In [45]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [46]:
```
df=pd.read_csv("AirQualityData.csv")
```

In [47]:
```
df.head()
```

Out[47]:

	Date	Time	CO(GT)	NOx(GT)	NO2(GT)	O3(GT)	SO2(GT)	PM2.5	
0	2024-01-01	00:00	3.807947	172.026768	144.333317	118.120832	1.215679	147.349671	208.8
1	2024-01-01	01:00	9.512072	241.824266	137.769318	15.325830	1.016178	40.979839	145.5
2	2024-01-01	02:00	7.346740	228.288118	20.055086	44.377036	24.140910	72.594740	26.1
3	2024-01-01	03:00	6.026719	47.016072	184.591909	139.488603	2.435392	134.339724	276.3
4	2024-01-01	04:00	1.644585	45.625591	114.125968	95.634768	48.752095	99.007422	294.2

5 rows × 23 columns

In [48]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 23 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Date                 4000 non-null   object 
 1   Time                 4000 non-null   object 
 2   CO(GT)               4000 non-null   float64
 3   NOx(GT)              4000 non-null   float64
 4   NO2(GT)              4000 non-null   float64
 5   O3(GT)               4000 non-null   float64
 6   SO2(GT)              4000 non-null   float64
 7   PM2.5                4000 non-null   float64
 8   PM10                 4000 non-null   float64
 9   Temperature          4000 non-null   float64
 10  Humidity             4000 non-null   float64
 11  Pressure             4000 non-null   float64
 12  WindSpeed            4000 non-null   float64
 13  WindDirection        4000 non-null   float64
 14  CO_NOx_Ratio         4000 non-null   float64
 15  NOx_NO2_Ratio        4000 non-null   float64
 16  Temp_Humidity_Index  4000 non-null   float64
 17  AirQualityIndex      4000 non-null   float64
 18  CO_MA3               4000 non-null   float64
 19  NO2_MA3              4000 non-null   float64
 20  O3_MA3               4000 non-null   float64
 21  DayOfWeek            4000 non-null   int64  
 22  Hour                 4000 non-null   int64  
dtypes: float64(19), int64(2), object(2)
memory usage: 718.9+ KB
```

In [49]:
```
df.isnull().sum()
```

Out[49]:
```
Date                    0
Time                    0
CO(GT)                  0
NOx(GT)                 0
NO2(GT)                 0
O3(GT)                  0
SO2(GT)                 0
PM2.5                   0
PM10                    0
Temperature             0
Humidity                0
Pressure                0
WindSpeed               0
WindDirection           0
CO_NOx_Ratio            0
NOx_NO2_Ratio           0
Temp_Humidity_Index     0
AirQualityIndex         0
CO_MA3                  0
NO2_MA3                 0
O3_MA3                  0
DayOfWeek               0
Hour                    0
dtype: int64
```

In [50]:
```
df.describe()
```

Out[50]:

	CO(GT)	NOx(GT)	NO2(GT)	O3(GT)	SO2(GT)	PM2.5	
count	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000
mean	5.025385	148.126633	100.213189	89.914815	26.081045	104.765999	153
std	2.874632	85.999247	57.074947	52.003484	14.059684	56.344868	83
min	0.100115	1.009185	1.010513	1.055442	1.012370	5.009384	10
25%	2.514242	73.636615	51.326622	44.179487	14.220565	56.544378	82
50%	5.054973	146.440690	99.508855	88.956924	26.321359	105.502686	154
75%	7.524652	221.823697	149.666167	136.333683	37.833728	153.751364	222
max	9.997205	299.838744	199.934968	179.986544	49.993700	199.980691	299

8 rows × 21 columns

In [51]:
```
print(df.duplicated().sum())
```

```
0
```

In [52]:
```
print("Original Shape:", df.shape)
```

```
Original Shape: (4000, 23)
```

In [53]:

```python
# calculate mean, median, variance, and standard deviation
import pandas as pd
df = pd.read_csv("AirQualityData.csv")

numeric_df = df.select_dtypes(include=['number'])
if numeric_df.empty:
    print("\nNo numeric columns found in the dataset.")
else:
    mean = numeric_df.mean()
    median = numeric_df.median()
    var = numeric_df.var()
    std = numeric_df.std()
    print("\nMean:\n",mean)
    print("\nMedian:\n",median)
    print("\nVariance:\n",var)
    print("\nStandard Deviation:\n",std)
```

```
Mean:
 CO(GT)                   5.025385
NOx(GT)                 148.126633
NO2(GT)                 100.213189
O3(GT)                   89.914815
SO2(GT)                  26.081045
PM2.5                   104.765999
PM10                    153.591417
Temperature              17.305228
Humidity                 54.626284
Pressure                999.862679
WindSpeed                9.984818
WindDirection          179.571724
CO_NOx_Ratio             0.082564
NOx_NO2_Ratio            3.412176
Temp_Humidity_Index      9.418823
AirQualityIndex        249.602455
CO_MA3                   5.025846
NO2_MA3                 100.222672
O3_MA3                   89.901700
DayOfWeek                2.978000
Hour                    11.484000
dtype: float64

Median:
 CO(GT)                   5.054973
NOx(GT)                 146.440690
NO2(GT)                  99.508855
O3(GT)                   88.956924
SO2(GT)                  26.321359
PM2.5                   105.502686
PM10                    154.714484
Temperature              17.184773
Humidity                 55.113650
Pressure                999.857722
WindSpeed                9.869314
WindDirection          179.401393
CO_NOx_Ratio             0.034021
NOx_NO2_Ratio            1.467460
Temp_Humidity_Index      6.975287
AirQualityIndex        250.552671
CO_MA3                   5.042259
NO2_MA3                 100.128466
O3_MA3                   90.452301
DayOfWeek                3.000000
Hour                    11.000000
dtype: float64

Variance:
 CO(GT)                     8.263510
NOx(GT)                 7395.870548
NO2(GT)                 3257.549529
O3(GT)                  2704.362320
SO2(GT)                  197.674722
PM2.5                   3174.744196
PM10                    6902.437792
```

```
Temperature              167.537601
Humidity                 667.912501
Pressure                 835.043404
WindSpeed                 33.694077
WindDirection          10970.207895
CO_NOx_Ratio               0.046480
NOx_NO2_Ratio             63.937013
Temp_Humidity_Index       80.677933
AirQualityIndex        20612.611529
CO_MA3                     2.797976
NO2_MA3                 1081.414677
O3_MA3                   930.575246
DayOfWeek                  3.970509
Hour                      47.885715
dtype: float64

Standard Deviation:
 CO(GT)                    2.874632
NOx(GT)                  85.999247
NO2(GT)                  57.074947
O3(GT)                   52.003484
SO2(GT)                  14.059684
PM2.5                    56.344868
PM10                     83.080911
Temperature              12.943632
Humidity                 25.844003
Pressure                 28.897118
WindSpeed                 5.804660
WindDirection           104.738760
CO_NOx_Ratio              0.215593
NOx_NO2_Ratio             7.996062
Temp_Humidity_Index       8.982090
AirQualityIndex         143.570929
CO_MA3                    1.672715
NO2_MA3                  32.884870
O3_MA3                   30.505331
DayOfWeek                 1.992614
Hour                      6.919951
dtype: float64
```

In [54]:
```
#standard scaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['Temperature', 'Pressure', 'Humidity']] = scaler.fit_transform(df[['Temperature
df[['Temperature', 'Pressure', 'Humidity']].head()
```

Out[54]:

	Temperature	Pressure	Humidity
0	0.869985	1.446573	-1.634126
1	-0.812241	1.562935	-1.699246
2	0.551021	-1.710477	-0.945502
3	0.707674	-0.628348	1.056958
4	-0.523481	1.172413	1.424145

In [58]:
```
# Random Forest Algorithm
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

In [59]:
```
df = pd.read_csv("AirQualityData.csv")

le = LabelEncoder()
for col in df.select_dtypes(include='object').columns:
    df[col] = le.fit_transform(df[col])

X = df.drop('Time', axis=1)
y = df['Time']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)
```

In [60]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,y_pred))
```

```
Accuracy: 0.94625

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      1.00      1.00        35
           2       0.97      1.00      0.99        37
           3       1.00      0.97      0.98        30
           4       0.94      1.00      0.97        33
           5       0.97      0.97      0.97        30
           6       1.00      0.93      0.96        44
           7       0.96      0.96      0.96        25
           8       0.90      1.00      0.95        37
           9       0.85      0.85      0.85        34
          10       0.91      0.81      0.86        37
          11       0.77      0.84      0.81        32
          12       0.92      0.92      0.92        24
          13       1.00      0.94      0.97        35
          14       0.82      0.96      0.89        28
          15       0.91      0.88      0.90        34
          16       0.91      0.91      0.91        32
          17       0.91      0.86      0.89        36
          18       0.97      0.92      0.95        39
          19       1.00      1.00      1.00        33
          20       1.00      1.00      1.00        32
          21       1.00      1.00      1.00        27
          22       1.00      1.00      1.00        32
          23       1.00      1.00      1.00        31

    accuracy                           0.95       800
   macro avg       0.95      0.95      0.95       800
weighted avg       0.95      0.95      0.95       800
```

In [74]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
mae = mean_absolute_error(y_test,y_pred)
mse = mean_squared_error(y_test,y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("R-squared (R²):",r2)
```

```
Mean Absolute Error (MAE): 3.8
Mean Squared Error (MSE): 22.525
R-squared (R²): 0.5378960499472312
```

In [75]:
```
#svm algorithm with confusion matrix
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder
```

In [76]:
```
df = pd.read_csv('AirQualityData.csv')
df = df.dropna()
df['Temperature'] = df['Temperature'].astype(str)

X = df['Temperature']
y = df['Date']

le = LabelEncoder()
y = le.fit_transform(y)

vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, rand

svm_clf = svm.SVC(kernel='linear')
svm_clf.fit(X_train, y_train)

y_pred = svm_clf.predict(X_test)
```

In [77]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

```
Accuracy: 0.00375
Confusion Matrix:
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

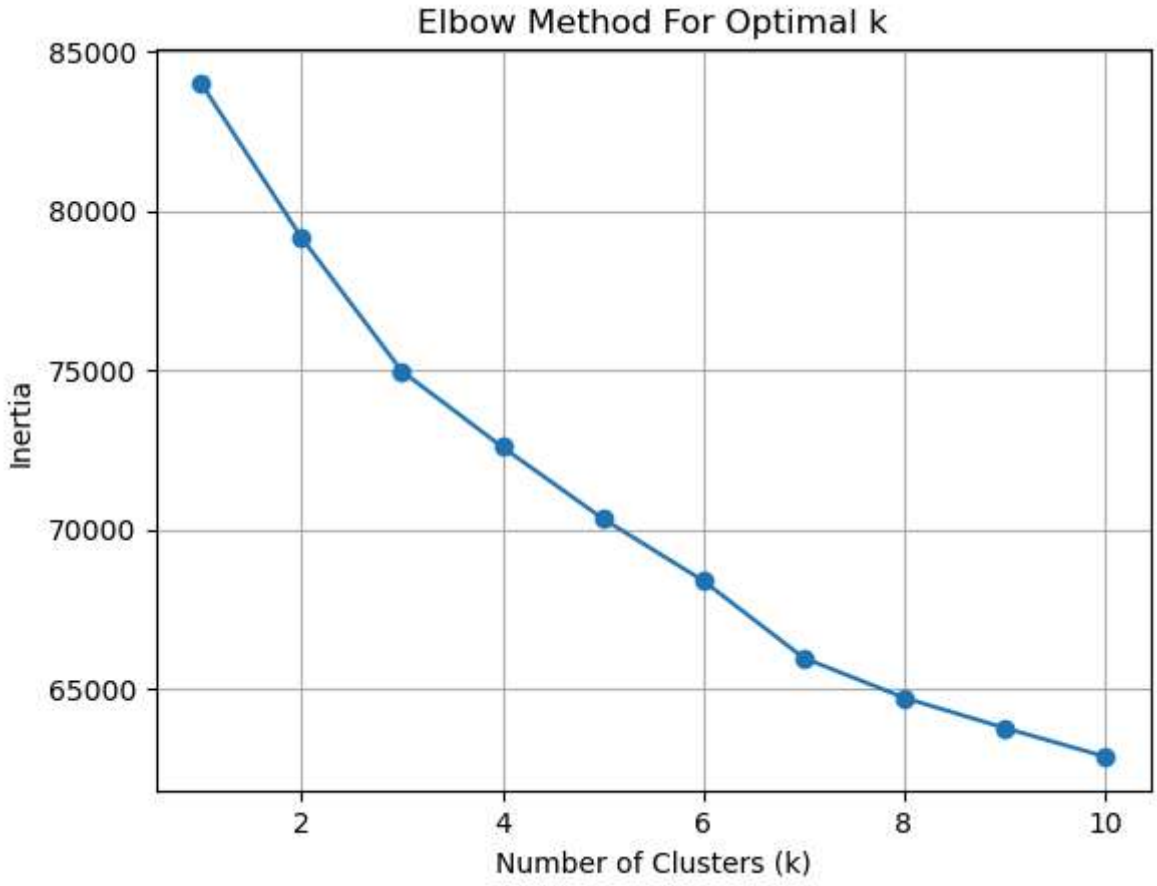
In [78]:
```
# KNN-Algorithm
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

df = pd.read_csv("AirQualityData.csv")

le = LabelEncoder()
for col in df.select_dtypes(include='object').columns:
    df[col] = le.fit_transform(df[col])

X = df.drop('Date', axis=1)
y = df['Time']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, ran
```

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

print("Predictions:", y_pred[:10])
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred,zero_divisio
```

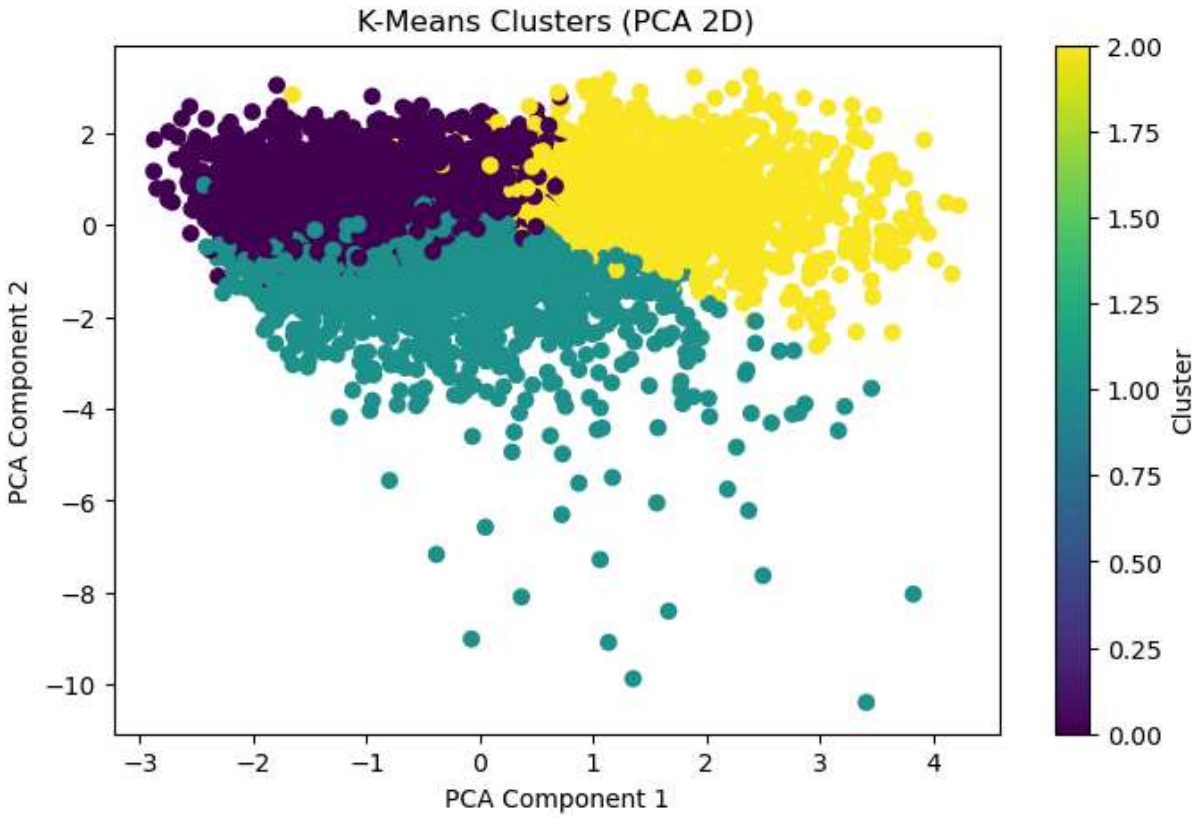
```
Predictions: [ 5  7 12 11 10 19  6 12  3 11]
Accuracy: 0.07625
Classification Report:
              precision    recall  f1-score   support

           0       0.18      0.30      0.23        43
           1       0.02      0.03      0.02        35
           2       0.13      0.19      0.16        37
           3       0.02      0.03      0.03        30
           4       0.07      0.09      0.08        33
           5       0.05      0.07      0.06        30
           6       0.05      0.05      0.05        44
           7       0.05      0.08      0.06        25
           8       0.03      0.03      0.03        37
           9       0.06      0.06      0.06        34
          10       0.03      0.03      0.03        37
          11       0.07      0.06      0.07        32
          12       0.05      0.08      0.06        24
          13       0.13      0.14      0.14        35
          14       0.06      0.07      0.07        28
          15       0.04      0.03      0.03        34
          16       0.09      0.06      0.07        32
          17       0.05      0.03      0.04        36
          18       0.05      0.03      0.03        39
          19       0.10      0.06      0.07        33
          20       0.05      0.03      0.04        32
          21       0.13      0.07      0.10        27
          22       0.23      0.16      0.19        32
          23       0.00      0.00      0.00        31

    accuracy                           0.08       800
   macro avg       0.07      0.07      0.07       800
weighted avg       0.07      0.08      0.07       800
```

In [79]:
```
#k means clustering
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

df = pd.read_csv('AirQualityData.csv')
df_numeric = df.select_dtypes(include=[np.number])
```

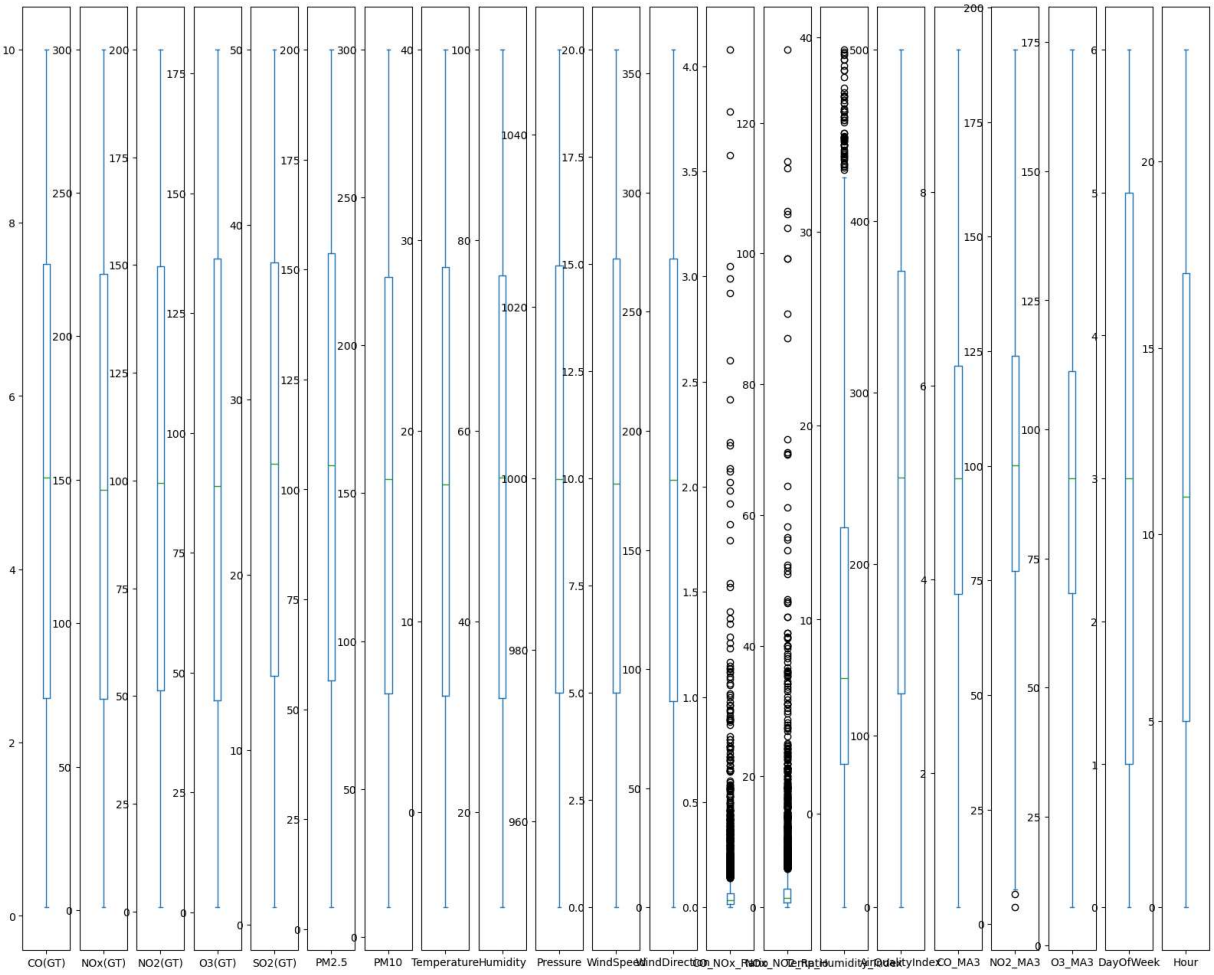
```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_numeric)

inertia = []
k_range = range(1, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    inertia.append(kmeans.inertia_)

plt.plot(k_range, inertia, marker='o')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
plt.grid(True)
plt.show()

kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_data)

print(df[['Cluster']].value_counts())

pca = PCA(n_components=2)
reduced_data = pca.fit_transform(scaled_data)

plt.figure(figsize=(8, 5))
plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=df['Cluster'], cmap='viridis'
plt.title('K-Means Clusters (PCA 2D)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster')
plt.show()
```



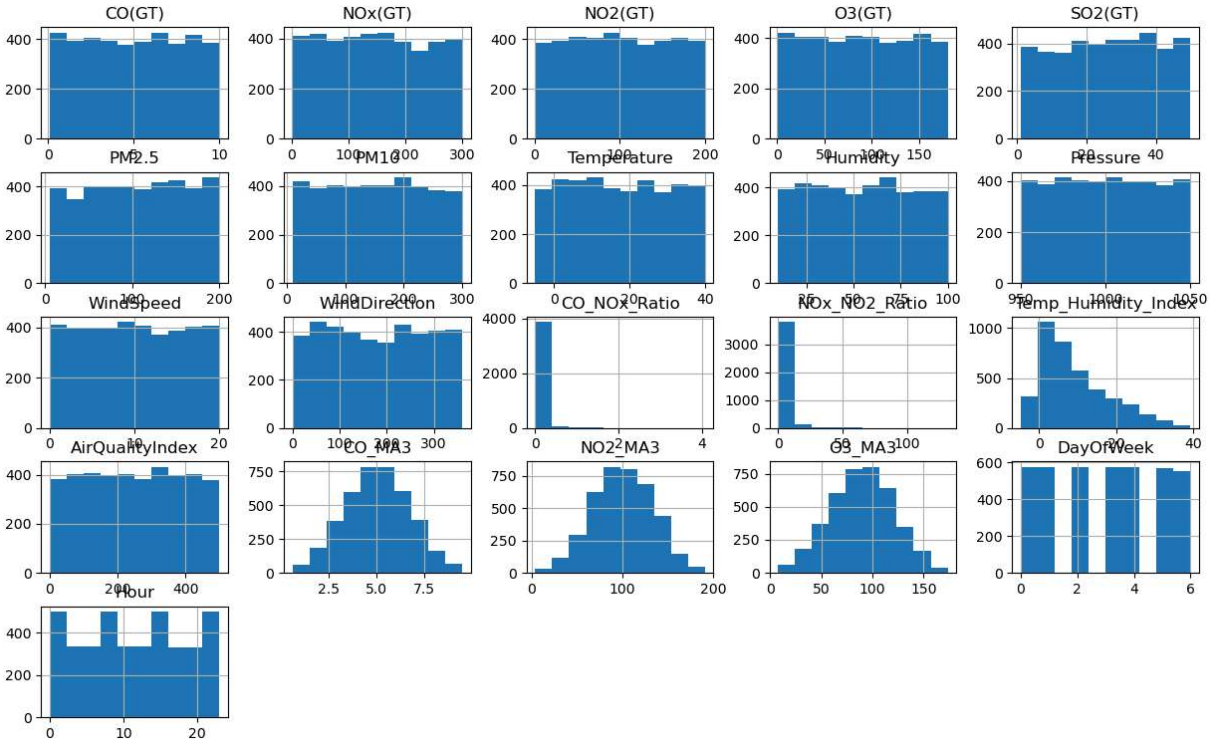

```
Cluster
0          1522
1          1350
2          1128
Name: count, dtype: int64
```



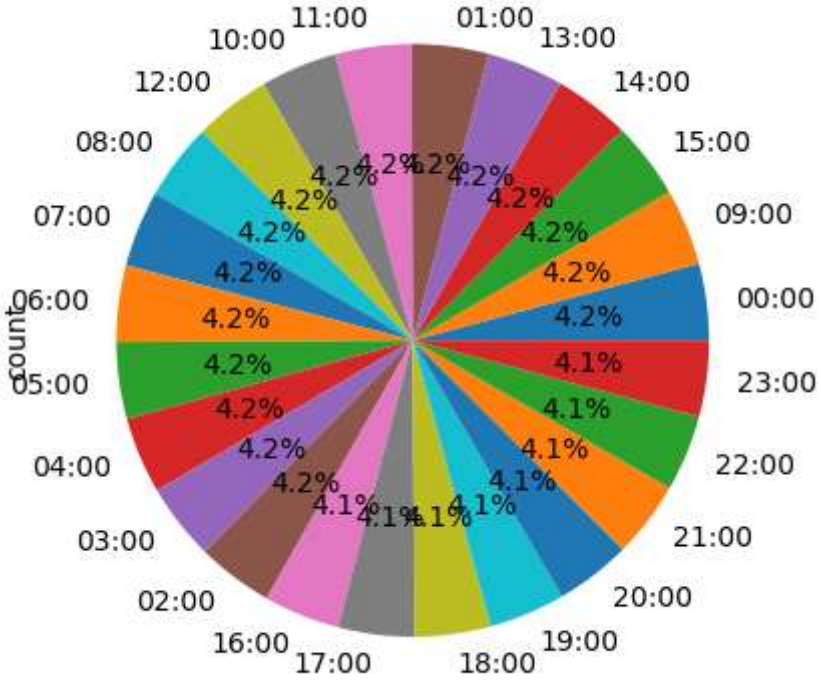
In [80]:
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('AirQualityData.csv')
df.plot(kind='box', figsize=(20,16), subplots=True, layout=(1, len(df.select_dtypes
plt.show()
```



In [81]:
```
# histogram plotting
import matplotlib.pyplot as plt
df.hist(figsize=(15,9))
plt.show()
```

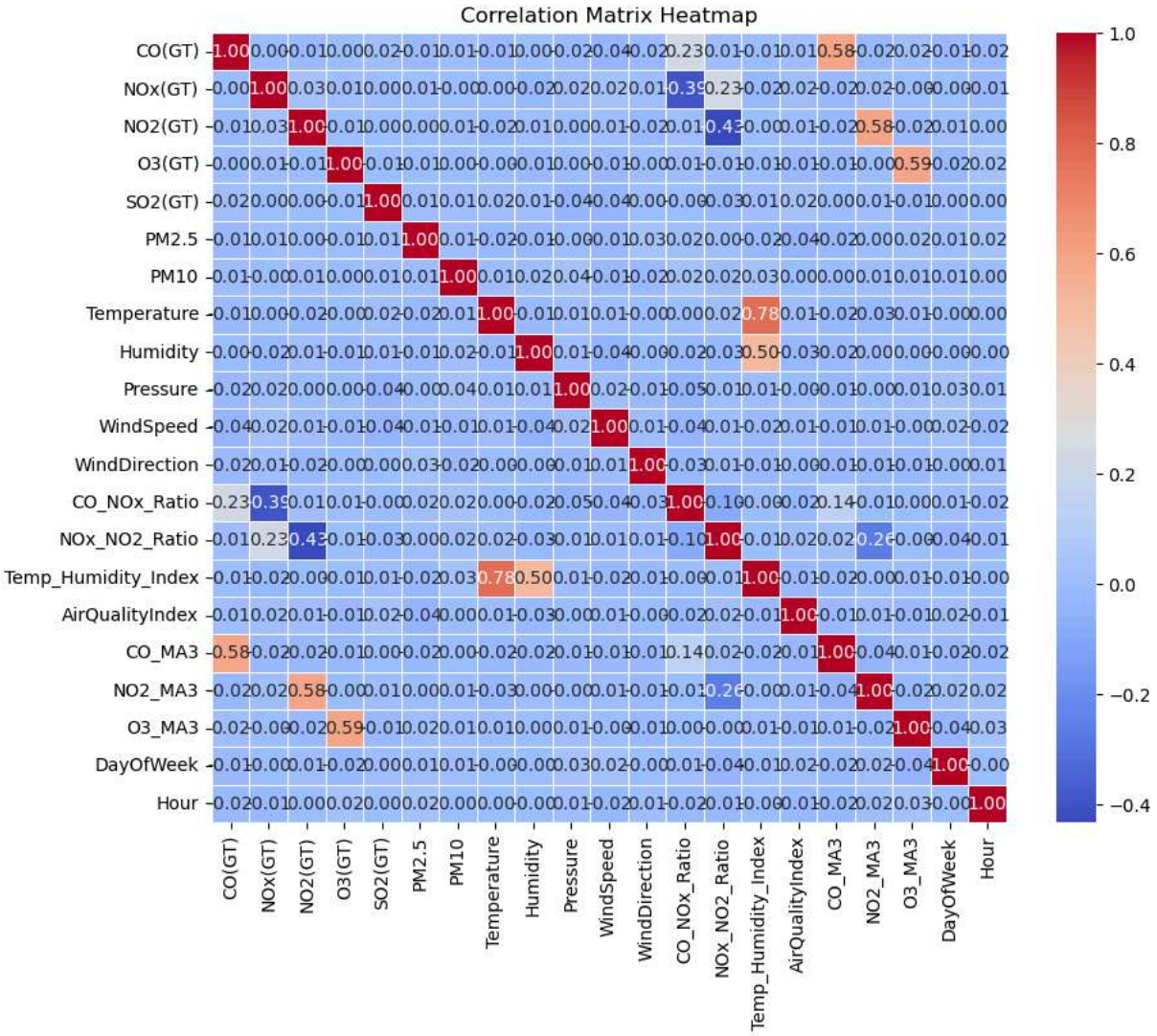


In [82]:
```
import matplotlib.pyplot as plt
df['Time'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.show()
```

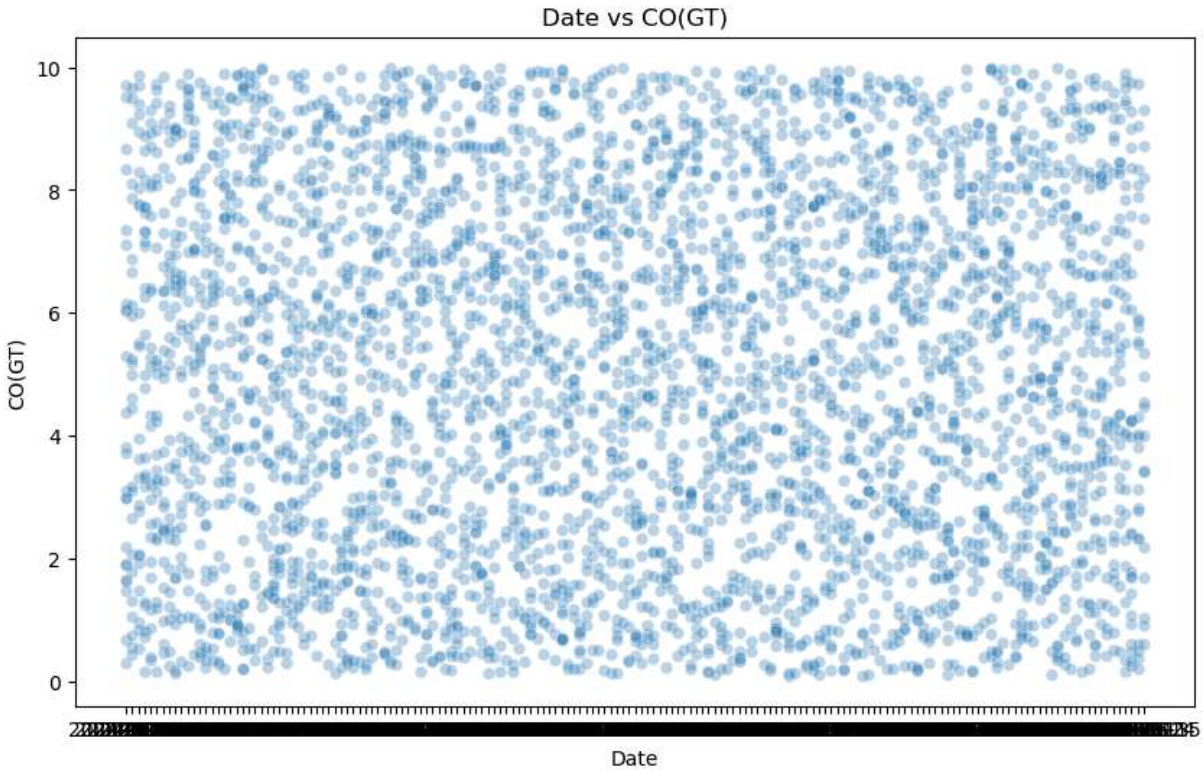


In [83]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
corr_matrix = df.corr(numeric_only=True)  # Ensures only numeric columns are used
plt.figure(figsize=(10, 8))
```

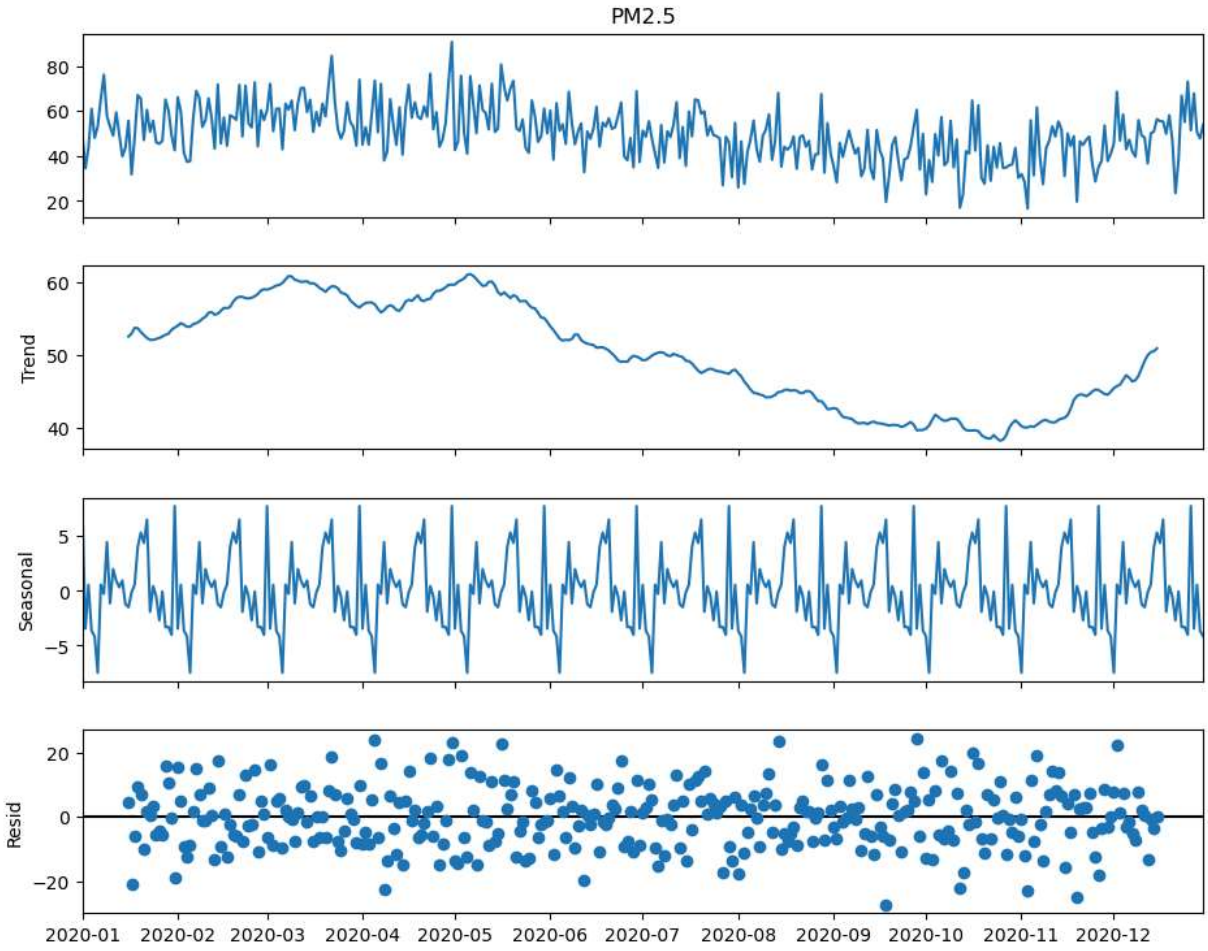
```
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



In [84]:
```
# Scatter plot
plt.figure(figsize=(10,6))
sns.scatterplot(x=df['Date'], y=df['CO(GT)'], alpha=0.3)
plt.title('Date vs CO(GT)')
plt.xlabel('Date')
plt.ylabel('CO(GT)')
plt.show()
```



In [85]:
```
#Time series Decomposition
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose

date_range = pd.date_range(start='2020-01-01', periods=365, freq='D')
import numpy as np
data = np.random.normal(loc=50, scale=10, size=365) + np.sin(np.linspace(0, 3.14*2,
df = pd.DataFrame({'date': date_range, 'PM2.5': data})
df.set_index('date', inplace=True)

decomposition = seasonal_decompose(df['PM2.5'], model='additive', period=30)

fig = decomposition.plot()
fig.set_size_inches(10,8)
plt.show()
```


PM2.5
80
60
40
20
Trend
60
50
40
Seasonal
5
0
−5
Resid
20
0
−20
2020-01
2020-02
2020-03
2020-04
2020-05
2020-06
2020-07
2020-08
2020-09
2020-10
2020-11
2020-12

In [ ]:

In [ ]:

In [ ]: