Data Cleaning

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(r"C:\Users\user\Downloads\22_countries.csv")
df
```

Out[2]:

phone_code	capital	currency	currency_name	currency_symbol	tld	native	region	subı
93	Kabul	AFN	Afghan afghani	؋	.af	افغانستان	Asia	So
+358-18	Mariehamn	EUR	Euro	€	.ax	Åland	Europe	No E
355	Tirana	ALL	Albanian lek	Lek	.al	Shqipëria	Europe	So E
213	Algiers	DZD	Algerian dinar	دج	.dz	الجزائر	Africa	No
+1-684	Pago Pago	USD	US Dollar	$	.as	American Samoa	Oceania	Pol
...	...	...	...	...	...	...	...	
681	Mata Utu	XPF	CFP franc	₣	.wf	Wallis et Futuna	Oceania	Pol
212	El-Aaiun	MAD	Moroccan Dirham	MAD	.eh	الصحراء الغربية	Africa	No
967	Sanaa	YER	Yemeni rial	ريال	.ye	اليَمَن	Asia	W
260	Lusaka	ZMW	Zambian kwacha	ZK	.zm	Zambia	Africa	E
263	Harare	ZWL	Zimbabwe Dollar	$	.zw	Zimbabwe	Africa	E

In [3]:
```
# to display info
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 [4]:
```
# t display summerize the data
df.describe()
```

Out[4]:

	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 [5]:
```
# to display columes
df.columns
```

Out[5]:
```
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')
```

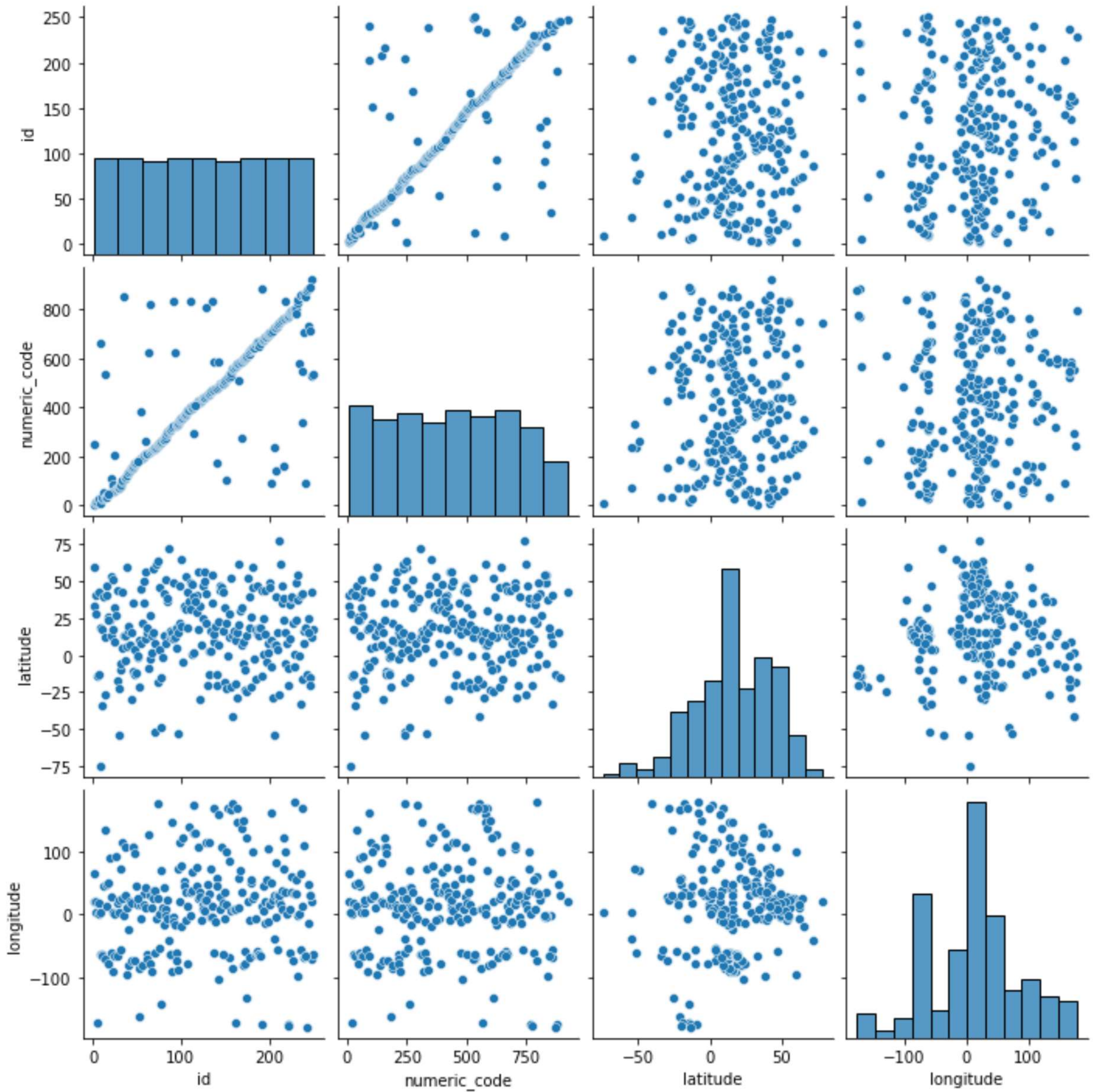
In [6]:
```
df.isna().sum()
```

Out[6]:
```
id                 0
name               0
iso3               0
iso2               1
numeric_code       0
phone_code         0
capital            5
currency           0
currency_name      0
currency_symbol    0
tld                0
native             1
region             2
subregion          3
timezones          0
latitude           0
longitude          0
emoji              0
emojiU             0
dtype: int64
```

EDA and visualization

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

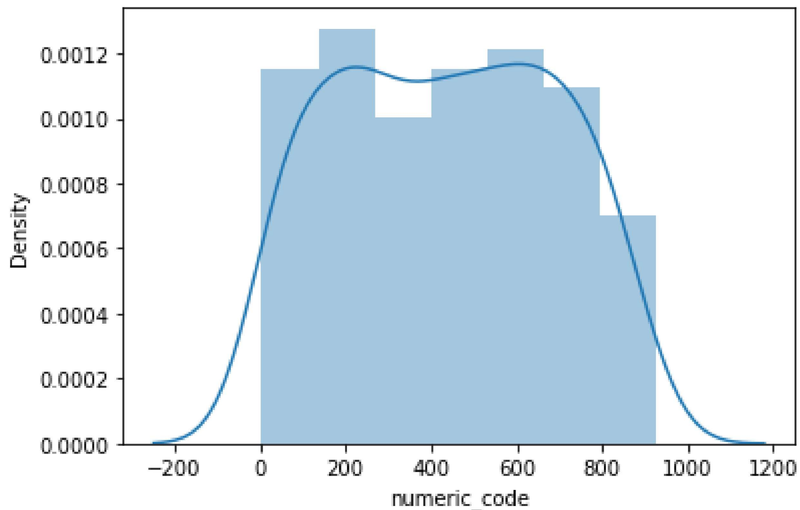
Out[7]: <seaborn.axisgrid.PairGrid at 0x1d41fe3f3d0>



In [9]:
```
# to display distribution graph for price column
sns.distplot(df['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)
```

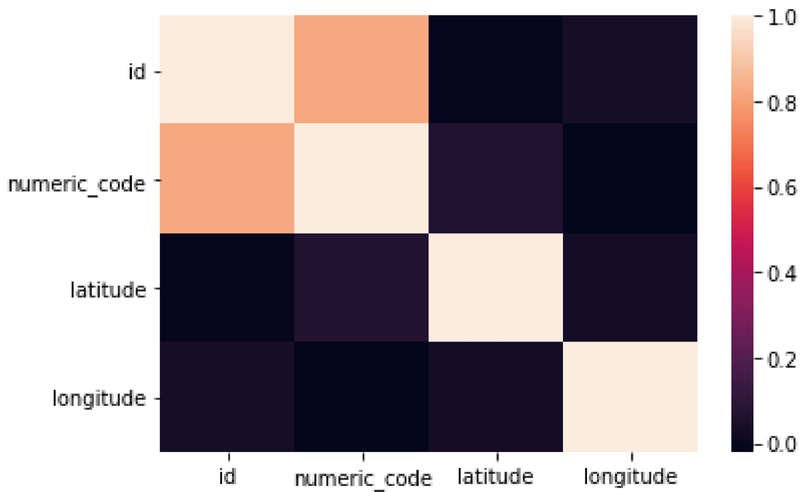
Out[9]: <AxesSubplot:xlabel='numeric_code', ylabel='Density'>



In [10]:
```
df1 = df[['id', 'numeric_code','latitude', 'longitude']]
```

In [11]:
```
# correlation map to find relationship
sns.heatmap(df1.corr())
```

Out[11]: <AxesSubplot:>



In [12]:
```
# Assign x and y for linear regression
x = df1[['id', 'numeric_code','latitude']]
y = df1['longitude']
```

In [13]:
```
# to split 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 [14]:
```
#Linear Regression

from sklearn.linear_model import LinearRegression

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

Out[14]:
```
LinearRegression()
```

In [15]:
```
# intercept is value of c
print(lr.intercept_)
```

```
18.44737019543547
```

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

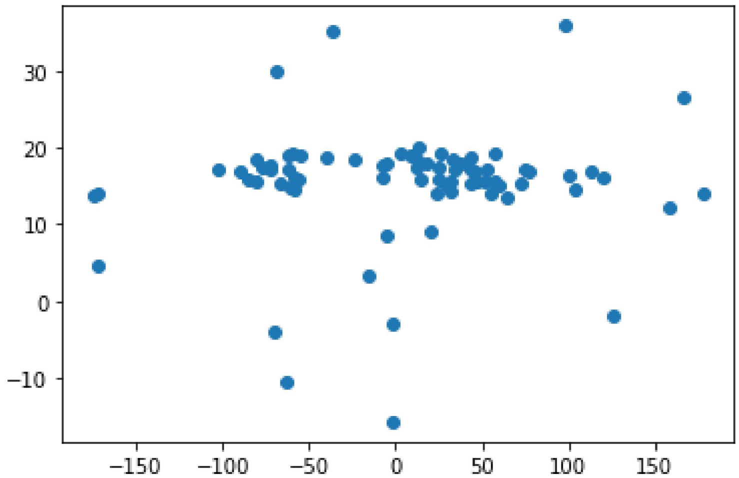
Out[16]:

	Co-efficient
id	0.143539
numeric_code	-0.046333
latitude	0.028612

In [17]:
```
#predict the graph in linear regression graph

prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]: <matplotlib.collections.PathCollection at 0x1d426929b20>



In [18]:
```
#Accuracy of linear regression

print(lr.score(x_test,y_test))
```

-0.0109386297800258

In [19]:
```
lr.score(x_train,y_train)
```

Out[19]: 0.006954979533883909

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

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

Out[21]: -0.010939317258110037

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

Out[22]: 0.006954979521747173

In [23]:
```
lr = Lasso(alpha=10)
lr.fit(x_train,y_train)
lr.score(x_test,y_test)
```

Out[23]: -0.012734744581641477

In [24]:
```
lr.score(x_train,y_train)
```

Out[24]: 0.006901034188739441

Elastic

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

Out[25]:
```
ElasticNet()
```

In [26]:
```
print(es.coef_)
```

```
[ 0.14297874 -0.04618173  0.02779912]
```

In [27]:
```
print(es.intercept_)
```

```
18.46710337701815
```

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

```
-0.011031323827793482
```

Evaluation Model

In [29]:
```
from sklearn import metrics
```

In [30]:
```
print("Mean absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean absolute Error: 58.069559412711335
```

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

```
Mean squared Error: 5550.598305838867
```

In [32]:
```
print("Root Mean squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prec
```

```
Root Mean squared Error: 74.50233758640643
```

model saving

In [33]:
```
import pickle # pickle is used to model saving
```

In [34]:
```
filename ="22_countries prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [38]:
```
df.head(5)
```

Out[38]:

ency	currency_name	currency_symbol	tld	native	region	subregion	
AFN	Afghan afghani	؋	.af	افغانستان	Asia	Southern Asia	[{zoneName:'Asia\/Kabul',(
EUR	Euro	€	.ax	Åland	Europe	Northern Europe	[{zoneName:'Europe\/Marie
ALL	Albanian lek	Lek	.al	Shqipëria	Europe	Southern Europe	[{zoneName:'Europe\/Tirai
DZD	Algerian dinar	دج	.dz	الجزائر	Africa	Northern Africa	[{zoneName:'Africa\/Algi(
USD	US Dollar	$	.as	American Samoa	Oceania	Polynesia	[{zoneName:'Pacific\/Pagc

In [37]:
```
df.tail(5)
```

Out[37]:

ırrency	currency_name	currency_symbol	tld	native	region	subregion	
XPF	CFP franc	₣	.wf	Wallis et Futuna	Oceania	Polynesia	[{zoneName:'Pacific\
MAD	Moroccan Dirham	MAD	.eh	الصحراء الغربية	Africa	Northern Africa	[{zoneName:'Africa\/E
YER	Yemeni rial	﷼	.ye	اليَمَن	Asia	Western Asia	[{zoneName:'Asia\/Ade
ZMW	Zambian kwacha	ZK	.zm	Zambia	Africa	Eastern Africa	[{zoneName:'Africa\/Lu
ZWL	Zimbabwe Dollar	$	.zw	Zimbabwe	Africa	Eastern Africa	[{zoneName:'Africa\/H

In [ ]: