In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\22_countries.csv")
df.fillna(0,inplace=True)
df
```

Out[2]:

ncy	currency_name	currency_symbol	tld	native	region	subregion	
FN	Afghan afghani	؋	.af	افغانستان	Asia	Southern Asia	[{zoneName:'Asia\/Kabul',
UR	Euro	€	.ax	Åland	Europe	Northern Europe	[{zoneName:'Europe\/Marie
LL	Albanian lek	Lek	.al	Shqipëria	Europe	Southern Europe	[{zoneName:'Europe\/Tira
ZD	Algerian dinar	دج	.dz	الجزائر	Africa	Northern Africa	[{zoneName:'Africa\/Algi
SD	US Dollar	$	.as	American Samoa	Oceania	Polynesia	[{zoneName:'Pacific\/Pago
...	...	...	...	...	...	...	
PF	CFP franc	₣	.wf	Wallis et Futuna	Oceania	Polynesia	[{zoneName:'Pacific\/Wa
AD	Moroccan Dirham	MAD	.eh	الصحراء الغربية	Africa	Northern Africa	[{zoneName:'Africa\/El_A
ER	Yemeni rial	﷼	.ye	اليَمَن	Asia	Western Asia	[{zoneName:'Asia\/Aden',
MW	Zambian kwacha	ZK	.zm	Zambia	Africa	Eastern Africa	[{zoneName:'Africa\/Lusa
WL	Zimbabwe Dollar	$	.zw	Zimbabwe	Africa	Eastern Africa	[{zoneName:'Africa\/Hara

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

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

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             250 non-null    object 
 4   numeric_code     250 non-null    int64  
 5   phone_code       250 non-null    object 
 6   capital          250 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           250 non-null    object 
 12  region           250 non-null    object 
 13  subregion        250 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]:
```
import seaborn as sns
```

In [6]:

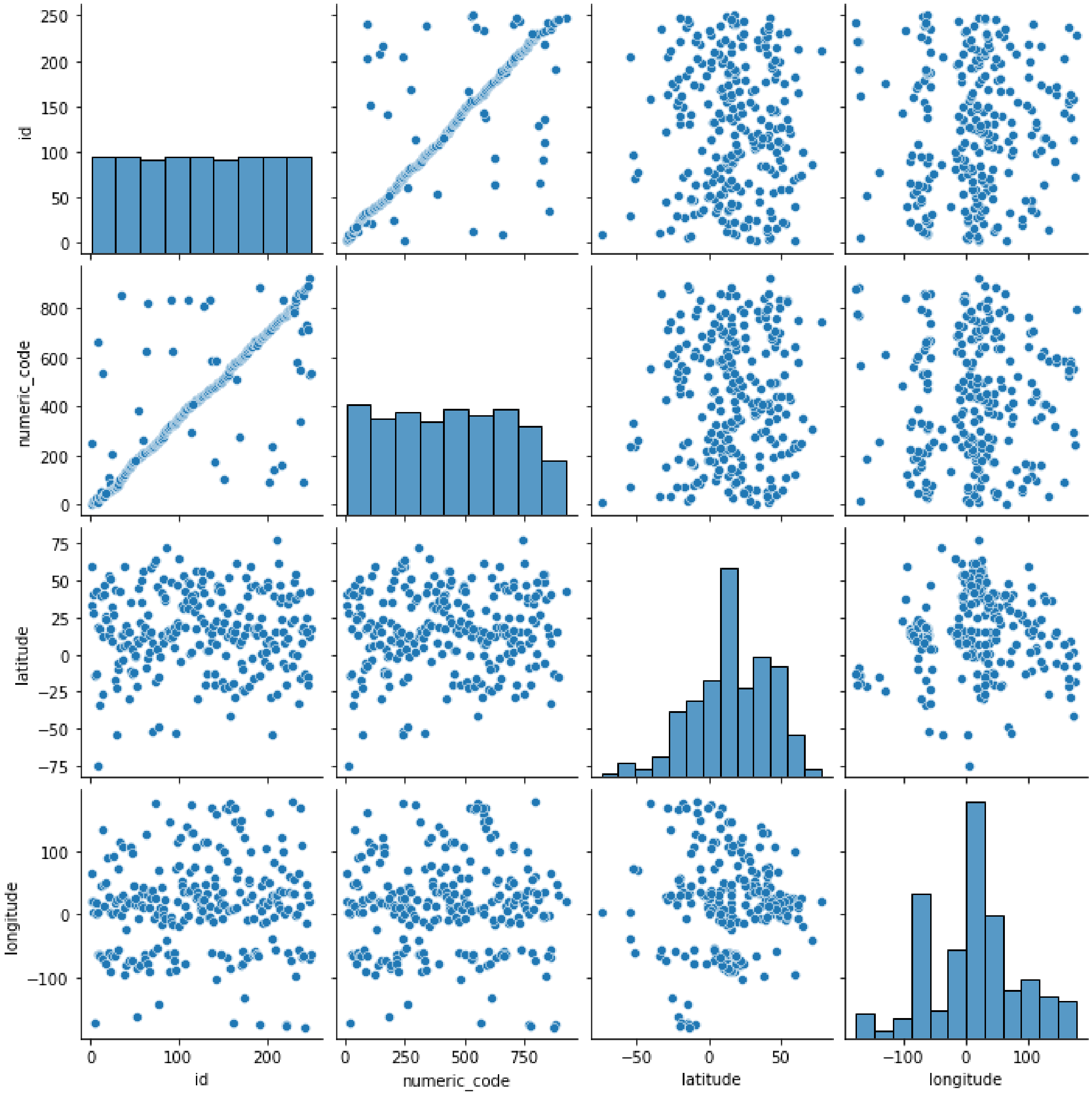
```
df.describe()
```

Out[6]:

	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 [7]: sns.pairplot(df)
```

```
Out[7]: <seaborn.axisgrid.PairGrid at 0x22374ff0640>
```



In [8]:

```
df1=df.drop(['name'],axis=1)
df1
df1=df1.drop(df1.index[1537:])
df1.isna().sum()
```

Out[8]:

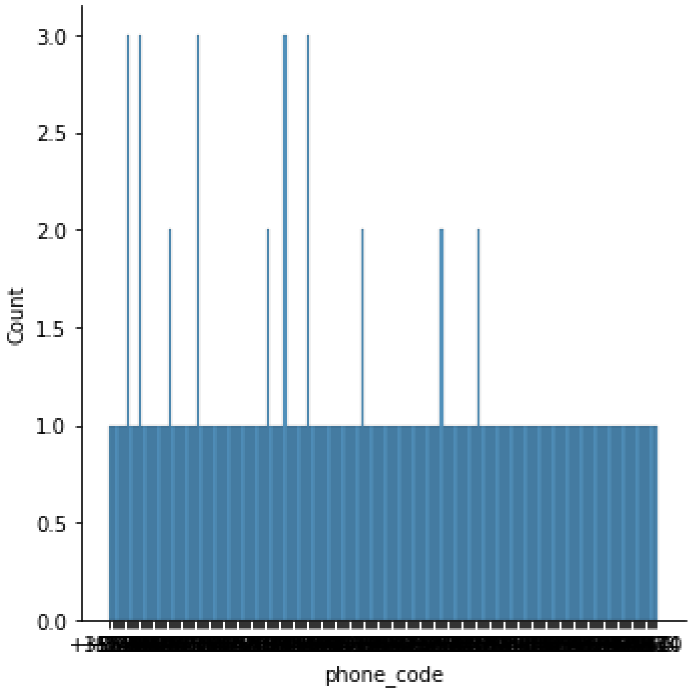
```
id                 0
iso3               0
iso2               0
numeric_code       0
phone_code         0
capital            0
currency           0
currency_name      0
currency_symbol    0
tld                0
native             0
region             0
subregion          0
timezones          0
latitude           0
longitude          0
emoji              0
emojiU             0
dtype: int64
```

In [9]:

```
sns.displot(df['phone_code'])
```

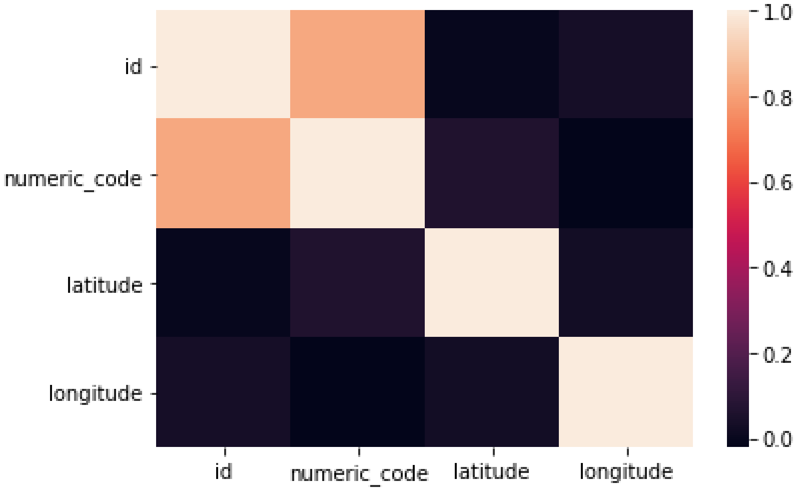
Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x22372244370>
```



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

```
Out[10]: <AxesSubplot:>
```



```
In [11]: from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression
```

```
In [12]: df1.isna().sum()
```

```
Out[12]: id              0
         country_id      0
         country_code    0
         country_name    0
         state_code      0
         type            0
         latitude        0
         longitude       0
         dtype: int64
```

In [18]:

```
['iso3','iso2','capital','currency','currency_name','currency_symbol','tld','nat
est,y_train,y_test=train_test_split(x,y,test_size=0.3)
.n)
```

```
      id  numeric_code   latitude   longitude
220  219           764  15.000000  100.000000
93    93           624  12.000000  -15.000000
99    99           348  47.000000   20.000000
232  230           804  49.000000   32.000000
206  116           410  37.000000  127.500000
..   ...           ...        ...         ...
113  113           404   1.000000   38.000000
18    20            52  13.166667  -59.533333
97    97           340  15.000000  -86.500000
38    39           124  60.000000  -95.000000
100  100           352  65.000000  -18.000000

[175 rows x 4 columns]
```

In [19]:

```
print(x)
```

```
      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 x 4 columns]
```

In [20]:

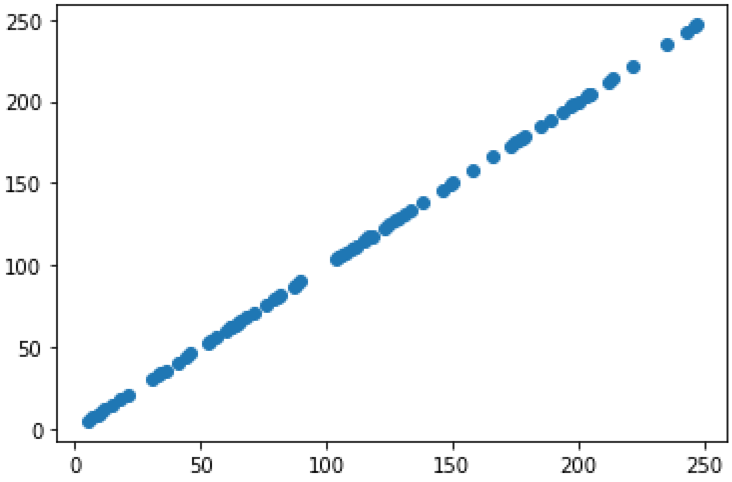
```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

Out[20]:

```
0.0
```

```
In [21]: prediction=model.predict(x_test)
         plt.scatter(y_test,prediction)

Out[21]: <matplotlib.collections.PathCollection at 0x22378aeaa60>
```



```
In [22]: model.score(x_test,y_test)

Out[22]: 1.0

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

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

Out[24]: Ridge(alpha=10)

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

Out[25]: 0.9999999997533163

In [26]: la =Lasso(alpha=10)
         la.fit(x_train,y_train)

Out[26]: Lasso(alpha=10)

In [27]: la.score(x_test,y_test)

Out[27]: 0.9999955974874192

In [28]: from sklearn.linear_model import ElasticNet
         en=ElasticNet()
         en.fit(x_train,y_train)

Out[28]: ElasticNet()

In [29]: print(en.coef_)

         [ 9.99714197e-01  5.83061746e-05 -0.00000000e+00  0.00000000e+00]
```

In [30]:
```
print(en.intercept_)
```

```
0.011358011414245084
```

In [31]:
```
print(en.predict(x_test))
```

```
[177.99721796 175.99697328 213.99427562 105.00303859  18.00901225
 184.99690821  21.01188644  10.01013255 127.00083235  60.00948605
 117.00205781 115.00227958 221.99315532 148.99815967 157.99850275
 137.99955432  36.00736617  33.0075239   31.00692939 204.96670356
  53.00717201  79.00428901 203.99445157  65.04047526  62.00611574
 234.99422099 193.99567703 129.0215425   90.00429371 211.99438078
 118.00194693  68.00545044   7.01075674 172.99713101 104.00309117
 110.02843041 133.00028366 202.99450415 196.99528607 125.00117073
 245.99317618 246.98251188  56.00654783 199.99518664  15.00940321
  44.00764522  81.00383402 123.00127589  76.00444675  12.01090199
 106.00298601  54.01831422  64.00577736  63.02985208 174.99702585
   5.01086189  66.00543898  46.00765667 165.99376746  41.00756973
 145.99855063  88.00439887 188.99535686 131.00038881   9.00936885
 242.99298408  34.00747132 178.99716538  71.00494286 149.99810709
  87.00445145 112.00255393  82.00401466 197.99523349 108.00311408]
```

In [32]:
```
print(en.score(x_test,y_test))
```

```
0.999999977680215
```

In [33]:
```
from sklearn import metrics
```

In [34]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 2.073600550526559e-14
```

In [35]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 2.1197113249104705e-27
```

In [36]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
Root Mean Squared Error: 4.604032281501152e-14
```

In [37]:
```
import pickle
filename="MUKESH G"
pickle.dump(model,open(filename,'wb'))
model=pickle.load(open(filename,"rb"))
real=[[10,20,30,40],[12,13,21,43]]
result=model.predict(real)
result
```

Out[37]:
```
array([10., 12.])
```

In [ ]: