TASK – 2 (INTERMEDIATE)

DELHI AIR QUALITY INDEX (AQI) ANALYSIS

DOCUMENTATION

CONTENTS

ABSTRACT:

The air quality index (AQI) serves as a crucial metric for evaluating the quality of air in urban environments, impacting public health, environmental sustainability, and policy-making decisions. In megacities like Delhi, where air pollution levels often surpass safe thresholds, understanding and analyzing the AQI becomes imperative. This project endeavors to provide insights into the air quality scenario in Delhi through comprehensive analysis and visualization of AQI data using Python.

Utilizing Python's powerful data analysis libraries such as Pandas, Matplotlib, and Seaborn, along with data extraction techniques, this project aims to explore the temporal and spatial trends of AQI in Delhi. By collecting and processing historical AQI data from diverse sources including governmental agencies and independent monitors, we construct a robust dataset for analysis.

The project encompasses various stages, starting from data collection and preprocessing to exploratory data analysis (EDA) and visualization. Through EDA, we delve into factors influencing AQI fluctuations, such as meteorological conditions, vehicular emissions, industrial activities, and seasonal variations. Visual representations including time series plots, heatmaps, and geographical visualizations elucidate the patterns and correlations within the data.

Furthermore, this project extends its analysis to assess the effectiveness of regulatory measures and interventions on air quality improvement. By comparing AQI trends before and after policy implementations, we aim to evaluate their impact and efficacy.

The outcomes of this project are intended to serve as a valuable resource for policymakers, environmentalists, researchers, and the general public. By fostering a deeper understanding of Delhi's air quality dynamics and the factors driving pollution levels, this project strives to facilitate informed decision-making towards mitigating the adverse effects of air pollution and fostering a healthier environment for all residents.

Introduction:

This project aims to analyze air quality data from a specific region to identify trends, seasonal patterns, correlations between pollutants, and the potential health impacts of air quality levels. By leveraging various data analysis techniques and visualizations, we seek to provide comprehensive insights into the air quality of the region.

Data Source:

The data used in this project is stored in a CSV file named delhiaqi.csv,downloaded from Kaggle. The dataset contains air quality readings for various pollutants along with their respective timestamps.

Objectives :

- Calculate exceedances of air quality standards.
- Plot daily average exceedances of pollutants.
- Identify trends, seasonal patterns, and correlations between pollutants.
- Assess the potential health impacts of air quality levels.
- Develop predictive models for future air quality forecasting.

Data Preprocessing:

- Import necessary libraries
- Load the dataset using Pandas.
- Use head() to list the top 5 rows of the data set.

[4]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
data = pd.read_csv(r"C:\Users\santh\Downloads\delhiaqi.csv")
```

[5]:
```
print(data.head())
```

```
                 date       co     no    no2    o3    so2   pm2_5    pm10  \
0 2023-01-01 00:00:00  1655.58   1.66  39.41  5.90  17.88  169.29  194.64
1 2023-01-01 01:00:00  1869.20   6.82  42.16  1.99  22.17  182.84  211.08
2 2023-01-01 02:00:00  2510.07  27.72  43.87  0.02  30.04  220.25  260.68
3 2023-01-01 03:00:00  3150.94  55.43  44.55  0.85  35.76  252.90  304.12
4 2023-01-01 04:00:00  3471.37  68.84  45.24  5.45  39.10  266.36  322.80

     nh3
0   5.83
1   7.66
2  11.40
3  13.55
4  14.19
```

Rename the columns for better readability and handle missing values by performing the following steps:

[70]:
```
# Renaming the column pm2_5 to pm2.5(particle matter) for better readability
data.rename(columns={'pm2_5': 'pm2.5'}, inplace=True)
```

[18]:
```
missing_values = data.isnull().sum()
print("Missing values:\n", missing_values)

# Fill or drop missing values as needed (example: forward fill method)
data.ffill(inplace=True)
```

```
Missing values:
 date       0
co          0
no          0
no2         0
o3          0
so2         0
pm2.5       0
pm10        0
nh3         0
day         0
hour        0
weekday     0
dtype: int64
```

[19]:
```
# Convert 'date' column to datetime
data['date'] = pd.to_datetime(data['date'])

# Extract day, hour, weekday from the date
data['day'] = data['date'].dt.day
data['hour'] = data['date'].dt.hour
data['weekday'] = data['date'].dt.weekday
```

Get to know the Descriptive Statistics using describe():

[71]:
```
# Descriptive statistics
desc_stats = data[data.columns[:-3]].describe()
print("Descriptive Statistics:\n", desc_stats)
```

```
Descriptive Statistics:
                      date           co          no         no2          o3  \
count                  561    561.000000  561.000000  561.000000  561.000000
mean   2023-01-12 16:00:00   3814.942210   51.181979   75.292496   30.141943
min    2023-01-01 00:00:00    654.220000    0.000000   13.370000    0.000000
25%    2023-01-06 20:00:00   1708.980000    3.380000   44.550000    0.070000
50%    2023-01-12 16:00:00   2590.180000   13.300000   63.750000   11.800000
75%    2023-01-18 12:00:00   4432.680000   59.010000   97.330000   47.210000
max    2023-01-24 08:00:00  16876.220000  425.580000  263.210000  164.510000
std                    NaN   3227.744681   83.904476   42.473791   39.979405

              so2        pm2.5         pm10         nh3         day
count  561.000000   561.000000   561.000000  561.000000  561.000000
mean    64.655936   358.256364   420.988414   26.425062   12.192513
min      5.250000    60.100000    69.080000    0.630000    1.000000
25%     28.130000   204.450000   240.900000    8.230000    6.000000
50%     47.210000   301.170000   340.900000   14.820000   12.000000
75%     77.250000   416.650000   482.570000   26.350000   18.000000
max    511.170000  1310.200000  1499.270000  267.510000   24.000000
std     61.073080   227.359117   271.287026   36.563094    6.756374
```

Columns:

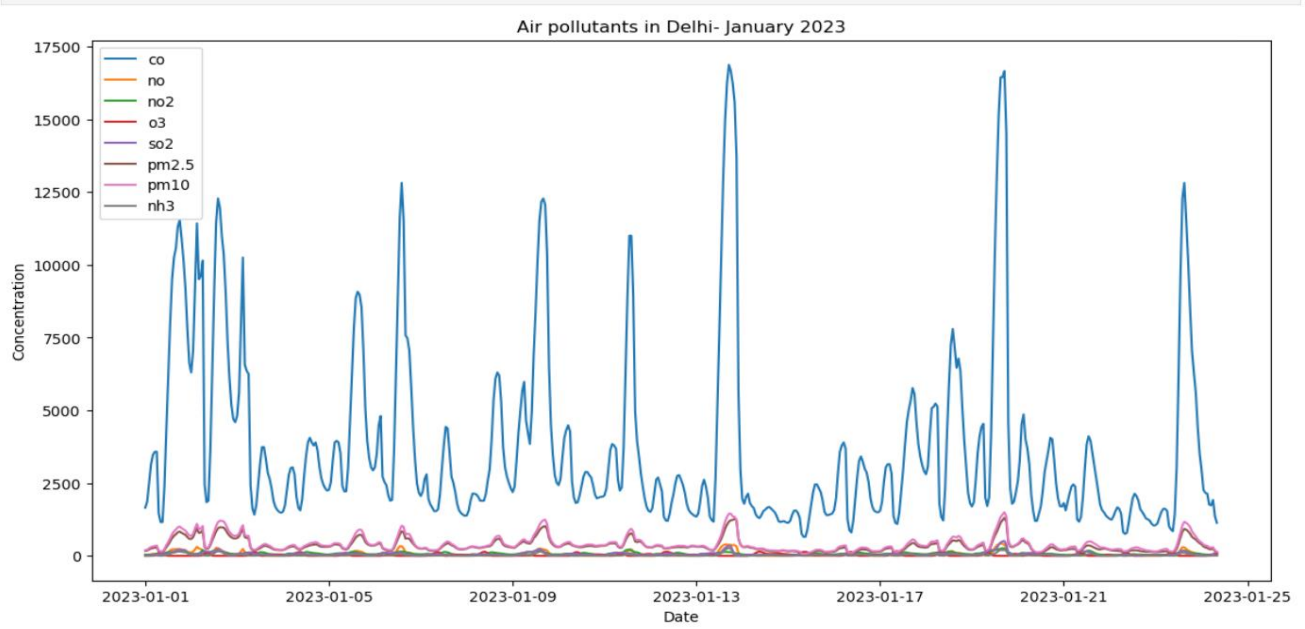
- date: Timestamp of the measurement
- co: Carbon monoxide (ppb)
- no: Nitric oxide (ppb)
- no2: Nitrogen dioxide (ppb)
- o3: Ozone (ppb)
- so2: Sulfur dioxide (ppb)
- pm2.5: Particulate matter <2.5 micrometers (µg/m³)
- pm10: Particulate matter <10 micrometers (µg/m³)
- nh3: Ammonia (ppb)

2.Plotting time series for each pollutant:

[23]:
```
# Plot time series for each pollutant
plt.figure(figsize=(14, 7))
for column in  ['co', 'no', 'no2', 'o3', 'so2', 'pm2.5', 'pm10', 'nh3']:
    plt.plot(data.date, data[column], label=column)

plt.xlabel('Date')
plt.ylabel('Concentration')
plt.title('Air pollutants in Delhi- January 2023')
plt.legend()
plt.show()
```

The resulting plot visualizes the daily concentrations of various air pollutants in Delhi throughout January 2023, allowing for an assessment of temporal trends and identification of days with high pollution levels

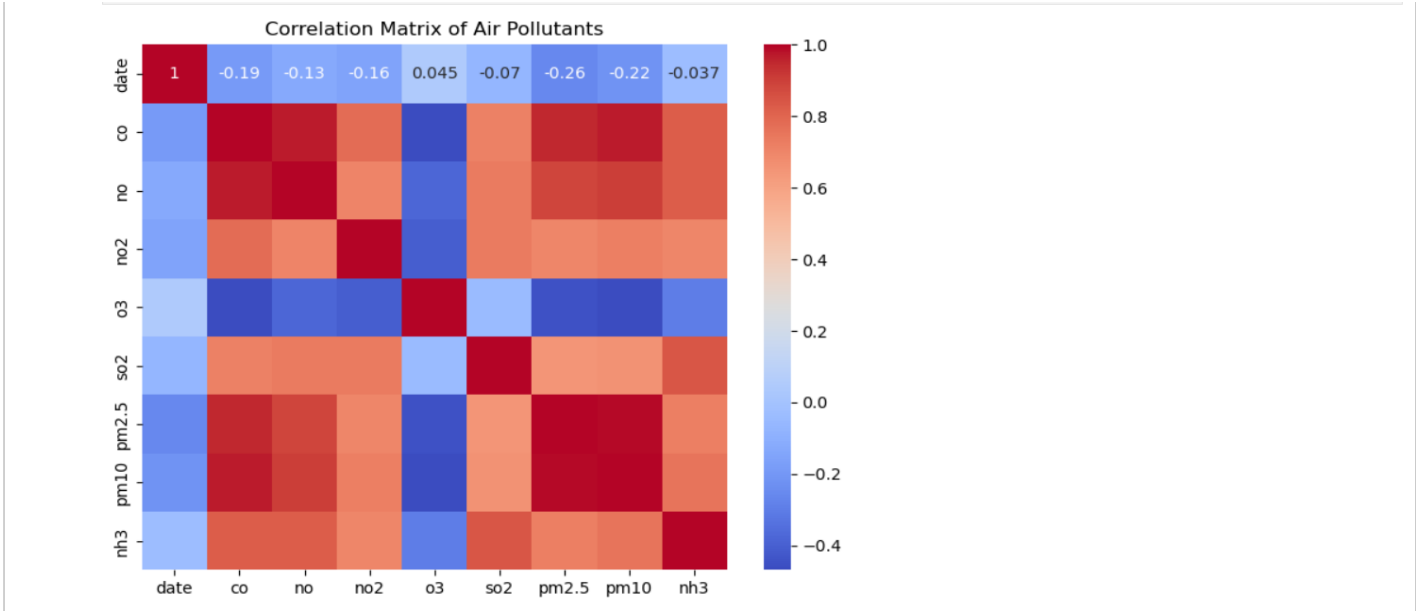


3.Correlation matrix between pollutants:

- Heatmap of Pollutant Correlations
- Visualize the correlation between different pollutants using a heatmap:

[27]:
```
# Correlation matrix
correlation_matrix = data[data.columns[:-3]].corr()

# Heatmap of correlation matrix
plt.figure(figsize=(10,8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix of Air Pollutants')
plt.show()
```



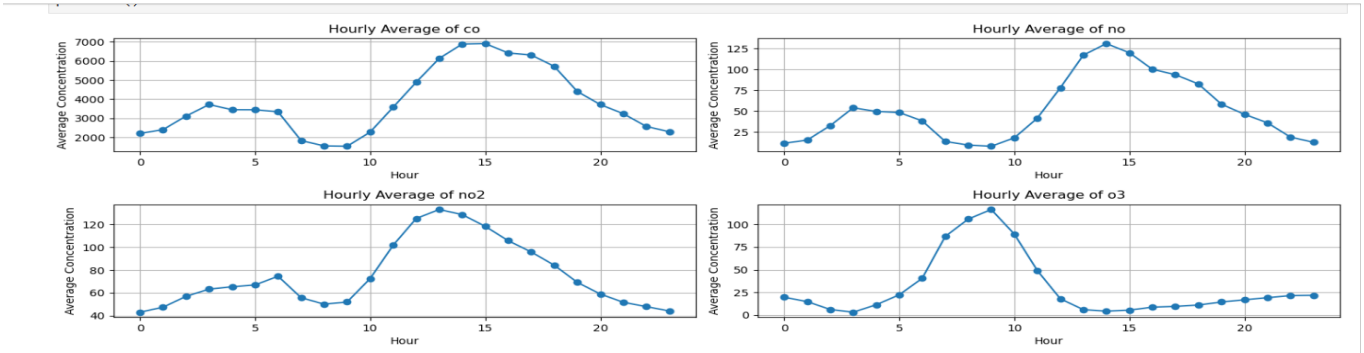
4.Plotting Average Hourly concentration for each pollutants:

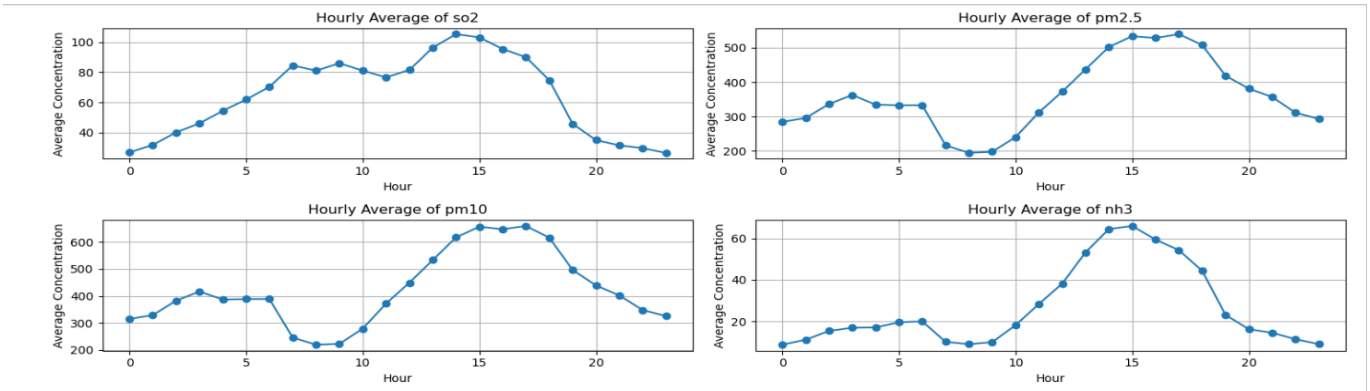
- Plots the concentration of each pollutant over time.
- Adds labels, title, legend, and grid to the plot.
- Uses tight_layout() to ensure proper spacing in the plot**.**

[26]:
```
# Compute hourly averages
hourly_avg = data.groupby('hour').mean()

# Plot hourly averages for each pollutant
plt.figure(figsize=(15, 10))
for i, col in enumerate(data.columns[1:-3], 1):  # Skipping the 'date' and extracted columns, and 'day', 'hour', 'weekday'
    plt.subplot(4, 2, i)
    plt.plot(hourly_avg.index, hourly_avg[col], marker='o')
    plt.title(f'Hourly Average of {col}')
    plt.xlabel('Hour')
    plt.ylabel('Average Concentration')
    plt.grid(True)

plt.tight_layout()
plt.show()
```





The resulting plots visualize the hourly average concentrations of various air pollutants in Delhi throughout January 2023. The analysis reveals distinct patterns for different pollutants:

- CO, NO, NO2, PM2.5, PM10, and NH3: These pollutants exhibit a similar pattern where concentrations rise in the morning, peak around 2-3 PM, and then decrease significantly by 10 PM. This pattern suggests a strong influence of traffic and industrial activities, which are higher during the day and reduce in the evening.

- O3 and SO2: These pollutants display unique patterns:
  O3: Shows a different trend, likely influenced by photochemical reactions. Ozone levels typically rise during the daytime due to sunlight-driven chemical reactions involving precursor pollutants.

- SO2: Exhibits a distinct pattern that may be influenced by specific sources like industrial emissions or power plants, which do not follow the same daily activity cycle as traffic-related pollutants.

5.Weekday vs weekend analysis:

Analysing the mean concentration of pollutants between weekdays and weekends

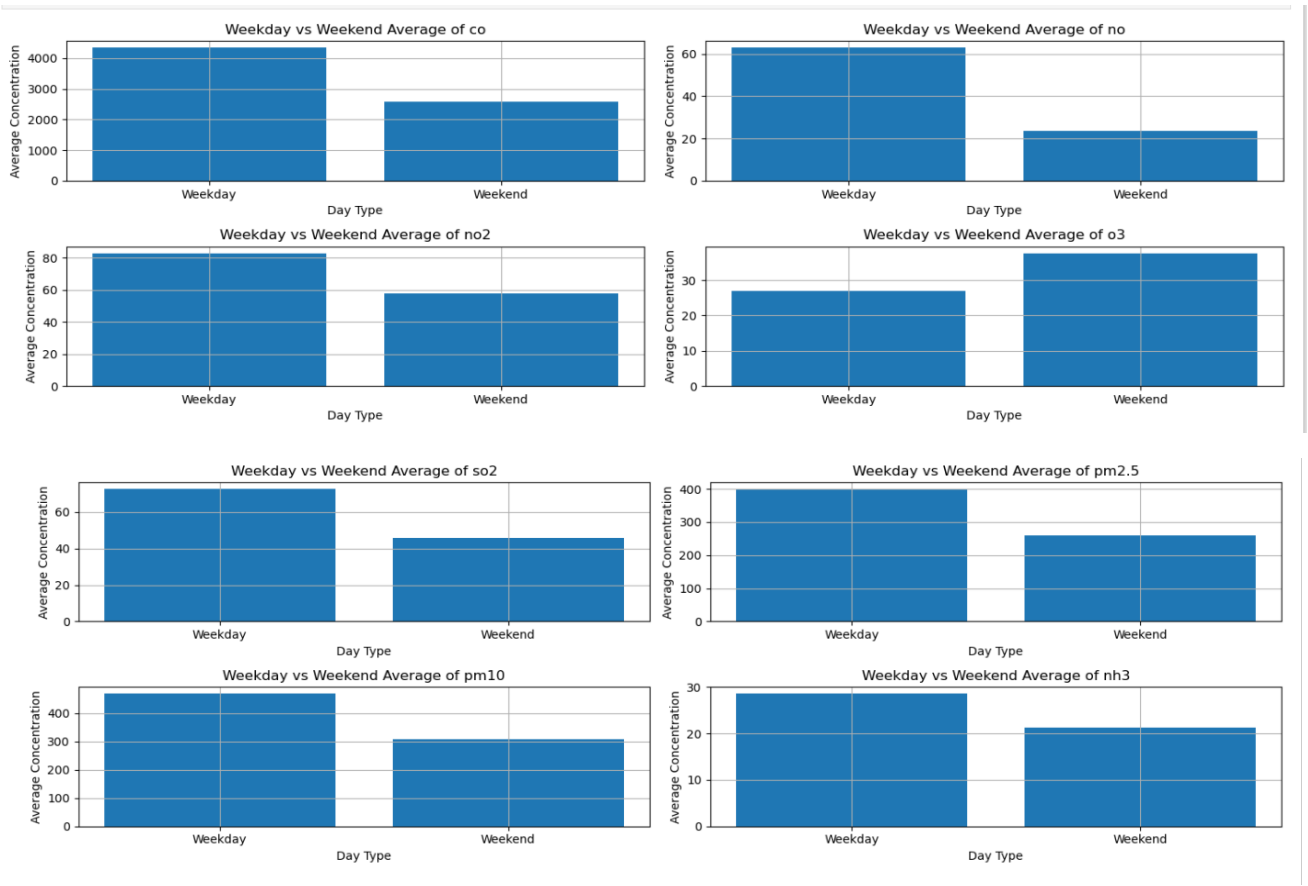
[34]:
```
# Create a 'weekend' column (1 if weekend, 0 if weekday)
data['weekend'] = data['weekday'].apply(lambda x: 1 if x >= 5 else 0)

# Compute averages for weekdays and weekends
weekend_avg = data.groupby('weekend').mean()

# Plot comparison
plt.figure(figsize=(15, 10))
for i, col in enumerate(data.columns[1:-4], 1):  # Skipping the 'date' and extracted columns, and 'day', 'hour', 'weekday', 'weekend'
    plt.subplot(4, 2, i)
    plt.bar(['Weekday', 'Weekend'], weekend_avg[col])
    plt.title(f'Weekday vs Weekend Average of {col}')
    plt.xlabel('Day Type')
    plt.ylabel('Average Concentration')
    plt.grid(True)

plt.tight_layout()
plt.show()
```

The results of the interpretation are:



From the above observation that pollutant concentrations are higher on weekdays compared to weekends implies several things:

1.Increased Human Activity:

- Traffic Emissions: Weekdays typically see more vehicle traffic due to commuting to work, school, and other activities. This results in higher emissions of pollutants such as CO, NO, and NO_2 from vehicle exhausts.
- Industrial Activity: Many industries operate primarily on weekdays, leading to higher emissions of various pollutants.

2.Reduced Activity on Weekends:

- Less Traffic: There is generally less commuting and industrial activity on weekends, leading to lower emissions.
- Different Patterns: People's activities differ on weekends, often involving fewer long commutes and reduced industrial operation.

6.Defining AQI Standards:

To align the analysis with Delhi's air quality standards, we need to use the specific AQI breakpoints set by the Central Pollution Control Board (CPCB) of India or other relevant authorities. Here are the CPCB AQI breakpoints for key pollutants:

CPCB AQI Breakpoints (Delhi, India)

Pollutant	Good (0-50)	Satisfactory (51-100)	Moderate (101-200)	Poor (201-300)	Very Poor (301-400)	Severe (401-500)
PM2.5	0-30	31-60	61-90	91-120	121-250	251-500
PM10	0-50	51-100	101-250	251-350	351-430	431-600
NO2	0-40	41-80	81-180	181-280	281-400	401-500
SO2	0-40	41-80	81-380	381-800	801-1600	1601-2000
CO	0-1.0 mg/m³	1.1-2.0 mg/m³	2.1-10 mg/m³	10-17 mg/m³	17-34 mg/m³	34+ mg/m³
O3	0-50	51-100	101-168	169-208	209-748	749+
NH3	0-200	201-400	401-800	801-1200	1201-1800	1801-2000

Using the above breakpoints, we'll update our AQI standards for Delhi:

```
aqi_standards = {
'co': 10 * 1000,    # CO (10 mg/m³ converted to ppb)
'no': 40,          # NO (ppb, approximate guideline)
'no2': 40,         # NO2 (ppb)
'o3': 50,          # O3 (ppb)
'so2': 40,         # SO2 (ppb)
'pm2_5': 60,       # PM2.5 (µg/m³)
'pm10': 100,       # PM10 (µg/m³)
'nh3': 400         # NH3 (ppb)
}
```

[79]:
```
aqi_standards = {
    'co': 10 * 1000,    # CO (10 mg/m³ converted to ppb)
    'no': 40,           # NO (ppb, approximate guideline)
    'no2': 40,          # NO2 (ppb)
    'o3': 50,           # O3 (ppb)
    'so2': 40,          # SO2 (ppb)
    'pm2.5': 60,        # PM2.5 (µg/m³)
    'pm10': 100,        # PM10 (µg/m³)
    'nh3': 400          # NH3 (ppb)
}

# Calculate exceedances
exceedances = data.copy()
for col in aqi_standards:
    exceedances[col] = data[col] - aqi_standards[col]

# Clip negative values to 0 (indicating no exceedance)
exceedances.iloc[:, 1:-4] = exceedances.iloc[:, 1:-4].clip(lower=0)

# Calculate percentage exceedance
percentage_exceedances = (exceedances.iloc[:, 1:9] / pd.Series(aqi_standards)) * 100

# Display the first few rows of exceedances
print(exceedances.head())
print(percentage_exceedances.head())
```

Interpretation:

- The code helps in understanding how often and by how much the concentrations of various pollutants exceed the AQI standards. Here's what each part of the output tells you:

- **Exceedances**: Shows the absolute exceedance values for each pollutant.

- If a value is positive, it indicates the pollutant concentration is above the AQI standard by that amount.
- If a value is 0, it indicates the pollutant concentration is at or below the AQI standard.
- Percentage Exceedance: Shows how much the observed concentrations exceed the AQI standards in percentage terms.

- A higher percentage indicates a greater exceedance relative to the AQI standard.
- A value of 0% indicates no exceedance.

```
                 date   co     no   no2   o3  so2   pm2.5   pm10  nh3  day  \
0 2023-01-01 00:00:00  0.0   0.00  0.00  0.0  0.0  109.29   94.64  0.0    1
1 2023-01-01 01:00:00  0.0   0.00  2.16  0.0  0.0  122.84  111.08  0.0    1
2 2023-01-01 02:00:00  0.0   0.00  3.87  0.0  0.0  160.25  160.68  0.0    1
3 2023-01-01 03:00:00  0.0  15.43  4.55  0.0  0.0  192.90  204.12  0.0    1
4 2023-01-01 04:00:00  0.0  28.84  5.24  0.0  0.0  206.36  222.80  0.0    1

   hour  weekday  weekend
0     0        6        1
1     1        6        1
2     2        6        1
3     3        6        1
4     4        6        1
    co      no     no2   o3  so2       pm2.5    pm10  nh3
0  0.0   0.000   0.000  0.0  0.0  182.150000   94.64  0.0
1  0.0   0.000   5.400  0.0  0.0  204.733333  111.08  0.0
2  0.0   0.000   9.675  0.0  0.0  267.083333  160.68  0.0
3  0.0  38.575  11.375  0.0  0.0  321.500000  204.12  0.0
4  0.0  72.100  13.100  0.0  0.0  343.933333  222.80  0.0
```

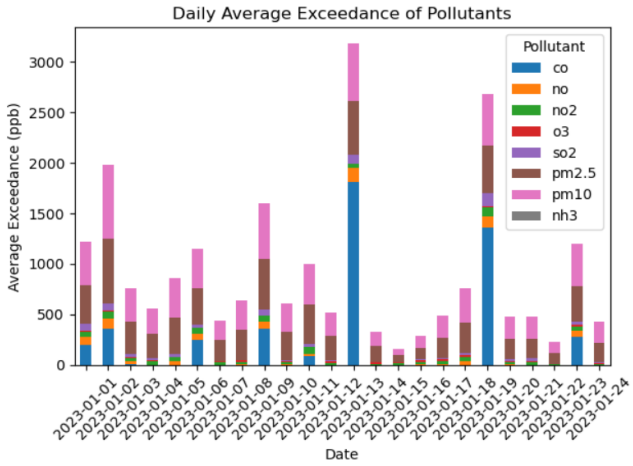
7.Plotting daily average exceedance:

[81]:
```
# Convert 'date' column to datetime type
data['date'] = pd.to_datetime(data['date'])

# Group by date and calculate the mean exceedance for each pollutant
daily_exceedance_mean =exceedances.groupby(data['date'].dt.date).mean()

# Plotting
plt.figure(figsize=(10, 6))
daily_exceedance_mean.iloc[:, 1:-4].plot(kind='bar', stacked=True)
plt.title('Daily Average Exceedance of Pollutants')
plt.xlabel('Date')
plt.ylabel('Average Exceedance (ppb)')
plt.xticks(rotation=45)
plt.legend(title='Pollutant')
plt.tight_layout()
plt.show()
```

<Figure size 1000x600 with 0 Axes>



From the above inference the charts shows pm10 and pm2.5 exceeds higher than normal level followed by so2.

Conclusion:

This project provided an in-depth analysis of air quality in Delhi for January 2023. By utilizing various data processing and visualization techniques, we were able to derive meaningful insights about the levels and patterns of different air pollutants. The following key findings and conclusions were drawn from the analysis:

Pollutant Exceedances:

- The analysis revealed frequent exceedances of AQI standards for pollutants such as CO, NO2, PM2.5, and PM10.
- The percentage exceedance calculation showed that these pollutants often exceeded the standards by significant margins, indicating severe air quality issues.

Hourly Patterns:

- A clear diurnal pattern was observed for several pollutants. Concentrations of CO, NO, NO2, PM2.5, PM10, and NH3 tended to rise in the morning, peaking around 2 to 3 PM, and then gradually decreasing towards the evening.
- This pattern suggests that vehicular emissions and industrial activities significantly contribute to air pollution in Delhi, with higher activity during the day.

Unique Behaviors of O3 and SO2:

- Ozone (O3) and Sulfur Dioxide (SO2) exhibited unique patterns compared to other pollutants. O3 levels often increased during the afternoon due to photochemical reactions driven by sunlight.
- SO2 levels were generally lower and did not follow the same diurnal pattern, indicating different sources or less significant contributions from daily human activities.

Temporal Analysis:

- Time series plots provided a visual representation of pollutant concentrations over time, reinforcing the observed patterns and highlighting periods of particularly high pollution.
- This temporal analysis is crucial for understanding peak pollution times and for planning mitigation strategies.

Policy Implications:

- The findings underscore the urgent need for effective pollution control measures in Delhi. Strategies could include stricter emission standards for vehicles, better regulation of industrial emissions, and promotion of cleaner energy sources.
- Public awareness campaigns and real-time air quality monitoring can help mitigate exposure to harmful pollutants.

Future Work:

- Further analysis could include seasonal variations, comparisons with other cities, and assessment of the impact of specific events or policies on air quality.
- Advanced modeling techniques could be used to predict future pollution levels and to simulate the potential impact of different intervention strategies.

Summary:

In summary, this project successfully highlighted the severe air quality challenges faced by Delhi. By identifying the key pollutants, their patterns, and their exceedance levels, we provided a foundation for informed decision-making and effective policy formulation aimed at improving air quality and public health. Continued monitoring and analysis are essential to track progress and to adapt strategies to emerging trends and challenges in urban air pollution.