In [1]:
```
import pandas as pd
import seaborn as sns
import pprint
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_excel('./DataForTable2.1WHR2023.xls');
```

In [3]:
```
null_val = df.isnull().sum()
print(null_val)
```

```
Country name                         0
year                                 0
Life Ladder                          0
Log GDP per capita                  20
Social support                      13
Healthy life expectancy at birth    54
Freedom to make life choices        33
Generosity                          73
Perceptions of corruption          116
Positive affect                     24
Negative affect                     16
dtype: int64
```

Q1

- What is your evaluation of the quality of the data?

> According to the statistical appendix, The World Happiness Report 2023 evaluates the quality of life across different countries. The data is primarily sourced from The Gallup World Poll. Respondents ranked their own current lives on a scale from 0 to 10, with 10 being the best conceivable life and 0 being the worst. These life evaluation findings are correlated in the report with a number of other life characteristics, including GDP per capita, social support, healthy life expectancy, freedom to make decisions in life, generosity, corruption etc.
> Not all the countries appear in all the years. This can be seen in the dataset, for example, Albania has records ranging from years 2007 - 2022 whereas Algeria has records ranging from 2011 - 2021.
> To enhance the utility of the dataset, it would be beneficial to categorize countries according to their respective regions, such as North America, Europe, Asia, etc. Currently, the countries are listed without any regional classification, necessitating manual categorization for each country.
> There are missing values in the dataset which might need to be addressed by imputation. Perceptions of corruption is the most prominent and has the highest number (116) of missing values.

> The numeric features in the dataset exhibit potential outliers as shown by further analysis below.

- How large is the dataset?

In [4]:
```
df.shape
```

Out[4]:
```
(2199, 11)
```

> The dataframe has 2199 rows and 11 columns

- What are the features (columns) and how are they represented?

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Country name', 'year', 'Life Ladder', 'Log GDP per capita',
       'Social support', 'Healthy life expectancy at birth',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption', 'Positive affect', 'Negative affect'],
      dtype='object')
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2199 entries, 0 to 2198
Data columns (total 11 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Country name                      2199 non-null   object 
 1   year                              2199 non-null   int64  
 2   Life Ladder                       2199 non-null   float64
 3   Log GDP per capita                2179 non-null   float64
 4   Social support                    2186 non-null   float64
 5   Healthy life expectancy at birth  2145 non-null   float64
 6   Freedom to make life choices      2166 non-null   float64
 7   Generosity                        2126 non-null   float64
 8   Perceptions of corruption         2083 non-null   float64
 9   Positive affect                   2175 non-null   float64
 10  Negative affect                   2183 non-null   float64
dtypes: float64(9), int64(1), object(1)
memory usage: 189.1+ KB
```

> Each of these columns, except for Country name stores numerical data with integers for Year and floating point numbers for the others. The Country name column stores textual data.

- Are there any interesting distributions in the (numeric) features?

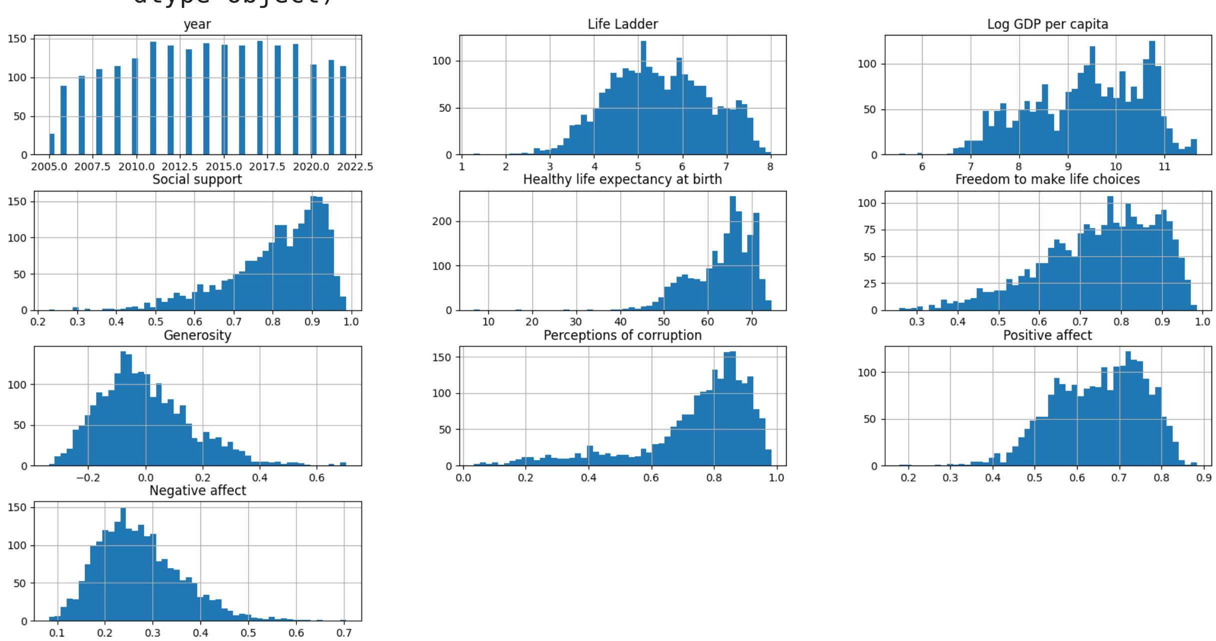
In [7]:
```python
df.describe()
```

Out[7]:

	year	Life Ladder	Log GDP per capita	Social support	Healthy life expectancy at birth	Freedom make li choic
count	2199.000000	2199.000000	2179.000000	2186.000000	2145.000000	2166.00000
mean	2014.161437	5.479226	9.389766	0.810679	63.294583	0.7478
std	4.718736	1.125529	1.153387	0.120952	6.901104	0.1401
min	2005.000000	1.281271	5.526723	0.228217	6.720000	0.2575
25%	2010.000000	4.646750	8.499764	0.746609	59.119999	0.6565
50%	2014.000000	5.432437	9.498955	0.835535	65.050003	0.7698
75%	2018.000000	6.309460	10.373216	0.904792	68.500000	0.8593
max	2022.000000	8.018934	11.663788	0.987343	74.474998	0.9851

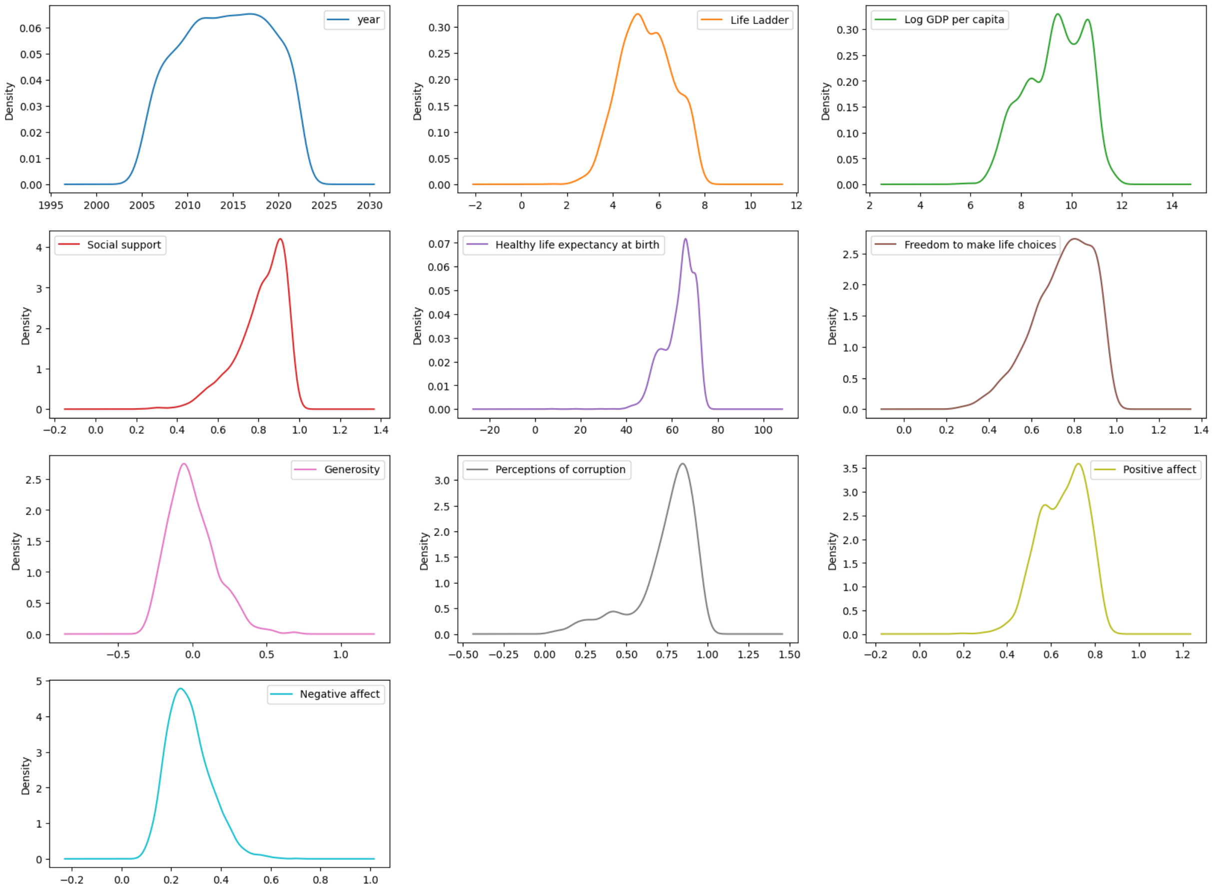
In [8]:
```python
df.hist(bins = 50, figsize = (20,10))
```

Out[8]:
```
array([[<Axes: title={'center': 'year'}>,
        <Axes: title={'center': 'Life Ladder'}>,
        <Axes: title={'center': 'Log GDP per capita'}>],
       [<Axes: title={'center': 'Social support'}>,
        <Axes: title={'center': 'Healthy life expectancy at birth'}>,
        <Axes: title={'center': 'Freedom to make life choices'}>],
       [<Axes: title={'center': 'Generosity'}>,
        <Axes: title={'center': 'Perceptions of corruption'}>,
        <Axes: title={'center': 'Positive affect'}>],
       [<Axes: title={'center': 'Negative affect'}>, <Axes: >, <Axes: >]],
      dtype=object)
```



In [9]:
```python
df.plot(kind = 'density', subplots = True, layout = (4,3), sharex = False, f
```

Out[9]:
```
array([[<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>]], dtype=object)
```

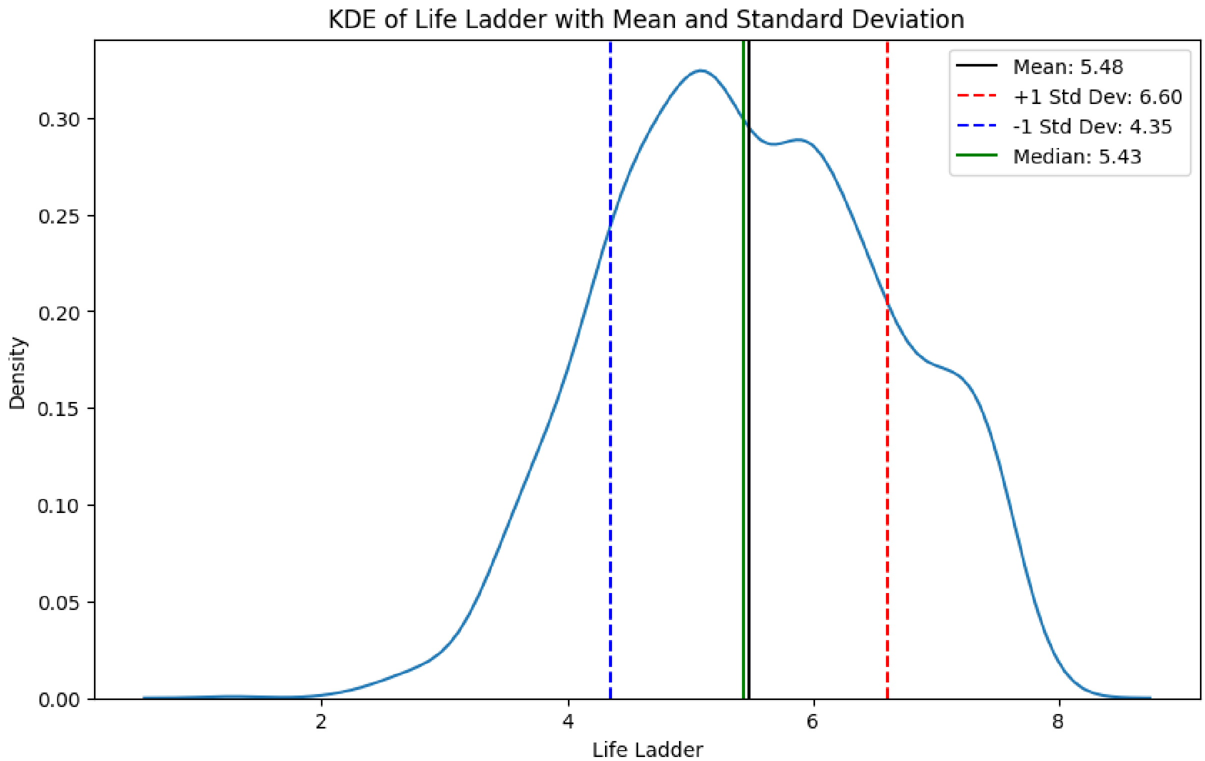


In [10]:
```python
print('Life Ladder skew', df['Life Ladder'].skew())
print('Life Ladder kurt', df['Life Ladder'].kurt())
```

```
Life Ladder skew -0.017819331787346102
Life Ladder kurt -0.5918479723083578
```

In [11]:
```python
# Calculate basic statistics for 'Life Ladder' column
mean = df['Life Ladder'].mean()
std = df['Life Ladder'].std()
median = df['Life Ladder'].median()
# Set up the figure for plotting
plt.figure(figsize=(10, 6))
# Create a KDE plot for the 'Life Ladder' scores
sns.kdeplot(df['Life Ladder'])
# Add vertical lines to the plot representing the mean, standard deviations,
plt.axvline(mean, color='k', label=f'Mean: {mean:.2f}')
plt.axvline(mean + std, color='r', linestyle='--', label=f'+1 Std Dev: {mear
plt.axvline(mean - std, color='b', linestyle='--', label=f'-1 Std Dev: {mear
plt.axvline(median, color='g', label=f'Median: {median:.2f}')
plt.legend()
plt.title('KDE of Life Ladder with Mean and Standard Deviation')
```

```
plt.xlabel('Life Ladder')
plt.ylabel('Density')
plt.show()
```



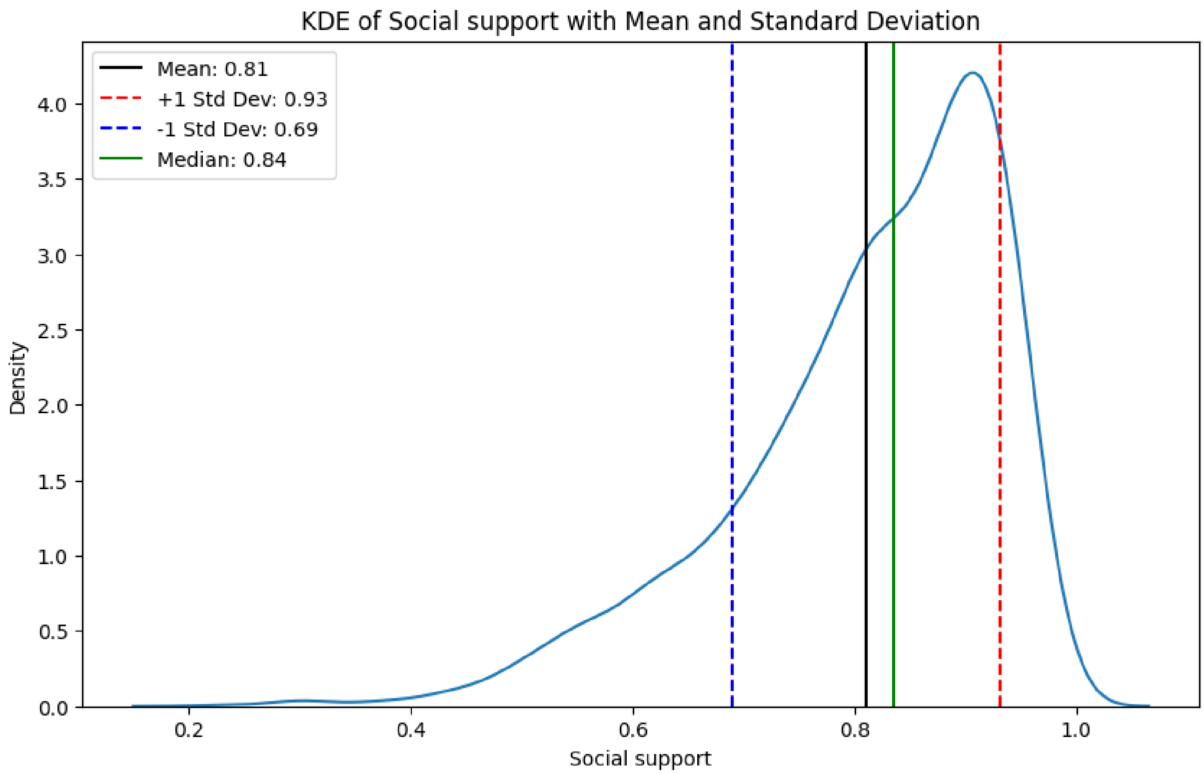
> Life ladder skew :
> The skewness of the Life Ladder distribution is approximately -0.018 which is very close to zero indicating a symmetrical shape of the distribution around the mean. This suggests that there is a balanced spread of data points on either side of the mean value, with no significant skew to the left or right. The KDE plot supports this observation as the mean and median lines are closely aligned.
> The tails of the Life Ladder distribution are lighter than those of a normal distribution resulting in fewer outliers than would be expected in a normal distribution.

In [12]:
```
print('Social support skew', df['Social support'].skew())
print('Social support kurt', df['Social support'].kurt())
```

```
Social support skew -1.1188344045866705
Social support kurt 1.1739089147806463
```

In [13]:
```
# Calculate basic statistics for 'Social support' column
mean = df['Social support'].mean()
std = df['Social support'].std()
median = df['Social support'].median()
# Set up the figure for plotting
plt.figure(figsize=(10, 6))
# Create a KDE plot for the 'Social support' scores
```

```
sns.kdeplot(df['Social support'])
# Add vertical lines to the plot representing the mean, standard deviations,
plt.axvline(mean, color='k', label=f'Mean: {mean:.2f}')
plt.axvline(mean + std, color='r', linestyle='--', label=f'+1 Std Dev: {mear
plt.axvline(mean - std, color='b', linestyle='--', label=f'-1 Std Dev: {mear
plt.axvline(median, color='g', label=f'Median: {median:.2f}')
plt.legend()
plt.title('KDE of Social support with Mean and Standard Deviation')
plt.xlabel('Social support')
plt.ylabel('Density')
plt.show()
```



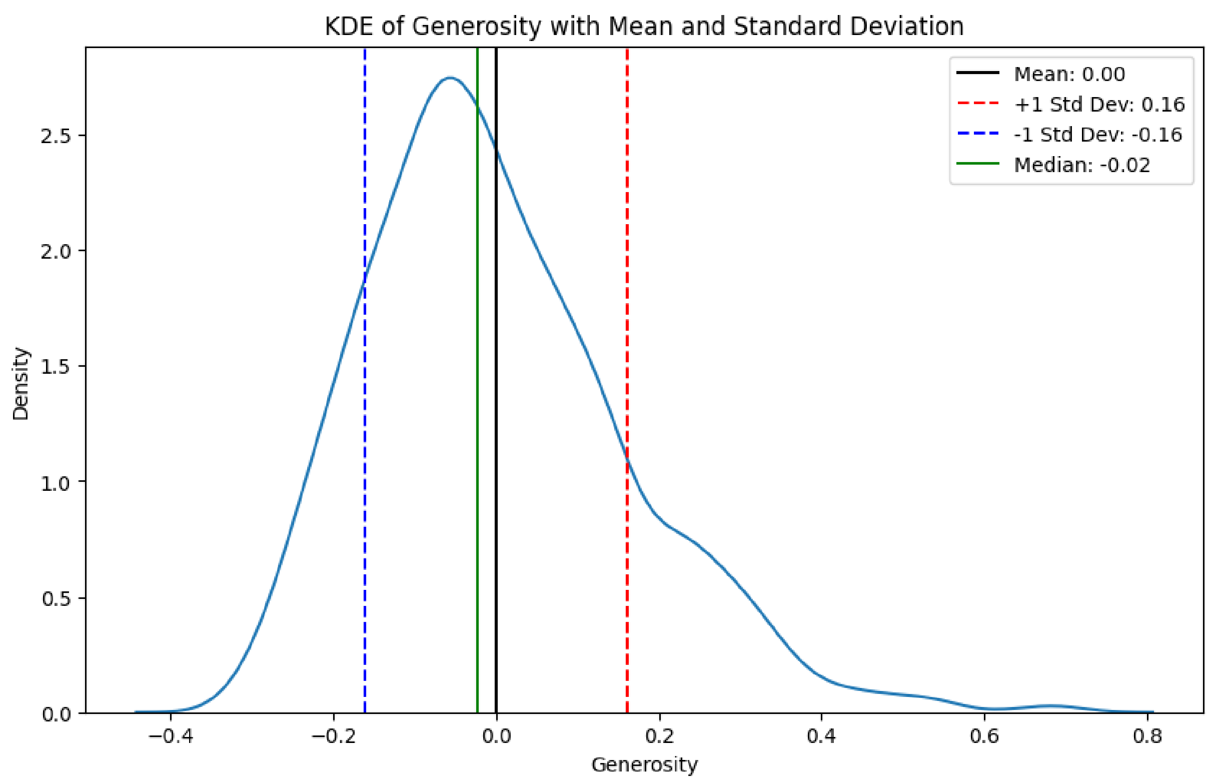
> Social support skew and kurtosis:
> The skewness is negative (notice the order of the lines in the KDE plot Mean and then Median), indicating that the distribution is skewed to the left. This means there are more data points with higher values and fewer with lower values.
> The kurtosis is greater than 1, which suggests that the distribution has heavier tails and a sharper peak than the normal distribution.

In [14]:
```
print('Generosity skew', df['Generosity'].skew())
print('Generosity kurt', df['Generosity'].kurt())
```

```
Generosity skew 0.7770869685066418
Generosity kurt 0.8315897969858366
```

In [15]:
```
# Calculate basic statistics for 'Generosity' column
mean = df['Generosity'].mean()
std = df['Generosity'].std()
```

```
median = df['Generosity'].median()
# Set up the figure for plotting
plt.figure(figsize=(10, 6))
# Create a KDE plot for the 'Generosity' scores
sns.kdeplot(df['Generosity'])
# Add vertical lines to the plot representing the mean, standard deviations,
plt.axvline(mean, color='k', label=f'Mean: {mean:.2f}')
plt.axvline(mean + std, color='r', linestyle='--', label=f'+1 Std Dev: {mear
plt.axvline(mean - std, color='b', linestyle='--', label=f'-1 Std Dev: {mear
plt.axvline(median, color='g', label=f'Median: {median:.2f}')
plt.legend()
plt.title('KDE of Generosity with Mean and Standard Deviation')
plt.xlabel('Generosity')
plt.ylabel('Density')
plt.show()
```

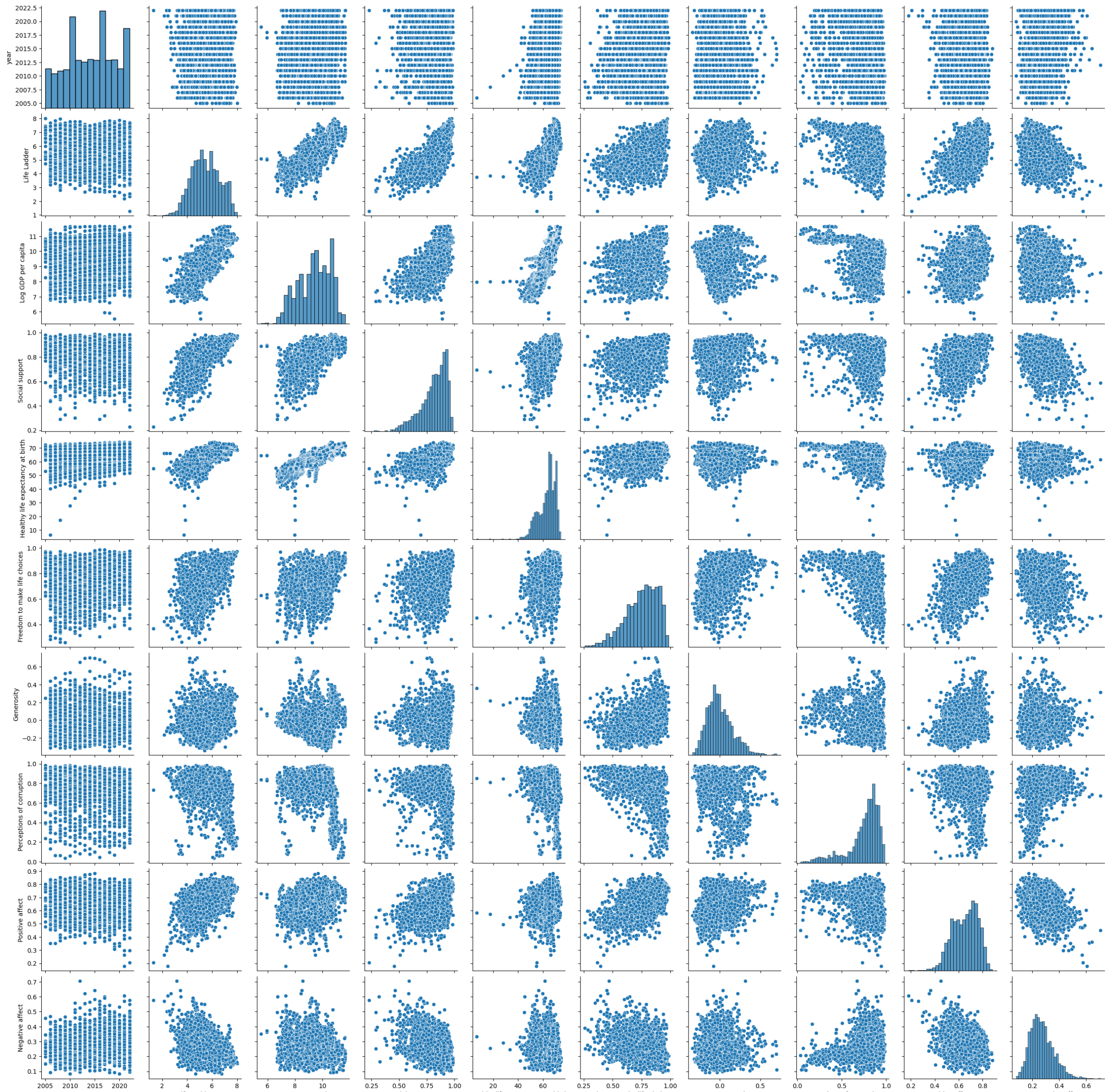


> Generosity skew and kurtosis:
> The skewness is positive (notice the order of the lines in the KDE plot Median and then Mean), indicating that the distribution is skewed to the right .This means there are more data points with lower values and fewer with higher values.
> The kurtosis is close to 0.8315 which is less extreme than Social support indicating a less sharp peak but it still suggests a distribution with slightly heavier tails than the normal distribution.
>
> If other features are similarly skewed or have high kurtosis, it would suggest that the data for these variables are not normally distributed. The presence of skewness and kurtosis in the data is not inherently a problem

> but it requires careful consideration during analysis especially for methods that are sensitive to these characteristics of the data distribution.

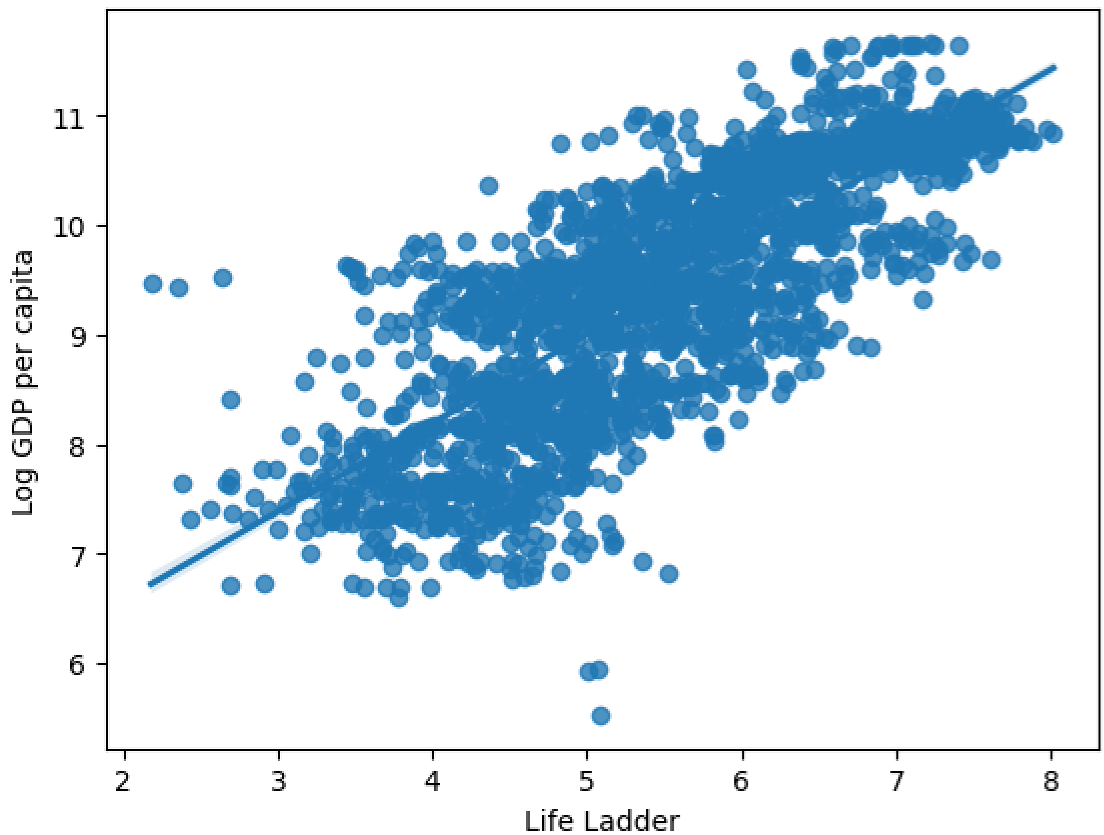
In [16]:
```
sns.pairplot(df)
```

Out[16]: <seaborn.axisgrid.PairGrid at 0x16a8e7760>



In [17]:
```
sns.regplot(data=df, x="Life Ladder", y="Log GDP per capita")
```

Out[17]: <Axes: xlabel='Life Ladder', ylabel='Log GDP per capita'>



> The data points suggest a positive linear relationship. Countries with higher GDP per capita tend to report higher happiness scores. This is further supported by the regression line which shows an upward trend indicating that happiness increases with GDP per capita. While the trend is apparent the spread of the data points suggests that there is variability that is not captured by GDP alone.

- Are there any interesting correlations in the data?

In [18]:
```
df.corr()
```

```
/var/folders/1q/p62jltqs1g10s6hzbjjqmw8c0000gq/T/ipykernel_45070/1134722465.
py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  df.corr()
```

Out[18]:

	year	Life Ladder	Log GDP per capita	Social support	Healthy life expectancy at birth	Freedom to make life choices	Gen
year	1.000000	0.045943	0.077772	-0.029750	0.163500	0.234105	0.
Life Ladder	0.045943	1.000000	0.784871	0.721663	0.713493	0.534532	0
Log GDP per capita	0.077772	0.784871	1.000000	0.683619	0.818126	0.367560	-0.
Social support	-0.029750	0.721663	0.683619	1.000000	0.597682	0.409439	0.
Healthy life expectancy at birth	0.163500	0.713493	0.818126	0.597682	1.000000	0.373448	0.
Freedom to make life choices	0.234105	0.534532	0.367560	0.409439	0.373448	1.000000	0.
Generosity	0.005726	0.181658	-0.000800	0.068593	0.010876	0.325107	1.
Perceptions of corruption	-0.081358	-0.431569	-0.352884	-0.222584	-0.299055	-0.476537	-0.
Positive affect	0.019182	0.518207	0.237986	0.431139	0.223119	0.578752	0.
Negative affect	0.205369	-0.339992	-0.247560	-0.441800	-0.140726	-0.275470	-0.

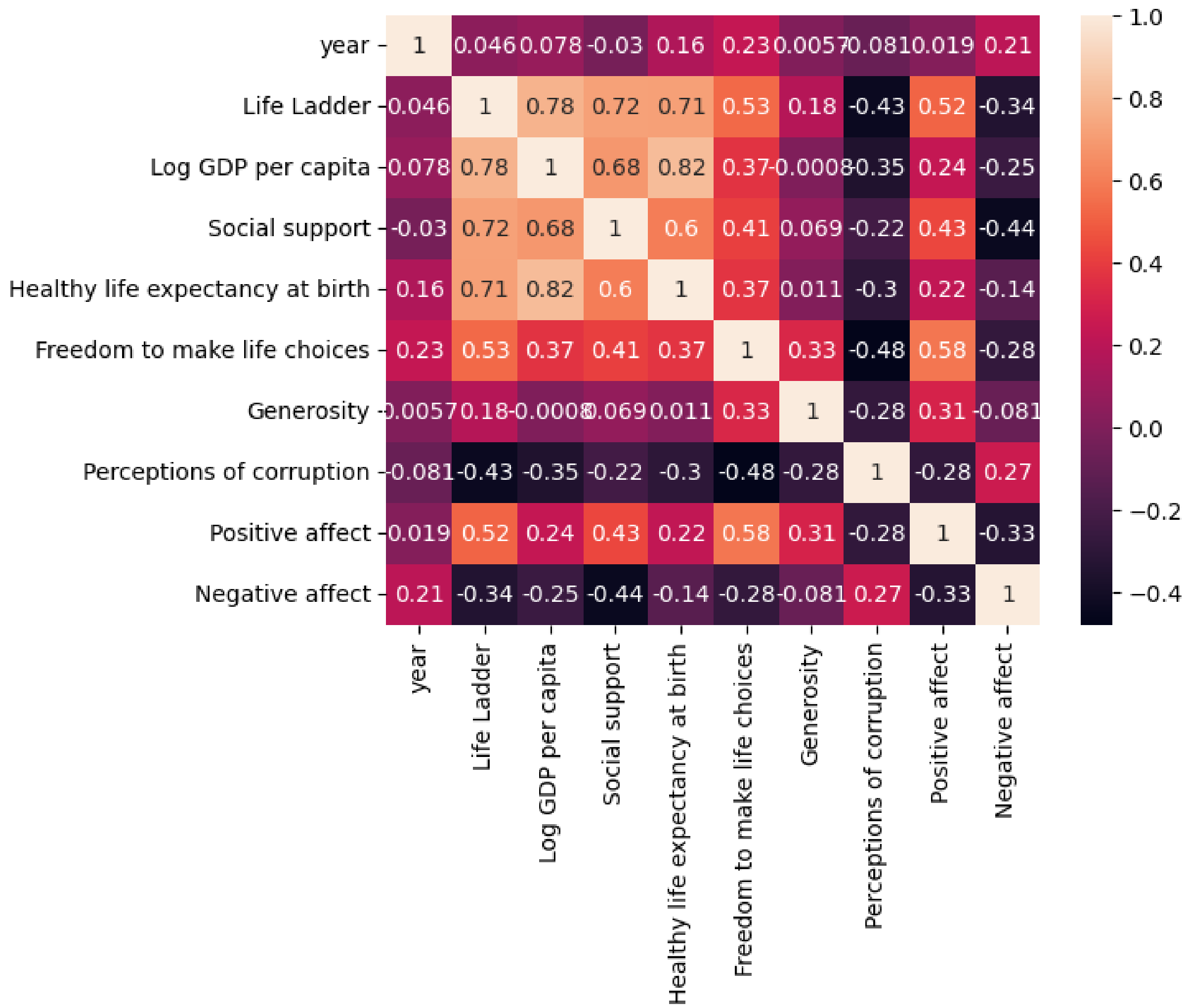
In [19]:

```
sns.heatmap(df.corr(), annot = True)
```

```
/var/folders/1q/p62jltqs1g10s6hzbjjqmw8c0000gq/T/ipykernel_45070/3028576344.
py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot = True)
```

Out[19]:

```
<Axes: >
```



> Life Ladder and Log GDP per capita: There is a strong positive correlation (0.78) indicating that countries with higher GDP per capita tend to have higher life satisfaction scores.
> Life Ladder and Social support: Another strong positive correlation (0.72) suggests that higher levels of social support are associated with higher life satisfaction.
> Healthy life expectancy at birth and Log GDP per capita: This also shows a strong positive correlation (0.82) which implies that higher GDP per capita is associated with longer healthy life expectancy.
> Freedom to make life choices and Life Ladder: There is a moderately strong positive correlation (0.53), indicating that the more freedom people have to make life choices the higher their life satisfaction is.
> Generosity does not show a strong correlation with any other variable: It's interesting to note generosity has very low correlation coefficients with all other variables suggesting that it behaves independently of factors like economic wealth, social support, and life expectancy.

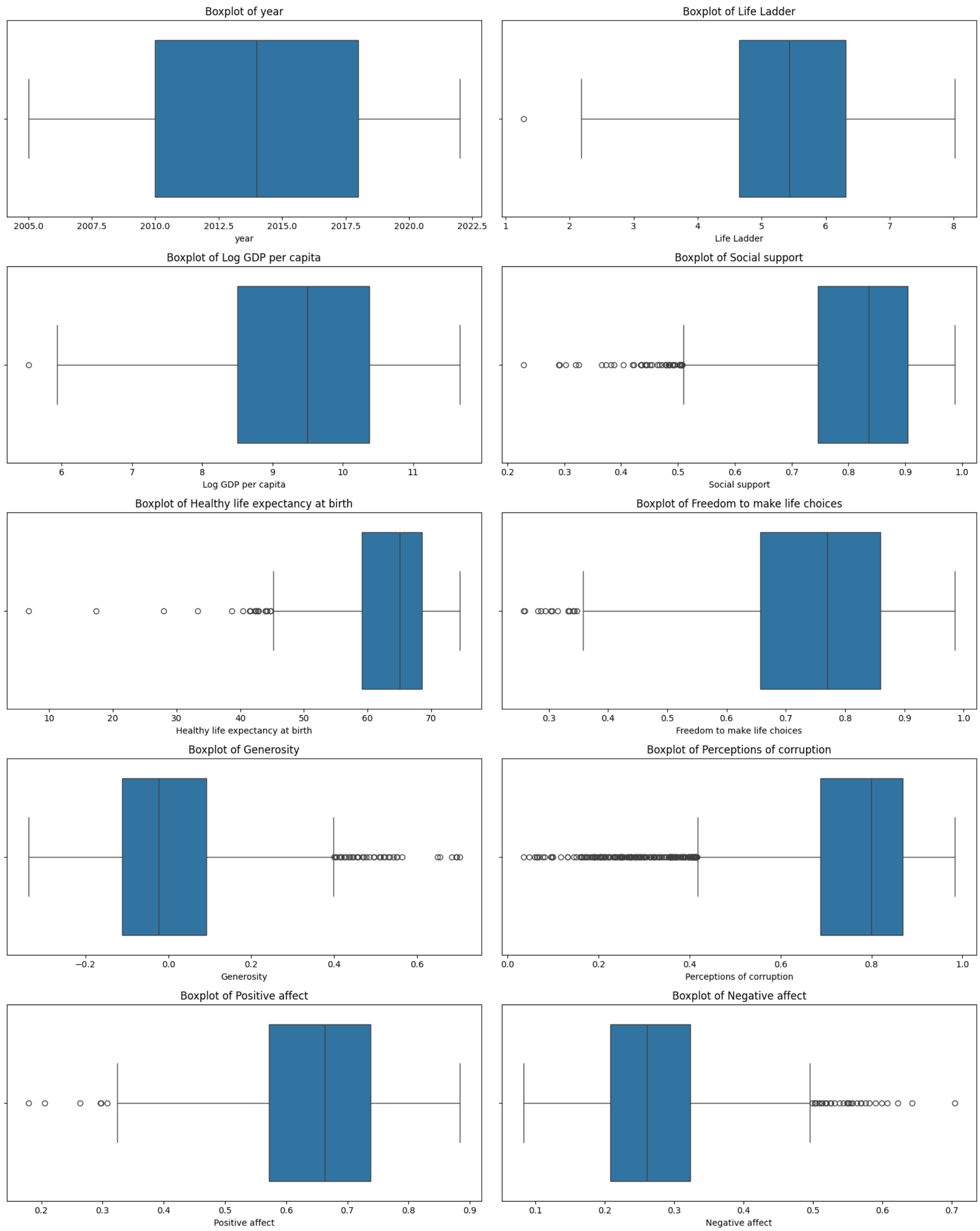
- Can you identify any limitations, missing values or distortions of the data?

In [20]:
```
df.year.value_counts()
```

Out[20]:
```
2017    147
2011    146
2014    144
2019    143
2015    142
2012    141
2016    141
2018    141
2013    136
2010    124
2021    122
2020    116
2009    114
2022    114
2008    110
2007    102
2006     89
2005     27
Name: year, dtype: int64
```

> Not all countries have data for all years.

In [21]:
```
# Select only the numeric features from the DataFrame
numeric_features = df.select_dtypes(include=['float64', 'int64'])
num_features = len(numeric_features.columns)
# Calculate the number of rows needed for subplots based on the number of fe
# We divide by 2 because we want 2 plots per row
num_rows = (num_features) // 2
# Create a subplot grid of the calculated size, with 2 columns
fig, axes = plt.subplots(num_rows, 2, figsize = (16, num_rows * 4))
axes = axes.flatten()
# Iterate over the numeric features and their respective axes to create boxp
for i, feature in enumerate(numeric_features.columns):
    sns.boxplot(x = df[feature], ax = axes[i])
    axes[i].set_title(f'Boxplot of {feature}')
plt.tight_layout()
plt.show()
```



> Missing Values: There are missing values in several columns such as Log GDP per capita, Social support, Healthy life expectancy at birth, Freedom to make life choices, Generosity, Perceptions of corruption, Positive affect, and Negative affect. These missing values can bias the analysis.
> Skewed Distributions: Several features show skewed distributions (e.g Social support, Healthy life expectancy at birth). Skewness can lead to biases in analyses particularly in modeling that assumes normality of the

> features.
> Outliers : After reviewing the boxplots for each numeric feature in the dataset it's apparent that all columns except the year column exhibit potential outliers. These outliers are indicated by points that lie beyond the whiskers of the boxplots which extend to 1.5 times the iqr from the upper and lower quartiles. The presence of outliers across all features suggests that there may be extreme values or data entry errors that could impact the analysis.

- What would you like to see in this dataset?

In [22]:
```
df.columns
```

Out[22]:
```
Index(['Country name', 'year', 'Life Ladder', 'Log GDP per capita',
       'Social support', 'Healthy life expectancy at birth',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption', 'Positive affect', 'Negative affect'],
      dtype='object')
```

> Few other additional variables :
> Measures of education or literacy rates of a country
> Employment rates
> Environmental quality
> Detailed breakdowns of income brackets
> Data regarding political events or economic crises in a country.

Q2

- What is the happiest country in 2023? In 2013? What is the least happy country in 2023, 2013 ?

In [23]:
```
def find_happiness(year, df):
    # Filter the DataFrame for the specified year
    filtered = df[df['year'] == year]
    # Check if the filtered DataFrame is not empty (i.e there is data for th
    if not filtered.empty:
        # Identify countries as the one with the max/min Life Ladder score f
        happiest_country = filtered.loc[filtered['Life Ladder'].idxmax()]
        least_happy_country = filtered.loc[filtered['Life Ladder'].idxmin()]
        # Check if the happiest and least happy countries are found not None
        if happiest_country is not None and least_happy_country is not None:
            # Print its name and Life Ladder score
            print(f"The happiest country in {year} is {happiest_country['Cou
            print(f"The least happy country in {year} is {least_happy_countr
    else:
```

```
        # If there is no data for the given year
        print(f"No data available for the year {year}.")
```

In [24]:
```
find_happiness(2023, df)
find_happiness(2022, df)
find_happiness(2013, df)
```

```
No data available for the year 2023.
The happiest country in 2022 is Finland with a Life Ladder score of 7.728998
184204102.
The least happy country in 2022 is Afghanistan with a Life Ladder score of
1.2812711000442505.
The happiest country in 2013 is Canada with a Life Ladder score of 7.5937938
69018555.
The least happy country in 2013 is Syria with a Life Ladder score of 2.68755
29289245605.
```

- What is the happiest country of all time? What is the least happy country of all time?

In [25]:
```
# Find the row with the highest Life Ladder score in the entire DataFrame
happiest_all_time = df.loc[df['Life Ladder'].idxmax()]
print(f"The happiest country of all time is {happiest_all_time['Country name
# Find the row with the lowest Life Ladder score in the entire DataFrame
least_all_time = df.loc[df['Life Ladder'].idxmin()]
print(f"The least happy country of all time is {least_all_time['Country name
```

```
The happiest country of all time is Denmark with a Life Ladder score of 8.01
893424987793.
The least happy country of all time is Afghanistan with a Life Ladder score
of 1.2812711000442505.
```

In [26]:
```
# Group the DataFrame by Country name and calculate the mean Life Ladder sco
average_happiness = df.groupby('Country name')['Life Ladder'].mean()
# Find the country name of the max/min value in average happiness
happiest_all_time = average_happiness.idxmax()
least_all_time = average_happiness.idxmin()
print(f"The happiest country of all time is {happiest_all_time} with an aver
print(f"The least happy country of all time is {least_all_time} with an aver
```

```
The happiest country of all time is Denmark with an average Life Ladder scor
e of 7.673428395215203.
The least happy country of all time is Afghanistan with an average Life Ladd
er score of 3.3466316887310574.
```

Q3

- Of the countries in this dataset, which country had the largest increase in happiness from its start of participation to 2023? Which had the largest decrease in happiness?

In [27]:
```
country_names = df['Country name'].unique()
# Group the DataFrame by Country name
grouped = df.groupby('Country name')
```

```
results = []
for i in country_names:
    # Get the subset of the DataFrame corresponding to the current country
    grouped1 = grouped.get_group(i)
    # Extract the first/last entry for the country
    first_entry = grouped1.iloc[:1]
    last_entry = grouped1.iloc[-1:]
    # Calculate the difference in Life Ladder score from the first to last e
    diff = last_entry['Life Ladder'].array[0] - first_entry['Life Ladder'].a
    percent = diff / first_entry['Life Ladder'].array[0]
    results.append({'Country name': i , 'Value' : percent})
results_df = pd.DataFrame(results)
# Find the country with the min/max value of difference
max_increase = results_df.loc[results_df['Value'].idxmax()]
min_decrease = results_df.loc[results_df['Value'].idxmin()]
print(f"The country with the largest increase in happiness is {max_increase[
print(f"The country with the largest decrease in happiness is {min_decrease[
```

```
The country with the largest increase in happiness is Congo (Brazzaville) wi
th an increase of 0.5196948825225308.
The country with the largest decrease in happiness is Afghanistan with a dec
rease of -0.6559043462270266.
```

In [ ]: