Type *Markdown* and LaTeX: α^2

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

In [2]:
```
#import dataset
df=pd.read_csv(r"E:\154\drive-download-20230731T110444Z-001\21_cities.csv").dropna()
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

146959 rows × 11 columns

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 146959 entries, 0 to 150453
Data columns (total 11 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   id            146959 non-null  int64  
 1   name          146959 non-null  object 
 2   state_id      146959 non-null  int64  
 3   state_code    146959 non-null  object 
 4   state_name    146959 non-null  object 
 5   country_id    146959 non-null  int64  
 6   country_code  146959 non-null  object 
 7   country_name  146959 non-null  object 
 8   latitude      146959 non-null  float64
 9   longitude     146959 non-null  float64
 10  wikiDataId    146959 non-null  object 
dtypes: float64(2), int64(3), object(6)
memory usage: 13.5+ MB
```

In [4]:
```
#to display top 5 rows
df.head()
```

Out[4]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_name	latitud
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanistan	36.6833
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanistan	37.1166
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanistan	36.8647
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanistan	36.9512
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanistan	37.6607

Data cleaning and Pre-Processing

In [5]:
```
#To find null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 146959 entries, 0 to 150453
Data columns (total 11 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   id            146959 non-null  int64  
 1   name          146959 non-null  object 
 2   state_id      146959 non-null  int64  
 3   state_code    146959 non-null  object 
 4   state_name    146959 non-null  object 
 5   country_id    146959 non-null  int64  
 6   country_code  146959 non-null  object 
 7   country_name  146959 non-null  object 
 8   latitude      146959 non-null  float64
 9   longitude     146959 non-null  float64
 10  wikiDataId    146959 non-null  object 
dtypes: float64(2), int64(3), object(6)
memory usage: 13.5+ MB
```

In [6]:
```
# To display summary of statistics
df.describe()
```

Out[6]:

	id	state_id	country_id	latitude	longitude
count	146959.000000	146959.000000	146959.000000	146959.000000	146959.000000
mean	74815.628618	2664.711124	139.865929	32.037738	2.043877
std	43408.845239	1351.363226	70.916362	22.701273	68.681166
min	1.000000	1.000000	1.000000	-75.000000	-179.121980
25%	37448.500000	1452.000000	82.000000	19.521795	-61.751760
50%	74428.000000	2131.000000	142.000000	40.896880	8.837560
75%	112849.500000	3904.000000	207.000000	47.349195	27.623705
max	153528.000000	5116.000000	247.000000	73.508190	179.466000

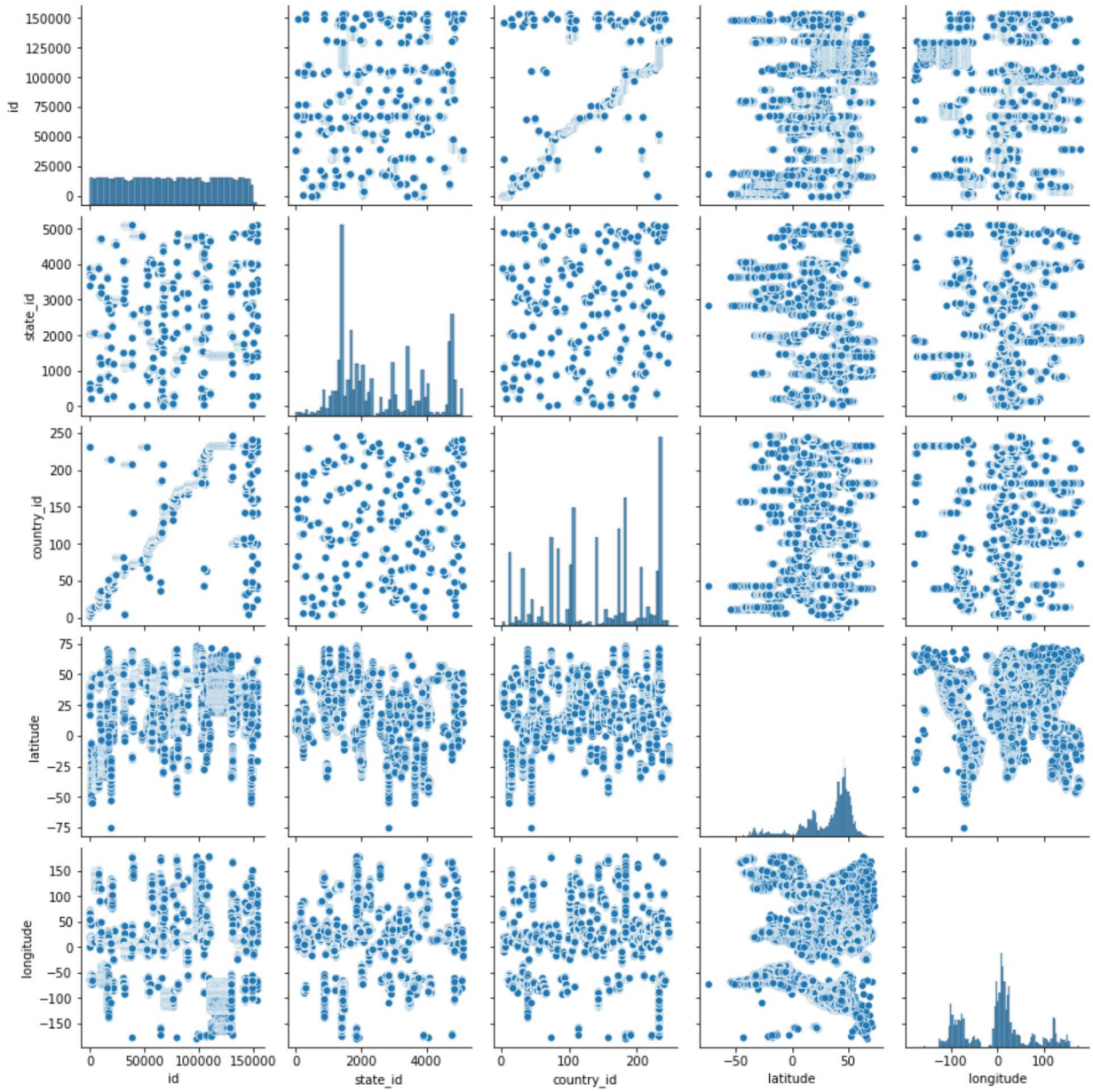
In [7]:
```
#To Display column heading
df.columns
```

Out[7]:
```
Index(['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataId'],
      dtype='object')
```

EDA and VISUALIZATION

In [8]:
```
sns.pairplot(df)
```

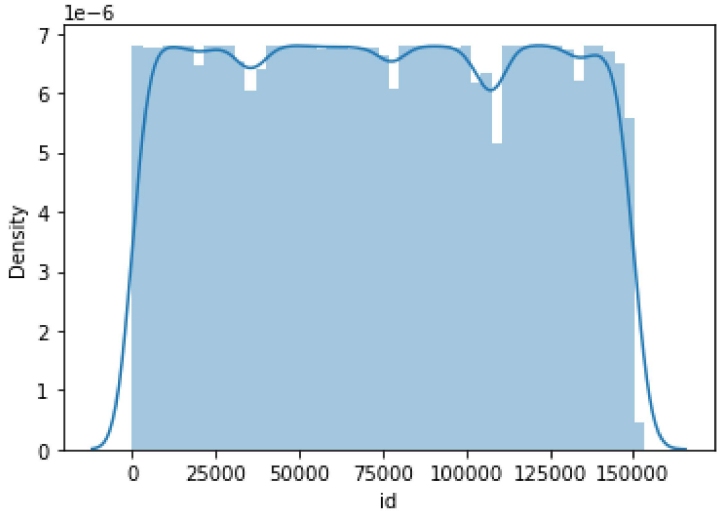
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x2a714844f40>
```



In [9]:
```
sns.distplot(df['id'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWa
rning: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with si
milar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]: <AxesSubplot:xlabel='id', ylabel='Density'>

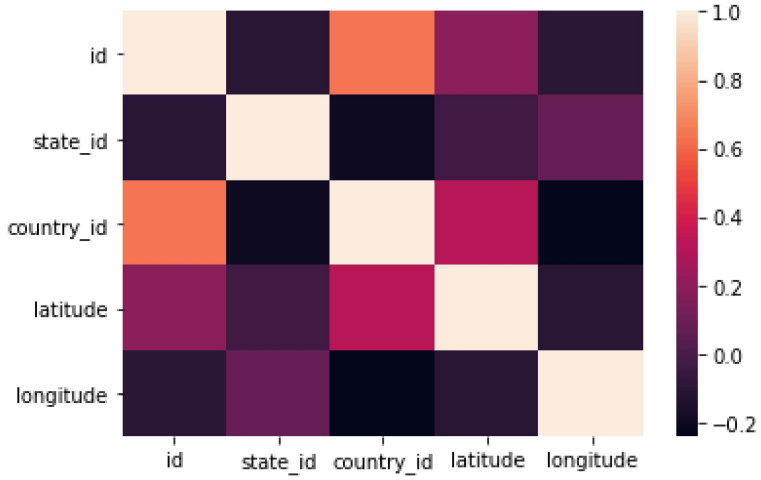


In [10]:
```
df1=df[['id','state_id','country_id', 'latitude','longitude'
       ]]
```

Plot Using Heat Map

In [11]:
```
sns.heatmap(df1.corr())
```

Out[11]: <AxesSubplot:>



To Train The Model-Model Building

we are going to train Linera Regression Model;We need to split out data into two variables x and y where x is independent variable(input) and y is dependent on x(output) we could ignore address column as it required for our model

In [12]:
```
x=df1[[
       'id','state_id','country_id', 'latitude']]
y=df1[ 'longitude']
```

To Split my dataset into training and test data

In [13]:
```
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]:
```
lr.intercept_
```

Out[15]:
```
27.369725744708344
```

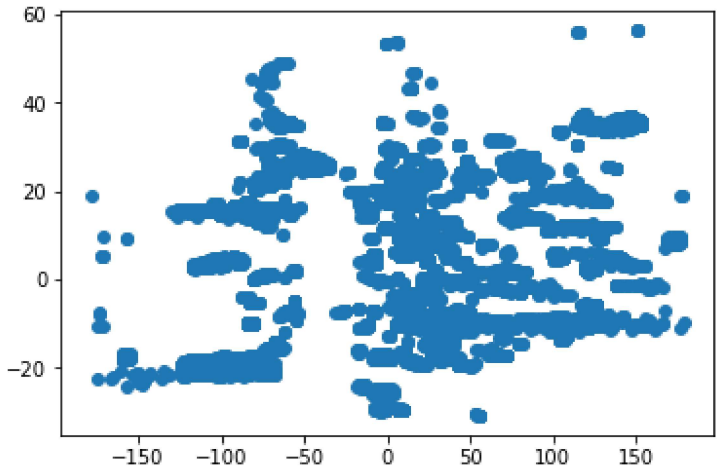
In [16]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
id	0.000142
state_id	0.002041
country_id	-0.270356
latitude	-0.104293

In [17]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x2a717be3b20>
```



In [18]:
```
lr.score(x_test,y_test)
```

Out[18]:
```
0.06326362525502371
```

Accuracy

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

Out[19]:
```
0.06326362525502371
```

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

Out[20]:
```
0.06522524226677806
```

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

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

Out[22]:
```
Ridge(alpha=10)
```

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

Out[23]:
```
0.06326362540720365
```

In [24]:
```
la =Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]:
```
Lasso(alpha=10)
```

In [25]:
```
la.score(x_test,y_test)
```

Out[25]:
```
0.0633559075576785
```

ElasticNet

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

Out[26]:
```
ElasticNet()
```

In [27]:
```
print(en.coef_)
```

```
[ 1.41307502e-04  2.04140385e-03 -2.70253957e-01 -1.03249443e-01]
```

In [28]:
```
print(en.intercept_)
```

```
27.33574886307409
```

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

```
[-18.97412408   9.61248672  17.83953064 ...   3.57094072  18.81246087
   4.82437061]
```

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

```
0.06327108515987312
```

Evaluation Metrics

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

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

```
Mean Absolute Error 52.188409995857796
```

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

```
Mean Squared Error: 4384.723564693465
```

In [34]:
```
print("Root Mean Absolute Error:",np.sqrt(metrics.mean_squared_error(y_test,predict
```

```
Root Mean Absolute Error: 66.21724522126743
```

In [35]:
```
print("Root Mean Absolute Error:",np.sqrt(metrics.mean_squared_error(y_test,predict
```

```
Root Mean Absolute Error: 66.21724522126743
```

Model Saving

In [36]:
```
import pickle
```

In [37]:
```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

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

In [39]:
```
real=[[10,20,30,40],[11,45,10,56]]
```

In [40]:
```
result =model.predict(real)
```

In [41]:
```
print(result)
```

```
[15.12954967 18.91914944]
```

In [ ]:
```

```