# Weather Trends Analysis – Climate Pattern Insights

---

## 🟦 Title Page

**Project Title:** Weather Trends Analysis – Climate Pattern Insights

**Author:** Rahul Mahakal

**Domain:** Weather & Climate Analytics

**Tools & Technologies Used:**

- Python
- Pandas & NumPy
- Matplotlib & Seaborn
- Jupyter Notebook
- Modular Python Architecture
- OS-independent file handling using `pathlib`

---

## 🟦 Executive Summary

This project analyzes historical weather data to identify temperature trends, seasonal variations, and relationships among key meteorological variables such as humidity, wind speed, and atmospheric pressure. Using statistical analysis and advanced visualizations, the study transforms raw weather data into meaningful climate insights that can support environmental monitoring, forecasting, and long-term planning.

---

## 🟦 Introduction

### Problem Statement

Understanding weather patterns and climate trends is essential for sectors such as agriculture, energy, transportation, and urban planning. Raw weather data alone does not provide actionable insight without structured analysis and visualization.

### Objectives

- Analyze temperature trends over time
- Identify seasonal temperature patterns
- Examine humidity, wind, and pressure distributions

- Explore relationships between weather variables
- Generate climate insights and recommendations

---

## 🟦 Data Source

Publicly available historical weather dataset used for analytical practice.

### Dataset Size

- Records: Based on CSV file
- Time Span: Multiple consecutive years
- Missing Values: Validated and handled during preprocessing

---

# 🟦 Methodology

## Analytics Pipeline

1. Data ingestion and validation
2. Datetime parsing and feature engineering
3. Exploratory data analysis
4. Statistical summarization
5. Correlation analysis
6. Insight generation

## Tools & Techniques

- Time-series analysis
- GroupBy aggregations
- Descriptive statistics (mean, median)
- Correlation analysis
- Static data visualizations

---

### 🟦 DATA LOADING & COLUMN SAFETY

In [1]:
```
from pathlib import Path
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

import sys
from pathlib import Path

# Add project root to Python path
PROJECT_ROOT = Path("..").resolve()
if str(PROJECT_ROOT) not in sys.path:
    sys.path.append(str(PROJECT_ROOT))
```

```python
sns.set_theme(style="whitegrid")

BASE_DIR = Path.cwd().parent
DATA_DIR = BASE_DIR / "datasets"

df = pd.read_csv(DATA_DIR / "weatherHistory.csv")
df.head()
```

Out[1]:

| | Formatted Date | Summary | Precip Type | Temperature (C) | Apparent Temperature (C) | Humidity | Wind Speed (km/h) | Wind Bearing (degrees) |
|---|---|---|---|---|---|---|---|---|
| **0** | 2006-04-01 00:00:00.000 +0200 | Partly Cloudy | rain | 9.472222 | 7.388889 | 0.89 | 14.1197 | 251.0 |
| **1** | 2006-04-01 01:00:00.000 +0200 | Partly Cloudy | rain | 9.355556 | 7.227778 | 0.86 | 14.2646 | 259.0 |
| **2** | 2006-04-01 02:00:00.000 +0200 | Mostly Cloudy | rain | 9.377778 | 9.377778 | 0.89 | 3.9284 | 204.0 |
| **3** | 2006-04-01 03:00:00.000 +0200 | Partly Cloudy | rain | 8.288889 | 5.944444 | 0.83 | 14.1036 | 269.0 |
| **4** | 2006-04-01 04:00:00.000 +0200 | Mostly Cloudy | rain | 8.755556 | 6.977778 | 0.83 | 11.0446 | 259.0 |

In [2]:

```python
# Notebook is now a consumer only

from src.weather_trends_analysis.preprocessing import preprocess_weather_data
from src.weather_trends_analysis.analysis import temperature_overview
from src.weather_trends_analysis.insights import generate_weather_insights

df = preprocess_weather_data(df)
stats = temperature_overview(df)
stats
```

Out[2]:

```
{'mean_temperature': np.float64(11.93267843751188),
 'median_temperature': np.float64(12.0),
 'min_temperature': np.float64(-21.822222222222223),
 'max_temperature': np.float64(39.90555555555555)}
```

## 🟦 COLUMN INSPECTION

In [3]:

```python
df.columns
```

Out[3]:
```
Index(['Formatted Date', 'Summary', 'Precip Type', 'Temperature (C)',
       'Apparent Temperature (C)', 'Humidity', 'Wind Speed (km/h)',
       'Wind Bearing (degrees)', 'Visibility (km)', 'Loud Cover',
       'Pressure (millibars)', 'Daily Summary', 'Year', 'Month', 'Month_Name'],
      dtype='object')
```

In [4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96453 entries, 0 to 96452
Data columns (total 15 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Formatted Date            96453 non-null  datetime64[ns, UTC]
 1   Summary                   96453 non-null  object
 2   Precip Type               95936 non-null  object
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  float64
 8   Visibility (km)           96453 non-null  float64
 9   Loud Cover                96453 non-null  float64
 10  Pressure (millibars)      96453 non-null  float64
 11  Daily Summary             96453 non-null  object
 12  Year                      96453 non-null  int32
 13  Month                     96453 non-null  int32
 14  Month_Name                96453 non-null  object
dtypes: datetime64[ns, UTC](1), float64(8), int32(2), object(4)
memory usage: 10.3+ MB
```

In [5]:
```python
df.isnull().sum()
```

Out[5]:
```
Formatted Date                0
Summary                       0
Precip Type                 517
Temperature (C)               0
Apparent Temperature (C)      0
Humidity                      0
Wind Speed (km/h)             0
Wind Bearing (degrees)        0
Visibility (km)               0
Loud Cover                    0
Pressure (millibars)          0
Daily Summary                 0
Year                          0
Month                         0
Month_Name                    0
dtype: int64
```

## 🟦 FEATURE ENGINEERING

In [6]:
```python
# Convert date column safely
df["Formatted Date"] = pd.to_datetime(df["Formatted Date"], utc=True)

# Time-based features
df["Year"] = df["Formatted Date"].dt.year
df["Month"] = df["Formatted Date"].dt.month
df["Month_Name"] = df["Formatted Date"].dt.month_name()
```

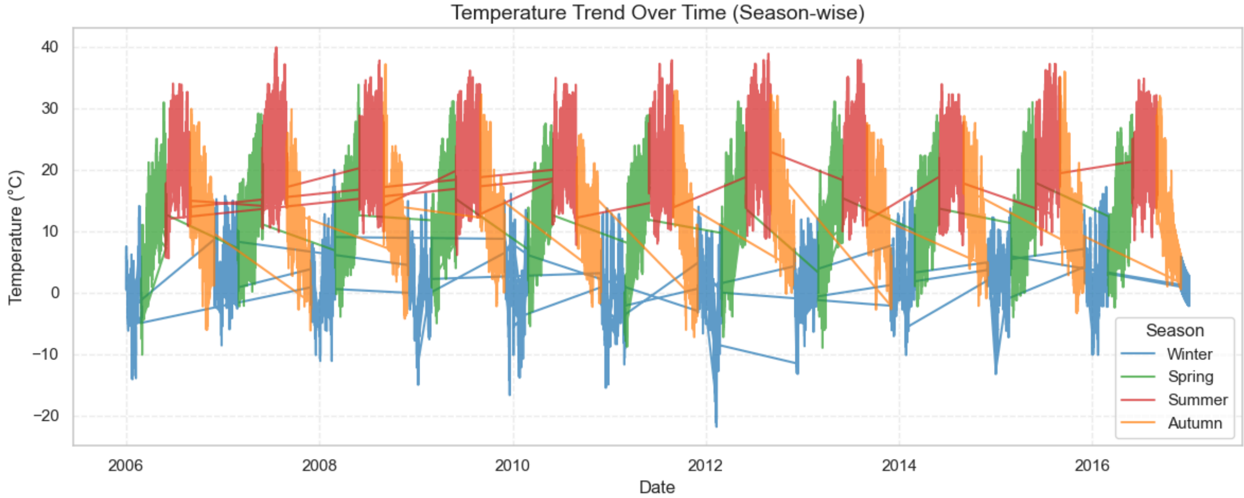
## Create season column

In [7]:
```python
df["Month"] = df["Formatted Date"].dt.month

def get_season(month):
    if month in [12, 1, 2]:
        return "Winter"
    elif month in [3, 4, 5]:
        return "Spring"
    elif month in [6, 7, 8]:
        return "Summer"
    else:
        return "Autumn"

df["Season"] = df["Month"].apply(get_season)
```

# 🟦 VISUALIZATIONS

### Temperature Trend Over Time (EDA 1)

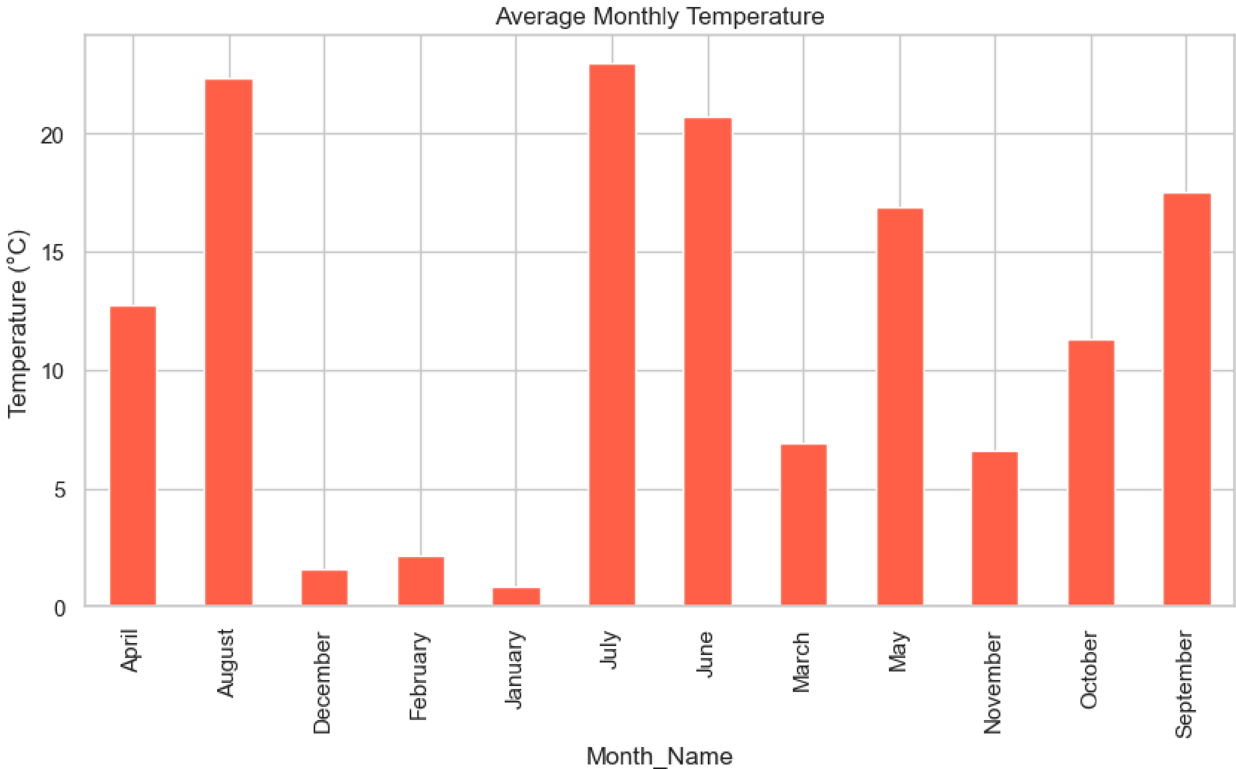
In [8]:
```python
plt.figure(figsize=(12, 5))

season_colors = {
    "Winter": "tab:blue",
    "Spring": "tab:green",
    "Summer": "tab:red",
    "Autumn": "tab:orange"
}

for season, color in season_colors.items():
    subset = df[df["Season"] == season]
    plt.plot(
        subset["Formatted Date"],
        subset["Temperature (C)"],
        color=color,
        label=season,
        alpha=0.7
    )

plt.title("Temperature Trend Over Time (Season-wise)", fontsize=14)
plt.xlabel("Date")
plt.ylabel("Temperature (°C)")
plt.legend(title="Season")
plt.grid(True, linestyle="--", alpha=0.3)

plt.tight_layout()
plt.show()
```




**Caption: -** Visualizes long-term temperature changes across the dataset timeline.

### Monthly Average Temperature (EDA 2)

In [9]:
```python
monthly_temp = df.groupby("Month_Name")["Temperature (C)"].mean()

monthly_temp.plot(kind="bar", figsize=(10, 5), color="tomato")
plt.title("Average Monthly Temperature")
plt.ylabel("Temperature (°C)")
plt.show()
```

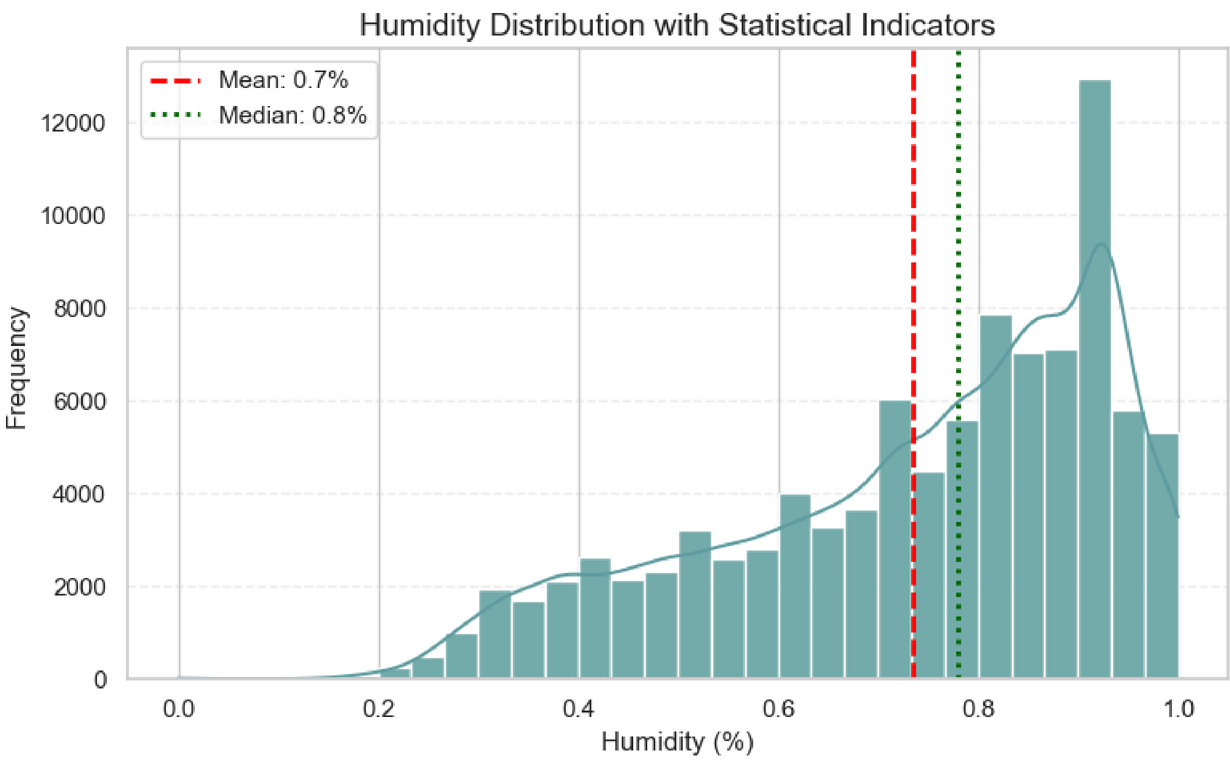



**Caption: -** Highlights seasonal temperature variations across months.

### Humidity Distribution (EDA 3)

In [10]:
```python
plt.figure(figsize=(8, 5))

sns.histplot(
```

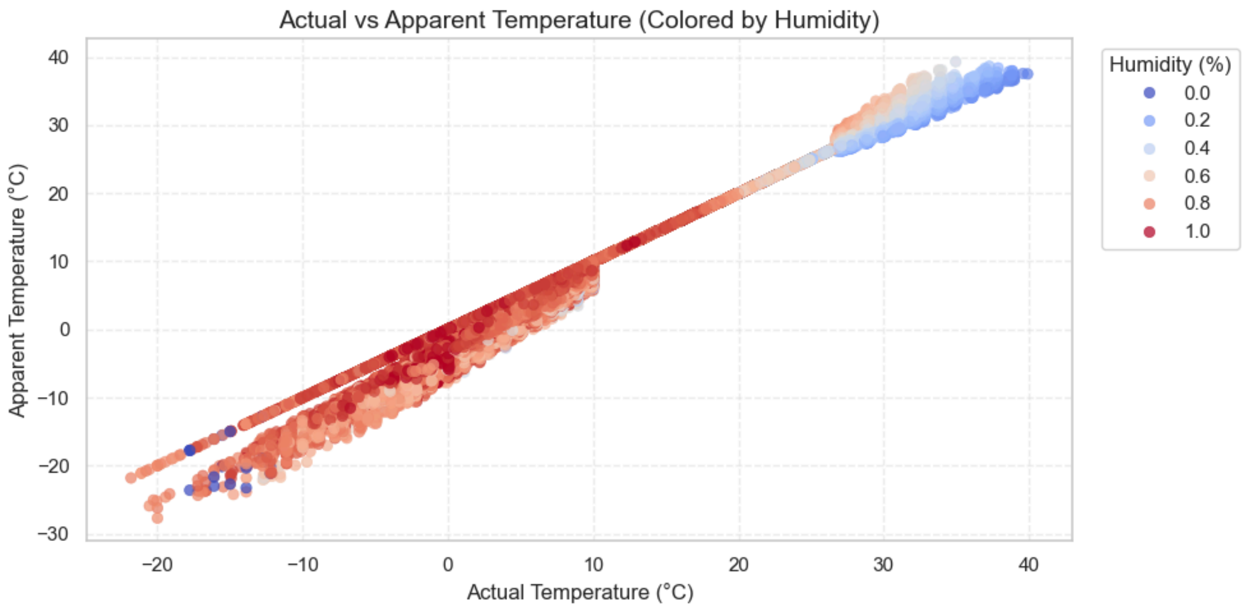
```
    df["Humidity"],
    bins=30,
    kde=True,
    color="cadetblue",
    edgecolor="white",
    alpha=0.85
)

plt.axvline(
    df["Humidity"].mean(),
    color="red",
    linestyle="--",
    linewidth=2,
    label=f"Mean: {df['Humidity'].mean():.1f}%"
)

plt.axvline(
    df["Humidity"].median(),
    color="darkgreen",
    linestyle=":",
    linewidth=2,
    label=f"Median: {df['Humidity'].median():.1f}%"
)

plt.title("Humidity Distribution with Statistical Indicators", fontsize=14)
plt.xlabel("Humidity (%)")
plt.ylabel("Frequency")
plt.legend()
plt.grid(axis="y", linestyle="--", alpha=0.3)

plt.tight_layout()
plt.show()
```




**Caption: -** Shows how humidity values are distributed across observations.

### Apparent vs Actual Temperature (EDA 4)

In [11]:
```
plt.figure(figsize=(10, 5))

sns.scatterplot(
    data=df,
    x="Temperature (C)",
    y="Apparent Temperature (C)",
    hue="Humidity",
    palette="coolwarm",
    alpha=0.7,
    edgecolor=None
)

plt.title("Actual vs Apparent Temperature (Colored by Humidity)", fontsize=14)
plt.xlabel("Actual Temperature (°C)")
plt.ylabel("Apparent Temperature (°C)")
plt.legend(title="Humidity (%)", bbox_to_anchor=(1.02, 1))
plt.grid(True, linestyle="--", alpha=0.3)

plt.tight_layout()
plt.show()
```

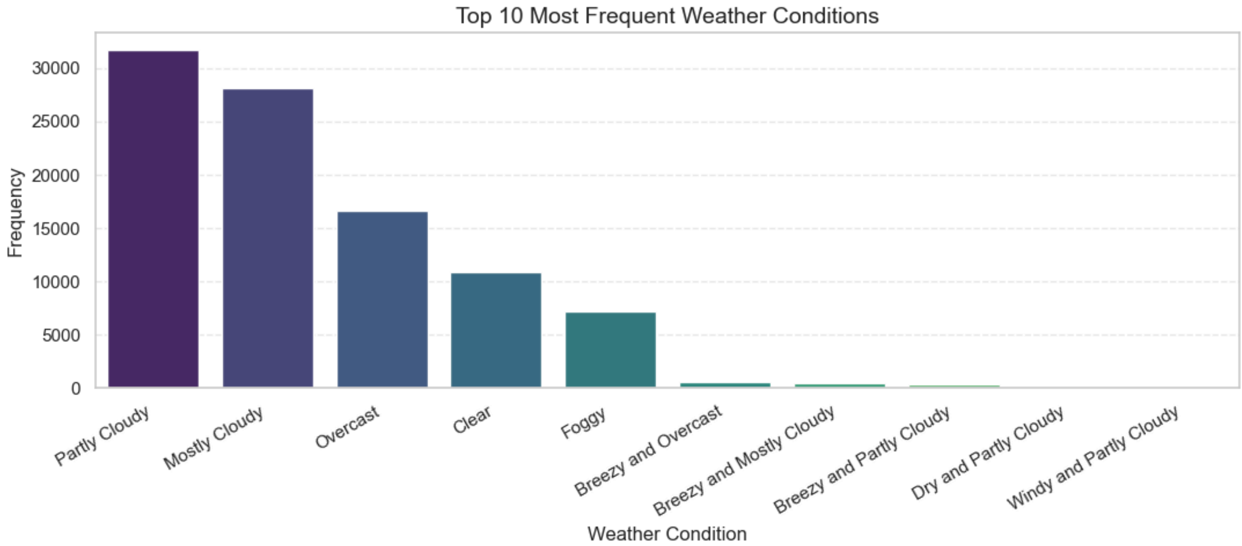



**Caption: -** Examines the difference between measured and perceived temperature.

### Weather Type Frequency (EDA 5)

In [12]:
```
top_conditions = df["Summary"].value_counts().head(10)

plt.figure(figsize=(11, 5))

sns.barplot(
    x=top_conditions.index,
    y=top_conditions.values,
    hue=top_conditions.index,
    palette="viridis",
    legend=False
)
```

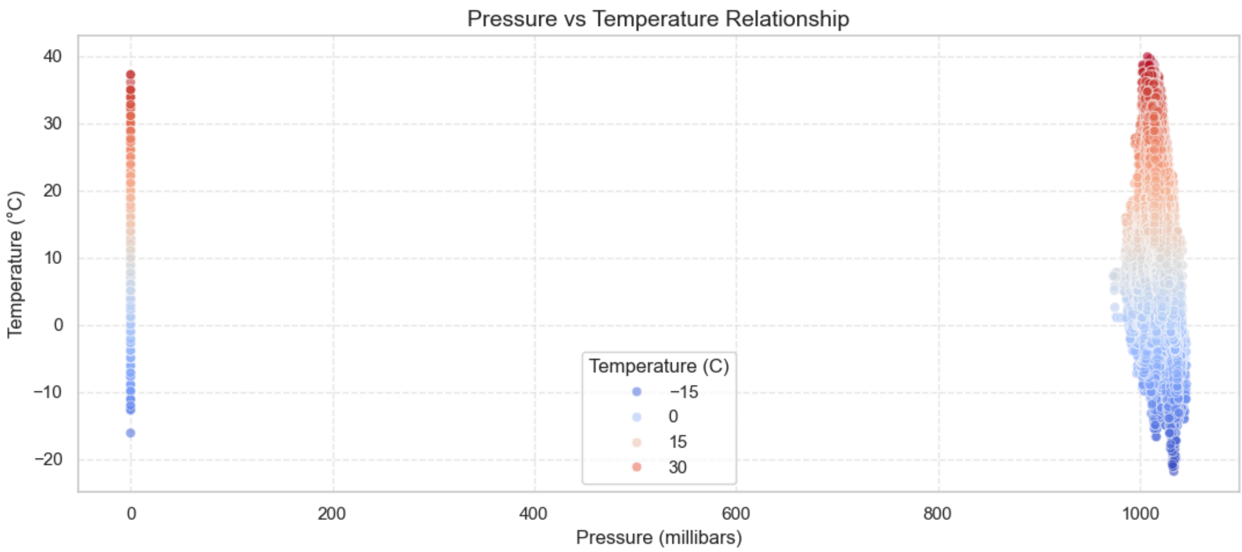
```
plt.title("Top 10 Most Frequent Weather Conditions", fontsize=14)
plt.xlabel("Weather Condition")
plt.ylabel("Frequency")
plt.xticks(rotation=30, ha="right")
plt.grid(axis="y", linestyle="--", alpha=0.4)

plt.tight_layout()
plt.show()
```




**Caption: -** Displays the most frequently occurring weather conditions.

### Pressure vs Temperature Scatter (EDA 6)

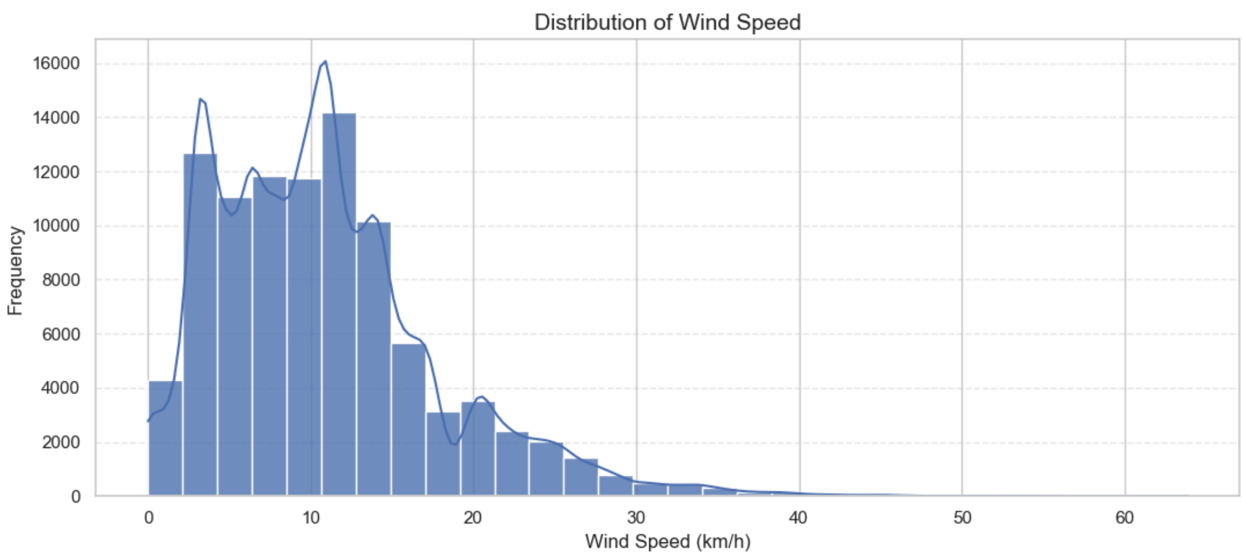
In [13]:
```
plt.figure(figsize=(11, 5))

sns.scatterplot(
    data=df,
    x="Pressure (millibars)",
    y="Temperature (C)",
    hue="Temperature (C)",          # meaningful color encoding
    palette="coolwarm",
    alpha=0.6,
    legend=True
)

plt.title("Pressure vs Temperature Relationship", fontsize=14)
plt.xlabel("Pressure (millibars)")
plt.ylabel("Temperature (°C)")
plt.grid(True, linestyle="--", alpha=0.4)

plt.tight_layout()
plt.show()
```




**Caption: -** Shows atmospheric behavior & Signals deeper meteorological understanding

### Wind Speed Distribution (EDA 7)

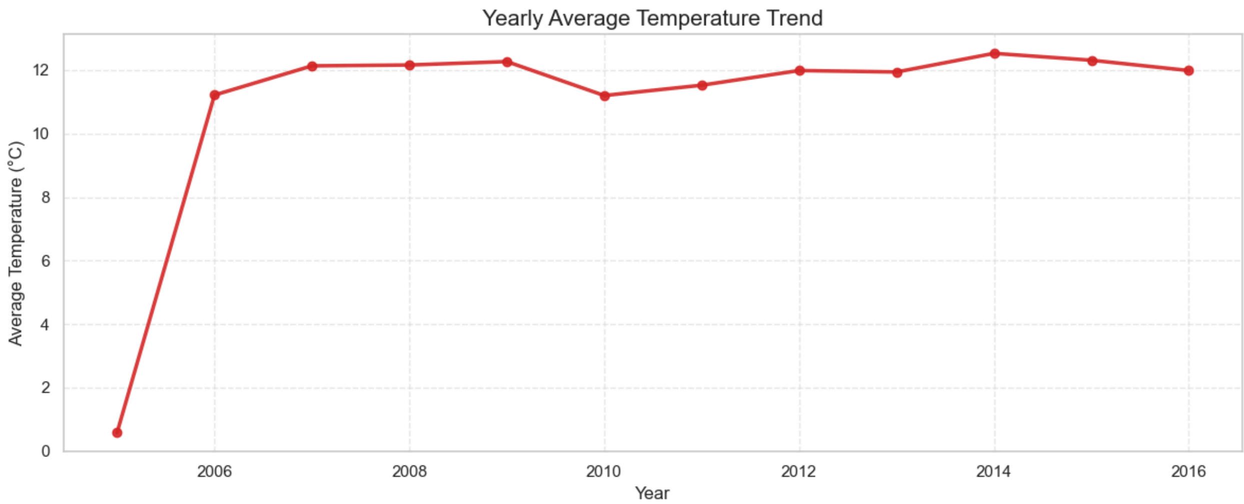
In [14]:
```
plt.figure(figsize=(11, 5))

sns.histplot(
    data=df,
    x="Wind Speed (km/h)",
    bins=30,
    kde=True,
    color="#4C72B0",
    edgecolor="white",
    alpha=0.8
)

plt.title("Distribution of Wind Speed", fontsize=14)
plt.xlabel("Wind Speed (km/h)")
plt.ylabel("Frequency")
plt.grid(axis="y", linestyle="--", alpha=0.4)

plt.tight_layout()
plt.show()
```



**Caption: -** Completes weather variable coverage & Supports forecasting insights

### Yearly Average Temperature Trend (EDA 8)

In [15]:
```
yearly_avg = df.groupby("Year")["Temperature (C)"].mean()

plt.figure(figsize=(12, 5))

plt.plot(
    yearly_avg.index,
    yearly_avg.values,
    marker="o",
    linewidth=2.5,
    color="#D62728",          # warm red → temperature semantics
    alpha=0.9
)

plt.title("Yearly Average Temperature Trend", fontsize=15)
plt.xlabel("Year")
plt.ylabel("Average Temperature (°C)")
plt.grid(True, linestyle="--", alpha=0.4)

plt.tight_layout()
plt.show()
```

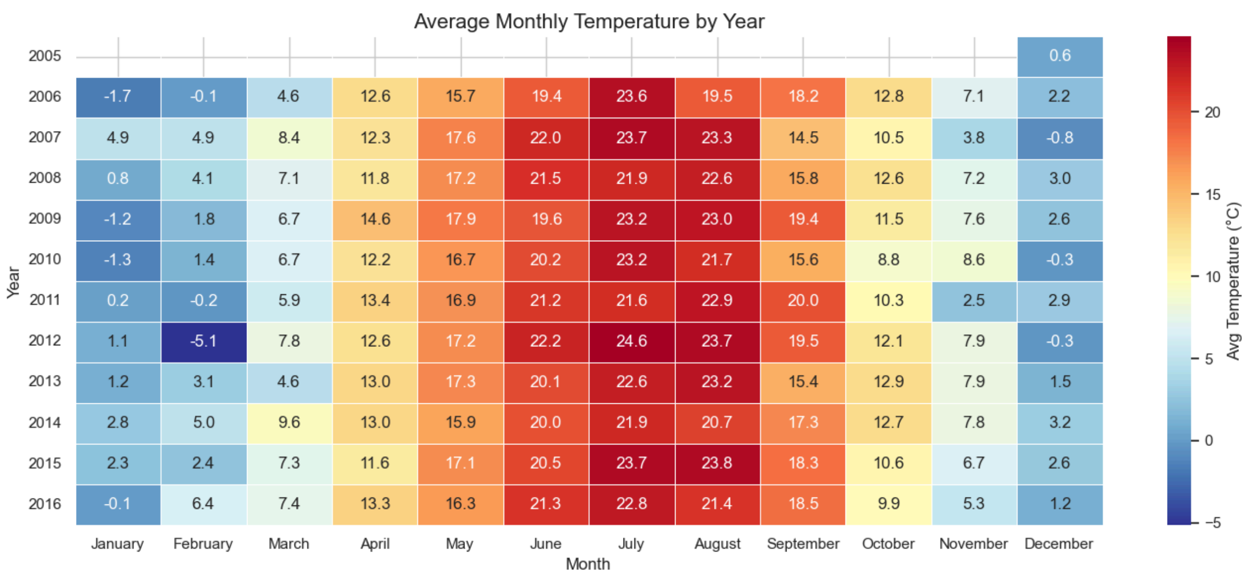



**Caption: -** Makes climate trend clearer than raw daily data & Excellent for climate change discussion

### Heatmap of Temperature by Month & Year (EDA 9)

In [16]:
```
# Ensure months are ordered correctly
month_order = [
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
]

pivot = (
    df.pivot_table(
        index="Year",
        columns="Month_Name",
        values="Temperature (C)",
        aggfunc="mean"
```

```
    )
    .reindex(columns=month_order)
)

plt.figure(figsize=(14, 6))

sns.heatmap(
    pivot,
    cmap="RdYlBu_r",          # intuitive: blue=cold, red=hot
    annot=True,
    fmt=".1f",
    linewidths=0.4,
    linecolor="white",
    cbar_kws={"label": "Avg Temperature (°C)"}
)

plt.title("Average Monthly Temperature by Year", fontsize=15)
plt.xlabel("Month")
plt.ylabel("Year")
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```

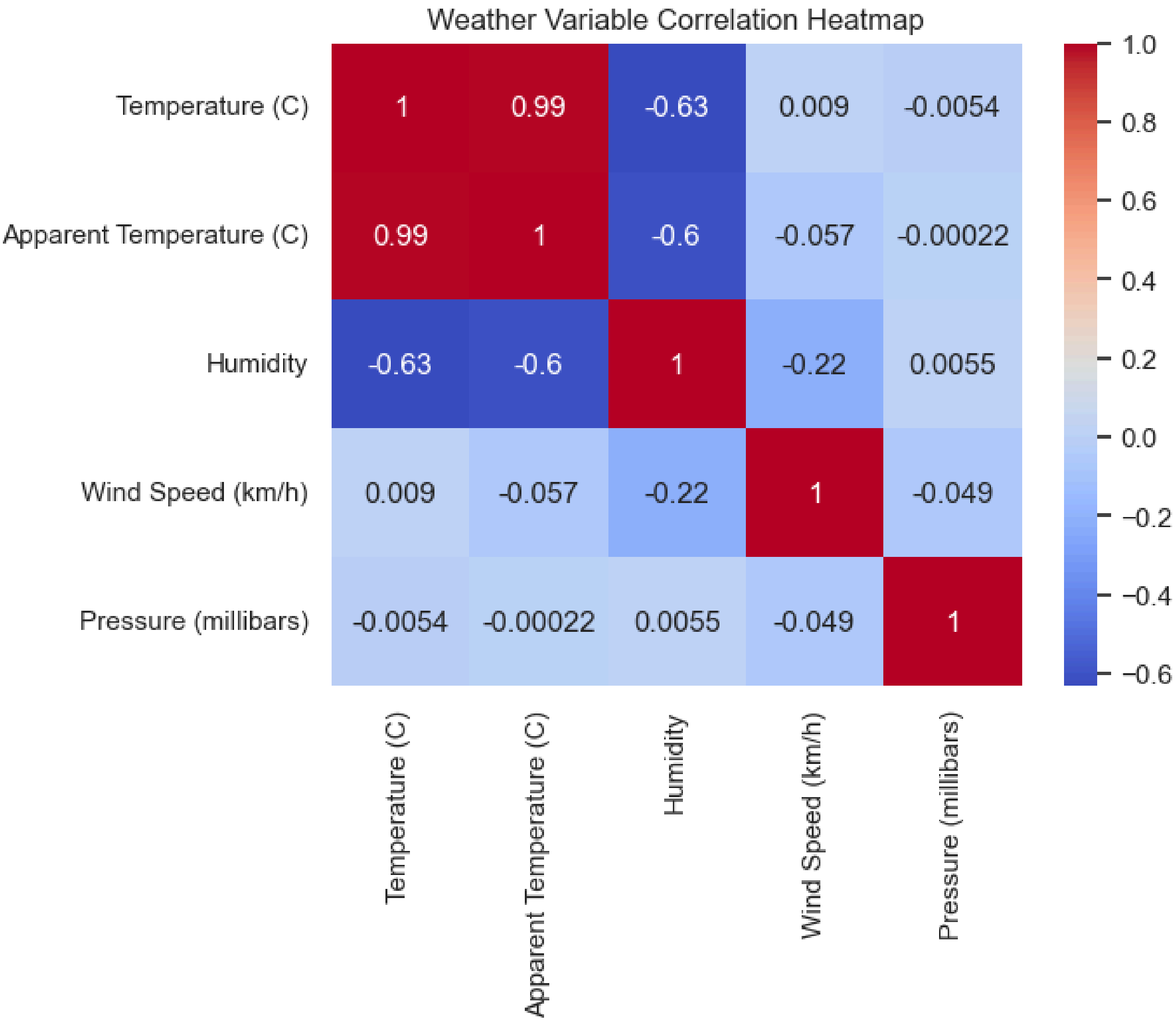



**Caption: -** shows warming/cooling patterns at a glance

### Correlation Heatmap (Core Weather Metrics) (EDA 10)

In [17]:
```
corr_cols = [
    "Temperature (C)",
    "Apparent Temperature (C)",
    "Humidity",
    "Wind Speed (km/h)",
    "Pressure (millibars)"
]

sns.heatmap(
    df[corr_cols].corr(),
    annot=True,
    cmap="coolwarm"
)
```

```
plt.title("Weather Variable Correlation Heatmap")
plt.show()
```




**Caption: -** Illustrates relationships among key weather variables.

## 🟦 STATISTICAL ANALYSIS

### Key Metrics

- Mean, median, minimum, and maximum temperature
- Monthly and yearly average temperatures
- Correlation between temperature and humidity
- Correlation between temperature and wind speed

### Interpretation

Statistical analysis indicates variability in temperature across seasons, with noticeable differences between actual and apparent temperature. Correlation results suggest meaningful relationships between temperature, humidity, and wind speed, reflecting expected atmospheric behavior.

---

In [18]:

```
df[corr_cols].describe()
```

Out[18]:

| | Temperature (C) | Apparent Temperature (C) | Humidity | Wind Speed (km/h) | Pressure (millibars) |
|---|---|---|---|---|---|
| **count** | 96453.000000 | 96453.000000 | 96453.000000 | 96453.000000 | 96453.000000 |
| **mean** | 11.932678 | 10.855029 | 0.734899 | 10.810640 | 1003.235956 |
| **std** | 9.551546 | 10.696847 | 0.195473 | 6.913571 | 116.969906 |
| **min** | -21.822222 | -27.716667 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 4.688889 | 2.311111 | 0.600000 | 5.828200 | 1011.900000 |
| **50%** | 12.000000 | 12.000000 | 0.780000 | 9.965900 | 1016.450000 |
| **75%** | 18.838889 | 18.838889 | 0.890000 | 14.135800 | 1021.090000 |
| **max** | 39.905556 | 39.344444 | 1.000000 | 63.852600 | 1046.380000 |

In [19]:

```
df.groupby("Year")["Temperature (C)"].mean()
```

Out[19]:

```
Year
2005     0.577778
2006    11.215225
2007    12.134677
2008    12.161819
2009    12.269682
2010    11.200176
2011    11.524934
2012    11.986824
2013    11.941017
2014    12.528228
2015    12.312088
2016    11.987381
Name: Temperature (C), dtype: float64
```

## 🟦 EXPORT VISUALS FROM NOTEBOOK

In [21]:

```
from pathlib import Path
from src.weather_trends_analysis.visualization import *

output_dir = Path("../visualizations/weather")

plot_temperature_trend(df, output_dir)
plot_monthly_average_temperature(df, output_dir)
plot_humidity_distribution(df, output_dir)
plot_actual_vs_apparent_temperature(df, output_dir)
plot_weather_summary_frequency(df, output_dir)
plot_weather_correlation_heatmap(df, output_dir)
plot_yearly_avg_temperature(df, output_dir)
plot_pressure_vs_temperature(df, output_dir)
plot_wind_speed_distribution(df, output_dir)
plot_temperature_heatmap(df, output_dir)

print("✅ All the visualizations exported successfully")
```

```
✅ All the visualizations exported successfully
```

---

# Key Findings

- Temperature exhibits clear seasonal and long-term variation
- Apparent temperature often deviates from actual temperature due to humidity and wind effects
- Certain weather conditions occur significantly more frequently
- Humidity tends to show a negative correlation with temperature
- Wind speed has a moderate relationship with temperature changes

---

# Climate Insights

1. Long-term temperature trends suggest gradual warming across years
2. Seasonal patterns strongly influence monthly temperature averages
3. Humidity plays a critical role in perceived temperature
4. Weather conditions are not uniformly distributed and show dominant patterns

---

# Recommendations

1. Monitor long-term temperature trends for climate planning
2. Use seasonal insights for agricultural and energy forecasting
3. Incorporate humidity and wind metrics into weather prediction models
4. Apply data-driven insights to environmental and urban planning

---

# Conclusion & Future Scope

## Conclusion

This project demonstrates how structured weather data analysis can uncover meaningful climate patterns and relationships. By combining statistical analysis with clear visualizations, raw meteorological data is transformed into actionable climate intelligence.

## Future Scope

- Predictive weather modeling
- Extreme weather event detection
- Climate anomaly analysis
- Integration with real-time weather APIs

---

**End of Report**