Load the dataset and inspect the first few rows

In [2]:
```
import pandas as pd
df = pd.read_csv("/home/mohammadseyfi/Desktop/deeplearning/WorldHappinessRep
```

In [4]:
```
df.head() # Display the first 5 rows by default
df.head(10) # Display the first 10 rows
```

Out[4]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Free
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.6
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.6
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.6
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.6
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.6
5	Finland	Western Europe	6	7.406	0.03140	1.29025	1.31826	0.88911	0.6
6	Netherlands	Western Europe	7	7.378	0.02799	1.32944	1.28017	0.89284	0.6
7	Sweden	Western Europe	8	7.364	0.03157	1.33171	1.28907	0.91087	0.6
8	New Zealand	Australia and New Zealand	9	7.286	0.03371	1.25018	1.31967	0.90837	0.6
9	Australia	Australia and New Zealand	10	7.284	0.04083	1.33358	1.30923	0.93156	0.6

In [7]:
```
df.shape
df.info
```

Out[7]:
```
<bound method DataFrame.info of          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 [8]:
```
df.dtypes
```

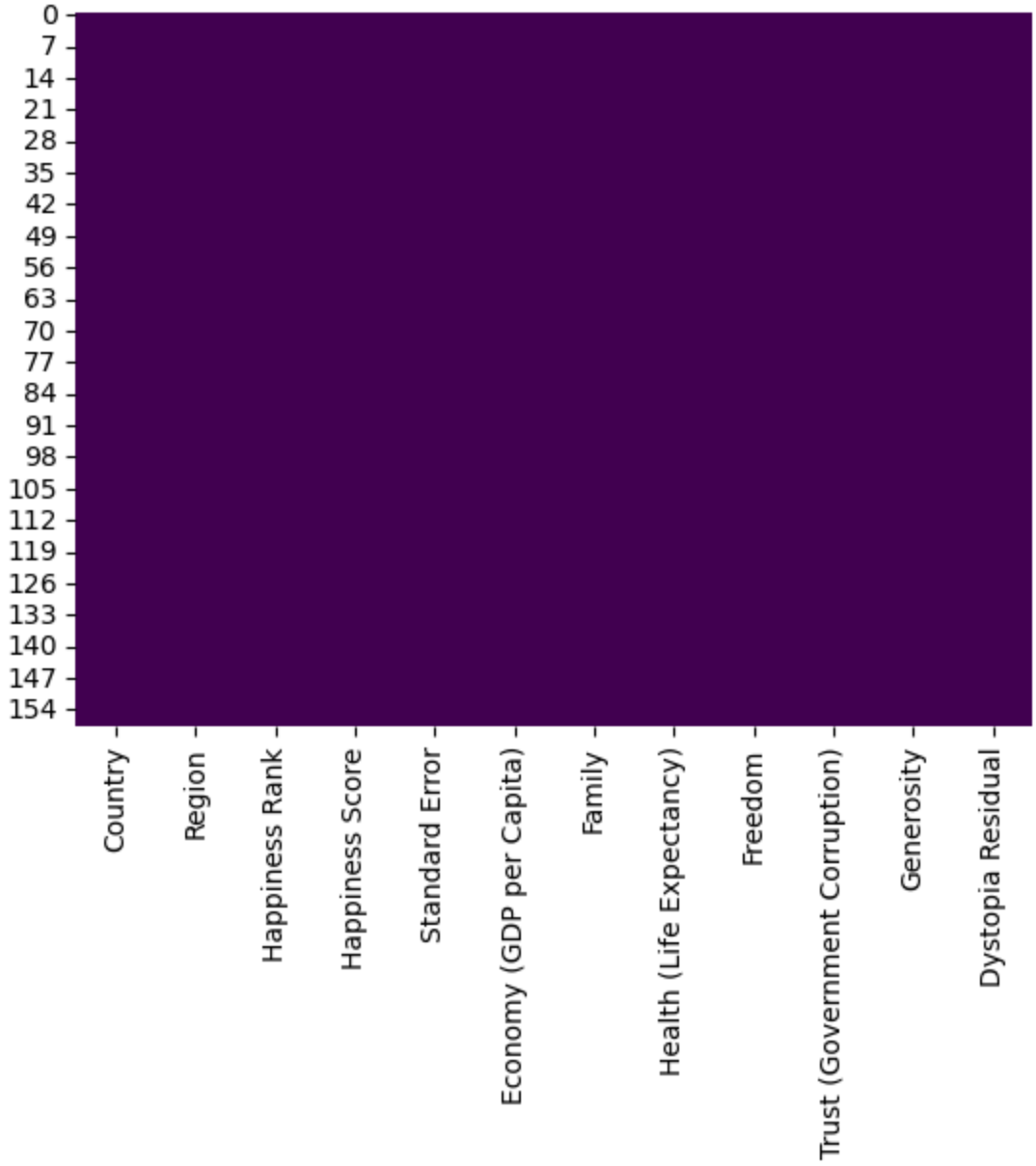
Out[8]:
```
Country                           object
Region                            object
Happiness Rank                     int64
Happiness Score                  float64
Standard Error                   float64
Economy (GDP per Capita)         float64
Family                           float64
Health (Life Expectancy)         float64
Freedom                          float64
Trust (Government Corruption)    float64
Generosity                       float64
Dystopia Residual                float64
dtype: object
```

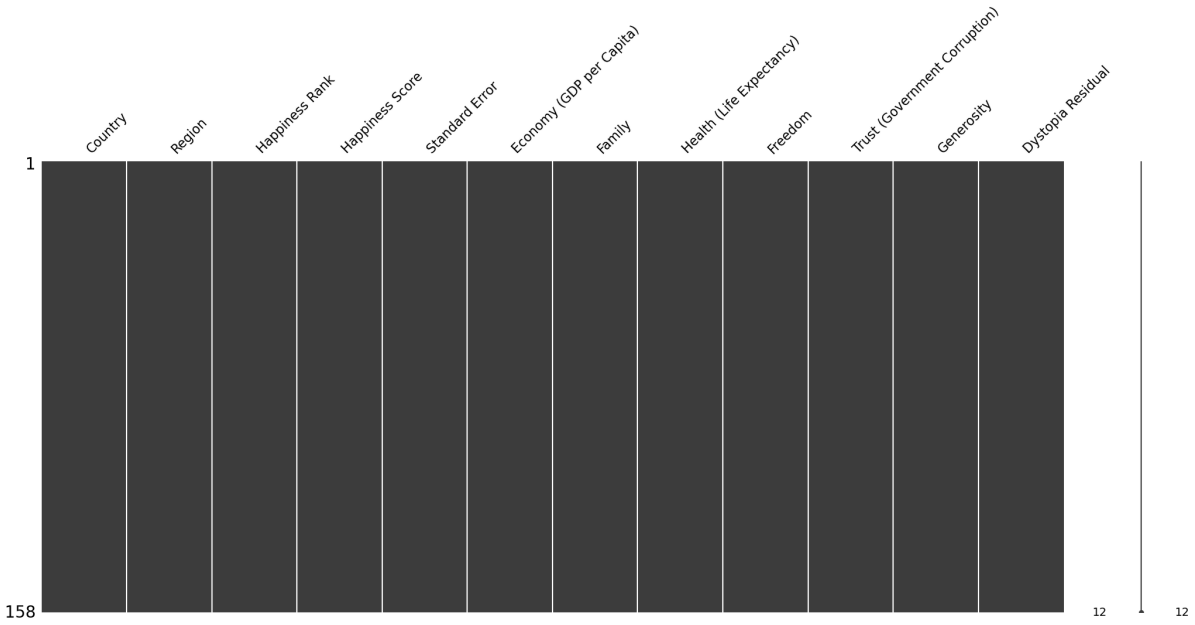
In [13]:
```
df.isnull().sum()  # Counts missing values per column
```

Out[13]:
```
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
```

In [16]:
```
# These are the two options that let's you VISUALIZE THE MISSING VALUES
import seaborn as sns
sns.heatmap(df.isnull(), cmap = "viridis", cbar= False)

import missingno as msno
msno.matrix(df)
```

Out[16]:
```
<Axes: >
```


0
7
14
21
28
35
42
49
56
63
70
77
84
91
98
105
112
119
126
133
140
147
154
Country
Region
Happiness Rank
Happiness Score
Standard Error
Economy (GDP per Capita)
Family
Health (Life Expectancy)
Freedom
Trust (Government Corruption)
Generosity
Dystopia Residual


Country
Region
Happiness Rank
Happiness Score
Standard Error
Economy (GDP per Capita)
Family
Health (Life Expectancy)
Freedom
Trust (Government Corruption)
Generosity
Dystopia Residual
1
158
12
12

In [19]:
```
df.duplicated().sum()
df[df.duplicated()]
```

Out[19]:

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

In [32]:
```
# Add a new 'Year' column
df['Year'] = 2015
df.head()
```

Out[32]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freed
0	Switzerland	Western Europe	1	1.937360	0.03411	1.39651	1.34951	0.94143	0.66
1	Iceland	Western Europe	2	1.914581	0.04884	1.30232	1.40223	0.94784	0.62
2	Denmark	Western Europe	3	1.884792	0.03328	1.32548	1.36058	0.87464	0.64
3	Norway	Western Europe	4	1.880411	0.03880	1.45900	1.33095	0.88521	0.66
4	Canada	North America	5	1.797179	0.03553	1.32629	1.32261	0.90563	0.63

In [24]:
```
# Standardize categorical variables like country names or regions
df["Country"].unique()
```

Out[24]:
```
array(['Switzerland', 'Iceland', 'Denmark', 'Norway', 'Canada', 'Finland',
       'Netherlands', 'Sweden', 'New Zealand', 'Australia', 'Israel',
       'Costa Rica', 'Austria', 'Mexico', 'United States', 'Brazil',
       'Luxembourg', 'Ireland', 'Belgium', 'United Arab Emirates',
       'United Kingdom', 'Oman', 'Venezuela', 'Singapore', 'Panama',
       'Germany', 'Chile', 'Qatar', 'France', 'Argentina',
       'Czech Republic', 'Uruguay', 'Colombia', 'Thailand',
       'Saudi Arabia', 'Spain', 'Malta', 'Taiwan', 'Kuwait', 'Suriname',
       'Trinidad and Tobago', 'El Salvador', 'Guatemala', 'Uzbekistan',
       'Slovakia', 'Japan', 'South Korea', 'Ecuador', 'Bahrain', 'Italy',
       'Bolivia', 'Moldova', 'Paraguay', 'Kazakhstan', 'Slovenia',
       'Lithuania', 'Nicaragua', 'Peru', 'Belarus', 'Poland', 'Malaysia',
       'Croatia', 'Libya', 'Russia', 'Jamaica', 'North Cyprus', 'Cyprus',
       'Algeria', 'Kosovo', 'Turkmenistan', 'Mauritius', 'Hong Kong',
       'Estonia', 'Indonesia', 'Vietnam', 'Turkey', 'Kyrgyzstan',
       'Nigeria', 'Bhutan', 'Azerbaijan', 'Pakistan', 'Jordan',
       'Montenegro', 'China', 'Zambia', 'Romania', 'Serbia', 'Portugal',
       'Latvia', 'Philippines', 'Somaliland region', 'Morocco',
       'Macedonia', 'Mozambique', 'Albania', 'Bosnia and Herzegovina',
       'Lesotho', 'Dominican Republic', 'Laos', 'Mongolia', 'Swaziland',
       'Greece', 'Lebanon', 'Hungary', 'Honduras', 'Tajikistan',
       'Tunisia', 'Palestinian Territories', 'Bangladesh', 'Iran',
       'Ukraine', 'Iraq', 'South Africa', 'Ghana', 'Zimbabwe', 'Liberia',
       'India', 'Sudan', 'Haiti', 'Congo (Kinshasa)', 'Nepal', 'Ethiopia',
       'Sierra Leone', 'Mauritania', 'Kenya', 'Djibouti', 'Armenia',
       'Botswana', 'Myanmar', 'Georgia', 'Malawi', 'Sri Lanka',
       'Cameroon', 'Bulgaria', 'Egypt', 'Yemen', 'Angola', 'Mali',
       'Congo (Brazzaville)', 'Comoros', 'Uganda', 'Senegal', 'Gabon',
       'Niger', 'Cambodia', 'Tanzania', 'Madagascar',
       'Central African Republic', 'Chad', 'Guinea', 'Ivory Coast',
       'Burkina Faso', 'Afghanistan', 'Rwanda', 'Benin', 'Syria',
       'Burundi', 'Togo'], dtype=object)
```

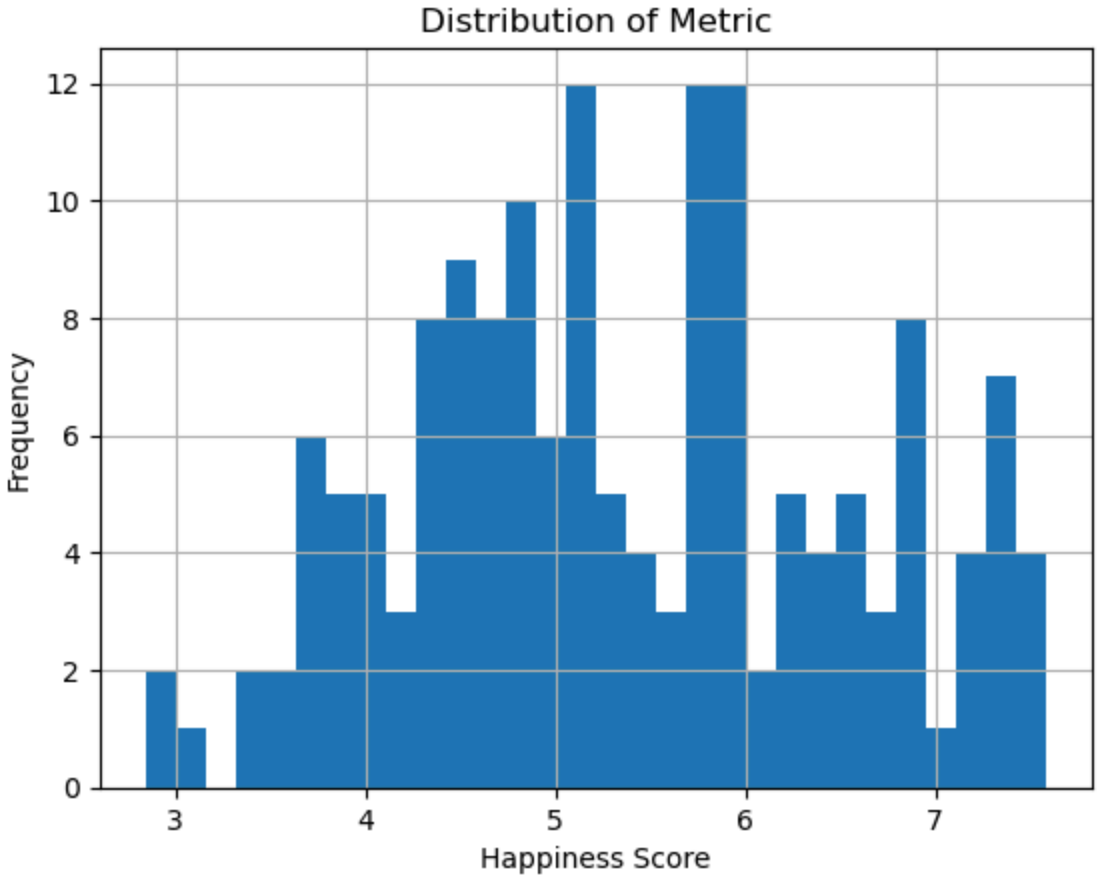
In [25]:
```
# Understanding basic statistics about the data
df.describe()
```

Out[25]:

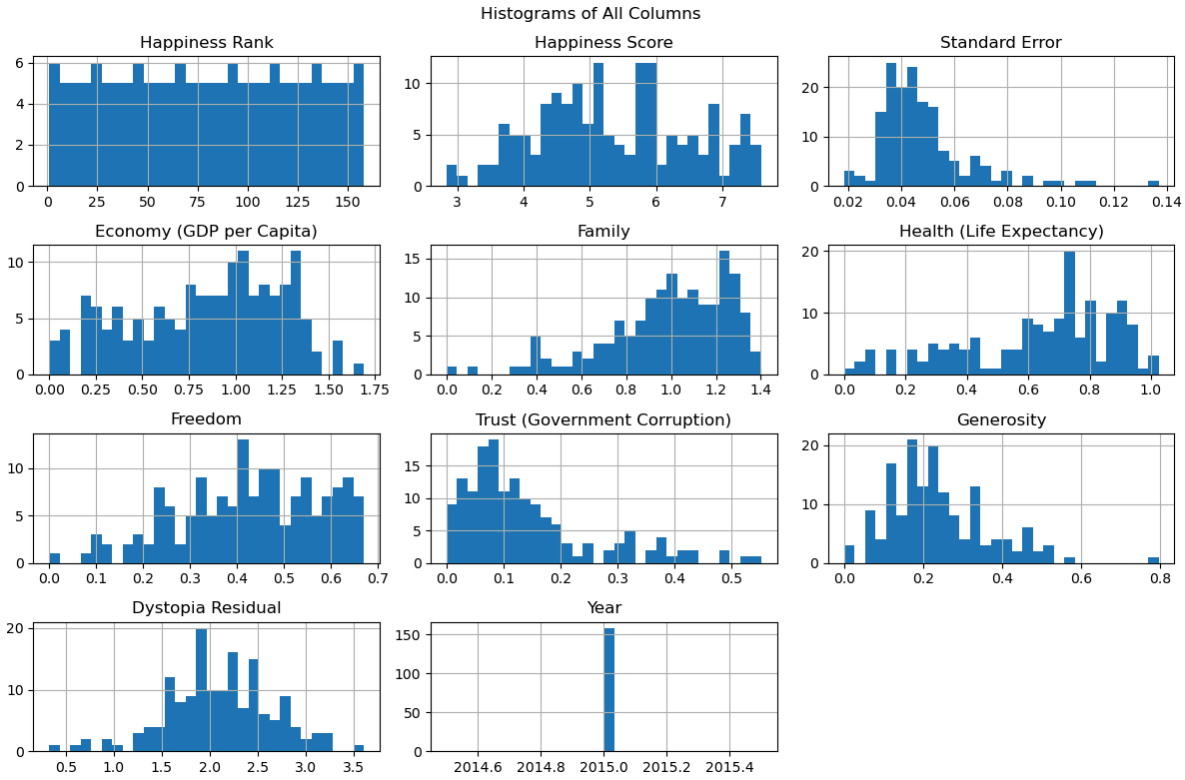
	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(G
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	
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 [29]:
```
import matplotlib.pyplot as plt
df['Happiness Score'].hist(bins=30)
```

```
plt.xlabel("Happiness Score")
plt.ylabel("Frequency")
plt.title("Distribution of Metric")
plt.show()
```



In [31]:
```
df.hist(figsize=(12, 8), bins=30)
plt.suptitle("Histograms of All Columns")
plt.tight_layout()
plt.show()
```

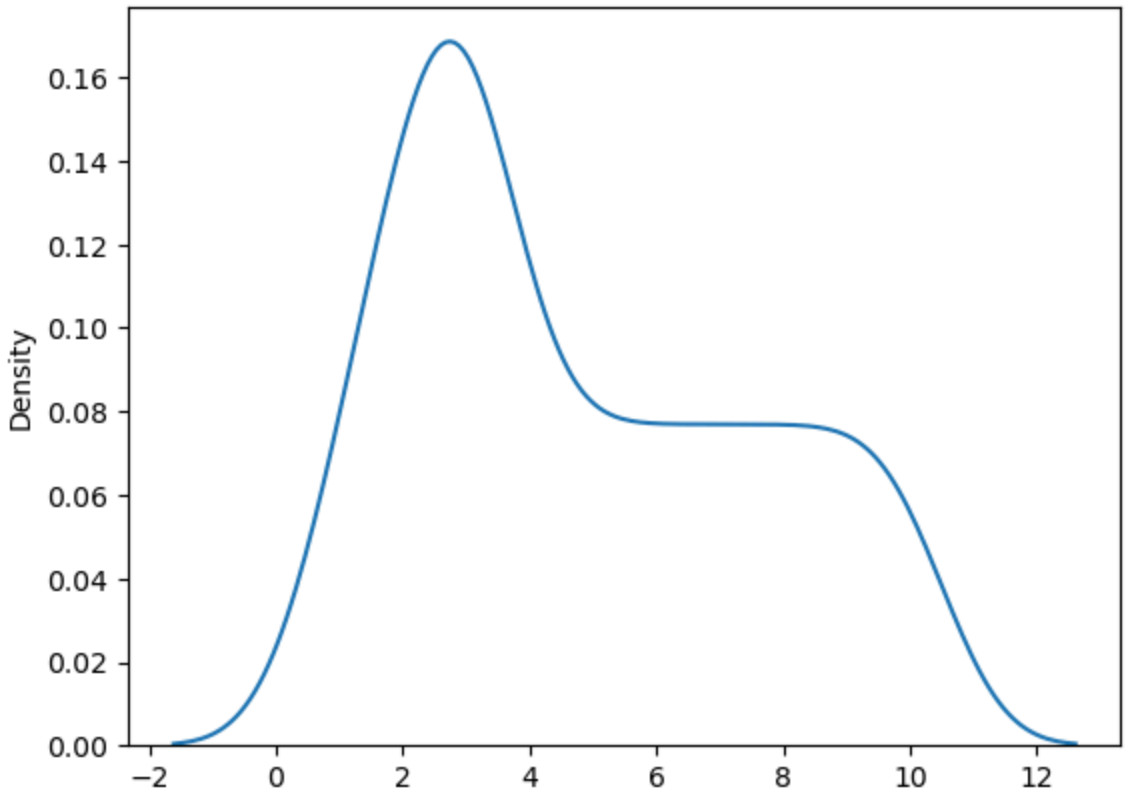


Kernel Density Estimation (KDE) is a non-parametric method used to estimate the probability density function (PDF) of a dataset. It helps visualize the distribution of data points in a smooth and continuous way, unlike histograms, which rely on discrete bins.

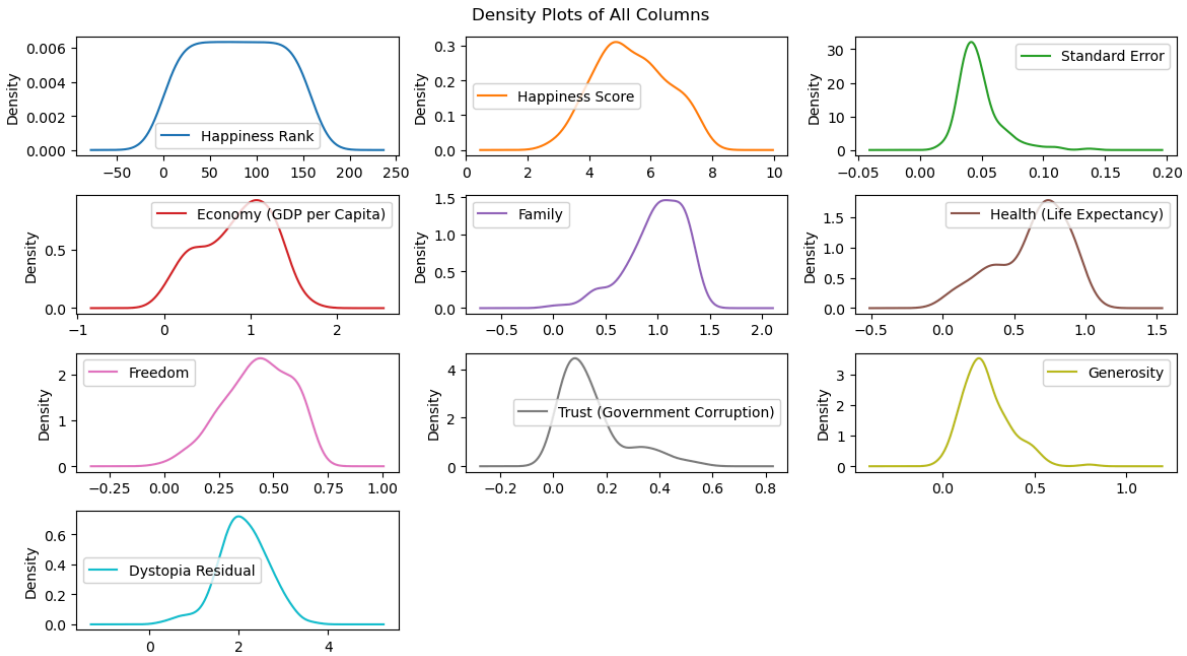
KDE places a kernel function (such as Gaussian) on each data point and sums them up to create a smooth curve. The key parameter in KDE is the bandwidth, which controls how much smoothing is applied

For Example:

In [32]:
```
data = [1, 2, 2, 3, 3, 3, 4, 5, 6, 7, 8, 9, 10]
sns.kdeplot(data, bw_adjust=0.5)  # Adjust bandwidth for smoothing
plt.show()
```



In [7]:
```
import matplotlib.pyplot as plt

num_columns = len(df.columns)
df.plot(kind="density", subplots=True, layout=(num_columns // 3 + 1, 3), fig
plt.suptitle("Density Plots of All Columns")
plt.tight_layout()
plt.show()
```



Looking for outliers in numerical features :

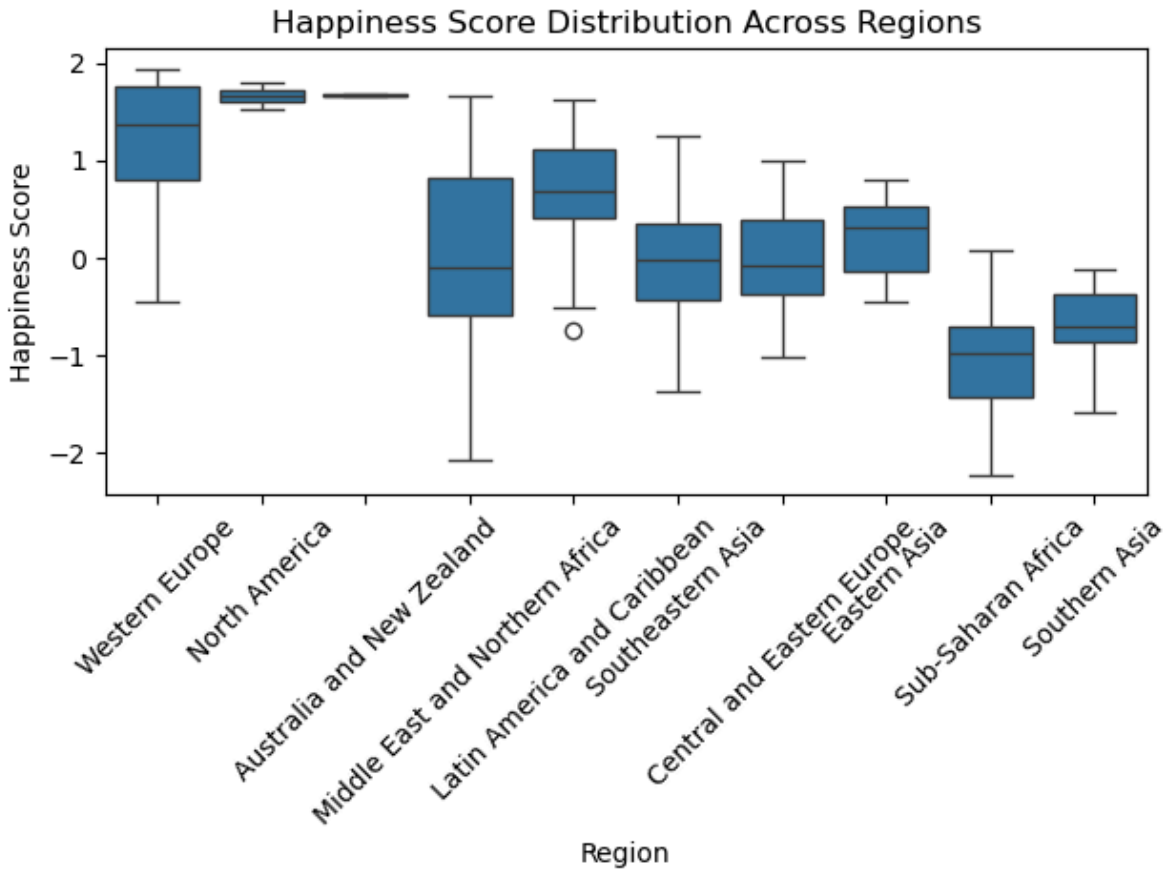
In [14]:
```
from scipy.stats import zscore

df["Happiness Score"] = zscore(df["Happiness Score"])
outliers = df[df["Happiness Score"].abs() > 3]  # Threshold of 3 standard de
print(outliers)
```

```
Empty DataFrame
Columns: [Country, Region, Happiness Rank, Happiness Score, Standard Error,
Economy (GDP per Capita), Family, Health (Life Expectancy), Freedom, Trust
(Government Corruption), Generosity, Dystopia Residual]
Index: []
```

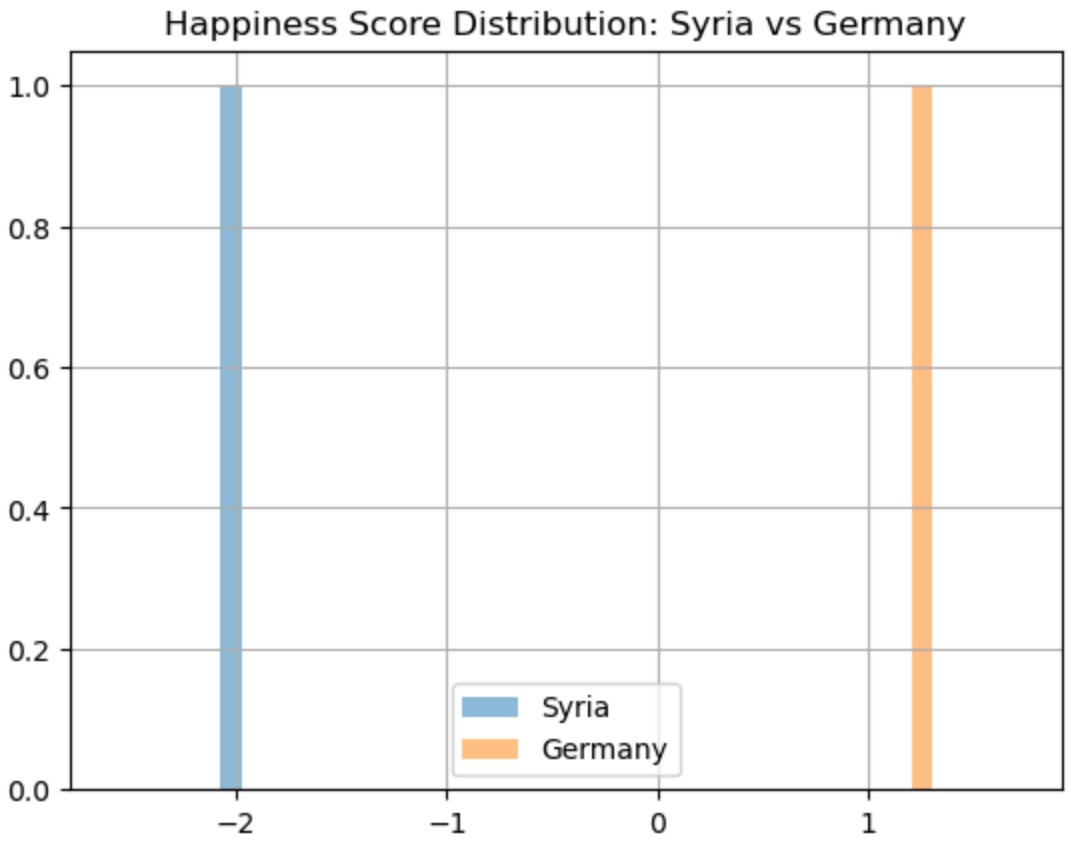
In [15]:
```
import numpy as np

Q1 = df["Happiness Score"].quantile(0.25)
Q3 = df["Happiness Score"].quantile(0.75)
IQR = Q3 - Q1

outliers = df[(df["Happiness Score"] < Q1 - 1.5 * IQR) | (df["Happiness Scor
print(outliers)
```

```
Empty DataFrame
Columns: [Country, Region, Happiness Rank, Happiness Score, Standard Error,
Economy (GDP per Capita), Family, Health (Life Expectancy), Freedom, Trust
(Government Corruption), Generosity, Dystopia Residual]
Index: []
```

In [19]:
```
import seaborn as sns

sns.boxplot(x="Region", y="Happiness Score", data=df)
plt.xticks(rotation=45)  # Rotate labels for readability
plt.title("Happiness Score Distribution Across Regions")
plt.tight_layout()
plt.show()
```

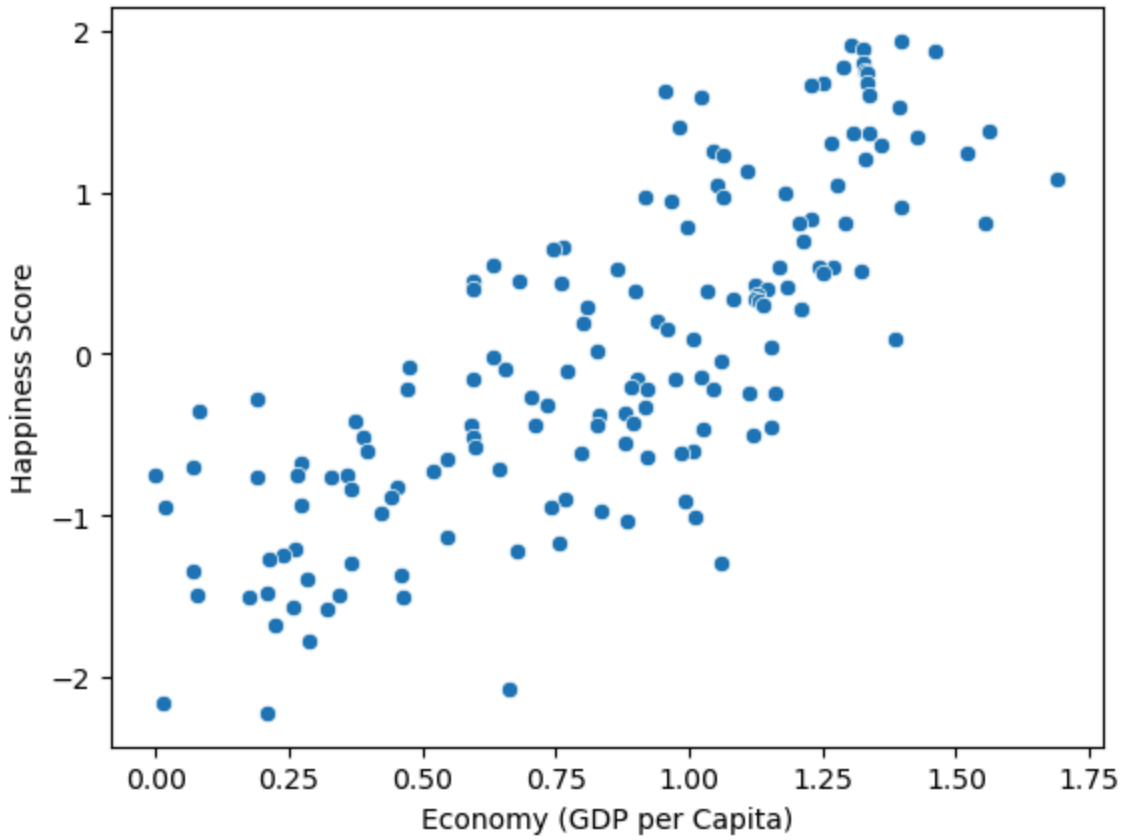


In [25]:
```
df[df["Country"] == "Syria"]["Happiness Score"].hist(alpha=0.5, label="Syria
df[df["Country"] == "Germany"]["Happiness Score"].hist(alpha=0.5, label="Ger
plt.legend()
plt.title("Happiness Score Distribution: Syria vs Germany")
plt.show()
```

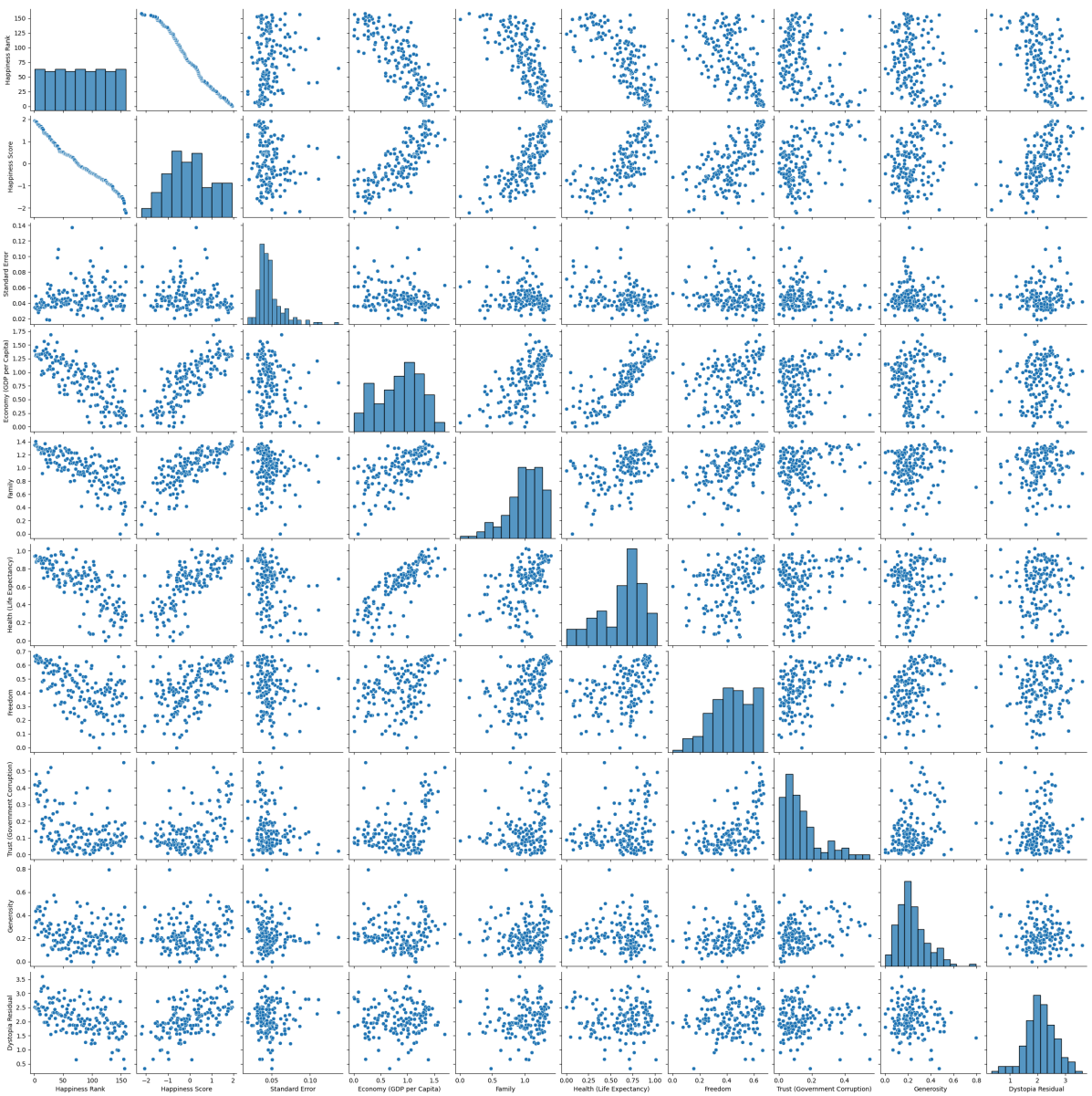


In [29]:
```
sns.scatterplot(x=df["Economy (GDP per Capita)"], y=df["Happiness Score"])
```

Out[29]:
```
<Axes: xlabel='Economy (GDP per Capita)', ylabel='Happiness Score'>
```



In [30]:
```
sns.pairplot(df)  # Automatically creates scatter plots for all numerical va
plt.show()
```



In [3]:
```
df_2015 = pd.read_csv("/home/mohammadseyfi/Desktop/deeplearning/WorldHappine
print("2015 Columns:", df_2015.columns.tolist())
print("============================================================")
df_2016 = pd.read_csv("/home/mohammadseyfi/Desktop/deeplearning/WorldHappine
print("2016 Columns:", df_2016.columns.tolist())
print("============================================================")
df_2017 = pd.read_csv("/home/mohammadseyfi/Desktop/deeplearning/WorldHappine
print("2017 Columns:", df_2017.columns.tolist())
print("============================================================")
df_2018 = pd.read_csv("/home/mohammadseyfi/Desktop/deeplearning/WorldHappine
print("2018 Columns:", df_2018.columns.tolist())
print("============================================================")
df_2019 = pd.read_csv("/home/mohammadseyfi/Desktop/deeplearning/WorldHappine
print("2019 Columns:", df_2019.columns.tolist())
```

