In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
df = pd.read_csv("states.csv").dropna()
df
```

Out[2]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	long
170	3656	Buenos Aires	11	AR	Argentina	B	province	-37.201729	-59.84
171	3647	Catamarca	11	AR	Argentina	K	province	-28.471588	-65.78
172	3640	Chaco	11	AR	Argentina	H	province	-27.425718	-59.02
173	3651	Chubut	11	AR	Argentina	U	province	-43.293425	-65.11
174	4880	Ciudad Autónoma de Buenos Aires	11	AR	Argentina	C	city	-34.603684	-58.38
...	...	...	...	...	...	...	...	...	
4968	2041	Yaracuy	239	VE	Venezuela	U	state	10.339389	-68.81
4969	2042	Zulia	239	VE	Venezuela	V	state	10.291024	-72.14
5033	5074	Saint Croix	242	VI	Virgin Islands (US)	SC	district	17.729352	-64.73
5034	5073	Saint John	242	VI	Virgin Islands (US)	SJ	district	18.335617	-64.80
5035	5072	Saint Thomas	242	VI	Virgin Islands (US)	ST	district	18.342846	-65.07

1580 rows × 9 columns

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

Out[3]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	longit
170	3656	Buenos Aires	11	AR	Argentina	B	province	-37.201729	-59.841
171	3647	Catamarca	11	AR	Argentina	K	province	-28.471588	-65.787
172	3640	Chaco	11	AR	Argentina	H	province	-27.425718	-59.024
173	3651	Chubut	11	AR	Argentina	U	province	-43.293425	-65.111

	id	name	country_id	country_code	country_name	state_code	type	latitude	longit
174	4880	Ciudad Autónoma de Buenos Aires	11	AR	Argentina	C	city	-34.603684	-58.381

Data cleaning and pre processing

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1580 entries, 170 to 5035
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            1580 non-null   int64  
 1   name          1580 non-null   object 
 2   country_id    1580 non-null   int64  
 3   country_code  1580 non-null   object 
 4   country_name  1580 non-null   object 
 5   state_code    1580 non-null   object 
 6   type          1580 non-null   object 
 7   latitude      1580 non-null   float64
 8   longitude     1580 non-null   float64
dtypes: float64(2), int64(2), object(5)
memory usage: 123.4+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	id	country_id	latitude	longitude
count	1580.000000	1580.000000	1580.000000	1580.000000
mean	2685.916456	134.000633	26.988930	15.009671
std	1611.169440	66.055166	19.635279	66.200355
min	48.000000	11.000000	-54.805400	-178.116500
25%	1339.750000	75.000000	13.752013	-7.622388
50%	2210.500000	139.000000	30.887089	11.665277
75%	4013.250000	178.000000	42.938004	45.682217
max	5220.000000	242.000000	77.874972	166.649935

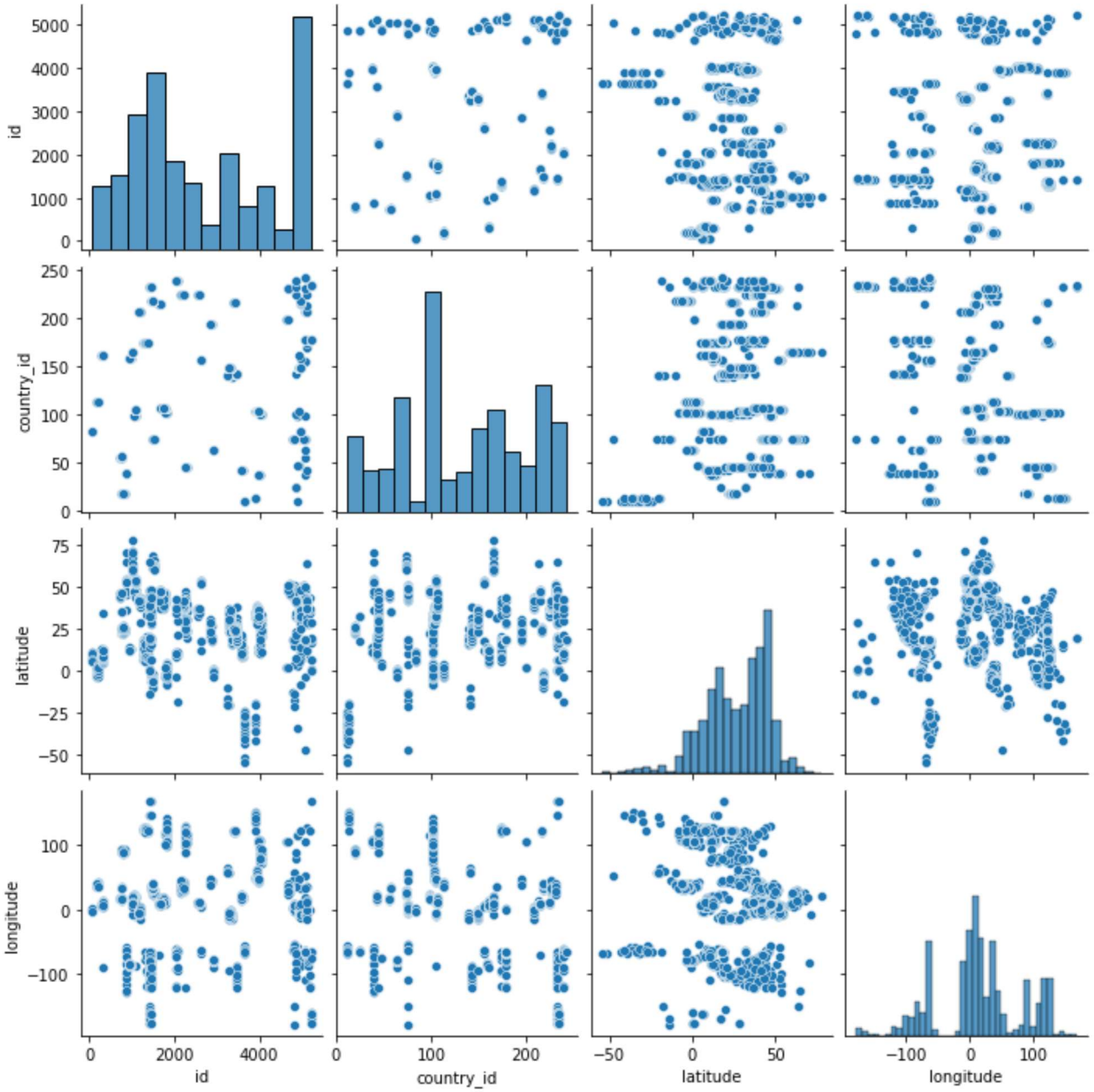
In [6]:
```
df.columns
```

Out[6]:
```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

EDA and VISUALIZATION

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

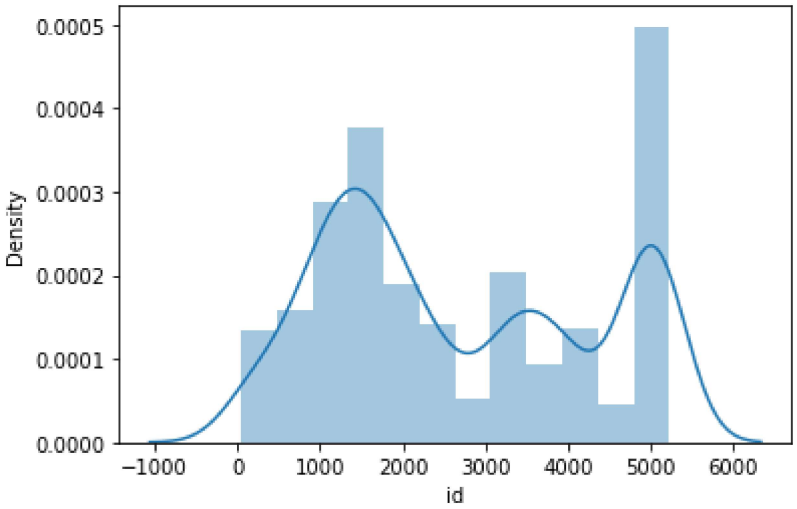
Out[7]: <seaborn.axisgrid.PairGrid at 0x1a0dfa784c0>



In [8]:
```
sns.distplot(df['id'])
```

```
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 adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

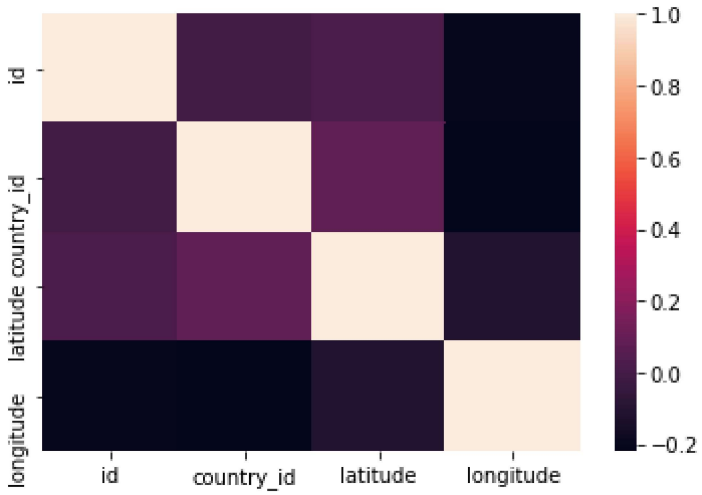
Out[8]: <AxesSubplot:xlabel='id', ylabel='Density'>



In [9]:
```
df1 = df[['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude']]
```

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

Out[10]: <AxesSubplot:>



In [11]:
```
x = df1[[ 'id','country_id','latitude', ]]
y = df1[  'longitude']
```

split the data into training and test data

In [12]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

In [13]:
```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[13]: LinearRegression()

In [14]:
```
lr.intercept_
```

Out[14]: 71.72030586537088

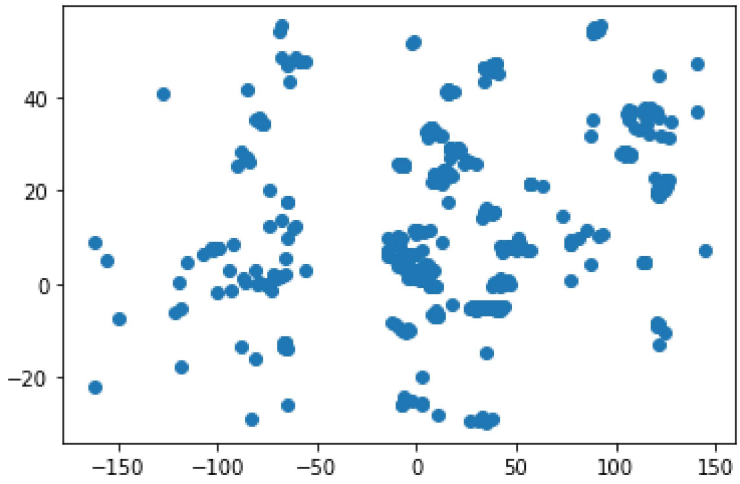
In [15]:
```
coeff = pd.DataFrame(lr.coef_, x.columns, columns =['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
id	-0.008164
country_id	-0.211728
latitude	-0.285410

In [16]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

Out[16]: <matplotlib.collections.PathCollection at 0x1a0e0e7e310>



In [17]:
```
lr.score(x_test,y_test)
```

Out[17]: 0.09089056655038252

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [19]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

Out[19]: 0.09309894107374084

In [20]:
```
rr.score(x_test,y_test)
```

Out[20]: 0.09089058804969075

In [21]:
```
la = Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[21]: Lasso(alpha=10)

In [22]:
```
la.score(x_test,y_test)
```

Out[22]: 0.09090171311470485

In [23]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[23]: ElasticNet()

In [24]:
```
print(en.coef_)
```

```
[-0.00816375 -0.21161806 -0.28376439]
```

In [25]:
```
print(en.intercept_)
```

```
71.66111349722851
```

In [26]:
```
print(en.predict(x_test))
```

```
[  2.78066968 -13.21948028  36.46321718  41.15978801  26.52965804
  -0.18612289   7.45059281   7.69820457   0.39386783  -4.61684436
  32.63781437   8.64297654   6.60212562   4.24060313 -13.48450436
  -8.58106823  36.81994093  55.30634239  22.12197696   9.60294091
  54.40705442 -13.56469124  47.39971155  54.33481076 -26.00427662
   7.74436764   6.44351753 -13.71676975  25.9110113   23.662018
   4.637796    22.82705143  28.21326034  51.51259539  -5.33283945
  -8.83103889  26.50344289  54.77570774  14.54685719  54.52034508
   2.80298492   2.16878783  -5.36635404 -25.54103606   0.35787668
  -4.88111676  -4.91882823  26.32297525   7.56323084   8.65859792
 -14.85672489  47.32397832  23.38332928 -25.19609591 -17.90940696
   6.14250815  21.22129581  35.52288211  46.41635301   8.86205033
  -0.56272691 -13.27768735  -0.10848745  -0.31594314   3.04218064
  27.00935124   1.32673972   1.26872908  22.30001376  -4.60079487
  54.75500787  21.10768738 -13.63530044  21.70270835  -6.7953903
  -7.29305524  32.38932881   4.45244629 -25.71596151  35.03003631
  20.06089024  53.9263728    4.70502564  15.40909707  11.15021547
  -4.87144371  25.54916743  34.92736638  25.54093938  47.42838155
   8.26791806  29.15494426  21.87227759  -0.69082123  21.81436722
  -9.36153451   9.69667814  -6.63948466  22.62969249  17.55965739
  24.3554651   -0.45246768  22.96102456  46.33286193  41.16230048
  36.10379411   1.60823016  22.26954694   1.7183622   54.4965484
 -29.25003069   8.35114393  48.613825    -8.32391381  31.7118988
  -6.00075605   1.61261288  -8.70163573   1.38800198  40.9888288
  -4.9544191   20.31897181  17.4688299   29.11956081 -13.24453026
  31.59675993  19.01729861  21.77243204  35.68098535 -12.72210212
 -24.34692436  -0.26452312  46.37103124 -13.25959224  19.21384818
   2.34373249  21.86180724   2.73654979   4.63948628   2.15145741
```

```
 2.19384829   0.96482001   4.59298341   4.67187928   1.90689627
 1.55932317  54.18272061  12.24266777  25.36124097  26.83544633
-4.86572646   2.78724267  -4.87300043  20.18533885  14.89424651
46.2473332   54.00313488   7.05001026  47.25292124  22.55210557
 4.67963249  22.20823744  -4.30267297  20.11454819  23.42391973
53.92561122  20.44918262   3.46964187 -22.25098826  34.76453015
28.78248667  20.62048831  -6.35601805  54.70126213   3.45790471
18.8379576   28.15919971  27.42153542  -6.73200332  34.36847069
-1.16638631  -0.5953149   21.25503454  43.47141039  41.1530576
-4.9387612    2.11941378  21.0137115   45.7134753   28.36647911
27.81665963  22.18687987  41.55974551  22.10537278  33.2537554
54.08696546   3.64896559  -9.78862623  47.18867553 -19.72709773
21.82997646  36.80903909  47.25243535  41.79420635  -0.16310112
-9.78837385  -5.19056675  -5.53955132  41.01901354   7.52102661
-0.56097004  17.50833614   4.02088329  31.72080303   9.67555576
25.54794411  48.57194906  25.84928098   7.53725178  28.3660859
-27.85098018  7.10331515  21.1417075   -5.14987601  28.8704378
-13.03048916  8.21182679   6.31163147  -8.66155723 -28.48350996
37.00445273  47.24400565  46.21174467  11.4217743   25.9464007
21.56516888  52.11159554 -13.42124585   2.0667634    1.74963176
 4.12650369  23.01283744 -28.7172263   36.03442802   2.62122002
10.15310883   7.17490863  54.58556054   9.84409703  54.84286929
-13.66729878 10.62036522  45.20753494  54.80286673   2.90662268
46.02787427 -15.99030119  -0.50289552  -5.75096666   4.61536928
 5.22894514  28.66323946   3.93454813   2.95550066   1.29351714
28.31946373  24.58497263   9.3160659    0.89953924   6.58350077
20.38564429  34.35940447 -13.3294921    5.98033009  22.24015015
-5.49638538  34.97658257   6.6896295    7.80570132   8.26418679
28.03826162  -1.91949381   0.28354207  46.57194363  45.47137129
43.59373592  -5.08839618   4.03856347  -5.1548879   29.11884829
27.90300634  53.93939663   4.64944733 -12.66806306  54.54229897
11.75158853   9.16741175 -10.32318209  31.50272563  25.43512461
21.08636782   9.83495206  25.27119474  -0.5810521   21.20992565
 7.01187787   8.00509822  37.72242572   7.20941969  21.37570838
41.59733922   2.55385095  -0.41656564  -0.21006161  23.13147286
10.55237231  12.22859958  20.16573678  22.89362771  54.19605026
13.76455768  22.52459034  33.1690973   -5.4650817   -7.08764867
 0.89150675  37.5358143  -29.7039955   20.49490767   8.0524344
21.91826806  22.29172237  -4.84581016  -8.19195043  28.50991124
54.97013435  54.77271607   7.06772111  -4.66528363   5.18906258
-5.4088676    6.08180676  -0.26029615   4.57439132  11.58122911
 8.01618493   1.55417664   7.18717705  -4.71533342  34.31335948
-25.90266889 40.99660489   3.7396178   28.67492453  21.83586581
15.25464665  10.35707553   0.5941295    1.49379648  14.1542731
27.21554305  21.04502417  27.32557713  35.09912361  35.19688708
26.58601022  54.38470858  26.41044386  46.20292694  33.54670933
27.66994376 -13.41027613  37.11742915  46.37829351   8.39520893
-4.73453825  21.19632848  -9.45755189 -12.64455158   5.06596668
-4.58457268   1.38129834 -13.14172883  36.58508665  24.29148874
 2.41737398  -9.86444672   1.29692433  11.3547184   47.36739934
22.69820483   3.127976     7.69989778 -10.31743039   6.79414583
15.29181936  46.31057098   3.96747025  27.67509997  54.841145
25.41107269  28.72027525  55.08745304  53.65463372  44.66341819
19.45006943   3.03391836  -1.44028957  32.67480757  -8.98519767
31.3057446   33.27391809 -25.71958894 -13.51191256  54.36297852
-4.98031357  19.83026044 -29.34715558   5.97168142  24.08196148
 2.36226849  31.98565366  33.00784936   2.70074329  -4.99923298
32.50935439  55.17509876  16.06865384  54.63874985  -5.37598594
19.06681327   4.15673684  19.11929655   8.23087177  22.15972434
 1.23290671   4.14619377  19.16660415  -5.2879037   -5.00191482
29.00897788 -13.3527321    0.72113976   9.70128327   2.18239976
-0.51382342  25.60987695   4.62612687  23.27508505  -9.76256798
-0.48415983   3.20153835  31.67740843   0.40541578   5.3672734
46.55563919  11.79757531  21.8886819    4.14427648  25.9009335
 6.48648088  46.63006956  -5.19626551  22.02666406  33.56969806
```

```
 1.29838001 -28.9323833   -5.45128822  25.50835792  45.95446249
34.26383232   2.90734448  25.62646606  -6.99453228]
```

In [27]:
```
print(en.score(x_test,y_test))
```

```
0.09089540832542065
```

Evaluation Metrics

In [28]:
```
from sklearn import metrics
```

In [29]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 47.73228923464871
```

In [30]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 3835.1091966366057
```

In [31]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 61.9282584660396
```

In [32]:
```
import pickle
```

In [33]:
```
filename='prediction'
pickle.dump(lr,open(filename,'wb'))
```

In [34]:
```
import pandas as pd
import pickle
```

In [35]:
```
filename='prediction'
model = pickle.load(open(filename,'rb'))
```

In [37]:
```
real = [[10,20,30],[11,45,55]]
result = model.predict(real)
```

In [38]:
```
result
```

Out[38]:
```
array([58.84180743, 46.40519511])
```