Importing libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing dataset

In [2]:

```
data=pd.read_csv(r"C:\Users\user\Downloads\cities.csv")
data
```

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

info

In [3]:

```
# to identify missing values
data.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
```

describe

In [4]:

```
# to display summary of the dataset
data.describe()
```

Out[4]:

	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

columns

In [5]:

```
# to display headings of the dataset
data.columns
```

Out[5]:

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

In [6]:

```
a=data.dropna(axis=1)
a
```

Out[6]:

	id	name	state_id	state_name	country_id	country_name	latitude	longitude
0	52	Ashkāsham	3901	Badakhshan	1	Afghanistan	36.68333	71.53333
1	68	Fayzabad	3901	Badakhshan	1	Afghanistan	37.11664	70.58002
2	78	Jurm	3901	Badakhshan	1	Afghanistan	36.86477	70.83421

	id	name	state_id	state_name	country_id	country_name	latitude	longitude
3	84	Khandūd	3901	Badakhshan	1	Afghanistan	36.95127	72.31800
4	115	Rāghistān	3901	Badakhshan	1	Afghanistan	37.66079	70.67346
...	...	...	...	...	...	...	...	...
150449	131496	Redcliff	1957	Midlands Province	247	Zimbabwe	-19.03333	29.78333
150450	131502	Shangani	1957	Midlands Province	247	Zimbabwe	-19.78333	29.36667
150451	131503	Shurugwi	1957	Midlands Province	247	Zimbabwe	-19.67016	30.00589
150452	131504	Shurugwi District	1957	Midlands Province	247	Zimbabwe	-19.75000	30.16667
150453	131508	Zvishavane District	1957	Midlands Province	247	Zimbabwe	-20.30345	30.07514

150454 rows × 8 columns

In [7]:

```
a.columns
```

Out[7]:

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

To train the model-Model Building

In [8]:

```
x=a[[ 'id']]
y=a['country_id']
```

In [9]:

```
# to split my dataset into training 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)
```

Linear regression

In [10]:

```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[10]:

```
LinearRegression()
```

In [11]:

```
print(lr.intercept_)
```

```
63.976158623252175
```

In [12]:

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

Out[12]:

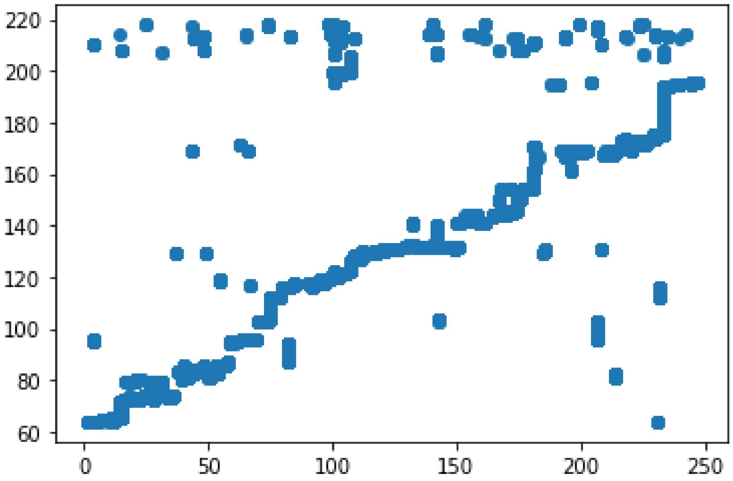
	Co-efficient
id	0.001002

In [13]:

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

Out[13]:

```
<matplotlib.collections.PathCollection at 0x1e60cbd11f0>
```



In [14]:

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

```
0.4002211834789693
```

In [15]:

```
lr.score(x_train,y_train)
```

Out[15]:

```
0.39602347546627725
```

Ridge regression

In [16]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [17]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[17]:

```
0.4002211834789692
```

In [18]:

```
rr.score(x_train,y_train)
```

Out[18]:
```
0.39602347546627725
```

Lasso regression

In [19]:
```python
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_train,y_train)
```

Out[19]:
```
0.3960234754561025
```

In [20]:
```python
la.score(x_test,y_test)
```

Out[20]:
```
0.4002211608355648
```

Elastic net regression

In [21]:
```python
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[21]:
```
ElasticNet()
```

In [22]:
```python
print(en.coef_)
```

```
[0.00100244]
```

In [23]:
```python
print(en.intercept_)
```

```
63.976178067479765
```

In [24]:
```python
predict=en.predict(x_test)
```

In [25]:
```python
print(en.score(x_test,y_test))
```

```
0.40022118234614756
```

In [26]:
```python
from sklearn import metrics
```

In [27]:
```python
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute error: 43.914578294509376
```

In [28]:
```python
print("Mean Squared error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Squared error: 2994.8423056261063
```

In [29]:

```
print("Root squared error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root squared error: 54.725152403863675
```

Model saving

In [30]:

```
import pickle
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
filename='prediction'
model=pickle.load(open(filename,'rb'))
```

In [31]:

```
real=[[10],[7]]
result=model.predict(real)
result
```

Out[31]:

```
array([63.98618302, 63.9831757 ])
```

In [ ]: