DATA COLLECTION

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

In [5]:
```
# To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\20_states.csv")
sd
```

Out[5]:

	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
...	...	...	...	...	...	...	...	...
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	NaN	-17.485103
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	NaN	-20.624151
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	NaN	-18.533157
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	NaN	-21.052337
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	NaN	-19.055201

5077 rows × 9 columns

In [6]:
```
# to display top 10 rows
sd.head(10)
```

Out[6]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	long
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70.8
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63.7
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68.7
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66.8
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67.8
5	3892	Daykundi	1	AF	Afghanistan	DAY	NaN	33.669495	66.0
6	3899	Farah	1	AF	Afghanistan	FRA	NaN	32.495328	62.2
7	3889	Faryab	1	AF	Afghanistan	FYB	NaN	36.079561	64.9
8	3870	Ghazni	1	AF	Afghanistan	GHA	NaN	33.545059	68.4
9	3888	Ghōr	1	AF	Afghanistan	GHO	NaN	34.099578	64.9

DATA CLEANING AND PRE_PROCESSING

In [7]:
```
sd.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  5063 non-null   object 
 4   country_name  5077 non-null   object 
 5   state_code    5072 non-null   object 
 6   type          1597 non-null   object 
 7   latitude      5008 non-null   float64
 8   longitude     5008 non-null   float64
dtypes: float64(2), int64(2), object(5)
memory usage: 357.1+ KB
```

In [8]:
```
# to display summary of statistics
sd.describe()
```

Out[8]:

	id	country_id	latitude	longitude
count	5077.000000	5077.000000	5008.000000	5008.000000
mean	2609.765413	133.467599	27.576415	17.178713
std	1503.376799	72.341160	22.208161	61.269334
min	1.000000	1.000000	-54.805400	-178.116500
25%	1324.000000	74.000000	11.399747	-3.943859
50%	2617.000000	132.000000	34.226432	17.501792
75%	3905.000000	201.000000	45.802822	41.919647
max	5220.000000	248.000000	77.874972	179.852222

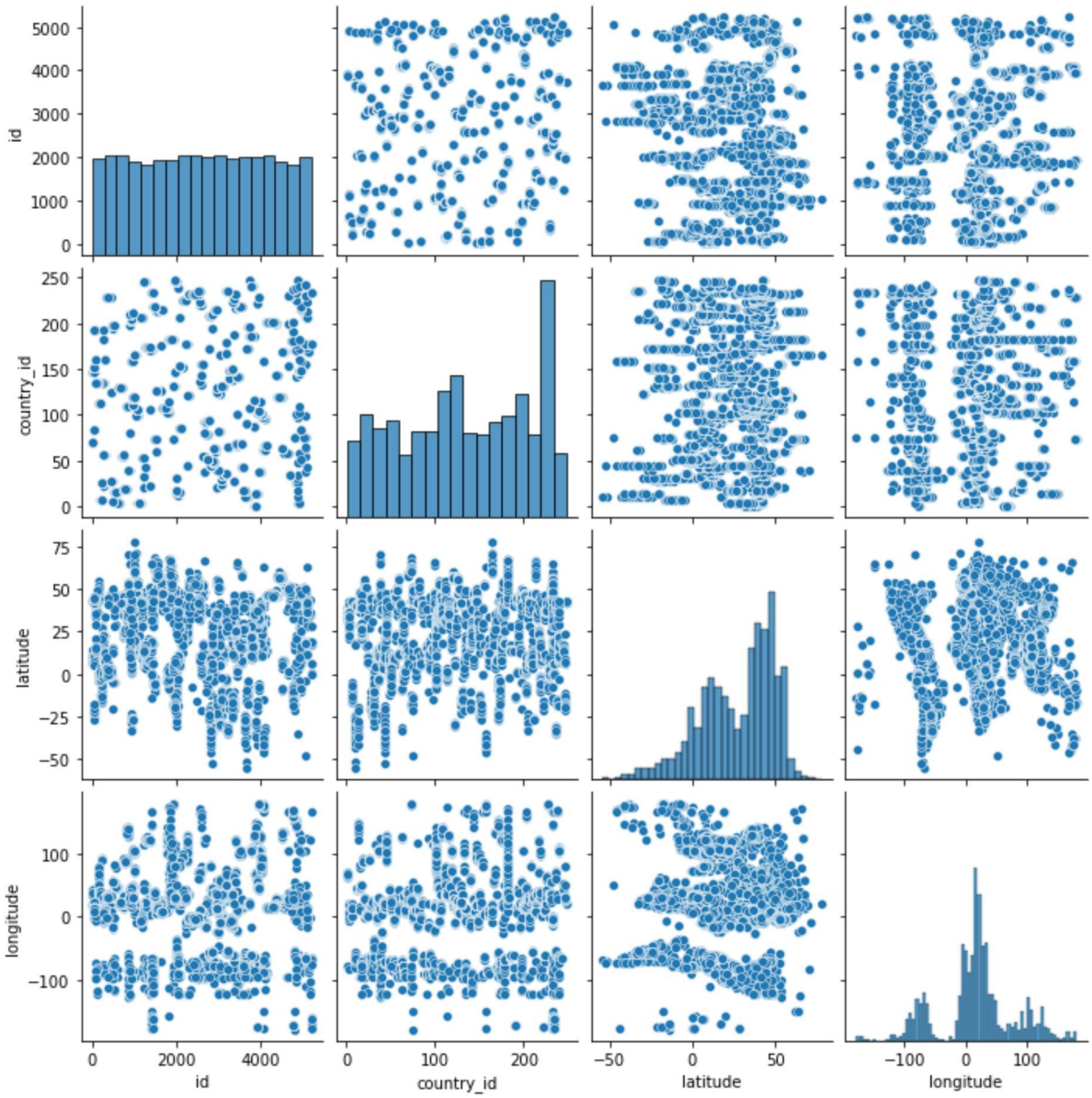
In [9]:
```
#to display colums heading
sd.columns
```

Out[9]:
```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

EDA and visualization

In [10]:
```
sns.pairplot(sd)
```

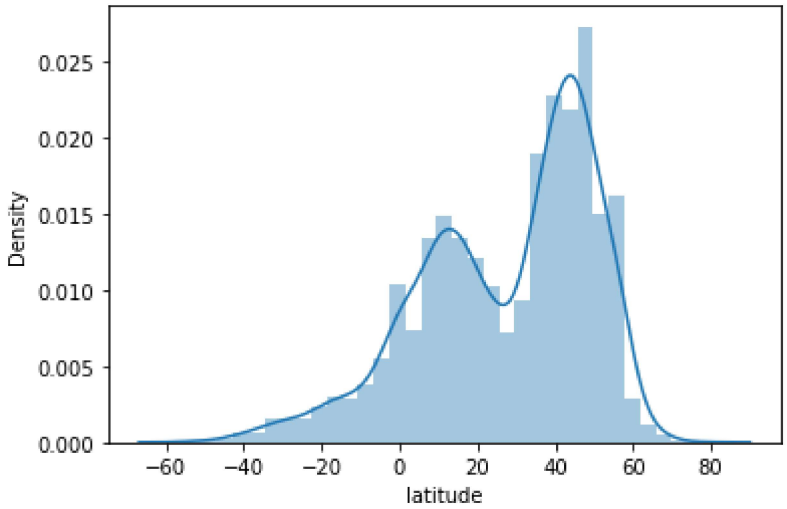
Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x1bcce9fd700>
```



In [11]:
```
sns.distplot(sd['latitude'])
```

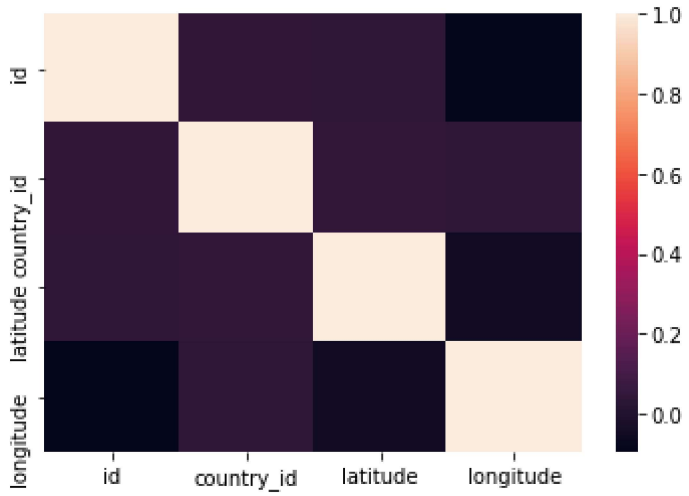
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[11]:
```
<AxesSubplot:xlabel='latitude', ylabel='Density'>
```



In [12]:
```
sns.heatmap(sd.corr())
```

Out[12]:
```
<AxesSubplot:>
```



In [18]:
```
ssd=sd.head(20)
ssd
```

Out[18]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	lo
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67
5	3892	Daykundi	1	AF	Afghanistan	DAY	NaN	33.669495	66
6	3899	Farah	1	AF	Afghanistan	FRA	NaN	32.495328	62
7	3889	Faryab	1	AF	Afghanistan	FYB	NaN	36.079561	64
8	3870	Ghazni	1	AF	Afghanistan	GHA	NaN	33.545059	68
9	3888	Ghōr	1	AF	Afghanistan	GHO	NaN	34.099578	64
10	3873	Helmand	1	AF	Afghanistan	HEL	NaN	39.298936	-76
11	3887	Herat	1	AF	Afghanistan	HER	NaN	34.352865	62
12	3886	Jowzjan	1	AF	Afghanistan	JOW	NaN	36.896969	65
13	3902	Kabul	1	AF	Afghanistan	KAB	NaN	34.555349	69
14	3890	Kandahar	1	AF	Afghanistan	KAN	NaN	31.628871	65
15	3879	Kapisa	1	AF	Afghanistan	KAP	NaN	34.981057	69
16	3878	Khost	1	AF	Afghanistan	KHO	NaN	33.333847	69
17	3876	Kunar	1	AF	Afghanistan	KNR	NaN	34.846589	71
18	3900	Kunduz Province	1	AF	Afghanistan	KDZ	NaN	36.728551	68
19	3891	Laghman	1	AF	Afghanistan	LAG	NaN	34.689769	70

In [24]:
```
sd1=ssd[['id','country_id','latitude', 'longitude']]
```

TO TRAIN THE MODEL _MODEL BUILDING

we are goint train Liner Regression model; we need to split out the data into two varibles x and y where x is independent on x (output) and y is dependent on x(output) adress coloumn as it is not required our model

In [25]:
```
x= sd1[['id','country_id','latitude']]
y=sd1['longitude']
```

In [26]:
```
# To split my dataset  into training data and test data
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 [27]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[27]:
```
LinearRegression()
```

In [28]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[28]:
```
LinearRegression()
```

In [29]:
```
print(lr.intercept_)
```

```
-3534.231972993141
```

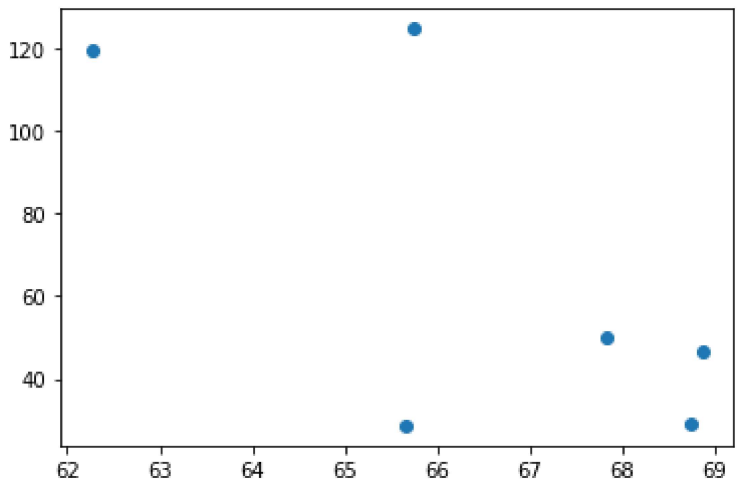
In [30]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[30]:

	Co-efficient
id	1.082073
country_id	0.000000
latitude	-17.401896

In [31]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[31]:
```
<matplotlib.collections.PathCollection at 0x1bcd1824eb0>
```



In [32]:
```
print(lr.score(x_test,y_test))
```

```
-330.0922073527318
```

In [33]:
```
lr.score(x_train,y_train)
```

Out[33]:
```
0.6284318900939427
```

In [34]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [35]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

Out[35]:
```
Ridge(alpha=10)
```

In [36]:
```
dr.score(x_test,y_test)
```

Out[36]:
```
-220.44711915756713
```

In [37]:
```
dr.score(x_train,y_train)
```

Out[37]:
```
0.6008532637538653
```

In [38]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[38]:
```
Lasso(alpha=10)
```

In [39]:
```
la.score(x_test,y_test)
```

Out[39]:
```
-209.94721689900862
```

In [40]:
```
la.score(x_train,y_train)
```

Out[40]:
```
0.5980504297804835
```

ElasticNet

In [41]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[41]:
```
ElasticNet()
```

In [42]:
```
print(en.coef_)
```

```
[  1.09438352   0.         -14.23055535]
```

In [43]:
```
print(en.intercept_)
```

```
-3693.523789669675
```

In [44]:
```
prediction=en.predict(x_test)
```

In [45]:
```
print(en.score(x_test,y_test))
```

```
-240.50617652557546
```

Evaluation metric

In [46]:
```
from sklearn import metrics
```

In [47]:
```
print("mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
mean Absolute Error: 33.44378689572778
```

In [48]:
```
print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean squared Error: 1272.938552920116
```

In [49]:
```
print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root mean Absolytre Error: 35.678264432566166
```

In [50]:
```
import pickle
```

In [51]:
```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [ ]: