PROJECT DEVELOPMENT PHASE
SPRINT II

Assignment Date	06-10-2022
Team ID	PNT2022TMID19759
Project Name	Efficient Water Quality Analysis and Prediction using Machine Learning
Maximum Marks	8 Marks

DATA PRE-PROCESSING

1. Importing Required Package:

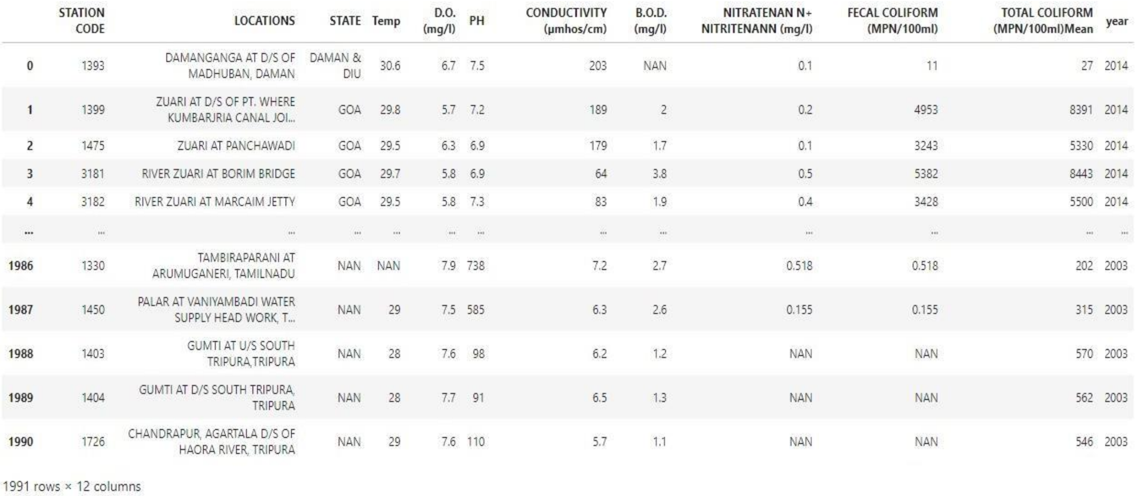
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
```

2. Upload dataset

```
 Data= pd.read_csv
(r"C:\Users\karthick\Desktop\water_dataX.csv",encoding='ISO-8859-1')
```

3.

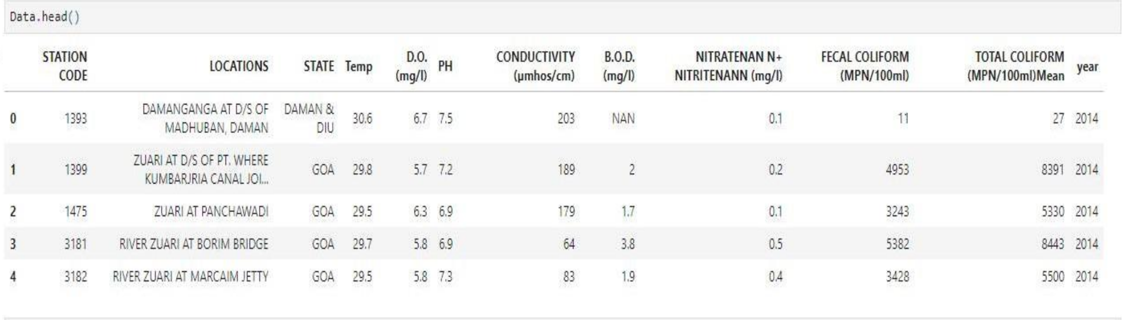
Data



	STATION CODE	LOCATIONS	STATE	Temp	D.O. (mg/l)	PH	CONDUCTIVITY (µmhos/cm)	B.O.D. (mg/l)	NITRATENAN N+ NITRITENANN (mg/l)	FECAL COLIFORM (MPN/100ml)	TOTAL COLIFORM (MPN/100ml)Mean	year
0	1393	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.6	6.7	7.5	203	NAN	0.1	11	27	2014
1	1399	ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...	GOA	29.8	5.7	7.2	189	2	0.2	4953	8391	2014
2	1475	ZUARI AT PANCHAWADI	GOA	29.5	6.3	6.9	179	1.7	0.1	3243	5330	2014
3	3181	RIVER ZUARI AT BORIM BRIDGE	GOA	29.7	5.8	6.9	64	3.8	0.5	5382	8443	2014
4	3182	RIVER ZUARI AT MARCAIM JETTY	GOA	29.5	5.8	7.3	83	1.9	0.4	3428	5500	2014
...	...	...	...	...	...	...	...	...	...	...	...	...
1986	1330	TAMBIRAPARANI AT ARUMUGANERI, TAMILNADU	NAN	NAN	7.9	738	7.2	2.7	0.518	0.518	202	2003
1987	1450	PALAR AT VANIYAMBADI WATER SUPPLY HEAD WORK, T...	NAN	29	7.5	585	6.3	2.6	0.155	0.155	315	2003
1988	1403	GUMTI AT U/S SOUTH TRIPURA,TRIPURA	NAN	28	7.6	98	6.2	1.2	NAN	NAN	570	2003
1989	1404	GUMTI AT D/S SOUTH TRIPURA, TRIPURA	NAN	28	7.7	91	6.5	1.3	NAN	NAN	562	2003
1990	1726	CHANDRAPUR, AGARTALA D/S OF HAORA RIVER, TRIPURA	NAN	29	7.6	110	5.7	1.1	NAN	NAN	546	2003

1991 rows × 12 columns

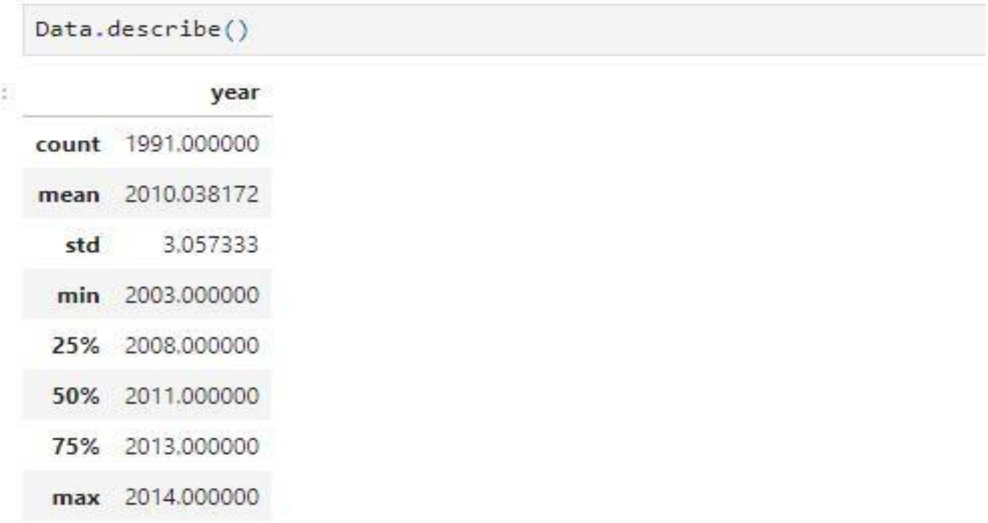
4.



```
Data.head()
```

	STATION CODE	LOCATIONS	STATE	Temp	D.O. (mg/l)	PH	CONDUCTIVITY (µmhos/cm)	B.O.D. (mg/l)	NITRATENAN N+ NITRITENANN (mg/l)	FECAL COLIFORM (MPN/100ml)	TOTAL COLIFORM (MPN/100ml)Mean	year
0	1393	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.6	6.7	7.5	203	NAN	0.1	11	27	2014
1	1399	ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...	GOA	29.8	5.7	7.2	189	2	0.2	4953	8391	2014
2	1475	ZUARI AT PANCHAWADI	GOA	29.5	6.3	6.9	179	1.7	0.1	3243	5330	2014
3	3181	RIVER ZUARI AT BORIM BRIDGE	GOA	29.7	5.8	6.9	64	3.8	0.5	5382	8443	2014
4	3182	RIVER ZUARI AT MARCAIM JETTY	GOA	29.5	5.8	7.3	83	1.9	0.4	3428	5500	2014

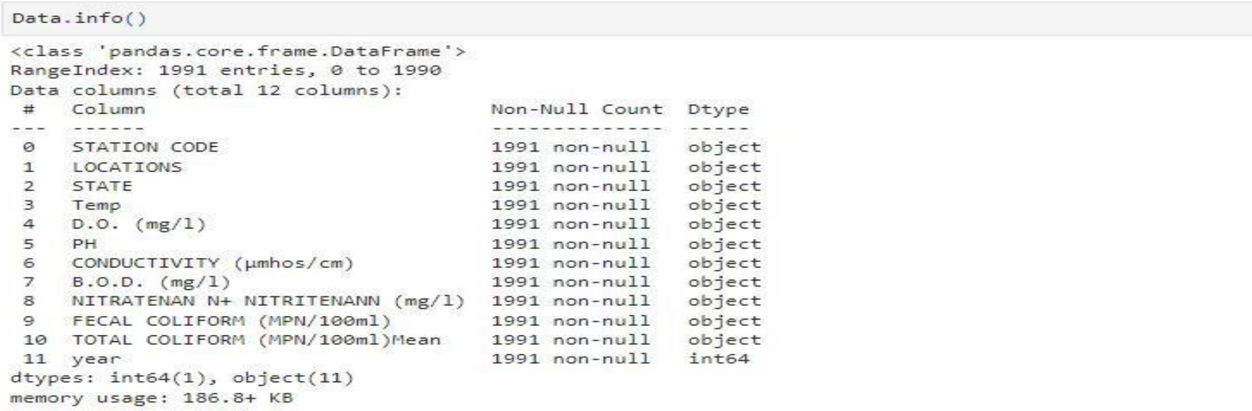
5.



```
Data.describe()
```

	year
count	1991.000000
mean	2010.038172
std	3.057333
min	2003.000000
25%	2008.000000
50%	2011.000000
75%	2013.000000
max	2014.000000

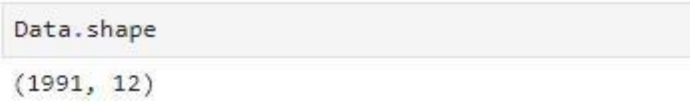
6.



```
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1991 entries, 0 to 1990
Data columns (total 12 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   STATION CODE                        1991 non-null   object
 1   LOCATIONS                           1991 non-null   object
 2   STATE                               1991 non-null   object
 3   Temp                                1991 non-null   object
 4   D.O. (mg/l)                         1991 non-null   object
 5   PH                                  1991 non-null   object
 6   CONDUCTIVITY (µmhos/cm)             1991 non-null   object
 7   B.O.D. (mg/l)                       1991 non-null   object
 8   NITRATENAN N+ NITRITENANN (mg/l)    1991 non-null   object
 9   FECAL COLIFORM (MPN/100ml)          1991 non-null   object
 10  TOTAL COLIFORM (MPN/100ml)Mean      1991 non-null   object
 11  year                                1991 non-null   int64
dtypes: int64(1), object(11)
memory usage: 186.8+ KB
```

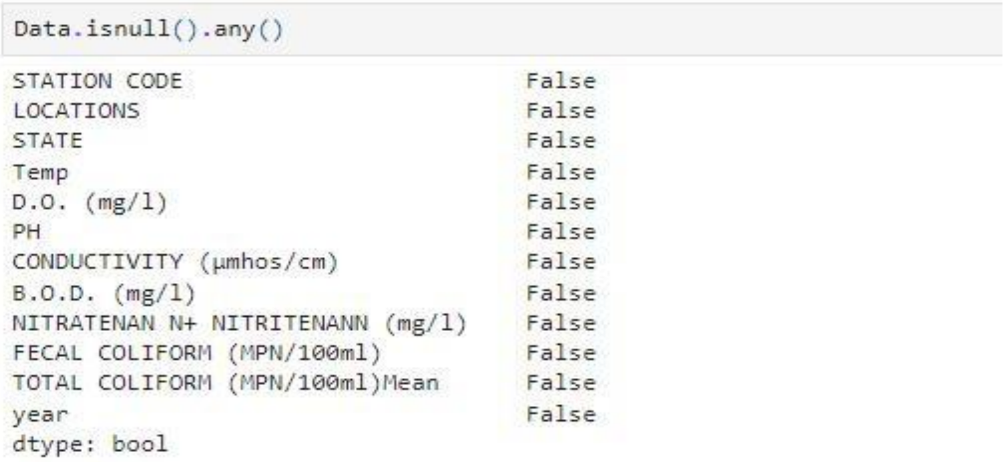
7.



```
Data.shape
```

```
(1991, 12)
```

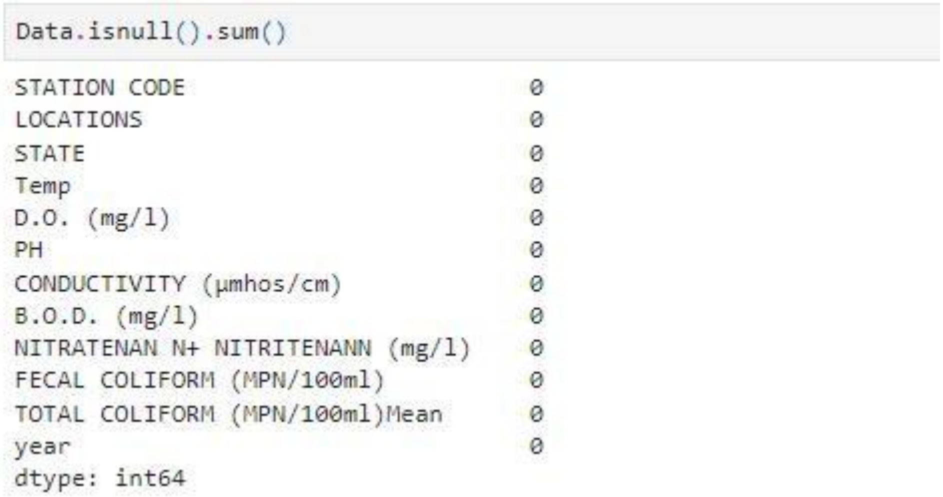
8.



```
Data.isnull().any()
```

```
STATION CODE                        False
LOCATIONS                           False
STATE                               False
Temp                                False
D.O. (mg/l)                         False
PH                                  False
CONDUCTIVITY (µmhos/cm)             False
B.O.D. (mg/l)                       False
NITRATENAN N+ NITRITENANN (mg/l)    False
FECAL COLIFORM (MPN/100ml)          False
TOTAL COLIFORM (MPN/100ml)Mean      False
year                                False
dtype: bool
```

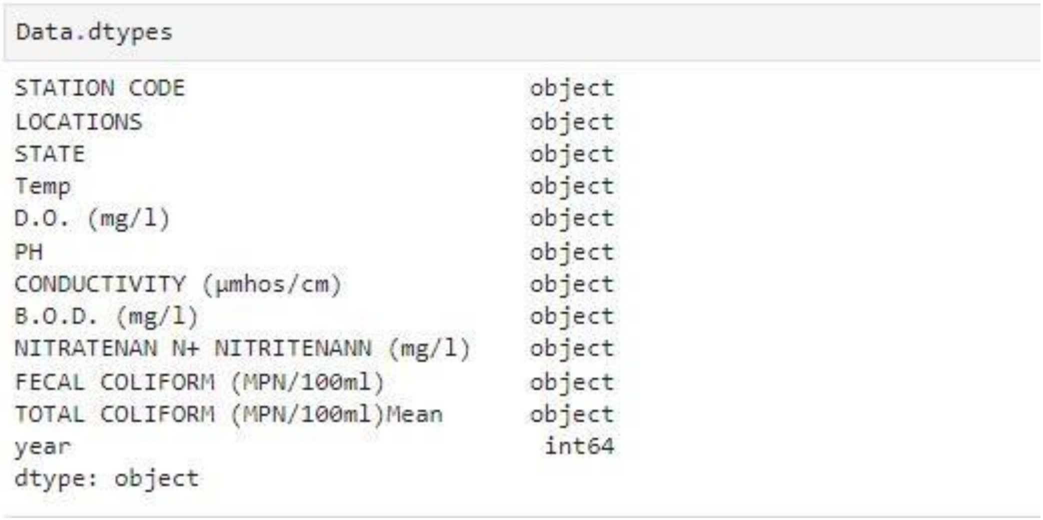
9.



```
Data.isnull().sum()
```

```
STATION CODE                          0
LOCATIONS                             0
STATE                                 0
Temp                                  0
D.O. (mg/l)                           0
PH                                    0
CONDUCTIVITY (µmhos/cm)               0
B.O.D. (mg/l)                         0
NITRATENAN N+ NITRITENANN (mg/l)      0
FECAL COLIFORM (MPN/100ml)            0
TOTAL COLIFORM (MPN/100ml)Mean        0
year                                  0
dtype: int64
```

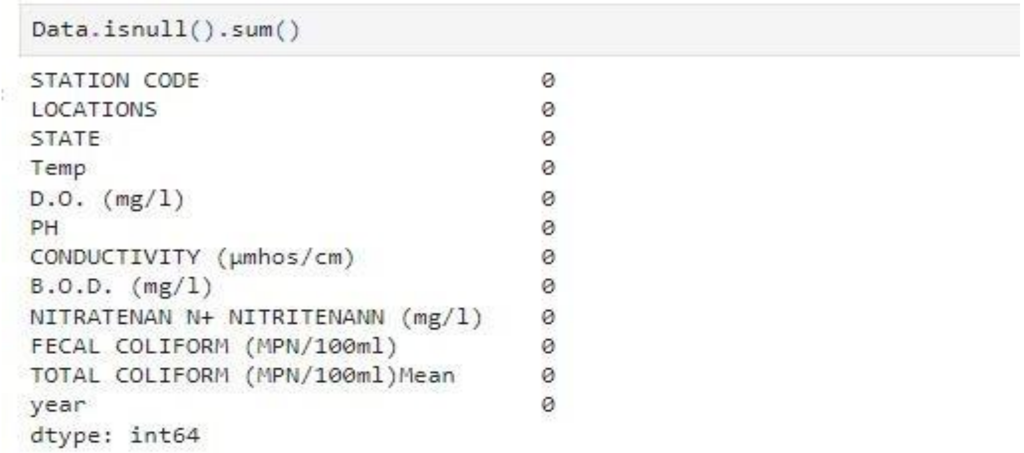
10.



```
Data.dtypes
```

```
STATION CODE                        object
LOCATIONS                           object
STATE                               object
Temp                                object
D.O. (mg/l)                         object
PH                                  object
CONDUCTIVITY (µmhos/cm)             object
B.O.D. (mg/l)                       object
NITRATENAN N+ NITRITENANN (mg/l)    object
FECAL COLIFORM (MPN/100ml)          object
TOTAL COLIFORM (MPN/100ml)Mean      object
year                                 int64
dtype: object
```

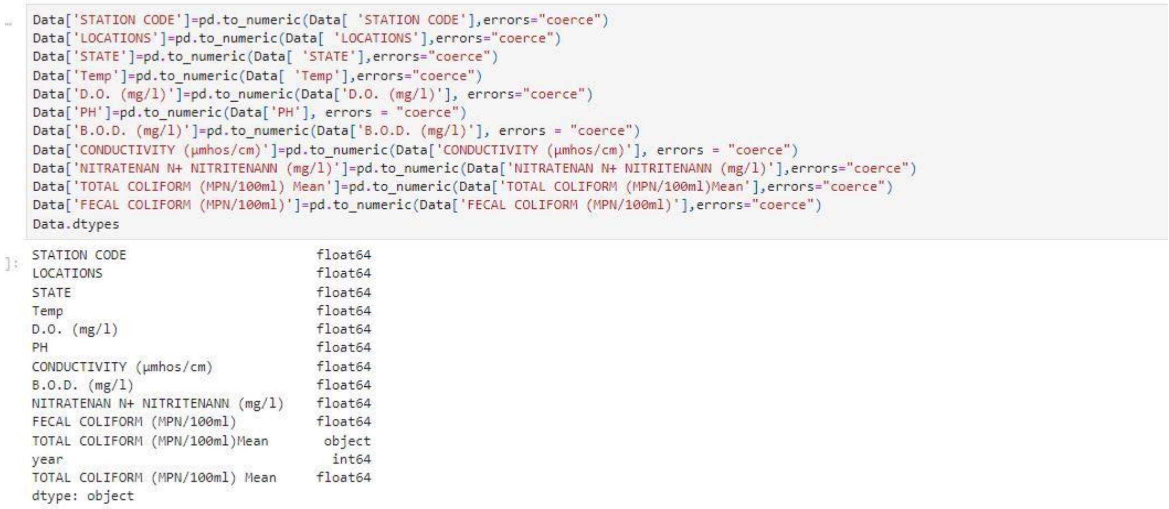
11.



```
Data.isnull().sum()
```

```
STATION CODE                          0
LOCATIONS                             0
STATE                                 0
Temp                                  0
D.O. (mg/l)                           0
PH                                    0
CONDUCTIVITY (µmhos/cm)               0
B.O.D. (mg/l)                         0
NITRATENAN N+ NITRITENANN (mg/l)      0
FECAL COLIFORM (MPN/100ml)            0
TOTAL COLIFORM (MPN/100ml)Mean        0
year                                  0
dtype: int64
```

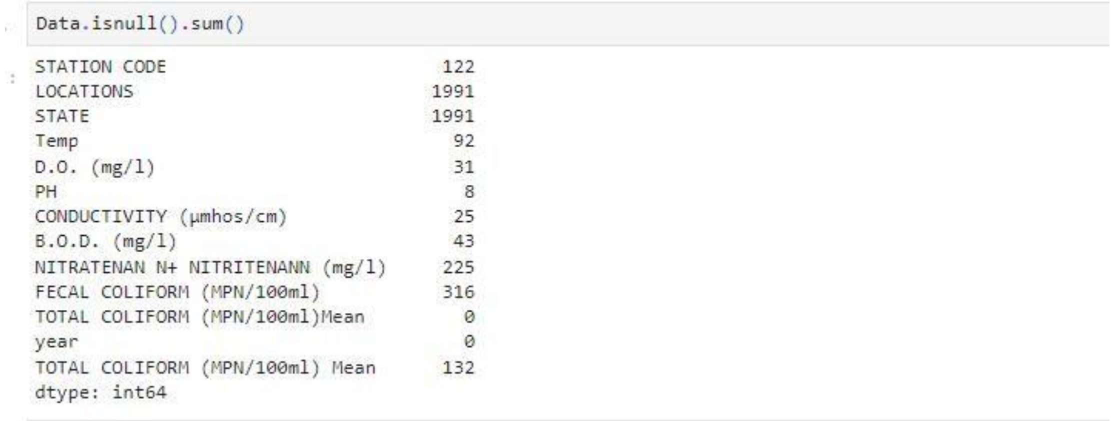
12.



```
Data['STATION CODE']=pd.to_numeric(Data[ 'STATION CODE'],errors="coerce")
Data['LOCATIONS']=pd.to_numeric(Data[ 'LOCATIONS'],errors="coerce")
Data['STATE']=pd.to_numeric(Data[ 'STATE'],errors="coerce")
Data['Temp']=pd.to_numeric(Data[ 'Temp'],errors="coerce")
Data['D.O. (mg/l)']=pd.to_numeric(Data['D.O. (mg/l)'], errors="coerce")
Data['PH']=pd.to_numeric(Data['PH'], errors = "coerce")
Data['B.O.D. (mg/l)']=pd.to_numeric(Data['B.O.D. (mg/l)'], errors = "coerce")
Data['CONDUCTIVITY (µmhos/cm)']=pd.to_numeric(Data['CONDUCTIVITY (µmhos/cm)'], errors = "coerce")
Data['NITRATENAN N+ NITRITENANN (mg/l)']=pd.to_numeric(Data['NITRATENAN N+ NITRITENANN (mg/l)'],errors="coerce")
Data['TOTAL COLIFORM (MPN/100ml) Mean']=pd.to_numeric(Data['TOTAL COLIFORM (MPN/100ml)Mean'],errors="coerce")
Data['FECAL COLIFORM (MPN/100ml)']=pd.to_numeric(Data['FECAL COLIFORM (MPN/100ml)'],errors="coerce")
Data.dtypes
```

```
STATION CODE                        float64
LOCATIONS                           float64
STATE                               float64
Temp                                float64
D.O. (mg/l)                         float64
PH                                  float64
CONDUCTIVITY (µmhos/cm)             float64
B.O.D. (mg/l)                       float64
NITRATENAN N+ NITRITENANN (mg/l)    float64
FECAL COLIFORM (MPN/100ml)          float64
TOTAL COLIFORM (MPN/100ml)Mean       object
year                                  int64
TOTAL COLIFORM (MPN/100ml) Mean     float64
dtype: object
```

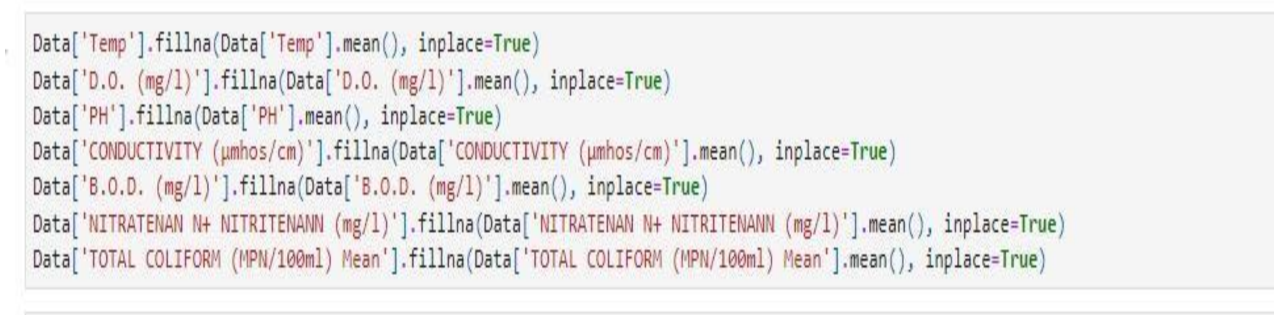
13.



```
Data.isnull().sum()
```

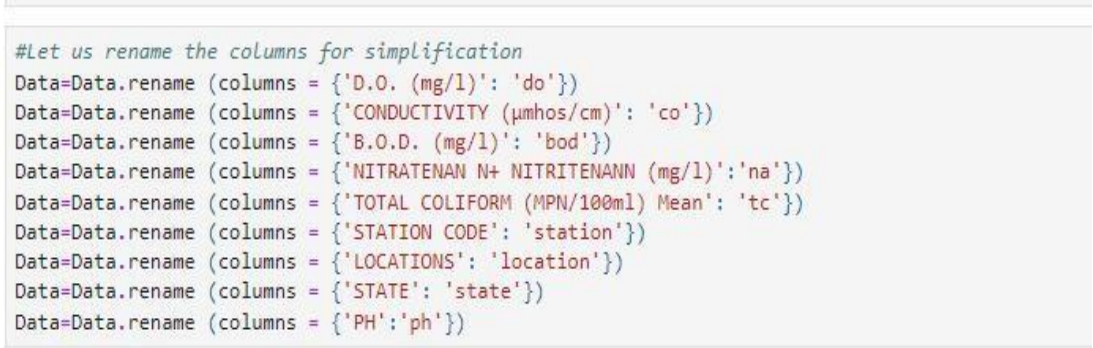
```
STATION CODE                         122
LOCATIONS                           1991
STATE                               1991
Temp                                  92
D.O. (mg/l)                           31
PH                                     8
CONDUCTIVITY (µmhos/cm)               25
B.O.D. (mg/l)                         43
NITRATENAN N+ NITRITENANN (mg/l)     225
FECAL COLIFORM (MPN/100ml)           316
TOTAL COLIFORM (MPN/100ml)Mean         0
year                                   0
TOTAL COLIFORM (MPN/100ml) Mean      132
dtype: int64
```

14.



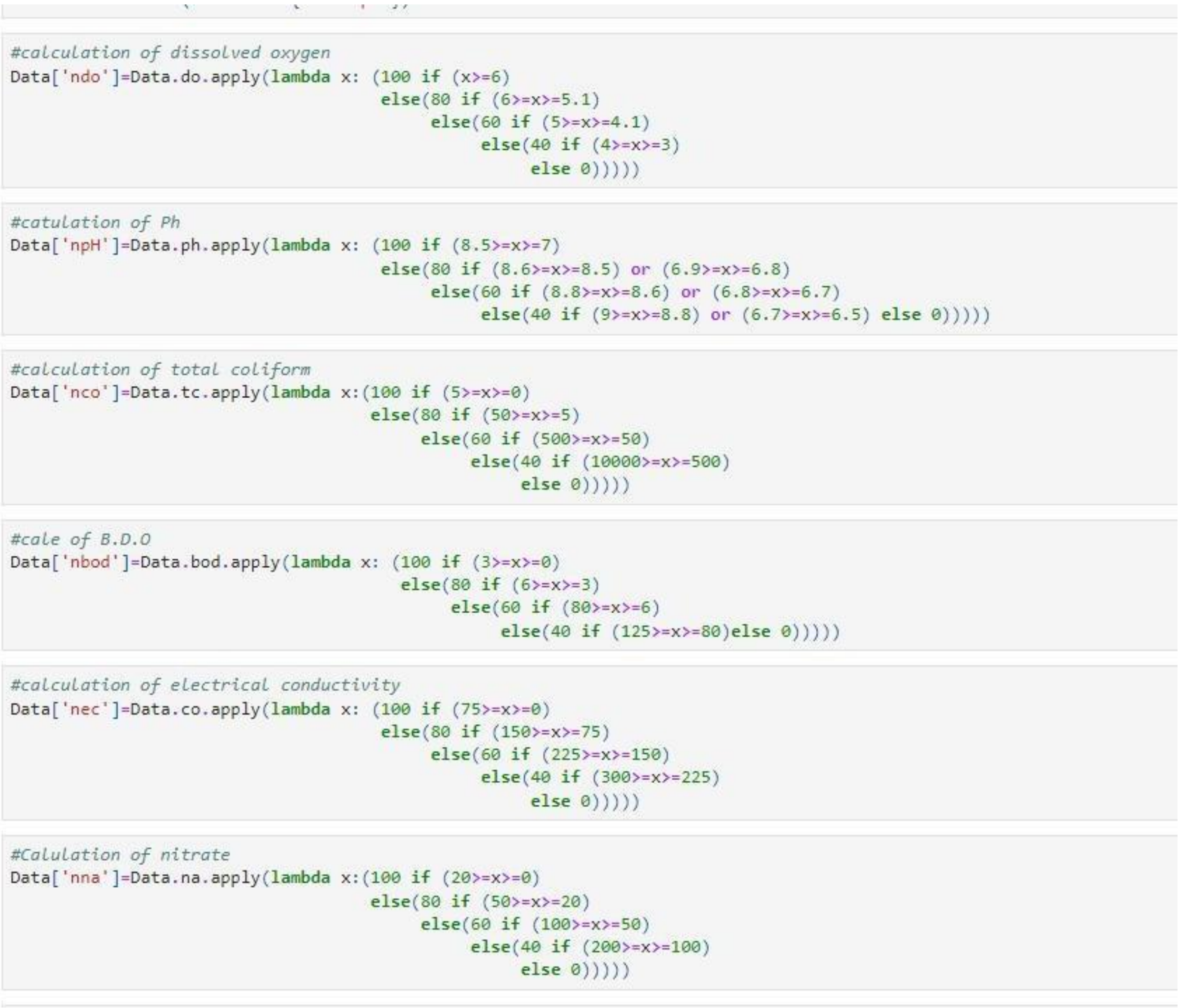
```
Data['Temp'].fillna(Data['Temp'].mean(), inplace=True)
Data['D.O. (mg/l)'].fillna(Data['D.O. (mg/l)'].mean(), inplace=True)
Data['PH'].fillna(Data['PH'].mean(), inplace=True)
Data['CONDUCTIVITY (µmhos/cm)'].fillna(Data['CONDUCTIVITY (µmhos/cm)'].mean(), inplace=True)
Data['B.O.D. (mg/l)'].fillna(Data['B.O.D. (mg/l)'].mean(), inplace=True)
Data['NITRATENAN N+ NITRITENANN (mg/l)'].fillna(Data['NITRATENAN N+ NITRITENANN (mg/l)'].mean(), inplace=True)
Data['TOTAL COLIFORM (MPN/100ml) Mean'].fillna(Data['TOTAL COLIFORM (MPN/100ml) Mean'].mean(), inplace=True)
```

15.



```
#Let us rename the columns for simplification
Data=Data.rename (columns = {'D.O. (mg/l)': 'do'})
Data=Data.rename (columns = {'CONDUCTIVITY (µmhos/cm)': 'co'})
Data=Data.rename (columns = {'B.O.D. (mg/l)': 'bod'})
Data=Data.rename (columns = {'NITRATENAN N+ NITRITENANN (mg/l)':'na'})
Data=Data.rename (columns = {'TOTAL COLIFORM (MPN/100ml) Mean': 'tc'})
Data=Data.rename (columns = {'STATION CODE': 'station'})
Data=Data.rename (columns = {'LOCATIONS': 'location'})
Data=Data.rename (columns = {'STATE': 'state'})
Data=Data.rename (columns = {'PH':'ph'})
```

16.



```
#calculation of dissolved oxygen
Data['ndo']=Data.do.apply(lambda x: (100 if (x>=6)
                                     else(80 if (6>=x>=5.1)
                                          else(60 if (5>=x>=4.1)
                                               else(40 if (4>=x>=3)
                                                    else 0)))))
```

```
#catulation of Ph
Data['npH']=Data.ph.apply(lambda x: (100 if (8.5>=x>=7)
                                     else(80 if (8.6>=x>=8.5) or (6.9>=x>=6.8)
                                          else(60 if (8.8>=x>=8.6) or (6.8>=x>=6.7)
                                               else(40 if (9>=x>=8.8) or (6.7>=x>=6.5) else 0)))))
```

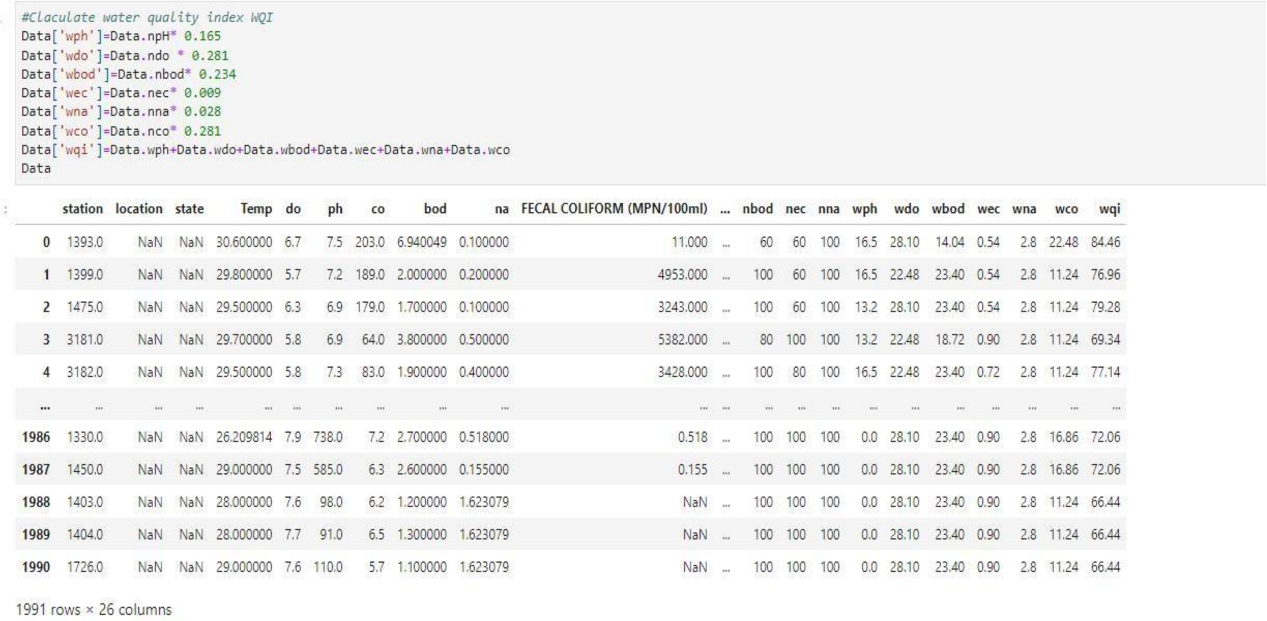
```
#calculation of total coliform
Data['nco']=Data.tc.apply(lambda x:(100 if (5>=x>=0)
                                    else(80 if (50>=x>=5)
                                         else(60 if (500>=x>=50)
                                              else(40 if (10000>=x>=500)
                                                   else 0)))))
```

```
#calc of B.D.O
Data['nbod']=Data.bod.apply(lambda x: (100 if (3>=x>=0)
                                       else(80 if (6>=x>=3)
                                            else(60 if (80>=x>=6)
                                                 else(40 if (125>=x>=80)else 0)))))
```

```
#calculation of electrical conductivity
Data['nec']=Data.co.apply(lambda x: (100 if (75>=x>=0)
                                     else(80 if (150>=x>=75)
                                          else(60 if (225>=x>=150)
                                               else(40 if (300>=x>=225)
                                                    else 0)))))
```

```
#Calulation of nitrate
Data['nna']=Data.na.apply(lambda x:(100 if (20>=x>=0)
                                    else(80 if (50>=x>=20)
                                         else(60 if (100>=x>=50)
                                              else(40 if (200>=x>=100)
                                                   else 0)))))
```

17.

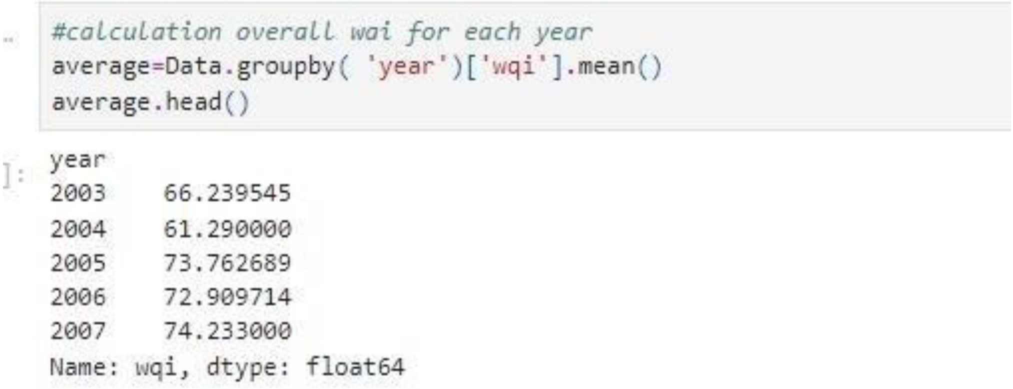


```
#Claculate water quality index WQI
Data['wph']=Data.npH* 0.165
Data['wdo']=Data.ndo * 0.281
Data['wbod']=Data.nbod* 0.234
Data['wec']=Data.nec* 0.009
Data['wna']=Data.nna* 0.028
Data['wco']=Data.nco* 0.281
Data['wqi']=Data.wph+Data.wdo+Data.wbod+Data.wec+Data.wna+Data.wco
Data
```

	station	location	state	Temp	do	ph	co	bod	na	FECAL COLIFORM (MPN/100ml)	...	nbod	nec	nna	wph	wdo	wbod	wec	wna	wco	wqi
0	1393.0	NaN	NaN	30.600000	6.7	7.5	203.0	6.940049	0.100000	11.000	...	60	60	100	16.5	28.10	14.04	0.54	2.8	22.48	84.46
1	1399.0	NaN	NaN	29.800000	5.7	7.2	189.0	2.000000	0.200000	4953.000	...	100	60	100	16.5	22.48	23.40	0.54	2.8	11.24	76.96
2	1475.0	NaN	NaN	29.500000	6.3	6.9	179.0	1.700000	0.100000	3243.000	...	100	60	100	13.2	28.10	23.40	0.54	2.8	11.24	79.28
3	3181.0	NaN	NaN	29.700000	5.8	6.9	64.0	3.800000	0.500000	5382.000	...	80	100	100	13.2	22.48	18.72	0.90	2.8	11.24	69.34
4	3182.0	NaN	NaN	29.500000	5.8	7.3	83.0	1.900000	0.400000	3428.000	...	100	80	100	16.5	22.48	23.40	0.72	2.8	11.24	77.14
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1986	1330.0	NaN	NaN	26.209814	7.9	738.0	7.2	2.700000	0.518000	0.518	...	100	100	100	0.0	28.10	23.40	0.90	2.8	16.86	72.06
1987	1450.0	NaN	NaN	29.000000	7.5	585.0	6.3	2.600000	0.155000	0.155	...	100	100	100	0.0	28.10	23.40	0.90	2.8	16.86	72.06
1988	1403.0	NaN	NaN	28.000000	7.6	98.0	6.2	1.200000	1.623079	NaN	...	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44
1989	1404.0	NaN	NaN	28.000000	7.7	91.0	6.5	1.300000	1.623079	NaN	...	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44
1990	1726.0	NaN	NaN	29.000000	7.6	110.0	5.7	1.100000	1.623079	NaN	...	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44

1991 rows × 26 columns

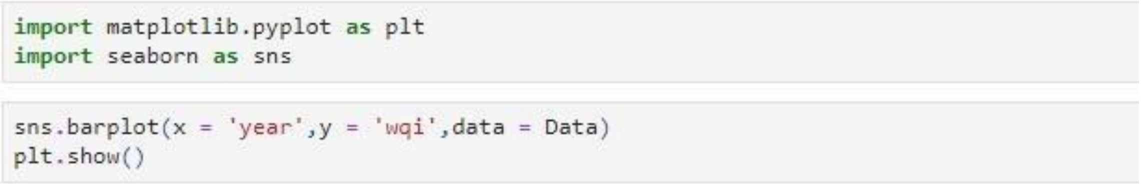
18.



```
#calculation overall wai for each year
average=Data.groupby( 'year')['wqi'].mean()
average.head()
```

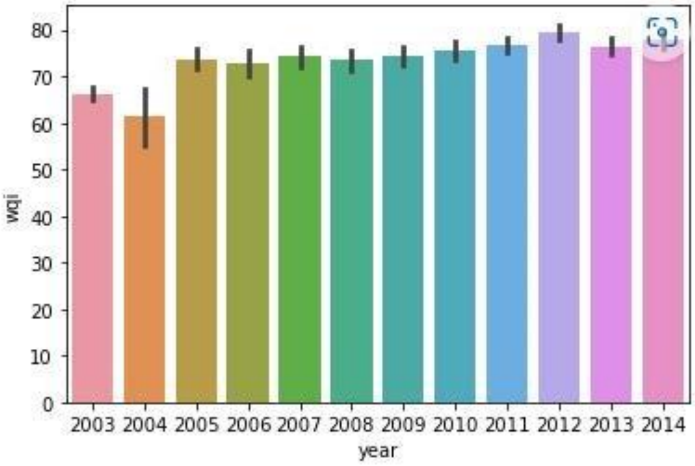
```
year
2003    66.239545
2004    61.290000
2005    73.762689
2006    72.909714
2007    74.233000
Name: wqi, dtype: float64
```

19.

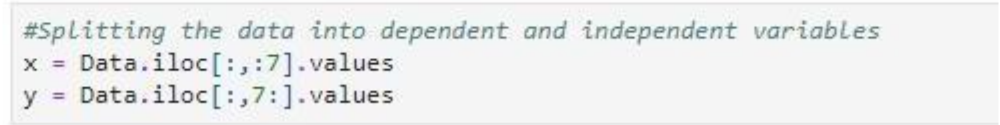


```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
sns.barplot(x = 'year',y = 'wqi',data = Data)
plt.show()
```

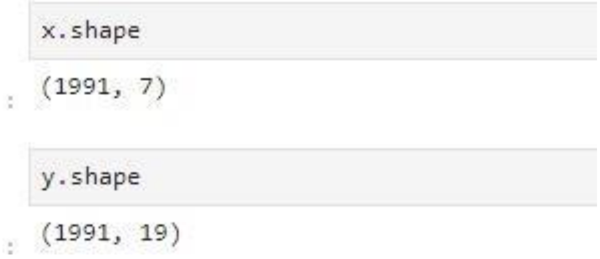


20.



```
#Splitting the data into dependent and independent variables
x = Data.iloc[:,:7].values
y = Data.iloc[:,7:].values
```

21.



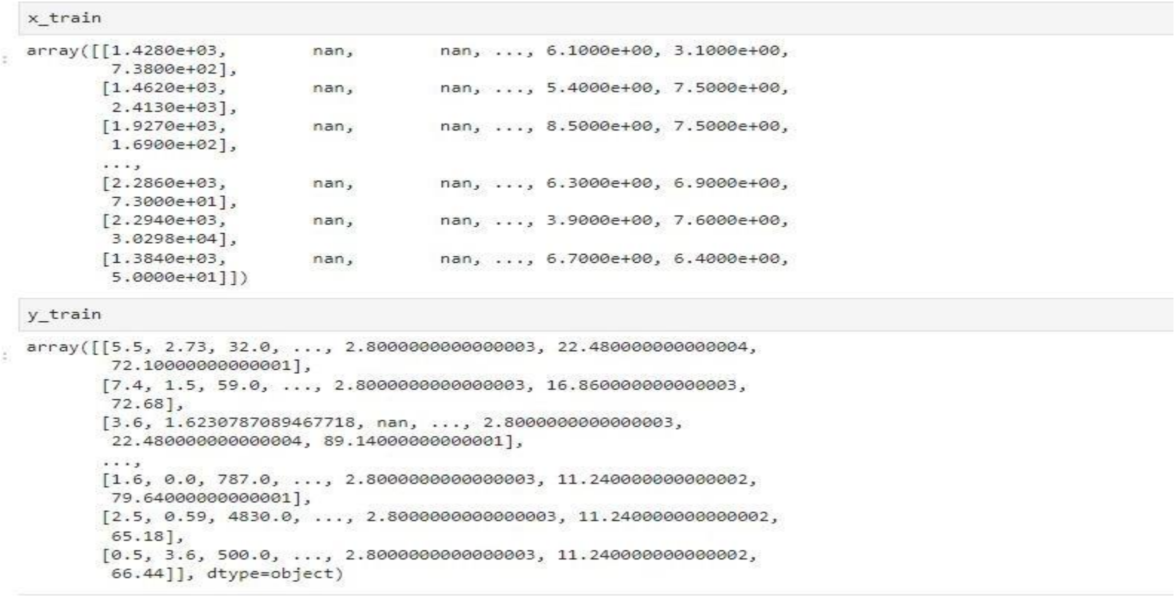
```
x.shape
```

```
(1991, 7)
```

```
y.shape
```

```
(1991, 19)
```

22.



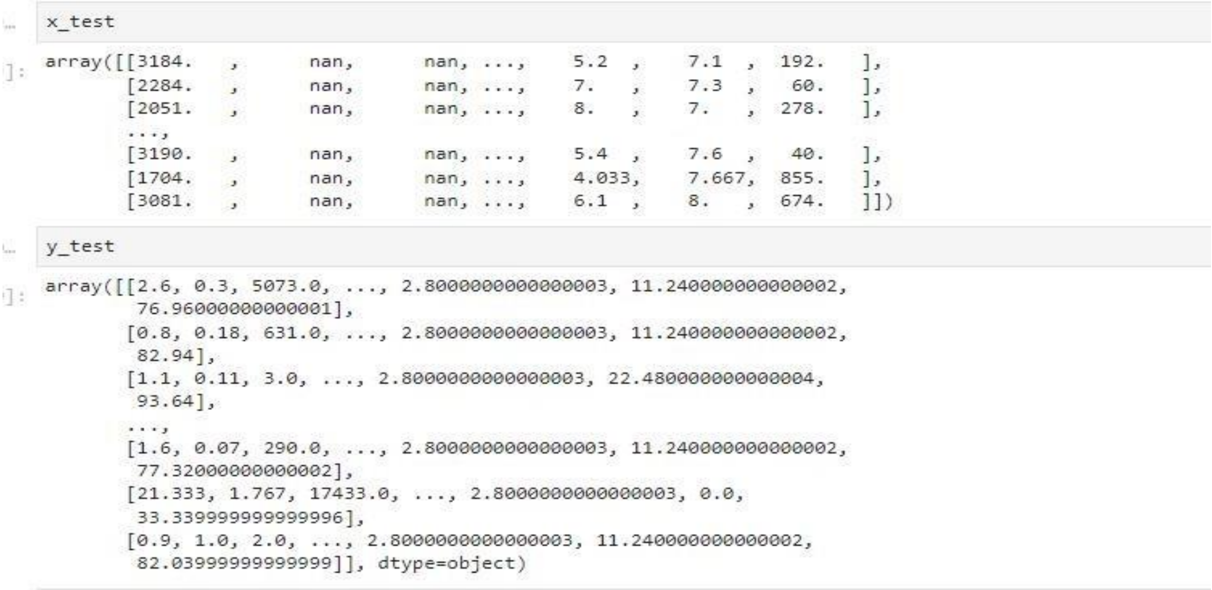
```
x_train
```

```
array([[1.4280e+03,        nan,        nan, ..., 6.1000e+00, 3.1000e+00,
        7.3800e+02],
       [1.4620e+03,        nan,        nan, ..., 5.4000e+00, 7.5000e+00,
        2.4130e+03],
       [1.9270e+03,        nan,        nan, ..., 8.5000e+00, 7.5000e+00,
        1.6900e+02],
       ...,
       [2.2860e+03,        nan,        nan, ..., 6.3000e+00, 6.9000e+00,
        7.3000e+01],
       [2.2940e+03,        nan,        nan, ..., 3.9000e+00, 7.6000e+00,
        3.0298e+04],
       [1.3840e+03,        nan,        nan, ..., 6.7000e+00, 6.4000e+00,
        5.0000e+01]])
```

```
y_train
```

```
array([[5.5, 2.73, 32.0, ..., 2.8000000000000003, 22.480000000000004,
        72.10000000000001],
       [7.4, 1.5, 59.0, ..., 2.8000000000000003, 16.860000000000003,
        72.68],
       [3.6, 1.6230787089467718, nan, ..., 2.8000000000000003,
        22.480000000000004, 89.14000000000001],
       ...,
       [1.6, 0.0, 787.0, ..., 2.8000000000000003, 11.240000000000002,
        79.64000000000001],
       [2.5, 0.59, 4830.0, ..., 2.8000000000000003, 11.240000000000002,
        65.18],
       [0.5, 3.6, 500.0, ..., 2.8000000000000003, 11.240000000000002,
        66.44]], dtype=object)
```

23.



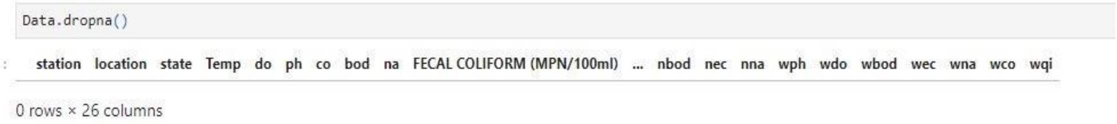
```
x_test
```

```
array([[3184.   ,      nan,      nan, ...,    5.2  ,    7.1  ,  192.   ],
       [2284.   ,      nan,      nan, ...,    7.   ,    7.3  ,   60.   ],
       [2051.   ,      nan,      nan, ...,    8.   ,    7.   ,  278.   ],
       ...,
       [3190.   ,      nan,      nan, ...,    5.4  ,    7.6  ,   40.   ],
       [1704.   ,      nan,      nan, ...,    4.033,    7.667,  855.   ],
       [3081.   ,      nan,      nan, ...,    6.1  ,    8.   ,  674.   ]])
```

```
y_test
```

```
array([[2.6, 0.3, 5073.0, ..., 2.8000000000000003, 11.240000000000002,
        76.96000000000001],
       [0.8, 0.18, 631.0, ..., 2.8000000000000003, 11.240000000000002,
        82.94],
       [1.1, 0.11, 3.0, ..., 2.8000000000000003, 22.480000000000004,
        93.64],
       ...,
       [1.6, 0.07, 290.0, ..., 2.8000000000000003, 11.240000000000002,
        77.32000000000002],
       [21.333, 1.767, 17433.0, ..., 2.8000000000000003, 0.0,
        33.339999999999996],
       [0.9, 1.0, 2.0, ..., 2.8000000000000003, 11.240000000000002,
        82.03999999999999]], dtype=object)
```

24.

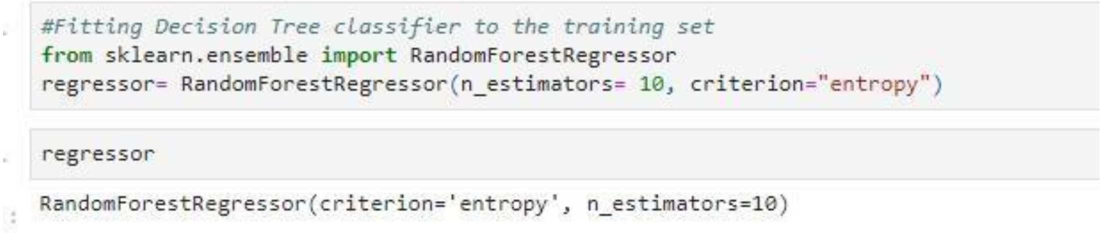


```
Data.dropna()
```

station	location	state	Temp	do	ph	co	bod	na	FECAL COLIFORM (MPN/100ml)	...	nbod	nec	nna	wph	wdo	wbod	wec	wna	wco	wqi

0 rows × 26 columns

25.



```
#Fitting Decision Tree classifier to the training set
from sklearn.ensemble import RandomForestRegressor
regressor= RandomForestRegressor(n_estimators= 10, criterion="entropy")
```

```
regressor
```

```
RandomForestRegressor(criterion='entropy', n_estimators=10)
```