# Problem Statement

A real estate agent want help to predict the house price for regions in USA.He gave us the dataset to work on to use linear regression model.Create a model that helps him to estimate of what the house would sell for

# Import libraries

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

In [2]:
```
# To import dataset
df=pd.read_csv('18 world csv')
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]:
```python
# To display top 10 rows
df.head(10)
```

Out[3]:

| | 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 |
| **5** | Antigua and Barbuda | 223 | AG | 20.50% | 443 | 0 | 15.33 | 1.0 |
| **6** | Argentina | 17 | AR | 54.30% | 2,780,400 | 105,000 | 17.02 | 54.0 |
| **7** | Armenia | 104 | AM | 58.90% | 29,743 | 49,000 | 13.99 | 374.0 |
| **8** | Australia | 3 | AU | 48.20% | 7,741,220 | 58,000 | 12.60 | 61.0 |
| **9** | Austria | 109 | AT | 32.40% | 83,871 | 21,000 | 9.70 | 43.0 |

10 rows × 35 columns

# Data Cleaning and Pre-Processing

In [4]:
```
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 [5]:
```
df.describe()
```

Out[5]:

| | 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 |
| **mean** | 20.214974 | 360.546392 | 2.698138 | 21.332804 | 72.279679 | 160.392265 | 1.839840 |
| **std** | 9.945774 | 323.236419 | 1.282267 | 19.548058 | 7.483661 | 233.502024 | 1.684261 |
| **min** | 5.900000 | 1.000000 | 0.980000 | 1.400000 | 52.800000 | 2.000000 | 0.010000 |
| **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 |
| **75%** | 28.750000 | 506.750000 | 3.597500 | 32.700000 | 77.500000 | 186.000000 | 2.935000 |
| **max** | 46.080000 | 1876.000000 | 6.910000 | 84.500000 | 85.400000 | 1150.000000 | 8.420000 |

In [6]:
```
df.columns
```

Out[6]:
```
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')
```

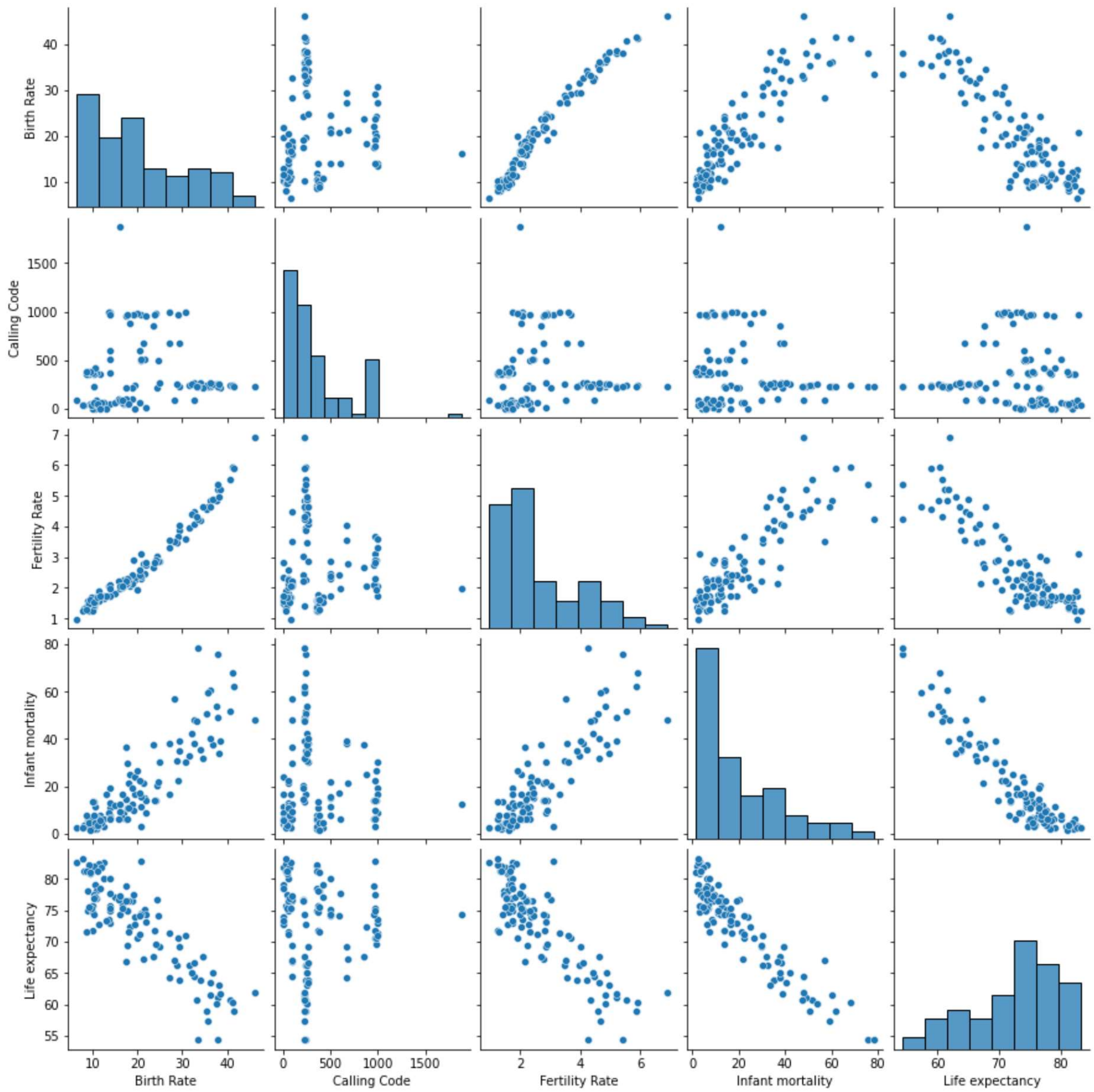
In [9]:
```
a = df.dropna()
a.columns
```

Out[9]:
```
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')
```

# EDA and Visualization

In [10]:
```
sns.pairplot(a[['Birth Rate', 'Calling Code', 'Fertility Rate', 'Infant mortal
        'Life expectancy']])
```

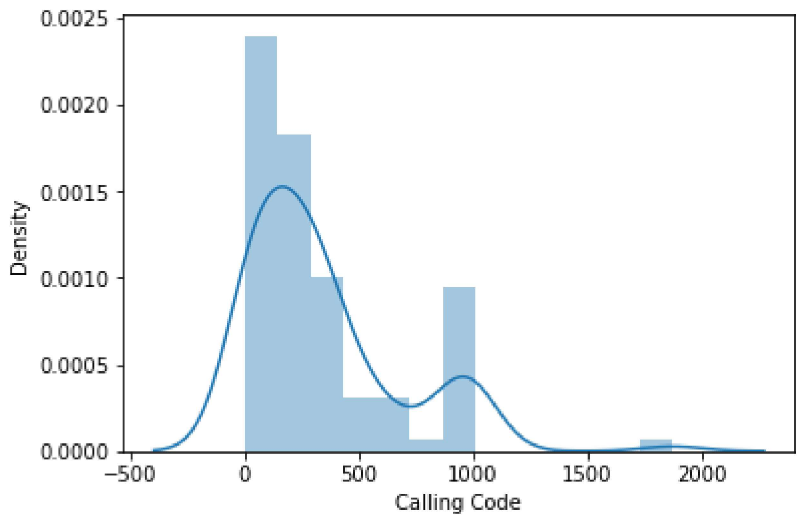
Out[10]: <seaborn.axisgrid.PairGrid at 0x18807174a30>



In [11]:
```
sns.distplot(a['Calling Code'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[11]: <AxesSubplot:xlabel='Calling Code', ylabel='Density'>




In [12]:
```
a1=a[[ 'Birth Rate', 'Calling Code', 'Fertility Rate', 'Infant mortality',
       'Life expectancy', 'Maternal mortality ratio','Physicians per thousand'
       'Latitude',  'Longitude']]
```

In [13]:
```
sns.heatmap(a1.corr())
```

Out[13]:
```
<AxesSubplot:>
```


# To Train the Model - Model Building

We are going to train Linear Regression model;We need to split out data into two variables x and y where x is independent variable (input) and y is dependent on x(output). We could ignore address column as it is not required for our model.

In [14]:
```
x=a1[['Birth Rate', 'Calling Code', 'Fertility Rate', 'Infant mortality',
       'Life expectancy', 'Maternal mortality ratio','Physicians per thousand'
       'Latitude',  'Longitude']]
y=a1['Calling Code']
```

# To split my dataset into training and test data

In [15]:
```
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 [16]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[16]:
```
LinearRegression()
```

In [17]:
```
print(lr.intercept_)
```

```
1.1368683772161603e-13
```

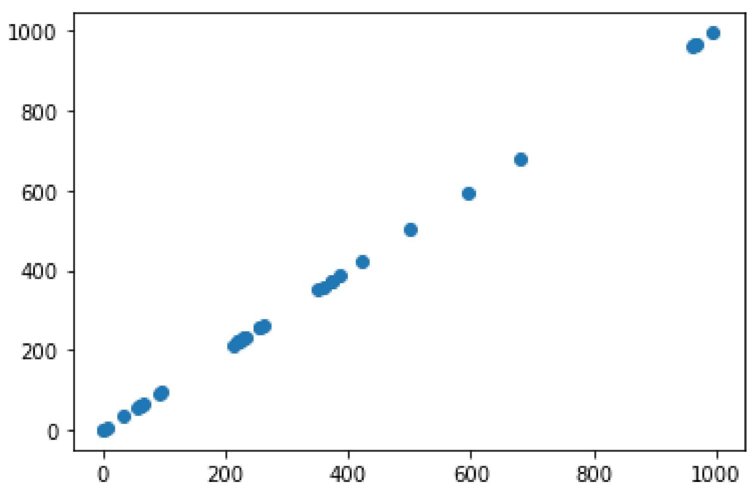
In [18]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[18]:

| | Co-efficient |
|---|---|
| **Birth Rate** | 4.544961e-14 |
| **Calling Code** | 1.000000e+00 |
| **Fertility Rate** | -3.031892e-13 |
| **Infant mortality** | -5.384855e-15 |
| **Life expectancy** | 1.378741e-16 |
| **Maternal mortality ratio** | 6.061242e-16 |
| **Physicians per thousand** | 8.679063e-15 |
| **Latitude** | 5.509456e-16 |
| **Longitude** | 3.513600e-17 |

In [19]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[19]: <matplotlib.collections.PathCollection at 0x18809355490>




In [20]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

# ACCURACY

In [21]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [22]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

Out[22]: 0.9999999999988388

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

Out[23]: 0.9999999999988685

In [24]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]: Lasso(alpha=10)

In [25]:
```
la.score(x_test,y_test)
```

Out[25]: 0.9999999938780108

In [ ]: