Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

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("world-data")
```

To display top 10 rows

In [3]:
```
df.head(10)
```

Out[3]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital/Major City	Co2-Emissions	...	Out of pocket health expenditure
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	Kabul	8,672	...	78.40%
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	Tirana	4,536	...	56.90%
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	Algiers	150,006	...	28.10%
3	Andorra	164	AD	40.00%	468	NaN	7.20	376.0	Andorra la Vella	469	...	36.40%
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	Luanda	34,693	...	33.40%
5	Antigua and Barbuda	223	AG	20.50%	443	0	15.33	1.0	St. John's, Saint John	557	...	24.30%
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0	Buenos Aires	201,348	...	17.60%
7	Armenia	104	AM	58.90%	29,743	49,000	13.99	374.0	Yerevan	5,156	...	81.60%
8	Australia	3	AU	48.20%	7,741,220	58,000	12.60	61.0	Canberra	375,908	...	19.60%
9	Austria	109	AT	32.40%	83,871	21,000	9.70	43.0	Vienna	61,448	...	17.90%

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]:

```
# Display the statistical summary
df.describe()
```

Out[5]:

	Birth Rate	Calling Code	Fertility Rate	Infant mortality	Life expectancy	Maternal mortality ratio	Physicians per thousand	Latitude	Longitude
count	189.000000	194.000000	188.000000	189.000000	187.000000	181.000000	188.000000	194.000000	194.000000
mean	20.214974	360.546392	2.698138	21.332804	72.279679	160.392265	1.839840	19.092351	20.232434
std	9.945774	323.236419	1.282267	19.548058	7.483661	233.502024	1.684261	23.961779	66.716110
min	5.900000	1.000000	0.980000	1.400000	52.800000	2.000000	0.010000	-40.900557	-175.198242
25%	11.300000	82.500000	1.705000	6.000000	67.000000	13.000000	0.332500	4.544175	-7.941496
50%	17.950000	255.500000	2.245000	14.000000	73.200000	53.000000	1.460000	17.273849	20.972652
75%	28.750000	506.750000	3.597500	32.700000	77.500000	186.000000	2.935000	40.124603	48.281523
max	46.080000	1876.000000	6.910000	84.500000	85.400000	1150.000000	8.420000	64.963051	178.065032

In [6]:

```
# To display the col headings
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 expenditure',
       'Physicians per thousand', 'Population',
       'Population: Labor force participation (%)', 'Tax revenue (%)',
       'Total tax rate', 'Unemployment rate', 'Urban_population', 'Latitude',
       'Longitude'],
      dtype='object')
```

In [37]:
```
1 cols=df.dropna()
2 cols
```

Out[37]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital/Major City	Co2-Emissions	...	O pc h expend
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	Kabul	8,672	...	78
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	Tirana	4,536	...	56
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	Algiers	150,006	...	28
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	Luanda	34,693	...	33
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0	Buenos Aires	201,348	...	17
...	...	...	...	...	...	...	...	...	...	...	...	
185	United Kingdom	281	GB	71.70%	243,610	148,000	11.00	44.0	London	379,025	...	14
186	United States	36	US	44.40%	9,833,517	1,359,000	11.60	1.0	Washington, D.C.	5,006,302	...	11
187	Uruguay	20	UY	82.60%	176,215	22,000	13.86	598.0	Montevideo	6,766	...	16
191	Vietnam	314	VN	39.30%	331,210	522,000	16.75	84.0	Hanoi	192,668	...	43
193	Zambia	25	ZM	32.10%	752,618	16,000	36.19	260.0	Lusaka	5,141	...	27

110 rows × 35 columns

In [36]:
```
cols.columns
```

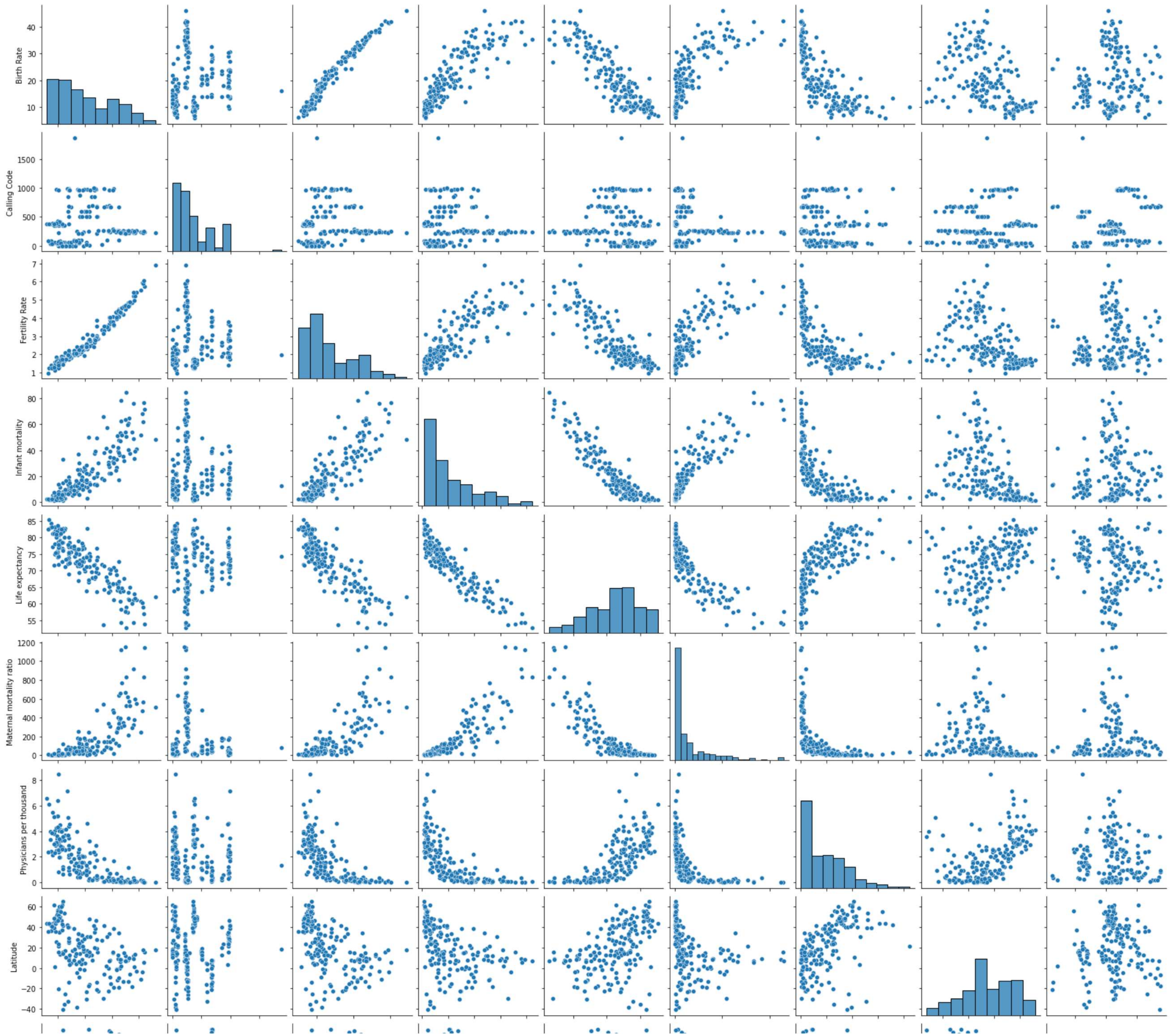
Out[36]:
```
Index(['Country', 'Density\n(P/Km2)', 'Abbreviation', 'Agricultural Land( %)',
       'Land Area(Km2)', 'Armed Forces size', 'Birth Rate', 'Calling Code',
       'Capital/Major City', 'Co2-Emissions', 'CPI', 'CPI Change (%)',
       'Currency-Code', 'Fertility Rate', 'Forested Area (%)',
       '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

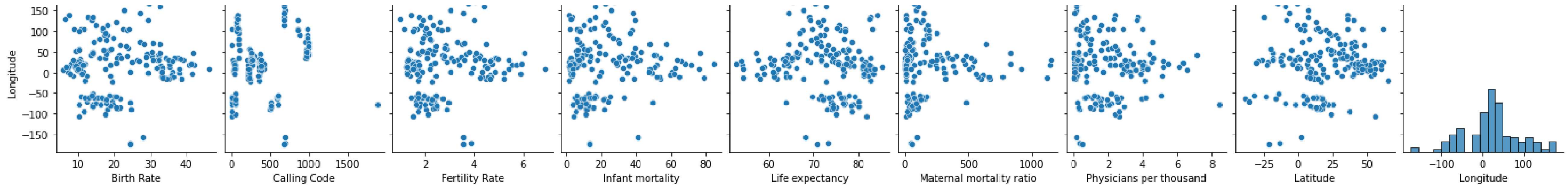
In [10]:

```
sns.pairplot(df)
```

Out[10]: <seaborn.axisgrid.PairGrid at 0x22c06241fa0>



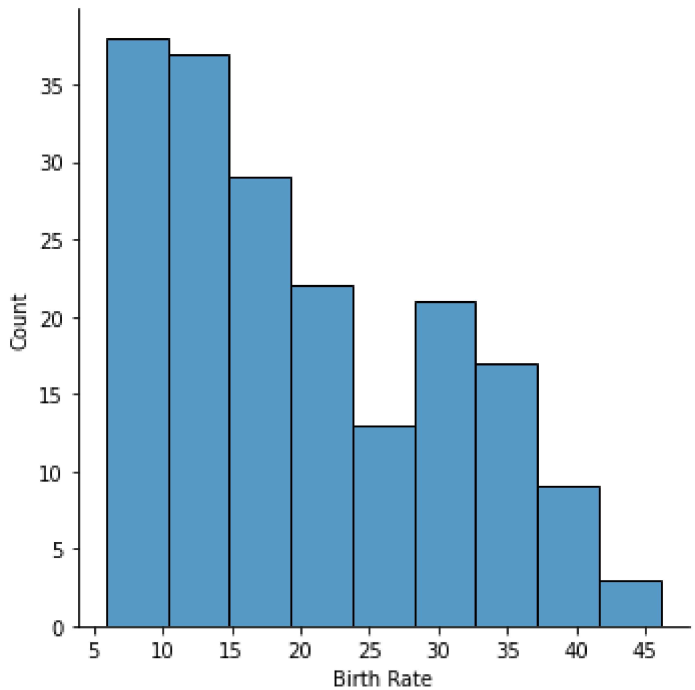
Birth Rate
Calling Code
Fertility Rate
Infant mortality
Life expectancy
Maternal mortality ratio
Physicians per thousand
Latitude



In [38]:

```
sns.displot(df['Birth Rate'])
```

Out[38]: <seaborn.axisgrid.FacetGrid at 0x22c0be40eb0>



In [48]:

```
df1=cols[['Birth Rate', 'Calling Code','Latitude',
        'Longitude']]
df1
```

Out[48]:

	Birth Rate	Calling Code	Latitude	Longitude
0	32.49	93.0	33.939110	67.709953
1	11.78	355.0	41.153332	20.168331
2	24.28	213.0	28.033886	1.659626
4	40.73	244.0	-11.202692	17.873887
6	17.02	54.0	-38.416097	-63.616672
...	...	...	...	...
185	11.00	44.0	55.378051	-3.435973
186	11.60	1.0	37.090240	-95.712891
187	13.86	598.0	-32.522779	-55.765835
191	16.75	84.0	14.058324	108.277199
193	36.19	260.0	-13.133897	27.849332

110 rows × 4 columns

In [49]:
```
sns.heatmap(df.corr())
```
Out[49]: <AxesSubplot:>

To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [50]:
```
x=df1[['Birth Rate', 'Calling Code','Latitude',
        'Longitude']]
y=df1[['Birth Rate']]
```

To split the dataset into test data

In [51]:
```
# importing lib for splitting test data
from sklearn.model_selection import train_test_split
```

In [52]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [53]:
```
from sklearn.linear_model import LinearRegression

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

Out[53]:
```
LinearRegression()
```

In [54]:
```
print(lr.intercept_)
```

```
[-3.55271368e-15]
```

In [55]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

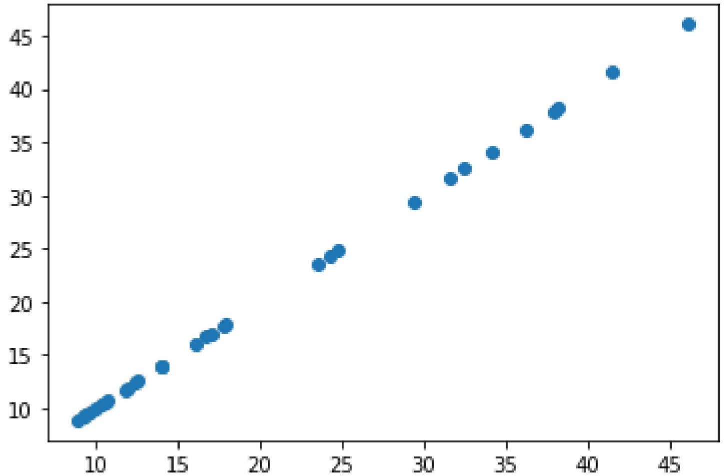
In [56]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[56]:

	0	1	2	3
0	1.0	2.756087e-17	-8.430790e-17	2.977896e-18

In [57]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```

Out[57]: <matplotlib.collections.PathCollection at 0x22c0be40820>



In [58]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [59]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[59]: Ridge(alpha=10)

In [60]:
```
rr.score(x_test,y_test)
```

Out[60]: 0.9999958918418803

In [61]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[61]: Lasso(alpha=10)

In [62]:
```
la.score(x_test,y_test)
```

Out[62]: 0.9863459147333142

In [ ]:

```
 
```