INM433 Visual Analytics

Visual Analytics coursework

City University of London

Name: RAJANI MOHAN JANIPALLI

Title: Visual Analysis of Air Pollution in India

This analysis is to study the spatio-temporal patterns of Air pollution in India between the year 2015 and 2021 and also try to find if there is any relation between Air pollution and power generation for the years 2017 tot 2020.

Datasets used:

https://kaggle.com/rohanrao/air-quality-data-in-india

https://kaggle.com/twinkle0705/state-wise-power-consumption-in-india

https://kaggle.com/navinmundhra/daily-power-generation-in-india-20172020

Initial import of necessary libraries.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import altair as alt
from sklearn.preprocessing import MinMaxScaler
import statsmodels.api as sm
```

Load primary data set into a pandas data frame.

In [2]:
```
station_day = pd.read_csv('station_day.csv')
```

Have a glance of first 5 rows of the data frame.

In [3]:
```
station_day.head()
```

Out[3]:

	StationId	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	AP001	2017-11-24	71.36	115.75	1.75	20.65	12.40	12.19	0.10	10.76	109.26	0.17	5.92	0.10	NaN	NaN
1	AP001	2017-11-25	81.40	124.50	1.44	20.50	12.08	10.72	0.12	15.24	127.09	0.20	6.50	0.06	184.0	Moderate
2	AP001	2017-11-26	78.32	129.06	1.26	26.00	14.85	10.28	0.14	26.96	117.44	0.22	7.95	0.08	197.0	Moderate
3	AP001	2017-11-27	88.76	135.32	6.60	30.85	21.77	12.91	0.11	33.59	111.81	0.29	7.63	0.12	198.0	Moderate
4	AP001	2017-11-28	64.18	104.09	2.56	28.07	17.01	11.42	0.09	19.00	138.18	0.17	5.02	0.07	188.0	Moderate

Check dimensions of the data frame.

In [4]:
```
station_day.shape
```

Out[4]:
```
(108035, 16)
```

Check the data types and presence of missing values in all the columns of the data frame.

In [5]:
```
station_day.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 108035 entries, 0 to 108034
Data columns (total 16 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   StationId   108035 non-null  object 
 1   Date        108035 non-null  object 
 2   PM2.5       86410 non-null   float64
 3   PM10        65329 non-null   float64
 4   NO          90929 non-null   float64
 5   NO2         91488 non-null   float64
 6   NOx         92535 non-null   float64
 7   NH3         59930 non-null   float64
 8   CO          95037 non-null   float64
 9   SO2         82831 non-null   float64
 10  O3          82467 non-null   float64
 11  Benzene     76580 non-null   float64
 12  Toluene     69333 non-null   float64
 13  Xylene      22898 non-null   float64
 14  AQI         87025 non-null   float64
 15  AQI_Bucket  87025 non-null   object 
```

```
dtypes: float64(13), object(3)
memory usage: 13.2+ MB
```

Change the data type of date column to datetime.

In [6]:
```
station_day.Date = pd.to_datetime(station_day.Date)
```

Confirm the change of data type.

In [7]:
```
station_day.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 108035 entries, 0 to 108034
Data columns (total 16 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   StationId   108035 non-null  object
 1   Date        108035 non-null  datetime64[ns]
 2   PM2.5       86410 non-null   float64
 3   PM10        65329 non-null   float64
 4   NO          90929 non-null   float64
 5   NO2         91488 non-null   float64
 6   NOx         92535 non-null   float64
 7   NH3         59930 non-null   float64
 8   CO          95037 non-null   float64
 9   SO2         82831 non-null   float64
 10  O3          82467 non-null   float64
 11  Benzene     76580 non-null   float64
 12  Toluene     69333 non-null   float64
 13  Xylene      22898 non-null   float64
 14  AQI         87025 non-null   float64
 15  AQI_Bucket  87025 non-null   object
dtypes: datetime64[ns](1), float64(13), object(2)
memory usage: 13.2+ MB
```

In [8]:
```
station_day.head()
```

Out[8]:

	StationId	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	AP001	2017-11-24	71.36	115.75	1.75	20.65	12.40	12.19	0.10	10.76	109.26	0.17	5.92	0.10	NaN	NaN
1	AP001	2017-11-25	81.40	124.50	1.44	20.50	12.08	10.72	0.12	15.24	127.09	0.20	6.50	0.06	184.0	Moderate
2	AP001	2017-11-26	78.32	129.06	1.26	26.00	14.85	10.28	0.14	26.96	117.44	0.22	7.95	0.08	197.0	Moderate
3	AP001	2017-11-27	88.76	135.32	6.60	30.85	21.77	12.91	0.11	33.59	111.81	0.29	7.63	0.12	198.0	Moderate
4	AP001	2017-11-28	64.18	104.09	2.56	28.07	17.01	11.42	0.09	19.00	138.18	0.17	5.02	0.07	188.0	Moderate

Check the percentage of missing values in each column of the data frame.

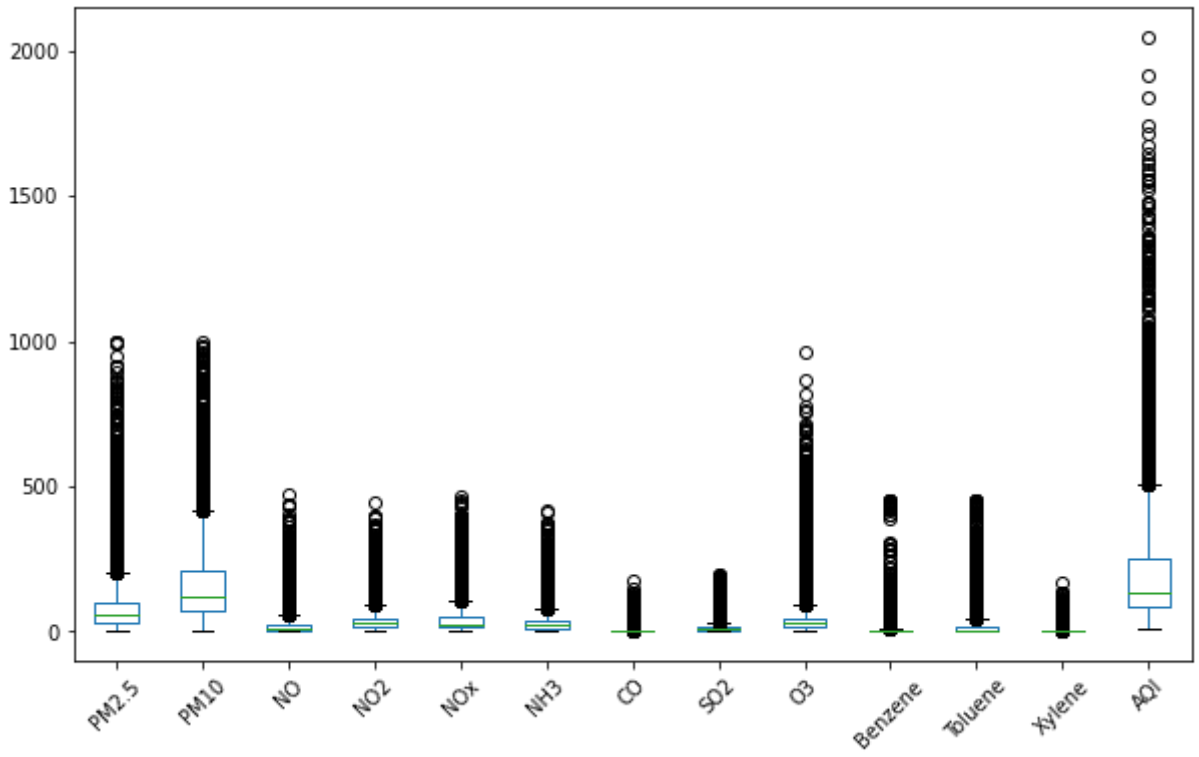
In [9]:
```
((station_day.isnull().sum())/(station_day.shape[0]))*100
```

Out[9]:
```
StationId      0.000000
Date           0.000000
PM2.5         20.016661
PM10          39.529782
NO            15.833758
NO2           15.316333
NOx           14.347202
NH3           44.527237
CO            12.031286
SO2           23.329477
O3            23.666404
Benzene       29.115564
Toluene       35.823576
Xylene        78.805017
AQI           19.447401
AQI_Bucket    19.447401
dtype: float64
```

Produce boxplot of each column in the data frame to visualize missing values in them.

In [10]:
```
station_day.boxplot(grid=False,rot=45,figsize=(10,6))
```

Out[10]:
```
<AxesSubplot:>
```



Check the minimum and maximum values of dates to understand the time range of the dataset.

In [11]:
```
station_day.Date.min()
```

Out[11]:
```
Timestamp('2015-01-01 00:00:00')
```

In [12]:
```
station_day.Date.max()
```

Out[12]:
```
Timestamp('2020-07-01 00:00:00')
```

Create an additional column of year from data, to make plotting and analysis easy.

In [13]:
```
station_day['Year'] = station_day.Date.dt.year
```

In [14]:
```
station_day.head()
```

Out[14]:

	StationId	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket	Year
0	AP001	2017-11-24	71.36	115.75	1.75	20.65	12.40	12.19	0.10	10.76	109.26	0.17	5.92	0.10	NaN	NaN	2017
1	AP001	2017-11-25	81.40	124.50	1.44	20.50	12.08	10.72	0.12	15.24	127.09	0.20	6.50	0.06	184.0	Moderate	2017
2	AP001	2017-11-26	78.32	129.06	1.26	26.00	14.85	10.28	0.14	26.96	117.44	0.22	7.95	0.08	197.0	Moderate	2017
3	AP001	2017-11-27	88.76	135.32	6.60	30.85	21.77	12.91	0.11	33.59	111.81	0.29	7.63	0.12	198.0	Moderate	2017
4	AP001	2017-11-28	64.18	104.09	2.56	28.07	17.01	11.42	0.09	19.00	138.18	0.17	5.02	0.07	188.0	Moderate	2017

Check the descriptive statistics of all the columns of the data frame.

In [15]:
```
station_day.describe()
```

Out[15]:

	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	
count	86410.000000	65329.000000	90929.000000	91488.000000	92535.000000	59930.000000	95037.000000	82831.000000	82467.000000	76580.000000	6933
mean	80.272571	157.968427	23.123424	35.240760	41.195055	28.732875	1.605749	12.257634	38.134836	3.358029	1
std	76.526403	123.418672	34.491019	29.510827	45.145976	24.897797	4.369578	12.984723	39.128004	11.156234	2
min	0.020000	0.010000	0.010000	0.010000	0.000000	0.010000	0.000000	0.010000	0.010000	0.000000	
25%	31.880000	70.150000	4.840000	15.090000	13.970000	11.900000	0.530000	5.040000	18.895000	0.160000	
50%	55.950000	122.090000	10.290000	27.210000	26.660000	23.590000	0.910000	8.950000	30.840000	1.210000	
75%	99.920000	208.670000	24.980000	46.930000	50.500000	38.137500	1.450000	14.920000	47.140000	3.610000	1
max	1000.000000	1000.000000	470.000000	448.050000	467.630000	418.900000	175.810000	195.650000	963.000000	455.030000	45

NOTE: *Before the start of data analysis, spatial data has to be added by merging data frames of other datasets. So, missing values are not handle at this stage.*

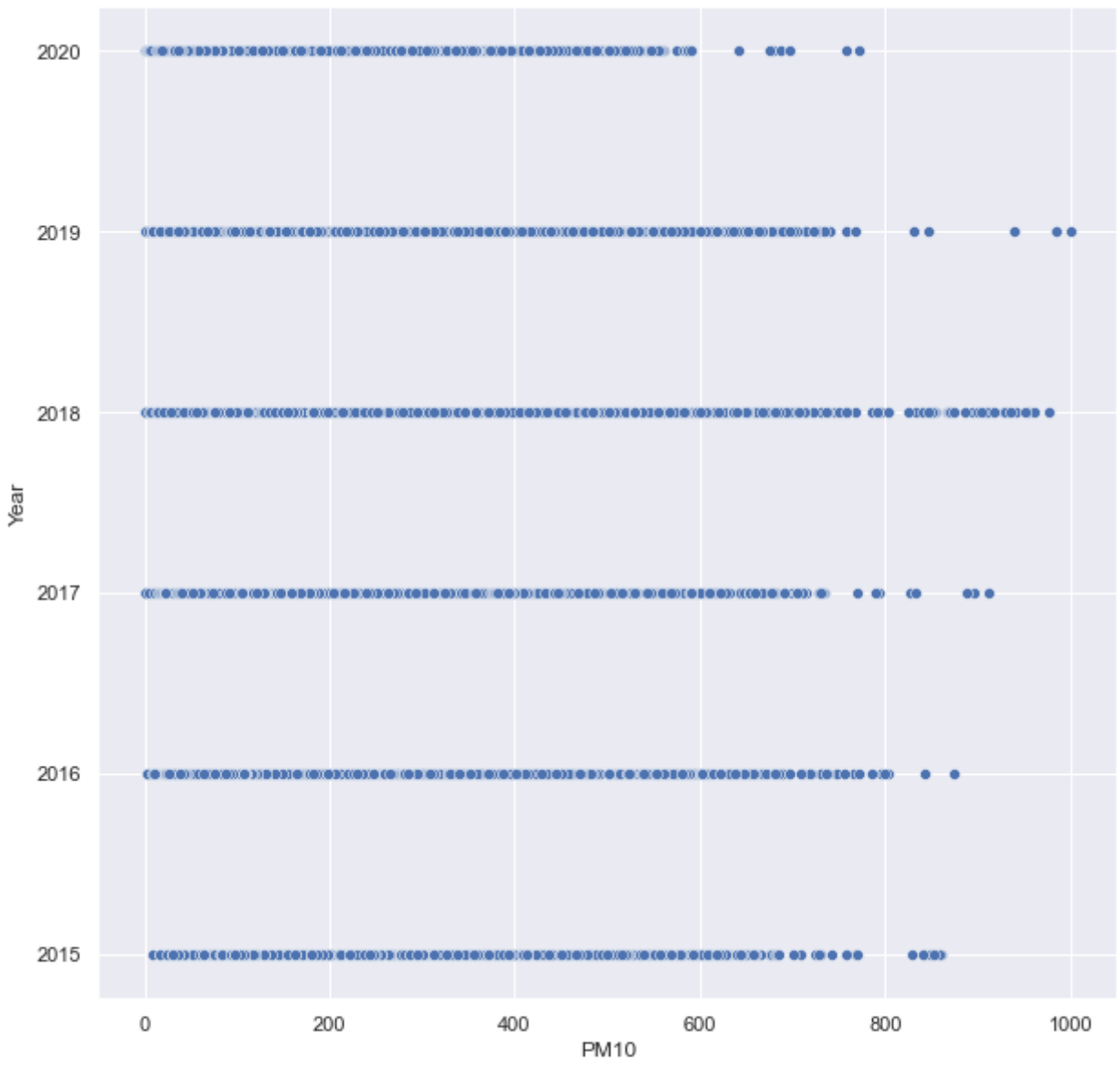
Make scatterplots of all the pollutant columns to see their yearly variation and also to visiually identify the presence of outliers.

In [16]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(data=station_day, x=station_day['PM2.5'], y=station_day.Year)
```

Out[16]: <AxesSubplot:xlabel='PM2.5', ylabel='Year'>

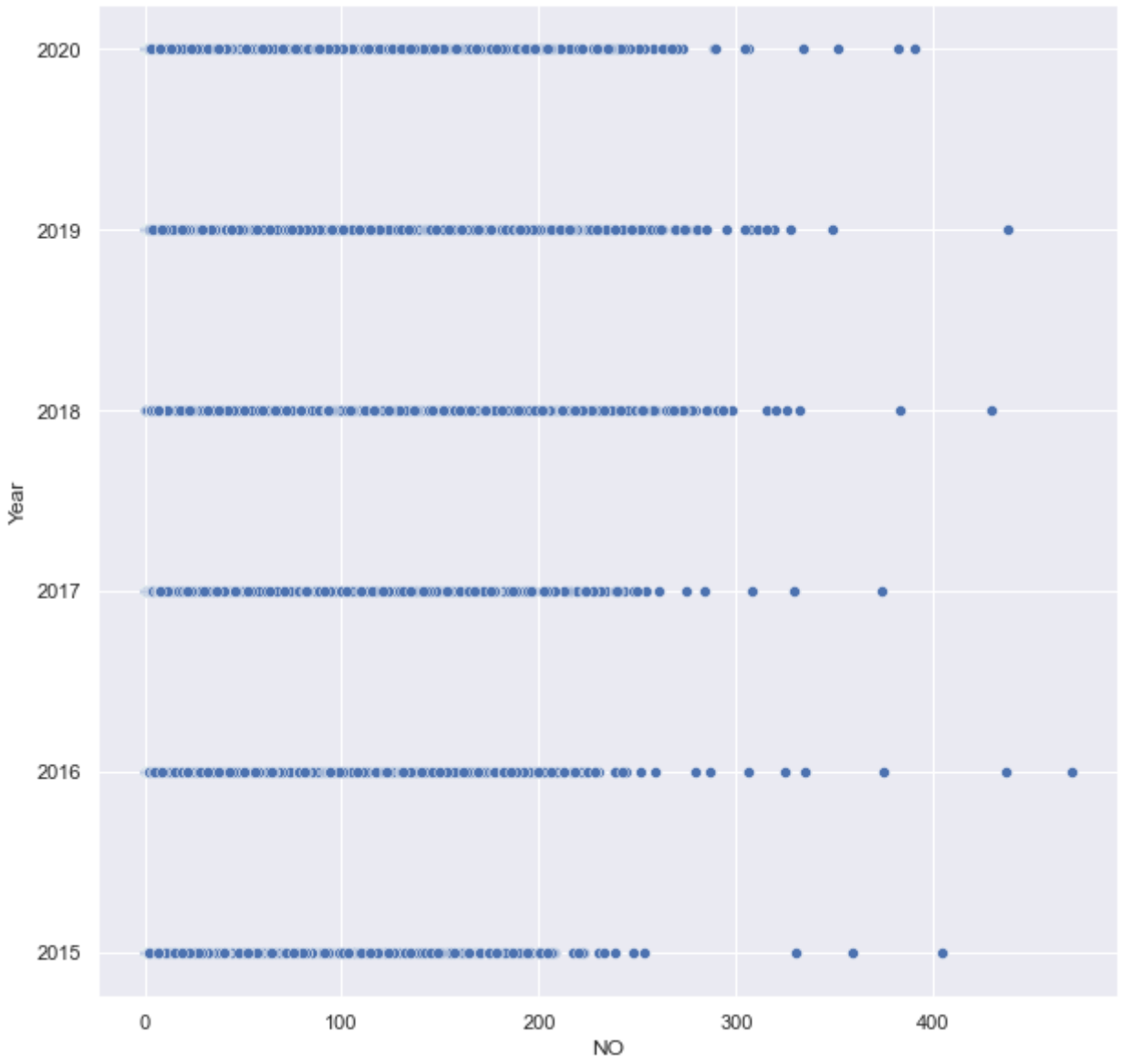
In [17]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.PM10,y=station_day.Year)
```

Out[17]: <AxesSubplot:xlabel='PM10', ylabel='Year'>



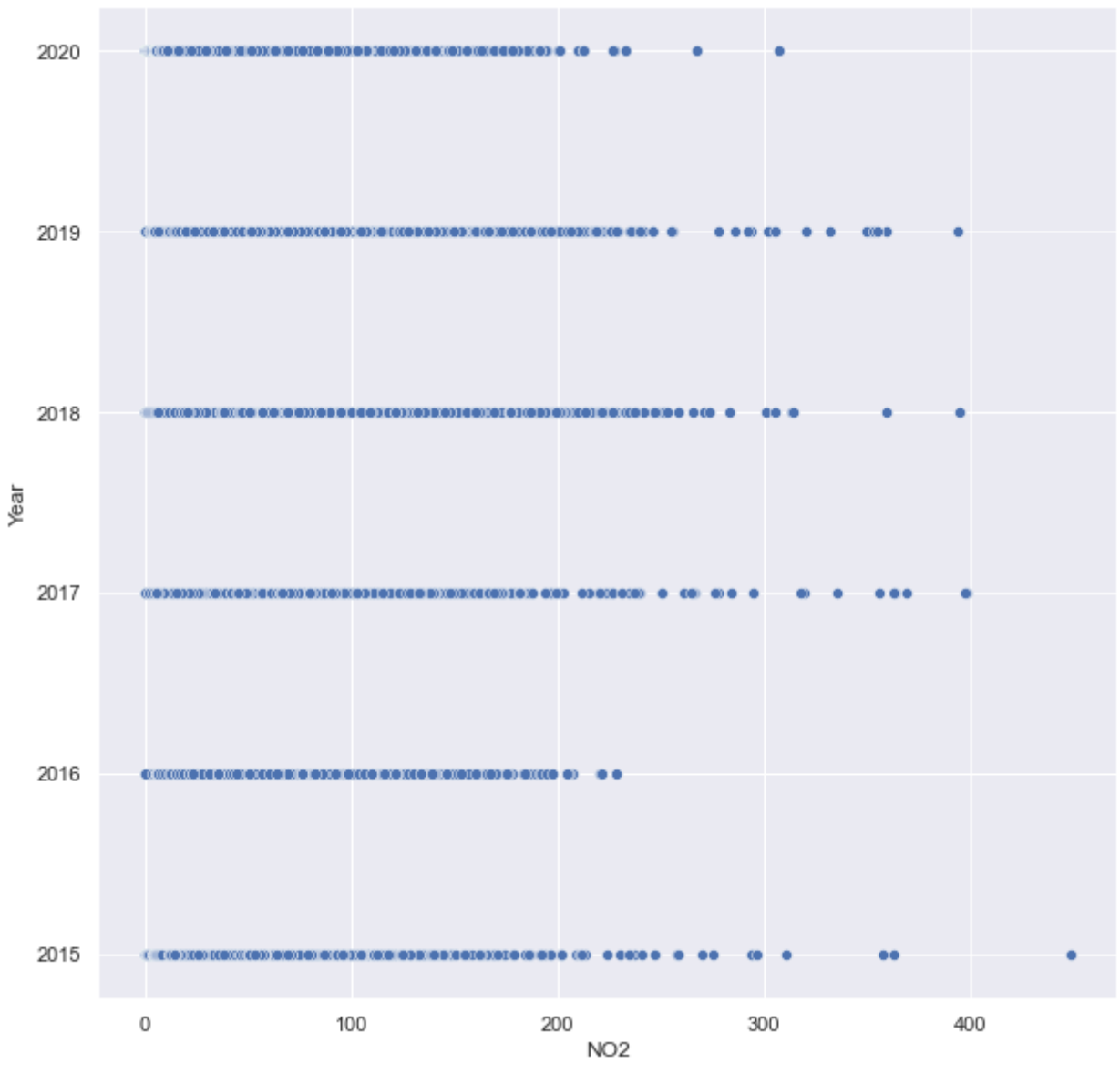
In [18]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.NO,y=station_day.Year)
```

Out[18]: <AxesSubplot:xlabel='NO', ylabel='Year'>



In [19]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.NO2,y=station_day.Year)
```

Out[19]: <AxesSubplot:xlabel='NO2', ylabel='Year'>

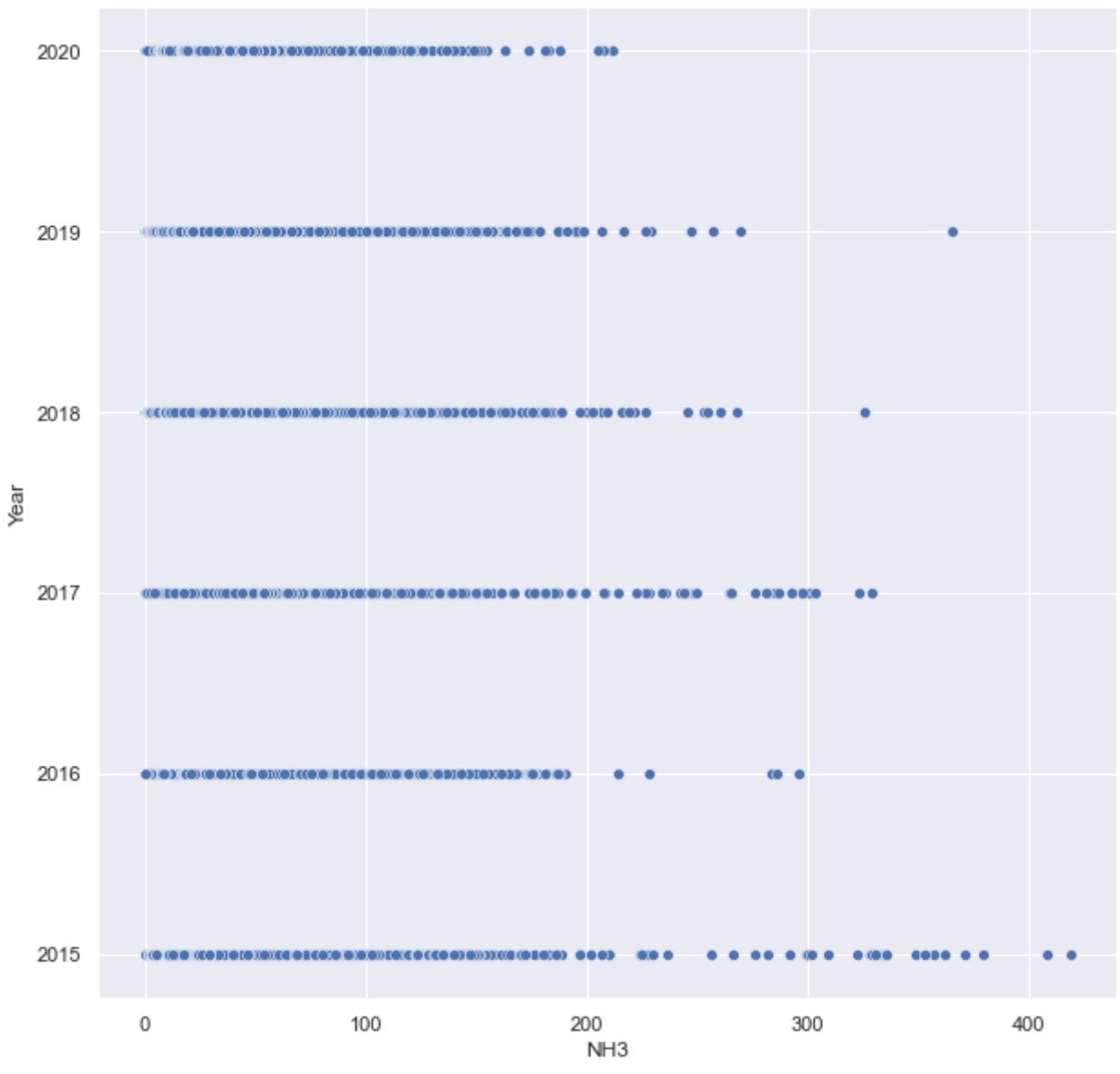


In [20]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.NOx,y=station_day.Year)
```

Out[20]: <AxesSubplot:xlabel='NOx', ylabel='Year'>

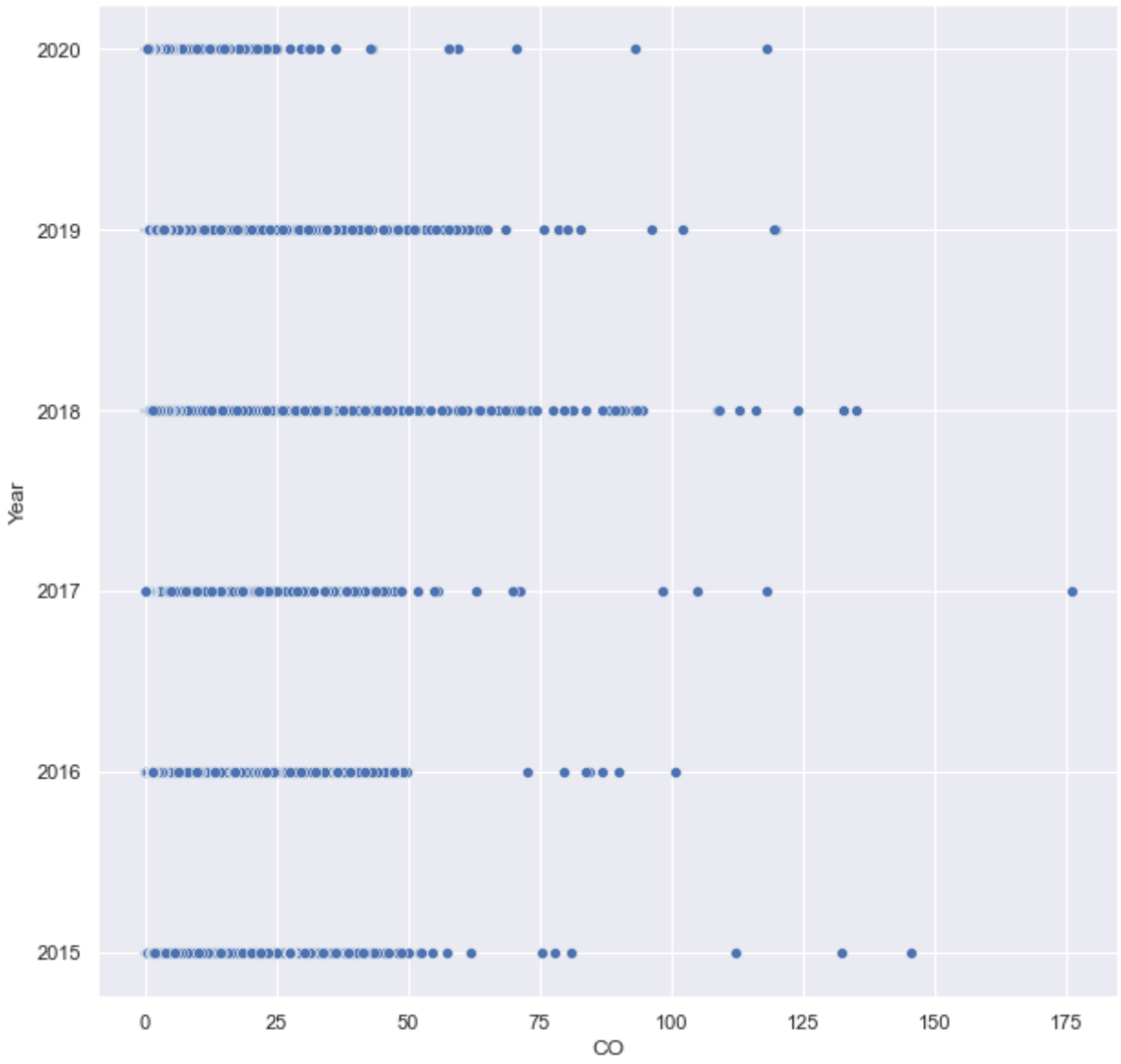
In [21]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.NH3,y=station_day.Year)
```

Out[21]: <AxesSubplot:xlabel='NH3', ylabel='Year'>



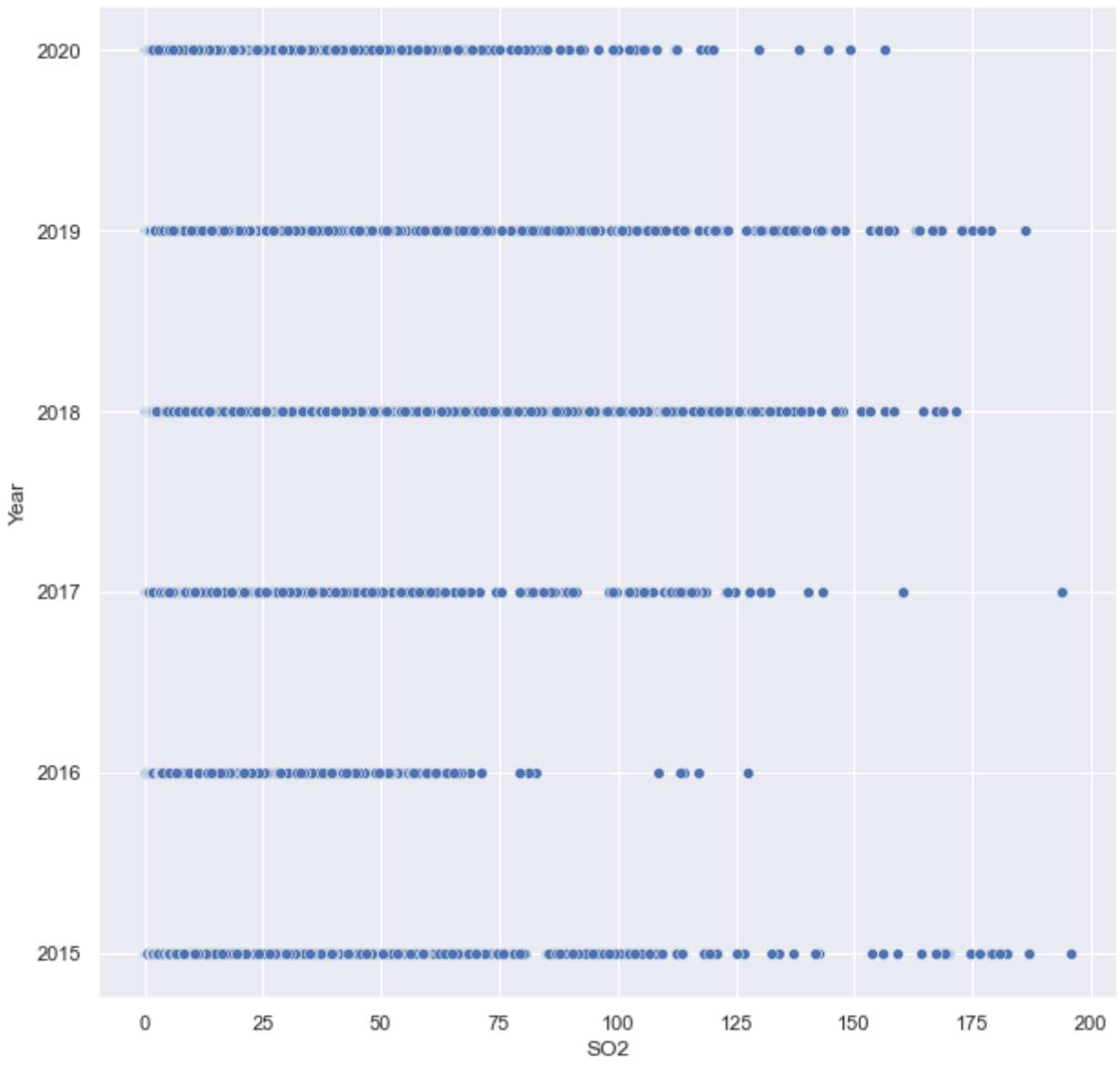
In [22]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.CO,y=station_day.Year)
```

Out[22]: <AxesSubplot:xlabel='CO', ylabel='Year'>



In [23]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.SO2,y=station_day.Year)
```

Out[23]: <AxesSubplot:xlabel='SO2', ylabel='Year'>

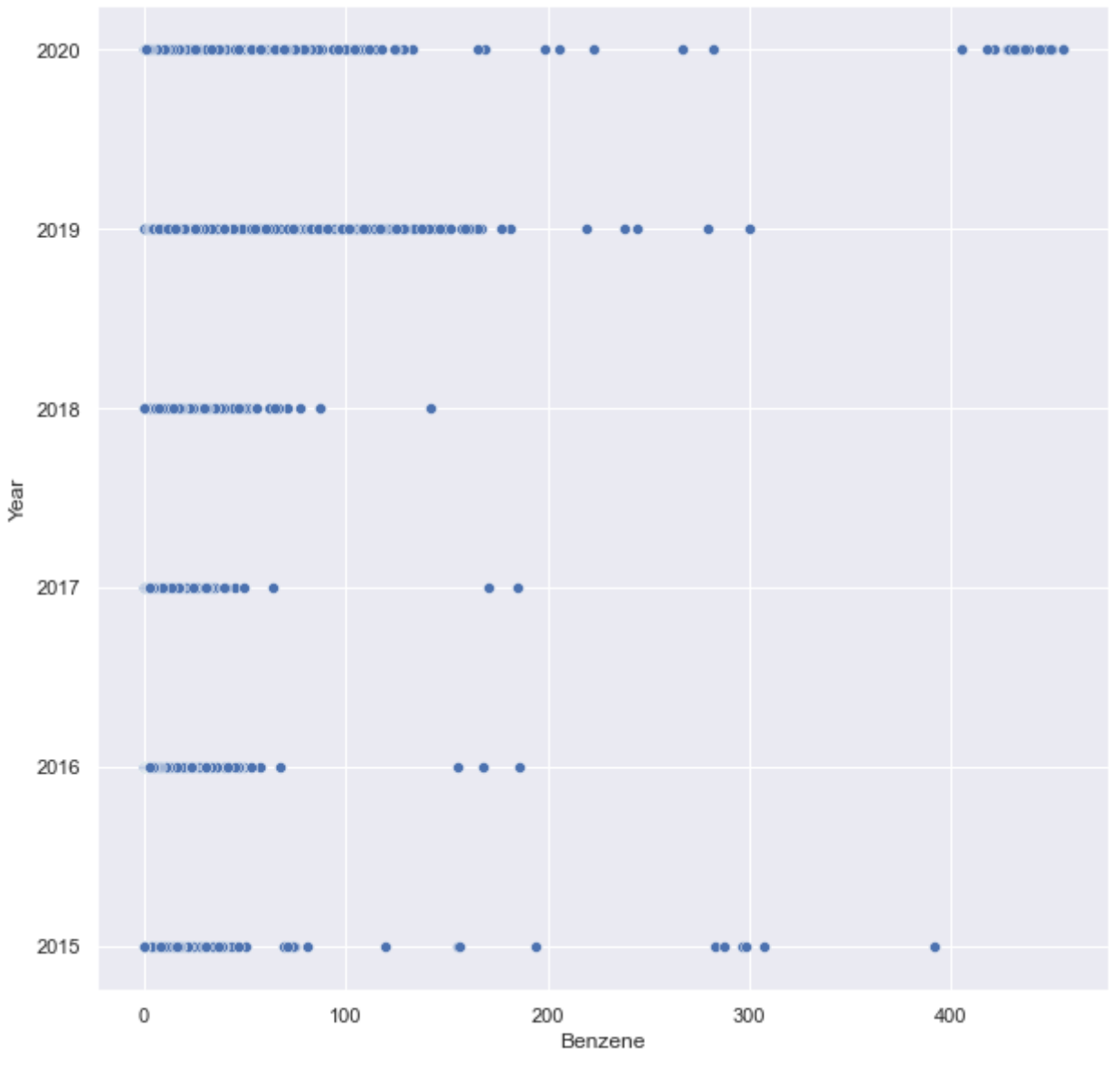


In [24]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.O3,y=station_day.Year)
```

Out[24]: <AxesSubplot:xlabel='O3', ylabel='Year'>

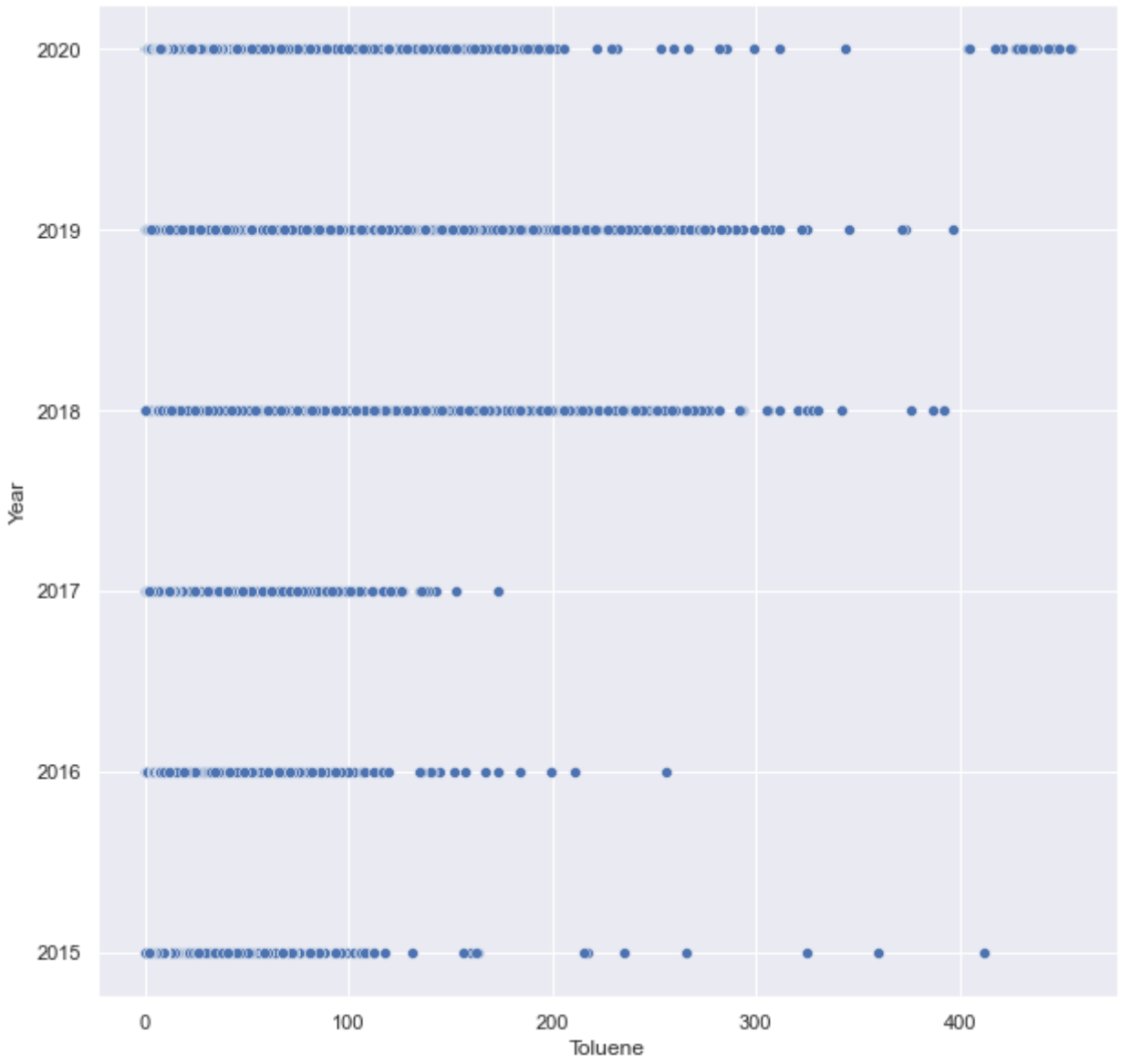
In [25]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.Benzene,y=station_day.Year)
```

Out[25]:
```
<AxesSubplot:xlabel='Benzene', ylabel='Year'>
```



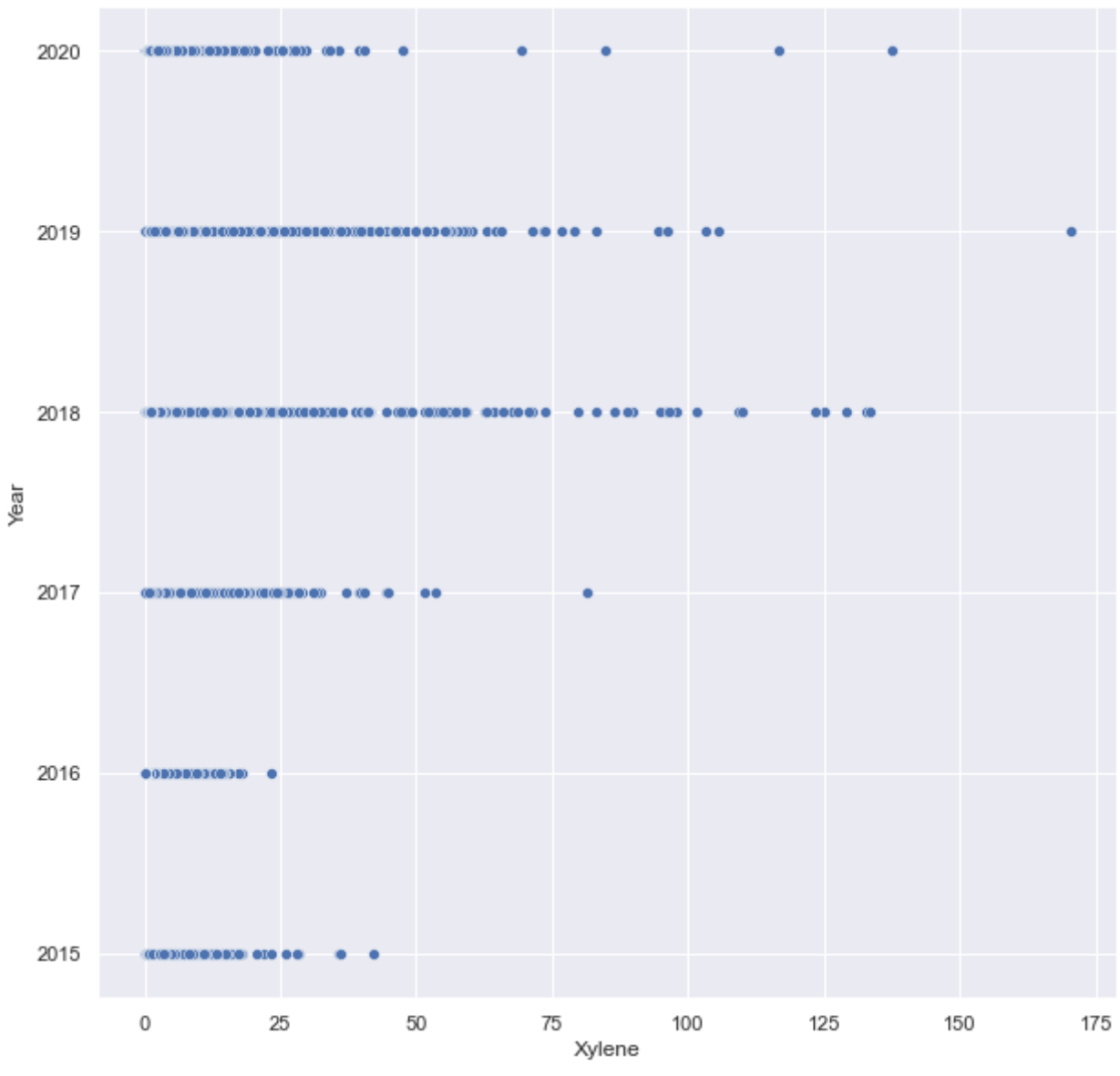
In [26]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.Toluene,y=station_day.Year)
```

Out[26]:
```
<AxesSubplot:xlabel='Toluene', ylabel='Year'>
```



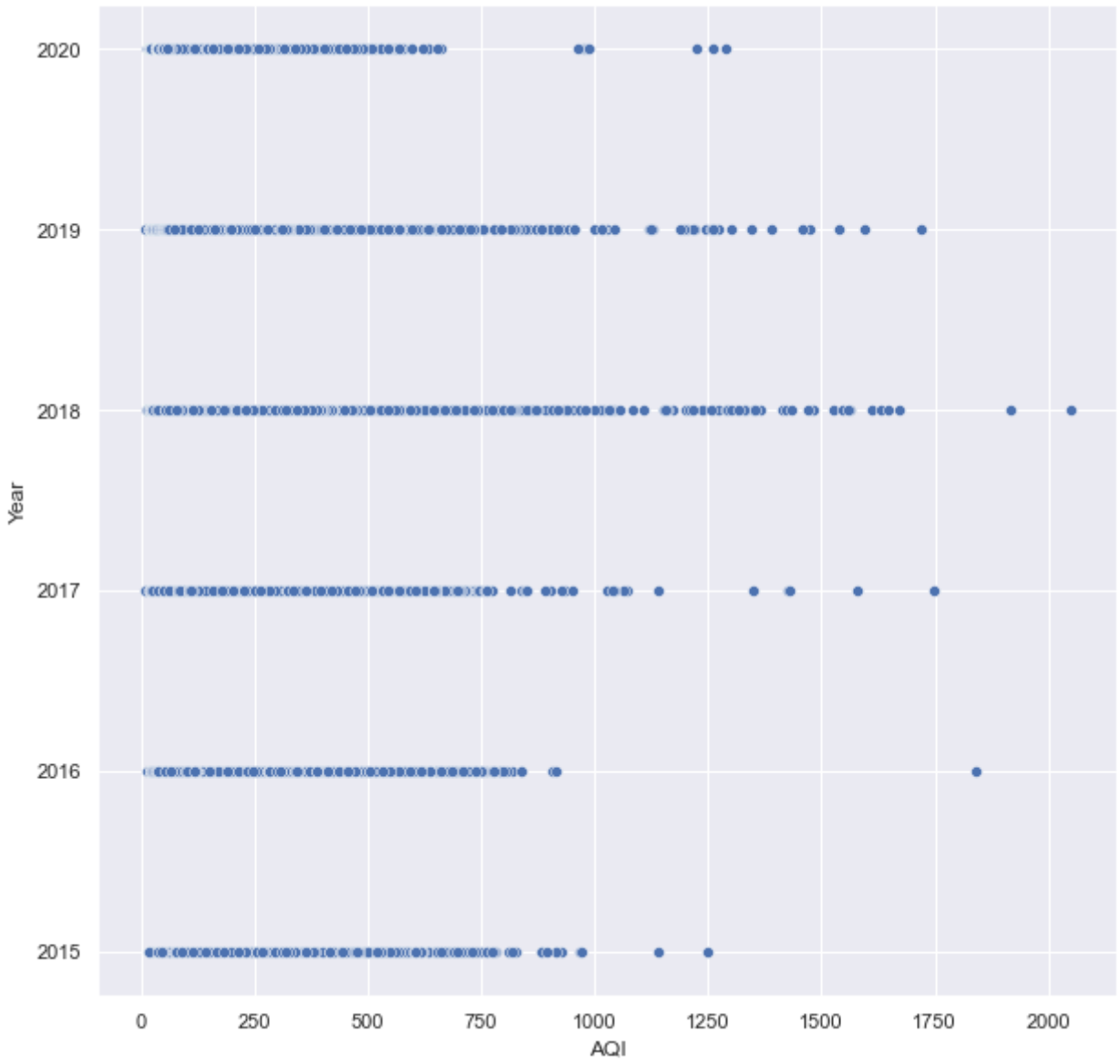
In [27]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.Xylene,y=station_day.Year)
```

Out[27]: <AxesSubplot:xlabel='Xylene', ylabel='Year'>



In [28]:
```
sns.set_theme()
plt.figure(figsize=(10,10))
sns.scatterplot(x=station_day.AQI,y=station_day.Year)
```

Out[28]: <AxesSubplot:xlabel='AQI', ylabel='Year'>



Since most of the apparent outliers are practically possible values and considering that these are values measured by sensors, it is decided not to remove these apparent outliers.

Check the mean AQI value.

In [29]:
```
station_day.AQI.mean()
```

Out[29]:
```
179.7492904337834
```

Check the median AQI value.

In [30]:
```
station_day.AQI.median()
```

Out[30]:
```
132.0
```

Check the AQI category of median AQI value.

In [31]:
```
station_day[station_day.AQI == 132.0].AQI_Bucket.value_counts()
```

Out[31]:
```
Moderate    344
Name: AQI_Bucket, dtype: int64
```

Load stations dataset into a pandas data frame.

In [32]:
```
stations = pd.read_csv('stations.csv')
```

In [33]:
```
stations.head()
```

Out[33]:

	StationId	StationName	City	State	Status
0	AP001	Secretariat, Amaravati - APPCB	Amaravati	Andhra Pradesh	Active
1	AP002	Anand Kala Kshetram, Rajamahendravaram - APPCB	Rajamahendravaram	Andhra Pradesh	NaN
2	AP003	Tirumala, Tirupati - APPCB	Tirupati	Andhra Pradesh	NaN
3	AP004	PWD Grounds, Vijayawada - APPCB	Vijayawada	Andhra Pradesh	NaN
4	AP005	GVM Corporation, Visakhapatnam - APPCB	Visakhapatnam	Andhra Pradesh	Active

Check the columns information of primary dataset data frame and stations data frame.

In [34]:
```
station_day.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 108035 entries, 0 to 108034
Data columns (total 17 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
```

```
 0   StationId   108035 non-null  object
 1   Date        108035 non-null  datetime64[ns]
 2   PM2.5       86410 non-null   float64
 3   PM10        65329 non-null   float64
 4   NO          90929 non-null   float64
 5   NO2         91488 non-null   float64
 6   NOx         92535 non-null   float64
 7   NH3         59930 non-null   float64
 8   CO          95037 non-null   float64
 9   SO2         82831 non-null   float64
 10  O3          82467 non-null   float64
 11  Benzene     76580 non-null   float64
 12  Toluene     69333 non-null   float64
 13  Xylene      22898 non-null   float64
 14  AQI         87025 non-null   float64
 15  AQI_Bucket  87025 non-null   object
 16  Year        108035 non-null  int64
dtypes: datetime64[ns](1), float64(13), int64(1), object(2)
memory usage: 14.0+ MB
```

In [35]:
```
stations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 230 entries, 0 to 229
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   StationId    230 non-null    object
 1   StationName  230 non-null    object
 2   City         230 non-null    object
 3   State        230 non-null    object
 4   Status       133 non-null    object
dtypes: object(5)
memory usage: 9.1+ KB
```

Merge both data frames on common column.

In [36]:
```
station_day = pd.merge(station_day,stations,how='left',left_on='StationId',right_on='StationId')
```

In [37]:
```
station_day.head()
```

Out[37]:

	StationId	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	...	Benzene	Toluene	Xylene	AQI	AQI_Bucket	Year	StationName	Cit
0	AP001	2017-11-24	71.36	115.75	1.75	20.65	12.40	12.19	0.10	10.76	...	0.17	5.92	0.10	NaN	NaN	2017	Secretariat, Amaravati - APPCB	Amarava
1	AP001	2017-11-25	81.40	124.50	1.44	20.50	12.08	10.72	0.12	15.24	...	0.20	6.50	0.06	184.0	Moderate	2017	Secretariat, Amaravati - APPCB	Amarava
2	AP001	2017-11-26	78.32	129.06	1.26	26.00	14.85	10.28	0.14	26.96	...	0.22	7.95	0.08	197.0	Moderate	2017	Secretariat, Amaravati - APPCB	Amarava
3	AP001	2017-11-27	88.76	135.32	6.60	30.85	21.77	12.91	0.11	33.59	...	0.29	7.63	0.12	198.0	Moderate	2017	Secretariat, Amaravati - APPCB	Amarava
4	AP001	2017-11-28	64.18	104.09	2.56	28.07	17.01	11.42	0.09	19.00	...	0.17	5.02	0.07	188.0	Moderate	2017	Secretariat, Amaravati - APPCB	Amarava

5 rows × 21 columns

Drop unnecessary columns.

In [38]:
```
station_day.drop(['Status','City','StationName'],axis=1,inplace=True)
```

In [39]:
```
station_day.head()
```

Out[39]:

	StationId	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket	Year	State
0	AP001	2017-11-24	71.36	115.75	1.75	20.65	12.40	12.19	0.10	10.76	109.26	0.17	5.92	0.10	NaN	NaN	2017	Andhra Pradesh
1	AP001	2017-11-25	81.40	124.50	1.44	20.50	12.08	10.72	0.12	15.24	127.09	0.20	6.50	0.06	184.0	Moderate	2017	Andhra Pradesh
2	AP001	2017-11-26	78.32	129.06	1.26	26.00	14.85	10.28	0.14	26.96	117.44	0.22	7.95	0.08	197.0	Moderate	2017	Andhra Pradesh
3	AP001	2017-11-27	88.76	135.32	6.60	30.85	21.77	12.91	0.11	33.59	111.81	0.29	7.63	0.12	198.0	Moderate	2017	Andhra Pradesh
4	AP001	2017-11-28	64.18	104.09	2.56	28.07	17.01	11.42	0.09	19.00	138.18	0.17	5.02	0.07	188.0	Moderate	2017	Andhra Pradesh

Check the current dimensions of the data frame.

In [40]:
```
station_day.shape
```

Out[40]:
```
(108035, 18)
```

Since 'StationId' column is no more need, it is to be dropped.

In [41]:
```
station_day.drop('StationId',axis=1,inplace=True)
```

In [42]:
```
station_day.head()
```

Out[42]:

	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket	Year	State
0	2017-11-24	71.36	115.75	1.75	20.65	12.40	12.19	0.10	10.76	109.26	0.17	5.92	0.10	NaN	NaN	2017	Andhra Pradesh
1	2017-11-25	81.40	124.50	1.44	20.50	12.08	10.72	0.12	15.24	127.09	0.20	6.50	0.06	184.0	Moderate	2017	Andhra Pradesh
2	2017-11-26	78.32	129.06	1.26	26.00	14.85	10.28	0.14	26.96	117.44	0.22	7.95	0.08	197.0	Moderate	2017	Andhra Pradesh
3	2017-11-27	88.76	135.32	6.60	30.85	21.77	12.91	0.11	33.59	111.81	0.29	7.63	0.12	198.0	Moderate	2017	Andhra Pradesh
4	2017-11-28	64.18	104.09	2.56	28.07	17.01	11.42	0.09	19.00	138.18	0.17	5.02	0.07	188.0	Moderate	2017	Andhra Pradesh

Load longitude and lattitude data into a pandas data frame.

In [43]:
```
long_data = pd.read_csv('long_data_.csv')
```

In [44]:
```
long_data.head()
```

Out[44]:

	States	Regions	latitude	longitude	Dates	Usage
0	Punjab	NR	31.519974	75.980003	02/01/2019 00:00:00	119.9
1	Haryana	NR	28.450006	77.019991	02/01/2019 00:00:00	130.3
2	Rajasthan	NR	26.449999	74.639981	02/01/2019 00:00:00	234.1
3	Delhi	NR	28.669993	77.230004	02/01/2019 00:00:00	85.8
4	UP	NR	27.599981	78.050006	02/01/2019 00:00:00	313.9

Drop unnecessary columns.

In [45]:
```
long_data.drop(['Usage', 'Dates','Regions'],axis=1,inplace=True)
```

In [46]:
```
long_data.head()
```

Out[46]:

	States	latitude	longitude
0	Punjab	31.519974	75.980003
1	Haryana	28.450006	77.019991
2	Rajasthan	26.449999	74.639981
3	Delhi	28.669993	77.230004
4	UP	27.599981	78.050006

Check the values of state columns in both data frames to make any necessary changes.

In [47]:
```
station_day.State.unique()
```

Out[47]:
```
array(['Andhra Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Delhi',
       'Gujarat', 'Haryana', 'Jharkhand', 'Karnataka', 'Kerala',
       'Maharashtra', 'Meghalaya', 'Madhya Pradesh', 'Mizoram', 'Odisha',
       'Punjab', 'Rajasthan', 'Telangana', 'Tamil Nadu', 'Uttar Pradesh',
       'West Bengal'], dtype=object)
```

In [48]:
```
long_data.sort_values('States', inplace=True)
```

In [49]:
```
station_day.State.unique()
```

Out[49]:
```
array(['Andhra Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Delhi',
       'Gujarat', 'Haryana', 'Jharkhand', 'Karnataka', 'Kerala',
       'Maharashtra', 'Meghalaya', 'Madhya Pradesh', 'Mizoram', 'Odisha',
       'Punjab', 'Rajasthan', 'Telangana', 'Tamil Nadu', 'Uttar Pradesh',
       'West Bengal'], dtype=object)
```

In [50]:
```
long_data.States.unique()
```

Out[50]:
```
array(['Andhra Pradesh', 'Arunachal Pradesh', 'Assam', 'Bihar',
       'Chandigarh', 'Chhattisgarh', 'DNH', 'Delhi', 'Goa', 'Gujarat',
       'HP', 'Haryana', 'J&K', 'Jharkhand', 'Karnataka', 'Kerala', 'MP',
       'Maharashtra', 'Manipur', 'Meghalaya', 'Mizoram', 'Nagaland',
       'Odisha', 'Pondy', 'Punjab', 'Rajasthan', 'Sikkim', 'Tamil Nadu',
       'Telangana', 'Tripura', 'UP', 'Uttarakhand', 'West Bengal'],
      dtype=object)
```

Replace values in state column to make them same is that in primary data set.

In [51]:
```
long_data.States.replace({'MP':'Madhya Pradesh', 'UP':'Uttar Pradesh'},inplace=True)
```

Confirm that the changes have occured.

In [52]:
```
long_data.States.unique()
```

Out[52]:
```
array(['Andhra Pradesh', 'Arunachal Pradesh', 'Assam', 'Bihar',
       'Chandigarh', 'Chhattisgarh', 'DNH', 'Delhi', 'Goa', 'Gujarat',
       'HP', 'Haryana', 'J&K', 'Jharkhand', 'Karnataka', 'Kerala',
       'Madhya Pradesh', 'Maharashtra', 'Manipur', 'Meghalaya', 'Mizoram',
       'Nagaland', 'Odisha', 'Pondy', 'Punjab', 'Rajasthan', 'Sikkim',
       'Tamil Nadu', 'Telangana', 'Tripura', 'Uttar Pradesh',
       'Uttarakhand', 'West Bengal'], dtype=object)
```

In [53]:
```
long_data.head()
```

Out[53]:

	States	latitude	longitude
1401	Andhra Pradesh	14.750429	78.570026
13776	Andhra Pradesh	14.750429	78.570026
8727	Andhra Pradesh	14.750429	78.570026
4338	Andhra Pradesh	14.750429	78.570026
1698	Andhra Pradesh	14.750429	78.570026

In [54]:
```
long_data.shape
```

Out[54]:
```
(16599, 3)
```

Check the total number of duplicate entries in the states column of the data frame.

In [55]:
```
long_data.States.duplicated().sum()
```

Out[55]:
```
16566
```

Check the total number of duplicate entries in the data frame as whole.

In [56]:
```
long_data.duplicated().sum()
```

Out[56]:
```
16566
```

Drop duplicate rows in the states column.

In [57]:
```
long_data.States.drop_duplicates(inplace=True)
```

In [58]:
```
long_data.shape
```

Out[58]:
```
(16599, 3)
```

In [59]:
```
long_data.duplicated().sum()
```

Out[59]:
```
16566
```

In [60]:
```
long_data.States.duplicated().sum()
```

Out[60]:
```
16566
```

Drop duplicate rows in the data frame as a whole.

In [61]:
```
long_data.drop_duplicates(inplace=True)
```

In [62]:
```
long_data.shape
```

Out[62]:
```
(33, 3)
```

Check if all the values in states columns of primary dataset and long-lat dataset match.

In [63]:
```
long_data.States.unique()
```

Out[63]:
```
array(['Andhra Pradesh', 'Arunachal Pradesh', 'Assam', 'Bihar',
       'Chandigarh', 'Chhattisgarh', 'DNH', 'Delhi', 'Goa', 'Gujarat',
       'HP', 'Haryana', 'J&K', 'Jharkhand', 'Karnataka', 'Kerala',
       'Madhya Pradesh', 'Maharashtra', 'Manipur', 'Meghalaya', 'Mizoram',
       'Nagaland', 'Odisha', 'Pondy', 'Punjab', 'Rajasthan', 'Sikkim',
       'Tamil Nadu', 'Telangana', 'Tripura', 'Uttar Pradesh',
       'Uttarakhand', 'West Bengal'], dtype=object)
```

In [64]:
```
station_day.State.unique()
```

Out[64]:
```
array(['Andhra Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Delhi',
       'Gujarat', 'Haryana', 'Jharkhand', 'Karnataka', 'Kerala',
       'Maharashtra', 'Meghalaya', 'Madhya Pradesh', 'Mizoram', 'Odisha',
       'Punjab', 'Rajasthan', 'Telangana', 'Tamil Nadu', 'Uttar Pradesh',
       'West Bengal'], dtype=object)
```

Drop rows from long-lat dataset for states column values not present in primary dataset.

In [65]:
```
remove_state_list = ['Arunachal Pradesh', 'Chhattisgarh', 'DNH', 'Goa', 'HP', 'J&K', 'Manipur',
                     'Nagaland', 'Pondy', 'Sikkim', 'Tripura', 'Uttarakhand']

for i in long_data.States:
    if i in remove_state_list:
        long_data.drop(index=long_data[long_data.States == i].index, inplace=True)
```

In [66]:
```
long_data.shape
```

Out[66]:
```
(21, 3)
```

Check if all the values in states columns of primary dataset and long-lat dataset match.

In [67]:
```
long_data.States.unique()
```

Out[67]:
```
array(['Andhra Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Delhi',
       'Gujarat', 'Haryana', 'Jharkhand', 'Karnataka', 'Kerala',
       'Madhya Pradesh', 'Maharashtra', 'Meghalaya', 'Mizoram', 'Odisha',
       'Punjab', 'Rajasthan', 'Tamil Nadu', 'Telangana', 'Uttar Pradesh',
       'West Bengal'], dtype=object)
```

In [68]:
```
station_day.State.unique()
```

Out[68]:
```
array(['Andhra Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Delhi',
       'Gujarat', 'Haryana', 'Jharkhand', 'Karnataka', 'Kerala',
       'Maharashtra', 'Meghalaya', 'Madhya Pradesh', 'Mizoram', 'Odisha',
       'Punjab', 'Rajasthan', 'Telangana', 'Tamil Nadu', 'Uttar Pradesh',
       'West Bengal'], dtype=object)
```

Check the number of unique values in states columns of both data frames matches.

In [69]:
```
station_day.State.unique().size
```

Out[69]:
```
21
```

In [70]:
```
station_day.head()
```

Out[70]:

	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket	Year	State
0	2017-11-24	71.36	115.75	1.75	20.65	12.40	12.19	0.10	10.76	109.26	0.17	5.92	0.10	NaN	NaN	2017	Andhra Pradesh
1	2017-11-25	81.40	124.50	1.44	20.50	12.08	10.72	0.12	15.24	127.09	0.20	6.50	0.06	184.0	Moderate	2017	Andhra Pradesh
2	2017-11-26	78.32	129.06	1.26	26.00	14.85	10.28	0.14	26.96	117.44	0.22	7.95	0.08	197.0	Moderate	2017	Andhra Pradesh
3	2017-11-27	88.76	135.32	6.60	30.85	21.77	12.91	0.11	33.59	111.81	0.29	7.63	0.12	198.0	Moderate	2017	Andhra Pradesh
4	2017-11-28	64.18	104.09	2.56	28.07	17.01	11.42	0.09	19.00	138.18	0.17	5.02	0.07	188.0	Moderate	2017	Andhra Pradesh

In [71]:
```
station_day.shape
```

Out[71]:
```
(108035, 17)
```

Rename the state column name in long-lat dataset to match with that in primary dataset, to avoid creation of duplicate columns.

In [72]:
```
long_data.rename(columns={'States':'State'}, inplace=True)
```

In [73]:
```
long_data.head()
```

Out[73]:

	State	latitude	longitude
1401	Andhra Pradesh	14.750429	78.570026
4746	Assam	26.749981	94.216667
5763	Bihar	25.785414	87.479973
14462	Chandigarh	30.719997	76.780006
4590	Delhi	28.669993	77.230004

Merge long-lat data frame with the data frame of primary data set.

In [74]:
```
station_day = pd.merge(station_day,long_data,how='left',left_on='State',right_on='State')
```

In [75]:
```
station_day.head()
```

Out[75]:

	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket	Year	State	latitude	longitude
0	2017-11-24	71.36	115.75	1.75	20.65	12.40	12.19	0.10	10.76	109.26	0.17	5.92	0.10	NaN	NaN	2017	Andhra Pradesh	14.750429	78.57002
1	2017-11-25	81.40	124.50	1.44	20.50	12.08	10.72	0.12	15.24	127.09	0.20	6.50	0.06	184.0	Moderate	2017	Andhra Pradesh	14.750429	78.57002
2	2017-11-26	78.32	129.06	1.26	26.00	14.85	10.28	0.14	26.96	117.44	0.22	7.95	0.08	197.0	Moderate	2017	Andhra Pradesh	14.750429	78.57002
3	2017-11-27	88.76	135.32	6.60	30.85	21.77	12.91	0.11	33.59	111.81	0.29	7.63	0.12	198.0	Moderate	2017	Andhra Pradesh	14.750429	78.57002
4	2017-11-28	64.18	104.09	2.56	28.07	17.01	11.42	0.09	19.00	138.18	0.17	5.02	0.07	188.0	Moderate	2017	Andhra Pradesh	14.750429	78.57002

In [76]:
```
station_day.shape
```

Out[76]:
```
(108035, 19)
```

In [77]:
```
station_day.columns
```

Out[77]:
```
Index(['Date', 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3',
       'Benzene', 'Toluene', 'Xylene', 'AQI', 'AQI_Bucket', 'Year', 'State',
       'latitude', 'longitude'],
      dtype='object')
```

Make a true copy of the data frame of the primary dataset, on which further data modification will be done.

In [78]:
```
station_day_1 = station_day.copy()
```

Verify that all the columns are present in the copy data frame.

In [79]:
```
station_day_1.columns
```

Out[79]:
```
Index(['Date', 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3',
       'Benzene', 'Toluene', 'Xylene', 'AQI', 'AQI_Bucket', 'Year', 'State',
       'latitude', 'longitude'],
      dtype='object')
```

To avoid any skewing of data, missing values are filled with median values of the columns rather than mean values.

In [80]:
```
s_d_1_col_fill_list = ['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3',
       'Benzene', 'Toluene', 'Xylene', 'AQI']

for col in s_d_1_col_fill_list:
    station_day_1[col].fillna(station_day_1[col].median(), inplace=True)
```

In [375…
```
# AQI_null = pd.DataFrame(station_day[station_day.AQI.isnull()])
```

In [376…
```
# AQI_null.AQI_Bucket.unique()
```

Out[376…
```
array([nan], dtype=object)
```

In [81]:
```
station_day_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 108035 entries, 0 to 108034
Data columns (total 19 columns):
 #   Column      Non-Null Count   Dtype         
---  ------      --------------   -----         
 0   Date        108035 non-null  datetime64[ns]
```

```
 1   PM2.5       108035 non-null  float64
 2   PM10        108035 non-null  float64
 3   NO          108035 non-null  float64
 4   NO2         108035 non-null  float64
 5   NOx         108035 non-null  float64
 6   NH3         108035 non-null  float64
 7   CO          108035 non-null  float64
 8   SO2         108035 non-null  float64
 9   O3          108035 non-null  float64
 10  Benzene     108035 non-null  float64
 11  Toluene     108035 non-null  float64
 12  Xylene      108035 non-null  float64
 13  AQI         108035 non-null  float64
 14  AQI_Bucket  87025 non-null   object
 15  Year        108035 non-null  int64
 16  State       108035 non-null  object
 17  latitude    108035 non-null  float64
 18  longitude   108035 non-null  float64
dtypes: datetime64[ns](1), float64(15), int64(1), object(2)
memory usage: 16.5+ MB
```

As it was observed above in the notebook, for median value of AQI, corresponding value of AQI_Bucket is 'Moderate'. Since missing values in AQI were filled with median values, missing values in AQI_Bucket are to be filled with 'Moderate'.

In [82]:
```
station_day_1.AQI_Bucket.fillna('Moderate', inplace=True)
```

In [83]:
```
station_day_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 108035 entries, 0 to 108034
Data columns (total 19 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   Date        108035 non-null  datetime64[ns]
 1   PM2.5       108035 non-null  float64
 2   PM10        108035 non-null  float64
 3   NO          108035 non-null  float64
 4   NO2         108035 non-null  float64
 5   NOx         108035 non-null  float64
 6   NH3         108035 non-null  float64
 7   CO          108035 non-null  float64
 8   SO2         108035 non-null  float64
 9   O3          108035 non-null  float64
 10  Benzene     108035 non-null  float64
 11  Toluene     108035 non-null  float64
 12  Xylene      108035 non-null  float64
 13  AQI         108035 non-null  float64
 14  AQI_Bucket  108035 non-null  object
 15  Year        108035 non-null  int64
 16  State       108035 non-null  object
 17  latitude    108035 non-null  float64
 18  longitude   108035 non-null  float64
dtypes: datetime64[ns](1), float64(15), int64(1), object(2)
memory usage: 16.5+ MB
```

In [402…
```
#sns.lineplot(x=station_day_1.AQI, y=station_day_1.Year)
```

In [403…
```
#station_day_1.to_csv('station_day_1.csv')
```

Since, the number of rows is more than 5000, Altair requires the following command line to avoid Max rows errors.

In [84]:
```
alt.data_transformers.disable_max_rows()
```

Out[84]:
```
DataTransformerRegistry.enable('default')
```

In [85]:
```
AQI_chart = alt.Chart(station_day_1).mark_line().encode(
    x='yearmonth(Date):T',
    y='mean(AQI):Q'
)


PM10_chart = alt.Chart(station_day_1).mark_line().encode(
    x='yearmonth(Date):T',
    y='mean(PM10):Q'
)

alt.hconcat(AQI_chart,PM10_chart).resolve_scale(x='shared')
```

Out[85]: Error loading script: Load timeout for modules: vega-embed http://requirejs.org/docs/errors.html#timeout

Considering the fact that the dataset is not small and the flexibility of Tableau platform to handle visualizations for bigger size datasets, it is decided to do all the further visulizations only in Tableau, except for regression plots and other modelling related plots.

Create another copy of the data frame before applying further transformations on the data frame.

In [86]:

```
station_day_2 = station_day_1.copy()
```

In [87]:
```
station_day_2.describe()
```

Out[87]:

	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Ber
count	108035.000000	108035.000000	108035.000000	108035.000000	108035.000000	108035.000000	108035.000000	108035.000000	108035.000000	108035.00
mean	75.404005	143.785763	21.091411	34.010742	39.109682	26.442895	1.522042	11.485980	36.408410	2.73
std	69.128573	97.563192	31.987720	27.310521	42.091568	18.719146	4.104542	11.455365	34.326055	9.44
min	0.020000	0.010000	0.010000	0.010000	0.000000	0.010000	0.000000	0.010000	0.010000	0.00
25%	37.430000	102.620000	5.770000	17.100000	15.940000	21.260000	0.590000	6.160000	22.580000	0.44
50%	55.950000	122.090000	10.290000	27.210000	26.660000	23.590000	0.910000	8.950000	30.840000	1.21
75%	83.795000	144.955000	20.740000	42.240000	45.040000	26.000000	1.350000	12.570000	41.020000	2.35
max	1000.000000	1000.000000	470.000000	448.050000	467.630000	418.900000	175.810000	195.650000	963.000000	455.03

In [88]:
```
scale = MinMaxScaler()
```

In [89]:
```
s_d_2_col_fill_list = ['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3',
       'Benzene', 'Toluene', 'Xylene', 'AQI']

for colm in s_d_2_col_fill_list:
    station_day_2[colm] = scale.fit_transform(station_day_2[[colm]])
```

In [ ]:
```

```

In [90]:
```
station_day_2.describe()
```

Out[90]:

	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Ber
count	108035.000000	108035.000000	108035.000000	108035.000000	108035.000000	108035.000000	108035.000000	108035.000000	108035.000000	108035.00
mean	0.075386	0.143777	0.044855	0.075888	0.083634	0.063102	0.008657	0.058659	0.037797	0.00
std	0.069130	0.097564	0.068060	0.060956	0.090010	0.044687	0.023346	0.058553	0.035645	0.02
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.037411	0.102611	0.012256	0.038144	0.034087	0.050729	0.003356	0.031435	0.023437	0.00
50%	0.055931	0.122081	0.021873	0.060709	0.057011	0.056292	0.005176	0.045696	0.032015	0.00
75%	0.083777	0.144946	0.044107	0.094255	0.096315	0.062045	0.007679	0.064200	0.042586	0.00
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

In [91]:
```
station_day_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 108035 entries, 0 to 108034
Data columns (total 19 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   Date        108035 non-null  datetime64[ns]
 1   PM2.5       108035 non-null  float64
 2   PM10        108035 non-null  float64
 3   NO          108035 non-null  float64
 4   NO2         108035 non-null  float64
 5   NOx         108035 non-null  float64
 6   NH3         108035 non-null  float64
 7   CO          108035 non-null  float64
 8   SO2         108035 non-null  float64
 9   O3          108035 non-null  float64
 10  Benzene     108035 non-null  float64
 11  Toluene     108035 non-null  float64
 12  Xylene      108035 non-null  float64
 13  AQI         108035 non-null  float64
 14  AQI_Bucket  108035 non-null  object
 15  Year        108035 non-null  int64
 16  State       108035 non-null  object
 17  latitude    108035 non-null  float64
 18  longitude   108035 non-null  float64
dtypes: datetime64[ns](1), float64(15), int64(1), object(2)
memory usage: 16.5+ MB
```

In [424…
```
#station_day_2.to_csv('station_day_2.csv')
```

Create a copy of data frame to drop columns that are not required in correlation matrix.

In [92]:
```
station_day_2crop = station_day_2.copy()
```

In [93]:
```
station_day_2crop.head()
```

Out[93]:

	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket	Ye
0	2017-11-24	0.071341	0.115741	0.003702	0.046067	0.026517	0.029077	0.000569	0.054948	0.113449	0.000374	0.013015	0.000587	0.060755	Moderate	20
1	2017-11-25	0.081382	0.124491	0.003043	0.045733	0.025832	0.025568	0.000683	0.077847	0.131964	0.000440	0.014290	0.000352	0.086232	Moderate	20
2	2017-11-26	0.078302	0.129051	0.002660	0.058008	0.031756	0.024517	0.000796	0.137753	0.121943	0.000483	0.017478	0.000470	0.092602	Moderate	20
3	2017-11-27	0.088742	0.135311	0.014022	0.068833	0.046554	0.030796	0.000626	0.171642	0.116097	0.000637	0.016775	0.000704	0.093092	Moderate	20
4	2017-11-28	0.064161	0.104081	0.005426	0.062628	0.036375	0.027239	0.000512	0.097066	0.143480	0.000374	0.011037	0.000411	0.088192	Moderate	20

In [94]:
```
station_day_2crop.drop(['latitude','longitude','Year'],axis=1,inplace=True)
```

Check that the columns are dropped.

In [95]:
```
station_day_2crop.head()
```

Out[95]:

	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket	
0	2017-11-24	0.071341	0.115741	0.003702	0.046067	0.026517	0.029077	0.000569	0.054948	0.113449	0.000374	0.013015	0.000587	0.060755	Moderate	Ar Pra
1	2017-11-25	0.081382	0.124491	0.003043	0.045733	0.025832	0.025568	0.000683	0.077847	0.131964	0.000440	0.014290	0.000352	0.086232	Moderate	Ar Pra
2	2017-11-26	0.078302	0.129051	0.002660	0.058008	0.031756	0.024517	0.000796	0.137753	0.121943	0.000483	0.017478	0.000470	0.092602	Moderate	Ar Pra
3	2017-11-27	0.088742	0.135311	0.014022	0.068833	0.046554	0.030796	0.000626	0.171642	0.116097	0.000637	0.016775	0.000704	0.093092	Moderate	Ar Pra
4	2017-11-28	0.064161	0.104081	0.005426	0.062628	0.036375	0.027239	0.000512	0.097066	0.143480	0.000374	0.011037	0.000411	0.088192	Moderate	Ar Pra

Generate and plot correlation matrix.

In [96]:
```
pollut_correlation = station_day_2crop.corr()
```

In [97]:
```
pollut_correlation
```

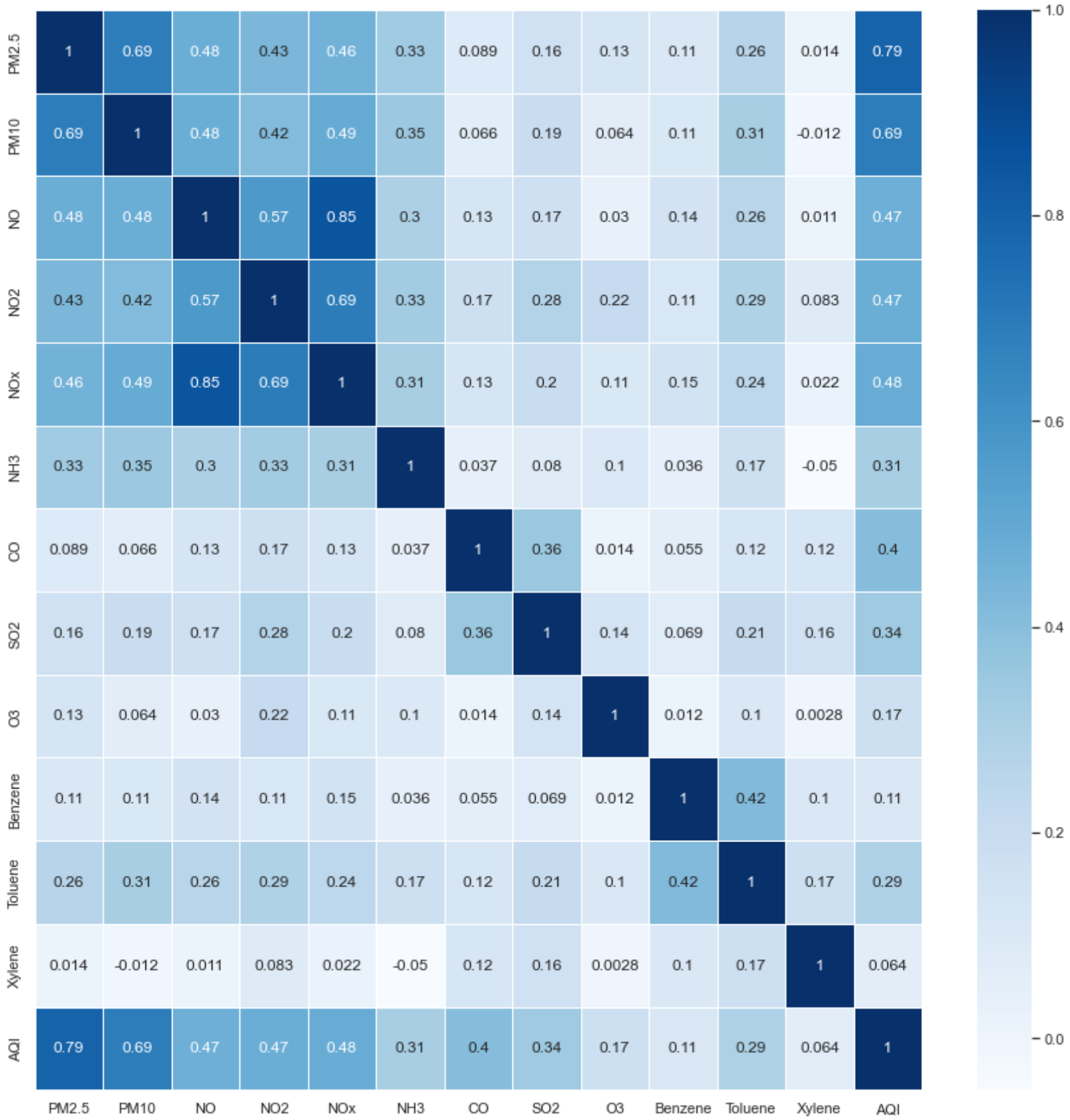
Out[97]:

	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI
PM2.5	1.000000	0.689865	0.475926	0.430728	0.456854	0.334874	0.089130	0.164388	0.133146	0.105547	0.262099	0.014183	0.793215
PM10	0.689865	1.000000	0.475274	0.415650	0.489424	0.348346	0.065876	0.192791	0.063613	0.106865	0.314234	-0.012207	0.692166
NO	0.475926	0.475274	1.000000	0.572610	0.850129	0.295311	0.130215	0.168216	0.029948	0.143590	0.259061	0.010639	0.469517
NO2	0.430728	0.415650	0.572610	1.000000	0.694988	0.330701	0.172530	0.277746	0.215619	0.114045	0.285210	0.082606	0.474933
NOx	0.456854	0.489424	0.850129	0.694988	1.000000	0.313174	0.130318	0.197838	0.108883	0.149844	0.243740	0.021693	0.483184
NH3	0.334874	0.348346	0.295311	0.330701	0.313174	1.000000	0.037150	0.079512	0.099755	0.036483	0.173028	-0.050312	0.314119
CO	0.089130	0.065876	0.130215	0.172530	0.130318	0.037150	1.000000	0.355839	0.013817	0.054529	0.118493	0.118008	0.402749
SO2	0.164388	0.192791	0.168216	0.277746	0.197838	0.079512	0.355839	1.000000	0.137527	0.069165	0.209961	0.163223	0.342670
O3	0.133146	0.063613	0.029948	0.215619	0.108883	0.099755	0.013817	0.137527	1.000000	0.012108	0.101639	0.002803	0.174652
Benzene	0.105547	0.106865	0.143590	0.114045	0.149844	0.036483	0.054529	0.069165	0.012108	1.000000	0.420456	0.103685	0.113580
Toluene	0.262099	0.314234	0.259061	0.285210	0.243740	0.173028	0.118493	0.209961	0.101639	0.420456	1.000000	0.174521	0.294295
Xylene	0.014183	-0.012207	0.010639	0.082606	0.021693	-0.050312	0.118008	0.163223	0.002803	0.103685	0.174521	1.000000	0.063742
AQI	0.793215	0.692166	0.469517	0.474933	0.483184	0.314119	0.402749	0.342670	0.174652	0.113580	0.294295	0.063742	1.000000

In [98]:
```
plt.style = sns
plt.figure(figsize=(15,15))
sns.heatmap(pollut_correlation, cmap='Blues',annot=True,linewidths=1,)
```

```
<AxesSubplot:>
```

Out[98]:



In [ ]:

In [ ]:

Load power generation data into a pandas data frame.

In [99]:
```
Ind_pgd = pd.read_csv('file_02.csv')
```

In [100…
```
Ind_pgd.head()
```

Out[100…

	index	Date	Region	Thermal Generation Actual (in MU)	Thermal Generation Estimated (in MU)	Nuclear Generation Actual (in MU)	Nuclear Generation Estimated (in MU)	Hydro Generation Actual (in MU)	Hydro Generation Estimated (in MU)
0	0	2017-09-01	Northern	624.23	484.21	30.36	35.57	273.27	320.81
1	1	2017-09-01	Western	1,106.89	1,024.33	25.17	3.81	72.00	21.53
2	2	2017-09-01	Southern	576.66	578.55	62.73	49.80	111.57	64.78
3	3	2017-09-01	Eastern	441.02	429.39	NaN	NaN	85.94	69.36
4	4	2017-09-01	NorthEastern	29.11	15.91	NaN	NaN	24.64	21.21

In [101…
```
Ind_pgd.shape
```

Out[101…
```
(4945, 9)
```

In [102…
```
Ind_pgd.Date.min()
```

Out[102…
```
'2017-09-01'
```

In [103…
```
Ind_pgd.Date.max()
```

Out[103…
```
'2020-08-01'
```

Check the number of missing values in all the columns of the data frame.

In [104…
```
Ind_pgd.isnull().sum()
```

Out[104…
```
index                                  0
Date                                   0
Region                                 0
Thermal Generation Actual (in MU)      0
Thermal Generation Estimated (in MU)   0
Nuclear Generation Actual (in MU)      1978
Nuclear Generation Estimated (in MU)   1978
Hydro Generation Actual (in MU)        0
Hydro Generation Estimated (in MU)     0
dtype: int64
```

Create a true copy of the data frame, on which further modifications would be done.

In [105…
```
Ind_pgd_copy = Ind_pgd.copy()
```

In [106…
```
Ind_pgd_copy.columns
```

Out[106…
```
Index(['index', 'Date', 'Region', 'Thermal Generation Actual (in MU)',
       'Thermal Generation Estimated (in MU)',
       'Nuclear Generation Actual (in MU)',
       'Nuclear Generation Estimated (in MU)',
       'Hydro Generation Actual (in MU)',
       'Hydro Generation Estimated (in MU)'],
      dtype='object')
```

In [107…
```
Ind_pgd_copy.drop(['index','Thermal Generation Estimated (in MU)',
       'Nuclear Generation Actual (in MU)',
       'Nuclear Generation Estimated (in MU)',
       'Hydro Generation Actual (in MU)',
       'Hydro Generation Estimated (in MU)'], axis=1,inplace=True)
```

In [108…
```
Ind_pgd_copy.head()
```

Out[108…

	Date	Region	Thermal Generation Actual (in MU)
0	2017-09-01	Northern	624.23
1	2017-09-01	Western	1,106.89
2	2017-09-01	Southern	576.66
3	2017-09-01	Eastern	441.02
4	2017-09-01	NorthEastern	29.11

In [109…
```
Ind_pgd_copy.shape
```

Out[109…
```
(4945, 3)
```

In [110…
```
Ind_pgd_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4945 entries, 0 to 4944
Data columns (total 3 columns):
 #   Column                            Non-Null Count  Dtype 
---  ------                            --------------  ----- 
 0   Date                              4945 non-null   object
 1   Region                            4945 non-null   object
 2   Thermal Generation Actual (in MU)  4945 non-null   object
dtypes: object(3)
memory usage: 116.0+ KB
```

Load lat-long data for different regions into a pandas data frame.

In [111…
```
reg_cord = pd.read_csv('region_cordinates.csv')
```

In [112…
```
reg_cord
```

Out[112...

	Region	Latitude	Longitude
0	NorthEastern	25.5736	93.2473
1	Eastern	22.8962	85.9800
2	Western	23.9074	72.7097
3	Northern	28.6139	77.2090
4	Southern	12.2602	77.1461

Check the unique values in the region column.

In [113...
```
Ind_pgd_copy.Region.unique()
```

Out[113...
```
array(['Northern', 'Western', 'Southern', 'Eastern', 'NorthEastern'],
      dtype=object)
```

Rename lat-long column names to avoid confusion later after merging of data frames.

In [114...
```
reg_cord.rename(columns={'Latitude':'Reg_Lat','Longitude':'Reg_Lon'}, inplace=True)
```

In [115...
```
reg_cord
```

Out[115...

	Region	Reg_Lat	Reg_Lon
0	NorthEastern	25.5736	93.2473
1	Eastern	22.8962	85.9800
2	Western	23.9074	72.7097
3	Northern	28.6139	77.2090
4	Southern	12.2602	77.1461

Loading data set of states and regions into a pandas data frame.

In [116...
```
state_reg = pd.read_csv('State_Region_corrected.csv')
```

In [117...
```
state_reg.head()
```

Out[117...

	State / Union territory (UT)	Area (km2)	Region	National Share (%)
0	Rajasthan	342239	Northern	10.55
1	Madhya Pradesh	308350	Central	9.37
2	Maharashtra	307713	Western	9.36
3	Uttar Pradesh	240928	Northern	7.33
4	Gujarat	196024	Western	5.96

In [118...
```
state_reg.columns
```

Out[118...
```
Index(['State / Union territory (UT)', 'Area (km2)', 'Region',
       'National Share (%)'],
      dtype='object')
```

In [119...
```
state_reg.drop(['Area (km2)','National Share (%)'],axis=1,inplace=True)
```

In [120...
```
state_reg.head()
```

Out[120...

	State / Union territory (UT)	Region
0	Rajasthan	Northern
1	Madhya Pradesh	Central
2	Maharashtra	Western
3	Uttar Pradesh	Northern
4	Gujarat	Western

In [121...
```
state_reg.rename(columns={'State / Union territory (UT)': 'State'}, inplace=True)
```

In [122...
```
state_reg.head()
```

Out[122...

	State	Region

	State	Region
0	Rajasthan	Northern
1	Madhya Pradesh	Central
2	Maharashtra	Western
3	Uttar Pradesh	Northern
4	Gujarat	Western

Check the rows with region value as 'Central'.

In [450…
```
state_reg[state_reg.Region == 'Central']
```

Out[450…

	State	Region
1	Madhya Pradesh	Central
8	Chhattisgarh	Central

Replace 'Central' region values with 'Northern' for ease of plotting on map with lat-long values.

In [123…
```
state_reg.Region.replace('Central','Northern', inplace=True)
```

In [124…
```
state_reg.head()
```

Out[124…

	State	Region
0	Rajasthan	Northern
1	Madhya Pradesh	Northern
2	Maharashtra	Western
3	Uttar Pradesh	Northern
4	Gujarat	Western

In [125…
```
state_reg.sort_values('State', inplace=True)
```

In [126…
```
station_day_2.State.unique()
```

Out[126…
```
array(['Andhra Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Delhi',
       'Gujarat', 'Haryana', 'Jharkhand', 'Karnataka', 'Kerala',
       'Maharashtra', 'Meghalaya', 'Madhya Pradesh', 'Mizoram', 'Odisha',
       'Punjab', 'Rajasthan', 'Telangana', 'Tamil Nadu', 'Uttar Pradesh',
       'West Bengal'], dtype=object)
```

In [127…
```
state_reg.State.unique()
```

Out[127…
```
array(['Andhra Pradesh', 'Arunachal Pradesh', 'Assam', 'Bihar',
       'Chandigarh', 'Chhattisgarh',
       'Dadra and Nagar Haveli and Daman and Diu', 'Delhi', 'Goa',
       'Gujarat', 'Haryana', 'Himachal Pradesh', 'Jammu and Kashmir',
       'Jharkhand', 'Karnataka', 'Kerala', 'Ladakh', 'Madhya Pradesh',
       'Maharashtra', 'Manipur', 'Meghalaya', 'Mizoram', 'Nagaland',
       'Odisha', 'Puducherry', 'Punjab', 'Rajasthan', 'Sikkim',
       'Tamil Nadu', 'Telangana', 'Tripura', 'Uttar Pradesh',
       'Uttarakhand', 'West Bengal'], dtype=object)
```

Drop rows with state values not present in primary dataset.

In [128…
```
drop_state_list = ['Arunachal Pradesh', 'Chhattisgarh', 'Dadra and Nagar Haveli and Daman and Diu',
                   'Goa', 'Himachal Pradesh', 'Jammu and Kashmir', 'Ladakh', 'Manipur',
                    'Nagaland', 'Puducherry', 'Sikkim', 'Tripura', 'Uttarakhand']

for j in state_reg.State:
    if j in drop_state_list:
        state_reg.drop(index=state_reg[state_reg.State == j].index, inplace=True)
```

Check that the values of state columns in both data frames match.

In [129…
```
station_day_2.State.unique()
```

Out[129…
```
array(['Andhra Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Delhi',
       'Gujarat', 'Haryana', 'Jharkhand', 'Karnataka', 'Kerala',
       'Maharashtra', 'Meghalaya', 'Madhya Pradesh', 'Mizoram', 'Odisha',
       'Punjab', 'Rajasthan', 'Telangana', 'Tamil Nadu', 'Uttar Pradesh',
       'West Bengal'], dtype=object)
```

In [130…
```
state_reg.State.unique()
```

Out[130…
```
array(['Andhra Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Delhi',
       'Gujarat', 'Haryana', 'Jharkhand', 'Karnataka', 'Kerala',
       'Madhya Pradesh', 'Maharashtra', 'Meghalaya', 'Mizoram', 'Odisha',
       'Punjab', 'Rajasthan', 'Tamil Nadu', 'Telangana', 'Uttar Pradesh',
       'West Bengal'], dtype=object)
```

In [471…
```
state_reg.head()
```

Out[471…

	State	Region
6	Andhra Pradesh	Southern
15	Assam	Northeastern
11	Bihar	Eastern
33	Chandigarh	Northern
30	Delhi	Northern

In [131…
```
state_reg.duplicated().sum()
```

Out[131…
```
0
```

In [132…
```
station_day_2.shape
```

Out[132…
```
(108035, 19)
```

Create a true copy of the data frame of primary dataset to modify it further.

In [133…
```
station_day_3 = station_day_2.copy()
```

In [134…
```
station_day_3.shape
```

Out[134…
```
(108035, 19)
```

In [135…
```
station_day_3 = pd.merge(station_day_2,state_reg,how='left',left_on='State',right_on='State')
```

In [136…
```
station_day_3.shape
```

Out[136…
```
(108035, 20)
```

In [137…
```
station_day_3.head()
```

Out[137…

	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket	Ye
0	2017-11-24	0.071341	0.115741	0.003702	0.046067	0.026517	0.029077	0.000569	0.054948	0.113449	0.000374	0.013015	0.000587	0.060755	Moderate	20
1	2017-11-25	0.081382	0.124491	0.003043	0.045733	0.025832	0.025568	0.000683	0.077847	0.131964	0.000440	0.014290	0.000352	0.086232	Moderate	20
2	2017-11-26	0.078302	0.129051	0.002660	0.058008	0.031756	0.024517	0.000796	0.137753	0.121943	0.000483	0.017478	0.000470	0.092602	Moderate	20
3	2017-11-27	0.088742	0.135311	0.014022	0.068833	0.046554	0.030796	0.000626	0.171642	0.116097	0.000637	0.016775	0.000704	0.093092	Moderate	20
4	2017-11-28	0.064161	0.104081	0.005426	0.062628	0.036375	0.027239	0.000512	0.097066	0.143480	0.000374	0.011037	0.000411	0.088192	Moderate	20

Check for unique values of Region columns in different data frame to identify any mismatch in the values.

In [138…
```
station_day_3.Region.unique()
```

Out[138…
```
array(['Southern', 'Northeastern', 'Eastern', 'Northern', 'Western'],
      dtype=object)
```

In [139…
```
reg_cord.Region.unique()
```

Out[139…
```
array(['NorthEastern', 'Eastern', 'Western ', 'Northern', 'Southern'],
      dtype=object)
```

Rectify the mismatches identified.

In [145…
```
reg_cord.Region.replace({'NorthEastern':'Northeastern', 'Western ':'Western'}, inplace=True)
```

Cofirm that the rectifications are effected.

In [146…
```
reg_cord.Region.unique()
```

Out[146…
```
array(['Northeastern', 'Eastern', 'Western', 'Northern', 'Southern'],
      dtype=object)
```

In [147…
```
station_day_3.shape
```

Out[147…
```
(108035, 20)
```

In [148…
```
reg_cord.duplicated().sum()
```

Out[148…
```
0
```

In [149…
```
station_day_3 = pd.merge(station_day_3,reg_cord,how='left',left_on='Region',right_on='Region')
```

In [150…
```
station_day_3.shape
```

Out[150…
```
(108035, 22)
```

In [151…
```
station_day_3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 108035 entries, 0 to 108034
Data columns (total 22 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   Date        108035 non-null  datetime64[ns]
 1   PM2.5       108035 non-null  float64
 2   PM10        108035 non-null  float64
 3   NO          108035 non-null  float64
 4   NO2         108035 non-null  float64
 5   NOx         108035 non-null  float64
 6   NH3         108035 non-null  float64
 7   CO          108035 non-null  float64
 8   SO2         108035 non-null  float64
 9   O3          108035 non-null  float64
 10  Benzene     108035 non-null  float64
 11  Toluene     108035 non-null  float64
 12  Xylene      108035 non-null  float64
 13  AQI         108035 non-null  float64
 14  AQI_Bucket  108035 non-null  object
 15  Year        108035 non-null  int64
 16  State       108035 non-null  object
 17  latitude    108035 non-null  float64
 18  longitude   108035 non-null  float64
 19  Region      108035 non-null  object
 20  Reg_Lat     108035 non-null  float64
 21  Reg_Lon     108035 non-null  float64
dtypes: datetime64[ns](1), float64(17), int64(1), object(3)
memory usage: 19.0+ MB
```

In [152…
```
station_day_3.head()
```

Out[152…

	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	…	Xylene	AQI	AQI_Bucket	Year	State	la
0	2017-11-24	0.071341	0.115741	0.003702	0.046067	0.026517	0.029077	0.000569	0.054948	0.113449	…	0.000587	0.060755	Moderate	2017	Andhra Pradesh	14.7
1	2017-11-25	0.081382	0.124491	0.003043	0.045733	0.025832	0.025568	0.000683	0.077847	0.131964	…	0.000352	0.086232	Moderate	2017	Andhra Pradesh	14.7
2	2017-11-26	0.078302	0.129051	0.002660	0.058008	0.031756	0.024517	0.000796	0.137753	0.121943	…	0.000470	0.092602	Moderate	2017	Andhra Pradesh	14.7
3	2017-11-27	0.088742	0.135311	0.014022	0.068833	0.046554	0.030796	0.000626	0.171642	0.116097	…	0.000704	0.093092	Moderate	2017	Andhra Pradesh	14.7
4	2017-11-28	0.064161	0.104081	0.005426	0.062628	0.036375	0.027239	0.000512	0.097066	0.143480	…	0.000411	0.088192	Moderate	2017	Andhra Pradesh	14.7

5 rows × 22 columns

In [490…
```
#station_day_3.to_csv('station_day_3.csv')
```

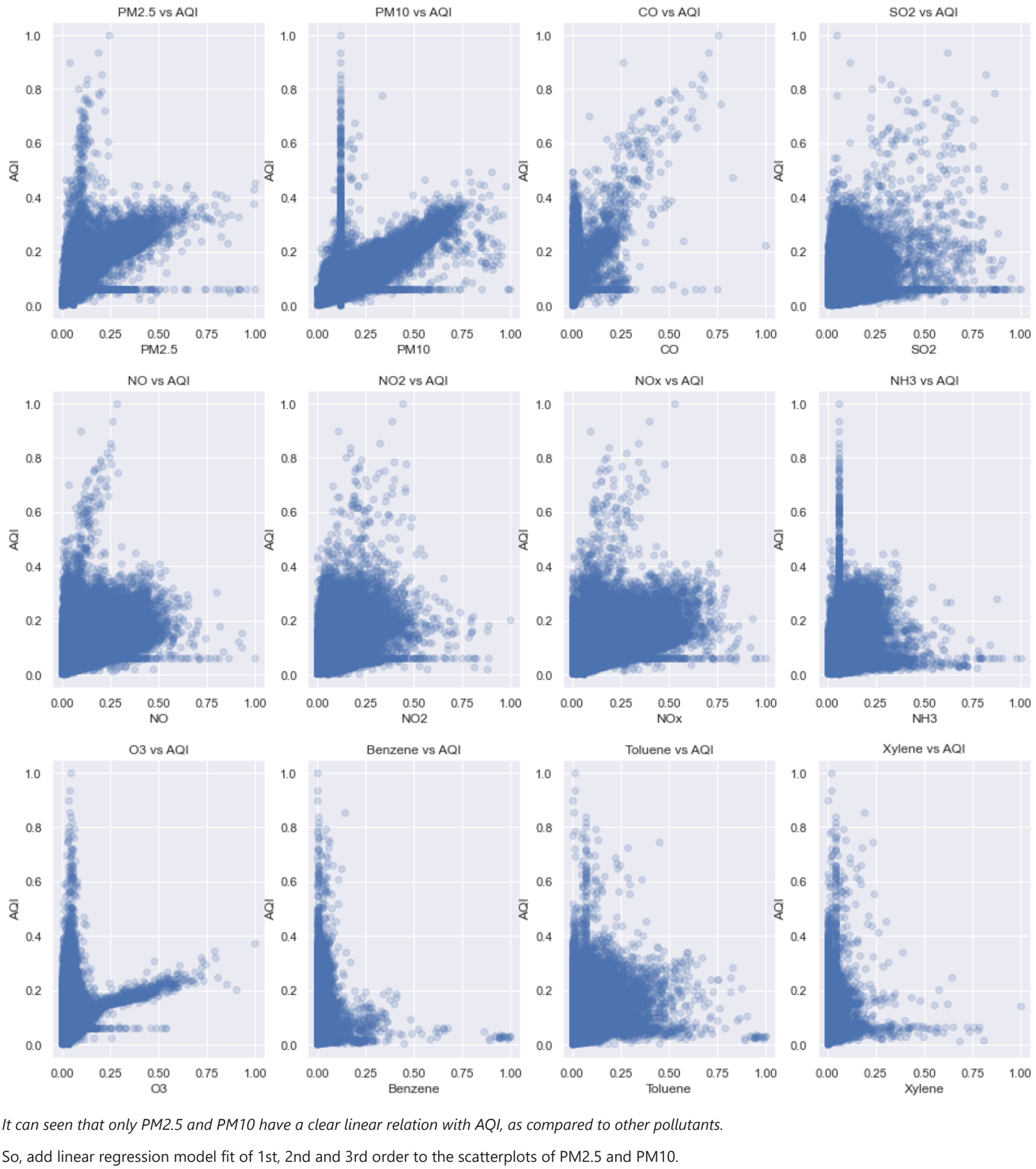
In [154…
```
station_day_3.columns
```

Out[154…
```
Index(['Date', 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3',
       'Benzene', 'Toluene', 'Xylene', 'AQI', 'AQI_Bucket', 'Year', 'State',
       'latitude', 'longitude', 'Region', 'Reg_Lat', 'Reg_Lon'],
      dtype='object')
```

Make scatterplots to see the variation in AQI values for variation in values of different pollutants.

In [155…

```
fig = plt.figure(figsize=(16,5))

scp = fig.add_subplot(141)
scp.scatter(station_day_3['PM2.5'],station_day_3.AQI, alpha=.2)
scp.set_title('PM2.5 vs AQI')
scp.set_xlabel('PM2.5')
scp.set_ylabel('AQI')

scp = fig.add_subplot(142)
scp.scatter(station_day_3.PM10,station_day_3.AQI, alpha=.2)
scp.set_title('PM10 vs AQI')
scp.set_xlabel('PM10')
scp.set_ylabel('AQI')

scp = fig.add_subplot(143)
scp.scatter(station_day_3.CO,station_day_3.AQI, alpha=.2)
scp.set_title('CO vs AQI')
scp.set_xlabel('CO')
scp.set_ylabel('AQI')

scp = fig.add_subplot(144)
scp.scatter(station_day_3.SO2,station_day_3.AQI, alpha=.2)
scp.set_title('SO2 vs AQI')
scp.set_xlabel('SO2')
scp.set_ylabel('AQI')

plt.show()

fig = plt.figure(figsize=(16,5))

scp = fig.add_subplot(141)
scp.scatter(station_day_3.NO,station_day_3.AQI, alpha=.2)
scp.set_title('NO vs AQI')
scp.set_xlabel('NO')
scp.set_ylabel('AQI')

scp = fig.add_subplot(142)
scp.scatter(station_day_3.NO2,station_day_3.AQI, alpha=.2)
scp.set_title('NO2 vs AQI')
scp.set_xlabel('NO2')
scp.set_ylabel('AQI')

scp = fig.add_subplot(143)
scp.scatter(station_day_3.NOx,station_day_3.AQI, alpha=.2)
scp.set_title('NOx vs AQI')
scp.set_xlabel('NOx')
scp.set_ylabel('AQI')

scp = fig.add_subplot(144)
scp.scatter(station_day_3.NH3,station_day_3.AQI, alpha=.2)
scp.set_title('NH3 vs AQI')
scp.set_xlabel('NH3')
scp.set_ylabel('AQI')

plt.show()

fig = plt.figure(figsize=(16,5))

scp = fig.add_subplot(141)
scp.scatter(station_day_3.O3,station_day_3.AQI, alpha=.2)
scp.set_title('O3 vs AQI')
scp.set_xlabel('O3')
scp.set_ylabel('AQI')

scp = fig.add_subplot(142)
scp.scatter(station_day_3.Benzene,station_day_3.AQI, alpha=.2)
scp.set_title('Benzene vs AQI')
scp.set_xlabel('Benzene')
scp.set_ylabel('AQI')

scp = fig.add_subplot(143)
scp.scatter(station_day_3.Toluene,station_day_3.AQI, alpha=.2)
scp.set_title('Toluene vs AQI')
scp.set_xlabel('Toluene')
scp.set_ylabel('AQI')

scp = fig.add_subplot(144)
scp.scatter(station_day_3.Xylene,station_day_3.AQI, alpha=.2)
scp.set_title('Xylene vs AQI')
scp.set_xlabel('Xylene')
scp.set_ylabel('AQI')

plt.show()
```

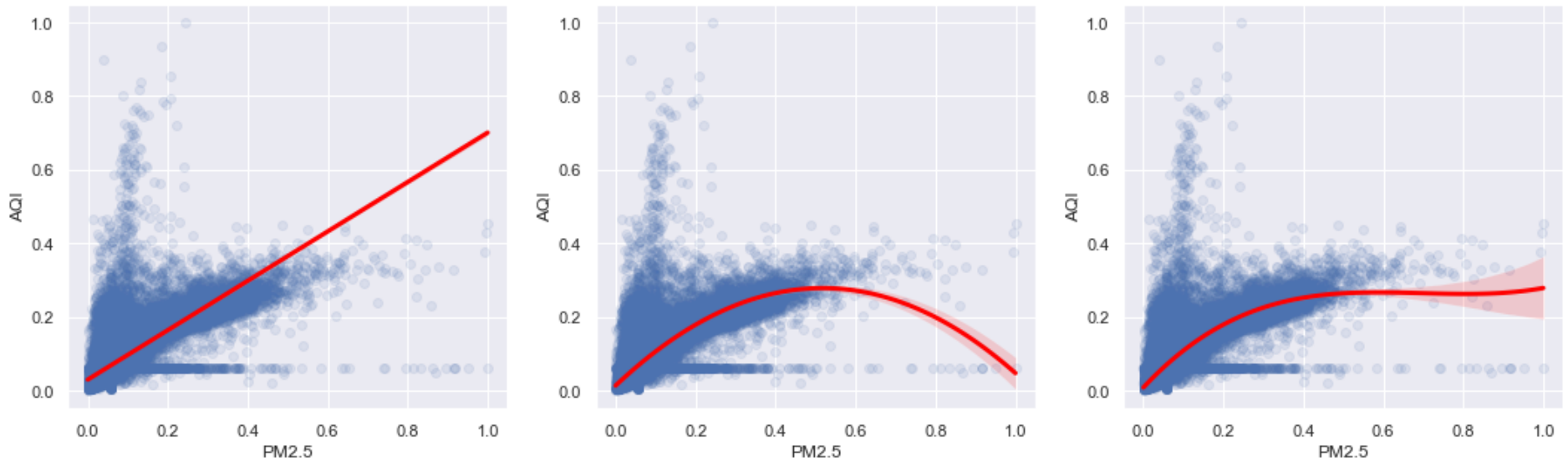


It can seen that only PM2.5 and PM10 have a clear linear relation with AQI, as compared to other pollutants.

So, add linear regression model fit of 1st, 2nd and 3rd order to the scatterplots of PM2.5 and PM10.

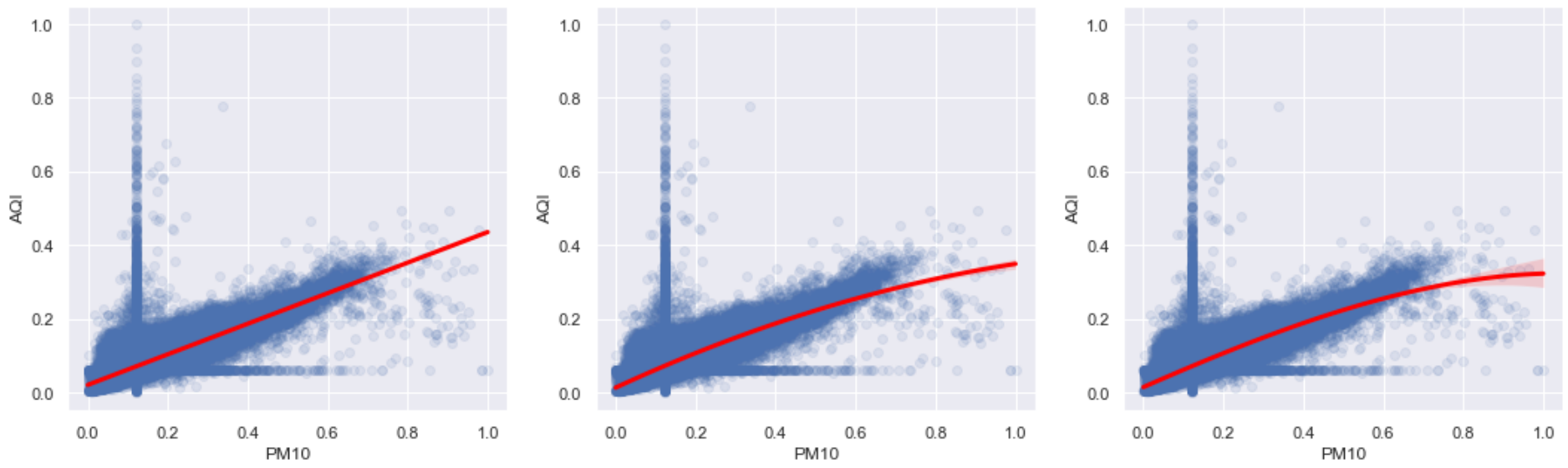
In [156…

```
sns.set(color_codes=True)

fig,axes = plt.subplots(1,3,figsize=(18,5))
sns.regplot(x=station_day_3['PM2.5'], y=station_day_3.AQI, data=station_day_3, scatter_kws={'alpha':0.1},
            line_kws={'lw': 3, 'color': 'red'}, ax = axes[0]);
sns.regplot(x=station_day_3['PM2.5'], y=station_day_3.AQI, data=station_day_3, order=2, scatter_kws={'alpha':0.1},
            line_kws={'lw': 3, 'color': 'red'}, ax = axes[1]);
sns.regplot(x=station_day_3['PM2.5'], y=station_day_3.AQI, data=station_day_3, order=3, scatter_kws={'alpha':0.1},
            line_kws={'lw': 3, 'color': 'red'}, ax = axes[2]);
plt.show()
```



In [157…

```
sns.set(color_codes=True)

fig,axes = plt.subplots(1,3,figsize=(18,5))
sns.regplot(x=station_day_3.PM10, y=station_day_3.AQI, data=station_day_3, scatter_kws={'alpha':0.1},
            line_kws={'lw': 3, 'color': 'red'}, ax = axes[0]);
sns.regplot(x=station_day_3.PM10, y=station_day_3.AQI, data=station_day_3, order=2, scatter_kws={'alpha':0.1},
            line_kws={'lw': 3, 'color': 'red'}, ax = axes[1]);
sns.regplot(x=station_day_3.PM10, y=station_day_3.AQI, data=station_day_3, order=3, scatter_kws={'alpha':0.1},
            line_kws={'lw': 3, 'color': 'red'}, ax = axes[2]);
plt.show()
```

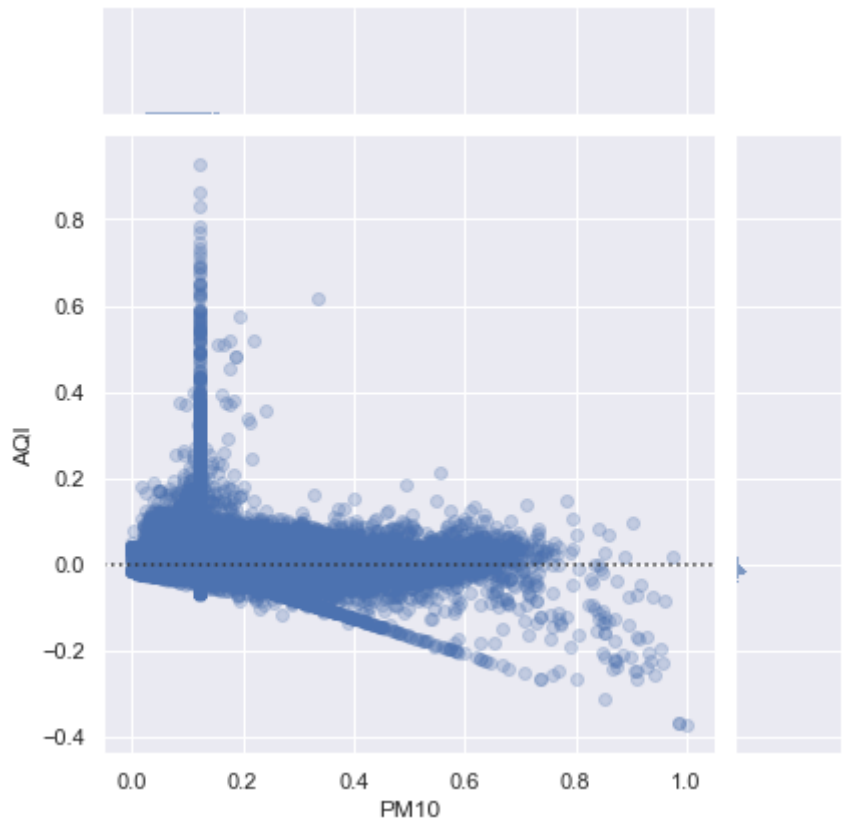


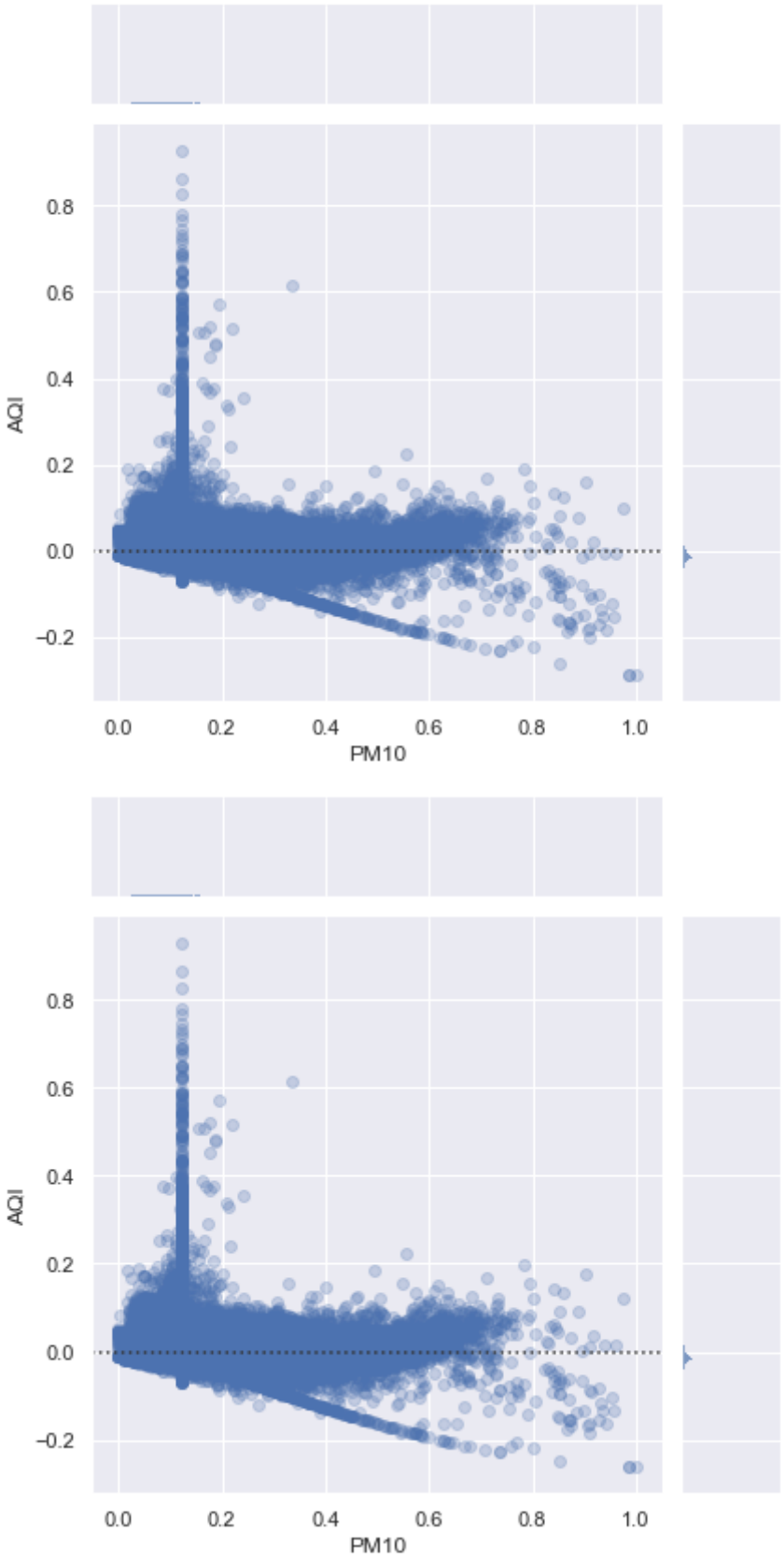
It is clear that PM10 is more linearly related tot AQI than PM2.5.

Generate joitplot AQI and PM10 to understand theri variation between them anf also thier individual distributions for 1st, 2nd and 3rd order linear regressions.

In [158…

```
sns.jointplot(x=station_day_3.PM10, y=station_day_3.AQI, data=station_day_3, kind="resid", scatter_kws={'alpha':0.25});
plt.show()

sns.jointplot(x=station_day_3.PM10, y=station_day_3.AQI, data=station_day_3, order=2, kind="resid", scatter_kws={'alpha':0.25});
plt.show()

sns.jointplot(x=station_day_3.PM10, y=station_day_3.AQI, data=station_day_3, order=3, kind="resid", scatter_kws={'alpha':0.25});
plt.show()
```





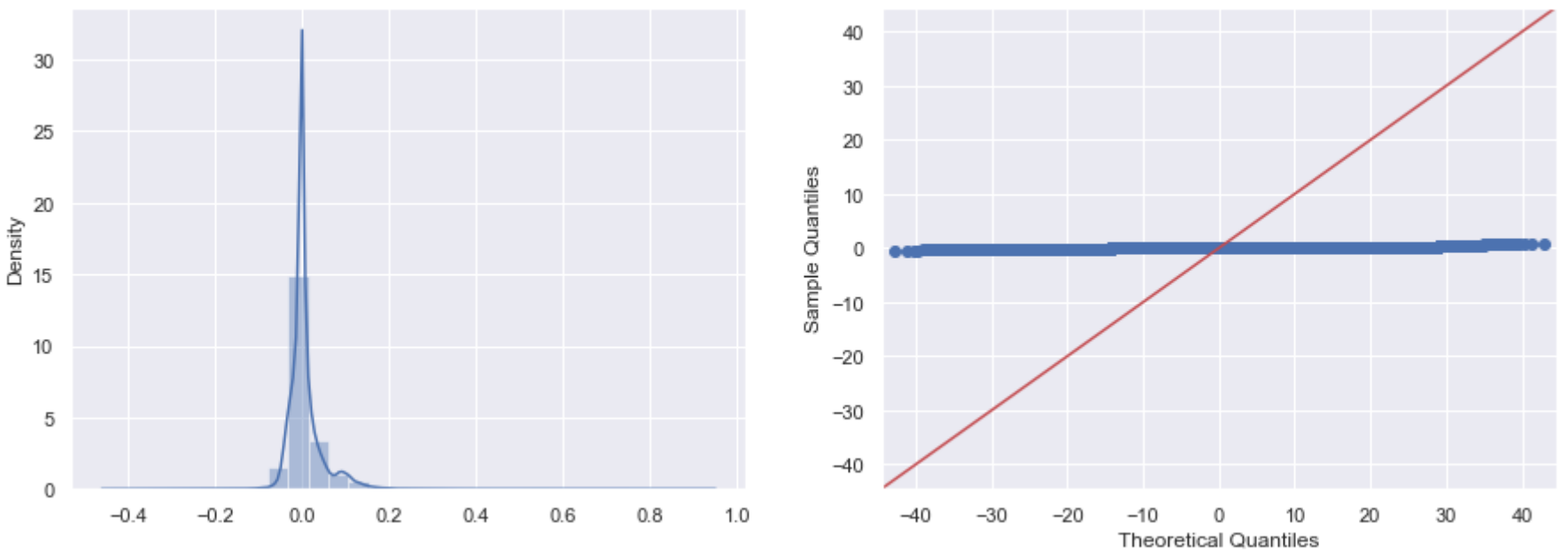
The jointplots don't seem to give a fair idea of the relaionships and distributions.

So, plot distribution plot and quartile-quartile plot of Ordinary least squares to underdstand how good the linear regerssion fit is.

In [159…

```
olsmod = sm.OLS(station_day_3.loc[:,'AQI'], station_day_3.loc[:,'PM10'])
olsres = olsmod.fit()
#print(olsres.summary())

fig,axes = plt.subplots(1,2,figsize=(15,5))
sns.distplot(olsres.resid,bins=30,ax=axes[0])
sm.qqplot(olsres.resid, scale = 10, line ='45',ax=axes[1])

plt.show()
```

```
C:\Users\jraja\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Further process data before exporting to Tableau for visual analysis.

In [160…
```
Ind_pgd_copy.head()
```

Out[160…

	Date	Region	Thermal Generation Actual (in MU)
0	2017-09-01	Northern	624.23
1	2017-09-01	Western	1,106.89
2	2017-09-01	Southern	576.66
3	2017-09-01	Eastern	441.02
4	2017-09-01	NorthEastern	29.11

In [161…
```
reg_cord
```

Out[161…

	Region	Reg_Lat	Reg_Lon
0	Northeastern	25.5736	93.2473
1	Eastern	22.8962	85.9800
2	Western	23.9074	72.7097
3	Northern	28.6139	77.2090
4	Southern	12.2602	77.1461

In [162…
```
Ind_pgd_copy.columns
```

Out[162…
```
Index(['Date', 'Region', 'Thermal Generation Actual (in MU)'], dtype='object')
```

In [163…
```
reg_cord.columns
```

Out[163…
```
Index(['Region', 'Reg_Lat', 'Reg_Lon'], dtype='object')
```

In [164…
```
reg_cord.Region.unique()
```

Out[164…
```
array(['Northeastern', 'Eastern', 'Western', 'Northern', 'Southern'],
      dtype=object)
```

In [165…
```
Ind_pgd_copy.Region.unique()
```

Out[165…
```
array(['Northern', 'Western', 'Southern', 'Eastern', 'NorthEastern'],
      dtype=object)
```

In [166…
```
Ind_pgd_copy.Region.replace('NorthEastern','Northeastern', inplace=True)
```

In [167…
```
Ind_pgd_copy.Region.unique()
```

Out[167…
```
array(['Northern', 'Western', 'Southern', 'Eastern', 'Northeastern'],
      dtype=object)
```

In [168…
```
Ind_pgd_copy.shape
```

Out[168…
```
(4945, 3)
```

In [169…
```
Ind_pgd_copy = pd.merge(Ind_pgd_copy,reg_cord,how='left',left_on='Region',right_on='Region')
```

In [170…
```
Ind_pgd_copy.shape
```

Out[170…
```
(4945, 5)
```

In [171…
```
Ind_pgd_copy.head()
```

Out[171…

	Date	Region	Thermal Generation Actual (in MU)	Reg_Lat	Reg_Lon
0	2017-09-01	Northern	624.23	28.6139	77.2090
1	2017-09-01	Western	1,106.89	23.9074	72.7097
2	2017-09-01	Southern	576.66	12.2602	77.1461
3	2017-09-01	Eastern	441.02	22.8962	85.9800
4	2017-09-01	Northeastern	29.11	25.5736	93.2473

In [172…
```
Ind_pgd_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4945 entries, 0 to 4944
Data columns (total 5 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Date                              4945 non-null   object 
 1   Region                            4945 non-null   object 
 2   Thermal Generation Actual (in MU)  4945 non-null   object 
 3   Reg_Lat                           4945 non-null   float64
 4   Reg_Lon                           4945 non-null   float64
dtypes: float64(2), object(3)
memory usage: 231.8+ KB
```

In [173…
```
Ind_pgd_copy.Date = pd.to_datetime(Ind_pgd_copy.Date)
```

In [174…
```
Ind_pgd_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4945 entries, 0 to 4944
Data columns (total 5 columns):
 #   Column                            Non-Null Count  Dtype         
---  ------                            --------------  -----         
 0   Date                              4945 non-null   datetime64[ns]
 1   Region                            4945 non-null   object        
 2   Thermal Generation Actual (in MU)  4945 non-null   object        
 3   Reg_Lat                           4945 non-null   float64       
 4   Reg_Lon                           4945 non-null   float64       
dtypes: datetime64[ns](1), float64(2), object(2)
memory usage: 231.8+ KB
```

In [175…
```
Ind_pgd_copy['Year'] = Ind_pgd_copy.Date.dt.year
```

In [176…
```
Ind_pgd_copy.head()
```

Out[176…

	Date	Region	Thermal Generation Actual (in MU)	Reg_Lat	Reg_Lon	Year
0	2017-09-01	Northern	624.23	28.6139	77.2090	2017
1	2017-09-01	Western	1,106.89	23.9074	72.7097	2017
2	2017-09-01	Southern	576.66	12.2602	77.1461	2017
3	2017-09-01	Eastern	441.02	22.8962	85.9800	2017
4	2017-09-01	Northeastern	29.11	25.5736	93.2473	2017

In [177…
```
#Ind_pgd_copy.to_csv('Ind_powgen.csv')
```

In [178…
```
Ind_pgd_copy.Year.unique()
```

Out[178…
```
array([2017, 2018, 2019, 2020], dtype=int64)
```

In [179…
```
station_day_3yrs = station_day_3.copy()
```

In [180…
```
station_day_3yrs.head()
```

Out[180…

	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	…	Xylene	AQI	AQI_Bucket	Year	State	la
0	2017-11-24	0.071341	0.115741	0.003702	0.046067	0.026517	0.029077	0.000569	0.054948	0.113449	…	0.000587	0.060755	Moderate	2017	Andhra Pradesh	14.7
1	2017-11-25	0.081382	0.124491	0.003043	0.045733	0.025832	0.025568	0.000683	0.077847	0.131964	…	0.000352	0.086232	Moderate	2017	Andhra Pradesh	14.7
2	2017-11-26	0.078302	0.129051	0.002660	0.058008	0.031756	0.024517	0.000796	0.137753	0.121943	…	0.000470	0.092602	Moderate	2017	Andhra Pradesh	14.7
3	2017-11-27	0.088742	0.135311	0.014022	0.068833	0.046554	0.030796	0.000626	0.171642	0.116097	…	0.000704	0.093092	Moderate	2017	Andhra Pradesh	14.7
4	2017-11-28	0.064161	0.104081	0.005426	0.062628	0.036375	0.027239	0.000512	0.097066	0.143480	…	0.000411	0.088192	Moderate	2017	Andhra Pradesh	14.7

5 rows × 22 columns

In [181…
```
station_day_3yrs.Year.unique()
```

Out[181…
```
array([2017, 2018, 2019, 2020, 2016, 2015], dtype=int64)
```

In [182…
```
station_day_3yrs.shape
```

Out[182…
```
(108035, 22)
```

Drop rows with year values 2015 and 2016 from the copy of primary dataset as these are not present in the power generation dataset and hence not useful for analysis.

In [183…
```
station_day_3yrs.drop(index=station_day_3yrs[station_day_3yrs.Year == 2015].index, inplace=True)
```

In [184…
```
station_day_3yrs.drop(index=station_day_3yrs[station_day_3yrs.Year == 2016].index, inplace=True)
```

In [185…
```
station_day_3yrs.Year.unique()
```

Out[185…
```
array([2017, 2018, 2019, 2020], dtype=int64)
```

In [186…
```
station_day_3yrs.shape
```

Out[186…
```
(90853, 22)
```

In [187…
```
station_day_3yrs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 90853 entries, 0 to 108034
Data columns (total 22 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Date        90853 non-null  datetime64[ns]
 1   PM2.5       90853 non-null  float64
 2   PM10        90853 non-null  float64
 3   NO          90853 non-null  float64
 4   NO2         90853 non-null  float64
 5   NOx         90853 non-null  float64
 6   NH3         90853 non-null  float64
 7   CO          90853 non-null  float64
 8   SO2         90853 non-null  float64
 9   O3          90853 non-null  float64
 10  Benzene     90853 non-null  float64
 11  Toluene     90853 non-null  float64
 12  Xylene      90853 non-null  float64
 13  AQI         90853 non-null  float64
 14  AQI_Bucket  90853 non-null  object
 15  Year        90853 non-null  int64
 16  State       90853 non-null  object
 17  latitude    90853 non-null  float64
 18  longitude   90853 non-null  float64
 19  Region      90853 non-null  object
 20  Reg_Lat     90853 non-null  float64
 21  Reg_Lon     90853 non-null  float64
dtypes: datetime64[ns](1), float64(17), int64(1), object(3)
memory usage: 15.9+ MB
```

In [525…
```
#station_day_3yrs.to_csv('station_day_3yrs.csv')
```

All the data processig is over and only visual analysis part remains, which will be done in Tableau.

END