In [249]:

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

In [250]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\21_cities.csv")
a
```

Out[250]:

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

150454 rows × 11 columns

In [251]:

```
a=a.head(10)
a
```

Out[251]:

	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

In [252]:

```
# to find
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            10 non-null     int64  
 1   name          10 non-null     object 
 2   state_id      10 non-null     int64  
 3   state_code    10 non-null     object 
 4   state_name    10 non-null     object 
 5   country_id    10 non-null     int64  
 6   country_code  10 non-null     object 
 7   country_name  10 non-null     object 
 8   latitude      10 non-null     float64
 9   longitude     10 non-null     float64
 10  wikiDataId    10 non-null     object 
dtypes: float64(2), int64(3), object(6)
memory usage: 1008.0+ bytes
```

In [253]:

```
# to display summary of statastic
a.describe()
```

Out[253]:

	id	state_id	country_id	latitude	longitude
count	10.000000	10.000000	10.0	10.000000	10.000000
mean	90.200000	3889.800000	1.0	36.508872	69.339683
std	31.371608	14.520484	0.0	0.801155	3.430057
min	52.000000	3871.000000	1.0	34.987350	63.128910
25%	69.000000	3875.000000	1.0	35.962298	68.543590
50%	81.000000	3901.000000	1.0	36.774050	70.626740
75%	113.250000	3901.000000	1.0	37.030642	71.358550
max	140.000000	3901.000000	1.0	37.660790	73.349280

In [254]:

```
# to display colum heading
a.columns
```

Out[254]:

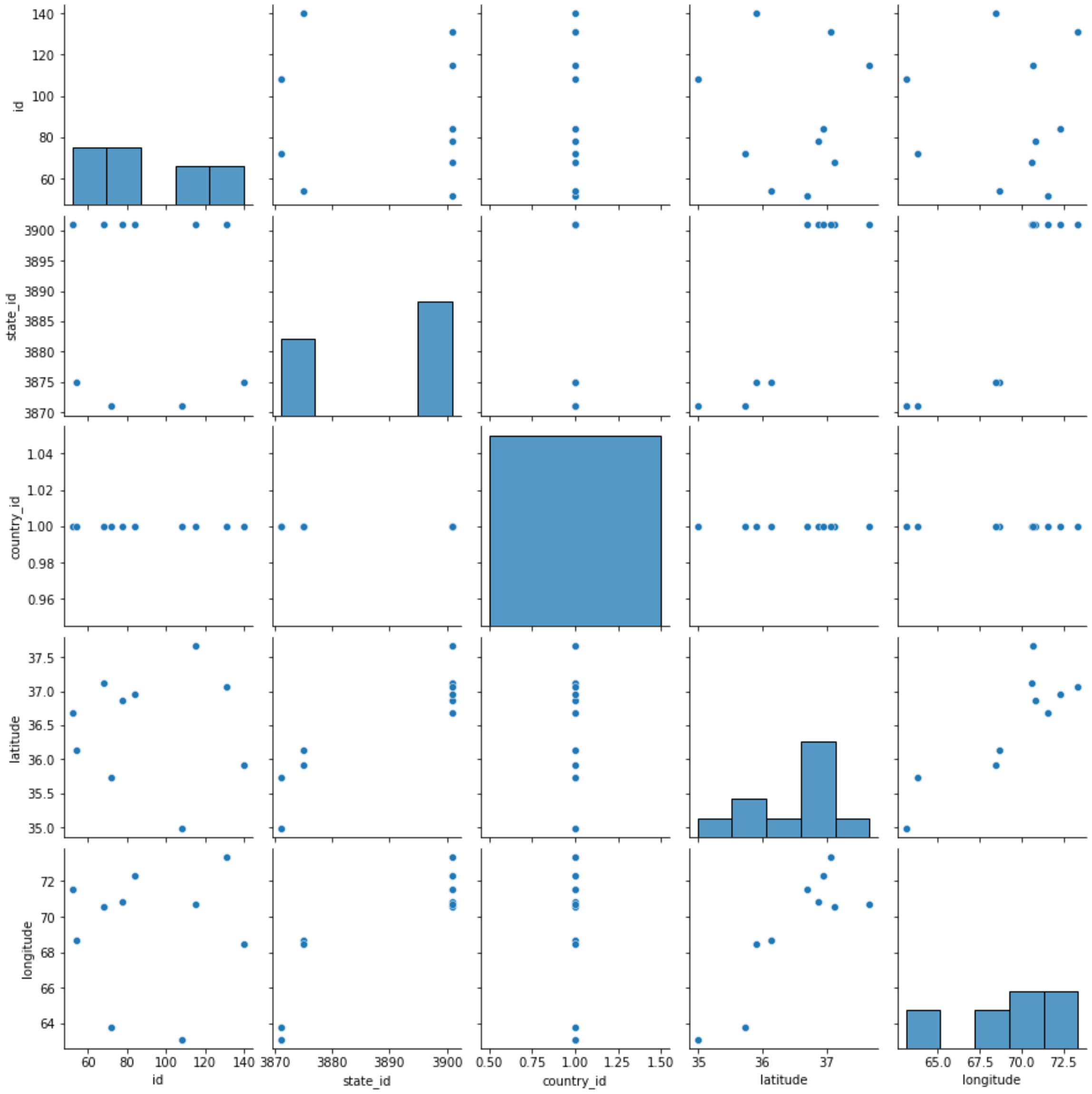
```
Index(['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataI
d'],
      dtype='object')
```

In [256]:

```
sns.pairplot(a)
```

Out[256]:

```
<seaborn.axisgrid.PairGrid at 0x243fb8f4cd0>
```

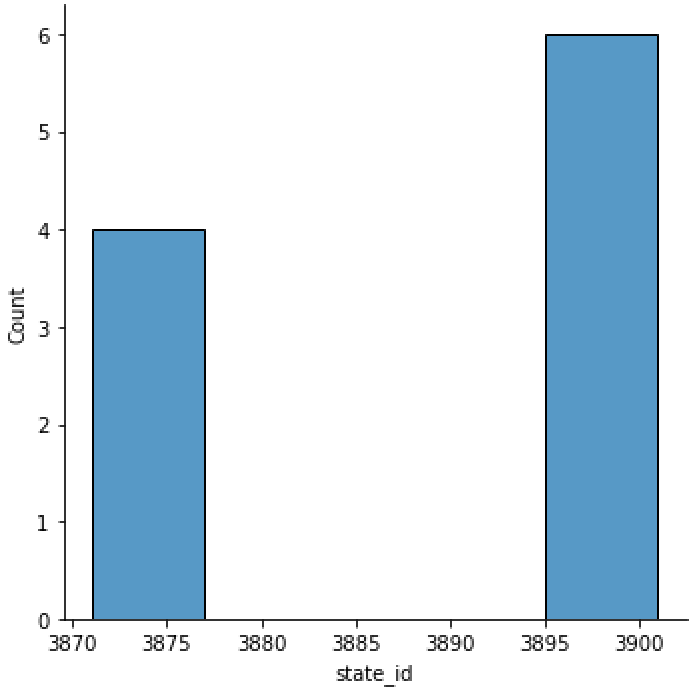


In [257]:

```
sns.displot(a["state_id"])
```

Out[257]:

```
<seaborn.axisgrid.FacetGrid at 0x243fc5e2d30>
```



In [258]:

```
b=a[['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude']]
b
```

Out[258]:

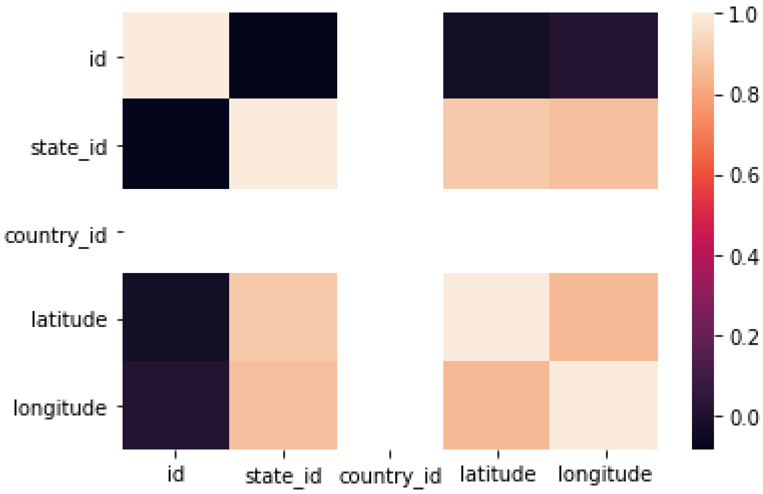
	id	name	state_id	state_code	state_name	country_id	country_code	country_nam
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanista
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanista
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanista
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanista
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanista
5	131	Wākhān	3901	BDS	Badakhshan	1	AF	Afghanista
6	72	Ghormach	3871	BDG	Badghis	1	AF	Afghanista
7	108	Qala i Naw	3871	BDG	Badghis	1	AF	Afghanista
8	54	Baghlān	3875	BGL	Baghlan	1	AF	Afghanista
9	140	Ḩukūmatī Dahanah-ye Ghōrī	3875	BGL	Baghlan	1	AF	Afghanista

In [259]:

```
sns.heatmap(b.corr())
```

Out[259]:

```
<AxesSubplot:>
```



In [288]:

```
x=a[['id','state_id', 'country_id','latitude', 'longitude']]
y=a['latitude']
```

In [289]:

```
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 [290]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[290]:

```
LinearRegression()
```

In [291]:

```
lr.intercept_
```

Out[291]:

```
-4.973799150320701e-14
```

In [292]:

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

Out[292]:

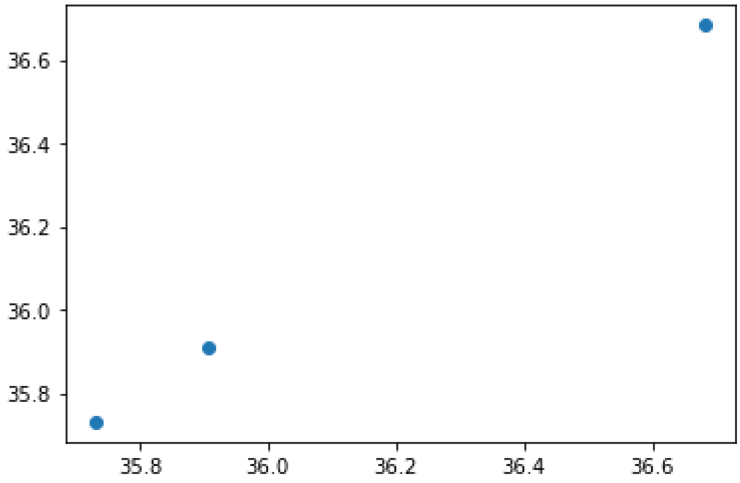
	Co-efficient
id	-8.201756e-19
state_id	1.422498e-17
country_id	1.276756e-15
latitude	1.000000e+00
longitude	-1.427683e-16

In [293]:

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

Out[293]:

```
<matplotlib.collections.PathCollection at 0x243fcd50df0>
```



In [294]:

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

Out[294]:

```
1.0
```

In [295]:

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

Out[295]:

```
1.0
```

In [296]:

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

In [297]:

```
rr=Ridge(alpha=10)
rr.fit(x_test,y_test)
```

Out[297]:

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

In [298]:

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

Out[298]:

```
0.9995385518751715
```

In [299]:

```
la=Lasso(alpha=10)
la.fit(x_test,y_test)
```

Out[299]:

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

In [300]:

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

Out[300]:

```
0.0
```

In [301]:

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

Out[301]:

```
ElasticNet()
```

In [302]:

```
en.coef_
```

Out[302]:

```
array([-0.00130246,  0.05499757,  0.        ,  0.        ,  0.        ])
```

In [303]:

```
en.intercept_
```

Out[303]:

```
-177.3056023416714
```

In [304]:

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

Out[304]:

```
array([35.62763794, 35.49621479, 37.17219106])
```

In [305]:

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

Out[305]:

```
0.27742758376612886
```

EVALUATION METRICS

In [306]:

```
from sklearn import metrics
```

In [307]:

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

```
Mean Absolute Error: 0.33393277570921026
```

In [308]:

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

```
Mean Squared Error 0.12383701476285075
```

In [309]:

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

```
Root Mean Squared Error 0.35190483765195774
```

MODEL SAVING

In [310]:

```
import pickle
```

In [311]:

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

In [312]:

```
import pandas as pd
import pickle
```

In [313]:

```
filename='prediction'
model=pickle.load(open(filename,'rb'))
```

In [315]:

```
real=[[10,20,30,40,50],[13,23,33,43,56]]
result=model.predict(real)
result
```

Out[315]:

```
array([40., 43.])
```

In [ ]:

```

```