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

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler


from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression

from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score,r2_score
from sklearn.metrics import confusion_matrix, classification_report

df=pd.read_csv('water_dataX.csv',encoding= 'unicode_escape')
df.head()
```

```
  STATION CODE                                        LOCATIONS  \
0         1393                DAMANGANGA AT D/S OF MADHUBAN, DAMAN
1         1399  ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...
2         1475                                ZUARI AT PANCHAWADI
3         3181                          RIVER ZUARI AT BORIM BRIDGE
4         3182                         RIVER ZUARI AT MARCAIM JETTY

         STATE  Temp D.O. (mg/l)   PH CONDUCTIVITY (µmhos/cm) B.O.D.
(mg/l)  \
0  DAMAN & DIU  30.6         6.7  7.5                     203
NAN
1          GOA  29.8         5.7  7.2                     189
2
2          GOA  29.5         6.3  6.9                     179
1.7
3          GOA  29.7         5.8  6.9                      64
3.8
4          GOA  29.5         5.8  7.3                      83
1.9

  NITRATENAN N+ NITRITENANN (mg/l) FECAL COLIFORM (MPN/100ml)  \
0                             0.1                         11
1                             0.2                       4953
2                             0.1                       3243
```

```
3                                0.5                                5382
4                                0.4                                3428

  TOTAL COLIFORM (MPN/100ml)Mean  year
0                             27  2014
1                           8391  2014
2                           5330  2014
3                           8443  2014
4                           5500  2014
```

```
l=['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']
df[df[l]=="NAN"]
```

```
     STATION CODE LOCATIONS STATE Temp D.O. (mg/l)   PH  \
0             NaN       NaN   NaN  NaN         NaN  NaN
1             NaN       NaN   NaN  NaN         NaN  NaN
2             NaN       NaN   NaN  NaN         NaN  NaN
3             NaN       NaN   NaN  NaN         NaN  NaN
4             NaN       NaN   NaN  NaN         NaN  NaN
...           ...       ...   ...  ...         ...  ...
1986          NaN       NaN   NaN  NAN         NaN  NaN
1987          NaN       NaN   NaN  NaN         NaN  NaN
1988          NaN       NaN   NaN  NaN         NaN  NaN
1989          NaN       NaN   NaN  NaN         NaN  NaN
1990          NaN       NaN   NaN  NaN         NaN  NaN

     CONDUCTIVITY (µmhos/cm) B.O.D. (mg/l) NITRATENAN N+ NITRITENANN
(mg/l)  \
0                        NaN           NAN
NaN
1                        NaN           NaN
NaN
2                        NaN           NaN
NaN
3                        NaN           NaN
NaN
4                        NaN           NaN
NaN
...                      ...           ...
...
1986                     NaN           NaN
NaN
1987                     NaN           NaN
NaN
1988                     NaN           NaN
NAN
1989                     NaN           NaN
NAN
```

```
1990                     NaN           NaN
NAN

     FECAL COLIFORM (MPN/100ml) TOTAL COLIFORM (MPN/100ml)Mean  year
0                            NaN                                NaN   NaN
1                            NaN                                NaN   NaN
2                            NaN                                NaN   NaN
3                            NaN                                NaN   NaN
4                            NaN                                NaN   NaN
...                          ...                                ...   ...
1986                         NaN                                NaN   NaN
1987                         NaN                                NaN   NaN
1988                         NAN                                NaN   NaN
1989                         NAN                                NaN   NaN
1990                         NAN                                NaN   NaN

[1991 rows x 12 columns]
```

```
for i in l:
    df.drop(df.index[df[i]=="NAN"],inplace=True,axis=0)
    df.drop(df.index[df[i]==" "],inplace=True,axis=0)
```

```
for i in l:
    df[i]=df[i].astype('float')
```

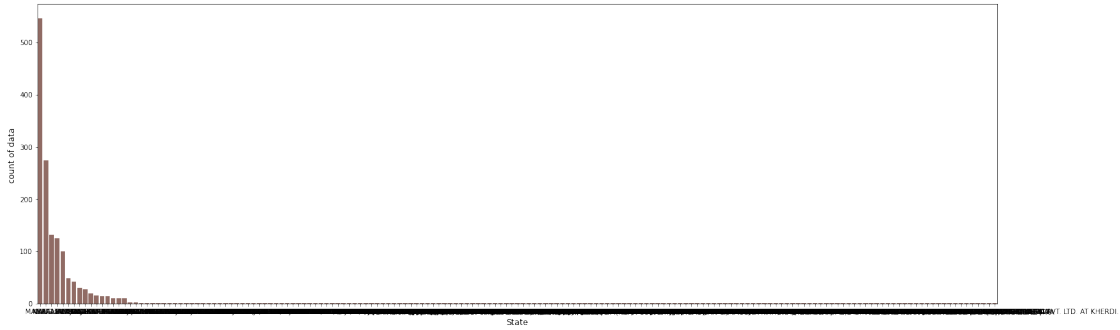
```
df.describe()
```

```
              Temp  D.O. (mg/l)            PH  CONDUCTIVITY (µmhos/cm)
\
count  1577.000000  1577.000000   1577.000000              1577.000000

mean     26.301354     6.338509     68.809670              1812.476303

std       3.264131     1.286977   1111.322252              5321.828994

min      10.000000     0.000000      2.600000                 3.700000

25%      25.000000     5.900000      6.900000                75.000000

50%      27.000000     6.700000      7.200000               170.000000

75%      28.300000     7.100000      7.600000               605.000000

max      35.000000    10.000000  28598.000000             47156.000000


       B.O.D. (mg/l)  NITRATENAN N+ NITRITENANN (mg/l)  \
count    1577.000000                       1577.000000
mean        5.204965                          1.397830
std        20.486062                          2.800052
```

```
min         0.100000                          0.000000
25%         1.100000                          0.250000
50%         1.800000                          0.510000
75%         3.500000                          1.460000
max       534.500000                         58.100000

       FECAL COLIFORM (MPN/100ml)  TOTAL COLIFORM (MPN/100ml)Mean
year
count                1.577000e+03                   1.577000e+03
1577.000000
mean                 3.841859e+05                   6.248131e+05
2010.407102
std                  9.032673e+06                   1.545279e+07
2.878751
min                  0.000000e+00                   4.000000e+00
2003.000000
25%                  2.900000e+01                   1.410000e+02
2008.000000
50%                  2.280000e+02                   5.890000e+02
2011.000000
75%                  1.000000e+03                   2.309000e+03
2013.000000
max                  2.725216e+08                   5.110909e+08
2014.000000
```

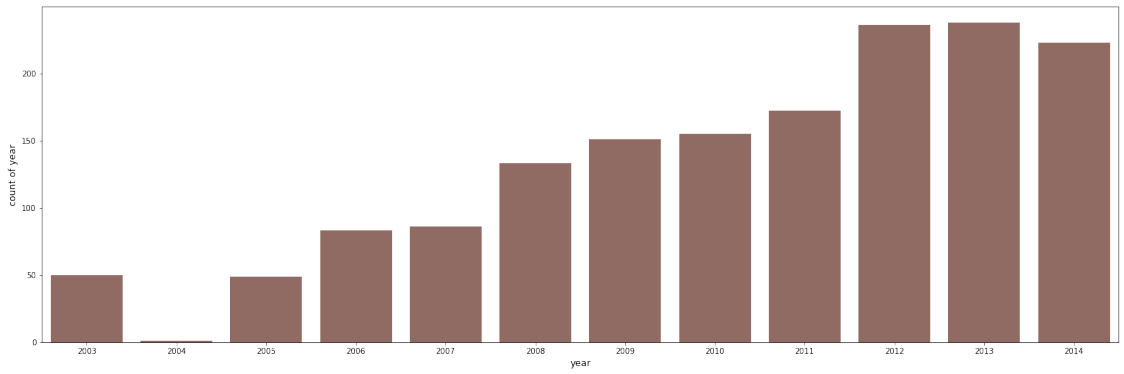
```
color=sns.color_palette()

int_level = df['STATE'].value_counts()

plt.figure(figsize=(25,8))
sns.barplot(int_level.index,int_level.values,alpha=0.9,color=color[5])
plt.ylabel('count of data ',fontsize=12)
plt.xlabel('State',fontsize=12)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  FutureWarning
```




```
color=sns.color_palette()

int_level = df['year'].value_counts()

plt.figure(figsize=(25,8))
sns.barplot(int_level.index,int_level.values,alpha=0.9,color=color[5])
plt.ylabel('count of year',fontsize=12)
plt.xlabel('year',fontsize=12)
plt.show()
```

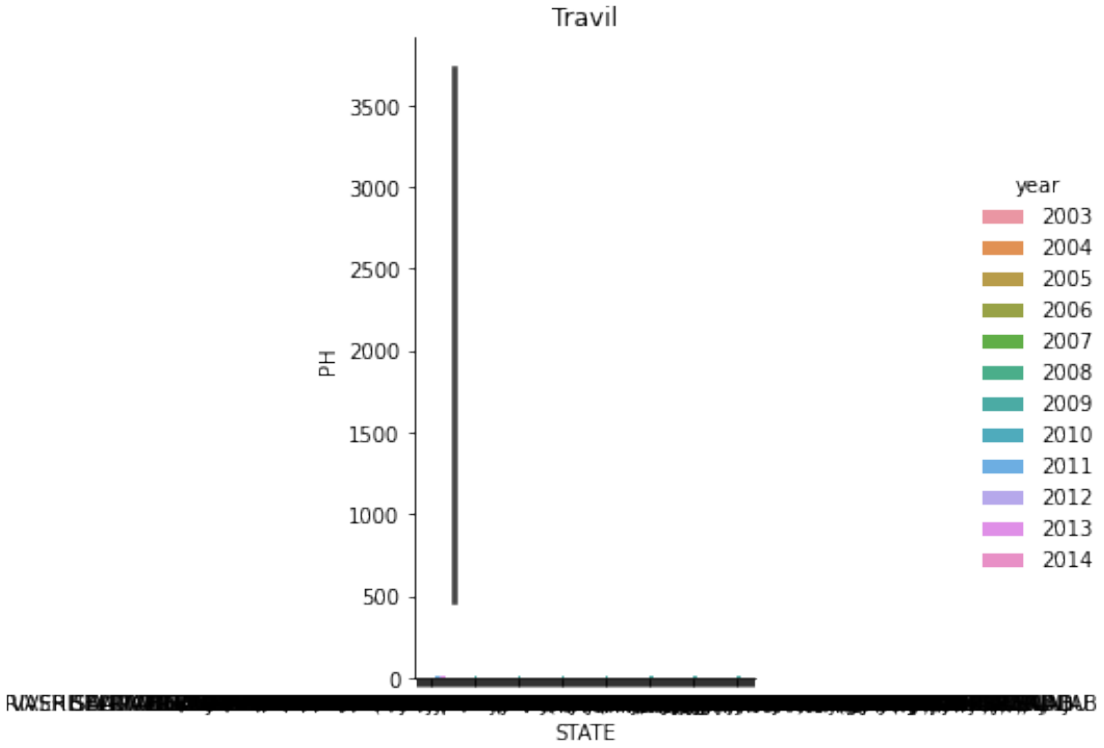
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  FutureWarning
```




```
# State and year comparision with ph rate

plt.figure(figsize=(20,20))
g=sns.catplot(data=df,kind="bar",x="STATE",y="PH",hue="year")
plt.title("Travil")
```

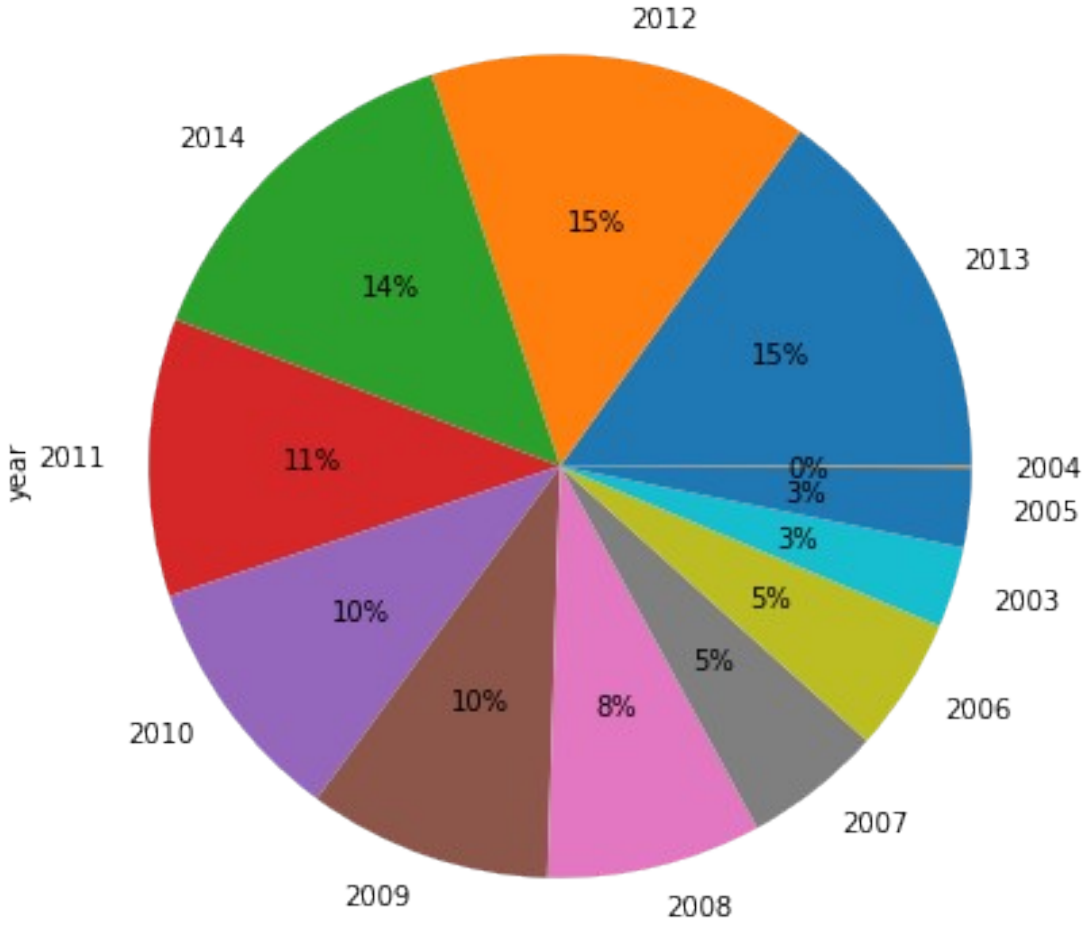
```
Text(0.5, 1.0, 'Travil')
```

```
<Figure size 1440x1440 with 0 Axes>
```

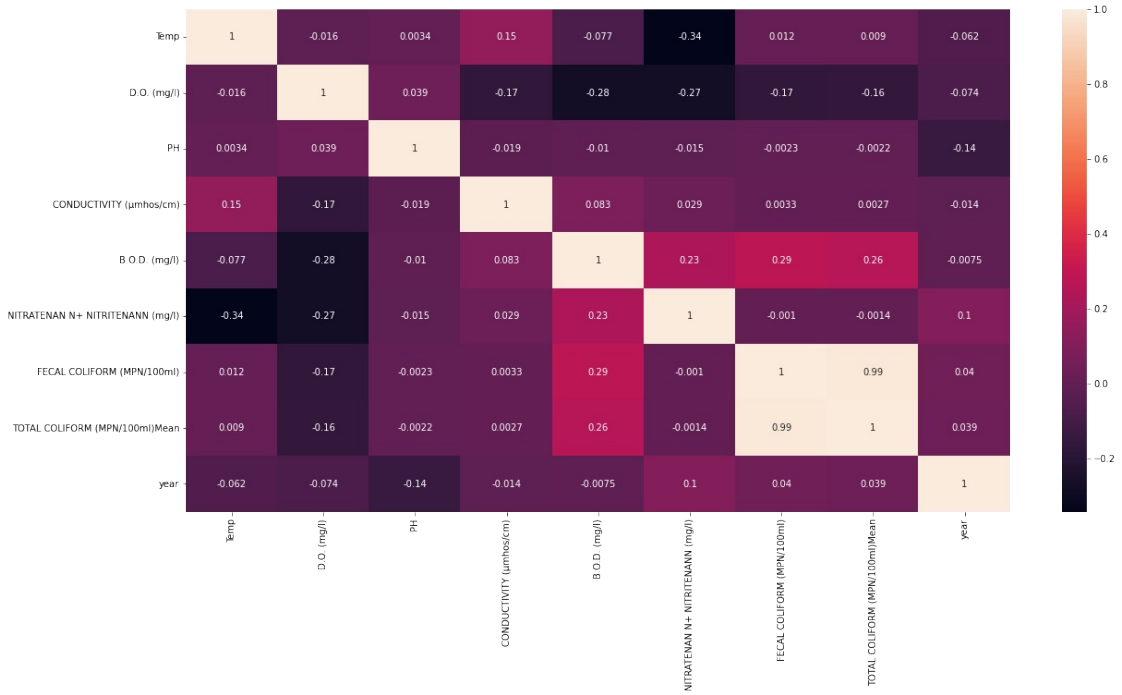



```
df['year'].value_counts().plot(kind='pie',figsize=(7,7),autopct='%1.0f
%%')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f16dc567250>
```




```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),annot=True)
plt.show()
```

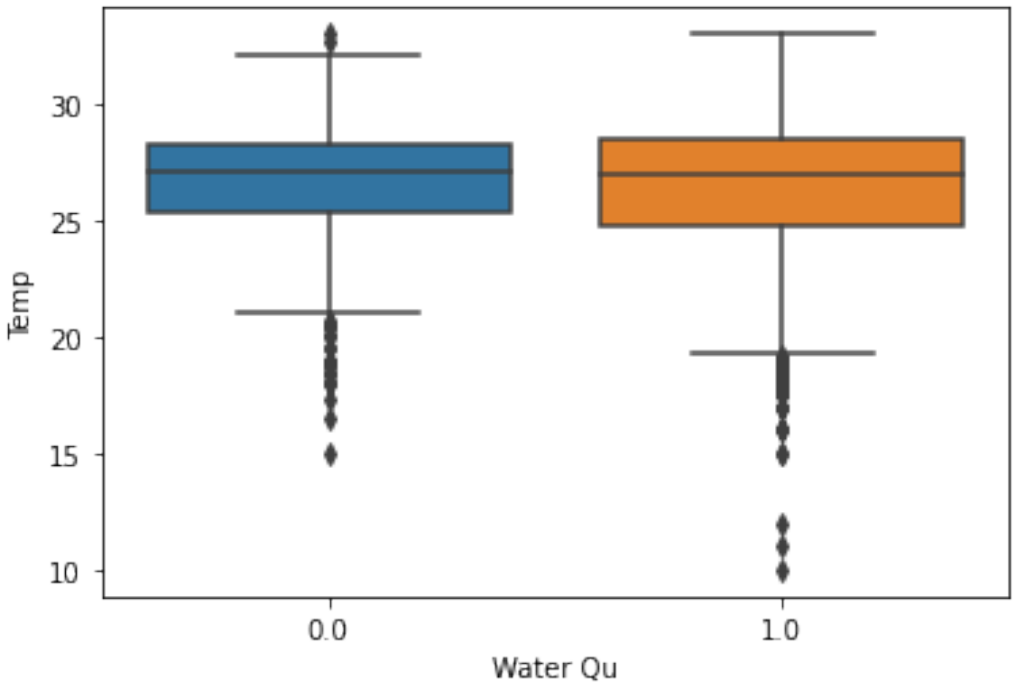



```
df['PH Range']=pd.cut(x=df['PH'],bins=[0,6.49,7.5,14],labels=['0-
6.49','6.5-7.5','7.5-14'])
df['Water Qu']=df['PH Range'].map({'6.5-7.5':1,'7.5-14':0,'0-6.49':0})

df.drop(df.index[df['PH Range']=="NaN"],inplace=True,axis=0)

df.describe()
```

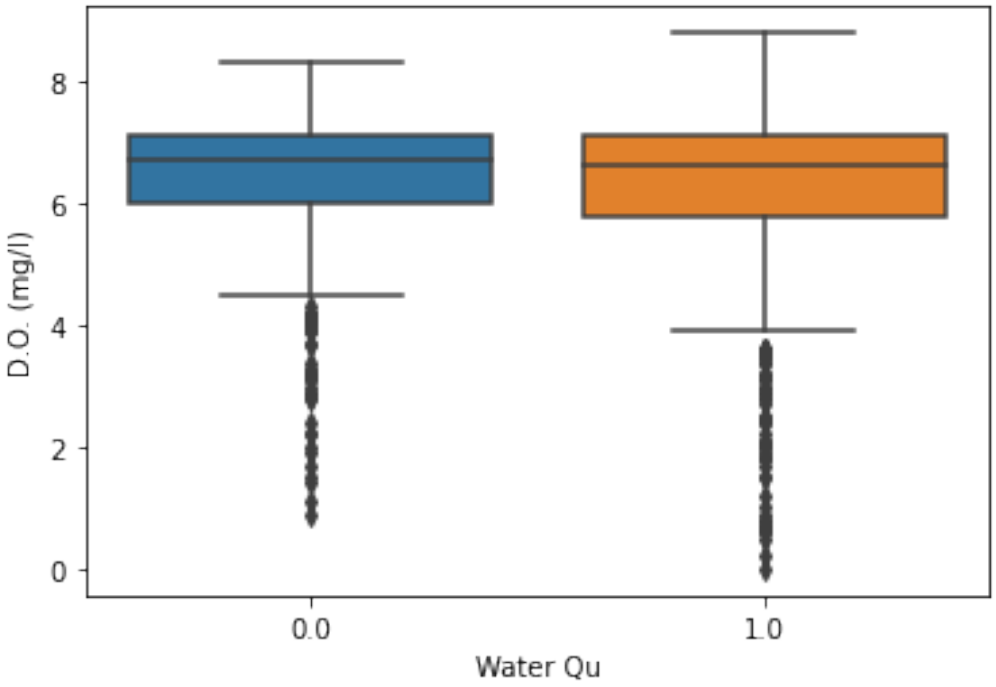
```
              Temp  D.O. (mg/l)           PH  CONDUCTIVITY (µmhos/cm)
\
count  1577.000000  1577.000000  1577.000000              1577.000000

mean     26.301354     6.338509    68.809670              1812.476303

std       3.264131     1.286977  1111.322252              5321.828994

min      10.000000     0.000000     2.600000                 3.700000

25%      25.000000     5.900000     6.900000                75.000000

50%      27.000000     6.700000     7.200000               170.000000

75%      28.300000     7.100000     7.600000               605.000000

max      35.000000    10.000000  28598.000000             47156.000000


       B.O.D. (mg/l)  NITRATENAN N+ NITRITENANN (mg/l)  \
```

```
count    1577.000000                          1577.000000
mean        5.204965                             1.397830
std        20.486062                             2.800052
min         0.100000                             0.000000
25%         1.100000                             0.250000
50%         1.800000                             0.510000
75%         3.500000                             1.460000
max       534.500000                            58.100000

       FECAL COLIFORM (MPN/100ml)  TOTAL COLIFORM (MPN/100ml)Mean  \
count                1.577000e+03                     1.577000e+03
mean                 3.841859e+05                     6.248131e+05
std                  9.032673e+06                     1.545279e+07
min                  0.000000e+00                     4.000000e+00
25%                  2.900000e+01                     1.410000e+02
50%                  2.280000e+02                     5.890000e+02
75%                  1.000000e+03                     2.309000e+03
max                  2.725216e+08                     5.110909e+08

              year     Water Qu
count  1577.000000  1526.000000
mean   2010.407102     0.659895
std       2.878751     0.473899
min    2003.000000     0.000000
25%    2008.000000     0.000000
50%    2011.000000     1.000000
75%    2013.000000     1.000000
max    2014.000000     1.000000
```

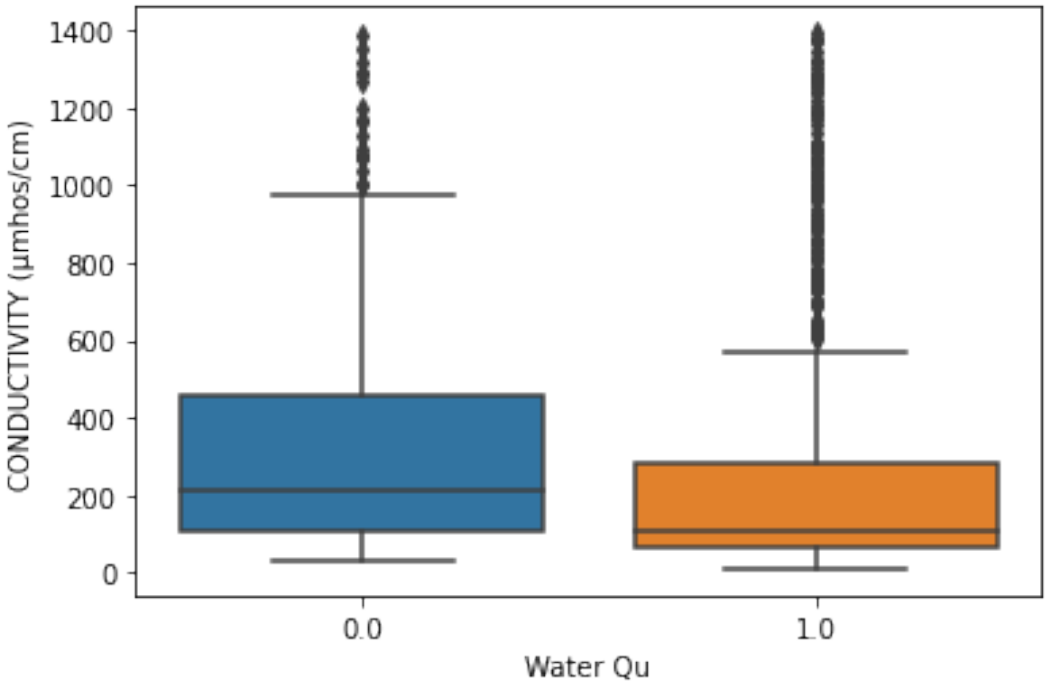
```
col_pruning=['Temp','D.O. (mg/l)','CONDUCTIVITY (µmhos/cm)','B.O.D.
(mg/l)','NITRATENAN N+ NITRITENANN (mg/l)','FECAL COLIFORM
(MPN/100ml)']

for col in col_pruning:
    print("\n\n")
    coldesc=df[col].describe()
    col_IQR=coldesc[6]-coldesc[4]
    col_Lower=coldesc[4]-(1.5*col_IQR)

    col_Higher=coldesc[6]+(1.5*col_IQR)

#     print(col_Lower,col_Higher)
#     df.drop(df.index[(df[col]col_Higher)],inplace=True,axis=0)
    df.drop(df.index[(df[col]>col_Higher)],inplace=True,axis=0)
    sns.boxplot(x='Water Qu',y=df[col],data=df)
    plt.show()


    print(df[col].describe())
```



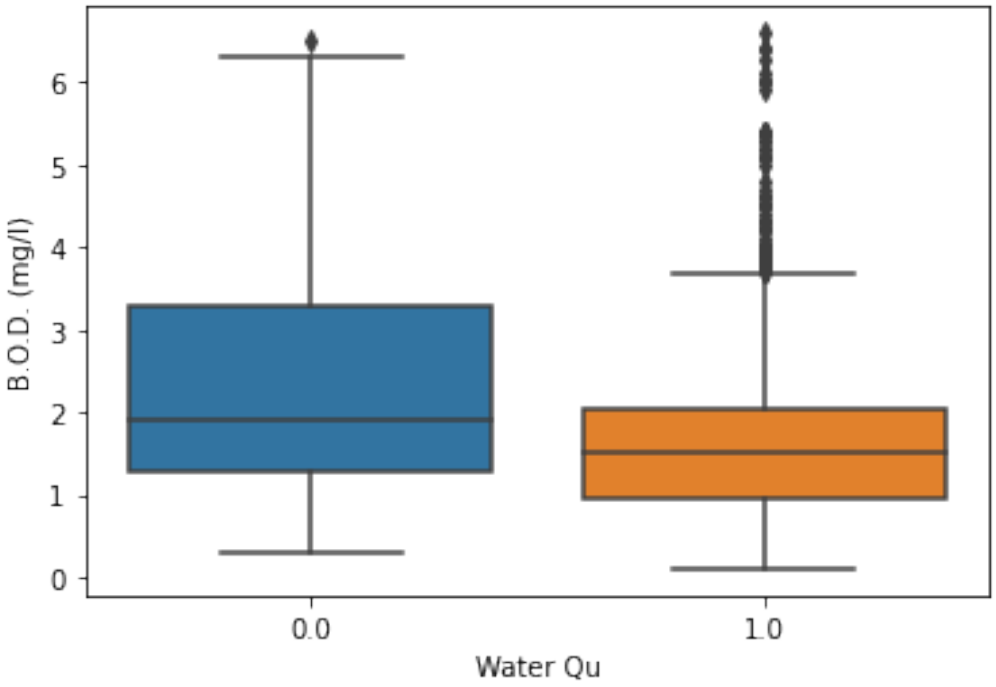

```
count    1574.000000
mean       26.285728
std         3.247477
min        10.000000
25%        25.000000
50%        27.000000
75%        28.300000
max        33.000000
Name: Temp, dtype: float64
```



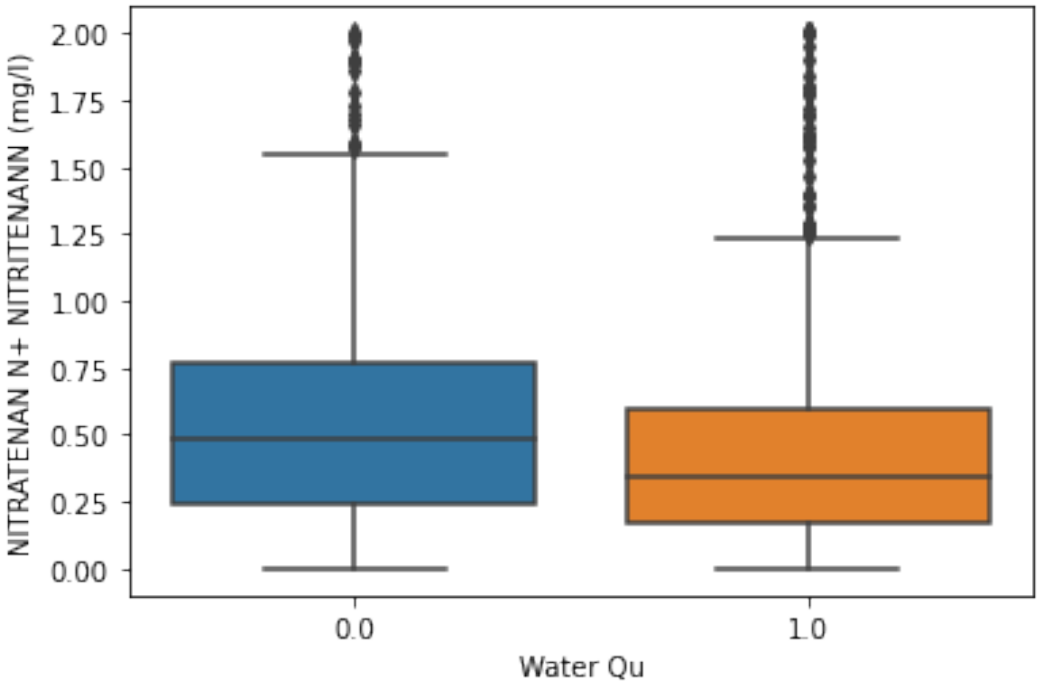

```
count    1570.000000
mean        6.332821
std         1.276125
min         0.000000
25%         5.900000
50%         6.700000
75%         7.100000
max         8.800000
Name: D.O. (mg/l), dtype: float64
```



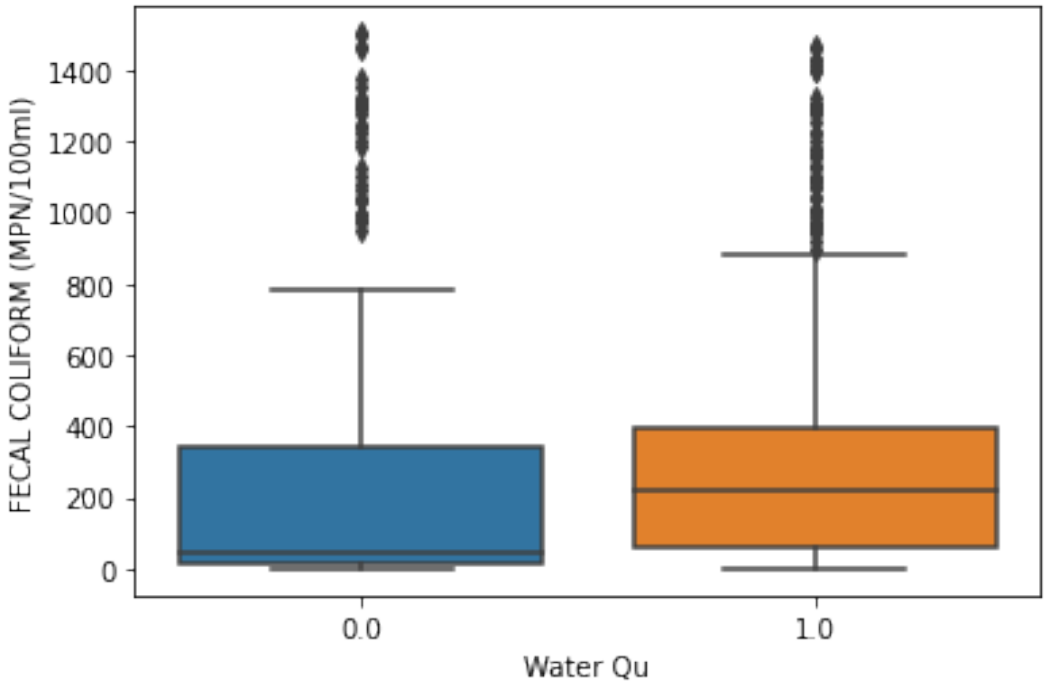

```
count    1360.000000
mean      262.216125
std       300.650740
min         3.700000
25%        68.000000
50%       129.000000
75%       316.000000
max      1392.000000
Name: CONDUCTIVITY (µmhos/cm), dtype: float64
```




```
count    1200.000000
mean        1.928923
std         1.260142
min         0.100000
25%         1.000000
50%         1.600000
75%         2.400000
max         6.600000
Name: B.O.D. (mg/l), dtype: float64
```




```
count    1051.000000
mean        0.495952
std         0.419315
min         0.000000
25%         0.200000
50%         0.400000
75%         0.652500
max         2.000000
Name: NITRATENAN N+ NITRITENANN (mg/l), dtype: float64
```




```
count     909.000000
mean      267.710273
std       341.424936
min         0.000000
25%        18.000000
50%       142.000000
75%       358.000000
max      1503.000000
Name: FECAL COLIFORM (MPN/100ml), dtype: float64
```

```
df.drop(['year'],inplace=True,axis=1)
```

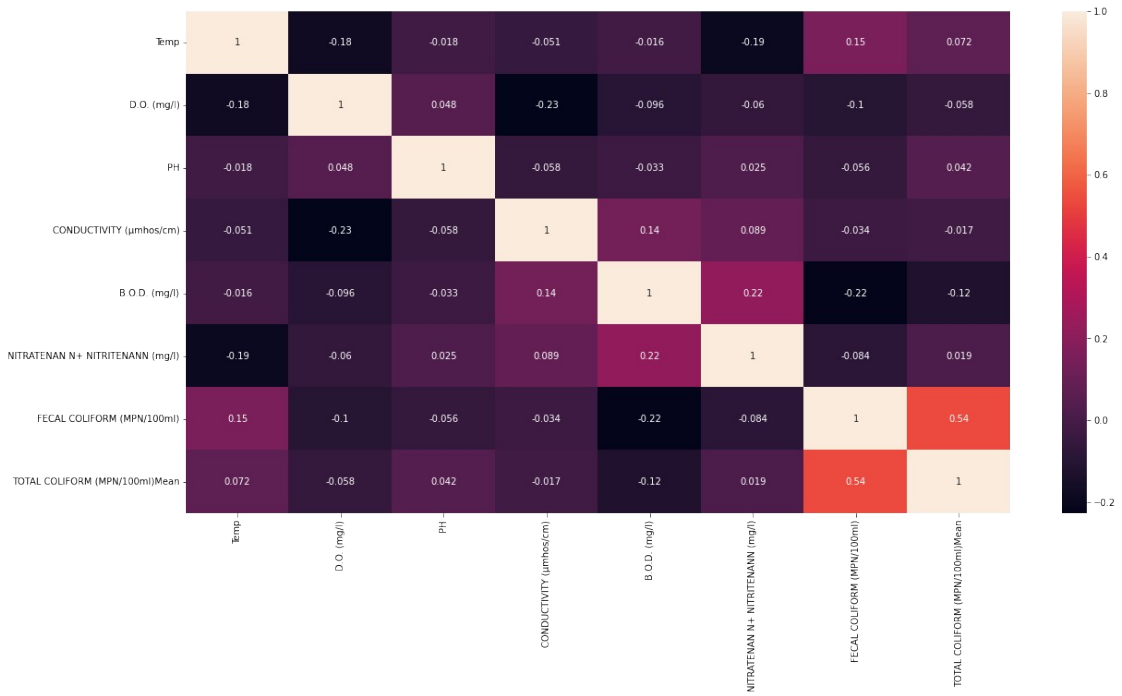
```
df.drop(['STATION CODE','LOCATIONS','STATE','PH Range','Water
Qu'],inplace=True,axis=1)
```

```
mm=MinMaxScaler()
df[l]=mm.fit_transform(df[l])
df.describe()
```

```
             Temp  D.O. (mg/l)          PH  CONDUCTIVITY (µmhos/cm)  \
count  909.000000   909.000000  909.000000                909.000000
mean     0.668897     0.731552    0.003845                  0.134526
std      0.124451     0.118140    0.051129                  0.163506
min      0.000000     0.000000    0.000000                  0.000000
25%      0.611111     0.707317    0.000147                  0.042714
50%      0.689833     0.756098    0.000161                  0.074408
75%      0.744444     0.792683    0.000175                  0.155802
max      1.000000     1.000000    1.000000                  1.000000
```

```
       B.O.D. (mg/l)  NITRATENAN N+ NITRITENANN (mg/l)  \
count     909.000000                       909.000000
mean        0.279528                         0.243960
std         0.198256                         0.204683
min         0.000000                         0.000000
25%         0.137031                         0.100000
50%         0.218750                         0.195000
75%         0.375000                         0.315000
max         1.000000                         1.000000

       FECAL COLIFORM (MPN/100ml)  TOTAL COLIFORM (MPN/100ml)Mean
count                  909.000000                     909.000000
mean                     0.178117                       0.034113
std                      0.227162                       0.060747
min                      0.000000                       0.000000
25%                      0.011976                       0.005208
50%                      0.094478                       0.017454
75%                      0.238190                       0.040116
max                      1.000000                       1.000000
```

```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),annot=True)
plt.show()
```




```
df
```

```
          Temp  D.O. (mg/l)        PH  CONDUCTIVITY (µmhos/cm)  B.O.D.
(mg/l)  \
15    0.777778     0.817073  0.000154                 0.175250
0.218750
```

```
16    0.783333     0.817073  0.000154                0.132752
0.140625
27    0.838889     0.719512  0.000171                0.296262
0.296875
29    0.561111     0.731707  0.000182                0.065764
0.750000
30    0.811111     0.768293  0.000185                0.068645
0.765625
...        ...          ...       ...                     ...
...
1981  0.888889     0.817073  0.006134                0.002161
0.218750
1982  0.777778     0.792683  0.007393                0.002233
0.343750
1984  0.722222     0.768293  0.005015                0.002449
0.296875
1985  0.722222     0.780488  0.004665                0.002737
0.203125
1987  0.777778     0.841463  0.020367                0.001873
0.390625

      NITRATENAN N+ NITRITENANN (mg/l)  FECAL COLIFORM (MPN/100ml)  \
15                                0.1000                    0.728543
16                                0.0500                    0.855622
27                                0.0500                    0.574850
29                                0.1000                    0.010645
30                                0.2000                    0.009980
...                                  ...                         ...
1981                              0.0690                    0.000092
1982                              0.2925                    0.000389
1984                              0.2530                    0.000337
1985                              0.3045                    0.000405
1987                              0.0775                    0.000103

      TOTAL COLIFORM (MPN/100ml)Mean
15                           0.099638
16                           0.123846
27                           0.062411
29                           0.001302
30                           0.001221
...                               ...
1981                         0.007567
1982                         0.014077
1984                         0.001383
1985                         0.008178
1987                         0.012653

[909 rows x 8 columns]
```

```
l=['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']
split=l.copy()
y=df['PH']
split.remove('PH')
x=df[split]

x_train, x_test, y_train, y_test= train_test_split(x, y, test_size=
0.25, random_state=42)

x_train
```

```
          Temp  D.O. (mg/l)  CONDUCTIVITY (µmhos/cm)  B.O.D. (mg/l)  \
1196  0.361111     0.902439                 0.287618       0.109375
216   0.255556     0.865854                 0.033350       0.234375
428   0.627778     0.658537                 0.119066       0.593750
1475  0.666667     0.353659                 0.072967       0.093750
1067  0.850000     0.768293                 0.059281       0.546875
...        ...          ...                      ...            ...
244   0.644444     0.731707                 0.148599       0.640625
580   0.611111     0.792683                 0.045595       0.140625
1906  0.722222     0.780488                 0.002953       0.718750
906   0.622222     0.756098                 0.031910       0.062500
231   0.144444     0.817073                 0.041273       0.328125

      NITRATENAN N+ NITRITENANN (mg/l)  FECAL COLIFORM (MPN/100ml)  \
1196                             0.050                    0.007984
216                              1.000                    0.012641
428                              1.000                    0.014637
1475                             0.075                    0.176314
1067                             0.200                    0.210246
...                                ...                         ...
244                              0.950                    0.339321
580                              0.170                    0.113107
1906                             0.050                    0.000067
906                              0.220                    0.815037
231                              1.000                    0.016633

      TOTAL COLIFORM (MPN/100ml)Mean
1196                         0.000854
216                          0.001302
428                          0.003255
1475                         0.032385
1067                         0.023516
...                               ...
244                          0.023109
580                          0.021726
1906                         0.005737
906                          0.082225
231                          0.001953
```

```
[681 rows x 7 columns]
```

```
regressor= LinearRegression()
regressor.fit(x_train, y_train)
y_pred= regressor.predict(x_test)
# x_pred= regressor.predict(x_train)
```

```
ypred_pd=pd.DataFrame({'WQ':y_test.values,'WQ_Pred':y_pred})
ypred_pd['predicted']=ypred_pd['WQ_Pred'].map(lambda x:1 if x>0.5 else
0)
ypred_pd['WQ']=ypred_pd['WQ'].map(lambda x:1 if x>0.7 else 0)
ypred_pd.head()
```

```
   WQ   WQ_Pred  predicted
0   0  0.012644          0
1   0  0.007889          0
2   0  0.003795          0
3   0  0.011636          0
4   0  0.001840          0
```

```
confusion=confusion_matrix(ypred_pd['WQ'],ypred_pd['predicted'])
print(confusion)
```

```
[[227   0]
 [  1   0]]
```

```
print(accuracy_score(ypred_pd['WQ'],ypred_pd['predicted']))
```

```
0.9956140350877193
```

```
clf_gini = DecisionTreeRegressor(random_state = 0)

clf_gini.fit(x_train, y_train)

y_pred = clf_gini.predict(x_test)
```

```
ypred_pd=pd.DataFrame({'WQ':y_test.values,'WQ_Pred':y_pred})
ypred_pd['predicted']=ypred_pd['WQ_Pred'].map(lambda x:1 if x>0.7 else
0)
ypred_pd['WQ']=ypred_pd['WQ'].map(lambda x:1 if x>0.7 else 0)
ypred_pd.head()
```

```
   WQ   WQ_Pred  predicted
0   0  0.001273          0
1   0  0.000161          0
2   0  0.000154          0
3   0  0.000192          0
4   0  0.000161          0
```

```
print('Model accuracy score with criterion gini index: {0:0.4f}'.
format(accuracy_score(ypred_pd['WQ'],ypred_pd['predicted'])))
```

```
Model accuracy score with criterion gini index: 0.9956
```

```
forest_model = RandomForestRegressor(random_state=1)
forest_model.fit(x_train, y_train)
melb_preds = forest_model.predict(x_test)
```

```
ypred_pd=pd.DataFrame({'WQ':y_test.values,'WQ_Pred':y_pred})
ypred_pd['predicted']=ypred_pd['WQ_Pred'].map(lambda x:1 if x>0.7 else
0)
ypred_pd['WQ']=ypred_pd['WQ'].map(lambda x:1 if x>0.7 else 0)
ypred_pd.head()
```

```
   WQ   WQ_Pred  predicted
0   0  0.001273          0
1   0  0.000161          0
2   0  0.000154          0
3   0  0.000192          0
4   0  0.000161          0
```

```
print(accuracy_score(ypred_pd['WQ'],ypred_pd['predicted']))
```

```
0.9956140350877193
```

```
import pickle
```

```
with open('model_pkl', 'wb') as files:
    pickle.dump(regressor, files)
with open('model_pkl' , 'rb') as f:
    lr = pickle.load(f)
lr.predict([list(x_train.iloc[1])])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  "X does not have valid feature names, but"
```

```
array([0.01363712])
```

```
with open('model_pkl', 'wb') as files:
    pickle.dump(clf_gini, files)
with open('model_pkl' , 'rb') as f:
    lr = pickle.load(f)
lr.predict([list(x_train.iloc[1])])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451:
UserWarning: X does not have valid feature names, but
DecisionTreeRegressor was fitted with feature names
  "X does not have valid feature names, but"
```

```
array([0.00016087])
```

```
with open('model_pkl', 'wb') as files:
    pickle.dump(forest_model, files)
with open('model_pkl' , 'rb') as f:
    lr = pickle.load(f)
lr.predict([list(x_train.iloc[1])])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451:
UserWarning: X does not have valid feature names, but
RandomForestRegressor was fitted with feature names
  "X does not have valid feature names, but"
```

```
array([0.00015866])
```