```
import pandas as pd
```

```
df = pd.read_csv('/content/Combine.csv')
```

```
df.head()
```

	Timestamp	PM2.5 (µg/m³)	PM10 (µg/m³)
0	06-09-2017	6.82	NaN
1	07-09-2017	51.65	97.00
2	08-09-2017	43.45	80.57
3	09-09-2017	70.70	139.33
4	10-09-2017	46.42	89.83

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

```
df.tail()
```

	Timestamp	PM2.5 (µg/m³)	PM10 (µg/m³)
2646	27-12-2024	122.81	126.14
2647	28-12-2024	68.66	104.18
2648	29-12-2024	62.89	71.82
2649	30-12-2024	66.95	101.48
2650	31-12-2024	95.25	135.80

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2651 entries, 0 to 2650
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Timestamp      2651 non-null   object 
 1   PM2.5 (µg/m³)  2602 non-null   float64
 2   PM10 (µg/m³)   2596 non-null   float64
dtypes: float64(2), object(1)
memory usage: 62.3+ KB
```

```
df['Timestamp'] = pd.to_datetime(df['Timestamp'], format='%d-%m-%Y')
```

```
df = df.set_index('Timestamp')
```

```
df.isnull().sum()
```

	0
PM2.5 (µg/m³)	24
PM10 (µg/m³)	30

dtype: int64

```
df = df.dropna(subset = ['PM2.5 (µg/m³)','PM10 (µg/m³)'], how = 'all')
```

```
df['Year'] = df.index.year
df['Month'] = df.index.month
df['Month_Name'] = df.index.month_name()
```

```
<ipython-input-16-74e120e23321>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  df['Year'] = df.index.year
<ipython-input-16-74e120e23321>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  df['Month'] = df.index.month
<ipython-input-16-74e120e23321>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  df['Month_Name'] = df.index.month_name()
```

```
df
```

Timestamp	PM2.5 (µg/m³)	PM10 (µg/m³)	Year	Month	Month_Name
2017-09-06	6.82	NaN	2017	9	September
2017-09-07	51.65	97.00	2017	9	September
2017-09-08	43.45	80.57	2017	9	September
2017-09-09	70.70	139.33	2017	9	September
2017-09-10	46.42	89.83	2017	9	September
...	...	...	...	...	...
2024-12-27	122.81	126.14	2024	12	December
2024-12-28	68.66	104.18	2024	12	December
2024-12-29	62.89	71.82	2024	12	December
2024-12-30	66.95	101.48	2024	12	December
2024-12-31	95.25	135.80	2024	12	December

2626 rows × 5 columns

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

```
def get_season(month):
    if month in [3, 4, 5]:
        return 'Summer'
    elif month in [6, 7, 8, 9]:
        return 'Monsoon'
    elif month in [10, 11]:
        return 'Post-Monsoon'
    else:
        return 'Winter'

df['Season'] = df['Month'].apply(get_season)
```

```
<ipython-input-18-a7e096451ac8>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  df['Season'] = df['Month'].apply(get_season)
```

```
df
```

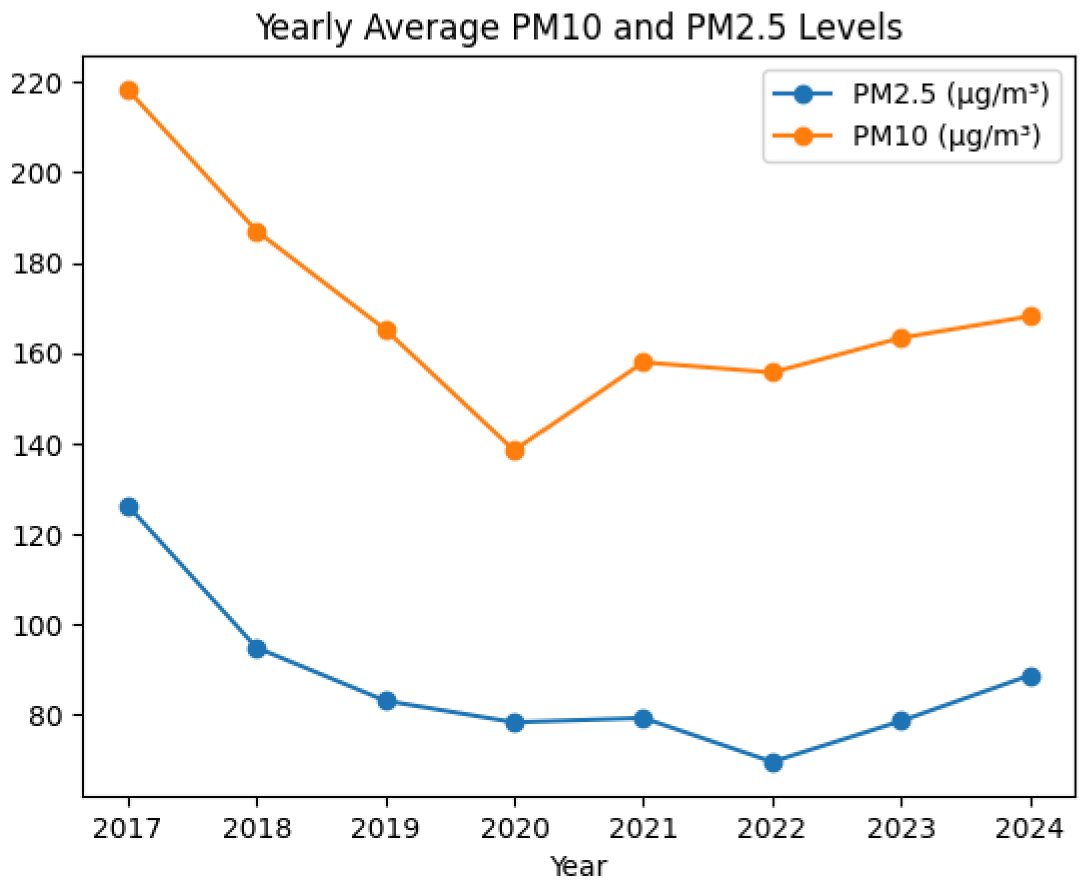
```
            PM2.5 (µg/m³)  PM10 (µg/m³)  Year  Month  Month_Name   Season
Timestamp
2017-09-06           6.82           NaN  2017      9   September  Monsoon
2017-09-07          51.65         97.00  2017      9   September  Monsoon
2017-09-08          43.45         80.57  2017      9   September  Monsoon
2017-09-09          70.70        139.33  2017      9   September  Monsoon
2017-09-10          46.42         89.83  2017      9   September  Monsoon
   ...               ...           ...    ...    ...         ...      ...
2024-12-27         122.81        126.14  2024     12    December   Winter
2024-12-28          68.66        104.18  2024     12    December   Winter
2024-12-29          62.89         71.82  2024     12    December   Winter
2024-12-30          66.95        101.48  2024     12    December   Winter
2024-12-31          95.25        135.80  2024     12    December   Winter

2626 rows × 6 columns
```

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

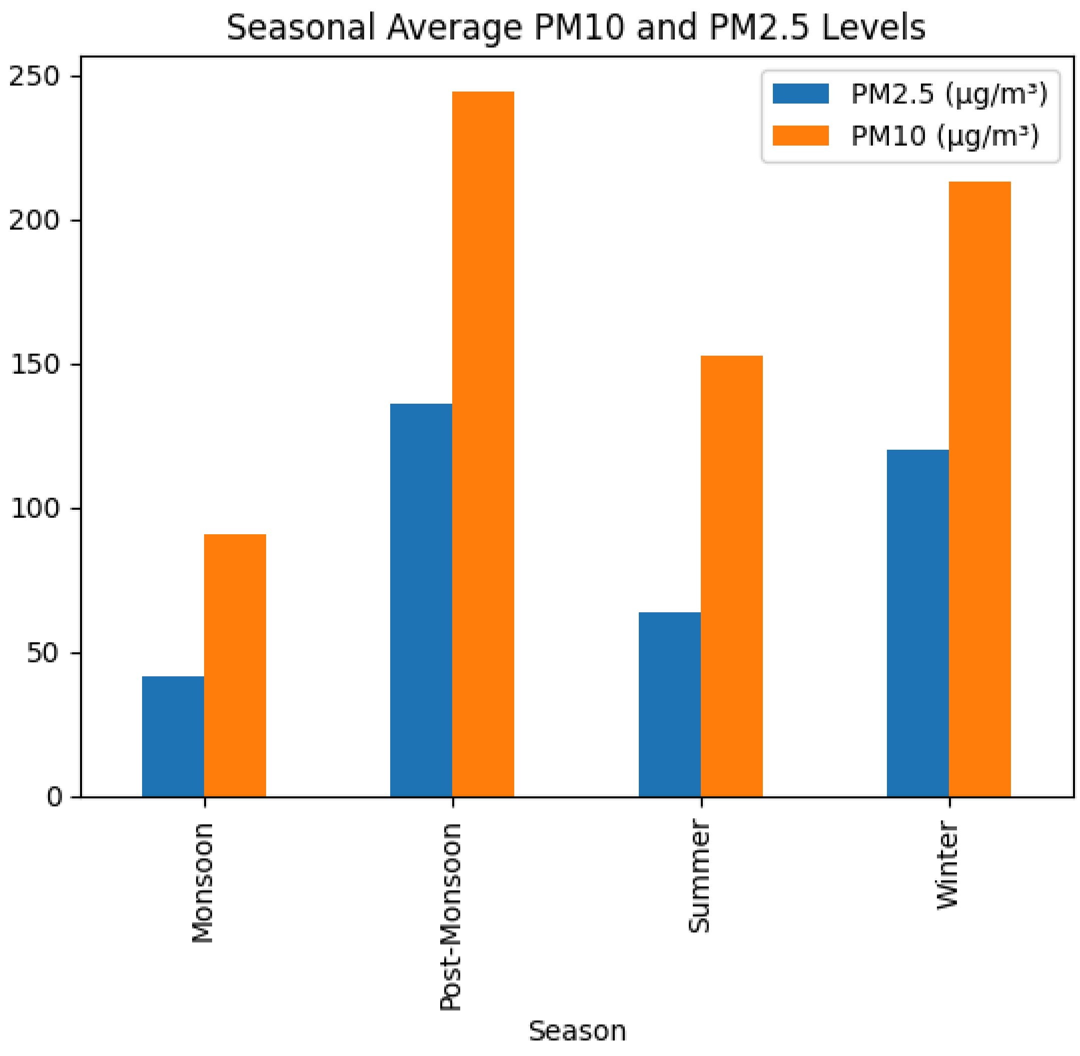
```
yearly_avg = df.groupby('Year')[['PM2.5 (µg/m³)','PM10 (µg/m³)']].mean()

yearly_avg.plot(kind='line', marker='o', title='Yearly Average PM10 and PM2.5
<Axes: title={'center': 'Yearly Average PM10 and PM2.5 Levels'}, xlabel='Year'>
```



```
seasonal_avg = df.groupby('Season')[['PM2.5 (µg/m³)','PM10 (µg/m³)']].mean()
seasonal_avg.plot(kind='bar', title='Seasonal Average PM10 and PM2.5 Levels')
```

```
<Axes: title={'center': 'Seasonal Average PM10 and PM2.5 Levels'}, xlabel='Season'>
```



```
top_pm10_days = df.sort_values('PM10 (µg/m³)', ascending=False).head(10)
top_pm25_days = df.sort_values('PM2.5 (µg/m³)', ascending=False).head(10)

top_pm10_days[['PM10 (µg/m³)','Month']]
```

Timestamp	PM10 (µg/m³)	Month
2017-11-09	842.820000	11
2017-11-07	787.070000	11
2024-11-18	748.770000	11
2018-06-13	701.990000	6
2020-11-09	665.410000	11
2017-11-12	663.180000	11
2024-11-17	620.120000	11
2023-11-04	607.997546	11
2018-06-14	604.040000	6
2019-11-03	591.420000	11

```
top_pm25_days[['PM2.5 (µg/m³)','Month']]
```

Timestamp	PM2.5 (µg/m³)	Month
2017-11-08	623.48	11
2024-11-18	558.02	11
2019-11-03	556.03	11
2020-11-09	516.11	11
2017-11-09	496.78	11
2017-11-07	464.10	11
2020-11-10	450.48	11
2024-11-17	429.44	11
2021-11-05	405.31	11
2019-11-01	401.48	11

```
import seaborn as sns
import plotly.express as px
```

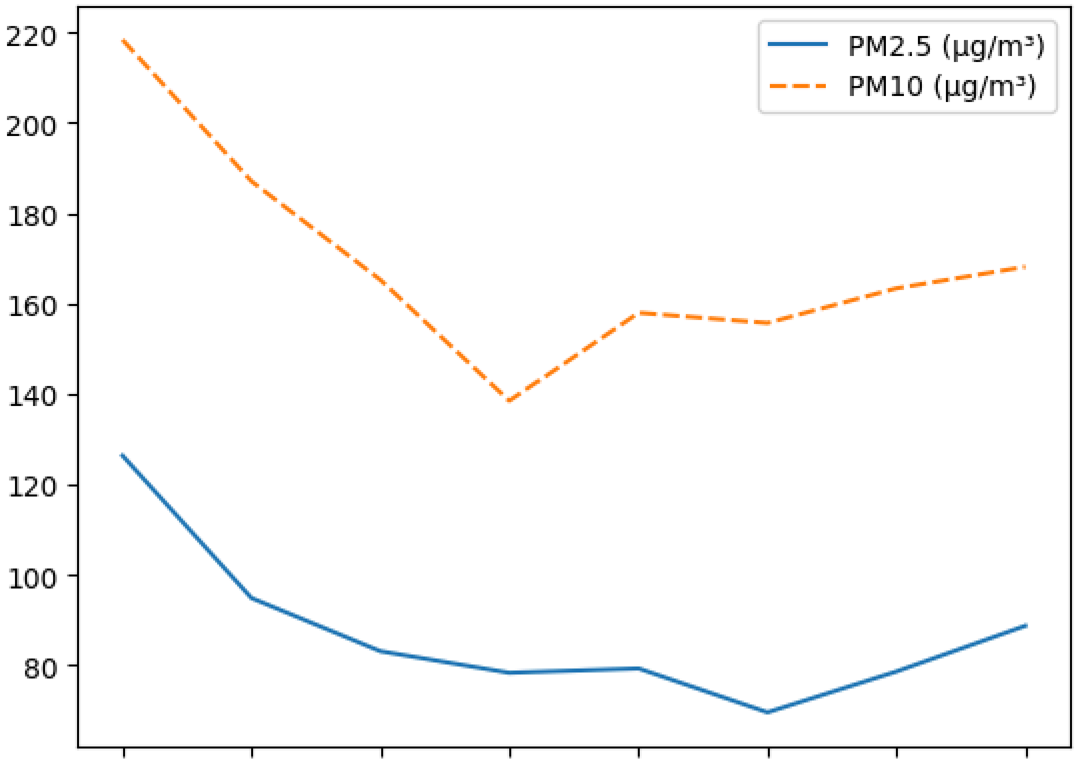
```
yearly_avg
```

Year	PM2.5 (µg/m³)	PM10 (µg/m³)
2017	126.421458	218.404409
2018	94.880966	187.074678
2019	83.106889	165.243323
2020	78.329497	138.464463
2021	79.284322	157.951551
2022	69.568380	155.697933
2023	78.614804	163.368600
2024	88.732630	168.136858

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

```
sns.lineplot(data=yearly_avg)
```

```
<Axes: xlabel='Year'>
```


PM2.5 (µg/m³)
PM10 (µg/m³)