

In [35]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [36]:

```
df1=pd.read_csv(r'C:\Users\user\Downloads\18_world-data-2023.csv')
df1
```

Out[36]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Call Co
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	9
1	Albania	105	AL	43.10%	28,748	9,000	11.78	35
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	21
3	Andorra	164	AD	40.00%	468	NaN	7.20	37
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	24
...
190	Venezuela	32	VE	24.50%	912,050	343,000	17.88	5
191	Vietnam	314	VN	39.30%	331,210	522,000	16.75	8
192	Yemen	56	YE	44.60%	527,968	40,000	30.45	96
193	Zambia	25	ZM	32.10%	752,618	16,000	36.19	26
194	Zimbabwe	38	ZW	41.90%	390,757	51,000	30.68	26

195 rows × 35 columns

In [67]:

```
df=df1.head(50)  
df
```

Out[67]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Ca
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	
1	Albania	105	AL	43.10%	28,748	9,000	11.78	3
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	2
3	Andorra	164	AD	40.00%	468	NaN	7.20	3
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	2
5	Antigua and Barbuda	223	AG	20.50%	443	0	15.33	
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	
7	Armenia	104	AM	58.90%	29,743	49,000	13.99	3
8	Australia	3	AU	48.20%	7,741,220	58,000	12.60	
9	Austria	109	AT	32.40%	83,871	21,000	9.70	
10	Azerbaijan	123	AZ	57.70%	86,600	82,000	14.00	9
11	The Bahamas	39	BS	1.40%	13,880	1,000	13.97	
12	Bahrain	2,239	BH	11.10%	765	19,000	13.99	9
13	Bangladesh	1,265	BD	70.60%	148,460	221,000	18.18	8
14	Barbados	668	BB	23.30%	430	1,000	10.65	
15	Belarus	47	BY	42.00%	207,600	155,000	9.90	3
16	Belgium	383	BE	44.60%	30,528	32,000	10.30	
17	Belize	17	BZ	7.00%	22,966	2,000	20.79	5
18	Benin	108	BJ	33.30%	112,622	12,000	36.22	2
19	Bhutan	20	BT	13.60%	38,394	6,000	17.26	9
20	Bolivia	11	BO	34.80%	1,098,581	71,000	21.75	5
21	Bosnia and Herzegovina	64	BA	43.10%	51,197	11,000	8.11	3
22	Botswana	4	BW	45.60%	581,730	9,000	24.82	2
23	Brazil	25	BR	33.90%	8,515,770	730,000	13.92	
24	Brunei	83	BN	2.70%	5,765	8,000	14.90	6
25	Bulgaria	64	BG	46.30%	110,879	31,000	8.90	3
26	Burkina Faso	76	BF	44.20%	274,200	11,000	37.93	2
27	Burundi	463	BI	79.20%	27,830	31,000	39.01	2
28	Ivory Coast	83	CI	64.80%	322,463	27,000	35.74	2
29	Cape Verde	138	CV	19.60%	4,033	1,000	19.49	2
30	Cambodia	95	KH	30.90%	181,035	191,000	22.46	8
31	Cameroon	56	CM	20.60%	475,440	24,000	35.39	2
32	Canada	4	CA	6.90%	9,984,670	72,000	10.10	

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Ca
33	Central African Republic	8	CF	8.20%	622,984	8,000	35.35	2
34	Chad	13	TD	39.70%	1,284,000	35,000	42.17	2
35	Chile	26	CL	21.20%	756,096	122,000	12.43	
36	China	153	CN	56.20%	9,596,960	2,695,000	10.90	
37	Colombia	46	CO	40.30%	1,138,910	481,000	14.88	
38	Comoros	467	KM	71.50%	2,235	NaN	31.88	2
39	Republic of the Congo	16	NaN	31.10%	342,000	12,000	32.86	2
40	Costa Rica	100	CR	34.50%	51,100	10,000	13.97	5
41	Croatia	73	HR	27.60%	56,594	18,000	9.00	3
42	Cuba	106	CU	59.90%	110,860	76,000	10.17	
43	Cyprus	131	CY	12.20%	9,251	16,000	10.46	3
44	Czech Republic	139	CZ	45.20%	78,867	23,000	10.70	4
45	Democratic Republic of the Congo	40	CD	11.60%	2,344,858	134,000	41.18	2
46	Denmark	137	DK	62.00%	43,094	15,000	10.60	
47	Djibouti	43	DJ	73.40%	23,200	13,000	21.47	2
48	Dominica	96	DM	33.30%	751	NaN	12.00	
49	Dominican Republic	225	DO	48.70%	48,670	71,000	19.51	

50 rows × 35 columns

In [68]:

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 50 entries, 0 to 49

Data columns (total 35 columns):

#	Column	Non-Null Count	Dtype
0	Country	50 non-null	object
1	Density (P/Km2)	50 non-null	object
2	Abbreviation	49 non-null	object
3	Agricultural Land(%)	50 non-null	object
4	Land Area(Km2)	50 non-null	object
5	Armed Forces size	47 non-null	object
6	Birth Rate	50 non-null	float64
7	Calling Code	50 non-null	float64
8	Capital/Major City	50 non-null	object
9	Co2-Emissions	50 non-null	object
10	CPI	47 non-null	object
11	CPI Change (%)	48 non-null	object
12	Currency-Code	46 non-null	object
13	Fertility Rate	50 non-null	float64
14	Forested Area (%)	50 non-null	object
15	Gasoline Price	48 non-null	object
16	GDP	50 non-null	object
17	Gross primary education enrollment (%)	49 non-null	object
18	Gross tertiary education enrollment (%)	48 non-null	object
19	Infant mortality	50 non-null	float64
20	Largest city	49 non-null	object
21	Life expectancy	49 non-null	float64
22	Maternal mortality ratio	48 non-null	float64
23	Minimum wage	42 non-null	object
24	Official language	50 non-null	object
25	Out of pocket health expenditure	49 non-null	object
26	Physicians per thousand	50 non-null	float64
27	Population	50 non-null	object
28	Population: Labor force participation (%)	47 non-null	object
29	Tax revenue (%)	44 non-null	object
30	Total tax rate	48 non-null	object
31	Unemployment rate	47 non-null	object
32	Urban_population	50 non-null	object
33	Latitude	50 non-null	float64
34	Longitude	50 non-null	float64

dtypes: float64(9), object(26)

memory usage: 13.8+ KB

In [69]:

```
df.describe()
```

Out[69]:

	Birth Rate	Calling Code	Fertility Rate	Infant mortality	Life expectancy	Maternal mortality ratio	Physicians per thousand	La
count	50.00000	50.00000	50.00000	50.00000	49.00000	48.00000	50.00000	50.0
mean	19.64860	291.82000	2.62600	22.61800	72.31224	174.04167	1.92980	17.6
std	10.67511	272.35366	1.41232	22.04237	7.98849	248.70754	1.78245	24.0
min	7.20000	1.00000	1.27000	1.90000	52.80000	2.00000	0.04000	-38.4
25%	10.75000	56.25000	1.66000	5.22500	66.60000	13.75000	0.27250	5.0
50%	14.89000	240.00000	1.94000	11.75000	74.90000	50.00000	1.66500	16.7
75%	24.68500	375.75000	2.98250	35.37500	78.10000	242.75000	2.97250	39.0
max	42.17000	994.00000	5.92000	84.50000	82.70000	1140.00000	8.42000	56.2

In [70]:

```
df.columns
```

Out[70]:

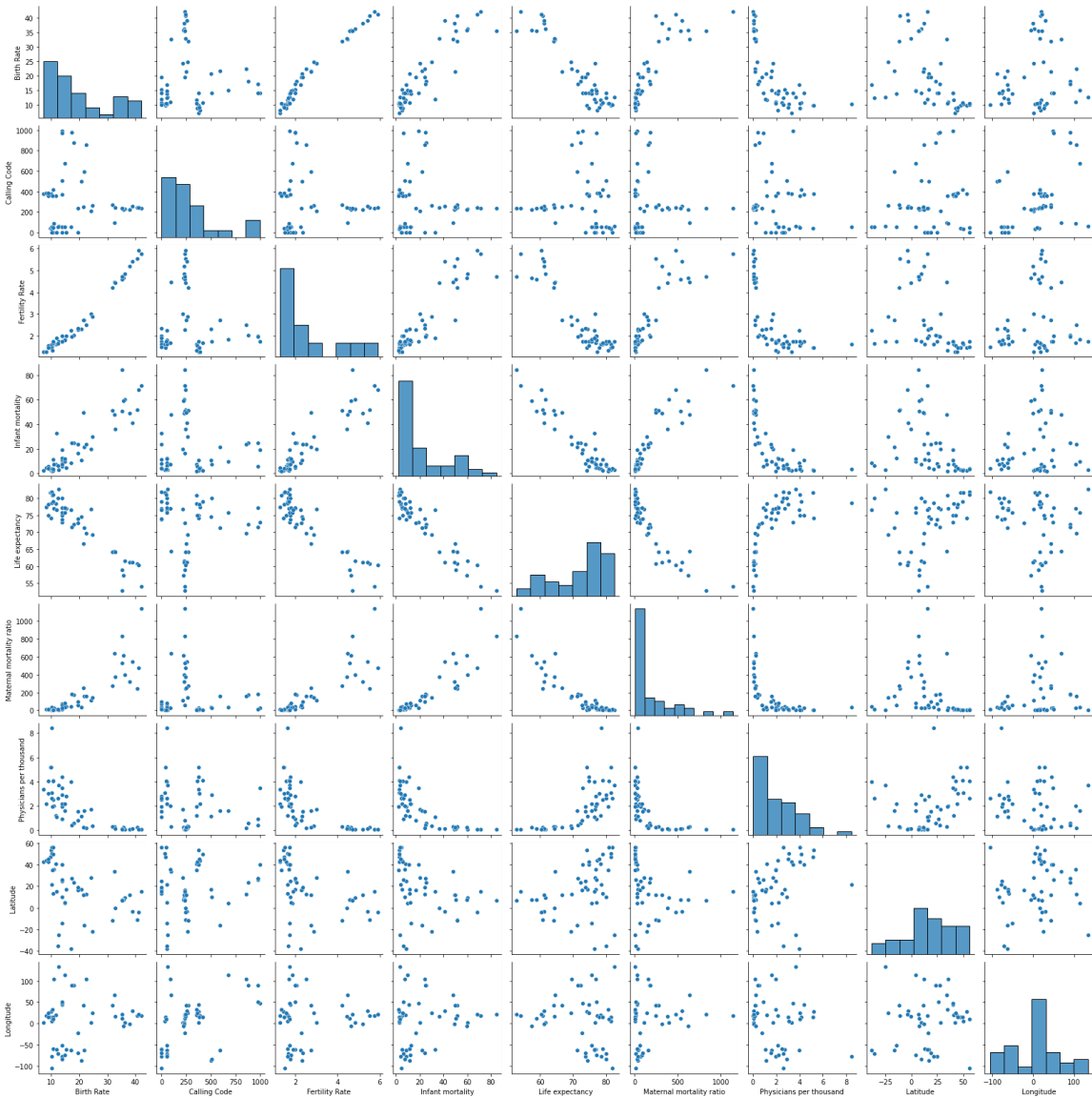
```
Index(['Country', 'Density\n(P/Km2)', 'Abbreviation', 'Agricultural Land(
%)',
      'Land Area(Km2)', 'Armed Forces size', 'Birth Rate', 'Calling Cod
e',
      'Capital/Major City', 'Co2-Emissions', 'CPI', 'CPI Change (%)',
      'Currency-Code', 'Fertility Rate', 'Forested Area (%)',
      'Gasoline 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 expendit
ure',
      'Physicians per thousand', 'Population',
      'Population: Labor force participation (%)', 'Tax revenue (%)',
      'Total tax rate', 'Unemployment rate', 'Urban_population', 'Latitud
e',
      'Longitude'],
      dtype='object')
```

In [71]:

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

Out[71]:

<seaborn.axisgrid.PairGrid at 0x167278a6d90>



In [72]:

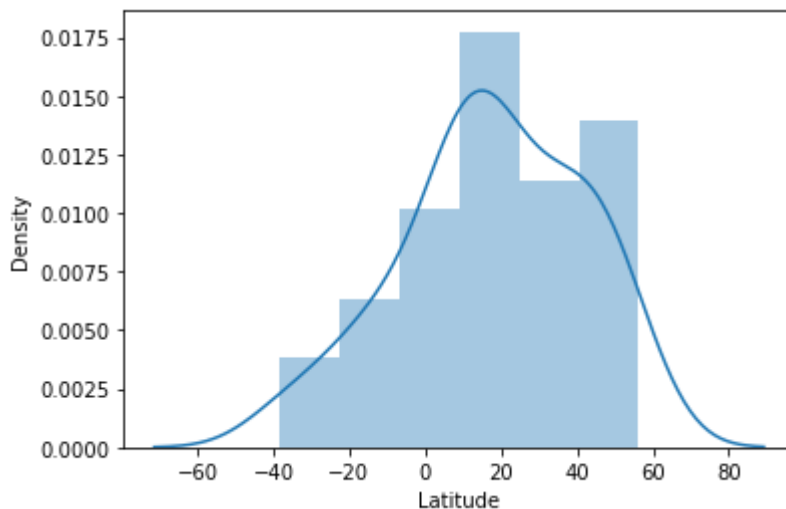
```
sns.distplot(df['Latitude'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `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 flexibility) or `histplot` (an axes-level fun
ction for histograms).

```
warnings.warn(msg, FutureWarning)
```

Out[72]:

<AxesSubplot:xlabel='Latitude', ylabel='Density'>

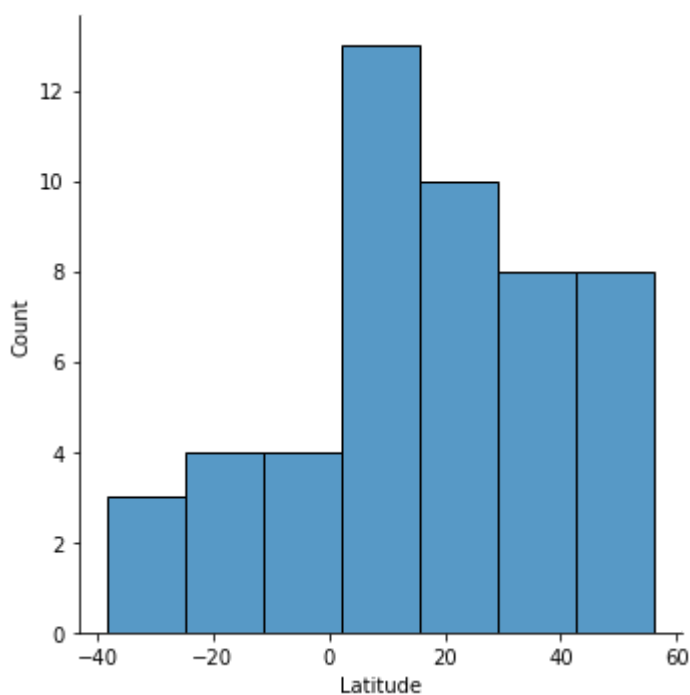


In [73]:

```
sns.displot(df["Latitude"])
```

Out[73]:

<seaborn.axisgrid.FacetGrid at 0x1672bf451c0>



In [74]:

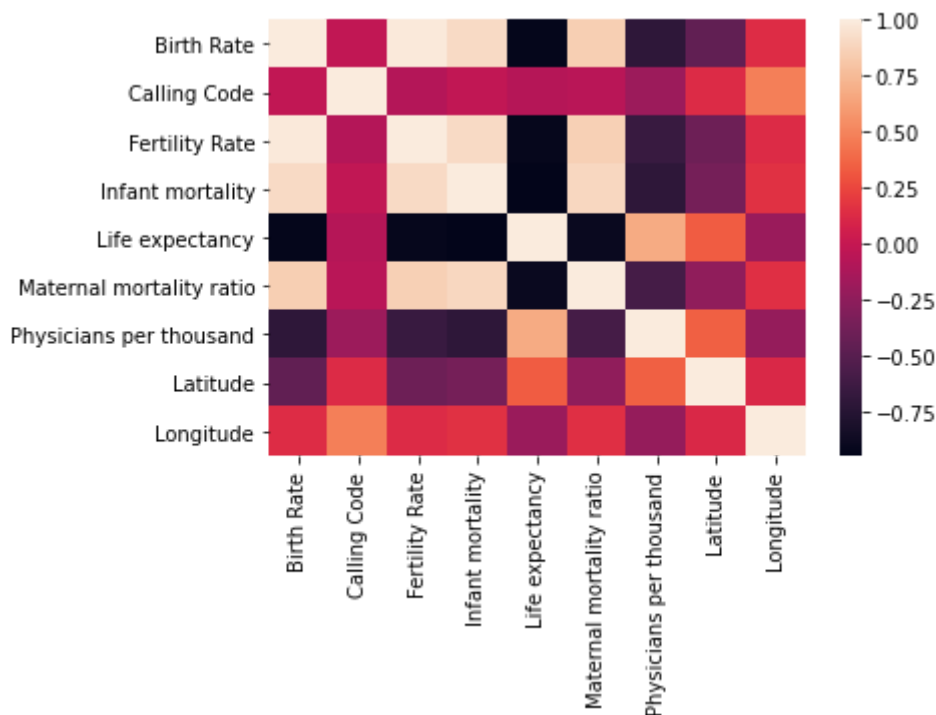
```
df1=df[['Country', 'Density\n(P/Km2)', 'Abbreviation', 'Agricultural Land( %)',
'Land Area(Km2)', 'Armed Forces size', 'Birth Rate', 'Calling Code',
'Capital/Major City', 'Co2-Emissions', 'CPI', 'CPI Change (%)',
'Currency-Code', 'Fertility Rate', 'Forested Area (%)',
'Gasoline Price', 'GDP', 'Gross primary education enrollment (%)',
'Gross tertiary education enrollment (%)', 'Infant mortality',
'Large 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 [75]:

```
sns.heatmap(df1.corr())
```

Out[75]:

<AxesSubplot:>



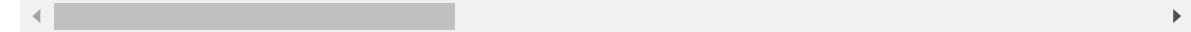
In [76]:

```
df2=df.dropna()  
df2
```

Out[76]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Cal C
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	1
1	Albania	105	AL	43.10%	28,748	9,000	11.78	38
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	2
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	2
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	1
7	Armenia	104	AM	58.90%	29,743	49,000	13.99	31
8	Australia	3	AU	48.20%	7,741,220	58,000	12.60	0
10	Azerbaijan	123	AZ	57.70%	86,600	82,000	14.00	90
13	Bangladesh	1,265	BD	70.60%	148,460	221,000	18.18	80
14	Barbados	668	BB	23.30%	430	1,000	10.65	
16	Belgium	383	BE	44.60%	30,528	32,000	10.30	1
17	Belize	17	BZ	7.00%	22,966	2,000	20.79	50
18	Benin	108	BJ	33.30%	112,622	12,000	36.22	20
22	Botswana	4	BW	45.60%	581,730	9,000	24.82	20
23	Brazil	25	BR	33.90%	8,515,770	730,000	13.92	1
25	Bulgaria	64	BG	46.30%	110,879	31,000	8.90	30
26	Burkina Faso	76	BF	44.20%	274,200	11,000	37.93	20
28	Ivory Coast	83	CI	64.80%	322,463	27,000	35.74	20
29	Cape Verde	138	CV	19.60%	4,033	1,000	19.49	20
31	Cameroon	56	CM	20.60%	475,440	24,000	35.39	20
32	Canada	4	CA	6.90%	9,984,670	72,000	10.10	
35	Chile	26	CL	21.20%	756,096	122,000	12.43	1
36	China	153	CN	56.20%	9,596,960	2,695,000	10.90	1
37	Colombia	46	CO	40.30%	1,138,910	481,000	14.88	1
40	Costa Rica	100	CR	34.50%	51,100	10,000	13.97	50
41	Croatia	73	HR	27.60%	56,594	18,000	9.00	30
44	Czech Republic	139	CZ	45.20%	78,867	23,000	10.70	40
45	Democratic Republic of the Congo	40	CD	11.60%	2,344,858	134,000	41.18	20
49	Dominican Republic	225	DO	48.70%	48,670	71,000	19.51	

29 rows × 35 columns



In [77]:

```
x=df2[['Birth Rate', 'Calling Code', 'Fertility Rate', 'Infant mortality', 'Life expectancy',
        'Physicians per thousand', 'Longitude']]
y=df2[['Latitude']]
```

In [78]:

```
from sklearn.model_selection import train_test_split
```

In [79]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [80]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for
```

Out[80]:

```
LinearRegression()
```

In [81]:

```
print(lr.intercept_)
```

```
[8.13004065]
```

In [82]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

Out[82]:

	0	1	2	3	4	5	6	7
0	-11.674941	0.083692	81.155168	-1.536518	0.749901	0.107865	-18.267723	-0.044176

In [83]:

```
print(lr.score(x_test,y_test))
```

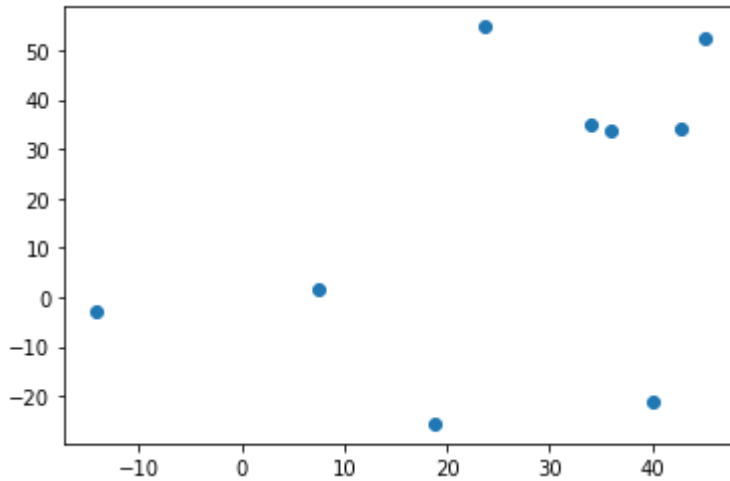
```
-1.3194307371800518
```

In [84]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[84]:

<matplotlib.collections.PathCollection at 0x1672ca548b0>



In [85]:

```
lr.score(x_test,y_test)
```

Out[85]:

-1.3194307371800518

In [86]:

```
lr.score(x_train,y_train)
```

Out[86]:

0.504657025540173

In [87]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [88]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[88]:

Ridge(alpha=10)

In [89]:

```
rr.score(x_test,y_test)
```

Out[89]:

-0.5220340659120757

In [90]:

```
la=Lasso(alpha=10)  
la.fit(x_train,y_train)
```

Out[90]:

Lasso(alpha=10)

In [91]:

```
la.score(x_test,y_test)
```

Out[91]:

-0.19761713924337143