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\20_states.csv')
df1
```

Out[2]:

	id	name	country_id	country_code	country_name	state_code	type	latitu
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.7347
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.1671
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.1789
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.7550
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.8100
...	...	...	...	...	...	...	...	
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	NaN	-17.4851
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	NaN	-20.6241
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	NaN	-18.5331
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	NaN	-21.0523
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	NaN	-19.0552

5077 rows × 9 columns

In [3]:

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

Out[3]:

	id	name	country_id	country_code	country_name	state_code	type	latitude
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007
5	3892	Daykundi	1	AF	Afghanistan	DAY	NaN	33.669495
6	3899	Farah	1	AF	Afghanistan	FRA	NaN	32.495328
7	3889	Faryab	1	AF	Afghanistan	FYB	NaN	36.079561
8	3870	Ghazni	1	AF	Afghanistan	GHA	NaN	33.545059
9	3888	Ghōr	1	AF	Afghanistan	GHO	NaN	34.099578
10	3873	Helmand	1	AF	Afghanistan	HEL	NaN	39.298936
11	3887	Herat	1	AF	Afghanistan	HER	NaN	34.352865
12	3886	Jowzjan	1	AF	Afghanistan	JOW	NaN	36.896969
13	3902	Kabul	1	AF	Afghanistan	KAB	NaN	34.555349
14	3890	Kandahar	1	AF	Afghanistan	KAN	NaN	31.628871
15	3879	Kapisa	1	AF	Afghanistan	KAP	NaN	34.981057
16	3878	Khost	1	AF	Afghanistan	KHO	NaN	33.333847
17	3876	Kunar	1	AF	Afghanistan	KNR	NaN	34.846589
18	3900	Kunduz Province	1	AF	Afghanistan	KDZ	NaN	36.728551
19	3891	Laghman	1	AF	Afghanistan	LAG	NaN	34.689769
20	3897	Logar	1	AF	Afghanistan	LOG	NaN	34.014552
21	3882	Nangarhar	1	AF	Afghanistan	NAN	NaN	34.171831
22	3896	Nimruz	1	AF	Afghanistan	NIM	NaN	31.026149
23	3880	Nuristan	1	AF	Afghanistan	NUR	NaN	35.325022
24	3894	Paktia	1	AF	Afghanistan	PIA	NaN	33.706199
25	3877	Paktika	1	AF	Afghanistan	PKA	NaN	32.264539
26	3881	Panjshir	1	AF	Afghanistan	PAN	NaN	38.880239
27	3895	Parwan	1	AF	Afghanistan	PAR	NaN	34.963098
28	3883	Samangan	1	AF	Afghanistan	SAM	NaN	36.315551
29	3885	Sar-e Pol	1	AF	Afghanistan	SAR	NaN	36.216628

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            30 non-null     int64  
 1   name          30 non-null     object 
 2   country_id    30 non-null     int64  
 3   country_code  30 non-null     object 
 4   country_name  30 non-null     object 
 5   state_code    30 non-null     object 
 6   type          0 non-null      object 
 7   latitude      30 non-null     float64
 8   longitude     30 non-null     float64
dtypes: float64(2), int64(2), object(5)
memory usage: 2.2+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	id	country_id	latitude	longitude
count	30.000000	30.0	30.000000	30.000000
mean	3885.766667	1.0	34.924384	57.902269
std	9.565323	0.0	1.897661	36.735298
min	3870.000000	1.0	31.026149	-77.171724
25%	3878.250000	1.0	33.783287	65.095930
50%	3885.500000	1.0	34.828298	68.190842
75%	3893.500000	1.0	36.207197	69.339202
max	3902.000000	1.0	39.298936	71.097317

In [6]:

```
df.columns
```

Out[6]:

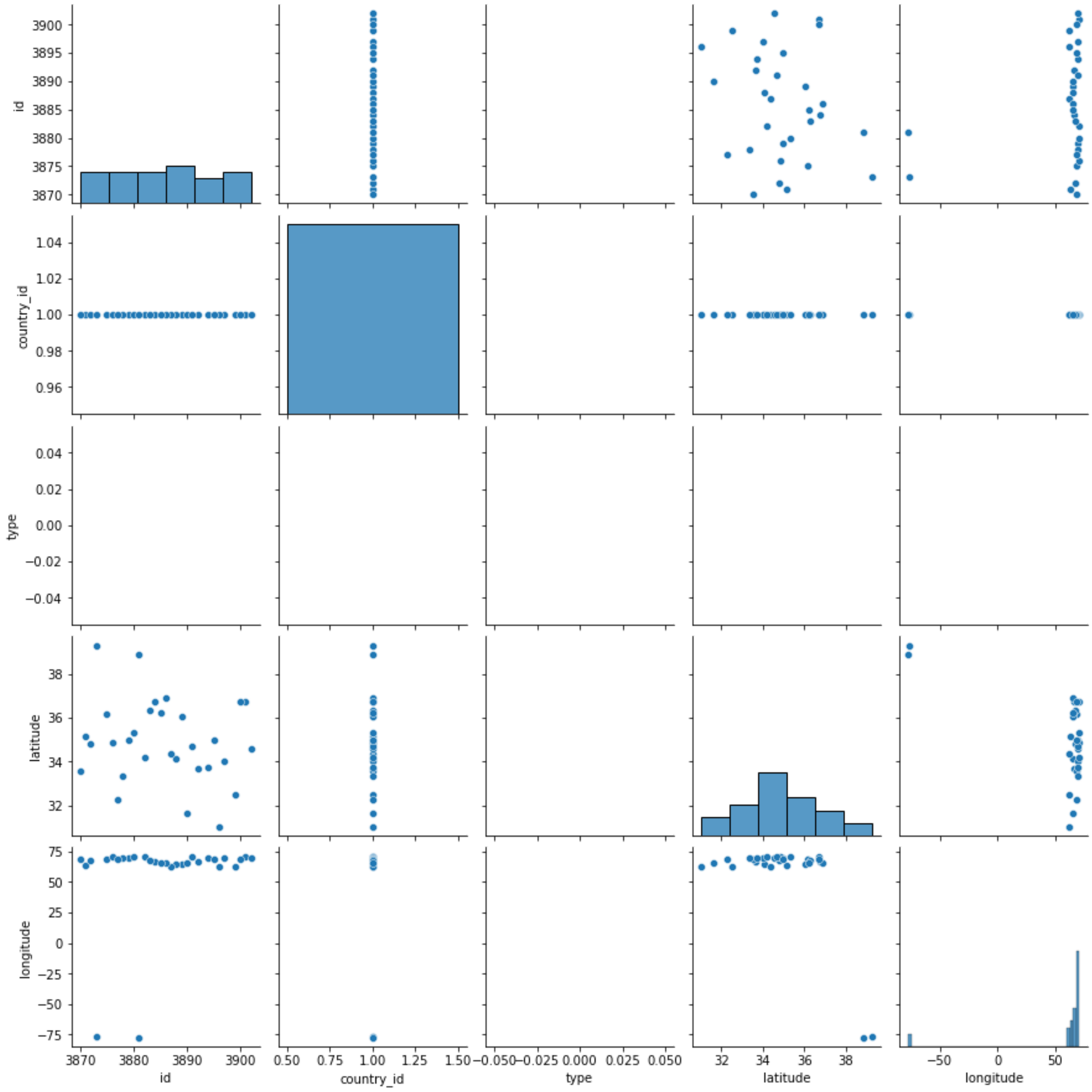
```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

In [7]:

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

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x2a44a7b3ca0>
```



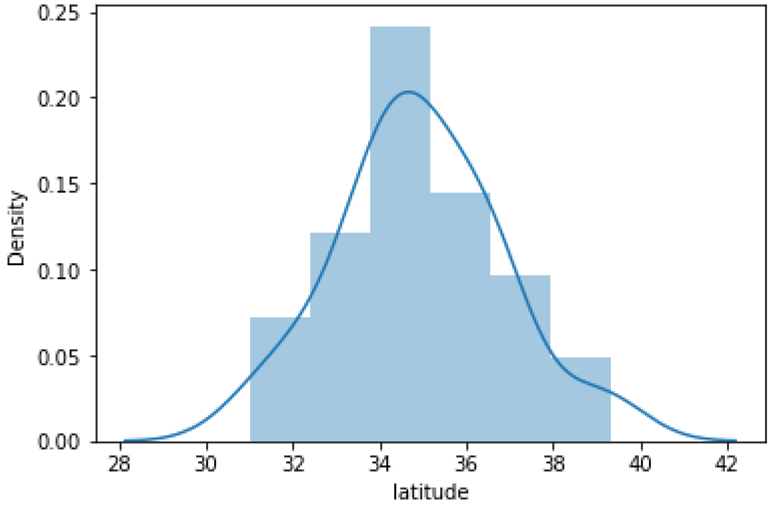
In [8]:

```
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'>
```

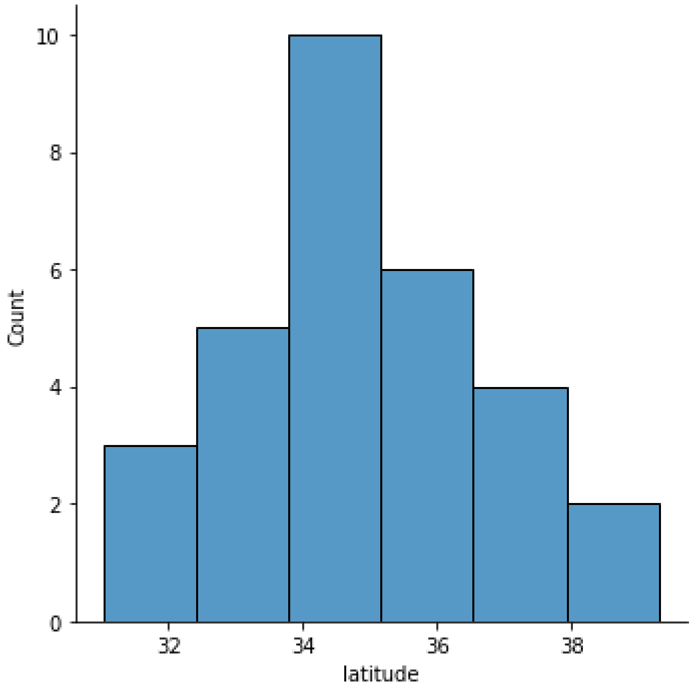


In [9]:

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

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x2a44c0022e0>
```



In [10]:

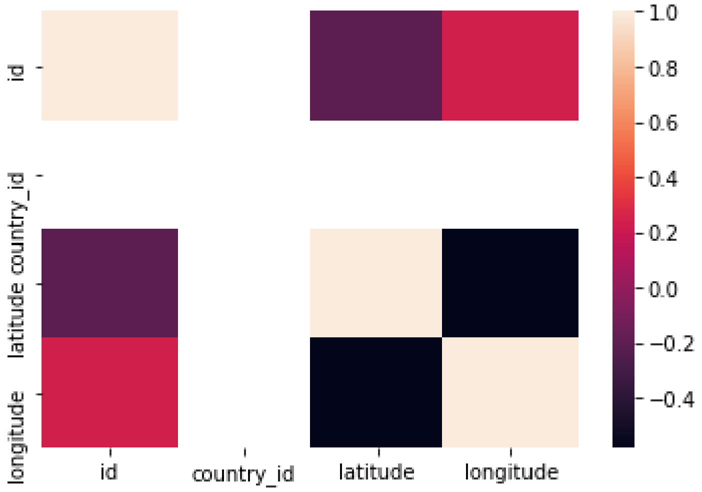
```
df1=df[['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude']]
```

In [11]:

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

Out[11]:

```
<AxesSubplot:>
```



In [12]:

```
x=df1[['id', 'country_id', '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_)
```

```
[22.30676221]
```

In [17]:

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

Out[17]:

	0	1	2
0	0.003703	0.0	-0.030379

In [18]:

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

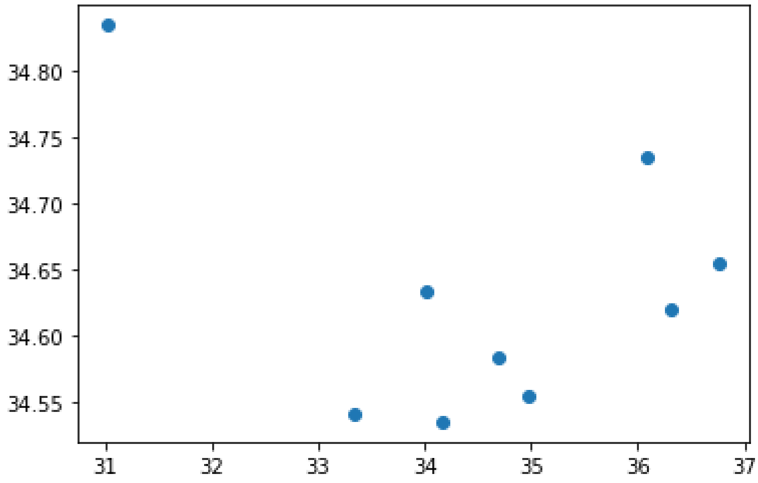
```
-0.037719693477049177
```

In [19]:

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

Out[19]:

```
<matplotlib.collections.PathCollection at 0x2a44c96fdc0>
```



In [20]:

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

Out[20]:

```
-0.037719693477049177
```

In [21]:

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

Out[21]:

```
0.4390960068802542
```

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]:

```
-0.037624337075771175
```

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]:

```
-0.024796356849838608
```

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.          0.         -0.02985577]
```

In [29]:

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

```
[36.66484629]
```

In [30]:

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

```
[34.57059598 34.6357202  34.58624412 34.80034106 34.55638168 34.66756882
 34.59905417 34.72702904 34.57681832]
```

In [31]:

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

```
-0.025235182449752758
```

Evaluation Metrics

In [32]:

```
from sklearn import metrics
```

In [33]:

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

```
Mean Absolute Error: 1.2911173425965217
```

In [34]:

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

```
Mean Squared Error: 2.8286555938239157
```

In [35]:

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

```
Root Mean Squared Error: 1.681860753399019
```

Model Saving

In [36]:

```
import pickle
```

In [37]:

```
filename="prediction1"
pickle.dump(lr,open(filename,'wb'))
```