Weather Prediction

In [29]:
```
# import the libraries
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt

from scipy.stats import zscore
from sklearn.preprocessing import RobustScaler
```

In [30]:
```
# Load the dataset
df = pd.read_excel('W Data.xlsx')
df
```

Out[30]:

	country	location_name	timezone	last_updated	temperature_celsius	condition_text	wind_kp
0	India	New Delhi	Asia/Nicosia	2023-08-29 15:00:00	34.0	Mist	6.
1	India	New Delhi	Asia/Nicosia	2023-08-30 08:30:00	29.0	Mist	11.
2	India	New Delhi	Asia/Nicosia	2023-08-31 05:15:00	29.0	Mist	3.
3	India	New Delhi	Asia/Nicosia	2023-09-01 05:15:00	29.0	Mist	6.
4	India	New Delhi	Asia/Nicosia	2023-09-02 05:00:00	31.3	Clear	12.
...	...	...	...	...	...	...	.
91	India	New Delhi	Asia/Nicosia	2023-12-02 00:45:00	20.8	Partly cloudy	6.
92	India	New Delhi	Asia/Nicosia	2023-12-04 02:00:00	18.0	Mist	3.
93	India	New Delhi	Asia/Nicosia	2023-12-06 01:15:00	14.0	Mist	3.
94	India	New Delhi	Asia/Nicosia	2023-12-07 01:15:00	16.0	Mist	3.
95	India	New Delhi	Asia/Nicosia	2023-12-08 01:00:00	16.0	Mist	3.

96 rows × 14 columns

In [31]:
```
# it give information about the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96 entries, 0 to 95
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   country              96 non-null     object
 1   location_name        96 non-null     object
 2   timezone             96 non-null     object
 3   last_updated         96 non-null     datetime64[ns]
 4   temperature_celsius  96 non-null     float64
 5   condition_text       96 non-null     object
 6   wind_kph             96 non-null     float64
 7   wind_degree          96 non-null     int64
 8   wind_direction       96 non-null     object
 9   pressure_in          96 non-null     float64
 10  humidity             96 non-null     int64
 11  cloud                96 non-null     int64
 12  feels_like_celsius   96 non-null     float64
 13  visibility_km        96 non-null     float64
dtypes: datetime64[ns](1), float64(5), int64(3), object(5)
memory usage: 10.6+ KB
```

In [32]:
```
# check the missing values
df.isnull().sum()
```

Out[32]:
```
country                0
location_name          0
timezone               0
last_updated           0
temperature_celsius    0
condition_text         0
wind_kph               0
wind_degree            0
wind_direction         0
pressure_in            0
humidity               0
cloud                  0
feels_like_celsius     0
visibility_km          0
dtype: int64
```

In [33]:
```
# check duplicated values
df[df.duplicated()]
```

Out[33]:

country	location_name	timezone	last_updated	temperature_celsius	condition_text	wind_kph	w

In [34]:
```
# drop null values
df.dropna()
```

Out[34]:

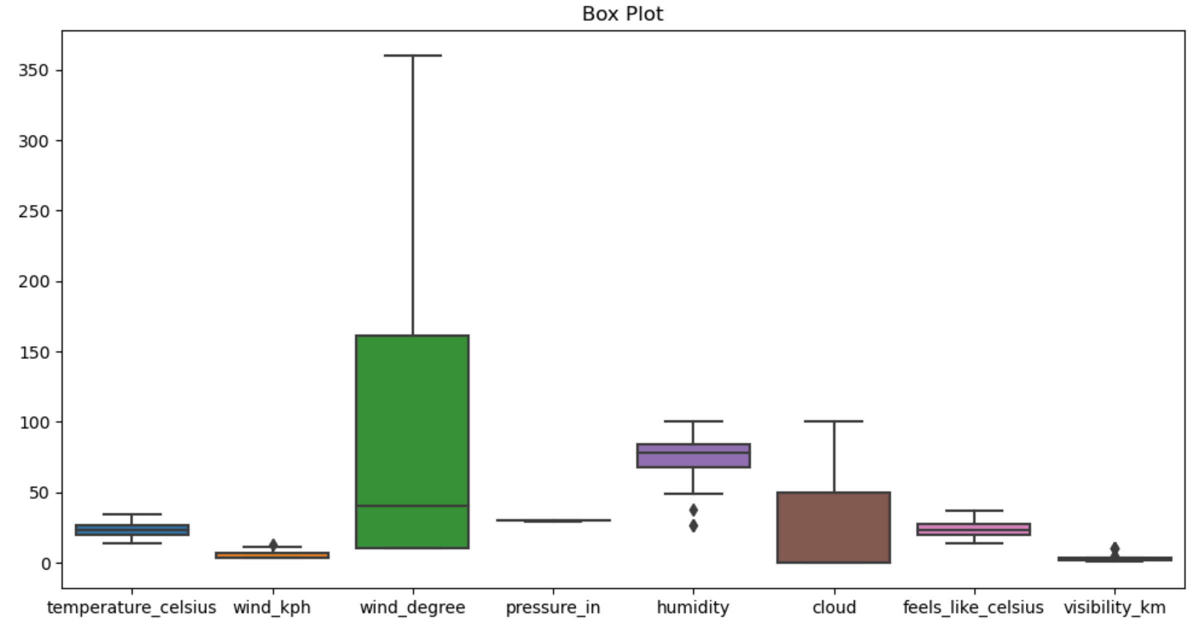
	country	location_name	timezone	last_updated	temperature_celsius	condition_text	wind_kp
0	India	New Delhi	Asia/Nicosia	2023-08-29 15:00:00	34.0	Mist	6.
1	India	New Delhi	Asia/Nicosia	2023-08-30 08:30:00	29.0	Mist	11.
2	India	New Delhi	Asia/Nicosia	2023-08-31 05:15:00	29.0	Mist	3.
3	India	New Delhi	Asia/Nicosia	2023-09-01 05:15:00	29.0	Mist	6.
4	India	New Delhi	Asia/Nicosia	2023-09-02 05:00:00	31.3	Clear	12.
...	...	...	...	...	...	...	.
91	India	New Delhi	Asia/Nicosia	2023-12-02 00:45:00	20.8	Partly cloudy	6.
92	India	New Delhi	Asia/Nicosia	2023-12-04 02:00:00	18.0	Mist	3.
93	India	New Delhi	Asia/Nicosia	2023-12-06 01:15:00	14.0	Mist	3.
94	India	New Delhi	Asia/Nicosia	2023-12-07 01:15:00	16.0	Mist	3.
95	India	New Delhi	Asia/Nicosia	2023-12-08 01:00:00	16.0	Mist	3.

96 rows × 14 columns

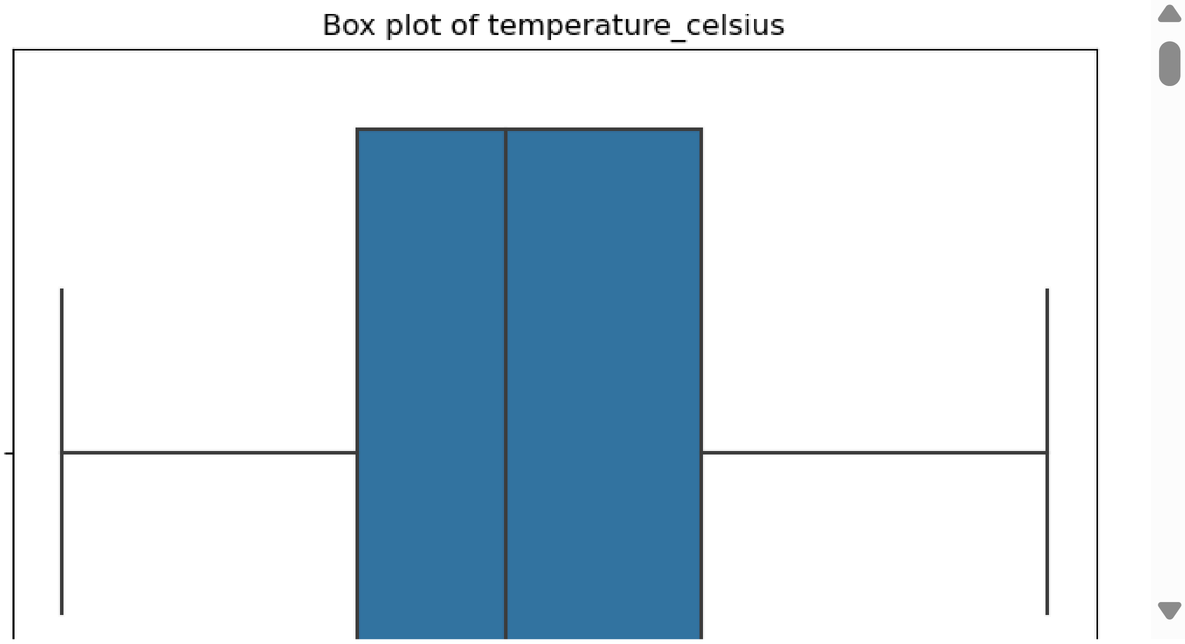
Exploratory Data Analysis

In [35]:
```
# only numerical columns
numerical_columns = ['temperature_celsius', 'wind_kph', 'wind_degree', 'pressure_
                     'humidity', 'cloud', 'feels_like_celsius',
                     'visibility_km']
```

In [36]:
```
# Box plot : Detect the outliers
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[numerical_columns])
plt.title("Box Plot")
plt.show()
```



In [37]:
```
# Box plots
for column in numerical_columns:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=df[column])
    plt.title(f'Box plot of {column}')
    plt.xlabel(column)
    plt.show()
```



Descriptive statistics

In [38]:
```
# Assuming your DataFrame is named 'df'
# You can use df.describe() for numerical columns and df[column].value_counts() f
print(df.describe())  # Summary statistics for numerical columns
```

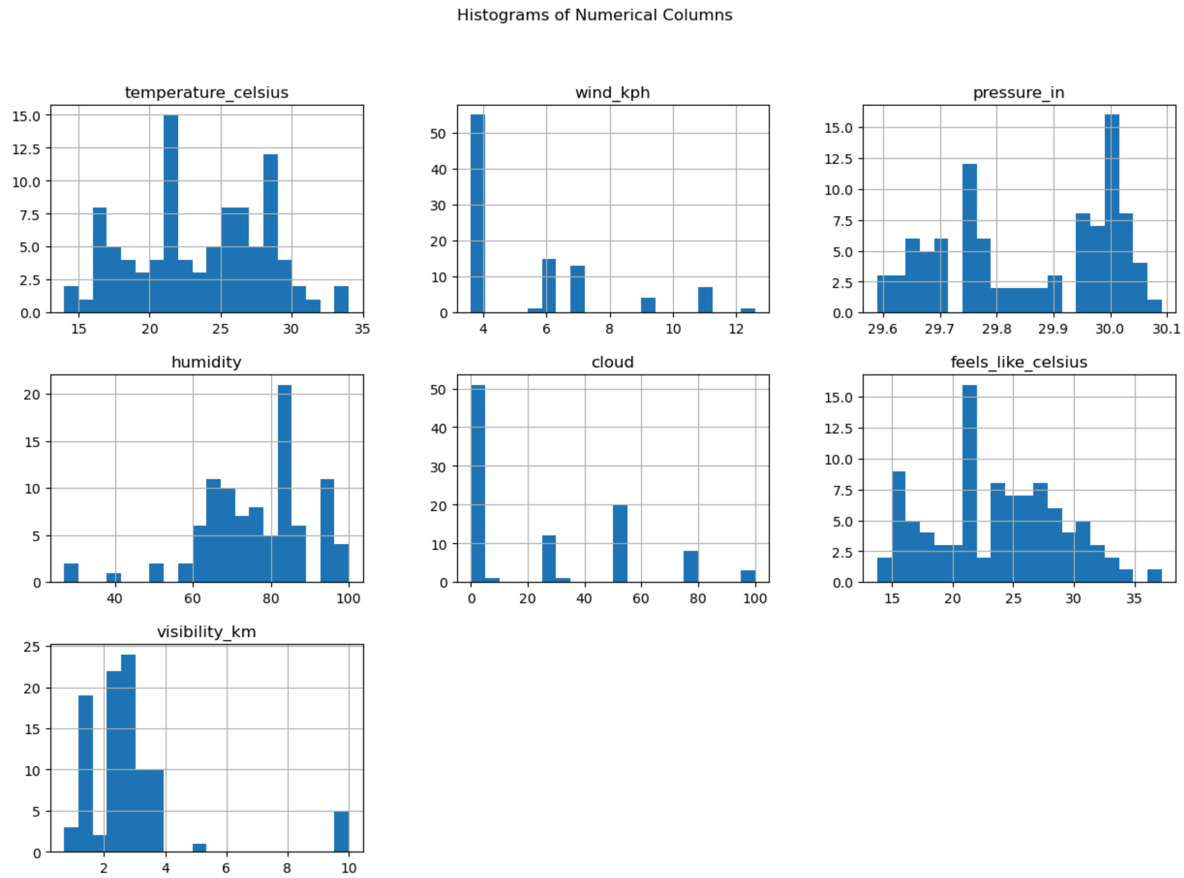
```
       temperature_celsius   wind_kph  wind_degree  pressure_in   humidity  \
count            96.000000  96.000000    96.000000    96.000000  96.000000
mean             23.044792   5.336458    97.239583    29.854167  76.218750
std               4.666360   2.391657   109.834971     0.148109  14.327157
min              14.000000   3.600000    10.000000    29.590000  27.000000
25%              20.000000   3.600000    10.000000    29.740000  67.750000
50%              23.000000   3.600000    40.000000    29.880000  78.000000
75%              27.000000   6.800000   161.500000    30.000000  84.000000
max              34.000000  12.600000   360.000000    30.090000 100.000000

            cloud  feels_like_celsius  visibility_km
count   96.000000           96.000000      96.000000
mean    23.354167           23.604167       2.884375
std     29.272577            5.283498       1.858031
min      0.000000           13.800000       0.700000
25%      0.000000           20.000000       2.025000
50%      0.000000           23.550000       2.800000
75%     50.000000           27.350000       3.200000
max    100.000000           37.200000      10.000000
```

In [ ]:

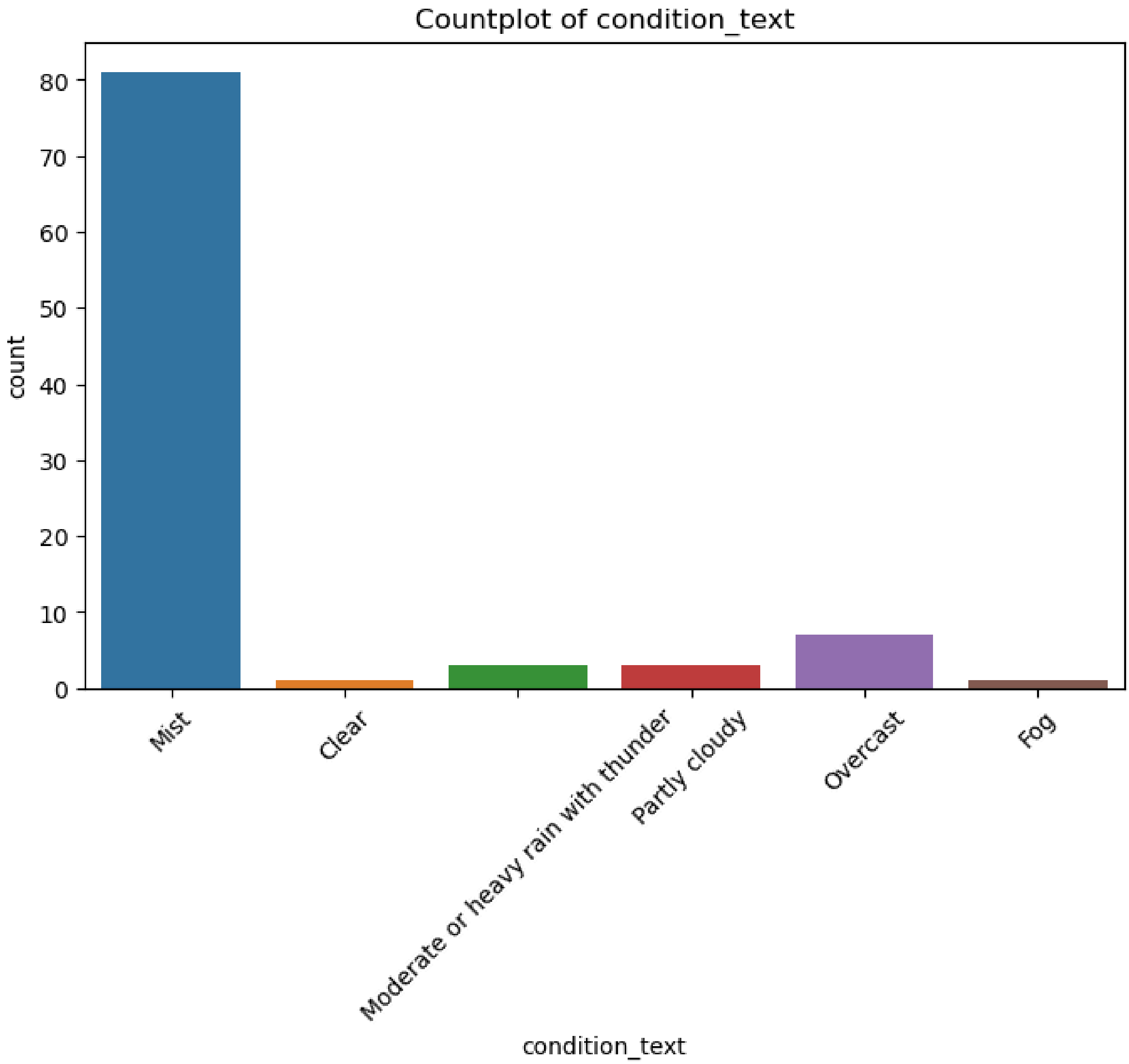
Check the distribution

In [39]:

```
# Histograms for numerical columns
numerical_columns = ['temperature_celsius', 'wind_kph', 'pressure_in',
                     'humidity', 'cloud', 'feels_like_celsius', 'visibility_km']
df[numerical_columns].hist(bins=20, figsize=(15, 10))
plt.suptitle('Histograms of Numerical Columns')
plt.show()
```



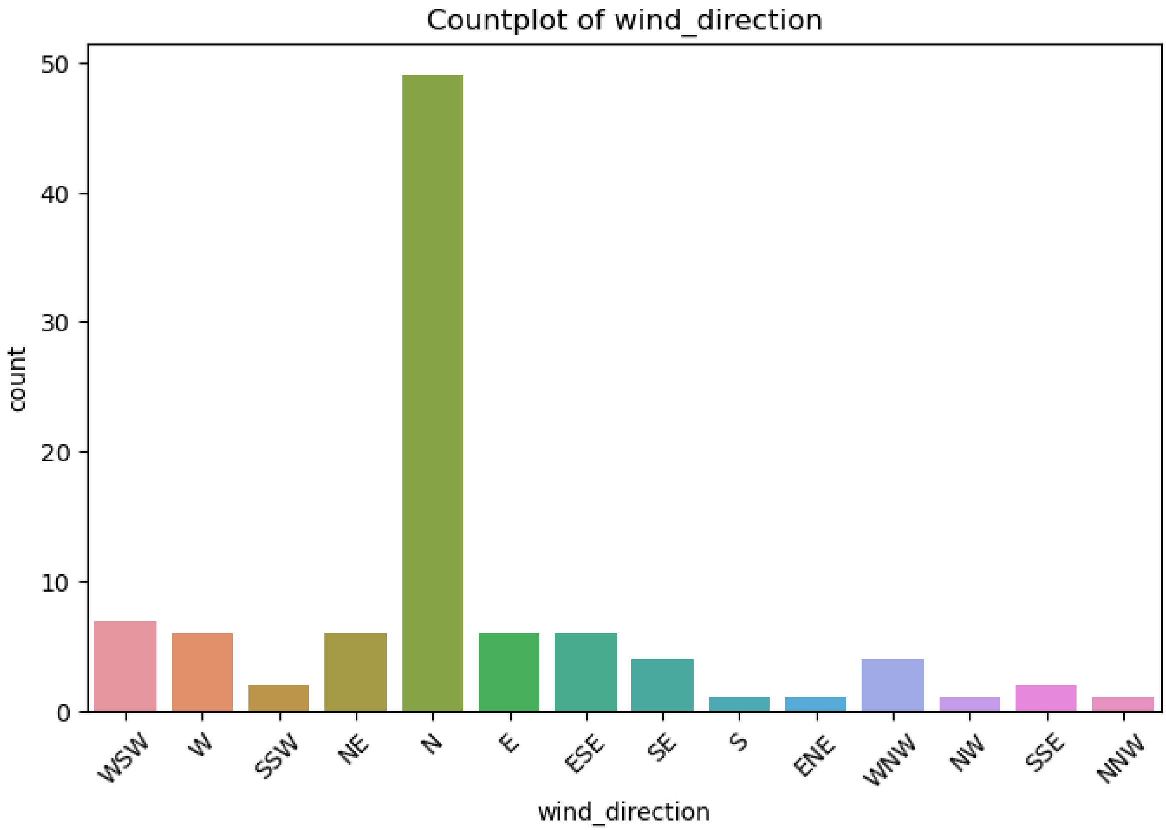
In [40]:
```
# Bar plots for categorical columns
categorical_columns = ['condition_text']
for column in categorical_columns:
    plt.figure(figsize=(8, 5))
    sns.countplot(data=df, x=column)
    plt.title(f'Countplot of {column}')
    plt.xticks(rotation=45)  # Rotate x-axis labels for better readability if ne
    plt.show()
```



Interpretation :

    From the above figure we can observe that the condition text is high in
    mist condition.

In [41]:
```
# Bar plots for categorical columns
categorical_columns = ['wind_direction']
for column in categorical_columns:
    plt.figure(figsize=(8, 5))
    sns.countplot(data=df, x=column)
    plt.title(f'Countplot of {column}')
    plt.xticks(rotation=45)  # Rotate x-axis labels for better readability if nee
    plt.show()
```

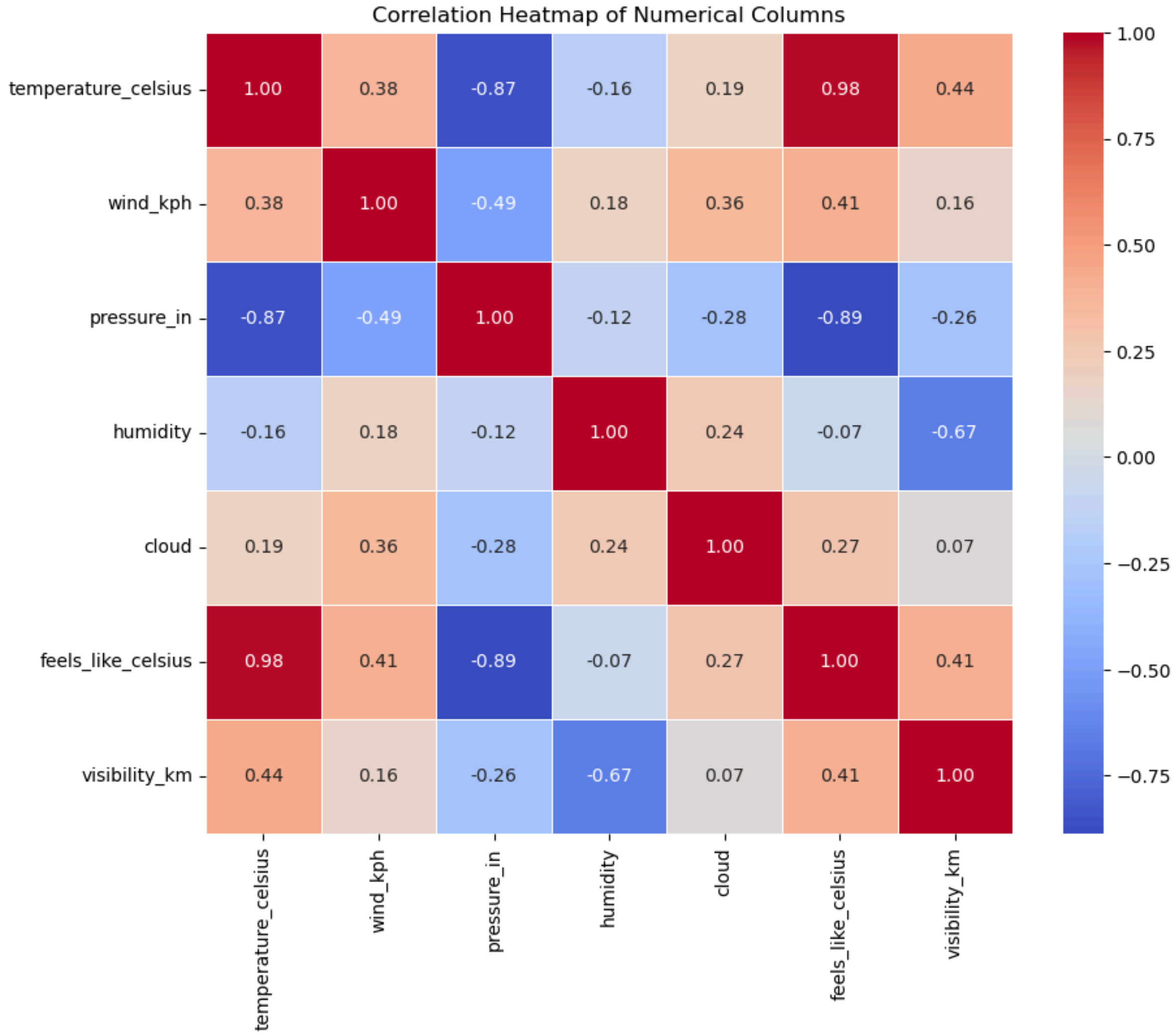


Interpretation :

    From the above figure we can observe that the wind direction is high in
    North direction.

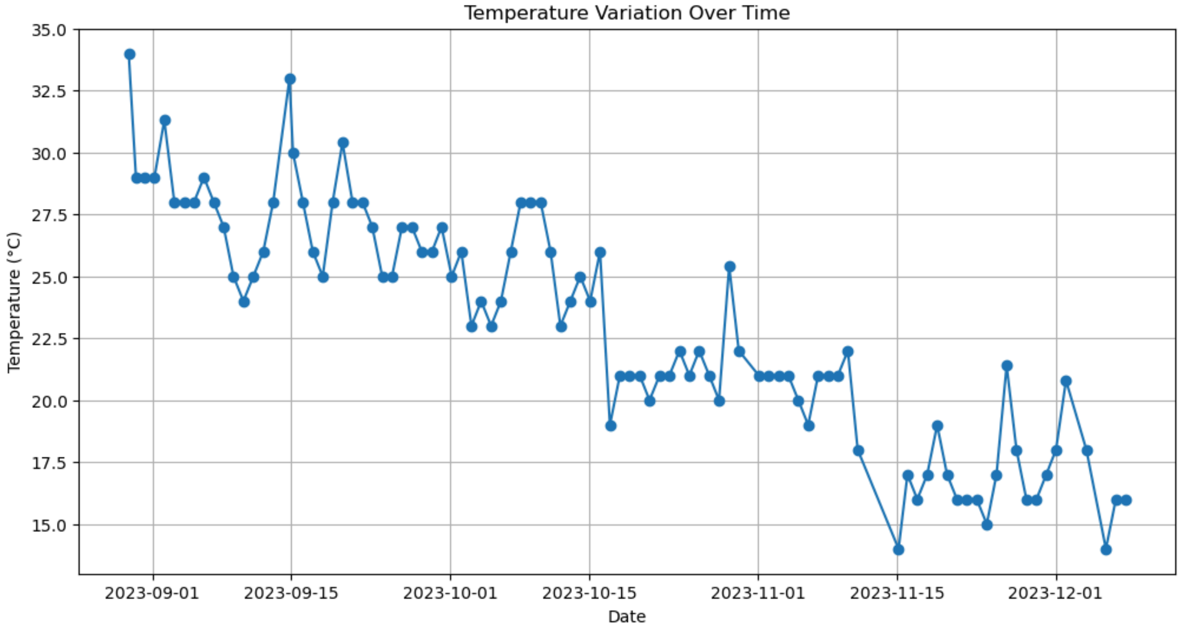
Correlation Heatmap

In [42]:
```
# Correlation matrix and heatmap for numerical columns
correlation_matrix = df[numerical_columns].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidt
plt.title('Correlation Heatmap of Numerical Columns')
plt.show()
```



Interpretation : From the above heatmap results we can see temperature_celsius correlated with feels_like_celsius is highly correlated with each other.

In [43]:
```
# Assuming 'last_updated' is a timestamp column
df['last_updated'] = pd.to_datetime(df['last_updated'])
plt.figure(figsize=(12, 6))
plt.plot(df['last_updated'], df['temperature_celsius'], marker='o', linestyle='-
plt.title('Temperature Variation Over Time')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.show()
```



Interpretation :

From the above figure we can observe that the tempuratue is decrease.

Ordinal Logistic regression

Ordinal logistic regression is used to model the relationship between an ordered multilevel dependent variable and independent variables. In the modeling, values of the dependent variable have a natural order or ranking.

In [44]:
```
pip install mord
```

```
Requirement already satisfied: mord in c:\users\shweta\anaconda3\lib\site-packa
ges (0.7)
Note: you may need to restart the kernel to use updated packages.
```

Coding on condition text column

Clear = 0

Fog = 1

Here we use variable selection method for better result, we check which column condition text is related to this column by using R programming. Then we got 4 columns 'visibility_km', 'pressure_in', 'cloud', 'wind_kph'.

In [45]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from mord import LogisticAT
from sklearn.metrics import accuracy_score

# Load your dataset (assuming it's already loaded into weather_data DataFrame)
weather_data = pd.read_excel('coding(W data).xlsx')

# Define independent variables (X) and dependent variable (y)
X = weather_data[['visibility_km', 'pressure_in', 'cloud', 'wind_kph']]
y = weather_data['condition_text']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Initialize the ordinal logistic regression model
model = LogisticAT()

# Train the model
model.fit(X_train, y_train)

# Predict the target variable
y_pred = model.predict(X_test)

# Calculate accuracy
acc1 = accuracy_score(y_test, y_pred)
print("testing Accuracy:", acc1)
```

```
testing Accuracy: 0.8
```

Interpretation :

Accuracy of ordinal logistic regression is 0.8%.

In [46]:
```
from sklearn.metrics import confusion_matrix, classification_report

# Draw confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Calculate evaluation measures
eval_report = classification_report(y_test, y_pred)
print("Evaluation Measures:")
print(eval_report)
```

```
Confusion Matrix:
[[ 0  0  0  1  0]
 [ 0 15  0  0  0]
 [ 0  1  0  0  0]
 [ 0  1  0  0  1]
 [ 0  0  0  0  1]]
Evaluation Measures:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           2       0.88      1.00      0.94        15
           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         2
           5       0.50      1.00      0.67         1

    accuracy                           0.80        20
   macro avg       0.28      0.40      0.32        20
weighted avg       0.69      0.80      0.74        20


C:\Users\Shweta\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Shweta\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Shweta\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [47]:
```
import pandas as pd
from mord import LogisticAT

# Load your trained model (assuming it's already trained and saved)
# model = LogisticAT.load('your_model_path.pkl')

# Prepare input data for prediction
new_data = pd.DataFrame({
    'visibility_km': [10.5],  # Example visibility in kilometers
    'pressure_in': [29.5],      # Example pressure in inches
    'cloud': [3],               # Example cloud cover level
    'wind_kph': [15.0]          # Example wind speed in kilometers per hour
})

# Make predictions
predictions = model.predict(new_data)

# Print predictions
print("Predicted weather condition:", predictions)
```

```
Predicted weather condition: [3]
```

Interpretation :

```
From the above code, if we enter the values of visibility_km, pressure_i
n, cloud, wind_kph in our mind, then this model give us the weather cond
ition.
i.e.  Moderate or heavy rain with thunder = 3
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: