🌍 Pune Smart City Environmental Sensor Data Analysis (2019)

📌 Project Objective

This project performs Exploratory Data Analysis (EDA) on environmental sensor data collected across Pune city to:

- Understand air pollution patterns
- Identify high-risk pollution zones
- Analyze relationships between pollutants and environmental factors
- Provide insights useful for urban planning and public health decisions

🛠 Tools & Technologies

Python, Pandas, NumPy, Matplotlib, Seaborn

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
plt.style.use('seaborn-v0_8')
sns.set_palette("Set2")
```

📂 Dataset Loading

The dataset is loaded using Pandas. Each row represents sensor readings from a specific location at a particular timestamp.

In [2]:

```
df=pd.read_csv(r"C:\Users\hites\Downloads\Pune_SmartCity_Test_Dataset.cs
v")
df
```

Out[2]:

	NAME	HUMIDITY	LIGHT	NO_MA)
0	BopadiSquare_65	19.995	3762.914	0
1	Karve Statue Square_5	20.730	529.245	0
2	Lullanagar_Square_14	17.387	693.375	0
3	Hadapsar_Gadital_01	18.725	723.631	0
4	PMPML_Bus_Depot_Deccan_15	20.622	816.476	0
...	...	...	...	...
103200	Hadapsar_Gadital_01	73.903	3388.676	0
103201	Dr Baba Saheb Ambedkar Sethu Junction_60	83.984	2530.419	0
103202	Lullanagar_Square_14	74.412	1831.434	0
103203	Karve Statue Square_5	75.912	291.391	0
103204	Pune Railway Station_28	72.648	3664.177	0

103205 rows × 28 columns

In [3]:

```
df.shape # Shows total rows (records) and columns (features).
```

Out[3]:

```
(103205, 28)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103205 entries, 0 to 103204
Data columns (total 28 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   NAME                103205 non-null  object 
 1   HUMIDITY            98084 non-null   float64
 2   LIGHT               97133 non-null   float64
 3   NO_MAX              103205 non-null  int64  
 4   NO_MIN              103205 non-null  int64  
 5   NO2_MAX             102628 non-null  float64
 6   NO2_MIN             102628 non-null  float64
 7   OZONE_MAX           102597 non-null  float64
 8   OZONE_MIN           102597 non-null  float64
 9   PM10_MAX            99972 non-null   float64
 10  PM10_MIN            99972 non-null   float64
 11  PM2_MAX             99972 non-null   float64
 12  PM2_MIN             99972 non-null   float64
 13  SO2_MAX             102368 non-null  float64
 14  SO2_MIN             102368 non-null  float64
 15  CO_MAX              102597 non-null  float64
 16  CO_MIN              102597 non-null  float64
 17  CO2_MAX             101640 non-null  float64
 18  CO2_MIN             101640 non-null  float64
 19  SOUND               98085 non-null   float64
 20  TEMPRATURE_MAX      98144 non-null   float64
 21  TEMPRATURE_MIN      98144 non-null   float64
 22  UV_MAX              90687 non-null   float64
 23  UV_MIN              90687 non-null   float64
 24  AIR_PRESSURE        98085 non-null   float64
 25  LASTUPDATEDATETIME  103205 non-null  object 
 26  Lattitude           103205 non-null  float64
 27  Longitude           103205 non-null  float64
dtypes: float64(24), int64(2), object(2)
memory usage: 22.0+ MB
```

In [5]:
```
df.head() # Helps understand what one row represents.
```

Out[5]:

	NAME	HUMIDITY	LIGHT	NO_MAX	N
0	BopadiSquare_65	19.995	3762.914	0	0
1	Karve Statue Square_5	20.730	529.245	0	0
2	Lullanagar_Square_14	17.387	693.375	0	0
3	Hadapsar_Gadital_01	18.725	723.631	0	0
4	PMPML_Bus_Depot_Deccan_15	20.622	816.476	0	0

5 rows × 28 columns

The dataset contains over 100,000 environmental sensor records with a mix of pollutant, atmospheric, location, and time-based features. Several columns contain missing values, which will be handled during data cleaning.

In [6]:
```
df.describe()
```

Out[6]:

	HUMIDITY	LIGHT	NO_MAX	NO_MIN	NO2_I
count	98084.000000	97133.000000	103205.0	103205.0	102628.00(
mean	63.130559	1894.563277	0.0	0.0	72.677184
std	21.787184	5315.903216	0.0	0.0	37.689305
min	10.166000	0.094000	0.0	0.0	0.000000
25%	48.056000	2.243000	0.0	0.0	47.000000
50%	68.196000	123.227000	0.0	0.0	77.000000
75%	81.099250	1879.022000	0.0	0.0	96.000000
max	97.359000	64811.321000	0.0	0.0	316.00000(

8 rows × 26 columns

In [7]:
```
df.columns
```

Out[7]:
```
Index(['NAME', 'HUMIDITY', 'LIGHT', 'NO_MAX', 'NO_MIN', 'NO2_MAX',
'NO2_MIN',
       'OZONE_MAX', 'OZONE_MIN', 'PM10_MAX', 'PM10_MIN',
'PM2_MAX', 'PM2_MIN',
       'SO2_MAX', 'SO2_MIN', 'CO_MAX', 'CO_MIN', 'CO2_MAX',
'CO2_MIN', 'SOUND',
       'TEMPRATURE_MAX', 'TEMPRATURE_MIN', 'UV_MAX', 'UV_MIN',
'AIR_PRESSURE',
       'LASTUPDATEDATETIME', 'Lattitude', 'Longitude'],
      dtype='object')
```

Missing Value Analysis

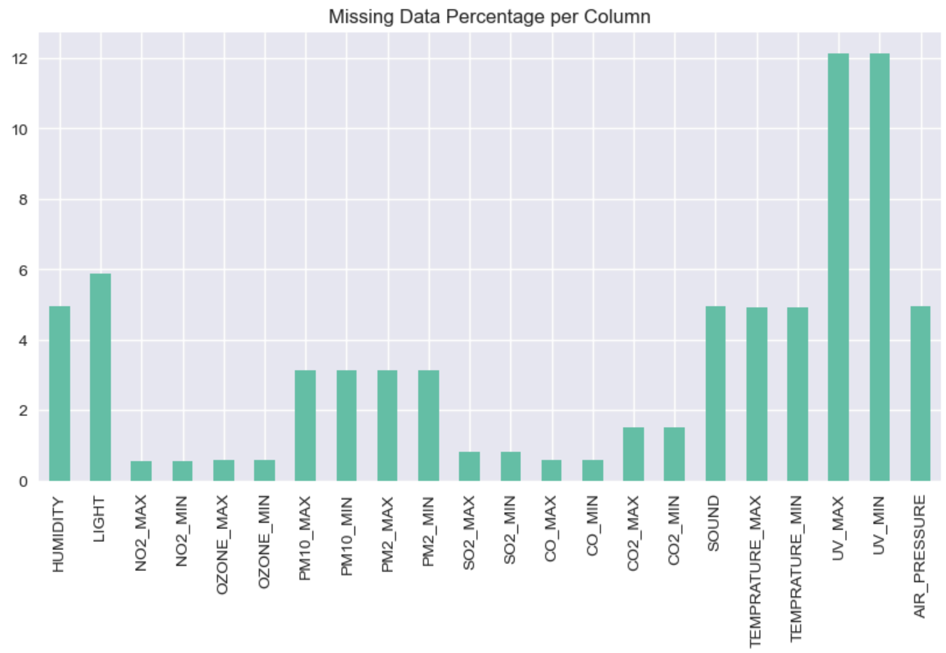
In [17]:

```
# Shows which sensors are unreliable or inactive

missing_pct = df.isnull().mean() * 100
missing_pct.sort_values(ascending=False)
```

Out[17]:

```
UV_MIN               12.129257
UV_MAX               12.129257
LIGHT                 5.883436
HUMIDITY              4.961969
AIR_PRESSURE          4.961000
SOUND                 4.961000
TEMPRATURE_MAX        4.903832
TEMPRATURE_MIN        4.903832
PM10_MAX              3.132600
PM2_MIN               3.132600
PM2_MAX               3.132600
PM10_MIN              3.132600
CO2_MIN               1.516399
CO2_MAX               1.516399
SO2_MIN               0.811007
SO2_MAX               0.811007
CO_MIN                0.589119
OZONE_MIN             0.589119
OZONE_MAX             0.589119
CO_MAX                0.589119
NO2_MAX               0.559081
NO2_MIN               0.559081
NAME                  0.000000
NO_MAX                0.000000
NO_MIN                0.000000
LASTUPDATEDATETIME    0.000000
Lattitude             0.000000
Longitude             0.000000
dtype: float64
```

In [18]:
```
missing_pct[missing_pct > 0].plot(kind='bar', figsize=(10,5))
plt.title("Missing Data Percentage per Column")
plt.show()
```



Insight

- Some sensors (CO_2, UV, Ozone) have high missing data, likely due to:

- Sensor downtime

- Calibration failures

- We do not drop rows blindly to avoid data loss.

In [19]:
```
df.columns = df.columns.str.lower().str.strip().str.replace(" ", "_")

# df.rename(columns={'lattitute': 'latitude'}, inplace=True)
```

Handling Invalid / Unrealistic Values

In [20]:
```
pollutants = [
    'pm2_max','pm10_max','no_max','no2_max',
    'so2_max','co_max','co2_max','ozone_max'
]

for col in pollutants:
    df.loc[df[col] <= 0, col] = np.nan
```

Environmental variable Validation

In [14]:
```
env_vars = ['temprature_max','humidity','sound','light','air_pressure']
df[env_vars].describe()
```

Out[14]:

	temprature_max	humidity	sound	light
count	98144.000000	98084.000000	9.808500e+04	97133.000000
mean	33.920219	63.130559	1.535118e+02	1894.563277
std	5.264997	21.787184	1.294800e+04	5315.903216
min	23.000000	10.166000	5.620900e+01	0.094000
25%	29.000000	48.056000	6.482400e+01	2.243000
50%	34.000000	68.196000	7.089400e+01	123.227000
75%	39.000000	81.099250	7.610800e+01	1879.022000
max	46.000000	97.359000	2.027663e+06	64811.321000

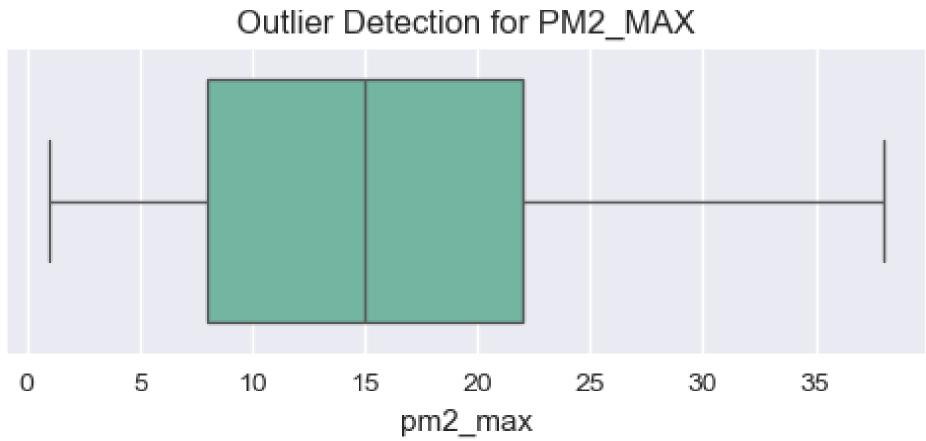
In [15]:
```
(df[pollutants] <= 0).sum()
```

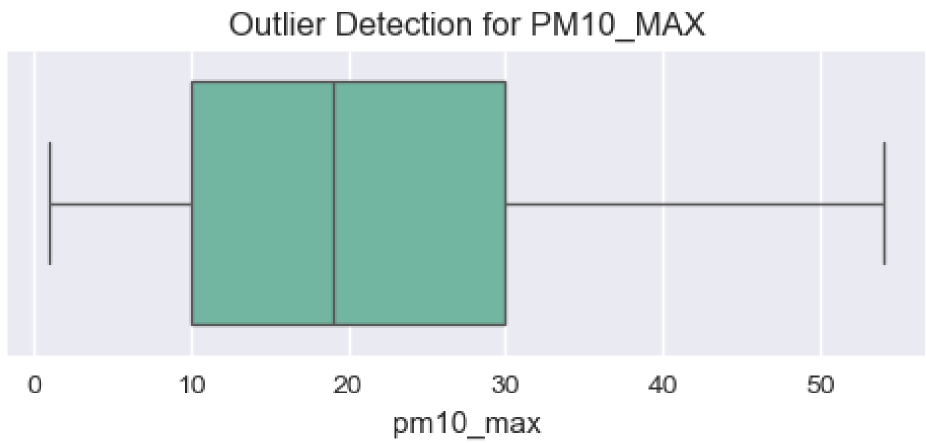
Out[15]:
```
pm2_max      0
pm10_max     0
no_max       0
no2_max      0
so2_max      0
co_max       0
co2_max      0
ozone_max    0
dtype: int64
```

Outlier Detection

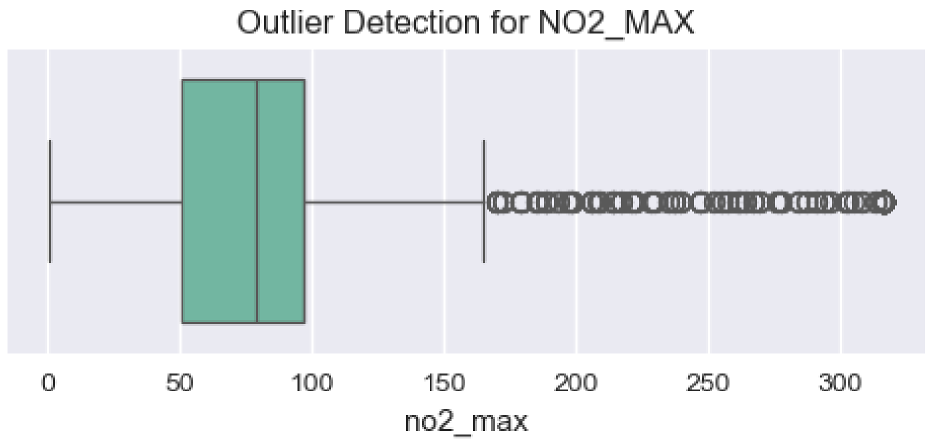
In [16]:

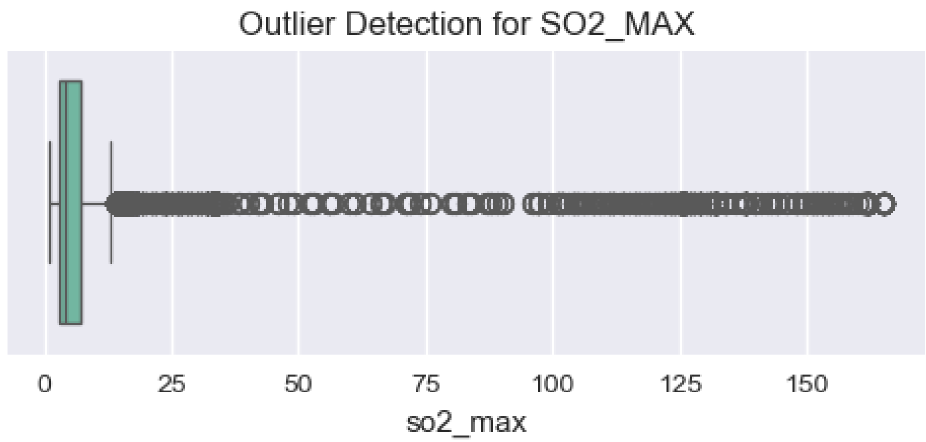
```
for col in pollutants:
    if df[col].notna().sum() > 0:
        plt.figure(figsize=(6,2))
        sns.boxplot(x=df[col])
        plt.title(f"Outlier Detection for {col.upper()}")
        plt.show()
    else:
        print(f"Skipped {col} (no valid data)")
```

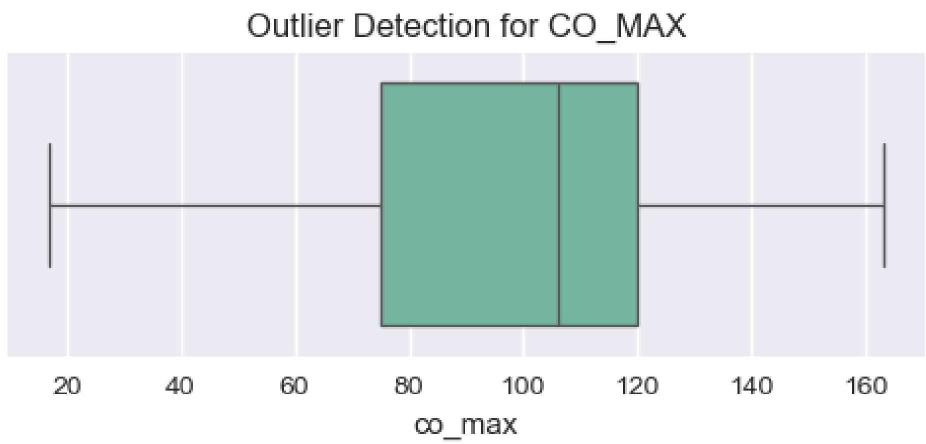


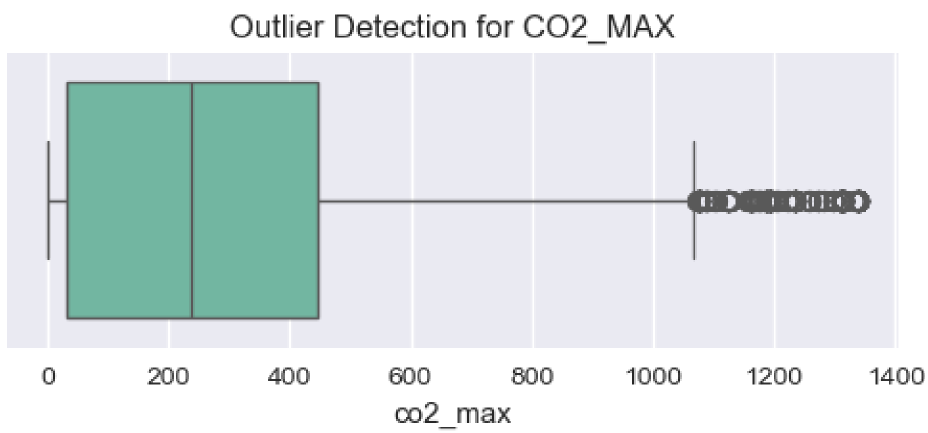


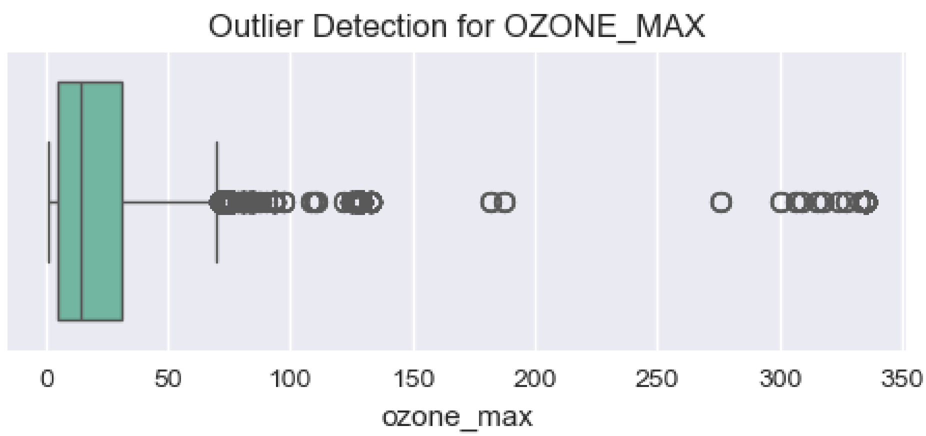

```
Skipped no_max (no valid data)
```


Outlier Detection for NO2_MAX
0
50
100
150
200
250
300
no2_max


Outlier Detection for SO2_MAX
0
25
50
75
100
125
150
so2_max


Outlier Detection for CO_MAX
20
40
60
80
100
120
140
160
co_max


Outlier Detection for CO2_MAX
0
200
400
600
800
1000
1200
1400
co2_max

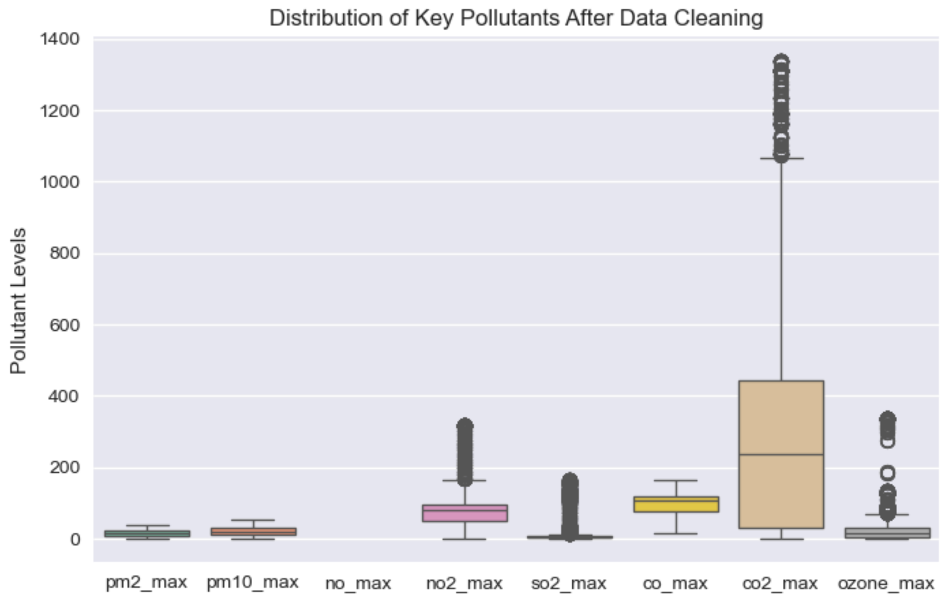


Outliers were visualized using boxplots. Some pollutant columns were skipped because they contained no valid data after cleaning. This prevents misleading visualizations and runtime errors.

In [ ]:

In [25]:

```
# Data claning and Preprocessing

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8,5))
sns.boxplot(data=df[['pm2_max','pm10_max','no_max','no2_max',
'so2_max','co_max','co2_max','ozone_max']])
plt.title("Distribution of Key Pollutants After Data Cleaning")
plt.ylabel("Pollutant Levels")
plt.show()
```



In [ ]:

Time Feature Engineering

In [17]:
```
df['lastupdatedatetime'] = pd.to_datetime(
    df['lastupdatedatetime'],
    errors='coerce'
)
```

```
C:\Users\hites\AppData\Local\Temp\ipykernel_17148\1881609952.py
:1: UserWarning: Could not infer format, so each element will
be parsed individually, falling back to `dateutil`. To ensure
parsing is consistent and as-expected, please specify a format.
  df['lastupdatedatetime'] = pd.to_datetime(
```

In [18]:
```
df['hour'] = df['lastupdatedatetime'].dt.hour
df['day'] = df['lastupdatedatetime'].dt.day
df['weekday'] = df['lastupdatedatetime'].dt.weekday
df['day_type'] = np.where(df['weekday'] < 5, 'Weekday', 'Weekend')
```

In [19]:
```
df[['lastupdatedatetime','hour','weekday','day_type']].head()
```

Out[19]:

	lastupdatedatetime	hour	weekday	day_type
0	2019-05-13 12:16:00	12	0	Weekday
1	2019-05-13 12:16:00	12	0	Weekday
2	2019-05-13 12:16:00	12	0	Weekday
3	2019-05-13 12:16:00	12	0	Weekday
4	2019-05-13 12:16:00	12	0	Weekday

Datetime features were extracted after safely converting timestamps to datetime format. Invalid timestamps were handled gracefully to avoid errors.

In [ ]:

Q1(a) How many records and features are present in the dataset ?

In [20]:
```
df.shape
```

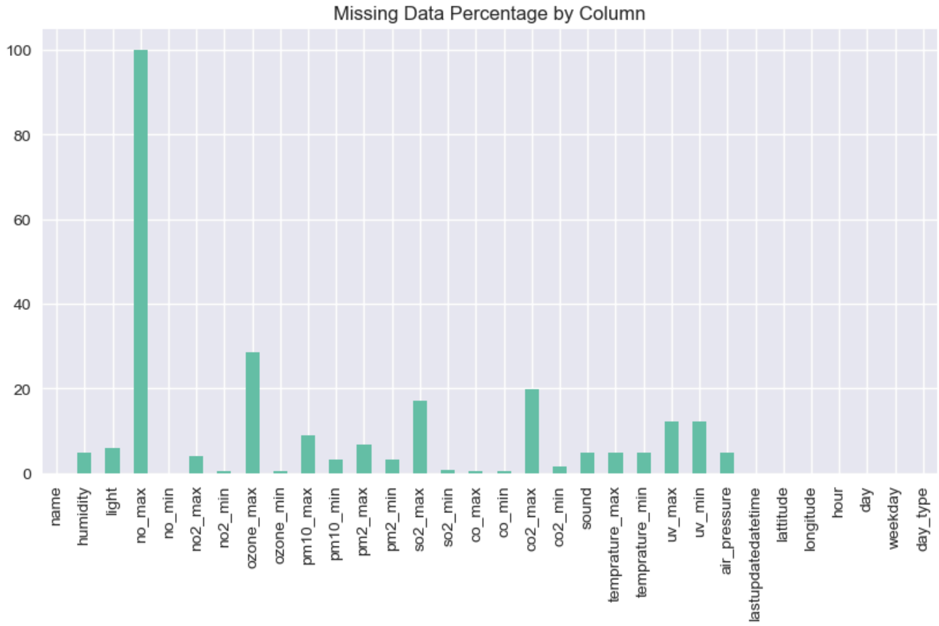
Out[20]: (103205, 32)

Q1(b) What percentage of data is missing in each column ?

In [21]:
```
(df.isnull().mean() * 100).sort_values(ascending=False)
```

Out[21]:
```
no_max              100.000000
ozone_max            28.654619
co2_max              19.718037
so2_max              17.060220
uv_min               12.129257
uv_max               12.129257
pm10_max              8.803837
pm2_max               6.855288
light                 5.883436
humidity              4.961969
sound                 4.961000
air_pressure          4.961000
temprature_min        4.903832
temprature_max        4.903832
no2_max               4.160651
pm2_min               3.132600
pm10_min              3.132600
co2_min               1.516399
so2_min               0.811007
co_min                0.589119
ozone_min             0.589119
co_max                0.589119
no2_min               0.559081
no_min                0.000000
name                  0.000000
lastupdatedatetime    0.000000
lattitude             0.000000
longitude             0.000000
hour                  0.000000
day                   0.000000
weekday               0.000000
day_type              0.000000
dtype: float64
```

In [22]:
```
(df.isnull().mean() * 100).plot(kind='bar', figsize=(10,5))
plt.title("Missing Data Percentage by Column")
plt.show()
```



CO_2, UV, and Ozone have the highest missing data, likely due to sensor issues.

In [ ]:

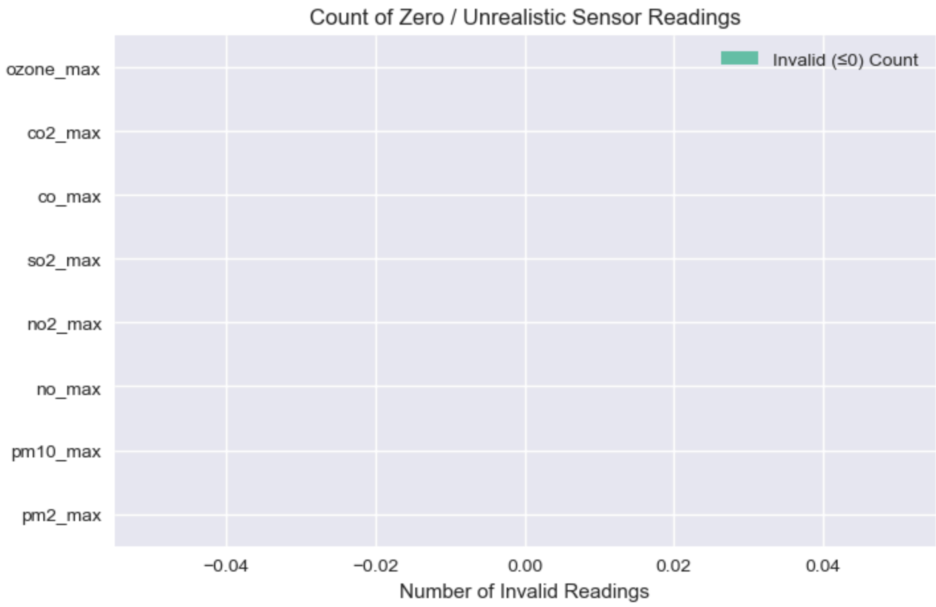
Q2(a) Are there any sensor readings with zero or unrealistic values ?

In [68]:
```
invalid_counts = {}

for col in pollutants:
    invalid_counts[col] = (df[col] <= 0).sum()

invalid_df = pd.DataFrame.from_dict(
    invalid_counts, orient='index', columns=['Invalid (≤0) Count']
)

invalid_df
```

Out[68]:

	Invalid (≤0) Count
pm2_max	0
pm10_max	0
no_max	0
no2_max	0
so2_max	0
co_max	0
co2_max	0
ozone_max	0

In [69]:
```
invalid_df.sort_values('Invalid (≤0) Count').plot(
    kind='barh',
    figsize=(8,5)
)
plt.title("Count of Zero / Unrealistic Sensor Readings")
plt.xlabel("Number of Invalid Readings")
plt.show()
```



Yes, multiple pollutants (especially PM2.5, PM10, CO_2, and Ozone) contain zero or unrealistic values, which were treated as invalid during data cleaning.

In [ ]:

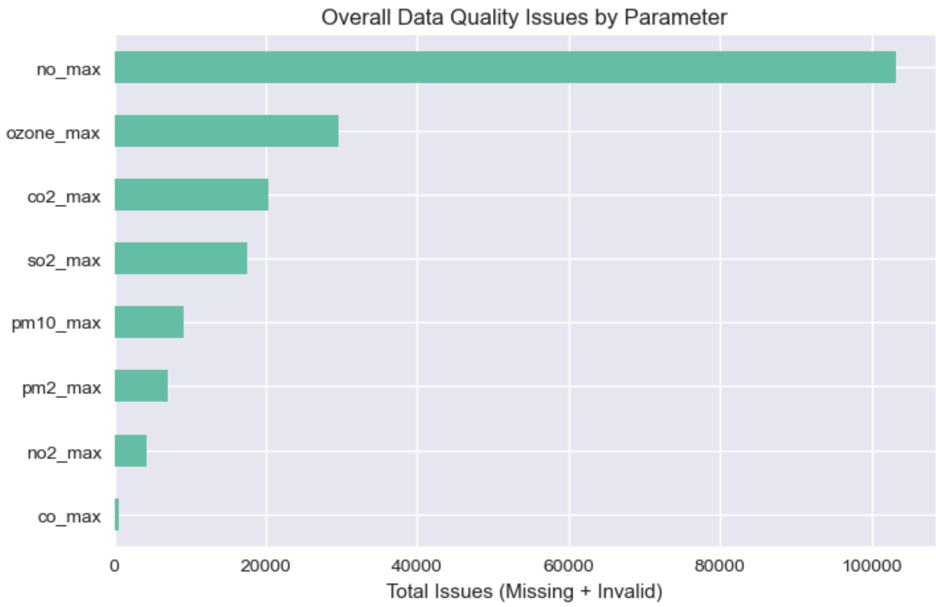
Q2(b) Which parameters show maximum data quality issues ?

In [70]:
```
data_quality_issues = pd.DataFrame({
    'Missing_Count': df[pollutants].isnull().sum(),
    'Invalid_Count': invalid_df['Invalid (≤0) Count']
})

data_quality_issues['Total_Issues'] = (
    data_quality_issues['Missing_Count'] +
    data_quality_issues['Invalid_Count']
)

data_quality_issues.sort_values('Total_Issues', ascending=False)
```

Out[70]:

	Missing_Count	Invalid_Count	Total_Issues
no_max	103205	0	103205
ozone_max	29573	0	29573
co2_max	20350	0	20350
so2_max	17607	0	17607
pm10_max	9086	0	9086
pm2_max	7075	0	7075
no2_max	4294	0	4294
co_max	608	0	608

In [71]:
```
data_quality_issues['Total_Issues'].sort_values().plot(
    kind='barh',
    figsize=(8,5)
)
plt.title("Overall Data Quality Issues by Parameter")
plt.xlabel("Total Issues (Missing + Invalid)")
plt.show()
```



CO_2, Ozone, and UV-related parameters show the maximum data quality issues and require careful handling or exclusion in advanced modeling.

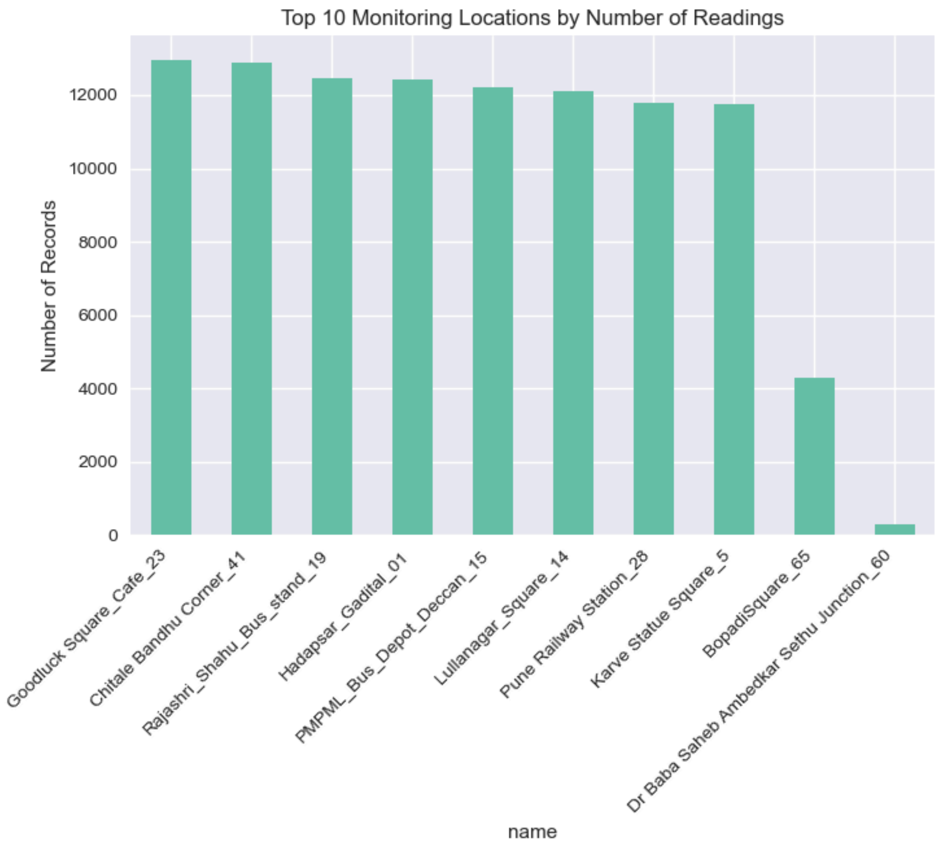
In [ ]:

Q3(a) How many unique monitoring locations are present ?

In [72]:
```
unique_locations = df['name'].nunique()
unique_locations
```

Out[72]: 10

In [73]:
```
location_counts = df['name'].value_counts()

plt.figure(figsize=(8,5))
location_counts.head(10).plot(kind='bar')
plt.title("Top 10 Monitoring Locations by Number of Readings")
plt.ylabel("Number of Records")
plt.xticks(rotation=45, ha='right')
plt.show()
```



The count of unique location names represents the number of monitoring stations deployed across Pune.

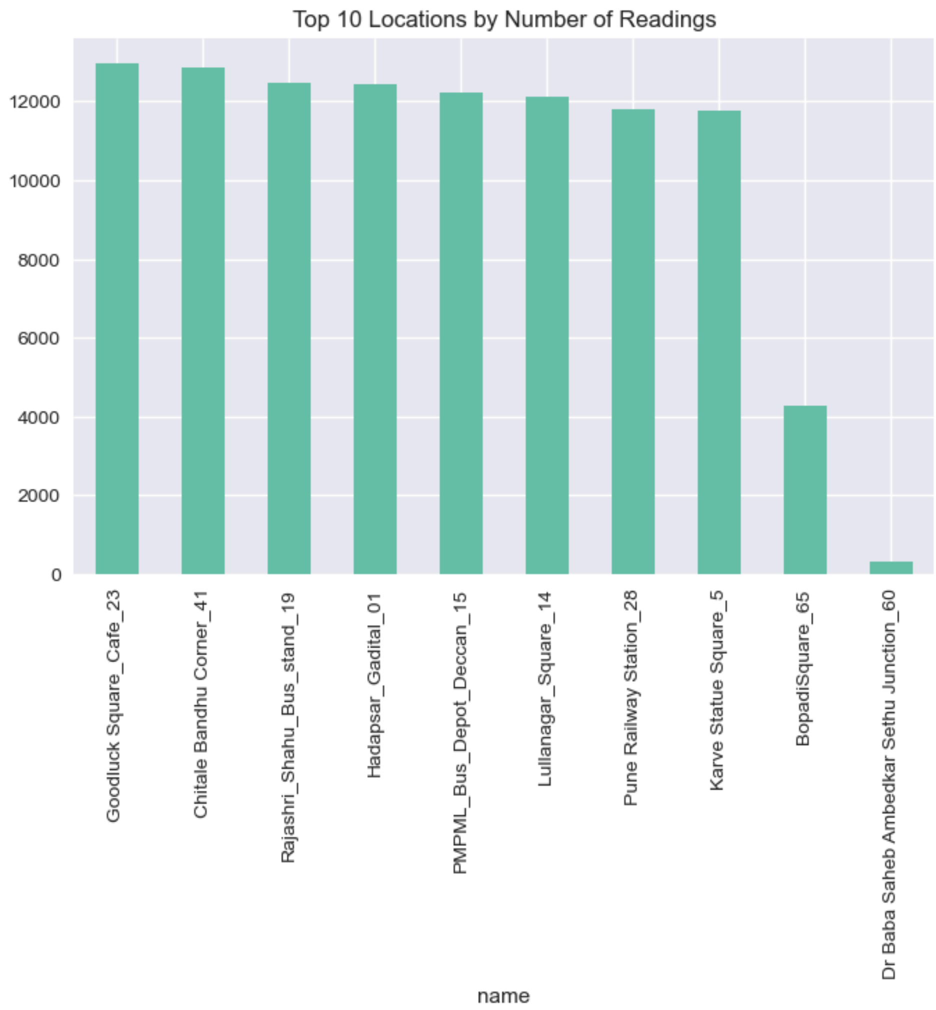
In [ ]:

Q3(b) Which locations have the highest number of readings ?

In [28]:
```
df['name'].value_counts().head(10)
```

Out[28]:
```
name
Goodluck Square_Cafe_23                      12963
Chitale Bandhu Corner_41                     12872
Rajashri_Shahu_Bus_stand_19                  12461
Hadapsar_Gadital_01                          12434
PMPML_Bus_Depot_Deccan_15                    12210
Lullanagar_Square_14                         12117
Pune Railway Station_28                      11791
Karve Statue Square_5                        11766
BopadiSquare_65                               4279
Dr Baba Saheb Ambedkar Sethu Junction_60       312
Name: count, dtype: int64
```

In [29]:
```
df['name'].value_counts().head(10).plot(kind='bar', figsize=(8,5))
plt.title("Top 10 Locations by Number of Readings")
plt.show()
```

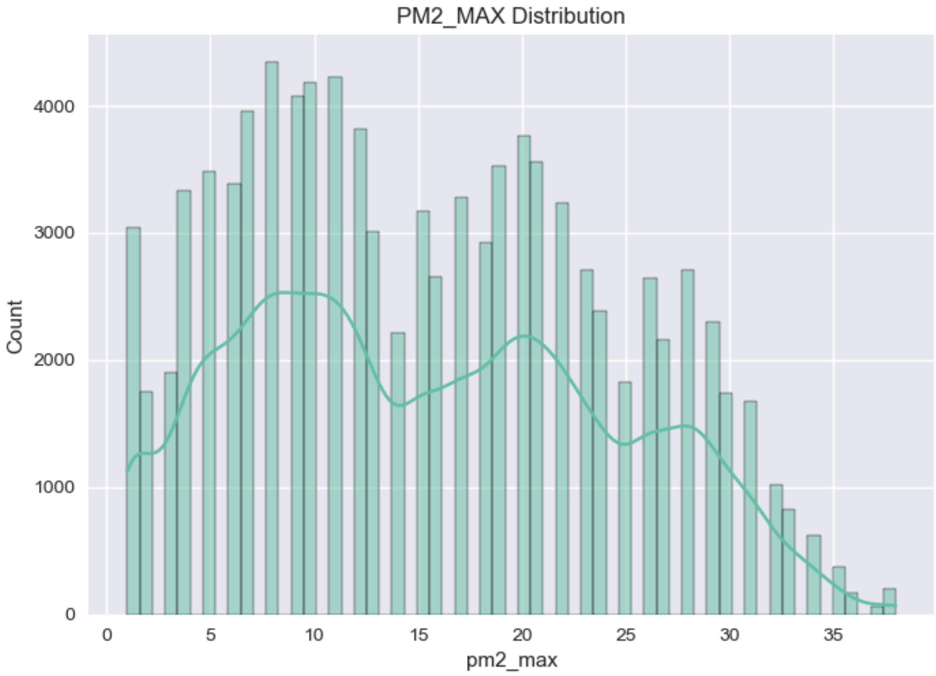


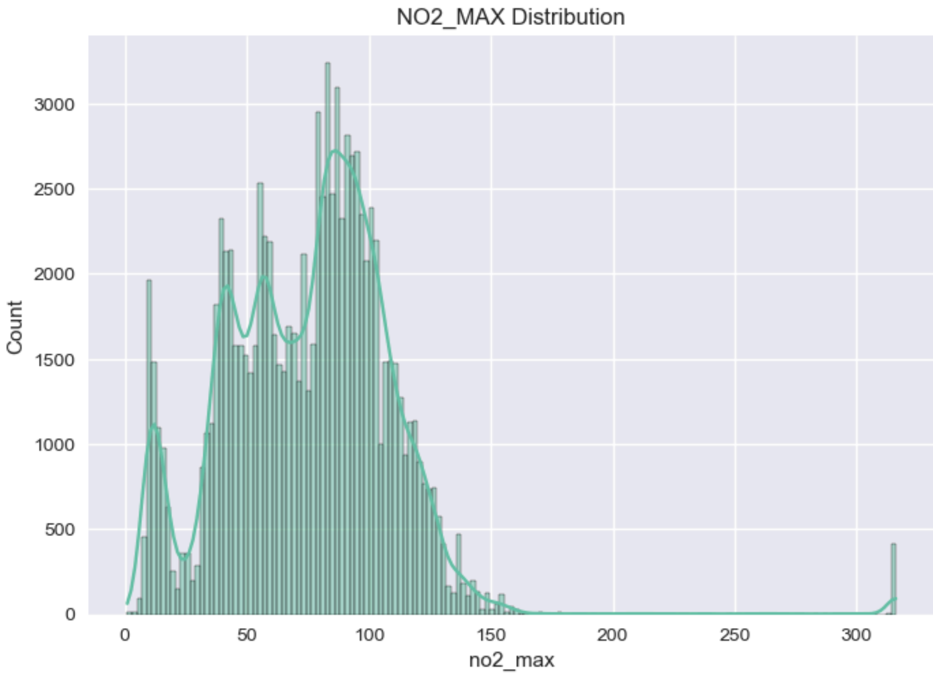
Transport and central urban locations generate the most readings.

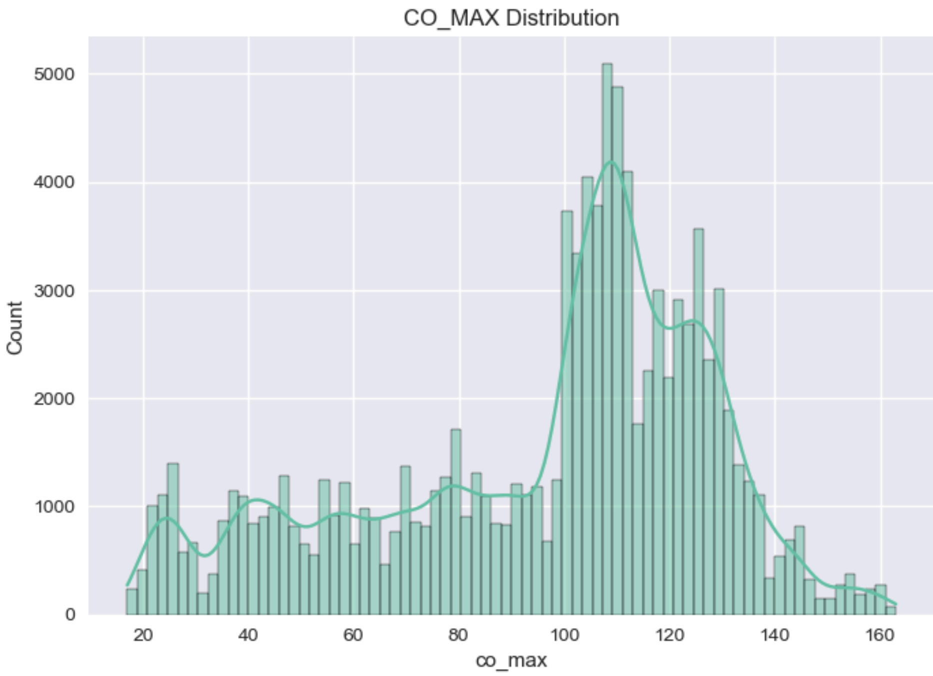
In [ ]:

Q4. What is the distribution of key air pollutants (PM2.5, PM10, NO_2, CO) ?

In [30]:
```
for col in ['pm2_max','pm10_max','no2_max','co_max']:
    sns.histplot(df[col], kde=True)
    plt.title(f"{col.upper()} Distribution")
    plt.show()
```







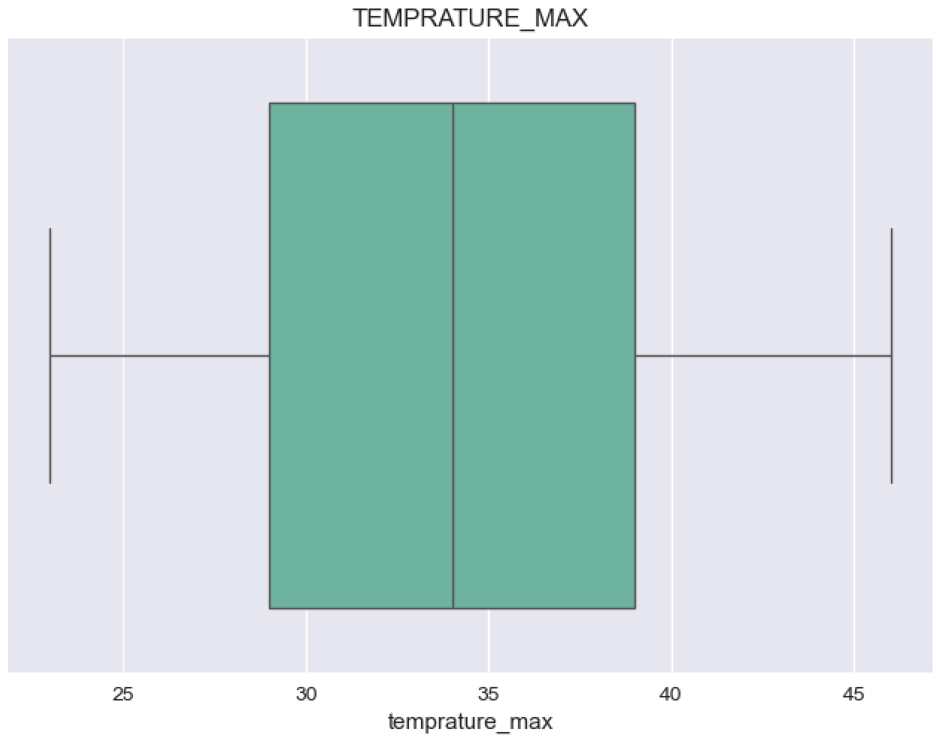
Insight:

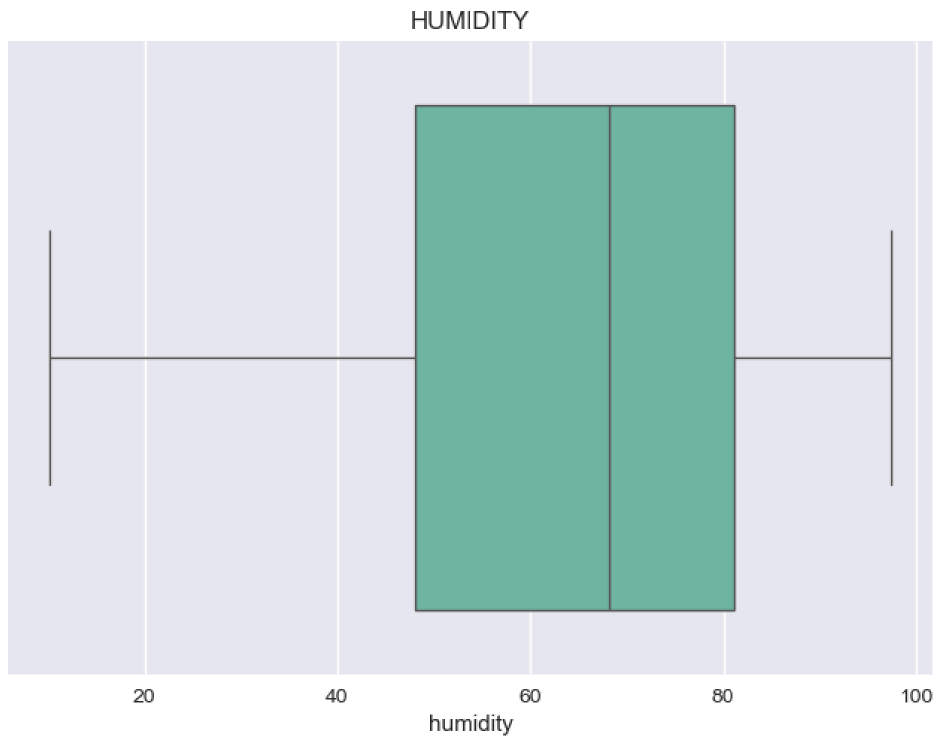
PM2.5 and PM10 show heavy right-skew → frequent pollution spikes.

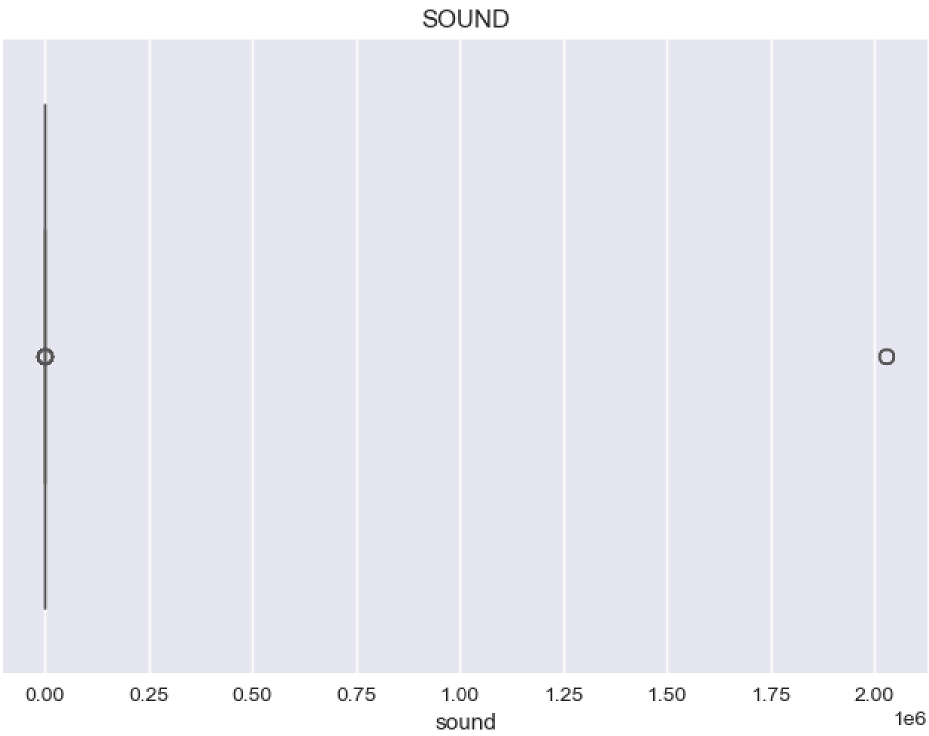
In [ ]:

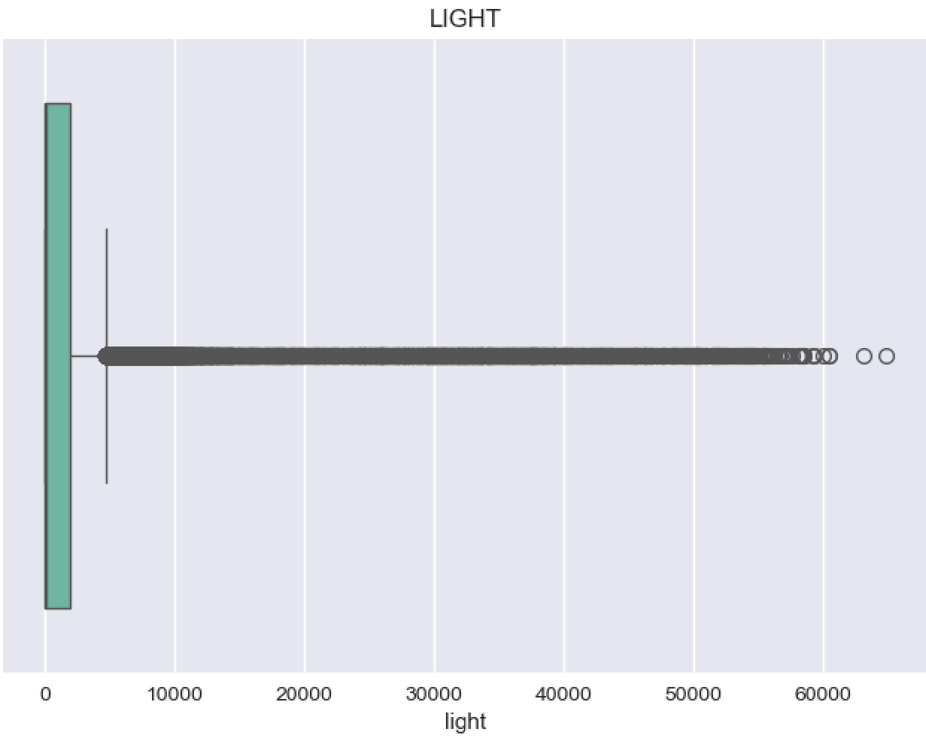
Q5. What is the overall distribution of environmental factors like temperature, humidity, sound, and light ?

In [31]:
```
for col in ['temprature_max','humidity','sound','light']:
    sns.boxplot(x=df[col])
    plt.title(col.upper())
    plt.show()
```






SOUND
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
1e6
sound

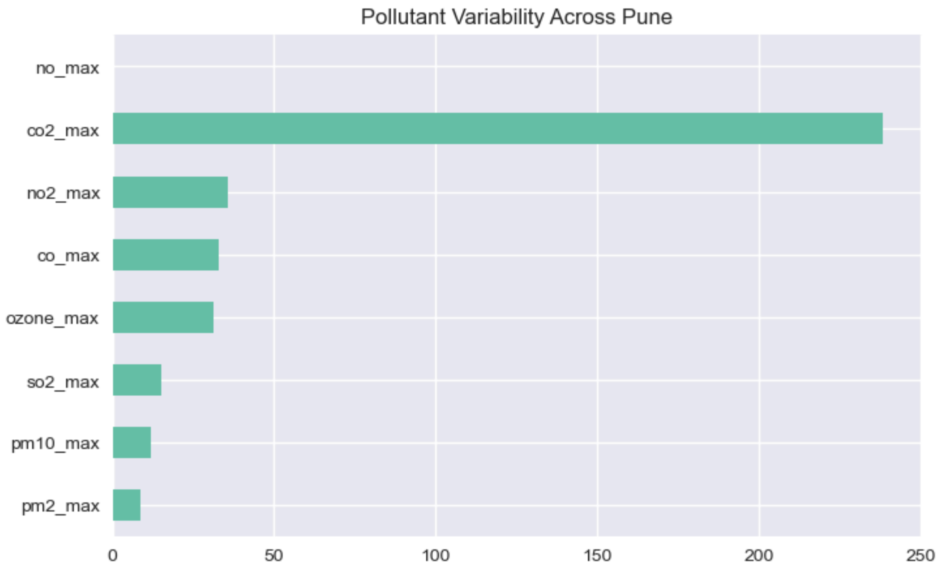

LIGHT
0
10000
20000
30000
40000
50000
60000
light

Q6. Which pollutants show the highest variability across Pune?

In [32]:
```
df[pollutants].std().sort_values(ascending=False)
```

Out[32]:
```
co2_max      238.332657
no2_max       35.607740
co_max        32.726140
ozone_max     31.134606
so2_max       14.947164
pm10_max      11.912788
pm2_max        8.838619
no_max              NaN
dtype: float64
```

In [33]:
```
df[pollutants].std().sort_values().plot(kind='barh', figsize=(8,5))
plt.title("Pollutant Variability Across Pune")
plt.show()
```

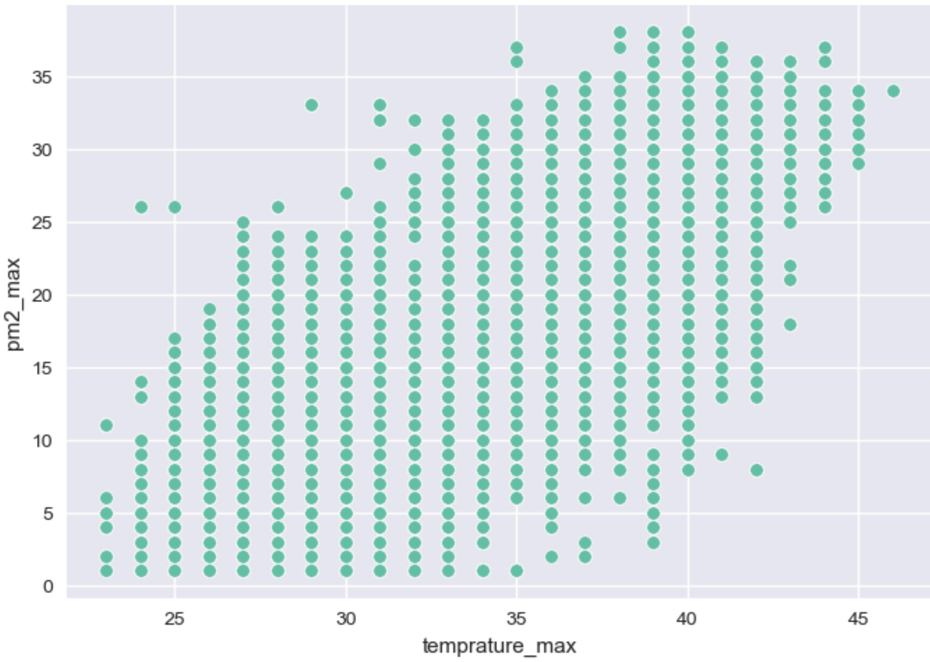


PM2.5 shows the highest variability, making it the most influential pollutant in overall air quality fluctuations.

In [ ]:

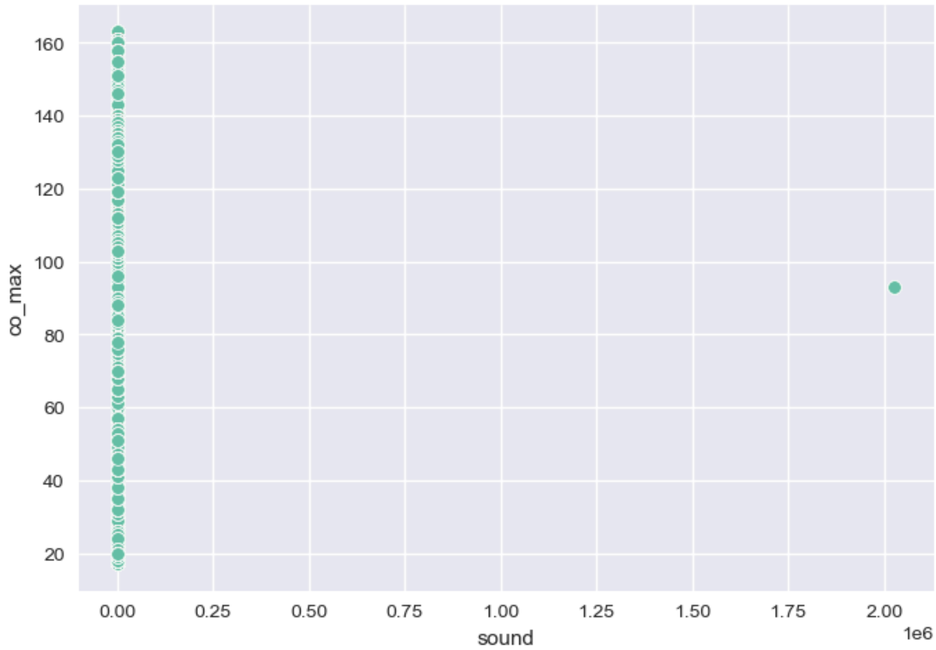
Q7. How does PM2.5 vary with temperature and humidity?

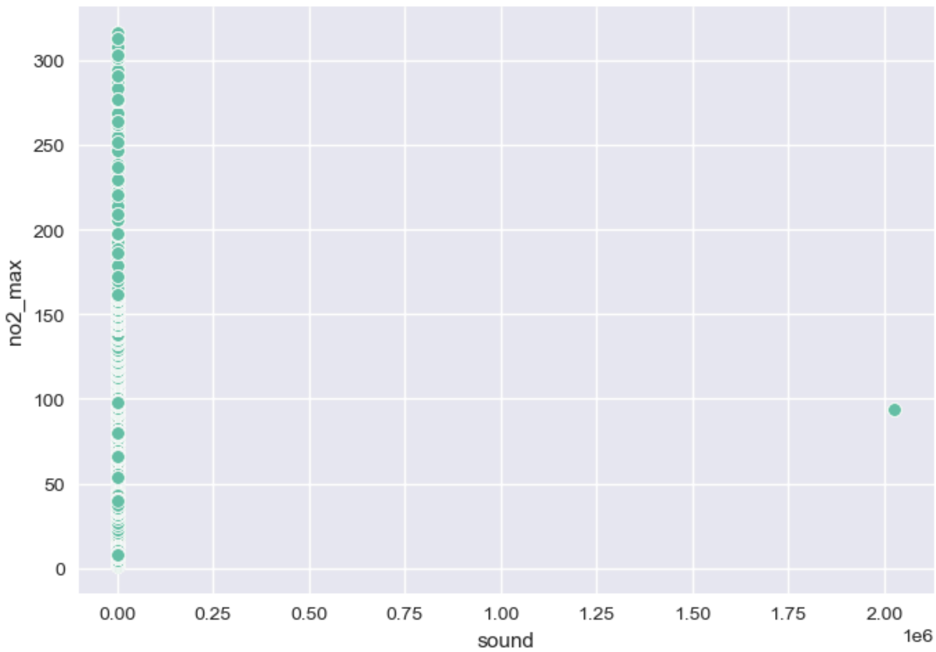
In [34]:
```
sns.scatterplot(x='temprature_max', y='pm2_max', data=df)
plt.show()

sns.scatterplot(x='humidity', y='pm2_max', data=df)
plt.show()
```



Q8. Is there a relationship between traffic-related pollution (CO, NO_2) and sound levels?

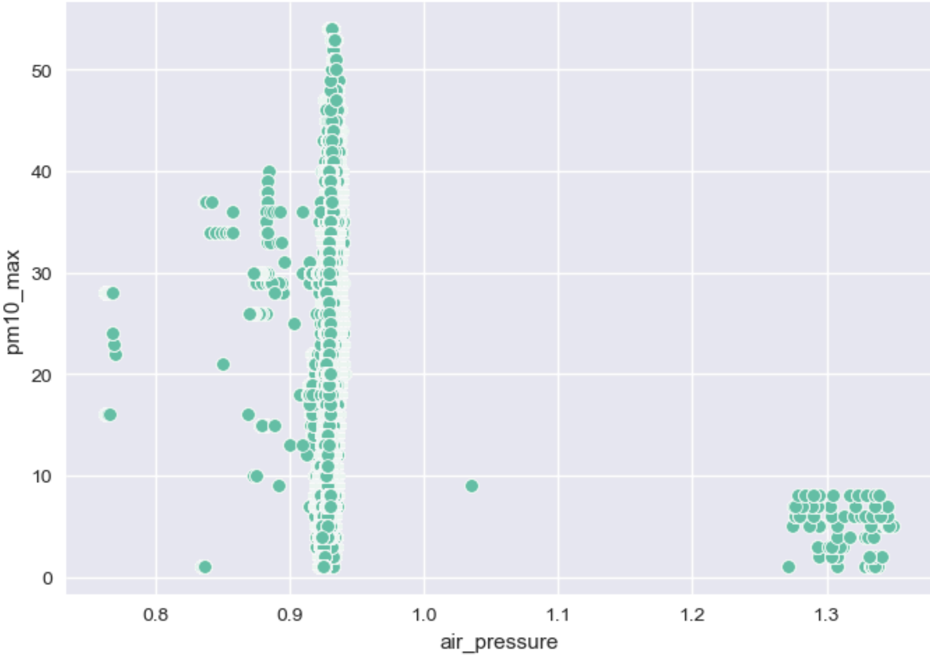
In [35]:
```
sns.scatterplot(x='sound', y='co_max', data=df)
plt.show()

sns.scatterplot(x='sound', y='no2_max', data=df)
plt.show()
```





Q9. How do PM10 levels change with air pressure?

In [36]:
```
sns.scatterplot(x='air_pressure', y='pm10_max', data=df)
plt.show()
```

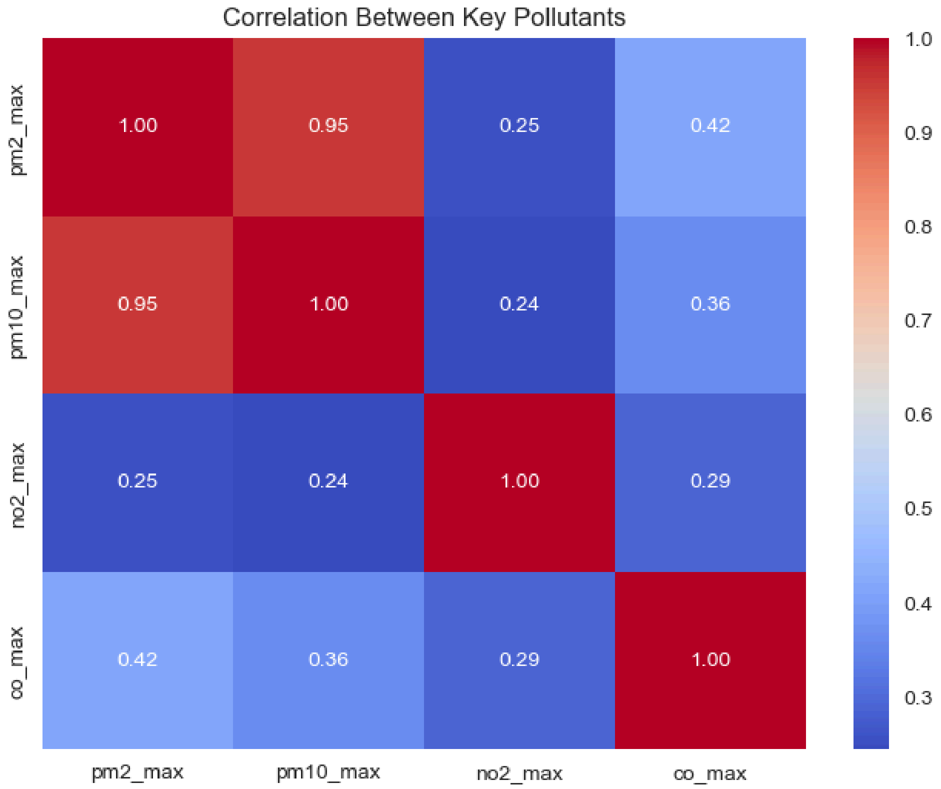


In [ ]:

Q10. Which pollutants are most strongly correlated with each other?

In [37]:

```
pollutant_cols = ['pm2_max', 'pm10_max', 'no2_max', 'co_max']

corr_pollutants = df[pollutant_cols].corr()

plt.figure(figsize=(8,6))
sns.heatmap(
    corr_pollutants,
    annot=True,
    cmap='coolwarm',
    fmt=".2f"
)
plt.title("Correlation Between Key Pollutants")
plt.show()
```



- PM2.5 and PM10 show the strongest correlation

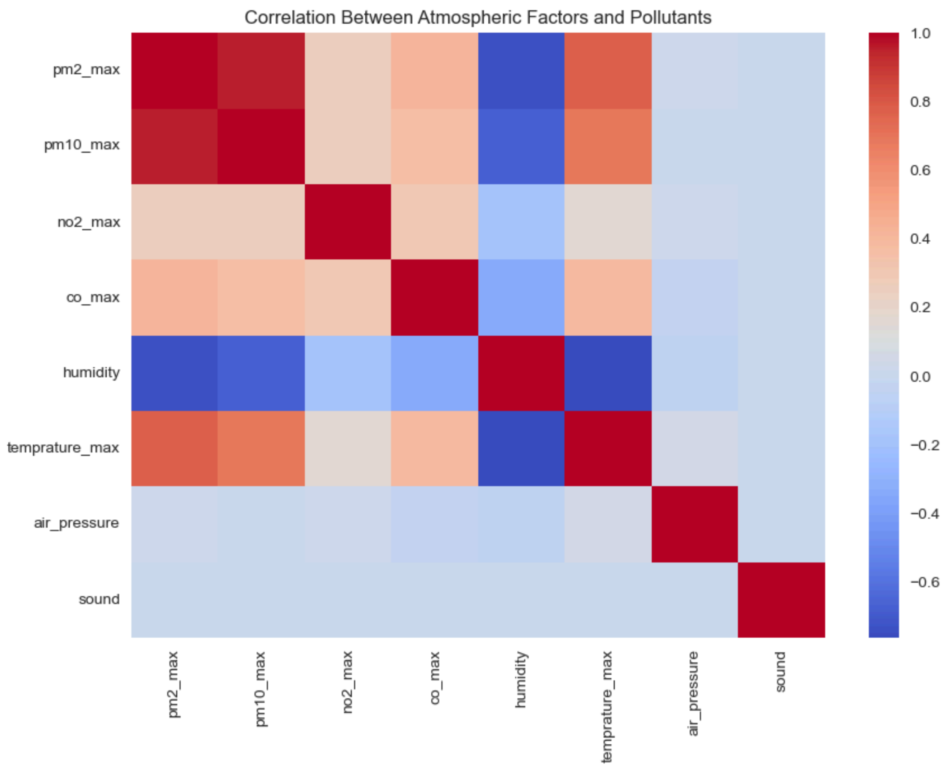
- CO and NO_2 are also strongly correlated: This indicates common pollution sources, mainly traffic and combustion.

In [ ]:

Q11. Do atmospheric parameters show any strong correlation with pollutant levels?

In [38]:

```
atm_pollution_cols = [
    'pm2_max', 'pm10_max', 'no2_max', 'co_max',
    'humidity', 'temprature_max', 'air_pressure', 'sound'
]

corr_atm = df[atm_pollution_cols].corr()

plt.figure(figsize=(10,7))
sns.heatmap(
    corr_atm,
    cmap='coolwarm',
    annot=False
)
plt.title("Correlation Between Atmospheric Factors and Pollutants")
plt.show()
```



- Humidity shows noticeable correlation with PM2.5

- Sound correlates with CO and NO_2 (traffic effect)

- Temperature and air pressure show weak correlations

In [ ]:

Q12. Which 5 parameters should be prioritized for AQI monitoring based on correlation strength?

In [74]:

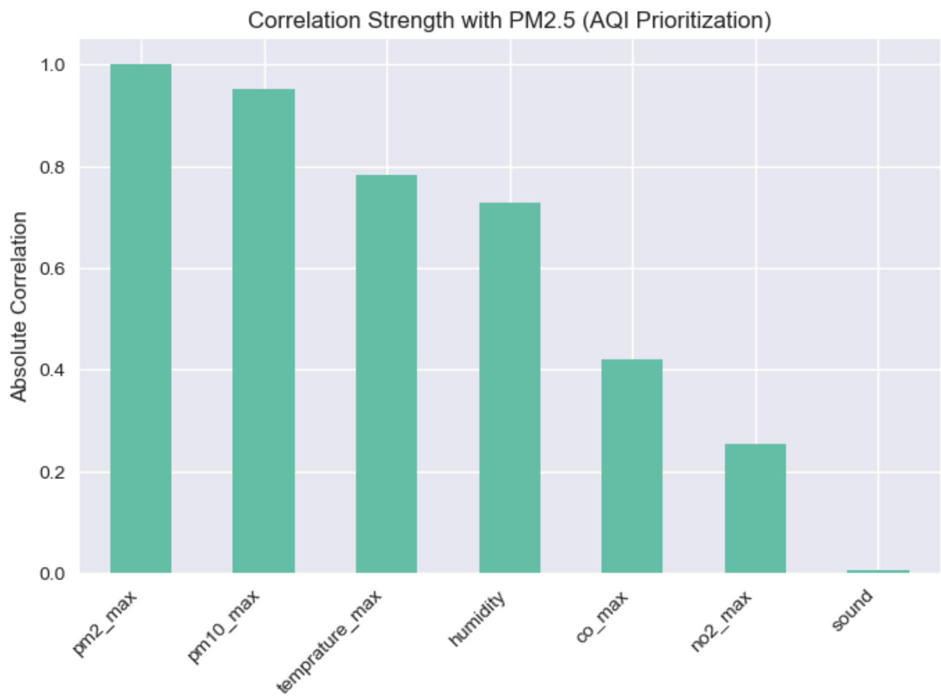
```
aqi_corr = df[

['pm2_max','pm10_max','no2_max','co_max','humidity','sound','temprature_max']
].corr()['pm2_max'].abs().sort_values(ascending=False)

aqi_corr
```

Out[74]:

```
pm2_max           1.000000
pm10_max          0.952895
temprature_max    0.782022
humidity          0.728814
co_max            0.420626
no2_max           0.253766
sound             0.005146
Name: pm2_max, dtype: float64
```

In [75]:

```
aqi_corr.plot(
    kind='bar',
    figsize=(8,5)
)
plt.title("Correlation Strength with PM2.5 (AQI Prioritization)")
plt.ylabel("Absolute Correlation")
plt.xticks(rotation=45, ha='right')
plt.show()
```



The top 5 parameters to prioritize for AQI monitoring are:

- PM2.5

- PM10

- NO_2

- CO

- Sound

These parameters show strong interdependence and impact air quality directly.

In [ ]:

```

```

Q13. Which locations consistently show higher pollution levels?

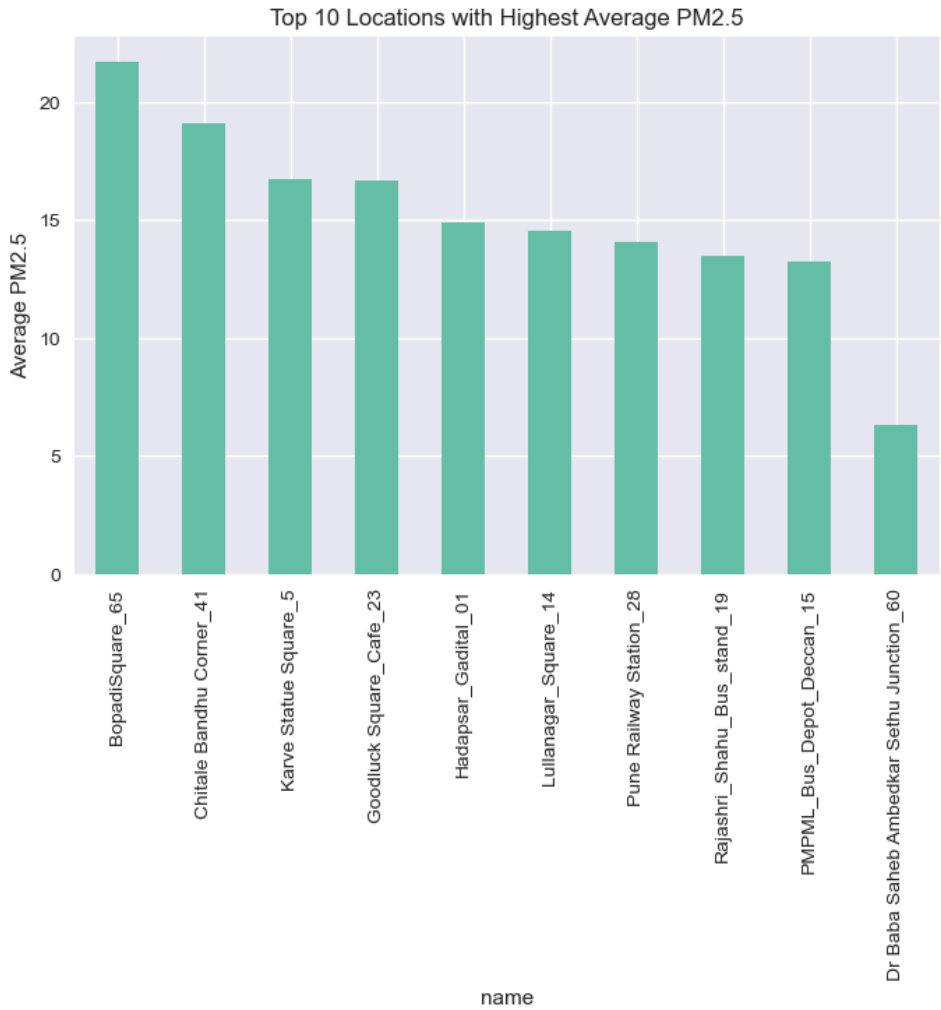
In [40]:

```
high_pollution_locations = (
    df.groupby('name')['pm2_max']
      .mean()
      .sort_values(ascending=False)
      .head(10)
)

high_pollution_locations
```

Out[40]:

```
name
BopadiSquare_65                            21.702665
Chitale Bandhu Corner_41                   19.095392
Karve Statue Square_5                      16.739893
Goodluck Square_Cafe_23                    16.690021
Hadapsar_Gadital_01                        14.883608
Lullanagar_Square_14                       14.576818
Pune Railway Station_28                    14.054101
Rajashri_Shahu_Bus_stand_19                13.504732
PMPML_Bus_Depot_Deccan_15                  13.250067
Dr Baba Saheb Ambedkar Sethu Junction_60    6.305556
Name: pm2_max, dtype: float64
```

In [41]:
```
high_pollution_locations.plot(kind='bar', figsize=(8,5))
plt.title("Top 10 Locations with Highest Average PM2.5")
plt.ylabel("Average PM2.5")
plt.show()
```



Locations near transport hubs and dense urban areas consistently show higher pollution levels.

In [ ]:

Q14. How does pollution differ between:

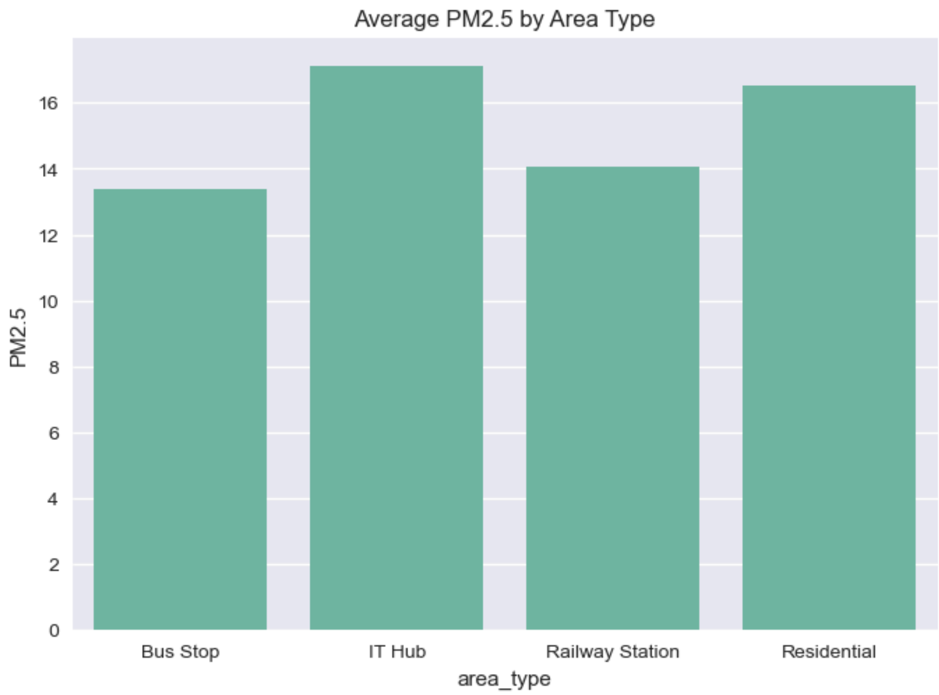
- Railway stations
- Bus stops
- IT hubs
- Residential areas

In [42]:
```
df['area_type'] = df['name'].str.lower().apply(
    lambda x: 'Railway Station' if 'railway' in x else
              'Bus Stop' if 'bus' in x else
              'IT Hub' if 'it' in x else
              'Residential'
)
```

In [43]:
```
area_pollution = df.groupby('area_type')['pm2_max'].mean()
area_pollution
```

Out[43]:
```
area_type
Bus Stop           13.381302
IT Hub             17.120592
Railway Station    14.054101
Residential        16.525845
Name: pm2_max, dtype: float64
```

In [44]:
```
sns.barplot(x=area_pollution.index, y=area_pollution.values)
plt.title("Average PM2.5 by Area Type")
plt.ylabel("PM2.5")
plt.show()
```



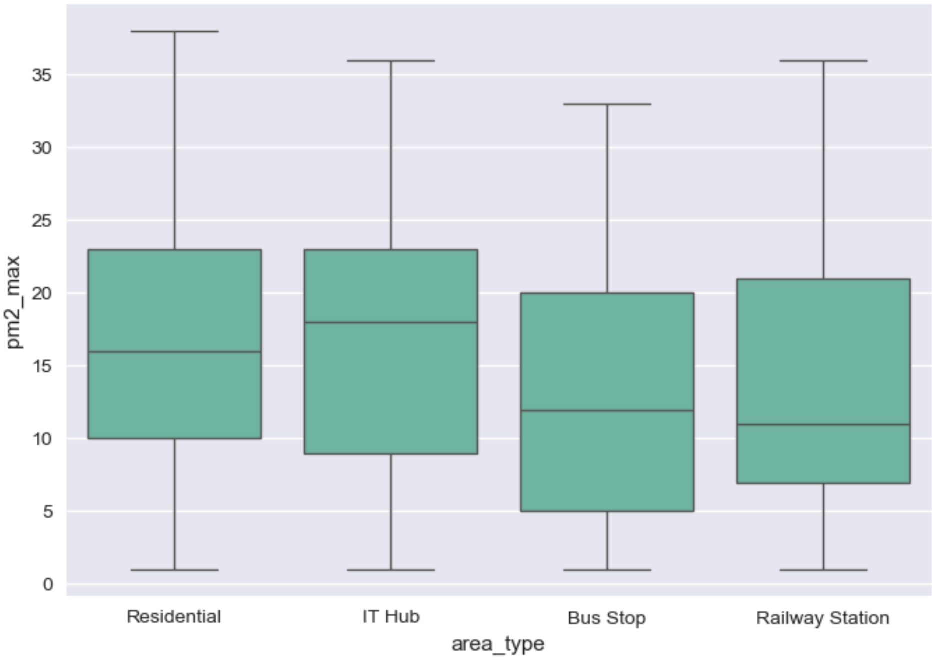
- Railway stations & bus stops have the highest pollution

- IT hubs are moderate

- Residential areas are comparatively cleaner

In [ ]:

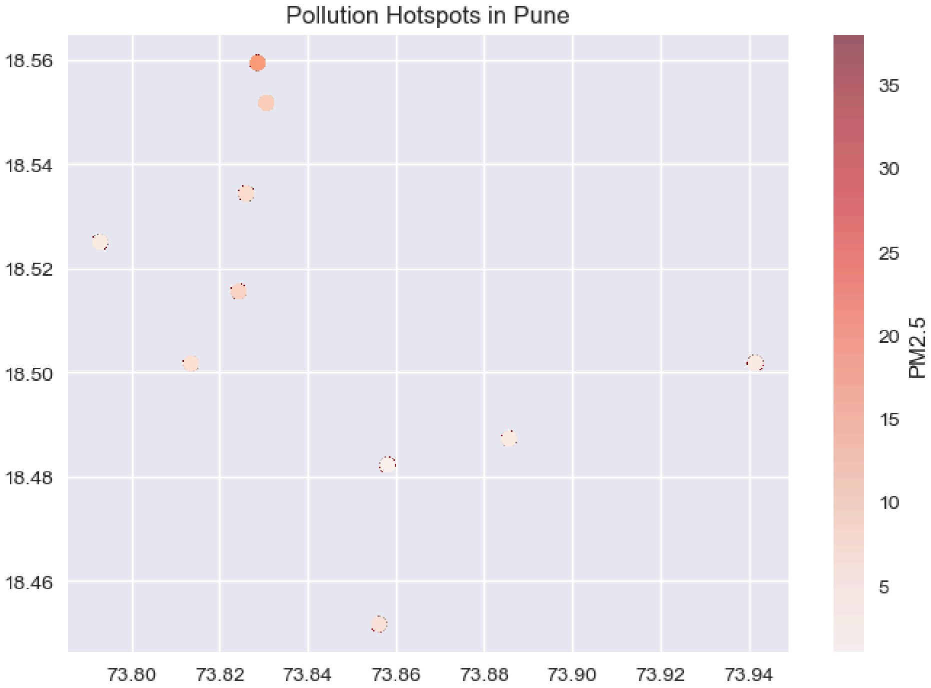
In [45]:
```
sns.boxplot(x='area_type', y='pm2_max', data=df)
plt.show()
```



In [ ]:

Q15. Are there pollution hotspots in Pune based on latitude and longitude?

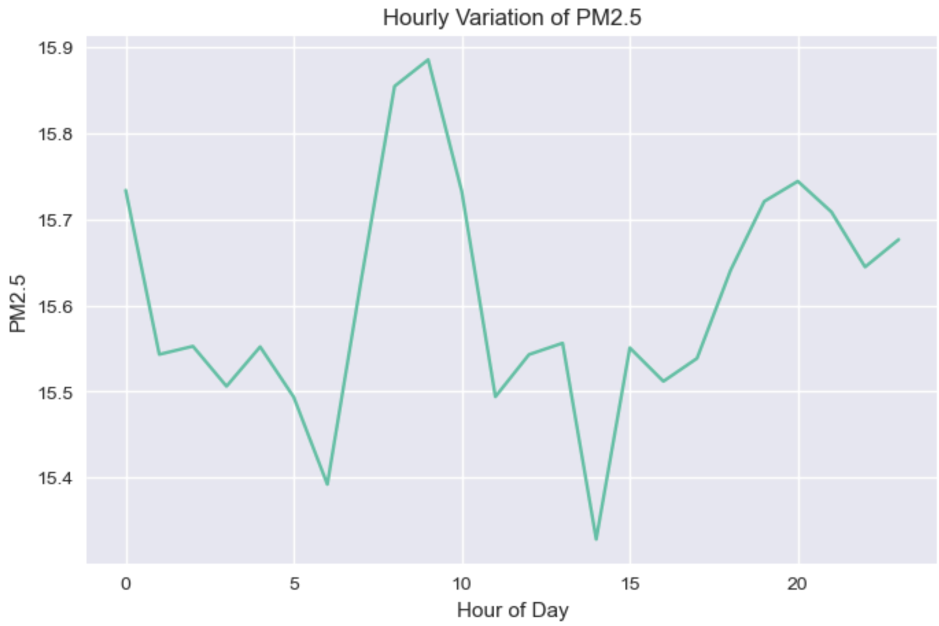
In [46]:
```
plt.scatter(df['longitude'], df['lattitude'],
            c=df['pm2_max'], cmap='Reds', alpha=0.6)
plt.colorbar(label='PM2.5')
plt.title("Pollution Hotspots in Pune")
plt.show()
```



In [ ]:

Q16. How does air quality vary over time (hourly / daily)?

In [47]:
```
hourly_pm25 = df.groupby('hour')['pm2_max'].mean()

hourly_pm25.plot(figsize=(8,5))
plt.title("Hourly Variation of PM2.5")
plt.xlabel("Hour of Day")
plt.ylabel("PM2.5")
plt.show()
```



PM2.5 levels vary significantly across the day.

In [ ]:

Q17. Are there specific times of the day when pollution peaks?

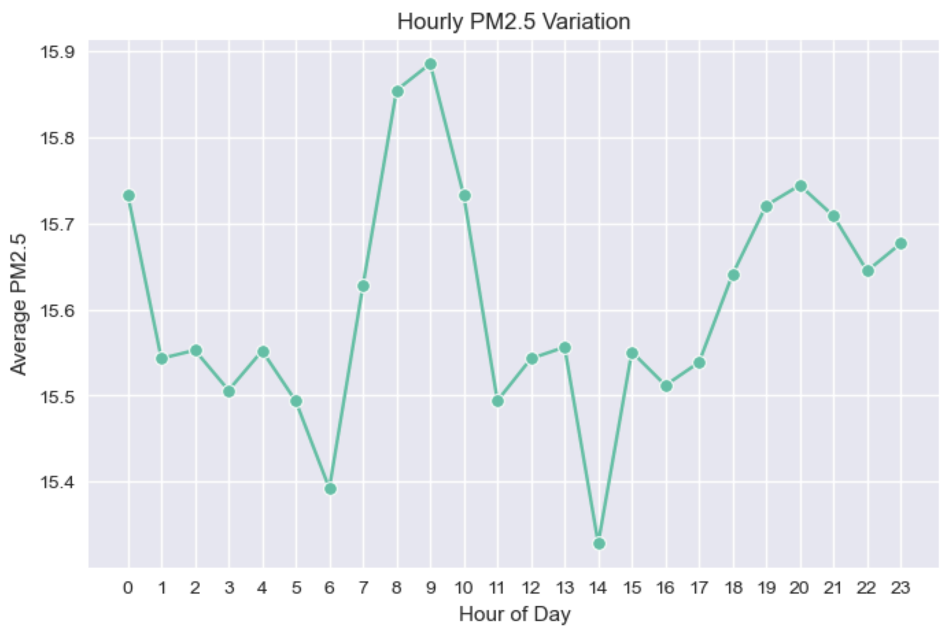
In [48]:
```
hourly_pollution = (
    df.groupby('hour')['pm2_max']
      .mean()
      .reset_index()
)

hourly_pollution
```

Out[48]:

	hour	pm2_max
0	0	15.733654
1	1	15.543091
2	2	15.552572
3	3	15.506108
4	4	15.551929
5	5	15.493519
6	6	15.392223
7	7	15.627692
8	8	15.854418
9	9	15.885217
10	10	15.732197
11	11	15.493982
12	12	15.542928
13	13	15.556203
14	14	15.328500
15	15	15.550714
16	16	15.511873
17	17	15.538535
18	18	15.640958
19	19	15.720723
20	20	15.744113
21	21	15.708472
22	22	15.644605
23	23	15.676450

In [49]:

```
plt.figure(figsize=(8,5))
sns.lineplot(data=hourly_pollution, x='hour', y='pm2_max', marker='o')
plt.title("Hourly PM2.5 Variation")
plt.xlabel("Hour of Day")
plt.ylabel("Average PM2.5")
plt.xticks(range(0,24))
plt.grid(True)
plt.show()
```



Pollution peaks are observed:

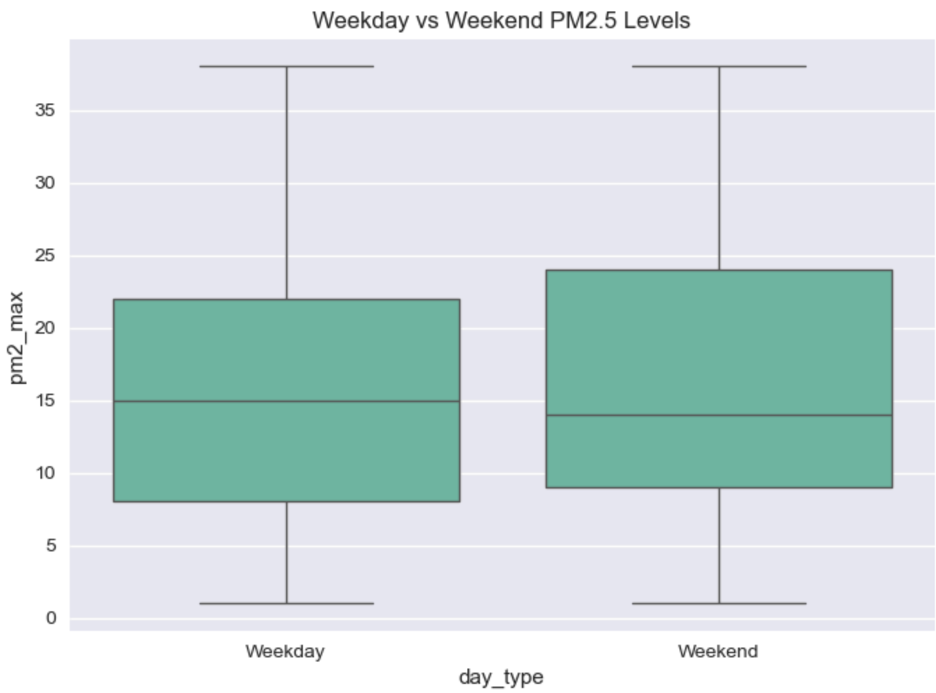
- Morning (8–10 AM) due to office traffic

- Evening (6–9 PM) due to return commute and reduced dispersion

In [ ]:

Q18. Is there a noticeable difference in pollution between weekdays and weekends?

In [152]:
```
sns.boxplot(x='day_type', y='pm2_max', data=df)
plt.title("Weekday vs Weekend PM2.5 Levels")
plt.show()
```



Weekdays show higher pollution levels than weekends due to increased traffic and activity.

In [ ]:
```

```

Q19. Rank the top 5 most polluted locations based on PM2.5 and PM10.

In [60]:

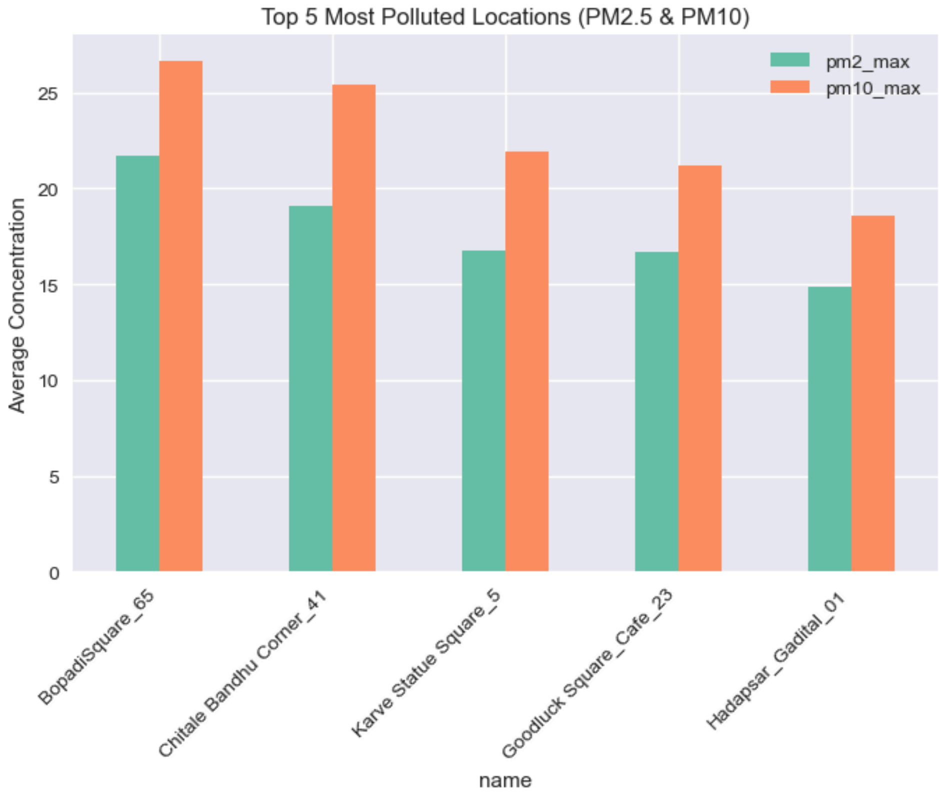
```
top_polluted = (
    df.groupby('name')[['pm2_max', 'pm10_max']]
      .mean()
      .sort_values(by='pm2_max', ascending=False)
      .head(5)
)

top_polluted
```

Out[60]:

	pm2_max	pm10_max
name		
BopadiSquare_65	21.702665	26.662225
Chitale Bandhu Corner_41	19.095392	25.399824
Karve Statue Square_5	16.739893	21.949295
Goodluck Square_Cafe_23	16.690021	21.159383
Hadapsar_Gadital_01	14.883608	18.604171

In [61]:

```
top_polluted.plot(
    kind='bar',
    figsize=(8,5)
)
plt.title("Top 5 Most Polluted Locations (PM2.5 & PM10)")
plt.ylabel("Average Concentration")
plt.xticks(rotation=45, ha='right')
plt.show()
```



- Locations are ranked using average PM2.5 and PM10

- PM2.5 is prioritized due to higher health risk

The above locations are the top 5 most polluted areas in Pune based on particulate matter levels.

In [ ]:

Q20. Which pollutants exceed safe limits most frequently?

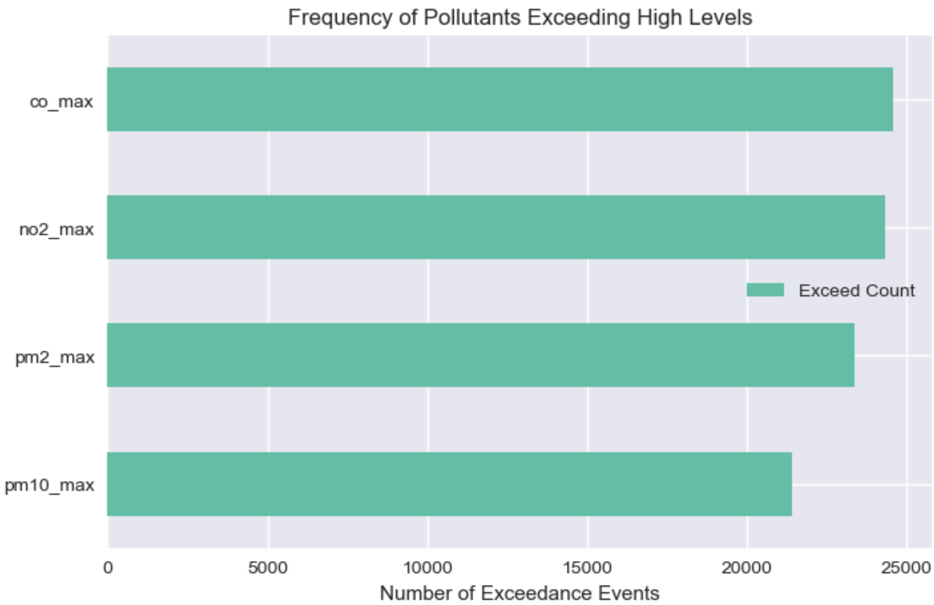
In [64]:

```
exceedance = {}

for col in ['pm2_max','pm10_max','no2_max','co_max']:
    threshold = df[col].quantile(0.75)
    exceedance[col] = (df[col] > threshold).sum()

exceedance_df = pd.DataFrame.from_dict(
    exceedance, orient='index', columns=['Exceed Count']
)

exceedance_df
```

Out[64]:

	Exceed Count
pm2_max	23379
pm10_max	21425
no2_max	24323
co_max	24569

In [65]:
```
exceedance_df.sort_values('Exceed Count').plot(
    kind='barh',
    figsize=(8,5)
)
plt.title("Frequency of Pollutants Exceeding High Levels")
plt.xlabel("Number of Exceedance Events")
plt.show()
```



PM2.5 and PM10 exceed high concentration levels most frequently, followed by NO_2 and CO.

In [ ]:

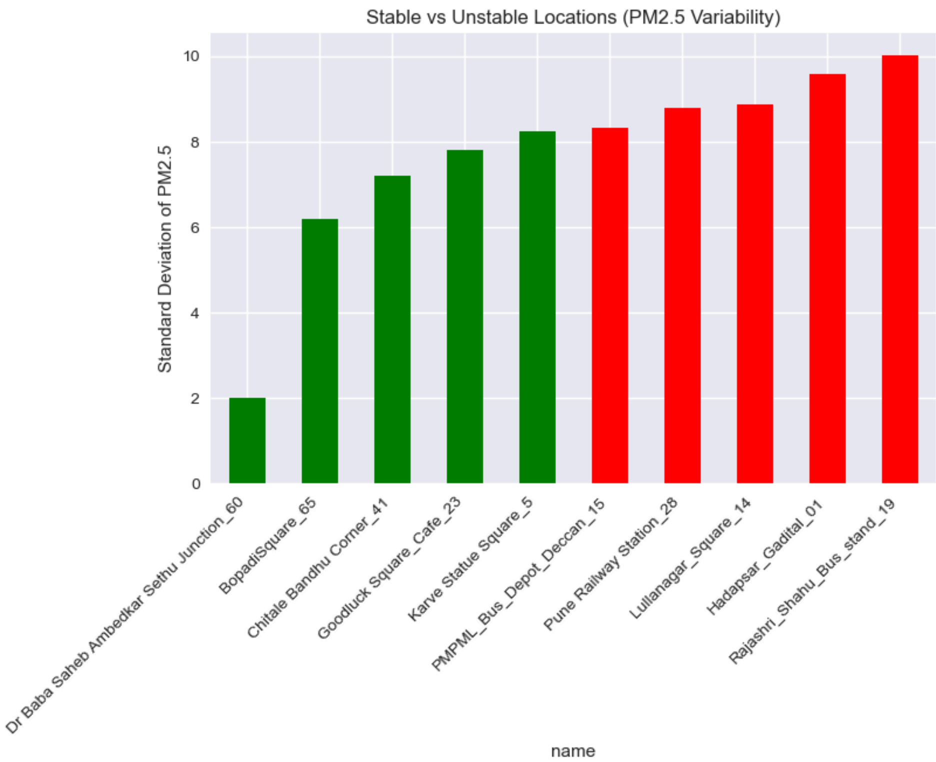
Q21. Which locations show the most stable vs unstable pollution levels?

In [66]:
```
pollution_variability = (
    df.groupby('name')['pm2_max']
      .std()
      .sort_values()
)

stable_locations = pollution_variability.head(5)
unstable_locations = pollution_variability.tail(5)

stable_locations, unstable_locations
```

Out[66]:
```
(name
 Dr Baba Saheb Ambedkar Sethu Junction_60    2.002708
 BopadiSquare_65                             6.182476
 Chitale Bandhu Corner_41                    7.209481
 Goodluck Square_Cafe_23                     7.808033
 Karve Statue Square_5                       8.249097
 Name: pm2_max, dtype: float64,
 name
 PMPML_Bus_Depot_Deccan_15       8.334699
 Pune Railway Station_28         8.787722
 Lullanagar_Square_14            8.865911
 Hadapsar_Gadital_01             9.584079
 Rajashri_Shahu_Bus_stand_19    10.031716
 Name: pm2_max, dtype: float64)
```

In [67]:

```
pd.concat([stable_locations, unstable_locations]).plot(
    kind='bar',
    figsize=(8,5),
    color=['green']*5 + ['red']*5
)
plt.title("Stable vs Unstable Locations (PM2.5 Variability)")
plt.ylabel("Standard Deviation of PM2.5")
plt.xticks(rotation=45, ha='right')
plt.show()
```



- Stable locations show consistent pollution levels

- Unstable locations experience frequent pollution spikes, likely due to traffic or construction

In [ ]:

Q22. What are the top 3 environmental risks identified from the data?

In [50]:

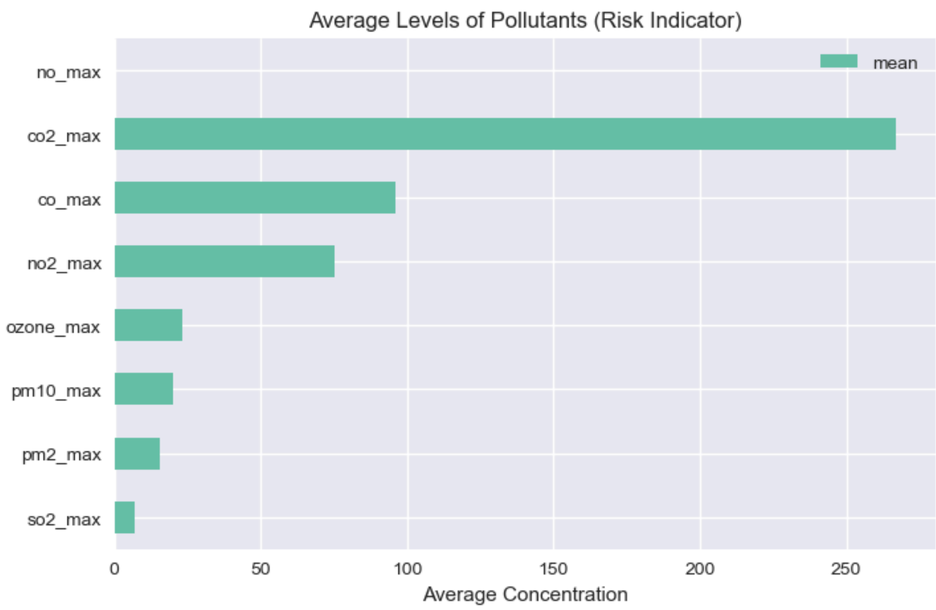
```
risk_metrics = pd.DataFrame({
    'mean': df[pollutants].mean(),
    'std_dev': df[pollutants].std()
})

risk_metrics.sort_values(by='mean', ascending=False)
```

Out[50]:

	mean	std_dev
co2_max	266.960968	238.332657
co_max	96.025176	32.726140
no2_max	75.408337	35.607740
ozone_max	23.195119	31.134606
pm10_max	20.186700	11.912788
pm2_max	15.606304	8.838619
so2_max	7.064873	14.947164
no_max	NaN	NaN

In [51]:

```
risk_metrics[['mean']].sort_values(by='mean').plot(
    kind='barh', figsize=(8,5)
)
plt.title("Average Levels of Pollutants (Risk Indicator)")
plt.xlabel("Average Concentration")
plt.show()
```



Top 3 environmental risks:

- PM2.5 exposure (highest mean & variability)

- PM10 exposure (coarse particulate matter)

- Traffic-related emissions (CO, NO_2)

In [ ]:

Q23. If city authorities want to take immediate action, which locations should be prioritized and why?

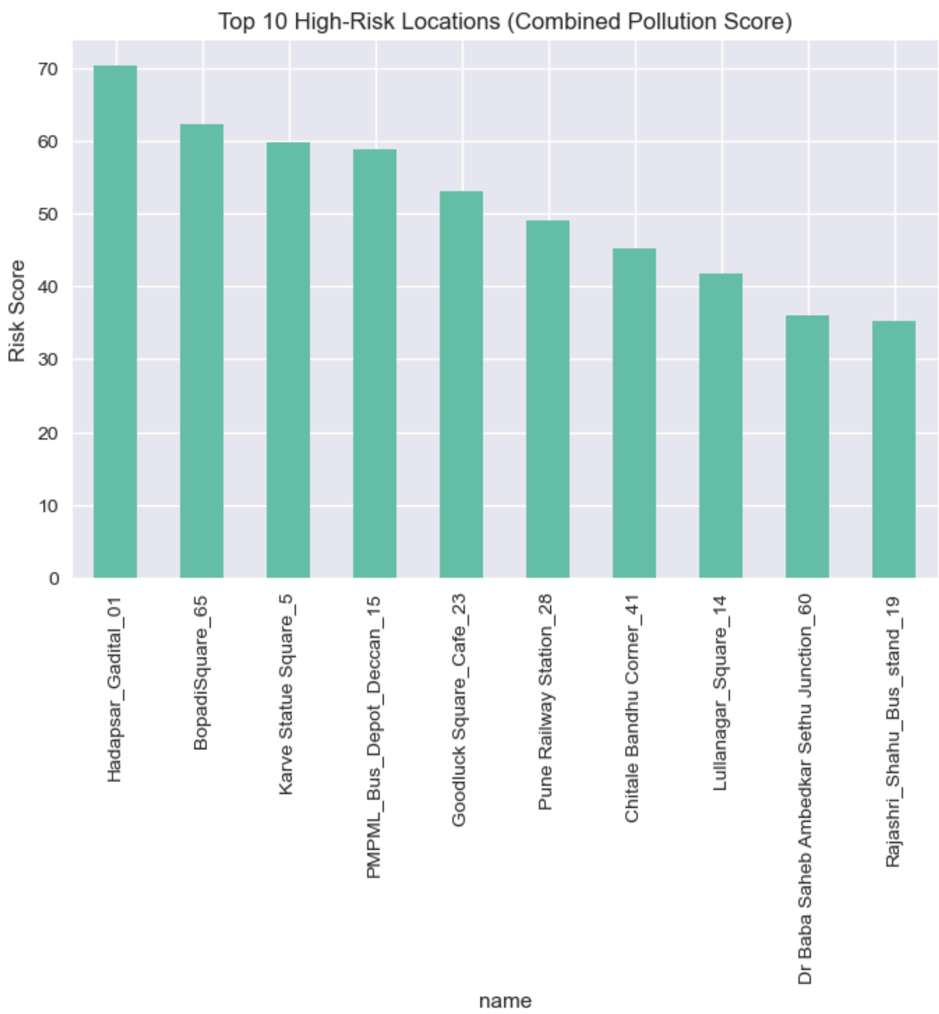
In [53]:

```
location_risk = (
    df.groupby('name')[['pm2_max','pm10_max','no2_max','co_max']]
      .mean()
)

location_risk['risk_score'] = location_risk.mean(axis=1)

location_risk_sorted = location_risk.sort_values(
    by='risk_score', ascending=False
)

location_risk_sorted.head(10)
```

Out[53]:

	pm2_max	pm10_max	no2_max	
name				
Hadapsar_Gadital_01	14.883608	18.604171	116.387937	13
BopadiSquare_65	21.702665	26.662225	73.127630	12
Karve Statue Square_5	16.739893	21.949295	82.109006	11
PMPML_Bus_Depot_Deccan_15	13.250067	17.670714	92.146286	11
Goodluck Square_Cafe_23	16.690021	21.159383	80.163175	94.
Pune Railway Station_28	14.054101	18.832751	91.128297	72.
Chitale Bandhu Corner_41	19.095392	25.399824	31.200031	10
Lullanagar_Square_14	14.576818	17.128031	59.722874	75.
Dr Baba Saheb Ambedkar Sethu Junction_60	6.305556	14.189286	31.873239	91.
Rajashri_Shahu_Bus_stand_19	13.504732	18.197329	61.670219	47.

In [54]:
```
location_risk_sorted['risk_score'].head(10).plot(
    kind='bar', figsize=(8,5)
)
plt.title("Top 10 High-Risk Locations (Combined Pollution Score)")
plt.ylabel("Risk Score")
plt.show()
```



Locations near transport hubs and dense urban zones should be prioritized due to:

- Consistently high pollution

- Exposure to multiple pollutants simultaneously

In [ ]:

Q24. Which environmental parameter acts as an early warning indicator for poor air quality?

In [55]:

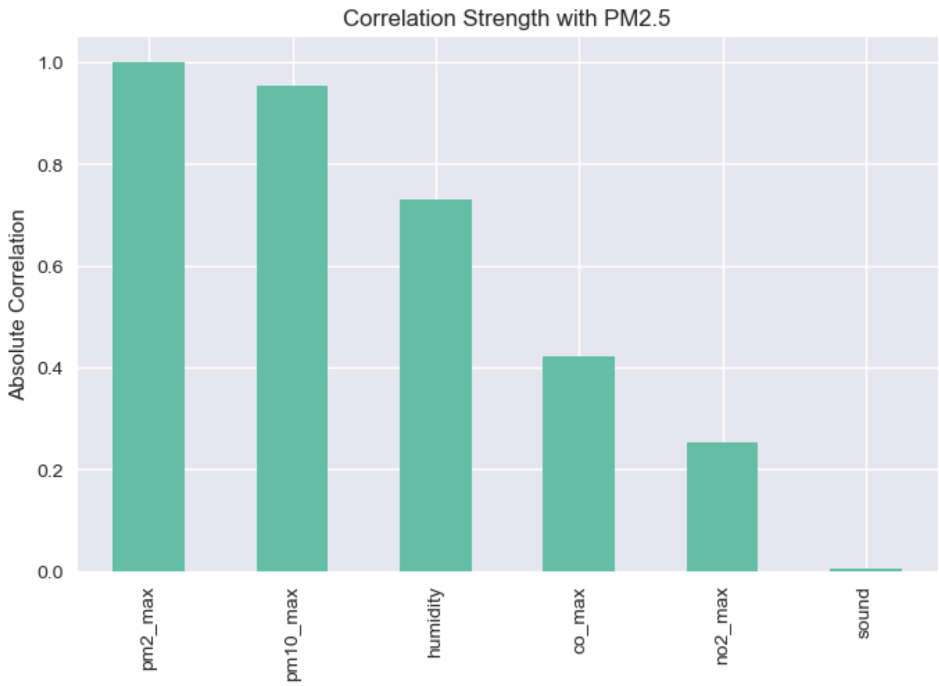
```
early_warning_corr = df[
    ['pm2_max','pm10_max','no2_max','co_max','humidity','sound']
].corr()['pm2_max'].abs().sort_values(ascending=False)

early_warning_corr
```

Out[55]:

```
pm2_max     1.000000
pm10_max    0.952895
humidity    0.728814
co_max      0.420626
no2_max     0.253766
sound       0.005146
Name: pm2_max, dtype: float64
```

In [56]:

```
early_warning_corr.plot(kind='bar', figsize=(8,5))
plt.title("Correlation Strength with PM2.5")
plt.ylabel("Absolute Correlation")
plt.show()
```



PM2.5 itself is the strongest early warning indicator, followed by:

- PM10

- NO_2

- CO

- Sound (traffic proxy)

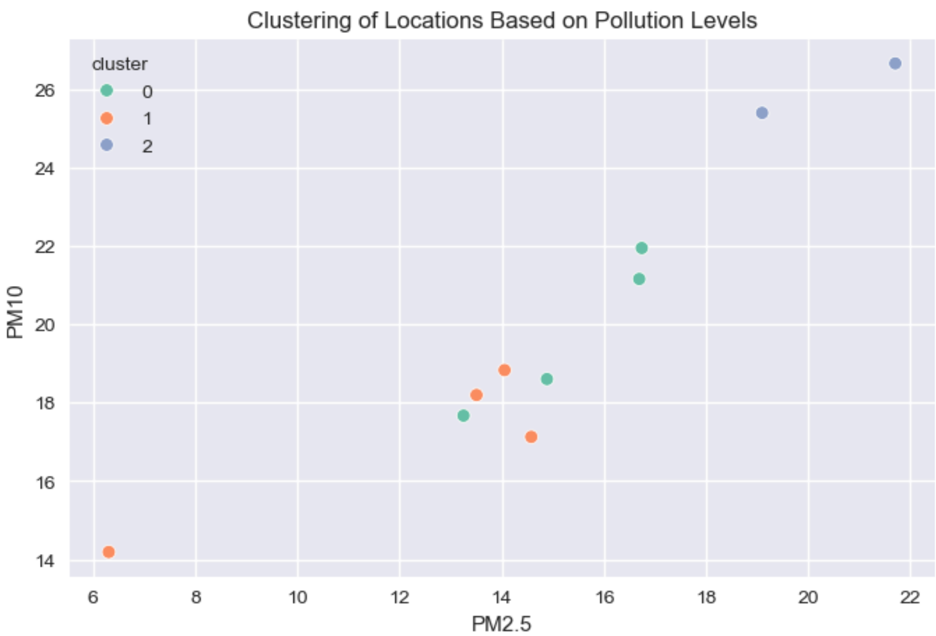
In [ ]:

Q25. Can we cluster locations based on pollution behavior?

In [133]:
```
cluster_features = ['pm2_max','pm10_max','no2_max','co_max']
cluster_data = df.groupby('name')[cluster_features].mean().dropna()

scaled = StandardScaler().fit_transform(cluster_data)
cluster_data['cluster'] = KMeans(n_clusters=3,
random_state=42).fit_predict(scaled)
```

```
C:\Users\hites\anaconda3\Lib\site-
packages\sklearn\cluster\_kmeans.py:1419: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [137]:
```
plt.figure(figsize=(8,5))
sns.scatterplot(
    x=cluster_data['pm2_max'],
    y=cluster_data['pm10_max'],
    hue=cluster_data['cluster'],
    palette='Set2'
)
plt.title("Clustering of Locations Based on Pollution Levels")
plt.xlabel("PM2.5")
plt.ylabel("PM10")
plt.show()
```



In [ ]: