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

In [2]:
```python
df=pd.read_csv("18_world-data-2023.csv")
df.fillna(0,inplace=True)
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 | 0 | 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                               195 non-null    object 
 3   Agricultural Land( %)                      195 non-null    object 
 4   Land Area(Km2)                             195 non-null    object 
 5   Armed Forces size                          195 non-null    object 
 6   Birth Rate                                 195 non-null    float64
 7   Calling Code                               195 non-null    float64
 8   Capital/Major City                         195 non-null    object 
 9   Co2-Emissions                              195 non-null    object 
 10  CPI                                        195 non-null    object 
 11  CPI Change (%)                             195 non-null    object 
 12  Currency-Code                              195 non-null    object 
 13  Fertility Rate                             195 non-null    float64
 14  Forested Area (%)                          195 non-null    object 
 15  Gasoline Price                             195 non-null    object 
 16  GDP                                        195 non-null    object 
 17  Gross primary education enrollment (%)     195 non-null    object 
 18  Gross tertiary education enrollment (%)    195 non-null    object 
 19  Infant mortality                           195 non-null    float64
 20  Largest city                               195 non-null    object 
 21  Life expectancy                            195 non-null    float64
 22  Maternal mortality ratio                   195 non-null    float64
 23  Minimum wage                               195 non-null    object 
 24  Official language                          195 non-null    object 
 25  Out of pocket health expenditure           195 non-null    object 
 26  Physicians per thousand                    195 non-null    float64
 27  Population                                 195 non-null    object 
 28  Population: Labor force participation (%)  195 non-null    object 
 29  Tax revenue (%)                            195 non-null    object 
 30  Total tax rate                             195 non-null    object 
 31  Unemployment rate                          195 non-null    object 
 32  Urban_population                           195 non-null    object 
 33  Latitude                                   195 non-null    float64
 34  Longitude                                  195 non-null    float64
dtypes: float64(9), object(26)
memory usage: 53.4+ KB
```

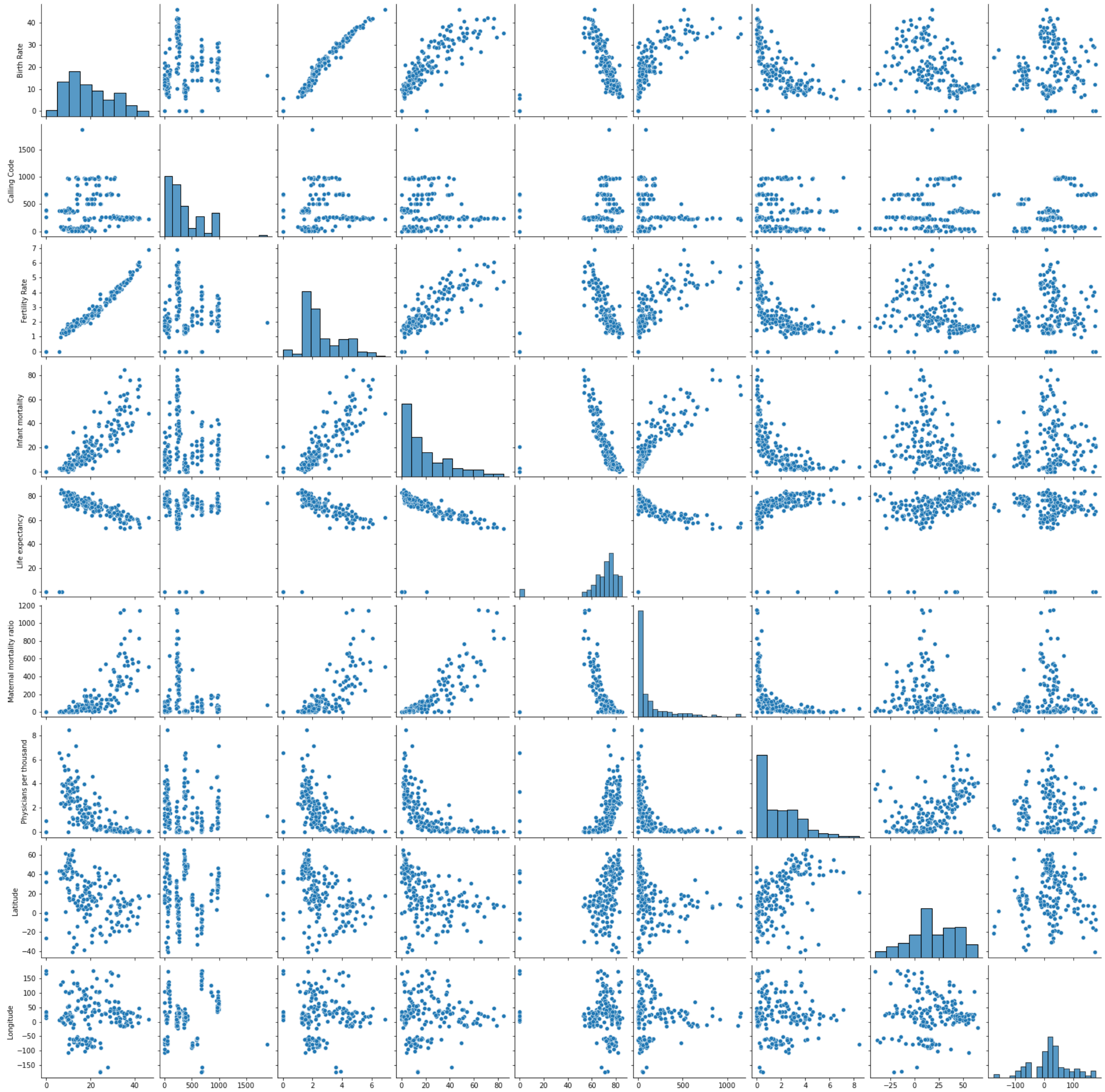
In [4]:
```
df.describe()
```

Out[4]:

| | Birth Rate | Calling Code | Fertility Rate | Infant mortality | Life expectancy | Maternal mortality ratio | Physicians per thousand | |
|---|---|---|---|---|---|---|---|---|
| **count** | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 19 |
| **mean** | 19.592974 | 358.697436 | 2.601282 | 20.676410 | 69.314359 | 148.876923 | 1.773795 | 1 |
| **std** | 10.397534 | 323.434462 | 1.355777 | 19.594644 | 16.133643 | 228.717593 | 1.688826 | 2 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | -4 |
| **25%** | 10.675000 | 81.500000 | 1.625000 | 5.000000 | 66.150000 | 9.000000 | 0.245000 | |
| **50%** | 17.800000 | 255.000000 | 2.200000 | 13.700000 | 72.800000 | 43.000000 | 1.300000 | 1 |
| **75%** | 28.445000 | 506.500000 | 3.565000 | 31.550000 | 77.250000 | 175.000000 | 2.875000 | 4 |
| **max** | 46.080000 | 1876.000000 | 6.910000 | 84.500000 | 85.400000 | 1150.000000 | 8.420000 | 6 |

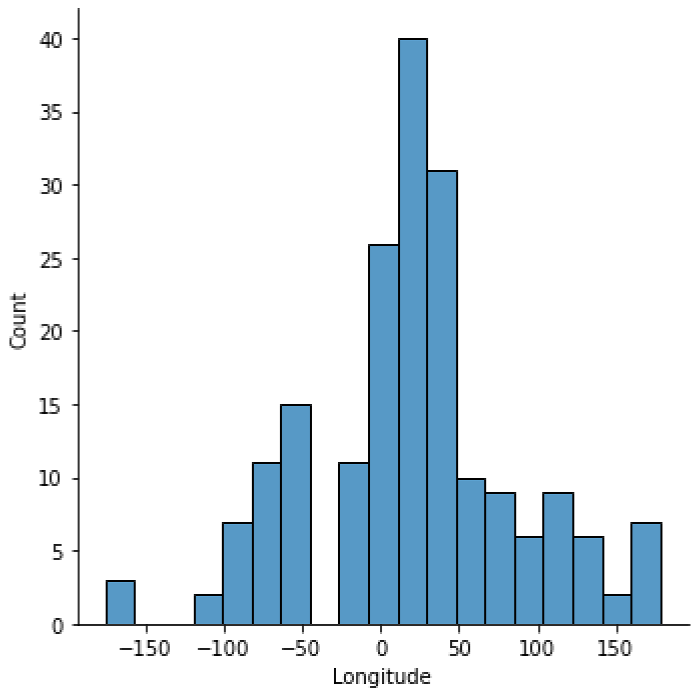
```
In [5]: sns.pairplot(df)
```

```
Out[5]: <seaborn.axisgrid.PairGrid at 0x2184d7addc0>
```



In [6]:
```
sns.displot(df['Longitude'])
```

Out[6]: <seaborn.axisgrid.FacetGrid at 0x2184fd86790>




In [7]:
```
df1=df.drop(['Country'],axis=1)
df1
```

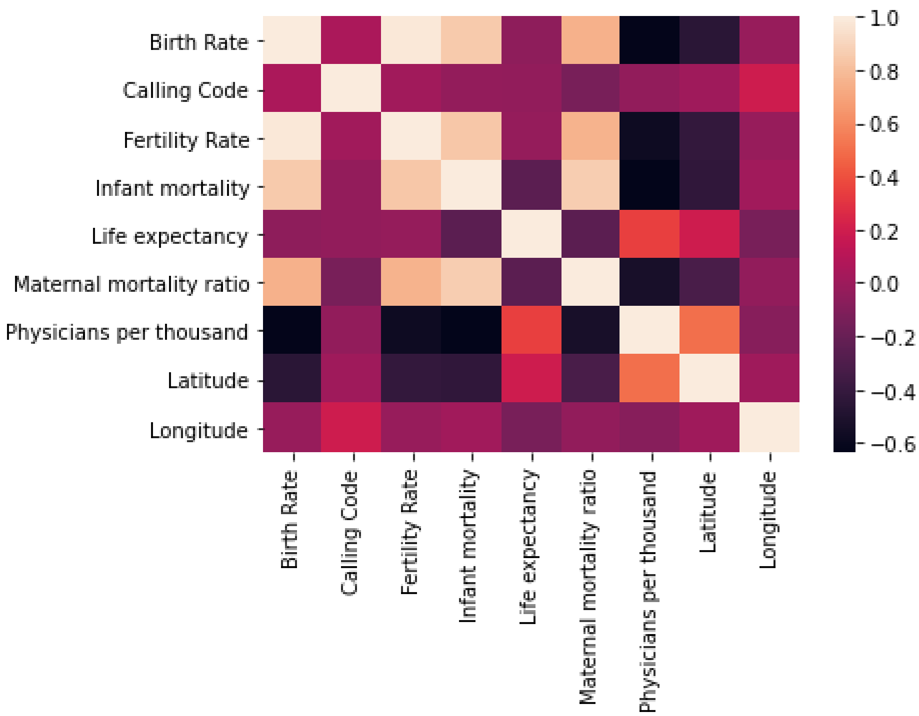
Out[7]:

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

195 rows × 34 columns

In [8]:
```python
sns.heatmap(df1.corr())
```

Out[8]: <AxesSubplot:>




In [9]:
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [10]:
```python
y=df['Longitude']
x=df1.drop(['Longitude','Abbreviation','Agricultural Land( %)','Land Area(Km2)'
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train)
```

```
     Birth Rate  Calling Code   Latitude
190       17.88          58.0   6.423750
105       41.54         223.0  17.570692
154       17.10         248.0  -4.679574
152       34.52         221.0  14.497401
165       15.83          94.0   7.873054
..          ...           ...        ...
27        39.01         257.0  -3.373056
16        10.30          32.0  50.503887
73         0.00         379.0  41.902916
23        13.92          55.0 -14.235004
157       10.60         421.0  48.669026

[136 rows x 3 columns]
```

In [11]:
```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

Out[11]: 11.085740356550236

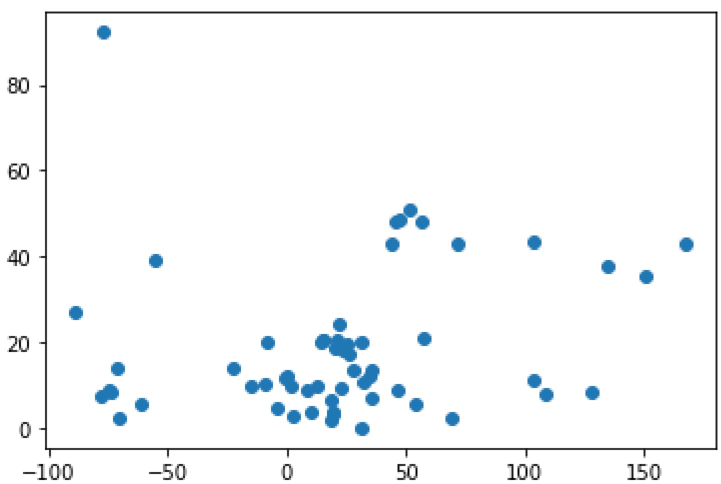
In [12]:
```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

Out[12]:

| | Coefficient |
|---|---|
| **Birth Rate** | -0.320364 |
| **Calling Code** | 0.047229 |
| **Latitude** | -0.127072 |

In [13]:
```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[13]: <matplotlib.collections.PathCollection at 0x21852125a60>




In [14]:
```
model.score(x_test,y_test)
```

Out[14]: 0.003921725830336453

In [15]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [16]:
```
rr = Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[16]: Ridge(alpha=10)

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

Out[17]: 0.0039329137307800854

In [18]:
```
la = Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[18]:
```
Lasso(alpha=10)
```

In [19]:
```
la.score(x_test,y_test)
```

Out[19]:
```
0.009611494267665388
```

In [20]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
print(en.coef_)
print(en.intercept_)
print(en.predict(x_test))
print(en.score(x_test,y_test))
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
```

```
[-0.31032468  0.04720037 -0.12413397]
10.840855658686868
[ 5.43752531 48.42471327 42.71879915  8.81853357 43.60305383 24.0367542
 38.80275824  2.64600812  8.9939488  47.91008766 50.70612272 35.45293817
 11.01018304 20.41420935  6.92816362  9.87989343 13.69221578  2.50788548
  5.61318929  6.87450685 27.20688905  8.26783546 10.03799384  3.49619554
 47.9475763  11.72545888 13.97328858  3.15337859 17.38290986  9.13973656
 20.62163358 14.05523827 10.91788959  2.01042378  3.70648427  3.85418879
 10.6930735  20.07164009 19.87956289  8.34624545 10.13011985 37.65969699
  4.9711875  20.32813595 43.29406037 19.71917194 20.50526372  0.26915615
 12.27864679  7.86263266  7.51489203  9.54397631 12.33410349 13.51266428
 43.19364334 92.14450706 18.35334401 21.0575571  18.83283546]
0.004320887974378906
Mean Absolute Error: 37.66836110039882
Mean Squared Error: 3100.045426966432
Root Mean Squared Error: 55.67805157300704
```