SUMESH R -20104169

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:

```
df = pd.read_csv("2_2015.csv")
df
```

Out[2]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297
...	...	...	...	...	...	...	...	...	...
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59201
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36453

158 rows × 12 columns

In [3]:

```
df.head()
```

Out[3]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(Go C
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	

Data cleaning and pre processing

In [4]:

```
df.info()
```

```
<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 [5]:

```
df.describe()
```

Out[5]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trus (Governmer Corruptior
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.00000
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	0.14342
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	0.12003

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trus (Governmer Corruptior
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	0.00000
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	0.06167
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	0.10722
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	0.18025
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	0.55191

In [6]:

```
df.columns
```

Out[6]:

```
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')
```

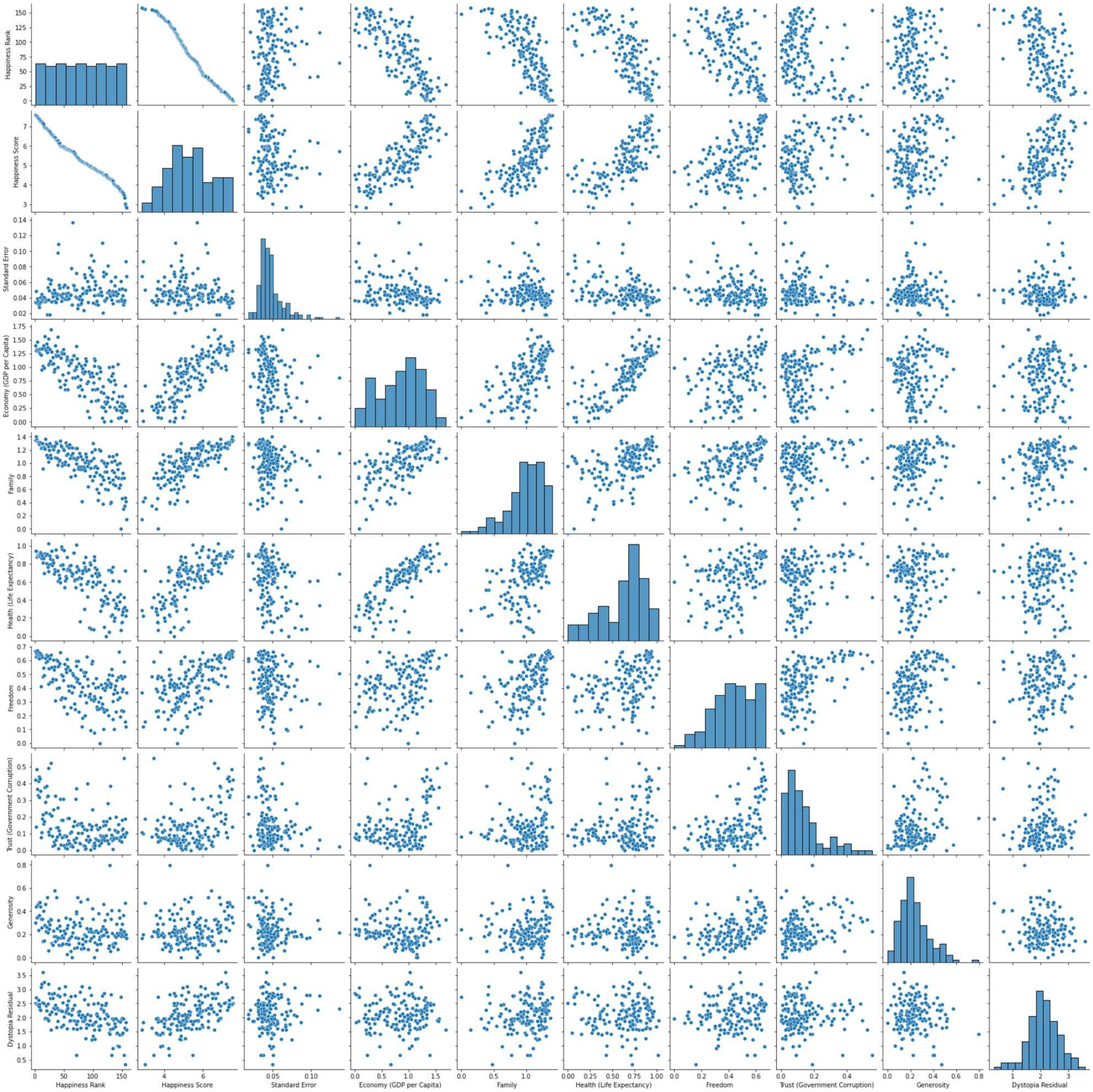
EDA and VISUALIZATION

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x25a846cd7f0>
```



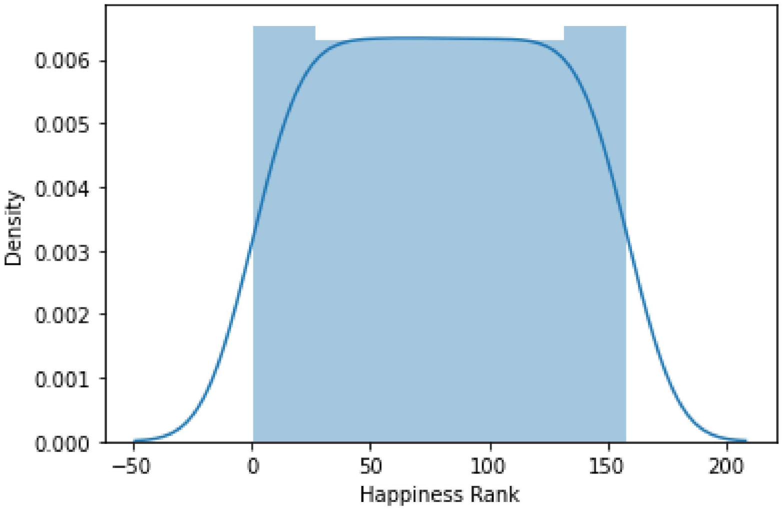
In [8]:

```
sns.distplot(df["Happiness Rank"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:

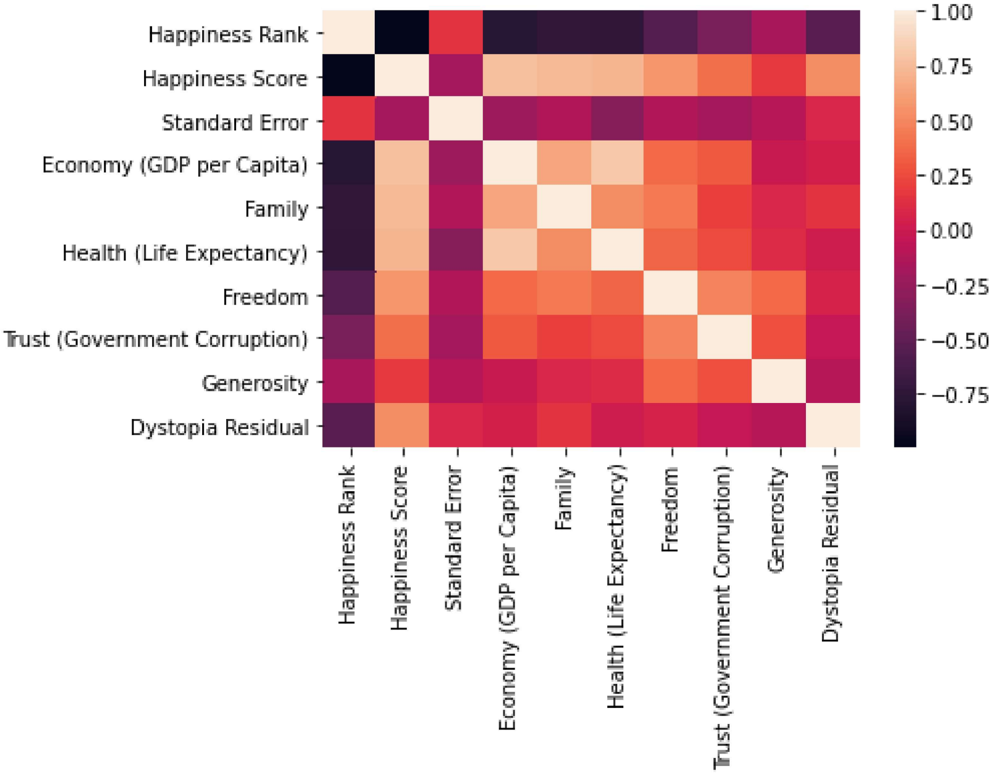
```
<AxesSubplot:xlabel='Happiness Rank', ylabel='Density'>
```



In [9]:
```
df1 = df[['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual']]
```

In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]:
```
<AxesSubplot:>
```



In [11]:

```
x = df1[['Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual']]
y = df1['Happiness Rank']
```

split the data into training and test data

In [12]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

In [13]:

```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[13]:

```
LinearRegression()
```

In [14]:

```
lr.intercept_
```

Out[14]:

```
295.83061870931977
```

In [15]:

```
coeff = pd.DataFrame(lr.coef_, x.columns, columns =['Co-efficient'])
coeff
```

Out[15]:

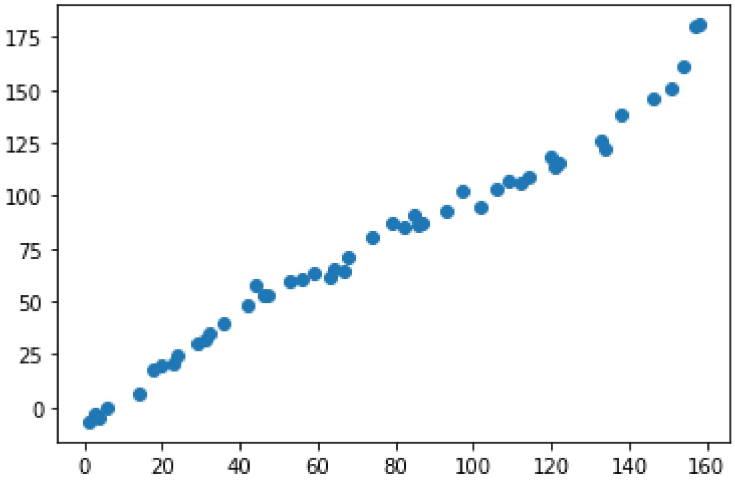
	Co-efficient
Happiness Score	2335.328912
Standard Error	-31.457479
Economy (GDP per Capita)	-2376.542365
Family	-2373.181887
Health (Life Expectancy)	-2381.355277
Freedom	-2375.097262
Trust (Government Corruption)	-2366.036909
Generosity	-2370.170485
Dystopia Residual	-2375.072240

In [16]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x25a89945910>
```



In [17]:

```
lr.score(x_test,y_test)
```

Out[17]:

```
0.9763443018997207
```