In [1]:

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

In [2]:

```
df1=pd.read_csv(r'C:\Users\user\Downloads\19_nuclear_explosions.csv')
df1
```

Out[2]:

	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 [3]:

```
df=df1.head(30)
df
```

Out[3]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.L
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	
5	USA	Enewetak	DOE	11.30	
6	USA	Enewetak	DOE	11.30	
7	USA	Enewetak	DOE	11.30	
8	USSR	Semi Kazakh	DOE	48.00	
9	USA	Nts	DOE	37.00	
10	USA	Nts	DOE	37.00	
11	USA	Nts	DOE	37.00	
12	USA	Nts	DOE	37.00	
13	USA	Nts	DOE	37.00	
14	USA	Enewetak	DOE	11.30	
15	USA	Enewetak	DOE	11.30	
16	USA	Enewetak	DOE	11.30	
17	USA	Enewetak	DOE	11.30	
18	USSR	Semi Kazakh	DOE	48.00	
19	USSR	Semi Kazakh	DOE	48.00	
20	USA	Nts	DOE	37.00	
21	USA	Nts	DOE	37.00	
22	USA	Nts	DOE	37.00	
23	USA	Nts	DOE	37.00	
24	USA	Nts	DOE	37.00	
25	USA	Nts	DOE	37.00	
26	USA	Nts	DOE	37.00	
27	USA	Nts	DOE	37.00	
28	USA	Nts	DOE	37.00	
29	USA	Nts	DOE	37.00	

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   WEAPON SOURCE COUNTRY           30 non-null     object
 1   WEAPON DEPLOYMENT LOCATION      30 non-null     object
 2   Data.Source                     30 non-null     object
 3   Location.Cordinates.Latitude    30 non-null     float64
 4   Location.Cordinates.Longitude   30 non-null     float64
 5   Data.Magnitude.Body             30 non-null     float64
 6   Data.Magnitude.Surface          30 non-null     float64
 7   Location.Cordinates.Depth       30 non-null     float64
 8   Data.Yeild.Lower                30 non-null     float64
 9   Data.Yeild.Upper                30 non-null     float64
 10  Data.Purpose                    30 non-null     object
 11  Data.Name                       30 non-null     object
 12  Data.Type                       30 non-null     object
 13  Date.Day                        30 non-null     int64
 14  Date.Month                      30 non-null     int64
 15  Date.Year                       30 non-null     int64
dtypes: float64(7), int64(3), object(6)
memory usage: 3.9+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.
count	30.000000	30.000000	30.0	
mean	30.000667	3.655667	0.0	
std	12.973211	131.675323	0.0	
min	11.300000	-116.000000	0.0	
25%	11.350000	-116.000000	0.0	
50%	37.000000	-110.785000	0.0	
75%	37.000000	162.150000	0.0	
max	48.000000	165.200000	0.0	

In [6]:

```
df.columns
```

Out[6]:

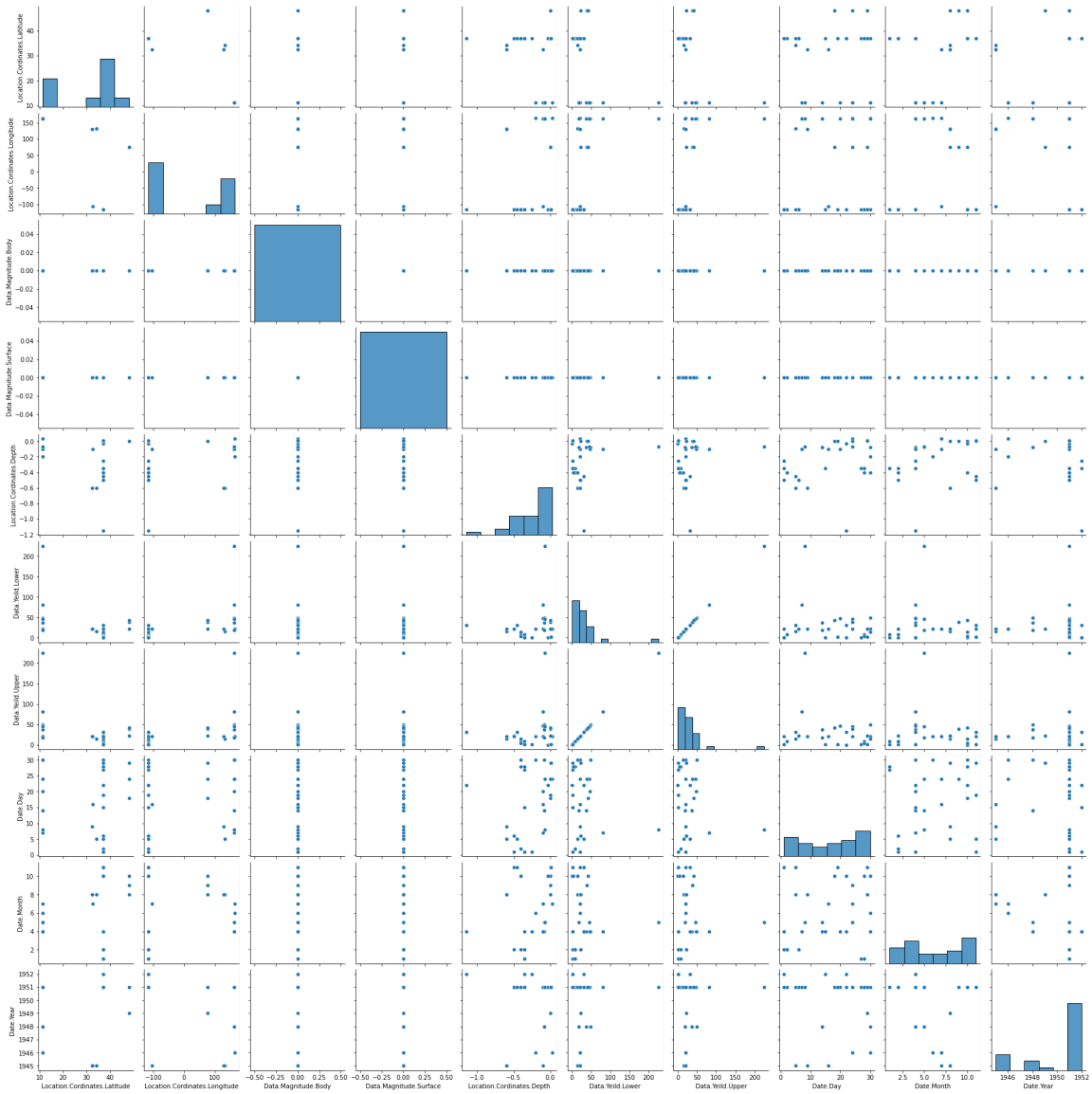
```
Index(['WEAPON SOURCE COUNTRY', 'WEAPON DEPLOYMENT LOCATION', 'Data.Sourc
e',
       'Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Uppe
r',
       'Data.Purpose',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

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1bea64e15e0>
```



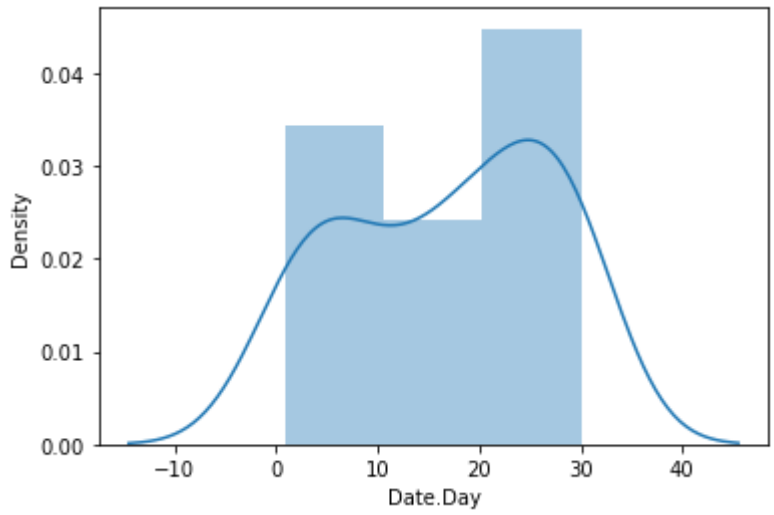
In [8]:

```
sns.distplot(df['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[8]:

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

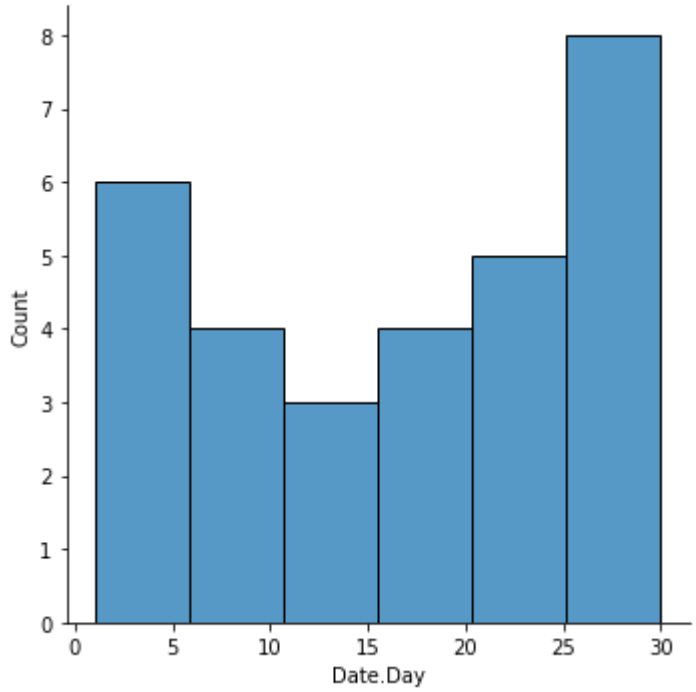


In [9]:

```
sns.displot(df["Date.Day"])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1bea92e74c0>
```



In [10]:

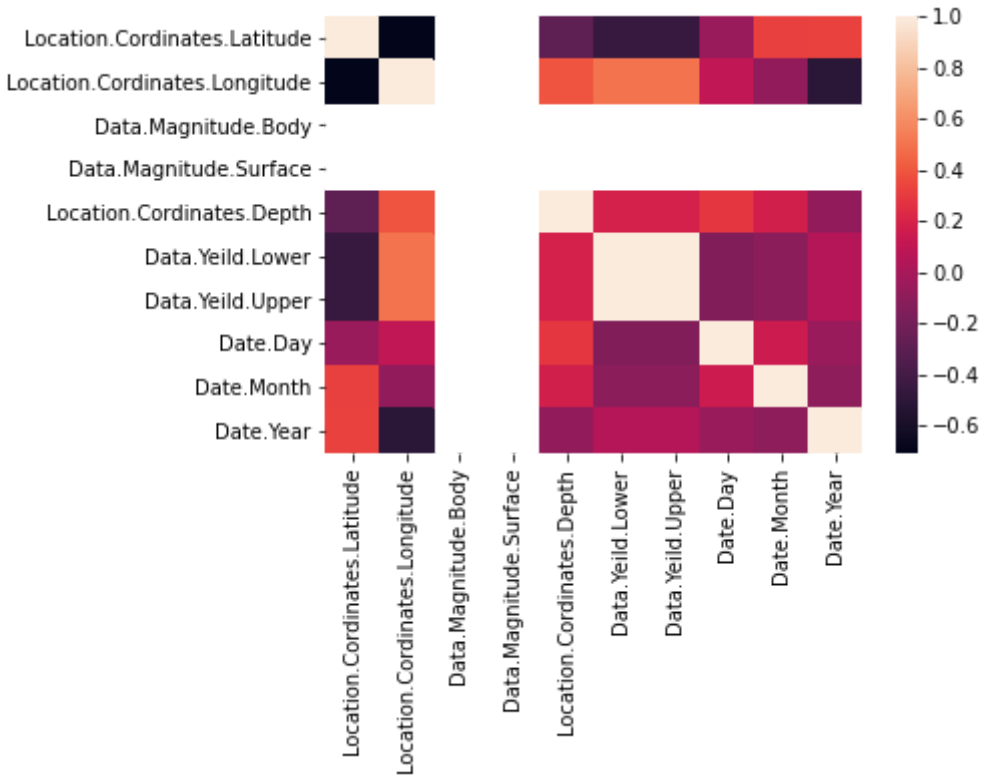
```
df1=df[['WEAPON SOURCE COUNTRY', 'WEAPON DEPLOYMENT LOCATION', 'Data.Source',
       'Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper',
       'Data.Purpose', 'Data.Name', 'Data.Type', 'Date.Day', 'Date.Month',
       'Date.Year']]
```

In [11]:

```
sns.heatmap(df1.corr())
```

Out[11]:

<AxesSubplot:>



In [12]:

```
x=df1[[ 'Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper', 'Date.Day', '
       'Date.Year']]
y=df1[['Date.Day']]
```

In [13]:

```
from sklearn.model_selection import train_test_split
```

In [14]:

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

In [15]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for
```

Out[15]:

```
LinearRegression()
```

In [16]:

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

```
[-2.38031816e-13]
```

In [17]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

Out[17]:

	0	1	2	3	4	5	6	7	
0	-9.165257e-17	-1.005594e-16	1.387779e-16	0.0	2.346633e-14	7.722610e-17	1.743706e-16	1.0	-1.636705

In [18]:

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

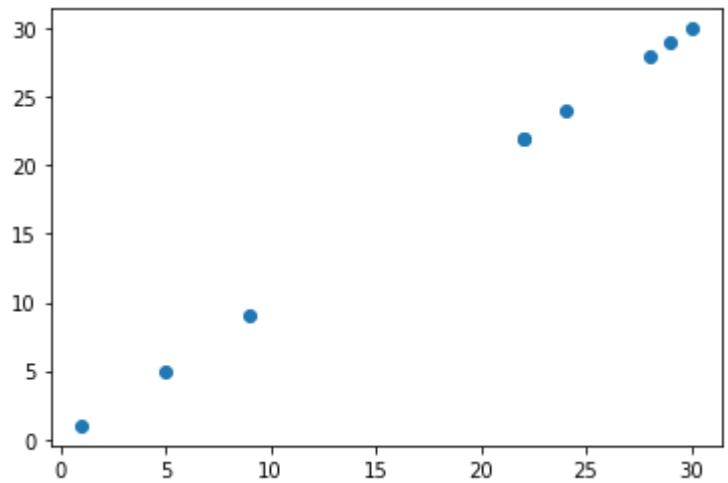
```
1.0
```

In [19]:

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

Out[19]:

```
<matplotlib.collections.PathCollection at 0x1beabf37760>
```



In [20]:

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

Out[20]:

```
1.0
```

In [21]:

```
lr.score(x_train,y_train)
```

Out[21]:

```
1.0
```

In [22]:

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

In [23]:

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

Out[23]:

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

In [24]:

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

Out[24]:

```
0.9999647146900132
```

In [25]:

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

Out[25]:

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

In [26]:

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

Out[26]:

```
0.9876463368036928
```

Elastic Net

In [27]:

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

Out[27]:

```
ElasticNet()
```

In [28]:

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

```
[-0.00000000e+00  1.75570211e-04  0.00000000e+00  0.00000000e+00
  0.00000000e+00 -3.95112145e-04 -2.08691527e-05  9.88855397e-01
  0.00000000e+00 -0.00000000e+00]
```

In [29]:

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

```
[0.18917113]
```

In [30]:

```
prediction=en.predict(x_test)
print(prediction)
```

```
[21.92362164  9.10287395  5.10018655 23.93124222 29.82864316 21.9107283
 28.8701694  27.85530017  1.1572444 ]
```

In [31]:

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

```
0.9998641977232955
```

Evaluation Metrics

In [32]:

```
from sklearn import metrics
```

In [33]:

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

```
Mean Absolute Error: 0.11562222231747334
```

In [34]:

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

```
Mean Squared Error: 0.014559345319782603
```

In [35]:

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

```
Root Mean Squared Error: 0.1206621121967563
```

Model Saving

In [36]:

```
import pickle
```

In [37]:

```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```