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

In [2]:
```
df=pd.read_csv("19_nuclear_explosions.csv")
df
```

Out[2]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longitude
0	USA	Alamogordo	DOE	32.54	-105.57
1	USA	Hiroshima	DOE	34.23	132.27
2	USA	Nagasaki	DOE	32.45	129.52
3	USA	Bikini	DOE	11.35	165.20
4	USA	Bikini	DOE	11.35	165.20
...	...	...	...	...	...
2041	CHINA	Lop Nor	HFS	41.69	88.35
2042	INDIA	Pokhran	HFS	27.07	71.70
2043	INDIA	Pokhran	NRD	27.07	71.70
2044	PAKIST	Chagai	HFS	28.90	64.89
2045	PAKIST	Kharan	HFS	28.49	63.78

2046 rows × 16 columns

In [3]:
```
df.head()
```

Out[3]:

	WEAPON SOURCE COUNTRY	WEAPON DEPLOYMENT LOCATION	Data.Source	Location.Cordinates.Latitude	Location.Cordinates.Longitude	D
0	USA	Alamogordo	DOE	32.54	-105.57	
1	USA	Hiroshima	DOE	34.23	132.27	
2	USA	Nagasaki	DOE	32.45	129.52	
3	USA	Bikini	DOE	11.35	165.20	
4	USA	Bikini	DOE	11.35	165.20	

Data Cleaning and Data Preprocessing

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2046 entries, 0 to 2045
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   WEAPON SOURCE COUNTRY           2046 non-null   object 
 1   WEAPON DEPLOYMENT LOCATION      2046 non-null   object 
 2   Data.Source                     2046 non-null   object 
 3   Location.Cordinates.Latitude    2046 non-null   float64
 4   Location.Cordinates.Longitude   2046 non-null   float64
 5   Data.Magnitude.Body             2046 non-null   float64
 6   Data.Magnitude.Surface          2046 non-null   float64
 7   Location.Cordinates.Depth       2046 non-null   float64
 8   Data.Yeild.Lower                2046 non-null   float64
 9   Data.Yeild.Upper                2046 non-null   float64
 10  Data.Purpose                    2046 non-null   object 
 11  Data.Name                       2046 non-null   object 
 12  Data.Type                       2046 non-null   object 
 13  Date.Day                        2046 non-null   int64  
 14  Date.Month                      2046 non-null   int64  
 15  Date.Year                       2046 non-null   int64  
dtypes: float64(7), int64(3), object(6)
memory usage: 255.9+ KB
```

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

Out[5]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magnitude
count	2046.000000	2046.000000	2046.000000	204
mean	35.462429	-36.015037	2.145406	
std	23.352702	100.829355	2.625453	
min	-49.500000	-169.320000	0.000000	
25%	37.000000	-116.051500	0.000000	
50%	37.100000	-116.000000	0.000000	
75%	49.870000	78.000000	5.100000	
max	75.100000	179.220000	7.400000	

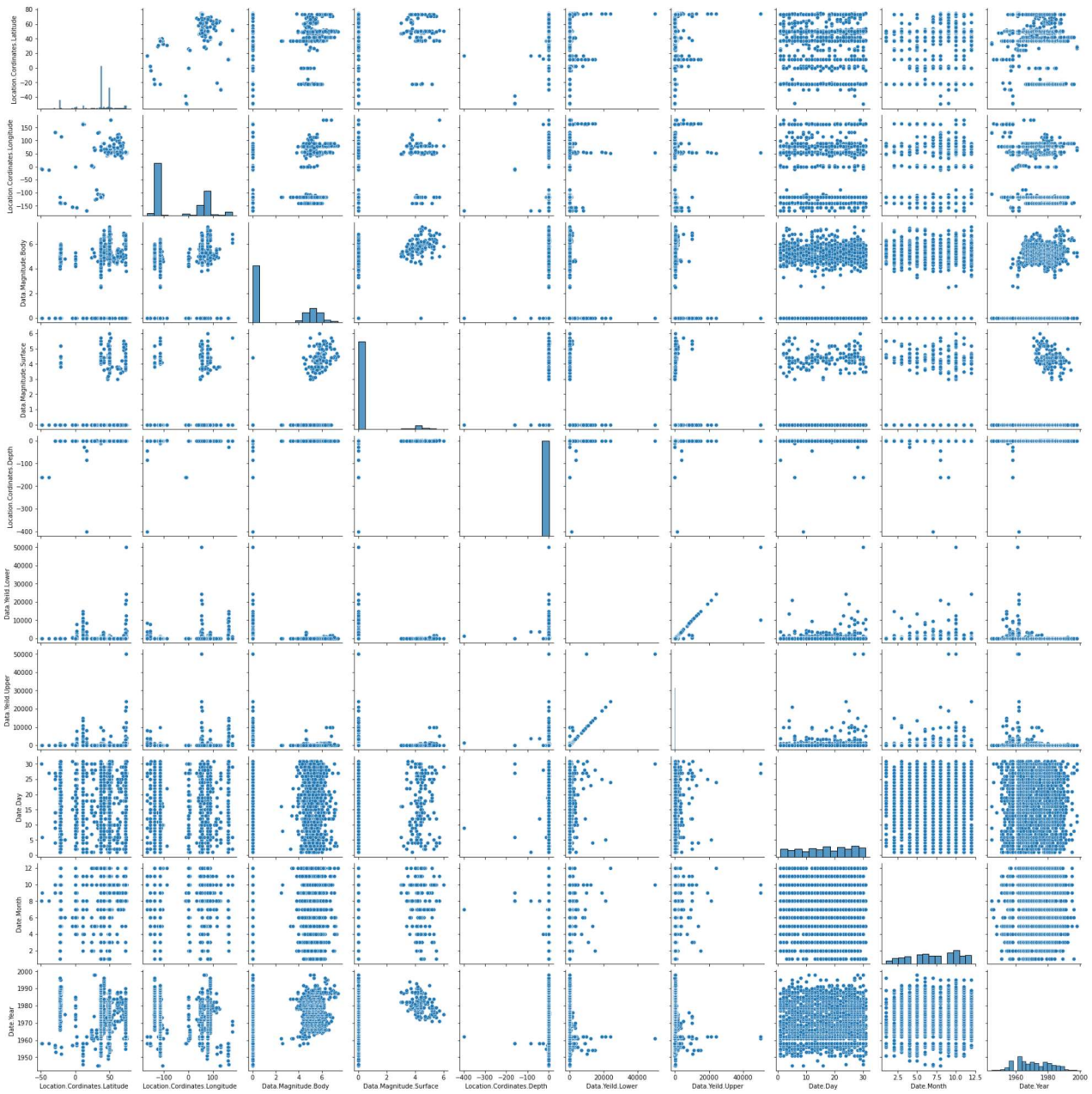
In [6]:
```
df.columns
```

Out[6]:
```
Index(['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'],
      dtype='object')
```

EDA and Visualization

In [7]:
```
sns.pairplot(df)
```

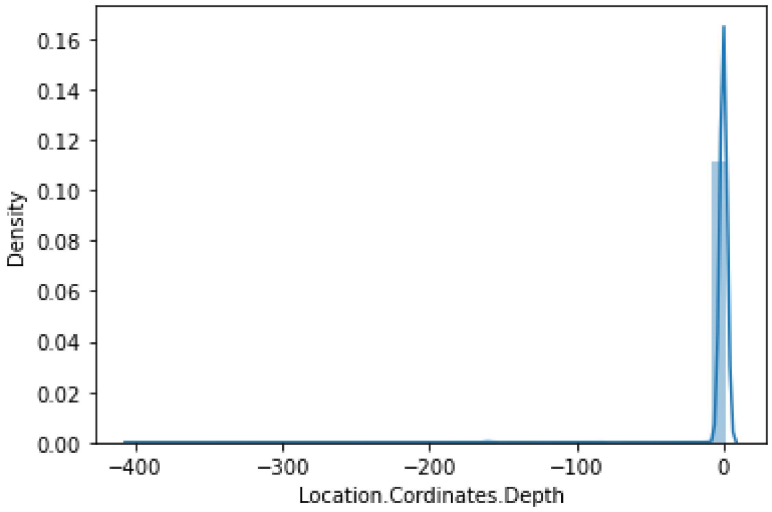
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x227b1ca3400>
```



In [8]:
```
sns.distplot(df['Location.Cordinates.Depth'])
```

```
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='Location.Cordinates.Depth', ylabel='Density'>
```



In [9]:
```
df1=df[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper',
        'Date.Day', 'Date.Month',
       'Date.Year']]
df1
```

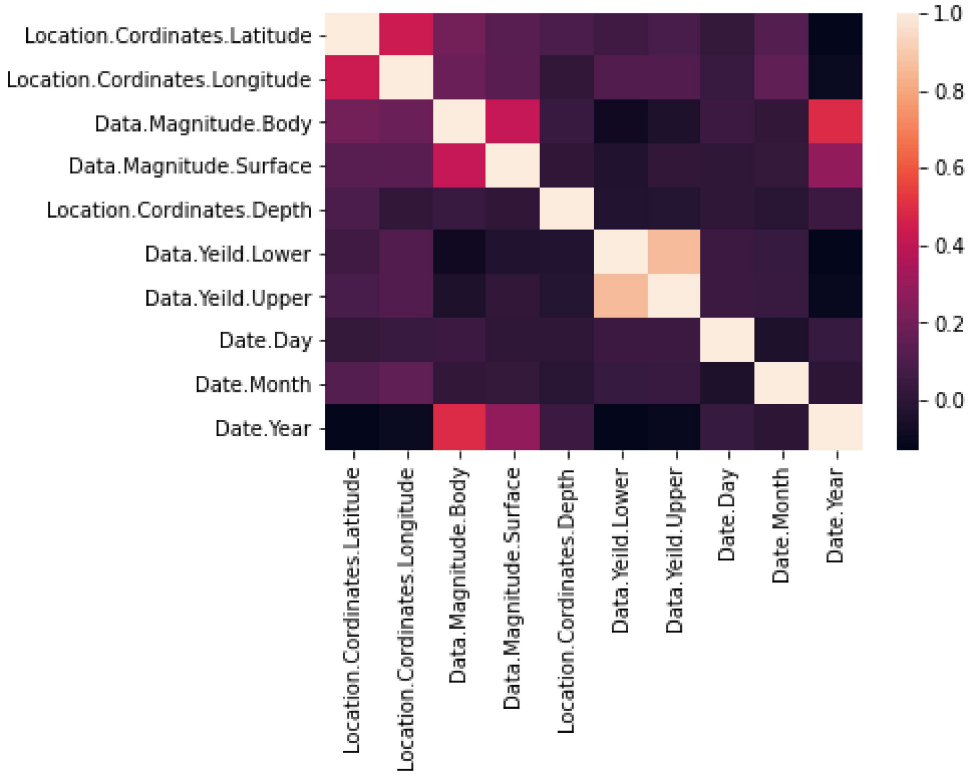
Out[9]:

	Location.Cordinates.Latitude	Location.Cordinates.Longitude	Data.Magnitude.Body	Data.Magnitude.
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	
...	...	...	...	
2041	41.69	88.35	5.3	
2042	27.07	71.70	5.3	
2043	27.07	71.70	0.0	
2044	28.90	64.89	0.0	
2045	28.49	63.78	5.0	

2046 rows × 10 columns

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

Out[10]: <AxesSubplot:>



To Train the Model -Model Building

We are going to train Linear Regression model;We need to spilt out data into two variables x and y where x is independent variable (input) and y is dependent variable on x(output) we could ignore address column as it is not requiired for our model

In [11]:
```
x=df1[['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
        'Data.Yeild.Lower', 'Data.Yeild.Upper',
        'Date.Day', 'Date.Month',
       'Date.Year']]
y=df1['Location.Cordinates.Depth']
```

In [12]:
```
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 [13]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
print(lr.intercept_)
```

```
-179.88742026434423
```

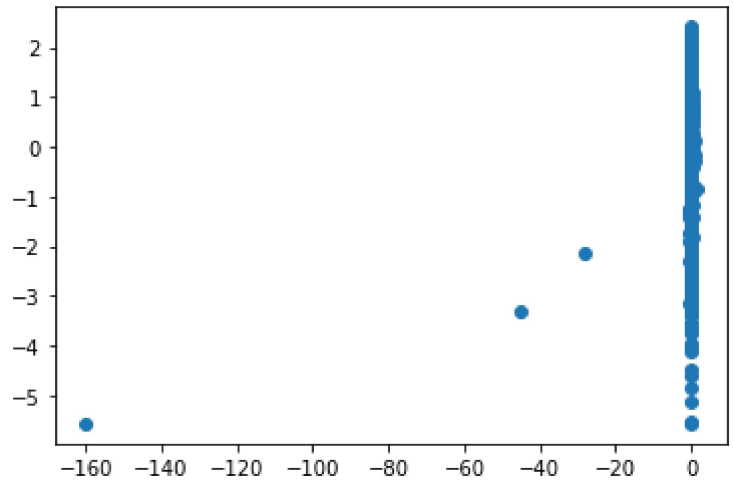
In [15]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
Location.Cordinates.Latitude	0.052286
Location.Cordinates.Longitude	0.000096
Data.Magnitude.Body	-0.046126
Data.Magnitude.Surface	-0.156925
Data.Yeild.Lower	-0.000186
Data.Yeild.Upper	0.000027
Date.Day	0.033545
Date.Month	-0.051510
Date.Year	0.090055

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

Out[16]:
```
<matplotlib.collections.PathCollection at 0x227c5aee820>
```



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

Out[17]:
```
0.024414836267849904
```

In [18]:
```
lr.score(x_train,y_train)
```

Out[18]:
```
0.012981353642119164
```

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]:
```
Ridge(alpha=10)
```

In [21]:
```
rr.score(x_test,y_test)
```

Out[21]:
```
0.024420679897683706
```

In [22]:
```
rr.score(x_train,y_train)
```

Out[22]:
```
0.012981347308700375
```

In [23]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[23]:
```
Lasso(alpha=10)
```

In [24]:
```
la.score(x_test,y_test)
```

Out[24]:
```
0.019869627637321474
```

In [25]:
```
la.score(x_train,y_train)
```

Out[25]:
```
0.006443516713328656
```

In [26]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[26]:
```
ElasticNet()
```

In [27]:
```
en.coef_
```

Out[27]:
```
array([ 4.84156956e-02, -4.07646847e-04, -0.00000000e+00, -0.00000000e+00,
       -1.65907437e-04,  9.95124406e-06,  2.85390787e-02, -0.00000000e+00,
        7.17542262e-02])
```

In [28]:
```
en.intercept_
```

Out[28]:
```
-144.15037095560285
```

In [29]:
```
prediction=en.predict(x_test)
```

In [30]:
```
en.score(x_test,y_test)
```

Out[30]: 0.02522848027069302

Evaluation Metrics

In [31]:
```
from sklearn import metrics
```

In [32]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 1.4474289448820254
```

In [33]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 45.00041128346145
```

In [34]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 6.708234587688586
```

Model Saving

In [35]:
```
import pickle
```

In [36]:
```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [37]:
```
import pandas as pd
import pickle
```

In [38]:
```
filename="prediction"
model=pickle.load(open(filename,'rb'))
```

In [39]:
```
real=[[19,21,45,37,56,62,70,65,60],[11,45,10,25,33,55,23,90,64]]
result=model.predict(real)
```

In [40]:
```
result
```

Out[40]: array([-182.37928115, -181.79782072])