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\20_states.csv").dropna
df
```

Out[2]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	lo
170	3656	Buenos Aires	11	AR	Argentina	B	province	-37.201729	-59
171	3647	Catamarca	11	AR	Argentina	K	province	-28.471588	-65
172	3640	Chaco	11	AR	Argentina	H	province	-27.425718	-59
173	3651	Chubut	11	AR	Argentina	U	province	-43.293425	-65
174	4880	Ciudad Autónoma de Buenos Aires	11	AR	Argentina	C	city	-34.603684	-58
...	...	...	...	...	...	...	...	...	
4968	2041	Yaracuy	239	VE	Venezuela	U	state	10.339389	-68
4969	2042	Zulia	239	VE	Venezuela	V	state	10.291024	-72
5033	5074	Saint Croix	242	VI	Virgin Islands (US)	SC	district	17.729352	-64
5034	5073	Saint John	242	VI	Virgin Islands (US)	SJ	district	18.335617	-64
5035	5072	Saint Thomas	242	VI	Virgin Islands (US)	ST	district	18.342846	-65

1580 rows × 9 columns

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1580 entries, 170 to 5035
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            1580 non-null   int64  
 1   name          1580 non-null   object 
 2   country_id    1580 non-null   int64  
 3   country_code  1580 non-null   object 
 4   country_name  1580 non-null   object 
 5   state_code    1580 non-null   object 
 6   type          1580 non-null   object 
 7   latitude      1580 non-null   float64
 8   longitude     1580 non-null   float64
dtypes: float64(2), int64(2), object(5)
memory usage: 123.4+ KB
```

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

Out[4]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	lon
170	3656	Buenos Aires	11	AR	Argentina	B	province	-37.201729	-59.8
171	3647	Catamarca	11	AR	Argentina	K	province	-28.471588	-65.7
172	3640	Chaco	11	AR	Argentina	H	province	-27.425718	-59.0
173	3651	Chubut	11	AR	Argentina	U	province	-43.293425	-65.
174	4880	Ciudad Autónoma de Buenos Aires	11	AR	Argentina	C	city	-34.603684	-58.3

Data cleaning and Pre-Processing

In [5]:

```
#To find null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1580 entries, 170 to 5035
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            1580 non-null   int64  
 1   name          1580 non-null   object 
 2   country_id    1580 non-null   int64  
 3   country_code  1580 non-null   object 
 4   country_name  1580 non-null   object 
 5   state_code    1580 non-null   object 
 6   type          1580 non-null   object 
 7   latitude      1580 non-null   float64
 8   longitude     1580 non-null   float64
dtypes: float64(2), int64(2), object(5)
memory usage: 123.4+ KB
```

In [6]:

```
# To display summary of statistics
df.describe()
```

Out[6]:

	id	country_id	latitude	longitude
count	1580.000000	1580.000000	1580.000000	1580.000000
mean	2685.916456	134.000633	26.988930	15.009671
std	1611.169440	66.055166	19.635279	66.200355
min	48.000000	11.000000	-54.805400	-178.116500
25%	1339.750000	75.000000	13.752013	-7.622388
50%	2210.500000	139.000000	30.887089	11.665277
75%	4013.250000	178.000000	42.938004	45.682217
max	5220.000000	242.000000	77.874972	166.649935

In [7]:

```
#To Display column heading
df.columns
```

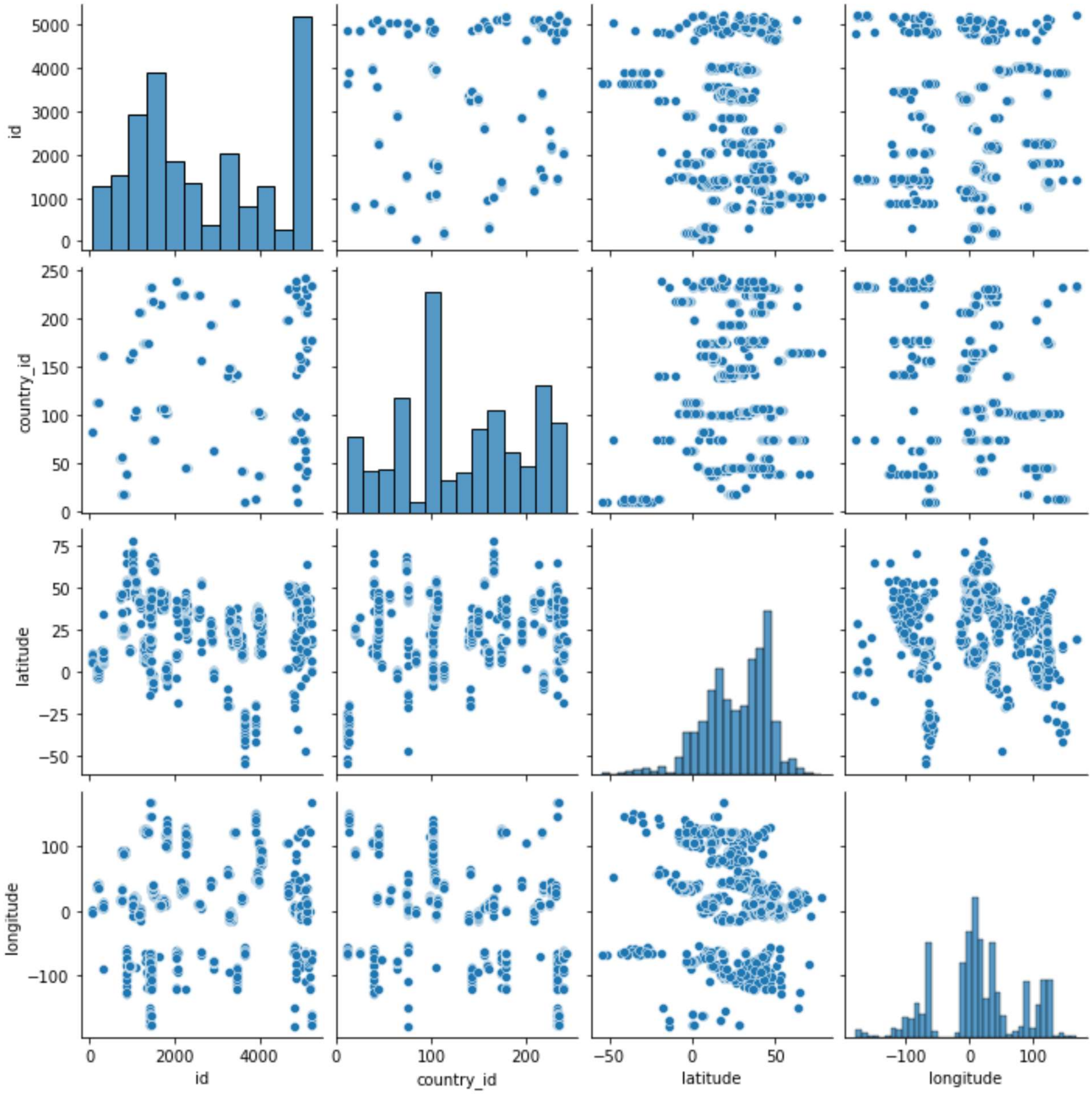
Out[7]:

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

EDA and VISUALIZATION

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

Out[8]: <seaborn.axisgrid.PairGrid at 0x159f74529d0>
```



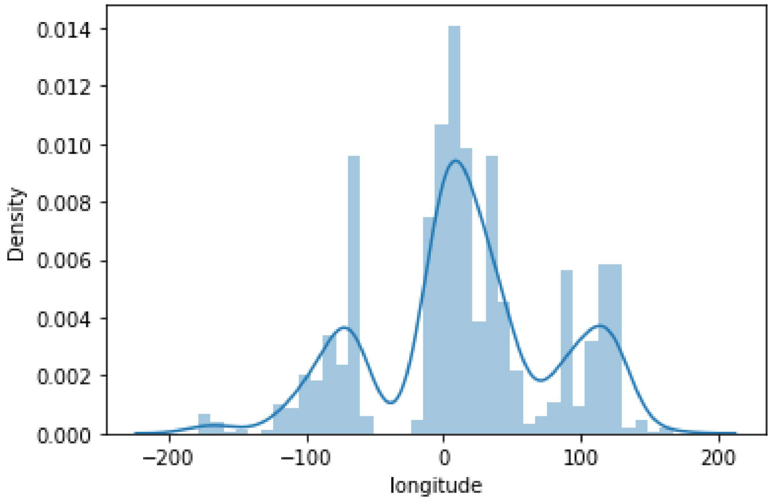
In [9]:

```
sns.distplot(df['longitude'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureW
arning: `distplot` is a deprecated function and will be removed in a future versi
on. 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[9]:

```
<AxesSubplot:xlabel='longitude', ylabel='Density'>
```



In [10]:

```
df1=df[['id','country_id', 'latitude','longitude'
       ]]
```

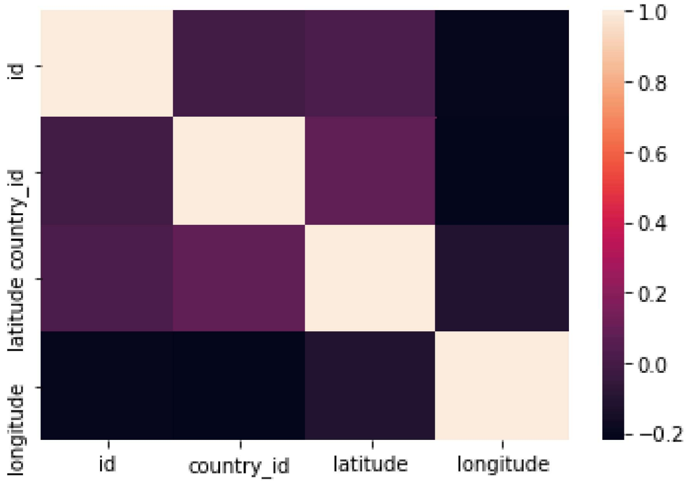
Plot Using Heat Map

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','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]:
```
69.88512017384633
```

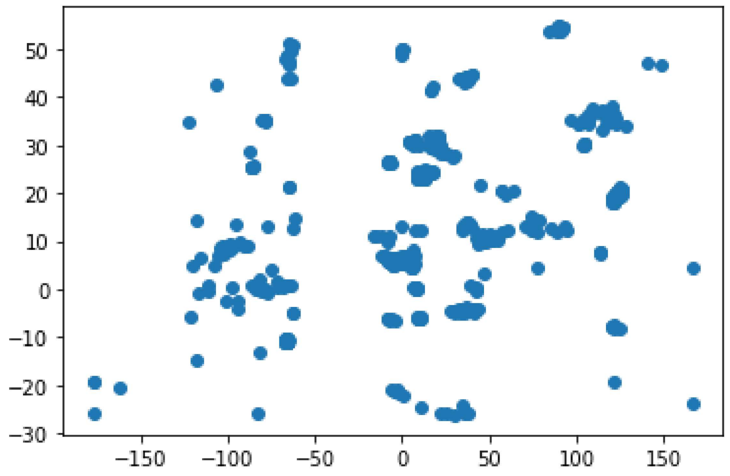
In [16]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
id	-0.007278
country_id	-0.217729
latitude	-0.236816

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

Out[17]:
```
<matplotlib.collections.PathCollection at 0x159f9042640>
```



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

Out[18]: 0.10055703466560706
```

Accuracy

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

Out[19]: 0.10055703466560706
```

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

Out[20]: 0.0904811044679853
```

```
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]: 0.10055692068940869
```

```
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]: 0.09996733626958476
```

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_)

         [-0.00727859 -0.21763718 -0.23528058]
```

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

         69.83122292494286
```

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

```
[-4.22485152e+00  1.12566122e+01  3.52311015e+01 -6.02626880e+00
  6.50846781e+00  1.17778772e+01  4.32514695e+01  6.51429911e+00
 -2.09029462e+01  2.37112117e+01 -1.09340160e+01  3.42660808e+01
 -4.32407664e+00  1.25859531e-01  9.63913303e+00  2.11002486e+01
  4.35922537e+00  1.04490732e+01  4.37809134e+01  2.43111730e+01
  2.03274788e+01 -6.21787685e+00  2.31463592e+01  2.65334598e+01
  2.63910542e+01  1.97665144e+01  5.44014220e+01  2.93340491e+01
  6.42631596e+00  3.00716425e+01  1.33007572e+01  2.16071254e+00
  2.36315904e+01  2.38769891e+01 -2.42455387e+01  1.98402507e+01
  3.38810100e+00  2.66045166e+01  3.60171637e+01  2.97632254e+01
  2.06525774e+01  5.42817474e+01  2.08586698e+01 -2.09721671e+01
  3.51532005e+01  2.55216500e+01 -7.17057998e-02  2.02255619e+01
 -1.11044568e+01  4.91257303e+00  1.54456160e+00  2.44017844e+01
  1.07826037e+01  6.55170807e+00  1.97115516e+01  7.10066803e+00
 -8.09098789e+00 -6.19920185e-01  1.36962229e+01  1.88686876e+01
  3.09161514e+01 -7.88755020e+00 -1.92278229e+01  2.95651494e+01
  1.08308905e+01  5.96914035e+00  3.00187408e+01  2.76914353e+01
  9.31109301e+00  7.66158883e+00  2.34114130e+01  5.10491971e+01
  6.56717229e+00  5.43499430e+00  1.18291642e+01 -4.87125718e+00
```

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

```
0.10052365459337198
```

Evaluation Metrics

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

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

```
Mean Absolute Error 49.979246514739344
```

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

```
Mean Squared Error: 4195.473860876993
```

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

```
Root Mean Absolute Error: 64.77247764966995
```

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

```
Root Mean Absolute Error: 64.77247764966995
```

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],[11,45,10]]
```

In [40]:

```
result =model.predict(real)
```

In [41]:

```
print(result)
```

```
[58.35327921 57.63907598]
```

In [ ]: