42539-dmv-5

October 20, 2024

Problem Statement: Analyzing Air Quality Index (AQI) Trends in a City

Tasks to Perform: 1. Import the "City_Air_Quality.csv" dataset. 2. Explore the dataset to understand its structure and content. 3. Identify the relevant variables for visualizing AQI trends, such as date, pollutant levels, and AQI values. 4. Create line plots or time series plots to visualize the overall AQI trend over time. 5. Plot individual pollutant levels (e.g., PM2.5, PM10, CO) on separate line plots to visualize their trends over time. 6. Use bar plots or stacked bar plots to compare the AQI values across different dates or time periods. 7. Create box plots or violin plots to analyze the distribution of AQI values for different pollutant categories. 8. Use scatter plots or bubble charts to explore the relationship between AQI values and pollutant levels. 9. Customize the visualizations by adding labels, titles, legends, and appropriate color schemes.

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

1. Import the "City_Air_Quality.csv" dataset.

[5]:
```
data = pd.read_csv("AirQuality.csv")
```

2. Explore the dataset to understand its structure and content.

[6]:
```
print(data.head())
print(data.info())
```

```
        City        Date  PM2.5  PM10     NO    NO2    NOx  NH3     CO    SO2  \
0  Ahmedabad  2015-01-01    NaN   NaN   0.92  18.22  17.15  NaN   0.92  27.64   \
1  Ahmedabad  2015-01-02    NaN   NaN   0.97  15.69  16.46  NaN   0.97  24.55
2  Ahmedabad  2015-01-03    NaN   NaN  17.40  19.30  29.70  NaN  17.40  29.07
3  Ahmedabad  2015-01-04    NaN   NaN   1.70  18.48  17.97  NaN   1.70  18.59
4  Ahmedabad  2015-01-05    NaN   NaN  22.10  21.42  37.76  NaN  22.10  39.33

       O3  Benzene  Toluene  Xylene  AQI AQI_Bucket
0  133.36     0.00     0.02    0.00  NaN        NaN
1   34.06     3.68     5.50    3.77  NaN        NaN
2   30.70     6.80    16.40    2.25  NaN        NaN
3   36.08     4.43    10.14    1.00  NaN        NaN
4   39.31     7.01    18.89    2.78  NaN        NaN
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
```

```
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   City        29531 non-null  object
 1   Date        29531 non-null  object
 2   PM2.5       24933 non-null  float64
 3   PM10        18391 non-null  float64
 4   NO          25949 non-null  float64
 5   NO2         25946 non-null  float64
 6   NOx         25346 non-null  float64
 7   NH3         19203 non-null  float64
 8   CO          27472 non-null  float64
 9   SO2         25677 non-null  float64
 10  O3          25509 non-null  float64
 11  Benzene     23908 non-null  float64
 12  Toluene     21490 non-null  float64
 13  Xylene      11422 non-null  float64
 14  AQI         24850 non-null  float64
 15  AQI_Bucket  24850 non-null  object
dtypes: float64(13), object(3)
memory usage: 3.6+ MB
None
```

[7]:
```
print(data.isnull().sum())
```

```
City              0
Date              0
PM2.5          4598
PM10          11140
NO             3582
NO2            3585
NOx            4185
NH3           10328
CO             2059
SO2            3854
O3             4022
Benzene        5623
Toluene        8041
Xylene        18109
AQI            4681
AQI_Bucket     4681
dtype: int64
```

[8]:
```
data.dropna(inplace=True)
cleaned_data = data.dropna()
```

[9]:
```
print(cleaned_data)
```

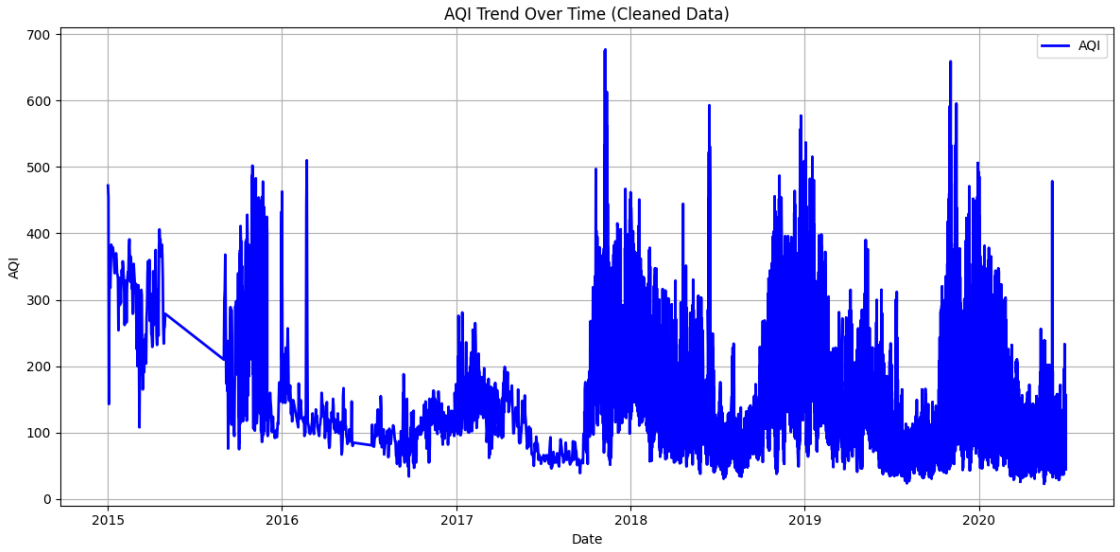
```
                City        Date  PM2.5    PM10    NO    NO2    NOx    NH3
2123       Amaravati  2017-11-25  81.40  124.50  1.44  20.50  12.08  10.72  \
2124       Amaravati  2017-11-26  78.32  129.06  1.26  26.00  14.85  10.28
2125       Amaravati  2017-11-27  88.76  135.32  6.60  30.85  21.77  12.91
2126       Amaravati  2017-11-28  64.18  104.09  2.56  28.07  17.01  11.42
2127       Amaravati  2017-11-29  72.47  114.84  5.23  23.20  16.59  12.25
…                …         …    …     …    …     …    …
29525  Visakhapatnam  2020-06-26   7.63   32.27  5.91  23.27  17.19  11.15
29526  Visakhapatnam  2020-06-27  15.02   50.94  7.68  25.06  19.54  12.47
29527  Visakhapatnam  2020-06-28  24.38   74.09  3.42  26.06  16.53  11.99
29528  Visakhapatnam  2020-06-29  22.91   65.73  3.45  29.53  18.33  10.71
29529  Visakhapatnam  2020-06-30  16.64   49.97  4.05  29.26  18.80  10.03

         CO    SO2      O3  Benzene  Toluene  Xylene    AQI    AQI_Bucket
2123   0.12  15.24  127.09     0.20     6.50    0.06  184.0      Moderate
2124   0.14  26.96  117.44     0.22     7.95    0.08  197.0      Moderate
2125   0.11  33.59  111.81     0.29     7.63    0.12  198.0      Moderate
2126   0.09  19.00  138.18     0.17     5.02    0.07  188.0      Moderate
2127   0.16  10.55  109.74     0.21     4.71    0.08  173.0      Moderate
…     …    …     …       …        …       …    …           …
29525  0.46   6.87   19.90     1.45     5.37    1.45   47.0          Good
29526  0.47   8.55   23.30     2.24    12.07    0.73   41.0          Good
29527  0.52  12.72   30.14     0.74     2.21    0.38   70.0  Satisfactory
29528  0.48   8.42   30.96     0.01     0.01    0.00   68.0  Satisfactory
29529  0.52   9.84   28.30     0.00     0.00    0.00   54.0  Satisfactory

[6236 rows x 16 columns]
```

[10]:
```
print(cleaned_data.isnull().sum())
```

```
City          0
Date          0
PM2.5         0
PM10          0
NO            0
NO2           0
NOx           0
NH3           0
CO            0
SO2           0
O3            0
Benzene       0
Toluene       0
Xylene        0
AQI           0
AQI_Bucket    0
dtype: int64
```

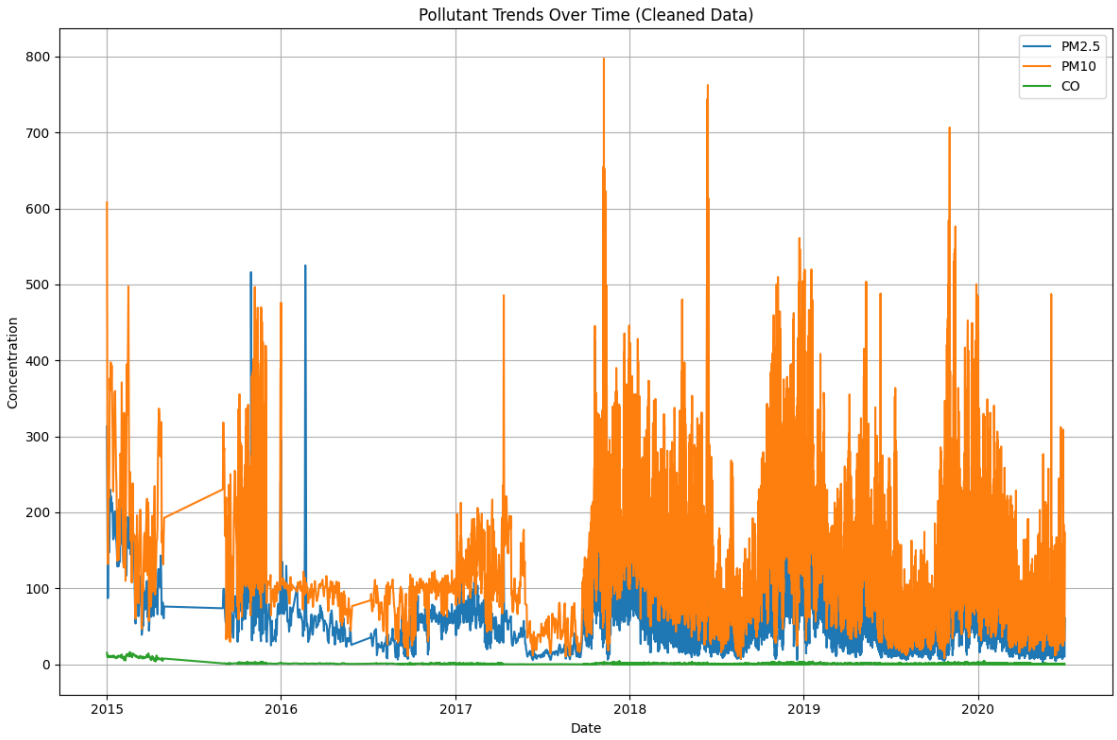
4. Create line plots or time series plots to visualize the overall AQI trend over time.

[11]:
```
import matplotlib.pyplot as plt

# Convert the 'Date' column to a datetime format for proper time series plotting
cleaned_data['Date'] = pd.to_datetime(cleaned_data['Date'])

# Sort the DataFrame by the 'Date' column (optional but recommended for time␣
 ↪series)
cleaned_data.sort_values(by='Date', inplace=True)

# Create the line plot
plt.figure(figsize=(12, 6))
plt.plot(cleaned_data['Date'], cleaned_data['AQI'], label='AQI', color='blue',␣
 ↪linewidth=2)
plt.xlabel('Date')
plt.ylabel('AQI')
plt.title('AQI Trend Over Time (Cleaned Data)')
plt.legend()
plt.grid(True)
plt.tight_layout()

# Display the plot
plt.show()
```



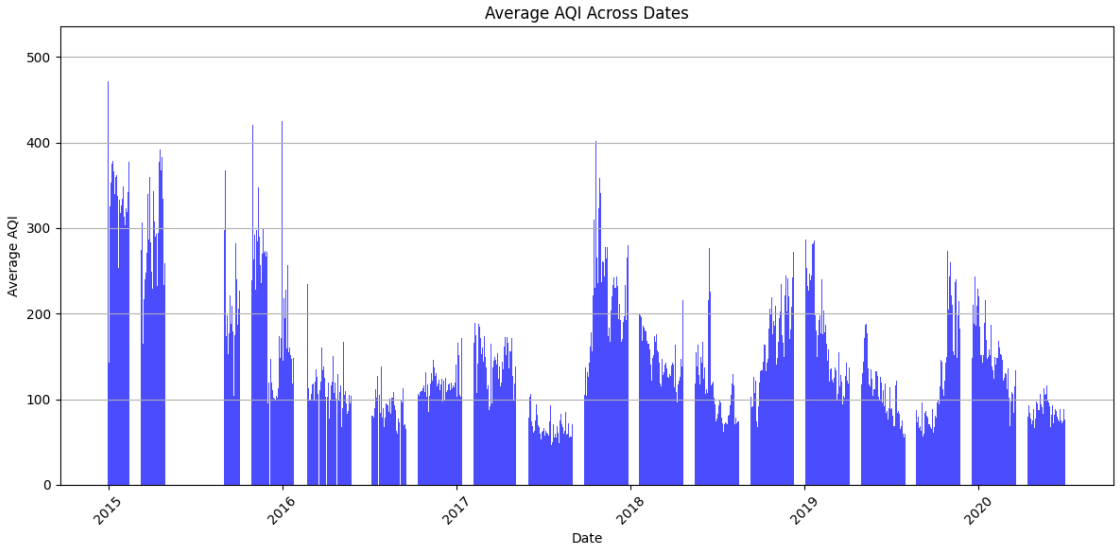
5. Plot individual pollutant levels (e.g., PM2.5, PM10, CO) on separate line plots to visualize their trends over time.

[13]:
```
cleaned_data['Date'] = pd.to_datetime(cleaned_data['Date'])

# Sort the DataFrame by the 'Date' column (optional but recommended for time␣
 ↪series)
cleaned_data.sort_values(by='Date', inplace=True)

# Define the pollutants you want to plot
pollutants = ['PM2.5', 'PM10', 'CO']

# Create separate line plots for each pollutant
plt.figure(figsize=(12, 8))

for pollutant in pollutants:
    plt.plot(cleaned_data['Date'], cleaned_data[pollutant], label=pollutant)

plt.xlabel('Date')
plt.ylabel('Concentration')
plt.title('Pollutant Trends Over Time (Cleaned Data)')
plt.legend()
plt.grid(True)
plt.tight_layout()

# Display the plots
plt.show()
```



6. Use bar plots or stacked bar plots to compare the AQI values across different dates or time periods.

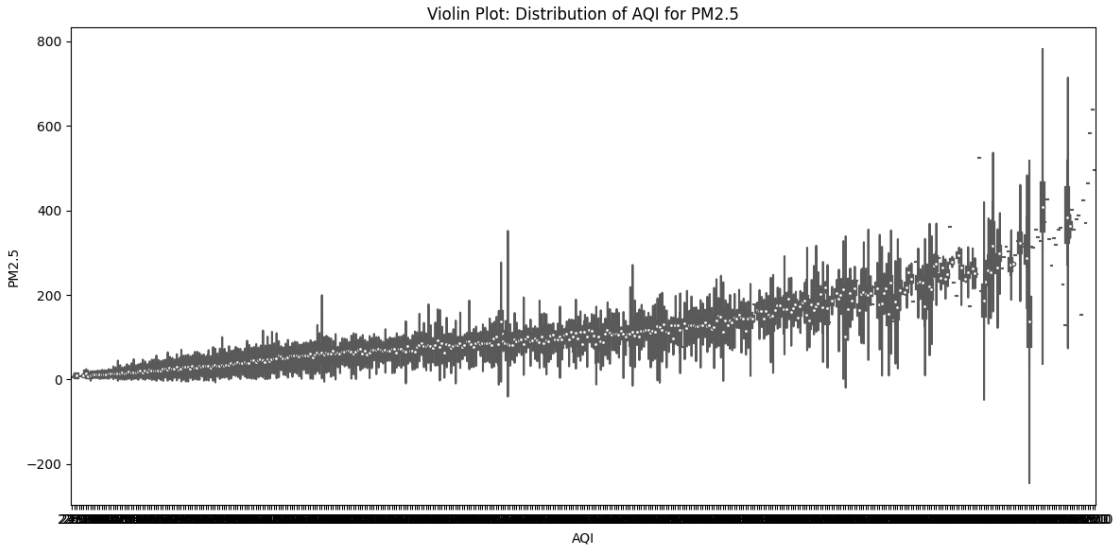
[14]:
```
cleaned_data['Date'] = pd.to_datetime(cleaned_data['Date'])

# Group the data by 'Date' and calculate the mean AQI for each date
aqi_by_date = cleaned_data.groupby('Date')['AQI'].mean().reset_index()

# Create a bar plot to compare AQI values across dates
plt.figure(figsize=(12, 6))
plt.bar(aqi_by_date['Date'], aqi_by_date['AQI'], color='blue', alpha=0.7)
plt.xlabel('Date')
plt.ylabel('Average AQI')
plt.title('Average AQI Across Dates')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

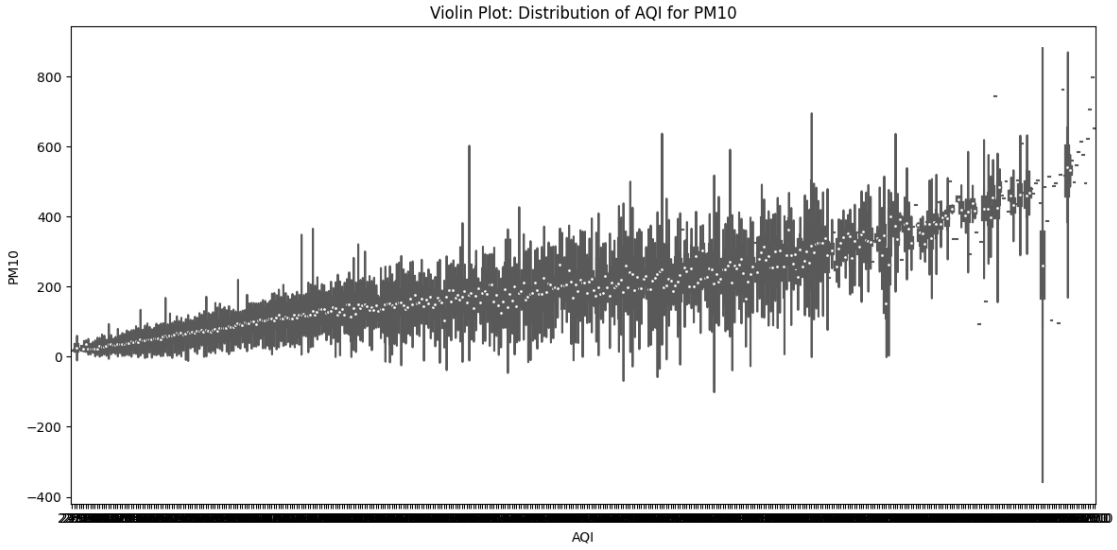


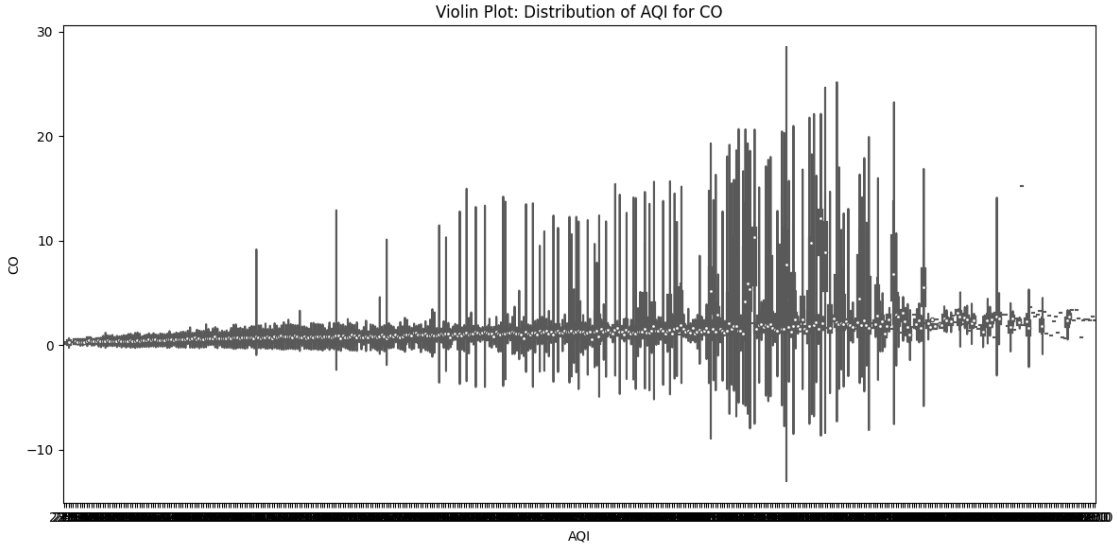
7.violin plots to analyze the distribution of AQI values for different pollutant categories.

[17]:
```
import seaborn as sns
import matplotlib.pyplot as plt

pollutants = ['PM2.5', 'PM10', 'CO']


# Create violin plots for the same analysis (if desired)
```

```
plt.figure(figsize=(12, 6))
sns.violinplot(data=cleaned_data, x='AQI', y='PM2.5', palette='Set2',␣
 ↪color='red')
plt.xlabel('AQI')
plt.ylabel('PM2.5')
plt.title('Violin Plot: Distribution of AQI for PM2.5')
plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 6))
sns.violinplot(data=cleaned_data, x='AQI', y='PM10', palette='Set2',␣
 ↪color='red')
plt.xlabel('AQI')
plt.ylabel('PM10')
plt.title('Violin Plot: Distribution of AQI for PM10')
plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 6))
sns.violinplot(data=cleaned_data, x='AQI', y='CO', palette='Set2', color='red')
plt.xlabel('AQI')
plt.ylabel('CO')
plt.title('Violin Plot: Distribution of AQI for CO')
plt.tight_layout()
plt.show()
```



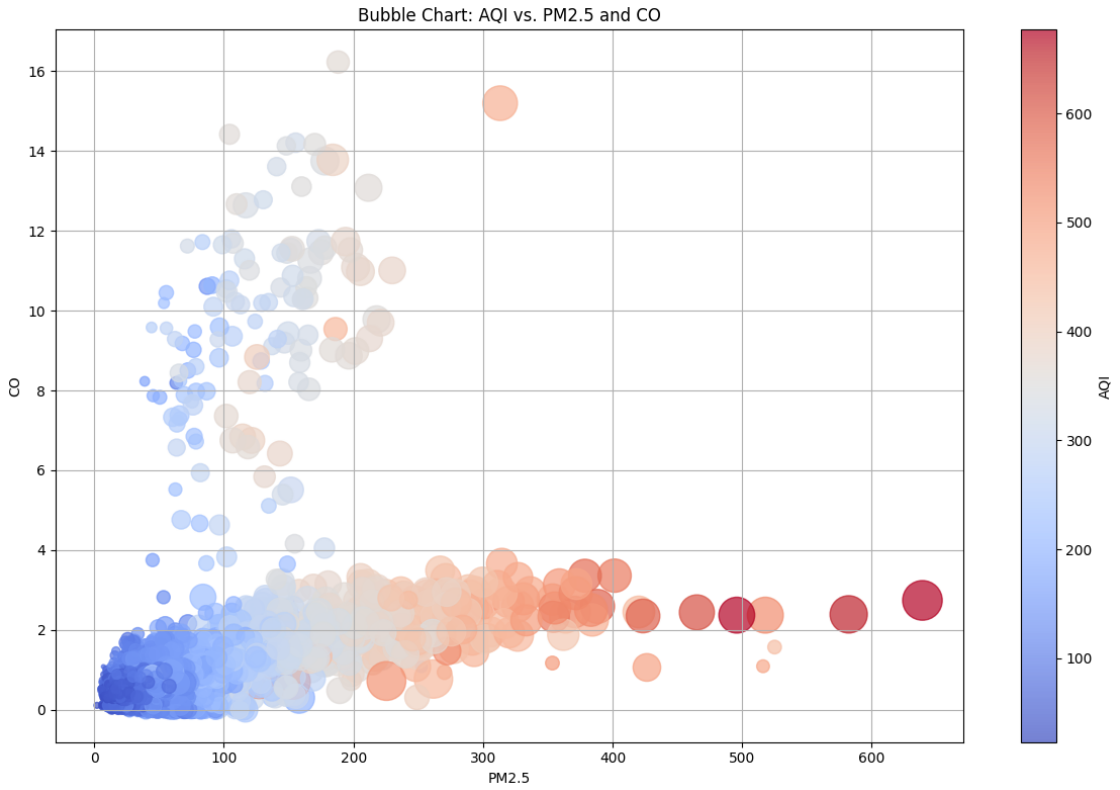




8.bubble charts to explore the relationship between AQI values and pollutant levels.

[18]:
```
import matplotlib.pyplot as plt


# Define variables for the bubble chart
x = cleaned_data['PM2.5']
y = cleaned_data['CO']
bubble_size = cleaned_data['PM10']  # Represent 'PM10' using bubble size
aqi_values = cleaned_data['AQI']
```

```
# Create the bubble chart
plt.figure(figsize=(12, 8))
plt.scatter(x, y, s=bubble_size, c=aqi_values, cmap='coolwarm', alpha=0.7)
plt.xlabel('PM2.5')
plt.ylabel('CO')
plt.title('Bubble Chart: AQI vs. PM2.5 and CO')
plt.colorbar(label='AQI')  # Add colorbar to show AQI values
plt.grid(True)
plt.tight_layout()
plt.show()
```



[ ]: