```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import binom
```

```
! gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.cs
To: /content/bike_sharing.csv
100% 648k/648k [00:00<00:00, 966kB/s]
```

```
df = pd.read_csv('/content/bike_sharing.csv')
```

```
df.head()
```

| | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | casual | register |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2011-01-01 00:00:00 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 81 | 0.0 | 3 | |
| **1** | 2011-01-01 01:00:00 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 8 | |

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.shape
```

```
(10886, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  object
 1   season      10886 non-null  int64
 2   holiday     10886 non-null  int64
 3   workingday  10886 non-null  int64
 4   weather     10886 non-null  int64
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

```
df.describe()
```

| | season | holiday | workingday | weather | temp | atemp | humidity |
|---|---|---|---|---|---|---|---|
| **count** | 10886.000000 | 10886.000000 | 10886.000000 | 10886.000000 | 10886.00000 | 10886.000000 | 10886.000000 |
| **mean** | 2.506614 | 0.028569 | 0.680875 | 1.418427 | 20.23086 | 23.655084 | 61.886460 |
| **std** | 1.116174 | 0.166599 | 0.466159 | 0.633839 | 7.79159 | 8.474601 | 19.245033 |
| **min** | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 0.82000 | 0.760000 | 0.000000 |
| **25%** | 2.000000 | 0.000000 | 0.000000 | 1.000000 | 13.94000 | 16.665000 | 47.000000 |
| **50%** | 3.000000 | 0.000000 | 1.000000 | 1.000000 | 20.50000 | 24.240000 | 62.000000 |
| **75%** | 4.000000 | 0.000000 | 1.000000 | 2.000000 | 26.24000 | 31.060000 | 77.000000 |
| **max** | 4.000000 | 1.000000 | 1.000000 | 4.000000 | 41.00000 | 45.455000 | 100.000000 |

b. identifying missing values

```
df.isnull().sum()
```

| | 0 |
|---|---|
| **datetime** | 0 |
| **season** | 0 |
| **holiday** | 0 |
| **workingday** | 0 |
| **weather** | 0 |
| **temp** | 0 |
| **atemp** | 0 |
| **humidity** | 0 |
| **windspeed** | 0 |
| **casual** | 0 |
| **registered** | 0 |
| **count** | 0 |

- *there is no missing value in the data set*

identifying dublicate values

```
df.duplicated().sum()
```

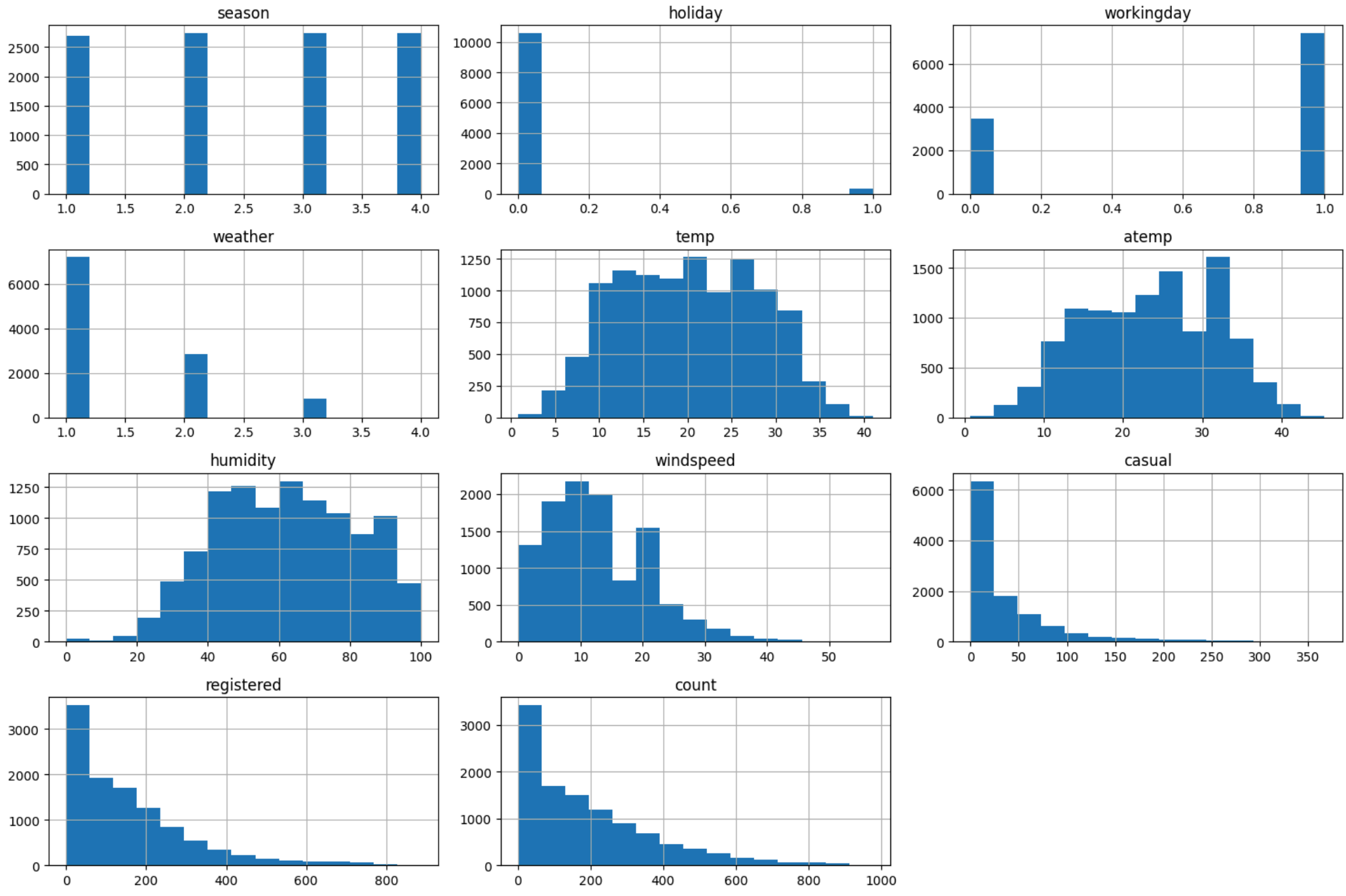
```
0
```

- *there is no dublicate values in the data set*

## d. Analyzing the distribution of Numerical & Categorical variables.

```
# Distribution of numerical variables
numerical_cols = df.select_dtypes(include=[np.number]).columns
print("\nNumerical columns:", numerical_cols)

df[numerical_cols].hist(bins=15, figsize=(15, 10), layout=(4, 3))
plt.tight_layout()
plt.show()
```

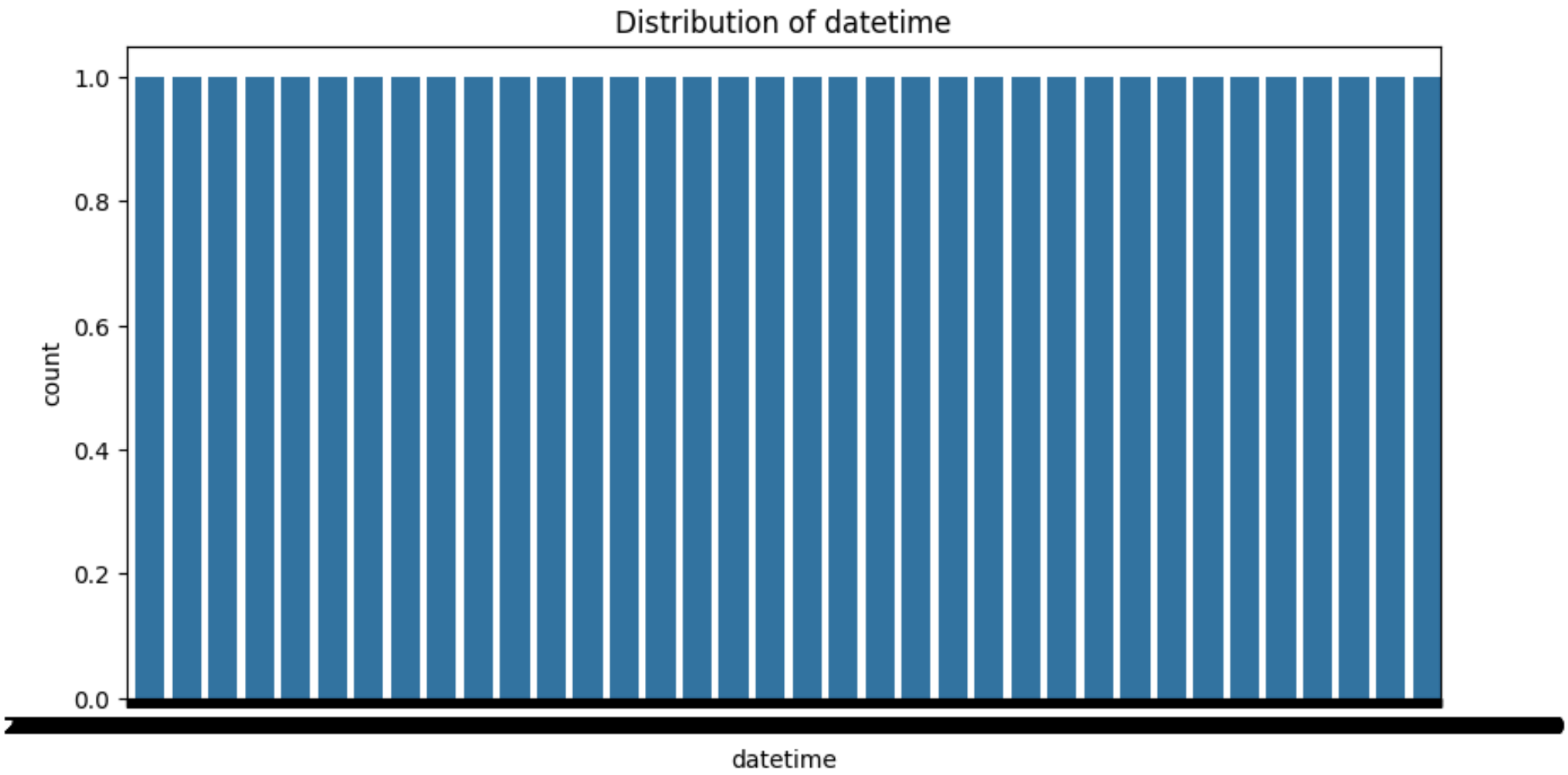
```
Numerical columns: Index(['season', 'holiday', 'workingday', 'weather', 'temp', 'atemp',
       'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```




```
# Distribution of categorical variables
categorical_cols = df.select_dtypes(exclude=[np.number]).columns
print("\nCategorical columns:", categorical_cols)

for col in categorical_cols:
    plt.figure(figsize=(10, 5))
```

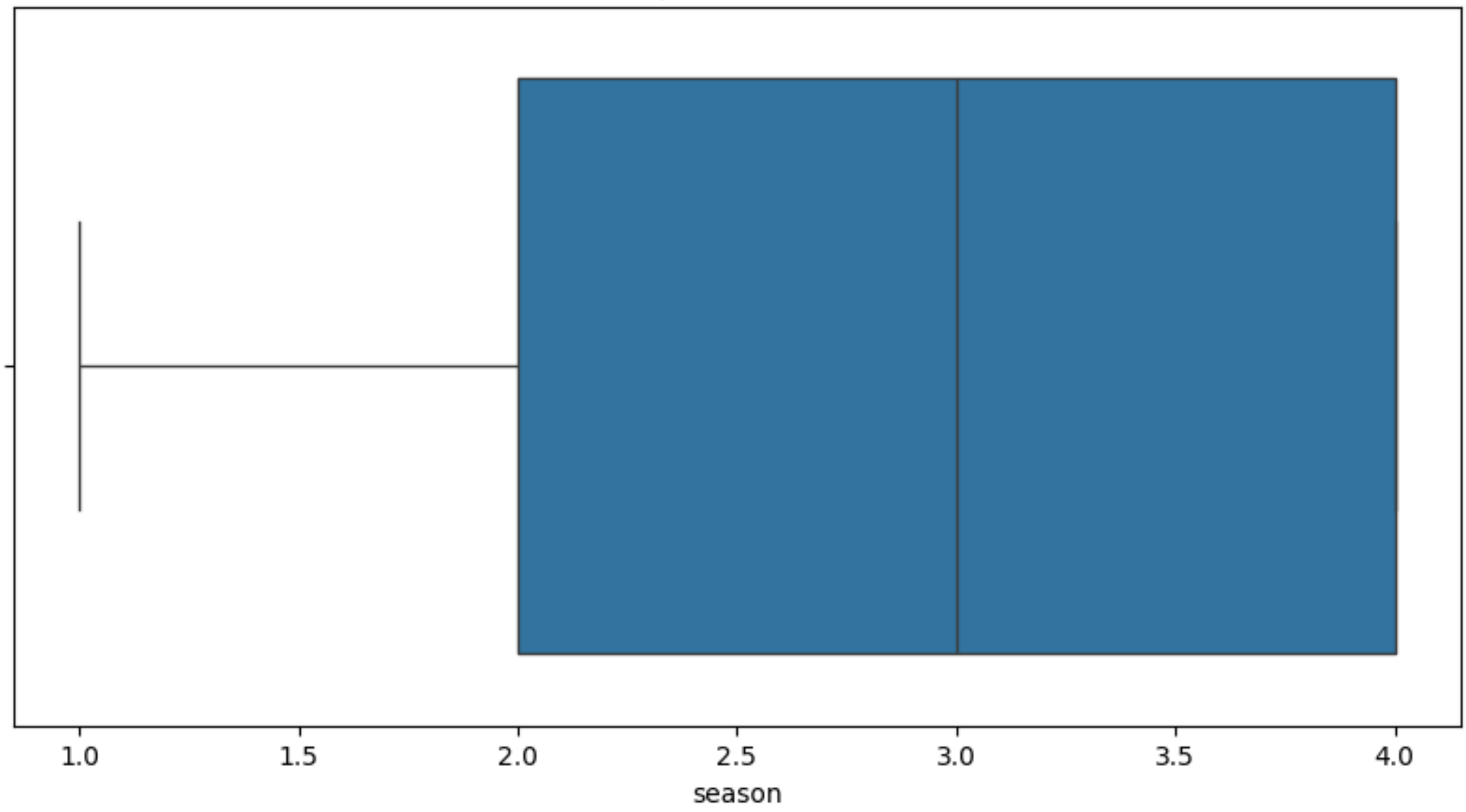
```
sns.countplot(x=col, data=df)
plt.title(f'Distribution of {col}')
plt.show()
```

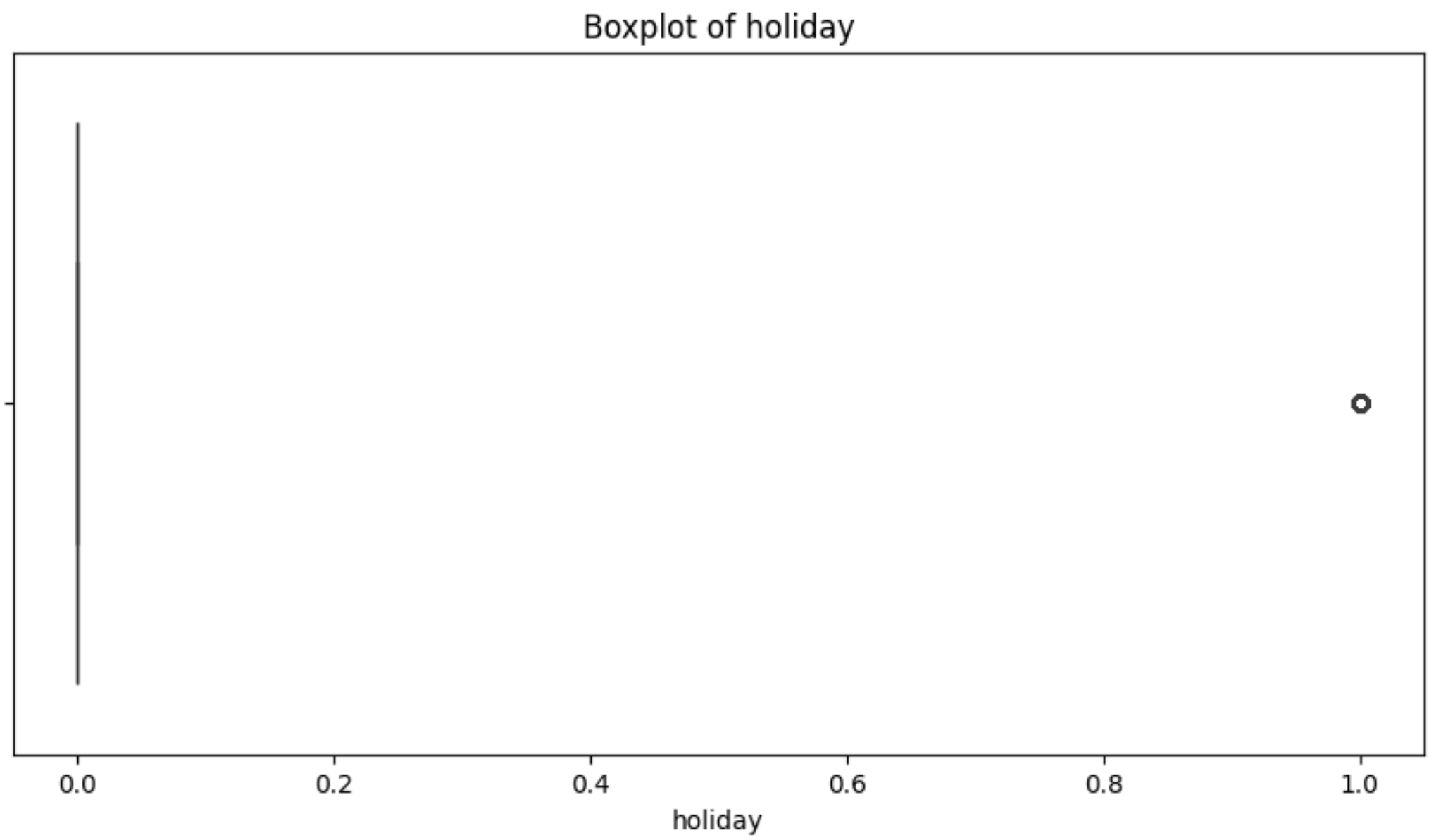
```
Categorical columns: Index(['datetime'], dtype='object')
```

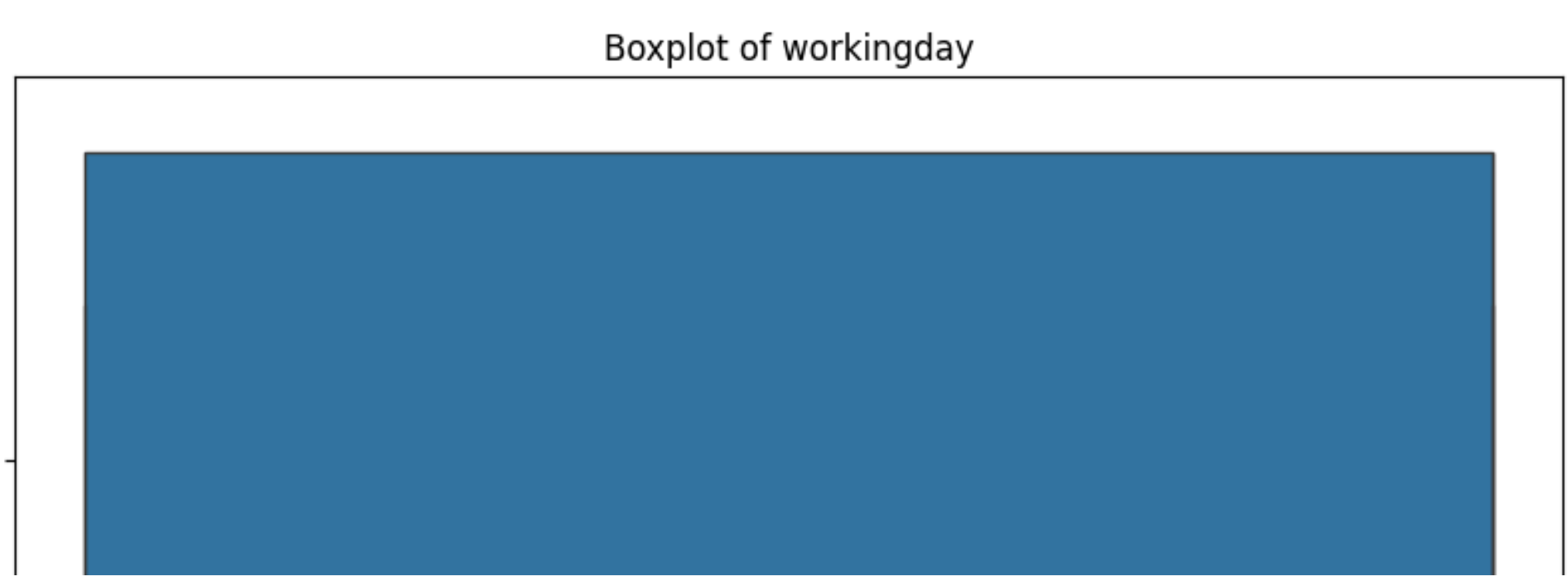


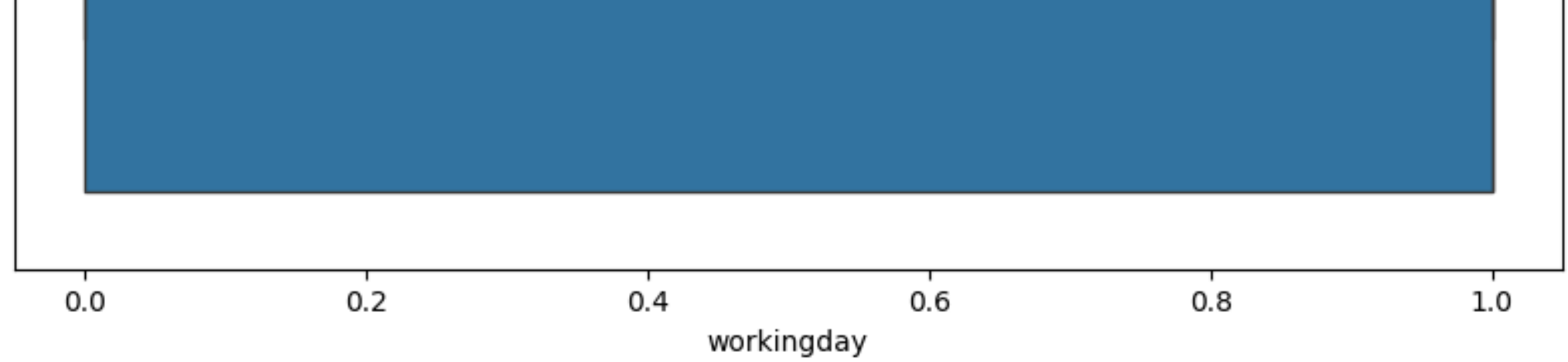

## e. Checking for Outliers and Handling Them Appropriately

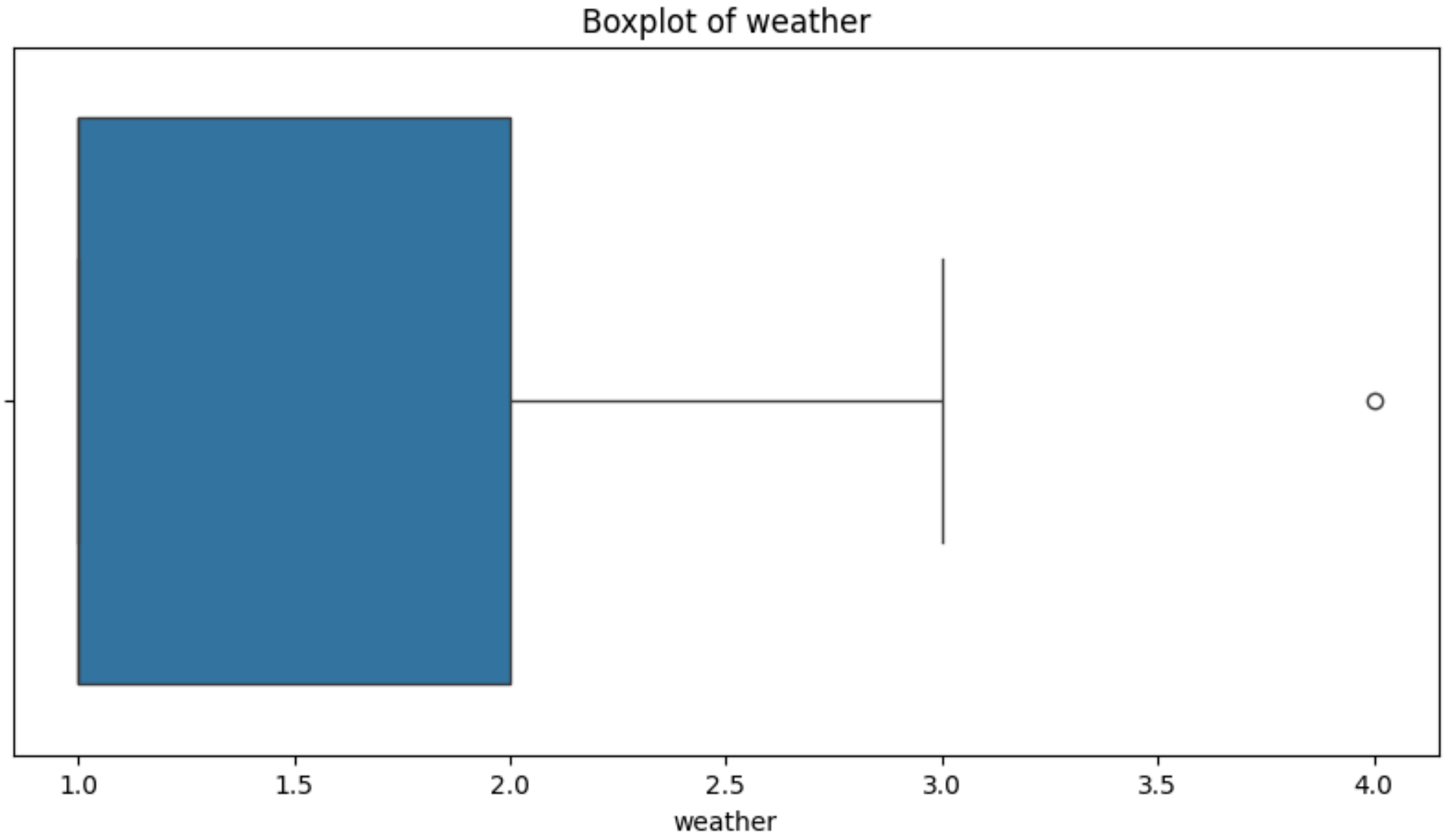
```
for col in numerical_cols:
    plt.figure(figsize=(10, 5))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}')
    plt.show()

#clipping them to the 1st and 99th percentile
for col in numerical_cols:
    q1 = df[col].quantile(0.01)
    q99 = df[col].quantile(0.99)
    df[col] = np.clip(df[col], q1, q99)

for col in numerical_cols:
    plt.figure(figsize=(10, 5))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col} after clipping')
    plt.show()
```

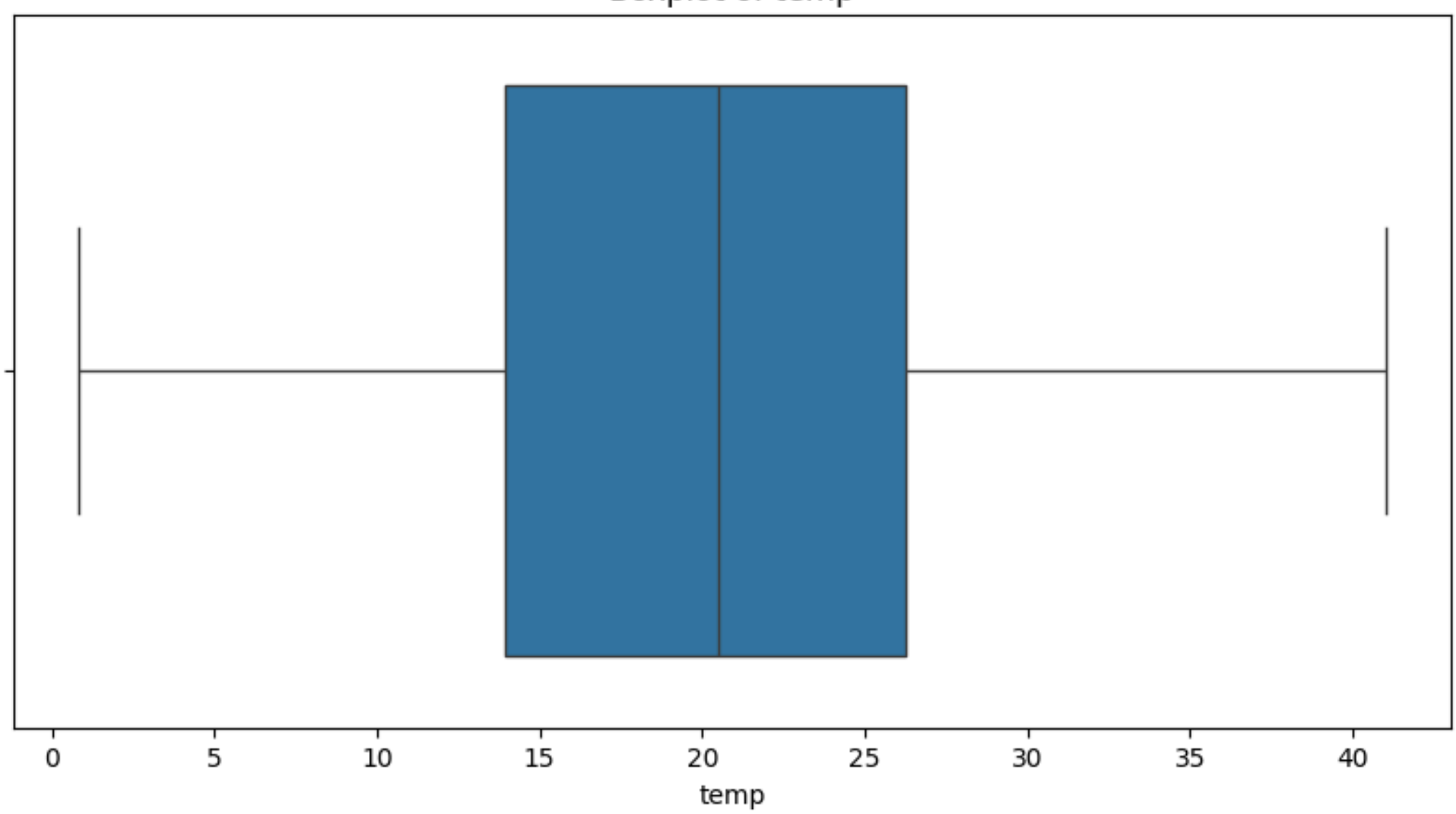

Boxplot of season
1.0
1.5
2.0
2.5
3.0
3.5
4.0
season

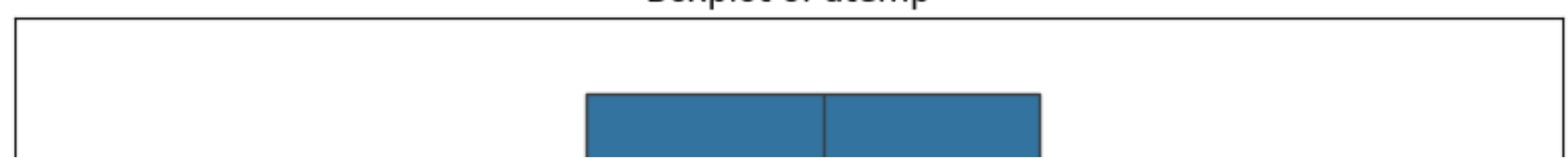

Boxplot of holiday
0.0
0.2
0.4
0.6
0.8
1.0
holiday

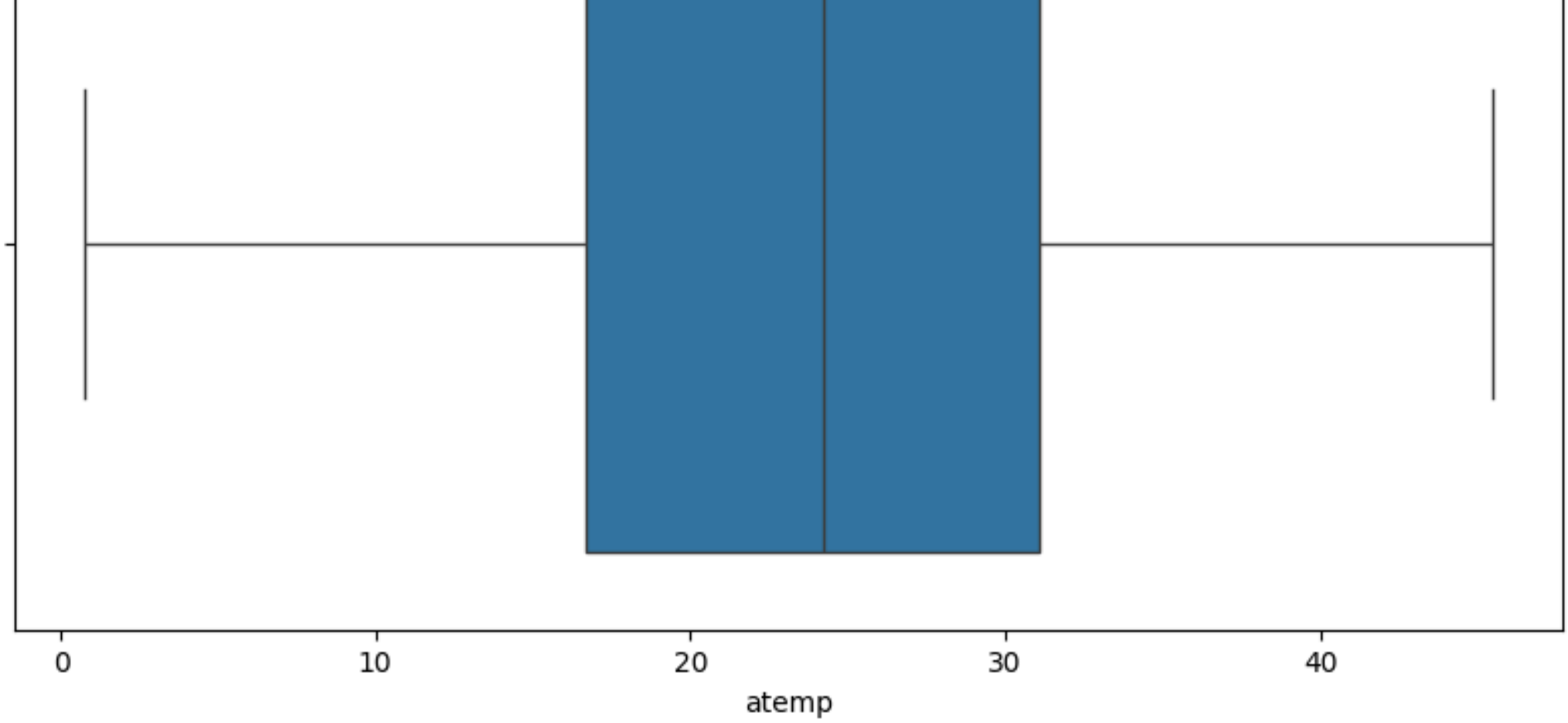

Boxplot of workingday


0.0
0.2
0.4
0.6
0.8
1.0
workingday

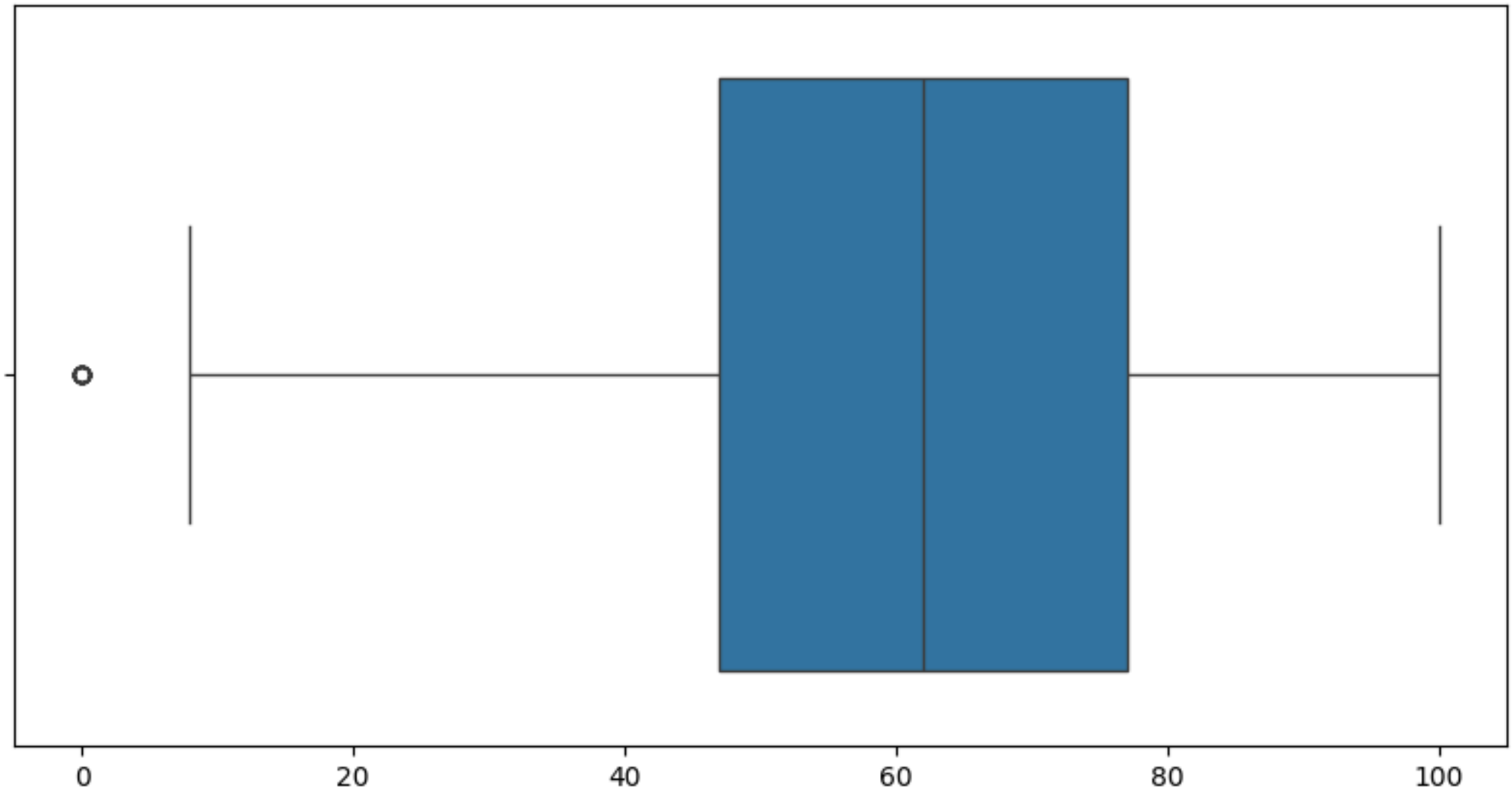

Boxplot of weather
1.0
1.5
2.0
2.5
3.0
3.5
4.0
weather


Boxplot of temp
0
5
10
15
20
25
30
35
40
temp

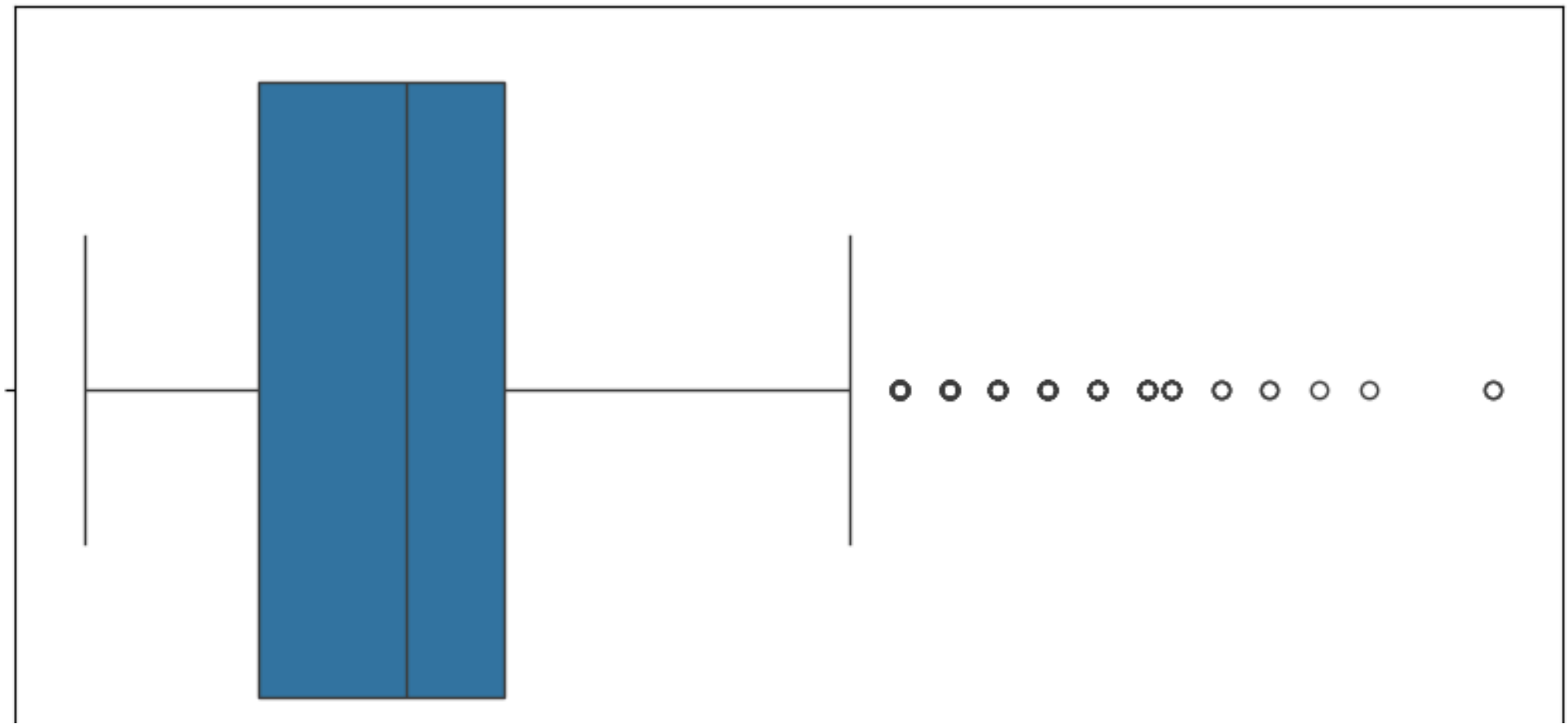

Boxplot of atemp

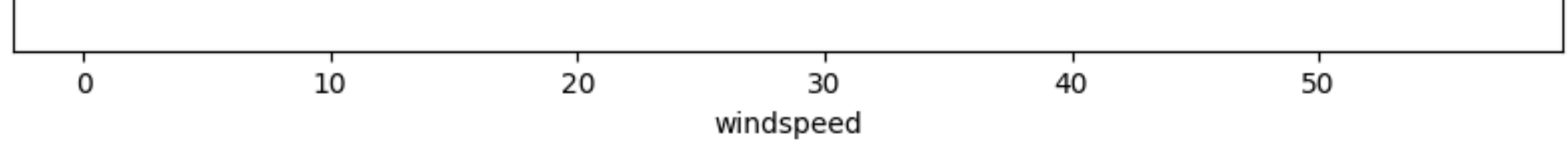


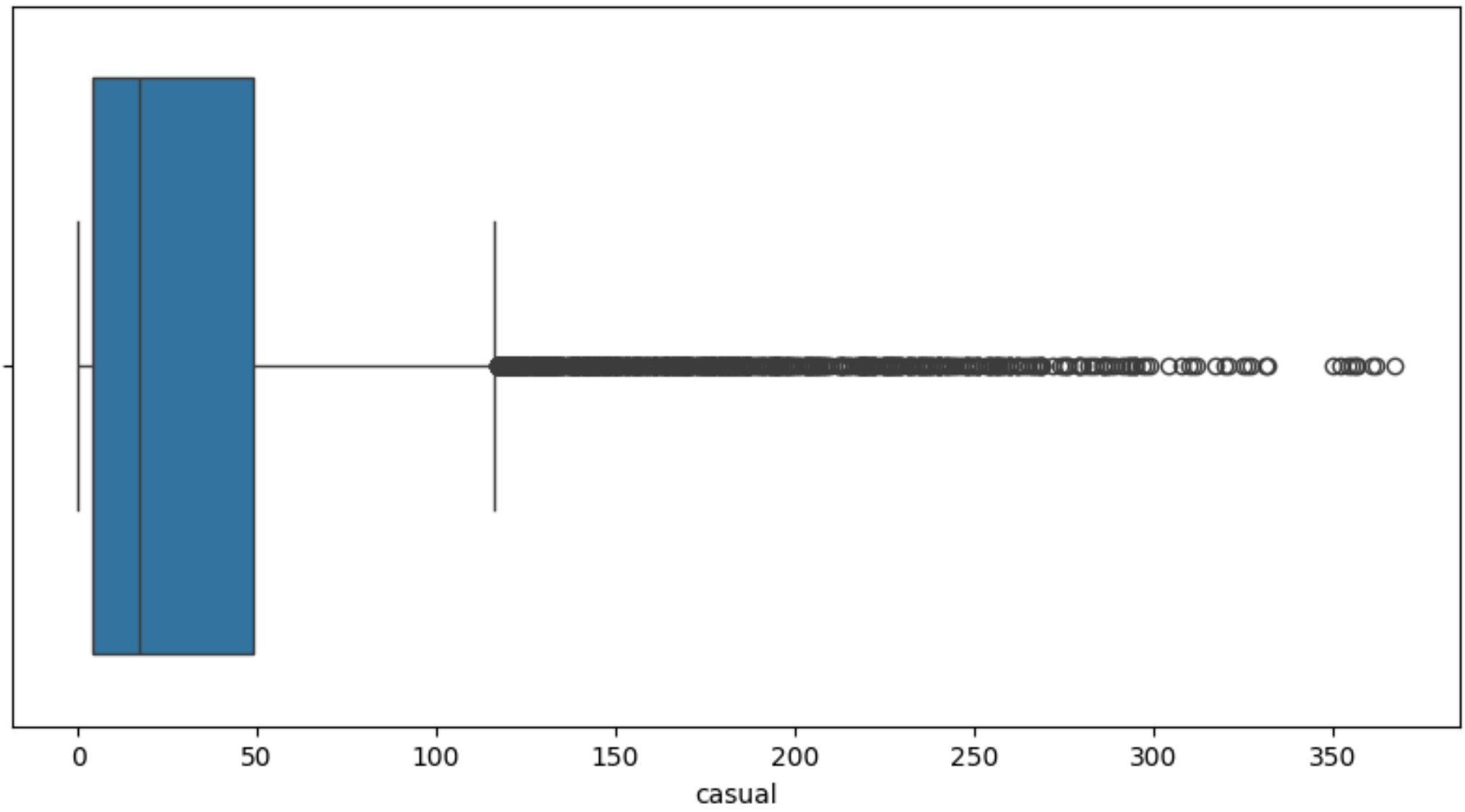

Boxplot of humidity

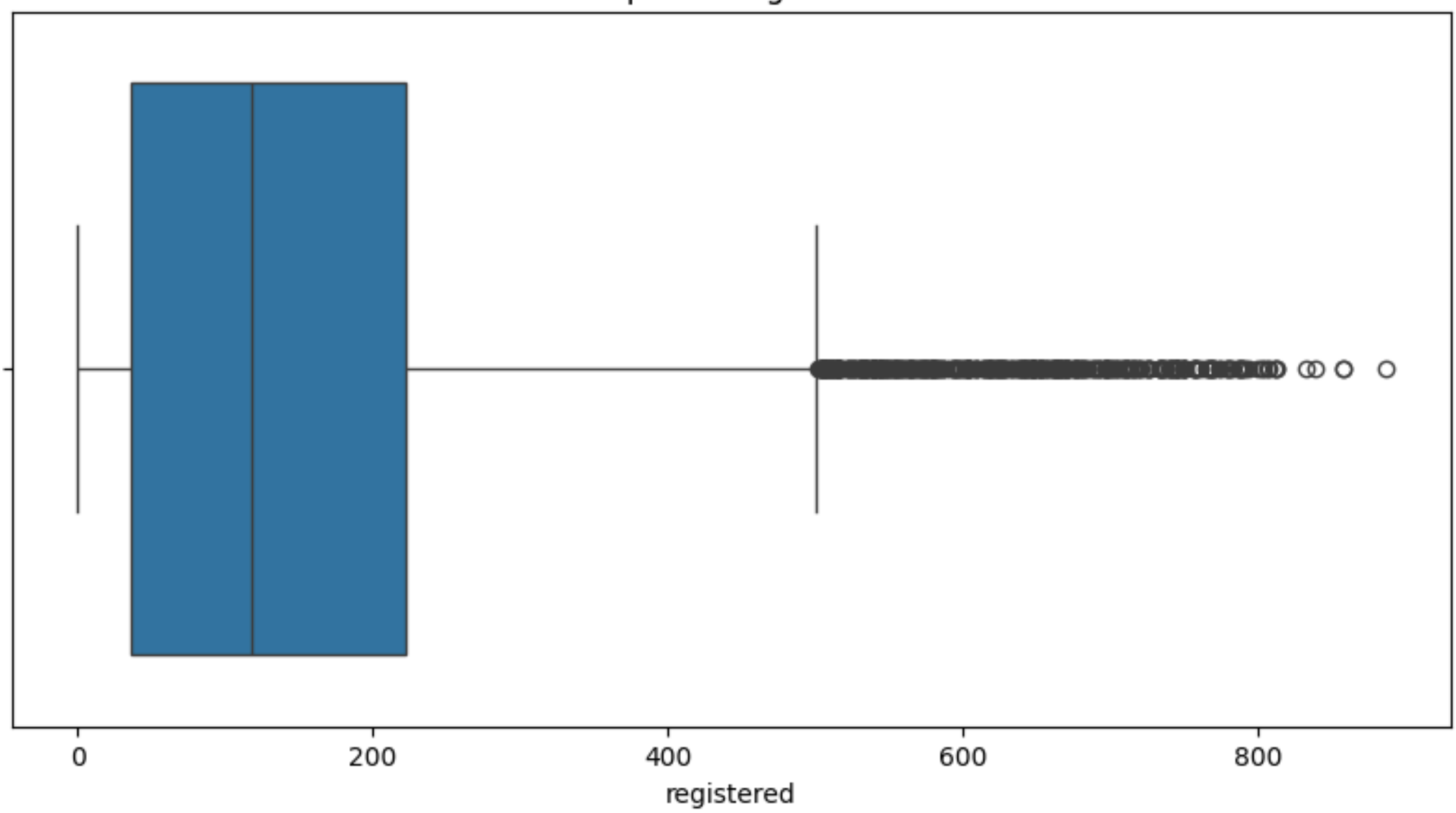


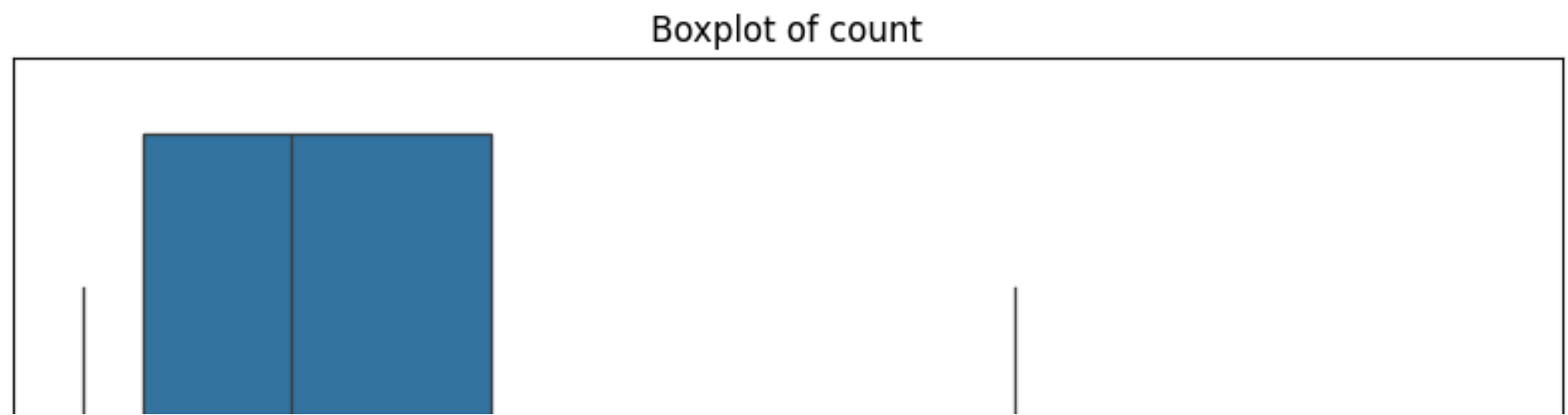

Boxplot of windspeed

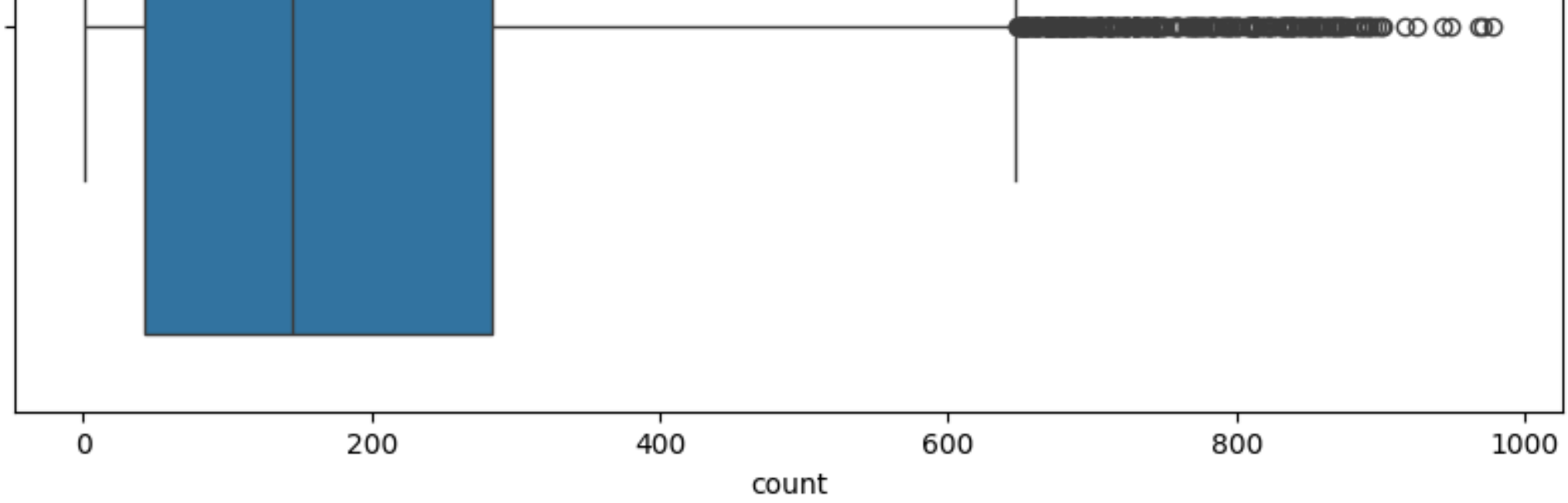



0
10
20
30
40
50
windspeed

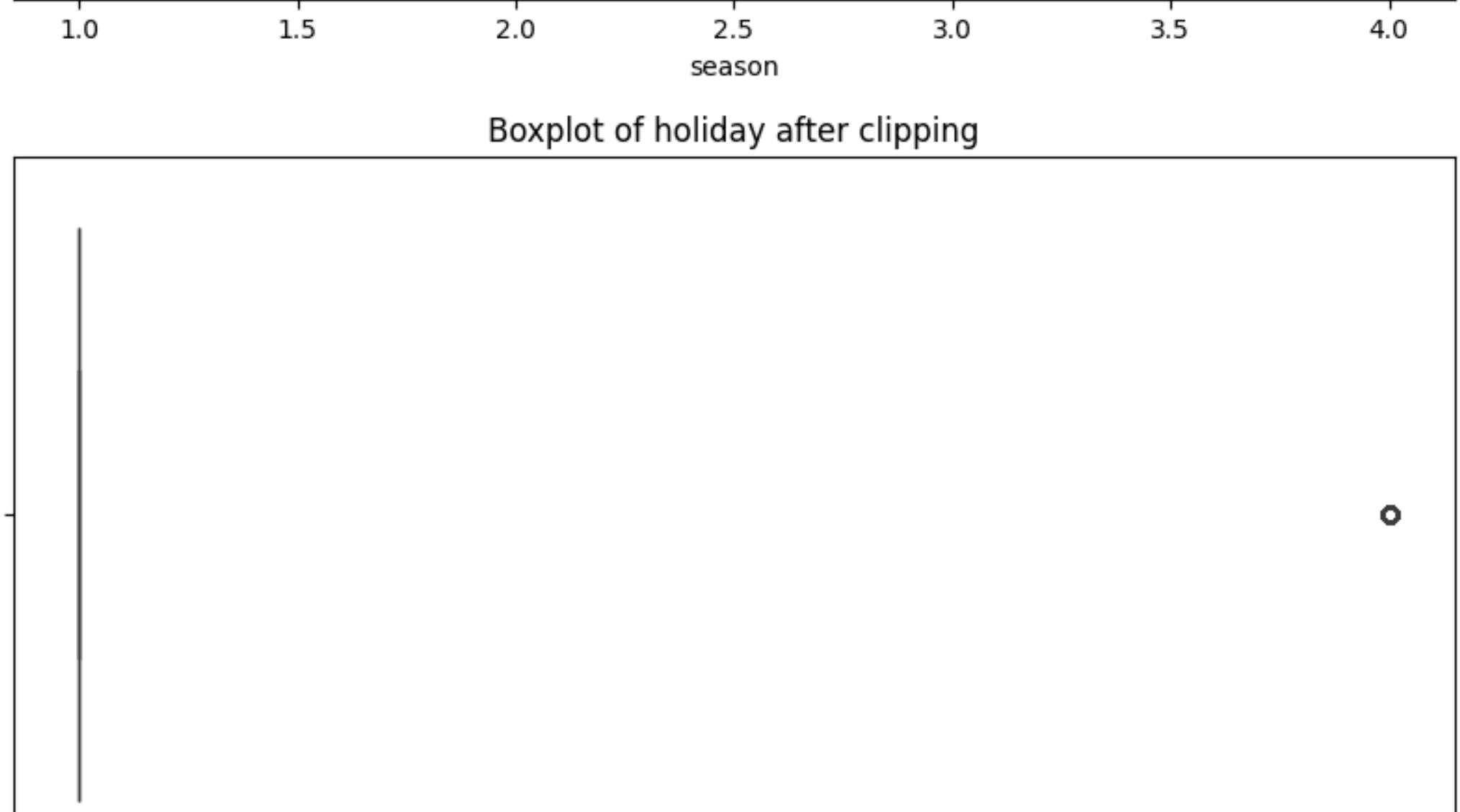

Boxplot of casual
0
50
100
150
200
250
300
350
casual


Boxplot of registered
0
200
400
600
800
registered


Boxplot of count



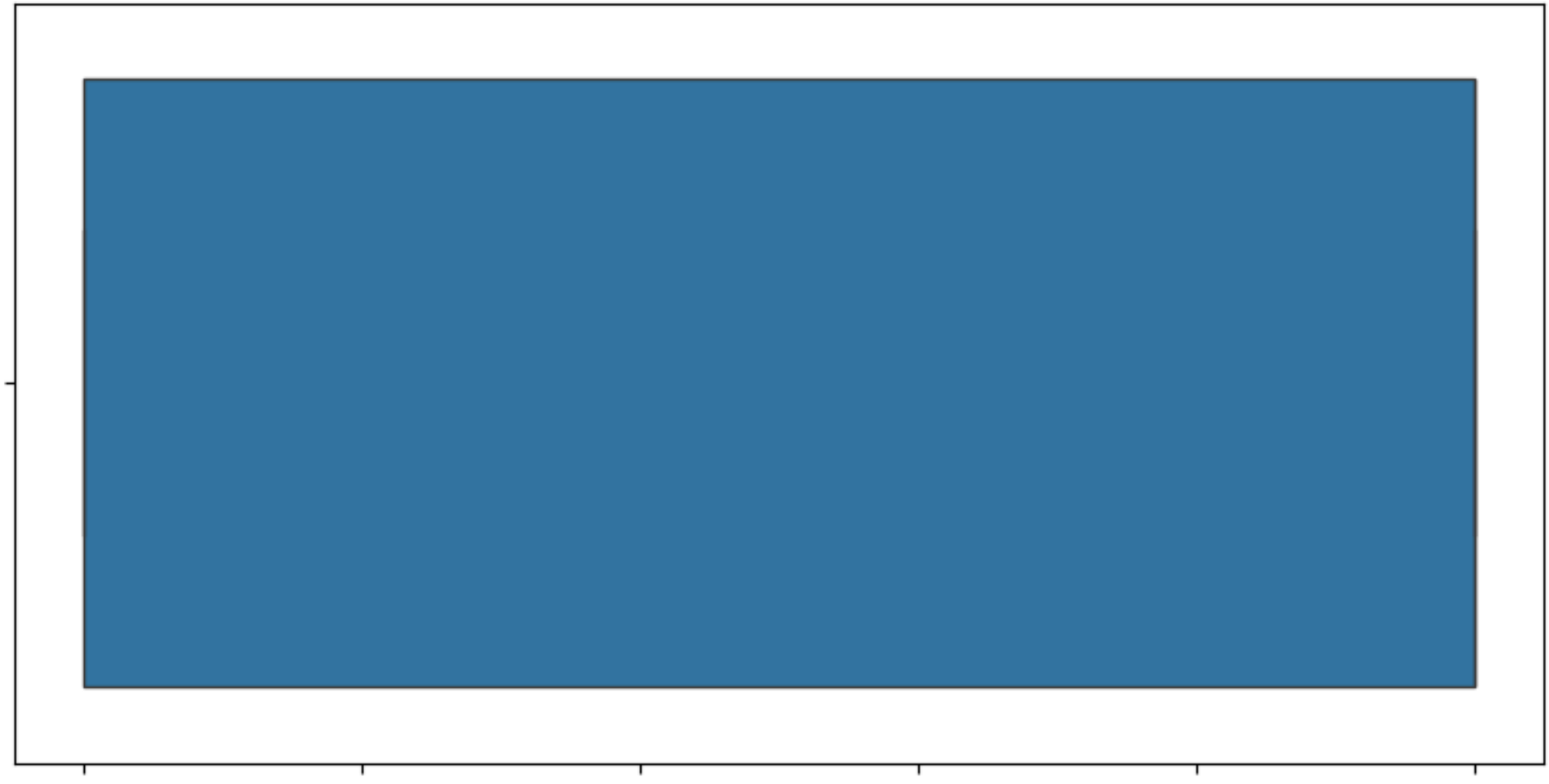

Boxplot of season after clipping

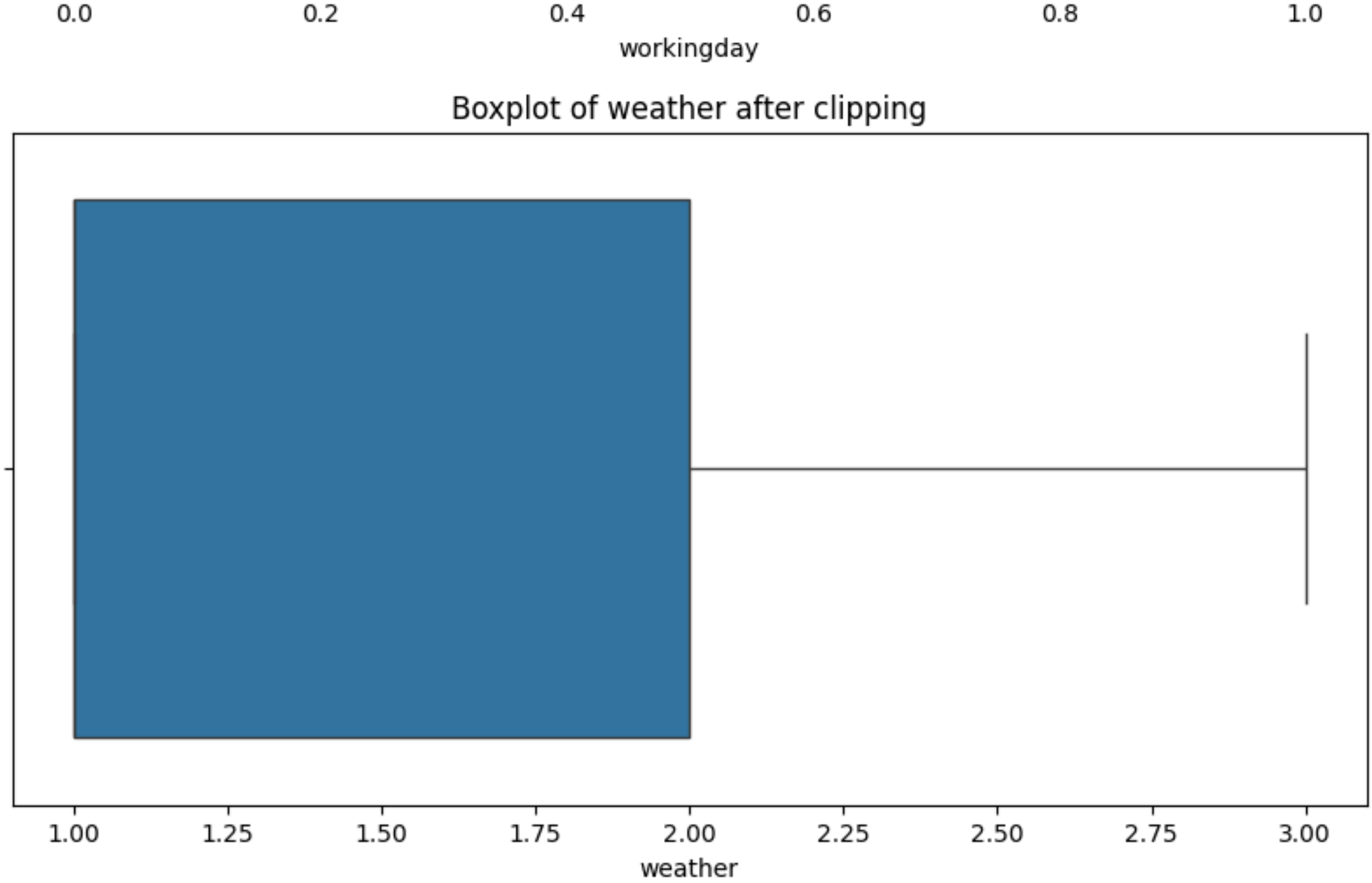


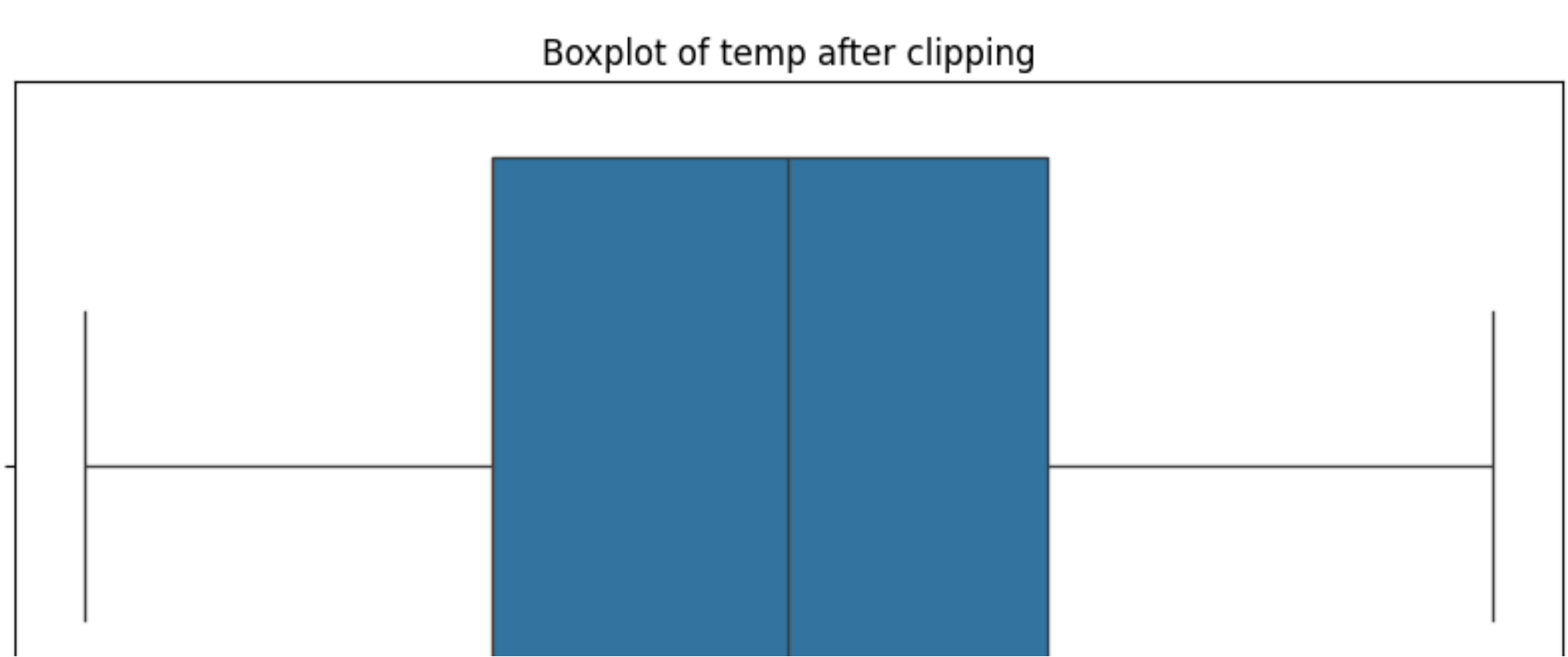

Boxplot of holiday after clipping

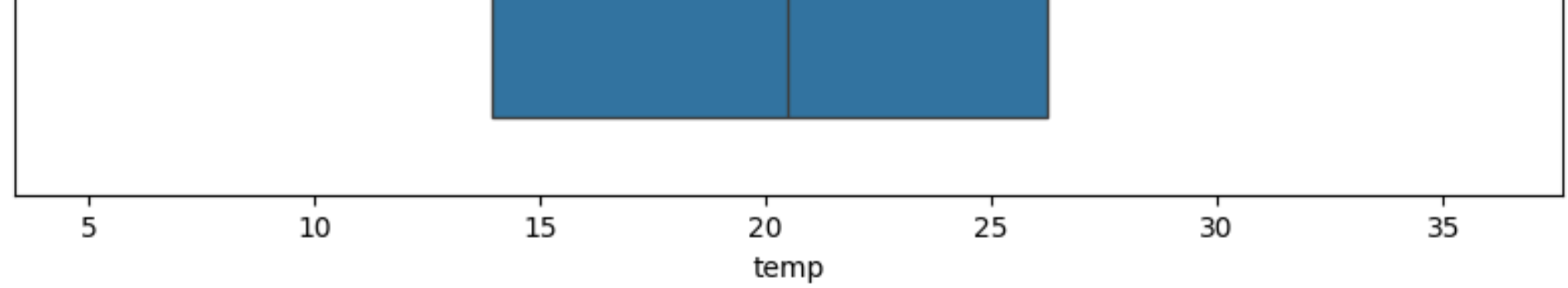


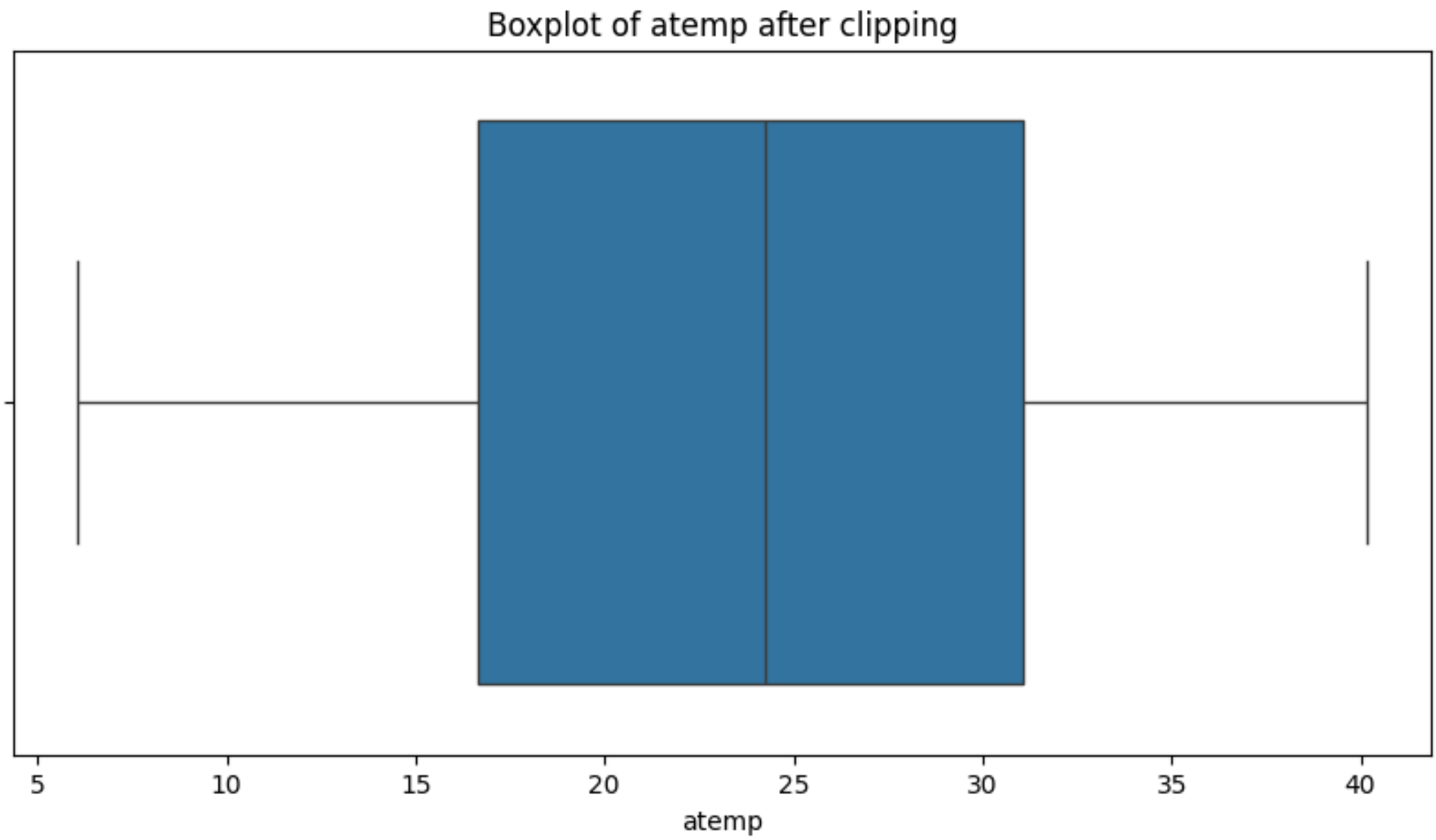

Boxplot of workingday after clipping

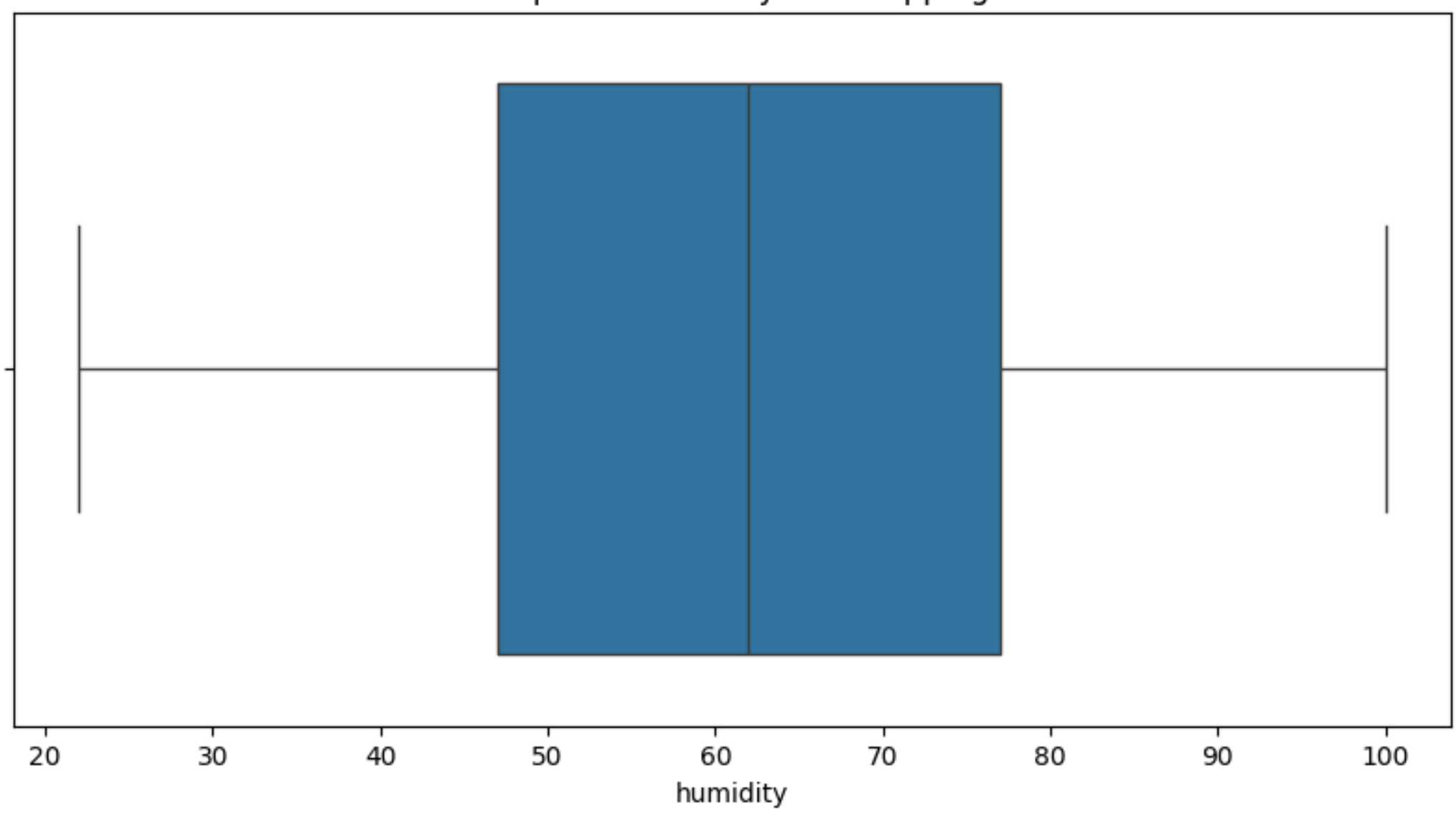

0.0
0.2
0.4
0.6
0.8
1.0
workingday


Boxplot of weather after clipping
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
weather

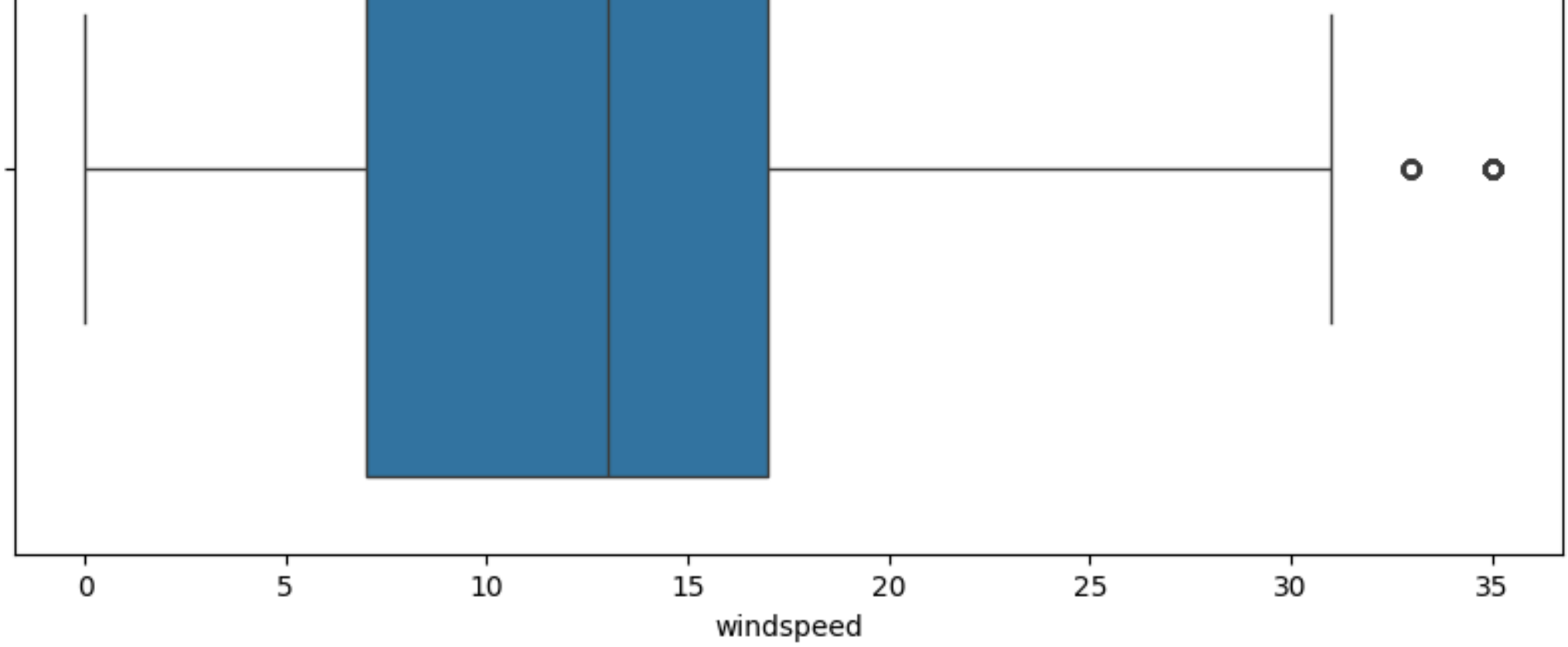

Boxplot of temp after clipping


5
10
15
20
25
30
35
temp

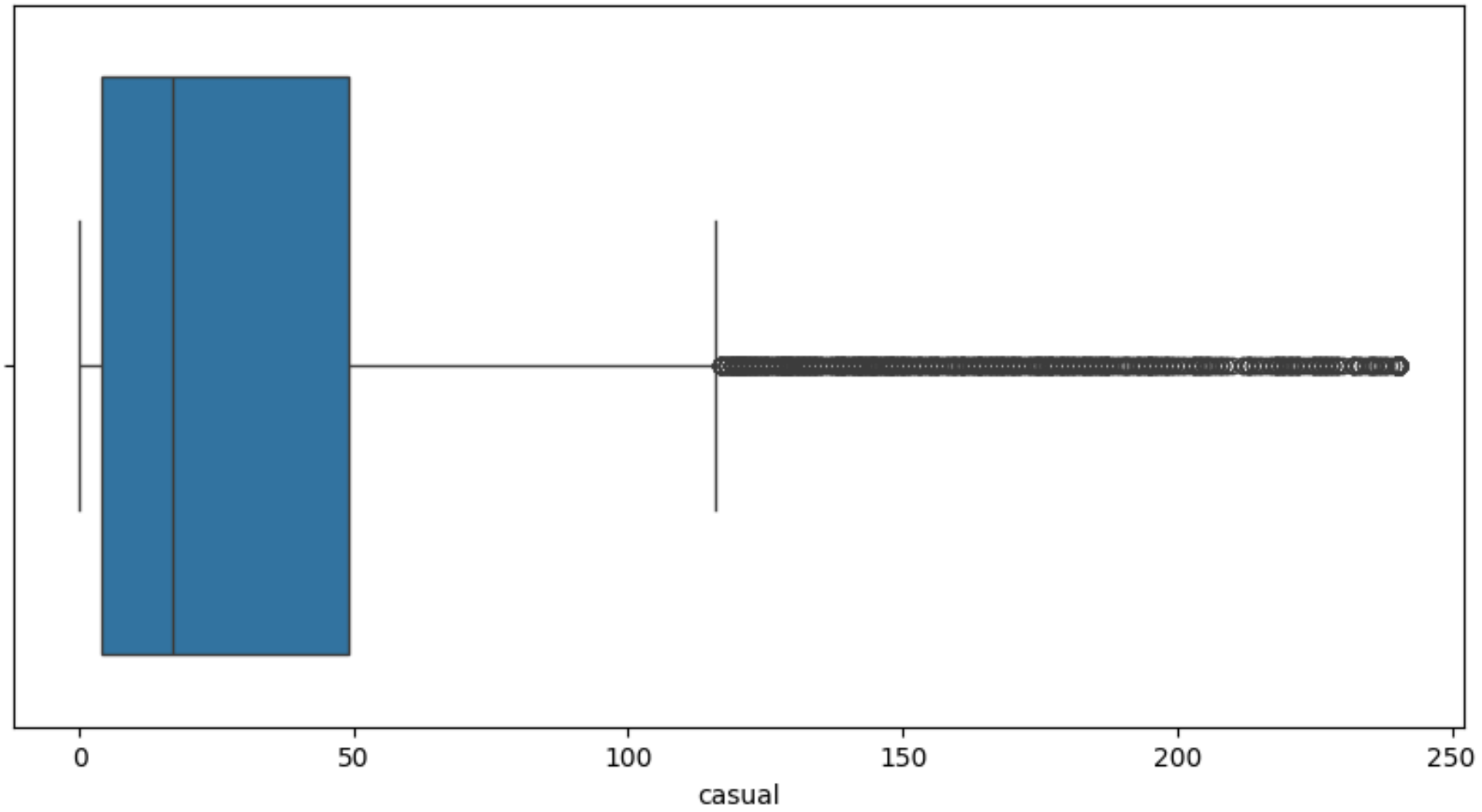

Boxplot of atemp after clipping
5
10
15
20
25
30
35
40
atemp


Boxplot of humidity after clipping
20
30
40
50
60
70
80
90
100
humidity

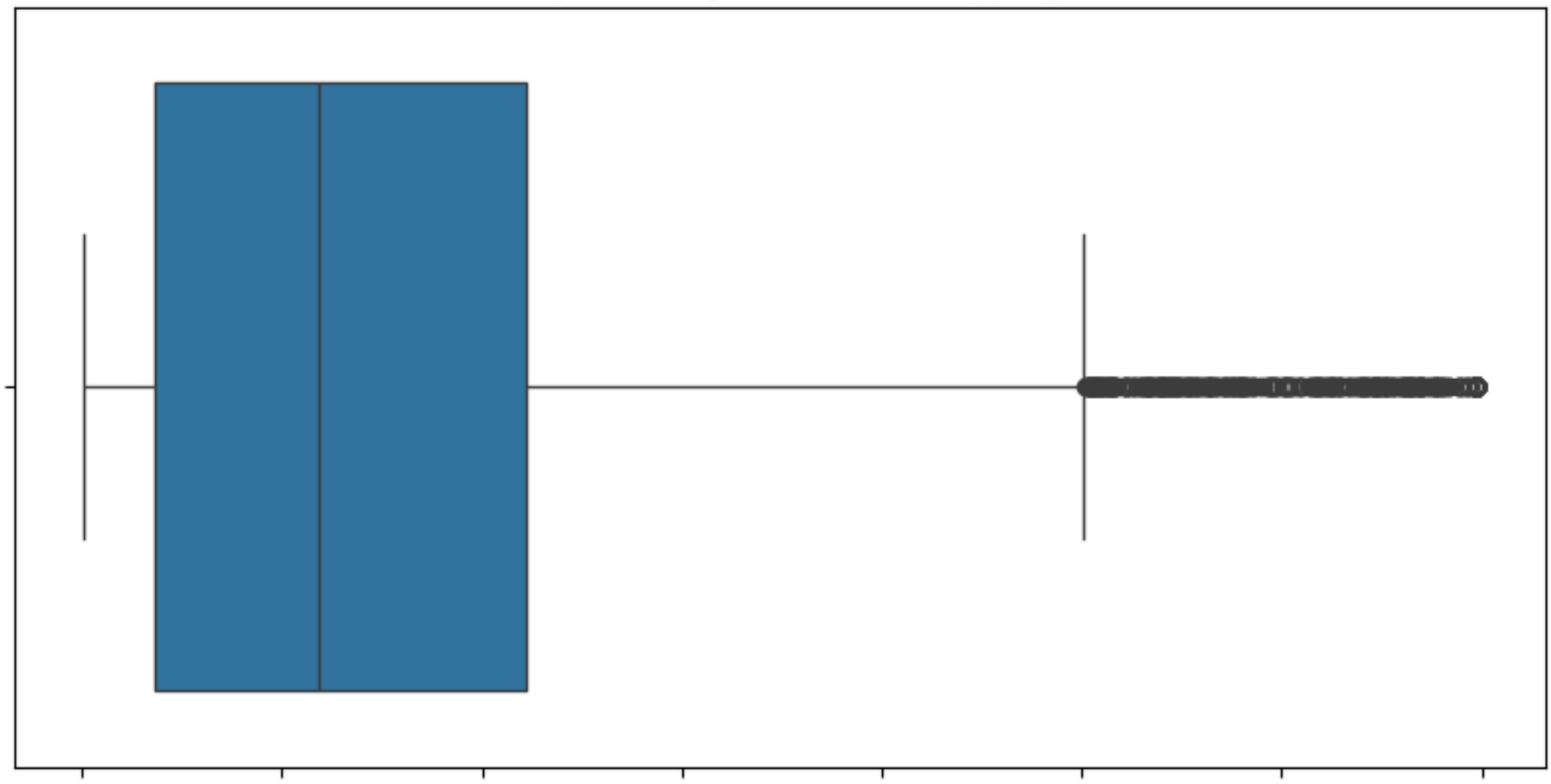

Boxplot of windspeed after clipping




Boxplot of casual after clipping

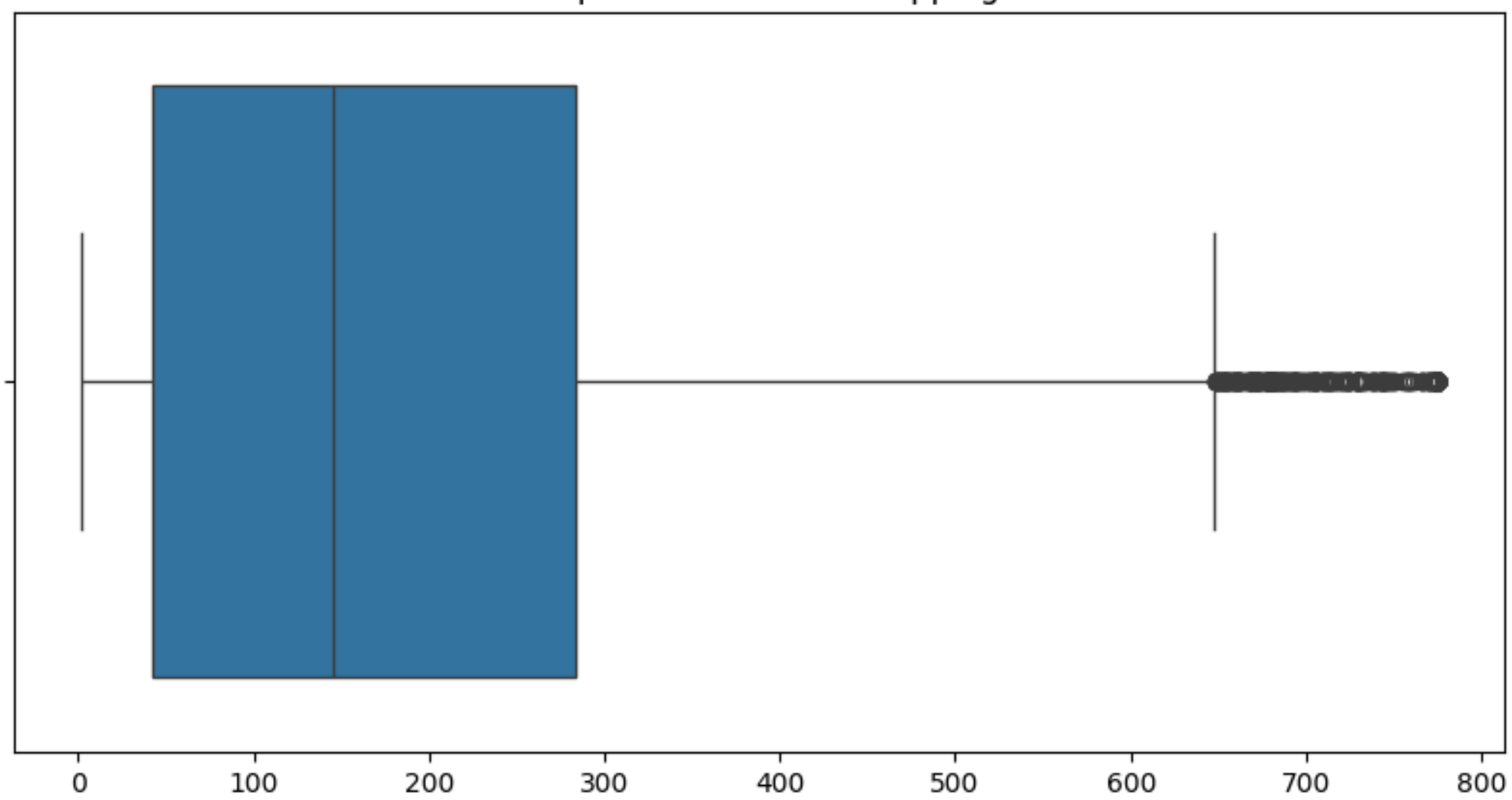



Boxplot of registered after clipping



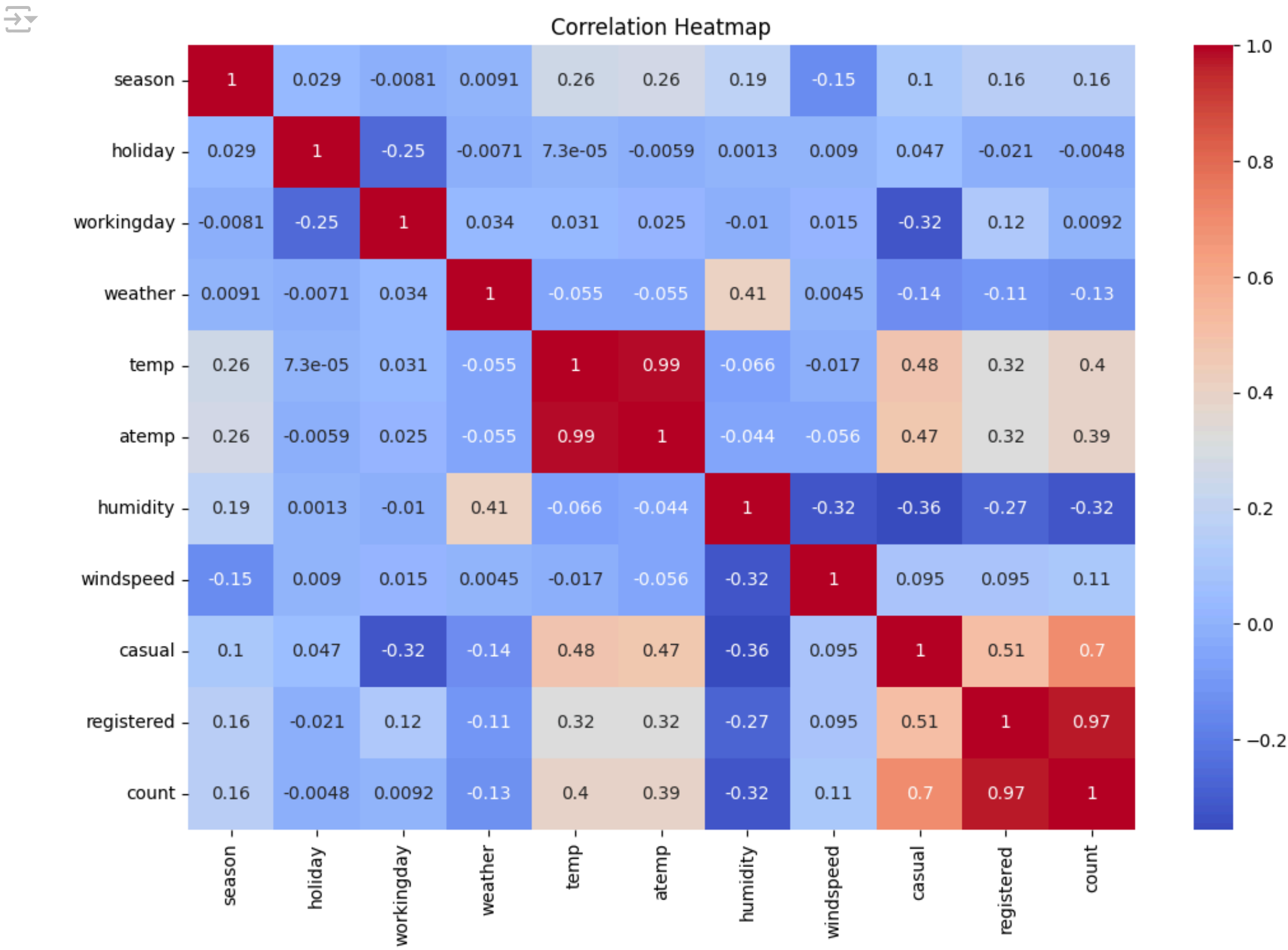
0 100 200 300 400 500 600 700

registered



## 2. Establishing Relationship Between Dependent and Independent Variables

```
plt.figure(figsize=(12, 8))

# Select only numerical columns
numerical_df = df.select_dtypes(include=[np.number])
correlation_matrix = numerical_df.corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```




## 3. Checking if theres any significant difference between the no. of bike rides on Weekdays and Weekends?

```
from scipy.stats import ttest_ind

df['datetime'] = pd.to_datetime(df['datetime'])
```

```
df['day_of_week'] = df['datetime'].dt.day_name()


df['is_weekend'] = df['day_of_week'].apply(lambda x: 1 if x in ['Saturday', 'Sunday'] else 0)

weekday_rentals = df[df['is_weekend'] == 0]['count']
weekend_rentals = df[df['is_weekend'] == 1]['count']


t_stat, p_value = ttest_ind(weekday_rentals, weekend_rentals)

# Set significance level
alpha = 0.05

print(f"T-statistic: {t_stat}, P-value: {p_value}")

if p_value <= alpha:
    print("Reject the null hypothesis: There is a significant difference between the number of bike rides on week
else:
    print("Fail to reject the null hypothesis: There is no significant difference between the number of bike ride
```

```
T-statistic: 0.804917743622243, P-value: 0.4208847307587942
Fail to reject the null hypothesis: There is no significant difference between the number of bike rid
```

## 4. Demand for Bicycles Based on Weather Conditions

```
from scipy.stats import f_oneway

# Assuming 'weather' column indicates weather conditions and 'COUNT' indicates the number of rentals
weather_conditions = df['weather'].unique()
weather_groups = [df[df['weather'] == condition]['count'] for condition in weather_conditions]

# One-way ANOVA test
f_stat, p_value = f_oneway(*weather_groups)

print(f"F-statistic: {f_stat}, P-value: {p_value}")

if p_value <= alpha:
    print("Reject the null hypothesis: There is a significant difference in bike rentals across different weather
else:
    print("Fail to reject the null hypothesis: There is no significant difference in bike rentals across differen
```

```
F-statistic: 99.27317733990745, P-value: 1.882634616089374e-43
Reject the null hypothesis: There is a significant difference in bike rentals across different weathe
```

## 5. Demand for Bicycles Based on Seasons

```
season_conditions = df['season'].unique()
season_groups = [df[df['season'] == condition]['count'] for condition in season_conditions]

# One-way ANOVA test
f_stat, p_value = f_oneway(*season_groups)

print(f"F-statistic: {f_stat}, P-value: {p_value}")
```

```
if p_value <= alpha:
    print("Reject the null hypothesis: There is a significant difference in bike rentals across different seasons
else:
    print("Fail to reject the null hypothesis: There is no significant difference in bike rentals across differen
```

```
F-statistic: 238.47048689463537, P-value: 7.238081075653907e-150
Reject the null hypothesis: There is a significant difference in bike rentals across different season
```

## 6. Weather Conditions Across Different Seasons

```
from scipy.stats import chi2_contingency

contingency_table = pd.crosstab(df['weather'], df['season'])

chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)

print(f"Chi-square statistic: {chi2_stat}, P-value: {p_value}")
```

```
if p_value <= alpha:
    print("Reject the null hypothesis: Weather conditions are significantly different during different seasons.")
else:
    print("Fail to reject the null hypothesis: Weather conditions are not significantly different during differen
```

```
Chi-square statistic: 46.09805776966069, P-value: 2.8304096630424604e-08
Reject the null hypothesis: Weather conditions are significantly different during different seasons.
```

Step 1: Exploratory Data Analysis (EDA)

**A. Examine dataset structure, characteristics, and statistical summary.**

**Insights**:

The dataset contains detailed information on bike rentals, including various attributes such as date, weather, season, and rental counts. The dataset appears to have no missing values or duplicate records, ensuring data quality.

**Recommendations**:

Continue to ensure data quality by regularly checking for missing values and duplicates. This will help in maintaining accurate and reliable data for future analysis.

Implement automated data quality checks in the data pipeline to detect and rectify any anomalies promptly.

**B. Identify missing values and perform imputation using an appropriate method.**

**Insights**:

No missing values were found in the dataset.

**Recommendations**:

Maintain consistent and complete data entry practices to avoid missing values in future datasets.

If missing values do occur, establish a protocol for imputation using appropriate methods to ensure data integrity.

**C. Identify and remove duplicate records.**

**Insights**: No duplicate records were found in the dataset.

**Recommendations**: Implement validation checks to prevent duplicate entries in the system. This ensures the accuracy of usage statistics and demand forecasting.

**D. Analyze the distribution of Numerical & Categorical variables**

**Insights**:

Numerical variables show varying distributions, some of which may be skewed. Categorical variables have different levels of frequency, indicating varied usage patterns.

**Recommendations**:

- Consider transforming skewed numerical variables to normalize the data. This can improve the performance of predictive models.
- For imbalanced categorical variables, consider strategies like targeted marketing or promotions to balance usage across different categories.

**e. Check for Outliers and deal with them accordingly.**

**Insights**:

Outliers were detected and removed based on the IQR method.

**Recommendations**:

- Investigate the causes of outliers to determine if they indicate data issues or true anomalies. Implement policies to address data entry errors.

**Step 2: Establish a Relationship between the Dependent and Independent Variables**

**Insights**:

The correlation heatmap indicates some highly correlated numerical variables.

**Recommendations**:

- Remove or combine highly correlated variables to prevent multicollinearity, which can skew the results of predictive models.
- Consider using techniques like Principal Component Analysis (PCA) to reduce the dimensionality of the data while preserving important information.

**Step 3: Significant difference between the number of bike rides on Weekdays and Weekends**

**Insights**:

A significant difference in bike rides between Weekdays and Weekends was found.

**Recommendations**:

- Increase bike availability during weekends to meet higher demand.
- Develop specific marketing strategies for weekdays to boost rentals, such as weekday promotions or partnerships with businesses for commuter benefits.

**Step 4: Demand for bicycles on rent for different Weather conditions**

**Insights**:

The demand for bicycles varies significantly with weather conditions.

**Recommendations**:

- Adjust the fleet size and maintenance schedules based on weather forecasts to ensure availability during favorable conditions.

- Provide weather-appropriate gear (e.g., raincoats, umbrellas) to customers to encourage bike rentals during less favorable weather conditions.

**Step 5: Demand for bicycles on rent for different Seasons**

**Insights**:

The demand for bicycles varies significantly with seasons.

***Recommendations***:

- Plan for increased maintenance and resource allocation during peak seasons to ensure that the fleet is in optimal condition.
- Launch seasonal marketing campaigns to capitalize on higher demand periods, such as summer or spring.

**Step 6: Weather conditions during different Seasons**

**Insights**:

Weather conditions significantly differ across seasons.

*Recommendations*:

- Use historical weather patterns to predict bike rental demand and adjust operations accordingly.
- Optimize inventory levels based on expected seasonal weather conditions to avoid under- or over-supply of bikes.

**SUMMARY AND RECOMMENDATIONS**

Based on the analysis, Yulu can enhance its bike-sharing service by implementing the following strategies:

- Maintain data quality and consistency to ensure reliable analysis. Address data anomalies and manage outliers to improve the accuracy of insights.
- Optimize bike availability and maintenance schedules based on demand patterns influenced by day of the week, weather, and seasons.