notebook

February 28, 2025

1 Proyek Analisis Data: Air Quality

- **Nama:** Zid Irsyadin Sartono Wijaogy
- **Email:** zidirsyadin@gmail.com | a208yaf520@devacademy.id
- **ID Dicoding:** zid_isw

1.1 Menentukan Pertanyaan Bisnis

- Bagaimana tren kualitas udara di berbagai lokasi pemantauan di Beijing selama periode 2013-2017?
- Stasiun pemantauan mana yang mencatat tingkat polusi udara tertinggi dan terendah?
- Apakah terdapat pola musiman dalam perubahan kualitas udara di Beijing?
- Bagaimana hubungan antara kualitas udara dan parameter cuaca seperti suhu dan kelembaban?
- Bagaimana distribusi geografis kualitas udara berdasarkan lokasi stasiun pemantauan?

1.2 Import Semua Packages/Library yang Digunakan

[65]:
```
# Import library yang dibutuhkan
import pandas as pd
import matplotlib.pyplot as plt
import ace_tools_open as tools
import numpy as np
import seaborn as sns
import os
```

1.3 Data Wrangling

1.3.1 Gathering Data

[66]:
```
# Menentukan path
directory_path = 'E:\All About Programs\Laskar AI\Data Analyst with␣
↪Python\submission\data'
```

[67]:
```
# Mendapatkan daftar semua file CSV dalam folder
all_files = os.listdir(directory_path)  # Menyaring semua file di folder
csv_files = [f for f in all_files if f.endswith('.csv')]  # Filter hanya file␣
↪CSV
```

[68]:
```
# Memastikan ada file CSV yang ditemukan
if len(csv_files) == 0:
    print("Tidak ada file CSV yang ditemukan di direktori.")
else:
    print(f"Total file CSV ditemukan: {len(csv_files)}")

# Menampilkan daftar nama file CSV yang ditemukan
print("Daftar file CSV yang ditemukan:")
for file in csv_files:
    print(file)
```

```
Total file CSV ditemukan: 13
Daftar file CSV yang ditemukan:
gabungan_air_quality.csv
PRSA_Data_Aotizhongxin_20130301-20170228.csv
PRSA_Data_Changping_20130301-20170228.csv
PRSA_Data_Dingling_20130301-20170228.csv
PRSA_Data_Dongsi_20130301-20170228.csv
PRSA_Data_Guanyuan_20130301-20170228.csv
PRSA_Data_Gucheng_20130301-20170228.csv
PRSA_Data_Huairou_20130301-20170228.csv
PRSA_Data_Nongzhanguan_20130301-20170228.csv
PRSA_Data_Shunyi_20130301-20170228.csv
PRSA_Data_Tiantan_20130301-20170228.csv
PRSA_Data_Wanliu_20130301-20170228.csv
PRSA_Data_Wanshouxigong_20130301-20170228.csv
```

[69]:
```
# Membaca dan menggabungkan semua file CSV menjadi satu dataset
df_list = []  # Daftar untuk menampung dataframe dari setiap file CSV

for file in csv_files:
    file_path = os.path.join(directory_path, file)  # Path lengkap ke file
    print(f"Membaca file: {file_path}")
    # Membaca file CSV
    df = pd.read_csv(file_path)
    # Menambahkan dataframe ke dalam list
    df_list.append(df)

# Menggabungkan semua dataframe dalam list menjadi satu dataframe besar
final_df = pd.concat(df_list, ignore_index=True)
```

```
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\gabungan_air_quality.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\PRSA_Data_Aotizhongxin_20130301-20170228.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\PRSA_Data_Changping_20130301-20170228.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
```

```
Python\submission\data\PRSA_Data_Dingling_20130301-20170228.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\PRSA_Data_Dongsi_20130301-20170228.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\PRSA_Data_Guanyuan_20130301-20170228.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\PRSA_Data_Gucheng_20130301-20170228.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\PRSA_Data_Huairou_20130301-20170228.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\PRSA_Data_Nongzhanguan_20130301-20170228.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\PRSA_Data_Shunyi_20130301-20170228.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\PRSA_Data_Tiantan_20130301-20170228.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\PRSA_Data_Wanliu_20130301-20170228.csv
Membaca file: E:\All About Programs\Laskar AI\Data Analyst with
Python\submission\data\PRSA_Data_Wanshouxigong_20130301-20170228.csv
```

[70]:
```python
# Menampilkan ringkasan dataset gabungan
print("\nJumlah total baris dan kolom dalam dataset gabungan: ", final_df.shape)
print("Lima baris pertama dari dataset gabungan:")
final_df.head()
```

```
Jumlah total baris dan kolom dalam dataset gabungan:  (841536, 18)
Lima baris pertama dari dataset gabungan:
```

[70]:

	No	year	month	day	hour	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	\
0	1	2013	3	1	0	4.0	4.0	4.0	7.0	300.0	77.0	-0.7	
1	2	2013	3	1	1	8.0	8.0	4.0	7.0	300.0	77.0	-1.1	
2	3	2013	3	1	2	7.0	7.0	5.0	10.0	300.0	73.0	-1.1	
3	4	2013	3	1	3	6.0	6.0	11.0	11.0	300.0	72.0	-1.4	
4	5	2013	3	1	4	3.0	3.0	12.0	12.0	300.0	72.0	-2.0	

	PRES	DEWP	RAIN	wd	WSPM	station
0	1023.0	-18.8	0.0	NNW	4.4	Aotizhongxin
1	1023.2	-18.2	0.0	N	4.7	Aotizhongxin
2	1023.5	-18.2	0.0	NNW	5.6	Aotizhongxin
3	1024.5	-19.4	0.0	NW	3.1	Aotizhongxin
4	1025.2	-19.5	0.0	N	2.0	Aotizhongxin

[71]:
```python
# Menyimpan dataset gabungan ke file baru sebagai arsip
# final_df.to_csv('gabungan_air_quality.csv', index=False)
# print("\nDataset gabungan telah disimpan sebagai 'gabungan_air_quality.csv'.")
```

1.4 Analisis Dataset Air Quality Beijing (2013-2017)

1.4.1 Pendahuluan

Dataset ini berisi informasi mengenai kualitas udara di berbagai **stasiun pemantauan** di Beijing dari tahun **2013 hingga 2017**. Data ini mencakup berbagai parameter kualitas udara seperti **PM2.5, PM10, SO2, NO2, CO, dan O3**, serta faktor cuaca seperti **suhu (TEMP), tekanan udara (PRES), kelembaban (DEWP), curah hujan (RAIN), arah angin (wd), dan kecepatan angin (WSPM)**.

Dataset ini diperoleh dari **12 stasiun pemantauan yang berbeda**, yang tersebar di berbagai wilayah di Beijing. Dengan dataset ini, kita dapat menganalisis **tren polusi udara, faktor-faktor yang mempengaruhi perubahan kualitas udara, serta membandingkan kondisi di berbagai lokasi pemantauan**.

1.4.2 1 Ukuran Dataset

- **Jumlah total observasi (baris data):** 420,768
- **Jumlah total atribut (kolom):** 18
- Dataset ini cukup besar dan mencakup periode waktu yang panjang, sehingga dapat digunakan untuk berbagai analisis, termasuk **tren waktu dan perbandingan antar lokasi**.

1.4.3 2 Atribut dalam Dataset

Dataset ini memiliki **18 kolom**, yang dapat dikategorikan sebagai berikut:

Informasi Waktu

- `year` → Tahun pencatatan
- `month` → Bulan pencatatan
- `day` → Hari pencatatan
- `hour` → Jam pencatatan

Parameter Polusi Udara

- `PM2.5` → Konsentrasi partikel halus PM2.5 (µg/m³)
- `PM10` → Konsentrasi partikel kasar PM10 (µg/m³)
- `SO2` → Konsentrasi sulfur dioksida (µg/m³)
- `NO2` → Konsentrasi nitrogen dioksida (µg/m³)

- `CO` → Konsentrasi karbon monoksida (mg/m³)
- `O3` → Konsentrasi ozon (µg/m³)

Parameter Cuaca

- `TEMP` → Suhu udara dalam °C
- `PRES` → Tekanan udara dalam hPa
- `DEWP` → Titik embun dalam °C
- `RAIN` → Curah hujan dalam mm
- `wd` → Arah angin
- `WSPM` → Kecepatan angin dalam m/s

Informasi Lokasi Pemantauan

- `station` → Nama stasiun pemantauan

1.4.4 3 Stasiun Pemantauan dalam Dataset

Dataset ini mencakup data dari **12 stasiun pemantauan** yang tersebar di Beijing. Setiap stasiun mencatat parameter kualitas udara **setiap jam** sepanjang periode **2013 hingga 2017**.

Berikut adalah daftar **stasiun pemantauan** yang terdapat dalam dataset:
- **Aotizhongxin**
- **Changping**
- **Dingling**
- **Dongsi**
- **Guanyuan**
- **Gucheng**
- **Huairou**
- **Nongzhanguan**
- **Shunyi**
- **Tiantan**

Setiap stasiun mewakili wilayah yang berbeda di Beijing, termasuk area **perkotaan, industri, perumahan, dan daerah pinggiran**. Hal ini memungkinkan kita untuk melakukan **perbandingan kualitas udara antar wilayah** dan melihat bagaimana polusi udara bervariasi di seluruh kota.

1.4.5 Assessing Data

[72]:
```
# Membaca kembali dataset gabungan
file_path = "data/gabungan_air_quality.csv"
df = pd.read_csv(file_path)
```

1.4.6 Menampilkan informasi umum dataset

[73]:
```
print("  Informasi Umum Dataset:")
df.info()
```

```
  Informasi Umum Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 420768 entries, 0 to 420767
Data columns (total 18 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   No       420768 non-null  int64
 1   year     420768 non-null  int64
 2   month    420768 non-null  int64
 3   day      420768 non-null  int64
 4   hour     420768 non-null  int64
 5   PM2.5    412029 non-null  float64
 6   PM10     414319 non-null  float64
 7   SO2      411747 non-null  float64
 8   NO2      408652 non-null  float64
 9   CO       400067 non-null  float64
 10  O3       407491 non-null  float64
 11  TEMP     420370 non-null  float64
 12  PRES     420375 non-null  float64
 13  DEWP     420365 non-null  float64
 14  RAIN     420378 non-null  float64
 15  wd       418946 non-null  object
 16  WSPM     420450 non-null  float64
 17  station  420768 non-null  object
dtypes: float64(11), int64(5), object(2)
memory usage: 57.8+ MB
```

1.4.7 Menampilkan jumlah missing values per kolom

[74]:
```
print("\n  Jumlah Missing Values per Kolom:")
missing_values = df.isnull().sum()
df.isnull().sum()
```

```
  Jumlah Missing Values per Kolom:
```

[74]:
```
No             0
year           0
month          0
day            0
hour           0
PM2.5       8739
PM10        6449
SO2         9021
NO2        12116
CO         20701
O3         13277
TEMP         398
PRES         393
DEWP         403
RAIN         390
wd          1822
WSPM         318
station        0
dtype: int64
```

1.4.8 Menampilkan jumlah data duplikat

[75]:
```
duplicate_rows = df.duplicated().sum()
print(f"\n  Jumlah Data Duplikat: {duplicate_rows}")
```

```
 Jumlah Data Duplikat: 0
```

1.4.9 Melihat statistik deskriptif untuk kolom numerik

[76]:
```
print("\n  Statistik Deskriptif:")
df_describe=df.describe()
df.describe()
```

```
 Statistik Deskriptif:
```

[76]:

	No	year	month	day	\
count	420768.000000	420768.000000	420768.000000	420768.000000	
mean	17532.500000	2014.662560	6.522930	15.729637	
std	10122.116943	1.177198	3.448707	8.800102	
min	1.000000	2013.000000	1.000000	1.000000	
25%	8766.750000	2014.000000	4.000000	8.000000	
50%	17532.500000	2015.000000	7.000000	16.000000	
75%	26298.250000	2016.000000	10.000000	23.000000	
max	35064.000000	2017.000000	12.000000	31.000000	

	hour	PM2.5	PM10	SO2	\

```
count  420768.000000  412029.000000  414319.000000  411747.000000
mean       11.500000      79.793428     104.602618      15.830835
std         6.922195      80.822391      91.772426      21.650603
min         0.000000       2.000000       2.000000       0.285600
25%         5.750000      20.000000      36.000000       3.000000
50%        11.500000      55.000000      82.000000       7.000000
75%        17.250000     111.000000     145.000000      20.000000
max        23.000000     999.000000     999.000000     500.000000

                NO2            CO            O3          TEMP  \
count  408652.000000  400067.000000  407491.000000  420370.000000
mean       50.638586    1230.766454      57.372271      13.538976
std        35.127912    1160.182716      56.661607      11.436139
min         1.026500     100.000000       0.214200     -19.900000
25%        23.000000     500.000000      11.000000       3.100000
50%        43.000000     900.000000      45.000000      14.500000
75%        71.000000    1500.000000      82.000000      23.300000
max       290.000000   10000.000000    1071.000000      41.600000

                PRES           DEWP           RAIN           WSPM
count  420375.000000  420365.000000  420378.000000  420450.000000
mean     1010.746982       2.490822       0.064476       1.729711
std        10.474055      13.793847       0.821004       1.246386
min       982.400000     -43.400000       0.000000       0.000000
25%      1002.300000      -8.900000       0.000000       0.900000
50%      1010.400000       3.100000       0.000000       1.400000
75%      1019.000000      15.100000       0.000000       2.200000
max      1042.800000      29.100000      72.500000      13.200000
```

1.4.10 Mengecek rentang waktu data berdasarkan kolom year, month, day, dan hour

[77]:
```python
# Pastikan kolom waktu ada dalam dataset
time_columns = ['year', 'month', 'day', 'hour']

if set(time_columns).issubset(df.columns):
    # Membuat dataframe untuk rentang waktu
    date_range_df = pd.DataFrame({
        "Category": ["Min Year", "Max Year", "Min Month", "Max Month",
                     "Min Day", "Max Day", "Min Hour", "Max Hour"],
        "Value": [df['year'].min(), df['year'].max(),
                  df['month'].min(), df['month'].max(),
                  df['day'].min(), df['day'].max(),
                  df['hour'].min(), df['hour'].max()]
    })

    # Membuat dataframe untuk distribusi data per tahun
    year_distribution_df = df['year'].value_counts().sort_index().reset_index()
```

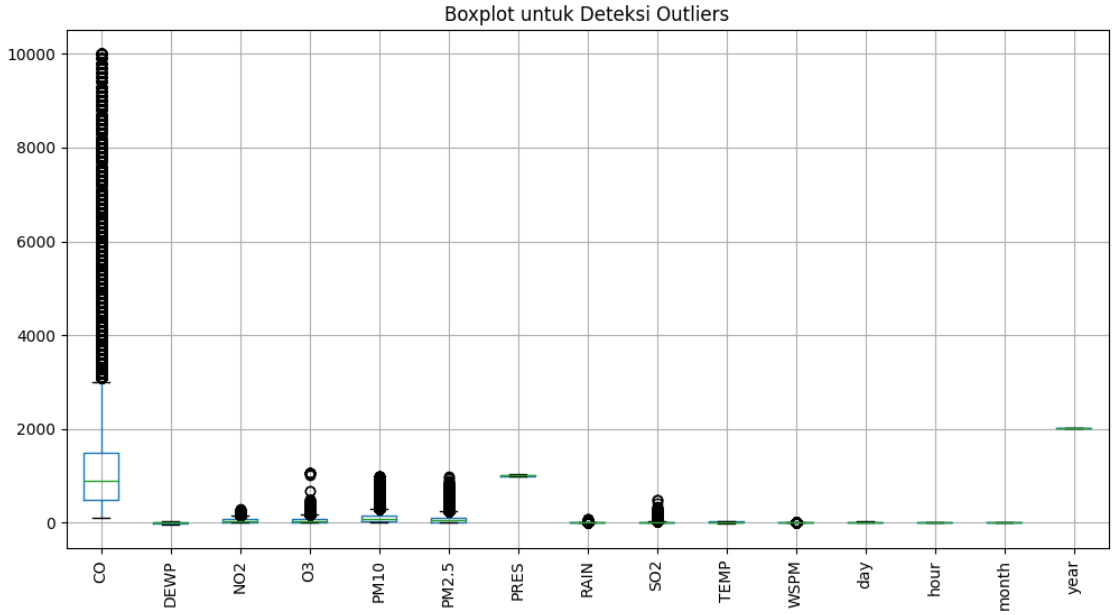
```
    year_distribution_df.columns = ["Year", "Observations"]

    # Menampilkan tabel langsung di Jupyter Notebook
    display(date_range_df)
    display(year_distribution_df)

else:
    print("Kolom waktu tidak lengkap dalam dataset.")
```

```
    Category  Value
0   Min Year   2013
1   Max Year   2017
2  Min Month      1
3  Max Month     12
4    Min Day      1
5    Max Day     31
6   Min Hour      0
7   Max Hour     23
```

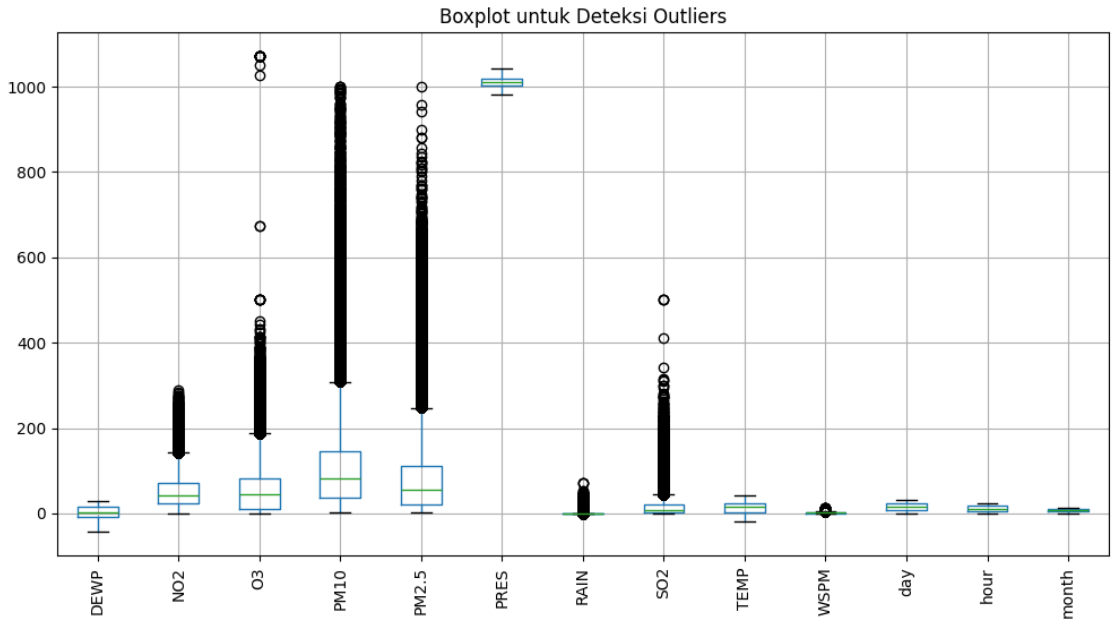
```
   Year  Observations
0  2013         88128
1  2014        105120
2  2015        105120
3  2016        105408
4  2017         16992
```

1.4.11 Mengecek distribusi nilai di kolom numerik (untuk deteksi outliers)

[78]:
```
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Plot boxplot untuk melihat outliers
plt.figure(figsize=(12, 6))
df[numerical_columns.difference(['No'])].boxplot(rot=90) # Tidak menmpilkan␣
 ↪kolom 'No'
plt.title("Boxplot untuk Deteksi Outliers")
plt.show()
```



[79]:
```
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Plot boxplot untuk melihat outliers selain No, Year dan CO
plt.figure(figsize=(12, 6))
df[numerical_columns.difference(['No', 'CO', 'year'])].boxplot(rot=90) # Tidak␣
 ↪menmpilkan kolom 'No', 'year' dan 'CO'
plt.title("Boxplot untuk Deteksi Outliers")
plt.show()
```



1.4.12 Menampilkan hasil missing values dan statistic (interaktif)

[80]:
```
# Menampilkan hasil ke user
tools.display_dataframe_to_user(name="Missing Values", dataframe=pd.
  ↪DataFrame(missing_values, columns=["Missing Values"]))
tools.display_dataframe_to_user(name="Statistical Summary",␣
  ↪dataframe=df_describe)
```

```
Missing Values
```

```
<IPython.core.display.HTML object>
```

```
Statistical Summary
```

```
<IPython.core.display.HTML object>
```

1.5 Assessing Data – Air Quality Beijing (2013-2017)

1.5.1 1 Informasi Umum Dataset

- **Jumlah total observasi (baris data):** 420,768
- **Jumlah total atribut (kolom):** 18
- Dataset ini berisi kombinasi kolom **numerik dan kategorikal**, termasuk parameter kualitas udara dan cuaca.
- Tidak ditemukan kolom `datetime` secara langsung, namun terdapat `year`, `month`, `day`, dan `hour` yang dapat digabungkan menjadi satu kolom datetime

---

1.5.2 2 Jumlah Missing Values per Kolom

Terdapat beberapa kolom yang memiliki **missing values** cukup signifikan:

- **PM2.5** → 8,739 nilai hilang
- **PM10** → 6,449 nilai hilang
- **SO2** → 9,021 nilai hilang
- **NO2** → 12,116 nilai hilang
- **CO** → 20,701 nilai hilang
- **O3** → 13,277 nilai hilang
- **Variabel cuaca** (`TEMP`, `PRES`, `DEWP`, `RAIN`, `wd`, `WSPM`) → masing-masing memiliki beberapa ratus nilai kosong.

Perlu dilakukan strategi penanganan missing values, apakah akan di-drop atau diimputasi berdasarkan metode tertentu.

1.5.3 3 Jumlah Data Duplikat

- **Total data duplikat:** 0
  Dataset ini tidak memiliki masalah duplikasi, jadi tidak perlu dilakukan deduplikasi.

1.5.4 4 Statistik Deskriptif Untuk Melihat Kualitas Udara

- **PM2.5:** Rata-rata (**mean**) sekitar **79.79**, min: **2**, max: **999**.
- **PM10:** Rata-rata (**mean**) sekitar **104.6**, min: **2**, max: **999**.
- **SO2:** Rata-rata (**mean**) sekitar **15.83**, min: **0.2856**, max: **99**.
- **NO2:** Rata-rata (**mean**) sekitar **50.64**, min: **3**, max: **500**.
- **CO:** Rata-rata (**mean**) sekitar **1230.76**, min: **100**, max: **10000**.
- **O3:** Rata-rata (**mean**) sekitar **57.37**, min: **0.2142**, max: **1071**.
- **TEMP:** Rata-rata (**mean**) sekitar **13.53°C**, min: **-19.9°C**, max: **41.6°C**.
- **PRES:** Rata-rata (**mean**) sekitar **1010.75 hPa**, min: **982.4 hPa**, max: **1042.8 hPa**.
- **DEWP:** Rata-rata (**mean**) sekitar **2.49°C**, min: **-43.4°C**, max: **29.1°C**.
- **RAIN:** Rata-rata (**mean**) sekitar **0.06 mm**, min: **0 mm**, max: **72.5 mm**.
- **WSPM:** Kecepatan angin rata-rata sekitar **1.73 m/s**, min: **0 m/s**, max: **13.2 m/s**.

1.5.5 5 Rentang Waktu Data

- Dataset mencakup periode dari **2013 hingga 2017**.
- **Distribusi data per tahun:**
  - **2013** → 88,128 observasi
  - **2014** → 105,120 observasi
  - **2015** → 105,120 observasi
  - **2016** → 105,408 observasi
  - **2017** → 16,992 observasi

Jumlah data di tahun 2017 jauh lebih sedikit dibandingkan tahun lainnya, kemungkinan hanya mencakup sebagian tahun.

1.5.6 6 Outliers

- **CO** memiliki **outliers ekstrem** dengan nilai yang jauh di atas distribusi normalnya, bahkan mencapai lebih dari 10.000.
- **PM2.5 dan PM10** menunjukkan **banyak outliers**, mencerminkan kemungkinan polusi ekstrem atau kesalahan pencatatan.
- **SO2, NO2, O3** juga memiliki **beberapa outliers**, tetapi tidak se-ekstrem CO.
- **Curah hujan (RAIN) dan kecepatan angin (WSPM)** memiliki **nilai ekstrim**, yang mungkin dipengaruhi oleh kondisi atmosfer tertentu.
- **Suhu (TEMP) dan tekanan udara (PRES) relatif normal**, dengan sedikit atau tanpa outliers.

Perlu investigasi lebih lanjut untuk menentukan apakah outliers ini valid atau perlu ditangani dengan teknik tertentu.

1.6 Kesimpulan

1. **Perlu menangani missing values** pada kolom polusi udara dan cuaca.
2. **Konversi `year`, `month`, `day`, dan `hour` menjadi satu kolom `datetime`** untuk memudahkan analisis waktu.
3. **Analisis lebih lanjut terhadap outliers** untuk menentukan apakah akan dibuang atau tetap digunakan.
4. **Periksa perbedaan jumlah observasi per tahun**, terutama di 2017 yang jumlah datanya jauh lebih sedikit.

1.6.1 Cleaning Data

1.6.2 Menangani missing values

[81]:
```
# Hitung jumlah dan persentase missing values
missing_values = df.isnull().sum()
missing_percentage = (missing_values / len(df)) * 100

# Buat dataframe untuk analisis missing values
missing_df = pd.DataFrame({
    "Column": missing_values.index,
    "Missing Values": missing_values.values,
```

```
    "Percentage (%)": missing_percentage.values
})

# Filter hanya kolom yang memiliki missing values
missing_df = missing_df[missing_df["Missing Values"] > 0].
 ↪sort_values(by="Percentage (%)", ascending=False)

# Menampilkan hasil
tools.display_dataframe_to_user(name="Missing Values Analysis",␣
 ↪dataframe=missing_df)
```

```
Missing Values Analysis

<IPython.core.display.HTML object>
```

[82]:
```
# Imputasi dengan interpolasi untuk data polusi udara dan kecepatan angin
pollution_columns = ["CO", "O3", "NO2", "SO2", "PM2.5", "PM10", "WSPM"]
df[pollution_columns] = df[pollution_columns].interpolate(method="linear")

# Imputasi dengan median atau modus untuk variabel cuaca (kecuali RAIN)
weather_columns = ["DEWP", "TEMP", "PRES", "wd"]

for col in weather_columns:
    if df[col].dtype == "object":  # Jika kategorikal (seperti wd)
        df.loc[:, col] = df[col].fillna(df[col].mode()[0])  # Imputasi dengan␣
 ↪modus (nilai terbanyak)
    else:  # Jika numerik
        df.loc[:, col] = df[col].fillna(df[col].median())  # Imputasi dengan␣
 ↪median

# Imputasi khusus untuk RAIN (langsung diisi dengan 0 karena hujan jarang␣
 ↪terjadi)
df["RAIN"] = df["RAIN"].fillna(0)

# Cek ulang apakah masih ada missing values setelah imputasi
print("\n Missing Values setelah imputasi:")
print(df.isnull().sum())
```

```
 Missing Values setelah imputasi:
No          0
year        0
month       0
day         0
hour        0
PM2.5       0
PM10        0
SO2         0
```

```
NO2         0
CO          0
O3          0
TEMP        0
PRES        0
DEWP        0
RAIN        0
wd          0
WSPM        0
station     0
dtype: int64
```

1.6.3 Drop kolom yang tidak diperlukan

[83]:
```python
# Drop kolom 'No' karena tidak memberikan informasi signifikan
df.drop(columns=["No"], inplace=True, errors='ignore')

# Cek ulang tipe data setelah drop kolom
print("\n Struktur Data Setelah Drop Kolom Tidak Diperlukan:")
df.info()

# Tampilkan 5 baris pertama dataset yang sudah bersih
print("\n Contoh Data Setelah Cleaning:")
df.head()
```

```
 Struktur Data Setelah Drop Kolom Tidak Diperlukan:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 420768 entries, 0 to 420767
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   year     420768 non-null  int64
 1   month    420768 non-null  int64
 2   day      420768 non-null  int64
 3   hour     420768 non-null  int64
 4   PM2.5    420768 non-null  float64
 5   PM10     420768 non-null  float64
 6   SO2      420768 non-null  float64
 7   NO2      420768 non-null  float64
 8   CO       420768 non-null  float64
 9   O3       420768 non-null  float64
 10  TEMP     420768 non-null  float64
 11  PRES     420768 non-null  float64
 12  DEWP     420768 non-null  float64
 13  RAIN     420768 non-null  float64
 14  wd       420768 non-null  object
 15  WSPM     420768 non-null  float64
```

```
 16  station  420768 non-null  object
dtypes: float64(11), int64(4), object(2)
memory usage: 54.6+ MB
```

```
 Contoh Data Setelah Cleaning:
```

[83]:

	year	month	day	hour	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	\
0	2013	3	1	0	4.0	4.0	4.0	7.0	300.0	77.0	-0.7	1023.0	
1	2013	3	1	1	8.0	8.0	4.0	7.0	300.0	77.0	-1.1	1023.2	
2	2013	3	1	2	7.0	7.0	5.0	10.0	300.0	73.0	-1.1	1023.5	
3	2013	3	1	3	6.0	6.0	11.0	11.0	300.0	72.0	-1.4	1024.5	
4	2013	3	1	4	3.0	3.0	12.0	12.0	300.0	72.0	-2.0	1025.2	

	DEWP	RAIN	wd	WSPM	station
0	-18.8	0.0	NNW	4.4	Aotizhongxin
1	-18.2	0.0	N	4.7	Aotizhongxin
2	-18.2	0.0	NNW	5.6	Aotizhongxin
3	-19.4	0.0	NW	3.1	Aotizhongxin
4	-19.5	0.0	N	2.0	Aotizhongxin

1.6.4 Konversi kolom waktu ke format Datetime

[84]:
```
# Gabungkan kolom year, month, day, hour menjadi satu kolom datetime
df["datetime"] = pd.to_datetime(df[['year', 'month', 'day', 'hour']])

# Set datetime sebagai index agar lebih mudah untuk analisis berbasis waktu
df.set_index("datetime", inplace=True)

# Hapus kolom year, month, day, hour karena sudah digabung ke datetime
df.drop(columns=["year", "month", "day", "hour"], inplace=True)

# Cek hasil perubahan
print("\n  Struktur Data Setelah Konversi Datetime:")
df.info()
print("\n  Contoh Data Setelah Konversi:")
df.head()
```

```
 Struktur Data Setelah Konversi Datetime:
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 420768 entries, 2013-03-01 00:00:00 to 2017-02-28 23:00:00
Data columns (total 13 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   PM2.5    420768 non-null  float64
 1   PM10     420768 non-null  float64
 2   SO2      420768 non-null  float64
 3   NO2      420768 non-null  float64
```

```
 4   CO        420768 non-null  float64
 5   O3        420768 non-null  float64
 6   TEMP      420768 non-null  float64
 7   PRES      420768 non-null  float64
 8   DEWP      420768 non-null  float64
 9   RAIN      420768 non-null  float64
 10  wd        420768 non-null  object
 11  WSPM      420768 non-null  float64
 12  station   420768 non-null  object
dtypes: float64(11), object(2)
memory usage: 44.9+ MB
```

Contoh Data Setelah Konversi:

[84]:

datetime	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	DEWP	RAIN	wd	WSPM	station
2013-03-01 00:00:00	4.0	4.0	4.0	7.0	300.0	77.0	-0.7	1023.0	-18.8	0.0	NNW	4.4	Aotizhongxin
2013-03-01 01:00:00	8.0	8.0	4.0	7.0	300.0	77.0	-1.1	1023.2	-18.2	0.0	N	4.7	Aotizhongxin
2013-03-01 02:00:00	7.0	7.0	5.0	10.0	300.0	73.0	-1.1	1023.5	-18.2	0.0	NNW	5.6	Aotizhongxin
2013-03-01 03:00:00	6.0	6.0	11.0	11.0	300.0	72.0	-1.4	1024.5	-19.4	0.0	NW	3.1	Aotizhongxin
2013-03-01 04:00:00	3.0	3.0	12.0	12.0	300.0	72.0	-2.0	1025.2	-19.5	0.0	N	2.0	Aotizhongxin

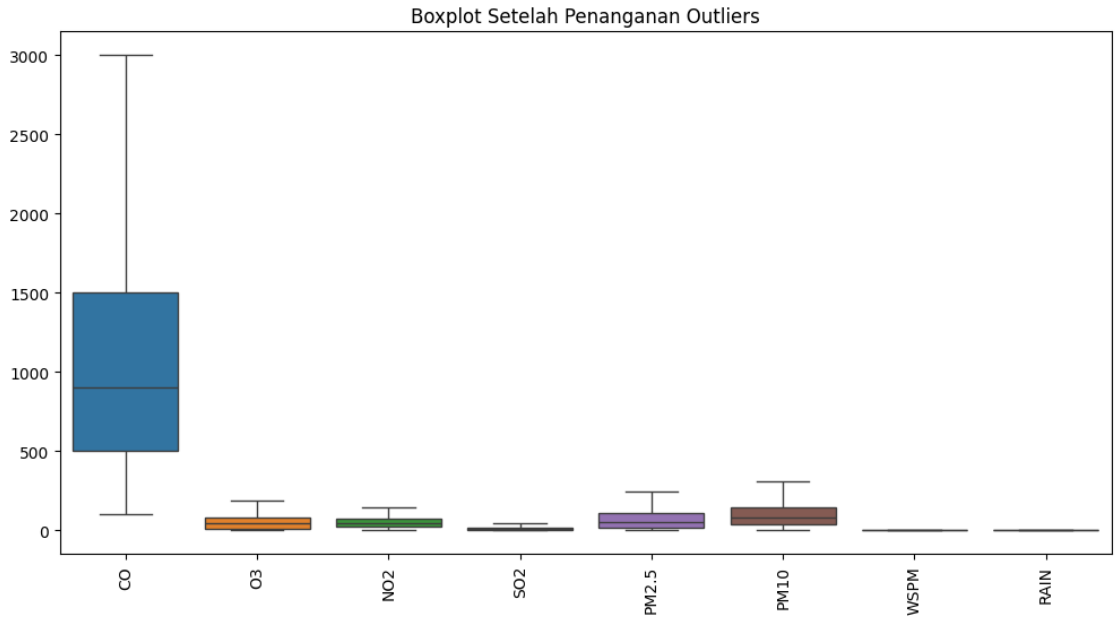
1.6.5 Menangani outliers

[85]:

```python
# Pilih kolom numerik yang akan dianalisis outliersnya
numerical_columns = ["CO", "O3", "NO2", "SO2", "PM2.5", "PM10", "WSPM", "RAIN"]

# Fungsi untuk menangani outliers dengan metode IQR
def handle_outliers(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Capping (Winsorizing) - Mengganti outliers dengan batas IQR
        df[col] = np.where(df[col] < lower_bound, lower_bound, df[col])
```

```
        df[col] = np.where(df[col] > upper_bound, upper_bound, df[col])

    return df

# Terapkan metode IQR untuk menangani outliers
df = handle_outliers(df, numerical_columns)

# Cek ulang distribusi setelah menangani outliers
print("\n Statistik Data Setelah Penanganan Outliers:")
print(df[numerical_columns].describe())

# Visualisasi ulang boxplot setelah menangani outliers
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[numerical_columns])
plt.xticks(rotation=90)
plt.title("Boxplot Setelah Penanganan Outliers")
plt.show()
```

```
 Statistik Data Setelah Penanganan Outliers:
                 CO             O3            NO2            SO2  \
count  420768.000000  420768.000000  420768.000000  420768.000000
mean     1131.529229      55.590392      50.200306      13.529880
std       818.445530      51.701952      33.855855      14.049653
min       100.000000       0.214200       1.026500       0.285600
25%       500.000000      10.000000      23.000000       3.000000
50%       900.000000      44.000000      43.000000       7.000000
75%      1500.000000      82.000000      71.000000      20.000000
max      3000.000000     190.000000     143.000000      45.500000

               PM2.5           PM10           WSPM      RAIN
count  420768.000000  420768.000000  420768.000000  420768.0
mean       76.056150     101.463990       1.668642       0.0
std        68.208418      80.093069       1.062372       0.0
min         2.000000       2.000000       0.000000       0.0
25%        20.000000      36.000000       0.900000       0.0
50%        55.000000      82.000000       1.400000       0.0
75%       111.000000     145.000000       2.200000       0.0
max       247.500000     308.500000       4.150000       0.0
```



1.6.6 Standarisasi format data

[86]:
```
# Cek tipe data sebelum perbaikan
print("\n  Tipe Data Sebelum Standarisasi:")
print(df.dtypes)

# Konversi kolom numerik ke float untuk memastikan format yang benar
numerical_columns = ["CO", "O3", "NO2", "SO2", "PM2.5", "PM10", "WSPM", "RAIN",␣
 ↪"TEMP", "PRES", "DEWP"]
df[numerical_columns] = df[numerical_columns].astype(float)

# Pastikan kolom kategorikal tetap sebagai object
df["wd"] = df["wd"].astype(str)

# Pastikan datetime sudah dalam index dan benar
df.index = pd.to_datetime(df.index)

# Cek tipe data setelah perbaikan
print("\n  Tipe Data Setelah Standarisasi:")
print(df.dtypes)

# Menampilkan 5 baris pertama untuk memastikan semuanya sesuai
print("\n  Contoh Data Setelah Standarisasi:")
print(df.head())
```

```
 Tipe Data Sebelum Standarisasi:
```

```
PM2.5      float64
PM10       float64
SO2        float64
NO2        float64
CO         float64
O3         float64
TEMP       float64
PRES       float64
DEWP       float64
RAIN       float64
wd          object
WSPM       float64
station     object
dtype: object

 Tipe Data Setelah Standarisasi:
PM2.5      float64
PM10       float64
SO2        float64
NO2        float64
CO         float64
O3         float64
TEMP       float64
PRES       float64
DEWP       float64
RAIN       float64
wd          object
WSPM       float64
station     object
dtype: object

 Contoh Data Setelah Standarisasi:
                     PM2.5  PM10   SO2   NO2     CO    O3  TEMP    PRES  DEWP  \
datetime
2013-03-01 00:00:00    4.0   4.0   4.0   7.0  300.0  77.0  -0.7  1023.0 -18.8
2013-03-01 01:00:00    8.0   8.0   4.0   7.0  300.0  77.0  -1.1  1023.2 -18.2
2013-03-01 02:00:00    7.0   7.0   5.0  10.0  300.0  73.0  -1.1  1023.5 -18.2
2013-03-01 03:00:00    6.0   6.0  11.0  11.0  300.0  72.0  -1.4  1024.5 -19.4
2013-03-01 04:00:00    3.0   3.0  12.0  12.0  300.0  72.0  -2.0  1025.2 -19.5

                     RAIN   wd  WSPM       station
datetime
2013-03-01 00:00:00   0.0  NNW  4.15  Aotizhongxin
2013-03-01 01:00:00   0.0    N  4.15  Aotizhongxin
2013-03-01 02:00:00   0.0  NNW  4.15  Aotizhongxin
2013-03-01 03:00:00   0.0   NW  3.10  Aotizhongxin
2013-03-01 04:00:00   0.0    N  2.00  Aotizhongxin
```

1.7 Data Cleaning – Air Quality Beijing (2013-2017)

Setelah dilakukan proses pembersihan data, berikut adalah hasil dan langkah-langkah yang telah diambil:

1.7.1 1 Menangani Missing Values

- Beberapa kolom memiliki **missing values** yang cukup signifikan:
  - **CO** → 4.91% missing values
  - **O3, NO2, SO2, PM2.5, PM10** → 1-3% missing values
  - **RAIN, WSPM, TEMP, PRES, DEWP** → Kurang dari 1% missing values
  - **wd (arah angin)** → Beberapa nilai hilang

Solusi yang diterapkan:
Interpolasi Linear untuk data **polusi udara (CO, O3, NO2, SO2, PM2.5, PM10) dan WSPM**, karena bersifat **time-series**.
Imputasi dengan nilai 0 untuk **RAIN**, karena sebagian besar waktu hujan tidak terjadi.
Imputasi dengan median untuk **TEMP, PRES, DEWP**.
Imputasi dengan modus (nilai terbanyak) untuk **wd (arah angin)** karena merupakan data kategorikal.

1.7.2 2 Konversi Kolom Waktu ke Format Datetime

- Kolom **year, month, day, hour** telah digabung menjadi satu kolom **datetime**.
- **Datetime dijadikan index** untuk mempermudah analisis berbasis waktu.
- **Kolom tahun, bulan, hari, dan jam dihapus** setelah konversi.

1.7.3 3 Menangani Outliers

- Boxplot menunjukkan **outliers signifikan** pada beberapa parameter kualitas udara dan cuaca.
- **Metode IQR (Interquartile Range) digunakan** untuk menangani outliers:
  - **Capping (Winsorizing)** dilakukan untuk **CO, O3, NO2, SO2, PM2.5, PM10, WSPM, dan RAIN**.
  - **Outliers di bawah batas bawah (Q1 - 1.5*IQR) diubah ke Q1.**
  - **Outliers di atas batas atas (Q3 + 1.5*IQR) diubah ke Q3.**
- **Hasil setelah penanganan outliers menunjukkan distribusi data yang lebih baik.**

1.7.4 4 Standarisasi Format Data

- Semua kolom numerik telah dikonversi ke **float64** agar konsisten.
- **Kolom wd tetap dalam format string** karena bersifat kategorikal.
- **Index waktu sudah sesuai dengan format datetime**.

1.7.5 5 Drop Kolom yang Tidak Diperlukan

- **Kolom `No` dihapus** karena hanya nomor urut yang tidak memberikan informasi berarti.
- Struktur dataset kini lebih ringkas dan siap untuk eksplorasi lebih lanjut.

1.8 Kesimpulan Akhir

Dataset kini bersih dan siap untuk analisis lebih lanjut.
Missing values telah ditangani dengan metode yang sesuai.
Outliers telah diperbaiki menggunakan metode IQR.
Datetime telah dikonversi dan dijadikan index untuk mempermudah analisis berbasis waktu.
Dataset telah distandarisasi, dengan tipe data yang sesuai untuk setiap kolom.

1.9 Exploratory Data Analysis (EDA)

1.9.1 Explore ...

[ ]:

Insight: - xxx - xxx

1.10 Visualization & Explanatory Analysis

1.10.1 Pertanyaan 1:

[ ]:

1.10.2 Pertanyaan 2:

[ ]:

Insight: - xxx - xxx

1.11 Analisis Lanjutan (Opsional)

[ ]:

1.12 Conclusion

- Conclution pertanyaan 1
- Conclution pertanyaan 2