In [1]:

```
import pandas as pd
df=pd.read_csv('happyness_score_dataset.csv')
print (df)
```

```
         Country                           Region  Happiness Rank  \
0    Switzerland                   Western Europe               1   
1        Iceland                   Western Europe               2   
2        Denmark                   Western Europe               3   
3         Norway                   Western Europe               4   
4         Canada                    North America               5   
..           ...                              ...             ...   
153       Rwanda               Sub-Saharan Africa             154   
154        Benin               Sub-Saharan Africa             155   
155        Syria  Middle East and Northern Africa             156   
156      Burundi               Sub-Saharan Africa             157   
157         Togo               Sub-Saharan Africa             158   

     Happiness Score  Standard Error  Economy (GDP per Capita)   Family  \
0              7.587         0.03411                   1.39651  1.34951   
1              7.561         0.04884                   1.30232  1.40223   
2              7.527         0.03328                   1.32548  1.36058   
3              7.522         0.03880                   1.45900  1.33095   
4              7.427         0.03553                   1.32629  1.32261   
..               ...             ...                       ...      ...   
153            3.465         0.03464                   0.22208  0.77370   
154            3.340         0.03656                   0.28665  0.35386   
155            3.006         0.05015                   0.66320  0.47489   
156            2.905         0.08658                   0.01530  0.41587   
157            2.839         0.06727                   0.20868  0.13995   

     Health (Life Expectancy)  Freedom  Trust (Government Corruption)  \
0                     0.94143  0.66557                        0.41978   
1                     0.94784  0.62877                        0.14145   
2                     0.87464  0.64938                        0.48357   
3                     0.88521  0.66973                        0.36503   
4                     0.90563  0.63297                        0.32957   
..                        ...      ...                            ...   
153                   0.42864  0.59201                        0.55191   
154                   0.31910  0.48450                        0.08010   
155                   0.72193  0.15684                        0.18906   
156                   0.22396  0.11850                        0.10062   
157                   0.28443  0.36453                        0.10731   

     Generosity  Dystopia Residual  
0       0.29678            2.51738  
1       0.43630            2.70201  
2       0.34139            2.49204  
3       0.34699            2.46531  
4       0.45811            2.45176  
..          ...                ...  
153     0.22628            0.67042  
154     0.18260            1.63328  
155     0.47179            0.32858  
156     0.19727            1.83302  
157     0.16681            1.56726  

[158 rows x 12 columns]
```

In [2]:

```
df.shape
```

Out[2]:

```
(158, 12)
```

key observation-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	Trust (Government Corruption)	Generosity	Dystop Residu
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.517
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.702
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.492
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.465
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.451

In [4]:

```
import numpy as np
df.isnull().sum()
```

Out[4]:

```
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
```

key observation- no missing value present

In [5]:

```
df.describe()
```

Out[5]:

	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

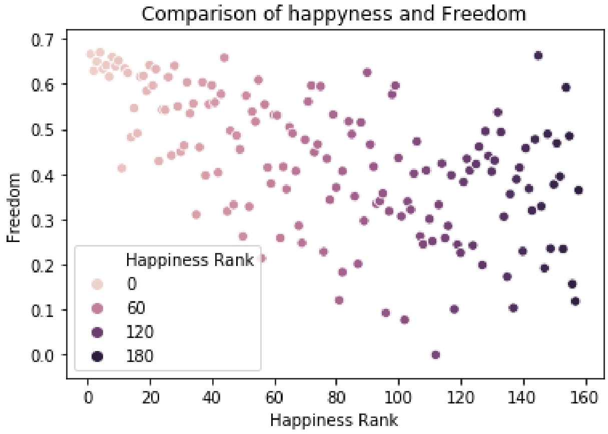
key observation- mean and median values are allmost same so data is normally distributed

In [7]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
plt.title('Comparison of happyness and Freedom')
sns.scatterplot(df['Happiness Rank'], df['Freedom'],hue = df['Happiness Rank'],s = 40);
plt.show
```

Out[7]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```



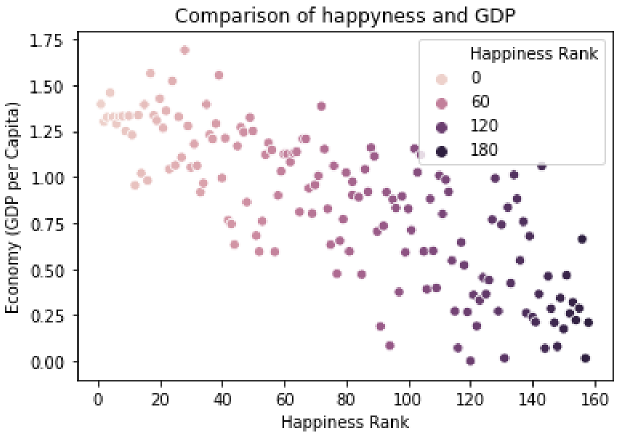
key observation-happyness rank increases with decrease in freedom

In [8]:

```
plt.title('Comparison of happyness and GDP')
sns.scatterplot(df['Happiness Rank'], df['Economy (GDP per Capita)'],hue = df['Happiness Rank'],s
= 40);
plt.show
```

Out[8]:

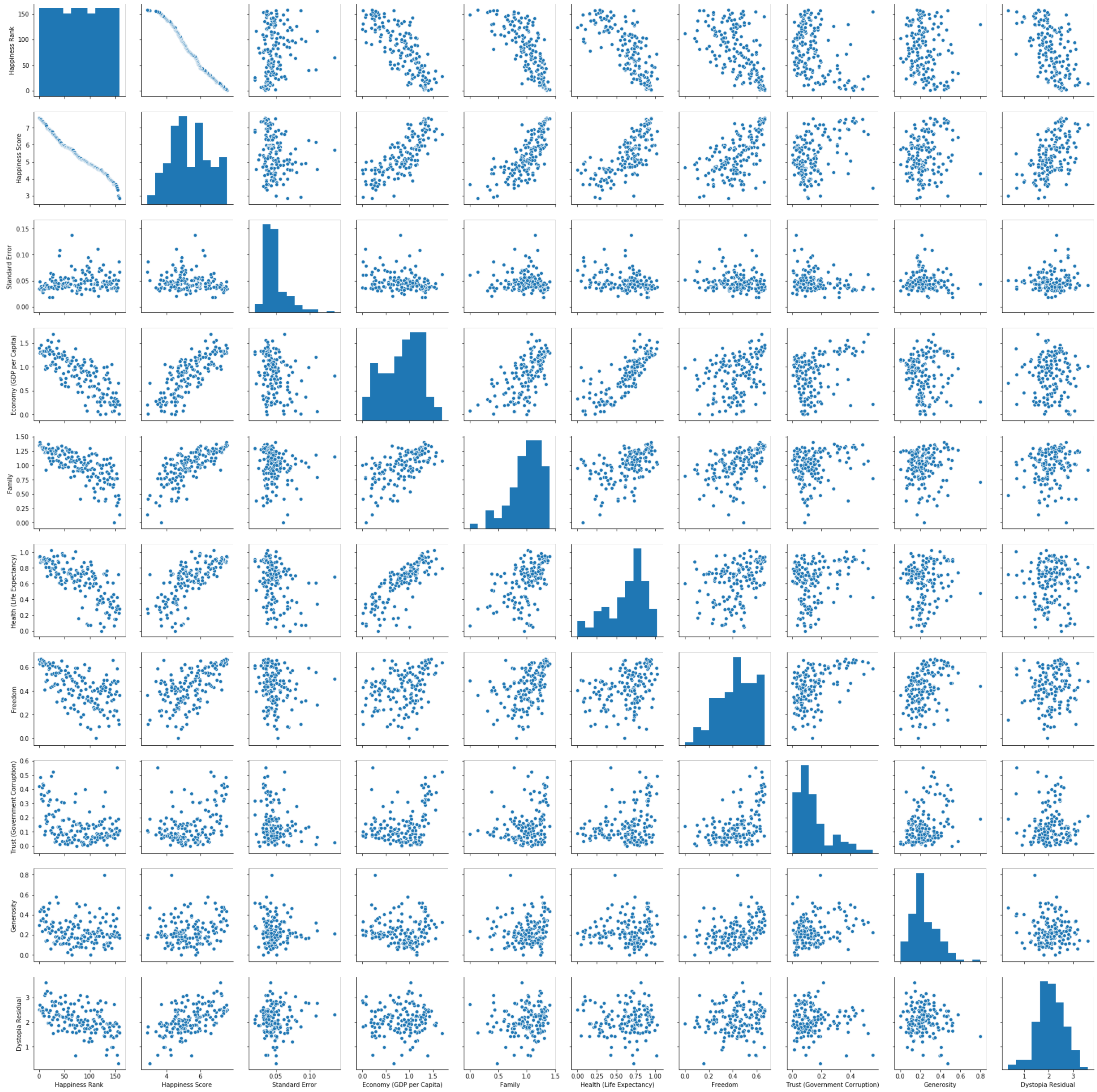
```
<function matplotlib.pyplot.show(*args, **kw)>
```



key observation-happyness rank increases with decraese in gdp key observation-both gdp and freedom has direct influence on happiness

In [10]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
df=pd.read_csv('happyness_score_dataset.csv')
sns.pairplot(df);
plt.show()
```



observation- This pairs plot compares the importance of each of the six factors of happiness to each of the others. If there is a strong positive linear correlation between two factors, we can say that if one factor is important in evaluating a country's overall happiness, it is likely that the other factor is important as well. Based on the plots, it seems that the importances of Economy & Health are strongly correlated, as well as Economy & Family.

In [11]:

```
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
df["Country"]=LE.fit_transform(df["Country"])
```

In [12]:

```
df["Country"].value_counts()
```

Out[12]:

```
157    1
49     1
56     1
55     1
54     1
      ..
104    1
```

```
103    1
102    1
101    1
0      1
Name: Country, Length: 158, dtype: int64
```

In [13]:

```
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
df["Region"]=LE.fit_transform(df["Region"])
```

In [14]:

```
df["Region"].value_counts()
```

Out[14]:

```
8    40
1    29
3    22
9    21
4    20
6     9
7     7
2     6
5     2
0     2
Name: Region, dtype: int64
```

In [15]:

```
import numpy as np
from scipy.stats import zscore
z=np.abs(zscore(df))
z
```

Out[15]:

```
array([[1.23877001, 1.30025593, 1.72099989, ..., 2.30965159, 0.47103971,
        0.75825809],
       [0.44946522, 1.30025593, 1.69907456, ..., 0.01647953, 1.57585637,
        1.09285682],
       [0.90989302, 1.30025593, 1.67714922, ..., 2.8427738 , 0.8242928 ,
        0.71233526],
       ...,
       [1.26069514, 0.37544095, 1.67742676, ..., 0.38141902, 1.85689094,
        3.20843049],
       [1.26069514, 0.96511655, 1.69935209, ..., 0.35771452, 0.31694987,
        0.48198451],
       [1.37032081, 0.96511655, 1.72127743, ..., 0.30180313, 0.5581534 ,
        0.96361241]])
```

In [16]:

```
threshold=3
print(np.where(z>3))
```

```
(array([ 27,  40,  64, 115, 128, 147, 153, 155, 157], dtype=int64), array([ 9,  4,  4,  4, 10,  6,
9, 11,  6], dtype=int64))
```

In [17]:

```
df_new=df[(z<3).all(axis=1)]
```

In [18]:

```
df_new
```

Out[18]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopi Residua
0	135	9	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.5173
1	58	9	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.7020
2	37	9	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.4920
3	105	9	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.4653
4	24	5	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.4517
...	...	...	...	...	...	...	...	...	...	...	...	.
150	66	8	151	3.655	0.05141	0.46534	0.77115	0.15185	0.46866	0.17922	0.20165	1.4172
151	20	8	152	3.587	0.04324	0.25812	0.85188	0.27125	0.39493	0.12832	0.21747	1.4649
152	0	7	153	3.575	0.03084	0.31982	0.30285	0.30335	0.23414	0.09719	0.36510	1.9521
154	13	8	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450	0.08010	0.18260	1.6332
156	21	8	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850	0.10062	0.19727	1.8330

149 rows × 12 columns

In [19]:

```
df.shape
```

Out[19]:

```
(158, 12)
```

In [20]:

```
df_new.shape
```

Out[20]:

```
(149, 12)
```

key observation-outlier part removed

In [21]:

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

In [22]:

```
x=df_new.iloc[:,0:-1]
x.head()
```

Out[22]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity
0	135	9	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678
1	58	9	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630
2	37	9	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139
3	105	9	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699
4	24	5	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811

In [23]:

```
y=df_new.iloc[:,-1]
y.head()
```

Out[23]:

```
0    2.51738
1    2.70201
2    2.49204
3    2.46531
4    2.45176
Name: Dystopia Residual, dtype: float64
```

In [24]:

```
x.shape
```

Out[24]:

```
(149, 11)
```

In [25]:

```
y.shape
```

Out[25]:

```
(149,)
```

In [26]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
```

In [27]:

```
x_train.shape
```

Out[27]:

```
(99, 11)
```

In [28]:

```
y_train.shape
```

Out[28]:

```
(99,)
```

In [29]:

```
x_test.shape
```

Out[29]:

```
(50, 11)
```

In [30]:

```
y_test.shape
```

Out[30]:

```
(50,)
```

In [32]:

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

Out[32]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [33]:

```
pred=lm.predict(x_test)
print("Predicted result price:",pred)
print("actual price",y_test)
```

```
Predicted result price: [2.2323283  2.41453596 2.44850926 1.87961004 2.00059658 2.85762902
 2.43237004 1.38101077 0.65434309 1.93112358 1.71923632 2.53346244
 2.32309996 2.2664063  2.2470872  2.13126096 1.44419931 1.58813412
 2.59442833 1.94922031 2.6777479  1.46155546 2.67619979 1.95068457
 0.89935078 1.41703743 2.30873719 1.84415815 2.53915921 2.32135248
 3.19124171 3.26044896 2.76613993 3.17770201 2.24627578 2.51767202
 2.24700331 2.21125703 2.45188595 2.31987671 1.78529703 1.62249643
 1.75334038 1.9694091  3.08896815 2.63449008 1.7931085  1.94262322
 1.63800623 1.9529531 ]
actual price 76     2.23270
18     2.41484
121    2.44876
81     1.87996
79     2.00073
32     2.85737
67     2.43209
145    1.38079
71     0.65429
85     1.93129
112    1.71956
12     2.53320
37     2.32323
9      2.26646
19     2.24743
58     2.13090
141    1.44395
72     1.58782
57     2.59450
136    1.94939
30     2.67782
127    1.46181
26     2.67585
132    1.95071
133    0.89991
150    1.41723
113    2.30919
104    1.84408
47     2.53942
31     2.32142
22     3.19131
15     3.26001
68     2.76579
11     3.17728
44     2.24639
108    2.51767
53     2.24729
28     2.21126
4      2.45176
33     2.31945
124    1.78555
88     1.62215
89     1.75360
16     1.96961
10     3.08854
84     2.63430
137    1.79293
148    1.94296
78     1.63794
111    1.95335
Name: Dystopia Residual, dtype: float64
```

Name: Dystopia Residual, dtype: float64

In [34]:

```
from sklearn.metrics import r2_score
print(r2_score(y_test,pred))
```

0.9999997486068977

KEY OBSERVATION-very less difference between the actual value and the predicted value so linear regression works as best model

In [ ]:

```

```

In [ ]:

```

```