```
In [1]: import pandas as pd
        from statsmodels.tsa.seasonal import STL
        import seaborn as sns
        import matplotlib.pyplot as plt
        import numpy as np
        import statsmodels.api as sm
        from statsmodels.tsa.seasonal import seasonal_decompose
```

1. Wczytanie danych

```
In [3]: data = pd.read_excel('PyłyBydgoszcz.xlsx', names = ['Date', 'PM 2.5 [ug/m3]'], skiprows = 5, index_col = 'Date')
        data.index = pd.to_datetime(data.index)
```

```
In [4]: data
```

Out[4]:

Date	PM 2.5 [ug/m3]
2022-01-01	4.934921
2022-01-02	10.341562
2022-01-03	5.007348
2022-01-04	9.688094
2022-01-05	5.914476
...	...
2022-12-27	2.195314
2022-12-28	4.317930
2022-12-29	5.061757
2022-12-30	5.950761
2022-12-31	8.273009

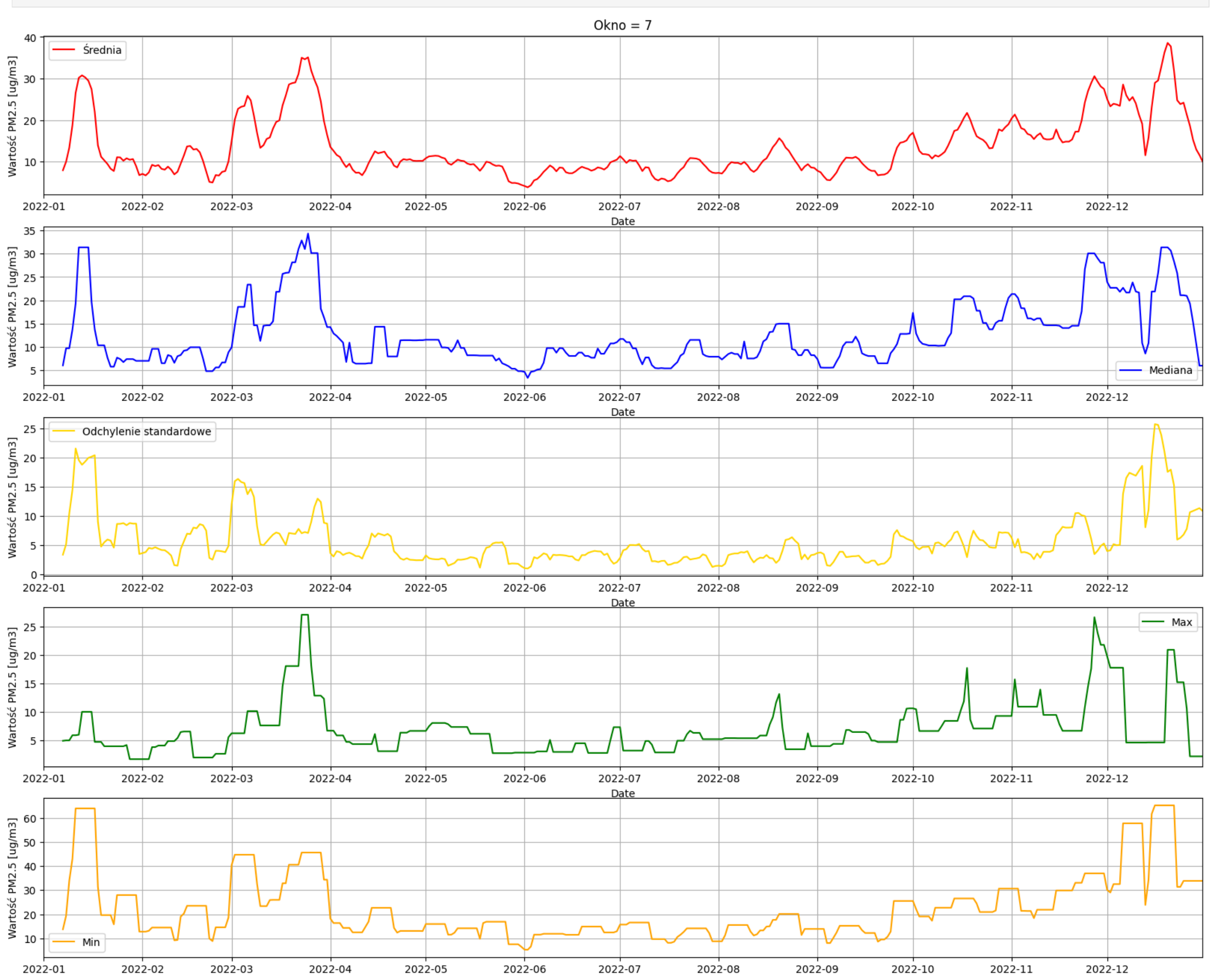
365 rows × 1 columns

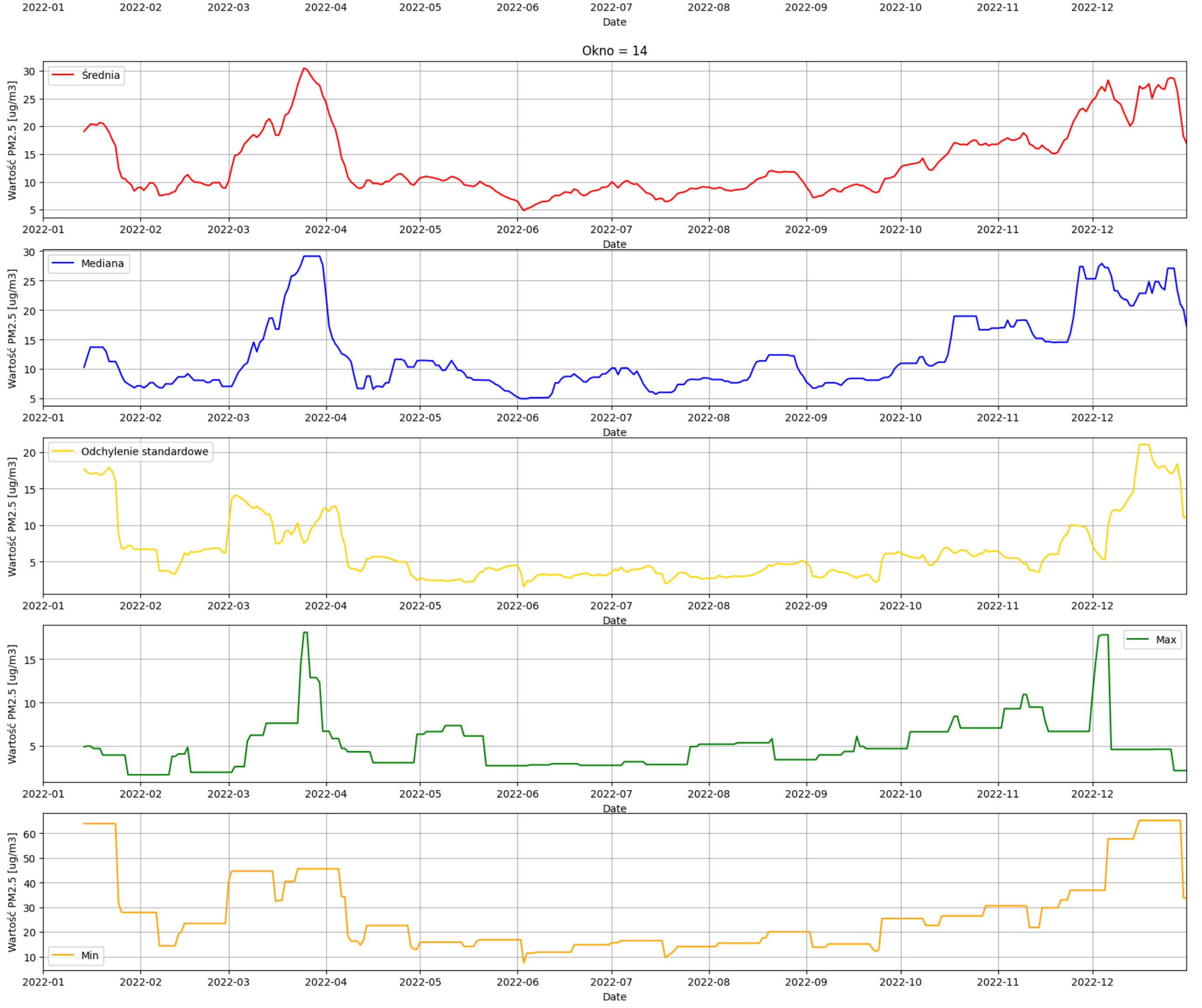
2. Zbiór cech

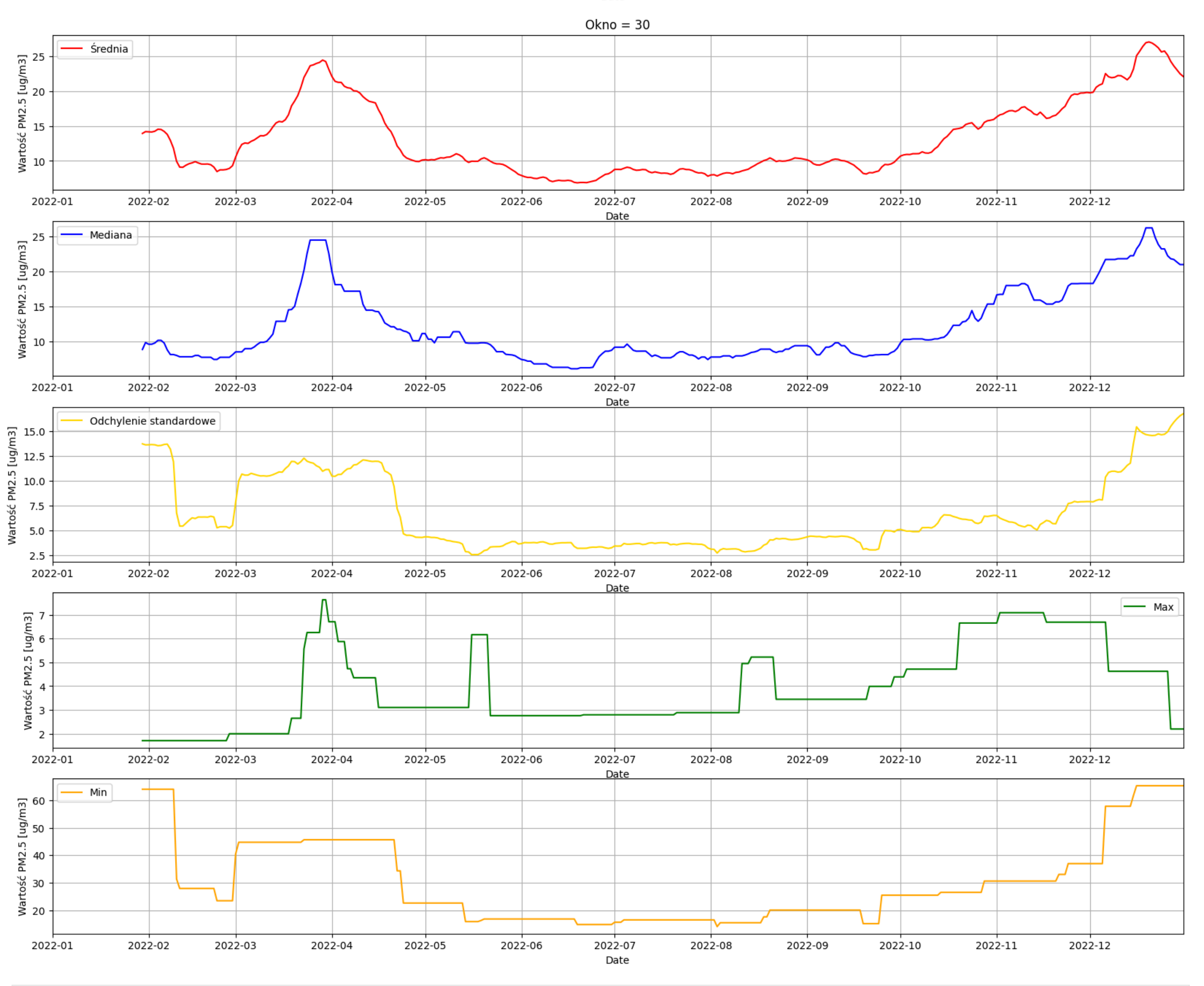
```
In [6]: stats = ["mean", "median", "std", "max", "min"]
```

2.1 Okna czasowe typu rolling

```
In [8]: windows = [7, 14, 30]
        for window in windows:
            rolling_df = data.rolling(window=window).agg(stats)
            fig, axes = plt.subplots(5, 1, figsize = [20, 16])
            axes[0].set_title(f'Okno = {window}')
            sns.lineplot(data = rolling_df, x = rolling_df.index, y = ('PM 2.5 [ug/m3]', 'mean'), ax = axes[0], color = 'red', label = 'Średnia')
            axes[0].grid()
            axes[0].set_xlim(np.min(data.index), np.max(data.index))
            axes[0].set_ylabel('Wartość PM2.5 [ug/m3]')

            sns.lineplot(data = rolling_df, x = rolling_df.index, y = ('PM 2.5 [ug/m3]', 'median'), ax = axes[1], color = 'blue', label = 'Mediana')
            axes[1].grid()
            axes[1].set_xlim(np.min(data.index), np.max(data.index))
            axes[1].set_ylabel('Wartość PM2.5 [ug/m3]')

            sns.lineplot(data = rolling_df, x = rolling_df.index, y = ('PM 2.5 [ug/m3]', 'std'), ax = axes[2], color = 'gold', label = 'Odchylenie standardowe')
            axes[2].grid()
            axes[2].set_xlim(np.min(data.index), np.max(data.index))
            axes[2].set_ylabel('Wartość PM2.5 [ug/m3]')

            sns.lineplot(data = rolling_df, x = rolling_df.index, y = ('PM 2.5 [ug/m3]', 'min'), ax = axes[3], color = 'green', label = 'Max')
            axes[3].grid()
            axes[3].set_xlim(np.min(data.index), np.max(data.index))
            axes[3].set_ylabel('Wartość PM2.5 [ug/m3]')

            sns.lineplot(data = rolling_df, x = rolling_df.index, y = ('PM 2.5 [ug/m3]', 'max'), ax = axes[4], color = 'orange', label = 'Min')
            axes[4].grid()
            axes[4].set_xlim(np.min(data.index), np.max(data.index))
            axes[4].set_ylabel('Wartość PM2.5 [ug/m3]')
```





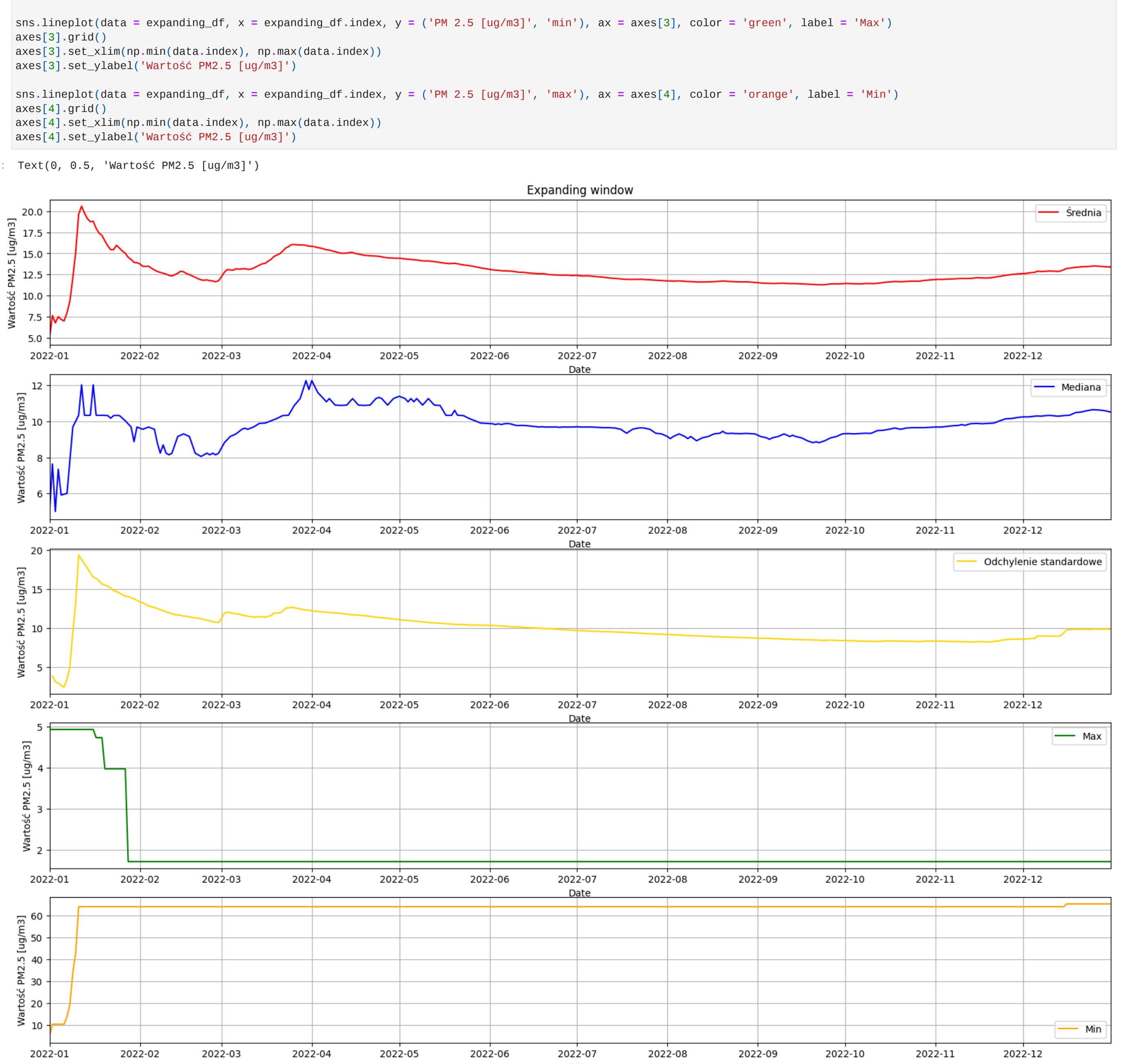


```
In [9]: rolling = data.rolling(window = 14).agg(stats)
        data['Mean'] = rolling['PM 2.5 [ug/m3]', 'mean']
        data['Median'] = rolling['PM 2.5 [ug/m3]', 'median']
        data['Std'] = rolling['PM 2.5 [ug/m3]', 'std']
        data['Min'] = rolling['PM 2.5 [ug/m3]', 'min']
        data['Max'] = rolling['PM 2.5 [ug/m3]', 'max']
```

2.2 Okna czasowe typu expanding

```
In [11]: expanding_df = data.expanding().agg(stats)
         fig, axes = plt.subplots(5, 1, figsize = [20, 16])
         axes[0].set_title(f'Expanding window')
         sns.lineplot(data = expanding_df, x = expanding_df.index, y = ('PM 2.5 [ug/m3]', 'mean'), ax = axes[0], color = 'red', label = 'Średnia')
         axes[0].grid()
         axes[0].set_xlim(np.min(data.index), np.max(data.index))
         axes[0].set_ylabel('Wartość PM2.5 [ug/m3]')

         sns.lineplot(data = expanding_df, x = expanding_df.index, y = ('PM 2.5 [ug/m3]', 'median'), ax = axes[1], color = 'blue', label = 'Mediana')
         axes[1].grid()
         axes[1].set_xlim(np.min(data.index), np.max(data.index))
         axes[1].set_ylabel('Wartość PM2.5 [ug/m3]')

         sns.lineplot(data = expanding_df, x = expanding_df.index, y = ('PM 2.5 [ug/m3]', 'std'), ax = axes[2], color = 'gold', label = 'Odchylenie standardowe')
         axes[2].grid()
         axes[2].set_xlim(np.min(data.index), np.max(data.index))
         axes[2].set_ylabel('Wartość PM2.5 [ug/m3]')

         sns.lineplot(data = expanding_df, x = expanding_df.index, y = ('PM 2.5 [ug/m3]', 'min'), ax = axes[3], color = 'green', label = 'Max')
         axes[3].grid()
         axes[3].set_xlim(np.min(data.index), np.max(data.index))
         axes[3].set_ylabel('Wartość PM2.5 [ug/m3]')

         sns.lineplot(data = expanding_df, x = expanding_df.index, y = ('PM 2.5 [ug/m3]', 'max'), ax = axes[4], color = 'orange', label = 'Min')
         axes[4].grid()
         axes[4].set_xlim(np.min(data.index), np.max(data.index))
         axes[4].set_ylabel('Wartość PM2.5 [ug/m3]')
```

```
Out[11]: Text(0, 0.5, 'Wartość PM2.5 [ug/m3]')
```



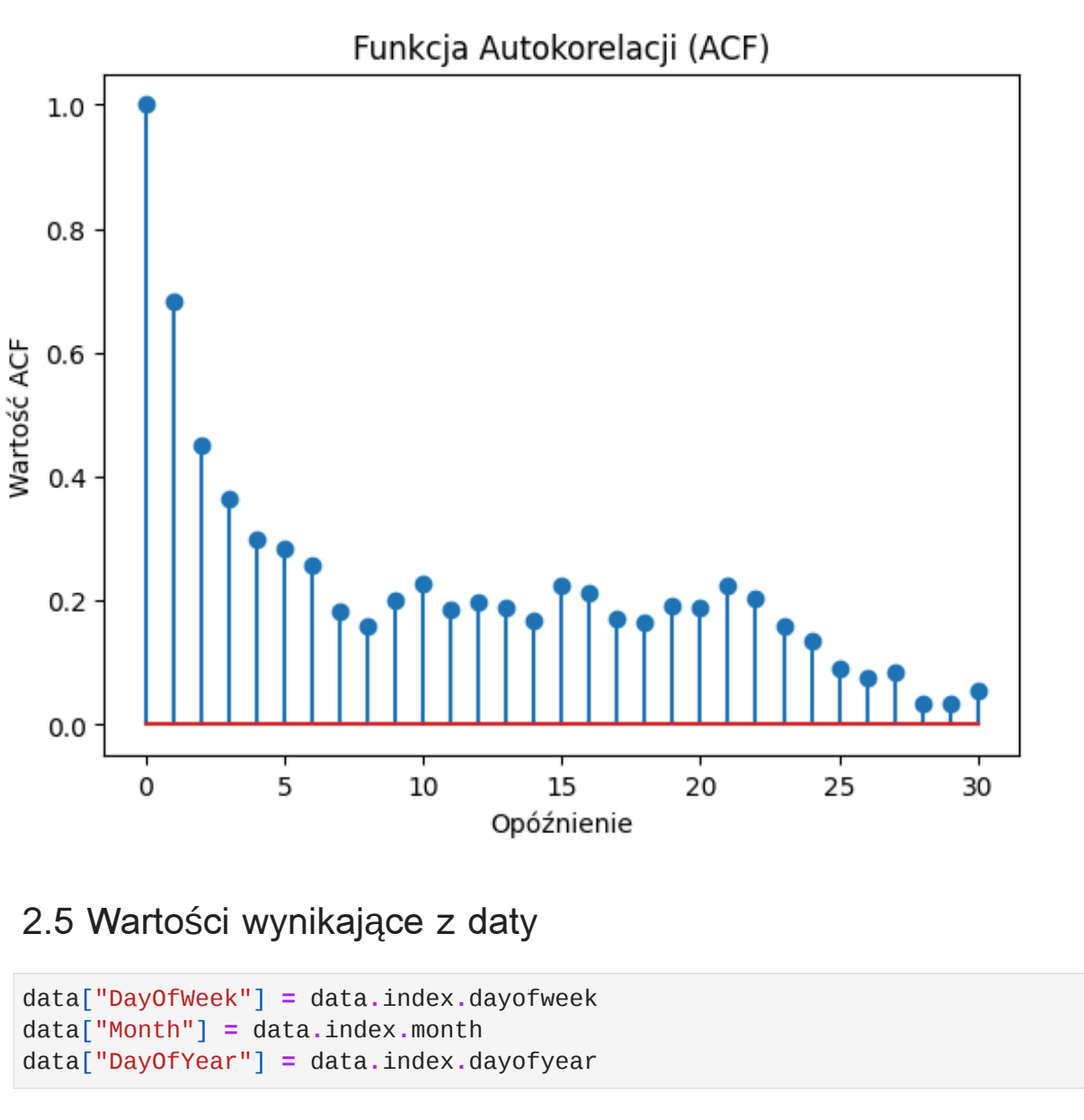
2.3 Okna czasowe typu nested

```
In [13]: nested_windows = [(7, 14), (7, 30), (14, 30)]
         for outer_window, inner_window in nested_windows:
             outer_stats = data.rolling(window=outer_window).agg(stats)
             nested_df = outer_stats.rolling(window=inner_window).agg(stats)
```

2.4 Wartości opóznione bazując na ACF

```
In [15]: acf = sm.tsa.acf(data["PM 2.5 [ug/m3]"], nlags = 30)

         plt.stem(range(len(acf)), acf)
         plt.title("Funkcja Autokorelacji (ACF)")
         plt.xlabel("Opóźnienie")
         plt.ylabel("Wartość ACF")
```

```
Out[15]: Text(0, 0.5, 'Wartość ACF')
```



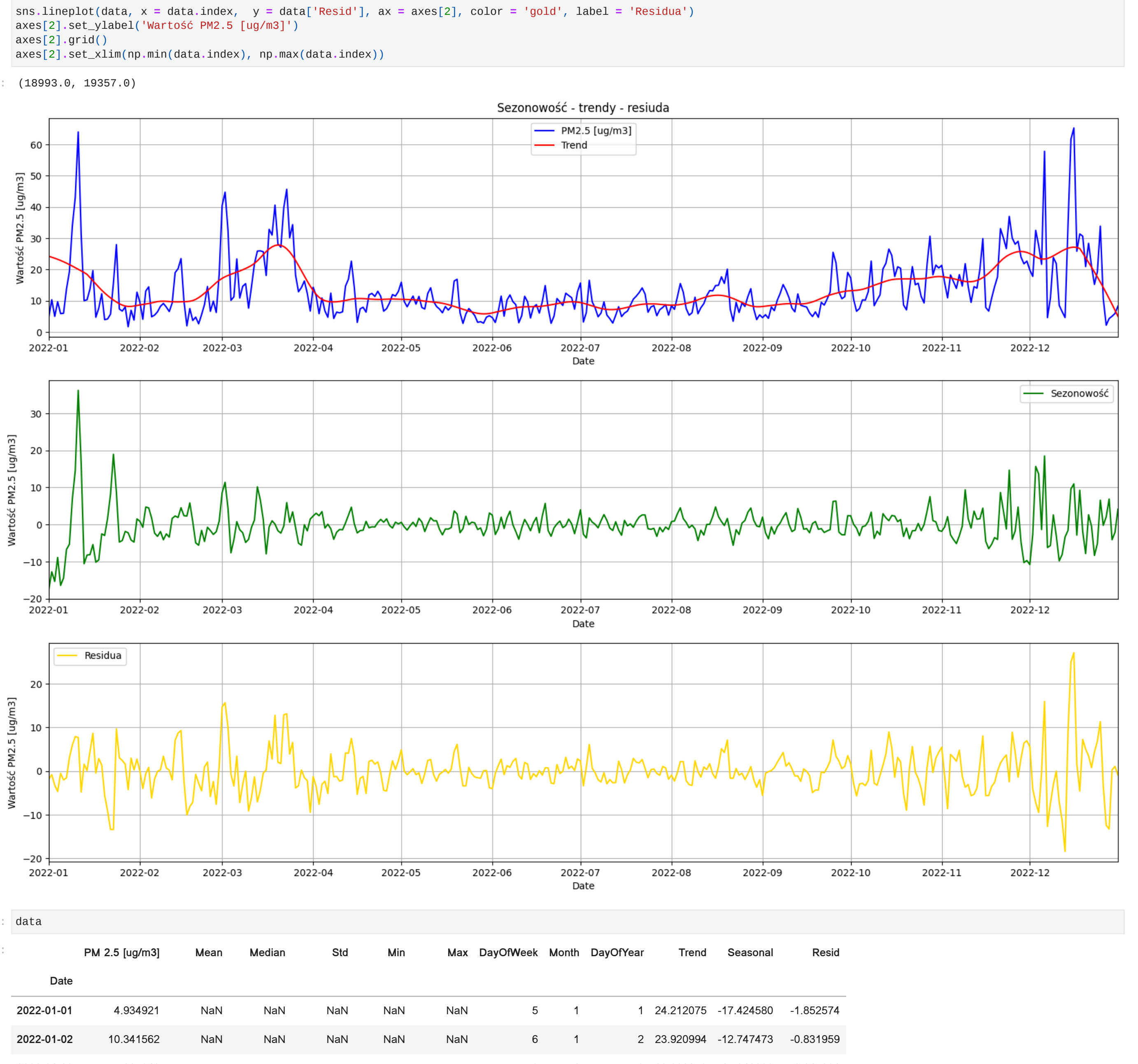
2.5 Wartości wynikające z daty

```
In [17]: data["DayOfWeek"] = data.index.dayofweek
         data["Month"] = data.index.month
         data["DayOfYear"] = data.index.dayofyear
```

2.6 STL

```
In [19]: result = STL(data['PM 2.5 [ug/m3]'], period = 12, seasonal = 5).fit()
         data['Trend'], data['Seasonal'], data['Resid'] = result.trend, result.seasonal, result.resid

         fig, axes = plt.subplots(3, 1, figsize = [20, 14])
         sns.lineplot(data, x = data.index,  y = data['PM 2.5 [ug/m3]'], ax = axes[0], color = 'blue', label = 'PM2.5 [ug/m3]')
         sns.lineplot(data, x = data.index,  y = data['Trend'], ax = axes[0], color = 'red', label = 'Trend')

         axes[0].set_ylabel('Wartość PM2.5 [ug/m3]')
         axes[0].grid()
         axes[0].set_title('Sezonowość - trendy - resiuda')
         axes[0].set_xlim(np.min(data.index), np.max(data.index))

         sns.lineplot(data, x = data.index,  y = data['Seasonal'], ax = axes[1], color = 'green', label = 'Sezonowość')
         axes[1].set_ylabel('Wartość PM2.5 [ug/m3]')
         axes[1].grid()
         axes[1].set_xlim(np.min(data.index), np.max(data.index))

         sns.lineplot(data, x = data.index,  y = data['Resid'], ax = axes[2], color = 'gold', label = 'Residua')
         axes[2].set_ylabel('Wartość PM2.5 [ug/m3]')
         axes[2].grid()
         axes[2].set_xlim(np.min(data.index), np.max(data.index))
```

```
Out[19]: (18993.0, 19357.0)
```



```
In [20]: data
```

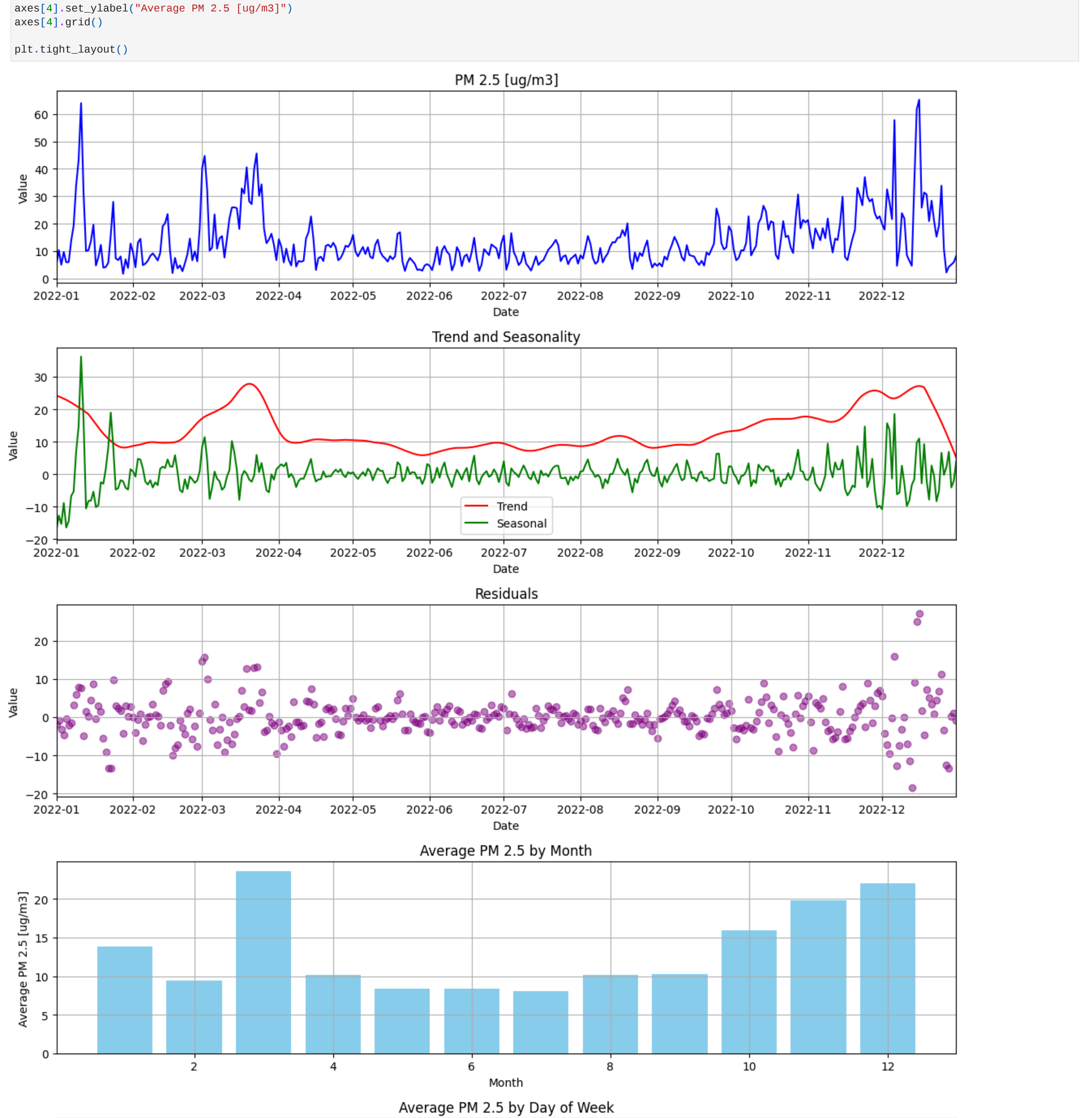
Out[20]:

Date	PM 2.5 [ug/m3]	Mean	Median	Std	Min	Max	DayOfWeek	Month	DayOfYear	Trend	Seasonal	Resid
2022-01-01	4.934921	NaN	NaN	NaN	NaN	NaN	5	1	1	24.212075	-17.424580	-1.852574
2022-01-02	10.341562	NaN	NaN	NaN	NaN	NaN	6	1	2	23.920994	-12.747473	-0.831959
2022-01-03	5.007348	NaN	NaN	NaN	NaN	NaN	0	1	3	23.603979	-15.349300	-3.247331
2022-01-04	9.688094	NaN	NaN	NaN	NaN	NaN	1	1	4	23.258334	-8.882773	-4.687467
2022-01-05	5.914476	NaN	NaN	NaN	NaN	NaN	2	1	5	22.882371	-16.434513	-0.533382
...	...	...	...	...	...	...	...	...	...	...	...	...
2022-12-27	2.195314	28.624122	27.105343	17.358268	2.195314	65.242437	1	12	361	12.521737	2.098572	-12.424995
2022-12-28	4.317930	26.482004	23.458649	18.420872	2.195314	65.242437	2	12	362	10.721940	6.884442	-13.288452
2022-12-29	5.061757	22.437507	21.018141	16.176562	2.195314	65.242437	3	12	363	8.870863	-4.102361	0.293255
2022-12-30	5.950761	18.202387	20.129194	11.060319	2.195314	33.855008	4	12	364	6.975471	-2.080389	1.055679
2022-12-31	8.273009	16.947908	17.271721	11.123749	2.195314	33.855008	5	12	365	5.042661	4.126937	-0.896589

365 rows × 12 columns

3. Przedstaw uzyskany zbiór cech na w przejrzysty sposób na wykresach i opisz co można na ich podstawie zinterpretować.

```
In [22]: import matplotlib.pyplot as plt

         fig, axes = plt.subplots(5, 1, figsize=(12, 16))

         axes[0].plot(data.index, data["PM 2.5 [ug/m3]"], color="blue")
         axes[0].set_title("PM 2.5 [ug/m3]")
         axes[0].set_xlabel("Date")
         axes[0].set_ylabel("Value")
         axes[0].set_xlim(np.min(data.index), np.max(data.index))
         axes[0].grid()

         axes[1].plot(data.index, data["Trend"], color="red", label="Trend")
         axes[1].plot(data.index, data["Seasonal"], color="green", label="Seasonal")
         axes[1].set_title("Trend and Seasonality")
         axes[1].set_xlabel("Date")
         axes[1].set_ylabel("Value")
         axes[1].set_xlim(np.min(data.index), np.max(data.index))
         axes[1].grid()
         axes[1].legend()

         axes[2].scatter(data.index, data["Resid"], color="purple", alpha=0.5)
         axes[2].set_title("Residuals")
         axes[2].set_xlabel("Date")
         axes[2].set_ylabel("Value")
         axes[2].set_xlim(np.min(data.index), np.max(data.index))
         axes[2].grid()

         monthly_avg = data.groupby(data.index.month)["PM 2.5 [ug/m3]"].mean()
         axes[3].bar(monthly_avg.index, monthly_avg, color="skyblue")
         axes[3].set_title("Average PM 2.5 by Month")
         axes[3].set_xlabel("Month")
         axes[3].set_ylabel("Average PM 2.5 [ug/m3]")
         axes[3].grid()

         dayofweek_avg = data.groupby(data.index.dayofweek)["PM 2.5 [ug/m3]"].mean()
         axes[4].bar(dayofweek_avg.index, dayofweek_avg, color="lightgreen")
         axes[4].set_title("Average PM 2.5 by Day of Week")
         axes[4].set_xlabel("Day of Week")
         axes[4].set_ylabel("Average PM 2.5 [ug/m3]")
         axes[4].grid()

         plt.tight_layout()
```



Przedstawione powyżej wykresy zbierają otrzymane informację o cechach stworzonych dla zbioru, dzięki nim widzimy jak rozkładają się wartości PM2.5 na przestrzeni roku. Dodatkowo ukazana jest sezonowść oraz trend. Ostatnie dwa wkresy to rozkład średniej wartości PM2.5 w poszczególnym miesiącu - pokazuję nam to ciekawe wnioski o zmianie średniej w miesiącu - dużych wartościach w okresie marca oraz październik - grudzień oraz wykres średniej wartości pyłków z podziałem na dzien tygodnia - w miarę równomierny rozkład.

```
In [24]: data.to_csv('pm25_fe.csv')
```