kaviyadevi 20106064

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

In [7]:
```
#to import dataset
data1=pd.read_csv(r"C:\Users\user\Downloads\21_cities - 21_cities.csv")
data1
```

Out[7]:

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

150454 rows × 11 columns

In [9]:
```
#to display top 5 rows
data=data1.head(100)
data
```

Out[9]:

	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
...	...	...	...	...	...	...	...	...	
95	180	Bashkia Poliçan	629	BR	Berat District	3	AL	Albania	4
96	186	Bashkia Skrapar	629	BR	Berat District	3	AL	Albania	4
97	191	Berat	629	BR	Berat District	3	AL	Albania	4
98	280	Çorovodë	629	BR	Berat District	3	AL	Albania	4
99	219	Kuçovë	629	BR	Berat District	3	AL	Albania	4

100 rows × 11 columns

DATA CLEANING AND PREPROCESSING

In [10]:
```
#
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            100 non-null    int64  
 1   name          100 non-null    object 
 2   state_id      100 non-null    int64  
 3   state_code    100 non-null    object 
 4   state_name    100 non-null    object 
 5   country_id    100 non-null    int64  
 6   country_code  100 non-null    object 
 7   country_name  100 non-null    object 
 8   latitude      100 non-null    float64
 9   longitude     100 non-null    float64
 10  wikiDataId    100 non-null    object 
dtypes: float64(2), int64(3), object(6)
memory usage: 8.7+ KB
```

In [11]:
```
#to display summary of statistics(here to know min max value)
data.describe()
```

Out[11]:

	id	state_id	country_id	latitude	longitude
count	100.000000	100.000000	100.00000	100.000000	100.000000
mean	103.190000	3626.360000	1.16000	35.269691	63.111251
std	38.269564	888.371175	0.54532	2.385821	13.109976
min	50.000000	629.000000	1.00000	30.150000	19.840740
25%	74.750000	3876.750000	1.00000	34.079350	63.255742
50%	99.500000	3886.000000	1.00000	35.028155	66.726500
75%	124.250000	3895.000000	1.00000	36.714017	69.122600
max	280.000000	3902.000000	3.00000	40.824920	73.349280

In [12]:
```
#to display the column heading
data.columns
```

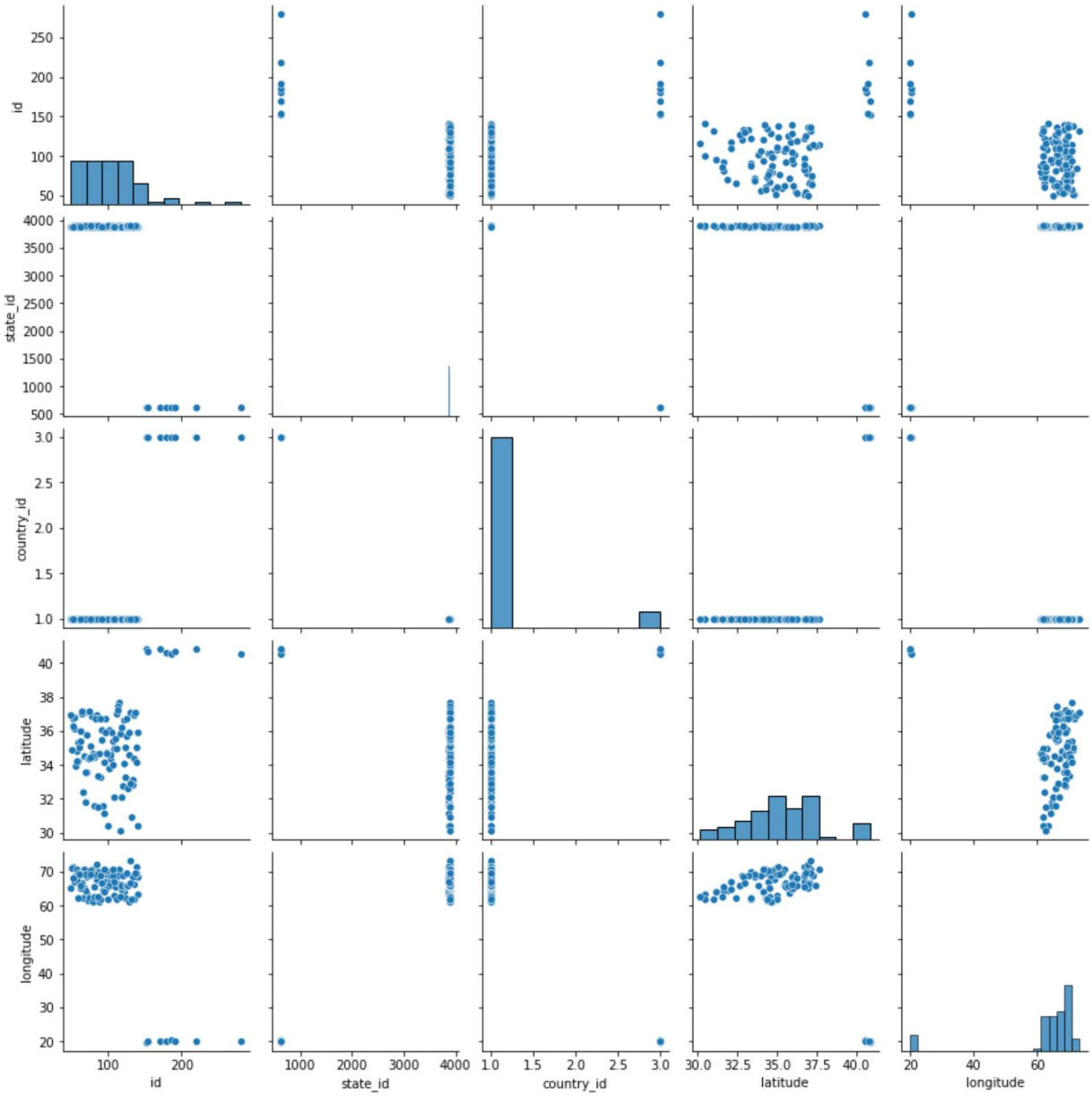
Out[12]:
```
Index(['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataId'],
      dtype='object')
```

In [13]:
```
#here there is no missing values (identified through info() 5000 data are describ
```

EDA and DATA VISUALIZATION

In [14]:
```
sns.pairplot(data)
```

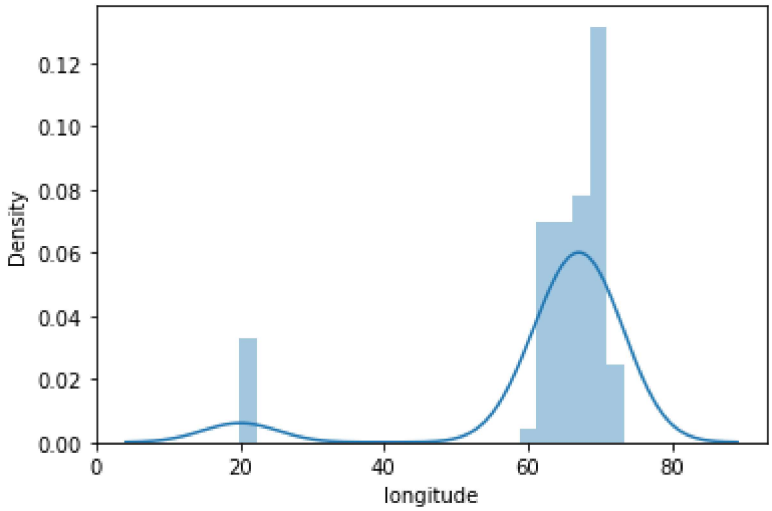
Out[14]: <seaborn.axisgrid.PairGrid at 0x2b549449fd0>



In [16]:
```
sns.distplot(data['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 adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

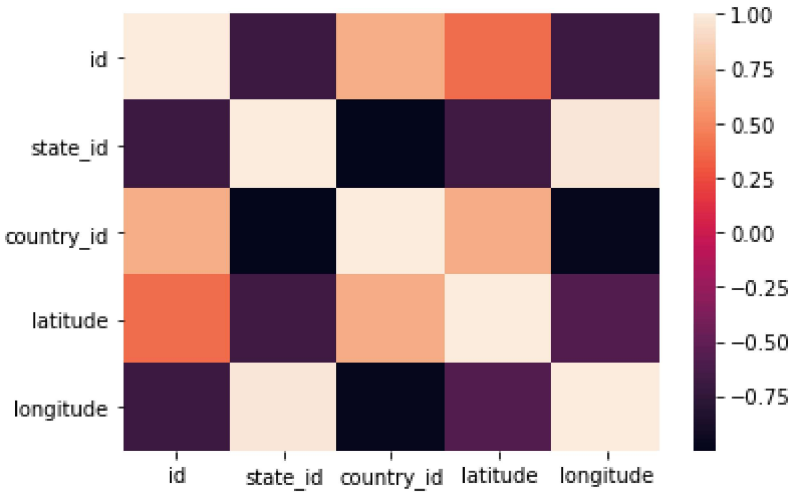
Out[16]: <AxesSubplot:xlabel='longitude', ylabel='Density'>



In [18]:
```
df=data[['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataId']]
```

In [19]:
```
sns.heatmap(df.corr())
```

Out[19]: <AxesSubplot:>



TO TRAIN MODEL

In [24]:
```
x=df[['id', 'state_id','country_id', 'latitude']]
y=df[ 'longitude']
```

In [25]:
```
#to split my dataset into trainning and test

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 [26]:
```
from sklearn.linear_model import LinearRegression

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

Out[26]: LinearRegression()

In [27]:
```
#to find intercept
print(lr.intercept_)
```

```
-118.08163726652421
```

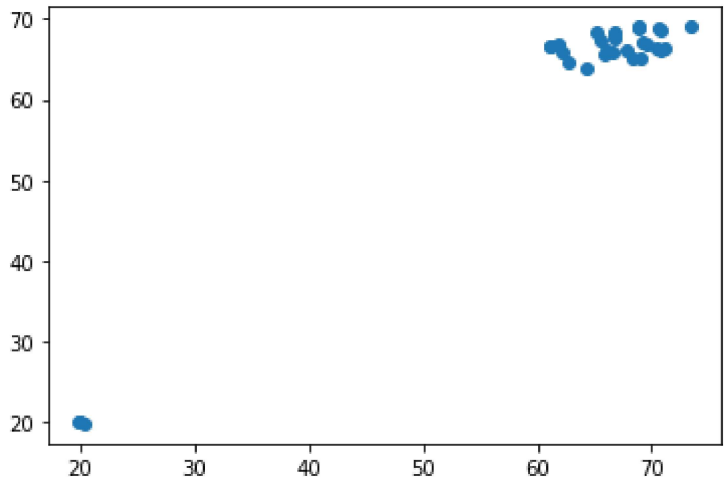
In [28]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[28]:

	Co-efficient
id	0.002659
state_id	0.033507
country_id	28.899066
latitude	0.732341

In [29]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[29]:
```
<matplotlib.collections.PathCollection at 0x2b54cd6dd30>
```



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

```
0.9599527718020827
```

RIDGE AND LASSO REGRESSION

In [31]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [32]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[32]:
```
Ridge(alpha=10)
```

In [33]:
```
rr.score(x_test,y_test)
```

Out[33]:
```
0.9604153086778835
```

In [34]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[34]:
```
Lasso(alpha=10)
```

In [35]:
```
la.score(x_test,y_test)
```

Out[35]:
```
0.9540482845272277
```

In [36]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[36]:
```
ElasticNet()
```

In [37]:
```
print(en.coef_)
```

```
[0.         0.015276   0.         0.50928868]
```

In [38]:
```
print(en.predict(x_test))
```

```
[68.20110764 64.93910898 20.08504898 65.25754173 65.89153263 19.95031157
 68.05127226 68.14922375 67.88081776 65.59016791 66.0234085  65.84871363
 66.65504228 66.56768443 66.36611705 68.1795468  66.33423956 66.71794076
 66.95541255 66.00036452 68.21670206 66.85726177 66.87652395 20.02439779
 66.5192296  67.91285911 67.85444679 67.36457729 67.13587855 66.71143205]
```

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

```
0.9600554139069357
```

Evaluation metrics

In [40]:
```
from sklearn import metrics
```

In [41]:
```
print("Mean Absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error 2.2662145189921987
```

In [42]:
```
print("Mean Squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared error 8.376022978373195
```

In [43]:
```
print("Root Mean Absolute error",np.sqrt(metrics.mean_squared_error(y_test,predic
```

```
Root Mean Absolute error 2.894135964044052
```

MODEL SAVING ¶

In [44]:
```
import pickle
```

In [45]:
```
filename='predict5'
pickle.dump(lr,open(filename,'wb'))
```