In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [128]:
```
s=pd.read_csv(r"C:\Users\user\Downloads\21_cities - 21_cities.csv")
s
```

Out[128]:

	id	name	state_id	state_code	state_name	country_id	country_code	country
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afgha
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afgha
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afgha
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afgha
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afgha
...	...	...	...	...	...	...	...	
150449	131496	Redcliff	1957	MI	Midlands Province	247	ZW	Zim
150450	131502	Shangani	1957	MI	Midlands Province	247	ZW	Zim
150451	131503	Shurugwi	1957	MI	Midlands Province	247	ZW	Zim
150452	131504	Shurugwi District	1957	MI	Midlands Province	247	ZW	Zim
150453	131508	Zvishavane District	1957	MI	Midlands Province	247	ZW	Zim

150454 rows × 11 columns

In [129]:
```
s=s.head(50)
s
```

Out[129]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_name
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanistan
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanistan
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanistan
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanistan
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanistan
5	131	Wākhān	3901	BDS	Badakhshan	1	AF	Afghanistan
6	72	Ghormach	3871	BDG	Badghis	1	AF	Afghanistan
7	108	Qala i Naw	3871	BDG	Badghis	1	AF	Afghanistan
8	54	Baghlān	3875	BGL	Baghlan	1	AF	Afghanistan
9	140	Ḩukūmatī Dahanah-ye Ghōrī	3875	BGL	Baghlan	1	AF	Afghanistan
10	101	Nahrīn	3875	BGL	Baghlan	1	AF	Afghanistan
11	105	Pul-e Khumrī	3875	BGL	Baghlan	1	AF	Afghanistan
12	55	Balkh	3884	BAL	Balkh	1	AF	Afghanistan
13	65	Dowlatābād	3884	BAL	Balkh	1	AF	Afghanistan
14	85	Khulm	3884	BAL	Balkh	1	AF	Afghanistan
15	91	Lab-Sar	3884	BAL	Balkh	1	AF	Afghanistan
16	97	Mazār-e Sharīf	3884	BAL	Balkh	1	AF	Afghanistan
17	112	Qarchī Gak	3884	BAL	Balkh	1	AF	Afghanistan
18	57	Bāmyān	3872	BAM	Bamyan	1	AF	Afghanistan
19	104	Panjāb	3872	BAM	Bamyan	1	AF	Afghanistan
20	102	Nīlī	3892	DAY	Daykundi	1	AF	Afghanistan
21	66	Farah	3899	FRA	Farah	1	AF	Afghanistan
22	50	Andkhoy	3889	FYB	Faryab	1	AF	Afghanistan
23	96	Maymana	3889	FYB	Faryab	1	AF	Afghanistan
24	71	Ghazni	3870	GHA	Ghazni	1	AF	Afghanistan
25	67	Fayrōz Kōh	3888	GHO	Ghōr	1	AF	Afghanistan
26	121	Shahrak	3888	GHO	Ghōr	1	AF	Afghanistan
27	141	ʻAlāqahdārī Dīshū	3873	HEL	Helmand	1	AF	Afghanistan
28	70	Gereshk	3873	HEL	Helmand	1	AF	Afghanistan
29	93	Lashkar Gāh	3873	HEL	Helmand	1	AF	Afghanistan
30	95	Markaz-e Ḩukūmat-e Darwēshān	3873	HEL	Helmand	1	AF	Afghanistan

	id	name	state_id	state_code	state_name	country_id	country_code	country_name
31	118	Sangīn	3873	HEL	Helmand	1	AF	Afghanistan
32	60	Chahār Burj	3887	HER	Herat	1	AF	Afghanistan
33	73	Ghōriyān	3887	HER	Herat	1	AF	Afghanistan
34	74	Herāt	3887	HER	Herat	1	AF	Afghanistan
35	80	Kafir Qala	3887	HER	Herat	1	AF	Afghanistan
36	82	Karukh	3887	HER	Herat	1	AF	Afghanistan
37	88	Kuhsān	3887	HER	Herat	1	AF	Afghanistan
38	90	Kushk	3887	HER	Herat	1	AF	Afghanistan
39	111	Qarah Bāgh	3887	HER	Herat	1	AF	Afghanistan
40	123	Shīnḏanḏ	3887	HER	Herat	1	AF	Afghanistan
41	129	Tīr Pul	3887	HER	Herat	1	AF	Afghanistan
42	135	Zindah Jān	3887	HER	Herat	1	AF	Afghanistan
43	136	Āqchah	3886	JOW	Jowzjan	1	AF	Afghanistan
44	63	Darzāb	3886	JOW	Jowzjan	1	AF	Afghanistan
45	113	Qarqīn	3886	JOW	Jowzjan	1	AF	Afghanistan
46	122	Shibirghān	3886	JOW	Jowzjan	1	AF	Afghanistan
47	79	Kabul	3902	KAB	Kabul	1	AF	Afghanistan
48	99	Mīr Bachah Kōṯ	3902	KAB	Kabul	1	AF	Afghanistan
49	103	Paghmān	3902	KAB	Kabul	1	AF	Afghanistan

In [136]:
```
s.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            50 non-null     int64  
 1   name          50 non-null     object 
 2   state_id      50 non-null     int64  
 3   state_code    50 non-null     object 
 4   state_name    50 non-null     object 
 5   country_id    50 non-null     int64  
 6   country_code  50 non-null     object 
 7   country_name  50 non-null     object 
 8   latitude      50 non-null     float64
 9   longitude     50 non-null     float64
 10  wikiDataId    50 non-null     object 
dtypes: float64(2), int64(3), object(6)
memory usage: 4.4+ KB
```

In [137]:
```
s.describe()
```

Out[137]:

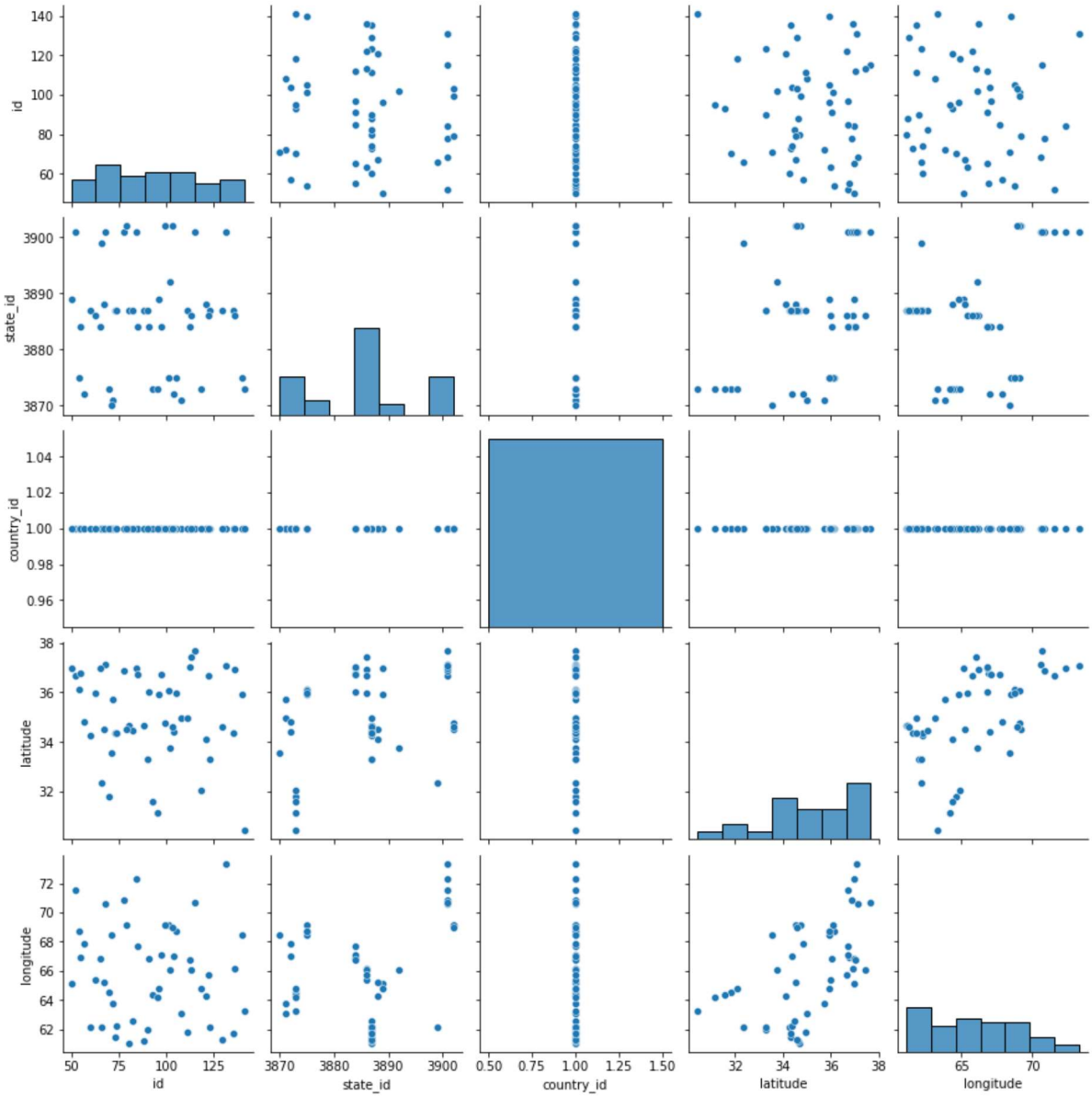
	id	state_id	country_id	latitude	longitude
count	50.000000	50.000000	50.0	50.000000	50.000000
mean	92.480000	3885.660000	1.0	35.056388	65.916456
std	25.584785	9.902813	0.0	1.767959	3.243999
min	50.000000	3870.000000	1.0	30.432060	61.066670
25%	71.250000	3875.000000	1.0	34.343180	63.171188
50%	92.000000	3887.000000	1.0	34.880895	65.898240
75%	111.750000	3888.750000	1.0	36.693853	68.471757
max	141.000000	3902.000000	1.0	37.660790	73.349280

In [138]:
```
s.columns
```

Out[138]:
```
Index(['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataI
d'],
      dtype='object')
```

In [139]:
```
sns.pairplot(s)
```

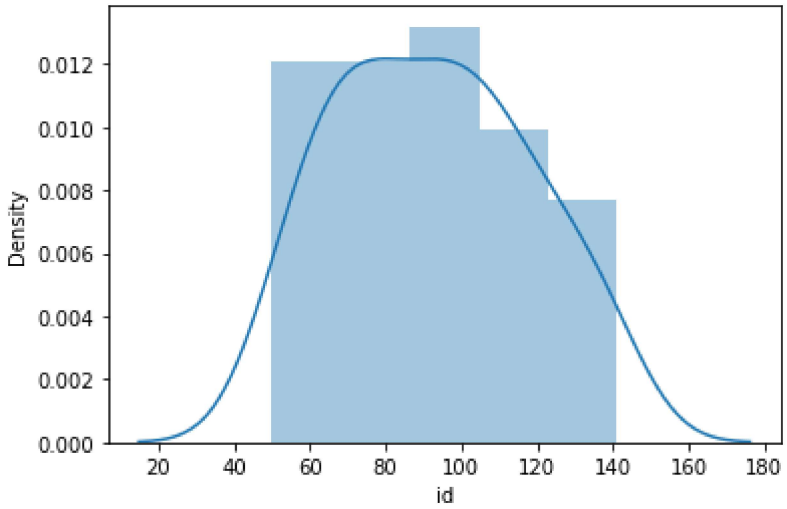
Out[139]: <seaborn.axisgrid.PairGrid at 0x23cf9d69670>



In [140]:
```
sns.distplot(s['id'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[140]: <AxesSubplot:xlabel='id', ylabel='Density'>



In [141]:
```
s1=s[['id','country_id','latitude', 'longitude']]
s1
```

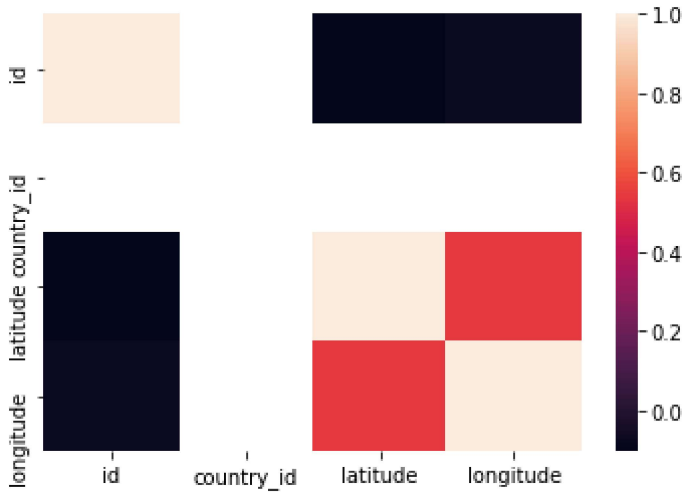
Out[141]:

	id	country_id	latitude	longitude
0	52	1	36.68333	71.53333
1	68	1	37.11664	70.58002
2	78	1	36.86477	70.83421
3	84	1	36.95127	72.31800
4	115	1	37.66079	70.67346
5	131	1	37.05710	73.34928
6	72	1	35.73062	63.78264
7	108	1	34.98735	63.12891
8	54	1	36.13068	68.70829
9	140	1	35.90617	68.48869
10	101	1	36.06490	69.13343
11	105	1	35.94458	68.71512
12	55	1	36.75635	66.89720
13	65	1	36.98821	66.82069
14	85	1	36.69736	67.69826
15	91	1	36.02634	66.83799
16	97	1	36.70904	67.11087
17	112	1	37.03999	66.78891
18	57	1	34.82156	67.82734
19	104	1	34.38795	67.02327
20	102	1	33.76329	66.07617
21	66	1	32.37451	62.11638
22	50	1	36.95293	65.12376
23	96	1	35.92139	64.78361
24	71	1	33.55391	68.42096
25	67	1	34.51952	65.25093
26	121	1	34.10737	64.30520
27	141	1	30.43206	63.29802
28	70	1	31.82089	64.57005
29	93	1	31.59382	64.37161
30	95	1	31.13231	64.19340
31	118	1	32.07275	64.83590
32	60	1	34.24475	62.19165
33	73	1	34.34480	61.49321
34	74	1	34.34817	62.19967

	id	country_id	latitude	longitude
35	80	1	34.66667	61.06667
36	82	1	34.48108	62.58630
37	88	1	34.65389	61.19778
38	90	1	33.29565	61.95221
39	111	1	34.94023	61.77589
40	123	1	33.30294	62.14740
41	129	1	34.59431	61.26895
42	135	1	34.34264	61.74675
43	136	1	36.90500	66.18341
44	63	1	35.97744	65.37828
45	113	1	37.41853	66.04358
46	122	1	36.66757	65.75290
47	79	1	34.52813	69.17233
48	99	1	34.74999	69.11899
49	103	1	34.58787	68.95091

In [142]:
```
sns.heatmap(s1.corr())
```

Out[142]: <AxesSubplot:>



In [143]:
```
x=s1[['id','country_id','latitude', 'longitude']]
y=s1['id']
```

In [144]:
```
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 [145]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[145]:
```
LinearRegression()
```

In [146]:
```
lr.intercept_
```

Out[146]:
```
4.263256414560601e-14
```

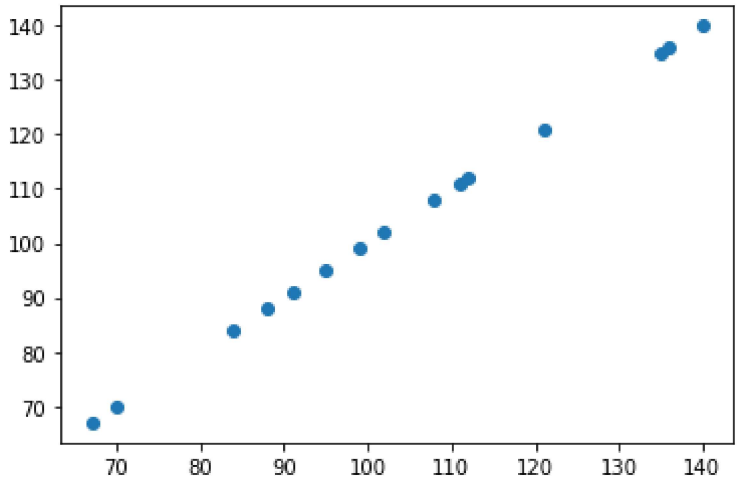
In [147]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[147]:

	Co-efficient
id	1.000000e+00
country_id	0.000000e+00
latitude	-9.350533e-17
longitude	-4.946788e-17

In [148]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[148]:
```
<matplotlib.collections.PathCollection at 0x23cfada0250>
```



In [149]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [150]:
```
from sklearn.linear_model import Ridge,Lasso
from sklearn.linear_model import Ridge,Lasso
```

In [151]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[151]: 0.9999995883238496

In [152]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_test,y_test)
```

Out[152]: 0.9996086583834574

In [153]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[153]: ElasticNet()

In [154]:
```
print(en.coef_)
```

```
[ 0.99841787  0.         -0.         -0.        ]
```

In [155]:
```
print(en.intercept_)
```

```
0.13854941251302932
```

In [156]:
```
print(en.predict(x_test))
```

```
[ 98.98191851 110.96293295  84.00565047 135.92337969  67.03254668
 120.94711164  87.99932194 111.96135082 101.97717212  70.02780029
 134.92496182 139.91705117  94.98824703  90.99457555 107.96767934]
```

In [157]:
```
print(en.score(x_test,y_test))
```

```
0.99999609277292
```

In [158]:
```
from sklearn import metrics
```

In [159]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 1.9895196601282804e-14
```

In [160]:
```
print("Mean squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean squared Error 7.943303408305441e-28
```

In [161]:
```
import pickle
```

In [162]:
```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [163]:
```
import pandas as pd
import pickle
```

In [164]:
```
filename='prediction'
model=pickle.load(open(filename,'rb'))
```

In [168]:
```
real=[[55,200,50,40],[80,55,66,88]]
result=model.predict(real)
```

In [166]:
```
result
```

Out[166]:
```
array([10., 40.])
```

In [ ]:
```

```