In [1]:
```
# Step 1: Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
# Step 2: Load the weather data from the CSV file
# Assuming 'weatherDataFile.csv' is in the same directory as the script
df = pd.read_csv("weatherDataFile.csv")
```

In [3]:
```
# Step 3: Display the first few rows of the dataset to understand its structure
print(df.head())
```

```
         Date  Temperature (°C)  Humidity (%)  WindSpeed (km/h)  \
0  01/02/2025              25.3            60                15
1  02/02/2025              26.1            65                10
2  03/02/2025              24.8            70                12
3  04/02/2025              22.3            75                20
4  05/02/2025              23.1            72                18

   Rainfall (mm)  Pressure (hPa) WeatherCondition       City
0            0.0            1015            Clear     Mumbai
1            1.2            1013    Partly Cloudy      Delhi
2            2.5            1010           Cloudy  Bangalore
3            0.0            1012            Rainy    Kolkata
4            0.0            1011            Clear    Chennai
```

In [4]:
```
# Step 4: Preprocess the data (e.g., converting date to datetime format)
df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y')
```

In [5]:
```
# Step 5: Perform basic analysis
# Descriptive statistics
print("Descriptive Statistics:")
print(df.describe())
```

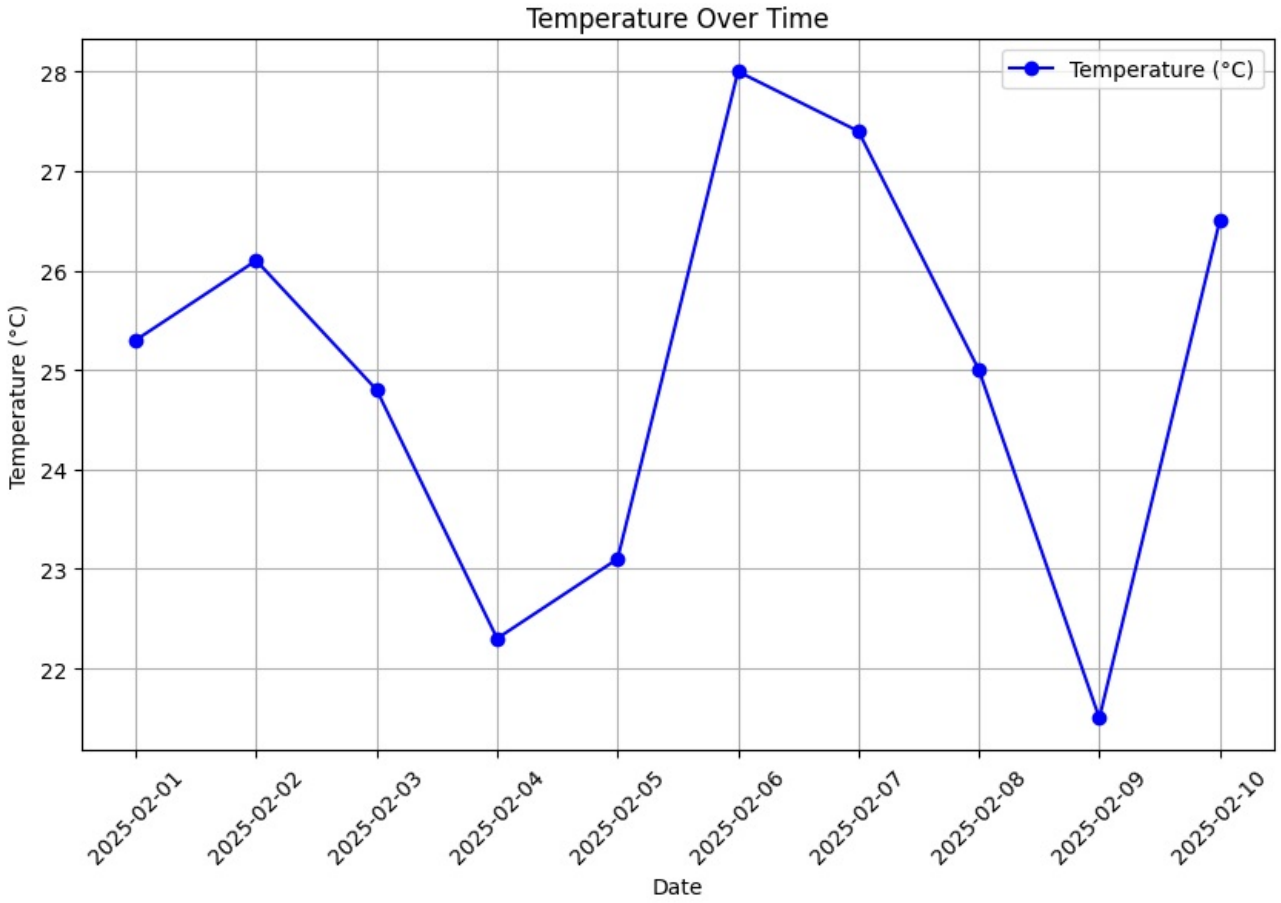
```
Descriptive Statistics:
                      Date  Temperature (°C)  Humidity (%)  WindSpeed (km/h)  \
count                   10          10.00000      10.00000         10.000000
mean   2025-02-05 12:00:00          25.00000      69.00000         15.000000
min    2025-02-01 00:00:00          21.50000      55.00000         10.000000
25%    2025-02-03 06:00:00          23.52500      61.25000         12.000000
50%    2025-02-05 12:00:00          25.15000      70.00000         15.000000
75%    2025-02-07 18:00:00          26.40000      74.25000         18.000000
max    2025-02-10 00:00:00          28.00000      85.00000         20.000000
std                    NaN           2.14735       9.64941          3.887301

       Rainfall (mm)  Pressure (hPa)
count      10.000000       10.000000
mean        1.320000     1012.500000
min         0.000000     1010.000000
25%         0.000000     1011.000000
50%         0.750000     1012.500000
75%         2.175000     1013.750000
max         5.000000     1016.000000
std         1.683779        2.068279
```

In [6]:
```
#to get information of dataframe
df.info()
```

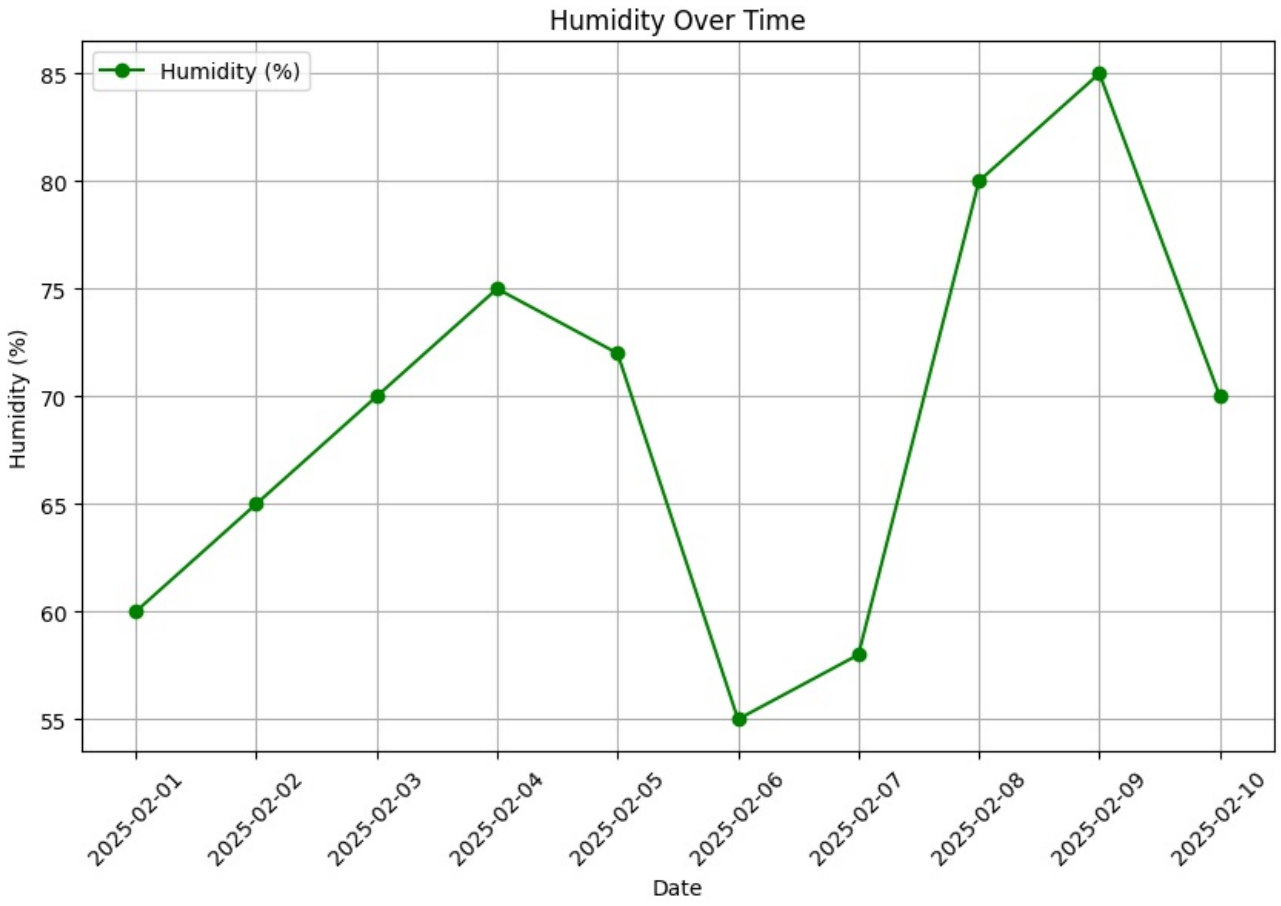
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Date              10 non-null     datetime64[ns]
 1   Temperature (°C)  10 non-null     float64
 2   Humidity (%)      10 non-null     int64
 3   WindSpeed (km/h)  10 non-null     int64
 4   Rainfall (mm)     10 non-null     float64
 5   Pressure (hPa)    10 non-null     int64
 6   WeatherCondition  10 non-null     object
 7   City              10 non-null     object
dtypes: datetime64[ns](1), float64(2), int64(3), object(2)
memory usage: 772.0+ bytes
```

In [7]:
```
# Step 6: Data Visualization
# Plot temperature over time
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['Temperature (°C)'], marker='o', color='b', label='Temperature (°C)')
plt.title('Temperature Over Time')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.grid(True)
```

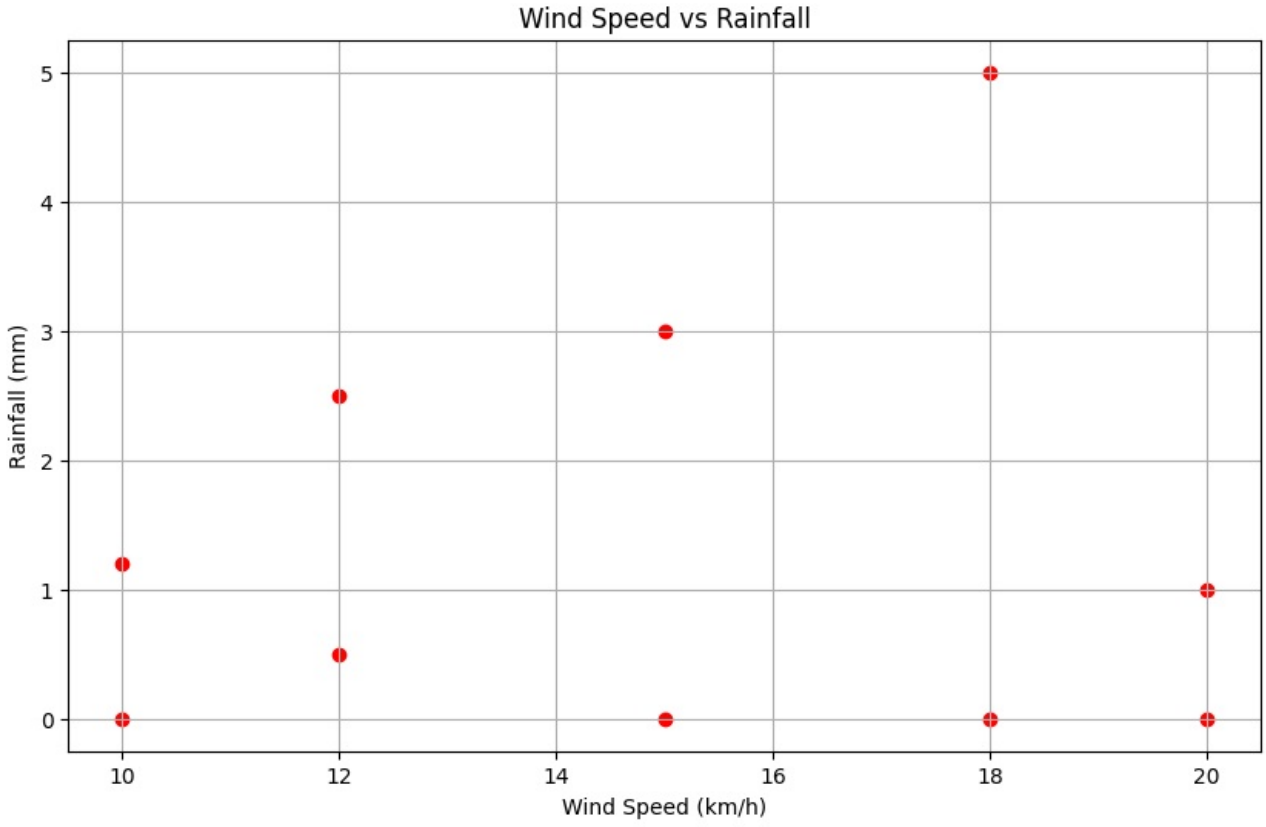
```
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



In [8]:
```
# Plot humidity over time
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['Humidity (%)'], marker='o', color='g', label='Humidity (%)')
plt.title('Humidity Over Time')
plt.xlabel('Date')
plt.ylabel('Humidity (%)')
plt.grid(True)
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



In [9]:
```
# Plot Wind Speed vs Rainfall
plt.figure(figsize=(10, 6))
plt.scatter(df['WindSpeed (km/h)'], df['Rainfall (mm)'], color='r')
plt.title('Wind Speed vs Rainfall')
plt.xlabel('Wind Speed (km/h)')
plt.ylabel('Rainfall (mm)')
plt.grid(True)
plt.show()
```



In [10]:
```
# Step 7: Group by city and calculate the average values
city_group = df.groupby('City').agg({
    'Temperature (°C)': 'mean',
    'Humidity (%)': 'mean',
    'WindSpeed (km/h)': 'mean',
```

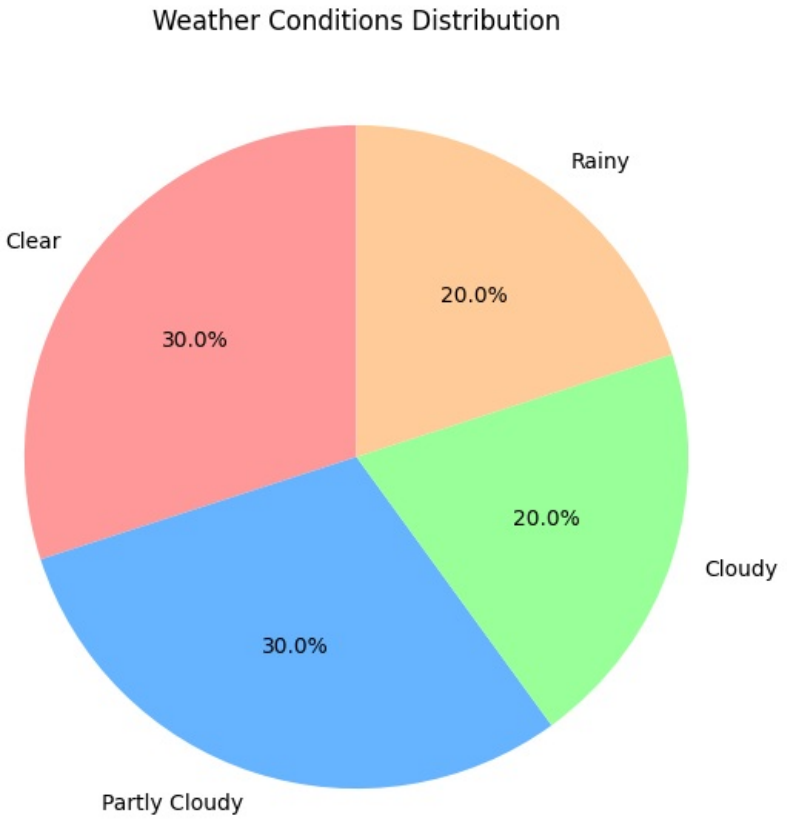
```
    'Rainfall (mm)': 'mean',
    'Pressure (hPa)': 'mean'
}).sort_values(by='Temperature (°C)', ascending=False)

print("Average Weather Conditions by City:")
print(city_group)
```

```
Average Weather Conditions by City:
           Temperature (°C)  Humidity (%)  WindSpeed (km/h)  Rainfall (mm)  \
City
Hyderabad              28.0          55.0              10.0            0.0
Pune                   27.4          58.0              12.0            0.5
Jaipur                 26.5          70.0              20.0            1.0
Delhi                  26.1          65.0              10.0            1.2
Mumbai                 25.3          60.0              15.0            0.0
Ahmedabad              25.0          80.0              15.0            3.0
Bangalore              24.8          70.0              12.0            2.5
Chennai                23.1          72.0              18.0            0.0
Kolkata                22.3          75.0              20.0            0.0
Surat                  21.5          85.0              18.0            5.0

           Pressure (hPa)
City
Hyderabad          1016.0
Pune               1014.0
Jaipur             1013.0
Delhi              1013.0
Mumbai             1015.0
Ahmedabad          1011.0
Bangalore          1010.0
Chennai            1011.0
Kolkata            1012.0
Surat              1010.0
```

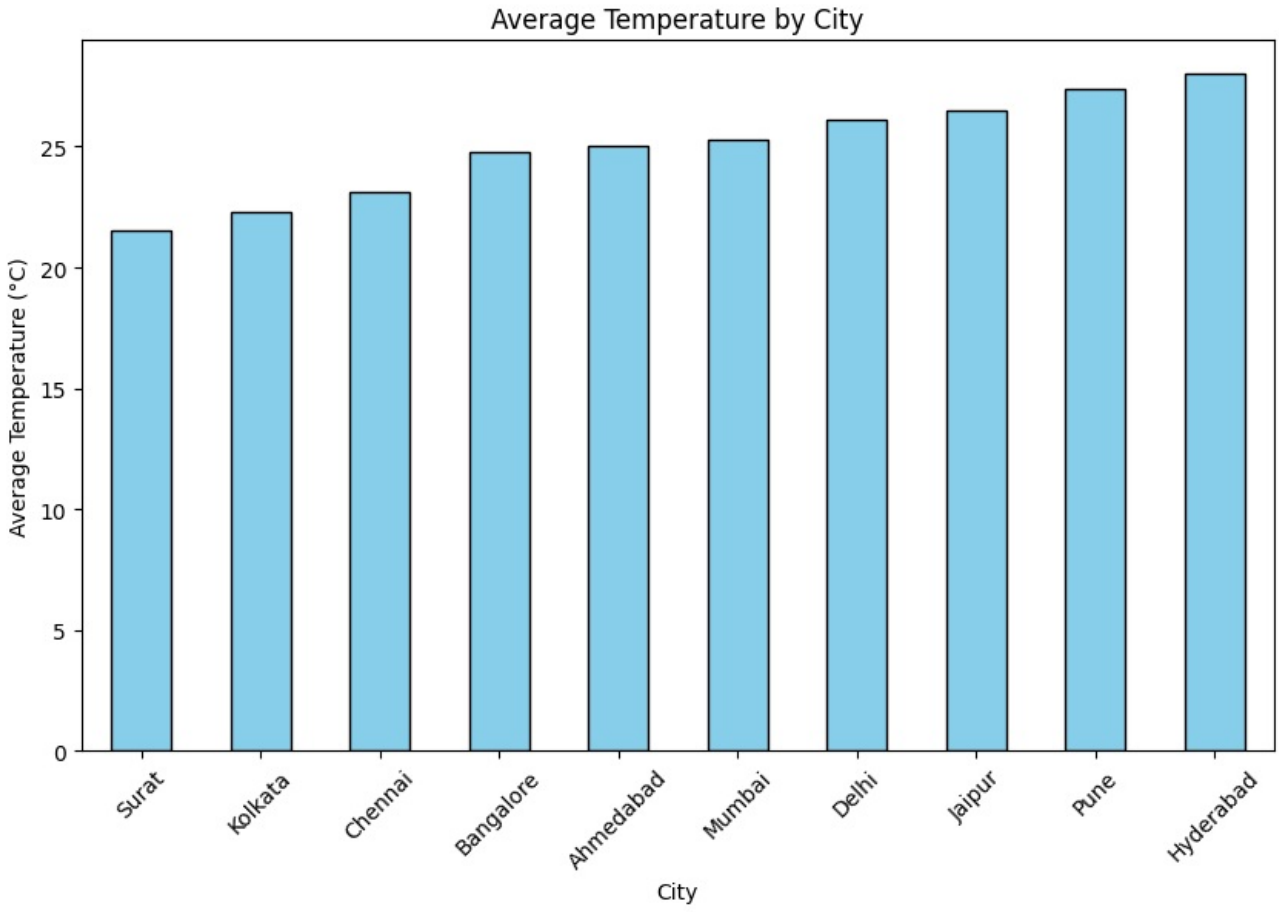
In [11]:
```
# Step 8: Filter the data for specific weather conditions (e.g., 'Clear' weather)
clear_weather = df[df['WeatherCondition'] == 'Clear']
print("Data for Clear Weather:")
print(clear_weather)

# You can also create additional plots, such as:
# - Rainfall over time
# - Wind speed comparison between cities
```

```
Data for Clear Weather:
        Date  Temperature (°C)  Humidity (%)  WindSpeed (km/h)  Rainfall (mm)  \
0 2025-02-01              25.3            60                15            0.0
4 2025-02-05              23.1            72                18            0.0
5 2025-02-06              28.0            55                10            0.0

   Pressure (hPa) WeatherCondition       City
0            1015            Clear     Mumbai
4            1011            Clear    Chennai
5            1016            Clear  Hyderabad
```

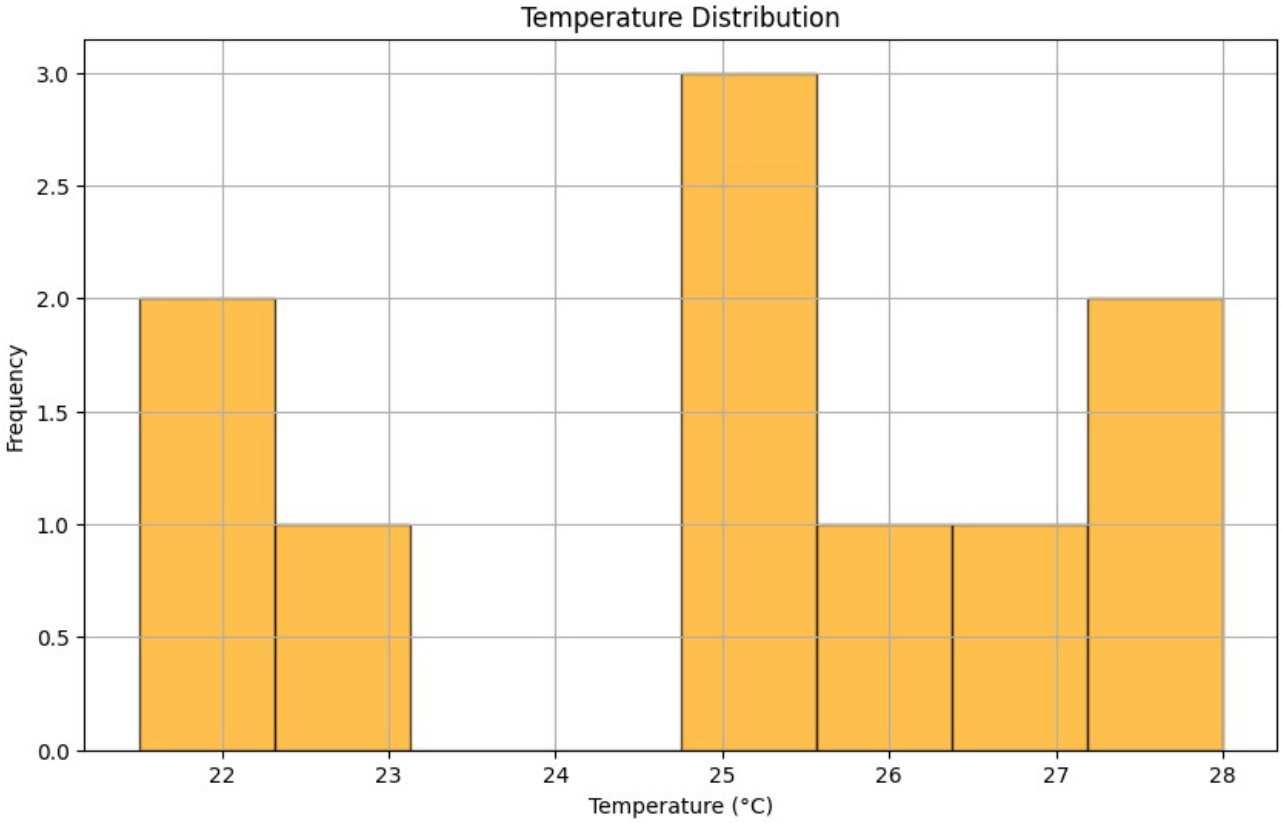
In [12]:
```
# 1. **Pie Chart** for Weather Conditions Distribution
weather_condition_counts = df['WeatherCondition'].value_counts()

# Create Pie Chart
plt.figure(figsize=(7, 7))
plt.pie(weather_condition_counts, labels=weather_condition_counts.index, autopct='%1.1f%%', startangle=90, colo
plt.title('Weather Conditions Distribution')
plt.ylabel('')  # To remove the y-label that pandas adds
plt.show()
```



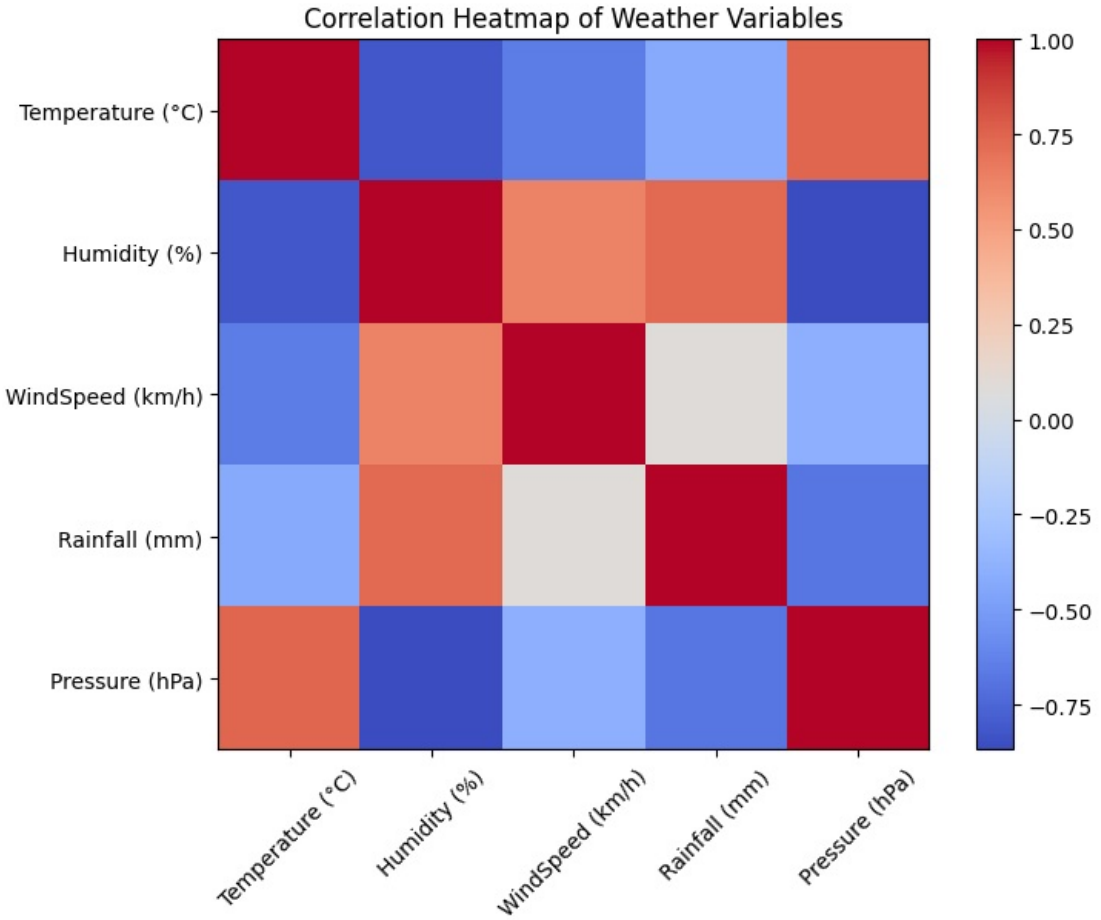
In [13]:
```
# 2. **Bar Plot** for Average Temperature by City
avg_temp_by_city = df.groupby('City')['Temperature (°C)'].mean().sort_values()

# Create Bar Plot for Average Temperature
plt.figure(figsize=(10, 6))
avg_temp_by_city.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Average Temperature by City')
plt.xlabel('City')
plt.ylabel('Average Temperature (°C)')
plt.xticks(rotation=45)
plt.show()
```



In [14]:
```
# 3. **Histogram** for Temperature Distribution
plt.figure(figsize=(10, 6))
plt.hist(df['Temperature (°C)'], bins=8, color='orange', edgecolor='black', alpha=0.7)
plt.title('Temperature Distribution')
plt.xlabel('Temperature (°C)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

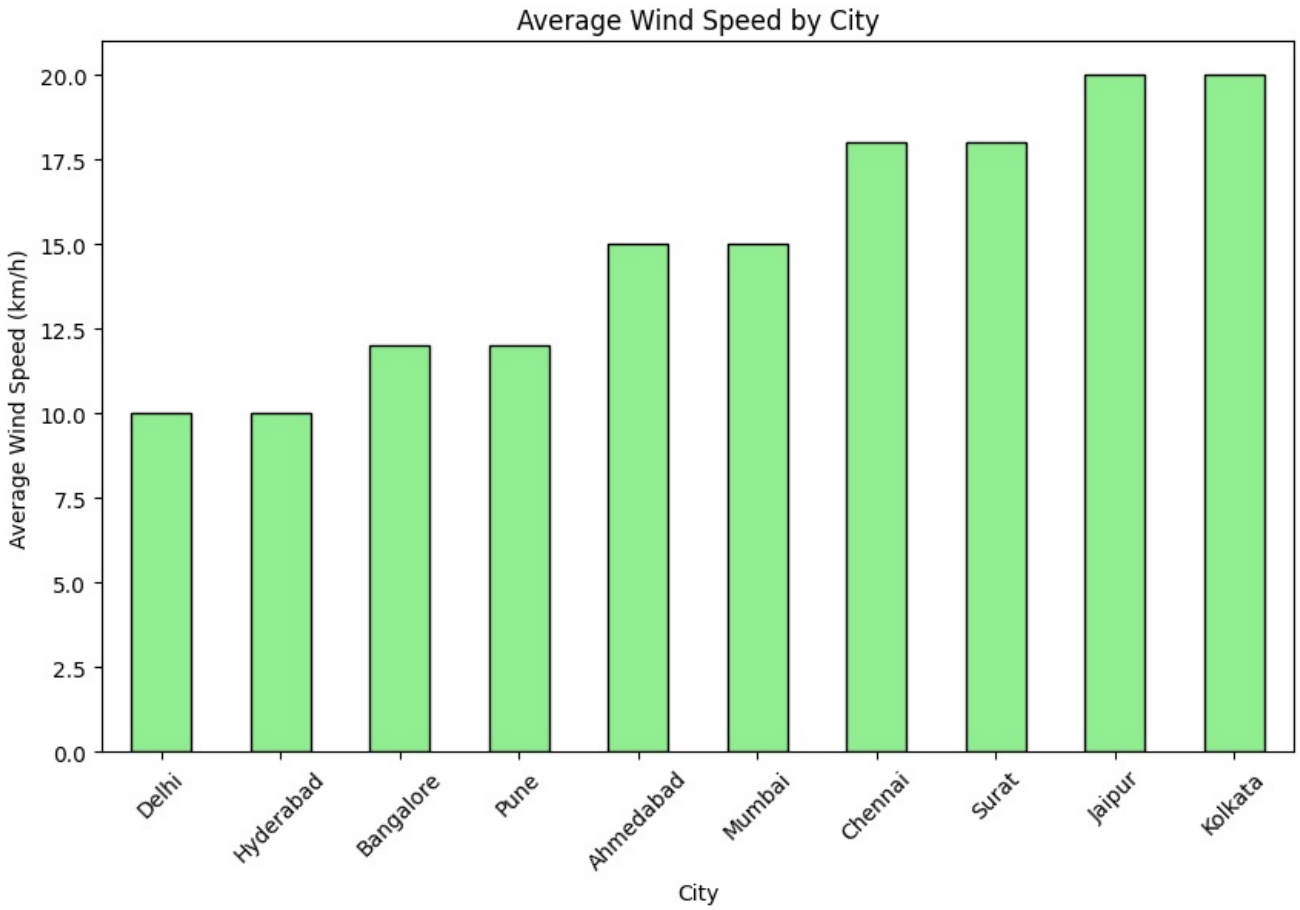


In [15]:
```
# 4. **Correlation Chart** for Weather Variables (Heatmap)

# Calculate the correlation matrix for numerical columns
correlation_matrix = df[['Temperature (°C)', 'Humidity (%)', 'WindSpeed (km/h)', 'Rainfall (mm)', 'Pressure (hP

# Create a heatmap of correlations
plt.figure(figsize=(8, 6))
plt.imshow(correlation_matrix, cmap='coolwarm', interpolation='none')
plt.colorbar()
plt.xticks(np.arange(len(correlation_matrix.columns)), correlation_matrix.columns, rotation=45)
plt.yticks(np.arange(len(correlation_matrix.columns)), correlation_matrix.columns)
plt.title('Correlation Heatmap of Weather Variables')
plt.show()
```



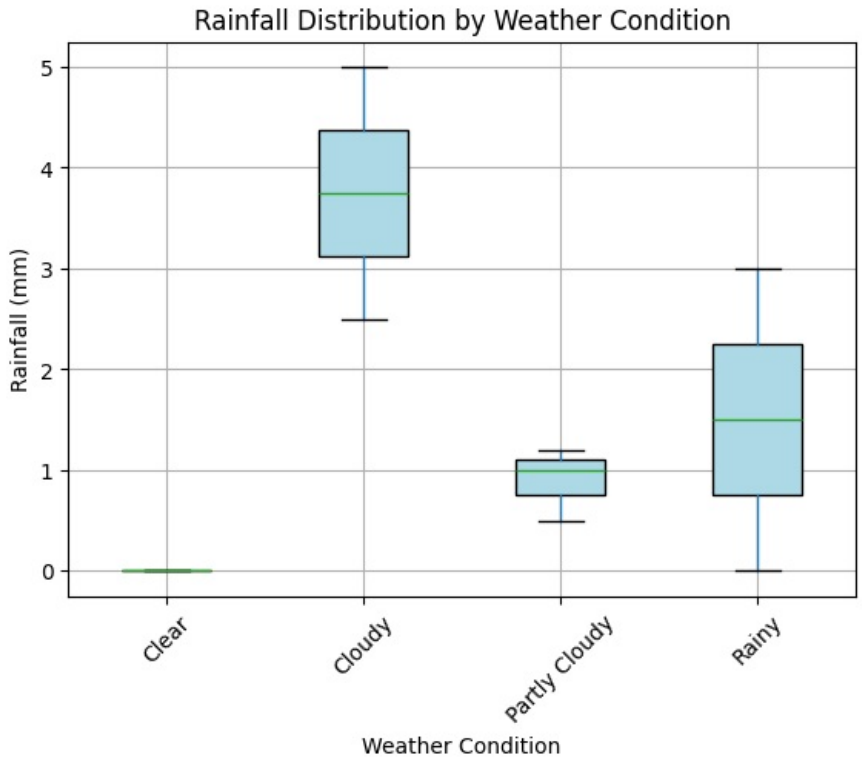
In [16]:

```
# 5. **Bar Plot** for Average Wind Speed by City
avg_wind_by_city = df.groupby('City')['WindSpeed (km/h)'].mean().sort_values()

# Create Bar Plot for Average Wind Speed
plt.figure(figsize=(10, 6))
avg_wind_by_city.plot(kind='bar', color='lightgreen', edgecolor='black')
plt.title('Average Wind Speed by City')
plt.xlabel('City')
plt.ylabel('Average Wind Speed (km/h)')
plt.xticks(rotation=45)
plt.show()
```



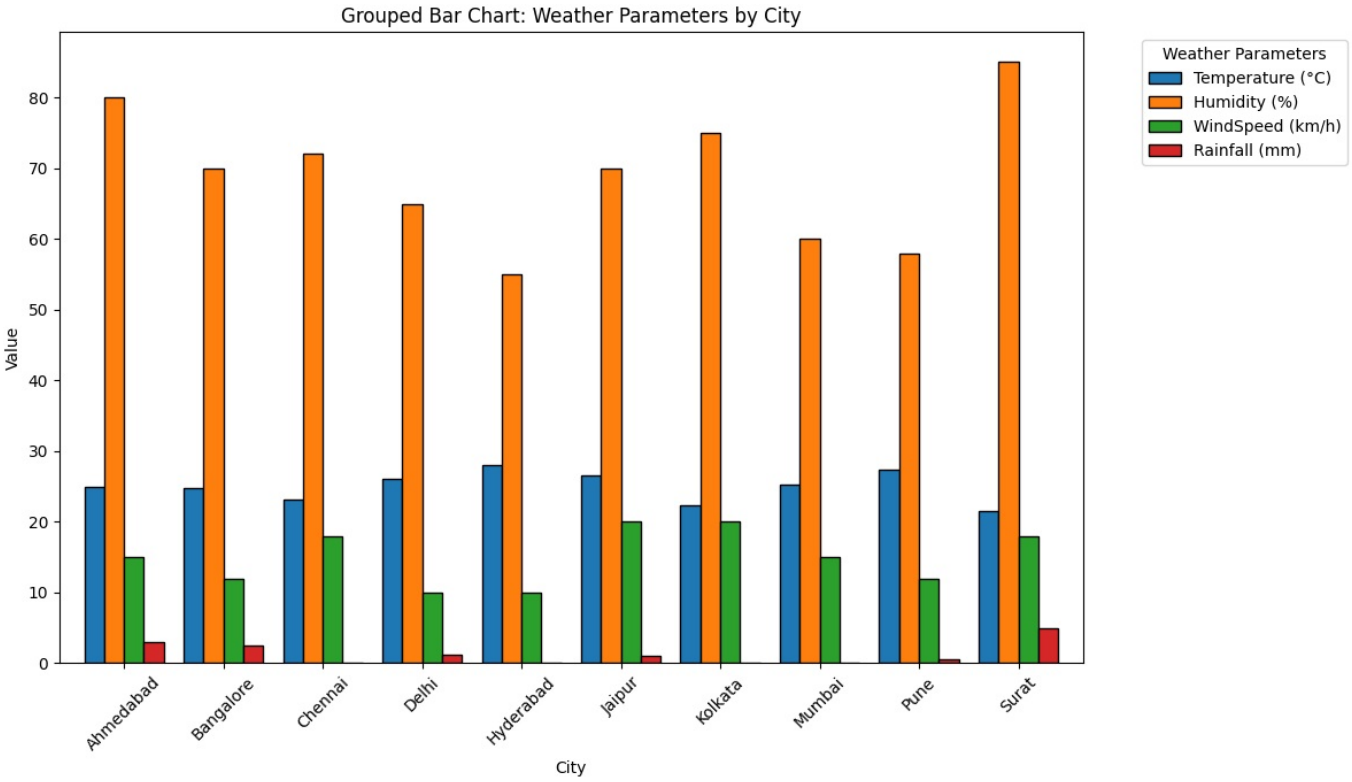
In [17]:
```
# 6. **Box Plot** for Rainfall Distribution by Weather Condition
# Create Box Plot for Rainfall based on Weather Condition
plt.figure(figsize=(10, 6))
df.boxplot(column='Rainfall (mm)', by='WeatherCondition', patch_artist=True, boxprops=dict(facecolor='lightblue
plt.title('Rainfall Distribution by Weather Condition')
plt.suptitle('')  # To remove the default title added by pandas
plt.xlabel('Weather Condition')
plt.ylabel('Rainfall (mm)')
plt.xticks(rotation=45)
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



In [18]:
```
# 7. **Grouped Bar Chart** for Multiple Weather Parameters by City
# We will create a grouped bar chart for Temperature, Humidity, Wind Speed, and Rainfall for each city.
weather_params = ['Temperature (°C)', 'Humidity (%)', 'WindSpeed (km/h)', 'Rainfall (mm)']
```

```
# Calculate the mean values for each weather parameter by city
weather_data_by_city = df.groupby('City')[weather_params].mean()

# Create Grouped Bar Chart
weather_data_by_city.plot(kind='bar', figsize=(12, 7), width=0.8, edgecolor='black')
plt.title('Grouped Bar Chart: Weather Parameters by City')
plt.xlabel('City')
plt.ylabel('Value')
plt.xticks(rotation=45)
plt.legend(title='Weather Parameters', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()  # To ensure the labels fit
plt.show()
```



In [ ]: