

# Problem Statement:

A real estate agent want to help to predict the house price for regions in USA. He gave us the dataset to work on to use Linear Regression model. Create a Model that helps him to estimate of what the house would sell for.

```
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("bottle.csv")
df
```

C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3165: DtypeWarning: Columns (47,73) have mixed types. Specify dtype option on import or set low\_memory=False.  
has\_raised = await self.run\_ast\_nodes(code\_ast.body, cell\_name,

Out[2]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	R
--	---------	---------	--------	----------	--------	--------	--------	--------	--------	-------	---

<b>0</b>	1	1	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0000A-3	0	10.500	33.4400	NaN	25.64900	NaN	...
<b>1</b>	1	2	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0008A-3	8	10.460	33.4400	NaN	25.65600	NaN	...
<b>2</b>	1	3	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0010A-7	10	10.460	33.4370	NaN	25.65400	NaN	...
<b>3</b>	1	4	054.0 056.0	19- 4903CR- HY-060- 0930- 05400560- 0019A-3	19	10.450	33.4200	NaN	25.64300	NaN	...
<b>4</b>	1	5	054.0 056.0	19- 4903CR- HY-060- 0930-	20	10.450	33.4210	NaN	25.64300	NaN	...

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R
					05400560-0020A-7							
	...	...	...	...	...						...	...
					20-1611SR-MX-310-2239-09340264-0000A-7							
<b>864858</b>	34404	864859	093.4 026.4			0	18.744	33.4083	5.805	23.87055	108.74	...
					20-1611SR-MX-310-2239-09340264-0002A-3							
<b>864859</b>	34404	864860	093.4 026.4			2	18.744	33.4083	5.805	23.87072	108.74	...
					20-1611SR-MX-310-2239-09340264-0005A-3							
<b>864860</b>	34404	864861	093.4 026.4			5	18.692	33.4150	5.796	23.88911	108.46	...
					20-1611SR-MX-310-2239-09340264-0010A-3							
<b>864861</b>	34404	864862	093.4 026.4			10	18.161	33.4062	5.816	24.01426	107.74	...
					20-1611SR-MX-310-2239-09340264-0015A-3							
<b>864862</b>	34404	864863	093.4 026.4			15	17.533	33.3880	5.774	24.15297	105.66	...

864863 rows × 74 columns

In [3]:

df.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 74 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   T_degC          853900 non-null  float64 
 6   Salnty          817509 non-null  float64 
 7   O2ml_L          696201 non-null  float64 

```

8	STheta	812174	non-null	float64
9	O2Sat	661274	non-null	float64
10	Oxy_µmol/Kg	661268	non-null	float64
11	BtlNum	118667	non-null	float64
12	RecInd	864863	non-null	int64
13	T_prec	853900	non-null	float64
14	T_qual	23127	non-null	float64
15	S_prec	817509	non-null	float64
16	S_qual	74914	non-null	float64
17	P_qual	673755	non-null	float64
18	O_qual	184676	non-null	float64
19	SThtaq	65823	non-null	float64
20	O2Satq	217797	non-null	float64
21	ChlorA	225272	non-null	float64
22	Chlqua	639166	non-null	float64
23	Phaeop	225271	non-null	float64
24	Phaqua	639170	non-null	float64
25	P04uM	413317	non-null	float64
26	P04q	451786	non-null	float64
27	SiO3uM	354091	non-null	float64
28	SiO3qu	510866	non-null	float64
29	NO2uM	337576	non-null	float64
30	NO2q	529474	non-null	float64
31	NO3uM	337403	non-null	float64
32	NO3q	529933	non-null	float64
33	NH3uM	64962	non-null	float64
34	NH3q	808299	non-null	float64
35	C14As1	14432	non-null	float64
36	C14A1p	12760	non-null	float64
37	C14A1q	848605	non-null	float64
38	C14As2	14414	non-null	float64
39	C14A2p	12742	non-null	float64
40	C14A2q	848623	non-null	float64
41	DarkAs	22649	non-null	float64
42	DarkAp	20457	non-null	float64
43	DarkAq	840440	non-null	float64
44	MeanAs	22650	non-null	float64
45	MeanAp	20457	non-null	float64
46	MeanAq	840439	non-null	float64
47	IncTim	14437	non-null	object
48	LightP	18651	non-null	float64
49	R_Depth	864863	non-null	float64
50	R_TEMP	853900	non-null	float64
51	R_POTEMP	818816	non-null	float64
52	R_SALINITY	817509	non-null	float64
53	R_SIGMA	812007	non-null	float64
54	R_SVA	812092	non-null	float64
55	R_DYNHT	818206	non-null	float64
56	R_O2	696201	non-null	float64
57	R_O2Sat	666448	non-null	float64
58	R_SI03	354099	non-null	float64
59	R_P04	413325	non-null	float64
60	R_NO3	337411	non-null	float64
61	R_NO2	337584	non-null	float64
62	R_NH4	64982	non-null	float64
63	R_CHLA	225276	non-null	float64
64	R_PHAEAO	225275	non-null	float64
65	R_PRES	864863	non-null	int64
66	R_SAMP	122006	non-null	float64
67	DIC1	1999	non-null	float64
68	DIC2	224	non-null	float64
69	TA1	2084	non-null	float64
70	TA2	234	non-null	float64
71	pH2	10	non-null	float64
72	pH1	84	non-null	float64

```
73 DIC Quality Comment 55 non-null      object
dtypes: float64(65), int64(5), object(4)
memory usage: 488.3+ MB
```

In [4]:

`df.head()`

Out[4]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R_PHAE0
0	1	1	054.0 056.0	HY-060- 0930- 05400560- 0000A-3	19- 4903CR-	0	10.50	33.440	NaN	25.649	NaN	NaN
1	1	2	054.0 056.0	HY-060- 0930- 05400560- 0008A-3	19- 4903CR-	8	10.46	33.440	NaN	25.656	NaN	NaN
2	1	3	054.0 056.0	HY-060- 0930- 05400560- 0010A-7	19- 4903CR-	10	10.46	33.437	NaN	25.654	NaN	NaN
3	1	4	054.0 056.0	HY-060- 0930- 05400560- 0019A-3	19- 4903CR-	19	10.45	33.420	NaN	25.643	NaN	NaN
4	1	5	054.0 056.0	HY-060- 0930- 05400560- 0020A-7	19- 4903CR-	20	10.45	33.421	NaN	25.643	NaN	NaN

5 rows × 74 columns



## Data cleaning and Pre-Processing

In [5]:

`df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 74 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	T_degC	853900	non-null	float64
6	Salnty	817509	non-null	float64
7	O2ml_L	696201	non-null	float64
8	STheta	812174	non-null	float64
9	O2Sat	661274	non-null	float64
10	Oxy_µmol/Kg	661268	non-null	float64
11	BtlNum	118667	non-null	float64
12	RecInd	864863	non-null	int64
13	T_prec	853900	non-null	float64
14	T_qual	23127	non-null	float64
15	S_prec	817509	non-null	float64
16	S_qual	74914	non-null	float64
17	P_qual	673755	non-null	float64
18	O_qual	184676	non-null	float64
19	SThtaq	65823	non-null	float64
20	O2Satq	217797	non-null	float64
21	ChlorA	225272	non-null	float64
22	Chlqua	639166	non-null	float64
23	Phaeop	225271	non-null	float64
24	Phaqua	639170	non-null	float64
25	P04uM	413317	non-null	float64
26	P04q	451786	non-null	float64
27	SiO3uM	354091	non-null	float64
28	SiO3qu	510866	non-null	float64
29	NO2uM	337576	non-null	float64
30	NO2q	529474	non-null	float64
31	NO3uM	337403	non-null	float64
32	NO3q	529933	non-null	float64
33	NH3uM	64962	non-null	float64
34	NH3q	808299	non-null	float64
35	C14As1	14432	non-null	float64
36	C14A1p	12760	non-null	float64
37	C14A1q	848605	non-null	float64
38	C14As2	14414	non-null	float64
39	C14A2p	12742	non-null	float64
40	C14A2q	848623	non-null	float64
41	DarkAs	22649	non-null	float64
42	DarkAp	20457	non-null	float64
43	DarkAq	840440	non-null	float64
44	MeanAs	22650	non-null	float64
45	MeanAp	20457	non-null	float64
46	MeanAq	840439	non-null	float64
47	IncTim	14437	non-null	object
48	LightP	18651	non-null	float64
49	R_Depth	864863	non-null	float64
50	R_TEMP	853900	non-null	float64
51	R_POTEMP	818816	non-null	float64
52	R_SALINITY	817509	non-null	float64
53	R_SIGMA	812007	non-null	float64
54	R_SVA	812092	non-null	float64
55	R_DYNHT	818206	non-null	float64
56	R_O2	696201	non-null	float64
57	R_O2Sat	666448	non-null	float64
58	R_SI03	354099	non-null	float64
59	R_P04	413325	non-null	float64
60	R_N03	337411	non-null	float64
61	R_N02	337584	non-null	float64
62	R_NH4	64982	non-null	float64
63	R_CHLA	225276	non-null	float64
64	R_PHAE0	225275	non-null	float64
65	R_PRES	864863	non-null	int64

```

66 R_SAMP          122006 non-null   float64
67 DIC1           1999 non-null   float64
68 DIC2           224 non-null    float64
69 TA1            2084 non-null   float64
70 TA2            234 non-null    float64
71 pH2            10 non-null    float64
72 pH1            84 non-null    float64
73 DIC Quality Comment 55 non-null   object
dtypes: float64(65), int64(5), object(4)
memory usage: 488.3+ MB

```

In [6]:

```
df.describe()
```

Out[6]:

	Cst_Cnt	Btl_Cnt	Depthm	T_degC	Salnty	O2ml_L	
<b>count</b>	864863.000000	864863.000000	864863.000000	853900.000000	817509.000000	696201.000000	81217
<b>mean</b>	17138.790958	432432.000000	226.831951	10.799677	33.840350	3.392468	2
<b>std</b>	10240.949817	249664.587267	316.050259	4.243825	0.461843	2.073256	
<b>min</b>	1.000000	1.000000	0.000000	1.440000	28.431000	-0.010000	2
<b>25%</b>	8269.000000	216216.500000	46.000000	7.680000	33.488000	1.360000	2
<b>50%</b>	16848.000000	432432.000000	125.000000	10.060000	33.863000	3.440000	2
<b>75%</b>	26557.000000	648647.500000	300.000000	13.880000	34.196900	5.500000	2
<b>max</b>	34404.000000	864863.000000	5351.000000	31.140000	37.034000	11.130000	25

8 rows × 70 columns



In [7]:

```
a = df.dropna(axis='columns')
a
```

Out[7]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	Reclnd	R_Depth	R_PRES
<b>0</b>	1	1	054.0 056.0	19-4903CR-HY-060-0930- 05400560-0000A-3	0	3	0.0	0
<b>1</b>	1	2	054.0 056.0	19-4903CR-HY-060-0930- 05400560-0008A-3	8	3	8.0	8
<b>2</b>	1	3	054.0 056.0	19-4903CR-HY-060-0930- 05400560-0010A-7	10	7	10.0	10
<b>3</b>	1	4	054.0 056.0	19-4903CR-HY-060-0930- 05400560-0019A-3	19	3	19.0	19
<b>4</b>	1	5	054.0 056.0	19-4903CR-HY-060-0930- 05400560-0020A-7	20	7	20.0	20
...	...	...	...	...	...	...	...	...
<b>864858</b>	34404	864859	093.4 026.4	20-1611SR-MX-310-2239- 09340264-0000A-7	0	7	0.0	0
<b>864859</b>	34404	864860	093.4 026.4	20-1611SR-MX-310-2239- 09340264-0002A-3	2	3	2.0	2

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	Reclnd	R_Depth	R_PRES
<b>864860</b>	34404	864861	093.4 026.4	20-1611SR-MX-310-2239- 09340264-0005A-3	5	3	5.0	5
<b>864861</b>	34404	864862	093.4 026.4	20-1611SR-MX-310-2239- 09340264-0010A-3	10	3	10.0	10
<b>864862</b>	34404	864863	093.4 026.4	20-1611SR-MX-310-2239- 09340264-0015A-3	15	3	15.0	15

864863 rows × 8 columns

In [8]: `a.columns`

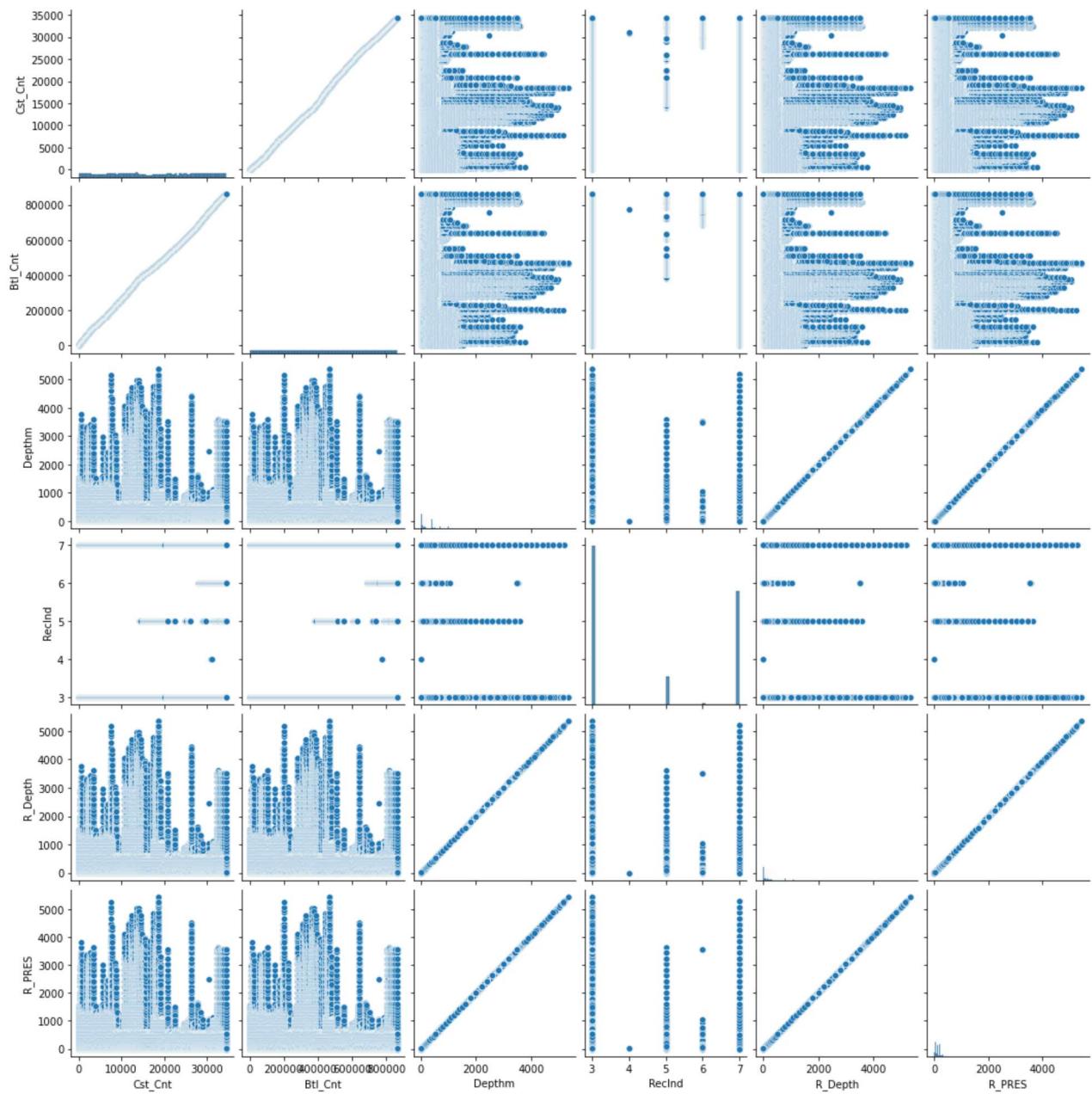
Out[8]: `Index(['Cst_Cnt', 'Btl_Cnt', 'Sta_ID', 'Depth_ID', 'Depthm', 'RecInd', 'R_Depth', 'R_PRES'], dtype='object')`

## EDA and VISUALIZATION

In [9]: `sns.pairplot(a)`

Out[9]: `<seaborn.axisgrid.PairGrid at 0x2e38f077970>`

## Linear Regression-Bottle

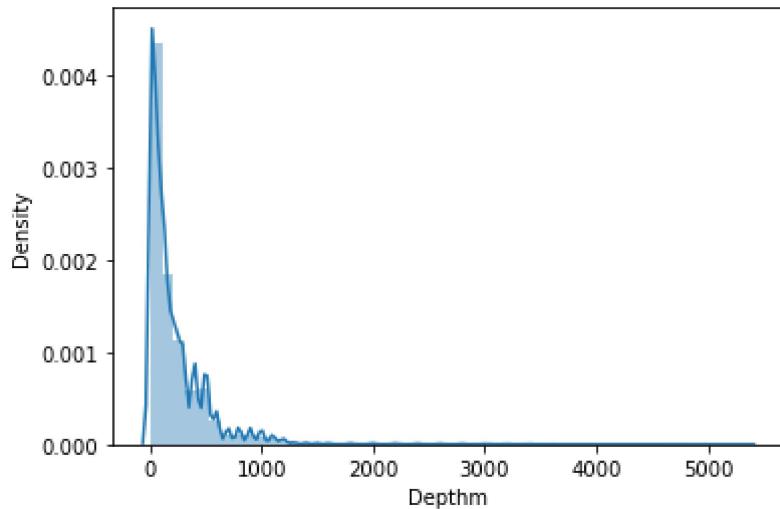


```
In [10]: sns.distplot(a["Depthm"])
```

```
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 function for histograms).
```

```
    warnings.warn(msg, FutureWarning)
```

```
Out[10]: <AxesSubplot:xlabel='Depthm', ylabel='Density'>
```

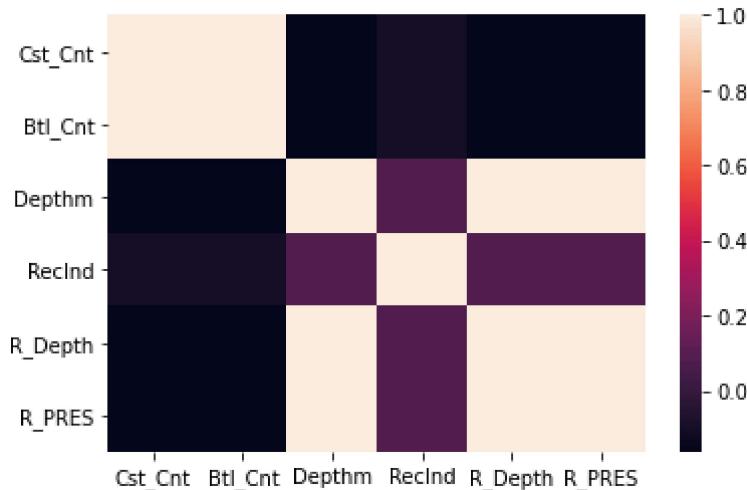


```
In [11]: df1=a[['Cst_Cnt', 'Btl_Cnt', 'Sta_ID', 'Depth_ID', 'Depthm', 'RecInd', 'R_Depth', 'R_PRES']]
```

## Plot Using Heat Map

```
In [12]: sns.heatmap(df1.corr())
```

```
Out[12]: <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 [13]: x=df1[['Cst_Cnt', 'Btl_Cnt', 'RecInd', 'R_Depth', ]]
y=df1['Depthm']
```

# To Split my dataset into training and test data

In [14]:

```
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 [15]:

```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]:

```
LinearRegression()
```

In [16]:

```
lr.intercept_
```

Out[16]:

```
0.00323805872506
```

In [17]:

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

Out[17]:

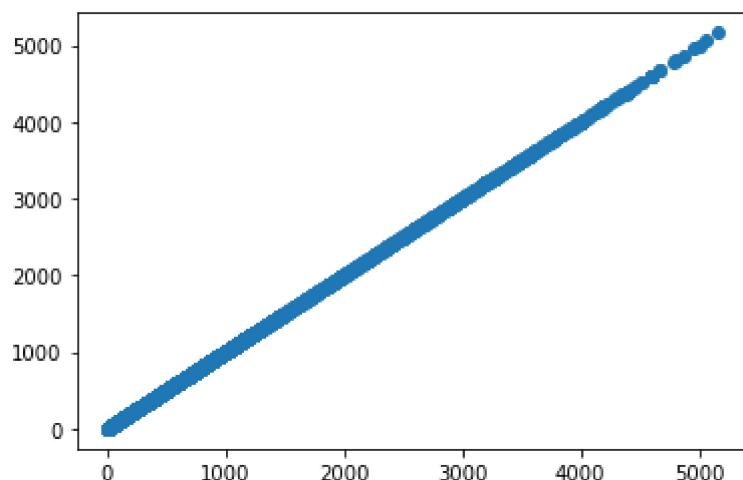
	Co-efficient
<b>Cst_Cnt</b>	1.733398e-06
<b>Btl_Cnt</b>	-7.481548e-08
<b>Reclnd</b>	-2.628260e-04
<b>R_Depth</b>	1.000000e+00

In [18]:

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

Out[18]:

```
<matplotlib.collections.PathCollection at 0x2e3b8e33370>
```



In [19]:

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

```
Out[19]: 0.9999999946812042
```

```
In [20]: from sklearn.linear_model import Ridge,Lasso
```

```
In [21]: rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

```
Out[21]: 0.999999945318518
```

```
In [22]: rr.score(x_test,y_test)
```

```
Out[22]: 0.9999999946812042
```

```
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]: 0.9999999838291133
```

```
In [ ]:
```

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

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

```
In [28]: print(en.coef_)
```

```
[-1.71574416e-05  7.02656578e-07 -0.00000000e+00  9.99990197e-01]
```

```
In [30]: print(en.intercept_)
```

```
-0.008115961125838567
```

```
In [33]: print(en.predict(x_test))
```

```
[ 2.75989035e+02  2.30006228e+02  4.14009539e+02 ...  1.00051557e+01
 7.60985905e+02 -1.33518082e-03]
```

```
In [34]: print(en.score(x_test,y_test))
```

0.999999994087528

# Evaluation Metrics

```
In [35]: from sklearn import metrics
```

```
In [37]: print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

Mean Absolute Error: 0.0015793850103647687

```
In [39]: print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

Mean Squared Error: 0.0005378247043309956

```
In [46]: print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

Root Mean Squared Error: 0.0231910479351623