World Happiness Report 2021 Analysis

World Happiness Report is basically a survey of state of global happiness that ranks 149 countries around the world on how happy their citizens perceive themselves to be based on various indicators.The happiness study ranks the countries on the basis of questions from the Gallup World Poll. The results are then equated with other factors such as GDP, life expectancy, generosity, etc. In the year 2021, it focused on the effects of the Covid-19 pandemic and how people all over the world have managed to survive and prosper.

This analysis therefore brings forward those relationships between various indicators and represent them as a visualization on various graphs to bring a reasonable understanding.

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
sns.set_style('darkgrid')
plt.rcParams['font.size'] = 15
plt.rcParams['figure.figsize'] = (10,7)
plt.rcParams['figure.facecolor'] = '#FFE5B4'
```

In [3]:
```
df = pd.read_csv("World Happiness Report 2021.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	Country name	Regional indicator	Ladder score	Standard error of ladder score	upperwhisker	lowerwhisker	Logged GDP per capita	Social support	He expec
0	Finland	Western Europe	7.842	0.032	7.904	7.780	10.775	0.954	
1	Denmark	Western Europe	7.620	0.035	7.687	7.552	10.933	0.954	
2	Switzerland	Western Europe	7.571	0.036	7.643	7.500	11.117	0.942	
3	Iceland	Western Europe	7.554	0.059	7.670	7.438	10.878	0.983	
4	Netherlands	Western Europe	7.464	0.027	7.518	7.410	10.932	0.942	

In [5]:
```
df.rename(columns={'Ladder score': 'Happiness score'}, inplace=True)
```

In [6]:
```
df.head()
```

Out[6]:

	Country name	Regional indicator	Happiness score	Standard error of ladder score	upperwhisker	lowerwhisker	Logged GDP per capita	Social support	ex
0	Finland	Western Europe	7.842	0.032	7.904	7.780	10.775	0.954	
1	Denmark	Western Europe	7.620	0.035	7.687	7.552	10.933	0.954	
2	Switzerland	Western Europe	7.571	0.036	7.643	7.500	11.117	0.942	
3	Iceland	Western Europe	7.554	0.059	7.670	7.438	10.878	0.983	
4	Netherlands	Western Europe	7.464	0.027	7.518	7.410	10.932	0.942	

In [7]:
```
df_columns = ['Country name','Regional indicator','Happiness score','Logged GDP per
```

In [8]:
```
data =df[df_columns].copy()
```

In [9]:
```
df = data.rename(columns={'Country name':'country_name',
                          'Regional indicator':'regional_indicator',
                          'Happiness score':'happiness_score',
                          'Logged GDP per capita':'logged_GDP_per_capita',
                          'Social support': 'social_support',
                          'Healthy life expectancy':'healthy_life_expectancy',
                          'Freedom to make life choices':'freedom_to_make_life_choice
                          'Generosity':'generosity',
                          'Perceptions of corruption':'perceptions_of_corruption'},ir
```

In [10]:
```
data.head()
```

Out[10]:

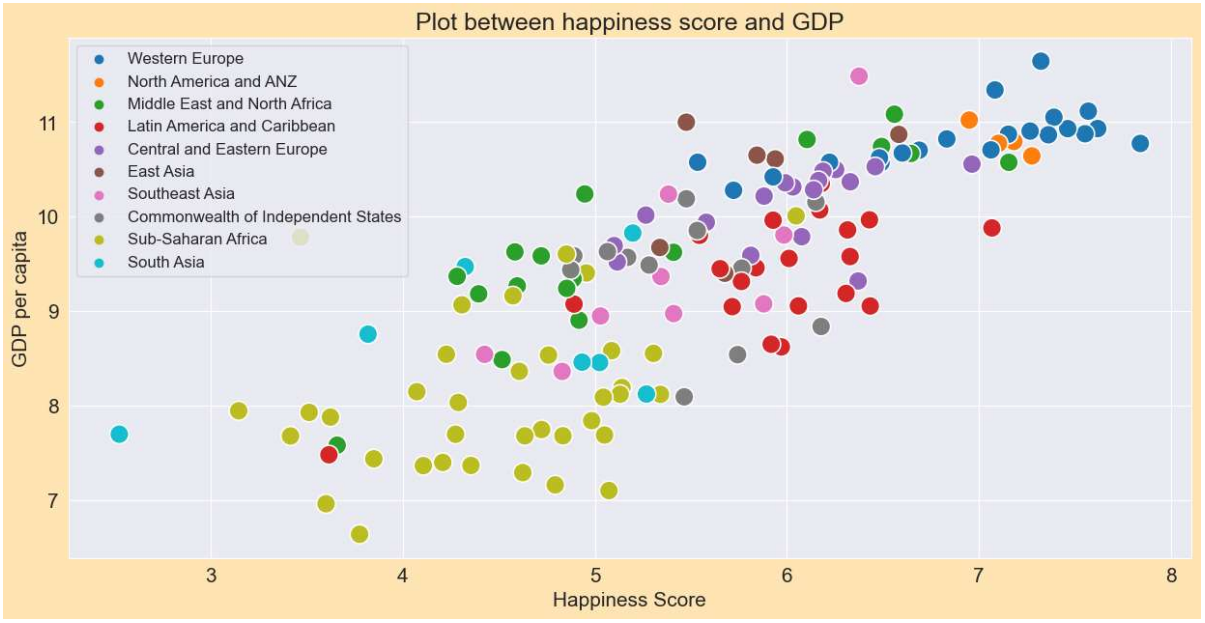
	country_name	regional_indicator	happiness_score	logged_GDP_per_capita	social_support	healt
0	Finland	Western Europe	7.842	10.775	0.954	
1	Denmark	Western Europe	7.620	10.933	0.954	
2	Switzerland	Western Europe	7.571	11.117	0.942	
3	Iceland	Western Europe	7.554	10.878	0.983	
4	Netherlands	Western Europe	7.464	10.932	0.942	

In [11]:
```
data.isnull().sum()
```

Out[11]:
```
country_name                  0
regional_indicator            0
happiness_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]:
```
#plot b/w happiness score and GDP
plt.rcParams['figure.figsize'] = (15,7)
plt.title('Plot between happiness score and GDP')
sns.scatterplot(x = data.happiness_score , y = data.logged_GDP_per_capita,hue = dat
plt.legend(loc ='upper left',fontsize = '12')
plt.xlabel('Happiness Score')
plt.ylabel('GDP per capita')
```

Out[12]:
```
Text(0, 0.5, 'GDP per capita')
```

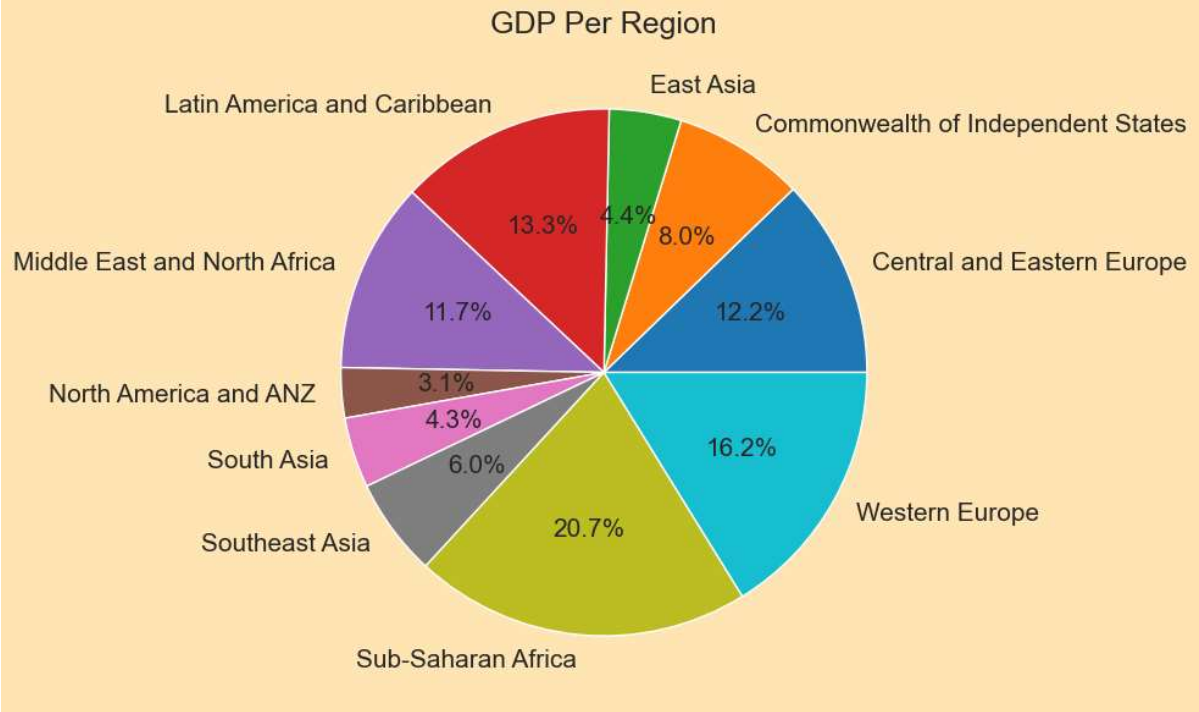


In [13]:
```
gdp_region = data.groupby('regional_indicator')['logged_GDP_per_capita'].sum()
gdp_region
```

Out[13]:
```
regional_indicator
Central and Eastern Europe            171.854
Commonwealth of Independent States    112.822
East Asia                              62.206
Latin America and Caribbean           187.400
Middle East and North Africa          164.324
North America and ANZ                  43.238
South Asia                             60.778
Southeast Asia                         84.793
Sub-Saharan Africa                    290.707
Western Europe                        227.277
Name: logged_GDP_per_capita, dtype: float64
```

In [14]:
```
gdp_region.plot.pie(autopct = '%1.1f%%')
plt.title('GDP Per Region')
plt.ylabel('')
```

Out[14]:
```
Text(0, 0.5, '')
```



In [15]:
```
#Total Countries

total_country = data.groupby('regional_indicator')[['country_name']].count()
print(total_country)
```

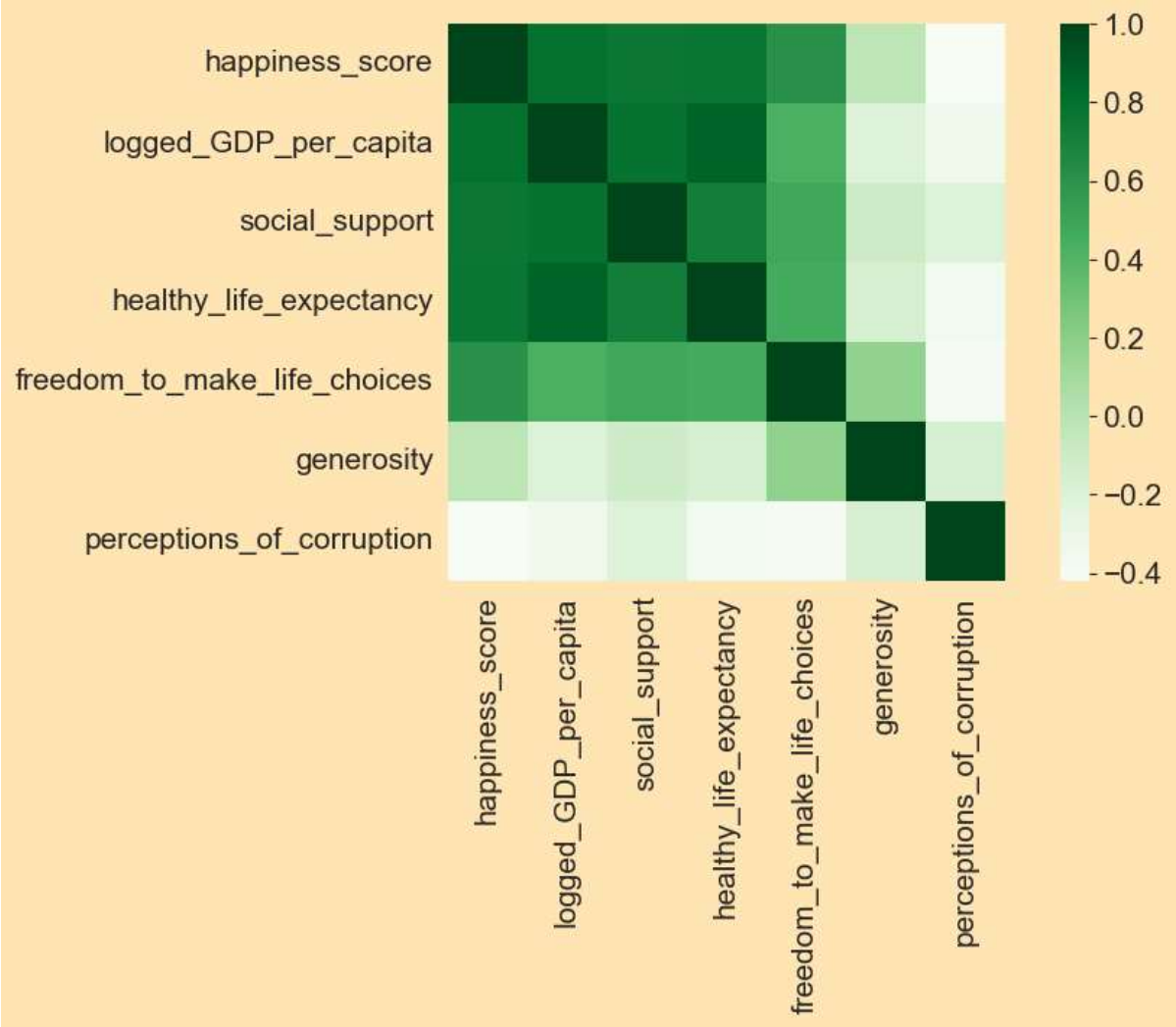
```
                                country_name
regional_indicator
Central and Eastern Europe                17
Commonwealth of Independent States        12
East Asia                                  6
Latin America and Caribbean               20
Middle East and North Africa              17
North America and ANZ                      4
South Asia                                 7
Southeast Asia                             9
Sub-Saharan Africa                        36
Western Europe                            21
```

In [16]:
```
#Correlation Map

cor = data.corr(method ="pearson")
f, ax = plt.subplots(figsize = (10,5))
sns.heatmap(cor,mask = np.zeros_like(cor, dtype = np.bool),
            cmap = "Greens",square=True, ax=ax)
```

```
C:\Users\Garima\AppData\Local\Temp\ipykernel_21840\4236780290.py:3: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
sion, it will default to False. Select only valid columns or specify the value of
numeric_only to silence this warning.
  cor = data.corr(method ="pearson")
C:\Users\Garima\AppData\Local\Temp\ipykernel_21840\4236780290.py:5: DeprecationWar
ning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this warn
ing, use `bool` by itself. Doing this will not modify any behavior and is safe. If
you specifically wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdoc
s/release/1.20.0-notes.html#deprecations
  sns.heatmap(cor,mask = np.zeros_like(cor, dtype = np.bool),
```

Out[16]:
```
<Axes: >
```



#Corruption in Regions

In [17]:
```
corruption = data.groupby('regional_indicator')[['perceptions_of_corruption']].mean
corruption
```

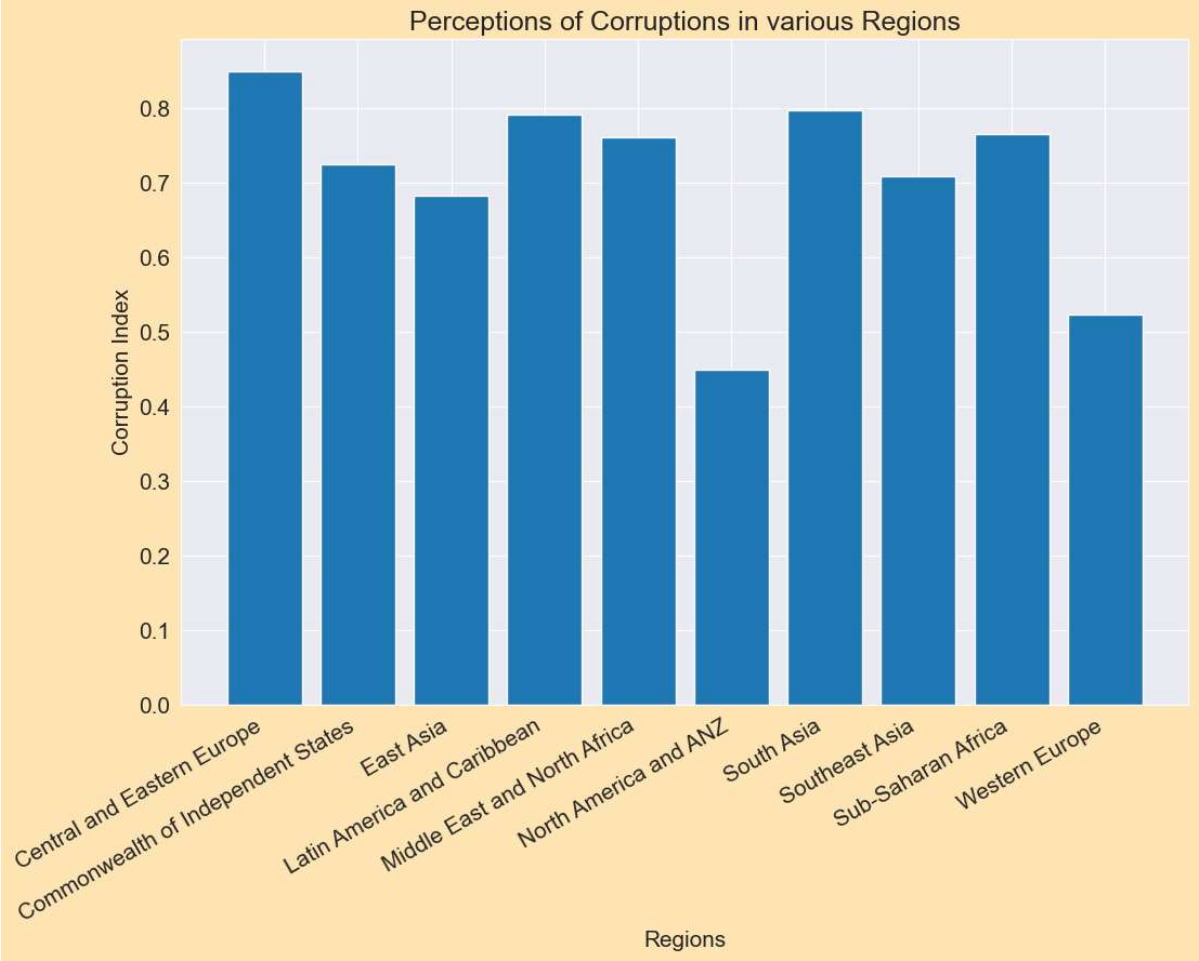
Out[17]:

regional_indicator	perceptions_of_corruption
Central and Eastern Europe	0.850529
Commonwealth of Independent States	0.725083
East Asia	0.683333
Latin America and Caribbean	0.792600
Middle East and North Africa	0.762235
North America and ANZ	0.449250
South Asia	0.797429
Southeast Asia	0.709111
Sub-Saharan Africa	0.765944
Western Europe	0.523095

In [18]:
```
plt.rcParams['figure.figsize'] = (12,8)
plt.title('Perceptions of Corruptions in various Regions')
plt.xlabel('Regions', fontsize =15)
plt.ylabel('Corruption Index', fontsize = 15)
```

```
plt.xticks(rotation = 30,ha='right')
plt.bar(corruption.index,corruption.perceptions_of_corruption)
```

Out[18]:
```
<BarContainer object of 10 artists>
```



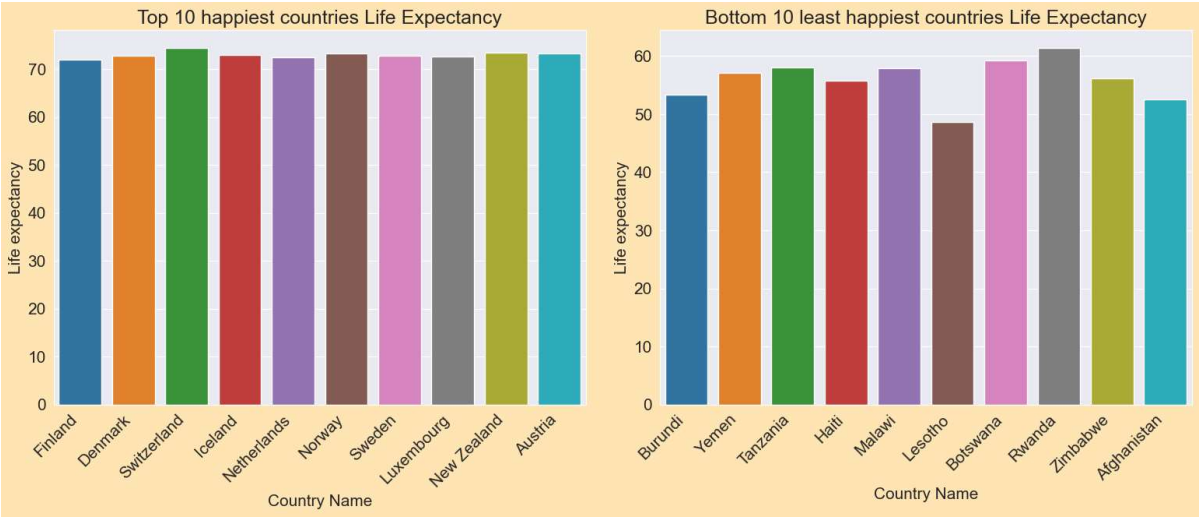
In [19]:
```
top_10 = data.head(10)
bottom_10 =data.tail(10)
```

In [20]:
```
fig,axes = plt.subplots(1,2, figsize = (16,6))
plt.tight_layout(pad=2)
xlabels= top_10.country_name
axes[0].set_title('Top 10 happiest countries Life Expectancy')
axes[0].set_xticklabels(xlabels,rotation=45,ha='right')
sns.barplot(x = top_10.country_name,y = top_10.healthy_life_expectancy,ax= axes[0])
axes[0].set_xlabel('Country Name')
axes[0].set_ylabel('Life expectancy')

xlabels= bottom_10.country_name
axes[1].set_title('Bottom 10 least happiest countries Life Expectancy')
axes[1].set_xticklabels(xlabels,rotation=45,ha='right')
sns.barplot(x = bottom_10.country_name, y = bottom_10.healthy_life_expectancy, ax=
axes[1].set_xlabel('Country Name')
axes[1].set_ylabel('Life expectancy')
```

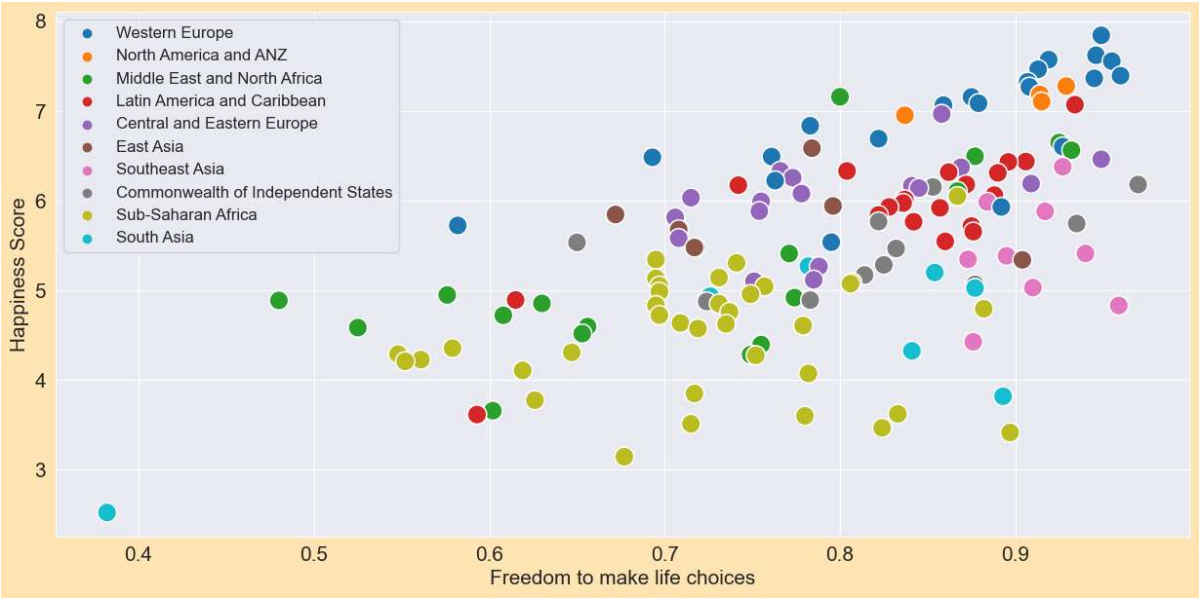
```
C:\Users\Garima\AppData\Local\Temp\ipykernel_21840\468301256.py:5: UserWarning: Fi
xedFormatter should only be used together with FixedLocator
  axes[0].set_xticklabels(xlabels,rotation=45,ha='right')
C:\Users\Garima\AppData\Local\Temp\ipykernel_21840\468301256.py:12: UserWarning: F
ixedFormatter should only be used together with FixedLocator
  axes[1].set_xticklabels(xlabels,rotation=45,ha='right')
```

Out[20]:
```
Text(832.0858585858584, 0.5, 'Life expectancy')
```



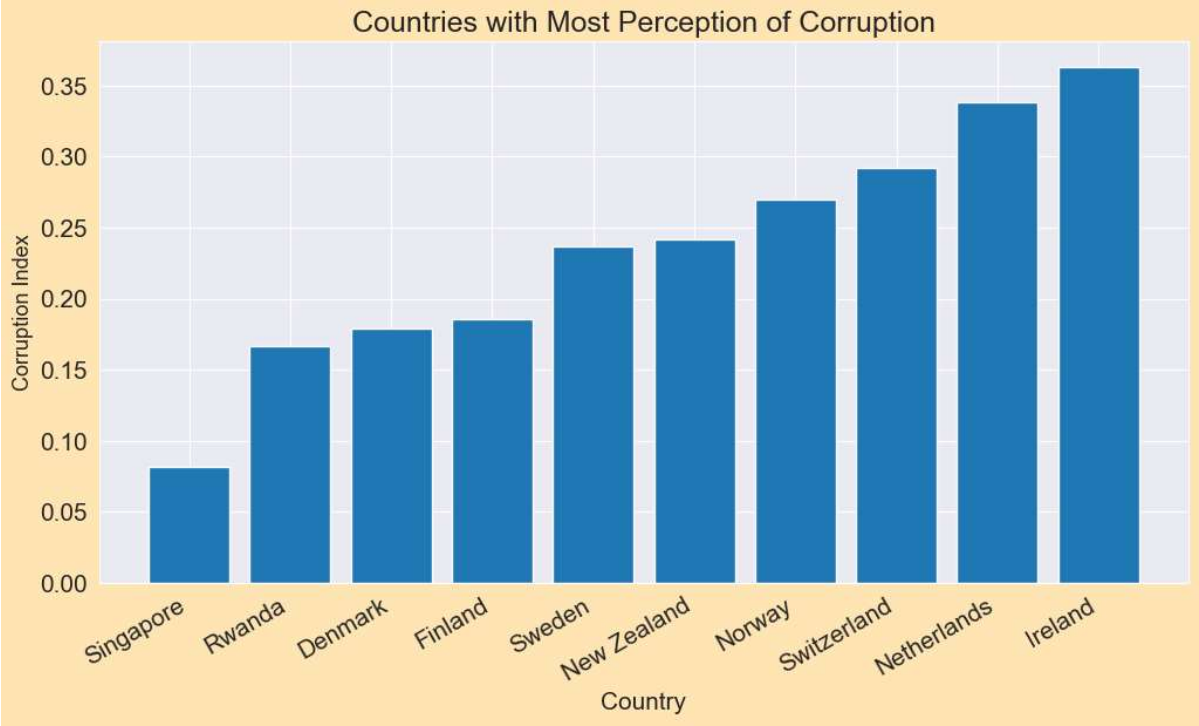
In [21]:
```
plt.rcParams['figure.figsize'] = (15,7)
sns.scatterplot(x = data.freedom_to_make_life_choices , y = data.happiness_score,hu
plt.legend(loc ='upper left',fontsize = '12')
plt.xlabel('Freedom to make life choices')
plt.ylabel('Happiness Score')
```

Out[21]:
```
Text(0, 0.5, 'Happiness Score')
```



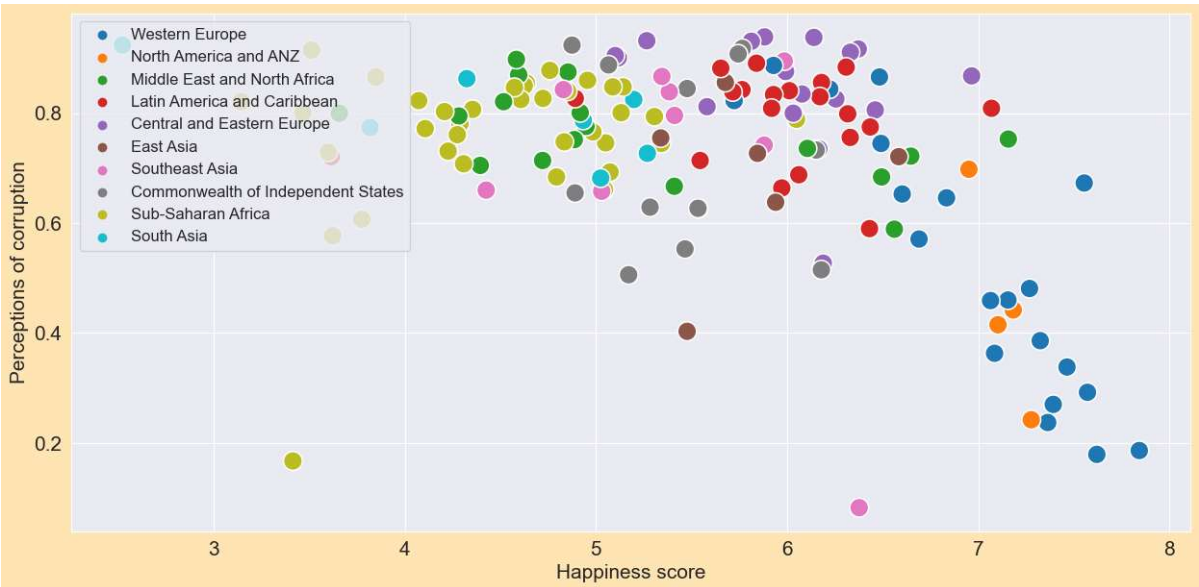
In [22]:
```
country = data.sort_values(by = 'perceptions_of_corruption').head(10)
plt.rcParams['figure.figsize'] = (12,6)
plt.title('Countries with Most Perception of Corruption')
plt.xlabel('Country', fontsize =15)
plt.ylabel('Corruption Index', fontsize = 13)
plt.xticks(rotation = 30,ha='right')
plt.bar(country.country_name,country.perceptions_of_corruption)
```

Out[22]:
```
<BarContainer object of 10 artists>
```



In [23]:
```
#Corruption Vs Happiness
plt.rcParams['figure.figsize'] = (15,7)
sns.scatterplot(x = data.happiness_score , y = data.perceptions_of_corruption,hue =
plt.legend(loc ='upper left',fontsize = '12')
plt.xlabel('Happiness score')
plt.ylabel('Perceptions of corruption')
```

Out[23]:
```
Text(0, 0.5, 'Perceptions of corruption')
```



In [ ]: