In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\18_world-data-2023 - 18_world-data-20
df
```

Out[2]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0
3	Andorra	164	AD	40.00%	468	NaN	7.20	376.0
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0
...	...	...	...	...	...	...	...	...
190	Venezuela	32	VE	24.50%	912,050	343,000	17.88	58.0
191	Vietnam	314	VN	39.30%	331,210	522,000	16.75	84.0
192	Yemen	56	YE	44.60%	527,968	40,000	30.45	967.0
193	Zambia	25	ZM	32.10%	752,618	16,000	36.19	260.0
194	Zimbabwe	38	ZW	41.90%	390,757	51,000	30.68	263.0

195 rows × 35 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 35 columns):
 #   Column                                   Non-Null Count  Dtype  
---  ------                                   --------------  -----  
 0   Country                                  195 non-null    object 
 1   Density
(P/Km2)                          195 non-null    object 
 2   Abbreviation                             188 non-null    object 
 3   Agricultural Land( %)                    188 non-null    object 
 4   Land Area(Km2)                           194 non-null    object 
 5   Armed Forces size                        171 non-null    object 
 6   Birth Rate                               189 non-null    float64
 7   Calling Code                             194 non-null    float64
 8   Capital/Major City                       192 non-null    object 
 9   Co2-Emissions                            188 non-null    object 
 10  CPI                                      178 non-null    object 
 11  CPI Change (%)                           179 non-null    object 
 12  Currency-Code                            180 non-null    object 
 13  Fertility Rate                           188 non-null    float64
 14  Forested Area (%)                        188 non-null    object 
 15  Gasoline Price                           175 non-null    object 
 16  GDP                                      193 non-null    object 
 17  Gross primary education enrollment (%)   188 non-null    object 
 18  Gross tertiary education enrollment (%)  183 non-null    object 
 19  Infant mortality                         189 non-null    float64
 20  Largest city                             189 non-null    object 
 21  Life expectancy                          187 non-null    float64
 22  Maternal mortality ratio                 181 non-null    float64
 23  Minimum wage                             150 non-null    object 
 24  Official language                        194 non-null    object 
 25  Out of pocket health expenditure         188 non-null    object 
 26  Physicians per thousand                  188 non-null    float64
 27  Population                               194 non-null    object 
 28  Population: Labor force participation (%)  176 non-null    object 
 29  Tax revenue (%)                          169 non-null    object 
 30  Total tax rate                           183 non-null    object 
 31  Unemployment rate                        176 non-null    object 
 32  Urban_population                         190 non-null    object 
 33  Latitude                                 194 non-null    float64
 34  Longitude                                194 non-null    float64
dtypes: float64(9), object(26)
memory usage: 53.4+ KB
```

In [4]:
```
df.describe()
```

Out[4]:

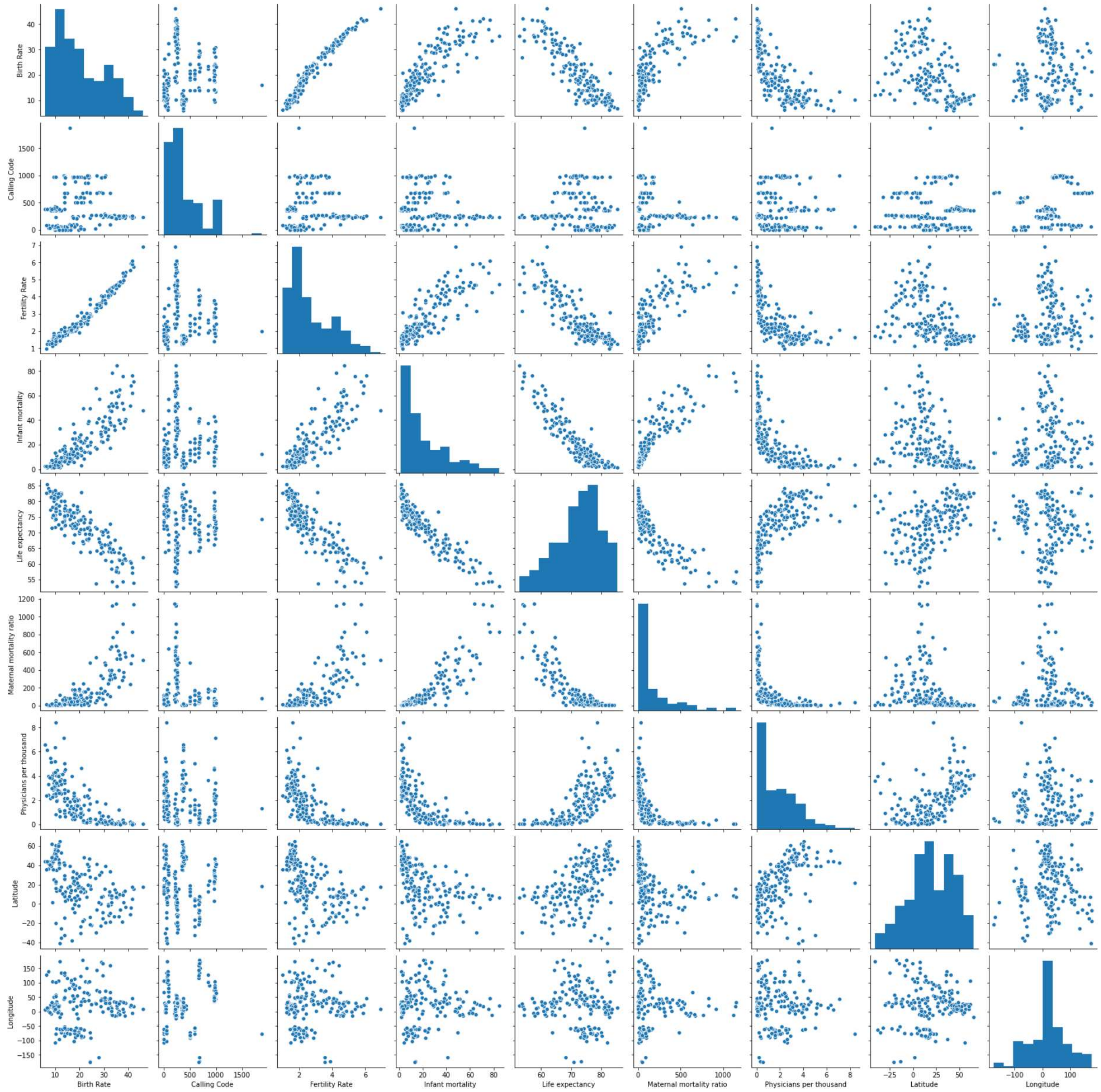
	Birth Rate	Calling Code	Fertility Rate	Infant mortality	Life expectancy	Maternal mortality ratio	Physicians per thousand	
count	189.000000	194.000000	188.000000	189.000000	187.000000	181.000000	188.000000	19
mean	20.214974	360.546392	2.698138	21.332804	72.279679	160.392265	1.839840	1
std	9.945774	323.236419	1.282267	19.548058	7.483661	233.502024	1.684261	2
min	5.900000	1.000000	0.980000	1.400000	52.800000	2.000000	0.010000	-4
25%	11.300000	82.500000	1.705000	6.000000	67.000000	13.000000	0.332500	
50%	17.950000	255.500000	2.245000	14.000000	73.200000	53.000000	1.460000	1
75%	28.750000	506.750000	3.597500	32.700000	77.500000	186.000000	2.935000	4
max	46.080000	1876.000000	6.910000	84.500000	85.400000	1150.000000	8.420000	6

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Country', 'Density\n(P/Km2)', 'Abbreviation', 'Agricultural Land(
%)',
       'Land Area(Km2)', 'Armed Forces size', 'Birth Rate', 'Calling Code',
       'Capital/Major City', 'Co2-Emissions', 'CPI', 'CPI Change (%)',
       'Currency-Code', 'Fertility Rate', 'Forested Area (%)',
       'Gasoline Price', 'GDP', 'Gross primary education enrollment (%)',
       'Gross tertiary education enrollment (%)', 'Infant mortality',
       'Largest city', 'Life expectancy', 'Maternal mortality ratio',
       'Minimum wage', 'Official language', 'Out of pocket health expenditur
e',
       'Physicians per thousand', 'Population',
       'Population: Labor force participation (%)', 'Tax revenue (%)',
       'Total tax rate', 'Unemployment rate', 'Urban_population', 'Latitude',
       'Longitude'],
      dtype='object')
```

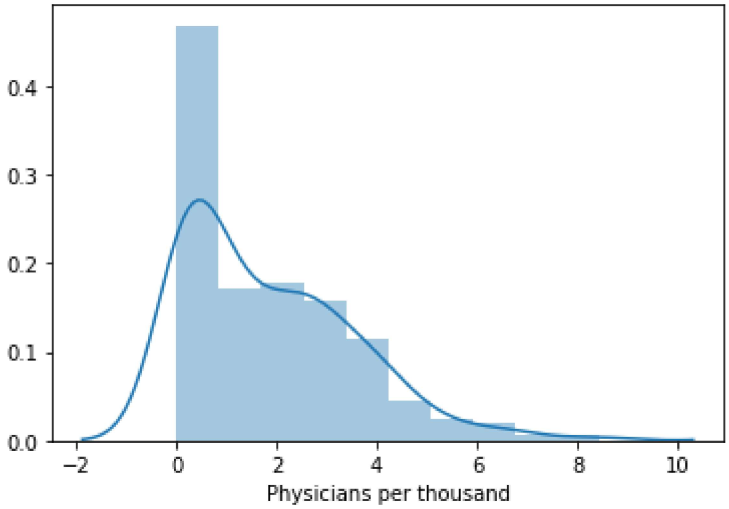
In [6]:
```
sns.pairplot(df)
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x216470c3df0>



In [7]:
```
sns.distplot(df['Physicians per thousand'])
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x2164a4a93d0>



In [8]:
```
df1=df[['Birth Rate','Calling Code','Fertility Rate','Infant mortality','Life e
df1
```

Out[8]:

	Birth Rate	Calling Code	Fertility Rate	Infant mortality	Life expectancy	Maternal mortality ratio	Physicians per thousand	Latitude	Longitude
0	32.49	93.0	4.47	47.9	64.5	638.0	0.28	33.939110	67.709953
1	11.78	355.0	1.62	7.8	78.5	15.0	1.20	41.153332	20.168331
2	24.28	213.0	3.02	20.1	76.7	112.0	1.72	28.033886	1.659626
3	7.20	376.0	1.27	2.7	NaN	NaN	3.33	42.506285	1.521801
4	40.73	244.0	5.52	51.6	60.8	241.0	0.21	-11.202692	17.873887
...	...	...	...	...	...	...	...	...	...
190	17.88	58.0	2.27	21.4	72.1	125.0	1.92	6.423750	-66.589730
191	16.75	84.0	2.05	16.5	75.3	43.0	0.82	14.058324	108.277199
192	30.45	967.0	3.79	42.9	66.1	164.0	0.31	15.552727	48.516388
193	36.19	260.0	4.63	40.4	63.5	213.0	1.19	-13.133897	27.849332
194	30.68	263.0	3.62	33.9	61.2	458.0	0.21	-19.015438	29.154857

195 rows × 9 columns

In [9]:
```
sns.heatmap(df1.corr())
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2164af81c70>
```

In [10]:
```
df2=df1.head(10)
```

In [11]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Birth Rate                10 non-null     float64
 1   Calling Code              10 non-null     float64
 2   Fertility Rate            10 non-null     float64
 3   Infant mortality          10 non-null     float64
 4   Life expectancy           9 non-null      float64
 5   Maternal mortality ratio  9 non-null      float64
 6   Physicians per thousand   10 non-null     float64
 7   Latitude                  10 non-null     float64
 8   Longitude                 10 non-null     float64
dtypes: float64(9)
memory usage: 848.0 bytes
```

In [12]:
```
x=df2[['Birth Rate','Calling Code','Fertility Rate','Infant mortality']]
y=df2[['Physicians per thousand']]
```

In [13]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [14]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

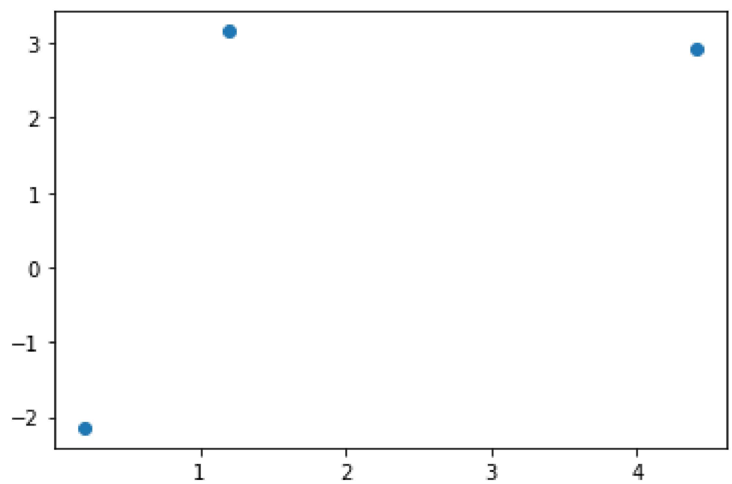
Out[14]: LinearRegression()

In [15]:
```
print(lr.intercept_)
```

[7.51406493]

In [16]:
```
prediction= lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[16]: <matplotlib.collections.PathCollection at 0x2164b3ab2e0>



In [17]:
```
print(lr.score(x_test,y_test))
```

-0.20174763826329856

In [18]:
```
print(lr.score(x_train,y_train))
```

0.8573500905913856

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]: Ridge(alpha=10)

In [21]:
```
rr.score(x_test,y_test)
```

Out[21]: 0.08119579878375216

In [22]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[22]:
```
Lasso(alpha=10)
```

In [23]:
```
la.score(x_test,y_test)
```

Out[23]:
```
0.3081204951917943
```

In [24]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[24]:
```
ElasticNet()
```

In [25]:
```
print(en.intercept_)
```

```
[5.56330459]
```

In [26]:
```
print(en.predict(x_test))
```

```
[-1.07693353  2.5165552   2.90747588]
```

In [27]:
```
print(en.score(x_test,y_test))
```

```
0.1535675887254958
```

Evaluation

In [29]:
```
from sklearn import metrics
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
Mean Absolute Error 1.9268086168097331
Mean Squared Error 3.8424144875769266
Root Mean Squared Error: 1.9602077664311317
```

In [ ]: