SKILL UP

IBM

AIR QUALITY ANALYSIS IN TAMIL NADU

DAC_Phase5

Team Members:

1. **Nithyasri V G**
2. **Sakshi Rajesh Bhavsar**
3. **Mathew P**
4. **Sarfraj Ansari**

AIR QUALITY ANALYSIS IN TAMIL NADU

Project Definition:

The primary objective of this project is to conduct a comprehensive assessment and visualization of air quality data collected from monitoring stations situated across Tamil Nadu. Through this project, we aim to extract valuable insights into the prevailing patterns of air pollution, identify regions exhibiting elevated levels of pollution, and establish a predictive model capable of estimating RSPM/PM10 levels. This analysis will be based on the concentrations of SO2 and NO2.

To achieve these objectives, the project encompasses several key phases. Firstly, we will precisely define the specific goals and targets we intend to accomplish through this study. Next, we will carefully devise the approach to be used in our analytical process. This will involve selecting appropriate statistical techniques and data processing methods. Additionally, we will thoughtfully choose visualization methods that effectively convey the patterns and trends identified in the air quality data. Finally, a predictive model will be developed using the Python programming language along with relevant libraries, allowing for accurate assessments of RSPM/PM10 levels based on the concentrations of SO2 and NO2. This comprehensive approach will enable us not only to ascertain the current state of air quality in Tamil Nadu but also to provide valuable insights for potential interventions and policy decisions aimed at mitigating air pollution.

Design Thinking:

Project Objectives:

Our objectives include analysing air quality trends, pinpointing pollution hotspots, and constructing a predictive model for RSPM/PM10 levels. We aspire to obtain comprehensive insights into the state of air pollution, thereby strengthening our capacity to make well-informed decisions concerning air quality management and the formulation of effective mitigation strategies.

Analysis Approach:

To plan the steps for loading, preprocessing, analyzing, and visualizing air quality data in Python, we follow this approach:

- **Import Relevant Libraries**:
  Begin by importing necessary libraries such as Pandas for data handling and Matplotlib/Seaborn for visualization.

- **Load Data:**
  Retrieve the air quality data from its source, which could be a CSV file or a database, using Pandas' data import functions.

- **Data Preprocessing:**
  In this stage, perform tasks like handling missing values, converting data types, and potentially scaling or normalizing numerical features.

- **Exploratory Data Analysis (EDA):**

  - ✓ Conducting exploratory data analysis to gain insights from the data.

  - ✓ Calculating descriptive statistics using functions like describe() in Pandas.

  - ✓ Creating visualizations with libraries like Matplotlib and Seaborn, including line graphs for trends, histograms for distributions, and scatter plots for relationships.

  - ✓ If necessary, performing statistical tests using modules like Scipy to uncover patterns and relationships.

- **Feature Selection:** Begin by identifying relevant features, such as SO2 and NO2 levels, that will be used in the modeling process.

- **Model Selection:** Choose appropriate regression or machine learning models for predicting RSPM/PM10 levels based on the selected features.

- **Model Training and Validation:** Split the data into training and testing sets, train the selected model, and validate its performance on the testing data

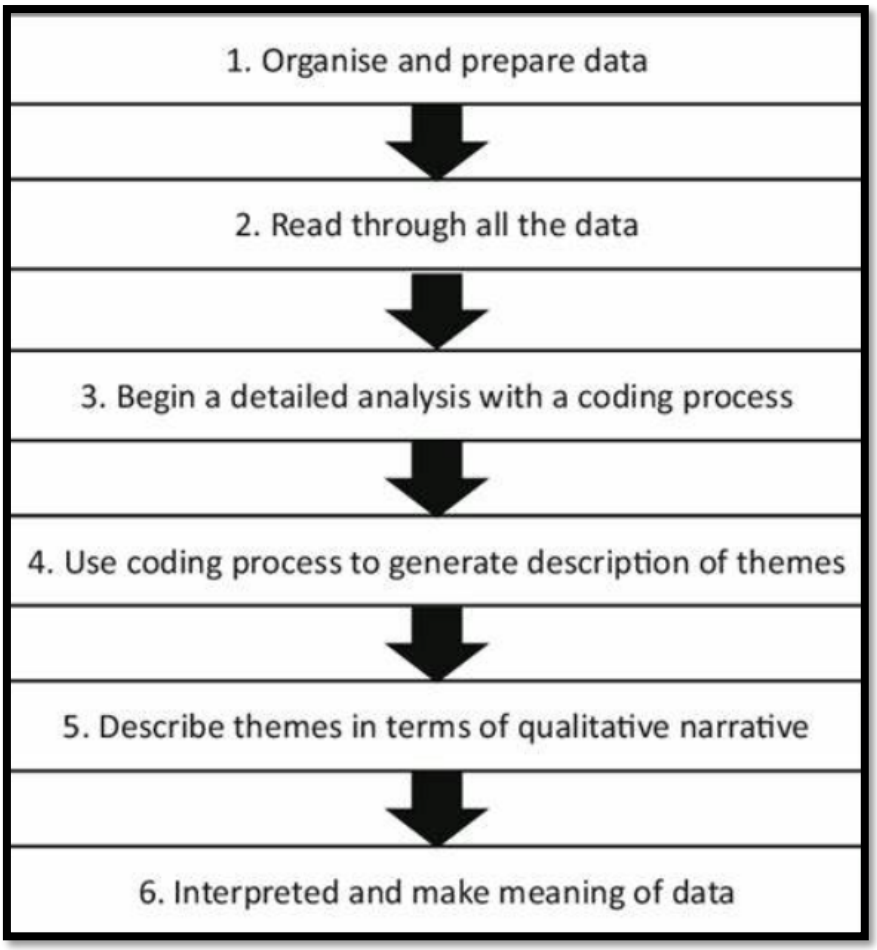
- **Final Reporting:** Ultimately, document and present the results of the analysis. This can be done using tools like Jupyter Notebook or by generating reports in formats such as PDF or HTML.

Visualization Selection:

Determining appropriate visualization techniques, such as line charts, heatmaps, scatter plots, and geographic maps, to effectively represent air quality trends and pollution levels. Additionally, considering radar charts, box plots, and bubble charts in specific contexts for a comprehensive understanding of air quality dynamics.

Dataset Link: https://tn.data.gov.in/resource/location-wise-daily-ambient-air-quality-tamil-nadu-year-2014

Case process diagram:



Python libraries:

Algorithm: Air Quality Data:

Import Relevant Libraries:

- import pandas as pd
- import numpy as np
- import matplotlib.pyplot as plt
- import seaborn as sns
- import scipy.stats as stats
- import geopandas as gpd
- from sklearn.model_selection import air_test_split
- from sklearn.linear_model import LinearRegression

Load Data:

Step 1: Import the pandas library

- import pandas as pd

Step 2: Read the CSV file

- df = pd.read_csv("air_quality_data.csv")

Step 3: Print the DataFrame

- print(df)

Data Preprocessing:

Handle missing values and outliers:

- df.dropna() # Drop rows with missing values
- new_df = df.drop(['State','Agency'],axis =1)
- new_df = df.dropna(subset=['SO2','NO2'])
- new_df = df.fillna(0)

Exploratory Data Analysis (EDA):

Input:- DataFrame (df) containing the dataset

Output:- Summary statistics, visualizations, and insights about the dataset

Descriptive Statistics:

Step1: Calculate basic statistics for each numerical column.

- df.describe()

Step2: Data Types and Missing Values- Identify data types and check for missing values:

- df.info()

Step3: Data Distribution Visualization

- import matplotlib.pyplot as plt
- import seaborn as sns
- # Create data visualizations
- sns.histplot(df['City/Town/Village/Area'])
- plt.show()

Data Transformation and Feature Engineering:

Create new features or transform existing ones to make the data more suitable for analysis. This might involve scaling, encoding categorical variables, or creating time-based features.

Step 1: This means that the categorical values in 'RSPM/PM10' are converted into binary columns.

- df_encoded = pd.get_dummies(df, columns=['RSPM/PM10'])
- print(df_encoded)

Step2: reads a CSV file, performs one-hot encoding on the specified column, and prints the resulting DataFrame with the encoded values.

Data Distribution Visualization

Step 1:

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

Step 2:

Box Plots:

- Identify outliers and distribution of numerical variables:
  - sns.boxplot(x='City/Town/Village/Area', y='RSPM/PM10', data=df)

Scatter Plots:

- Visualize relationships between two numerical variables:
  - #Scatter Plots for Correlation Analysis:
  - plt.scatter(df['SO2'], df['NO2'])

Machine Learning:

If your goal is predictive modeling or classification, you can use machine learning libraries such as Scikit-Learn to train and evaluate models.

Step 1: Import Libraries

- import pandas as pd: Import the pandas library for data manipulation
- from sklearn.linear_model import LinearRegression
- Import the LinearRegression model from scikit-learn

Step 2: Load and Prepare Data

- new_df = pd.read_csv("air_quality_data.csv"):
- Read the data from the CSV file "air_quality_data.csv" and store it in the DataFrame new_df.
- new_df = df.dropna(subset=['SO2','NO2','RSPM/PM10']):
- Remove rows with missing values in the columns 'SO2', 'NO2', and 'RSPM/PM10'.

Step 3: Define Independent and Dependent Variables

- X = new_df[['SO2']]: Define the independent variable X as the 'SO2' column.
- y = new_df['RSPM/PM10']: Define the dependent variable y as the 'RSPM/PM10' column.

Step 4: Create and Train the Linear Regression Model

- model = LinearRegression(): Create an instance of the LinearRegression model.
- model.fit(X, y): Train the model using the independent variable X and dependent variable y.

Step 5: Print Coefficients and Make Predictions

- print("Intercept (b0):", model.intercept_): Print the intercept (b0) of the regression line.
- print("Slope (b1):", model.coef_[0]): Print the slope (b1) of the regression line.
- X_new = [[5]]: Define a new value of 'SO2' (5 in this case) for prediction.
- y_pred = model.predict(X_new):
- Use the model to predict the corresponding 'RSPM/PM10' value.
- print("Predicted y for X =", X_new[0][0], ":", y_pred[0]): Print the predicted value.

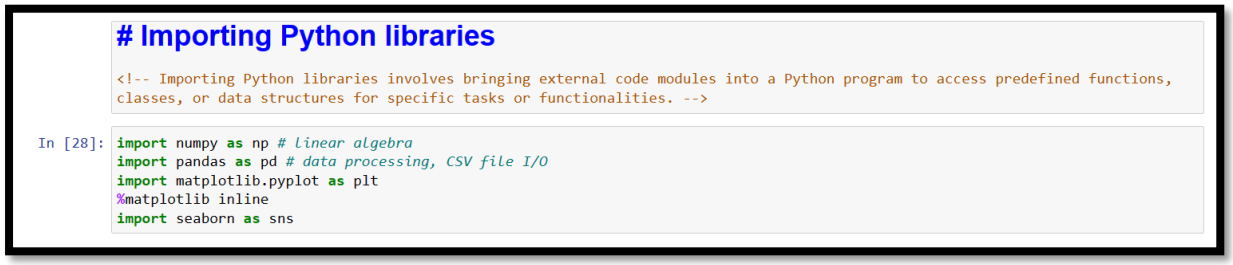
Visualizations and Reports:

- Creating estimating RSPM/PM10 levels with analysis will based on the concentrations of SO2 and NO2.

End.

Importing Python libraries

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```



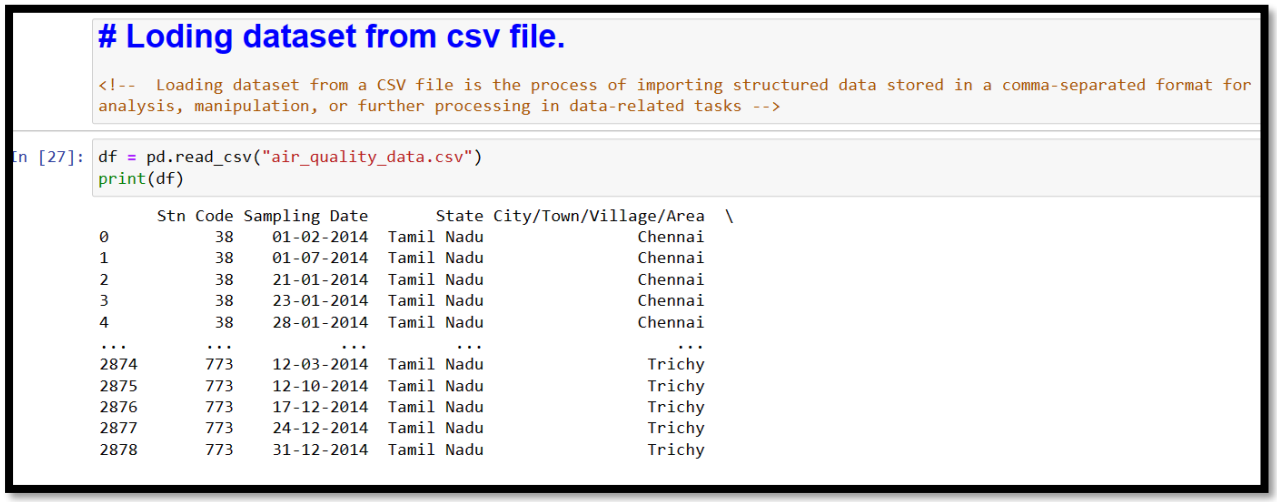
Importing Python libraries

<!-- Importing Python libraries involves bringing external code modules into a Python program to access predefined functions, classes, or data structures for specific tasks or functionalities. -->

In [28]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Loding dataset from csv file.

```
df = pd.read_csv("air_quality_data.csv")
print(df)
```



Loding dataset from csv file.

<!-- Loading dataset from a CSV file is the process of importing structured data stored in a comma-separated format for analysis, manipulation, or further processing in data-related tasks -->

In [27]:
```
df = pd.read_csv("air_quality_data.csv")
print(df)
```

```
      Stn Code Sampling Date       State City/Town/Village/Area  \
0           38    01-02-2014  Tamil Nadu                Chennai
1           38    01-07-2014  Tamil Nadu                Chennai
2           38    21-01-2014  Tamil Nadu                Chennai
3           38    23-01-2014  Tamil Nadu                Chennai
4           38    28-01-2014  Tamil Nadu                Chennai
...        ...           ...         ...                    ...
2874       773    12-03-2014  Tamil Nadu                 Trichy
2875       773    12-10-2014  Tamil Nadu                 Trichy
2876       773    17-12-2014  Tamil Nadu                 Trichy
2877       773    24-12-2014  Tamil Nadu                 Trichy
2878       773    31-12-2014  Tamil Nadu                 Trichy
```

```
                                          Agency  \
0     Tamilnadu State Pollution Control Board
1     Tamilnadu State Pollution Control Board
2     Tamilnadu State Pollution Control Board
3     Tamilnadu State Pollution Control Board
4     Tamilnadu State Pollution Control Board
...                                       ...
2874  Tamilnadu State Pollution Control Board
2875  Tamilnadu State Pollution Control Board
2876  Tamilnadu State Pollution Control Board
2877  Tamilnadu State Pollution Control Board
2878  Tamilnadu State Pollution Control Board

                        Type of Location   SO2   NO2  RSPM/PM10  PM 2.5
0                        Industrial Area  11.0  17.0       55.0     NaN
1                        Industrial Area  13.0  17.0       45.0     NaN
2                        Industrial Area  12.0  18.0       50.0     NaN
3                        Industrial Area  15.0  16.0       46.0     NaN
4                        Industrial Area  13.0  14.0       42.0     NaN
...                                  ...   ...   ...        ...     ...
2874  Residential, Rural and other Areas  15.0  18.0      102.0     NaN
2875  Residential, Rural and other Areas  12.0  14.0       91.0     NaN
2876  Residential, Rural and other Areas  19.0  22.0      100.0     NaN
2877  Residential, Rural and other Areas  15.0  17.0       95.0     NaN
2878  Residential, Rural and other Areas  14.0  16.0       94.0     NaN

[2879 rows x 11 columns]
```

df.shape

```
In [5]: # Representing the dimensions of a DataFrame.
        # It provides the number of rows and columns in the DataFrame.
        df.shape

Out[5]: (2879, 11)
```

Describing and defining imported dataset

print(df.head())

```
In [8]: print(df.head())

   Stn Code Sampling Date       State City/Town/Village/Area  \
0        38    01-02-2014  Tamil Nadu                Chennai
1        38    01-07-2014  Tamil Nadu                Chennai
2        38    21-01-2014  Tamil Nadu                Chennai
3        38    23-01-2014  Tamil Nadu                Chennai
4        38    28-01-2014  Tamil Nadu                Chennai

                     Location of Monitoring Station  \
0  Kathivakkam, Municipal Kalyana Mandapam, Chennai
1  Kathivakkam, Municipal Kalyana Mandapam, Chennai
2  Kathivakkam, Municipal Kalyana Mandapam, Chennai
3  Kathivakkam, Municipal Kalyana Mandapam, Chennai
4  Kathivakkam, Municipal Kalyana Mandapam, Chennai

                                  Agency Type of Location   SO2   NO2  \
0  Tamilnadu State Pollution Control Board  Industrial Area  11.0  17.0
1  Tamilnadu State Pollution Control Board  Industrial Area  13.0  17.0
2  Tamilnadu State Pollution Control Board  Industrial Area  12.0  18.0
3  Tamilnadu State Pollution Control Board  Industrial Area  15.0  16.0
4  Tamilnadu State Pollution Control Board  Industrial Area  13.0  14.0

   RSPM/PM10  PM 2.5
0       55.0     NaN
1       45.0     NaN
2       50.0     NaN
3       46.0     NaN
4       42.0     NaN
```

print(df.tail())

```
In [9]: print(df.tail())

      Stn Code Sampling Date       State City/Town/Village/Area  \
2874       773    12-03-2014  Tamil Nadu                 Trichy
2875       773    12-10-2014  Tamil Nadu                 Trichy
2876       773    17-12-2014  Tamil Nadu                 Trichy
2877       773    24-12-2014  Tamil Nadu                 Trichy
2878       773    31-12-2014  Tamil Nadu                 Trichy

     Location of Monitoring Station                                  Agency  \
2874      Central Bus Stand, Trichy  Tamilnadu State Pollution Control Board
2875      Central Bus Stand, Trichy  Tamilnadu State Pollution Control Board
2876      Central Bus Stand, Trichy  Tamilnadu State Pollution Control Board
2877      Central Bus Stand, Trichy  Tamilnadu State Pollution Control Board
2878      Central Bus Stand, Trichy  Tamilnadu State Pollution Control Board

                        Type of Location   SO2   NO2  RSPM/PM10  PM 2.5
2874  Residential, Rural and other Areas  15.0  18.0      102.0     NaN
2875  Residential, Rural and other Areas  12.0  14.0       91.0     NaN
2876  Residential, Rural and other Areas  19.0  22.0      100.0     NaN
2877  Residential, Rural and other Areas  15.0  17.0       95.0     NaN
2878  Residential, Rural and other Areas  14.0  16.0       94.0     NaN
```

df.describe()

```
In [10]: df.describe()
```

Out[10]:

	Stn Code	SO2	NO2	RSPM/PM10	PM 2.5
count	2879.000000	2868.000000	2866.000000	2875.000000	0.0
mean	475.750261	11.503138	22.136776	62.494261	NaN
std	277.675577	5.051702	7.128694	31.368745	NaN
min	38.000000	2.000000	5.000000	12.000000	NaN
25%	238.000000	8.000000	17.000000	41.000000	NaN
50%	366.000000	12.000000	22.000000	55.000000	NaN
75%	764.000000	15.000000	25.000000	78.000000	NaN
max	773.000000	49.000000	71.000000	269.000000	NaN

df.info()

```
In [11]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2879 entries, 0 to 2878
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Stn Code                        2879 non-null   int64
 1   Sampling Date                   2879 non-null   object
 2   State                           2879 non-null   object
 3   City/Town/Village/Area          2879 non-null   object
 4   Location of Monitoring Station  2879 non-null   object
 5   Agency                          2879 non-null   object
 6   Type of Location                2879 non-null   object
 7   SO2                             2868 non-null   float64
 8   NO2                             2866 non-null   float64
 9   RSPM/PM10                       2875 non-null   float64
 10  PM 2.5                          0 non-null      float64
dtypes: float64(4), int64(1), object(6)
memory usage: 247.5+ KB
```

Data preprocessing

```
import pandas as pd

df = pd.read_csv("air_quality_data.csv")

df.drop(labels=['Stn Code','Location of Monitoring Station','Agency'], axis = 1, inplace = True)

df.sample(5)
```

Data preprocessing

<!-- Data preprocessing: Preparing raw data for analysis by cleaning, transforming, and organizing it to enhance quality, relevance, and usability. -->

```
In [71]: import pandas as pd
df = pd.read_csv("air_quality_data.csv")
df.drop(labels=['Stn Code','Location of Monitoring Station','Agency'], axis = 1, inplace = True)
df.sample(5)
```

Out[71]:

	Sampling Date	State	City/Town/Village/Area	Type of Location	SO2	NO2	RSPM/PM10	PM 2.5
40	06-12-2014	Tamil Nadu	Chennai	Industrial Area	14.0	18.0	39.0	NaN
1735	07-05-2014	Tamil Nadu	Madurai	Industrial Area	9.0	24.0	35.0	NaN
1990	21-01-2014	Tamil Nadu	Mettur	Industrial Area	10.0	27.0	67.0	NaN
477	09-11-2014	Tamil Nadu	Chennai	Residential, Rural and other Areas	14.0	27.0	64.0	NaN
2046	08-07-2014	Tamil Nadu	Mettur	Industrial Area	9.0	69.0	44.0	NaN

df.isnull().sum()

```
In [45]: df.isnull().sum()

Out[45]: Sampling Date                 0
         State                         0
         City/Town/Village/Area        0
         Type of Location              0
         SO2                          11
         NO2                          13
         RSPM/PM10                     4
         PM 2.5                     2879
         dtype: int64
```

Omitting PM 2.5 column from the dataset:

```
import pandas as pd

df = pd.read_csv("air_quality_data.csv")

new_df=df.drop(labels=['PM 2.5'], axis = 1, inplace = True)

new_df=df.sample(2)
```

```
In [47]: # PM 2.5 has 2879 data missing. So omitting PM 2.5 column from the dataset.
         import pandas as pd
         df = pd.read_csv("air_quality_data.csv")
         new_df=df.drop(labels=['PM 2.5'], axis = 1, inplace = True)
         new_df=df.sample(2)

In [9]: print(new_df)

      Stn Code Sampling Date       State City/Town/Village/Area  \
2052       763    26-08-2014  Tamil Nadu                Mettur
493        765    11-12-2014  Tamil Nadu               Chennai

        Location of Monitoring Station  \
2052  SIDCO Industrial Complex, Mettur
493               Anna Nagar, Chennai

                                    Agency  \
2052  Tamilnadu State Pollution Control Board
493   Tamilnadu State Pollution Control Board

                      Type of Location   SO2   NO2  RSPM/PM10
2052                   Industrial Area   9.0  24.0       44.0
493   Residential, Rural and other Areas  15.0  26.0       62.0
```

Fill the missing values:

df=new_df

df.dtypes

```
In [10]: # In order to fill the missing values:
         # the values are first need to be sorted in Chronological order
         df=new_df
         df.dtypes

Out[10]: Stn Code                             int64
         Sampling Date                       object
         State                               object
         City/Town/Village/Area              object
         Location of Monitoring Station      object
         Agency                              object
         Type of Location                    object
         SO2                                float64
         NO2                                float64
         RSPM/PM10                          float64
         dtype: object
```

Converting "object" data type to "datetime"

df['Sampling Date'] = pd.to_datetime(df['Sampling Date'],format='%d-%m-%Y')

df.info()

```
In [12]: # To sort based on dates, the date should be of "datetime" datatype.
         #So converting "object" data type to "datetime" datatype
         df['Sampling Date'] = pd.to_datetime(df['Sampling Date'],format='%d-%m-%Y')
         df.info()

         <class 'pandas.core.frame.DataFrame'>
         Index: 2 entries, 2052 to 493
         Data columns (total 10 columns):
          #   Column                          Non-Null Count  Dtype
         ---  ------                          --------------  -----
          0   Stn Code                        2 non-null      int64
          1   Sampling Date                   2 non-null      datetime64[ns]
          2   State                           2 non-null      object
          3   City/Town/Village/Area          2 non-null      object
          4   Location of Monitoring Station  2 non-null      object
          5   Agency                          2 non-null      object
          6   Type of Location                2 non-null      object
          7   SO2                             2 non-null      float64
          8   NO2                             2 non-null      float64
          9   RSPM/PM10                       2 non-null      float64
         dtypes: datetime64[ns](1), float64(3), int64(1), object(5)
         memory usage: 176.0+ bytes
```

df.sort_values(by='Sampling Date')

```
In [48]: df.sort_values(by='Sampling Date')
```

Out[48]:

	Stn Code	Sampling Date	State	City/Town/Village/Area	Location of Monitoring Station	Agency	Type of Location	SO2	NO2	RSPM/PM10
0	38	01-02-2014	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	11.0	17.0	55.0
2512	769	01-02-2014	Tamil Nadu	Trichy	Gandhi Market, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	16.0	19.0	102.0
508	764	01-02-2014	Tamil Nadu	Chennai	Adyar, Chennai	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	15.0	19.0	61.0
624	767	01-02-2014	Tamil Nadu	Chennai	Kilpauk, Chennai	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	18.0	23.0	83.0
2661	771	01-02-2014	Tamil Nadu	Trichy	Bishop Heber College, Tirchy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	11.0	14.0	53.0
...	...	...	...	...	...	...	...	...	...	...
623	764	31-12-2014	Tamil Nadu	Chennai	Adyar, Chennai	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	14.0	20.0	42.0
1785	307	31-12-2014	Tamil Nadu	Madurai	Fenner (I) Ltd. Employees Assiciation Building...	Tamilnadu State Pollution Control Board	Industrial Area	13.0	26.0	25.0
1588	760	31-12-2014	Tamil Nadu	Cuddalore	SIPCOT Industrial Complex, Cuddalore	Tamilnadu State Pollution Control Board	Industrial Area	6.0	17.0	44.0
2218	309	31-12-2014	Tamil Nadu	Salem	Sowdeswari College Building, Salem	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	8.0	29.0	57.0
2878	773	31-12-2014	Tamil Nadu	Trichy	Central Bus Stand, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	14.0	16.0	94.0

Handle missing data:

```
# data preprocessing to handle missing data
new_df['SO2'].fillna(method='ffill',inplace = True);
new_df['NO2'].fillna(method='ffill',inplace = True);
new_df['RSPM/PM10'].fillna(method='ffill',inplace = True);
print(df.head(2))
```

```
In [14]: # data preprocessing to handle missing data
         new_df['SO2'].fillna(method='ffill',inplace = True);
         new_df['NO2'].fillna(method='ffill',inplace = True);
         new_df['RSPM/PM10'].fillna(method='ffill',inplace = True);
         print(df.head(2))

      Stn Code Sampling Date       State City/Town/Village/Area  \
2052       763    2014-08-26  Tamil Nadu                Mettur
493        765    2014-12-11  Tamil Nadu                Chennai

        Location of Monitoring Station  \
2052  SIDCO Industrial Complex, Mettur
493               Anna Nagar, Chennai

                                    Agency  \
2052  Tamilnadu State Pollution Control Board
493   Tamilnadu State Pollution Control Board

                       Type of Location   SO2   NO2  RSPM/PM10
2052                     Industrial Area   9.0  24.0       44.0
493   Residential, Rural and other Areas  15.0  26.0       62.0
```

df.isnull().sum()

```
In [71]: df.isnull().sum()

Out[71]: Stn Code                         0
         Sampling Date                    0
         State                            0
         City/Town/Village/Area           0
         Location of Monitoring Station   0
         Agency                           0
         Type of Location                 0
         SO2                              0
         NO2                              0
         RSPM/PM10                        0
         dtype: int64
```

Finding hidden missing values:

missing_val= (df == 0).astype(int).sum(axis=0)

print(missing_val)

```
In [51]: #Finding hidden missing values; eg:0
         missing_val= (df == 0).astype(int).sum(axis=0)
         print(missing_val)

         Stn Code                         0
         Sampling Date                    0
         State                            0
         City/Town/Village/Area           0
         Location of Monitoring Station   0
         Agency                           0
         Type of Location                 0
         SO2                              0
         NO2                              0
         RSPM/PM10                        0
         dtype: int64
```

Chronological order:

df.dtypes

```
In [19]: # sorted in Chronological order
         df.dtypes

Out[19]: Stn Code                             int64
         Sampling Date                datetime64[ns]
         State                               object
         City/Town/Village/Area              object
         Location of Monitoring Station      object
         Agency                              object
         Type of Location                    object
         SO2                                float64
         NO2                                float64
         RSPM/PM10                          float64
         dtype: object
```

Sorting the date wise:

df.sort_values(by='Sampling Date')

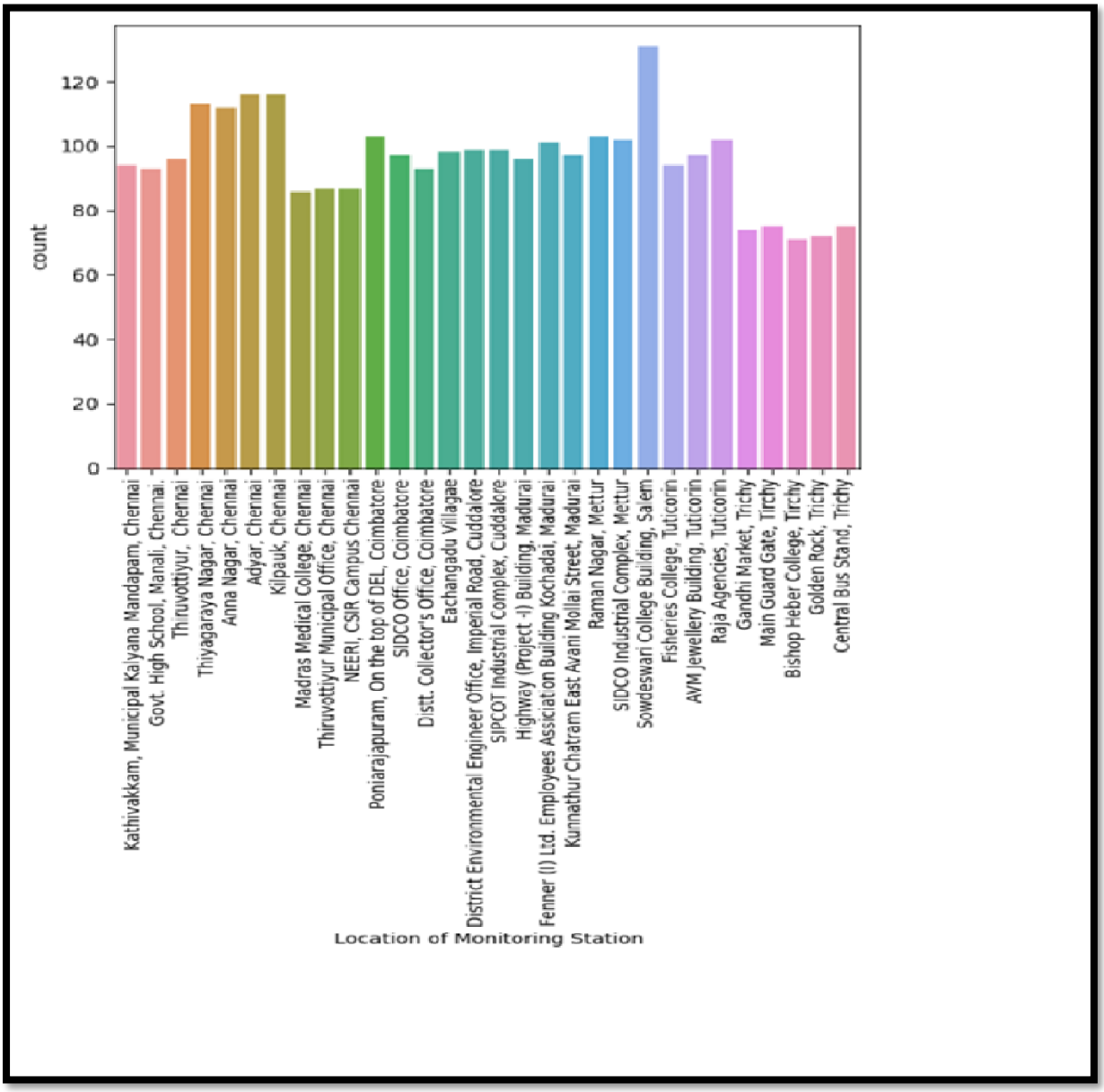
```
In [77]: # sorting the date wise
         df.sort_values(by='Sampling Date')

Out[77]:
```

	Stn Code	Sampling Date	State	City/Town/Village/Area	Location of Monitoring Station	Agency	Type of Location	SO2	NO2	RSPM/PM10
1151	237	2014-06-27	Tamil Nadu	Coimbatore	SIDCO Office, Coimbatore	Tamilnadu State Pollution Control Board	Industrial Area	3.0	29.0	39.0
1643	306	2014-07-28	Tamil Nadu	Madurai	Highway (Project -I) Building, Madurai	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	10.0	27.0	43.0

Modifying the insights data:

```
import pandas as pd

new_df = pd.read_csv("air_quality_data.csv")

datacount =sns.countplot(x ="Location of Monitoring Station",data =new_df);

datacount.set_xticklabels(datacount.get_xticklabels(), rotation=90);
```

```
In [29]: # Data Visualization
         # Modifying the insights datas
         import pandas as pd
         new_df = pd.read_csv("air_quality_data.csv")
         datacount =sns.countplot(x ="Location of Monitoring Station",data =new_df);
         datacount.set_xticklabels(datacount.get_xticklabels(), rotation=90);
```



Type of Location:

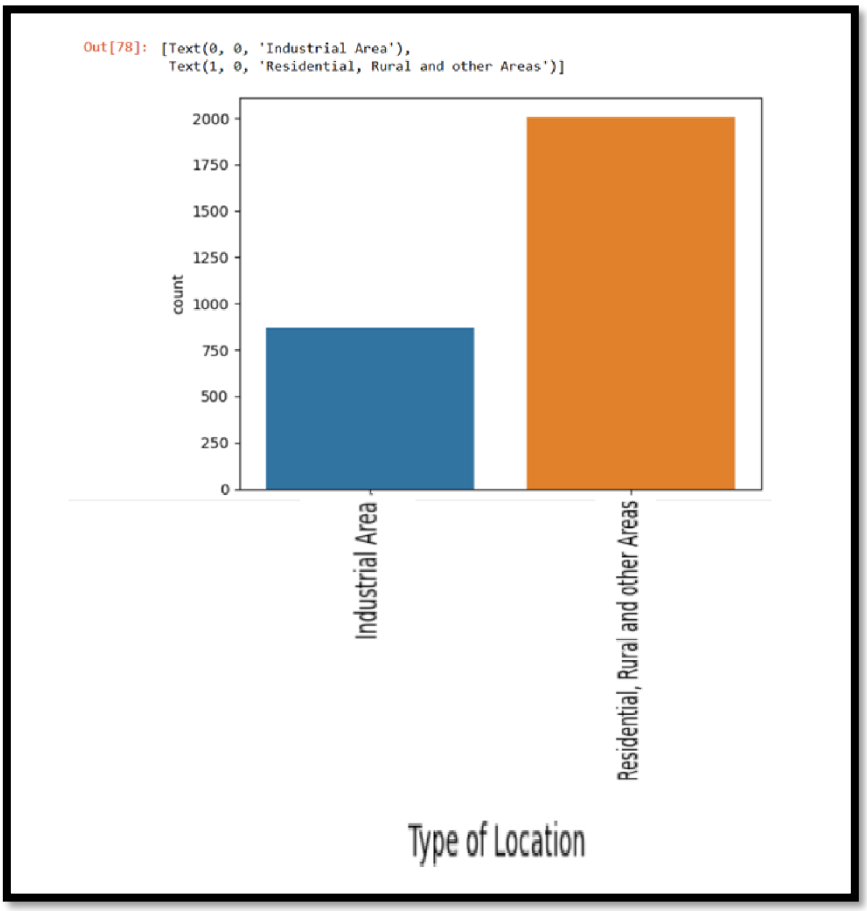
df['Type of Location'].unique()

```
In [24]: # Type of Location
         df['Type of Location'].unique()

Out[24]: array(['Industrial Area', 'Residential, Rural and other Areas'],
               dtype=object)
```

Type of Location as Industrial & Residential area

```
import pandas as pd
new_df= pd.read_csv("air_quality_data.csv")
typ=sns.countplot(x ="Type of Location",data =new_df)
typ.set_xticklabels(typ.get_xticklabels(), rotation=90)
```

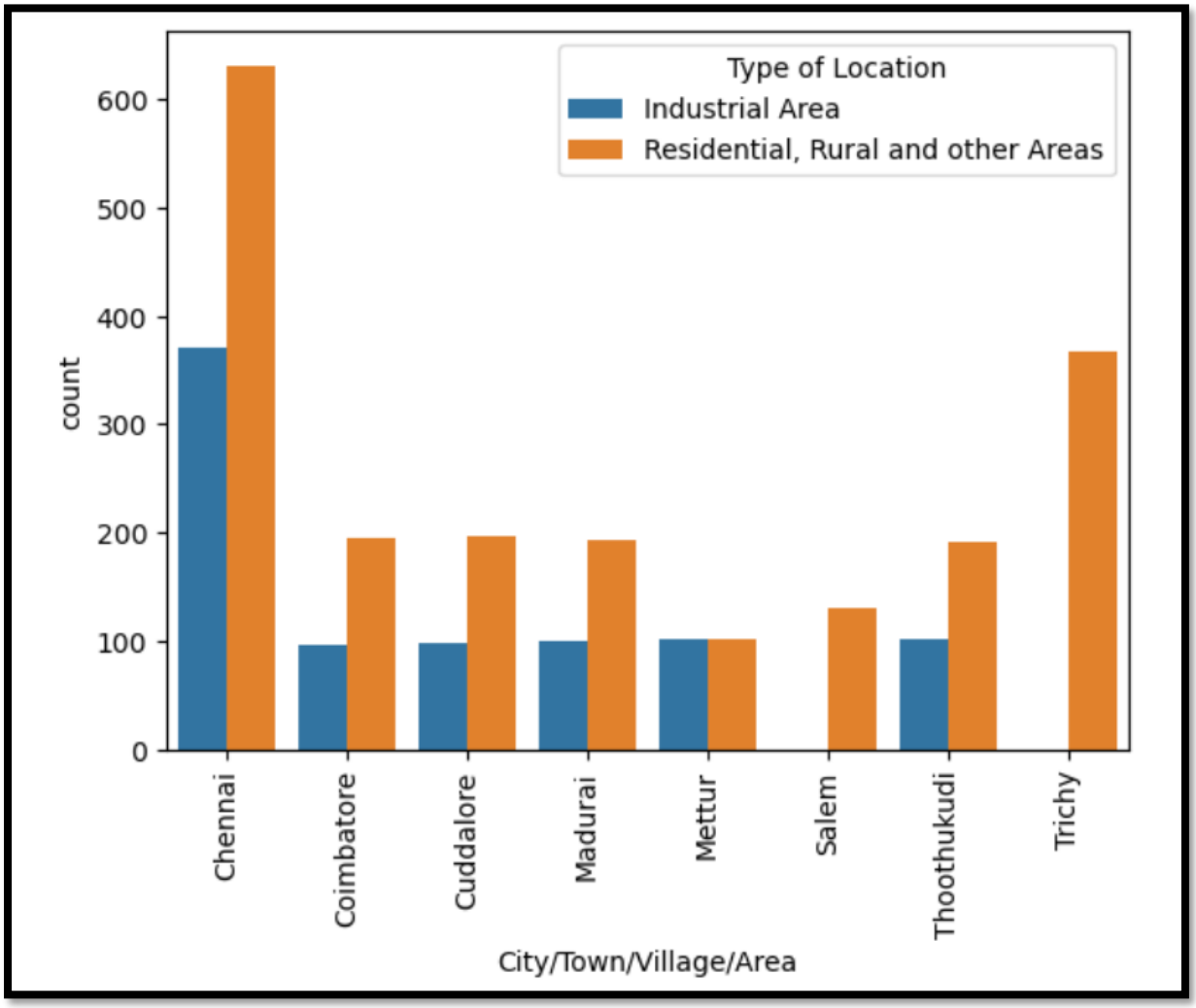
```
In [78]: import pandas as pd
         new_df= pd.read_csv("air_quality_data.csv")
         typ=sns.countplot(x ="Type of Location",data =new_df)
         typ.set_xticklabels(typ.get_xticklabels(), rotation=90)
```



City area splitting Type of Location as Industrial & Residential area

```
datacount_ty =sns.countplot(x ='City/Town/Village/Area',hue ='Type of Location',data = df);
datacount_ty.set_xticklabels(datacount_ty.get_xticklabels(), rotation=90);
```

```
In [160]: datacount_ty =sns.countplot(x ='City/Town/Village/Area',hue ='Type of Location',data = df);
          datacount_ty.set_xticklabels(datacount_ty.get_xticklabels(), rotation=90);
```

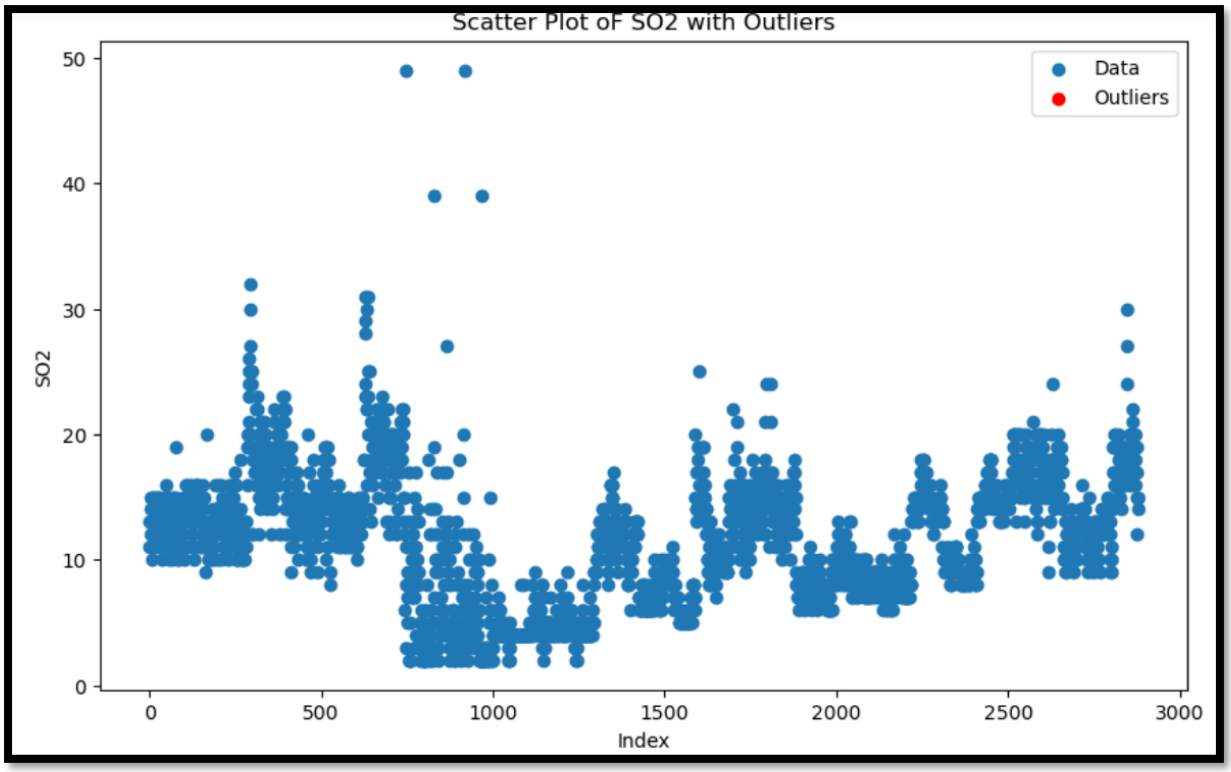


Checking of Outliers:

Scatter Plot of SO2 with Outliers:

```
from scipy.stats import zscore

column_name = 'SO2'

column_data = new_df[column_name]

z_scores = zscore(column_data)

threshold = 3

outliers = (z_scores > threshold) | (z_scores < -threshold)

# scatter plot

plt.figure(figsize=(10, 6))

plt.scatter(new_df.index, column_data, label='Data')

plt.scatter(new_df.index[outliers], column_data[outliers], color='red', label='Outliers')

plt.xlabel('Index')

plt.ylabel(column_name)

plt.title('Scatter Plot oF SO2 with Outliers')

plt.legend()

plt.show()
```

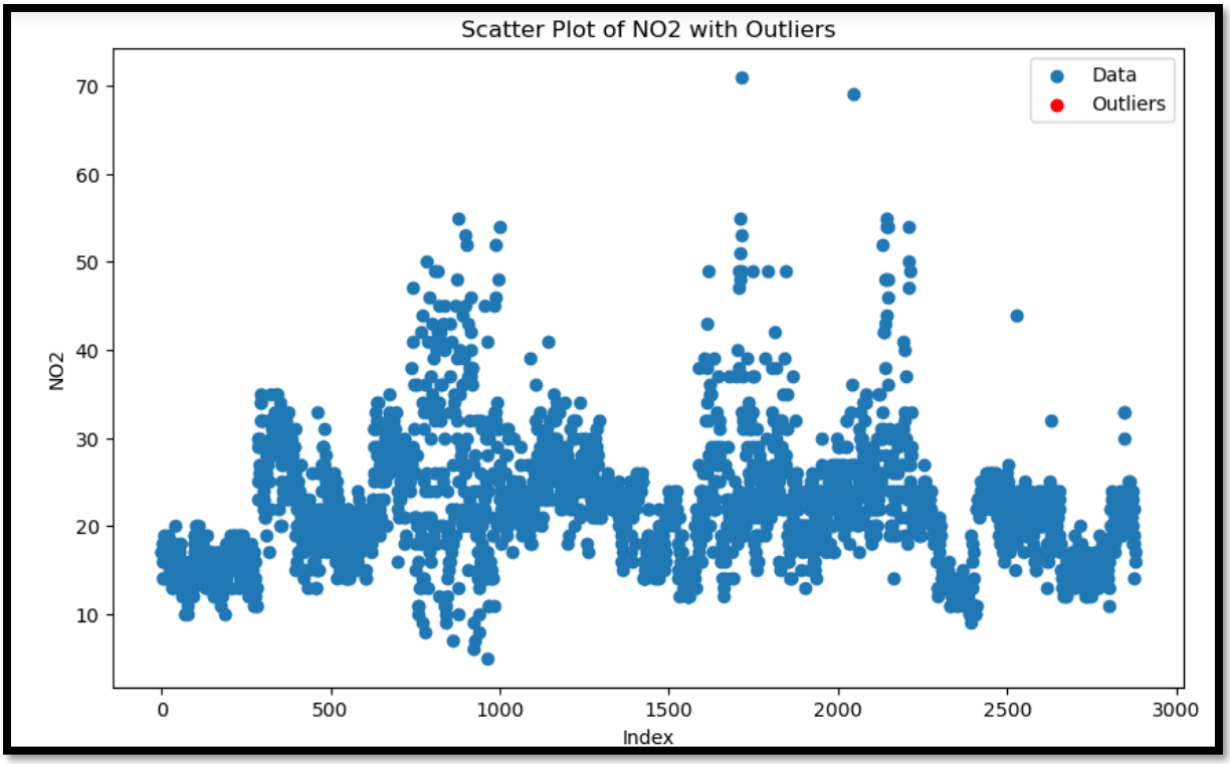
In [31]:
```
# Checking of Outliers
# SO2
from scipy.stats import zscore
column_name = 'SO2'
column_data = new_df[column_name]
z_scores = zscore(column_data)
threshold = 3
outliers = (z_scores > threshold) | (z_scores < -threshold)
# scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(new_df.index, column_data, label='Data')
plt.scatter(new_df.index[outliers], column_data[outliers], color='red', label='Outliers')
plt.xlabel('Index')
plt.ylabel(column_name)
plt.title('Scatter Plot oF SO2 with Outliers')
plt.legend()
plt.show()
```



Scatter Plot of NO2 with Outliers:

```
from scipy.stats import zscore

new_df = pd.read_csv("air_quality_data.csv")

column_name = 'NO2'

column_data = new_df[column_name]

z_scores = zscore(column_data)

threshold = 3

outliers = (z_scores > threshold) | (z_scores < -threshold)

# scatter plot

plt.figure(figsize=(10, 6))

plt.scatter(new_df.index, column_data, label='Data')

plt.scatter(new_df.index[outliers], column_data[outliers], color='red', label='Outliers')

plt.xlabel('Index')

plt.ylabel(column_name)

plt.title('Scatter Plot of NO2 with Outliers')

plt.legend()
```

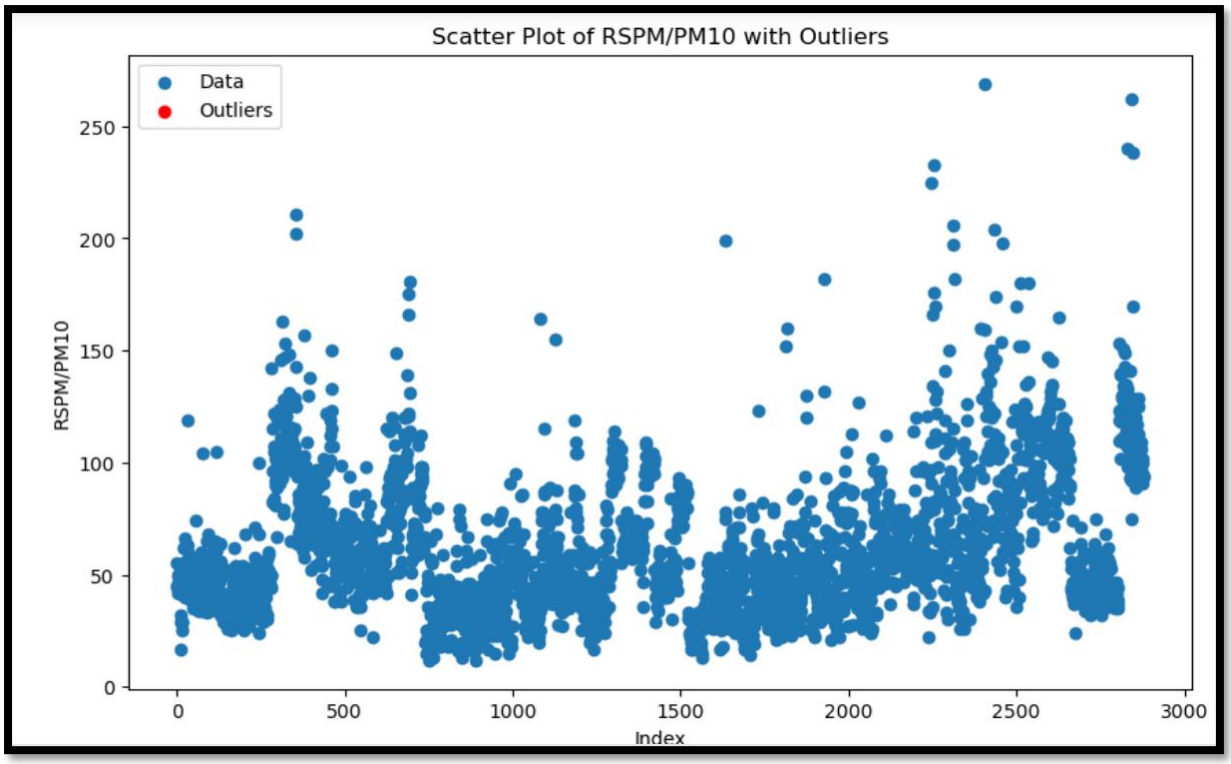
```
In [32]: # Checking of Outliers
         # NO2
         from scipy.stats import zscore
         new_df = pd.read_csv("air_quality_data.csv")
         column_name = 'NO2'
         column_data = new_df[column_name]
         z_scores = zscore(column_data)
         threshold = 3
         outliers = (z_scores > threshold) | (z_scores < -threshold)
         # scatter plot
         plt.figure(figsize=(10, 6))
         plt.scatter(new_df.index, column_data, label='Data')
         plt.scatter(new_df.index[outliers], column_data[outliers], color='red', label='Outliers')
         plt.xlabel('Index')
         plt.ylabel(column_name)
         plt.title('Scatter Plot of NO2 with Outliers')
         plt.legend()
         plt.show()
```



Scatter Plot of RSPM/PM10

```
#RSPM/PM10
# Checking of Outliers
from scipy.stats import zscore
new_df = pd.read_csv("air_quality_data.csv")
column_name = 'RSPM/PM10'
column_data = new_df[column_name]
z_scores = zscore(column_data)
threshold = 3
outliers = (z_scores > threshold) | (z_scores < -threshold)
# scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(new_df.index, column_data, label='Data')
plt.scatter(new_df.index[outliers], column_data[outliers], color='red', label='Outliers')
plt.xlabel('Index')
plt.ylabel(column_name)
plt.title('Scatter Plot of RSPM/PM10 with Outliers')
```

```
In [80]: # RSPM/PM10
         # Checking of Outliers
         from scipy.stats import zscore
         new_df = pd.read_csv("air_quality_data.csv")
         column_name = 'RSPM/PM10'
         column_data = new_df[column_name]
         z_scores = zscore(column_data)
         threshold = 3
         outliers = (z_scores > threshold) | (z_scores < -threshold)
         # scatter plot
         plt.figure(figsize=(10, 6))
         plt.scatter(new_df.index, column_data, label='Data')
         plt.scatter(new_df.index[outliers], column_data[outliers], color='red', label='Outliers')
         plt.xlabel('Index')
         plt.ylabel(column_name)
         plt.title('Scatter Plot of RSPM/PM10 with Outliers')
         plt.legend()
         plt.show()
```



- Dataset imported successfully by importing various python libraries.
- Data describing and defining.
- Removing null values.
- Converting "object" data type to "datetime" datatype
- No Outliers found in data columns of -SO2, NO2, RSPM/PM10
- Hence Dataset is now cleaned and ready for further analysis process.

Importing the initial data:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Updating the initial data:

```
import pandas as pd
new_df = pd.read_csv("air_quality_data.csv")
new_df
```

[118] Python

	Stn Code	Sampling Date	State	City/Town/Village/Area	Location of Monitoring Station	Agency	Type of Location	SO2	NO2	RSPM/PM10	PM 2.5
0	38	01-02-2014	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	11.0	17.0	55.0	NaN
1	38	01-07-2014	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0	17.0	45.0	NaN
2	38	21-01-2014	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	12.0	18.0	50.0	NaN
3	38	23-01-2014	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	15.0	16.0	46.0	NaN
4	38	28-01-2014	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0	14.0	42.0	NaN
...	...	...	...	...	...	...	...	...	...	...	...
2874	773	12-03-2014	Tamil Nadu	Trichy	Central Bus Stand, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	15.0	18.0	102.0	NaN
2875	773	12-10-2014	Tamil Nadu	Trichy	Central Bus Stand, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	12.0	14.0	91.0	NaN
2876	773	17-12-2014	Tamil Nadu	Trichy	Central Bus Stand, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	19.0	22.0	100.0	NaN
2877	773	24-12-2014	Tamil Nadu	Trichy	Central Bus Stand, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	15.0	17.0	95.0	NaN
2878	773	31-12-2014	Tamil Nadu	Trichy	Central Bus Stand, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	14.0	16.0	94.0	NaN

2879 rows × 11 columns

Feature Engineering:

```
Feature Engineering
```

markdown

```
Feature engineering is a subset of data preprocessing in the context of data analysis and machine learning. -Removing unnecessary datas -Data Cleaning -Data Transformation -Data Reduction -Dealing with Imbalanced Data
```

markdown

```
import pandas as pd
df = pd.read_csv("air_quality_data.csv")
df.drop(labels=['Stn Code','Location of Monitoring Station','Agency'], axis = 1, inplace = True)
new_df=df.drop(labels=['PM 2.5'], axis = 1, inplace = True)
df['SO2'].fillna(0, inplace=True)
df['NO2'].fillna(0, inplace=True)
df['RSPM/PM10'].fillna(0, inplace=True)
# Convert the column to integer
df['SO2'] = df['SO2'].astype(int)
df['NO2'] = df['NO2'].astype(int)
df['RSPM/PM10'] =df['RSPM/PM10'].astype(int)
df.to_csv("air_quality_data.csv", index=False)
df.info()
```

[119] Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2879 entries, 0 to 2878
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Sampling Date           2879 non-null   object
 1   State                   2879 non-null   object
 2   City/Town/Village/Area  2879 non-null   object
 3   Type of Location        2879 non-null   object
 4   SO2                     2879 non-null   int32
 5   NO2                     2879 non-null   int32
 6   RSPM/PM10               2879 non-null   int32
dtypes: int32(3), object(4)
memory usage: 123.8+ KB
```

```
df.isnull().sum()
```

[24]

```
Sampling Date             0
State                     0
City/Town/Village/Area    0
Type of Location          0
SO2                       0
NO2                       0
RSPM/PM10                 0
dtype: int64
```

```
df = pd.read_csv("air_quality_data.csv")
df['SO2'].fillna(0, inplace=True)
df['NO2'].fillna(0, inplace=True)
df['RSPM/PM10'].fillna(0, inplace=True)
```

[83]

```
new_df = pd.read_csv("air_quality_data.csv")
column_name = 'SO2'
column_name = 'NO2'
column_name = 'RSPM/PM10'
# Count NaN values
nan_count = df[column_name].isna().sum()
# Get indices of NaN values
nan_indices = df.index[df[column_name].isna()].tolist()

print(f"Number of NaN values in column '{column_name}': {nan_count}")
print(f"Indices of NaN values: {nan_indices}")
```

[84]

```
Number of NaN values in column 'RSPM/PM10': 0
Indices of NaN values: []
```

- Missing values in columns 'SO2', 'NO2', and 'RSPM/PM10' are filled with zeros using the fillna() function. This helps in handling missing data appropriately.

- The columns 'SO2', 'NO2', and 'RSPM/PM10' are converted to integers using the astype() function. This ensures that the data in these columns is treated as integers.

- Finally, df.info() provides summary information about the DataFrame, including data types and non-null counts.

```
[85]  df['SO2'].unique()
```

```
array([11, 13, 12, 15, 14, 10, 16, 19,  9, 20, 17, 18, 25, 21, 23, 26, 24,
       32, 27, 30, 22,  0,  8, 31, 28, 29,  6, 49,  3,  7,  5,  2,  4, 39],
      dtype=int64)
```

```
[28]  df['NO2'].unique()
```

```
array([17, 18, 16, 14, 19, 15, 13, 20, 12, 10, 11, 23, 30, 29, 25, 26, 27,
       34, 35, 32, 22, 24, 21, 28, 31, 33,  0, 38, 41, 47, 36, 42,  9, 44,
        8, 50, 46, 37, 43, 39, 49, 40, 45,  7, 48, 55, 53, 52,  6,  5, 54,
       51, 71, 69], dtype=int64)
```

```
[29]  df['RSPM/PM10'].unique()
```

```
array([ 55,  45,  50,  46,  42,  43,  51,  48,  32,  29,  17,  44,  25,
        41,  54,  62,  66,  40,  56,  49,  63, 119,  61,  52,  53,  57,
        39,  47,  35,  58,  74,  34,  60,  38, 104,  65,  33,  68,  59,
        64, 105,  36,  28,  26,  37,  27,  31,  30,  71,  24, 100, 142,
       115,  83,  96,  82,  84, 122, 107,  92,  90, 102,  81,  89, 120,
        99,  67, 103,  95, 106, 124,  91,  98, 146, 111, 117,  93, 163,
       118,  79,  77, 128, 147, 153, 121, 114, 109, 101, 148, 131, 125,
       108, 116, 110, 129, 211, 202, 143,  76,  94,  69,  86,  72,  75,
        87, 157,  88,  78, 130, 138,  73,  70,  80,  85,  97, 112, 133,
       150, 123,  22, 149, 113, 139, 175, 166, 181,  20,  21,  15,  14,
        16,  12,  13,  19,  18,  23, 164, 155,   0, 199, 152, 160, 182,
       132, 127, 225, 134, 233, 176, 170, 141, 197, 206, 126, 269, 159,
       140, 136, 204, 174, 154, 198, 180, 135, 145, 165, 151, 240, 262,
       238], dtype=int64)
```

Data Visualization:

```
Data Visualization
```

```
df
```

[86]

	Sampling Date	State	City/Town/Village/Area	Type of Location	SO2	NO2	RSPM/PM10
0	01-02-2014	Tamil Nadu	Chennai	Industrial Area	11	17	55
1	01-07-2014	Tamil Nadu	Chennai	Industrial Area	13	17	45
2	21-01-2014	Tamil Nadu	Chennai	Industrial Area	12	18	50
3	23-01-2014	Tamil Nadu	Chennai	Industrial Area	15	16	46
4	28-01-2014	Tamil Nadu	Chennai	Industrial Area	13	14	42
...	...	...	...	...	...	...	...
2874	12-03-2014	Tamil Nadu	Trichy	Residential, Rural and other Areas	15	18	102
2875	12-10-2014	Tamil Nadu	Trichy	Residential, Rural and other Areas	12	14	91
2876	17-12-2014	Tamil Nadu	Trichy	Residential, Rural and other Areas	19	22	100
2877	24-12-2014	Tamil Nadu	Trichy	Residential, Rural and other Areas	15	17	95
2878	31-12-2014	Tamil Nadu	Trichy	Residential, Rural and other Areas	14	16	94

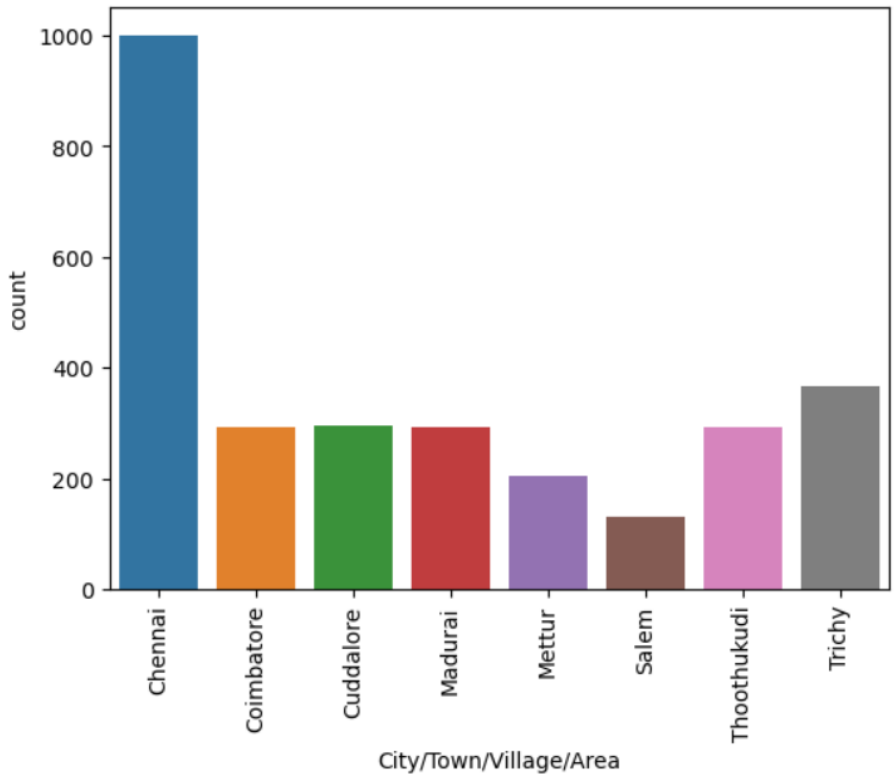
2879 rows × 7 columns

```
print(df["City/Town/Village/Area"])
```

[31]

```
0       Chennai
1       Chennai
2       Chennai
3       Chennai
4       Chennai
         ...
2874     Trichy
2875     Trichy
2876     Trichy
2877     Trichy
2878     Trichy
Name: City/Town/Village/Area, Length: 2879, dtype: object
```

```
datacount =sns.countplot(x ="City/Town/Village/Area",data = df);
datacount.set_xticklabels(datacount.get_xticklabels(), rotation=90);
```



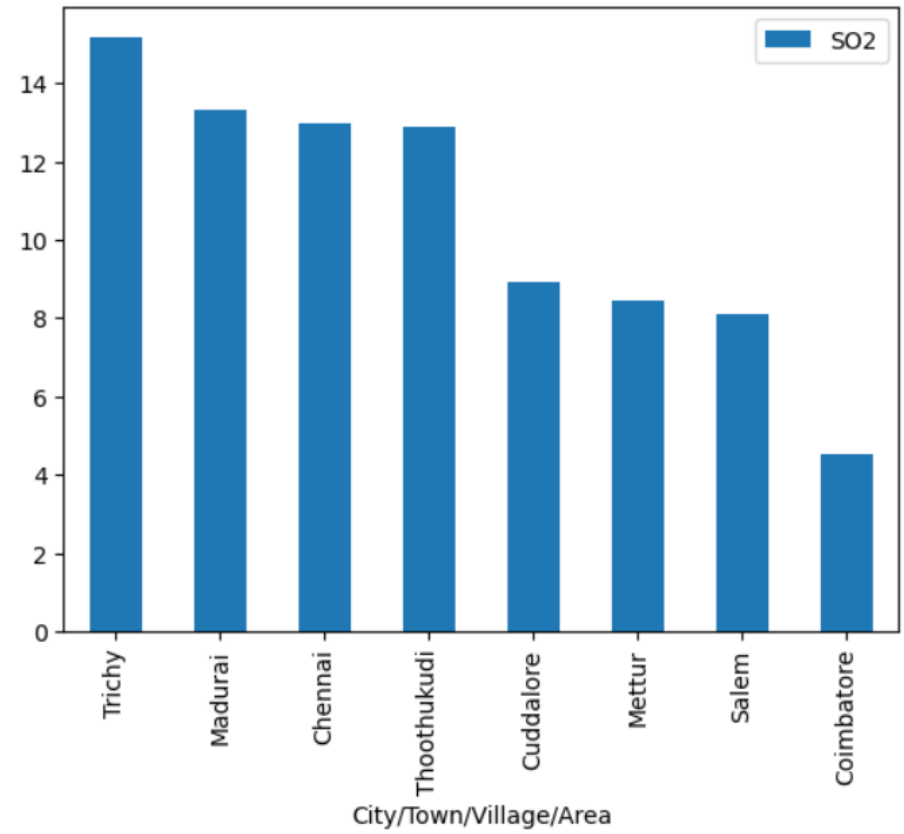

```
loc = pd.pivot_table(df, values=['SO2','NO2','RSPM/PM10'],index='City/Town/Village/Area')
# Aggfunc: default-np.mean()
loc
```

City/Town/Village/Area	NO2	RSPM/PM10	SO2
Chennai	21.978000	58.998000	12.975000
Coimbatore	25.238908	48.713311	4.525597
Cuddalore	19.577703	61.881757	8.905405
Madurai	25.768707	45.724490	13.319728
Mettur	23.185366	52.721951	8.429268
Salem	28.664122	62.954198	8.114504
Thoothukudi	18.385666	83.174061	12.901024
Trichy	18.542234	85.054496	15.168937

```
maxso2 = loc.sort_values(by='SO2',ascending=False)
maxso2.loc[:,['SO2']].head(10).plot(kind='bar')
# Based on average values
```

[88]

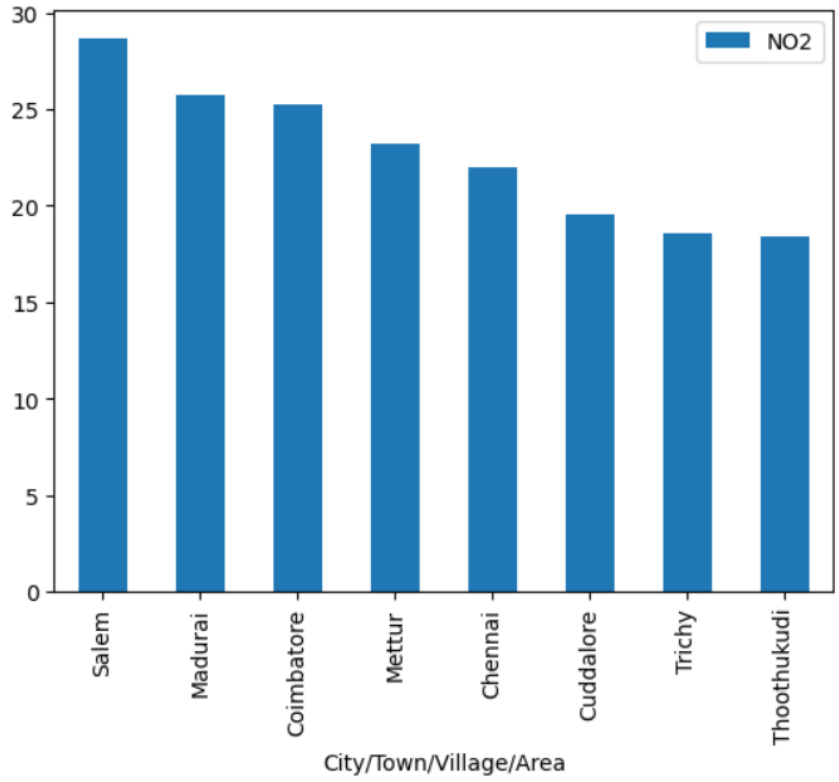
```
<Axes: xlabel='City/Town/Village/Area'>
```



- The resulting bar chart represents the ten locations with the highest recorded levels of SO2.
- This visualization allows for a quick comparison of SO2 levels among these locations.

```
maxno2 = loc.sort_values(by='NO2',ascending=False)
maxno2.loc[:,['NO2']].head(10).plot(kind='bar')
```

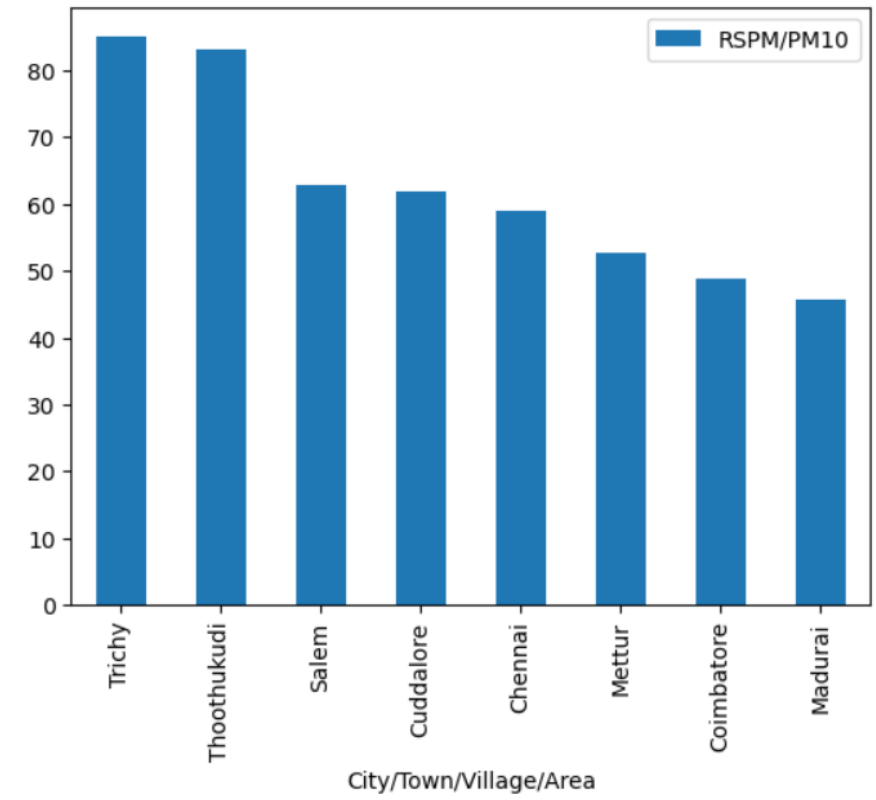
```
<Axes: xlabel='City/Town/Village/Area'>
```



- The resulting bar chart represents the ten locations with the highest recorded levels of NO2.
- This visualization allows for a quick comparison of NO2 levels among these locations.

```
maxrspm_pm10 = loc.sort_values(by='RSPM/PM10',ascending=False)
maxrspm_pm10.loc[:,['RSPM/PM10']].head(10).plot(kind='bar')
```

```
<Axes: xlabel='City/Town/Village/Area'>
```



- The resulting bar chart represents the ten locations with the highest recorded levels of RSPM/PM10.
- This visualization allows for a quick comparison of RSPM/PM10 levels among these locations.

Calculating AQI-Air Quality Index:

```
# Sulfur Dioxide (SO2):
# A pungent gas released by volcanic eruptions and industrial processes.
def calculate_si(SO2):
    si=0
    if (SO2<=40):
     si= "s1"
    if (SO2>40 and SO2<=80):
     si= "s2"
    if (SO2>80 and SO2<=380):
     si= "s3"
    if (SO2>380 and SO2<=800):
     si= "s4"
    if (SO2>800 and SO2<=1600):
     si= "s5"
    if (SO2>1600):
     si= "s6"
    return si
df['si']=df['SO2'].apply(calculate_si)
ds= df[['SO2','si']]
ds.tail()
```

[120]

	SO2	si
2874	15	s1
2875	12	s1
2876	19	s1
2877	15	s1
2878	14	s1

- The code defines a function calculate_si(SO2) that takes the concentration of Sulfur Dioxide (SO2) as input and returns the corresponding Sulfur Dioxide Index (si) based on predefined concentration ranges.

- The si values range from 's1' to 's6', representing different levels of SO2 concentration.
  For example, 's1' corresponds to low SO2 concentration, while 's6' indicates very high SO2 concentration.

```
# Nitrogen Dioxide (NO2): A reddish-brown gas that is a byproduct of burning fossil fuels.
def calculate_ni(NO2):
    ni=0
    if (NO2<=40):
     ni= "n1"
    if (NO2>40 and NO2<=80):
     ni= "n2"
    if (NO2>80 and NO2<=180):
     ni= "n3"
    if (NO2>180 and NO2<=280):
     ni= "n4"
    if (NO2>280 and NO2<=400):
     ni= "n5"
    if (NO2>400):
     ni= "n6"
    return ni
df['ni']=df['NO2'].apply(calculate_ni)
dn= df[['NO2','ni']]
dn.tail()
```

[121]

	NO2	ni
2874	18	n1
2875	14	n1
2876	22	n1
2877	17	n1
2878	16	n1

- The code defines a function calculate_ni(NO2) that takes the concentration of Nitrogen Dioxide (NO2) as input and returns the corresponding Nitrogen Dioxide Index (ni) based on predefined concentration ranges.

- The si values range from 'n1' to 'n6', representing different levels of NO2 concentration.
  For example, 'n1' corresponds to low NO2 concentration, while 'n6' indicates very high NO2 concentration.

```
# RSPM (Respirable Suspended Particulate Matter)
# PM10 (Particulate Matter with a diameter of 10 micrometers or less)
def calculate_spi(rspm_pm10):
    spi=0
    if (rspm_pm10<=40):
      spi= "sp1"
    if (rspm_pm10>40 and rspm_pm10<=80):
      spi= "sp2"
    if (rspm_pm10>80 and rspm_pm10<=180):
      spi= "sp3"
    if (rspm_pm10>180 and rspm_pm10<=280):
      spi= "sp4"
    if (rspm_pm10>280 and rspm_pm10<=400):
      spi= "sp5"
    if (rspm_pm10>400):
      spi= "sp6"
    return  spi
df['spi']=df['RSPM/PM10'].apply(calculate_spi)
dsp= df[['RSPM/PM10','spi']]
dsp.tail()
```

[122]

	RSPM/PM10	spi
2874	102	sp3
2875	91	sp3
2876	100	sp3
2877	95	sp3
2878	94	sp3

- The spi values range from 'sp1' to 'sp6', representing different levels of RSPM/PM10 concentration.
- For example, 'sp1' corresponds to low RSPM/PM10 concentration, while 'sp6' indicates very high RSPM/PM10 concentration.
-
- This code is used to categorize RSPM/PM10 concentrations into different levels represented by the spi values. The resulting DataFrame dsp provides a snapshot of RSPM/PM10 concentrations and their respective spi classifications.

```
df.head()
```

[35]

	Sampling Date	State	City/Town/Village/Area	Type of Location	SO2	NO2	RSPM/PM10	si	ni	spi
0	01-02-2014	Tamil Nadu	Chennai	Industrial Area	11	17	55	s1	n1	sp2
1	01-07-2014	Tamil Nadu	Chennai	Industrial Area	13	17	45	s1	n1	sp2
2	21-01-2014	Tamil Nadu	Chennai	Industrial Area	12	18	50	s1	n1	sp2
3	23-01-2014	Tamil Nadu	Chennai	Industrial Area	15	16	46	s1	n1	sp2
4	28-01-2014	Tamil Nadu	Chennai	Industrial Area	13	14	42	s1	n1	sp2

```
# AQI
def calculate_aqi(si,ni,spi):
    aqi=0
    if(si>ni and si>spi):
     aqi=si
    if (spi>ni and spi>si):
     aqi=spi
    if(ni>si and ni>spi):
     aqi= ni
    return aqi
df['AQI']=df.apply(lambda x:calculate_aqi(x['SO2'],x['NO2'],x['RSPM/PM10']),axis=1)
```

5]

```
df.head()
```

4]

	Sampling Date	State	City/Town/Village/Area	Type of Location	SO2	NO2	RSPM/PM10	si	ni	spi	AQI
0	01-02-2014	Tamil Nadu	Chennai	Industrial Area	11	17	55	s1	n1	sp2	55
1	01-07-2014	Tamil Nadu	Chennai	Industrial Area	13	17	45	s1	n1	sp2	45
2	21-01-2014	Tamil Nadu	Chennai	Industrial Area	12	18	50	s1	n1	sp2	50
3	23-01-2014	Tamil Nadu	Chennai	Industrial Area	15	16	46	s1	n1	sp2	46
4	28-01-2014	Tamil Nadu	Chennai	Industrial Area	13	14	42	s1	n1	sp2	42

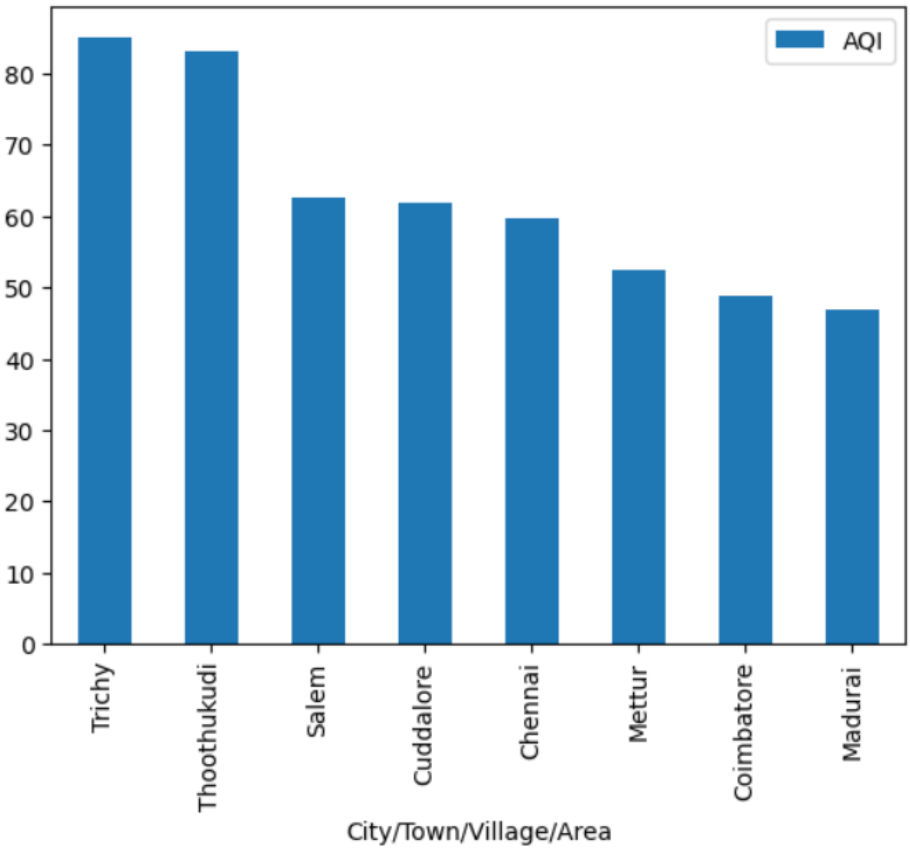
```
aq_wise = pd.pivot_table(df, values=['AQI'],index='City/Town/Village/Area')
aq_wise
```

[125]

City/Town/Village/Area	AQI
Chennai	59.691000
Coimbatore	48.914676
Cuddalore	61.885135
Madurai	46.945578
Mettur	52.541463
Salem	62.541985
Thoothukudi	83.252560
Trichy	85.054496

```
[126]
maxaqi = aq_wise.sort_values(by='AQI',ascending=False)
maxaqi.loc[:,['AQI']].head(37).plot(kind='bar')
```

```
<Axes: xlabel='City/Town/Village/Area'>
```



```
[127]
date_wise = pd.pivot_table(df, values=['AQI'],index='Sampling Date')
date_wise
```

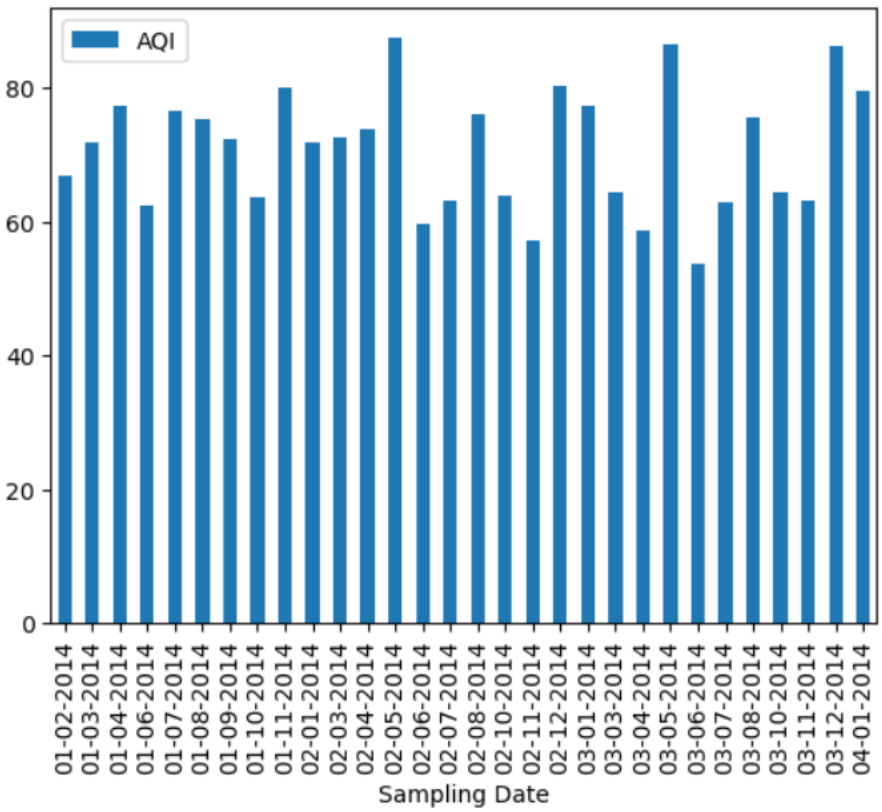
Sampling Date	AQI
01-02-2014	66.818182
01-03-2014	71.769231
01-04-2014	77.250000
01-06-2014	62.454545
01-07-2014	76.461538
...	...
31-03-2014	62.000000
31-05-2014	50.000000
31-07-2014	58.076923
31-10-2014	59.700000
31-12-2014	57.300000

302 rows × 1 columns

- The resulting date_wise pivot table provides a view of the Air Quality Index (AQI) data sorted by each unique sampling date.
- This allows for a quick summary and analysis of AQI trends over time.

```
[128]  date_wise.loc[:,['AQI']].head(30).plot(kind='bar')
```

```
<Axes: xlabel='Sampling Date'>
```



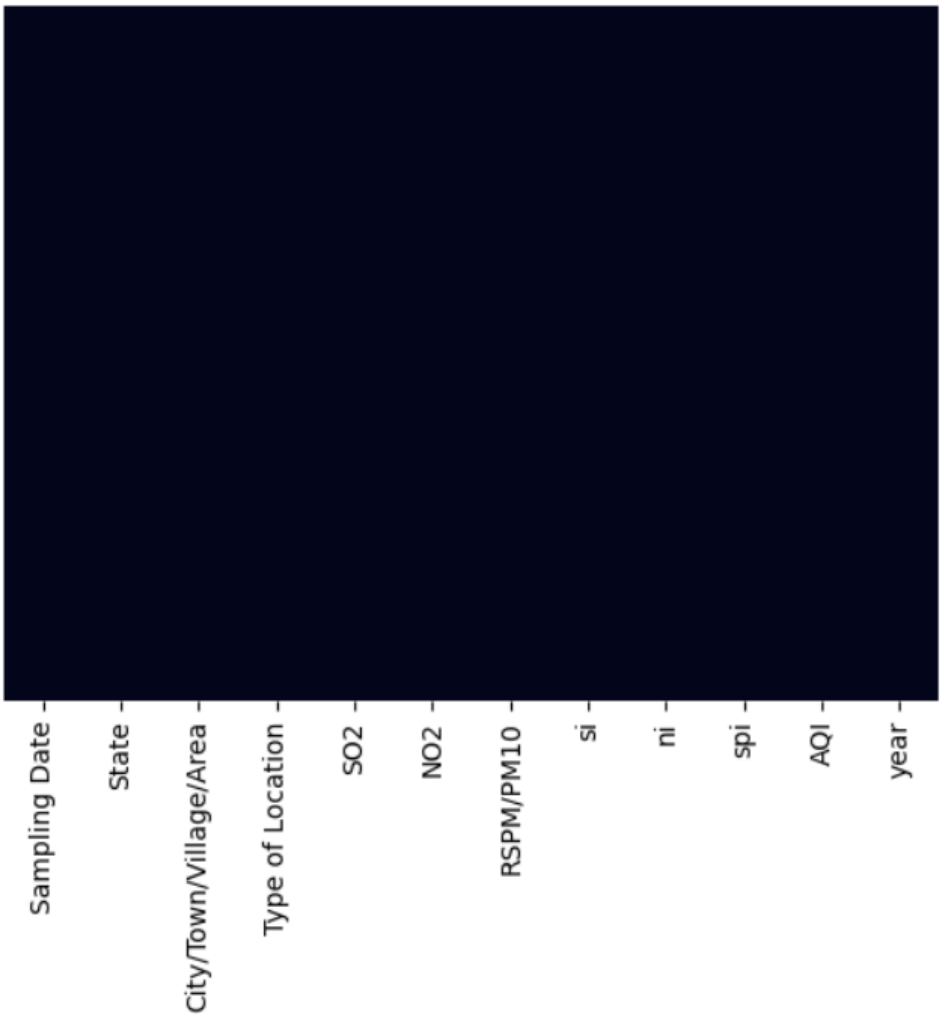
- This visualization provides a snapshot of AQI trends over the initial 30 sampling dates. It allows for a quick assessment of air quality variations during this period.

Training Data:

```
dum1 = pd.get_dummies(df['Type of Location'])
dum2 = pd.get_dummies(df['City/Town/Village/Area'])
df['year'] = df['Sampling Date']
```

```
[132]  sns.heatmap(df.isna(), yticklabels=False, cbar=False)
```

```
<Axes: >
```



```
df = pd.concat([df, dum1, dum2], axis = 1)
df.head()
```

[98] Python

	Sampling Date	State	City/Town/Village/Area	Type of Location	SO2	NO2	RSPM/PM10	si	ni	spi	...	Industrial Area	Residential, Rural and other Areas	Chennai	Coimbatore	Cuddalore	Madurai	Mettur	Salem	Thoothukudi
0	01-02-2014	Tamil Nadu	Chennai	Industrial Area	11	17	55	s1	n1	sp2	...	True	False	True	False	False	False	False	False	False
1	01-07-2014	Tamil Nadu	Chennai	Industrial Area	13	17	45	s1	n1	sp2	...	True	False	True	False	False	False	False	False	False
2	21-01-2014	Tamil Nadu	Chennai	Industrial Area	12	18	50	s1	n1	sp2	...	True	False	True	False	False	False	False	False	False
3	23-01-2014	Tamil Nadu	Chennai	Industrial Area	15	16	46	s1	n1	sp2	...	True	False	True	False	False	False	False	False	False
4	28-01-2014	Tamil Nadu	Chennai	Industrial Area	13	14	42	s1	n1	sp2	...	True	False	True	False	False	False	False	False	False

5 rows × 22 columns

```
df.info()
```

[130]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2879 entries, 0 to 2878
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Sampling Date          2879 non-null   object
 1   State                  2879 non-null   object
 2   City/Town/Village/Area 2879 non-null   object
 3   Type of Location       2879 non-null   object
 4   SO2                    2879 non-null   int32
 5   NO2                    2879 non-null   int32
 6   RSPM/PM10              2879 non-null   int32
 7   si                     2879 non-null   object
 8   ni                     2879 non-null   object
 9   spi                    2879 non-null   object
 10  AQI                    2879 non-null   int64
dtypes: int32(3), int64(1), object(7)
memory usage: 213.8+ KB
```

```
df.corr()
```

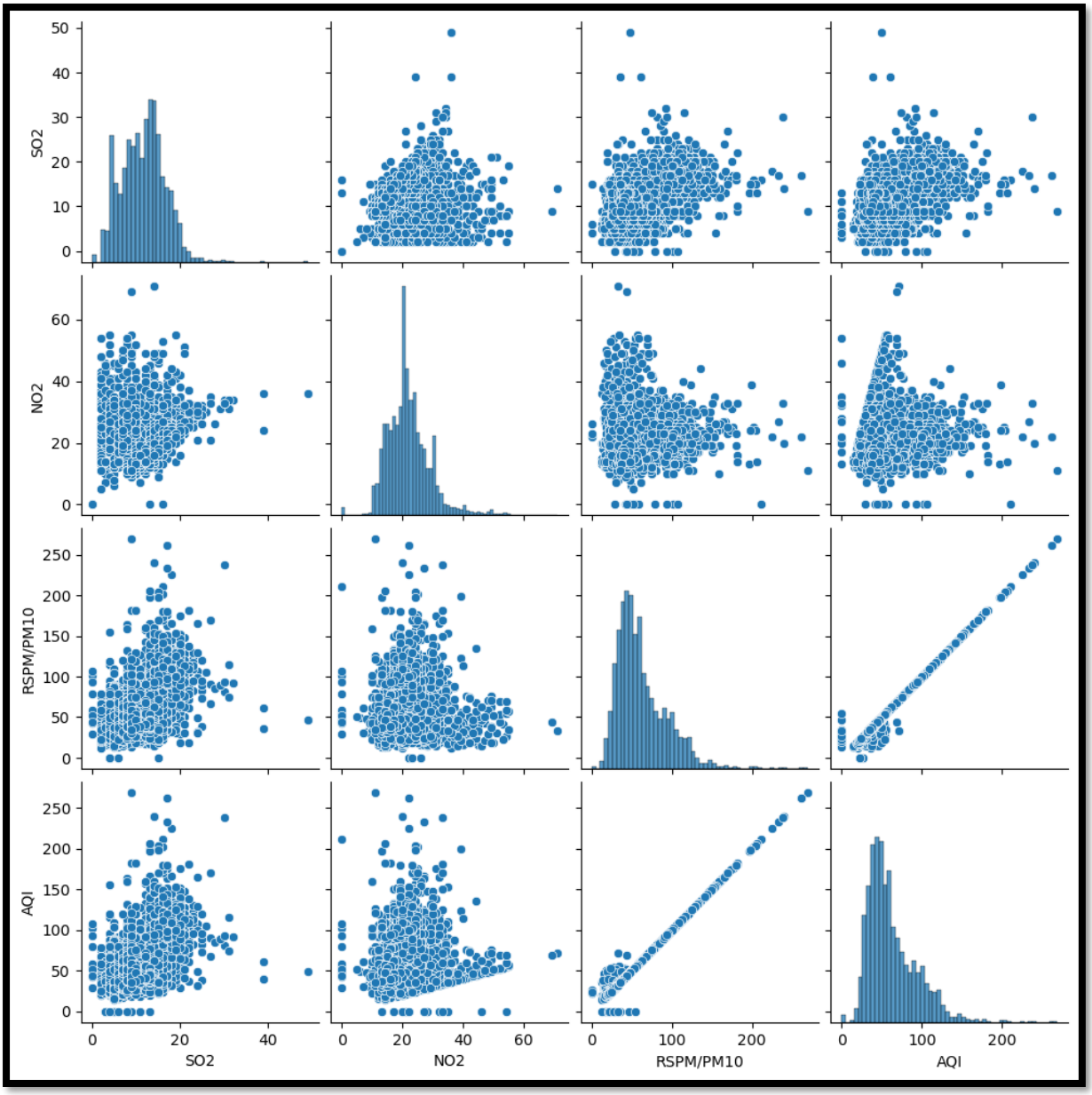
[164]

	SO2	NO2	RSPM/PM10	AQI
SO2	1.000000	0.100779	0.440141	0.441655
NO2	0.100779	1.000000	0.060369	0.093855
RSPM/PM10	0.440141	0.060369	1.000000	0.993675
AQI	0.441655	0.093855	0.993675	1.000000

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

```
sns.pairplot(df)
```

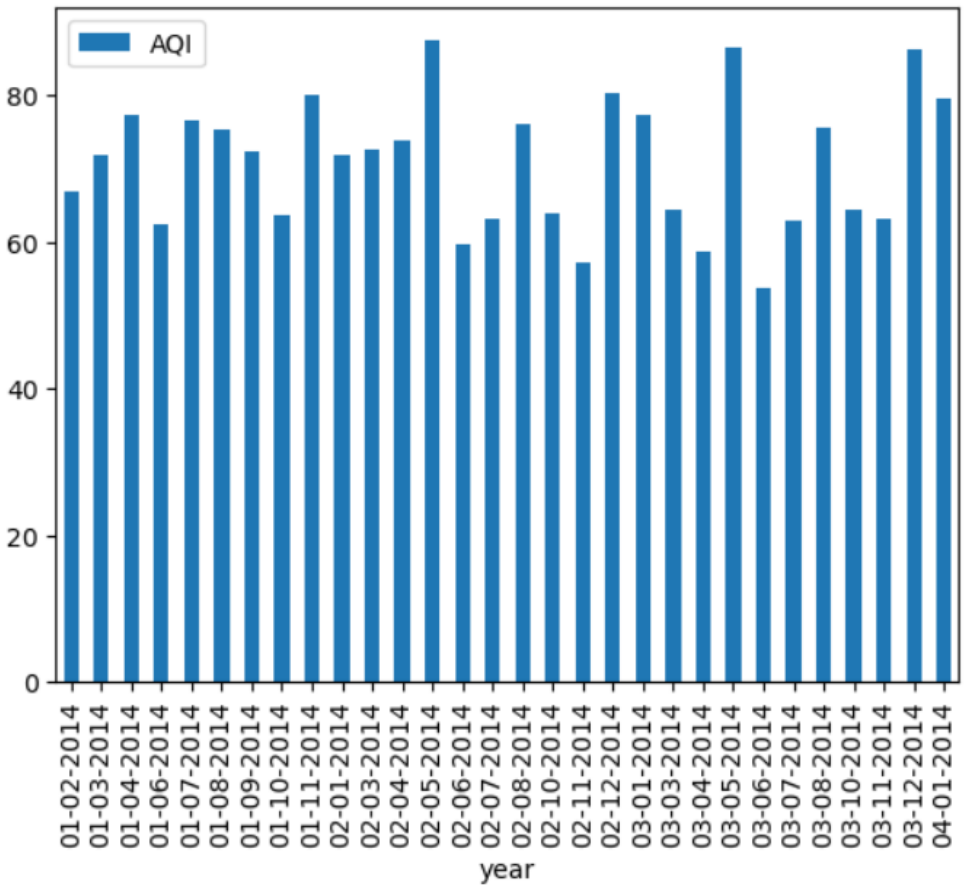
```
c:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)

<seaborn.axisgrid.PairGrid at 0x181191f7d10>
```



```
yr_wise = pd.pivot_table(df, values=['AQI'],index='year')
yr_wise.loc[:,['AQI']].head(30).plot(kind='bar')
```

[33]

```
<Axes: xlabel='year'>
```




```
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.30,random_state=25)
```

Training Data:

Model fittings

Simple Linear Regression

```
[161]
from sklearn.linear_model import LinearRegression
```

```
[167]
from sklearn.model_selection import train_test_split
X=df[['SO2','NO2','RSPM/PM10']]
y=df[['AQI']]
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.30,random_state=25)
```

```
[173]
from sklearn.linear_model import LinearRegression
lin_mod = LinearRegression()
lin_mod.fit(x_train, y_train)
lin_mod.score(x_train, y_train )
```

```
0.9887589817873087
```

Conclusion:

- The high R^2 value of approximately 0.9888 indicates that the Linear Regression model explains about 98.88% of the variance in the target variable based on the features in the training data.

- This suggests that the model is performing very well on the training data and is able to capture the relationships between the features and the target variable effectively.

- Overall, the code demonstrates the successful training of a Linear Regression model, and the high R^2 score indicates its strong performance on the training data.