Table of Contents:

Students, course, and instructor information

#Student name: HaoYu Tan, WeiGuang Chen, JianHeng Chen
#Student ID: 1006147386, 999187259,1005680746
#Instructor: Dr. Maher Elshakankiri
#Course code: INF 1340
#Course name: Programming for Data Science
#Program: MI
#Faculty of Information
#University of Toronto

Project Description

Objective
Our project sought to uncover the intricate relationships between the Human Development Index (HDI) and various socioeconomic indicators across different nations. We aimed to discern how factors such as life expectancy, education (mean and expected years of schooling), and Gross National Income (GNI) per capita interact to shape the overall human development landscape.

Methodology
Our analytical approach was subdivided into several key segments:

Descriptive Analytics: We commenced with summary statistics and visualization to grasp the distribution and central tendencies of our data, such as HDI, life expectancy, and GNI per capita. Ranking and sorting processes were implemented to understand the ordinal position of nations within these metrics.

Diagnostic Analytics: The team dove into diagnostic analysis to identify patterns and outliers within the data, seeking to understand the causes of anomalies and to compare the top-performing countries across various indicators.

Predictive Analytics: Through model validation methods, including train-test split, we analyzed the predictability of HDI using variables such as life expectancy and education levels. Multiple linear regressions were conducted to establish the degree of influence each factor has on HDI.

Mapping and Quantile Analysis: After correlation analysis, we employed a mapping process using quantile thresholds to categorize HDI, life expectancy, and schooling. We investigated the alignment of HDI with these factors to understand their distribution across different quantiles.

Key Findings

The analysis demonstrated that while life expectancy and education are strongly correlated with HDI, their relationship with economic wealth, as denoted by GNI per capita, is not always linear.

Discrepancies in ranks between life expectancy, schooling, and HDI indicate diverse aspects of human development, suggesting that higher ranking in one aspect does not always mean higher rankings in others.

The correlation between HDI and other development indicators like life expectancy and education is strong, yet nuanced, varying across different countries.

Our mapping process highlighted that, despite close correlations, the distribution of these indicators across countries may not align perfectly when using quantile-based thresholds.

4. Features

- Comprehensive Data Collection: Utilizes a rich dataset encompassing various socioeconomic indicators such as HDI, life expectancy, education levels, and GNI per capita.

- Summary Statistics and Visualization: Provides a summary of key statistical measures and visual representations to understand the central tendencies and distribution of data.

- Ranking and Sorting Analysis: Includes functionality to rank and sort countries based on different indicators, facilitating a comparative analysis of human development across nations.

- Data Distribution, Skewness, and Kurtosis Analysis: Examines the distribution characteristics of the dataset, including skewness and kurtosis, to understand the symmetry and tail behavior of the data distribution.

- Outlier Detection: Identifies and analyzes outliers in the dataset, providing insights into countries that deviate significantly from the norm.

- Linear Regression Models: Features multiple linear regression analyses to explore the relationships between HDI and various socioeconomic factors.

- Predictive Analytics: Implements predictive modeling to forecast HDI based on key indicators, using techniques like train-test split for model validation.

- Diagnostic Analytics: Delivers in-depth diagnostics to explore patterns, relationships, and correlations among different development indicators.

- Mapping and Quantile Analysis: Categorizes data based on quantile thresholds for a more nuanced analysis, mapping HDI against life expectancy and education levels.

- Rank Comparison and Correlation Analysis: Compares the ranks of various indicators with HDI, analyzing the differences and correlations to draw meaningful conclusions.

- Multifaceted Interpretation and Insights: Provides a holistic view of human development, emphasizing the complexity and interconnectedness of various development indicators.

- Interactive and User-Friendly Interface: Ensures ease of use with clear prompts and instructions, enabling users to engage with the analysis effectively.

Usage

The following steps were undertaken to conduct our analysis on the Human Development Index (HDI) and its associated socioeconomic indicators:

1. Data Preparation:
   - The dataset was loaded from the 'cleaned_data_final.csv' file, which had been pre-processed to ensure quality and consistency.

2. Statistical Analysis Execution:
   - Summary statistics were computed for various indicators to assess central tendencies and data distribution, including mean, median, maximum, minimum, and standard deviation values.

3. Visualization:
   - Data visualizations such as histograms and kernel density estimates were generated to illustrate the distribution of HDI values and other indicators across different countries.

4. Diagnostic Analysis:
   - The team performed diagnostic analytics, including ranking and sorting of countries based on selected indicators and identifying outliers within the dataset.

5. Correlation and Regression Analysis:
   - Correlation coefficients were calculated to explore the relationships between HDI and life expectancy, education, and GNI per capita. Regression analyses were conducted to understand the impact of these indicators on HDI.

6. Mapping and Quantile Analysis:
   - Quantile thresholds were established to categorize the data and map HDI against life expectancy and mean years of schooling. This allowed the team to analyze how these indicators correspond to HDI levels.

7. Reporting:
   - The findings were compiled into an analytical report detailing the relationships and patterns observed. The report included a discussion on the implications of these findings for understanding human development.

Each step was meticulously documented to ensure the reproducibility of the results and to maintain the integrity of the analysis. The insights gained from this study provide a comprehensive understanding of the factors that influence human development across nations.

Execution Instructions

To run the data analysis scripts, follow these steps:

1. Ensure Python is installed on your system. These scripts were developed under Python

2. Place the scripts in the desired directory on your local machine.

3. The scripts are expected to operate on a dataset named 'cleaned_data_final.csv'. Ensure this file is in the same directory as the scripts or modify the script to point to the correct path of your dataset.

4. Open the command prompt window and enter ‘cleaned_data_final.csv' file

5. The scripts will execute and display the results in the terminal or command prompt window. If the scripts are designed to save output to a file, check the directory for the generated files.

6. Review the output carefully for insights and ensure the results align with the project's objectives.

Example of Running the Program

Descriptive Analyses:

Task 1 is to summarize statistics in the file, except country and HDI rank, to run this code you have to enter the 'cleaned_data_final.csv' file

```
import numpy as np
import pandas as pd

# Task 1
data = pd.read_csv("cleaned_data_final.csv", index_col='Country')
pd.set_option('display.max_columns', None)
# Head for df
print(data.head(10))
# Provide the types for df
print(data.dtypes)
# Summary statistic for df, except Country and HDI rank
print(data.drop('HDI Rank', axis = 1).describe())
```

Output:

```
Human Development Index (HDI)  \
Country
Switzerland                    0.962
Norway                         0.961
Iceland                        0.959
Hong Kong, China (SAR)                    0.952
Australia                      0.951
Denmark                        0.948
Sweden                         0.947
Ireland                        0.945
Germany                        0.942
Netherlands                    0.941
```

Country	Life expectancy at birth	Expected years of schooling \
Switzerland	83.9872	16.500299
Norway	83.2339	18.185200
Iceland	82.6782	19.163059
Hong Kong, China (SAR)	85.4734	17.278170
Australia	84.5265	21.054590
Denmark	81.3753	18.714800
Sweden	82.9833	19.418530
Ireland	81.9976	18.945221
Germany	80.6301	17.010139
Netherlands	81.6873	18.693165

Country	Mean years of schooling \
Switzerland	13.859660
Norway	13.003630
Iceland	13.767170
Hong Kong, China (SAR)	12.226210
Australia	12.726820
Denmark	12.960490
Sweden	12.609720
Ireland	11.582223
Germany	14.090967
Netherlands	12.581630

Country	Gross national income (GNI) per capita \
Switzerland	66933.00454
Norway	64660.10622
Iceland	55782.04981
Hong Kong, China (SAR)	62606.84540
Australia	49238.43335
Denmark	60364.78595
Sweden	54489.37401
Ireland	76168.98443
Germany	54534.21682
Netherlands	55979.41100

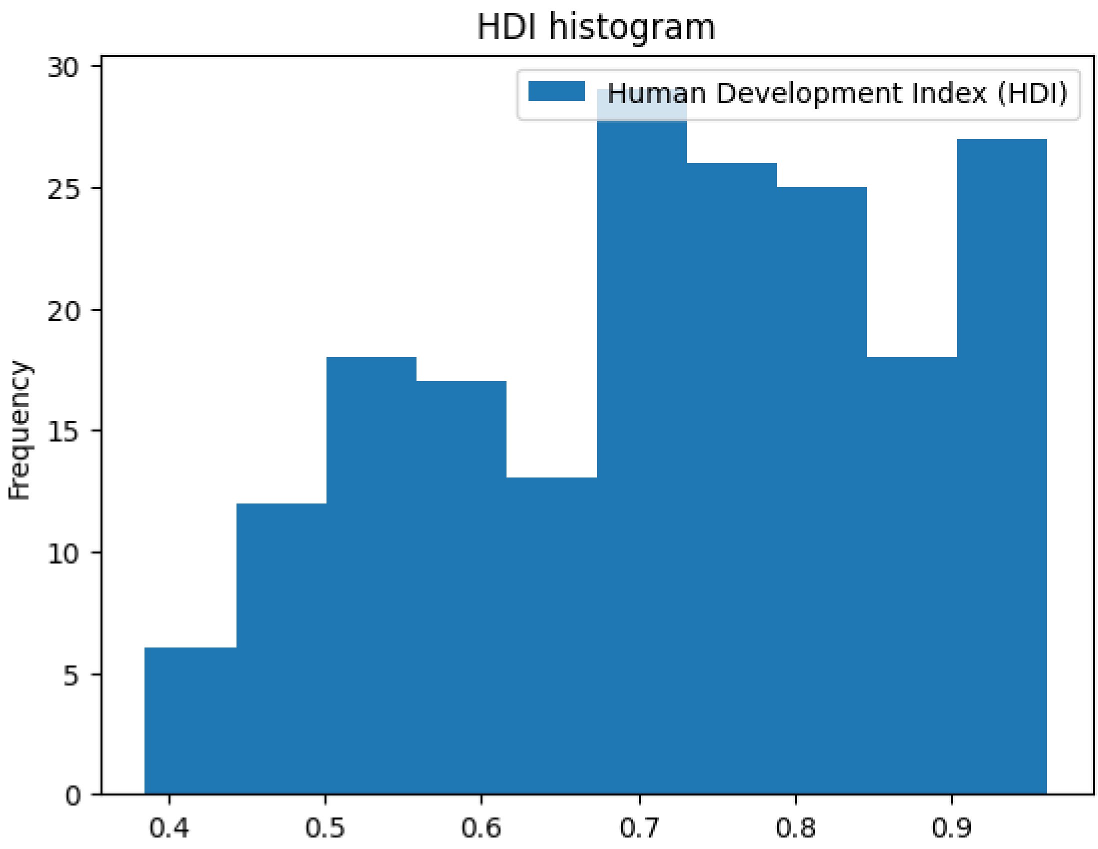
Country	GNI per capita rank minus HDI rank	HDI Rank
Switzerland	5	3
Norway	6	1
Iceland	11	2
Hong Kong, China (SAR)	6	4
Australia	18	5
Denmark	6	5
Sweden	9	9
Ireland	-3	8

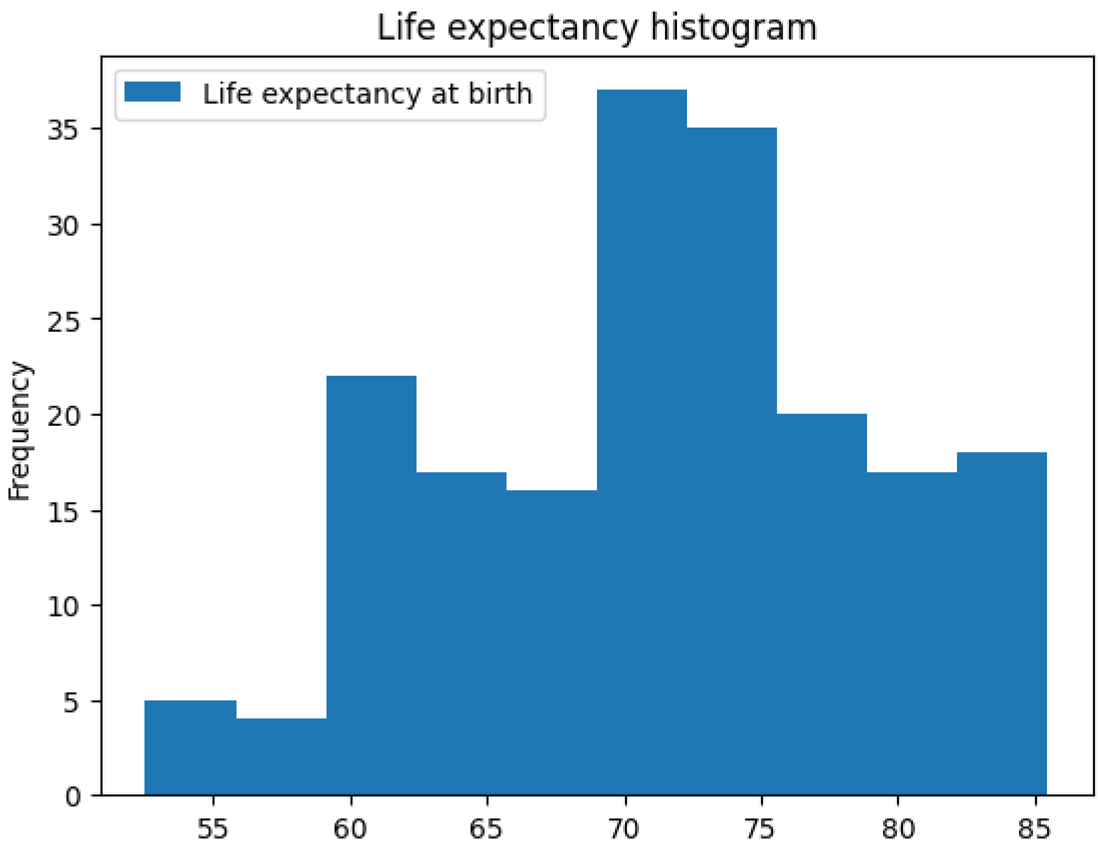
```
Germany                                    6      7
Netherlands                                3     10
Human Development Index (HDI)           float64
Life expectancy at birth                float64
Expected years of schooling             float64
Mean years of schooling                 float64
Gross national income (GNI) per capita  float64
GNI per capita rank minus HDI rank        int64
HDI Rank                                  int64
dtype: object
       Human Development Index (HDI)  Life expectancy at birth  \
count                     191.000000                 191.00000
mean                        0.720576                  71.31286
std                         0.150661                   7.64596
min                         0.385000                  52.52540
25%                         0.599500                  65.74720
50%                         0.739000                  71.69400
75%                         0.835000                  76.69930
max                         0.962000                  85.47340

       Expected years of schooling  Mean years of schooling  \
count                   191.000000               191.000000
mean                     13.534658                 8.986916
std                       2.923911                 3.173693
min                       5.542510                 2.114962
25%                      11.601258                 6.251659
50%                      13.404920                 9.306864
75%                      15.623665                11.497702
max                      21.054590                14.090967

       Gross national income (GNI) per capita  \
count                              191.000000
mean                             20249.088223
std                              21825.277076
min                                731.786709
25%                               4592.919612
50%                              12306.341000
75%                              30079.789725
max                             146829.700600

       GNI per capita rank minus HDI rank
count                          191.000000
mean                             0.188482
std                             14.060249
min                            -47.000000
25%                             -8.000000
50%                              2.000000
75%                              9.000000
max                             37.000000
```

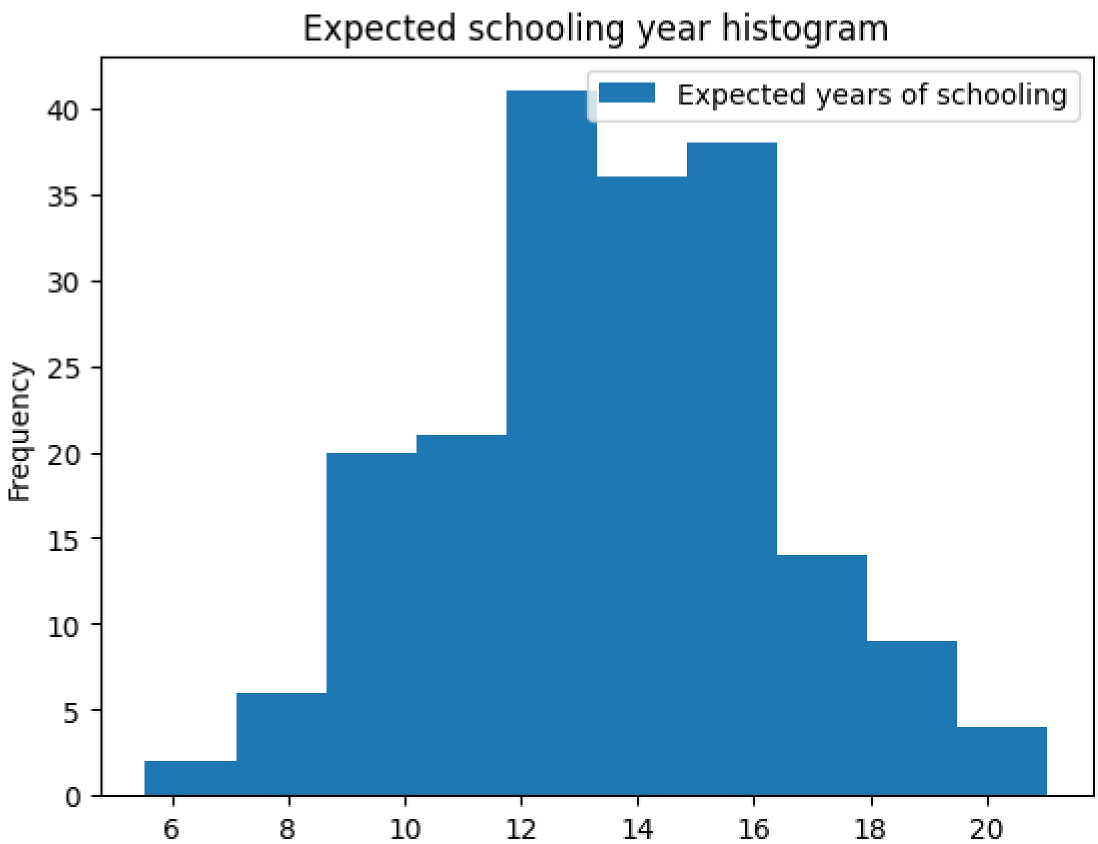
Task 2: Diagrams

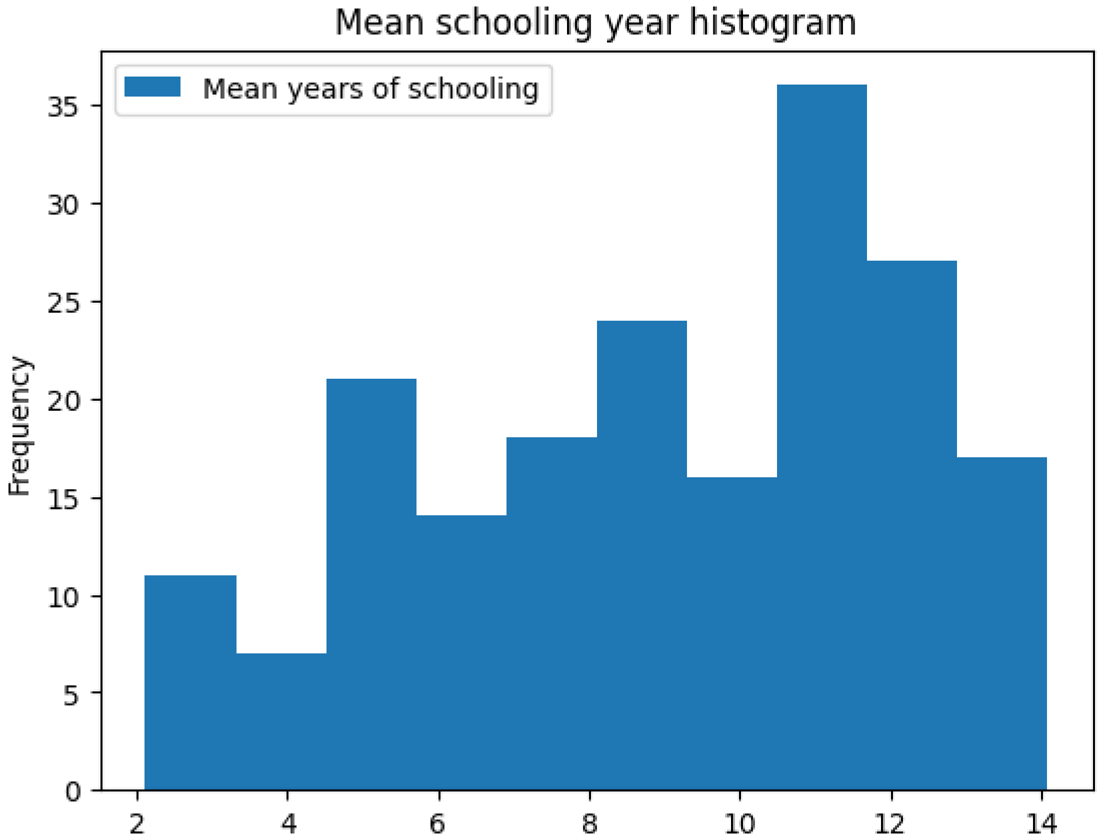
```
# Task 2
# For columns, create histograms
data[['Human Development Index (HDI)']].plot(kind='hist', title = 'HDI
histogram')
data[['Life expectancy at birth']].plot(kind='hist', title = 'Life
expectancy histogram')
data[['Expected years of schooling']].plot(kind='hist', title =
'Expected schooling year histogram')
data[['Mean years of schooling']].plot(kind='hist', title = 'Mean
schooling year histogram')
data[['Gross national income (GNI) per capita']].plot(kind='hist',
title = 'GNI per capita histogram')
data[['GNI per capita rank minus HDI rank']].plot(kind='hist', title =
'GNI per capita-HDI rank histogram')
```

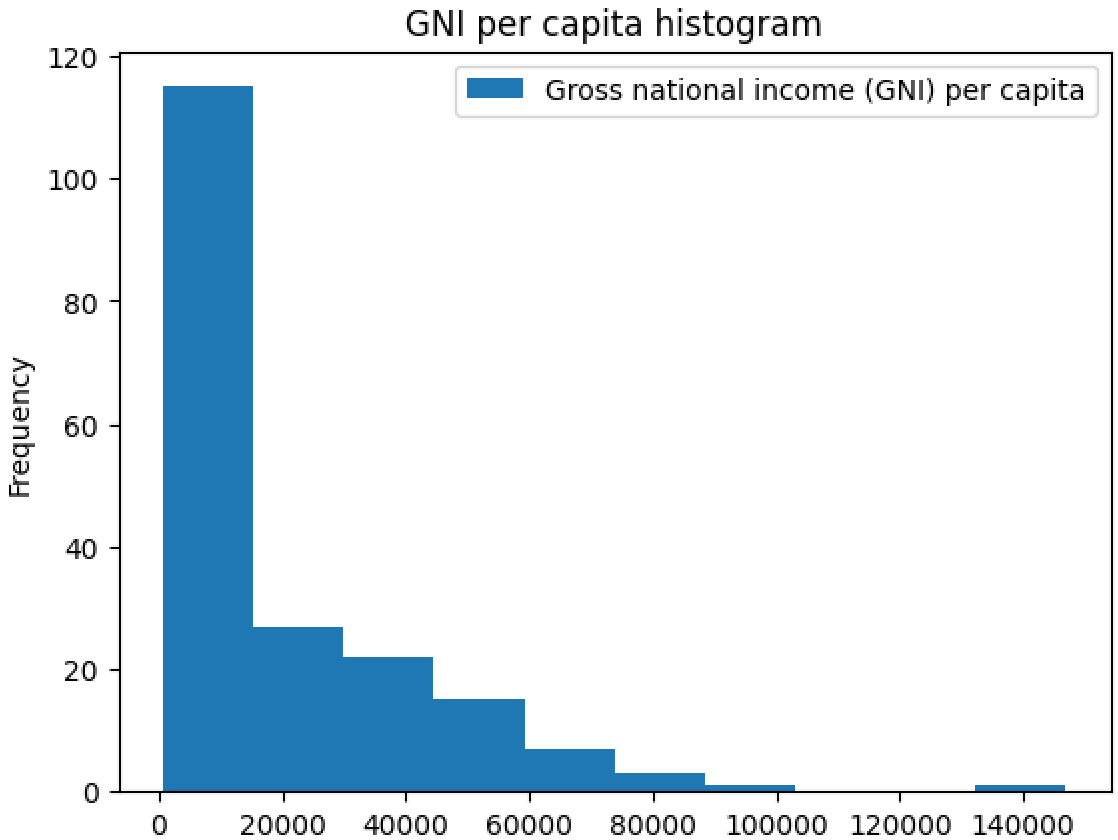
Output:

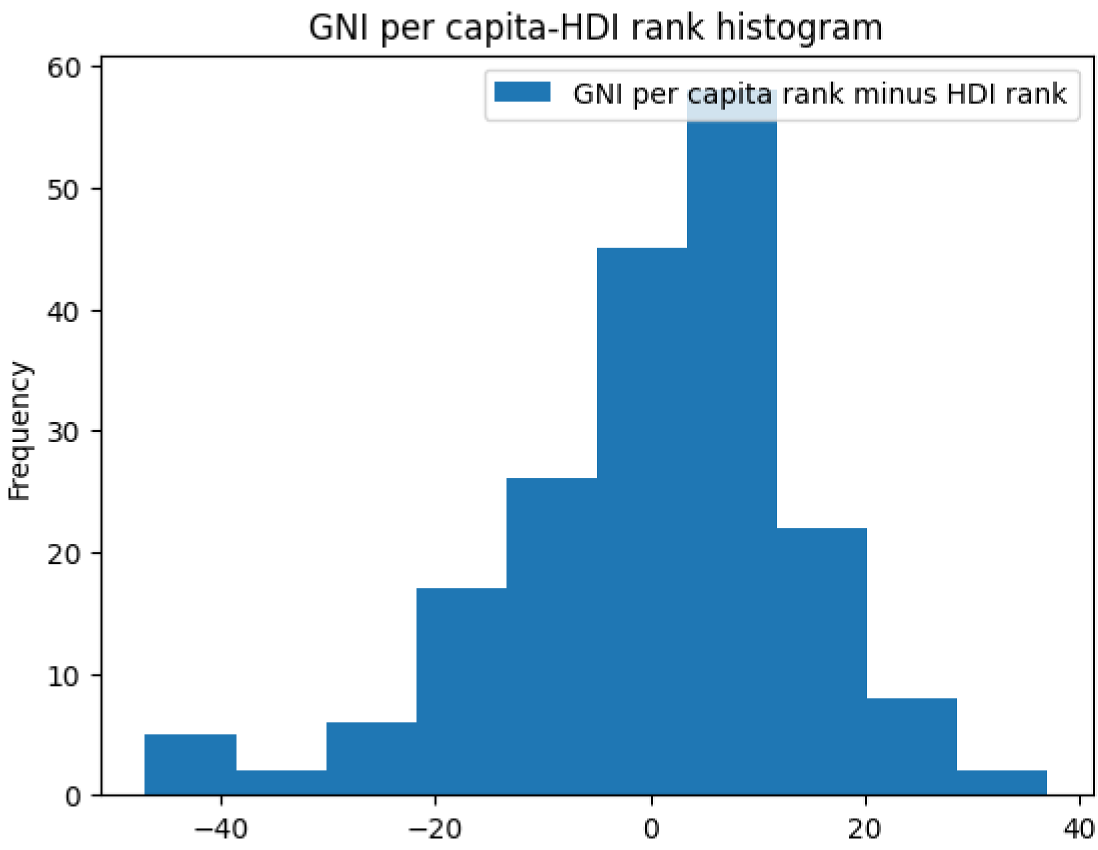



Life expectancy histogram
Life expectancy at birth
Frequency
0
5
10
15
20
25
30
35
55
60
65
70
75
80
85


Expected schooling year histogram
Expected years of schooling
Frequency
0
5
10
15
20
25
30
35
40
6
8
10
12
14
16
18
20


Mean schooling year histogram
Mean years of schooling
Frequency
35
30
25
20
15
10
5
0
2
4
6
8
10
12
14


GNI per capita histogram
Gross national income (GNI) per capita
Frequency
120
100
80
60
40
20
0
0
20000
40000
60000
80000
100000
120000
140000



This code is to find the highest and the lowest values of our analysis, if the code runs successfully you can see the results.

```
# By sorting values to figure out which country has the highest and lowest value
print(data['Human Development Index (HDI)'].sort_values())
print(data['GNI per capita rank minus HDI rank'].sort_values())
print(data['Life expectancy at birth'].sort_values())
print(data['Expected years of schooling'].sort_values())
print(data['Mean years of schooling'].sort_values())
print(data['Gross national income (GNI) per capita'].sort_values())
```

Output:

```
Country
South Sudan            0.385
Chad                   0.394
Niger                  0.400
Central African Republic    0.404
Burundi                0.426
                     ...
Australia              0.951
Hong Kong, China (SAR)      0.952
Iceland                0.959
Norway                 0.961
```

```
Switzerland              0.962
Name: Human Development Index (HDI), Length: 191, dtype: float64
Country
Guyana            -47
Equatorial Guinea   -47
Botswana          -43
Brunei Darussalam   -42
Qatar             -39
                 ..
Samoa             24
Kyrgyzstan        26
Barbados          26
Tonga             34
Cuba              37
Name: GNI per capita rank minus HDI rank, Length: 191, dtype: int64
Country
Chad                     52.5254
Nigeria                  52.6760
Lesotho                  53.0620
Central African Republic    53.8947
South Sudan              54.9752
                          ...
Malta                    83.7769
Switzerland              83.9872
Australia                84.5265
Japan                    84.7839
Hong Kong, China (SAR)     85.4734
Name: Life expectancy at birth, Length: 191, dtype: float64
Country
South Sudan     5.542510
Niger          6.957112
Mali           7.423038
Djibouti       7.432980
Sudan          7.945188
              ...
Sweden         19.418530
Belgium        19.604219
Greece         20.028790
New Zealand    20.283890
Australia      21.054590
Name: Expected years of schooling, Length: 191, dtype: float64
Country
Burkina Faso     2.114962
Niger            2.116717
Guinea           2.202126
Mali             2.310000
Chad             2.573774
                ...
United States    13.683430
Iceland          13.767170
Canada           13.834427
Switzerland      13.859660
```

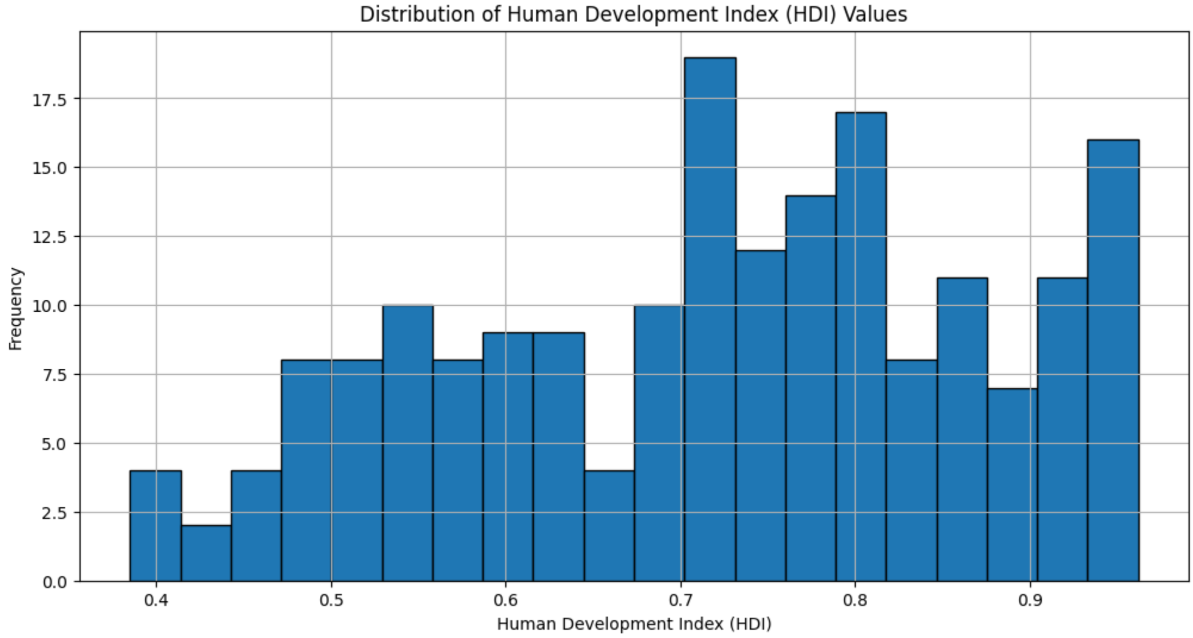
```
Germany          14.090967
Name: Mean years of schooling, Length: 191, dtype: float64
Country
Burundi                              731.786709
South Sudan                          767.787000
Central African Republic             966.058611
Congo (Democratic Republic of the)   1076.098781
Mozambique                           1198.073924
                                      ...
Ireland                              76168.984430
Luxembourg                           84649.474670
Qatar                                87134.134690
Singapore                            90918.644710
Liechtenstein                        146829.700600
Name: Gross national income (GNI) per capita, Length: 191, dtype: float64
```

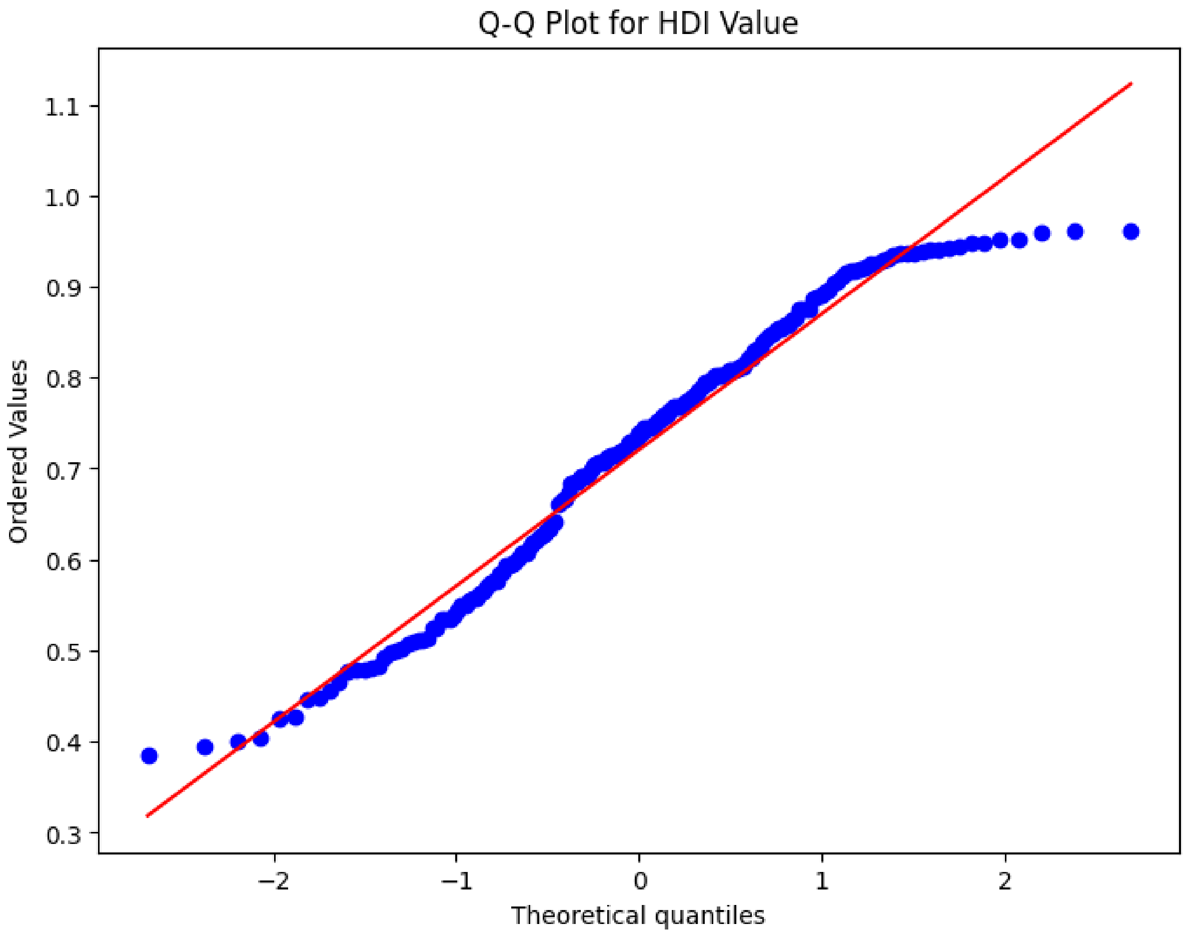
Task 3:

```
# Task 3
# Part of data distribution
import matplotlib.pyplot as plt
import scipy.stats as stats
# Plotting the histogram
plt.figure(figsize=(12, 6))
plt.hist(data['Human Development Index (HDI)'], bins=20,
edgecolor='black')
plt.title('Distribution of Human Development Index (HDI) Values')
plt.xlabel('Human Development Index (HDI)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
# Creating the Q-Q plot
plt.figure(figsize=(8, 6))
stats.probplot(data['Human Development Index (HDI)'], dist="norm",
plot=plt)
plt.title('Q-Q Plot for HDI Value')
plt.show()
# Calculating and printing skewness and kurtosis
skewness = data['Human Development Index (HDI)'].skew()
kurtosis = data['Human Development Index (HDI)'].kurtosis()
# Shapiro-Wilk Test for normality
shapiro_test = stats.shapiro(data['Human Development Index (HDI)'])
# Printing skewness, kurtosis, and Shapiro-Wilk Test result
print(f"Skewness: {skewness}")
print(f"Kurtosis: {kurtosis}")
```

```
print(f"Shapiro-Wilk Test: W={shapiro_test.statistic},
pvalue={shapiro_test.pvalue}")
```

Output:





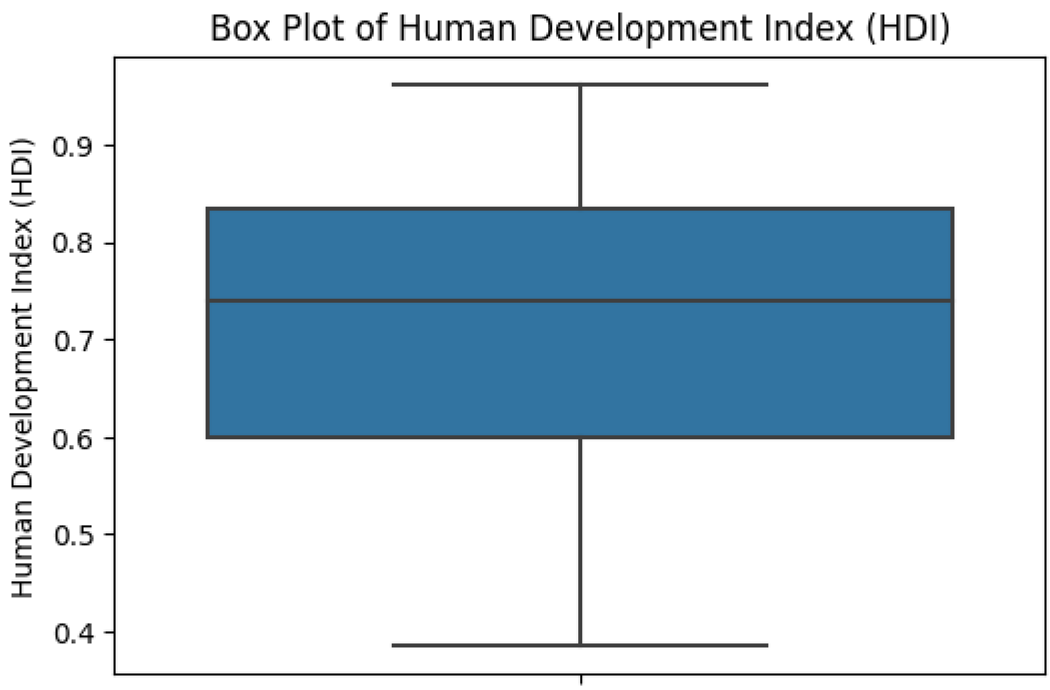
```
Skewness: -0.2810486251677713
Kurtosis: -0.8695813819786249
Shapiro-Wilk Test: W=0.9647785425186157, pvalue=0.00010099347127834335
```

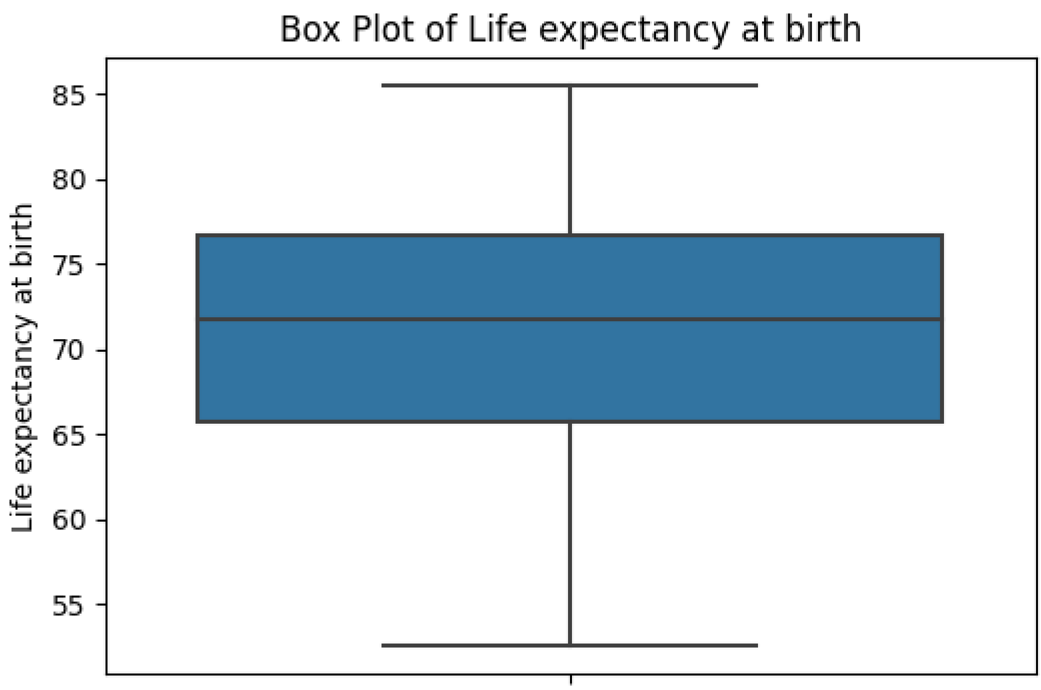
Task 4:

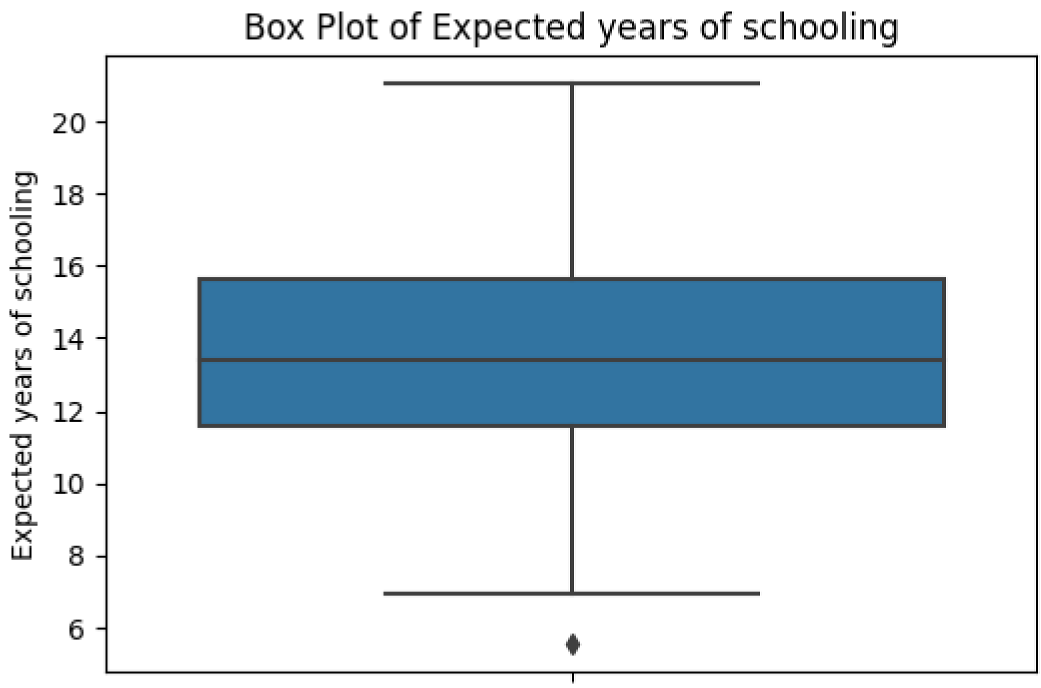
```python
#Task4: Boxplots and outliers for variables

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Setting up the figure size for each box plot
fig_size = (6, 4)

# Create the box plot for the Human Development Index (HDI)
# the new figure for the specified size
plt.figure(figsize=fig_size)
# the box plot created by the seaborn, which will automatically add the
outliers
sns.boxplot(data= data, y='Human Development Index (HDI)')
# write the tittle for the box plot
plt.title('Box Plot of Human Development Index (HDI)')
#displaying the polot on the screen
plt.show()

# create a box plot for Life expectancy at birth
plt.figure(figsize=fig_size)
sns.boxplot(data=data, y='Life expectancy at birth')
plt.title('Box Plot of Life expectancy at birth')
plt.show()

# create a box plot for Expected years of schooling
plt.figure(figsize=fig_size)
sns.boxplot(data=data, y='Expected years of schooling')
plt.title('Box Plot of Expected years of schooling')
plt.show()

# create a box plot for Mean years of schooling
plt.figure(figsize=fig_size)
sns.boxplot(data=data, y='Mean years of schooling')
plt.title('Box Plot of Mean years of schooling')
plt.show()
```

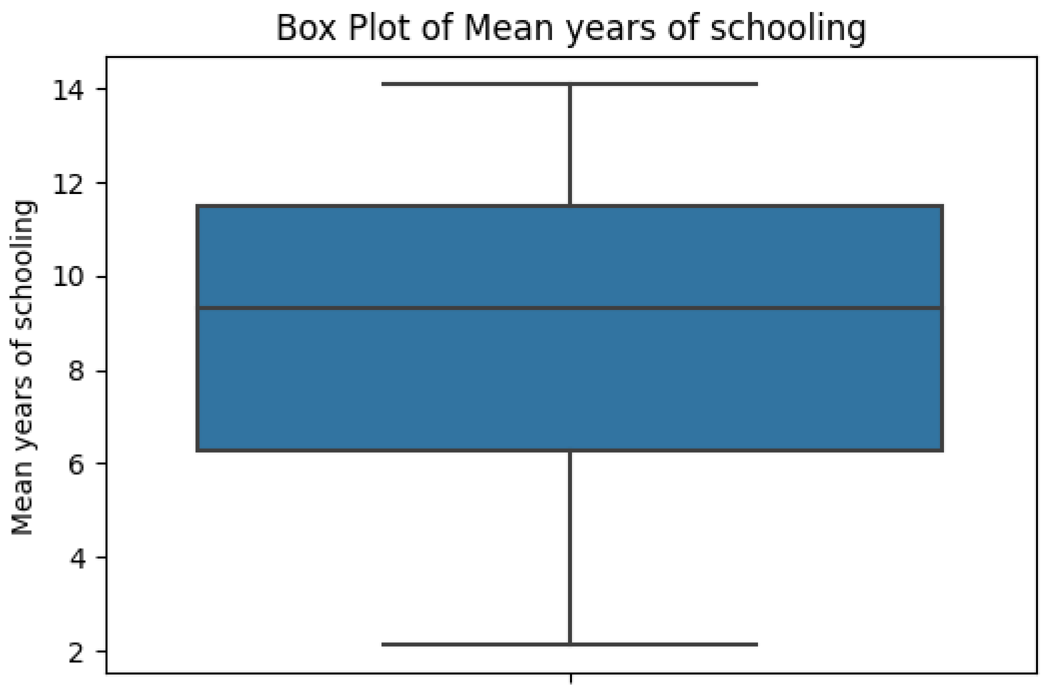
```
# create a box plot for Gross national income (GNI) per capita
plt.figure(figsize=fig_size)
sns.boxplot(data=data, y='Gross national income (GNI) per capita')
plt.title('Box Plot of Gross national income (GNI) per capita')
plt.show()
```

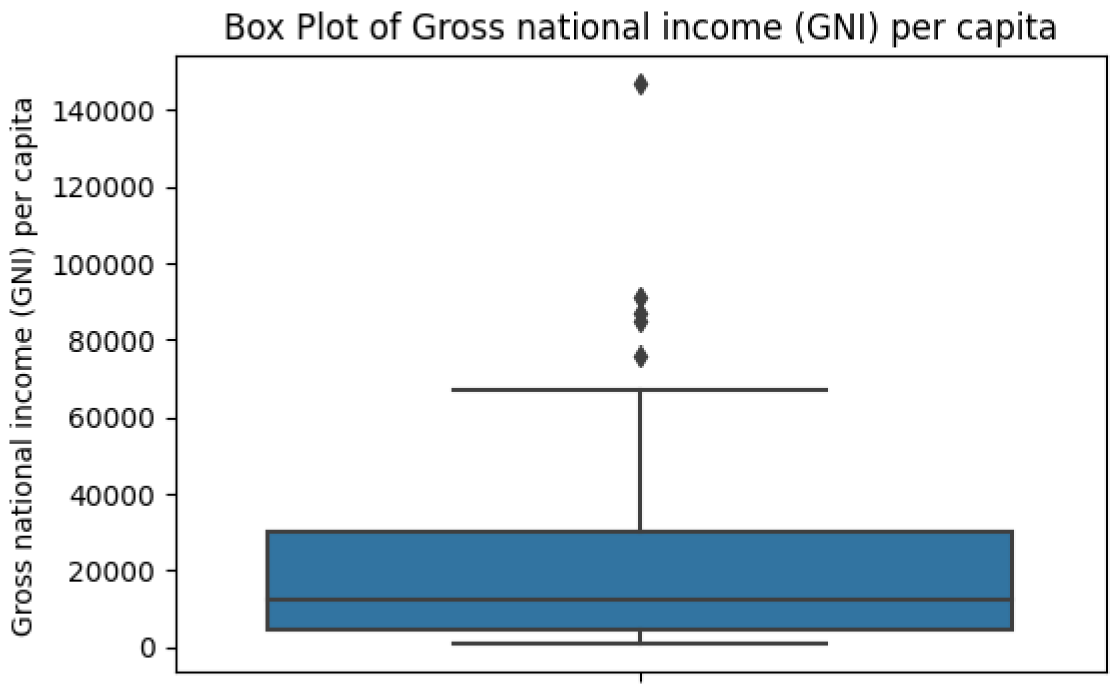
Output:






Box Plot of Expected years of schooling
Expected years of schooling
6
8
10
12
14
16
18
20


Box Plot of Mean years of schooling
Mean years of schooling
2
4
6
8
10
12
14



Task5:

```
#Task5: sorting and ranking
#sort the countries with different income groups
data.reset_index(inplace = True)
# Assigning income groups based on GNI per Capita
data['Income Group'] = ['High' if gni > 50000 else 'Middle' if gni >
20000 else 'Low' for gni in data['Gross national income (GNI) per
capita']]

# Create an empty DataFrame to store the sorted data
sorted_data = pd.DataFrame(data)

# List of unique income groups
income_groups = data['Income Group'].unique()

# Iterate through income groups
for income_group in income_groups:
    # Filter data for the current income group
    income_group_data = data[data['Income Group'] == income_group]

    # Sort the data within the income group by HDI in descending order
    sorted_income_group_data = income_group_data.sort_values('Human
Development Index (HDI)', ascending=False)

    # Append the sorted data to the result DataFrame
```

```
    sorted_data = pd.concat([sorted_data, sorted_income_group_data],
ignore_index=True)

# Resetting index without adding it as a column
sorted_data.reset_index(drop=True, inplace=True)

# Extracting the HDI Rank, Country Name, and Income Group for the top
10 countries in each income group
top_10_high_income = sorted_data[sorted_data['Income Group'] ==
'High'][['HDI Rank', 'Country', 'Income Group']].head(10)
top_10_middle_income = sorted_data[sorted_data['Income Group'] ==
'Middle'][['HDI Rank', 'Country', 'Income Group']].head(10)
top_10_low_income = sorted_data[sorted_data['Income Group'] ==
'Low'][['HDI Rank', 'Country', 'Income Group']].head(10)

# Convert DataFrames to strings without index
top_10_high_income_info = top_10_high_income.to_string(index=False)
top_10_middle_income_info = top_10_middle_income.to_string(index=False)
top_10_low_income_info = top_10_low_income.to_string(index=False)

# Print the results
print("Top 10 High Income Group:")
print(top_10_high_income_info)
print("\nTop 10 Middle Income Group:")
print(top_10_middle_income_info)
print("\nTop 10 Low Income Group:")
print(top_10_low_income_info)
```

Output:

```
index                    Country  Human Development Index (HDI)  \
0         0              Switzerland                        0.962
1         1                   Norway                        0.961
2         2                  Iceland                        0.959
3         3     Hong Kong, China (SAR)                      0.952
4         4                Australia                        0.951
..      ...                      ...                          ...
186     186                  Burundi                        0.426
187     187  Central African Republic                       0.404
188     188                    Niger                        0.400
189     189                     Chad                        0.394
190     190              South Sudan                        0.385

     Life expectancy at birth  Expected years of schooling  \
0                      83.9872                    16.500299
1                      83.2339                    18.185200
```

```
2                      82.6782                     19.163059
3                      85.4734                     17.278170
4                      84.5265                     21.054590
..                         ...                           ...
186                    61.6627                     10.722722
187                    53.8947                      8.040172
188                    61.5763                      6.957112
189                    52.5254                      8.035914
190                    54.9752                      5.542510

     Mean years of schooling  Gross national income (GNI) per capita  \
0                  13.859660                            66933.004540
1                  13.003630                            64660.106220
2                  13.767170                            55782.049810
3                  12.226210                            62606.845400
4                  12.726820                            49238.433350
..                       ...                                     ...
186                 3.129267                              731.786709
187                 4.334000                              966.058611
188                 2.116717                             1239.866936
189                 2.573774                             1364.169417
190                 5.726140                              767.787000

     GNI per capita rank minus HDI rank  HDI Rank Income Group
0                                     5         3         High
1                                     6         1         High
2                                    11         2         High
3                                     6         4         High
4                                    18         5       Middle
..                                  ...       ...          ...
186                                   4       187          Low
187                                   1       188          Low
188                                  -3       189          Low
189                                  -7       190          Low
190                                  -1       191          Low
```

Diagnostic Analysis:

Task1:

```python
# Task 1: For the top 10 countries in each group, find out the
similarities. Are they similar in HDI, life expectancy, etc
import pandas as pd

# Load the dataset from the provided CSV file path
csv_file_path = 'cleaned_data_final.csv'
data = pd.read_csv(csv_file_path)
```

```
# Assign income groups based on GNI per Capita
data['Income Group'] = ['High' if gni > 50000 else 'Middle' if gni >
20000 else 'Low' for gni in data['Gross national income (GNI) per
capita']]

# Sort data within each income group by HDI in descending order
sorted_data = data.sort_values(['Income Group', 'Human Development
Index (HDI)'], ascending=[True, False])

# Function to calculate mean, standard deviation, and range for the top
10 countries in each group
def calculate_top10_statistics(income_grouped_data):
   statistics = {}
   for group in ['High', 'Middle', 'Low']:
       income_group_data =
income_grouped_data[income_grouped_data['Income Group'] ==
group].head(10)
       stats = {}
       for column in ['Human Development Index (HDI)', 'Life expectancy
at birth',
                      'Expected years of schooling', 'Mean years of
schooling',  # Corrected column name
                      'Gross national income (GNI) per capita']:
           stats[column] = {
               'Mean': income_group_data[column].mean(),
               'Std': income_group_data[column].std(),
               'Range': income_group_data[column].max() -
income_group_data[column].min()
           }
       statistics[group] = stats
   return statistics

# Calculate statistics for the top 10 countries in each income group
top10_countries_statistics = calculate_top10_statistics(sorted_data)

# Convert the list of dictionaries to a DataFrame for well-structured
display
stats_df = pd.DataFrame(top10_countries_statistics)

# Display the DataFrame containing the statistics for each income group
stats_df
```

Output:

	High	Middle	Low
Human Development Index (HDI)	{'Mean': 0.9496, 'Std': 0.008514040691051989, ...	{'Mean': 0.9298, 'Std': 0.010982814858779222, ...	{'Mean': 0.7972, 'Std': 0.00871524845059254, '...
Life expectancy at birth	{'Mean': 82.68008, 'Std': 1.3853062267158767, ...	{'Mean': 82.76185999999998, 'Std': 1.428563647...	{'Mean': 75.7929, 'Std': 2.4197554697393135, '...
Expected years of schooling	{'Mean': 18.043290368, 'Std': 1.11519530155309...	{'Mean': 17.63841339, 'Std': 1.902217957705810...	{'Mean': 15.471450530000002, 'Std': 1.38586795...
Mean years of schooling	{'Mean': 12.860657960000001, 'Std': 0.84441065...	{'Mean': 13.002351511, 'Std': 0.48387726516806...	{'Mean': 10.412915060700001, 'Std': 1.47783254...
Gross national income (GNI) per capita	{'Mean': 64243.742289, 'Std': 11586.4625988606...	{'Mean': 44171.010834, 'Std': 3649.79401201645...	{'Mean': 15893.220514999999, 'Std': 2832.37385...

Task 2:

```
#Task 2: Where are the Outliers from, and which countries? Why is this
happening? Do they have any other similarities? (GNI rank and HDI rank
# List of variables for finding outliers
variables_to_find_outliers = [
   "Human Development Index (HDI)",
   "Life expectancy at birth",
   "Expected years of schooling",
   "Mean years of schooling",
   "Gross national income (GNI) per capita"
]

# Function to find outliers using the Interquartile Range (IQR) method
def find_outliers(data, column):
   # Calculate the first and third quartile (25th and 75th percentiles)
   Q1 = data[column].quantile(0.25)
   Q3 = data[column].quantile(0.75)
   # Interquartile range
   IQR = Q3 - Q1

   # Determine the outliers with outside of Q1 - 1.5*IQR and Q3 +
1.5*IQR
   outliers_cap = (data[column] < (Q1 - 1.5 * IQR)) | (data[column] >
(Q3 + 1.5 * IQR))
   return data[outliers_cap]

# Loop through each variable and disolay the result
for variable in variables_to_find_outliers:
   outliers = find_outliers(data, variable)
   print(f"Outliers in {variable}:")

   if not outliers.empty:
       # Display each outlier's country, GNI rank, and HDI rank
```

```
        for index, row in outliers.iterrows():
            country = row['Country']
            gni_rank = row['GNI per capita rank minus HDI rank'] +
row['HDI Rank']
            hdi_rank = row['HDI Rank']
            print(f"  Country: {country}, GNI Rank: {gni_rank}, HDI
Rank: {hdi_rank}")
    else:
        print("  No outliers found.")

    print()  # Add an empty line for better readability between
variables
```

Output:

```
Outliers in Human Development Index (HDI):
  No outliers found.

Outliers in Life expectancy at birth:
  No outliers found.

Outliers in Expected years of schooling:
  Country: South Sudan, GNI Rank: 190, HDI Rank: 191

Outliers in Mean years of schooling:
  No outliers found.

Outliers in Gross national income (GNI) per capita:
  Country: Ireland, GNI Rank: 5, HDI Rank: 8
  Country: Singapore, GNI Rank: 0, HDI Rank: 10
  Country: Liechtenstein, GNI Rank: -1, HDI Rank: 14
  Country: Luxembourg, GNI Rank: 4, HDI Rank: 17
  Country: Qatar, GNI Rank: 3, HDI Rank: 42
```

Task 3:

```
#Task 3: Correlation analysis

# List of variables for finding correlations with HDI
columns_for_analysis = [
    'Human Development Index (HDI)',
    'Life expectancy at birth',
    'Expected years of schooling',
    'Mean years of schooling',
    'Gross national income (GNI) per capita'
```

```
]

# Calculating the correlation matrix
correlation_matrix = data[columns_for_analysis].corr()

# Getting the correlation of other variables with HDI
hdi_correlation = correlation_matrix['Human Development Index (HDI)']

# Displaying the correlations of other variables with HDI
print(hdi_correlation)
```

Output:

```
Human Development Index (HDI)             1.000000
Life expectancy at birth                  0.905964
Expected years of schooling               0.895600
Mean years of schooling                   0.909126
Gross national income (GNI) per capita    0.788036
Name: Human Development Index (HDI), dtype: float64
```

Tasks 4 and 5:

```
# Task 4
# Calculate ranks for 'Life expectancy at birth' and 'Mean years of
schooling'
data['Life Expectancy Rank'] = data['Life expectancy at
birth'].rank(ascending=False)
data['Mean Schooling Rank'] = data['Mean years of
schooling'].rank(ascending=False)

# Calculate the difference between these ranks and HDI rank
data['Life Expectancy Rank - HDI Rank'] = data['Life Expectancy Rank']
- data['HDI Rank']
data['Mean Schooling Rank - HDI Rank'] = data['Mean Schooling Rank'] -
data['HDI Rank']

# Task 5:
# Define the quantile thresholds for HDI, life expectancy, and mean
years of schooling
hdi_quantiles = data['Human Development Index (HDI)'].quantile([0.25,
0.50, 0.75])
life_expectancy_quantiles = data['Life expectancy at
birth'].quantile([0.25, 0.50, 0.75])
```

```
schooling_quantiles = data['Mean years of schooling'].quantile([0.25, 0.50, 0.75])

# Map categories based on these quantiles
data['HDI Category'] = pd.cut(data['Human Development Index (HDI)'],
                              bins=[0, hdi_quantiles[0.25], hdi_quantiles[0.50], hdi_quantiles[0.75], 1],
                              labels=['Low', 'Medium', 'High', 'Very High'])
data['Life Expectancy Category'] = pd.cut(data['Life expectancy at birth'],
                                          bins=[0, life_expectancy_quantiles[0.25], life_expectancy_quantiles[0.50], life_expectancy_quantiles[0.75], data['Life expectancy at birth'].max()],
                                          labels=['Low', 'Medium', 'High', 'Very High'])
data['Mean Schooling Category'] = pd.cut(data['Mean years of schooling'],
                                         bins=[0, schooling_quantiles[0.25], schooling_quantiles[0.50], schooling_quantiles[0.75], data['Mean years of schooling'].max()],
                                         labels=['Low', 'Medium', 'High', 'Very High'])

# Creating the comparison frame
comparison_frame = data[['Country', 'Human Development Index (HDI)',
                         'Life expectancy at birth', 'Mean years of schooling',
                         'Gross national income (GNI) per capita',
                         'HDI Rank', 'Life Expectancy Rank', 'Mean Schooling Rank',
                         'Life Expectancy Rank - HDI Rank', 'Mean Schooling Rank - HDI Rank',
                         'GNI per capita rank minus HDI rank',
                         'HDI Category', 'Life Expectancy Category', 'Mean Schooling Category']]

# Display the first few rows of this comparison frame
comparison_frame.head()
```

Output:

	Country	Human Development Index (HDI)	Life expectancy at birth	Mean years of schooling	Gross national income (GNI) per capita	HDI Rank	Life Expectancy Rank	Mean Schooling Rank	Life Expectancy Rank - HDI Rank	Mean Schooling Rank - HDI Rank	GNI per capita rank minus HDI rank	HDI Category	Life Expectancy Category	Mean Schooling Category
0	Switzerland	0.962	83.9872	13.85966	66933.00454	3	4.0	2.0	1.0	-1.0	5	Very High	Very High	Very High
1	Norway	0.961	83.2339	13.00363	64660.10622	1	8.0	14.0	7.0	13.0	6	Very High	Very High	Very High
2	Iceland	0.959	82.6782	13.76717	55782.04981	2	13.0	4.0	11.0	2.0	11	Very High	Very High	Very High
3	Hong Kong, China (SAR)	0.952	85.4734	12.22621	62606.84540	4	1.0	37.0	-3.0	33.0	6	Very High	Very High	Very High
4	Australia	0.951	84.5265	12.72682	49238.43335	5	3.0	23.0	-2.0	18.0	18	Very High	Very High	Very High

Task 6:

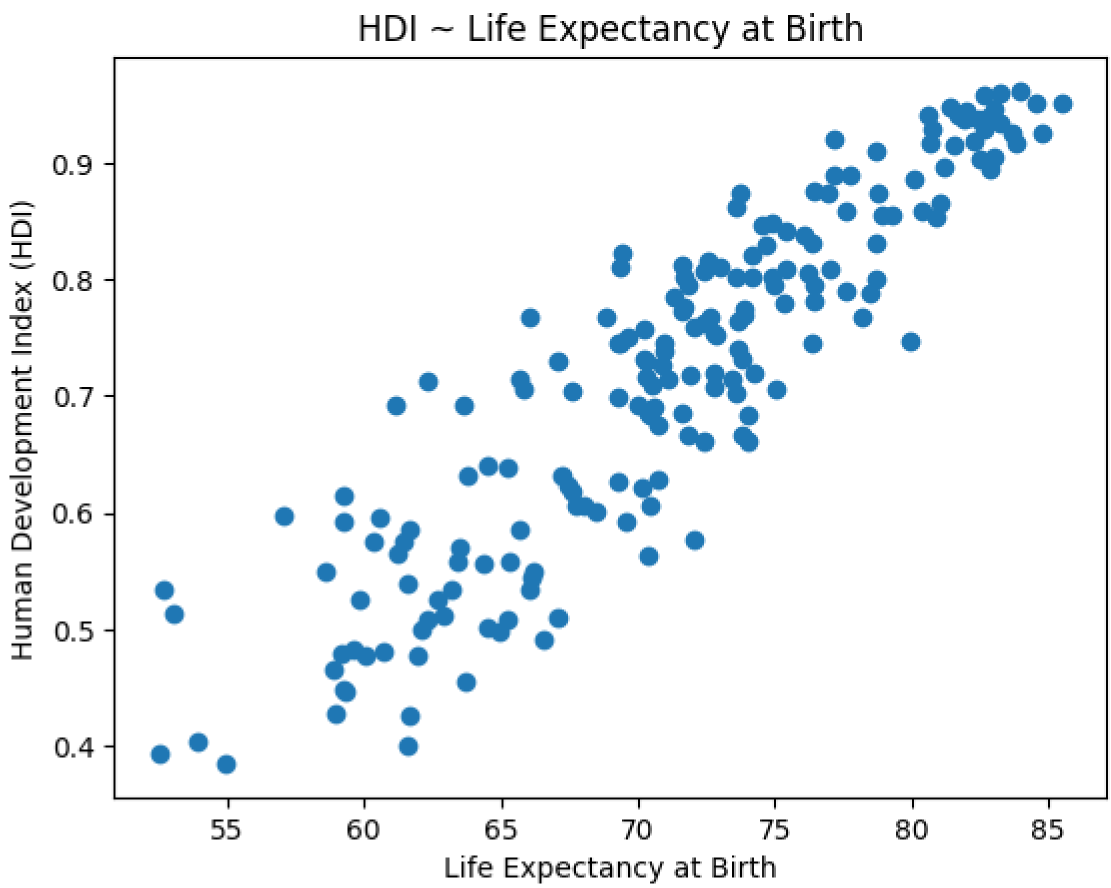
```
# Task 6
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
# The independent variables we need are 'Life expectancy at birth',
# 'Expected years of schooling', 'Mean years of schooling',
# 'Gross national income (GNI) per capita'
x1 = data[['Life expectancy at birth']].values.reshape(-1, 1)
y = data['Human Development Index (HDI)']
model1 = LinearRegression().fit(x1, y)
# Plotting for Life expectancy at birth
r_sq = model1.score(x1, y)
print('coefficient of determination:', r_sq)
# Print the Intercept:
print('intercept:', model1.intercept_)
# Print the Slope:
print('slope:', model1.coef_)
plt.scatter(x1, y)
plt.title('HDI ~ Life Expectancy at Birth')
plt.xlabel('Life Expectancy at Birth')
plt.ylabel('Human Development Index (HDI)')
plt.show()

# For Expected years of schooling
x2 = data[['Expected years of schooling']].values.reshape(-1, 1)
model2 = LinearRegression().fit(x2, y)
r_sq2 = model2.score(x2, y)
print('coefficient of determination:', r_sq2)
# Print the Intercept:
print('intercept:', model2.intercept_)
```

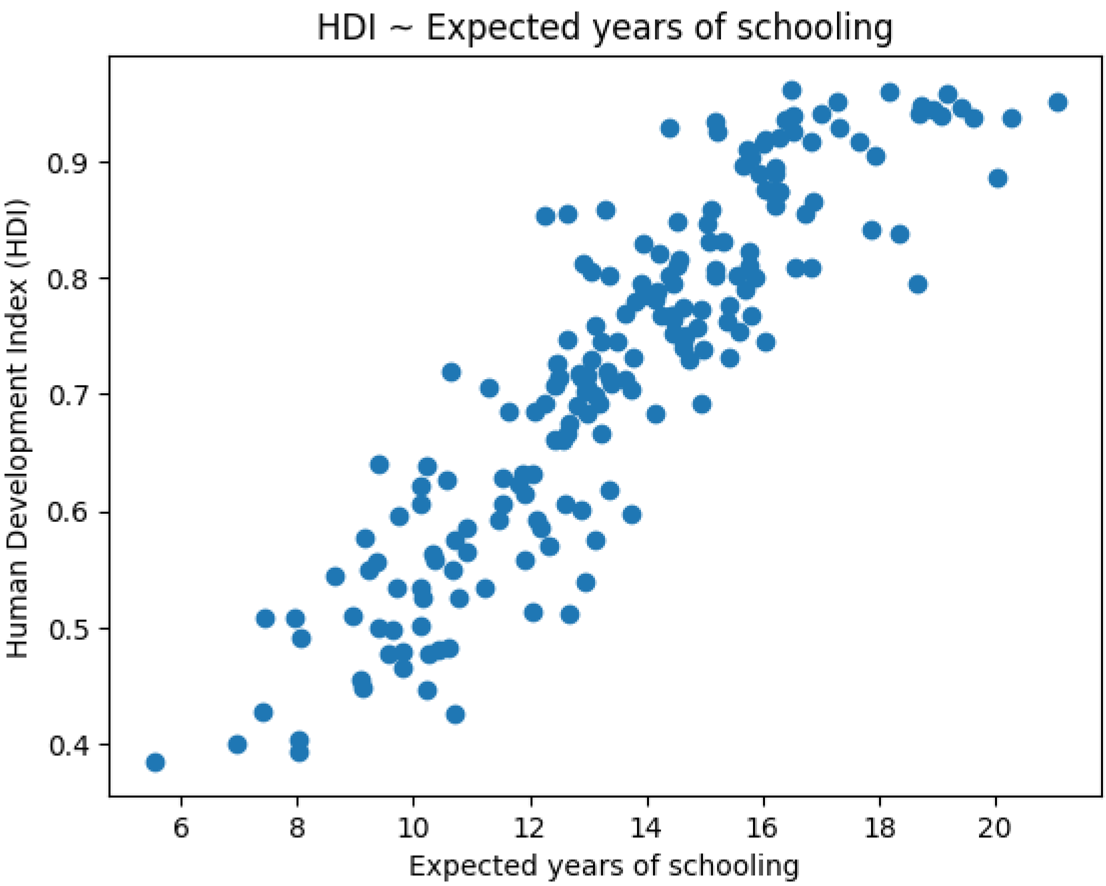
```
# Print the Slope:
print('slope:', model2.coef_)
plt.scatter(x2, y)
plt.title('HDI ~ Expected years of schooling')
plt.xlabel('Expected years of schooling')
plt.ylabel('Human Development Index (HDI)')
plt.show()


# For Mean years of schooling
x3 = data[['Mean years of schooling']].values.reshape(-1, 1)
model3 = LinearRegression().fit(x3, y)
r_sq3 = model3.score(x3, y)
print('coefficient of determination:', r_sq3)
# Print the Intercept:
print('intercept:', model3.intercept_)
# Print the Slope:
print('slope:', model3.coef_)
plt.scatter(x3, y)
plt.title('HDI ~ Mean years of schooling')
plt.xlabel('Mean years of schooling')
plt.ylabel('Human Development Index (HDI)')
plt.show()


# For Gross national income (GNI) per capita
x4 = data[['Gross national income (GNI) per
capita']].values.reshape(-1, 1)
model4 = LinearRegression().fit(x4, y)
r_sq4 = model4.score(x4, y)
print('coefficient of determination:', r_sq4)
# Print the Intercept:
print('intercept:', model4.intercept_)
# Print the Slope:
print('slope:', model4.coef_)
plt.scatter(x4, y)
plt.title('HDI ~ Gross national income (GNI) per capita')
plt.xlabel('Gross national income (GNI) per capita')
plt.ylabel('Human Development Index (HDI)')
plt.show()

# Now run a multiple linear regression
x = data[['Life expectancy at birth', 'Expected years of schooling',
```

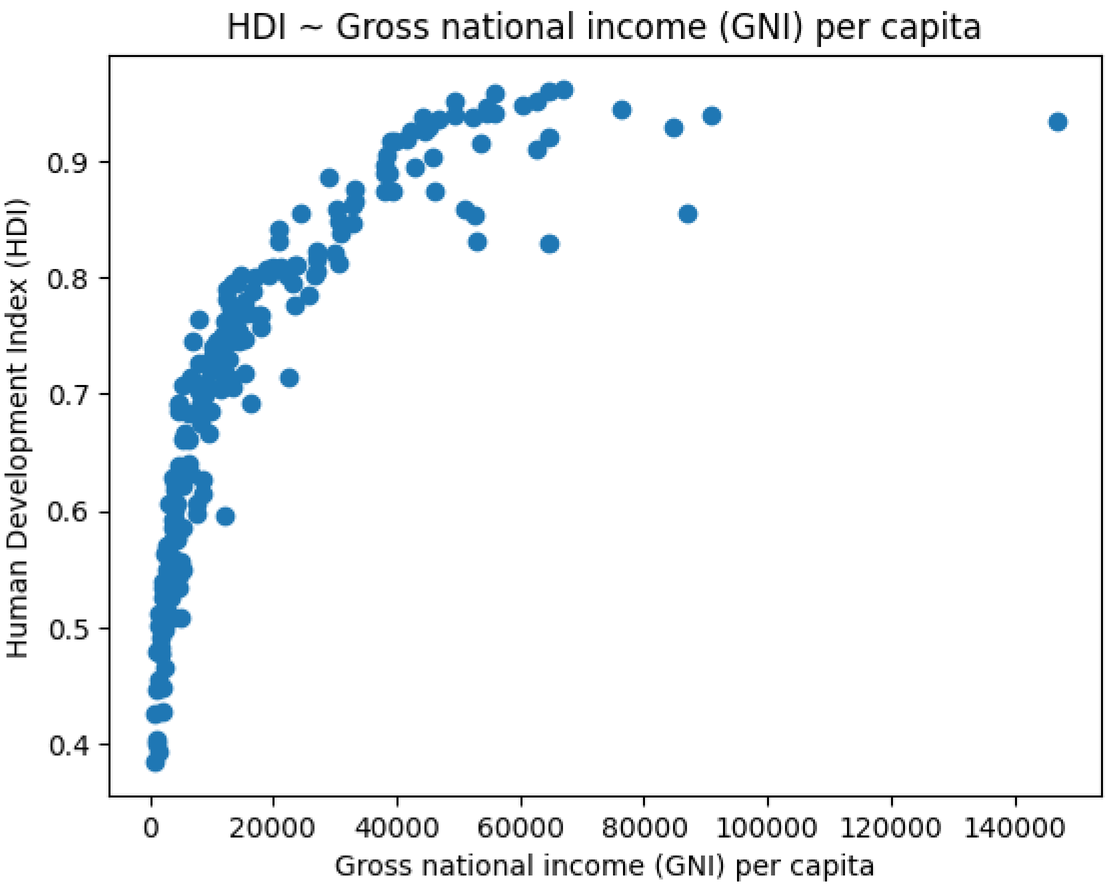
```
'Mean years of schooling', 'Gross national income (GNI) per capita']]
model5 = LinearRegression().fit(x, y)
r_sq5 = model5.score(x, y)
print('coefficient of determination:', r_sq5)
# Print the Intercept:
print('intercept:', model5.intercept_)
# Print the Slope:
print('slope:', model5.coef_)
```

Output:

```
coefficient of determination: 0.8207703718155178
intercept: -0.5524769465886521
slope: [0.01785166]
```




HDI ~ Expected years of schooling
Human Development Index (HDI)
0.4
0.5
0.6
0.7
0.8
0.9
6
8
10
12
14
16
18
20
Expected years of schooling


HDI ~ Gross national income (GNI) per capita
Human Development Index (HDI)
0.4
0.5
0.6
0.7
0.8
0.9
0
20000
40000
60000
80000
100000
120000
140000
Gross national income (GNI) per capita

```
coefficient of determination: 0.9792528605489262
intercept: -0.09672263242506951
slope: [6.37598035e-03 1.33947044e-02 1.80525532e-02 9.42245904e-07]
```

Predictive Analytics:

```
# Predictive analytics
# We are going to apply model validating to get the best predictive
model
# using train-test model
from sklearn.model_selection import train_test_split
# Now we are going it for model5, the multiple linear regression
# Split to 25% test data and 75% train data
x_train, x_test, y_train,y_test = train_test_split(x, y, test_size
=0.25, random_state=1)
x_train
model5_split = LinearRegression()
print(model5_split.fit(x_train, y_train))
# Predict the output by passing the x_test variable
y_pred = model5_split.predict(x_test)
print('The HDI values predicted by the model by passing the x_test
variable:', y_pred)
print('The R square is:', model5_split.score(x_test,y_test))

# Now we are going it for model1,
# Split to 25% test data and 75% train data
x1_train, x1_test, y_train,y_test = train_test_split(x1, y, test_size
=0.25, random_state=1)
x1_train
model1_split = LinearRegression()
print(model1_split.fit(x1_train, y_train))
# Predict the output by passing the x_test variable
y1_pred = model1_split.predict(x1_test)
print('The HDI values predicted by the model by passing the x_test
variable:', y1_pred)
print('The R square is:', model1_split.score(x1_test,y_test))

# Now we are going it for model2,
# Split to 25% test data and 75% train data
x2_train, x2_test, y_train,y_test = train_test_split(x2, y, test_size
=0.25, random_state=1)
x2_train
model2_split = LinearRegression()
print(model2_split.fit(x2_train, y_train))
# Predict the output by passing the x_test variable
```

```
y2_pred = model1_split.predict(x2_test)
print('The HDI values predicted by the model by passing the x_test
variable:', y2_pred)
print('The R square is:', model2_split.score(x2_test,y_test))

# Now we are going it for model3,
# Split to 25% test data and 75% train data
x3_train, x3_test, y_train,y_test = train_test_split(x3, y, test_size
=0.25, random_state=1)
x3_train
model3_split = LinearRegression()
print(model3_split.fit(x3_train, y_train))
# Predict the output by passing the x_test variable
y3_pred = model3_split.predict(x3_test)
print('The HDI values predicted by the model by passing the x_test
variable:', y3_pred)
print('The R square is:', model3_split.score(x3_test,y_test))

# Choosing variables except Gross national income (GNI) per capita
xecpt = data[['Life expectancy at birth', 'Expected years of
schooling',
'Mean years of schooling']]
xecpt_train, xecpt_test, y_train,y_test = train_test_split(xecpt, y,
test_size =0.25, random_state=1)
xecpt_train
modelecpt_split = LinearRegression()
print(modelecpt_split.fit(xecpt_train, y_train))
# Predict the output by passing the x_test variable
yecpt_pred = modelecpt_split.predict(xecpt_test)
print('The HDI values predicted by the model by passing the x_test
variable:', yecpt_pred)
print('The R square is:', modelecpt_split.score(xecpt_test,y_test))
```

Output:

```
LinearRegression()
The HDI values predicted by the model by passing the x_test variable:
[0.88338631 0.79943563 0.68289816 0.5949755  0.77196656 0.69097857
 0.86884048 0.68528617 0.44484693 0.71262482 0.66320722 0.93612532
 0.7710328  0.52623677 0.49840556 0.77280984 0.66497942 0.84695239
 1.00166996 0.82094631 0.81120226 0.50781985 0.65324324 0.57840496
```

```
 0.88313356 0.79497268 0.95358179 0.69842427 0.49754954 0.79290851
 0.94186343 0.66511246 0.50748477 0.92506524 0.74682767 0.74224281
 0.69735386 0.75252178 0.8032142  0.55758859 0.87222279 0.73599793
 0.66072315 0.79286423 0.73744631 0.84011153 0.93336773 0.96379144]
The R square is: 0.979873820461274

LinearRegression()
The HDI values predicted by the model by passing the x_test variable:
[0.83357339 0.81120028 0.65401578 0.70311041 0.62661901 0.70526694
 0.76272255 0.73074758 0.50728868 0.70181296 0.76858584 0.92019518
 0.79065492 0.62798894 0.54959024 0.7287466  0.70364424 0.83095904
 0.95374159 0.80359586 0.68669427 0.64557173 0.61333337 0.52609291
 0.92411052 0.74187315 0.92241888 0.5615802  0.61329625 0.74947934
 0.92068659 0.62048528 0.60813649 0.9390931  0.84210052 0.80968187
 0.786181   0.73307911 0.74005955 0.62753465 0.81110483 0.71420594
 0.71018454 0.79304654 0.76183696 0.86085349 0.88684851 0.89803949]
The R square is: 0.735785180894186

LinearRegression()
The HDI values predicted by the model by passing the x_test variable:
[-0.25368207 -0.28499914 -0.29735687 -0.357814   -0.26137771
-0.31074157
 -0.25252638 -0.31317048 -0.37878873 -0.30996462 -0.31846751
-0.28588048
 -0.29645349 -0.38733973 -0.35084891 -0.29473374 -0.32672492 -0.2732602
 -0.16821792 -0.21623375 -0.26169501 -0.38194134 -0.35939376
-0.30867502
 -0.25355641 -0.28278091 -0.24829711 -0.29921614 -0.39994592
-0.28345328
 -0.25056078 -0.31938479 -0.36979191 -0.24834273 -0.28874452
-0.28183688
 -0.34089444 -0.30853076 -0.27223726 -0.34186487 -0.25708535
-0.27558907
 -0.31640408 -0.24284782 -0.281884   -0.31700733 -0.23441403
-0.20957716]
The R square is: 0.7630180440163556


LinearRegression()
The HDI values predicted by the model by passing the x_test variable:
[0.88987713 0.81980694 0.69895713 0.57302901 0.87314904 0.71184317
 0.91771254 0.65661348 0.41380128 0.75297723 0.59852181 0.89633506
 0.78659979 0.52099093 0.45870274 0.82541685 0.67014222 0.85985327
 0.88356933 0.70234939 0.88567066 0.45023297 0.80195289 0.59248321
 0.79562124 0.83607929 0.84806888 0.82364523 0.48928027 0.83415733
 0.93260113 0.70165604 0.46190129 0.87410082 0.65661938 0.67738853
 0.70613416 0.8217479  0.85775036 0.51607548 0.90295572 0.73734611
 0.63673461 0.71781715 0.71049386 0.7626813  0.91364075 0.8939135 ]
The R square is: 0.8217693068504446

LinearRegression()
```

```
The HDI values predicted by the model by passing the x_test variable:
[0.88086349 0.81570525 0.68589801 0.60433668 0.77712267 0.7026742
 0.86263844 0.68848282 0.43436196 0.71916892 0.6761132  0.89485629
 0.78327835 0.52524139 0.49481166 0.77405687 0.67118842 0.85091009
 1.00110941 0.81949785 0.80836924 0.50750105 0.66047084 0.57578834
 0.88070316 0.79414068 0.90643755 0.69664535 0.49505129 0.79609529
 0.93949472 0.65424172 0.50610114 0.92466586 0.75722734 0.75753726
 0.71072642 0.76302549 0.81118791 0.56045676 0.87377055 0.74631926
 0.66845081 0.79948169 0.75058712 0.78688058 0.93003249 0.94706581]
The R square is: 0.9713563147679019
```