31-07-2023

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

In [471]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\22_countries.csv")
a
```

Out[471]:

	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	
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 [472]:
```
a=a.head(10)
a
```

Out[472]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	currency_s
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 [473]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               10 non-null     int64  
 1   name             10 non-null     object 
 2   iso3             10 non-null     object 
 3   iso2             10 non-null     object 
 4   numeric_code     10 non-null     int64  
 5   phone_code       10 non-null     object 
 6   capital          9 non-null      object 
 7   currency         10 non-null     object 
 8   currency_name    10 non-null     object 
 9   currency_symbol  10 non-null     object 
 10  tld              10 non-null     object 
 11  native           10 non-null     object 
 12  region           10 non-null     object 
 13  subregion        9 non-null      object 
 14  timezones        10 non-null     object 
 15  latitude         10 non-null     float64
 16  longitude        10 non-null     float64
 17  emoji            10 non-null     object 
 18  emojiU           10 non-null     object 
dtypes: float64(2), int64(2), object(15)
memory usage: 1.6+ KB
```

In [474]:
```
a.columns
```

Out[474]:
```
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 [478]:
```
d=a[['id', 'name', 'iso3', 'iso2', 'numeric_code', 'phone_code', 'capital',
       'currency', 'currency_name', 'currency_symbol', 'tld', 'native',
       'region', 'subregion', 'timezones', 'latitude', 'longitude', 'emoji',
       'emojiU']]
d
```

Out[478]:

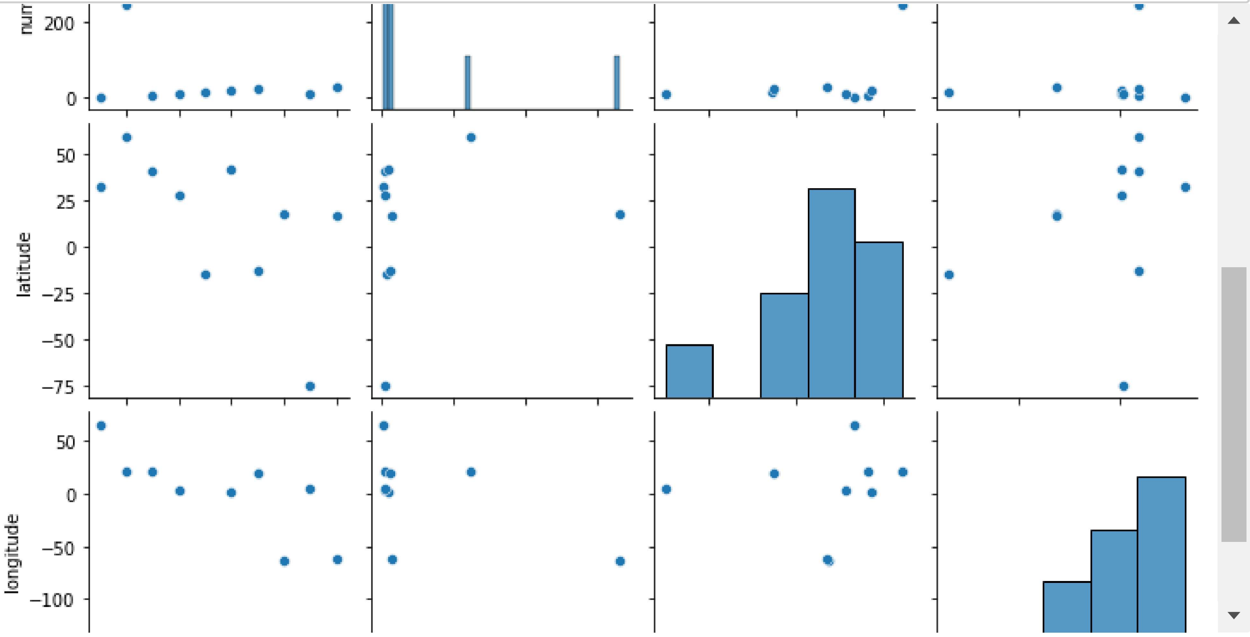
	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	currency_s
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan afghani	
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	Euro	
2	3	Albania	ALB	AL	8	355	Tirana	ALL	Albanian lek	
3	4	Algeria	DZA	DZ	12	213	Algiers	DZD	Algerian dinar	
4	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 [479]:
```
d.describe()
```

Out[479]:

	id	numeric_code	latitude	longitude
count	10.00000	10.0000	10.000000	10.000000
mean	5.50000	103.0000	13.843333	-16.258667
std	3.02765	209.0598	38.895825	66.215478
min	1.00000	4.0000	-74.650000	-170.000000
25%	3.25000	10.5000	-5.112500	-45.975000
50%	5.50000	18.0000	23.125000	3.740000
75%	7.75000	27.0000	39.000000	19.550000
max	10.00000	660.0000	60.116667	65.000000

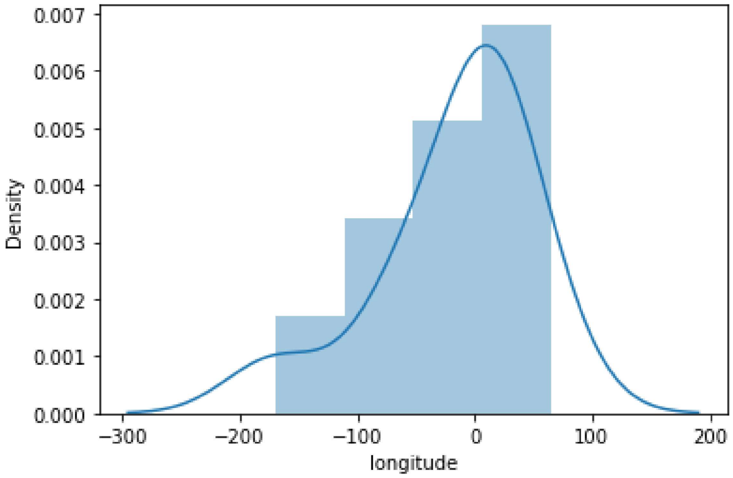
In [480]:
```
sns.pairplot(d)
```



In [481]:
```
sns.distplot(a['longitude'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `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 flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

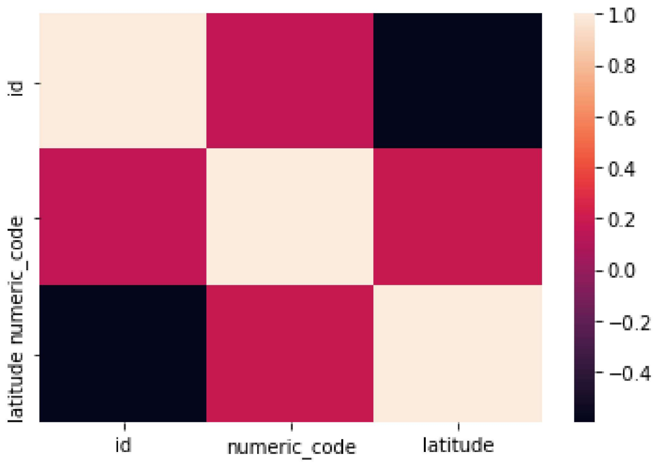
Out[481]:
```
<AxesSubplot:xlabel='longitude', ylabel='Density'>
```



In [482]:
```
x1=a[['id','numeric_code','latitude']]
```

In [483]:
```
sns.heatmap(x1.corr())
```

Out[483]:
```
<AxesSubplot:>
```



In [484]:
```
x=a[['id','numeric_code','latitude']]
y=a['longitude']
```

In [485]:
```
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 [486]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[486]:
```
LinearRegression()
```

In [487]:
```
print(lr.intercept_)
```

```
-67.29381514741908
```

In [488]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[488]:

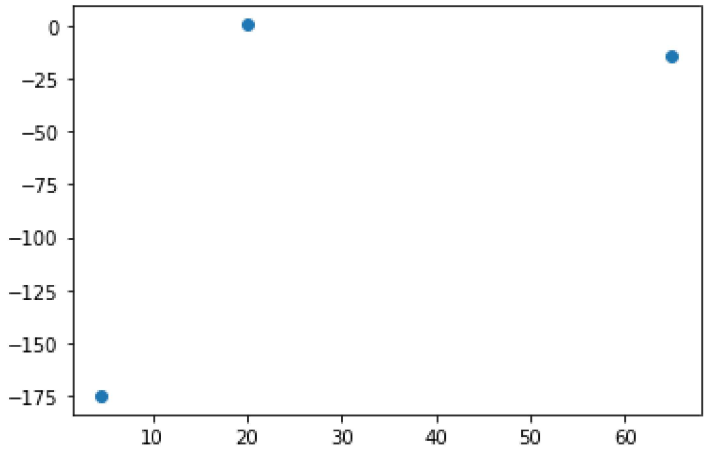
	Co-efficient
id	1.187099
numeric_code	-0.049721
latitude	1.576395

In [489]:

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

Out[489]:

```
<matplotlib.collections.PathCollection at 0x190c7a39be0>
```



In [490]:

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

```
-18.634930589621092
```

In [491]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [492]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[492]:

```
Ridge(alpha=10)
```

In [493]:

```
rr.score(x_test,y_test)
```

Out[493]:

```
-18.411824611886356
```

In [494]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[494]:

```
Lasso(alpha=10)
```

In [495]:

```
la.score(x_test,y_test)
```

Out[495]:

```
-17.735451690334546
```

In [496]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[496]:

```
ElasticNet()
```

In [497]:

```
print(en.coef_)
```

```
[ 0.95999386 -0.04914654  1.56357728]
```

```
In [498]: print(en.intercept_)

          -65.75917636168431

In [499]: print(en.predict(x_test))

          [   0.83430117 -174.33174064  -13.39771858]

In [500]: en.score(x_test,y_test)

Out[500]: -18.47553344683703

In [501]: from sklearn import metrics

In [502]: print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))

          Mean Absolute Error 92.68255502677836

In [503]: print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))

          Mean Squared Error 12934.054122859634

In [504]: print(" Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction))

           Root Mean Squared Error 113.72798302467002

In [505]: import pickle

In [506]: filename="prediction"
          pickle.dump(lr,open(filename,'wb'))

In [507]: import pandas as pd
          import pickle

In [508]: filename="prediction"
          model=pickle.load(open(filename,"rb"))

In [509]: real=[[10,20,24,],[15,30,36]]
          result=model.predict(real)

In [510]: result

Out[510]: array([-18.58375523,   5.77127472])
```