Final Project Submission

Student name: Ryan Keats Instructor name: Hardik Idnani

In [1]:

```
# Imported any necessary libraries to assist my coding
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.api as sm
import seaborn as sns
```

In [2]:

```
# Load the dataset
df = pd.read_csv('Life Expectancy Data.csv')
# Remove leading and trailing whitespaces
df.columns = df.columns.str.strip()
# Tidying number displays
pd.options.display.float_format = '{:,.2f}'.format
# Display first 5 rows
df.head()
```

Out[2]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
0	Afghanistan	2015	Developing	65.00	263.00	62	0.01	71.28	65.00
1	Afghanistan	2014	Developing	59.90	271.00	64	0.01	73.52	62.00
2	Afghanistan	2013	Developing	59.90	268.00	66	0.01	73.22	64.00
3	Afghanistan	2012	Developing	59.50	272.00	69	0.01	78.18	67.00
4	Afghanistan	2011	Developing	59.20	275.00	71	0.01	7.10	68.00

5 rows × 22 columns

In [3]:

```
# Sorted the 'Life Expectancy' column from highest to lowest
df_sort = df.sort_values('Life expectancy', ascending=False)
# Displaying .head() function
df_sort.head()
```

Out[3]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
2433	Spain	2007	Developed	89.00	72.00	2	11.05	510.93	96.0(
1916	Norway	2009	Developed	89.00	67.00	0	6.68	142.37	NaN
938	France	2007	Developing	89.00	89.00	3	12.20	64.74	42.0(
915	Finland	2014	Developing	89.00	78.00	0	8.80	6,164.46	NaN
2513	Sweden	2007	Developed	89.00	63.00	0	6.90	7,593.39	NaN

5 rows × 22 columns

In [4]:

```
# Returns shape of dataset
df.shape
```

Out[4]:

```
(2938, 22)
```

In [5]:

```
# Returns columns of the dataset
df.columns
```

Out[5]:

```
Index(['Country', 'Year', 'Status', 'Life expectancy', 'Adult Mortali
ty',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatit
is B',
       'Measles', 'BMI', 'under-five deaths', 'Polio', 'Total expendi
ture',
       'Diphtheria', 'HIV/AIDS', 'GDP', 'Population', 'thinness  1-19
years',
       'thinness 5-9 years', 'Income composition of resources', 'Scho
oling'],
      dtype='object')
```

In [6]:

```
# Retruns the data types of each column
df.dtypes
```

Out[6]:

```
Country                           object
Year                               int64
Status                            object
Life expectancy                  float64
Adult Mortality                  float64
infant deaths                      int64
Alcohol                          float64
percentage expenditure           float64
Hepatitis B                      float64
Measles                            int64
BMI                              float64
under-five deaths                  int64
Polio                            float64
Total expenditure                float64
Diphtheria                       float64
HIV/AIDS                         float64
GDP                              float64
Population                       float64
thinness  1-19 years             float64
thinness 5-9 years               float64
Income composition of resources  float64
Schooling                        float64
dtype: object
```

In [7]:

```
# Returns the count of each unique values for each column from the dataset
df.nunique()
```

Out[7]:

```
Country                           193
Year                               16
Status                              2
Life expectancy                   362
Adult Mortality                   425
infant deaths                     209
Alcohol                          1076
percentage expenditure           2328
Hepatitis B                        87
Measles                           958
BMI                               608
under-five deaths                 252
Polio                              73
Total expenditure                 818
Diphtheria                         81
HIV/AIDS                          200
GDP                              2490
Population                       2278
thinness  1-19 years              200
thinness 5-9 years                207
Income composition of resources   625
Schooling                         173
dtype: int64
```

In [8]:

```
# Returns comprehensive information of the dataset
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
```

In [9]:

```
# Checking for any duplicated data
duplicates = df.duplicated()
if duplicates.any():
    print("There are duplicated values in the DataFrame.")
else:
    print("There are no duplicated values in the DataFrame.")
print(duplicates.value_counts())
# Count the number of duplicate rows
duplicate_count = df.duplicated().sum()
# Display the count
print(f"Number of duplicate rows: {duplicate_count}")
```

```
There are no duplicated values in the DataFrame.
False    2938
dtype: int64
Number of duplicate rows: 0
```

In [10]:

```
# Group the data by 'Country' & Status' and calculate the mean life expectancy for
grouped_df = df.groupby(['Country', 'Status'])['Life expectancy'].mean().reset_ind
# Sort the grouped DataFrame by mean life expectancy in descending order
grouped_df_sorted = grouped_df.sort_values(by='Life expectancy', ascending=False)
# Get the number of unique countries in the grouped DataFrame
num_countries = grouped_df_sorted['Country'].nunique()
# Print the sorted grouped data and the number of unique countries
print("Number of unique countries:", num_countries)
grouped_df_sorted.head(20)
```

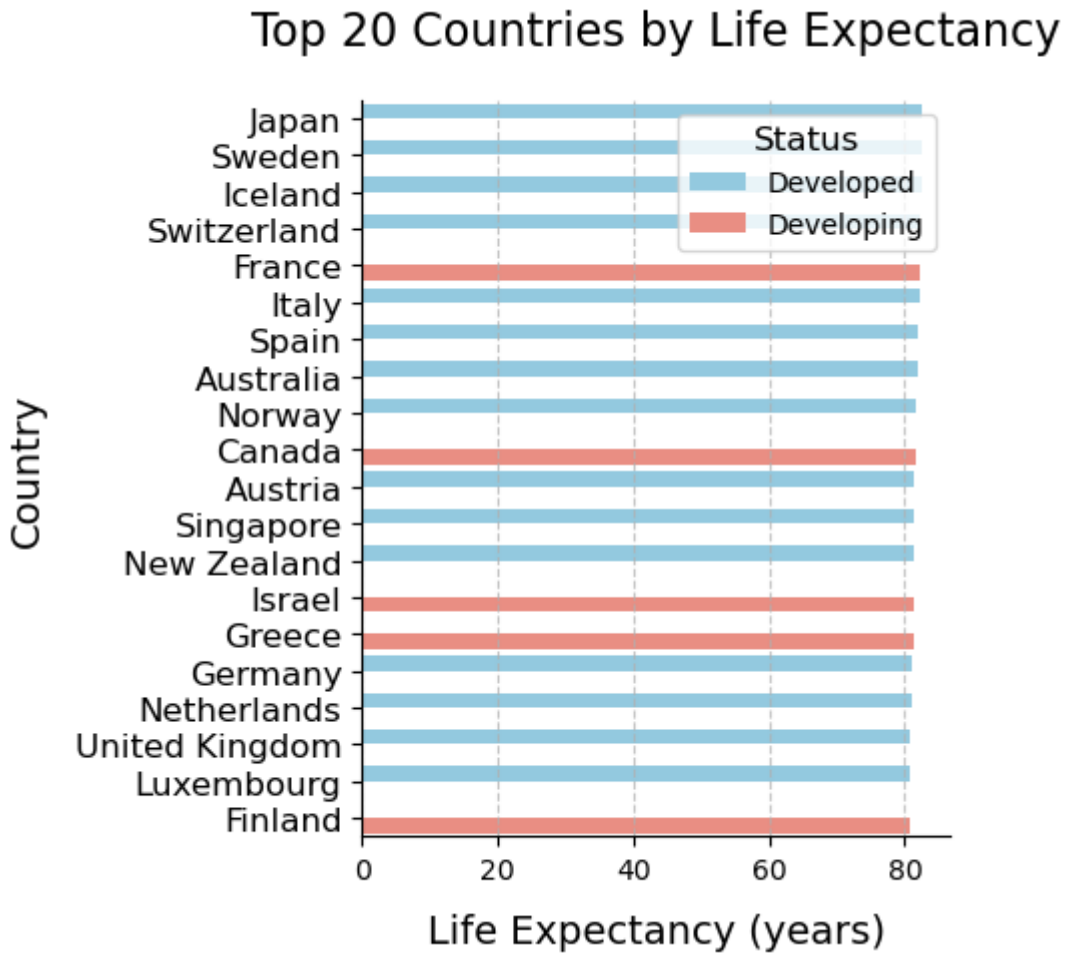
```
Number of unique countries: 193
```

Out[10]:

	Country	Status	Life expectancy
84	Japan	Developed	82.54
165	Sweden	Developed	82.52
75	Iceland	Developed	82.44
166	Switzerland	Developed	82.33
60	France	Developing	82.22
82	Italy	Developed	82.19
160	Spain	Developed	82.07
7	Australia	Developed	81.81
125	Norway	Developed	81.79
30	Canada	Developing	81.69
8	Austria	Developed	81.48
153	Singapore	Developed	81.47
120	New Zealand	Developed	81.34
81	Israel	Developing	81.30
66	Greece	Developing	81.22
64	Germany	Developed	81.17
119	Netherlands	Developed	81.13
182	United Kingdom of Great Britain and Northern I...	Developed	80.79
98	Luxembourg	Developed	80.78
59	Finland	Developing	80.71

In [1]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
# Sample data
data = {
    'Country': ['Japan', 'Sweden', 'Iceland', 'Switzerland', 'France', 'Italy', 'S
                'Austria', 'Singapore', 'New Zealand', 'Israel', 'Greece', 'German
    'Status': ['Developed', 'Developed', 'Developed', 'Developed', 'Developing', '
               'Developed', 'Developed', 'Developed', 'Developing', 'Developing',
    'Life expectancy': [82.54, 82.52, 82.44, 82.33, 82.22, 82.19, 82.07, 81.81, 81
grouped_df_sorted = pd.DataFrame(data)
# Set a custom color palette
custom_palette = {'Developed': 'skyblue', 'Developing': 'salmon'}
# Create a bar plot using Seaborn
plt.figure(figsize=(5, 5))
sns.barplot(x='Life expectancy', y='Country', hue='Status', data=grouped_df_sorted
# Customize the plot
plt.xlabel('Life Expectancy (years)', fontsize=14, labelpad=10)
plt.ylabel('Country', fontsize=14, labelpad=10)
plt.title('Top 20 Countries by Life Expectancy', fontsize=16, pad=20)
plt.legend(title='Status', title_fontsize=12, fontsize=10, loc='upper right')
# Remove the right and top spines
sns.despine(right=True, top=True)
# Increase the font size of country labels
plt.yticks(fontsize=12)
# Add grid lines
plt.grid(axis='x', linestyle='--', alpha=0.7)
# Show the plot
plt.tight_layout()
plt.show()
```



In [12]:

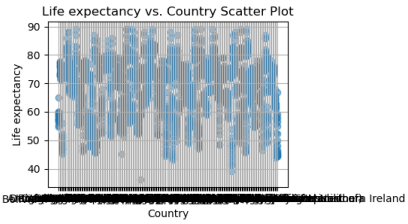
```
# Returns a descriptive analysis of the original dataset
df.describe()
```

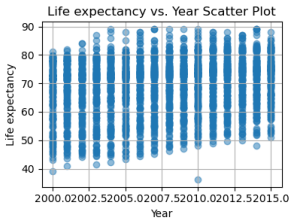
Out[12]:

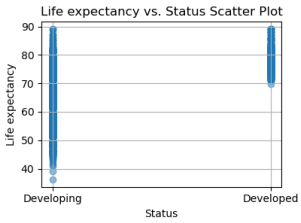
	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	
count	2,938.00	2,928.00	2,928.00	2,938.00	2,744.00	2,938.00	2,385.00	2,938.00	2
mean	2,007.52	69.22	164.80	30.30	4.60	738.25	80.94	2,419.59	
std	4.61	9.52	124.29	117.93	4.05	1,987.91	25.07	11,467.27	
min	2,000.00	36.30	1.00	0.00	0.01	0.00	1.00	0.00	
25%	2,004.00	63.10	74.00	0.00	0.88	4.69	77.00	0.00	
50%	2,008.00	72.10	144.00	3.00	3.75	64.91	92.00	17.00	
75%	2,012.00	75.70	228.00	22.00	7.70	441.53	97.00	360.25	
max	2,015.00	89.00	723.00	1,800.00	17.87	19,479.91	99.00	212,183.00	

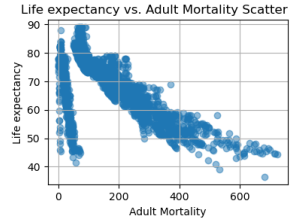
In [13]:

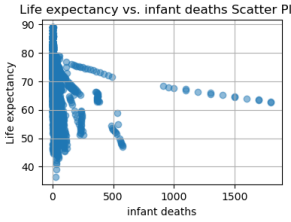
```
# Scatter plots for original dataset before cleaning

# Define my independent variables
dependent_var = 'Life expectancy'
independent_vars = ['Country', 'Year', 'Status', 'Adult Mortality','infant deaths'
                    'Alcohol', 'percentage expenditure', 'Hepatitis B','Measles',
                    'under-five deaths', 'Polio', 'Total expenditure','Diphtheria'
                    'HIV/AIDS', 'GDP', 'Population', 'thinness  1-19 years',
                    'thinness 5-9 years', 'Income composition of resources', 'Scho
# Calculate number of rows and columns for the subplot grid
num_cols = 3
num_rows = (len(independent_vars) + num_cols - 1) // num_cols
# Create the subplot grid
fig, axs = plt.subplots(num_rows, num_cols, figsize=(15, 3 * num_rows))
for i, col in enumerate(independent_vars):
    row_idx = i // num_cols
    col_idx = i % num_cols
    axs[row_idx, col_idx].scatter(df[col], df[dependent_var], alpha=0.5)
    axs[row_idx, col_idx].set_xlabel(col)
    axs[row_idx, col_idx].set_ylabel(dependent_var)
    axs[row_idx, col_idx].set_title(f'{dependent_var} vs. {col} Scatter Plot')
    axs[row_idx, col_idx].grid(True)
# Hide empty subplots
for i in range(len(independent_vars), num_cols * num_rows):
    row_idx = i // num_cols
    col_idx = i % num_cols
    fig.delaxes(axs[row_idx, col_idx])
# Adjust spacing of subplots
plt.tight_layout()
# Display the plots
plt.show()
```

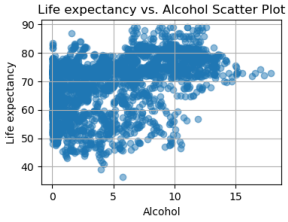

Life expectancy vs. Country Scatter Plot
Life expectancy
Country
Ireland

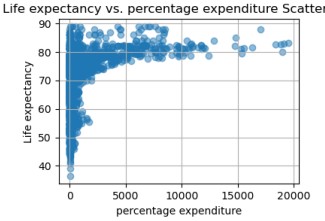

Life expectancy vs. Year Scatter Plot
Life expectancy
Year

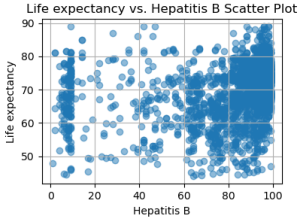

Life expectancy vs. Status Scatter Plot
Life expectancy
Developing
Developed
Status

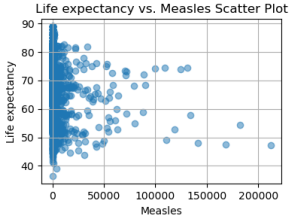

Life expectancy vs. Adult Mortality Scatter Plot
Life expectancy
Adult Mortality

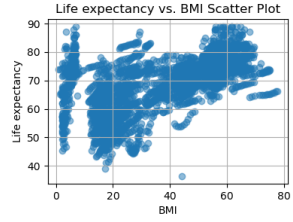

Life expectancy vs. infant deaths Scatter Plot
Life expectancy
infant deaths

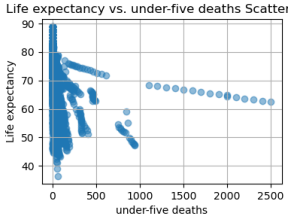

Life expectancy vs. Alcohol Scatter Plot
Life expectancy
Alcohol

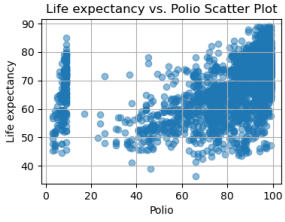

Life expectancy vs. percentage expenditure Scatter Plot
Life expectancy
percentage expenditure

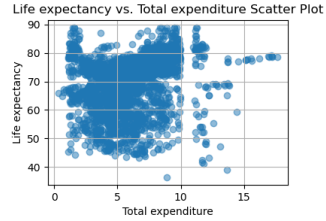

Life expectancy vs. Hepatitis B Scatter Plot
Life expectancy
Hepatitis B

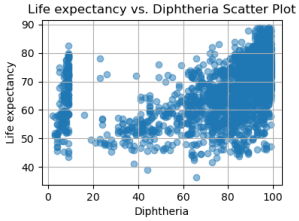

Life expectancy vs. Measles Scatter Plot
Life expectancy
Measles

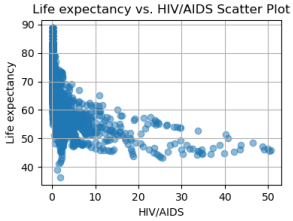

Life expectancy vs. BMI Scatter Plot
Life expectancy
BMI

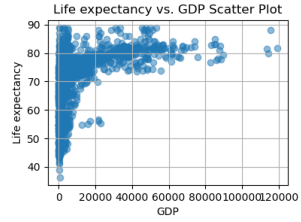

Life expectancy vs. under-five deaths Scatter Plot
Life expectancy
under-five deaths

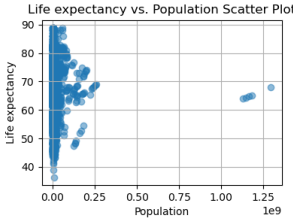

Life expectancy vs. Polio Scatter Plot
Life expectancy
Polio

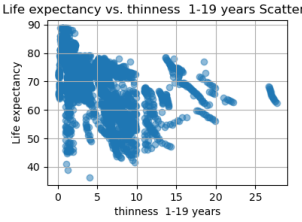

Life expectancy vs. Total expenditure Scatter Plot
Life expectancy
Total expenditure

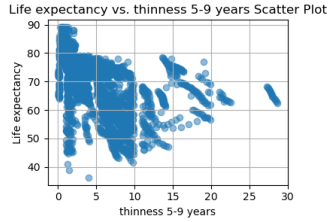

Life expectancy vs. Diphtheria Scatter Plot
Life expectancy
Diphtheria

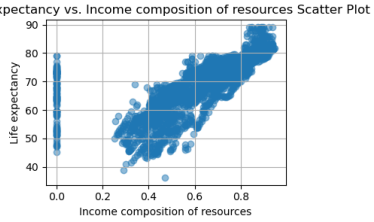

Life expectancy vs. HIV/AIDS Scatter Plot
Life expectancy
HIV/AIDS

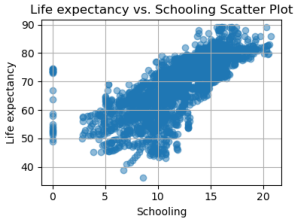

Life expectancy vs. GDP Scatter Plot
Life expectancy
GDP


Life expectancy vs. Population Scatter Plot
Life expectancy
Population
1e9


Life expectancy vs. thinness 1-19 years Scatter Plot
Life expectancy
thinness 1-19 years

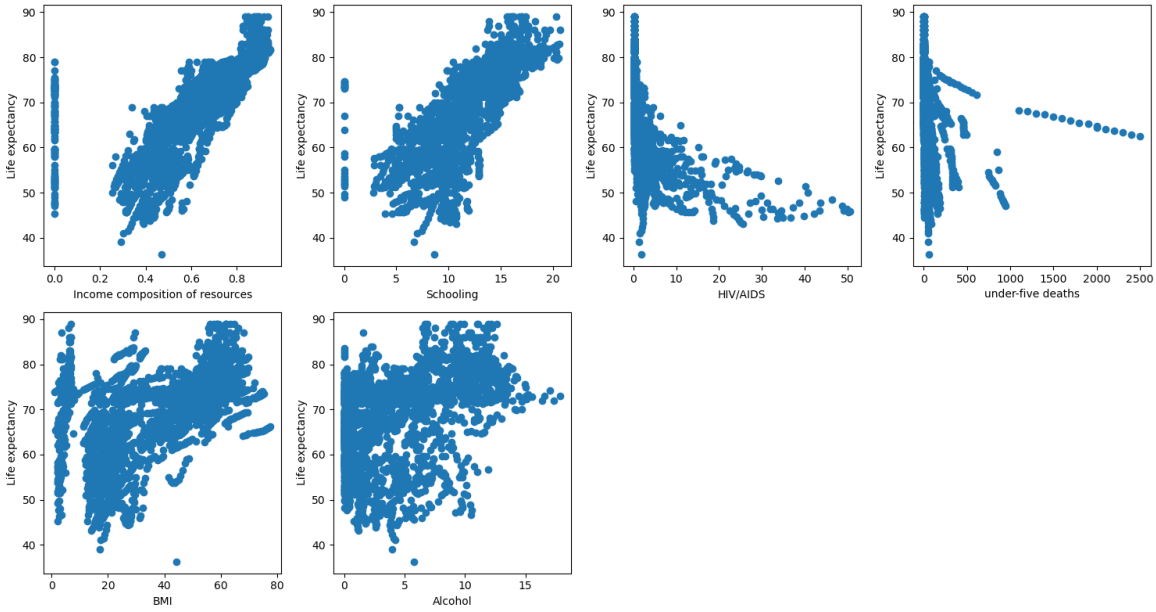

Life expectancy vs. thinness 5-9 years Scatter Plot
Life expectancy
thinness 5-9 years


Life expectancy vs. Income composition of resources Scatter Plot
Life expectancy
Income composition of resources


Life expectancy vs. Schooling Scatter Plot
Life expectancy
Schooling

In [14]:

```
# Further scatter plots for a selection of variables from the dataset

# Build scatter plots for each independent variable
independent_vars = ['Income composition of resources', 'Schooling',
                    'HIV/AIDS','under-five deaths', 'BMI', 'Alcohol']
dependent_var = 'Life expectancy'
# Set the size of the overall figure
plt.figure(figsize=(15, 8))
for i, var in enumerate(independent_vars, 1):
    plt.subplot(2, 4, i)
    plt.scatter(df[var], df[dependent_var])
    plt.xlabel(var)
    plt.ylabel(dependent_var)
# Adjust the spacing between subplots
plt.tight_layout()
# Display the plots
plt.show()
```



Recording my observations on linearity here:

These are just a few of the predictors I have chosen to illustrate. Income composition of resources and Schooling show the strongest positive Linear relationship to my target variable.
HIV/AIDS & under five deaths show a negative Linear relationship. Whilst not being as strong as the postive predictors.

Alcohol & BMI are showing a slight resemblance with a positive Linear relationship, whilst displaying large variances.

In [15]:

```
# Returns the presence of missing values (NaN) as a boolean dataframe
# Displays first 5 rows
df.isna().head()
```

Out[15]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Mea
0	False	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	F

5 rows × 22 columns

In [16]:

```
# Returns a count of the missing values (NaN) in each column of the dataset
df.isna().sum()
```

Out[16]:

```
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
```

In [17]:

```
# Remove missing values
df_clean = pd.read_csv('Life Expectancy Data.csv').dropna()
# Remove leading and trailing whitespaces
df_clean.columns = df.columns.str.strip()
```

In [18]:

```
# Drop specific columns from df dataset
columns_to_drop = ['Country', 'Status']
df_clean.drop(columns=columns_to_drop, inplace=True)
```

In [19]:

```
# Returns shape of cleaned dataset and rremoval of non numerical dtypes
df_clean.shape
```

Out[19]:

```
(1649, 20)
```

In [20]:

```
# Define my dependent variable
y = df_clean['Life expectancy']
# Define my independent variables
X = df_clean[['Year', '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'
# Add a constant term to the independent variables matrix
X = sm.add_constant(X)
# Create the OLS model
model = sm.OLS(y, X)
# Fit the model to the data
results = model.fit()
# Display the results summary
results.summary()
```

Out[20]:

OLS Regression Results

Dep. Variable:	Life expectancy	**R-squared:**	0.838
Model:	OLS	**Adj. R-squared:**	0.836
Method:	Least Squares	**F-statistic:**	443.1
Date:	Thu, 27 Jul 2023	**Prob (F-statistic):**	0.00
Time:	18:15:55	**Log-Likelihood:**	-4424.8
No. Observations:	1649	**AIC:**	8890.
Df Residuals:	1629	**BIC:**	8998.
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	313.3528	46.266	6.773	0.000	222.606	404.100
Year	-0.1299	0.023	-5.622	0.000	-0.175	-0.085
Adult Mortality	-0.0164	0.001	-17.449	0.000	-0.018	-0.015
infant deaths	0.0888	0.011	8.368	0.000	0.068	0.110
Alcohol	-0.0983	0.031	-3.139	0.002	-0.160	-0.037
percentage expenditure	0.0003	0.000	1.734	0.083	-4.08e-05	0.001
Hepatitis B	-0.0023	0.004	-0.524	0.600	-0.011	0.006
Measles	-1.107e-05	1.07e-05	-1.033	0.302	-3.21e-05	9.95e-06
BMI	0.0316	0.006	5.290	0.000	0.020	0.043
under-five deaths	-0.0666	0.008	-8.671	0.000	-0.082	-0.052
Polio	0.0057	0.005	1.104	0.270	-0.004	0.016
Total expenditure	0.0961	0.040	2.375	0.018	0.017	0.175
Diphtheria	0.0135	0.006	2.301	0.022	0.002	0.025
HIV/AIDS	-0.4495	0.018	-25.222	0.000	-0.484	-0.415
GDP	2.95e-05	2.83e-05	1.044	0.297	-2.59e-05	8.49e-05
Population	-6.527e-10	1.74e-09	-0.376	0.707	-4.06e-09	2.75e-09
thinness 1-19 years	-0.0023	0.053	-0.043	0.965	-0.105	0.101
thinness 5-9 years	-0.0531	0.052	-1.023	0.307	-0.155	0.049
Income composition of resources	10.4701	0.834	12.551	0.000	8.834	12.106
Schooling	0.9063	0.059	15.348	0.000	0.790	1.022

Omnibus:	34.044	**Durbin-Watson:**	0.715
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	65.019
Skew:	-0.098	**Prob(JB):**	7.61e-15
Kurtosis:	3.953	**Cond. No.**	3.79e+10

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.79e+10. This might indicate that there are
strong multicollinearity or other numerical problems.

In [21]:

```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X, y)
```

Out[21]:

```
LinearRegression()
```

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

In [22]:

```
# Coefficients
linreg.coef_
```

Out[22]:

```
array([ 0.00000000e+00, -1.29874789e-01, -1.64364041e-02,  8.88086801
e-02,
       -9.83150370e-02,  3.10650869e-04, -2.32723002e-03, -1.10707305
e-05,
        3.15528061e-02, -6.66491382e-02,  5.66159110e-03,  9.61044099
e-02,
        1.35422582e-02, -4.49509516e-01,  2.95015608e-05, -6.52688026
e-10,
       -2.28139560e-03, -5.31044786e-02,  1.04700581e+01,  9.06296651
e-01])
```

In [23]:

```
# Intercept
linreg.intercept_
```

Out[23]:

```
313.35279423771493
```

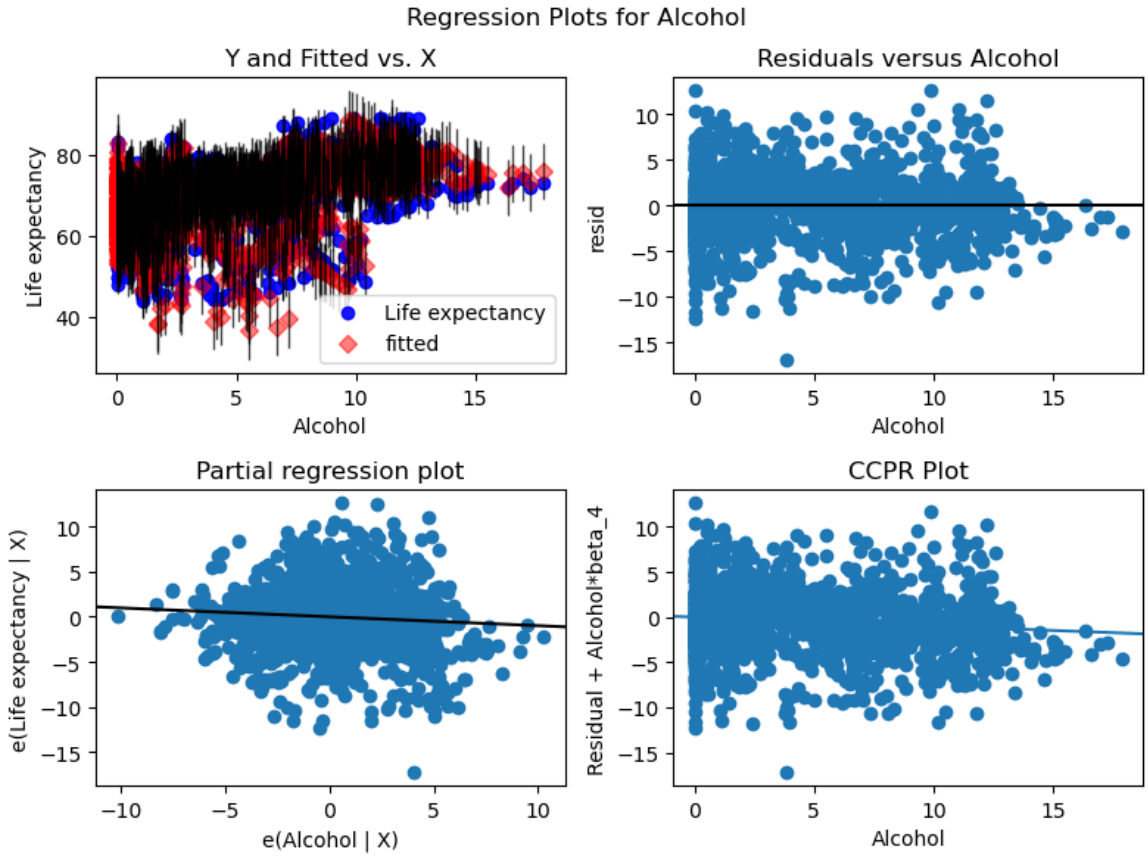
My documentation of the coefficients on my baseline model are as follows:

Life expectancy decreases by about 0.13 years per year. Higher adult mortality reduces life expectancy by 0.016 years. Each additional infant death lowers life expectancy by 0.089 years. Alcohol consumption decreases life expectancy by 0.098 years per unit increase. Healthcare expenditure shows a positive but inconclusive association with life expectancy. Hepatitis B and Measles do not significantly affect life expectancy. Higher BMI slightly raises life expectancy by 0.032 years per unit increase. Deaths in children under 5 years decrease life expectancy by 0.067 years per additional death. Polio and GDP do not significantly impact life expectancy. Higher total healthcare expenditure raises life expectancy by 0.096 years per unit increase. Diphtheria is associated with a slightly higher life expectancy, increasing it by 0.014 years. Each increase in HIV/AIDS prevalence reduces life expectancy by 0.45 years. Population size and thinness in

age groups 1-19 and 5-9 do not significantly impact life expectancy. Higher income composition of resources raises life expectancy by approximately 10.47 years per increase. Each increase in schooling

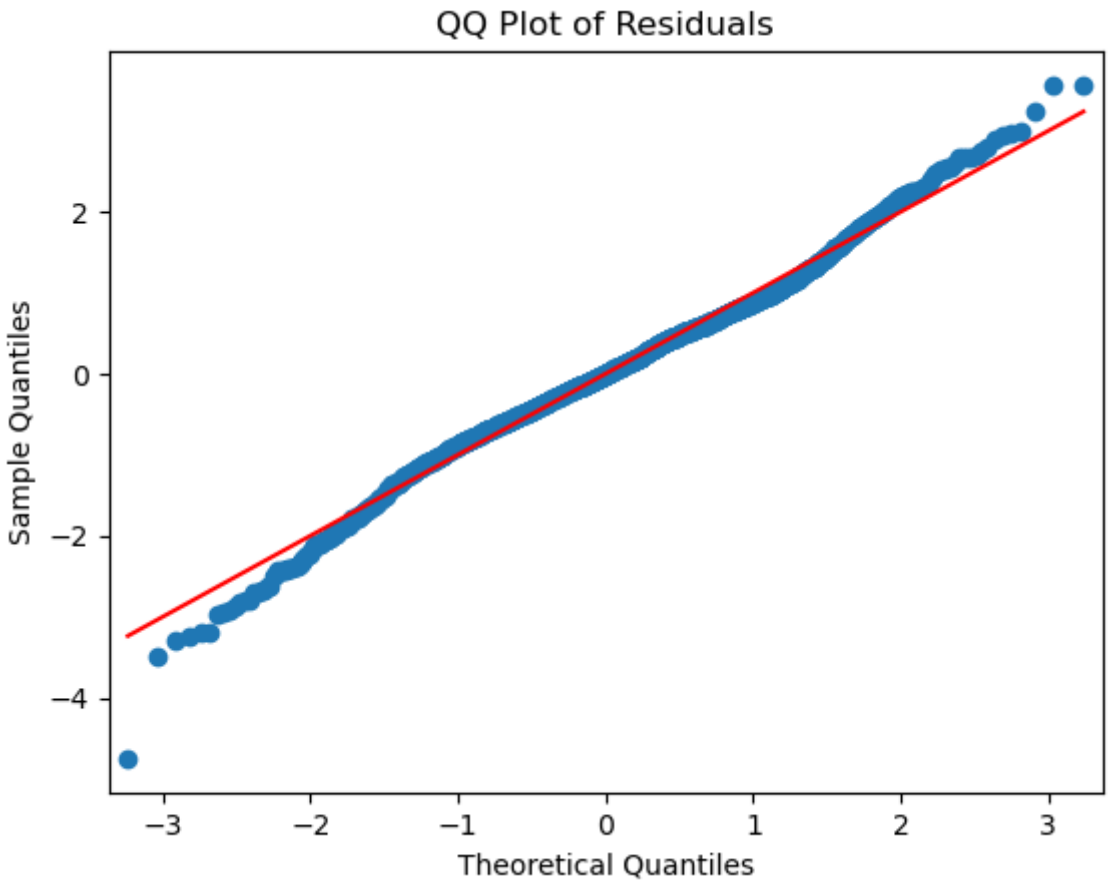
In [24]:

```
# Create the figure
fig = plt.figure(figsize=(8, 6))
# Plot the regression results for the 'Alcohol' variable
sm.graphics.plot_regress_exog(results, 'Alcohol', fig=fig)
# Display the plot
plt.show()
```



In [25]:

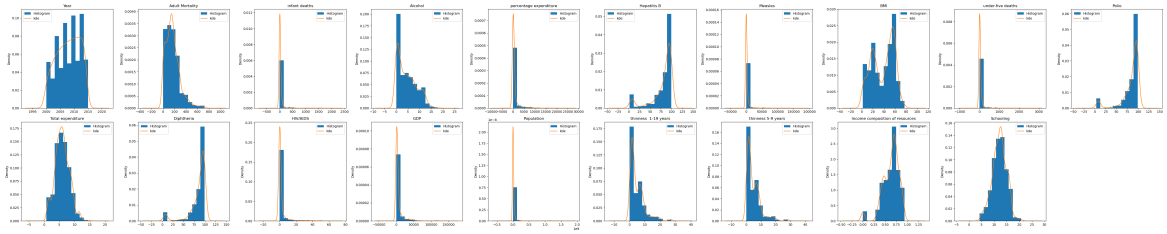
```
# Fit the linear regression model
model = sm.OLS(y, sm.add_constant(X))
results = model.fit()
# Create the QQ plot for residuals
sm.qqplot(results.resid, line='s', fit=True)
# Set plot title and labels
plt.title("QQ Plot of Residuals")
plt.xlabel("Theoretical Quantiles")
plt.ylabel("Sample Quantiles")
# Display the plot
plt.show()
```



Based on my observations of the QQ plot, I noticed that in a normally distributed data, there appears to be fewer data points in the extreme highest and lowest quantiles. As shown in the above QQ plot it shows that it is not perfectly normal, but it does come really close to a normal distribution without deviating significantly.

In [26]:

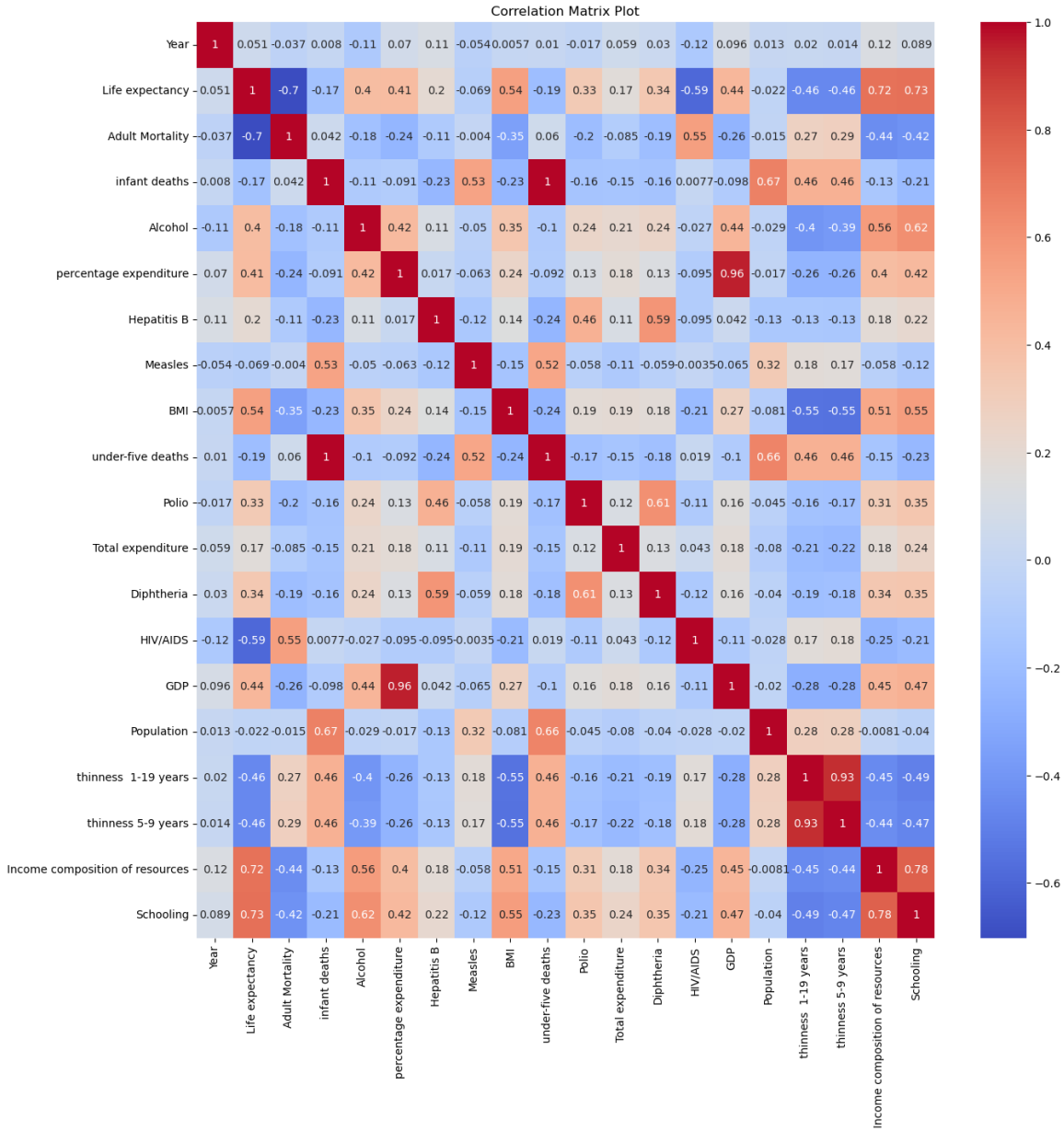
```
# Define the independent variables
independent_variables = ['Year', 'Adult Mortality', 'infant deaths',
                         'Alcohol', 'percentage expenditure', 'Hepatitis B', 'Meas
                         'under-five deaths', 'Polio', 'Total expenditure', 'Dipht
                         'HIV/AIDS', 'GDP', 'Population', 'thinness  1-19 years',
                         'thinness 5-9 years', 'Income composition of resources',
# Create subplots for each independent variable
num_rows = 2
num_cols = 10
fig, axs = plt.subplots(num_rows, num_cols, figsize=(5 * num_cols, 5 * num_rows))
# Iterate over each independent variable
for i, var in enumerate(independent_variables):
    # Calculate row and column index for each subplot
    row_idx = i // num_cols
    col_idx = i % num_cols
     # Plot histogram with kde
    ax = axs[row_idx, col_idx]
    df_clean[var].plot.hist(density=True, ax=ax, label='Histogram')
    df_clean[var].plot.kde(ax=ax, label='kde')
    ax.legend()
    ax.set_title(var)
# Hide empty subplots
for i in range(len(independent_variables), num_cols * num_rows):
    row_idx = i // num_cols
    col_idx = i % num_cols
    fig.delaxes(axs[row_idx, col_idx])
# Adjust spacing of subplots
plt.tight_layout()
# Display the plots
plt.show()
```



Recording my observations here: Most of the variables do not exhibit any resemblance of a normal distribution, and most of them display various forms of skewness. In saying that, the two predictors that are showing the closest to normal are Total expenditure and Schooling.

In [27]:

```
# Compute the correlation matrix
correlation_matrix = df_clean.corr()
# Create a correlation matrix plot using a heatmap
plt.figure(figsize=(15, 15))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Plot')
plt.show()
```



Continuing my observations:

Despite the absence of normal distributions, the correlation matrix reveals that certain predictors have a more significant impact on life expectancy than others.

In [28]:

```
# Removal of outliers using the Z-score method
# Define the threshold for Z-score
z_threshold = 3
# Identify numeric columns in the dataset
numeric_columns = df_clean.select_dtypes(include=[np.number]).columns
# Create a new DataFrame df_removed to store the dataset with outliers removed
df_removed = df_clean.copy()
# Remove outliers using the Z-score method for each numeric column
for column in numeric_columns:
    z_scores = np.abs((df_removed[column] - df_removed[column].mean()) / df_remove
    df_removed = df_removed[z_scores <= z_threshold]
# Display the dataset with outliers removed
df_removed.head()
```

Out[28]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	unde fi deat
14	2001	55.30	316.00	88	0.01	10.57	63.00	8762	12.60	1
16	2015	77.80	74.00	0	4.60	364.98	99.00	0	58.00	
17	2014	77.50	8.00	0	4.51	428.75	98.00	0	57.20	
18	2013	77.20	84.00	0	4.76	430.88	99.00	0	56.50	
19	2012	76.90	86.00	0	5.14	412.44	99.00	9	55.80	

In [29]:

```
# Returns updated shape of dataset after removing outliers
df_removed.shape
```

Out[29]:

```
(1162, 20)
```

In [30]:

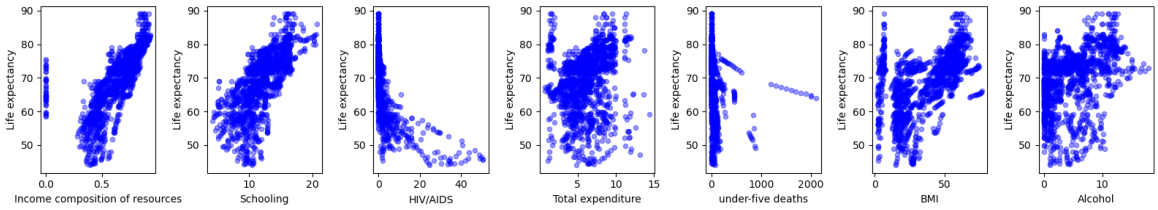
```
# Returns a descriptive analysis of the updated dataset
df_removed.describe()
```

Out[30]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	
count	1,162.00	1,162.00	1,162.00	1,162.00	1,162.00	1,162.00	1,162.00	1,162.00	1,
mean	2,008.04	70.12	156.41	11.76	4.61	394.67	84.42	694.10	
std	4.11	7.54	102.85	19.93	3.97	610.90	20.08	2,324.71	
min	2,000.00	45.30	1.00	0.00	0.01	0.10	6.00	0.00	
25%	2,005.00	65.90	81.00	0.00	0.87	40.93	81.00	0.00	
50%	2,008.00	72.20	146.00	3.00	4.00	161.37	93.00	4.00	
75%	2,012.00	74.90	217.75	14.00	7.32	479.46	96.00	141.00	
max	2,015.00	89.00	527.00	159.00	16.58	4,057.64	99.00	22,004.00	

In [31]:

```
%matplotlib inline
fig, axes = plt.subplots(nrows=1, ncols=7, figsize=(16,3))
for xcol, ax in zip(['Income composition of resources', 'Schooling', 'HIV/AIDS','T
    df_clean.plot(kind='scatter', x=xcol, y='Life expectancy', ax=ax, alpha=0.4, c
plt.tight_layout()
```



In [32]:

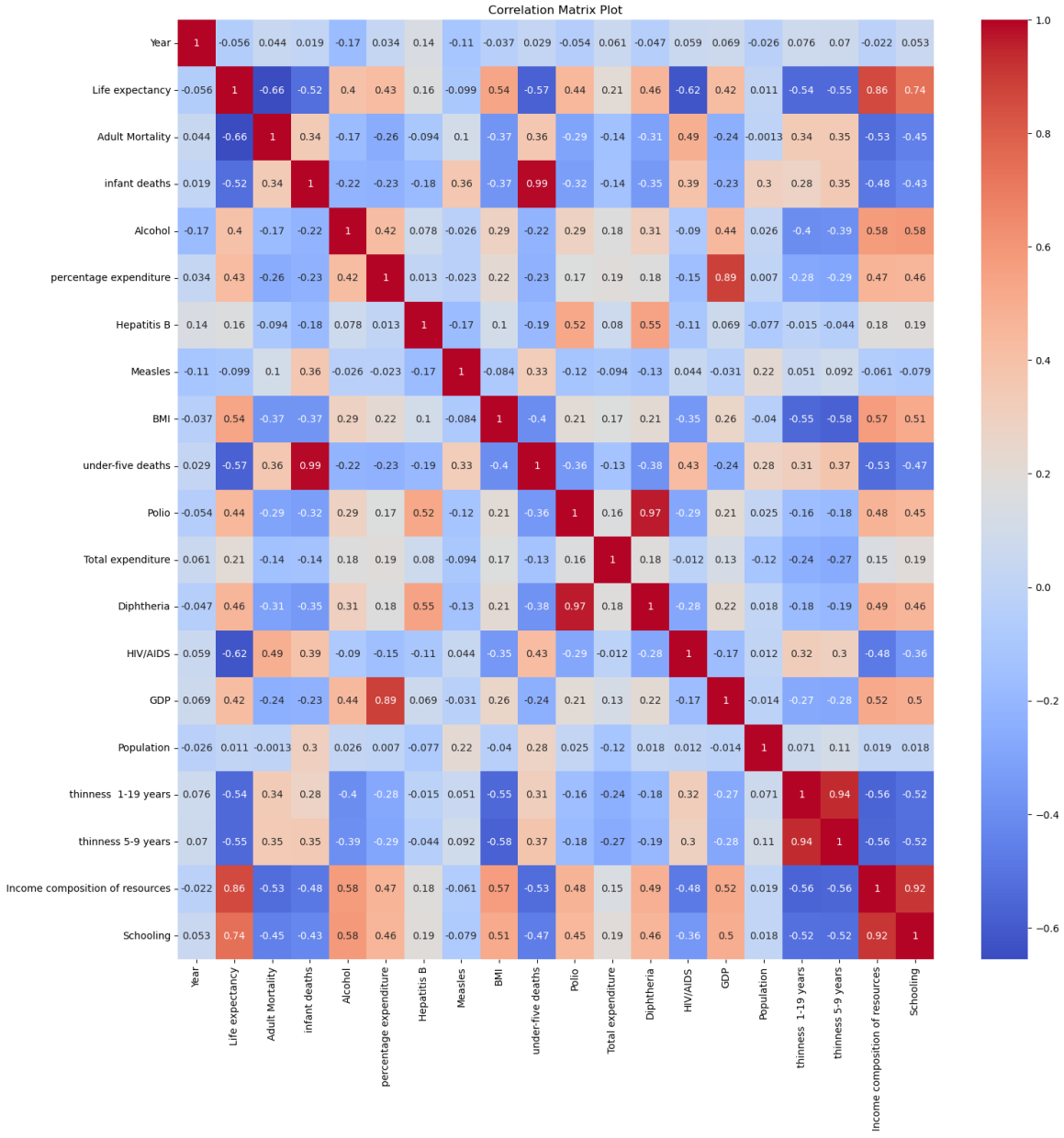
```
# Create dummy variables for 'Schooling'
schooling_dummies = pd.get_dummies(df_removed['Schooling'], prefix='Schooling')
# Create dummy variables for 'Income composition of resources'
icr_dummies = pd.get_dummies(df_removed['Income composition of resources'], prefix
# Concatenate the dummy variables with the original DataFrame
df_dummies = pd.concat([df_removed, schooling_dummies, icr_dummies], axis=1)
```

In [33]:

```
# Bins and labels for 'Schooling' dummy variables
schooling_bins = [0, 6, 12, 18, 24, float('inf')]
schooling_labels = ['Primary', 'Lower Secondary', 'Upper Secondary', 'Tertiary', '
df_removed['Schooling_Category'] = pd.cut(df_removed['Schooling'], bins=schooling_
# Bins and labels for 'Income composition of resources' dummy variables
icr_bins = [0, 0.25, 0.5, 0.75, 1.0, float('inf')]
icr_labels = ['Very Low', 'Low', 'Moderate', 'High', 'Very High']
df_removed['ICR_Category'] = pd.cut(df_removed['Income composition of resources'],
# Create dummy variables for the new categorical variables
schooling_dummies = pd.get_dummies(df_removed['Schooling_Category'], prefix='Schoo
icr_dummies = pd.get_dummies(df_removed['ICR_Category'], prefix='ICR')
# Concatenate the dummy variables with the dataset
df_dummies = pd.concat([df_removed, schooling_dummies, icr_dummies], axis=1)
```

In [34]:

```
# Compute the correlation matrix
correlation_matrix = df_removed.corr(numeric_only=True)
# Create a correlation matrix plot using a heatmap
plt.figure(figsize=(17, 17))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Plot')
plt.show()
```



In [35]:

```
# List of columns to remove
columns_to_remove = ['Year', 'percentage expenditure', 'Hepatitis B',
                     'Measles', 'Polio', 'Diphtheria','GDP', 'Population',
                     'thinness  1-19 years', 'thinness 5-9 years']
# Drop the columns
df_removed.drop(columns_to_remove, axis=1, inplace=True)
```

I chose to remove these variables as they didn't show the strength of correlation that the remaining columns I chose to use showed as predictors. In doing this I hope that my model predictors of life expectancy illustrate a better correlation.

In [36]:

```
# Defining my predictors
predictor_columns = ['Adult Mortality', 'infant deaths', 'Alcohol',
                     'BMI','under-five deaths', 'HIV/AIDS','Total expenditure',
                     'Income composition of resources', 'Schooling']
target_column = 'Life expectancy'
# Create the design matrix
X = df_removed[predictor_columns]
y = df_removed[target_column]
# Add constant to the design matrix
X = sm.add_constant(X)
# Create and fit the OLS model
model = sm.OLS(y, X)
results = model.fit()
# Display the model summary
results.summary()
```

Out[36]:

OLS Regression Results

Dep. Variable:	Life expectancy	**R-squared:**	0.852
Model:	OLS	**Adj. R-squared:**	0.851
Method:	Least Squares	**F-statistic:**	736.3
Date:	Thu, 27 Jul 2023	**Prob (F-statistic):**	0.00
Time:	18:16:08	**Log-Likelihood:**	-2886.0
No. Observations:	1162	**AIC:**	5792.
Df Residuals:	1152	**BIC:**	5843.
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	48.7196	0.786	61.950	0.000	47.177	50.263
Adult Mortality	-0.0137	0.001	-13.014	0.000	-0.016	-0.012
infant deaths	0.1102	0.032	3.422	0.001	0.047	0.173
Alcohol	-0.0957	0.029	-3.343	0.001	-0.152	-0.040
BMI	-0.0005	0.006	-0.097	0.923	-0.011	0.010
under-five deaths	-0.0997	0.024	-4.224	0.000	-0.146	-0.053
HIV/AIDS	-0.5428	0.049	-11.012	0.000	-0.639	-0.446
Total expenditure	0.3309	0.041	7.981	0.000	0.250	0.412
Income composition of resources	45.2149	1.895	23.858	0.000	41.497	48.933
Schooling	-0.5612	0.085	-6.564	0.000	-0.729	-0.393

Omnibus:	49.413	**Durbin-Watson:**	0.782
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	135.820
Skew:	0.133	**Prob(JB):**	3.21e-30
Kurtosis:	4.654	**Cond. No.**	4.36e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.36e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

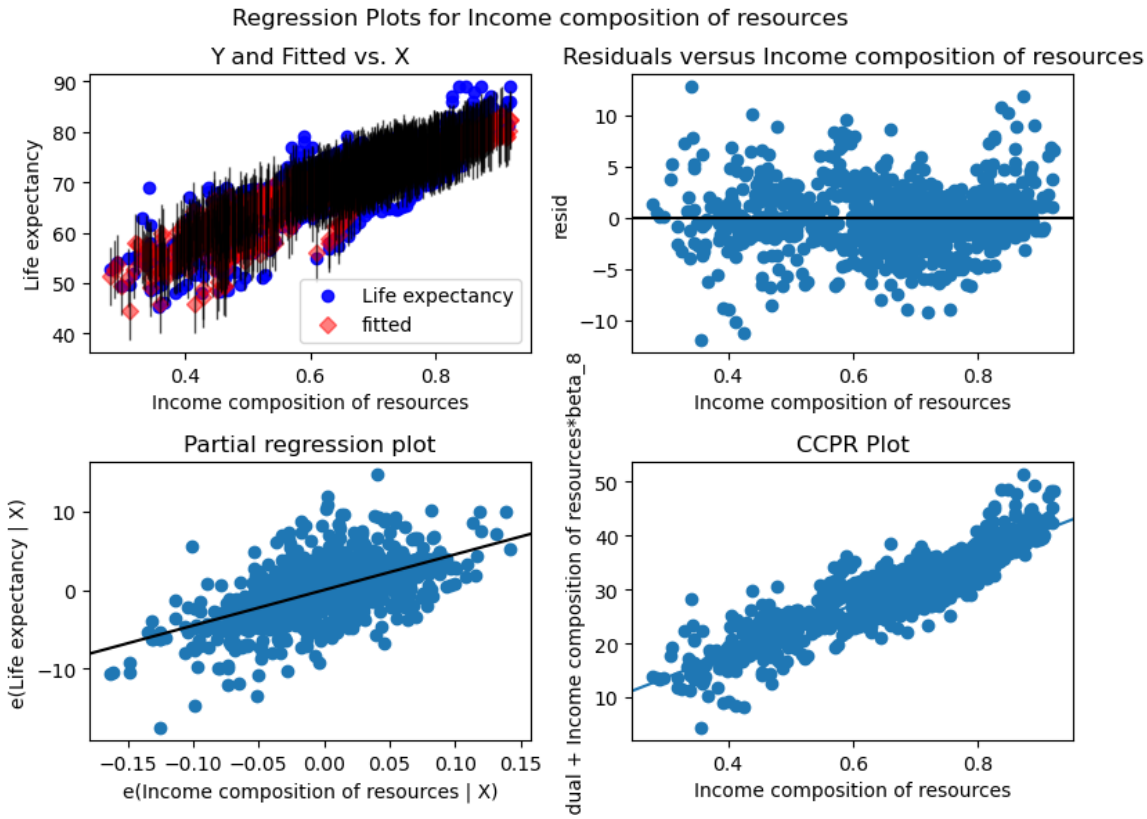
My documentation of the coefficients on my models second iteration here:

Higher adult mortality is linked to shorter life expectancy, and each one-unit increase in adult mortality reduces life expectancy by approximately 0.0137 years. Each additional infant death is associated with a slight increase of approximately 0.1102 years in life expectancy. Higher alcohol consumption is linked to lower life expectancy, with each unit increase reducing it by about 0.0957 years. BMI has little to no significant effect on life expectancy in this model. More deaths in children under five years old lead to lower life expectancy, with each additional death reducing it by about 0.0997 years. Higher HIV/AIDS prevalence

significantly reduces life expectancy, with each one-unit increase decreasing it by about 0.5428 years. Higher healthcare expenditure is associated with longer life expectancy, with each unit increase raising it by about 0.3309 years. Higher income composition leads to substantially longer life expectancy, with each increase raising it by approximately 45.2149 years. More years of schooling are associated with a longer life expectancy, with each increase raising it by about 0.5612 years.

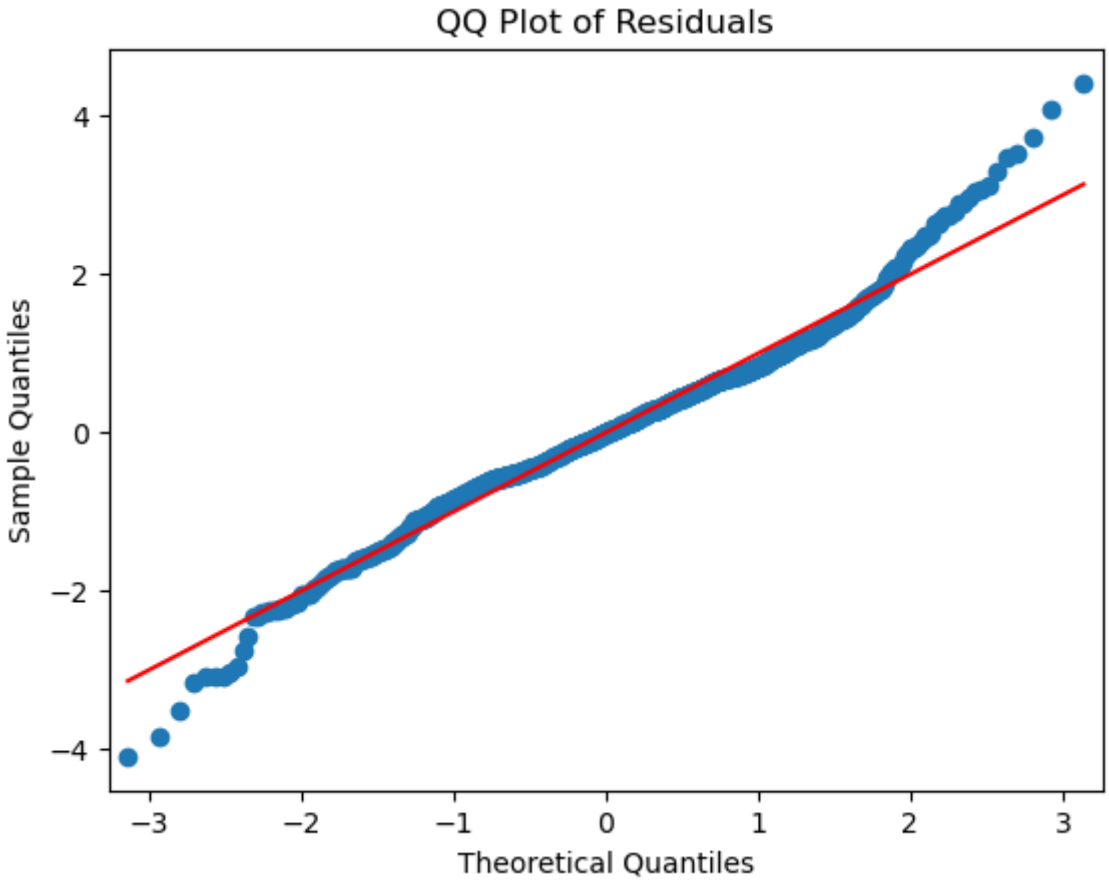
In [37]:

```
# Create the figure
fig = plt.figure(figsize=(8, 6))
# Plot the regression results for variable
sm.graphics.plot_regress_exog(results, 'Income composition of resources', fig=fig)
# Display the plot
plt.show()
```



In [38]:

```
# Fit the linear regression model
X = sm.add_constant(X)
model = sm.OLS(y, X)
results = model.fit()
# Create the QQ plot for residuals
sm.qqplot(results.resid, line='s', fit=True)
# Set plot title and labels
plt.title("QQ Plot of Residuals")
plt.xlabel("Theoretical Quantiles")
plt.ylabel("Sample Quantiles")
# Display the plot
plt.show()
```



Observing this QQ plot shows the removal of outliers, although the heaviers tails are showing a skew.

In [39]:

```
# Log transform the target variable
df_removed['Life expectancy_log'] = np.log(df_removed['Life expectancy'])
# Apply inverse transformation for predictions in the original scale
df_removed['Life expectancy_predictions'] = np.exp(df_removed['Life expectancy_log
```

In [40]:

```
import warnings
# Ignore the warning for log transformation
warnings.filterwarnings("ignore", message="divide by zero encountered in log")
# Define the variables to be log-transformed
variables_transformed = ['Adult Mortality', 'infant deaths', 'Alcohol',
                         'BMI','under-five deaths', 'HIV/AIDS',
                         'Income composition of resources', 'Schooling']
# Perform log transformations
for variable in variables_transformed:
    df_removed[f'{variable}_log'] = np.log(df_removed[variable])
# Apply inverse transformations for predictions in the original scale
for variable in variables_transformed:
    df_removed[f'{variable}_predictions'] = np.exp(df_removed[f'{variable}_log'])
```

In [41]:

```
# Define the dependent variable (y) and independent variables (X) for the OLS mode
y = df_removed['Life expectancy']
X = df_removed[['Adult Mortality_log', 'infant deaths_log', 'Alcohol_log', 'BMI_lo
                'under-five deaths_log', 'HIV/AIDS_log', 'Income composition of res
# Create the design matrix
X = df_removed[predictor_columns]
y = df_removed[target_column]
# Add constant to the design matrix
X = sm.add_constant(X)
# Create and fit the OLS model
model = sm.OLS(y, X)
results = model.fit()
# Display the model summary
results.summary()
```

Out[41]:

OLS Regression Results

Dep. Variable:	Life expectancy	**R-squared:**	0.852
Model:	OLS	**Adj. R-squared:**	0.851
Method:	Least Squares	**F-statistic:**	736.3
Date:	Thu, 27 Jul 2023	**Prob (F-statistic):**	0.00
Time:	18:16:10	**Log-Likelihood:**	-2886.0
No. Observations:	1162	**AIC:**	5792.
Df Residuals:	1152	**BIC:**	5843.
Df Model:	9		
Covariance Type:	nonrobust		

In [42]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Defining my predictors
predictor_columns = ['Adult Mortality', 'infant deaths', 'Alcohol',
                     'BMI','under-five deaths', 'HIV/AIDS',
                     'Income composition of resources', 'Schooling']
target_column = 'Life expectancy'
# Create the design matrix (predictors) and target variable
X = df_removed[predictor_columns]
y = df_removed[target_column]
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_st
# Create and fit the linear regression model on the training data
model = LinearRegression()
model.fit(X_train, y_train)
# Make predictions on both training and testing data
y_train_pred, y_test_pred = model.predict(X_train), model.predict(X_test)
# Calculate evaluation metrics for training and testing sets
train_mse, train_r2 = mean_squared_error(y_train, y_train_pred), r2_score(y_train,
test_mse, test_r2 = mean_squared_error(y_test, y_test_pred), r2_score(y_test, y_te
# Print and record the evaluation metrics
print("Training Set: MSE = %.4f, R-squared = %.4f" % (train_mse, train_r2))
print("Testing Set:  MSE = %.4f, R-squared = %.4f" % (test_mse, test_r2))
```

```
Training Set: MSE = 9.0539, R-squared = 0.8387
Testing Set:  MSE = 8.5347, R-squared = 0.8536
```

Training set: My model predictions on the training set have an average difference of approximately 9.05 units from the actual values of Life expectancy, which can be considered low. My model explains around 83.87% of the variability in Life expectancy in the training data with an R-squared of 0.8387.

Testing set: My model's predictions have an average difference of about 8.53 units from the actual 'Life expectancy' values. Which is considered quite a low difference. My model explains around 85.36% of the variability in the testing data with an R-squared of 0.8536

In [43]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_st
# Create and fit the model on the training data
model = LinearRegression()
model.fit(X_train, y_train)
# Make predictions on the testing data
y_pred = model.predict(X_test)
# Evaluate the model performance using metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
# Print the evaluation metrics
print("Mean Squared Error (MSE):", mse)
print("R-squared (R2):", r2)
```

```
Mean Squared Error (MSE): 8.534654010593574
R-squared (R2): 0.853647611780384
```

In [44]:

```
# Create and fit the final model on the entire dataset
final_model = LinearRegression()
final_model.fit(X, y)
# Use the final model for predictions on new, unseen data
us_data_pred = final_model.predict(df_removed[predictor_columns])
```

In [45]:

```
# Access the coefficient values
coefficients = final_model.coef_
# Access the intercept value
intercept = final_model.intercept_
# Print the coefficients and intercept
print("Coefficients:", coefficients)
print("Intercept:", intercept)
```

```
Coefficients: [-1.45531700e-02  9.78244298e-02 -7.37539149e-02  3.995
43654e-03
 -9.29430958e-02 -5.19180242e-01  4.34738820e+01 -4.75570933e-01]
Intercept: 50.6669714844501
```

In [46]:

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_st
# Train the model on the training set
model = LinearRegression()
model.fit(X_train, y_train)
# Make predictions on the training set and calculate evaluation metrics
y_train_pred = model.predict(X_train)
train_mse = mean_squared_error(y_train, y_train_pred)
train_r2 = r2_score(y_train, y_train_pred)
# Make predictions on the testing set and calculate evaluation metrics
y_test_pred = model.predict(X_test)
test_mse = mean_squared_error(y_test, y_test_pred)
test_r2 = r2_score(y_test, y_test_pred)
# Compare the performance on the training set and testing set
if train_mse < test_mse:
    print("The model may be overfitting.")
elif train_mse > test_mse:
    print("The model may be underfitting.")
else:
    print("The model's performance is similar on both training and testing sets.")
print("Training Set: MSE = %.4f, R-squared = %.4f" % (train_mse, train_r2))
print("Testing Set:  MSE = %.4f, R-squared = %.4f" % (test_mse, test_r2))
```

```
The model may be underfitting.
Training Set: MSE = 9.0539, R-squared = 0.8387
Testing Set:  MSE = 8.5347, R-squared = 0.8536
```

My observations of the coefficients and intercept In accordance to life expectancy

A one unit increase in higher adult mortality is associated with a slight decrease of approximately 0.0146. An increase of one infant death is linked to a higher life expectancy of approximately 0.0978 years. For every one unit increase in alcohol consumption, I have observed a decrease of approximately 0.0738 years. A one unit increase in BMI shows a small increase of approximately 0.00399 years. An increase of one under-five death is associated with a decrease of approximately 0.0929 years. For every one unit increase of HIV/AIDS

is linked to a decrease of approximately 0.5192 years. For every one unit increase in income composition of resources, shows a significant increase of around 43.4739 years. Higher levels of schooling are linked to a decrease of approximately 0.4756 years in life expectancy.

The intercept in the linear regression model indicates that predicted life expectancy, when all predictors are zero is roughly 50.67 years