HAPPINES SCORE DATASET

In [264]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv ("happiness_score_dataset.csv")
df.head()
```

Out[2]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176

In [3]:
```
# upto here we are uploded "happiness_score_dataset.csv" to jupyter notebook.
# and make df as a instance of our insurance dataset.
```

In [4]:
```
df.shape
# here we finds the shape of our dataset, i.e it containing 158 ROWS & 12 COLUMNS
```

Out[4]:
```
(158, 12)
```

In [7]:
```
df.columns
# here we finds the names of different columns of the data set.
```

Out[7]:
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

In [9]:
```
df.dtypes
# here can see that there are some different types of data is present in the given dataset like : [ int64, object, flo
#  here in the following details we find that there are only 2 columns which are having "object" datatype.
# & rest of the columns are having "int64" & "flot64" datatype.
```

Out[9]:
```
Country                          object
Region                           object
Happiness Rank                    int64
Happiness Score                 float64
Standard Error                  float64
Economy (GDP per Capita)        float64
Family                          float64
Health (Life Expectancy)        float64
Freedom                         float64
Trust (Government Corruption)   float64
Generosity                      float64
Dystopia Residual               float64
dtype: object
```

In [18]:
```
df.info()
#  here we can see that
# 1) total number for columns present : 12
# 2) total number of rows presnet : 158
# 3) total "data types present in data set" : 3 (i.e "object, int64 & float64")
#  out of which   9 columns of - float64
#                 1 column of - int64
#                 2 columns of - object
# 4) NO NULL VALUES are present in our dataset.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 12 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Country                        158 non-null    object 
 1   Region                         158 non-null    object 
 2   Happiness Rank                 158 non-null    int64  
 3   Happiness Score                158 non-null    float64
 4   Standard Error                 158 non-null    float64
 5   Economy (GDP per Capita)       158 non-null    float64
 6   Family                         158 non-null    float64
 7   Health (Life Expectancy)       158 non-null    float64
 8   Freedom                        158 non-null    float64
 9   Trust (Government Corruption)  158 non-null    float64
 10  Generosity                     158 non-null    float64
 11  Dystopia Residual              158 non-null    float64
dtypes: float64(9), int64(1), object(2)
memory usage: 14.9+ KB
```

In [14]:
```
df["Country"].unique()
#  here below we find the total unique countries name available in the "Country" column
```

Out[14]:
```
array(['Switzerland', 'Iceland', 'Denmark', 'Norway', 'Canada', 'Finland',
       'Netherlands', 'Sweden', 'New Zealand', 'Australia', 'Israel',
       'Costa Rica', 'Austria', 'Mexico', 'United States', 'Brazil',
       'Luxembourg', 'Ireland', 'Belgium', 'United Arab Emirates',
       'United Kingdom', 'Oman', 'Venezuela', 'Singapore', 'Panama',
       'Germany', 'Chile', 'Qatar', 'France', 'Argentina',
       'Czech Republic', 'Uruguay', 'Colombia', 'Thailand',
       'Saudi Arabia', 'Spain', 'Malta', 'Taiwan', 'Kuwait', 'Suriname',
       'Trinidad and Tobago', 'El Salvador', 'Guatemala', 'Uzbekistan',
       'Slovakia', 'Japan', 'South Korea', 'Ecuador', 'Bahrain', 'Italy',
       'Bolivia', 'Moldova', 'Paraguay', 'Kazakhstan', 'Slovenia',
       'Lithuania', 'Nicaragua', 'Peru', 'Belarus', 'Poland', 'Malaysia',
       'Croatia', 'Libya', 'Russia', 'Jamaica', 'North Cyprus', 'Cyprus',
       'Algeria', 'Kosovo', 'Turkmenistan', 'Mauritius', 'Hong Kong',
       'Estonia', 'Indonesia', 'Vietnam', 'Turkey', 'Kyrgyzstan',
       'Nigeria', 'Bhutan', 'Azerbaijan', 'Pakistan', 'Jordan',
       'Montenegro', 'China', 'Zambia', 'Romania', 'Serbia', 'Portugal',
       'Latvia', 'Philippines', 'Somaliland region', 'Morocco',
       'Macedonia', 'Mozambique', 'Albania', 'Bosnia and Herzegovina',
```

In [15]:
```
df["Country"].nunique()
# there are total 158 unique country names are present in the country column
```

Out[15]: 158

In [16]:
```
df["Region"].nunique()
# there 10 unique Regions are present in the region column
```

Out[16]: 10

In [17]:
```
df["Region"].unique()
# here we can find the all unique regions present in the region clumn.
```

Out[17]:
```
array(['Western Europe', 'North America', 'Australia and New Zealand',
       'Middle East and Northern Africa', 'Latin America and Caribbean',
       'Southeastern Asia', 'Central and Eastern Europe', 'Eastern Asia',
       'Sub-Saharan Africa', 'Southern Asia'], dtype=object)
```

In [29]:
```
df['Region'].value_counts()
# here we can see that the MAXIMUM COUNTRIES present in dataset is from - "Sub-Saharan Africa Region" = 40
# and then it will go furthe deacresing to other regions.
```

Out[29]:
```
Sub-Saharan Africa                 40
Central and Eastern Europe         29
Latin America and Caribbean        22
Western Europe                     21
Middle East and Northern Africa    20
Southeastern Asia                   9
Southern Asia                       7
Eastern Asia                        6
North America                       2
Australia and New Zealand           2
Name: Region, dtype: int64
```

In [20]:
```
df["Happiness Rank"].unique()
# here we can find that the happiness for all countries in between 1 - 158
```

Out[20]:
```
array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,  13,
        14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,
        27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  84,  85,  86,  87,  88,  89,  90,  91,  92,
        93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103, 104, 105,
       106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118,
       119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131,
       132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144,
       145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157,
       158], dtype=int64)
```

In [26]:
```
# from the above we find that the "region" & "country" is the only "categorical" columns present in the data set
```

In [30]:
```
df.describe()
```

Out[30]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	0.143422	0.237296	2.098977
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	0.120034	0.126685	0.553550
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.328580
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	0.061675	0.150553	1.759410
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	0.107220	0.216130	2.095415
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	0.180255	0.309883	2.462415
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	0.551910	0.795880	3.602140

In [31]:
```
#  here we are getting information like count, mean,std,min,max,25%,50% and 75%
#  here we observe that there is huge difference between 75 percentile & max in "Trust (Government Corruption)" & "Ha
#  so from this we can assume that the presence of OUTLIERS is there in the "Trust (Government Corruption)" & "Happine
#  but we can also conform this further from some other techniques also.
# here above we also finds that "STANDARD DEVIATION" is very high in "Happiness Rank". = very high SKEWNESS
```

CHECKING NULL VALUES
==

In [32]:
```
#  Checking Null Values ====>
```

In [33]:
```
df.isnull().sum()
# here we find that there is not a single null value present in our dataset.
```

Out[33]:
```
Country                          0
Region                           0
Happiness Rank                   0
Happiness Score                  0
Standard Error                   0
Economy (GDP per Capita)         0
Family                           0
Health (Life Expectancy)         0
Freedom                          0
Trust (Government Corruption)    0
Generosity                       0
Dystopia Residual                0
dtype: int64
```

In [ ]:

CHECKING CORRELATION (NON GRAPHICALLY)
===

In [41]:
```
#  Checking Correlations between the columns ====>
```

In [42]:
```
dfcor = df.corr()
dfcor
# for strogly Negative correlation = -1
#  for strogly positive correlation = +1
```

Out[42]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
Happiness Rank	1.000000	-0.992105	0.158516	-0.785267	-0.733644	-0.735613	-0.556886	-0.372315	-0.160142	-0.521999
Happiness Score	-0.992105	1.000000	-0.177254	0.780966	0.740605	0.724200	0.568211	0.395199	0.180319	0.530474
Standard Error	0.158516	-0.177254	1.000000	-0.217651	-0.120728	-0.310287	-0.129773	-0.178325	-0.088439	0.083981
Economy (GDP per Capita)	-0.785267	0.780966	-0.217651	1.000000	0.645299	0.816478	0.370300	0.307885	-0.010465	0.040059
Family	-0.733644	0.740605	-0.120728	0.645299	1.000000	0.531104	0.441518	0.205605	0.087513	0.148117
Health (Life Expectancy)	-0.735613	0.724200	-0.310287	0.816478	0.531104	1.000000	0.360477	0.248335	0.108335	0.018979
Freedom	-0.556886	0.568211	-0.129773	0.370300	0.441518	0.360477	1.000000	0.493524	0.373916	0.062783
Trust (Government Corruption)	-0.372315	0.395199	-0.178325	0.307885	0.205605	0.248335	0.493524	1.000000	0.276123	-0.033105
Generosity	-0.160142	0.180319	-0.088439	-0.010465	0.087513	0.108335	0.373916	0.276123	1.000000	-0.101301
Dystopia Residual	-0.521999	0.530474	0.083981	0.040059	0.148117	0.018979	0.062783	-0.033105	-0.101301	1.000000

In [ ]:

UNIVARIATE ANALYSIS
===

In [43]:
```
# here we can analyse individual columns with graphical representation.
```

In [45]:
```
df.head(2)
```

Out[45]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201

In [173]:
```
top_10_Happiest_countries = df['Country'].head(10)
top_10_Happiest_countries

# here following we can see the TOP 10 HAPPIEST COUNTRIES in the WORLD
```

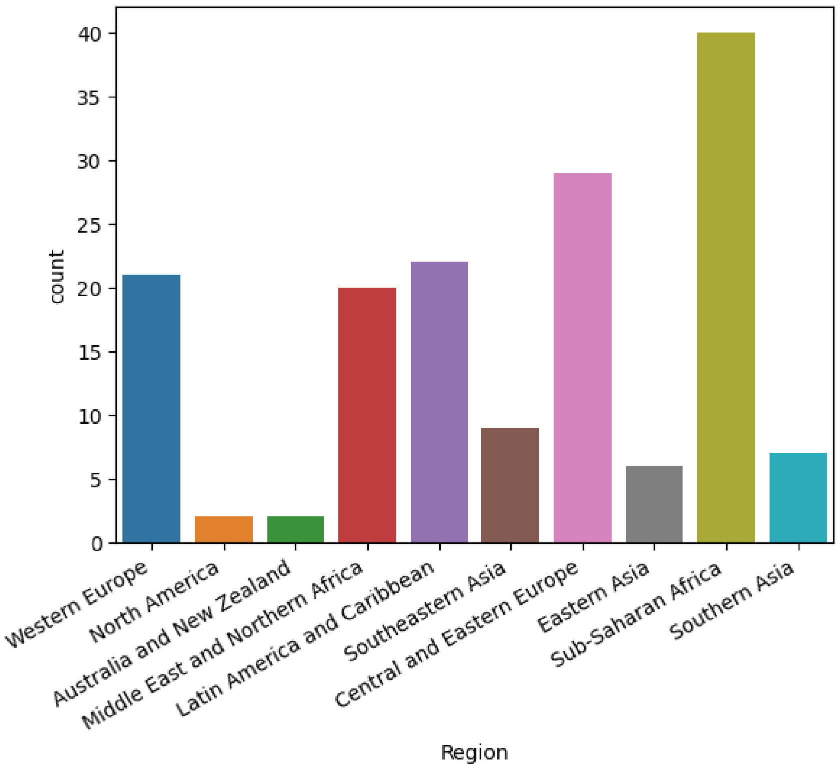
Out[173]:
```
0    Switzerland
1        Iceland
2        Denmark
3         Norway
4         Canada
5        Finland
6    Netherlands
7         Sweden
8    New Zealand
9      Australia
Name: Country, dtype: object
```

In [174]:
```
Bottom_10_Happiest_countries = df['Country'].tail(10)
Bottom_10_Happiest_countries

# here following we can see the BOTTOM 10 HAPPIEST COUNTRIES in the WORLD
```

Out[174]:
```
148            Chad
149          Guinea
150     Ivory Coast
151    Burkina Faso
152     Afghanistan
153          Rwanda
154           Benin
155           Syria
156         Burundi
157            Togo
Name: Country, dtype: object
```

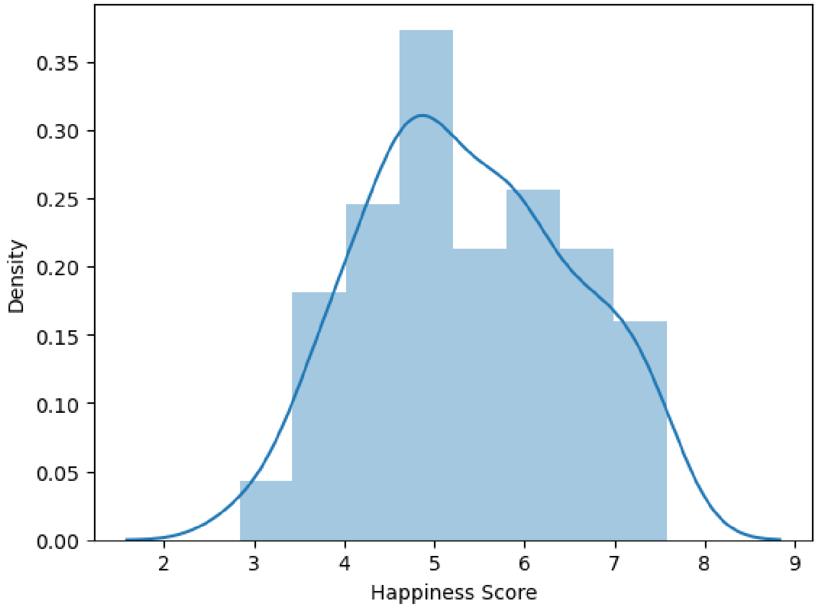
In [50]:
```
sns.countplot(x='Region', data = df)
plt.xticks(rotation=30, ha = 'right')
# here in the following graph we can clearly seen that,
#  Maximum Countries are from = Sub-Saharan Africa
#  Minimum Countries are from = North America
```

Out[50]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'Western Europe'),
  Text(1, 0, 'North America'),
  Text(2, 0, 'Australia and New Zealand'),
  Text(3, 0, 'Middle East and Northern Africa'),
  Text(4, 0, 'Latin America and Caribbean'),
  Text(5, 0, 'Southeastern Asia'),
  Text(6, 0, 'Central and Eastern Europe'),
  Text(7, 0, 'Eastern Asia'),
  Text(8, 0, 'Sub-Saharan Africa'),
  Text(9, 0, 'Southern Asia')])
```



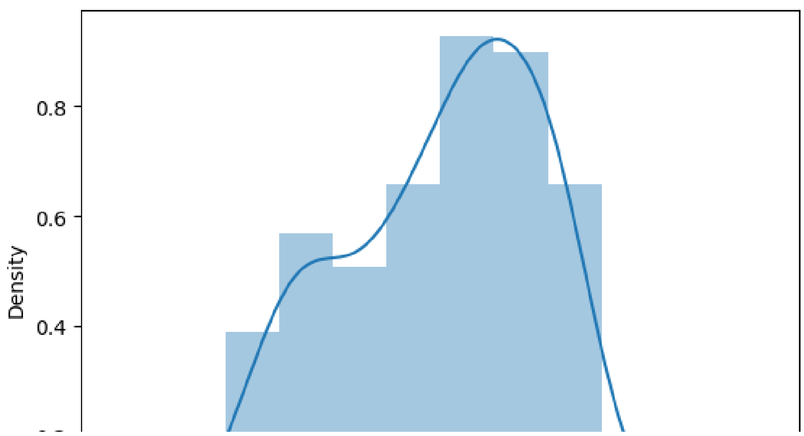
In [70]:
```
sns.distplot(df['Happiness Score'])
#  here we can see the average happiness score distribution is in between [4-6]
#  due to wich we can conclude that most of the countries are having there happiness score in between [4-6]
```

Out[70]: <AxesSubplot:xlabel='Happiness Score', ylabel='Density'>



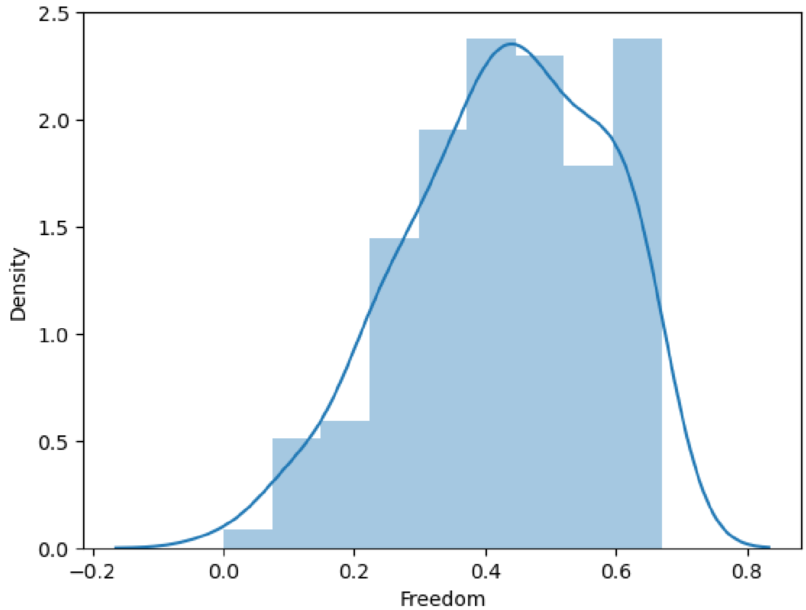
In [72]:
```
sns.distplot(df['Economy (GDP per Capita)'])
# here we can find that the higher density in score of "Economy (GDP per Capita)" in between [.6 - 1.4]
```

Out[72]: <AxesSubplot:xlabel='Economy (GDP per Capita)', ylabel='Density'>



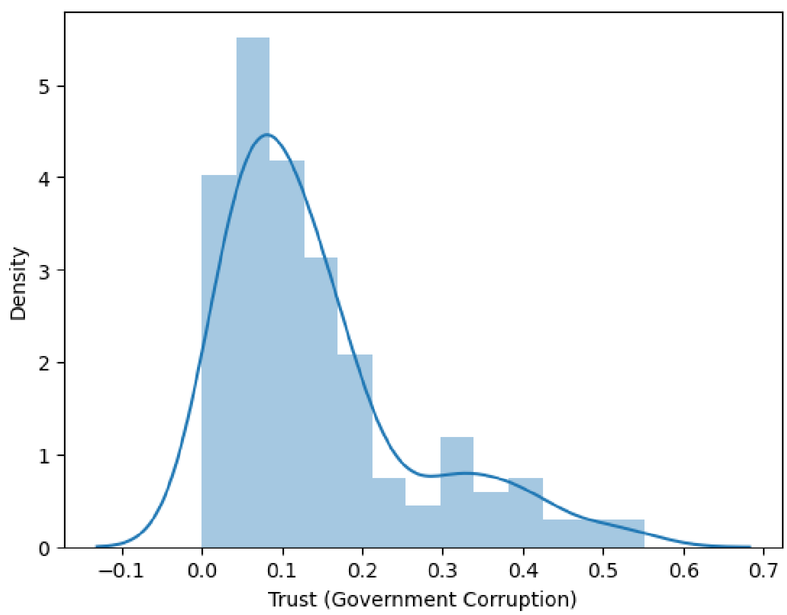
In [73]:
```
sns.distplot(df['Freedom'])
# most of the conutries are having "freedom score " in between [0.3 - 0.7]
```

Out[73]: <AxesSubplot:xlabel='Freedom', ylabel='Density'>



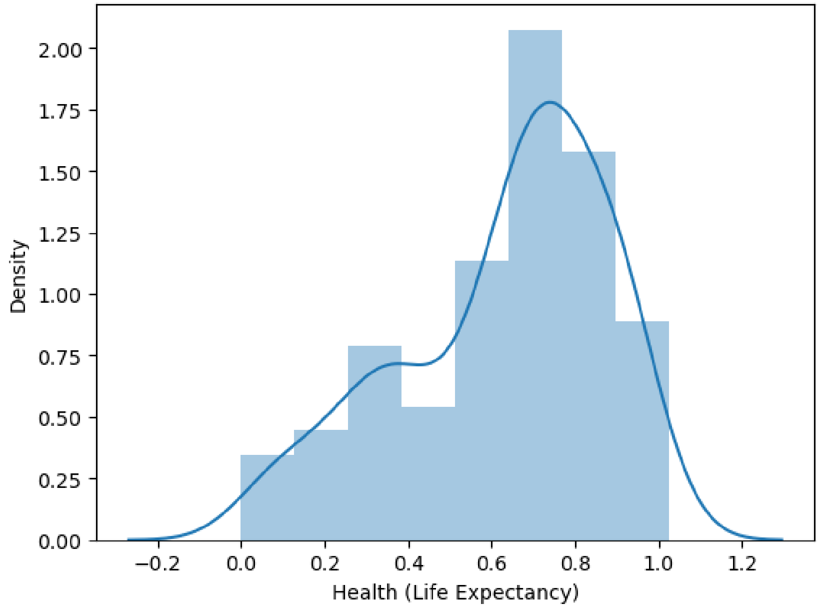
In [74]:
```
sns.distplot(df['Trust (Government Corruption)'])
# here we can see that the data "right skewed" and most of the data is in between [0.0 - 0.2]
```

Out[74]: <AxesSubplot:xlabel='Trust (Government Corruption)', ylabel='Density'>



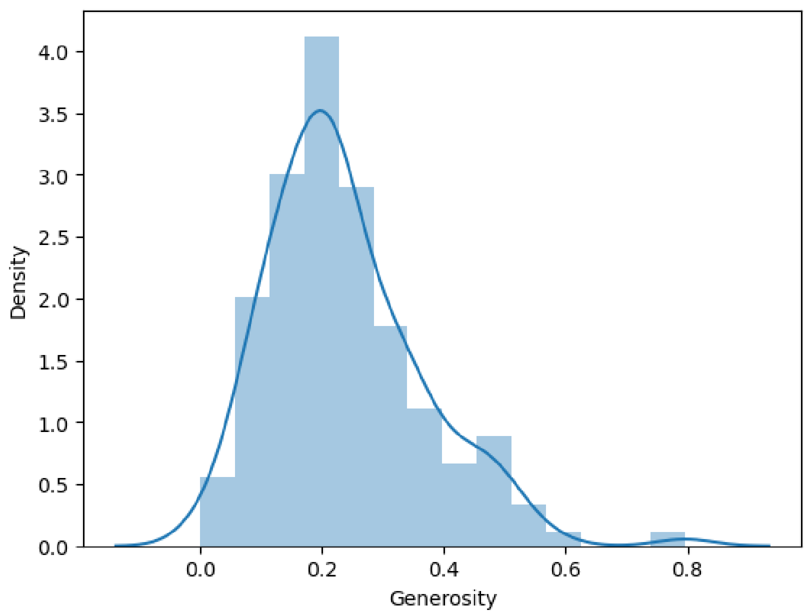
In [77]:
```
sns.distplot(df['Health (Life Expectancy)'])
# here the data is left skewed and
# we can find the healt"life expectency" score having higher density in between [0.6 - 0.9]
```

Out[77]: <AxesSubplot:xlabel='Health (Life Expectancy)', ylabel='Density'>



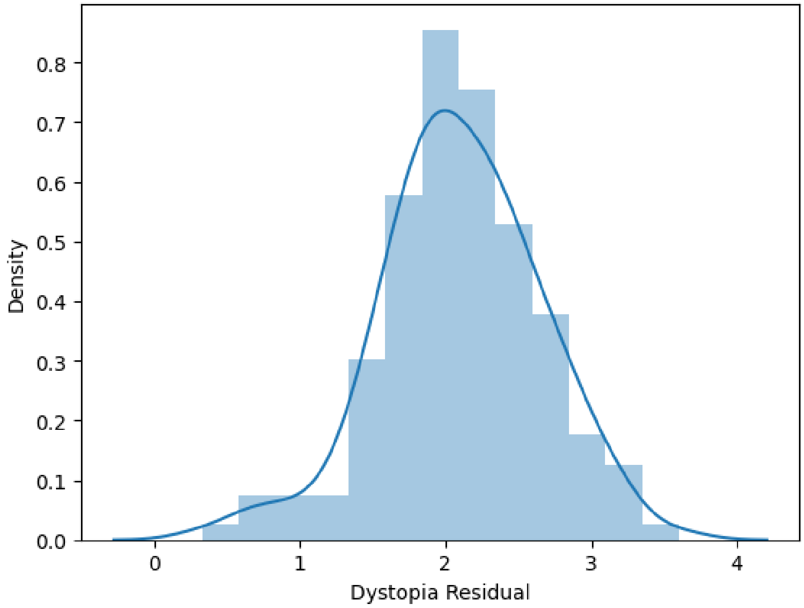
In [78]:
```
sns.distplot(df['Generosity'])
# here we can see there meay be presence of outlier
# the most of the desinty is lying betwwen the [0.1 - 0.3] Generosity score
```

Out[78]: <AxesSubplot:xlabel='Generosity', ylabel='Density'>



In [79]:
```
sns.distplot(df['Dystopia Residual'])
# here we can say that Dystopia Residual score is lying between [1.5 - 2.5]
# the Higher Dysopia is Leat Happines
```

Out[79]: <AxesSubplot:xlabel='Dystopia Residual', ylabel='Density'>

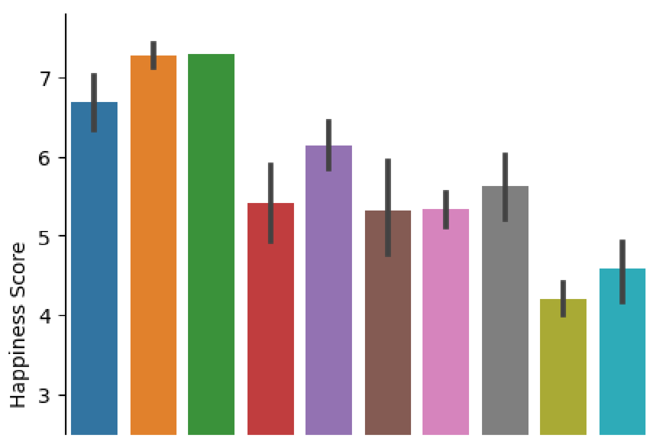


In [ ]:

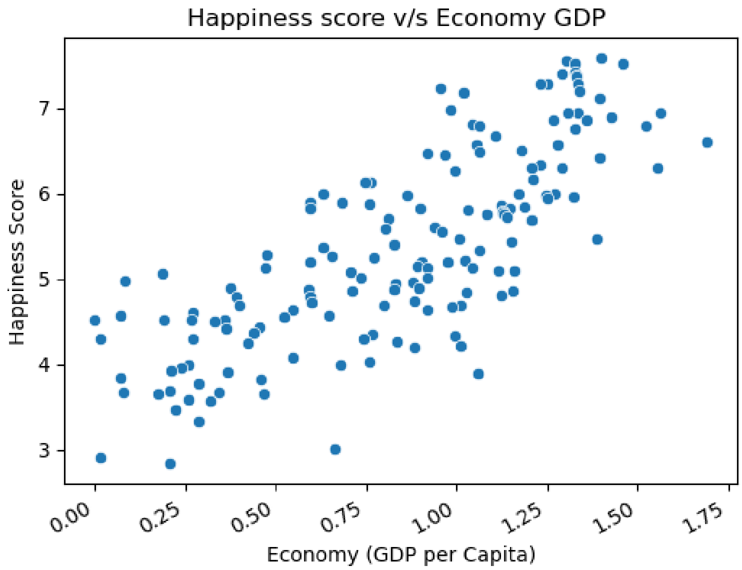
BIVARIATE ANALYSIS
===

In [128]:
```
plt.figure (figsize = (10,6), facecolor = "white")
sns.catplot (x = 'Region', y = 'Happiness Score', data = df, kind = "bar")
plt.xticks(rotation=30, ha = 'right')
plt.show()
# Highest Happiness score is of "NORTH AMERICA", "AUSTRALIA & NEWZEALAND"
# LOWES HAPPINESS SCORE IS FROM "SUB SHAARAN AFRICA"
```

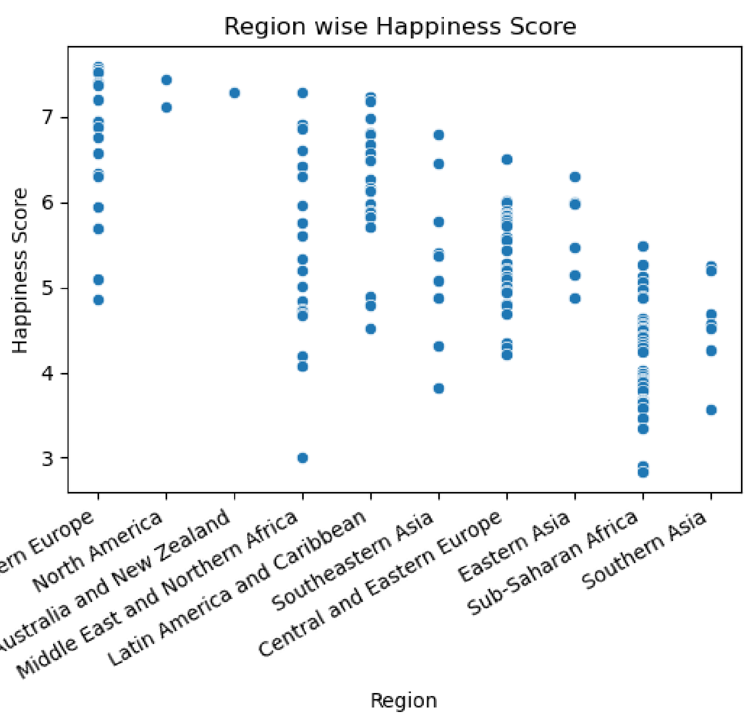
<Figure size 1000x600 with 0 Axes>



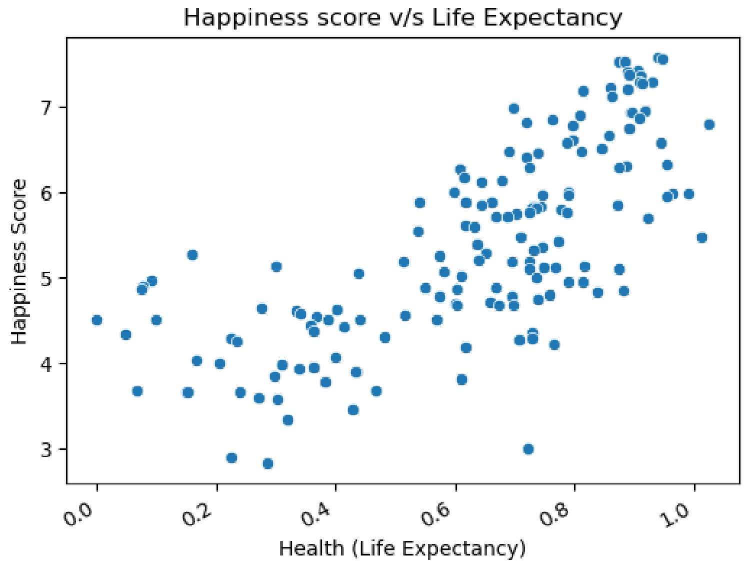
In [129]:
```
plt.figure (figsize = (6,4), facecolor = "white")
plt.title('Happiness score v/s Economy GDP ')
sns.scatterplot (x = 'Economy (GDP per Capita)', y = 'Happiness Score', data = df)
plt.xticks(rotation=30, ha = 'right')
plt.show()

# here finds that "higher the happiness score" is higher to economy gdp
# that mean the Happiness score & GDP is stron positive correlation (directly propertional to each other)
```



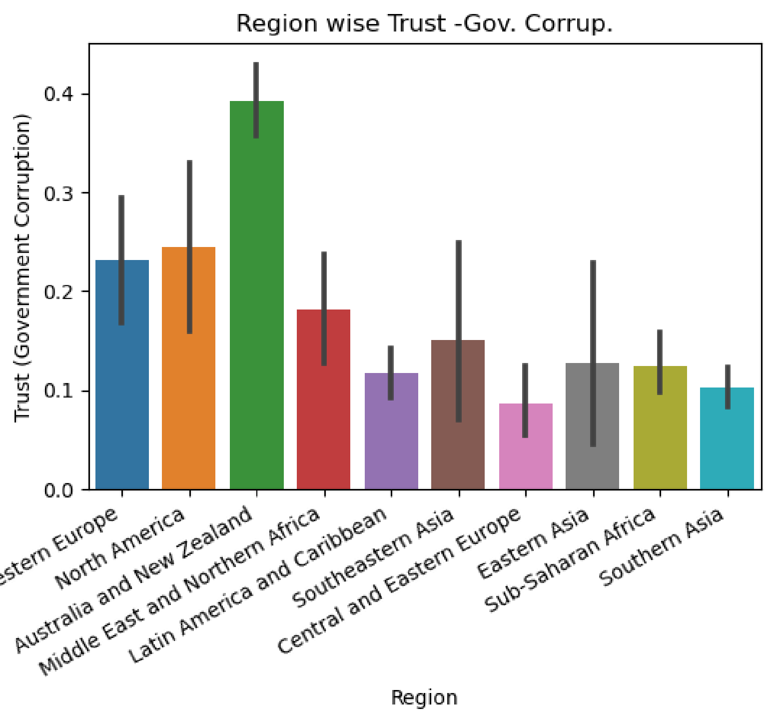
In [130]:
```
plt.figure (figsize = (6,4), facecolor = "white")
plt.title('Region wise Happiness Score ')
sns.scatterplot (x = 'Region', y = 'Happiness Score', data = df)
plt.xticks(rotation=30, ha = 'right')
plt.show()
```



In [131]:
```
plt.figure (figsize = (6,4), facecolor = "white")
plt.title('Happiness score v/s Life Expectancy')
sns.scatterplot (x = 'Health (Life Expectancy)', y = 'Happiness Score', data = df)
plt.xticks(rotation=30, ha = 'right')
plt.show()
# here also strong positive correlation between happiness score and life excpectancy
# that means the country having higher happiness score , the life expectancy of people of that country is also high.
```

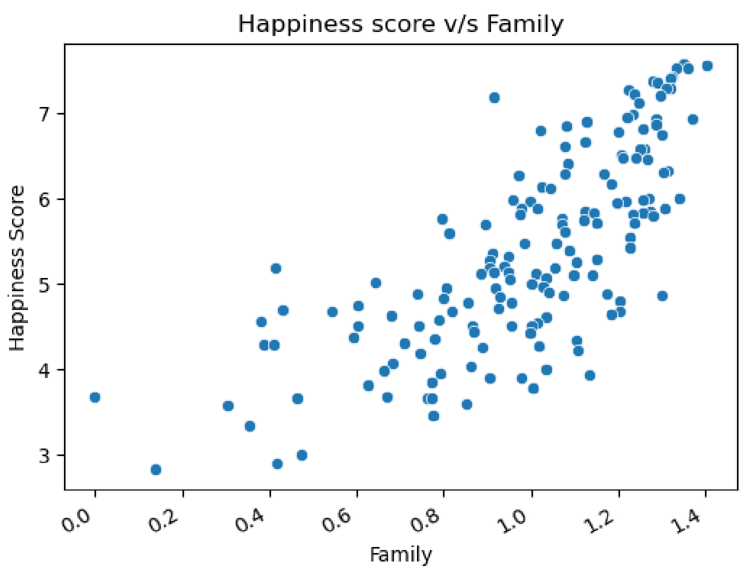


In [132]:
```
plt.figure (figsize = (6,4), facecolor = "white")
plt.title('Region wise Trust -Gov. Corrup.')
sns.barplot (x = 'Region', y = 'Trust (Government Corruption)', data = df)
plt.xticks(rotation=30, ha = 'right')
plt.show()
# australia and newzealand having the highest "trust(government corruption score)"
```

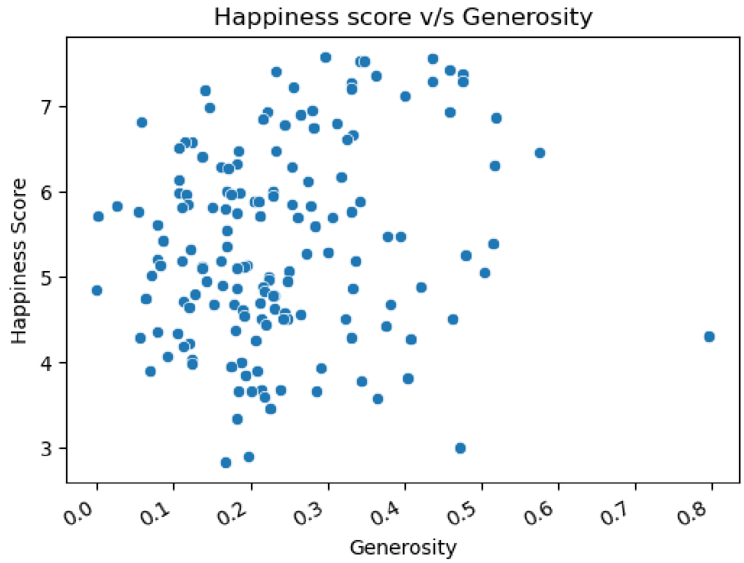


In [133]:
```
plt.figure (figsize = (6,4), facecolor = "white")
plt.title('Happiness Score v/s Freedom')
sns.scatterplot (x = 'Freedom', y = 'Happiness Score', data = df)
plt.xticks(rotation=30, ha = 'right')
plt.show()
# here we can see the strong positive correlation between "happiness score " & "freemdom"
# that means higher the happiness is higher to the Freedom
```

In [134]:
```
plt.figure (figsize = (6,4), facecolor = "white")
plt.title('Happiness score v/s Family')
sns.scatterplot (x = 'Family', y = 'Happiness Score', data = df)
plt.xticks(rotation=30, ha = 'right')
plt.show()
# it is having very strong correlation , higher the happiness score of any country is higher to ist family score.
```

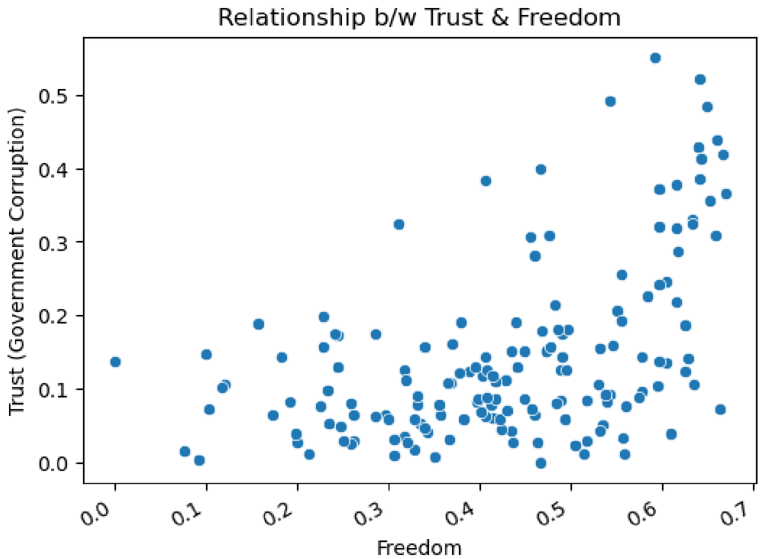


In [135]:
```
plt.figure (figsize = (6,4), facecolor = "white")
plt.title('Happiness score v/s Generosity')
sns.scatterplot (x = 'Generosity', y = 'Happiness Score', data = df)
plt.xticks(rotation=30, ha = 'right')
plt.show()
# here 'generosity' & 'happiness score' not find much relationship or we can say very slightly relationship
```

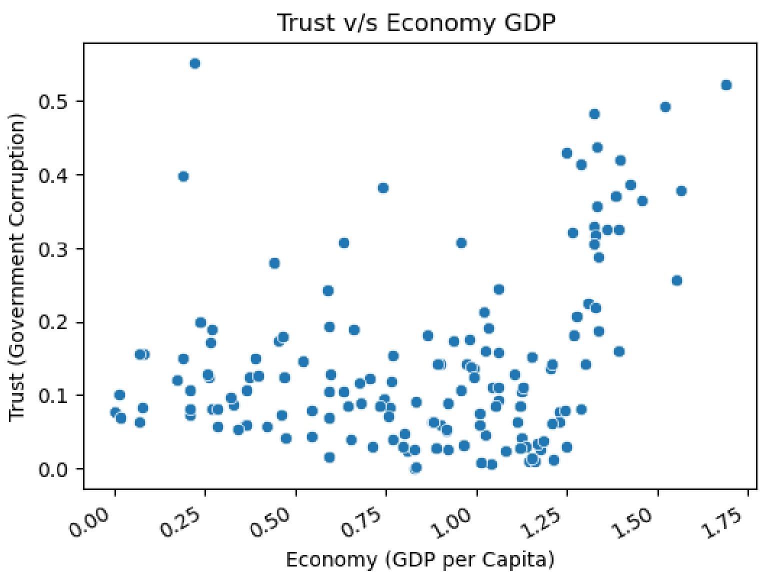


In [136]:
```
plt.figure (figsize = (6,4), facecolor = "white")
plt.title('Happiness score v/s standard error')
sns.scatterplot (x = 'Standard Error', y = 'Happiness Score', data = df)
plt.xticks(rotation=30, ha = 'right')
plt.show()
```

In [137]:
```
plt.figure (figsize = (6,4), facecolor = "white")
plt.title('Relationship b/w Trust & Freedom')
sns.scatterplot (x = 'Freedom', y = 'Trust (Government Corruption)', data = df)
plt.xticks(rotation=30, ha = 'right')
plt.show()
#  here we can say that the higher the freedom of the people higher to the trust score
#  but it is refleting only freedom >0.6
```



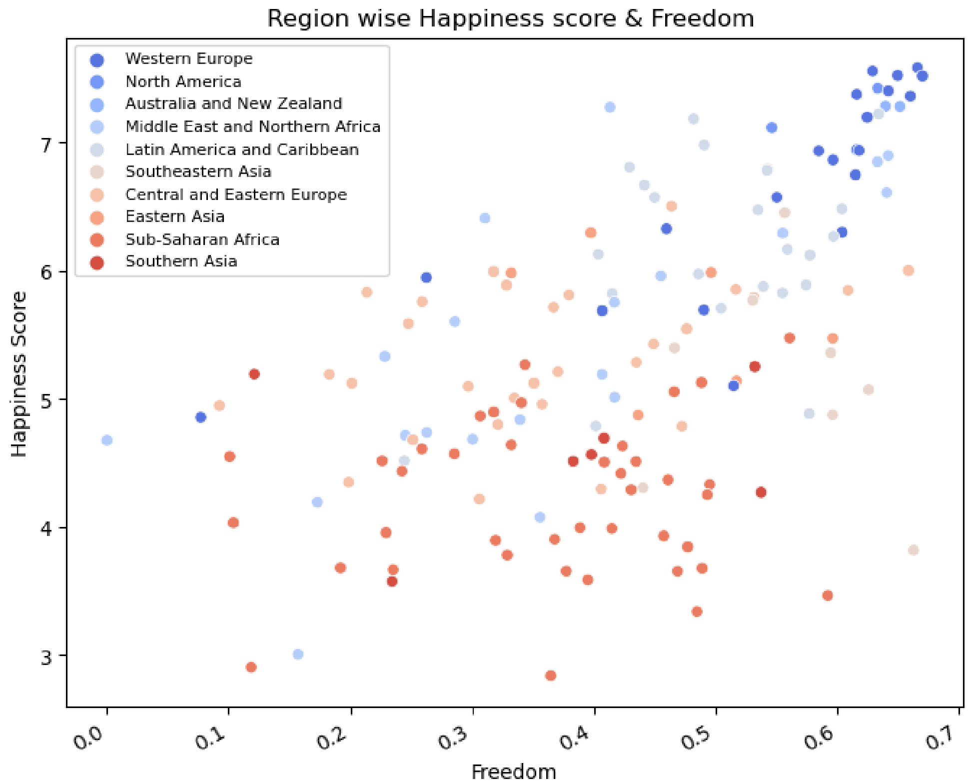
In [126]:
```
plt.figure (figsize = (6,4), facecolor = "white")
plt.title('Trust v/s Economy GDP ')
sns.scatterplot (x = 'Economy (GDP per Capita)', y = 'Trust (Government Corruption)', data = df)
plt.xticks(rotation=30, ha = 'right')
plt.show()
```



In [ ]:

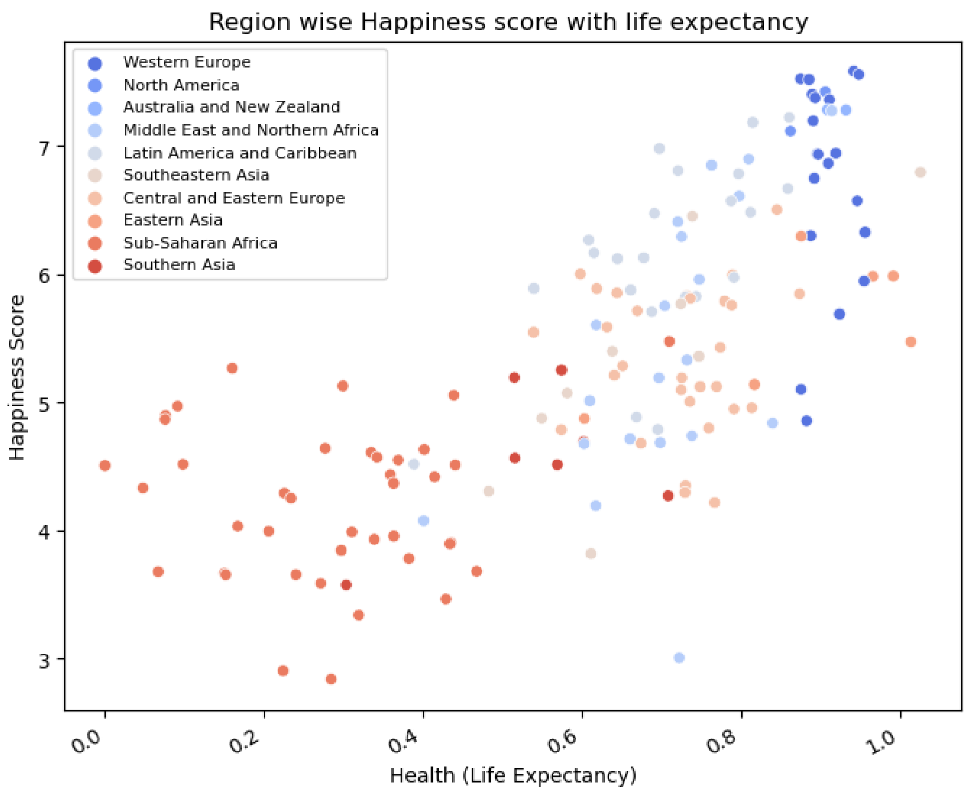
MULTIVARIATE ANALYSIS
==

In [149]:
```
plt.figure (figsize = (8,6), facecolor = "white")
plt.title('Region wise Happiness score & Freedom ')
sns.scatterplot (x= 'Freedom', y = 'Happiness Score', hue = 'Region', data= df, palette = "coolwarm")
plt.xticks(rotation=30, ha = 'right')
plt.legend(loc= 'upper left', fontsize=8)
plt.show()

# here in the below graph we can se the distribution "happiness score" and "freedom " by "region wise"
#  we can clearly see that Highest Happiness & freedom scores are in "western europe, north america & autralia-newzal
```



In [154]:
```
plt.figure (figsize = (8,6), facecolor = "white")
plt.title('Region wise Happiness score with GDP ')
sns.scatterplot (x= 'Economy (GDP per Capita)', y = 'Happiness Score', hue = 'Region', data= df, palette = "coolwarm"
plt.xticks(rotation=30, ha = 'right')
plt.legend(loc='upper left', fontsize=8)
plt.show()

# here we can see the Region wise distribution of Happiness score with GDP
# and we can find that the regions with blue dots are having higher happiness score and also having higher GDP Score
```

In [155]:
```
plt.figure (figsize = (8,6), facecolor = "white")
plt.title('Region wise Happiness score with life expectancy ')
sns.scatterplot (x= 'Health (Life Expectancy)', y = 'Happiness Score', hue = 'Region', data= df, palette = "coolwarm"
plt.xticks(rotation=30, ha = 'right')
plt.legend(loc='upper left', fontsize=8)
plt.show()
# Life Expectancy also having strong positive realtion with happiness score, we can clearly see this below region wise
```



In [ ]:

CHECKING FOR OUTLIERS
==

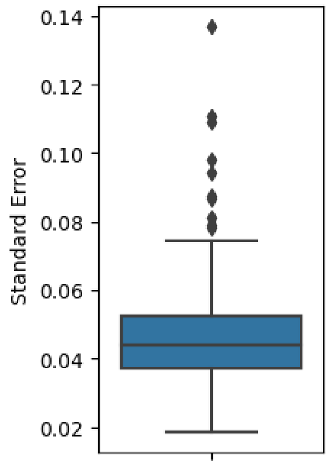
In [194]:
```
df.describe()
```

Out[194]:

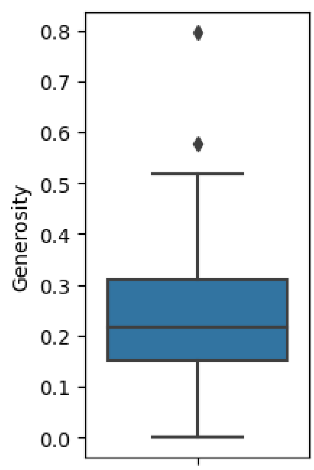
	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	0.143422	0.237296	2.098977
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	0.120034	0.126685	0.553550
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.328580
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	0.061675	0.150553	1.759410
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	0.107220	0.216130	2.095415
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	0.180255	0.309883	2.462415
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	0.551910	0.795880	3.602140

In [ ]:
```
# here as we can see in the above table, we see a huge difference between 75% & Max of some columns,
#  due to which we can assume that there may presence of outliers, so we have to check this with "BOXPLOT METHOD"
```

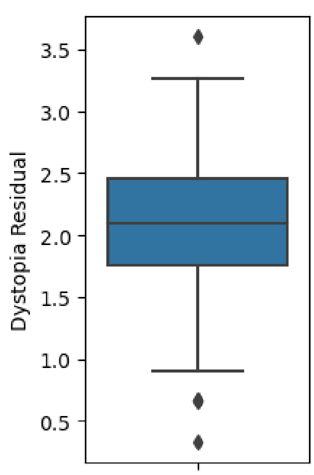
In [193]:
```
plt.figure (figsize = (2,4), facecolor = "white")
sns.boxplot(y='Standard Error',data=df)
plt.show()
# here we can see the presence of outliers.
```



In [196]:
```
plt.figure (figsize = (2,4), facecolor = "white")
sns.boxplot(y='Generosity',data=df)
plt.show()
# also find some outliers "generosity"
```



In [197]:
```
plt.figure (figsize = (2,4), facecolor = "white")
sns.boxplot(y='Dystopia Residual',data=df)
plt.show()
# here might be possibility of outliers
```



In [198]:
```
df.columns
```

Out[198]:
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

In [199]:
```
df_new = df[['Country', 'Region', 'Happiness Rank','Standard Error', 'Economy (GDP per Capita)', 'Family','Health (Li
       'Generosity', 'Dystopia Residual']]
df_new
```

Out[199]:

	Country	Region	Happiness Rank	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	Switzerland	Western Europe	1	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	Iceland	Western Europe	2	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
2	Denmark	Western Europe	3	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
3	Norway	Western Europe	4	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
4	Canada	North America	5	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176
...	...	...	...	...	...	...	...	...	...	...	...
153	Rwanda	Sub-Saharan Africa	154	0.03464	0.22208	0.77370	0.42864	0.59201	0.55191	0.22628	0.67042
154	Benin	Sub-Saharan Africa	155	0.03656	0.28665	0.35386	0.31910	0.48450	0.08010	0.18260	1.63328
155	Syria	Middle East and Northern Africa	156	0.05015	0.66320	0.47489	0.72193	0.15684	0.18906	0.47179	0.32858
156	Burundi	Sub-Saharan Africa	157	0.08658	0.01530	0.41587	0.22396	0.11850	0.10062	0.19727	1.83302
157	Togo	Sub-Saharan Africa	158	0.06727	0.20868	0.13995	0.28443	0.36453	0.10731	0.16681	1.56726

158 rows × 11 columns

In [202]:
```
df_new.shape
```

Out[202]:
```
(158, 11)
```

In [203]:
```
df.shape
```

Out[203]:
```
(158, 12)
```

In [204]:
```
# here as you can see the difference, in [df_new] we are dropping our "target column" i.e = Happiness score
# because we can't filter outliers from our target column therefore first we dropping our target column and make it to
#  DataFrame as df_new, and now we are going to remove outliers from this new dataset, i.e  df_new
```

In [ ]:
```
# As we ideally we can call outliers whose 'z-score value' is less then 3 and more then 3
# so first of all we have to check the z-score and remove the outliers whose z-score is more then 3 an dless then 3.
```

In [207]:
```
df_new.dtypes

# as below you can see in our new dataset all columns are of either 'float64' or 'int64' and 'object'
# so first of all we have to ENCODE out CATEGORICAL COLUMN, 'country' & 'region'
```

Out[207]:
```
Country                          object
Region                           object
Happiness Rank                    int64
Standard Error                  float64
Economy (GDP per Capita)        float64
Family                          float64
Health (Life Expectancy)        float64
Freedom                         float64
Trust (Government Corruption)   float64
Generosity                      float64
Dystopia Residual               float64
dtype: object
```

In [ ]:
```
#  here i think that there NO SUCH RELEVANCE of COLUMN= 'COUNTRY' 'REGION' 'HAPPINESS RANK' in PREDICTING of HAPPINESS
# therefore we should drop those columns ==>>
```

In [227]:
```
df_new1 =  df_new[['Standard Error', 'Economy (GDP per Capita)', 'Family','Health (Life Expectancy)', 'Freedom', 'Tru
       'Generosity', 'Dystopia Residual']]
df_new1.head(5)
```

Out[227]:

	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
2	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
3	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
4	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176

In [221]:
```
df_new1.shape
```

Out[221]: (158, 8)

In [226]:
```
df.shape
```

Out[226]: (158, 12)

In [ ]:
```
# here out of 12 columns, we dropped 3 'country' 'region' 'happiness rank', and 1 is our target column i.e 'Happiness
# so after droping 4 columns out of 12,  8 columns should be remain in our newdataset  'dr_new1'
```

In [316]:
```
df_new1.shape
```

Out[316]: (158, 8)

In [317]:
```
df_new1.columns
```

Out[317]:
```
Index(['Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

In [225]:
```
df_new1.head(2)
```

Out[225]:

	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201

In [ ]:

In [321]:
```
df.columns
```

Out[321]:
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

In [328]:
```
df_new4.shape
```

Out[328]:
```
(149, 9)
```

In [329]:
```
yd = df_new4.iloc[:,0]
yd
```

Out[329]:
```
0      7.587
1      7.561
2      7.527
3      7.522
4      7.427
       ...
150    3.655
151    3.587
152    3.575
154    3.340
156    2.905
Name: Happiness Score, Length: 149, dtype: float64
```

In [ ]:

APPLYING Z-SCORE
===>>>>>>>>>

In [229]:
```
from scipy.stats import zscore
```

In [231]:
```
z = np.abs(zscore(df_new1))
z.head(5)

# by applying 'abs' (absolute method), we are getting
```

Out[231]:

	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	0.805926	1.369621	1.320281	1.263408	1.577438	2.309652	0.471040	0.758258
1	0.055889	1.135226	1.514458	1.289434	1.332456	0.016480	1.575856	1.092857
2	0.854487	1.192861	1.361054	0.992229	1.469659	2.842774	0.824293	0.712335
3	0.531526	1.525130	1.251922	1.035145	1.605131	1.852081	0.868638	0.663893
4	0.722845	1.194876	1.221204	1.118054	1.360416	1.555725	1.748563	0.639337

In [232]:
```
threshold = 3
print(np.where(z>3))
```

```
(array([ 27,  40,  64, 115, 128, 147, 153, 155, 157], dtype=int64), array([5, 0, 0, 0, 6, 2, 5, 7, 2], dtype=int64))
```

In [ ]:
```
# here above we found 9 those values whose z-score is more then > 3
```

In [244]:
```
df_new2 = df_new1[(z<3).all(axis=1)]
df_new2.shape
df_new2
```

Out[244]:

	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
2	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
3	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
4	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176
...	...	...	...	...	...	...	...	...
150	0.05141	0.46534	0.77115	0.15185	0.46866	0.17922	0.20165	1.41723
151	0.04324	0.25812	0.85188	0.27125	0.39493	0.12832	0.21747	1.46494
152	0.03084	0.31982	0.30285	0.30335	0.23414	0.09719	0.36510	1.95210
154	0.03656	0.28665	0.35386	0.31910	0.48450	0.08010	0.18260	1.63328
156	0.08658	0.01530	0.41587	0.22396	0.11850	0.10062	0.19727	1.83302

149 rows × 8 columns

In [245]:
```
df_new1.shape
```

Out[245]: (158, 8)

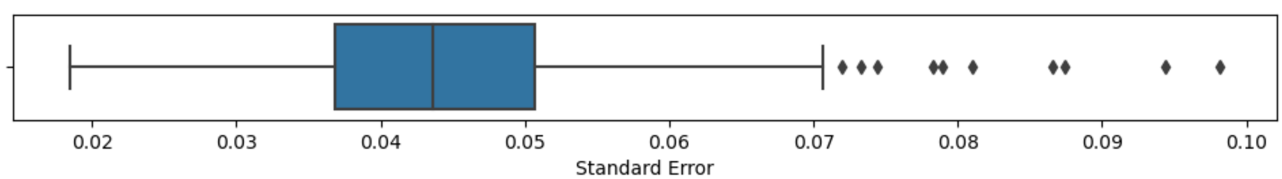
In [246]:
```
df_new2.shape
```

Out[246]: (149, 8)

In [ ]:
```
# here you can see that there is difference of 9,
#  so here we dropped those values whose z-score is >3
```

In [253]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='Standard Error',data=df)
plt.show()

# it is the EARLIER (df dataset) PRESENCE OF OUTLIERS
```

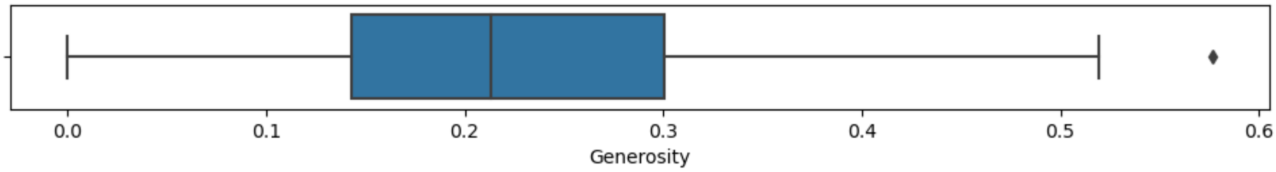
In [252]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='Standard Error',data=df_new2)
plt.show()
# AND THIS IS AFTER APLLYING Z-SCORE , you can clearly see the diffrence between earlier one and this
# in earlier one the presence of OUTLIERS is [.10 - .14]
# but in this case those outliers are removed
```



In [ ]:
```
# Similarly we can check the difference for 'GENEROSITY' & 'Dystopia Residual'
```

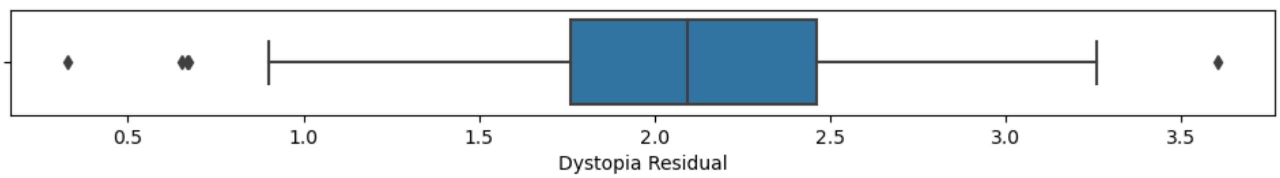
In [254]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='Generosity',data=df)
plt.show()
```

In [255]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='Generosity',data=df_new2)
plt.show()
```

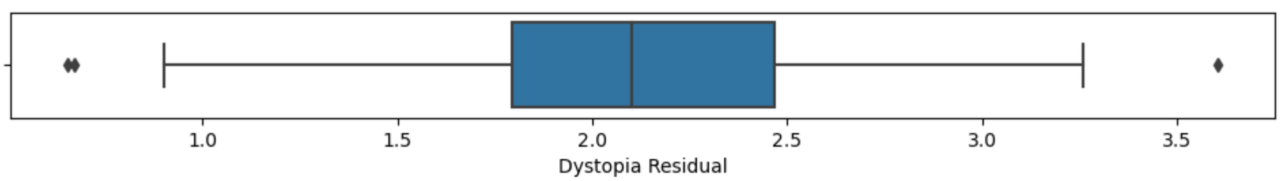


In [ ]:

In [257]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='Dystopia Residual',data=df)
plt.show()
```



In [258]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='Dystopia Residual',data=df_new2)
plt.show()
```



In [ ]:
```
# SUCCESFULLY REMOVED OUTLIERS FROM DATASET
```

In [259]:
```
df_new2.head(2)
```

Out[259]:

	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201

In [ ]:

CHECKING SKEWNESS

===>>>

In [ ]:
```
# the skewness shows the distribution of data, if the data is widely skewed that means it is not good for our model.
#  ideal range of skewness is ( -0.5 to +0.5)
```

In [260]:
```
df_new2.skew()
```

Out[260]:
```
Standard Error                    1.243048
Economy (GDP per Capita)         -0.390657
Family                           -0.811340
Health (Life Expectancy)         -0.747711
Freedom                          -0.400867
Trust (Government Corruption)     1.272530
Generosity                        0.654710
Dystopia Residual                -0.021144
dtype: float64
```

In [261]:
```
# here we can see that the column 'Standard error' & 'Trust' are slightly skewed , so we have to remove that skewness
#  by using 'cuberoot' method.
```

In [266]:
```
df_new2['Standard Error'] = np.cbrt(df_new2['Standard Error'])
df_new2.skew()
# here you can see the difference between skewness present earlier and present in 'Standard'Error' column
```

Out[266]:
```
Standard Error                    0.528395
Economy (GDP per Capita)         -0.390657
Family                           -0.811340
Health (Life Expectancy)         -0.747711
Freedom                          -0.400867
Trust (Government Corruption)     1.272530
Generosity                        0.654710
Dystopia Residual                -0.021144
dtype: float64
```

In [267]:
```
df_new2['Trust (Government Corruption)'] = np.cbrt(df_new2['Trust (Government Corruption)'])
df_new2.skew()

# Similarly in "Trust (Government Corruption)" columns skewness is removed successfully
```

Out[267]:
```
Standard Error                    0.528395
Economy (GDP per Capita)         -0.390657
Family                           -0.811340
Health (Life Expectancy)         -0.747711
Freedom                          -0.400867
Trust (Government Corruption)    -0.064568
Generosity                        0.654710
Dystopia Residual                -0.021144
dtype: float64
```

In [268]:
```
df_new2.shape
# shape is still same bofore removing skewness, there is no such any difference occurs in shape.
```

Out[268]:
```
(149, 8)
```

In [ ]:
```

```

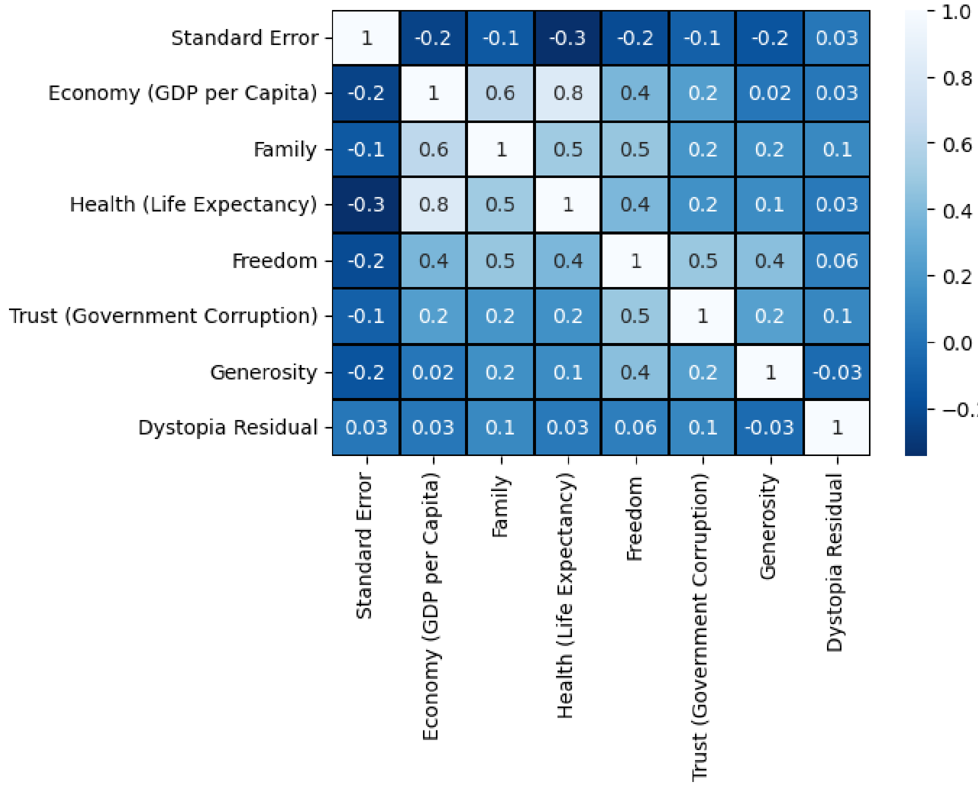
CHECKING CORRELATION (GRAPHICALLY)

===

In [269]:
```
# FINDING CORRELATION GRAPHICALLY BETWEEN INDEPENDENT VARIABLES
```

In [270]:
```
cor = df_new2.corr()
```

In [277]:
```
plt.figure (figsize = (6,4), facecolor = "white")
sns.heatmap(df_new2.corr(),linewidth=0.1,fmt="0.1g",linecolor="black",annot=True,cmap="Blues_r")
plt.yticks(rotation=0);
plt.show()

# here we can see that there is such any correlation in between the variables
#  just a slightly correlation between 'health' & 'GDP'
```



In [ ]:
```
cor['']
```

In [ ]:
```
# FINDING CORRELATION OF WHOLE DATASET (INCLUDING TARGET COLUMN)
```

In [278]:
```
df.head(2)
```

Out[278]:

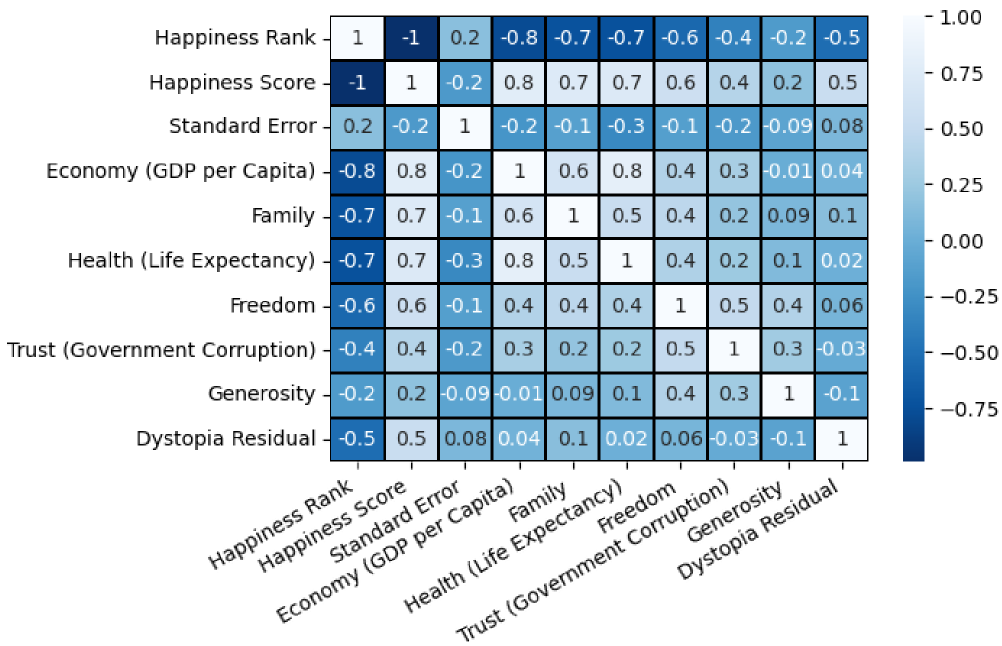
	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201

In [279]:
```
cor1 = df.corr()
cor1
```

Out[279]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
Happiness Rank	1.000000	-0.992105	0.158516	-0.785267	-0.733644	-0.735613	-0.556886	-0.372315	-0.160142	-0.521999
Happiness Score	-0.992105	1.000000	-0.177254	0.780966	0.740605	0.724200	0.568211	0.395199	0.180319	0.530474
Standard Error	0.158516	-0.177254	1.000000	-0.217651	-0.120728	-0.310287	-0.129773	-0.178325	-0.088439	0.083981
Economy (GDP per Capita)	-0.785267	0.780966	-0.217651	1.000000	0.645299	0.816478	0.370300	0.307885	-0.010465	0.040059
Family	-0.733644	0.740605	-0.120728	0.645299	1.000000	0.531104	0.441518	0.205605	0.087513	0.148117
Health (Life Expectancy)	-0.735613	0.724200	-0.310287	0.816478	0.531104	1.000000	0.360477	0.248335	0.108335	0.018979
Freedom	-0.556886	0.568211	-0.129773	0.370300	0.441518	0.360477	1.000000	0.493524	0.373916	0.062783
Trust (Government Corruption)	-0.372315	0.395199	-0.178325	0.307885	0.205605	0.248335	0.493524	1.000000	0.276123	-0.033105
Generosity	-0.160142	0.180319	-0.088439	-0.010465	0.087513	0.108335	0.373916	0.276123	1.000000	-0.101301
Dystopia Residual	-0.521999	0.530474	0.083981	0.040059	0.148117	0.018979	0.062783	-0.033105	-0.101301	1.000000

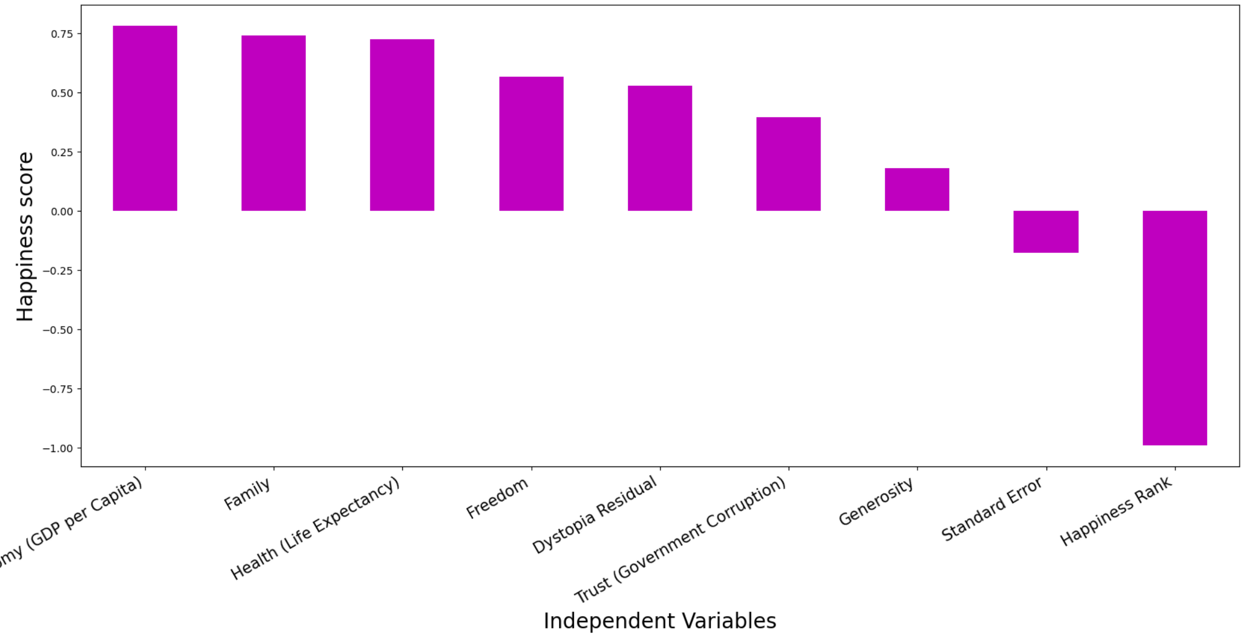
In [281]:
```
plt.figure (figsize = (6,4), facecolor = "white")
sns.heatmap(df.corr(),linewidth=0.1,fmt="0.1g",linecolor="black",annot=True,cmap="Blues_r")
plt.yticks(rotation=0);
plt.xticks(rotation=30,ha='right')
plt.show()
```



In [283]:
```
cor1['Happiness Score'].sort_values(ascending=False)
# here we can see in the earlier dataset (df) the correlation with "Happiness Score"
```

Out[283]:
```
Happiness Score                  1.000000
Economy (GDP per Capita)         0.780966
Family                           0.740605
Health (Life Expectancy)         0.724200
Freedom                          0.568211
Dystopia Residual                0.530474
Trust (Government Corruption)    0.395199
Generosity                       0.180319
Standard Error                  -0.177254
Happiness Rank                  -0.992105
Name: Happiness Score, dtype: float64
```

In [288]:
```
plt.figure(figsize=(20,8))
df.corr()['Happiness Score'].sort_values(ascending=False).drop(['Happiness Score']).plot(kind='bar',color="m")
plt.xlabel('Independent Variables',fontsize=20)
plt.xticks(rotation=30,ha='right',fontsize=15)
plt.ylabel('Happiness score',fontsize =20)
plt.title(("Correlation with Happiness Score")
plt.show()
```



In [289]:
```
df_new2.head(5)
```

Out[289]:

	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	0.324310	1.39651	1.34951	0.94143	0.66557	0.748756	0.29678	2.51738
1	0.365532	1.30232	1.40223	0.94784	0.62877	0.521036	0.43630	2.70201
2	0.321658	1.32548	1.36058	0.87464	0.64938	0.784910	0.34139	2.49204
3	0.338540	1.45900	1.33095	0.88521	0.66973	0.714677	0.34699	2.46531
4	0.328749	1.32629	1.32261	0.90563	0.63297	0.690742	0.45811	2.45176

In [293]:
```
x= df_new2
x.head(5)
```

Out[293]:

	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	0.324310	1.39651	1.34951	0.94143	0.66557	0.748756	0.29678	2.51738
1	0.365532	1.30232	1.40223	0.94784	0.62877	0.521036	0.43630	2.70201
2	0.321658	1.32548	1.36058	0.87464	0.64938	0.784910	0.34139	2.49204
3	0.338540	1.45900	1.33095	0.88521	0.66973	0.714677	0.34699	2.46531
4	0.328749	1.32629	1.32261	0.90563	0.63297	0.690742	0.45811	2.45176

In [294]:
```
x.shape
```

Out[294]: (149, 8)

In [ ]:

APPLYING SCALING TECHNIQUES

==

In [290]:
```
from sklearn.preprocessing import StandardScaler
```

In [291]:
```
st = StandardScaler()
```

In [295]:
```
x = st.fit_transform(x)
x
```

Out[295]:
```
array([[-0.91109675,  1.38191593,  1.35787859, ...,  1.88683529,
         0.54630526,  0.7568764 ],
       [ 0.28953441,  1.13832385,  1.5678818 , ...,  0.27941205,
         1.71389767,  1.10929978],
       [-0.98834276,  1.19821973,  1.40197448, ...,  2.14203323,
         0.91963022,  0.70850719],
       ...,
       [-1.22314025, -1.40259581, -2.81135429, ..., -0.15304962,
         1.11805063, -0.32213507],
       [-0.69014938, -1.48837933, -2.60816264, ..., -0.35566049,
        -0.40922585, -0.9307015 ],
       [ 2.52813796, -2.19013866, -2.36115394, ..., -0.11531156,
        -0.28645792, -0.54943602]])
```

In [296]:
```
xf = pd.DataFrame(data=x)
print(xf)

# here we get our dataset (xf) after applying SCALING TECHING (STANDARD SCALER)
```

```
            0         1         2         3         4         5         6  \
0   -0.911097  1.381916  1.357879  1.235390  1.583704  1.886835  0.546305
1    0.289534  1.138324  1.567882  1.261541  1.338953  0.279412  1.713898
2   -0.988343  1.198220  1.401974  0.962900  1.476027  2.142033  0.919630
3   -0.496623  1.543526  1.283947  1.006023  1.611371  1.646273  0.966495
4   -0.781797  1.200315  1.250726  1.089333  1.366887  1.477326  1.896418
..        ...       ...       ...       ...       ...       ...       ...
144  0.473095 -1.026255 -0.945943 -1.985941  0.274090  0.581308 -0.249803
145 -0.134003 -1.562163 -0.624365 -1.498813 -0.216276  0.161899 -0.117411
146 -1.223140 -1.402596 -2.811354 -1.367851 -1.285662 -0.153050  1.118051
147 -0.690149 -1.488379 -2.608163 -1.303594  0.379439 -0.355660 -0.409226
148  2.528138 -2.190139 -2.361154 -1.691747 -2.054764 -0.115312 -0.286458

            7
0    0.756876
1    1.109300
2    0.708507
3    0.657485
4    0.631620
..        ...
144 -1.343100
145 -1.252030
146 -0.322135
147 -0.930702
148 -0.549436

[149 rows x 8 columns]
```

In [297]:
```
df.columns
```

Out[297]:
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

In [343]:
```
yf = yd
yf
```

Out[343]:
```
0      7.587
1      7.561
2      7.527
3      7.522
4      7.427
       ...
150    3.655
151    3.587
152    3.575
154    3.340
156    2.905
Name: Happiness Score, Length: 149, dtype: float64
```

In [344]:
```
yf.value_counts()
```

Out[344]:
```
5.192    2
7.587    1
4.739    1
4.874    1
4.867    1
        ..
5.889    1
5.878    1
5.855    1
5.848    1
2.905    1
Name: Happiness Score, Length: 148, dtype: int64
```

In [310]:
```
xf.shape
```

Out[310]:
```
(149, 8)
```

In [332]:
```
yf.shape
```

Out[332]:
```
(149,)
```

In [334]:
```
df.columns
```

Out[334]:
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

In [336]:
```
column = ['Standard Error', 'Economy (GDP per Capita)', 'Family','Health (Life Expectancy)', 'Freedom', 'Trust (Govern
```

In [337]:
```
xf.columns = column
```

In [348]:
```
xf.shape
```

Out[348]:
```
(149, 8)
```

In [ ]:

In [345]:
```
column1 = [['Happiness Score']]
yf.columns = column1
```

In [346]:
```
yf.head(5)
```

Out[346]:
```
0    7.587
1    7.561
2    7.527
3    7.522
4    7.427
Name: Happiness Score, dtype: float64
```

In [350]:
```
yf = pd.DataFrame(yf)
yf.head(1)
```

Out[350]:

	Happiness Score
0	7.587

In [351]:
```
yf.shape
```

Out[351]: (149, 1)

In [ ]:

FINDING MULTICOLINEARITY
===

In [ ]:
```
# We have to find the multicollinearity between the features and to remove it we can use VIF (VARIANCE INFLATION FACTO
# we can not apply VIF on the TARGET COLUMN
# for apllyin VIF we have to import some libraries as follows
```

In [352]:
```
import statsmodels.api as sm
from scipy import stats
from statsmodels .stats.outliers_influence import variance_inflation_factor
```

In [353]:
```
# here we are making "def function" for calculating VIF
def calc_vif(xf):
    vif = pd.DataFrame()
    vif["FETURES"] = xf.columns
    vif["VIF FACTOR"] = [variance_inflation_factor(xf.values,i) for i in range (xf.shape[1])]
    return (vif)
```

In [354]:
```
xf.shape
```

Out[354]: (149, 8)

In [355]:
```
calc_vif(xf)
# here we can't find huge multicolinearity between our 'independent columns'
# there is only slightly higher relation b/w 'economy' & 'health'
# but we can't drop any of the column because we already have very few cloumns.
```

Out[355]:

	FETURES	VIF FACTOR
0	Standard Error	1.161693
1	Economy (GDP per Capita)	4.105092
2	Family	1.946484
3	Health (Life Expectancy)	3.417996
4	Freedom	1.930776
5	Trust (Government Corruption)	1.380158
6	Generosity	1.319227
7	Dystopia Residual	1.039449

NO MULTICOLINEARITY FOUND
===

In [ ]:

APPLYING ML MODEL ===

In [362]:
```
# NOW HERE WE CAN SEE THAT OUR TARGET/LABEL COLUMN IN NOT A CATEGORICAL DATA, IT IS HAVING FLOATING DATA,
# AND WHEN WE ARE HAVING "Y" (TARGET) IN DECIMAL FORM THEN WE CAN APPLY "REGRESSION MODEL",
# SO HERE WE CAN APPLY REGRESSION MODEL ON OUR DATASET TO PREDICT, "HAPPINESS SCORE".
```

In [366]:
```
from sklearn.linear_model import LinearRegression
```

In [367]:
```
lr = LinearRegression()
```

In [369]:
```
from sklearn.model_selection import train_test_split
```

In [373]:
```
 x_train,x_test,y_train,y_test = train_test_split(xf,yf,test_size=0.20,random_state=42)
```

In [374]:
```
lr.fit(x_train,y_train)
y_pred = lr.predict(x_test)
y_test.head(),y_pred[0:4]
```

Out[374]:
```
(     Happiness Score
 76             5.286
 18             6.937
 121            4.512
 81             5.192
 79             5.212,
 array([[5.2850254 ],
        [6.96339574],
        [4.51558998],
        [5.21588221]]))
```

In [375]:
```
#  here above we can see the similarity between "actual values" and "predicted values"
```

In [377]:
```
from sklearn.metrics import mean_squared_error
```

In [378]:
```
mean_squared_error (y_test,y_pred)
```

Out[378]:
```
0.0009146300422051524
```

In [ ]:
```
# as we can see the mean squared error is very low that means our model working very good.
```

In [385]:
```
from sklearn.metrics import r2_score
```

In [386]:
```
r2_score(y_test,y_pred)
```

Out[386]:
```
0.9991983352687518
```

In [ ]:
```
# r2 score is also very high.
```

In [387]:
```
xf.shape
```

Out[387]:
```
(149, 8)
```

In [391]:
```
def pred_func(q):
    q= q.reshape(1,8)
    qt = lr.predict(q)
    print(qt)

# making 'def' function to predict HAPPINESS SCORE of any given value
```

In [392]:
```
q= np.array([0.911097,1.381916,1.357879,1.235390,1.583704,1.886835,0.546305,0.756876])
pred_func(q)

# here we are giving values to the model, and the model is predicting the HAPPINESS SCORE of the given country.
```

```
[[7.51941471]]
```

In [ ]:

SAVING THE MODEL
==

In [395]:
```
import pickle
```

In [397]:
```
file_name = 'happiness final model.pkl'
pickle.dump(lr,open(file_name,'wb'))
```

In [ ]: