DATA COLLECTION

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

In [2]:
```
# To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\22_countries.csv")
sd
```

Out[2]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_nai
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan afgh
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	Eı
2	3	Albania	ALB	AL	8	355	Tirana	ALL	Albanian
3	4	Algeria	DZA	DZ	12	213	Algiers	DZD	Algerian dir
4	5	American Samoa	ASM	AS	16	+1-684	Pago Pago	USD	US Dol
...	...	...	...	...	...	...	...	...	
245	243	Wallis And Futuna Islands	WLF	WF	876	681	Mata Utu	XPF	CFP fra
246	244	Western Sahara	ESH	EH	732	212	El-Aaiun	MAD	Morocc Dirh
247	245	Yemen	YEM	YE	887	967	Sanaa	YER	Yemeni
248	246	Zambia	ZMB	ZM	894	260	Lusaka	ZMW	Zambi kwac
249	247	Zimbabwe	ZWE	ZW	716	263	Harare	ZWL	Zimbab Dol

250 rows × 19 columns

In [3]:
```
# to display top 10 rows
sd.head(10)
```

Out[3]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan afghani
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	Euro
2	3	Albania	ALB	AL	8	355	Tirana	ALL	Albanian lek
3	4	Algeria	DZA	DZ	12	213	Algiers	DZD	Algerian dinar
4	5	American Samoa	ASM	AS	16	+1-684	Pago Pago	USD	US Dollar
5	6	Andorra	AND	AD	20	376	Andorra la Vella	EUR	Euro
6	7	Angola	AGO	AO	24	244	Luanda	AOA	Angolan kwanza
7	8	Anguilla	AIA	AI	660	+1-264	The Valley	XCD	East Caribbean dollar
8	9	Antarctica	ATA	AQ	10	672	NaN	AAD	Antarctican dollar
9	10	Antigua And Barbuda	ATG	AG	28	+1-268	St. John's	XCD	Eastern Caribbean dollar

DATA CLEANING AND PRE_PROCESSING

In [4]:
```
sd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               250 non-null    int64  
 1   name             250 non-null    object 
 2   iso3             250 non-null    object 
 3   iso2             249 non-null    object 
 4   numeric_code     250 non-null    int64  
 5   phone_code       250 non-null    object 
 6   capital          245 non-null    object 
 7   currency         250 non-null    object 
 8   currency_name    250 non-null    object 
 9   currency_symbol  250 non-null    object 
 10  tld              250 non-null    object 
 11  native           249 non-null    object 
 12  region           248 non-null    object 
 13  subregion        247 non-null    object 
 14  timezones        250 non-null    object 
 15  latitude         250 non-null    float64
 16  longitude        250 non-null    float64
 17  emoji            250 non-null    object 
 18  emojiU           250 non-null    object 
dtypes: float64(2), int64(2), object(15)
memory usage: 37.2+ KB
```

In [5]:
```
# to display summary of statistics
sd.describe()
```

Out[5]:

	id	numeric_code	latitude	longitude
count	250.000000	250.00000	250.000000	250.00000
mean	125.500000	435.80400	16.402597	13.52387
std	72.312977	254.38354	26.757204	73.45152
min	1.000000	4.00000	-74.650000	-176.20000
25%	63.250000	219.00000	1.000000	-49.75000
50%	125.500000	436.00000	16.083333	17.00000
75%	187.750000	653.50000	39.000000	48.75000
max	250.000000	926.00000	78.000000	178.00000

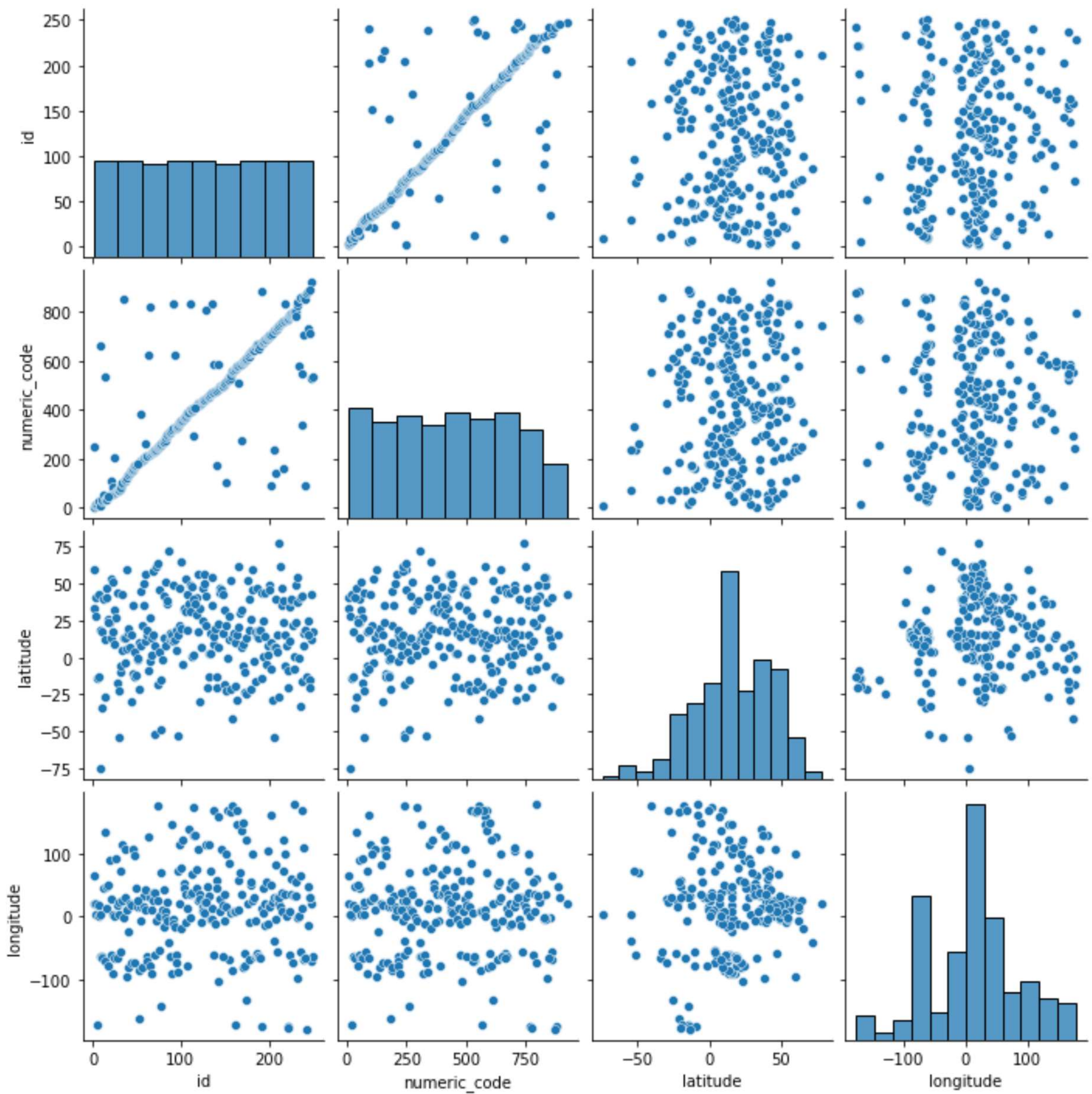
In [6]:
```
#to display colums heading
sd.columns
```

Out[6]:
```
Index(['id', 'name', 'iso3', 'iso2', 'numeric_code', 'phone_code', 'capital',
       'currency', 'currency_name', 'currency_symbol', 'tld', 'native',
       'region', 'subregion', 'timezones', 'latitude', 'longitude', 'emoji',
       'emojiU'],
      dtype='object')
```

EDA and visualization

In [7]:
```
sns.pairplot(sd)
```

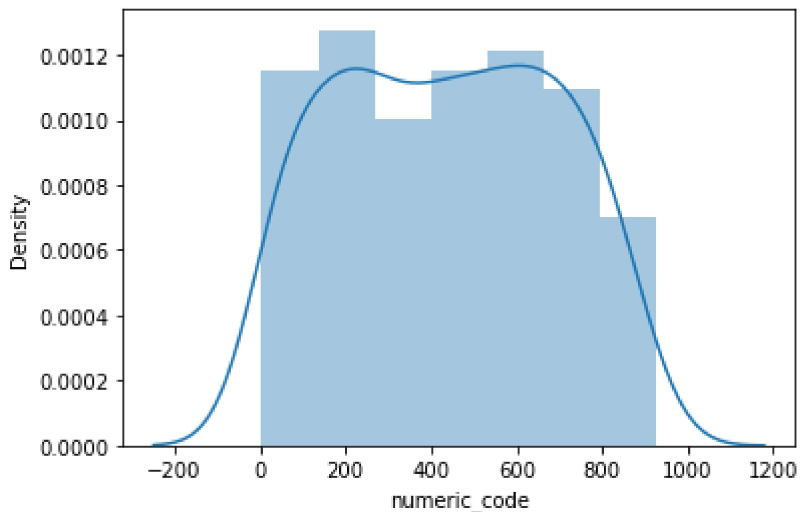
Out[7]: <seaborn.axisgrid.PairGrid at 0x1ebb7bb47f0>



In [8]:
```python
sns.distplot(sd['numeric_code'])
```

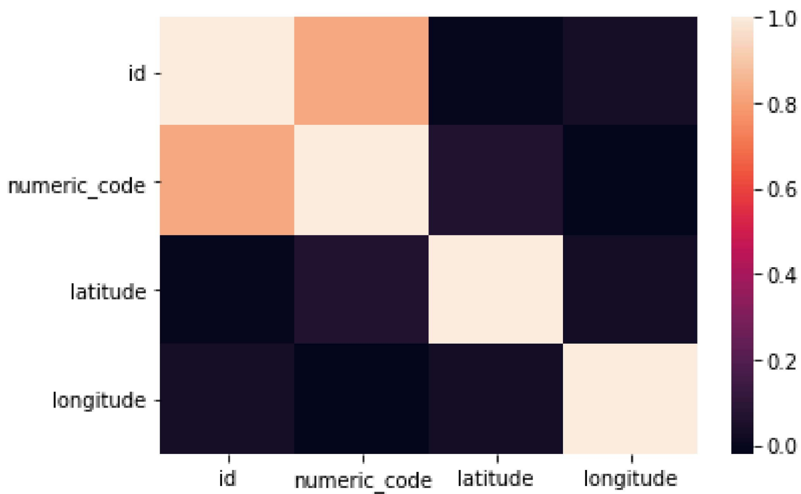
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[8]: <AxesSubplot:xlabel='numeric_code', ylabel='Density'>



In [9]:
```python
sns.heatmap(sd.corr())
```

Out[9]: <AxesSubplot:>



In [10]:
```python
sd1=sd[['numeric_code', 'phone_code', 'latitude', 'longitude']]
```

TO TRAIN THE MODEL _MODEL BUILDING

we are goint train Liner Regression model; we need to split out the data into two varibles x and y where x is independent on x (output) and y is dependent on x(output) adress coloumn as it is not required our model

In [14]:
```
x= sd1[['numeric_code', 'latitude']]
y=sd1['longitude']
```

In [15]:
```
# To split my dataset  into training data 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)
```

In [16]:
```
from sklearn.linear_model import LinearRegression

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

Out[16]:
```
LinearRegression()
```

In [17]:
```
from sklearn.linear_model import LinearRegression

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

Out[17]:
```
LinearRegression()
```

In [18]:
```
print(lr.intercept_)
```

```
15.555017701137126
```

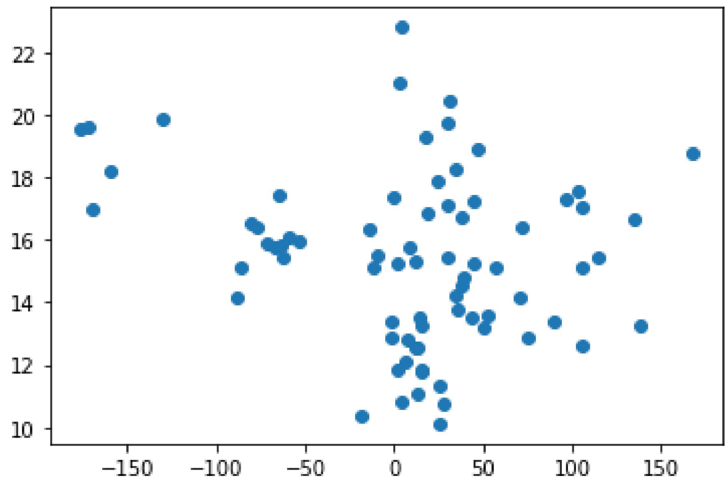
In [19]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[19]:

	Co-efficient
numeric_code	0.003094
latitude	-0.096791

In [20]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[20]: <matplotlib.collections.PathCollection at 0x1ebbe703430>



In [21]:
```
print(lr.score(x_test,y_test))
```

```
-0.024339356129793543
```

In [22]:
```
lr.score(x_train,y_train)
```

Out[22]: 0.0012457965482874922

In [23]:
```
from sklearn.linear_model import Ridge,Lasso
```

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

Out[24]: Ridge(alpha=10)

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

Out[25]: -0.02433794075950435

In [26]:
```
dr.score(x_train,y_train)
```

Out[26]: 0.0012457965401555526

In [27]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[27]: Lasso(alpha=10)

In [28]:
```
la.score(x_test,y_test)
```

Out[28]: -0.021705256651429528

In [29]:
```
la.score(x_train,y_train)
```
Out[29]:
```
0.0012188590068976657
```

ElasticNet

In [30]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[30]:
```
ElasticNet()
```

In [31]:
```
print(en.coef_)
```
```
[ 0.00308466 -0.09598231]
```

In [32]:
```
print(en.intercept_)
```
```
15.545312897663981
```

In [33]:
```
prediction=en.predict(x_test)
```

In [34]:
```
print(en.score(x_test,y_test))
```
```
-0.024193767736777616
```

Evaluation metric

In [35]:
```
from sklearn import metrics
```

In [36]:
```
print("mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```
```
mean Absolute Error: 50.078893917568394
```

In [37]:
```
print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```
```
mean squared Error: 5067.799058628259
```

In [38]:
```
print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```
```
Root mean Absolytre Error: 71.18847560264413
```

Model Saving

In [39]:
```
import pickle
```

In [40]:
```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [ ]:

In [ ]:

In [ ]: