In [ ]:

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving city_day.csv to city_day.csv
```

In [ ]:

```
import io
import pandas as pd


rawdata_df = pd.read_csv(io.BytesIO(uploaded['city_day.csv']),sep = ',')
```

In [ ]:

```
rawdata_df
```

Out[ ]:

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benze
0	Ahmedabad	2015-01-01	NaN	NaN	0.92	18.22	17.15	NaN	0.92	27.64	133.36	0.
1	Ahmedabad	2015-01-02	NaN	NaN	0.97	15.69	16.46	NaN	0.97	24.55	34.06	3.
2	Ahmedabad	2015-01-03	NaN	NaN	17.40	19.30	29.70	NaN	17.40	29.07	30.70	6.
3	Ahmedabad	2015-01-04	NaN	NaN	1.70	18.48	17.97	NaN	1.70	18.59	36.08	4.
4	Ahmedabad	2015-01-05	NaN	NaN	22.10	21.42	37.76	NaN	22.10	39.33	39.31	7.
...	...	...	...	...	...	...	...	...	...	...	...	
29526	Visakhapatnam	2020-06-27	15.02	50.94	7.68	25.06	19.54	12.47	0.47	8.55	23.30	2.
29527	Visakhapatnam	2020-06-28	24.38	74.09	3.42	26.06	16.53	11.99	0.52	12.72	30.14	0.
29528	Visakhapatnam	2020-06-29	22.91	65.73	3.45	29.53	18.33	10.71	0.48	8.42	30.96	0.
29529	Visakhapatnam	2020-06-30	16.64	49.97	4.05	29.26	18.80	10.03	0.52	9.84	28.30	0.
29530	Visakhapatnam	2020-07-01	15.00	66.00	0.40	26.85	14.05	5.20	0.59	2.10	17.05	Na

29531 rows × 16 columns

In [ ]:

```
rawdata_df.columns
```

Out[ ]:

```
Index(['City', 'Date', 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2',
       'O3', 'Benzene', 'Toluene', 'Xylene', 'AQI', 'AQI_Bucket'],
      dtype='object')
```

In [ ]:

```
selected_columns = ['City', 'Date', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3', 'B
```

```
analysis_df = rawdata_df[selected_columns].copy()
analysis_df
```

Out[ ]:

	City	Date	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xy
0	Ahmedabad	2015-01-01	0.92	18.22	17.15	NaN	0.92	27.64	133.36	0.00	0.02	
1	Ahmedabad	2015-01-02	0.97	15.69	16.46	NaN	0.97	24.55	34.06	3.68	5.50	
2	Ahmedabad	2015-01-03	17.40	19.30	29.70	NaN	17.40	29.07	30.70	6.80	16.40	
3	Ahmedabad	2015-01-04	1.70	18.48	17.97	NaN	1.70	18.59	36.08	4.43	10.14	
4	Ahmedabad	2015-01-05	22.10	21.42	37.76	NaN	22.10	39.33	39.31	7.01	18.89	
...	...	...	...	...	...	...	...	...	...	...	...	
29526	Visakhapatnam	2020-06-27	7.68	25.06	19.54	12.47	0.47	8.55	23.30	2.24	12.07	
29527	Visakhapatnam	2020-06-28	3.42	26.06	16.53	11.99	0.52	12.72	30.14	0.74	2.21	
29528	Visakhapatnam	2020-06-29	3.45	29.53	18.33	10.71	0.48	8.42	30.96	0.01	0.01	
29529	Visakhapatnam	2020-06-30	4.05	29.26	18.80	10.03	0.52	9.84	28.30	0.00	0.00	
29530	Visakhapatnam	2020-07-01	0.40	26.85	14.05	5.20	0.59	2.10	17.05	NaN	NaN	

29531 rows × 14 columns

In [ ]:

```
analysis_df.shape
```

Out[ ]:

```
(29531, 14)
```

In [ ]:

```
analysis_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   City        29531 non-null  object 
 1   Date        29531 non-null  object 
 2   NO          25949 non-null  float64
 3   NO2         25946 non-null  float64
 4   NOx         25346 non-null  float64
 5   NH3         19203 non-null  float64
 6   CO          27472 non-null  float64
 7   SO2         25677 non-null  float64
 8   O3          25509 non-null  float64
 9   Benzene     23908 non-null  float64
 10  Toluene     21490 non-null  float64
 11  Xylene      11422 non-null  float64
 12  AQI         24850 non-null  float64
 13  AQI_Bucket  24850 non-null  object 
dtypes: float64(11), object(3)
memory usage: 3.2+ MB
```

In [ ]:

```
analysis_df.describe()
```

Out[ ]:

	NO	NO2	NOx	NH3	CO	SO2	
count	25949.000000	25946.000000	25346.000000	19203.000000	27472.000000	25677.000000	25509.00
mean	17.574730	28.560659	32.309123	23.483476	2.248598	14.531977	34.4
std	22.785846	24.474746	31.646011	25.684275	6.962884	18.133775	21.6
min	0.020000	0.010000	0.000000	0.010000	0.000000	0.010000	0.0
25%	5.630000	11.750000	12.820000	8.580000	0.510000	5.670000	18.8
50%	9.890000	21.690000	23.520000	15.850000	0.890000	9.160000	30.8
75%	19.950000	37.620000	40.127500	30.020000	1.450000	15.220000	45.5
max	390.680000	362.210000	467.630000	352.890000	175.810000	193.860000	257.7

In [ ]:

```
analysis_df.dropna(subset=['AQI'], inplace=True)
analysis_df.dropna(subset=['AQI_Bucket'], inplace=True)
analysis_df
```

Out[ ]:

	City	Date	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xy
28	Ahmedabad	2015-01-29	6.93	28.71	33.72	NaN	6.93	49.52	59.76	0.02	0.00	
29	Ahmedabad	2015-01-30	13.85	28.68	41.08	NaN	13.85	48.49	97.07	0.04	0.00	
30	Ahmedabad	2015-01-31	24.39	32.66	52.61	NaN	24.39	67.39	111.33	0.24	0.01	
31	Ahmedabad	2015-02-01	43.48	42.08	84.57	NaN	43.48	75.23	102.70	0.40	0.04	2
32	Ahmedabad	2015-02-02	54.56	35.31	72.80	NaN	54.56	55.04	107.38	0.46	0.06	3
...	...	...	...	...	...	...	...	...	...	...	...	
29526	Visakhapatnam	2020-06-27	7.68	25.06	19.54	12.47	0.47	8.55	23.30	2.24	12.07	
29527	Visakhapatnam	2020-06-28	3.42	26.06	16.53	11.99	0.52	12.72	30.14	0.74	2.21	
29528	Visakhapatnam	2020-06-29	3.45	29.53	18.33	10.71	0.48	8.42	30.96	0.01	0.01	
29529	Visakhapatnam	2020-06-30	4.05	29.26	18.80	10.03	0.52	9.84	28.30	0.00	0.00	
29530	Visakhapatnam	2020-07-01	0.40	26.85	14.05	5.20	0.59	2.10	17.05	NaN	NaN	

24850 rows × 14 columns

In [ ]:

```
analysis_df['Date']
```

Out[ ]:

```
28       2015-01-29
29       2015-01-30
30       2015-01-31
31       2015-02-01
32       2015-02-02
            ...
29526    2020-06-27
29527    2020-06-28
29528    2020-06-29
29529    2020-06-30
29530    2020-07-01
Name: Date, Length: 24850, dtype: object
```

In [ ]:

```
analysis_df['Date'] = pd.to_datetime(analysis_df.Date)
analysis_df['Date']
```

Out[ ]:

```
28      2015-01-29
29      2015-01-30
30      2015-01-31
31      2015-02-01
32      2015-02-02
           ...
29526   2020-06-27
29527   2020-06-28
29528   2020-06-29
29529   2020-06-30
29530   2020-07-01
Name: Date, Length: 24850, dtype: datetime64[ns]
```

In [ ]:

```
analysis_df['Year'] = pd.DatetimeIndex(analysis_df['Date']).year
analysis_df['month'] = pd.DatetimeIndex(analysis_df['Date']).month

analysis_df
```

Out[ ]:

	City	Date	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xy
28	Ahmedabad	2015-01-29	6.93	28.71	33.72	NaN	6.93	49.52	59.76	0.02	0.00	
29	Ahmedabad	2015-01-30	13.85	28.68	41.08	NaN	13.85	48.49	97.07	0.04	0.00	
30	Ahmedabad	2015-01-31	24.39	32.66	52.61	NaN	24.39	67.39	111.33	0.24	0.01	
31	Ahmedabad	2015-02-01	43.48	42.08	84.57	NaN	43.48	75.23	102.70	0.40	0.04	2
32	Ahmedabad	2015-02-02	54.56	35.31	72.80	NaN	54.56	55.04	107.38	0.46	0.06	3
...	...	...	...	...	...	...	...	...	...	...	...	
29526	Visakhapatnam	2020-06-27	7.68	25.06	19.54	12.47	0.47	8.55	23.30	2.24	12.07	

	City	Date	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xy
29527	Visakhapatnam	2020-06-28	3.42	26.06	16.53	11.99	0.52	12.72	30.14	0.74	2.21	
29528	Visakhapatnam	2020-06-29	3.45	29.53	18.33	10.71	0.48	8.42	30.96	0.01	0.01	
29529	Visakhapatnam	2020-06-30	4.05	29.26	18.80	10.03	0.52	9.84	28.30	0.00	0.00	
29530	Visakhapatnam	2020-07-01	0.40	26.85	14.05	5.20	0.59	2.10	17.05	NaN	NaN	

24850 rows × 16 columns

In [ ]:

```
analysis_df['month_alph'] = pd.to_datetime(analysis_df['month'], format='%m').dt.mon
analysis_df
```

Out[ ]:

	City	Date	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xy
28	Ahmedabad	2015-01-29	6.93	28.71	33.72	NaN	6.93	49.52	59.76	0.02	0.00	
29	Ahmedabad	2015-01-30	13.85	28.68	41.08	NaN	13.85	48.49	97.07	0.04	0.00	
30	Ahmedabad	2015-01-31	24.39	32.66	52.61	NaN	24.39	67.39	111.33	0.24	0.01	
31	Ahmedabad	2015-02-01	43.48	42.08	84.57	NaN	43.48	75.23	102.70	0.40	0.04	2
32	Ahmedabad	2015-02-02	54.56	35.31	72.80	NaN	54.56	55.04	107.38	0.46	0.06	3
...	...	...	...	...	...	...	...	...	...	...	...	
29526	Visakhapatnam	2020-06-27	7.68	25.06	19.54	12.47	0.47	8.55	23.30	2.24	12.07	
29527	Visakhapatnam	2020-06-28	3.42	26.06	16.53	11.99	0.52	12.72	30.14	0.74	2.21	
29528	Visakhapatnam	2020-06-29	3.45	29.53	18.33	10.71	0.48	8.42	30.96	0.01	0.01	
29529	Visakhapatnam	2020-06-30	4.05	29.26	18.80	10.03	0.52	9.84	28.30	0.00	0.00	
29530	Visakhapatnam	2020-07-01	0.40	26.85	14.05	5.20	0.59	2.10	17.05	NaN	NaN	

24850 rows × 17 columns

In [ ]:

```
analysis_df.shape
analysis_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24850 entries, 28 to 29530
```

```
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   City        24850 non-null  object
 1   Date        24850 non-null  datetime64[ns]
 2   NO          24463 non-null  float64
 3   NO2         24459 non-null  float64
 4   NOx         22993 non-null  float64
 5   NH3         18314 non-null  float64
 6   CO          24405 non-null  float64
 7   SO2         24245 non-null  float64
 8   O3          24043 non-null  float64
 9   Benzene     21315 non-null  float64
 10  Toluene     19024 non-null  float64
 11  Xylene      9478 non-null   float64
 12  AQI         24850 non-null  float64
 13  AQI_Bucket  24850 non-null  object
 14  Year        24850 non-null  int64
 15  month       24850 non-null  int64
 16  month_alph  24850 non-null  object
dtypes: datetime64[ns](1), float64(11), int64(2), object(3)
memory usage: 3.4+ MB
```

In [ ]:

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

sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (9, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

In [ ]:

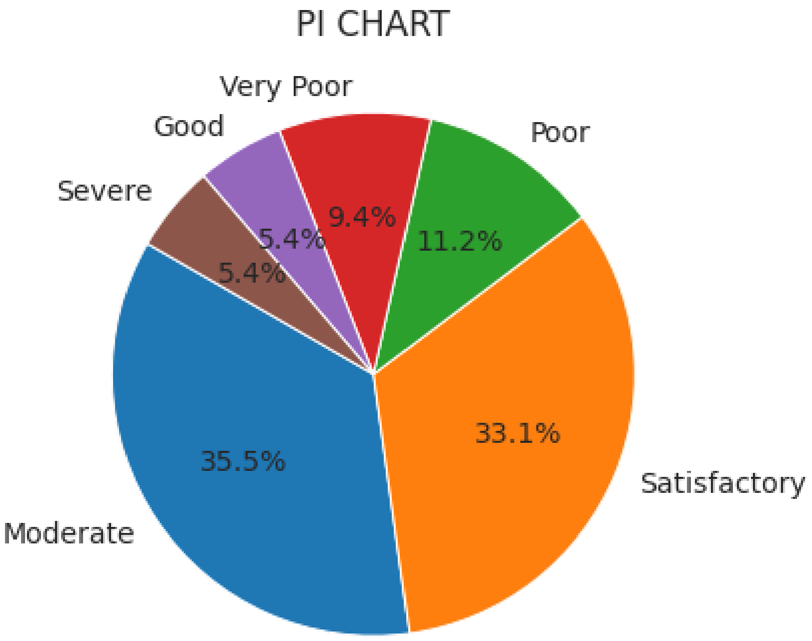
```
AQI_Bucket_distribution = analysis_df.AQI_Bucket.value_counts()
AQI_Bucket_distribution
```

Out[ ]:

```
Moderate        8829
Satisfactory    8224
Poor            2781
Very Poor       2337
Good            1341
Severe          1338
Name: AQI_Bucket, dtype: int64
```

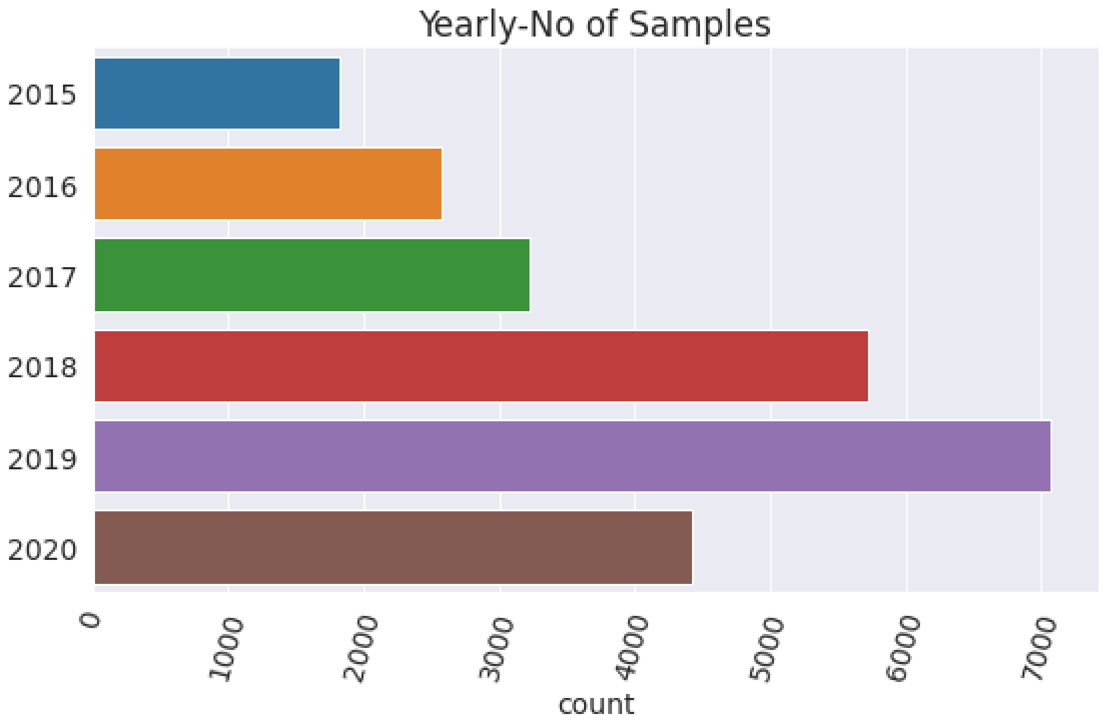
In [ ]:

```
plt.figure(figsize=(10,6))
plt.title('PI CHART')
plt.pie(AQI_Bucket_distribution, labels=AQI_Bucket_distribution.index, autopct='%1.1
```



In [ ]:

```
sns.countplot(y=analysis_df.Year)
plt.xticks(rotation=75);
plt.title('Yearly-No of Samples')
plt.ylabel(None);
```



In [ ]:

```
selected_columns1 = ['AQI', 'month','month_alph']
mon_df = analysis_df[selected_columns1].copy()
mon_df
```

Out[ ]:

	AQI	month	month_alph
28	209.0	1	Jan
29	328.0	1	Jan

	AQI	month	month_alph
30	514.0	1	Jan
31	782.0	2	Feb
32	914.0	2	Feb
...	...	...	...
29526	41.0	6	Jun
29527	70.0	6	Jun
29528	68.0	6	Jun
29529	54.0	6	Jun
29530	50.0	7	Jul

24850 rows × 3 columns

In [ ]:

```
month_df = mon_df.groupby('month_alph').mean()
```

In [ ]:

```
selected_columns2 = ['AQI', 'month']
month_df1 = month_df[selected_columns2].copy()
month_df1 = month_df1.sort_values(by='month', ascending=True)
```

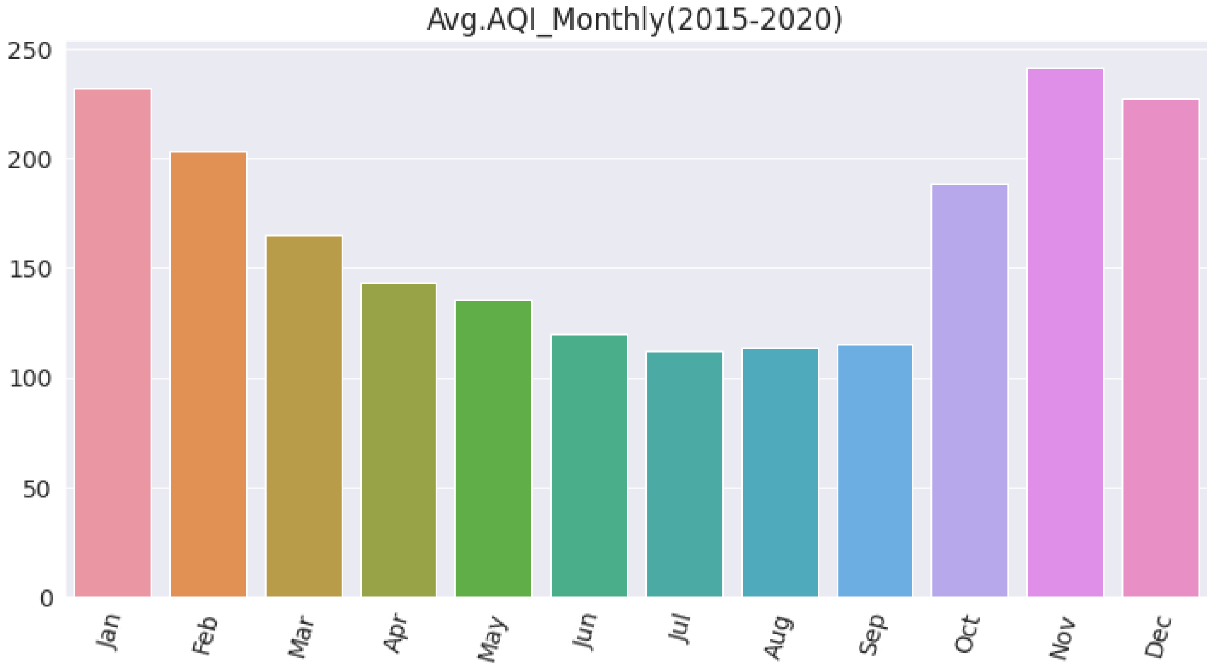
In [ ]:

```
X1 = list(month_df1.index)
X2 = month_df1["AQI"].tolist()
```

In [ ]:

```
plt.figure(figsize=(12,6))
plt.xticks(rotation=75)
plt.title('Avg.AQI_Monthly(2015-2020)')
#sns.barplot(month_df.index,month_df);
sns.barplot(x = X1,
            y = X2,)

# Show the plot
plt.show()
```



Which city is Worse and good for 2017 year?

In [ ]:

```
year2017 = analysis_df[analysis_df.Year == 2017]

year2017
```

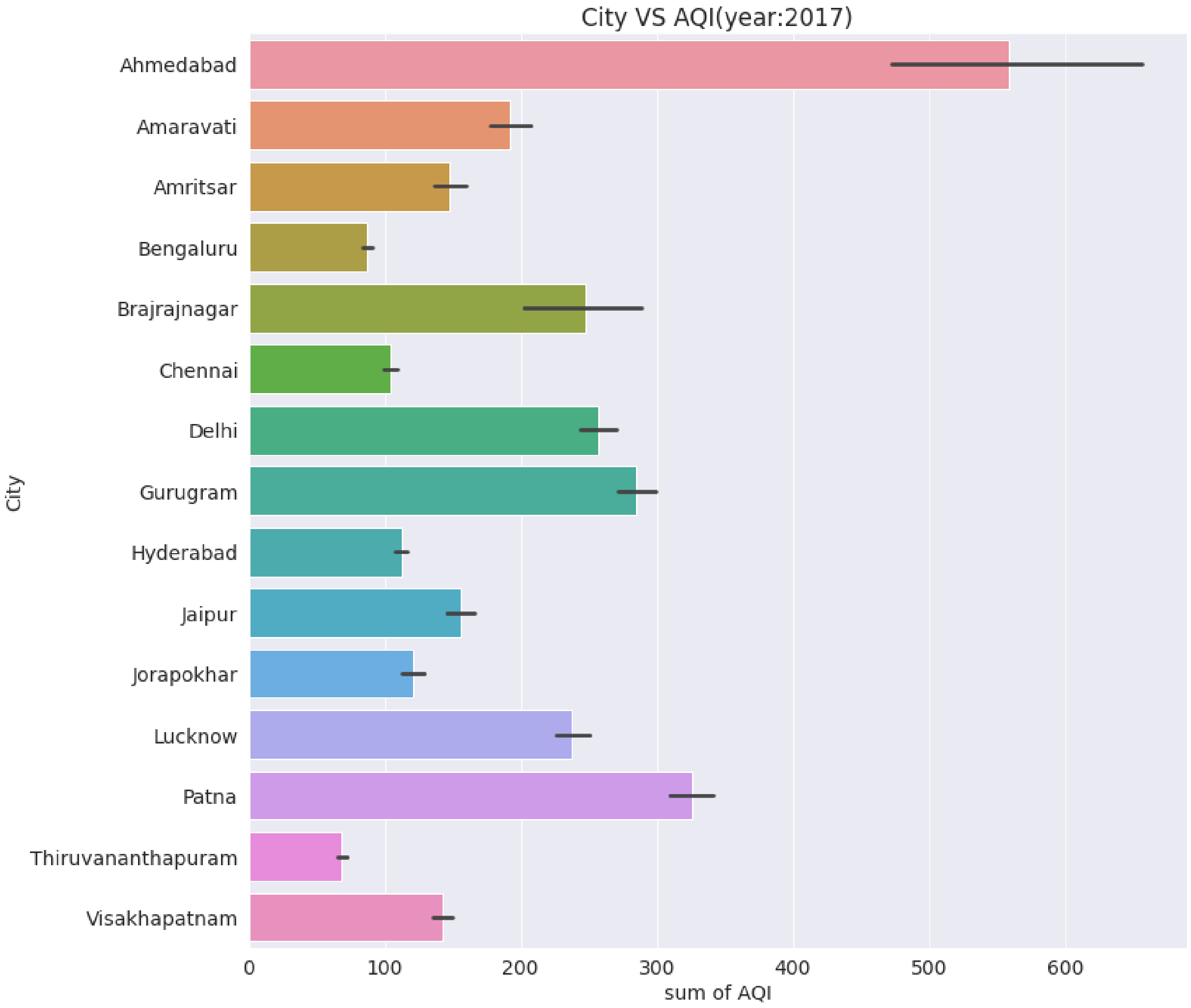
Out[ ]:

	City	Date	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xyl
1014	Ahmedabad	2017-10-11	NaN	NaN	36.60	NaN	NaN	NaN	9.73	4.67	15.99	
1024	Ahmedabad	2017-10-21	NaN	NaN	46.52	NaN	NaN	NaN	27.85	11.85	22.28	
1025	Ahmedabad	2017-10-22	NaN	NaN	28.25	NaN	NaN	NaN	26.22	5.08	10.27	
1026	Ahmedabad	2017-10-23	NaN	NaN	34.62	NaN	NaN	NaN	24.71	5.35	10.47	
1027	Ahmedabad	2017-10-24	NaN	NaN	37.53	NaN	NaN	NaN	19.89	7.54	15.47	
...	...	...	...	...	...	...	...	...	...	...	...	
28613	Visakhapatnam	2017-12-27	15.87	55.93	42.58	14.37	1.20	16.39	69.82	3.76	6.40	
28614	Visakhapatnam	2017-12-28	28.26	60.83	55.30	11.48	1.28	16.54	57.78	4.70	7.63	
28615	Visakhapatnam	2017-12-29	9.07	52.56	35.20	11.33	1.08	6.26	52.57	3.59	6.08	
28616	Visakhapatnam	2017-12-30	2.43	32.45	19.08	12.22	0.93	5.44	80.89	2.76	4.18	
28617	Visakhapatnam	2017-12-31	1.62	23.54	13.63	12.58	0.93	8.38	112.64	2.62	3.18	

3234 rows × 17 columns

In [ ]:

```
plt.figure(figsize=(12, 12))
sns.barplot(x = 'AQI',
            y = 'City',
           data = year2017)
plt.title("City VS AQI(year:2017)");
plt.xlabel('sum of AQI');
```



Which element is contributing more when compare to others?

In [ ]:

```
Elements = ['NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3', 'Benzene', 'Toluene', 'Xyl

Elements_whichimpacting_airquality = rawdata_df[Elements].copy()
Elements_whichimpacting_airquality
```

Out[ ]:

	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene
0	0.92	18.22	17.15	NaN	0.92	27.64	133.36	0.00	0.02	0.00
1	0.97	15.69	16.46	NaN	0.97	24.55	34.06	3.68	5.50	3.77
2	17.40	19.30	29.70	NaN	17.40	29.07	30.70	6.80	16.40	2.25
3	1.70	18.48	17.97	NaN	1.70	18.59	36.08	4.43	10.14	1.00
4	22.10	21.42	37.76	NaN	22.10	39.33	39.31	7.01	18.89	2.78
...	...	...	...	...	...	...	...	...	...	...
29526	7.68	25.06	19.54	12.47	0.47	8.55	23.30	2.24	12.07	0.73
29527	3.42	26.06	16.53	11.99	0.52	12.72	30.14	0.74	2.21	0.38

	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene
29528	3.45	29.53	18.33	10.71	0.48	8.42	30.96	0.01	0.01	0.00
29529	4.05	29.26	18.80	10.03	0.52	9.84	28.30	0.00	0.00	0.00
29530	0.40	26.85	14.05	5.20	0.59	2.10	17.05	NaN	NaN	NaN

29531 rows × 10 columns

In [ ]:

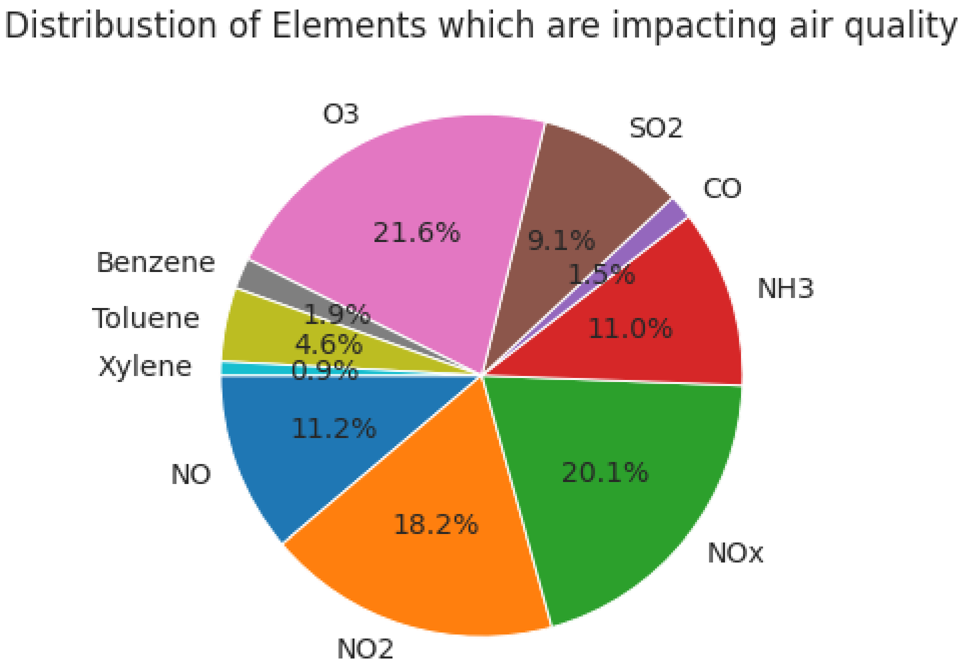
```
element_df = Elements_whichimpacting_airquality.sum()
element_df
```

Out[ ]:

```
NO          456046.66
NO2         741034.86
NOx         818907.04
NH3         450953.19
CO           61773.49
SO2         373137.58
O3          879841.90
Benzene      78438.33
Toluene     186983.89
Xylene       35067.00
dtype: float64
```

In [ ]:

```
plt.figure(figsize=(12,6))
plt.title('Distribustion of Elements which are impacting air quality')
plt.pie(element_df, labels=element_df.index, autopct='%1.1f%%', startangle=180);
```



Which year is top when compared to others?

In [ ]:

```
Yrs = ['Year', 'AQI']

yr_df = analysis_df[Yrs].copy()

z = yr_df.groupby('Year')['AQI'].mean()

z
```

Out[ ]:

```
Year
2015    212.463054
2016    197.150019
2017    181.472789
2018    182.684312
2019    156.518173
2020    113.520697
Name: AQI, dtype: float64
```

In [ ]:

```
Years = list(z.index)
Years
```

Out[ ]:

```
[2015, 2016, 2017, 2018, 2019, 2020]
```

In [ ]:

```
Sum_of_AQI = z.tolist()
Sum_of_AQI
```

Out[ ]:

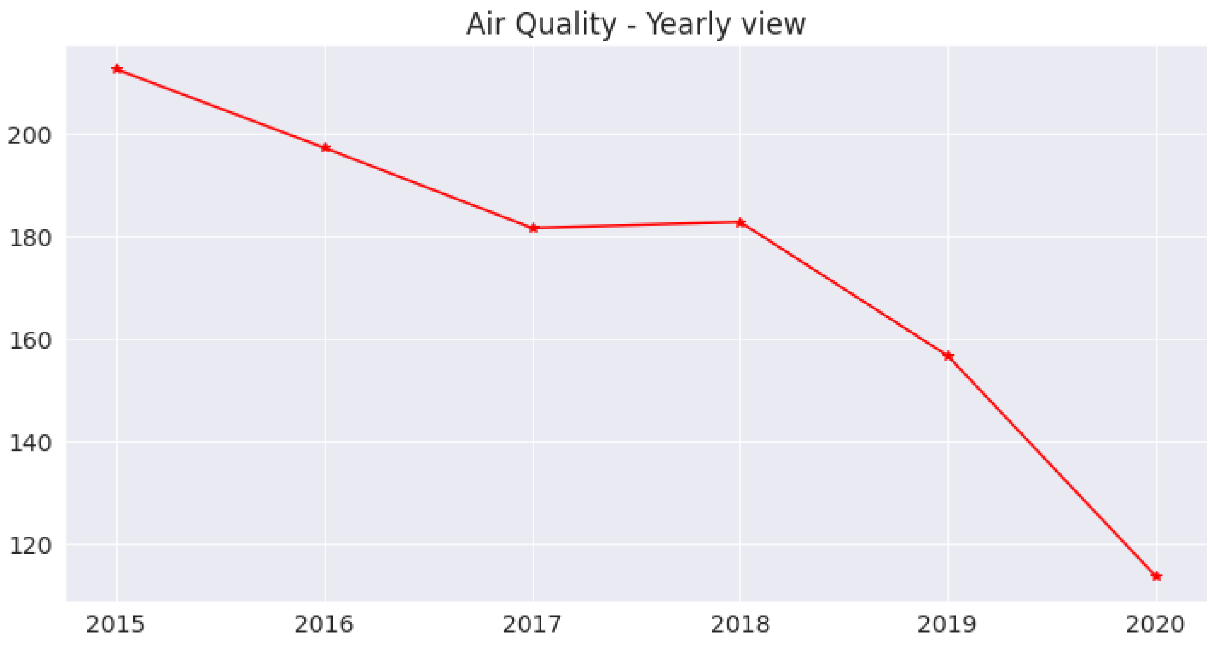
```
[212.4630541871921,
 197.150019432569,
 181.47278911564626,
 182.68431167016072,
 156.51817281855466,
 113.52069667496042]
```

In [ ]:

```
plt.figure(figsize=(12,6))
plt.title('Air Quality - Yearly view')
plt.plot(Years, Sum_of_AQI, color='red', marker='*')
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fa2e8c83bd0>]
```



Top 10 cities for the year 2020

In [ ]:

```
year2020 = analysis_df[analysis_df.Year == 2020]
year2020.head(10)
```

Out[ ]:

	City	Date	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene

	City	Date	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	
1826	Ahmedabad	2020-01-01	3.78	12.64	8.99	NaN	3.78	27.70	23.67	4.21	31.42	2.52	2
1827	Ahmedabad	2020-01-02	3.63	14.38	9.73	NaN	3.63	23.96	23.67	3.71	31.14	2.52	1
1828	Ahmedabad	2020-01-03	7.06	15.13	12.65	NaN	7.06	35.78	23.66	4.78	31.14	2.52	2
1829	Ahmedabad	2020-01-04	8.97	20.79	16.84	NaN	8.97	38.98	23.65	4.12	31.14	2.52	2
1830	Ahmedabad	2020-01-05	5.41	15.34	11.53	NaN	5.41	45.83	23.61	3.30	31.14	2.52	2
1831	Ahmedabad	2020-01-06	7.17	16.88	13.58	NaN	7.17	38.11	23.64	2.75	31.14	2.52	1
1832	Ahmedabad	2020-01-07	7.37	22.67	16.56	NaN	7.37	58.67	23.58	2.75	31.14	2.52	2
1833	Ahmedabad	2020-01-08	2.38	16.33	9.74	NaN	2.38	30.54	23.59	2.75	31.14	2.52	1
1834	Ahmedabad	2020-01-09	2.41	14.14	8.70	NaN	2.41	46.78	23.64	2.75	31.14	2.52	1
1835	Ahmedabad	2020-01-10	2.50	11.65	7.55	NaN	2.50	26.75	23.58	2.75	31.14	2.52	1

In [ ]:

```
ele = ['City', 'AQI']

year2020_df = year2020[ele].copy()
year2020_df.head(5)
```

Out[ ]:

	City	AQI
1826	Ahmedabad	216.0
1827	Ahmedabad	162.0
1828	Ahmedabad	220.0
1829	Ahmedabad	254.0
1830	Ahmedabad	255.0

In [ ]:

```
a = {
        'AQI' : 'sum'
}
b = year2020_df.groupby(['City'])
c = b.agg(a)
Cities_df = c.sort_values(by=['AQI'], ascending=False)
```

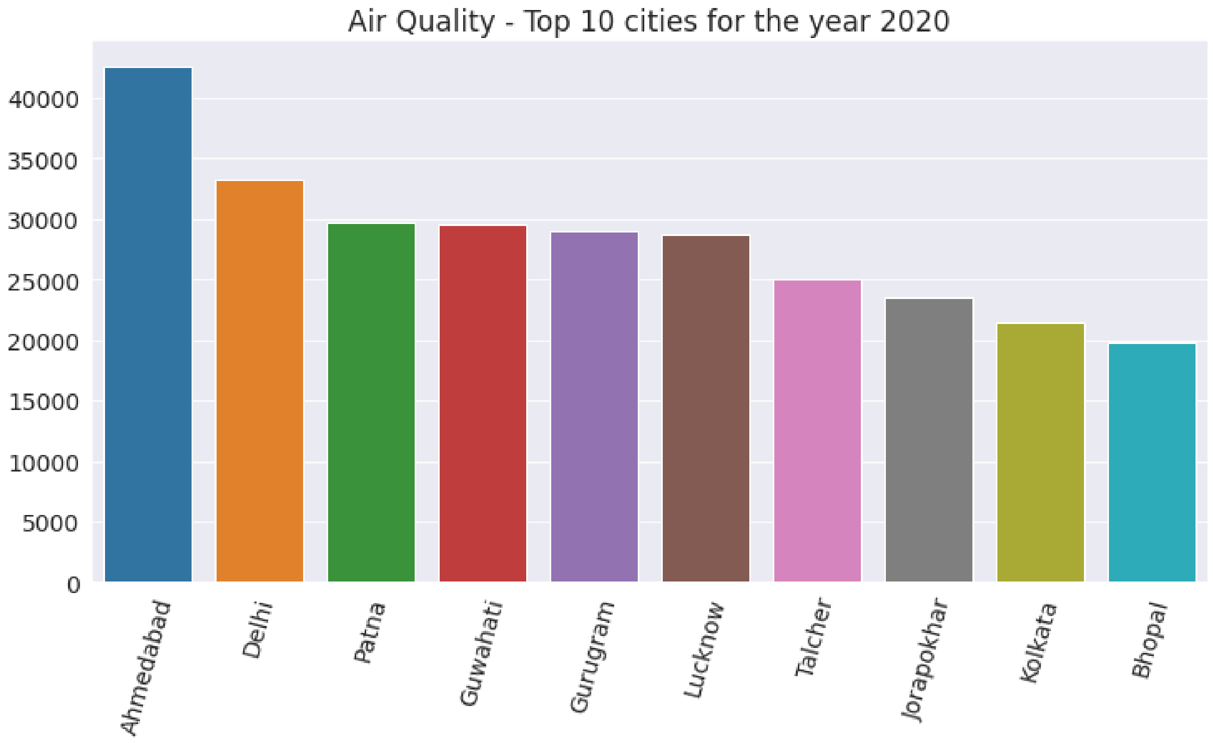
In [ ]:

```
Top_10_Cities = Cities_df.head(10)

AQI_list = Top_10_Cities["AQI"].tolist()
Cities = list(Top_10_Cities.index)
```

```
plt.figure(figsize=(12,6))
plt.xticks(rotation=75)
plt.title('Air Quality - Top 10 cities for the year 2020')
sns.barplot(Cities, AQI_list);
```

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



Which month is bad over the years?

In [ ]:

```
mnths = ['Year', 'month_alph', 'AQI']

mn_df = analysis_df[mnths].copy()

mn_df
```

Out[ ]:

	Year	month_alph	AQI
28	2015	Jan	209.0
29	2015	Jan	328.0
30	2015	Jan	514.0
31	2015	Feb	782.0
32	2015	Feb	914.0
...	...	...	...
29526	2020	Jun	41.0
29527	2020	Jun	70.0
29528	2020	Jun	68.0
29529	2020	Jun	54.0

	Year	month_alph	AQI
29530	2020	Jul	50.0

24850 rows × 3 columns

In [ ]:

```
df2 = mn_df.groupby(['Year', 'month_alph'])['AQI'].sum()
df2
```

Out[ ]:

```
Year  month_alph
2015  Apr            32486.0
      Aug            29077.0
      Dec            45995.0
      Feb            22198.0
      Jan            11662.0
                      ...
2020  Jan           119120.0
      Jul             1740.0
      Jun            56094.0
      Mar            81421.0
      May            68383.0
Name: AQI, Length: 67, dtype: float64
```

In [ ]:

```
Sum_of_AQI_list = list(df2)
df3 = list(df2.index)
Years_list = [i[0] for i in df3]
months_list = [i[1] for i in df3]

sub_df = pd.DataFrame(
    {'year': Years_list,
     'month': months_list,
     'sum_of_AQI': Sum_of_AQI_list
    })


sub_df.head(13)
```

Out[ ]:

	year	month	sum_of_AQI
0	2015	Apr	32486.0
1	2015	Aug	29077.0
2	2015	Dec	45995.0
3	2015	Feb	22198.0
4	2015	Jan	11662.0
5	2015	Jul	27531.0
6	2015	Jun	30933.0
7	2015	Mar	25940.0
8	2015	May	35738.0
9	2015	Nov	47709.0
10	2015	Oct	49684.0
11	2015	Sep	29217.0

	year	month	sum_of_AQI
12	2016	Apr	37410.0

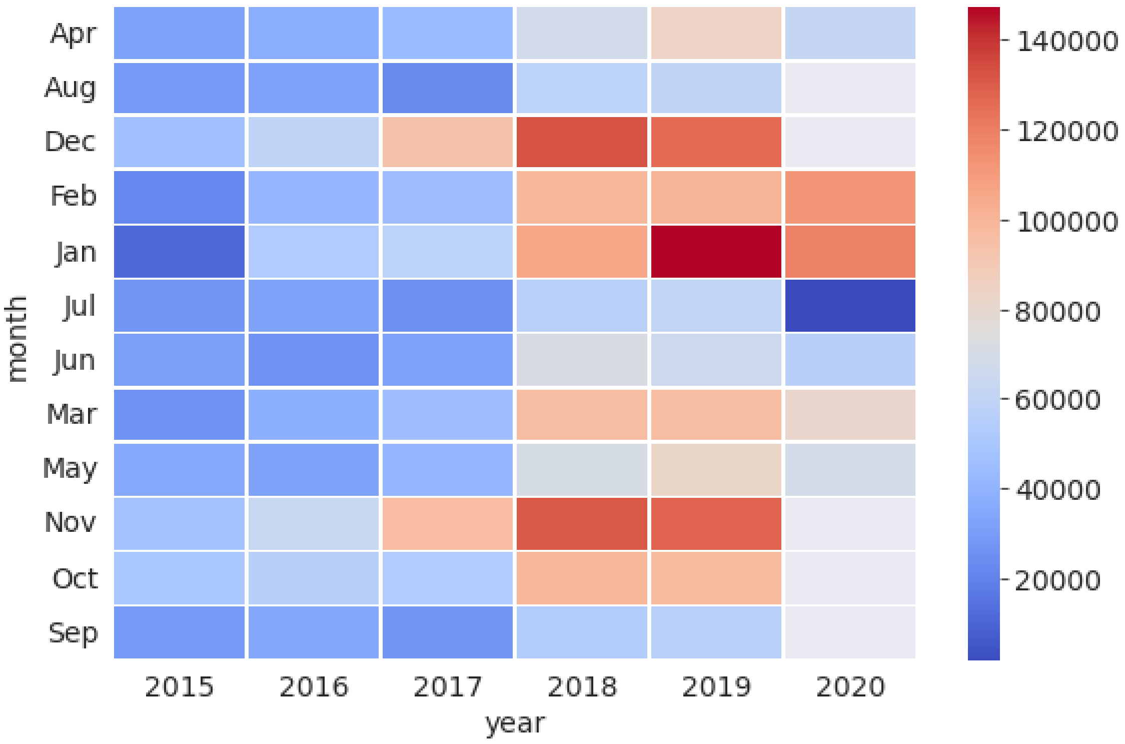
In [ ]:

```
sum_AQI = sub_df.pivot("month", "year", "sum_of_AQI")

f, ax = plt.subplots(figsize=(9, 6))
sns.heatmap(sum_AQI, annot=False, linewidths=.5, ax=ax,cmap='coolwarm')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa2e8c15b10>
```

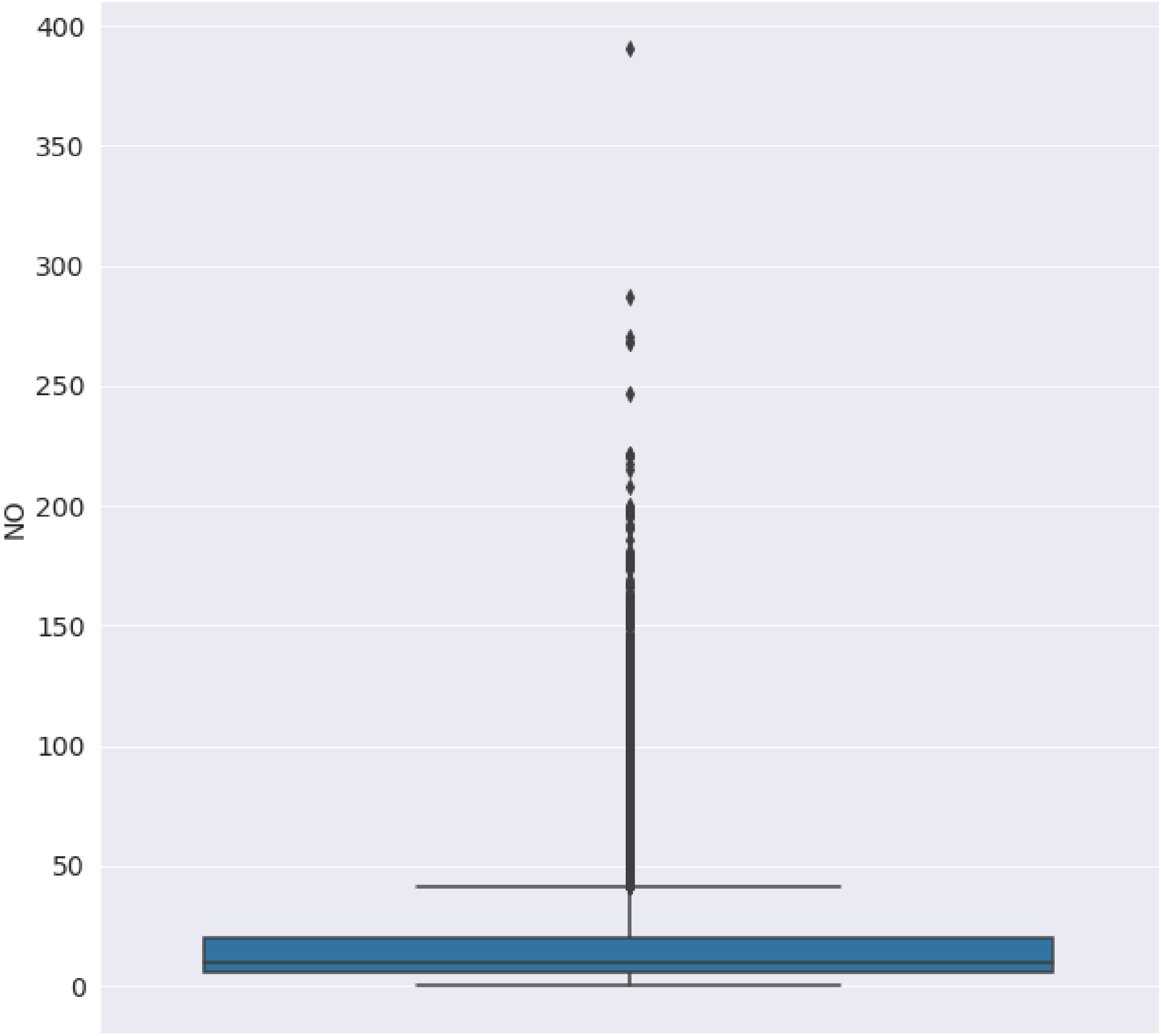


In [ ]:

```
analysis_df.columns
plt.figure(figsize = (10,10))
sns.boxplot(data = analysis_df,y= 'NO')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2c1b3d5c90>
```



In [ ]:

```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

In [ ]:

```
%cd /content/drive/MyDrive/Colab Notebooks
```

```
/content/drive/MyDrive/Colab Notebooks
```

In [ ]:

```
!sudo apt-get install textlive-xetex textlive-fonts-recommend


!jupyter nbconvert --to pdf Airquality.ipynb
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package textlive-xetex
E: Unable to locate package textlive-fonts-recommend
[NbConvertApp] WARNING | pattern 'Airquality.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and include
the error message in the cell output (the default behaviour is to abort conversion).
This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with defaul
t basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_for
mat=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_for
mat=notebook --FilesWriter.build_directory= --ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --TemplateExporter.
exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporte
r.exclude_input=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITI
CAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
```

```
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf',
'python', 'rst', 'script', 'slides']
            or a dotted object name that represents the import path for an
            `Exporter` class
    Default: 'html'
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template file to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_file]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    overwrite base name use for output files.
                can only be used when converting one notebook at a time.
    Default: ''
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                  to output to the directory of each notebook. To re
cover
                                  previous default behaviour (outputting to the curr
ent
                                  working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-sl
ideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb
            which will convert mynotebook.ipynb to the default format (probably HTM
L).

            You can specify the export format with `--to`.
            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'not
ebook', 'pdf', 'python', 'rst', 'script', 'slides'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX includes
            'base', 'article' and 'report'.  HTML includes 'basic' and 'full'. You
            can specify the flavor of the format used.

            > jupyter nbconvert --to html --template basic mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```

In [ ]: