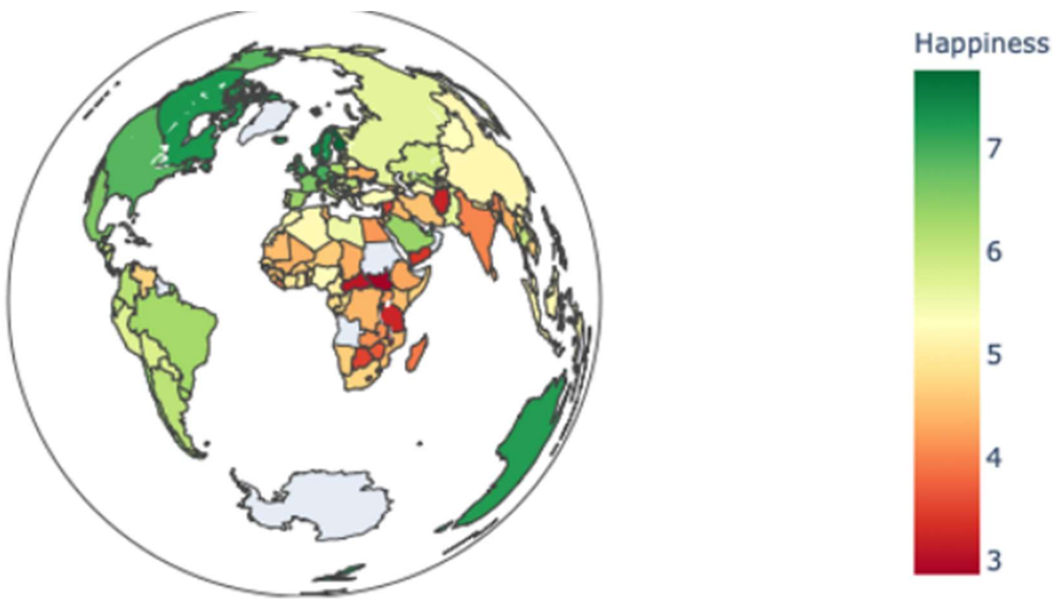
WORLD HAPPINESS REPORT USING
K-MEANS CLUSTERING

<u>Aim:</u>

- In this case study, we will train an unsupervised machine learning algorithm to cluster countries based on features such as economic expectancy, freedom, absence of corruption, and generosity.
- A dataset called **“The World Happiness Report”** has been taken from **Kaggle**.
- The World Happiness Report determines the state 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).



Description:

Why K-means clustering?

- The K-means clustering algorithm is used to find groups which have not been explicitly labeled in the data.
- It works really well with large datasets.
- It is an optimal solution for most problems.
- K-means is simple and easy to implement and run.
- All you need to do is choose "k" and run it a number of times.

Procedure:

First, we import the dataset and necessary Python libraries to start this task:

In [2]:
```python
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 [26]:
```python
# Import csv file into pandas dataframe
df=pd.read_csv(r'C:\Users\jai\Desktop\happiness_report.csv')
df
```

Out[26]:

	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

Now we perform Exploratory Data Analysis:

In [11]:
```python
df.describe()
```

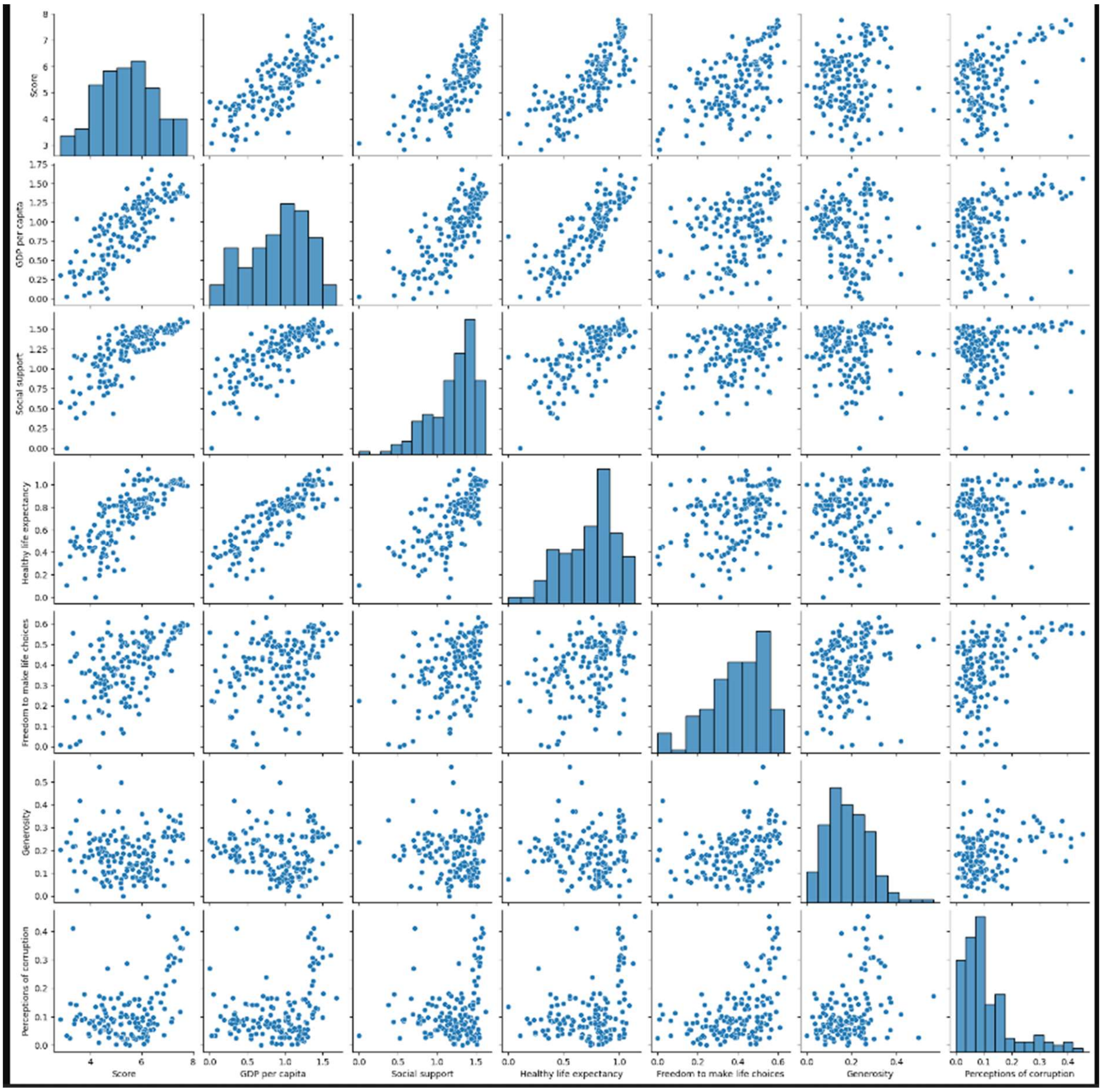
Out[11]:

	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

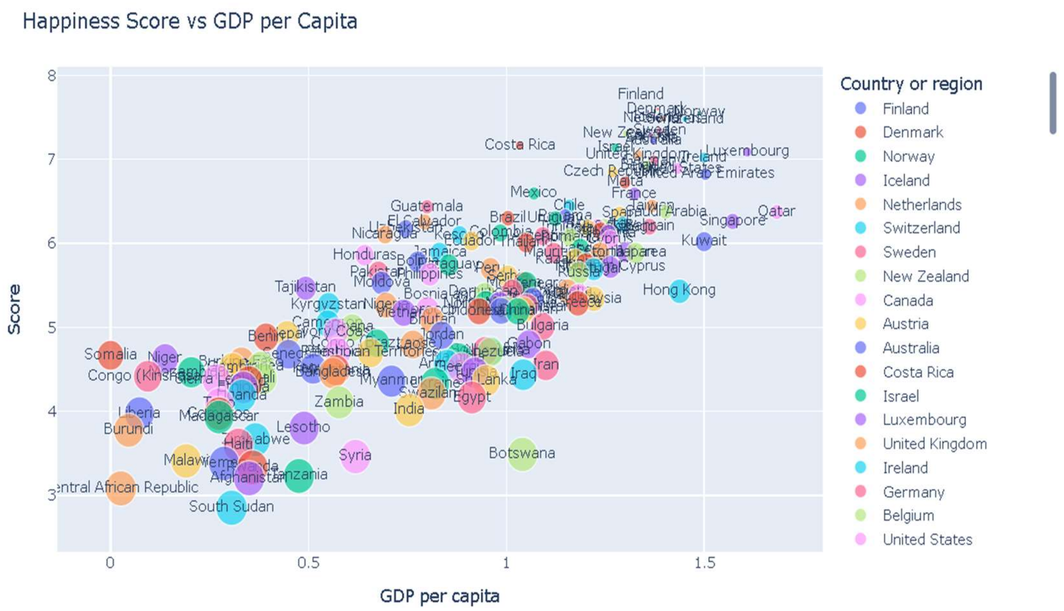
Let’s use some Data Visualization for better understanding:

- We get different insights from rows and columns for score, GDP per capita, etc.
- A scatter plot is observed which shows the correlation among different features.
- It looks like a linear curve or positive correlation between score and GDP per capita.
- Since GDP per capita goes up i.e; there are more jobs available, people tend to be happy.

In [22]:
```
from matplotlib import pyplot as plt
import seaborn as sns
fig = plt.figure(figsize = (20,20))
sns.pairplot(df[['Score', 'GDP per capita', 'Social support', 'Healthy life expectancy', 'Freedom to make
```

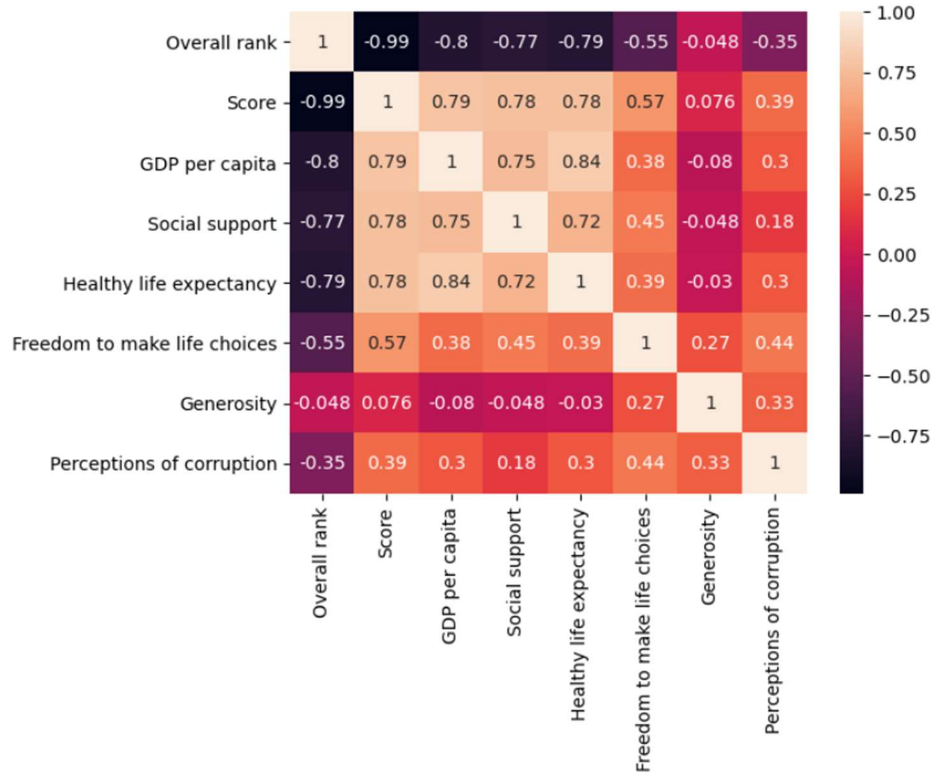

Score
GDP per capita
Social support
Healthy life expectancy
Freedom to make life choices
Generosity
Perceptions of corruption

```
In [13]: # Plot the relationship between score and GDP (while adding color and size)
         fig = px.scatter(df, x='GDP per capita', y='Score', text='Country or region', size='Overall rank', color='Country or region
         fig.update_layout(title_text = 'Happiness Score vs GDP per Capita')
         fig.show()
```



Heatmap:

```
In [24]: sns.heatmap(corr_matrix, annot=True)
Out[24]: <AxesSubplot:>
```



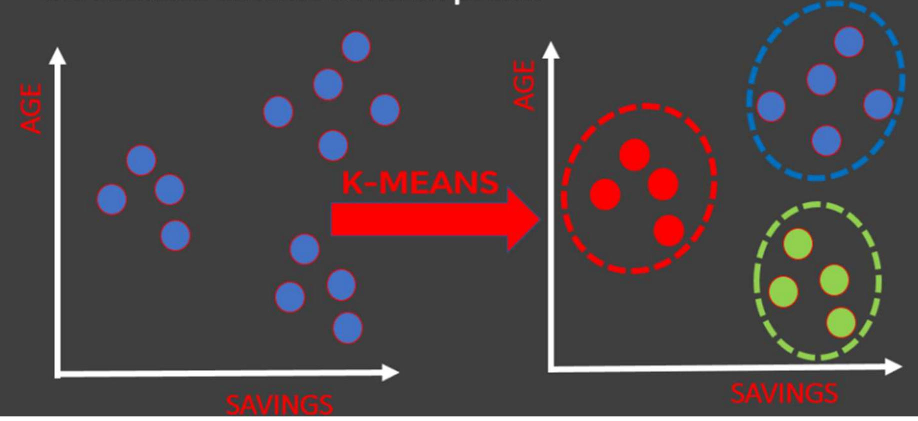
K-means algorithm:

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.

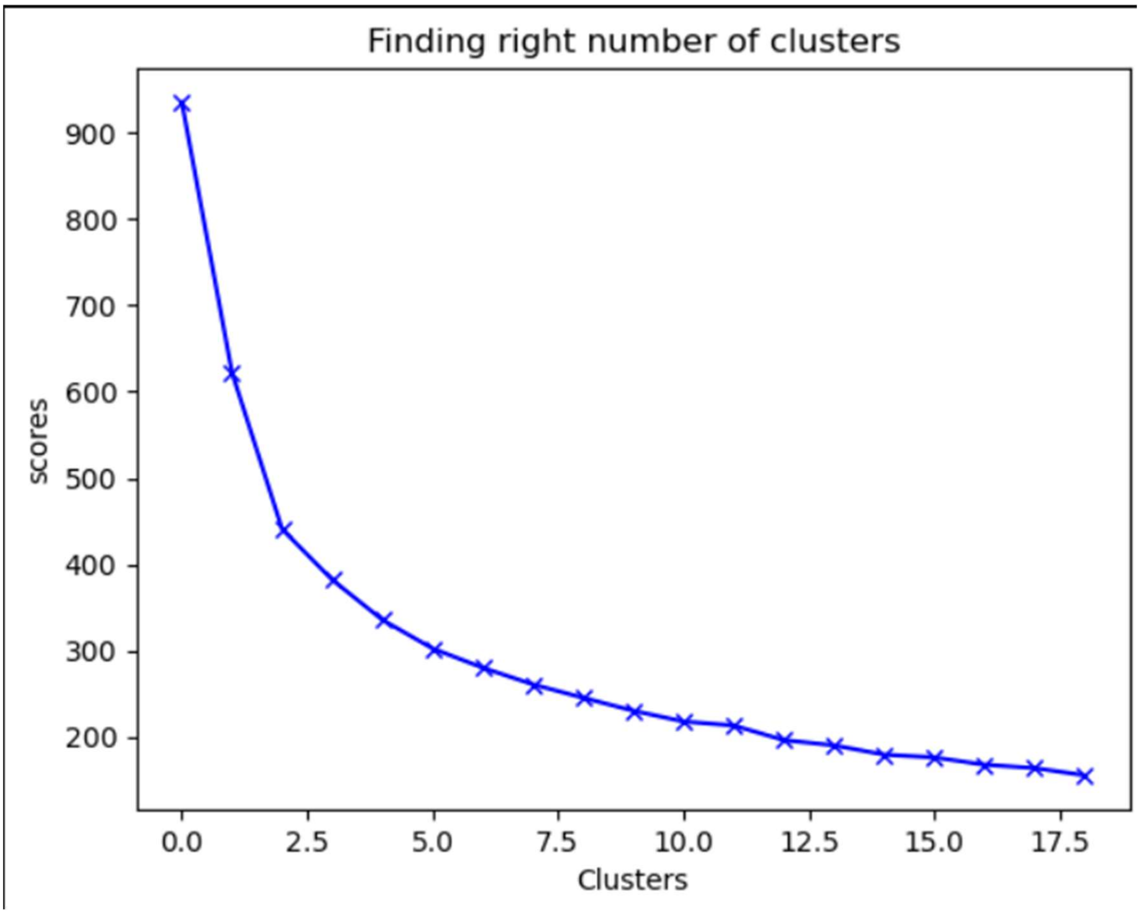
K-MEANS INTUITON

- **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.**



Finding the optimal number of clusters using KMeans library:

```
In [39]: from sklearn.cluster import KMeans
         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()
```



Finally, we visualize the clusters obtained from k-means:

- Red defines high healthy life expectancy
- Green defines intermediate healthy life expectancy

- White defines low healthy life expectancy

```
In [25]: # Visualizing the clusters geographically
         import plotly.graph_objects as go
         data = dict(type = 'choropleth',
                    locations = df_cluster["Country or region"],
                    locationmode = 'country names',
                    colorscale='RdYlGn',
                    z = df_cluster['cluster'],
                    text = df_cluster["Country or region"],
                    colorbar = {'title':'Clusters'})

         layout = dict(title = 'Geographical Visualization of Clusters',
                       geo = dict(showframe = True, projection = {'type': 'azimuthal equa

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

Geographical Visualization of Clusters

Conclusion:

So, this is how we create and visualize the “World Happiness Report” (which determines the state global happiness) using k-means clustering.