

```
pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.8.0-py3-none-any.whl.metadata (7.9 kB)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.26.4)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (2.2.2)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.0.1)
Requirement already satisfied: scikit-learn>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.6.1)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.13.1)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (0.14.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.5->category_encoders) (2024.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.5->category_encoders) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.5->category_encoders) (2024.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.6.0->category_encoders) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.6.0->category_encoders) (3.5.0)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from statsmodels>=0.9.0->category_encoders) (24.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.0.5->category_encoders) (1.17.0)
Downloading category_encoders-2.8.0-py3-none-any.whl (85 kB)
85.7/85.7 kB 6.6 MB/s eta 0:00:00
Installing collected packages: category_encoders
Successfully installed category_encoders-2.8.0
```

```
pip install pingouin
```


```
Collecting pingouin
  Downloading pingouin-0.5.5-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from pingouin) (3.10.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from pingouin) (1.26.4)
Requirement already satisfied: pandas>=1.5 in /usr/local/lib/python3.11/dist-packages (from pingouin) (2.2.2)
Collecting pandas_flavor (from pingouin)
  Downloading pandas_flavor-0.6.0-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: scikit-learn>=1.2 in /usr/local/lib/python3.11/dist-packages (from pingouin) (1.6.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from pingouin) (1.13.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (from pingouin) (0.13.2)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.11/dist-packages (from pingouin) (0.14.4)
Requirement already satisfied: tabulate in /usr/local/lib/python3.11/dist-packages (from pingouin) (0.9.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.5->pingouin) (2024.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.5->pingouin) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.5->pingouin) (2025.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.2->pingouin) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.2->pingouin) (3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->pingouin) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->pingouin) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->pingouin) (4.55.7)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->pingouin) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->pingouin) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib->pingouin) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->pingouin) (3.2.1)
Requirement already satisfied: xarray in /usr/local/lib/python3.11/dist-packages (from pandas_flavor->pingouin) (2025.1.1)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packages (from statsmodels->pingouin) (1.0.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.5->pingouin) (1.17.0)
Downloading pingouin-0.5.5-py3-none-any.whl (204 kB)
204.4/204.4 kB 14.8 MB/s eta 0:00:00
Downloading pandas_flavor-0.6.0-py3-none-any.whl (7.2 kB)
Installing collected packages: pandas_flavor, pingouin
Successfully installed pandas-flavor-0.6.0 pingouin-0.5.5
```



```
from scipy.stats import norm, poisson, binom, expon, geom, bernoulli, randint, ttest_1samp, ttest_rel, ttest_ind, chi2, chisquare, chi2_2
from statsmodels.graphics.gofplots import qqplot
import math
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.formula.api import ols
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
from category_encoders import TargetEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import pingouin as pg
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files bike_sharing.csv
• bike_sharing.csv(text/csv) - 648353 bytes, last modified: 5/2/2025 - 100% done
Saving bike_sharing.csv to bike_sharing.csv
```

```
df= pd.read_csv('bike_sharing.csv')
df.head(5)
```




	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	


Next steps:

[Generate code with df](#)[View recommended plots](#)[New interactive sheet](#)

df.shape


 (10886, 12)

df.describe()



	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	re
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	108
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	1
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	1
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	1
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	2
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	8

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
```

```
# Check for missing values
missing_values = df.isnull().sum()
missing_values
```

```

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

```

dtype: int64

```

# Check for duplicate records
duplicate_records = df.duplicated().sum()
duplicate_records

```

```
0
```

```

# Convert datetime column to proper datetime format
df['datetime'] = pd.to_datetime(df['datetime'])

```

```

# Extract useful time-based features
df['hour'] = df['datetime'].dt.hour
df['day_of_week'] = df['datetime'].dt.dayofweek
df['month'] = df['datetime'].dt.month

```

```
df.head(5)
```

```

datetime season holiday workingday weather temp atemp humidity windspeed casual registered count hour day_of_week mc
0 2011-01-01 1 0 0 1 9.84 14.395 81 0.0 3 13 16 0 5
1 2011-01-01 1 0 0 1 9.02 13.635 80 0.0 8 32 40 1 5
2 2011-01-01 1 0 0 1 9.02 13.635 80 0.0 5 27 32 2 5
3 2011-01-01 1 0 0 1 9.84 14.395 75 0.0 3 10 13 3 5

```

Next steps:

[Generate code with df](#)
[View recommended plots](#)
[New interactive sheet](#)

```

# Plot distributions of numerical features
num_features = ['temp', 'atemp', 'humidity', 'windspeed', 'count']

```

```

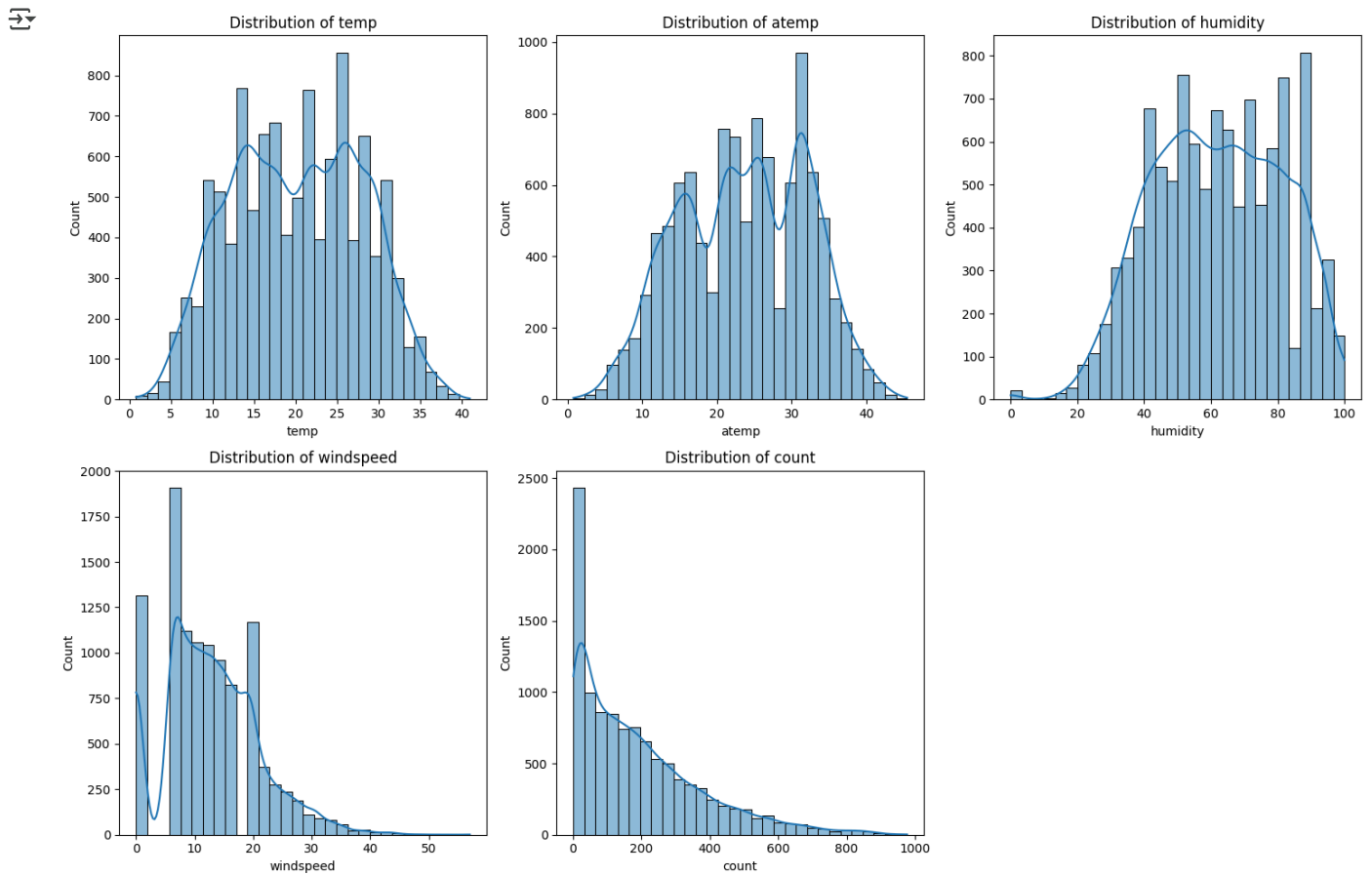
plt.figure(figsize=(15, 10))
for i, col in enumerate(num_features, 1):
    plt.subplot(2, 3, i)
    sns.histplot(df[col], bins=30, kde=True)
    plt.title(f'Distribution of {col}')

```

```

plt.tight_layout()
plt.show()

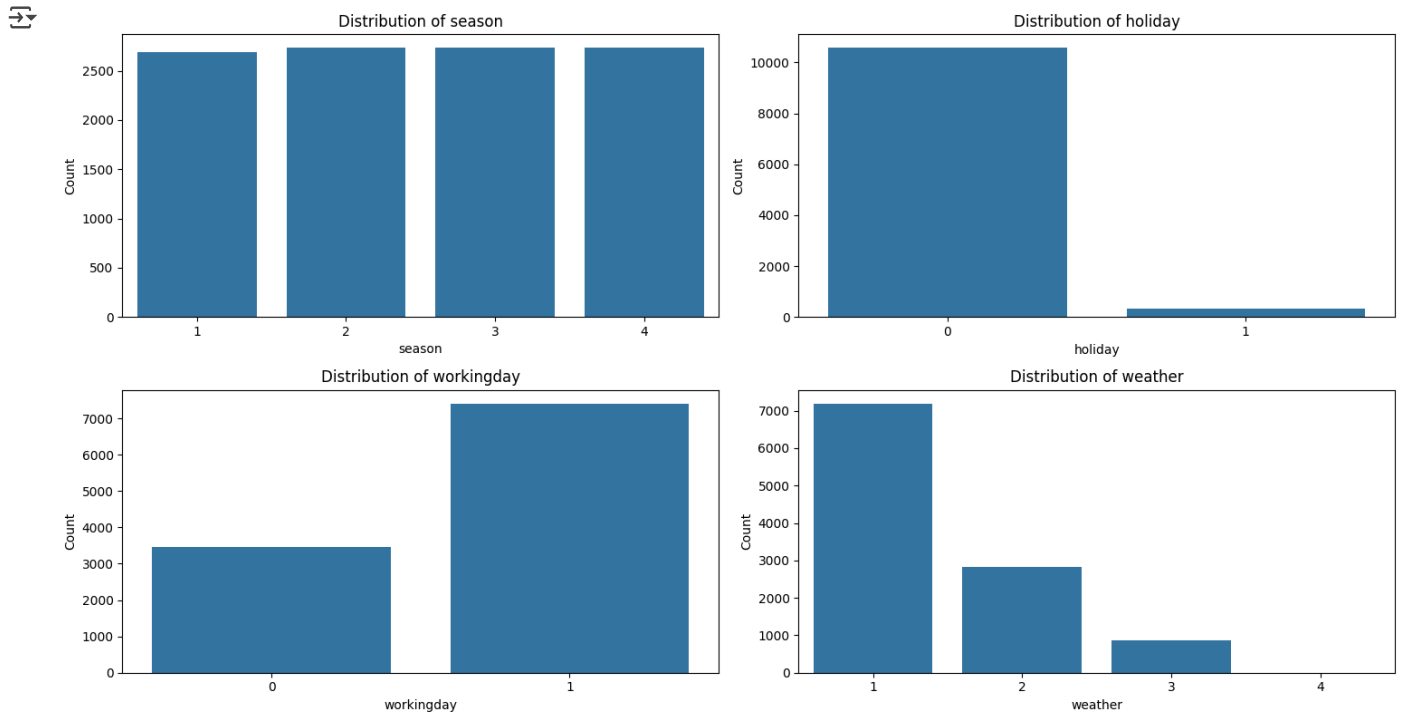
```



```
# Define categorical features
cat_features = ['season', 'holiday', 'workingday', 'weather']

# Plot count distributions of categorical features
plt.figure(figsize=(15, 8))
for i, col in enumerate(cat_features, 1):
    plt.subplot(2, 2, i)
    sns.countplot(x=df[col])
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Count')

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



```
# Define numerical features to check for outliers
num_features = ['temp', 'atemp', 'humidity', 'windspeed', 'count']

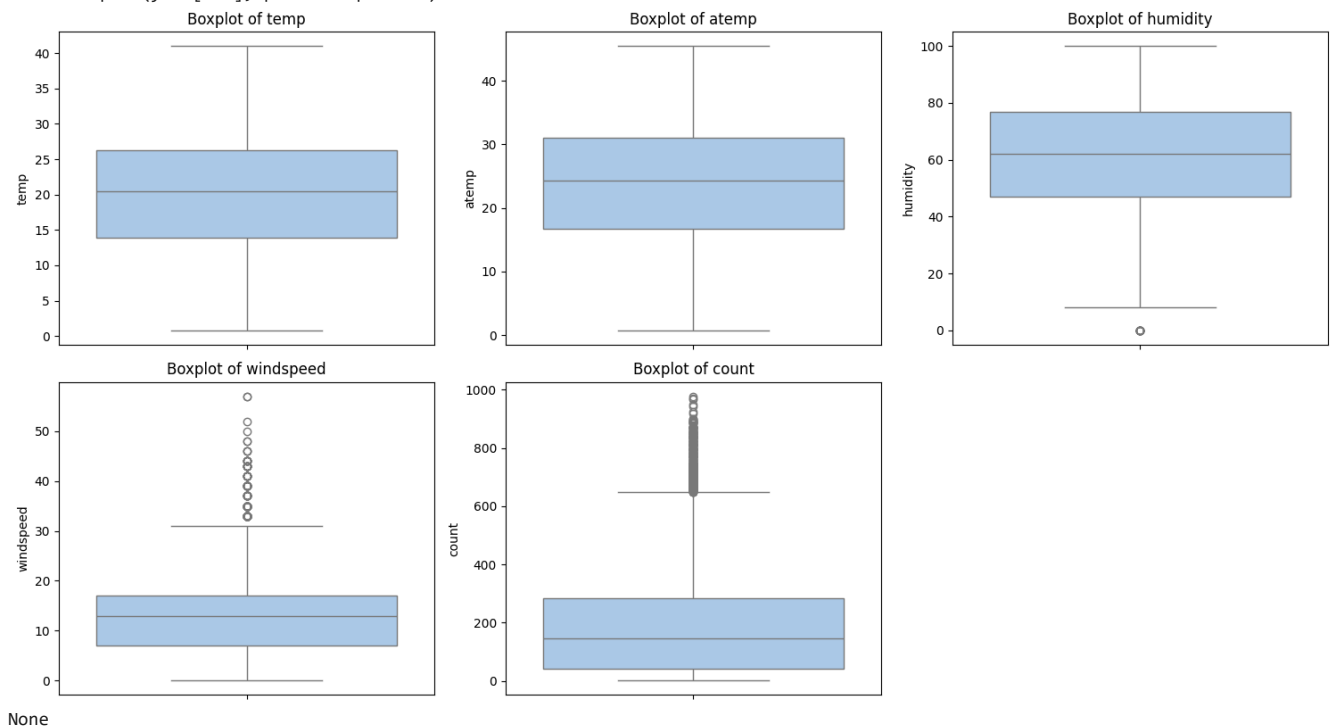
# Create boxplots for each numerical feature
plt.figure(figsize=(15, 8))
for i, col in enumerate(num_features, 1):
    plt.subplot(2, 3, i) # Creating subplots
    sns.boxplot(y=df[col], palette="pastel")
    plt.title(f'Boxplot of {col}')

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

```

<ipython-input-77-651797f8c8a8>:8: FutureWarning:
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set
sns.boxplot(y=df[col], palette="pastel")
<ipython-input-77-651797f8c8a8>:8: FutureWarning:
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set
sns.boxplot(y=df[col], palette="pastel")
<ipython-input-77-651797f8c8a8>:8: FutureWarning:
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set
sns.boxplot(y=df[col], palette="pastel")
<ipython-input-77-651797f8c8a8>:8: FutureWarning:
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set
sns.boxplot(y=df[col], palette="pastel")

```



```

# Define numerical features for outlier detection
num_features = ['temp', 'atemp', 'humidity', 'windspeed', 'count']

# Create a copy of the dataframe to store the cleaned data
df_cleaned = df.copy()

# Function to remove outliers using IQR method
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25) # 25th percentile
    Q3 = df[column].quantile(0.75) # 75th percentile
    IQR = Q3 - Q1 # Interquartile Range
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Remove rows where the column value is outside the bounds
    df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
    return df

# Apply IQR method to all numerical features
for col in num_features:
    df_cleaned = remove_outliers_iqr(df_cleaned, col)

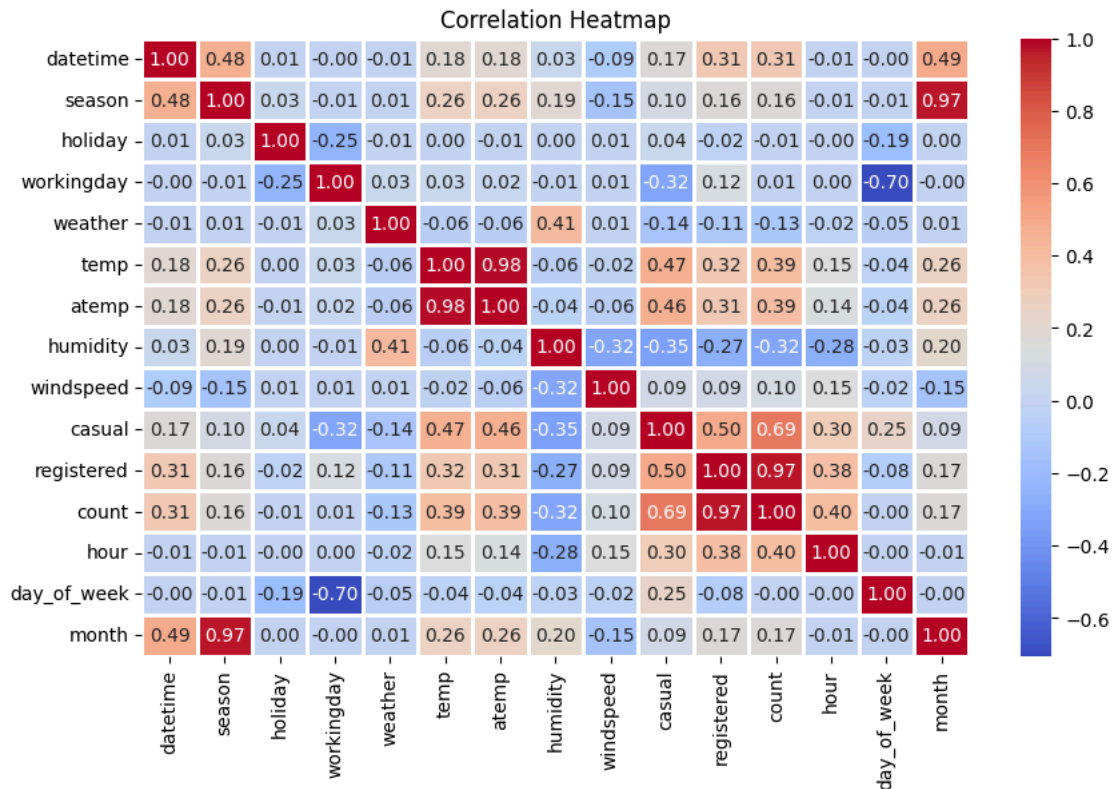
```

```
# Print the difference in row count after removing outliers
print(f"Original dataset size: {df.shape[0]} rows")
print(f"Cleaned dataset size: {df_cleaned.shape[0]} rows")
print(f"Total rows removed: {df.shape[0] - df_cleaned.shape[0]}")
```

```
Original dataset size: 10886 rows
Cleaned dataset size: 10352 rows
Total rows removed: 534
```

```
# Compute the correlation matrix
correlation_matrix = df.corr()
```

```
# Plot the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=1)
plt.title('Correlation Heatmap')
plt.show()
```



```
df = df.drop(columns=['atemp'])
df.head(10)
```



	datetime	season	holiday	workingday	weather	temp	humidity	windspeed	casual	registered	count	hour	day_of_week	month
0	2011-01-01 00:00:00	1	0	0	1	9.84	81	0.0000	3	13	16	0	5	1
1	2011-01-01 01:00:00	1	0	0	1	9.02	80	0.0000	8	32	40	1	5	1
2	2011-01-01 02:00:00	1	0	0	1	9.02	80	0.0000	5	27	32	2	5	1
3	2011-01-01 03:00:00	1	0	0	1	9.84	75	0.0000	3	10	13	3	5	1
4	2011-01-01 04:00:00	1	0	0	1	9.84	75	0.0000	0	1	1	4	5	1

Next steps:

[Generate code with df](#)[View recommended plots](#)[New interactive sheet](#)

T test two way

Data1=df[df['workingday'] == 1]['count'] # Weekdays (workingday = 1)

```
Data2=df[df['workingday'] == 0]['count'] # Weekends (workingday = 0)
alpha = 0.05
```

```
#Null Hypothesis (H0): There is no significant difference in the number of bike rentals between weekdays and weekends.
#Alternative Hypothesis (H1): There is a significant difference in the number of bike rentals between weekdays and weekends.
# Ho mu1=mu2
#Case1 : Ha : mu1!=mu2 two tail
t_stat, pvalue = ttest_ind(Data1, Data2)
```

```
print ("p value",pvalue)
print("T stat",t_stat)
if pvalue < alpha:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
↗ p value 0.22644804226361348
T stat 1.2096277376026694
Fail to reject H0
```

```
# checking skewness of count column
data=df['count'] # in list
data_series = pd.Series(data)
skewness=data_series.skew()
print ("skewness",skewness)
if skewness > 0:
    print("The distribution is positively or right skewed .")
elif skewness < 0:
    print("The distribution is negatively or left skewed.")
else:
    print("The distribution is symmetric.")
```

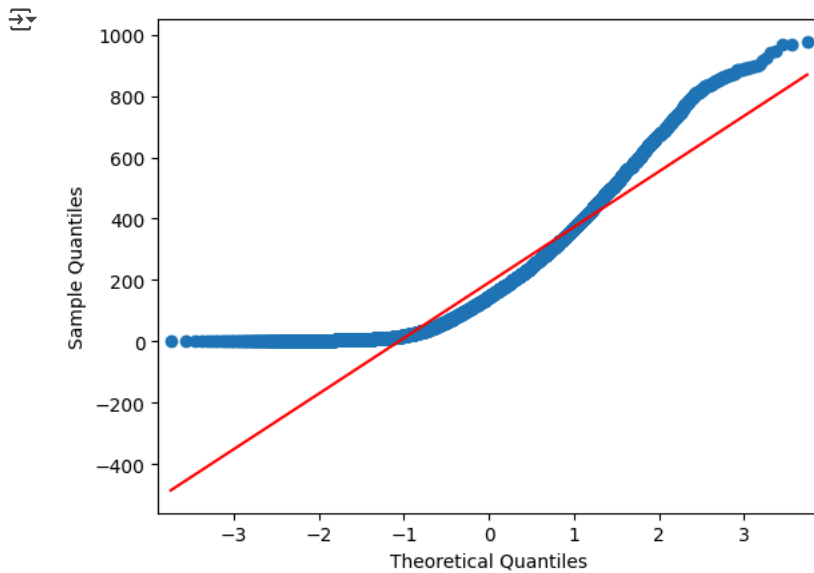
```
↗ skewness 1.2420662117180776
The distribution is positively or right skewed .
```

```
#Shapiro-Wilk Test to check normal distribution
#Ho: Data is Gaussian
#Ha: Data is not Gaussian
```

```
Data1=df['count'] # data set or list
Alpha= 0.05
test_stat, p_value = shapiro(Data1)
print ("test stat",test_stat)
print ("p value",p_value)
if p_value < Alpha:
    print("Reject H0")
    print("Data is not Gaussian")
else:
    print("Fail to reject H0")
    print("Data is Gaussian")
```

```
↗ test stat 0.8783658962690556
p value 5.369837893115507e-68
Reject H0
Data is not Gaussian
/usr/local/lib/python3.11/dist-packages/scipy/stats/_axis_nan_policy.py:531: UserWarning: scipy.stats.shapiro: For N > 5000, compute
res = hypotest_fun_out(*samples, **kwargs)
```

```
# checking normality by Q-Q plot
axis= df['count'] # list of groups or data
qqplot(axis, line="s")
plt.show()
```

#Homogeneity of Variance (Levene's Test) for weather

```
# H0: Variances are equal
# Ha: Variances are not equal
Data1 = df[df['weather'] == 1]['count']
Data2 = df[df['weather'] == 2]['count']
Data3 = df[df['weather'] == 3]['count']
Data4 = df[df['weather'] == 4]['count']
Alpha= 0.05
levene_stat, p_value = levene(Data1, Data2, Data3, Data4)
print ("levene stat",levene_stat)
print ("p value",p_value)
if p_value < Alpha:
    print("Reject H0")
    print("Variances are not equal")
else:
    print("Fail to reject H0")
    print("Variances are equal")
```

```
levene stat 54.85106195954556
p value 3.504937946833238e-35
Reject H0
Variances are not equal
```

#Kruskal Wallis Test for counts and weather

#Null Hypothesis (H_0): There is no significant difference in bike demand across different weather conditions.

#Alternative Hypothesis (H_1): There is a significant difference in bike demand across different weather conditions

```
Data1 = df[df['weather'] == 1]['count']
Data2 = df[df['weather'] == 2]['count']
Data3= df[df['weather'] == 3]['count']
Data4= df[df['weather'] == 4]['count']
Alpha= 0.05
stat, p_value = kruskal(Data1, Data2, Data3,Data4)
print("f_stats",stat)
print("p_value",p_value)
if p_value < Alpha:
    print("Reject H0")
else:
    print("Fail to reject H0")
```

```
f_stats 205.00216514479087
p_value 3.501611300708679e-44
Reject H0
```

#Homogeneity of Variance (Levene's Test) for season

```
# H0: Variances are equal
# Ha: Variances are not equal
Data1 = df[df['season'] == 1]['count']
```

```

Data2 = df[df['season'] == 2]['count']
Data3 = df[df['season'] == 3]['count']
Data4 = df[df['season'] == 4]['count']
Alpha= 0.05
levene_stat, p_value = levene(Data1, Data2, Data3, Data4)
print ("levene stat",levene_stat)
print ("p value",p_value)
if p_value < Alpha:
    print("Reject H0")
    print("Variances are not equal")
else:
    print("Fail to reject H0")
    print("Variances are equal")

```

```

↗ levene stat 187.7706624026276
  p value 1.0147116860043298e-118
  Reject H0
  Variances are not equal

```

```
#Kruskal Wallis Test for counts and seasons
```

```
#Null Hypothesis (H0): There is no significant difference in bike demand across different weather conditions.
```

```
#Alternative Hypothesis (H1): There is a significant difference in bike demand across different weather conditions
```

```

Data1 = df[df['season'] == 1]['count']
Data2 = df[df['season'] == 2]['count']
Data3 = df[df['season'] == 3]['count']
Data4 = df[df['season'] == 4]['count']
Alpha= 0.05
stat, p_value = kruskal(Data1, Data2, Data3,Data4)
print("f_stats",stat)
print("p_value",p_value)
if p_value < Alpha:
    print("Reject H0")

```

```

else:
    print("Fail to reject H0")

```

```

↗ f_stats 699.6668548181988
  p_value 2.479008372608633e-151
  Reject H0

```

```
# Chi-Square Test for Independence of "Weather" and "Season"
```

```

observed = pd.crosstab(index=df['season'],columns=df['weather'])
alpha= 0.05
chi_stat, p_value, df, exp_freq = chi2_contingency(observed)
# chi_statp_value, df, expected values
print ('chi_stat',chi_stat)
print ('p value',p_value)
if p_value < alpha:
    print("Reject H0, they are associated")
else:
    print("Fail to reject H0, they are independen")

```

```

↗ chi stat 49.158655596893624
  p value 1.549925073686492e-07
  Reject H0, they are associated

```

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.