Problem Statement

Linear Regression

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
a=pd.read_csv("countries.csv")
a
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...
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

To display top 10 rows

In [3]:
```
c=a.head(15)
c
```

Out[3]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	c
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	
10	11	Argentina	ARG	AR	32	54	Buenos Aires	ARS	Argentine peso	
11	12	Armenia	ARM	AM	51	374	Yerevan	AMD	Armenian dram	
12	13	Aruba	ABW	AW	533	297	Oranjestad	AWG	Aruban florin	
13	14	Australia	AUS	AU	36	61	Canberra	AUD	Australian dollar	
14	15	Austria	AUT	AT	40	43	Vienna	EUR	Euro	

To find Missing values

In [4]:
```
c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               15 non-null     int64
 1   name             15 non-null     object
```

```
 2   iso3             15 non-null     object
 3   iso2             15 non-null     object
 4   numeric_code     15 non-null     int64
 5   phone_code       15 non-null     object
 6   capital          14 non-null     object
 7   currency         15 non-null     object
 8   currency_name    15 non-null     object
 9   currency_symbol  15 non-null     object
 10  tld              15 non-null     object
 11  native           15 non-null     object
 12  region           15 non-null     object
 13  subregion        14 non-null     object
 14  timezones        15 non-null     object
 15  latitude         15 non-null     float64
 16  longitude        15 non-null     float64
 17  emoji            15 non-null     object
 18  emojiU           15 non-null     object
dtypes: float64(2), int64(2), object(15)
memory usage: 2.4+ KB
```

To display summary of statistics

In [5]:
```
a.describe()
```

Out[5]:

	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

To display column heading

In [6]:
```
a.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')
```

Pairplot

In [7]:
```
s=a.dropna(axis=1)
s
```

Out[7]:

	id	name	iso3	numeric_code	phone_code	currency	currency_name	currency_symbol
0	1	Afghanistan	AFG	4	93	AFN	Afghan afghani	؋
1	2	Aland Islands	ALA	248	+358-18	EUR	Euro	€
2	3	Albania	ALB	8	355	ALL	Albanian lek	Lek
3	4	Algeria	DZA	12	213	DZD	Algerian dinar	دج
4	5	American Samoa	ASM	16	+1-684	USD	US Dollar	$
...	...	...	...	...	...	...	...	...
245	243	Wallis And Futuna Islands	WLF	876	681	XPF	CFP franc	₣
246	244	Western Sahara	ESH	732	212	MAD	Moroccan Dirham	MAD
247	245	Yemen	YEM	887	967	YER	Yemeni rial	﷼
248	246	Zambia	ZMB	894	260	ZMW	Zambian kwacha	ZK
249	247	Zimbabwe	ZWE	716	263	ZWL	Zimbabwe Dollar	$

250 rows × 14 columns

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

Out[8]:
```
Index(['id', 'name', 'iso3', 'numeric_code', 'phone_code', 'currency',
       'currency_name', 'currency_symbol', 'tld', 'timezones', 'latitude',
       'longitude', 'emoji', 'emojiU'],
      dtype='object')
```

To train the Model

In [9]:
```
g=c[['id','numeric_code','latitude']]
h=c['longitude']
```

To split dataset into training end test

In [10]:
```
from sklearn.model_selection import train_test_split
g_train,g_test,h_train,h_test=train_test_split(g,h,test_size=0.6)
```

To run the model

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

In [12]:
```
lr=LinearRegression()
lr.fit(g_train,h_train)
```

Out[12]:
```
LinearRegression()
```

In [13]:
```
print(lr.intercept_)
```

```
-84.4739638818785
```

Coeffecient

In [14]:
```
coeff=pd.DataFrame(lr.coef_,g.columns,columns=['Co-effecient'])
coeff
```

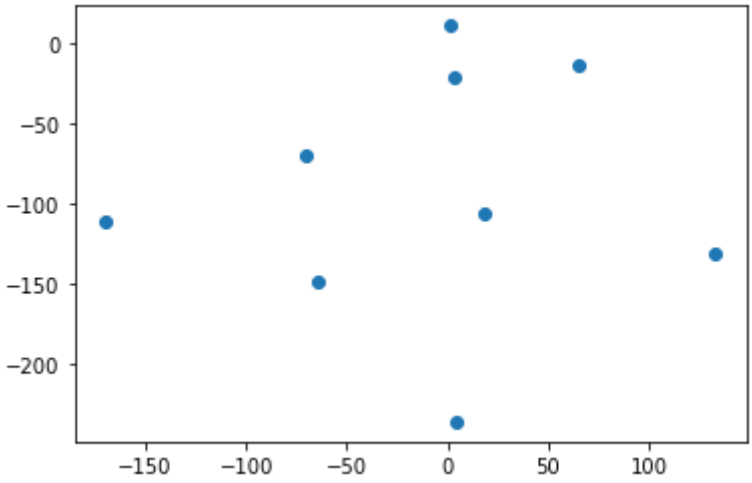
Out[14]:

	Co-effecient
id	0.961240
numeric_code	-0.045767
latitude	2.142645

Best Fit line

In [15]:
```
prediction=lr.predict(g_test)
plt.scatter(h_test,prediction)
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x1a477aedaf0>
```



To find score

In [16]:
```
print(lr.score(g_test,h_test))
```

```
-1.6912556515496209
```

Import Lasso and ridge

In [17]:
```
from sklearn.linear_model import Ridge,Lasso
```

Ridge

In [18]:
```
ri=Ridge(alpha=5)
ri.fit(g_train,h_train)
```

Out[18]:
```
Ridge(alpha=5)
```

In [19]:
```
ri.score(g_test,h_test)
```

Out[19]:
```
-1.6656431383033157
```

In [20]:
```
ri.score(g_train,h_train)
```

Out[20]:
```
0.7757877598651237
```

Lasso

In [21]:
```
l=Lasso(alpha=6)
l.fit(g_train,h_train)
```

Out[21]:
```
Lasso(alpha=6)
```

In [22]:
```
l.score(g_test,h_test)
```

Out[22]:
```
-1.573145737225631
```

In [23]:
```
ri.score(g_train,h_train)
```

Out[23]:
```
0.7757877598651237
```

ElasticNet

In [24]:
```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(g_train,h_train)
```

Out[24]:
```
ElasticNet()
```

Coeffecient,intercept

In [25]:
```
print(e.coef_)
```

```
[ 0.89862584 -0.04632288  2.12821823]
```

In [26]:
```
print(e.intercept_)
```

```
-83.31821224740827
```

Prediction

In [27]:
```
d=e.predict(g_test)
d
```

Out[27]:
```
array([-104.74230846, -234.56529971,  -12.3736762 ,   11.59636011,
       -129.86696657,  -69.72344428,  -20.68947291, -110.07071051,
       -147.2750802 ])
```

In [28]:
```
print(e.score(g_test,h_test))
```

```
-1.6659043661665063
```

Evaluation

In [29]:
```
from sklearn import metrics
print("Mean Absolute error:",metrics.mean_absolute_error(h_test,d))
```

```
Mean Absolute error: 97.75129733678523
```

In [30]:
```
print("Mean Squared error:",metrics.mean_squared_error(h_test,d))
```

```
Mean Squared error: 17622.94392019556
```

In [31]:
```
print("Mean Squared error:",np.sqrt(metrics.mean_squared_error(h_test,d)))
```

```
Mean Squared error: 132.7514366031327
```

Model Saving

In [32]:
```
import pickle
filename="pre"
pickle.dump(lr,open(filename,"wb"))
```

In [33]:
```
filename='pre'
model = pickle.load(open(filename,'rb'))
```

In [35]:
```
eral=[[15,10,65],[19,54,30]]
result=model.predict(eral)
result
```

Out[35]: array([68.75889952, -4.40244242])