In [10]:

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

19. DataSet Nuclear Explosion

In [3]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\19_nuclear_explosions - 19_nuclear_explosions.csv
a
```

Out[3]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinate
0	USA	Alamogordo	DOE	32.54	
1	USA	Hiroshima	DOE	34.23	
2	USA	Nagasaki	DOE	32.45	
3	USA	Bikini	DOE	11.35	
4	USA	Bikini	DOE	11.35	
...	...	...	...	...	
2041	CHINA	Lop Nor	HFS	41.69	
2042	INDIA	Pokhran	HFS	27.07	
2043	INDIA	Pokhran	NRD	27.07	
2044	PAKIST	Chagai	HFS	28.90	
2045	PAKIST	Kharan	HFS	28.49	

2046 rows × 16 columns

In [6]:

```
b=a.head(10)
b
```

Out[6]:

Data.Magnitude.Body	Data.Magnitude.Surface	Location.Cordinates.Depth	Data.Yeild.Lower	Dat
0.0	0.0	-0.10	21.0	
0.0	0.0	-0.60	15.0	
0.0	0.0	-0.60	21.0	
0.0	0.0	-0.20	21.0	
0.0	0.0	0.03	21.0	
0.0	0.0	-0.08	37.0	
0.0	0.0	-0.08	49.0	
0.0	0.0	-0.08	18.0	
0.0	0.0	0.00	22.0	
0.0	0.0	-0.35	1.0	

In [7]:

```
a.columns
```

Out[7]:

```
Index(['WEAPON SOURCE COUNTRY', 'WEAPON DEPLOYMENT LOCATION', 'Data.Sourc
e',
       'Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Uppe
r',
       'Data.Purpose', 'Data.Name', 'Data.Type', 'Date.Day', 'Date.Month',
       'Date.Year'],
      dtype='object')
```

In [8]:

```
c=b[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper']]
c
```

Out[8]:

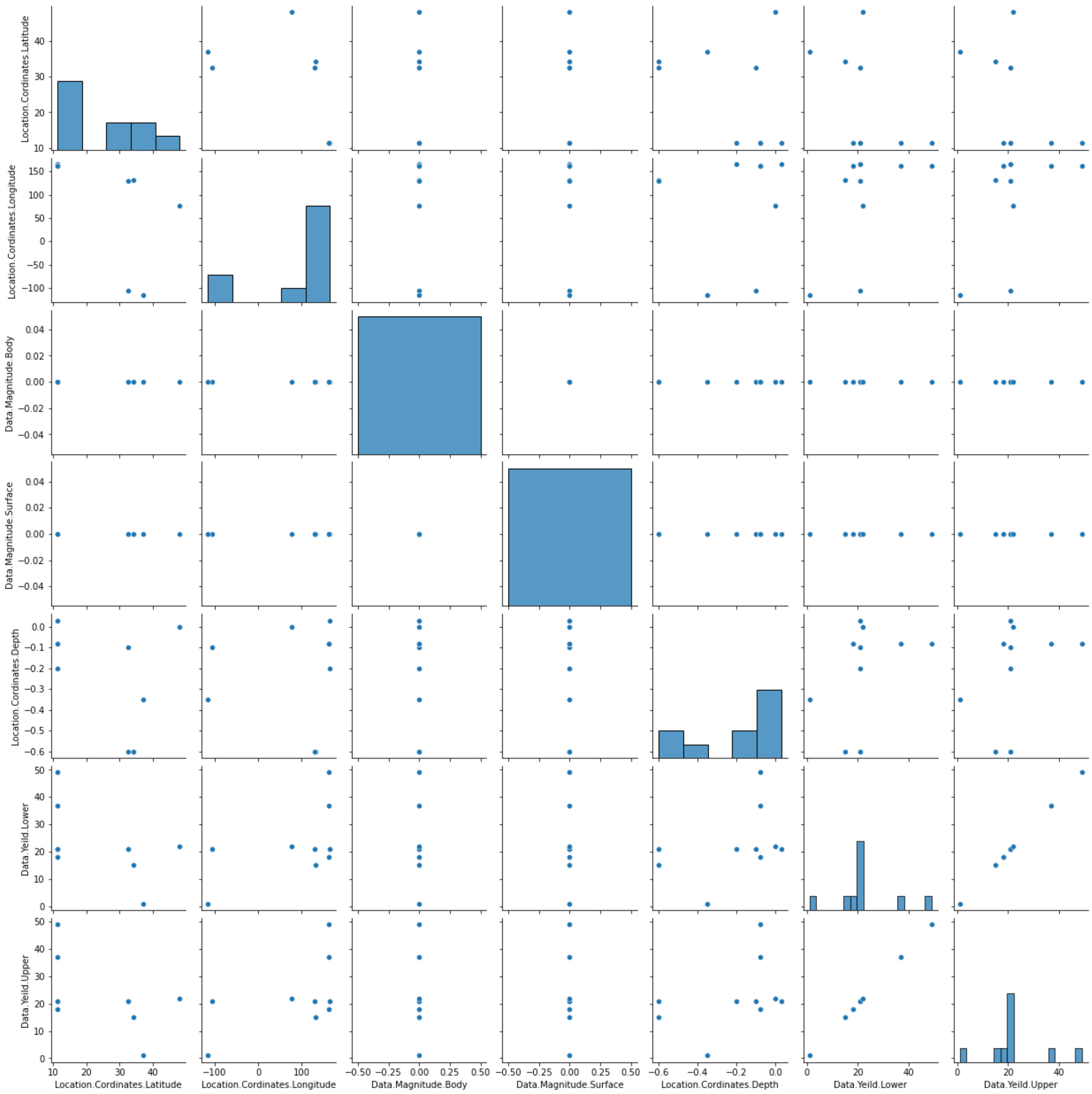
	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Mag
0	32.54	-105.57	0.0	
1	34.23	132.27	0.0	
2	32.45	129.52	0.0	
3	11.35	165.20	0.0	
4	11.35	165.20	0.0	
5	11.30	162.15	0.0	
6	11.30	162.15	0.0	
7	11.30	162.15	0.0	
8	48.00	76.00	0.0	
9	37.00	-116.00	0.0	

In [11]:

```
sns.pairplot(c)
```

Out[11]:

```
<seaborn.axisgrid.PairGrid at 0x228ae5ce8e0>
```



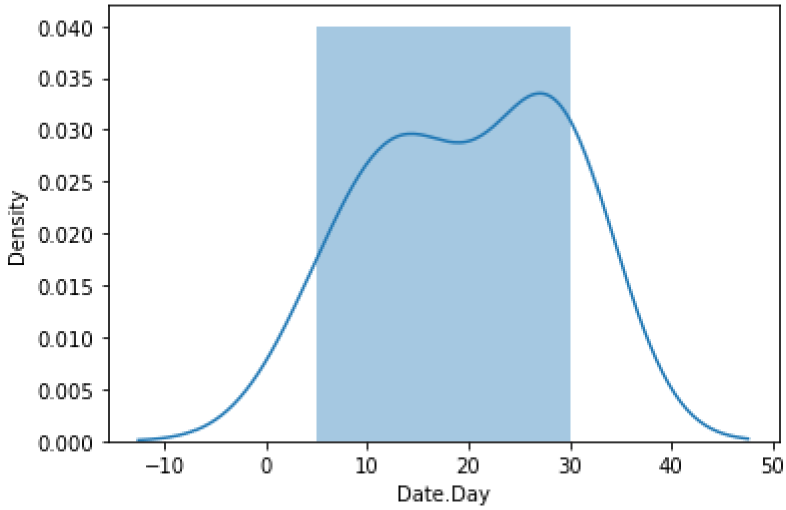
In [12]:

```
sns.distplot(b['Date.Day'])
```

```
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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]:

```
<AxesSubplot:xlabel='Date.Day', ylabel='Density'>
```

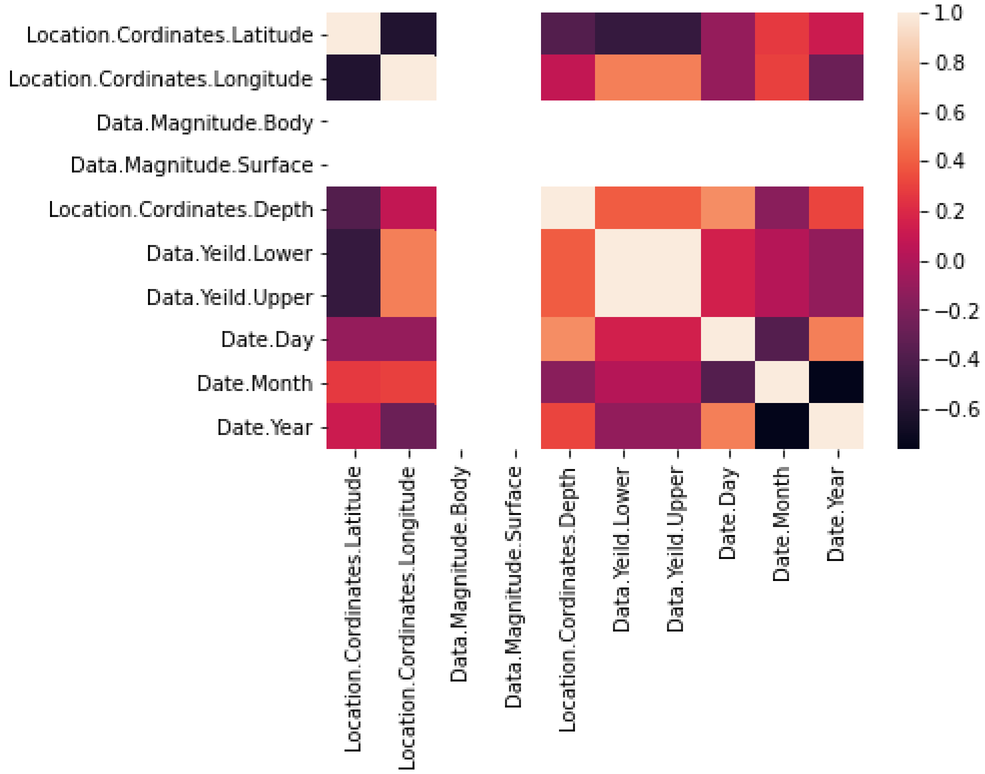


In [13]:

```
sns.heatmap(b.corr())
```

Out[13]:

```
<AxesSubplot:>
```



In [14]:

```
x=c[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower']]
y=b['Date.Day']
```

In [15]:

```
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 [16]:

```
from sklearn.linear_model import LinearRegression

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

Out[16]:

```
LinearRegression()
```

In [17]:

```
print(lr.intercept_)
```

```
19.413510754216745
```

In [18]:

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

Out[18]:

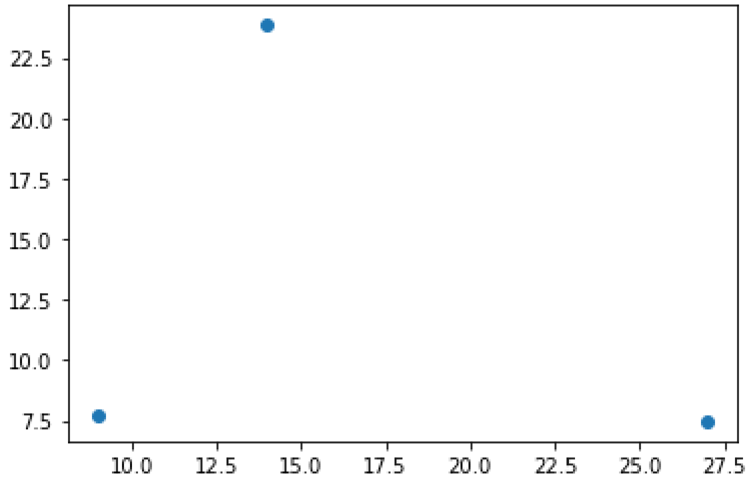
	Co-efficient
Location.Cordinates.Latitude	9.461816e-02
Location.Cordinates.Longitude	3.387283e-02
Data.Magnitude.Body	-2.593481e-13
Data.Magnitude.Surface	0.000000e+00
Location.Cordinates.Depth	3.297616e+01
Data.Yeild.Lower	2.833362e-02

In [19]:

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

Out[19]:

```
<matplotlib.collections.PathCollection at 0x228b0dbf670>
```



In [20]:

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

```
-1.7802507381804187
```

In [21]:

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

In [22]:

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

Out[22]:

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

In [23]:

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

Out[23]:

```
-1.1394700355681717
```

In [24]:

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

Out[24]:

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

In [25]:

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

Out[25]:

```
-0.9869375588232192
```

In [26]:

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

Out[26]:

```
ElasticNet()
```

In [27]:

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

```
[0.00152732 0.01198291 0.         0.         0.96649771 0.23611682]
```

In [28]:

```
print(en.intercept_)
```

```
13.679633247231063
```

In [29]:

```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
-1.1148190897180599
```

In [30]:

```
from sklearn import metrics
```

In [31]:

```
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 10.409502987773765
```

In [32]:

```
print("mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean Squared Error: 121.71958760821724
```

In [33]:

```
print("Root mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root mean Squared Error: 11.032660042266201
```

In [34]:

```
import pickle
```

In [35]:

```
filename19='ex'
pickle.dump(lr,open(filename19,'wb'))
```

In [63]:

```
filename19='ex'
model=pickle.load(open(filename19,'rb'))
```

In [64]:

```
real=[[10,20,30,40,45,60],[24,21,26,34,78,11]]
result=model.predict(real)
```

In [65]:

```
result
```

Out[65]:

```
array([1506.6642781 , 2594.84767294])
```

20. DataSet States

In [36]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\20_states - 20_states.csv")
a
```

Out[36]:

	id	name	country_id	country_code	country_name	state_code	type	latitu
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.7347
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.1671
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.1789
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.7550
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.8100
...	...	...	...	...	...	...	...	
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	NaN	-17.4851
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	NaN	-20.6241
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	NaN	-18.5331
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	NaN	-21.0523
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	NaN	-19.0552

5077 rows × 9 columns

In [37]:

```
b=a.fillna(value=50)
b
```

Out[37]:

	id	name	country_id	country_code	country_name	state_code	type	latitu
0	3901	Badakhshan	1	AF	Afghanistan	BDS	50	36.7347
1	3871	Badghis	1	AF	Afghanistan	BDG	50	35.1671
2	3875	Baghlan	1	AF	Afghanistan	BGL	50	36.1789
3	3884	Balkh	1	AF	Afghanistan	BAL	50	36.7550
4	3872	Bamyan	1	AF	Afghanistan	BAM	50	34.8100
...	...	...	...	...	...	...	...	
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	50	-17.4851
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	50	-20.6241
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	50	-18.5331
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	50	-21.0523
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	50	-19.0552

5077 rows × 9 columns

In [38]:

```
c=b.head(10)
c
```

Out[38]:

	id	name	country_id	country_code	country_name	state_code	type	latitude
0	3901	Badakhshan	1	AF	Afghanistan	BDS	50	36.734772
1	3871	Badghis	1	AF	Afghanistan	BDG	50	35.167134
2	3875	Baghlan	1	AF	Afghanistan	BGL	50	36.178903
3	3884	Balkh	1	AF	Afghanistan	BAL	50	36.755060
4	3872	Bamyan	1	AF	Afghanistan	BAM	50	34.810007
5	3892	Daykundi	1	AF	Afghanistan	DAY	50	33.669495
6	3899	Farah	1	AF	Afghanistan	FRA	50	32.495328
7	3889	Faryab	1	AF	Afghanistan	FYB	50	36.079561
8	3870	Ghazni	1	AF	Afghanistan	GHA	50	33.545059
9	3888	Ghōr	1	AF	Afghanistan	GHO	50	34.099578

In [39]:

```
c.columns
```

Out[39]:

```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

In [40]:

```
d=c[['id','country_id','type', 'latitude', 'longitude']]
d
```

Out[40]:

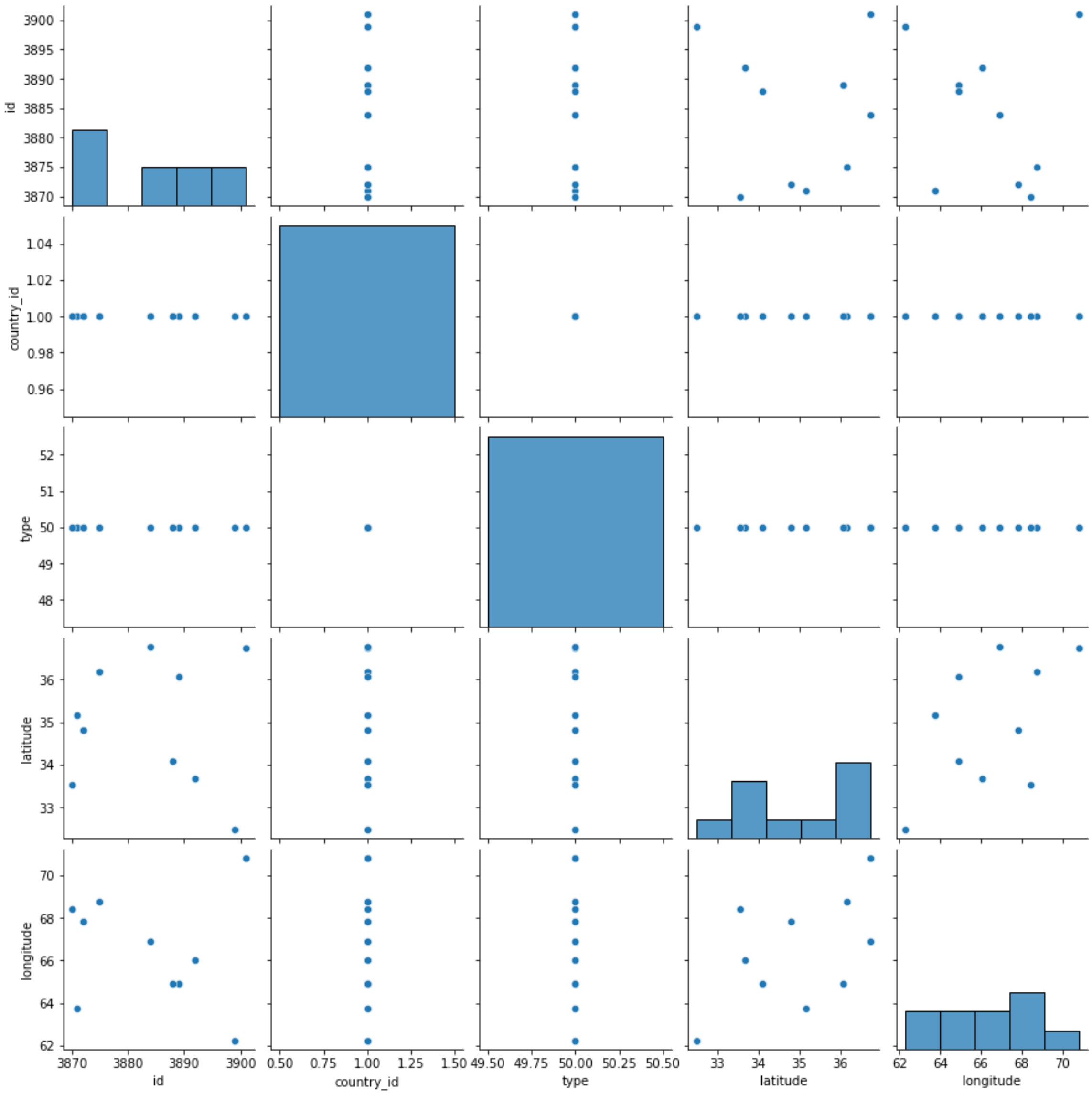
	id	country_id	type	latitude	longitude
0	3901	1	50	36.734772	70.811995
1	3871	1	50	35.167134	63.769538
2	3875	1	50	36.178903	68.745306
3	3884	1	50	36.755060	66.897537
4	3872	1	50	34.810007	67.821210
5	3892	1	50	33.669495	66.046353
6	3899	1	50	32.495328	62.262663
7	3889	1	50	36.079561	64.905955
8	3870	1	50	33.545059	68.417397
9	3888	1	50	34.099578	64.905955

In [41]:

```
sns.pairplot(d)
```

Out[41]:

```
<seaborn.axisgrid.PairGrid at 0x228b1bfadc0>
```



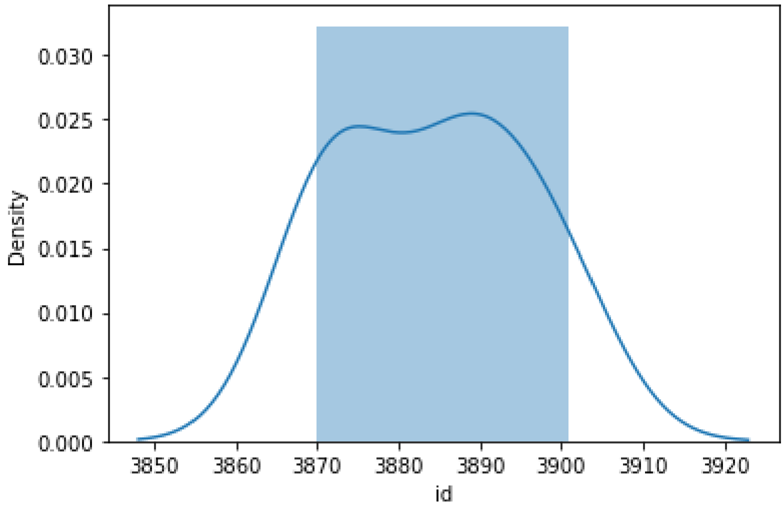
In [42]:

```
sns.distplot(d['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 adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[42]:

<AxesSubplot:xlabel='id', ylabel='Density'>

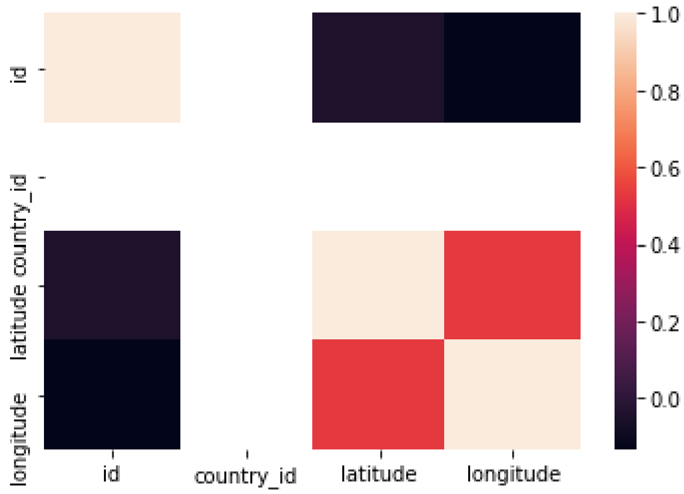


In [43]:

```
sns.heatmap(d.corr())
```

Out[43]:

<AxesSubplot:>



In [44]:

```
x=c[['id','country_id','type', 'latitude', 'longitude']]
y=c['longitude']
```

In [45]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [46]:

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

Out[46]:

```
LinearRegression()
```

In [47]:

```
print(lr.intercept_)
```

```
-2.842170943040401e-14
```

In [48]:

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

Out[48]:

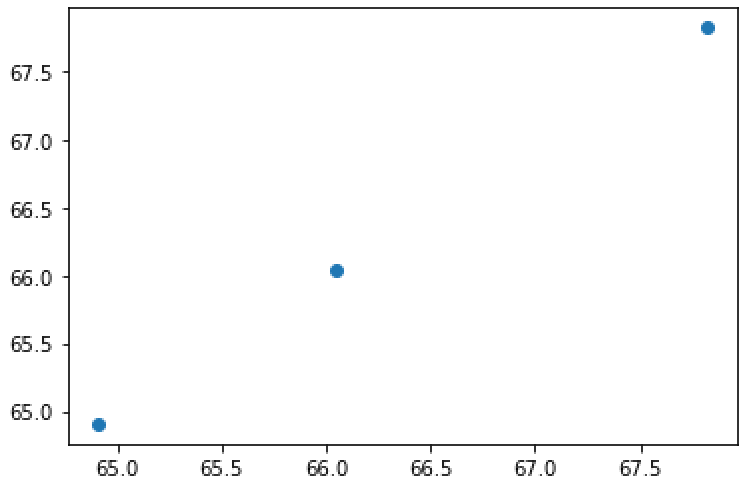
	Co-efficient
id	0.000000e+00
country_id	1.554312e-15
type	0.000000e+00
latitude	1.278645e-16
longitude	1.000000e+00

In [49]:

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

Out[49]:

```
<matplotlib.collections.PathCollection at 0x228b48dafa0>
```



In [50]:

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

```
1.0
```

In [51]:

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

Out[51]:

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

In [52]:

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

Out[52]:

```
0.9296189023867054
```

In [53]:

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

Out[53]:

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

In [54]:

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

Out[54]:

```
-0.05704934034014841
```

In [55]:

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

Out[55]:

```
ElasticNet()
```

In [56]:

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

```
[-0.          0.          0.          0.          0.88116825]
```

In [57]:

```
print(en.intercept_)
```

```
7.907580516118436
```

In [58]:

```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9850734225720338
```

In [59]:

```
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.1352005745886847
```

In [60]:

```
print("mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean Squared Error: 0.021476591675814147
```

In [61]:

```
print("Root mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root mean Squared Error: 0.14654893952469988
```

In [62]:

```
filename20='st'
pickle.dump(lr,open(filename20,'wb'))
```

In [66]:

```
filename20='st'
model=pickle.load(open(filename20,'rb'))
```

In [67]:

```
real=[[10,20,30,40,45],[21,26,34,78,11]]
result=model.predict(real)
```

In [68]:

```
result
```

Out[68]:

```
array([45., 11.])
```

21. DataSet Cities

In [69]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\21_cities - 21_cities.csv")
a
```

Out[69]:

	id	name	state_id	state_code	state_name	country_id	country_code	cou
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	/
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	/
2	78	Jurm	3901	BDS	Badakhshan	1	AF	/
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	/
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	/
...	...	...	...	...	...	...	...	
150449	131496	Redcliff	1957	MI	Midlands Province	247	ZW	
150450	131502	Shangani	1957	MI	Midlands Province	247	ZW	
150451	131503	Shurugwi	1957	MI	Midlands Province	247	ZW	
150452	131504	Shurugwi District	1957	MI	Midlands Province	247	ZW	
150453	131508	Zvishavane District	1957	MI	Midlands Province	247	ZW	

150454 rows × 11 columns

In [71]:

```
b=a.head(10)
b
```

Out[71]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_nam
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanista
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanista
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanista
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanista
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanista
5	131	Wākhān	3901	BDS	Badakhshan	1	AF	Afghanista
6	72	Ghormach	3871	BDG	Badghis	1	AF	Afghanista
7	108	Qala i Naw	3871	BDG	Badghis	1	AF	Afghanista
8	54	Baghlān	3875	BGL	Baghlan	1	AF	Afghanista
9	140	Ḩukūmatī Dahanah-ye Ghōrī	3875	BGL	Baghlan	1	AF	Afghanista

In [72]:

```
b.columns
```

Out[72]:

```
Index(['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataI
d'],
      dtype='object')
```

In [84]:

```
x=b[['id', 'state_id', 'country_id', 'latitude', 'longitude']]
y=b['state_id']
```

In [85]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [86]:

```

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

Out[86]:

```
LinearRegression()
```

In [87]:

```
print(lr.intercept_)
```

```
1.8189894035458565e-12
```

In [88]:

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

Out[88]:

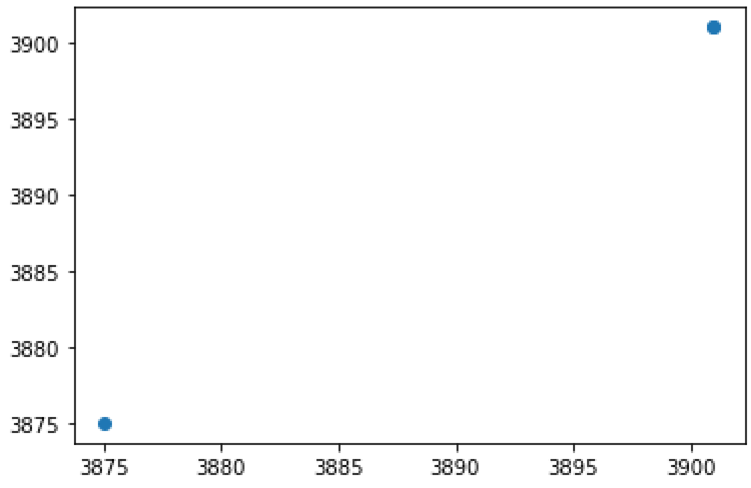
	Co-efficient
id	2.117186e-16
state_id	1.000000e+00
country_id	-1.387779e-16
latitude	-6.945293e-15
longitude	2.077097e-16

In [89]:

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

Out[89]:

```
<matplotlib.collections.PathCollection at 0x228b5401550>
```



In [90]:

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

```
1.0
```

In [91]:

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

Out[91]:

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

In [92]:

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

Out[92]:

```
0.9990974432389651
```

In [93]:

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

Out[93]:

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

In [94]:

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

Out[94]:

```
0.9973559268872856
```

In [95]:

```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[95]:

```
ElasticNet()
```

In [96]:

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

```
[5.09915125e-05 9.95053190e-01 0.00000000e+00 0.00000000e+00
 0.00000000e+00]
```

In [97]:

```
print(en.intercept_)
```

```
19.232244088721927
```

In [98]:

```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999721661179318
```

In [99]:

```
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.06452448815753087
```

In [100]:

```
print("mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean Squared Error: 0.004181267617353147
```

In [101]:

```
print("Root mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root mean Squared Error: 0.06466272200698905
```

In [102]:

```
filename21='ci'
pickle.dump(lr,open(filename21,'wb'))
```

In [103]:

```
filename21='ci'
model=pickle.load(open(filename21,'rb'))
```

In [104]:

```
real=[[10,20,30,40,45],[21,26,34,78,11]]
result=model.predict(real)
```

In [105]:

```
result
```

Out[105]:

```
array([20., 26.])
```

22. DataSet Countries

In [106]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\22_countries - 22_countries.csv")
a
```

Out[106]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	
2	3	Albania	ALB	AL	8	355	Tirana	ALL	Albar
3	4	Algeria	DZA	DZ	12	213	Algiers	DZD	Algeria
4	5	American Samoa	ASM	AS	16	+1-684	Pago Pago	USD	US
...	...	...	...	...	...	...	...	...	
245	243	Wallis And Futuna Islands	WLF	WF	876	681	Mata Utu	XPF	CF
246	244	Western Sahara	ESH	EH	732	212	El-Aaiun	MAD	Mc
247	245	Yemen	YEM	YE	887	967	Sanaa	YER	Yem
248	246	Zambia	ZMB	ZM	894	260	Lusaka	ZMW	Z I
249	247	Zimbabwe	ZWE	ZW	716	263	Harare	ZWL	Zim

250 rows × 19 columns

In [108]:

```
b=a.head(10)
b
```

Out[108]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_na
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan afgh
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	E
2	3	Albania	ALB	AL	8	355	Tirana	ALL	Albanian
3	4	Algeria	DZA	DZ	12	213	Algiers	DZD	Algerian d
4	5	American Samoa	ASM	AS	16	+1-684	Pago Pago	USD	US Dc
5	6	Andorra	AND	AD	20	376	Andorra la Vella	EUR	E
6	7	Angola	AGO	AO	24	244	Luanda	AOA	Angolan kwa
7	8	Anguilla	AIA	AI	660	+1-264	The Valley	XCD	East Caribb dc
8	9	Antarctica	ATA	AQ	10	672	NaN	AAD	Antarcti dc
9	10	Antigua And Barbuda	ATG	AG	28	+1-268	St. John's	XCD	East Caribbean dc

In [109]:

```
b.columns
```

Out[109]:

```
Index(['id', 'name', 'iso3', 'iso2', 'numeric_code', 'phone_code', 'capita
l',
       'currency', 'currency_name', 'currency_symbol', 'tld', 'native',
       'region', 'subregion', 'timezones', 'latitude', 'longitude', 'emoj
i',
       'emojiU'],
      dtype='object')
```

In [110]:

```
x=b[['id','numeric_code','latitude', 'longitude']]
y=b['longitude']
```

In [111]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [112]:

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

Out[112]:

```
LinearRegression()
```

In [113]:

```
print(lr.intercept_)
```

```
7.105427357601002e-15
```

In [114]:

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

Out[114]:

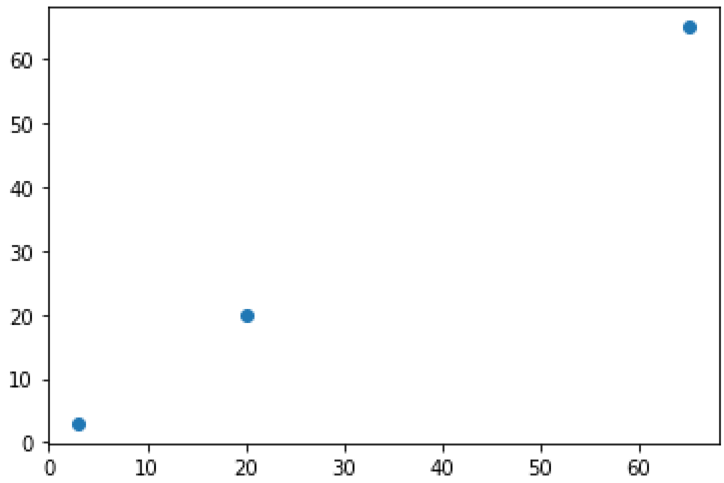
	Co-efficient
id	-1.659114e-16
numeric_code	-1.866388e-16
latitude	1.085731e-16
longitude	1.000000e+00

In [115]:

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

Out[115]:

```
<matplotlib.collections.PathCollection at 0x228b5497c10>
```



In [116]:

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

```
1.0
```

In [117]:

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

Out[117]:

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

In [118]:

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

Out[118]:

```
0.9999992913129515
```

In [119]:

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

Out[119]:

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

In [120]:

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

Out[120]:

```
0.9999565636922261
```

In [121]:

```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[121]:

```
ElasticNet()
```

In [122]:

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

```
[-0.         -0.          0.          0.99975441]
```

In [123]:

```
print(en.intercept_)
```

```
-0.008791643840119434
```

In [124]:

```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.9999995657435908
```

In [125]:

```
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.015995608676841872
```

In [126]:

```
print("mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean Squared Error: 0.00029712788530687593
```

In [127]:

```
print("Root mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root mean Squared Error: 0.017237397869367 52
```

In [129]:

```
filename22='co'
pickle.dump(lr,open(filename22,'wb'))
```

In [137]:

```
filename22='co'
model=pickle.load(open(filename22,'rb'))
```

In [138]:

```
real=[[10,20,30,40],[26,34,78,11]]
result=model.predict(real)
```

In [139]:

```
result
```

Out[139]:

```
array([40., 11.])
```

23. DataSet Vande Bharat

In [140]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\23_Vande Bharat - 23_Vande Bharat.csv")
a
```

Out[140]:

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City
0	1	New Delhi - Varanasi Vande Bharat Express	22435/22436	Delhi	New Delhi	Varanasi
1	2	New Delhi - Shri Mata Vaishno Devi Katra Vande...	22439/22440	Delhi	New Delhi	Katra
2	3	Mumbai Central - Gandhinagar Capital Vande Bha...	20901/20902	Mumbai	Mumbai Central	Gandhinagar
3	4	New Delhi - Amb Andaura Vande Bharat Express	22447/22448	Delhi	New Delhi	Andaura
4	5	MGR Chennai Central - Mysuru Vande Bharat Express	20607/20608	Chennai	Chennai Central	Mysuru
5	6	Bilaspur - Nagpur Vande Bharat Express	20825/20826	Bilaspur, Chhattisgarh	Bilaspur Junction	Nagpur
6	7	Howrah - New Jalpaiguri Vande Bharat Express	22301/22302	Kolkata	Howrah Junction	Siliguri
7	8	Visakhapatnam - Secunderabad Vande Bharat Express	20833/20834	Visakhapatnam	Visakhapatnam Junction	Hyderabad
8	9	Mumbai CSMT - Solapur Vande Bharat Express	22225/22226	Mumbai	Chhatrapati Shivaji Terminus	Solapur
9	10	Mumbai CSMT - Sainagar Shirdi Vande Bharat Exp...	22223/22224	Mumbai	Chhatrapati Shivaji Terminus	Shirdi
10	11	Rani Kamalapati (Habibganj) - Hazrat Nizamuddi...	20171/20172	Bhopal	Habibganj (Rani Kamalapati)	Delhi
11	12	Secunderabad - Tirupati Vande Bharat Express	20701/20702	Hyderabad	Secunderabad Junction	Tirupati
12	13	MGR Chennai Central - Coimbatore Vande Bharat ...	20643/20644	Chennai	Chennai Central	Coimbatore
13	14	Delhi Cantonment - Ajmer Vande Bharat Express	20977/20978	Delhi	Delhi Cantonment	Ajmer
14	15	Kasaragod - Thiruvananthapuram Vande Bharat Ex...	20633/20634	Kasaragod	Kasaragod	Thiruvananthapuram
15	16	Howrah - Puri Vande Bharat Express	22895/22896	Kolkata	Howrah Junction	Puri

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City
16	17	Anand Vihar Terminal - Dehradun Vande Bharat E...	22457/22458	Delhi	Anand Vihar Terminal	Dehradun
17	18	New Jalpaiguri - Guwahati Vande Bharat Express	22227/22228	Siliguri	New Jalpaiguri Junction	Guwahati
18	19	Mumbai CSMT - Madgaon Vande Bharat Express	22229/22230	Mumbai	Chhatrapati Shivaji Terminus	Madgaon
19	19	Mumbai CSMT - Madgaon Vande Bharat Express	22229/22230	Mumbai	Chhatrapati Shivaji Terminus	Madgaon
20	20	Patna - Ranchi Vande Bharat Express	22349/22350	Patna	Patna Junction	Ranchi
21	21	KSR Bengaluru - Dharwad Vande Bharat Express	20661/20662	Bangalore	Bangalore City	Hubbali - Dharwad
22	22	Rani Kamalapati (Habibganj) - Jabalpur Vande B...	20173/20174	Bhopal	Habibganj (Rani Kamalapati)	Jabalpur
23	23	Indore - Bhopal Vande Bharat Express	20911/20912	Indore	Indore Junction	Bhopal
24	24	Jodhpur - Sabarmati (Ahmedabad) Vande Bharat E...	12461/12462	Jodhpur	Jodhpur Junction	Ahmedabad
25	25	Gorakhpur - Lucknow Charbagh Vande Bharat Express	22549/22550	Gorakhpur	Gorakhpur Junction	Charbagh

In [142]:

```
b=a.head(10)
b
```

Out[142]:

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City	Terminal Station
0	1	New Delhi - Varanasi Vande Bharat Express	22435/22436	Delhi	New Delhi	Varanasi	Varanasi Junction
1	2	New Delhi - Shri Mata Vaishno Devi Katra Vande...	22439/22440	Delhi	New Delhi	Katra	Shri Mata Vaishno Devi Katra
2	3	Mumbai Central - Gandhinagar Capital Vande Bha...	20901/20902	Mumbai	Mumbai Central	Gandhinagar	Gandhinagar Capital
3	4	New Delhi - Amb Andaura Vande Bharat Express	22447/22448	Delhi	New Delhi	Andaura	Amb Andaura
4	5	MGR Chennai Central - Mysuru Vande Bharat Express	20607/20608	Chennai	Chennai Central	Mysuru	Mysore Junction
5	6	Bilaspur - Nagpur Vande Bharat Express	20825/20826	Bilaspur, Chhattisgarh	Bilaspur Junction	Nagpur	Nagpur Junction
6	7	Howrah - New Jalpaiguri Vande Bharat Express	22301/22302	Kolkata	Howrah Junction	Siliguri	New Jalpaiguri Junction
7	8	Visakhapatnam - Secunderabad Vande Bharat Express	20833/20834	Visakhapatnam	Visakhapatnam Junction	Hyderabad	Secunderabad Junction
8	9	Mumbai CSMT - Solapur Vande Bharat Express	22225/22226	Mumbai	Chhatrapati Shivaji Terminus	Solapur	Solapur
9	10	Mumbai CSMT - Sainagar Shirdi Vande Bharat Exp...	22223/22224	Mumbai	Chhatrapati Shivaji Terminus	Shirdi	Sainagar Shirdi

In [143]:

```
b.columns
```

Out[143]:

```
Index(['Sr. No.', 'Train Name', 'Train Number', 'Originating City',
       'Originating Station', 'Terminal City', 'Terminal Station', 'Operat
or',
       'No. of Cars', 'Frequency', 'Distance', 'Travel Time', 'Speed',
       'Average Speed', 'Inauguration', 'Average occupancy'],
      dtype='object')
```

In [144]:

```
x=b[['Sr. No.','No. of Cars']]
y=b['No. of Cars']
```

In [145]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [146]:

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

Out[146]:

```
LinearRegression()
```

In [147]:

```
print(lr.intercept_)
```

```
-3.552713678800501e-15
```

In [148]:

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

Out[148]:

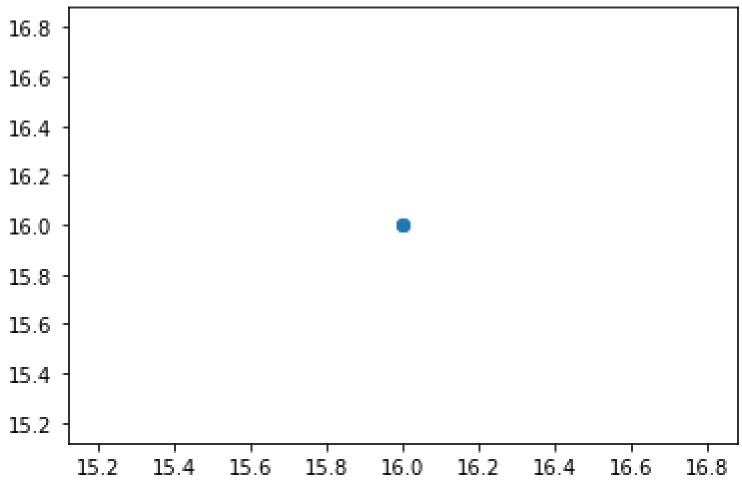
	Co-efficient
Sr. No.	1.199178e-16
No. of Cars	1.000000e+00

In [149]:

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

Out[149]:

```
<matplotlib.collections.PathCollection at 0x228b557bd00>
```



In [150]:

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

```
1.0
```

In [151]:

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

Out[151]:

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

In [152]:

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

Out[152]:

```
0.0
```

In [153]:

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

Out[153]:

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

In [154]:

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

Out[154]:

```
0.0
```

In [155]:

```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[155]:

```
ElasticNet()
```

In [156]:

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

```
[-0.          0.88004896]
```

In [157]:

```
print(en.intercept_)
```

```
1.7821297429620575
```

In [158]:

```
prediction=en.predict(x_test)
print(en.score(x_test,y_test))
```

```
0.0
```

In [159]:

```
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.13708690330477324
```

In [160]:

```
print("mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean Squared Error: 0.01879281905769225
```

In [161]:

```
print("Root mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root mean Squared Error: 0.13708690330477324
```

In [162]:

```
filename23='vb'
pickle.dump(lr,open(filename23,'wb'))
```

In [163]:

```
filename23='vb'
model=pickle.load(open(filename23,'rb'))
```

In [164]:

```
real=[[10,20],[21,26]]
result=model.predict(real)
```

In [165]:

```
result
```

Out[165]:

```
array([20., 26.])
```

In [ ]:

```
```