```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#plt.use('Agg')
%matplotlib inline
z = np.random.rand(21315)


data = pd.read_csv(r"C:\Users\kadam\OneDrive\Desktop\jyp_python\miniproject\archive (26)\city_day.csv")
data.to_pickle("data.pkl")
```

Data Processing

```python
data.shape
```

```
(29531, 16)
```

```python
data.head()
```

```
        City        Date  PM2.5  PM10     NO    NO2    NOx  NH3     CO
SO2  \
0  Ahmedabad  2015-01-01    NaN   NaN   0.92  18.22  17.15  NaN   0.92
27.64
1  Ahmedabad  2015-01-02    NaN   NaN   0.97  15.69  16.46  NaN   0.97
24.55
2  Ahmedabad  2015-01-03    NaN   NaN  17.40  19.30  29.70  NaN  17.40
29.07
3  Ahmedabad  2015-01-04    NaN   NaN   1.70  18.48  17.97  NaN   1.70
18.59
4  Ahmedabad  2015-01-05    NaN   NaN  22.10  21.42  37.76  NaN  22.10
39.33

       O3  Benzene  Toluene  Xylene  AQI AQI_Bucket
0  133.36     0.00     0.02    0.00  NaN        NaN
1   34.06     3.68     5.50    3.77  NaN        NaN
2   30.70     6.80    16.40    2.25  NaN        NaN
3   36.08     4.43    10.14    1.00  NaN        NaN
4   39.31     7.01    18.89    2.78  NaN        NaN
```

```python
data.tail()
```

```
                City        Date  PM2.5   PM10    NO    NO2    NOx
NH3  \
29526  Visakhapatnam  2020-06-27  15.02  50.94  7.68  25.06  19.54
12.47
29527  Visakhapatnam  2020-06-28  24.38  74.09  3.42  26.06  16.53
11.99
29528  Visakhapatnam  2020-06-29  22.91  65.73  3.45  29.53  18.33
```

```
10.71
29529  Visakhapatnam  2020-06-30  16.64  49.97  4.05  29.26  18.80
10.03
29530  Visakhapatnam  2020-07-01  15.00  66.00  0.40  26.85  14.05
5.20

         CO    SO2     O3  Benzene  Toluene  Xylene   AQI
AQI_Bucket
29526  0.47   8.55  23.30     2.24    12.07    0.73  41.0
Good
29527  0.52  12.72  30.14     0.74     2.21    0.38  70.0
Satisfactory
29528  0.48   8.42  30.96     0.01     0.01    0.00  68.0
Satisfactory
29529  0.52   9.84  28.30     0.00     0.00    0.00  54.0
Satisfactory
29530  0.59   2.10  17.05      NaN      NaN     NaN  50.0
Good
```

```
data.dtypes
```

```
City              object
Date              object
PM2.5            float64
PM10             float64
NO               float64
NO2              float64
NOx              float64
NH3              float64
CO               float64
SO2              float64
O3               float64
Benzene          float64
Toluene          float64
Xylene           float64
AQI              float64
AQI_Bucket        object
dtype: object
```

```
data.columns
```

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

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
```

```
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   City        29531 non-null  object
 1   Date        29531 non-null  object
 2   PM2.5       24933 non-null  float64
 3   PM10        18391 non-null  float64
 4   NO          25949 non-null  float64
 5   NO2         25946 non-null  float64
 6   NOx         25346 non-null  float64
 7   NH3         19203 non-null  float64
 8   CO          27472 non-null  float64
 9   SO2         25677 non-null  float64
 10  O3          25509 non-null  float64
 11  Benzene     23908 non-null  float64
 12  Toluene     21490 non-null  float64
 13  Xylene      11422 non-null  float64
 14  AQI         24850 non-null  float64
 15  AQI_Bucket  24850 non-null  object
dtypes: float64(13), object(3)
memory usage: 3.6+ MB
```

dataset have null values.

It doesn't have invalid datatypes.

```
data_null = np.where(data.isnull()==True)
data_null
```

```
(array([    0,     0,     0, ..., 29530, 29530, 29530]),
 array([ 2,  3,  7, ..., 11, 12, 13]))
```

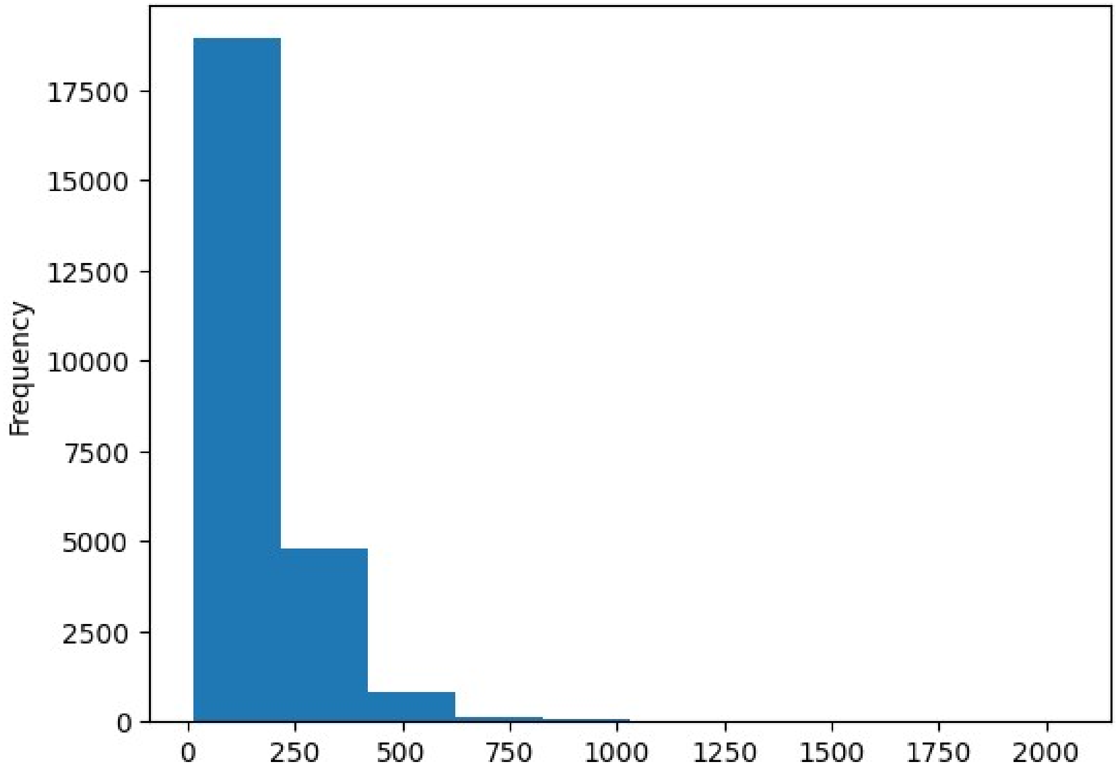
```
data.describe()
```

```
             PM2.5          PM10            NO           NO2
NOx  \
count  24933.000000  18391.000000  25949.000000  25946.000000
25346.000000
mean      67.450578    118.127103     17.574730     28.560659
32.309123
std       64.661449     90.605110     22.785846     24.474746
31.646011
min        0.040000      0.010000      0.020000      0.010000
0.000000
25%       28.820000     56.255000      5.630000     11.750000
12.820000
50%       48.570000     95.680000      9.890000     21.690000
23.520000
75%       80.590000    149.745000     19.950000     37.620000
40.127500
max      949.990000   1000.000000    390.680000    362.210000
467.630000
```

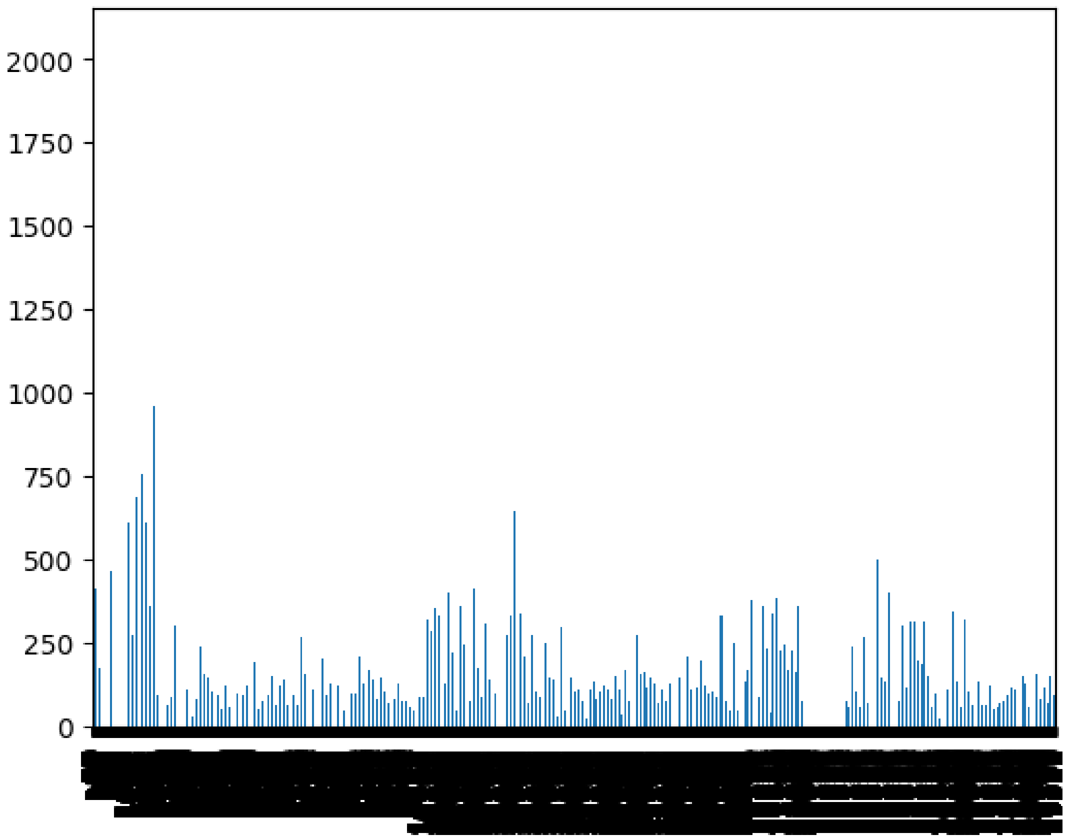
```
                NH3           CO          SO2           O3
Benzene  \
count  19203.000000  27472.000000  25677.000000  25509.000000
23908.000000
mean      23.483476      2.248598     14.531977     34.491430
3.280840
std       25.684275      6.962884     18.133775     21.694928
15.811136
min        0.010000      0.000000      0.010000      0.010000
0.000000
25%        8.580000      0.510000      5.670000     18.860000
0.120000
50%       15.850000      0.890000      9.160000     30.840000
1.070000
75%       30.020000      1.450000     15.220000     45.570000
3.080000
max      352.890000    175.810000    193.860000    257.730000
455.030000

            Toluene        Xylene           AQI
count  21490.000000  11422.000000  24850.000000
mean       8.700972      3.070128    166.463581
std       19.969164      6.323247    140.696585
min        0.000000      0.000000     13.000000
25%        0.600000      0.140000     81.000000
50%        2.970000      0.980000    118.000000
75%        9.150000      3.350000    208.000000
max      454.850000    170.370000   2049.000000
```

Data Cleaning

```
data['AQI'].plot(kind='hist')
plt.show()
```

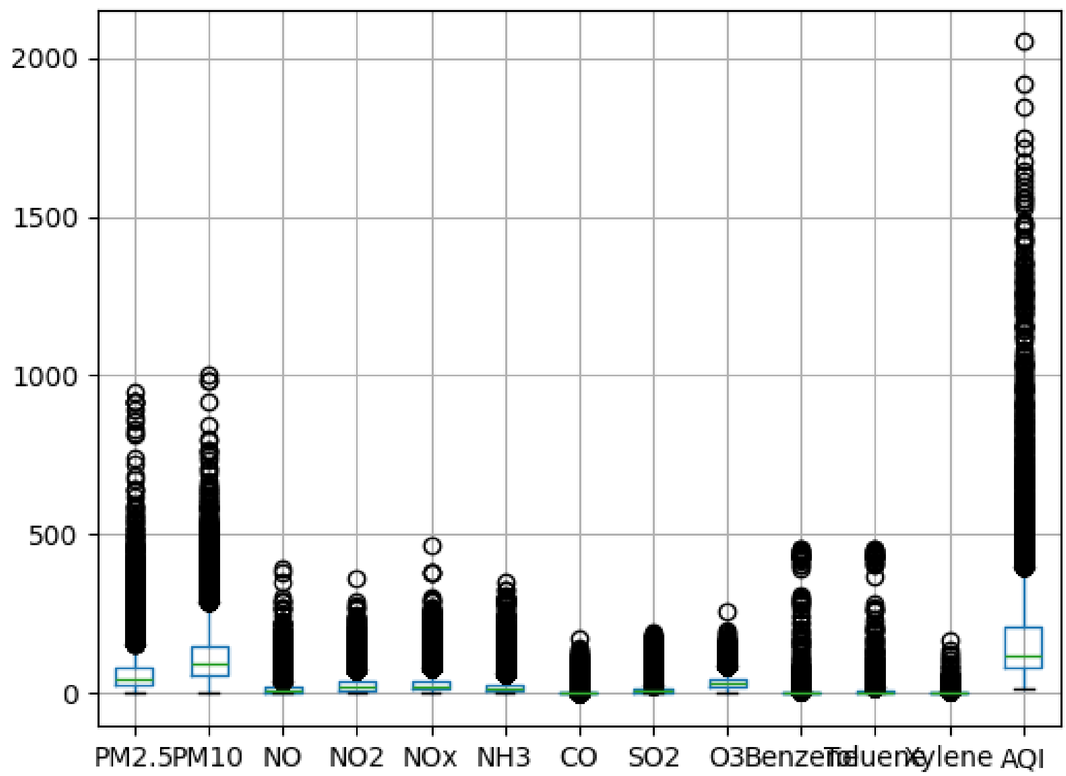



```
data['AQI'].plot(kind='bar')
plt.show()
```



```
data.boxplot()
```

```
<Axes: >
```




```
data['PM2.5'].fillna(data['PM2.5'].mean(), inplace=True)
#df['column_name'].fillna(df['column_name'].mean(), inplace=True)

C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\351580854.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['PM2.5'].fillna(data['PM2.5'].mean(), inplace=True)

data

                City        Date      PM2.5   PM10     NO    NO2
NOx  \
0          Ahmedabad  2015-01-01  67.450578    NaN   0.92  18.22
17.15
1          Ahmedabad  2015-01-02  67.450578    NaN   0.97  15.69
16.46
```

```
2           Ahmedabad  2015-01-03  67.450578    NaN  17.40  19.30
29.70
3           Ahmedabad  2015-01-04  67.450578    NaN   1.70  18.48
17.97
4           Ahmedabad  2015-01-05  67.450578    NaN  22.10  21.42
37.76
...               ...         ...        ...    ...    ...    ...    ..
.
29526  Visakhapatnam  2020-06-27  15.020000  50.94   7.68  25.06
19.54
29527  Visakhapatnam  2020-06-28  24.380000  74.09   3.42  26.06
16.53
29528  Visakhapatnam  2020-06-29  22.910000  65.73   3.45  29.53
18.33
29529  Visakhapatnam  2020-06-30  16.640000  49.97   4.05  29.26
18.80
29530  Visakhapatnam  2020-07-01  15.000000  66.00   0.40  26.85
14.05

         NH3     CO    SO2      O3  Benzene  Toluene  Xylene   AQI  \
0        NaN   0.92  27.64  133.36     0.00     0.02    0.00   NaN
1        NaN   0.97  24.55   34.06     3.68     5.50    3.77   NaN
2        NaN  17.40  29.07   30.70     6.80    16.40    2.25   NaN
3        NaN   1.70  18.59   36.08     4.43    10.14    1.00   NaN
4        NaN  22.10  39.33   39.31     7.01    18.89    2.78   NaN
...      ...    ...    ...     ...      ...      ...     ...   ...
29526  12.47   0.47   8.55   23.30     2.24    12.07    0.73  41.0
29527  11.99   0.52  12.72   30.14     0.74     2.21    0.38  70.0
29528  10.71   0.48   8.42   30.96     0.01     0.01    0.00  68.0
29529  10.03   0.52   9.84   28.30     0.00     0.00    0.00  54.0
29530   5.20   0.59   2.10   17.05      NaN      NaN     NaN  50.0

         AQI_Bucket
0               NaN
1               NaN
2               NaN
3               NaN
4               NaN
...             ...
29526          Good
29527  Satisfactory
29528  Satisfactory
29529  Satisfactory
29530          Good

[29531 rows x 16 columns]

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   City        29531 non-null  object
 1   Date        29531 non-null  object
 2   PM2.5       29531 non-null  float64
 3   PM10        18391 non-null  float64
 4   NO          25949 non-null  float64
 5   NO2         25946 non-null  float64
 6   NOx         25346 non-null  float64
 7   NH3         19203 non-null  float64
 8   CO          27472 non-null  float64
 9   SO2         25677 non-null  float64
 10  O3          25509 non-null  float64
 11  Benzene     23908 non-null  float64
 12  Toluene     21490 non-null  float64
 13  Xylene      11422 non-null  float64
 14  AQI         24850 non-null  float64
 15  AQI_Bucket  24850 non-null  object
dtypes: float64(13), object(3)
memory usage: 3.6+ MB

data['PM2.5'].fillna(data['PM2.5'].mean(), inplace=True)

C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\935467664.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['PM2.5'].fillna(data['PM2.5'].mean(), inplace=True)

data['PM10'].fillna(data['PM10'].mean(), inplace=True)

C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\715378189.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
```

```
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['PM10'].fillna(data['PM10'].mean(), inplace=True)
```

```
data['NO'].fillna(data['NO'].mean(), inplace=True)
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\3552883858.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['NO'].fillna(data['NO'].mean(), inplace=True)
```

```
data['NO2'].fillna(data['NO2'].mean(), inplace=True)
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\2390402205.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['NO2'].fillna(data['NO2'].mean(), inplace=True)
```

```
data['NOx'].fillna(data['NOx'].mean(), inplace=True)
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\2133220639.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
```

```
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['NOx'].fillna(data['NOx'].mean(), inplace=True)
```

```python
data['NH3'].fillna(data['NH3'].mean(), inplace=True)
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\2854111976.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['NH3'].fillna(data['NH3'].mean(), inplace=True)
```

```python
data['CO'].fillna(data['CO'].mean(), inplace=True)
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\1161907318.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['CO'].fillna(data['CO'].mean(), inplace=True)
```

```python
data['SO2'].fillna(data['SO2'].mean(), inplace=True)
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\3344645610.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
```

```
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['SO2'].fillna(data['SO2'].mean(), inplace=True)
```

```python
data['O3'].fillna(data['O3'].mean(), inplace=True)
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\2730838331.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['O3'].fillna(data['O3'].mean(), inplace=True)
```

```python
data['Benzene'].fillna(data['Benzene'].mean(), inplace=True)
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\4277266277.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['Benzene'].fillna(data['Benzene'].mean(), inplace=True)
```

```python
data['Toluene'].fillna(data['Toluene'].mean(), inplace=True)
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\4248952095.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
```

```
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['Toluene'].fillna(data['Toluene'].mean(), inplace=True)
```

```python
data['Xylene'].fillna(data['Xylene'].mean(), inplace=True)
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\2593633129.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['Xylene'].fillna(data['Xylene'].mean(), inplace=True)
```

```python
data['AQI'].fillna(data['AQI'].mean(), inplace=True)
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\2966707500.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data['AQI'].fillna(data['AQI'].mean(), inplace=True)
```

```python
data.describe()
```

```
              PM2.5          PM10            NO           NO2
NOx  \
count  29531.000000  29531.000000  29531.00000  29531.000000
29531.000000
mean      67.450578    118.127103     17.57473     28.560659
32.309123
std       59.414476     71.500953     21.35922     22.941051
29.317936
```

```
min         0.040000      0.010000      0.02000      0.010000
0.000000
25%        32.150000     79.315000      6.21000     12.980000
14.670000
50%        58.030000    118.127103     11.53000     25.240000
27.550000
75%        72.450000    118.127103     17.57473     34.665000
36.015000
max       949.990000   1000.000000    390.68000    362.210000
467.630000

                NH3            CO           SO2            O3
Benzene  \
count  29531.000000  29531.000000  29531.000000  29531.000000
29531.000000
mean      23.483476      2.248598     14.531977     34.491430
3.280840
std       20.711370      6.715753     16.909088     20.163443
14.226364
min        0.010000      0.000000      0.010000      0.010000
0.000000
25%       12.040000      0.540000      6.090000     20.740000
0.240000
50%       23.483476      0.950000     10.480000     34.491430
1.840000
75%       23.483476      1.710000     14.531977     42.730000
3.280840
max      352.890000    175.810000    193.860000    257.730000
455.030000

            Toluene        Xylene           AQI
count  29531.000000  29531.000000  29531.000000
mean       8.700972      3.070128    166.463581
std       17.034769      3.932426    129.064348
min        0.000000      0.000000     13.000000
25%        1.280000      2.000000     88.000000
50%        6.930000      3.070128    138.000000
75%        8.700972      3.070128    179.000000
max      454.850000    170.370000   2049.000000
```

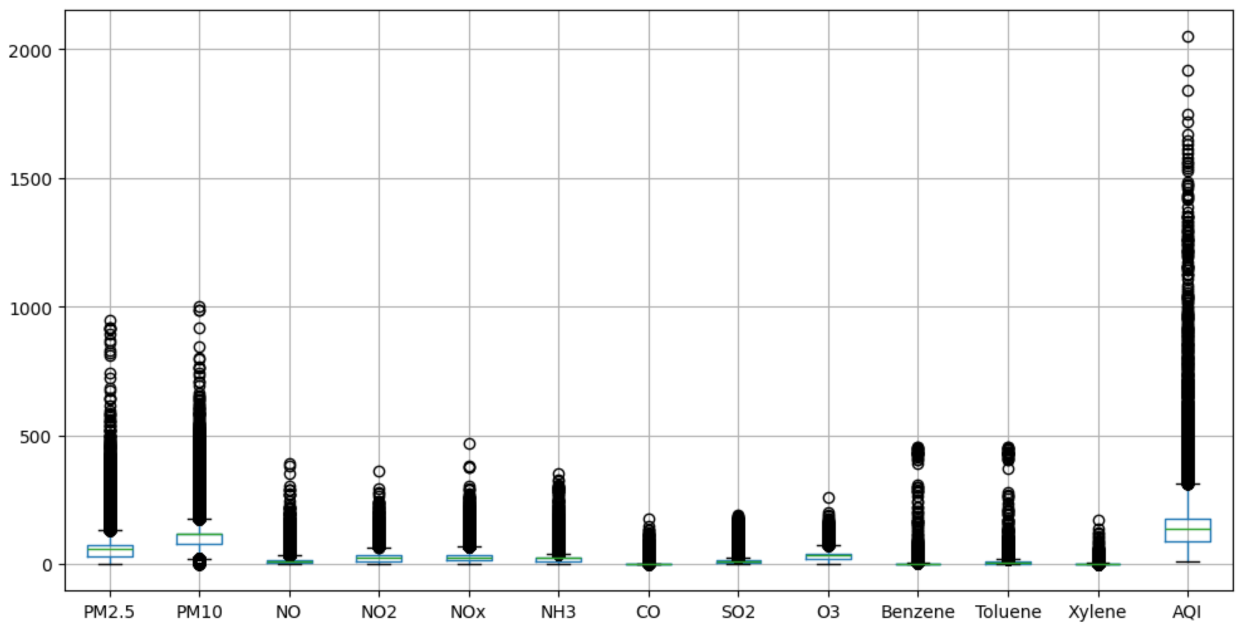
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   City        29531 non-null  object
 1   Date        29531 non-null  object
 2   PM2.5       29531 non-null  float64
```

```
 3   PM10         29531 non-null  float64
 4   NO           29531 non-null  float64
 5   NO2          29531 non-null  float64
 6   NOx          29531 non-null  float64
 7   NH3          29531 non-null  float64
 8   CO           29531 non-null  float64
 9   SO2          29531 non-null  float64
 10  O3           29531 non-null  float64
 11  Benzene      29531 non-null  float64
 12  Toluene      29531 non-null  float64
 13  Xylene       29531 non-null  float64
 14  AQI          29531 non-null  float64
 15  AQI_Bucket  24850 non-null  object
dtypes: float64(13), object(3)
memory usage: 3.6+ MB
```

```
plt.figure(figsize=(12,6))
data.boxplot()
```

```
<Axes: >
```



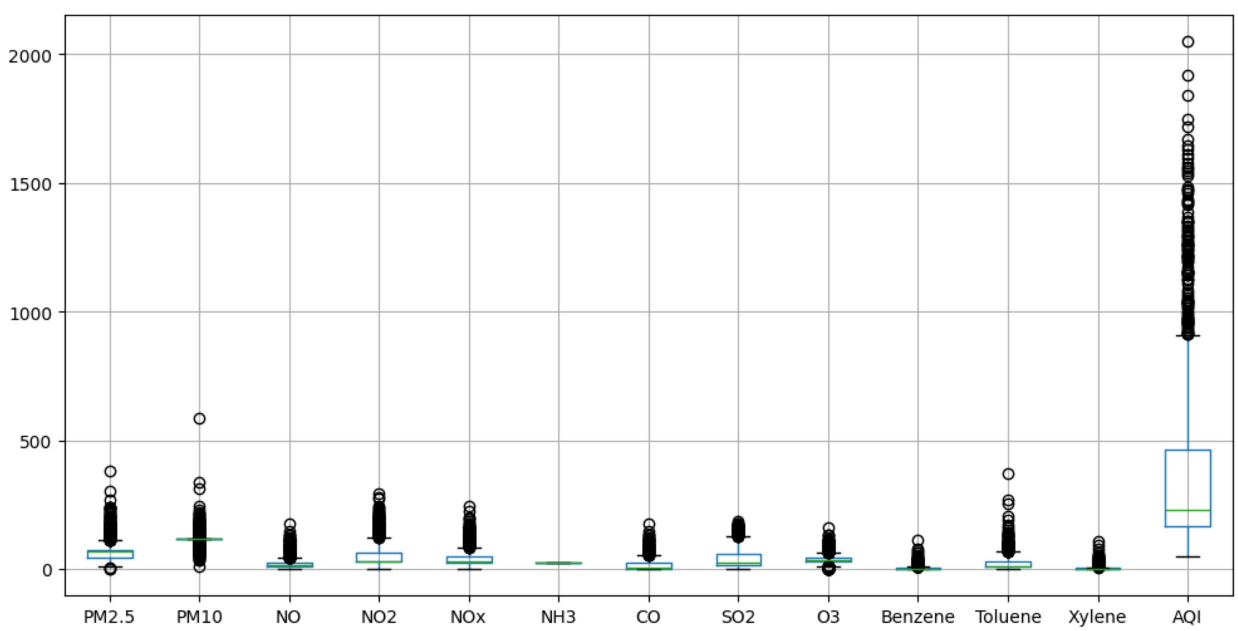

```
city = data.City.value_counts()
city
```

```
City
Ahmedabad                     2009
Bengaluru                     2009
Chennai                       2009
Mumbai                        2009
Lucknow                       2009
```

```
Delhi                 2009
Hyderabad             2006
Patna                 1858
Gurugram              1679
Visakhapatnam         1462
Amritsar              1221
Jorapokhar            1169
Jaipur                1114
Thiruvananthapuram    1112
Amaravati              951
Brajrajnagar           938
Talcher                925
Kolkata                814
Guwahati               502
Coimbatore             386
Shillong               310
Chandigarh             304
Bhopal                 289
Kochi                  162
Ernakulam              162
Aizawl                 113
Name: count, dtype: int64
```

```
plt.figure(figsize=(12,6))
datahm = data[(data['City']=='Ahmedabad')]
datahm.boxplot()
```

```
<Axes: >
```

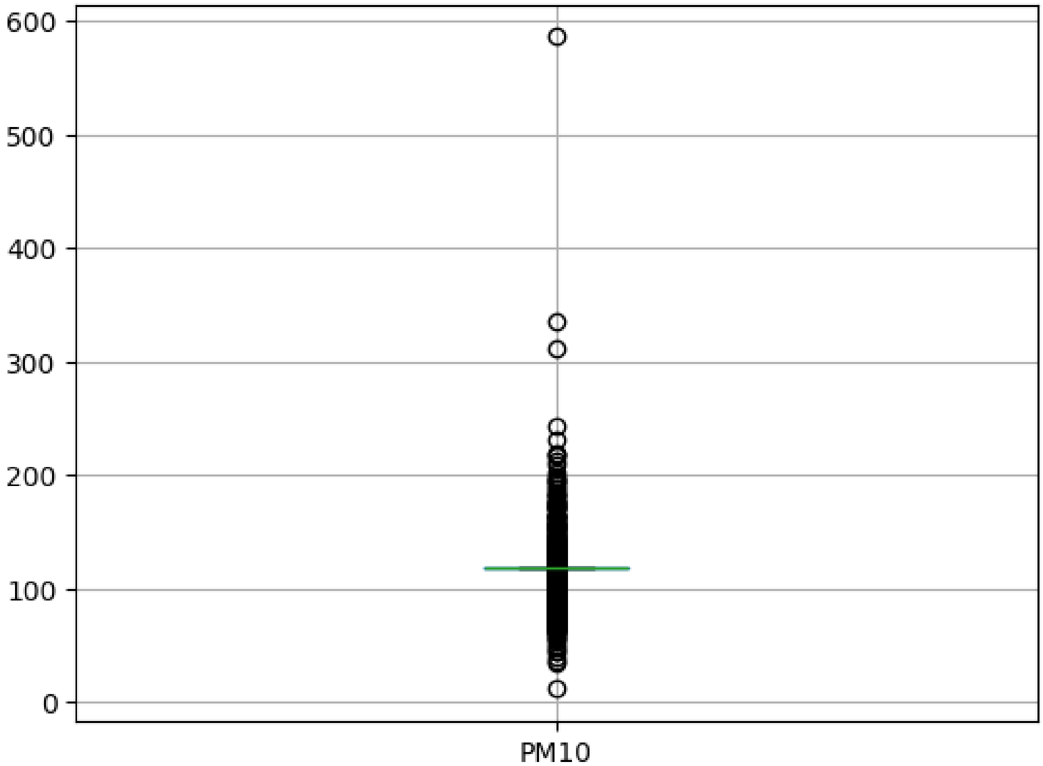



```
datahm.describe()
```

```
             PM2.5         PM10           NO          NO2          NOx  \
count  2009.000000  2009.000000  2009.000000  2009.000000  2009.000000   
mean     67.728234   117.409318    20.956815    49.805675    42.914773   
std      32.739185    20.608768    18.030998    41.889674    29.023976   
min       3.040000    11.500000     0.060000     0.080000     0.000000   
25%      46.910000   118.127103    10.380000    28.560659    27.840000   
50%      67.450578   118.127103    17.574730    28.560659    32.309123   
75%      73.070000   118.127103    23.750000    66.430000    51.030000   
max     381.690000   586.270000   175.810000   292.020000   246.030000   

                NH3           CO          SO2           O3      Benzene  \
count  2.009000e+03  2009.000000  2009.000000  2009.000000  2009.000000   
mean   2.348348e+01    16.147420    42.281148    37.565152     4.901003   
std    7.107196e-15    20.258113    37.926831    18.464239     6.953368   
min    2.348348e+01     0.060000     0.520000     0.380000     0.000000   
25%    2.348348e+01     2.248598    14.531977    32.100000     1.820000   
50%    2.348348e+01     8.510000    23.810000    34.491430     3.280840   
75%    2.348348e+01    23.750000    60.680000    45.650000     4.720000   
max    2.348348e+01   175.810000   186.080000   162.430000   115.140000   

           Toluene       Xylene          AQI  
count  2009.000000  2009.000000  2009.000000  
mean     23.163071     3.964491   356.144807  
std      26.787328     6.547374   287.617151  
min       0.000000     0.000000    48.000000  
25%       8.700972     0.660000   166.463581  
50%      11.320000     3.070128   229.000000  
75%      32.330000     3.850000   465.000000  
max     371.650000   109.230000  2049.000000  
```

```
datahm.boxplot(column='PM10')
```

```
<Axes: >
```




```
col =
['PM2.5','PM10','NO','NO2','NOx','NH3','CO','SO2','O3','Benzene','Tolu
ene','Xylene','AQI']

Q3 = datahm[col].quantile(0.75)
Q1 = datahm[col].quantile(0.25)

Q1,Q3
```

```
(PM2.5       46.910000
 PM10       118.127103
 NO          10.380000
 NO2         28.560659
 NOx         27.840000
 NH3         23.483476
 CO           2.248598
 SO2         14.531977
 O3          32.100000
 Benzene      1.820000
 Toluene      8.700972
 Xylene       0.660000
 AQI        166.463581
 Name: 0.25, dtype: float64,
 PM2.5       73.070000
 PM10       118.127103
 NO          23.750000
```

```
 NO2          66.430000
 NOx          51.030000
 NH3          23.483476
 CO           23.750000
 SO2          60.680000
 O3           45.650000
 Benzene       4.720000
 Toluene      32.330000
 Xylene        3.850000
 AQI         465.000000
 Name: 0.75, dtype: float64)
```

```
IQR = Q3 - Q1
IQR
```

```
PM2.5        26.160000
PM10          0.000000
NO           13.370000
NO2          37.869341
NOx          23.190000
NH3           0.000000
CO           21.501402
SO2          46.148023
O3           13.550000
Benzene       2.900000
Toluene      23.629028
Xylene        3.190000
AQI         298.536419
dtype: float64
```

```
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit
```

```
(PM2.5         7.670000
 PM10        118.127103
 NO           -9.675000
 NO2         -28.243352
 NOx          -6.945000
 NH3          23.483476
 CO          -30.003504
 SO2         -54.690057
 O3           11.775000
 Benzene      -2.530000
 Toluene     -26.742570
 Xylene       -4.125000
 AQI        -281.341046
 dtype: float64,
 PM2.5       112.310000
 PM10        118.127103
```

```
 NO           43.805000
 NO2         123.234011
 NOx          85.815000
 NH3          23.483476
 CO           56.002103
 SO2         129.902034
 O3           65.975000
 Benzene       9.070000
 Toluene      67.773542
 Xylene        8.635000
 AQI         912.804628
 dtype: float64)
```

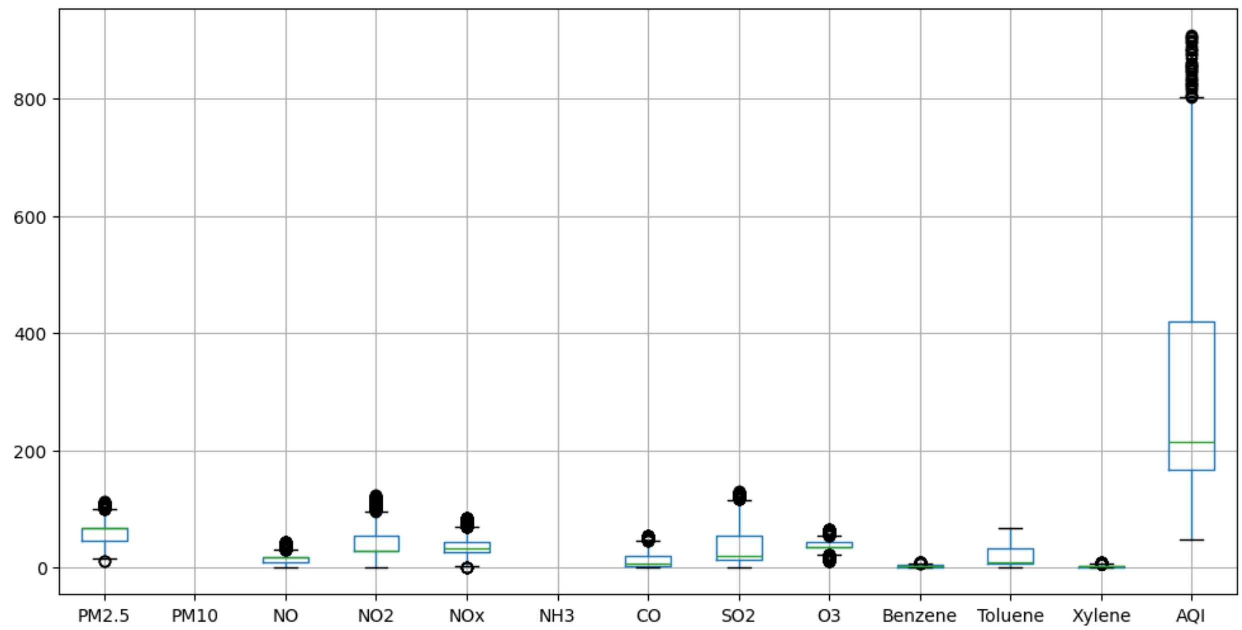
```
datahm[(datahm[col]<lower_limit)|(datahm[col]>upper_limit)]
```

```
     City Date  PM2.5    PM10  NO  NO2  NOx  NH3  CO  SO2      O3  
Benzene  \
0     NaN  NaN    NaN     NaN NaN  NaN  NaN  NaN NaN  NaN  133.36
NaN
1     NaN  NaN    NaN     NaN NaN  NaN  NaN  NaN NaN  NaN     NaN
NaN
2     NaN  NaN    NaN     NaN NaN  NaN  NaN  NaN NaN  NaN     NaN
NaN
3     NaN  NaN    NaN     NaN NaN  NaN  NaN  NaN NaN  NaN     NaN
NaN
4     NaN  NaN    NaN     NaN NaN  NaN  NaN  NaN NaN  NaN     NaN
NaN
...   ...  ...    ...     ...  ..  ...  ...  ...  ..  ...     ...
...
2004  NaN  NaN    NaN  118.67 NaN  NaN  NaN  NaN NaN  NaN   68.05
NaN
2005  NaN  NaN    NaN     NaN NaN  NaN  NaN  NaN NaN  NaN     NaN
NaN
2006  NaN  NaN    NaN  127.98 NaN  NaN  NaN  NaN NaN  NaN     NaN
NaN
2007  NaN  NaN    NaN  121.10 NaN  NaN  NaN  NaN NaN  NaN     NaN
NaN
2008  NaN  NaN    NaN     NaN NaN  NaN  NaN  NaN NaN  NaN    9.69
NaN

      Toluene  Xylene  AQI AQI_Bucket
0         NaN     NaN  NaN        NaN
1         NaN     NaN  NaN        NaN
2         NaN     NaN  NaN        NaN
3         NaN     NaN  NaN        NaN
4         NaN     NaN  NaN        NaN
...       ...     ...  ...        ...
2004      NaN     NaN  NaN        NaN
2005      NaN     NaN  NaN        NaN
2006      NaN     NaN  NaN        NaN
```

```
2007      NaN     NaN  NaN        NaN
2008      NaN     NaN  NaN        NaN

[2009 rows x 16 columns]

datahmo = datahm[(datahm[col]>lower_limit) &
(datahm[col]<upper_limit)]
datahmo

     City Date      PM2.5  PM10     NO    NO2    NOx  NH3     CO
SO2  \
0     NaN  NaN  67.450578   NaN   0.92  18.22  17.15  NaN   0.92
27.64
1     NaN  NaN  67.450578   NaN   0.97  15.69  16.46  NaN   0.97
24.55
2     NaN  NaN  67.450578   NaN  17.40  19.30  29.70  NaN  17.40
29.07
3     NaN  NaN  67.450578   NaN   1.70  18.48  17.97  NaN   1.70
18.59
4     NaN  NaN  67.450578   NaN  22.10  21.42  37.76  NaN  22.10
39.33
...   ...  ...        ...   ...    ...    ...    ...  ...    ...    ..
.
2004  NaN  NaN  62.120000   NaN   9.18  56.35  19.86  NaN   0.49
12.44
2005  NaN  NaN  31.570000   NaN   6.37  23.99  16.40  NaN   0.52
11.01
2006  NaN  NaN  29.750000   NaN   9.06  25.15  18.92  NaN   0.67
12.10
2007  NaN  NaN  40.020000   NaN   7.09  58.92  33.41  NaN   0.73
16.39
2008  NaN  NaN  37.630000   NaN   4.42  35.04  20.17  NaN   0.28
14.40

         O3  Benzene  Toluene  Xylene         AQI AQI_Bucket
0       NaN     0.00     0.02    0.00  166.463581        NaN
1     34.06     3.68     5.50    3.77  166.463581        NaN
2     30.70     6.80    16.40    2.25  166.463581        NaN
3     36.08     4.43    10.14    1.00  166.463581        NaN
4     39.31     7.01    18.89    2.78  166.463581        NaN
...     ...      ...      ...     ...         ...        ...
2004    NaN     1.32    37.76    1.62   92.000000        NaN
2005  26.34     1.37    49.58    1.34   82.000000        NaN
2006  34.99     1.39    60.21    0.79   74.000000        NaN
2007  41.64     1.21    44.10    1.35   98.000000        NaN
2008    NaN     1.73    47.05    1.87  119.000000        NaN

[2009 rows x 16 columns]
```

```
plt.figure(figsize=(12,6))
datahmo.boxplot()
```

```
<Axes: >
```




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

```
City             0
Date             0
PM2.5            0
PM10             0
NO               0
NO2              0
NOx              0
NH3              0
CO               0
SO2              0
O3               0
Benzene          0
Toluene          0
Xylene           0
AQI              0
AQI_Bucket    4681
dtype: int64
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
```

```
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   City        29531 non-null  object 
 1   Date        29531 non-null  object 
 2   PM2.5       29531 non-null  float64
 3   PM10        29531 non-null  float64
 4   NO          29531 non-null  float64
 5   NO2         29531 non-null  float64
 6   NOx         29531 non-null  float64
 7   NH3         29531 non-null  float64
 8   CO          29531 non-null  float64
 9   SO2         29531 non-null  float64
 10  O3          29531 non-null  float64
 11  Benzene     29531 non-null  float64
 12  Toluene     29531 non-null  float64
 13  Xylene      29531 non-null  float64
 14  AQI         29531 non-null  float64
 15  AQI_Bucket  24850 non-null  object 
dtypes: float64(13), object(3)
memory usage: 3.6+ MB
```

```
datac = data.dropna()
```

```
datac.to_csv('clean_city.csv', index=False)
```

```
cleandt = pd.read_csv(r'clean_city.csv')
cleandt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24850 entries, 0 to 24849
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   City        24850 non-null  object 
 1   Date        24850 non-null  object 
 2   PM2.5       24850 non-null  float64
 3   PM10        24850 non-null  float64
 4   NO          24850 non-null  float64
 5   NO2         24850 non-null  float64
 6   NOx         24850 non-null  float64
 7   NH3         24850 non-null  float64
 8   CO          24850 non-null  float64
 9   SO2         24850 non-null  float64
 10  O3          24850 non-null  float64
 11  Benzene     24850 non-null  float64
 12  Toluene     24850 non-null  float64
 13  Xylene      24850 non-null  float64
 14  AQI         24850 non-null  float64
 15  AQI_Bucket  24850 non-null  object 
```

```
dtypes: float64(13), object(3)
memory usage: 3.0+ MB
```

```
print(data["AQI_Bucket"].value_counts().sum())
print(cleandt["AQI_Bucket"].value_counts().sum())
```

```
24850
24850
```

```
data['AQI_Bucket'].unique()
```

```
array([nan, 'Poor', 'Very Poor', 'Severe', 'Moderate', 'Satisfactory',
       'Good'], dtype=object)
```

```
data['AQI_Bucket'].nunique()
```

```
6
```

```
data['AQI_Bucket'].value_counts()
```

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

```
data = data.drop_duplicates()
```

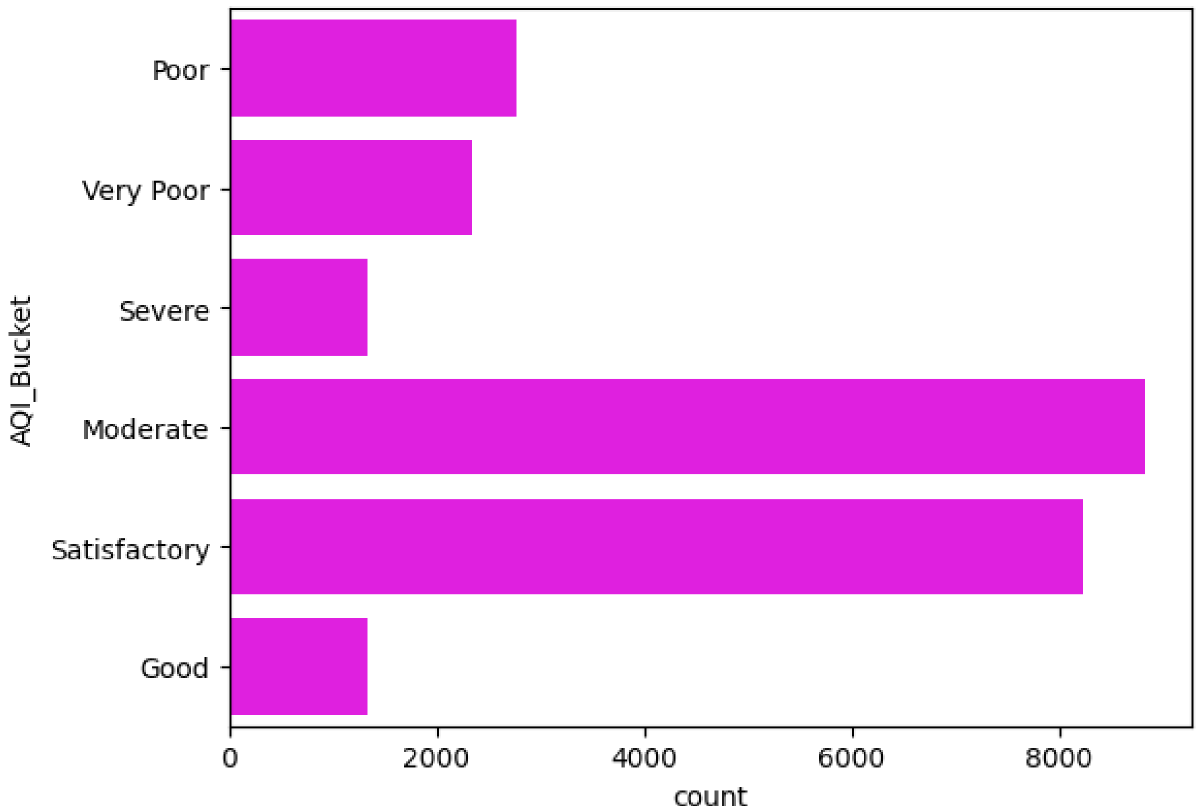
```
data.shape
```

```
(29531, 16)
```

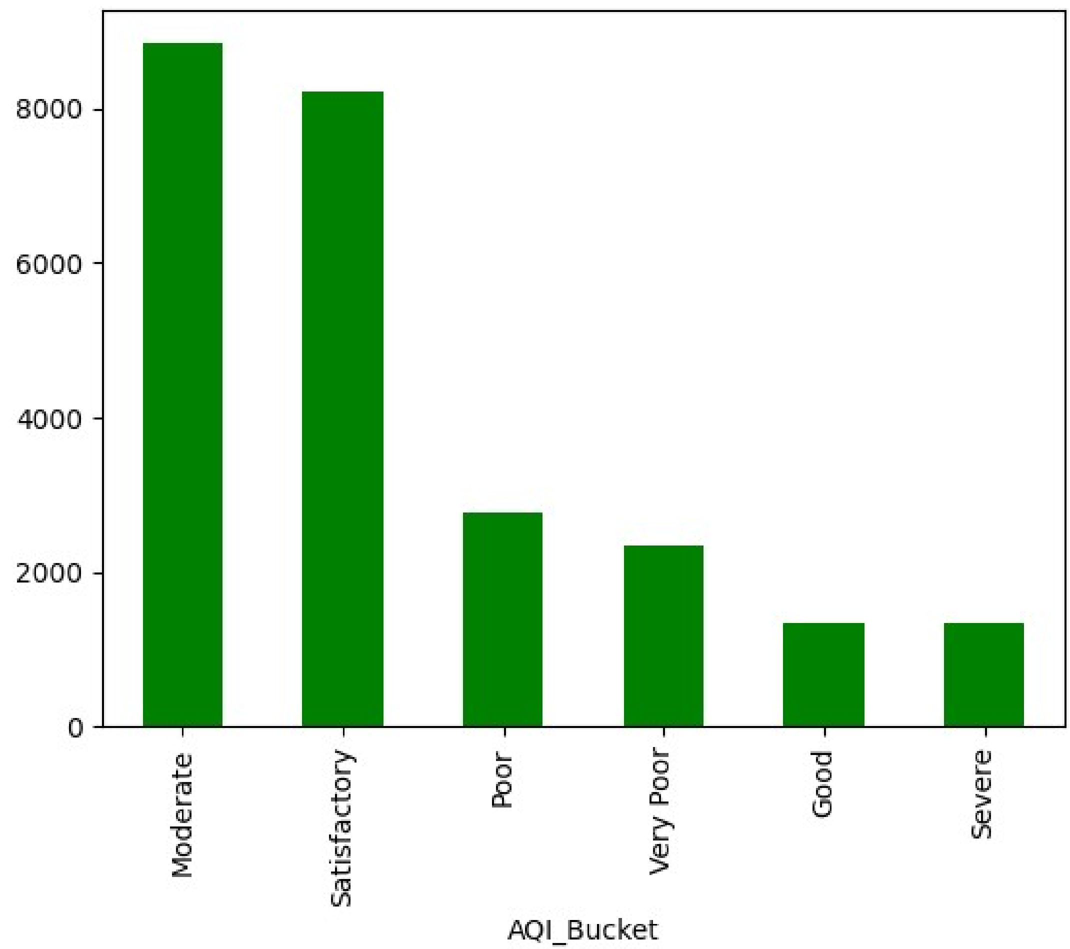
```
cleandt = cleandt.drop_duplicates()
cleandt.shape
```

```
(24850, 16)
```

```
sns.countplot(cleandt['AQI_Bucket'],color='magenta')
plt.show()
```

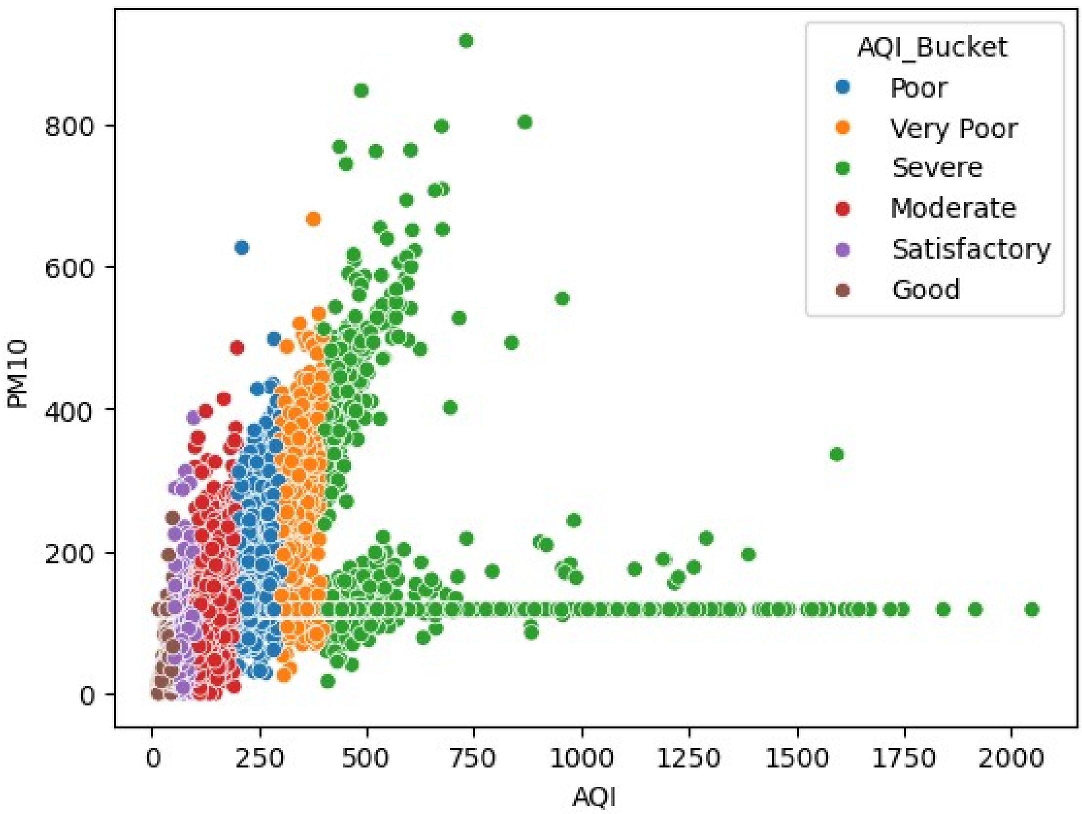



```
cleandt['AQI_Bucket'].value_counts().plot(kind='bar',color='green')
plt.show()
```



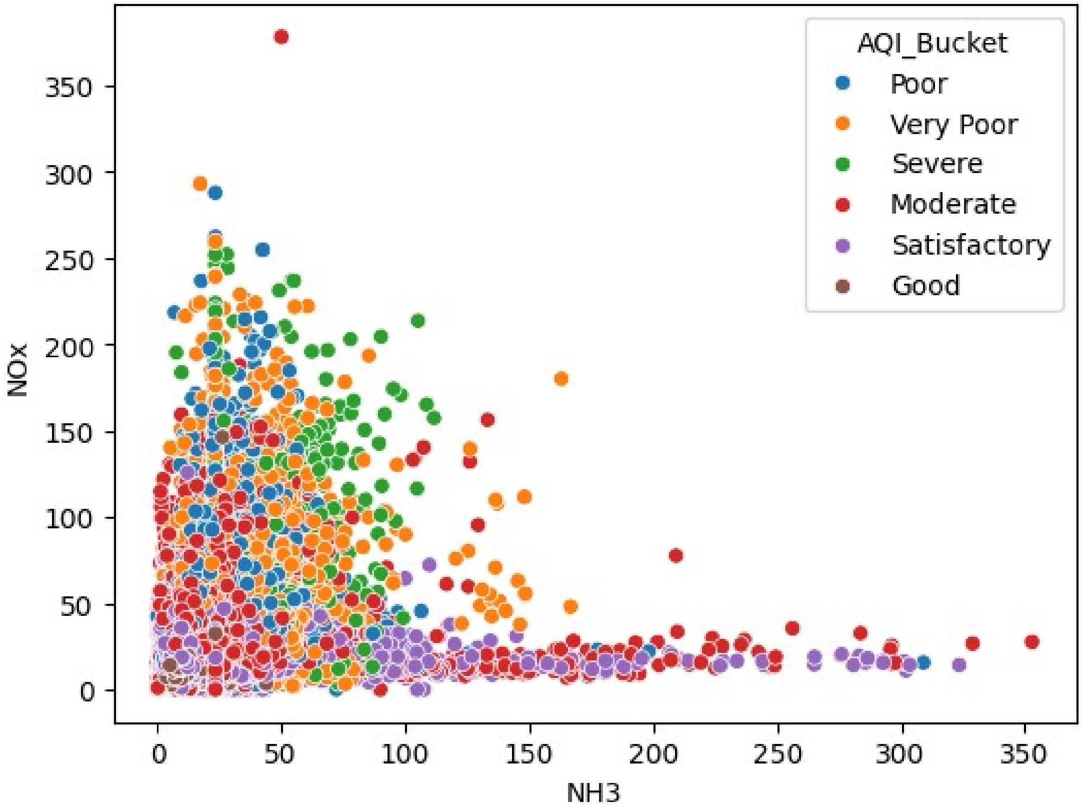

```
#how PM10 is affecting overall aqi
sns.scatterplot(x=cleandt['AQI'],
y=cleandt['PM10'],hue=cleandt['AQI_Bucket'])
```

```
<Axes: xlabel='AQI', ylabel='PM10'>
```




```
sns.scatterplot(x=cleandt['NH3'],
y=cleandt['NOx'],hue=cleandt['AQI_Bucket'])
```

```
<Axes: xlabel='NH3', ylabel='NOx'>
```

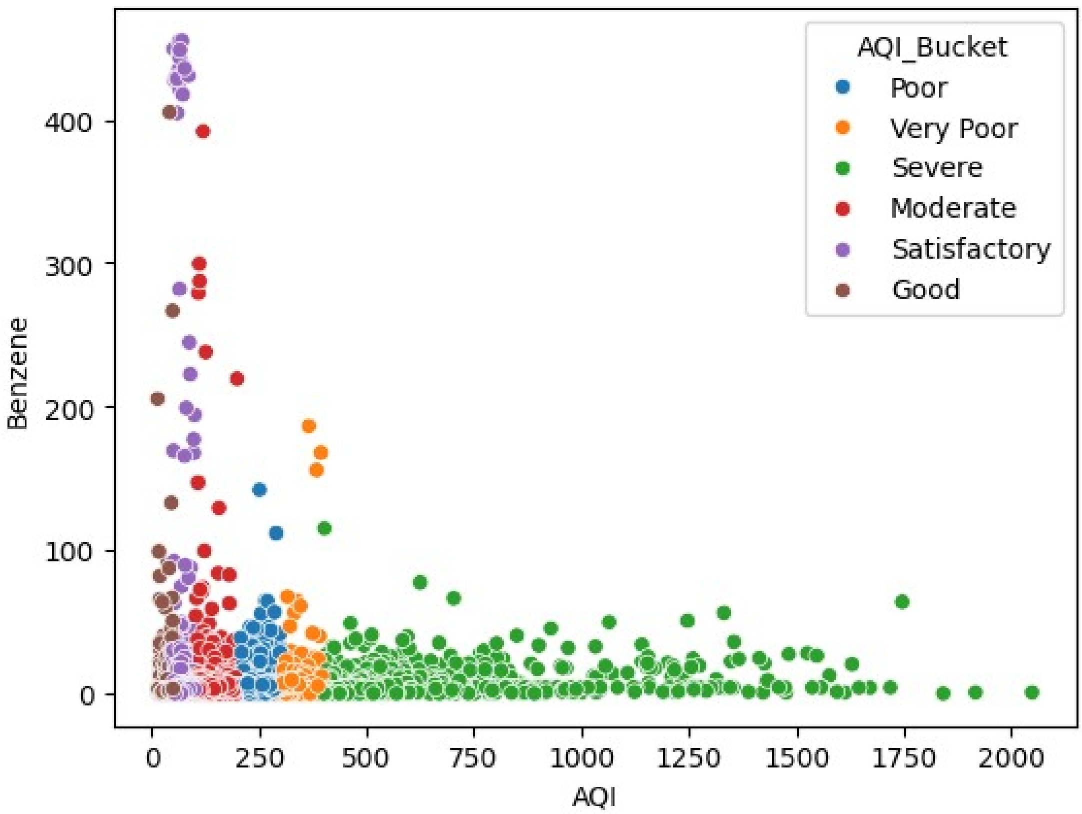


Combined effect of NOx and NH3 :

NOx(250) and NH3(100) this combination is really harmful

```
sns.scatterplot(x=cleandt['AQI'],y=cleandt['Benzene'],hue=cleandt['AQI_Bucket'])
```

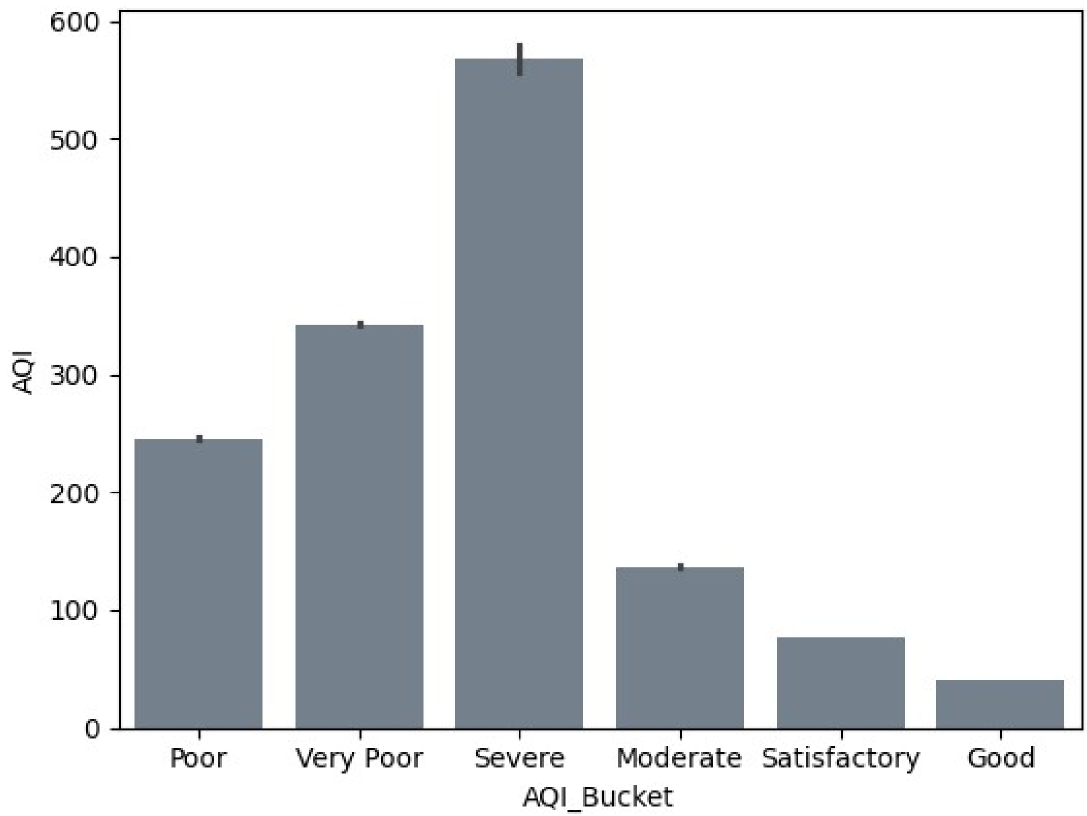
```
<Axes: xlabel='AQI', ylabel='Benzene'>
```



benzene doesn't impact thoroughly on air quality

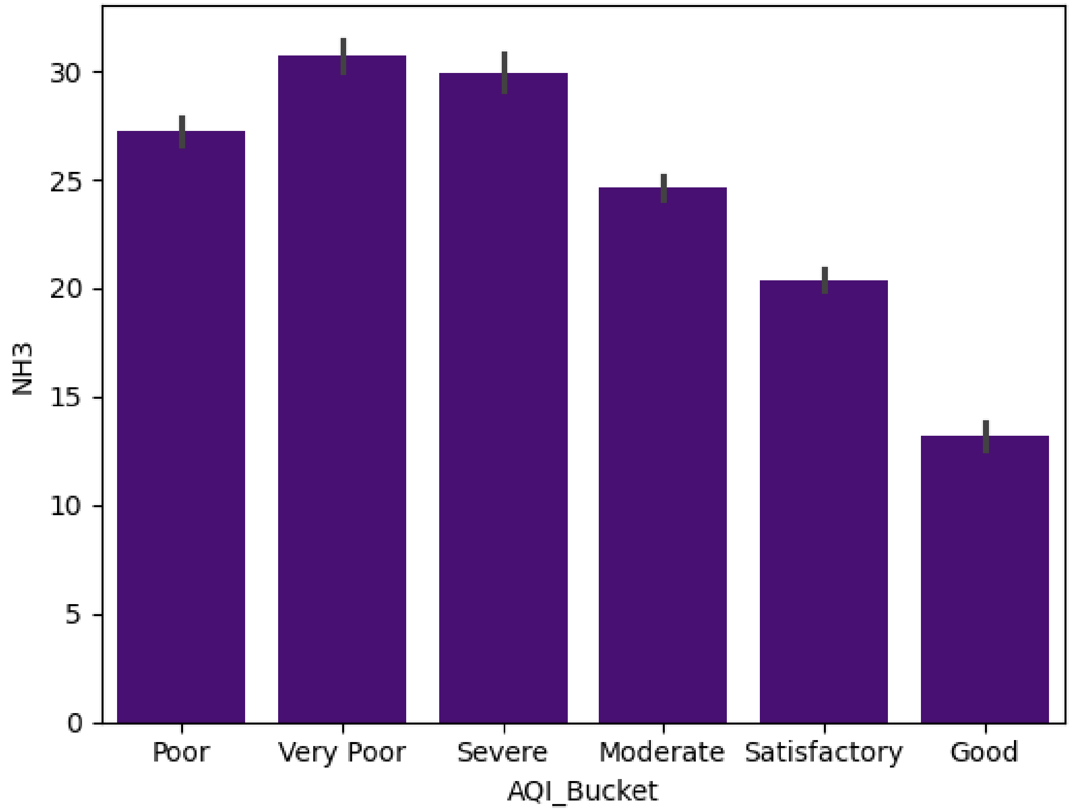
```
#categoriacl and numerical
sns.barplot(x=cleandt["AQI_Bucket"],y=cleandt["AQI"],color='slategray'
)
```

```
<Axes: xlabel='AQI_Bucket', ylabel='AQI'>
```




```
sns.barplot(x=cleandt["AQI_Bucket"],y=cleandt["NH3"],color='indigo')
```

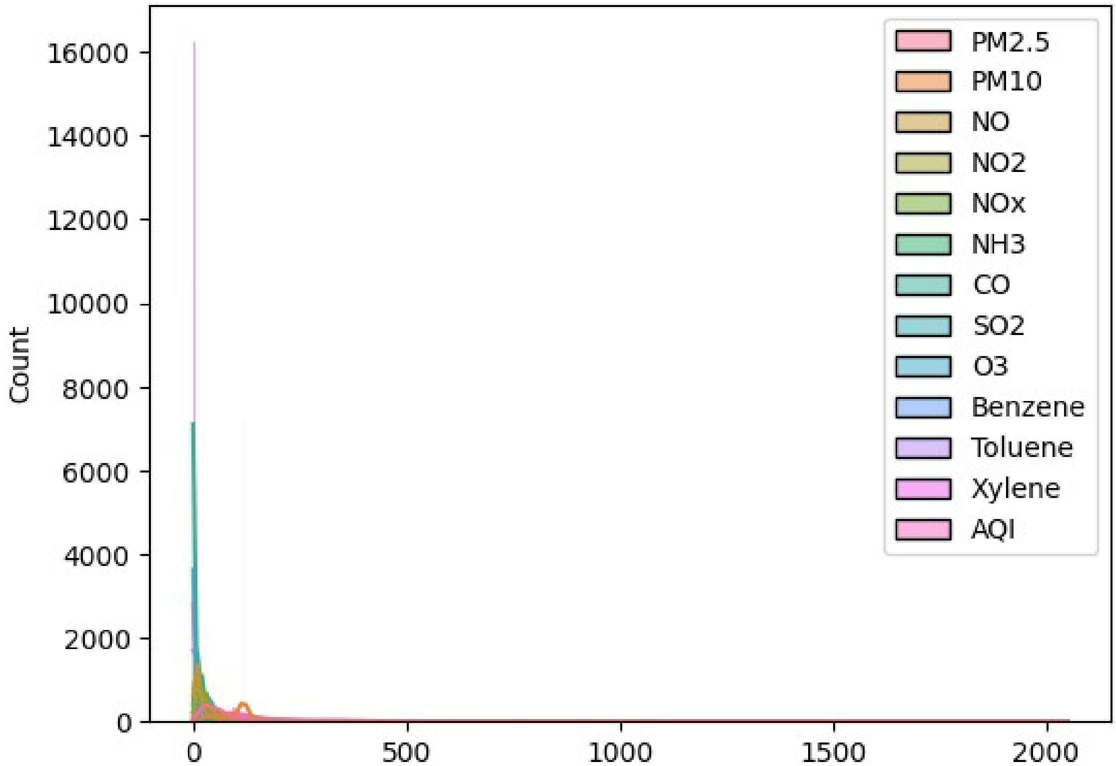
```
<Axes: xlabel='AQI_Bucket', ylabel='NH3'>
```




```
sns.histplot(data=cleandt,kde=True)
```

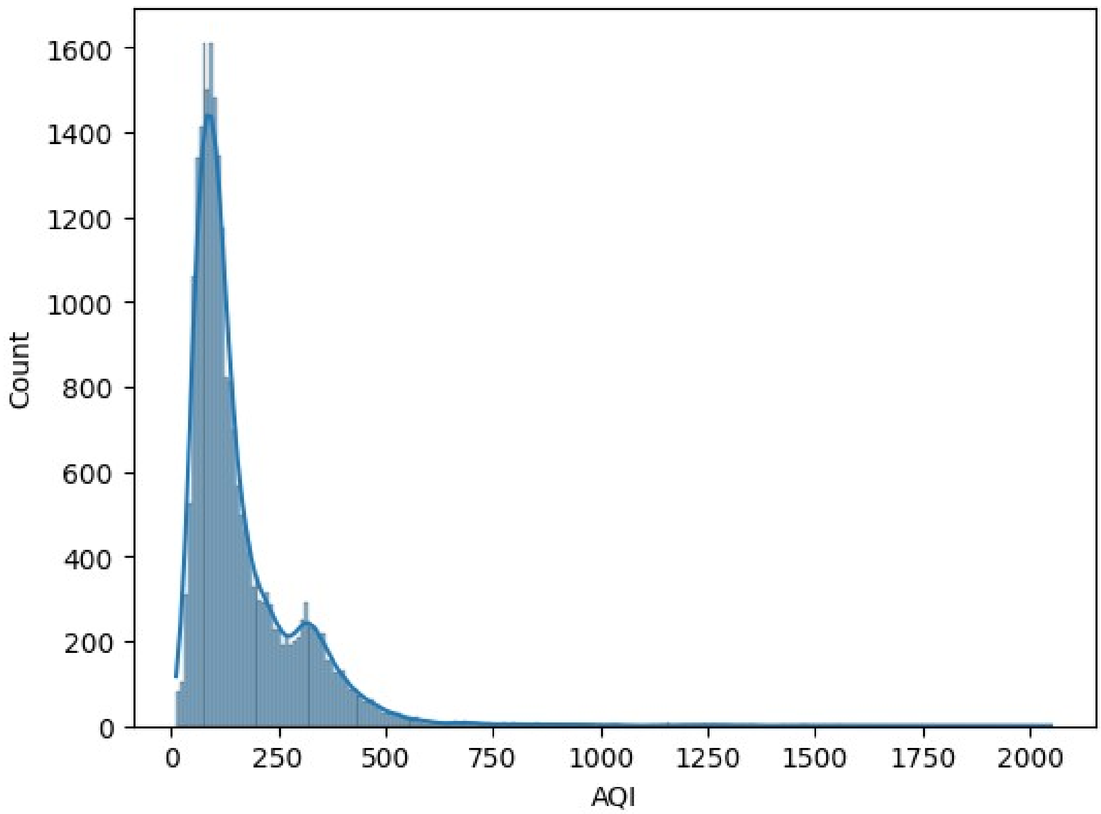
```
<Axes: ylabel='Count'>
```

```
C:\Users\kadam\AppData\Local\Programs\Python\Python312\Lib\site-packages\IPython\core\events.py:82: UserWarning: Creating legend with loc="best" can be slow with large amounts of data.
  func(*args, **kwargs)
C:\Users\kadam\AppData\Local\Programs\Python\Python312\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarning: Creating legend with loc="best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```




```
sns.histplot(data=cleandt,x=cleandt['AQI'],kde=True)
```

```
<Axes: xlabel='AQI', ylabel='Count'>
```



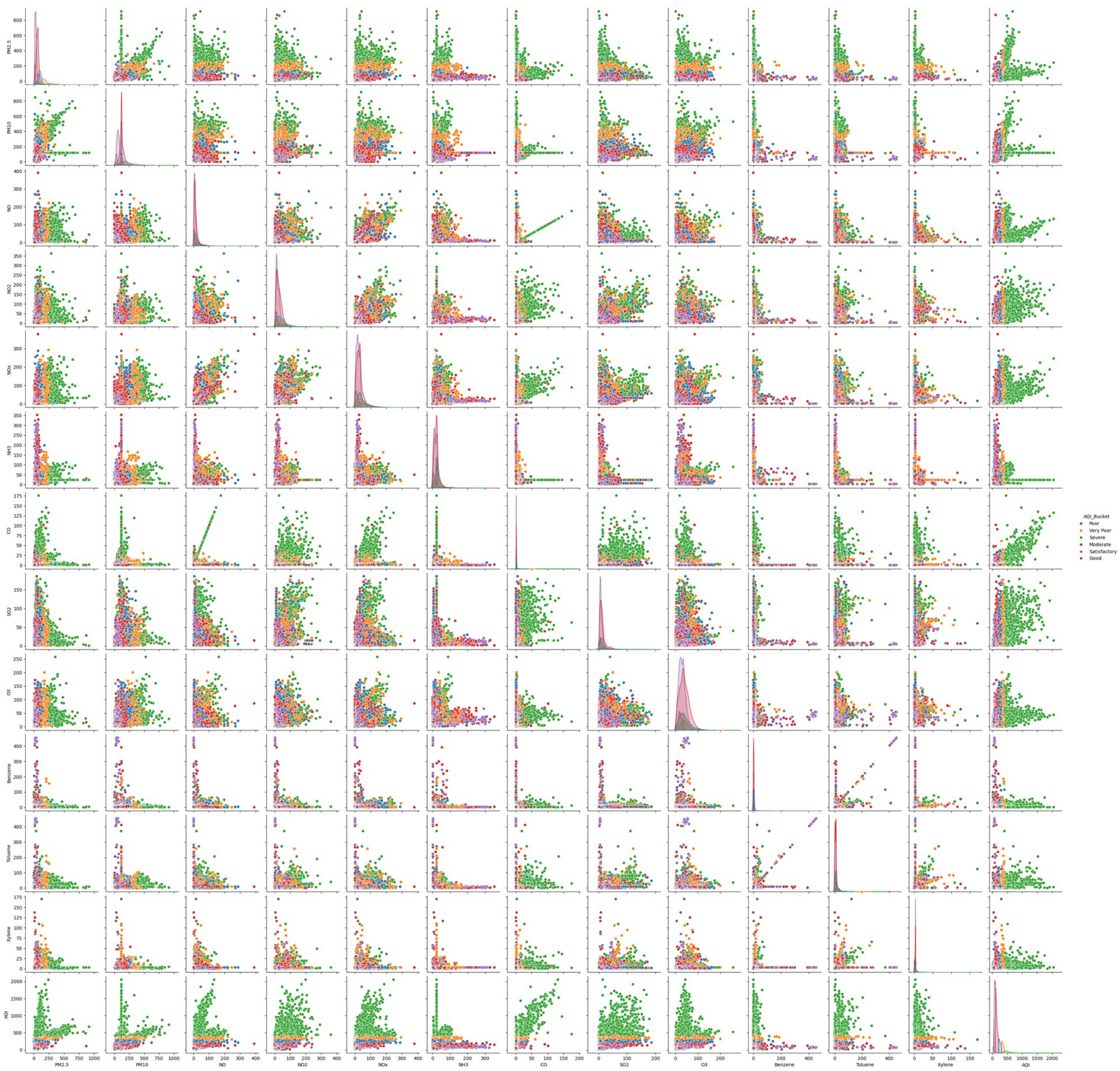
maximum no. of cities have AQI in between 50 to 200

```
relu=pd.crosstab(cleandt['AQI'] , cleandt['AQI_Bucket'])
sns.heatmap(relu)
```

```
<Axes: xlabel='AQI_Bucket', ylabel='AQI'>
```

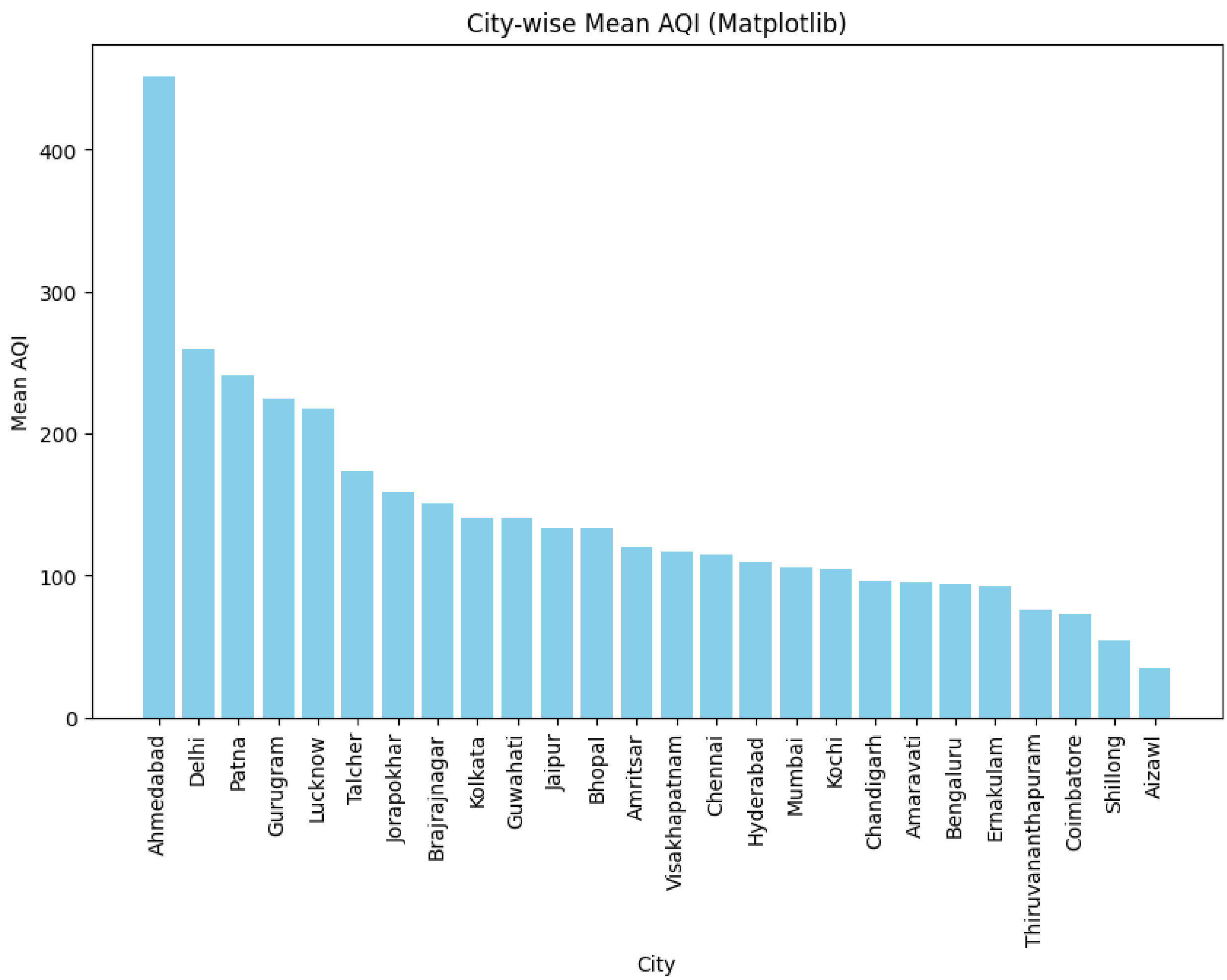

```
sns.pairplot(cleandt,hue='AQI_Bucket')
```

```
<seaborn.axisgrid.PairGrid at 0x215239d90d0>
```

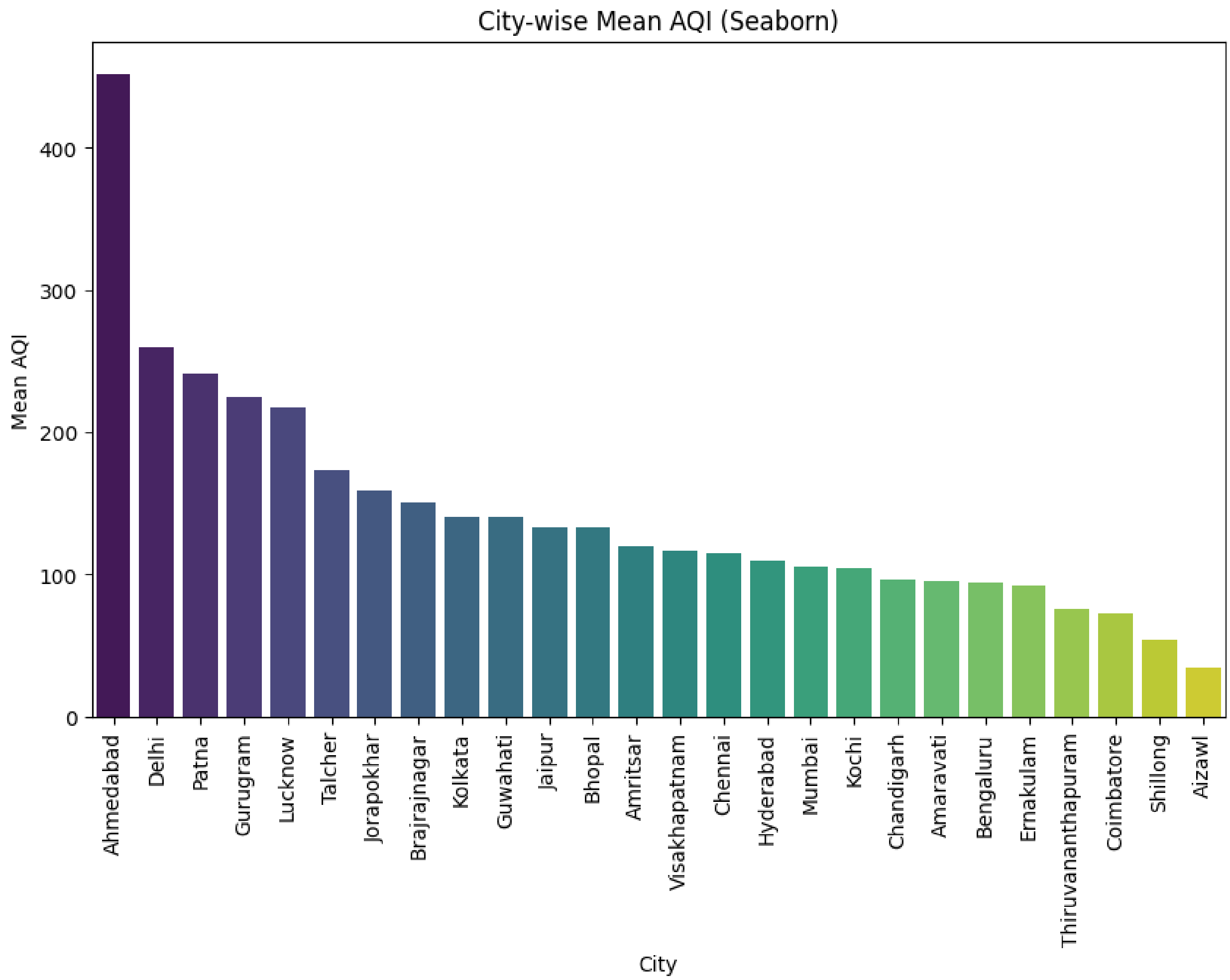



```
# Grouping the data by city and calculating the mean AQI for each city
city_aqi = cleandt.groupby('City')
['AQI'].mean().sort_values(ascending=False)

# Matplotlib bar plot
plt.figure(figsize=(10, 6))
plt.bar(city_aqi.index, city_aqi.values, color='skyblue')
plt.xticks(rotation=90)
plt.title('City-wise Mean AQI (Matplotlib)')
plt.ylabel('Mean AQI')
plt.xlabel('City')
plt.show()
```

```python
# Seaborn bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x=city_aqi.index, y=city_aqi.values, palette='viridis')
plt.xticks(rotation=90)
plt.title('City-wise Mean AQI (Seaborn)')
plt.ylabel('Mean AQI')
plt.xlabel('City')
plt.show()
```




```
C:\Users\kadam\AppData\Local\Temp\ipykernel_11664\2716694310.py:15:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=city_aqi.index, y=city_aqi.values, palette='viridis')
```



Ahmedabed has Highest Aqi.

```
# Group data by city to analyze each region's trends
city_grouped_data = cleandt.groupby('City')

# Function to generate region-specific conclusions
def region_specific_conclusions(city_group):
    # Get the average AQI for the city
    avg_aqi = city_group['AQI'].astype(float).mean()

    # Find the most frequent AQI bucket (category)
    most_common_aqi_bucket = city_group['AQI_Bucket'].mode()[0]

    # Get the average pollutant levels (PM2.5, PM10, NO2, CO)
    avg_pm25 = city_group['PM2.5'].mean()
    avg_pm10 = city_group['PM10'].mean()
    avg_no2 = city_group['NO2'].mean()
    avg_co = city_group['CO'].mean()

    # Form conclusions
    conclusion = f"For the city of {city_group['City'].iloc[0]}:\n"
```

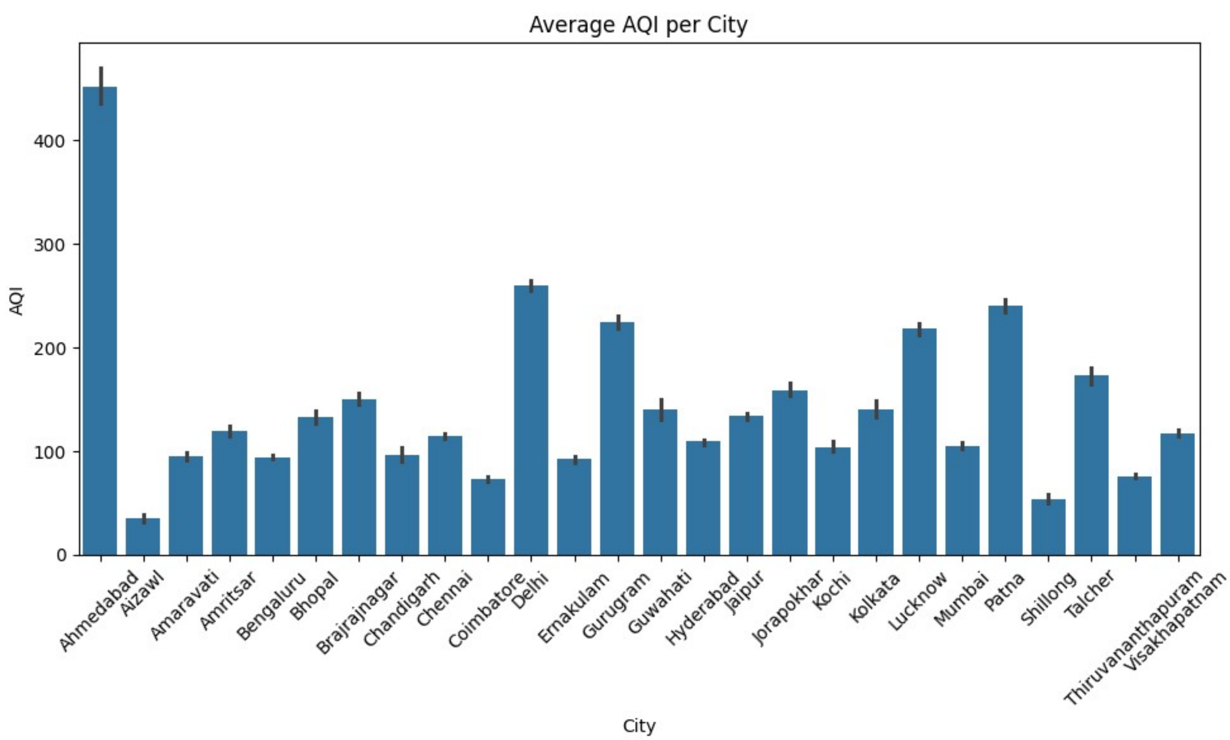
```
    conclusion += f"- The average AQI is {avg_aqi:.2f}, falling into
the '{most_common_aqi_bucket}' category.\n"
    conclusion += f"- Key pollutant levels on average are: PM2.5:
{avg_pm25:.2f}, PM10: {avg_pm10:.2f}, NO2: {avg_no2:.2f}, and CO:
{avg_co:.2f}.\n"

    # Identify a potential concern based on high average pollutant
levels
    if avg_pm25 > 60:
        conclusion += f"- PM2.5 levels are relatively high, which may
pose a health risk.\n"
    if avg_pm10 > 100:
        conclusion += f"- PM10 levels are also elevated, contributing
to reduced air quality.\n"
    if avg_no2 > 40:
        conclusion += f"- NO2 levels exceed safe limits, indicating a
concern for air pollution from combustion sources.\n"

    return conclusion

# Generate conclusions for each city
city_conclusions = {}
for city, group in city_grouped_data:
    city_conclusions[city] = region_specific_conclusions(group)

# Display the conclusions for each city
for city, conclusion in city_conclusions.items():
    print(conclusion)

# Visualization of AQI and Pollutants per city

# Convert AQI to numeric for analysis
cleandt['AQI'] = pd.to_numeric(cleandt['AQI'], errors='coerce')

# Plotting average AQI for each city
plt.figure(figsize=(10,6))
sns.barplot(x='City', y='AQI', data=cleandt, estimator=lambda x:
x.mean())
plt.title('Average AQI per City')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Plotting average PM2.5 and PM10 levels per city
plt.figure(figsize=(12,6))
sns.barplot(x='City', y='PM2.5', data=cleandt, estimator=lambda x:
x.mean(), color='blue', label='PM2.5')
sns.barplot(x='City', y='PM10', data=cleandt, estimator=lambda x:
x.mean(), color='orange', label='PM10')
plt.title('Average PM2.5 and PM10 Levels per City')
```

```
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()

For the city of Ahmedabad:
- The average AQI is 452.12, falling into the 'Severe' category.
- Key pollutant levels on average are: PM2.5: 67.87, PM10: 116.54,
NO2: 60.18, and CO: 22.09.
- PM2.5 levels are relatively high, which may pose a health risk.
- PM10 levels are also elevated, contributing to reduced air quality.
- NO2 levels exceed safe limits, indicating a concern for air
pollution from combustion sources.

For the city of Aizawl:
- The average AQI is 34.77, falling into the 'Good' category.
- Key pollutant levels on average are: PM2.5: 17.44, PM10: 24.14, NO2:
0.37, and CO: 0.28.

For the city of Amaravati:
- The average AQI is 95.30, falling into the 'Satisfactory' category.
- Key pollutant levels on average are: PM2.5: 38.25, PM10: 76.21, NO2:
21.73, and CO: 0.69.

For the city of Amritsar:
- The average AQI is 119.92, falling into the 'Satisfactory' category.
- Key pollutant levels on average are: PM2.5: 56.01, PM10: 115.10,
NO2: 18.54, and CO: 0.62.
- PM10 levels are also elevated, contributing to reduced air quality.

For the city of Bengaluru:
- The average AQI is 94.32, falling into the 'Satisfactory' category.
- Key pollutant levels on average are: PM2.5: 36.78, PM10: 88.39, NO2:
28.31, and CO: 1.67.

For the city of Bhopal:
- The average AQI is 132.83, falling into the 'Moderate' category.
- Key pollutant levels on average are: PM2.5: 50.21, PM10: 119.72,
NO2: 31.37, and CO: 0.88.
- PM10 levels are also elevated, contributing to reduced air quality.

For the city of Brajrajnagar:
- The average AQI is 150.28, falling into the 'Moderate' category.
- Key pollutant levels on average are: PM2.5: 63.99, PM10: 123.44,
NO2: 18.23, and CO: 1.84.
- PM2.5 levels are relatively high, which may pose a health risk.
- PM10 levels are also elevated, contributing to reduced air quality.

For the city of Chandigarh:
- The average AQI is 96.50, falling into the 'Satisfactory' category.
```

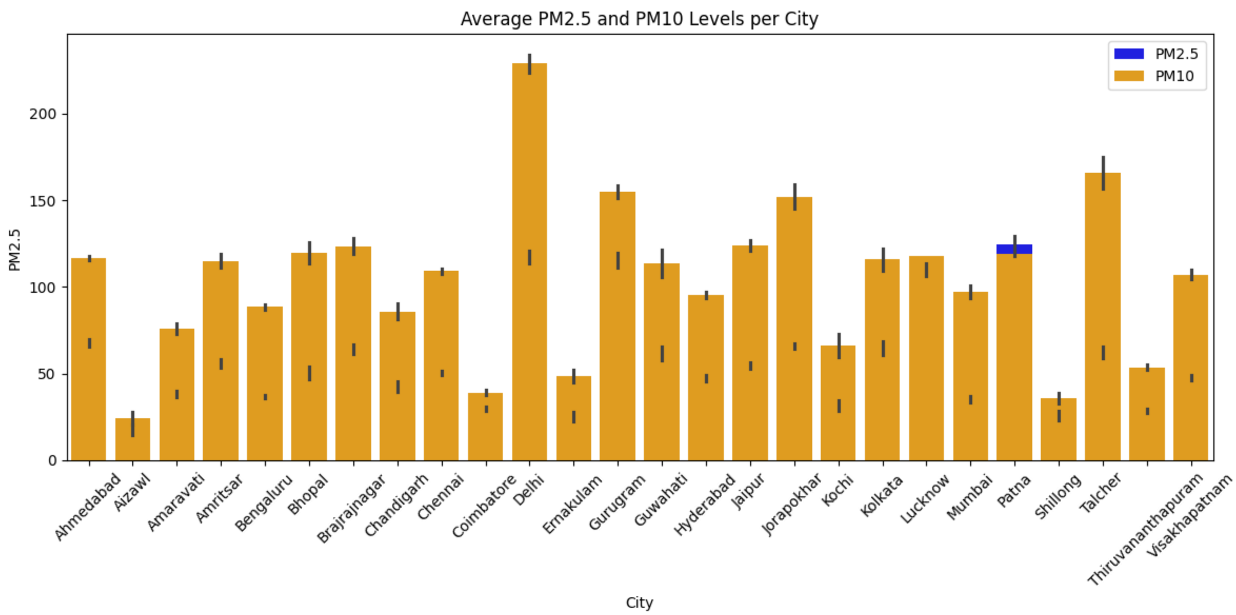
```
- Key pollutant levels on average are: PM2.5: 42.36, PM10: 85.80, NO2:
11.94, and CO: 0.63.

For the city of Chennai:
- The average AQI is 114.50, falling into the 'Satisfactory' category.
- Key pollutant levels on average are: PM2.5: 50.21, PM10: 109.26,
NO2: 16.59, and CO: 1.02.
- PM10 levels are also elevated, contributing to reduced air quality.

For the city of Coimbatore:
- The average AQI is 73.02, falling into the 'Satisfactory' category.
- Key pollutant levels on average are: PM2.5: 29.55, PM10: 39.13, NO2:
27.68, and CO: 0.95.

For the city of Delhi:
- The average AQI is 259.49, falling into the 'Poor' category.
- Key pollutant levels on average are: PM2.5: 117.52, PM10: 228.94,
NO2: 50.75, and CO: 1.99.
- PM2.5 levels are relatively high, which may pose a health risk.
- PM10 levels are also elevated, contributing to reduced air quality.
- NO2 levels exceed safe limits, indicating a concern for air
pollution from combustion sources.

For the city of Ernakulam:
- The average AQI is 92.36, falling into the 'Satisfactory' category.
- Key pollutant levels on average are: PM2.5: 25.16, PM10: 48.40, NO2:
12.12, and CO: 1.63.

For the city of Gurugram:
- The average AQI is 225.12, falling into the 'Moderate' category.
- Key pollutant levels on average are: PM2.5: 115.44, PM10: 155.23,
NO2: 24.04, and CO: 1.34.
- PM2.5 levels are relatively high, which may pose a health risk.
- PM10 levels are also elevated, contributing to reduced air quality.

For the city of Guwahati:
- The average AQI is 140.11, falling into the 'Satisfactory' category.
- Key pollutant levels on average are: PM2.5: 61.50, PM10: 113.82,
NO2: 13.54, and CO: 0.73.
- PM2.5 levels are relatively high, which may pose a health risk.
- PM10 levels are also elevated, contributing to reduced air quality.

For the city of Hyderabad:
- The average AQI is 109.21, falling into the 'Moderate' category.
- Key pollutant levels on average are: PM2.5: 47.15, PM10: 95.34, NO2:
28.50, and CO: 0.61.

For the city of Jaipur:
- The average AQI is 133.68, falling into the 'Moderate' category.
- Key pollutant levels on average are: PM2.5: 54.71, PM10: 123.78,
```

```
NO2: 32.46, and CO: 0.81.
- PM10 levels are also elevated, contributing to reduced air quality.

For the city of Jorapokhar:
- The average AQI is 159.25, falling into the 'Moderate' category.
- Key pollutant levels on average are: PM2.5: 66.02, PM10: 151.88,
NO2: 9.58, and CO: 1.20.
- PM2.5 levels are relatively high, which may pose a health risk.
- PM10 levels are also elevated, contributing to reduced air quality.

For the city of Kochi:
- The average AQI is 104.28, falling into the 'Satisfactory' category.
- Key pollutant levels on average are: PM2.5: 31.54, PM10: 66.11, NO2:
15.00, and CO: 1.29.

For the city of Kolkata:
- The average AQI is 140.57, falling into the 'Satisfactory' category.
- Key pollutant levels on average are: PM2.5: 64.65, PM10: 116.08,
NO2: 40.80, and CO: 0.85.
- PM2.5 levels are relatively high, which may pose a health risk.
- PM10 levels are also elevated, contributing to reduced air quality.
- NO2 levels exceed safe limits, indicating a concern for air
pollution from combustion sources.

For the city of Lucknow:
- The average AQI is 217.97, falling into the 'Moderate' category.
- Key pollutant levels on average are: PM2.5: 109.75, PM10: 118.13,
NO2: 33.88, and CO: 1.75.
- PM2.5 levels are relatively high, which may pose a health risk.
- PM10 levels are also elevated, contributing to reduced air quality.

For the city of Mumbai:
- The average AQI is 105.35, falling into the 'Satisfactory' category.
- Key pollutant levels on average are: PM2.5: 35.30, PM10: 97.24, NO2:
25.59, and CO: 1.23.

For the city of Patna:
- The average AQI is 240.78, falling into the 'Moderate' category.
- Key pollutant levels on average are: PM2.5: 124.65, PM10: 119.26,
NO2: 39.06, and CO: 1.59.
- PM2.5 levels are relatively high, which may pose a health risk.
- PM10 levels are also elevated, contributing to reduced air quality.

For the city of Shillong:
- The average AQI is 53.80, falling into the 'Good' category.
- Key pollutant levels on average are: PM2.5: 25.77, PM10: 35.84, NO2:
3.50, and CO: 0.27.

For the city of Talcher:
- The average AQI is 172.89, falling into the 'Moderate' category.
```

```
- Key pollutant levels on average are: PM2.5: 62.25, PM10: 165.72,
NO2: 15.24, and CO: 1.82.
- PM2.5 levels are relatively high, which may pose a health risk.
- PM10 levels are also elevated, contributing to reduced air quality.

For the city of Thiruvananthapuram:
- The average AQI is 75.88, falling into the 'Satisfactory' category.
- Key pollutant levels on average are: PM2.5: 28.61, PM10: 53.59, NO2:
9.54, and CO: 0.98.

For the city of Visakhapatnam:
- The average AQI is 117.27, falling into the 'Moderate' category.
- Key pollutant levels on average are: PM2.5: 47.75, PM10: 107.02,
NO2: 37.10, and CO: 0.86.
- PM10 levels are also elevated, contributing to reduced air quality.
```




```
C:\Users\kadam\AppData\Local\Temp\ipykernel_11664\633187323.py:62:
UserWarning: Creating legend with loc="best" can be slow with large
amounts of data.
  plt.tight_layout()
```



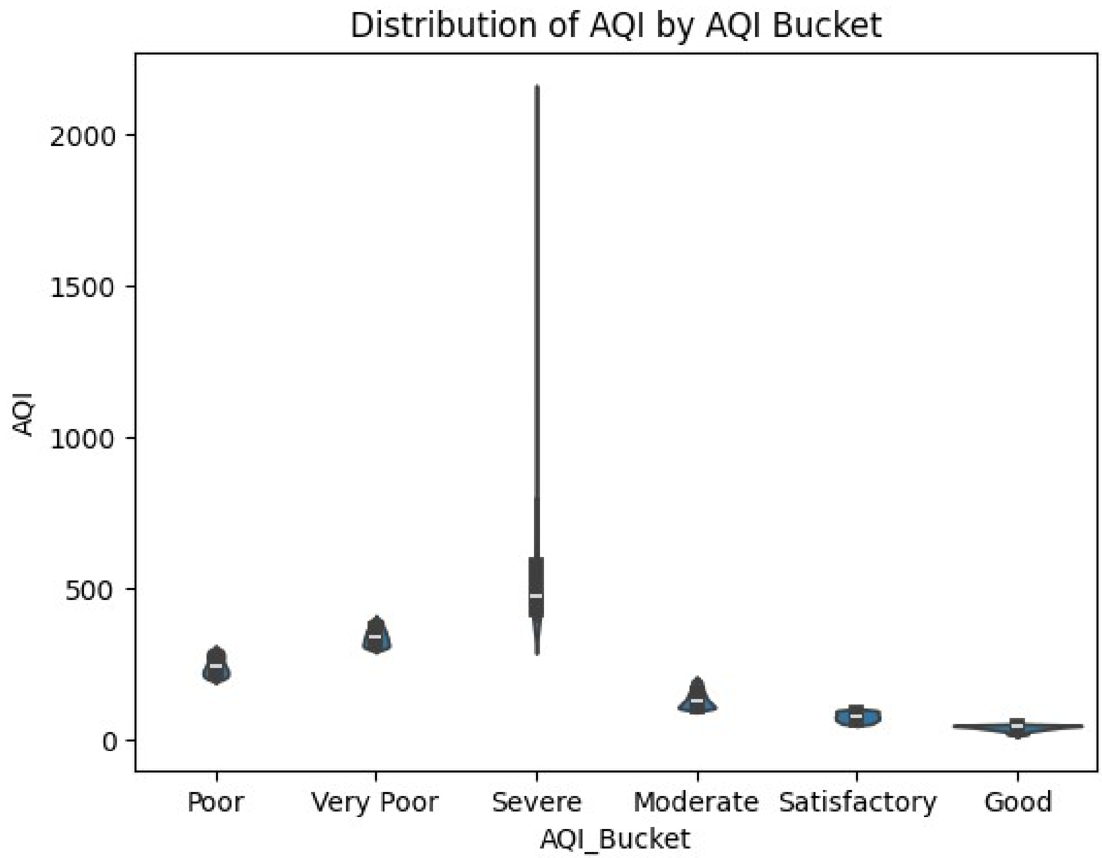

```
plt.hist(cleandt[], histtype='bar', rwidth=0.8)
plt.xlabel('age groups')
plt.ylabel('Number of people')
plt.title('Histogram')
plt.show()
```

```
  Cell In[14], line 1
    plt.hist(cleandt[], histtype='bar', rwidth=0.8)
                     ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

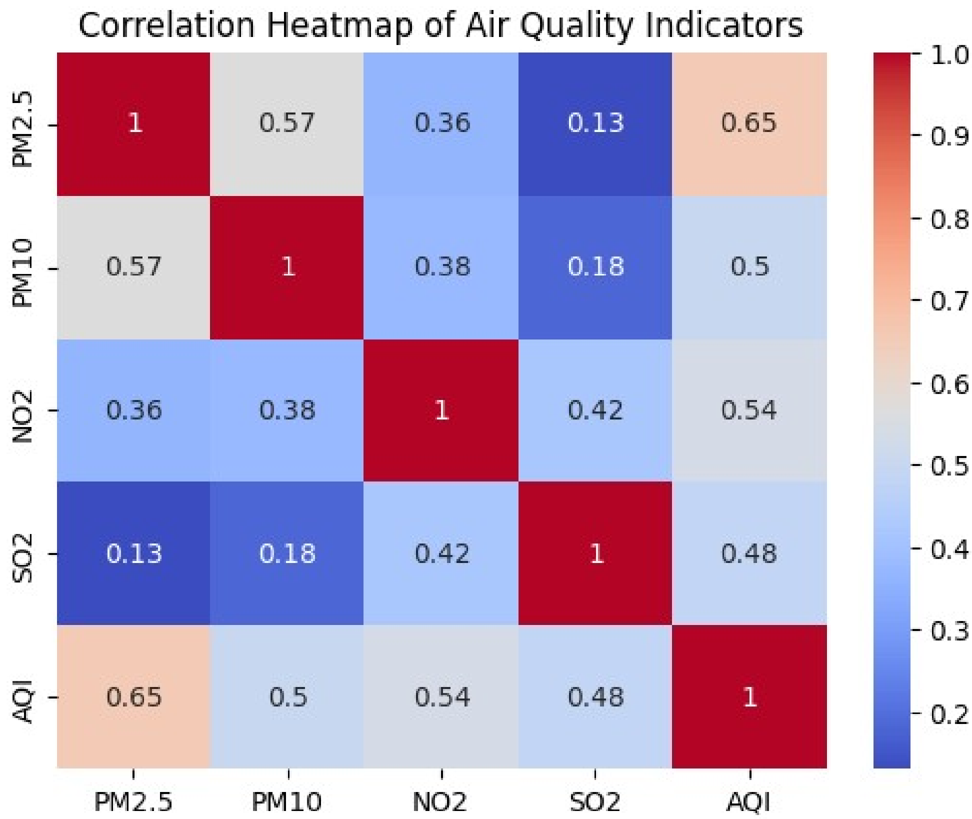
```
aqi_count= cleandt.AQI_Bucket.value_counts()
aqi_count
```

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

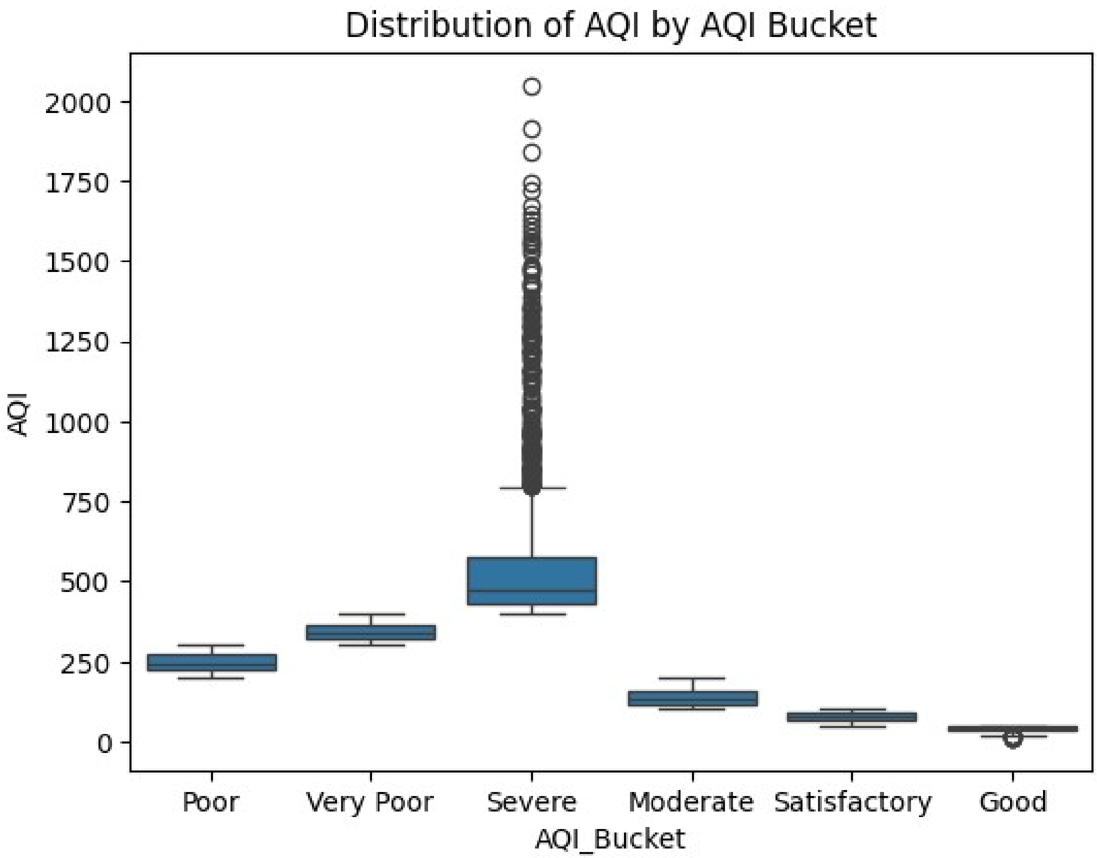
```
# Violin plot for AQI by AQI_Bucke
sns.violinplot(x='AQI_Bucket', y='AQI', data=cleandt)
plt.title('Distribution of AQI by AQI Bucket')
plt.show()
```



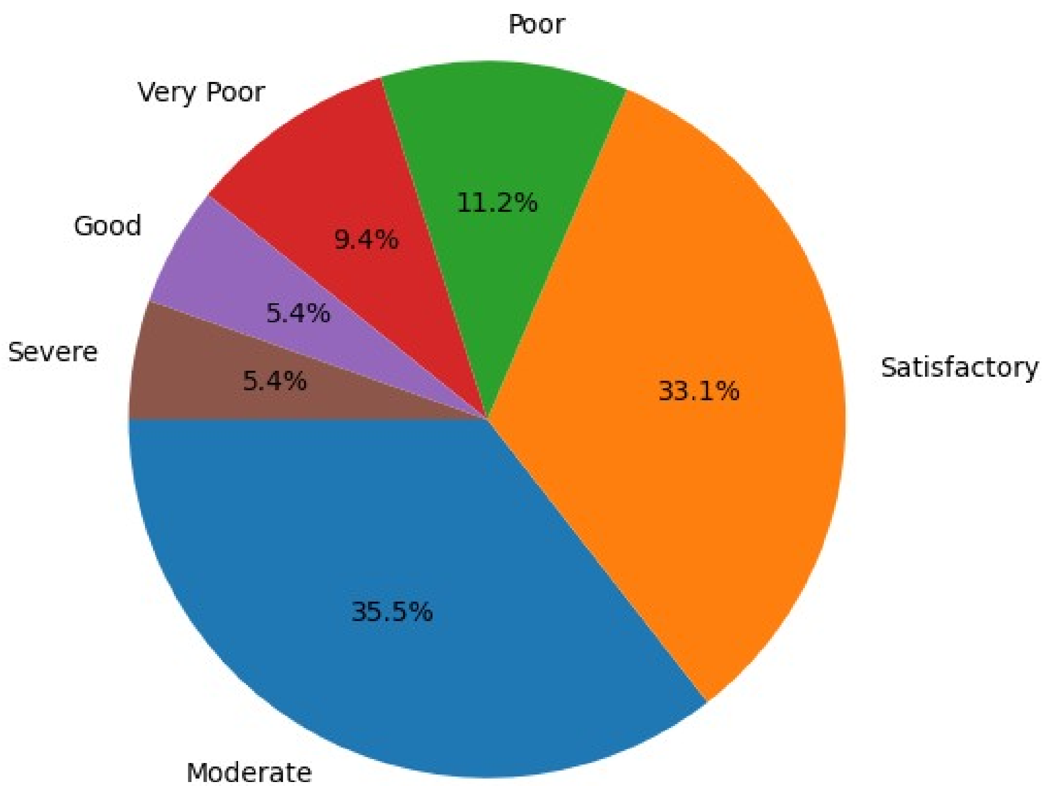

```
# Correlation matrix for numerical variables
correlation_matrix = cleandt[['PM2.5', 'PM10', 'NO2', 'SO2',
'AQI']].corr()

# Plotting heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap of Air Quality Indicators')
plt.show()
```




```
sns.boxplot(x='AQI_Bucket', y='AQI', data=cleandt)
plt.title('Distribution of AQI by AQI Bucket')
plt.show()
```




```
plt.figure(figsize=(12,6))
#plt.title(schema.Gender)
plt.pie(aqi_count, labels=aqi_count.index, autopct='%1.1f%%',
startangle=180);
```



Linear Regression Model

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# creating 2 datasets.
# x contains all the columns essential to predict the AQI.
# y contains the AQI itself.
# we are going to predict the AQI based on air components.
x = cleandt.drop(['City','Date','AQI','AQI_Bucket'],axis=1)
y = cleandt['AQI']
x
```

```
        PM2.5        PM10     NO    NO2    NOx        NH3     CO
SO2  \
0       83.13  118.127103   6.93  28.71  33.72  23.483476   6.93
49.52
1       79.84  118.127103  13.85  28.68  41.08  23.483476  13.85
48.49
```

```
2         94.52  118.127103  24.39  32.66  52.61  23.483476  24.39
67.39
3        135.99  118.127103  43.48  42.08  84.57  23.483476  43.48
75.23
4        178.33  118.127103  54.56  35.31  72.80  23.483476  54.56
55.04
...         ...         ...    ...    ...    ...        ...    ...    ..
.
24845     15.02   50.940000   7.68  25.06  19.54  12.470000   0.47
8.55
24846     24.38   74.090000   3.42  26.06  16.53  11.990000   0.52
12.72
24847     22.91   65.730000   3.45  29.53  18.33  10.710000   0.48
8.42
24848     16.64   49.970000   4.05  29.26  18.80  10.030000   0.52
9.84
24849     15.00   66.000000   0.40  26.85  14.05   5.200000   0.59
2.10

           O3  Benzene    Toluene     Xylene
0       59.76  0.02000   0.000000   3.140000
1       97.07  0.04000   0.000000   4.810000
2      111.33  0.24000   0.010000   7.670000
3      102.70  0.40000   0.040000  25.870000
4      107.38  0.46000   0.060000  35.610000
...       ...      ...        ...        ...
24845   23.30  2.24000  12.070000   0.730000
24846   30.14  0.74000   2.210000   0.380000
24847   30.96  0.01000   0.010000   0.000000
24848   28.30  0.00000   0.000000   0.000000
24849   17.05  3.28084   8.700972   3.070128

[24850 rows x 12 columns]
```

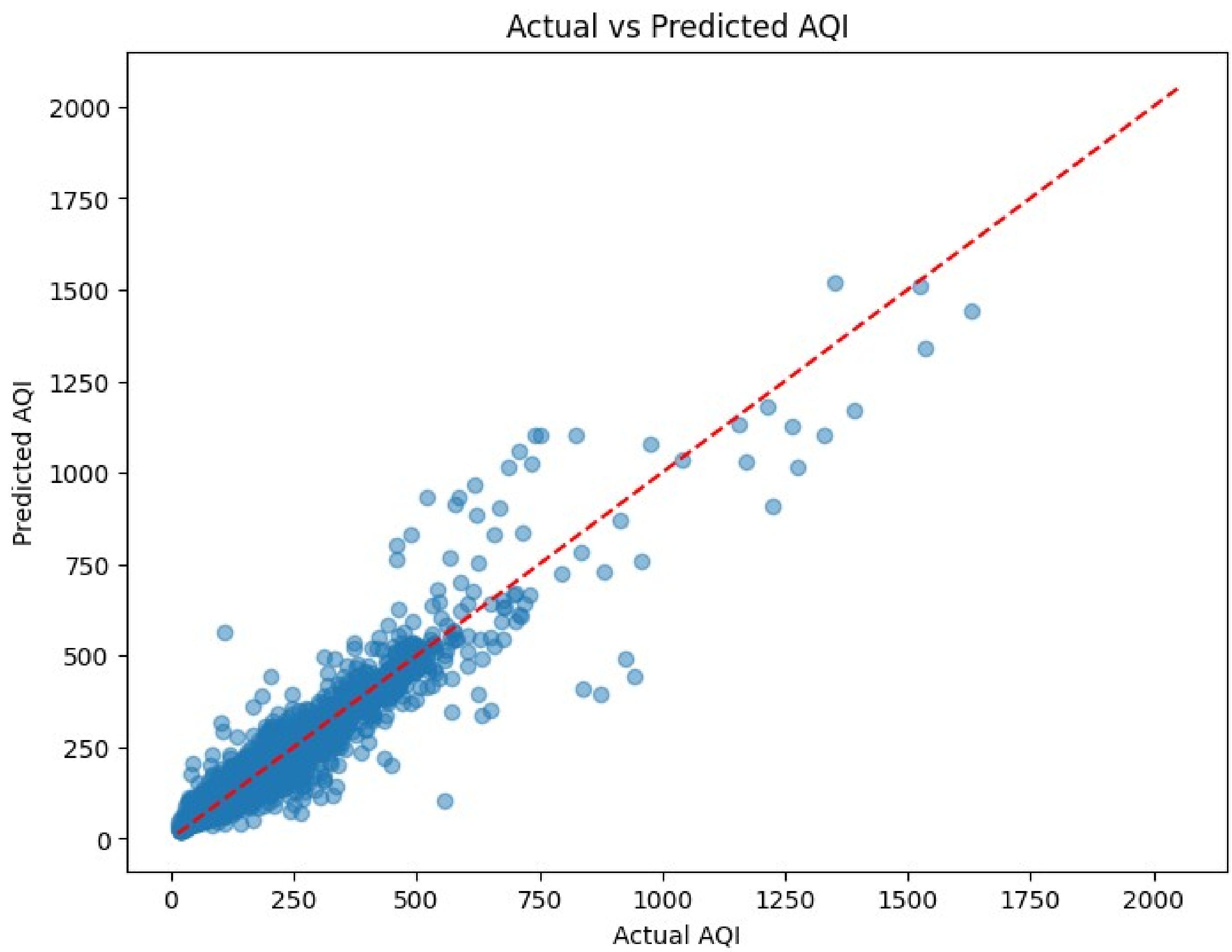
```python
#splitting the dataset
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)

#model initialization
model = LinearRegression()
model.fit(x_train, y_train)
```

```
LinearRegression()
```

```python
print(model.intercept_)
print(model.coef_)
```

```
6.768096690239048
[ 1.11978235  0.29197875 -0.1353422   0.24151263  0.09964323 -
0.0675648
```

```
 11.72232249  0.68818773  0.1982416   0.02099309 -0.02508842 -
0.13344967]

#predicting the AQI
y_pred = model.predict(x_test)

print(y_pred)

[114.04836463 200.79703022 116.08024452 ... 172.82832023 109.77860936
 120.67003006]

#rmse : Root Mean Squared Error
#RMSE is the square root of the average squared differences between
#the predicted values (y_pred) and the actual values (y_test)

#R² (R-squared), or the coefficient of determination, is a statistical
#measure that tells how well the independent variables (features)
#explain the variance in the dependent variable (target).
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R² Score: {r2}")

Root Mean Squared Error (RMSE): 40.783195026017644
R² Score: 0.9091656886933218

plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')
plt.title("Actual vs Predicted AQI")
plt.xlabel("Actual AQI")
plt.ylabel("Predicted AQI")
plt.show()
```




```
import pandas as pd

cgr_data = {'category': ['Severe', 'Very Poor','Poor' , 'Moderate',
'Satisfactory','Good']]}
cgr_num = pd.DataFrame()
# Mapping dictionary
mapping = {'Severe': 0, 'Very Poor': 1, 'Poor':
2,'Moderate':3,'Satisfactory':4,'Good':5,}

# Creating a new column with mapped values
cleandt['AQI_Bucket_num'] = cleandt['AQI_Bucket'].map(mapping)
cleandt=cleandt.drop('numeric_category',axis=1)

cleandt

                 City        Date  PM2.5        PM10     NO    NO2
NOx  \
0           Ahmedabad  2015-01-29  83.13  118.127103   6.93  28.71
33.72
1           Ahmedabad  2015-01-30  79.84  118.127103  13.85  28.68
41.08
```

```
2          Ahmedabad  2015-01-31   94.52  118.127103  24.39  32.66
52.61
3          Ahmedabad  2015-02-01  135.99  118.127103  43.48  42.08
84.57
4          Ahmedabad  2015-02-02  178.33  118.127103  54.56  35.31
72.80
...              ...         ...     ...         ...    ...    ...
...
24845  Visakhapatnam  2020-06-27   15.02   50.940000   7.68  25.06
19.54
24846  Visakhapatnam  2020-06-28   24.38   74.090000   3.42  26.06
16.53
24847  Visakhapatnam  2020-06-29   22.91   65.730000   3.45  29.53
18.33
24848  Visakhapatnam  2020-06-30   16.64   49.970000   4.05  29.26
18.80
24849  Visakhapatnam  2020-07-01   15.00   66.000000   0.40  26.85
14.05

             NH3     CO    SO2      O3  Benzene    Toluene     Xylene
AQI  \
0      23.483476   6.93  49.52   59.76  0.02000   0.000000   3.140000
209.0
1      23.483476  13.85  48.49   97.07  0.04000   0.000000   4.810000
328.0
2      23.483476  24.39  67.39  111.33  0.24000   0.010000   7.670000
514.0
3      23.483476  43.48  75.23  102.70  0.40000   0.040000  25.870000
782.0
4      23.483476  54.56  55.04  107.38  0.46000   0.060000  35.610000
914.0
...          ...    ...    ...     ...      ...        ...        ...
...
24845  12.470000   0.47   8.55   23.30  2.24000  12.070000   0.730000
41.0
24846  11.990000   0.52  12.72   30.14  0.74000   2.210000   0.380000
70.0
24847  10.710000   0.48   8.42   30.96  0.01000   0.010000   0.000000
68.0
24848  10.030000   0.52   9.84   28.30  0.00000   0.000000   0.000000
54.0
24849   5.200000   0.59   2.10   17.05  3.28084   8.700972   3.070128
50.0

        AQI_Bucket  AQI_Bucket_num
0             Poor               2
1        Very Poor               1
2           Severe               0
3           Severe               0
```

```
4            Severe                0
...             ...              ...
24845          Good                5
24846  Satisfactory                4
24847  Satisfactory                4
24848  Satisfactory                4
24849          Good                5

[24850 rows x 17 columns]
```

```python
a = cleandt.drop(['City','Date','AQI_Bucket_num','AQI_Bucket'],axis=1)
b = cleandt['AQI_Bucket_num']
a
```

```
        PM2.5        PM10     NO    NO2    NOx        NH3     CO
SO2  \
0       83.13  118.127103   6.93  28.71  33.72  23.483476   6.93
49.52
1       79.84  118.127103  13.85  28.68  41.08  23.483476  13.85
48.49
2       94.52  118.127103  24.39  32.66  52.61  23.483476  24.39
67.39
3      135.99  118.127103  43.48  42.08  84.57  23.483476  43.48
75.23
4      178.33  118.127103  54.56  35.31  72.80  23.483476  54.56
55.04
...       ...         ...    ...    ...    ...        ...    ...    ..
.
24845   15.02   50.940000   7.68  25.06  19.54  12.470000   0.47
8.55
24846   24.38   74.090000   3.42  26.06  16.53  11.990000   0.52
12.72
24847   22.91   65.730000   3.45  29.53  18.33  10.710000   0.48
8.42
24848   16.64   49.970000   4.05  29.26  18.80  10.030000   0.52
9.84
24849   15.00   66.000000   0.40  26.85  14.05   5.200000   0.59
2.10

           O3  Benzene    Toluene     Xylene    AQI
0       59.76  0.02000   0.000000   3.140000  209.0
1       97.07  0.04000   0.000000   4.810000  328.0
2      111.33  0.24000   0.010000   7.670000  514.0
3      102.70  0.40000   0.040000  25.870000  782.0
4      107.38  0.46000   0.060000  35.610000  914.0
...       ...      ...        ...        ...    ...
24845   23.30  2.24000  12.070000   0.730000   41.0
24846   30.14  0.74000   2.210000   0.380000   70.0
24847   30.96  0.01000   0.010000   0.000000   68.0
24848   28.30  0.00000   0.000000   0.000000   54.0
```

```
24849   17.05  3.28084   8.700972   3.070128   50.0

[24850 rows x 13 columns]
```

```
b
```

```
0        2
1        1
2        0
3        0
4        0
        ..
24845    5
24846    4
24847    4
24848    4
24849    5
Name: AQI_Bucket_num, Length: 24850, dtype: int64
```

```
a_train, a_test, b_train, b_test = train_test_split(a, b, test_size =
0.20, random_state = 42, stratify = cleandt['AQI_Bucket_num'] )
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(a_train, b_train)
b_predict = model.predict(a_test)
```

```
C:\Users\kadam\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
b_predict
```

```
array([4, 1, 3, ..., 3, 4, 1])
```

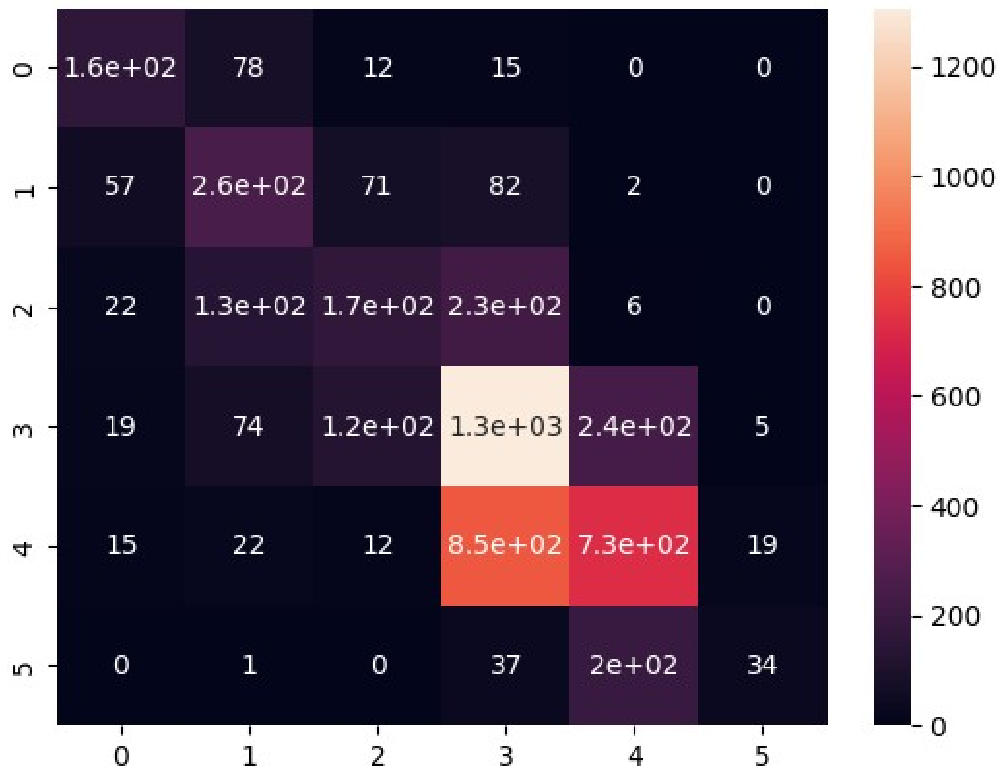
```
# Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(b_test, b_predict)
cm
```

```
array([[ 163,   78,   12,   15,    0,    0],
       [  57,  255,   71,   82,    2,    0],
```

```
       [  22,  130,  169,  229,    6,    0],
       [  19,   74,  124, 1306,  238,    5],
       [  15,   22,   12,  849,  728,   19],
       [   0,    1,    0,   37,  196,   34]])
```

```
sns.heatmap(pd.DataFrame(cm), annot=True)
```

```
<Axes: >
```




```
from sklearn.metrics import accuracy_score

accuracy =accuracy_score(b_test, b_predict)
accuracy
```

```
0.53420523138833
```

```
b_predict =
model.predict([[1,148,72,35,79.799,33.6,0.627,50,0,3,5,4,2]])
print(b_predict)
if b_predict==0:
    print("Severe")
else:
    print("Not Severe")
```

```
[5]
Not Severe
```

```
C:\Users\kadam\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid
feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

```python
import pandas as pd

# Step 2: Create a dictionary that maps each city to its state
city_to_state = {
    "Ahmedabad": "Gujarat",
    "Bengaluru": "Karnataka",
    "Chennai": "Tamil Nadu",
    "Mumbai": "Maharashtra",
    "Lucknow": "Uttar Pradesh",
    "Delhi": "Delhi",
    "Hyderabad": "Telangana",
    "Patna": "Bihar",
    "Gurugram": "Haryana",
    "Visakhapatnam": "Andhra Pradesh",
    "Amritsar": "Punjab",
    "Jorapokhar": "Jharkhand",
    "Jaipur": "Rajasthan",
    "Thiruvananthapuram": "Kerala",
    "Amaravati": "Andhra Pradesh",
    "Brajrajnagar": "Odisha",
    "Talcher": "Odisha",
    "Kolkata": "West Bengal",
    "Guwahati": "Assam",
    "Coimbatore": "Tamil Nadu",
    "Shillong": "Meghalaya",
    "Chandigarh": "Chandigarh",
    "Bhopal": "Madhya Pradesh",
    "Kochi": "Kerala",
    "Ernakulam": "Kerala",
    "Aizawl": "Mizoram"
}

# Step 3: Add a new column 'State' by mapping the cities to their
states
cleandt['State'] = cleandt['City'].map(city_to_state)

# Display the DataFrame
print(cleandt)
```

```
                City        Date   PM2.5        PM10     NO    NO2
NOx  \
0          Ahmedabad  2015-01-29   83.13  118.127103   6.93  28.71
33.72
1          Ahmedabad  2015-01-30   79.84  118.127103  13.85  28.68
41.08
```

```
2         Ahmedabad  2015-01-31   94.52  118.127103  24.39  32.66
52.61
3         Ahmedabad  2015-02-01  135.99  118.127103  43.48  42.08
84.57
4         Ahmedabad  2015-02-02  178.33  118.127103  54.56  35.31
72.80
...             ...         ...     ...         ...    ...    ...
...
24845  Visakhapatnam  2020-06-27   15.02   50.940000   7.68  25.06
19.54
24846  Visakhapatnam  2020-06-28   24.38   74.090000   3.42  26.06
16.53
24847  Visakhapatnam  2020-06-29   22.91   65.730000   3.45  29.53
18.33
24848  Visakhapatnam  2020-06-30   16.64   49.970000   4.05  29.26
18.80
24849  Visakhapatnam  2020-07-01   15.00   66.000000   0.40  26.85
14.05

             NH3     CO    SO2      O3  Benzene    Toluene     Xylene
AQI  \
0      23.483476   6.93  49.52   59.76  0.02000   0.000000   3.140000
209.0
1      23.483476  13.85  48.49   97.07  0.04000   0.000000   4.810000
328.0
2      23.483476  24.39  67.39  111.33  0.24000   0.010000   7.670000
514.0
3      23.483476  43.48  75.23  102.70  0.40000   0.040000  25.870000
782.0
4      23.483476  54.56  55.04  107.38  0.46000   0.060000  35.610000
914.0
...          ...    ...    ...     ...      ...        ...        ...
...
24845  12.470000   0.47   8.55   23.30  2.24000  12.070000   0.730000
41.0
24846  11.990000   0.52  12.72   30.14  0.74000   2.210000   0.380000
70.0
24847  10.710000   0.48   8.42   30.96  0.01000   0.010000   0.000000
68.0
24848  10.030000   0.52   9.84   28.30  0.00000   0.000000   0.000000
54.0
24849   5.200000   0.59   2.10   17.05  3.28084   8.700972   3.070128
50.0

        AQI_Bucket          State
0             Poor        Gujarat
1        Very Poor        Gujarat
2           Severe        Gujarat
3           Severe        Gujarat
```

```
4            Severe         Gujarat
...             ...             ...
24845          Good  Andhra Pradesh
24846  Satisfactory  Andhra Pradesh
24847  Satisfactory  Andhra Pradesh
24848  Satisfactory  Andhra Pradesh
24849          Good  Andhra Pradesh

[24850 rows x 17 columns]
```

cleandt

```
                City        Date   PM2.5        PM10     NO    NO2
NOx  \
0          Ahmedabad  2015-01-29   83.13  118.127103   6.93  28.71
33.72
1          Ahmedabad  2015-01-30   79.84  118.127103  13.85  28.68
41.08
2          Ahmedabad  2015-01-31   94.52  118.127103  24.39  32.66
52.61
3          Ahmedabad  2015-02-01  135.99  118.127103  43.48  42.08
84.57
4          Ahmedabad  2015-02-02  178.33  118.127103  54.56  35.31
72.80
...              ...         ...     ...         ...    ...    ...
...
24845  Visakhapatnam  2020-06-27   15.02   50.940000   7.68  25.06
19.54
24846  Visakhapatnam  2020-06-28   24.38   74.090000   3.42  26.06
16.53
24847  Visakhapatnam  2020-06-29   22.91   65.730000   3.45  29.53
18.33
24848  Visakhapatnam  2020-06-30   16.64   49.970000   4.05  29.26
18.80
24849  Visakhapatnam  2020-07-01   15.00   66.000000   0.40  26.85
14.05

             NH3     CO    SO2      O3  Benzene    Toluene     Xylene
AQI  \
0      23.483476   6.93  49.52   59.76  0.02000   0.000000   3.140000
209.0
1      23.483476  13.85  48.49   97.07  0.04000   0.000000   4.810000
328.0
2      23.483476  24.39  67.39  111.33  0.24000   0.010000   7.670000
514.0
3      23.483476  43.48  75.23  102.70  0.40000   0.040000  25.870000
782.0
4      23.483476  54.56  55.04  107.38  0.46000   0.060000  35.610000
914.0
...          ...    ...    ...     ...      ...        ...        ...
```

```
...
24845  12.470000   0.47   8.55   23.30  2.24000  12.070000   0.730000
41.0
24846  11.990000   0.52  12.72   30.14  0.74000   2.210000   0.380000
70.0
24847  10.710000   0.48   8.42   30.96  0.01000   0.010000   0.000000
68.0
24848  10.030000   0.52   9.84   28.30  0.00000   0.000000   0.000000
54.0
24849   5.200000   0.59   2.10   17.05  3.28084   8.700972   3.070128
50.0

         AQI_Bucket           State
0              Poor         Gujarat
1         Very Poor         Gujarat
2            Severe         Gujarat
3            Severe         Gujarat
4            Severe         Gujarat
...             ...             ...
24845          Good  Andhra Pradesh
24846  Satisfactory  Andhra Pradesh
24847  Satisfactory  Andhra Pradesh
24848  Satisfactory  Andhra Pradesh
24849          Good  Andhra Pradesh

[24850 rows x 17 columns]
```

```
state_aqi_avg = cleandt.groupby("State")["AQI"].mean()
```

```
state_aqi_avg
```

```
State
Andhra Pradesh    108.086481
Assam             140.111111
Bihar             240.782042
Chandigarh         96.498328
Delhi             259.487744
Gujarat           452.122939
Haryana           225.123882
Jharkhand         159.251621
Karnataka          94.318325
Kerala             81.021277
Madhya Pradesh    132.827338
Maharashtra       105.352258
Meghalaya          53.795122
Mizoram            34.765766
Odisha            161.463501
Punjab            119.920959
Rajasthan         133.679159
Tamil Nadu        108.098294
```

```
Telangana         109.207447
Uttar Pradesh     217.973059
West Bengal       140.566313
Name: AQI, dtype: float64
```

```
import pandas as pd



# Convert the 'Date' column to datetime format
cleandt['Date'] = pd.to_datetime(cleandt['Date'])

# Extract the year from the 'Date' column
cleandt['Year'] = cleandt['Date'].dt.year

# Group by 'Year' and calculate the average AQI for each year
yearly_aqi_avg15 = cleandt[cleandt['Date'].dt.year ==
2015].groupby("State")["AQI"].mean().reset_index()
yearly_aqi_avg = cleandt.groupby(["Year","State"])
["AQI"].mean().reset_index()

# Display the result
#print("Average AQI by Year:")
yearly_aqi_avg
```

```
    Year          State         AQI
0   2015          Bihar  350.555556
1   2015          Delhi  297.024658
2   2015        Gujarat  310.950570
3   2015      Karnataka  112.573427
4   2015     Tamil Nadu  148.333333
..   ...            ...         ...
82  2020      Rajasthan  105.120219
83  2020     Tamil Nadu   74.705015
84  2020      Telangana   78.174863
85  2020  Uttar Pradesh  157.125683
86  2020    West Bengal  117.295082

[87 rows x 3 columns]
```

```
yearly_aqi_avg16 = cleandt[cleandt['Date'].dt.year ==
2015].groupby("State")["AQI"].mean().reset_index()
yearly_aqi_avg16
```

```
           State         AQI
0          Bihar  350.555556
1          Delhi  297.024658
2        Gujarat  310.950570
3      Karnataka  112.573427
4     Tamil Nadu  148.333333
```

```
5      Telangana  143.419118
6  Uttar Pradesh  202.235915
```

```
yearly_aqi_avg17 = cleandt[cleandt['Date'].dt.year ==
2015].groupby("State")["AQI"].mean().reset_index()
yearly_aqi_avg17
```

	State	AQI
0	Bihar	350.555556
1	Delhi	297.024658
2	Gujarat	310.950570
3	Karnataka	112.573427
4	Tamil Nadu	148.333333
5	Telangana	143.419118
6	Uttar Pradesh	202.235915

```
yearly_aqi_avg18 = cleandt[cleandt['Date'].dt.year ==
2015].groupby("State")["AQI"].mean().reset_index()
yearly_aqi_avg18
```

	State	AQI
0	Bihar	350.555556
1	Delhi	297.024658
2	Gujarat	310.950570
3	Karnataka	112.573427
4	Tamil Nadu	148.333333
5	Telangana	143.419118
6	Uttar Pradesh	202.235915

```
yearly_aqi_avg19 = cleandt[cleandt['Date'].dt.year ==
2015].groupby("State")["AQI"].mean().reset_index()
yearly_aqi_avg19
```

	State	AQI
0	Bihar	350.555556
1	Delhi	297.024658
2	Gujarat	310.950570
3	Karnataka	112.573427
4	Tamil Nadu	148.333333
5	Telangana	143.419118
6	Uttar Pradesh	202.235915

```
yearly_aqi_avg20 = cleandt[cleandt['Date'].dt.year ==
2015].groupby("State")["AQI"].mean().reset_index()
yearly_aqi_avg20
```

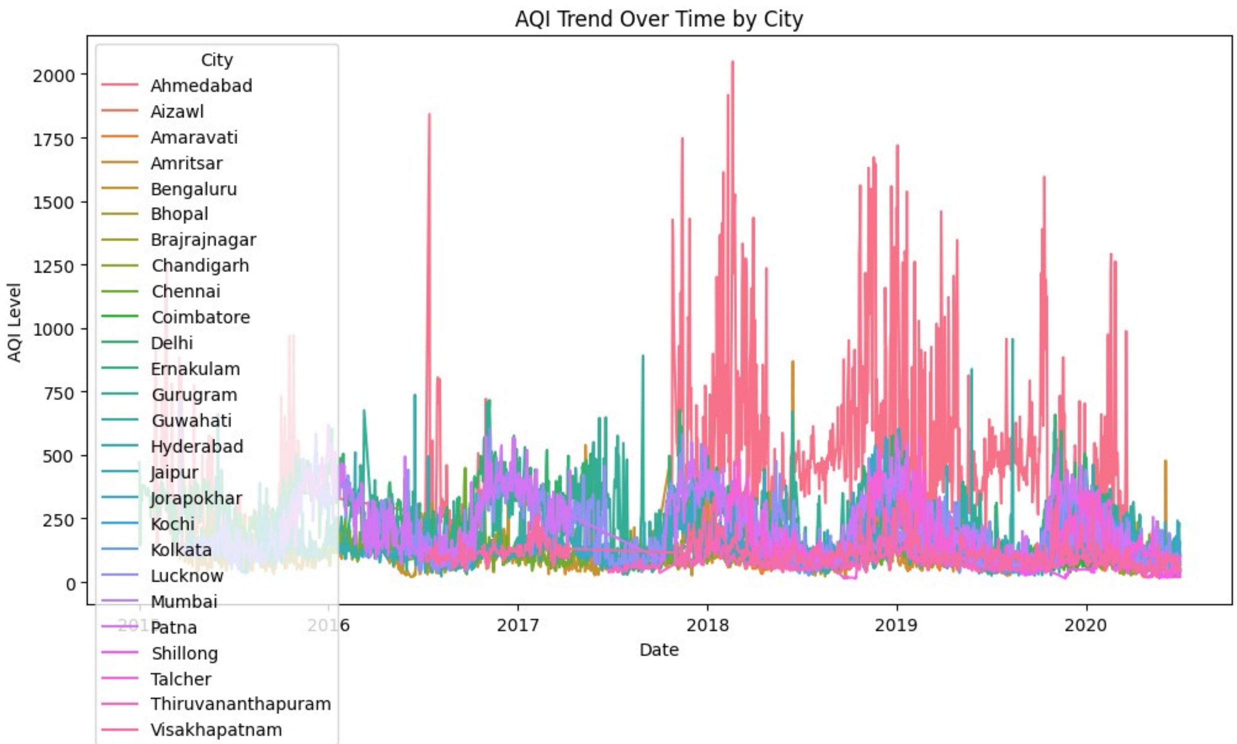
	State	AQI
0	Bihar	350.555556
1	Delhi	297.024658
2	Gujarat	310.950570
3	Karnataka	112.573427

```
4     Tamil Nadu  148.333333
5      Telangana  143.419118
6  Uttar Pradesh  202.235915


# Convert Date to datetime
cleandt['Date'] = pd.to_datetime(cleandt['Date'])

# Initial data exploration
print(cleandt.info())
print(cleandt.describe())

# Drop any missing values or fill them as appropriate
cleandt = cleandt.dropna()  # Adjust this step as necessary for your
data
```

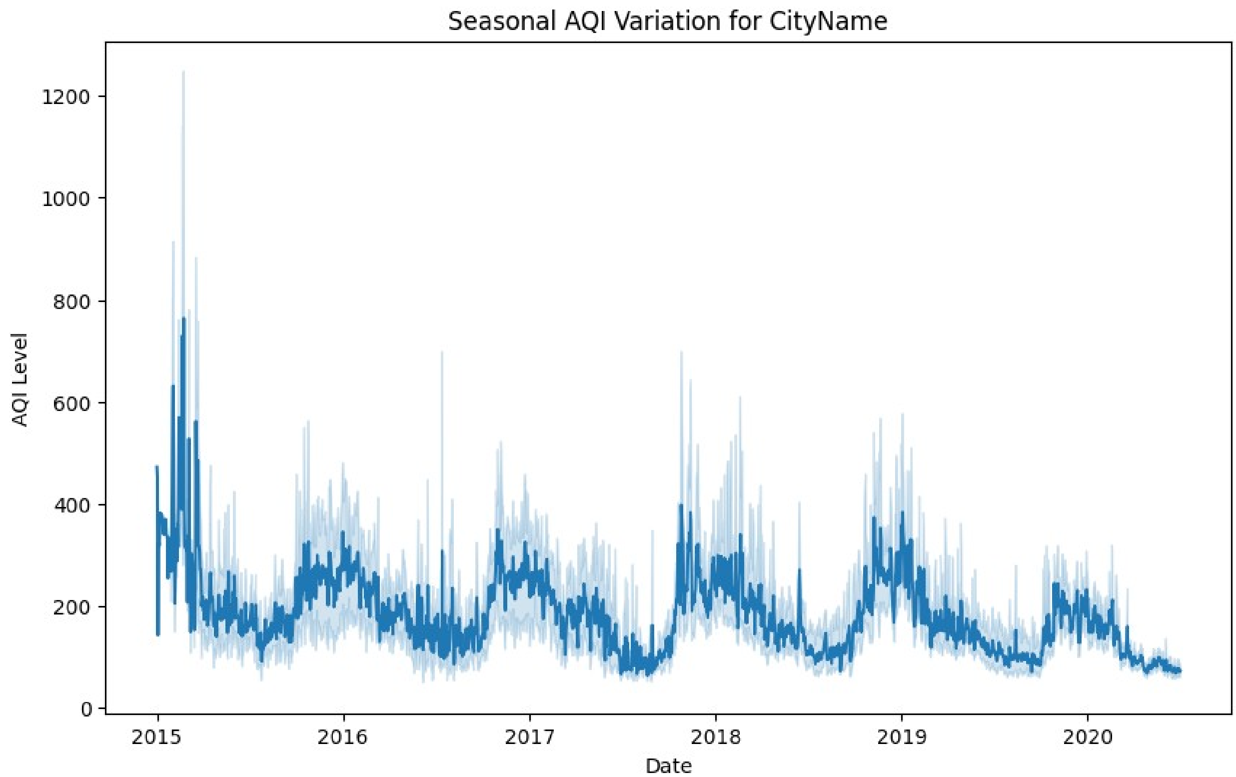
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24850 entries, 0 to 24849
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   City        24850 non-null  object
 1   Date        24850 non-null  datetime64[ns]
 2   PM2.5       24850 non-null  float64
 3   PM10        24850 non-null  float64
 4   NO          24850 non-null  float64
 5   NO2         24850 non-null  float64
 6   NOx         24850 non-null  float64
 7   NH3         24850 non-null  float64
 8   CO          24850 non-null  float64
 9   SO2         24850 non-null  float64
 10  O3          24850 non-null  float64
 11  Benzene     24850 non-null  float64
 12  Toluene     24850 non-null  float64
 13  Xylene      24850 non-null  float64
 14  AQI         24850 non-null  float64
 15  AQI_Bucket  24850 non-null  object
dtypes: datetime64[ns](1), float64(13), object(2)
memory usage: 3.0+ MB
None
                                Date         PM2.5          PM10  \
count                          24850  24850.000000  24850.000000
mean   2018-07-24 18:51:25.714285568     67.475903    118.361096
min              2015-01-01 00:00:00      0.040000      0.030000
25%              2017-08-16 00:00:00     29.560000     71.780000
50%              2018-11-05 00:00:00     50.165000    118.127103
75%              2019-10-11 00:00:00     79.507500    122.957500
max              2020-07-01 00:00:00    914.940000    917.080000
std                              NaN     62.208948     75.660501
```

```
                NO            NO2           NOx           NH3
CO  \
count  24850.000000  24850.000000  24850.000000  24850.000000
24850.000000
mean      17.621678     28.971818     32.290515     23.752394
2.343536
min        0.030000      0.010000      0.000000      0.010000
0.000000
25%        5.720000     12.090000     14.030000     11.280000
0.590000
50%       10.075000     22.535000     25.720000     23.483476
0.950000
75%       19.710000     37.910000     38.170000     24.710000
1.530000
max      390.680000    362.210000    378.240000    352.890000
175.810000
std       22.245860     24.432587     29.542968     22.214343
7.011582

                SO2            O3       Benzene       Toluene
Xylene  \
count  24850.000000  24850.000000  24850.000000  24850.000000
24850.000000
mean      14.367049     34.899199      3.433371      9.332356
3.267909
min        0.010000      0.010000      0.000000      0.000000
0.000000
25%        5.790000     19.640000      0.340000      1.580000
2.650000
50%        9.430000     32.060000      1.810000      6.790000
3.070128
75%       14.890000     45.397500      3.280840      8.700972
3.070128
max      186.080000    257.730000    455.030000    454.850000
170.370000
std       17.215237     21.368979     14.851776     18.273322
4.178816

                AQI
count  24850.000000
mean     166.463581
min       13.000000
25%       81.000000
50%      118.000000
75%      208.000000
max     2049.000000
std      140.696585

# Set Date as index for time series analysis
#cleandt.set_index('Date', inplace=True)
```

```
# Example: Plot AQI trends over time for each city
plt.figure(figsize=(12, 6))
sns.lineplot(data=cleandt, x=cleandt.index, y="AQI", hue="City")
plt.title("AQI Trend Over Time by City")
plt.xlabel("Date")
plt.ylabel("AQI Level")
plt.legend(title="City")
plt.show()
```

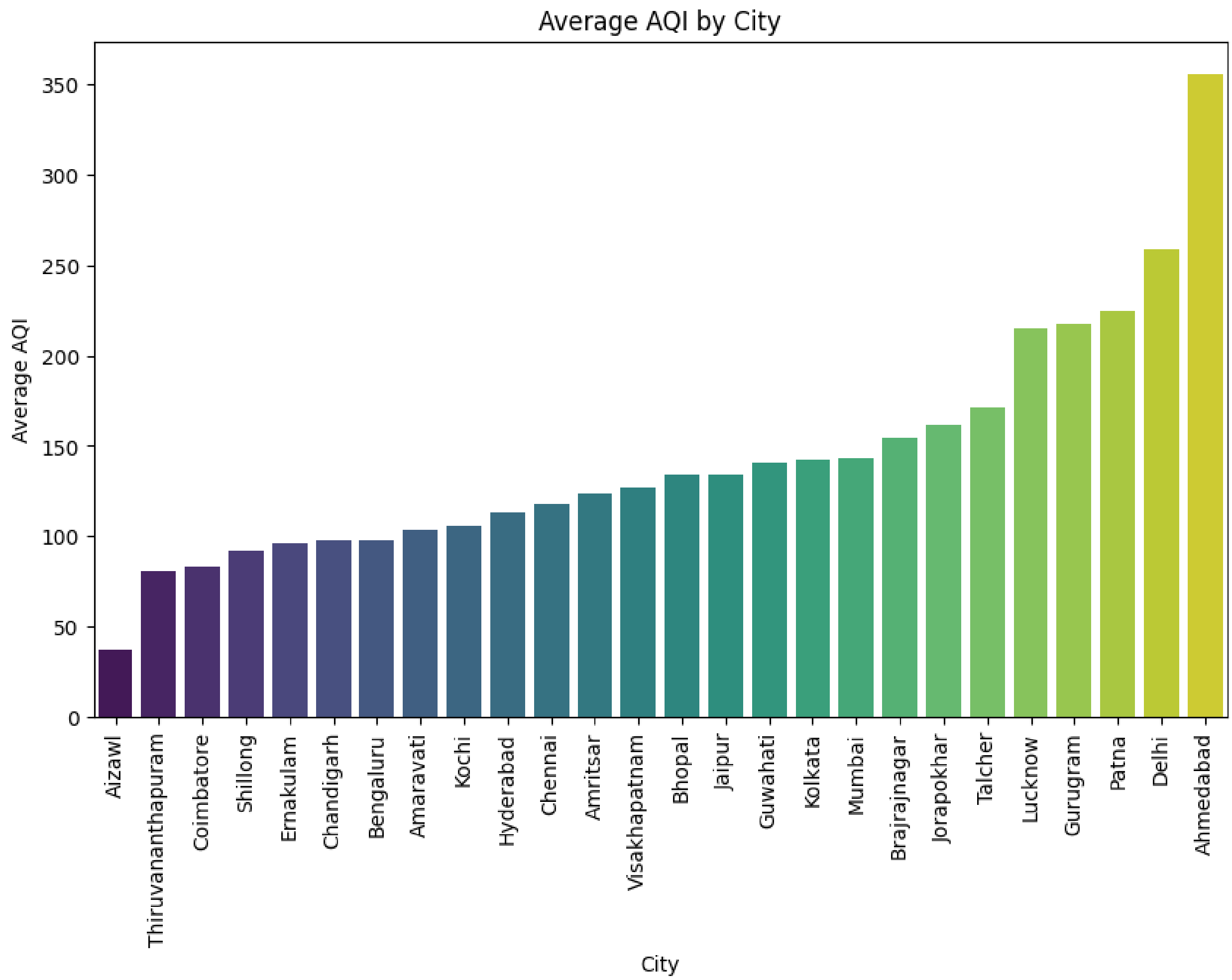



```
# Seasonal patterns for a particular city
#city_data = cleandt[cleandt['City'] == 'CityName']  # Replace with
actual city name
plt.figure(figsize=(10, 6))
sns.lineplot(x=cleandt.index, y=cleandt['AQI'])

plt.title("Seasonal AQI Variation for CityName")
plt.xlabel("Date")
plt.ylabel("AQI Level")
plt.show()
```

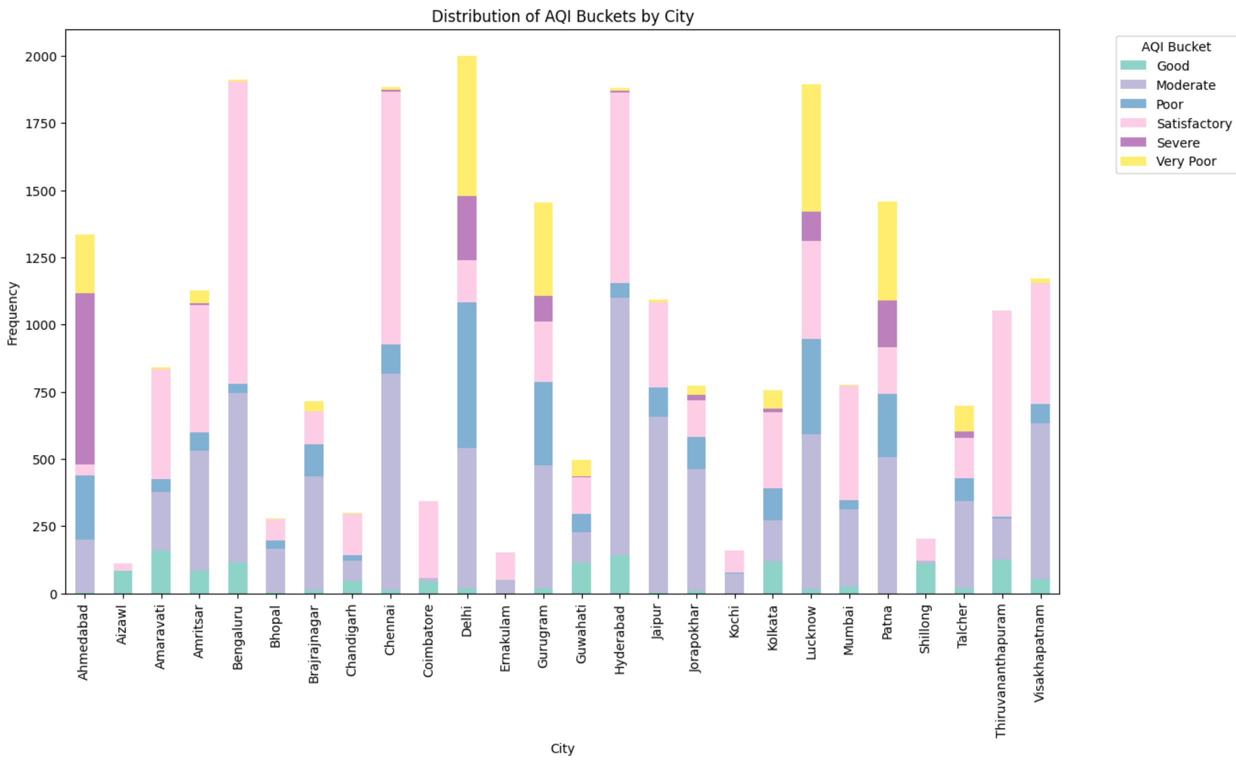



```
# Calculate average AQI by city
avg_aqi_by_city = data.groupby('City')['AQI'].mean().sort_values()

# Plot average AQI by city
plt.figure(figsize=(10, 6))
sns.barplot(x=avg_aqi_by_city.index, y=avg_aqi_by_city.values,
palette="viridis")
plt.xticks(rotation=90)
plt.title("Average AQI by City")
plt.xlabel("City")
plt.ylabel("Average AQI")
plt.show()
```

```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\600190197.py:6:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=avg_aqi_by_city.index, y=avg_aqi_by_city.values,
palette="viridis")
```

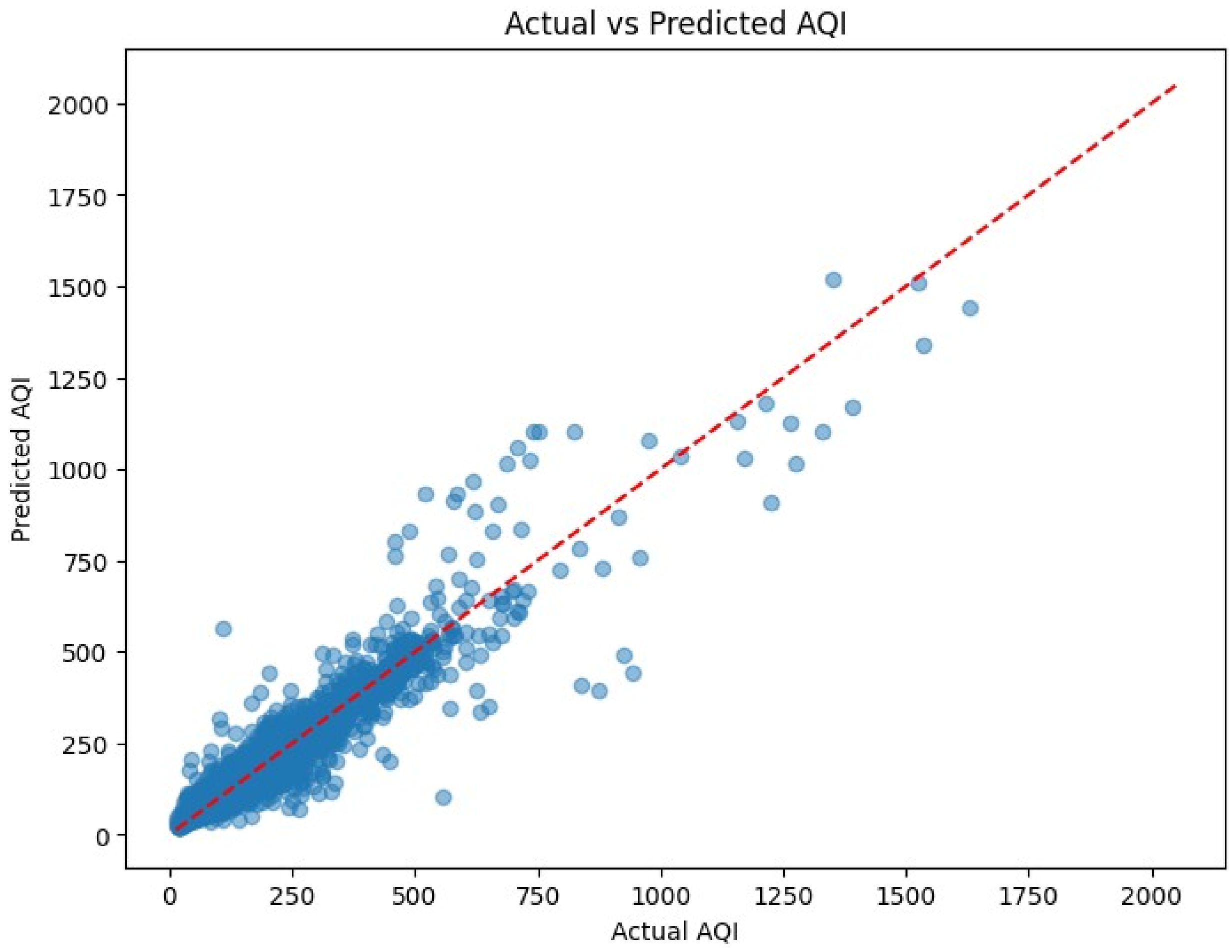



```
# Calculate distribution of AQI Buckets across cities
aqi_bucket_distribution = data.groupby(['City',
'AQI_Bucket']).size().unstack().fillna(0)

# Plot AQI Bucket distribution by city
aqi_bucket_distribution.plot(kind='bar', stacked=True, figsize=(14,
8), colormap="Set3")
plt.title("Distribution of AQI Buckets by City")
plt.xlabel("City")
plt.ylabel("Frequency")
plt.legend(title="AQI Bucket", bbox_to_anchor=(1.05, 1), loc='upper
left')
plt.show()

# Health impact assessment - percentage of "Unhealthy" AQI levels
unhealthy_percentage = (data['AQI_Bucket'] == 'Unhealthy').mean() *
100
print(f"Percentage of Unhealthy AQI Levels: {unhealthy_percentage:.2f}
%")
```




```
Percentage of Unhealthy AQI Levels: 0.00%

# Select features and target for prediction
# = cleandt[]  # Independent variables
from sklearn.ensemble import RandomForestRegressor
X = cleandt.drop(['City','AQI','AQI_Bucket'],axis=1)
y = cleandt['AQI']       # Target variable

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Model: Random Forest Regressor
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)

# Prediction
y_pred = model.predict(X_test)

# Model evaluation
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# Plot actual vs predicted AQI
plt.figure(figsize=(8, 6))
```

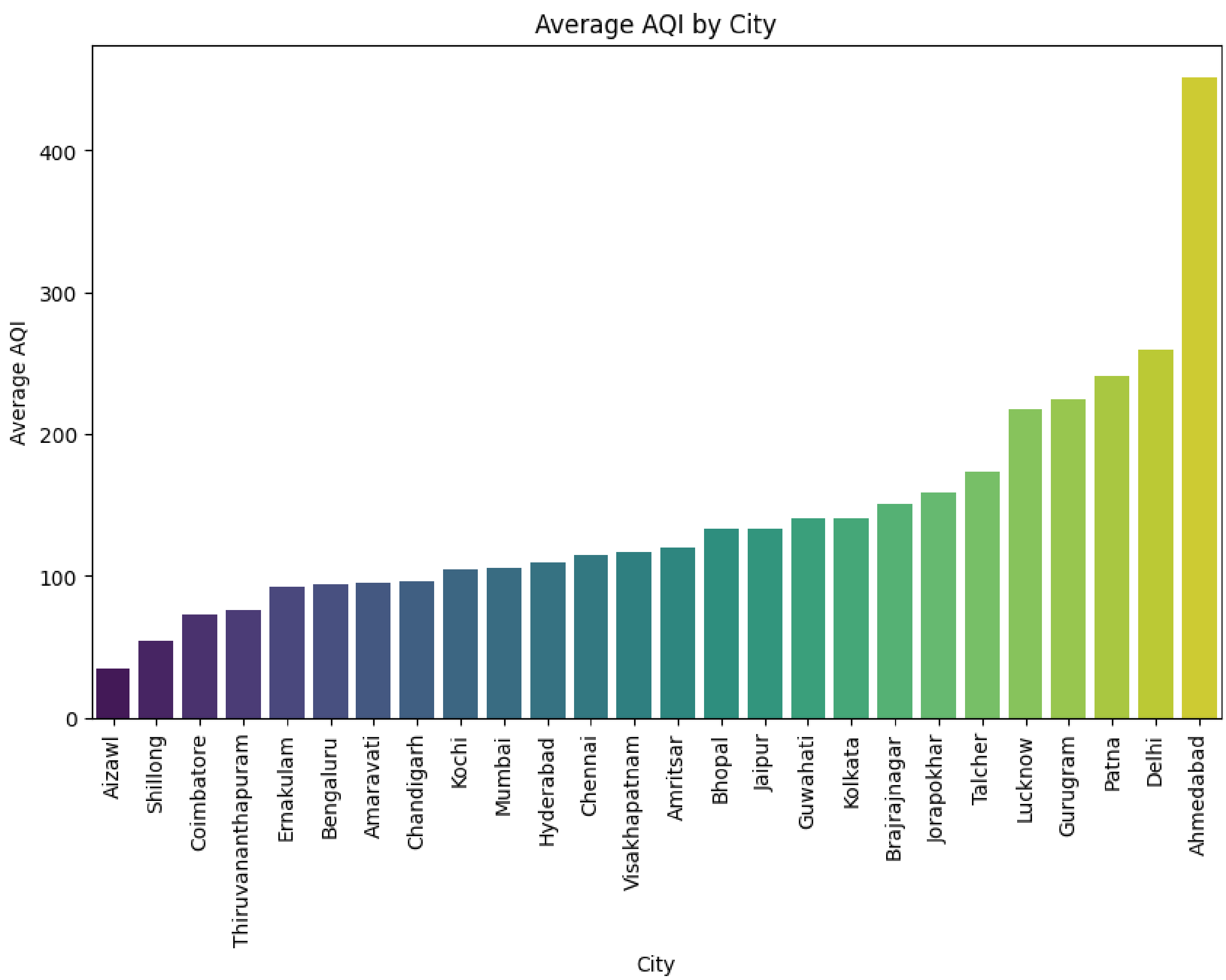
```
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')
plt.title("Actual vs Predicted AQI")
plt.xlabel("Actual AQI")
plt.ylabel("Predicted AQI")
plt.show()

Mean Squared Error: 1663.2689965301904
R-squared: 0.9091656886933218
```




```
# Calculate average AQI by city
avg_aqi_by_city = cleandt.groupby('City')['AQI'].mean().sort_values()

# Plot average AQI by city
plt.figure(figsize=(10, 6))
sns.barplot(x=avg_aqi_by_city.index, y=avg_aqi_by_city.values,
palette="viridis")
plt.xticks(rotation=90)
plt.title("Average AQI by City")
plt.xlabel("City")
```

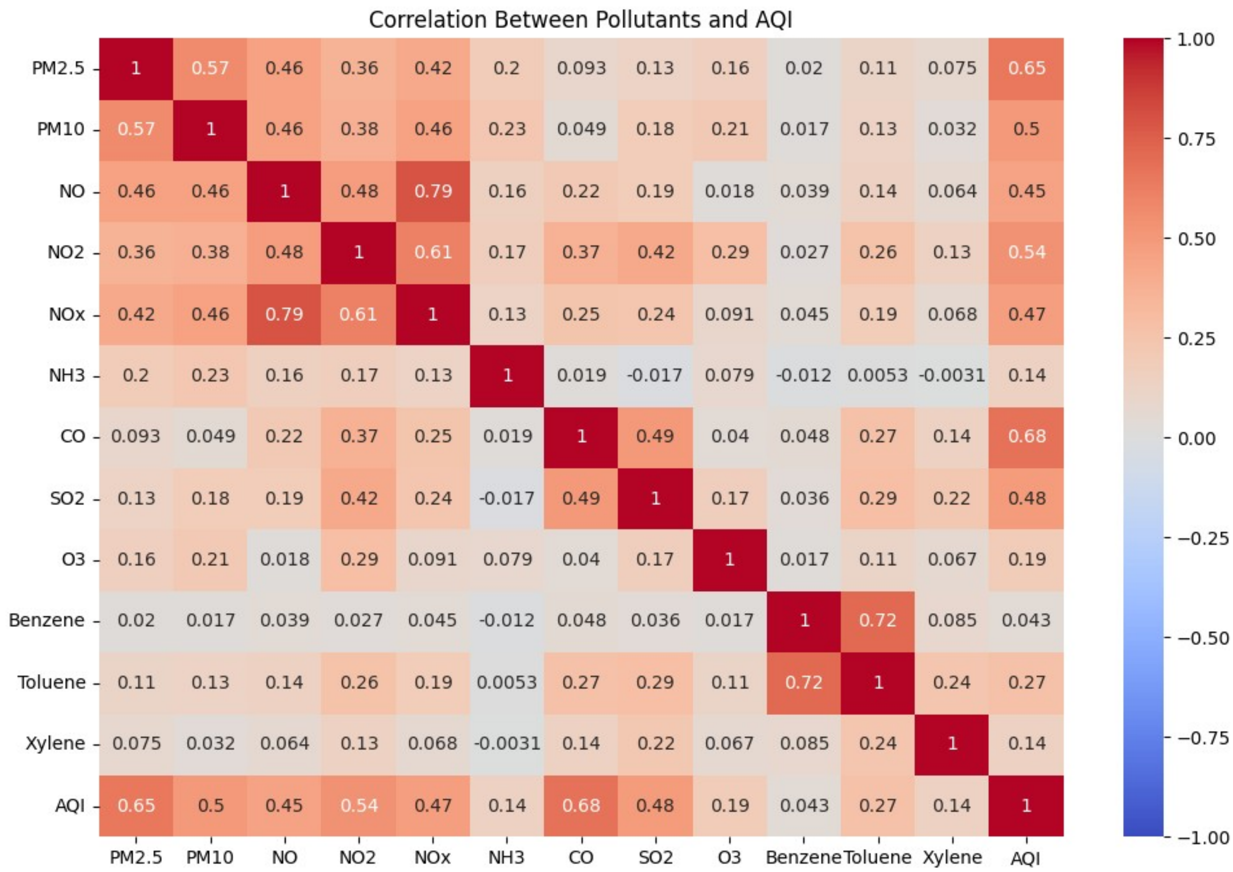
```
plt.ylabel("Average AQI")
plt.show()
```

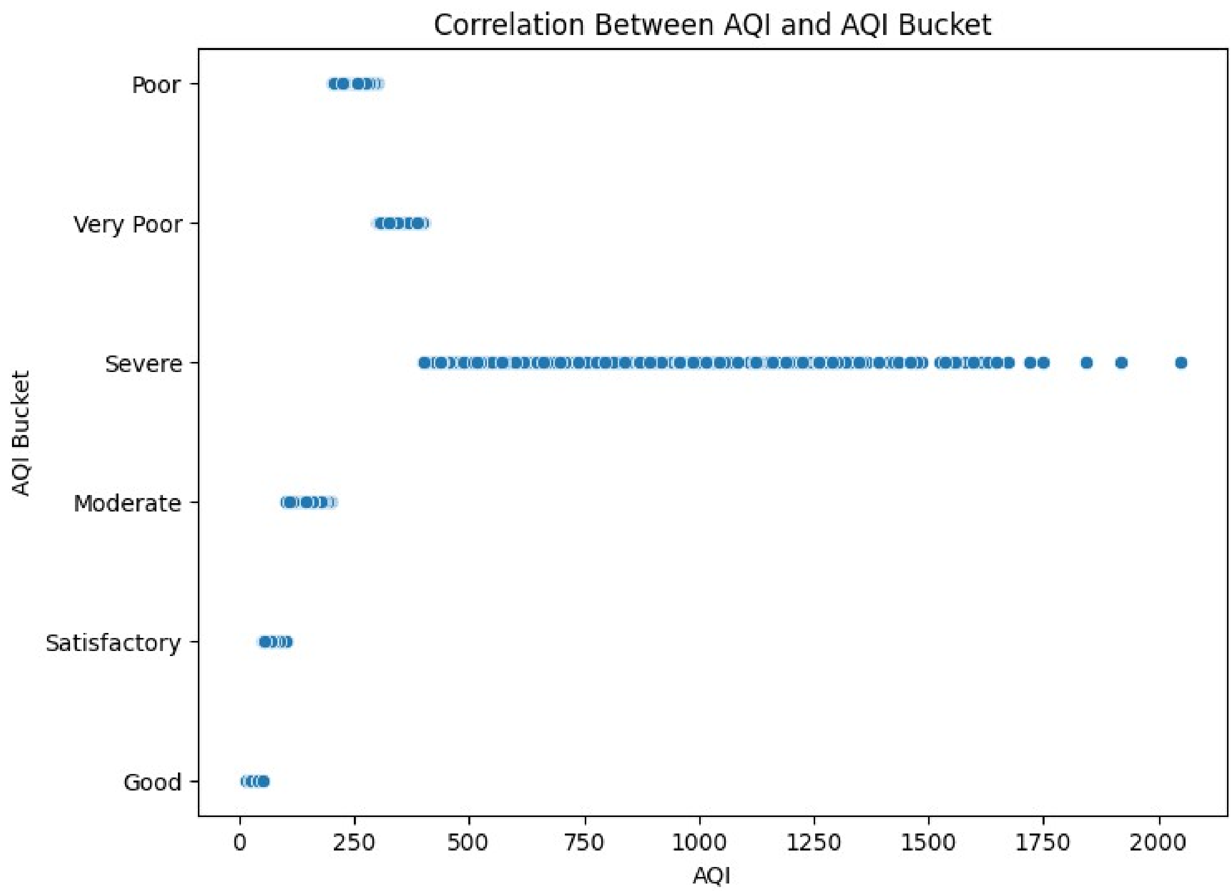
```
C:\Users\kadam\AppData\Local\Temp\ipykernel_21312\998449420.py:6:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=avg_aqi_by_city.index, y=avg_aqi_by_city.values,
palette="viridis")
```




```
# Correlation heatmap between pollutants and AQI
pollutants = ['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2',
'O3', 'Benzene', 'Toluene', 'Xylene']
plt.figure(figsize=(12, 8))
sns.heatmap(cleandt[pollutants + ['AQI']].corr(), annot=True,
cmap="coolwarm", vmin=-1, vmax=1)
```

```
plt.title("Correlation Between Pollutants and AQI")
plt.show()

# Correlation between AQI and AQI_Bucket (health impact)
data['AQI_Bucket'] = data['AQI_Bucket'].astype('category').cat.codes
# Convert AQI Bucket to numeric codes
plt.figure(figsize=(8, 6))
sns.scatterplot(data=cleandt, x="AQI", y="AQI_Bucket")
plt.title("Correlation Between AQI and AQI Bucket")
plt.xlabel("AQI")
plt.ylabel("AQI Bucket")
plt.show()
```




Correlation Between AQI and AQI Bucket
Poor
Very Poor
Severe
Moderate
Satisfactory
Good
AQI Bucket
0
250
500
750
1000
1250
1500
1750
2000
AQI