```
# prompt: import the necessary libraries  for data anaysis

# Common libraries
import pandas as pd
import numpy as np

# Data visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

# Statistical analysis
from scipy import stats

# Machine learning
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data=pd.read_csv("F:\datasets\indianEco.csv")
data

    Year Country Name  GDP (current US$)    GDP per capita (current
US$)   \
0   1960        India       3.702988e+10
82
1   1961        India       3.923244e+10
85
2   1962        India       4.216148e+10
90
3   1963        India       4.842192e+10
101
4   1964        India       5.648029e+10
116
..   ...          ...                ...
...
56  2016        India       2.294798e+12
1733
57  2017        India       2.651473e+12
1981
58  2018        India       2.702930e+12
1997
59  2019        India       2.831552e+12
2101
60  2020        India       2.667688e+12
1928

    GDP growth (annual %)  Imports of goods and services (% of GDP)  \
0                    0.00                                      6.83
```

1	3.72	5.96
2	2.93	6.03
3	5.99	5.91
4	7.45	5.69
..	...	...
56	8.26	20.92
57	6.80	21.95
58	6.53	23.69
59	4.04	21.27
60	-7.25	19.10

	Exports of goods and services (% of GDP) \
0	4.46
1	4.30
2	4.17
3	4.28
4	3.73
..	...
56	19.16
57	18.79
58	19.93
59	18.69
60	18.71

	Total reserves (includes gold, current US$) \
0	6.745366e+08
1	6.663571e+08
2	5.127918e+08
3	6.078625e+08
4	4.991451e+08
..	...
56	3.616943e+11
57	4.126138e+11
58	3.991672e+11
59	4.634699e+11
60	5.902274e+11

	Inflation, consumer prices (annual %)	Population, total \
0	1.78	445954579
1	1.70	456351876
2	3.63	467024193
3	2.95	477933619
4	13.36	489059309
..	...	...
56	4.95	1338636340
57	3.33	1354195680
58	3.94	1369003306
59	3.73	1383112050
60	6.62	1396387127

```
    Population growth (annual %)  Life expectancy at birth, total
(years)
0                           2.31
41.13
1                           2.33
41.74
2                           2.34
42.34
3                           2.34
42.94
4                           2.33
43.57
..                           ...
...
56                          1.19
68.67
57                          1.16
68.97
58                          1.09
69.27
59                          1.03
69.50
60                          0.96
69.73

[61 rows x 12 columns]
```

```
data.isnull().sum()
```

```
Year                                              0
Country Name                                      0
GDP (current US$)                                 0
 GDP per capita (current US$)                     0
GDP growth (annual %)                             0
Imports of goods and services (% of GDP)          0
Exports of goods and services (% of GDP)          0
 Total reserves (includes gold, current US$)      0
Inflation, consumer prices (annual %)             0
Population, total                                 0
Population growth (annual %)                      0
Life expectancy at birth, total (years)           0
dtype: int64
```

```
data.dtypes
```

```
Year                                                    int64
Country Name                                           object
GDP (current US$)                                     float64
 GDP per capita (current US$)                           int64
GDP growth (annual %)                                 float64
```

```
Imports of goods and services (% of GDP)         float64
Exports of goods and services (% of GDP)         float64
 Total reserves (includes gold, current US$)     float64
Inflation, consumer prices (annual %)            float64
Population, total                                  int64
Population growth (annual %)                     float64
Life expectancy at birth, total (years)          float64
dtype: object
```

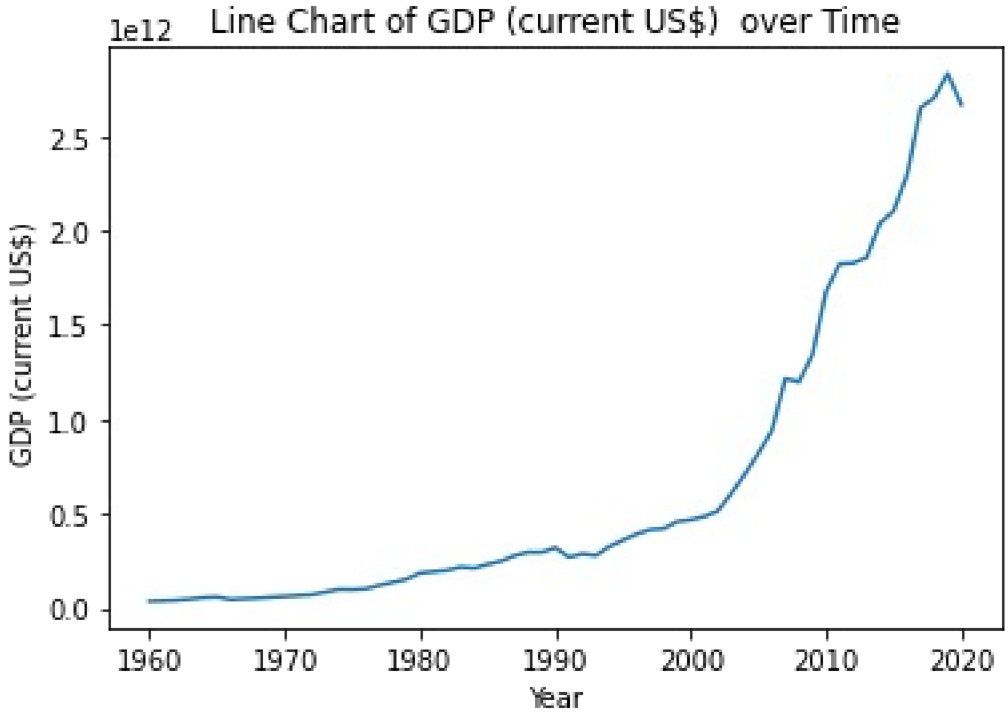
```
data.describe()
```

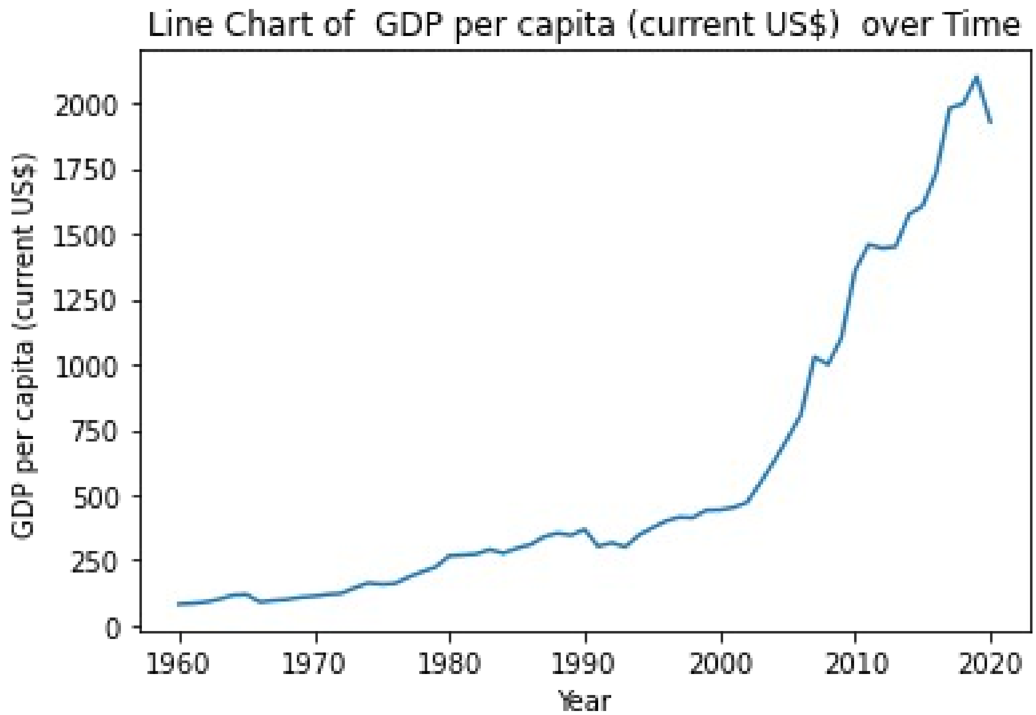
```
              Year  GDP (current US$)    GDP per capita (current US$)  \
count    61.000000       6.100000e+01                        61.000000
mean   1990.000000       6.584725e+11                       575.557377
std      17.752934       8.129606e+11                       584.079062
min    1960.000000       3.702988e+10                        82.000000
25%    1975.000000       9.952590e+10                       161.000000
50%    1990.000000       2.882084e+11                       340.000000
75%    2005.000000       8.203816e+11                       715.000000
max    2020.000000       2.831552e+12                      2101.000000

       GDP growth (annual %)  Imports of goods and services (% of GDP)  \
count              61.000000                                  61.000000
mean                4.938197                                  12.746393
std                 3.344891                                   8.155110
min                -7.250000                                   3.710000
25%                 3.720000                                   6.590000
50%                 5.530000                                   8.570000
75%                 7.450000                                  19.640000
max                 9.630000                                  31.260000

       Exports of goods and services (% of GDP)  \
count                                 61.000000
```

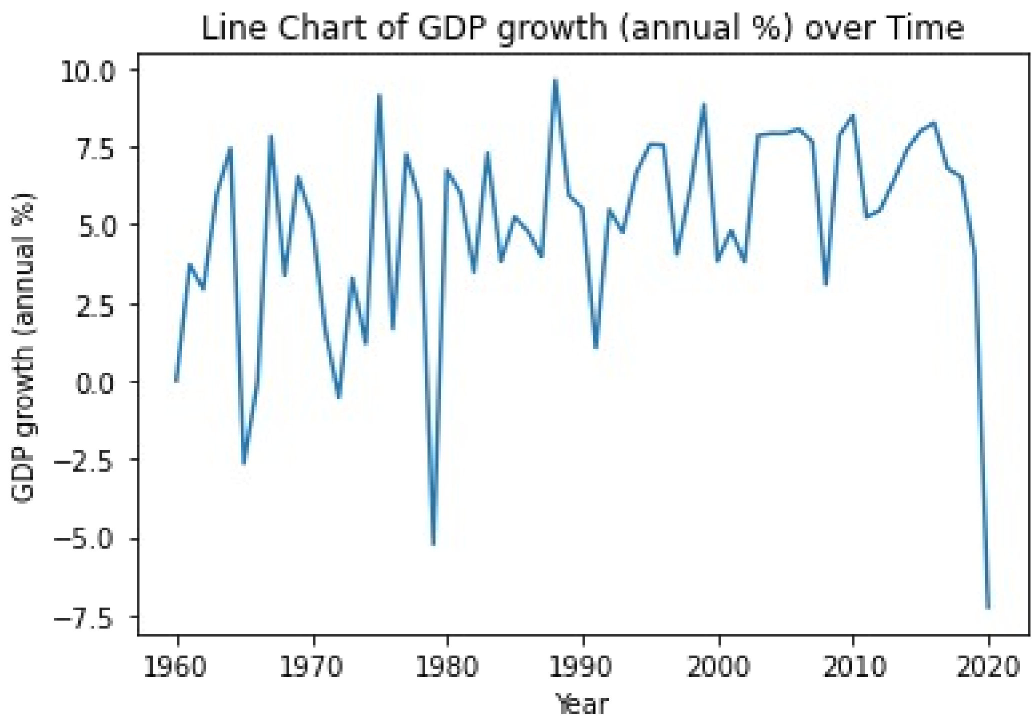
```
mean                                    10.885574
std                                      7.060458
min                                      3.310000
25%                                      5.200000
50%                                      7.050000
75%                                     18.690000
max                                     25.430000

        Total reserves (includes gold, current US$)   \
count                                     6.100000e+01
mean                                      9.802227e+10
std                                       1.497102e+11
min                                       4.991451e+08
25%                                       2.324650e+09
50%                                       1.151174e+10
75%                                       1.378248e+11
max                                       5.902274e+11

       Inflation, consumer prices (annual %)  Population, total  \
count                              61.000000       6.100000e+01
mean                                7.413279       8.913946e+08
std                                 4.940153       2.974496e+08
min                                -7.630000       4.459546e+08
25%                                 4.010000       6.235242e+08
50%                                 6.670000       8.704522e+08
75%                                10.020000       1.154639e+09
max                                28.600000       1.396387e+09

       Population growth (annual %)  Life expectancy at birth, total
(years)
count                     61.000000
61.000000
mean                       1.927705
57.146230
std                        0.419024
8.459559
min                        0.960000
41.130000
25%                        1.620000
50.630000
50%                        2.150000
57.660000
75%                        2.260000
64.310000
max                        2.340000
69.730000
```

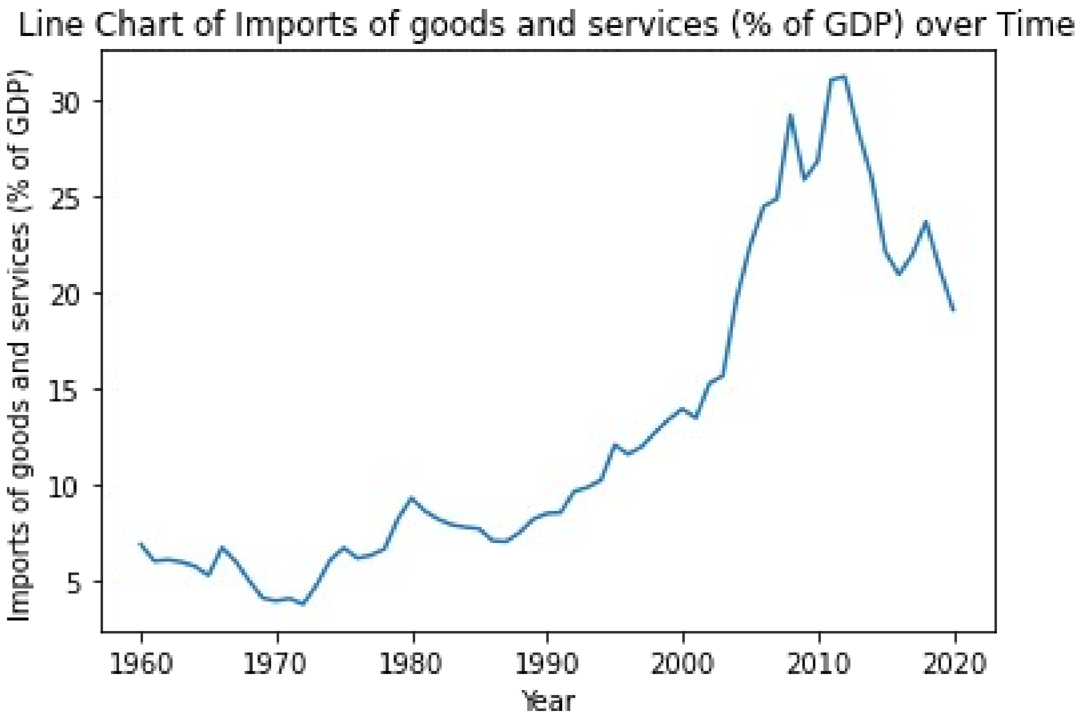
```
# prompt: plot the line chat of every column of the data with time
except the column year
```

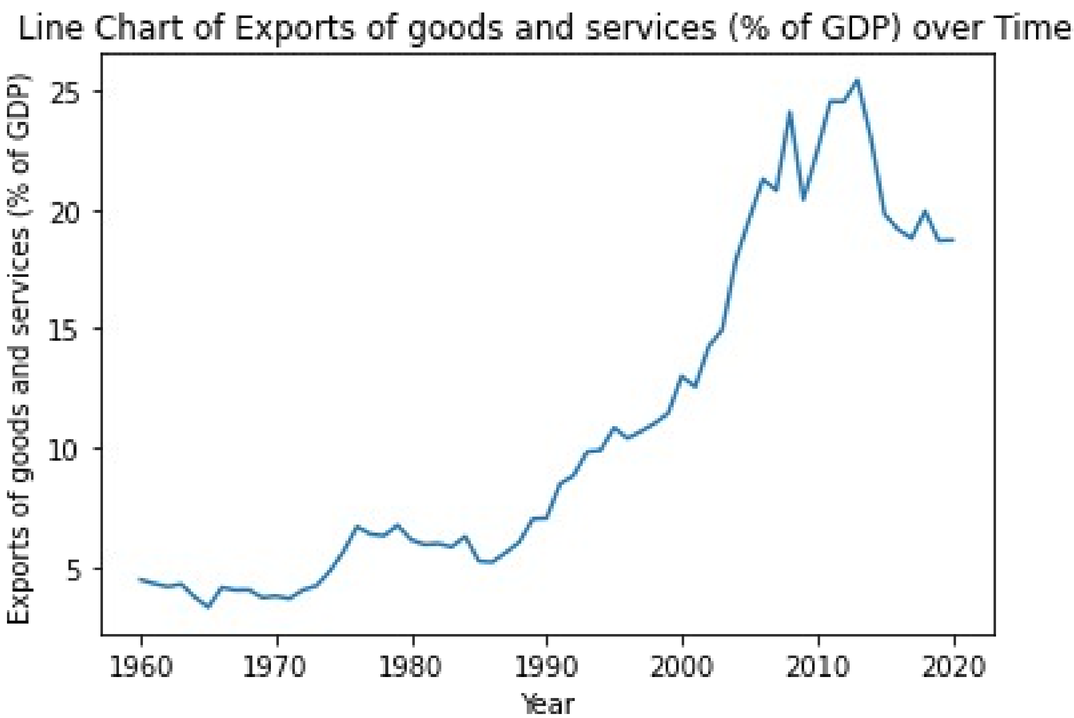
```
import matplotlib.pyplot as plt
for col in data.columns:
  if col not in ('Year', 'Country Name'):
    plt.plot(data['Year'], data[col])
    plt.xlabel('Year')
    plt.ylabel(col)
    plt.title(f'Line Chart of {col} over Time')
    plt.show()
```

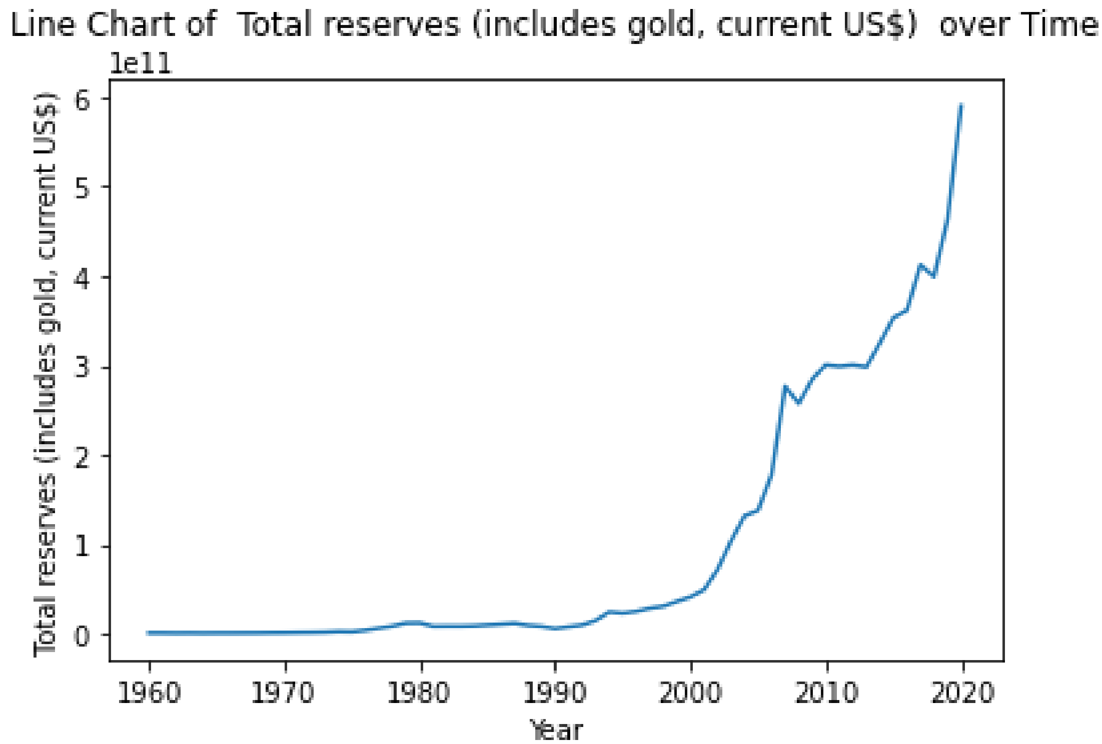


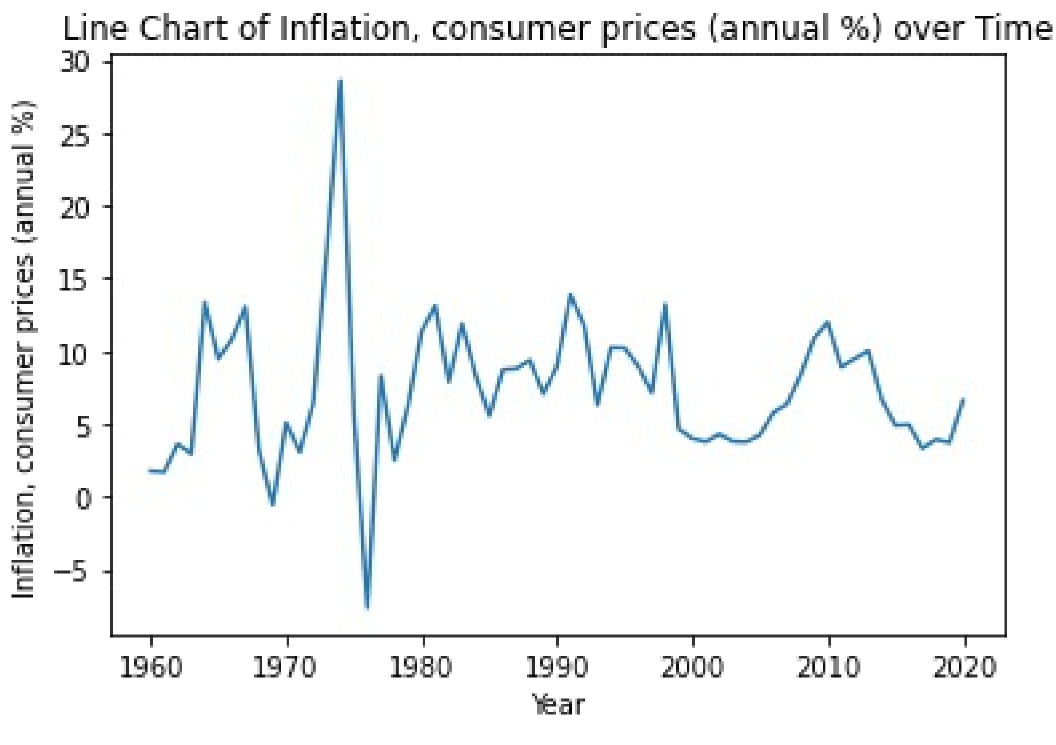

Line Chart of GDP per capita (current US$) over Time
GDP per capita (current US$)
0
250
500
750
1000
1250
1500
1750
2000
1960
1970
1980
1990
2000
2010
2020
Year

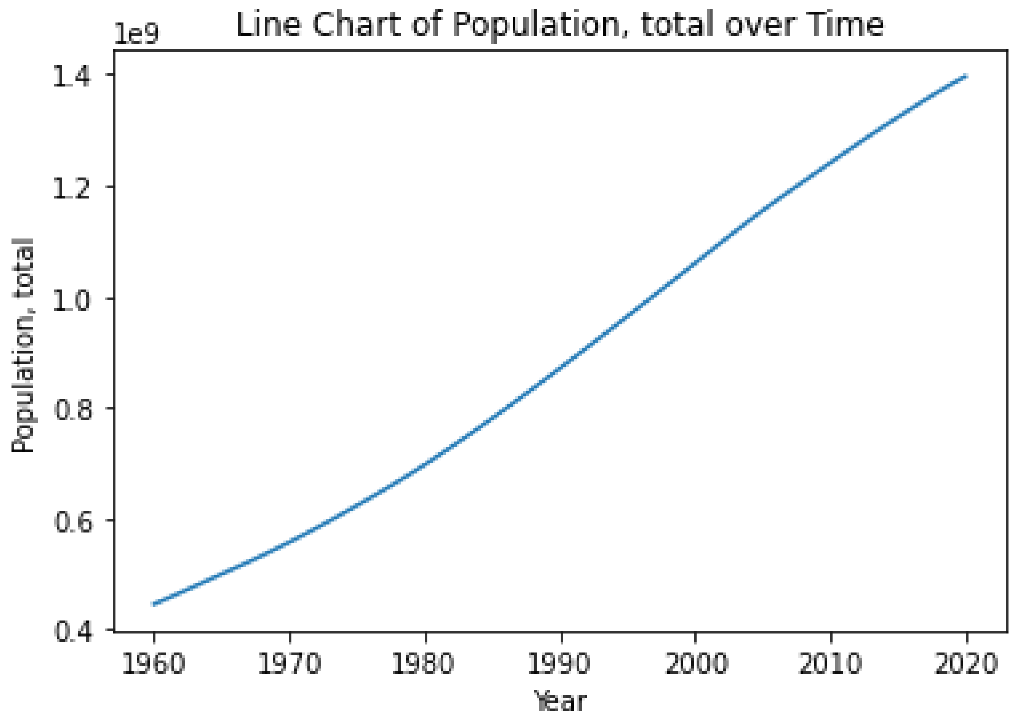

Line Chart of GDP growth (annual %) over Time
GDP growth (annual %)
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
1960
1970
1980
1990
2000
2010
2020
Year


Line Chart of Imports of goods and services (% of GDP) over Time
Imports of goods and services (% of GDP)
30
25
20
15
10
5
1960
1970
1980
1990
2000
2010
2020
Year

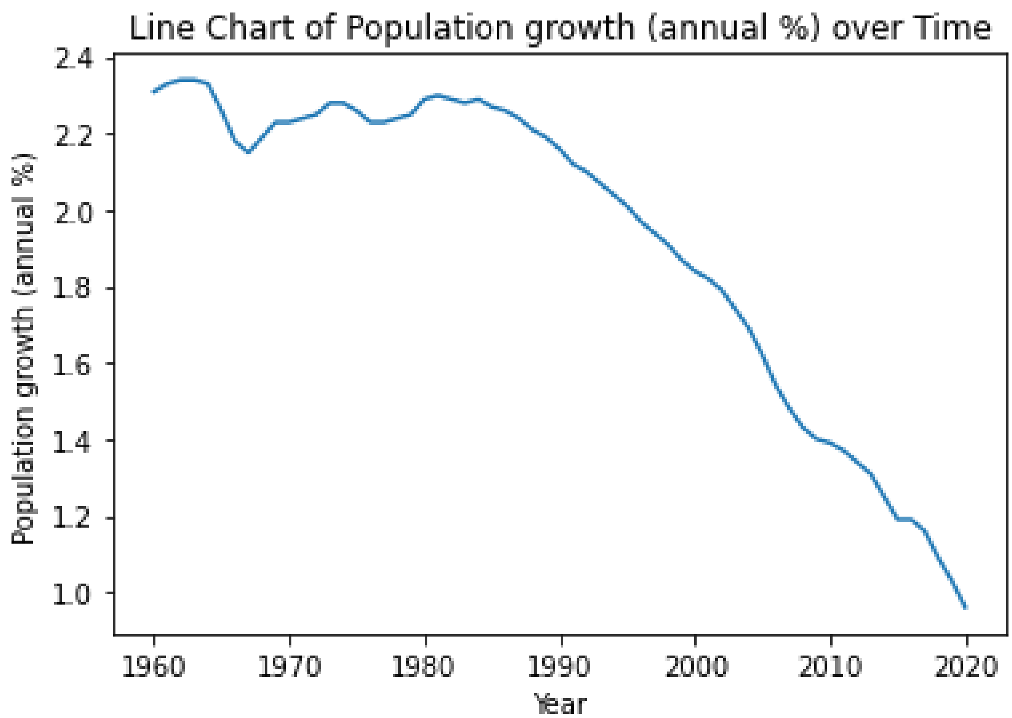

Line Chart of Exports of goods and services (% of GDP) over Time
Exports of goods and services (% of GDP)
25
20
15
10
5
1960
1970
1980
1990
2000
2010
2020
Year


Line Chart of Total reserves (includes gold, current US$) over Time
1e11
Total reserves (includes gold, current US$)
0
1
2
3
4
5
6
1960
1970
1980
1990
2000
2010
2020
Year

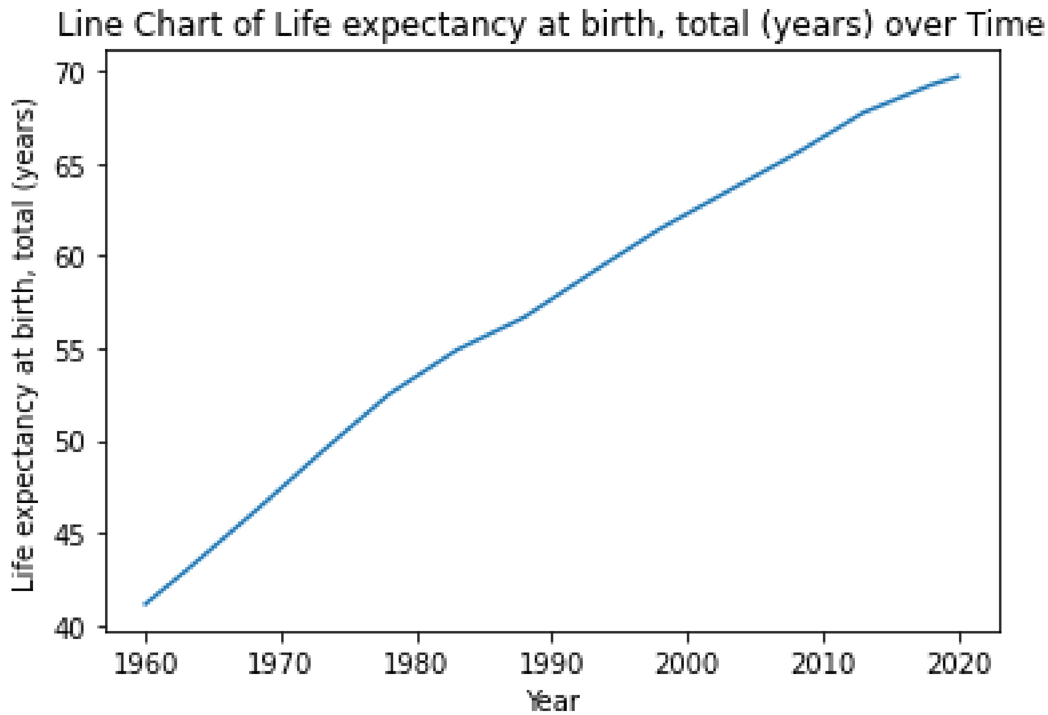

Line Chart of Inflation, consumer prices (annual %) over Time
Inflation, consumer prices (annual %)
30
25
20
15
10
5
0
−5
1960
1970
1980
1990
2000
2010
2020
Year



Line Chart of Population, total over Time
1e9
1.4
1.2
1.0
0.8
0.6
0.4
Population, total
1960
1970
1980
1990
2000
2010
2020
Year



Line Chart of Population growth (annual %) over Time
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
Population growth (annual %)
1960
1970
1980
1990
2000
2010
2020
Year




```
data = data.drop('Country Name', axis=1)

data.corr()

                                                  Year  GDP (current
US$)   \
Year                                          1.000000
0.846589
GDP (current US$)                             0.846589
1.000000
 GDP per capita (current US$)                 0.865053
0.998605
GDP growth (annual %)                         0.278268
0.119174
Imports of goods and services (% of GDP)      0.873956
0.835933
Exports of goods and services (% of GDP)      0.909573
0.847781
 Total reserves (includes gold, current US$)   0.814619
0.980297
Inflation, consumer prices (annual %)        -0.037177            -
0.105585
Population, total                             0.997523
0.863530
Population growth (annual %)                 -0.907750            -
0.957492
```

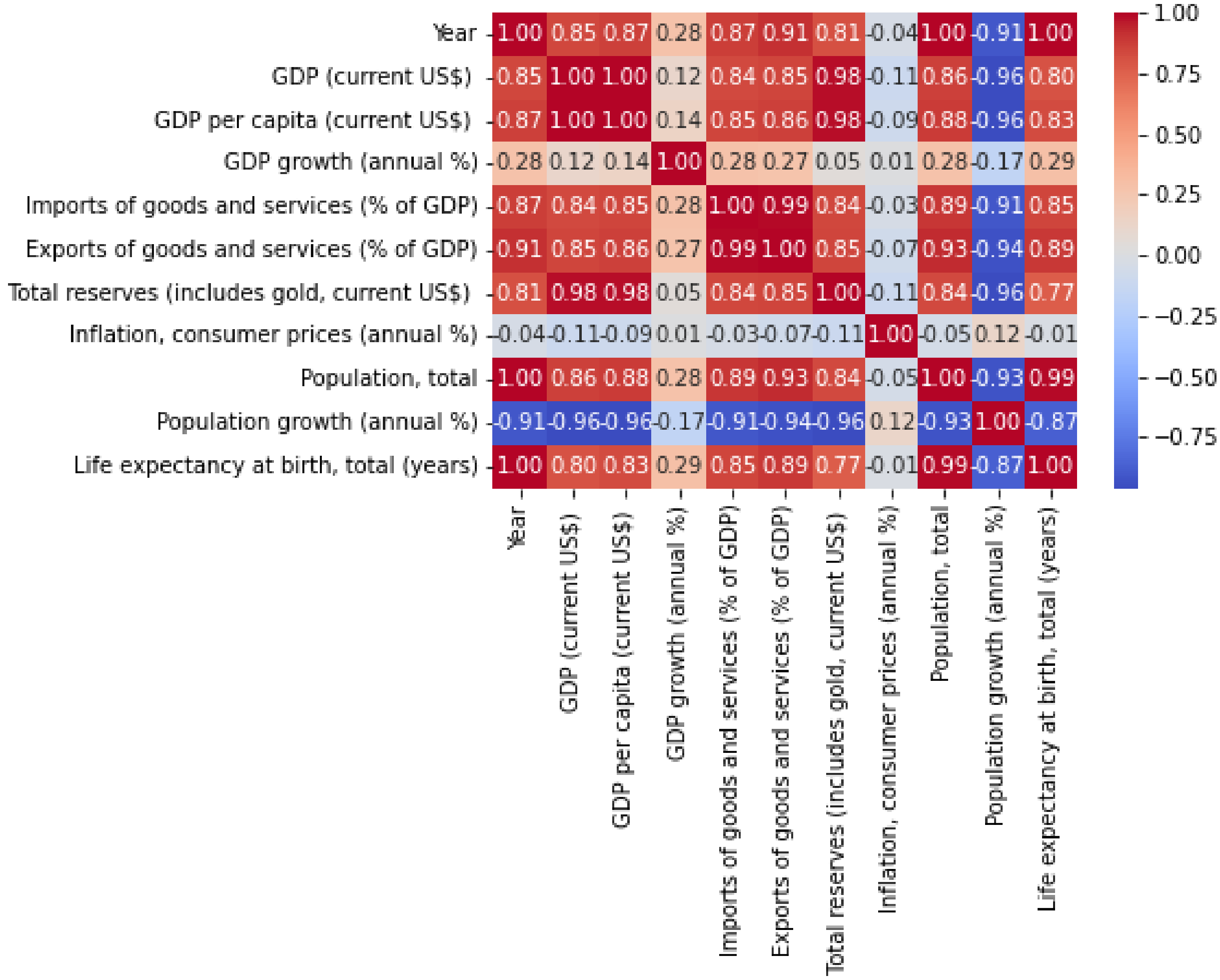
```
Life expectancy at birth, total (years)        0.995487
0.803927

                                                GDP per capita
(current US$)   \
Year
0.865053
GDP (current US$)
0.998605
 GDP per capita (current US$)
1.000000
GDP growth (annual %)
0.142764
Imports of goods and services (% of GDP)
0.853837
Exports of goods and services (% of GDP)
0.863811
 Total reserves (includes gold, current US$)
0.977189
Inflation, consumer prices (annual %)                                 -
0.091981
Population, total
0.880301
Population growth (annual %)                                          -
0.959680
Life expectancy at birth, total (years)
0.825702

                                               GDP growth (annual
%)  \
Year                                                           0.278268

GDP (current US$)                                              0.119174

 GDP per capita (current US$)                                  0.142764

GDP growth (annual %)                                          1.000000

Imports of goods and services (% of GDP)                       0.280289

Exports of goods and services (% of GDP)                       0.269356

 Total reserves (includes gold, current US$)                   0.049946

Inflation, consumer prices (annual %)                          0.007843

Population, total                                              0.276103

Population growth (annual %)                                  -0.168449
```

```
Life expectancy at birth, total (years)                      0.294472


                                               Imports of goods and 
services (% of GDP)  \
Year                                                                
0.873956   
GDP (current US$)                                                   
0.835933   
 GDP per capita (current US$)                                       
0.853837   
GDP growth (annual %)                                               
0.280289   
Imports of goods and services (% of GDP)                            
1.000000   
Exports of goods and services (% of GDP)                            
0.989499   
 Total reserves (includes gold, current US$)                        
0.841084   
Inflation, consumer prices (annual %)                               
-0.034099   
Population, total                                                   
0.894541   
Population growth (annual %)                                        
-0.912249   
Life expectancy at birth, total (years)                             
0.849597   

                                               Exports of goods and 
services (% of GDP)  \
Year                                                                
0.909573   
GDP (current US$)                                                   
0.847781   
 GDP per capita (current US$)                                       
0.863811   
GDP growth (annual %)                                               
0.269356   
Imports of goods and services (% of GDP)                            
0.989499   
Exports of goods and services (% of GDP)                            
1.000000   
 Total reserves (includes gold, current US$)                        
0.851635   
Inflation, consumer prices (annual %)                               
-0.073604   
Population, total                                                   
0.927934   
Population growth (annual %)                                        
-0.935063   
```

```
Life expectancy at birth, total (years)
0.886921

                                                Total reserves
(includes gold, current US$)   \
Year
0.814619
GDP (current US$)
0.980297
 GDP per capita (current US$)
0.977189
GDP growth (annual %)
0.049946
Imports of goods and services (% of GDP)
0.841084
Exports of goods and services (% of GDP)
0.851635
 Total reserves (includes gold, current US$)
1.000000
Inflation, consumer prices (annual %)
-0.107925
Population, total
0.835249
Population growth (annual %)
-0.957013
Life expectancy at birth, total (years)
0.767909

                                               Inflation, consumer
prices (annual %)  \
Year
-0.037177
GDP (current US$)
-0.105585
 GDP per capita (current US$)
-0.091981
GDP growth (annual %)
0.007843
Imports of goods and services (% of GDP)
-0.034099
Exports of goods and services (% of GDP)
-0.073604
 Total reserves (includes gold, current US$)
-0.107925
Inflation, consumer prices (annual %)
1.000000
Population, total
-0.053939
Population growth (annual %)
```

```
0.123497
Life expectancy at birth, total (years)
-0.014927

                                               Population, total  \
Year                                                    0.997523
GDP (current US$)                                       0.863530
 GDP per capita (current US$)                           0.880301
GDP growth (annual %)                                   0.276103
Imports of goods and services (% of GDP)                0.894541
Exports of goods and services (% of GDP)                0.927934
 Total reserves (includes gold, current US$)            0.835249
Inflation, consumer prices (annual %)                  -0.053939
Population, total                                       1.000000
Population growth (annual %)                           -0.928431
Life expectancy at birth, total (years)                 0.987134

                                               Population growth
(annual %)  \
Year                                                               -
0.907750
GDP (current US$)                                                  -
0.957492
 GDP per capita (current US$)                                      -
0.959680
GDP growth (annual %)                                              -
0.168449
Imports of goods and services (% of GDP)                           -
0.912249
Exports of goods and services (% of GDP)                           -
0.935063
 Total reserves (includes gold, current US$)                       -
0.957013
Inflation, consumer prices (annual %)
0.123497
Population, total                                                  -
0.928431
Population growth (annual %)
1.000000
Life expectancy at birth, total (years)                            -
0.868766

                                               Life expectancy at
birth, total (years)
Year
0.995487
GDP (current US$)
0.803927
 GDP per capita (current US$)
0.825702
```

```
GDP growth (annual %)
0.294472
Imports of goods and services (% of GDP)
0.849597
Exports of goods and services (% of GDP)
0.886921
 Total reserves (includes gold, current US$)
0.767909
Inflation, consumer prices (annual %)
-0.014927
Population, total
0.987134
Population growth (annual %)
-0.868766
Life expectancy at birth, total (years)
1.000000

import seaborn as sns

# Create the heatmap
sns.heatmap(data.corr(), annot=True, cmap="coolwarm", fmt=".2f",
            xticklabels=data.columns, yticklabels=data.columns)

# Set large figure size
plt.figure(figsize=(20, 20))

# Show the heatmap
plt.show()
```

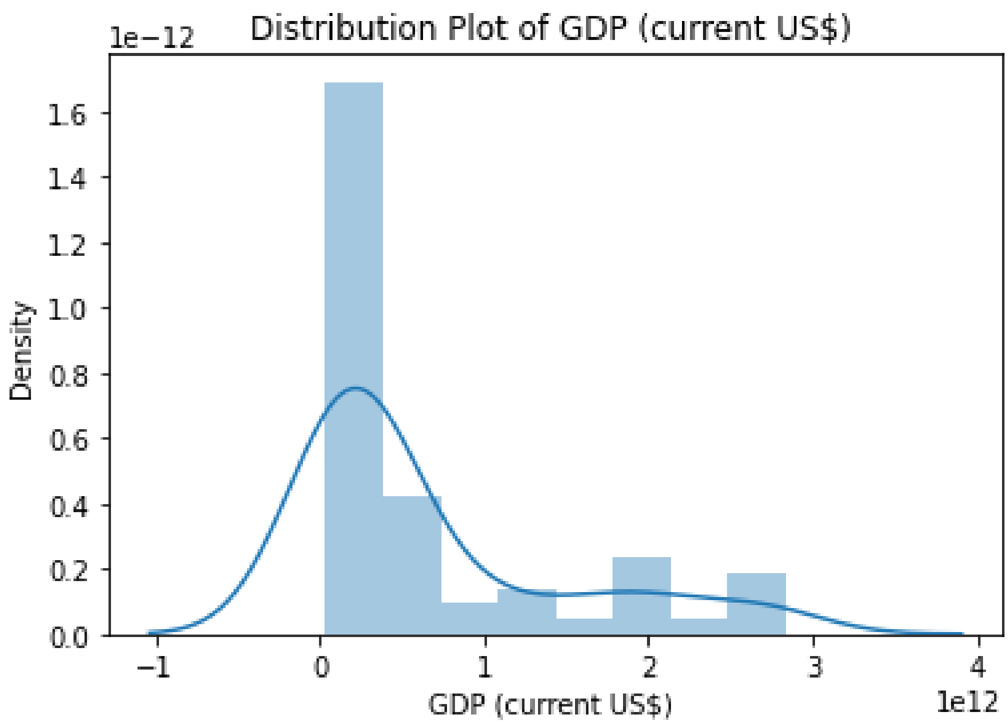



```
<Figure size 1440x1440 with 0 Axes>
```

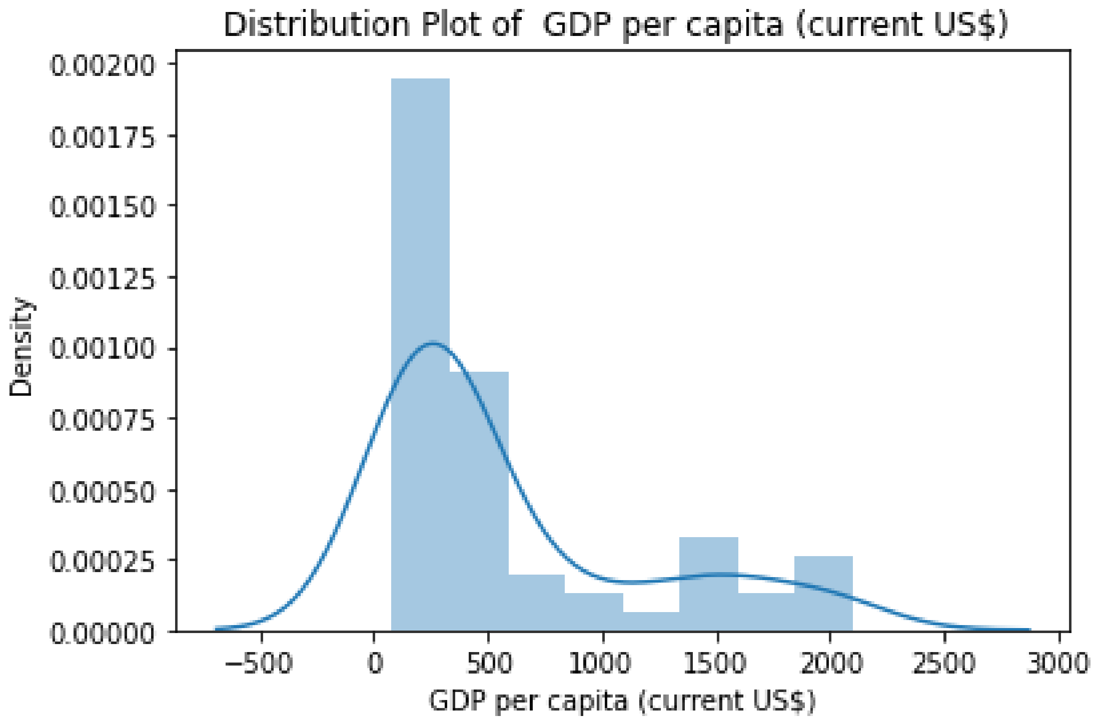
```python
# Iterate through each column except 'Year'
for col in data.columns:
  if col not in ('Year'):
    plt.figure()
    sns.distplot(data[col])
    plt.title(f'Distribution Plot of {col}')
    plt.show()
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_6448\4234885766.py:5:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
```

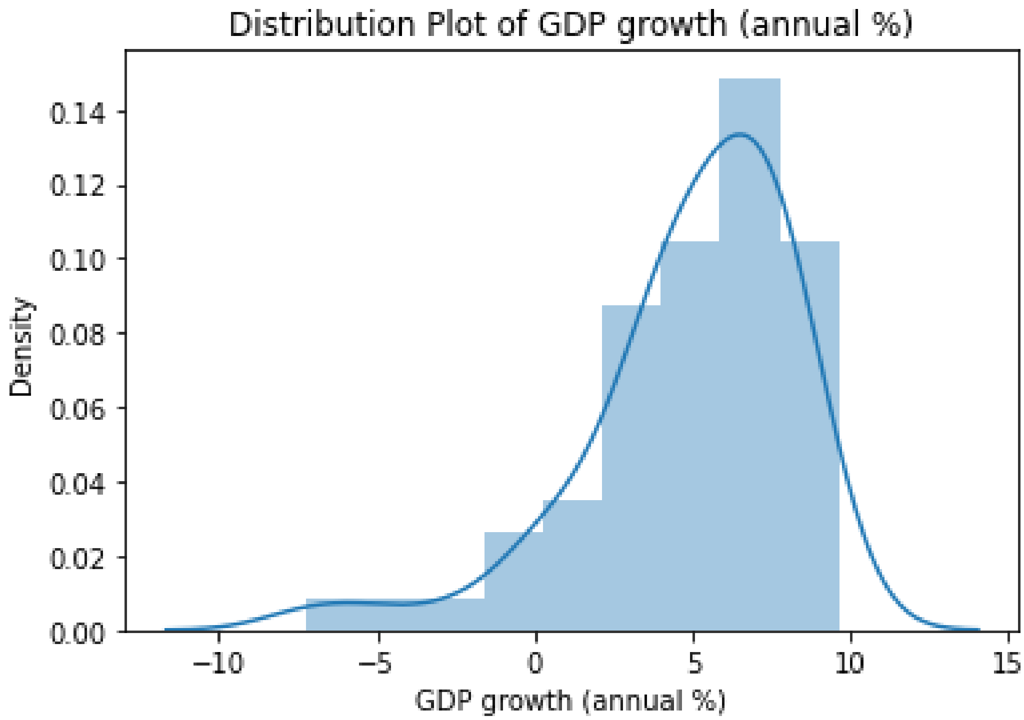
```
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data[col])
```



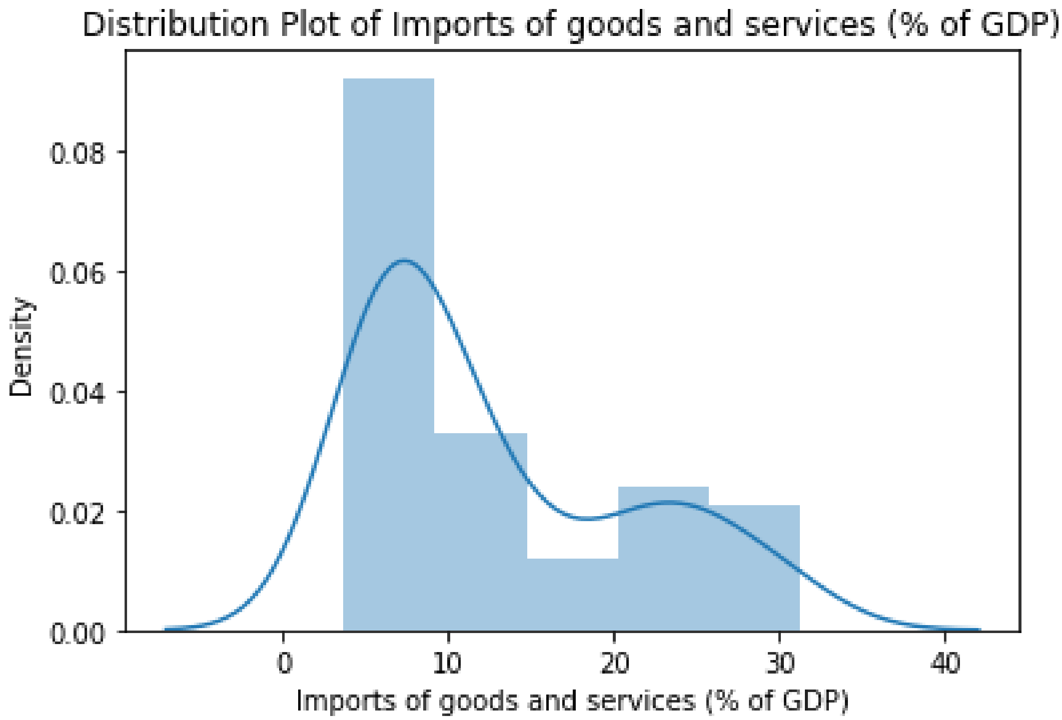

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_6448\4234885766.py:5:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data[col])
```




```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_6448\4234885766.py:5:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data[col])
```

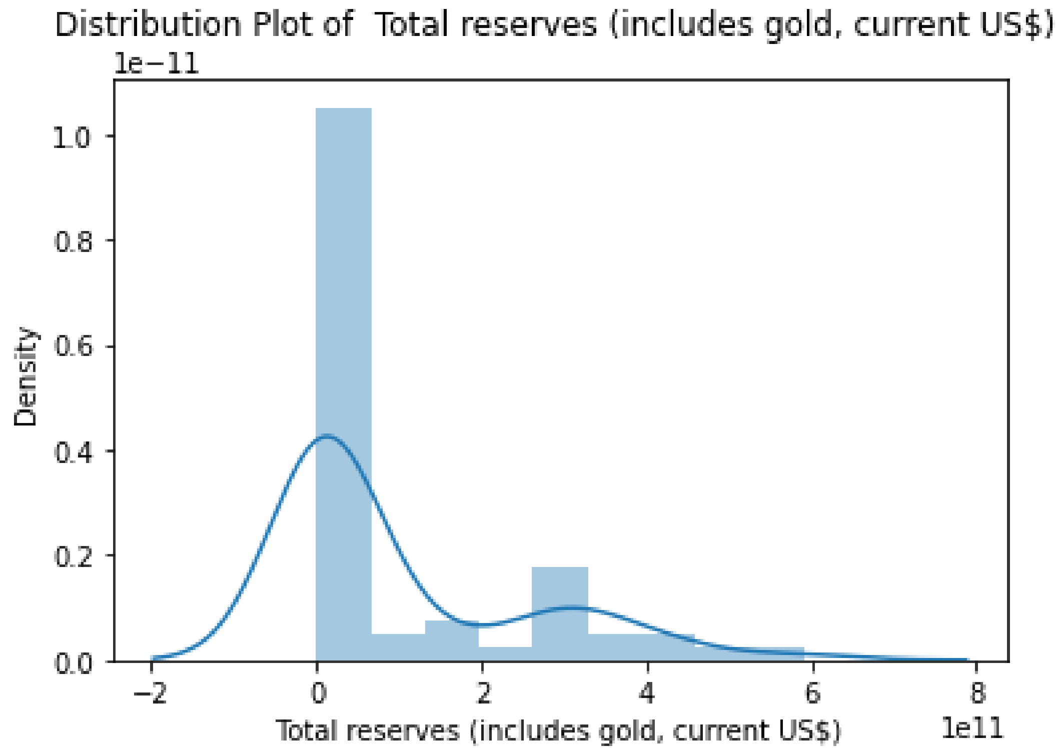



```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_6448\4234885766.py:5:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data[col])
```

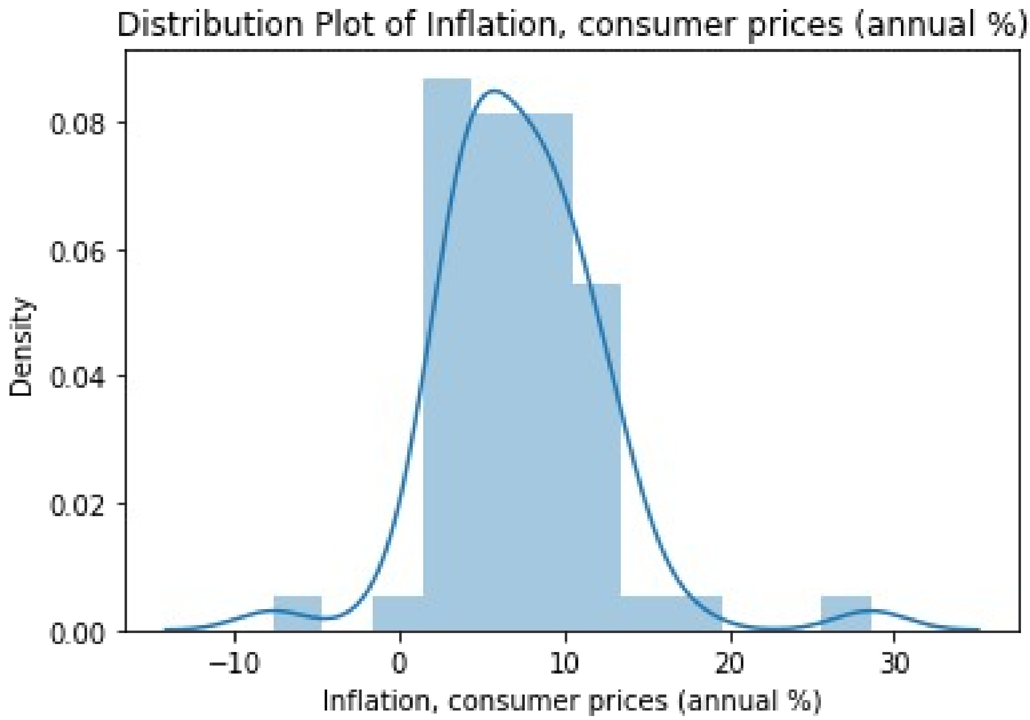



```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_6448\4234885766.py:5:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data[col])
```

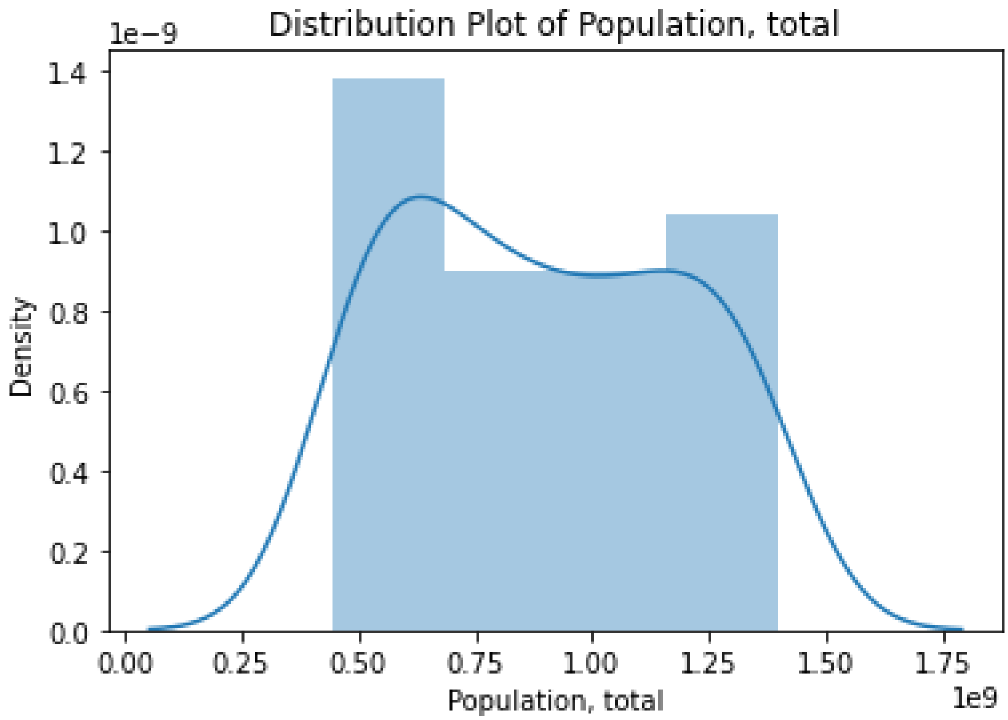

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_6448\4234885766.py:5:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data[col])
```



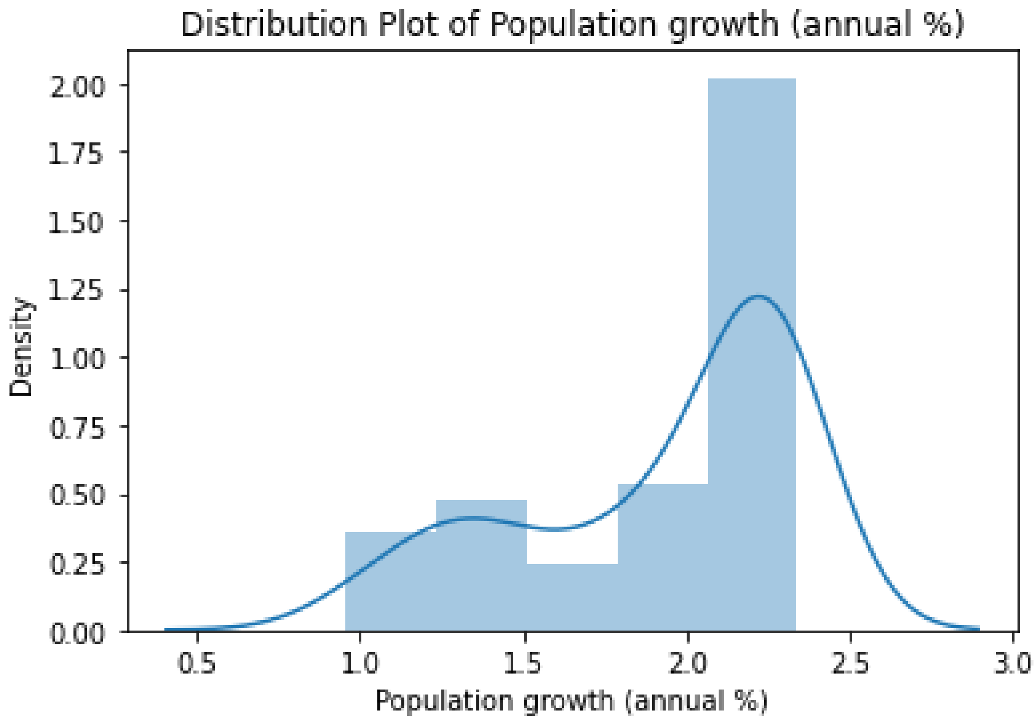

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_6448\4234885766.py:5:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data[col])
```



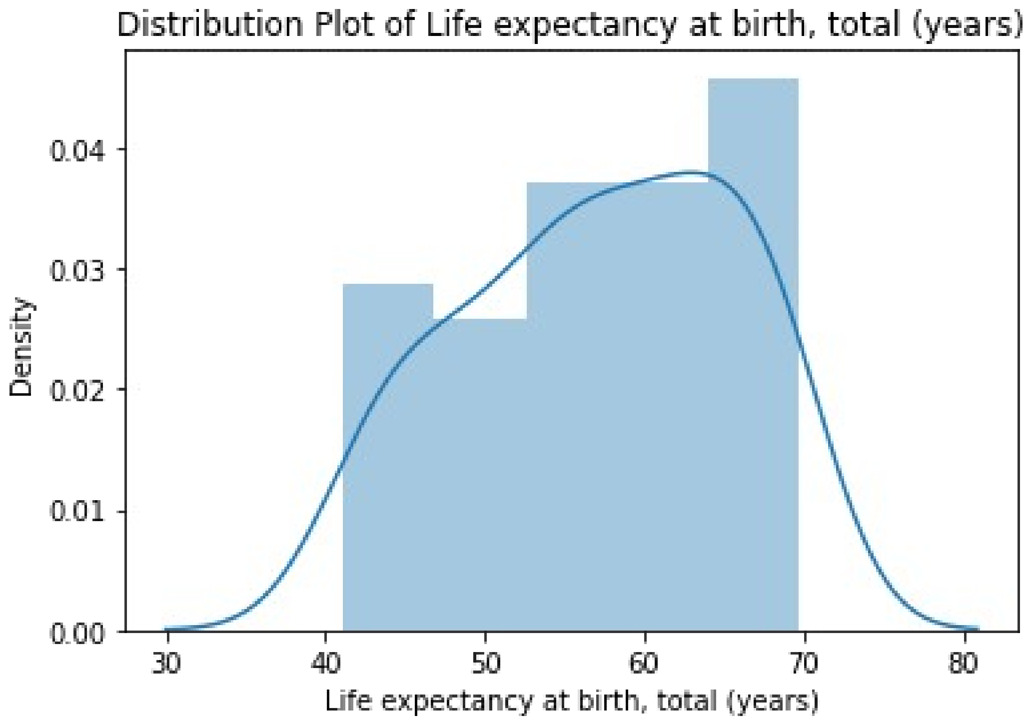

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_6448\4234885766.py:5:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data[col])
```



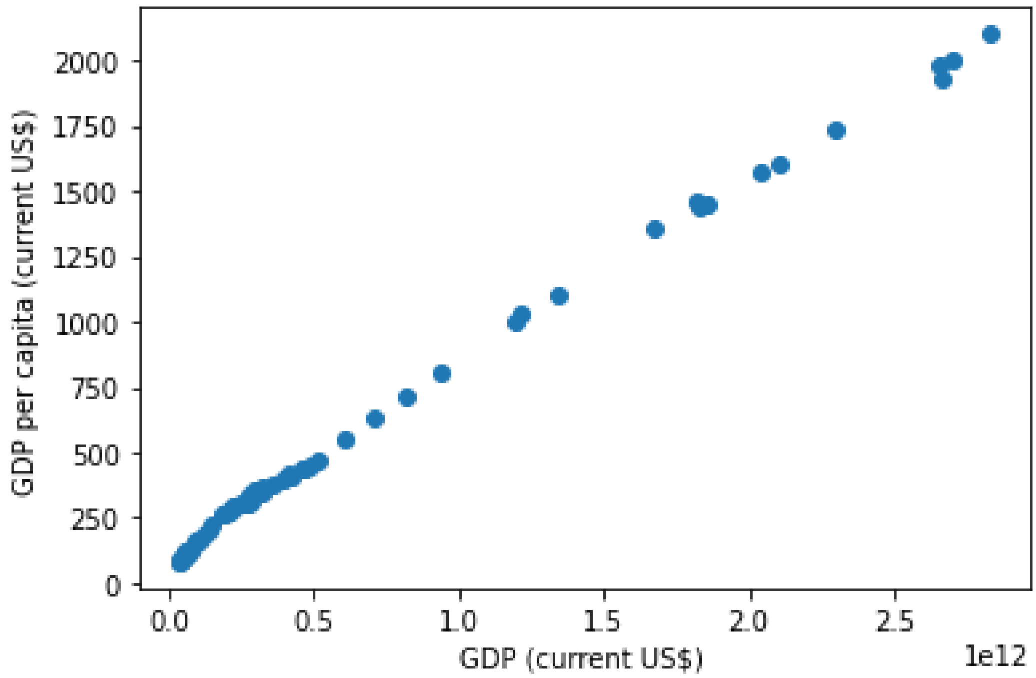

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_6448\4234885766.py:5:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data[col])
```

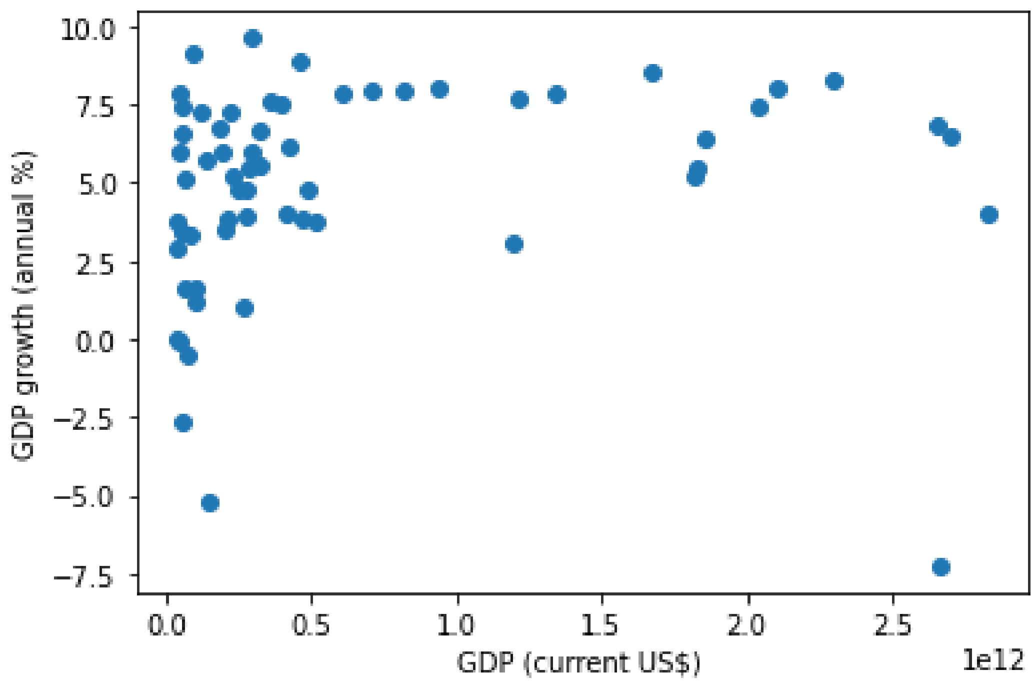


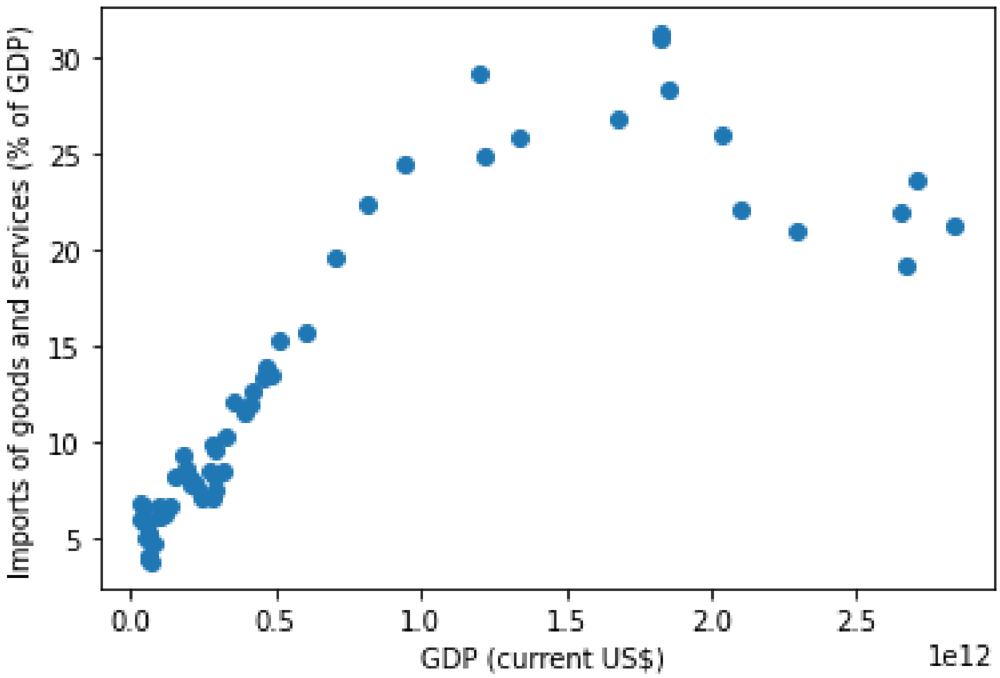

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_6448\4234885766.py:5:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data[col])
```

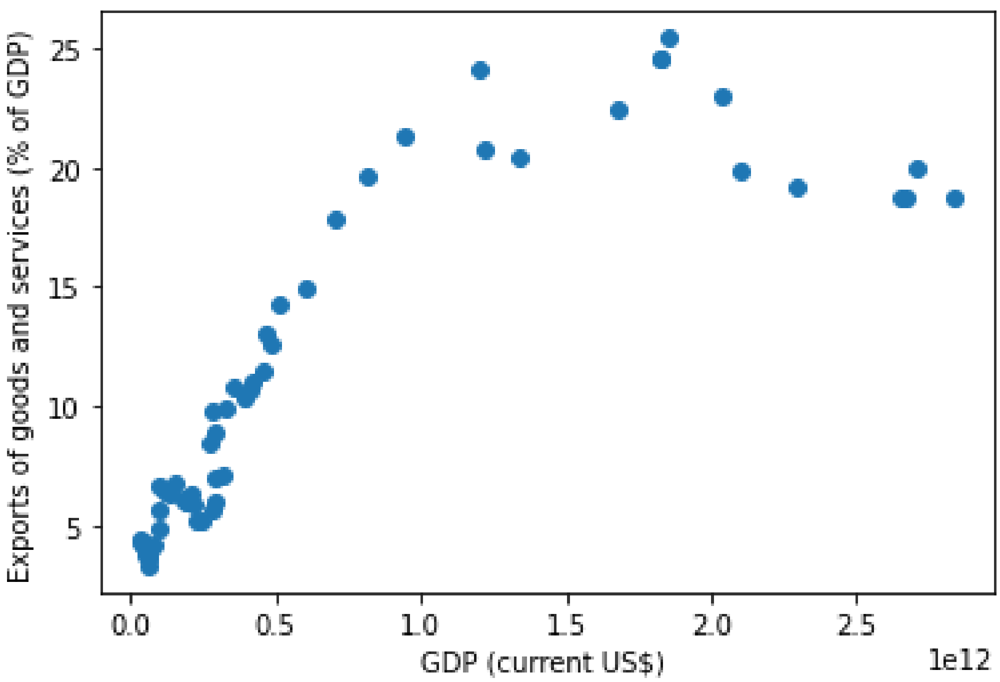


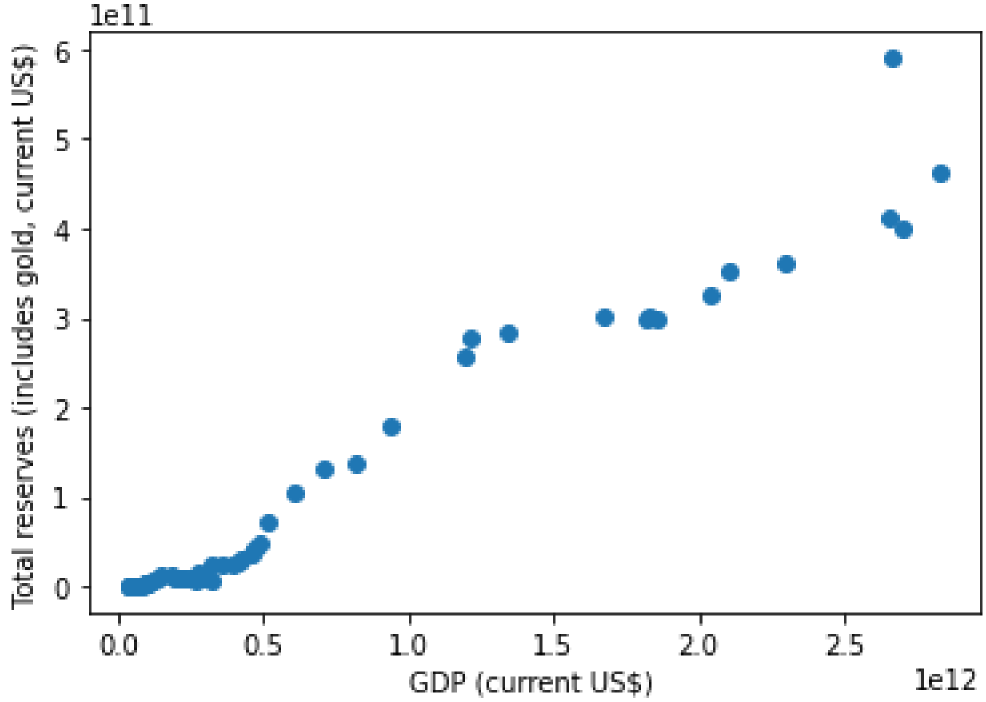

```
# prompt: make scatter plot with each columns without using pairplot

import matplotlib.pyplot as plt
for i in range(1, len(data.columns)):
  for j in range(i + 1, len(data.columns)):
    plt.scatter(data.iloc[:, i], data.iloc[:, j])
    plt.xlabel(data.columns[i])
    plt.ylabel(data.columns[j])
    plt.show()
```

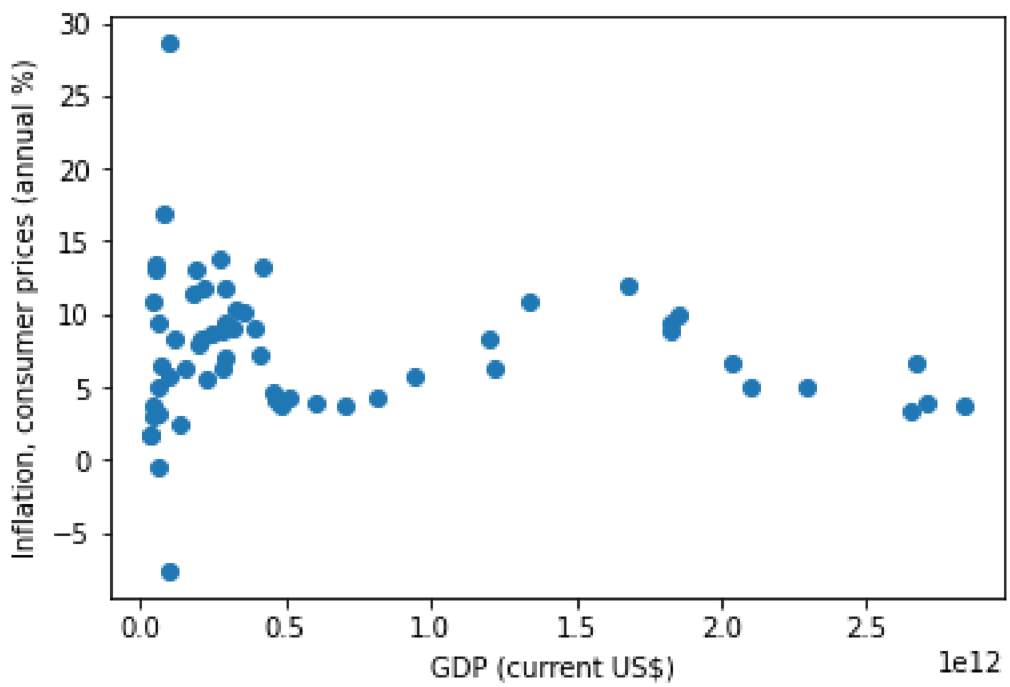

GDP per capita (current US$)
0
250
500
750
1000
1250
1500
1750
2000
0.0
0.5
1.0
1.5
2.0
2.5
GDP (current US$)
1e12

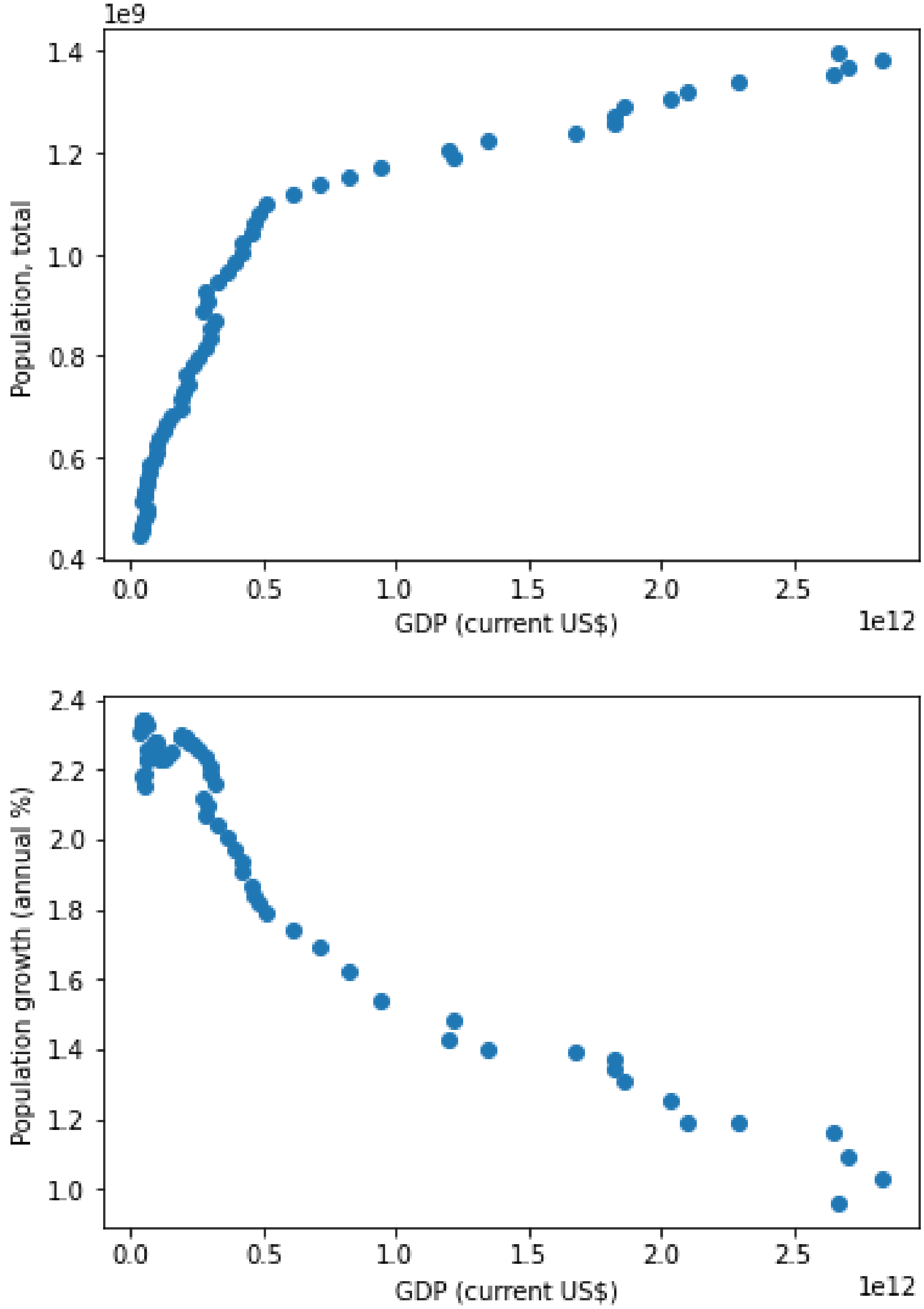

GDP growth (annual %)
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0.0
0.5
1.0
1.5
2.0
2.5
GDP (current US$)
1e12

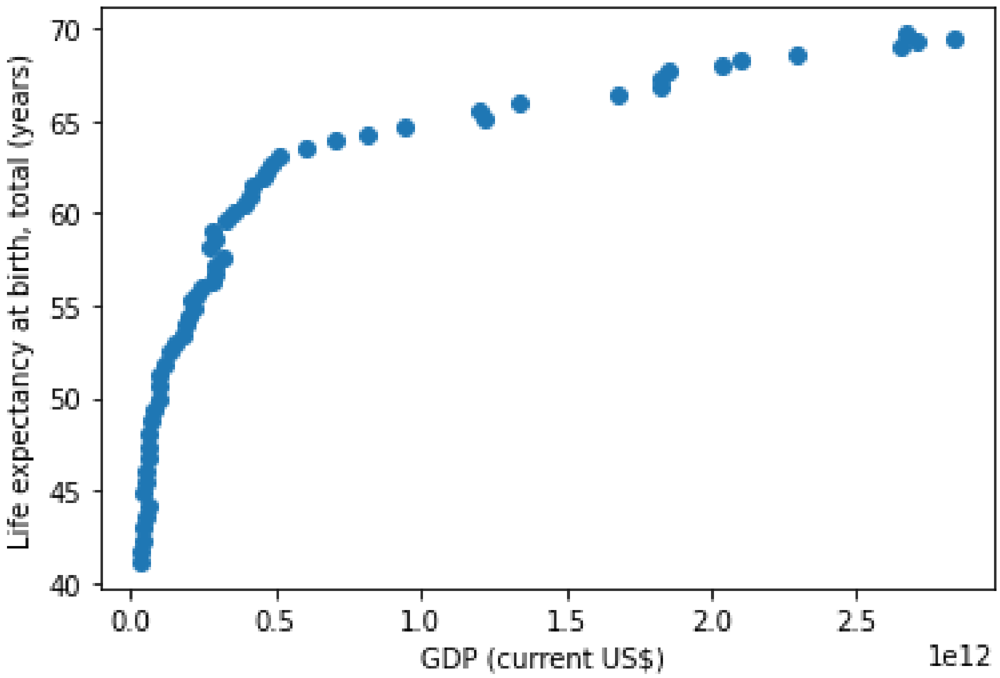

Imports of goods and services (% of GDP)
30
25
20
15
10
5
0.0
0.5
1.0
1.5
2.0
2.5
1e12
GDP (current US$)


Exports of goods and services (% of GDP)
25
20
15
10
5
0.0
0.5
1.0
1.5
2.0
2.5
1e12
GDP (current US$)

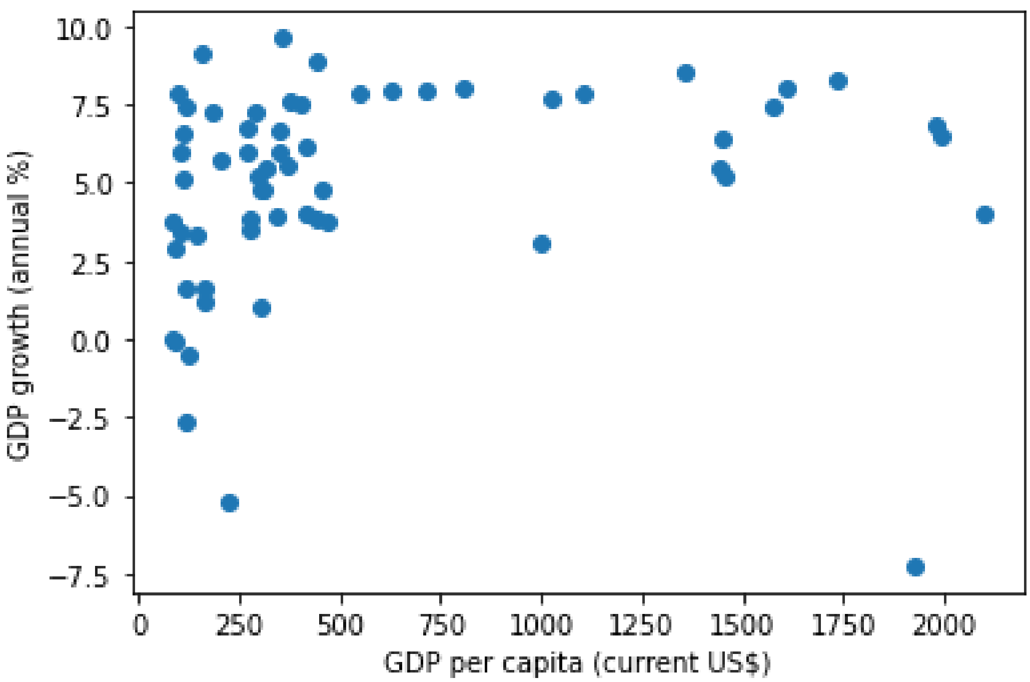

1e11
Total reserves (includes gold, current US$)
GDP (current US$)
1e12


Inflation, consumer prices (annual %)
GDP (current US$)
1e12

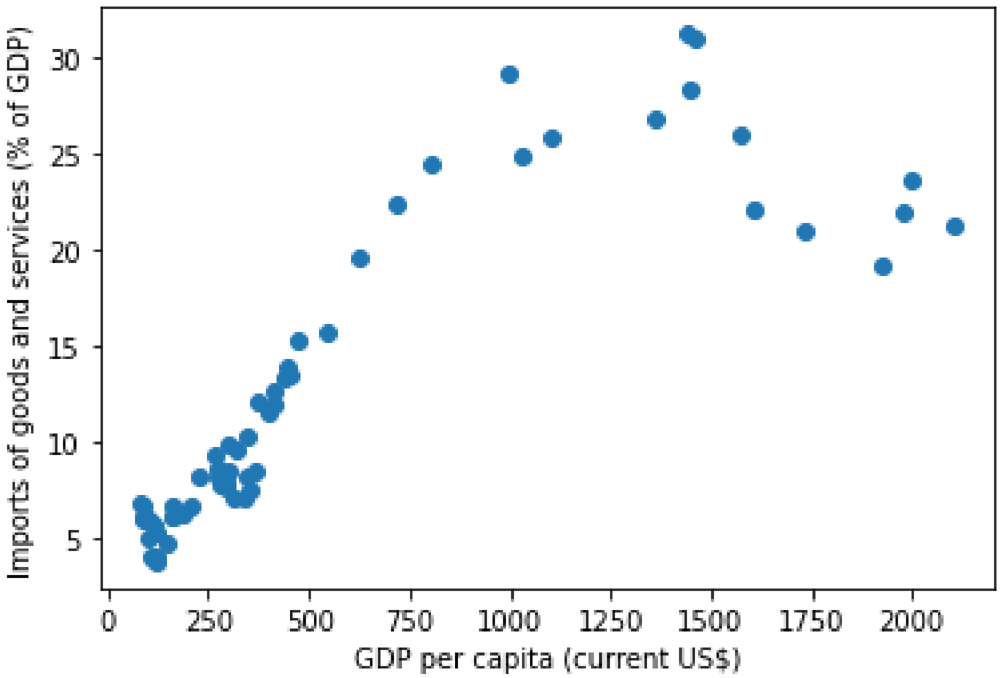

1e9
1.4
1.2
1.0
0.8
0.6
0.4
Population, total
0.0
0.5
1.0
1.5
2.0
2.5
GDP (current US$)
1e12
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
Population growth (annual %)
0.0
0.5
1.0
1.5
2.0
2.5
GDP (current US$)
1e12

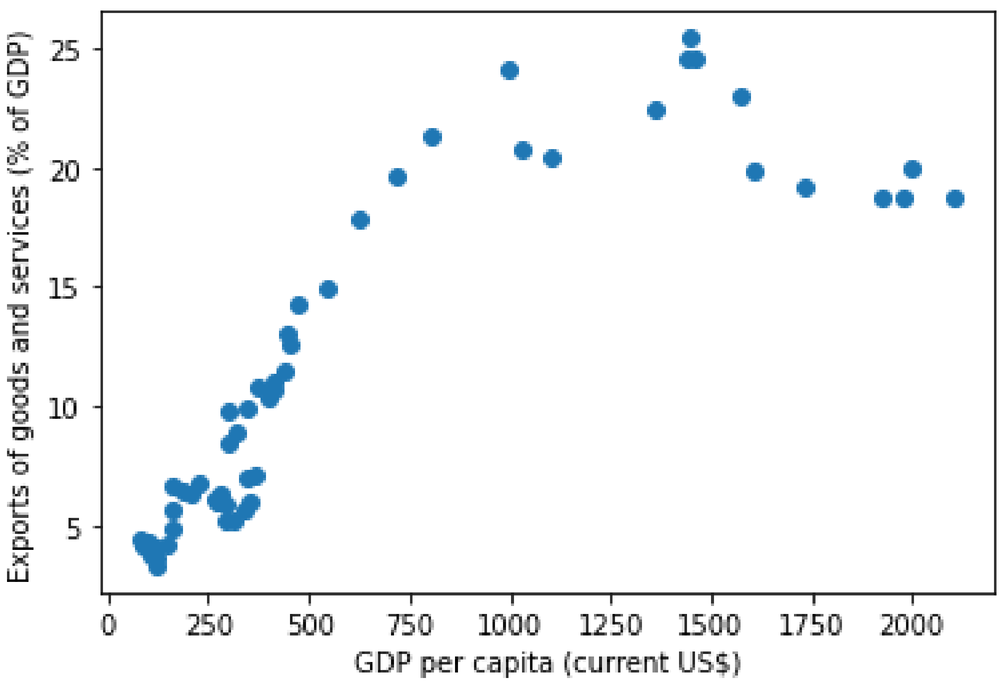


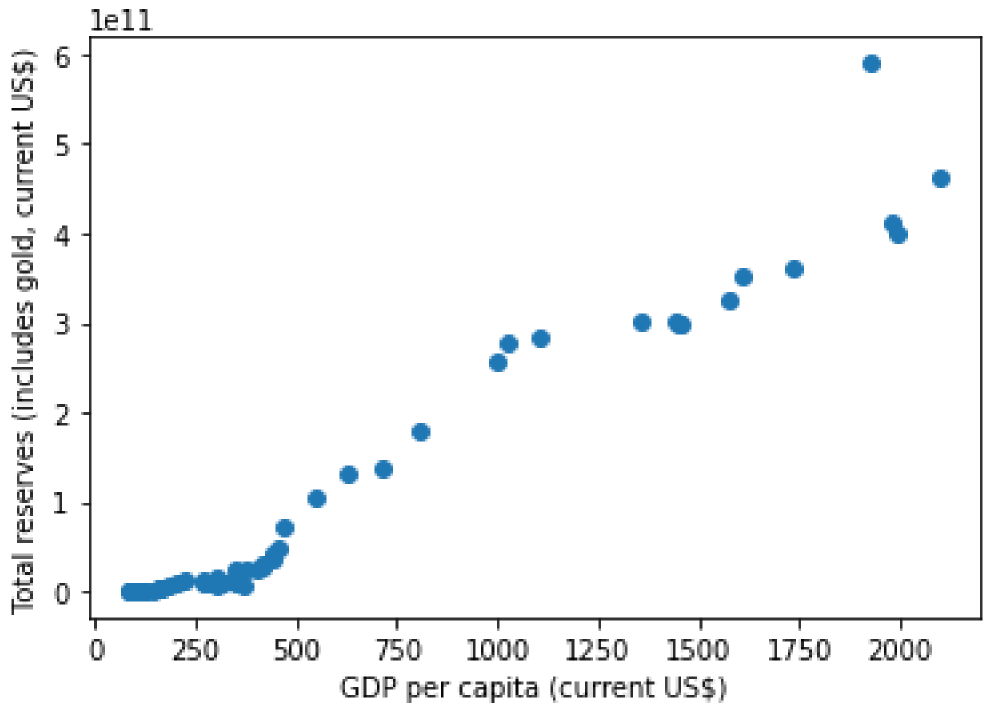
Life expectancy at birth, total (years)
70
65
60
55
50
45
40
0.0
0.5
1.0
1.5
2.0
2.5
1e12
GDP (current US$)

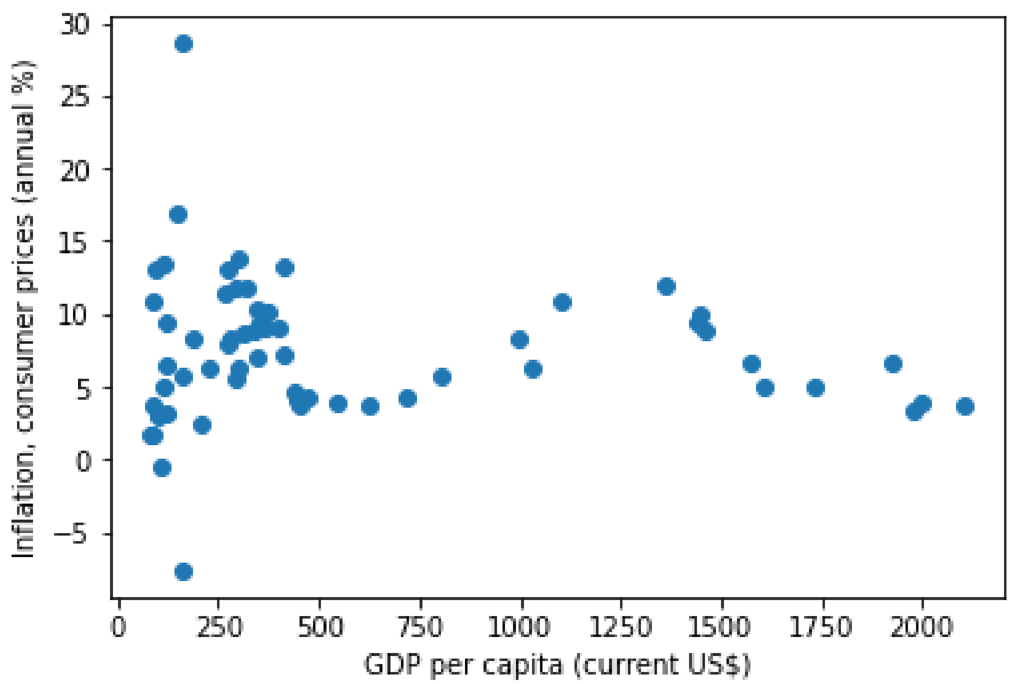


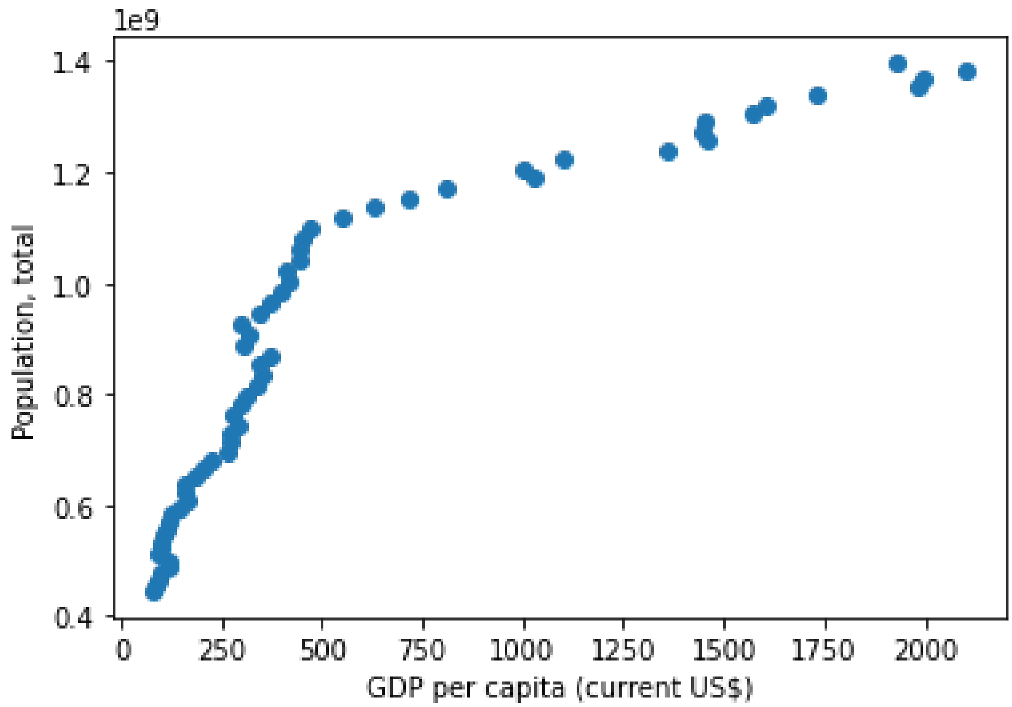
GDP growth (annual %)
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0
250
500
750
1000
1250
1500
1750
2000
GDP per capita (current US$)

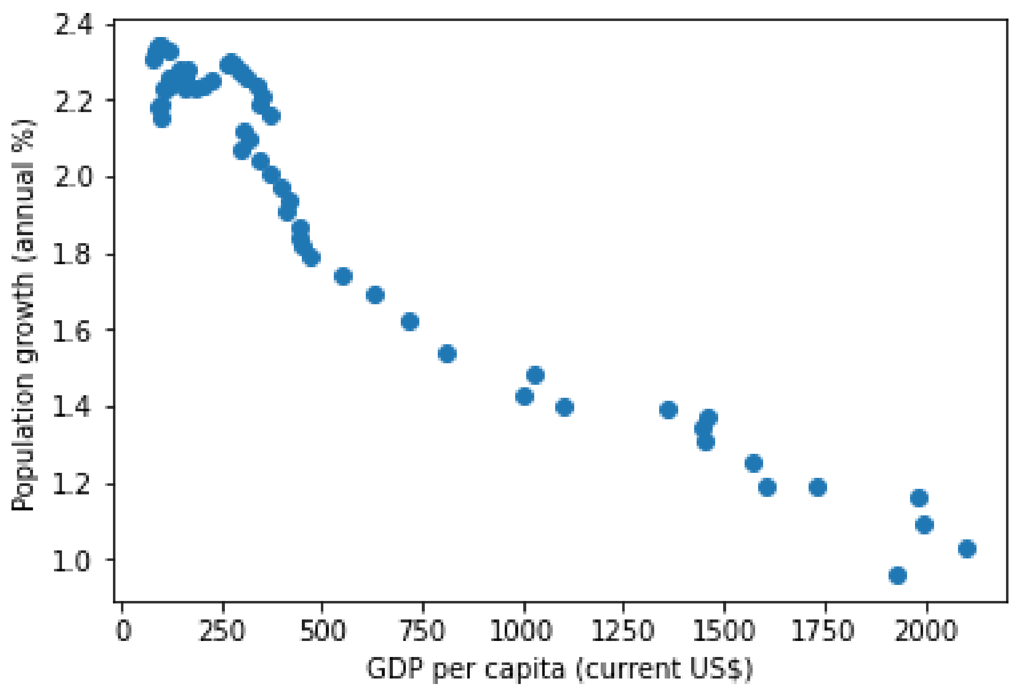

Imports of goods and services (% of GDP)
5
10
15
20
25
30
0
250
500
750
1000
1250
1500
1750
2000
GDP per capita (current US$)

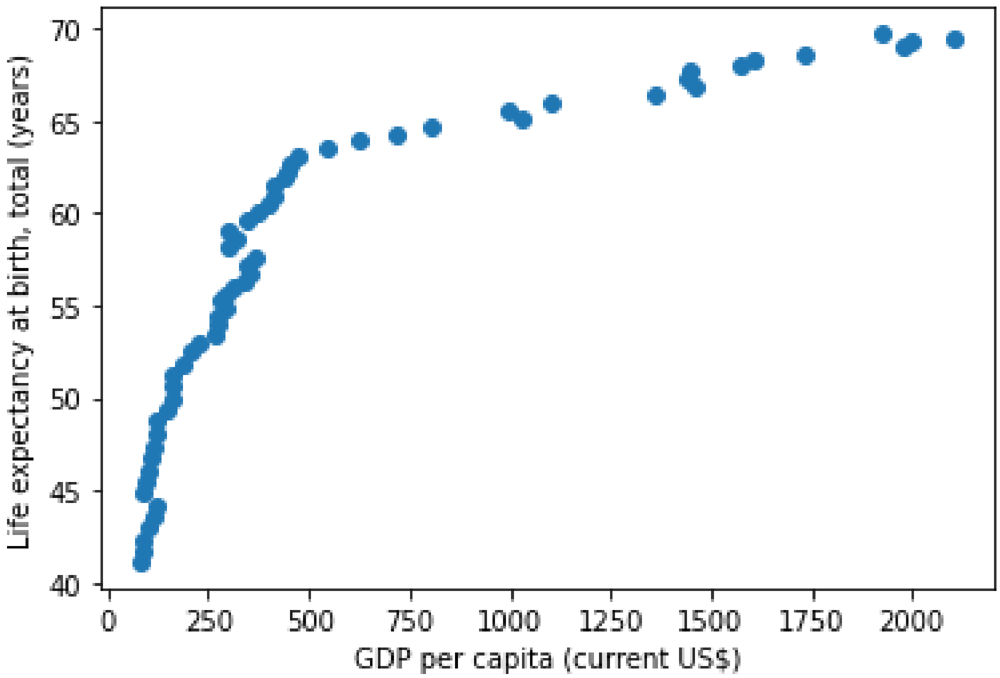

Exports of goods and services (% of GDP)
5
10
15
20
25
0
250
500
750
1000
1250
1500
1750
2000
GDP per capita (current US$)


1e11
Total reserves (includes gold, current US$)
GDP per capita (current US$)

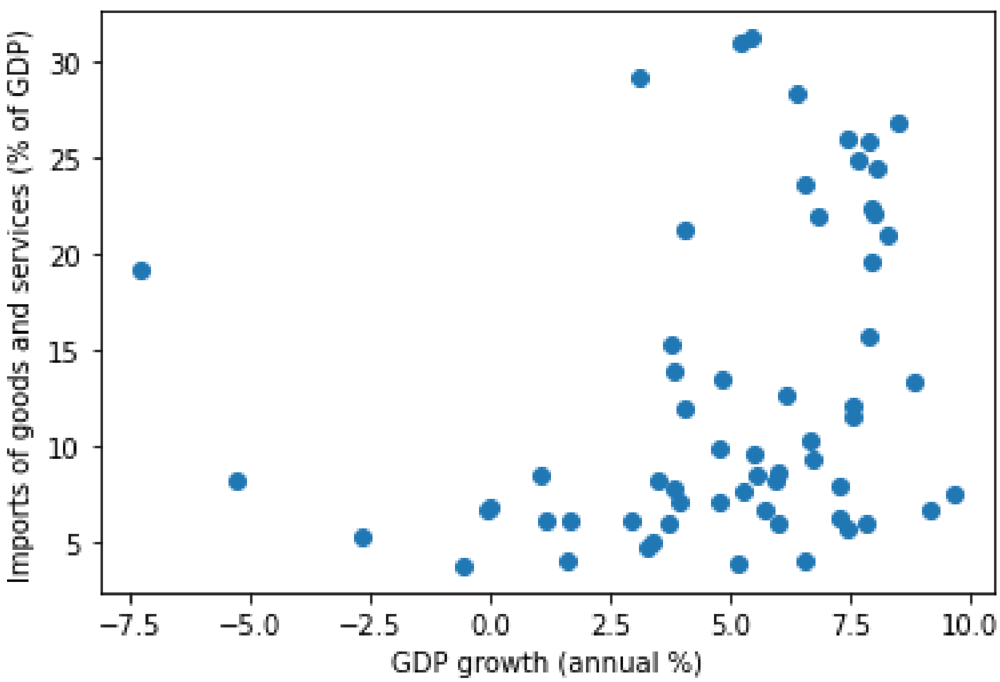

Inflation, consumer prices (annual %)
GDP per capita (current US$)


1e9
1.4
1.2
1.0
0.8
0.6
0.4
Population, total
0
250
500
750
1000
1250
1500
1750
2000
GDP per capita (current US$)

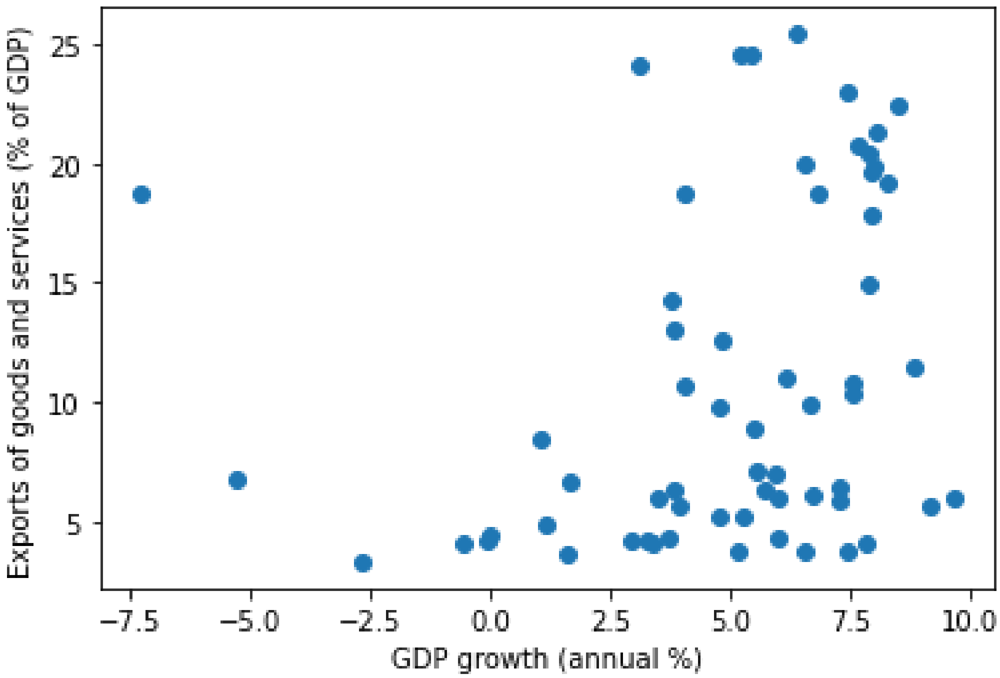

2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
Population growth (annual %)
0
250
500
750
1000
1250
1500
1750
2000
GDP per capita (current US$)

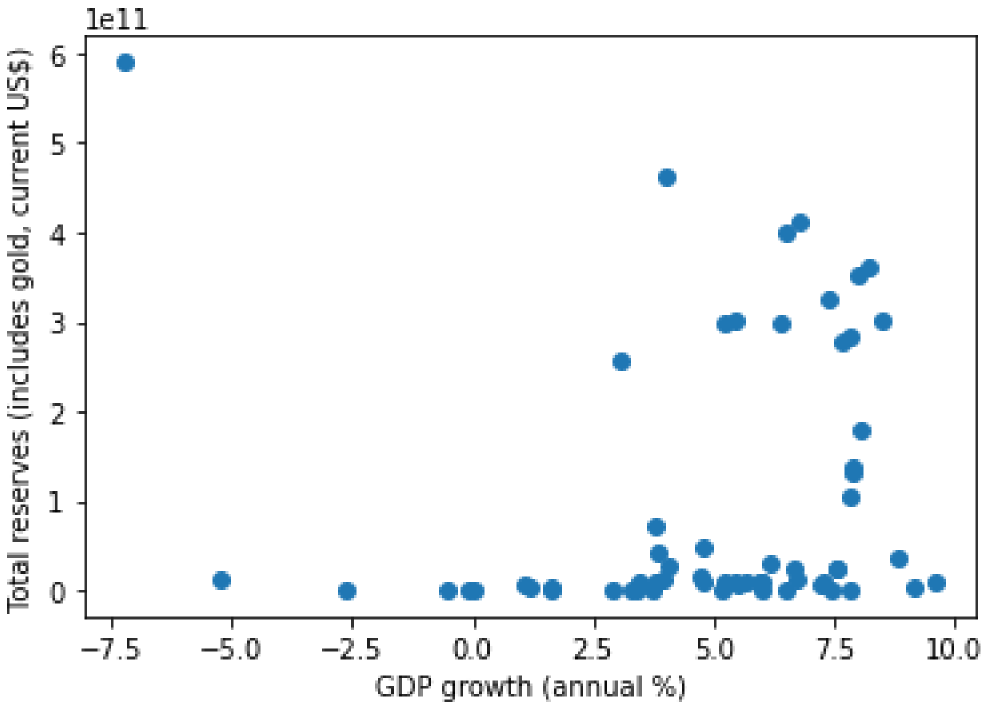


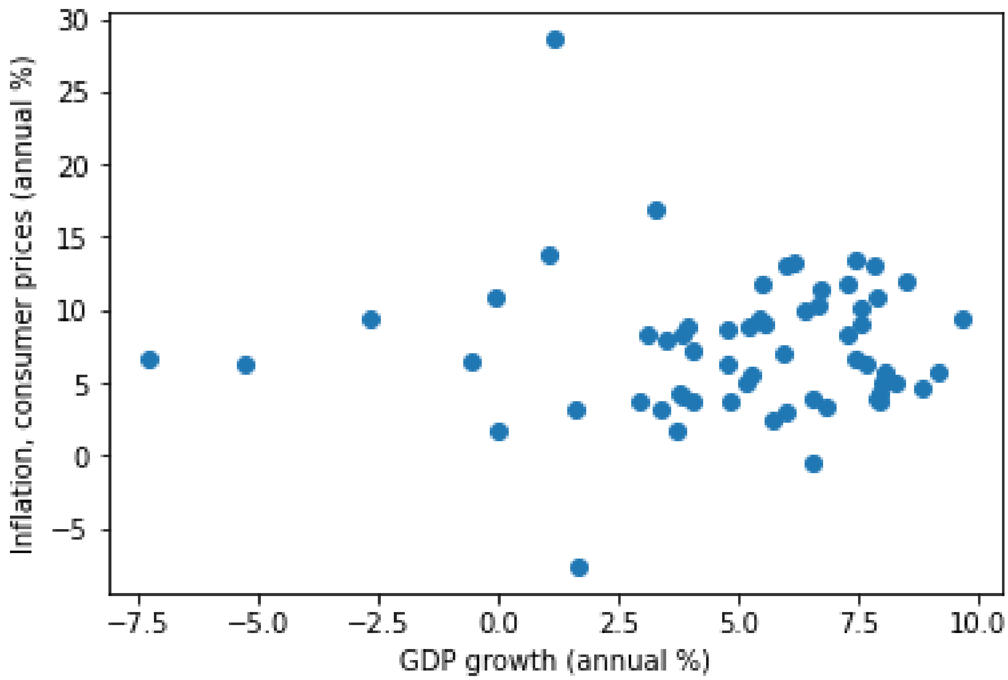
Life expectancy at birth, total (years)
70
65
60
55
50
45
40
0
250
500
750
1000
1250
1500
1750
2000
GDP per capita (current US$)



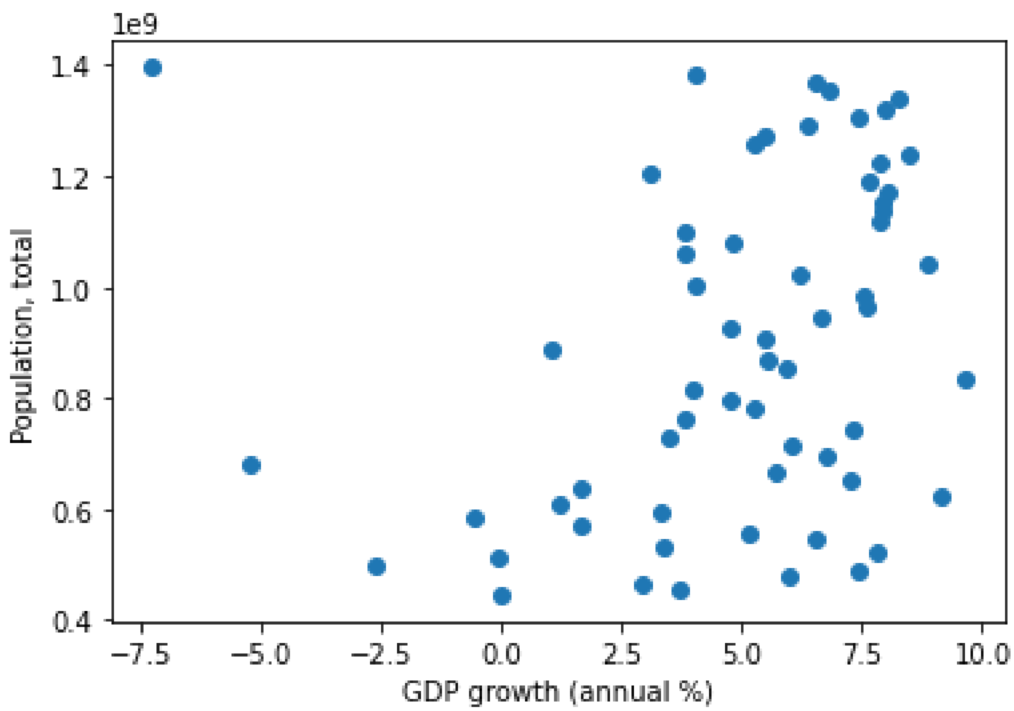
Imports of goods and services (% of GDP)
30
25
20
15
10
5
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
GDP growth (annual %)


Exports of goods and services (% of GDP)
25
20
15
10
5
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
GDP growth (annual %)

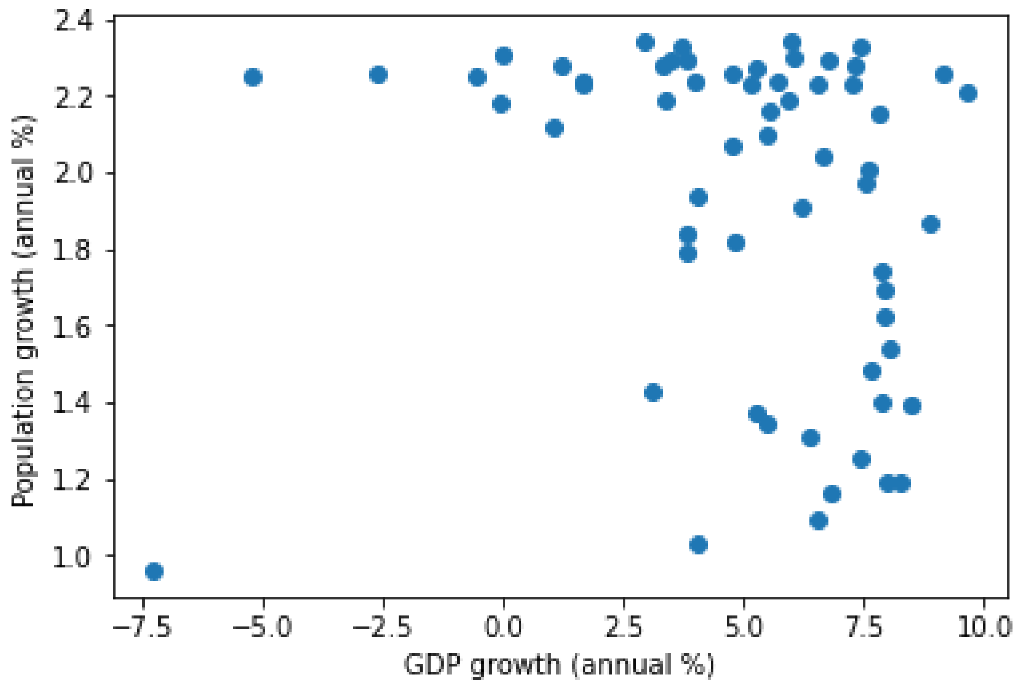

1e11
Total reserves (includes gold, current US$)
6
5
4
3
2
1
0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
GDP growth (annual %)



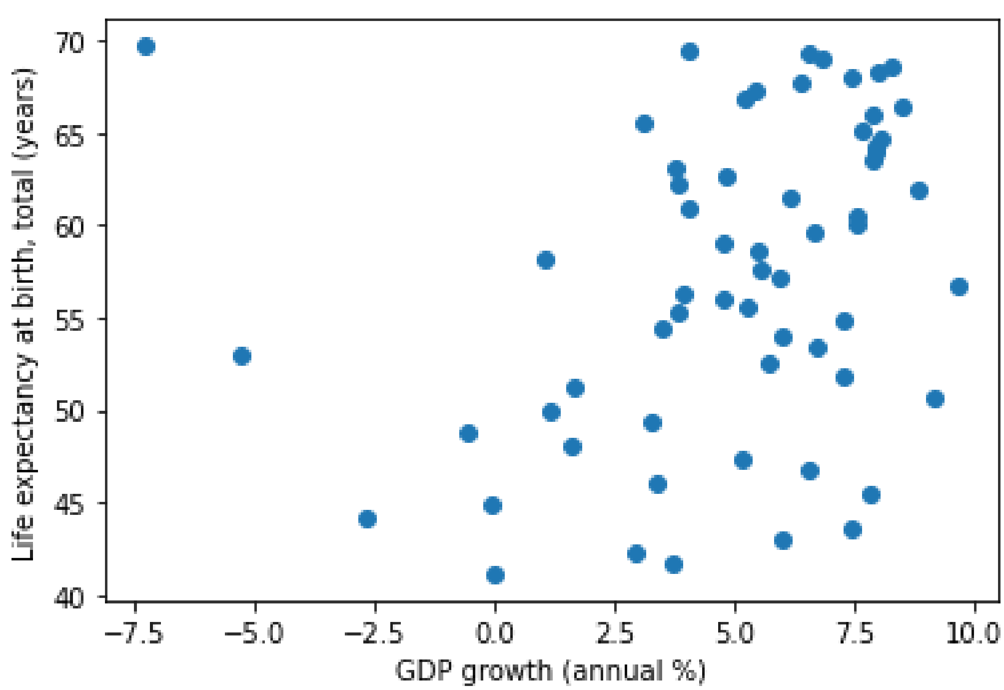
30
25
20
15
10
5
0
−5
Inflation, consumer prices (annual %)
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
GDP growth (annual %)



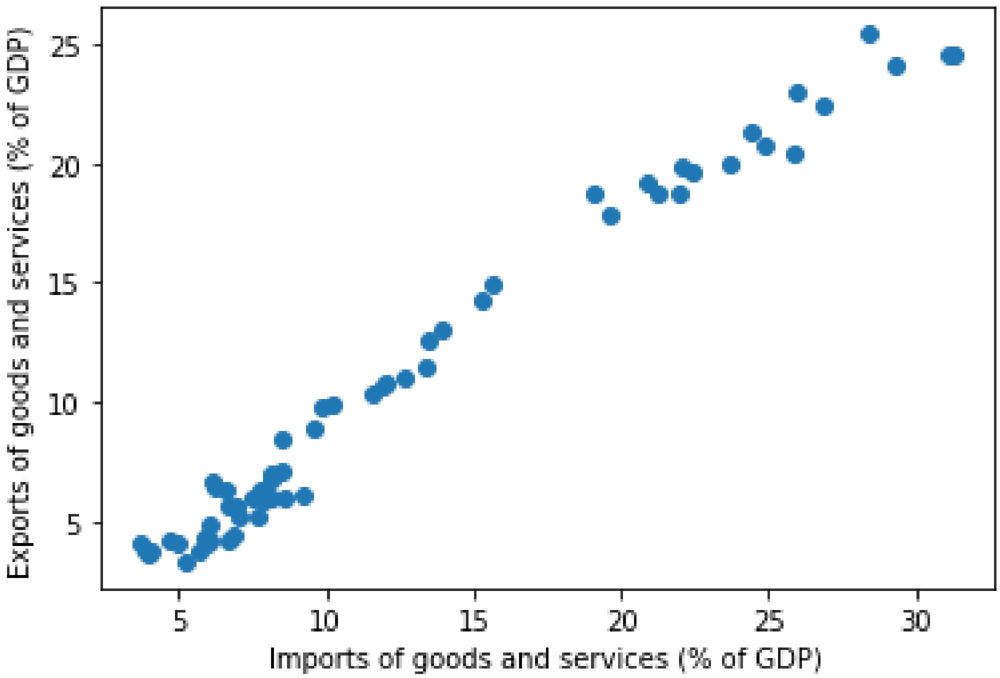
1e9
1.4
1.2
1.0
0.8
0.6
0.4
Population, total
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
GDP growth (annual %)

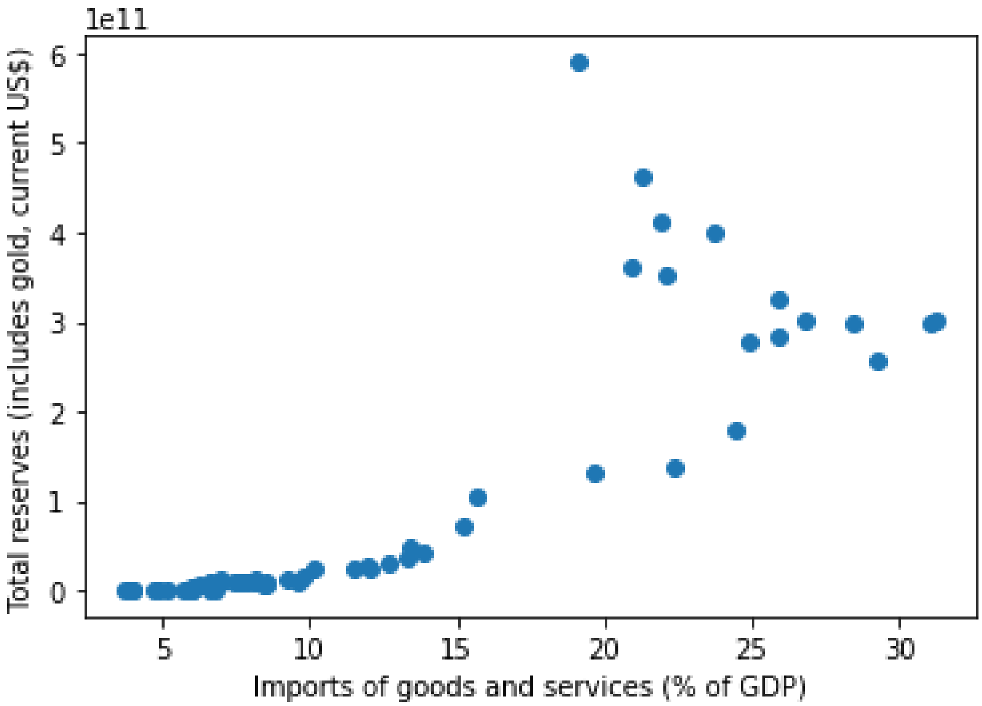


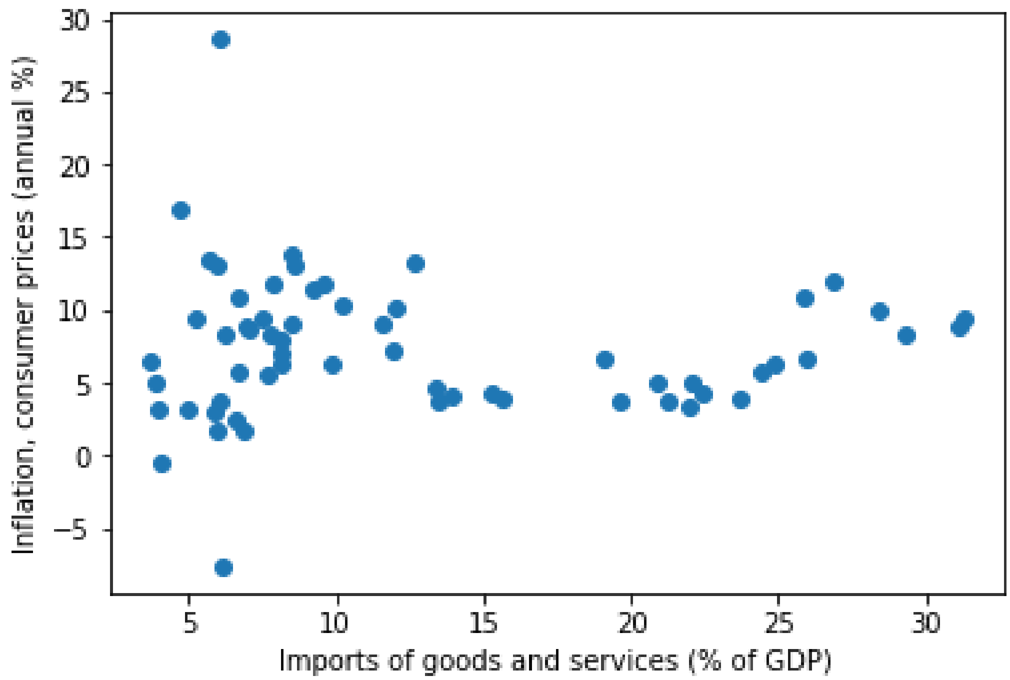
Population growth (annual %)
GDP growth (annual %)



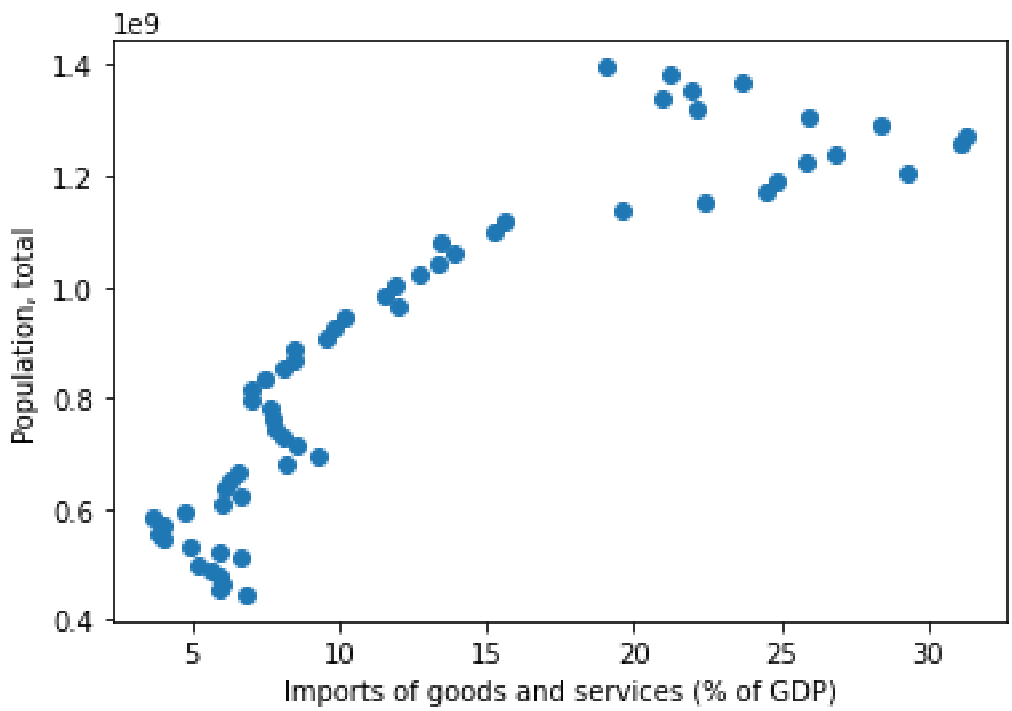
Life expectancy at birth, total (years)
GDP growth (annual %)


Exports of goods and services (% of GDP)
Imports of goods and services (% of GDP)
5
10
15
20
25
30

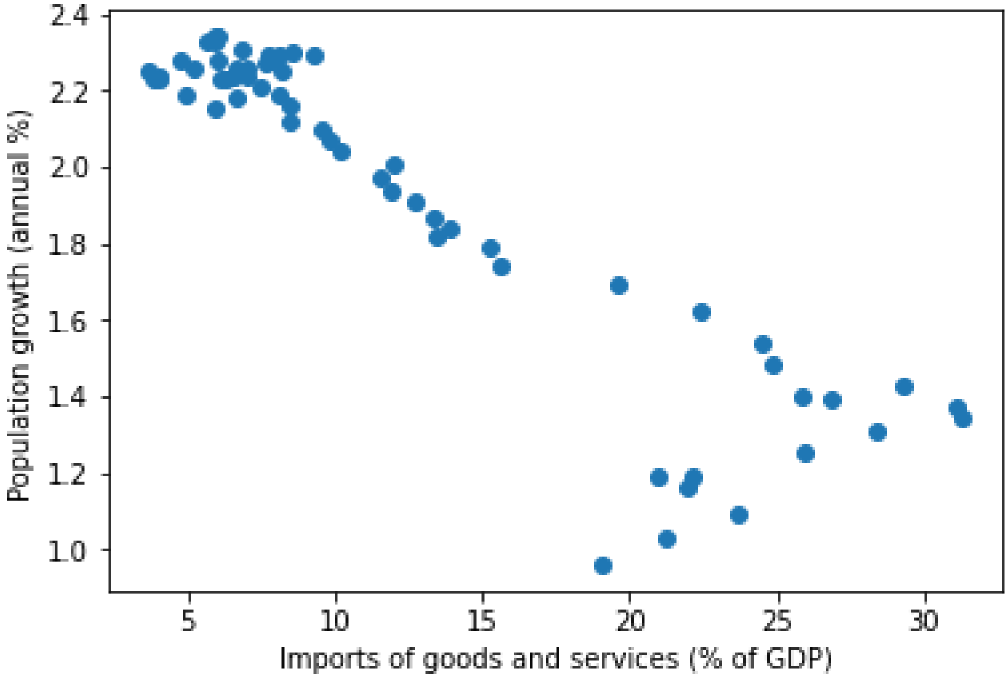

1e11
Total reserves (includes gold, current US$)
Imports of goods and services (% of GDP)
0
1
2
3
4
5
6
5
10
15
20
25
30

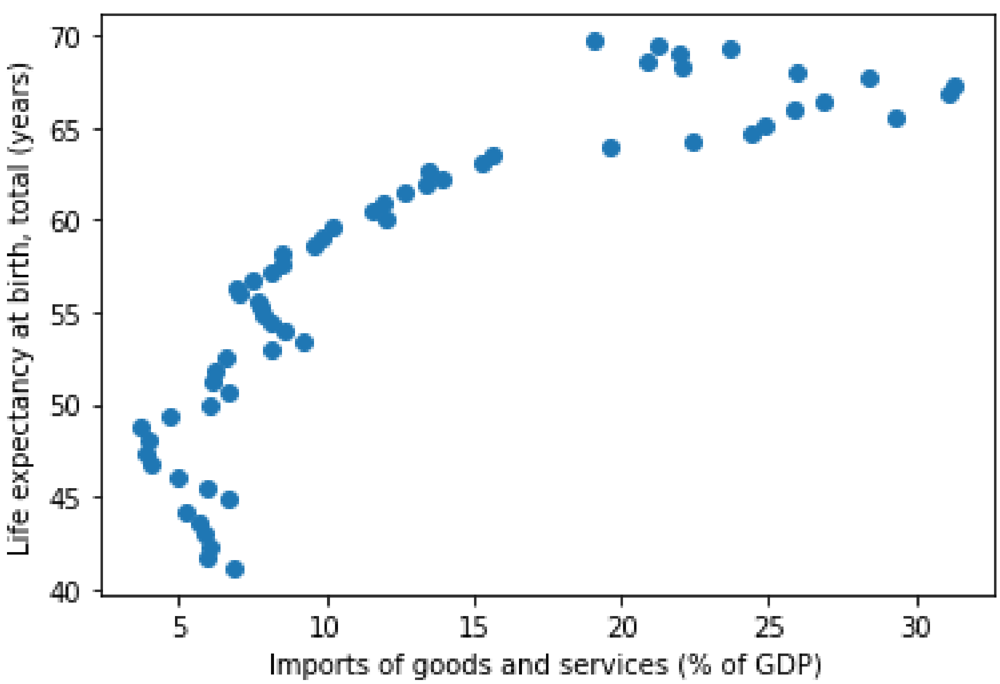


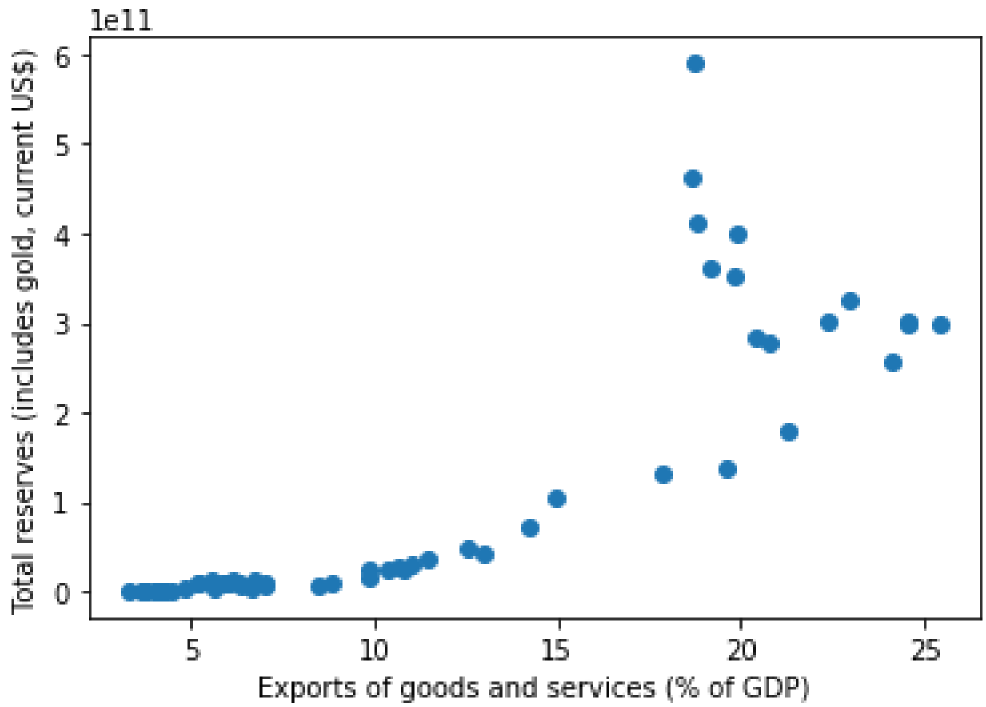
Inflation, consumer prices (annual %)
Imports of goods and services (% of GDP)
30
25
20
15
10
5
0
−5
5
10
15
20
25
30



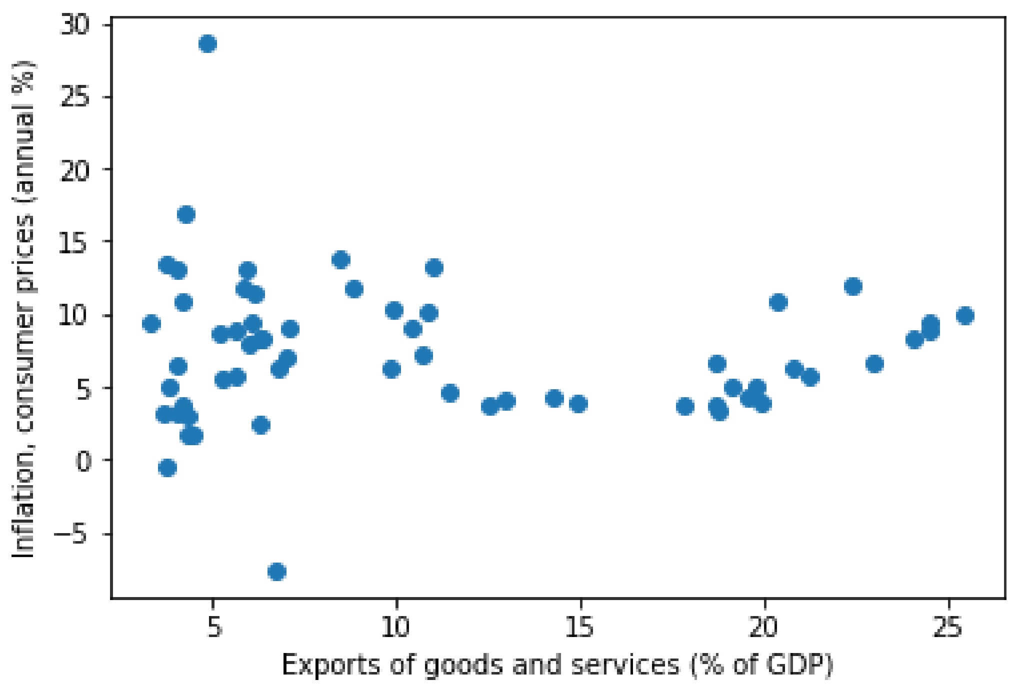
1e9
Population, total
Imports of goods and services (% of GDP)
1.4
1.2
1.0
0.8
0.6
0.4
5
10
15
20
25
30


Population growth (annual %)
Imports of goods and services (% of GDP)

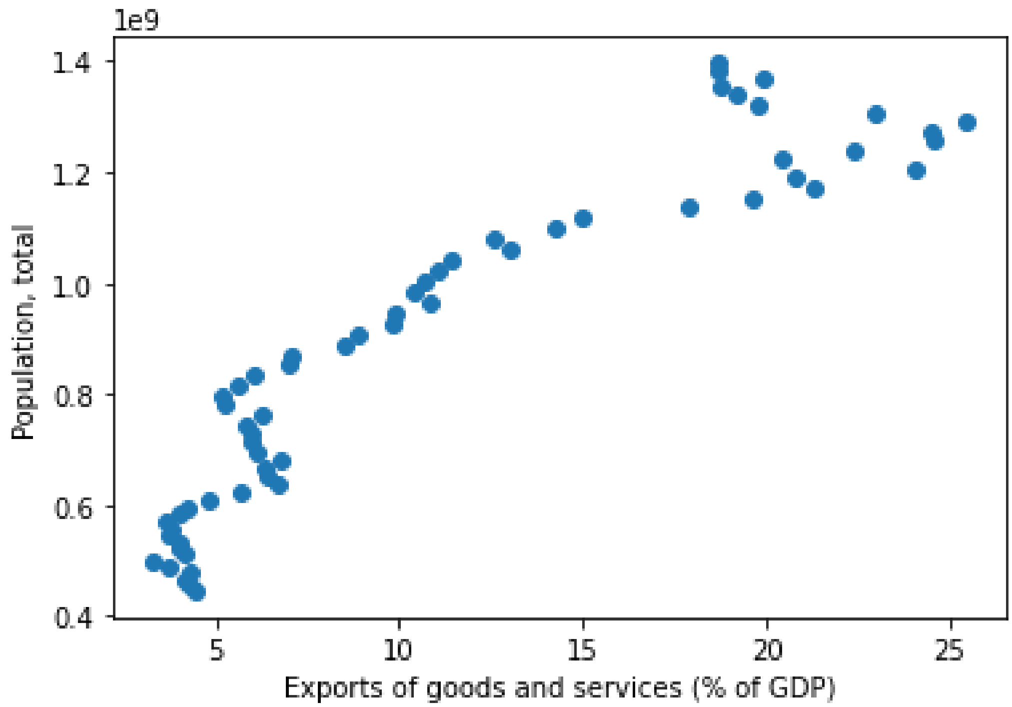

Life expectancy at birth, total (years)
Imports of goods and services (% of GDP)

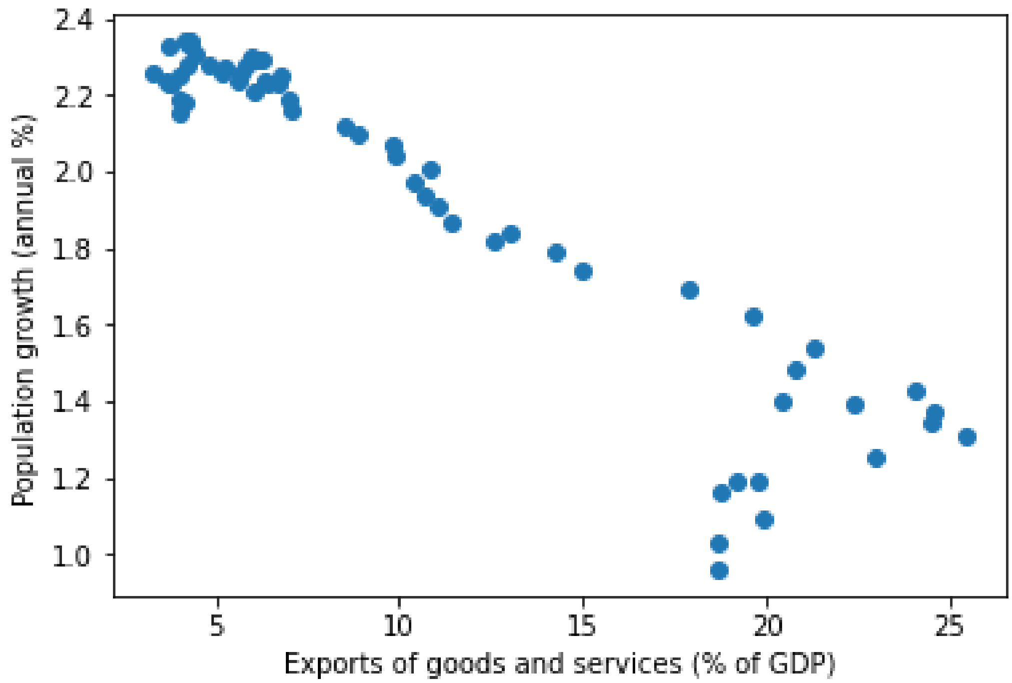


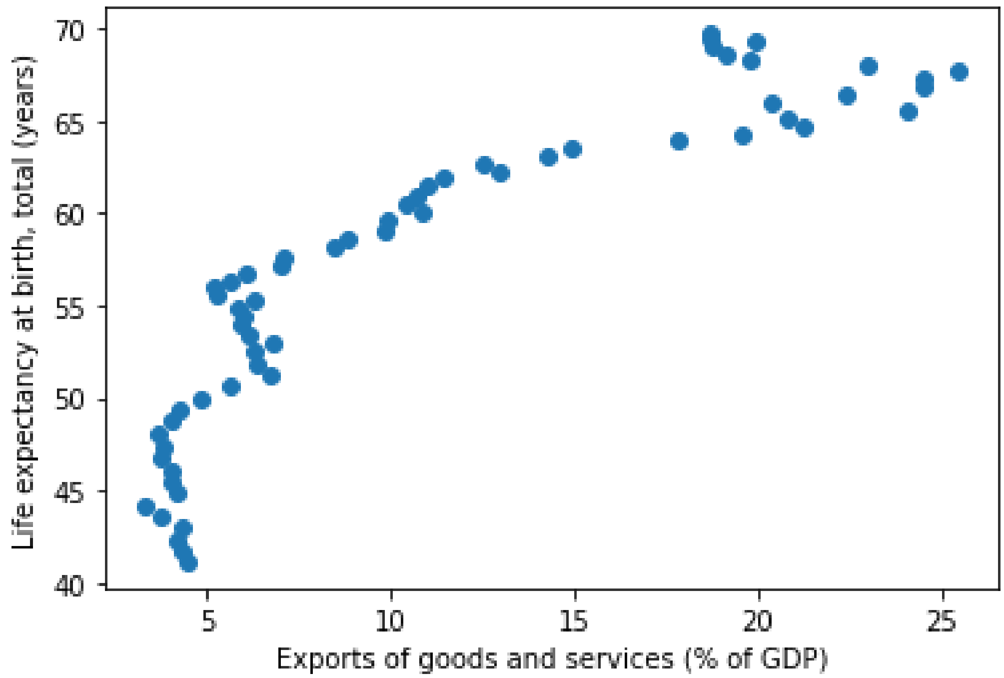
1e11
Total reserves (includes gold, current US$)
Exports of goods and services (% of GDP)



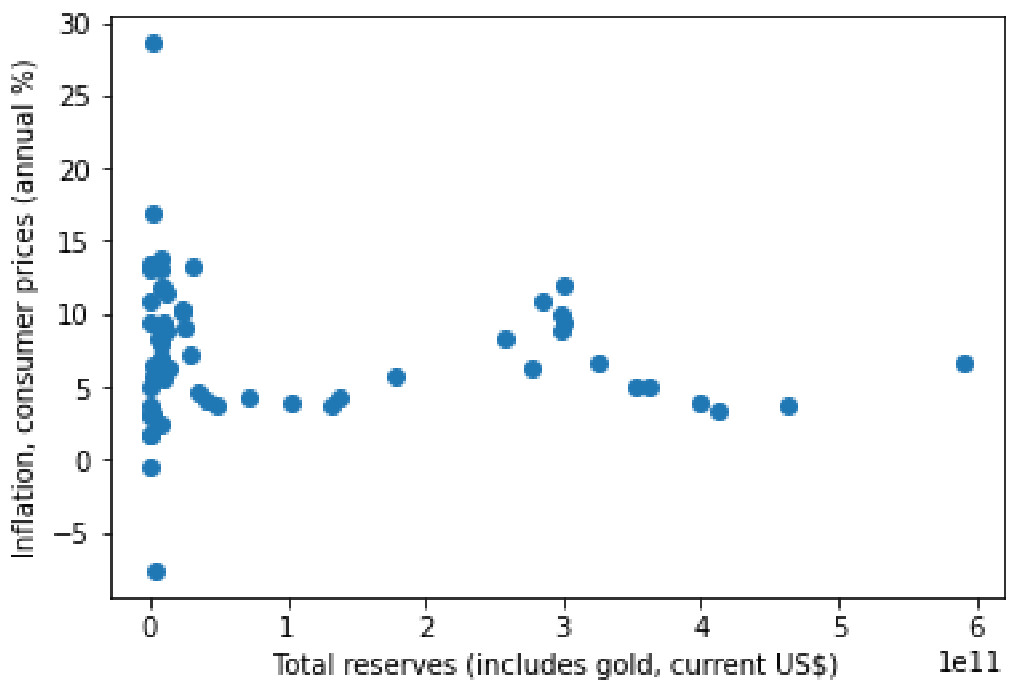
Inflation, consumer prices (annual %)
Exports of goods and services (% of GDP)


1e9
1.4
1.2
1.0
0.8
0.6
0.4
Population, total
5
10
15
20
25
Exports of goods and services (% of GDP)

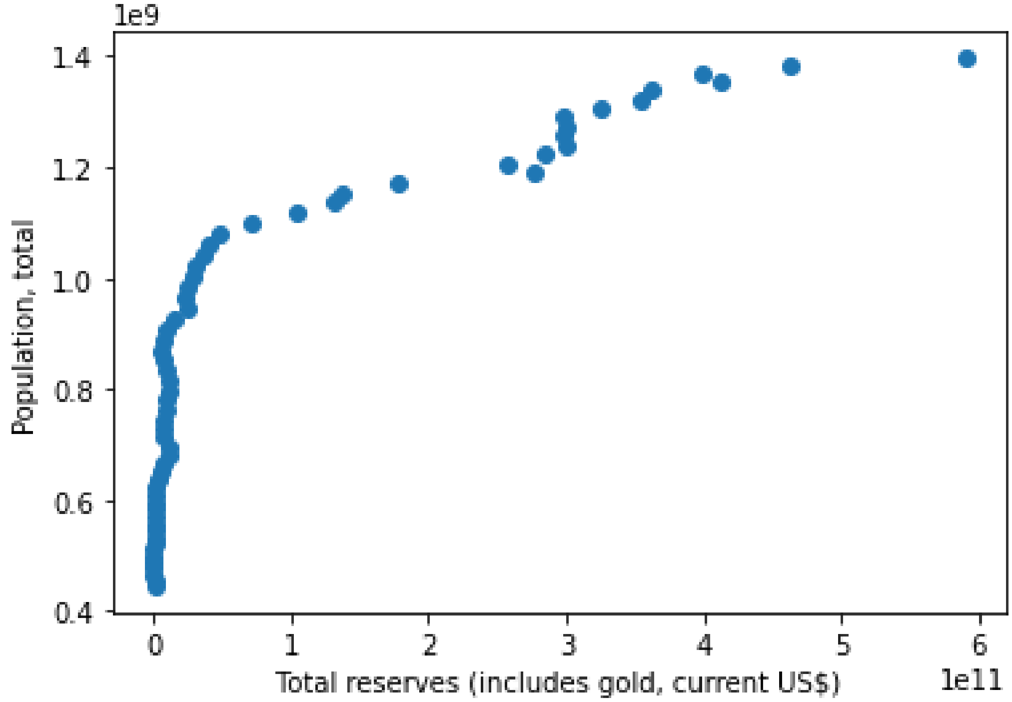

2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
Population growth (annual %)
5
10
15
20
25
Exports of goods and services (% of GDP)

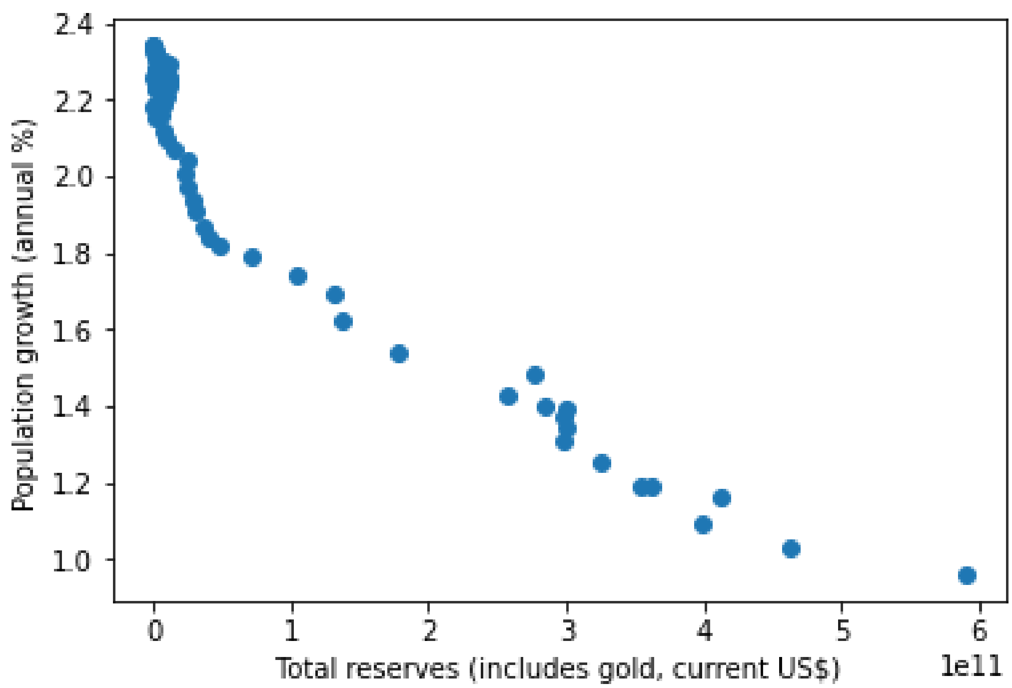


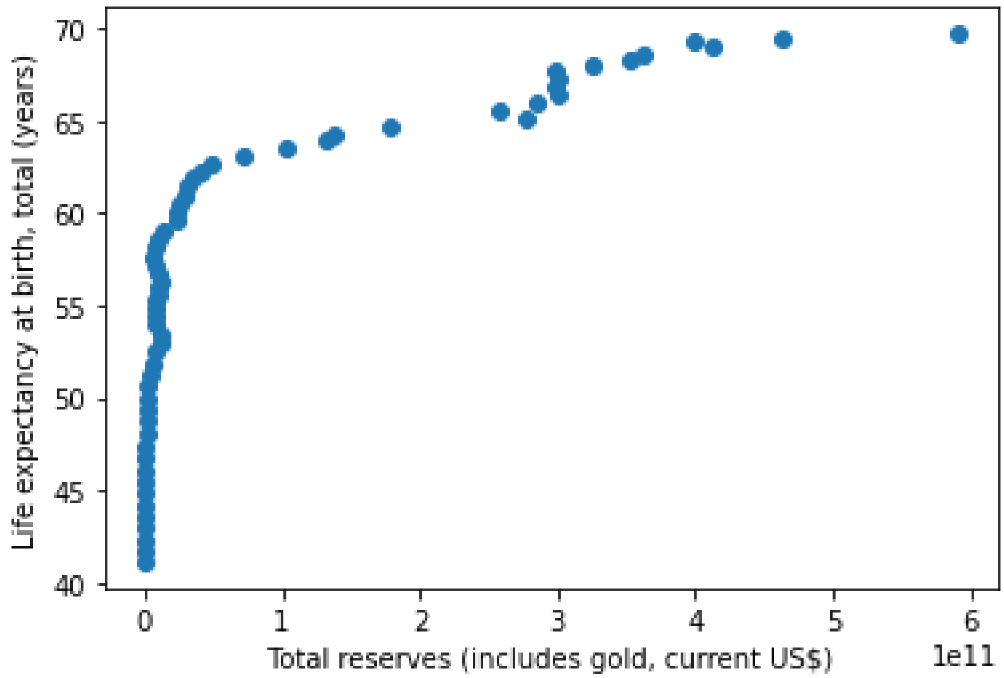
Life expectancy at birth, total (years)
Exports of goods and services (% of GDP)



Inflation, consumer prices (annual %)
Total reserves (includes gold, current US$)
1e11


1e9
1.4
1.2
1.0
0.8
0.6
0.4
Population, total
0
1
2
3
4
5
6
Total reserves (includes gold, current US$)
1e11


2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
Population growth (annual %)
0
1
2
3
4
5
6
Total reserves (includes gold, current US$)
1e11

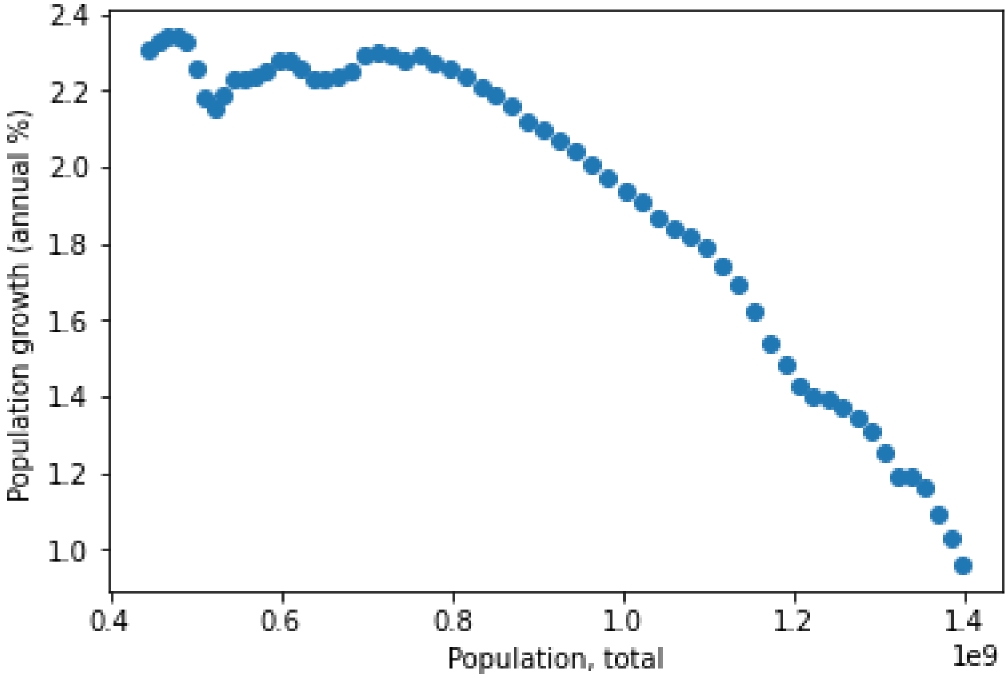


Life expectancy at birth, total (years)
Total reserves (includes gold, current US$)
1e11

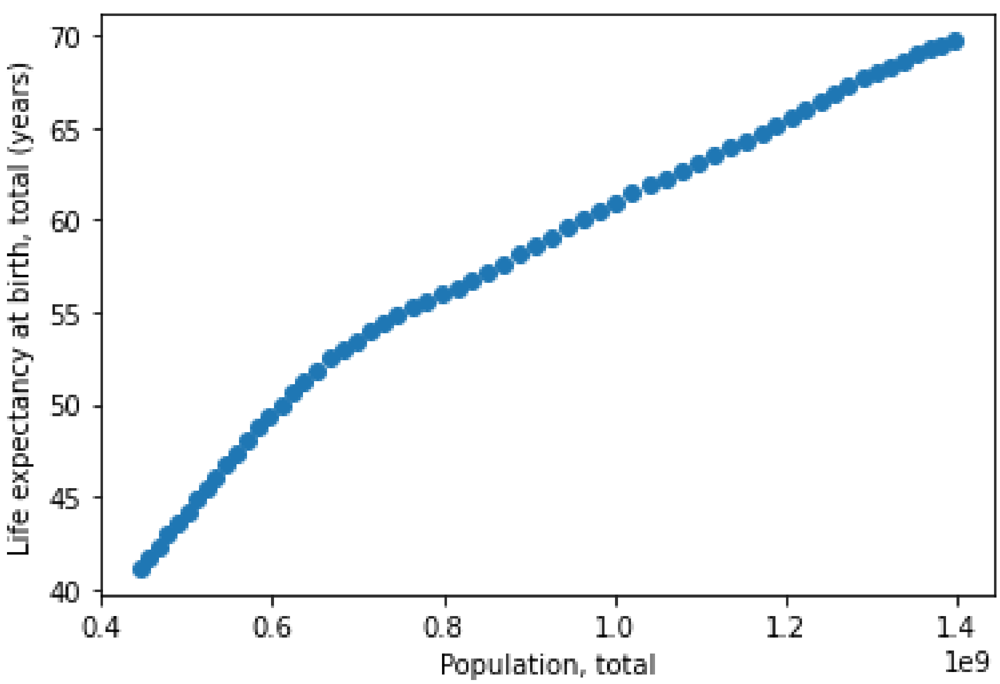
1e9
Population, total
Inflation, consumer prices (annual %)


Population growth (annual %)
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
−5
0
5
10
15
20
25
30
Inflation, consumer prices (annual %)

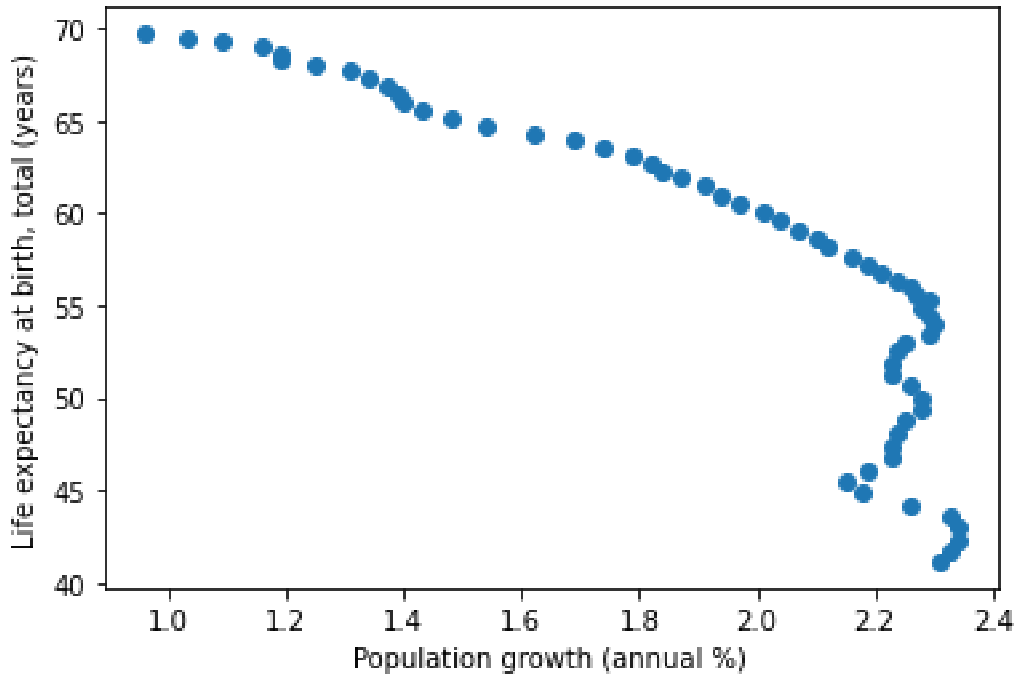

Life expectancy at birth, total (years)
70
65
60
55
50
45
40
−5
0
5
10
15
20
25
30
Inflation, consumer prices (annual %)



Population growth (annual %)
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.4
0.6
0.8
1.0
1.2
1.4
1e9
Population, total



Life expectancy at birth, total (years)
70
65
60
55
50
45
40
0.4
0.6
0.8
1.0
1.2
1.4
1e9
Population, total

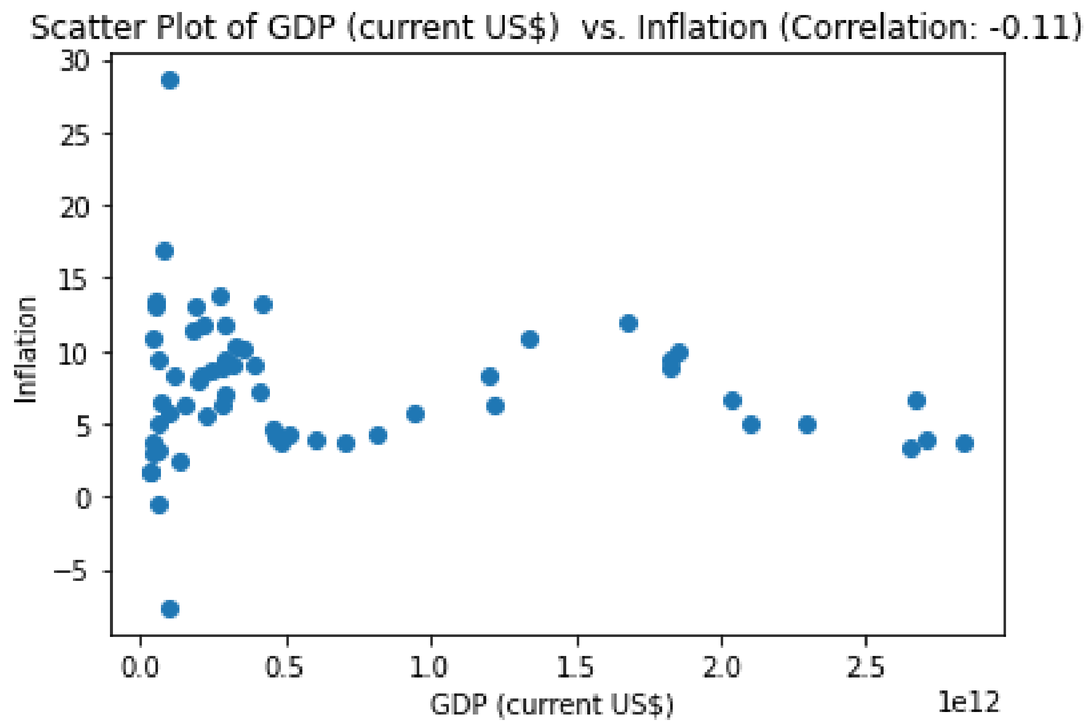


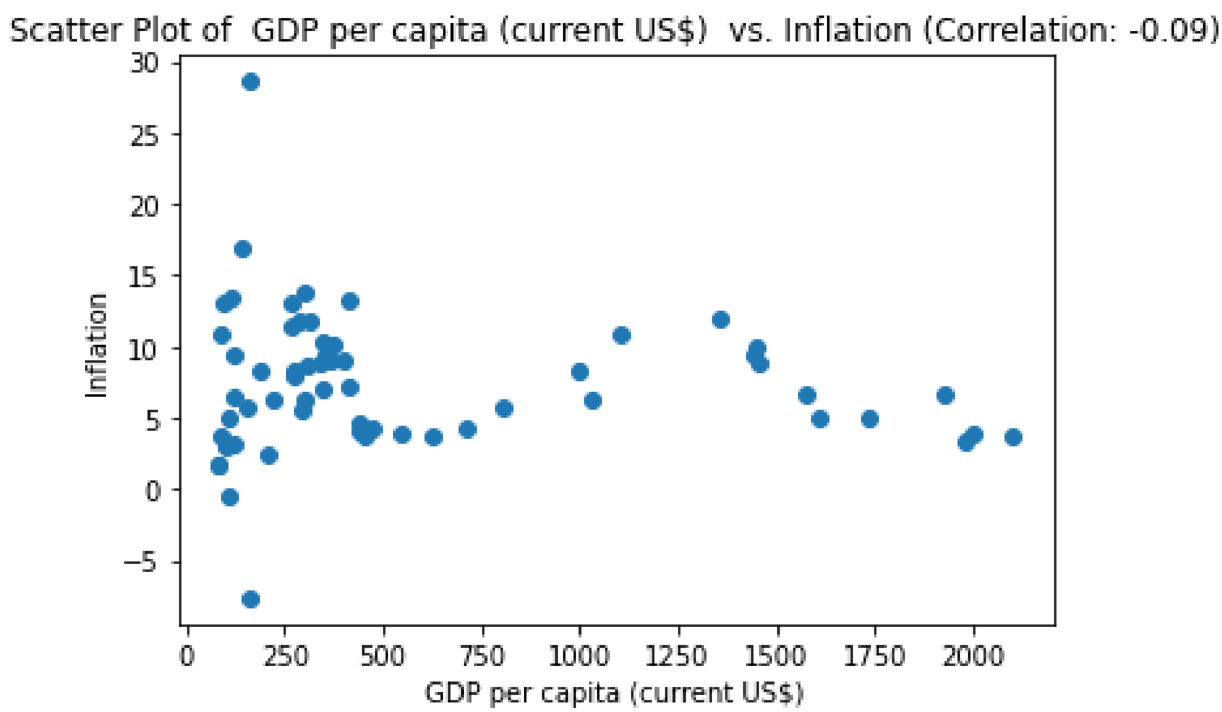

```
data.columns
```

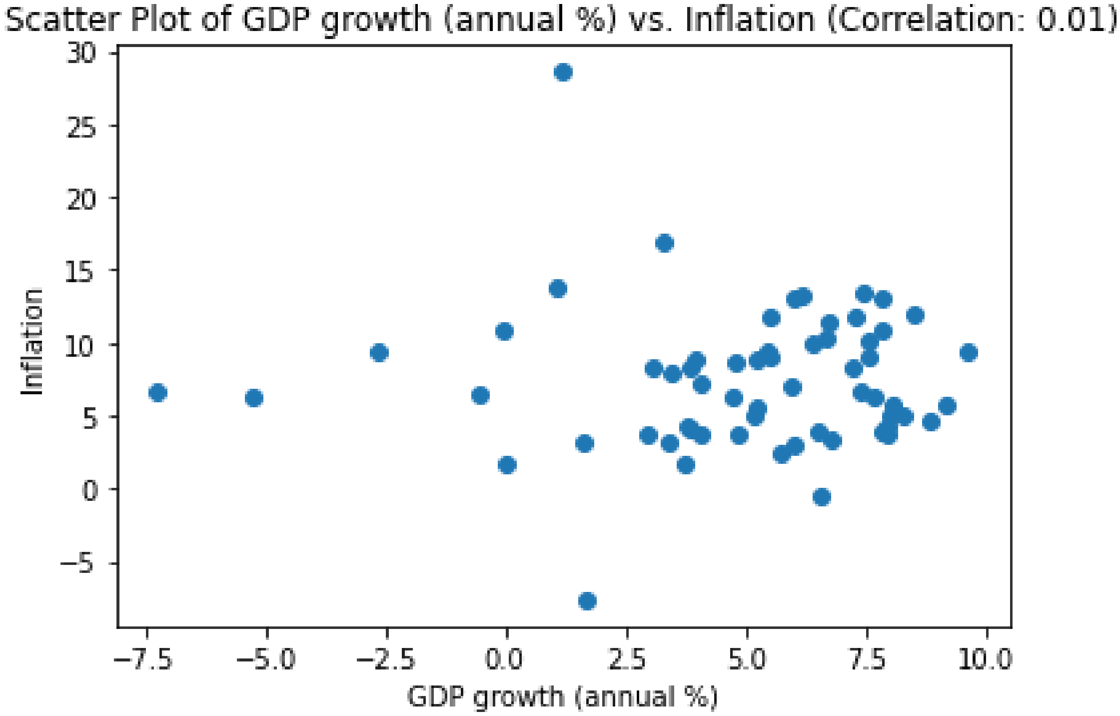
```
Index(['Year', 'GDP (current US$) ', ' GDP per capita (current US$) ',
       'GDP growth (annual %)', 'Imports of goods and services (% of
GDP)',
       'Exports of goods and services (% of GDP)',
       ' Total reserves (includes gold, current US$) ',
       'Inflation, consumer prices (annual %)', 'Population, total',
       'Population growth (annual %)',
       'Life expectancy at birth, total (years)'],
      dtype='object')
```

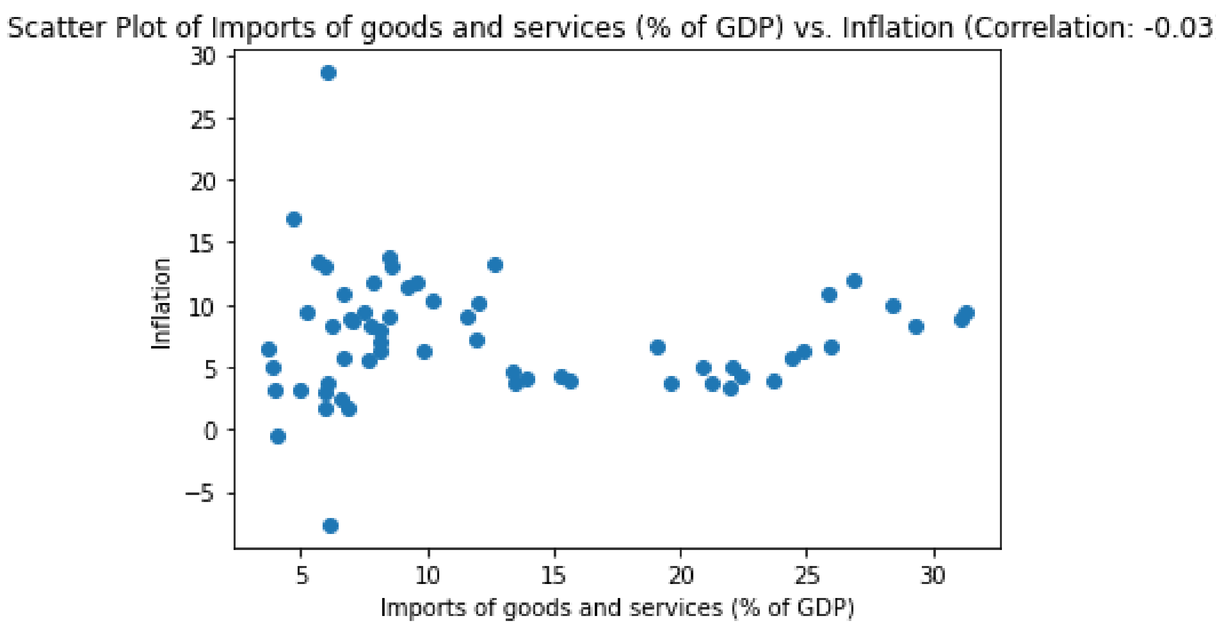
```
for col in data.columns:
    if col not in ("Year", "Inflation, consumer prices (annual %)"):
        # Create a scatter plot of the current column against the
"Inflation" column
        plt.scatter(data[col], data["Inflation, consumer prices
(annual %)"])
        plt.xlabel(col)
        plt.ylabel("Inflation")

        # Calculate the correlation between the current column and the
"Inflation" column
        correlation = np.corrcoef(data[col], data["Inflation, consumer
prices (annual %)"])[0, 1]

        # Display the correlation coefficient in the plot title
        plt.title(f"Scatter Plot of {col} vs. Inflation (Correlation:
```

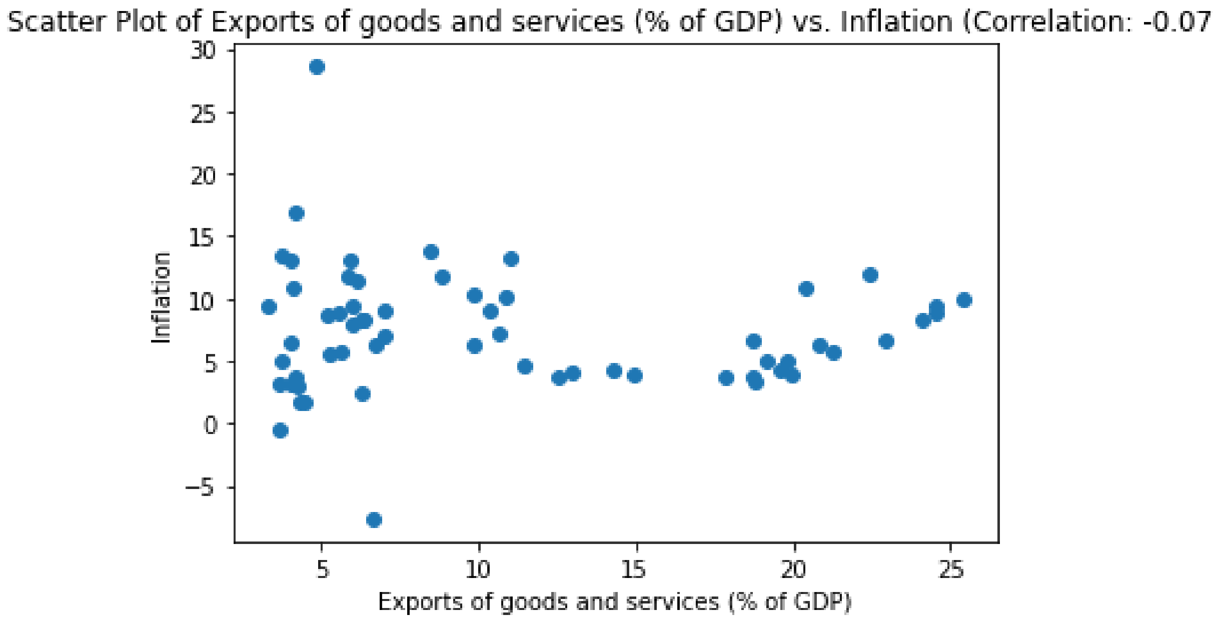
```
{correlation:.2f})")
        plt.show()
```

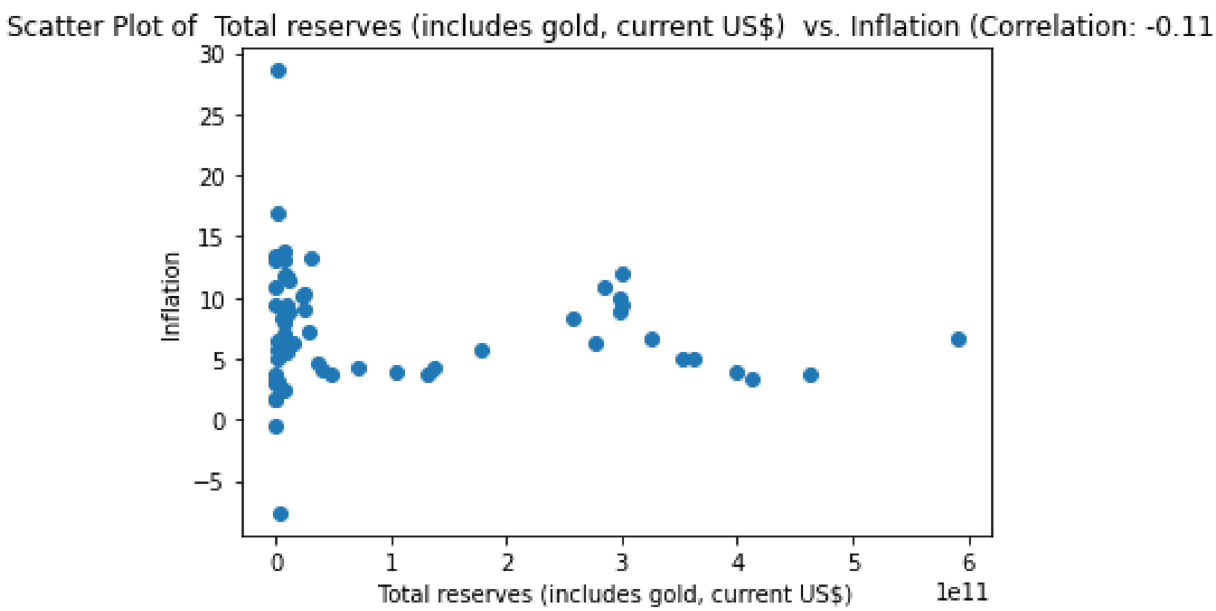




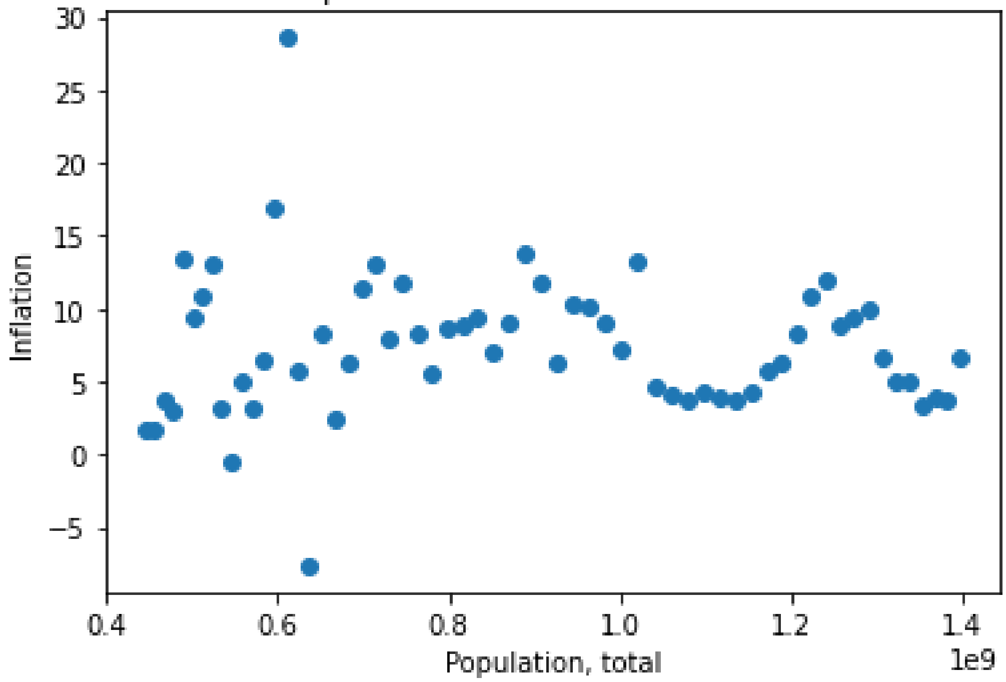

Scatter Plot of GDP growth (annual %) vs. Inflation (Correlation: 0.01)
Inflation
GDP growth (annual %)
30
25
20
15
10
5
0
−5
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Scatter Plot of Imports of goods and services (% of GDP) vs. Inflation (Correlation: -0.03)
Inflation
Imports of goods and services (% of GDP)
30
25
20
15
10
5
0
−5
5
10
15
20
25
30

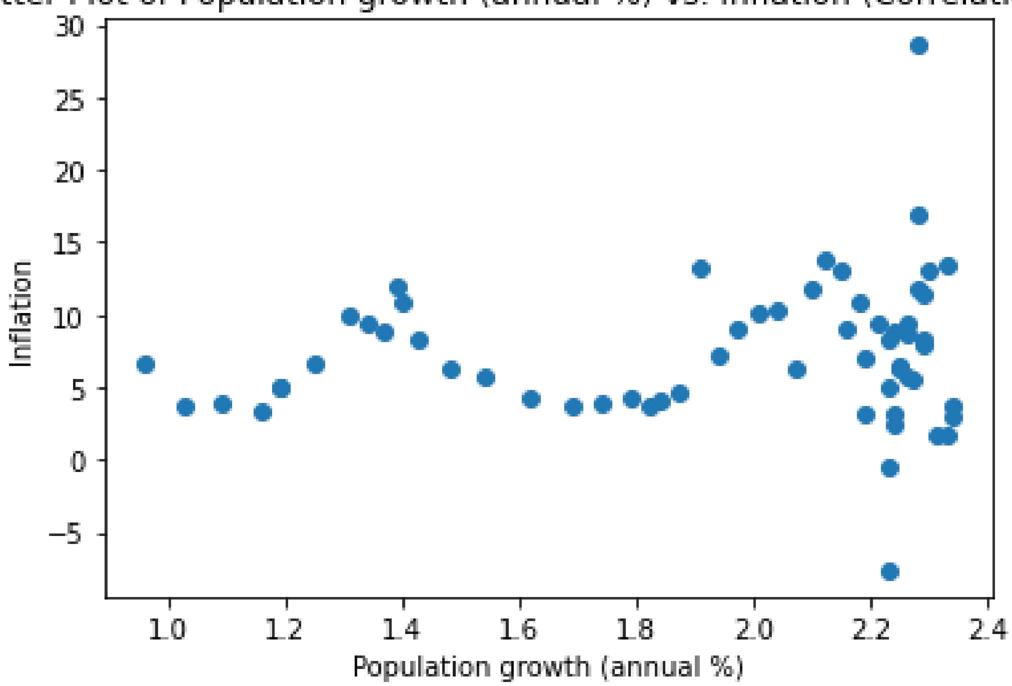

Scatter Plot of Exports of goods and services (% of GDP) vs. Inflation (Correlation: -0.07)
Inflation
Exports of goods and services (% of GDP)

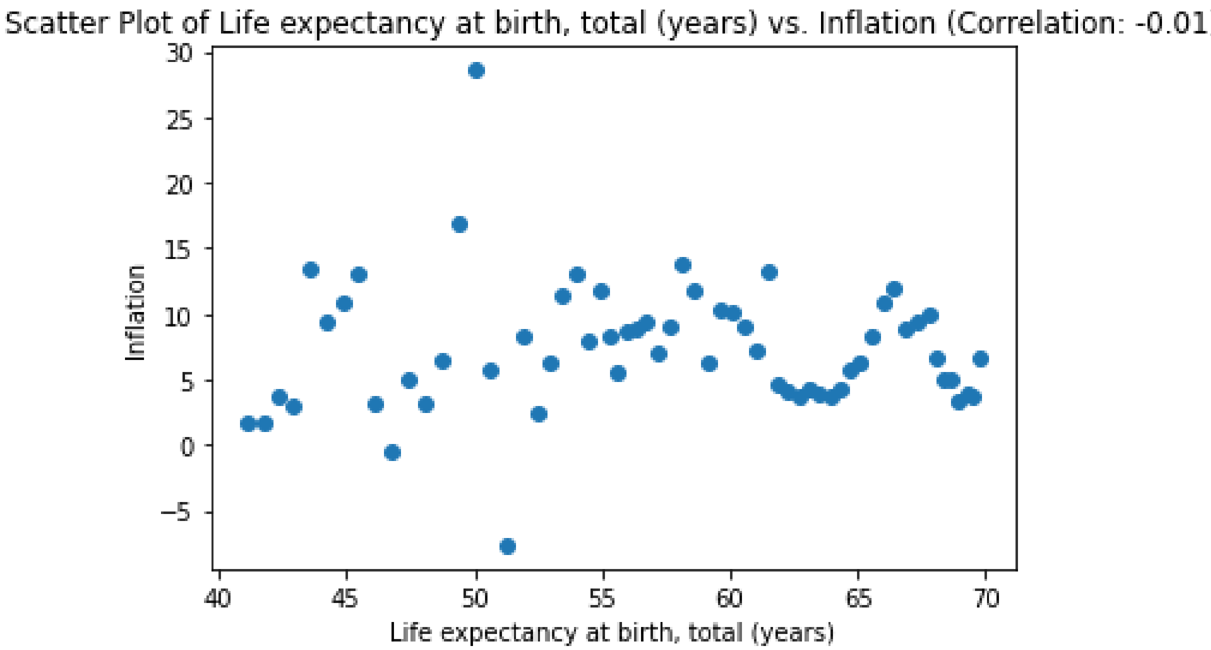

Scatter Plot of Total reserves (includes gold, current US$) vs. Inflation (Correlation: -0.11)
Inflation
Total reserves (includes gold, current US$)
1e11

Scatter Plot of Population, total vs. Inflation (Correlation: -0.05)



Scatter Plot of Population growth (annual %) vs. Inflation (Correlation: 0.12)






```
# prompt: now remove all columns and take only two colums, year and
inflation

data_inflation = data[['Year', 'Inflation, consumer prices (annual
%)']]
data_inflation
```

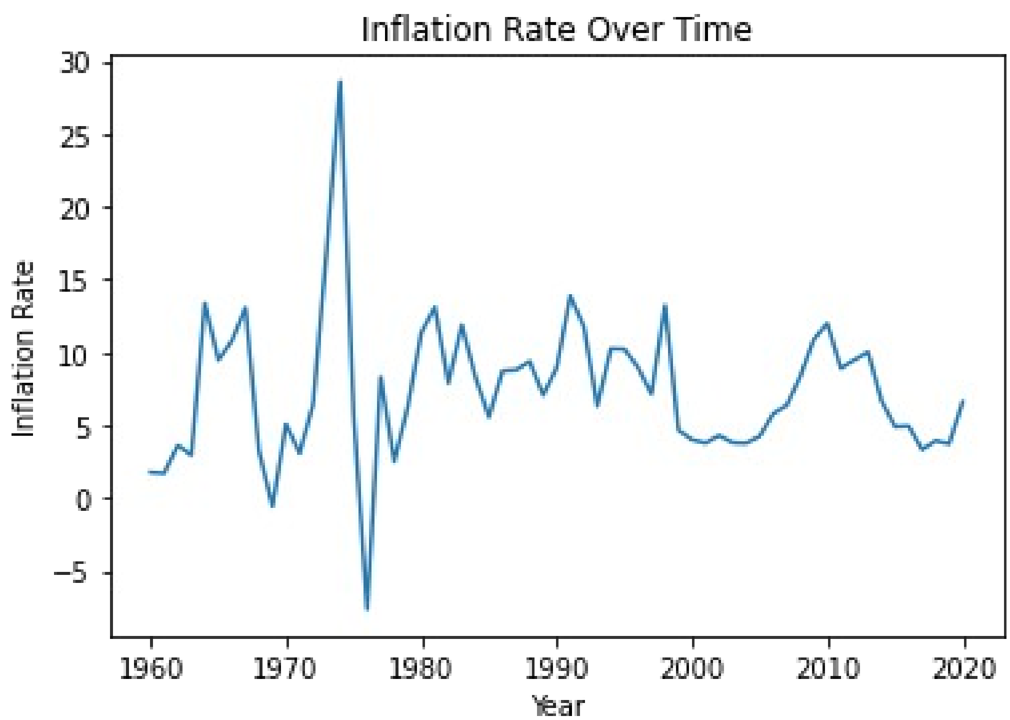
```
    Year  Inflation, consumer prices (annual %)
0   1960                                   1.78
1   1961                                   1.70
2   1962                                   3.63
3   1963                                   2.95
4   1964                                  13.36
..   ...                                    ...
56  2016                                   4.95
57  2017                                   3.33
58  2018                                   3.94
59  2019                                   3.73
60  2020                                   6.62

[61 rows x 2 columns]
```

```
# prompt: rename the inflation column with only "inflstipn"

data_inflation = data_inflation.rename(columns={'Inflation, consumer
prices (annual %)': 'inflation'})
data_inflation
```

```
    Year  inflation
0   1960       1.78
```

```
1   1961       1.70
2   1962       3.63
3   1963       2.95
4   1964      13.36
..   ...        ...
56  2016       4.95
57  2017       3.33
58  2018       3.94
59  2019       3.73
60  2020       6.62

[61 rows x 2 columns]
```

```
# @title Inflation Rate Over Time

import matplotlib.pyplot as plt
plt.plot(data_inflation['Year'], data_inflation['inflation'])
plt.title('Inflation Rate Over Time')
plt.xlabel('Year')
_ = plt.ylabel('Inflation Rate')
```

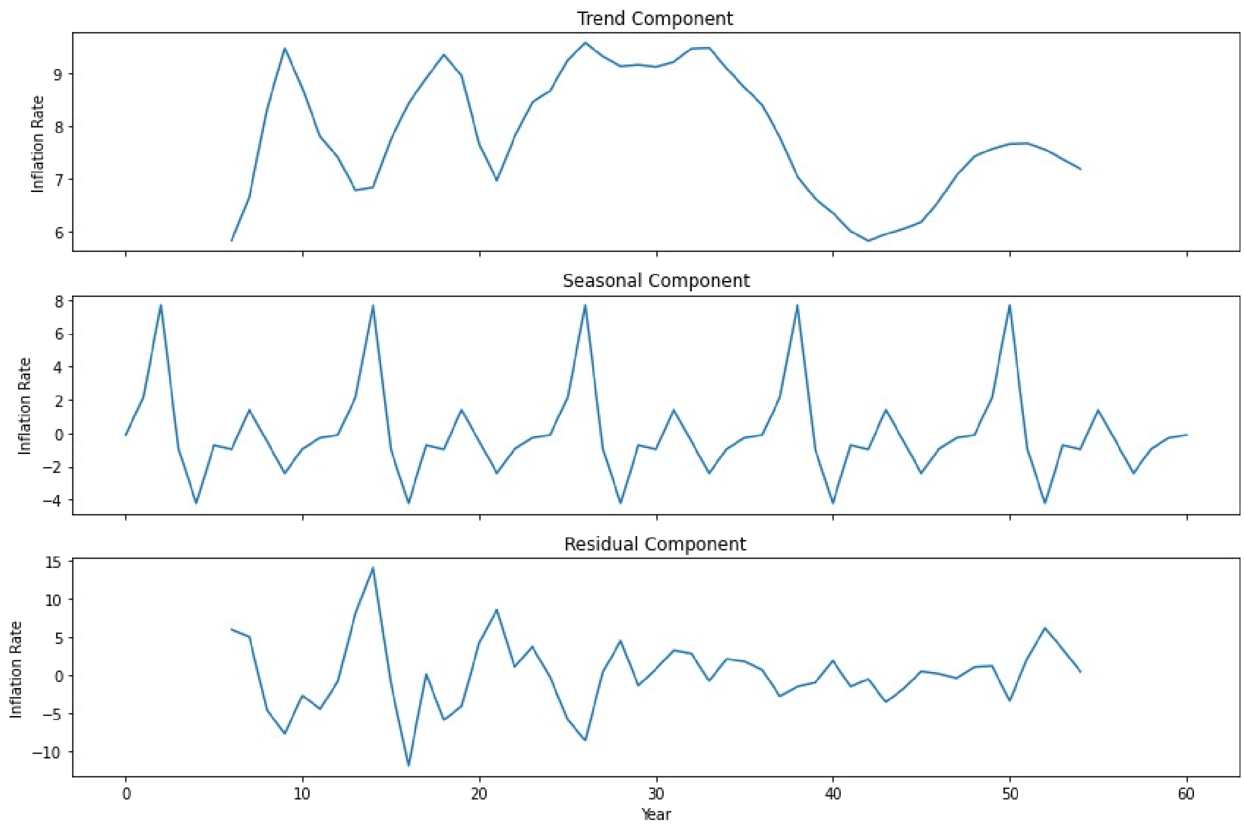



```
# prompt: now do the adfuller test and show whether it is stationary
or not

from statsmodels.tsa.stattools import adfuller

adf_result = adfuller(data_inflation['inflation'])
```

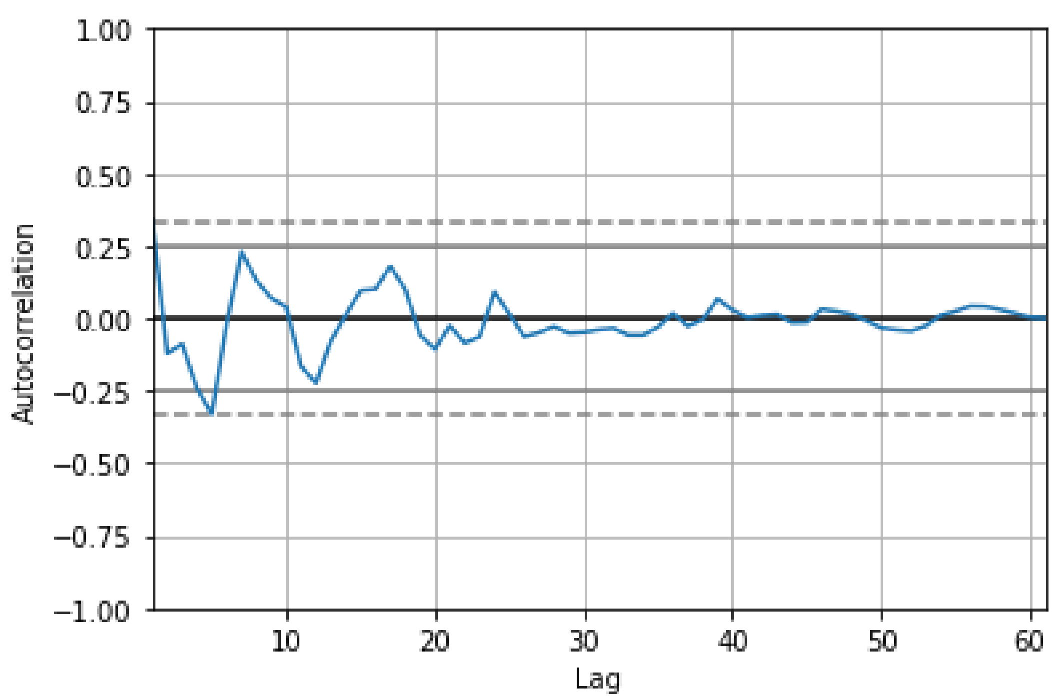
```python
print("ADF Statistic:", adf_result[0])
print("p-value:", adf_result[1])

if adf_result[1] > 0.05:
  print("The inflation rate is not stationary.")
else:
  print("The inflation rate is stationary.")
```

```
ADF Statistic: -4.853443951169606
p-value: 4.293062853742427e-05
The inflation rate is stationary.
```

```python
from statsmodels.tsa.seasonal import seasonal_decompose

# Decompose the time series
result = seasonal_decompose(data_inflation['inflation'], period=12,
model='additive')

# Plot the decomposed components
fig, axs = plt.subplots(3, 1, figsize=(12, 8), sharex=True)

# Plot the trend component
result.trend.plot(ax=axs[0], label='Trend')
axs[0].set_ylabel('Inflation Rate')
axs[0].set_title('Trend Component')

# Plot the seasonal component
result.seasonal.plot(ax=axs[1], label='Seasonality')
axs[1].set_ylabel('Inflation Rate')
axs[1].set_title('Seasonal Component')

# Plot the residual component
result.resid.plot(ax=axs[2], label='Residuals')
axs[2].set_xlabel('Year')
axs[2].set_ylabel('Inflation Rate')
axs[2].set_title('Residual Component')

plt.tight_layout()
plt.show()
```

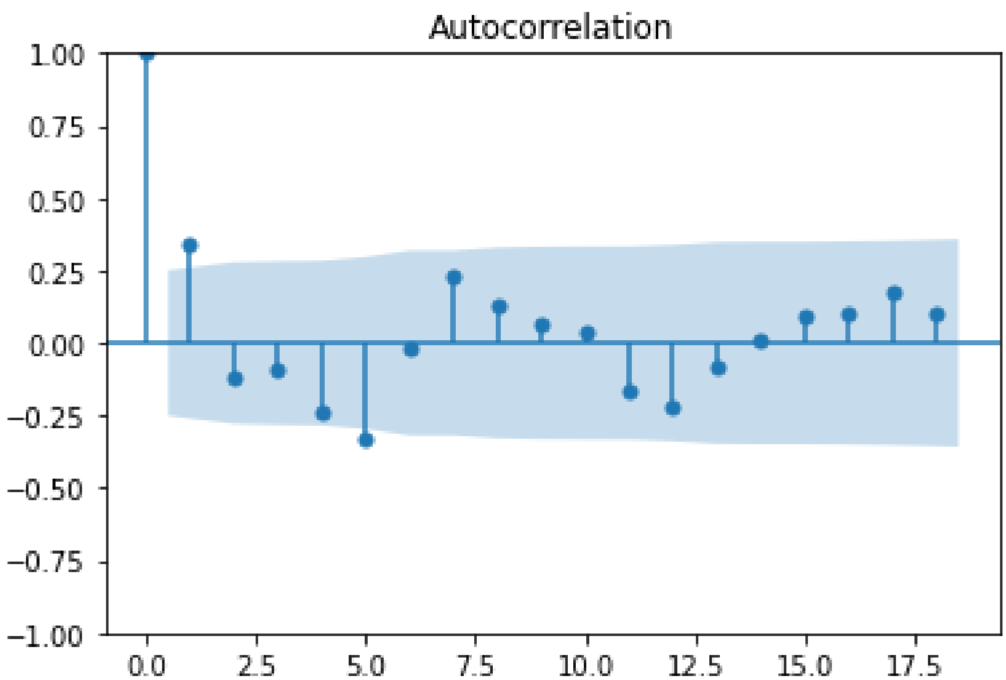


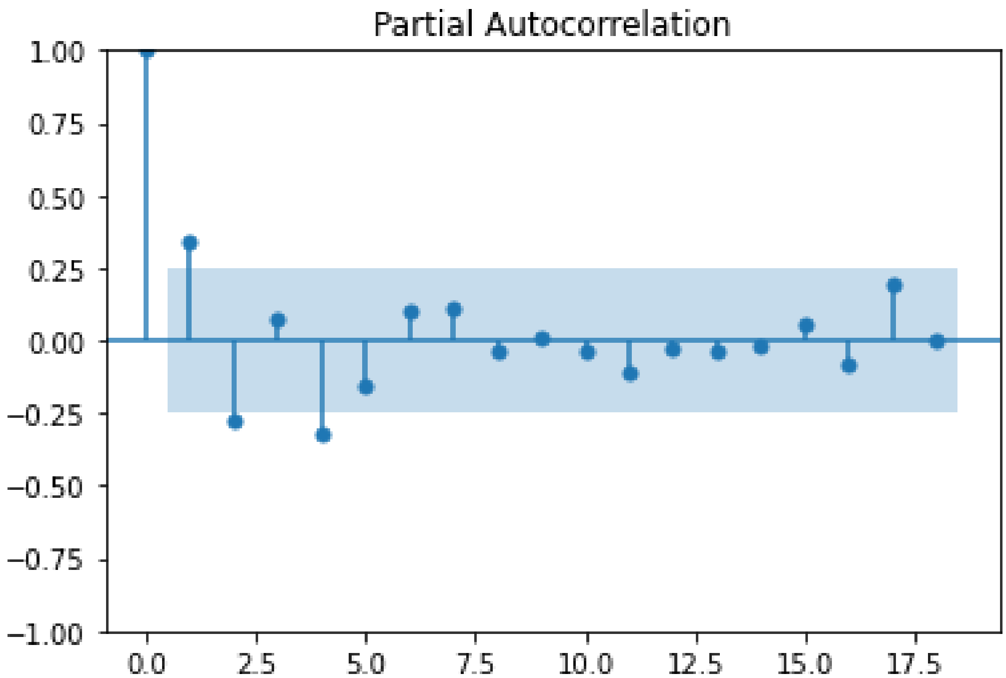

```python
from pandas.plotting import autocorrelation_plot
autocorrelation_plot(data_inflation['inflation'])
plt.show()
```



```python
# prompt: import plot_acf  and plot_pacf for acf and pacf

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
fig = plt.figure(figsize=(12,8))

# Plot the ACF of the inflation rate
plot_acf(data_inflation['inflation'])
plt.show()

# Plot the PACF of the inflation rate
plot_pacf(data_inflation['inflation'])
plt.show()
```

```
<Figure size 864x576 with 0 Axes>
```

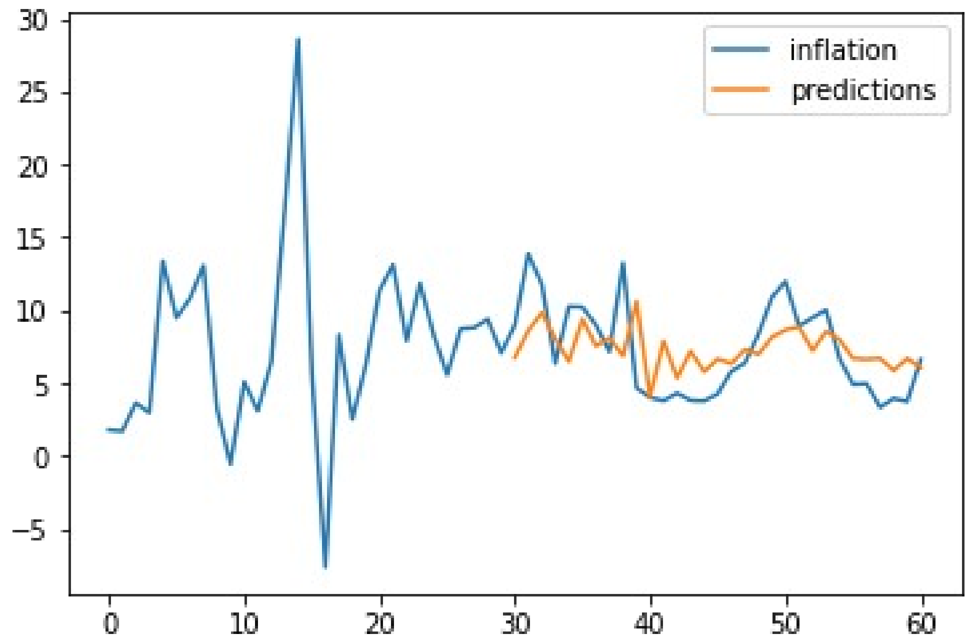





```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(data_inflation['inflation'], order=(1,0,1))

result= model.fit()
result.summary()

<class 'statsmodels.iolib.summary.Summary'>
"""
                               SARIMAX Results

======================================================================
========
Dep. Variable:              inflation   No. Observations:
61
Model:                 ARIMA(1, 0, 1)   Log Likelihood
-176.784
Date:                Sat, 04 May 2024   AIC
361.567
Time:                        14:19:59   BIC
370.011
Sample:                             0   HQIC
364.876
                                 - 61

Covariance Type:                  opg

======================================================================
========
```

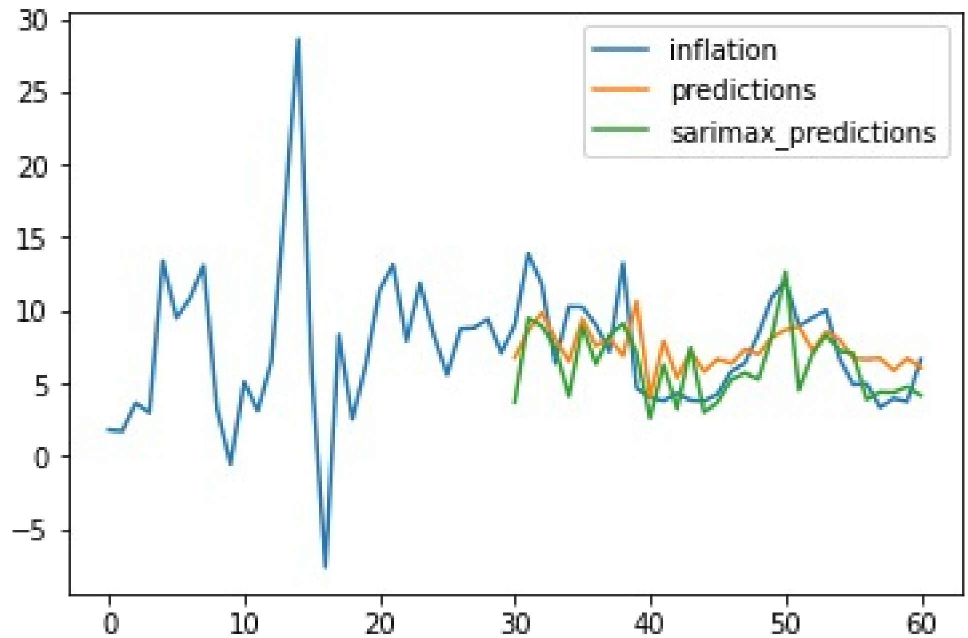
```
                 coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
--------
const          7.4082      0.919      8.060      0.000       5.607
9.210
ar.L1         -0.1399      0.189     -0.739      0.460      -0.511
0.231
ma.L1          0.6306      0.175      3.613      0.000       0.289
0.973
sigma2        19.1576      2.607      7.349      0.000      14.048
24.267
======================================================================
=============
Ljung-Box (L1) (Q):                  0.00   Jarque-Bera (JB):
22.68
Prob(Q):                             0.98   Prob(JB):
0.00
Heteroskedasticity (H):              0.12   Skew:
0.78
Prob(H) (two-sided):                 0.00   Kurtosis:
5.54
======================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
"""


# Make predictions
data_inflation['predictions'] =
result.predict( start=30,end=60,dynamics=False)

data_inflation

    Year  inflation  predictions
0   1960       1.78          NaN
1   1961       1.70          NaN
2   1962       3.63          NaN
3   1963       2.95          NaN
4   1964      13.36          NaN
..   ...        ...          ...
56  2016       4.95     6.635637
57  2017       3.33     6.689094
58  2018       3.94     5.860450
59  2019       3.73     6.682327
60  2020       6.62     6.060997
```

```
[61 rows x 3 columns]

data_inflation[['inflation','predictions']].plot()

<Axes: >
```




```
import statsmodels.api as sm

# prompt: now use sarimax

model = sm.tsa.statespace.SARIMAX(data_inflation['inflation'],
order=(1, 0, 1), seasonal_order=(1, 0, 1, 12))
result_sarimax = model.fit()
result_sarimax.summary()

data_inflation['sarimax_predictions'] =
result_sarimax.predict(start=30,end=60)
print(data_inflation)
data_inflation[['inflation', 'predictions',
'sarimax_predictions']].plot()
plt.show()

C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\statespace\sarimax.py:1009: UserWarning: Non-
invertible starting seasonal moving average Using zeros as starting
parameters.
  warn('Non-invertible starting seasonal moving average'
```

```
    Year  inflation  predictions  sarimax_predictions
0   1960       1.78          NaN                  NaN
1   1961       1.70          NaN                  NaN
2   1962       3.63          NaN                  NaN
3   1963       2.95          NaN                  NaN
4   1964      13.36          NaN                  NaN
..   ...        ...          ...                  ...
56  2016       4.95     6.635637             3.861878
57  2017       3.33     6.689094             4.387021
58  2018       3.94     5.860450             4.351068
59  2019       3.73     6.682327             4.750550
60  2020       6.62     6.060997             4.137554

[61 rows x 4 columns]
```




```
# prompt: check the models with accuracy check
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error

# Calculate the mean absolute error (MAE) for both models
mae_arima = mean_absolute_error(data_inflation['inflation'].iloc[-
31:], data_inflation['predictions'].dropna())
mae_sarimax = mean_absolute_error(data_inflation['inflation'].iloc[-
31:], data_inflation['sarimax_predictions'].dropna())

# Calculate the mean absolute percentage error (MAPE) for both models
mape_arima = mean_absolute_error(data_inflation['inflation'].iloc[-
31:], data_inflation['predictions'].dropna())
mape_sarimax = mean_absolute_error(data_inflation['inflation'].iloc[-
```

```
31:], data_inflation['sarimax_predictions'].dropna())


rmse = mean_squared_error(data_inflation['inflation'].iloc[-31:],
data_inflation['predictions'].dropna())
sarimax_rmse = mean_squared_error(data_inflation['inflation'].iloc[-
31:], data_inflation['sarimax_predictions'].dropna())


# Print the accuracy metrics for both models
print("ARIMA Model:")
print(f"  - RMSE: {rmse}")
print(f"  - MAE: {mae_arima}")
print(f"  - MAPE: {mape_arima}")

print("SARIMAX Model:")
print(f"  - RMSE: {sarimax_rmse}")
print(f"  - MAE: {mae_sarimax}")
print(f"  - MAPE: {mape_sarimax}")
```

```
ARIMA Model:
  - RMSE: 7.515786121078491
  - MAE: 2.246796794429391
  - MAPE: 2.246796794429391
SARIMAX Model:
  - RMSE: 6.854299786986503
  - MAE: 2.1446725526647663
  - MAPE: 2.1446725526647663
```

```
data_inflation['inflation'].mean()
```

```
7.41327868852459
```

```
data_inflation.tail()
```

	Year	inflation	predictions	sarimax_predictions
56	2016	4.95	6.635637	3.861878
57	2017	3.33	6.689094	4.387021
58	2018	3.94	5.860450	4.351068
59	2019	3.73	6.682327	4.750550
60	2020	6.62	6.060997	4.137554

```
# prompt: change the year column to date time

import pandas as pd
data_inflation['Year'] = pd.to_datetime(data_inflation['Year'],
format='%Y')

data_inflation.tail(10)
```

	Year	inflation	predictions	sarimax_predictions
51	2011-01-01	8.91	8.875637	4.510633

```
52 2012-01-01       9.48     7.219701                7.010053
53 2013-01-01      10.02     8.543648                8.285373
54 2014-01-01       6.67     7.973733                7.207264
55 2015-01-01       4.91     6.689282                7.038992
56 2016-01-01       4.95     6.635637                3.861878
57 2017-01-01       3.33     6.689094                4.387021
58 2018-01-01       3.94     5.860450                4.351068
59 2019-01-01       3.73     6.682327                4.750550
60 2020-01-01       6.62     6.060997                4.137554
```

```
future_dates = pd.date_range(start='2020-01-01', periods=20, freq='Y')
future_inflation = result_sarimax.predict(start=len(data_inflation),
end=len(data_inflation) + 19)


# Make predictions for the next 20 periods

# Create a new DataFrame with the predicted values
future_inflation_df = pd.DataFrame({'Year': future_dates, 'Inflation':
future_inflation})

# Print the predicted values
print(future_inflation_df)
```

```
         Year  Inflation
61 2020-12-31   7.943361
62 2021-12-31  10.541147
63 2022-12-31   4.893617
64 2023-12-31   4.390388
65 2024-12-31   5.855062
66 2025-12-31   5.022716
67 2026-12-31   6.314329
68 2027-12-31   5.280781
69 2028-12-31   3.953922
70 2029-12-31   4.878318
71 2030-12-31   5.057955
72 2031-12-31   5.312525
73 2032-12-31   6.578842
74 2033-12-31  10.047939
75 2034-12-31   4.716861
76 2035-12-31   4.324986
77 2036-12-31   5.827822
78 2037-12-31   5.010315
79 2038-12-31   6.305758
80 2039-12-31   5.274944
```

```
index_future_years = pd.date_range(start='2020-01-01' , end='2040-01-
01', freq='Y')

pred=result.predict(start=len(data_inflation),end=len(data_inflation)
+19,type='levels').rename('ARIMA PREDICTIONS')
```

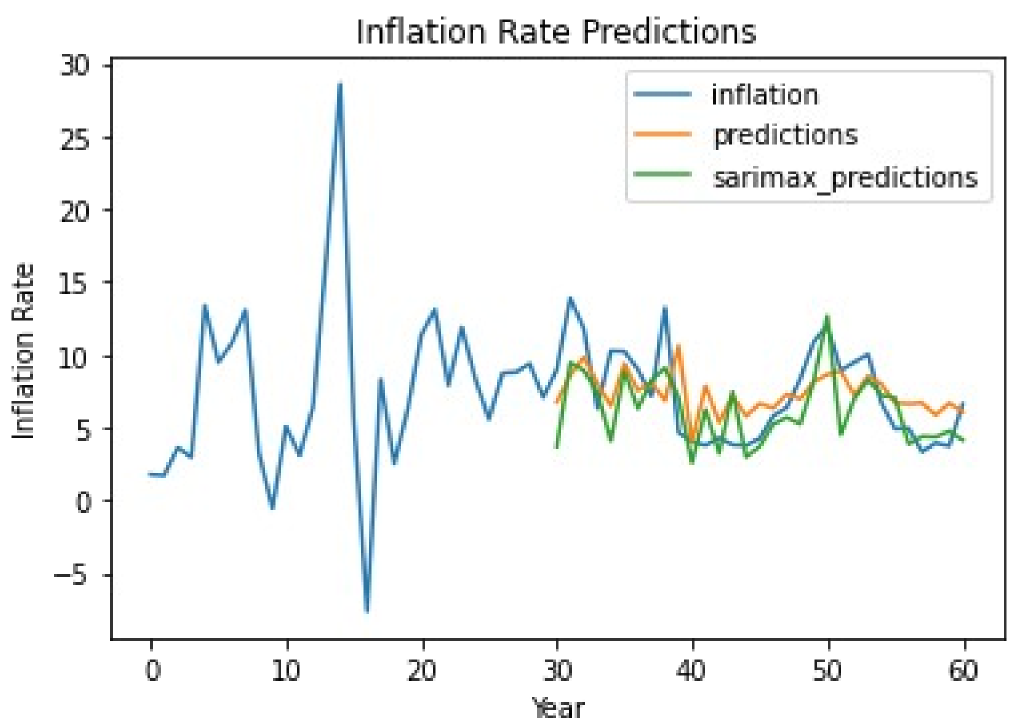
```
pred.index = index_future_years
future_inflation_df = pd.DataFrame({'Year': future_dates, 'Inflation':
future_inflation})


print(pred)
```

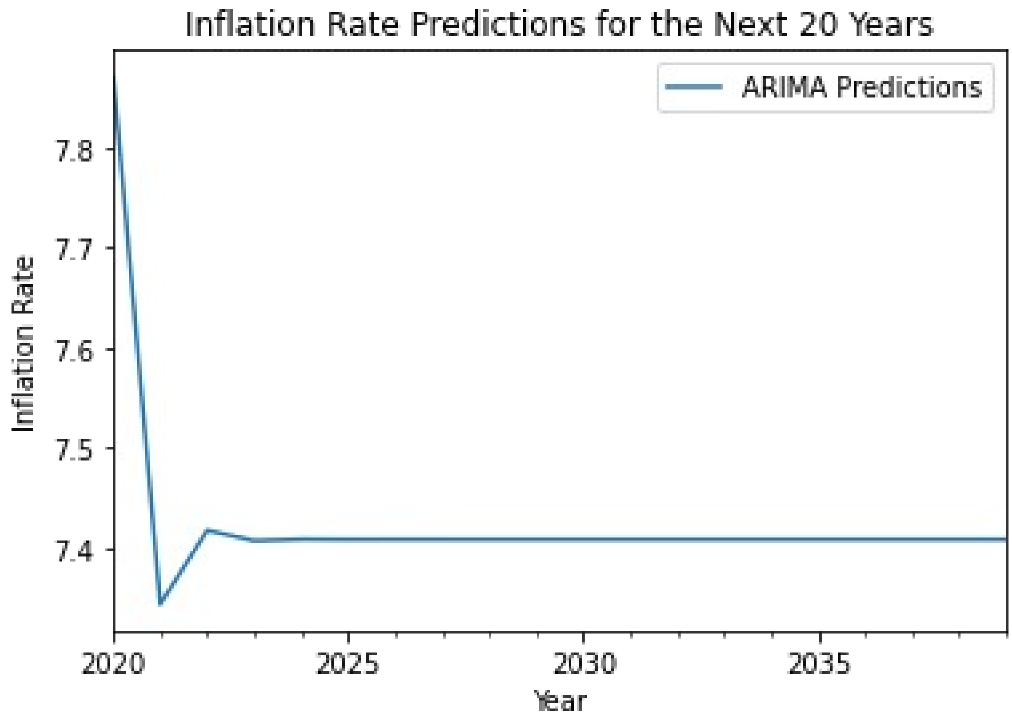
```
2020-12-31    7.870936
2021-12-31    7.343406
2022-12-31    7.417212
2023-12-31    7.406886
2024-12-31    7.408331
2025-12-31    7.408129
2026-12-31    7.408157
2027-12-31    7.408153
2028-12-31    7.408154
2029-12-31    7.408153
2030-12-31    7.408153
2031-12-31    7.408153
2032-12-31    7.408153
2033-12-31    7.408153
2034-12-31    7.408153
2035-12-31    7.408153
2036-12-31    7.408153
2037-12-31    7.408153
2038-12-31    7.408153
2039-12-31    7.408153
Freq: A-DEC, Name: ARIMA PREDICTIONS, dtype: float64
```

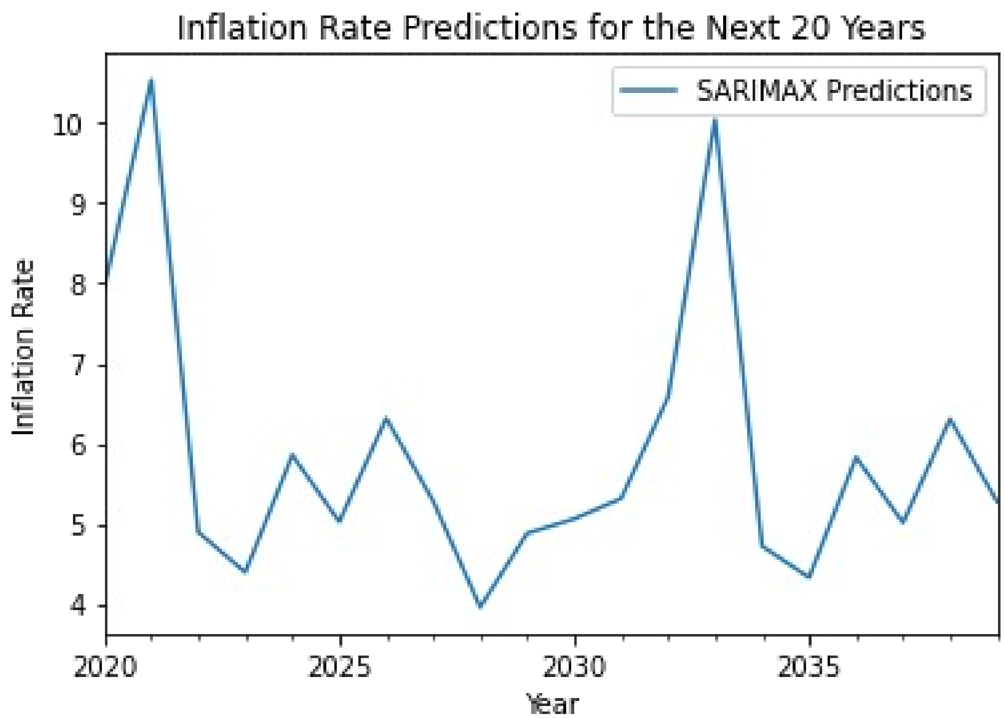
```
C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\statespace\representation.py:374:
FutureWarning: Unknown keyword arguments: dict_keys(['type']).Passing
unknown keyword arguments will raise a TypeError beginning in version
0.15.
  warnings.warn(msg, FutureWarning)
```

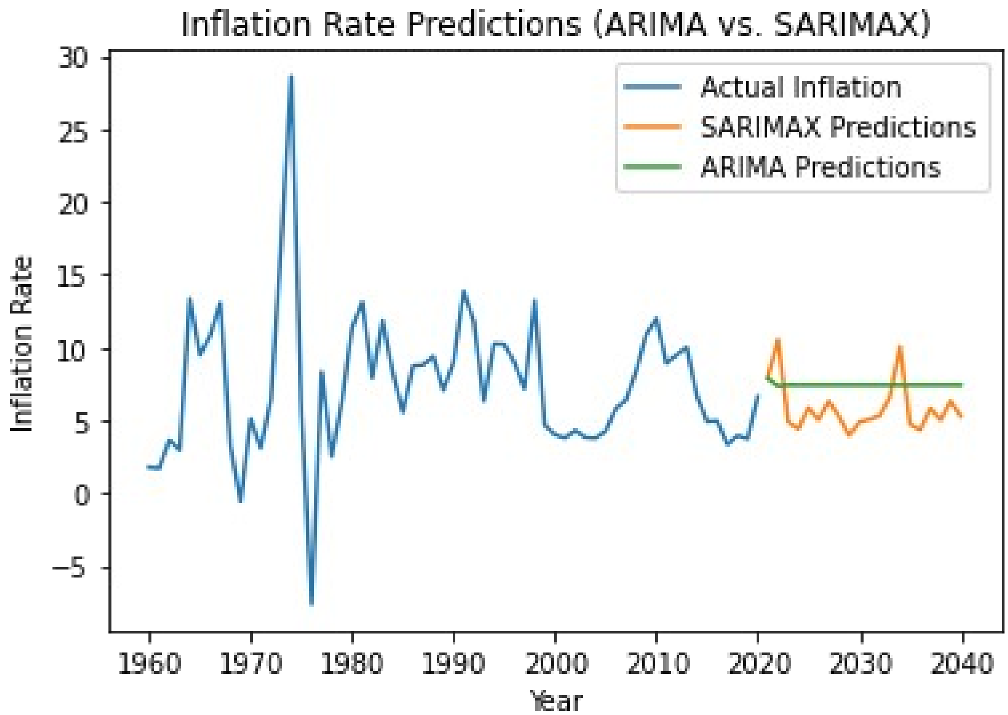
```
# prompt: now plot both these arima and sarimax predictions

import matplotlib.pyplot as plt
data_inflation[['inflation', 'predictions',
'sarimax_predictions']].plot()
plt.title('Inflation Rate Predictions')
plt.xlabel('Year')
plt.ylabel('Inflation Rate')
plt.legend()
plt.show()


pred.plot(x='Year', y='Inflation', label='ARIMA Predictions')
plt.title('Inflation Rate Predictions for the Next 20 Years')
```

```
plt.xlabel('Year')
plt.ylabel('Inflation Rate')
plt.legend()
plt.show()


# Plot the predicted values for the next 20 periods
future_inflation_df.plot(x='Year', y='Inflation', label='SARIMAX
Predictions')
plt.title('Inflation Rate Predictions for the Next 20 Years')
plt.xlabel('Year')
plt.ylabel('Inflation Rate')
plt.legend()
plt.show()

# Plot both ARIMA and SARIMAX predictions together
plt.plot(data_inflation['Year'], data_inflation['inflation'],
label='Actual Inflation')
plt.plot(future_dates, future_inflation, label='SARIMAX Predictions')
plt.plot(pred.index, pred, label='ARIMA Predictions')
plt.title('Inflation Rate Predictions (ARIMA vs. SARIMAX)')
plt.xlabel('Year')
plt.ylabel('Inflation Rate')
plt.legend()
plt.show()
```



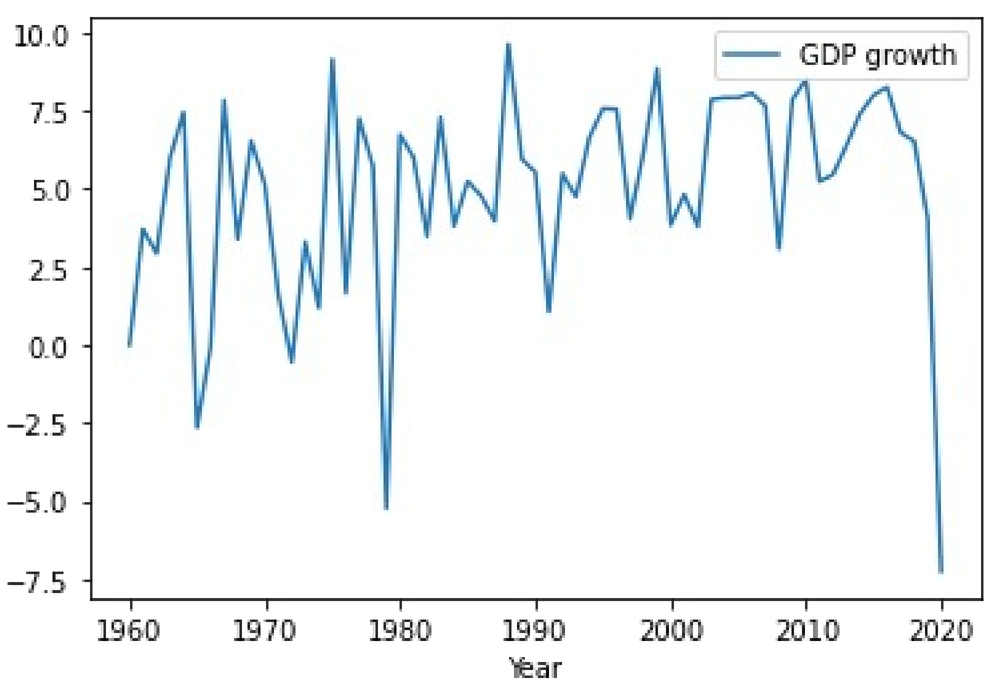

Inflation Rate Predictions for the Next 20 Years
ARIMA Predictions
Inflation Rate
7.8
7.7
7.6
7.5
7.4
2020
2025
2030
2035
Year


Inflation Rate Predictions for the Next 20 Years
SARIMAX Predictions
Inflation Rate
10
9
8
7
6
5
4
2020
2025
2030
2035
Year




```
data.columns

Index(['Year', 'GDP (current US$) ', ' GDP per capita (current US$) ',
       'GDP growth (annual %)', 'Imports of goods and services (% of
GDP)',
       'Exports of goods and services (% of GDP)',
       ' Total reserves (includes gold, current US$) ',
       'Inflation, consumer prices (annual %)', 'Population, total',
       'Population growth (annual %)',
       'Life expectancy at birth, total (years)'],
      dtype='object')

# prompt: now remove all columns and take only two colums, year and
inflation

data_growth = data[['Year', 'GDP growth (annual %)']]
data_growth

    Year  GDP growth (annual %)
0   1960                   0.00
1   1961                   3.72
2   1962                   2.93
3   1963                   5.99
4   1964                   7.45
..   ...                    ...
56  2016                   8.26
57  2017                   6.80
58  2018                   6.53
```

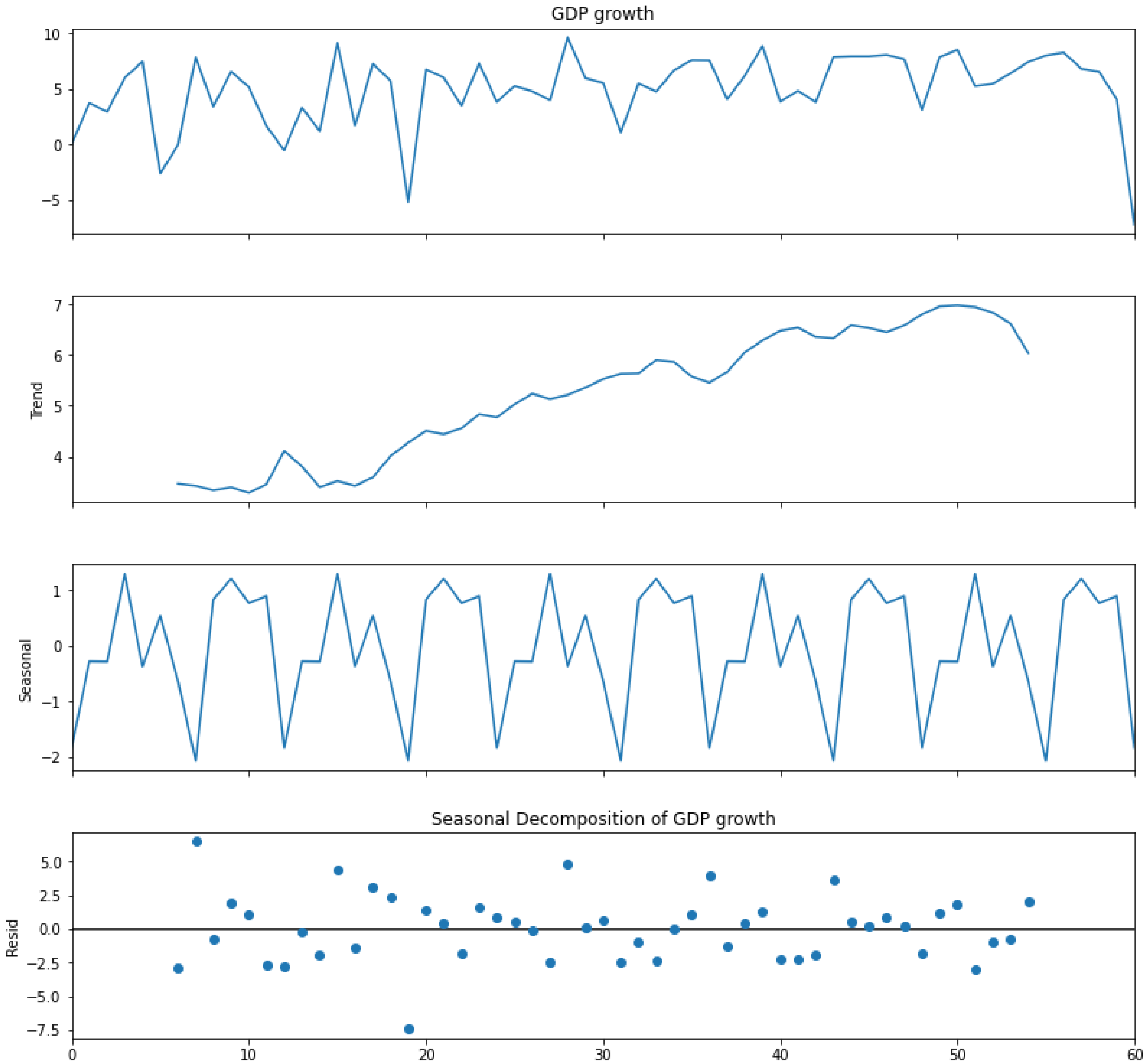
```
59  2019                         4.04
60  2020                        -7.25

[61 rows x 2 columns]

data_growth = data_growth.rename(columns={'GDP growth (annual %)':'GDP growth'})
data_growth

    Year  GDP growth
0   1960        0.00
1   1961        3.72
2   1962        2.93
3   1963        5.99
4   1964        7.45
..   ...         ...
56  2016        8.26
57  2017        6.80
58  2018        6.53
59  2019        4.04
60  2020       -7.25

[61 rows x 2 columns]

data_growth.plot(x="Year",y="GDP growth")

<Axes: xlabel='Year'>
```



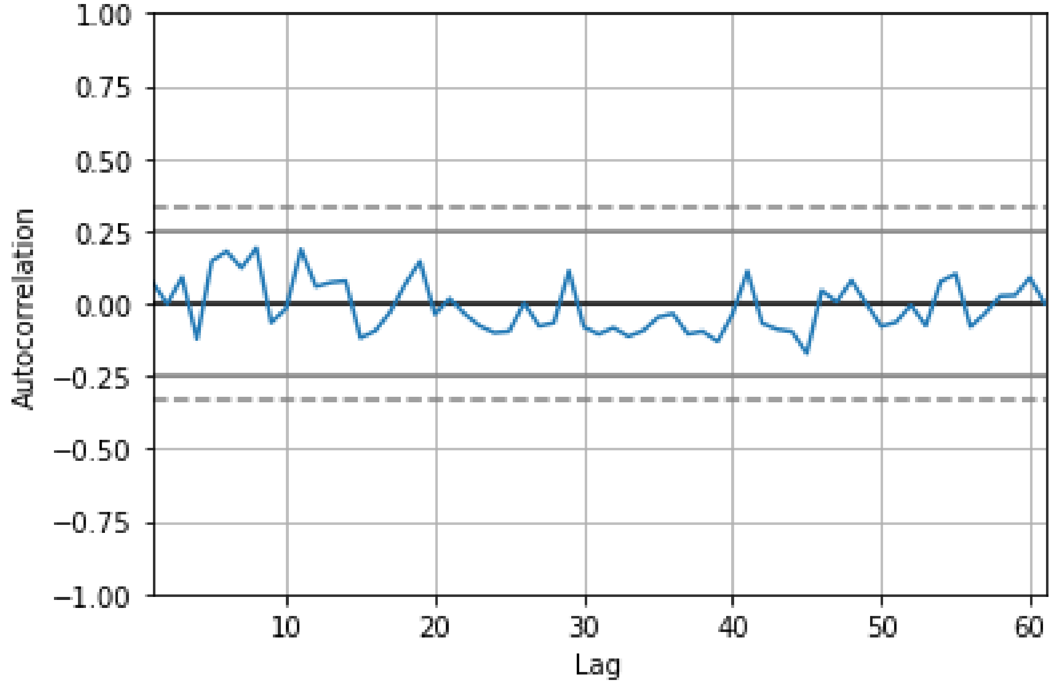
```
# prompt: now do the adfuller test and show whether it is stationary
or not

from statsmodels.tsa.stattools import adfuller

adf_result = adfuller(data_growth['GDP growth'])
print("ADF Statistic:", adf_result[0])
print("p-value:", adf_result[1])

if adf_result[1] > 0.05:
  print("The GDP per capita rate is not stationary.")
else:
  print("The GDP per capita rate is stationary.")
```

```
ADF Statistic: -6.203435900996275
p-value: 5.733303892352508e-08
The GDP per capita rate is stationary.
```

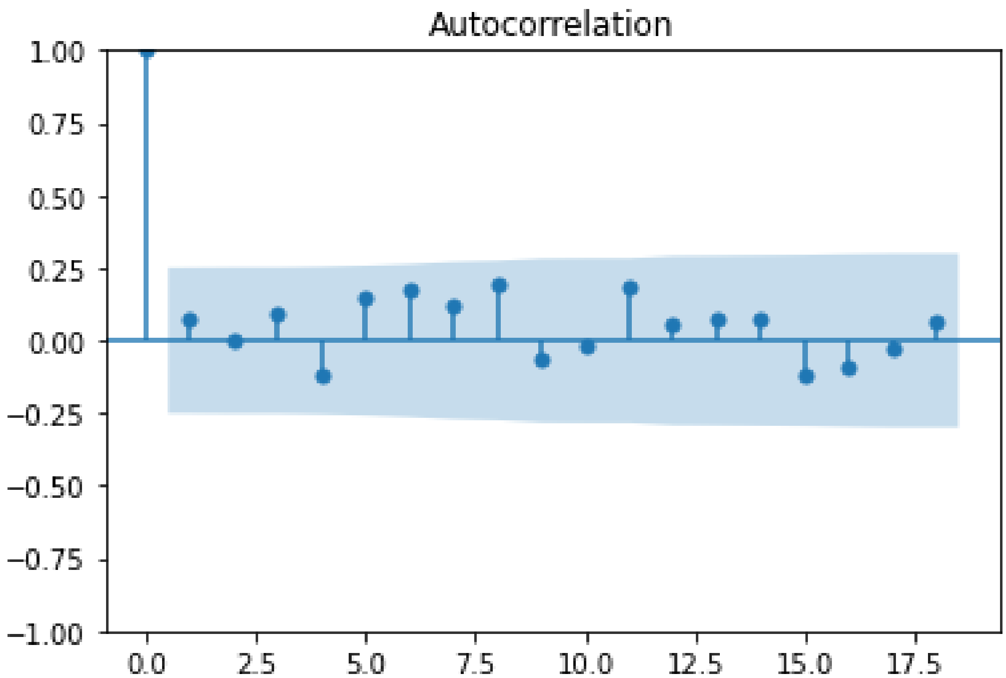
```
result = seasonal_decompose(data_growth['GDP growth'],
model='additive', period=12)

# Create the seasonal decomposition plot
fig = result.plot()  # Default size

# Optional: Adjust figure size
fig.set_size_inches(12, 12)  # Set your desired figure size

# Optional: Add title
plt.title("Seasonal Decomposition of GDP growth")

plt.show()
```

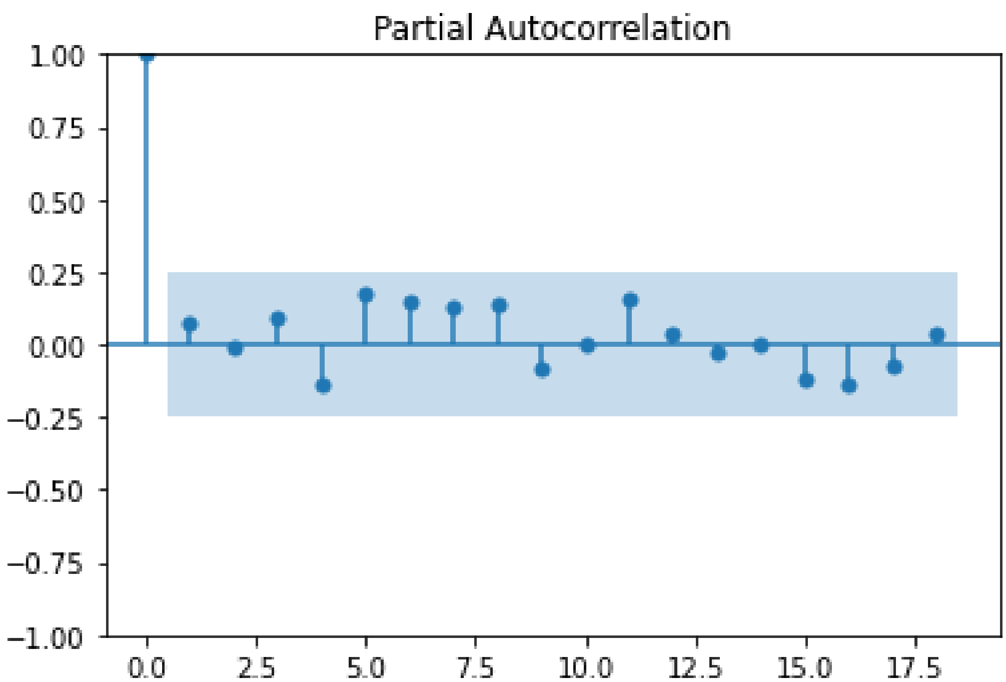



```
from pandas.plotting import autocorrelation_plot
autocorrelation_plot(data_growth['GDP growth'])
plt.show()
```




```
# prompt: import plot_acf  and plot_pacf for acf and pacf

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
fig = plt.figure(figsize=(12,8))

# Plot the ACF of the inflation rate
plot_acf(data_growth['GDP growth'])
plt.show()

# Plot the PACF of the inflation rate
plot_pacf(data_growth['GDP growth'])
plt.show()

<Figure size 864x576 with 0 Axes>
```






```
!pip install pmdarima

from pmdarima import auto_arima
from statsmodels.tsa.seasonal import seasonal_decompose  # Optional
for seasonal check
import matplotlib.pyplot as plt  # Optional for plotting
```

```
Requirement already satisfied: pmdarima in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (2.0.4)

WARNING: Ignoring invalid distribution -lotly (c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -lotly (c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages)

[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip


Requirement already satisfied: joblib>=0.11 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pmdarima) (1.2.0)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pmdarima) (3.0.10)
Requirement already satisfied: numpy>=1.21.2 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pmdarima) (1.23.5)
Requirement already satisfied: pandas>=0.19 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pmdarima) (2.0.3)
Requirement already satisfied: scikit-learn>=0.22 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pmdarima) (1.2.2)
Requirement already satisfied: scipy>=1.3.2 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pmdarima) (1.10.1)
Requirement already satisfied: statsmodels>=0.13.2 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pmdarima) (0.14.0)
Requirement already satisfied: urllib3 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pmdarima) (1.26.7)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pmdarima) (57.4.0)
Requirement already satisfied: packaging>=17.1 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pmdarima) (23.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from pandas>=0.19->pmdarima) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in c:\users\lenovo\appdata\local\programs\python\python310\lib\site-packages (from
```

```
pandas>=0.19->pmdarima) (2023.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\
lenovo\appdata\local\programs\python\python310\lib\site-packages (from
scikit-learn>=0.22->pmdarima) (3.1.0)
Requirement already satisfied: patsy>=0.5.2 in c:\users\lenovo\
appdata\local\programs\python\python310\lib\site-packages (from
statsmodels>=0.13.2->pmdarima) (0.5.3)
Requirement already satisfied: six in c:\users\lenovo\appdata\local\
programs\python\python310\lib\site-packages (from patsy>=0.5.2-
>statsmodels>=0.13.2->pmdarima) (1.16.0)

model = auto_arima(data_growth['GDP growth'], trace=True,
error_action='ignore')  # Adjust as needed

# Print the summary of the fitted model
print(model.summary())

Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.55 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=347.002, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=337.844, Time=0.04 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.18 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=345.051, Time=0.03 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=332.082, Time=0.11 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=334.063, Time=0.17 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.50 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.30 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.53 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=330.125, Time=0.06 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=335.902, Time=0.04 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=332.105, Time=0.08 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=323.491, Time=0.13 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=322.276, Time=0.08 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=320.280, Time=0.05 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=322.275, Time=0.08 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=322.761, Time=0.23 sec

Best model:  ARIMA(0,1,1)(0,0,0)[0]
Total fit time: 3.234 seconds
                               SARIMAX Results

======================================================================
========
Dep. Variable:                      y   No. Observations:
61
Model:               SARIMAX(0, 1, 1)   Log Likelihood
-158.140
Date:                Sat, 04 May 2024   AIC
320.280
Time:                        14:20:20   BIC
```

```
324.468
Sample:                                 0   HQIC
321.918
                                     - 61

Covariance Type:                      opg

======================================================================
========
                  coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
--------
ma.L1          -0.9042      0.108     -8.409      0.000      -1.115
-0.693
sigma2         11.0794      1.471      7.531      0.000       8.196
13.963
======================================================================
=============
Ljung-Box (L1) (Q):                   0.05   Jarque-Bera (JB):
82.78
Prob(Q):                              0.82   Prob(JB):
0.00
Heteroskedasticity (H):               0.81   Skew:
-1.67
Prob(H) (two-sided):                  0.63   Kurtosis:
7.69
======================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```

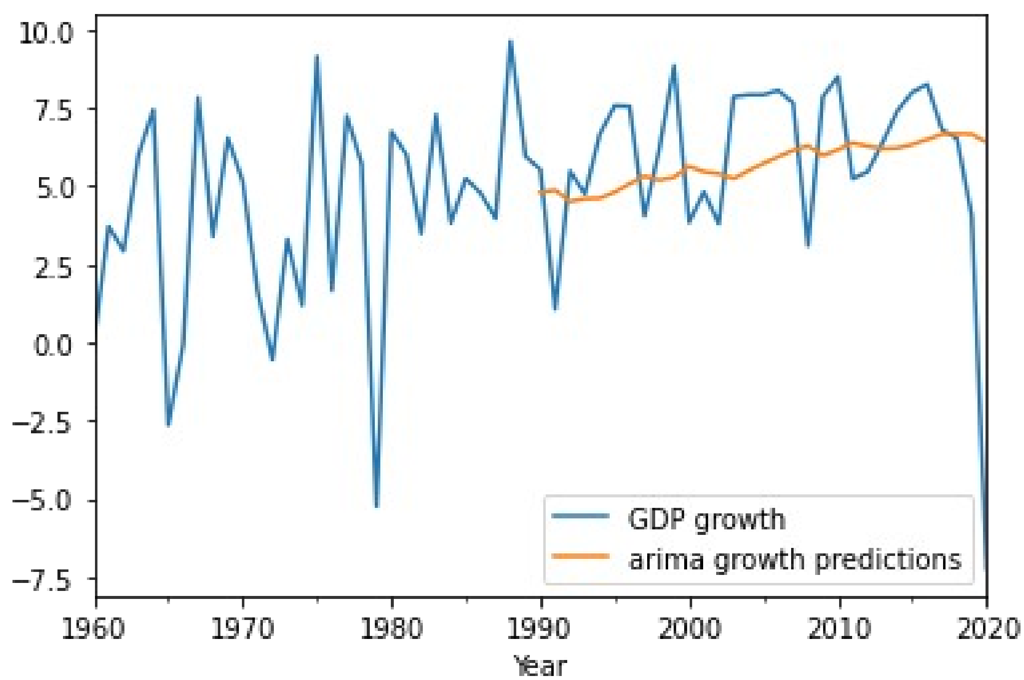
```python
model2 = ARIMA(data_growth['GDP growth'], order=(0, 1, 1))

# Fit the model
model2_fit = model2.fit()

# Print the summary of the fitted model
print(model2_fit.summary())
```

```
                               SARIMAX Results

======================================================================
========
Dep. Variable:             GDP growth   No. Observations:
61
Model:                 ARIMA(0, 1, 1)   Log Likelihood
-158.140
```

```
Date:                     Sat, 04 May 2024   AIC
320.280
Time:                             15:12:56   BIC
324.468
Sample:                                  0   HQIC
321.918
                                      - 61

Covariance Type:                       opg

======================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
--------
ma.L1         -0.9042      0.108     -8.409      0.000      -1.115
-0.693
sigma2        11.0794      1.471      7.531      0.000       8.196
13.963
======================================================================
=============
Ljung-Box (L1) (Q):                   0.05   Jarque-Bera (JB):
82.78
Prob(Q):                              0.82   Prob(JB):
0.00
Heteroskedasticity (H):               0.81   Skew:
-1.67
Prob(H) (two-sided):                  0.63   Kurtosis:
7.69
======================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).


# Make predictions
data_growth['arima growth predictions'] =
model2_fit .predict( start=30,end=60,dynamics=True)

data_growth

         Year  GDP growth  arima growth predictions  \
0  1960-01-01        0.00                       NaN
1  1961-01-01        3.72                       NaN
2  1962-01-01        2.93                       NaN
3  1963-01-01        5.99                       NaN
4  1964-01-01        7.45                       NaN
```

```
..         ...         ...                        ...
56 2016-01-01        8.26                   6.489604
57 2017-01-01        6.80                   6.659260
58 2018-01-01        6.53                   6.672747
59 2019-01-01        4.04                   6.659068
60 2020-01-01       -7.25                   6.408087

    sarimax_growth_predictions
0                          NaN
1                          NaN
2                          NaN
3                          NaN
4                          NaN
..                         ...
56                    8.424116
57                    8.834020
58                    8.215690
59                    8.290546
60                    5.158883

[61 rows x 4 columns]
```

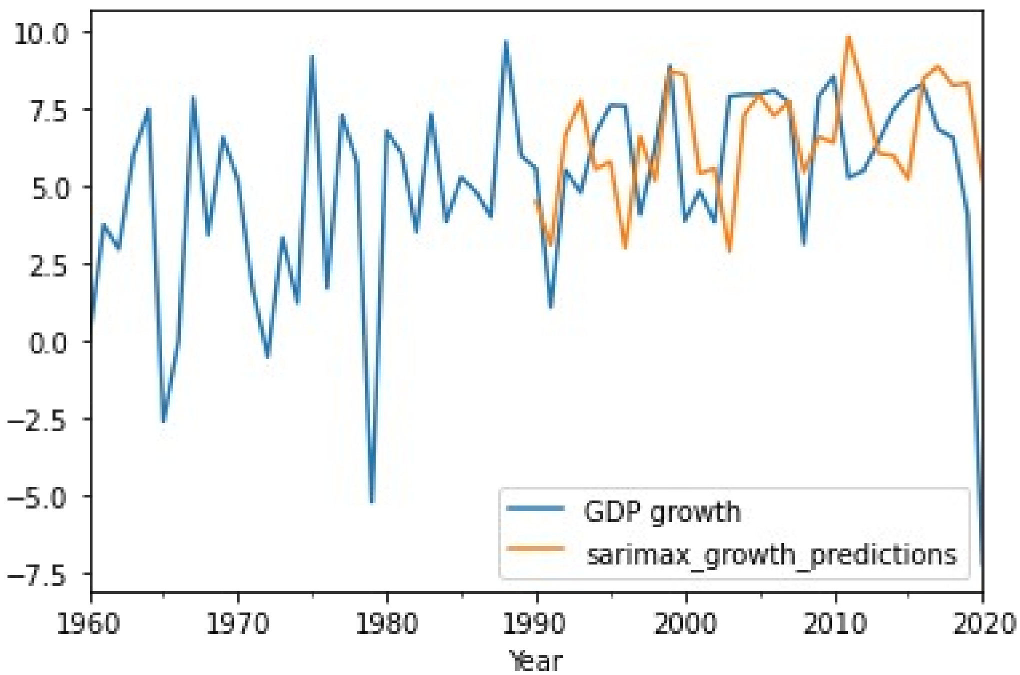
```
data_growth.plot(x='Year', y=['GDP growth', 'arima growth
predictions'])
plt.show()
```




```
# prompt: now use sarimax
```

```
model3 = sm.tsa.statespace.SARIMAX(data_growth['GDP growth'],
order=(0, 1, 1), seasonal_order=(0, 1, 1, 12))
result_sarimax = model3.fit()
result_sarimax.summary()

data_growth['sarimax_growth_predictions'] =
result_sarimax.predict(start=30,end=60)

data_growth.plot(x='Year', y=['GDP growth',
'sarimax_growth_predictions'])
plt.show()
```

```
C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\statespace\sarimax.py:1009: UserWarning: Non-
invertible starting seasonal moving average Using zeros as starting
parameters.
  warn('Non-invertible starting seasonal moving average'
```




```
data_growth
```

```
          Year  GDP growth  arima growth predictions  \
0  1960-01-01        0.00                       NaN
1  1961-01-01        3.72                       NaN
2  1962-01-01        2.93                       NaN
3  1963-01-01        5.99                       NaN
4  1964-01-01        7.45                       NaN
..         ...         ...                       ...
56 2016-01-01        8.26                  6.489604
```

```
57 2017-01-01       6.80                    6.659260
58 2018-01-01       6.53                    6.672747
59 2019-01-01       4.04                    6.659068
60 2020-01-01      -7.25                    6.408087

    sarimax_growth_predictions
0                          NaN
1                          NaN
2                          NaN
3                          NaN
4                          NaN
..                         ...
56                    8.424116
57                    8.834020
58                    8.215690
59                    8.290546
60                    5.158883

[61 rows x 4 columns]
```

```
# prompt: check the models with accuracy check
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error

# Calculate the mean absolute error (MAE) for both models
mae_arima = mean_absolute_error(data_growth['GDP growth'].iloc[-31:],
data_growth['arima growth predictions'].dropna())
mae_sarimax = mean_absolute_error(data_growth['GDP growth'].iloc[-
31:], data_growth['sarimax_growth_predictions'].dropna())

# Calculate the mean absolute percentage error (MAPE) for both models
mape_arima = mean_absolute_error(data_growth['GDP growth'].iloc[-31:],
data_growth['arima growth predictions'].dropna())
mape_sarimax = mean_absolute_error(data_growth['GDP growth'].iloc[-
31:], data_growth['sarimax_growth_predictions'].dropna())


rmse = mean_squared_error(data_growth['GDP growth'].iloc[-31:],
data_growth['arima growth predictions'].dropna())
sarimax_rmse = mean_squared_error(data_growth['GDP growth'].iloc[-
31:], data_growth['sarimax_growth_predictions'].dropna())


# Print the accuracy metrics for both models
print("ARIMA Model:")
print(f"  - RMSE: {rmse}")
print(f"  - MAE: {mae_arima}")
print(f"  - MAPE: {mape_arima}")

print("SARIMAX Model:")
```

```
print(f"  - RMSE: {sarimax_rmse}")
print(f"  - MAE: {mae_sarimax}")
print(f"  - MAPE: {mape_sarimax}")

ARIMA Model:
  - RMSE: 9.734222683828998
  - MAE: 2.077967505069251
  - MAPE: 2.077967505069251
SARIMAX Model:
  - RMSE: 10.583318221885921
  - MAE: 2.259842024797362
  - MAPE: 2.259842024797362

data_growth['Year'] = pd.to_datetime(data_growth['Year'], format='%Y')
future_dates = pd.date_range(start='2020-01-01', periods=20, freq='Y')
future_growth = result_sarimax.predict(start=len(data_growth),
end=len(data_growth) + 19)

future_growth_df = pd.DataFrame({'Year': future_dates, 'GDP growth':
future_growth})

# Print the predicted values
print(future_growth_df)

         Year  GDP growth
61 2020-12-31    6.115908
62 2021-12-31    5.995929
63 2022-12-31    7.919895
64 2023-12-31    6.887901
65 2024-12-31    5.633922
66 2025-12-31    5.757917
67 2026-12-31    5.181933
68 2027-12-31    7.637912
69 2028-12-31    7.683906
70 2029-12-31    7.257912
71 2030-12-31    6.919890
72 2031-12-31    2.697824
73 2032-12-31    6.637163
74 2033-12-31    6.517184
75 2034-12-31    8.441149
76 2035-12-31    7.409156
77 2036-12-31    6.155176
78 2037-12-31    6.279171
79 2038-12-31    5.703187
80 2039-12-31    8.159166

index_future_years = pd.date_range(start='2020-01-01' , end='2040-01-
01', freq='Y')

pred=model2_fit.predict(start=len(data_growth),end=len(data_growth)
```

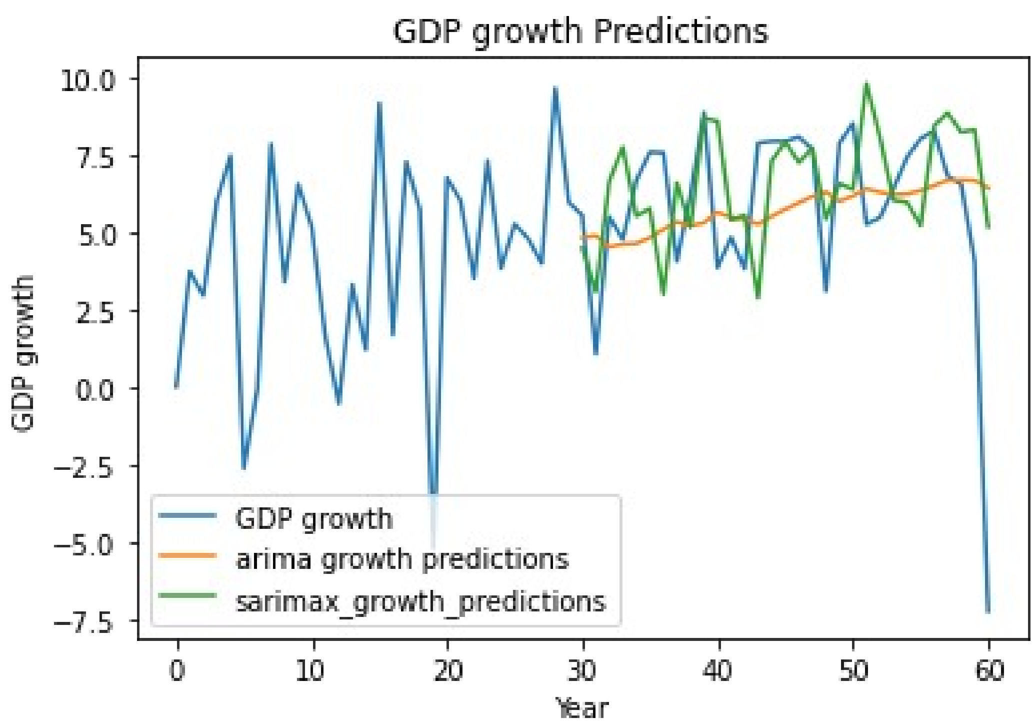
```
+19,type='levels').rename('ARIMA GROWTH PREDICTIONS')

pred.index = index_future_years
future_growth_df = pd.DataFrame({'Year': future_dates, 'GDP growth':
future_growth})


pred
```

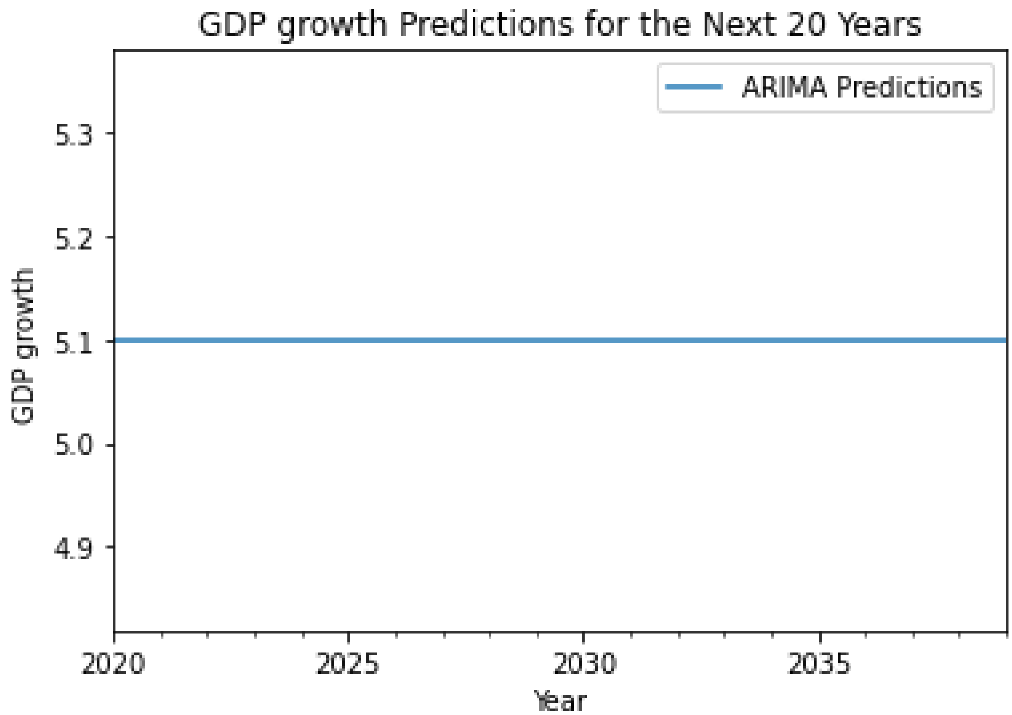
```
C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\lib\site-
packages\statsmodels\tsa\statespace\representation.py:374:
FutureWarning: Unknown keyword arguments: dict_keys(['type']).Passing
unknown keyword arguments will raise a TypeError beginning in version
0.15.
  warnings.warn(msg, FutureWarning)
```

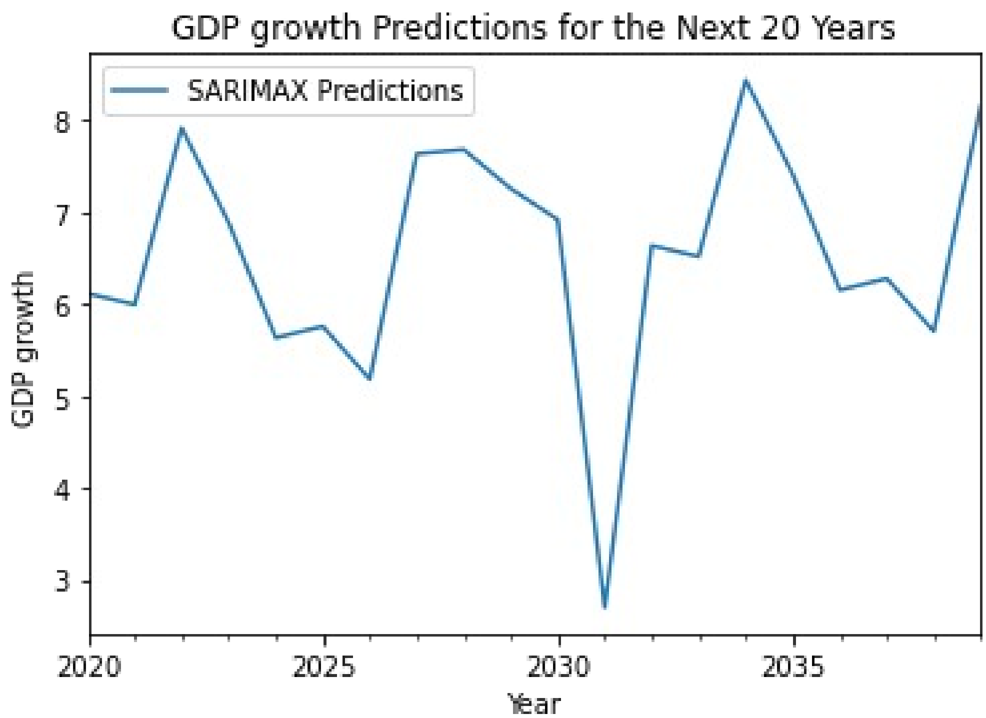
```
2020-12-31    5.099256
2021-12-31    5.099256
2022-12-31    5.099256
2023-12-31    5.099256
2024-12-31    5.099256
2025-12-31    5.099256
2026-12-31    5.099256
2027-12-31    5.099256
2028-12-31    5.099256
2029-12-31    5.099256
2030-12-31    5.099256
2031-12-31    5.099256
2032-12-31    5.099256
2033-12-31    5.099256
2034-12-31    5.099256
2035-12-31    5.099256
2036-12-31    5.099256
2037-12-31    5.099256
2038-12-31    5.099256
2039-12-31    5.099256
Freq: A-DEC, Name: ARIMA GROWTH PREDICTIONS, dtype: float64
```

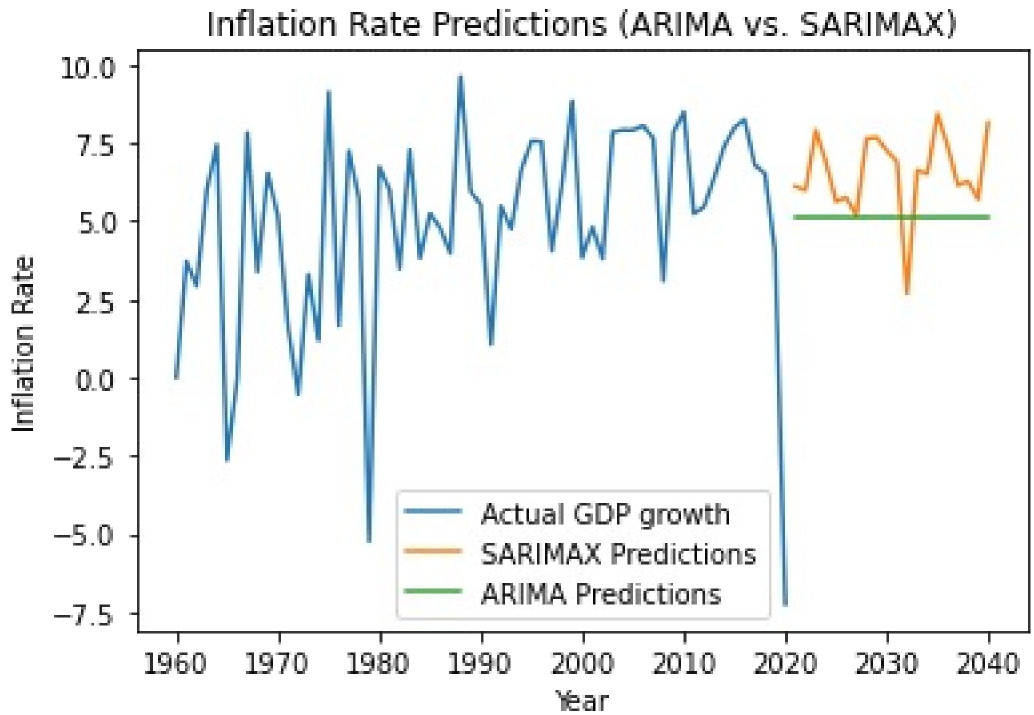
```
data_growth[['GDP growth', 'arima growth predictions',
'sarimax_growth_predictions']].plot()
plt.title('GDP growth Predictions')
plt.xlabel('Year')
plt.ylabel('GDP growth')
plt.legend()
plt.show()


pred.plot(x='Year', y='GDP growth', label='ARIMA Predictions')
plt.title('GDP growth Predictions for the Next 20 Years')
plt.xlabel('Year')
```

```
plt.ylabel('GDP growth')
plt.legend()
plt.show()


# Plot the predicted values for the next 20 periods
future_growth_df.plot(x='Year', y='GDP growth', label='SARIMAX
Predictions')
plt.title('GDP growth Predictions for the Next 20 Years')
plt.xlabel('Year')
plt.ylabel('GDP growth')
plt.legend()
plt.show()

# Plot both ARIMA and SARIMAX predictions together
plt.plot(data_growth['Year'], data_growth['GDP growth'], label='Actual
GDP growth')
plt.plot(future_dates, future_growth, label='SARIMAX Predictions')
plt.plot(pred.index, pred, label='ARIMA Predictions')
plt.title('Inflation Rate Predictions (ARIMA vs. SARIMAX)')
plt.xlabel('Year')
plt.ylabel('Inflation Rate')
plt.legend()
plt.show()
```




GDP growth Predictions for the Next 20 Years
ARIMA Predictions
5.3
5.2
5.1
5.0
4.9
GDP growth
2020
2025
2030
2035
Year


GDP growth Predictions for the Next 20 Years
SARIMAX Predictions
8
7
6
5
4
3
GDP growth
2020
2025
2030
2035
Year



Inflation Rate Predictions (ARIMA vs. SARIMAX)
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
Inflation Rate
1960
1970
1980
1990
2000
2010
2020
2030
2040
Year
Actual GDP growth
SARIMAX Predictions
ARIMA Predictions