In [1]:

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

In [2]:

```
df1=pd.read_csv(r'C:\Users\user\Downloads\22_countries.csv')
df1
```

Out[2]:

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

250 rows × 19 columns

In [3]:

```
df=df1.head(30)
df
```

Out[3]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan a
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	
2	3	Albania	ALB	AL	8	355	Tirana	ALL	Alban
3	4	Algeria	DZA	DZ	12	213	Algiers	DZD	Algeria
4	5	American Samoa	ASM	AS	16	+1-684	Pago Pago	USD	US
5	6	Andorra	AND	AD	20	376	Andorra la Vella	EUR	
6	7	Angola	AGO	AO	24	244	Luanda	AOA	Angolan k
7	8	Anguilla	AIA	AI	660	+1-264	The Valley	XCD	East Car
8	9	Antarctica	ATA	AQ	10	672	NaN	AAD	Anta
9	10	Antigua And Barbuda	ATG	AG	28	+1-268	St. John's	XCD	E Caribbean
10	11	Argentina	ARG	AR	32	54	Buenos Aires	ARS	Argentin
11	12	Armenia	ARM	AM	51	374	Yerevan	AMD	Armenia
12	13	Aruba	ABW	AW	533	297	Oranjestad	AWG	Aruba
13	14	Australia	AUS	AU	36	61	Canberra	AUD	Australian
14	15	Austria	AUT	AT	40	43	Vienna	EUR	
15	16	Azerbaijan	AZE	AZ	31	994	Baku	AZN	Azer
16	18	Bahrain	BHR	BH	48	973	Manama	BHD	Bahrain
17	19	Bangladesh	BGD	BD	50	880	Dhaka	BDT	Bangl
18	20	Barbados	BRB	BB	52	+1-246	Bridgetown	BBD	Barbadian
19	21	Belarus	BLR	BY	112	375	Minsk	BYN	Belarusia
20	22	Belgium	BEL	BE	56	32	Brussels	EUR	
21	23	Belize	BLZ	BZ	84	501	Belmopan	BZD	Belize
22	24	Benin	BEN	BJ	204	229	Porto-Novo	XOF	West CF

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_
23	25	Bermuda	BMU	BM	60	+1-441	Hamilton	BMD	Berr
24	26	Bhutan	BTN	BT	64	975	Thimphu	BTN	Bhu ng
25	27	Bolivia	BOL	BO	68	591	Sucre	BOB	E bc
26	155	Bonaire, Sint Eustatius and Saba	BES	BQ	535	599	Kralendijk	USD	United
27	28	Bosnia and Herzegovina	BIH	BA	70	387	Sarajevo	BAM	Bosr Herze convertibl
28	29	Botswana	BWA	BW	72	267	Gaborone	BWP	Botswar
29	30	Bouvet Island	BVT	BV	74	0055	NaN	NOK	Nor

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               30 non-null     int64  
 1   name             30 non-null     object 
 2   iso3             30 non-null     object 
 3   iso2             30 non-null     object 
 4   numeric_code     30 non-null     int64  
 5   phone_code       30 non-null     object 
 6   capital          28 non-null     object 
 7   currency         30 non-null     object 
 8   currency_name    30 non-null     object 
 9   currency_symbol  30 non-null     object 
 10  tld              30 non-null     object 
 11  native           30 non-null     object 
 12  region           29 non-null     object 
 13  subregion        28 non-null     object 
 14  timezones        30 non-null     object 
 15  latitude         30 non-null     float64
 16  longitude        30 non-null     float64
 17  emoji            30 non-null     object 
 18  emojiU           30 non-null     object 
dtypes: float64(2), int64(2), object(15)
memory usage: 4.6+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	id	numeric_code	latitude	longitude
count	30.000000	30.000000	30.000000	30.000000
mean	20.100000	110.066667	14.468889	-3.110667
std	26.955071	167.490593	33.010622	63.792168
min	1.000000	4.000000	-74.650000	-170.000000
25%	8.250000	28.750000	-7.000000	-62.825000
50%	15.500000	51.500000	21.125000	4.240000
75%	23.750000	73.500000	40.375000	27.000000
max	155.000000	660.000000	60.116667	133.000000

In [6]:

```
df.columns
```

Out[6]:

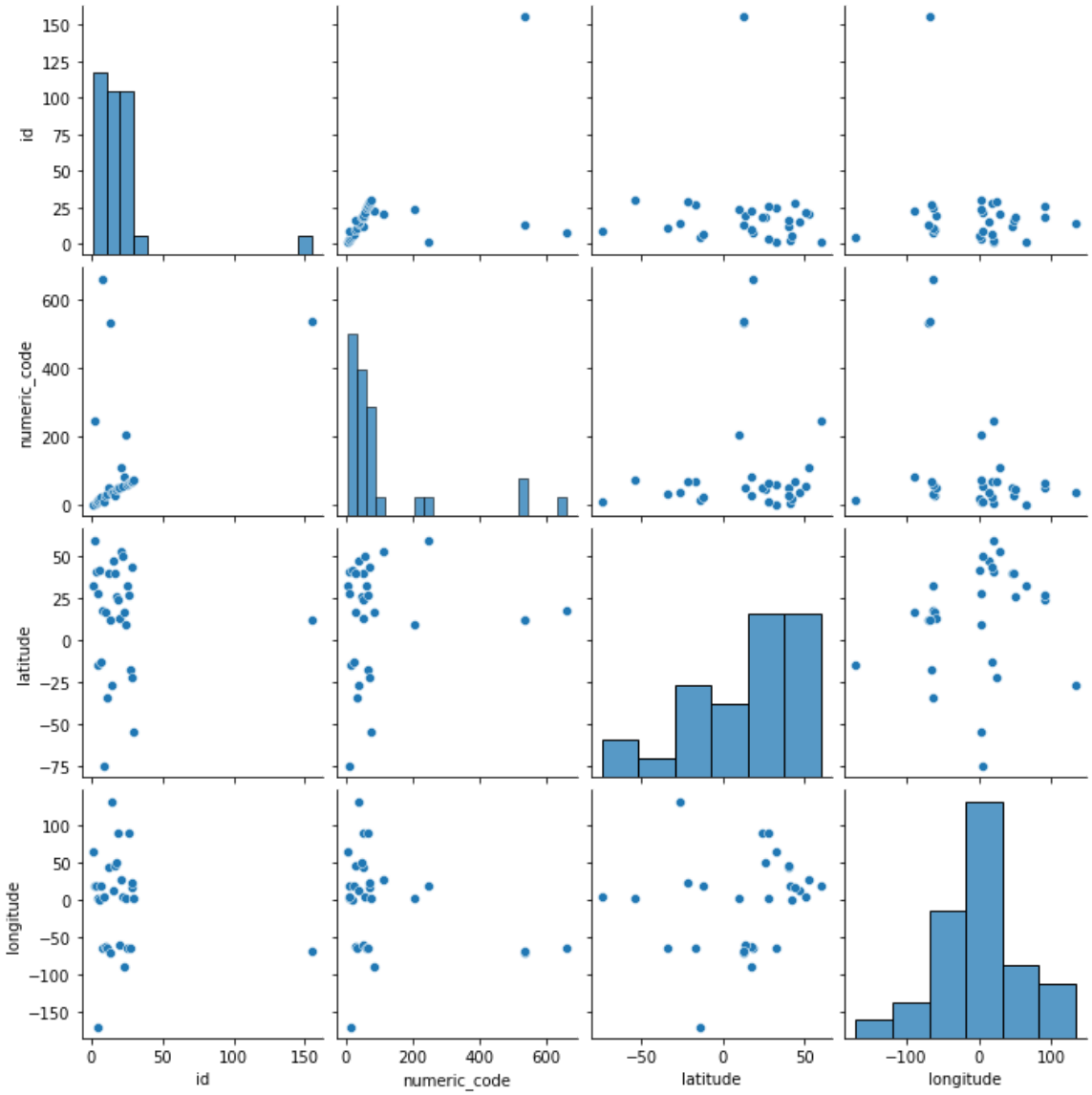
```
Index(['id', 'name', 'iso3', 'iso2', 'numeric_code', 'phone_code', 'capita
l',
       'currency', 'currency_name', 'currency_symbol', 'tld', 'native',
       'region', 'subregion', 'timezones', 'latitude', 'longitude', 'emoj
i',
       'emojiU'],
      dtype='object')
```

In [7]:

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

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x2004fe677f0>
```



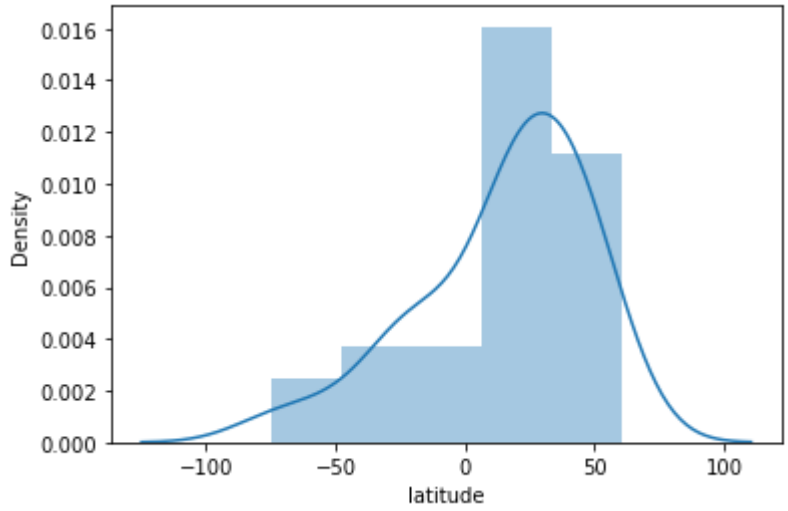
In [8]:

```
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[8]:

```
<AxesSubplot:xlabel='latitude', ylabel='Density'>
```

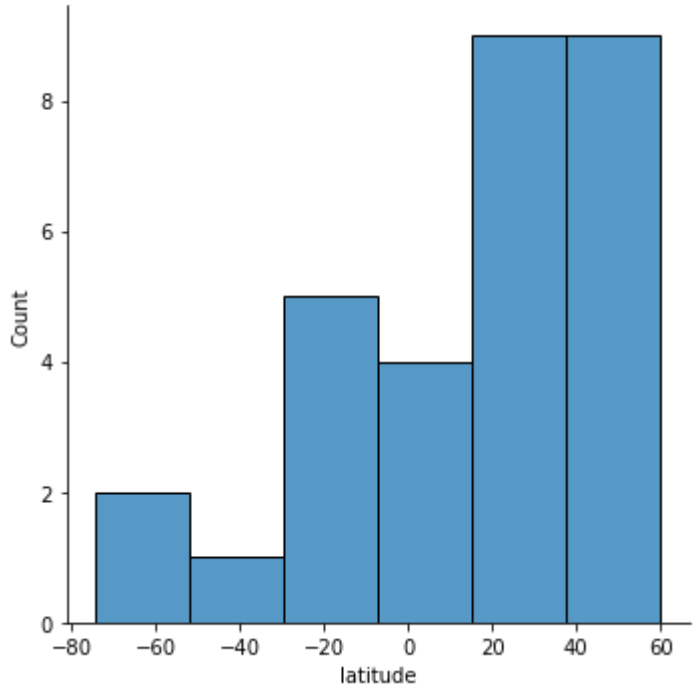


In [9]:

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

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x20051094bb0>
```



In [10]:

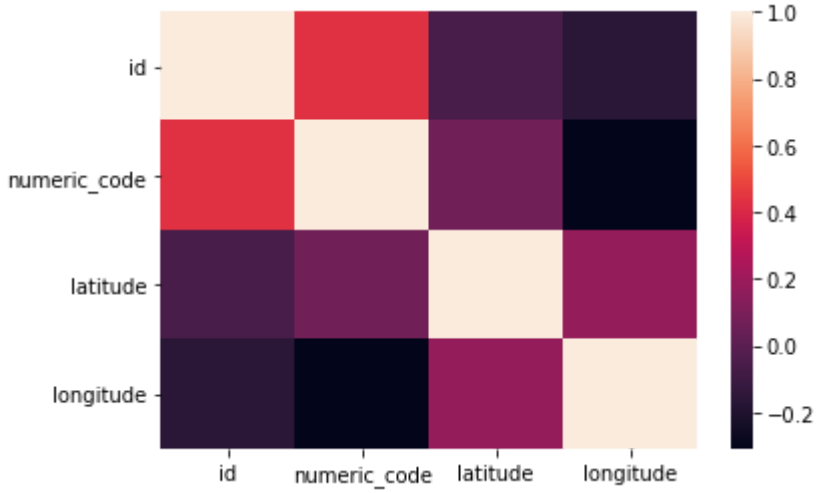
```
df1=df[['id', 'name', 'iso3', 'iso2', 'numeric_code', 'phone_code', 'capital',
       'currency', 'currency_name', 'currency_symbol', 'tld', 'native',
       'region', 'subregion', 'timezones', 'latitude', 'longitude', 'emoji',
       'emojiU']]
```

In [11]:

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

Out[11]:

<AxesSubplot:>



In [12]:

```
x=df1[['id', 'numeric_code', 'latitude', 'longitude']]
y=df1[['latitude']]
```

In [13]:

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

In [14]:

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

In [15]:

```
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[15]:

```
LinearRegression()
```

In [16]:

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

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

In [17]:

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

Out[17]:

	0	1	2	3
0	-2.853429e-17	-3.165907e-17	1.0	3.360846e-18

In [18]:

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

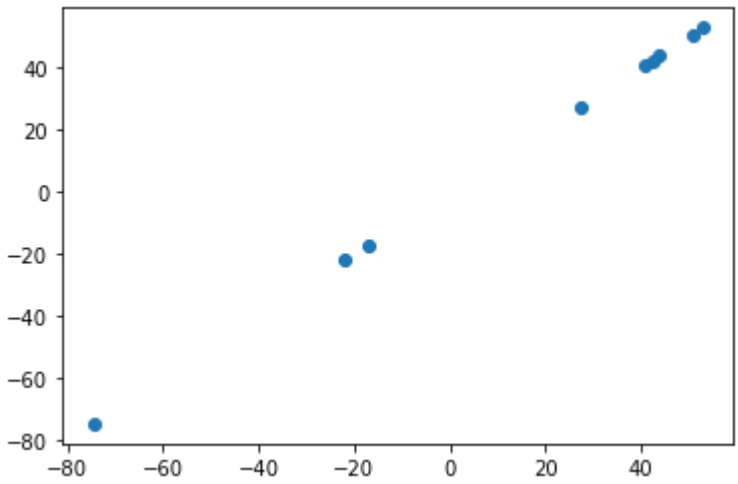
```
1.0
```

In [19]:

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

Out[19]:

```
<matplotlib.collections.PathCollection at 0x20051a1bc70>
```



In [20]:

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

Out[20]:

```
1.0
```

In [21]:

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

Out[21]:

```
1.0
```

In [22]:

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

In [23]:

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

Out[23]:

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

In [24]:

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

Out[24]:

```
0.9999995780047883
```

In [25]:

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

Out[25]:

```
Lasso(alpha=10)
```

In [26]:

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

Out[26]:

```
0.9998269493961823
```

Elastic Net

In [27]:

```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[27]:

```
ElasticNet()
```

In [28]:

```
print(en.coef_)
```

```
[-0.00000000e+00  2.20730930e-06  9.98686234e-01  0.00000000e+00]
```

In [29]:

```
print(en.intercept_)
```

```
[0.017776]
```

In [30]:

```
prediction=en.predict(x_test)
print(prediction)
```

```
[ 40.96392927 -16.95973989  52.94839364  27.48178871 -74.53412932
  50.78444985  43.96012482 -21.95316223  42.46198511]
```

In [31]:

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

```
0.9999982690570989
```

Evaluation Metrics

In [32]:

```
from sklearn import metrics
```

In [33]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.04840338753206789
```

In [34]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.002995709803454769
```

In [35]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 0.05473307778167393
```

Model Saving

In [36]:

```
import pickle
```

In [37]:

```
filename="prediction3"
pickle.dump(lr,open(filename,'wb'))
```