Prediction of Life Expectancy

Group Members:

Sheela Bhattacharjee
10800220066
Asansol Engineering College

Bishnu Sharma
10800220065
Asansol Engineering College

Biplab Gorain
10800221149
Asansol Engineering College

Priyanshu Burman
10800220058
Asansol Engineering College

Contents

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty, Prof. Arnab Chakraborty for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Sheela Bhattacharjee

Bishnu Sharma

Biplab Gorain

Priyanshu Burman

Objective

Life expectancy is a statistical measure of the average time a human being is expected to live. The term “life expectancy” refers to the number of years a person can expect to live. By definition, life expectancy is based on an estimate of the average age that members of a particular population group will be when they die. The purpose of this project is to predict the life expectancy of a person considering the various factors. The project will be helpful in improving the health condition of the society and give insights about some crucial factors such as [Alcohol intake, GDP growth, schooling, adult mortality, total and cost expenditure etc]. This project will help in building a tool that can be used by healthcare providers to identify patients who are at risk of developing life-threatening diseases. It can be used by insurance companies to develop more accurate pricing models and reduce the risk of losses due to unexpected deaths. The dataset used or the training of the model was downloaded from kaggle.com and Python is used to write the code for machine learning model. In this dataset our target attribute is ‘Life Expectancy’. We have pre-processed the given dataset whenever needed. Then, we would train 3 models i.e. ‘Linear Regression Model’, ‘Random Forest regressor model’ and ‘Decision Tree regressor model’ . After training the above mentioned models, we will need to find out the accuracy score for each of them. Our next step would be to use the trained models to predict the outcomes using the given test dataset and compare the outcome of each model. We would then choose the best model based on the accuracy score.

Dataset Information

We have used the dataset provided by the Global Health Observatory (GHO) data repository under World Health Organization (WHO) that keeps track of the health status as well as many other related factors for all countries. The dataset consists of 22 Columns and 2938 rows. The columns in the dataset are:

1) Country
2) Year
3) Status – Developed or Developing status
4) Life expectancy – Life Expectancy in age
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 – Alcohol, recorded per capita (15+) consumption

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) BMI – Average Body Mass Index of entire population
12) under-five deaths – Number of under-five deaths per 1000 population
13) Polio – Polio (Pol3) immunization coverage among 1-year-olds (%)
14) Total expenditure – General government expenditure on health as a percentage of total government expenditure (%)
15) Diphtheria – Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds (%)
16) HIV/AIDS – Deaths per 1 000 live births HIV/AIDS (0-4 years)
17) GDP – Gross Domestic Product per capita (in USD)
18) Population – Population of the country
19) thinness 1-19 years – Prevalence of thinness among children and adolescents for Age 10 to 19 (%)
20) thinness 5-9 years – Prevalence of thinness among children for Age 5to9(%)
21) Income composition – Human Development Index in terms of income composition of resources (index ranging from 0 to 1)
22) Schooling – Number of years of Schooling(years)

Methodology

In this project we have used the following steps for solving the problem:

- Load the dataset from Kaggle.
- Study the dataset(knowing the data we have).
- Describe the dataset.
- Study of every variable separately.
- Visualize the dataset.
- Data pre-processing :- To determine whether the dataset has null values or outliers which may cause noise or overfitting and can affect the accuracy of the model to be trained.
- Take necessary steps for pre-processing :- To fill or drop the null values and eliminating the outliers.
- Plotting graphs before and after pre-processing.
- Feature Selection.
- Encoding categorical data to numerical data.
- Split the dataset for training and testing purpose.
- Fit the previously splited data in the selected 3 models.
- Model Building and checking the accuracy for each of them.
- Comparing the accuracy of each model and choosing the best one.

Project Scope

The future scopes of a life expectancy prediction project using ML are vast and exciting. As technology continues to advance, the potential applications and benefits of this project are likely to grow. Some of the scopes of the project are listed below:

- **Real-time monitoring:** Real-time monitoring of an individual's health could be used to continually update a life expectancy prediction model. This could provide more accurate predictions and allow for early interventions to prevent life-threatening diseases.
- **Environmental factors:** The impact of environmental factors such as pollution, climate change, and access to healthcare on life expectancy is a growing area of research.
- **Integration with financial planning:** A life expectancy prediction model could be integrated with financial planning tools to help individuals better plan for their retirement and financial future.

Data Description

Source of the data: Kaggle. The given dataset is a shortened version of the original dataset in Kaggle.

Data Description: The given train dataset has 2938 rows and 22 columns.

Columns	Attribute Name	Type	Description	Target value
Country	Country	Categorical	Country Names (193 Unique Countries name)	No
Year	Year	Categorical	Year (2000 - 2013)	No
Status	Status	Categorical	Developed or Developing status	No
Life Expectancy	Life expectancy	Non-Categorical	Life Expectancy in age	**YES**
Adult Mortality	Adult Mortality	Non-Categorical	Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population)	No
Infant Deaths	infant deaths	Non-Categorical	Number of Infant Deaths per 1000 population	No
Alcohol	Alcohol	Non-Categorical	Alcohol, recorded per capita (15+) consumption (in litres of pure alcohol)	No
Percentage Expenditure	percentage expenditure	Non-Categorical	Expenditure on health as a percentage of Gross Domestic Product per capita(%)	No
Hepatitis B	Hepatitis B	Non-Categorical	Hepatitis B (HepB) immunization coverage among 1-year-olds (%)	No
Measles	Measles	Non-Categorical	Measles - number of reported cases per 1000 population	No
BMI	BMI	Non-Categorical	Average Body Mass Index of entire population	No
Under-Five Deaths	under-five deaths	Non-Categorical	Number of under-five deaths per 1000 population	No
Polio	Polio	Non-Categorical	Polio (Pol3) immunization coverage among 1-year-olds (%)	No
Total Expenditure	Total expenditure	Non-Categorical	General government expenditure on health as a percentage of total government expenditure (%)	No
Diphtheria	Diphtheria	Non-Categorical	Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds (%)	No
HIV/AIDS	HIV/AIDS	Non-Categorical	Deaths per 1 000 live births HIV/AIDS (0-4 years)	No
GDP	GDP	Non-Categorical	Gross Domestic Product per capita (in USD)	No
Population	Population	Non-Categorical	Population of the country	No
Thinness 1-19 years	thinness 1-19 years	Non-Categorical	thinness 1-19 years	No
Schooling	Schooling	Non-Categorical	Number of years of Schooling(years)	No

Number statistics

	Year	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
count	2938.000000	2928.000000	2928.000000	2938.000000	2744.000000	2938.000000	2385.000000	2938.000000	2904.000000	2938.000000	2919.000000	2712.00000	2919.000000	2938.000000	2490.000000	2.286000e+03	2904.000000	2904.000000	2771.000000	2775.000000
mean	2007.518720	69.224932	164.796448	30.303948	4.602861	738.251295	80.940461	2419.592240	38.321247	42.035739	82.550188	5.93819	82.324084	1.742103	7483.158469	1.275338e+07	4.839704	4.870317	0.627551	11.992793
std	4.613841	9.523867	124.292079	117.926501	4.052413	1987.914858	25.070016	11467.272489	20.044034	160.445548	23.428046	2.49832	23.716912	5.077785	14270.169342	6.101210e+07	4.420195	4.508882	0.210904	3.358920
min	2000.000000	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	0.000000	1.000000	0.000000	3.000000	0.37000	2.000000	0.100000	1.681350	3.400000e+01	0.100000	0.100000	0.000000	0.000000
25%	2004.000000	63.100000	74.000000	0.000000	0.877500	4.685343	77.000000	0.000000	19.300000	0.000000	78.000000	4.26000	78.000000	0.100000	463.935626	1.957932e+05	1.600000	1.500000	0.493000	10.100000
50%	2008.000000	72.100000	144.000000	3.000000	3.755000	64.912906	92.000000	17.000000	43.500000	4.000000	93.000000	5.75500	93.000000	0.100000	1766.947595	1.386542e+06	3.300000	3.300000	0.677000	12.300000
75%	2012.000000	75.700000	228.000000	22.000000	7.702500	441.534144	97.000000	360.250000	56.200000	28.000000	97.000000	7.49250	97.000000	0.800000	5910.806335	7.420359e+06	7.200000	7.200000	0.779000	14.300000
max	2015.000000	89.000000	723.000000	1800.000000	17.870000	19479.911610	99.000000	212183.000000	87.300000	2500.000000	99.000000	17.60000	99.000000	50.600000	119172.741800	1.293859e+09	27.700000	28.600000	0.948000	20.700000

Data Pre-processing

We converted our categorical attribute status to numeric values

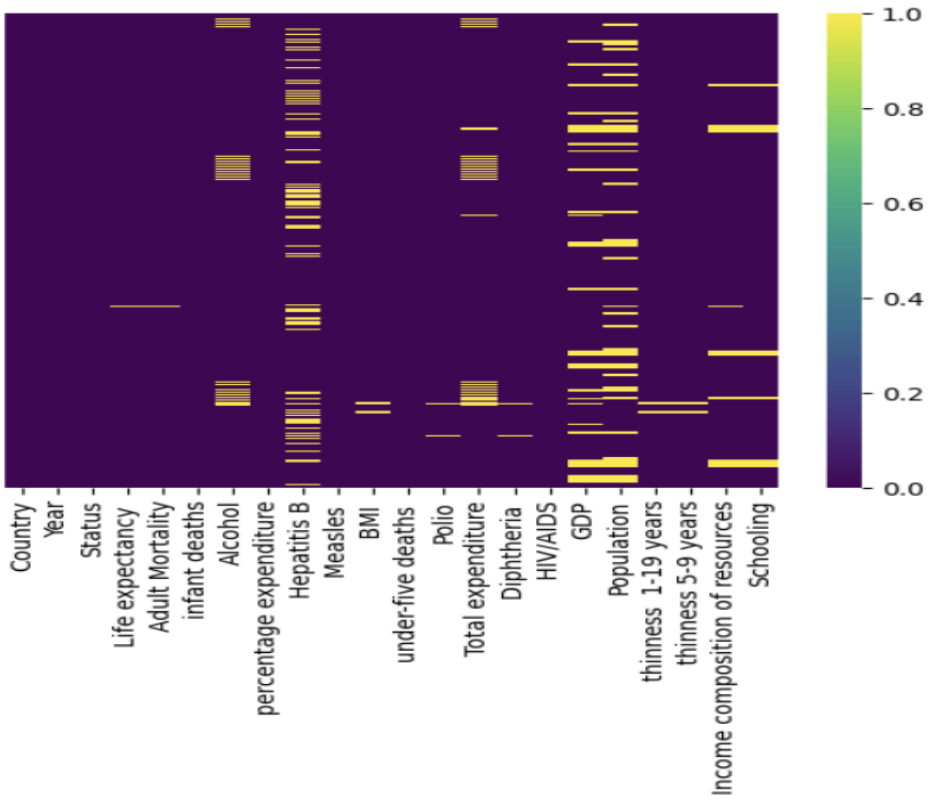
Non-numeric to numeric conversion		
Column	Initial Value	Final Value
Status	Developing	0
	Developed	1

We have identified all the null values present in the dataset:

Column	**Count of null values**
Country	0
Year	0
Status	0
Life Expectancy	10
Adult Mortality	0
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

To get a more clear idea about the null values we have created a heatmap using seaborn library function. The heatmap is given below:



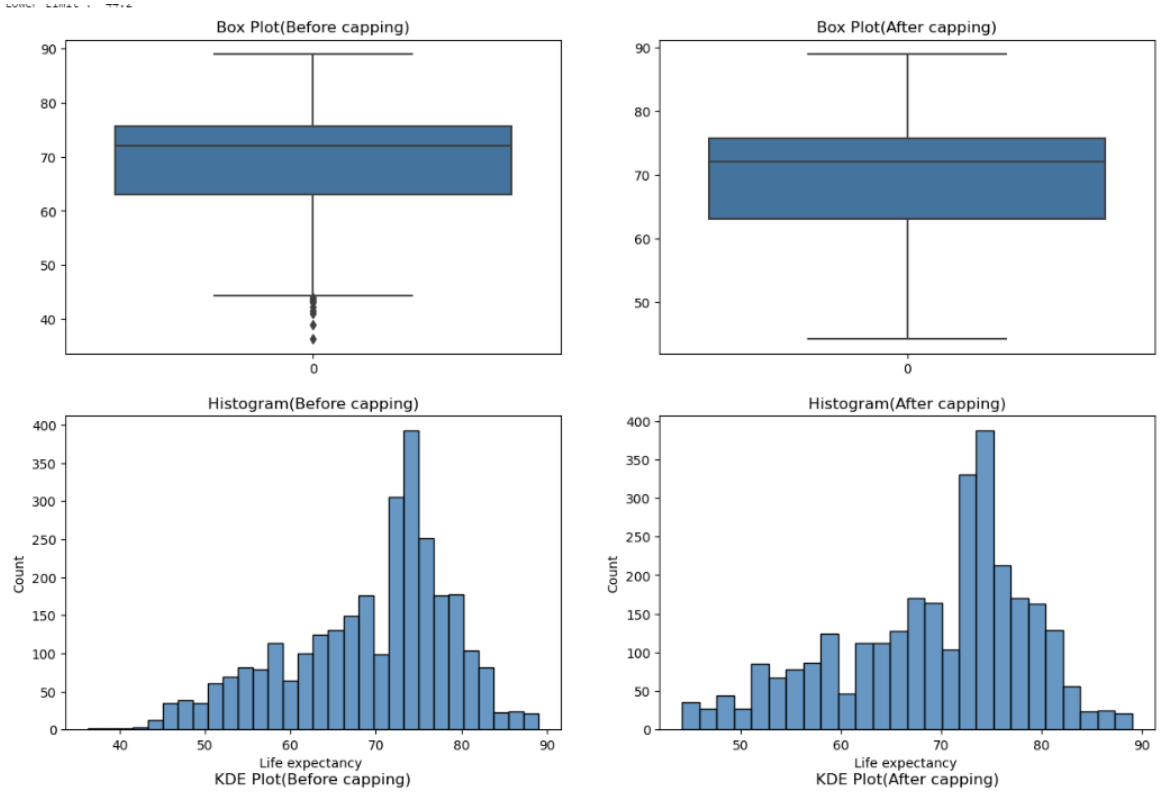
Filling and Dropping of Null Values: The feature variables which are containing less than 5% null values are directly removed others are filled by the mean and median of the feature variable. Mean is used when the data distribution is natural or like natural and median is used when the feature variable is right or left Skewed.

Handling Outliers

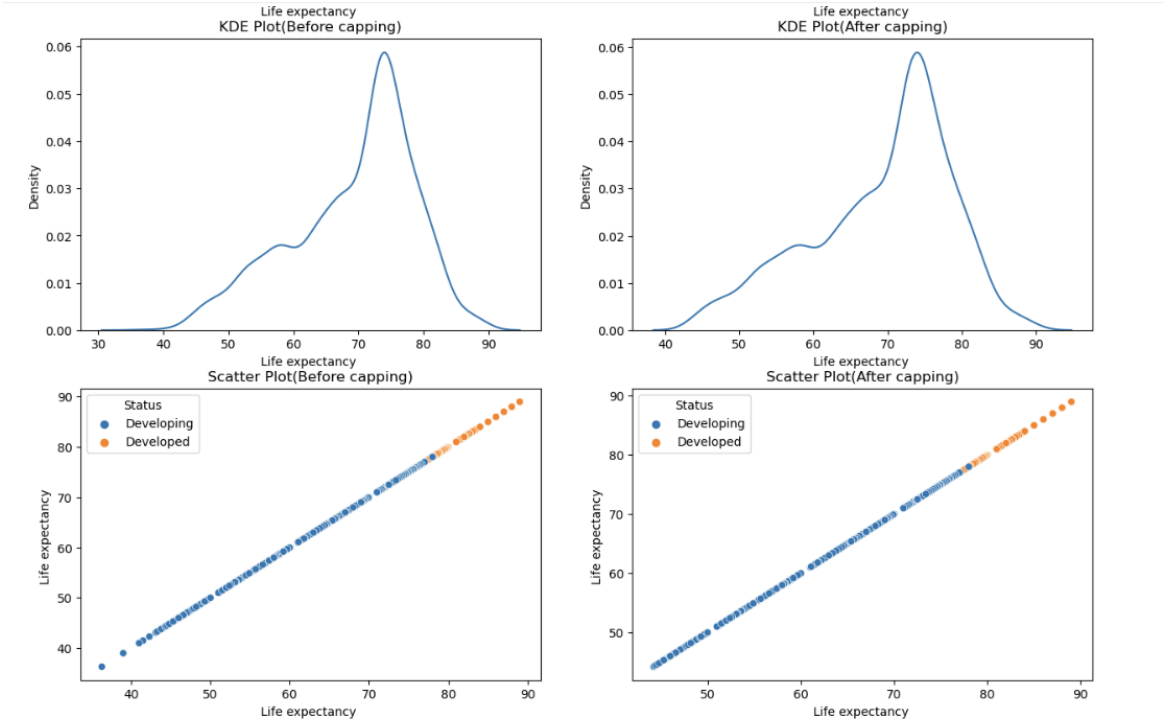
Outliers are extreme values that deviate from other observations on data, they may indicate a variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample.

We have found outliers in the following attributes:

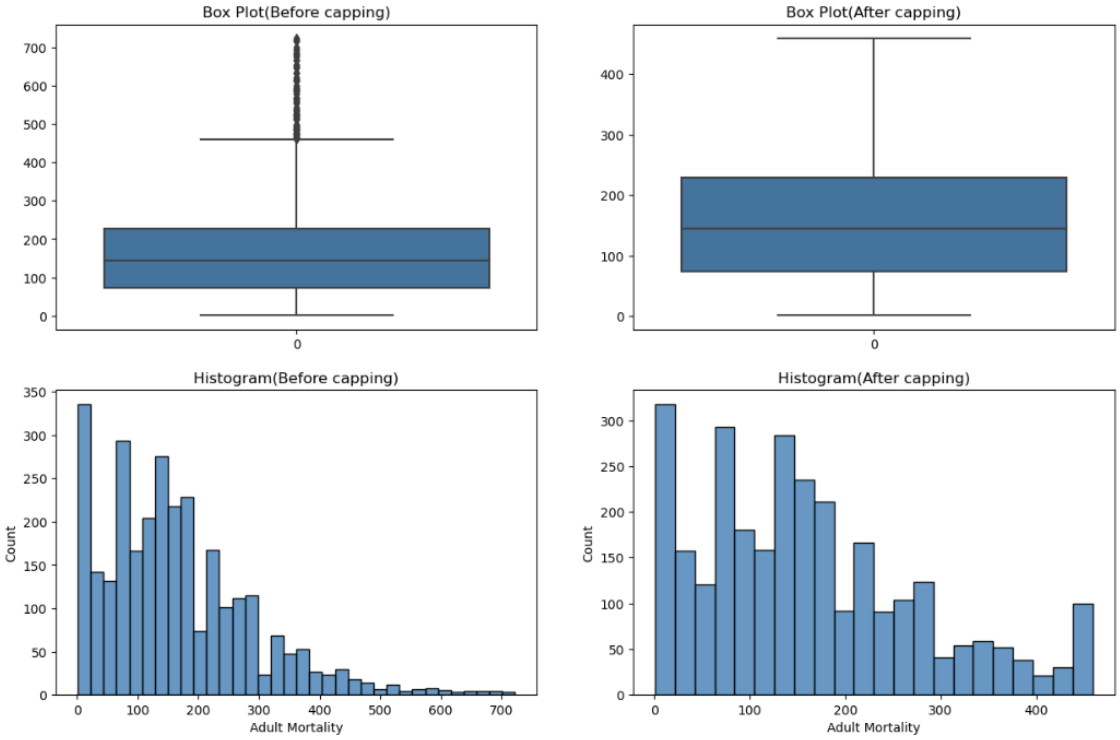
1. Life expectancy
2. Adult Mortality
3. infant deaths
4. percentage expenditure
5. Hepatitis B
6. Measles
7. under-five deaths
8. Polio
9. Total expenditure
10. Diphtheria
11. HIV/AIDS
12. GDP
13. Population
14. thinness 1-19 years
15. thinness 5-9 years
16. Income composition of resources
17. Schooling



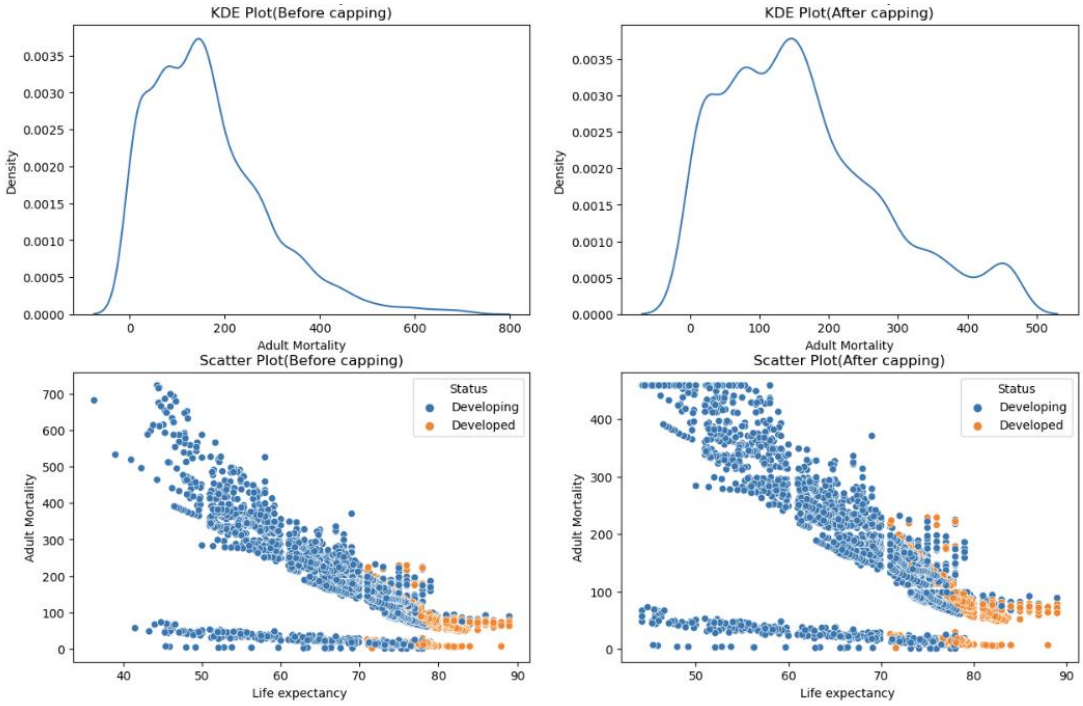
Boxplot and Histogram plot for Life expectancy



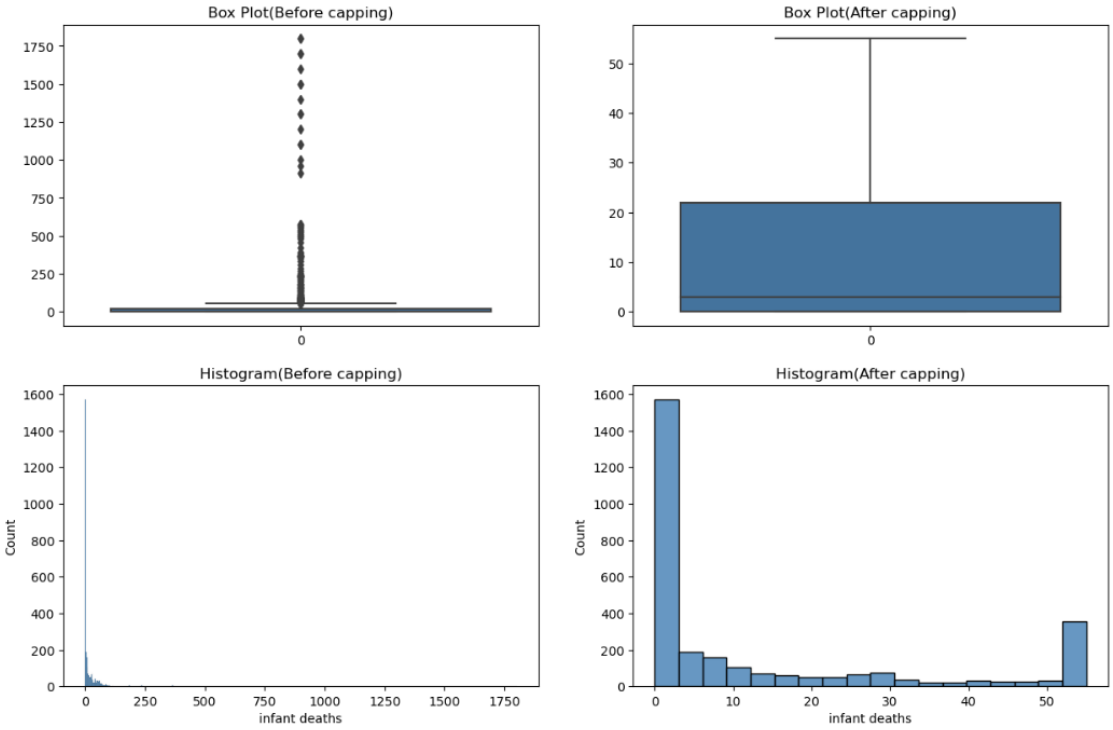
KDE and Scatterplot for Life expectancy



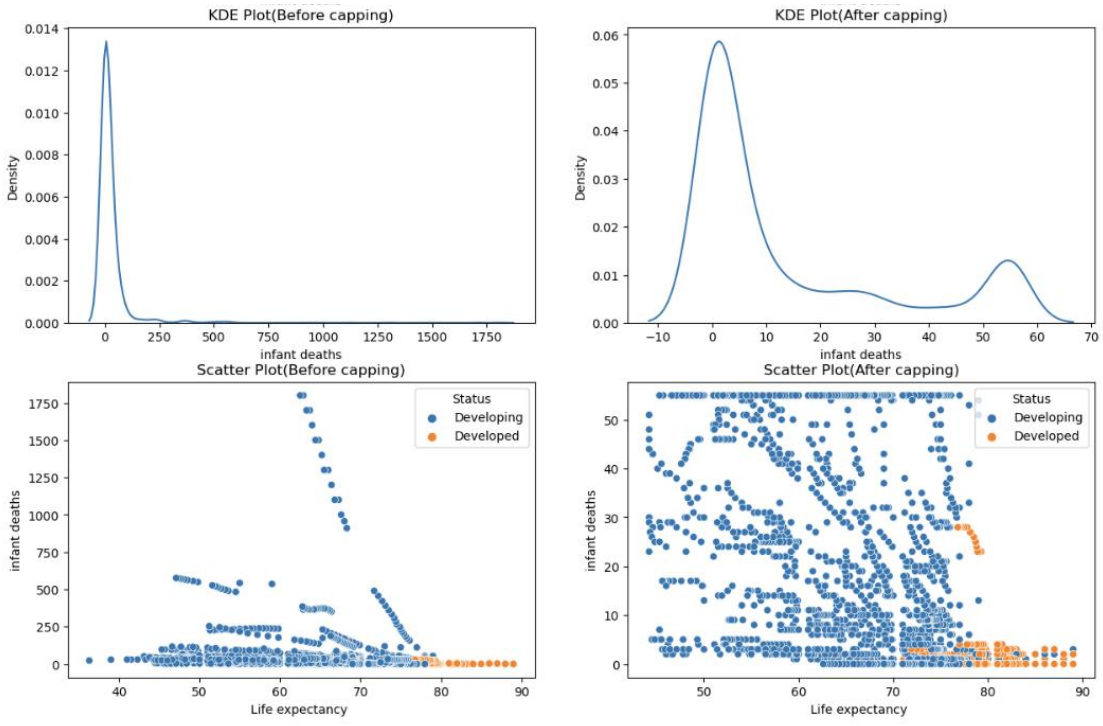
Boxplot and Histogram plot for Adult Mortality



KDE and Scatterplot for Adult Mortality

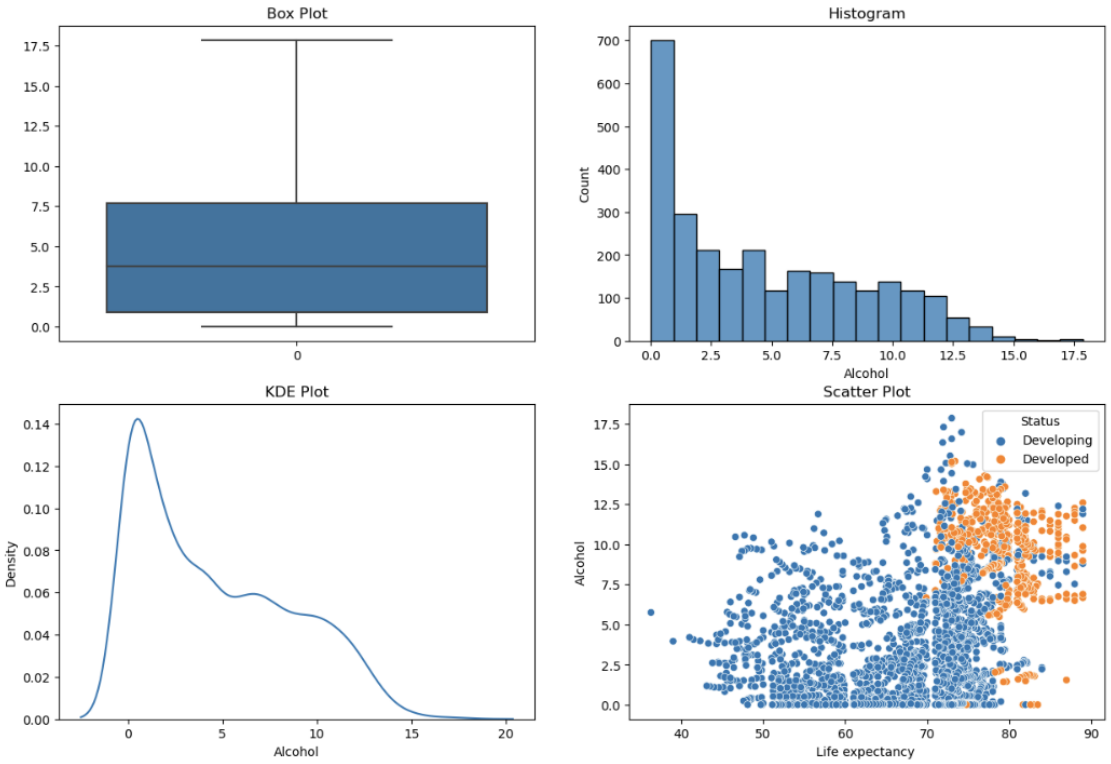


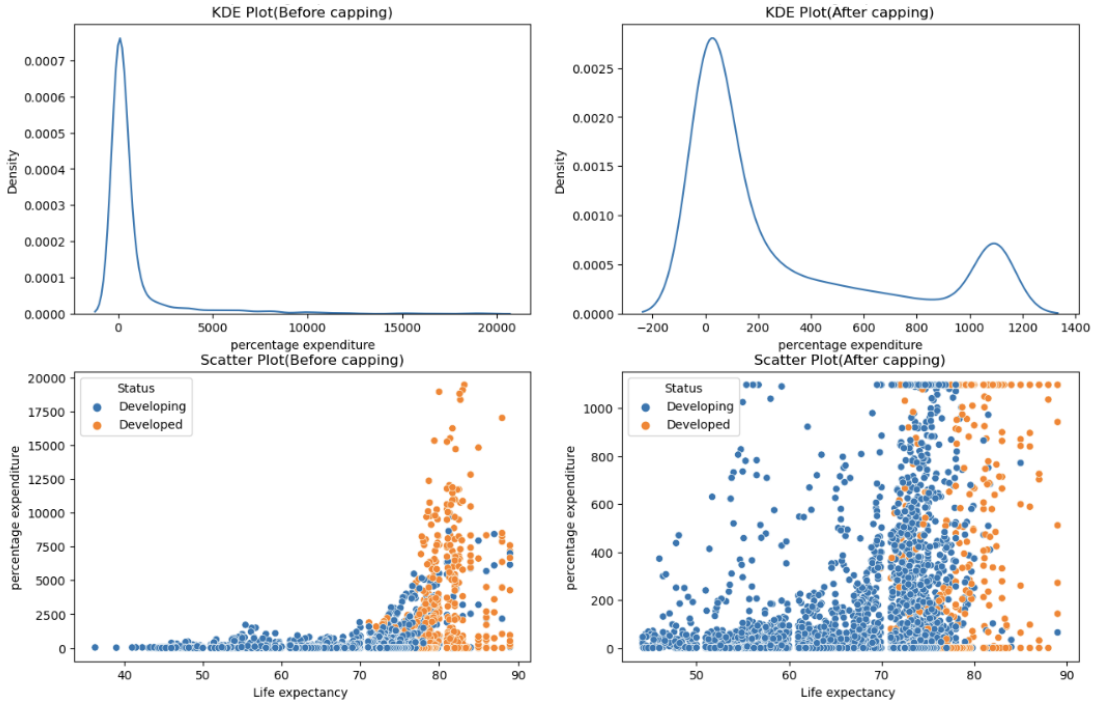
Boxplot and Histogram plot for Infant Deaths



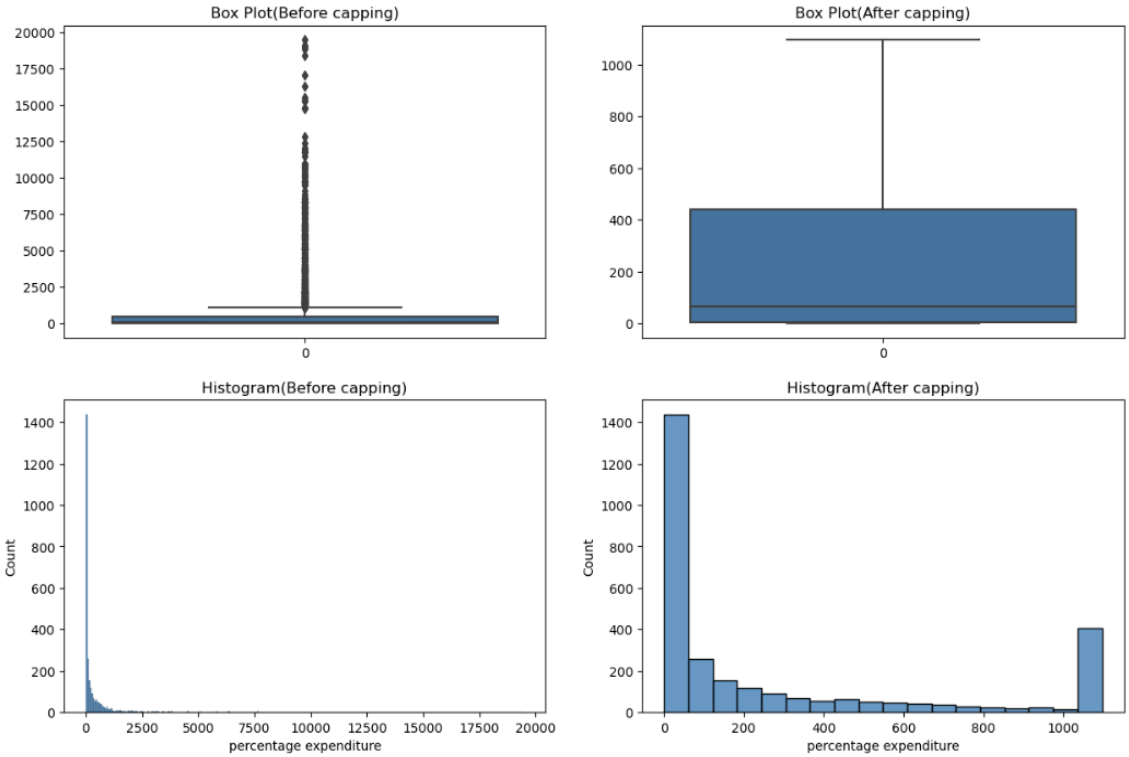
KDE and Scatterplot for Infant deaths

Boxplot, Histogram, KDE plot and Scatter plot for Alcohol

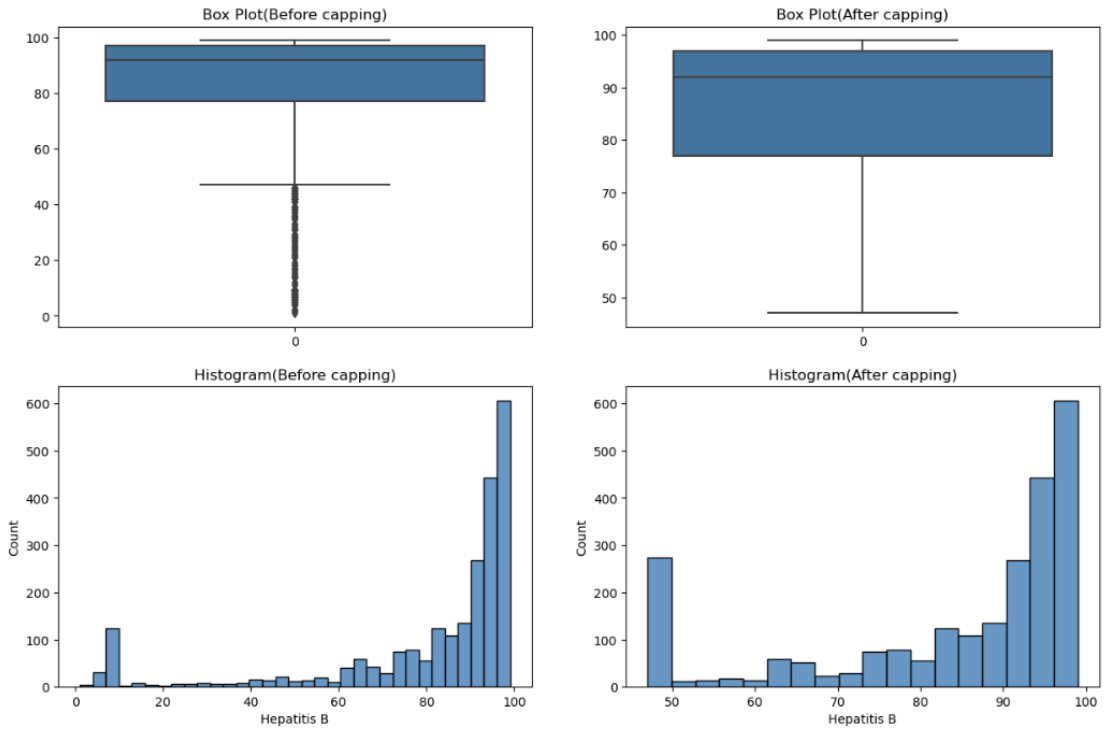




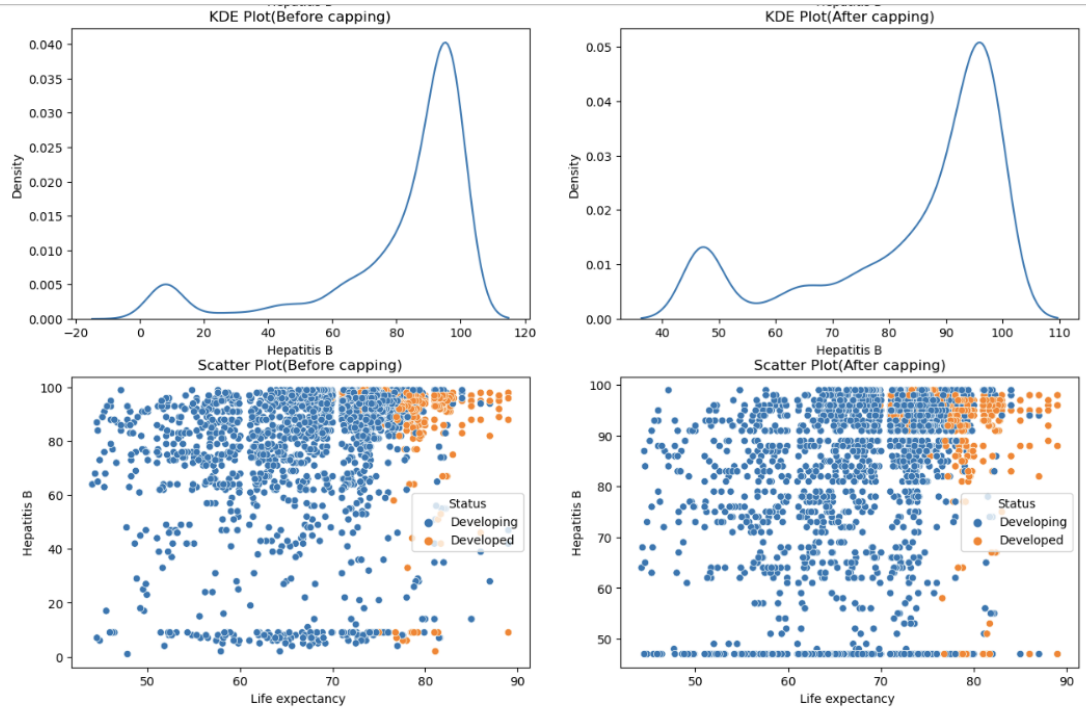
KDE and Scatter plot for Percentage Expenditure



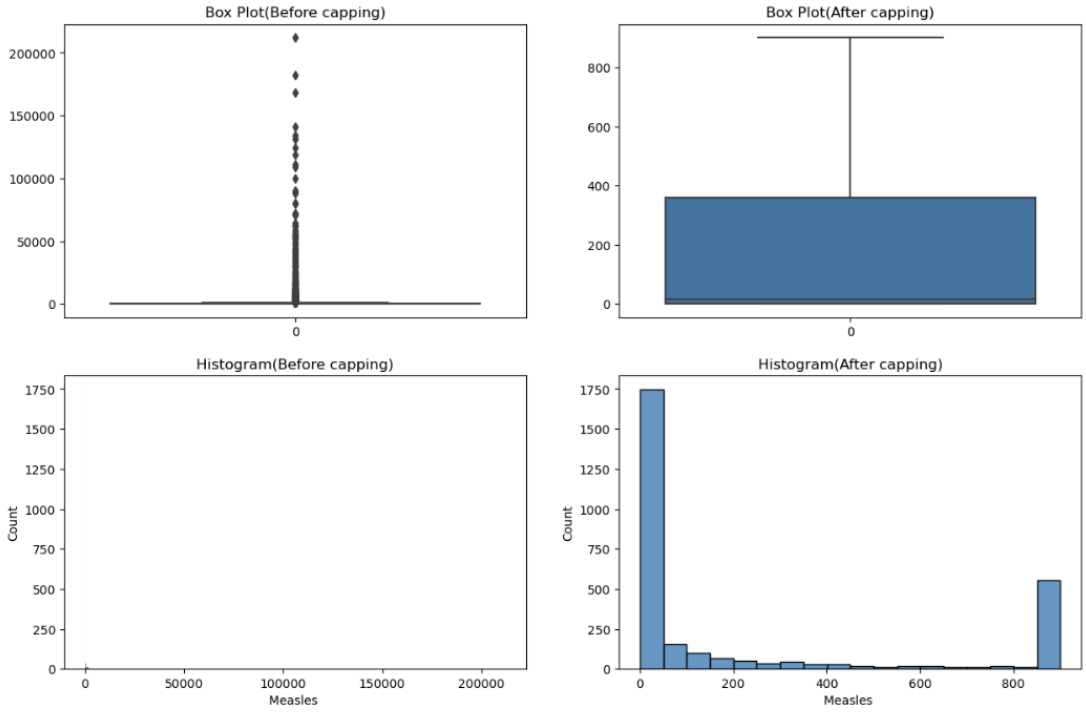
Boxplot and Histogram for Percentage Expenditure



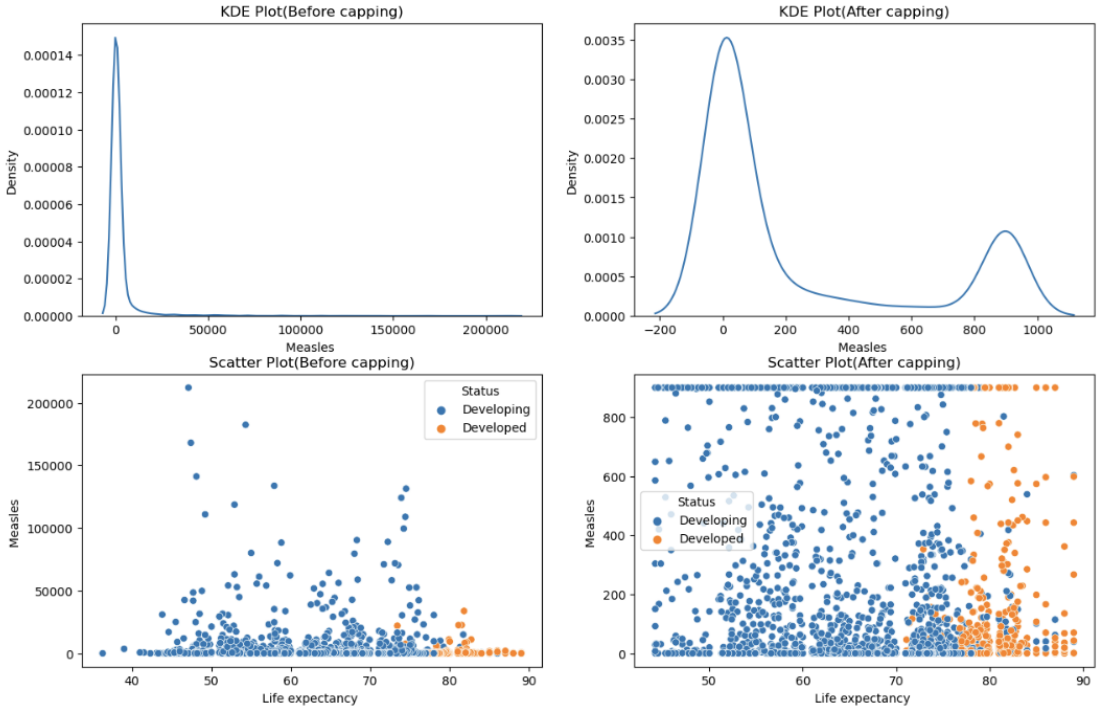
Boxplot and Histogram plot for Hepatitis-B



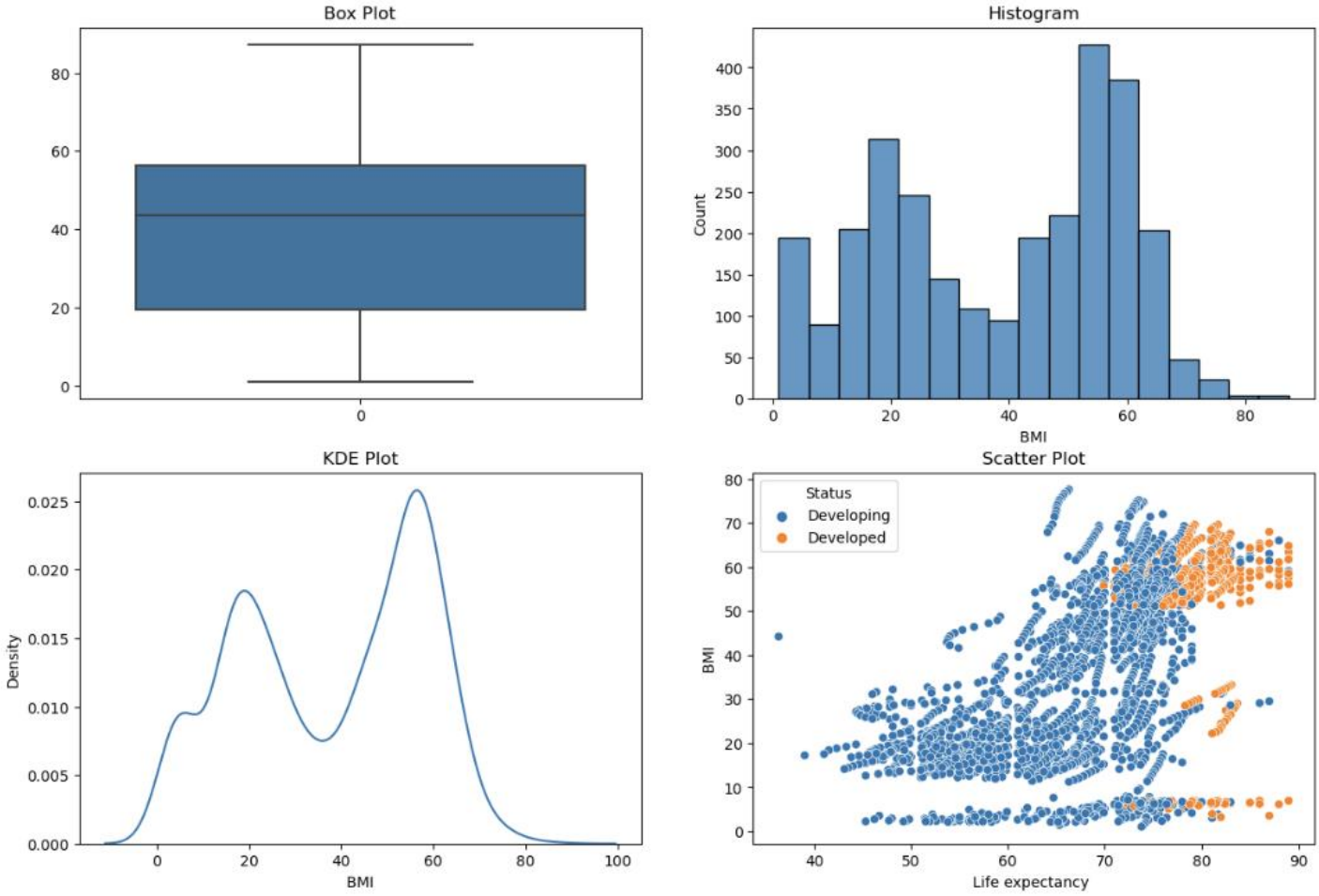
KDE and Scatterplot for Hepatitis-B



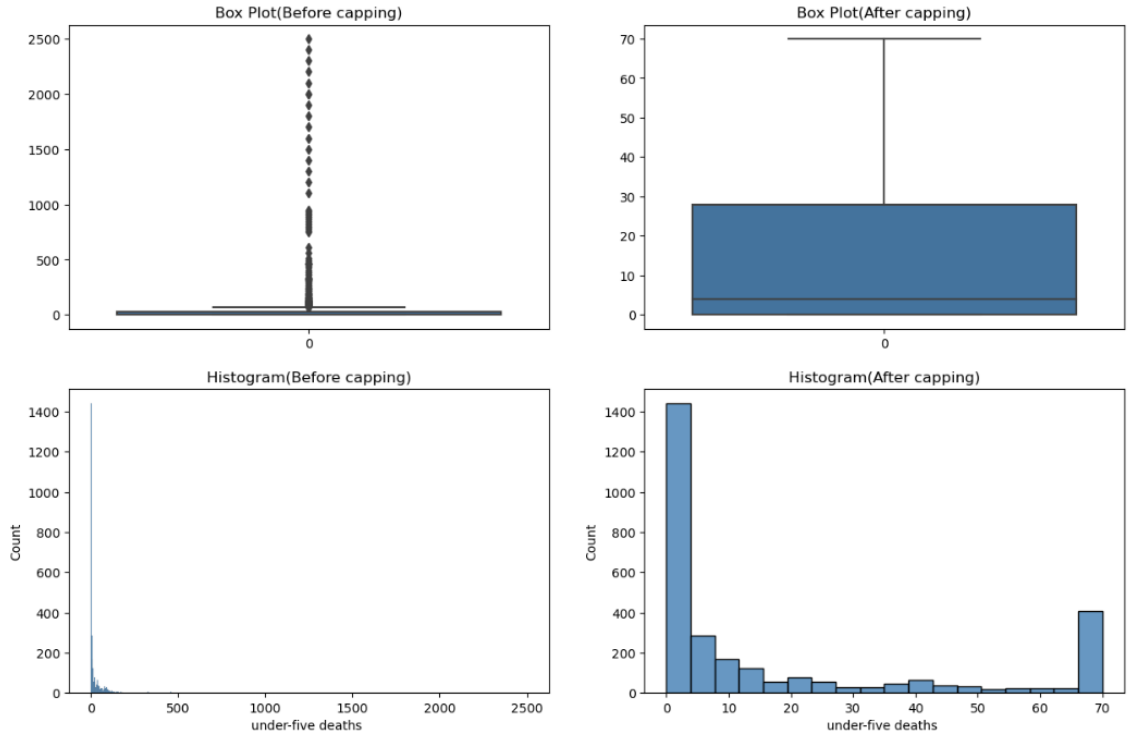
Boxplot and Histogram plot for Measles



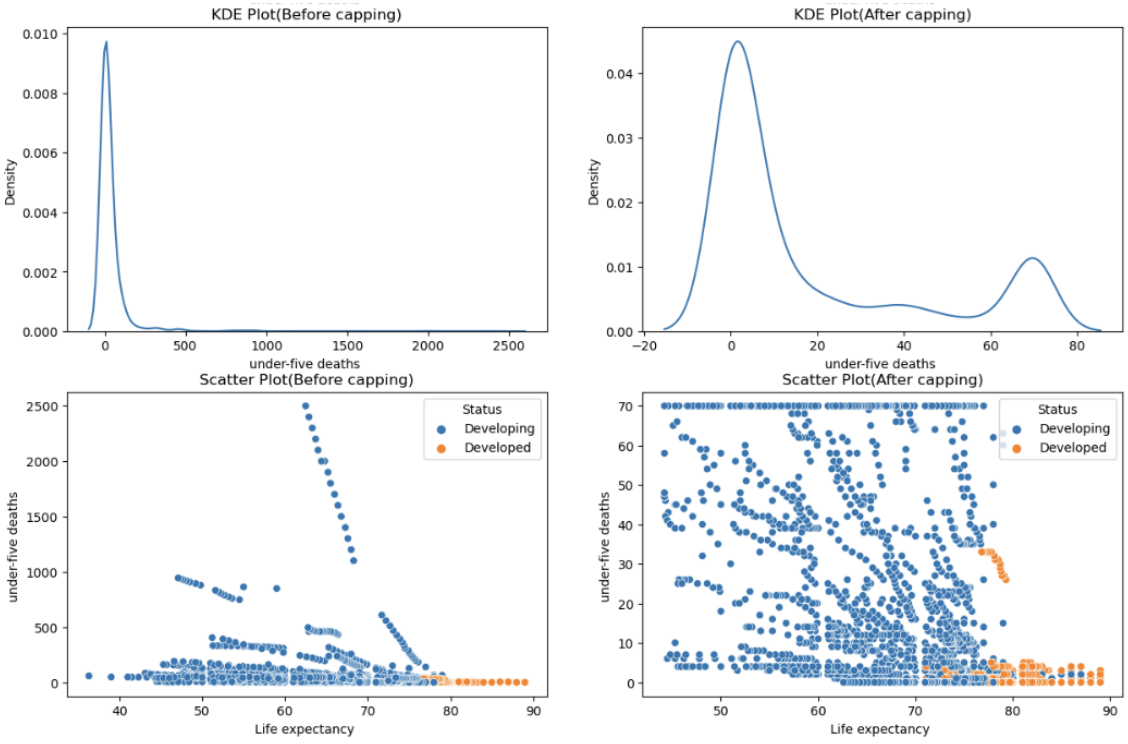
KDE and Scatterplot for Measles



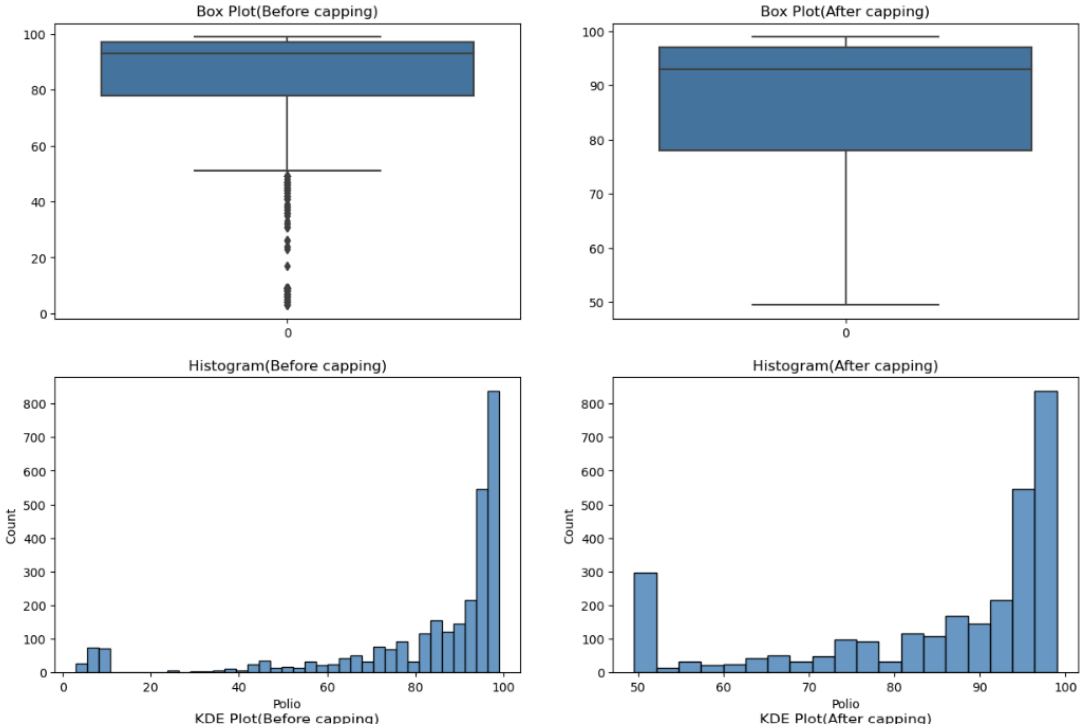
Boxplot and Histogram plot for BMI



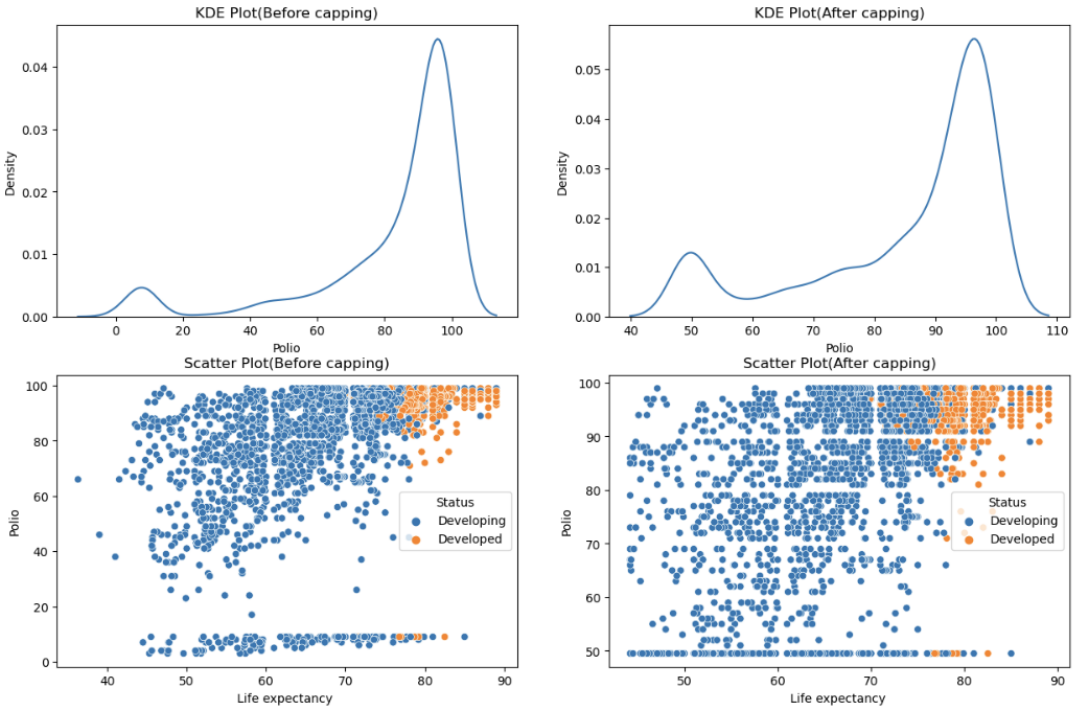
Boxplot and Histogram plot for Under-five-Death



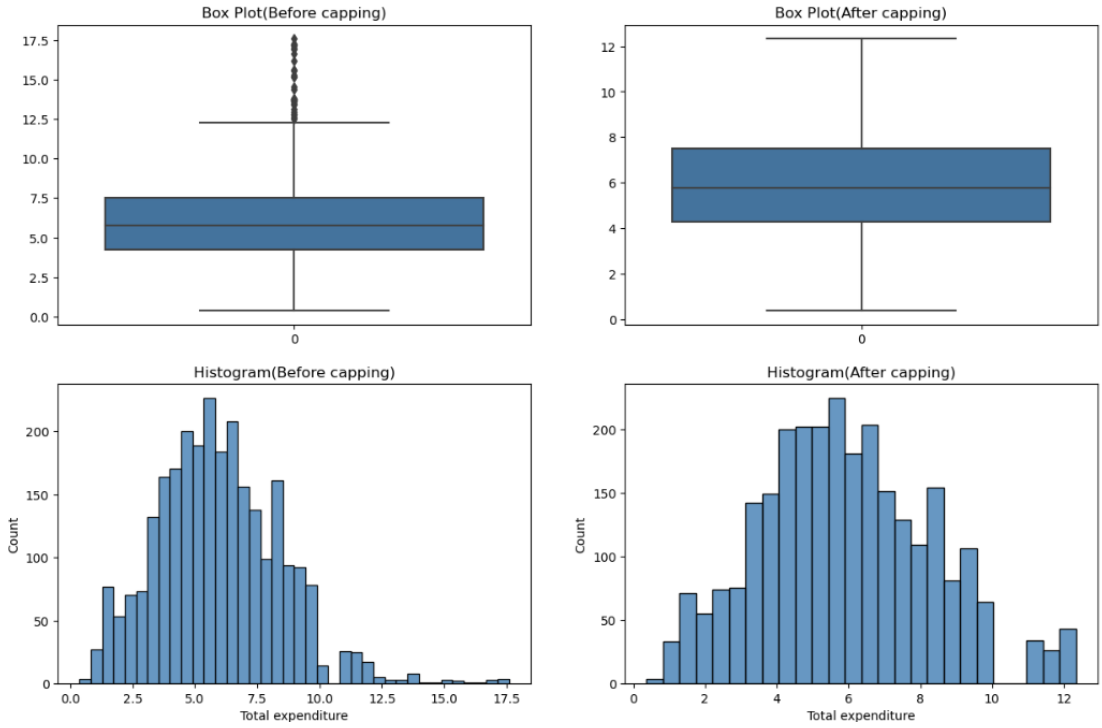
KDE and Scatterplot for Under-five-deaths



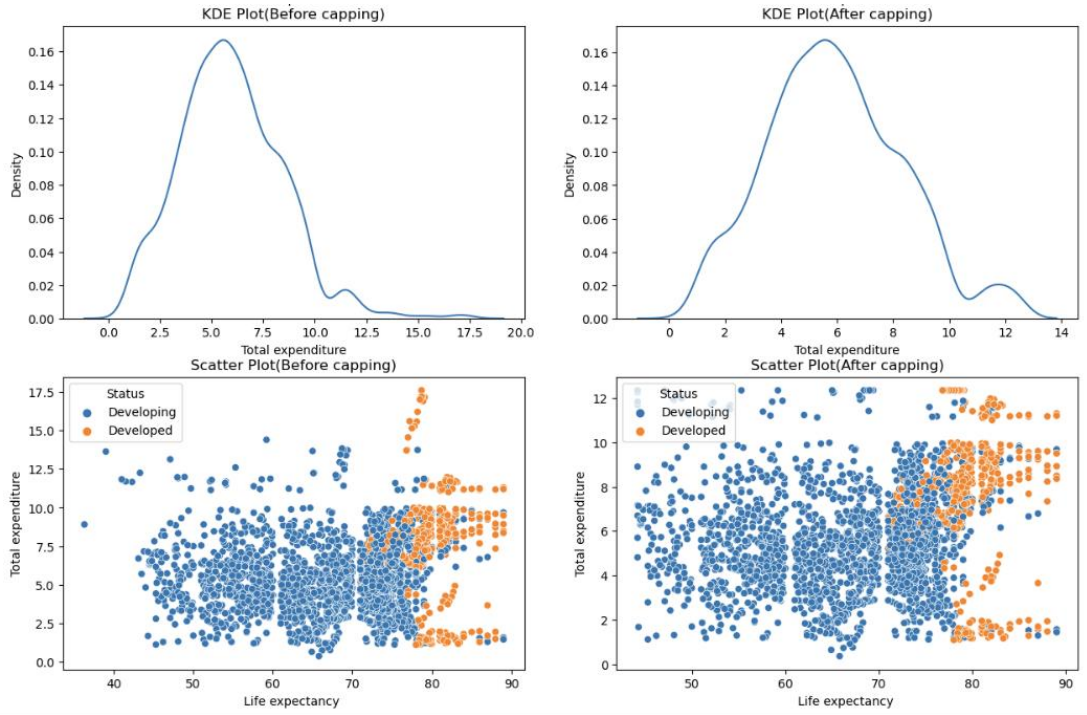
Boxplot and Histogram plot for Polio



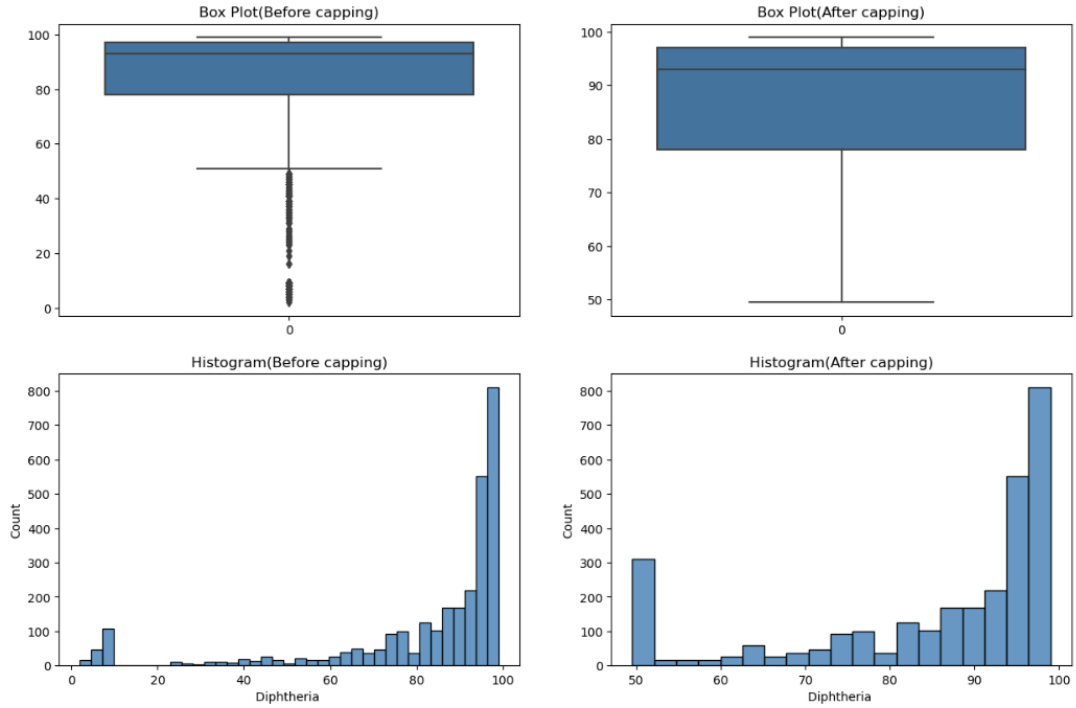
KDE and Scatterplot for Polio



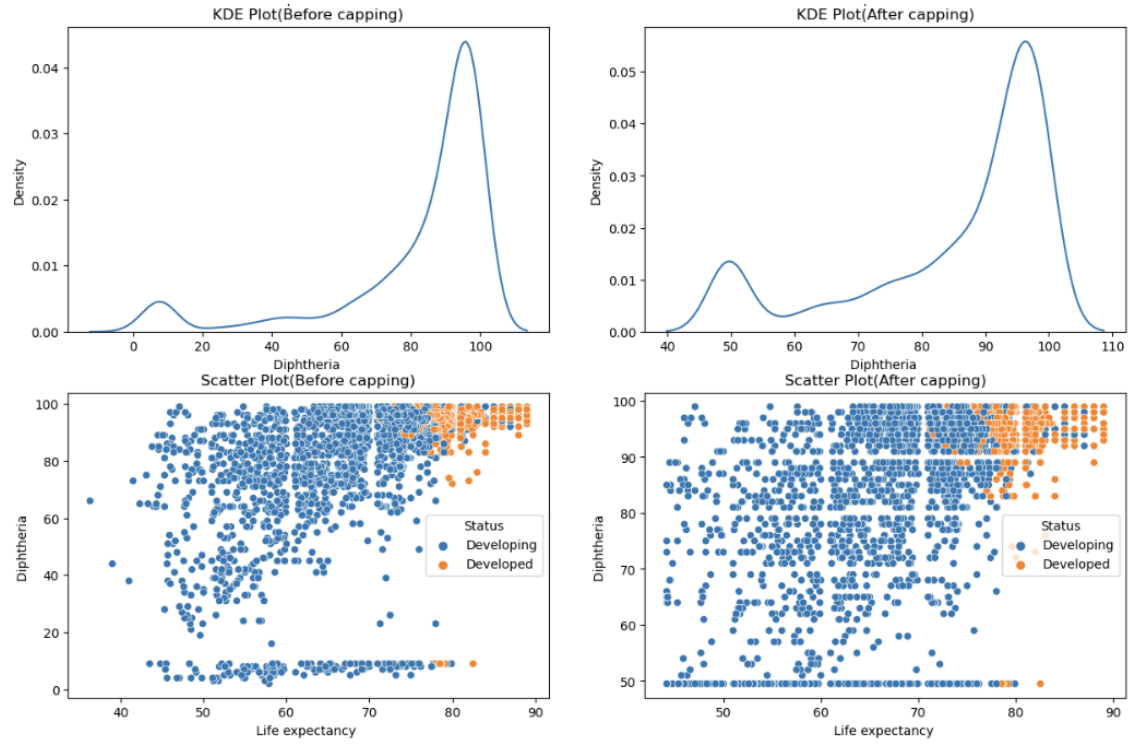
Boxplot and Histogram plot for Total expenditure



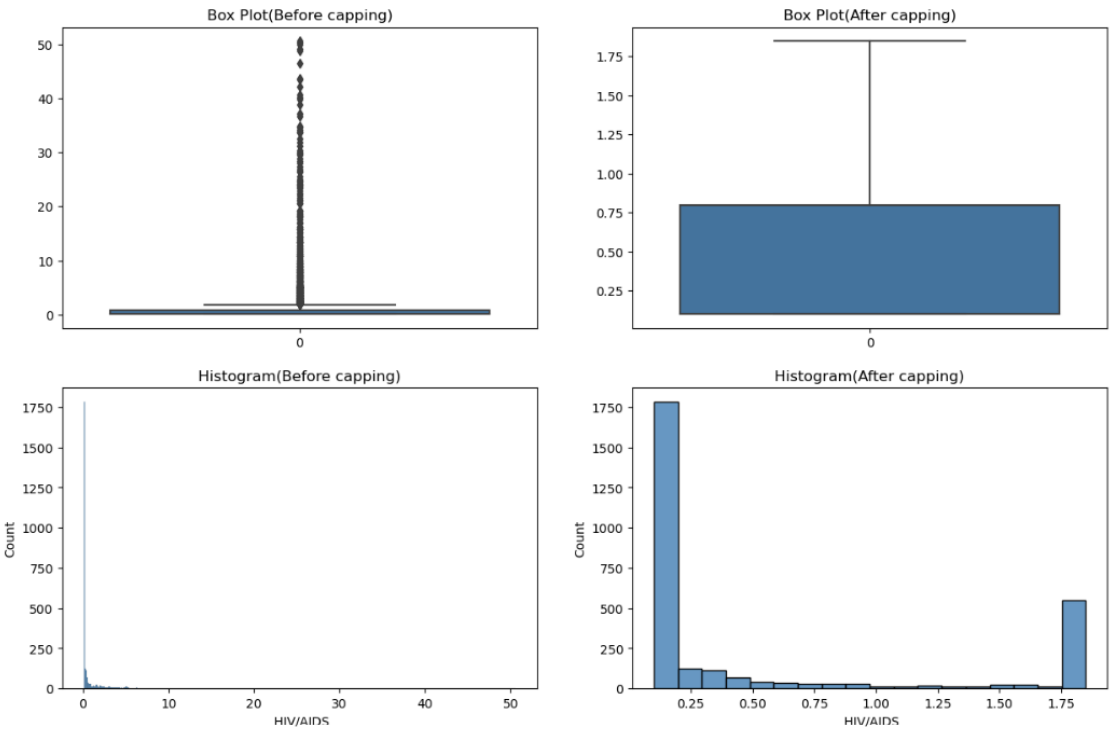
KDE and Scatterplot for Total expenditure



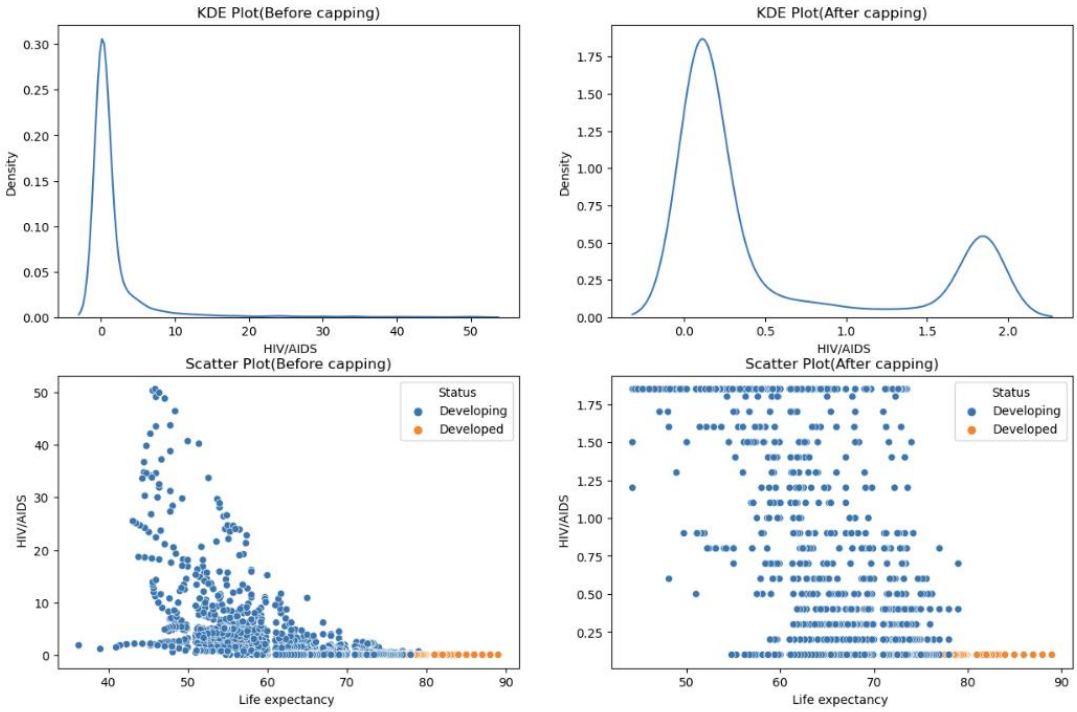
Boxplot and Histogram plot for Diphtheria



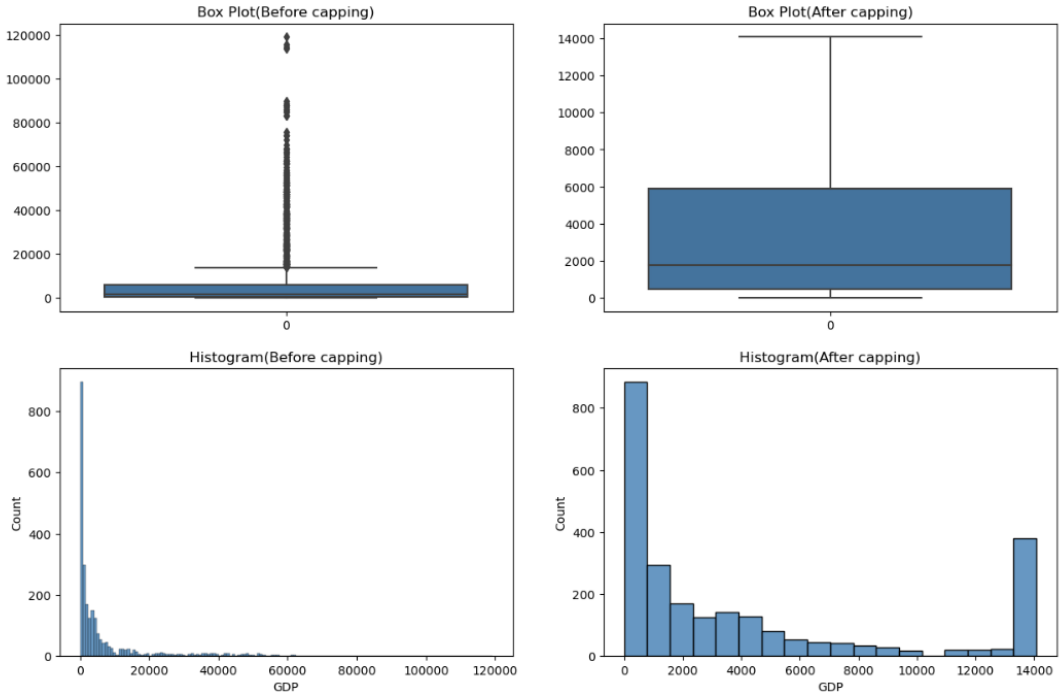
KDE and Scatterplot for Diphtheria



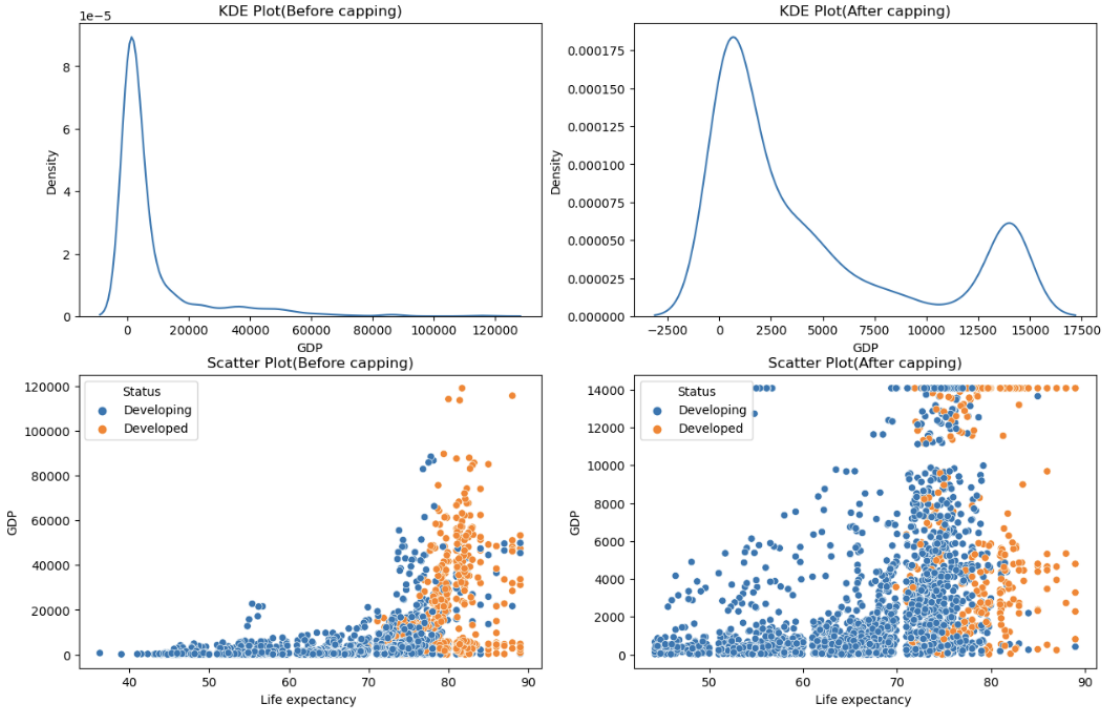
Boxplot and Histogram plot for HIV/AIDS



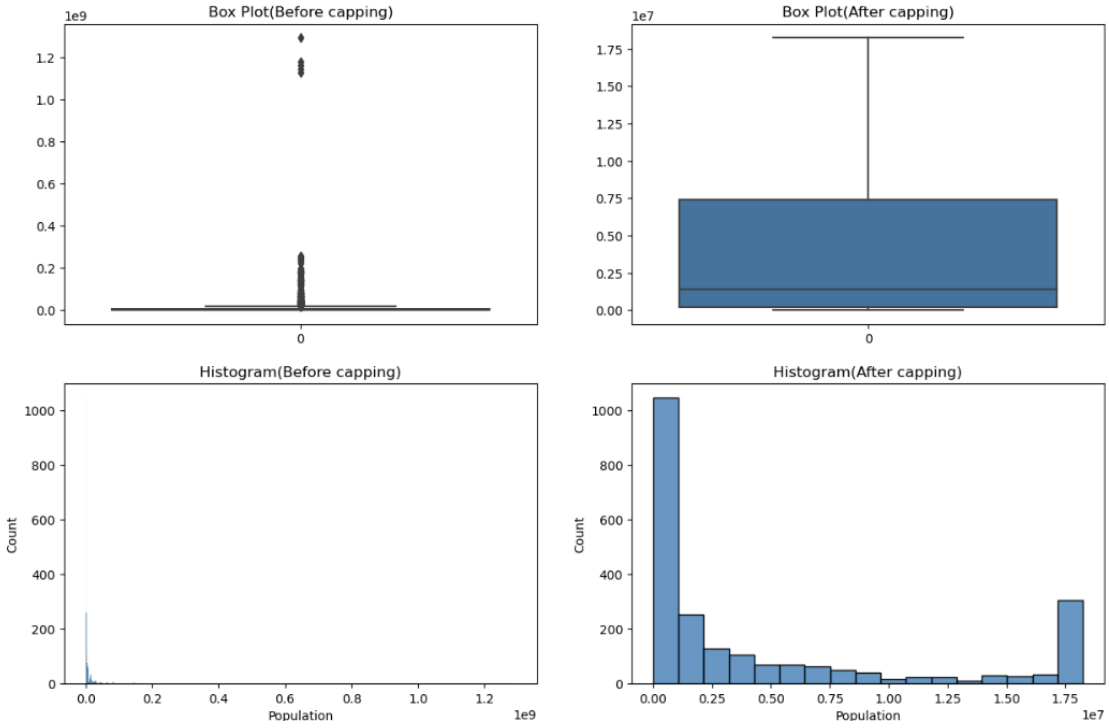
KDE and Scatterplot for HIV/AIDS



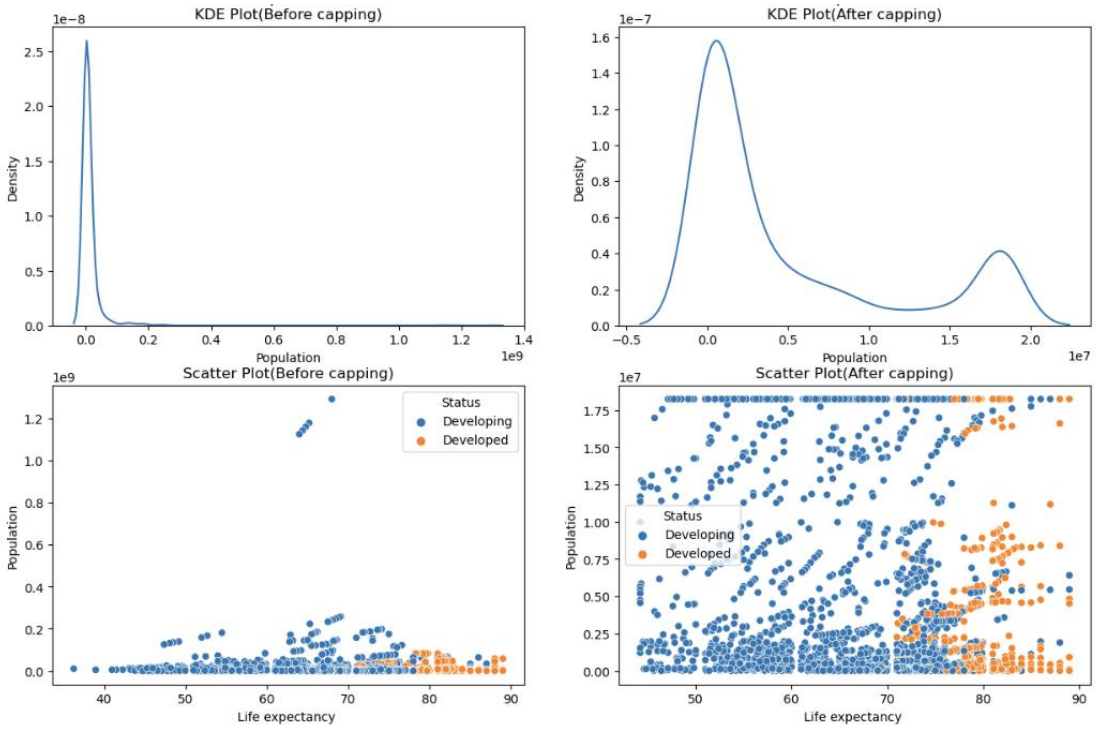
Boxplot and Histogram plot for GDP



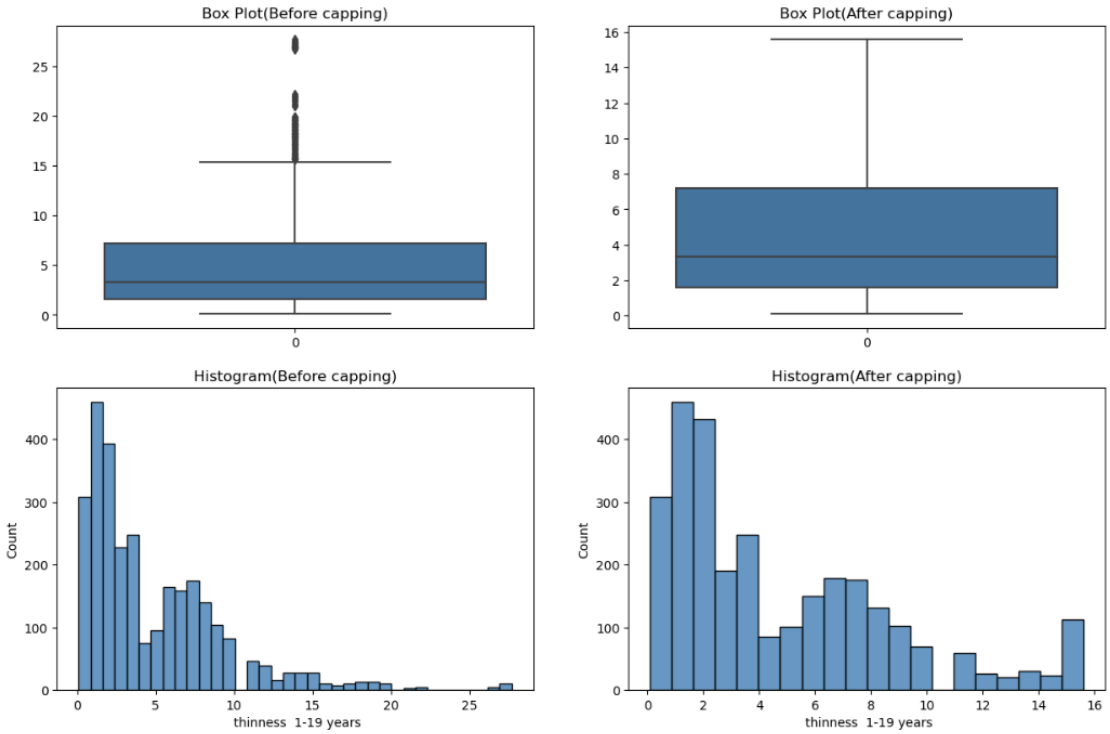
KDE and Scatterplot for GDP



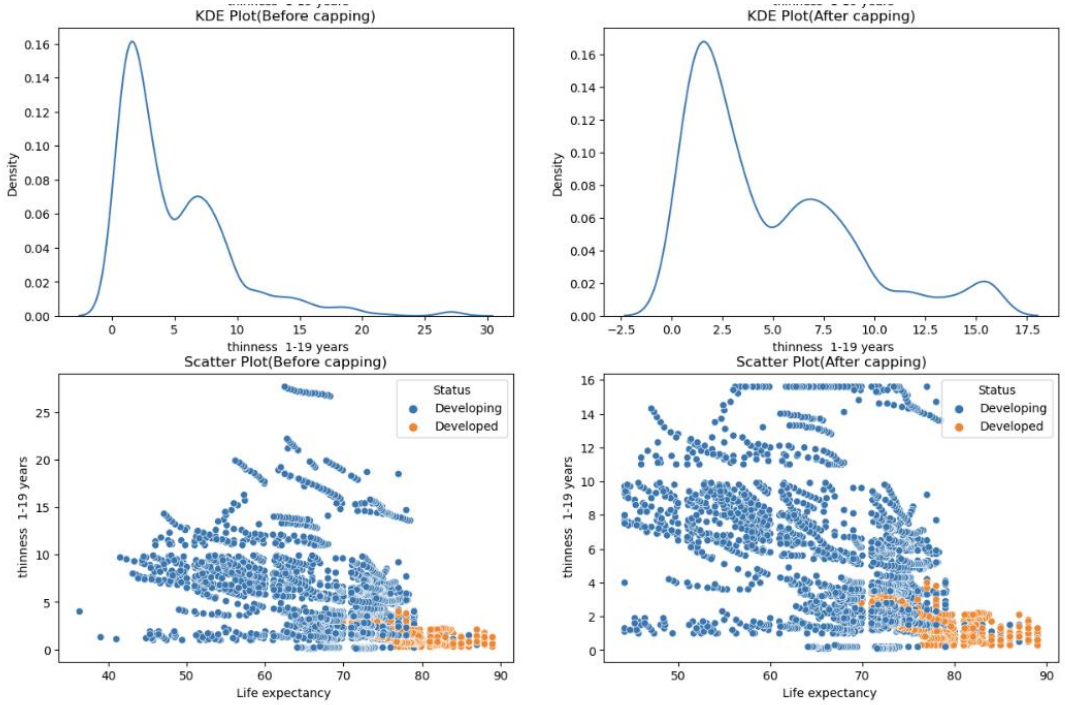
Boxplot and Histogram plot for Population



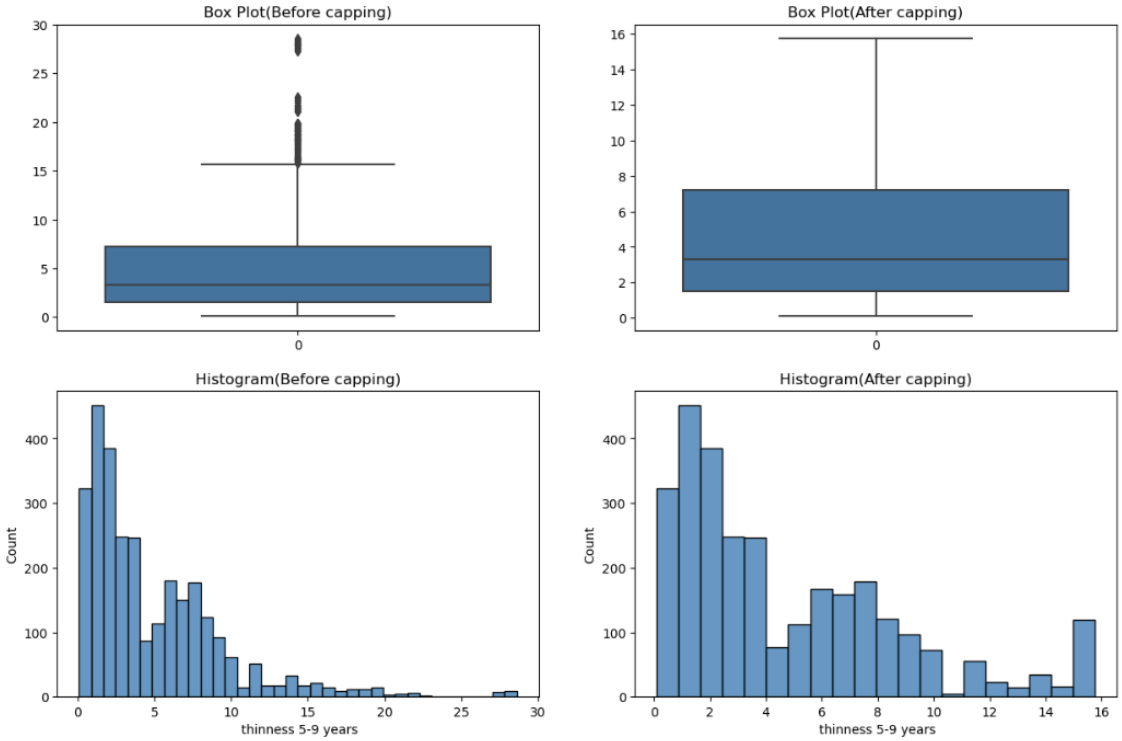
KDE and Scatterplot for Population



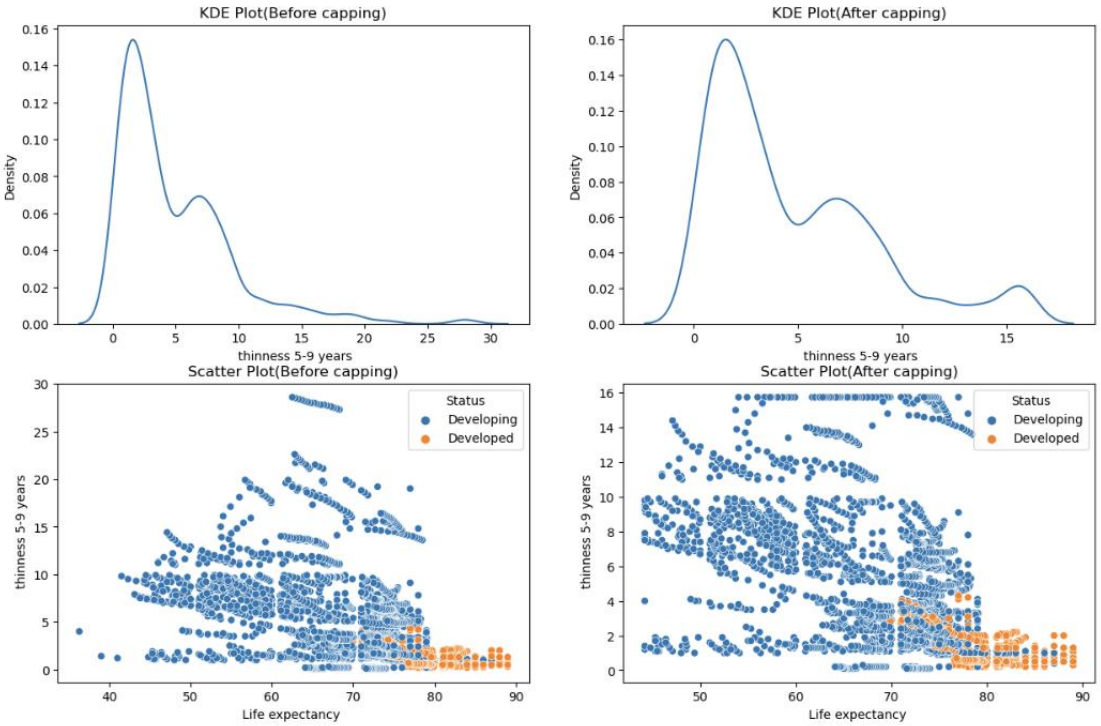
Boxplot and Histogram plot for thinness 1-19 years



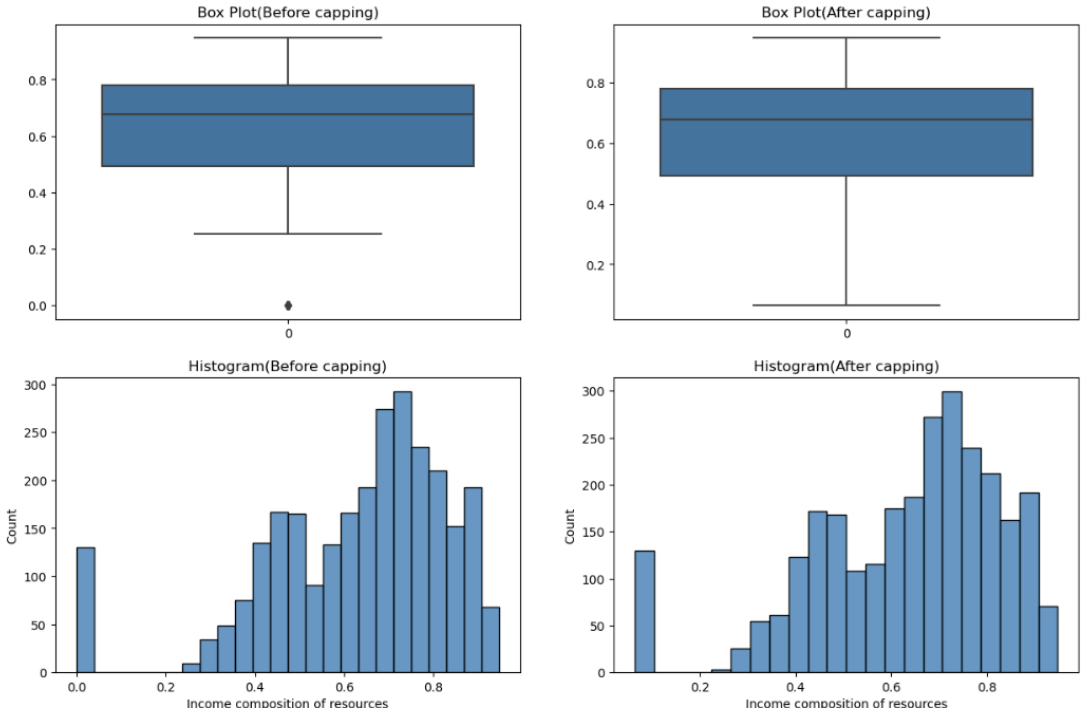
KDE and Scatterplot for thinnes 1-19 years



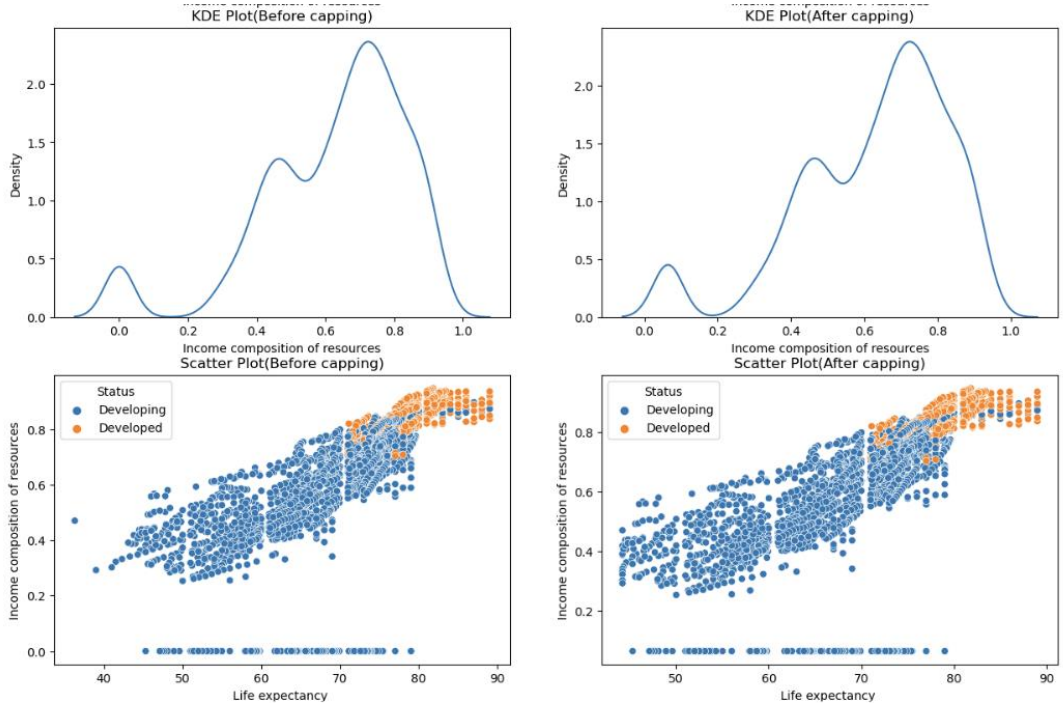
Boxplot and Histogram plot for thinness 5-9 years



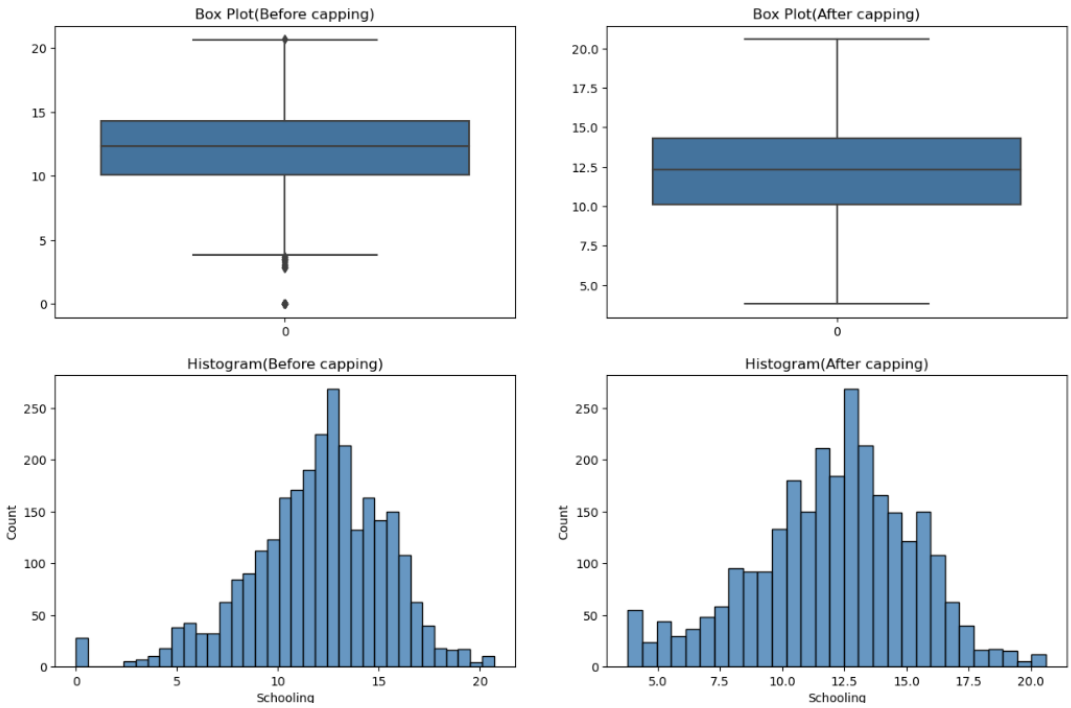
KDE and Scatterplot for thinness 5-9 years



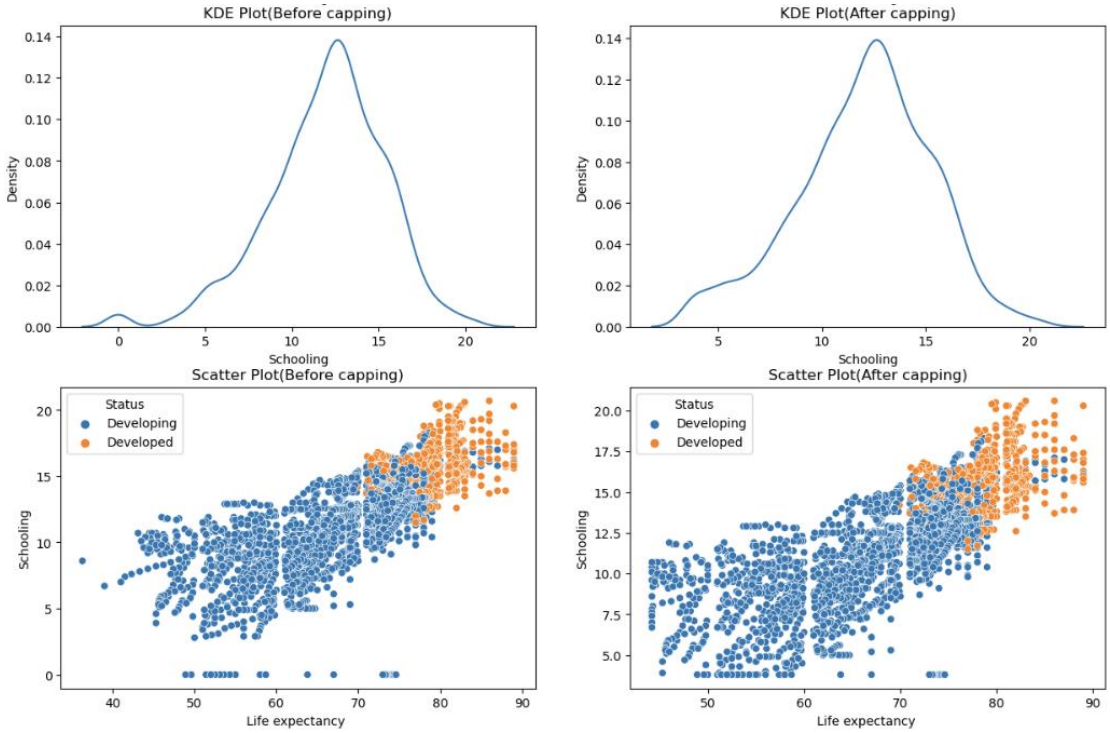
Boxplot and Histogram plot for Income composition of resource



KDE and Scatterplot for Income composition of resource

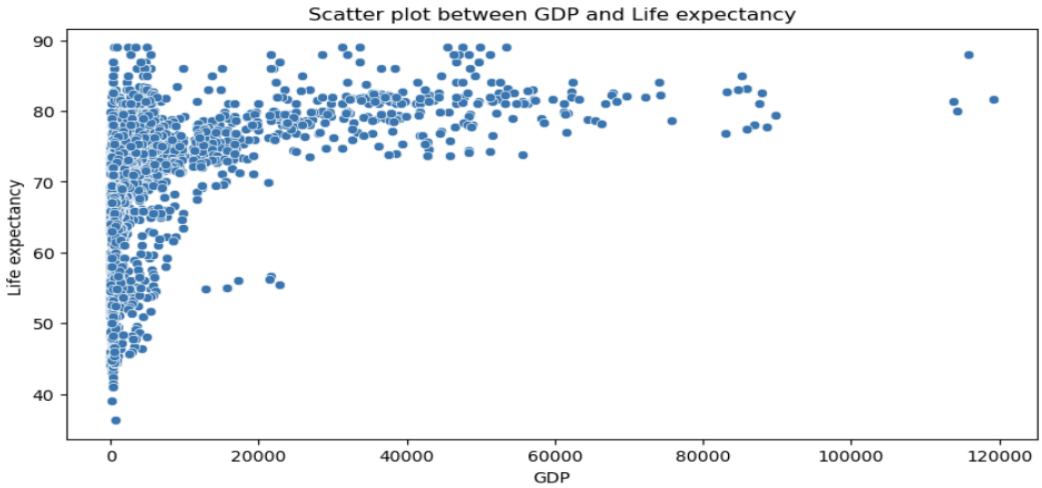


Boxplot and Histogram plot for schooling

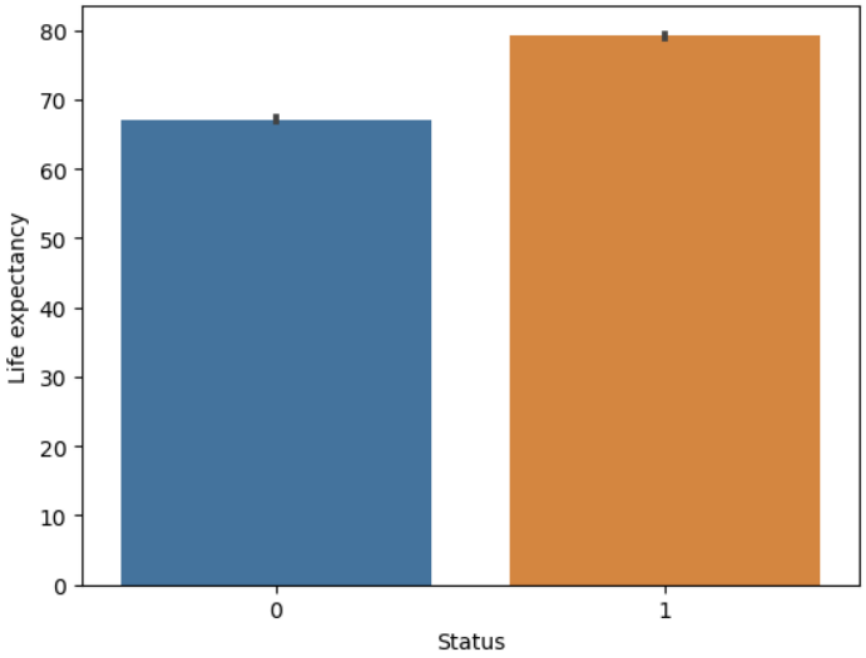


KDE and Scatterplot for schooling

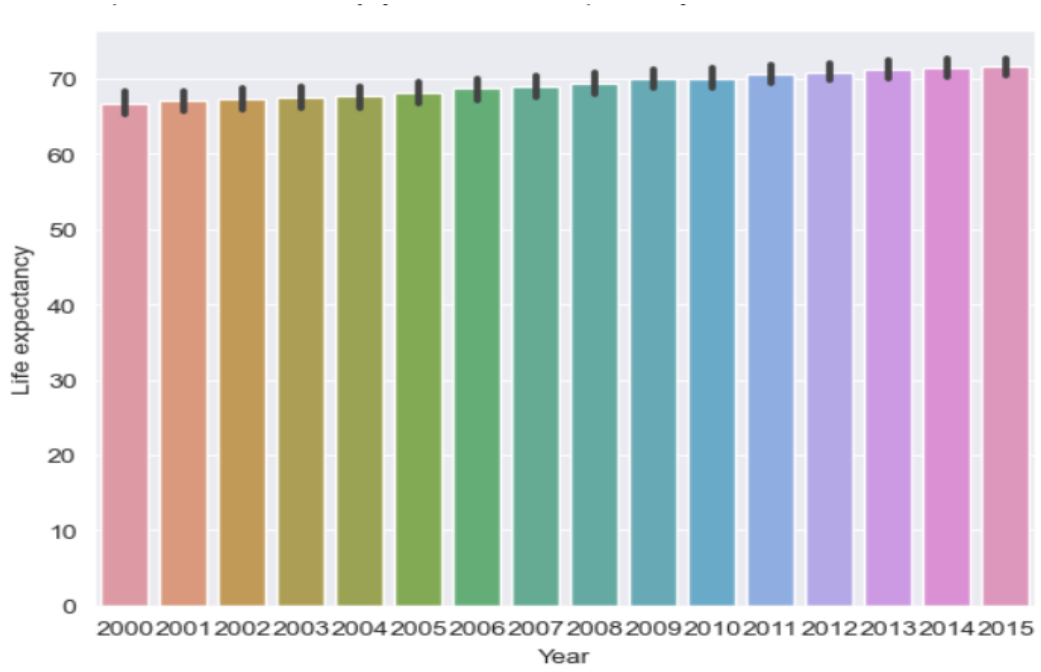
We have handled the outliers by using IQR method. After handling the outliers we have visualized other attributes with respect to our target column i.e. Life Expectancy.



Bubble plot for GDP with Life Expectancy

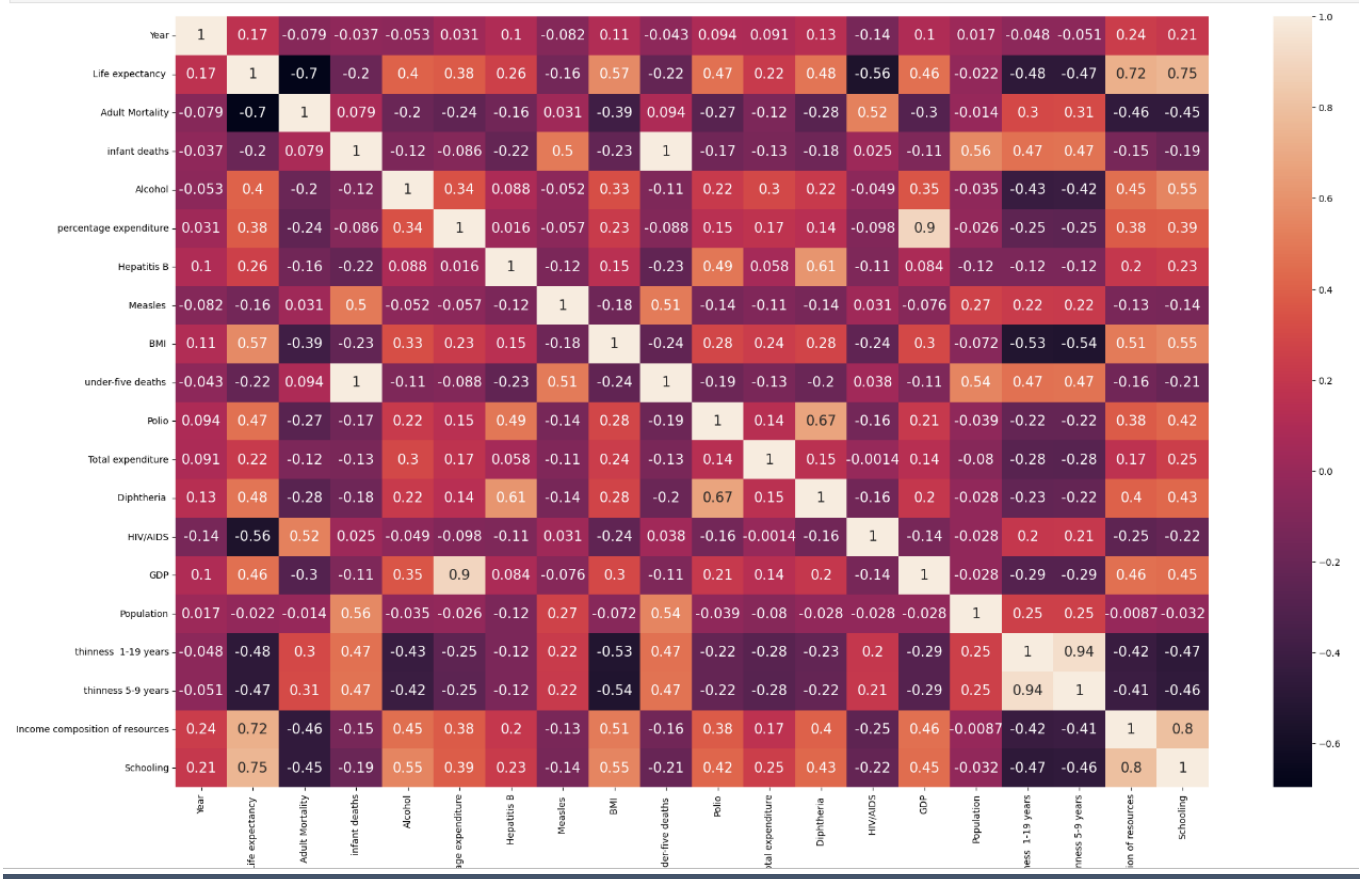


Bar plot for Status with Life Expectancy



Bar plot for Year with Life Expectancy

Determining the correlation between different columns with the help of heatmap



Model Building

Splitting data for training and testing purpose: We split the given train dataset into two parts for training and testing purpose. The split ratio we used is 0.75 which indicates we used 70% data for training purpose and 30% data for testing purpose. We will be using the same split ratio for all the models trained.

We have calculated the correlation coefficients for each column with respect to the target attribute to determine our training and testing sets.

Attribute	Correlation with target attribute
Country	-0.016763
Year	0.170033
Status	0.482136
Life expectancy	1.000000
Adult Mortality	-0.696359
infant deaths	-0.196557
Alcohol	0.404877
percentage expenditure	0.381864
Hepatitis B	0.256762
Measles	-0.157586
BMI	0.567694
under-five deaths	-0.222529
Polio	0.465556
Total expenditure	0.218086
Diphtheria	0.479495
HIV/AIDS	-0.556556
GDP	0.461455
Population	-0.021538
thinness 1-19 years	-0.477183
thinness 5-9 years	-0.471584
Income composition of resources	0.724776
Schooling	0.751975

Now we will be training our required 3 models which are Linear regression, Random Forest regressor model and Decision Tree regressor model. We will be using the final dataset after pre-processing it. We have done multiple attempts to achieve desired accuracy for the solution.

1.Linear Regression

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price**,** etc. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables.

Mathematically, we can represent a linear regression as:

$$y= a_0+a_1x+ \varepsilon$$

Y= Dependent Variable (Target Variable)

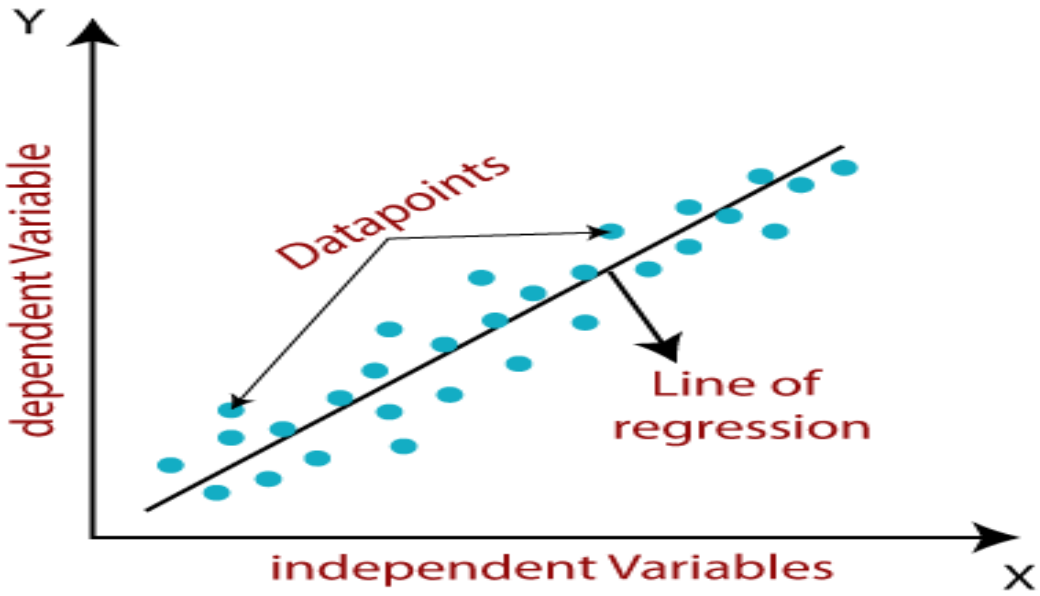
X= Independent Variable (predictor Variable)

a0= intercept of the line (Gives an additional degree of freedom)

a1 = Linear regression coefficient (scale factor to each input value).

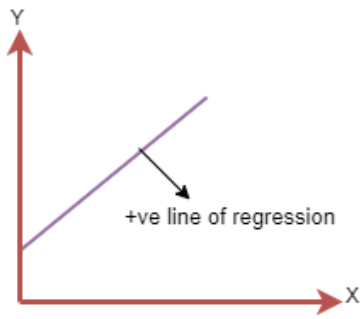
ε = random error

The values for x and y variables are training datasets for Linear Regression model representation.

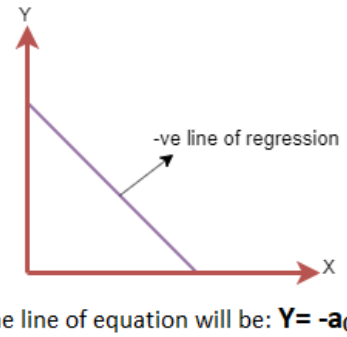


A linear line showing the relationship between the dependent and independent variables is called a **regression line**. When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

- **Positive Linear Relationship:**
  If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.
- **Negative Linear Relationship:** If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.



The line equation will be: $Y= a_0+a_1x$

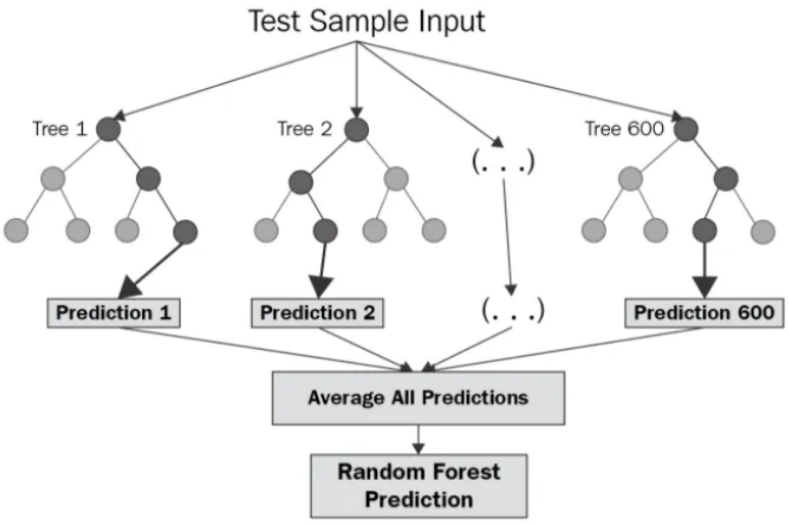


The line of equation will be: $Y= -a_0+a_1x$

2.Random Forest Regressor Model

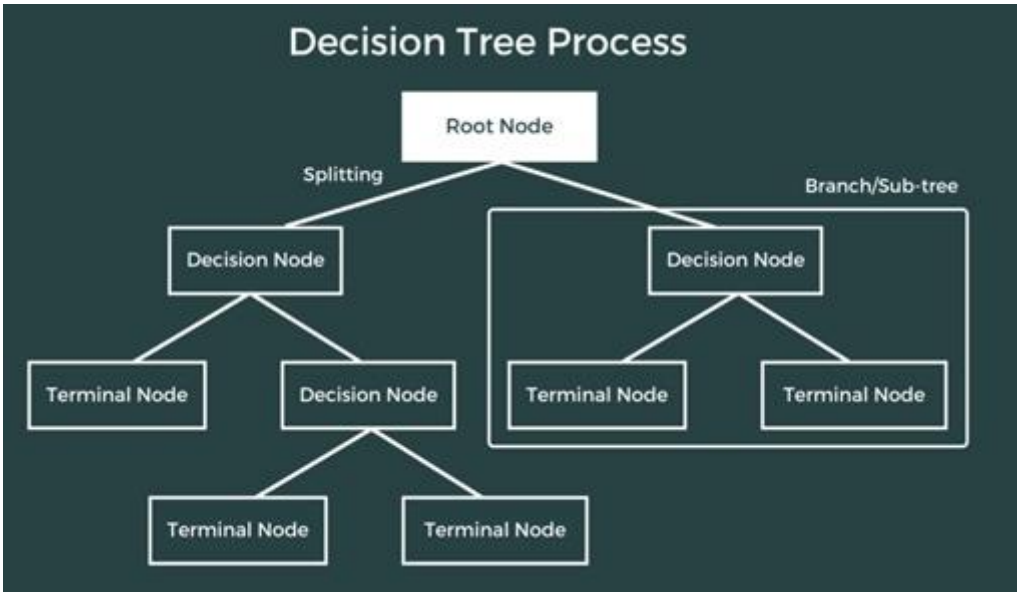
Random Forest Regression is a supervised learning algorithm that uses **ensemble learning** method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees. To get a better understanding of the Random Forest algorithm, let's walk through the steps:

1. Pick at random *k* data points from the training set.
2. Build a decision tree associated to these *k* data points.
3. Choose the number *N* of trees you want to build and repeat steps 1 and 2.
4. For a new data point, make each one of your *N*-tree trees predict the value of *y* for the data point in question and assign the new data point to the average across all of the predicted *y* values.



3.Decision Tree Regressor Model

Decision Tree Regressor observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. It works by splitting the data up in a tree-like pattern into smaller and smaller subsets. Then, when predicting the output value of a set of features, it will predict the output based on the subset that the set of features falls into.



Accuracy Comparison of the trained models

We have trained 3 models by using 3 algorithms.

1.Linear regression

2.Random Forest Regressor

3. Decision Tree Regressor

We have checked the accuracy by determining the R2(R-squared) score. R-squared is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

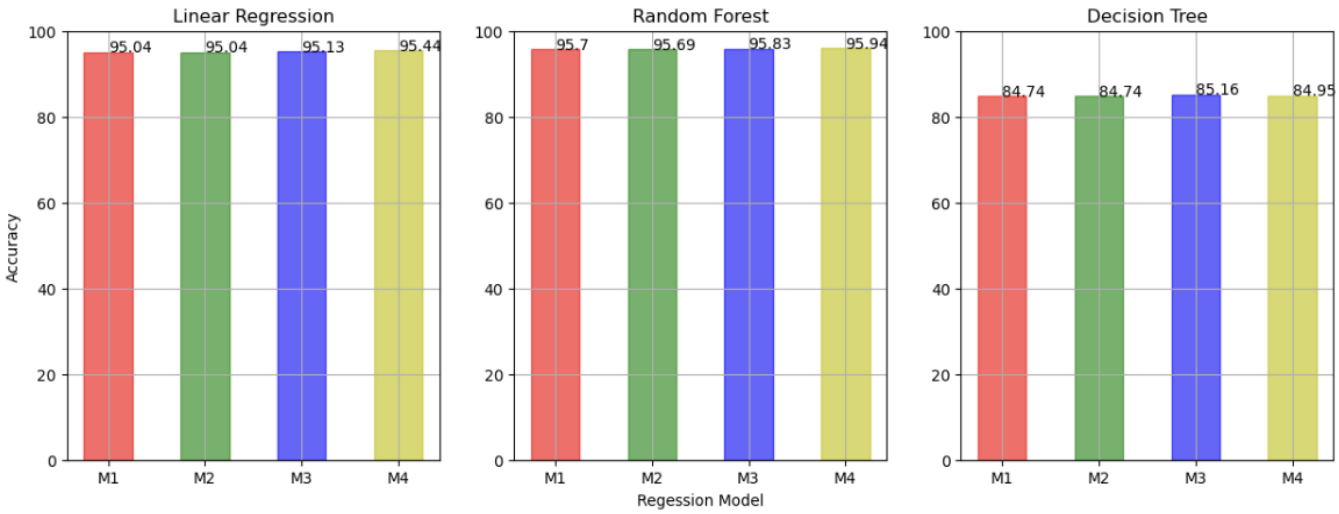
We have done 4 attempts for each of the models

1. With outliers without Standardization
2. With Outliers With Standardization
3. Without Outliers Without Standardization
4. Without Outliers With Standardization

Classifier Model Trained	Data Used	Data Used Accuracy %
Linear Regression	With outliers without Standardization	95.04
Linear Regression	With Outliers With Standardization	95.04
Linear Regression	Without Outliers Without Standardization	95.13
Linear Regression	Without Outliers With Standardization	95.44
Random Forest	With outliers without Standardization	95.70
Random Forest	With Outliers With Standardization	95.69
Random Forest	Without Outliers Without Standardization	95.83
Random Forest	Without Outliers With Standardization	95.94
Decision Tree Regressor	With outliers without Standardization	84.74
Decision Tree Regressor	With Outliers With Standardization	84.74
Decision Tree Regressor	Without Outliers Without Standardization	85.16
Decision Tree Regressor	Without Outliers With Standardization	84.95

We choose the following classifier models as they were most accurate and in accordance with each other:

Classifier Model Trained	Data Used	Data Used Accuracy %
Linear Regression	Without Outliers With Standardization	95.44
Random Forest	Without Outliers With Standardization	95.94
Decision Tree Regressor	Without Outliers Without Standardization	85.16



So from the above comparison we have concluded that the random forest regressor model has found the highest accuracy. Hence our selected model is **Random Forest Regressor Without Outliers With Standardization** with the accuracy score **95.94%.**

Code

Life Expectancy Prediction

About Dataset

Total Columns = 22

Total Rows = 2938

Columns

- **Country : Country Names (193 Unique Contries name)**

- **Year : Year (2000 - 2013)**
- **Status : Developed or Developing status**
- **Life expectancy : Life Expectancy in age**
- **Adult Mortality : Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population)**
- **infant deaths : Number of Infant Deaths per 1000 population**
- **Alcohol : Alcohol, recorded per capita (15+) consumption (in litres of pure alcohol)**
- **percentage expenditure : Expenditure on health as a percentage of Gross Domestic Product per capita(%)**
- **Hepatitis B : Hepatitis B (HepB) immunization coverage among 1-year-olds (%)**
- **Measles : Measles - number of reported cases per 1000 population**
- **BMI : Average Body Mass Index of entire population**
- **under-five deaths : Number of under-five deaths per 1000 population**
- **Polio : Polio (Pol3) immunization coverage among 1-year-olds (%)**
- **Total expenditure : General government expenditure on health as a percentage of total government expenditure (%)**
- **Diphtheria : Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds (%)**
- **HIV/AIDS : Deaths per 1 000 live births HIV/AIDS (0-4 years)**
- **GDP : Gross Domestic Product per capita (in USD)**
- **Population : Population of the country**
- **thinness 1-19 years : Prevalence of thinness among children and adolescents for Age 10 to 19 (%)**
- **thinness 5-9 years : Prevalence of thinness among children for Age 5 to 9(%)**
- **Income composition of resources : Human Development Index in terms of income composition of resources (index ranging from 0 to 1)**
- **Schooling : Number of years of Schooling(years)**

- **Dataset Link : https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who**

import Libraries

In [206…

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import OrdinalEncoder, FunctionTransformer, StandardScaler, OneHot
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
```

My Functions

In [207…

```
def graph(df, columnName):
    plt.figure(figsize=(15, 10))
    plt.subplot(2, 2, 1)
    plt.title("Box Plot")
    sns.boxplot(data=df[columnName])

    plt.subplot(2, 2, 2)
    plt.title("Histogram")
    sns.histplot(data=df[columnName])
```

```
    plt.subplot(2, 2, 3)
    plt.title("KDE Plot")
    sns.kdeplot(data=df[columnName])

    plt.subplot(2, 2, 4)
    plt.title("Scatter Plot")
    sns.scatterplot(data=df, x="Life expectancy ", y=columnName, hue="Status")

    plt.show()
```

In [208…

```
def cap(df_old, df):
    cols = df.columns
    for i in range(3, len(cols)):
        columnName = cols[i]
        persentile25 = df_old[columnName].quantile(0.25)
        persentile75 = df_old[columnName].quantile(0.75)
        iqr = persentile75 - persentile25
        upper_limit = persentile75 + 1.5 * iqr
        lower_limit = persentile25 - 1.5 * iqr
        print("###############################################################################")
        print("\nVariable Name : ", columnName)

        if ((df[columnName] > upper_limit).any() or (df[columnName] < lower_limit).any()):

            df[columnName] = np.where(
                df[columnName] > upper_limit, upper_limit,
                np.where(
                    df[columnName] < lower_limit, lower_limit,
                    df[columnName]
                )
            )

            print("Persentile 25 : ", persentile25)
            print("Persentile 75 : ", persentile75)
            print("IQR : ", iqr)
            print("Upper Limit : ", upper_limit)
            print("Lower Limit : ", lower_limit)
            print("Outliers Removed using IQR Method by Capping")

            plt.figure(figsize=(15, 20))
            plt.subplot(4, 2, 1)
            plt.title("Box Plot(Before capping)")
            sns.boxplot(data=df_old[columnName])

            plt.subplot(4, 2, 2)
            plt.title("Box Plot(After capping)")
            sns.boxplot(data=df[columnName])

            plt.subplot(4, 2, 3)
            plt.title("Histogram(Before capping)")
            sns.histplot(data=df_old[columnName])

            plt.subplot(4, 2, 4)
            plt.title("Histogram(After capping)")
            sns.histplot(data=df[columnName])

            plt.subplot(4, 2, 5)
            plt.title("KDE Plot(Before capping)")
            sns.kdeplot(data=df_old[columnName])

            plt.subplot(4, 2, 6)
            plt.title("KDE Plot(After capping)")
            sns.kdeplot(data=df[columnName])

            plt.subplot(4, 2, 7)
            plt.title("Scatter Plot(Before capping)")
            sns.scatterplot(data=df_old, x="Life expectancy ", y=columnName, hue="Status")
```

```
            plt.subplot(4, 2, 8)
            plt.title("Scatter Plot(After capping)")
            sns.scatterplot(data=df, x="Life expectancy ", y=columnName, hue="Status")

            plt.show()
        else:
            print("No Outliers found")
```

Load the dataset

In [209…
```
df = pd.read_csv("D:\Dataset\Life Expectancy Data.csv")
```

Case study on the whole dataset

Shape

In [210…
```
df.shape
```

Out[210]:
```
(2938, 22)
```

Data Sample

In [211…
```
df.head()
```

Out[211]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	..
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	..
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	..
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	..
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	..
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	..

5 rows × 22 columns

information of all Columns

In [212…
```
df.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
```

Null value count column wise

In [213…
```
df.isnull().sum()
```

Out[213]:
```
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
```

Null value percentage

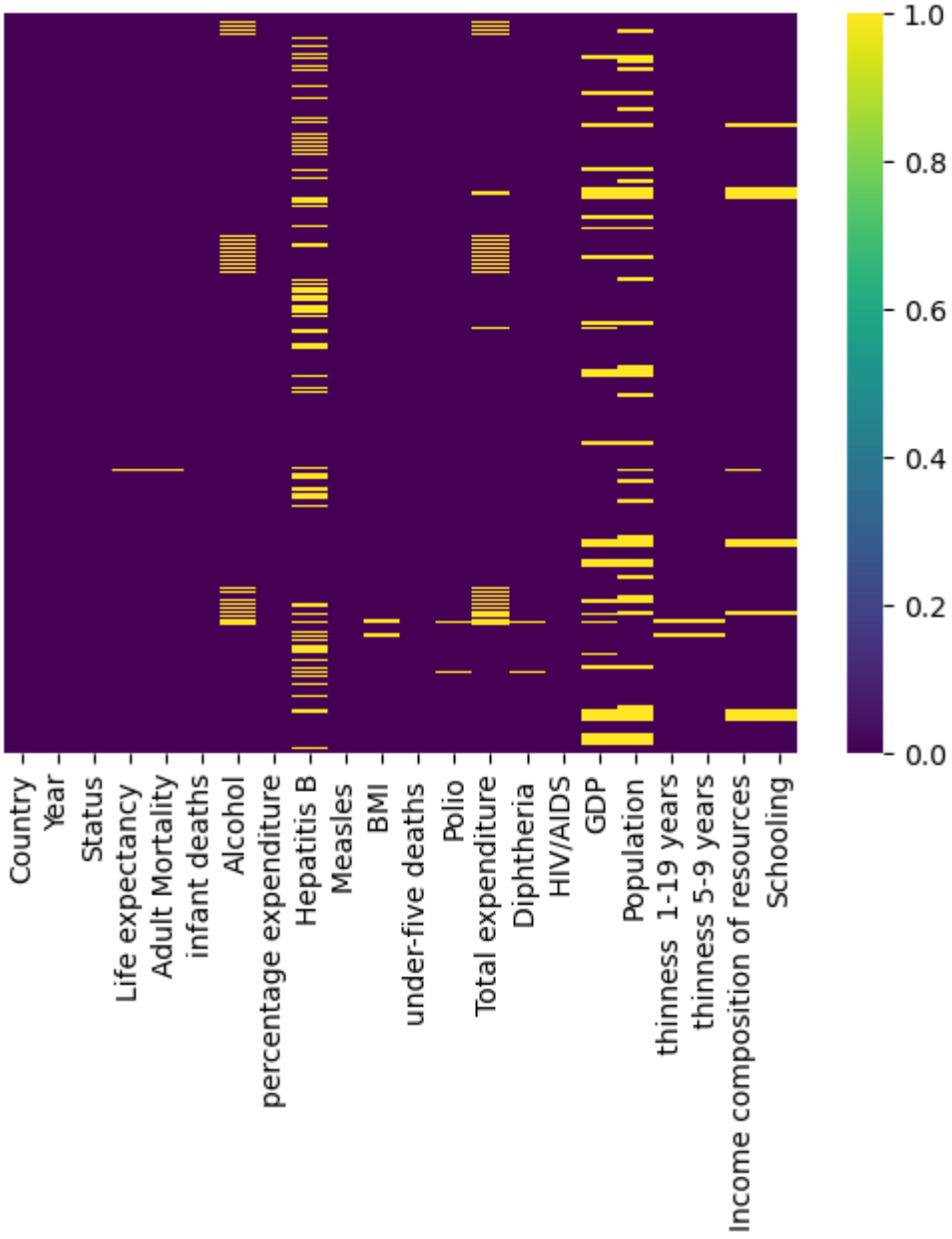
In [214…
```
df.isnull().mean()*100
```

Out[214]:
```
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
```

Null values by heatmap

In [215…
```
sns.heatmap(df.isnull(), yticklabels = False, cbar = True, cmap = 'viridis')
```

Out[215]:
```
<AxesSubplot:>
```



Columns Desription

In [216…

```
df.describe()
```

Out[216]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	
count	2938.000000	2928.000000	2928.000000	2938.000000	2744.000000	2938.000000	2385.000000	29
mean	2007.518720	69.224932	164.796448	30.303948	4.602861	738.251295	80.940461	24
std	4.613841	9.523867	124.292079	117.926501	4.052413	1987.914858	25.070016	114
min	2000.000000	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	
25%	2004.000000	63.100000	74.000000	0.000000	0.877500	4.685343	77.000000	
50%	2008.000000	72.100000	144.000000	3.000000	3.755000	64.912906	92.000000	
75%	2012.000000	75.700000	228.000000	22.000000	7.702500	441.534144	97.000000	3
max	2015.000000	89.000000	723.000000	1800.000000	17.870000	19479.911610	99.000000	2121

Duplicate row count

In [217…
```
df.duplicated().sum()
```

Out[217]:
```
0
```

Correlationn between each coloumns

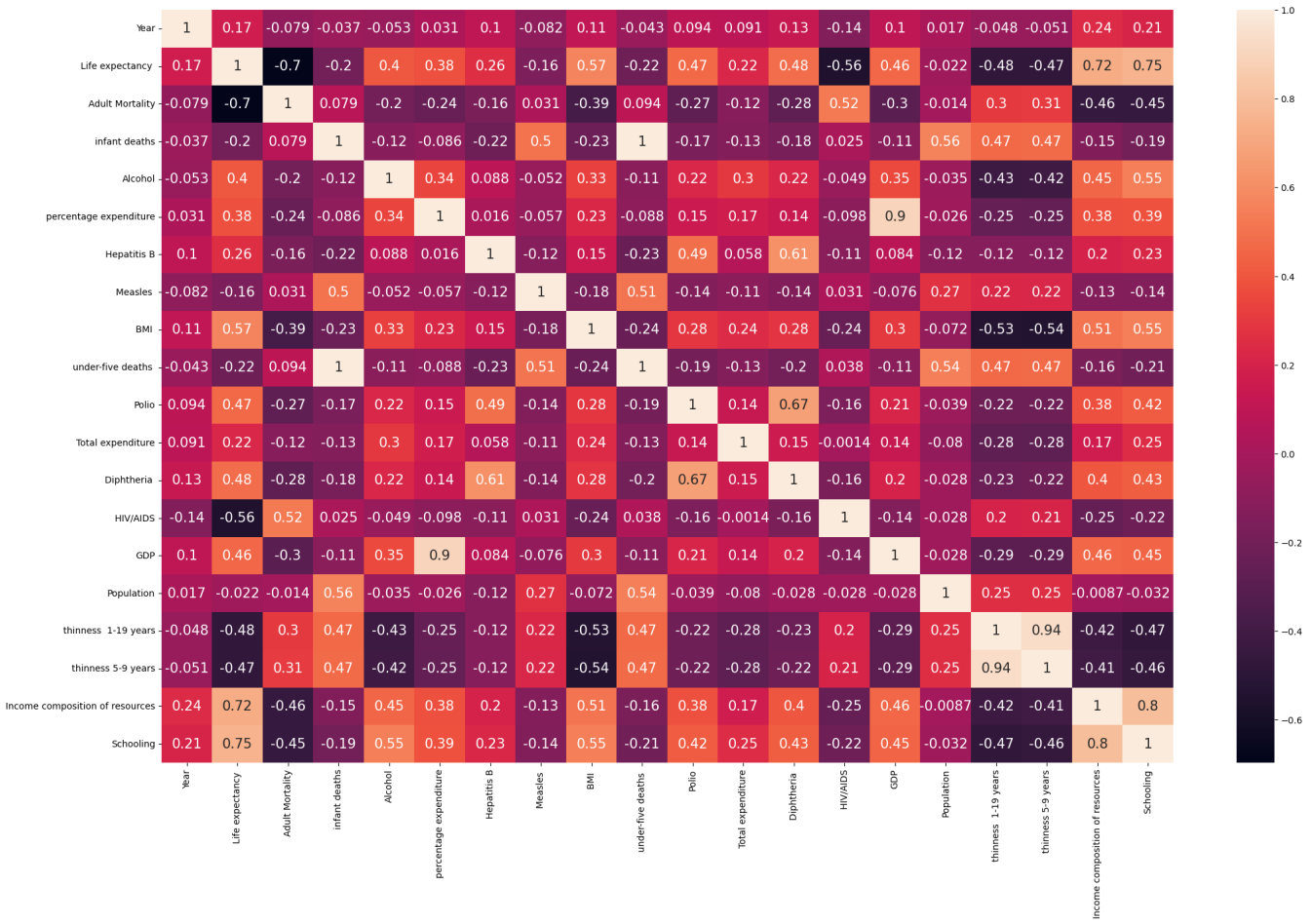
In [218…
```
df.corr()
```

Out[218]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	
Year	1.000000	0.170033	-0.079052	-0.037415	-0.052990	0.031400	0.104333	-0.082493	0.
Life expectancy	0.170033	1.000000	-0.696359	-0.196557	0.404877	0.381864	0.256762	-0.157586	0.
Adult Mortality	-0.079052	-0.696359	1.000000	0.078756	-0.195848	-0.242860	-0.162476	0.031176	-0.
infant deaths	-0.037415	-0.196557	0.078756	1.000000	-0.115638	-0.085612	-0.223566	0.501128	-0.
Alcohol	-0.052990	0.404877	-0.195848	-0.115638	1.000000	0.341285	0.087549	-0.051827	0.
percentage expenditure	0.031400	0.381864	-0.242860	-0.085612	0.341285	1.000000	0.016274	-0.056596	0.
Hepatitis B	0.104333	0.256762	-0.162476	-0.223566	0.087549	0.016274	1.000000	-0.120529	0.
Measles	-0.082493	-0.157586	0.031176	0.501128	-0.051827	-0.056596	-0.120529	1.000000	-0.
BMI	0.108974	0.567694	-0.387017	-0.227279	0.330408	0.228700	0.150380	-0.175977	1.
under-five deaths	-0.042937	-0.222529	0.094146	0.996629	-0.112370	-0.087852	-0.233126	0.507809	-0.
Polio	0.094158	0.465556	-0.274823	-0.170689	0.221734	0.147259	0.486171	-0.136166	0.
Total expenditure	0.090740	0.218086	-0.115281	-0.128616	0.296942	0.174420	0.058280	-0.106241	0.
Diphtheria	0.134337	0.479495	-0.275131	-0.175171	0.222020	0.143624	0.611495	-0.141882	0.
HIV/AIDS	-0.139741	-0.556556	0.523821	0.025231	-0.048845	-0.097857	-0.112675	0.030899	-0.
GDP	0.101620	0.461455	-0.296049	-0.108427	0.354712	0.899373	0.083903	-0.076466	0.
Population	0.016969	-0.021538	-0.013647	0.556801	-0.035252	-0.025662	-0.123321	0.265966	-0.
thinness 1-19 years	-0.047876	-0.477183	0.302904	0.465711	-0.428795	-0.251369	-0.120429	0.224808	-0.
thinness 5-9 years	-0.050929	-0.471584	0.308457	0.471350	-0.417414	-0.252905	-0.124960	0.221072	-0.
Income composition of resources	0.243468	0.724776	-0.457626	-0.145139	0.450040	0.381952	0.199549	-0.129568	0.
Schooling	0.209400	0.751975	-0.454612	-0.193720	0.547378	0.389687	0.231117	-0.137225	0.

Correlation between each coloumns by heatmap

In [219…
```
plt.figure(figsize=(25,15))
sns.heatmap(df.corr(), annot = True, annot_kws = {"size":15})
plt.show()
```



Correlation with **Life expectancy**

In [220…

```
df.corr()["Life expectancy "]
```

Out[220]:

```
Year                               0.170033
Life expectancy                    1.000000
Adult Mortality                   -0.696359
infant deaths                     -0.196557
Alcohol                            0.404877
percentage expenditure             0.381864
Hepatitis B                        0.256762
Measles                           -0.157586
 BMI                               0.567694
under-five deaths                 -0.222529
Polio                              0.465556
Total expenditure                  0.218086
Diphtheria                         0.479495
 HIV/AIDS                         -0.556556
GDP                                0.461455
Population                        -0.021538
 thinness  1-19 years             -0.477183
 thinness 5-9 years               -0.471584
Income composition of resources    0.724776
Schooling                          0.751975
Name: Life expectancy , dtype: float64
```

Case study on every veriable of the dataset

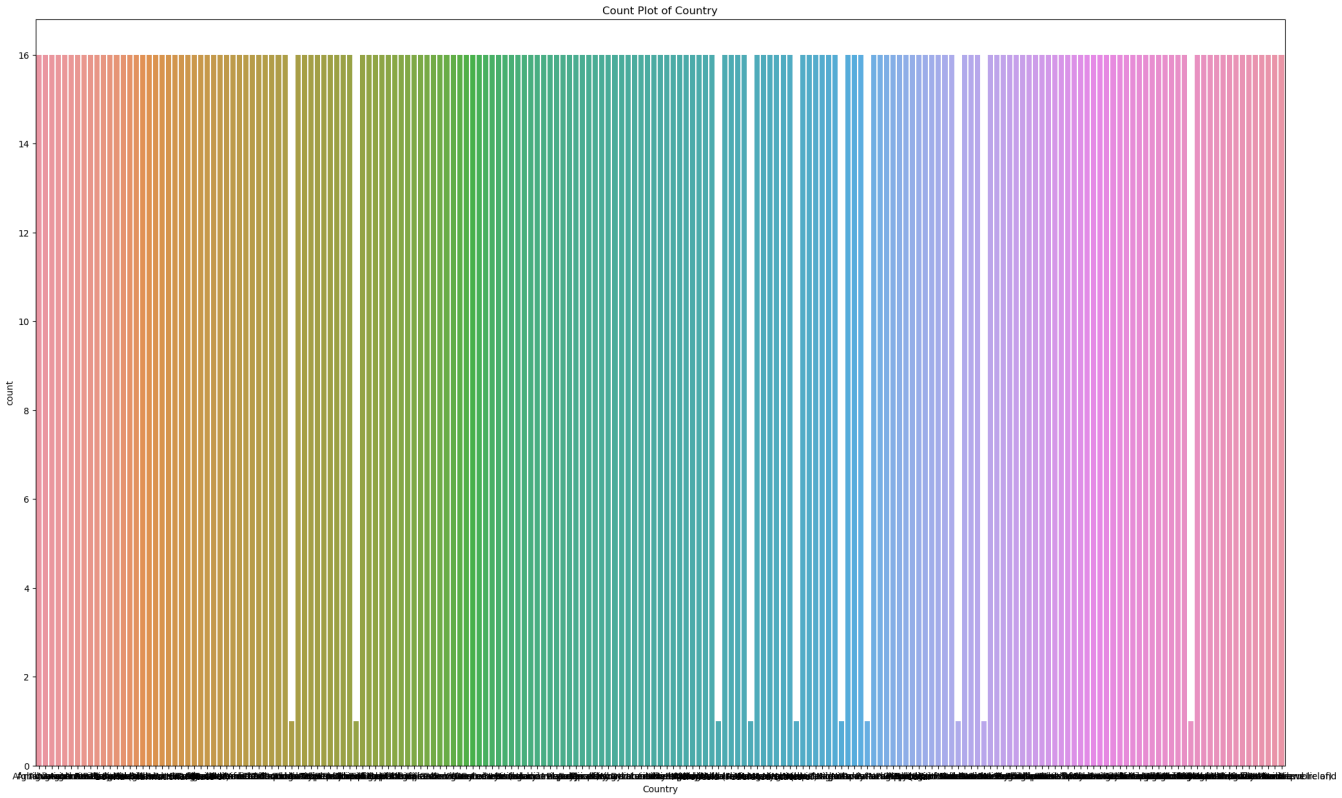
Country (catagorical)

In [221…

```
df["Country"].value_counts()
```

Out[221]:
```
Afghanistan                16
Peru                       16
Nicaragua                  16
Niger                      16
Nigeria                    16
                           ..
Niue                        1
San Marino                  1
Nauru                       1
Saint Kitts and Nevis       1
Dominica                    1
Name: Country, Length: 193, dtype: int64
```

In [222…
```
plt.figure(figsize=(25,15))
plt.title("Count Plot of Country")
sns.countplot(data=df, x="Country")
plt.show()
```



In [223…
```
df["Country"].unique()
```

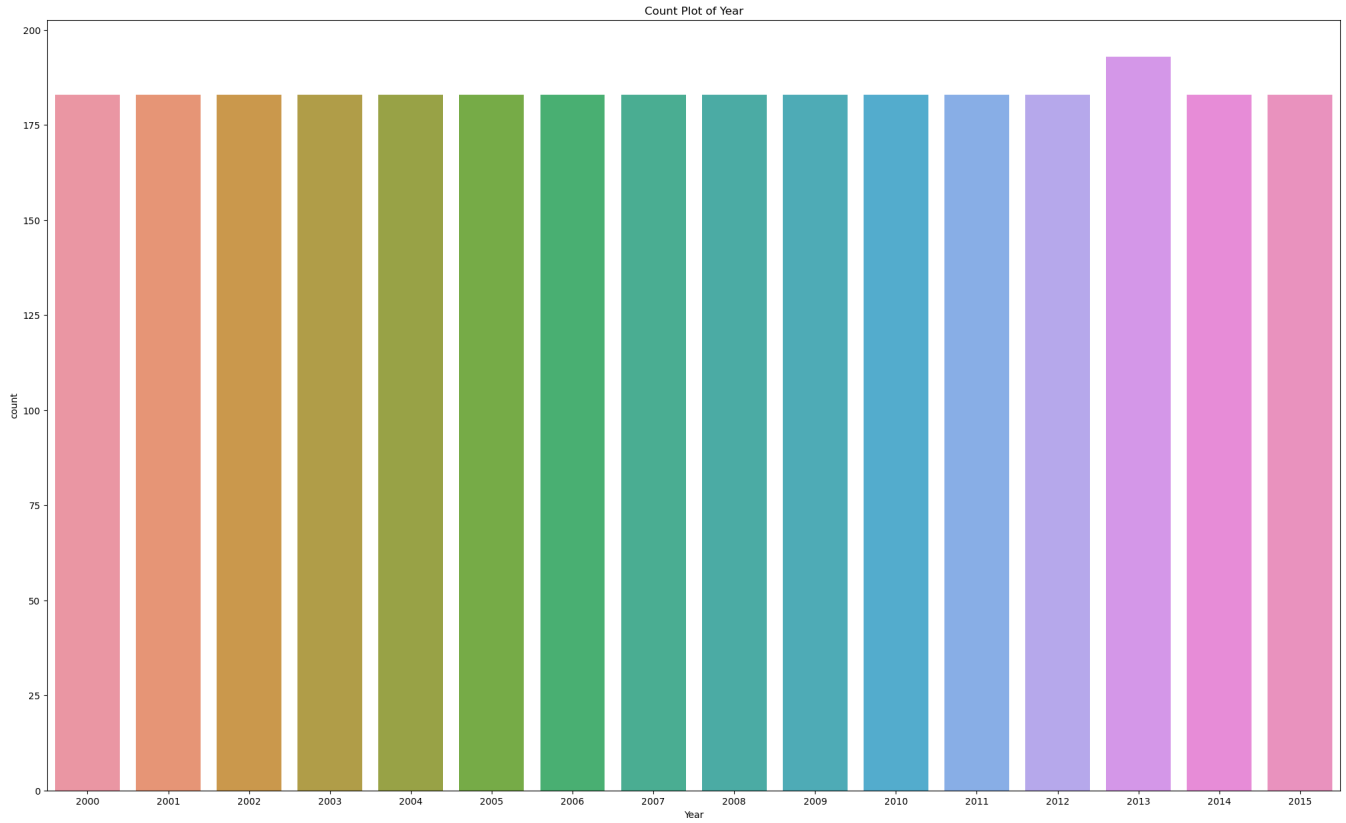
Out[223]:
```
array(['Afghanistan', 'Albania', 'Algeria', 'Angola',
       'Antigua and Barbuda', 'Argentina', 'Armenia', 'Australia',
       'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh',
       'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan',
       'Bolivia (Plurinational 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)
```

Year (catagorical)

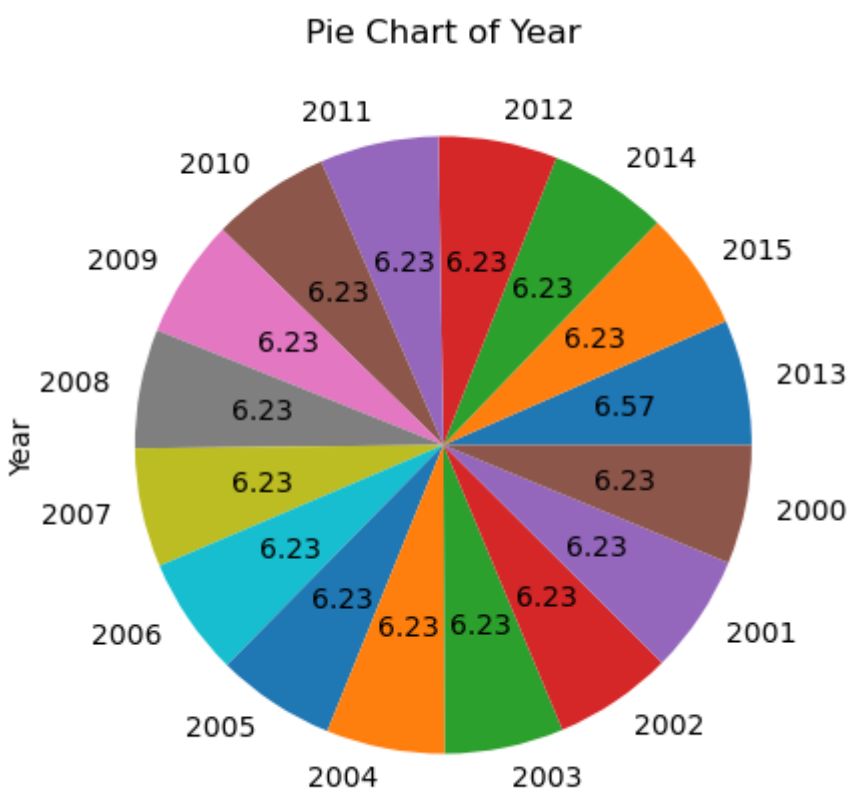
In [224…
```
df["Year"].value_counts()
```

Out[224]:
```
2013    193
2015    183
2014    183
2012    183
2011    183
2010    183
2009    183
2008    183
2007    183
2006    183
2005    183
2004    183
2003    183
2002    183
2001    183
2000    183
Name: Year, dtype: int64
```

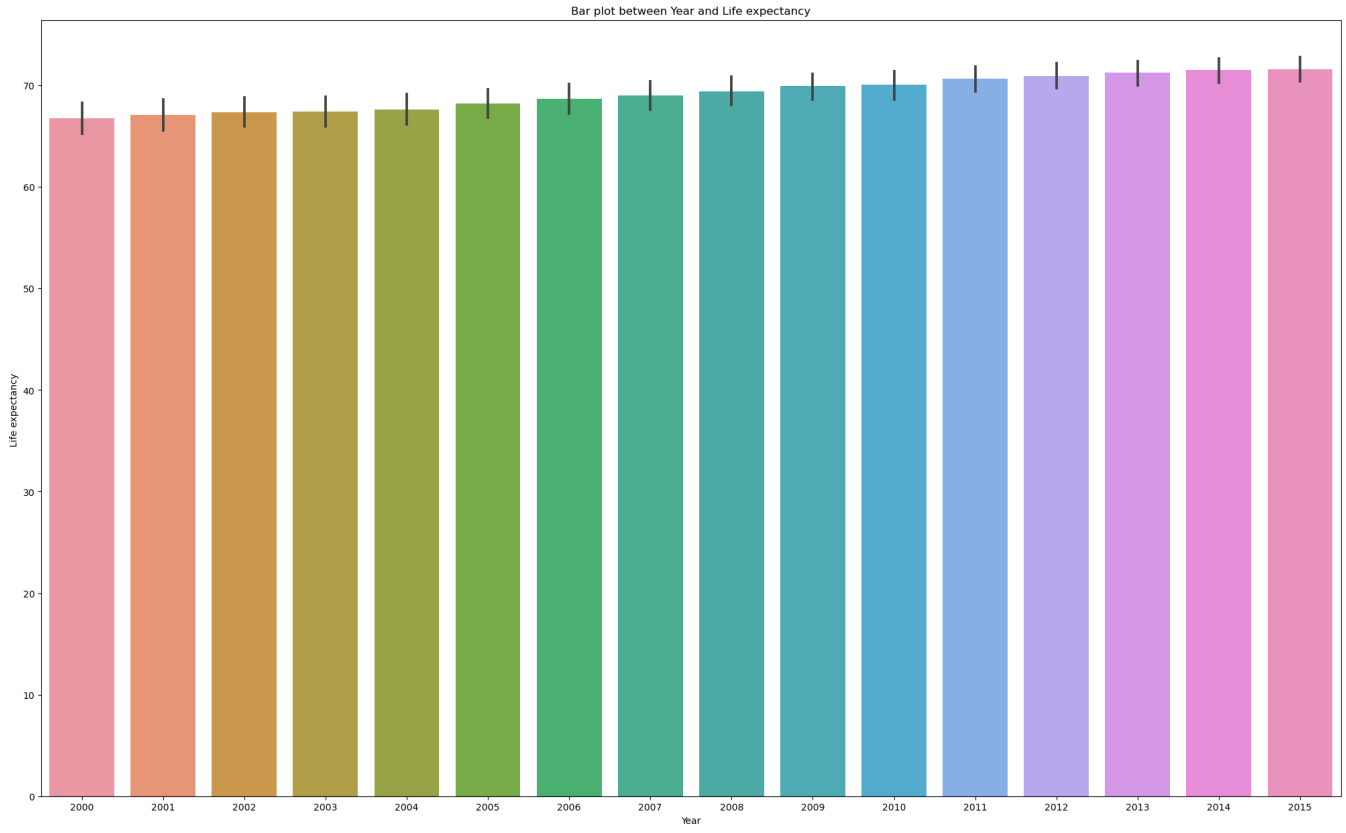
In [225…
```
plt.figure(figsize=(25,15))
plt.title("Count Plot of Year")
sns.countplot(data=df, x="Year")
plt.show()
```



In [226…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Year")
df["Year"].value_counts().plot(kind="pie", autopct="%.2f")
plt.show()
```



In [227…
```
plt.figure(figsize=(25,15))
plt.title("Bar plot between Year and Life expectancy")
sns.barplot(data=df, x="Year", y="Life expectancy ")
plt.show()
```



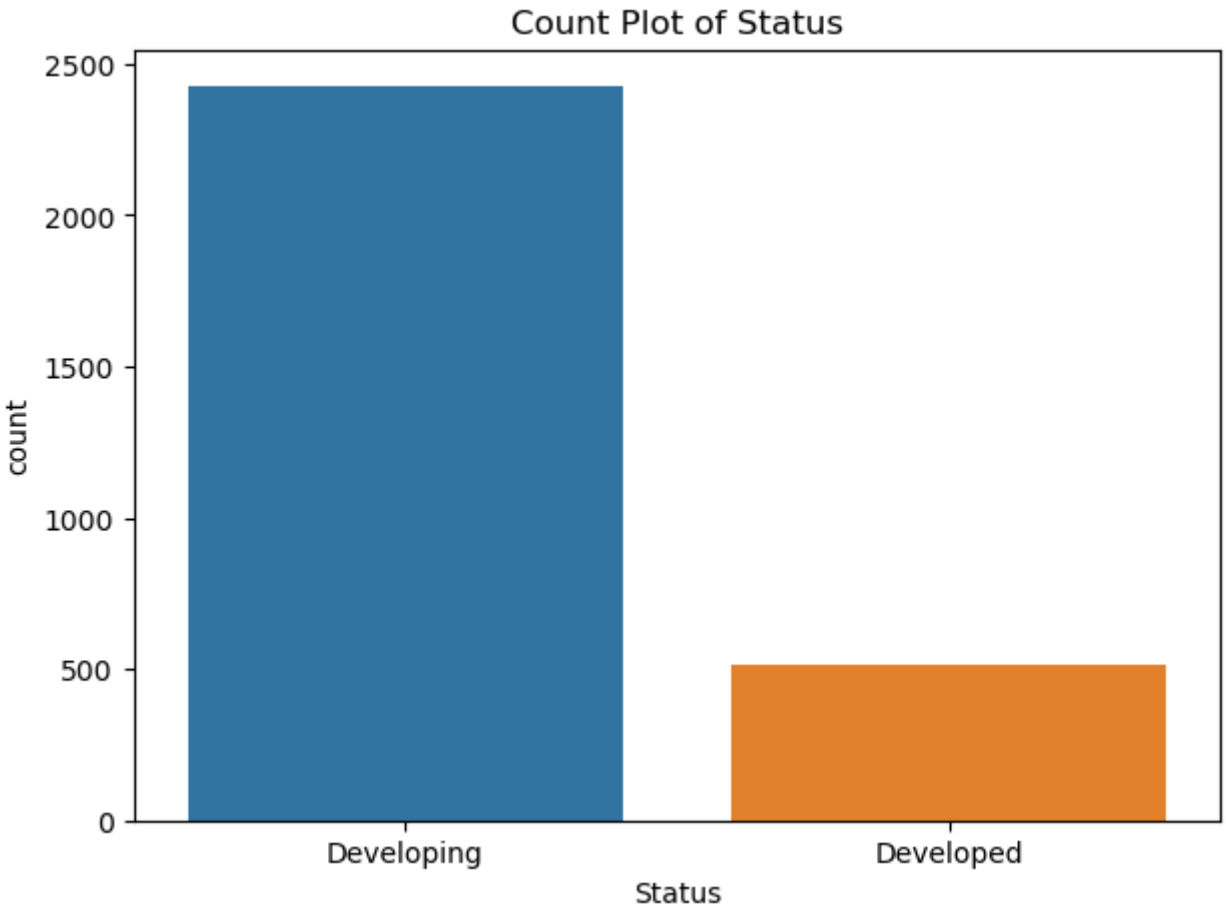
Status (catagorical)

In [228…
```
df["Status"].value_counts()
```

Out[228]:
```
Developing    2426
Developed      512
Name: Status, dtype: int64
```

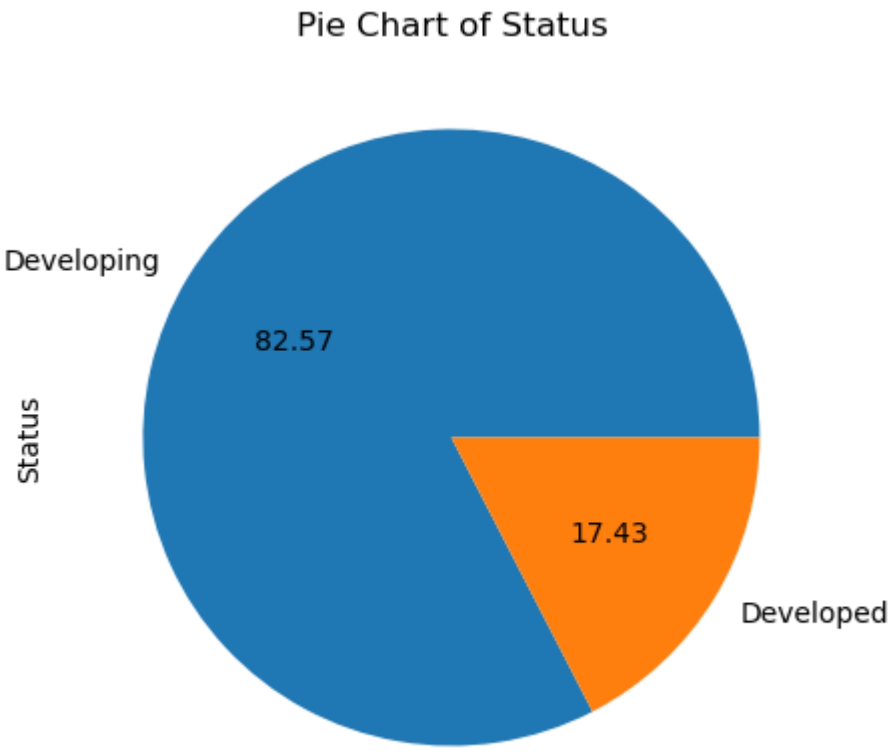
In [229…

```
plt.figure(figsize=(7,5))
plt.title("Count Plot of Status")
sns.countplot(data=df, x="Status")
plt.show()
```



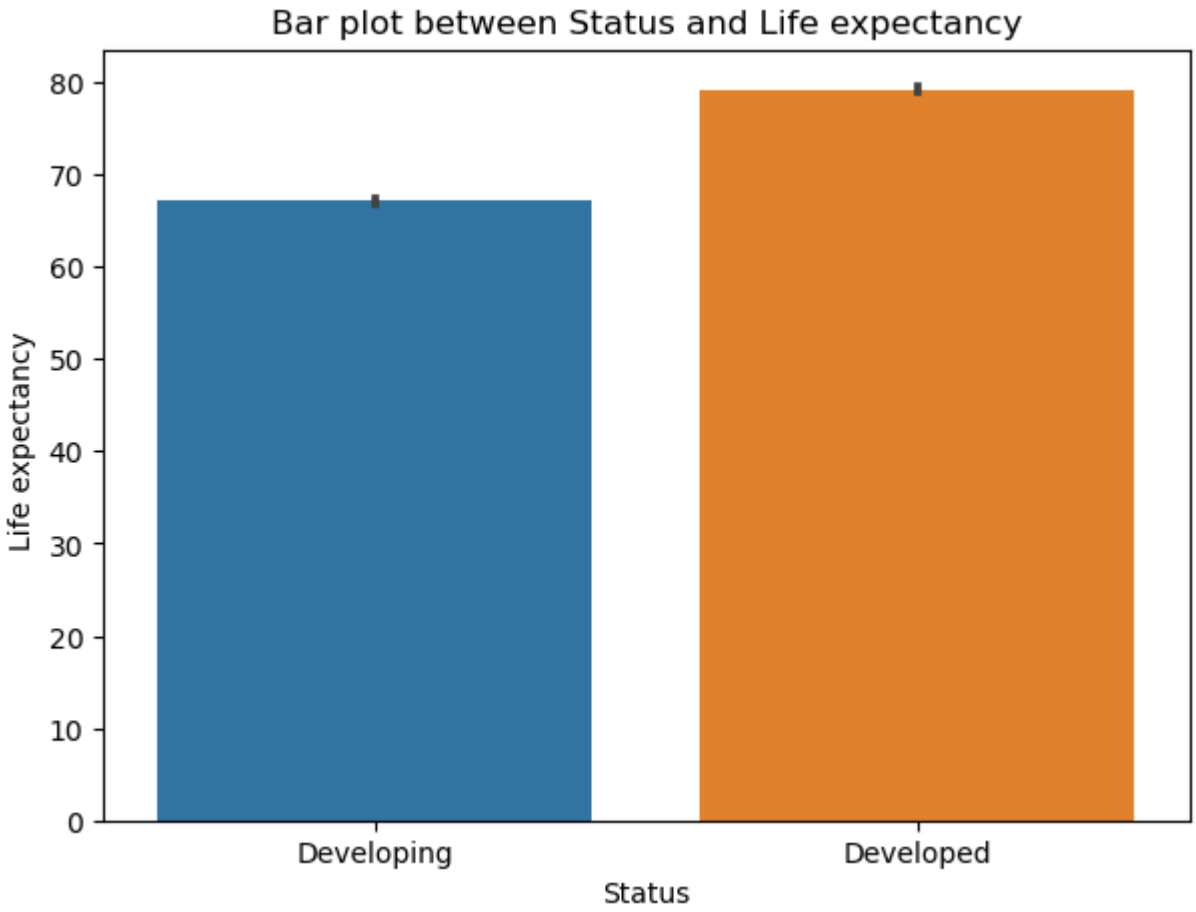
In [230…

```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Status")
df["Status"].value_counts().plot(kind="pie", autopct="%.2f")
plt.show()
```



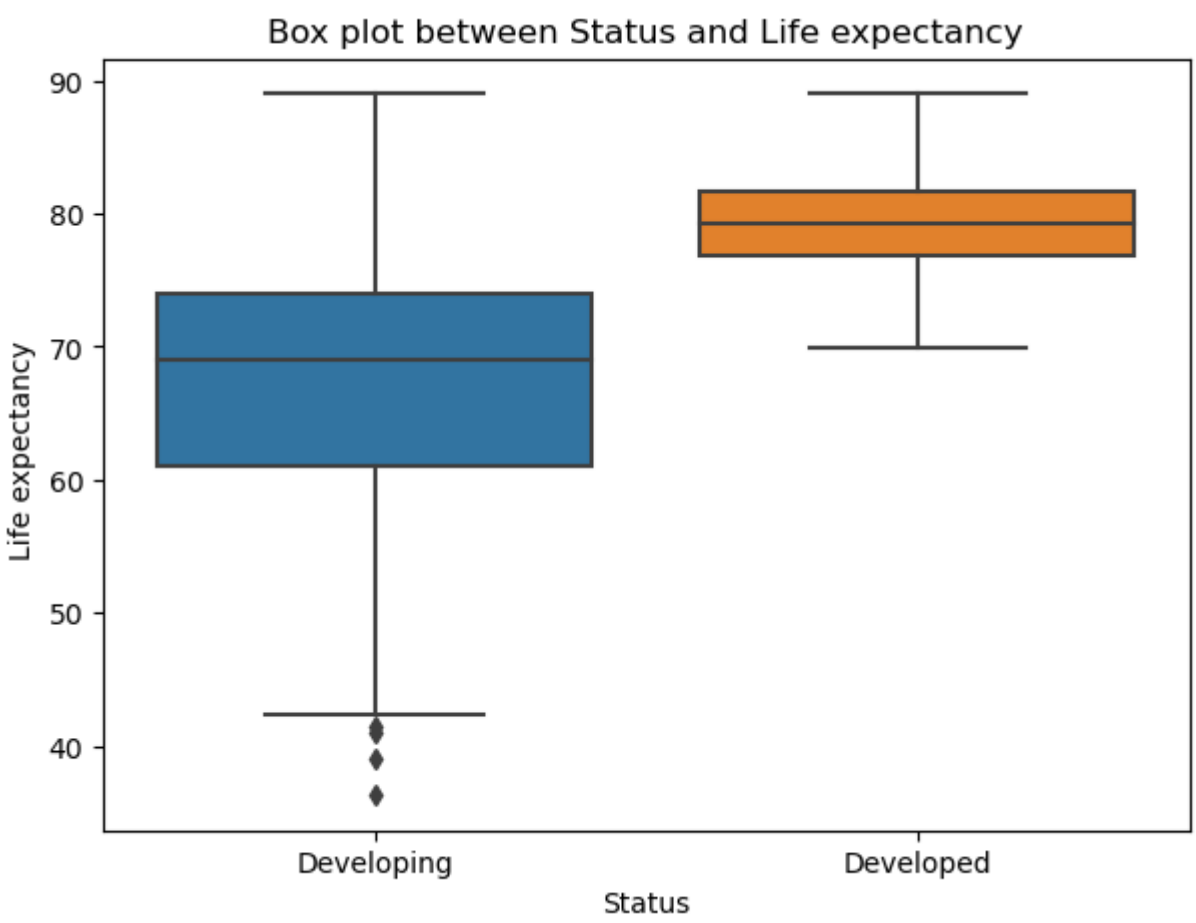
In [231…

```
plt.figure(figsize=(7,5))
plt.title("Bar plot between Status and Life expectancy")
sns.barplot(data=df, x="Status", y="Life expectancy ")
plt.show()
```



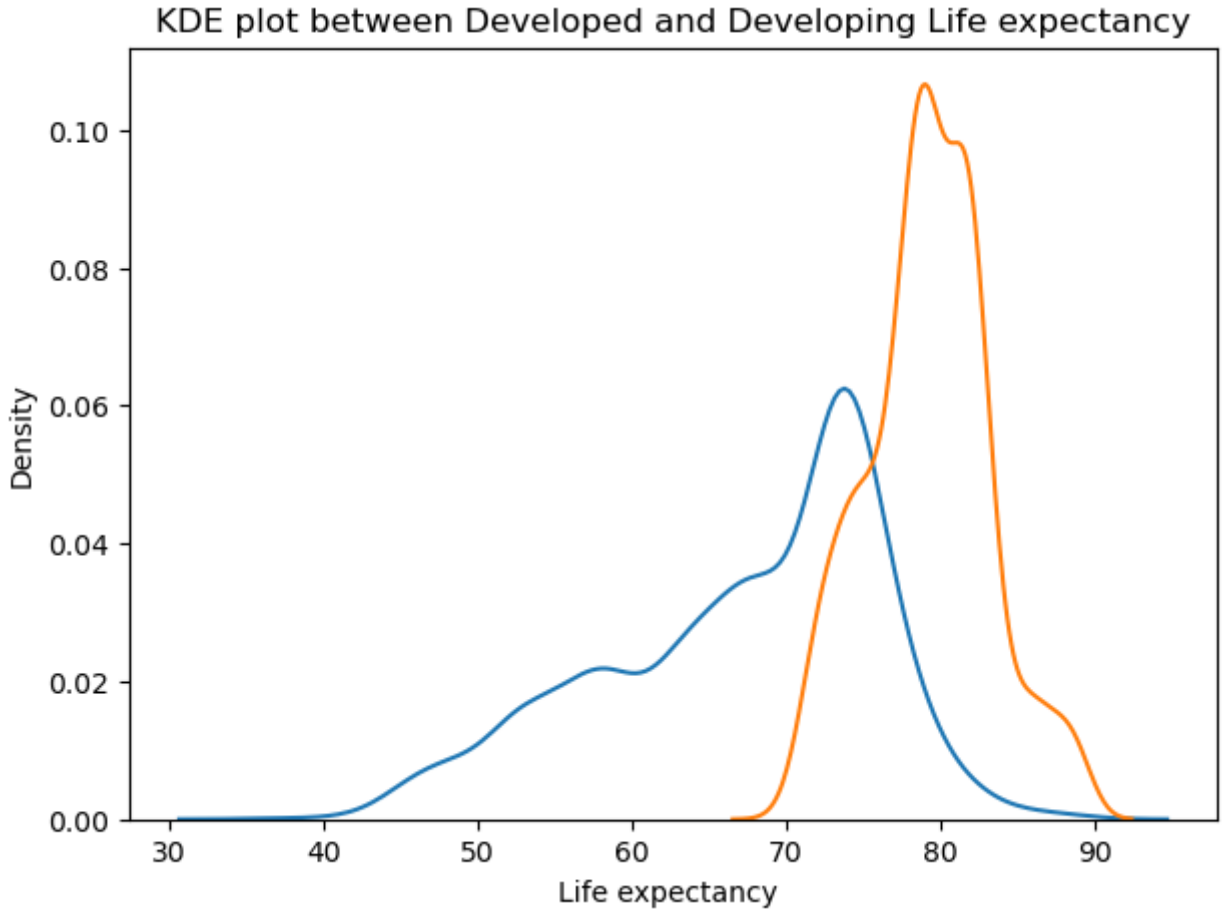
In [232…

```
plt.figure(figsize=(7,5))
plt.title("Box plot between Status and Life expectancy")
sns.boxplot(data=df, x="Status", y="Life expectancy ")
plt.show()
```



In [233…

```
plt.figure(figsize=(7,5))
plt.title("KDE plot between Developed and Developing Life expectancy")
sns.kdeplot(df[df["Status"]=="Developing"]["Life expectancy "])
sns.kdeplot(df[df["Status"]=="Developed"]["Life expectancy "])
plt.show()
```



Life expectancy (numerical)

In [234…
```
df["Life expectancy "].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: Life expectancy
Non-Null Count  Dtype
--------------  -----
2928 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

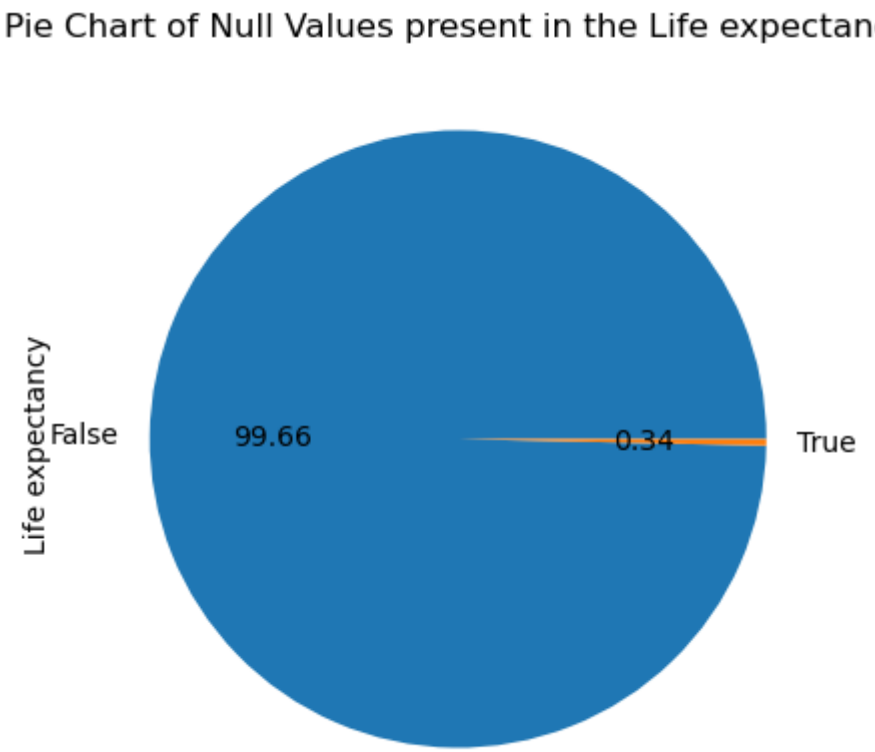
In [235…
```
df["Life expectancy "].describe()
```

Out[235]:
```
count    2928.000000
mean       69.224932
std         9.523867
min        36.300000
25%        63.100000
50%        72.100000
75%        75.700000
max        89.000000
Name: Life expectancy , dtype: float64
```

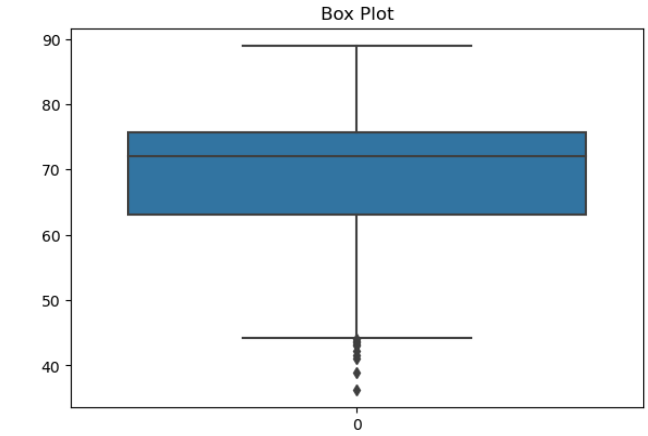
In [236…
```
df["Life expectancy "].isnull().sum()
```

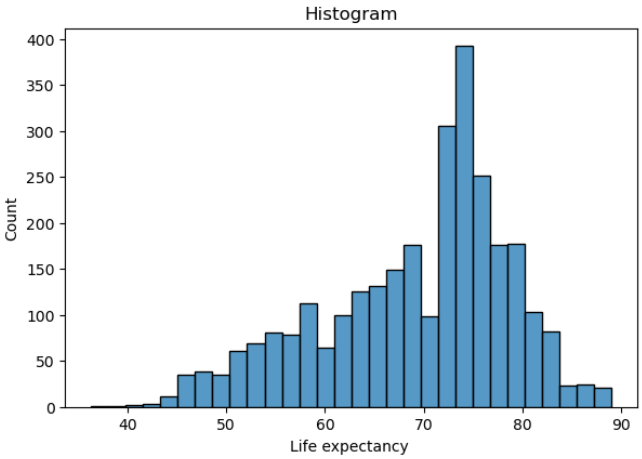
Out[236]:
```
10
```

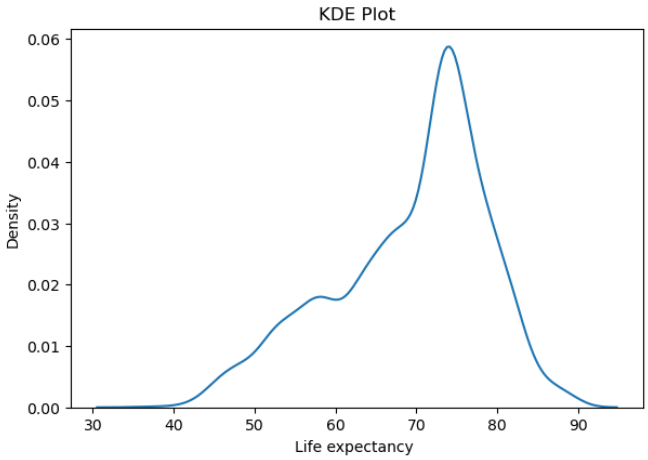
In [237…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the Life expectancy")
df["Life expectancy "].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="Life expectancy ")
plt.show()
```

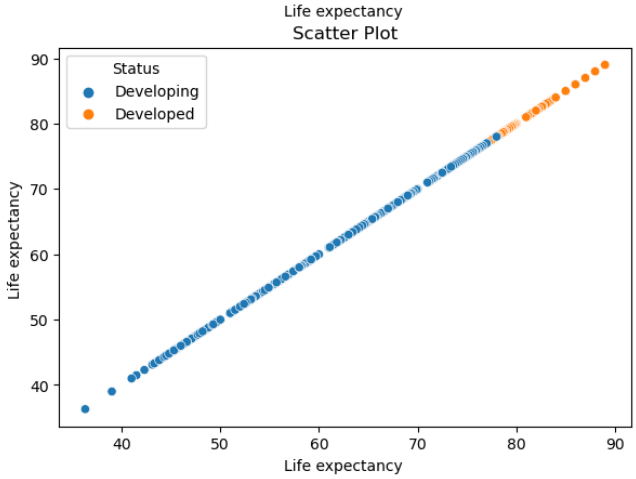


In [238…
```
graph(df, "Life expectancy ")
```









Adult Mortality (numerical)

In [239…
```
df["Adult Mortality"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: Adult Mortality
Non-Null Count  Dtype
--------------  -----
2928 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

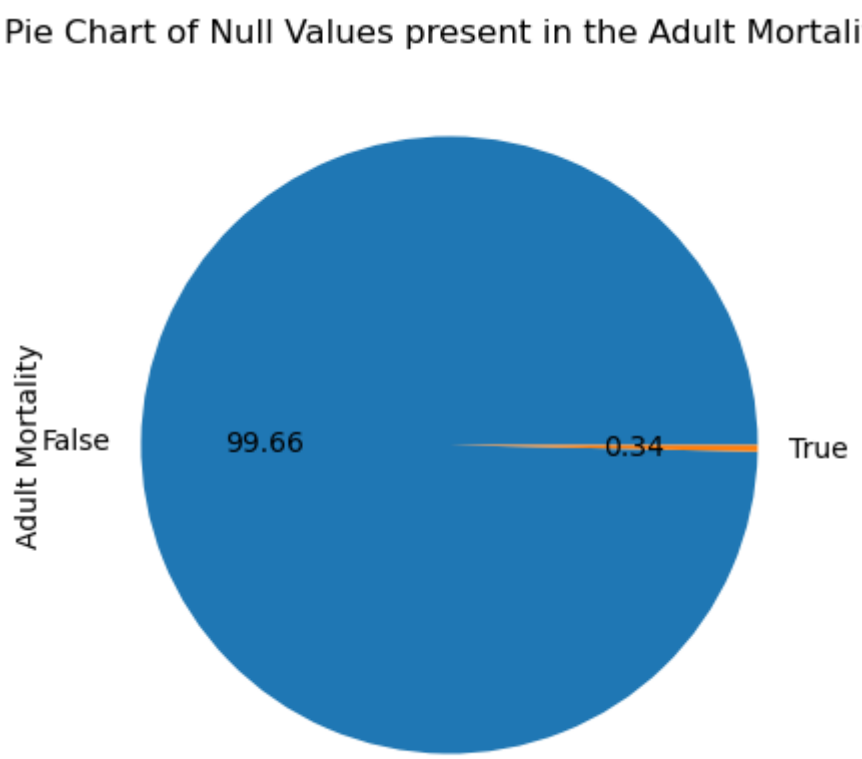
In [240…
```
df["Adult Mortality"].describe()
```

Out[240]:
```
count    2928.000000
mean      164.796448
std       124.292079
min         1.000000
25%        74.000000
50%       144.000000
75%       228.000000
max       723.000000
Name: Adult Mortality, dtype: float64
```

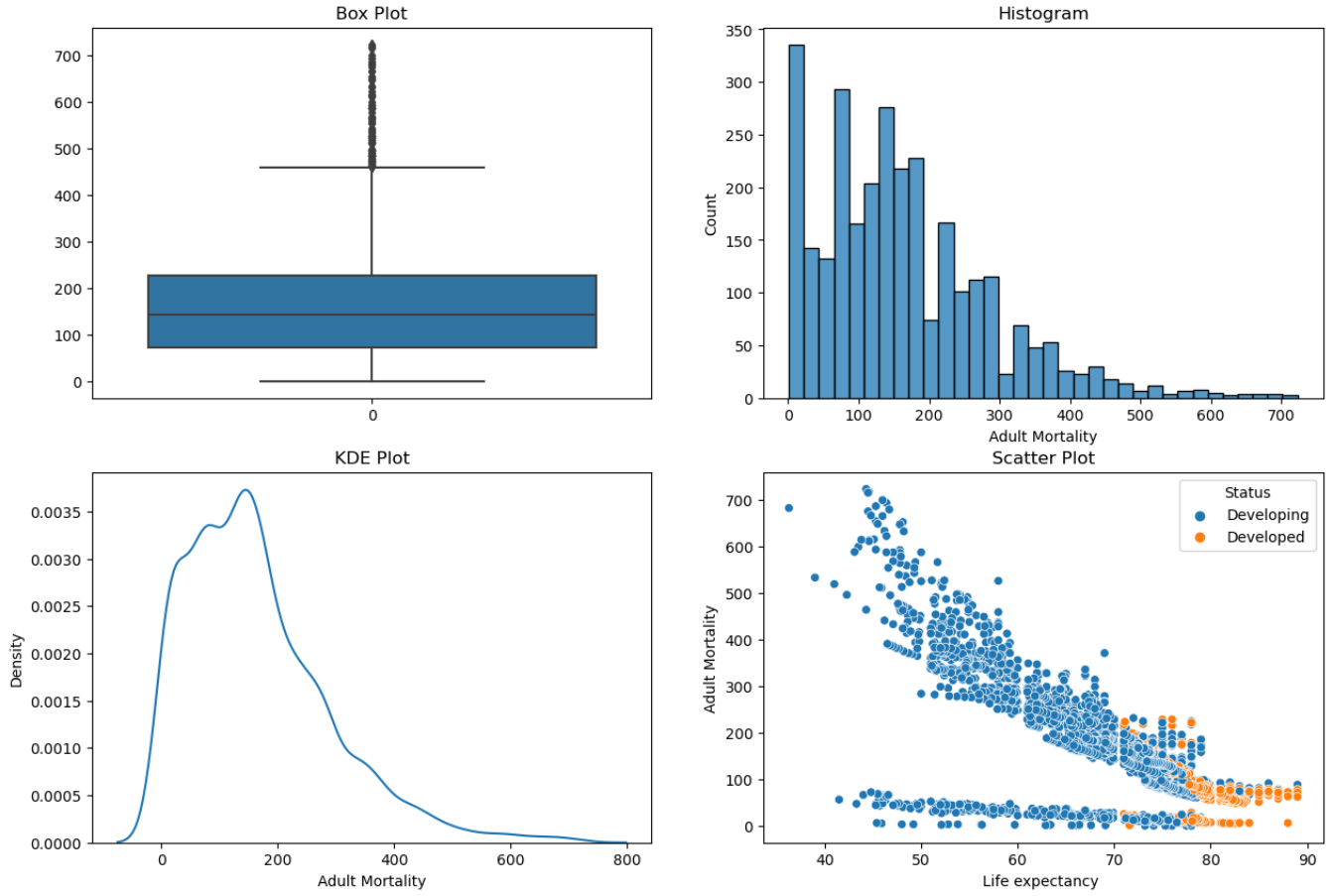
In [241…
```
df["Adult Mortality"].isnull().sum()
```

Out[241]:
```
10
```

In [242…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the Adult Mortality")
df["Adult Mortality"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="Adult Mortality")
plt.show()
```



In [243…
```
graph(df,"Adult Mortality")
```



Infant deaths (numerical)

In [244…
```
df["infant deaths"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: infant deaths
Non-Null Count  Dtype
--------------  -----
2938 non-null   int64
dtypes: int64(1)
memory usage: 23.1 KB
```

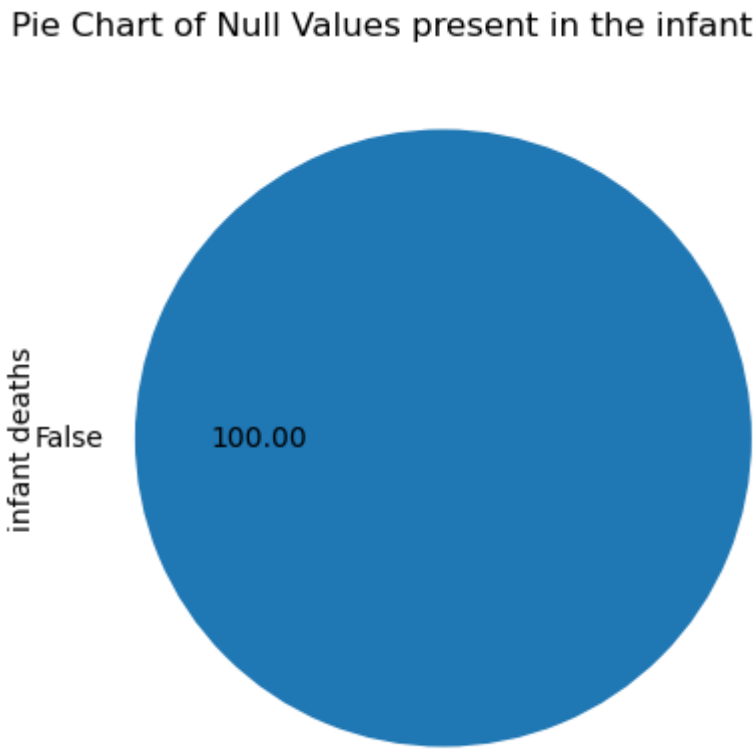
In [245…
```
df["infant deaths"].describe()
```

Out[245]:
```
count    2938.000000
mean       30.303948
std       117.926501
min         0.000000
25%         0.000000
50%         3.000000
75%        22.000000
max      1800.000000
Name: infant deaths, dtype: float64
```

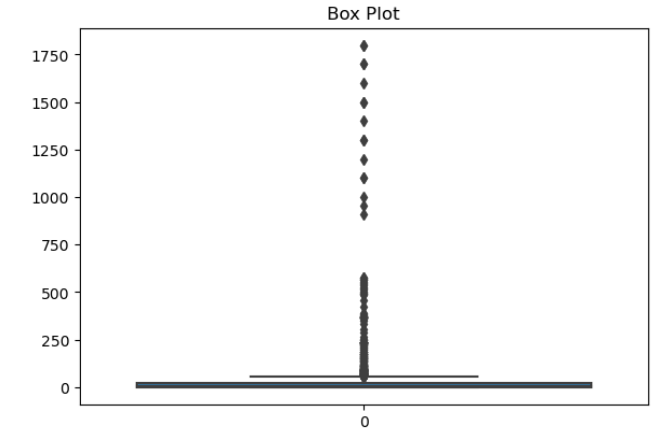
In [246…
```
df["infant deaths"].isnull().sum()
```

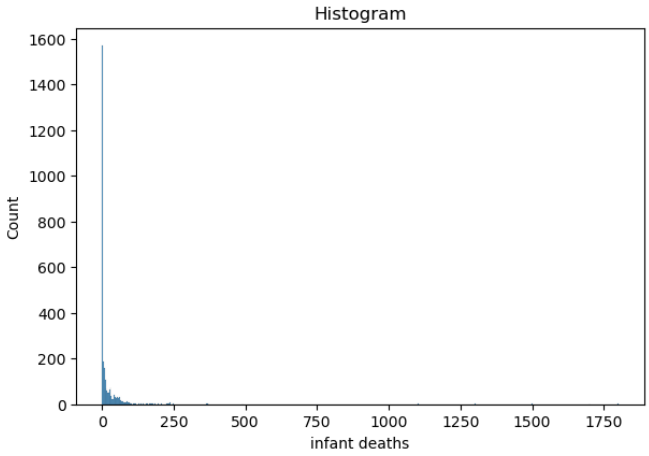
Out[246]:
```
0
```

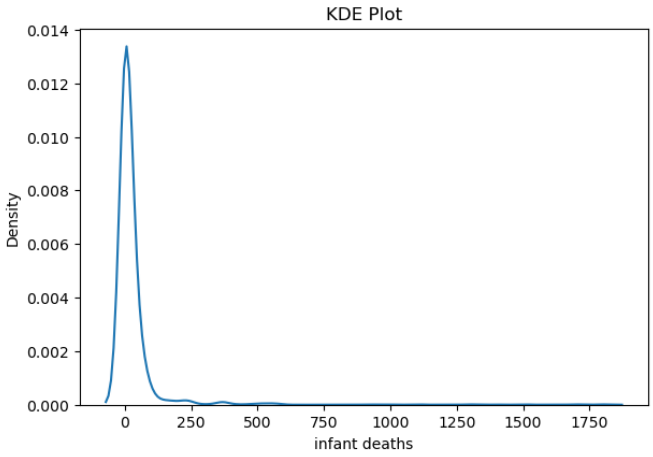
In [247…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the infant deaths")
df["infant deaths"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="Adult Mortality")
plt.show()
```

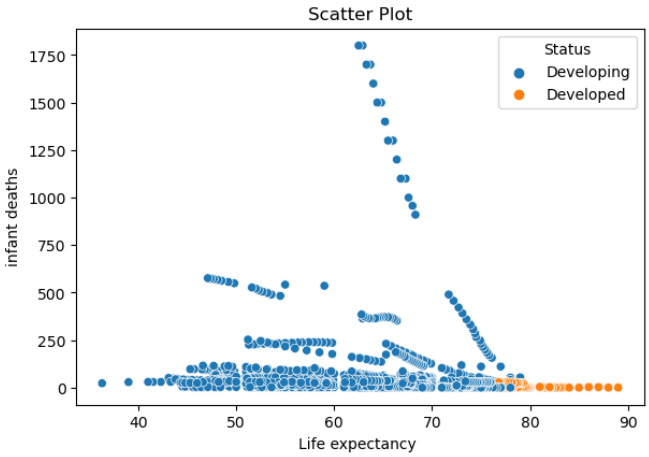


In [248…
```
graph(df, "infant deaths")
```









Alcohol (numerical)

In [249…
```
df["Alcohol"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: Alcohol
Non-Null Count  Dtype
--------------  -----
2744 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

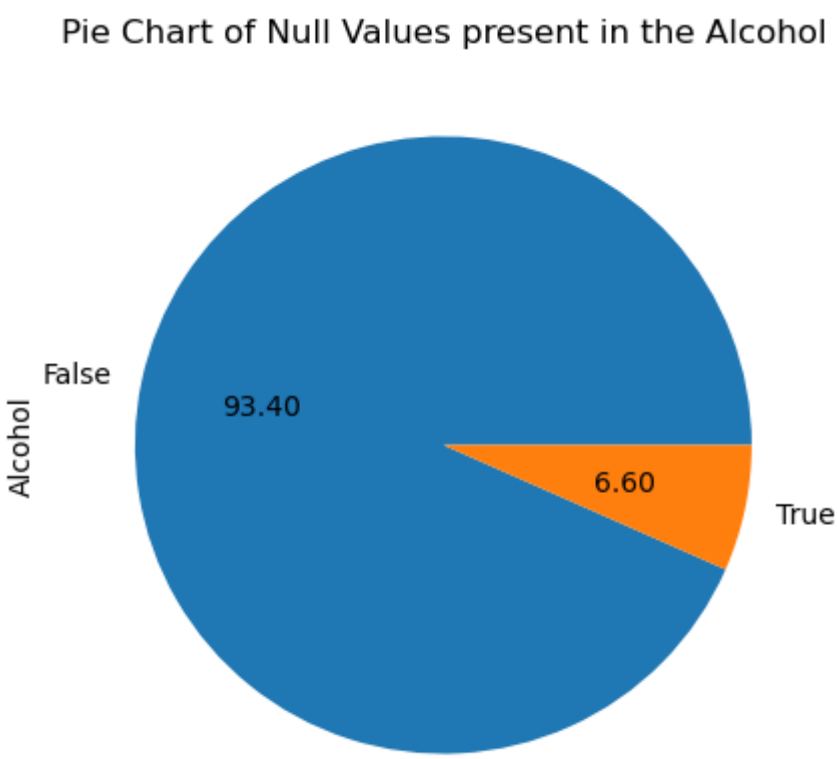
In [250…
```
df["Alcohol"].describe()
```

Out[250]:
```
count    2744.000000
mean        4.602861
std         4.052413
min         0.010000
25%         0.877500
50%         3.755000
75%         7.702500
max        17.870000
Name: Alcohol, dtype: float64
```

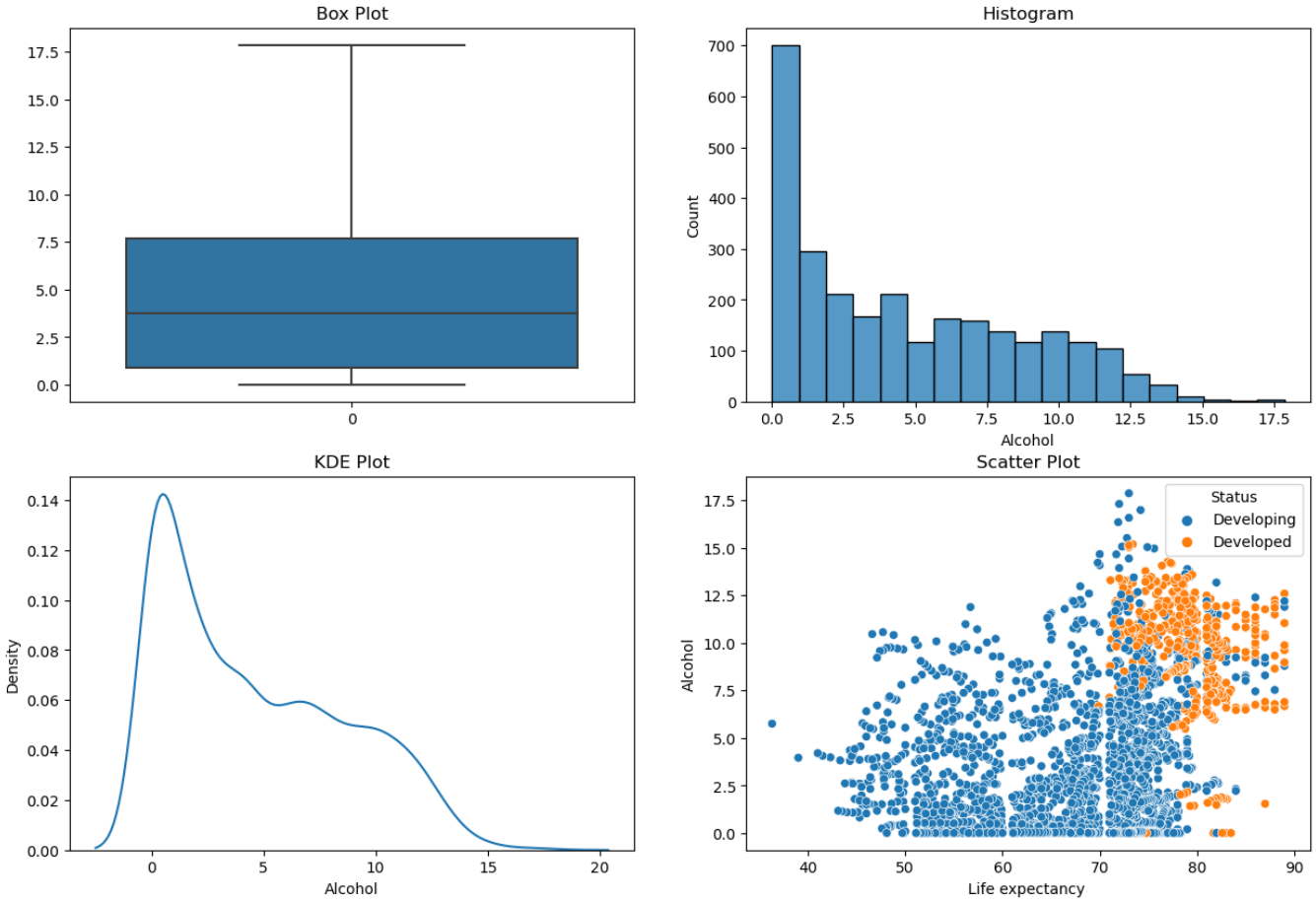
In [251…
```
df["Alcohol"].isnull().sum()
```

Out[251]:
```
194
```

In [252…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the Alcohol")
df["Alcohol"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="Alcohol")
plt.show()
```



In [253…
```
graph(df, "Alcohol")
```



Percentage expenditure (numerical)

In [254…
```
df["percentage expenditure"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: percentage expenditure
Non-Null Count  Dtype
--------------  -----
2938 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

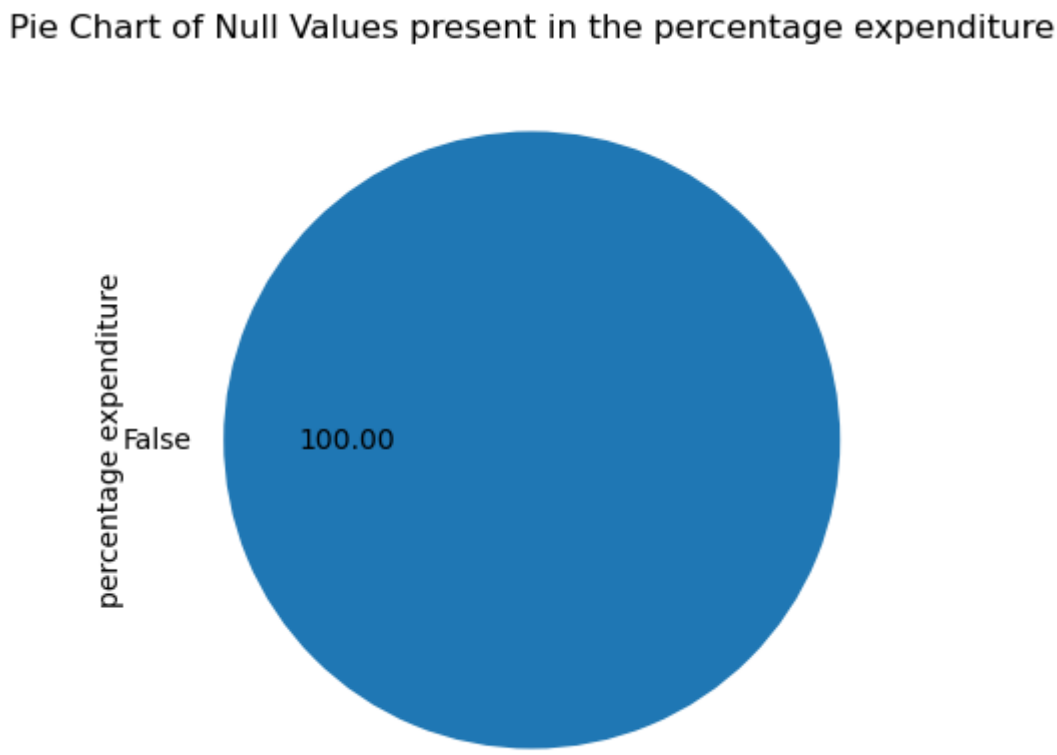
In [255…
```
df["percentage expenditure"].describe()
```

Out[255]:
```
count     2938.000000
mean       738.251295
std       1987.914858
min          0.000000
25%          4.685343
50%         64.912906
75%        441.534144
max      19479.911610
Name: percentage expenditure, dtype: float64
```

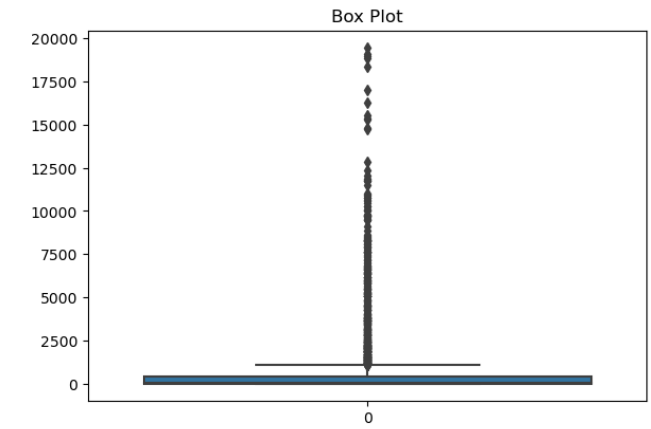
In [256…
```
df["percentage expenditure"].isnull().sum()
```

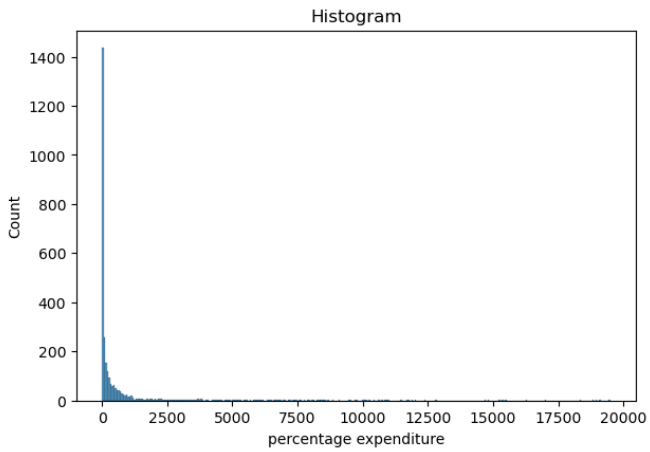
Out[256]:
```
0
```

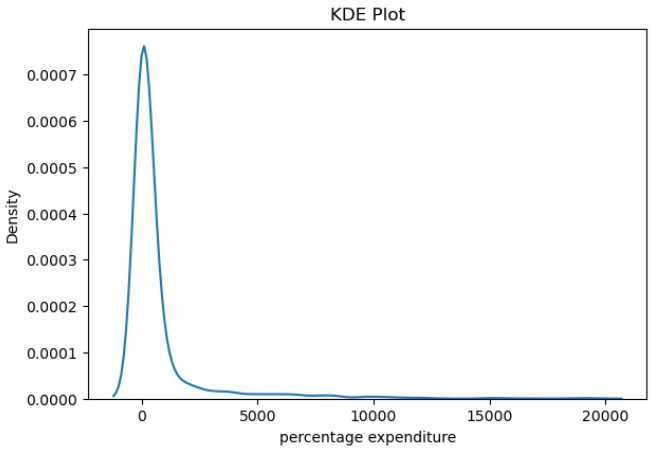
In [257…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the percentage expenditure")
df["percentage expenditure"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="percentage expenditure")
plt.show()
```

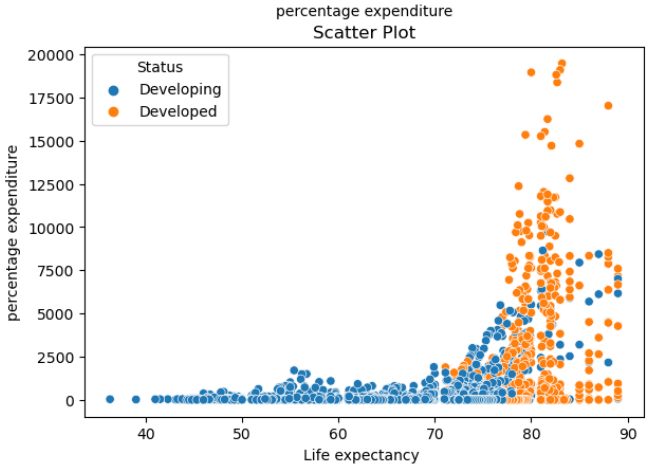


In [258…
```
graph(df, "percentage expenditure")
```









Hepatitis B (numerical)

In [259…
```
df["Hepatitis B"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: Hepatitis B
Non-Null Count  Dtype
--------------  -----
2385 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

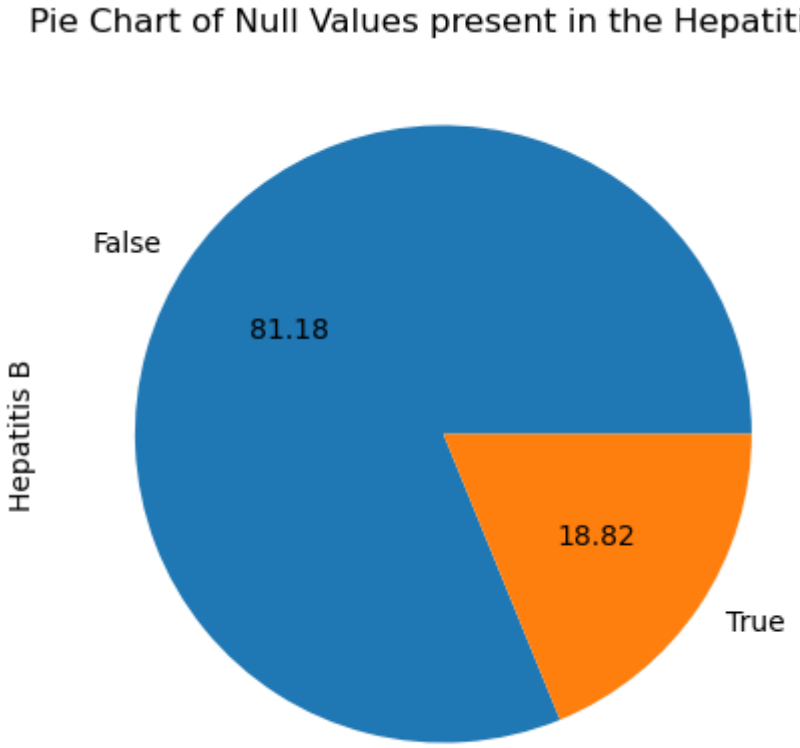
In [260…
```
df["Hepatitis B"].describe()
```

Out[260]:
```
count    2385.000000
mean       80.940461
std        25.070016
min         1.000000
25%        77.000000
50%        92.000000
75%        97.000000
max        99.000000
Name: Hepatitis B, dtype: float64
```

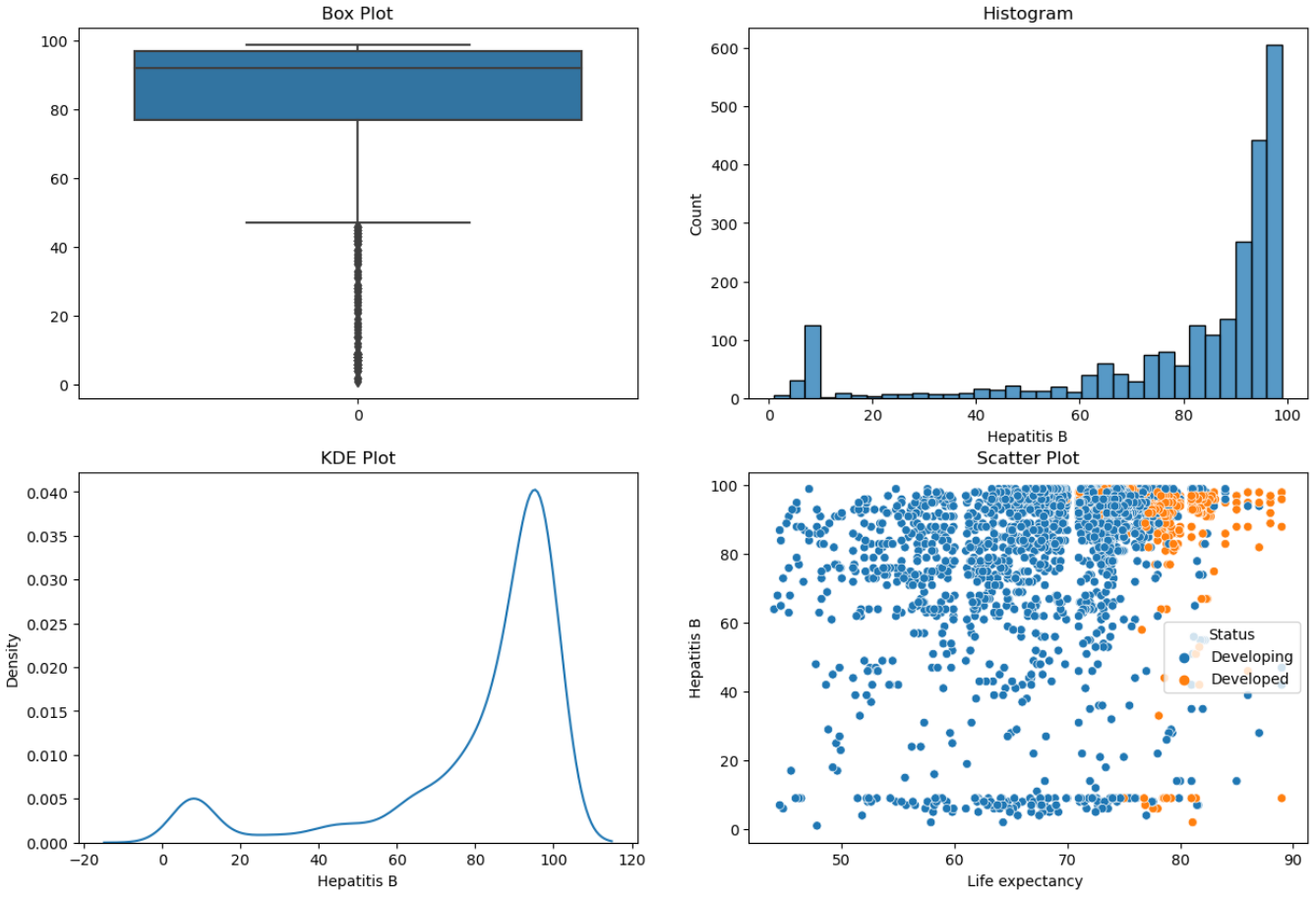
In [261…
```
df["Hepatitis B"].isnull().sum()
```

Out[261]:
```
553
```

In [262…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the Hepatitis B")
df["Hepatitis B"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="Hepatitis B")
plt.show()
```



In [263…
```
graph(df, "Hepatitis B")
```



Measles (numerical)

In [264…
```
df["Measles "].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: Measles
Non-Null Count  Dtype
--------------  -----
2938 non-null   int64
dtypes: int64(1)
memory usage: 23.1 KB
```

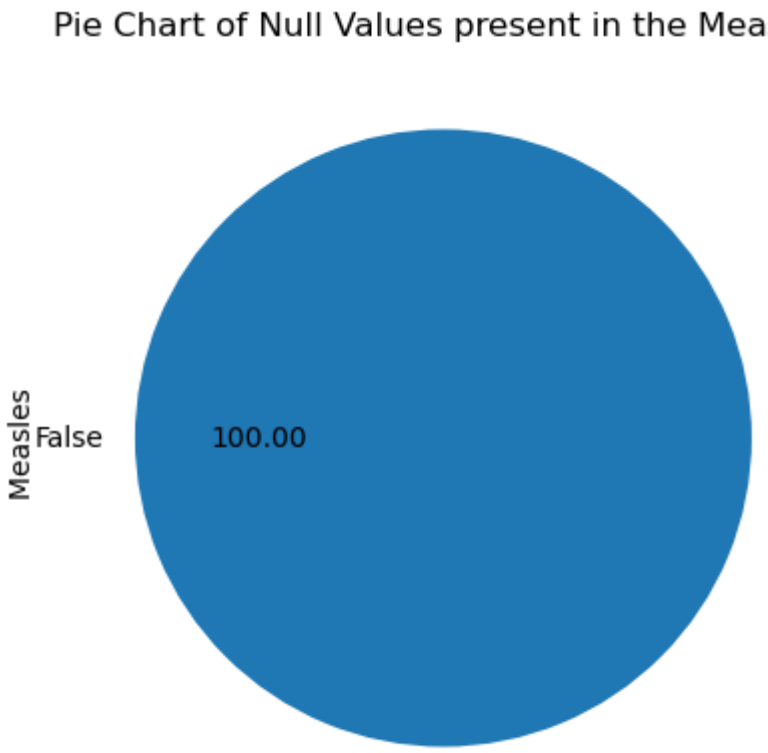
In [265…
```
df["Measles "].describe()
```

Out[265]:
```
count      2938.000000
mean       2419.592240
std       11467.272489
min           0.000000
25%           0.000000
50%          17.000000
75%         360.250000
max      212183.000000
Name: Measles , dtype: float64
```

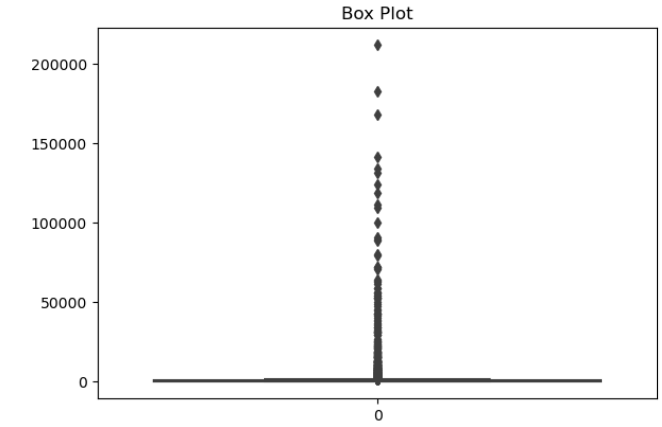
In [266…
```
df["Measles "].isnull().sum()
```

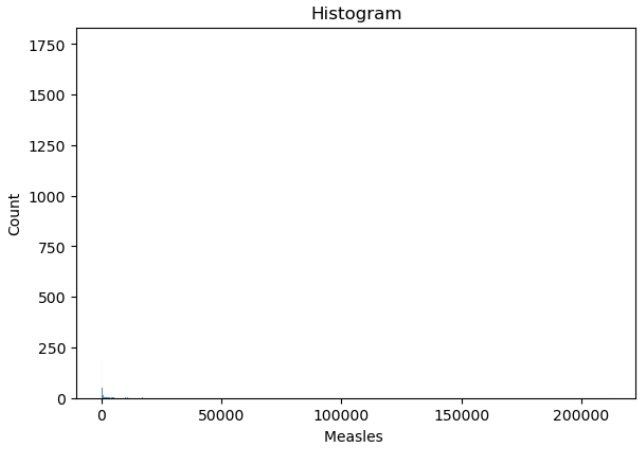
Out[266]:
```
0
```

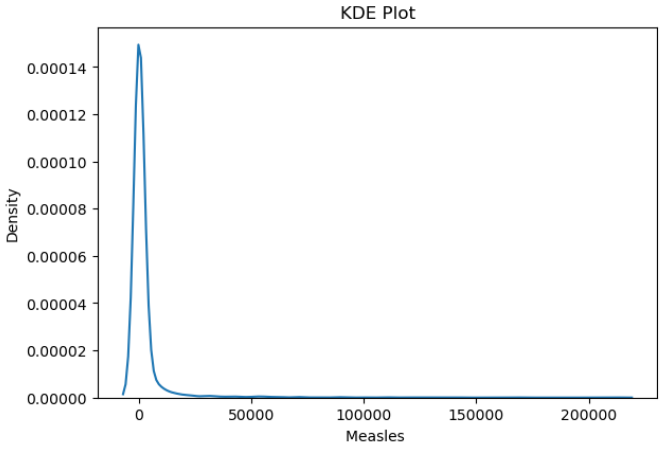
In [267…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the Measles")
df["Measles "].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="Measles ")
plt.show()
```

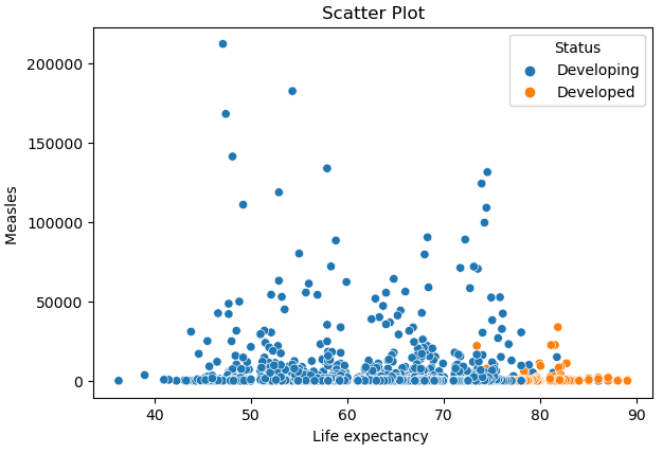


In [268…
```
graph(df, "Measles ")
```









BMI (numerical)

In [269…
```
df[" BMI "].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name:  BMI
Non-Null Count  Dtype
--------------  -----
2904 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

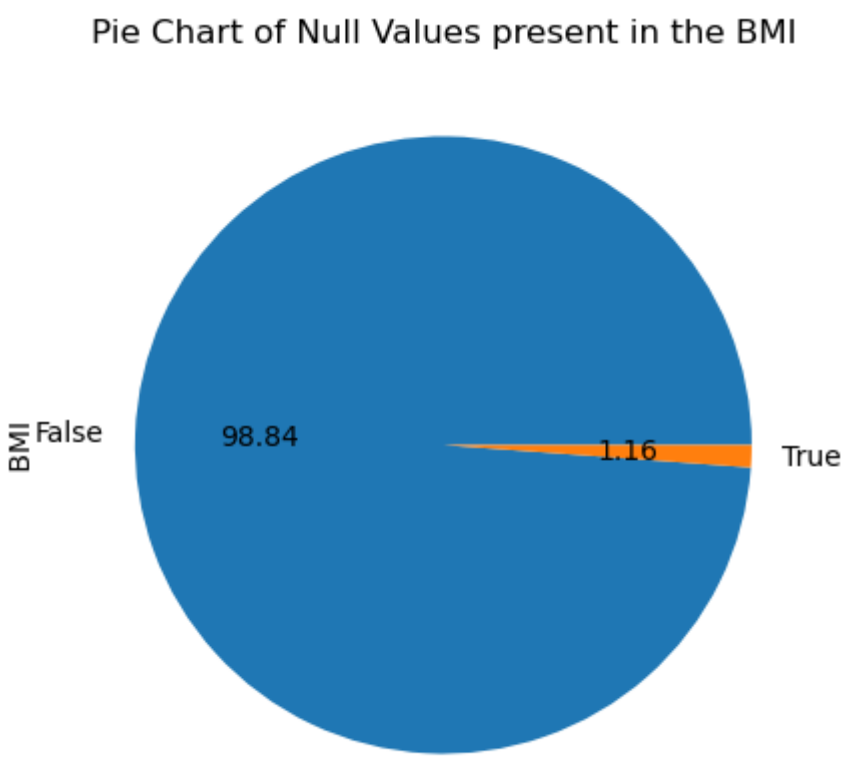
In [270…
```
df[" BMI "].describe()
```

Out[270]:
```
count    2904.000000
mean       38.321247
std        20.044034
min         1.000000
25%        19.300000
50%        43.500000
75%        56.200000
max        87.300000
Name:  BMI , dtype: float64
```

In [271…
```
df[" BMI "].isnull().sum()
```

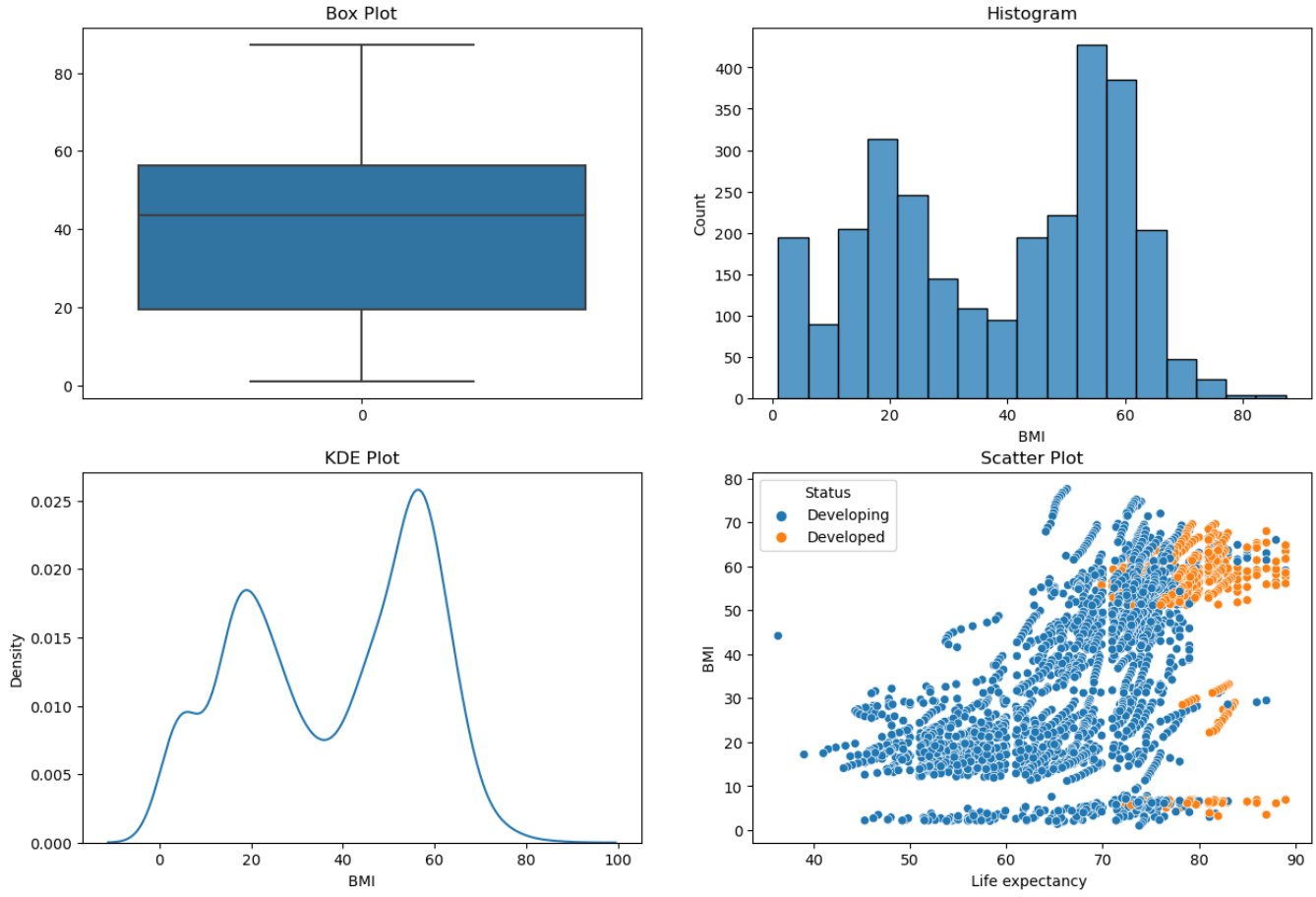
Out[271]:
```
34
```

In [272…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the BMI")
df[" BMI "].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x=" BMI ")
plt.show()
```



In [273…

```
graph(df, " BMI ")
```



Under-five deaths (numerical)

In [274…

```
df["under-five deaths "].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: under-five deaths
Non-Null Count  Dtype
--------------  -----
2938 non-null   int64
dtypes: int64(1)
memory usage: 23.1 KB
```

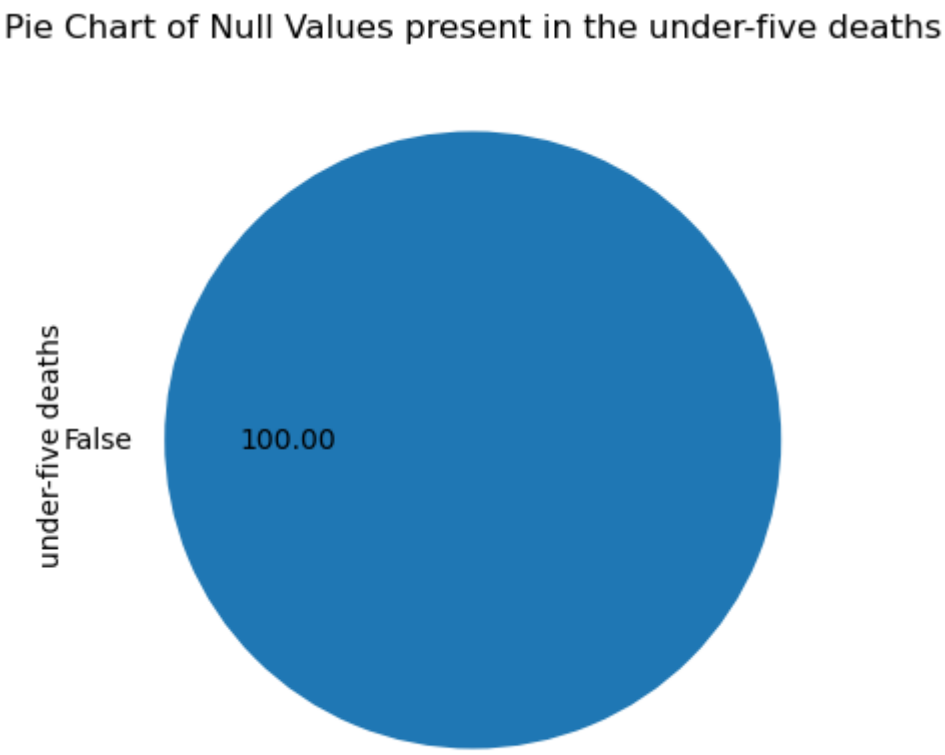
In [275…
```
df["under-five deaths "].describe()
```

Out[275]:
```
count    2938.000000
mean       42.035739
std       160.445548
min         0.000000
25%         0.000000
50%         4.000000
75%        28.000000
max      2500.000000
Name: under-five deaths , dtype: float64
```

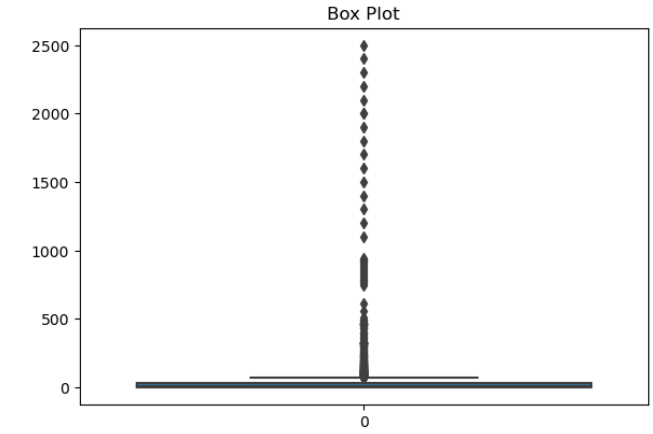
In [276…
```
df["under-five deaths "].isnull().sum()
```

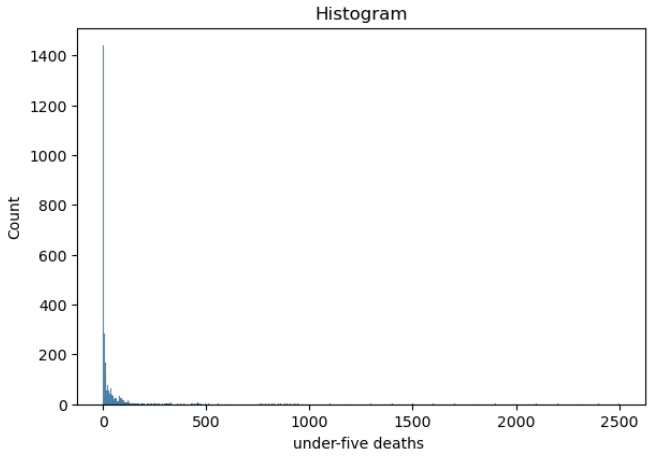
Out[276]:
```
0
```

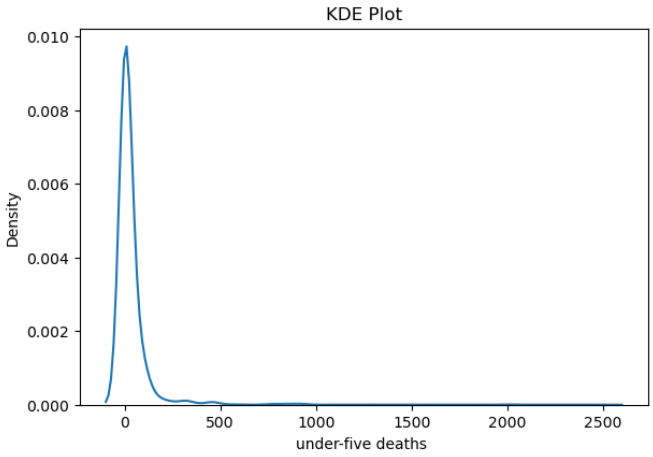
In [277…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the under-five deaths")
df["under-five deaths "].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="under-five deaths ")
plt.show()
```

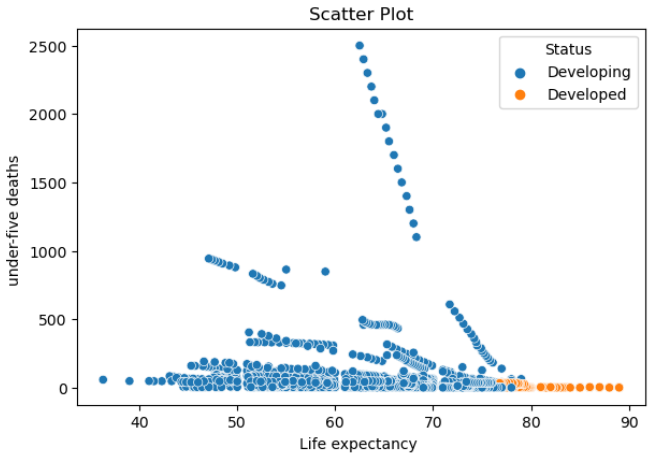


In [278…
```
graph(df, "under-five deaths ")
```









Polio (numerical)

In [279…
```
df["Polio"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: Polio
Non-Null Count  Dtype
--------------  -----
2919 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

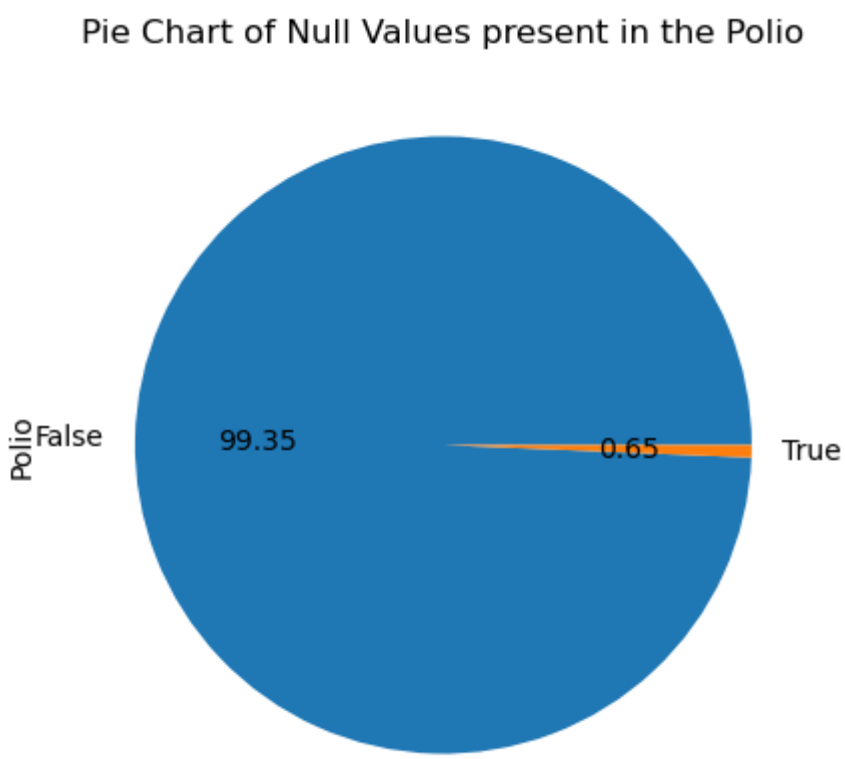
In [280…
```
df["Polio"].describe()
```

Out[280]:
```
count    2919.000000
mean       82.550188
std        23.428046
min         3.000000
25%        78.000000
50%        93.000000
75%        97.000000
max        99.000000
Name: Polio, dtype: float64
```

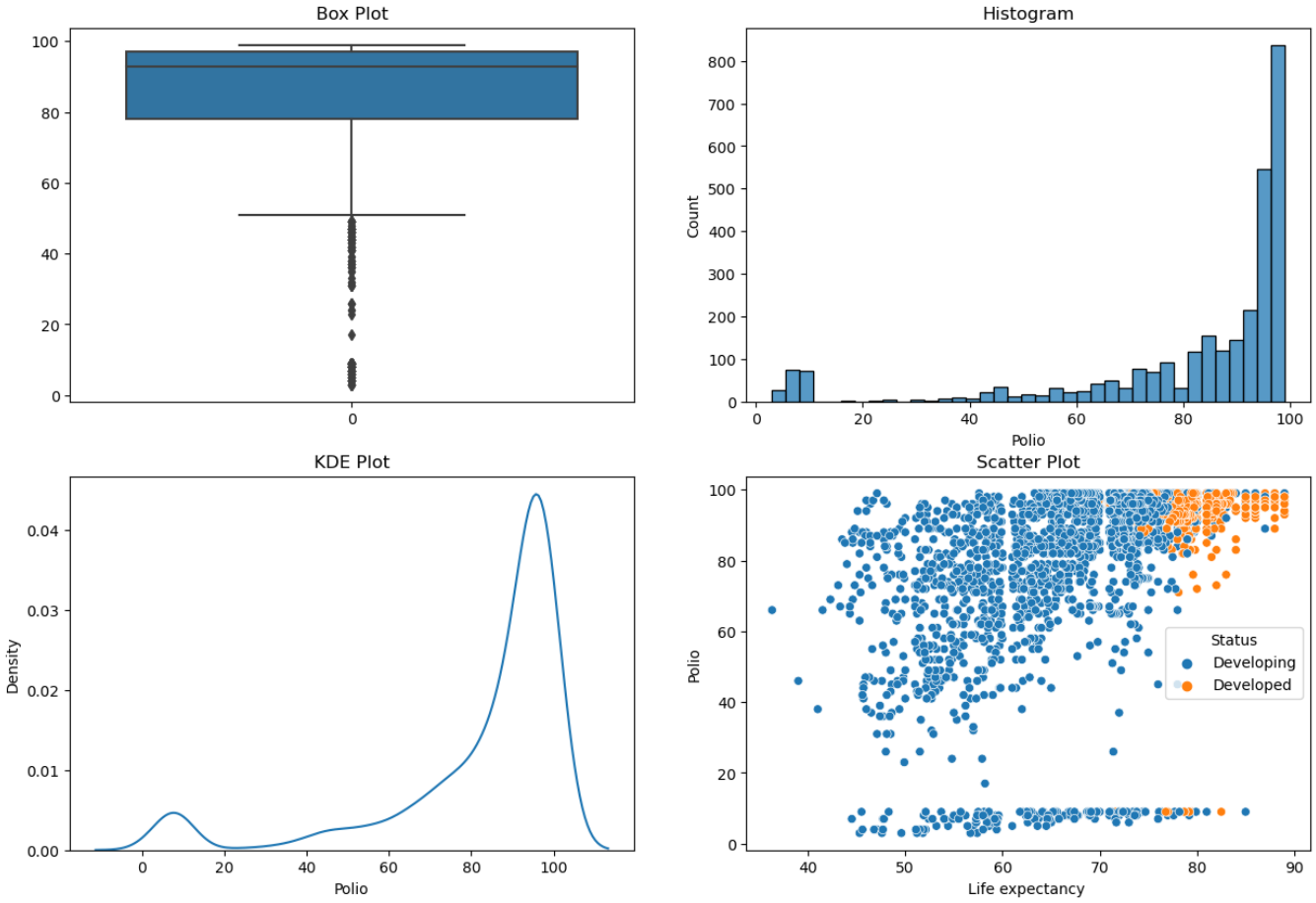
In [281…
```
df["Polio"].isnull().sum()
```

Out[281]:
```
19
```

In [282…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the Polio")
df["Polio"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="Polio")
plt.show()
```



In [283…
```
graph(df, "Polio")
```



Total expenditure (numerical)

In [284…
```
df["Total expenditure"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: Total expenditure
Non-Null Count  Dtype
--------------  -----
2712 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

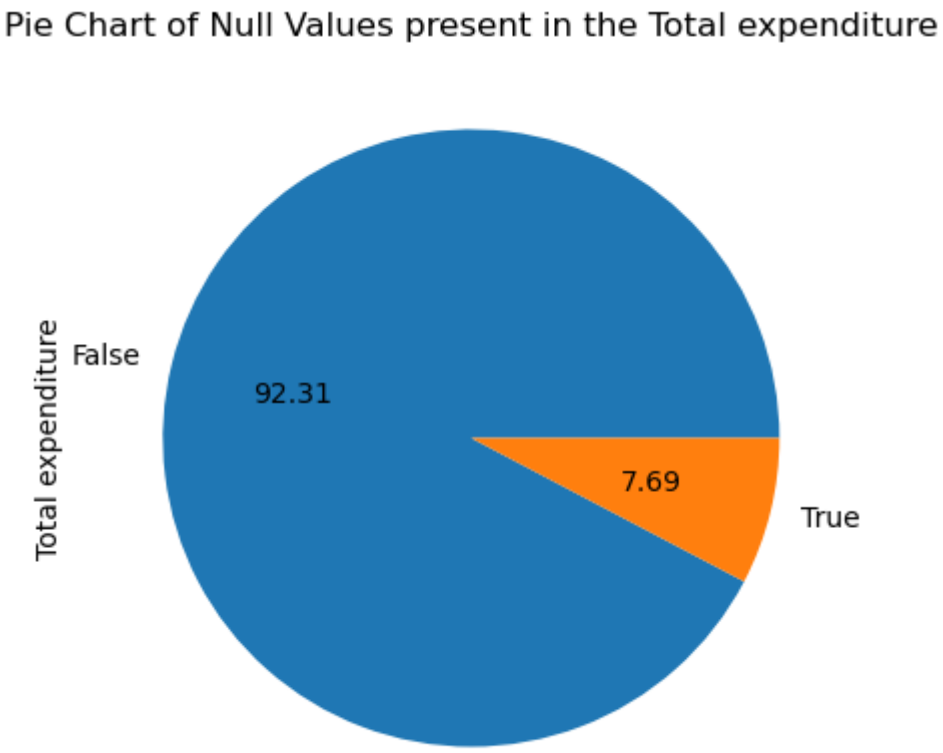
In [285…
```
df["Total expenditure"].describe()
```

Out[285]:
```
count    2712.00000
mean        5.93819
std         2.49832
min         0.37000
25%         4.26000
50%         5.75500
75%         7.49250
max        17.60000
Name: Total expenditure, dtype: float64
```

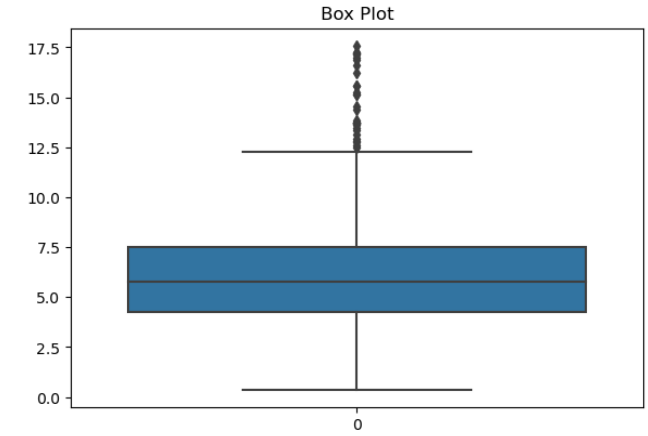
In [286…
```
df["Total expenditure"].isnull().sum()
```

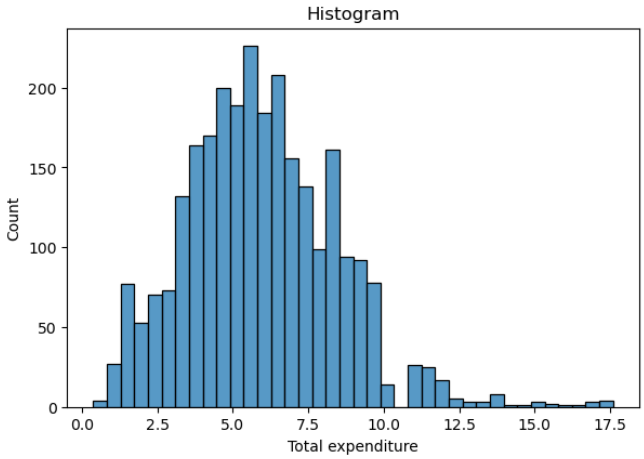
Out[286]:
```
226
```

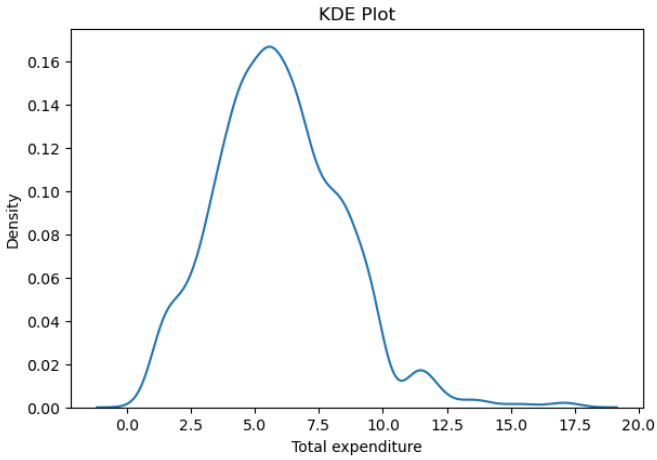
In [287…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the Total expenditure")
df["Total expenditure"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="Total expenditure")
plt.show()
```

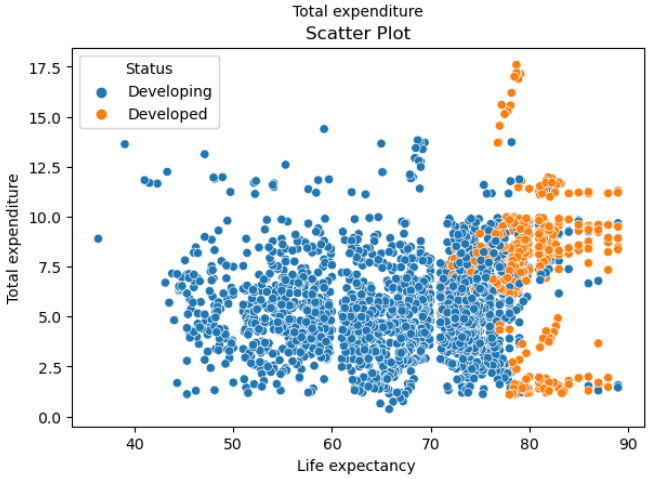


In [288…
```
graph(df, "Total expenditure")
```









Diphtheria (numerical)

In [289…
```
df["Diphtheria "].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: Diphtheria
Non-Null Count  Dtype
--------------  -----
2919 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

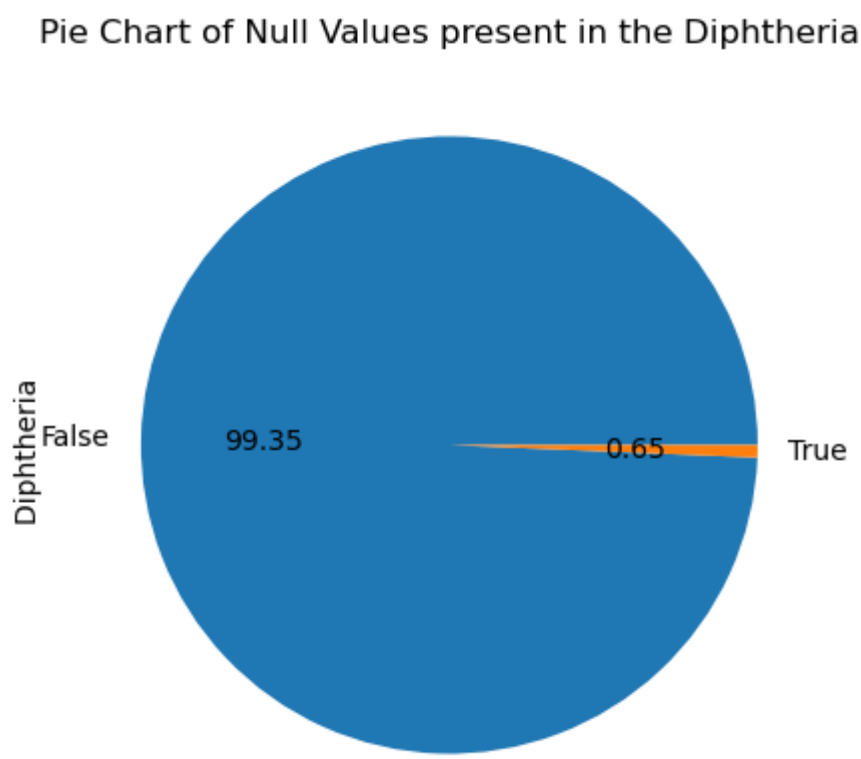
In [290…
```
df["Diphtheria "].describe()
```

Out[290]:
```
count    2919.000000
mean       82.324084
std        23.716912
min         2.000000
25%        78.000000
50%        93.000000
75%        97.000000
max        99.000000
Name: Diphtheria , dtype: float64
```

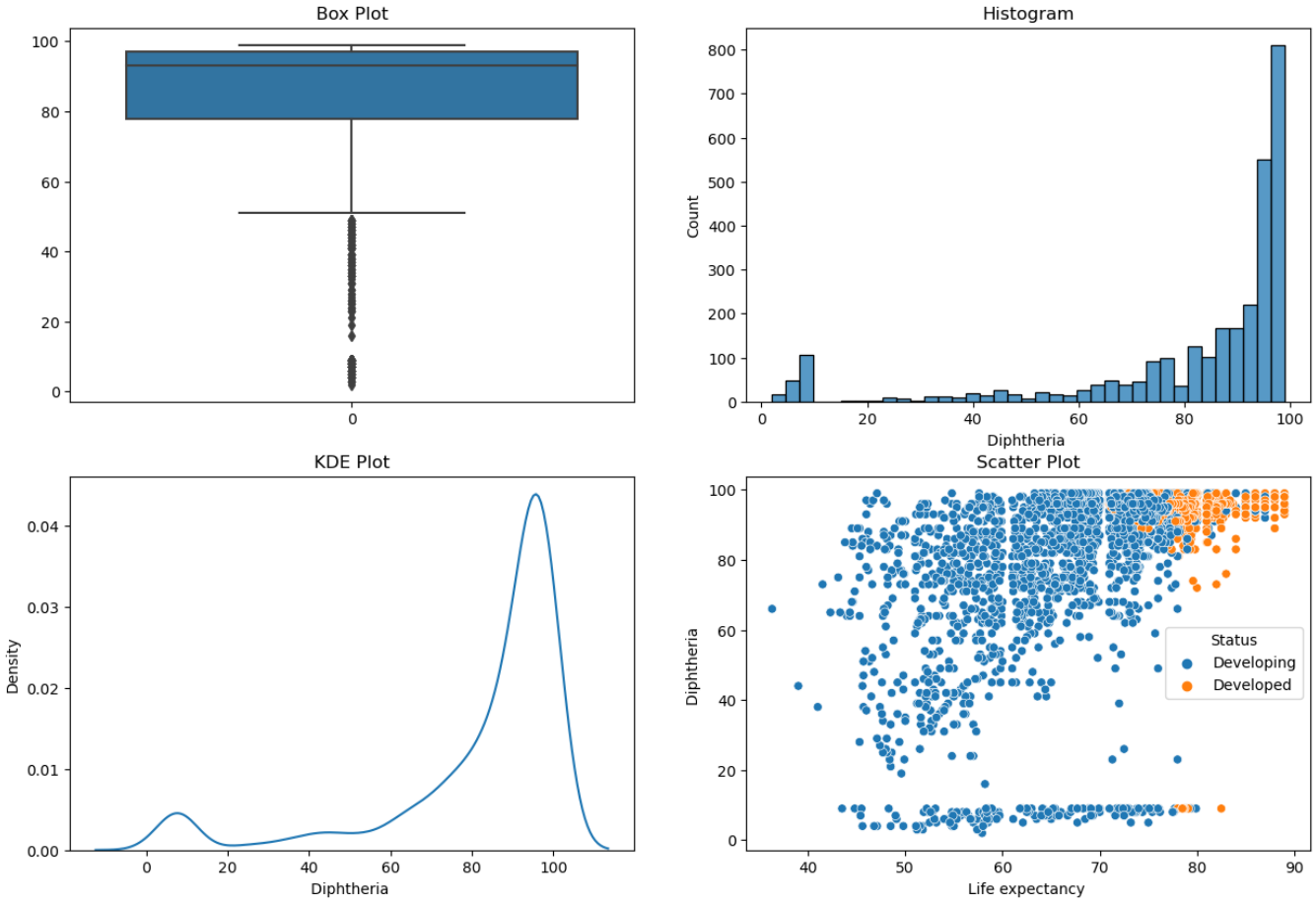
In [291…
```
df["Diphtheria "].isnull().sum()
```

Out[291]:
```
19
```

In [292…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the Diphtheria")
df["Diphtheria "].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="Diphtheria ")
plt.show()
```



In [293…
```
graph(df, "Diphtheria ")
```



HIV/AIDS (numerical)

In [294…
```
df[" HIV/AIDS"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name:  HIV/AIDS
Non-Null Count  Dtype
--------------  -----
2938 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

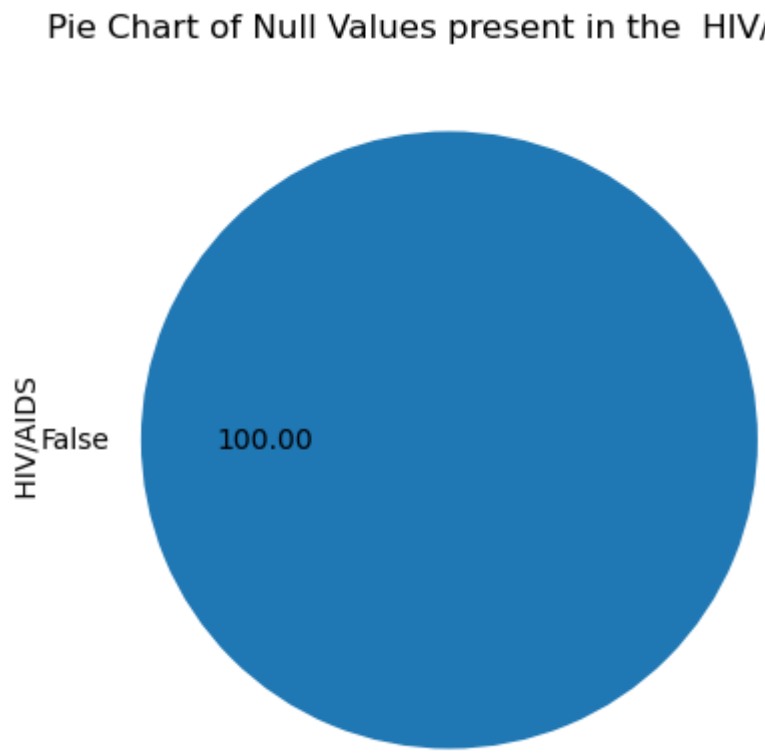
In [295…
```
df[" HIV/AIDS"].describe()
```

Out[295]:
```
count    2938.000000
mean        1.742103
std         5.077785
min         0.100000
25%         0.100000
50%         0.100000
75%         0.800000
max        50.600000
Name:  HIV/AIDS, dtype: float64
```

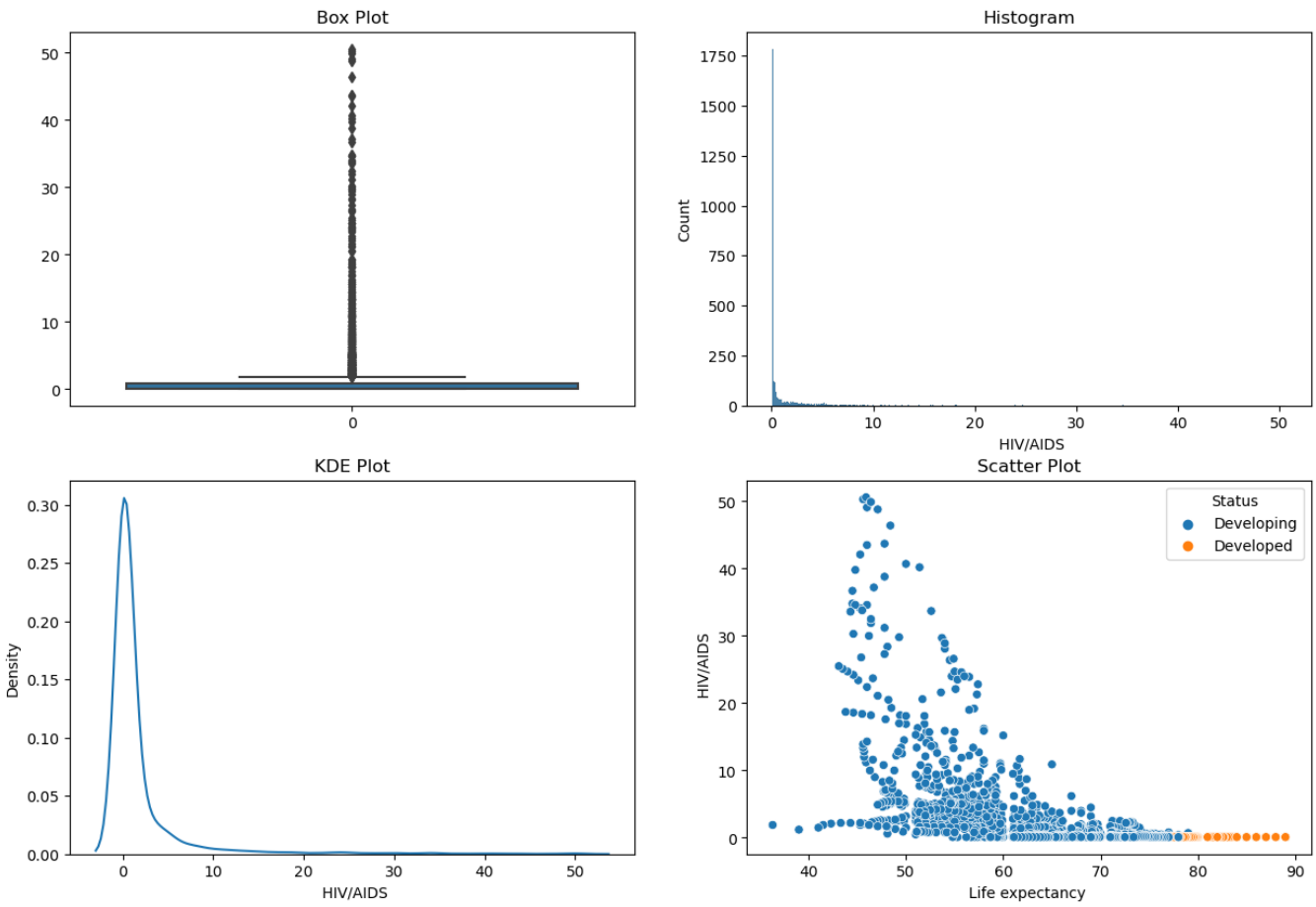
In [296…
```
df[" HIV/AIDS"].isnull().sum()
```

Out[296]:
```
0
```

In [297…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the  HIV/AIDS")
df[" HIV/AIDS"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x=" HIV/AIDS")
plt.show()
```



In [298…
```
graph(df, " HIV/AIDS")
```



GDP (numerical)

In [299…
```
df["GDP"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: GDP
Non-Null Count  Dtype
--------------  -----
2490 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

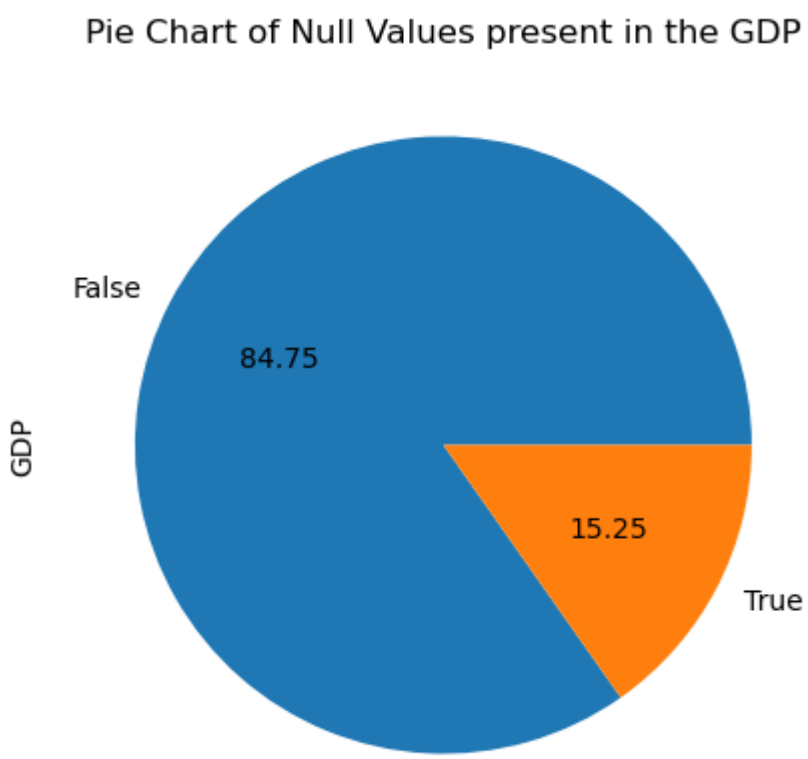
In [300…
```
df["GDP"].describe()
```

Out[300]:
```
count      2490.000000
mean       7483.158469
std       14270.169342
min           1.681350
25%         463.935626
50%        1766.947595
75%        5910.806335
max      119172.741800
Name: GDP, dtype: float64
```

In [301…
```
df["GDP"].isnull().sum()
```

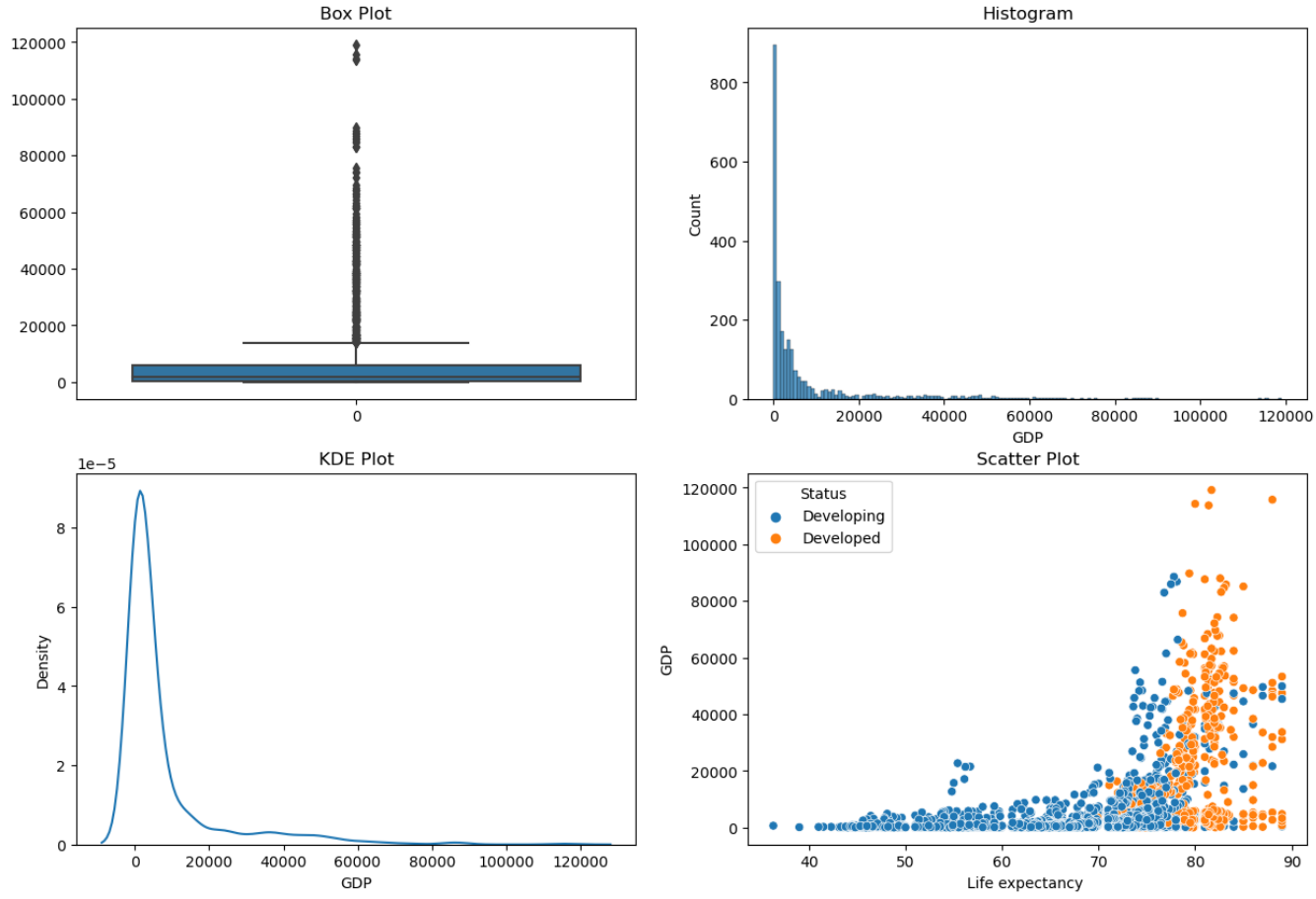
Out[301]:
```
448
```

In [302…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the GDP")
df["GDP"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="GDP")
plt.show()
```



In [303…

```
graph(df, "GDP")
```



Population (numerical)

In [304…

```
df["Population"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: Population
Non-Null Count  Dtype
--------------  -----
2286 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

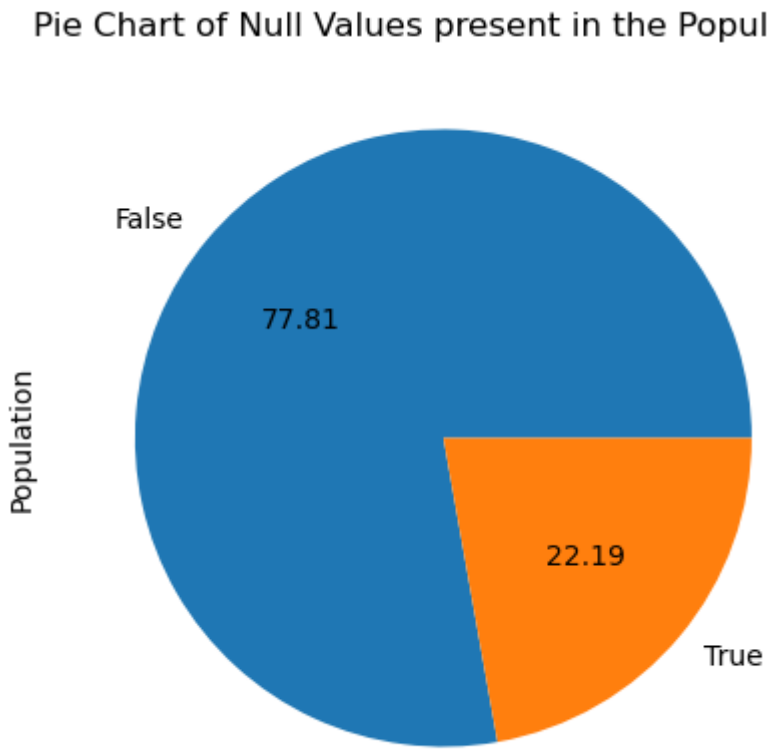
In [305…
```
df["Population"].describe()
```

Out[305]:
```
count    2.286000e+03
mean     1.275338e+07
std      6.101210e+07
min      3.400000e+01
25%      1.957932e+05
50%      1.386542e+06
75%      7.420359e+06
max      1.293859e+09
Name: Population, dtype: float64
```

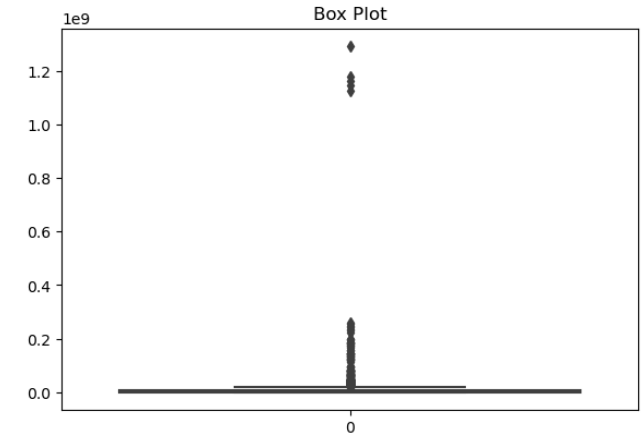
In [306…
```
df["Population"].isnull().sum()
```

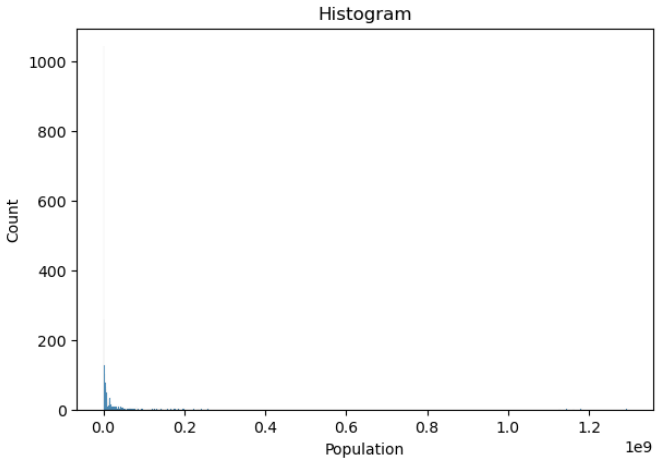
Out[306]:
```
652
```

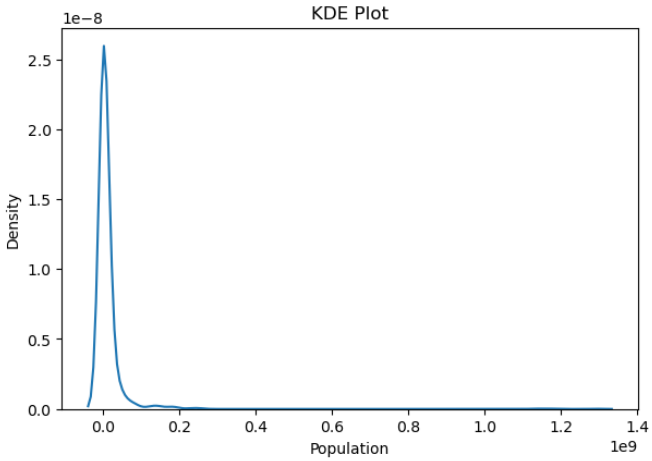
In [307…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the Population")
df["Population"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="Population")
plt.show()
```

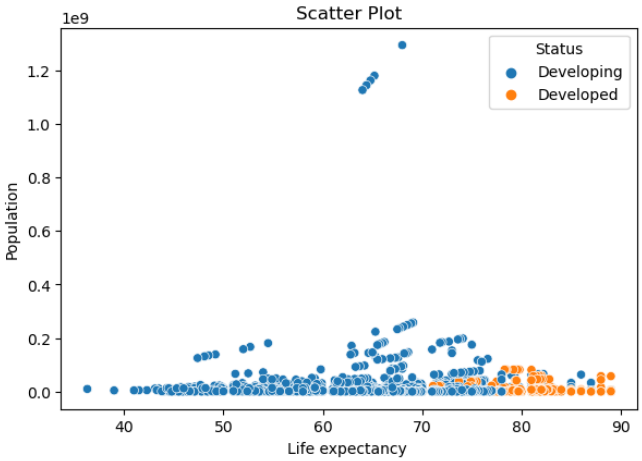


In [308…
```
graph(df, "Population")
```









Thinness 1-19 years (numerical)

In [309…
```
df[" thinness  1-19 years"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name:  thinness  1-19 years
Non-Null Count  Dtype
--------------  -----
2904 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

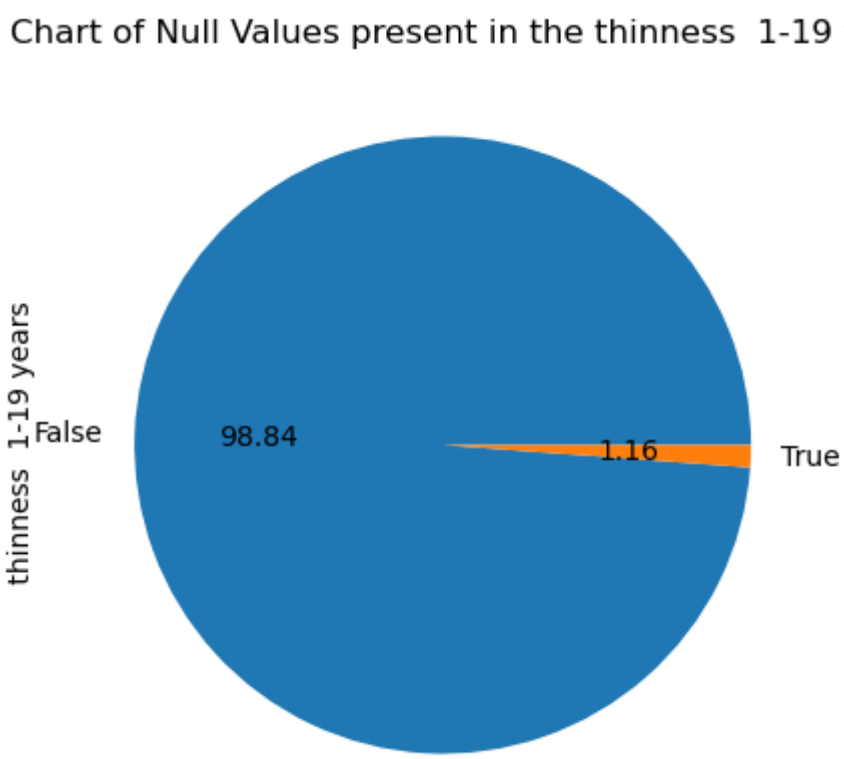
In [310…
```
df[" thinness  1-19 years"].describe()
```

Out[310]:
```
count    2904.000000
mean        4.839704
std         4.420195
min         0.100000
25%         1.600000
50%         3.300000
75%         7.200000
max        27.700000
Name:  thinness  1-19 years, dtype: float64
```

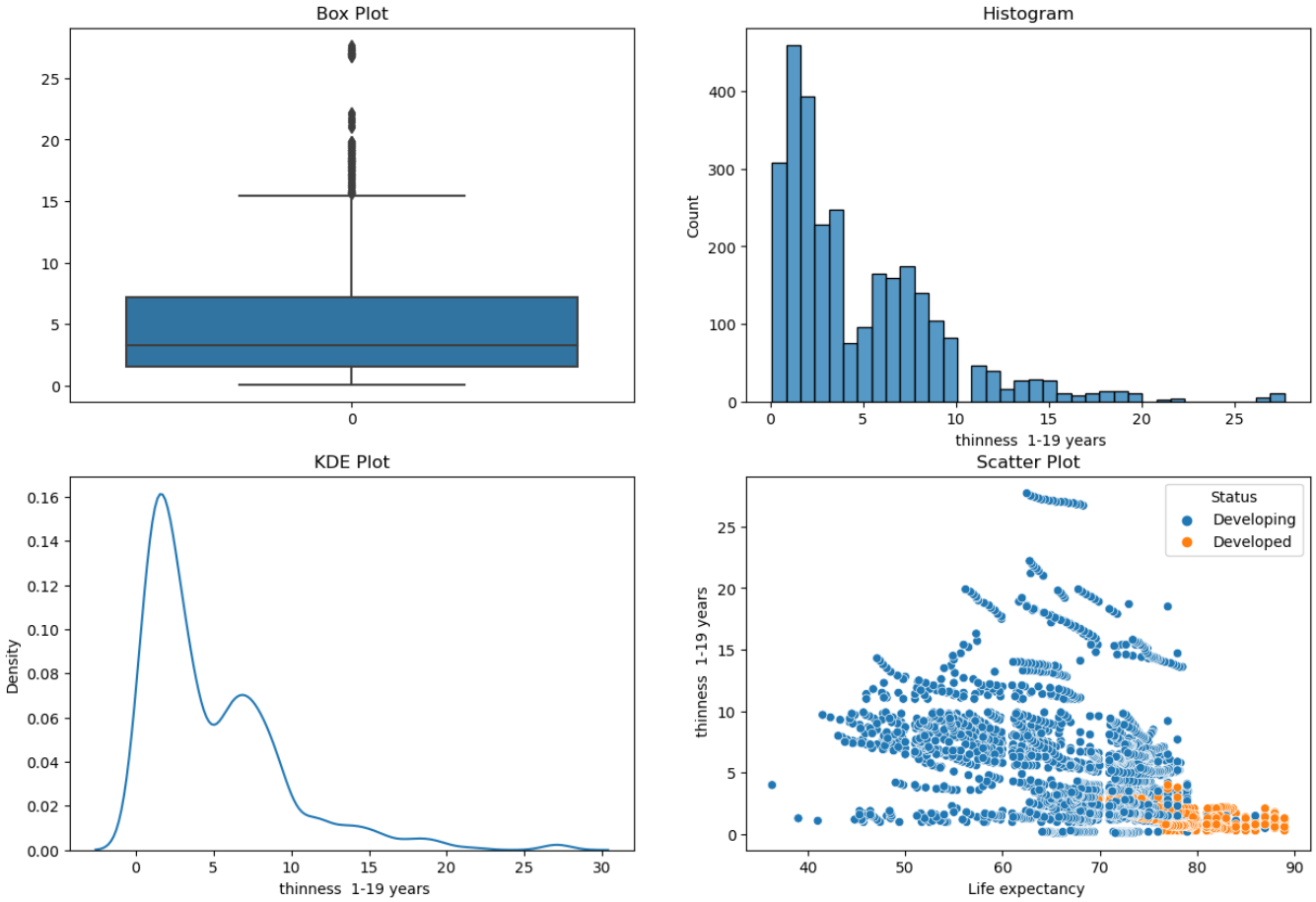
In [311…
```
df[" thinness  1-19 years"].isnull().sum()
```

Out[311]:
```
34
```

In [312…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the thinness  1-19 years")
df[" thinness  1-19 years"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x=" thinness  1-19 years")
plt.show()
```



In [313…
```
graph(df, " thinness  1-19 years")
```



Thinness 5-9 years (numerical)

In [314…
```
df[" thinness 5-9 years"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name:  thinness 5-9 years
Non-Null Count  Dtype
--------------  -----
2904 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

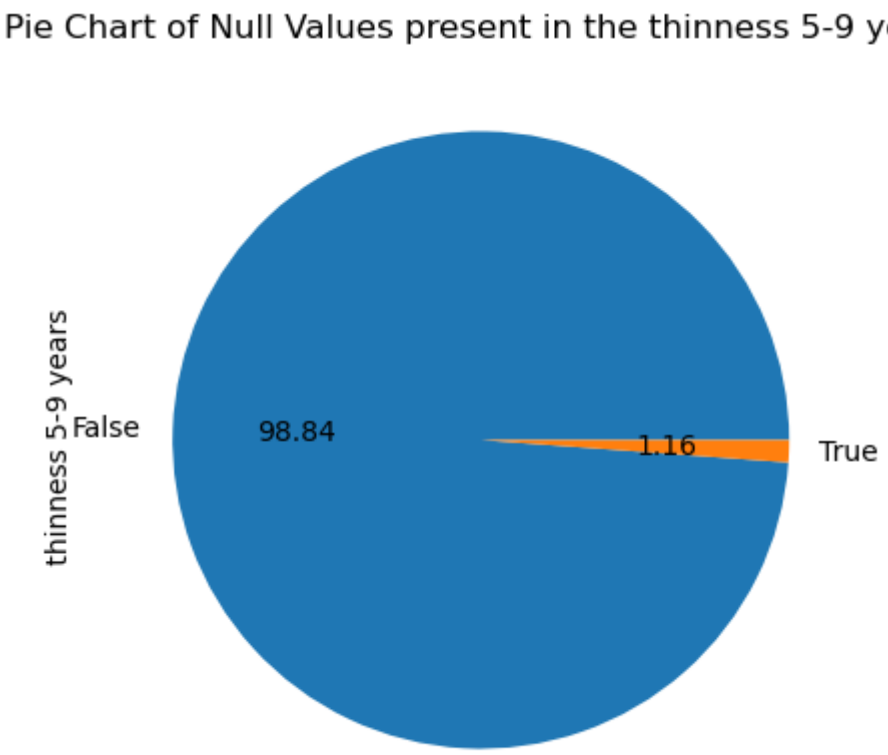
In [315…
```
df[" thinness 5-9 years"].describe()
```

Out[315]:
```
count    2904.000000
mean        4.870317
std         4.508882
min         0.100000
25%         1.500000
50%         3.300000
75%         7.200000
max        28.600000
Name:  thinness 5-9 years, dtype: float64
```

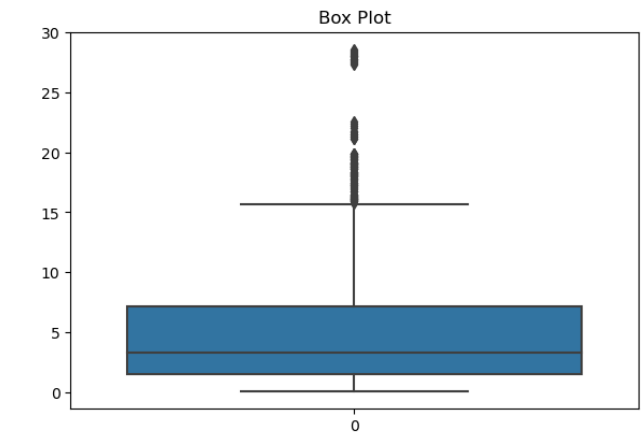
In [316…
```
df[" thinness 5-9 years"].isnull().sum()
```

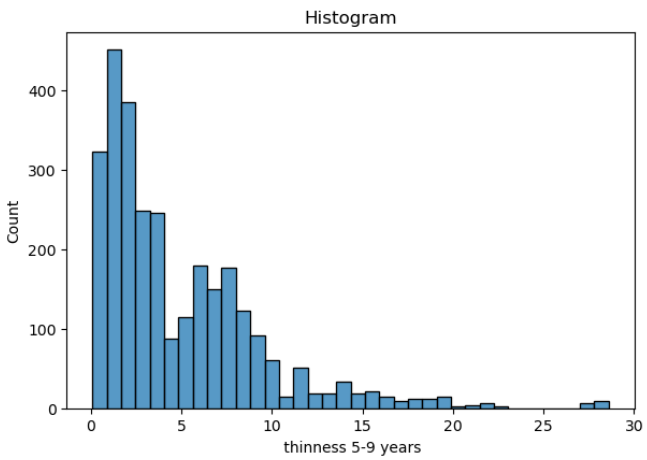
Out[316]:
```
34
```

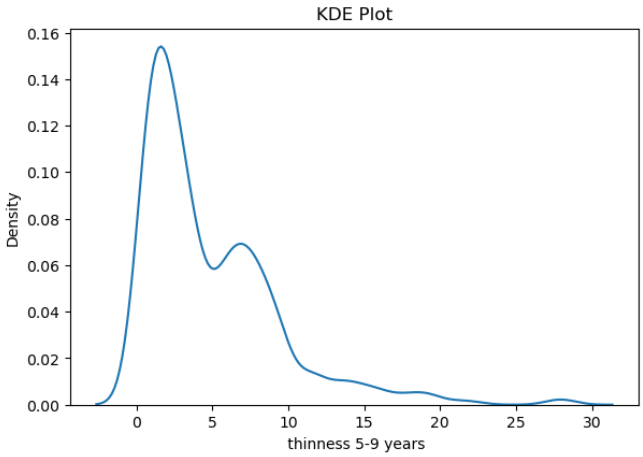
In [317…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the thinness 5-9 years")
df[" thinness 5-9 years"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x=" thinness 5-9 years")
plt.show()
```

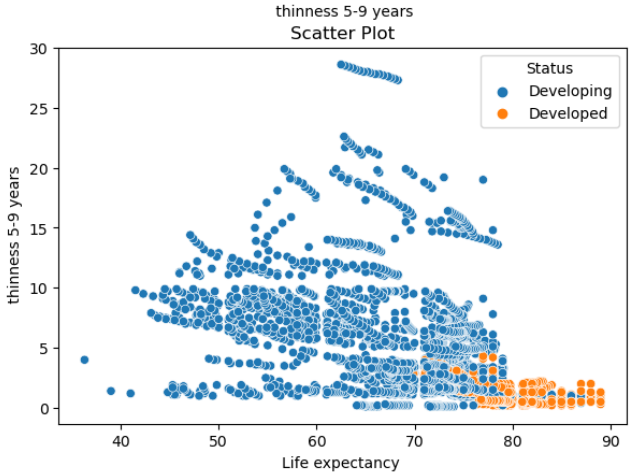


In [318…
```
graph(df, " thinness 5-9 years")
```









Income composition of resources (numerical)

In [319…
```
df["Income composition of resources"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: Income composition of resources
Non-Null Count  Dtype
--------------  -----
2771 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

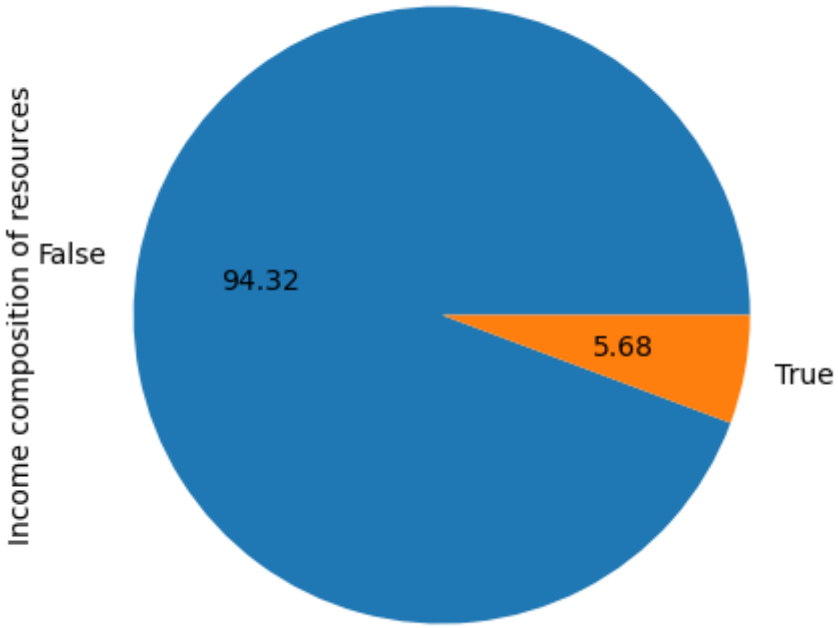
In [320…
```
df["Income composition of resources"].describe()
```

Out[320]:
```
count    2771.000000
mean        0.627551
std         0.210904
min         0.000000
25%         0.493000
50%         0.677000
75%         0.779000
max         0.948000
Name: Income composition of resources, dtype: float64
```

In [321…
```
df["Income composition of resources"].isnull().sum()
```

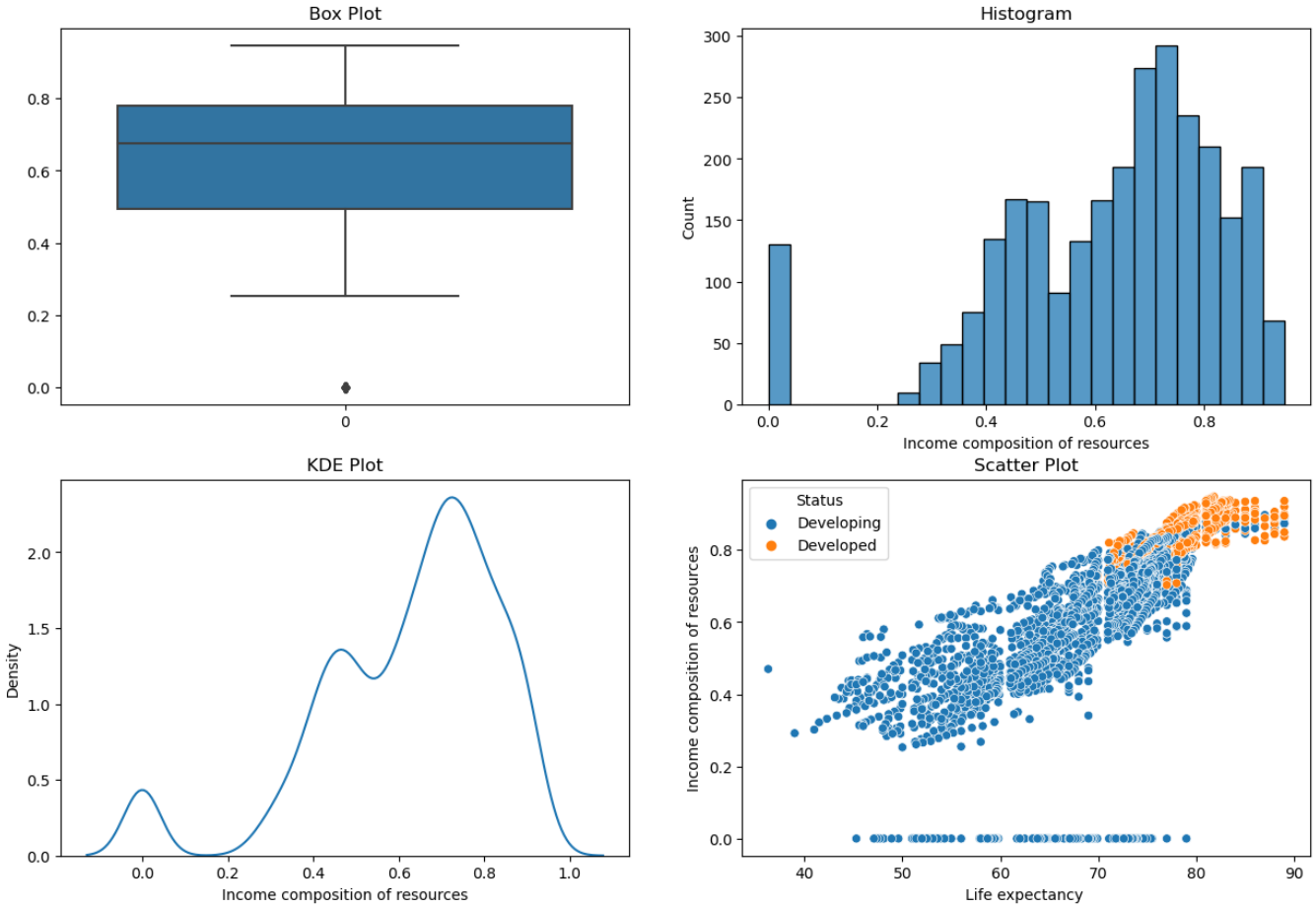
Out[321]:
```
167
```

In [322…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the Income composition of resources")
df["Income composition of resources"].isnull().value_counts().plot(kind="pie", autopct="%.2f"
# sns.countplot(data=df.isnull(), x="Income composition of resources")
plt.show()
```



In [323…

```
graph(df, "Income composition of resources")
```



Schooling (numerical)

In [324…

```
df["Schooling"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2938 entries, 0 to 2937
Series name: Schooling
Non-Null Count  Dtype
--------------  -----
2775 non-null   float64
dtypes: float64(1)
memory usage: 23.1 KB
```

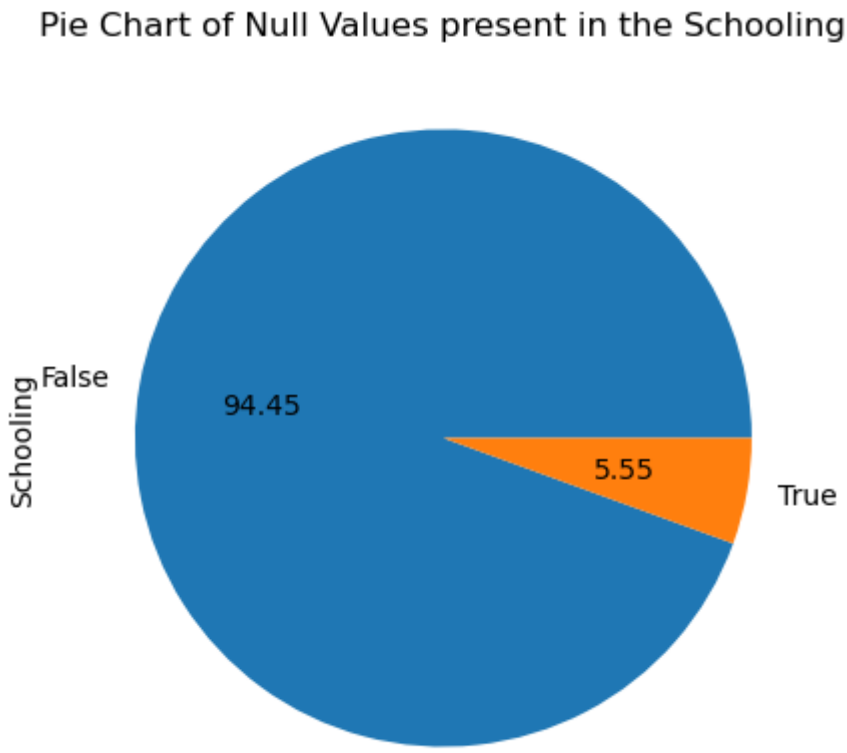
In [325…
```
df["Schooling"].describe()
```

Out[325]:
```
count    2775.000000
mean       11.992793
std         3.358920
min         0.000000
25%        10.100000
50%        12.300000
75%        14.300000
max        20.700000
Name: Schooling, dtype: float64
```

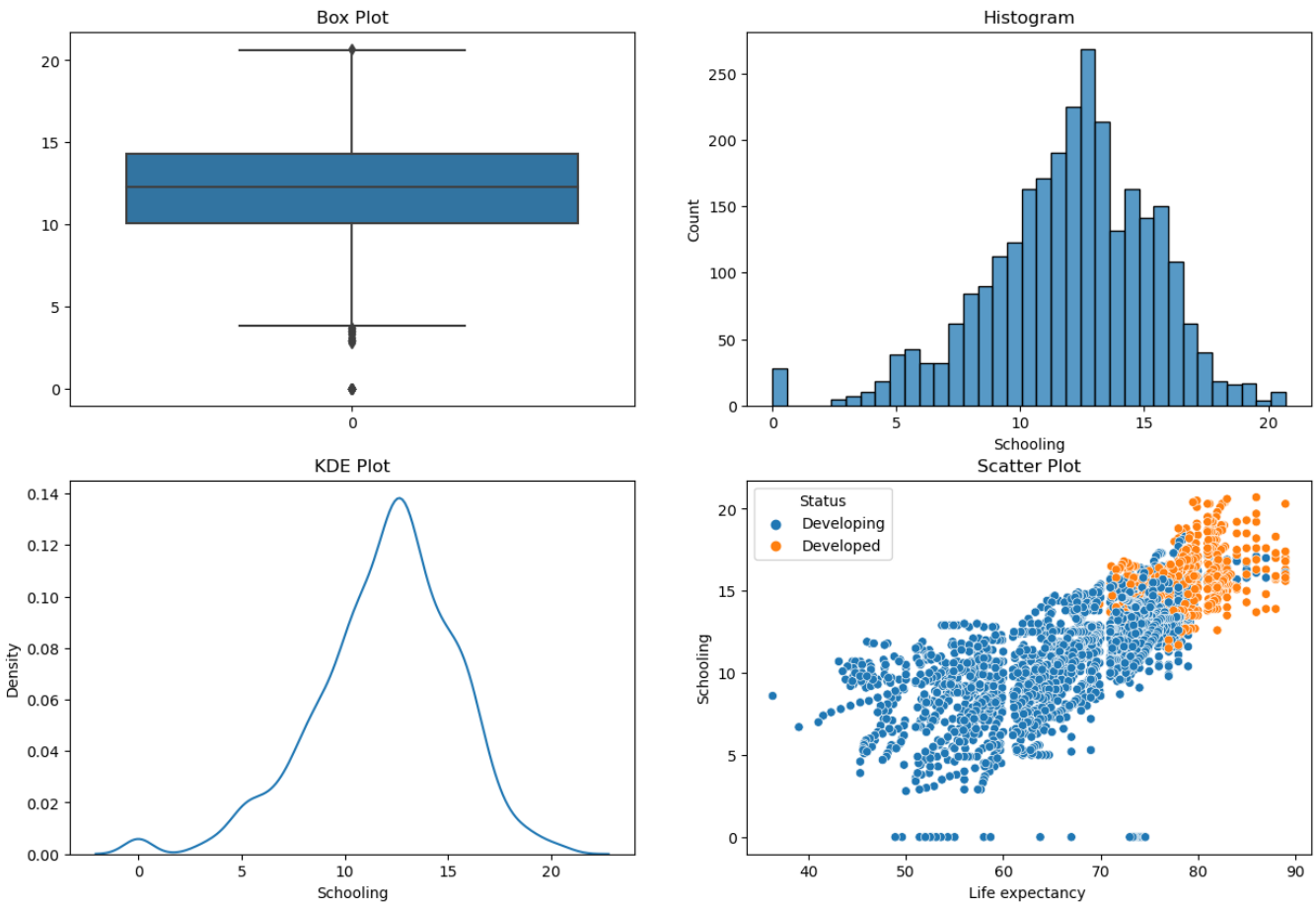
In [326…
```
df["Schooling"].isnull().sum()
```

Out[326]:
```
163
```

In [327…
```
plt.figure(figsize=(7,5))
plt.title("Pie Chart of Null Values present in the Schooling")
df["Schooling"].isnull().value_counts().plot(kind="pie", autopct="%.2f")
# sns.countplot(data=df.isnull(), x="Schooling")
plt.show()
```



In [328…
```
graph(df, "Schooling")
```



In [329…
```
df_new = df.copy()
```

Feature selection

Selected Columns

- **Country**
- **Year**
- **Staus**
- **Life expectancy**
- **Adult Mortality**
- **infant deaths**
- **Alcohol**
- **Percentage expenditure**
- **Measles**
- **BMI**
- **under-five deaths**
- **Polio**
- **Diphtheria**
- **HIV/AIDS**
- **GDP**
- **Thinness 1-19 years**
- **Income composition of resources**
- **Schooling**

In [330…
```
df_new = df_new.drop(['Population', 'Hepatitis B', ' thinness 5-9 years', 'Total expenditure'
```

In [331…
```
df_new.head()
```

Out[331]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Measles	BMI	under fiv death
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	1154	19.1	8
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	492	18.6	8
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	430	18.1	8
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	2787	17.6	9
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	3013	17.2	9

Filling & removing the missing values

Removing

In [332…]
```
col_drop = [i for i in df_new.columns if df_new[i].isnull().mean() < 0.05 and df_new[i].isnul
col_drop
```

Out[332]:
```
['Life expectancy ',
 'Adult Mortality',
 ' BMI ',
 'Polio',
 'Diphtheria ',
 ' thinness  1-19 years']
```

In [333…]
```
len(df_new[col_drop].dropna()) / len(df_new)
```

Out[333]:
```
0.9829816201497618
```

In [334…]
```
df_new_droped = df_new.dropna(subset=col_drop)
```

In [335…]
```
df_new_droped.isnull().sum()
```

Out[335]:
```
Country                            0
Year                               0
Status                             0
Life expectancy                    0
Adult Mortality                    0
infant deaths                      0
Alcohol                          175
percentage expenditure             0
Measles                            0
 BMI                               0
under-five deaths                  0
Polio                              0
Diphtheria                         0
 HIV/AIDS                          0
GDP                              435
 thinness  1-19 years              0
Income composition of resources  160
Schooling                        160
dtype: int64
```

Filling

In [336…]
```
median_Alcohol = df_new_droped["Alcohol"].median()
median_GDP = df_new_droped["GDP"].median()
mean_Income= df_new_droped["Income composition of resources"].mean()
mean_Schooling = df_new_droped["Schooling"].mean()
```

In [337…
```
df_new_droped["median_Alcohol"] = df_new_droped["Alcohol"].fillna(median_Alcohol)
df_new_droped["median_GDP"] = df_new_droped["GDP"].fillna(median_GDP)
df_new_droped["mean_Income"] = df_new_droped["Income composition of resources"].fillna(mean_I
df_new_droped["mean_Schooling"] = df_new_droped["Schooling"].fillna(mean_Schooling)
```

```
C:\Users\bipla\AppData\Local\Temp\ipykernel_2240\3208763924.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  df_new_droped["median_Alcohol"] = df_new_droped["Alcohol"].fillna(median_Alcohol)
C:\Users\bipla\AppData\Local\Temp\ipykernel_2240\3208763924.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  df_new_droped["median_GDP"] = df_new_droped["GDP"].fillna(median_GDP)
C:\Users\bipla\AppData\Local\Temp\ipykernel_2240\3208763924.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  df_new_droped["mean_Income"] = df_new_droped["Income composition of resources"].fillna(mean
_Income)
C:\Users\bipla\AppData\Local\Temp\ipykernel_2240\3208763924.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  df_new_droped["mean_Schooling"] = df_new_droped["Schooling"].fillna(mean_Schooling)
```

In [338…
```
df_new_droped.head()
```

Out[338]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Measles	BMI	…	D
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	1154	19.1	…	
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	492	18.6	…	
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	430	18.1	…	
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	2787	17.6	…	
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	3013	17.2	…	

5 rows × 22 columns

In [339…
```
df_new_droped["Alcohol"].var(), df_new_droped["median_Alcohol"].var()
```

Out[339]:
```
(16.431093845621554, 15.474688703251806)
```

In [340…
```
df_new_droped["GDP"].var(), df_new_droped["median_GDP"].var()
```

Out[340]:
```
(206105351.6132651, 179303113.42803004)
```

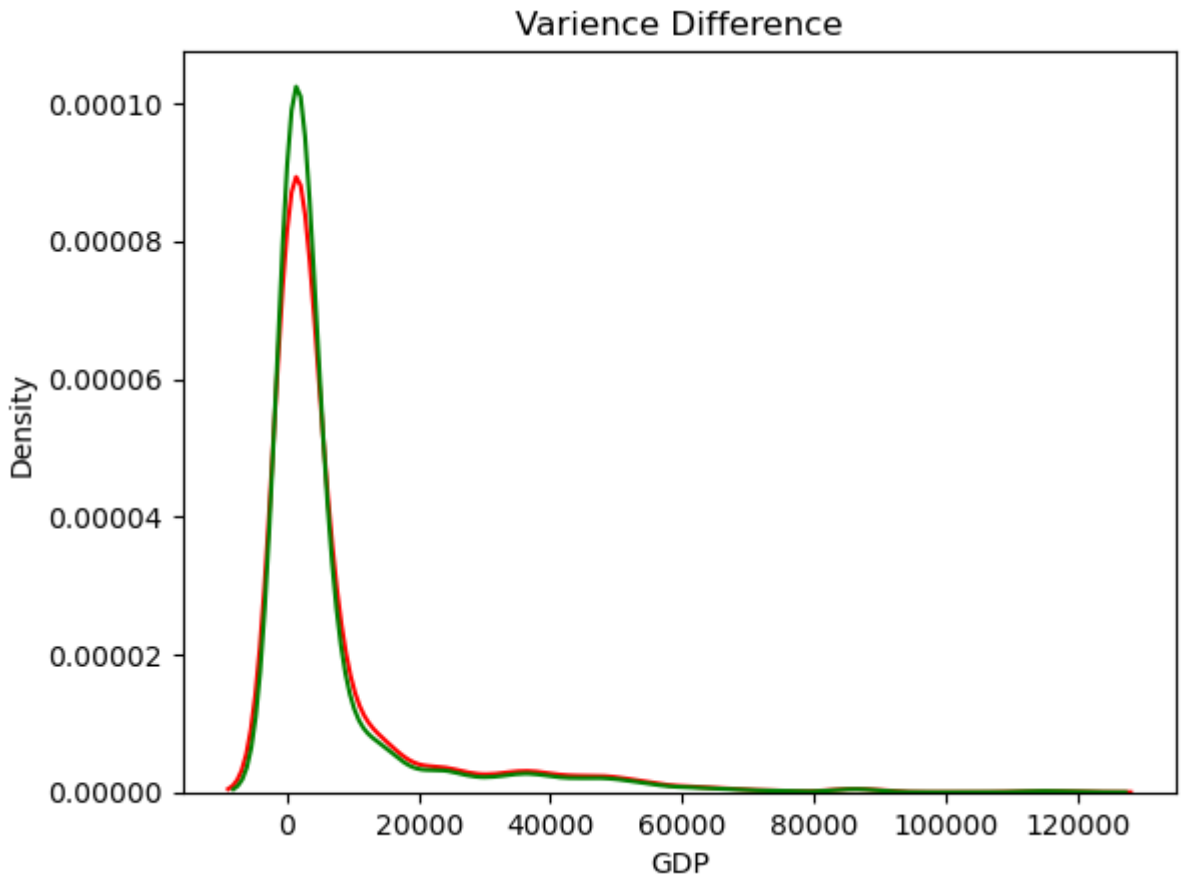
In [341…
```
df_new_droped["Income composition of resources"].var(), df_new_droped["mean_Income"].var()
```

Out[341]:
```
(0.04252259958468958, 0.040165960882386026)
```

In [342…]
```
df_new_droped["Schooling"].var(), df_new_droped["mean_Schooling"].var()
```

Out[342]:
```
(10.222084379136794, 9.65556775265191)
```

In [343…]
```
fig=plt.figure()
ax=fig.add_subplot(111)
plt.title("Varience Difference")
sns.kdeplot(data=df_new["GDP"], ax=ax, color="red")
sns.kdeplot(data=df_new_droped["median_GDP"], ax=ax, color="green")
plt.show()
```



In [344…]
```
final_df = df_new_droped.drop(['Alcohol', 'GDP', 'Income composition of resources', 'Schoolin
```

In [345…]
```
final_df.head()
```

Out[345]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	percentage expenditure	Measles	BMI	under-five deaths	Polio
0	Afghanistan	2015	Developing	65.0	263.0	62	71.279624	1154	19.1	83	6.0
1	Afghanistan	2014	Developing	59.9	271.0	64	73.523582	492	18.6	86	58.0
2	Afghanistan	2013	Developing	59.9	268.0	66	73.219243	430	18.1	89	62.0
3	Afghanistan	2012	Developing	59.5	272.0	69	78.184215	2787	17.6	93	67.0
4	Afghanistan	2011	Developing	59.2	275.0	71	7.097109	3013	17.2	97	68.0

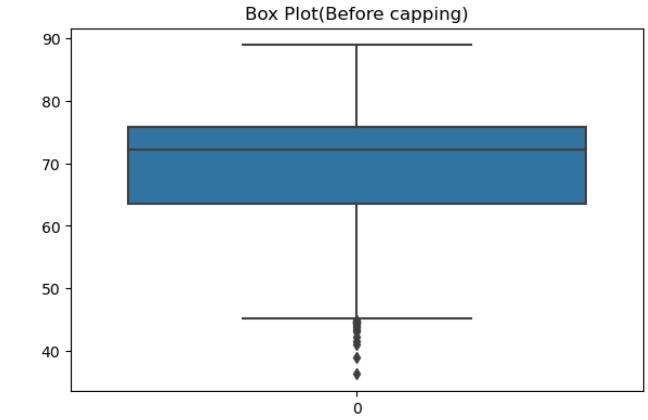
Handling Outliers (With IQR Method by capping)

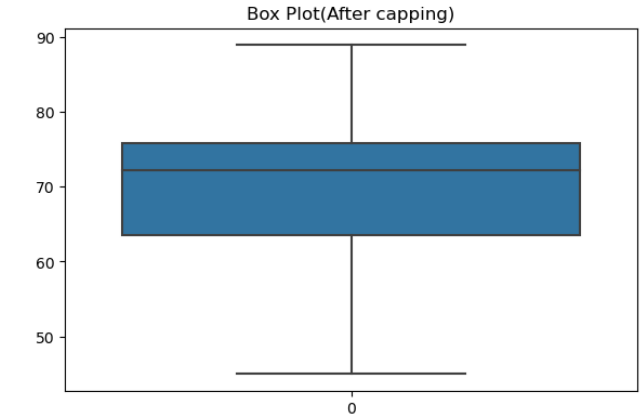
In [346…]
```
df_without_otliers = final_df.copy()
```

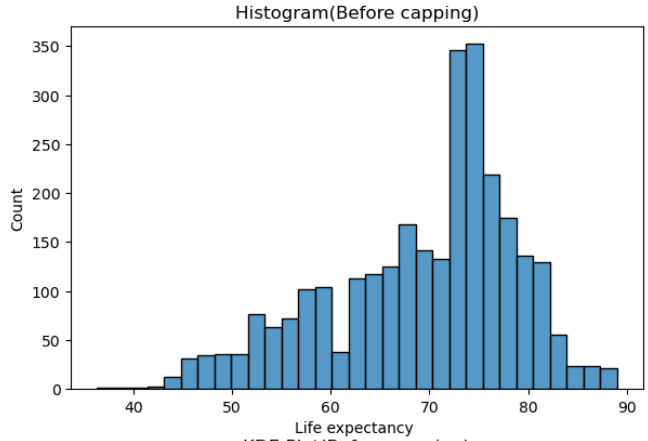
In [347…]
```
cap(final_df, df_without_otliers)
```

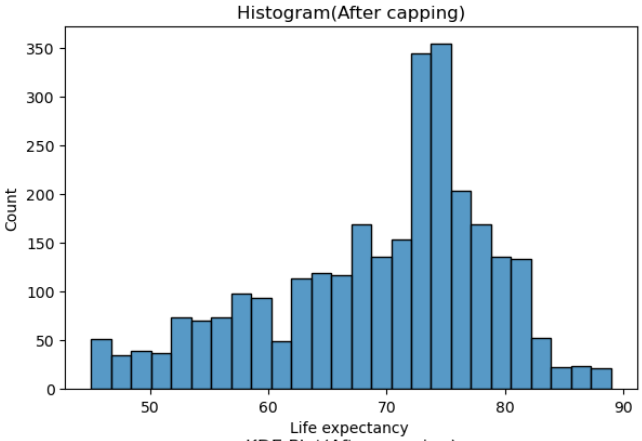
##

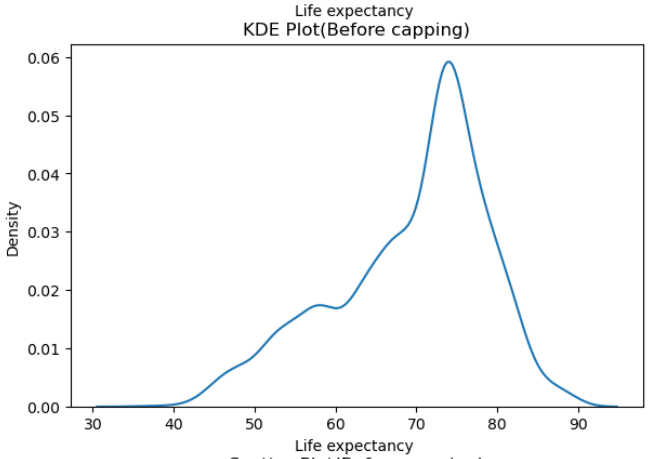
Variable Name : Life expectancy
Persentile 25 : 63.475
Persentile 75 : 75.8
IQR : 12.324999999999996
Upper Limit : 94.2875
Lower Limit : 44.98750000000001
Outliers Removed using IQR Method by Capping

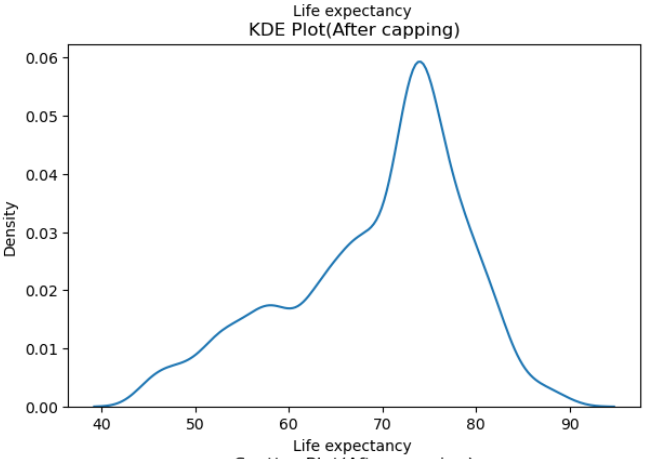


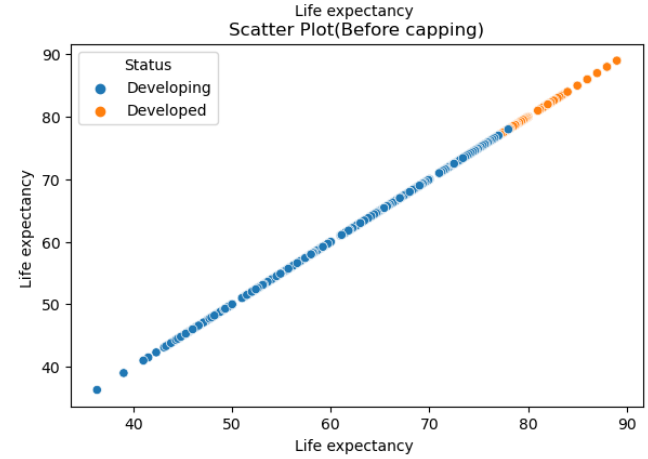


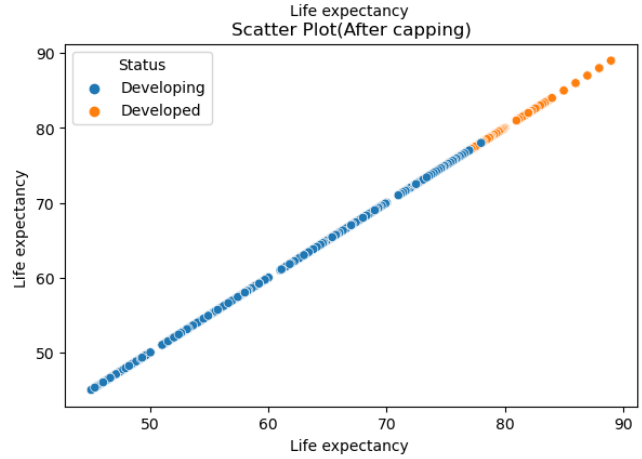






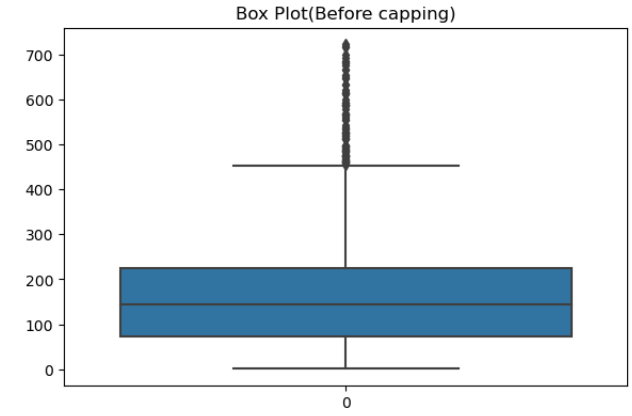


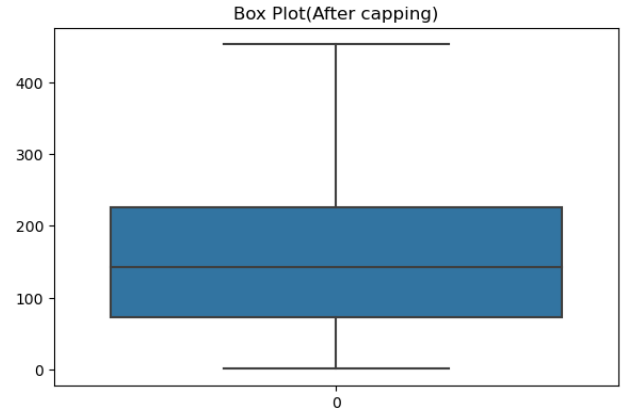


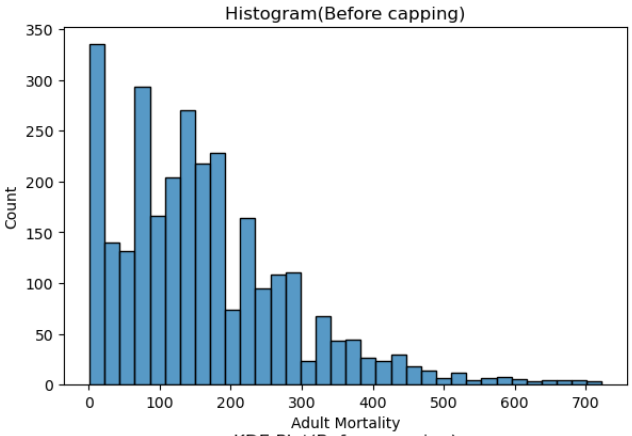


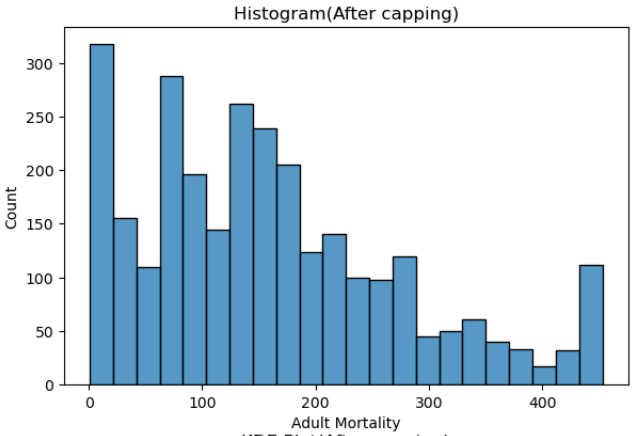
##

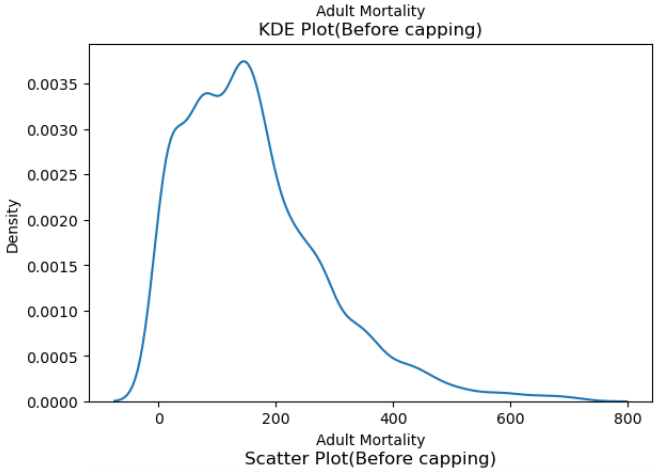
Variable Name : Adult Mortality
Persentile 25 : 73.0
Persentile 75 : 225.0
IQR : 152.0
Upper Limit : 453.0
Lower Limit : -155.0
Outliers Removed using IQR Method by Capping

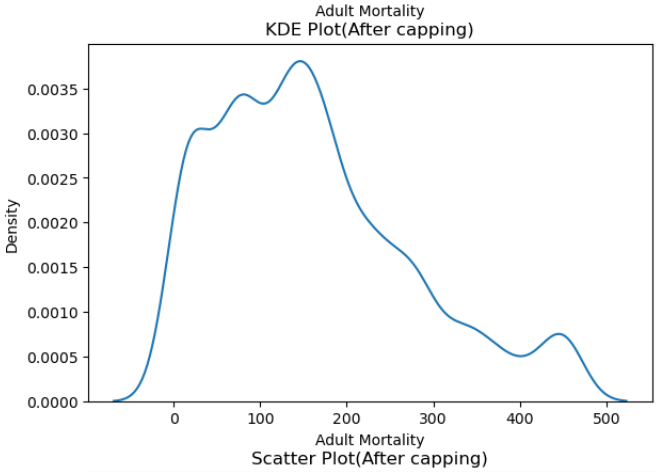


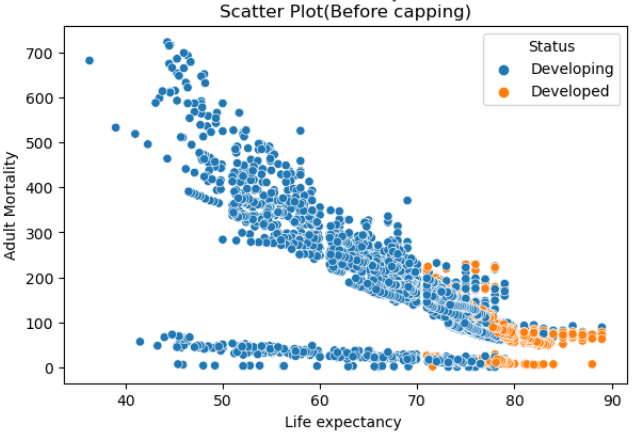


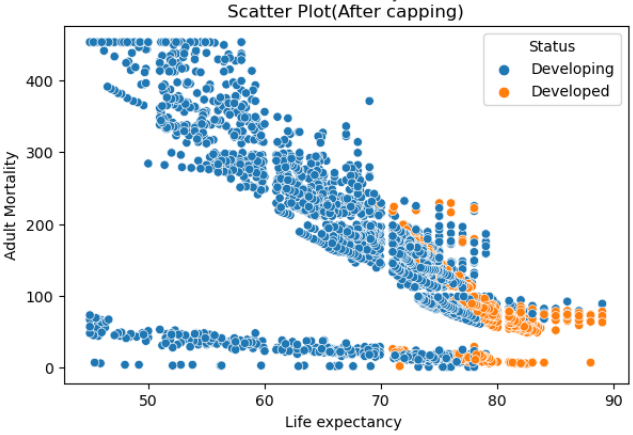






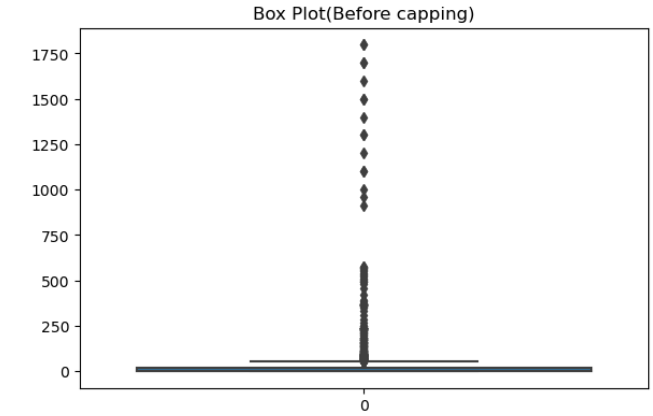


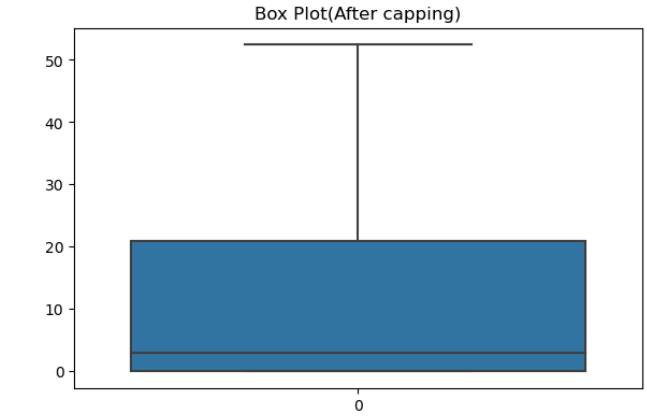


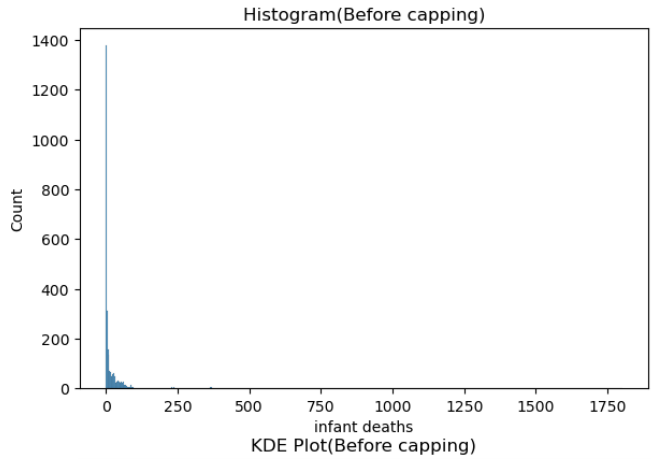


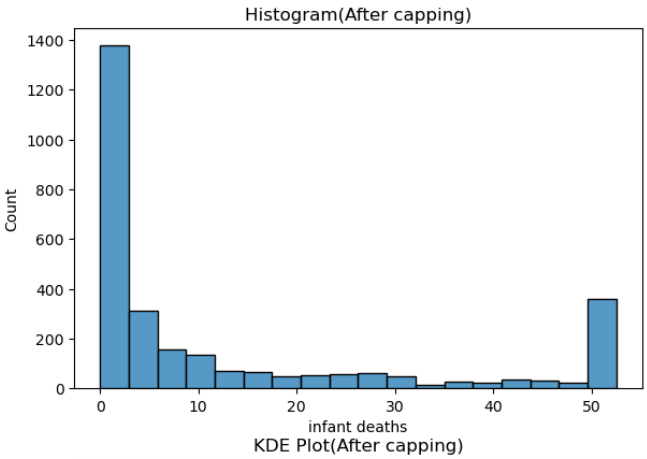
##

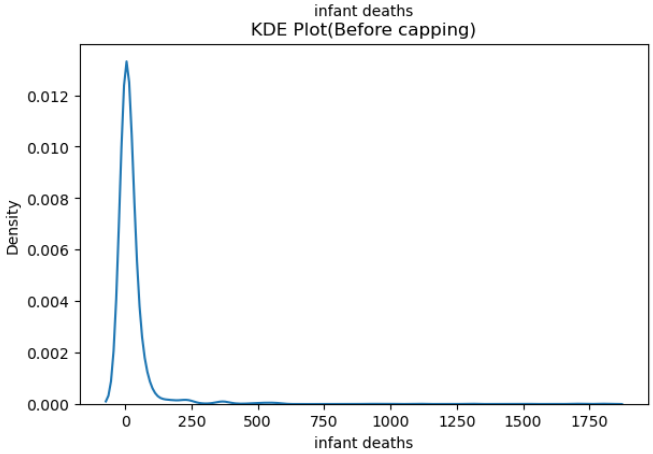
Variable Name : infant deaths
Persentile 25 : 0.0
Persentile 75 : 21.0
IQR : 21.0
Upper Limit : 52.5
Lower Limit : -31.5
Outliers Removed using IQR Method by Capping

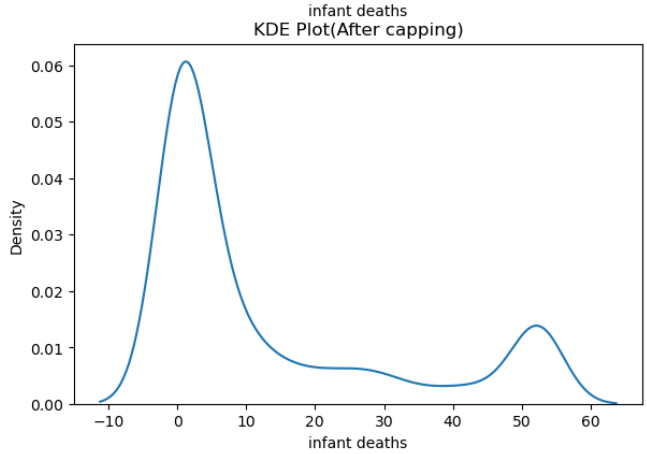


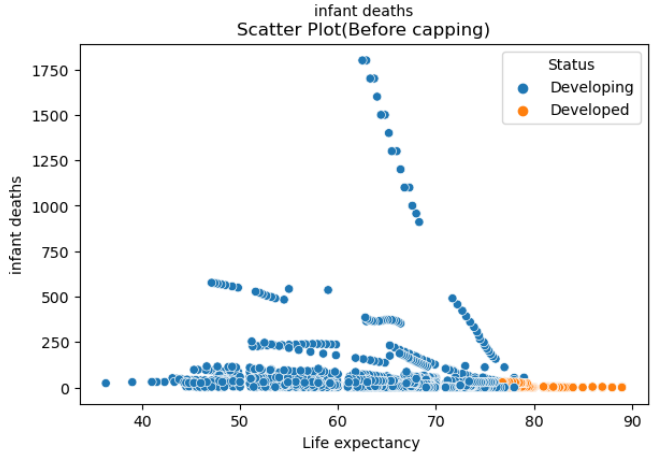


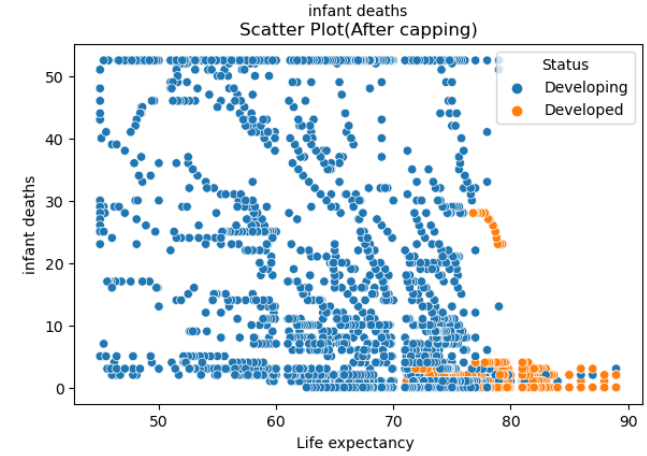




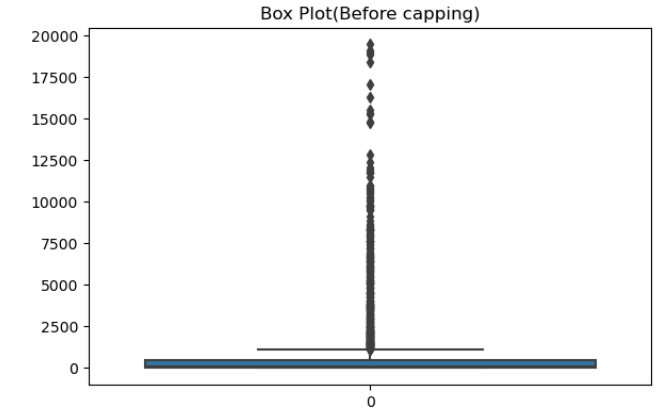


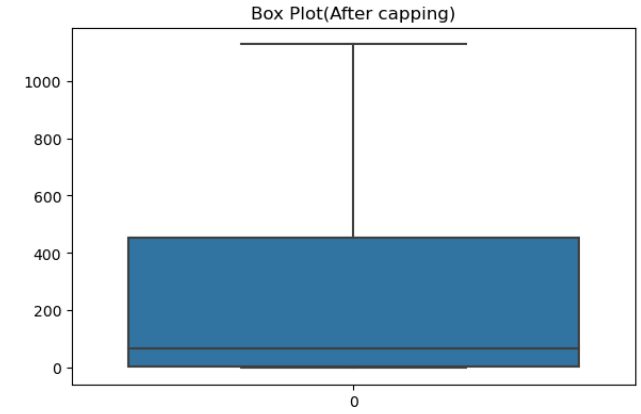


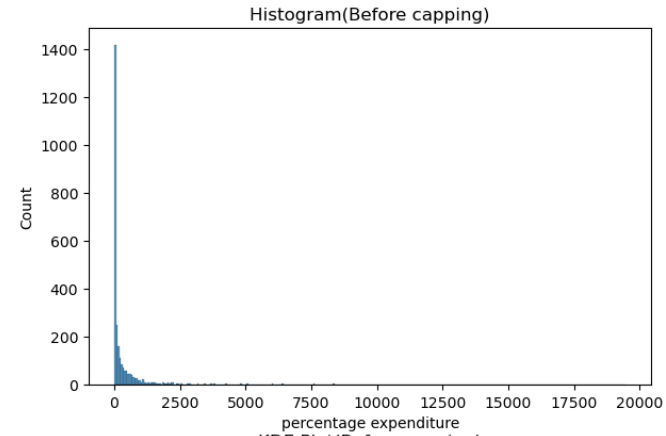


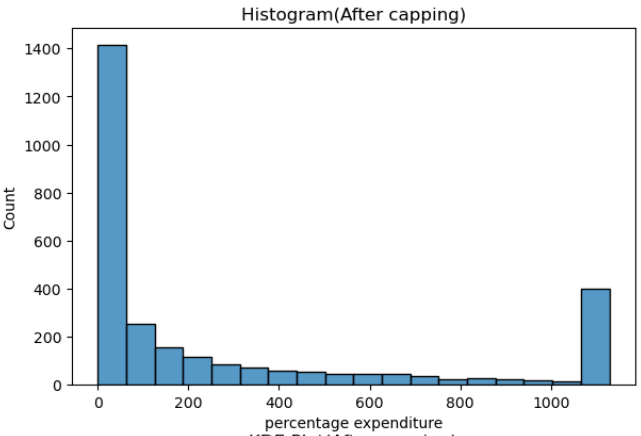


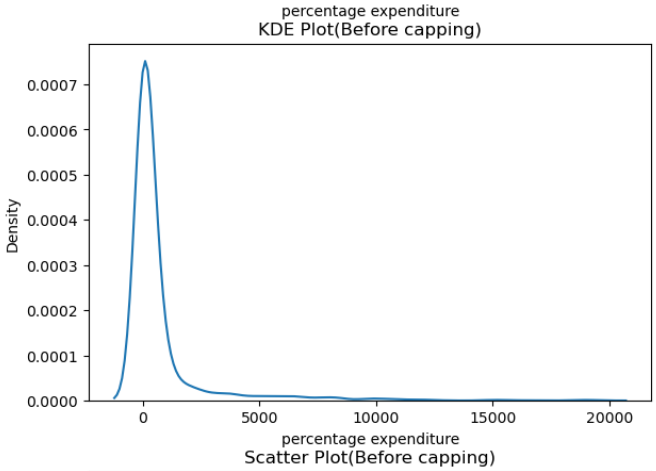
```
##########################################################################

Variable Name :  percentage expenditure
Persentile 25 :  5.049462446
Persentile 75 :  454.42242995000004
IQR :  449.37296750400003
Upper Limit :  1128.4818812060003
Lower Limit :  -669.0099888100001
Outliers Removed using IQR Method by Capping
```

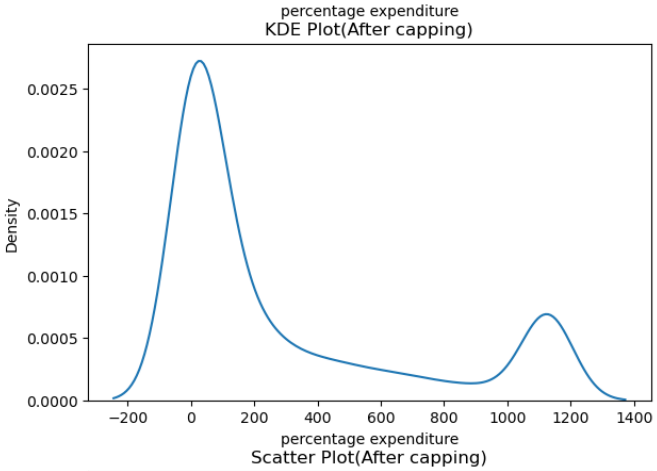


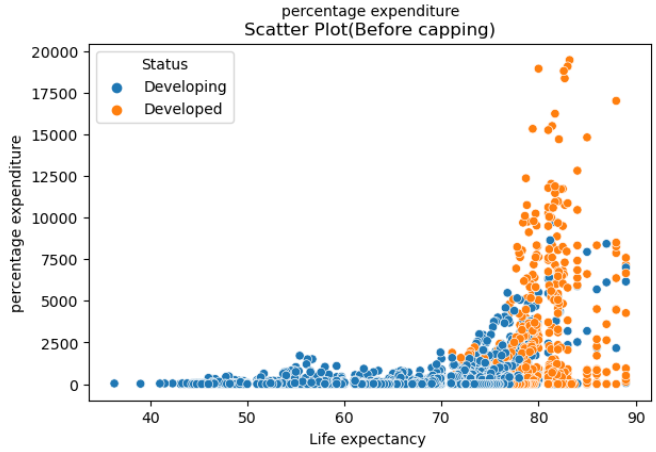


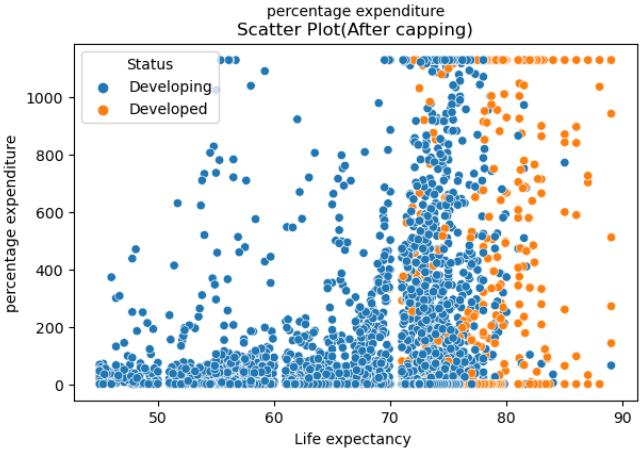






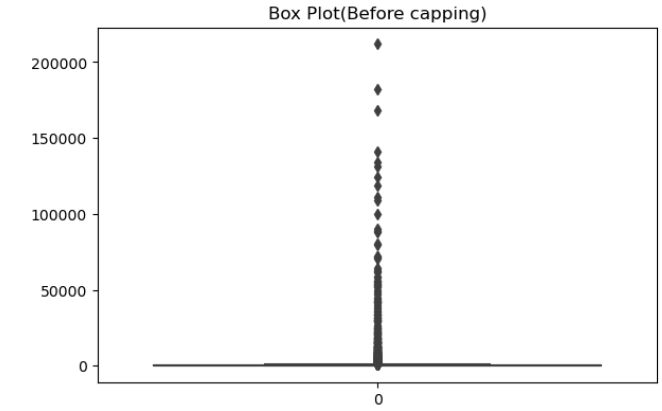


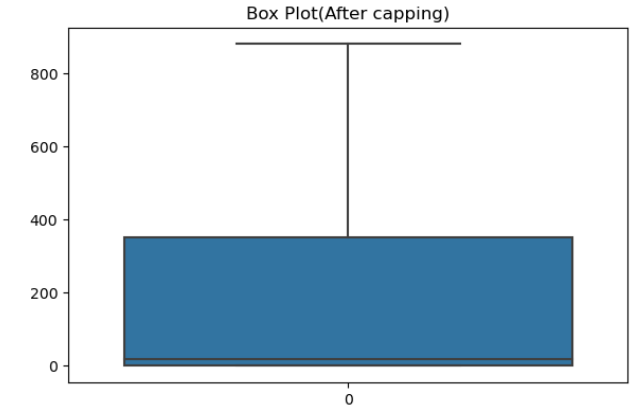


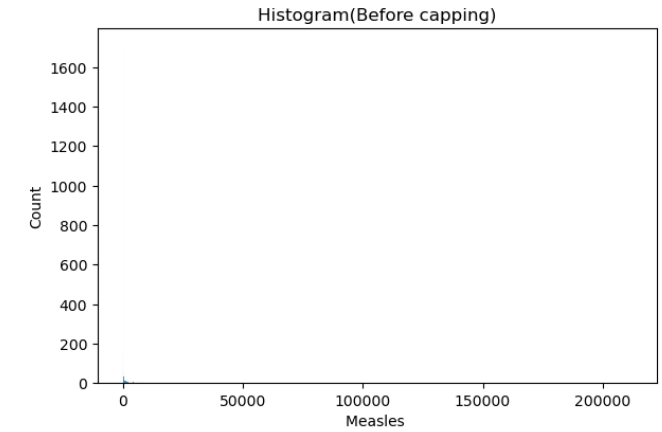


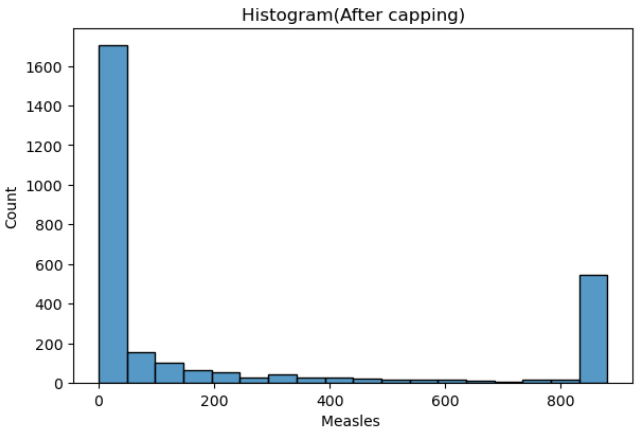
##

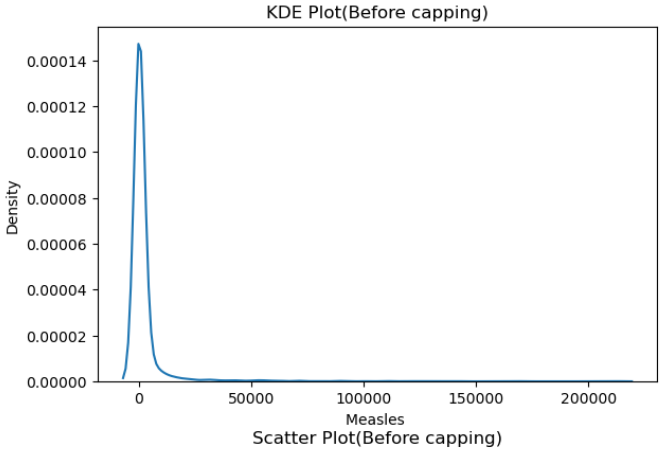
Variable Name : Measles
Persentile 25 : 0.0
Persentile 75 : 352.25
IQR : 352.25
Upper Limit : 880.625
Lower Limit : -528.375
Outliers Removed using IQR Method by Capping

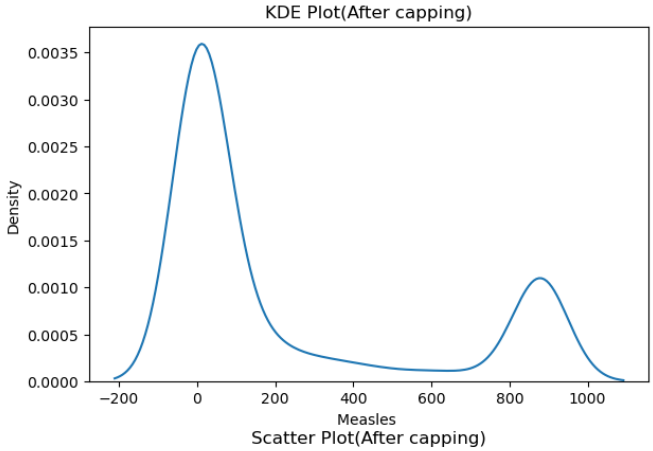


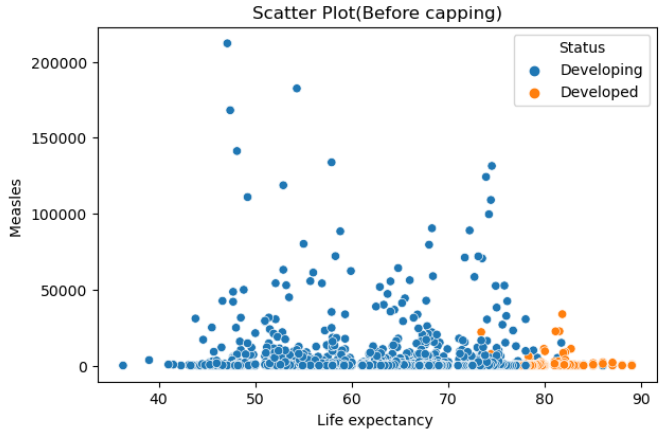


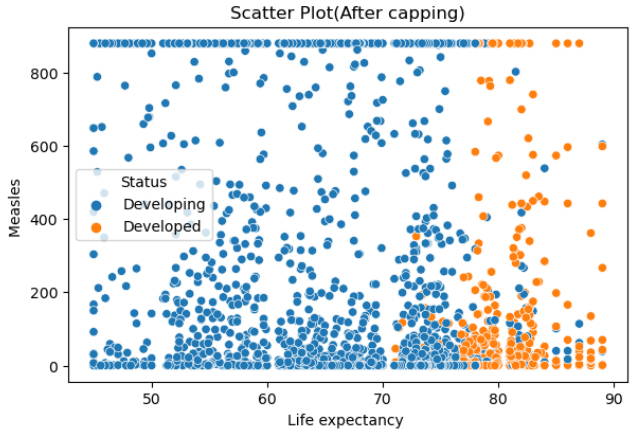




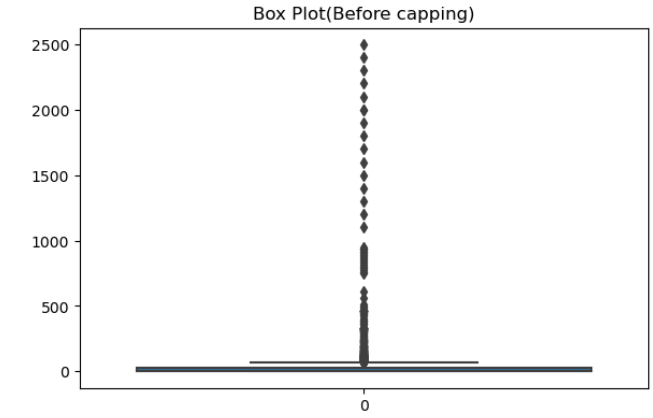


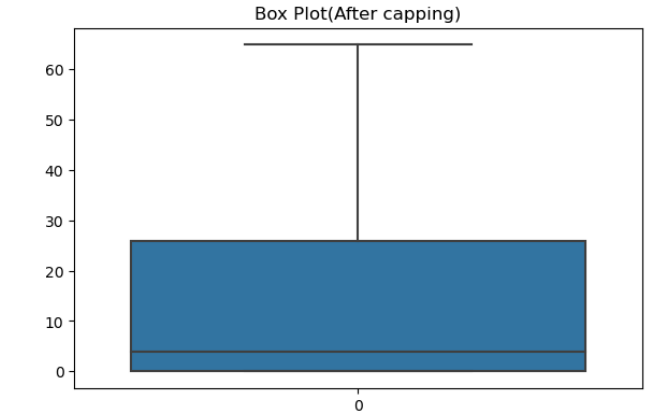


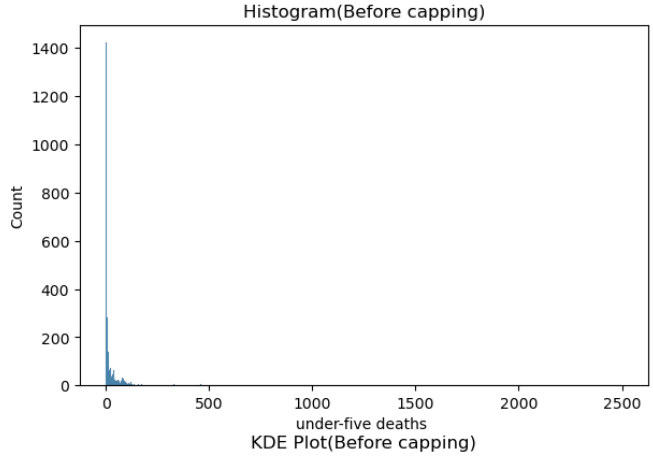


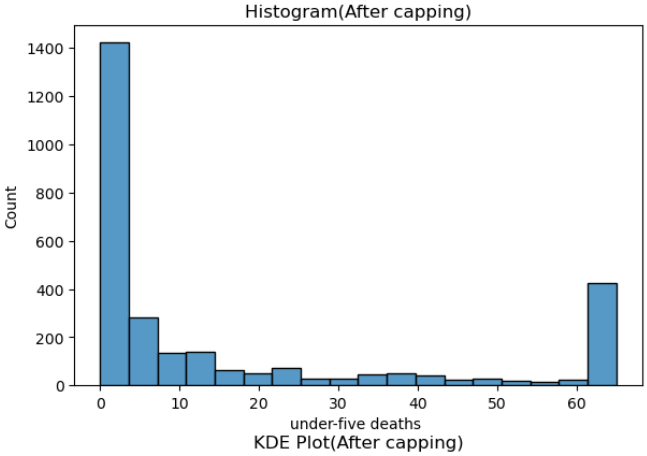


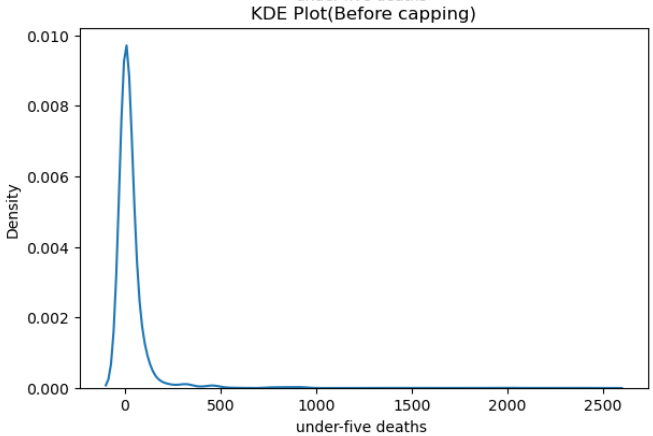
```
##############################################################################

Variable Name :   BMI
No Outliers found
##############################################################################

Variable Name :  under-five deaths
Persentile 25 :  0.0
Persentile 75 :  26.0
IQR :  26.0
Upper Limit :  65.0
Lower Limit :  -39.0
Outliers Removed using IQR Method by Capping
```

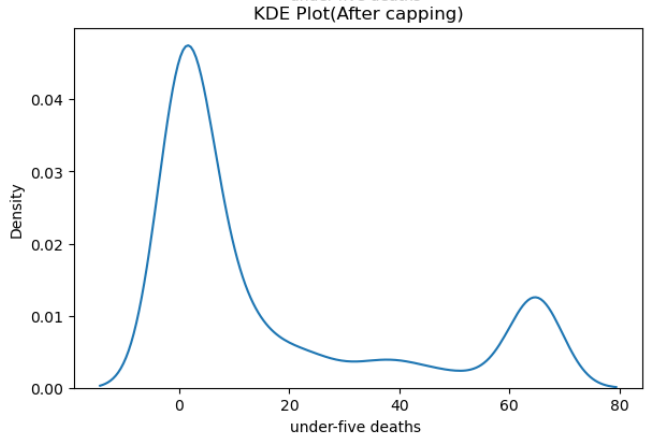


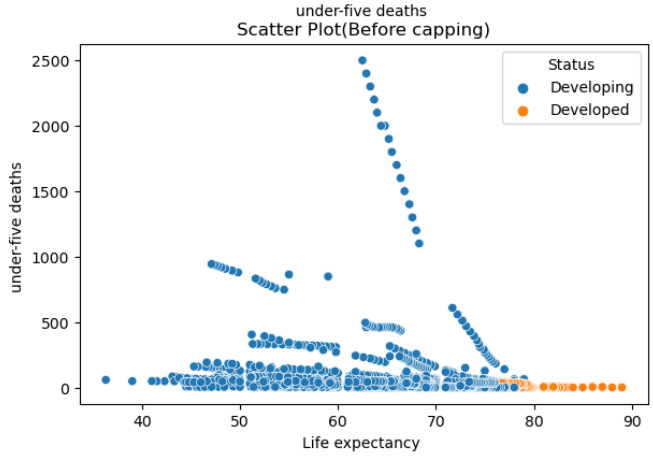


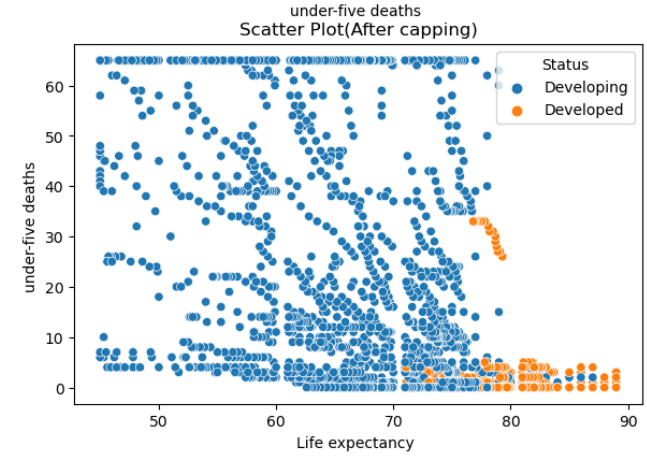






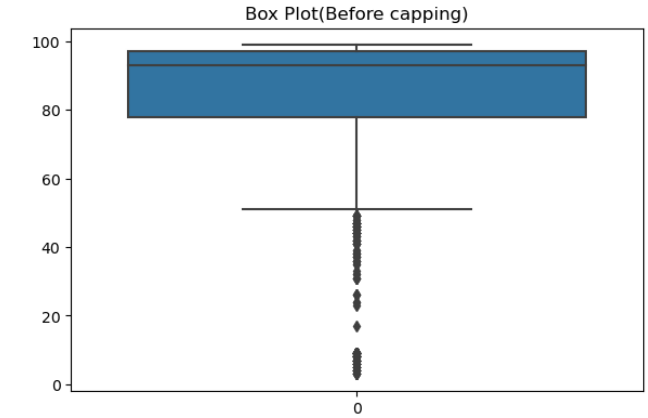


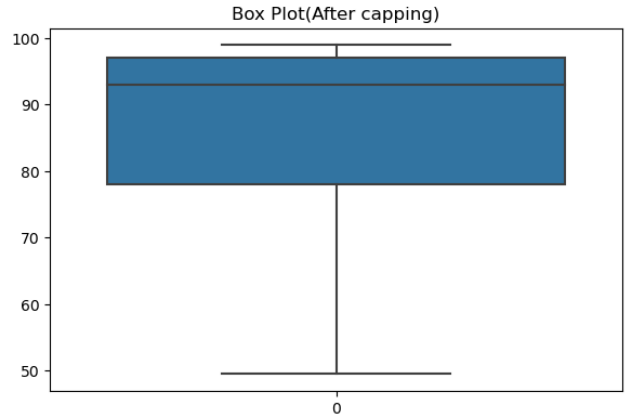


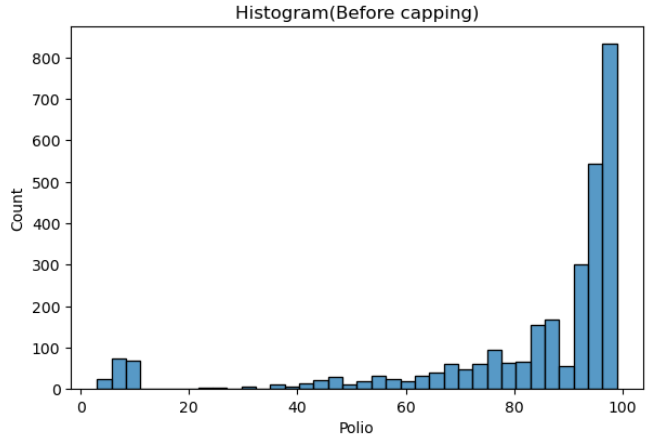


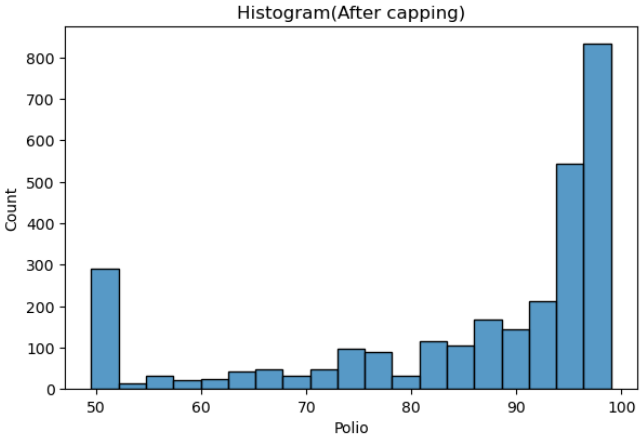
##

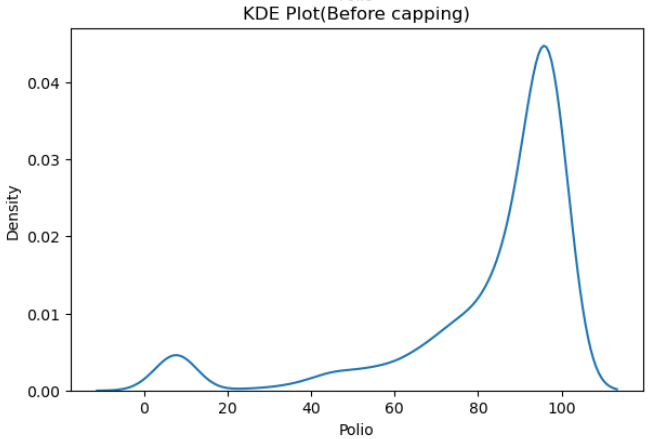
Variable Name : Polio
Persentile 25 : 78.0
Persentile 75 : 97.0
IQR : 19.0
Upper Limit : 125.5
Lower Limit : 49.5
Outliers Removed using IQR Method by Capping

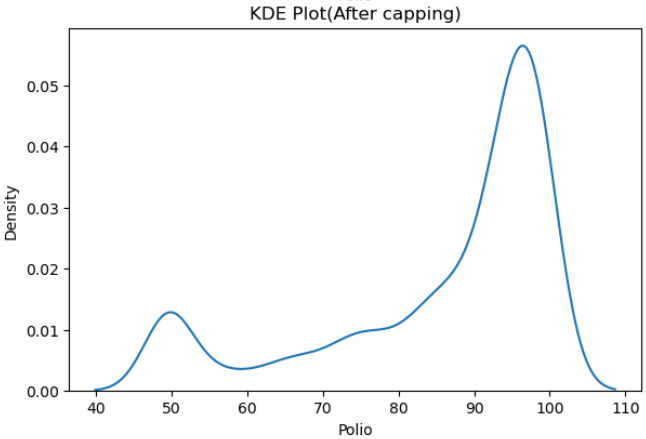


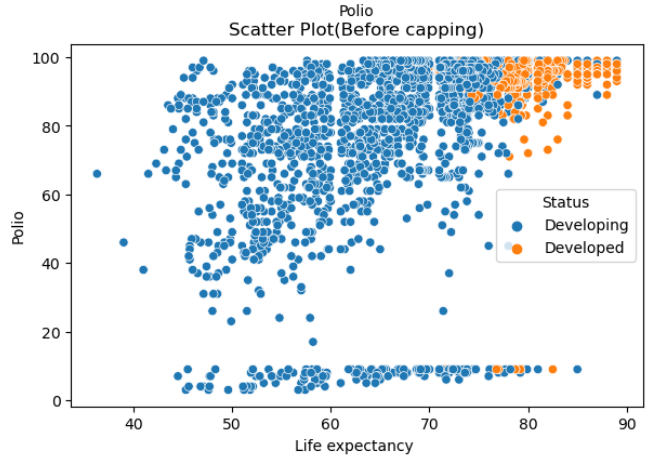


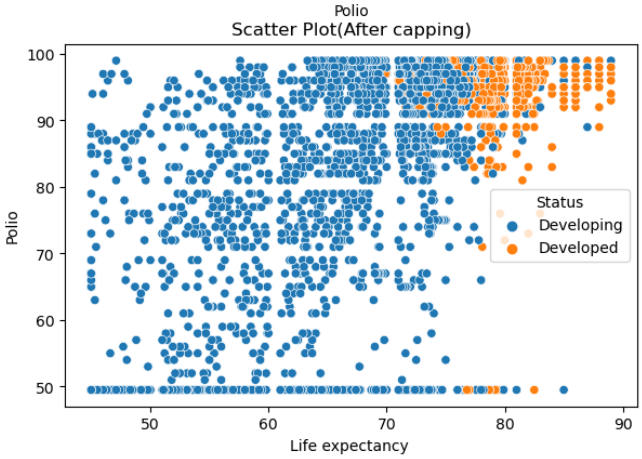






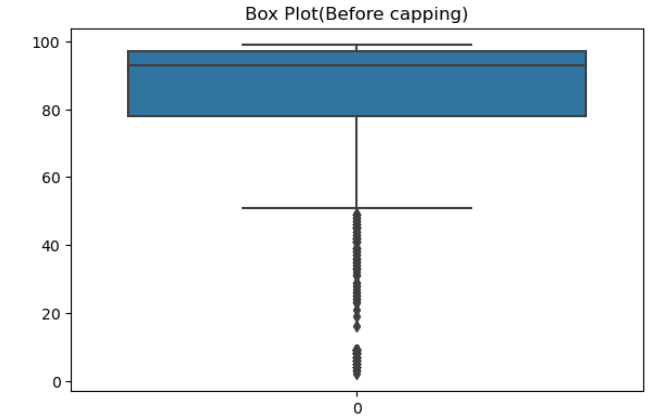


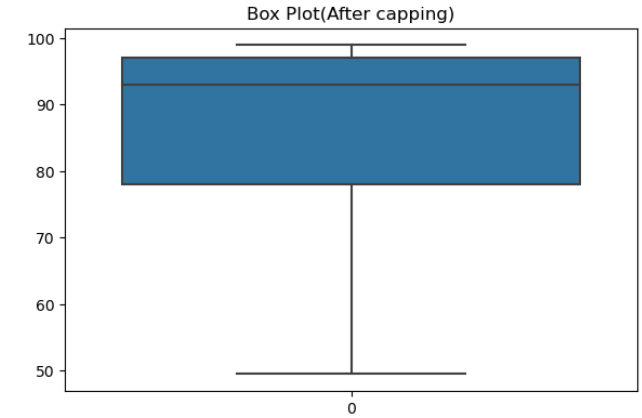


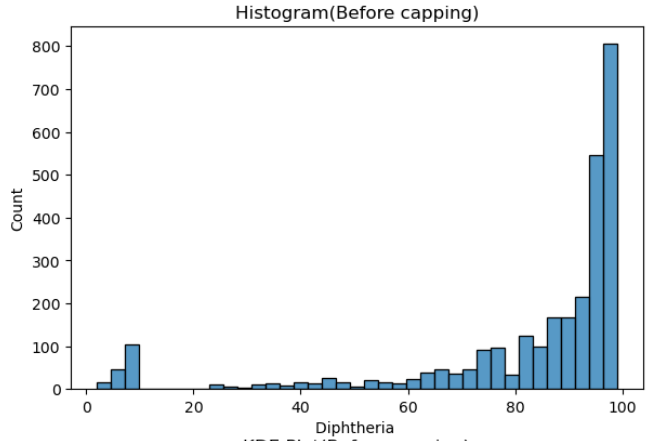


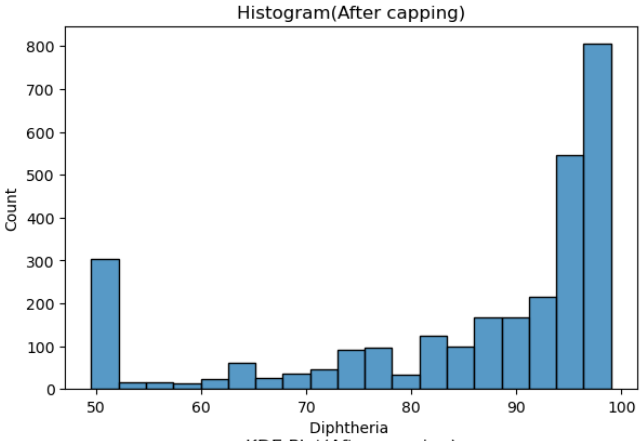
##

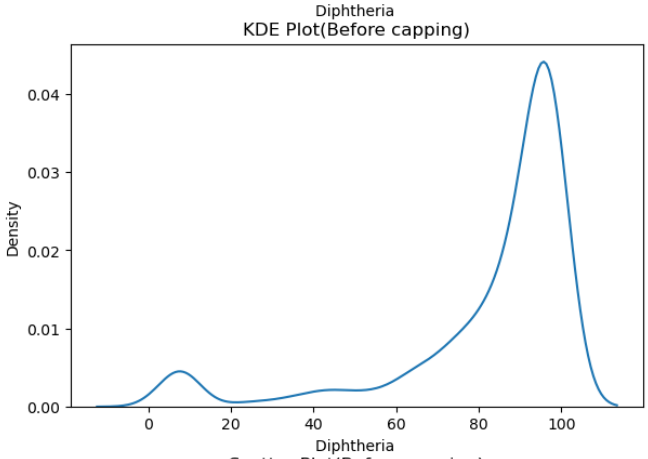
Variable Name : Diphtheria
Persentile 25 : 78.0
Persentile 75 : 97.0
IQR : 19.0
Upper Limit : 125.5
Lower Limit : 49.5
Outliers Removed using IQR Method by Capping

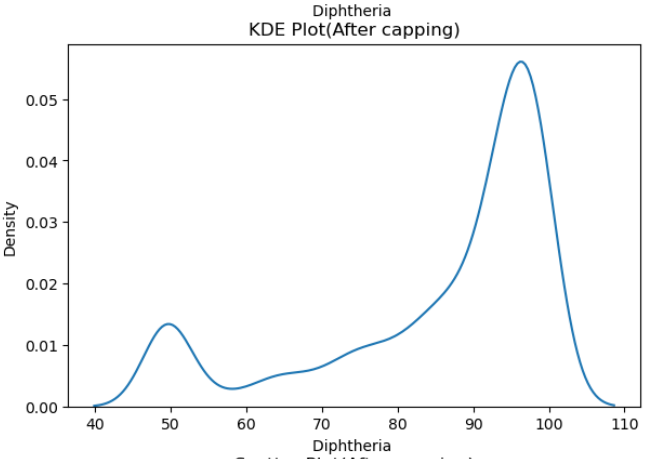


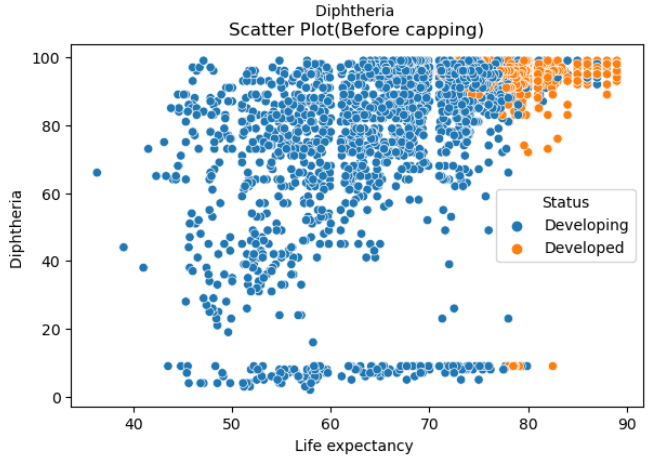


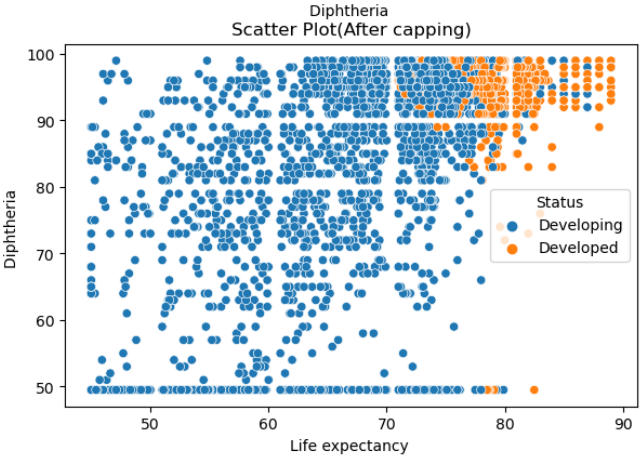






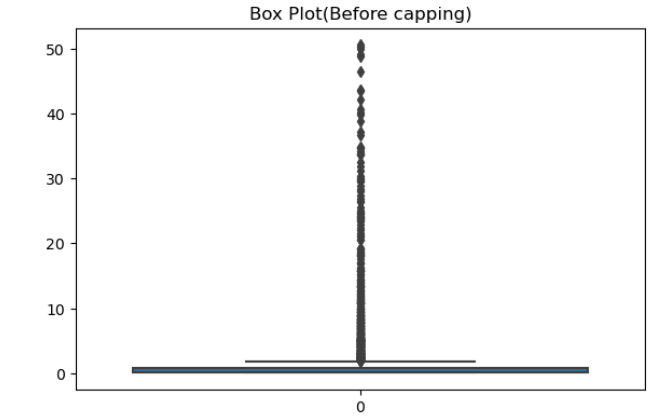


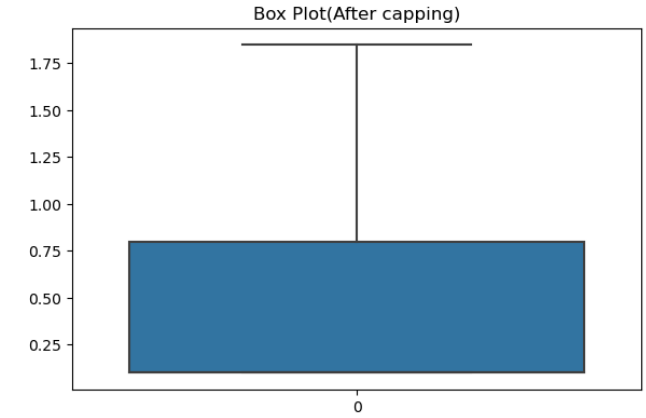


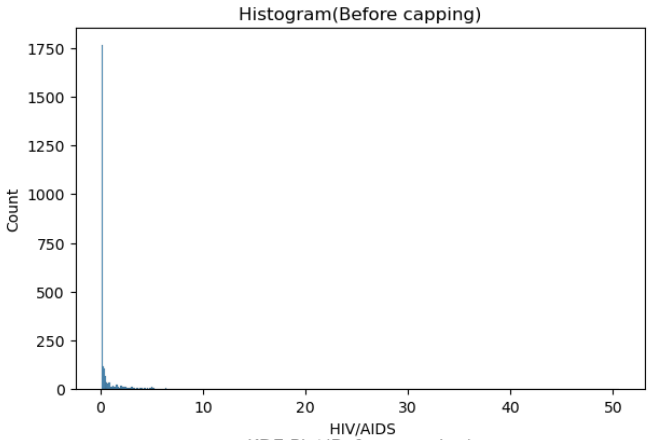


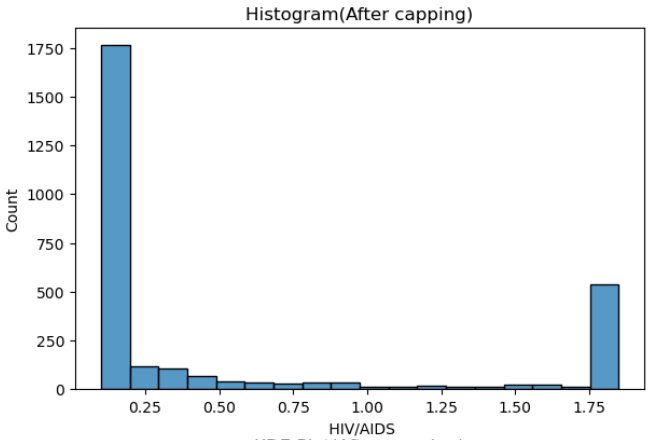
##

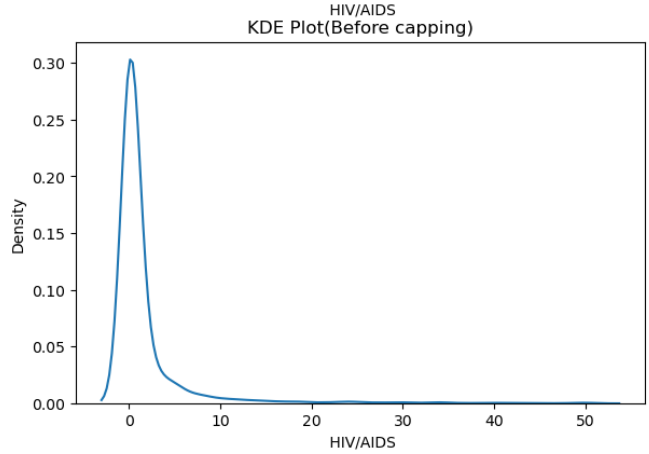
Variable Name : HIV/AIDS
Persentile 25 : 0.1
Persentile 75 : 0.8
IQR : 0.7000000000000001
Upper Limit : 1.85
Lower Limit : -0.9500000000000001
Outliers Removed using IQR Method by Capping

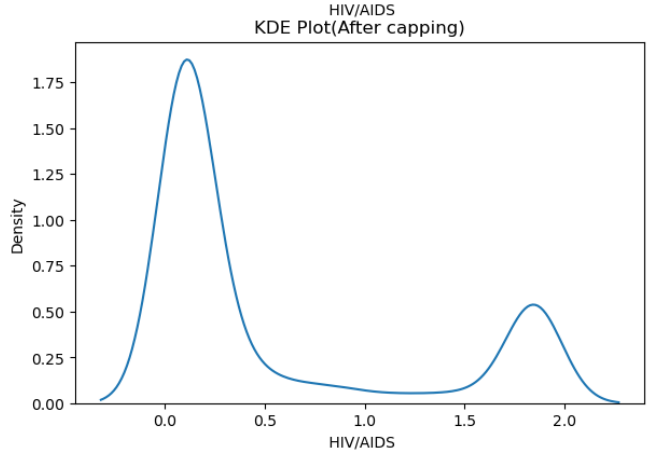


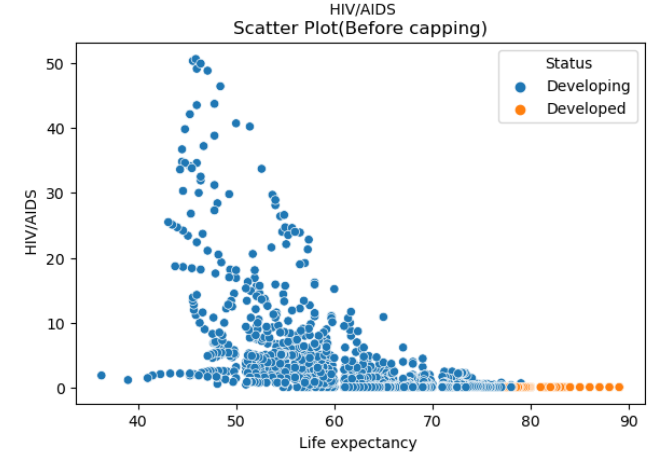


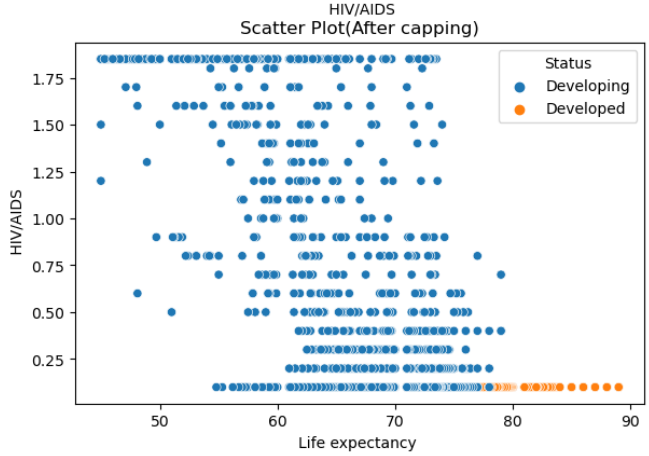




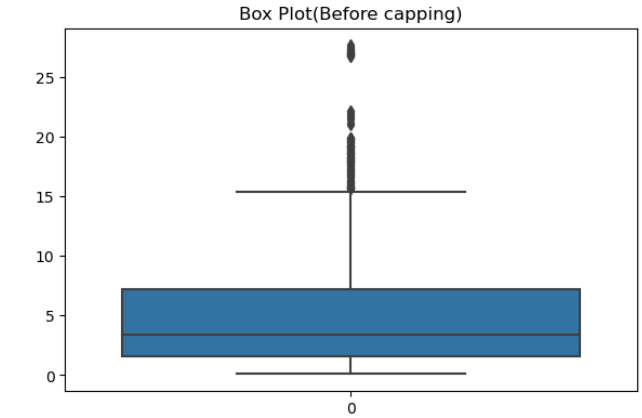


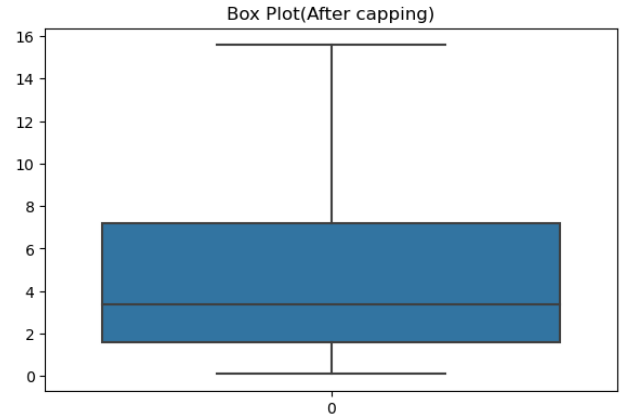


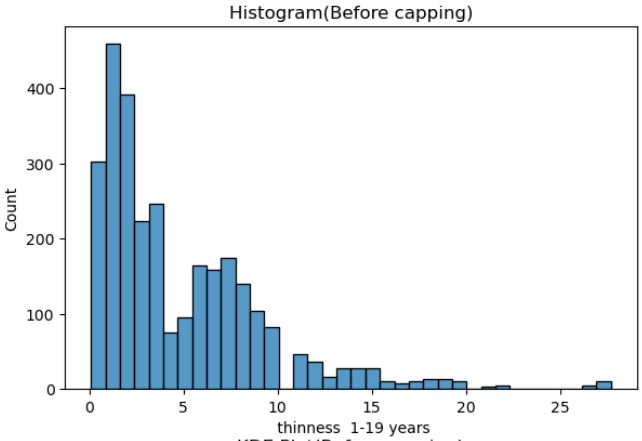


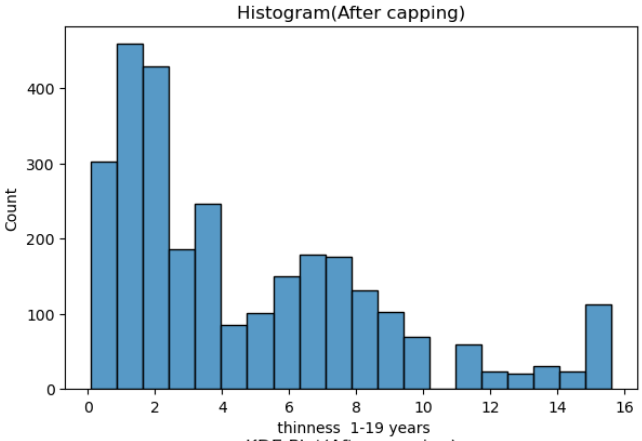


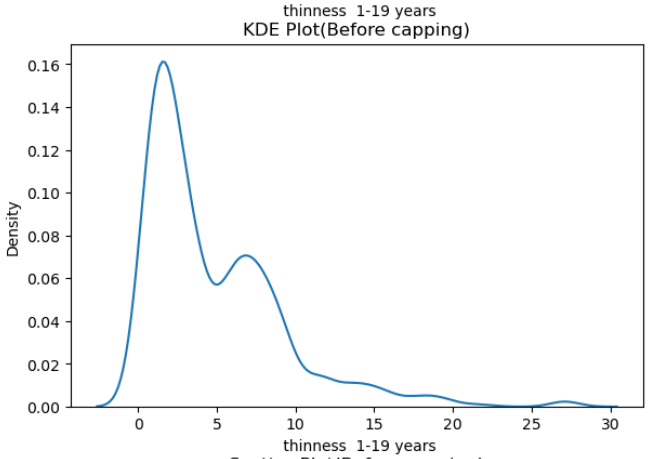
```
##########################################################################

Variable Name :   thinness  1-19 years
Persentile 25 :  1.6
Persentile 75 :  7.2
IQR :  5.6
Upper Limit :  15.599999999999998
Lower Limit :  -6.799999999999999
Outliers Removed using IQR Method by Capping
```

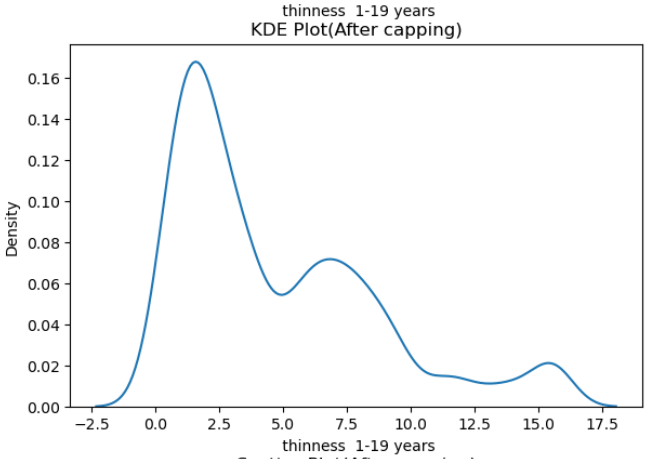


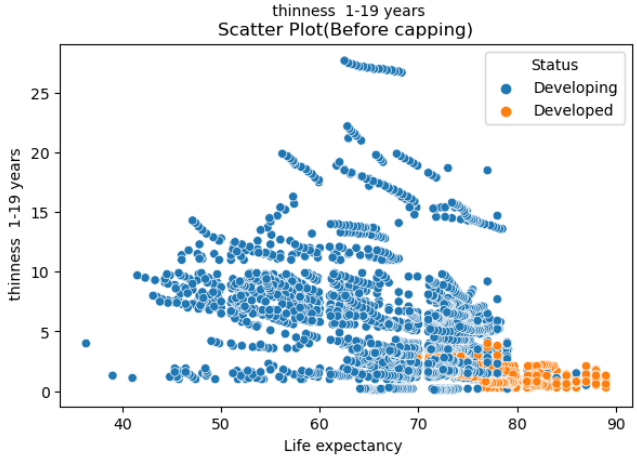


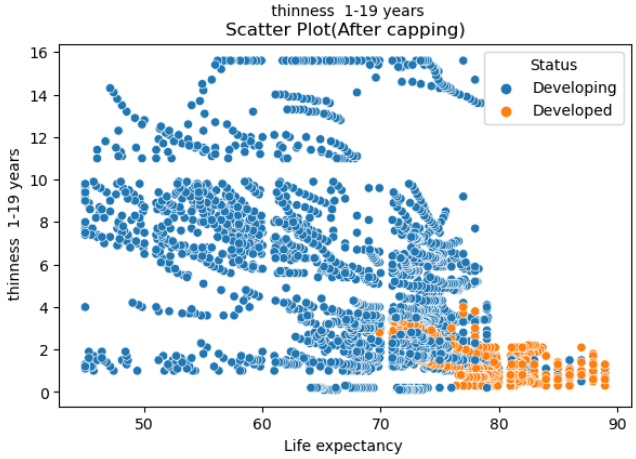






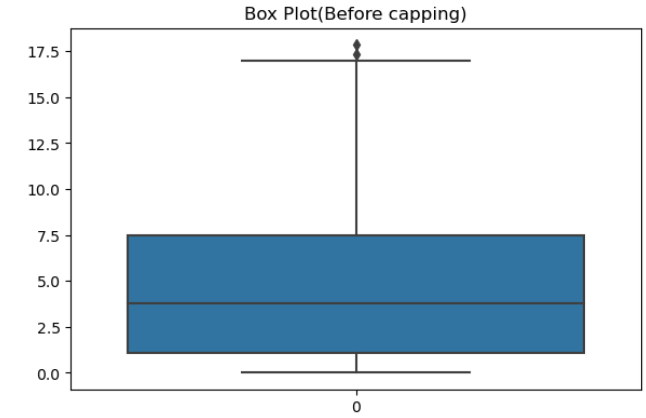


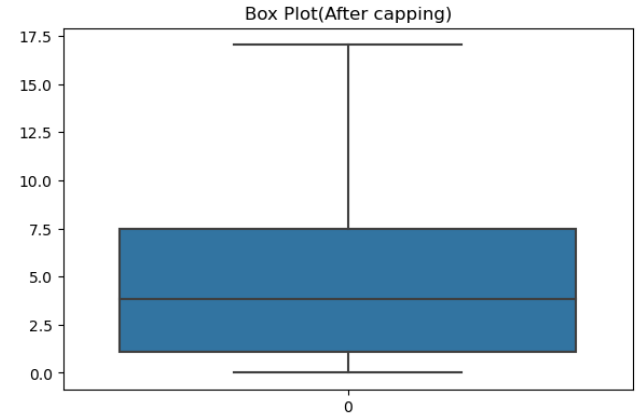


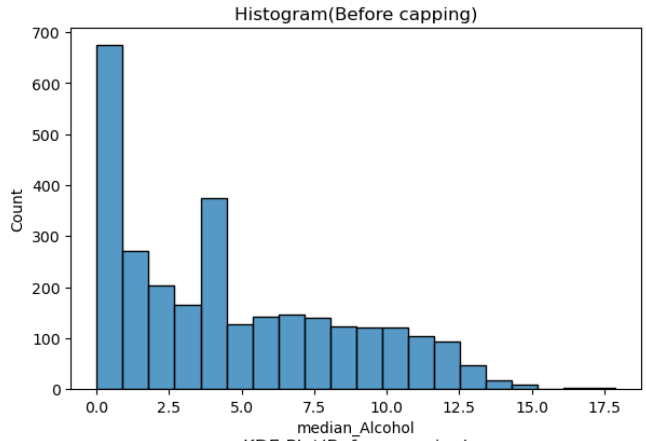


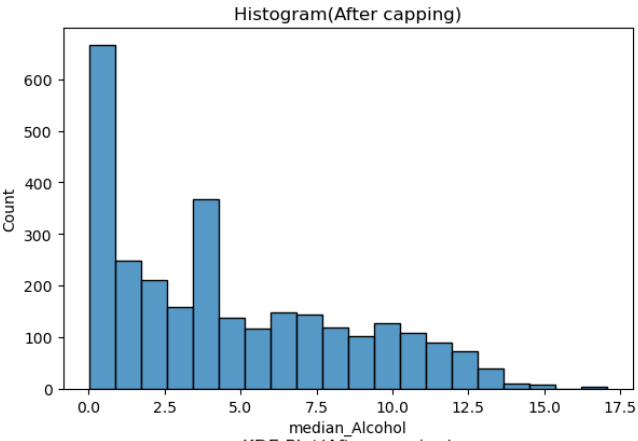
##

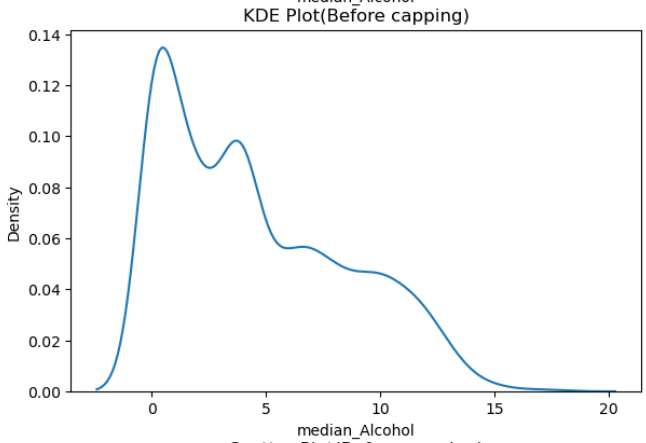
Variable Name : median_Alcohol
Persentile 25 : 1.1075000000000002
Persentile 75 : 7.49
IQR : 6.3825
Upper Limit : 17.06375
Lower Limit : -8.46625
Outliers Removed using IQR Method by Capping

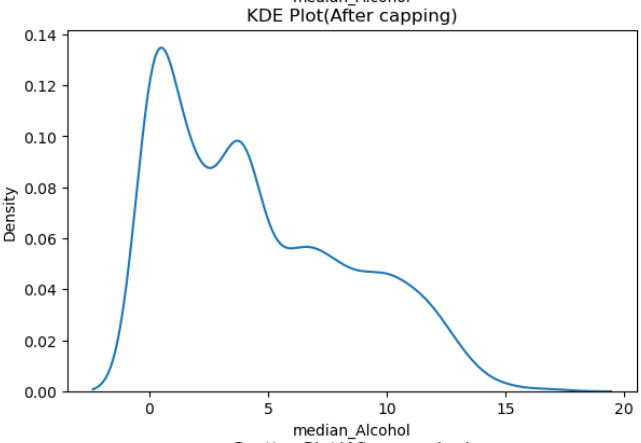


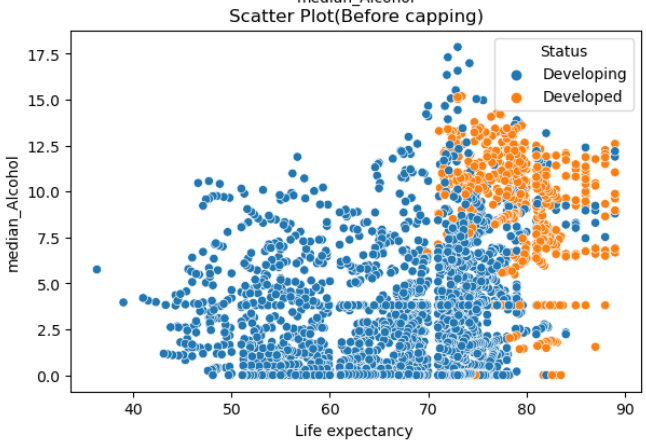


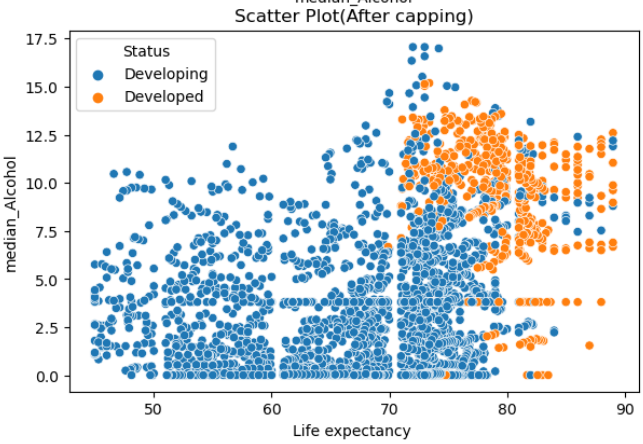






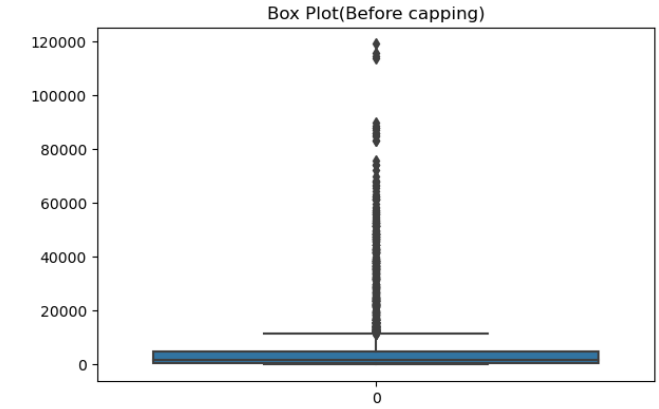


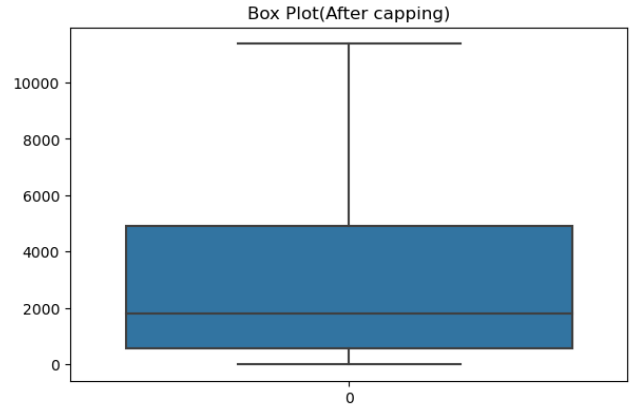


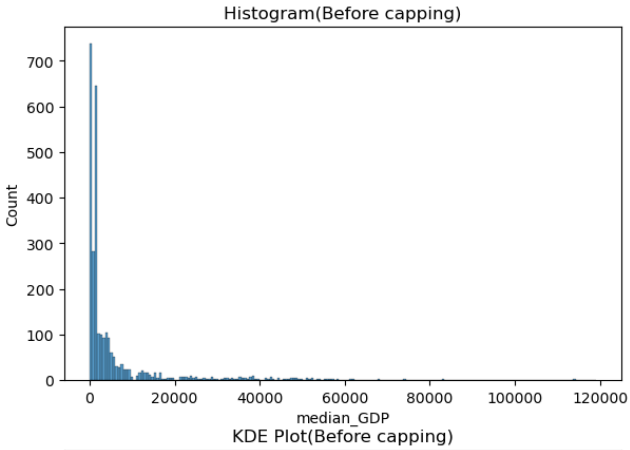


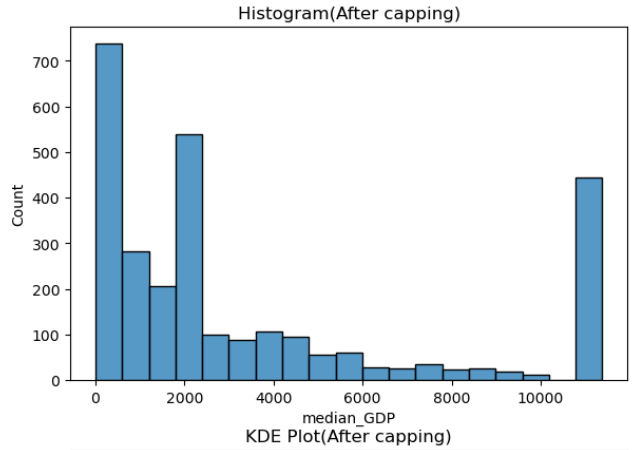
##

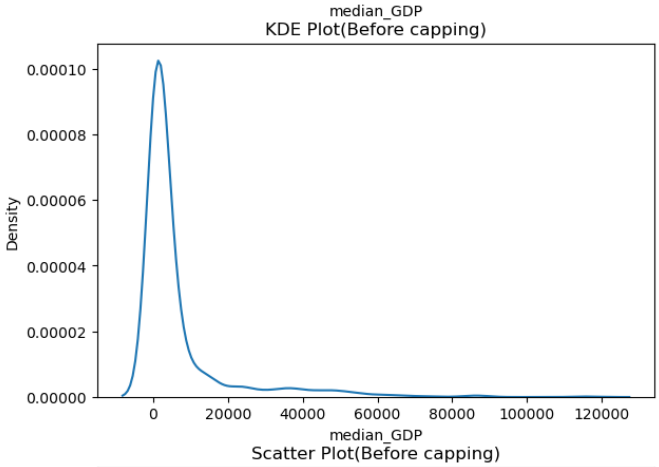
Variable Name : median_GDP
Persentile 25 : 579.651825
Persentile 75 : 4901.575091250001
IQR : 4321.923266250001
Upper Limit : 11384.459990625002
Lower Limit : -5903.233074375002
Outliers Removed using IQR Method by Capping

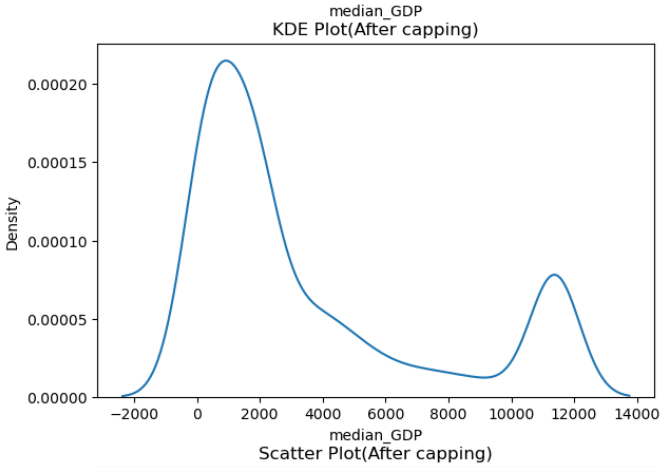


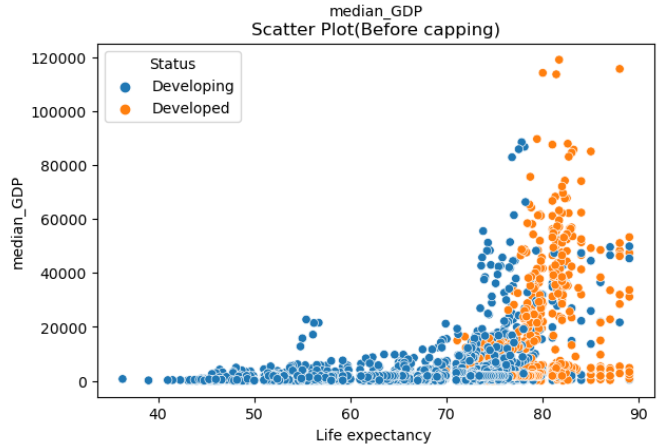


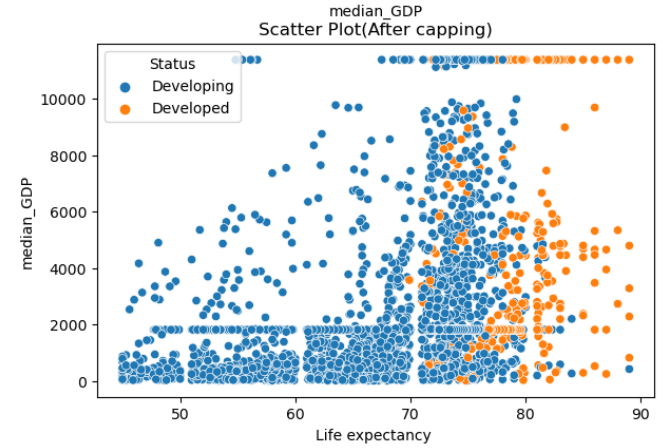




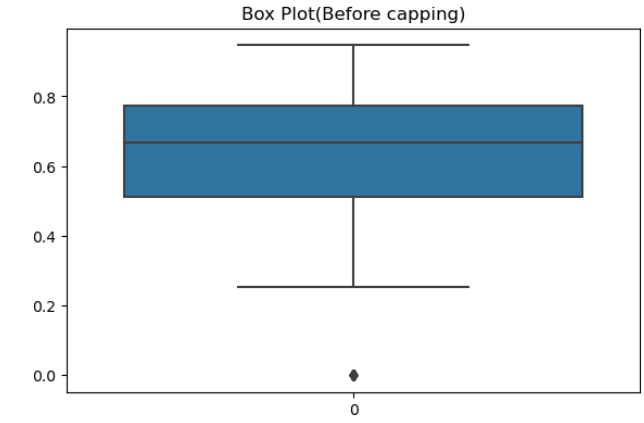


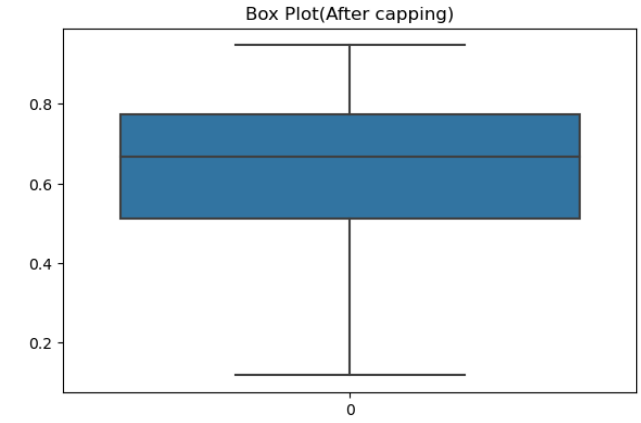


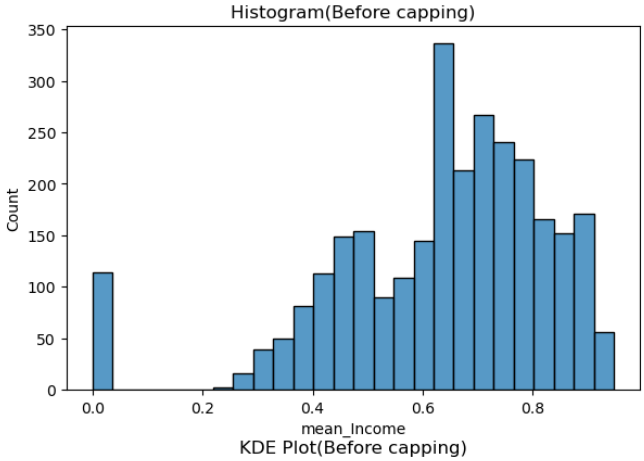


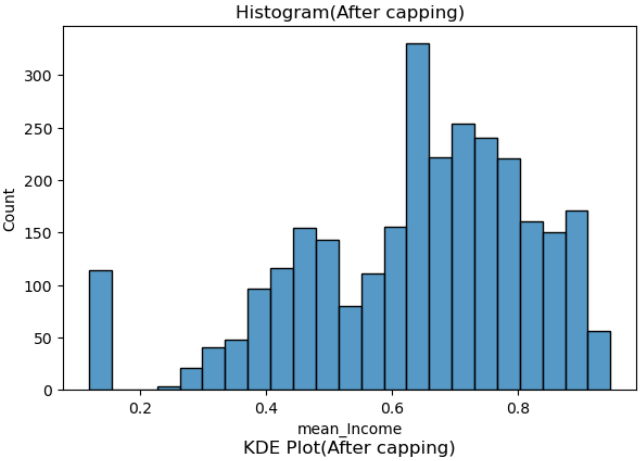


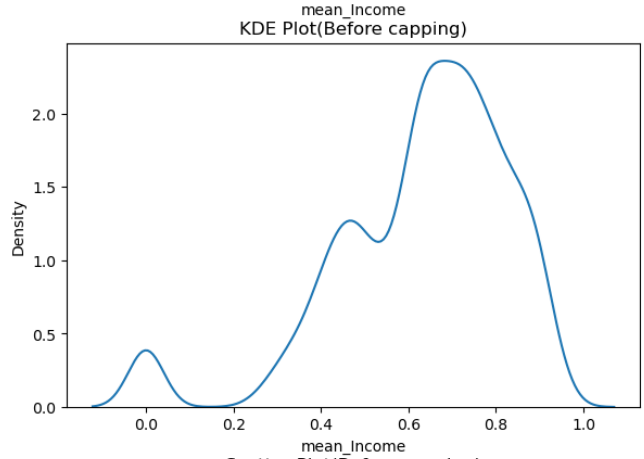
```
##########################################################################

Variable Name :  mean_Income
Persentile 25 :  0.51175
Persentile 75 :  0.774
IQR :  0.26225
Upper Limit :  1.167375
Lower Limit :  0.11837500000000006
Outliers Removed using IQR Method by Capping
```

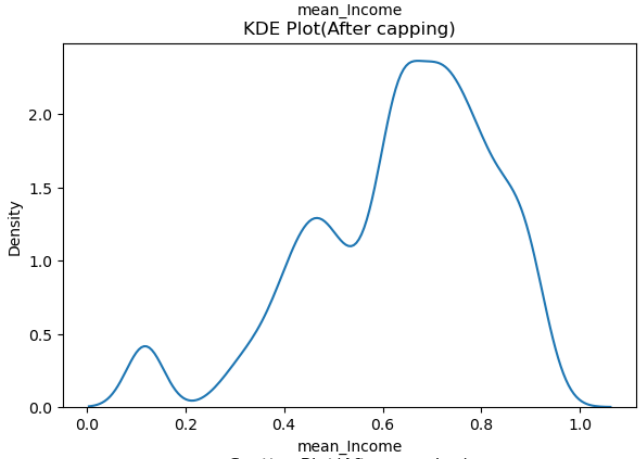


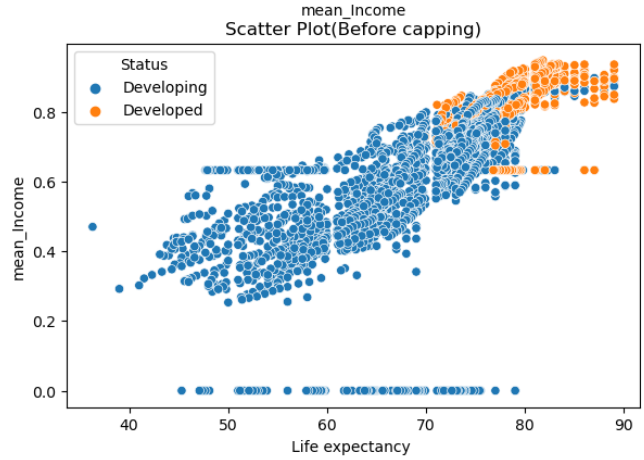


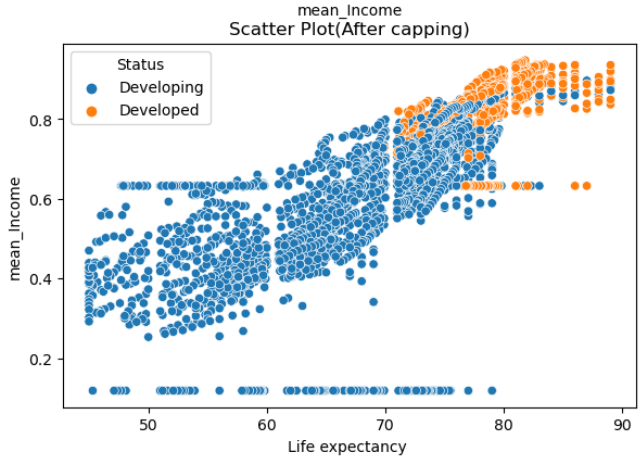




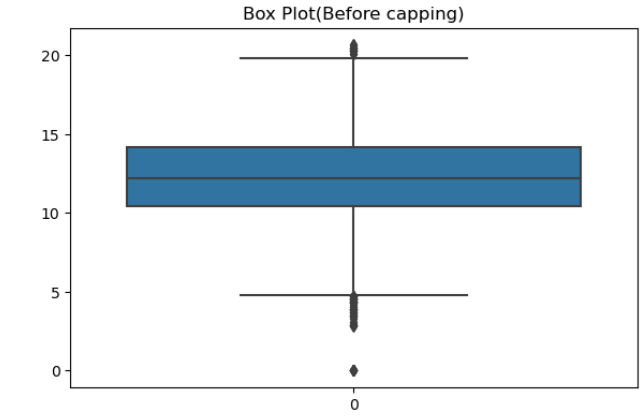


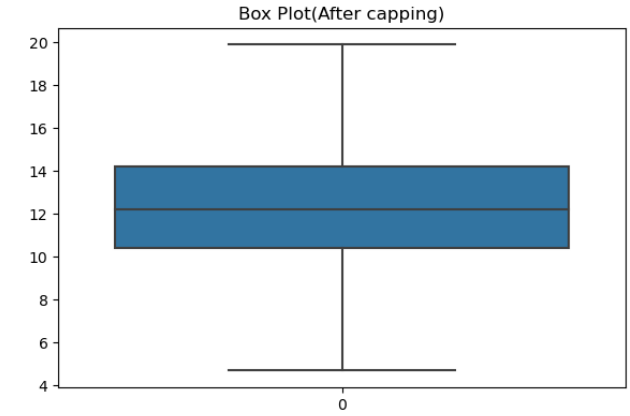


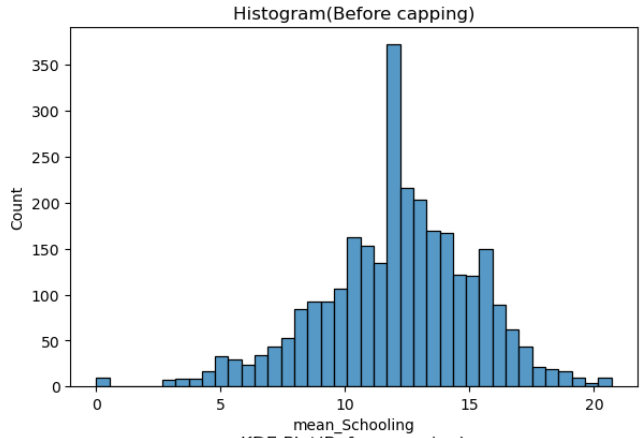


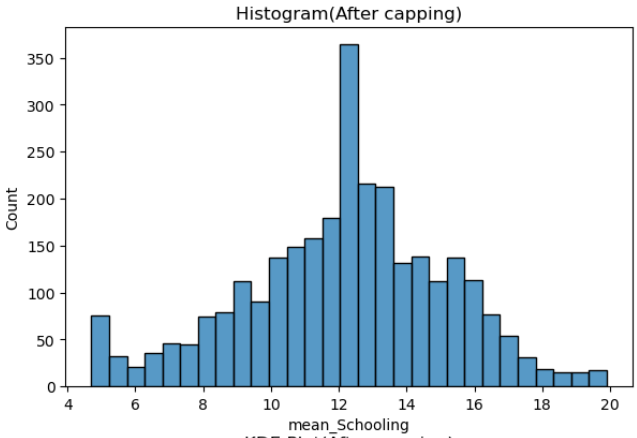


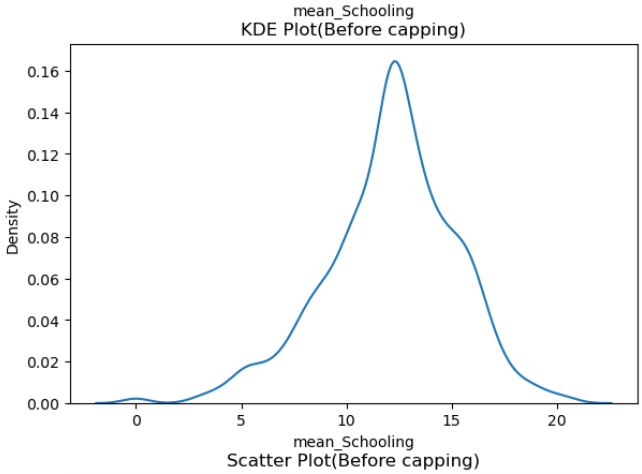
```
##########################################################################

Variable Name :  mean_Schooling
Persentile 25 :  10.4
Persentile 75 :  14.2
IQR :  3.799999999999999
Upper Limit :  19.9
Lower Limit :  4.700000000000002
Outliers Removed using IQR Method by Capping
```

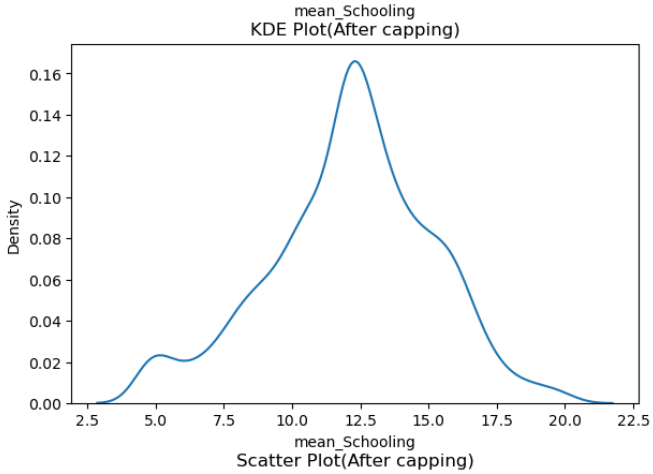


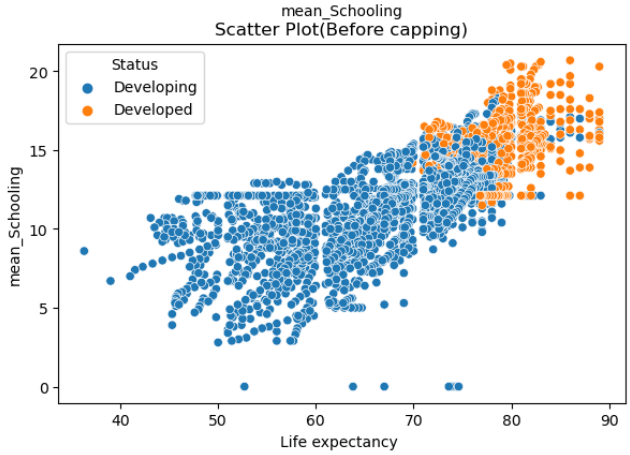


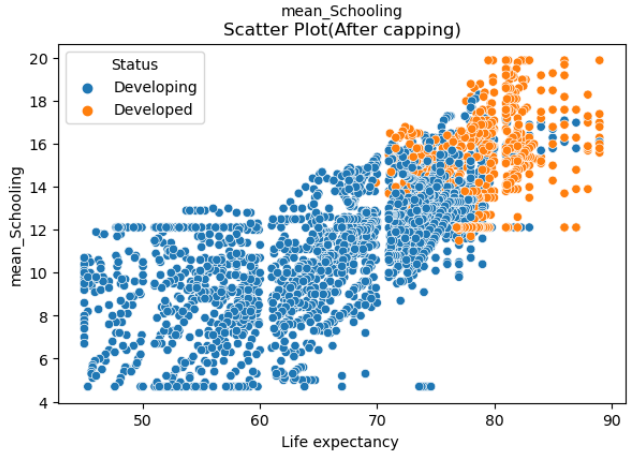












Train Test Split

In [348…]
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(final_df.drop(columns=["Life expectancy "
X_train_without_outliers, X_test_without_outliers, y_train_without_outliers, y_test_without_o
```

In [349…]
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[349]:
```
((2021, 17), (867, 17), (2021,), (867,))
```

In [350…]
```
X_train_without_outliers.shape, X_test_without_outliers.shape, y_train_without_outliers.shape
```

Out[350]:
```
((2021, 17), (867, 17), (2021,), (867,))
```

Encoding

In [351…]
```
trnsformer = ColumnTransformer(transformers=[
    ('oe', OrdinalEncoder(categories=[["Developing", "Developed"]]), ["Status"]),
    ('ohe', OneHotEncoder(sparse=False, drop="first"), ["Country"])
], remainder="passthrough")
```

In [352…]
```
X_train = trnsformer.fit_transform(X_train)
```

In [353…]
```
X_test = trnsformer.transform(X_test)
```

In [354…]
```
X_train_without_outliers = trnsformer.fit_transform(X_train_without_outliers)
```

In [355…]
```
X_test_without_outliers = trnsformer.transform(X_test_without_outliers)
```

Standardization

In [356…]
```
scaler1 = StandardScaler()
X_train_stand=scaler1.fit_transform(X_train)
X_test_stand=scaler1.transform(X_test)
```

In [357…]
```
X_train_stand.shape, X_test_stand.shape
```

Out[357]:
```
((2021, 196), (867, 196))
```

In [358…]
```
X_train_stand
```

Out[358]:
```
array([[-0.46476318, -0.07397727, -0.08933112, ..., -0.37809873,
         0.2651249 ,  0.53723124],
       [-0.46476318, -0.07397727, -0.08933112, ..., -0.0947713 ,
         0.06916371, -0.00773417],
       [-0.46476318, -0.07397727, -0.08933112, ..., -0.45065964,
         0.68154243,  1.11425344],
       ...,
       [-0.46476318, -0.07397727, -0.08933112, ..., -0.37125113,
        -0.01411979,  0.08843619],
       [-0.46476318, -0.07397727, -0.08933112, ...,  0.01957711,
         0.40229774, -0.00773417],
       [-0.46476318, -0.07397727, -0.08933112, ..., -0.49955921,
        -1.40544424, -2.18759582]])
```

In [359…]
```
scaler2 = StandardScaler()
X_train_without_outliers_stand=scaler1.fit_transform(X_train)
X_test_without_outliers_stand=scaler1.transform(X_test)
```

In [360…]
```
X_train_without_outliers_stand.shape, X_test_without_outliers_stand.shape
```

Out[360]:
```
((2021, 196), (867, 196))
```

Model Build

Linear Regression

With outliers without Standardization

In [361…]
```
lr1 = LinearRegression()
```

In [362…]
```
lr1.fit(X_train, y_train)
```

Out[362]:
```
LinearRegression()
```

In [363…]
```
y_pred1 = lr1.predict(X_test)
```

In [364…]
```
lr_accuracy1 = r2_score(y_test, y_pred1)*100
print("Linear Regression accuracy :", lr_accuracy1)
```

```
Linear Regression accuracy : 95.04463395398146
```

With Outliers With Standardization

In [365…]
```
lr2 = LinearRegression()
```

In [366…]
```
lr2.fit(X_train_stand, y_train)
```

Out[366]:
```
LinearRegression()
```

In [367…]
```
y_pred2 = lr2.predict(X_test_stand)
```

In [368…]
```
lr_accuracy2 = r2_score(y_test, y_pred2)*100
print("Linear Regression accuracy :", lr_accuracy2)
```

```
Linear Regression accuracy : 95.04802226704189
```

Without Outliers Without Standardization

In [369…]
```
lr3 = LinearRegression()
```

In [370…]
```
lr3.fit(X_train_without_outliers, y_train_without_outliers)
```

Out[370]:
```
LinearRegression()
```

In [371…]
```
y_pred3 = lr3.predict(X_test_without_outliers)
```

In [372…]
```
lr_accuracy3 = r2_score(y_test_without_outliers, y_pred3)*100
print("Linear Regression accuracy :", lr_accuracy3)
```

```
Linear Regression accuracy : 95.12939133840246
```

Without Outliers With Standardization

In [373…]
```
lr4 = LinearRegression()
```

In [374…]
```
lr4.fit(X_train_without_outliers_stand, y_train_without_outliers)
```

Out[374]:
```
LinearRegression()
```

In [375…]
```
y_pred4 = lr4.predict(X_test_without_outliers_stand)
```

In [376…
```
lr_accuracy4 = r2_score(y_test_without_outliers, y_pred4)*100
print("Linear Regression accuracy :", lr_accuracy4)
```

```
Linear Regression accuracy : 95.44365151821836
```

Random Forest

With outliers without Standardization

In [377…
```
rf1 = RandomForestRegressor(n_estimators=100, random_state=42)
```

In [378…
```
rf1.fit(X_train, y_train)
```

Out[378]:
```
RandomForestRegressor(random_state=42)
```

In [379…
```
y_pred1 = rf1.predict(X_test)
```

In [380…
```
rf_accuracy1 = r2_score(y_test, y_pred1)*100
print("Random Forest Regression accuracy :", rf_accuracy1)
```

```
Random Forest Regression accuracy : 95.69862353631908
```

With Outliers With Standardization

In [381…
```
rf2 = RandomForestRegressor(n_estimators=100, random_state=42)
```

In [382…
```
rf2.fit(X_train_stand, y_train)
```

Out[382]:
```
RandomForestRegressor(random_state=42)
```

In [383…
```
y_pred2 = rf2.predict(X_test_stand)
```

In [384…
```
rf_accuracy2 = r2_score(y_test, y_pred2)*100
print("Random Forest Regression accuracy :", rf_accuracy2)
```

```
Random Forest Regression accuracy : 95.69044887836719
```

Without Outliers Without Standardization

In [385…
```
rf3 = RandomForestRegressor(n_estimators=100, random_state=42)
```

In [386…
```
rf3.fit(X_train_without_outliers, y_train_without_outliers)
```

Out[386]:
```
RandomForestRegressor(random_state=42)
```

In [387…
```
y_pred3 = rf3.predict(X_test_without_outliers)
```

In [388…
```
rf_accuracy3 = r2_score(y_test_without_outliers, y_pred3)*100
print("Random Forest Regression accuracy :", rf_accuracy3)
```

```
Random Forest Regression accuracy : 95.83064149884851
```

Without Outliers With Standardization

In [389…
```
rf4 = RandomForestRegressor(n_estimators=100, random_state=42)
```

In [390…
```
rf4.fit(X_train_without_outliers_stand, y_train_without_outliers)
```

Out[390]:
```
RandomForestRegressor(random_state=42)
```

In [391…
```
y_pred4 = rf4.predict(X_test_without_outliers_stand)
```

In [392…
```
rf_accuracy4 = r2_score(y_test_without_outliers, y_pred4)*100
print("Random Forest Regression accuracy :", rf_accuracy4)
```

```
Random Forest Regression accuracy : 95.93971674136323
```

Decision Tree Regressor

With outliers without Standardization

In [393…
```
tree1 = DecisionTreeRegressor(max_depth=4)
```

In [394…
```
tree1.fit(X_train, y_train)
```

Out[394]:
```
DecisionTreeRegressor(max_depth=4)
```

In [395…
```
y_pred1 = tree1.predict(X_test)
```

In [396…
```
tree_accuracy1=r2_score(y_test, y_pred1)*100
print("Decision Tree Regression accuracy :", tree_accuracy1)
```

```
Decision Tree Regression accuracy : 84.73846452603169
```

With Outliers With Standardization

In [397…
```
tree2 = DecisionTreeRegressor(max_depth=4)
```

In [398…
```
tree2.fit(X_train_stand, y_train)
```

Out[398]:
```
DecisionTreeRegressor(max_depth=4)
```

In [399…
```
y_pred2 = tree2.predict(X_test_stand)
```

In [400…
```
tree_accuracy2=r2_score(y_test, y_pred2)*100
print("Decision Tree Regression accuracy :", tree_accuracy2)
```

```
Decision Tree Regression accuracy : 84.7384645260317
```

Without Outliers Without Standardization

In [401…
```
tree3 = DecisionTreeRegressor(max_depth=4)
```

In [402…
```
tree3.fit(X_train_without_outliers, y_train_without_outliers)
```

Out[402]:
```
DecisionTreeRegressor(max_depth=4)
```

In [403…
```
y_pred3 = tree3.predict(X_test_without_outliers)
```

In [404…
```
tree_accuracy3=r2_score(y_test_without_outliers, y_pred3)*100
print("Decision Tree Regression accuracy :", tree_accuracy3)
```

```
Decision Tree Regression accuracy : 85.15847990148373
```

Without Outliers With Standardization

In [405…
```
tree4 = DecisionTreeRegressor(max_depth=4)
```

In [406…
```
tree4.fit(X_train_without_outliers_stand, y_train_without_outliers)
```

Out[406]:
```
DecisionTreeRegressor(max_depth=4)
```

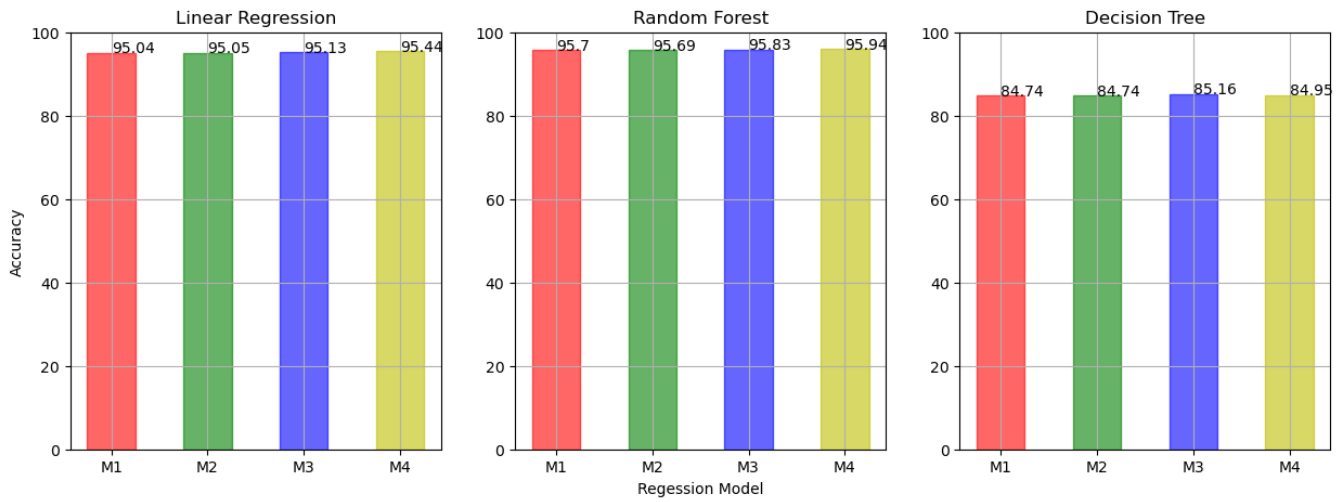
In [407…
```
y_pred4 = tree4.predict(X_test_without_outliers_stand)
```

In [408…
```
tree_accuracy4=r2_score(y_test_without_outliers, y_pred4)*100
print("Decision Tree Regression accuracy :", tree_accuracy4)
```

```
Decision Tree Regression accuracy : 84.94658591739268
```

Accuracy Comprison

In [409…
```
lr_models = ["M1", "M2", "M3", "M4"]
rf_models = ["M1", "M2", "M3", "M4"]
tree_models = ["M1", "M2", "M3", "M4"]

lr_model_accuracy = [lr_accuracy1, lr_accuracy2, lr_accuracy3, lr_accuracy4]
rf_model_accuracy = [rf_accuracy1, rf_accuracy2, rf_accuracy3, rf_accuracy4]
tree_model_accuracy = [tree_accuracy1, tree_accuracy2, tree_accuracy3, tree_accuracy4]

def addlabels(x,y):
    for i in range(len(x)):
        plt.text(i,y[i],round(y[i], 2))

plt.figure(figsize=(15,5))
plt.subplot(1, 3, 1)
plt.title("Linear Regression")
barlist = plt.bar(lr_models, lr_model_accuracy, width=0.5, alpha=0.6)
addlabels(lr_models, lr_model_accuracy)
plt.ylim(0, 100)
barlist[0].set_color('r')
barlist[1].set_color('g')
barlist[2].set_color('b')
barlist[3].set_color('y')
plt.ylabel("Accuracy")
plt.grid()

plt.subplot(1, 3, 2)
plt.title("Random Forest")
barlist = plt.bar(rf_models, rf_model_accuracy, width=0.5, alpha=0.6)
addlabels(rf_models, rf_model_accuracy)
plt.ylim(0, 100)
barlist[0].set_color('r')
barlist[1].set_color('g')
barlist[2].set_color('b')
barlist[3].set_color('y')
plt.grid()

plt.xlabel("Regession Model")

plt.subplot(1, 3, 3)
plt.title("Decision Tree")
barlist = plt.bar(tree_models, tree_model_accuracy, width=0.5, alpha=0.6)
addlabels(tree_models, tree_model_accuracy)
plt.ylim(0, 100)
barlist[0].set_color('r')
barlist[1].set_color('g')
barlist[2].set_color('b')
barlist[3].set_color('y')

plt.grid()
plt.show()
```



Final model

- **From the upper caht we select the best accurate model is - Random Forest(Without Outliers and with Standardization)**

In [412…
```
print("Accuracy :", rf_accuracy4)
```

```
Accuracy : 95.93971674136323
```

In [ ]:

DEPARTMENT OF INFORMATION TECHNOLOGY

ASANSOL ENGINEERING COLLEGE

Certificate

This is to certify that Mr. BIPLAB GORAIN department of Information Technology of Asansol Engineering College registration number: 211080100220004 has successfully completed a project on “Life Expectancy Prediction” using machine learning with python under the guidance of Prof. Arnab Chakraborty.

Signature of
The Project Guide
Prof. Arnab Chakraborty

Recommendation & Signature of
The Principal
Dr. P.P. Bhattacharya

Recommendation &Signature of
The Head of Department
Dr. Anup Kumar Mukhopadhay

Recommendation &Signature of
Internal / External Examiners

DEPARTMENT OF INFORMATION TECHNOLOGY

ASANSOL ENGINEERING COLLEGE

Certificate

This is to certify that Ms. SHEELA BHATTACHARJEE department of Information Technology of Asansol Engineering College registration number: 201080100210018 has successfully completed a project on “Life Expectancy Prediction” using machine learning with python under the guidance of Prof. Arnab Chakraborty.

Signature of
The Project Guide
Prof. Arnab Chakraborty

Recommendation & Signature of
The Principal
Dr. P.P. Bhattacharya

Recommendation &Signature of
The Head of Department
Dr. Anup Kumar Mukhopadhay

Recommendation &Signature of
Internal / External Examiners

DEPARTMENT OF INFORMATION TECHNOLOGY

ASANSOL ENGINEERING COLLEGE

Certificate

This is to certify that Mr. BISHNU SHARMA department of Information Technology of Asansol Engineering College registration number: 201080100210019 has successfully completed a project on “Life Expectancy Prediction” using machine learning with python under the guidance of Prof. Arnab Chakraborty.

Signature of
The Project Guide
Prof. Arnab Chakraborty

Recommendation & Signature of
The Principal
Dr. P.P. Bhattacharya

Recommendation &Signature of
The Head of Department
Dr. Anup Kumar Mukhopadhay

Recommendation &Signature of
Internal / External Examiners

DEPARTMENT OF INFORMATION TECHNOLOGY

ASANSOL ENGINEERING COLLEGE

Certificate

This is to certify that Ms. PRIYANSHU BURMAN department of Information Technology of Asansol Engineering College registration number: 201080100210026 has successfully completed a project on “Life Expectancy Prediction” using machine learning with python under the guidance of Prof. Arnab Chakraborty.

Signature of
The Project Guide
Prof. Arnab Chakraborty

Recommendation & Signature of
The Principal
Dr. P.P. Bhattacharya

Recommendation &Signature of
The Head of Department
Dr. Anup Kumar Mukhopadhay

Recommendation &Signature of
Internal / External Examiners