In [1]:
```python
import pandas as pd

file_path = "london_weather[1].csv"

# Read the CSV file into a pandas DataFrame
df = pd.read_csv(file_path)

# Dataframe columns
df.columns
```

Out[1]:
```
Index(['date', 'cloud_cover', 'sunshine', 'global_radiation', 'max_temp',
       'mean_temp', 'min_temp', 'precipitation', 'pressure', 'snow_depth'],
      dtype='object')
```

In [5]:
```python
# Assuming you've already loaded the data into the DataFrame df

# Get information about the data types and non-null counts for each column
data_info = df.info()

# Get summary statistics of numerical columns
summary_stats = df.describe(include='all').round()

# Concatenate the information and summary statistics into a single DataFrame
summary_df = pd.concat([data_info, summary_stats], axis=0)

# Print the summary DataFrame
print(summary_df)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15341 entries, 0 to 15340
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   date              15341 non-null  int64  
 1   cloud_cover       15322 non-null  float64
 2   sunshine          15341 non-null  float64
 3   global_radiation  15322 non-null  float64
 4   max_temp          15335 non-null  float64
 5   mean_temp         15305 non-null  float64
 6   min_temp          15339 non-null  float64
 7   precipitation     15335 non-null  float64
 8   pressure          15337 non-null  float64
 9   snow_depth        13900 non-null  float64
dtypes: float64(9), int64(1)
memory usage: 1.2 MB
```

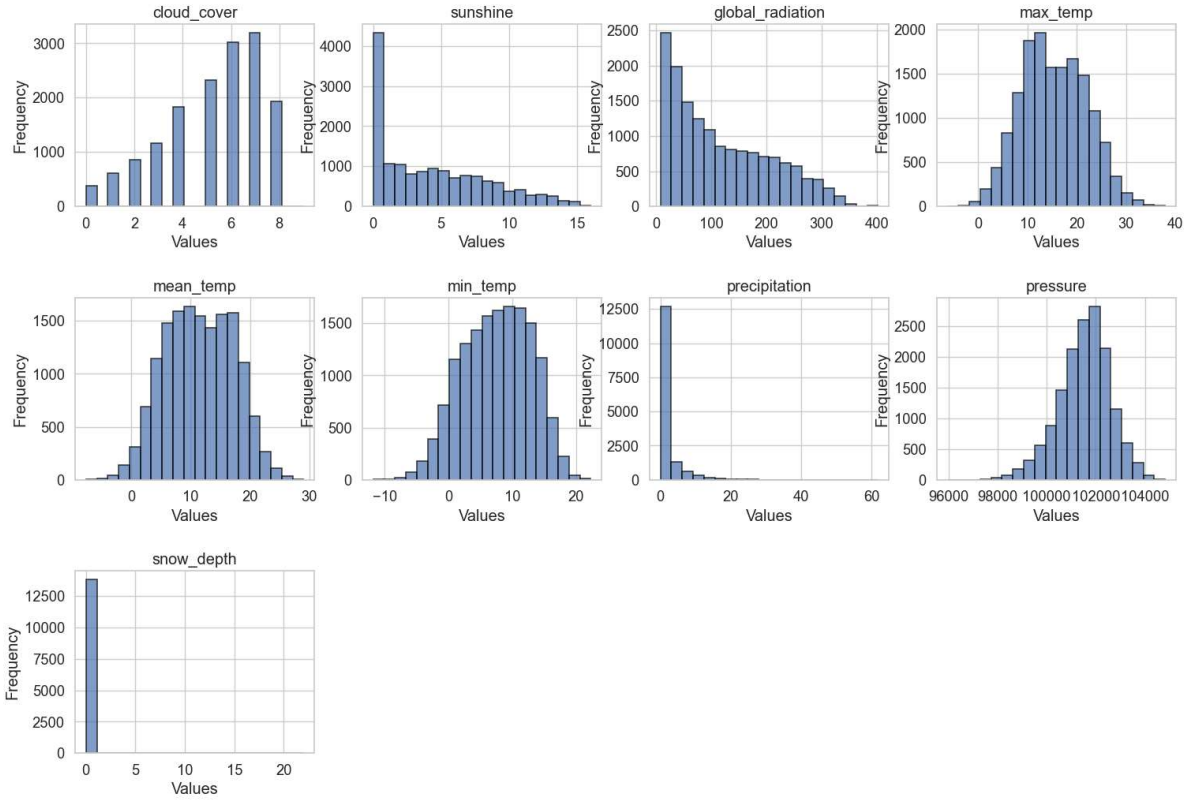
	date	cloud_cover	sunshine	global_radiation	max_temp	\
count	15341.0	15322.0	15341.0	15322.0	15335.0	
mean	19995672.0	5.0	4.0	119.0	15.0	
std	121218.0	2.0	4.0	89.0	7.0	
min	19790101.0	0.0	0.0	8.0	-6.0	
25%	19890702.0	4.0	0.0	41.0	10.0	
50%	20000101.0	6.0	4.0	95.0	15.0	
75%	20100702.0	7.0	7.0	186.0	20.0	
max	20201231.0	9.0	16.0	402.0	38.0	

	mean_temp	min_temp	precipitation	pressure	snow_depth
count	15305.0	15339.0	15335.0	15337.0	13900.0
mean	11.0	8.0	2.0	101537.0	0.0
std	6.0	5.0	4.0	1050.0	1.0
min	-8.0	-12.0	0.0	95960.0	0.0
25%	7.0	4.0	0.0	100920.0	0.0
50%	11.0	8.0	0.0	101620.0	0.0
75%	16.0	12.0	2.0	102240.0	0.0
max	29.0	22.0	62.0	104820.0	22.0

In [9]:

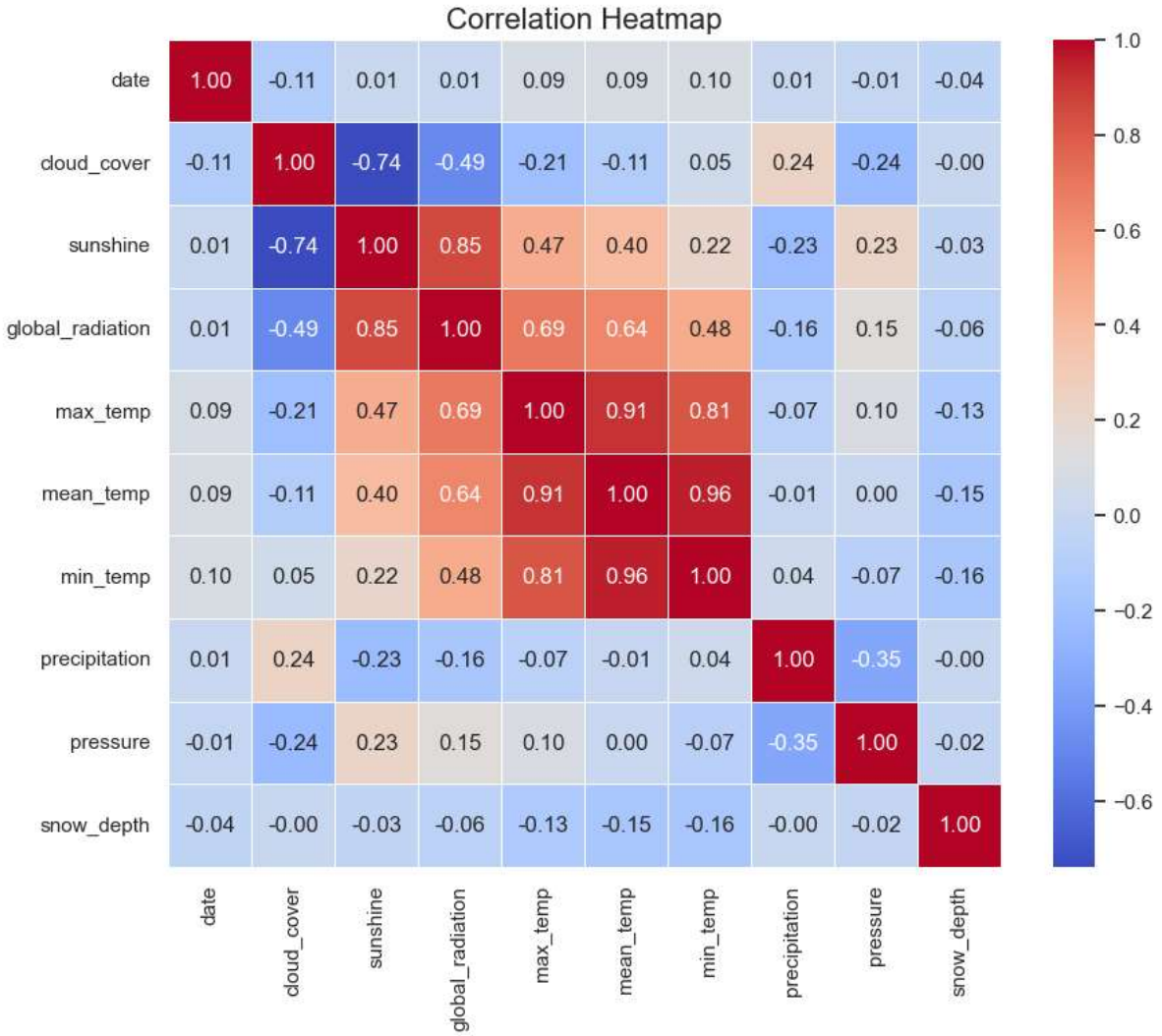
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming you've already loaded the data into the DataFrame df

# Set Seaborn style
sns.set(style="whitegrid", font_scale=1.2)

# Filter out the 'date' column from the DataFrame
numeric_columns = df.drop(columns=['date']).select_dtypes(include=[int, float]).co

# Calculate the number of subplots (excluding 'date' column)
num_plots = len(numeric_columns)

# Determine the number of subplot rows and columns
num_rows = (num_plots - 1) // 4 + 1
num_cols = min(num_plots, 4)

# Create a figure with subplots
fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(18, 12))
fig.subplots_adjust(hspace=0.5)  # Adjust the space between subplots

# Histograms with more professional look
for i, column in enumerate(numeric_columns):
    ax = axes[i // num_cols, i % num_cols]
    df[column].hist(ax=ax, bins=20, edgecolor='black', linewidth=1.2, alpha=0.7)
```

```
    ax.set_title(column)
    ax.set_xlabel('Values')
    ax.set_ylabel('Frequency')

# Remove any empty subplots if the number of plots is not a multiple of 4
if num_plots % 4 != 0:
    for i in range(num_plots % 4, 4):
        fig.delaxes(axes[-1, i])

plt.show()
```



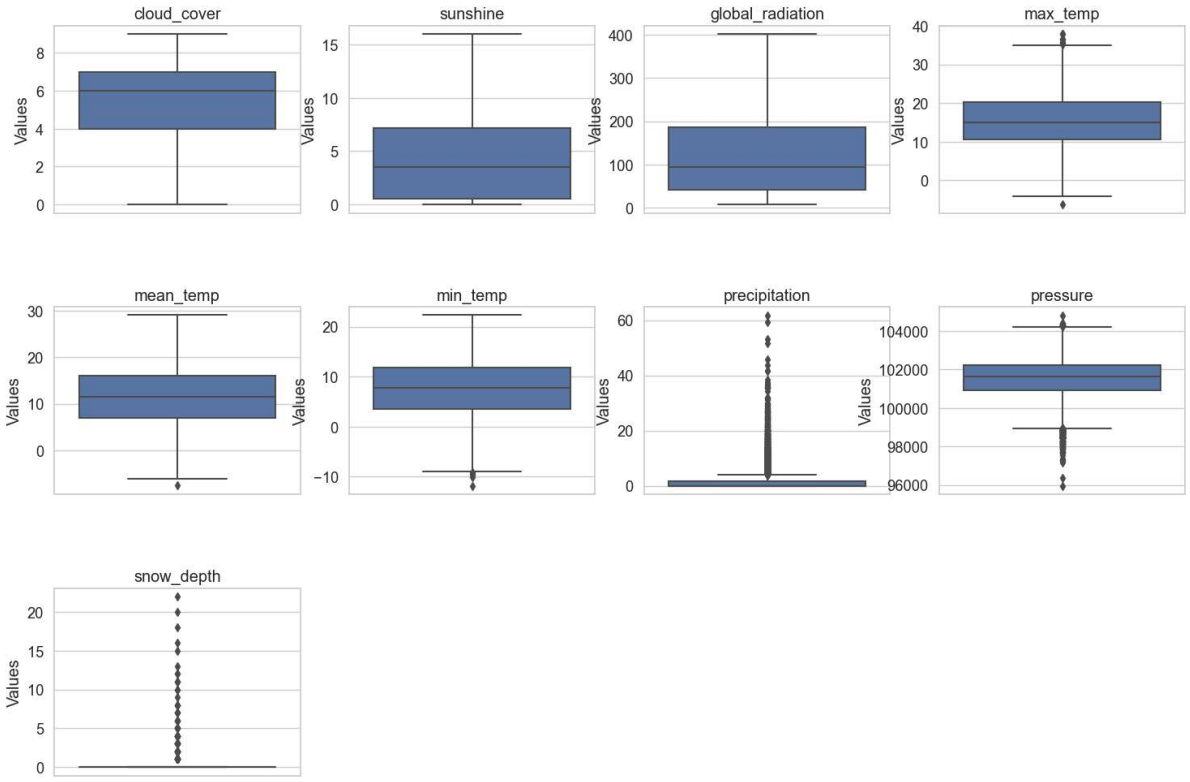
In [10]:

```
# Calculate the correlation matrix
correlation_matrix = df.corr()

# Set Seaborn style
sns.set(style="white")

# Create a heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=
plt.title('Correlation Heatmap', fontsize=16)

plt.show()
```

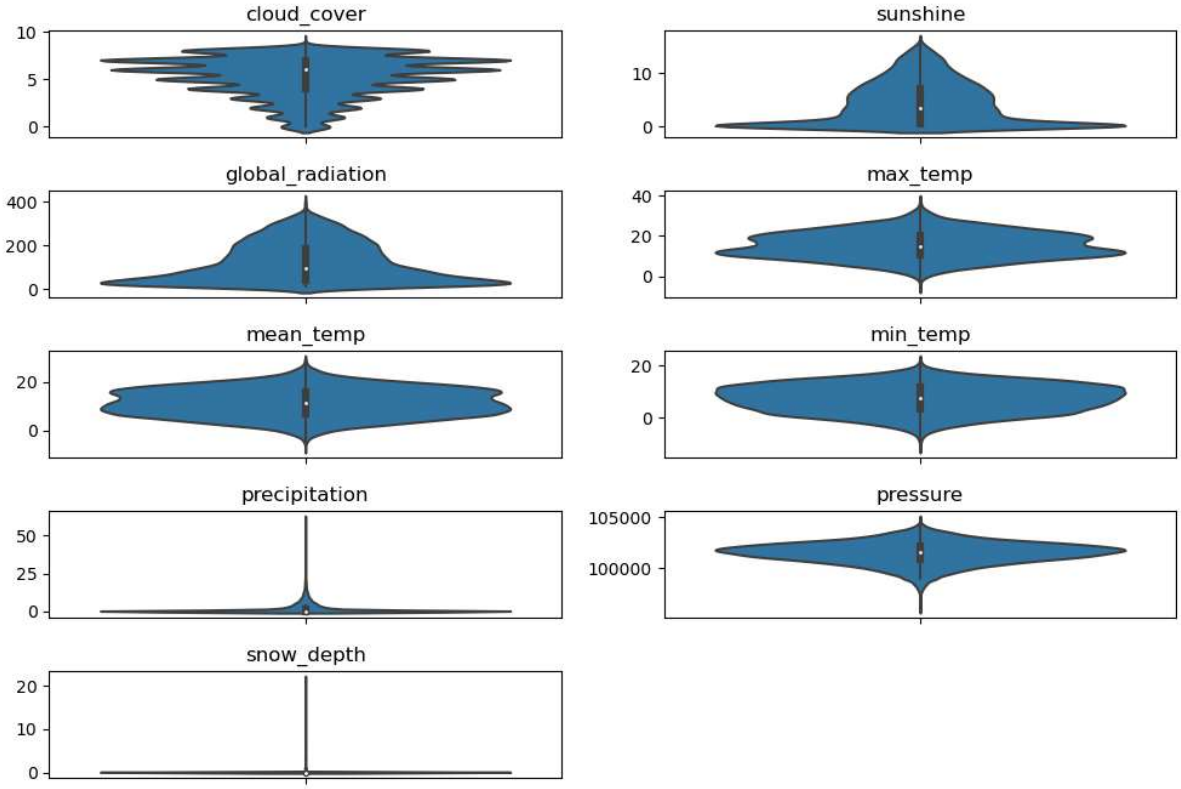


In [11]:

```
# Set Seaborn style
sns.set(style="whitegrid", font_scale=1.2)

# Filter out the 'date' column from the DataFrame
numeric_columns = df.drop(columns=['date']).select_dtypes(include=[int, float]).co

# Calculate the number of subplots (excluding 'date' column)
num_plots = len(numeric_columns)

# Determine the number of subplot rows and columns
num_rows = (num_plots - 1) // 4 + 1
num_cols = min(num_plots, 4)

# Create a figure with subplots
fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(18, 12))
fig.subplots_adjust(hspace=0.5)  # Adjust the space between subplots

# Boxplots with more professional look
for i, column in enumerate(numeric_columns):
    ax = axes[i // num_cols, i % num_cols]
    sns.boxplot(data=df, y=column, ax=ax)
    ax.set_title(column)
    ax.set_ylabel('Values')

# Remove any empty subplots if the number of plots is not a multiple of 4
if num_plots % 4 != 0:
    for i in range(num_plots % 4, 4):
        fig.delaxes(axes[-1, i])

plt.show()
```



In [6]:
```
# Filter out the 'date' column from the DataFrame
numeric_columns = df.drop(columns=['date']).select_dtypes(include=[int, float]).co

# Determine the number of subplot rows and columns
num_plots = len(numeric_columns)
num_rows = (num_plots - 1) // 2 + 1
num_cols = min(num_plots, 2)

# Create a figure with subplots
fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(12, 8))
fig.subplots_adjust(hspace=0.5)  # Adjust the space between subplots

# Violin plots for each numerical column
for i, column in enumerate(numeric_columns):
    ax = axes[i // num_cols, i % num_cols]
    sns.violinplot(data=df, y=column, ax=ax)
    ax.set_title(column)
    ax.set_ylabel('')

# Remove any empty subplots if the number of plots is not a multiple of 2
if num_plots % 2 != 0:
    fig.delaxes(axes[-1, -1])

plt.show()
```


cloud_cover
10
5
0
sunshine
10
0
global_radiation
400
200
0
max_temp
40
20
0
mean_temp
20
0
min_temp
20
0
precipitation
50
25
0
pressure
105000
100000
snow_depth
20
10
0