ShadowFox Data Science Internship

Name: Sivaranjini M

Level: Intermediate

Air Quality Index Analysis

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
```

Load the dataset

In [2]:
```
file_path = '/kaggle/input/delhi-aqi-123/delhiaqi.csv'
data = pd.read_csv(file_path)
```

In [3]:
```
data.head()
```

Out[3]:

	date	co	no	no2	o3	so2	pm2_5	pm10	nh3
0	2023-01-01 00:00:00	1655.58	1.66	39.41	5.90	17.88	169.29	194.64	5.83
1	2023-01-01 01:00:00	1869.20	6.82	42.16	1.99	22.17	182.84	211.08	7.66
2	2023-01-01 02:00:00	2510.07	27.72	43.87	0.02	30.04	220.25	260.68	11.40
3	2023-01-01 03:00:00	3150.94	55.43	44.55	0.85	35.76	252.90	304.12	13.55
4	2023-01-01 04:00:00	3471.37	68.84	45.24	5.45	39.10	266.36	322.80	14.19

Data Cleaning and Pre-processing

In [4]:
```
data.columns
```

Out[4]:
```
Index(['date', 'co', 'no', 'no2', 'o3', 'so2', 'pm2_5', 'pm10', 'nh3'], dt
ype='object')
```

In [5]:
```
data['date'] = pd.to_datetime(data['date'], errors='coerce')
```

In [6]:
```
data.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
```

In [7]:
```
data.set_index('date', inplace=True)
```

In [8]:
```
data.head()
```

Out[8]:

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

In [9]:
```
data = data.fillna(data.mean())
data
```

Out[9]:

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
...	...	...	...	...	...	...	...	...
2023-01-24 04:00:00	1762.39	4.64	37.01	33.26	30.52	231.15	289.84	6.27
2023-01-24 05:00:00	1735.69	6.82	34.96	46.49	34.33	225.08	280.52	9.12
2023-01-24 06:00:00	1922.61	8.16	40.10	56.51	43.39	242.49	296.07	12.54
2023-01-24 07:00:00	1361.85	9.05	52.78	71.53	100.14	165.67	191.82	7.47
2023-01-24 08:00:00	1134.87	8.61	56.89	80.11	110.63	123.76	140.26	5.51

561 rows × 8 columns

Calculate AQI

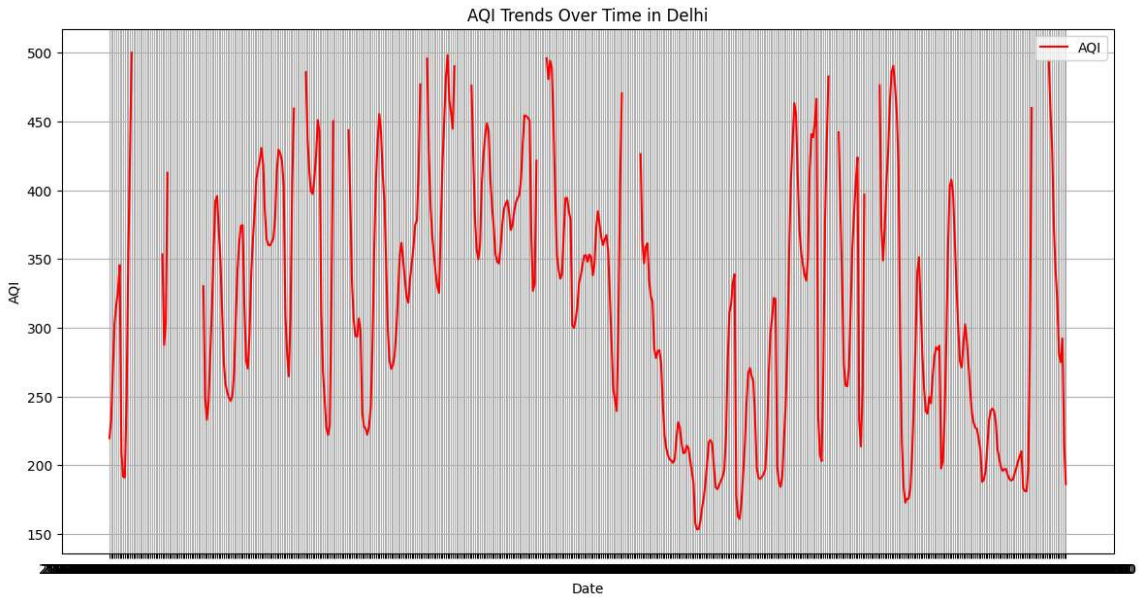
In [10]:

```python
# Breakpoints for AQI calculation (example values)
breakpoints = {
    'pm2_5': [
        (0.0, 12.0, 0, 50),
        (12.1, 35.4, 51, 100),
        (35.5, 55.4, 101, 150),
        (55.5, 150.4, 151, 200),
        (150.5, 250.4, 201, 300),
        (250.5, 350.4, 301, 400),
        (350.5, 500.4, 401, 500)
    ],
    'pm10': [
        (0, 54, 0, 50),
        (55, 154, 51, 100),
        (155, 254, 101, 150),
        (255, 354, 151, 200),
        (355, 424, 201, 300),
        (425, 504, 301, 400),
        (505, 604, 401, 500)
    ],
    # Add other pollutants here
}

def calculate_aqi(concentration, breakpoints):
    for (C_low, C_high, I_low, I_high) in breakpoints:
        if C_low <= concentration <= C_high:
            AQI = ((I_high - I_low) / (C_high - C_low)) * (concentration -
            return AQI
    return None

def overall_aqi(row):
    aqi_values = []
    for pollutant in breakpoints:
        if pollutant in row and pd.notnull(row[pollutant]):
            aqi = calculate_aqi(row[pollutant], breakpoints[pollutant])
            if aqi is not None:
                aqi_values.append(aqi)
    if aqi_values:
        return max(aqi_values)
    return None

# Load the dataset
file_path = '/kaggle/input/delhi-aqi-123/delhiaqi.csv'
data = pd.read_csv(file_path)

# Ensure relevant pollutant columns exist
pollutants = ['pm2_5', 'pm10']  # Add other pollutants as necessary
for pollutant in pollutants:
    if pollutant not in data.columns:
        raise ValueError(f"Missing column for pollutant: {pollutant}")

# Calculate AQI for each row
data['AQI'] = data.apply(overall_aqi, axis=1)

# Display the updated dataset with AQI
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         AQI
0   5.83  219.620721
1   7.66  233.048649
2  11.40  270.121622
3  13.55  303.378378
4  14.19  316.717117
```

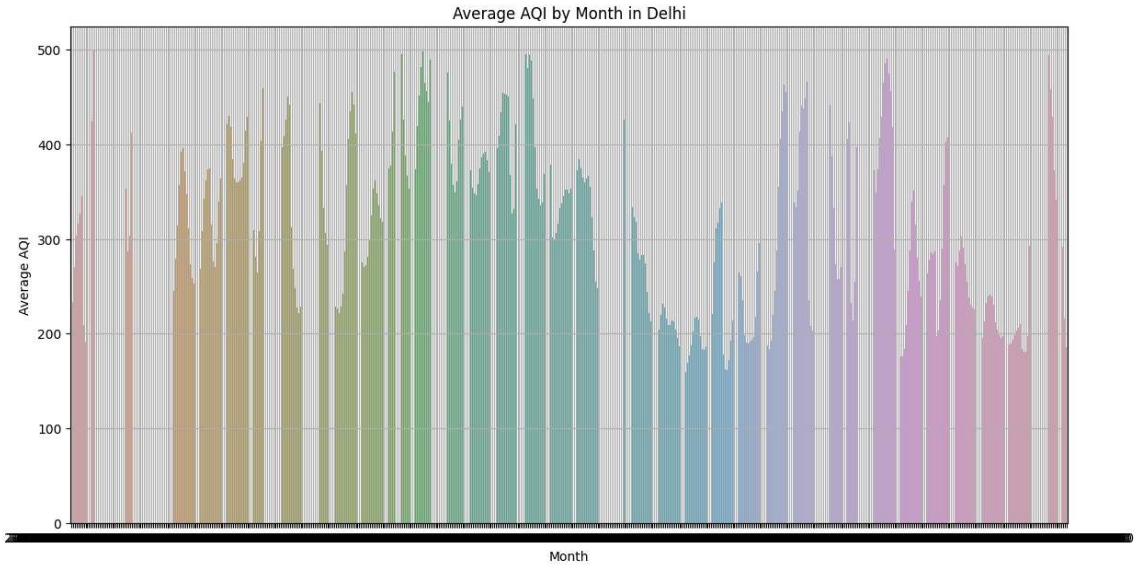
Plotting AQI trends over time

In [11]:
```
plt.figure(figsize=(14, 7))
plt.plot(data['date'], data['AQI'], label='AQI',color='red')
plt.xlabel('Date')
plt.ylabel('AQI')
plt.title('AQI Trends Over Time in Delhi')
plt.legend()
plt.grid(True)
plt.show()
```



Seasonal Variations

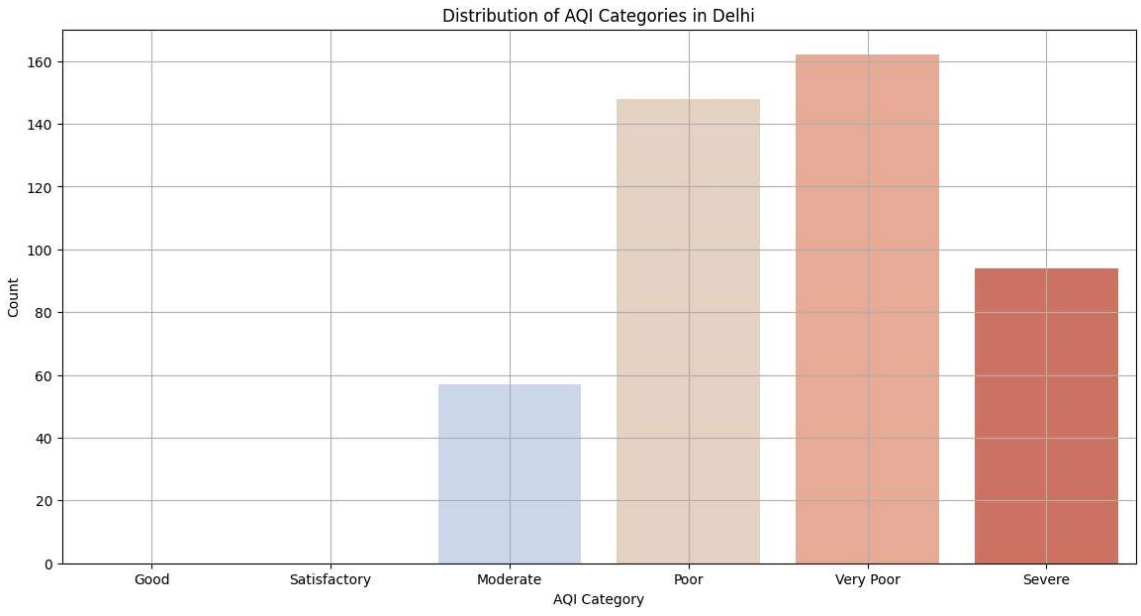
In [12]:
```
data['month'] = data['date']
monthly_aqi = data.groupby('month')['AQI'].mean()

plt.figure(figsize=(14, 7))
sns.barplot(x=monthly_aqi.index, y=monthly_aqi.values)
plt.xlabel('Month')
plt.ylabel('Average AQI')
plt.title('Average AQI by Month in Delhi')
plt.grid(True)
plt.show()
```



Assess AQI levels against benchmarks

In [13]:
```
aqi_benchmarks = {
    'Good': 50,
    'Satisfactory': 100,
    'Moderate': 200,
    'Poor': 300,
    'Very Poor': 400,
    'Severe': 500
}

# Categorizing AQI levels
data['AQI_Category'] = pd.cut(data['AQI'], bins=[0, 50, 100, 200, 300, 400,

plt.figure(figsize=(14, 7))
sns.countplot(x='AQI_Category', data=data, palette='coolwarm')
plt.xlabel('AQI Category')
plt.ylabel('Count')
plt.title('Distribution of AQI Categories in Delhi')
plt.grid(True)
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:641: Future
Warning: The default of observed=False is deprecated and will be changed t
o True in a future version of pandas. Pass observed=False to retain curren
t behavior or observed=True to adopt the future default and silence this w
arning.
  grouped_vals = vals.groupby(grouper)
```



Statistical Analysis

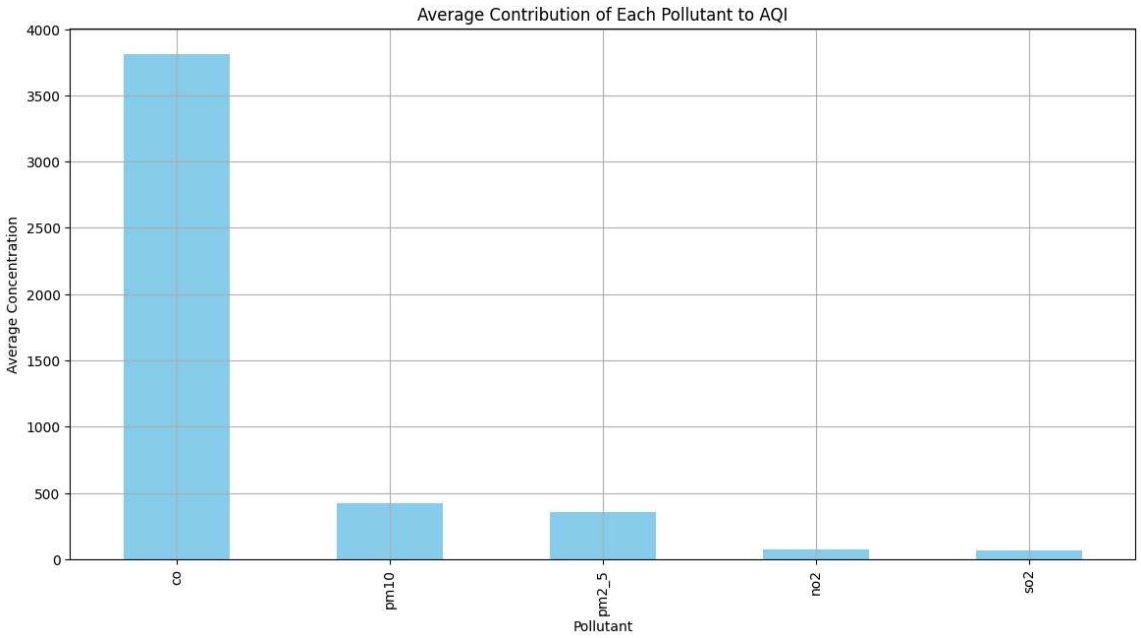
In [14]:
```
data.describe()
```

Out[14]:

	co	no	no2	o3	so2	pm2_5	pm
count	561.000000	561.000000	561.000000	561.000000	561.000000	561.000000	561.0000
mean	3814.942210	51.181979	75.292496	30.141943	64.655936	358.256364	420.9884
std	3227.744681	83.904476	42.473791	39.979405	61.073080	227.359117	271.2870
min	654.220000	0.000000	13.370000	0.000000	5.250000	60.100000	69.0800
25%	1708.980000	3.380000	44.550000	0.070000	28.130000	204.450000	240.9000
50%	2590.180000	13.300000	63.750000	11.800000	47.210000	301.170000	340.9000
75%	4432.680000	59.010000	97.330000	47.210000	77.250000	416.650000	482.5700
max	16876.220000	425.580000	263.210000	164.510000	511.170000	1310.200000	1499.2700

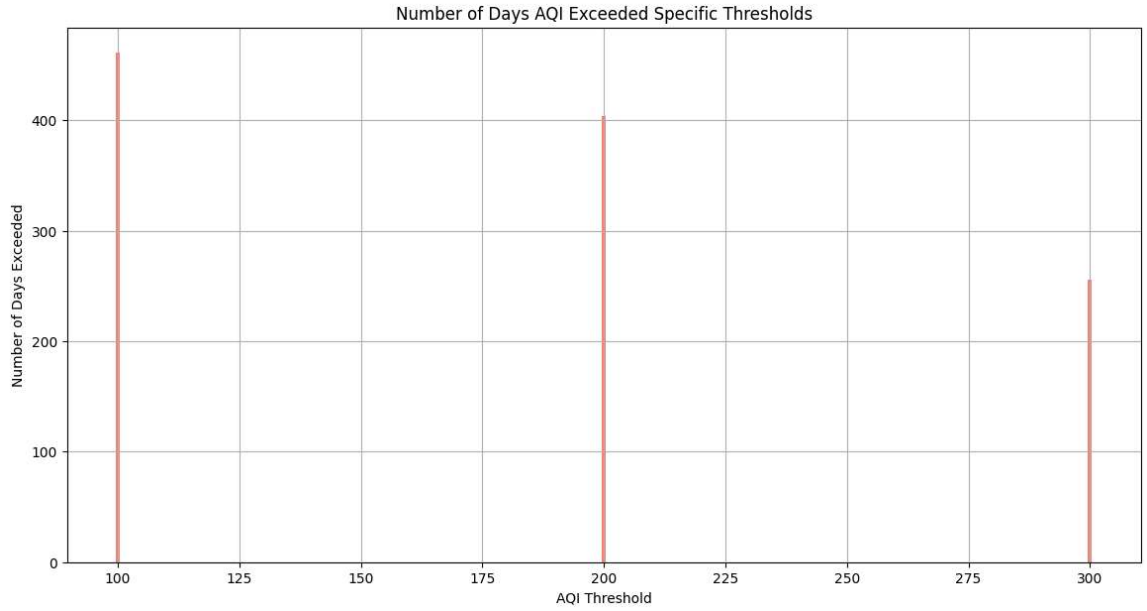
Top Pollutant Identification

In [16]:
```
pollutants = ['co','no2','so2', 'pm2_5', 'pm10']
contribution = data[pollutants].mean().sort_values(ascending=False)

plt.figure(figsize=(14, 7))
contribution.plot(kind='bar', color='skyblue')
plt.xlabel('Pollutant')
plt.ylabel('Average Concentration')
plt.title('Average Contribution of Each Pollutant to AQI')
plt.grid(True)
plt.show()
```



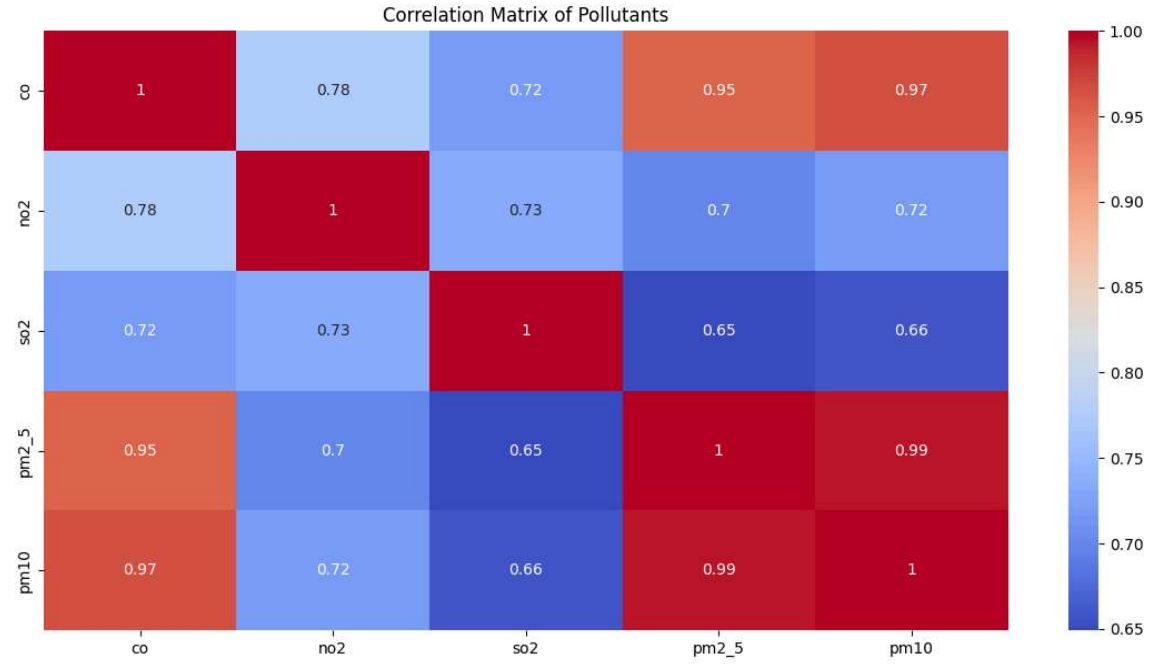
AQI Exceedance Days

In [21]:
```
# Define AQI thresholds
thresholds = [100, 200, 300]

# Count the number of days AQI exceeded each threshold
exceedance_counts = {threshold: (data['AQI'] > threshold).sum() for thresho

plt.figure(figsize=(14, 7))
plt.bar(exceedance_counts.keys(), exceedance_counts.values(), color='salmon
plt.xlabel('AQI Threshold')
plt.ylabel('Number of Days Exceeded')
plt.title('Number of Days AQI Exceeded Specific Thresholds')
plt.grid(True)
plt.show()
```



Correlation Between Pollutants

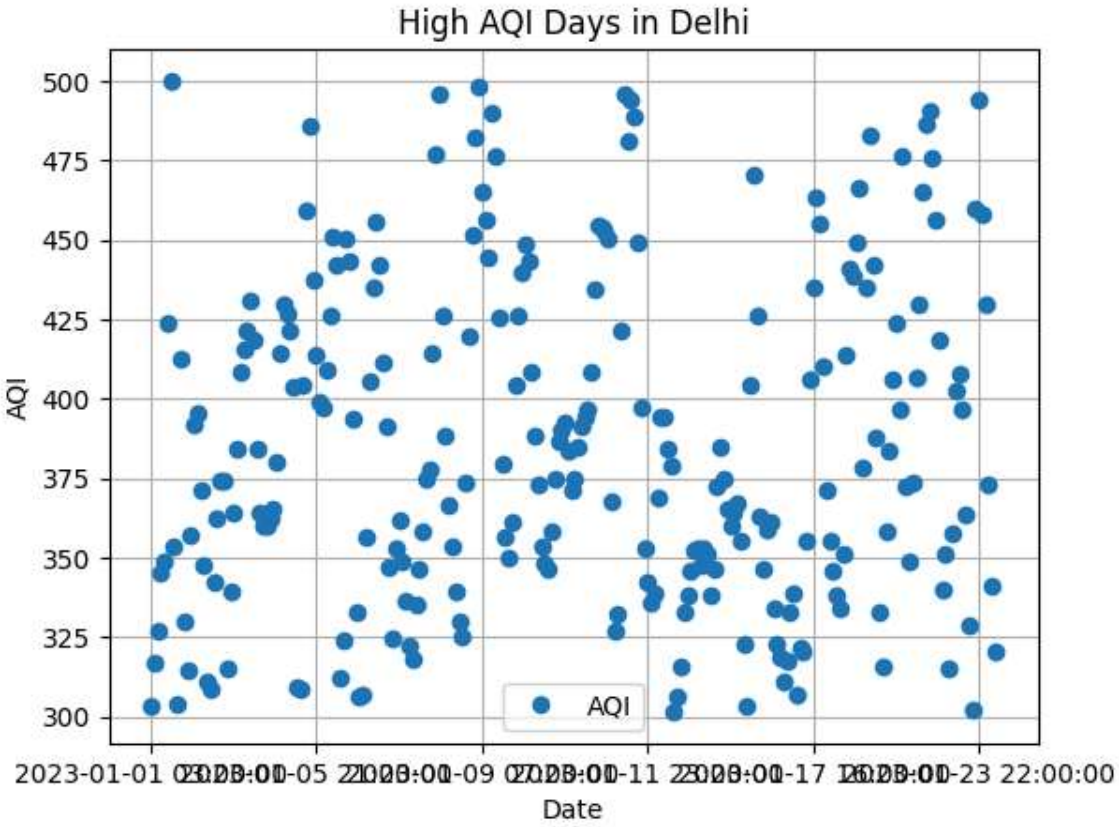
In [22]:
```
# Plot the correlation matrix for pollutants
plt.figure(figsize=(14, 7))
sns.heatmap(data[pollutants].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix of Pollutants')
plt.show()
```



Analysis of High Pollution Days

In [28]:
```
# Define a threshold for high AQI
high_aqi_threshold = 300

# Filter days with AQI above the threshold
high_aqi_days = data[data['AQI'] > high_aqi_threshold]

# Analyze high AQI days
high_aqi_summary = high_aqi_days.describe()

print("Summary of High AQI Days:")
print(high_aqi_summary)

# Plotting high AQI days
plt.figure(figsize=(14, 7))
high_aqi_days.plot(x='date', y='AQI', kind='line', marker='o', linestyle='N
plt.xlabel('Date')
plt.ylabel('AQI')
plt.title('High AQI Days in Delhi')
plt.grid(True)
plt.show()
```

```
Summary of High AQI Days:
                co          no         no2          o3         so2  \
count   256.000000  256.000000  256.000000  256.000000  256.000000
mean   3216.654023   28.396094   75.622500   23.450977   53.666172
std    1216.893590   33.753064   31.176229   31.676521   31.732734
min    1201.630000    0.000000   28.100000    0.000000    7.750000
25%    2309.800000    3.727500   53.470000    0.360000   29.560000
50%    2990.720000   16.430000   68.545000    9.570000   44.585000
75%    3911.972500   44.370000   96.302500   39.160000   67.710000
max    8010.860000  214.580000  180.960000  143.050000  209.810000

            pm2_5        pm10         nh3         AQI
count  256.000000  256.000000  256.000000  256.000000
mean   343.371680  392.984727   16.790781  385.365100
std     64.281782   80.000234   13.547532   51.385698
min    251.190000  266.750000    0.630000  301.683784
25%    296.595000  333.375000    7.350000  346.679730
50%    325.130000  367.640000   12.730000  374.957658
75%    385.970000  450.897500   20.835000  424.425817
max    528.930000  603.980000   80.050000  499.980000

<Figure size 1400x700 with 0 Axes>
```

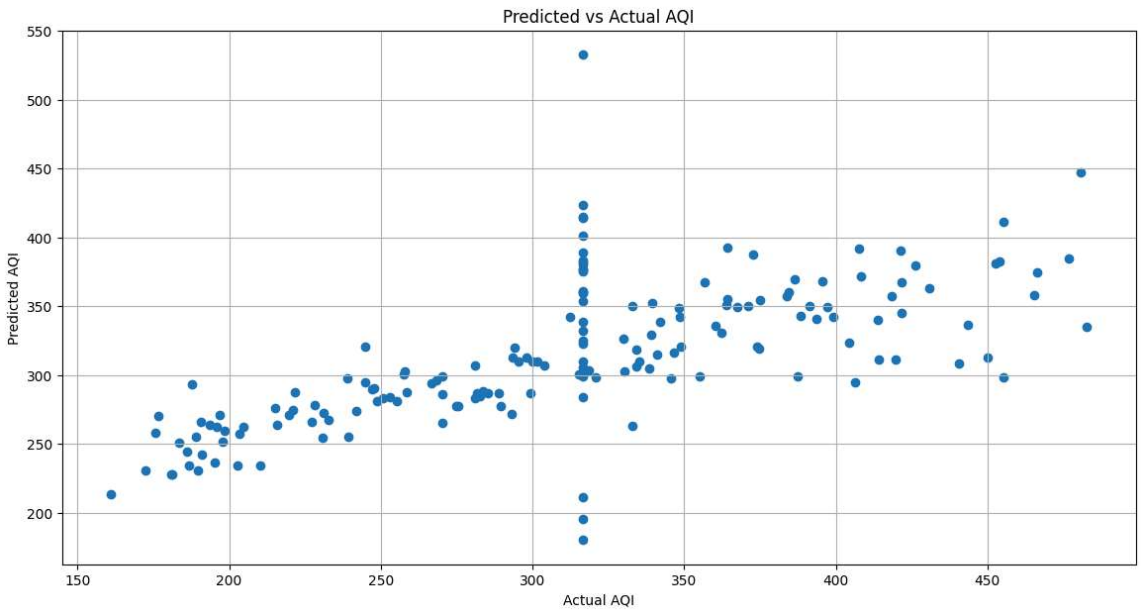


AQI Prediction Using Machine Learning

In [42]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Features and target variable
features = ['pm2_5', 'pm10', 'no2', 'so2', 'co']
X = data[features]
y = data['AQI'].fillna(data['AQI'].mean())

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra

# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict AQI on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

# Plot predicted vs actual AQI
plt.figure(figsize=(14, 7))
plt.scatter(y_test, y_pred)
plt.xlabel('Actual AQI')
plt.ylabel('Predicted AQI')
plt.title('Predicted vs Actual AQI')
plt.grid(True)
plt.show()
```

```
Mean Squared Error: 3370.6690948182363
R-squared: 0.4520081831198166
```



In [ ]: