In [1]:

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

In [2]:

```
data=pd.read_csv(r"C:\Users\user\Downloads\22_countries - 22_countries.csv")
data
```

Out[2]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_na
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan afgl
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	E
2	3	Albania	ALB	AL	8	355	Tirana	ALL	Albaniar
3	4	Algeria	DZA	DZ	12	213	Algiers	DZD	Algerian d
4	5	American Samoa	ASM	AS	16	+1-684	Pago Pago	USD	US D
...	...	...	...	...	...	...	...	...	
245	243	Wallis And Futuna Islands	WLF	WF	876	681	Mata Utu	XPF	CFP fı
246	244	Western Sahara	ESH	EH	732	212	El-Aaiun	MAD	Moroc Dirl
247	245	Yemen	YEM	YE	887	967	Sanaa	YER	Yemen
248	246	Zambia	ZMB	ZM	894	260	Lusaka	ZMW	Zaml kwa
249	247	Zimbabwe	ZWE	ZW	716	263	Harare	ZWL	Zimba D

250 rows × 19 columns

In [3]:
```
df=data.head(100)
df
```

Out[3]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_nan
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan afgha
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	Eu
2	3	Albania	ALB	AL	8	355	Tirana	ALL	Albanian l
3	4	Algeria	DZA	DZ	12	213	Algiers	DZD	Algerian dir
4	5	American Samoa	ASM	AS	16	+1-684	Pago Pago	USD	US Doll
...	...	...	...	...	...	...	...	...	
95	95	Haiti	HTI	HT	332	509	Port-au-Prince	HTG	Haitian gour
96	96	Heard Island and McDonald Islands	HMD	HM	334	672	NaN	AUD	Australian doll
97	97	Honduras	HND	HN	340	504	Tegucigalpa	HNL	Hondur lemp
98	98	Hong Kong S.A.R.	HKG	HK	344	852	Hong Kong	HKD	Hong Ko doll
99	99	Hungary	HUN	HU	348	36	Budapest	HUF	Hungarian for

100 rows × 19 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               100 non-null    int64  
 1   name             100 non-null    object 
 2   iso3             100 non-null    object 
 3   iso2             100 non-null    object 
 4   numeric_code     100 non-null    int64  
 5   phone_code       100 non-null    object 
 6   capital          97 non-null     object 
 7   currency         100 non-null    object 
 8   currency_name    100 non-null    object 
 9   currency_symbol  100 non-null    object 
 10  tld              100 non-null    object 
 11  native           99 non-null     object 
 12  region           98 non-null     object 
 13  subregion        97 non-null     object 
 14  timezones        100 non-null    object 
 15  latitude         100 non-null    float64
 16  longitude        100 non-null    float64
 17  emoji            100 non-null    object 
 18  emojiU           100 non-null    object 
dtypes: float64(2), int64(2), object(15)
memory usage: 15.0+ KB
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['id', 'name', 'iso3', 'iso2', 'numeric_code', 'phone_code', 'capital',
       'currency', 'currency_name', 'currency_symbol', 'tld', 'native',
       'region', 'subregion', 'timezones', 'latitude', 'longitude', 'emoji',
       'emojiU'],
      dtype='object')
```

In [6]:
```
x=df[['id', 'numeric_code','longitude']]
y=df['latitude']
```

In [7]:
```
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 [8]:
```
from sklearn.linear_model import LinearRegression

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

Out[8]:
```
LinearRegression()
```

In [9]:

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

```
8.887026578414687
```

In [10]:

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

Out[10]:

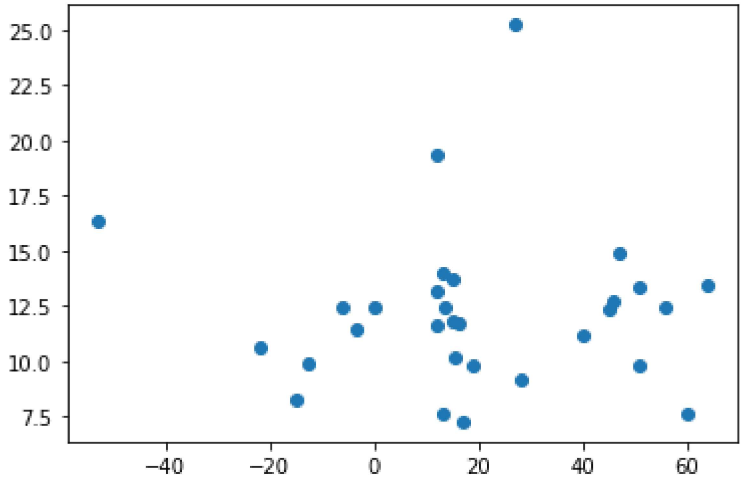
	Co-efficient
id	-0.016889
numeric_code	0.020127
longitude	0.032597

In [11]:

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

Out[11]:

```
<matplotlib.collections.PathCollection at 0x218fcd3a6d0>
```



In [12]:

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

```
-0.09511144376710368
```

In [13]:

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

```
0.01841159242525303
```

In [14]:

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

In [15]:

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

Out[15]:

```
Ridge(alpha=10)
```

In [16]:
```
rr.score(x_test,y_test)
```

Out[16]:
```
-0.0951104728831842
```

In [17]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[17]:
```
Lasso(alpha=10)
```

In [18]:
```
la.score(x_test,y_test)
```

Out[18]:
```
-0.09304615152200091
```

In [19]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[19]:
```
ElasticNet()
```

In [20]:
```
print(en.coef_)
```

```
[-0.01610395  0.0200367   0.03249673]
```

In [21]:
```
print(en.intercept_)
```

```
8.867243374326067
```

In [22]:
```
print(en.predict(x_test))
```

```
[12.46496572 13.68806109 25.18540607 16.37007578  8.24716422  7.65243371
 12.40971061 11.42636648 10.13954912 14.89565752  9.14075814  9.86021571
  7.25893321 12.39858944 11.70028798  9.76499846 14.00781118 13.25673801
 11.63353707 10.62279265 12.73361482 13.44949361 13.36931843 11.75764262
 12.45047297  7.63654999  9.83658603 11.15822055 12.31223461 19.38502354]
```

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

```
-0.09498091505607986
```

Evaluation metrics

In [24]:
```
from sklearn import metrics
```

In [25]:
```
print("Mean absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean absolute error 20.541337352791732
```

```
In [26]: print("Mean squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean squared error 756.1808699709907
```

```
In [27]: print("Mean squared error",np.sqrt(metrics.mean_squared_error(y_test,predictic
```

```
Mean squared error 27.498743061656302
```

Model Saving

```
In [28]: import pickle
```

```
In [29]: filename='prediction'
         pickle.dump(lr,open(filename,'wb'))
```

```
In [ ]:
```