In [210]:

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

In [211]:

```python
a=pd.read_csv(r"C:\Users\user\Downloads\20_states.csv")
a
```

Out[211]:

	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 [212]:

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

Out[212]:

	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

In [213]:

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            10 non-null     int64  
 1   name          10 non-null     object 
 2   country_id    10 non-null     int64  
 3   country_code  10 non-null     object 
 4   country_name  10 non-null     object 
 5   state_code    10 non-null     object 
 6   type          0 non-null      object 
 7   latitude      10 non-null     float64
 8   longitude     10 non-null     float64
dtypes: float64(2), int64(2), object(5)
memory usage: 848.0+ bytes
```

In [214]:

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

Out[214]:

	id	country_id	latitude	longitude
count	10.000000	10.0	10.000000	10.000000
mean	3884.100000	1.0	34.953490	66.458391
std	11.589746	0.0	1.477933	2.579742
min	3870.000000	1.0	32.495328	62.262663
25%	3872.750000	1.0	33.777016	64.905955
50%	3886.000000	1.0	34.988570	66.471945
75%	3891.250000	1.0	36.154067	68.268350
max	3901.000000	1.0	36.755060	70.811995

In [215]:

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

Out[215]:

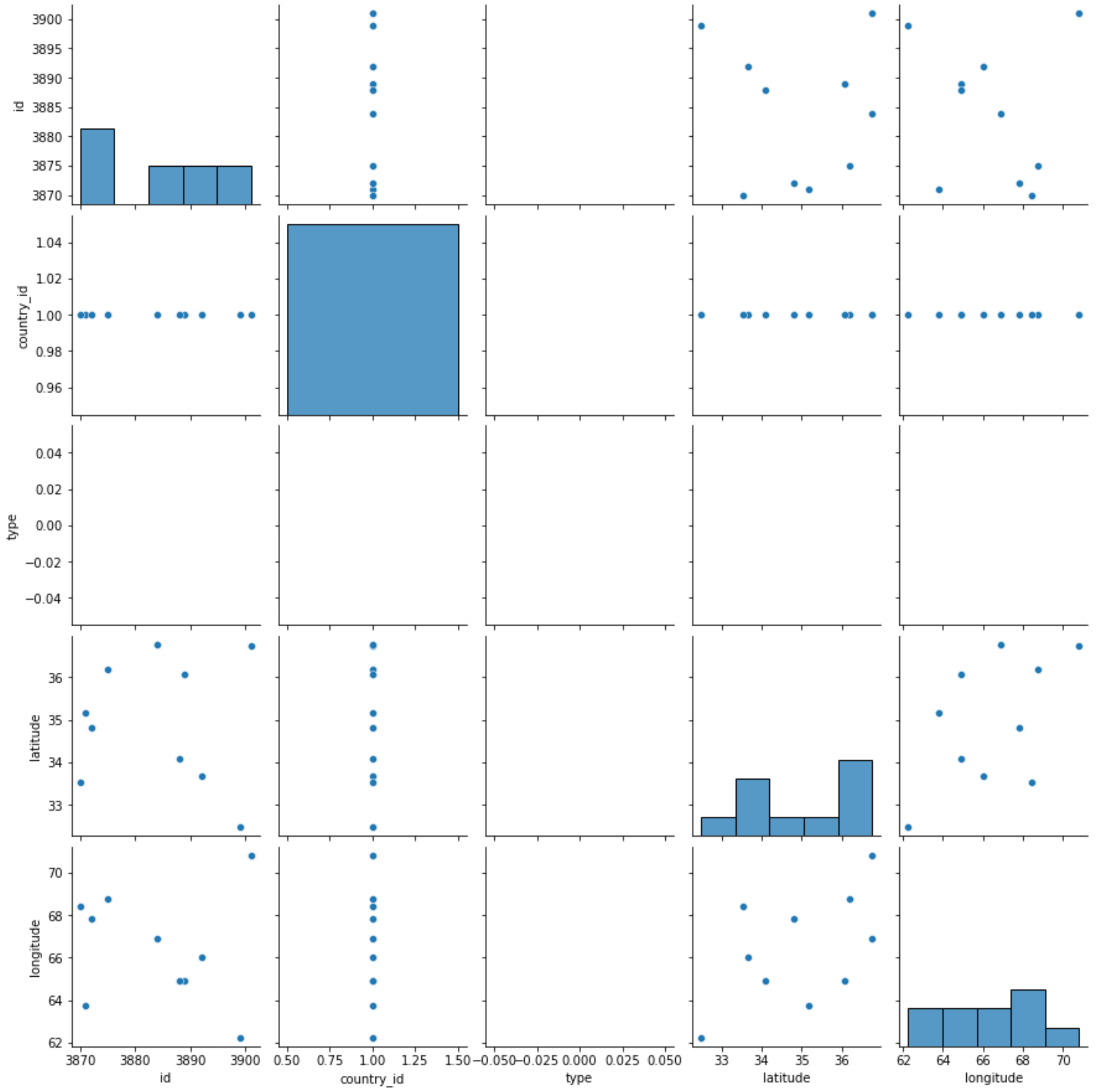
```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

In [216]:

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

Out[216]:

```
<seaborn.axisgrid.PairGrid at 0x243f8ba8d60>
```

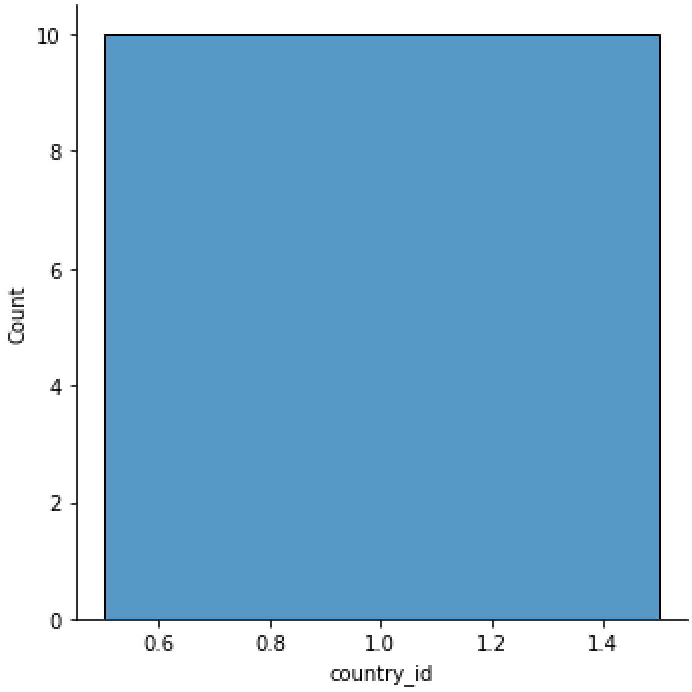


In [217]:

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

Out[217]:

```
<seaborn.axisgrid.FacetGrid at 0x243f968f730>
```



In [218]:

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

Out[218]:

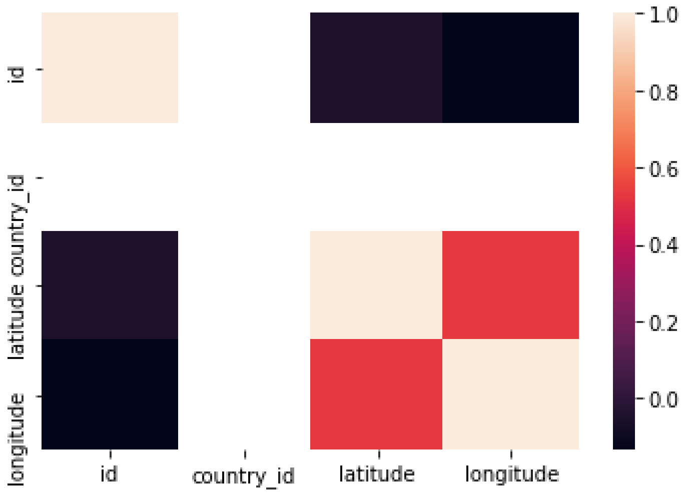
	id	name	country_id	country_code	country_name	state_code	type	latitude
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060
4	3872	Bamyan	1	AF	Afghanistan	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

In [219]:

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

Out[219]:

```
<AxesSubplot:>
```



In [221]:

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

In [222]:

```
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 [223]:

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

Out[223]:

```
LinearRegression()
```

In [224]:

```
lr.intercept_
```

Out[224]:

```
3.552713678800501e-14
```

In [225]:

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

Out[225]:

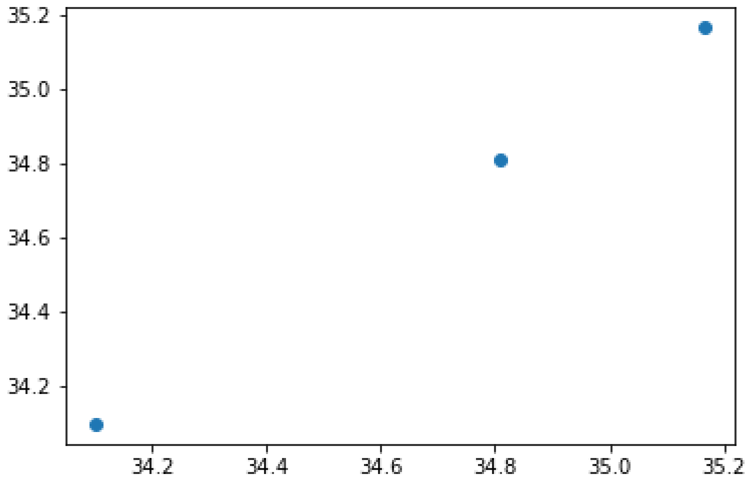
	Co-efficient
id	-7.797727e-18
country_id	7.771561e-16
latitude	1.000000e+00
longitude	-3.325573e-17

In [226]:

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

Out[226]:

```
<matplotlib.collections.PathCollection at 0x243f9c71a90>
```



In [227]:

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

Out[227]:

```
1.0
```

In [228]:

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

Out[228]:

```
1.0
```

In [229]:

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

In [230]:

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

Out[230]:

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

In [231]:

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

Out[231]:

```
0.9733158079069464
```

In [232]:

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

Out[232]:

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

In [233]:

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

Out[233]:

```
0.0
```

In [234]:

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

Out[234]:

```
ElasticNet()
```

In [235]:

```
en.coef_
```

Out[235]:

```
array([0.        , 0.        , 0.64476958, 0.0525972 ])
```

In [236]:

```
en.intercept_
```

Out[236]:

```
8.939161964166033
```

In [237]:

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

Out[237]:

```
array([34.33940401, 34.95080137, 34.9679595 ])
```

In [238]:

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

Out[238]:

```
0.8018930236223091
```

EVALUATION METRICS

In [239]:

```
from sklearn import metrics
```

In [240]:

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

```
Mean Absolute Error: 0.19326515931722335
```

In [241]:

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

```
Mean Squared Error 0.039003429107221245
```

In [242]:

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

```
Root Mean Squared Error 0.19749285837017308
```

MODEL SAVING

In [243]:

```
import pickle
```

In [244]:

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

In [245]:

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

In [246]:

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

In [248]:

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

Out[248]:

```
array([30., 33.])
```

In [ ]:

```

```