EVD - WEEK9

1. Yusuf Eka Maulana (164221020)
2. Fabyan Riza Kiram (164221068)
3. Shiba Salsabilla (164221078)

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from scipy import stats
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [ ]:
```
df = pd.read_csv("london_weather.csv", sep = ",")
df.head()
```

Out[ ]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	prec
0	19790101	2.0	7.0	52.0	2.3	-4.1	-7.5	
1	19790102	6.0	1.7	27.0	1.6	-2.6	-7.5	
2	19790103	5.0	0.0	13.0	1.3	-2.8	-7.2	
3	19790104	8.0	0.0	13.0	-0.3	-2.6	-6.5	
4	19790105	6.0	2.0	29.0	5.6	-0.8	-1.4	

Cek Missing Values

In [ ]:
```
import pandas as pd

# Misalnya, df adalah dataframe Anda
# Gantilah df dengan nama dataframe Anda
jumlah_sampel = df.shape

print(f"Jumlah total sampel: {jumlah_sampel}")
```

```
Jumlah total sampel: (13843, 10)
```

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

Out[ ]:
```
date                  0
cloud_cover          19
sunshine              0
global_radiation     19
max_temp              6
mean_temp            36
min_temp              2
precipitation         6
pressure              4
snow_depth         1441
dtype: int64
```

In [ ]:
```
lst_missval = []

for i in df.isnull().sum():
  lst_missval.append((i / len(df) * 100))

df_missval = pd.DataFrame({'Column Name':df.columns,
                           'Missing Value Percentage (%)':np.round(lst_miss
val,2),
                           'Data Types':df.dtypes})

df_missval = df_missval.sort_values(by='Missing Value Percentage (%)',
                       ascending = False).reset_index().drop(columns = 'ind
ex')

df_missval
```

Out[ ]:

	Column Name	Missing Value Percentage (%)	Data Types
0	snow_depth	9.39	float64
1	mean_temp	0.23	float64
2	cloud_cover	0.12	float64
3	global_radiation	0.12	float64
4	max_temp	0.04	float64
5	precipitation	0.04	float64
6	pressure	0.03	float64
7	min_temp	0.01	float64
8	date	0.00	int64
9	sunshine	0.00	float64

Cek tipe data

In [ ]:
```
df_dtypes = pd.DataFrame({'Columns':df.columns})
lst_nilai = []
for i in df_dtypes['Columns']:
  lst_nilai.append(df[[i]].sample(1).values[[0]])
df_dtypes['Value'] = lst_nilai
df_dtypes['Data Types'] = df.dtypes.values
df_dtypes = df_dtypes.reset_index().drop(columns = ['index'])
df_dtypes
```

Out[ ]:

	Columns	Value	Data Types
0	date	[[19830413]]	int64
1	cloud_cover	[[6.0]]	float64
2	sunshine	[[7.7]]	float64
3	global_radiation	[[317.0]]	float64
4	max_temp	[[2.4]]	float64
5	mean_temp	[[10.4]]	float64
6	min_temp	[[-1.3]]	float64
7	precipitation	[[0.0]]	float64
8	pressure	[[98860.0]]	float64
9	snow_depth	[[0.0]]	float64

In [ ]:
```
# Mengidentifikasi kolom dengan tipe data kategorikal
kategorikal_columns = df.select_dtypes(include=['object']).columns
# Menghapus kolom-kolom kategorikal
df = df.drop(columns=kategorikal_columns)
```

Preprocessing Data

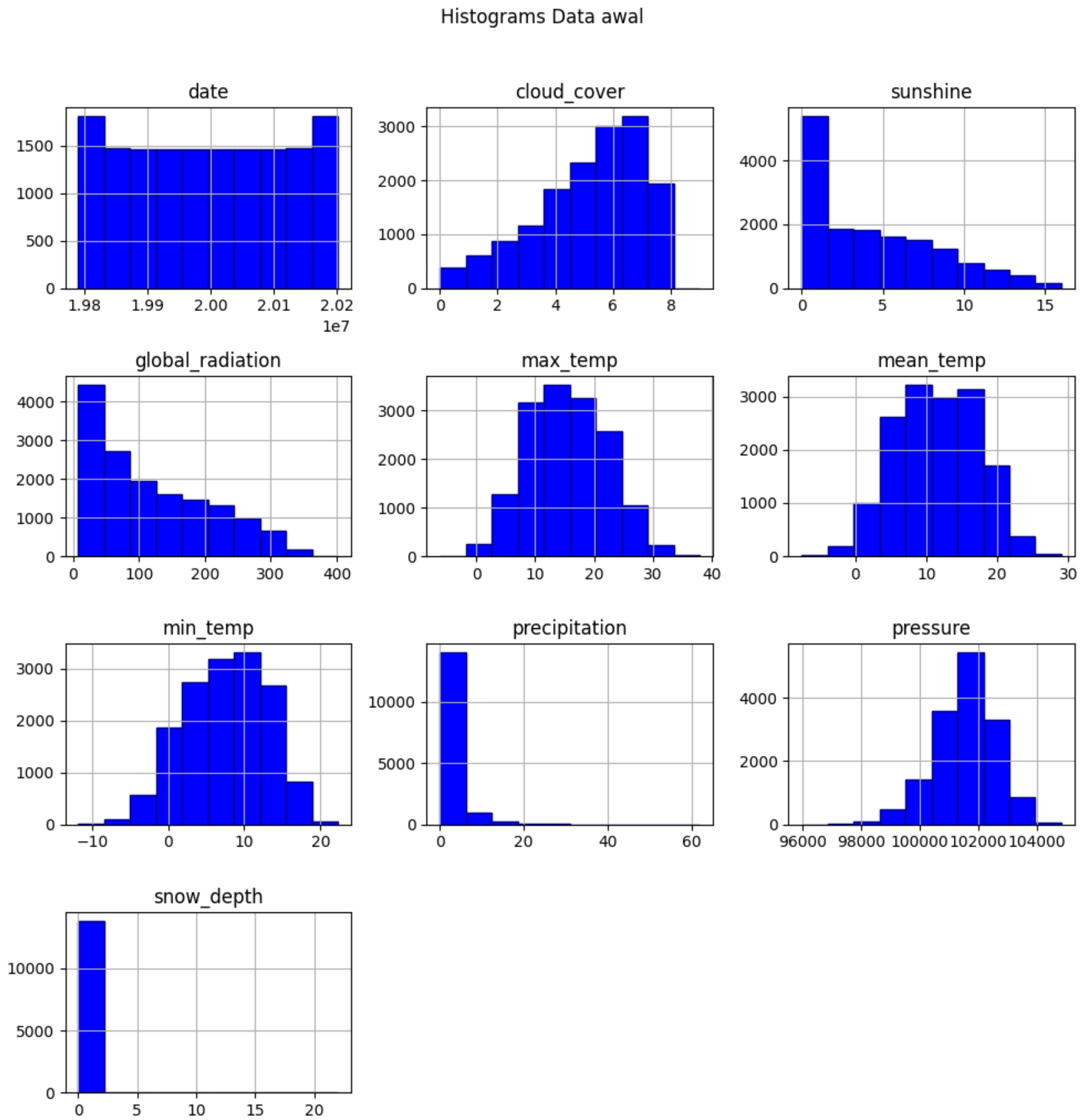
Data awal

In [ ]:
```
df_old = df.copy()
df_old.head()
```

Out[ ]:

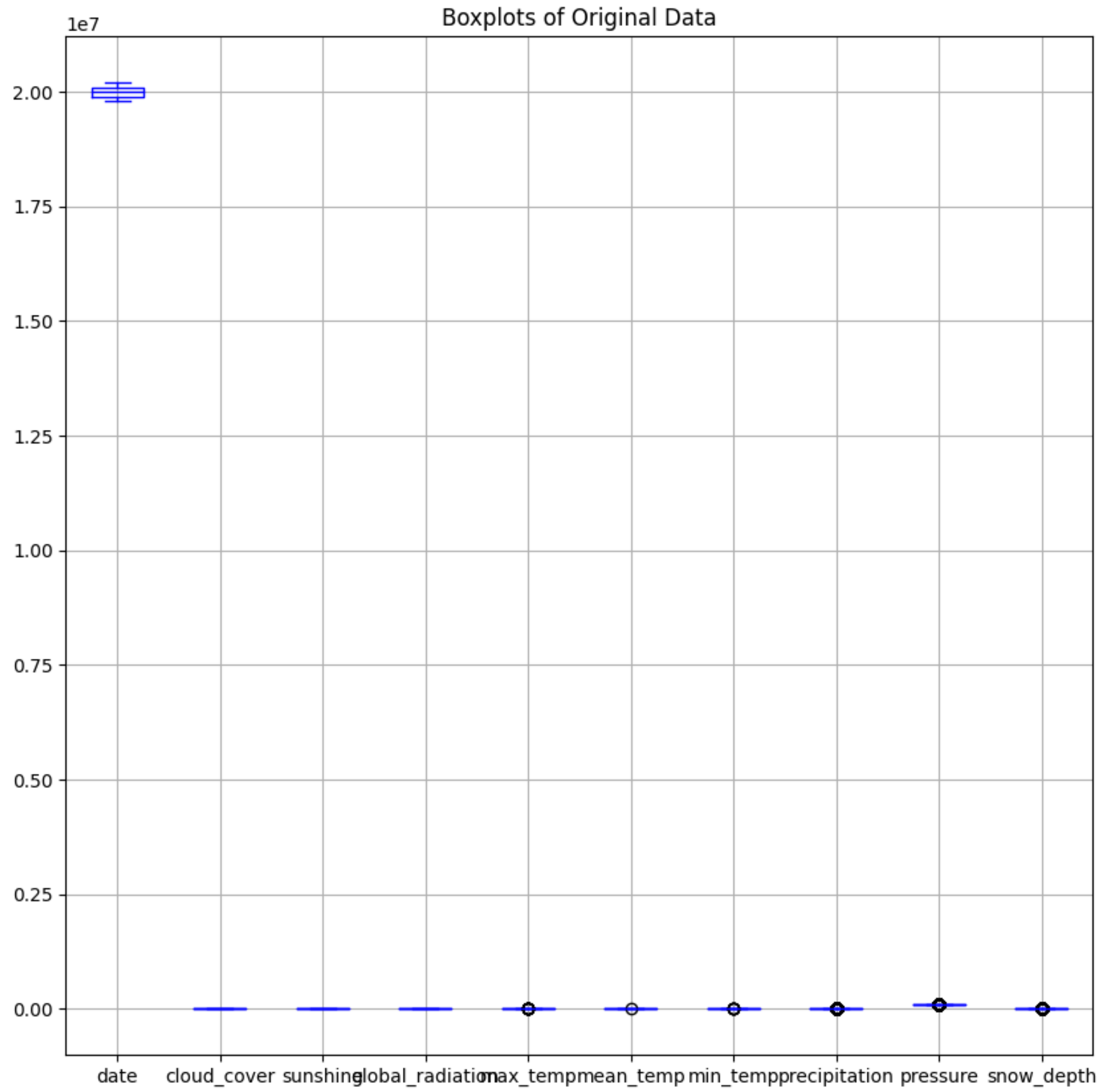
	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	prec
0	19790101	2.0	7.0	52.0	2.3	-4.1	-7.5	
1	19790102	6.0	1.7	27.0	1.6	-2.6	-7.5	
2	19790103	5.0	0.0	13.0	1.3	-2.8	-7.2	
3	19790104	8.0	0.0	13.0	-0.3	-2.6	-6.5	
4	19790105	6.0	2.0	29.0	5.6	-0.8	-1.4	

In [ ]:
```
# Histogram Data awal
df_old.hist(figsize=(10, 10), bins=10, edgecolor='navy', color =
'blue')
plt.suptitle("Histograms Data awal", y=1.02)
plt.tight_layout()
plt.show()
```



In [ ]:
```
# Make a copy of the original dataframe
df_old = df.copy()

# Display the first few rows of the dataframe
print("Head of the DataFrame:")
print(df.head())
print("\nHead of the Old DataFrame:")
print(df_old.head())

# Boxplot of the original data
plt.figure(figsize=(10, 10))
df_old.boxplot(color='blue')
plt.title("Boxplots of Original Data")
plt.show()
```

```
Head of the DataFrame:
       date  cloud_cover  sunshine  global_radiation  max_temp  mean_temp
\
0  19790101          2.0       7.0              52.0       2.3       -4.1
1  19790102          6.0       1.7              27.0       1.6       -2.6
2  19790103          5.0       0.0              13.0       1.3       -2.8
3  19790104          8.0       0.0              13.0      -0.3       -2.6
4  19790105          6.0       2.0              29.0       5.6       -0.8

   min_temp  precipitation  pressure  snow_depth
0      -7.5            0.4  101900.0         9.0
1      -7.5            0.0  102530.0         8.0
2      -7.2            0.0  102050.0         4.0
3      -6.5            0.0  100840.0         2.0
4      -1.4            0.0  102250.0         1.0

Head of the Old DataFrame:
       date  cloud_cover  sunshine  global_radiation  max_temp  mean_temp
\
0  19790101          2.0       7.0              52.0       2.3       -4.1
1  19790102          6.0       1.7              27.0       1.6       -2.6
2  19790103          5.0       0.0              13.0       1.3       -2.8
3  19790104          8.0       0.0              13.0      -0.3       -2.6
4  19790105          6.0       2.0              29.0       5.6       -0.8

   min_temp  precipitation  pressure  snow_depth
0      -7.5            0.4  101900.0         9.0
1      -7.5            0.0  102530.0         8.0
2      -7.2            0.0  102050.0         4.0
3      -6.5            0.0  100840.0         2.0
4      -1.4            0.0  102250.0         1.0
```



Missing value

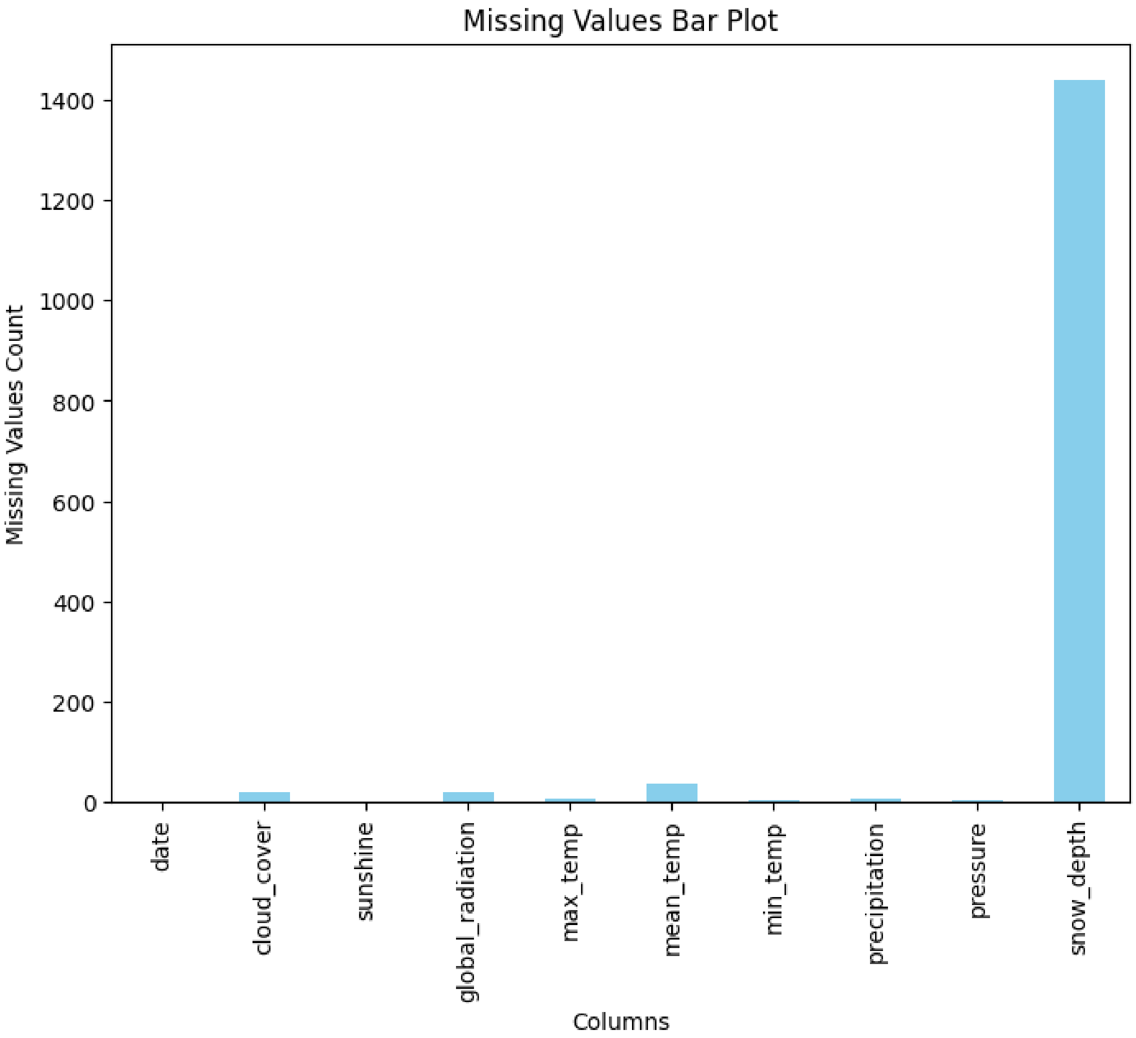
In [ ]:

```
## Cek Missing Value
df.isnull().sum()
```

Out[ ]:

```
date                   0
cloud_cover           19
sunshine               0
global_radiation      19
max_temp               6
mean_temp             36
min_temp               2
precipitation          6
pressure               4
snow_depth          1441
dtype: int64
```

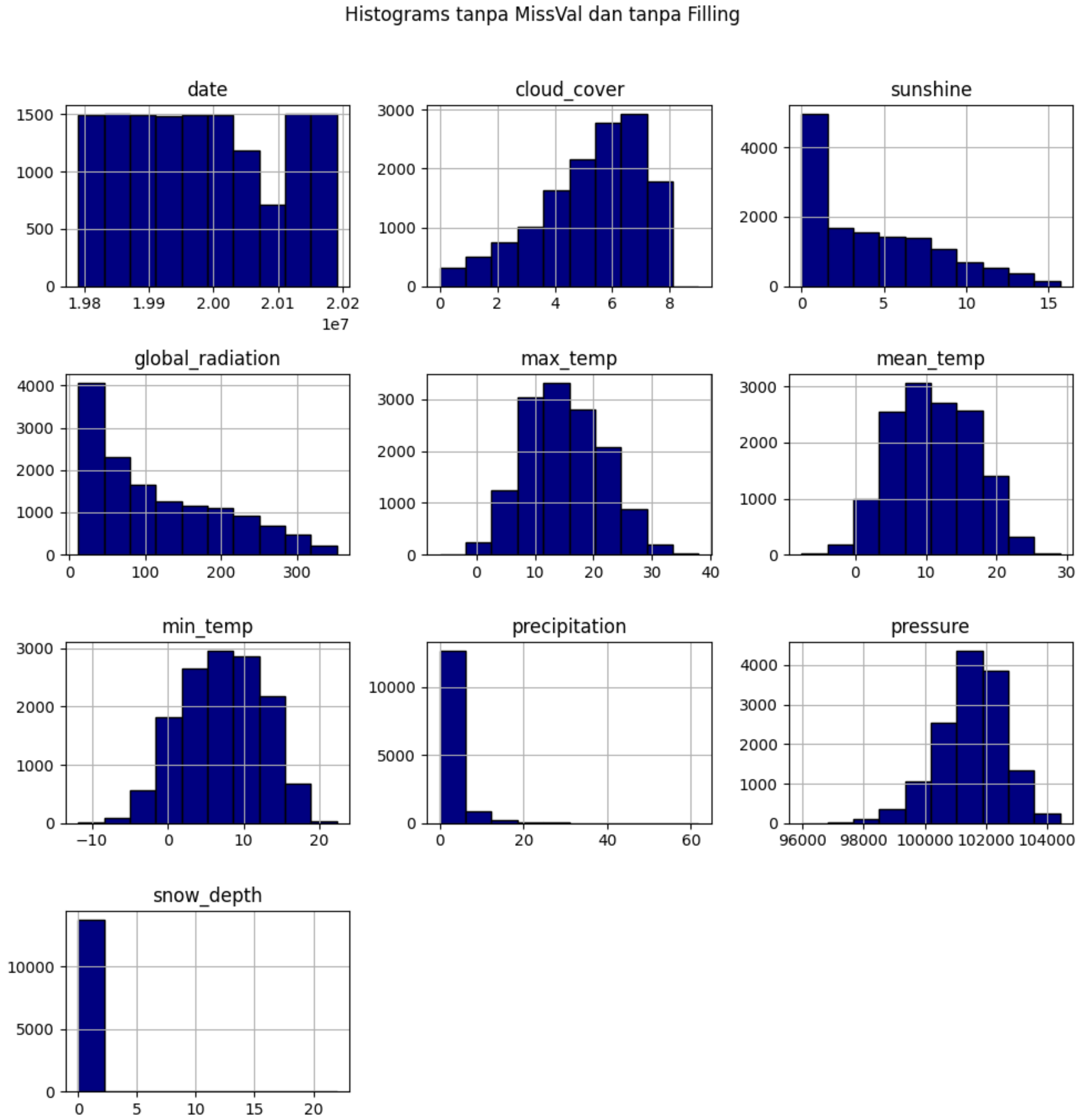
In [ ]:
```
missing_values = df.isnull().sum()

# Membuat bar plot
plt.figure(figsize=(8, 6))
missing_values.plot(kind='bar', color='skyblue')
plt.title('Missing Values Bar Plot')
plt.xlabel('Columns')
plt.ylabel('Missing Values Count')
plt.show()
```



Interpretasi : dari visualisasi bar chart di atas, dapat dilihat bahwa variabel snow_depth memiliki missing value paling besar, yaitu 1441 missing values (9,39%). Selain itu hampir semua variabel memiliki missing values, kecuali variabel date dan sunshine.

In [ ]:
```
# menghapus missing value tanpa filling
df = df.dropna()
```

In [ ]:
```
# missval dan tanpa filling
df.hist(figsize=(10, 10), bins=10, edgecolor='black', color = 'navy')
plt.suptitle("Histograms tanpa MissVal dan tanpa Filling", y=1.02)
plt.tight_layout()
plt.show()
```



Imputasi

- Median

In [ ]:
```
df_median = df_old.fillna(df.median())
df_median.head()
```

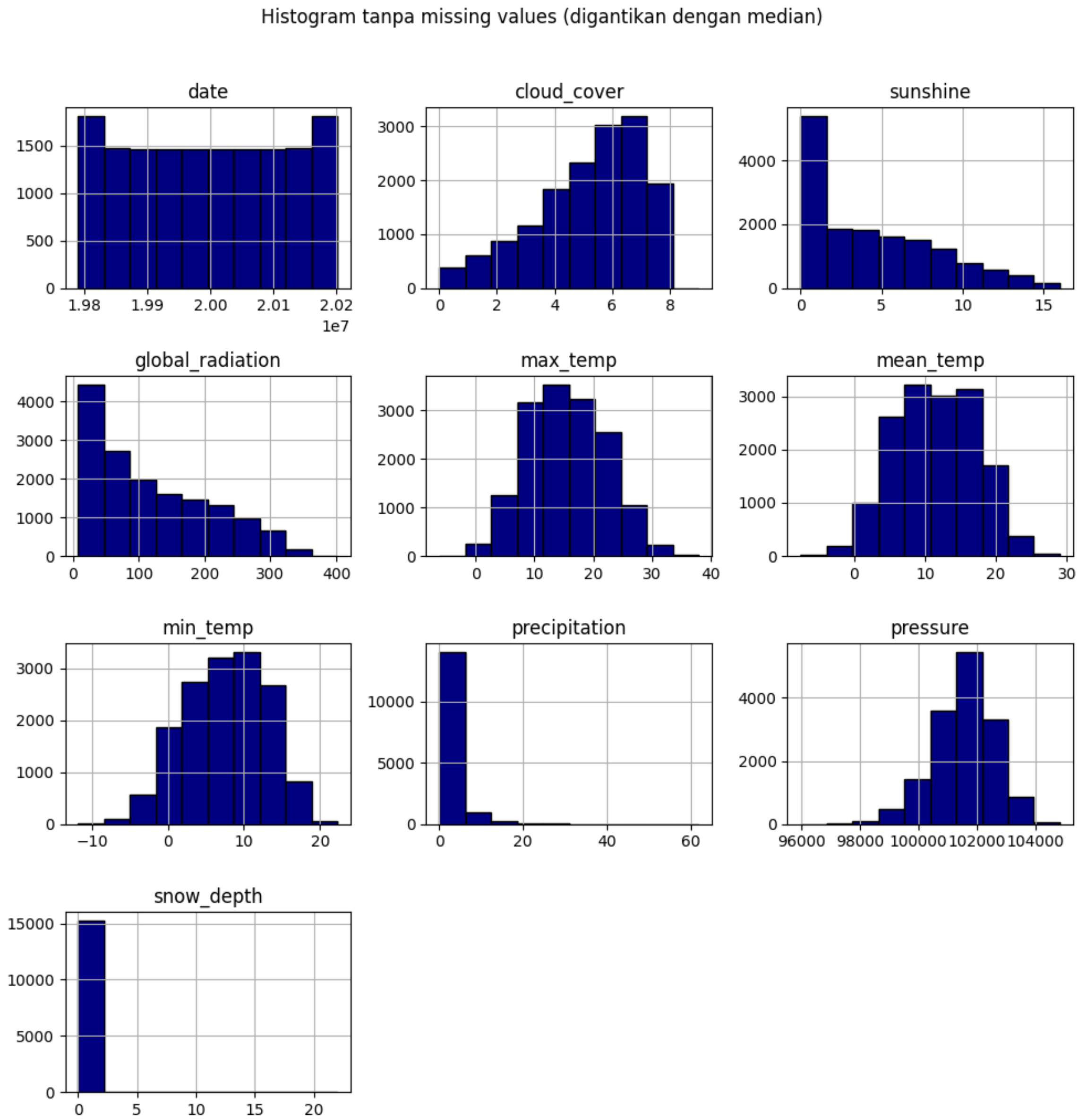
Out[ ]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	prec
0	19790101	2.0	7.0	52.0	2.3	-4.1	-7.5	
1	19790102	6.0	1.7	27.0	1.6	-2.6	-7.5	
2	19790103	5.0	0.0	13.0	1.3	-2.8	-7.2	
3	19790104	8.0	0.0	13.0	-0.3	-2.6	-6.5	
4	19790105	6.0	2.0	29.0	5.6	-0.8	-1.4	

In [ ]:
```
df_median.isnull().sum()
```

Out[ ]:
```
date                0
cloud_cover         0
sunshine            0
global_radiation    0
max_temp            0
mean_temp           0
min_temp            0
precipitation       0
pressure            0
snow_depth          0
dtype: int64
```

In [ ]:
```
# tanpa missing values dengan median
df_median.hist(figsize=(10, 10), bins=10, edgecolor='black', color = 'navy')
plt.suptitle("Histogram tanpa missing values (digantikan dengan median)", y=1.02)
plt.tight_layout()
plt.show()
```



Interpretasi = Membandingkan visualisasi awal dengan setelah dilakukan peng-imputasi dengan median, terlihat bahwa terdapat perubahan pada

- Mean

In [ ]:
```
df_mean = df_old.fillna(df.mean())
df_mean.head()
```

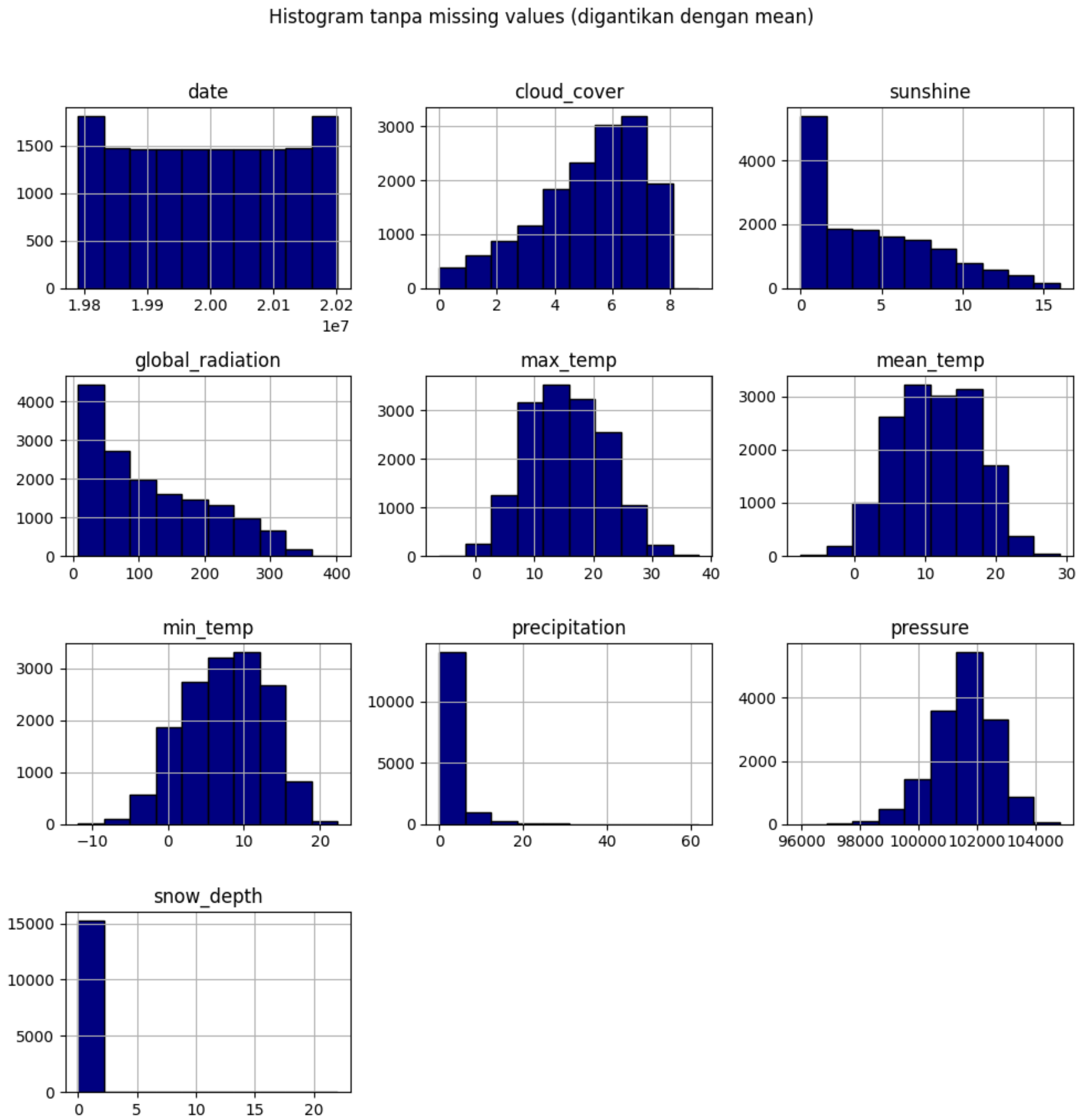
Out[ ]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	prec
0	19790101	2.0	7.0	52.0	2.3	-4.1	-7.5	
1	19790102	6.0	1.7	27.0	1.6	-2.6	-7.5	
2	19790103	5.0	0.0	13.0	1.3	-2.8	-7.2	
3	19790104	8.0	0.0	13.0	-0.3	-2.6	-6.5	
4	19790105	6.0	2.0	29.0	5.6	-0.8	-1.4	

In [ ]:
```
df_median.isnull().sum()
```

Out[ ]:
```
date                0
cloud_cover         0
sunshine            0
global_radiation    0
max_temp            0
mean_temp           0
min_temp            0
precipitation       0
pressure            0
snow_depth          0
dtype: int64
```

In [ ]:
```
# Histograms tanpa missing value dengan mean
df_median.hist(figsize=(10, 10), bins=10, edgecolor='black', color = 'navy')
plt.suptitle("Histogram tanpa missing values (digantikan dengan mean)", y=1.02)
plt.tight_layout()
plt.show()
```

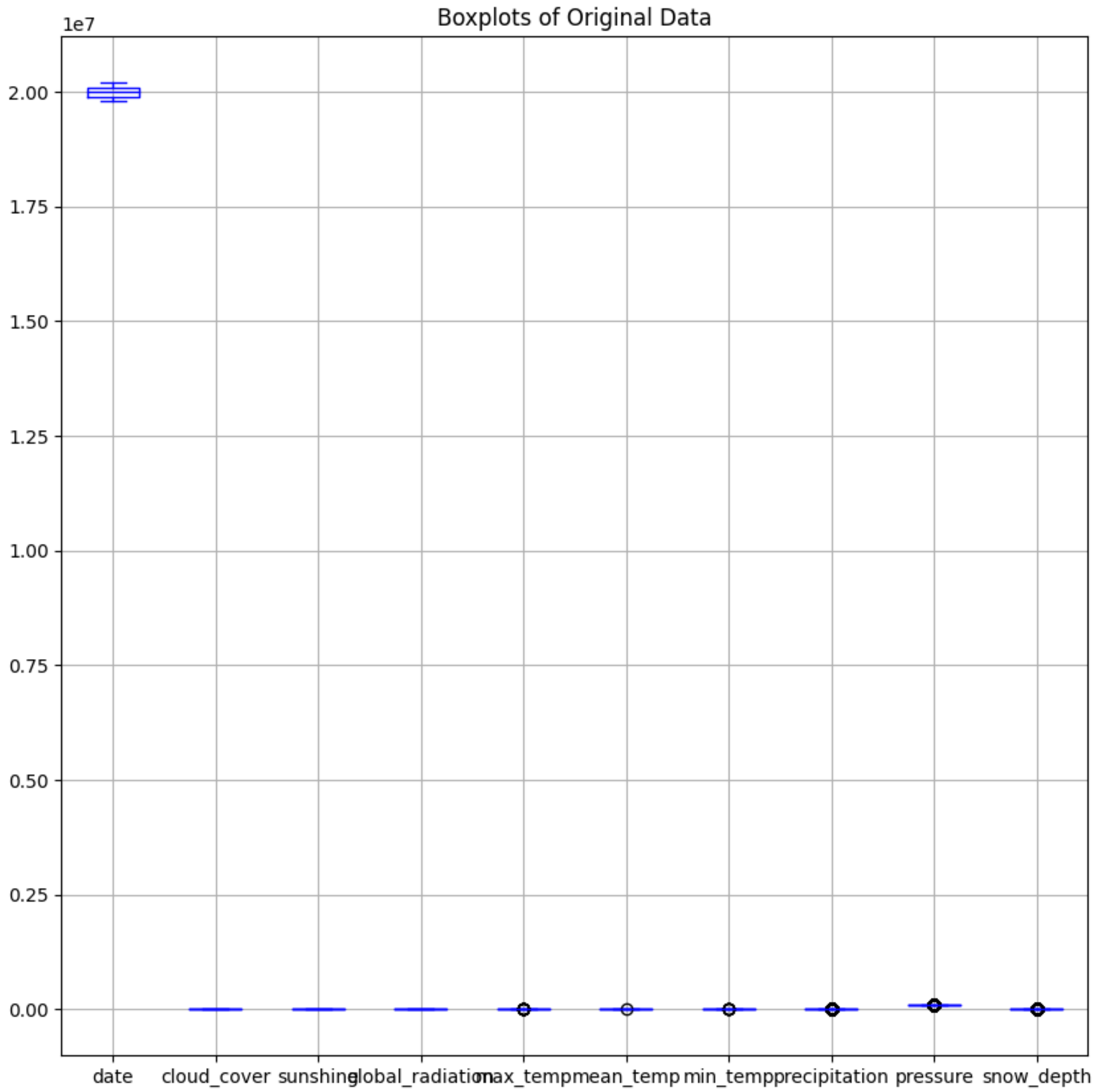


In [ ]:
```
import pandas as pd
import numpy as np

def handle_outliers(dataframe, multiplier=1.5):
    # Loop through each numeric column
    for col in dataframe.columns:
        if pd.api.types.is_numeric_dtype(dataframe[col]):
            # Identify lower and upper bounds
            lower_bound = np.percentile(dataframe[col], 25) - multiplier * (np.percentile(dataframe[col], 75) - np.percentile(dataframe[col], 25))
            upper_bound = np.percentile(dataframe[col], 75) + multiplier * (np.percentile(dataframe[col], 75) - np.percentile(dataframe[col], 25))

            # Handle outliers by replacing them with values close to the lower and upper bounds
            dataframe[col] = np.where(dataframe[col] < lower_bound, lower_bound, dataframe[col])
            dataframe[col] = np.where(dataframe[col] > upper_bound, upper_bound, dataframe[col])

    return dataframe

# Call the function to handle outliers on the DataFrame 'auto1'
df_old1 = handle_outliers(df_old)
print(df_old1.head())
```

```
         date  cloud_cover  sunshine  global_radiation  max_temp  mean_temp  \
0  19790101.0          2.0       7.0              52.0       2.3       -4.1
1  19790102.0          6.0       1.7              27.0       1.6       -2.6
2  19790103.0          5.0       0.0              13.0       1.3       -2.8
3  19790104.0          8.0       0.0              13.0      -0.3       -2.6
4  19790105.0          6.0       2.0              29.0       5.6       -0.8

   min_temp  precipitation  pressure  snow_depth
0      -7.5            0.4  101900.0         9.0
1      -7.5            0.0  102530.0         8.0
2      -7.2            0.0  102050.0         4.0
3      -6.5            0.0  100840.0         2.0
4      -1.4            0.0  102250.0         1.0
```

In [ ]:
```
# Make a copy of the original dataframe

# Display the first few rows of the dataframe
print("Head of the DataFrame:")
print(df_old1.head())
print("\nHead of the Old DataFrame:")
print(df_old1.head())

# Boxplot of the original data
plt.figure(figsize=(10, 10))
df_old1.boxplot(color='blue')
plt.title("Boxplots of Original Data")
plt.show()
```

```
Head of the DataFrame:
         date  cloud_cover  sunshine  global_radiation  max_temp  mean_temp  \
0  19790101.0          2.0       7.0              52.0       2.3       -4.1
1  19790102.0          6.0       1.7              27.0       1.6       -2.6
2  19790103.0          5.0       0.0              13.0       1.3       -2.8
3  19790104.0          8.0       0.0              13.0      -0.3       -2.6
4  19790105.0          6.0       2.0              29.0       5.6       -0.8

   min_temp  precipitation  pressure  snow_depth
0      -7.5            0.4  101900.0         9.0
1      -7.5            0.0  102530.0         8.0
2      -7.2            0.0  102050.0         4.0
3      -6.5            0.0  100840.0         2.0
4      -1.4            0.0  102250.0         1.0

Head of the Old DataFrame:
         date  cloud_cover  sunshine  global_radiation  max_temp  mean_temp  \
0  19790101.0          2.0       7.0              52.0       2.3       -4.1
1  19790102.0          6.0       1.7              27.0       1.6       -2.6
2  19790103.0          5.0       0.0              13.0       1.3       -2.8
3  19790104.0          8.0       0.0              13.0      -0.3       -2.6
4  19790105.0          6.0       2.0              29.0       5.6       -0.8

   min_temp  precipitation  pressure  snow_depth
0      -7.5            0.4  101900.0         9.0
1      -7.5            0.0  102530.0         8.0
2      -7.2            0.0  102050.0         4.0
3      -6.5            0.0  100840.0         2.0
4      -1.4            0.0  102250.0         1.0
```



Normalisasi Data

Min Max Method

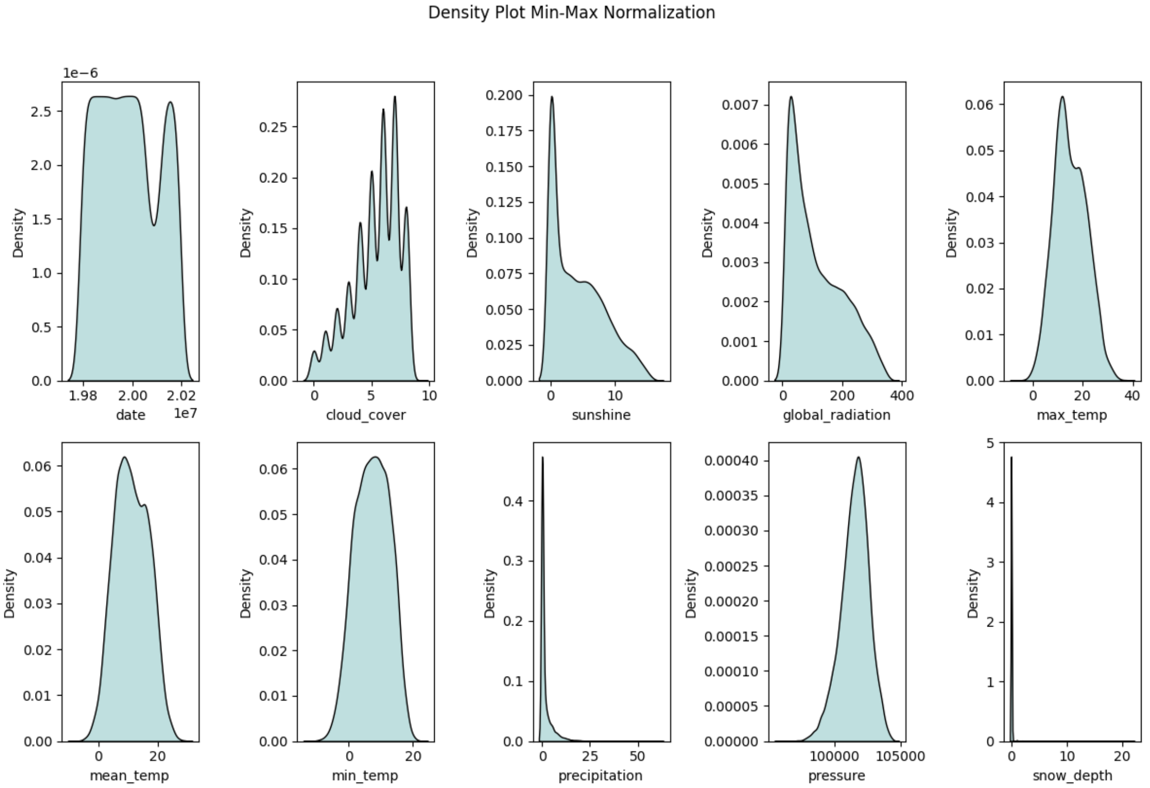
In [ ]:
```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_normalized = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
df_normalized.head()
```

Out[ ]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	prec
0	0.000000	0.222222	0.445860	0.117647	0.192744	0.095628	0.126100	(
1	0.000002	0.666667	0.108280	0.044118	0.176871	0.136612	0.126100	(
2	0.000005	0.555556	0.000000	0.002941	0.170068	0.131148	0.134897	(
3	0.000007	0.888889	0.000000	0.002941	0.133787	0.136612	0.155425	(
4	0.000010	0.666667	0.127389	0.050000	0.267574	0.185792	0.304985	(

Interpretasi : Untuk menyeimbangkan model, maka salah satu yang metode digunakan adalah min - max. Dimana setiap nilai yang ada di tiap variabel nya di buat menjadi skala (antara 0 hingga 1). Dari metode tersebut dapat dilihat bahwa rentang nilai dari tiap tiap variabel tidak terlalu berbeda dan dapat mempermudah interpretasi lain nantinya

In [ ]:
```
plt.figure(figsize=(12, 8))
for i in range(1, 11):
    plt.subplot(2, 5, i)
    sns.kdeplot(df.iloc[:, i-1], shade=True, edgecolor ='black', color
= 'teal')
plt.suptitle("Density Plot Min-Max Normalization", y=1.02)
plt.tight_layout()
plt.show()
```



Z score Method

In [ ]:
```
# Z-score normalization for all continuous variables
scaler = StandardScaler()
df_stdz = pd.DataFrame(scaler.fit_transform(df.iloc[:, :10]), columns=df.co
lumns[:10])

# Display the first 6 rows after Z-score normalization for all continuous v
ariables
print(df_stdz.head())
```

```
       date  cloud_cover  sunshine  global_radiation  max_temp  mean_temp  \
0 -1.615898    -1.636196  0.686520         -0.712543 -1.943463  -2.663765
1 -1.615890     0.330279 -0.642686         -0.997427 -2.050990  -2.400641
2 -1.615882    -0.161340 -1.069035         -1.156962 -2.097073  -2.435724
3 -1.615873     1.313517 -1.069035         -1.156962 -2.342849  -2.400641
4 -1.615865     0.330279 -0.567448         -0.974636 -1.436549  -2.084892

   min_temp  precipitation  pressure  snow_depth
0 -2.765877      -0.339463  0.339110   16.423453
1 -2.765877      -0.446593  0.930079   14.590918
2 -2.709478      -0.446593  0.479817    7.260775
3 -2.577880      -0.446593 -0.655219    3.595704
4 -1.619092      -0.446593  0.667426    1.763169
```

Interpretasi : Setelah menggunakan salah satu metode normalisasi yaitu z-score method, dapat dilihat bahwa beberapa nilai yang ada di variabel 'snow_depth' memiliki rentang yang jauh dengan mean nya. Dan variabel 'pressure' memiliki rentang yang sangat kecil terhadap nilai mean nya

In [ ]:

```
# Density plot for each feature after Z-score normalization
plt.figure(figsize=(12, 8))
for column in df_stdz.columns:
    sns.kdeplot(df_stdz[column], label=column)

plt.title("Density Plots after Z-score Normalization")
plt.xlabel("Standardized Value")
plt.ylabel("Density")
plt.legend()
plt.show()
```