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

df = pd.read_csv('C:/Users/shrad/Desktop/Yulu_data.csv')

df.head()
df.info()  # Shows data types and if there are any missing values
```

```
<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
```

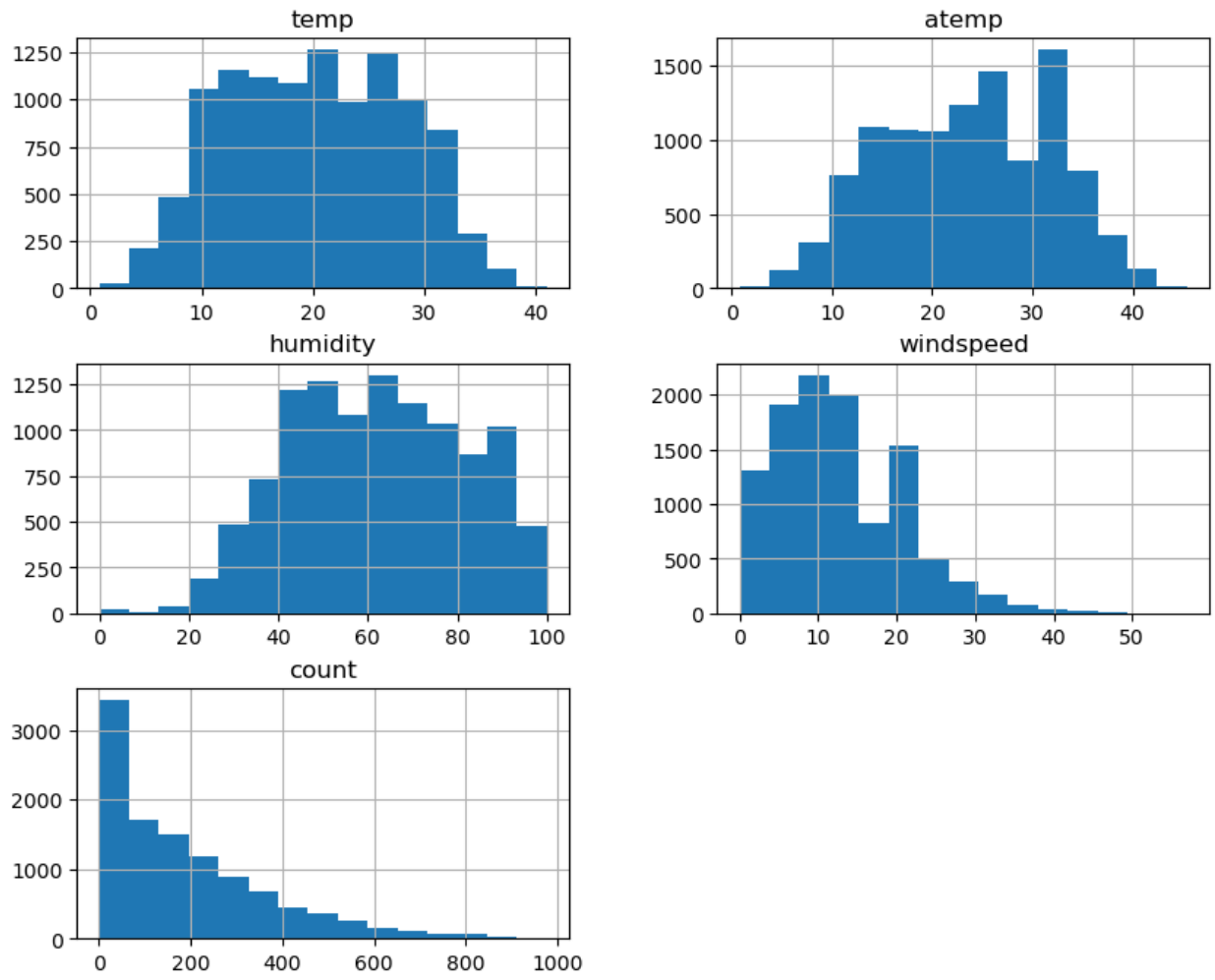
```python
print(df.isnull().sum())
```

```
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
```

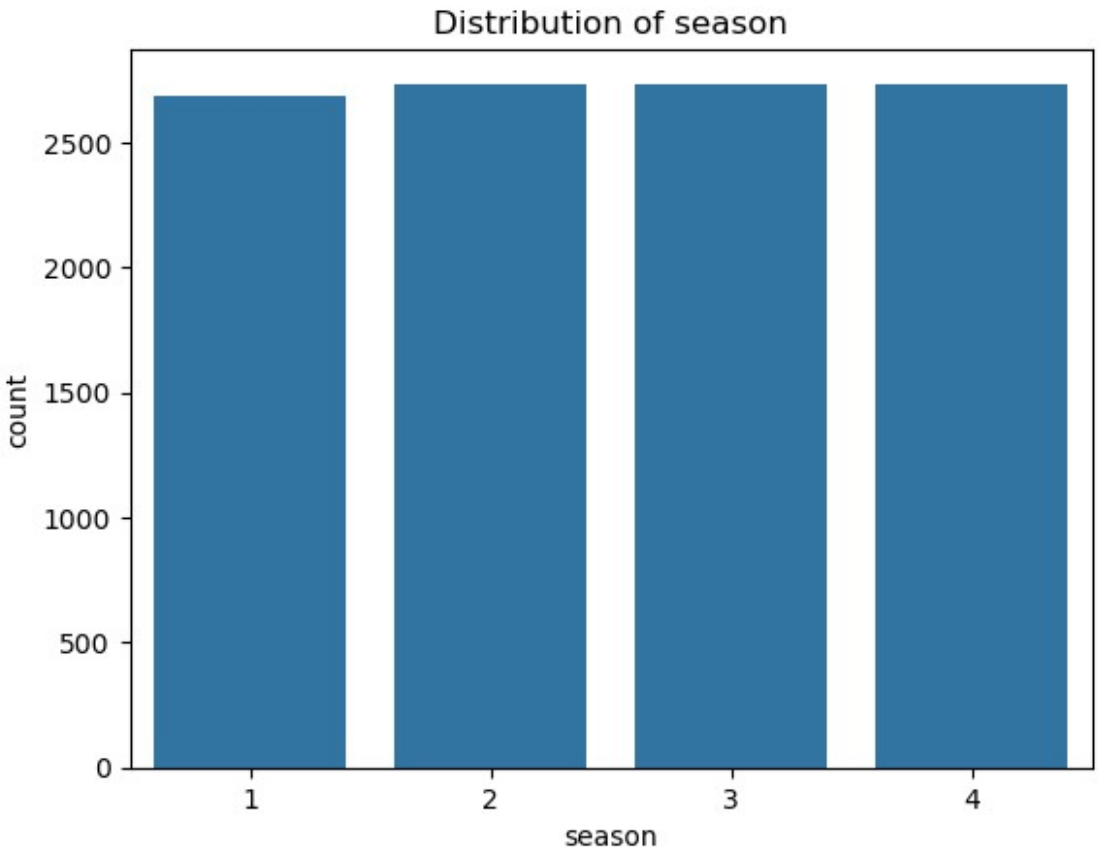
```python
categorical_columns = ['season', 'holiday', 'workingday', 'weather']
df[categorical_columns] = df[categorical_columns].astype('category')

df.describe()
```

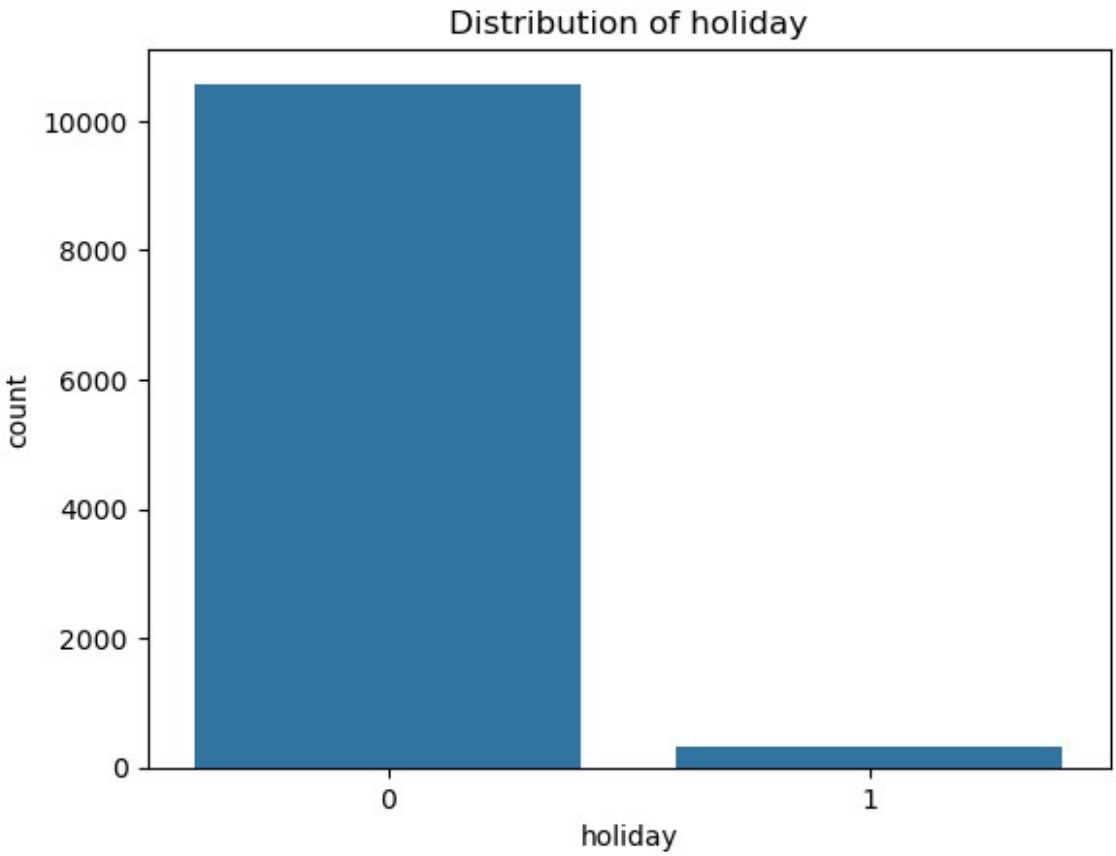
```
              temp         atemp      humidity     windspeed
casual  \
count  10886.00000  10886.000000  10886.000000  10886.000000
10886.000000
mean      20.23086     23.655084     61.886460     12.799395
36.021955
std        7.79159      8.474601     19.245033      8.164537
49.960477
min        0.82000      0.760000      0.000000      0.000000
0.000000
25%       13.94000     16.665000     47.000000      7.001500
4.000000
50%       20.50000     24.240000     62.000000     12.998000
17.000000
75%       26.24000     31.060000     77.000000     16.997900
49.000000
max       41.00000     45.455000    100.000000     56.996900
367.000000

         registered         count
count  10886.000000  10886.000000
mean     155.552177    191.574132
std      151.039033    181.144454
min        0.000000      1.000000
25%       36.000000     42.000000
50%      118.000000    145.000000
75%      222.000000    284.000000
max      886.000000    977.000000
```

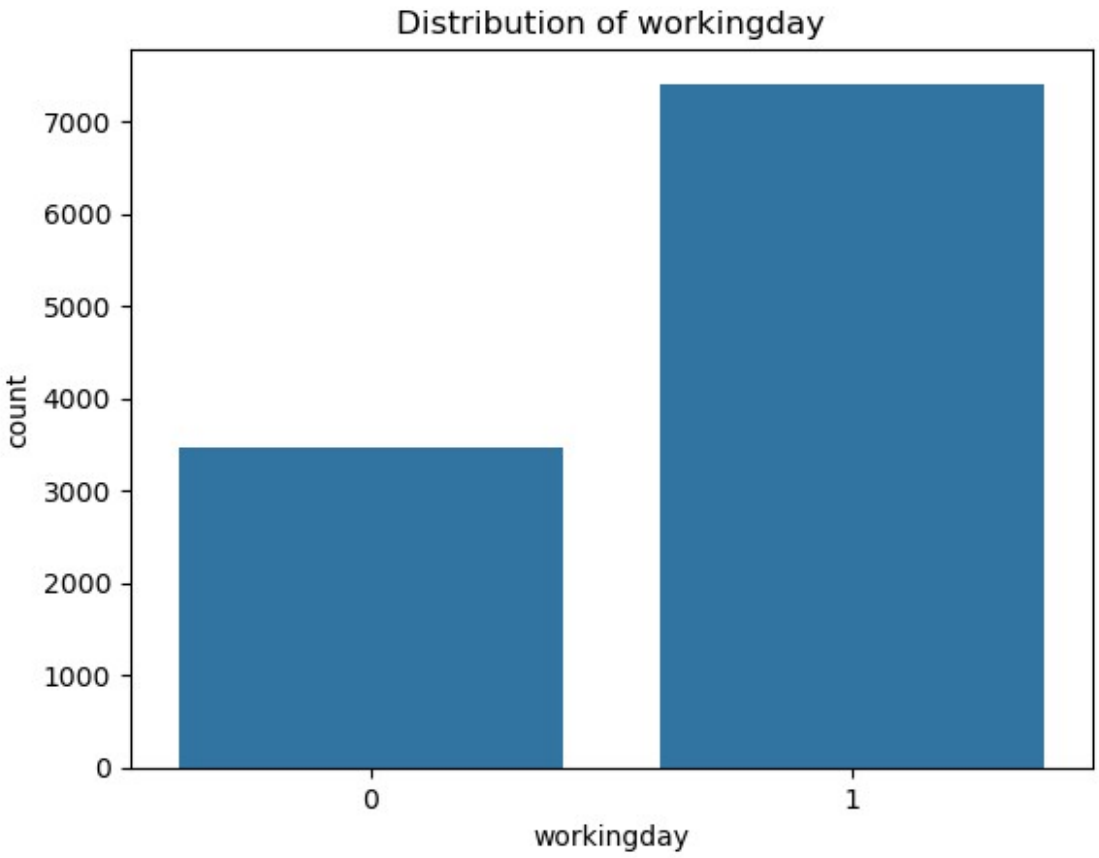
```
continuous_columns = ['temp', 'atemp', 'humidity', 'windspeed',
'count']
df[continuous_columns].hist(bins=15, figsize=(10, 8))
plt.suptitle('Distribution of Continuous Variables')
plt.show()
```

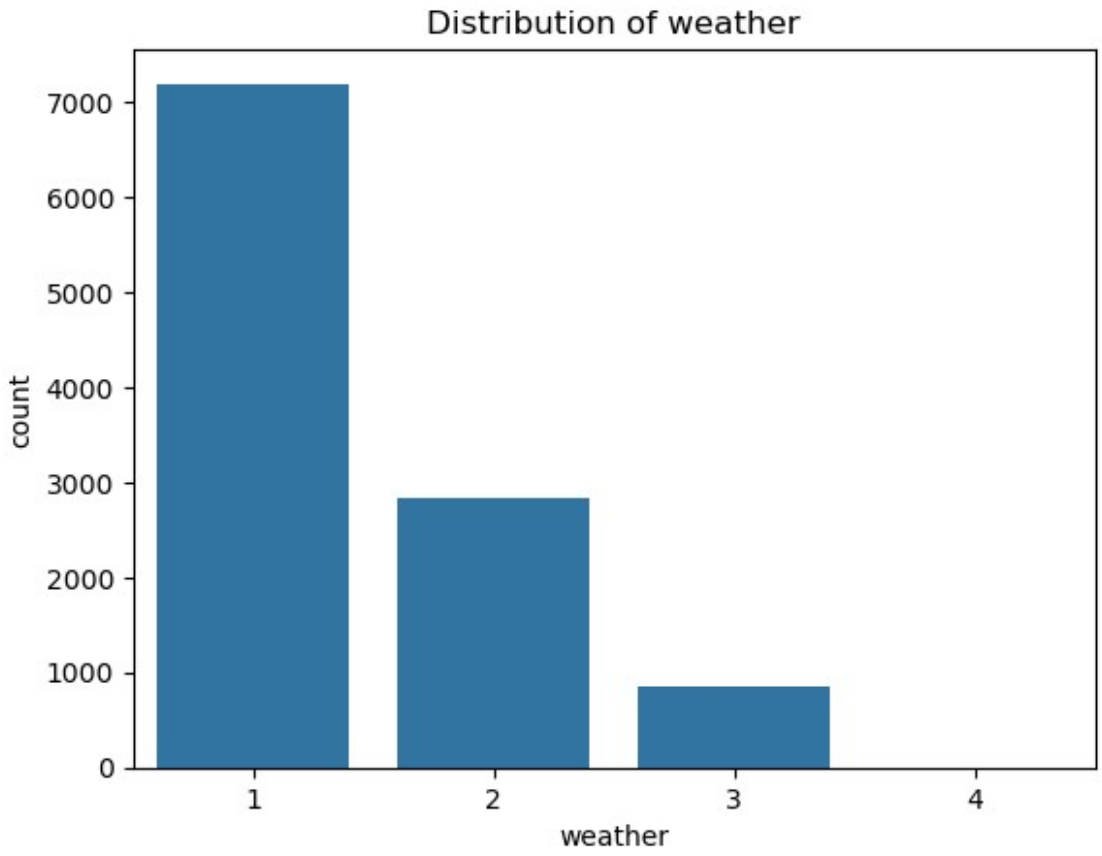



```
for col in categorical_columns:
    sns.countplot(data=df, x=col)
    plt.title(f'Distribution of {col}')
    plt.show()
```

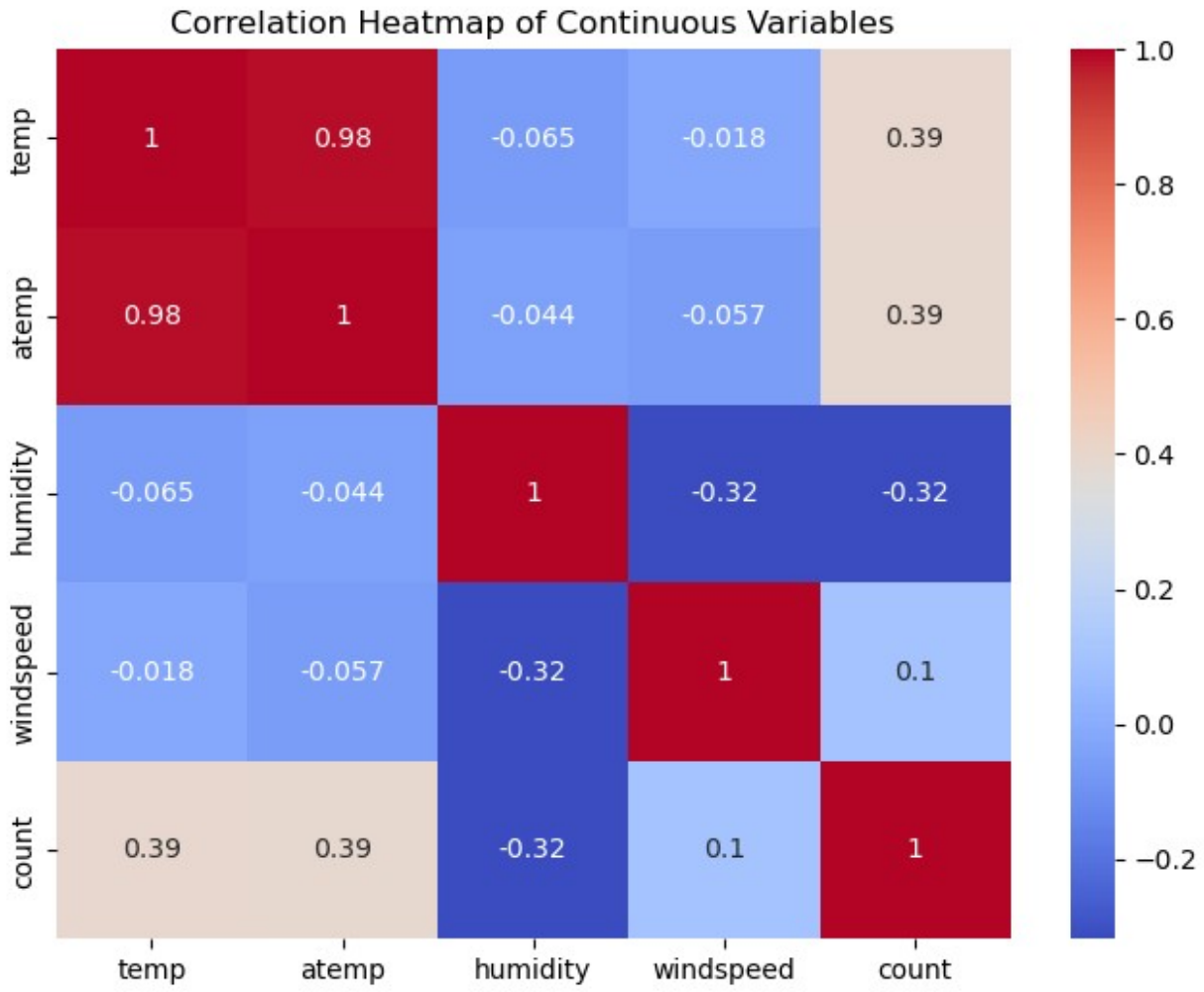

Distribution of season
2500
2000
1500
1000
500
0
count
1
2
3
4
season


Distribution of holiday
10000
8000
6000
4000
2000
0
count
0
1
holiday

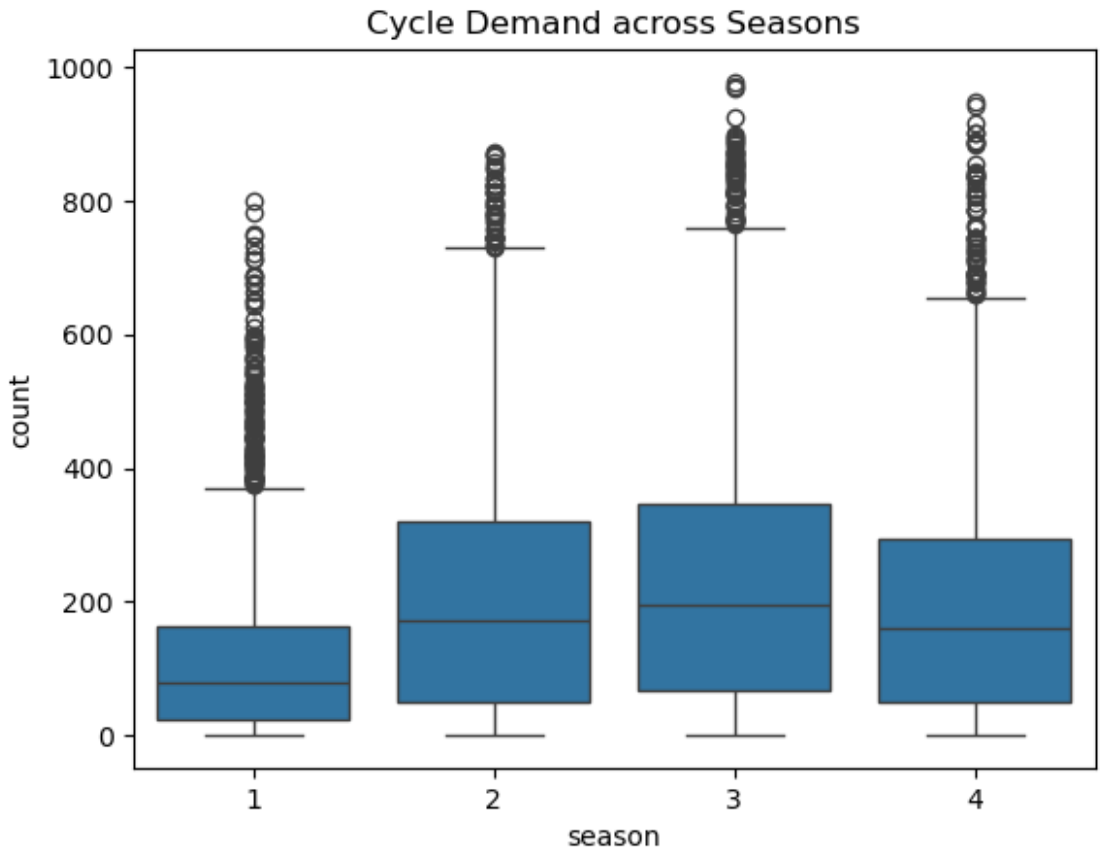

Distribution of workingday
7000
6000
5000
4000
3000
2000
1000
0
count
0
1
workingday

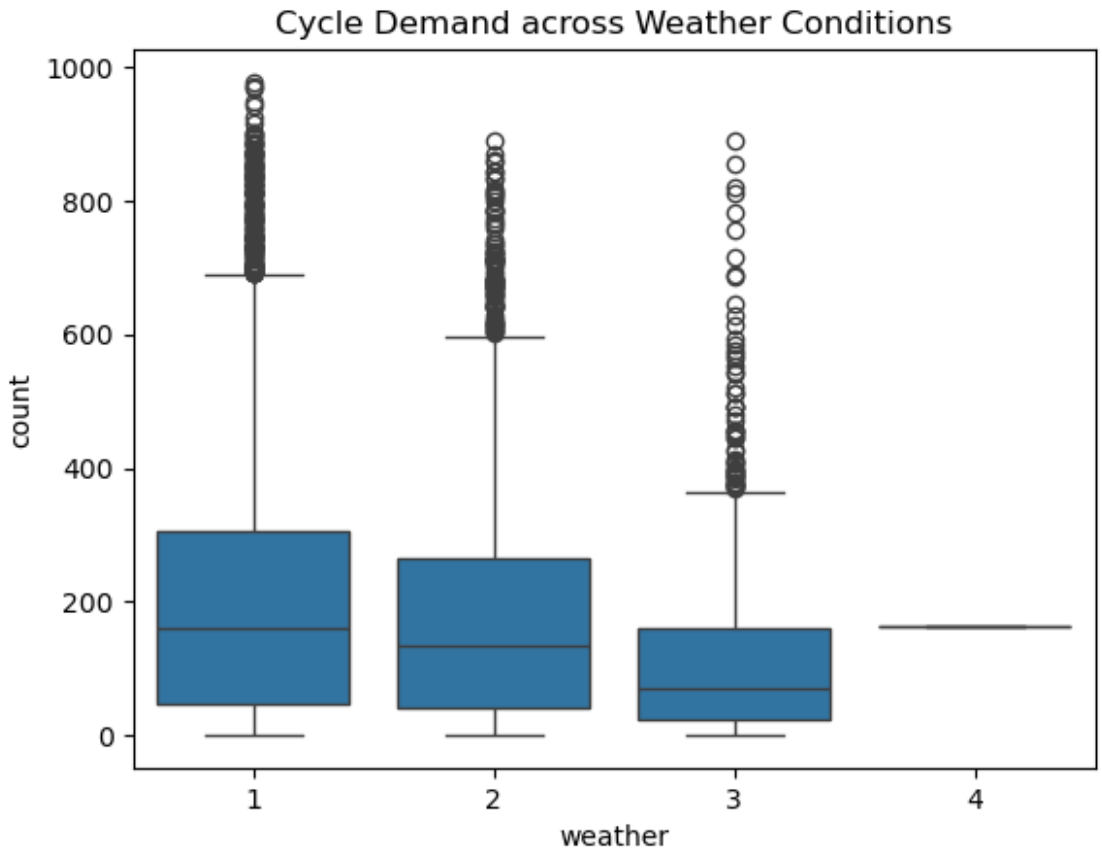


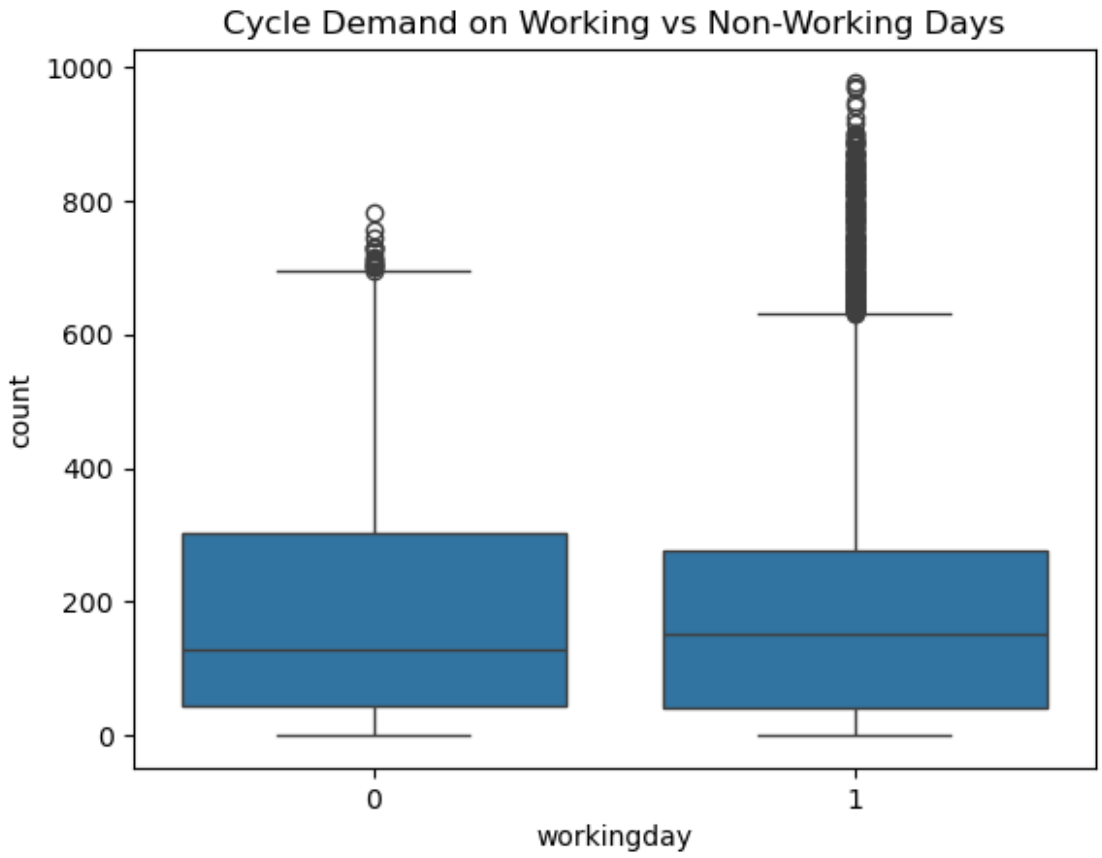

```
plt.figure(figsize=(8, 6))
sns.heatmap(df[continuous_columns].corr(), annot=True,
cmap='coolwarm')
plt.title('Correlation Heatmap of Continuous Variables')
plt.show()
```




```
sns.boxplot(data=df, x='season', y='count')
plt.title('Cycle Demand across Seasons')
plt.show()

sns.boxplot(data=df, x='weather', y='count')
plt.title('Cycle Demand across Weather Conditions')
plt.show()

sns.boxplot(data=df, x='workingday', y='count')
plt.title('Cycle Demand on Working vs Non-Working Days')
plt.show()
```


Cycle Demand across Seasons
1000
800
600
400
200
0
count
1
2
3
4
season


Cycle Demand across Weather Conditions
1000
800
600
400
200
0
count
1
2
3
4
weather



1. T-Test for Working Day Effect on Demand Objective: Check if there is a significant difference in cycle demand between working and non-working days. Hypotheses: H 0 H 0 : Working day has no effect on demand. H 1 H 1 : Working day affects demand.

```
# Subset demand based on working day status
working_day_count = df[df['workingday'] == 1]['count']
non_working_day_count = df[df['workingday'] == 0]['count']

# Shapiro-Wilk test for normality
stats.shapiro(working_day_count), stats.shapiro(non_working_day_count)
```

```
F:\anaconda\Lib\site-packages\scipy\stats\_axis_nan_policy.py:531:
UserWarning: scipy.stats.shapiro: For N > 5000, computed p-value may
not be accurate. Current N is 7412.
  res = hypotest_fun_out(*samples, **kwds)

(ShapiroResult(statistic=0.8702545795617624,
pvalue=2.2521124830019574e-61),
 ShapiroResult(statistic=0.885211755076074,
pvalue=4.4728547627911074e-45))
```

```
# T-Test
t_stat, p_value = stats.ttest_ind(working_day_count,
```

```
non_working_day_count, equal_var=False)
print("T-Test Results:", t_stat, p_value)
```

```
T-Test Results: 1.2362580418223226 0.21640312280695098
```

1. ANOVA for Seasonal and Weather Effects on Demand Objective: Check if mean cycle demand is different across seasons and weather. Hypotheses for Season: H 0 H 0 : Mean demand is the same across all seasons. H 1 H 1 : Mean demand differs across seasons.

```
anova_season = stats.f_oneway(df[df['season'] == 1]['count'],
                              df[df['season'] == 2]['count'],
                              df[df['season'] == 3]['count'],
                              df[df['season'] == 4]['count'])
print("ANOVA Results for Season:", anova_season)
```

```
ANOVA Results for Season: F_onewayResult(statistic=236.94671081032106,
pvalue=6.164843386499654e-149)
```

```
anova_weather = stats.f_oneway(df[df['weather'] == 1]['count'],
                               df[df['weather'] == 2]['count'],
                               df[df['weather'] == 3]['count'],
                               df[df['weather'] == 4]['count'])
print("ANOVA Results for Weather:", anova_weather)
```

```
ANOVA Results for Weather: F_onewayResult(statistic=65.53024112793271,
pvalue=5.482069475935669e-42)
```

1. Chi-Square Test for Dependence of Weather on Season Objective: Test if weather conditions depend on the season. Hypotheses: H 0 H 0 : Weather is independent of season. H 1 H 1 : Weather depends on season

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

chi2, p, dof, ex = stats.chi2_contingency(contingency_table)
print("Chi-Square Test Results:", chi2, p)
```

```
Chi-Square Test Results: 49.158655596893624 1.549925073686492e-07
```