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(r'C:\Users\user\Downloads\2_2015.csv')
df
```

Out[2]:

| | Country | Region | Happiness Rank | Happiness Score | Standard Error | Economy (GDP per Capita) | Family | Health (Life Expectancy) | Freedom | Tru (Governme Corruptio |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Switzerland | Western Europe | 1 | 7.587 | 0.03411 | 1.39651 | 1.34951 | 0.94143 | 0.66557 | 0.419 |
| **1** | Iceland | Western Europe | 2 | 7.561 | 0.04884 | 1.30232 | 1.40223 | 0.94784 | 0.62877 | 0.141 |
| **2** | Denmark | Western Europe | 3 | 7.527 | 0.03328 | 1.32548 | 1.36058 | 0.87464 | 0.64938 | 0.483 |
| **3** | Norway | Western Europe | 4 | 7.522 | 0.03880 | 1.45900 | 1.33095 | 0.88521 | 0.66973 | 0.365 |
| **4** | Canada | North America | 5 | 7.427 | 0.03553 | 1.32629 | 1.32261 | 0.90563 | 0.63297 | 0.329 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **153** | Rwanda | Sub-Saharan Africa | 154 | 3.465 | 0.03464 | 0.22208 | 0.77370 | 0.42864 | 0.59201 | 0.551 |
| **154** | Benin | Sub-Saharan Africa | 155 | 3.340 | 0.03656 | 0.28665 | 0.35386 | 0.31910 | 0.48450 | 0.080 |
| **155** | Syria | Middle East and Northern Africa | 156 | 3.006 | 0.05015 | 0.66320 | 0.47489 | 0.72193 | 0.15684 | 0.189 |
| **156** | Burundi | Sub-Saharan Africa | 157 | 2.905 | 0.08658 | 0.01530 | 0.41587 | 0.22396 | 0.11850 | 0.100 |
| **157** | Togo | Sub-Saharan Africa | 158 | 2.839 | 0.06727 | 0.20868 | 0.13995 | 0.28443 | 0.36453 | 0.107 |

158 rows × 12 columns

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) |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Switzerland | Western Europe | 1 | 7.587 | 0.03411 | 1.39651 | 1.34951 | 0.94143 | 0.66557 | 0.41978 |
| **1** | Iceland | Western Europe | 2 | 7.561 | 0.04884 | 1.30232 | 1.40223 | 0.94784 | 0.62877 | 0.14145 |
| **2** | Denmark | Western Europe | 3 | 7.527 | 0.03328 | 1.32548 | 1.36058 | 0.87464 | 0.64938 | 0.48357 |
| **3** | Norway | Western Europe | 4 | 7.522 | 0.03880 | 1.45900 | 1.33095 | 0.88521 | 0.66973 | 0.36503 |
| **4** | Canada | North America | 5 | 7.427 | 0.03553 | 1.32629 | 1.32261 | 0.90563 | 0.63297 | 0.32957 |
| **5** | Finland | Western Europe | 6 | 7.406 | 0.03140 | 1.29025 | 1.31826 | 0.88911 | 0.64169 | 0.41372 |
| **6** | Netherlands | Western Europe | 7 | 7.378 | 0.02799 | 1.32944 | 1.28017 | 0.89284 | 0.61576 | 0.31814 |
| **7** | Sweden | Western Europe | 8 | 7.364 | 0.03157 | 1.33171 | 1.28907 | 0.91087 | 0.65980 | 0.43844 |
| **8** | New Zealand | Australia and New Zealand | 9 | 7.286 | 0.03371 | 1.25018 | 1.31967 | 0.90837 | 0.63938 | 0.42922 |
| **9** | Australia | Australia and New Zealand | 10 | 7.284 | 0.04083 | 1.33358 | 1.30923 | 0.93156 | 0.65124 | 0.35637 |

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]:

```
df.describe()
```

Out[5]:

| | Happiness Rank | Happiness Score | Standard Error | Economy (GDP per Capita) | Family | Health (Life Expectancy) | Freedom | Trust (Government Corruption) | Gener |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 158.000000 | 158.000000 | 158.000000 | 158.000000 | 158.000000 | 158.000000 | 158.000000 | 158.000000 | 158.00 |
| **mean** | 79.493671 | 5.375734 | 0.047885 | 0.846137 | 0.991046 | 0.630259 | 0.428615 | 0.143422 | 0.23 |
| **std** | 45.754363 | 1.145010 | 0.017146 | 0.403121 | 0.272369 | 0.247078 | 0.150693 | 0.120034 | 0.12 |
| **min** | 1.000000 | 2.839000 | 0.018480 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| **25%** | 40.250000 | 4.526000 | 0.037268 | 0.545808 | 0.856823 | 0.439185 | 0.328330 | 0.061675 | 0.15 |
| **50%** | 79.500000 | 5.232500 | 0.043940 | 0.910245 | 1.029510 | 0.696705 | 0.435515 | 0.107220 | 0.21 |
| **75%** | 118.750000 | 6.243750 | 0.052300 | 1.158448 | 1.214405 | 0.811013 | 0.549092 | 0.180255 | 0.30 |
| **max** | 158.000000 | 7.587000 | 0.136930 | 1.690420 | 1.402230 | 1.025250 | 0.669730 | 0.551910 | 0.79 |

In [6]:

```
df.columns
```

Out[6]:

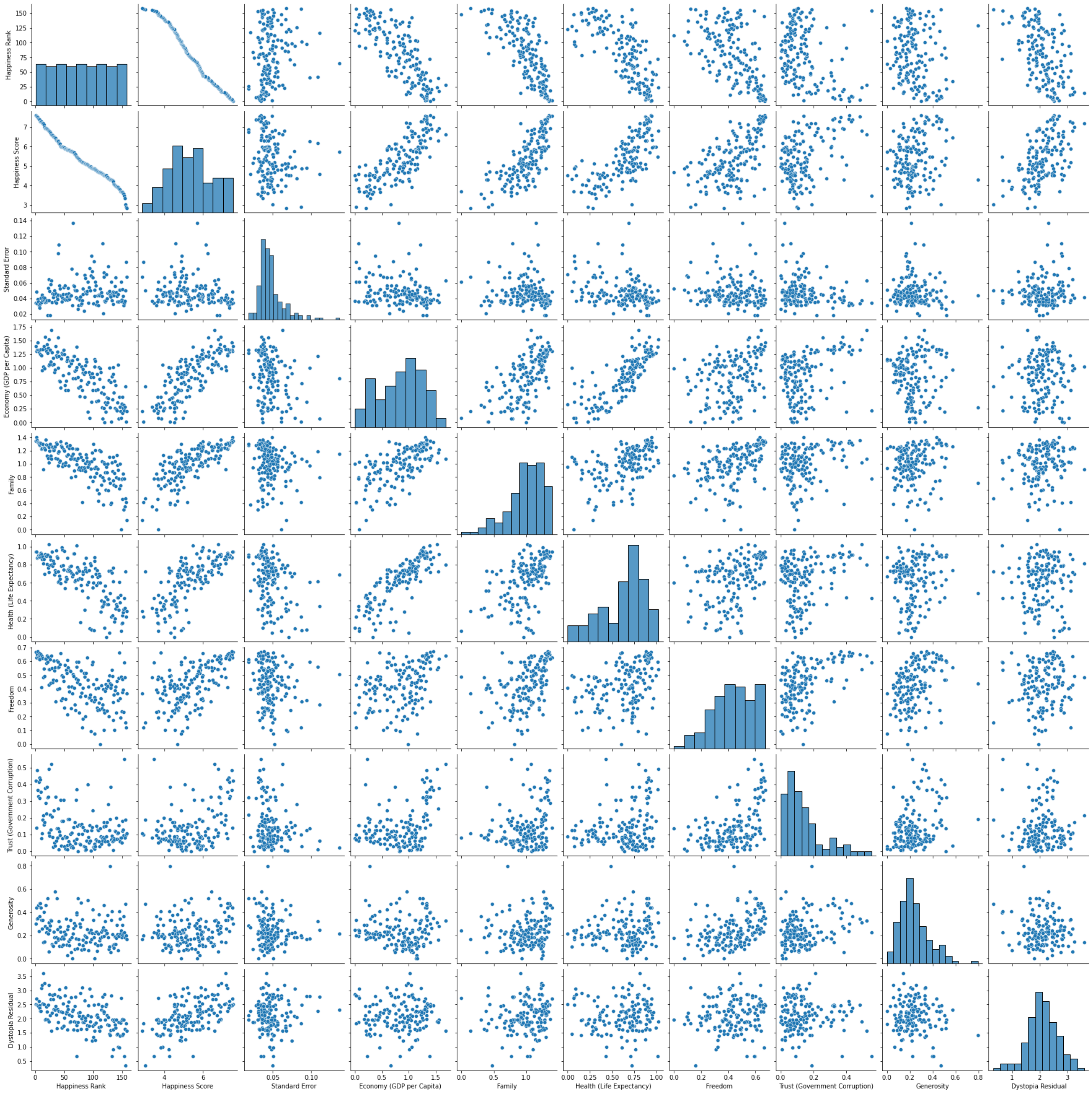
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1ed62df2f10>
```



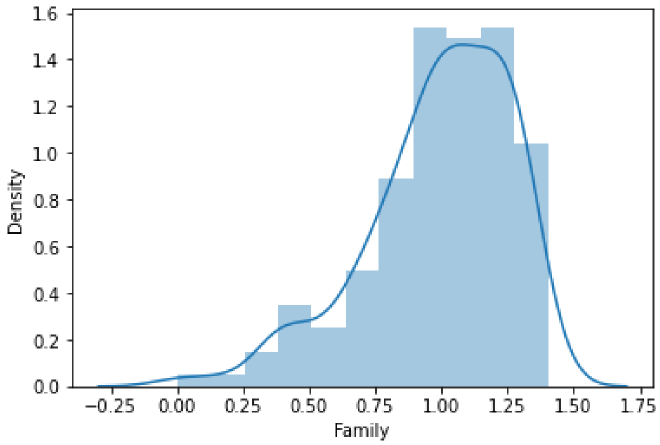
In [8]:

```
sns.distplot(df['Family'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `d
istplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `histp
lot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:

```
<AxesSubplot:xlabel='Family', ylabel='Density'>
```

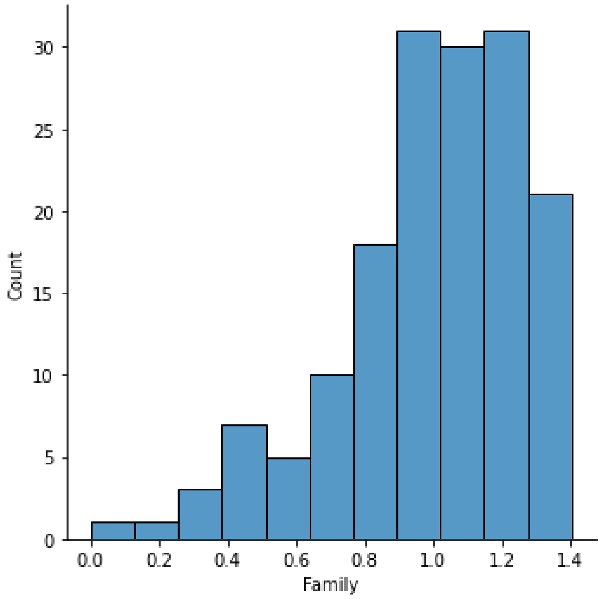


In [9]:

```
sns.displot(df["Family"])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1ed6820cf40>
```



In [10]:

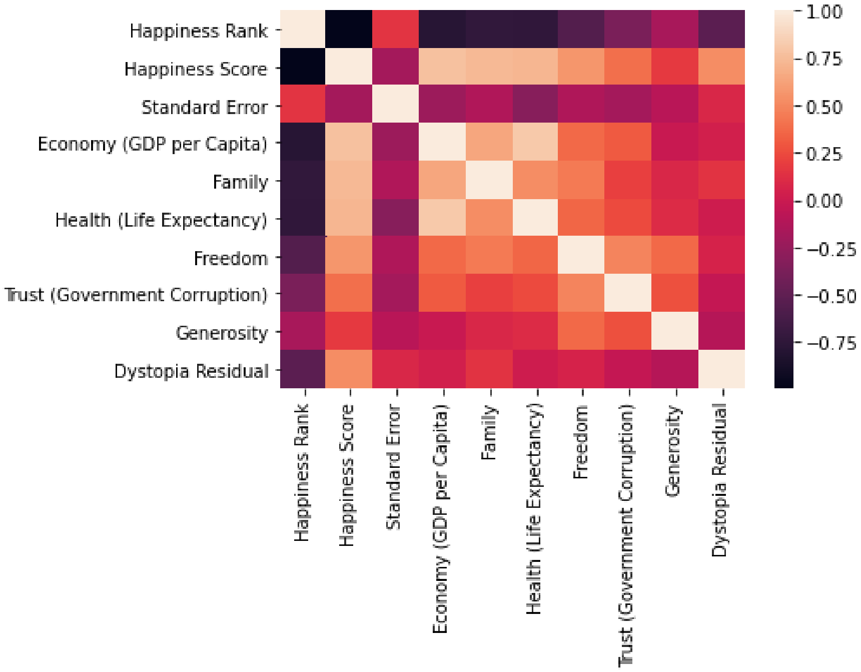
```
df1=df[['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual']]
```

In [11]:

```
sns.heatmap(df1.corr())
```

Out[11]:

```
<AxesSubplot:>
```



In [12]:

```
x=df1[['Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual']]
y=df1[['Family']]
```

In [13]:

```
from sklearn.model_selection import train_test_split
```

In [14]:

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

In [15]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for dtype('float64',
```

Out[15]:

```
LinearRegression()
```

In [16]:

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

```
[-0.00084756]
```

In [17]:

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

Out[17]:

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.000003 | 1.000055 | -0.00014 | -1.00001 | -0.999981 | -0.999632 | -0.999855 | -0.999738 | -0.999946 |

In [18]:

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

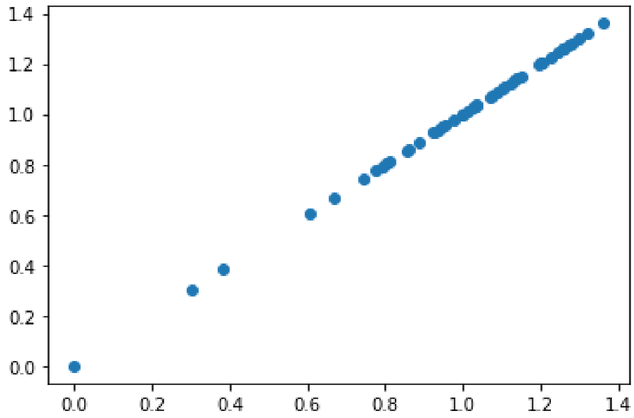
```
0.9999987483053541
```

In [19]:

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

Out[19]:

```
<matplotlib.collections.PathCollection at 0x1ed6988aee0>
```


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4