D19

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\22_countries.csv")
df
```

Out[3]:

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

250 rows × 19 columns

In [4]:
```
df.head(10)
```

Out[4]:

	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

In [5]:
```
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 [6]:
```
dff=df.dropna()
```

In [7]:
```
dff.describe()
```

Out[7]:

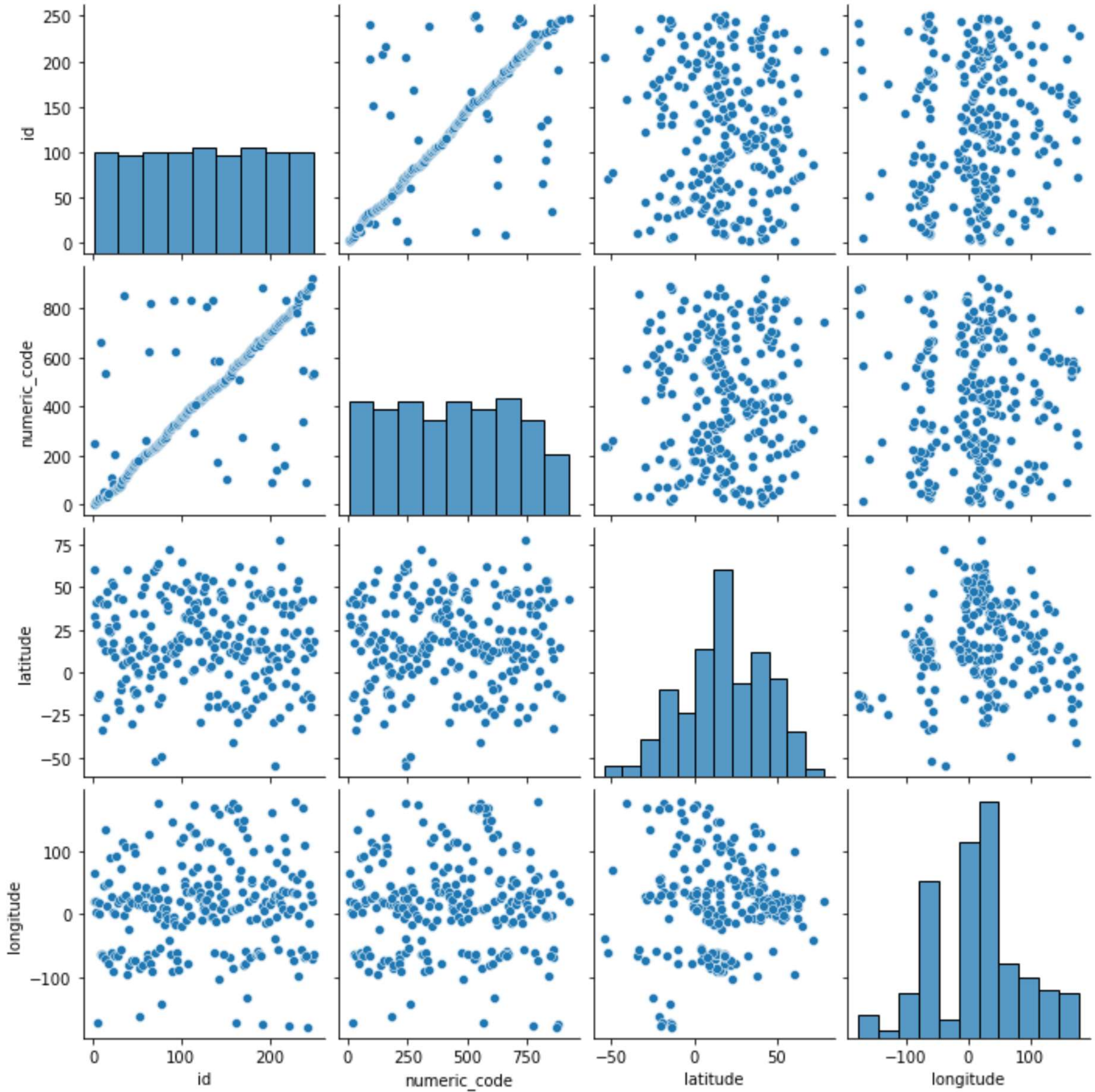
	id	numeric_code	latitude	longitude
count	243.000000	243.000000	243.000000	243.000000
mean	125.839506	437.366255	17.719476	14.241033
std	71.920662	254.274551	25.491128	73.423927
min	1.000000	4.000000	-54.500000	-176.200000
25%	64.500000	220.000000	1.708333	-54.000000
50%	126.000000	438.000000	17.000000	18.500000
75%	187.500000	656.500000	39.250000	49.775000
max	250.000000	926.000000	78.000000	178.000000

In [8]:
```
dff.columns
```

Out[8]:
```
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')
```

In [9]:
```
sns.pairplot(dff)
```

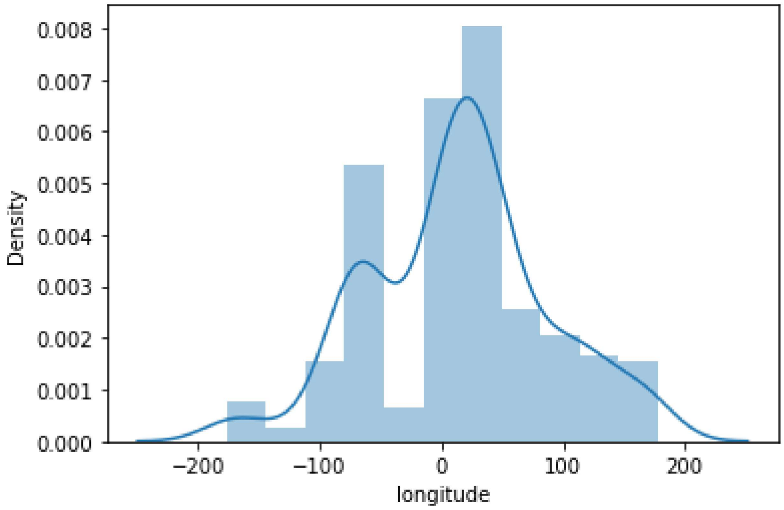
Out[9]: <seaborn.axisgrid.PairGrid at 0x1e813ac6c40>



In [10]:
```
sns.distplot(dff['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 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)
```

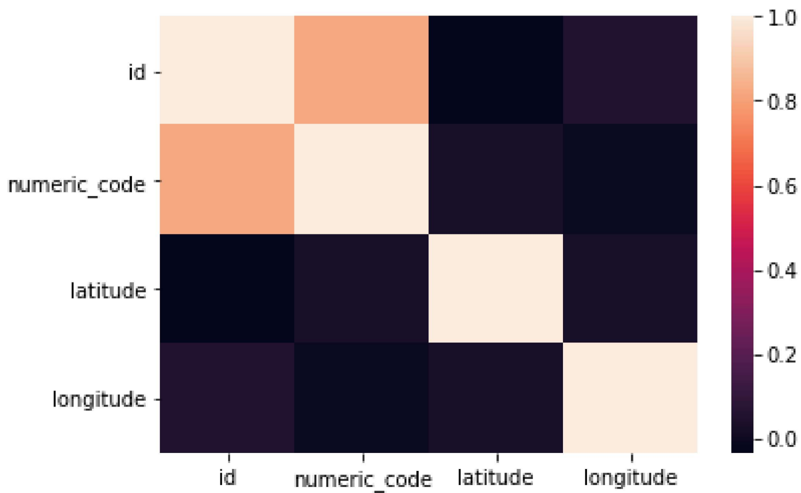
Out[10]: <AxesSubplot:xlabel='longitude', ylabel='Density'>



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

In [12]:
```
sns.heatmap(df1.corr())
```

Out[12]: <AxesSubplot:>



In [13]:
```
x=df1[['id', 'numeric_code' , 'latitude']]
y=df1['longitude']
```

In [14]:
```
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 [15]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]: LinearRegression()

In [16]:
```
print(lr.intercept_)
```

2.2949353281480604

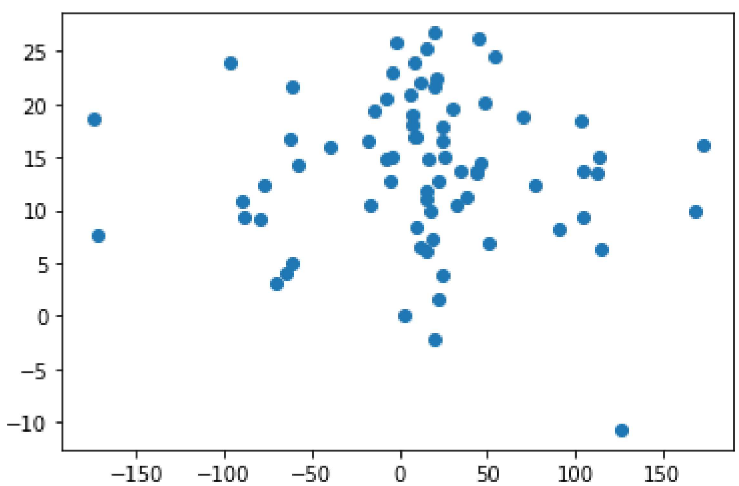
In [17]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
id	0.231902
numeric_code	-0.042811
latitude	0.094317

In [18]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[18]: <matplotlib.collections.PathCollection at 0x1e819dad490>



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

-0.026057481797213233

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[21]: Ridge(alpha=10)

In [22]:
```
rr.score(x_test,y_test)
```
Out[22]:
```
-0.02605691270360877
```

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]:
```
-0.02592857305329921
```

In [25]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[25]:
```
ElasticNet()
```

In [26]:
```
print(en.coef_)
```
```
[ 0.23149464 -0.04270898  0.09334494]
```

In [27]:
```
print(en.intercept_)
```
```
2.317344560640974
```

In [28]:
```
print(en.predict(x_train))
```

```
[ 18.39794438   6.54194112  44.48869528  15.49183606  13.67795579
   8.49973252  17.27053298  13.296066     5.31434681  -1.68391609
  25.79699725  11.32570962   6.65092023   7.49072241  21.19243126
  19.30668588  11.16738076   5.7118749   34.47900164   1.45352994
  15.38860943  10.43404317   9.97797412  12.38885651  47.76577559
  24.15238466  19.95046058  21.25578117  14.27281983  20.43340662
   7.20067207  13.59223964   6.09073801   5.09352325  19.22577732
  46.97400677   9.61199095   1.50043289  44.97155253  23.13464965
  14.35116656  18.77071502  17.84383872  22.89694336  19.00243773
  11.65144693  10.08784171  15.94710784   1.74597981  23.69119885
  16.60382023  11.65041918  13.84663503  16.92378014  14.16875968
  10.53602128   8.82328256  14.7638074   -7.49033979   7.43306444
  19.56652039  16.90417521   0.32337025   9.43623706   6.63717342
  22.58819407  16.99390022   5.49213146   7.63575484  15.24776368
  24.04750282  15.80289061  12.31937819  20.26959998   3.28800125
  24.41529497   5.9554124   32.67267446  16.66537274 -15.05113361
  19.67326274  13.6986944   11.40558218  16.34815056   6.82057261
   7.14188931  10.68137631   6.49729941  15.65121177   4.94109079
  18.20167411  23.04269673   7.72334185  12.82590369  20.83870296
  55.89880445  34.88489105  23.64789448  19.6999564   14.85613524
   4.67235418  13.6022634   15.47757295   9.3038453   18.32028076
  17.4837074   18.27176021  19.05769034  23.74836402   5.45838647
  21.53431904  29.31801576  17.13792142   5.06712195  29.82601751
  22.90053539  27.04999505  20.9650549  -24.84032537  24.21858231
  15.11380554  16.48385262  16.50686792  38.41207368   9.40295545
  13.73756713   3.75812681   7.82043048  15.46317871  16.93010252
   9.5506718   18.02476412   4.51458991  21.61555853   6.71427741
   9.93109736  22.55084367  10.80393757   6.4359555   13.03306769
   8.47775084  16.84076271  16.97425306   6.81929303  19.9159912
  39.06773225 -16.27029825   9.7635587  -22.31507818  11.99311049
  -3.15487526   5.66436758  15.63624626   5.34447386  18.46014338
  21.78518714   8.56032524  21.57618563   9.15148854  21.09257918
  13.82335242  12.33293901  15.72712908  15.68382471  18.61661672
  32.2835366   11.90350924   7.34260869  10.88052768  14.75980263]
```

In [29]:
```
print(en.score(x_train,y_train))
```

```
0.02038555333192482
```

In [30]:
```
from sklearn import metrics
```

In [31]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 46.24951625068402
```

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

```
Mean Square Error: 4282.361608652522
```

In [33]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 6.800699688317668
```

In [34]:
```
import pickle
```

In [35]:
```
f4="prediction"
pickle.dump(lr,open(f4,'wb'))
```

In [ ]: