```
In [1]: #The weather data set is a time series data set with per hour information about a particular location
#its record temperature, dew point,fog,relative,humadity,visibility,wind,speed,pressure and conditions
```

1. Import Libraries and Load the Data

```
In [31]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [4]: # Correct file path with double backslashes
file_path = 'C:\\Users\\Alok\\OneDrive\\文档\\Weather_Data.csv'

# Read the CSV file into the DataFrame
weather = pd.read_csv(file_path)

# Display the first few rows of the data
print(weather.head())
```

```
       Date/Time  Temp_C  Dew Point Temp_C  Rel Hum_%  Wind Speed_km/h  \
0  1/1/2012 0:00    -1.8              -3.9         86                4
1  1/1/2012 1:00    -1.8              -3.7         87                4
2  1/1/2012 2:00    -1.8              -3.4         89                7
3  1/1/2012 3:00    -1.5              -3.2         88                6
4  1/1/2012 4:00    -1.5              -3.3         88                7

   Visibility_km  Press_kPa               Weather
0            8.0     101.24                   Fog
1            8.0     101.24                   Fog
2            4.0     101.26  Freezing Drizzle,Fog
3            4.0     101.27  Freezing Drizzle,Fog
4            4.8     101.23                   Fog
```

Show the Top 5 row from the weather data

```
In [6]: weather.head()
```

Out[6]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
0	1/1/2012 0:00	-1.8	-3.9	86	4	8.0	101.24	Fog
1	1/1/2012 1:00	-1.8	-3.7	87	4	8.0	101.24	Fog
2	1/1/2012 2:00	-1.8	-3.4	89	7	4.0	101.26	Freezing Drizzle,Fog
3	1/1/2012 3:00	-1.5	-3.2	88	6	4.0	101.27	Freezing Drizzle,Fog
4	1/1/2012 4:00	-1.5	-3.3	88	7	4.8	101.23	Fog

Show the below 5 row from the weather data

```
In [8]: weather.tail()
```

Out[8]:

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather
8779	12/31/2012 19:00	0.1	-2.7	81	30	9.7	100.13	Snow
8780	12/31/2012 20:00	0.2	-2.4	83	24	9.7	100.03	Snow
8781	12/31/2012 21:00	-0.5	-1.5	93	28	4.8	99.95	Snow
8782	12/31/2012 22:00	-0.2	-1.8	89	28	9.7	99.91	Snow
8783	12/31/2012 23:00	0.0	-2.1	86	30	11.3	99.89	Snow

2. Preprocessing the Data

```
In [10]: # Convert 'Date/Time' to datetime format
weather['Date/Time'] = pd.to_datetime(weather['Date/Time'], errors='coerce')

# Check for missing values in the dataset
missing_data = weather.isnull().sum()

# Display missing data information
print("Missing Data in Each Column:")
print(missing_data)

# Drop rows with missing values (if any)
weather.dropna(inplace=True)
```

```
Missing Data in Each Column:
Date/Time           0
Temp_C              0
Dew Point Temp_C    0
Rel Hum_%           0
Wind Speed_km/h     0
Visibility_km       0
Press_kPa           0
Weather             0
dtype: int64
```

3. Basic Statistical Analysis

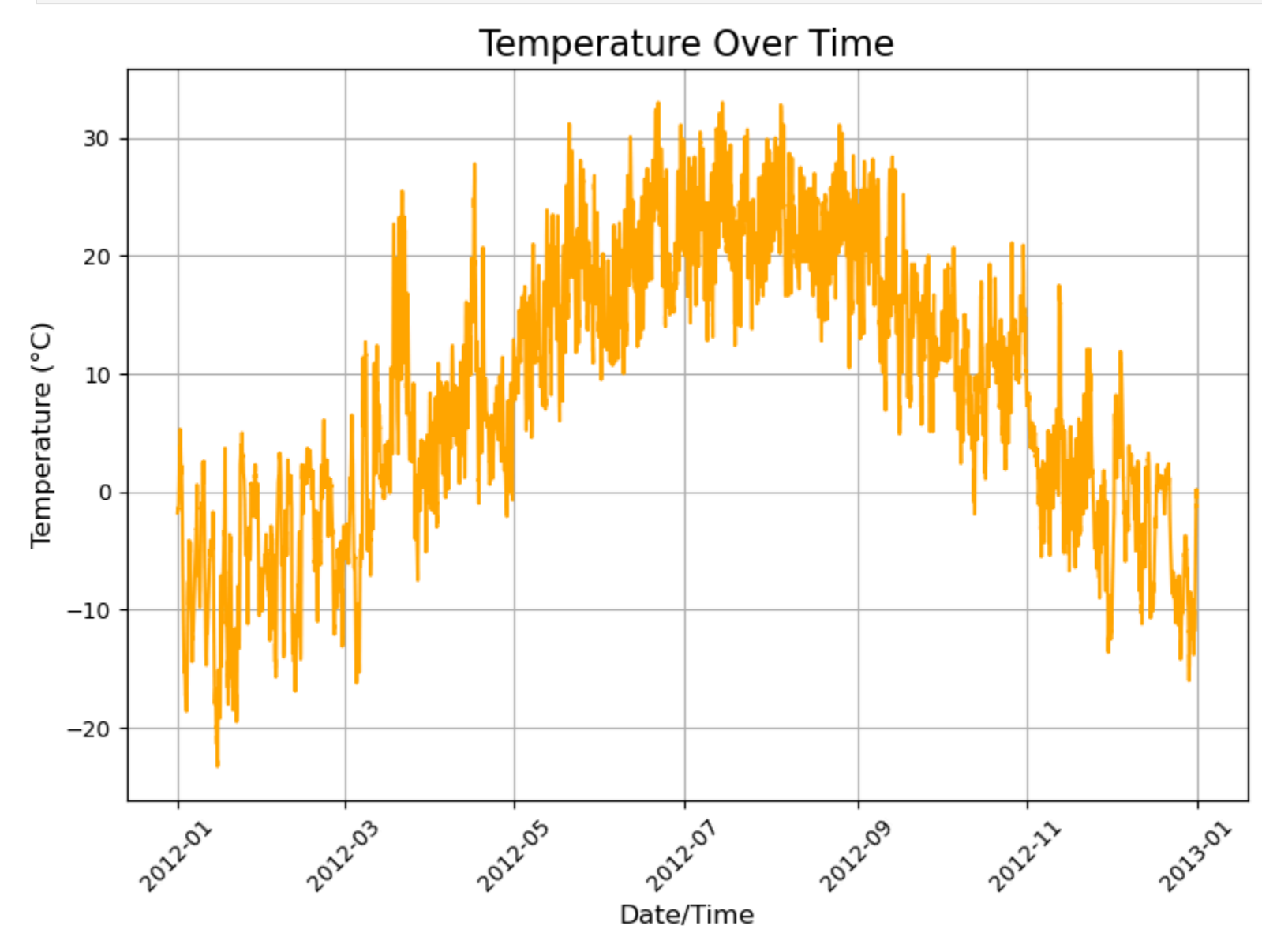
```
In [12]: # Basic statistical summary of the numerical columns
print("Basic Statistical Summary of Weather Data:")
print(weather.describe())
```

```
Basic Statistical Summary of Weather Data:
                 Date/Time       Temp_C  Dew Point Temp_C    Rel Hum_%  \
count                 8784  8784.000000       8784.000000  8784.000000
mean   2012-07-01 23:30:00     8.798144          2.555294    67.431694
min    2012-01-01 00:00:00   -23.300000        -28.500000    18.000000
25%    2012-04-01 11:45:00     0.100000         -5.900000    56.000000
50%    2012-07-01 23:30:00     9.300000          3.300000    68.000000
75%    2012-10-01 11:15:00    18.800000         11.800000    81.000000
max    2012-12-31 23:00:00    33.000000         24.400000   100.000000
std                    NaN    11.687883         10.883072    16.918881

       Wind Speed_km/h  Visibility_km    Press_kPa
count      8784.000000    8784.000000  8784.000000
mean         14.945469      27.664447   101.051623
min           0.000000       0.200000    97.520000
25%           9.000000      24.100000   100.560000
50%          13.000000      25.000000   101.070000
75%          20.000000      25.000000   101.590000
max          83.000000      48.300000   103.650000
std           8.688696      12.622688     0.844005
```

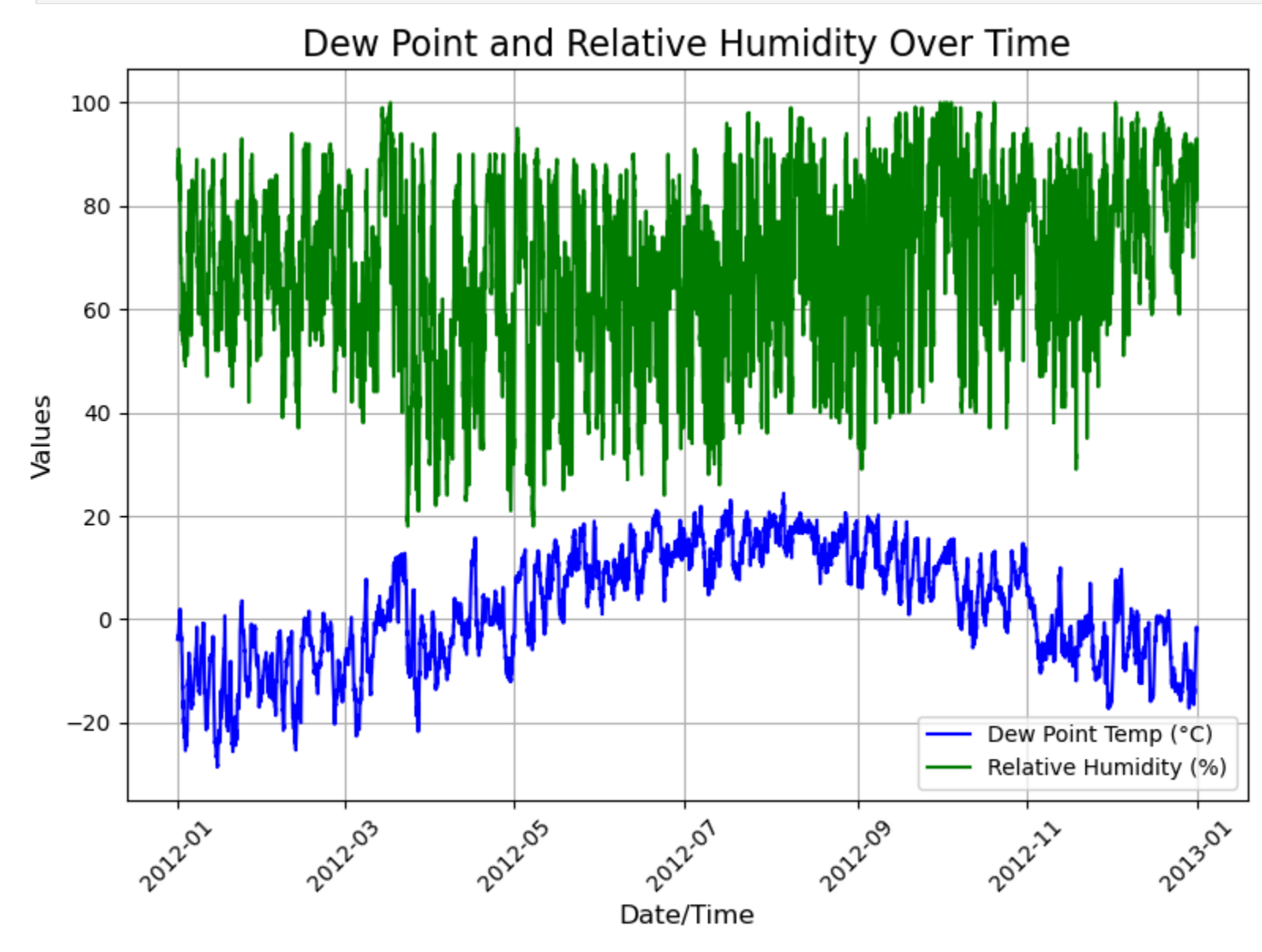
4. Visualization: Temperature over Time

```
In [14]: # Plot temperature over time
plt.figure(figsize=(8, 6))
plt.plot(weather['Date/Time'], weather['Temp_C'], color='orange')
plt.title('Temperature Over Time', fontsize=16)
plt.xlabel('Date/Time', fontsize=12)
plt.ylabel('Temperature (°C)', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



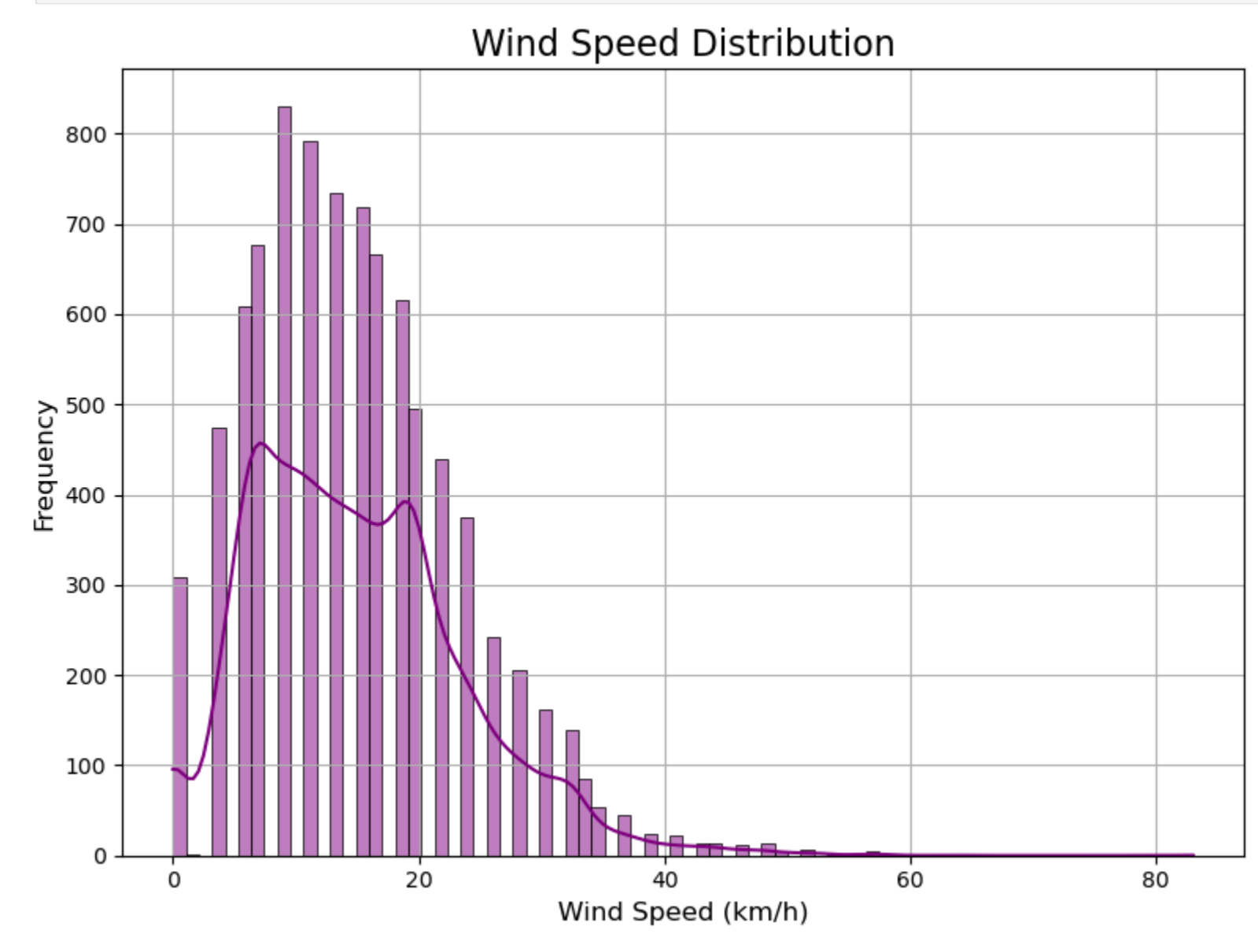
5. Visualization: Dew Point and Relative Humidity

```
In [16]: # Plot Dew Point Temperature and Relative Humidity
plt.figure(figsize=(8, 6))
plt.plot(weather['Date/Time'], weather['Dew Point Temp_C'], label='Dew Point Temp (°C)', color='blue')
plt.plot(weather['Date/Time'], weather['Rel Hum_%'], label='Relative Humidity (%)', color='green')
plt.title('Dew Point and Relative Humidity Over Time', fontsize=16)
plt.xlabel('Date/Time', fontsize=12)
plt.ylabel('Values', fontsize=12)
plt.legend(loc='best')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



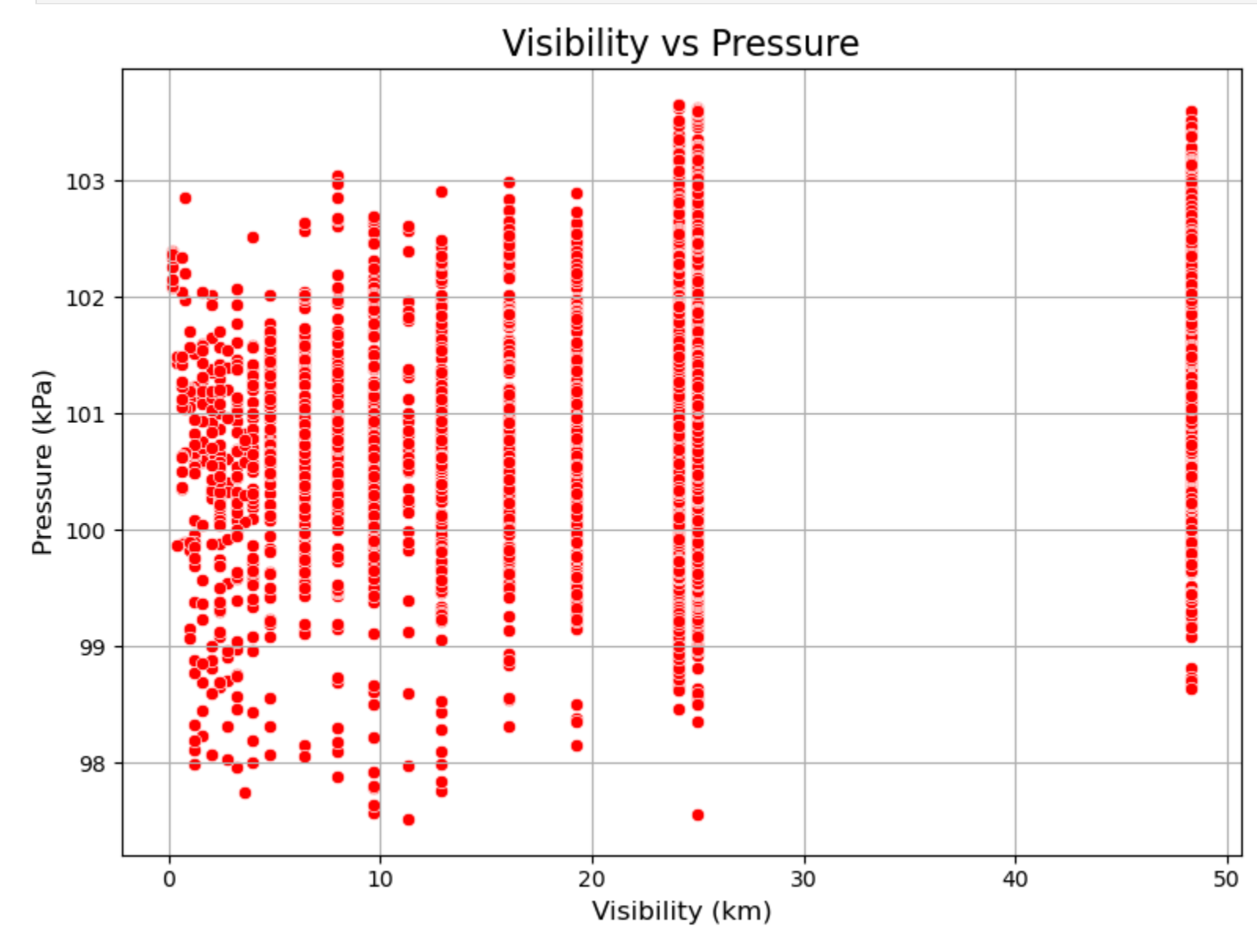
6. Visualization: Wind Speed Distribution

```
In [18]: # Plot Wind Speed distribution
plt.figure(figsize=(8, 6))
sns.histplot(weather['Wind Speed_km/h'], kde=True, color='purple')
plt.title('Wind Speed Distribution', fontsize=16)
plt.xlabel('Wind Speed (km/h)', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()
```



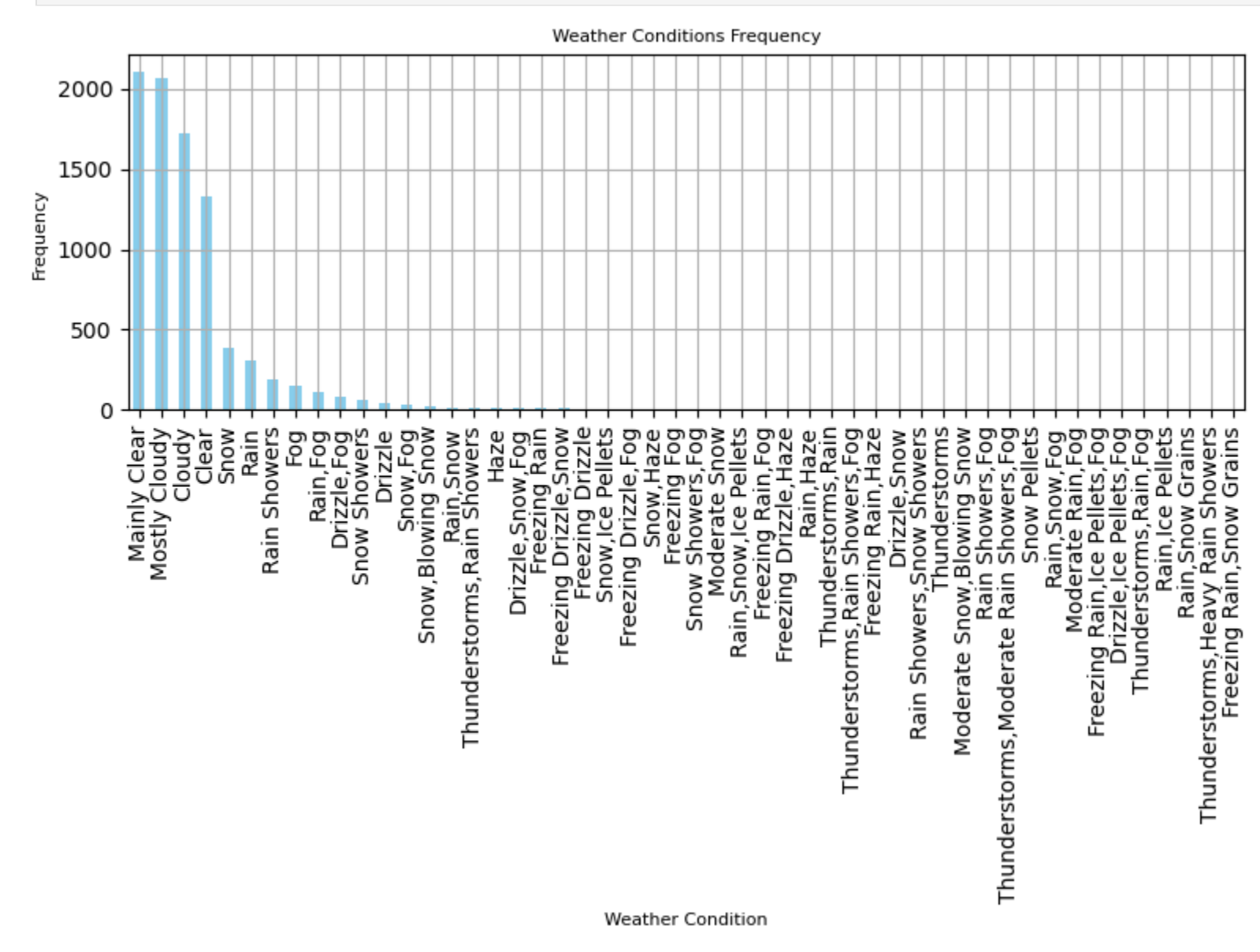
7. Visualization: Visibility vs Pressure

```
In [20]: # Scatter plot for Visibility vs Pressure
plt.figure(figsize=(8, 6))
sns.scatterplot(x=weather['Visibility_km'], y=weather['Press_kPa'], color='red')
plt.title('Visibility vs Pressure', fontsize=16)
plt.xlabel('Visibility (km)', fontsize=12)
plt.ylabel('Pressure (kPa)', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()
```



Visualization: Weather Conditions

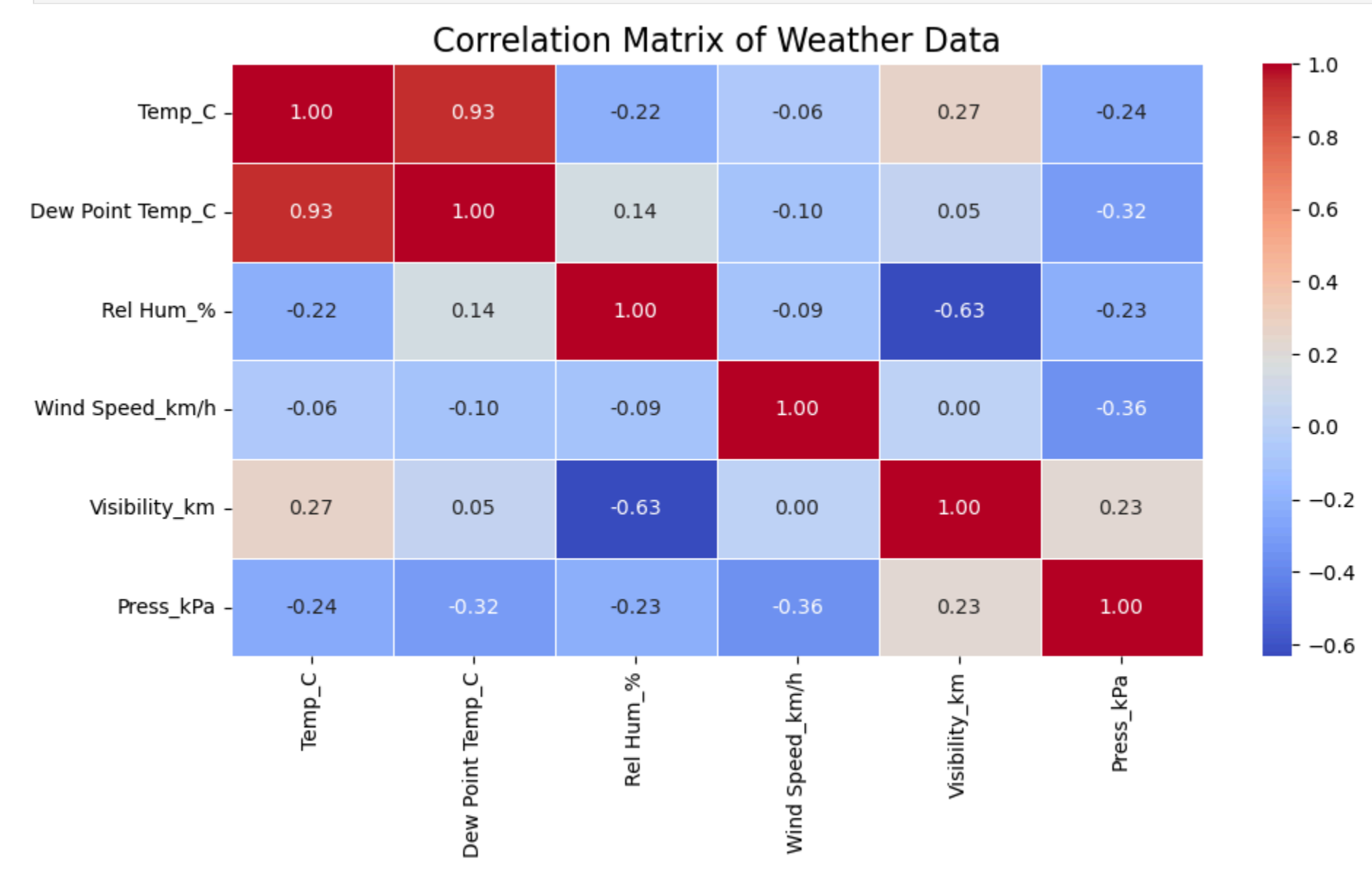
```
In [22]: # Plot Weather Conditions
plt.figure(figsize=(8, 6))
weather_conditions = weather['Weather'].value_counts()
weather_conditions.plot(kind='bar', color='skyblue')
plt.title('Weather Conditions Frequency', fontsize=8)
plt.xlabel('Weather Condition', fontsize=8)
plt.ylabel('Frequency', fontsize=8)
plt.xticks(rotation=90)
plt.grid(True)
plt.tight_layout()
plt.show()
```



9. Correlation Matrix of Numerical Variables

```
In [24]: # Select only numeric columns from the DataFrame
numeric_weather_data = weather.select_dtypes(include=['number'])

# Plot Correlation Matrix for numeric columns
plt.figure(figsize=(10, 6))
correlation_matrix = numeric_weather_data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix of Weather Data', fontsize=16)
plt.tight_layout()
plt.show()
```



10. Time Series Analysis: Seasonal Patterns

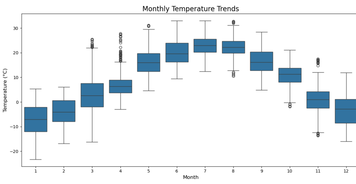
```
In [37]: # Step 1: Ensure 'Date/Time' is in datetime format
weather['Date/Time'] = pd.to_datetime(weather['Date/Time'], errors='coerce')

# Step 2: Extract Month and Hour from 'Date/Time'
weather['Month'] = weather['Date/Time'].dt.month
weather['Hour'] = weather['Date/Time'].dt.hour

# Step 3: Verify if the 'Month' and 'Hour' columns are created correctly
print(weather[['Date/Time', 'Month', 'Hour']].head())

# Step 4: Plot Monthly Temperature Trends
plt.figure(figsize=(12, 6))
sns.boxplot(x='Month', y='Temp_C', data=weather)
plt.title('Monthly Temperature Trends', fontsize=16)
plt.xlabel('Month', fontsize=12)
plt.ylabel('Temperature (°C)', fontsize=12)
plt.tight_layout()
plt.show()
```

```
            Date/Time  Month  Hour
0 2012-01-01 00:00:00      1     0
1 2012-01-01 01:00:00      1     1
2 2012-01-01 02:00:00      1     2
3 2012-01-01 03:00:00      1     3
4 2012-01-01 04:00:00      1     4
```


Monthly Temperature Trends
Temperature (°C)
30
20
10
0
−10
−20
1
2
3
4
5
6
7
8
9
10
11
12
Months