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

In [2]:
```
df=pd.read_csv("22_countries.csv")
df
```

Out[2]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	cur
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	
...	...	...	...	...	...	...	...	...	...	
245	243	Wallis And Futuna Islands	WLF	WF	876	681	Mata Utu	XPF	CFP franc	
246	244	Western Sahara	ESH	EH	732	212	El-Aaiun	MAD	Moroccan Dirham	
247	245	Yemen	YEM	YE	887	967	Sanaa	YER	Yemeni rial	
248	246	Zambia	ZMB	ZM	894	260	Lusaka	ZMW	Zambian kwacha	
249	247	Zimbabwe	ZWE	ZW	716	263	Harare	ZWL	Zimbabwe Dollar	

250 rows × 19 columns

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

Out[3]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	currenc
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	

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	currenc
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	

Data Cleaning and Data Preprocessing

In [4]:
```
df.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]:
```
df.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

	id	numeric_code	latitude	longitude
max	250.000000	926.00000	78.000000	178.00000

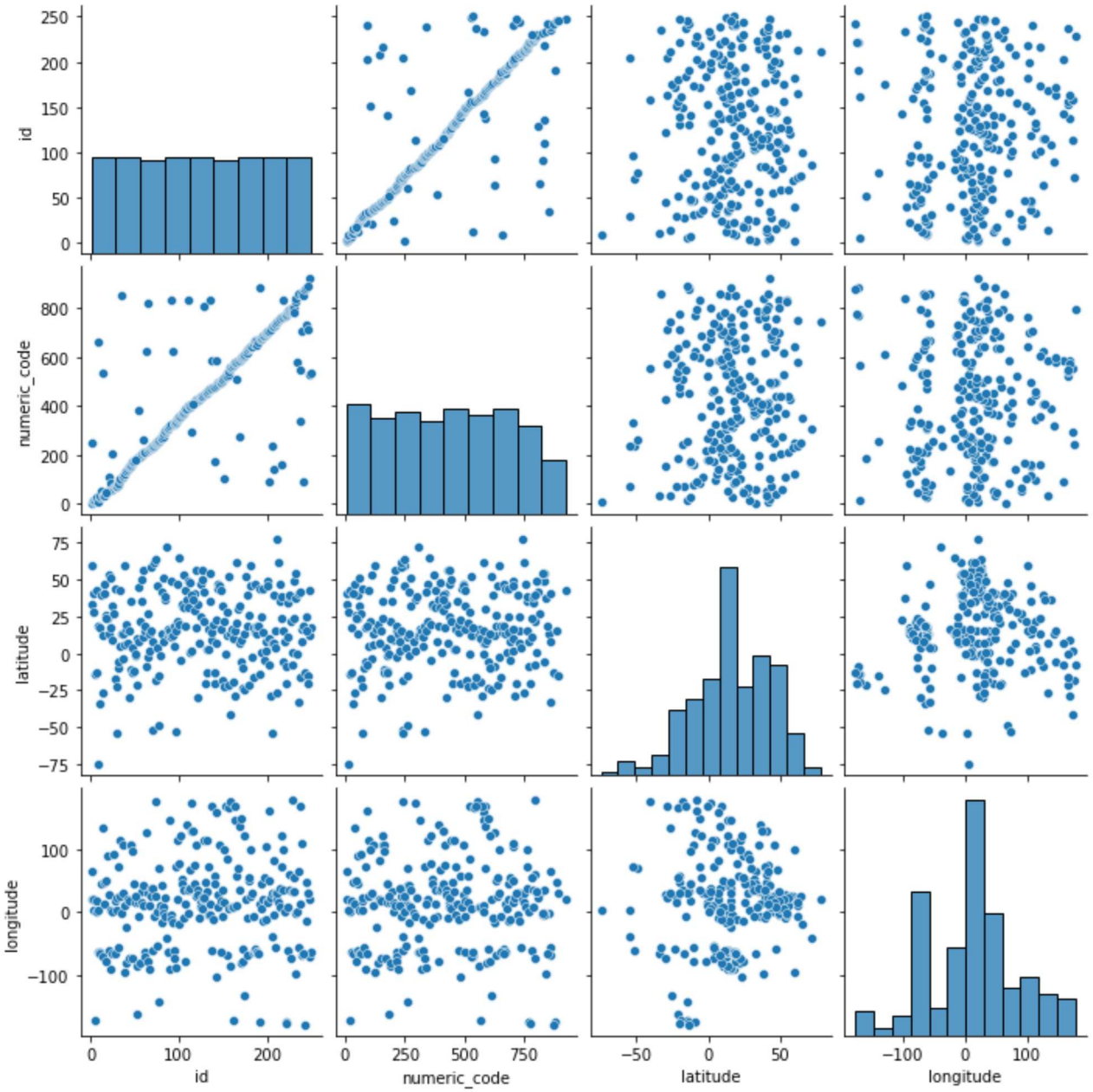
In [6]:
```
df.columns
```

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(df)
```

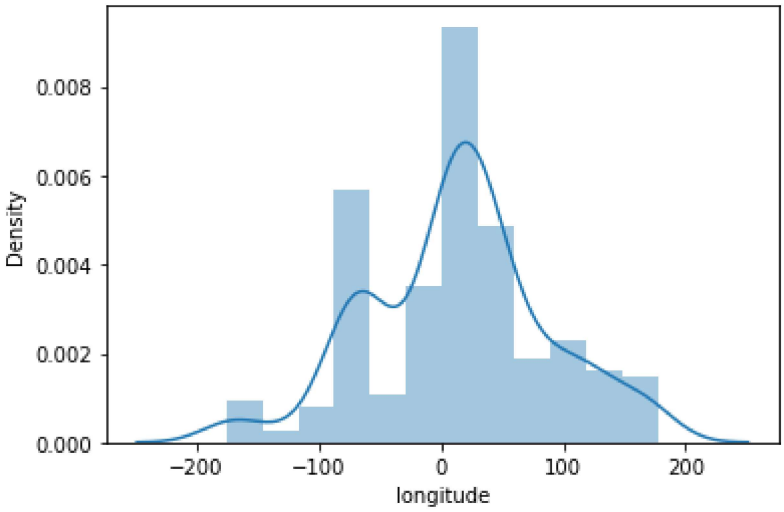
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x24c0c3c1d60>
```



In [8]:
```
sns.distplot(df['longitude'])
```

```
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='longitude', ylabel='Density'>



In [9]:
```
df1=df[['id','numeric_code','latitude','longitude']]
df1
```

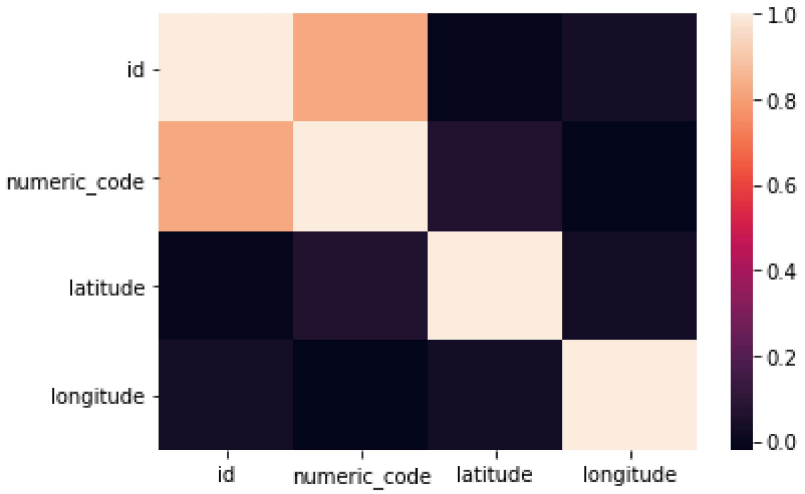
Out[9]:

	id	numeric_code	latitude	longitude
0	1	4	33.000000	65.0
1	2	248	60.116667	19.9
2	3	8	41.000000	20.0
3	4	12	28.000000	3.0
4	5	16	-14.333333	-170.0
...	...	...	...	...
245	243	876	-13.300000	-176.2
246	244	732	24.500000	-13.0
247	245	887	15.000000	48.0
248	246	894	-15.000000	30.0
249	247	716	-20.000000	30.0

250 rows × 4 columns

In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]: <AxesSubplot:>



To Train the Model -Model Building

We are going to train Linear Regression model;We need to spilt out data into two variables x and y where x is independent variable (input) and y is dependent variable on x(output) we could ignore address column as it is not requiired for our model

In [11]:
```
x=df1[['id','numeric_code','latitude']]
y=df1['longitude']
```

In [12]:
```
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 [13]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
print(lr.intercept_)
```

```
14.562418547471916
```

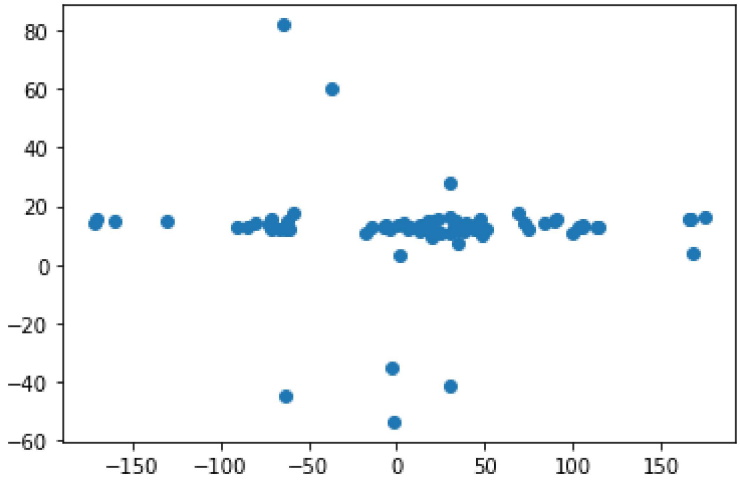
In [15]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
id	0.318339
numeric_code	-0.092246
latitude	-0.043757

In [16]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[16]: <matplotlib.collections.PathCollection at 0x24c0defe3d0>



In [17]:
```
lr.score(x_test,y_test)
```

Out[17]: -0.06828886162564762

In [18]:
```
lr.score(x_train,y_train)
```

Out[18]: 0.029945280708449817

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]: Ridge(alpha=10)

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

Out[21]: -0.06828421940268536

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

Out[22]: 0.029945280656274886

In [23]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[23]: Lasso(alpha=10)

In [24]:
```
la.score(x_test,y_test)
```

Out[24]: -0.06360880140376879

In [25]:
```
la.score(x_train,y_train)
```

Out[25]: 0.02989906133743714

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

Out[26]: ElasticNet()

In [27]:
```
en.coef_
```

Out[27]: array([0.3177984 , -0.09211182, -0.04304115])

In [28]:
```
en.intercept_
```

Out[28]: 14.561498779194935

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

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

Out[30]: -0.06801095514521993

Evaluation Metrics

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

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

Mean Absolute Error: 58.488513284794294

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

Mean Squared Error: 5952.4421025892025

In [34]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

Root Mean Squared Error: 77.15207127867147

Model Saving

In [35]:
```
import pickle
```

In [36]:
```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [37]:
```
import pandas as pd
import pickle
```

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

In [39]:
```
real=[[10,20,30],[11,45,10]]
result=model.predict(real)
```

In [40]:
```
result
```

Out[40]:
```
array([14.58819546, 13.47552903])
```