A real estate agent want 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

Data Collection

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

In [2]:
```
#import the dataset
data=pd.read_csv(r"C:\Users\user\Desktop\Vicky\9_bottle.csv")[0:500]
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3
165: DtypeWarning: Columns (47,73) have mixed types.Specify dtype option on i
mport or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [3]:
```
#to display top 10 rows
data.head()
```

Out[3]:

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

5 rows × 74 columns

In [4]:
```
#to display null values
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 74 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Cst_Cnt              500 non-null    int64  
 1   Btl_Cnt              500 non-null    int64  
 2   Sta_ID               500 non-null    object 
 3   Depth_ID             500 non-null    object 
 4   Depthm               500 non-null    int64  
 5   T_degC               499 non-null    float64
 6   Salnty               494 non-null    float64
 7   O2ml_L               0 non-null      float64
 8   STheta               493 non-null    float64
 9   O2Sat                0 non-null      float64
 10  Oxy_µmol/Kg          0 non-null      float64
 11  BtlNum               0 non-null      float64
 12  RecInd               500 non-null    int64  
 13  T_prec               499 non-null    float64
 14  T_qual               4 non-null      float64
 15  S_prec               494 non-null    float64
 16  S_qual               10 non-null     float64
 17  P_qual               500 non-null    float64
 18  O_qual               500 non-null    float64
 19  SThtaq               14 non-null     float64
 20  O2Satq               500 non-null    float64
 21  ChlorA               0 non-null      float64
 22  Chlqua               500 non-null    float64
 23  Phaeop               0 non-null      float64
 24  Phaqua               500 non-null    float64
 25  PO4uM                0 non-null      float64
 26  PO4q                 500 non-null    float64
 27  SiO3uM               0 non-null      float64
 28  SiO3qu               500 non-null    float64
 29  NO2uM                0 non-null      float64
 30  NO2q                 500 non-null    float64
 31  NO3uM                0 non-null      float64
 32  NO3q                 500 non-null    float64
 33  NH3uM                0 non-null      float64
 34  NH3q                 500 non-null    float64
 35  C14As1               0 non-null      float64
 36  C14A1p               0 non-null      float64
 37  C14A1q               500 non-null    float64
 38  C14As2               0 non-null      float64
 39  C14A2p               0 non-null      float64
 40  C14A2q               500 non-null    float64
 41  DarkAs               0 non-null      float64
 42  DarkAp               0 non-null      float64
 43  DarkAq               500 non-null    float64
 44  MeanAs               0 non-null      float64
 45  MeanAp               0 non-null      float64
 46  MeanAq               500 non-null    float64
 47  IncTim               0 non-null      object 
 48  LightP               0 non-null      float64
 49  R_Depth              500 non-null    float64
 50  R_TEMP               499 non-null    float64
 51  R_POTEMP             495 non-null    float64
```

```
 52  R_SALINITY           494 non-null    float64
 53  R_SIGMA              486 non-null    float64
 54  R_SVA                486 non-null    float64
 55  R_DYNHT              500 non-null    float64
 56  R_O2                 0 non-null      float64
 57  R_O2Sat              0 non-null      float64
 58  R_SIO3               0 non-null      float64
 59  R_PO4                0 non-null      float64
 60  R_NO3                0 non-null      float64
 61  R_NO2                0 non-null      float64
 62  R_NH4                0 non-null      float64
 63  R_CHLA               0 non-null      float64
 64  R_PHAEO              0 non-null      float64
 65  R_PRES               500 non-null    int64
 66  R_SAMP               0 non-null      float64
 67  DIC1                 0 non-null      float64
 68  DIC2                 0 non-null      float64
 69  TA1                  0 non-null      float64
 70  TA2                  0 non-null      float64
 71  pH2                  0 non-null      float64
 72  pH1                  0 non-null      float64
 73  DIC Quality Comment  0 non-null      object
dtypes: float64(65), int64(5), object(4)
memory usage: 289.2+ KB
```

In [5]:
```
#to display summary of statistics
data.describe()
```

Out[5]:

	Cst_Cnt	Btl_Cnt	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat
count	500.000000	500.000000	500.000000	499.000000	494.000000	0.0	493.000000	0.0
mean	8.548000	250.500000	341.490000	7.850421	33.628842	NaN	26.183400	NaN
std	4.570062	144.481833	355.166886	2.911584	0.560411	NaN	0.846325	NaN
min	1.000000	1.000000	0.000000	2.780000	32.630000	NaN	24.870000	NaN
25%	5.000000	125.750000	55.000000	5.030000	33.071000	NaN	25.259000	NaN
50%	9.000000	250.500000	200.000000	8.180000	33.799500	NaN	26.339000	NaN
75%	12.250000	375.250000	598.500000	10.450000	34.130000	NaN	26.983000	NaN
max	16.000000	500.000000	1352.000000	12.660000	34.450000	NaN	27.450000	NaN

8 rows × 70 columns

In [6]:
```
#to display columns name
data.columns
```

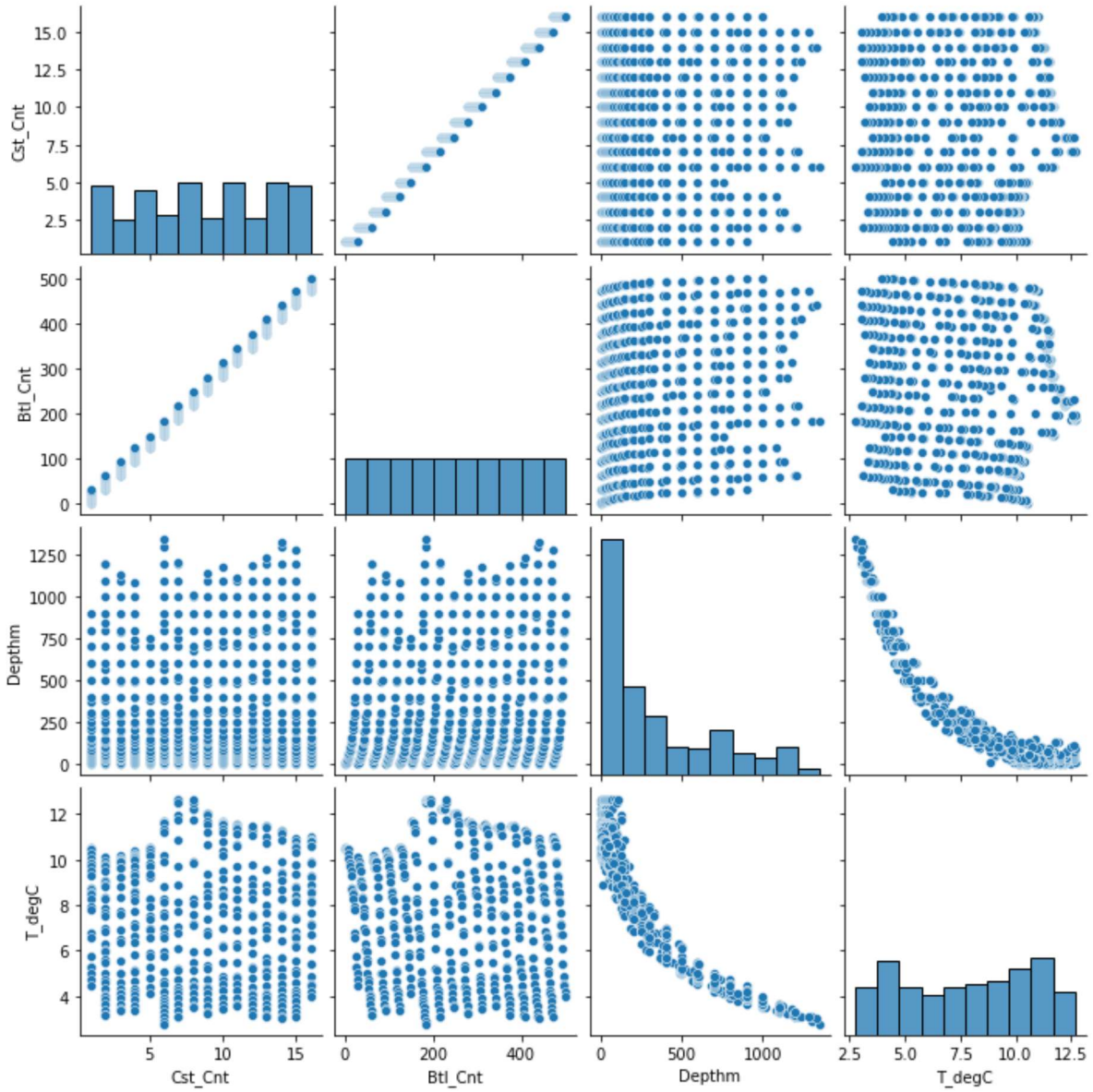
Out[6]:
```
Index(['Cst_Cnt', 'Btl_Cnt', 'Sta_ID', 'Depth_ID', 'Depthm', 'T_degC',
       'Salnty', 'O2ml_L', 'STheta', 'O2Sat', 'Oxy_µmol/Kg', 'BtlNum',
       'RecInd', 'T_prec', 'T_qual', 'S_prec', 'S_qual', 'P_qual', 'O_qual',
       'SThtaq', 'O2Satq', 'ChlorA', 'Chlqua', 'Phaeop', 'Phaqua', 'PO4uM',
       'PO4q', 'SiO3uM', 'SiO3qu', 'NO2uM', 'NO2q', 'NO3uM', 'NO3q', 'NH3uM',
       'NH3q', 'C14As1', 'C14A1p', 'C14A1q', 'C14As2', 'C14A2p', 'C14A2q',
       'DarkAs', 'DarkAp', 'DarkAq', 'MeanAs', 'MeanAp', 'MeanAq', 'IncTim',
       'LightP', 'R_Depth', 'R_TEMP', 'R_POTEMP', 'R_SALINITY', 'R_SIGMA',
       'R_SVA', 'R_DYNHT', 'R_O2', 'R_O2Sat', 'R_SIO3', 'R_PO4', 'R_NO3',
       'R_NO2', 'R_NH4', 'R_CHLA', 'R_PHAEO', 'R_PRES', 'R_SAMP', 'DIC1',
       'DIC2', 'TA1', 'TA2', 'pH2', 'pH1', 'DIC Quality Comment'],
      dtype='object')
```

In [7]:
```
data1=data[['Cst_Cnt', 'Btl_Cnt', 'Sta_ID', 'Depth_ID', 'Depthm', 'T_degC']]
```

EDA and Visualization

In [8]:
```
sns.pairplot(data1)
```

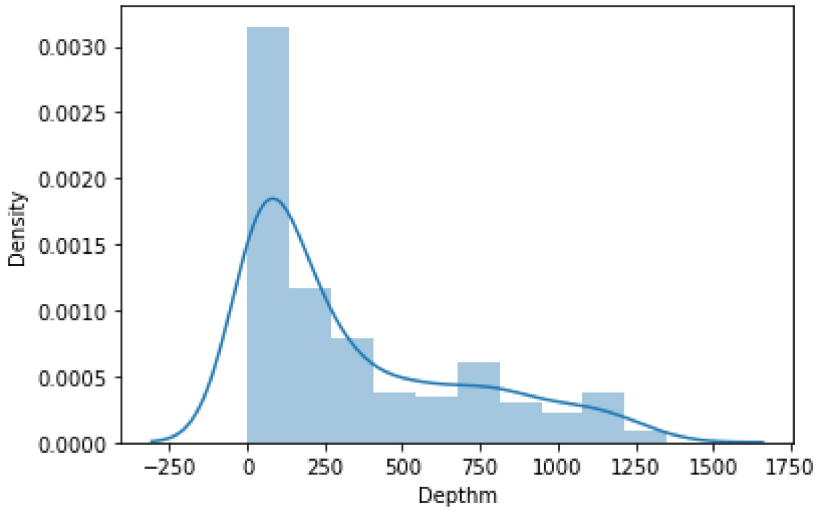
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1760abcd670>
```



In [10]:
```
sns.distplot(data['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[10]: <AxesSubplot:xlabel='Depthm', ylabel='Density'>

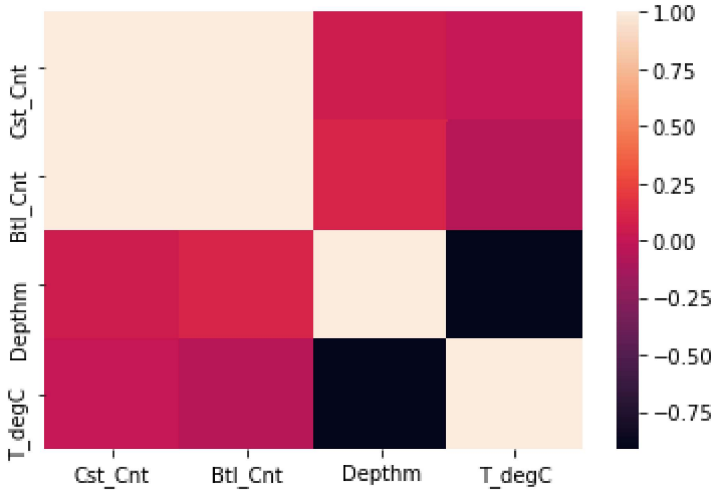


In [ ]:

In [ ]:

In [11]:
```
sns.heatmap(data1.corr())
```

Out[11]: <AxesSubplot:>



To train the model

we are going to train the linear regression model ;We need to split the two variable x and y where x in independent variable (input) and y is dependent of x(output) so we could ignore

In [15]:
```
x=data1[[ 'Btl_Cnt',  'Depthm', ]]
y=data1['Cst_Cnt']
```

In [16]:
```
#To split test and train data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.6)
```

In [17]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[17]:
```
LinearRegression()
```

In [18]:
```
lr.intercept_
```

Out[18]:
```
0.8439017302552436
```

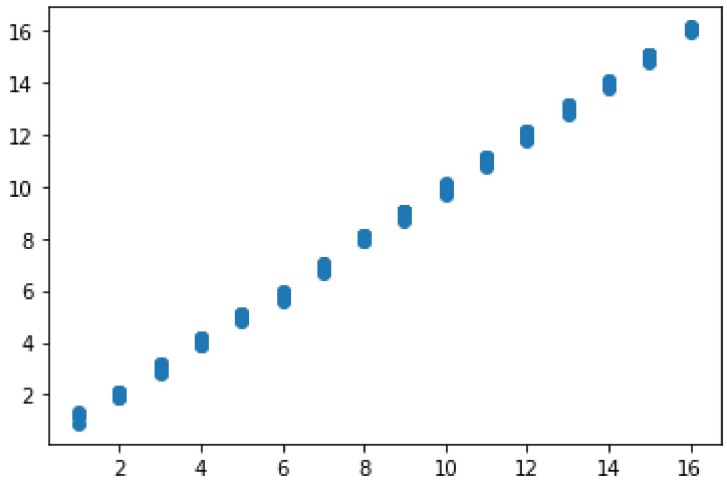
In [19]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[19]:

	Co-efficient
Btl_Cnt	0.031703
Depthm	-0.000710

In [20]:
```
prediction = lr.predict(x_train)
plt.scatter(y_train,prediction)
```

Out[20]:
```
<matplotlib.collections.PathCollection at 0x1760c9b57f0>
```



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

Out[21]:
```
0.9992384112418674
```

In [ ]:
```

```

In [ ]:
```

```