```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from io import StringIO
import datetime
# Read the CSV file—make sure it's in your working directory
df = pd.read_csv("/content/delhiaqi.csv", parse_dates=['date'])
df.set_index('date', inplace=True)
df.head()
```

date	co	no	no2	o3	so2	pm2_5	pm10	nh3
2023-01-01 00:00:00	1655.58	1.66	39.41	5.90	17.88	169.29	194.64	5.83
2023-01-01 01:00:00	1869.20	6.82	42.16	1.99	22.17	182.84	211.08	7.66
2023-01-01 02:00:00	2510.07	27.72	43.87	0.02	30.04	220.25	260.68	11.40
2023-01-01 03:00:00	3150.94	55.43	44.55	0.85	35.76	252.90	304.12	13.55
2023-01-01 04:00:00	3471.37	68.84	45.24	5.45	39.10	266.36	322.80	14.19

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.describe()
```

	co	no	no2	o3	so2	pm2_5	pm10	nh3
count	561.000000	561.000000	561.000000	561.000000	561.000000	561.000000	561.000000	561.000000
mean	3814.942210	51.181979	75.292496	30.141943	64.655936	358.256364	420.988414	26.425062
std	3227.744681	83.904476	42.473791	39.979405	61.073080	227.359117	271.287026	36.563094
min	654.220000	0.000000	13.370000	0.000000	5.250000	60.100000	69.080000	0.630000
25%	1708.980000	3.380000	44.550000	0.070000	28.130000	204.450000	240.900000	8.230000
50%	2590.180000	13.300000	63.750000	11.800000	47.210000	301.170000	340.900000	14.820000
75%	4432.680000	59.010000	97.330000	47.210000	77.250000	416.650000	482.570000	26.350000
max	16876.220000	425.580000	263.210000	164.510000	511.170000	1310.200000	1499.270000	267.510000

2. Key Pollutant Statistics Table

```
pollutants = ['pm2_5', 'pm10', 'no2', 'so2', 'co', 'o3', 'nh3']

# Generate descriptive statistics
stats_table = df[pollutants].agg(['mean', 'max', 'min', 'std']).T
stats_table.columns = ['Mean', 'Max', 'Min', 'Std Dev']
stats_table
```

	Mean	Max	Min	Std Dev
pm2_5	358.256364	1310.20	60.10	227.359117
pm10	420.988414	1499.27	69.08	271.287026
no2	75.292496	263.21	13.37	42.473791
so2	64.655936	511.17	5.25	61.073080
co	3814.942210	16876.22	654.22	3227.744681
o3	30.141943	164.51	0.00	39.979405
nh3	26.425062	267.51	0.63	36.563094

Next steps: Generate code with stats_table | View recommended plots | New interactive sheet

3. Time-Wise Statistics Table

```
import datetime

def get_time_of_day(time):
```

```
    if datetime.time(4, 0) <= time < datetime.time(12, 0):
        return 'Morning'
    elif datetime.time(12, 0) <= time < datetime.time(16, 0):
        return 'Noon'
    elif datetime.time(16, 0) <= time < datetime.time(21, 0):
        return 'Evening'
    else:
        return 'Night'
pollutants = ['pm2_5', 'pm10', 'no2', 'so2', 'co', 'o3', 'nh3']

df['time_of_day'] = df['time'].apply(get_time_of_day)

time_of_day_stats = df.groupby('time_of_day')[pollutants].mean().reindex(['Morning', 'Noon', 'Evening', 'Night'])
time_of_day_stats
```

time_of_day	pm2_5	pm10	no2	so2	co	o3	nh3
Morning	270.368360	312.832540	67.282540	74.372540	2629.174656	64.854974	16.518836
Noon	461.341087	564.297065	126.384239	96.531087	6204.066196	8.418587	55.462174
Evening	475.080783	572.037739	82.856348	68.175217	5315.266609	12.100609	39.479652
Night	320.027455	359.693818	50.708242	33.300364	2795.386788	15.066424	12.483152

4. Summary Table: Daily Averages by Pollutant

```
#daily Average
daily_stats = df.resample('D')[pollutants].mean()
daily_stats
```

date	pm2_5	pm10	no2	so2	co	o3	nh3
2023-01-01	443.940000	535.040417	93.236250	102.260417	5929.152500	21.290833	63.490833
2023-01-02	698.104167	830.148750	110.187083	110.189583	7610.322083	16.977083	49.090000
2023-01-03	381.810417	434.333750	71.801250	59.574583	3640.492500	39.477917	18.581667
2023-01-04	304.021667	350.490833	75.657500	52.073750	2769.867917	34.640833	13.959583
2023-01-05	423.604583	496.787917	81.712083	58.004583	4700.819583	17.712083	21.724583
2023-01-06	418.079583	494.499583	93.650833	68.695000	5184.808333	9.747500	25.464583
2023-01-07	265.905000	294.637917	62.633750	36.130417	2336.502500	39.790417	10.715000
2023-01-08	354.217917	396.266667	65.047083	37.968750	3091.970417	33.128333	11.099583
2023-01-09	557.806250	650.216667	101.675417	99.410417	6249.587083	2.570417	43.312500
2023-01-10	340.752917	376.388333	61.220000	44.445417	2702.554583	36.968333	10.025833
2023-01-11	450.968750	507.721250	101.548333	65.743750	4262.447500	27.332500	16.803750
2023-01-12	300.427083	326.666667	59.263333	45.438333	1969.337500	53.240000	8.845417
2023-01-13	590.368333	673.778333	85.196667	123.103333	7308.801250	14.156667	63.056250
2023-01-14	216.677917	239.563750	34.643750	31.808333	1538.197500	64.803333	10.581667
2023-01-15	127.660000	152.899167	48.217083	31.977500	1516.501250	52.130833	16.033750
2023-01-16	175.975000	221.794583	66.589583	44.594167	2359.311250	31.580417	21.626667
2023-01-17	259.332500	317.094167	70.786667	48.141250	3176.530833	34.425000	23.755833
2023-01-18	360.406667	439.072500	85.553333	58.322917	4384.278750	27.975417	29.918333
2023-01-19	526.485833	611.337917	132.177500	174.065833	6477.117917	27.857500	63.451667
2023-01-20	265.761667	318.207083	64.618750	55.958333	2770.979583	27.460833	19.518750
2023-01-21	245.029583	320.232083	66.103750	57.868750	2263.070000	25.228333	21.341667
2023-01-22	162.943333	216.407083	33.536667	22.238333	1404.961250	27.261667	15.687500
2023-01-23	409.547083	522.370417	77.383750	66.191667	4799.564583	26.542083	36.840417
2023-01-24	251.776667	305.728889	46.724444	45.672222	1938.926667	32.718889	7.360000

5. Yearly AQI-like Table (use PM2.5 as AQI proxy unless AQI column exists)

```
df['month'] = df.index.year
# Example: Using PM2.5 as AQI proxy
annual_pm25 = df.groupby('month')['pm2_5'].mean()
annual_pm25
```

month	pm2_5
2023	358.256364

dtype: float64

6. Correlation Matrix Table

```
corr_table = df[pollutants].corr()
corr_table
```

	pm2_5	pm10	no2	so2	co	o3	nh3
pm2_5	1.000000	0.994088	0.698696	0.648996	0.953083	-0.450458	0.720303
pm10	0.994088	1.000000	0.720050	0.658325	0.966801	-0.468477	0.754468
no2	0.698696	0.720050	1.000000	0.734961	0.776402	-0.407177	0.700254
so2	0.648996	0.658325	0.734961	1.000000	0.716831	-0.049158	0.843635
co	0.953083	0.966801	0.776402	0.716831	1.000000	-0.463082	0.826299
o3	-0.450458	-0.468477	-0.407177	-0.049158	-0.463082	1.000000	-0.299663
nh3	0.720303	0.754468	0.700254	0.843635	0.826299	-0.299663	1.000000

7. Table: Highest Pollutant Hours (Top 5 for each pollutant)

```
for p in pollutants:
    print(f"\nTop 5 {p} hours:")
    print(df[[p]].sort_values(p, ascending=False).head(5))
```

```
Top 5 pm2_5 hours:
                     pm2_5
date
2023-01-19 17:00:00  1310.20
2023-01-13 20:00:00  1278.35
2023-01-13 19:00:00  1232.62
2023-01-19 16:00:00  1228.04
2023-01-13 18:00:00  1225.39

Top 5 pm10 hours:
                      pm10
date
2023-01-19 17:00:00  1499.27
2023-01-13 18:00:00  1448.70
2023-01-13 17:00:00  1448.28
2023-01-19 16:00:00  1415.28
2023-01-13 19:00:00  1355.20

Top 5 no2 hours:
                       no2
date
2023-01-19 16:00:00  263.21
2023-01-19 17:00:00  252.25
2023-01-19 15:00:00  246.76
2023-01-19 14:00:00  227.57
2023-01-19 18:00:00  213.86

Top 5 so2 hours:
                       so2
date
2023-01-19 17:00:00  511.17
2023-01-19 16:00:00  495.91
2023-01-19 15:00:00  461.58
2023-01-19 14:00:00  381.47
2023-01-19 18:00:00  381.47

Top 5 co hours:
                         co
date
2023-01-13 17:00:00  16876.22
2023-01-19 17:00:00  16662.60
2023-01-13 18:00:00  16662.60
```

```
2023-01-19 16:00:00  16448.97
2023-01-19 15:00:00  16448.97

Top 5 o3 hours:
                         o3
date
2023-01-17 09:00:00  164.51
2023-01-18 09:00:00  157.36
2023-01-17 08:00:00  155.93
2023-01-18 08:00:00  148.77
2023-01-14 09:00:00  144.48

Top 5 nh3 hours:
                        nh3
date
```

8. Health Threshold Exceedance Table (Counts)

```
thresholds = {'pm2_5': 60, 'pm10': 100, 'no2': 40, 'so2': 20, 'co': 2000, 'o3': 100, 'nh3': 20}
exceedance = {pollutant: (df[pollutant] > value).sum() for pollutant, value in thresholds.items()}
exceedance_table = pd.Series(exceedance, name='Exceedance Count')
exceedance_table
```

	Exceedance Count
pm2_5	561
pm10	555
no2	454
so2	519
co	373
o3	55
nh3	200

dtype: int64

```
df['month(Jan)'] = df.index.month

# Group by year and calculate the mean PM2.5 (you can replace 'pm2_5' with your AQI column if present)
monthly_avg_aqi = df.groupby('month(Jan)')['pm2_5'].mean().round(2)
monthly_avg_aqi
```

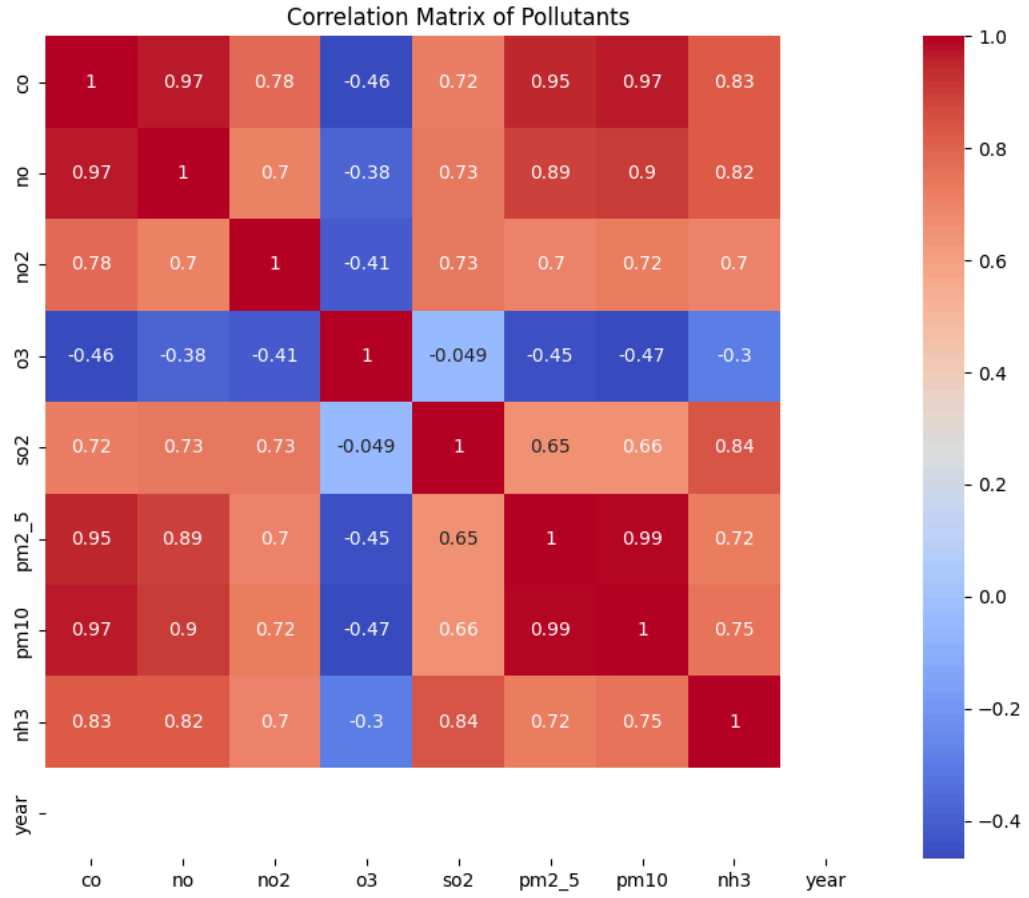
	pm2_5
month(Jan)	
1	358.26

dtype: float64

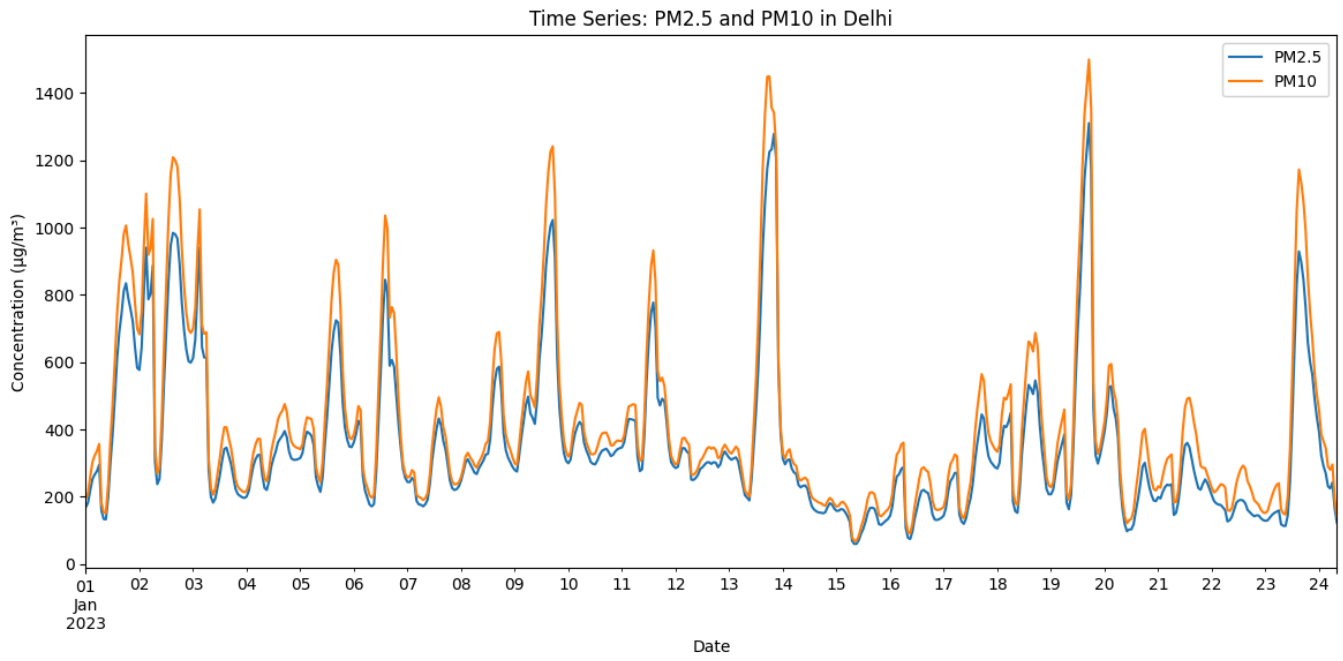
Major Visualizations

```
# Compute correlation of only numeric columns
numeric_df = df.select_dtypes(include=[np.number])
corr = numeric_df.corr()

plt.figure(figsize=(10,8))
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix of Pollutants")
plt.show()
```




```
plt.figure(figsize=(14,6))
df['pm2_5'].plot(label='PM2.5')
df['pm10'].plot(label='PM10')
plt.xlabel('Date')
plt.ylabel('Concentration (µg/m³)')
plt.title('Time Series: PM2.5 and PM10 in Delhi')
plt.legend()
plt.show()
```




```
plt.figure(figsize=(10,6))
sns.boxplot(x='season', y='pm2_5', data=df, order=['Winter','Spring','Summer','Autumn'])
plt.title('Seasonal Variation of PM2.5 in Delhi')
```

```
plt.xlabel('Season')
plt.ylabel('PM2.5 Concentration (µg/m³)')
plt.show()
```