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

In [2]:
```
df=pd.read_csv("20_states.csv")
df
```

Out[2]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	longit
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70.811
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63.769
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68.745
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66.897
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67.821
...	...	...	...	...	...	...	...	...	
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	NaN	-17.485103	29.788
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	NaN	-20.624151	31.262
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	NaN	-18.533157	27.549
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	NaN	-21.052337	29.045
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	NaN	-19.055201	29.603

5077 rows × 9 columns

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

Out[3]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	longitude
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70.811995
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63.769538
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68.745306
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66.897537
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67.821210

Data Cleaning and Data Preprocessing

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

Out[5]:

	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

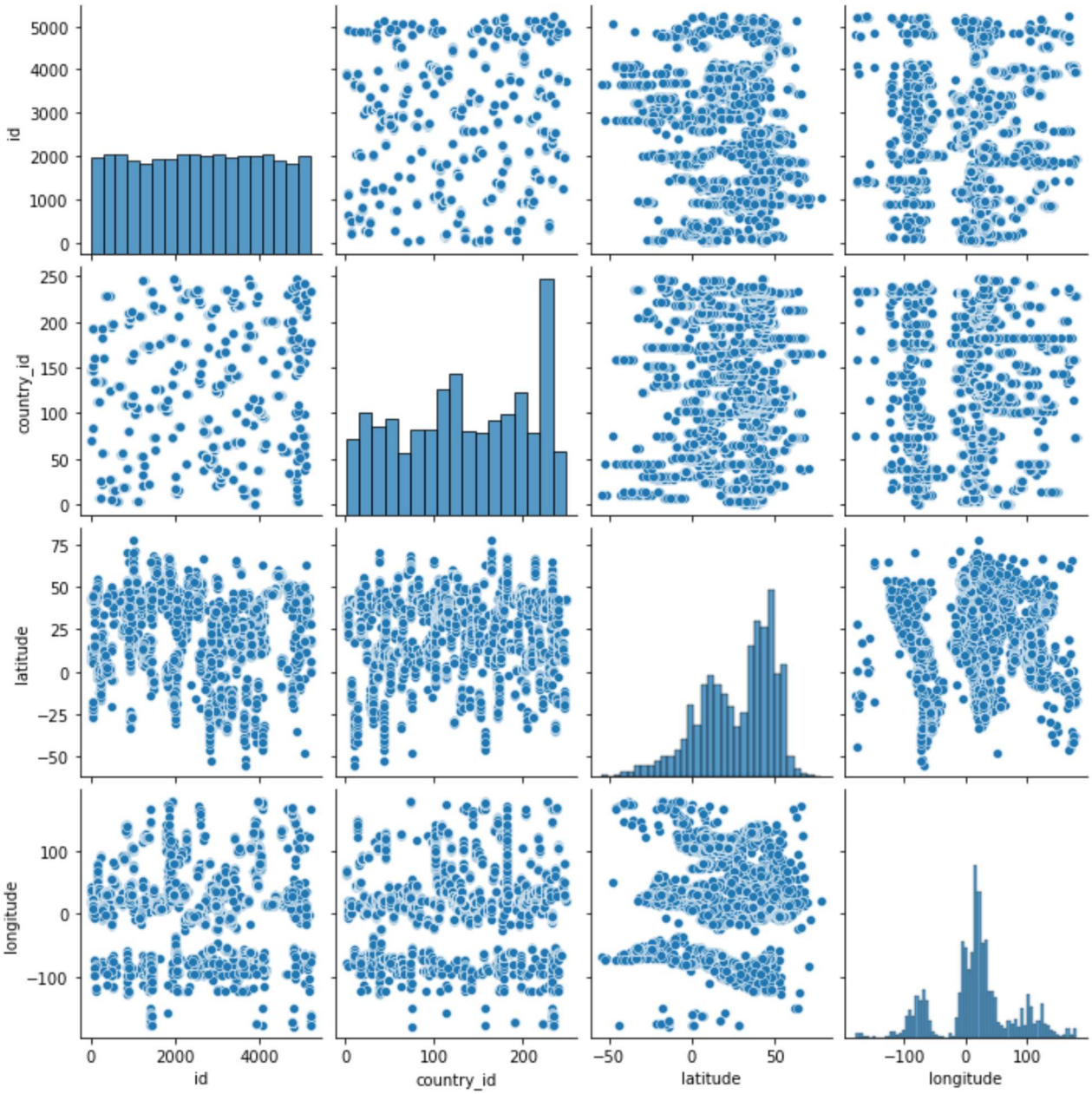
In [6]:
```
df.columns
```

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)
```

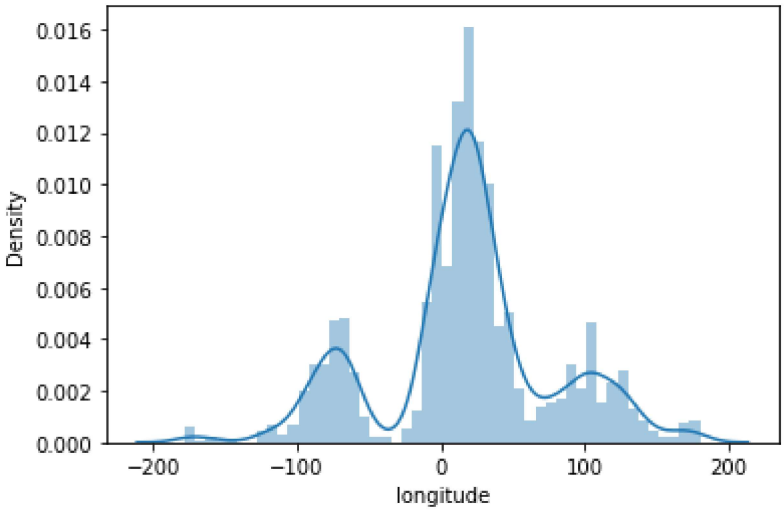
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x1f7c01879a0>
```



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

```
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='longitude', ylabel='Density'>



In [9]:
```
df1=df[['id','country_id', 'latitude', 'longitude']]
df1=df1.dropna()
df1
```

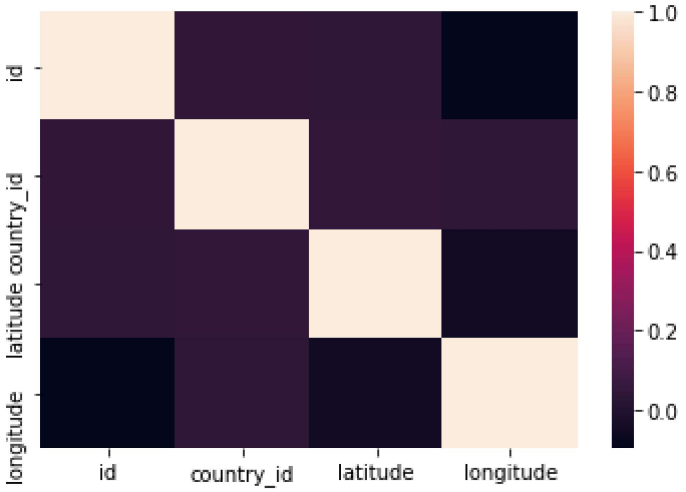
Out[9]:

	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
...	...	...	...	...
5072	1953	247	-17.485103	29.788925
5073	1960	247	-20.624151	31.262637
5074	1954	247	-18.533157	27.549585
5075	1952	247	-21.052337	29.045993
5076	1957	247	-19.055201	29.603549

5008 rows × 4 columns

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

Out[10]: <AxesSubplot:>



To Train the Model -Model Building

We are going to train Linear Regression model;We need to spilt out data into two variables x and y where x is independent variable (input) and y is dependent variable on x(output) we could ignore address column as it is not requiired for our model

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

In [12]:
```
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 [13]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
print(lr.intercept_)
```

```
24.294862108803947
```

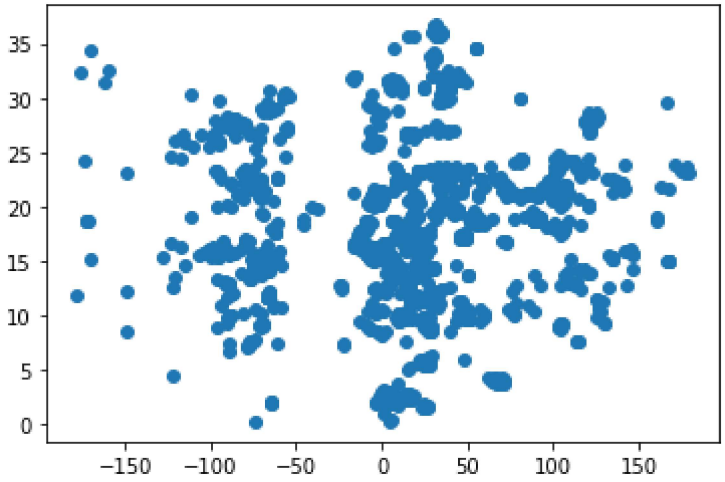
In [15]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
id	-0.003939
country_id	0.058664
latitude	-0.144113

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

Out[16]: <matplotlib.collections.PathCollection at 0x1f7c1ef5340>



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

Out[17]: -0.001399400776957993

In [18]:
```
lr.score(x_train,y_train)
```

Out[18]: 0.016018871870634444

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]: Ridge(alpha=10)

In [21]:
```
rr.score(x_test,y_test)
```

Out[21]: -0.001399372815299671

In [22]:
```
rr.score(x_train,y_train)
```

Out[22]: 0.016018871870542184

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]: -0.0005192053584071044

In [25]:
```
la.score(x_train,y_train)
```

Out[25]: 0.015959236313839553

In [26]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[26]: ElasticNet()

In [27]:
```
en.coef_
```

Out[27]: array([-0.00393902, 0.058545 , -0.14294578])

In [28]:
```
en.intercept_
```

Out[28]: 24.279340467825662

In [29]:
```
prediction=en.predict(x_test)
```

In [30]:
```
en.score(x_test,y_test)
```

Out[30]: -0.0013476874524447346

Evaluation Metrics

In [31]:
```
from sklearn import metrics
```

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

```
Mean Absolute Error: 42.86707357112891
```

In [33]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 3646.57880190704
```

In [34]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 60.38690919319385
```

Model Saving

In [35]:
```
import pickle
```

In [36]:
```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [37]:
```
import pandas as pd
import pickle
```

In [38]:
```
filename="prediction"
model=pickle.load(open(filename,'rb'))
```

In [39]:
```
real=[[10,20,30],[11,45,10]]
result=model.predict(real)
```

In [40]:
```
result
```

Out[40]:
```
array([21.10536107, 25.4502703 ])
```