Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

In [1]:

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

In [2]:

```
df=pd.read_csv("2015")
```

To display top 10 rows

```
In [3]: df.head(10)
```

```
Out[3]:
```

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residua
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176
5	Finland	Western Europe	6	7.406	0.03140	1.29025	1.31826	0.88911	0.64169	0.41372	0.23351	2.61955
6	Netherlands	Western Europe	7	7.378	0.02799	1.32944	1.28017	0.89284	0.61576	0.31814	0.47610	2.46570
7	Sweden	Western Europe	8	7.364	0.03157	1.33171	1.28907	0.91087	0.65980	0.43844	0.36262	2.37119
8	New Zealand	Australia and New Zealand	9	7.286	0.03371	1.25018	1.31967	0.90837	0.63938	0.42922	0.47501	2.26425
9	Australia	Australia and New Zealand	10	7.284	0.04083	1.33358	1.30923	0.93156	0.65124	0.35637	0.43562	2.26646

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]:
```
# Display the statistical summary
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

```
In [6]: # To display the col headings
        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')
```

```
In [7]: cols=df.dropna(axis=1)
```

```
In [8]: cols.columns
```

```
Out[8]: 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')
```

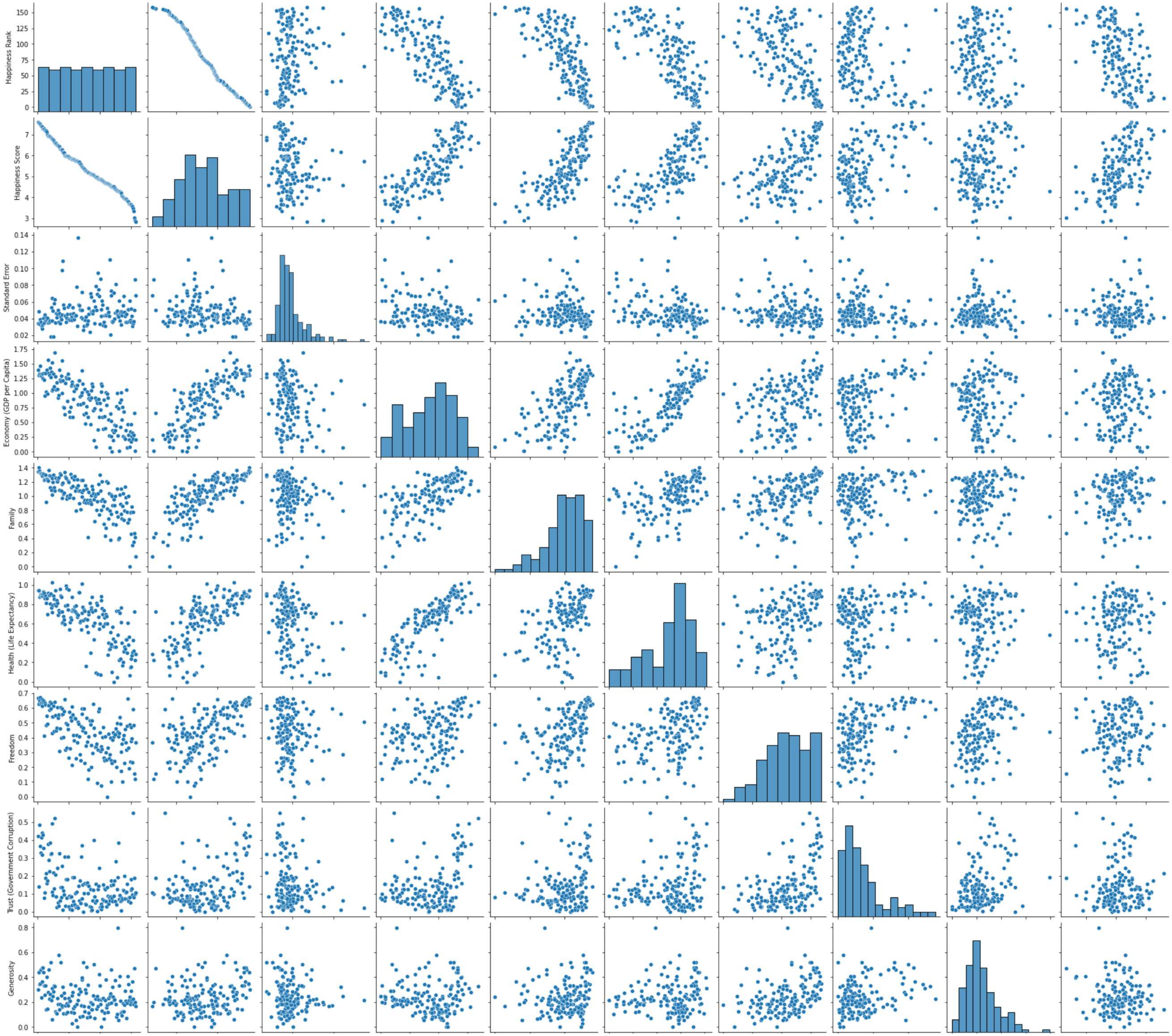
EDA and Visualization

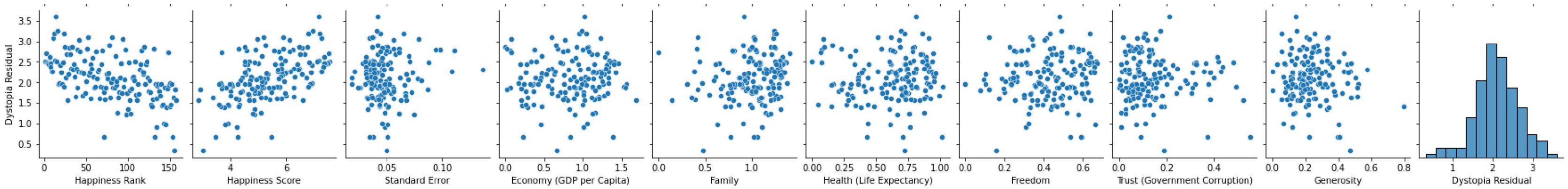
In [9]:

```
sns.pairplot(cols)
```

Out[9]:

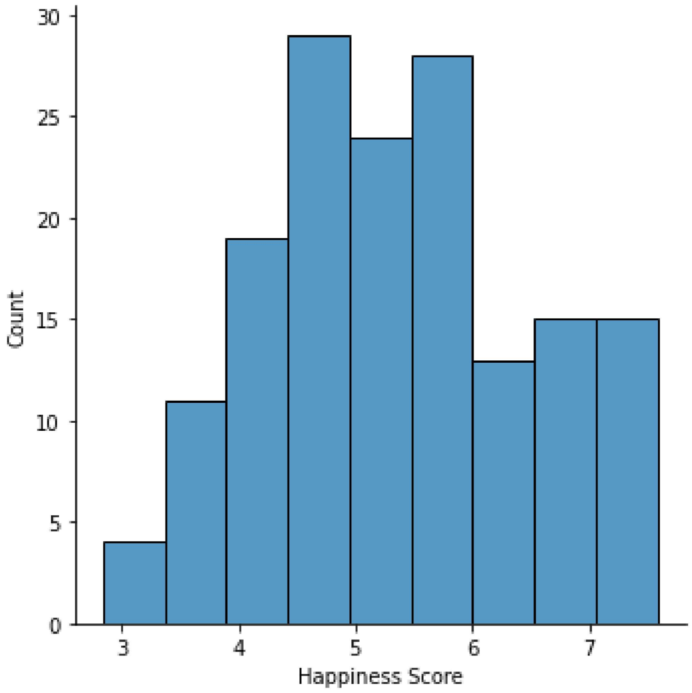
```
<seaborn.axisgrid.PairGrid at 0x23078632040>
```


Happiness Rank
150
125
100
75
50
25
0
Happiness Score
7
6
5
4
3
Standard Error
0.14
0.12
0.10
0.08
0.06
0.04
0.02
Economy (GDP per Capita)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Family
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Health (Life Expectancy)
1.0
0.8
0.6
0.4
0.2
0.0
Freedom
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Trust (Government Corruption)
0.5
0.4
0.3
0.2
0.1
0.0
Generosity
0.8
0.6
0.4
0.2
0.0



In [12]:
```
sns.displot(df['Happiness Score'])
```

Out[12]:
```
<seaborn.axisgrid.FacetGrid at 0x23000132a00>
```



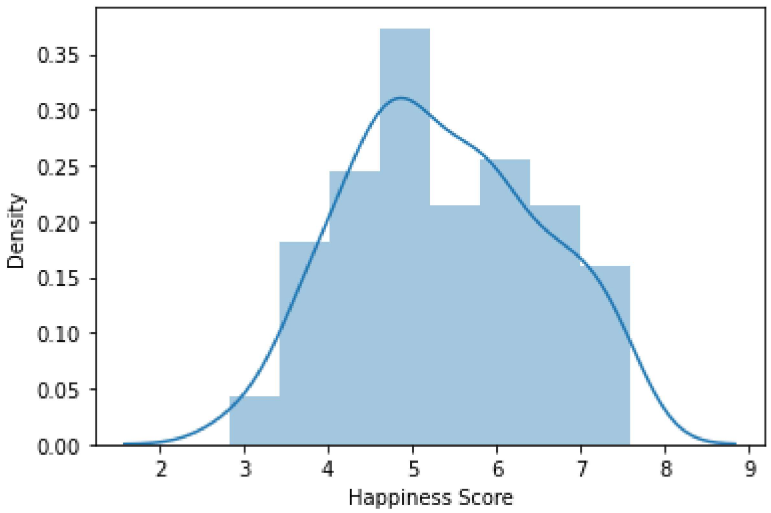
In [14]:

```
# We use displot in older version we get distplot use displot
sns.distplot(df['Happiness Score'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated 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 function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[14]:

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



In [17]:

```
df1=cols[['Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family']]
df1
```

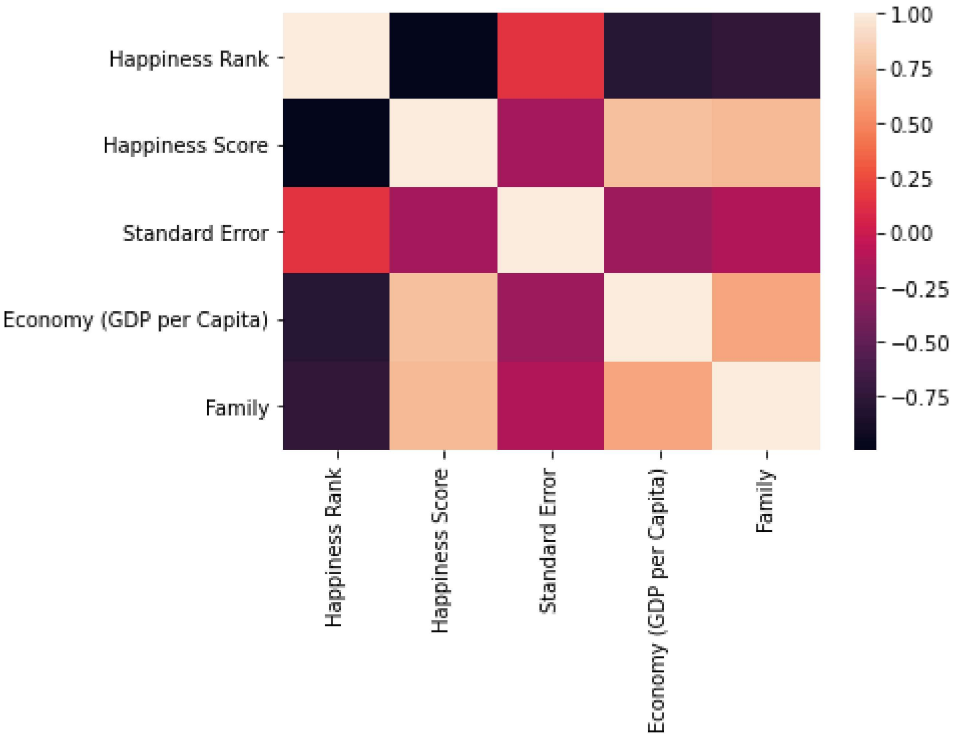
Out[17]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family
0	1	7.587	0.03411	1.39651	1.34951
1	2	7.561	0.04884	1.30232	1.40223
2	3	7.527	0.03328	1.32548	1.36058
3	4	7.522	0.03880	1.45900	1.33095
4	5	7.427	0.03553	1.32629	1.32261
...	...	...	...	...	...
153	154	3.465	0.03464	0.22208	0.77370
154	155	3.340	0.03656	0.28665	0.35386
155	156	3.006	0.05015	0.66320	0.47489
156	157	2.905	0.08658	0.01530	0.41587
157	158	2.839	0.06727	0.20868	0.13995

158 rows × 5 columns

In [18]:
```
sns.heatmap(df1.corr())
```

Out[18]:
```
<AxesSubplot:>
```



To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [20]:
```
x=df1[['Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family']]
y=df1[['Happiness Rank']]
```

To split the dataset into test data

In [21]:

```
# importing lib for splitting test data
from sklearn.model_selection import train_test_split
```

In [22]:

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

In [23]:

```
from sklearn.linear_model import LinearRegression

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

Out[23]:

```
LinearRegression()
```

In [24]:

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

```
[2.84217094e-14]
```

In [25]:

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

```
1.0
```

In [26]:

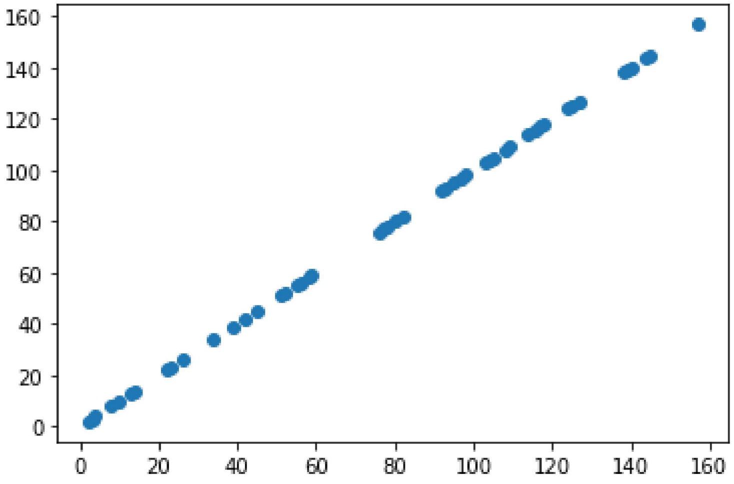
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[26]:

	0	1	2	3	4
0	1.0	1.409526e-15	-3.715877e-15	-7.549426e-16	-2.785267e-16

In [27]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```

Out[27]:
```
<matplotlib.collections.PathCollection at 0x230035fecd0>
```



In [ ]: