In [6]:
```
import pandas as pd
```

In [2]:
```
import pandas as pd
```

In [3]:
```
import seaborn as sns
```

In [5]:
```
import matplotlib.pyplot as plt
```

In [8]:
```
# Load the Excel file
df = pd.read_excel('delhiaqi.xlsx')
```

In [19]:
```
print("Dataset info:")
print(df.info())
```

```
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 561 entries, 0 to 560
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    561 non-null    datetime64[ns]
 1   co      561 non-null    float64
 2   no      561 non-null    float64
 3   no2     561 non-null    float64
 4   o3      561 non-null    float64
 5   so2     561 non-null    float64
 6   pm2.5   561 non-null    float64
 7   pm10    561 non-null    float64
 8   nh3     561 non-null    float64
dtypes: datetime64[ns](1), float64(8)
memory usage: 39.6 KB
None
```

In [20]:
```
#show first few rows data
print("Show first few rows data:")
print(df.head())
```

```
Show first few rows data:
                 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
```

In [13]:

```
# show missing values
print("Missing Values:")
print(df.isnull().sum())
```

```
Missing Values:
date     0
co       0
no       0
no2      0
o3       0
so2      0
pm2.5    0
pm10     0
nh3      0
dtype: int64
```

In [16]:

```
#show column names
print("Column Names:")
print(df.columns)
```

```
Column Names:
Index(['date', 'co', 'no', 'no2', 'o3', 'so2', 'pm2.5', 'pm10', 'nh3'], dtype='objec
t')
```

In [33]:

```
# convert date to datetime format
#df['date']=pd.to_datetime(df['date'])

# Set date as an Index
#df.set_index('date',inplace=True)
```

In [35]:

```
df.reset_index(inplace=True)  # brings 'date' back as a column
```

In [37]:

```
# convert date to datetime format
df['date']=pd.to_datetime(df['date'])

# Set date as an Index
df.set_index('date',inplace=True)
```

In [41]:

```
# Resample to daily average
daily_avg=df.resample('D').mean()
```

In [42]:

```
# Preview first few rows
print("Daily Average of Pollutants:")
print(daily_avg.head())
```

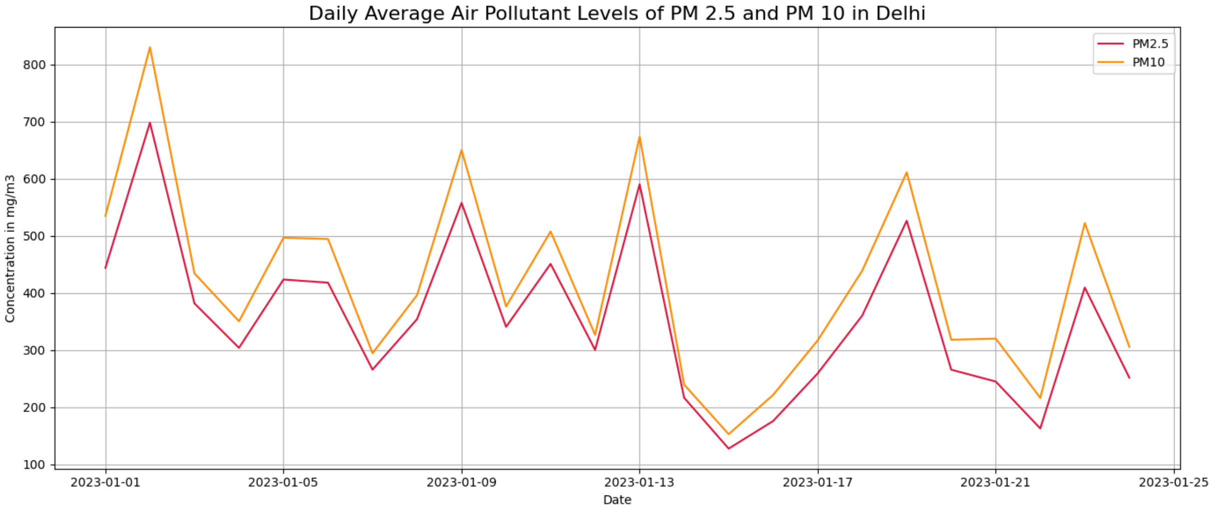
```
Daily Average of Pollutants:
                     co          no         no2         o3         so2  \
date
2023-01-01  5929.152500  112.348750   93.236250  21.290833  102.260417
2023-01-02  7610.322083  140.537500  110.187083  16.977083  110.189583
2023-01-03  3640.492500   39.717500   71.801250  39.477917   59.574583
2023-01-04  2769.867917    8.811667   75.657500  34.640833   52.073750
2023-01-05  4700.819583   62.289583   81.712083  17.712083   58.004583

                 pm2.5        pm10        nh3
date
2023-01-01  443.940000  535.040417  63.490833
2023-01-02  698.104167  830.148750  49.090000
2023-01-03  381.810417  434.333750  18.581667
2023-01-04  304.021667  350.490833  13.959583
2023-01-05  423.604583  496.787917  21.724583
```

In [68]:

```
# Plotting the graph of the pollutants Particulate matter of 2.5 and 10

# using matplotlib
plt.figure(figsize=(14,6))
plt.plot(daily_avg.index, daily_avg['pm2.5'], label='PM2.5', color='crimson')
plt.plot(daily_avg.index, daily_avg['pm10'], label='PM10', color='darkorange')
#plt.plot(daily_avg.index, daily_avg['no2'], label='NO2', color='royalblue')
#plt.plot(daily_avg.index, daily_avg['co'], label='CO', color='green')
plt.title('Daily Average Air Pollutant Levels of PM2.5 and PM10 in Delhi', fontsize
plt.xlabel('Date')
plt.ylabel('Concentration in mg/m3')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

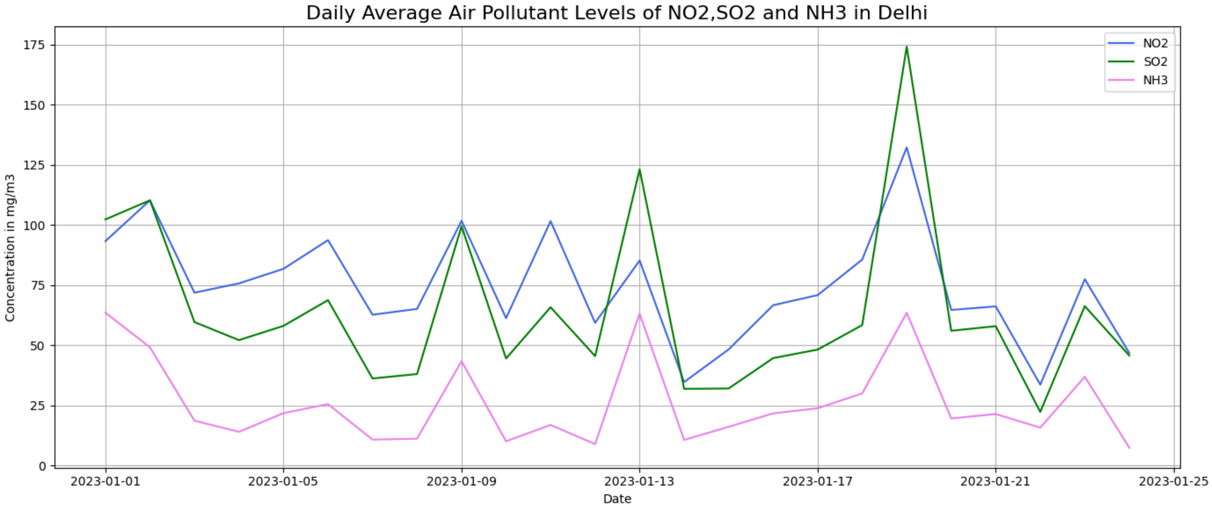


In [75]:

```
#Plotting the graph of the gaseous air pollutants

# using matplotlib
plt.figure(figsize=(14,6))
plt.plot(daily_avg.index, daily_avg['no2'], label='NO2', color='royalblue')
plt.plot(daily_avg.index, daily_avg['so2'], label='SO2', color='green')
#plt.plot(daily_avg.index, daily_avg['co'], label='CO', color='darkred')
```

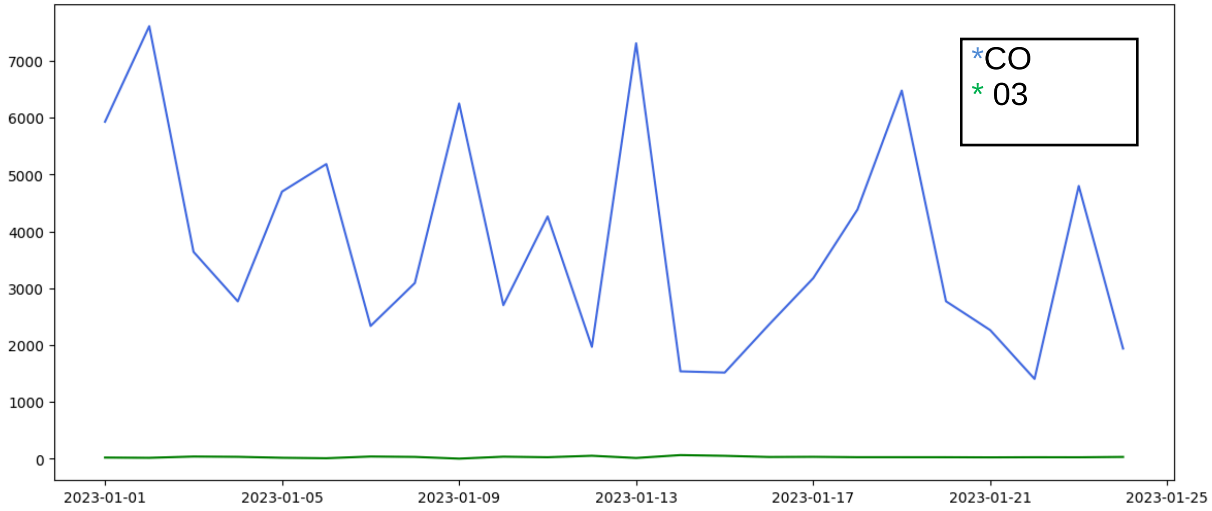
```
plt.plot(daily_avg.index, daily_avg['nh3'], label='NH3', color='violet')
plt.title('Daily Average Air Pollutant Levels of NO2,SO2 and NH3 in Delhi', fontsiz
plt.xlabel('Date')
plt.ylabel('Concentration in mg/m3')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [82]:

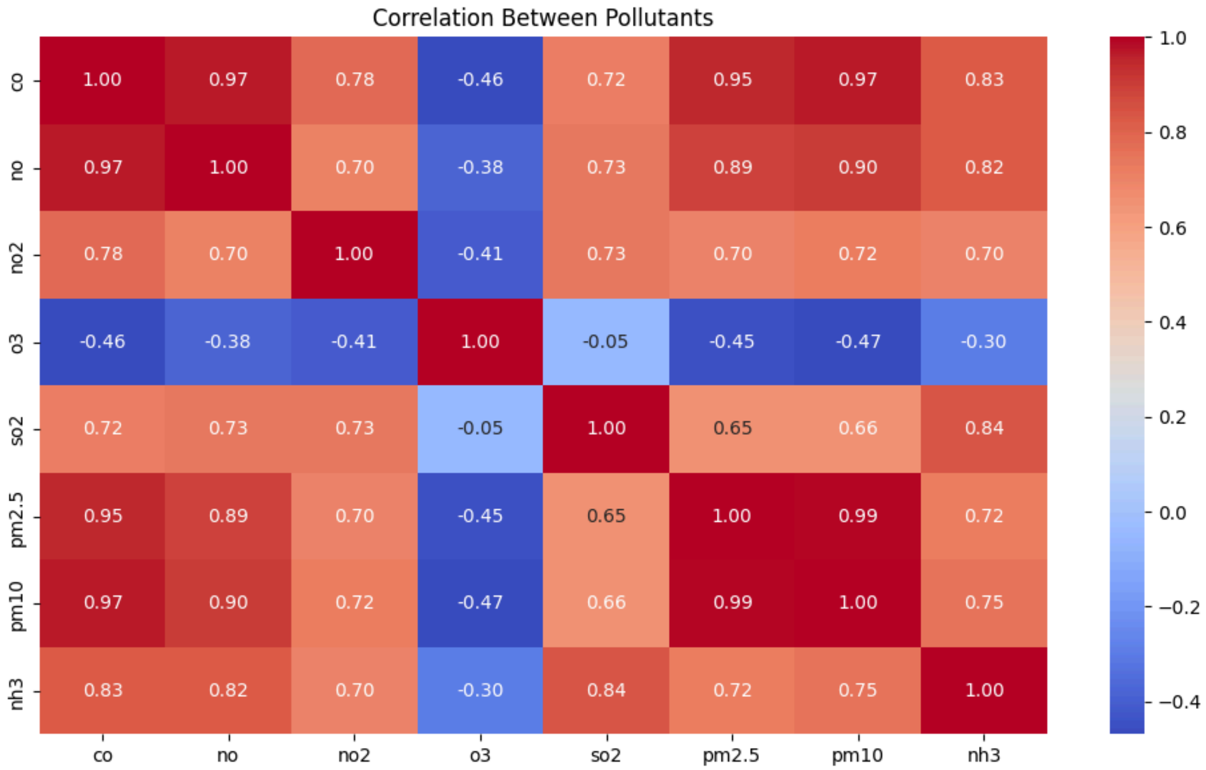
```
#Plotting the graph of the remaining gaseous air pollutants

# using matplotlib
plt.figure(figsize=(14,6))
plt.plot(daily_avg.index, daily_avg['co'], label='CO', color='royalblue')
plt.plot(daily_avg.index, daily_avg['o3'], label='O3', color='green')
# Plot each pollutant
for pollutant, color in zip(['pm2.5', 'pm10', 'no2', 'co'], ['crimson', 'darkorange
    plt.plot(daily_avg.index, daily_avg[pollutant], label=pollutant.upper(), color=
# Add value labels every 7 days
for i in range(0, len(daily_avg), 7):
    x = daily_avg.index[i]
    y = daily_avg[pollutant].iloc[i]
    plt.text(x, y, f'{y:.1f}', fontsize=8, rotation=45, color=color)
plt.title('Daily Average Air Pollutant Levels of CO and O3 in Delhi', fontsize=16)
plt.xlabel('Date')
plt.ylabel('Concentration in mg/m3')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [74]:

```
# CORRELATION OF THE POLLUTANTS
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Between Pollutants')
plt.tight_layout()
plt.show()
```


Correlation Between Pollutants
co 1.00 0.97 0.78 -0.46 0.72 0.95 0.97 0.83
no 0.97 1.00 0.70 -0.38 0.73 0.89 0.90 0.82
no2 0.78 0.70 1.00 -0.41 0.73 0.70 0.72 0.70
o3 -0.46 -0.38 -0.41 1.00 -0.05 -0.45 -0.47 -0.30
so2 0.72 0.73 0.73 -0.05 1.00 0.65 0.66 0.84
pm2.5 0.95 0.89 0.70 -0.45 0.65 1.00 0.99 0.72
pm10 0.97 0.90 0.72 -0.47 0.66 0.99 1.00 0.75
nh3 0.83 0.82 0.70 -0.30 0.84 0.72 0.75 1.00
co
no
no2
o3
so2
pm2.5
pm10
nh3
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

```
In [ ]:
```