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

In [2]:
```
#import dataset
df=pd.read_csv(r"E:\154\9_bottle.csv",low_memory=False).dropna(axis='columns')
df
```

Out[2]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	RecInd	R_Depth	R_PRES
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	3	0.0	0
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	3	8.0	8
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	7	10.0	10
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	3	19.0	19
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	7	20.0	20
...	...	...	...	...	...	...	...	...
864858	34404	864859	093.4 026.4	20-1611SR-MX-310-2239-09340264-0000A-7	0	7	0.0	0
864859	34404	864860	093.4 026.4	20-1611SR-MX-310-2239-09340264-0002A-3	2	3	2.0	2
864860	34404	864861	093.4 026.4	20-1611SR-MX-310-2239-09340264-0005A-3	5	3	5.0	5
864861	34404	864862	093.4 026.4	20-1611SR-MX-310-2239-09340264-0010A-3	10	3	10.0	10
864862	34404	864863	093.4 026.4	20-1611SR-MX-310-2239-09340264-0015A-3	15	3	15.0	15

864863 rows × 8 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 8 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   Cst_Cnt    864863 non-null  int64  
 1   Btl_Cnt    864863 non-null  int64  
 2   Sta_ID     864863 non-null  object 
 3   Depth_ID   864863 non-null  object 
 4   Depthm     864863 non-null  int64  
 5   RecInd     864863 non-null  int64  
 6   R_Depth    864863 non-null  float64
 7   R_PRES     864863 non-null  int64  
dtypes: float64(1), int64(5), object(2)
memory usage: 52.8+ MB
```

In [4]:
```
#to display top 5 rows
df.head()
```

Out[4]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	RecInd	R_Depth	R_PRES
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	3	0.0	0
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	3	8.0	8
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	7	10.0	10
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	3	19.0	19
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	7	20.0	20

Data cleaning and Pre-Processing

In [5]:
```
#To find null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 8 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   Cst_Cnt    864863 non-null  int64  
 1   Btl_Cnt    864863 non-null  int64  
 2   Sta_ID     864863 non-null  object 
 3   Depth_ID   864863 non-null  object 
 4   Depthm     864863 non-null  int64  
 5   RecInd     864863 non-null  int64  
 6   R_Depth    864863 non-null  float64
 7   R_PRES     864863 non-null  int64  
dtypes: float64(1), int64(5), object(2)
memory usage: 52.8+ MB
```

In [6]:
```
# To display summary of statistics
df.describe()
```

Out[6]:

	Cst_Cnt	Btl_Cnt	Depthm	RecInd	R_Depth	R_PRE
count	864863.000000	864863.000000	864863.000000	864863.000000	864863.000000	864863.00000
mean	17138.790958	432432.000000	226.831951	4.700273	226.832495	228.39569
std	10240.949817	249664.587267	316.050259	1.877428	316.050007	319.45673
min	1.000000	1.000000	0.000000	3.000000	0.000000	0.00000
25%	8269.000000	216216.500000	46.000000	3.000000	46.000000	46.00000
50%	16848.000000	432432.000000	125.000000	3.000000	125.000000	126.00000
75%	26557.000000	648647.500000	300.000000	7.000000	300.000000	302.00000
max	34404.000000	864863.000000	5351.000000	7.000000	5351.000000	5458.00000

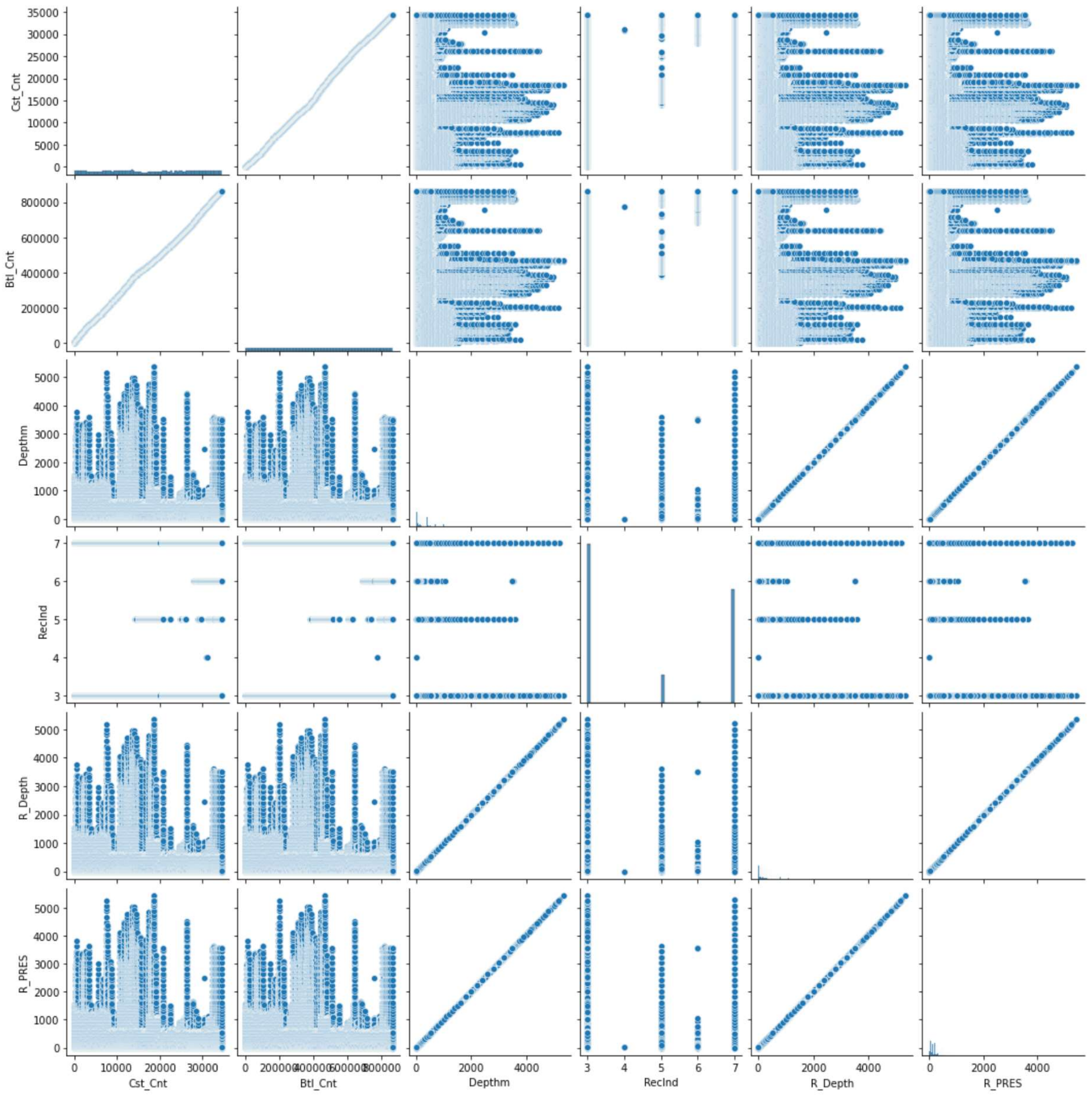
In [7]:
```
#To Display column heading
df.columns
```

Out[7]:
```
Index(['Cst_Cnt', 'Btl_Cnt', 'Sta_ID', 'Depth_ID', 'Depthm', 'RecInd',
       'R_Depth', 'R_PRES'],
      dtype='object')
```

EDA and VISUALIZATION

In [8]:
```
sns.pairplot(df)
```

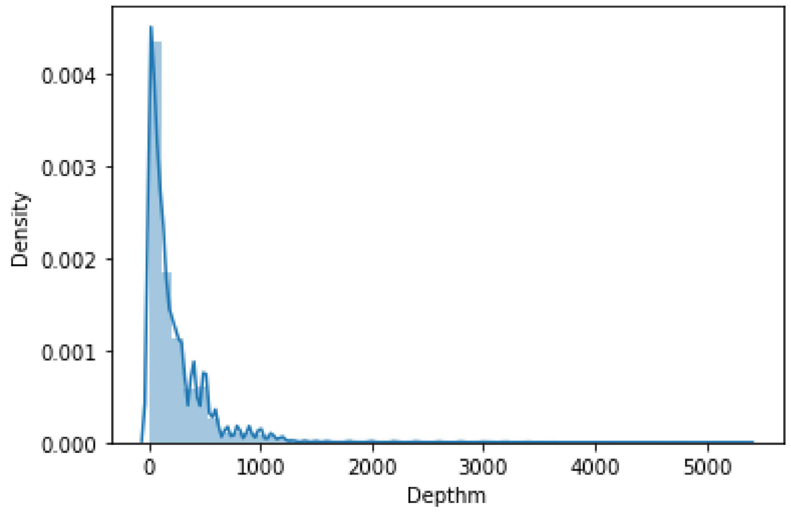
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x142058fdc10>
```



In [9]:
```
sns.distplot(df["Depthm"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:
```
<AxesSubplot:xlabel='Depthm', ylabel='Density'>
```

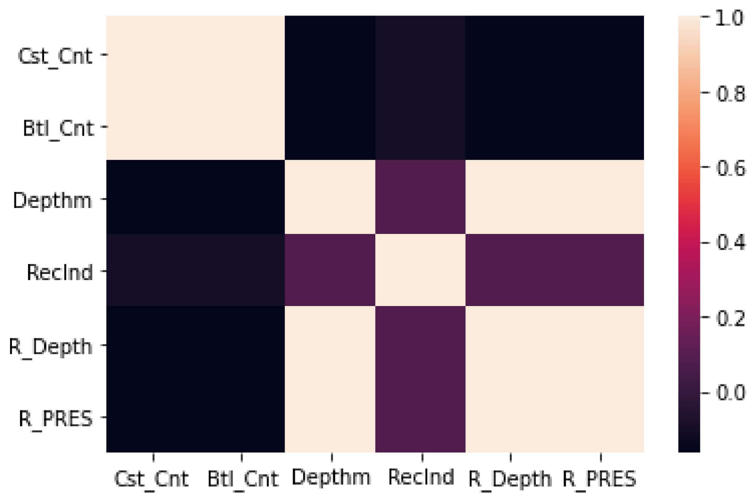


In [10]:
```
df1=df[['Cst_Cnt','Btl_Cnt','Sta_ID','Depth_ID','Depthm','RecInd',
       'R_Depth','R_PRES']]
```

Plot Using Heat Map

In [11]:
```
sns.heatmap(df1.corr())
```

Out[11]:
```
<AxesSubplot:>
```



To Train The Model-Model Building

we are going to train Linera Regression Model;We need to split out data into two variables x and y where x is independent variable(input) and y is dependent on x(output) we could ignore address column as it required for our model

In [12]:
```
x=df1[['Cst_Cnt','Btl_Cnt','RecInd', 'R_Depth','R_PRES']]
y=df1['Depthm']
```

To Split my dataset into training and test data

In [13]:
```

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 [14]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]:
```
LinearRegression()
```

In [15]:
```
lr.intercept_
```

Out[15]:
```
0.0029030334723927353
```

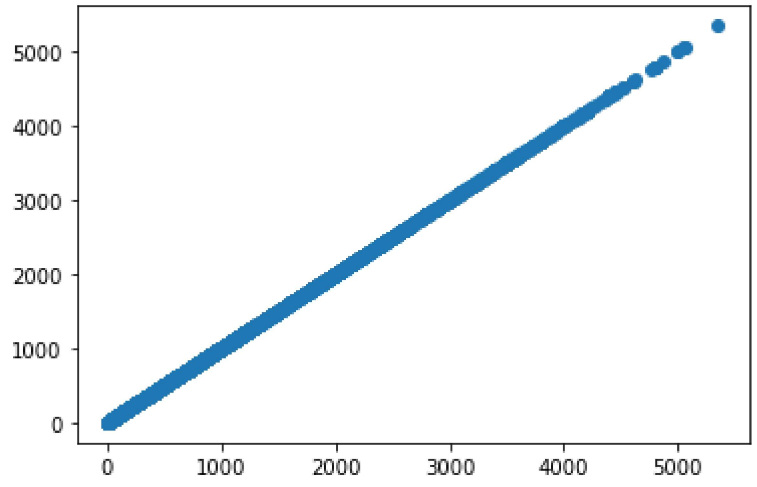
In [16]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Cst_Cnt	1.677602e-06
Btl_Cnt	-7.240490e-08
RecInd	-2.677534e-04
R_Depth	1.000287e+00
R_PRES	-2.835514e-04

```
In [17]: prediction = lr.predict(x_test)
         plt.scatter(y_test,prediction)

Out[17]: <matplotlib.collections.PathCollection at 0x142116a9340>
```



Accuracy

```
In [18]: lr.score(x_test,y_test)

Out[18]: 0.9999999945681218

In [19]: lr.score(x_train,y_train)

Out[19]: 0.9999999945827198

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]: 0.9999999945680983

In [27]: la =Lasso(alpha=10)
         la.fit(x_train,y_train)

Out[27]: Lasso(alpha=10)

In [28]: la.score(x_test,y_test)

Out[28]: 0.9999999837557709
```