In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:

```
df = pd.read_csv("countries.csv").dropna()
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	

243 rows × 19 columns

In [3]:

```
df.head()
```

Out[3]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	curren
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan afghani	
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	Euro	

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	currenc
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	

Data cleaning and pre processing

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 243 entries, 0 to 249
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               243 non-null    int64
 1   name             243 non-null    object
 2   iso3             243 non-null    object
 3   iso2             243 non-null    object
 4   numeric_code     243 non-null    int64
 5   phone_code       243 non-null    object
 6   capital          243 non-null    object
 7   currency         243 non-null    object
 8   currency_name    243 non-null    object
 9   currency_symbol  243 non-null    object
 10  tld              243 non-null    object
 11  native           243 non-null    object
 12  region           243 non-null    object
 13  subregion        243 non-null    object
 14  timezones        243 non-null    object
 15  latitude         243 non-null    float64
 16  longitude        243 non-null    float64
 17  emoji            243 non-null    object
 18  emojiU           243 non-null    object
dtypes: float64(2), int64(2), object(15)
memory usage: 38.0+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	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

	id	numeric_code	latitude	longitude
max	250.000000	926.000000	78.000000	178.000000

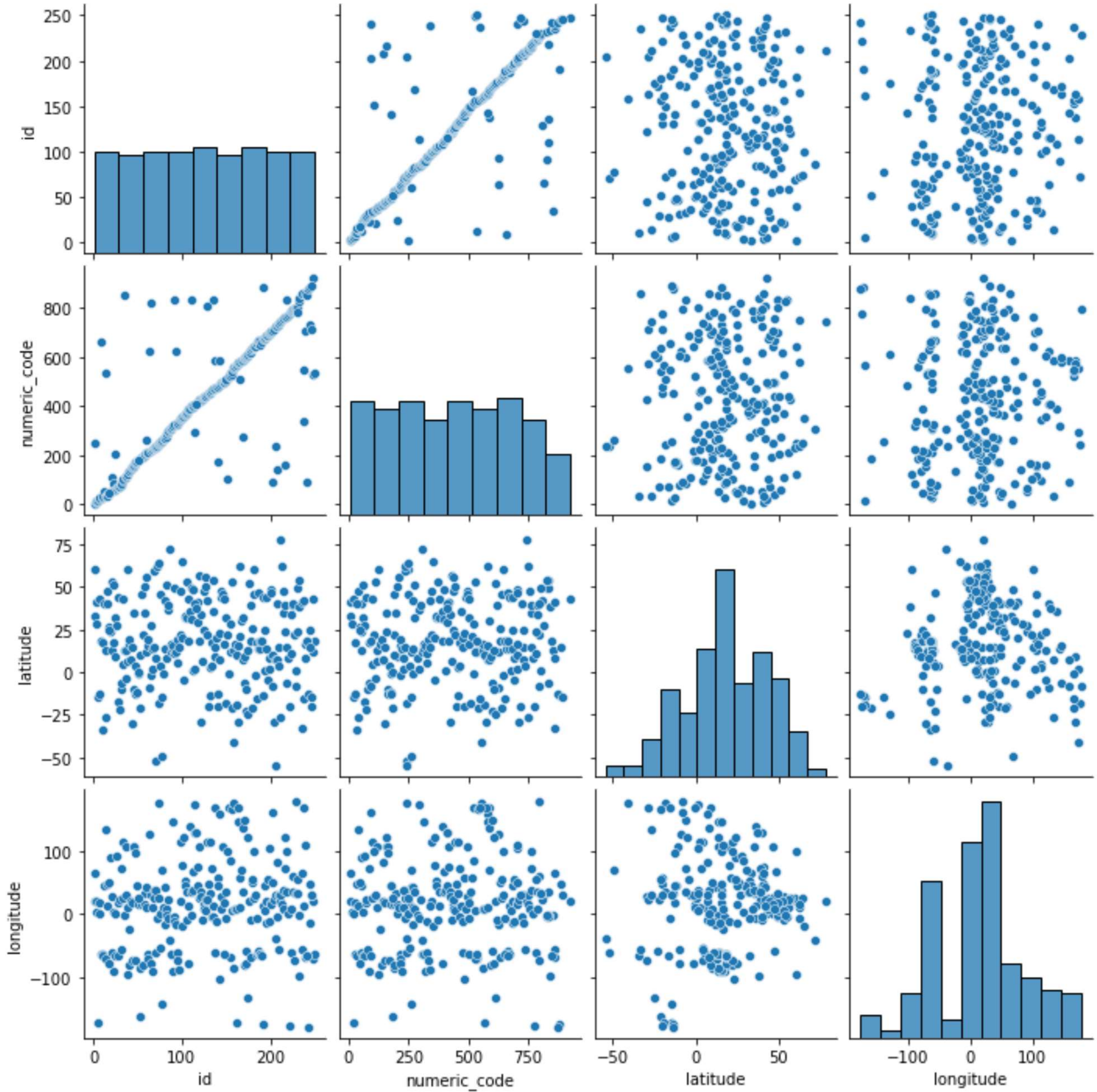
In [6]:
```
df.columns
```

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 0x200c0424fa0>
```

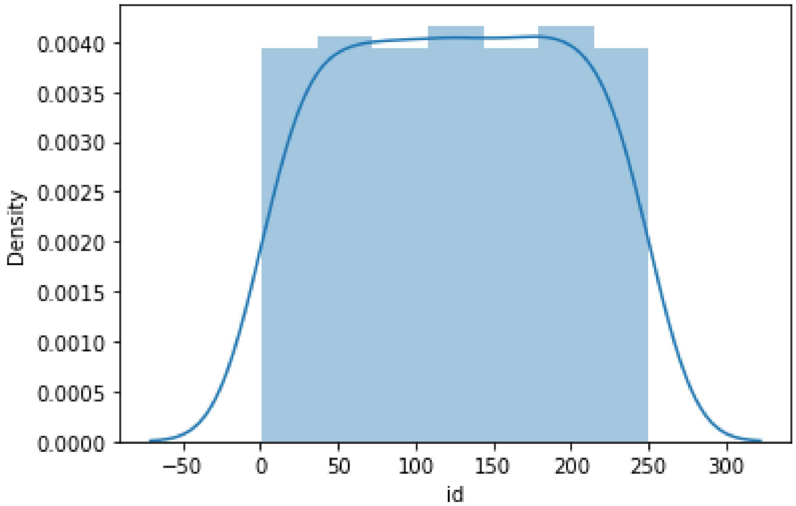


In [8]:

```
sns.distplot(df['id'])
```

```
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)
```

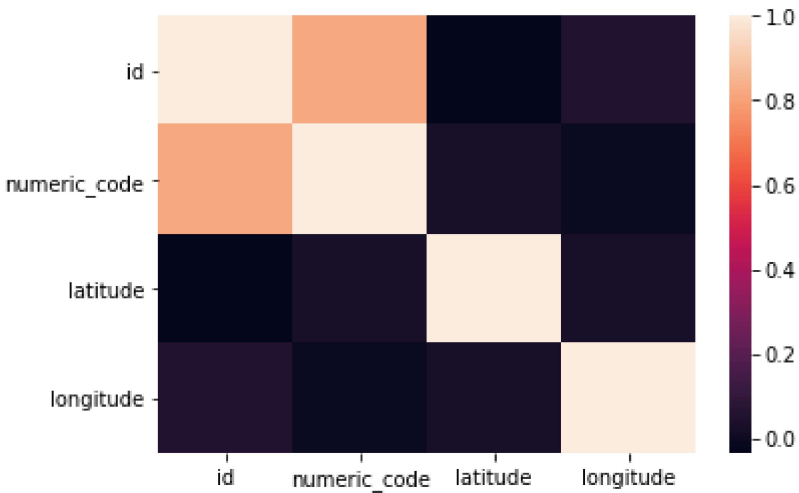
Out[8]: <AxesSubplot:xlabel='id', ylabel='Density'>



In [9]:
```
df1 = df[['id', 'name', 'iso3', 'iso2', 'numeric_code', 'phone_code', 'capital',
        'currency', 'currency_name', 'currency_symbol', 'tld', 'native',
        'region', 'subregion', 'timezones', 'latitude', 'longitude', 'emoji',
        'emojiU']]
```

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

Out[10]: <AxesSubplot:>



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

split the data into training and test data

In [12]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

In [13]:
```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[13]: LinearRegression()

In [14]:
```
lr.intercept_
```

Out[14]: 7.379600305767285

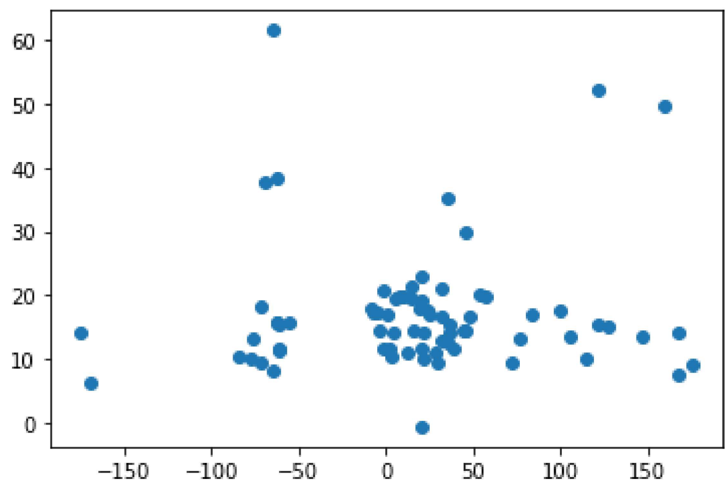
In [15]:
```
coeff = pd.DataFrame(lr.coef_, x.columns, columns =['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
id	0.239106
numeric_code	-0.057097
latitude	0.095792

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

Out[16]: <matplotlib.collections.PathCollection at 0x200c16eb130>



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

Out[17]: -0.0034708676690566875

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [19]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

Out[19]: 0.016388192152411607

In [20]:
```
rr.score(x_test,y_test)
```

Out[20]: -0.0034700016572182246

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

Out[21]: Lasso(alpha=10)

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

Out[22]: -0.0029147892727037217

In [23]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[23]: ElasticNet()

In [24]:
```
print(en.coef_)
```

```
[ 0.23855392 -0.05696216  0.0948655 ]
```

In [25]:
```
print(en.intercept_)
```

```
7.4022414128331935
```

In [26]:
```
print(en.predict(x_test))
```

```
[15.23003287 15.51088006 11.00120033 11.72610536 16.55622445 14.39460829
 10.2330479  10.32914509  9.50623972 19.82974032 22.75677837 29.85252518
 52.15920578 18.07733216 14.23007165 11.05998291 -0.54426958 38.33366774
 13.56239546 18.05693158 13.68755286 20.12663336 14.26126345 12.85403907
 17.05623959  8.35705673 14.38033467 12.51725543 11.76153997 20.81874139
 16.94760581  6.3238776  10.28536757 11.5516913  10.06220983 37.70471284
 16.89608686 19.84806373 15.72602622 19.56813521 14.5292469   9.32425524
 19.5210697  61.40171967 35.07946863 16.74549599  7.67201491 21.26034093
 14.29043024 11.83166871 15.38136817 15.64098671 14.03769191 17.54944008
 15.81613733 11.34056181 14.43681236 17.27354145 17.13063848 18.27542435
  9.56802788 17.70915166  9.54872391 14.28287604 19.72116908 10.11086177
 13.31826833 21.12047364 15.43884911 11.55603791 19.23204503 49.70461477
 13.11496738]
```

In [27]:
```
print(en.score(x_test,y_test))
```

```
-0.0034325515483795144
```

Evaluation Metrics

In [28]:
```
from sklearn import metrics
```

In [29]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 51.84693582011113
```

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

```
Mean Squared Error: 5069.305121217806
```

In [31]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 71.19905281124045
```

In [32]:
```
import pickle
```

In [33]:
```
filename='prediction'
pickle.dump(lr,open(filename,'wb'))
```

In [34]:
```
import pandas as pd
import pickle
```

In [35]:
```
filename='prediction'
model = pickle.load(open(filename,'rb'))
```

In [36]:
```
real = [[10,20,30],[11,45,55]]
result = model.predict(real)
```

In [37]:
```
result
```

Out[37]:
```
array([11.50247039, 12.70894242])
```