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

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\20_states.csv")
df.fillna(0,inplace=True)
df
```

Out[2]:

	id	name	country_id	country_code	country_name	state_code	type	latitude
0	3901	Badakhshan	1	AF	Afghanistan	BDS	0	36.734772
1	3871	Badghis	1	AF	Afghanistan	BDG	0	35.167134
2	3875	Baghlan	1	AF	Afghanistan	BGL	0	36.178903
3	3884	Balkh	1	AF	Afghanistan	BAL	0	36.755060
4	3872	Bamyan	1	AF	Afghanistan	BAM	0	34.810007
...	...	...	...	...	...	...	...	...
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	0	-17.485103
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	0	-20.624151
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	0	-18.533157
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	0	-21.052337
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	0	-19.055201

5077 rows × 9 columns

In [3]:
```
df.head()
```

Out[3]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	lon
0	3901	Badakhshan	1	AF	Afghanistan	BDS	0	36.734772	70.8
1	3871	Badghis	1	AF	Afghanistan	BDG	0	35.167134	63.7
2	3875	Baghlan	1	AF	Afghanistan	BGL	0	36.178903	68.7
3	3884	Balkh	1	AF	Afghanistan	BAL	0	36.755060	66.8
4	3872	Bamyan	1	AF	Afghanistan	BAM	0	34.810007	67.8

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5077 entries, 0 to 5076
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            5077 non-null   int64  
 1   name          5077 non-null   object 
 2   country_id    5077 non-null   int64  
 3   country_code  5077 non-null   object 
 4   country_name  5077 non-null   object 
 5   state_code    5077 non-null   object 
 6   type          5077 non-null   object 
 7   latitude      5077 non-null   float64
 8   longitude     5077 non-null   float64
dtypes: float64(2), int64(2), object(5)
memory usage: 357.1+ KB
```

In [5]:
```
import seaborn as sns
```

In [6]:
```
df.describe()
```

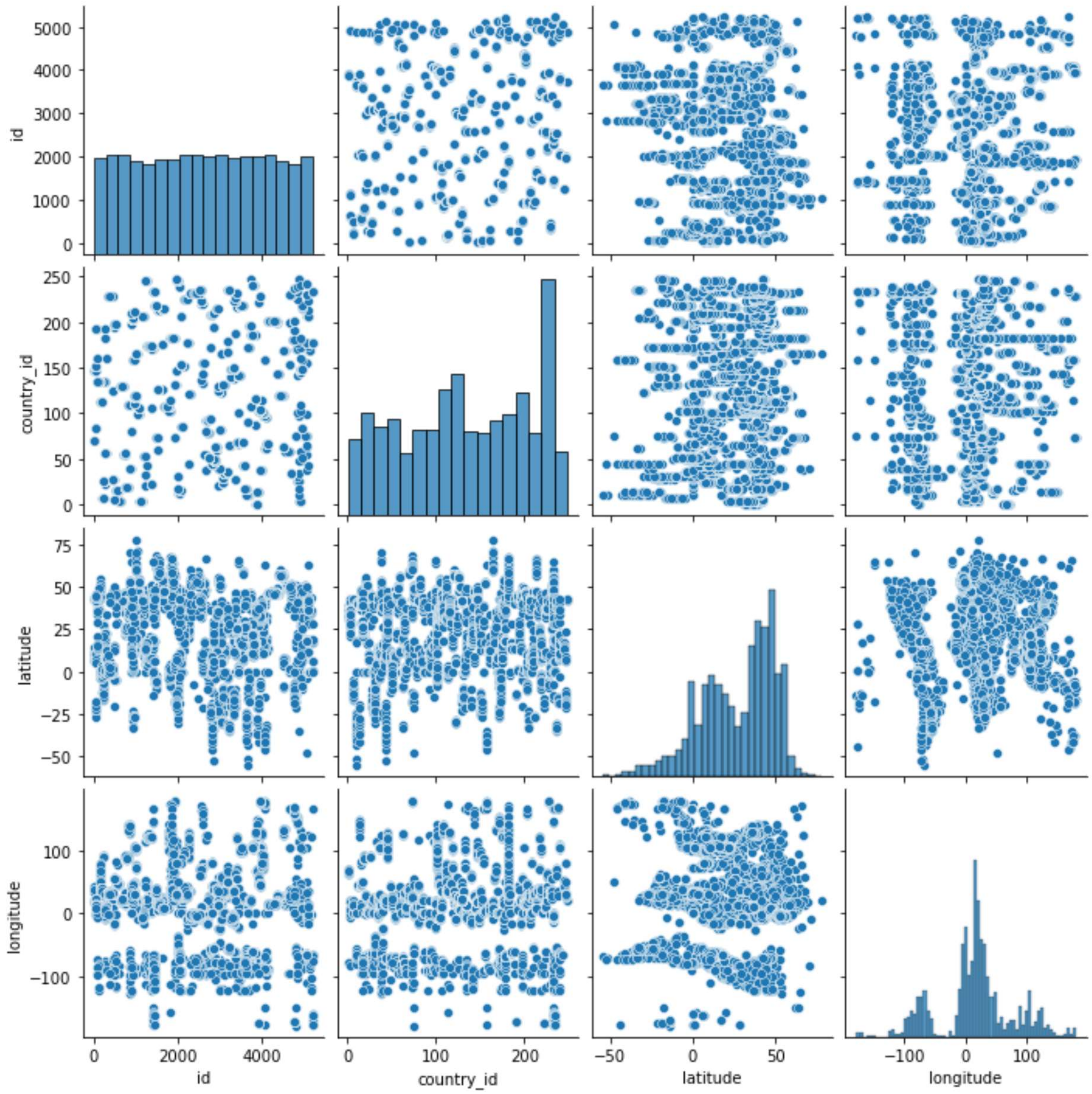
Out[6]:

	id	country_id	latitude	longitude
count	5077.000000	5077.000000	5077.000000	5077.000000
mean	2609.765413	133.467599	27.201632	16.945242
std	1503.376799	72.341160	22.286652	60.883985
min	1.000000	1.000000	-54.805400	-178.116500
25%	1324.000000	74.000000	10.720150	-3.581269
50%	2617.000000	132.000000	33.814390	16.882517
75%	3905.000000	201.000000	45.729683	41.023407
max	5220.000000	248.000000	77.874972	179.852222

Type *Markdown* and LaTeX: α^2

In [7]:
```
sns.pairplot(df)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x28adbcbe640>

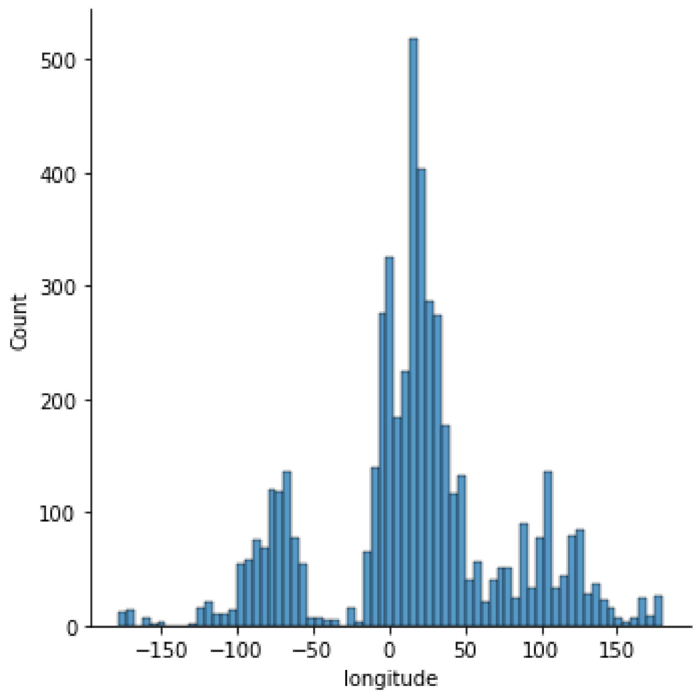


In [8]:
```
df1=df.drop(['name'],axis=1)
df1
df1=df1.drop(df1.index[1537:])
df1.isna().sum()
```

Out[8]:
```
id              0
country_id      0
country_code    0
country_name    0
state_code      0
type            0
latitude        0
longitude       0
dtype: int64
```

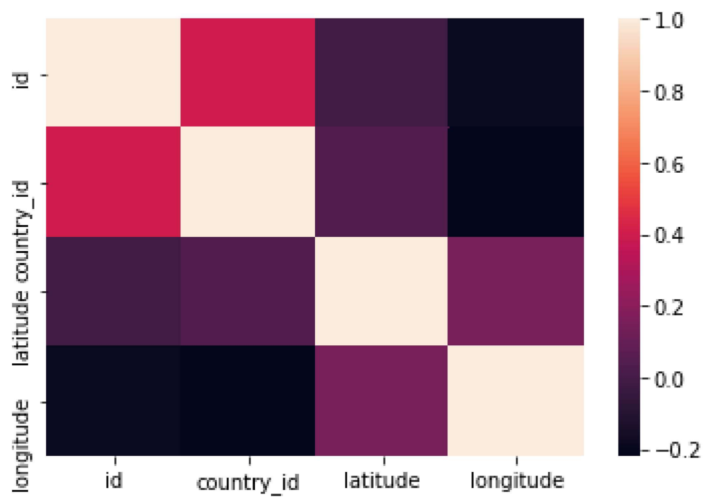
In [9]:
```
sns.displot(df['longitude'])
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x28ad8d218b0>



In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]: <AxesSubplot:>



In [11]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [12]:
```
df1.isna().sum()
```

Out[12]:
```
id              0
country_id      0
country_code    0
country_name    0
state_code      0
type            0
latitude        0
longitude       0
dtype: int64
```

In [16]:
```
y=df1['country_id']
x=df1.drop(['country_code','country_name','state_code','state_code',''],axis=1
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train)
```

```
        id  country_id   latitude  longitude
432    238          26  27.032286  89.887930
1251  4983          75  45.665848  -0.318458
338    819          19  22.944674  90.828191
480   3068          29  37.588446 -94.686378
659    876          39  70.299771 -83.107577
...    ...         ...        ...        ...
1437  2100          85  40.519269  21.268717
215   2065          15  48.108077  15.804956
1024  1531          59  56.302139   9.302777
249    525          16  40.102433  46.036487
951   4550          58  50.414572  16.165635

[1075 rows x 4 columns]
```

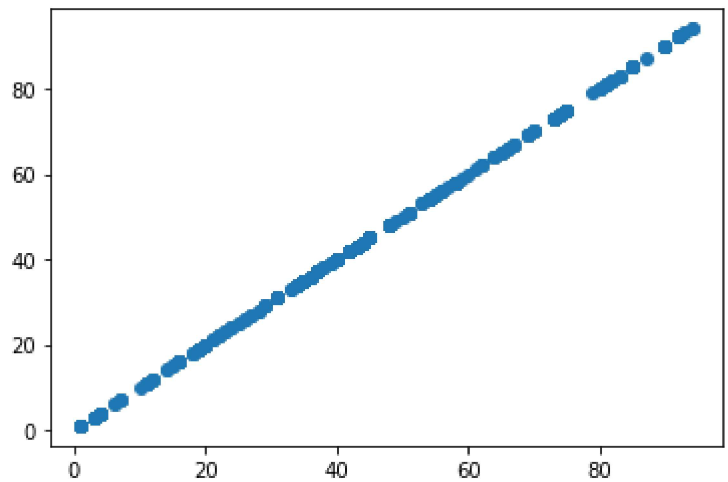
In [40]:
```
print(x)
```

```
        id  country_id   latitude  longitude
0     3901           1  36.734772  70.811995
1     3871           1  35.167134  63.769538
2     3875           1  36.178903  68.745306
3     3884           1  36.755060  66.897537
4     3872           1  34.810007  67.821210
...    ...         ...        ...        ...
1532  2764          94   7.488242 -59.656449
1533  2760          94   6.464214 -60.211075
1534  2767          94   6.546426 -58.098205
1535  2766          94   2.747792 -57.462726
1536  2768          94   6.572013 -58.463000

[1537 rows x 4 columns]
```

In [17]:
```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```
Out[17]:
```
-1.4210854715202004e-14
```

In [18]:
```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```
Out[18]:
```
<matplotlib.collections.PathCollection at 0x28adf492820>
```



In [19]:
```
model.score(x_test,y_test)
```
Out[19]:
```
1.0
```

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```
Out[21]:
```
Ridge(alpha=10)
```

In [22]:
```
rr.score(x_test,y_test)
```
Out[22]:
```
0.9999999997880403
```

In [23]:
```
la =Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[23]:
```
Lasso(alpha=10)
```

In [24]:
```
la.score(x_test,y_test)
```
Out[24]:
```
0.9997660256682711
```

In [25]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[25]:
```
ElasticNet()
```

In [26]:
```
print(en.coef_)
```

```
[ 1.22528840e-05  9.98366001e-01  0.00000000e+00 -0.00000000e+00]
```

In [27]:
```
print(en.intercept_)
```

```
0.042538010650531535
```

In [28]:
```
print(en.predict(x_test))
```

```
[19.02098802 16.02307185  7.03385692 50.9927157  80.92126071 44.99671183
 63.97355671 58.00426412 16.02288806 16.02354971 74.98158334 25.06132282
 52.97079882 55.95441587 20.02486782 53.98664968 16.02337817 68.97352263
 44.99665057 73.93995241  4.04961497 63.973716    6.03870117 91.92532966
 35.02395689 31.01662262 91.92494982 78.94687796 11.0843581  53.98656391
 69.92817034 91.92536641 61.99149141  7.03379565 65.98539652 43.01628842
 18.03753378 79.9445333   3.04498774 37.03080977 35.02360155 72.94694091
 65.98535976 35.02376084 18.03758279  4.09613917 66.97523451 58.00387203
 58.00417835 26.003056   40.01377742 74.9816201  55.9545384  44.99688337
 16.02276553 55.95451389 19.02081648 66.97522226 36.02287427 44.99678535
 10.07165622 74.9816446  41.98940995 74.98181614 19.02101252 41.98943446
 74.98114224  6.03873793  3.04534308 50.99296076 55.95452614 84.92962422
  3.04495099 44.00541574 35.02387112 74.98175488 11.0692748  61.99129536
 47.9995414  41.98944671 38.98945981 69.9282071  58.00381076 35.0240059
 53.98667419 54.96164943 58.00349219 20.02485556 35.02355254 84.92945268
 34.04461961 69.92818259 43.01610462 36.02283751 16.02319438 40.01392445
 58.00409258 20.02492908 38.98957009  1.08833493 58.00388428 74.97875292
 84.92961196 58.00389654 89.94043394 82.96767815 58.0035412  53.98668644
 72.94679388 16.02334141 14.06755855 74.98137504 24.04106092 41.98925067
 58.00457044  4.09618818 63.97360572 58.00431313 23.0082153  74.98119125
 16.02309636 91.92541543 58.96486673 68.97349812 18.03755828  7.03383241
 37.03088329  3.04507351 16.02361098 81.94548029 58.0046072  58.00382302
 44.00526871 63.97370374 11.06916452 74.98180389 19.02130659 21.04444356
 31.01638981 50.99297301  4.04945568 89.94033592  3.0450245  72.94695317
 22.02351127 82.90757776 58.00443566 44.99689562 84.9293424  84.92966098
 35.02351578 54.96166169  3.04518379 12.04774211 79.94450879 93.92276007
 31.05140855 53.9865149  74.97910826  4.04987228 36.022813   11.06914001
 44.9965648  74.98112998 43.01609237 35.02380985 72.94690416 66.97525902
  1.08848196  4.0497375  47.99939437 15.04331798 74.98132603 22.02352352
 33.00352781 58.00367598 72.94687965 31.01657361 41.98932418 64.97594166
 23.0081908  80.92128521 53.98669869 79.94449654 19.02079197 58.00373725
 74.9818284  49.99591807 41.98942221  1.08835943 44.99666282 22.02354803
 19.0213556  21.04446806 72.94678163 24.04099965 74.98094619 58.00459495
 35.02363831 72.94680613  4.09620043 28.00245913 44.99673634 91.92513361
 21.04441905 58.00356571 47.99946789 55.95448938 58.00352895 19.02118406
 58.00468072 44.99692013 64.97596616 58.00392104 19.02107379  4.04989679
  3.04546561 38.01301422 84.92964872 16.02316987 68.97344911 19.02117181
 93.92283359 91.92504784 61.99133212 61.99154042 31.01656135 91.92519487
 16.02353746 43.01615363 12.04775437 58.00471748 74.98195093 72.94673262
 44.00536673 19.0209145  40.01396121 35.02354029 35.02409167 92.92390396
 68.97346136 82.90760226 74.98135053  4.04978651 74.98087267  3.04514703
 74.98155883 91.92521938 26.00301924 91.92499883 73.94000142 53.98679672
 74.98124026 74.98111773 59.98042353 58.00450918 44.00539124 37.03077301
 16.02348845 27.03983479 50.9926912  28.0025449  47.99934536 35.02406716
 53.98650265 12.04771761 44.99680985 37.03085878  1.08861674 40.01383868
 43.01603111 53.98658842 41.9893732  40.01387544 16.02351295 47.99945563
  7.03375889 74.98165686 29.03267037 11.06921353  3.04529407 69.92821935
 19.02123307 19.02075521 43.01601885 44.99682211 44.00534223 35.02392013
 20.02490457 84.92963647  1.08850647  1.0884207  64.97603968 40.01397346
 80.92124845 35.02394463 41.98929968 53.98653941 54.96174746 74.98153433
 48.0244025  34.04466863 48.9970252  50.99281373 72.94685514 35.02416519
 16.02296157  1.08856773 89.94032366  4.04984777 81.94549254 19.0211228
 34.04458286 73.93994015  1.08869026  1.08838394 47.99967618 89.94029916
 44.99660155 86.947713    1.08853097 37.03074851 31.01659811 15.04325672
 54.96163718 23.00825206 65.98542102 34.0447544  80.92122395 74.98121575
 40.01381418 16.02277778 50.99270345 58.00362697 35.02404266 58.00436214
 89.94050746 44.99676084 24.04102416 35.02373633 19.02092675 37.03099356
```

```
 18.03754603 29.03270713 74.98160785 16.02347619 29.03275614 37.03096906
 27.03984704 40.01378967 89.9402869  58.00434989 58.00464396 52.97082332
 35.02350353 22.02343775  4.04967623 36.02276399 20.02480655 73.93991565
 44.99693238 64.97605193 44.00525646 43.01616589 66.97519775  3.04496324
 34.04468088 53.98675996  3.04545335 19.02126983 19.02136786 82.9075655
 84.92946493 16.0230596  53.98652715  4.04968849 14.0675708  27.03985929
 81.94546804 20.02491683  3.04540434 47.99940662 58.00383527 73.94003818
 26.00285995 41.98931193 58.00422736  4.09616368 80.92132197 68.97343686
 58.00355345 38.9894353  12.04784014 74.98149757 29.03265812 58.00377401
 74.98105647 53.98660067 29.03271938 16.02301059 38.01298971 11.06931155
 91.9250846  37.03089554 34.04452159 35.02388337 19.02073071 31.0165491
 31.01647558 12.04776662 19.02102477 37.0309568  56.95857748 84.92959971
 19.02144137 11.06925029  4.04957821 74.98109322 74.98090943 81.94538227
 58.00453368 79.94454555 58.00368824 35.02385886 89.9404707  44.99661381
  3.04539209 73.93997691 19.02122082 74.98088493  4.04954145 69.9282316
 47.99949239 36.02288652 16.02283904 58.00465621 38.01305098 35.02369958
 53.98674771 35.02347902 64.9759049  92.92383044 47.99968844 84.92933015
 16.02299833 31.01637756 58.00433764  4.05001931  1.0886535  31.01646333
 35.02378535 74.98106872 33.00355231 91.9253174  74.98202444 89.94048295
 89.94044619 25.06124931 52.97076206 65.98537201 60.99281908 35.02361381]
```

In [29]:
```
print(en.score(x_test,y_test))
```

```
0.9999976365157585
```

In [30]:
```
from sklearn import metrics
```

In [31]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 1.2992493084282189e-14
```

In [32]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 2.830101461216051e-28
```

In [33]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pre
```

```
Root Mean Squared Error: 1.682290540072092e-14
```

In [43]:
```
import pickle
filename="BHOOMISH"
pickle.dump(model,open(filename,'wb'))
model=pickle.load(open(filename,"rb"))
real=[[10,20,30,40],[12,13,21,43]]
result=model.predict(real)
result
```

Out[43]:
```
array([20., 13.])
```