In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df=pd.read_csv("/Users/bob/Downloads/2_2015.csv")
df.fillna(0,inplace=True)
df
```

Out[3]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Lif Expectancy
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.9414
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.9478
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.8746
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.8852
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.9056
...	...	...	...	...	...	...	...	.
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.4286
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.3191
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.7219
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.2239
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.2844

158 rows × 12 columns

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

Out[4]:

	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

In [5]:
```
df.info()
```

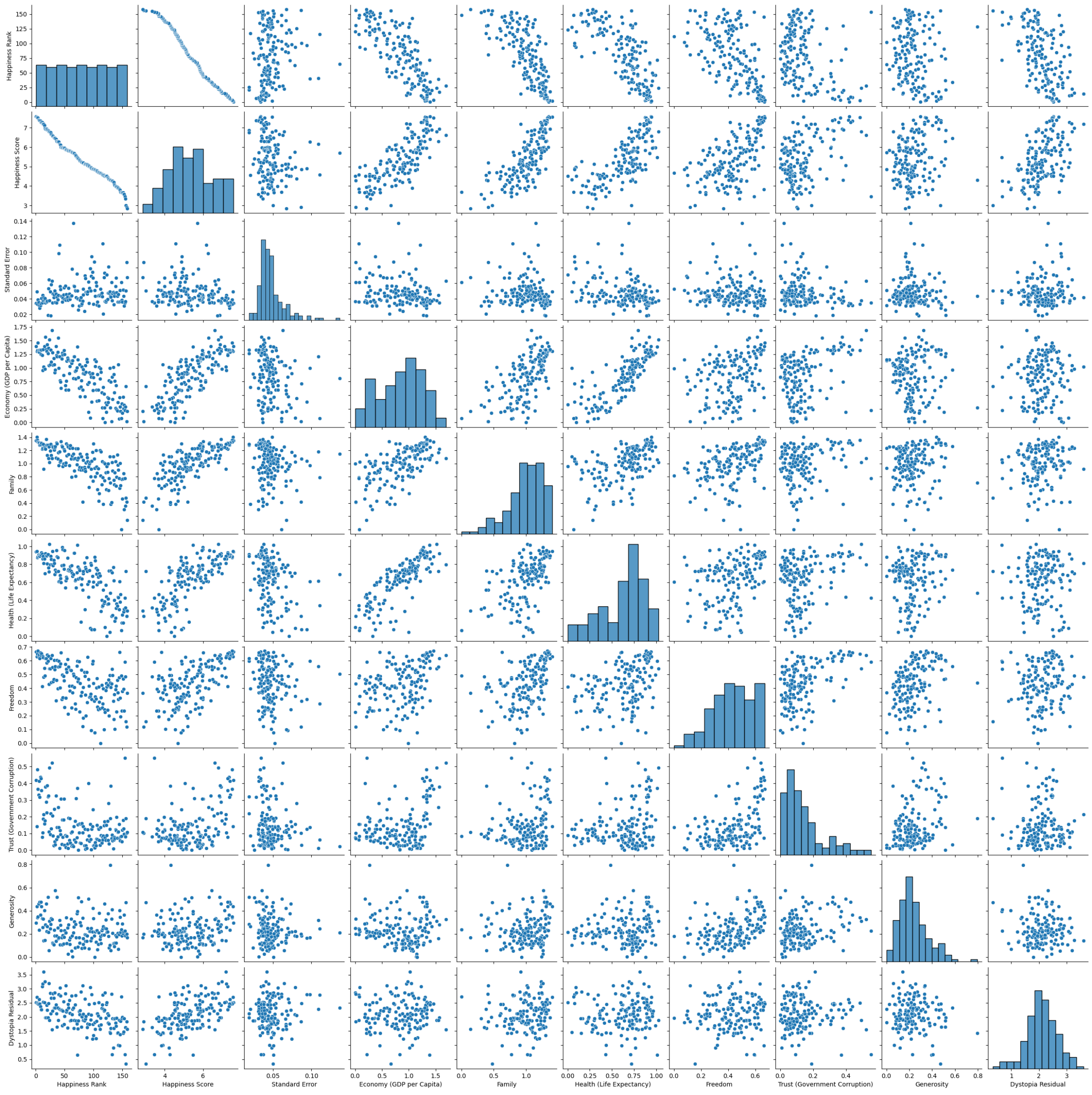
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 12 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Country                        158 non-null    object 
 1   Region                         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 [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 C
orruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

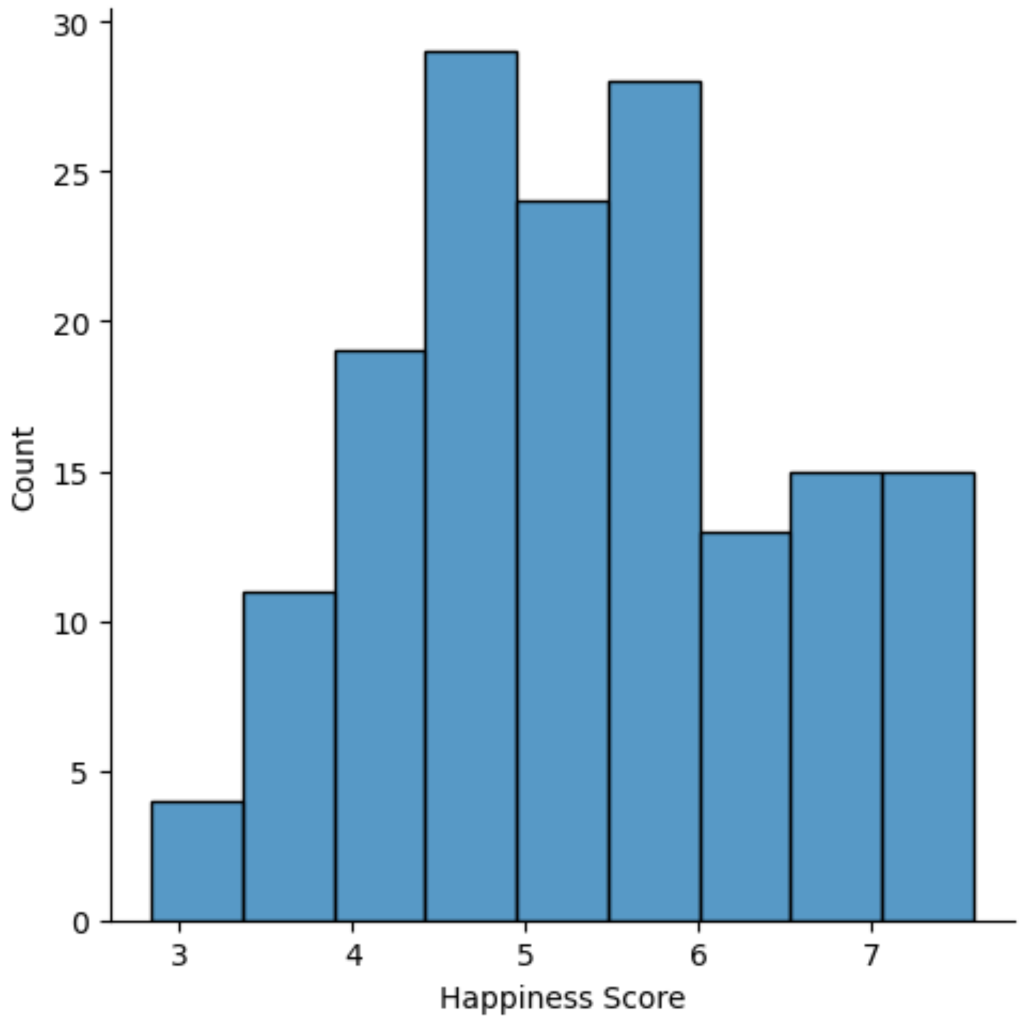
```
In [7]: sns.pairplot(df)
```

```
Out[7]: <seaborn.axisgrid.PairGrid at 0x7ff26b0e8220>
```



In [8]:
```
sns.displot(df['Happiness Score'])
```

Out[8]:
```
<seaborn.axisgrid.FacetGrid at 0x7ff240f11390>
```



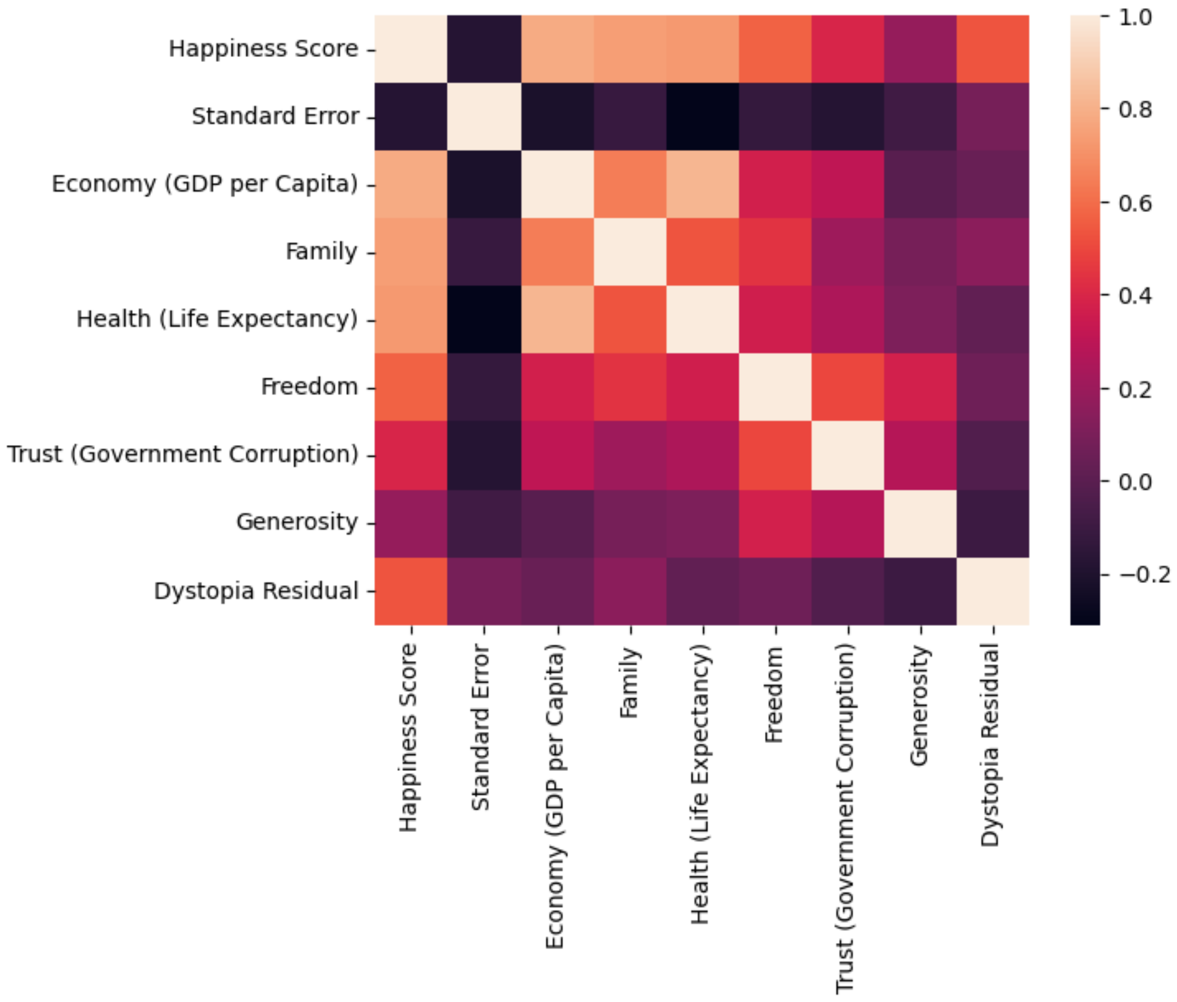
In [9]:
```
df1=df.drop(['Happiness Rank'],axis=1)
df1
df1=df1.drop(df1.index[1537:])
df1.isna().sum()
```

Out[9]:
```
Country                          0
Region                           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 [10]:
```
sns.heatmap(df1.corr())
```

```
/var/folders/2n/rrl24lws3pb1nz8_t911srvm0000gn/T/ipykernel_15173/7
81785195.py:1: FutureWarning: The default value of numeric_only in
DataFrame.corr is deprecated. In a future version, it will default
to False. Select only valid columns or specify the value of numeri
c_only to silence this warning.
  sns.heatmap(df1.corr())
```

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



In [11]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [12]:
```
df1.isna().sum()
```

Out[12]:
```
Country                          0
Region                           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 [17]:
```
y=df1['Happiness Score']
x=df1.drop(['Happiness Score','Country','Region'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train)
```

```
     Standard Error  Economy (GDP per Capita)   Family  \
29          0.04612                   1.05351  1.24823
116         0.02043                   0.64499  0.38174
143         0.03602                   0.06940  0.77265
36          0.04206                   1.20740  1.30203
47          0.04528                   0.86402  0.99903
..              ...                       ...      ...
90          0.06161                   0.18847  0.95152
57          0.04615                   0.90019  0.97459
71          0.05051                   1.38604  1.05818
12          0.03751                   1.33723  1.29704
7           0.03157                   1.33171  1.28907

     Health (Life Expectancy)  Freedom  Trust (Government Corrupti
on)  \
29                    0.78723  0.44974                        0.08
484
116                   0.51529  0.39786                        0.08
492
143                   0.29707  0.47692                        0.15
639
36                    0.88721  0.60365                        0.13
586
47                    0.79075  0.48574                        0.18
090
..                        ...      ...
...
90                    0.43873  0.46582                        0.39
928
57                    0.73017  0.41496                        0.05
989
71                    1.01328  0.59608                        0.37
124
```

```
12                     0.89042  0.62433                        0.18
676
7                      0.91087  0.65980                        0.43
844

     Generosity  Dystopia Residual
29      0.11451            2.83600
116     0.26475            2.27513
143     0.19387            1.87877
36      0.51752            1.64880
47      0.11541            2.53942
..          ...                ...
90      0.50318            2.11032
57      0.14982            2.59450
71      0.39478            0.65429
12      0.33088            2.53320
7       0.36262            2.37119

[110 rows x 8 columns]
```

In [18]:
```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

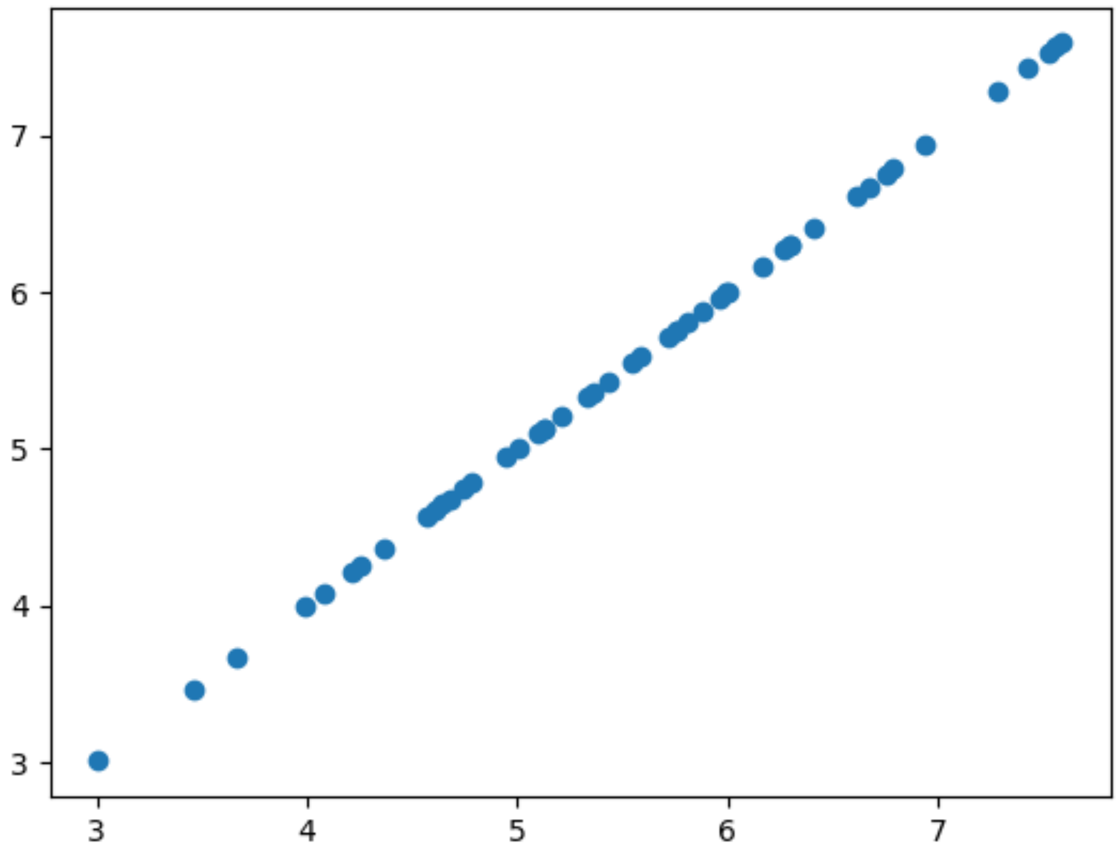
Out[18]:
```
6.179620002999542e-05
```

In [19]:
```
model.coef_
```

Out[19]:
```
array([0.00114922, 1.00010333, 0.99994382, 0.99990246, 0.99962586,
       0.9999552 , 1.00019083, 1.00001032])
```

In [20]:
```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[20]:
```
<matplotlib.collections.PathCollection at 0x7ff26daedcf0>
```



In [21]:
```
model.score(x_test,y_test)
```

Out[21]:
```
0.9999999328772439
```

In [ ]: