Importing Libraries

In [4]:

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import os

from matplotlib import rcParams

import warnings
```

In [5]

```
warnings.filterwarnings(action='ignore')

warnings.warn('this is a warning!')
```

Reading the Dataset

In [6]:

```
data = pd.read_csv(r'C:\Users\Cloud\Desktop\water quality
analysis\Data\water_dataX.csv',encoding='ISO-8859-1',low_memory=False)
```

Analysing the Data

In [7]:

```
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**
0	1393	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.6	6.7 6.7 7.5	7.5 \\	203	NAN	0.1	11	27

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
2	1475	ZUARI AT PANCHAWADI	GOA	29.5	6.3	6.9	179	1.7	0.1	3243	5330
3	3181	RIVER ZUARI AT BORIM BRIDGE	GOA	29.7	5.8	6.9	64	3.8	0.5	5382	8443
4	3182	RIVER ZUARI AT MARCAIM JETTY	GOA	29.5	5.8	7.3	83	1.9	0.4	3428	5500

```
data.describe()
```

Out[8]:

	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

```
data.info()
```

```
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
```

In [10]:

```
data.shape
```

Out[10]:

```
(1991, 12)
```

Checking for missing values

In [11]:

```
data.isnull().any()
```

Out[11]:

```
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
```

In [12]:

```
data.isnull().sum()
```

Out[12]:

```
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
```

In [13]:

```
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
```

```
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.dtypes

```
STATION CODE                          object
LOCATIONS                             object
STATE                                 object
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)            object
TOTAL COLIFORM (MPN/100ml)Mean       float64
year                                   int64
dtype: object
```

In [15]:

```
data.isnull().sum()
```

Out[15]:

```
STATION CODE                          0
LOCATIONS                             0
STATE                                 0
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)            0
TOTAL COLIFORM (MPN/100ml)Mean      132
year                                  0
dtype: int64
```

Fill the Null Values

In [16]:

```
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)
```

In [17]:

```
data.drop(["FECAL COLIFORM (MPN/100ml)"],axis=1,inplace=True)
```

Renaming the Column Names

In [18]:

```
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'})
```

In [19]:

```
data
```

stationlocationstateTempdophcobodnatcyear01393DAMANGANGA AT D/S OF MADHUBAN, DAMANDAMAN & DIU30.6000006.77.5203.06.9400490.10000027.0201411399ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...GOA29.8000005.77.2189.02.0000000.2000008391.0201421475ZUARI AT PANCHAWADIGOA29.5000006.36.9179.01.7000000.1000005330.0201433181RIVER ZUARI AT BORIM BRIDGEGOA29.7000005.86.964.03.8000000.5000008443.0201443182RIVER ZUARI AT MARCAIM JETTYGOA29.5000005.87.383.01.9000000.4000005500.02014.....................................19861330TAMBIRAPARANI AT ARUMUGANERI, TAMILNADUNAN26.2098147.9738.07.22.7000000.518000202.0200319871450PALAR AT VANIYAMBADI WATER SUPPLY HEAD WORK, T...NAN29.0000007.5585.06.32.6000000.155000315.0200319881403GUMTI AT U/S SOUTH TRIPURA,TRIPURANAN28.0000007.698.06.21.2000001.623079570.0200319891404GUMTI AT D/S SOUTH TRIPURA, TRIPURANAN28.0000007.791.06.51.3000001.623079562.0200319901726CHANDRAPUR, AGARTALA D/S OF HAORA RIVER, TRIPURANAN29.0000007.6110.05.71.1000001.623079546.02003

1991 rows × 11 columns

Water Quality Index (WQI) Calculation

a)Claculation of pH

In [20]:

```
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)))))
```

b)calculation of dissolved oxygen

In [21]:

```
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)))))
```

c)calculation of total coliform

In [22]:

```
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)))))
```

d)calculation of B.D.O

In [23]:

```
data['nbdo']=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)))))
```

e)calculation of electric conductivity

In [24]:

```
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)))))
```

f)calculation of nitrate

In [25]:

```python
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)))))
```

Calculation of Water Quality Index WQI

In [26]:

```python
data['wph']=data.npH*0.165
data['wdo']=data.ndo*0.281
data['wbdo']=data.nbdo*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.wbdo+data.wec+data.wna+data.wco
data
```

	station	location	state	Temp	do	ph	co	bod	na	tc	...	nbdo	nec	nna	wph	wdo	wbdo	wec	wna	wco	wqi
0	1393	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.600000	6.7	7.5	203.0	6.940049	0.100000	27.0	...	60	60	100	16.5	28.10	14.04	0.54	2.8	22.48	84.46
1	1399	ZUARI AT D/S OF	GOA	29.80	5.7	7.2	18	2.000	0.200	839	...	100	60	100	16	22.	23.	0.	2.8	11.	76.

		PT. WHERE KUMBARJRIA CANAL JOI...		0000			9.0	000	000	1.0					.5	48	40	54		24	96
2	1475	ZUARI AT PANCHAWADI	GOA	29.500000	6.3	6.9	179.0	1.700000	0.100000	5330.0	...	100	60	100	13.2	28.10	23.40	0.54	2.8	11.24	79.28
3	3181	RIVER ZUARI AT BORIM BRIDGE	GOA	29.700000	5.8	6.9	64.0	3.800000	0.500000	8443.0	...	80	100	100	13.2	22.48	18.72	0.90	2.8	11.24	69.34
4	3182	RIVER ZUARI AT MARCAIM JETTY	GOA	29.500000	5.8	7.3	83.0	1.900000	0.400000	5500.0	...	100	80	100	16.5	22.48	23.40	0.72	2.8	11.24	77.14
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1986	1330	TAMBIRAPARANI AT ARUMUGANERI, TAMILNADU	NAN	26.209814	7.9	738.0	7.2	2.700000	0.518000	202.0	...	100	100	100	0.0	28.10	23.40	0.90	2.8	16.86	72.06

1987	1450	PALAR AT VANIYAMBADI WATER SUPPLY HEAD WORK, T...	NAN	29.000000	7.5	585.0	6.3	2.600000	0.155000	315.0	...	100	100	100	0.0	28.10	23.40	0.90	2.8	16.86	72.06
1988	1403	GUMTI AT U/S SOUTH TRIPURA, TRIPURA	NAN	28.000000	7.6	98.0	6.2	1.200000	1.623079	570.0	...	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44
1989	1404	GUMTI AT D/S SOUTH TRIPURA, TRIPURA	NAN	28.000000	7.7	91.0	6.5	1.300000	1.623079	562.0	...	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44
1990	1726	CHANDRAPUR, AGARTALA D/S OF HAORA RIVER, TRIPURA	NAN	29.000000	7.6	110.0	5.7	1.100000	1.623079	546.0	...	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44

1991 rows × 11 columns

Water Quality Index (WQI) Calculation

a)Claculation of pH

In [20]:

```
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)))))
```

b)calculation of dissolved oxygen

In [21]:

```
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)))))
```

c)calculation of total coliform

In [22]:

```
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)))))
```

d)calculation of B.D.O

In [23]:

```
data['nbdo']=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)))))
```

e)calculation of electric conductivity

In [24]:

```
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)))))
```

f)calculation of nitrate

In [25]:

```
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)))))
```

Calculation of Water Quality Index WQI

In [26]:

```
data['wph']=data.npH*0.165

data['wdo']=data.ndo*0.281

data['wbdo']=data.nbdo*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.wbdo+data.wec+data.wna+data.wco
```

```
data
```

stationlocationstateTempdophcobodnatc...nbdonecnnawphwdowbdowecwnawcowqi01393DAMANGANGA AT D/S OF MADHUBAN, DAMANDAMAN & DIU30.6000006.77.5203.06.9400490.10000027.0...606010016.528.1014.040.542.822.4884.4611399ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...GOA29.8000005.77.2189.02.0000000.2000008391.0...1006010016.522.4823.400.542.811.2476.9621475ZUARI AT PANCHAWADIGOA29.5000006.36.9179.01.7000000.1000005330.0...1006010013.228.1023.400.542.811.2479.2833181RIVER ZUARI AT BORIM BRIDGEGOA29.7000005.86.964.03.8000000.5000008443.0...8010010013.222.4818.720.902.811.2469.3443182RIVER ZUARI AT MARCAIM JETTYGOA29.5000005.87.383.01.9000000.4000005500.0...1008010016.522.4823.400.722.811.2477.14..19861330TAMBIRAPARANI AT ARUMUGANERI, TAMILNADUNAN26.2098147.9738.07.22.7000000.518000202.0...1001001000.028.1023.400.902.816.8672.0619871450PALAR AT VANIYAMBADI WATER SUPPLY HEAD WORK, T...NAN29.0000007.5585.06.32.6000000.155000315.0...1001001000.028.1023.400.902.816.8672.0619881403GUMTI AT U/S SOUTH TRIPURA,TRIPURANAN28.0000007.698.06.21.2000001.623079570.0...1001001000.028.1023.400.902.811.2466.4419891404GUMTI AT D/S SOUTH TRIPURA, TRIPURANAN28.0000007.791.06.51.3000001.623079562.0...1001001000.028.1023.400.902.811.2466.4419901726CHANDRAPUR, AGARTALA D/S OF HAORA RIVER, TRIPURANAN29.0000007.6110.05.71.1000001.623079546.0...1001001000.028.1023.400.902.811.2466.44

1991 rows × 24 columns

Calculation of overall WQI for each year

In [27]:

```
average = data.groupby('year')['wqi'].mean()

average.head()
```

Out[27]:

```
year

2003    66.239545

2004    61.290000

2005    73.762689

2006    72.909714

2007    74.233000

Name: wqi, dtype: float64
```

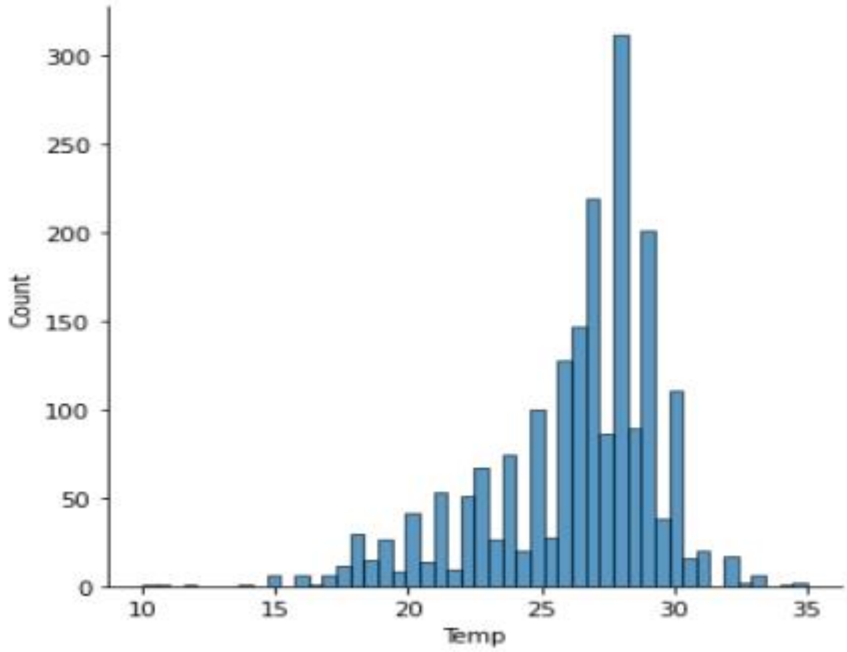
Data Visualization

Univariate analysis

a)displot

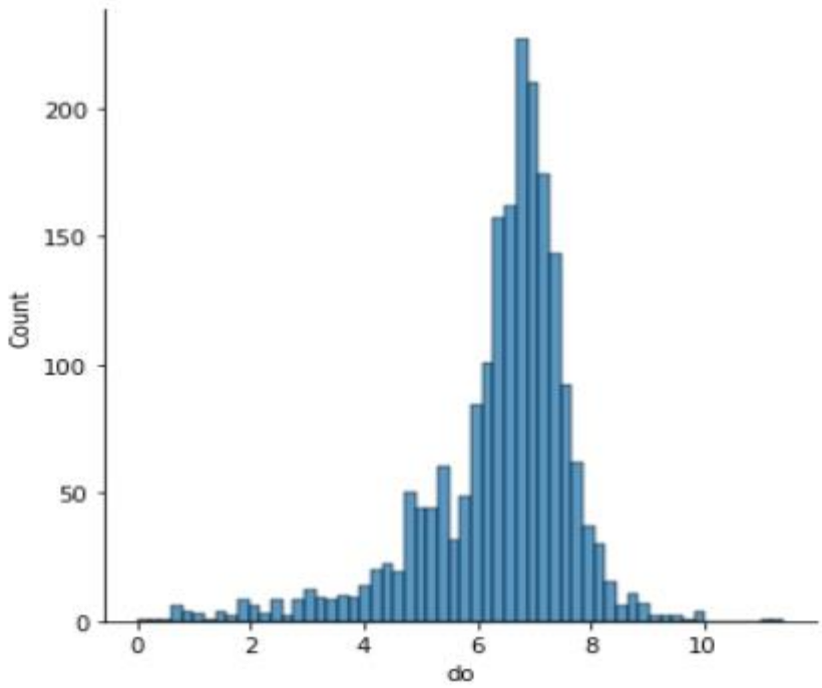
In [28]:

```
sns.displot(data.Temp)

plt.show()

sns.displot(data.Temp)

plt.show()
```



sns.displot(data.do)

plt.show()

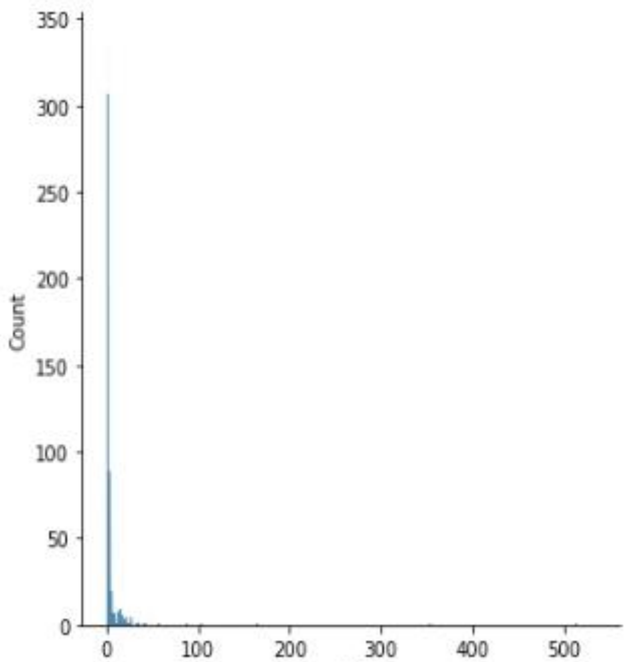


sns.displot(data.bod)

plt.show()

sns.displot(data.na)

plt.show()

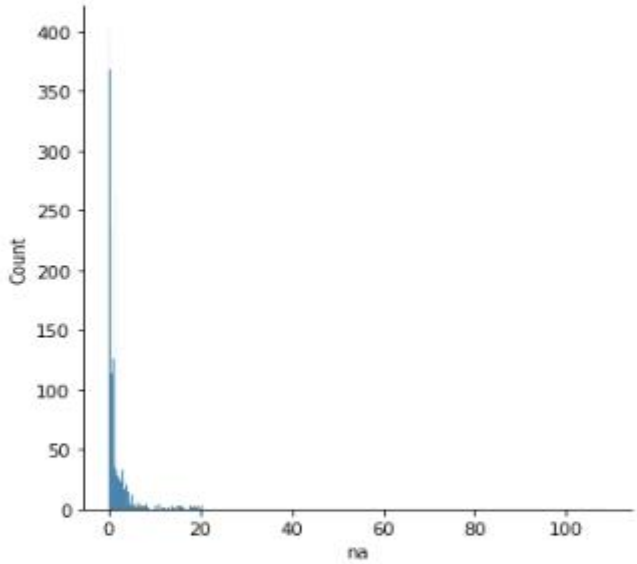


sns.displot(data.na)

plt.show()

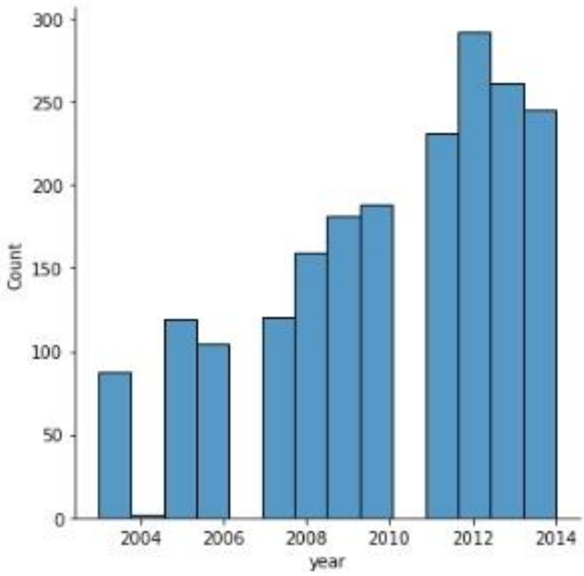
sns.displot(data.year)

plt.show()



sns.displot(data.year)

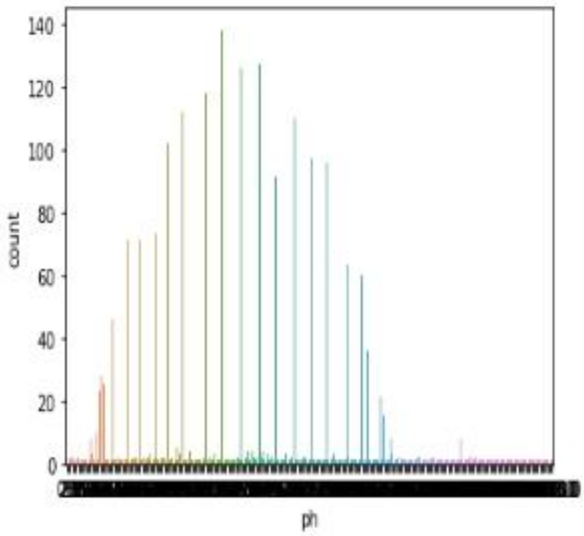
plt.show()



b)countplot

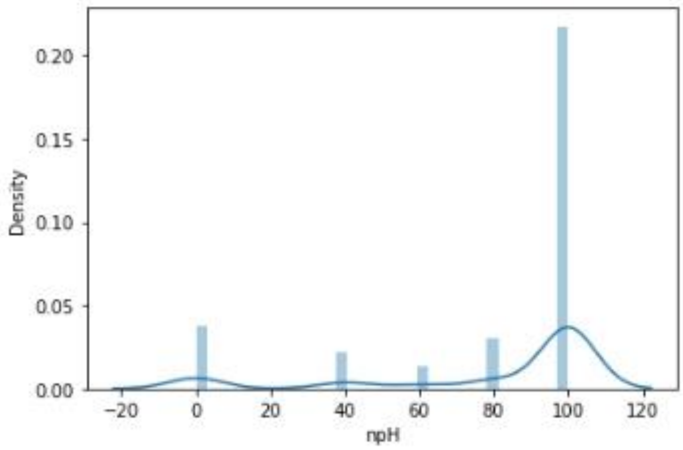
In [33]:

```
sns.countplot(data.ph)

plt.show()
```



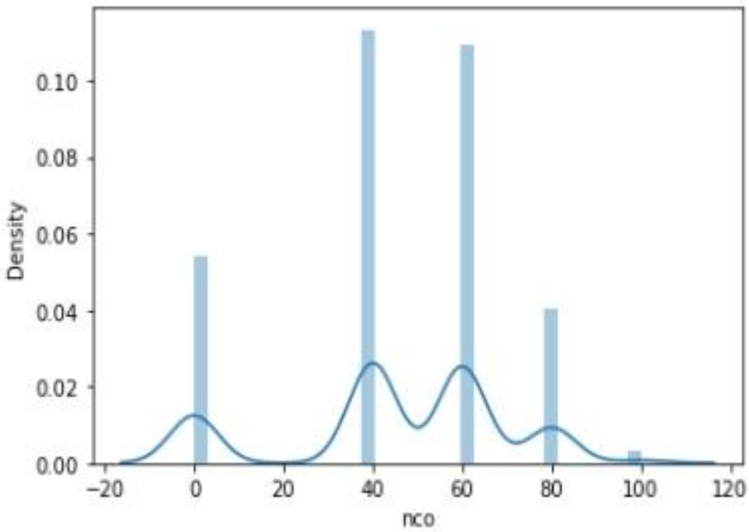
sns.distplot(data.npH)

plt.show()



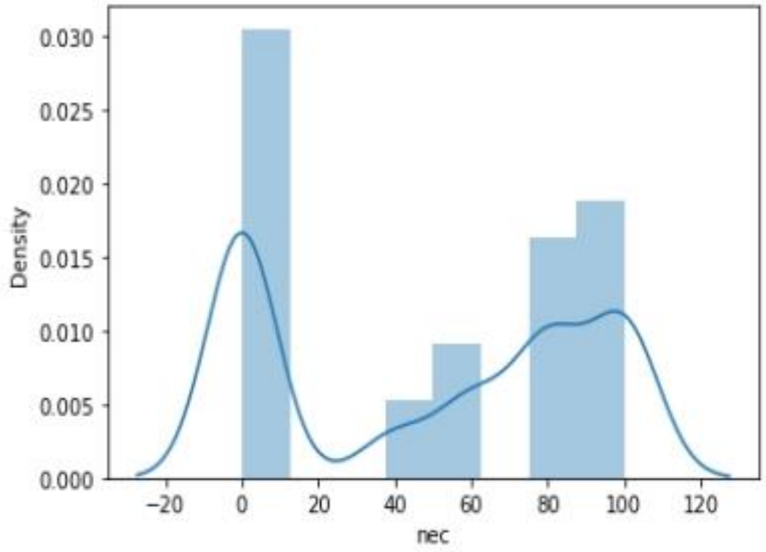
sns.distplot(data.nco)

plt.show()



sns.distplot(data.nec)

plt.show()



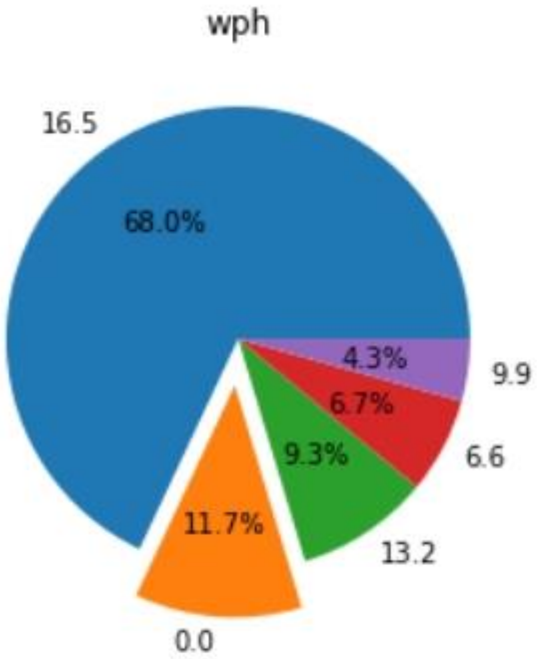
c)pie chart

In [37]:

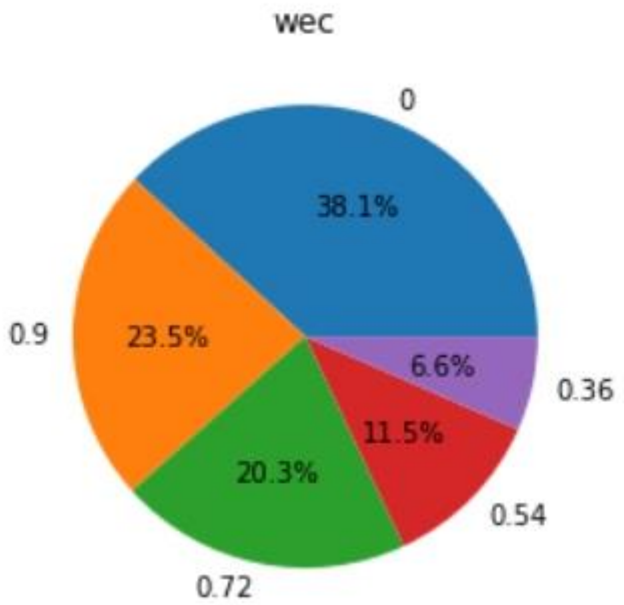
```
plt.pie(data.year.value_counts(),[0.1,0,0,0,0,0,0,0,0,0,0,0],labels=[2012,2
013,2014,2011,2010,2009,2008,2007,2005,2006,2003,2004 ],autopct='%1.1f%%')

plt.title('YEAR')

plt.show()
```


```
plt.pie(data.wph.value_counts(),[0,0.2,0,0,0],labels=[16.5,0.0,13.2,6.6,9.9],autopct='%1.1f%%')
```

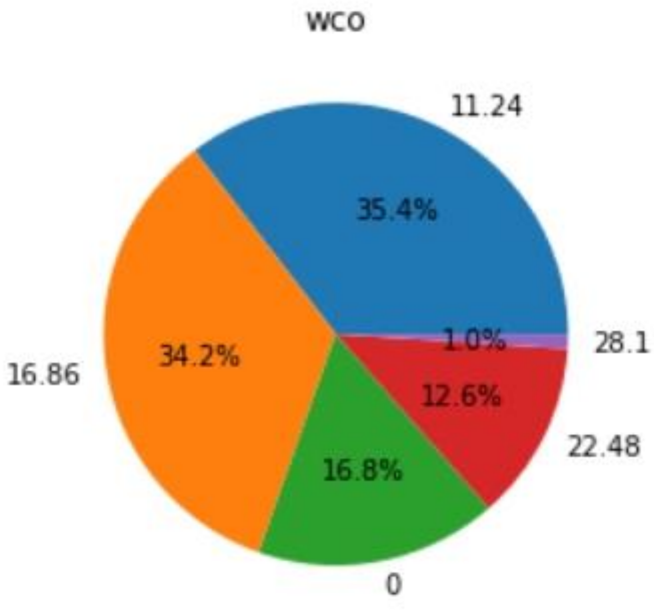
```
plt.title('wph')

plt.show()
```




```
plt.pie(data.wec.value_counts(),labels=[0,0.90,0.72,0.54,0.36],autopct='%1.1f%%')

plt.title('wec')

plt.show()
```



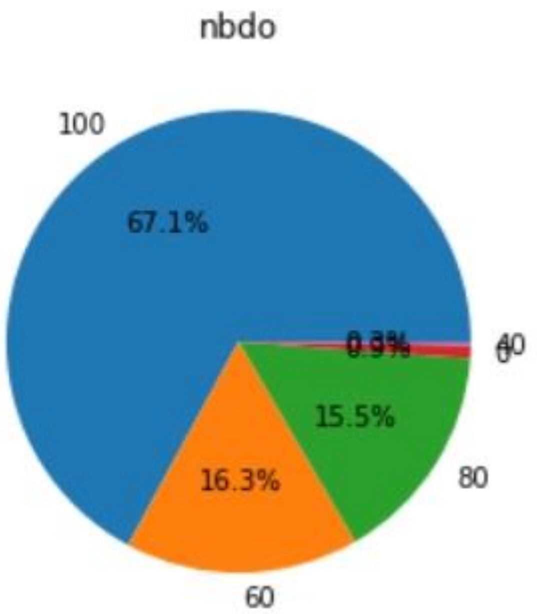

```
plt.pie(data.nbdo.value_counts(),labels=[100,60,80,0,40],autopct='%1.1f%%')
plt.title('nbdo')
plt.show()
```



plt.pie(data.wco.value_counts(),labels=[11.24,16.86,0,22.48,28.10],autopct='%1.1f%%')

plt.title('wco')

plt.show()

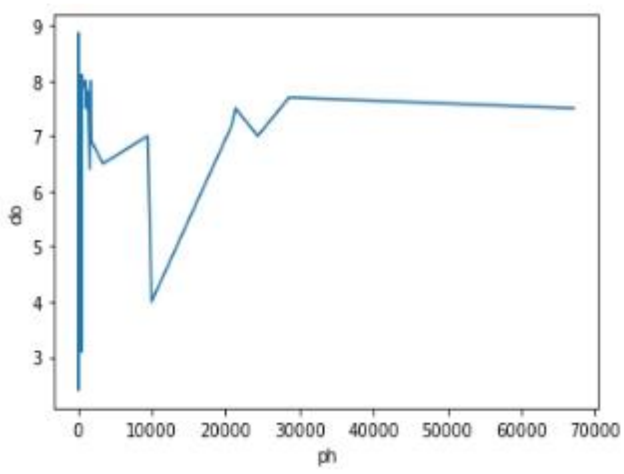


Bivariate analysis

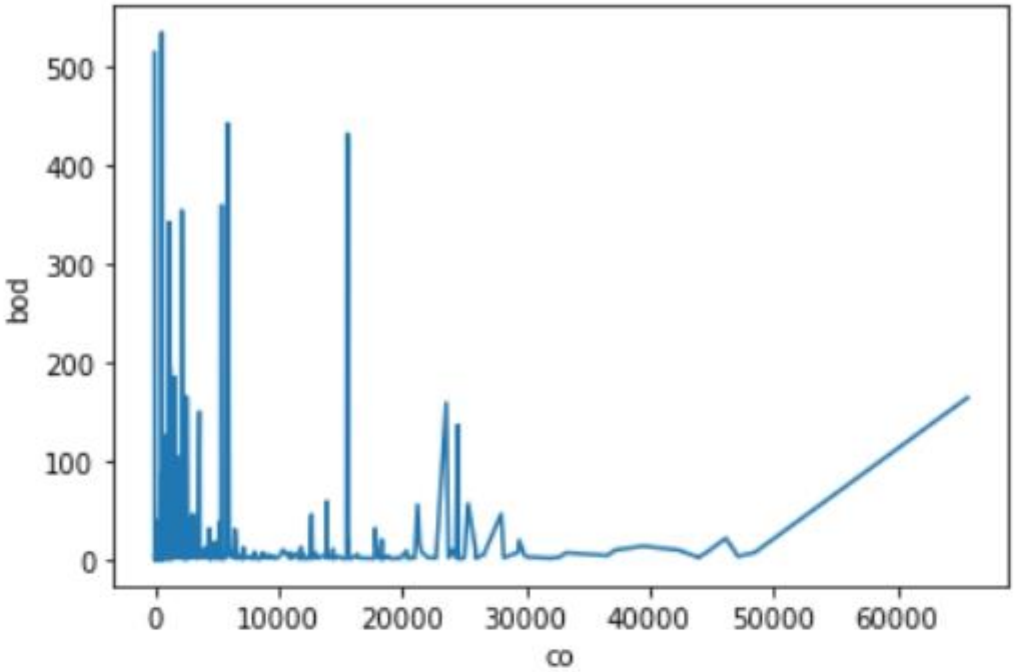
a)Line plot

In [42]:

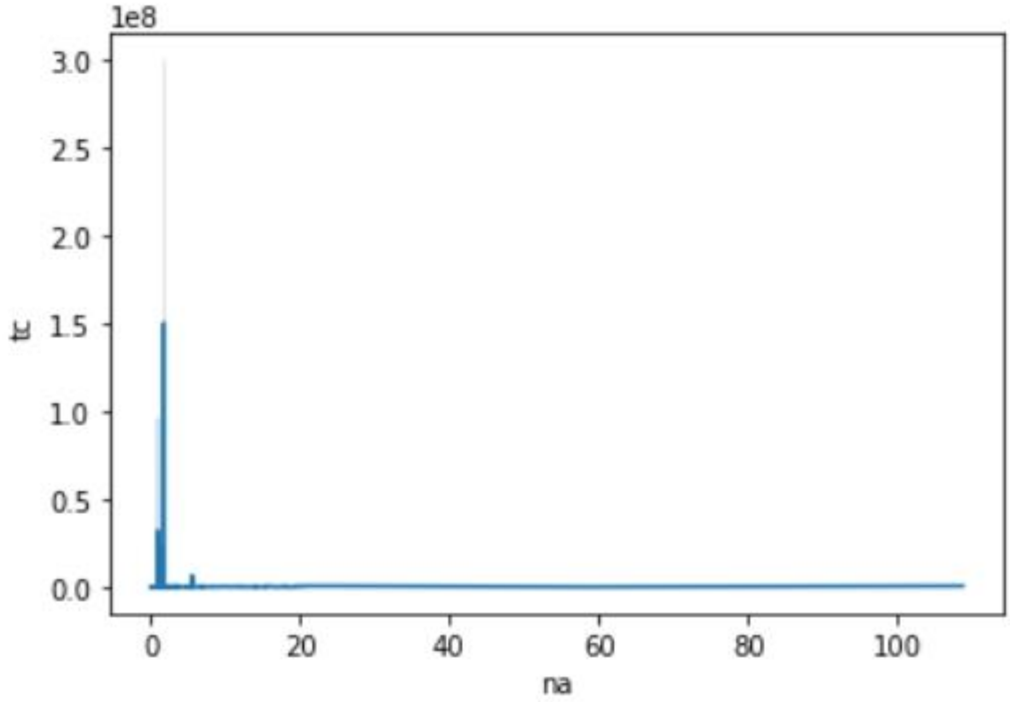
```
sns.lineplot(data.ph,data.do)

plt.show()
```



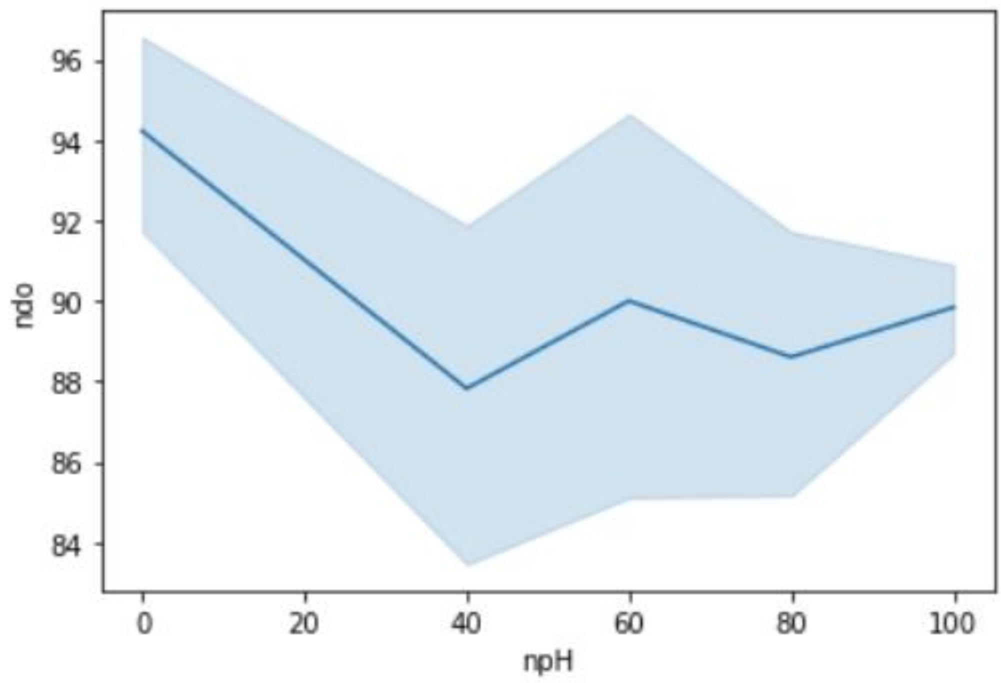
```
sns.lineplot(data.co,data.bod)

plt.show()
```




```
sns.lineplot(data.na,data.tc)

plt.show()
```

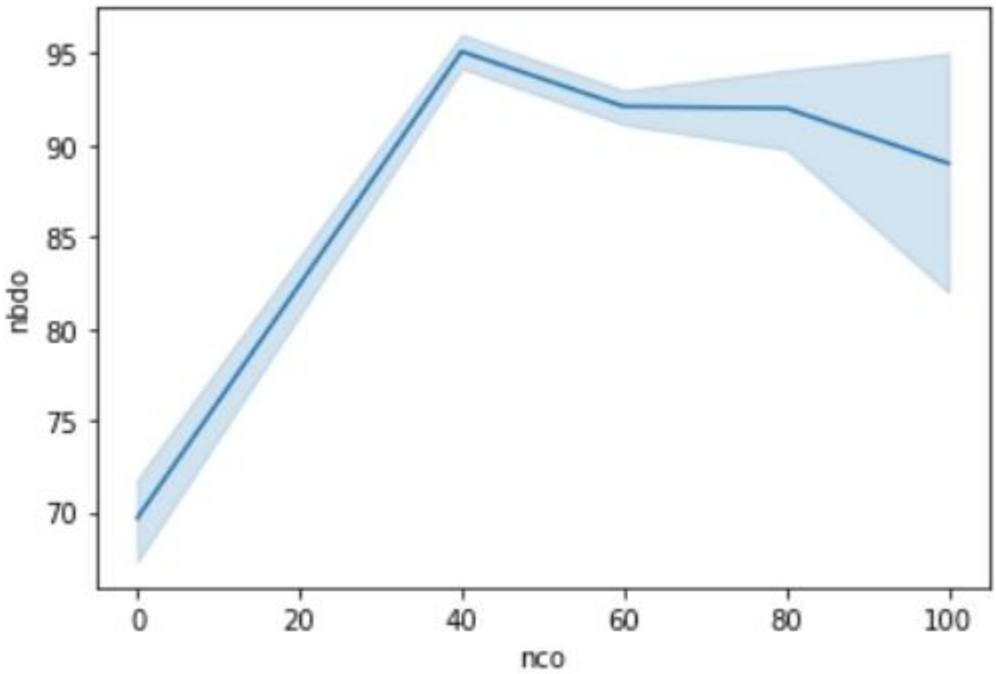



```
sns.lineplot(data.npH,data.ndo)

plt.show()
```



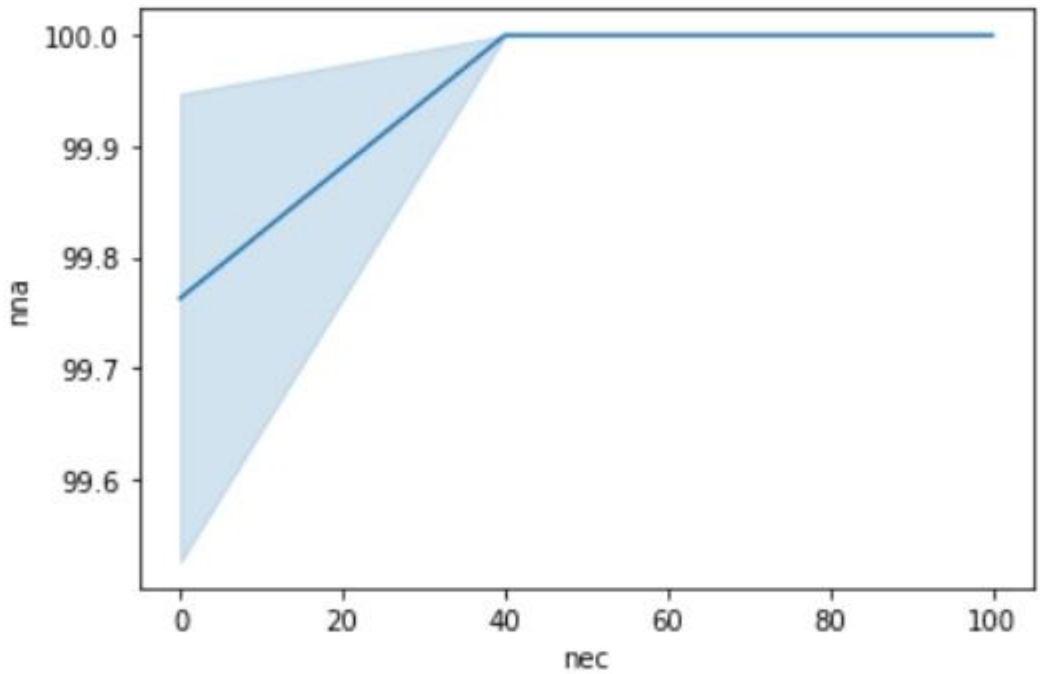

```
sns.lineplot(data.nco,data.nbdo)
```

plt.show()



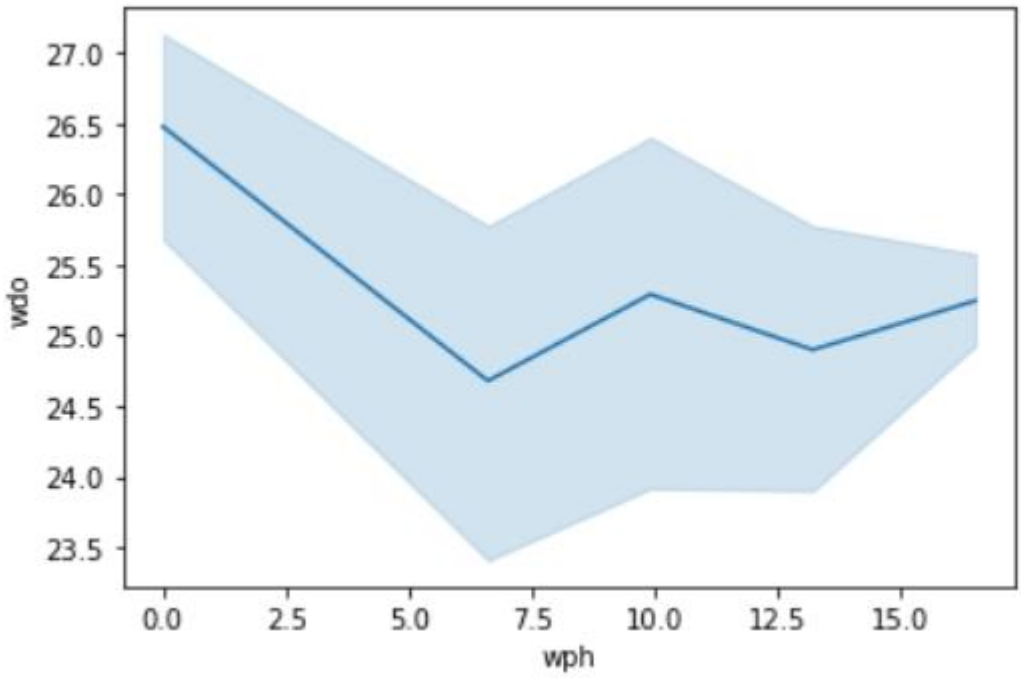
sns.lineplot(data.nec,data.nna)

plt.show()



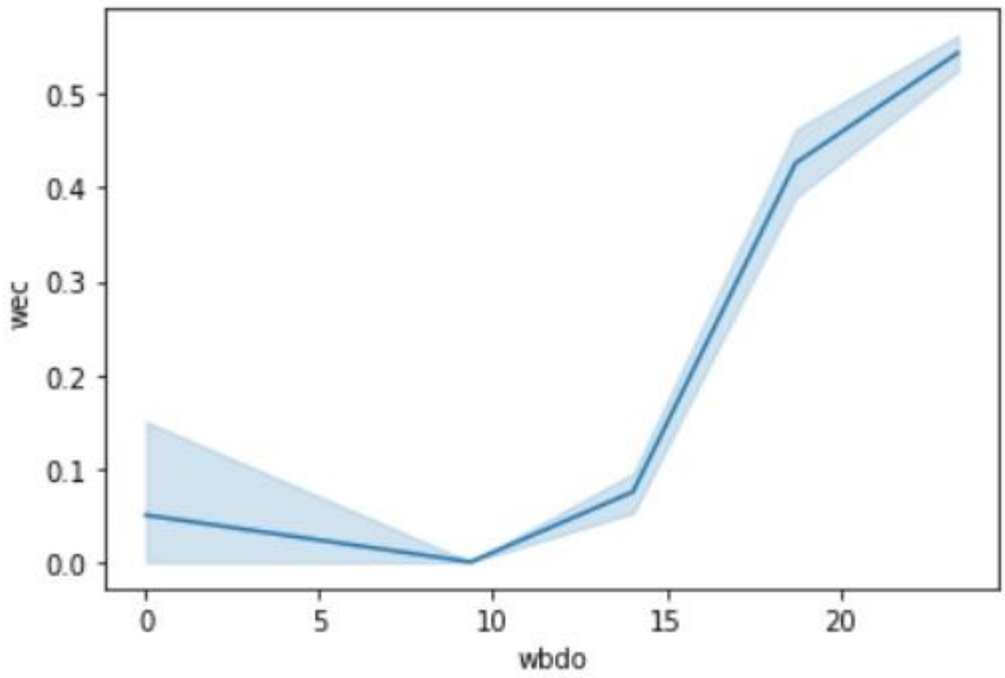
sns.lineplot(data.wph,data.wdo)

plt . show()



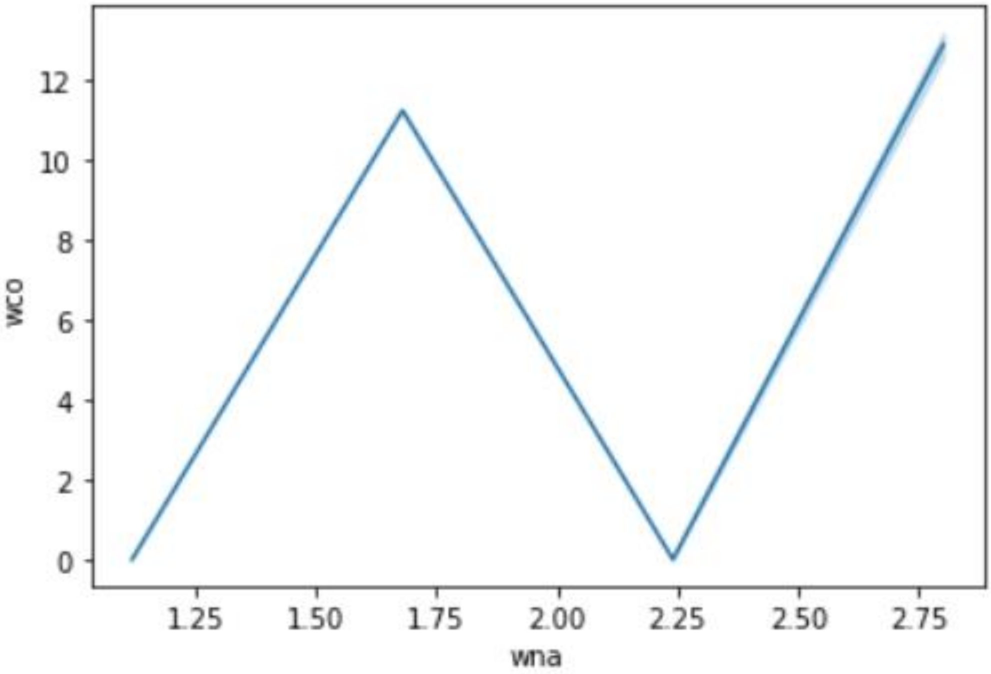
sns . lineplot(data . wbdo,data . wec)

plt . show()



sns . lineplot(data . wna,data . wco)

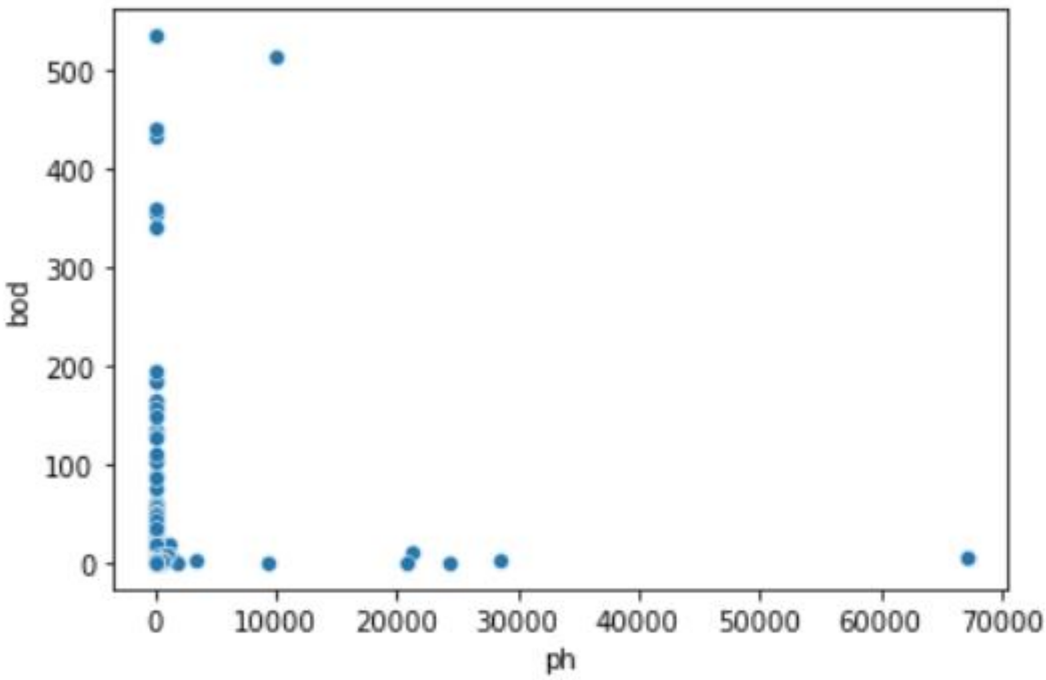
plt.show()



b)Scatter plot

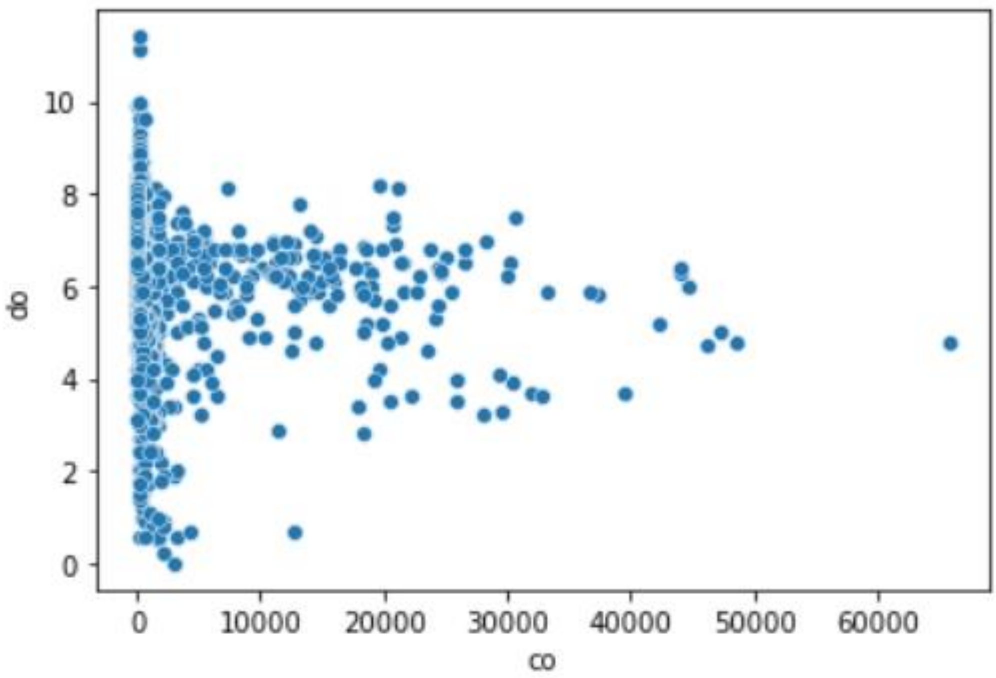
In [51]:

```
sns.scatterplot(data.ph,data.bod)

plt.show()
```



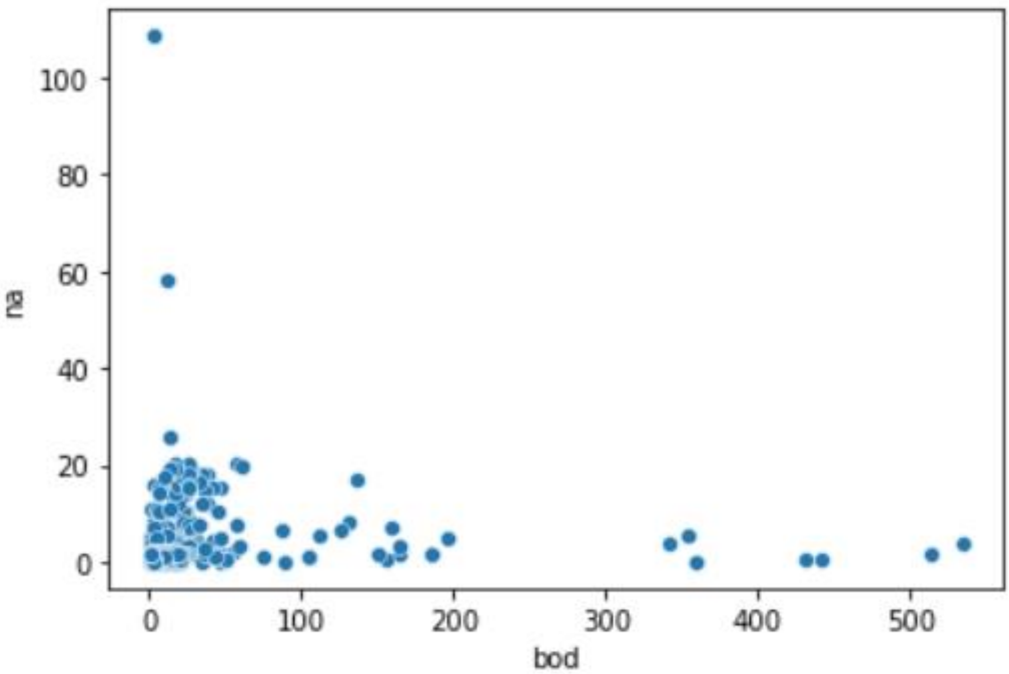
sns.scatterplot(data.co,data.do)

plt.show()



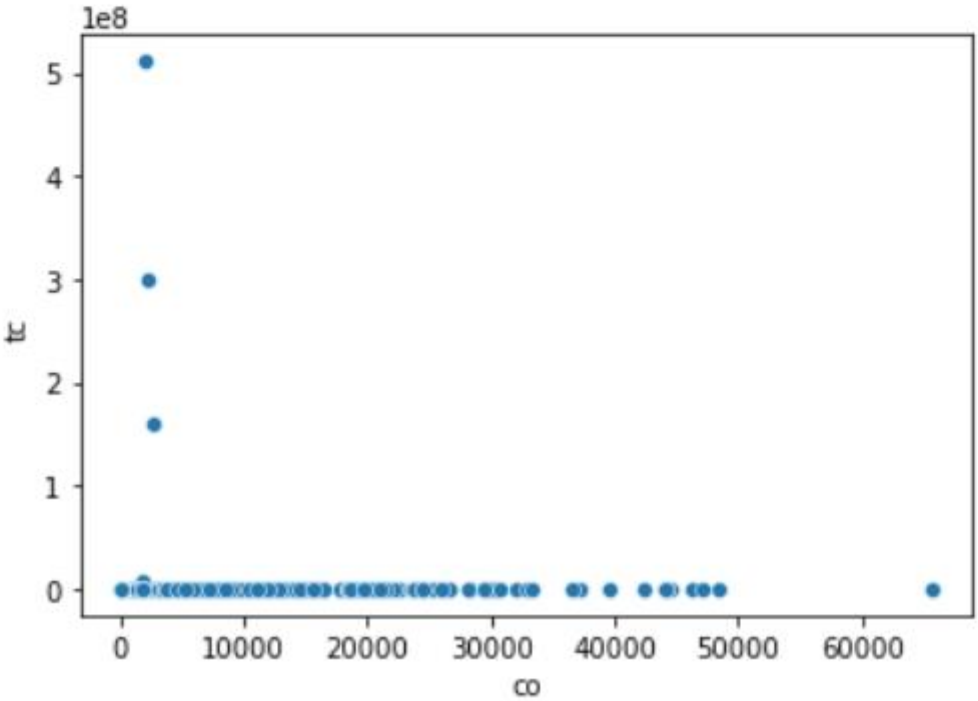
sns.scatterplot(data.bod,data.na)

plt.show()



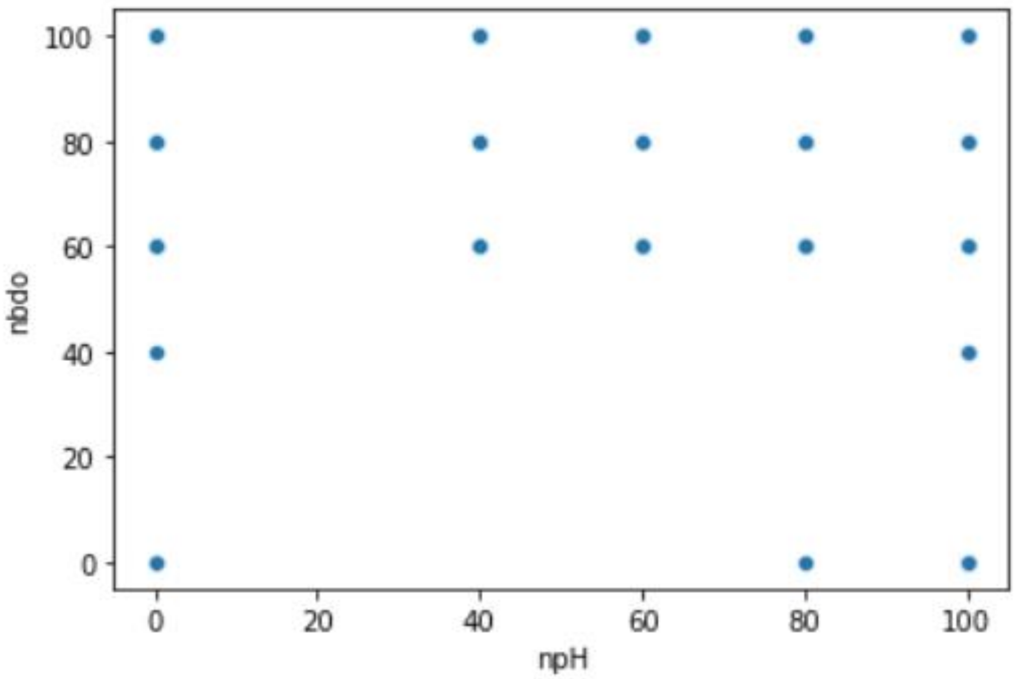
sns.scatterplot(data.co,data.tc)

plt.show()

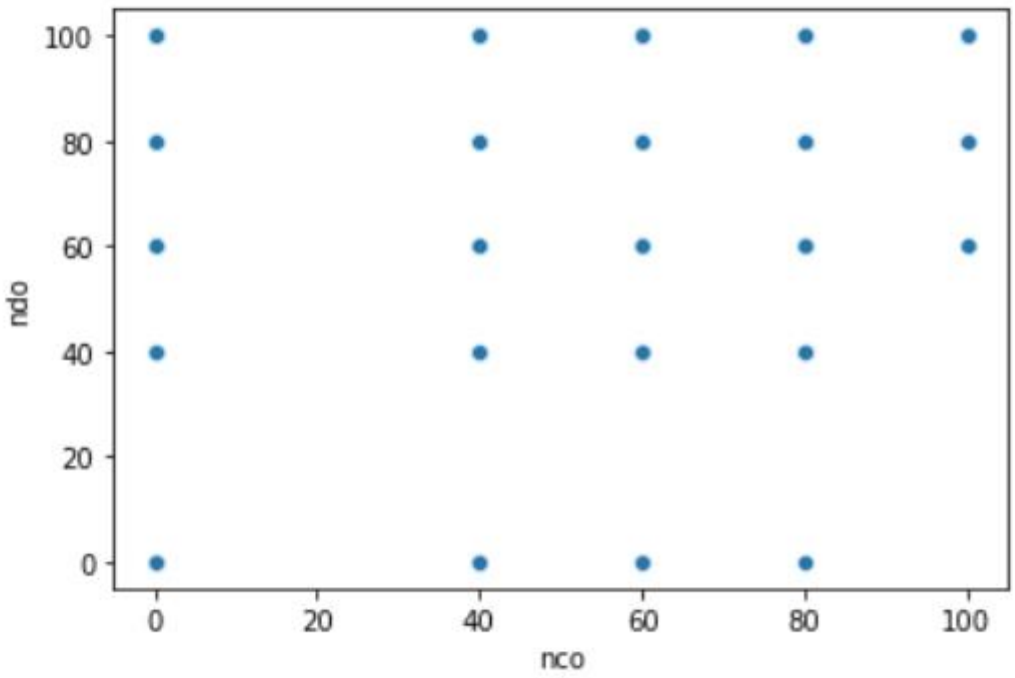


sns.scatterplot(data.npH,data.nbdo)

```
plt.show()
```

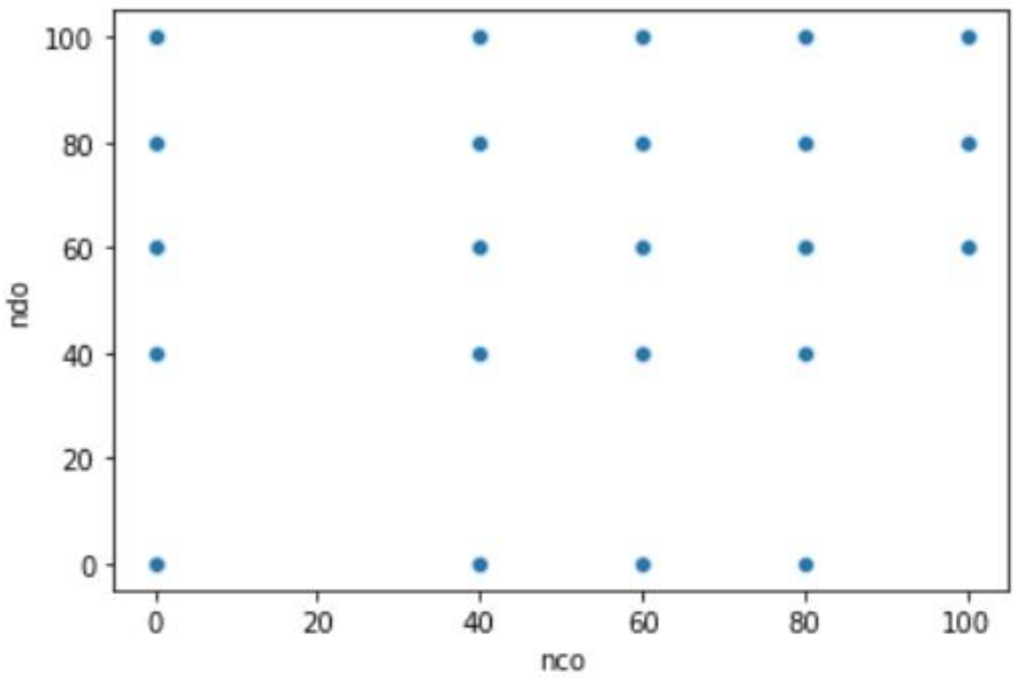



```
sns.scatterplot(data.nco,data.nna)

plt.show()
```



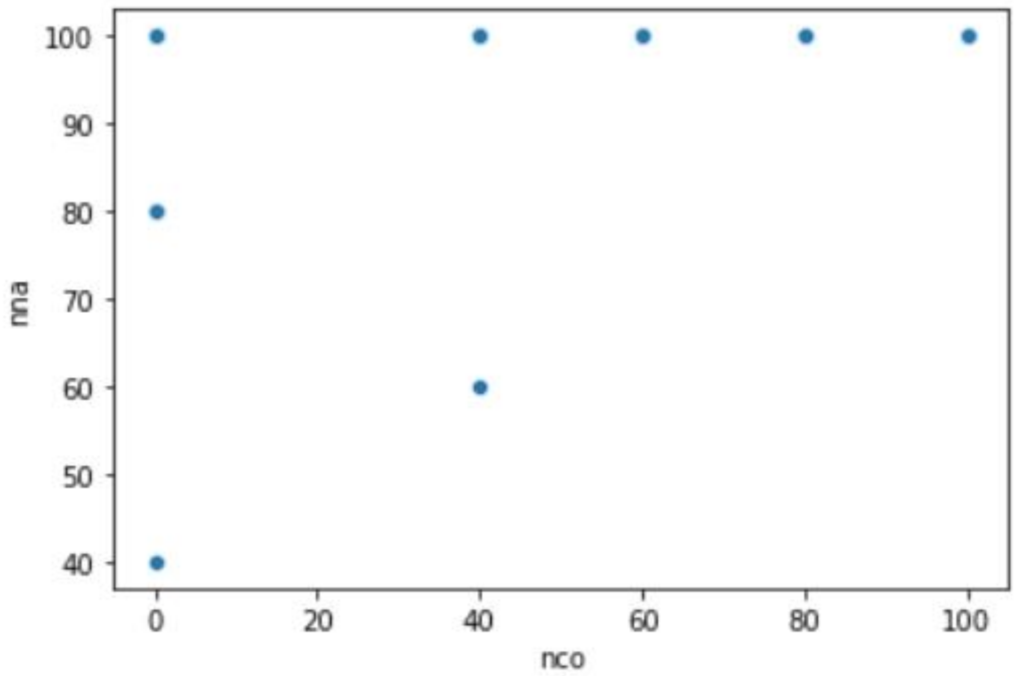
sns.scatterplot(data.nco,data.nna)

plt.show()



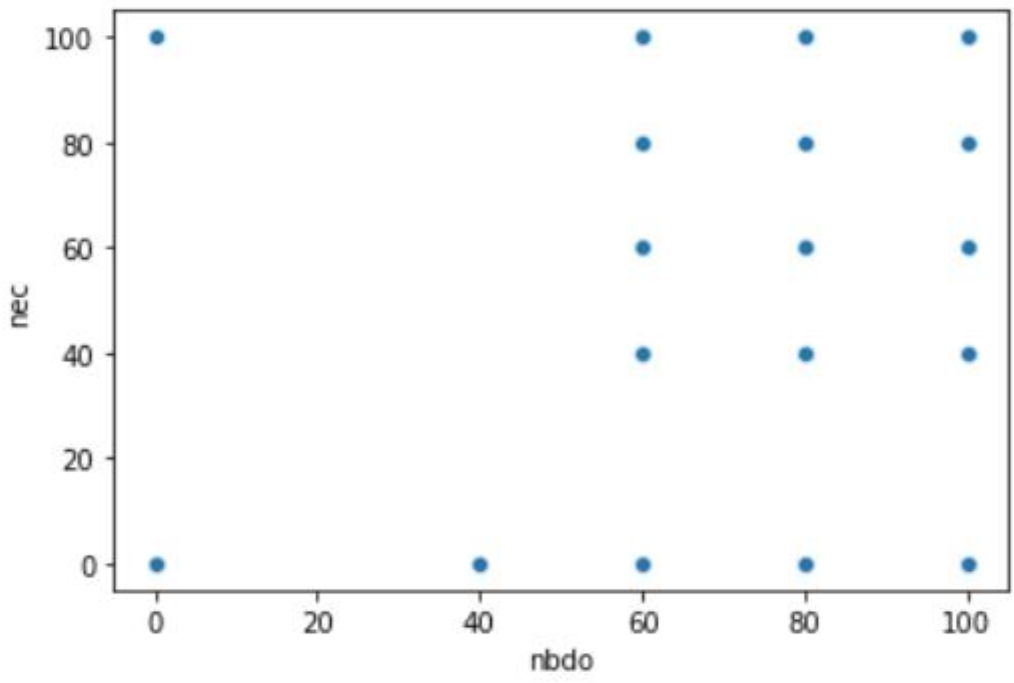
sns.scatterplot(data.nco,data.nna)

plt.show()



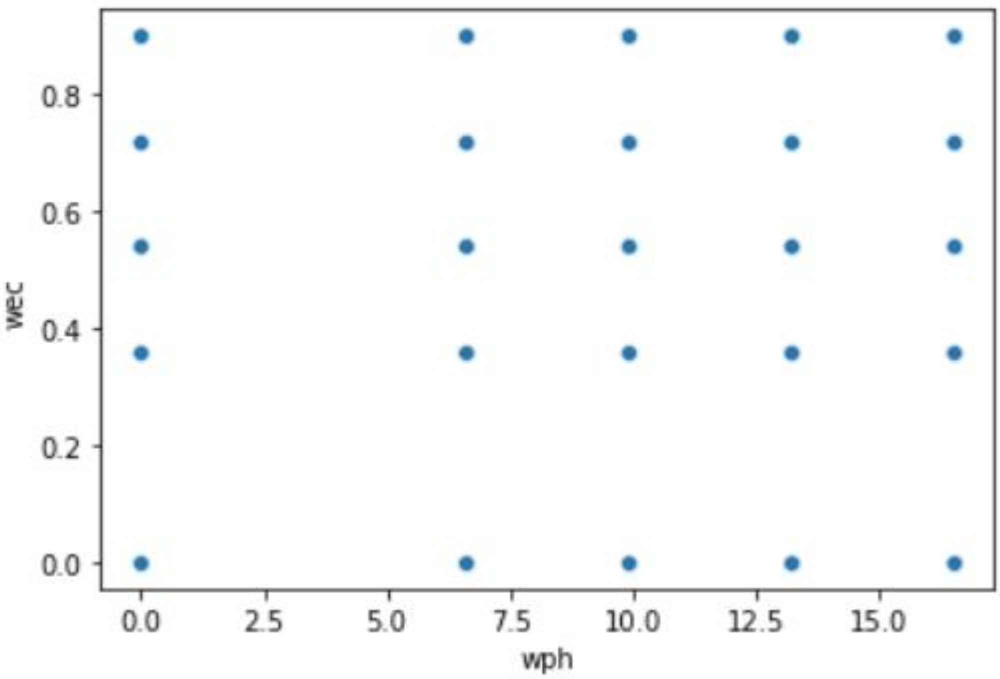
sns.scatterplot(data.nbdo,data.nec)

plt.show()



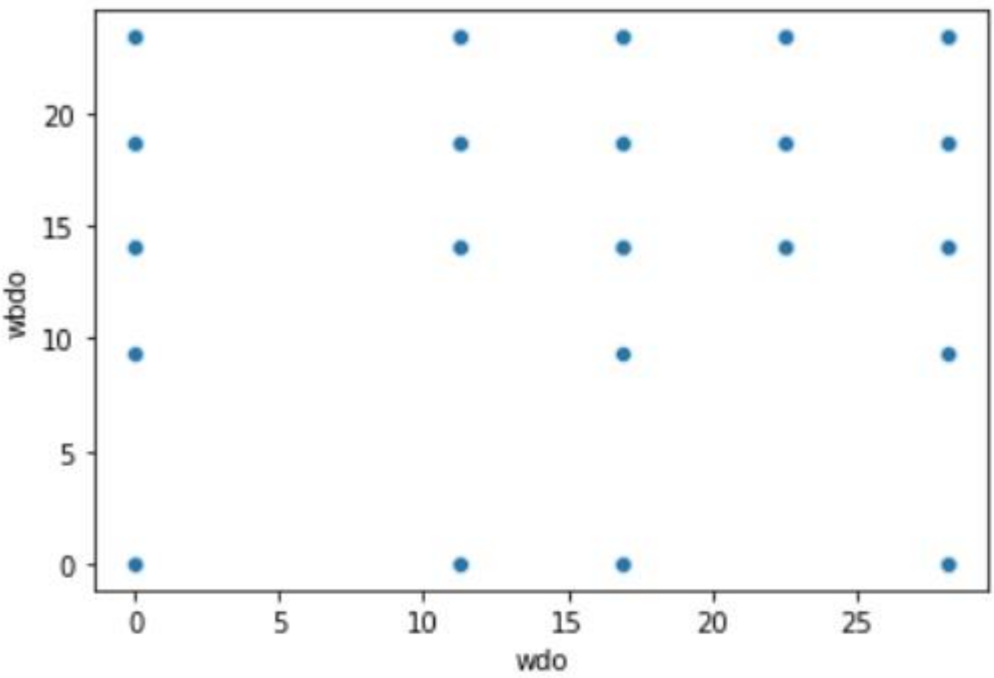
sns.scatterplot(data.wph,data.wec)

plt.show()



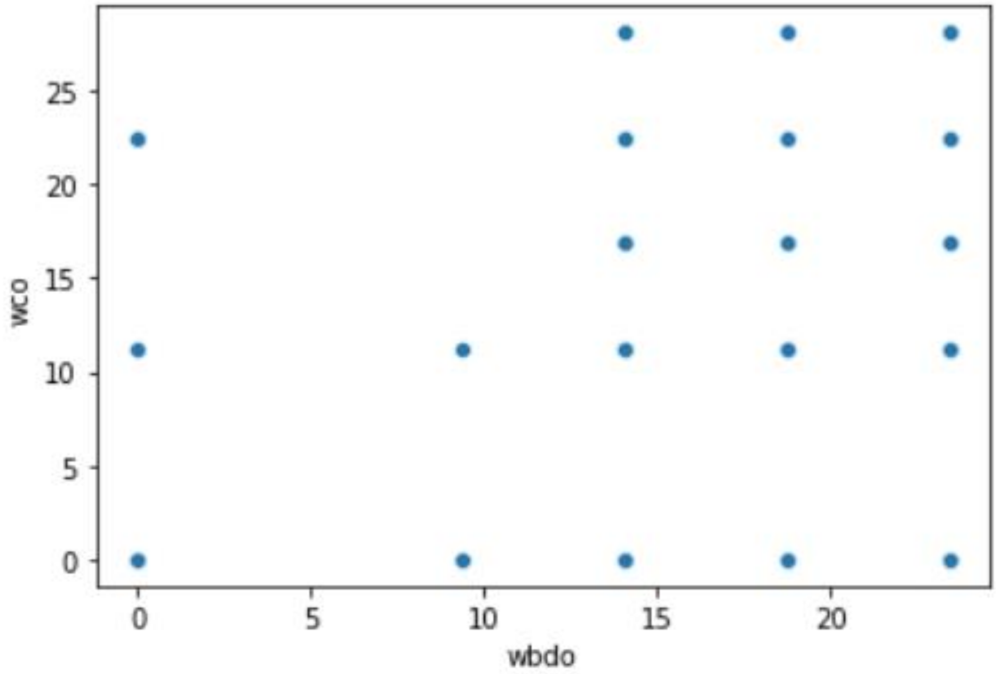
sns.scatterplot(data.wdo,data.wbdo)

plt.show()



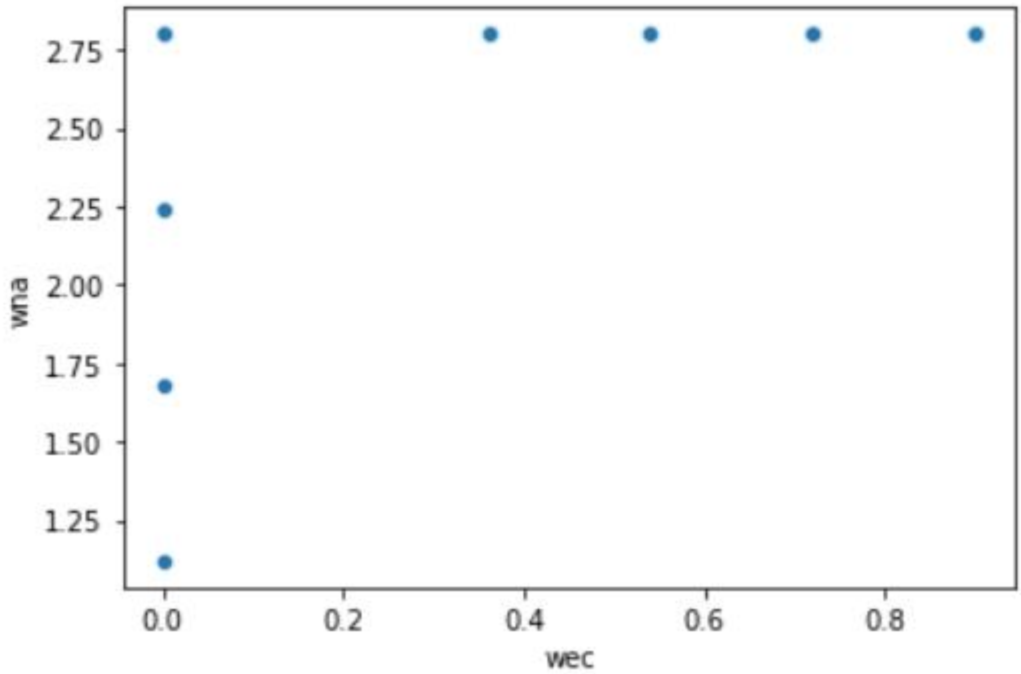
sns.scatterplot(data.wbdo,data.wco)

plt.show()



sns.scatterplot(data.wec,data.wna)

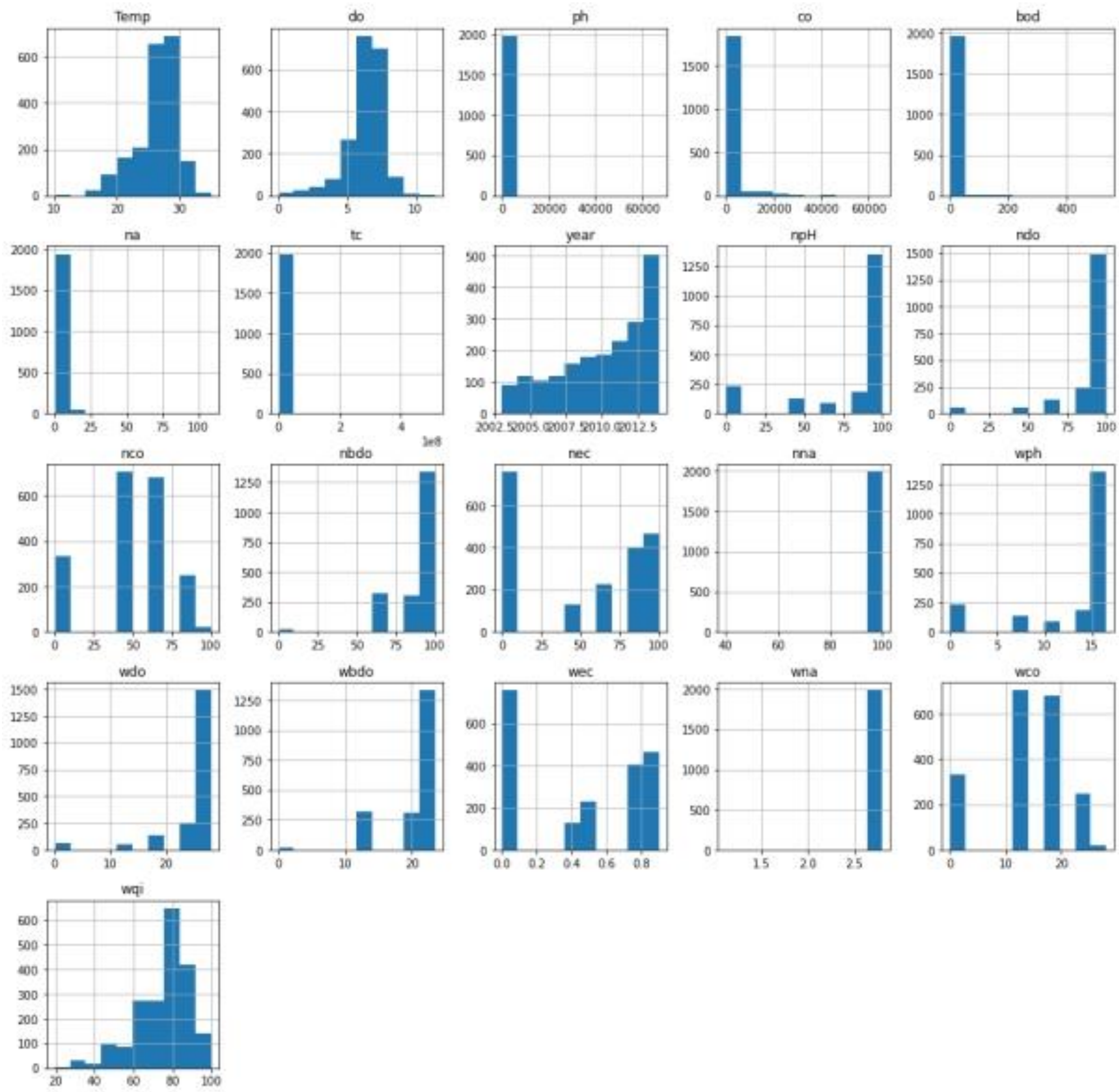
plt.show()



Multivariate analysis

In [63]:

```
data.hist(figsize=(17,17))

plt.show()
```



Label Encoding

In [64]:

```python
from sklearn.preprocessing import LabelEncoder
```

In [65]:

```python
le=LabelEncoder()
```

In [66]:

```python
data.location=le.fit_transform(data.location)

data.state=le.fit_transform(data.state)

data.head()
```

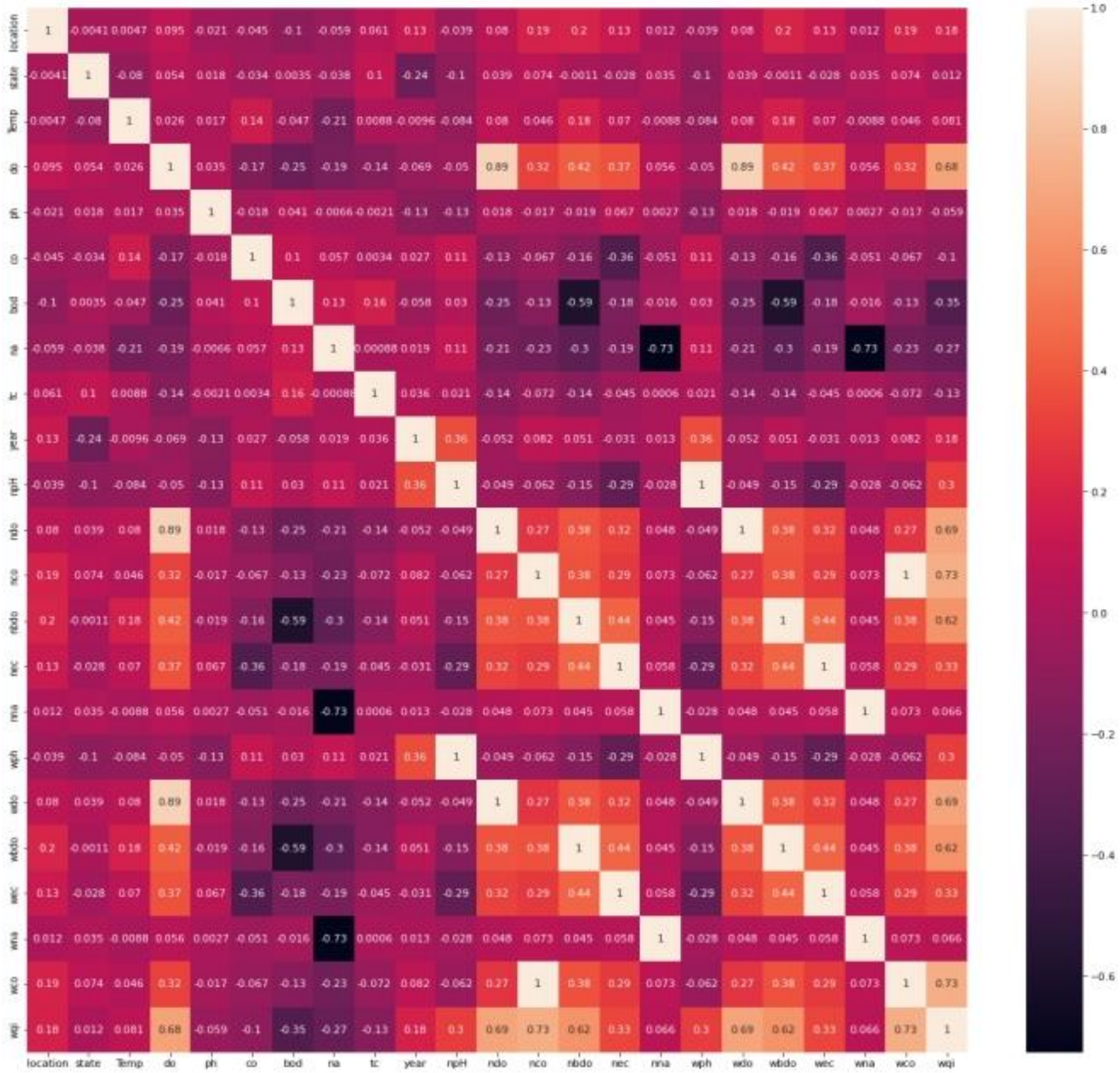
stationlocationstateTempdophcobodnatc...nbdonecnnawphwdowbdowecwnawcowqi01393832130.66.77.5203.06.9400490.127.0...606010016.528.1014.040.542.822.4884.46113996645129.85.77.2189.02.0000000.28391.0...1006010016.522.4823.400.542.811.2476.96214756655129.56.36.9179.01.7000000.15330.0...1006010013.228.1023.400.542.811.2479.28331814955129.75.86.964.03.8000000.58443.0...8010010013.222.4818.720.902.811.2469.34431824965129.55.87.383.01.9000000.45500.0...1008010016.522.4823.400.722.811.2477.14

5 rows × 24 columns

Finding correlation matrix using Heatmap

In [67]:

```python
plt.figure(figsize=(20,20))

sns.heatmap(data.corr(),annot=True)

plt.show()
```




```
df=data.drop(['nco','npH','ndo','nbdo','nec','nna','location','state','stat
ion','wph','wdo','wbdo','wec','wna','wco','Temp'],axis=1)
```

In [69]:

```
df
```

dophcobodnatcyearwqi06.77.5203.06.9400490.10000027.0201484.4615.77.2189.02.0000000.2000008391.0201476.9626.36.9179.01.7000000.1000005330.0201479.2835.86.964.03.8000000.5000008443.0201469.3445.87.383.01.9000000.4000005500.0201477.14...........................19867.9738.07.22.7000000.518000202.0200372.0619877.5585.06.32.6000000.155000315.0200372.0619887.698.06.21.2000001.623079570.0200366.4419897.791.06.51.3000001.623079562.0200366.4419907.6110.05.71.1000001.623079546.0200366.44

1991 rows × 8 columns

In [70]:

```
df.to_csv('df')
```

In [71]:

```
df.corr().wqi.sort_values(ascending=False)
```

Out[71]:

```
wqi     1.000000
do      0.678756
year    0.180629
ph     -0.059461
co     -0.104916
tc     -0.133946
na     -0.265051
bod    -0.349332
Name: wqi, dtype: float64
```

Splitting Dependent and Independent Columns

In [69]:

```
data.drop(['location','station','state'],axis =1,inplace=True)
```

In [70]:

```
data.head()
```

Tempdophcobodnatcyearnphndo...nbdonecnnawphwdowbdowecwnawcowqi030.66.77.5203.06.9400490.127.020 14100100...606010016.528.1014.040.542.822.4884.46129.85.77.2189.02.0000000.28391.0201410080...1006010016. 522.4823.400.542.811.2476.96229.56.36.9179.01.7000000.15330.0201480100...1006010013.228.1023.400.542.811.2 479.28329.75.86.964.03.8000000.58443.020148080...8010010013.222.4818.720.902.811.2469.34429.55.87.383.01.9 000000.45500.0201410080...1008010016.522.4823.400.722.811.2477.14

5 rows × 21 columns

In [71]:

```
x=df.iloc[:,0:7].values
```

In [72]:

```
x.shape
```

Out[72]:

```
(1991, 7)
```

In [73]:

```
y=df.iloc[:,-1:].values
```

In [74]:

```
y.shape
```

Out[74]:

```
(1991, 1)
```

In [75]:

```
print(x)
```

```
[[6.70000000e+00 7.50000000e+00 2.03000000e+02 ... 1.00000000e-01
  2.70000000e+01 2.01400000e+03]
 [5.70000000e+00 7.20000000e+00 1.89000000e+02 ... 2.00000000e-01
  8.39100000e+03 2.01400000e+03]
 [6.30000000e+00 6.90000000e+00 1.79000000e+02 ... 1.00000000e-01
  5.33000000e+03 2.01400000e+03]
 ...
 [7.60000000e+00 9.80000000e+01 6.20000000e+00 ... 1.62307871e+00
  5.70000000e+02 2.00300000e+03]
 [7.70000000e+00 9.10000000e+01 6.50000000e+00 ... 1.62307871e+00
  5.62000000e+02 2.00300000e+03]
 [7.60000000e+00 1.10000000e+02 5.70000000e+00 ... 1.62307871e+00
  5.46000000e+02 2.00300000e+03]]
```

In [76]:

```
print(y)
```

```
[[84.46]
 [76.96]
 [79.28]
 ...
 [66.44]
 [66.44]
 [66.44]]
```

Splitting the Data into Train and Test

In [77]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size =
0.2,random_state=10)
```

In [78]:

```
#Feature Scaling
#from sklearn.preprocessing import StandardScaler
#sc = StandardScaler()
#x_train = sc.fit_transform(x_train)
#x_test = sc.transform(x_test)
```

In [79]:

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
```

Model Evaluation

In [80]:

```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,y_pred))
print('MSE:',metrics.mean_squared_error(y_test,y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
MAE: 0.9425563909774494
MSE: 5.63627572932331
RMSE: 2.374084187497004
```

In [81]:

```
metrics.r2_score(y_test, y_pred)
```

Out[81]:

```
0.9692766700278257
```

In [82]:

```
import pickle
pickle.dump(regressor,open('wqi.pkl','wb'))
model=pickle.load(open('wqi.pkl','rb'))
```

In [83]:

```
regressor.predict([[5.7,7.2,189.0,2.000000,0.200000,8391.0,2014]])
```

Out[83]:

```
array([76.47])
```

In [84]:

```
regressor.predict([[6.7,7.5,203.0,6.940049,0.1,27.0,2014]])
```

Out[84]:

```
array([85.306])
```

In [83]: