In [ ]:
```
import numpy as np
import pandas as pd
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
Life = pd.read_csv('Life Expectancy Data.csv', encoding='latin1', sep=',')

print('It appears that column titles are not formatted well: not capitalized, additional spaces before and after th
display(Life.columns)

print("We'd need to fix this. Conversion performed in the following table 'Cols':")

Cols = pd.DataFrame(Life.columns.to_list(), columns=['Main'])

Cols['Beginning'] = Cols['Main'].str[0]
Cols['Ending'] = Cols['Main'].str[-1]
Cols['MainCorr'] = Cols['Main']

Cols.loc[Cols['Beginning'] == ' ', 'MainCorr'] = Cols['MainCorr'].str[1:]
Cols.loc[Cols['Ending'] == ' ', 'MainCorr'] = Cols['MainCorr'].str[:-1]

Cols['BeginningCorr'] = Cols['MainCorr'].str[0]
Cols['EndingCorr'] = Cols['MainCorr'].str[-1]

Cols['MainCorrBigFirst'] = Cols['MainCorr'].str.title()
Cols.loc[Cols['MainCorr'].isin(['BMI','HIV/AIDS','GDP']), 'MainCorrBigFirst'] = Cols['MainCorrBigFirst'].str.upper(

display(Cols)

print('Now the imported table looks as follows:')

Life.columns = Cols['MainCorrBigFirst'].tolist()
```

```
display(Life.columns)

print('And top 5 rows of the imported table looks as follows:')

display(Life.iloc[:, 0:12].head())
display(Life.iloc[:, 12:23].head())
```

```
It appears that column titles are not formatted well: not capitalized, additional spaces before and after the title
s:
Index(['Country', 'Year', 'Status', 'Life expectancy ', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',
       'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population',
       ' thinness  1-19 years', ' thinness 5-9 years',
       'Income composition of resources', 'Schooling'],
      dtype='object')
We'd need to fix this. Conversion performed in the following table 'Cols':
```

	Main	Beginning	Ending	MainCorr	BeginningCorr	EndingCorr	MainCorrBigFirst
0	Country	C	y	Country	C	y	Country
1	Year	Y	r	Year	Y	r	Year
2	Status	S	s	Status	S	s	Status
3	Life expectancy	L		Life expectancy	L	y	Life Expectancy
4	Adult Mortality	A	y	Adult Mortality	A	y	Adult Mortality
5	infant deaths	i	s	infant deaths	i	s	Infant Deaths
6	Alcohol	A	l	Alcohol	A	l	Alcohol
7	percentage expenditure	p	e	percentage expenditure	p	e	Percentage Expenditure
8	Hepatitis B	H	B	Hepatitis B	H	B	Hepatitis B
9	Measles	M		Measles	M	s	Measles
10	BMI			BMI	B	I	BMI
11	under-five deaths	u		under-five deaths	u	s	Under-Five Deaths
12	Polio	P	o	Polio	P	o	Polio
13	Total expenditure	T	e	Total expenditure	T	e	Total Expenditure
14	Diphtheria	D		Diphtheria	D	a	Diphtheria
15	HIV/AIDS		S	HIV/AIDS	H	S	HIV/AIDS
16	GDP	G	P	GDP	G	P	GDP
17	Population	P	n	Population	P	n	Population
18	thinness 1-19 years		s	thinness 1-19 years	t	s	Thinness 1-19 Years
19	thinness 5-9 years		s	thinness 5-9 years	t	s	Thinness 5-9 Years
20	Income composition of resources	I	s	Income composition of resources	I	s	Income Composition Of Resources
21	Schooling	S	g	Schooling	S	g	Schooling

```
Now the imported table looks as follows:
Index(['Country', 'Year', 'Status', 'Life Expectancy', 'Adult Mortality',
       'Infant Deaths', 'Alcohol', 'Percentage Expenditure', 'Hepatitis B',
       'Measles', 'BMI', 'Under-Five Deaths', 'Polio', 'Total Expenditure',
       'Diphtheria', 'HIV/AIDS', 'GDP', 'Population', 'Thinness  1-19 Years',
       'Thinness 5-9 Years', 'Income Composition Of Resources', 'Schooling'],
      dtype='object')
And top 5 rows of the imported table looks as follows:
```

	Country	Year	Status	Life Expectancy	Adult Mortality	Infant Deaths	Alcohol	Percentage Expenditure	Hepatitis B	Measles	BMI	Under-Five Deaths
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	19.1	83
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	18.6	86
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	18.1	89
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	17.6	93
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	17.2	97

	Polio	Total Expenditure	Diphtheria	HIV/AIDS	GDP	Population	Thinness 1-19 Years	Thinness 5-9 Years	Income Composition Of Resources	Schooling
0	6.0	8.16	65.0	0.1	584.259210	33736494.0	17.2	17.3	0.479	10.1
1	58.0	8.18	62.0	0.1	612.696514	327582.0	17.5	17.5	0.476	10.0
2	62.0	8.13	64.0	0.1	631.744976	31731688.0	17.7	17.7	0.470	9.9
3	67.0	8.52	67.0	0.1	669.959000	3696958.0	17.9	18.0	0.463	9.8
4	68.0	7.87	68.0	0.1	63.537231	2978599.0	18.2	18.2	0.454	9.5

In [ ]:
```
Life.shape
```

Out[ ]: (2938, 22)

In [ ]:
```
# Amount of unique countries in the table
len(Life['Country'].unique().tolist())
```

Out[ ]: 193

In [ ]:
```
# Amount of years available for how many countries:
Life['Country'].value_counts().to_frame().reset_index().rename(columns = {'count' : 'Count of years'}).groupby(by =
```

Out[ ]:

	Count of years	Country
0	1	10
1	16	183

Most of countries got data for 16 years except 10 countries which have stats for one year only.

We can drop countries which got stats only for one year, because that's not enough data to build regression

In [ ]:
```
List = Life['Country'].value_counts().to_frame().reset_index().rename(columns = {'count' : 'Count of years'})
List = List.where(List['Count of years'] == 16).dropna()['Country']
List = List.tolist()

print('Before the drop of these countries:', Life.shape)

Life = Life.loc[Life['Country'].isin(List)]

print('After the drop of these countries:', Life.shape)

print(' ')
print('Top 5 rows of the imported table looks as follows:')
display(Life.iloc[:, 0:12].head())
display(Life.iloc[:, 12:23].head())
```

```
Before the drop of these countries: (2938, 22)
After the drop of these countries: (2928, 22)

Top 5 rows of the imported table looks as follows:
```

	Country	Year	Status	Life Expectancy	Adult Mortality	Infant Deaths	Alcohol	Percentage Expenditure	Hepatitis B	Measles	BMI	Under-Five Deaths
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	19.1	83
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	18.6	86
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	18.1	89
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	17.6	93
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	17.2	97

	Polio	Total Expenditure	Diphtheria	HIV/AIDS	GDP	Population	Thinness 1-19 Years	Thinness 5-9 Years	Income Composition Of Resources	Schooling
0	6.0	8.16	65.0	0.1	584.259210	33736494.0	17.2	17.3	0.479	10.1
1	58.0	8.18	62.0	0.1	612.696514	327582.0	17.5	17.5	0.476	10.0
2	62.0	8.13	64.0	0.1	631.744976	31731688.0	17.7	17.7	0.470	9.9
3	67.0	8.52	67.0	0.1	669.959000	3696958.0	17.9	18.0	0.463	9.8
4	68.0	7.87	68.0	0.1	63.537231	2978599.0	18.2	18.2	0.454	9.5

In [ ]:
```
#We've got 183 coutries with data available for 16 years
len(Life['Country'].unique())
```

Out[ ]: 183

In [ ]:
```
# We've got 2 types of coutries: Developing and Developed.
Life['Status'].unique()
```

Out[ ]: array(['Developing', 'Developed'], dtype=object)

In [ ]:
```
#we've got 32 developed coutries & 151 developing coutries
Life[['Country', 'Status']].copy().drop_duplicates().value_counts().to_frame().reset_index().groupby(by = 'Status')
```

Out[ ]:

Status	count
Developed	32
Developing	151

In [ ]:
```
print('Example of Developed countries:')
display(Life.loc[Life['Status']=='Developed'][Life.columns[0:11]].head(1))
display(Life.loc[Life['Status']=='Developed'][Life.columns[12:23]].head(1))
print('')
print('Example of Developing countries:')
display(Life.loc[Life['Status']=='Developing'][Life.columns[0:11]].head(1))
display(Life.loc[Life['Status']=='Developing'][Life.columns[12:23]].head(1))
```

```
Example of Developed countries:
```

	Country	Year	Status	Life Expectancy	Adult Mortality	Infant Deaths	Alcohol	Percentage Expenditure	Hepatitis B	Measles	BMI
112	Australia	2015	Developed	82.8	59.0	1	NaN	0.0	93.0	74	66.6

	Polio	Total Expenditure	Diphtheria	HIV/AIDS	GDP	Population	Thinness 1-19 Years	Thinness 5-9 Years	Income Composition Of Resources	Schooling
112	93.0	NaN	93.0	0.1	56554.3876	23789338.0	0.6	0.6	0.937	20.4

```
Example of Developing countries:
```

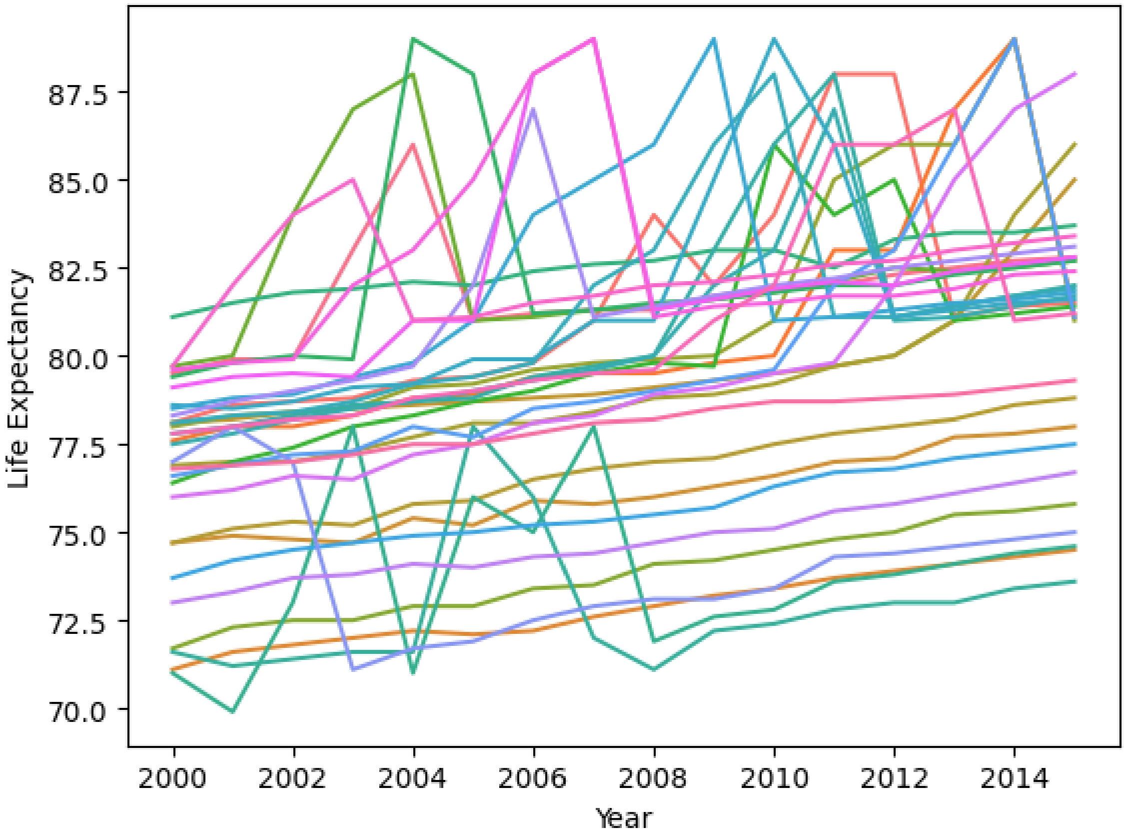
	Country	Year	Status	Life Expectancy	Adult Mortality	Infant Deaths	Alcohol	Percentage Expenditure	Hepatitis B	Measles	BMI
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	19.1

	Polio	Total Expenditure	Diphtheria	HIV/AIDS	GDP	Population	Thinness 1-19 Years	Thinness 5-9 Years	Income Composition Of Resources	Schooling
0	6.0	8.16	65.0	0.1	584.25921	33736494.0	17.2	17.3	0.479	10.1

In [ ]:
```
LiveDeveloped = Life.loc[Life['Status']=='Developed']
LiveDeveloping = Life.loc[Life['Status']=='Developing']
```

In [ ]:
```
sns.lineplot(data=LiveDeveloped, x='Year', y='Life Expectancy', hue='Country', legend=False)
```

Out[ ]: <Axes: xlabel='Year', ylabel='Life Expectancy'>



In [ ]:
```
#HIV/AIDS appear to be constant, so no correlation to any value, could be excluded.
LiveDeveloped['HIV/AIDS'].unique()
```

Out[ ]:
```
array([0.1])
```

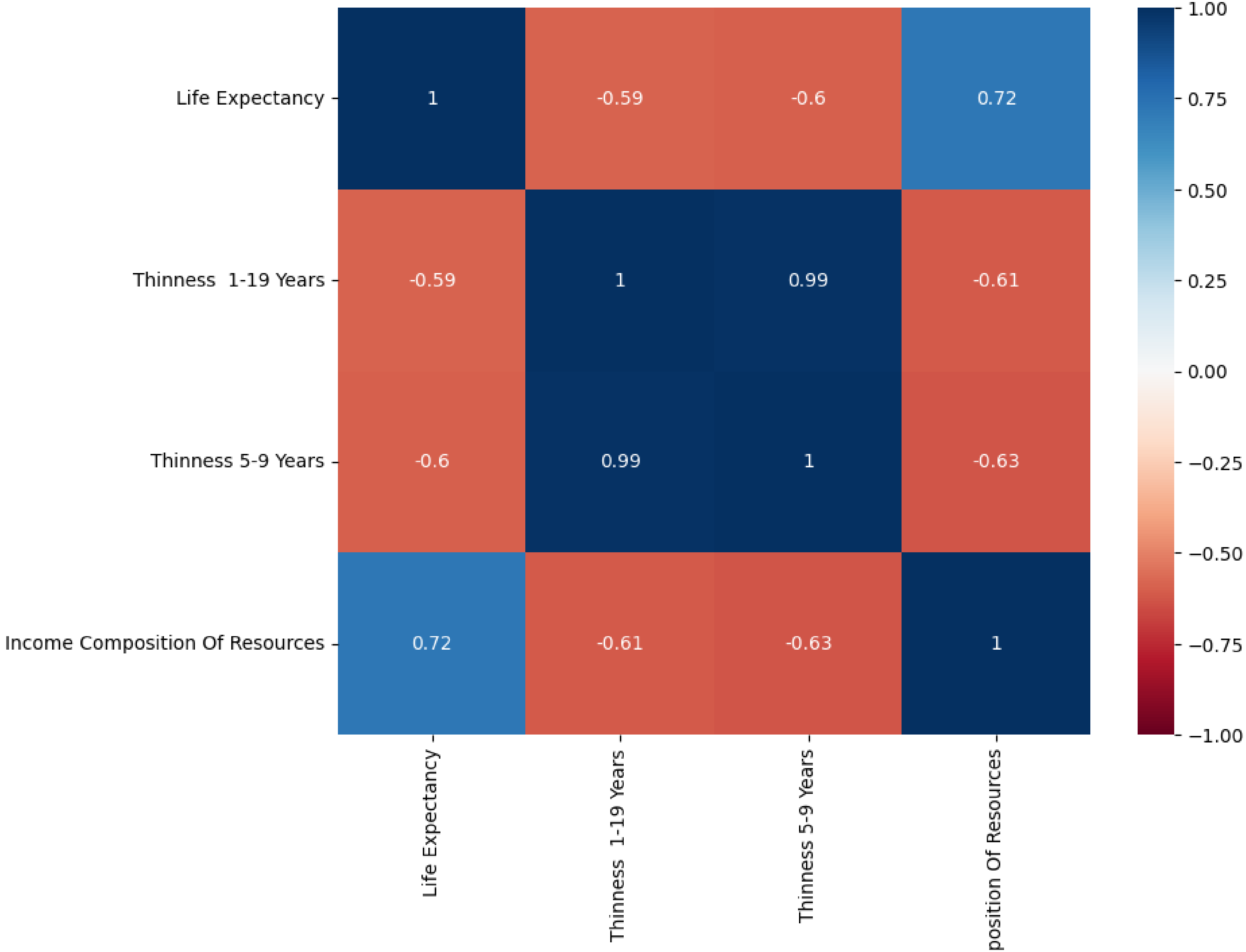
In [ ]:
```
#checking correlation
corrmatrix = pd.concat([LiveDeveloped[LiveDeveloped.columns[3:15]],LiveDeveloped[LiveDeveloped.columns[16:23]]], ax
corrmatrix = corrmatrix.round(2)

# Removing values where is no correlation
# 0.5+ - weak correlation
# 0.7+ - some correlation
# 0.9+ - strong correlation

for each in pd.concat([LiveDeveloped[LiveDeveloped.columns[3:15]],LiveDeveloped[LiveDeveloped.columns[16:23]]], axi
    corrmatrix.loc[(corrmatrix[each] > -0.5) & (corrmatrix[each] < 0.5), each] = ''
    corrmatrix.loc[(corrmatrix[each] == 1), each] = ''

corrmatrix = corrmatrix.iloc[[0]]

important = pd.melt(corrmatrix.reset_index(), id_vars=['index'])
important = important.loc[(important['value'] != '')]
important = pd.concat([important[['index']], important[['variable']].rename(columns={'variable' : 'index'})], axis=
important

print('Most important metrics for Developed countries:')
display(corrmatrix[important[1:len(important)]])
```

Most important metrics for Developed countries:

	Thinness 1-19 Years	Thinness 5-9 Years	Income Composition Of Resources
Life Expectancy	-0.59	-0.6	0.72

In [ ]:
```
plt.figure(figsize=(9,7))
sns.heatmap(LiveDeveloped[important].corr(),vmin=-1.0,vmax=1.0, cmap='RdBu', annot=True)
```

Out[ ]: <Axes: >


Life Expectancy
Thinness 1-19 Years
Thinness 5-9 Years
Income Composition Of Resources
1
-0.59
-0.6
0.72
-0.59
1
0.99
-0.61
-0.6
0.99
1
-0.63
0.72
-0.61
-0.63
1
Life Expectancy
Thinness 1-19 Years
Thinness 5-9 Years
position Of Resources
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Income Com

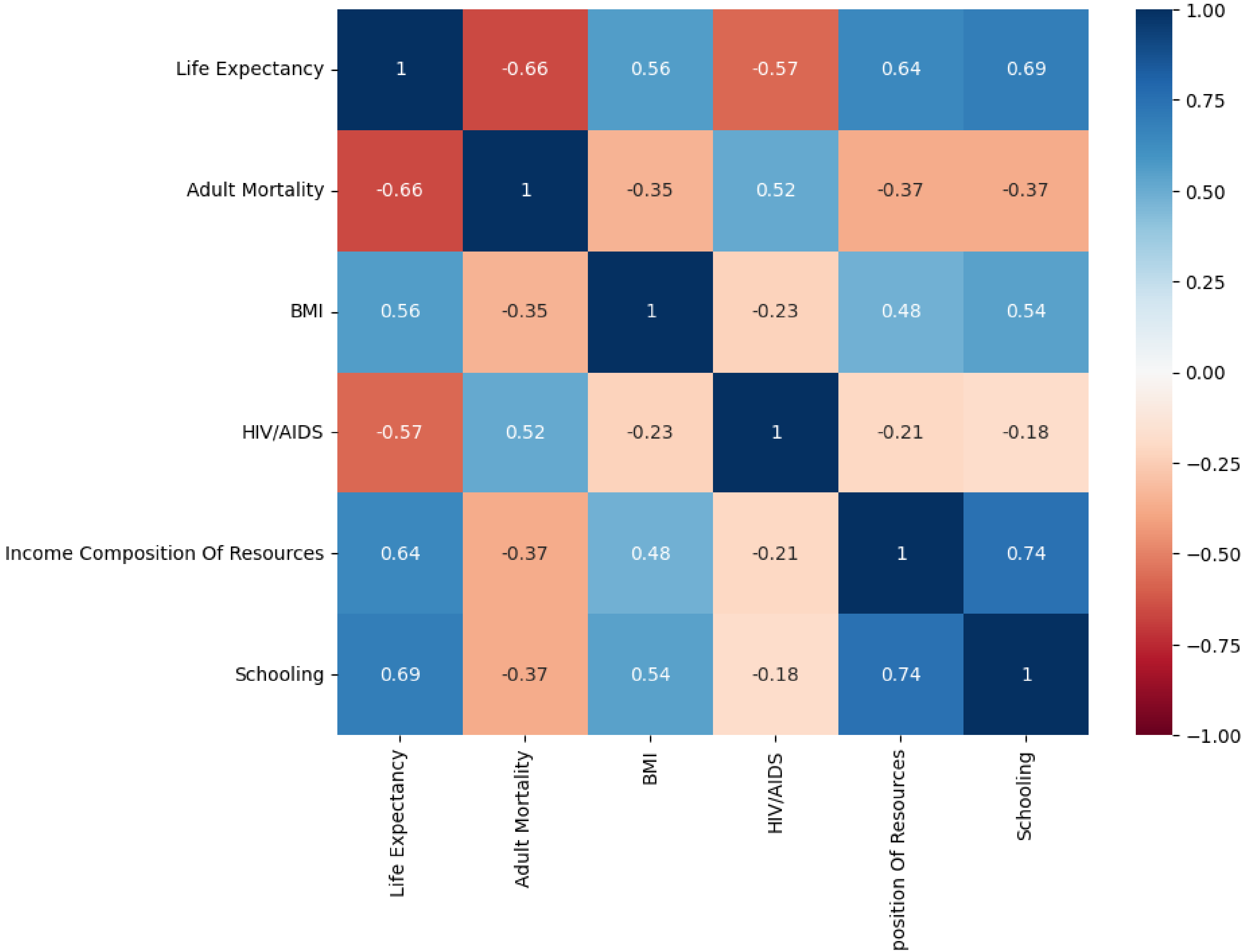
In [ ]:
```
#checking correlation
corrmatrix = LiveDeveloping[LiveDeveloped.columns[3:23]].corr()
corrmatrix = corrmatrix.round(2)

# Removing values where is no correlation
# 0.5+ - weak correlation
# 0.7+ - some correlation
# 0.9+ - strong correlation

for each in LiveDeveloping[LiveDeveloped.columns[3:23]].columns:
    corrmatrix.loc[(corrmatrix[each] > -0.5) & (corrmatrix[each] < 0.5), each] = ''
    corrmatrix.loc[(corrmatrix[each] == 1), each] = ''

corrmatrix = corrmatrix.iloc[[0]]

important2 = pd.melt(corrmatrix.reset_index(), id_vars=['index'])
important2 = important2.loc[(important2['value'] != '')]
important2 = pd.concat([important2[['index']], important2[['variable']].rename(columns={'variable' : 'index'})], ax
important2

print('Most important metrics for Developing countries:')
display(corrmatrix[important2[1:len(important2)]])
```

```
Most important metrics for Developing countries:
```

	Adult Mortality	BMI	HIV/AIDS	Income Composition Of Resources	Schooling
Life Expectancy	-0.66	0.56	-0.57	0.64	0.69

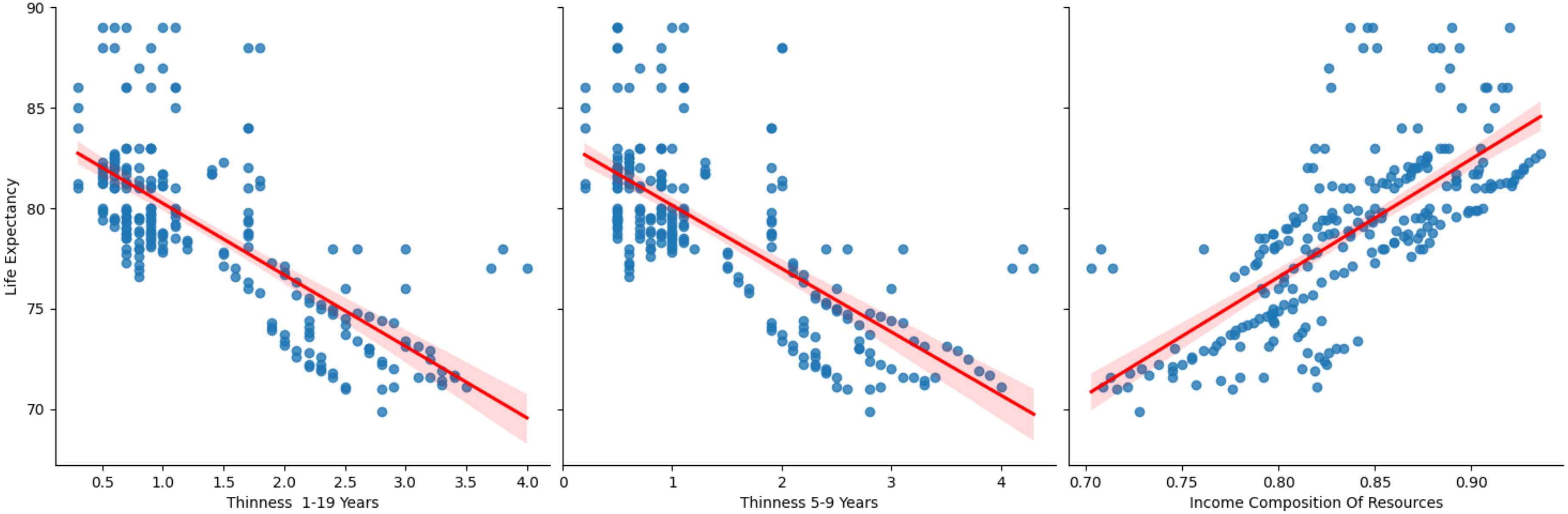
In [ ]:
```
plt.figure(figsize=(9,7))
sns.heatmap(LiveDeveloping[important2].corr(),vmin=-1.0,vmax=1.0, cmap='RdBu', annot=True)
```

Out[ ]: <Axes: >


Life Expectancy
Adult Mortality
BMI
HIV/AIDS
Income Composition Of Resources
Schooling
1
-0.66
0.56
-0.57
0.64
0.69
-0.66
1
-0.35
0.52
-0.37
-0.37
0.56
-0.35
1
-0.23
0.48
0.54
-0.57
0.52
-0.23
1
-0.21
-0.18
0.64
-0.37
0.48
-0.21
1
0.74
0.69
-0.37
0.54
-0.18
0.74
1
Life Expectancy
Adult Mortality
BMI
HIV/AIDS
position Of Resources
Schooling
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

In [ ]:
```
print('Regression for developed countries:')

LiveDevelopedModel = LiveDeveloped.copy().dropna(how='any')

sns.pairplot(LiveDevelopedModel, x_vars = important[1:len(important)], y_vars = 'Life Expectancy',
             height = 5, kind = 'reg', plot_kws={'line_kws':{'color':'red'}})
plt.show()
```

```
Regression for developed countries:
```



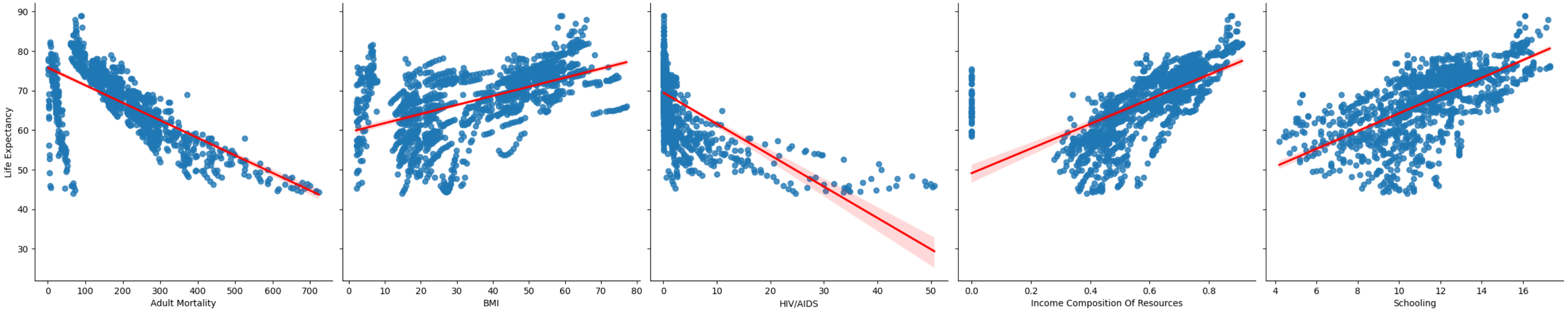
In [ ]:
```
Xdeveloped = LiveDevelopedModel[important[1:len(important)]].dropna(how='any')
Ydeveloped = LiveDevelopedModel['Life Expectancy']

X_train_developed, X_test_developed, y_train_developed, y_test_developed = train_test_split(Xdeveloped, Ydeveloped,

reg = linear_model.LinearRegression()
```

```
reg.fit(X_train_developed,y_train_developed)

print("Developed countries:")
print("Coefficients:" ,reg.coef_)
print("Intercept:", reg.intercept_)

# 3 Coefficients because 3 x-variables
# 1 intercept because 1 y-variable

Y_pred_developed = reg.predict(X_train_developed)

mse = mean_squared_error(y_train_developed, Y_pred_developed)
r2score = r2_score(y_train_developed, Y_pred_developed)

print('')
print('Train dataset for developed countried returns')
print("MSE = ", mse)
print("R2s = ", r2score)

Y_pred_developed = reg.predict(X_test_developed)

mse = mean_squared_error(y_test_developed, Y_pred_developed)
r2score = r2_score(y_test_developed, Y_pred_developed)

print('')
print('Test dataset for developed countried returns')
print("MSE = ", mse)
print("R2s = ", r2score)
```

```
Developed countries:
Coefficients: [-8.04829515  5.13688669 34.43174921]
Intercept: 54.14530391530003

Train dataset for developed countried returns
MSE =  7.449236291505592
R2s =  0.6350422263342237

Test dataset for developed countried returns
MSE =  4.221716089434478
R2s =  0.6730083137611444
```

In [ ]:
```
LiveDevelopingModel = LiveDeveloping.copy().dropna(how='any')

print('Regression for developing countries:')
sns.pairplot(LiveDevelopingModel, x_vars = important2[1:len(important2)], y_vars = 'Life Expectancy',
             height = 5, kind = 'reg', plot_kws={'line_kws':{'color':'red'}})
plt.show()
```

```
Regression for developing countries:
```



In [ ]:
```
Xdeveloping = LiveDevelopingModel[important2[1:len(important2)]].dropna(how='any')
Ydeveloping = LiveDevelopingModel['Life Expectancy']

X_train_developing, X_test_developing, y_train_developing, y_test_developing = train_test_split(Xdeveloping, Ydevel

reg = linear_model.LinearRegression()
reg.fit(X_train_developing,y_train_developing)

print("Developing countries:")
print("Coefficients:" ,reg.coef_)
print("Intercept:", reg.intercept_)

# 3 Coefficients because 3 x-variables
# 1 intercept because 1 y-variable

Y_pred_developing = reg.predict(X_train_developing)

mse = mean_squared_error(y_train_developing, Y_pred_developing)
r2score = r2_score(y_train_developing, Y_pred_developing)
```

```
print('')
print('Train dataset for developing countries returns')
print("MSE = ", mse)
print("R2s = ", r2score)

Y_pred_developing = reg.predict(X_test_developing)

mse = mean_squared_error(y_test_developing, Y_pred_developing)
r2score = r2_score(y_test_developing, Y_pred_developing)

print('')
print('Test dataset for developing countries returns')
print("MSE = ", mse)
print("R2s = ", r2score)
```

```
Developing countries:
Coefficients: [-0.01741115  0.04619914 -0.47396179  8.23696992  1.14560385]
Intercept: 52.21238298521717

Train dataset for developing countries returns
MSE =  14.392630042879325
R2s =  0.7980632369272288

Test dataset for developing countries returns
MSE =  14.89696961189324
R2s =  0.7748222668236822
```