

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\2015 - 2015.csv")
a
```

Out[2]:

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

158 rows × 12 columns



In [3]:

```
b=a.head(100)
b
```

Out[3]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.941
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.947
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.874
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.885
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.905
...
95	Bosnia and Herzegovina	Central and Eastern Europe	96	4.949	0.06913	0.83223	0.91916	0.790
96	Lesotho	Sub-Saharan Africa	97	4.898	0.09438	0.37545	1.04103	0.076
97	Dominican Republic	Latin America and Caribbean	98	4.885	0.07446	0.89537	1.17202	0.668
98	Laos	Southeastern Asia	99	4.876	0.06698	0.59066	0.73803	0.549
99	Mongolia	Eastern Asia	100	4.874	0.03313	0.82819	1.30060	0.602

100 rows × 12 columns



In [4]:

b.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 12 columns):
 #   Column                                Non-Null Count  Dtype
---  -
 0   Country                             100 non-null    object
 1   Region                             100 non-null    object
 2   Happiness Rank                      100 non-null    int64
 3   Happiness Score                    100 non-null    float64
 4   Standard Error                    100 non-null    float64
 5   Economy (GDP per Capita)          100 non-null    float64
 6   Family                            100 non-null    float64
 7   Health (Life Expectancy)          100 non-null    float64
 8   Freedom                            100 non-null    float64
 9   Trust (Government Corruption)      100 non-null    float64
10   Generosity                        100 non-null    float64
11   Dystopia Residual                  100 non-null    float64
dtypes: float64(9), int64(1), object(2)
memory usage: 9.5+ KB
```

In [5]:

b.describe()

Out[5]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
mean	50.490000	6.06081	0.047206	1.045210	1.119594	0.747963	0.480267
std	29.000347	0.79900	0.017788	0.299610	0.175886	0.175114	0.135930
min	1.000000	4.87400	0.018480	0.083080	0.414110	0.076120	0.092450
25%	25.750000	5.35300	0.037135	0.875007	1.007810	0.666432	0.401975
50%	50.500000	5.91900	0.042650	1.073035	1.140595	0.755560	0.500285
75%	75.250000	6.75900	0.052268	1.272500	1.258182	0.885710	0.596122
max	100.000000	7.58700	0.136930	1.690420	1.402230	1.025250	0.669730

In [6]:

b.columns

Out[6]:

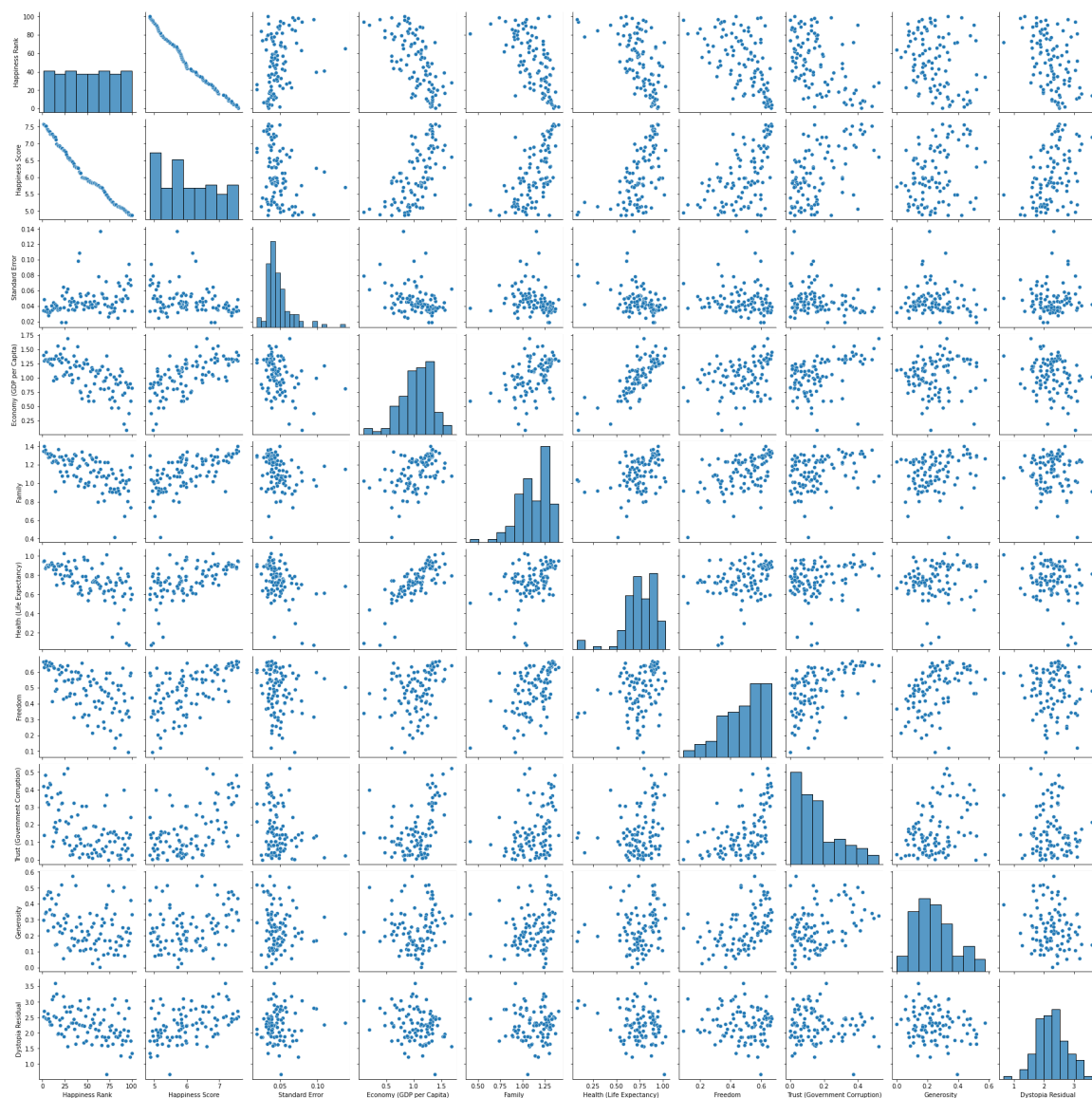
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
      'Standard Error', 'Economy (GDP per Capita)', 'Family',
      'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
      'Generosity', 'Dystopia Residual'],
      dtype='object')
```

In [7]:

```
sns.pairplot(b)
```

Out[7]:

<seaborn.axisgrid.PairGrid at 0x20594948850>



In [8]:

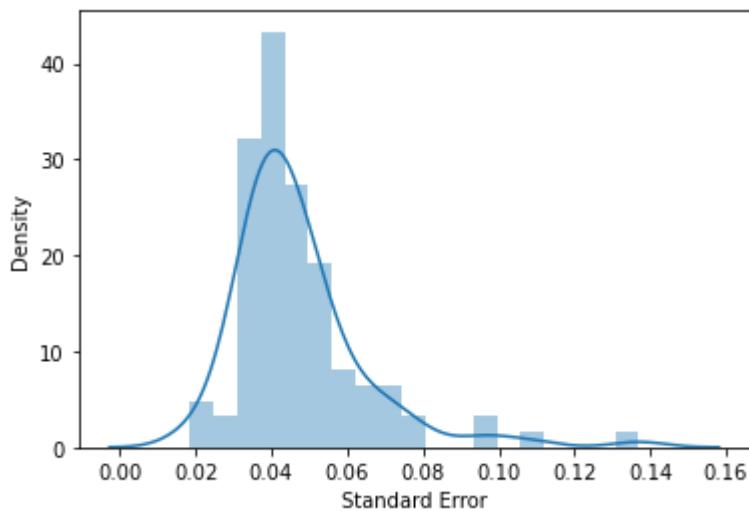
```
sns.distplot(b['Standard Error'])
```

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 adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).

```
warnings.warn(msg, FutureWarning)
```

Out[8]:

<AxesSubplot:xlabel='Standard Error', ylabel='Density'>



In [9]:

```
f=b[['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(f.corr())
```

Out[10]:

<AxesSubplot:>



In [13]:

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

In [14]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

In [15]:

```
from sklearn.linear_model import LinearRegression

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

Out[15]:

LinearRegression()

In [16]:

```
print(lr.intercept_)
```

-0.029172851465184522

In [17]:

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

Out[17]:

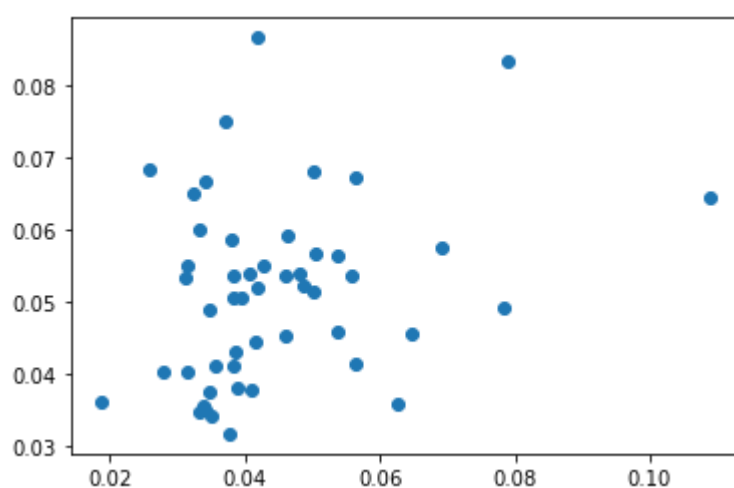
	Co-efficient
Happiness Rank	0.000493
Happiness Score	11.729629
Economy (GDP per Capita)	-11.711744
Family	-11.729818
Health (Life Expectancy)	-11.762069
Freedom	-11.693634
Trust (Government Corruption)	-11.746788
Generosity	-11.701080
Dystopia Residual	-11.712232

In [18]:

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

Out[18]:

<matplotlib.collections.PathCollection at 0x2059b570580>



In [19]:

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

-0.3853138914298673