In [8]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [9]:
```
df = pd.read_csv("world.csv").dropna()

df
```

Out[9]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital/M
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	K
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	Ti
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	Al
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	Lu
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0	Buenos /
...	...	...	...	...	...	...	...	...	
185	United Kingdom	281	GB	71.70%	243,610	148,000	11.00	44.0	Lo
186	United States	36	US	44.40%	9,833,517	1,359,000	11.60	1.0	Washin
187	Uruguay	20	UY	82.60%	176,215	22,000	13.86	598.0	Monte
191	Vietnam	314	VN	39.30%	331,210	522,000	16.75	84.0	H
193	Zambia	25	ZM	32.10%	752,618	16,000	36.19	260.0	Lu

110 rows × 35 columns

In [10]:
```
df.head()
```

Out[10]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital/Major City
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	Kabu
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	Tirana
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	Algiers
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	Luanda
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0	Buenos Aires

5 rows × 35 columns

Data cleaning and pre processing

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 110 entries, 0 to 193
Data columns (total 35 columns):
 #   Column                                     Non-Null Count  Dtype  
---  ------                                     --------------  -----  
 0   Country                                    110 non-null    object 
 1   Density
(P/Km2)                          110 non-null    object 
 2   Abbreviation                               110 non-null    object 
 3   Agricultural Land( %)                      110 non-null    object 
 4   Land Area(Km2)                             110 non-null    object 
 5   Armed Forces size                          110 non-null    object 
 6   Birth Rate                                 110 non-null    float64
 7   Calling Code                               110 non-null    float64
 8   Capital/Major City                         110 non-null    object 
 9   Co2-Emissions                              110 non-null    object 
 10  CPI                                        110 non-null    object 
 11  CPI Change (%)                             110 non-null    object 
 12  Currency-Code                              110 non-null    object 
 13  Fertility Rate                             110 non-null    float64
 14  Forested Area (%)                          110 non-null    object 
 15  Gasoline Price                             110 non-null    object 
 16  GDP                                        110 non-null    object 
 17  Gross primary education enrollment (%)     110 non-null    object 
 18  Gross tertiary education enrollment (%)    110 non-null    object 
 19  Infant mortality                           110 non-null    float64
 20  Largest city                               110 non-null    object 
 21  Life expectancy                            110 non-null    float64
 22  Maternal mortality ratio                   110 non-null    float64
 23  Minimum wage                               110 non-null    object 
 24  Official language                          110 non-null    object 
 25  Out of pocket health expenditure           110 non-null    object 
 26  Physicians per thousand                    110 non-null    float64
 27  Population                                 110 non-null    object 
 28  Population: Labor force participation (%)  110 non-null    object 
 29  Tax revenue (%)                            110 non-null    object 
 30  Total tax rate                             110 non-null    object 
 31  Unemployment rate                          110 non-null    object 
 32  Urban_population                           110 non-null    object 
 33  Latitude                                   110 non-null    float64
 34  Longitude                                  110 non-null    float64
dtypes: float64(9), object(26)
memory usage: 30.9+ KB
```

In [12]:
```
df.describe()
```

Out[12]:

	Birth Rate	Calling Code	Fertility Rate	Infant mortality	Life expectancy	Maternal mortality ratio	Physicians per thousand	Latitude
count	110.000000	110.000000	110.000000	110.000000	110.000000	110.000000	110.000000	110.000000
mean	20.196455	344.290909	2.672182	20.271818	72.671818	137.227273	1.919182	20.362677
std	10.039056	341.231562	1.308142	18.453214	7.000788	201.171462	1.598116	24.432140
min	6.400000	1.000000	0.980000	1.700000	54.300000	2.000000	0.010000	-40.900557
25%	11.075000	70.000000	1.682500	6.100000	67.625000	15.250000	0.467500	7.623255
50%	17.830000	239.500000	2.200000	13.600000	74.400000	41.000000	1.640000	21.033608
75%	27.962500	420.750000	3.505000	31.500000	77.350000	176.000000	3.007500	40.124603
max	46.080000	1876.000000	6.910000	78.500000	83.300000	1120.000000	7.120000	61.524010

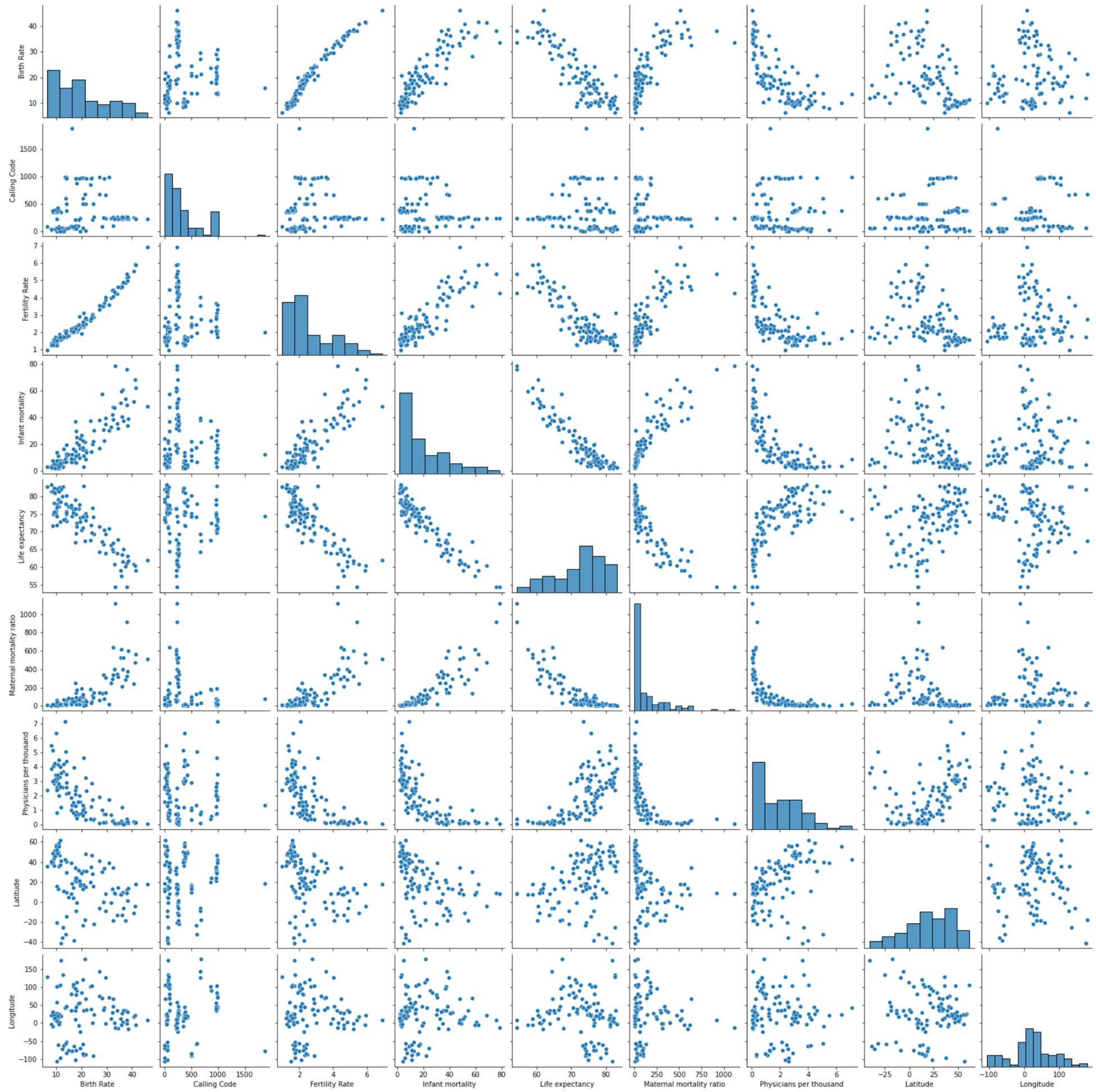
In [13]:
```
df.columns
```

Out[13]:
```
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 expenditure',
       'Physicians per thousand', 'Population',
       'Population: Labor force participation (%)', 'Tax revenue (%)',
       'Total tax rate', 'Unemployment rate', 'Urban_population', 'Latitude',
       'Longitude'],
      dtype='object')
```

EDA and VISUALIZATION

In [14]:
```
sns.pairplot(df)
```

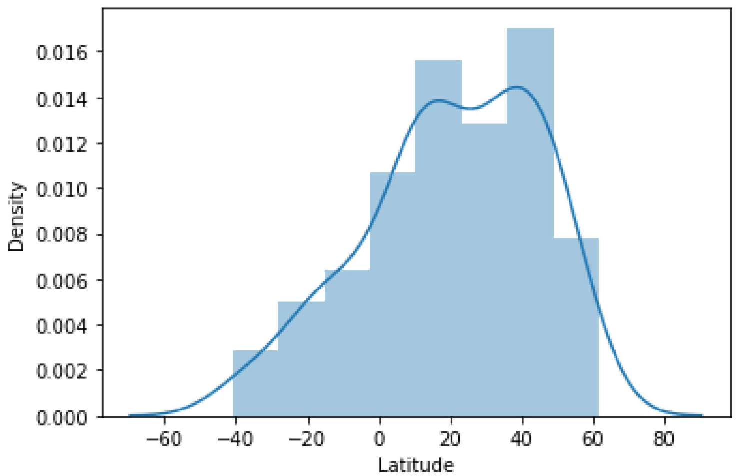
Out[14]: <seaborn.axisgrid.PairGrid at 0x206177becd0>



In [15]:
```
sns.distplot(df['Latitude'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[15]: <AxesSubplot:xlabel='Latitude', ylabel='Density'>



In [16]:
```
df1 = df[['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 expenditure',
       'Physicians per thousand', 'Population',
       'Population: Labor force participation (%)', 'Tax revenue (%)',
       'Total tax rate', 'Unemployment rate', 'Urban_population', 'Latitude',
       'Longitude']]
```

In [17]:
```
sns.heatmap(df1.corr())
```

Out[17]: <AxesSubplot:>

In [18]:
```
x = df1[[ 'Birth Rate','Calling Code','Fertility Rate','Infant mortality','Life expecta
y = df1[ 'Physicians per thousand']
```

split the data into training and test data

In [19]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

In [20]:
```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[20]: LinearRegression()

In [21]:
```
lr.intercept_
```

Out[21]: 1.1102230246251565e-14

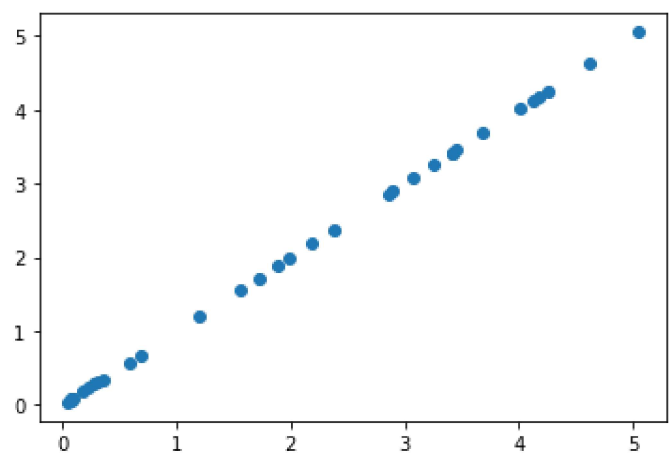
In [22]:
```
coeff = pd.DataFrame(lr.coef_, x.columns, columns =['Fertility Rate'])
coeff
```

Out[22]:

	Fertility Rate
Birth Rate	1.554247e-16
Calling Code	-1.763489e-18
Fertility Rate	1.198357e-15
Infant mortality	-2.629640e-16
Life expectancy	-1.627314e-16
Maternal mortality ratio	5.771328e-18
Physicians per thousand	1.000000e+00
Latitude	-7.941307e-18
Longitude	-4.445337e-18

In [23]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

Out[23]: <matplotlib.collections.PathCollection at 0x2061bf1d730>



In [24]:
```
lr.score(x_test,y_test)
```

Out[24]: 1.0

ACURACY

In [25]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [26]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

Out[26]: 0.994931296508725

In [27]:
```
rr.score(x_test,y_test)
```
Out[27]:
```
0.9948764479359316
```

In [28]:
```
la = Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[28]:
```
Lasso(alpha=10)
```

In [29]:
```
la.score(x_test,y_test)
```
Out[29]:
```
0.46994638257697163
```

In [30]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```
Out[30]:
```
ElasticNet()
```

In [31]:
```
print(en.intercept_)
```
```
1.944500190641182
```

In [32]:
```

print(en.predict(x_test))
```
```
[ 3.53045825  2.68122952  2.39110232  1.81993614  3.07893568  2.41689141
  1.33934525  0.2978623   2.92878239  3.24377094 -0.16892324  1.38545523
  1.72878448  3.11067205  0.09913052  0.46238285  2.62237527  3.2256435
  1.78924011  3.389855    3.23199662  1.1803987   2.60418248  0.2178042
  2.95030164  0.06101522  2.2288558   3.22648357  1.23939681  0.19362643
  3.60575579  2.2116378   0.45649973]
```

In [33]:
```
print(en.score(x_test,y_test))
```
```
0.8396666269015641
```

In [34]:
```
# Evaluation Metrics
from sklearn import metrics
```

In [35]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```
```
Mean Absolute Error: 1.5444295673999573e-15
```

In [36]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```
```
Mean Squared Error: 4.991031401463319e-30
```

In [38]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```
```
Root Mean Squared Error: 2.234061637794114e-15
```