In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
df=pd.read_csv('world_happiness.csv')
```

In [4]:

```
df.columns
```

Out[4]:

```
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 [5]:

```
df.head()
```

Out[5]:

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

checking the null values:

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
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
```

There are no null values

checking data types:

In [7]:
```
df.dtypes
```

Out[7]:
```
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
```

The data types are okay to proceed with

In [10]:
```
df['Happiness Score'].unique()
```

Out[10]:
```
array([7.587, 7.561, 7.527, 7.522, 7.427, 7.406, 7.378, 7.364, 7.286,
       7.284, 7.278, 7.226, 7.2  , 7.187, 7.119, 6.983, 6.946, 6.94 ,
       6.937, 6.901, 6.867, 6.853, 6.81 , 6.798, 6.786, 6.75 , 6.67 ,
       6.611, 6.575, 6.574, 6.505, 6.485, 6.477, 6.455, 6.411, 6.329,
       6.302, 6.298, 6.295, 6.269, 6.168, 6.13 , 6.123, 6.003, 5.995,
       5.987, 5.984, 5.975, 5.96 , 5.948, 5.89 , 5.889, 5.878, 5.855,
       5.848, 5.833, 5.828, 5.824, 5.813, 5.791, 5.77 , 5.759, 5.754,
       5.716, 5.709, 5.695, 5.689, 5.605, 5.589, 5.548, 5.477, 5.474,
       5.429, 5.399, 5.36 , 5.332, 5.286, 5.268, 5.253, 5.212, 5.194,
       5.192, 5.14 , 5.129, 5.124, 5.123, 5.102, 5.098, 5.073, 5.057,
       5.013, 5.007, 4.971, 4.959, 4.949, 4.898, 4.885, 4.876, 4.874,
       4.867, 4.857, 4.839, 4.8  , 4.788, 4.786, 4.739, 4.715, 4.694,
       4.686, 4.681, 4.677, 4.642, 4.633, 4.61 , 4.571, 4.565, 4.55 ,
       4.518, 4.517, 4.514, 4.512, 4.507, 4.436, 4.419, 4.369, 4.35 ,
       4.332, 4.307, 4.297, 4.292, 4.271, 4.252, 4.218, 4.194, 4.077,
       4.033, 3.995, 3.989, 3.956, 3.931, 3.904, 3.896, 3.845, 3.819,
       3.781, 3.681, 3.678, 3.667, 3.656, 3.655, 3.587, 3.575, 3.465,
       3.34 , 3.006, 2.905, 2.839])
```

In [11]:
```
df['Happiness Score'].nunique()
```

Out[11]:
```
157
```

In [12]:
```
df.shape
```

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

In [14]:
```
#checking blank spaces in happiness score column

df.loc[df['Happiness Score']==' ']
```

Out[14]:

Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom

This shows there are no blank spaces

Making a data frame:

In [22]:

```
df_visualization=df[['Economy (GDP per Capita)','Family','Health (Life Expect
                     'Trust (Government Corruption)','Generosity','Dystopia Re
```

In [23]:

```
df_visualization.columns
```

Out[23]:

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

Visualization of the data:

In [42]:

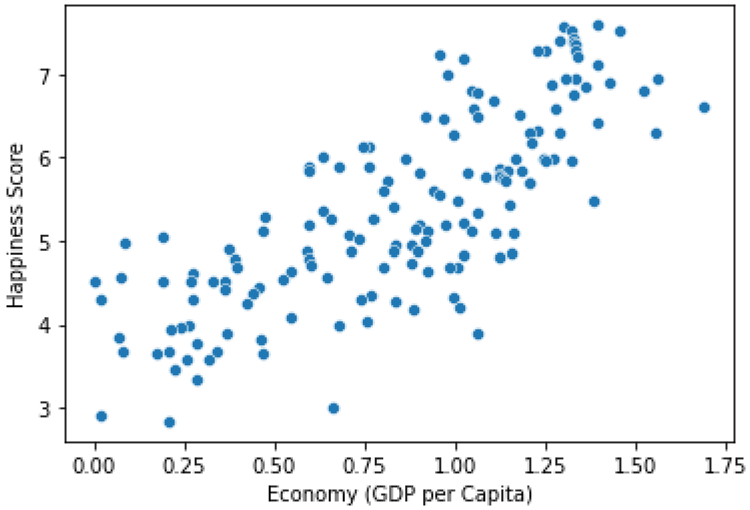
```
import seaborn as sns
```

In [30]:

```
sns.scatterplot(x='Economy (GDP per Capita)',y='Happiness Score',data=df)
```

Out[30]:

```
<AxesSubplot:xlabel='Economy (GDP per Capita)', ylabel='Happiness Score'>
```



In [ ]:

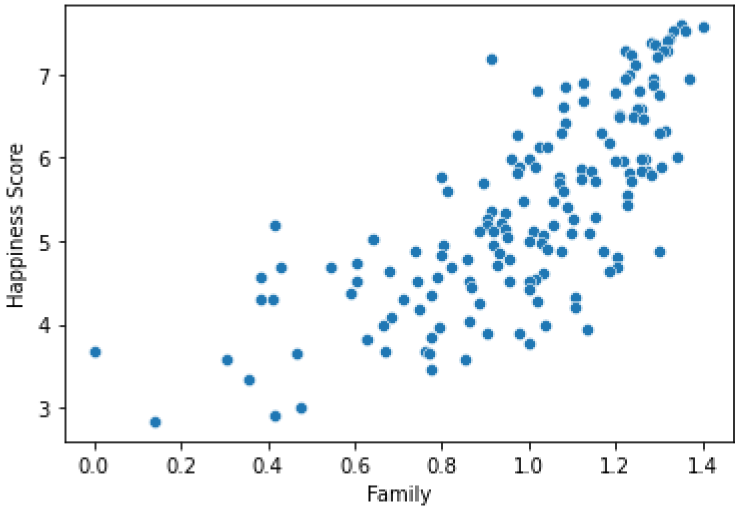
```
It could be seen that the data is not evenly distributed between the economy
thus the economy alone is not the best judge of the happiness score
```

In [31]:

```
sns.scatterplot(x='Family',y='Happiness Score',data=df)
```

Out[31]:

```
<AxesSubplot:xlabel='Family', ylabel='Happiness Score'>
```



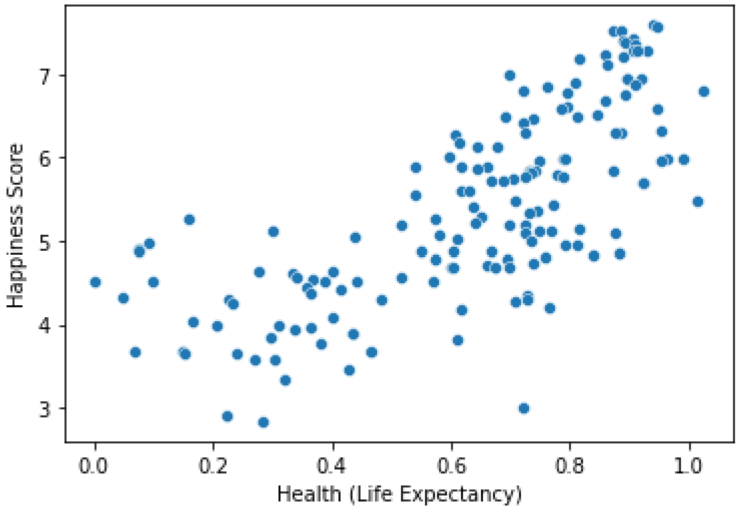
It could be seen that the data is not evenly distributed between the Family and happiness score, thus the family alone is not the best judge of the happiness score

In [33]:

```
sns.scatterplot(x='Health (Life Expectancy)',y='Happiness Score',data=df)
```

Out[33]:

```
<AxesSubplot:xlabel='Health (Life Expectancy)', ylabel='Happiness Score'>
```



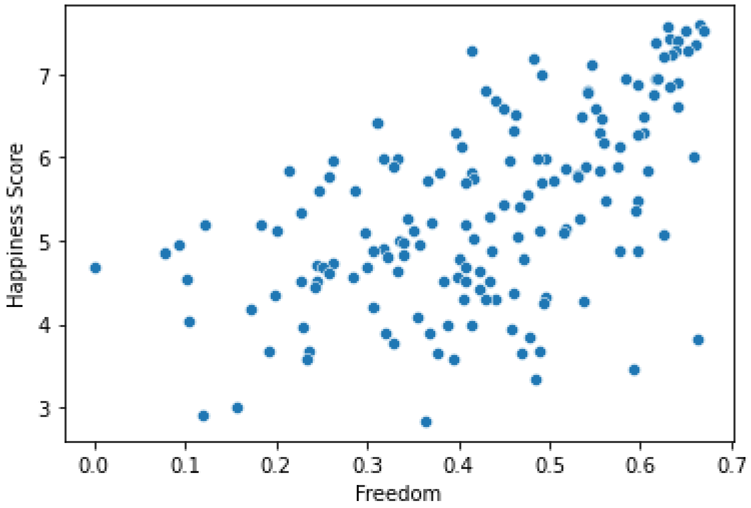
It could be seen that the data is not evenly distributed between the health and happiness score, thus the health alone is not the best judge of the happiness score

In [34]:

```
sns.scatterplot(x='Freedom',y='Happiness Score',data=df)
```

Out[34]:

```
<AxesSubplot:xlabel='Freedom', ylabel='Happiness Score'>
```



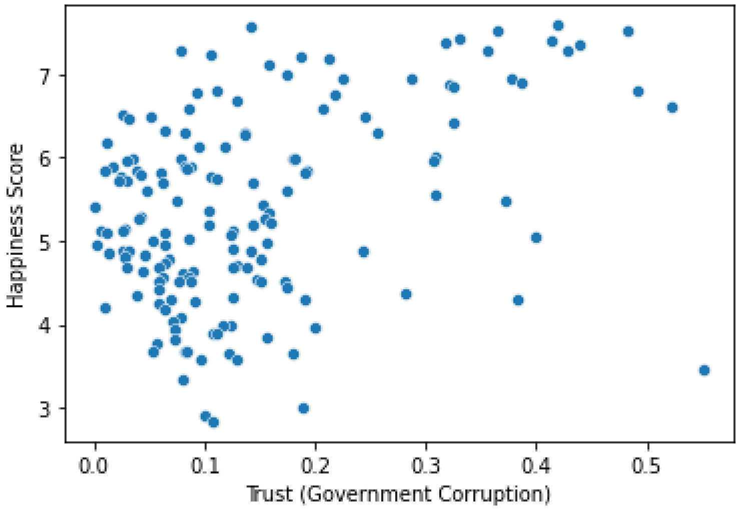
It could be seen that the data is not evenly distributed between the Freedom and happiness score, thus the Freedom alone is not the best judge of the happiness score

In [35]:

```
sns.scatterplot(x='Trust (Government Corruption)',y='Happiness Score',data=df
```

Out[35]:

```
<AxesSubplot:xlabel='Trust (Government Corruption)', ylabel='Happiness Score'>
```



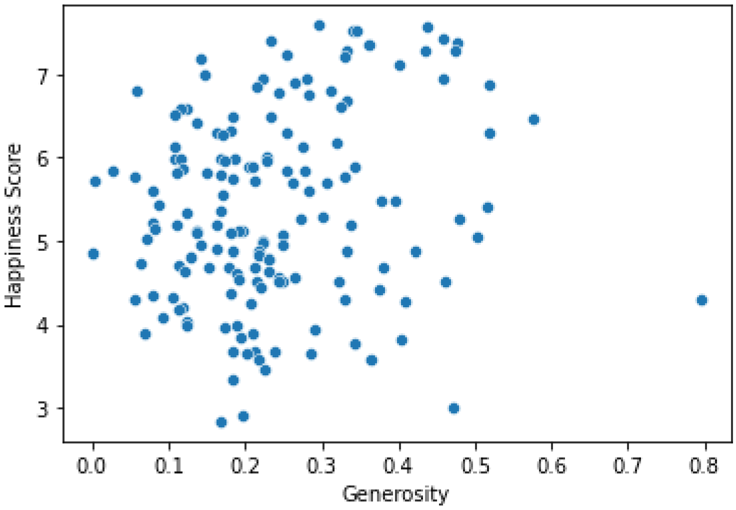
It could be seen that the data is not evenly distributed between the Trust and happiness score, thus the Trust alone is not the best judge of the happiness score

In [36]:

```
sns.scatterplot(x='Generosity',y="Happiness Score",data=df)
```

Out[36]:

```
<AxesSubplot:xlabel='Generosity', ylabel='Happiness Score'>
```



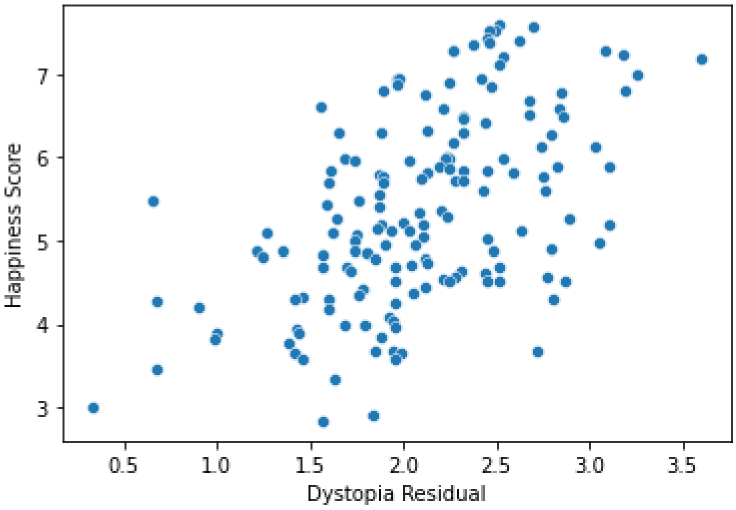
It could be seen that the data is not evenly distributed between the Generosity and happiness score, thus the Generosity alone is not the best judge of the happiness score. However the score of generosity is generally up to around 0.6.

In [38]:

```
sns.scatterplot(x='Dystopia Residual',y='Happiness Score',data=df)
```

Out[38]:

```
<AxesSubplot:xlabel='Dystopia Residual', ylabel='Happiness Score'>
```



It could be seen that the data is not evenly distributed between the dystopia residual and happiness score, thus dystopia residual alone is not the best judge of the happiness score

Describing the data:

In [39]:

```
df.describe()
```

Out[39]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
count	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
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000

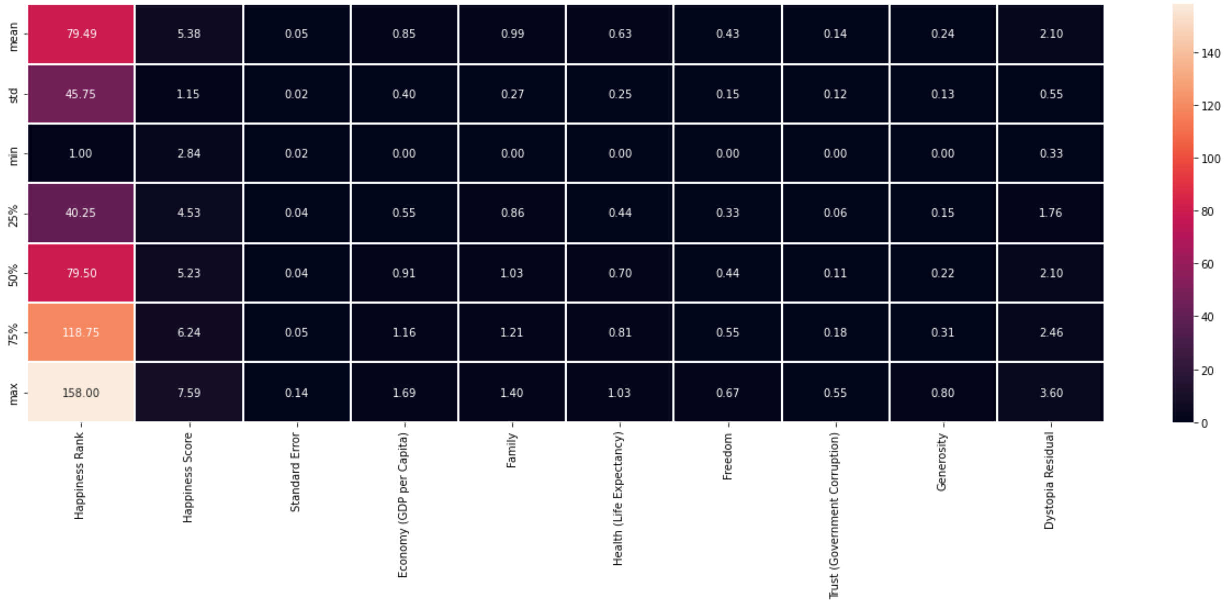
	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730

In [47]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(22,7))
sns.heatmap(df.describe()[1:],annot=True,linewidths=0.1,linecolor='white',fmt
```

Out[47]:

```
<AxesSubplot:>
```



Correlation of the Columns with the Target:

In [51]:

```
df.corr()['Happiness Score'][3:].sort_values()
```

Out[51]:

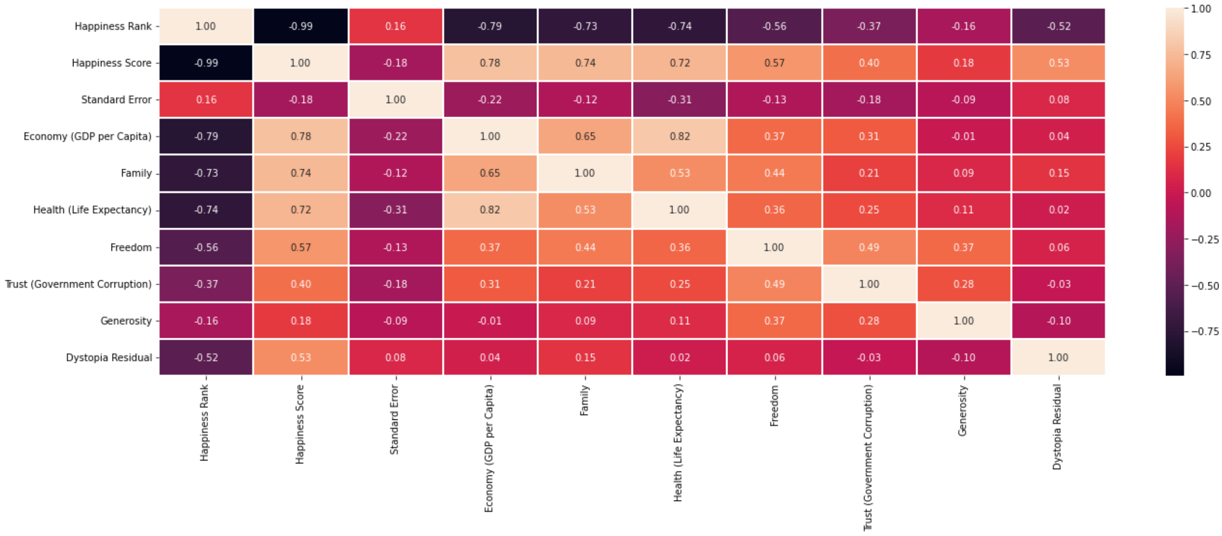
```
Generosity                       0.180319
Trust (Government Corruption)    0.395199
Dystopia Residual                0.530474
Freedom                          0.568211
Health (Life Expectancy)         0.724200
Family                           0.740605
Economy (GDP per Capita)         0.780966
Name: Happiness Score, dtype: float64
```

In [56]:

```
plt.figure(figsize=(22,7))
sns.heatmap(df.corr(),annot=True,linewidth=0.1,fmt='0.2f')
```

Out[56]:

```
<AxesSubplot:>
```

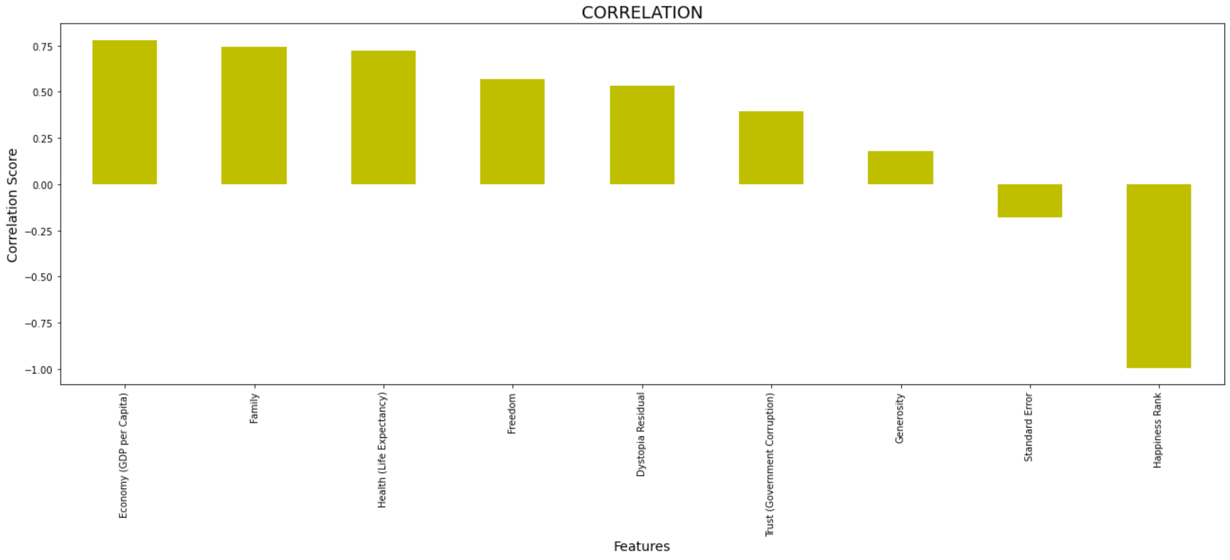


It could be seen from the above that there is a high correlation between Happiness score and Health,Family as well GDP. However, that can not be said about Happiness score and Generosity.

Checking the columns with positive and negative correlation

In [66]:

```
plt.figure(figsize=(22,7))
df.corr()['Happiness Score'].sort_values(ascending=False).drop(['Happiness Sc
plt.xlabel('Features',fontsize=14)
plt.ylabel('Correlation Score',fontsize=14)
plt.title('CORRELATION',fontsize=18)
plt.show()
```



In [77]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [78]:

```
df.skew().sort_values(ascending=False)
```

Out[78]:

```
Standard Error                   1.983439
Trust (Government Corruption)    1.385463
Generosity                       1.001961
```

```
Happiness Score                  0.097769
Happiness Rank                   0.000418
Dystopia Residual               -0.238911
Economy (GDP per Capita)        -0.317575
Freedom                         -0.413462
Health (Life Expectancy)        -0.705328
Family                          -1.006893
dtype: float64
```

Keeping +/-0.5 as the range for skewness, here are the columns which does not lie within this range;

- Trust
- Generosity
- Dystopia Residual
- Health
- Family

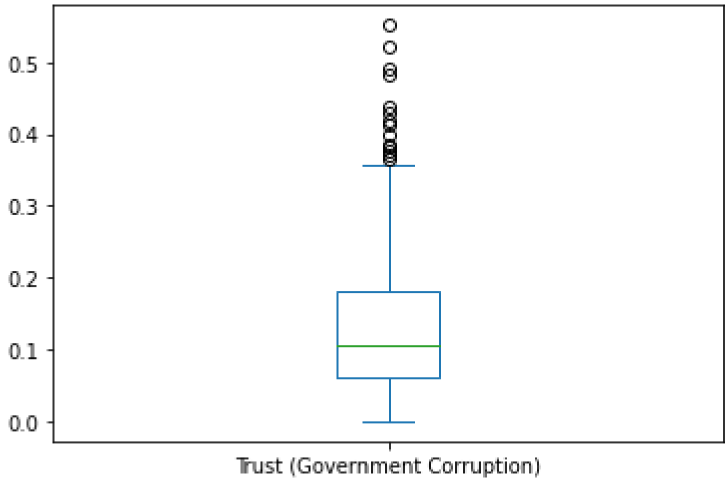
Checking Outliers

In [70]:

```
df['Trust (Government Corruption)'].plot.box()
```

Out[70]:

```
<AxesSubplot:>
```

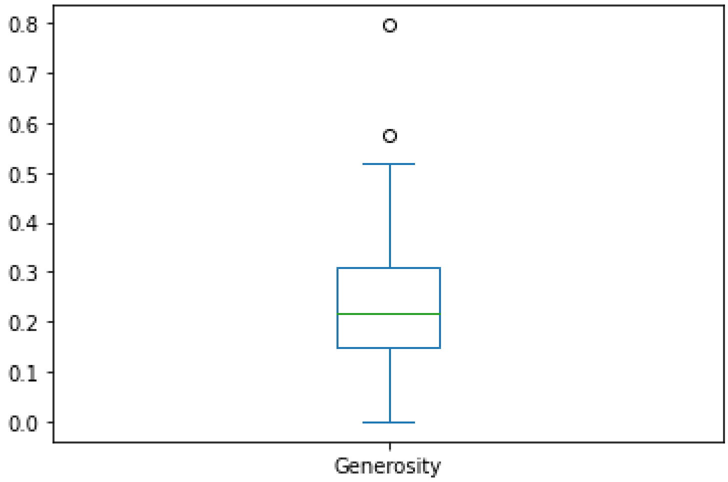


In [71]:

```
df['Generosity'].plot.box()
```

Out[71]:

```
<AxesSubplot:>
```

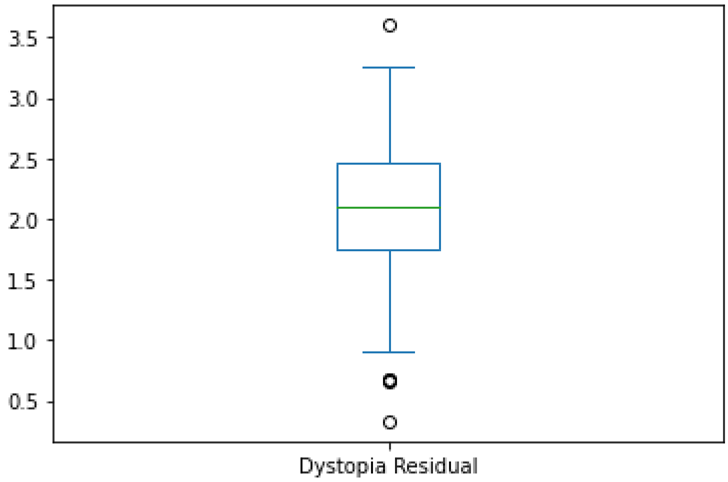


In [72]:

```
df['Dystopia Residual'].plot.box()
```

Out[72]:

```
<AxesSubplot:>
```

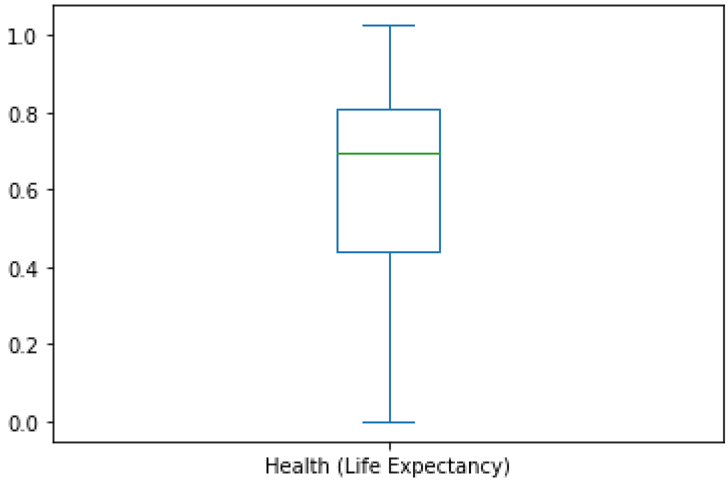


In [73]:

```
df['Health (Life Expectancy)'].plot.box()
```

Out[73]:

```
<AxesSubplot:>
```

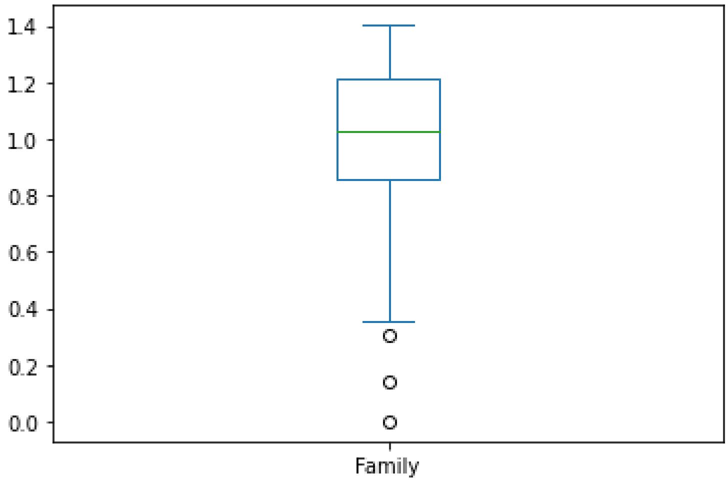


In [74]:

```
df['Family'].plot.box()
```

Out[74]:

```
<AxesSubplot:>
```



Removing Outliers:

In [85]:

```
df.dtypes
```

Out[85]:

```
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 [86]:

```
from sklearn.preprocessing import OrdinalEncoder
enc=OrdinalEncoder()
```

In [89]:

```
for i in df.columns:
    if df[i].dtypes=='object':
        df[i]=enc.fit_transform(df[i].values.reshape(-1,1))
```

In [90]:

```
from scipy.stats import zscore
```

In [93]:

```
zscore(df)
```

Out[93]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Expectar
0	1.238770	1.300256	-1.721000	1.937360	-0.805926	1.369621	1.320281	1.263
1	-0.449465	1.300256	-1.699075	1.914581	0.055889	1.135226	1.514458	1.289
2	-0.909893	1.300256	-1.677149	1.884792	-0.854487	1.192861	1.361054	0.992
3	0.581016	1.300256	-1.655224	1.880411	-0.531526	1.525130	1.251922	1.035
4	-1.194920	-0.040302	-1.633299	1.797179	-0.722845	1.194876	1.221204	1.118
...	...	...	...	...	...	...	...	
153	0.866043	0.965117	1.633576	-1.674055	-0.774917	-1.552987	-0.800520	-0.818
154	-1.436096	0.965117	1.655501	-1.783571	-0.662582	-1.392303	-2.346860	-1.263
155	1.260695	-0.375441	1.677427	-2.076199	0.132534	-0.455245	-1.901086	0.372
156	-1.260695	0.965117	1.699352	-2.164688	2.263962	-2.067566	-2.118467	-1.649
157	1.370321	0.965117	1.721277	-2.222513	1.134182	-1.586334	-3.134725	-1.404

158 rows × 12 columns

In [94]:

```
np.abs(zscore(df))
```

Out[94]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)
0	1.238770	1.300256	1.721000	1.937360	0.805926	1.369621	1.320281	1.263408
1	0.449465	1.300256	1.699075	1.914581	0.055889	1.135226	1.514458	1.289434
2	0.909893	1.300256	1.677149	1.884792	0.854487	1.192861	1.361054	0.992229
3	0.581016	1.300256	1.655224	1.880411	0.531526	1.525130	1.251922	1.035145
4	1.194920	0.040302	1.633299	1.797179	0.722845	1.194876	1.221204	1.118054
...	...	...	...	...	...	...	...	...
153	0.866043	0.965117	1.633576	1.674055	0.774917	1.552987	0.800520	0.818610
154	1.436096	0.965117	1.655501	1.783571	0.662582	1.392303	2.346860	1.263362
155	1.260695	0.375441	1.677427	2.076199	0.132534	0.455245	1.901086	0.372199
156	1.260695	0.965117	1.699352	2.164688	2.263962	2.067566	2.118467	1.649648
157	1.370321	0.965117	1.721277	2.222513	1.134182	1.586334	3.134725	1.404129

158 rows × 12 columns

In [91]:

```
from scipy.stats import zscore
import numpy as np
z=np.abs(zscore(df))
threshold=3
np.where(z>3)
```

Out[91]:

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

In [96]:

```
z.iloc[27]
```

Out[96]:

```
Country                          0.800267
Region                           0.375441
Happiness Rank                   1.129016
Happiness Score                  1.082256
Standard Error                   0.859197
Economy (GDP per Capita)         2.101026
Family                           0.322476
Health (Life Expectancy)         0.678336
Freedom                          1.409878
Trust (Government Corruption)    3.164619
Generosity                       0.700286
Dystopia Residual                0.982677
Name: 27, dtype: float64
```

This shows that at row 27, outlier is found at column 9

In [97]:

```
z.iloc[155]
```

Out[97]:

```
Country                          1.260695
Region                           0.375441
Happiness Rank                   1.677427
Happiness Score                  2.076199
Standard Error                   0.132534
Economy (GDP per Capita)         0.455245
Family                           1.901086
```

```
Health (Life Expectancy)         0.372199
Freedom                          1.809238
Trust (Government Corruption)    0.381419
Generosity                       1.856891
Dystopia Residual                3.208430
Name: 155, dtype: float64
```

This also shows that at row 155 the outlier is found at column 11

In [98]:

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

Out[98]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Fre
0	135.0	9.0	1	7.587	0.03411	1.39651	1.34951	0.94143	0.6
1	58.0	9.0	2	7.561	0.04884	1.30232	1.40223	0.94784	0.6
2	37.0	9.0	3	7.527	0.03328	1.32548	1.36058	0.87464	0.6
3	105.0	9.0	4	7.522	0.03880	1.45900	1.33095	0.88521	0.6
4	24.0	5.0	5	7.427	0.03553	1.32629	1.32261	0.90563	0.6
...	...	...	...	...	...	...	...	...	
150	66.0	8.0	151	3.655	0.05141	0.46534	0.77115	0.15185	0.4
151	20.0	8.0	152	3.587	0.04324	0.25812	0.85188	0.27125	0.3
152	0.0	7.0	153	3.575	0.03084	0.31982	0.30285	0.30335	0.2
154	13.0	8.0	155	3.340	0.03656	0.28665	0.35386	0.31910	0.4
156	21.0	8.0	157	2.905	0.08658	0.01530	0.41587	0.22396	0.

149 rows × 12 columns

In [102…

```
print('Old DataFram: ',df.shape)
print('New DataFrame: ',df_new.shape)
print('Total dropped rows: ',df.shape[0] - df_new.shape[0])
```

```
Old DataFram:  (158, 12)
New DataFrame:  (149, 12)
Total dropped rows:  9
```

In [111…

```
x=df_new.iloc[:,4:]
x
```

Out[111…

	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
...	...	...	...	...	...	...	...	...

	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
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 [177…

```
y=df_new.iloc[:,-9]
```

In [178…

```
y
```

Out[178…

```
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
```

Transforming data to remove Skewness:

In [143…

```
from sklearn.preprocessing import power_transform
```

In [145…

```
x=power_transform(x,method='yeo-johnson')
```

In [146…

```
type(x)
```

Out[146…

```
numpy.ndarray
```

In [147…

```
x.ndim
```

Out[147…

```
2
```

In [148…

```
from sklearn.preprocessing import StandardScaler

sc=StandardScaler()
x=sc.fit_transform(x)
x
```

Out[148…

```
array([[-0.95033703,  1.49245411,  1.65888016, ...,  1.81762514,
          0.65957855,  0.75525527],
        [ 0.37856181,  1.19406711,  2.00912496, ...,  0.32967608,
          1.56260401,  1.11116834],
        [-1.04140135,  1.26661161,  1.73078528, ...,  1.95339957,
          0.97722591,  0.70649452],
        ...,
        [-1.32081011, -1.36521585, -2.08738878, ..., -0.19284339,
          1.13425805, -0.32711524],
        [-0.69272644, -1.43304072, -2.01044293, ..., -0.4344978 ,
         -0.31207994, -0.93216364],
        [ 2.17876606, -1.93874732, -1.90641014, ..., -0.14727363,
         -0.17248608, -0.55359236]])
```

In [149…
```
x.mean()
```
Out[149…
```
1.490232247819002e-18
```

In [150…
```
x.std()
```
Out[150…
```
1.0
```

Splitting the dataframe:

In [157…
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state
```

In [158…
```
x_train.shape
```
Out[158…
```
(119, 8)
```

In [159…
```
x_test.shape
```
Out[159…
```
(30, 8)
```

In [160…
```
y_train.shape
```
Out[160…
```
(119,)
```

In [161…
```
y_test.shape
```
Out[161…
```
(30,)
```

In [207…
```
lr=LinearRegression()
```

In [208…
```
lr.fit(x_train,y_train)
```
Out[208…
```
LinearRegression()
```

In [209…
```
lr.coef_
```

Out[209…

```
array([0.00546828, 0.39011802, 0.24669554, 0.26076631, 0.15451022,
       0.07637051, 0.09156642, 0.51200131])
```

In [210…

```python
lr.intercept_
```

Out[210…

```
5.423939007127552
```

In [215…

```python
#predict the value
pred=lr.predict(x_test)
print('Predicted Happiness Score:',pred)
print('Actual Happiness Score:',y_test)
```

```
Predicted Happiness Score: [3.82469506 7.18946795 5.21839108 4.31296686 7.4277
8163 4.94266194
 5.12421968 4.68122172 6.7910441  5.15320319 4.31145442 4.77513266
 4.24062967 4.88142806 4.27413508 7.01407612 7.29941989 5.31674774
 5.08674888 4.66732645 5.15653771 5.8147805  5.93672823 5.08883486
 4.53723854 7.33907998 6.17137046 3.74264684 5.71192996 4.64301279]
Actual Happiness Score: 144    3.819
8      7.286
77     5.268
124    4.419
4      7.427
92     5.007
81     5.192
111    4.677
24     6.786
73     5.399
126    4.350
90     5.057
132    4.252
97     4.885
129    4.297
16     6.946
9      7.284
74     5.360
84     5.129
109    4.686
87     5.102
53     5.855
48     5.960
83     5.140
121    4.512
6      7.378
45     5.987
152    3.575
60     5.770
107    4.715
Name: Happiness Score, dtype: float64
```

In [205…

```python
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
from sklearn.metrics import r2_score

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score,confusion_matrix,classification_re
```

In [180…

```
import warnings
warnings.filterwarnings('ignore')
```

In [185…

```
for i in range(0,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_s
    lr.fit(x_train,y_train)
    pred_train=lr.predict(x_train)
    pred_test=lr.predict(x_test)
    print(f"At random state{i},the training accuracy is:- {r2_score(y_train,p
    print(f"At random state{i},the testing accuracy is:- {r2_score(y_test,pre
    print("\n")
```

```
At random state0,the training accuracy is:- 0.9941982955173955
At random state0,the testing accuracy is:- 0.994748240521683


At random state1,the training accuracy is:- 0.9940659187607843
At random state1,the testing accuracy is:- 0.9955203804156323


At random state2,the training accuracy is:- 0.9946763776418325
At random state2,the testing accuracy is:- 0.9923505339277224


At random state3,the training accuracy is:- 0.9945717527577527
At random state3,the testing accuracy is:- 0.9924390983409589


At random state4,the training accuracy is:- 0.9942864745534269
At random state4,the testing accuracy is:- 0.9934322147420919


At random state5,the training accuracy is:- 0.9948747070828916
At random state5,the testing accuracy is:- 0.9908303500241534


At random state6,the training accuracy is:- 0.9952331413322157
At random state6,the testing accuracy is:- 0.9895293341455182


At random state7,the training accuracy is:- 0.9951273808383169
At random state7,the testing accuracy is:- 0.9869561914382385


At random state8,the training accuracy is:- 0.9946779101905551
At random state8,the testing accuracy is:- 0.9927406715156951


At random state9,the training accuracy is:- 0.9947879096062683
At random state9,the testing accuracy is:- 0.9905409394865592


At random state10,the training accuracy is:- 0.9946253158028778
At random state10,the testing accuracy is:- 0.9926012876306312


At random state11,the training accuracy is:- 0.9937773957527946
At random state11,the testing accuracy is:- 0.9961692315226315


At random state12,the training accuracy is:- 0.9943529897998734
At random state12,the testing accuracy is:- 0.9935782871651744
At random state13,the training accuracy is:- 0.9946167264112867
At random state13,the testing accuracy is:- 0.9919721866352942


At random state14,the training accuracy is:- 0.9937439718619299
At random state14,the testing accuracy is:- 0.9955362193273237


At random state15,the training accuracy is:- 0.9937657641127642
At random state15,the testing accuracy is:- 0.9964539621881651


At random state16,the training accuracy is:- 0.9946688288395418
At random state16,the testing accuracy is:- 0.9922417788801161


At random state17,the training accuracy is:- 0.9943767226904531
At random state17,the testing accuracy is:- 0.9934436985419958


At random state18,the training accuracy is:- 0.9945904959953634
At random state18,the testing accuracy is:- 0.992700319436156


At random state19,the training accuracy is:- 0.9945376891808576
At random state19,the testing accuracy is:- 0.9924953015302361


At random state20,the training accuracy is:- 0.9945967794818821
At random state20,the testing accuracy is:- 0.9925928138771766


At random state21,the training accuracy is:- 0.9946614465917902
At random state21,the testing accuracy is:- 0.9913295178457557


At random state22,the training accuracy is:- 0.9952151158942136
At random state22,the testing accuracy is:- 0.9865790520619937


At random state23,the training accuracy is:- 0.9942161942080434
At random state23,the testing accuracy is:- 0.9942630098359643


At random state24,the training accuracy is:- 0.9940278792070908
At random state24,the testing accuracy is:- 0.9946195497756716


At random state25,the training accuracy is:- 0.9944061315387105
At random state25,the testing accuracy is:- 0.9933157697347529


At random state26,the training accuracy is:- 0.9945963939427603
At random state26,the testing accuracy is:- 0.9929930206779178


At random state27,the training accuracy is:- 0.9943987612641381
At random state27,the testing accuracy is:- 0.9931835619671577


At random state28,the training accuracy is:- 0.994447704752868
At random state28,the testing accuracy is:- 0.9934201832726481
At random state29,the training accuracy is:- 0.9942665066490344
At random state29,the testing accuracy is:- 0.993143766814812


At random state30,the training accuracy is:- 0.9935872862206718
At random state30,the testing accuracy is:- 0.995579668165009


At random state31,the training accuracy is:- 0.9940130540652238
At random state31,the testing accuracy is:- 0.9952845155671127


At random state32,the training accuracy is:- 0.994460973825763
At random state32,the testing accuracy is:- 0.9930646167776281


At random state33,the training accuracy is:- 0.9948057068837589
At random state33,the testing accuracy is:- 0.9917457514261152


At random state34,the training accuracy is:- 0.9945937057578438
At random state34,the testing accuracy is:- 0.9911818109043807


At random state35,the training accuracy is:- 0.9945552985305802
At random state35,the testing accuracy is:- 0.992837783220487


At random state36,the training accuracy is:- 0.9938672567589257
At random state36,the testing accuracy is:- 0.9960208305849694


At random state37,the training accuracy is:- 0.99315779961674
At random state37,the testing accuracy is:- 0.9974597733315452


At random state38,the training accuracy is:- 0.9946650371290245
At random state38,the testing accuracy is:- 0.9910632691379363


At random state39,the training accuracy is:- 0.9941221806748909
At random state39,the testing accuracy is:- 0.9947104639856633


At random state40,the training accuracy is:- 0.9944094045068502
At random state40,the testing accuracy is:- 0.9934613219054371


At random state41,the training accuracy is:- 0.9948180315431584
At random state41,the testing accuracy is:- 0.9910329120269092


At random state42,the training accuracy is:- 0.9938086944419373
At random state42,the testing accuracy is:- 0.9962654304730582


At random state43,the training accuracy is:- 0.993935294907153
At random state43,the testing accuracy is:- 0.9947764286910485


At random state44,the training accuracy is:- 0.9947898394424185
At random state44,the testing accuracy is:- 0.9919751948498898
At random state45,the training accuracy is:- 0.9943456712020234
At random state45,the testing accuracy is:- 0.9934316203005564


At random state46,the training accuracy is:- 0.9947520270303125
At random state46,the testing accuracy is:- 0.9909368853241983


At random state47,the training accuracy is:- 0.993887811854278
At random state47,the testing accuracy is:- 0.9950578877543679


At random state48,the training accuracy is:- 0.9949689795777287
At random state48,the testing accuracy is:- 0.9903634942080856


At random state49,the training accuracy is:- 0.9937808716020665
At random state49,the testing accuracy is:- 0.9961272169377937


At random state50,the training accuracy is:- 0.9942524471298133
At random state50,the testing accuracy is:- 0.993646847249328


At random state51,the training accuracy is:- 0.994336989564615
At random state51,the testing accuracy is:- 0.9936812516208076


At random state52,the training accuracy is:- 0.9949879169218917
At random state52,the testing accuracy is:- 0.9902360612842843


At random state53,the training accuracy is:- 0.9944600722092933
At random state53,the testing accuracy is:- 0.9926805453140161


At random state54,the training accuracy is:- 0.9947082856704073
At random state54,the testing accuracy is:- 0.9911993743481138


At random state55,the training accuracy is:- 0.99518264648022
At random state55,the testing accuracy is:- 0.9902562669697006


At random state56,the training accuracy is:- 0.9934717058085164
At random state56,the testing accuracy is:- 0.9969230831023292


At random state57,the training accuracy is:- 0.9947644272727969
At random state57,the testing accuracy is:- 0.9907049987659408


At random state58,the training accuracy is:- 0.9939287363881271
At random state58,the testing accuracy is:- 0.9955753822678169


At random state59,the training accuracy is:- 0.9942130268395091
At random state59,the testing accuracy is:- 0.9943081121133612


At random state60,the training accuracy is:- 0.9932932247972182
At random state60,the testing accuracy is:- 0.9977542532602311
At random state61,the training accuracy is:- 0.9950179025075516
At random state61,the testing accuracy is:- 0.9904467090546087


At random state62,the training accuracy is:- 0.9945589968217194
At random state62,the testing accuracy is:- 0.9925125585275582


At random state63,the training accuracy is:- 0.9943517571995281
At random state63,the testing accuracy is:- 0.9930933432641722


At random state64,the training accuracy is:- 0.994519989993832
At random state64,the testing accuracy is:- 0.9928826395232061


At random state65,the training accuracy is:- 0.9943757852023957
At random state65,the testing accuracy is:- 0.9934175996548522


At random state66,the training accuracy is:- 0.9940863147803344
At random state66,the testing accuracy is:- 0.9947308938720409


At random state67,the training accuracy is:- 0.9948701298292594
At random state67,the testing accuracy is:- 0.9904494334434402


At random state68,the training accuracy is:- 0.9938380191225089
At random state68,the testing accuracy is:- 0.9951807575315733


At random state69,the training accuracy is:- 0.994300087424532
At random state69,the testing accuracy is:- 0.9941500819528817


At random state70,the training accuracy is:- 0.9941661834000435
At random state70,the testing accuracy is:- 0.9947036444924617


At random state71,the training accuracy is:- 0.994123980143541
At random state71,the testing accuracy is:- 0.9946885682247192


At random state72,the training accuracy is:- 0.995040342199831
At random state72,the testing accuracy is:- 0.9917602479380591


At random state73,the training accuracy is:- 0.9942945242575043
At random state73,the testing accuracy is:- 0.9941452627470223


At random state74,the training accuracy is:- 0.994345641672881
At random state74,the testing accuracy is:- 0.9937824962271558


At random state75,the training accuracy is:- 0.9944286881013005
At random state75,the testing accuracy is:- 0.9932049007978031


At random state76,the training accuracy is:- 0.9951866515097408
At random state76,the testing accuracy is:- 0.9882662656015518
At random state77,the training accuracy is:- 0.993793089575987
At random state77,the testing accuracy is:- 0.9959430631492643


At random state78,the training accuracy is:- 0.9945810175454461
At random state78,the testing accuracy is:- 0.9914905656152967


At random state79,the training accuracy is:- 0.9940678100733245
At random state79,the testing accuracy is:- 0.9948254769254349


At random state80,the training accuracy is:- 0.9946405052295529
At random state80,the testing accuracy is:- 0.9914513007632682


At random state81,the training accuracy is:- 0.9938573366996322
At random state81,the testing accuracy is:- 0.9953341670274829


At random state82,the training accuracy is:- 0.9952538506290778
At random state82,the testing accuracy is:- 0.9886974884891839


At random state83,the training accuracy is:- 0.9946568910608673
At random state83,the testing accuracy is:- 0.9915271201916132


At random state84,the training accuracy is:- 0.993599605777899
At random state84,the testing accuracy is:- 0.9966876771530242


At random state85,the training accuracy is:- 0.9951858848347819
At random state85,the testing accuracy is:- 0.9891328405954961


At random state86,the training accuracy is:- 0.9943909928757673
At random state86,the testing accuracy is:- 0.9938283441772622


At random state87,the training accuracy is:- 0.9941351841143413
At random state87,the testing accuracy is:- 0.9940007952137104


At random state88,the training accuracy is:- 0.9943306148007421
At random state88,the testing accuracy is:- 0.993803134341047


At random state89,the training accuracy is:- 0.994298757196371
At random state89,the testing accuracy is:- 0.9935345507833064


At random state90,the training accuracy is:- 0.9946785237405779
At random state90,the testing accuracy is:- 0.991737283831643


At random state91,the training accuracy is:- 0.9937609718271604
At random state91,the testing accuracy is:- 0.996846667620555


At random state92,the training accuracy is:- 0.995392980646686
At random state92,the testing accuracy is:- 0.9889524919070365
At random state93,the training accuracy is:- 0.9952996733871701
At random state93,the testing accuracy is:- 0.9899701944445045


At random state94,the training accuracy is:- 0.9941734518819946
At random state94,the testing accuracy is:- 0.994736420854719


At random state95,the training accuracy is:- 0.9940874083523951
At random state95,the testing accuracy is:- 0.9945769449327602


At random state96,the training accuracy is:- 0.9946176028659457
At random state96,the testing accuracy is:- 0.9918254780965464


At random state97,the training accuracy is:- 0.9938547708534086
At random state97,the testing accuracy is:- 0.9952291293325625


At random state98,the training accuracy is:- 0.9950717620083237
At random state98,the testing accuracy is:- 0.9885575756461192


At random state99,the training accuracy is:- 0.9946773121778363
At random state99,the testing accuracy is:- 0.9922524718172714
```

In [186…

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

Out[186…

```
LinearRegression()
```

In [193…

```
lr.predict(x_test)
```

Out[193…

```
array([3.82469506, 7.18946795, 5.21839108, 4.31296686, 7.42778163,
       4.94266194, 5.12421968, 4.68122172, 6.7910441 , 5.15320319,
       4.31145442, 4.77513266, 4.24062967, 4.88142806, 4.27413508,
       7.01407612, 7.29941989, 5.31674774, 5.08674888, 4.66732645,
       5.15653771, 5.8147805 , 5.93672823, 5.08883486, 4.53723854,
       7.33907998, 6.17137046, 3.74264684, 5.71192996, 4.64301279])
```

In [195…

```
pred_test=lr.predict(x_test)
```

In [196…

```
pred_test
```

Out[196…

```
array([3.82469506, 7.18946795, 5.21839108, 4.31296686, 7.42778163,
       4.94266194, 5.12421968, 4.68122172, 6.7910441 , 5.15320319,
       4.31145442, 4.77513266, 4.24062967, 4.88142806, 4.27413508,
       7.01407612, 7.29941989, 5.31674774, 5.08674888, 4.66732645,
       5.15653771, 5.8147805 , 5.93672823, 5.08883486, 4.53723854,
       7.33907998, 6.17137046, 3.74264684, 5.71192996, 4.64301279])
```

In [198…

```
print(r2_score(y_test,pred_test))
```

```
0.9922524718172714
```

Cross Validation:

In [189…

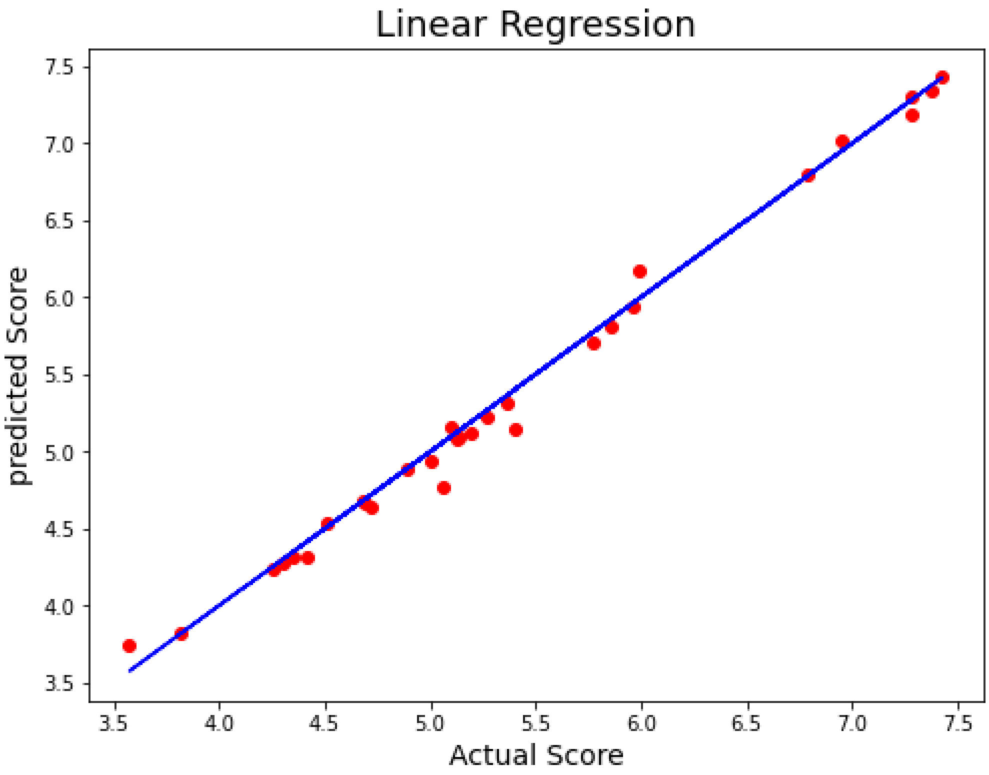
```python
Train_accuracy=r2_score(y_train,pred_train)
Test_accuracy=r2_score(y_test,pred_test)

from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score=cross_val_score(lr,x,y,cv=j)
    cv_mean=cv_score.mean()
    print(f"At cross fold {j} the cv score is {cv_mean} and accuracy score fo
    print("\n")
```

```
At cross fold 2 the cv score is 0.8496218971265873 and accuracy score for trai
ning is 0.9946773121778363 and accuracy for the testing is0.9922524718172714


At cross fold 3 the cv score is 0.8848139146287543 and accuracy score for trai
ning is 0.9946773121778363 and accuracy for the testing is0.9922524718172714


At cross fold 4 the cv score is 0.824000417758892 and accuracy score for train
ing is 0.9946773121778363 and accuracy for the testing is0.9922524718172714


At cross fold 5 the cv score is 0.8109522872943644 and accuracy score for trai
ning is 0.9946773121778363 and accuracy for the testing is0.9922524718172714


At cross fold 6 the cv score is 0.7412325287611082 and accuracy score for trai
ning is 0.9946773121778363 and accuracy for the testing is0.9922524718172714


At cross fold 7 the cv score is 0.6022631910382402 and accuracy score for trai
ning is 0.9946773121778363 and accuracy for the testing is0.9922524718172714


At cross fold 8 the cv score is 0.5756486133834751 and accuracy score for trai
ning is 0.9946773121778363 and accuracy for the testing is0.9922524718172714


At cross fold 9 the cv score is 0.4832960943167747 and accuracy score for trai
ning is 0.9946773121778363 and accuracy for the testing is0.9922524718172714
```

Fold 3 is selected, that is cv=3 which gives a high score of 0.884

In [191…

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_test,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual Score',fontsize=14)
plt.ylabel('predicted Score',fontsize=14)
plt.title('Linear Regression',fontsize=18)
plt.savefig('lr.png')
plt.show()
```



The BestFit line is covering most of the data point which is an indication of how good the model is.

```
In [ ]:
```