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("cities.csv")
df
```

Out[2]:

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

150454 rows × 11 columns

In [3]:

```
df.head()
```

Out[3]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_name	latitude
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanistan	36.68333
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanistan	37.11664
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanistan	36.86477
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanistan	36.95127
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanistan	37.66079

Data cleaning and pre processing

In [4]:

```
df.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]:

```
df.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]:

```
df.columns
```

Out[6]:

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

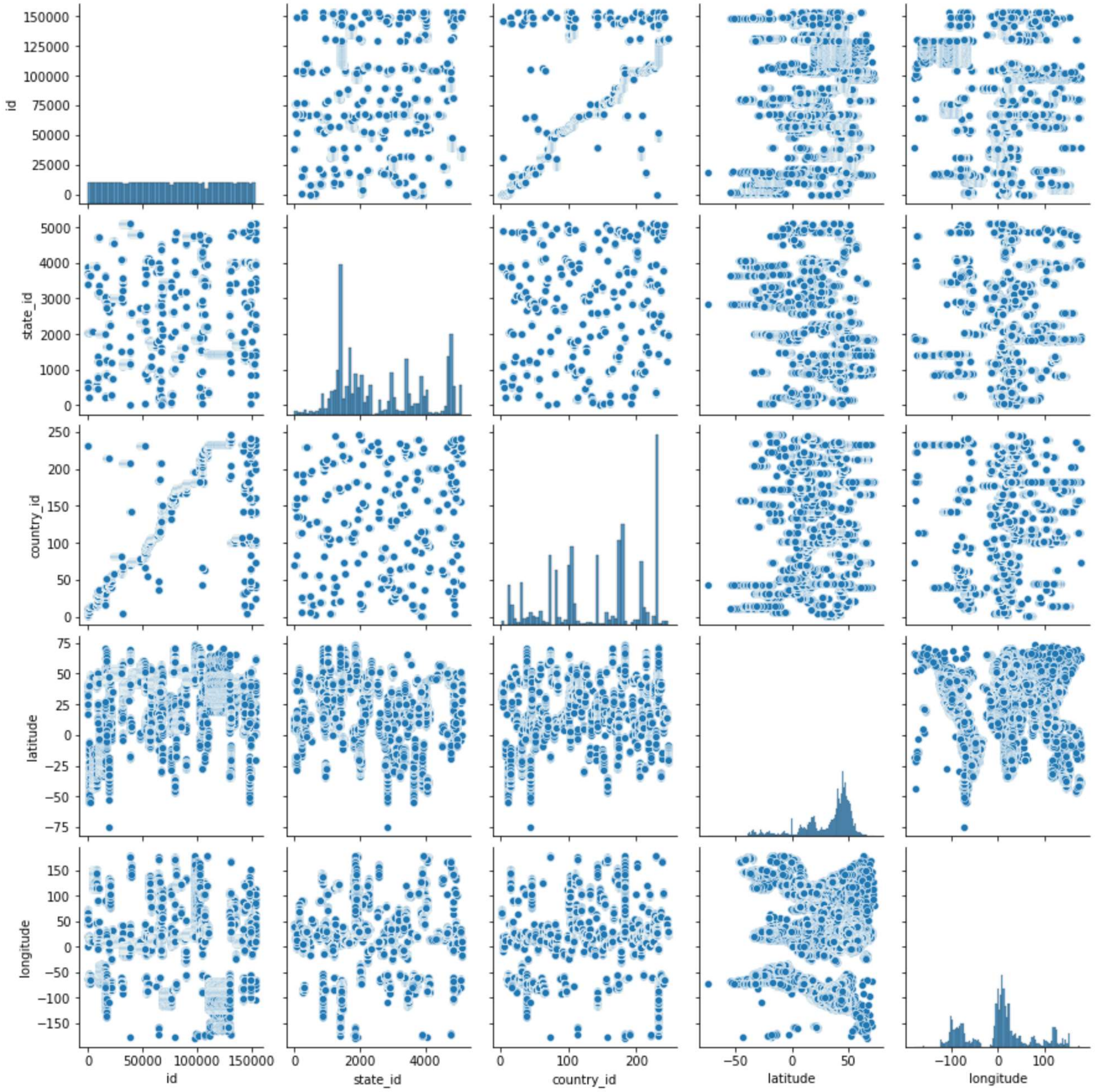
EDA and VISUALIZATION

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x239d5c4c040>
```



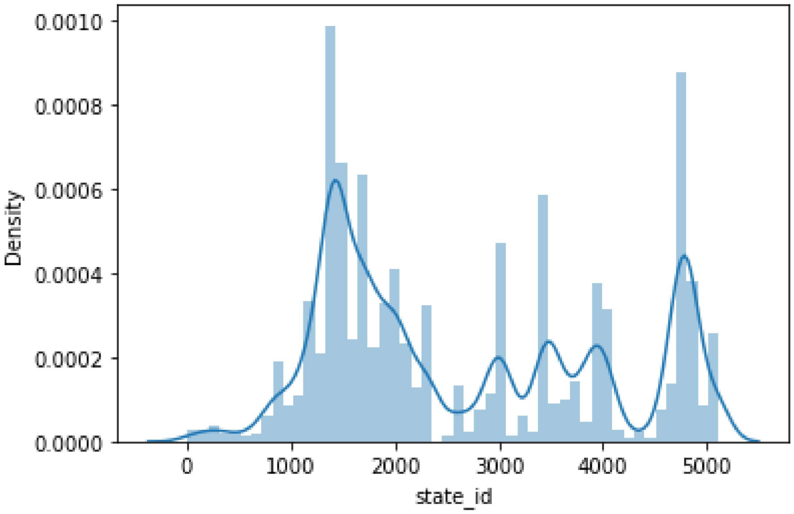
In [8]:

```
sns.distplot(df['state_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='state_id', ylabel='Density'>
```



In [9]:

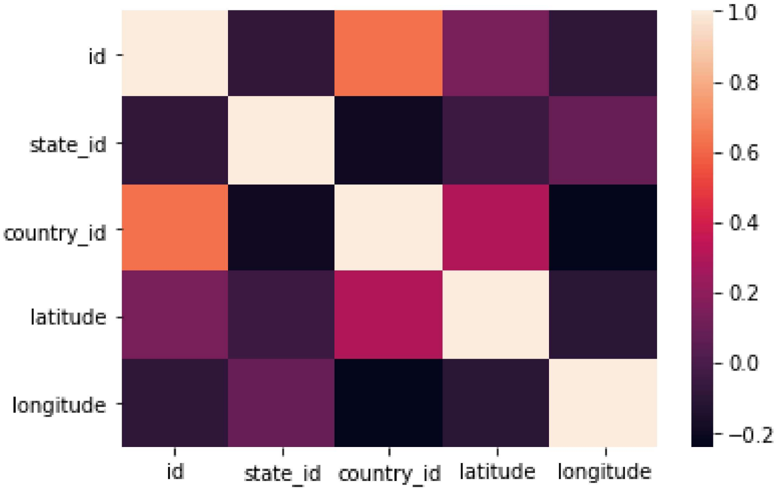
```
df1 = df[['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataId']]
```

In [10]:

```
sns.heatmap(df1.corr())
```

Out[10]:

```
<AxesSubplot:>
```



In [11]:

```
x = df1[[ 'state_id', 'country_id',
       'latitude', 'longitude' ]]
y = df1['id']
```

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]:

```
16761.512014101303
```

In [15]:

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

Out[15]:

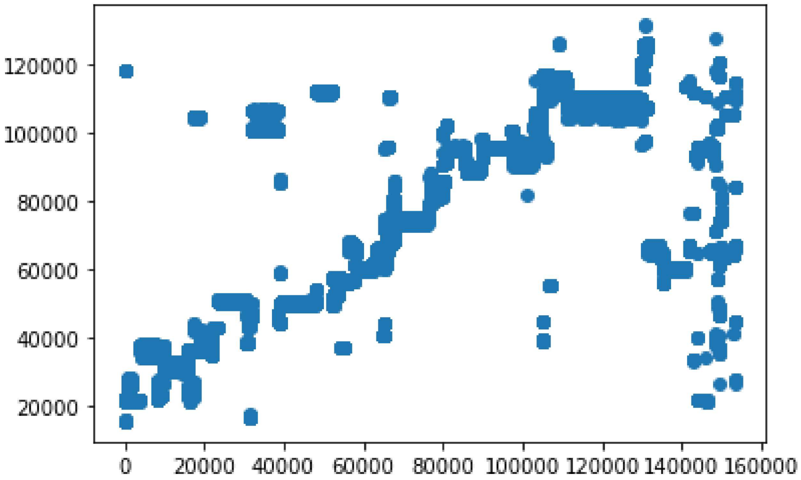
	Co-efficient
state_id	1.353642
country_id	419.405326
latitude	-96.578760
longitude	41.972795

In [16]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x239d78af640>
```



In [17]:

```
lr.score(x_test,y_test)
```

Out[17]:

```
0.4088022617416732
```

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]:

```
0.4037205135331342
```

In [20]:

```
rr.score(x_test,y_test)
```

Out[20]:

```
0.4088022618370707
```

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]:

```
0.408802357223611
```

In [23]:

```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:

```
ElasticNet()
```

In [24]:

```
print(en.coef_)
```

```
[  1.3531898  419.34436461 -96.43451329  41.95872567]
```

In [25]:

```
print(en.intercept_)
```

```
16766.7846160139
```

In [26]:

```
print(en.predict(x_test))
```

```
[ 27226.19833714  21885.93724355  32737.32980837 ...  80897.04414038
  25583.38344341 112246.2501921 ]
```

In [27]:

```
print(en.score(x_test,y_test))
```

```
0.40880275794908827
```

Evaluation Metrics

In [28]:

```
from sklearn import metrics
```

In [29]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 22616.64507184908
```

In [30]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 1156651516.1327608
```

In [31]:

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

```
Root Mean Squared Error: 34009.57977001128
```

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 [36]:

```
real = [[10,20,30,40],[11,45,42,13]]
result = model.predict(real)
```

In [37]:

```
result
```

Out[37]:

```
array([23944.70394288, 32138.98015648])
```