, 0, 0, 2, 0, 0, 2, 0, 2, 2, 2, 0, 2,
       0, 0, 2, 0, 0, 2, 2, 0, 2, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 2, 1, 0,
       2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
       0, 0])
```

In [37]:
```
# concatenate the clusters labels to our original dataframe
happy_df_cluster = pd.concat([happy_df, pd.DataFrame({'cluster':labels})], axis = 1)
happy_df_cluster
```

Out[37]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption	cluster
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393	0
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410	0
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341	0
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118	0
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298	0
...	...	...	...	...	...	...	...	...	...	...
151	152	Rwanda	3.334	0.359	0.711	0.614	0.555	0.217	0.411	1
152	153	Tanzania	3.231	0.476	0.885	0.499	0.417	0.276	0.147	1
153	154	Afghanistan	3.203	0.350	0.517	0.361	0.000	0.158	0.025	1
154	155	Central African Republic	3.083	0.026	0.000	0.105	0.225	0.235	0.035	1
155	156	South Sudan	2.853	0.306	0.575	0.295	0.010	0.202	0.091	1

156 rows × 10 columns

In [38]:
```
# Plot the histogram of various clusters
for i in df_seg.columns:
  plt.figure(figsize = (35, 10))
  for j in range(3):
    plt.subplot(1,3,j+1)
    cluster = happy_df_cluster[happy_df_cluster['cluster'] == j]
    cluster[i].hist(bins = 20)
    plt.title('{}    \nCluster {} '.format(i, j))

  plt.show()
```

MINI CHALLENGE #5:

- Try the same model with 4 clusters

In [39]:
```
kmeans = KMeans(4)
kmeans.fit(scaled_data)
labels = kmeans.labels_
y_kmeans = kmeans.fit_predict(scaled_data)
```

TASK #10: VISUALIZE THE CLUSTERS

In [40]:
```
happy_df_cluster
```

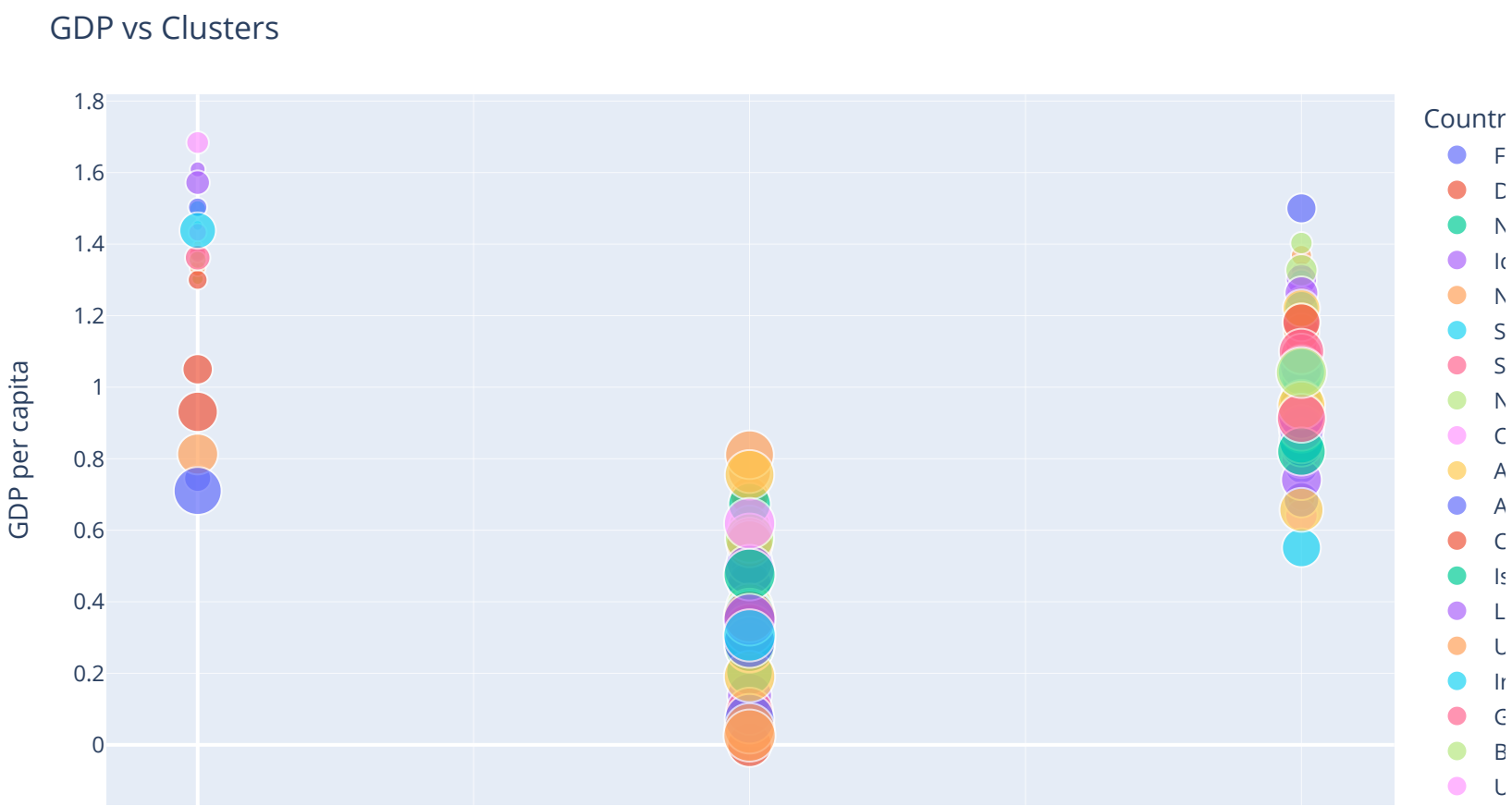
Out[40]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption	cluster
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393	0
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410	0
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341	0
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118	0
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298	0
...	...	...	...	...	...	...	...	...	...	...
151	152	Rwanda	3.334	0.359	0.711	0.614	0.555	0.217	0.411	1
152	153	Tanzania	3.231	0.476	0.885	0.499	0.417	0.276	0.147	1
153	154	Afghanistan	3.203	0.350	0.517	0.361	0.000	0.158	0.025	1
154	155	Central African Republic	3.083	0.026	0.000	0.105	0.225	0.235	0.035	1
155	156	South Sudan	2.853	0.306	0.575	0.295	0.010	0.202	0.091	1

156 rows × 10 columns

In [41]:
```
# Plot the relationship between cluster and score

fig = px.scatter(happy_df_cluster, x = 'cluster', y = "Score",
           size = 'Overall rank', color = "Country or region", hover_name = "Country or region",
          trendline = "ols")

fig.update_layout(
    title_text = 'Happiness Score vs Cluster'
)
fig.show()
```

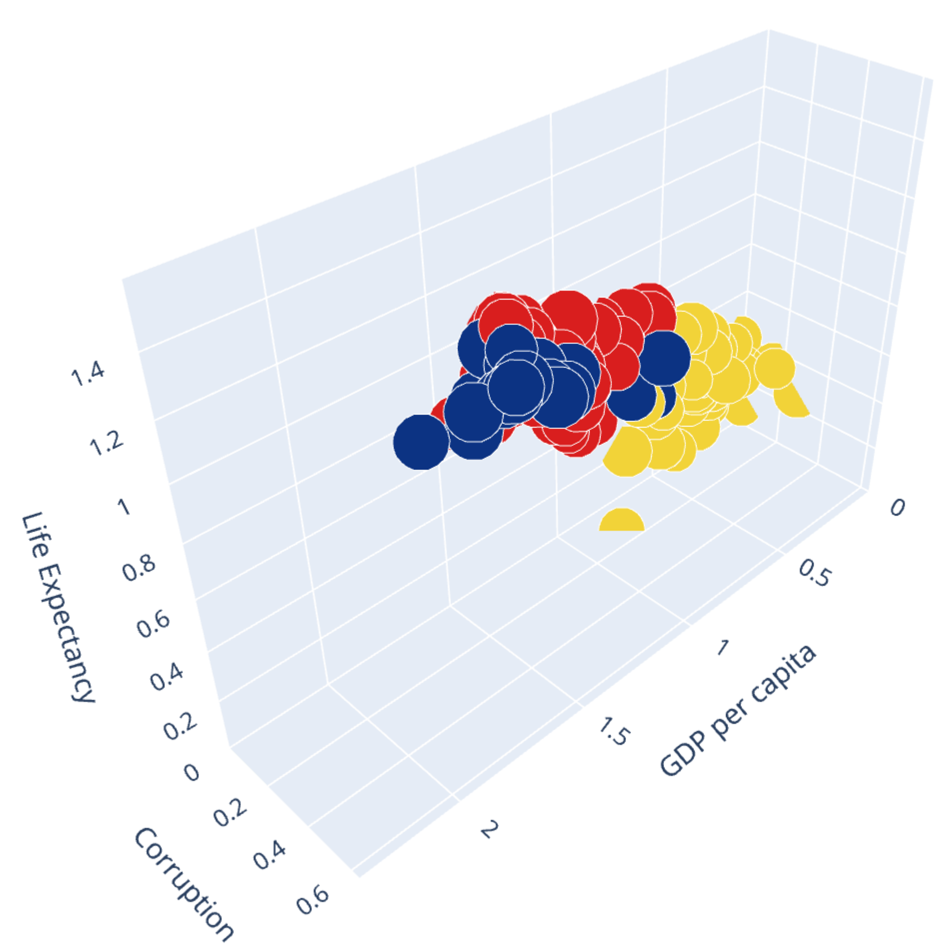
In [42]:
```
# Plot the relationship between cluster and GDP

fig = px.scatter(happy_df_cluster, x='cluster', y='GDP per capita',
           size='Overall rank', color="Country or region", hover_name="Country or region",
          trendline= "ols")

fig.update_layout(
    title_text='GDP vs Clusters'
)
fig.show()
```



In [43]:
```python
# Visaulizing the clusters with respect to economy, corruption, gdp, rank and their scores

from bubbly.bubbly import bubbleplot

figure = bubbleplot(dataset=happy_df_cluster,
    x_column='GDP per capita', y_column='Perceptions of corruption', bubble_column='Country or region',
    color_column='cluster', z_column='Healthy life expectancy', size_column='Score',
    x_title="GDP per capita", y_title="Corruption", z_title="Life Expectancy",
    title='Clusters based Impact of Economy, Corruption and Life expectancy on Happiness Scores of Nations',
    colorbar_title='Cluster', marker_opacity=1, colorscale='Portland',
    scale_bubble=0.8, height=650)

iplot(figure, config={'scrollzoom': True})
```

```
C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\plotly\offline\offline.py:157: UserWarning:


Unrecognized config options supplied: ['scrollzoom']
```

Clusters based Impact of Economy, Corruption and Life expectancy on Happiness Scores of Nations



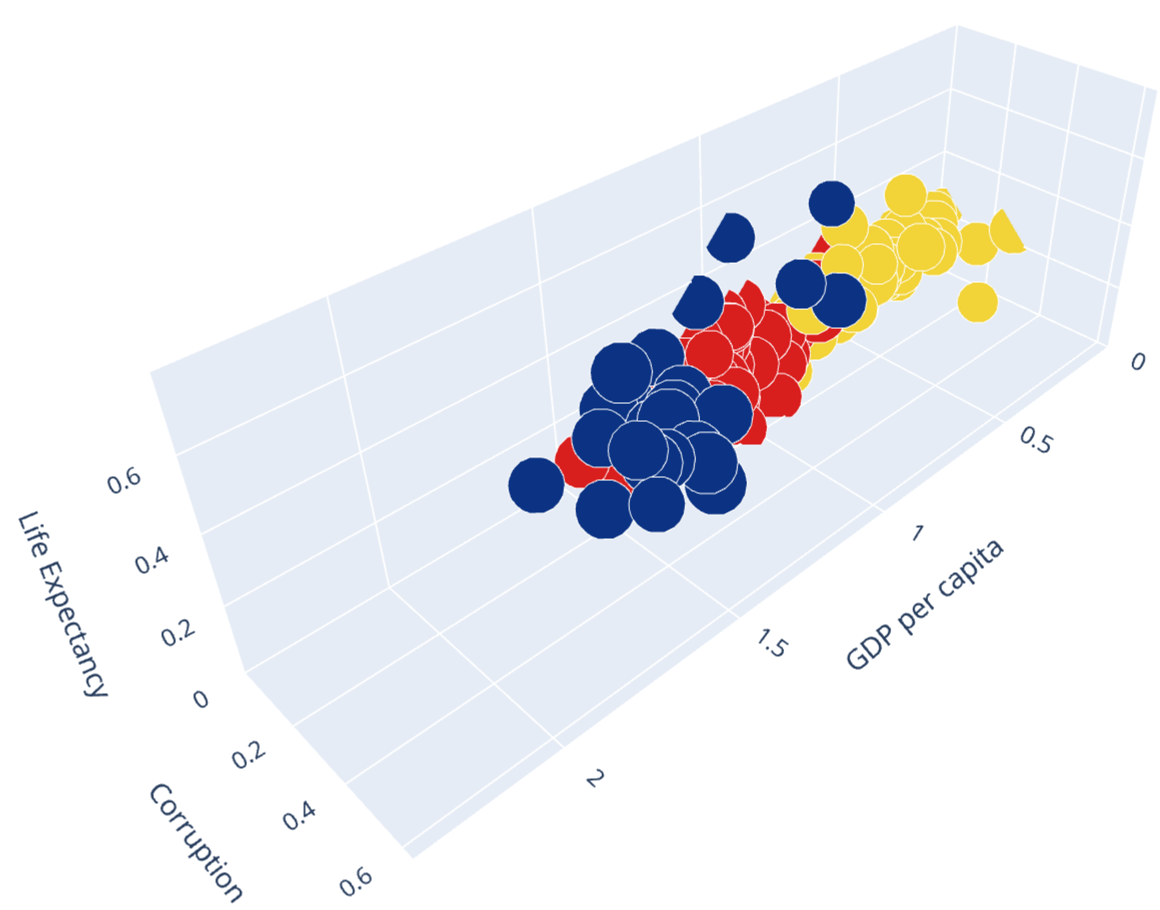
MINI CHALLENGE #6:

- Plot the similar type of visualization having 'Generosity' instead of 'Healthy life expectancy'

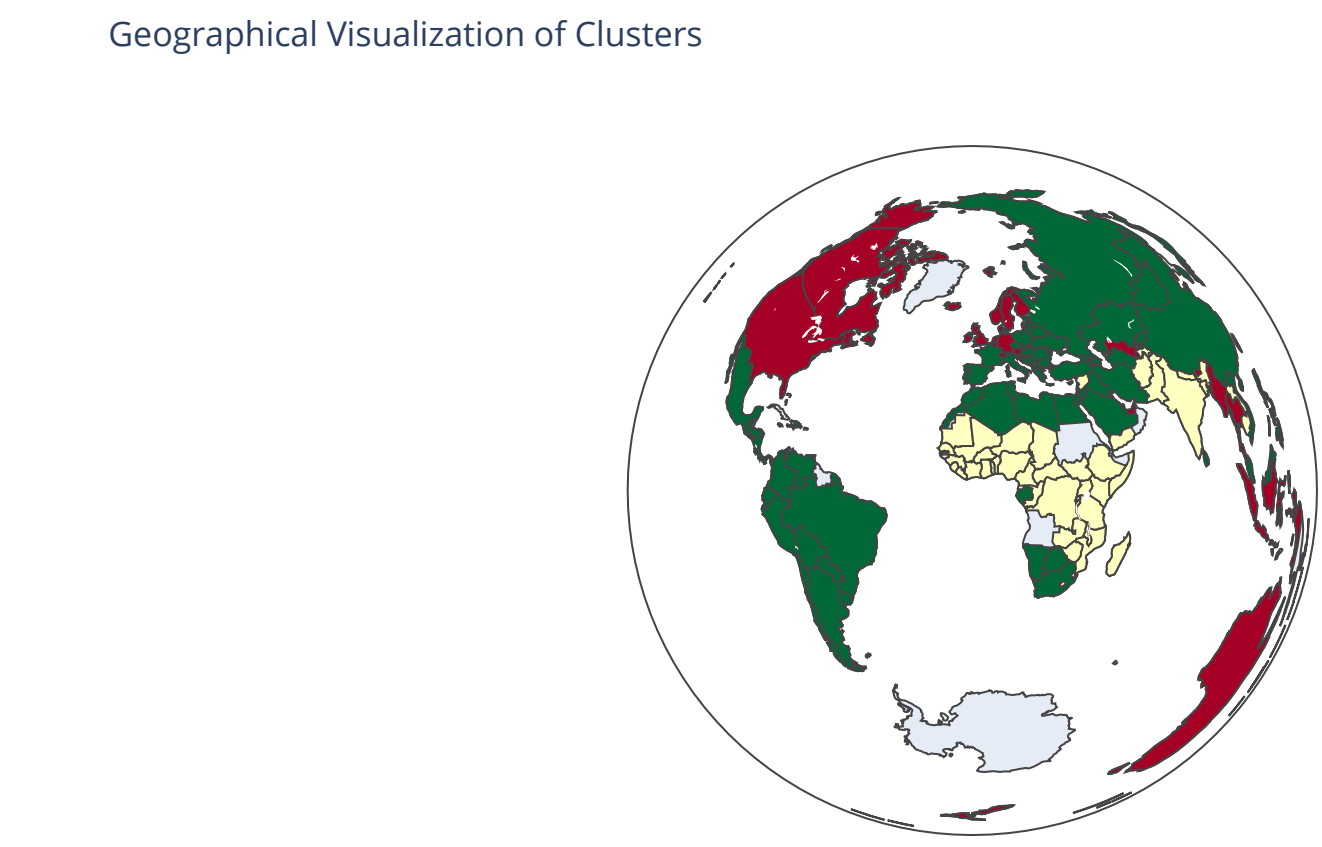
In [44]:
```python
from bubbly.bubbly import bubbleplot

figure = bubbleplot(dataset=happy_df_cluster,
    x_column='GDP per capita', y_column='Perceptions of corruption', bubble_column='Country or region',
    color_column='cluster', z_column='Generosity', size_column='Score',
    x_title="GDP per capita", y_title="Corruption", z_title="Life Expectancy",
    title='Clusters based Impact of Economy, Corruption and Generosity on Happiness Scores of Nations',
    colorbar_title='Cluster', marker_opacity=1, colorscale='Portland',
    scale_bubble=0.8, height=650)

iplot(figure, config={'scrollzoom': True})
```

```
C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\bubbly\bubbly.py:189: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.

C:\Users\SOMNATH\anaconda3\lib\site-packages\plotly\offline\offline.py:157: UserWarning:


Unrecognized config options supplied: ['scrollzoom']
```

Clusters based Impact of Economy, Corruption and Generosity on Happiness Scores of Nations



In [45]:
```
# Visualizing the clusters geographically
data = dict(type = 'choropleth',
            locations = happy_df_cluster["Country or region"],
            locationmode = 'country names',
            colorscale='RdYlGn',
            z = happy_df_cluster['cluster'],
            text = happy_df_cluster["Country or region"],
            colorbar = {'title':'Clusters'})

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

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



CONGRATULTIONS ON COMPLETING THE PROJECT!

MINI CHALLENGE #1 SOLUTION:

- Find out how many samples exist in the DataFrame using two different methods
- Select your own country from the dataframe and explore scores. Perform sanity check.

In [36]:
```
len(happy_df)
```

Out[36]: 156

In [37]:
```
happy_df
```

Out[37]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298
...	...	...	...	...	...	...	...	...	...
151	152	Rwanda	3.334	0.359	0.711	0.614	0.555	0.217	0.411
152	153	Tanzania	3.231	0.476	0.885	0.499	0.417	0.276	0.147
153	154	Afghanistan	3.203	0.350	0.517	0.361	0.000	0.158	0.025
154	155	Central African Republic	3.083	0.026	0.000	0.105	0.225	0.235	0.035
155	156	South Sudan	2.853	0.306	0.575	0.295	0.010	0.202	0.091

156 rows × 9 columns

In [38]:
```
happy_df[happy_df['Country or region']=='Canada']
```

Out[38]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
8	9	Canada	7.278	1.365	1.505	1.039	0.584	0.285	0.308

MINI CHALLENGE #2 SOLUTION:

- What is the country that has the maximum happiness score? What is the perception of corruption in this country?

In [39]:
```
happy_df[happy_df['Score'] == 7.769000]
```

Out[39]:

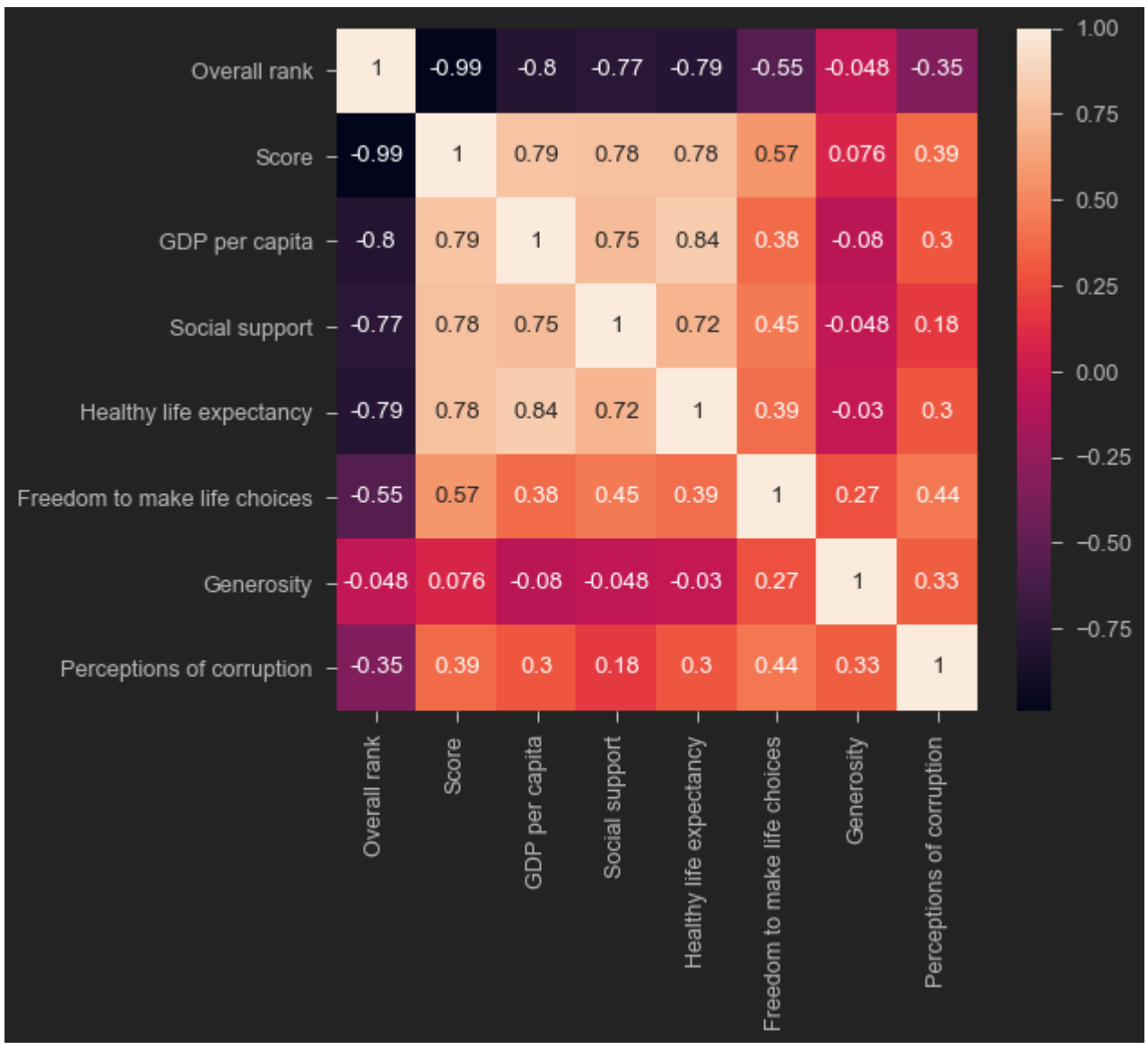
	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.34	1.587	0.986	0.596	0.153	0.393

MINI CHALLENGE #3 SOLUTION:

- Plot the correlation matrix and comment on the results.

In [40]:
```
# Get the correlation matrix
corr_matrix = happy_df.corr()
corr_matrix
sns.heatmap(corr_matrix, annot = True)
```

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x132ab89bb88>



MINI CHALLENGE #4 SOLUTION:

- Plot the similar type of plot for 'Healthy life expectancy' and 'Score'

In [41]:
```
# Plot the relationship between score and healthy life expectancy
fig = px.scatter(happy_df, x = 'Healthy life expectancy', y = "Score", text = 'Country or region')
fig.update_traces(textposition = 'top center')
fig.update_layout(height = 1000)
fig.show()
```

In [42]:
```
# Plot the relationship between score and healthy life expectancy
fig = px.scatter(happy_df, x = 'Healthy life expectancy', y = "Score",
           size = 'Overall rank', color = "Country or region", hover_name = "Country or region",
          trendline = "ols")

fig.update_layout(
    title_text = 'Happiness Score vs Healthy life expectancy'
)
fig.show()
```

MINI CHALLENGE #5 SOLUTION:

- Try the same model with 4 clusters

In [43]:
```
kmeans = KMeans(4)
kmeans.fit(scaled_data)
labels = kmeans.labels_
y_kmeans = kmeans.fit_predict(scaled_data)
```

MINI CHALLENGE #6 SOLUTION:

- Plot the similar type of visualization having 'Generosity' instead of 'Healthy life expectancy'

In [44]:
```
from bubbly.bubbly import bubbleplot

figure = bubbleplot(dataset=happy_df_cluster,
    x_column='GDP per capita', y_column='Perceptions of corruption', bubble_column='Country or region',
    color_column='cluster', z_column='Generosity', size_column='Score',
    x_title="GDP per capita", y_title="Corruption", z_title="Life Expectancy",
    title='Clusters based Impact of Economy, Corruption and Generosity on Happiness Scores of Nations',
    colorbar_title='Cluster', marker_opacity=1, colorscale='Portland',
    scale_bubble=0.8, height=650)

iplot(figure, config={'scrollzoom': True})
```

```
C:\Users\Administrator\anaconda3\lib\site-packages\plotly\offline\offline.py:160: UserWarning:


Unrecognized config options supplied: ['scrollzoom']
```