CMSC 320: THE FINAL PROJECT

Name: Ritika Munshi and UID: 118345048

TABLE OF CONTENTS:

INTRODUCTION!!

LIFE EXPECTANCY:

It is basically an average or mean additional number of years a person is expected to live.

There are many factors that influence the **LIFE EXPECTANCY** of an individual or the population or the sample.
Some of the significant factors in life expectancy include gender, genetics, facility to health care, hygiene, diet and nutrition, exercise, lifestyle, and crime rates.
Evidence-based studies indicate that longevity is based on two major factors, genetics, and lifestyle choices.
In our project we would try to analyze how some of the factors influence the life expectancy of the person and how that impacts the economy of the country in the positive as well as negative way.

We collected a data set from a website termed as Kaggle. The data provides Statistical Analysis on factors influencing **Life Expectancy by the World Health Organization (WHO)**. Life Expectancy of a population or a sample or an individual depends on several factors such as demographic situations, economic situations, moratlity rate, income composition and many more. From several research and health factors, it has been found that immunization plays a significant role in the life expectancy; few immunization that we would consider from the collected Kagggle dataset are Hepatitis-B, Polio and Diphtheria. Our dataset is based on different countries. Therefore, this highlights that we would get an insight of how the life expectancy of the population is different based on several different factors we have in our dataframe for different countries in terms of low, high, average and many other statistical analysis. The information we get would help us to gain knowledge about which country needs to focus more on improvement and growth and which country is doing good and innovating well for the well-being of their population and people.

GOAL/MOTIVE:

Theortically we have learned the meaning, role, importance and factors influencing and affecting the life expectancy of an individual.
We have a dataset from Kaggle which is basically about the Life Expectancy. The dataset includes a lot of information about the LIFE EXPECTANCY of more than a 100 different countries with many different attributes and columns which is basically how we would explore the how life expectancy depends of those attributes or is impacted by those attributes positively and negatively.
Basically we learned the theoritical perspective in our schools and classes and now we are trying to find our practically in our DATA SCIENCE class by working on a dataset that has those information by applying the MACHINE LEARNING algorithms.

IMPORTING the required and necessary python or pandas libraries for our final project analysis and experimentations!!

In [194]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
from scipy.stats import shapiro
import scipy.stats as stats
from numpy.random import randn
from scipy.stats import normaltest
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn import metrics
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import DecisionTreeClassifier
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVC
from sklearn.model_selection import KFold
from statsmodels.stats import weightstats as stests
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn import model_selection
from sklearn import preprocessing
from sklearn.metrics import classification_report
import statsmodels.formula.api as smf
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestRegressor
from scipy.stats import shapiro
from seaborn import load_dataset, pairplot
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
from sklearn.linear_model import LogisticRegression
```

DATA COLLECTION

SOURCE:

Data collection from Kaggle: https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who

In [3]:
```
data = pd.read_csv("Life Expectancy Data.csv")
```

In [4]:
```
data_copy = data.copy()
data2 = data.copy()
```

LET'S EXPLORE WHAT DO WE HAVE IN OUR DATASET

In [5]:
```
data.head()
```

Out[5]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	0.1	584.
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	0.1	612.
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1	631.
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	8.52	67.0	0.1	669.
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	7.87	68.0	0.1	63.

5 rows × 22 columns

data.head(): This prints or displays first five (5) rows or records of our dataframe by default.

In [6]:
```
data.tail()
```

Out[6]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	
2933	Zimbabwe	2004	Developing	44.3	723.0	27	4.36	0.0	68.0	31	...	67.0	7.13	65.0	33.6	45
2934	Zimbabwe	2003	Developing	44.5	715.0	26	4.06	0.0	7.0	998	...	7.0	6.52	68.0	36.7	45
2935	Zimbabwe	2002	Developing	44.8	73.0	25	4.43	0.0	73.0	304	...	73.0	6.53	71.0	39.8	5
2936	Zimbabwe	2001	Developing	45.3	686.0	25	1.72	0.0	76.0	529	...	76.0	6.16	75.0	42.1	54
2937	Zimbabwe	2000	Developing	46.0	665.0	24	1.68	0.0	79.0	1483	...	78.0	7.10	78.0	43.5	54

5 rows × 22 columns

data.tail(): This prints or displays last five (5) rows or records of our dataframe by default.

Presenting the number of rows and columns of the dataframe.

In [7]:
```
data.shape
```

Out[7]: (2938, 22)

'The above results delivers that our dataframe has 1987 rows and 22 columns.'

Presenting the all the column name that we have in our dataframe from the dataset that we chose to work on

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

Out[8]:
```
Index(['Country', 'Year', 'Status', 'Life expectancy ', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',
       'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population',
       ' thinness  1-19 years', ' thinness 5-9 years',
       'Income composition of resources', 'Schooling'],
      dtype='object')
```

We have **22 columns** in our dataset and columns might or might not be related to each other dependently or independently which we would further analyze in our exploration process of Machine Learning/ Data Science using several techniques.

1) Country: Our dataset has information for 193 unique countries
2) Year: Our data has information from 2000 to 2015 year
3) Status: Provided information if it is a DEVELOPING or DEVELOPED country
4) Life Expectancy: The number of years a person can expect to live.
5) Adult Mortality: Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population).
6) Infant Deaths: Number of Infant Deaths per 1000 population.
7) Alcohol: Recorded per capita (15+) consumption (in litres of pure alcohol).
8) Percentage Expenditure: Expenditure on health as a percentage of Gross Domestic Product per capita(%).
9) Hepatitis-B: Hepatitis B (HepB) immunization coverage among 1-year-olds (%).
10) Measles: Number of reported cases per 1000 population.

11) Body Mass Index (BMI): A person's weight in kilograms (or pounds) divided by the square of height in meters (or feet).
12) Under-five deaths
13) Polio: A disabling and life-threatening disease caused by the poliovirus.
14) Total Expenditure
15) Diphtheria: A serious infection caused by strains of bacteria called Corynebacterium diphtheriae that make toxin.
16) HIV/AIDS: HIV (human immunodeficiency virus) and AIDS (acquired immunodeficiency syndrome).
17) Gross Domestic Product (GDP): The standard measure of the value added created through the production of goods and services in a country during a certain period.
18) Population
19) Thinness_1-19_years
20) Thinness_5-9_years
21) Income Composition Of Resources
22) Schooling

Renaming or editing the column names

In [9]:
```
data.columns = ['Country', 'Year', 'Status', 'Life_Expectancy', 'Adult_Mortality','Infant_deaths', 'Alcohol',
                'Percentage_expenditure', 'Hepatitis_B','Measles','BMI', 'under-five-deaths', 'Polio',
                'Total_expenditure','Diphtheria', 'HIV/AIDS', 'GDP', 'Population','thinness_1-19_years',
                'thinness_5-9_years','Income_composition_of_resources', 'Schooling']
```

In [10]:
```
data_copy.columns = ['Country', 'Year', 'Status', 'Life_Expectancy', 'Adult_Mortality','Infant_deaths', 'Alcohol',
                'Percentage_expenditure', 'Hepatitis_B','Measles','BMI', 'under-five-deaths', 'Polio',
                'Total_expenditure','Diphtheria', 'HIV/AIDS', 'GDP', 'Population','thinness_1-19_years',
                'thinness_5-9_years','Income_composition_of_resources', 'Schooling']
```

In [11]:
```
data2.columns = ['Country', 'Year', 'Status', 'Life_Expectancy', 'Adult_Mortality','Infant_deaths', 'Alcohol',
                'Percentage_expenditure', 'Hepatitis_B','Measles','BMI', 'under-five-deaths', 'Polio',
                'Total_expenditure','Diphtheria', 'HIV/AIDS', 'GDP', 'Population','thinness_1-19_years',
                'thinness_5-9_years','Income_composition_of_resources', 'Schooling']
```

Checking the datatypes of the columns of our dataframe

In [12]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Country                          2938 non-null   object 
 1   Year                             2938 non-null   int64  
 2   Status                           2938 non-null   object 
 3   Life_Expectancy                  2928 non-null   float64
 4   Adult_Mortality                  2928 non-null   float64
 5   Infant_deaths                    2938 non-null   int64  
 6   Alcohol                          2744 non-null   float64
 7   Percentage_expenditure           2938 non-null   float64
 8   Hepatitis_B                      2385 non-null   float64
 9   Measles                          2938 non-null   int64  
 10  BMI                              2904 non-null   float64
 11  under-five-deaths                2938 non-null   int64  
 12  Polio                            2919 non-null   float64
 13  Total_expenditure                2712 non-null   float64
 14  Diphtheria                       2919 non-null   float64
 15  HIV/AIDS                         2938 non-null   float64
 16  GDP                              2490 non-null   float64
 17  Population                       2286 non-null   float64
 18  thinness_1-19_years              2904 non-null   float64
 19  thinness_5-9_years               2904 non-null   float64
 20  Income_composition_of_resources  2771 non-null   float64
 21  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

data.info(): It provides of the information of what type of data do our columns of the dataframe hold, for instance; our dataframe hold data of type object, int64 and float64'

Computing the statistical values overall of all the column which have numerical datatypes using desribe function

In [13]:
```
data.describe()
```

Out[13]:

	Year	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	BMI	under-five-deaths
count	2938.000000	2928.000000	2928.000000	2938.000000	2744.000000	2938.000000	2385.000000	2938.000000	2904.000000	2938.0
mean	2007.518720	69.224932	164.796448	30.303948	4.602861	738.251295	80.940461	2419.592240	38.321247	42.0
std	4.613841	9.523867	124.292079	117.926501	4.052413	1987.914858	25.070016	11467.272489	20.044034	160.4
min	2000.000000	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	0.000000	1.000000	0.0
25%	2004.000000	63.100000	74.000000	0.000000	0.877500	4.685343	77.000000	0.000000	19.300000	0.0
50%	2008.000000	72.100000	144.000000	3.000000	3.755000	64.912906	92.000000	17.000000	43.500000	4.0
75%	2012.000000	75.700000	228.000000	22.000000	7.702500	441.534144	97.000000	360.250000	56.200000	28.0
max	2015.000000	89.000000	723.000000	1800.000000	17.870000	19479.911610	99.000000	212183.000000	87.300000	2500.0

data.describe(): This basically provided us overall statistical information for all the columns fo numeric datatypes with different statistical analysis such as mean, median, standard deviation, minimum values, maximum values, 25%, 75%, total number of values.

Dealing with Missing Values

Dealing with missing values is an important step towards Machine Learning algorithm and Data Science application. Missing values in the sense null or nan values in our dataset which could affect the performance of our analysis we do using different techniques which therefore might end up giving not effective or accurate result that we want or expect to get. Therefore, Dealing with missing values is step of Data Cleaning to work with clean and pretty data forward.

This gives us sum of missing values we have for every column or attribute in our dataset or dataframe

In [14]:
```
data.isnull().sum()
```

Out[14]:
```
Country                            0
Year                               0
Status                             0
Life_Expectancy                   10
Adult_Mortality                   10
Infant_deaths                      0
Alcohol                          194
Percentage_expenditure             0
Hepatitis_B                      553
Measles                            0
BMI                               34
under-five-deaths                  0
Polio                             19
Total_expenditure                226
Diphtheria                        19
HIV/AIDS                           0
GDP                              448
Population                       652
thinness_1-19_years               34
thinness_5-9_years                34
Income_composition_of_resources  167
Schooling                        163
dtype: int64
```

The above command gives us the total number of rows for that particular column, how many rows have nothing or null values. For instance, our column termed as Life_Expectancy has 10 rows with null values where as column termed as Country has no missing values means all of its rows has the name of the countries.

Presenting the number of missing values in terms of percentage that we have in our dataframe

In [15]:
```
data.isnull().sum()*100/len(data)
```

Out[15]:
```
Country                           0.000000
Year                              0.000000
Status                            0.000000
Life_Expectancy                   0.340368
Adult_Mortality                   0.340368
Infant_deaths                     0.000000
Alcohol                           6.603131
Percentage_expenditure            0.000000
Hepatitis_B                      18.822328
Measles                           0.000000
BMI                               1.157250
under-five-deaths                 0.000000
Polio                             0.646698
Total_expenditure                 7.692308
Diphtheria                        0.646698
HIV/AIDS                          0.000000
GDP                              15.248468
Population                       22.191967
thinness_1-19_years               1.157250
thinness_5-9_years                1.157250
Income_composition_of_resources   5.684139
Schooling                         5.547992
dtype: float64
```

Checking the number of countries we have in our dataset

In [16]:
```
len(pd.unique(data['Country']))
```

Out[16]:
```
193
```

Presenting the name of the countries that we have are going to work on in our dataset

In [17]:
```
pd.unique(data['Country'])
```

Out[17]:
```
array(['Afghanistan', 'Albania', 'Algeria', 'Angola',
       'Antigua and Barbuda', 'Argentina', 'Armenia', 'Australia',
       'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh',
       'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan',
       'Bolivia (Plurinational State of)', 'Bosnia and Herzegovina',
       'Botswana', 'Brazil', 'Brunei Darussalam', 'Bulgaria',
       'Burkina Faso', 'Burundi', "Côte d'Ivoire", 'Cabo Verde',
       'Cambodia', 'Cameroon', 'Canada', 'Central African Republic',
       'Chad', 'Chile', 'China', 'Colombia', 'Comoros', 'Congo',
       'Cook Islands', 'Costa Rica', 'Croatia', 'Cuba', 'Cyprus',
       'Czechia', "Democratic People's Republic of Korea",
       'Democratic Republic of the Congo', 'Denmark', 'Djibouti',
       'Dominica', 'Dominican Republic', 'Ecuador', 'Egypt',
       'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia',
       'Ethiopia', 'Fiji', 'Finland', 'France', 'Gabon', 'Gambia',
       'Georgia', 'Germany', 'Ghana', 'Greece', 'Grenada', 'Guatemala',
       'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti', 'Honduras',
       'Hungary', 'Iceland', 'India', 'Indonesia',
       'Iran (Islamic Republic of)', 'Iraq', 'Ireland', 'Israel', 'Italy',
       'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati',
       'Kuwait', 'Kyrgyzstan', "Lao People's Democratic Republic",
       'Latvia', 'Lebanon', 'Lesotho', 'Liberia', 'Libya', 'Lithuania',
       'Luxembourg', 'Madagascar', 'Malawi', 'Malaysia', 'Maldives',
       'Mali', 'Malta', 'Marshall Islands', 'Mauritania', 'Mauritius',
       'Mexico', 'Micronesia (Federated States of)', 'Monaco', 'Mongolia',
       'Montenegro', 'Morocco', 'Mozambique', 'Myanmar', 'Namibia',
       'Nauru', 'Nepal', 'Netherlands', 'New Zealand', 'Nicaragua',
       'Niger', 'Nigeria', 'Niue', 'Norway', 'Oman', 'Pakistan', 'Palau',
       'Panama', 'Papua New Guinea', 'Paraguay', 'Peru', 'Philippines',
       'Poland', 'Portugal', 'Qatar', 'Republic of Korea',
       'Republic of Moldova', 'Romania', 'Russian Federation', 'Rwanda',
       'Saint Kitts and Nevis', 'Saint Lucia',
       'Saint Vincent and the Grenadines', 'Samoa', 'San Marino',
       'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia',
       'Seychelles', 'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia',
       'Solomon Islands', 'Somalia', 'South Africa', 'South Sudan',
       'Spain', 'Sri Lanka', 'Sudan', 'Suriname', 'Swaziland', 'Sweden',
       'Switzerland', 'Syrian Arab Republic', 'Tajikistan', 'Thailand',
       'The former Yugoslav republic of Macedonia', 'Timor-Leste', 'Togo',
       'Tonga', 'Trinidad and Tobago', 'Tunisia', 'Turkey',
       'Turkmenistan', 'Tuvalu', 'Uganda', 'Ukraine',
       'United Arab Emirates',
       'United Kingdom of Great Britain and Northern Ireland',
       'United Republic of Tanzania', 'United States of America',
       'Uruguay', 'Uzbekistan', 'Vanuatu',
       'Venezuela (Bolivarian Republic of)', 'Viet Nam', 'Yemen',
       'Zambia', 'Zimbabwe'], dtype=object)
```

hecking the number of years we have in our dataset

In [18]:
```
len(pd.unique(data['Year']))
```

Out[18]:
```
16
```

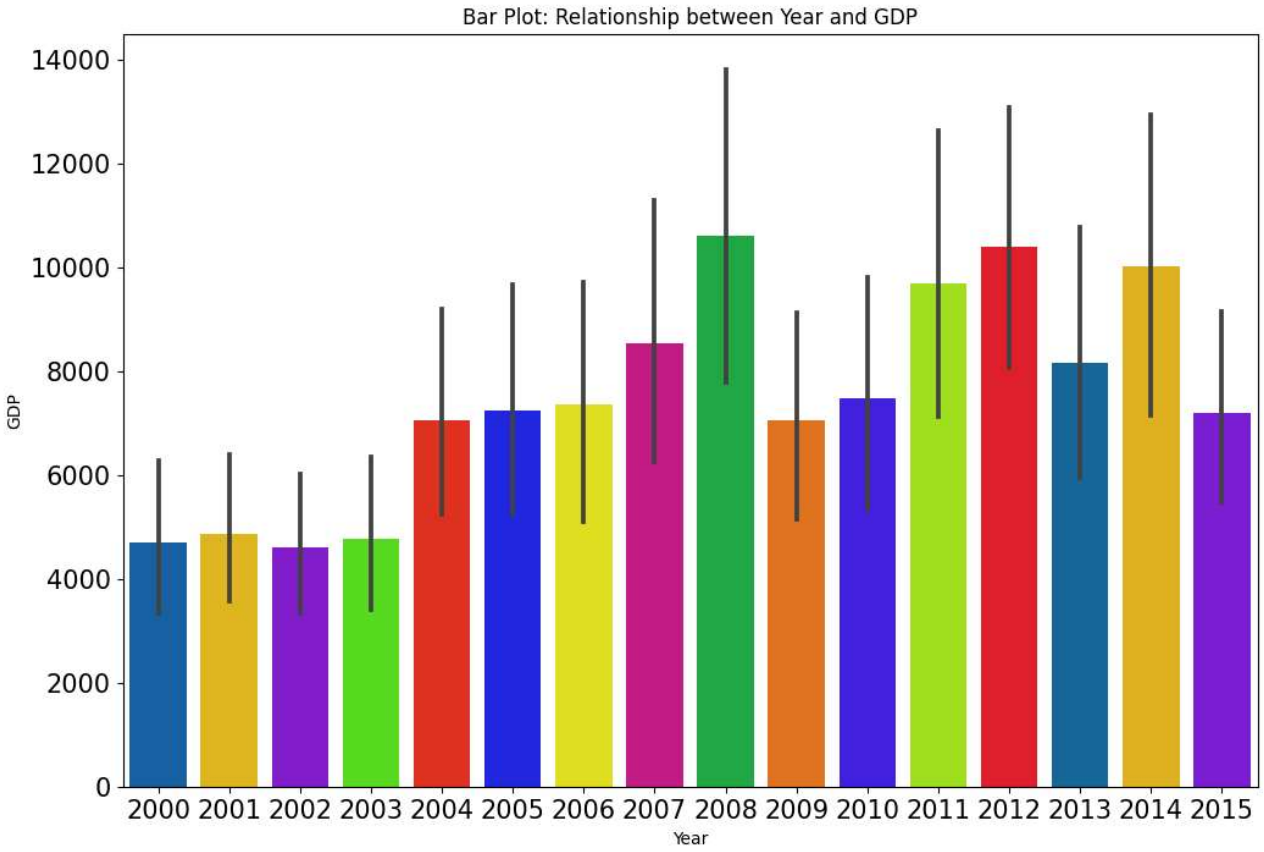
Checking the years we have in our dataset

In [19]:
```
pd.unique(data['Year'])
```

Out[19]:
```
array([2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 2006, 2005,
       2004, 2003, 2002, 2001, 2000], dtype=int64)
```

Bar charts to compute averages

In [20]:
```
plt.figure(figsize=(12,8))
graph = sns.barplot(x="Year",y = "GDP", data=data, palette="prism")
plt.xticks(size=15)
plt.yticks(size=15)
plt.ylabel("GDP")
plt.xlabel("Year")
plt.title("Bar Plot: Relationship between Year and GDP")
plt.show()
```



Above Plot:
We can see from the plot that there is fluctuating relationship between GDP (Gross Domestic Product) and growing years. We can observe that GDP was highest in the year 2008 which is more than 10000 approximately where as seems to be lowest at the year 2002 with slightly more than 4000. So we see the difference of approximately close to 6000 for the year 2008 and 2002.

In [21]:
```
data2002 = data.loc[data['Year'] == 2002]
data2002.describe()
```

Out[21]:

	Year	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	BMI	under-five-deaths	
count	183.0	183.000000	183.000000	183.000000	182.000000	183.000000	113.000000	183.000000	181.000000	183.000000	18
mean	2002.0	67.351366	171.437158	35.584699	4.660934	476.794487	76.522124	3204.754098	37.110497	50.300546	7
std	0.0	10.062469	133.472330	141.030361	3.967477	1162.688712	27.957269	8591.234477	18.449467	194.206932	2
min	2002.0	44.000000	3.000000	0.000000	0.010000	0.000000	4.000000	0.000000	1.000000	0.000000	
25%	2002.0	59.350000	74.000000	0.000000	1.242500	5.161358	66.000000	0.000000	18.000000	0.500000	7
50%	2002.0	71.400000	146.000000	3.000000	3.745000	49.735114	88.000000	36.000000	42.000000	4.000000	9
75%	2002.0	74.800000	234.500000	25.000000	7.382500	242.749329	95.000000	1518.000000	54.000000	35.000000	9
max	2002.0	84.000000	699.000000	1700.000000	14.170000	6853.628494	99.000000	58341.000000	69.700000	2300.000000	9

As we saw above that 2002 had the lowest GDP when we tried to find the relation between GDP and year. Here we created a separated dataframe and extarcted all the information from our original dataframe for only year 2002, and used describe commnand to find our the statistical information of the year 2002 for different countries and repsective other factors.

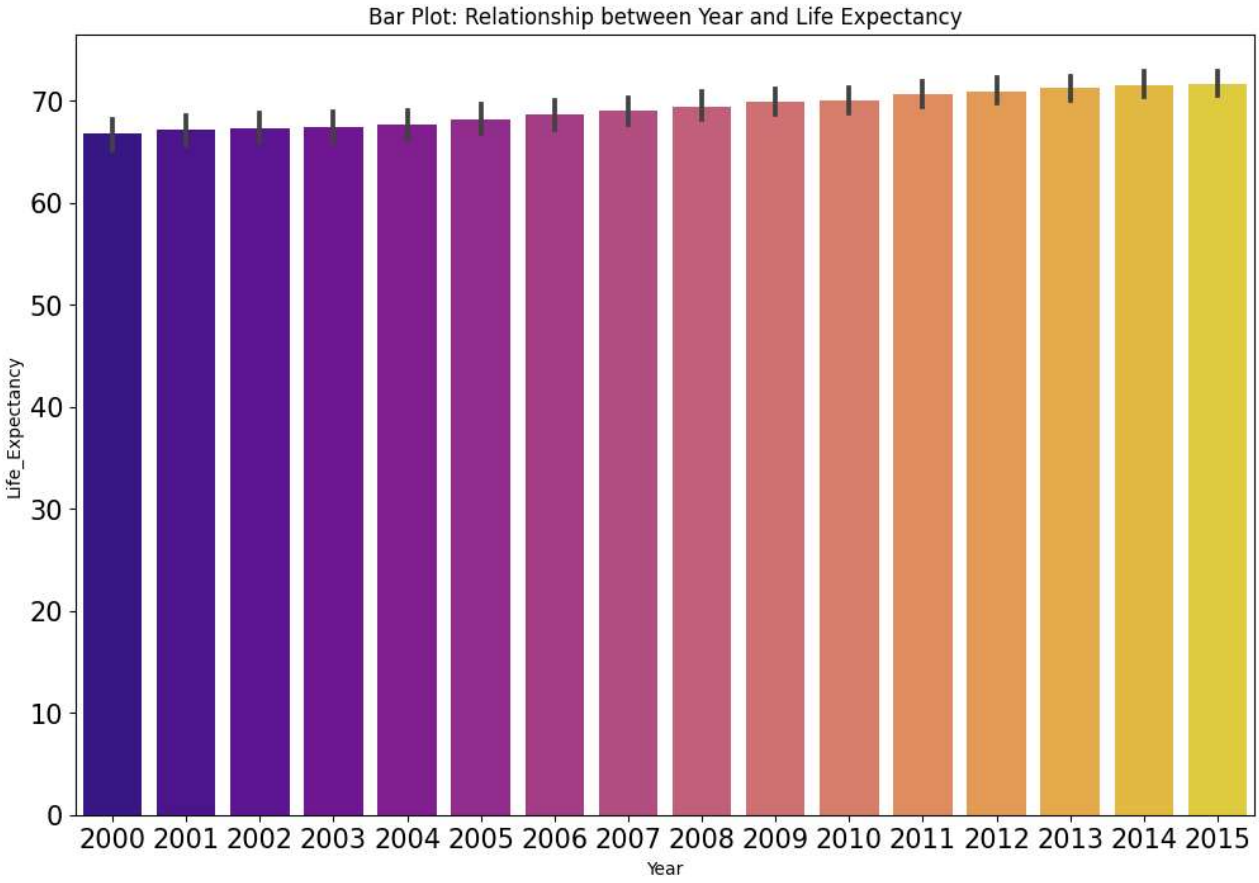
In [22]:
```
data2008 = data.loc[data['Year'] == 2008]
data2008.describe()
```

Out[22]:

	Year	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	BMI	under-five-deaths
count	183.0	183.000000	183.000000	183.000000	182.000000	183.000000	163.000000	183.000000	181.000000	183.000000
mean	2008.0	69.427869	174.519126	29.568306	5.007088	1095.802669	83.644172	1523.229508	38.225414	41.322404
std	0.0	9.202612	120.419771	111.310646	4.098128	2615.641873	21.084686	10375.663062	20.183855	155.823759
min	2008.0	46.200000	1.000000	0.000000	0.010000	0.000000	9.000000	0.000000	2.300000	0.000000
25%	2008.0	62.700000	87.500000	0.000000	1.375000	16.904198	81.000000	0.000000	19.600000	0.000000
50%	2008.0	72.400000	157.000000	3.000000	4.185000	93.367890	92.000000	4.000000	43.400000	4.000000
75%	2008.0	75.350000	234.500000	22.000000	8.292500	529.524032	97.000000	134.500000	56.600000	28.000000
max	2008.0	89.000000	632.000000	1300.000000	16.990000	18961.348600	99.000000	131441.000000	74.100000	1800.000000

As we saw above that 2008 had the highest GDP when we tried to find the relation between GDP and year. Here we created a separated dataframe and extarcted all the information from our original dataframe for only year 2008, and used describe commnand to find our the statistical information of the year 2008 for different countries and repsective other factors.

In [23]:
```
plt.figure(figsize=(12,8))
graph = sns.barplot(x="Year",y = "Life_Expectancy", data=data, palette="plasma")
plt.xticks(size=15)
plt.yticks(size=15)
plt.ylabel("Life_Expectancy")
plt.xlabel("Year")
plt.title("Bar Plot: Relationship between Year and Life Expectancy")
plt.show()
```



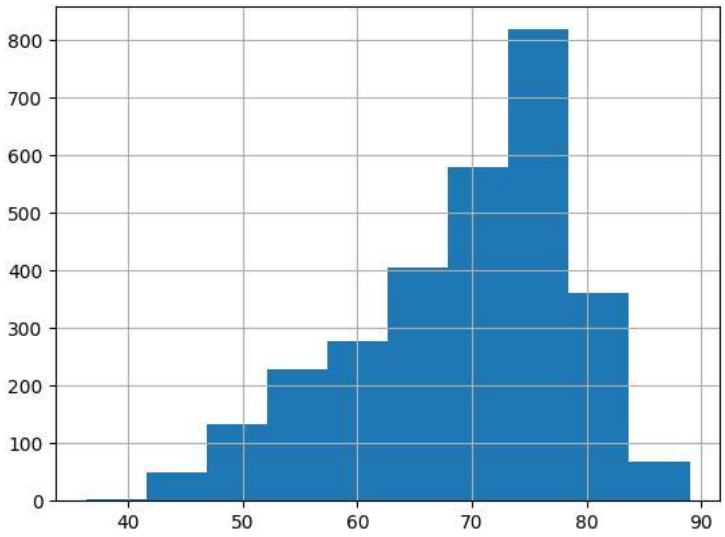
Above Plot:
Analyzing the bar plot that we made above which demonstrates the relationship between the Year and Life Expectancy, We can see a positive linear relationship following that the life expectancy seems to be increasing or improving as the growing age with slighting inclinment.

LIFE EXPECTANCY would be our independent variable and is our main consideration for the analysis or during our data exploration. Analyzing the dataset that we have and exploring the relationships that could be formed with the column that is given to us; I believe Life Excpectancy would be one of the columns I would take as an important column for consideration and try to understand and observe the relationship between life expectany with other columns from our

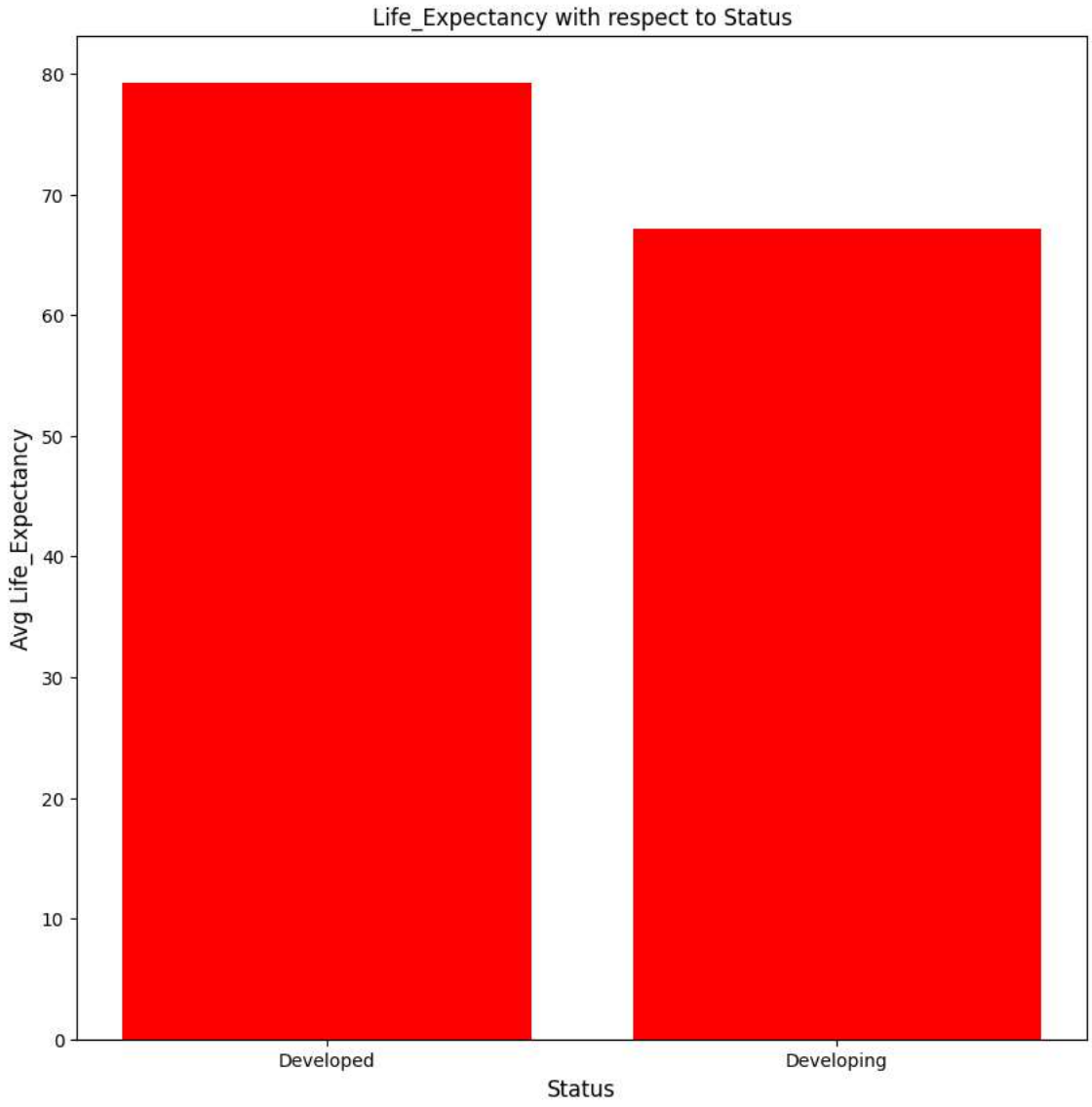
original dataset or dataframe. Along with analyzing the dataset with life expectancy and relations with other columns, I would be working with identifying outliers as well and move forward to removing outliers as part/process of data curation or data cleaning.

In [24]:
```
data['Life_Expectancy'].hist()
```

Out[24]: <AxesSubplot: >

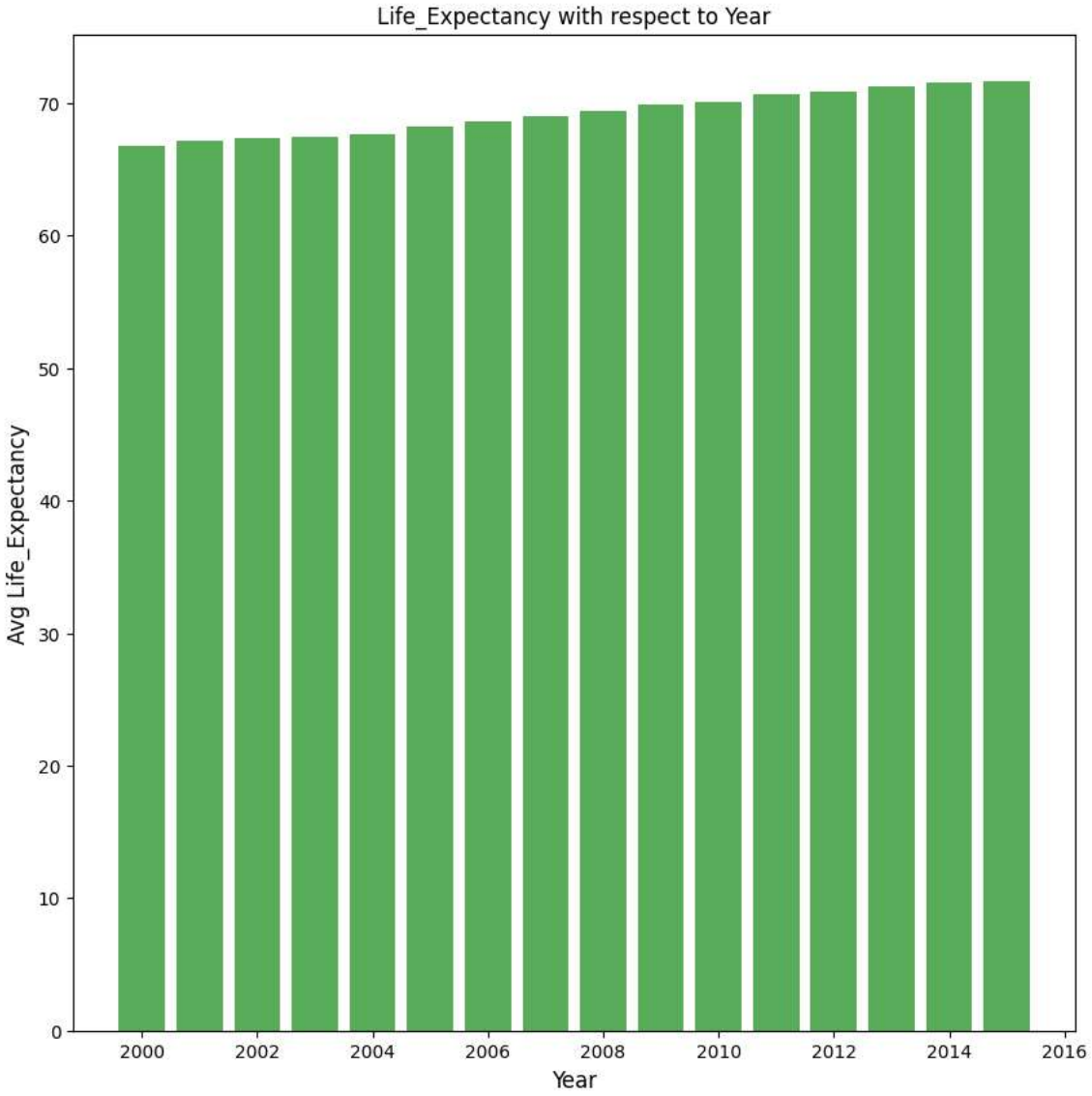


In [25]:
```
plt.figure(figsize=(10,10))
plt.bar(data.groupby('Status')['Status'].count().index, data.groupby('Status')['Life_Expectancy'].mean(), color="red")
plt.xlabel("Status",fontsize=12)
plt.ylabel("Avg Life_Expectancy",fontsize=12)
plt.title("Life_Expectancy with respect to Status")
plt.show()
```

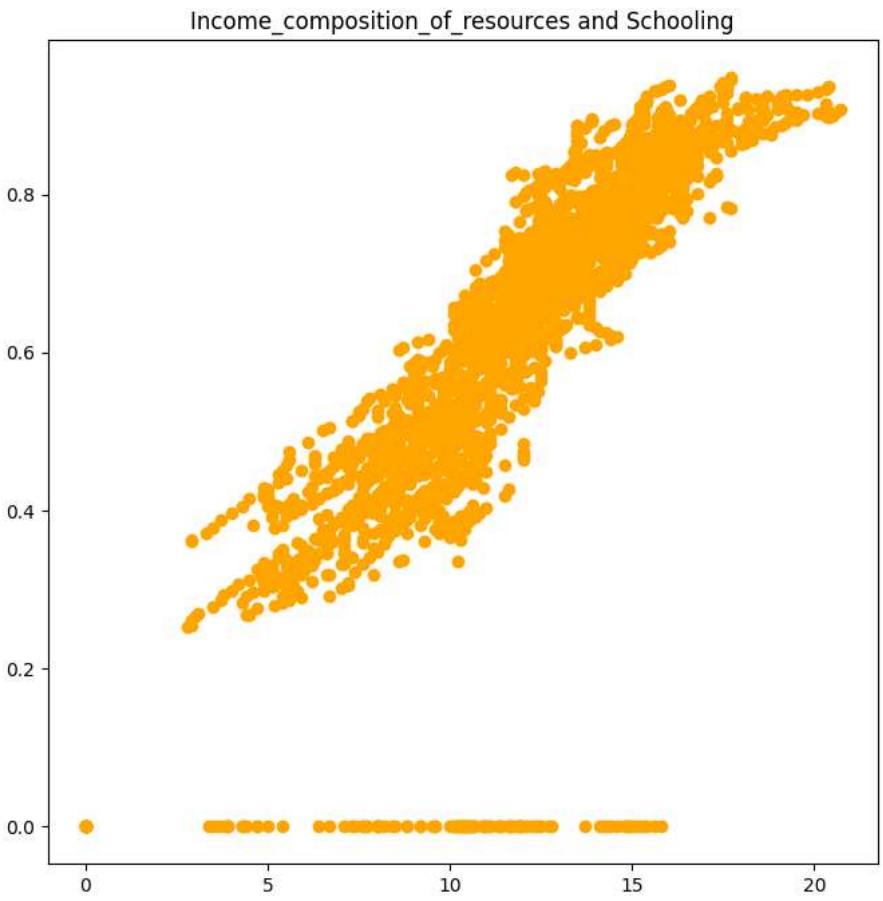


Above Plot:
The plot above gives us a relationship between the Status which is DEVELOPING and DEVELOPED with the Life Expectancies we have. We can analyze the bar plot and say that the life expectany of the developed countries is comparatively higher than the life expectancy of the developing countries. This might definitely be due to several reasons or factors of considerations that we would further experiement and explore.

In [26]:
```
# Life_Expectancy with respect to Year using bar plot.
plt.figure(figsize=(10,10))
plt.bar(data.groupby('Year')['Year'].count().index, data.groupby('Year')['Life_Expectancy'].mean(),color='green',alpha=0.65)
plt.xlabel("Year",fontsize=12)
plt.ylabel("Avg Life_Expectancy",fontsize=12)
plt.title("Life_Expectancy with respect to Year")
plt.show()
```



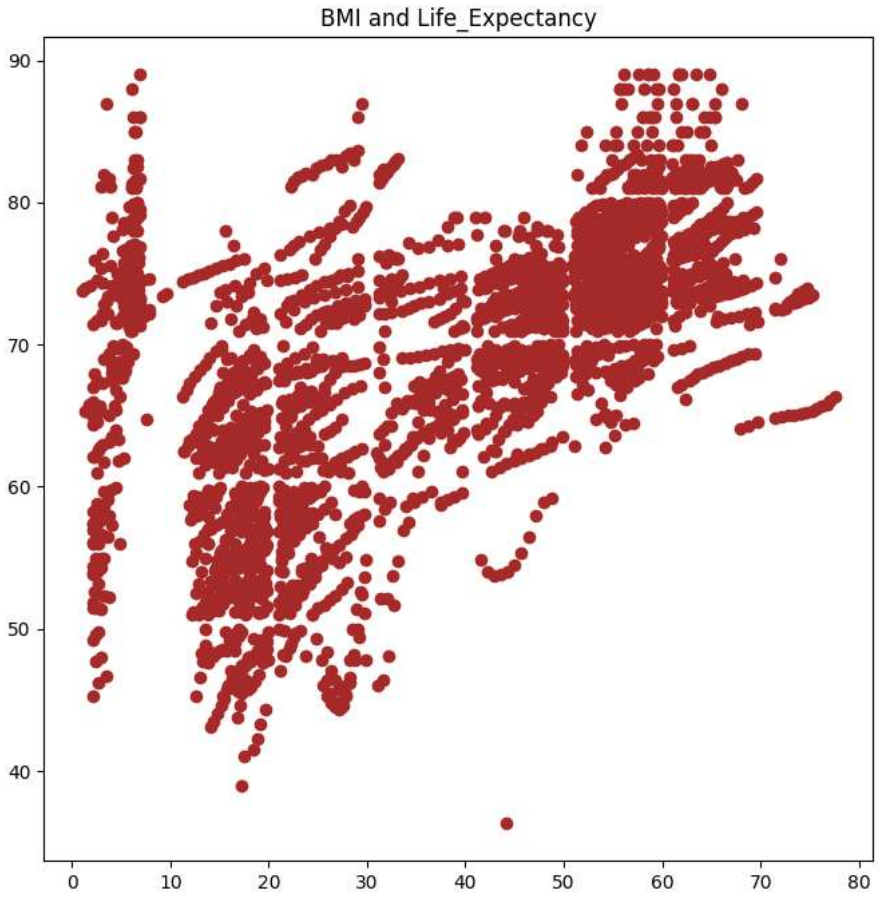
In [27]:
```
plt.figure(figsize=(8, 8))
plt.scatter(data["Schooling"], data["Income_composition_of_resources"], color="orange")
plt.title("Income_composition_of_resources and Schooling")
plt.show()
```



Above Plot:
The Above checks if there is a relationship between SCHOOLING and INCOME COMPOSITION OF RESOURCES. I believe it is hard to say if there is any relationship because we cannot really see any trend.

Observations:
Life expectancy has a negative correlation with adult mortality.
Life expectancy has a strong correlations with schooling and income composition of resourses.
We can also observe strong correlations between thinness_1-19_years and thinness_5-9_years.
There is a non-negligible correkation between life expectancy and Body Mass Index (BMI). and body decreases.

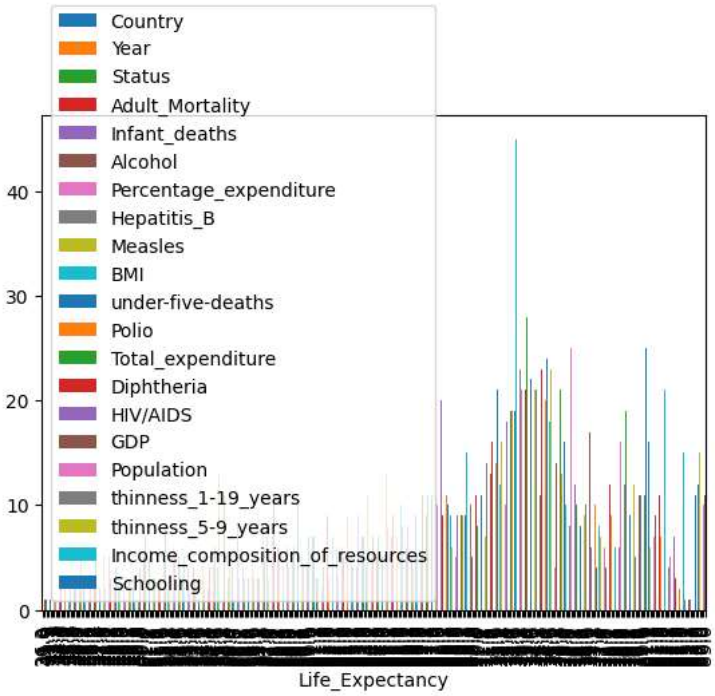
In [28]:
```
plt.figure(figsize=(8, 8))
plt.scatter(data["BMI"], data["Life_Expectancy"], color="brown")
plt.title("BMI and Life_Expectancy")
plt.show()
```



Above Plot:
The Above checks if there is a relationship between BODY MASS INDEX (BMI) and LIFE EXPECTANCY. I believe it is hard to say if there is any relationship because we cannot really see any trend.

In [29]:
```
data.groupby('Life_Expectancy').count().plot(kind='bar')
```

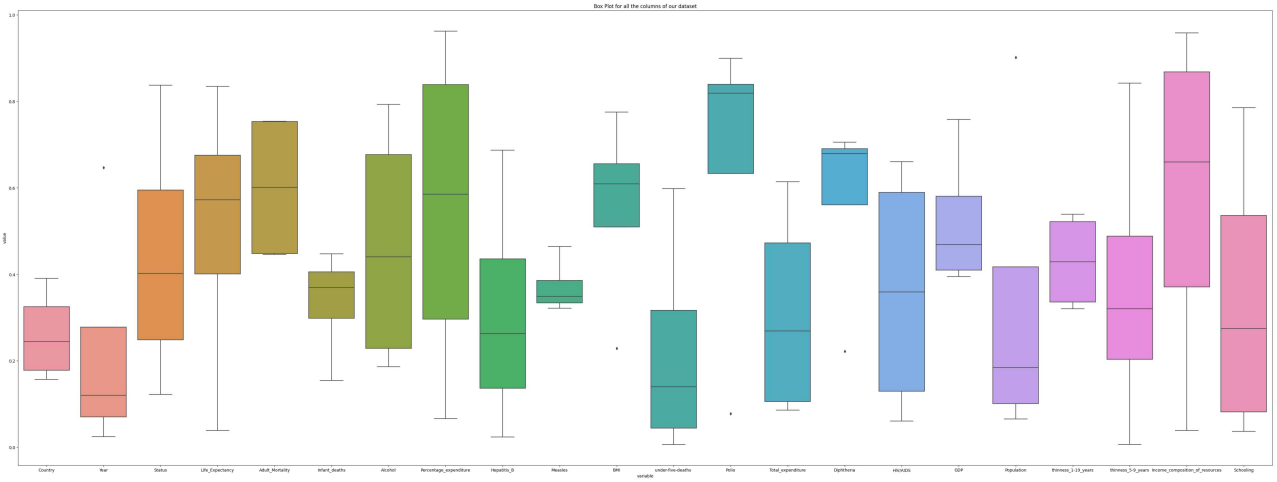
Out[29]: <AxesSubplot: xlabel='Life_Expectancy'>



Boxplot of all the column of our dataset

In [30]:
```
data1 = pd.DataFrame(data = np.random.random(size=(4,22)), columns = ['Country', 'Year', 'Status', 'Life_Expectancy',
                'Adult_Mortality','Infant_deaths', 'Alcohol', 'Percentage_expenditure', 'Hepatitis_B','Measles','BMI',
                'under-five-deaths ', 'Polio', 'Total_expenditure','Diphtheria', 'HIV/AIDS', 'GDP', 'Population',
                'thinness_1-19_years', 'thinness_5-9_years','Income_composition_of_resources', 'Schooling'])

fig, ax = plt.subplots(figsize=(55,20))
sns.boxplot(x="variable", y="value", data=pd.melt(data1)).set(title='Box Plot for all the columns of our dataset')
```

Out[30]: [Text(0.5, 1.0, 'Box Plot for all the columns of our dataset')]

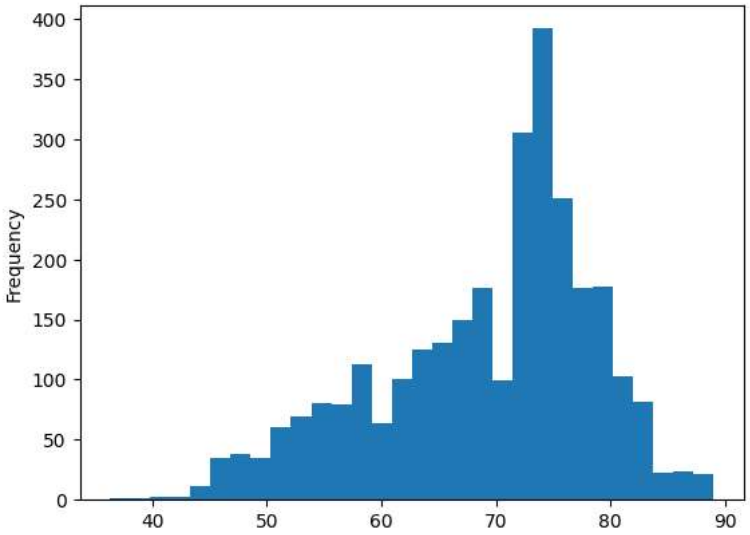


Above Plot:

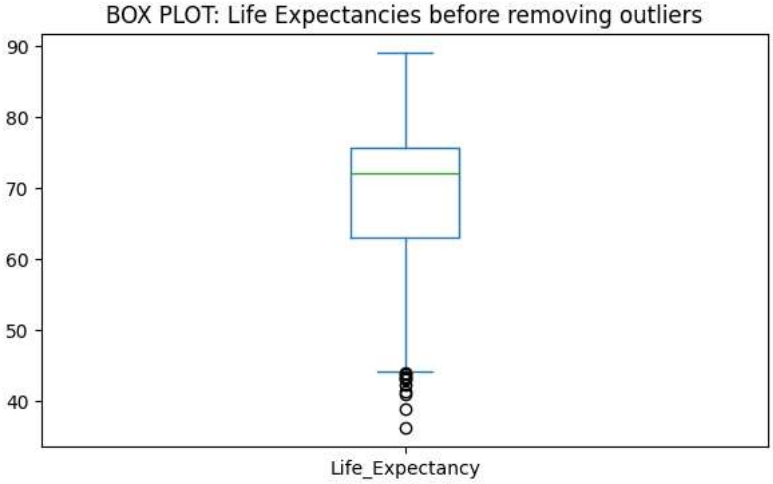
Above we have plotted using box plot all the columns of our dataframe which gives us an inituitive analysis of how related and differently related our datasets are in terms of mean, median and many more statistical terms and dependent and independent variables. We can also see how the attributes or columns in our dataset or dataframe value on the basis of the values they are holding and also the relationships which is the motive of our project.

In [31]:
```
data['Life_Expectancy'].plot(kind='hist', bins=30)
```

Out[31]: <AxesSubplot: ylabel='Frequency'>

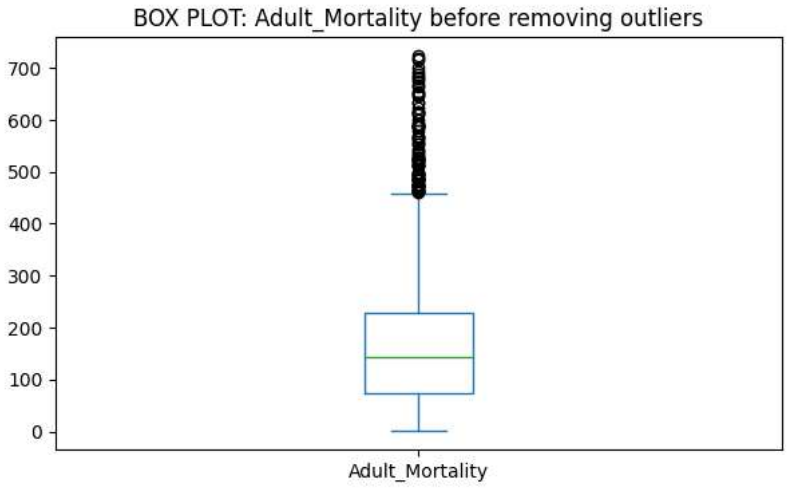


In [32]:
```
data['Life_Expectancy'].plot(kind='box', title='BOX PLOT: Life Expectancies before removing outliers',figsize=(7,4))
plt.show()
```



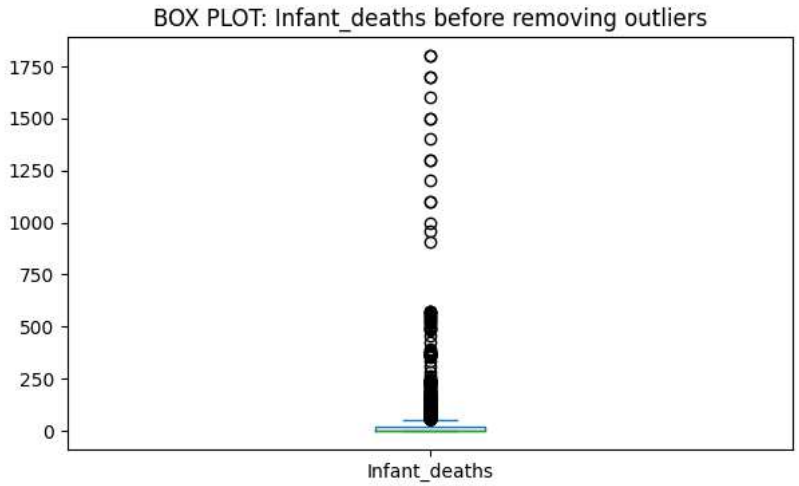
Above Plot:
Here we made a BOX PLOT for Life Expectancies as we will now be working and analyzing the outliers. We see that Life Expectancy column in our dataframe does have outliers.

In [33]:
```
data['Adult_Mortality'].plot(kind='box', title='BOX PLOT: Adult_Mortality before removing outliers',figsize=(7,4))
plt.show()
```



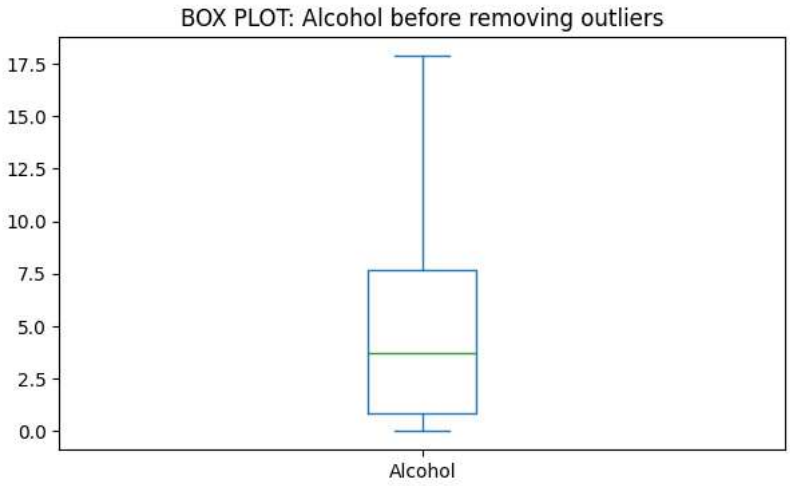
Above Plot:
Here we made a BOX PLOT for Adult Mortality as we will now be working and analyzing the outliers. We see that Adult Mortality column in our dataframe does have outliers.

In [34]:
```
data['Infant_deaths'].plot(kind='box', title='BOX PLOT: Infant_deaths before removing outliers',figsize=(7,4))
plt.show()
```



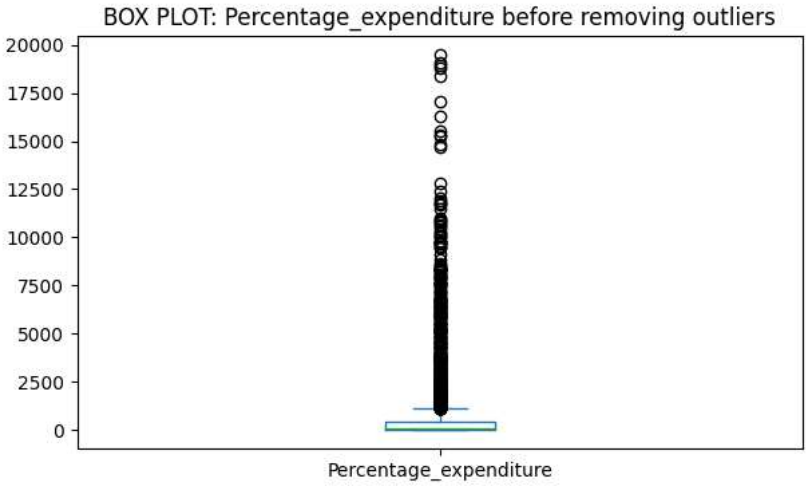
Above Plot:
Here we made a BOX PLOT for Infant deaths as we will now be working and analyzing the outliers. We see that Infant deaths column in our dataframe does have outliers.

In [35]:
```
data['Alcohol'].plot(kind='box', title='BOX PLOT: Alcohol before removing outliers',figsize=(7,4))
plt.show()
```



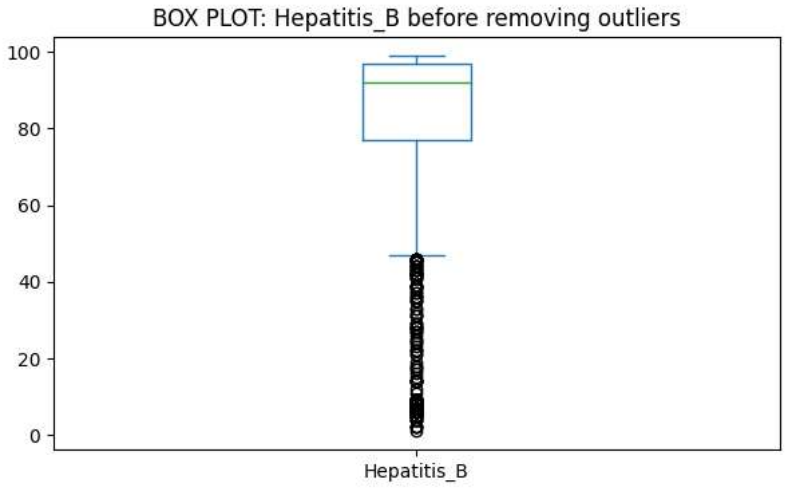
Above Plot:
Here we made a BOX PLOT for Alcohol as we will now be working and analyzing the outliers. We see that Alcohol column in our dataframe does have outliers.

In [36]:
```
data['Percentage_expenditure'].plot(kind='box', title='BOX PLOT: Percentage_expenditure before removing outliers',figsize=(7,
plt.show()
```



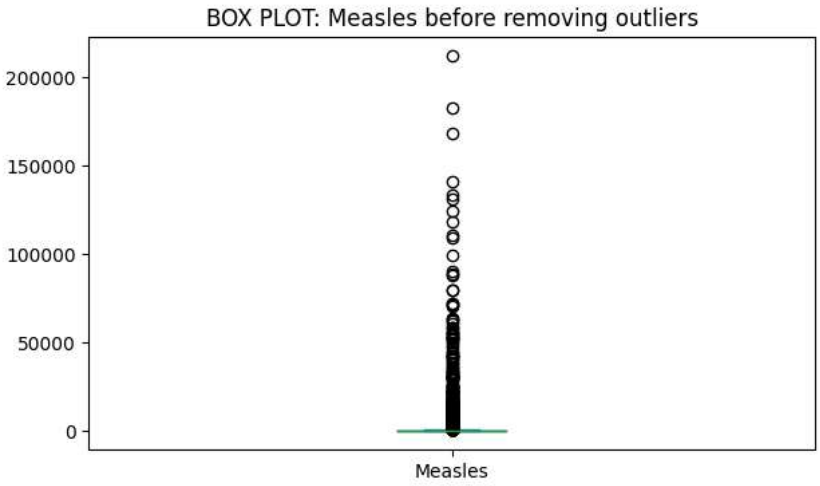
Above Plot:
Here we made a BOX PLOT for Percentage Expenditure as we will now be working and analyzing the outliers. We see that Percentage Expenditure column in our dataframe does have outliers.

In [37]:
```
data['Hepatitis_B'].plot(kind='box', title='BOX PLOT: Hepatitis_B before removing outliers',figsize=(7,4))
plt.show()
```



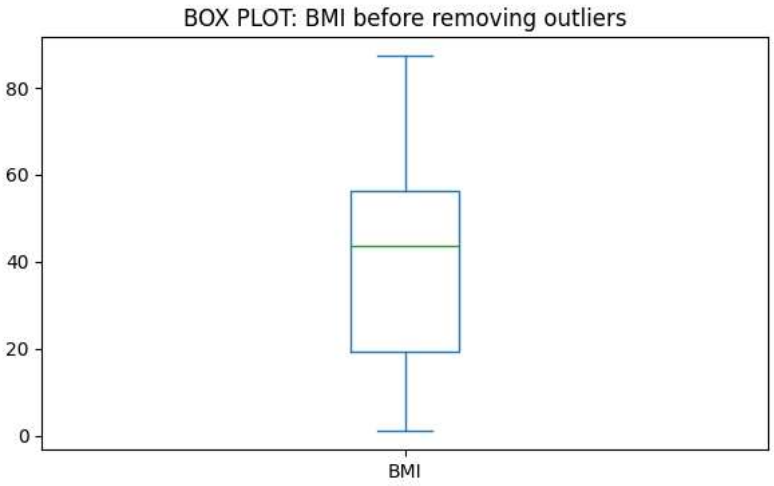
Above Plot:
Here we made a BOX PLOT for Hepatitis_B as we will now be working and analyzing the outliers. We see that Hepatitis_B column in our dataframe does have outliers.

In [38]:
```
data['Measles'].plot(kind='box', title='BOX PLOT: Measles before removing outliers',figsize=(7,4))
plt.show()
```



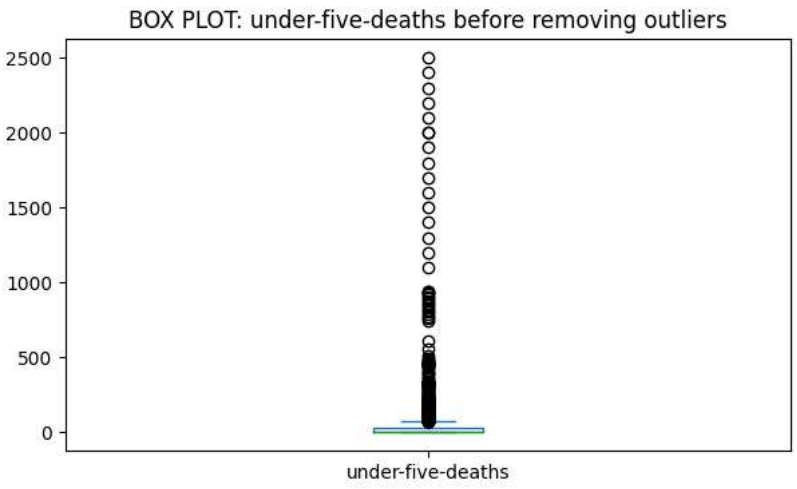
Above Plot:
Here we made a BOX PLOT for Measles as we will now be working and analyzing the outliers. We see that Measles column in our dataframe does have outliers.

In [39]:
```
data['BMI'].plot(kind='box', title='BOX PLOT: BMI before removing outliers',figsize=(7,4))
plt.show()
```



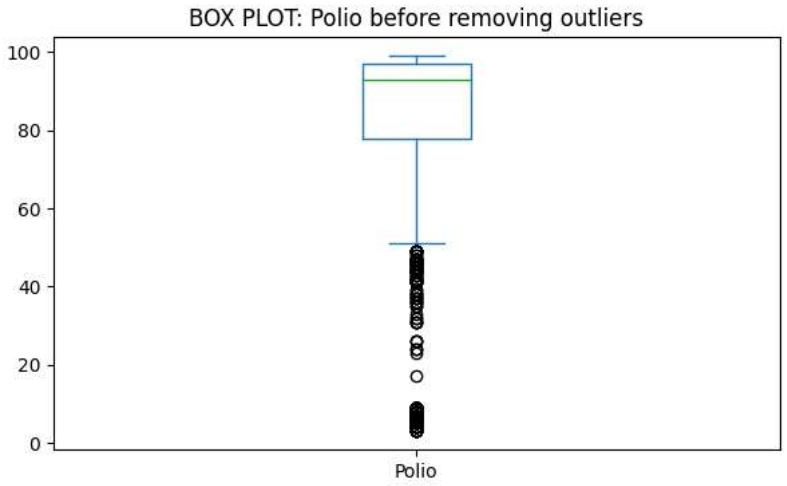
Above Plot:
Here we made a BOX PLOT for BMI as we will now be working and analyzing the outliers. We see that BMI column in our dataframe does have outliers.

In [40]:
```
data['under-five-deaths'].plot(kind='box', title='BOX PLOT: under-five-deaths before removing outliers',figsize=(7,4))
plt.show()
```



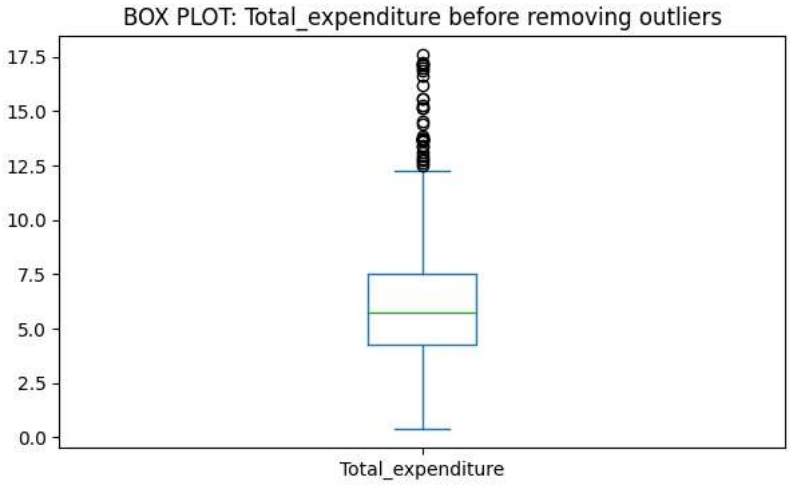
Above Plot:
Here we made a BOX PLOT for under-five-deaths as we will now be working and analyzing the outliers. We see that under-five-deaths column in our dataframe does have outliers.

In [41]:
```
data['Polio'].plot(kind='box', title='BOX PLOT: Polio before removing outliers',figsize=(7,4))
plt.show()
```



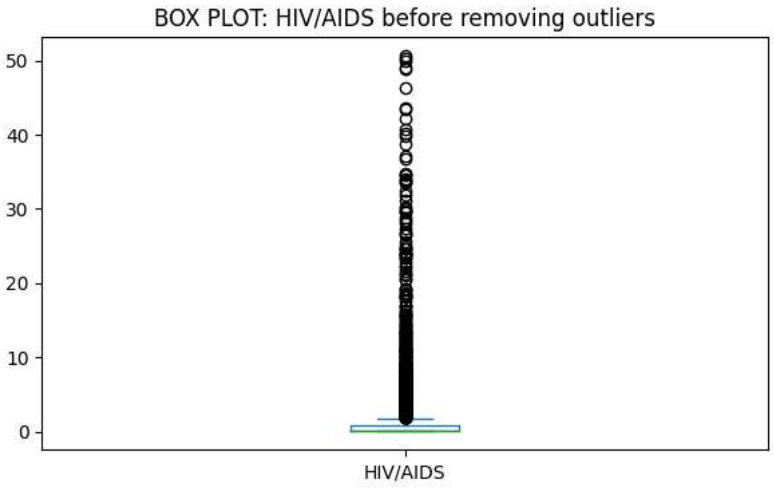
Above Plot:
Here we made a BOX PLOT for Polio as we will now be working and analyzing the outliers. We see that Polio column in our dataframe does have outliers.

In [42]:
```
data['Total_expenditure'].plot(kind='box', title='BOX PLOT: Total_expenditure before removing outliers',figsize=(7,4))
plt.show()
```



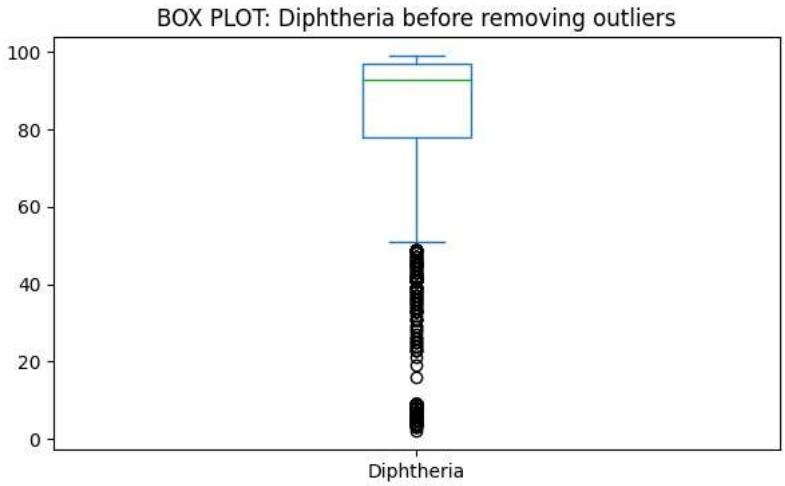
Above Plot:
Here we made a BOX PLOT for Total_expenditure as we will now be working and analyzing the outliers. We see that Total_expenditure column in our dataframe does have outliers.

In [43]:
```
data['HIV/AIDS'].plot(kind='box', title='BOX PLOT: HIV/AIDS before removing outliers',figsize=(7,4))
plt.show()
```



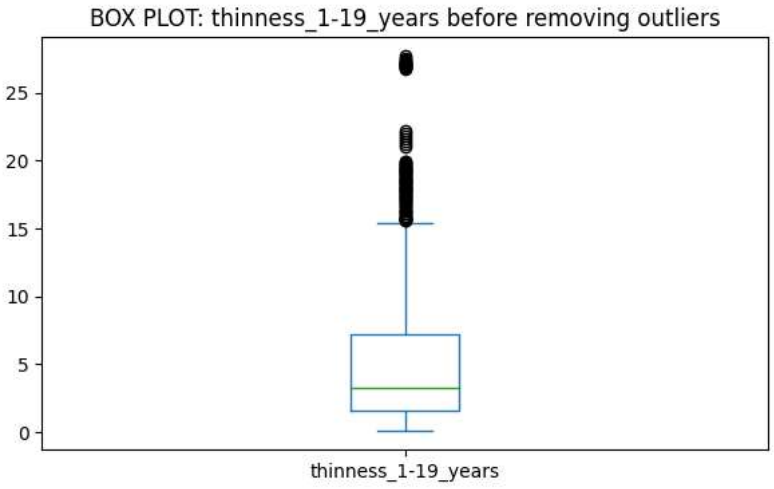
Above Plot:
Here we made a BOX PLOT for HIV/AIDS as we will now be working and analyzing the outliers. We see that HIV/AIDS column in our dataframe does have outliers.

In [44]:
```
data['Diphtheria'].plot(kind='box', title='BOX PLOT: Diphtheria before removing outliers',figsize=(7,4))
plt.show()
```



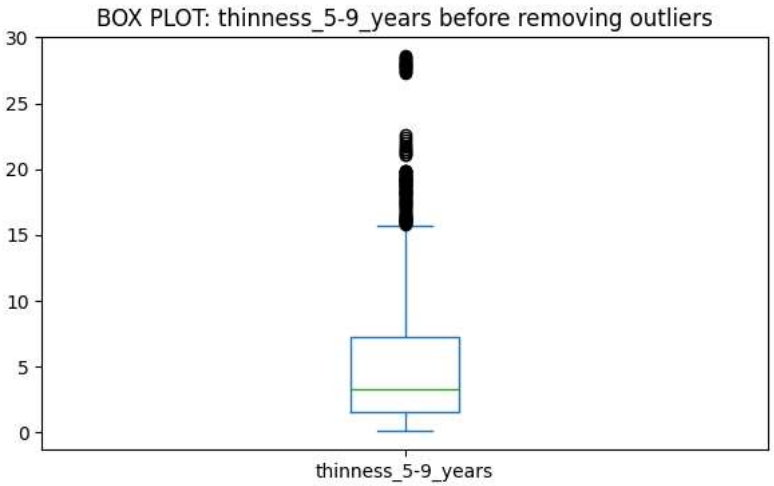
Above Plot:
Here we made a BOX PLOT for Diphtheria as we will now be working and analyzing the outliers. We see that Diphtheria column in our dataframe does have outliers.

In [45]:
```
data['thinness_1-19_years'].plot(kind='box', title='BOX PLOT: thinness_1-19_years before removing outliers',figsize=(7,4))
plt.show()
```



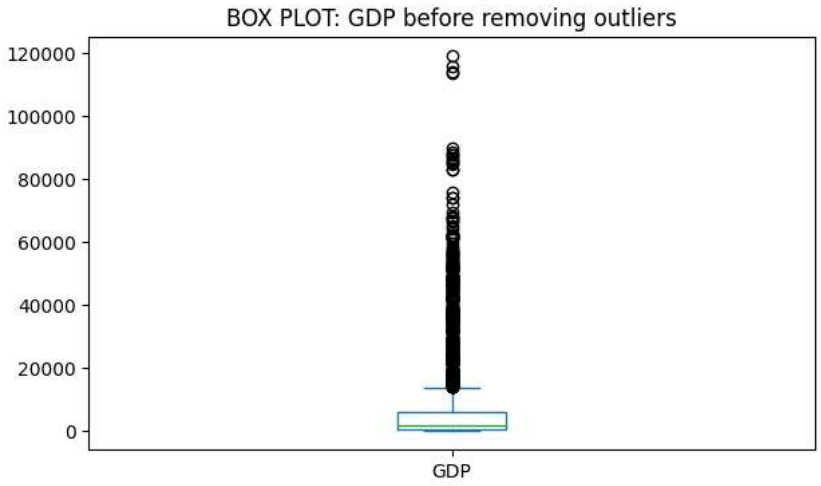
Above Plot:
Here we made a BOX PLOT for thinness_1-19_years as we will now be working and analyzing the outliers. We see that thinness_1-19_years column in our dataframe does have outliers.

In [46]:
```
data['thinness_5-9_years'].plot(kind='box', title='BOX PLOT: thinness_5-9_years before removing outliers',figsize=(7,4))
plt.show()
```



Above Plot:
Here we made a BOX PLOT for thinness_5-9_years as we will now be working and analyzing the outliers. We see that thinness_5-9_years column in our dataframe does have outliers.

In [47]:
```
data['GDP'].plot(kind='box', title='BOX PLOT: GDP before removing outliers',figsize=(7,4))
plt.show()
```



Above Plot:
Here we made a BOX PLOT for GDP as we will now be working and analyzing the outliers. We see that GDP column in our dataframe does have outliers.

DATA CLEANING

In [48]:
```
countries = data.Country.unique()
```

In [49]:
```
columns = ['Year', 'Status', 'Life_Expectancy', 'Adult_Mortality','Infant_deaths', 'Alcohol',
                'Percentage_expenditure', 'Hepatitis_B','Measles','BMI', 'under-five-deaths', 'Polio',
                'Total_expenditure','Diphtheria', 'HIV/AIDS', 'GDP', 'Population','thinness_1-19_years',
                'thinness_5-9_years','Income_composition_of_resources', 'Schooling']
```

Treat null values using interpolation.

In [50]:
```
for c in countries:
    data.loc[data['Country'] == c,columns] = data.loc[data['Country'] == c,columns].interpolate()
```

In [51]:
```
data[np.isnan(data['Life_Expectancy'])]
```

Out[51]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_e
624	Cook Islands	2013	Developing	NaN	NaN	0	0.01	0.000000	98.0	0	...	98.0	
769	Dominica	2013	Developing	NaN	NaN	0	0.01	11.419555	96.0	0	...	96.0	
1650	Marshall Islands	2013	Developing	NaN	NaN	0	0.01	871.878317	8.0	0	...	79.0	
1715	Monaco	2013	Developing	NaN	NaN	0	0.01	0.000000	99.0	0	...	99.0	
1812	Nauru	2013	Developing	NaN	NaN	0	0.01	15.606596	87.0	0	...	87.0	
1909	Niue	2013	Developing	NaN	NaN	0	0.01	0.000000	99.0	0	...	99.0	
1958	Palau	2013	Developing	NaN	NaN	0	NaN	344.690631	99.0	0	...	99.0	
2167	Saint Kitts and Nevis	2013	Developing	NaN	NaN	0	8.54	0.000000	97.0	0	...	96.0	
2216	San Marino	2013	Developing	NaN	NaN	0	0.01	0.000000	69.0	0	...	69.0	
2713	Tuvalu	2013	Developing	NaN	NaN	0	0.01	78.281203	9.0	0	...	9.0	

10 rows × 22 columns

Droping the rows which has null or nan values with respect to our Life Expectancy column in our dataframe

In [52]:
```
data = data.drop(data.index[[624, 769, 1650,1715,1812,1909,1958,2167,2216,2713]])
```

In [53]:
```
data[np.isnan(data['Adult_Mortality'])]
```

Out[53]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [54]:
```
data[np.isnan(data['Infant_deaths'])]
```

Out[54]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [55]:
```
data[np.isnan(data.Alcohol)]
```

Out[55]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_
32	Algeria	2015	Developing	75.6	19.0	21	NaN	0.0	95.0	63	...	95.0	
48	Angola	2015	Developing	52.4	335.0	66	NaN	0.0	64.0	118	...	7.0	
64	Antigua and Barbuda	2015	Developing	76.4	13.0	0	NaN	0.0	99.0	0	...	86.0	
80	Argentina	2015	Developing	76.3	116.0	8	NaN	0.0	94.0	0	...	93.0	
96	Armenia	2015	Developing	74.8	118.0	1	NaN	0.0	94.0	33	...	96.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2858	Venezuela (Bolivarian Republic of)	2015	Developing	74.1	157.0	9	NaN	0.0	87.0	0	...	87.0	
2874	Viet Nam	2015	Developing	76.0	127.0	28	NaN	0.0	97.0	256	...	97.0	
2890	Yemen	2015	Developing	65.7	224.0	37	NaN	0.0	69.0	468	...	63.0	
2906	Zambia	2015	Developing	61.8	33.0	27	NaN	0.0	9.0	9	...	9.0	
2922	Zimbabwe	2015	Developing	67.0	336.0	22	NaN	0.0	87.0	0	...	88.0	

192 rows × 22 columns

In [56]:
```
data['Alcohol'] = data['Alcohol'].replace('-', np.nan)
data = data.dropna(axis=0, subset=['Alcohol'])
```

In [57]:
```
data[np.isnan(data['Percentage_expenditure'])]
```

Out[57]:

Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [58]:
```
data[np.isnan(data['Hepatitis_B'])]
```

Out[58]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_ex
738	Denmark	2014	Developed	84.0	73.0	0	9.64	10468.762920	NaN	27	...	94.0	
739	Denmark	2013	Developed	81.0	75.0	0	9.50	10261.763000	NaN	17	...	94.0	
740	Denmark	2012	Developed	80.0	76.0	0	9.26	928.417079	NaN	2	...	94.0	
741	Denmark	2011	Developed	79.7	79.0	0	10.47	10251.108720	NaN	84	...	91.0	
742	Denmark	2010	Developed	79.2	84.0	0	10.28	954.486593	NaN	5	...	9.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2773	United Kingdom of Great Britain and Northern I...	2004	Developed	78.8	83.0	4	12.22	0.000000	NaN	189	...	92.0	
2774	United Kingdom of Great Britain and Northern I...	2003	Developed	78.3	86.0	4	11.85	0.000000	NaN	460	...	91.0	
2775	United Kingdom of Great Britain and Northern I...	2002	Developed	78.2	87.0	4	11.44	0.000000	NaN	314	...	91.0	
2776	United Kingdom of Great Britain and Northern I...	2001	Developed	78.0	88.0	4	10.91	0.000000	NaN	73	...	91.0	
2777	United Kingdom of Great Britain and Northern I...	2000	Developed	77.8	89.0	4	10.59	0.000000	NaN	104	...	91.0	

137 rows × 22 columns

In [59]:
```
data['Hepatitis_B'] = data['Hepatitis_B'].replace('-', np.nan)
data = data.dropna(axis=0, subset=['Hepatitis_B'])
```

In [60]:
```
data[np.isnan(data['Measles'])]
```

Out[60]:

Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [61]:
```
data[np.isnan(data['BMI'])]
```

Out[61]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_e:
2458	Sudan	2014	Developing	63.8	229.0	59	0.01	253.608651	94.0	676	...	94.0	
2459	Sudan	2013	Developing	63.5	232.0	60	0.01	227.835321	93.0	2813	...	93.0	
2460	Sudan	2012	Developing	63.2	235.0	61	0.01	220.522192	92.0	8523	...	92.0	
2461	Sudan	2011	Developing	62.7	241.0	61	2.12	196.689215	93.0	5616	...	93.0	
2462	Sudan	2010	Developing	62.5	243.0	62	1.77	172.009788	75.0	680	...	9.0	
2463	Sudan	2009	Developing	62.0	248.0	63	1.99	17.053693	72.0	68	...	81.0	
2464	Sudan	2008	Developing	61.8	251.0	64	2.01	128.636271	78.0	129	...	85.0	
2465	Sudan	2007	Developing	61.4	254.0	65	2.01	86.131669	78.0	327	...	84.0	
2466	Sudan	2006	Developing	61.0	26.0	66	1.90	60.336857	6.0	228	...	77.0	
2467	Sudan	2005	Developing	67.0	261.0	66	1.55	37.590396	22.0	1374	...	78.0	
2468	Sudan	2004	Developing	59.7	278.0	68	1.59	37.044800	22.0	9562	...	74.0	
2469	Sudan	2003	Developing	59.6	278.0	69	1.74	35.352647	22.0	4381	...	69.0	
2470	Sudan	2002	Developing	59.4	277.0	70	1.59	30.622875	22.0	4529	...	6.0	
2471	Sudan	2001	Developing	58.9	283.0	71	1.81	28.880697	22.0	4362	...	66.0	
2472	Sudan	2000	Developing	58.6	284.0	71	1.76	30.860010	22.0	2875	...	62.0	

15 rows × 22 columns

In [62]:
```
data['BMI'] = data['BMI'].replace('-', np.nan)
data = data.dropna(axis=0, subset=['BMI'])
```

In [63]:
```
data[np.isnan(data['under-five-deaths'])]
```

Out[63]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [64]:
```
data[np.isnan(data['Polio'])]
```

Out[64]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [65]:
```
data[np.isnan(data['Total_expenditure'])]
```

Out[65]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total
706	Democratic People's Republic of Korea	2014	Developing	73.0	142.0	6	0.01	0.0	93.0	3	...	99.0	
707	Democratic People's Republic of Korea	2013	Developing	71.0	146.0	6	3.35	0.0	93.0	0	...	99.0	
708	Democratic People's Republic of Korea	2012	Developing	69.8	149.0	7	3.61	0.0	96.0	0	...	99.0	
709	Democratic People's Republic of Korea	2011	Developing	69.4	153.0	8	3.39	0.0	94.0	0	...	99.0	
710	Democratic People's Republic of Korea	2010	Developing	69.0	157.0	8	3.12	0.0	93.0	0	...	99.0	
711	Democratic People's Republic of Korea	2009	Developing	68.7	161.0	9	3.35	0.0	93.0	0	...	98.0	
712	Democratic People's Republic of Korea	2008	Developing	68.6	164.0	9	3.16	0.0	92.0	8	...	98.0	
713	Democratic People's Republic of Korea	2007	Developing	68.5	166.0	9	3.13	0.0	92.0	3550	...	99.0	
714	Democratic People's Republic of Korea	2006	Developing	68.5	165.0	10	3.28	0.0	96.0	0	...	98.0	
715	Democratic People's Republic of Korea	2005	Developing	68.5	165.0	10	3.21	0.0	92.0	0	...	97.0	
716	Democratic People's Republic of Korea	2004	Developing	68.4	165.0	11	3.13	0.0	98.0	0	...	99.0	
717	Democratic People's Republic of Korea	2003	Developing	68.1	165.0	12	3.13	0.0	27.0	0	...	99.0	
718	Democratic People's Republic of Korea	2002	Developing	67.6	167.0	14	3.08	0.0	27.0	0	...	99.0	
719	Democratic People's Republic of Korea	2001	Developing	66.6	177.0	16	2.53	0.0	27.0	0	...	98.0	
720	Democratic People's Republic of Korea	2000	Developing	65.4	192.0	18	3.52	0.0	27.0	0	...	93.0	
1845	New Zealand	2015	Developed	81.6	66.0	0	8.70	0.0	92.0	10	...	92.0	
2313	Singapore	2015	Developed	83.1	55.0	0	1.79	0.0	96.0	0	...	96.0	
2378	Somalia	2014	Developing	54.3	321.0	51	0.01	0.0	42.0	10229	...	47.0	
2379	Somalia	2013	Developing	54.2	318.0	51	0.01	0.0	42.0	3173	...	47.0	
2380	Somalia	2012	Developing	53.1	336.0	51	0.01	0.0	42.0	9983	...	47.0	
2381	Somalia	2011	Developing	53.1	329.0	51	0.01	0.0	42.0	17298	...	49.0	
2382	Somalia	2010	Developing	52.4	336.0	52	0.01	0.0	42.0	115	...	49.0	
2383	Somalia	2009	Developing	52.2	335.0	52	0.01	0.0	42.0	13	...	41.0	
2384	Somalia	2008	Developing	51.9	336.0	52	0.01	0.0	42.0	1081	...	4.0	
2385	Somalia	2007	Developing	51.5	34.0	52	0.01	0.0	42.0	1149	...	4.0	
2386	Somalia	2006	Developing	51.5	337.0	51	0.01	0.0	42.0	7	...	26.0	
2387	Somalia	2005	Developing	51.6	334.0	50	0.01	0.0	42.0	0	...	35.0	
2388	Somalia	2004	Developing	51.2	341.0	49	0.01	0.0	42.0	12008	...	3.0	
2389	Somalia	2003	Developing	51.1	344.0	48	0.01	0.0	42.0	8257	...	4.0	

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total
2390	Somalia	2002	Developing	58.0	348.0	47	0.01	0.0	42.0	9559	...	4.0	
2391	Somalia	2001	Developing	57.0	352.0	46	0.01	0.0	42.0	3571	...	33.0	
2392	Somalia	2000	Developing	55.0	355.0	45	0.01	0.0	42.0	3965	...	37.0	

32 rows × 22 columns

In [66]:
```
data['Total_expenditure'] = data['Total_expenditure'].replace('-', np.nan)
data = data.dropna(axis=0, subset=['Total_expenditure'])
```

In [67]:
```
data[np.isnan(data['Diphtheria'])]
```

Out[67]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [68]:
```
data[np.isnan(data['HIV/AIDS'])]
```

Out[68]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [69]:
```
data[np.isnan(data['GDP'])]
```

Out[69]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_e
161	Bahamas	2014	Developing	75.4	16.0	0	9.45	0.0	96.0	0	...	96.0	
162	Bahamas	2013	Developing	74.8	172.0	0	9.42	0.0	97.0	0	...	97.0	
163	Bahamas	2012	Developing	74.9	167.0	0	9.50	0.0	96.0	0	...	99.0	
164	Bahamas	2011	Developing	75.0	162.0	0	9.34	0.0	95.0	0	...	97.0	
165	Bahamas	2010	Developing	75.0	161.0	0	9.19	0.0	98.0	0	...	97.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2901	Yemen	2004	Developing	62.2	247.0	42	0.06	0.0	43.0	12708	...	72.0	
2902	Yemen	2003	Developing	61.9	249.0	43	0.04	0.0	38.0	8536	...	61.0	
2903	Yemen	2002	Developing	61.5	25.0	45	0.07	0.0	31.0	890	...	64.0	
2904	Yemen	2001	Developing	61.1	251.0	46	0.08	0.0	19.0	485	...	73.0	
2905	Yemen	2000	Developing	68.0	252.0	48	0.07	0.0	14.0	0	...	74.0	

358 rows × 22 columns

In [70]:
```
data['GDP'] = data['GDP'].replace('-', np.nan)
data = data.dropna(axis=0, subset=['GDP'])
```

In [71]:
```
data[np.isnan(data['Population'])]
```

Out[71]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_e
65	Antigua and Barbuda	2014	Developing	76.2	131.0	0	8.56	2422.999774	99.0	0	...	96.0	
66	Antigua and Barbuda	2013	Developing	76.1	133.0	0	8.58	1991.430372	99.0	0	...	98.0	
67	Antigua and Barbuda	2012	Developing	75.9	134.0	0	8.18	2156.229842	98.0	0	...	97.0	
68	Antigua and Barbuda	2011	Developing	75.7	136.0	0	7.84	1810.875316	99.0	0	...	99.0	
69	Antigua and Barbuda	2010	Developing	75.6	138.0	0	7.84	1983.956937	98.0	0	...	99.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2757	United Arab Emirates	2004	Developing	75.1	95.0	1	1.77	2972.448675	92.0	22	...	94.0	
2758	United Arab Emirates	2003	Developing	74.9	98.0	1	1.74	277.181833	92.0	42	...	94.0	
2759	United Arab Emirates	2002	Developing	74.7	11.0	1	1.72	2598.842827	92.0	53	...	94.0	
2760	United Arab Emirates	2001	Developing	74.5	14.0	1	1.67	243.753913	92.0	30	...	94.0	
2761	United Arab Emirates	2000	Developing	74.2	17.0	1	1.64	262.958958	92.0	69	...	94.0	

207 rows × 22 columns

In [72]:
```
data['Population'] = data['Population'].replace('-', np.nan)
data = data.dropna(axis=0, subset=['Population'])
```

In [73]:
```
data[np.isnan(data['thinness_1-19_years'])]
```

Out[73]:

Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [74]:
```
data['thinness_1-19_years'] = data['thinness_1-19_years'].replace('-', np.nan)
data = data.dropna(axis=0, subset=['thinness_1-19_years'])
```

In [75]:
```
data[np.isnan(data['thinness_5-9_years'])]
```

Out[75]:

Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [76]:
```
data['thinness_5-9_years'] = data['thinness_5-9_years'].replace('-', np.nan)
data = data.dropna(axis=0, subset=['thinness_5-9_years'])
```

In [77]:
```
data[np.isnan(data['Income_composition_of_resources'])]
```

Out[77]:

Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [78]:
```
data['Income_composition_of_resources'] = data['Income_composition_of_resources'].replace('-', np.nan)
data = data.dropna(axis=0, subset=['Income_composition_of_resources'])
```

In [79]:
```
data[np.isnan(data['Schooling'])]
```

Out[79]:

Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_expenditur

0 rows × 22 columns

In [80]:
```
data['Schooling'] = data['Schooling'].replace('-', np.nan)
data = data.dropna(axis=0, subset=['Schooling'])
```

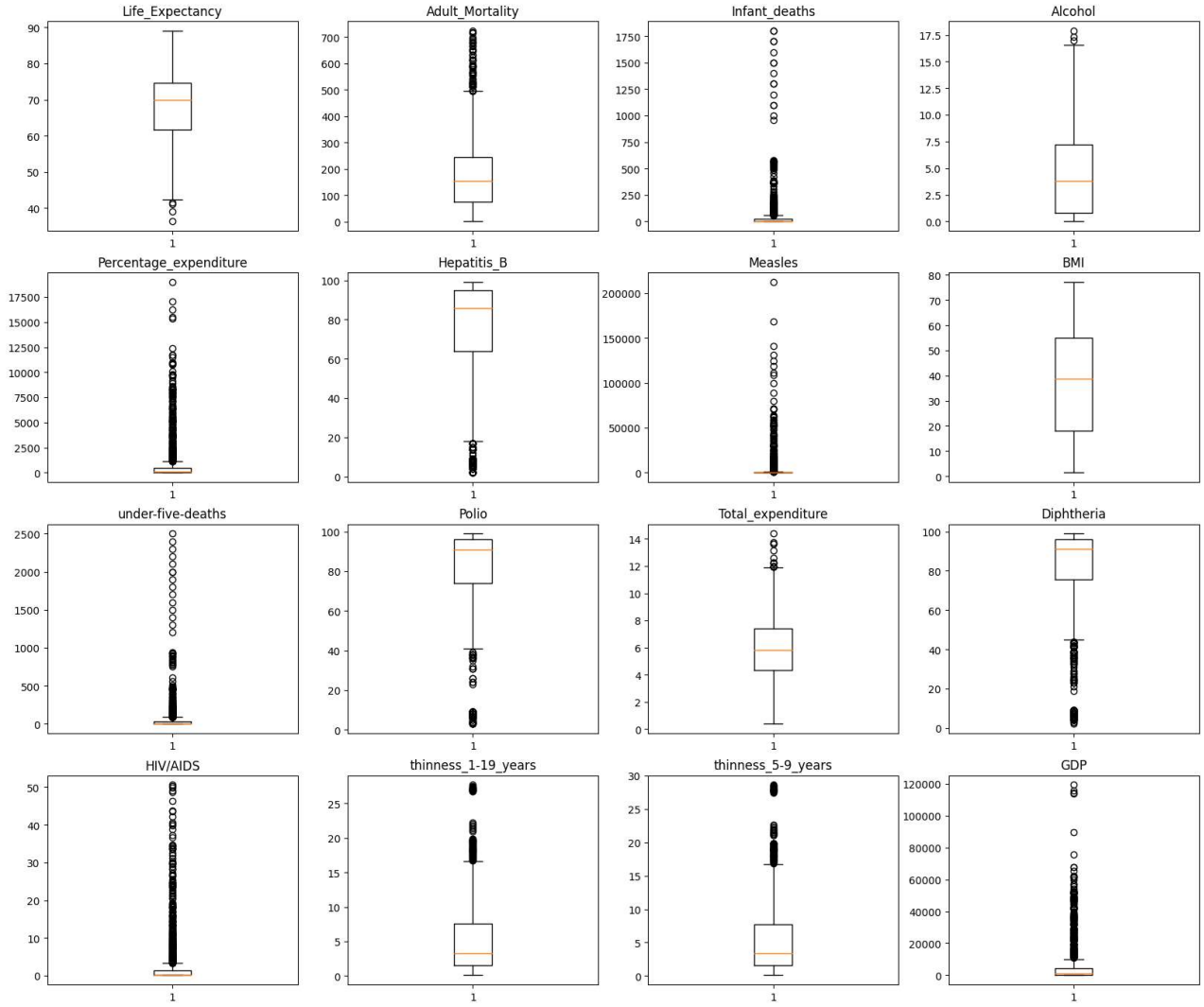
In [81]:
```
data.isnull().sum()
```

Out[81]:
```
Country                          0
Year                             0
Status                           0
Life_Expectancy                  0
Adult_Mortality                  0
Infant_deaths                    0
Alcohol                          0
Percentage_expenditure           0
Hepatitis_B                      0
Measles                          0
BMI                              0
under-five-deaths                0
Polio                            0
Total_expenditure                0
Diphtheria                       0
HIV/AIDS                         0
GDP                              0
Population                       0
thinness_1-19_years              0
thinness_5-9_years               0
Income_composition_of_resources  0
Schooling                        0
dtype: int64
```

Above we started checking for the null or missing values for every column or every attribute we have in our dataframe and further we dropped or removed the rows that had null or missing values for all the columns.

We started the data cleaning process by checking for every column if we have missing values or null or nan values and if they do we dropped or removed those. It is an important part of data cleaning because null values adversely affect the performance and accuracy of any machine learning and data science process and algorithm. Also, null values are bad because when we are applying our Machine Learning or Data Science algorithms and techniques, when we compare null values with anything the result would be unknown and out of unknown no conclusion would be made which affects our process and goal of the process and project.

In [82]:
```
# Create a dictionary of columns.
col_dict = {'Life_Expectancy':1, 'Adult_Mortality':2,'Infant_deaths':3, 'Alcohol':4,'Percentage_expenditure':5,
            'Hepatitis_B': 6,'Measles':7,'BMI':8,'under-five-deaths':9,'Polio':10,'Total_expenditure':11,
            'Diphtheria':12,'HIV/AIDS':13,'thinness_1-19_years':14, 'thinness_5-9_years': 15, 'GDP':16}

# Detect outliers in each variable using box plots.
plt.figure(figsize=(20,30))

for variable,i in col_dict.items():
                    plt.subplot(7,4,i)
                    plt.boxplot(data[variable])
                    plt.title(variable)

plt.show()
```



We can see that after one step of data cleaning and droping all the null or nan values for all the respective columns we do not have any missing data in our dataframe which is a part fo cleaning data and gives us a clean and pretty data and box plots.

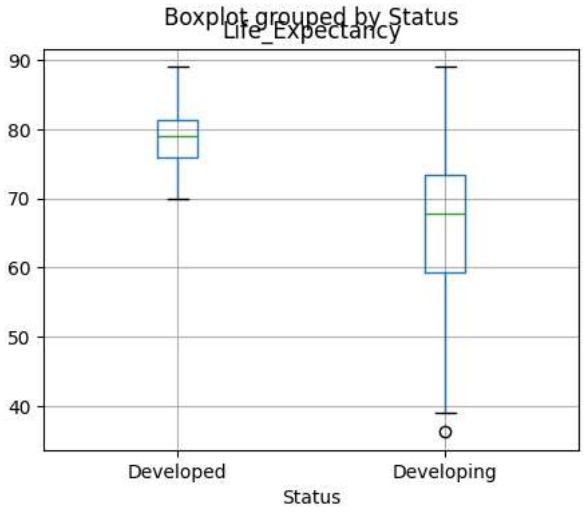
Above Plot

Above we see the BOX PLOT for all the columns which gives un an insight of how might outliers are in which portion of data for every attributes of our dataset or dataframe.

Box plot of Life Expectancy with respect to status before removing outliers

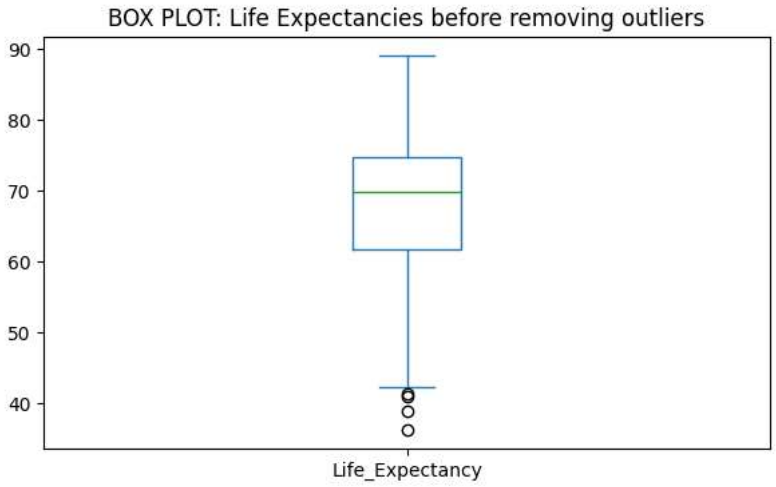
In [83]:
```
data.boxplot(by='Status',column=['Life_Expectancy'], figsize=(5,4))
```

Out[83]: <AxesSubplot: title={'center': 'Life_Expectancy'}, xlabel='Status'>



We have constructed a box plot of Life Expectancy in relation to status which is for Developing and Developed countries. We can see that the life expectancy of the developed country is definitely good and pretty compared to the life expectancy of developing countries. Also the outliers in the developing countries shows us that developed countries have better analysis of life expectancy because we know that outliers are not good to have in our dataset.

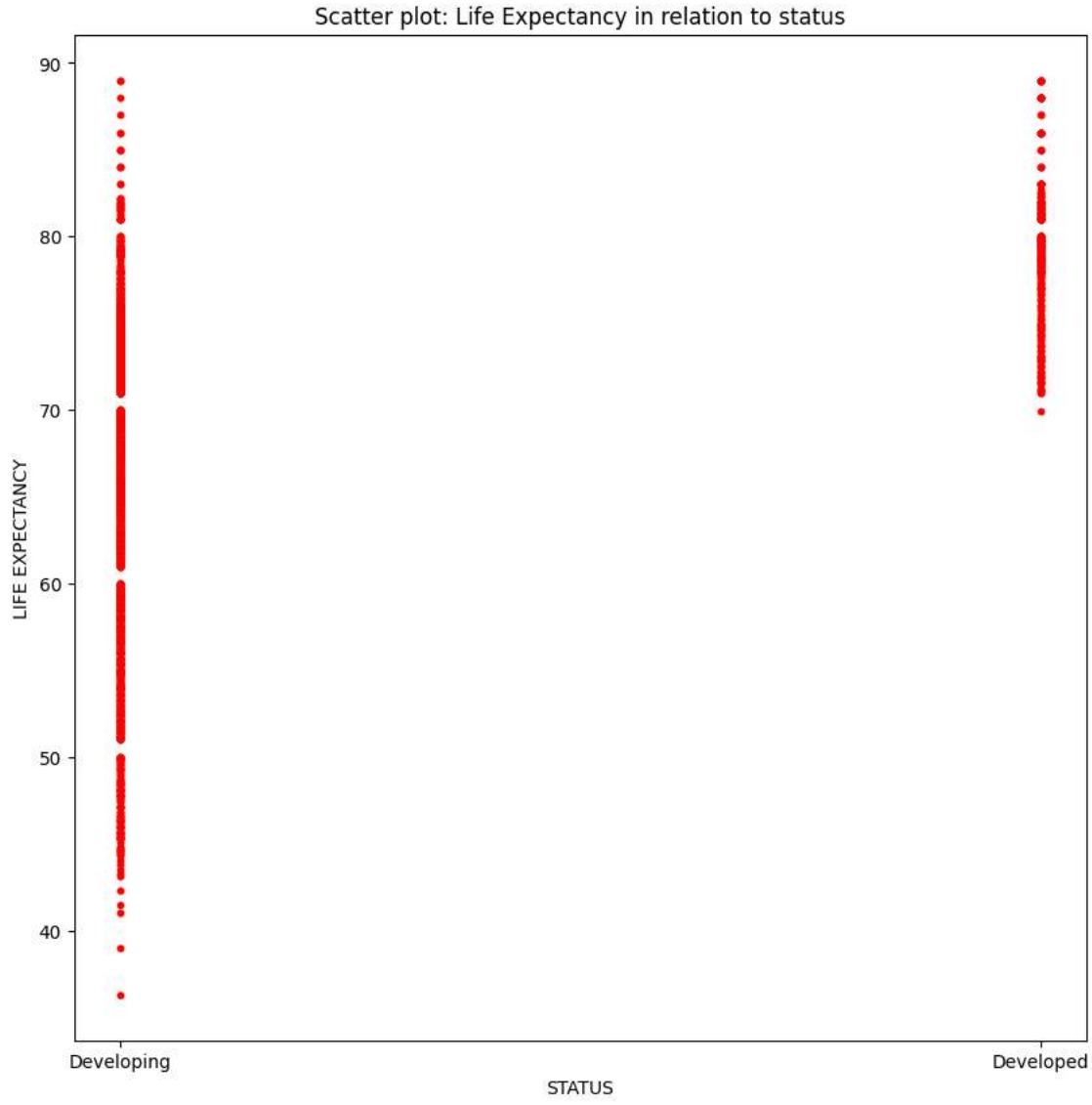
In [84]:
```
data['Life_Expectancy'].plot(kind='box', title='BOX PLOT: Life Expectancies before removing outliers',figsize=(7,4))
plt.show()
```

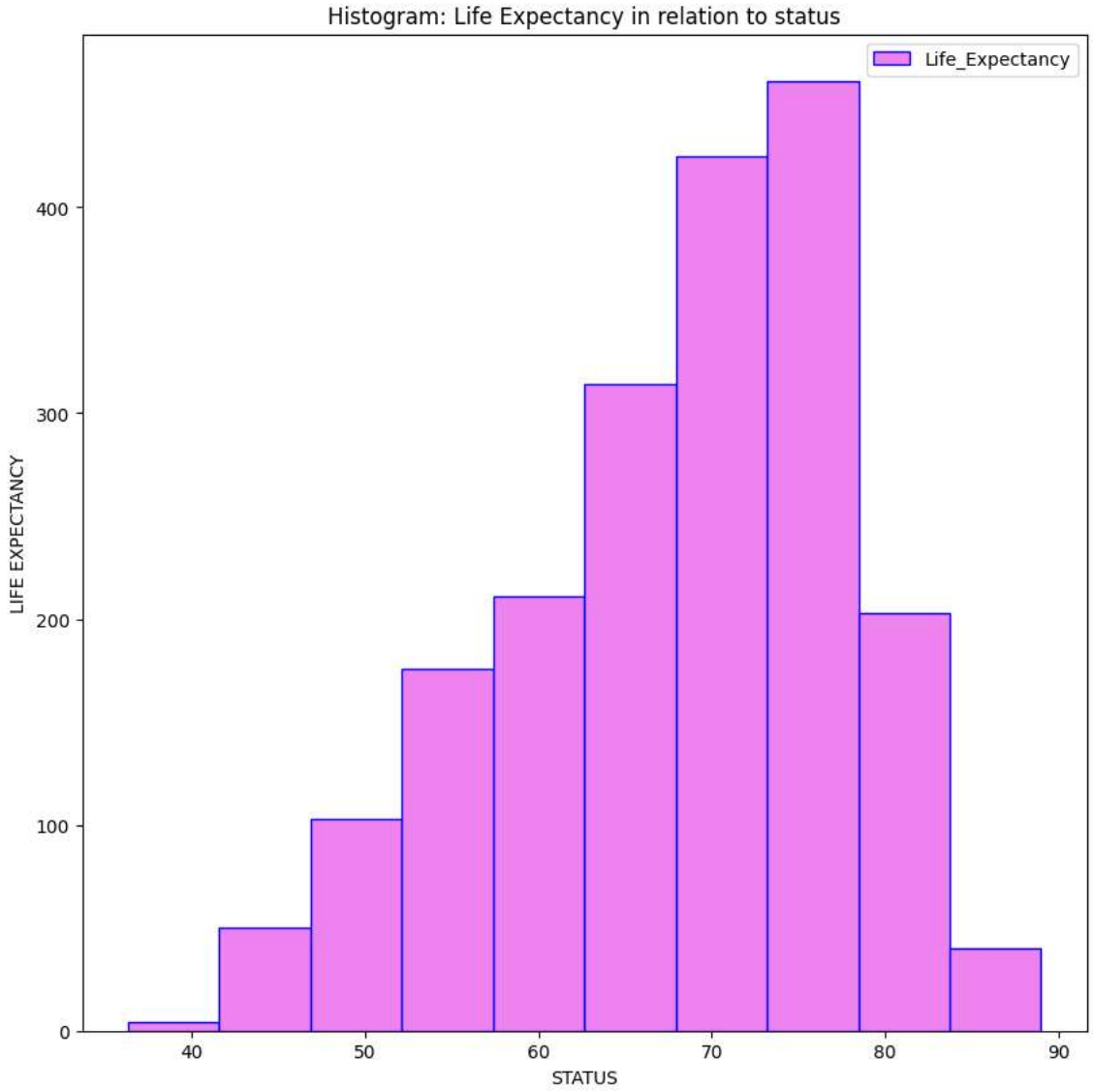


We have constructed a box plot of life expectancy of our dataframe and we can see the outliers in our plot.

Scatter plot of Life Expectancy with respect to status before removing outliers

In [85]:
```
data.plot.scatter(x='Status',  y='Life_Expectancy', c = 'red', s = 10,colormap='viridis',figsize=(10,10))
plt.title("Scatter plot: Life Expectancy in relation to status")
plt.xlabel("STATUS")
plt.ylabel("LIFE EXPECTANCY")
plt.show()

data.plot(x='Status',  y='Life_Expectancy',figsize=(10,10), kind= 'hist', color='violet', edgecolor='blue')
plt.title("Histogram: Life Expectancy in relation to status")
plt.xlabel("STATUS")
plt.ylabel("LIFE EXPECTANCY")
plt.show()
```

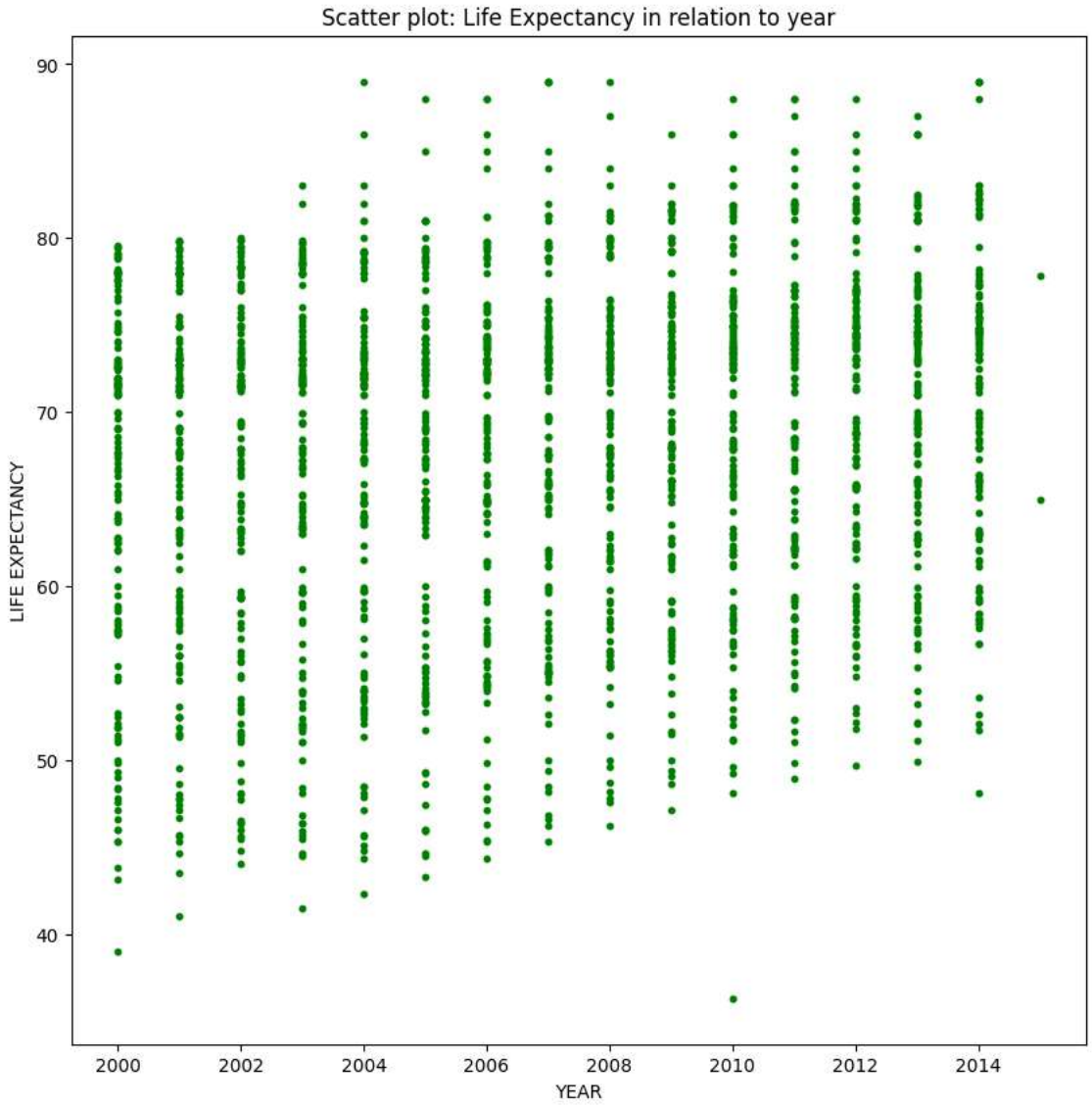




Scatter Plot and Histogram:

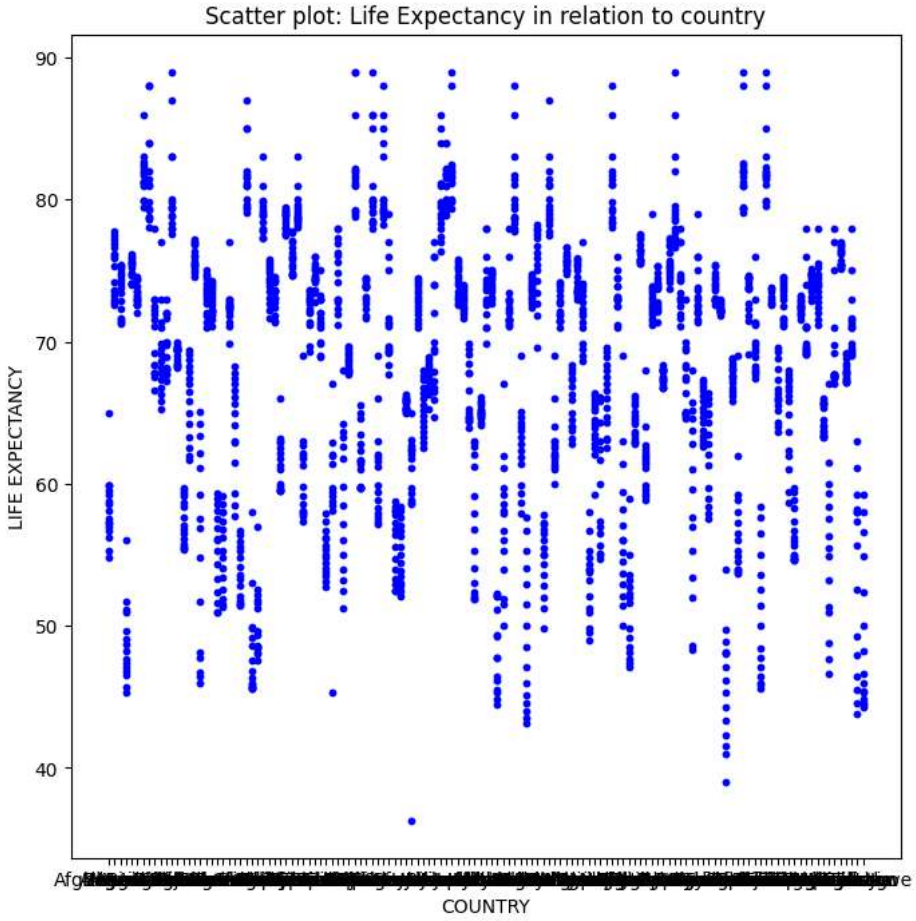
Scatter Plot or histograms are not only a way portraying pretty plots and graphs but also a good way of analyzing the data to observe the relationships. We can see a good relationship between status and life expectancy. It is so because developing countries have low life expectancy compared to developed countries as we also saw above in our box plot.

In [86]:
```
data.plot.scatter(x='Year',  y='Life_Expectancy', c = 'green', s = 10,colormap='viridis',figsize=(10,10))
plt.title("Scatter plot: Life Expectancy in relation to year")
plt.xlabel("YEAR")
plt.ylabel("LIFE EXPECTANCY")
plt.show()
```

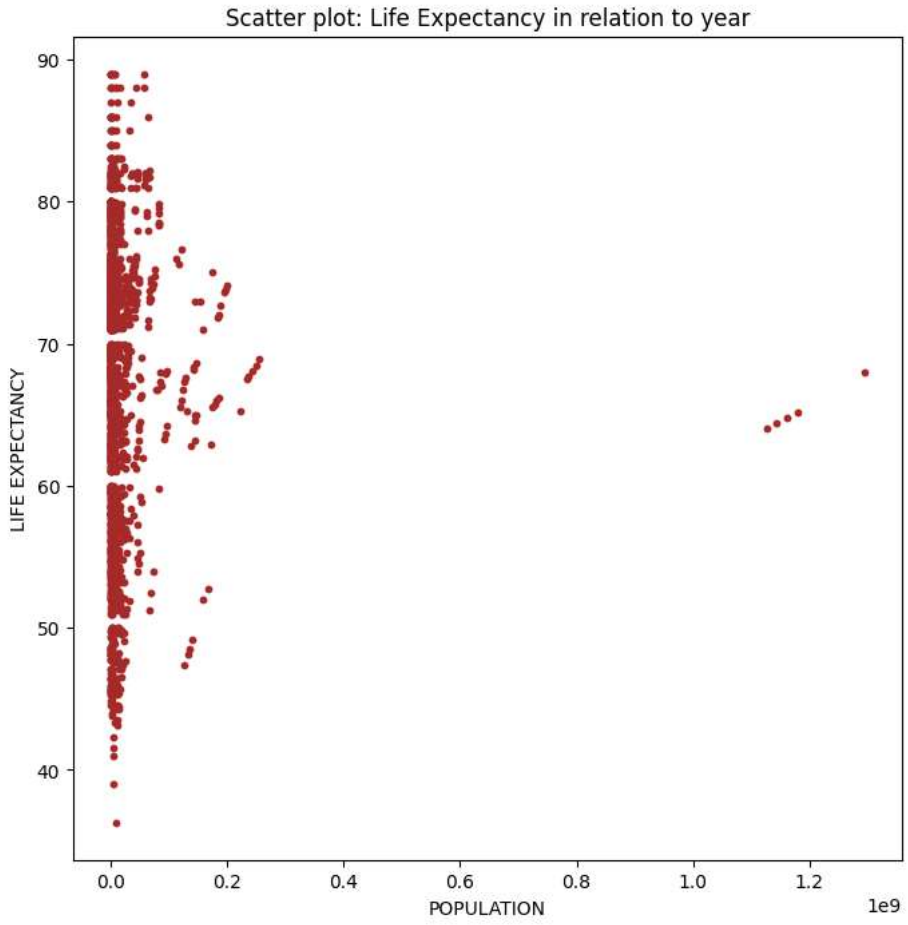


Scatter Plot:
The Scatter Plot was drawn to analyze the relationship between year and life expectancy. I believe they do show us a form of a relation they have but I believe we might need more columns from our dataframe to understand the relationships because inituitively we might not be good or accurate enough to make assumptions.

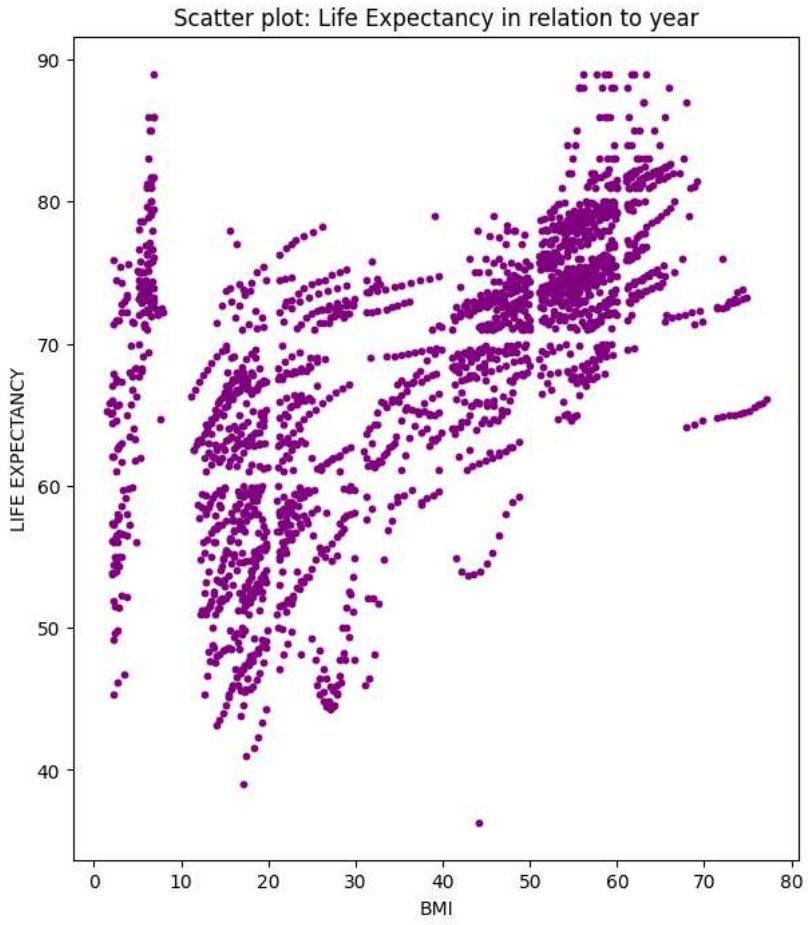
In [87]:
```
data.plot.scatter(x='Country',  y='Life_Expectancy', c = 'blue', s = 10,colormap='viridis',figsize=(8,8))
plt.title("Scatter plot: Life Expectancy in relation to country")
plt.xlabel("COUNTRY")
plt.ylabel("LIFE EXPECTANCY")
plt.show()
```



In [88]:
```
data.plot.scatter(x='Population',  y='Life_Expectancy', c = 'brown', s = 10,colormap='viridis',figsize=(8,8))
plt.title("Scatter plot: Life Expectancy in relation to year")
plt.xlabel("POPULATION")
plt.ylabel("LIFE EXPECTANCY")
plt.show()
```



In [89]:
```
data.plot.scatter(x='BMI',  y='Life_Expectancy', c = 'purple', s = 10,colormap='viridis',figsize=(7,8))
plt.title("Scatter plot: Life Expectancy in relation to year")
plt.xlabel("BMI")
plt.ylabel("LIFE EXPECTANCY")
plt.show()
```



Above Plots:
I do not think we find a relationship of BMI, Population with life expectancy. We do see some sort of relationship between life expectancy and country in the sense which country has high and which country has low but it is difficult to say look at the plot above as we have a lot of countries.

Auxiliary function that helps to remove outliers

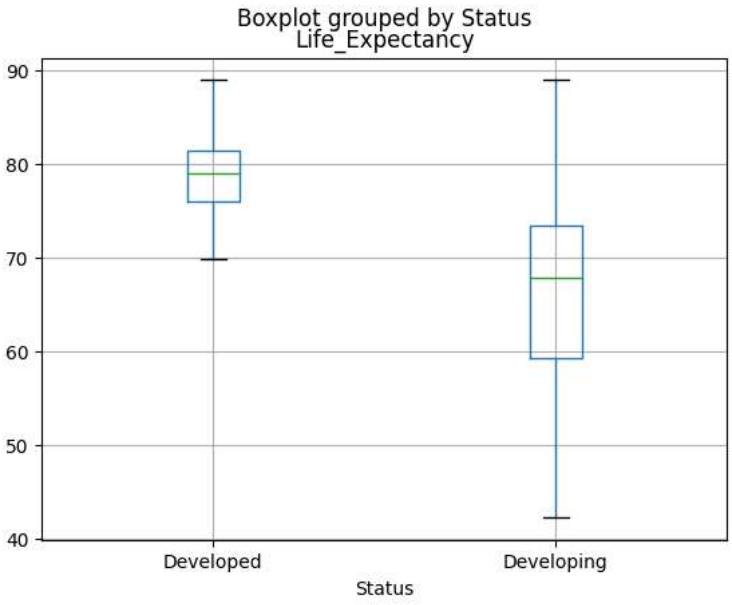
In [90]:
```
def remove_outlier(df):
    IQR = df['Life_Expectancy'].quantile(0.75) - df['Life_Expectancy'].quantile(0.25)

    lower_range = df['Life_Expectancy'].quantile(0.25) - (1.5 * IQR)
    upper_range = df['Life_Expectancy'].quantile(0.75) + (1.5 * IQR)

    df.loc[df['Life_Expectancy'] <= lower_range, 'Life_Expectancy'] = lower_range
    df.loc[df['Life_Expectancy'] >= upper_range, 'Life_Expectancy'] = upper_range
remove_outlier(data)
```

There are several ways to remove outliers. Here we are using Inter-Quartile-Range also known as IQR to remove outliers from our dataset for LIFE EXPECTANCY. Since the IQR is simply the range of the middle 50% of data values, it's not affected by extreme outliers.

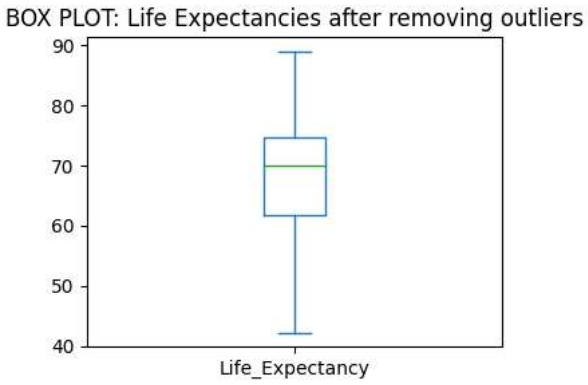
Box plot of Life Expectancy with respect to status after removing outliers

In [91]:
```
data.boxplot(by='Status',column=['Life_Expectancy'])
```

Out[91]:
```
<AxesSubplot: title={'center': 'Life_Expectancy'}, xlabel='Status'>
```

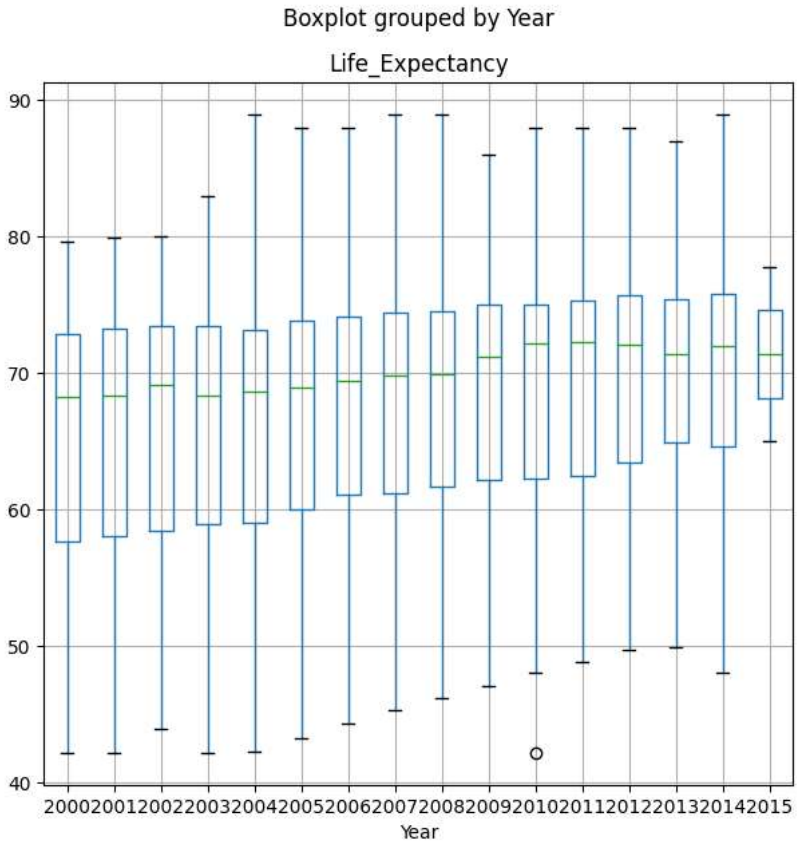


In [92]:
```
data['Life_Expectancy'].plot(kind='box', title='BOX PLOT: Life Expectancies after removing outliers',figsize=(4,3))
plt.show()
```



In [93]:
```
data.boxplot(by='Year',column=['Life_Expectancy'], figsize=(7,7))
```

Out[93]:
```
<AxesSubplot: title={'center': 'Life_Expectancy'}, xlabel='Year'>
```



After removing the outliers:
We did remove outliers above but looking at the box plot above between life expectancy and year, they do have share a relation and we do see outliers in the year 2010. It is so because we removed outliers or sort of range using IQR(Inter Quartile Range) but we did not for the ranges for these years. And it is not necessary or could say possible to remove all the outliers for all the relations we are trying to form or understand because we might intentionally or unintentionally loose data or information which we do not want.

In [94]:
```
data.isnull().sum()*100/len(data)
```

Out[94]:
```
Country                          0.0
Year                             0.0
Status                           0.0
Life_Expectancy                  0.0
Adult_Mortality                  0.0
Infant_deaths                    0.0
Alcohol                          0.0
Percentage_expenditure           0.0
Hepatitis_B                      0.0
Measles                          0.0
BMI                              0.0
under-five-deaths                0.0
Polio                            0.0
Total_expenditure                0.0
Diphtheria                       0.0
HIV/AIDS                         0.0
GDP                              0.0
Population                       0.0
thinness_1-19_years              0.0
thinness_5-9_years               0.0
Income_composition_of_resources  0.0
Schooling                        0.0
dtype: float64
```

Therefore, We can see that when we check the number of missing values in our dataset or dataframe for each and every column we get zeroes (0) for all after doing the data cleaning where we dropped the missing values or null or nan values and removed our outliers and made our data clean and pretty to work and do our analysis further using machine learning and data science techniques.

DATA EXPLORATION

We are trying to test and explore the Status which is Developing anf Developed countries

In [95]:
```
developing = []
developed = []
```

In [96]:
```
for index,row in data_copy.iterrows():
    if row[2] == "Developing":
        developing.append(row)
    else:
        pass
developing = pd.DataFrame(developing)
```

Here we created a new dataframe where we added all the information from our original dataset or dataframe which only for status Developing and stored all the data and information in our dataframe named developing.

In [97]:
```
for index,row in data_copy.iterrows():
    if row[2] == "Developed":
        developed.append(row)
    else:
        pass
developed = pd.DataFrame(developed)
```

Here we created a new dataframe where we added all the information from our original dataset or dataframe which only for status Developed and stored all the data and information in our dataframe named developed.

In [98]:
```
developing.describe()
```

Out[98]:

	Year	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	BMI	under d
count	2426.000000	2416.000000	2416.000000	2426.000000	2260.000000	2426.000000	2046.000000	2426.000000	2392.000000	2426.0
mean	2007.522671	67.111465	182.833195	36.384171	3.484119	323.470285	79.763930	2824.926216	35.435326	50.5
std	4.614690	9.006092	127.974557	128.942509	3.347537	846.655356	25.564884	12528.811419	19.425091	175.3
min	2000.000000	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	0.000000	1.000000	0.0
25%	2004.000000	61.100000	92.000000	1.000000	0.517500	3.616102	75.000000	0.000000	18.300000	1.0
50%	2008.000000	69.000000	163.000000	6.000000	2.560000	48.431829	91.000000	18.000000	35.200000	7.0
75%	2012.000000	74.000000	253.000000	28.000000	5.750000	257.702204	97.000000	514.500000	53.200000	39.0
max	2015.000000	89.000000	723.000000	1800.000000	17.870000	9748.636237	99.000000	212183.000000	87.300000	2500.0

We are displaying the statistical information in terms of number of rows or tuples, mean or average, standard deviation, 25%, median, 75% and maximum values for all the columns for developing countries.

In [99]:
```
developed.describe()
```

Out[99]:

	Year	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	BMI	under-five-deaths
count	512.000000	512.000000	512.000000	512.000000	484.000000	512.000000	339.000000	512.000000	512.000000	512.000000
mean	2007.500000	79.197852	79.685547	1.494141	9.826736	2703.600380	88.041298	499.005859	51.803906	1.810547
std	4.614281	3.930942	47.877583	4.585774	2.765858	3824.200588	20.489240	2529.084588	17.196829	5.384006
min	2000.000000	69.900000	1.000000	0.000000	0.010000	0.000000	2.000000	0.000000	3.200000	0.000000
25%	2003.750000	76.800000	58.000000	0.000000	8.617500	92.904052	89.000000	0.000000	53.775000	0.000000
50%	2007.500000	79.250000	73.000000	0.000000	10.320000	846.615644	95.000000	12.000000	57.450000	0.000000
75%	2011.250000	81.700000	96.000000	1.000000	11.697500	4102.863046	97.000000	96.500000	61.300000	2.000000
max	2015.000000	89.000000	229.000000	28.000000	15.190000	19479.911610	99.000000	33812.000000	69.600000	33.000000

We are displaying the statistical information in terms of number of rows or tuples, mean or average, standard deviation, 25%, median, 75% and maximum values for all the columns for developed countries.

In [100]:
```
developed.head()
```

Out[100]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_exp
112	Australia	2015	Developed	82.8	59.0	1	NaN	0.00000	93.0	74	...	93.0	
113	Australia	2014	Developed	82.7	6.0	1	9.71	10769.36305	91.0	340	...	92.0	
114	Australia	2013	Developed	82.5	61.0	1	9.87	11734.85381	91.0	158	...	91.0	
115	Australia	2012	Developed	82.3	61.0	1	10.03	11714.99858	91.0	199	...	92.0	
116	Australia	2011	Developed	82.0	63.0	1	10.30	10986.26527	92.0	190	...	92.0	

5 rows × 22 columns

In [101]:
```
developing.head()
```

Out[101]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_ex
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	

5 rows × 22 columns

In [102]:
```
developing.isnull().sum()*100/len(developing)
```

Out[102]:
```
Country                          0.000000
Year                             0.000000
Status                           0.000000
Life_Expectancy                  0.412201
Adult_Mortality                  0.412201
Infant_deaths                    0.000000
Alcohol                          6.842539
Percentage_expenditure           0.000000
Hepatitis_B                     15.663644
Measles                          0.000000
BMI                              1.401484
under-five-deaths                0.000000
Polio                            0.783182
Total_expenditure                7.996702
Diphtheria                       0.783182
HIV/AIDS                         0.000000
GDP                             15.828524
Population                      22.918384
thinness_1-19_years              1.401484
thinness_5-9_years               1.401484
Income_composition_of_resources  4.905194
Schooling                        4.740313
dtype: float64
```

We are computing the percentage of null values or missing values in our columns for the dataframe that we created separately for developing countries to make sure our data is clean, pretty and good enough so that when we apply machine learning and data science algorithms we do not run into problems and efficiency issues.

In [103]:
```
developing.loc[(developing.Hepatitis_B.isna()), 'Hepatitis_B'] = developing.Hepatitis_B.mean()
developing.loc[(developing.GDP.isna()), 'GDP'] = developing.GDP.mean()
developing.loc[(developing.Population.isna()), 'Population'] = developing.Population.mean()

developing.loc[(developing.Alcohol.isna()), 'Alcohol'] = developing.Alcohol.mean()
developing.loc[(developing.BMI.isna()), 'BMI'] = developing.BMI.mean()
developing.loc[(developing['thinness_1-19_years'].isna()), 'thinness_1-19_years'] = developing['thinness_1-19_years'].mean()
developing.loc[(developing['thinness_5-9_years'].isna()), 'thinness_5-9_years'] = developing['thinness_5-9_years'].mean()
developing.loc[(developing.Income_composition_of_resources.isna()), 'Income_composition_of_resources'] = developing.Income_co
developing.loc[(developing.Schooling.isna()), 'Schooling'] = developing.Schooling.mean()
developing.loc[(developing.Total_expenditure.isna()), 'Total_expenditure'] = developing.Total_expenditure.mean()
```

In [104]:
```
developing.isnull().sum()*100/len(developing)
```

Out[104]:
```
Country                          0.000000
Year                             0.000000
Status                           0.000000
Life_Expectancy                  0.412201
Adult_Mortality                  0.412201
Infant_deaths                    0.000000
Alcohol                          0.000000
Percentage_expenditure           0.000000
Hepatitis_B                      0.000000
Measles                          0.000000
BMI                              0.000000
under-five-deaths                0.000000
Polio                            0.783182
Total_expenditure                0.000000
Diphtheria                       0.783182
HIV/AIDS                         0.000000
GDP                              0.000000
Population                       0.000000
thinness_1-19_years              0.000000
thinness_5-9_years               0.000000
Income_composition_of_resources  0.000000
Schooling                        0.000000
dtype: float64
```

As previously we dropped null values but here we are applying another technique of handling missing values or nan or null values is by replacing them by the average or mean of those specific column.

In [105]:
```
developed.isnull().sum()*100/len(developed)
```

Out[105]:
```
Country                          0.000000
Year                             0.000000
Status                           0.000000
Life_Expectancy                  0.000000
Adult_Mortality                  0.000000
Infant_deaths                    0.000000
Alcohol                          5.468750
Percentage_expenditure           0.000000
Hepatitis_B                     33.789062
Measles                          0.000000
BMI                              0.000000
under-five-deaths                0.000000
Polio                            0.000000
Total_expenditure                6.250000
Diphtheria                       0.000000
HIV/AIDS                         0.000000
GDP                             12.500000
Population                      18.750000
thinness_1-19_years              0.000000
thinness_5-9_years               0.000000
Income_composition_of_resources  9.375000
Schooling                        9.375000
dtype: float64
```

We are computing the percentage of null values or missing values in our columns for the dataframe that we created separately for developed countries to make sure our data is clean, pretty and good enough so that when we apply machine learning and data science algorithms we do not run into problems and efficiency issues.

In [106]:
```
developed.loc[(developed.Hepatitis_B.isna()), 'Hepatitis_B'] = developed.Hepatitis_B.mean()
developed.loc[(developed.GDP.isna()), 'GDP'] = developed.GDP.mean()
developed.loc[(developed.Population.isna()), 'Population'] = developed.Population.mean()

developed.loc[(developed.Alcohol.isna()), 'Alcohol'] = developed.Alcohol.mean()
developed.loc[(developed.BMI.isna()), 'BMI'] = developed.BMI.mean()
developed.loc[(developed.Income_composition_of_resources.isna()), 'Income_composition_of_resources'] = developed.Income_compo
developed.loc[(developed.Schooling.isna()), 'Schooling'] = developed.Schooling.mean()
developed.loc[(developed.Total_expenditure.isna()), 'Total_expenditure'] = developed.Total_expenditure.mean()
```

In [107]:
```
developed.isnull().sum()*100/len(developed)
```

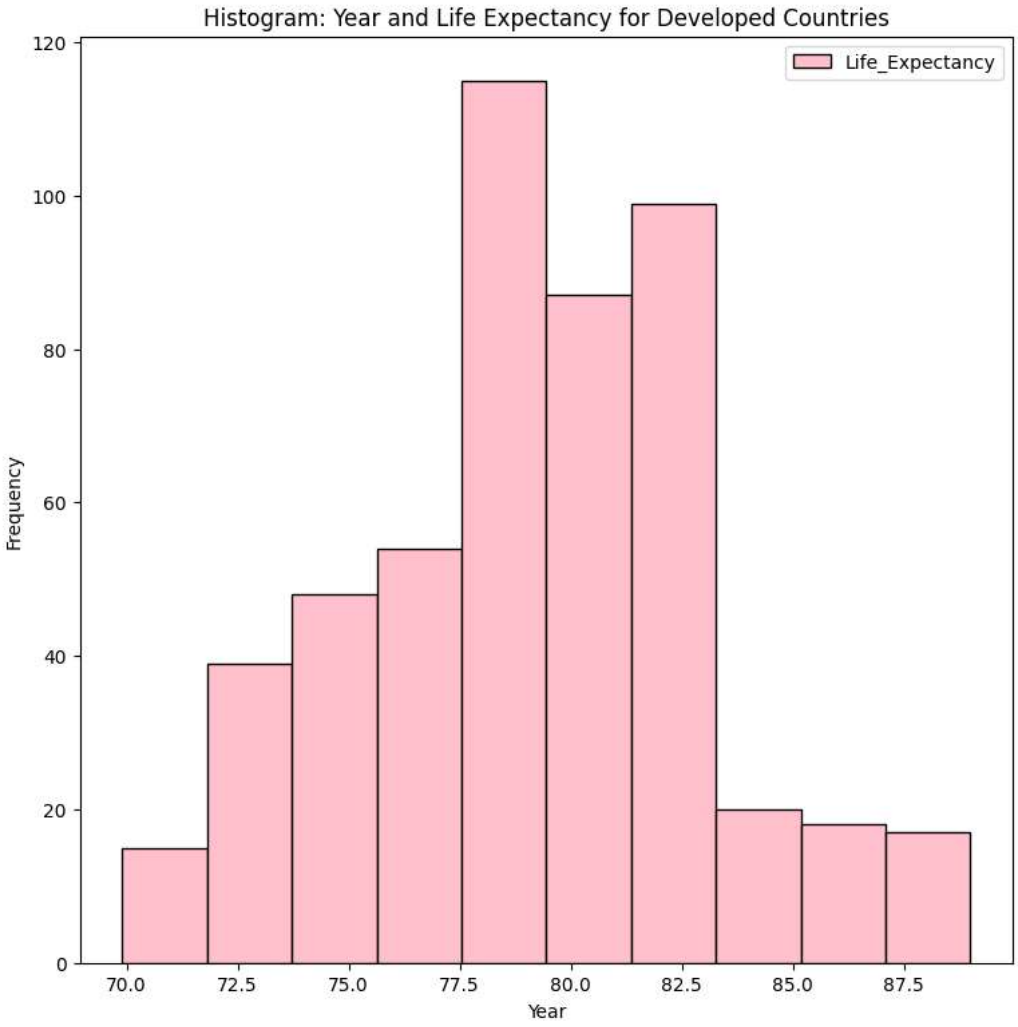
Out[107]:
```
Country                          0.0
Year                             0.0
Status                           0.0
Life_Expectancy                  0.0
Adult_Mortality                  0.0
Infant_deaths                    0.0
Alcohol                          0.0
Percentage_expenditure           0.0
Hepatitis_B                      0.0
Measles                          0.0
BMI                              0.0
under-five-deaths                0.0
Polio                            0.0
Total_expenditure                0.0
Diphtheria                       0.0
HIV/AIDS                         0.0
GDP                              0.0
Population                       0.0
thinness_1-19_years              0.0
thinness_5-9_years               0.0
Income_composition_of_resources  0.0
Schooling                        0.0
dtype: float64
```

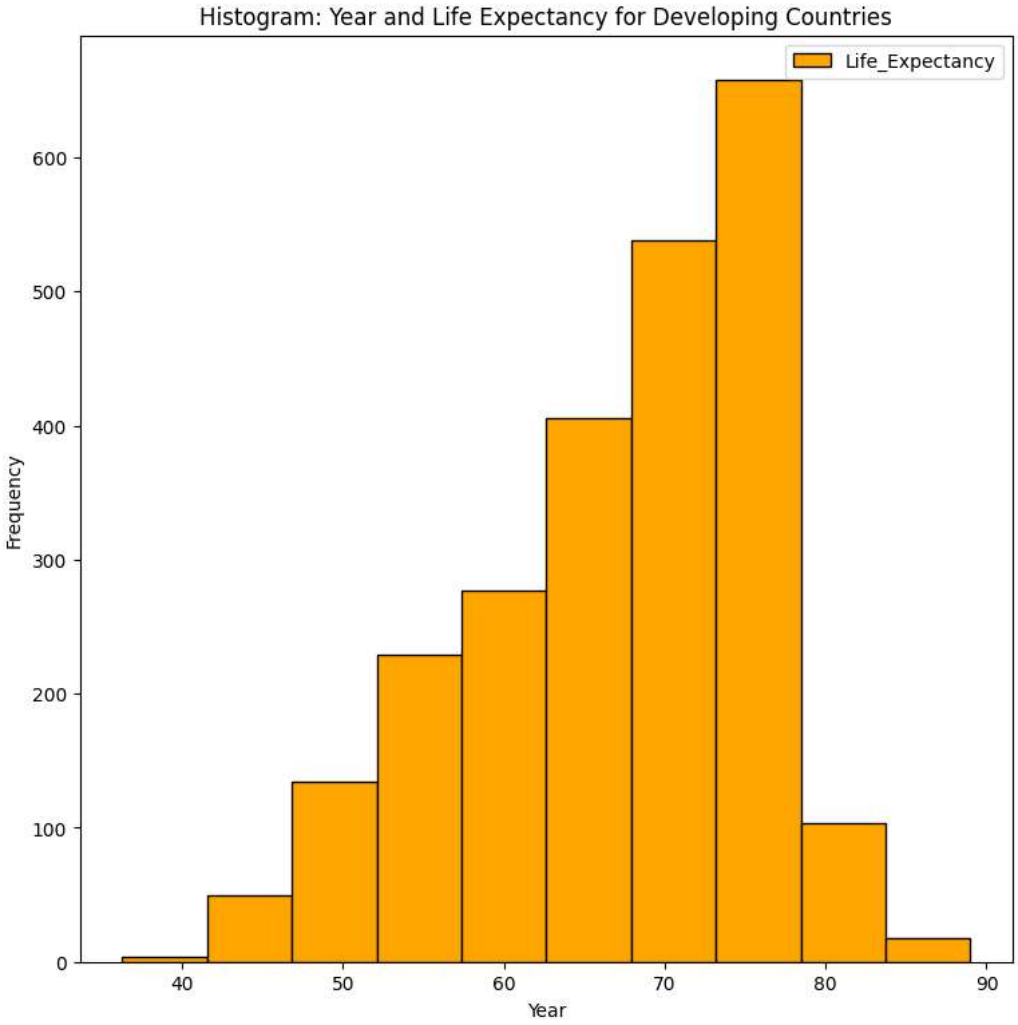
As previously we dropped null values but here we are applying another technique of handling missing values or nan or null values is by replacing them by the average or mean of those specific column. We could also use other methods such as replace with median, standard deviation or intepolate().

DATA VISUALIZATION

In [108]:
```
developed.plot(x='Year', y='Life_Expectancy',color = 'pink', kind = 'hist', figsize=(9,9), edgecolor = "black")
plt.xlabel("Year")
plt.ylabel("Frequency")
plt.title("Histogram: Year and Life Expectancy for Developed Countries")
developing.plot(x='Year', y='Life_Expectancy', color = 'orange', kind = 'hist', figsize=(9,9),edgecolor = "black")
plt.xlabel("Year")
plt.ylabel("Frequency")
plt.title("Histogram: Year and Life Expectancy for Developing Countries")
```

Out[108]:
```
Text(0.5, 1.0, 'Histogram: Year and Life Expectancy for Developing Countries')
```

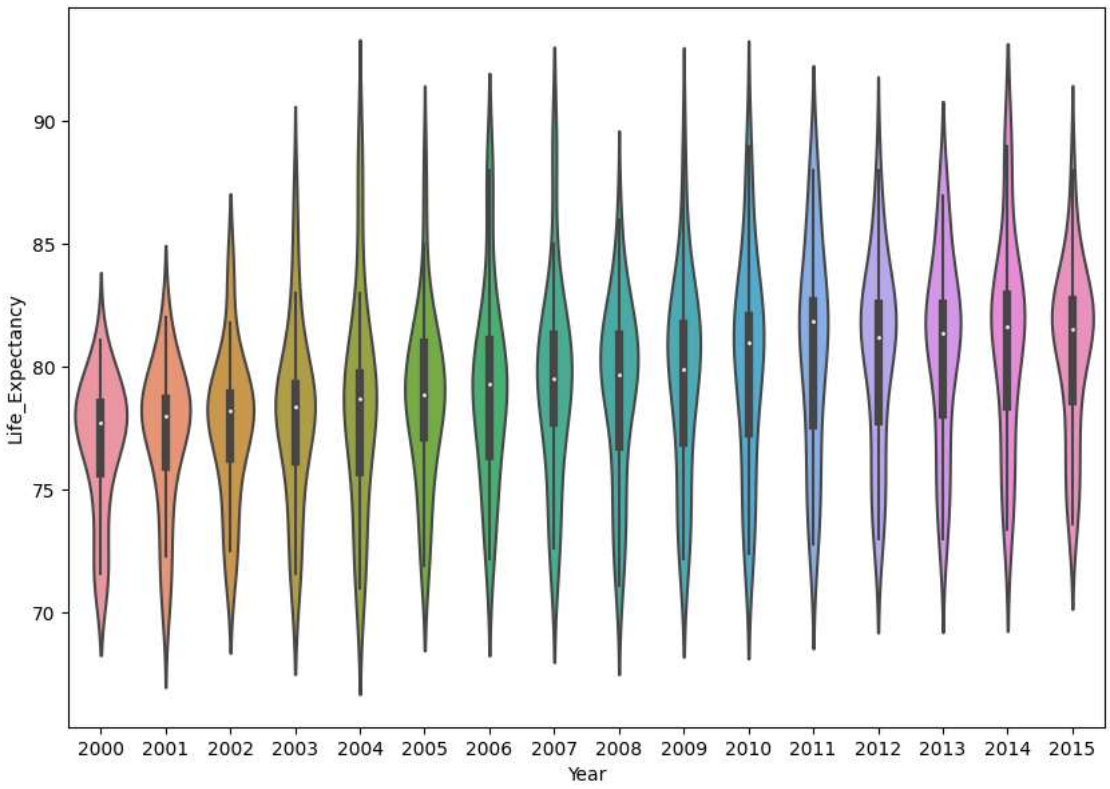




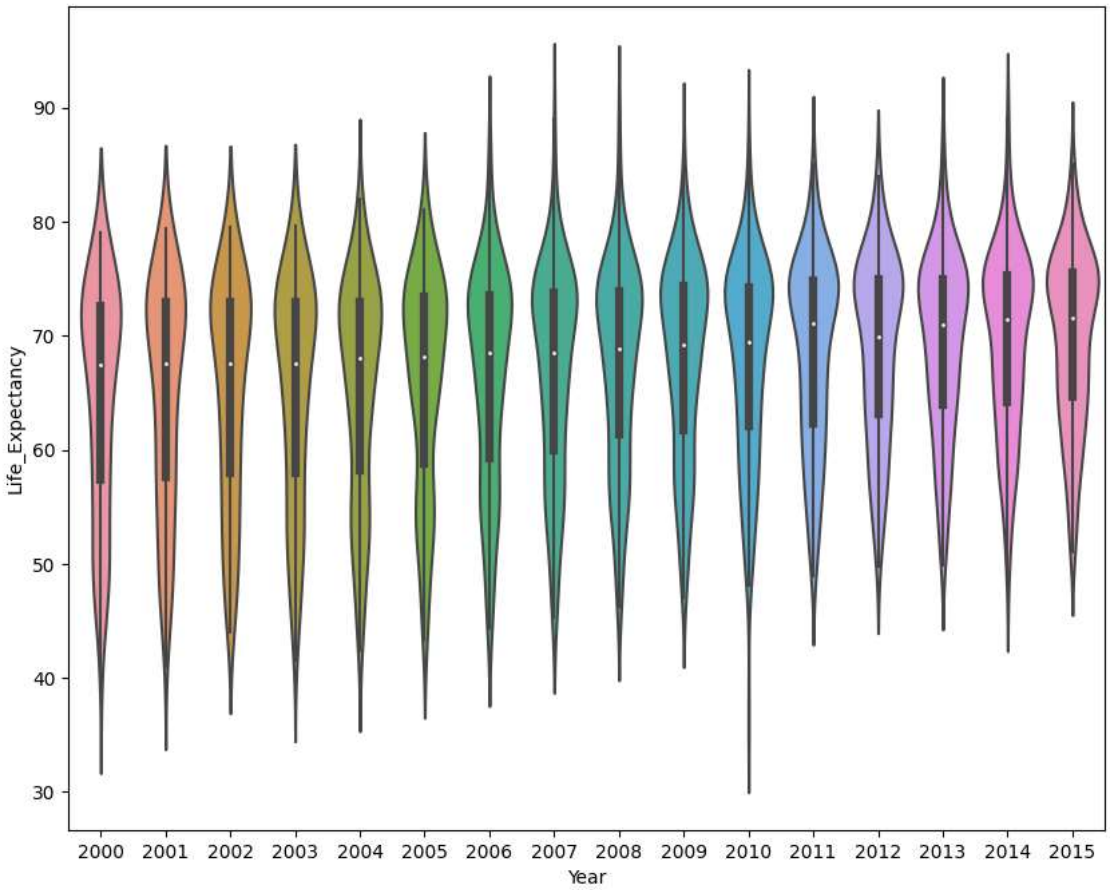
Above Plots:
Here we make histograms for developing countries as well as developed countries from our separate data sets for developing and developed.
We can see and with analyzing can say that those histograms look different very much and many different statistical as well as analysis perspective.
For Developed Countries; we can see fluctuation in the relationship between year and life expectancy though the life expectancy declined later year.
For Developing Countries; we can see an increasing trend but then a sudden huge decline which definitely is influenced due to many factors of consideration comapred to developed countries.

The motive of making two separate dataframes for developing and developed status; was to see if our intuition and analysis that we made in our orginial dataset matches when we individually work with those datasets and analyze our predictions or intuitions.
I believe yes, our motive succeeds because we can say that developed countries has a better statistical analysis compared to the developing countries.

In [109]:
```
plt.figure(figsize=(10,7))
sns.violinplot(x="Year",y="Life_Expectancy",data=developed)
plt.show()
```

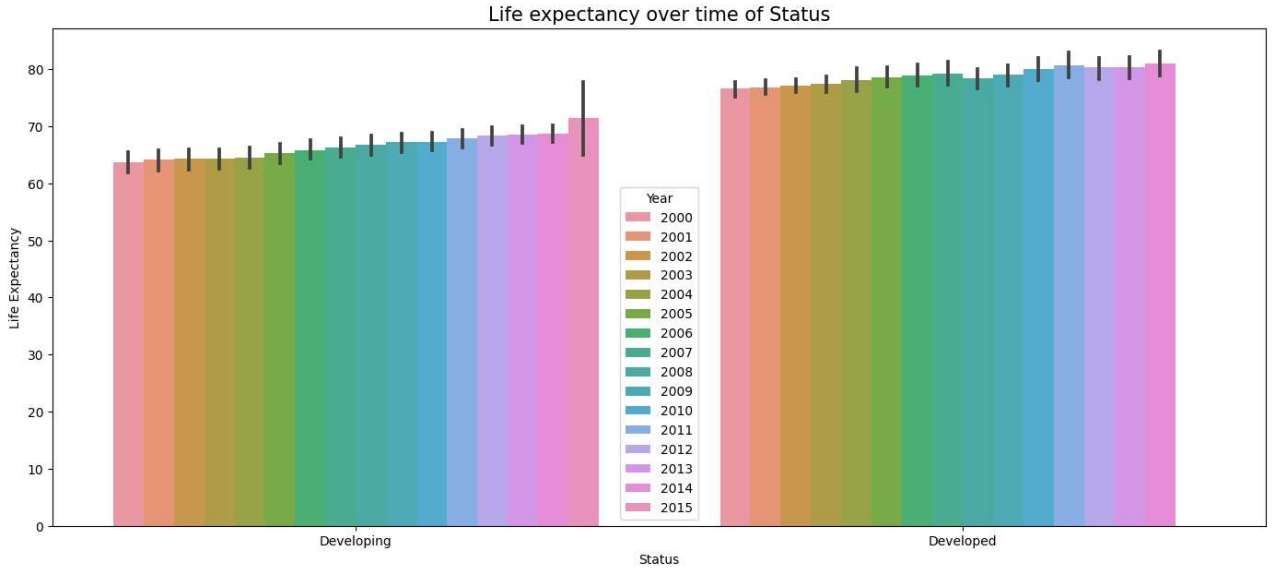


In [110]:
```
plt.figure(figsize=(10,8))
sns.violinplot(x="Year",y="Life_Expectancy",data=developing)
plt.show()
```



Above Plots: Above we have violin plot as a part of visualization for developing and developed countries after dealing with missing values by replacing them with the average which is mean. We can observe and visualize that the statistical values and the plots of developed countries do some good source in terms of good economic situation for instance of the economies or countries compared to the developing countries.

In [111]:
```
plt.figure(figsize=(17,7))
sns.barplot(x="Status",y="Life_Expectancy",hue="Year",data=data)
plt.title("Life expectancy over time of Status", size = 15)
plt.xlabel("Status")
plt.ylabel("Life Expectancy")
plt.show()
```



Above Plots:
We saw above that as in overall there was some sort of relation between life expectancy and year but here we are individually seeing and trying to observe relation between life expectancy and year for developing and developed countries. Here too we see a positive relationship between the life expectancy of a countries respective to the status though the life expectancy of the developed countries is higher and better compared to the life expectancy of the developing countries.

CORRELATION MATRIX

In [112]:
```
data.corr()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_17484\2627137660.py:1: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numer
ic_only to silence this warning.
  data.corr()
```

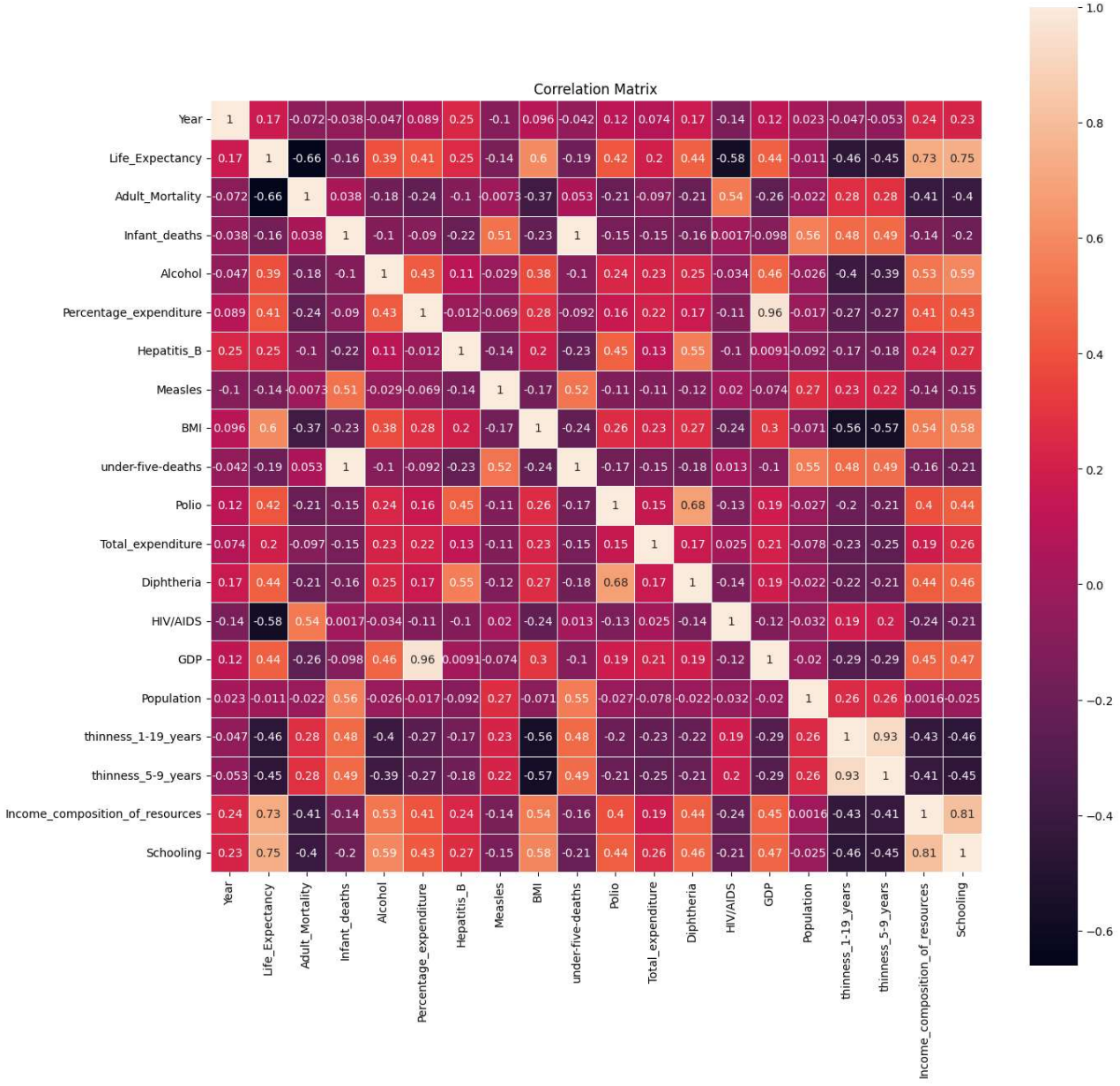
Out[112]:

	Year	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	
Year	1.000000	0.171593	-0.072108	-0.037601	-0.046859	0.089096	0.247259	-0.099554	0.(
Life_Expectancy	0.171593	1.000000	-0.659853	-0.161230	0.393336	0.414151	0.249875	-0.138435	0.(
Adult_Mortality	-0.072108	-0.659853	1.000000	0.038304	-0.181469	-0.242438	-0.103382	-0.007269	-0.:
Infant_deaths	-0.037601	-0.161230	0.038304	1.000000	-0.104406	-0.089772	-0.216949	0.509747	-0.:
Alcohol	-0.046859	0.393336	-0.181469	-0.104406	1.000000	0.430835	0.106383	-0.029252	0.:
Percentage_expenditure	0.089096	0.414151	-0.242438	-0.089772	0.430835	1.000000	-0.011530	-0.069316	0.:
Hepatitis_B	0.247259	0.249875	-0.103382	-0.216949	0.106383	-0.011530	1.000000	-0.142059	0.'
Measles	-0.099554	-0.138435	-0.007269	0.509747	-0.029252	-0.069316	-0.142059	1.000000	-0.'
BMI	0.096059	0.600424	-0.372519	-0.227769	0.379327	0.277788	0.198627	-0.168172	1.(
under-five-deaths	-0.042479	-0.187739	0.052865	0.996729	-0.099713	-0.092480	-0.226512	0.519173	-0.:
Polio	0.117642	0.415326	-0.208006	-0.152153	0.239854	0.162606	0.451299	-0.113574	0.:
Total_expenditure	0.074139	0.200832	-0.096727	-0.147961	0.227108	0.217103	0.130435	-0.111638	0.:
Diphtheria	0.166006	0.443123	-0.210136	-0.156470	0.245454	0.168910	0.552732	-0.119828	0.:
HIV/AIDS	-0.142581	-0.577381	0.536273	0.001739	-0.033756	-0.109680	-0.104034	0.019600	-0.:
GDP	0.119355	0.444095	-0.256955	-0.097720	0.458112	0.963177	0.009077	-0.073858	0.:
Population	0.022775	-0.010911	-0.022403	0.562805	-0.025556	-0.016607	-0.092418	0.269840	-0.(
thinness_1-19_years	-0.047477	-0.459704	0.278842	0.481580	-0.398731	-0.268347	-0.166903	0.227961	-0.:
thinness_5-9_years	-0.053483	-0.451058	0.284581	0.487596	-0.387087	-0.268131	-0.181161	0.223325	-0.:
Income_composition_of_resources	0.242900	0.727357	-0.411010	-0.137175	0.532173	0.408174	0.239386	-0.137648	0.:
Schooling	0.225046	0.745245	-0.404160	-0.195815	0.590261	0.426801	0.268951	-0.146625	0.:

The above command which is data.corr() basically gives us correlations for all the numerical columns in our dataframe.

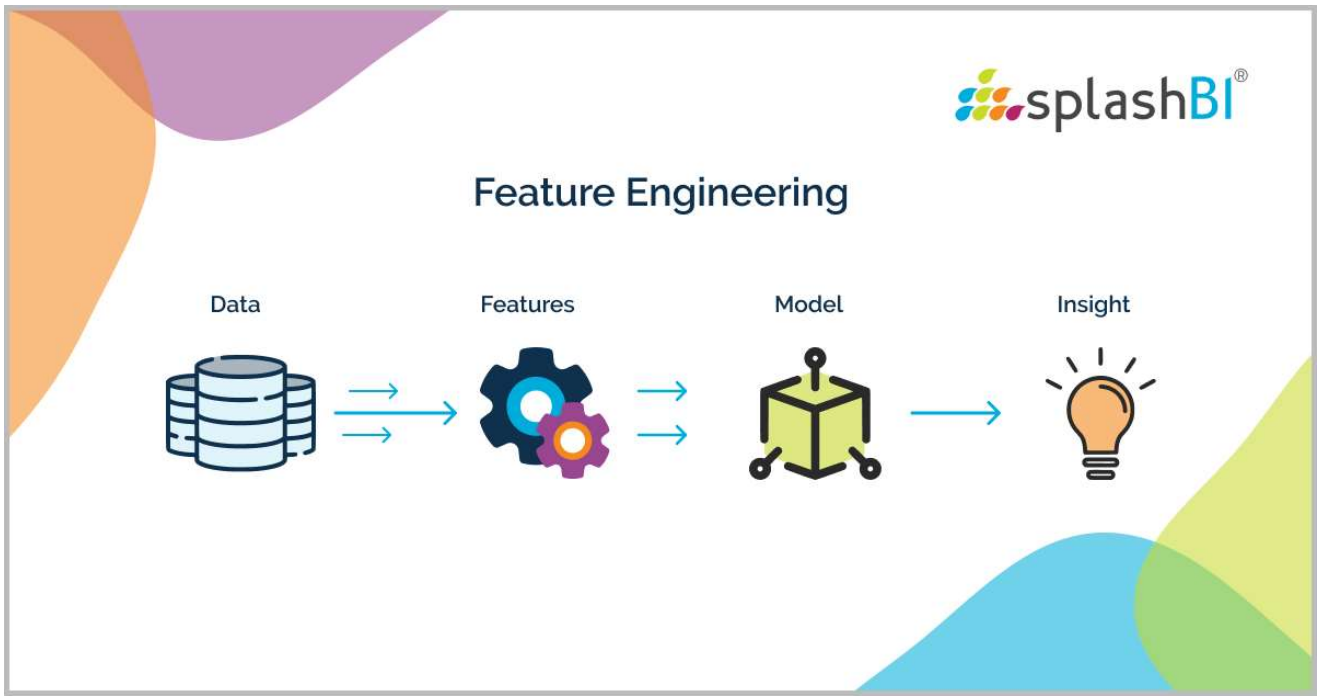
In [113]:
```
plt.figure(figsize=(15,15))
correlation = data.corr()
sns.heatmap(correlation, annot = True, square = True, linewidths=.5)#,cmap="Greens")
plt.title("Correlation Matrix")
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_17484\3253367397.py:2: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numer
ic_only to silence this warning.
  correlation = data.corr()
```



We know that a correlation coefficient can take the values from -1 through +1.

FEATURE ENGINEERING



Feature Engineering: There are several techniques of Feature Engineering we did Label Encoding, One-Hot Encoding and Binning.

Label Encoding: It basically converts categorical data into numerical data which is some numbers

Created a copy of our orginial dataframe and converted all categorical data into the numerical form

In [114]:
```
def Encoder(data_copy):
          columnsToEncode = list(data_copy.select_dtypes(include=['category','object']))
          le = LabelEncoder()
          for feature in columnsToEncode:
              try:
                  data_copy[feature] = le.fit_transform(data_copy[feature])
              except:
                  print('Error encoding '+feature)
          return data_copy
Encoder(data_copy)
```

Out[114]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_exper
0	0	2015	1	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	
1	0	2014	1	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	
2	0	2013	1	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	
3	0	2012	1	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	
4	0	2011	1	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2933	192	2004	1	44.3	723.0	27	4.36	0.000000	68.0	31	...	67.0	
2934	192	2003	1	44.5	715.0	26	4.06	0.000000	7.0	998	...	7.0	
2935	192	2002	1	44.8	73.0	25	4.43	0.000000	73.0	304	...	73.0	
2936	192	2001	1	45.3	686.0	25	1.72	0.000000	76.0	529	...	76.0	
2937	192	2000	1	46.0	665.0	24	1.68	0.000000	79.0	1483	...	78.0	

2938 rows × 22 columns

One-Hot Encoding: In this technique, it spreads the values in a feature to multiple flag features and assigns values 0 or 1 to them.

In [115]:
```
OneHotEncoder().fit_transform(data_copy).toarray()
```

Out[115]:
```
array([[1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [116]:
```
pd.get_dummies(data_copy, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False, dtype=No
```

Out[116]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Polio	Total_ex
0	0	2015	1	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	
1	0	2014	1	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	
2	0	2013	1	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	
3	0	2012	1	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	
4	0	2011	1	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2933	192	2004	1	44.3	723.0	27	4.36	0.000000	68.0	31	...	67.0	
2934	192	2003	1	44.5	715.0	26	4.06	0.000000	7.0	998	...	7.0	
2935	192	2002	1	44.8	73.0	25	4.43	0.000000	73.0	304	...	73.0	
2936	192	2001	1	45.3	686.0	25	1.72	0.000000	76.0	529	...	76.0	

Binning the data in 4 and computing the mean for life expectancy and alcohol consumption. So basically here we are considering life expectancy and alcohol as two of our columns or attributes.

In [117]:
```
data['discretize'] = pd.cut(data.Year, 4, ordered=True, labels=None)
data['label'] = pd.cut(data.Year, 4, ordered=True, labels=['first','second','third','fourth'])
```

Here we bin into four parts based on year and labeled as first, second, third, and fourth.

In [118]:
```
data.head()
```

Out[118]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	Diphtheria	HI\
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	65.0	
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	62.0	
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	64.0	
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	

5 rows × 24 columns

In [119]:
```
first=[]
second=[]
third=[]
fourth=[]
```

In [120]:
```
for index,row in data.iterrows():
    if("first" not in row[23]):
        pass
    else:
        first.append(row)
first = pd.DataFrame(first)
```

In [121]:
```
for index,row in data.iterrows():
    if("second" not in row[23]):
        pass
    else:
        second.append(row)
second = pd.DataFrame(second)
```

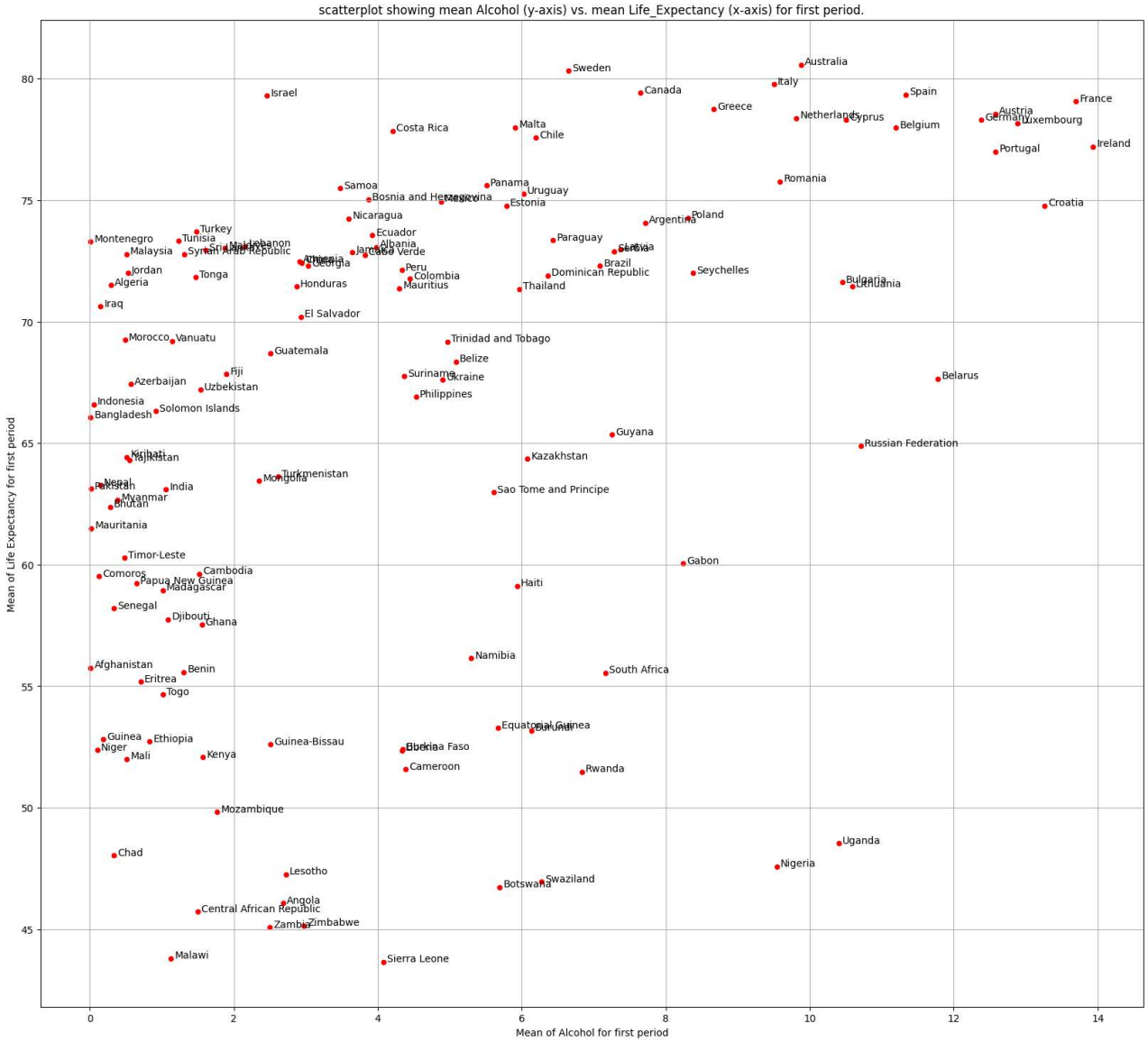
In [122]:
```
for index,row in data.iterrows():
    if("third" not in row[23]):
        pass
    else:
        third.append(row)
third = pd.DataFrame(third)
```

In [123]:
```
for index,row in data.iterrows():
    if("fourth" not in row[23]):
        pass
    else:
        fourth.append(row)
fourth = pd.DataFrame(fourth)
```

Above:
Above we did the binning process into four groups or four parts based on the years or time frame.
We then created four dataframes and stored all the information for the label separately and individually for the dataframes. Now we would be try to do find a relation between life expectancy and alcohol consumption with respect to average for different countries for four labels or four bins.

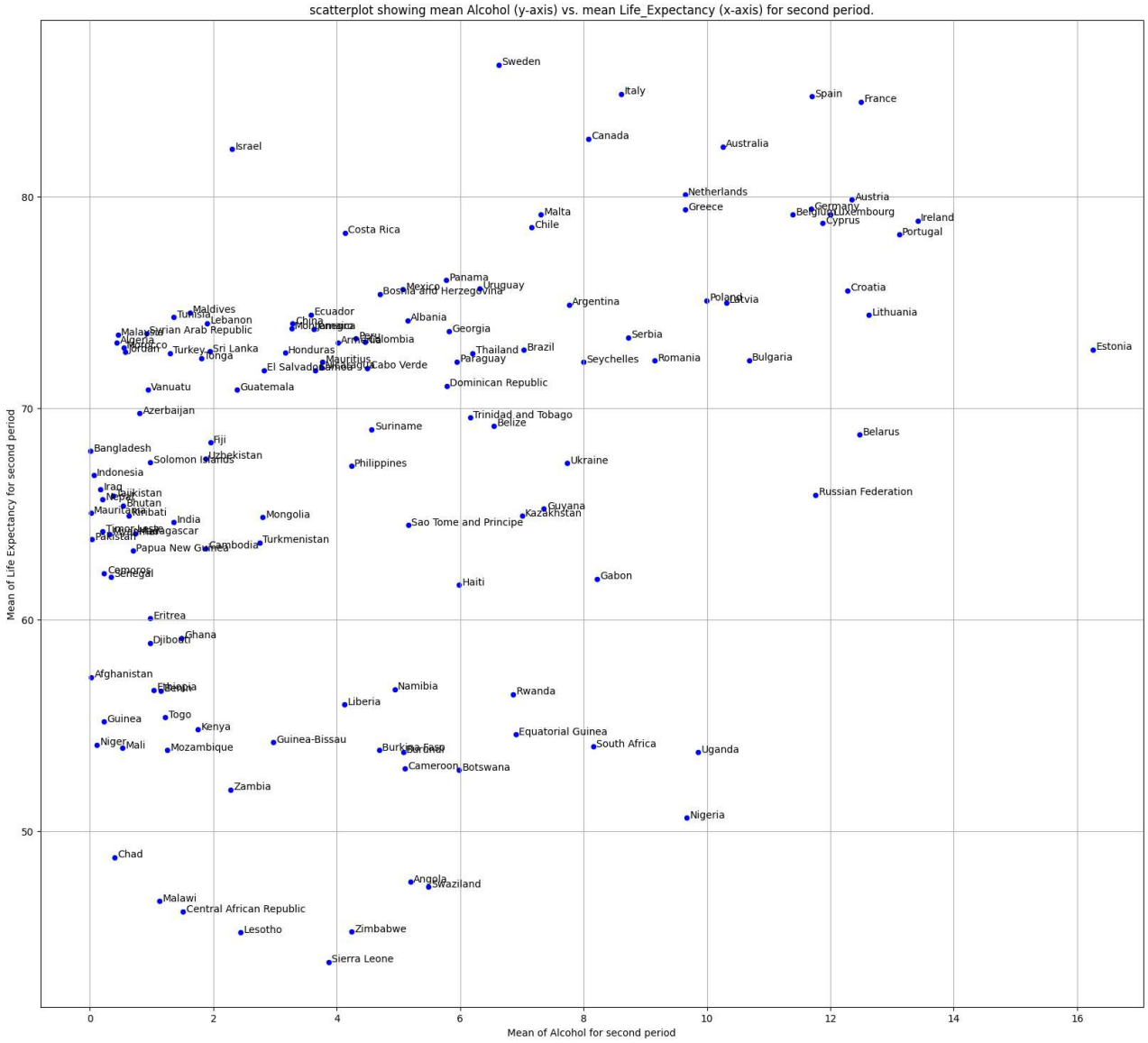
In [124]:
```
meansbyCountryfirst = first.groupby('Country')['Life_Expectancy', 'Alcohol'].mean()
df = pd.DataFrame(meansbyCountryfirst, columns=["Alcohol", "Life_Expectancy"])

ax = df.plot(x="Alcohol", y="Life_Expectancy", kind="scatter", figsize=(20,18), color="red")
plt.xlabel("Mean of Alcohol for first period")
plt.ylabel("Mean of Life Expectancy for first period")
plt.title("scatterplot showing mean Alcohol (y-axis) vs. mean Life_Expectancy (x-axis) for first period.")
df.reset_index(inplace=True)
for i, txt in enumerate(df.Country):
   ax.annotate(txt, (df.Alcohol.iat[i]+0.05, df.Life_Expectancy.iat[i]))
plt.grid(True)
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_17484\570783628.py:1: FutureWarning: Indexing with multiple keys (implicitly conver
ted to a tuple of keys) will be deprecated, use a list instead.
  meansbyCountryfirst = first.groupby('Country')['Life_Expectancy', 'Alcohol'].mean()
```



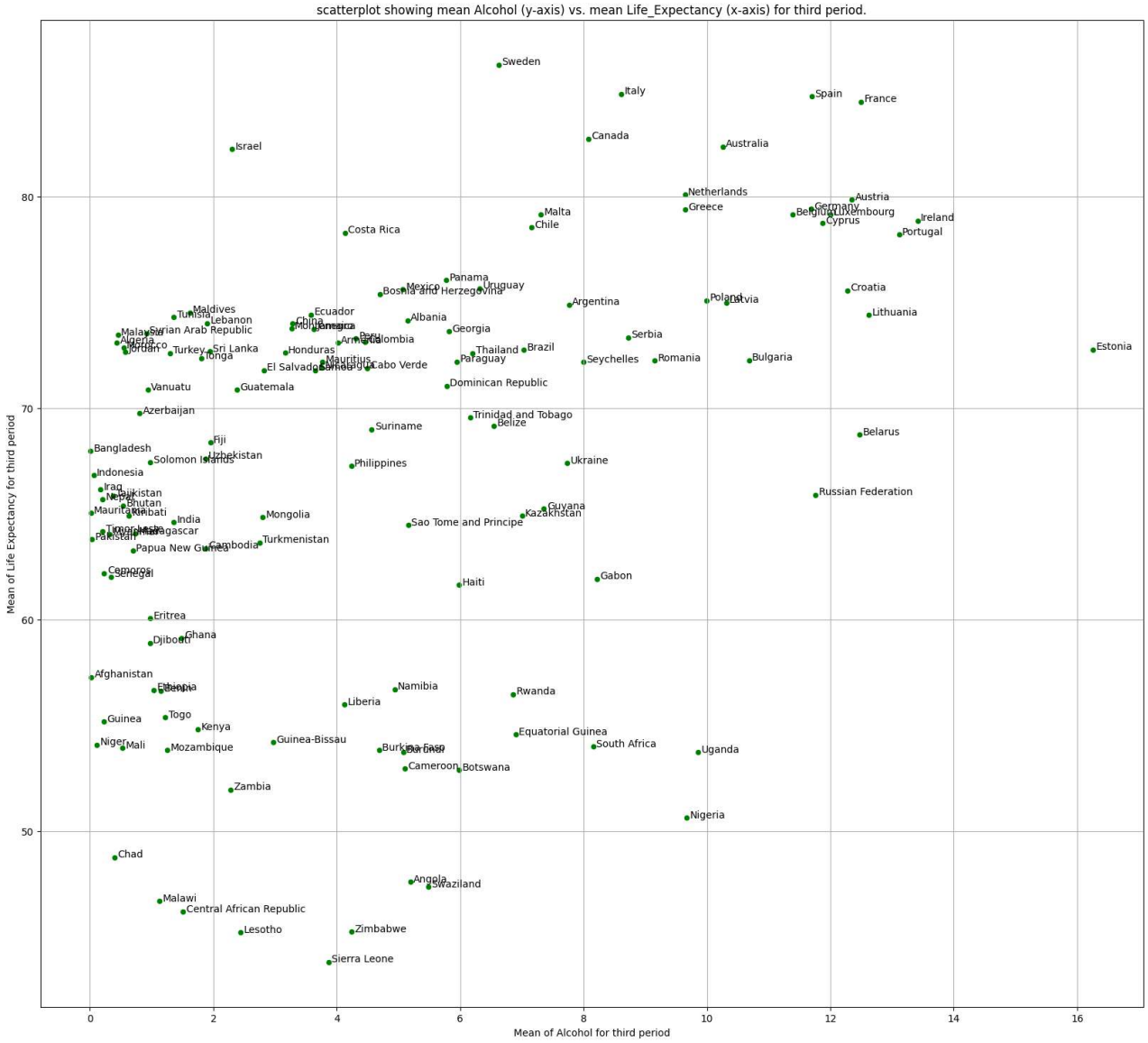
In [125]:
```
meansbyCountrysecond = second.groupby('Country')['Life_Expectancy', 'Alcohol'].mean()
df = pd.DataFrame(meansbyCountrysecond, columns=["Alcohol", "Life_Expectancy"])

ax = df.plot(x="Alcohol", y="Life_Expectancy", kind="scatter", figsize=(20,18), color="blue")
plt.xlabel("Mean of Alcohol for second period")
plt.ylabel("Mean of Life Expectancy for second period")
plt.title("scatterplot showing mean Alcohol (y-axis) vs. mean Life_Expectancy (x-axis) for second period.")
df.reset_index(inplace=True)
for i, txt in enumerate(df.Country):
   ax.annotate(txt, (df.Alcohol.iat[i]+0.05, df.Life_Expectancy.iat[i]))
plt.grid(True)
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_17484\1031961739.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  meansbyCountrysecond = second.groupby('Country')['Life_Expectancy', 'Alcohol'].mean()
```



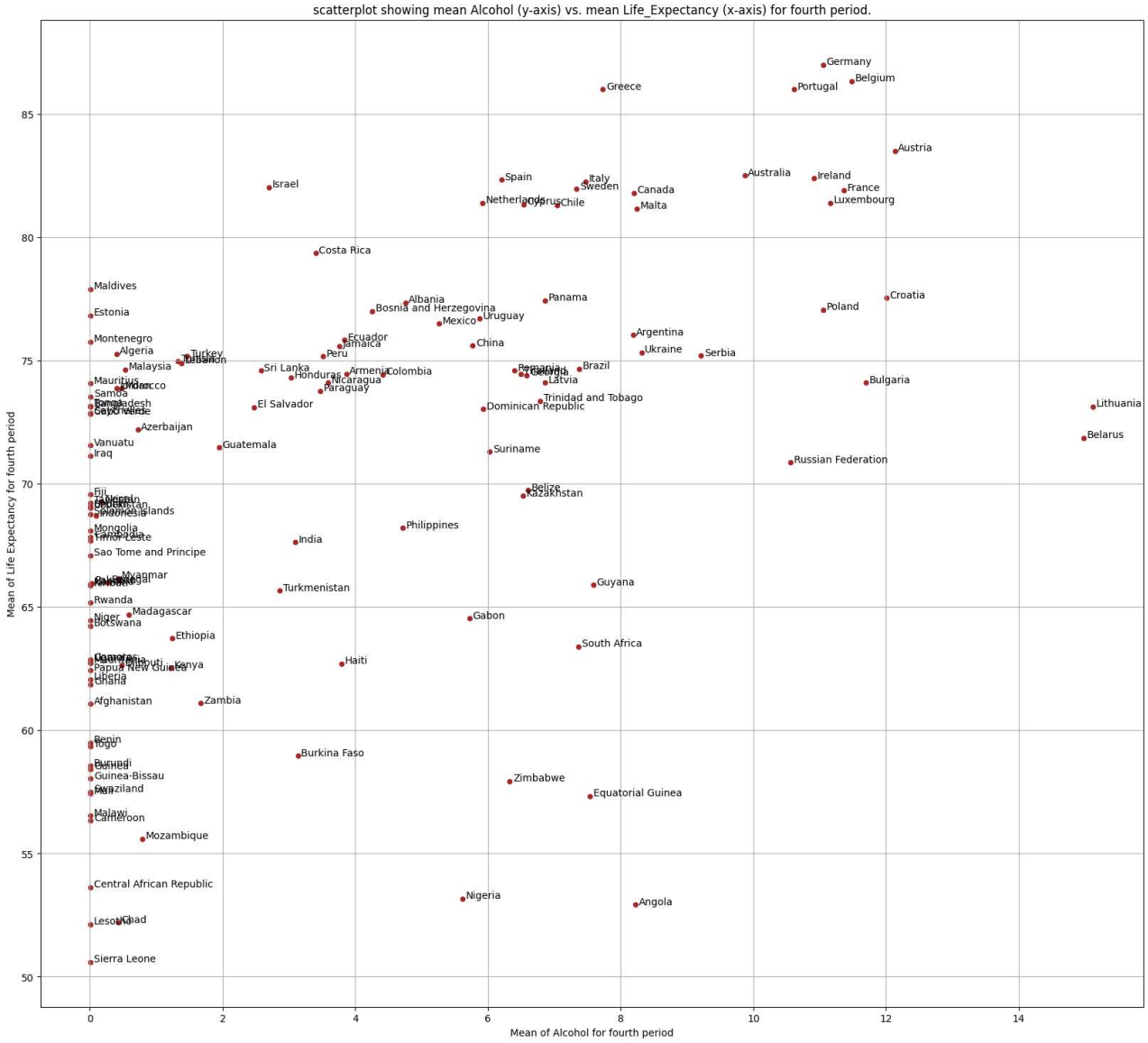
In [126]:
```
meansbyCountrythird = third.groupby('Country')['Life_Expectancy', 'Alcohol'].mean()
df = pd.DataFrame(meansbyCountrysecond, columns=["Alcohol", "Life_Expectancy"])

ax = df.plot(x="Alcohol", y="Life_Expectancy", kind="scatter", figsize=(20,18), color="green")
plt.xlabel("Mean of Alcohol for third period")
plt.ylabel("Mean of Life Expectancy for third period")
plt.title("scatterplot showing mean Alcohol (y-axis) vs. mean Life_Expectancy (x-axis) for third period.")
df.reset_index(inplace=True)
for i, txt in enumerate(df.Country):
   ax.annotate(txt, (df.Alcohol.iat[i]+0.05, df.Life_Expectancy.iat[i]))
plt.grid(True)
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_17484\2990627058.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  meansbyCountrythird = third.groupby('Country')['Life_Expectancy', 'Alcohol'].mean()
```



In [127]:
```
meansbyCountryfourth = fourth.groupby('Country')['Life_Expectancy', 'Alcohol'].mean()
df = pd.DataFrame(meansbyCountryfourth, columns=["Alcohol", "Life_Expectancy"])

ax = df.plot(x="Alcohol", y="Life_Expectancy", kind="scatter", figsize=(20,18), color="brown")
plt.xlabel("Mean of Alcohol for fourth period")
plt.ylabel("Mean of Life Expectancy for fourth period")
plt.title("scatterplot showing mean Alcohol (y-axis) vs. mean Life_Expectancy (x-axis) for fourth period.")
df.reset_index(inplace=True)
for i, txt in enumerate(df.Country):
   ax.annotate(txt, (df.Alcohol.iat[i]+0.05, df.Life_Expectancy.iat[i]))
plt.grid(True)
plt.show()
```

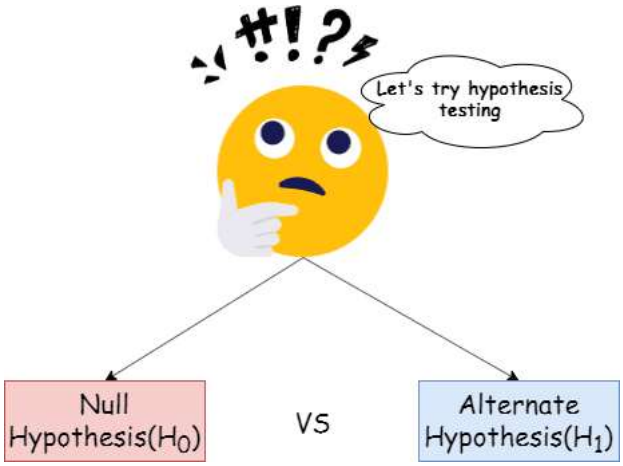
```
C:\Users\HP\AppData\Local\Temp\ipykernel_17484\3601612452.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  meansbyCountryfourth = fourth.groupby('Country')['Life_Expectancy', 'Alcohol'].mean()
```



Above Plots:

Above all the four scatter plots that we have created for mean of life expectancy vs mean of alcohol for all the countries:
We can see that for the first period: Malawi has the lowest and switzerland has the highest;
For the second period: Sierra Leone has the lowest and Sweden has the highest,
For the third period: Sierra Leone has the lowest and Sweden has the highest and,
For the fourth period: Sierra Leone has the lowest and Germany seems to be highest.
So one thing we can say is the in all the periods Sierra Leone has been the lowest.

HYPOTHESIS TESTING



Steps of Hypothesis Testing:

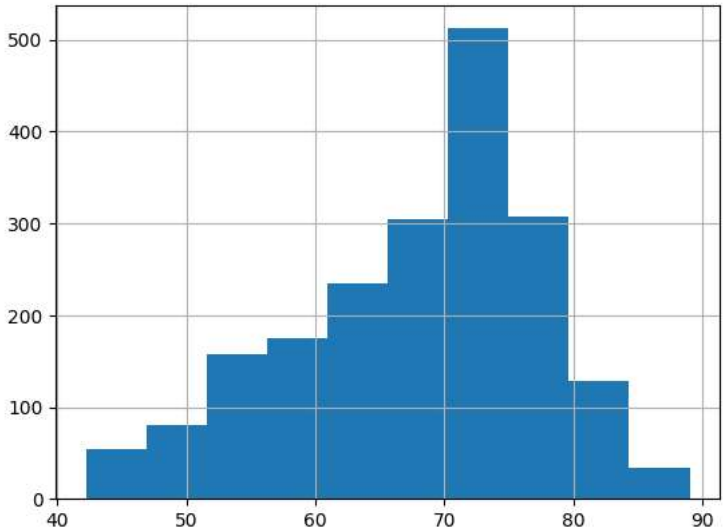
Purpose of test,
Assumptions of test,
How to do that test,
Interpret the result of the hypothesis test

Normality test using Shapiro-Wilk test

The Shapiro-Wilk test is a way we can analyze if a random sample comes from a **NORMAL DISTRIBUTION.**
The test gives us a value; small values indicate your sample is not normally distributed (you can reject the null hypothesis that your population is normally distributed if your values are under a certain threshold).

In [128]:
```
data['Life_Expectancy'].hist()
```

Out[128]: <AxesSubplot: >



In [129]:
```
DataToTest = data['Life_Expectancy']
stat, p = shapiro(DataToTest)

print('stat=%.2f, p = %.30f' % (stat,p))

if p > 0.05:
    print('Normal Distribution')
else:
    print('Not a Normal Distribution')
```

```
stat=0.96, p = 0.000000000000000000001508767845
Not a Normal Distribution
```

Here we ran Shapiro-Wilk test for a specific column of our dataset or dataframe which oru main consideration of the dataframe that is LIFE EXPECTANCY and we see that our p values is 0.0; so which is less than 0.05 therefore it is **NOT NORMALLY DISTRIBUTED.**

Lets generate normally distributed data from python

In [130]:
```
DataToTest = randn(100)
```

In [131]:
```
stat, p = shapiro(DataToTest)

print('stat=%.2f, p = %.30f' % (stat,p))

if p > 0.05:
    print('Normal Distribution')
else:
    print('Not a Normal Distribution')
```

```
stat=0.98, p = 0.111630119383335113525390625000
Normal Distribution
```

Above:
As we can see we randombly generated to run our test to our data to see the result; we know as per our intuition thhat this should result to normal distribution becasue we randomly generated the data and we can see from the result we get above that the p value is greater than 0.05 and hence is normally distributed.

In [132]:
```
stats.ttest_1samp(a=data_copy, popmean=310)
```

Out[132]:
```
Ttest_1sampResult(statistic=array([-2.07103082e+02,  1.99424196e+04, -4.41699317e+04,            nan,
                    nan, -1.28558506e+02,            nan,  1.16768796e+01,
                    nan,  9.97158699e+00,            nan, -9.05263686e+01,
                    nan,            nan,            nan, -3.29052987e+03,
                    nan,            nan,            nan,            nan,
                    nan,            nan]), pvalue=array([0.00000000e+00, 0.00000000e+00, 0.00000000e+00,            nan,
                   nan, 0.00000000e+00,            nan, 7.94503419e-31,
                   nan, 4.69949919e-23,            nan, 0.00000000e+00,
                   nan,            nan,            nan, 0.00000000e+00,
                   nan,            nan,            nan,            nan,
                   nan,            nan]))
```

ONE SAMPLE Z-TEST

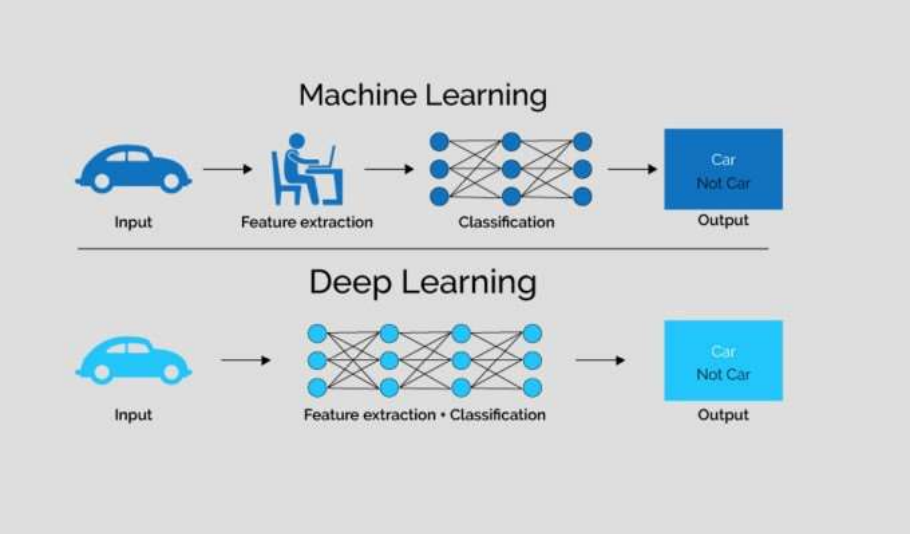
In [133]:
```
ztest ,pval = stests.ztest(data['Life_Expectancy'], x2=None, value=156)
print(float(pval))
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
0.0
reject null hypothesis
```

We run Z-test only if:
Your sample size is greater than 30. Otherwise, use a t test. Data points should be independent from each other. In other words, one data point isn't related or doesn't affect another data point.

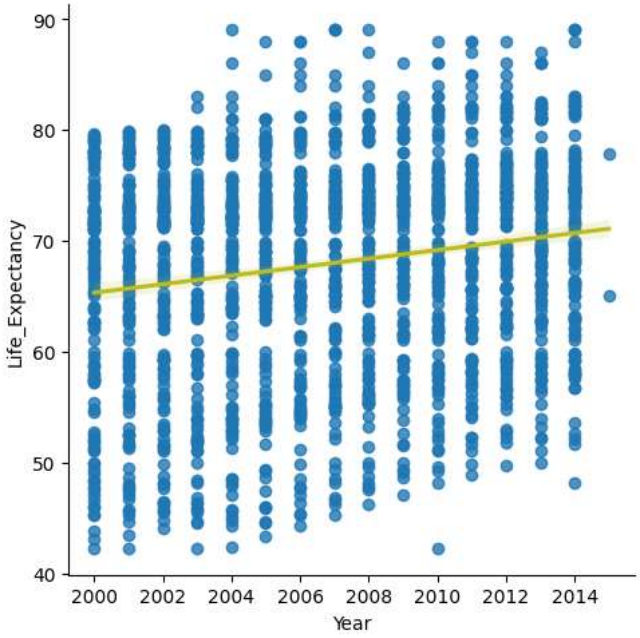
We get: Here we ran a one sample Z-Test and see that our p values comes out to be 0.0 which is definetly less than 0.05 so that rejects our null hypothesis.

TRAINING



In [134]:
```
sns.lmplot(x='Year',y='Life_Expectancy',data=data,fit_reg=True, line_kws={"color": "C8"})
```

Out[134]: <seaborn.axisgrid.FacetGrid at 0x241ded59a50>



Here we are fitting a **LIENAR REGRESSION MODEL** for Life Expectancy vs year

In [135]:
```
models = smf.ols(formula='Life_Expectancy ~ Year', data=data).fit()
```

In [136]:
```
models.summary()
```

Out[136]:

OLS Regression Results

Dep. Variable:	Life_Expectancy	**R-squared:**	0.029
Model:	OLS	**Adj. R-squared:**	0.029
Method:	Least Squares	**F-statistic:**	60.22
Date:	Sun, 11 Dec 2022	**Prob (F-statistic):**	1.35e-14
Time:	18:28:15	**Log-Likelihood:**	-7302.0
No. Observations:	1987	**AIC:**	1.461e+04
Df Residuals:	1985	**BIC:**	1.462e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-703.9591	99.476	-7.077	0.000	-899.048	-508.870
Year	0.3846	0.050	7.760	0.000	0.287	0.482

Omnibus:	116.097	**Durbin-Watson:**	0.161
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	99.112
Skew:	-0.473	**Prob(JB):**	3.01e-22
Kurtosis:	2.451	**Cond. No.**	9.32e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.32e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

Analysis:
We can see that our dependent variable is Life Expectancy; and we computed a lot of statistical information using stats model which also tells us that the covariance type if not robust.

Here we are doing **RESIDUAL VS YEAR** for the linear model

In [137]:
```
predict = models.predict(data.loc[:,['Year']])
```

In [138]:
```
data['newpredict'] = predict
```

In [139]:
```
data['Residual'] = data['Life_Expectancy'] - data['newpredict']
```

In [140]:
```
data.head()
```

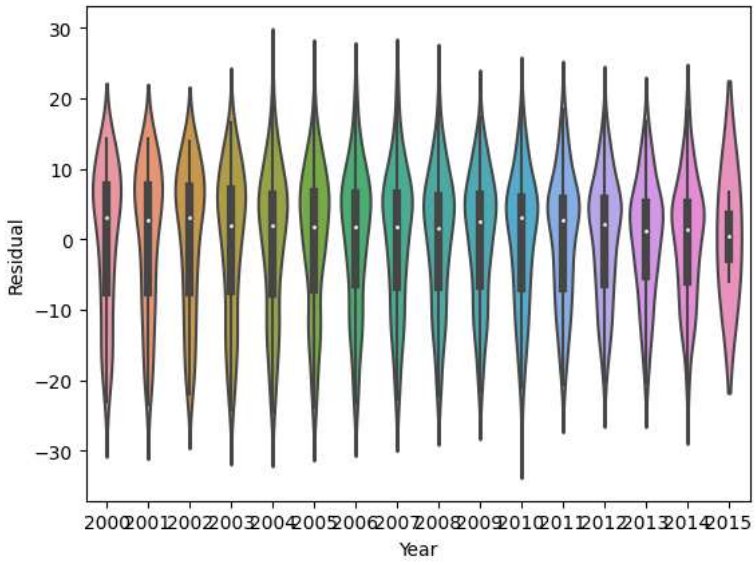
Out[140]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	GDP	P
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	584.259210	33
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	612.696514	
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	631.744976	31
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	669.959000	3
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	63.537231	2

5 rows × 26 columns

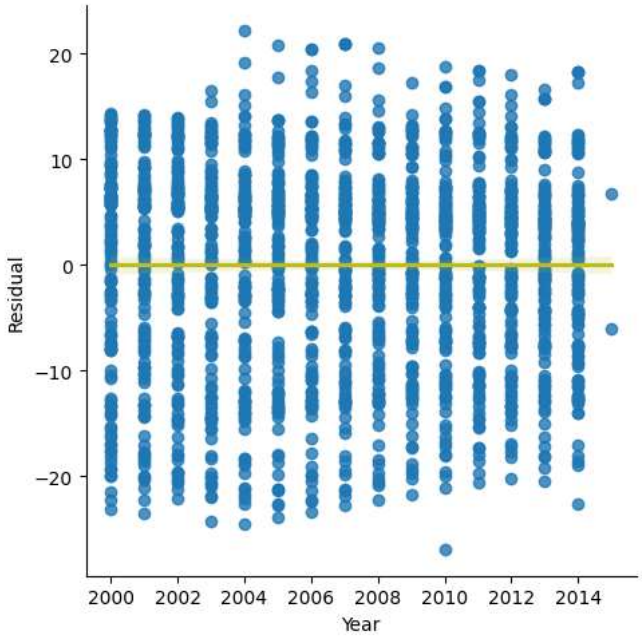
In [141]:
```
sns.violinplot(x='Year', y='Residual',data=data, title='Violin Plot using Seaborn via statsmodel')
```

Out[141]: <AxesSubplot: xlabel='Year', ylabel='Residual'>



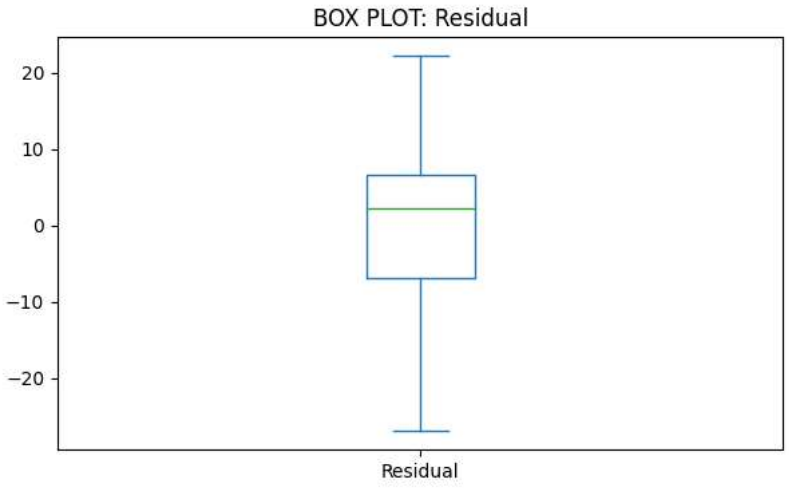
In [142]:
```
sns.lmplot(x='Year',y='Residual',data=data,fit_reg=True, line_kws={"color": "C8"})
```

Out[142]: <seaborn.axisgrid.FacetGrid at 0x241dfa45060>



Above:
We can see the differences in both of our scatter plots we for Life Expectancy Vs Year and Residual Vs Year; one of teh first thing we see in our above plot for Residual Vs Year is the Linear regression Line seems to be a straight or kind of a constant line.

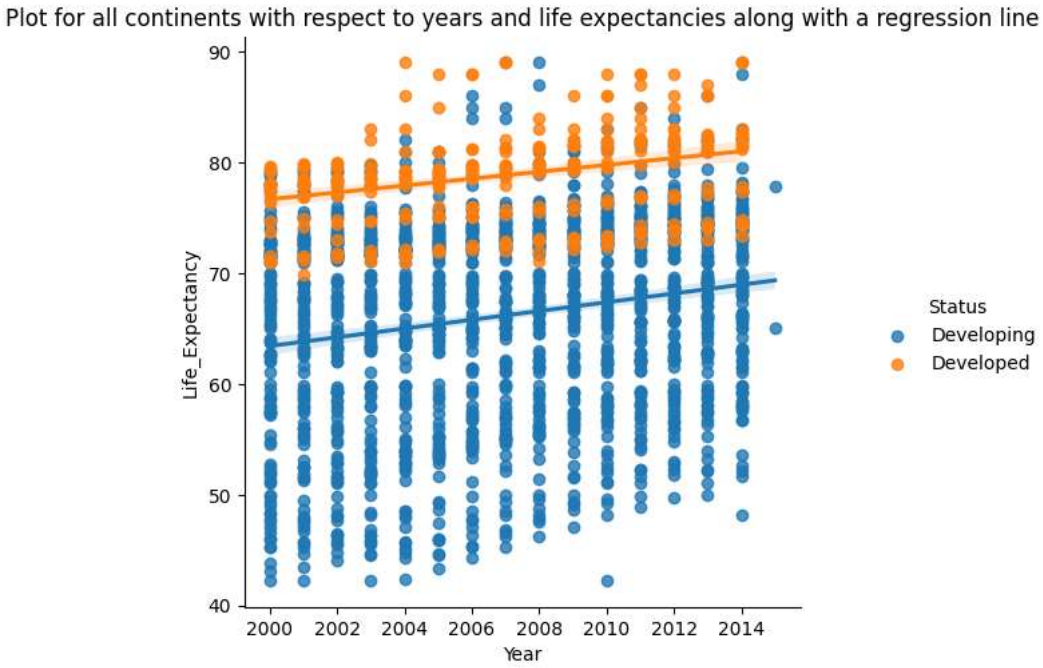
In [143]:
```
data['Residual'].plot(kind='box', title='BOX PLOT: Residual',figsize=(7,4))
plt.show()
```



Here we made a box plot of our residual that we computed and seems to be good as we do not have any outliers which is good!!

In [144]:
```
sns.lmplot(x="Year",y="Life_Expectancy",hue="Status",data=data)
ax = plt.gca()
ax.set_title("Plot for all continents with respect to years and life expectancies along with a regression line")
```

Out[144]:
```
Text(0.5, 1.0, 'Plot for all continents with respect to years and life expectancies along with a regression line')
```

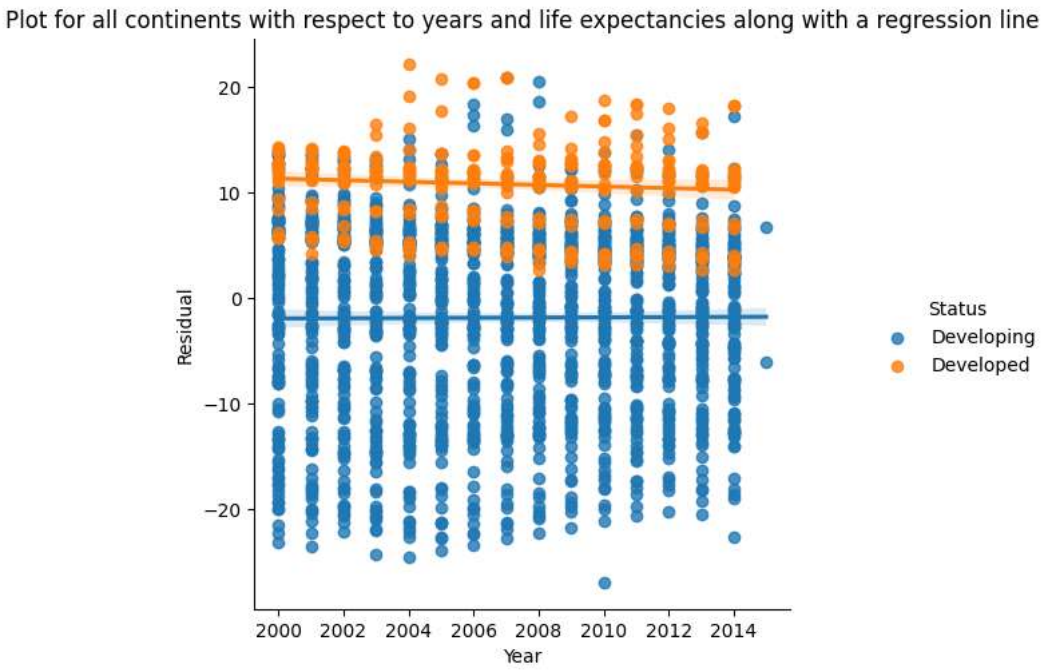


Above Plot:
We made a scatter plot for the life expectancies over years or time period with a regression line.

What do we observe:
We see a positive relationship or a linear relationship between the year and life expectancy as we saw that at the beginning of our project when we were exploring the data. So, we basically see a positive or a linear relationship for life expectancy and year for developing as well as developed status where developed countries have good life expectancy compared to developing countries.

In [145]:
```
sns.lmplot(x="Year",y="Residual",hue="Status",data=data)
ax = plt.gca()
ax.set_title("Plot for all continents with respect to years and life expectancies along with a regression line")
```

Out[145]:
```
Text(0.5, 1.0, 'Plot for all continents with respect to years and life expectancies along with a regression line')
```



Above Plot:
We made a scatter plot for the residual that we computed over years or time period with the regression line.

What do we observe:
We kind of see a linear relationship though they do not see to be increasing or inclining but have a relationship where both seems to be declining slowly where one seemd to be more of a straight or constant regression line.

Linear Regression

In [146]:
```
data.columns
```

Out[146]:
```
Index(['Country', 'Year', 'Status', 'Life_Expectancy', 'Adult_Mortality',
       'Infant_deaths', 'Alcohol', 'Percentage_expenditure', 'Hepatitis_B',
       'Measles', 'BMI', 'under-five-deaths', 'Polio', 'Total_expenditure',
       'Diphtheria', 'HIV/AIDS', 'GDP', 'Population', 'thinness_1-19_years',
       'thinness_5-9_years', 'Income_composition_of_resources', 'Schooling',
       'discretize', 'label', 'newpredict', 'Residual'],
      dtype='object')
```

Converting categorical data into numerical data using Label encoding

In [147]:
```
def Encoder(data):
          columnsToEncode = list(data.select_dtypes(include=['category','object']))
          le = LabelEncoder()
          for feature in columnsToEncode:
              try:
                  data[feature] = le.fit_transform(data[feature])
              except:
                  print('Error encoding '+feature)
          return data
Encoder(data)
```

Out[147]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	GDP	Popul
0	0	2015	1	65.0	263.0	62	0.01	71.279624	65.0	1154	...	584.259210	33736
1	0	2014	1	59.9	271.0	64	0.01	73.523582	62.0	492	...	612.696514	327
2	0	2013	1	59.9	268.0	66	0.01	73.219243	64.0	430	...	631.744976	31731
3	0	2012	1	59.5	272.0	69	0.01	78.184215	67.0	2787	...	669.959000	3696
4	0	2011	1	59.2	275.0	71	0.01	7.097109	68.0	3013	...	63.537231	2978
...	...	...	...	...	...	...	...	...	...	...	...	...	
2933	132	2004	1	44.3	723.0	27	4.36	0.000000	68.0	31	...	454.366654	12777
2934	132	2003	1	44.5	715.0	26	4.06	0.000000	7.0	998	...	453.351155	12633
2935	132	2002	1	44.8	73.0	25	4.43	0.000000	73.0	304	...	57.348340	125
2936	132	2001	1	45.3	686.0	25	1.72	0.000000	76.0	529	...	548.587312	12366
2937	132	2000	1	46.0	665.0	24	1.68	0.000000	79.0	1483	...	547.358878	12222

1987 rows × 26 columns

In [148]:
```
X = data[['Country','Year', 'Status', 'Adult_Mortality',
       'Infant_deaths', 'Alcohol', 'Percentage_expenditure', 'Hepatitis_B',
       'Measles', 'BMI', 'under-five-deaths', 'Polio', 'Total_expenditure',
       'Diphtheria', 'HIV/AIDS', 'GDP', 'Population', 'thinness_1-19_years',
       'thinness_5-9_years', 'Income_composition_of_resources', 'Schooling',
       'discretize', 'label', 'newpredict']]
y = data['Life_Expectancy']
```

In [149]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

In [150]:
```
model = LinearRegression()
```

In [151]:
```
model.fit(x_train, y_train)
```

Out[151]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [152]:
```
print("Model Cooeficient: ",model.coef_)
```

```
Model Cooeficient:  [ 1.90990608e-03 -1.78622357e-01 -1.31506496e+00 -1.48602770e-02
  9.20675290e-02 -1.74915620e-01  3.20181751e-04 -1.43517253e-03
  3.08617227e-07  5.79629880e-02 -6.79653079e-02  9.20289727e-03
  6.30206120e-02  2.06165615e-02 -4.55328940e-01  2.66079354e-05
 -1.43637620e-09 -7.62714790e-02  5.77338694e-03  9.15155662e+00
  8.84643133e-01  5.69496329e-01  4.33145076e-03 -6.87035038e-02]
```

In [153]:
```
print("Model Intercept: ",model.intercept_)
```

```
Model Intercept:  415.2820949977114
```

In [154]:
```
pd.DataFrame(model.coef_, X.columns, columns = ['Coeff'])
```

Out[154]:

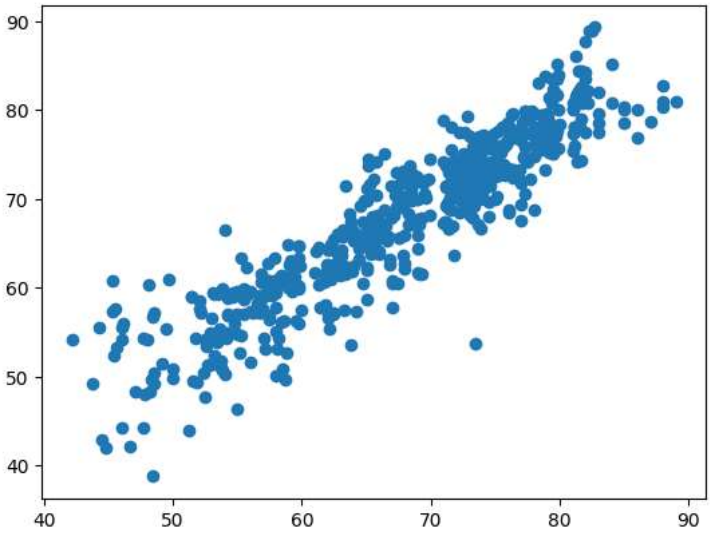
	Coeff
Country	1.909906e-03
Year	-1.786224e-01
Status	-1.315065e+00
Adult_Mortality	-1.486028e-02
Infant_deaths	9.206753e-02
Alcohol	-1.749156e-01
Percentage_expenditure	3.201818e-04
Hepatitis_B	-1.435173e-03
Measles	3.086172e-07
BMI	5.796299e-02
under-five-deaths	-6.796531e-02
Polio	9.202897e-03
Total_expenditure	6.302061e-02
Diphtheria	2.061656e-02
HIV/AIDS	-4.553289e-01
GDP	2.660794e-05
Population	-1.436376e-09
thinness_1-19_years	-7.627148e-02
thinness_5-9_years	5.773387e-03
Income_composition_of_resources	9.151557e+00
Schooling	8.846431e-01
discretize	5.694963e-01
label	4.331451e-03
newpredict	-6.870350e-02

Making predictions for our model

In [155]:
```
predictions = model.predict(x_test)
```

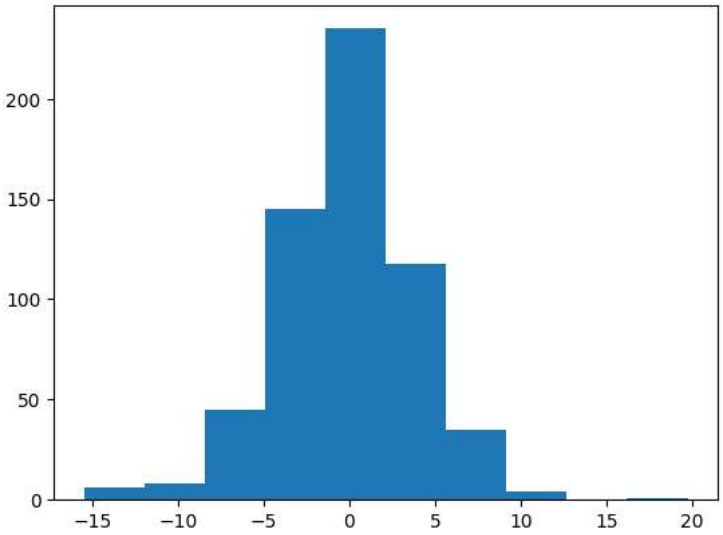
In [156]:
```
plt.scatter(y_test, predictions)
```

Out[156]:
```
<matplotlib.collections.PathCollection at 0x241dea4fc40>
```



In [157]:
```
plt.hist(y_test - predictions)
```

Out[157]:
```
(array([  6.,   8.,  45., 145., 235., 118.,  35.,   4.,   0.,   1.]),
 array([-15.47486166, -11.95338139,  -8.43190111,  -4.91042084,
         -1.38894056,   2.13253972,   5.65401999,   9.17550027,
         12.69698054,  16.21846082,  19.7399411 ]),
 <BarContainer object of 10 artists>)
```



Testing the Performance of our Model

In [158]:
```
print("Mean absolute Error: ",metrics.mean_absolute_error(y_test, predictions))
```

```
Mean absolute Error:  3.007531505775279
```

In [159]:
```
print("Mean Squared Error: ",metrics.mean_squared_error(y_test, predictions))
```

```
Mean Squared Error:  15.682662983452122
```

In [160]:
```
print("Root mean squared Error: ",np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
Root mean squared Error:  3.96013421280796
```

Support Vector Machine

In [161]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=4)
```

In [162]:
```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(1589, 24)
(1589,)
(398, 24)
(398,)
```

In [163]:
```
sc = StandardScaler()
#Fitting the X training set
x_train = sc.fit_transform(x_train)
#Fitting the Y training set
x_test = sc.fit_transform(x_test)
```

In [164]:
```
svm_regress = svm.SVR()
```

In [165]:
```
#Fitting the X training and Y training set
svm_regress.fit(x_train, y_train)
```

Out[165]:
```
SVR()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

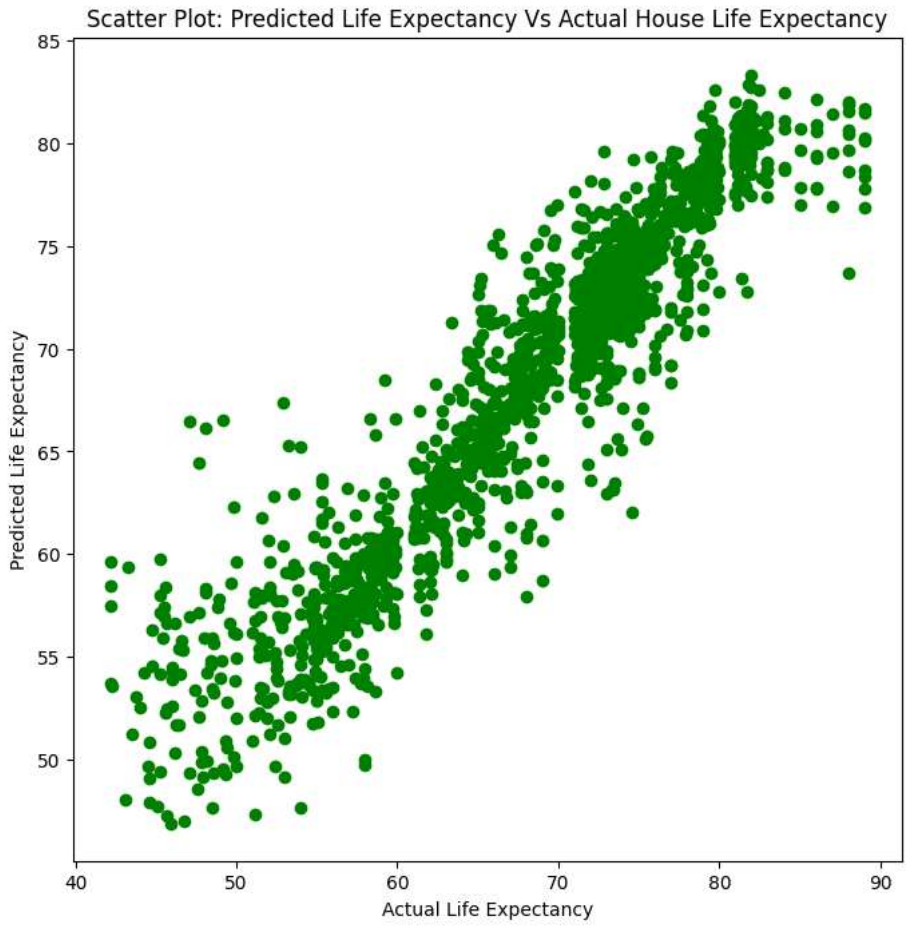
In [166]:
```
y_prediction = svm_regress.predict(x_train)
print(y_prediction.shape)
```

```
(1589,)
```

In [167]:
```
#Evaluating the model
R_squared = metrics.r2_score(y_train, y_prediction)
print("R_squared: ", R_squared)
Adjusted_R_squared = 1-(1-metrics.r2_score(y_train, y_prediction))*(len(y_train)-1)/(len(y_train)-x_train.shape[1]-1)
print("Adjusted_R_squared: ",Adjusted_R_squared)

Mean_Absolute_Error = metrics.mean_absolute_error(y_train, y_prediction)
print("Mean_Absolute_Error: ",Mean_Absolute_Error)

Mean_Squared_Error = metrics.mean_squared_error(y_train, y_prediction)
print("Mean_Squared_Error: ",Mean_Squared_Error)

Root_Mean_Squared_Error = np.sqrt(metrics.mean_squared_error(y_train, y_prediction))
print("Root_Mean_Squared_Error: ",Root_Mean_Squared_Error)
```

```
R_squared:  0.8588189184804081
Adjusted_R_squared:  0.8566524568714119
Mean_Absolute_Error:  2.439039817561648
Mean_Squared_Error:  13.401352384364326
Root_Mean_Squared_Error:  3.660785760511577
```

In [168]:
```
plt.figure(figsize=(8, 8))
plt.scatter(y_train, y_prediction, color='green')
plt.xlabel("Actual Life Expectancy")
plt.ylabel("Predicted Life Expectancy")
plt.title("Scatter Plot: Predicted Life Expectancy Vs Actual House Life Expectancy")
plt.show()
```



Testing Set

In [169]:
```
Y_PredictionTest = svm_regress.predict(x_test)
```

In [170]:
```
# Evaluating the model
R_squared = metrics.r2_score(y_test, Y_PredictionTest)
print("R_squared: ",R_squared)

Adjusted_R_squared = 1-(1-metrics.r2_score(y_test, Y_PredictionTest))*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1)
print("Adjusted_R_squared: ",Adjusted_R_squared)

Mean_Absolute_Error = metrics.mean_absolute_error(y_test, Y_PredictionTest)
print("Mean_Absolute_Error: ",Mean_Absolute_Error)

Mean_Squared_Error = metrics.mean_squared_error(y_test, Y_PredictionTest)
print("Mean_Squared_Error: ",Mean_Squared_Error)

Root_Mean_Squared_Error = np.sqrt(metrics.mean_squared_error(y_test, Y_PredictionTest))
print("Root_Mean_Squared_Error: ",Root_Mean_Squared_Error)
```

```
R_squared:  0.8364568608433269
Adjusted_R_squared:  0.8259339778949083
Mean_Absolute_Error:  2.7033223569668343
Mean_Squared_Error:  14.654081564781793
Root_Mean_Squared_Error:  3.8280649896235817
```

In [171]:
```
svm_regress.score(x_test,y_test)
```

Out[171]:
```
0.8364568608433269
```

In [172]:
```
##Holdout Validation
result = svm_regress.score(x_test, y_test)
print("Accuracy using Hold-out Validation: %.2f%%" % (result*100.0))
```

```
Accuracy using Hold-out Validation: 83.65%
```

In [173]:
```
cutoff = 0.7                              # decide on a cutoff limit
y_pred_classes = np.zeros_like(Y_PredictionTest)    # initialise a matrix full with zeros
y_pred_classes[Y_PredictionTest > cutoff] = 1       # add a 1 if the cutoff was breached

y_test_classes = np.zeros_like(Y_PredictionTest)
y_test_classes[y_test > cutoff] = 1

confusion_matrix(y_test_classes, y_pred_classes)
```

Out[173]:
```
array([[398]], dtype=int64)
```

In [174]:
```
#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)

#build multiple linear regression model
model = LinearRegression()

#use k-fold CV to evaluate model
scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error',cv=cv, n_jobs=-1)

#view mean absolute error
print("MAE using cross-validation: ",np.mean(np.absolute(scores)))

scores1 = cross_val_score(model, X, y, scoring='neg_mean_squared_error',cv=cv, n_jobs=-1)

#view RMSE
print("RMAE using cross-validation: ",np.sqrt(np.mean(np.absolute(scores1))))
```

```
MAE using cross-validation:  2.9791160893707134
RMAE using cross-validation:  3.9597743687204146
```

Suppose we change our X and y means variables as in our target variable changes from Life Expectancy to Status and we fit the model and compute accuracy, confusion matrix and classficiation report and we use Logistic Regression and Regression both.

In [175]:
```
X = data[['Country','Year','Adult_Mortality', 'Life_Expectancy',
       'Infant_deaths', 'Alcohol', 'Percentage_expenditure', 'Hepatitis_B',
       'Measles', 'BMI', 'under-five-deaths', 'Polio', 'Total_expenditure',
       'Diphtheria', 'HIV/AIDS', 'GDP', 'Population', 'thinness_1-19_years',
       'thinness_5-9_years', 'Income_composition_of_resources', 'Schooling']]
y = data['Status']
```

In [176]:
```
classifier = LogisticRegression()
classifier2 = LinearRegression()
classifier.fit(X, y)
classifier2.fit(X, y)
```

Out[176]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [177]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

In [178]:
```
predictions = classifier.predict(x_test)
predictions2 = classifier2.predict(x_test)
```

In [179]:
```
print("Analysis for Logistic Regression")
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
Analysis for Logistic Regression
0.8894472361809045
[[ 33  60]
 [  6 498]]
              precision    recall  f1-score   support

           0       0.85      0.35      0.50        93
           1       0.89      0.99      0.94       504

    accuracy                           0.89       597
   macro avg       0.87      0.67      0.72       597
weighted avg       0.89      0.89      0.87       597
```

In [180]:
```
print("Analysis for Linear Regression")
print(accuracy_score(y_test, predictions2.round()))
print(confusion_matrix(y_test, predictions2.round()))
print(classification_report(y_test, predictions2.round()))
```

```
Analysis for Linear Regression
0.9279731993299832
[[ 57  36]
 [  7 497]]
              precision    recall  f1-score   support

           0       0.89      0.61      0.73        93
           1       0.93      0.99      0.96       504

    accuracy                           0.93       597
   macro avg       0.91      0.80      0.84       597
weighted avg       0.93      0.93      0.92       597
```

Random Forest

In [181]:
```
X = data[['Country','Year', 'Status', 'Adult_Mortality',
       'Infant_deaths', 'Alcohol', 'Percentage_expenditure', 'Hepatitis_B',
       'Measles', 'BMI', 'under-five-deaths', 'Polio', 'Total_expenditure',
       'Diphtheria', 'HIV/AIDS', 'GDP', 'Population', 'thinness_1-19_years',
       'thinness_5-9_years', 'Income_composition_of_resources', 'Schooling',
       'discretize', 'label', 'newpredict']]
y = data['Life_Expectancy']
```

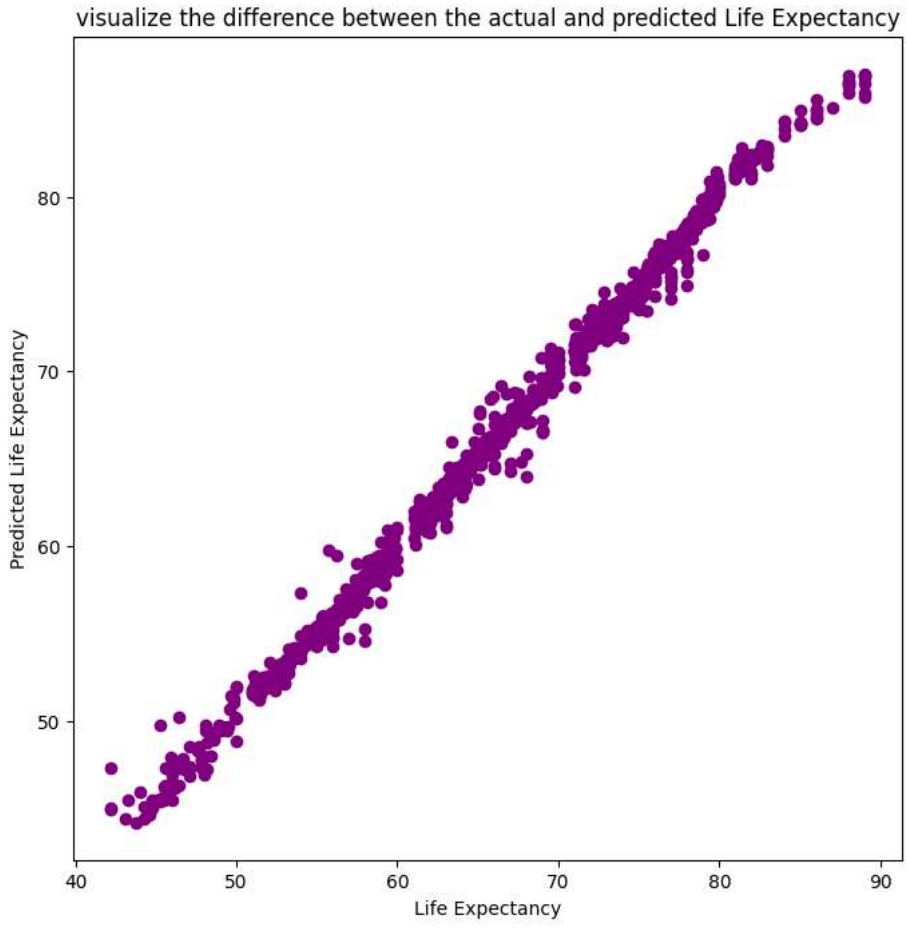
In [182]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=66)
```

In [183]:
```
regression=RandomForestRegressor()
regression.fit(X_train, y_train)

#prediction
y_predect=regression.predict(X_train)
```

In [184]:
```
# Evaluating the model
R_squared = metrics.r2_score(y_train, y_predect)
print("R_squared: ",R_squared)

Adjusted_R_squared = 1-(1-metrics.r2_score(y_train, y_predect))*(len(y_train)-1)/(len(y_train)-X_train.shape[1]-1)
print("Adjusted_R_squared: ",Adjusted_R_squared)

Mean_Absolute_Error = metrics.mean_absolute_error(y_train, y_predect)
print("Mean_Absolute_Error: ",Mean_Absolute_Error)

Mean_Squared_Error = metrics.mean_squared_error(y_train, y_predect)
print("Mean_Squared_Error: ",Mean_Squared_Error)

Root_Mean_Squared_Error = np.sqrt(metrics.mean_squared_error(y_train, y_predect))
print("Root_Mean_Squared_Error: ",Root_Mean_Squared_Error)
```

```
R_squared:  0.994173322601528
Adjusted_R_squared:  0.9940662473660278
Mean_Absolute_Error:  0.4483959429000752
Mean_Squared_Error:  0.5431332103681425
Root_Mean_Squared_Error:  0.7369757189813939
```

In [185]:
```
plt.figure(figsize=(8, 8))
plt.scatter(y_train, y_predect, color="purple")
plt.xlabel("Life Expectancy")
plt.ylabel("Predicted Life Expectancy")
plt.title("visualize the difference between the actual and predicted Life Expectancy")
plt.show()
```



In [186]:
```
# Predicting the Test data with model
y_test_predect=regression.predict(X_test)
```

In [187]:
```
# Evaluating the model
R_squared = metrics.r2_score(y_test, y_test_predect)
print("R_squared: ",R_squared)

Adjusted_R_squared = 1-(1-metrics.r2_score(y_test, y_test_predect))*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1)
print("Adjusted_R_squared: ",Adjusted_R_squared)

Mean_Absolute_Error = metrics.mean_absolute_error(y_test, y_test_predect)
print("Mean_Absolute_Error: ",Mean_Absolute_Error)

Mean_Squared_Error = metrics.mean_squared_error(y_test, y_test_predect)
print("Mean_Squared_Error: ",Mean_Squared_Error)

Root_Mean_Squared_Error = np.sqrt(metrics.mean_squared_error(y_test, y_test_predect))
print("Root_Mean_Squared_Error: ",Root_Mean_Squared_Error)
```

```
R_squared:  0.9601944913150153
Adjusted_R_squared:  0.9586804941542553
Mean_Absolute_Error:  1.2820884146341487
Mean_Squared_Error:  3.7783941615853665
Root_Mean_Squared_Error:  1.9438091885741682
```

In [188]:
```
regression.score(X_test,y_test)
```

Out[188]:
```
0.9601944913150153
```

In [189]:
```
##Holdout Validation
result = regression.score(X_test, y_test)
print("Accuracy using Hold-out Validation: %.2f%%" % (result*100.0))
```

```
Accuracy using Hold-out Validation: 96.02%
```

In [190]:
```
cutoff = 0.7                              # decide on a cutoff limit
y_pred_classes = np.zeros_like(y_test_predect)    # initialise a matrix full with zeros
y_pred_classes[y_test_predect > cutoff] = 1       # add a 1 if the cutoff was breached

y_test_classes = np.zeros_like(y_test_predect)
y_test_classes[y_test > cutoff] = 1

confusion_matrix(y_test_classes, y_pred_classes)
```

Out[190]:
```
array([[656]], dtype=int64)
```

In [191]:
```
data.head()
```

Out[191]:

	Country	Year	Status	Life_Expectancy	Adult_Mortality	Infant_deaths	Alcohol	Percentage_expenditure	Hepatitis_B	Measles	...	GDP	Populatic
0	0	2015	1	65.0	263.0	62	0.01	71.279624	65.0	1154	...	584.259210	33736494
1	0	2014	1	59.9	271.0	64	0.01	73.523582	62.0	492	...	612.696514	327582
2	0	2013	1	59.9	268.0	66	0.01	73.219243	64.0	430	...	631.744976	31731688
3	0	2012	1	59.5	272.0	69	0.01	78.184215	67.0	2787	...	669.959000	3696958
4	0	2011	1	59.2	275.0	71	0.01	7.097109	68.0	3013	...	63.537231	2978599

5 rows × 26 columns

In [192]:
```
#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)

#build multiple linear regression model
model = LinearRegression()

#use k-fold CV to evaluate model
scores = cross_val_score(model, X, y, scoring='neg_mean_absolute_error',cv=cv, n_jobs=-1)

#view mean absolute error
print("MAE using cross-validation: ",np.mean(np.absolute(scores)))

scores1 = cross_val_score(model, X, y, scoring='neg_mean_squared_error',cv=cv, n_jobs=-1)

#view RMSE
print("RMAE using cross-validation: ",np.sqrt(np.mean(np.absolute(scores1))))
```

```
MAE using cross-validation:  2.9791160893707134
RMAE using cross-validation:  3.9597743687204146
```

CONCLUSION

Our goal was to understand, observe, and analyze the relationship between the column with the Life Expectancy for different countries over the years and also based on the status whether developing or else developed.
We made a lot of analysis and did explore the data set using different Machine Learning and Data Science techniques and algorithms and do see that Life Expectancy is our main column for our dataset in many different perspectives.
We also observed that how different different column have impact on the life expectancy; for instance; alcohol; the more you consumed the lesser your life expectancy becomes and that has its own reasons behind it.
This definitely tells us how important LIFE EXPECTANCY is not just as an individual but as a sample or as a population as a whole as well as for the economy. The better and good the life expectancy is the more better the performance of the economy and vice versa. Like we say for the status: DEVELOPED and DEVELOPING.
The higher the life expectancy the more you come under the developed country category and lower the life expectancy you come under the category of the developing because even one individual can impact the performance of themselves as well as the performance of the economy as a whole which we could also see when we trained our data using **LINEAR REGRESSION, SUPPORT VECTOR MACHINE, RANDOM FOREST.**