```
2015 Columns: ['Country', 'Region', 'Happiness Rank', 'Happiness Score', 'St
andard Error', 'Economy (GDP per Capita)', 'Family', 'Health (Life Expectanc
y)', 'Freedom', 'Trust (Government Corruption)', 'Generosity', 'Dystopia Res
idual']
================================================================
2016 Columns: ['Country', 'Region', 'Happiness Rank', 'Happiness Score', 'Lo
wer Confidence Interval', 'Upper Confidence Interval', 'Economy (GDP per Cap
ita)', 'Family', 'Health (Life Expectancy)', 'Freedom', 'Trust (Government C
orruption)', 'Generosity', 'Dystopia Residual']
================================================================
2017 Columns: ['Country', 'Happiness Rank', 'Happiness Score', 'Whisker.hig
h', 'Whisker.low', 'Economy (GDP per Capita)', 'Family', 'Health (Life Expec
tancy)', 'Freedom', 'Generosity', 'Trust (Government Corruption)', 'Dystopia
Residual']
================================================================
2018 Columns: ['Happiness Rank', 'Country', 'Happiness Score', 'Economy (GDP
per Capita)', 'Family', 'Health (Life Expectancy)', 'Freedom', 'Generosity',
'Trust (Government Corruption)']
================================================================
2019 Columns: ['Happiness Rank', 'Country', 'Happiness Score', 'Economy (GDP
per Capita)', 'Family', 'Health (Life Expectancy)', 'Freedom', 'Generosity',
'Trust (Government Corruption)']
```

In [73]:
```
columns_to_keep = ['Country', 'Happiness Score', 'Economy (GDP per Capita)',
                   'Generosity','Freedom', 'Health (Life Expectancy)', 'Happ
df_2015 = df_2015[columns_to_keep]
df_2016 = df_2015[columns_to_keep]
df_2017 = df_2015[columns_to_keep]
df_2018 = df_2015[columns_to_keep]
df_2019 = df_2015[columns_to_keep]
```

In [55]:
```
print("2015 Columns:", df_2015.columns.tolist())
```

```
2015 Columns: ['Country', 'Happiness Score', 'Economy (GDP per Capita)', 'Fa
mily', 'Trust (Government Corruption)', 'Generosity', 'Freedom', 'Health (Li
fe Expectancy)', 'Happiness Rank']
```

In [74]:
```
df_2015['Year'] = 2015
df_2016['Year'] = 2016
df_2017['Year'] = 2017
df_2018['Year'] = 2018
df_2019['Year'] = 2019
```

In [57]:
```
print("2015 Columns:", df_2015.columns.tolist())
```

```
2015 Columns: ['Country', 'Happiness Score', 'Economy (GDP per Capita)', 'Fa
mily', 'Trust (Government Corruption)', 'Generosity', 'Freedom', 'Health (Li
fe Expectancy)', 'Happiness Rank', 'Year']
```

In [58]:
```
combined_df = pd.concat([df_2015, df_2016, df_2017, df_2018, df_2019], ignor
```

In [59]:
```
print(combined_df.shape)           # How many rows and columns?
print(combined_df.columns.tolist())# Column names
print(combined_df.dtypes)          # Data types
print(combined_df.isnull().sum())  # Missing values
print(combined_df.describe())      # Summary stats
```

```
(790, 10)
['Country', 'Happiness Score', 'Economy (GDP per Capita)', 'Family', 'Trust
(Government Corruption)', 'Generosity', 'Freedom', 'Health (Life Expectanc
y)', 'Happiness Rank', 'Year']
Country                          object
Happiness Score                 float64
Economy (GDP per Capita)        float64
Family                          float64
Trust (Government Corruption)   float64
Generosity                      float64
Freedom                         float64
Health (Life Expectancy)        float64
Happiness Rank                    int64
Year                              int64
dtype: object
Country                         0
Happiness Score                 0
Economy (GDP per Capita)        0
Family                          0
Trust (Government Corruption)   0
Generosity                      0
Freedom                         0
Health (Life Expectancy)        0
Happiness Rank                  0
Year                            0
dtype: int64
       Happiness Score  Economy (GDP per Capita)      Family  \
count       790.000000                790.000000  790.000000
mean          5.375734                  0.846137    0.991046
std           1.142104                  0.402098    0.271678
min           2.839000                  0.000000    0.000000
25%           4.518000                  0.545580    0.855630
50%           5.232500                  0.910245    1.029510
75%           6.269000                  1.159910    1.216240
max           7.587000                  1.690420    1.402230

       Trust (Government Corruption)  Generosity     Freedom  \
count                     790.000000  790.000000  790.000000
mean                        0.143422    0.237296    0.428615
std                         0.119729    0.126363    0.150310
min                         0.000000    0.000000    0.000000
25%                         0.061460    0.149820    0.328180
50%                         0.107220    0.216130    0.435515
75%                         0.180600    0.311050    0.550110
max                         0.551910    0.795880    0.669730

       Health (Life Expectancy)  Happiness Rank         Year
count                790.000000      790.000000   790.000000
mean                   0.630259       79.493671  2017.000000
std                    0.246451       45.638235     1.415109
min                    0.000000        1.000000  2015.000000
25%                    0.438730       40.000000  2016.000000
50%                    0.696705       79.500000  2017.000000
75%                    0.811600      119.000000  2018.000000
max                    1.025250      158.000000  2019.000000
```

In [60]:
```
print(df_2015.shape)
```

```
(158, 10)
```

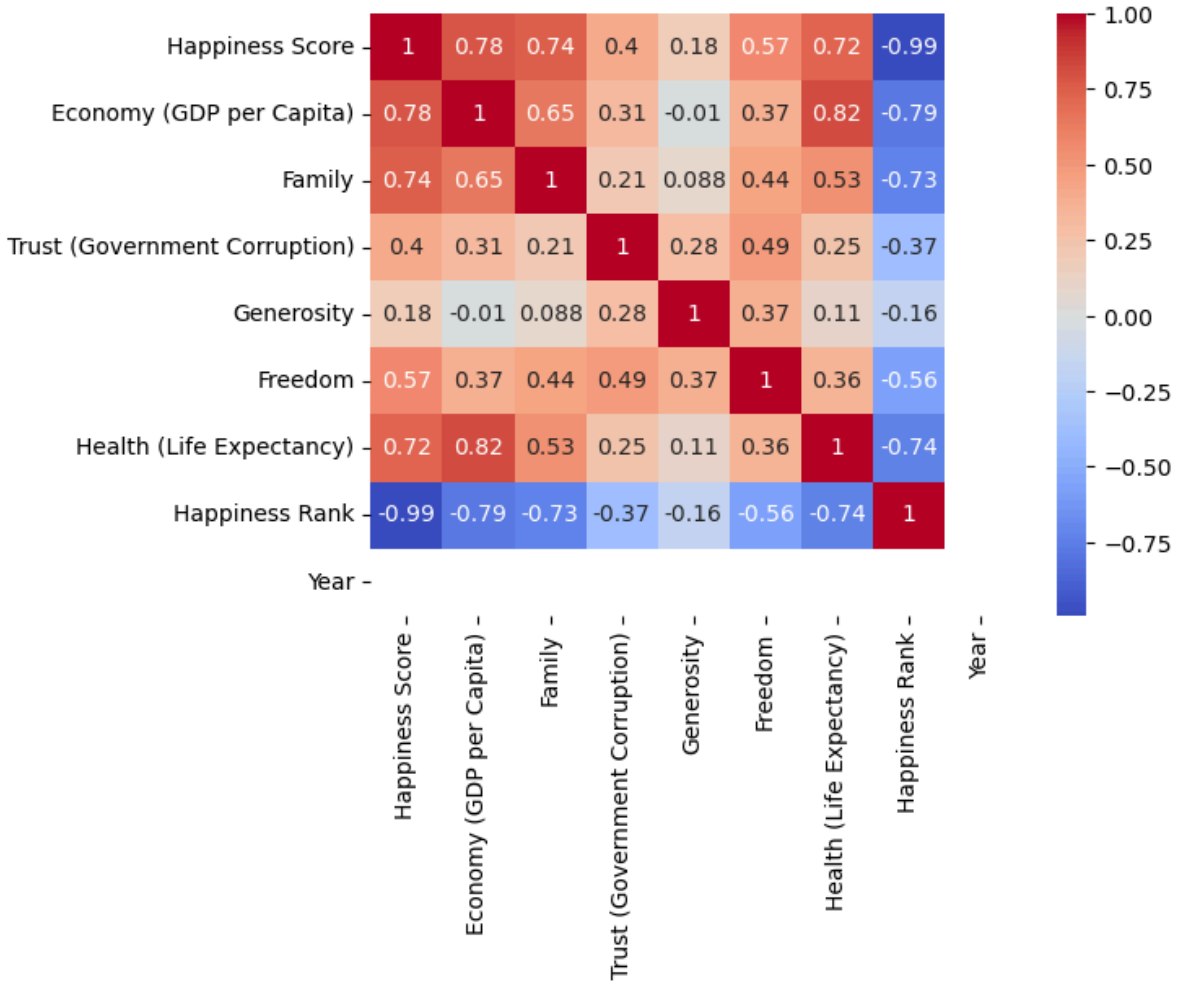
In [63]:
```
combined_df.loc[157:160]
```

Out[63]:

	Country	Happiness Score	Economy (GDP per Capita)	Family	Trust (Government Corruption)	Generosity	Freedom	Health (Life Expectancy)
157	Togo	2.839	0.20868	0.13995	0.10731	0.16681	0.36453	0.28443
158	Switzerland	7.587	1.39651	1.34951	0.41978	0.29678	0.66557	0.94143
159	Iceland	7.561	1.30232	1.40223	0.14145	0.43630	0.62877	0.94784
160	Denmark	7.527	1.32548	1.36058	0.48357	0.34139	0.64938	0.87464

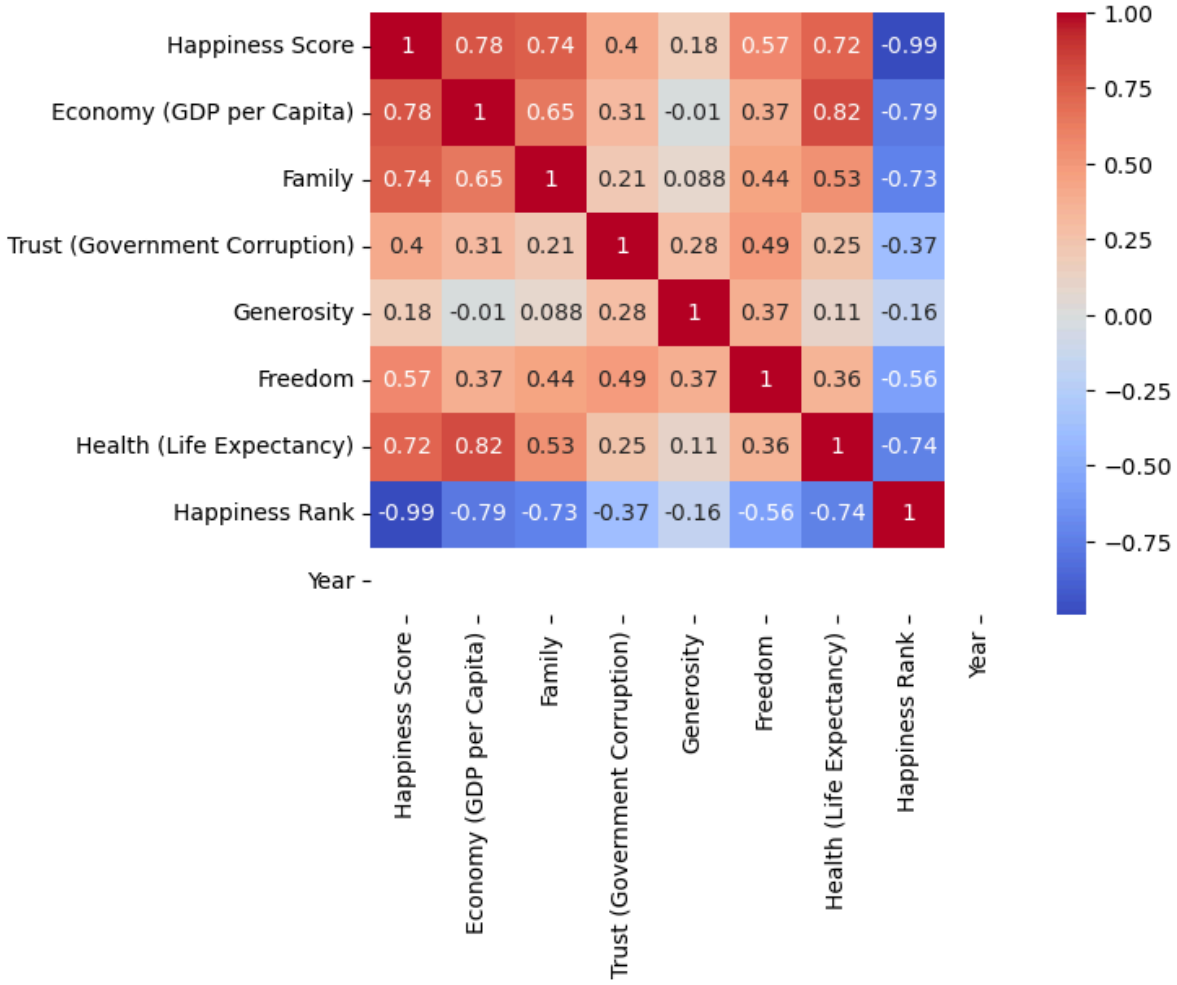
In [75]:
```
corr = df_2015.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

Out[75]: <Axes: >



In [76]:
```
corr = df_2016.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

Out[76]: <Axes: >



In [9]:
```
df_2019.columns = df_2019.columns.str.strip().str.lower().str.replace(' ', '

df_2019.head()
```

Out[9]:

	happiness_rank	country	happiness_score	economy_(gdp_per_capita)	family	health_(life_
0	1	Finland	7.769	1.340	1.587	
1	2	Denmark	7.600	1.383	1.573	
2	3	Norway	7.554	1.488	1.582	
3	4	Iceland	7.494	1.380	1.624	
4	5	Netherlands	7.488	1.396	1.522	

Let's Train two Machine Learning model on CSV.2019 :

In [17]:
```
features = df_2019.columns.tolist()
features = ['economy_(gdp_per_capita)', 'family', 'health_(life_expectancy)'
            'freedom', 'generosity', 'trust_(government_corruption)']
target = 'happiness_score'
print(features)
```

```
['economy_(gdp_per_capita)', 'family', 'health_(life_expectancy)', 'freedo
m', 'generosity', 'trust_(government_corruption)']
```

In [18]:
```
X = df_2019[features]
y = df_2019[target]
```

In [19]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

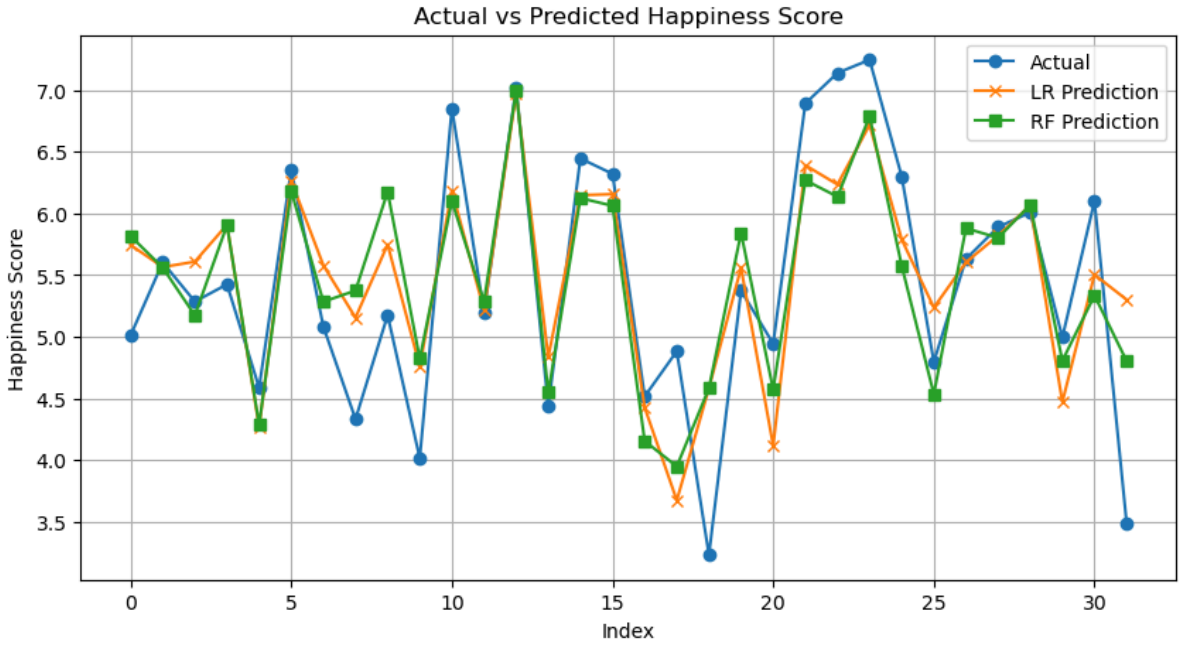
In [20]:
```
from sklearn.linear_model import LinearRegression

linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

linear_preds = linear_model.predict(X_test)
```

In [21]:
```
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(random_state= 42)
rf_model.fit(X_train, y_train)

rf_preds= rf_model.predict(X_test)
```

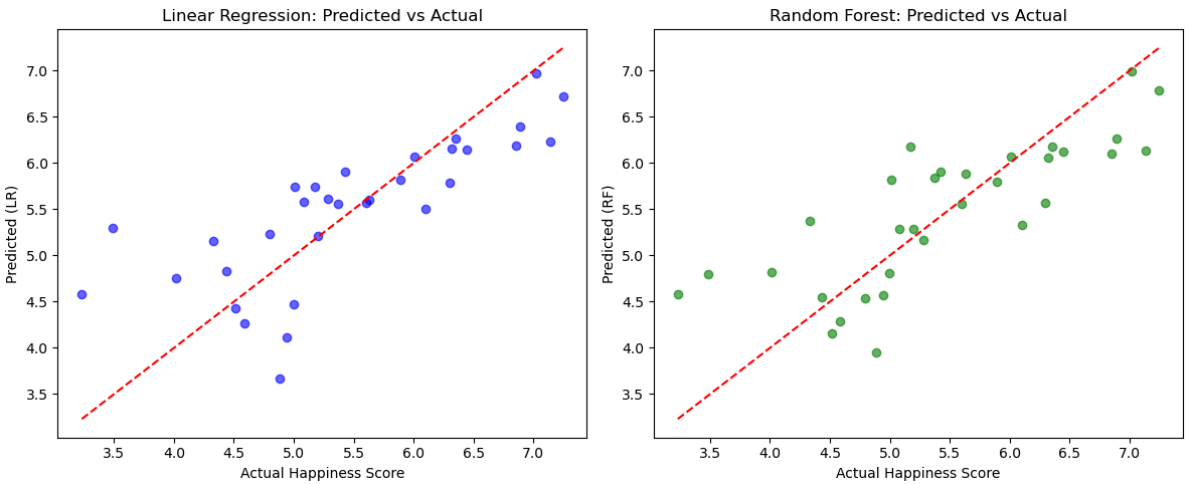
In [24]:
```
from sklearn.metrics import mean_squared_error, r2_score

def evaluate(model_name, y_test, y_pred):
    print(f"=== {model_name} ===")
    print("MSE :", mean_squared_error(y_test, y_pred))
    print("R2  :", r2_score(y_test, y_pred))
    print()

evaluate("Linear Regression", y_test, linear_preds)
evaluate("Random Forest", y_test, rf_preds)
```

```
=== Linear Regression ===
MSE : 0.41446413835283524
R2  : 0.6017537913445683

=== Random Forest ===
MSE : 0.3863682043125004
R2  : 0.6287503350133744
```

In [27]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
plt.plot(y_test.values, label='Actual', marker='o')
plt.plot(linear_preds, label='LR Prediction', marker='x')
plt.plot(rf_preds, label='RF Prediction', marker='s')
plt.legend()
plt.title("Actual vs Predicted Happiness Score")
plt.xlabel("Index")
plt.ylabel("Happiness Score")
plt.grid()
plt.show()
```



In [29]:
```
# Scatter plot Linear Regression
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_test, linear_preds, color='blue', alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel('Actual Happiness Score')
plt.ylabel('Predicted (LR)')
plt.title('Linear Regression: Predicted vs Actual')

# Scatter plot Random Forest
plt.subplot(1, 2, 2)
plt.scatter(y_test, rf_preds, color='green', alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel('Actual Happiness Score')
plt.ylabel('Predicted (RF)')
plt.title('Random Forest: Predicted vs Actual')

plt.tight_layout()
plt.show()
```



In [ ]: