DATA COLLECTION

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

In [2]:
```
# To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\21_cities.csv")
sd
```

Out[2]:

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

150454 rows × 11 columns

In [3]:
```
# to display top 10 rows
sd.head(10)
```

Out[3]:

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

DATA CLEANING AND PRE_PROCESSING

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150454 entries, 0 to 150453
Data columns (total 11 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   id            150454 non-null  int64  
 1   name          150454 non-null  object 
 2   state_id      150454 non-null  int64  
 3   state_code    150129 non-null  object 
 4   state_name    150454 non-null  object 
 5   country_id    150454 non-null  int64  
 6   country_code  150406 non-null  object 
 7   country_name  150454 non-null  object 
 8   latitude      150454 non-null  float64
 9   longitude     150454 non-null  float64
 10  wikiDataId    147198 non-null  object 
dtypes: float64(2), int64(3), object(6)
memory usage: 12.6+ MB
```

In [5]:
```
# to display summary of statistics
sd.describe()
```

Out[5]:

	id	state_id	country_id	latitude	longitude
count	150454.000000	150454.000000	150454.000000	150454.000000	150454.000000
mean	76407.091689	2678.377677	140.658460	31.556175	2.369557
std	44357.755335	1363.513591	70.666123	22.813220	68.012770
min	1.000000	1.000000	1.000000	-75.000000	-179.121980
25%	38160.250000	1451.000000	82.000000	19.000000	-58.468150
50%	75975.500000	2174.000000	142.000000	40.684720	8.669980
75%	115204.750000	3905.000000	207.000000	47.239220	27.750000
max	153528.000000	5116.000000	247.000000	73.508190	179.466000

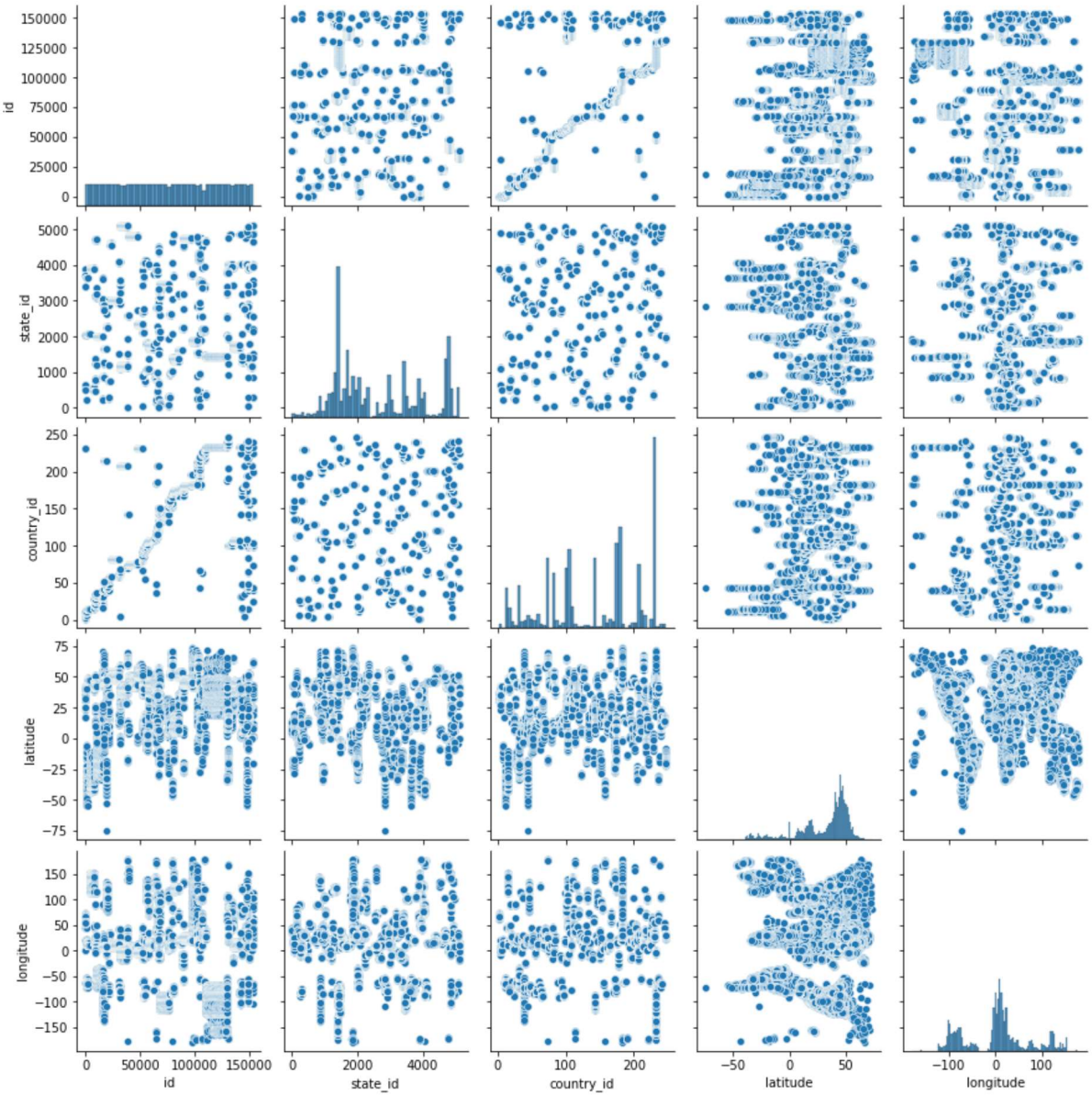
In [6]:
```
#to display colums heading
sd.columns
```

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

EDA and visualization

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

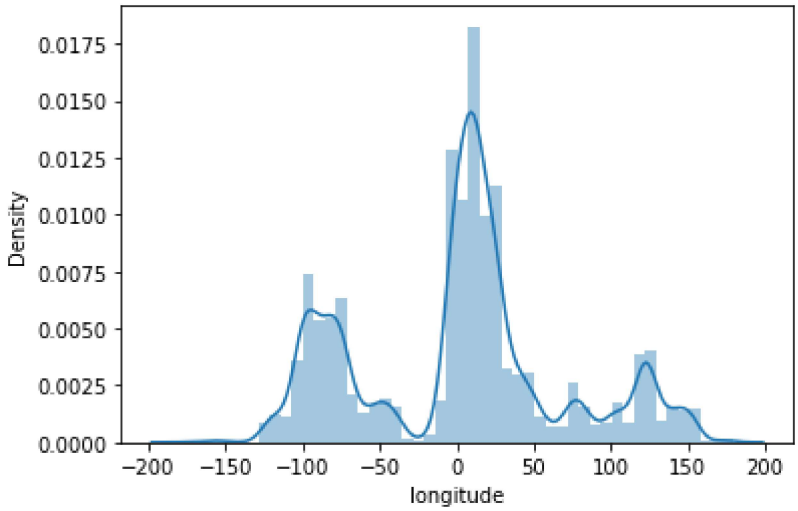
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x1ff41d02700>
```



In [8]:
```
sns.distplot(sd['longitude'])
```

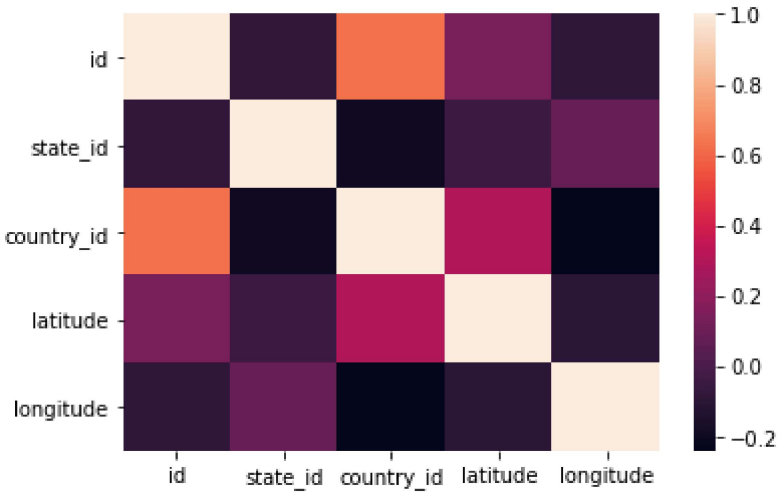
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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



In [9]:
```
sns.heatmap(sd.corr())
```

Out[9]:
```
<AxesSubplot:>
```



In [10]:
```
sd1=sd[['id','state_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 [12]:
```
x= sd1[['id','state_id','country_id', 'latitude']]
y=sd1['longitude']
```

In [13]:
```
# 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 [14]:
```
from sklearn.linear_model import LinearRegression

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

Out[14]:
```
LinearRegression()
```

In [15]:
```
from sklearn.linear_model import LinearRegression

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

Out[15]:
```
LinearRegression()
```

In [16]:
```
print(lr.intercept_)
```

```
27.202502382736327
```

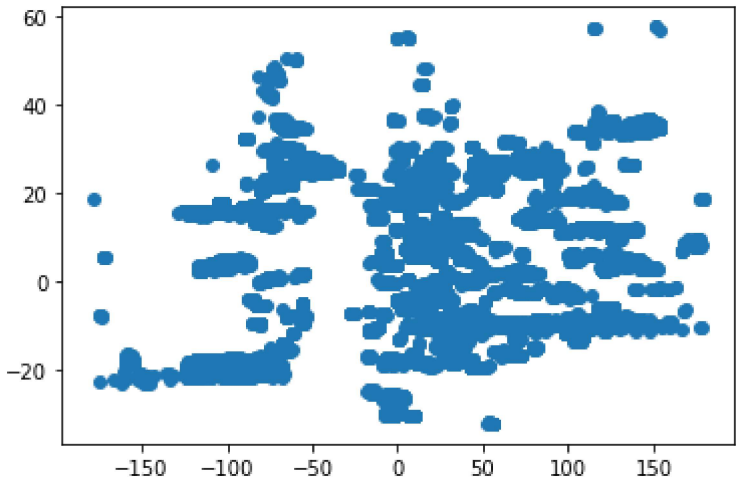
In [17]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
id	0.000153
state_id	0.002047
country_id	-0.275661
latitude	-0.099632

In [18]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[18]: <matplotlib.collections.PathCollection at 0x1ff45517e20>



In [19]:
```
print(lr.score(x_test,y_test))
```

```
0.0653857823547388
```

In [20]:
```
lr.score(x_train,y_train)
```

Out[20]: 0.06759200477057536

In [21]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [22]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

Out[22]: Ridge(alpha=10)

In [23]:
```
dr.score(x_test,y_test)
```

Out[23]: 0.06538578240795034

In [24]:
```
dr.score(x_train,y_train)
```

Out[24]: 0.06759200477057536

In [25]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[25]: Lasso(alpha=10)

In [26]:
```
la.score(x_test,y_test)
```

Out[26]: 0.06539900560124579

In [27]:
```
la.score(x_train,y_train)
```

Out[27]:
```
0.06754853515040748
```

ElasticNet

In [28]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

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

In [29]:
```
print(en.coef_)
```

```
[ 1.52659430e-04  2.04779315e-03 -2.75557254e-01 -9.86195414e-02]
```

In [30]:
```
print(en.intercept_)
```

```
27.166826326990268
```

In [31]:
```
prediction=en.predict(x_test)
```

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

```
0.06538875937056554
```

Evaluation metric

In [33]:
```
from sklearn import metrics
```

In [34]:
```
print("mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
mean Absolute Error: 51.705709014760394
```

In [35]:
```
print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean squared Error: 4334.598735648923
```

In [36]:
```
print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root mean Absolytre Error: 65.83766957942028
```

Model Saving

In [37]:
```
import pickle
```

In [38]:
```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [ ]:

In [ ]:

In [ ]: