Importing Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import statsmodels.api as sm
from statsmodels.graphics.gofplots import qqplot
import scipy.stats as stats
from scipy.stats import ttest_ind, ttest_1samp, ttest_rel,
chi2_contingency, chisquare,f_oneway, levene, shapiro, boxcox
%matplotlib inline
import os
```

Downloading the given Dataset

```
!gdown
"https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/42
8/original/bike_sharing.csv?1642089089"

Downloading...
From:
https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428
/original/bike_sharing.csv?1642089089
To: /content/bike_sharing.csv?1642089089
   0% 0.00/648k [00:00<?, ?B/s] 100% 648k/648k [00:00<00:00, 19.9MB/s]
```

Problem Statement

The objective is to comprehend the factors influencing shared electric cycle demand in the Indian market and identify significant predictors. Yulu, grappling with revenue decline due to customer attrition, examines variables like temperature, windspeed, season, and weather from the dataset. The analysis aims to assess their impact on the target attribute, i.e., the Number of cycles rented. It also explores the influence of factors such as user type (registered or casual) and days (working day or holiday), along with weather and season. The study seeks to pinpoint key predictors for demand in the Indian market.

Constructing a DataFrame by reading the provided CSV file

```
df = pd.read_csv("/content/bike_sharing.csv?1642089089")
df

                  datetime  season  holiday  workingday  weather
temp  \
0      2011-01-01 00:00:00       1        0           0        1
9.84
1      2011-01-01 01:00:00       1        0           0        1
```

```
9.02
2      2011-01-01 02:00:00       1        0           0        1
9.02
3      2011-01-01 03:00:00       1        0           0        1
9.84
4      2011-01-01 04:00:00       1        0           0        1
9.84
...                    ...     ...      ...         ...      ...    ..
.
10881  2012-12-19 19:00:00       4        0           1        1
15.58
10882  2012-12-19 20:00:00       4        0           1        1
14.76
10883  2012-12-19 21:00:00       4        0           1        1
13.94
10884  2012-12-19 22:00:00       4        0           1        1
13.94
10885  2012-12-19 23:00:00       4        0           1        1
13.12

        atemp  humidity  windspeed  casual  registered  count
0      14.395        81     0.0000       3          13     16
1      13.635        80     0.0000       8          32     40
2      13.635        80     0.0000       5          27     32
3      14.395        75     0.0000       3          10     13
4      14.395        75     0.0000       0           1      1
...       ...       ...        ...     ...         ...    ...
10881  19.695        50    26.0027       7         329    336
10882  17.425        57    15.0013      10         231    241
10883  15.910        61    15.0013       4         164    168
10884  17.425        61     6.0032      12         117    129
10885  16.665        66     8.9981       4          84     88

[10886 rows x 12 columns]
```

The dataset comprises 10,886 rows and 12 columns, each representing attributes to be analyzed in order to identify the most relevant factors impacting the revenue of Yulu Bike sharing company.

```
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
```

Upon analyzing the data, it is evident that the dataset does not contain any NULL values. The dataset consists of a total of 10,886 rows and 12 columns. Except for the 'datetime' attribute, all other attributes are of integer or floating-point data types.

```
df.describe()

             season       holiday    workingday       weather
temp  \
count  10886.000000  10886.000000  10886.000000  10886.000000
10886.00000
mean       2.506614      0.028569      0.680875      1.418427
20.23086
std        1.116174      0.166599      0.466159      0.633839
7.79159
min        1.000000      0.000000      0.000000      1.000000
0.82000
25%        2.000000      0.000000      0.000000      1.000000
13.94000
50%        3.000000      0.000000      1.000000      1.000000
20.50000
75%        4.000000      0.000000      1.000000      2.000000
26.24000
max        4.000000      1.000000      1.000000      4.000000
41.00000

              atemp      humidity     windspeed        casual
registered  \
count  10886.000000  10886.000000  10886.000000  10886.000000
10886.000000
mean      23.655084     61.886460     12.799395     36.021955
155.552177
std        8.474601     19.245033      8.164537     49.960477
151.039033
min        0.760000      0.000000      0.000000      0.000000
```

```
0.000000
25%        16.665000     47.000000      7.001500      4.000000
36.000000
50%        24.240000     62.000000     12.998000     17.000000
118.000000
75%        31.060000     77.000000     16.997900     49.000000
222.000000
max        45.455000    100.000000     56.996900    367.000000
886.000000

              count
count  10886.000000
mean     191.574132
std      181.144454
min        1.000000
25%       42.000000
50%      145.000000
75%      284.000000
max      977.000000
```

From the provided descriptive table, we deduce the following insights:

The median temperature is 20.5 degrees Celsius, with 75% of the dataset's temperatures recorded at around 26.24 degrees Celsius. The average temperature stands at 20.36 degrees Celsius.

Yulu observes a median of 145 counted users (combining casual and registered), with 75% of users totaling 284. The average count of users is approximately 191.574, and the highest count reaches 977.

Approximately 68% of the data instances correspond to working days, which aligns with the common pattern of increased public transportation usage on these days.

The average temperature is recorded as 20.23 degrees Celsius, with 20.5 degrees Celsius occurring half of the time.

```
df.nunique()

datetime      10886
season            4
holiday           2
workingday        2
weather           4
temp             49
atemp            60
humidity         89
windspeed        28
casual          309
registered      731
```

```
count          822
dtype: int64
```

```
#missing values
df.isna().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
```

It is evident that our dataset does not contain any missing values.

```
df.drop("datetime", axis = 1, inplace = True)
```

We observe that the columns 'season,' 'holiday,' 'workingday,' and 'weather' each possess unique values of 4, 2, 2, and 4, respectively. Thus, we will convert these four columns into 'category'

```
#changing it from object dtype to category to save memory
df["season"]=df["season"].astype("category")
df["holiday"]=df["holiday"].astype("category")
df["workingday"]=df["workingday"].astype("category")
df["weather"]=df["weather"].astype("category")

cat_cols = df.dtypes == 'category'
cat_cols = list(cat_cols[cat_cols].index)
cat_cols
```

```
['season', 'holiday', 'workingday', 'weather']
```

Gathering all categorical variables into a single array.

```
nume_cols = df.dtypes != "category"
nume_cols = list(nume_cols[nume_cols].index)
nume_cols
```

```
['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered',
'count']
```

Gathering all numerical variables into a single array.

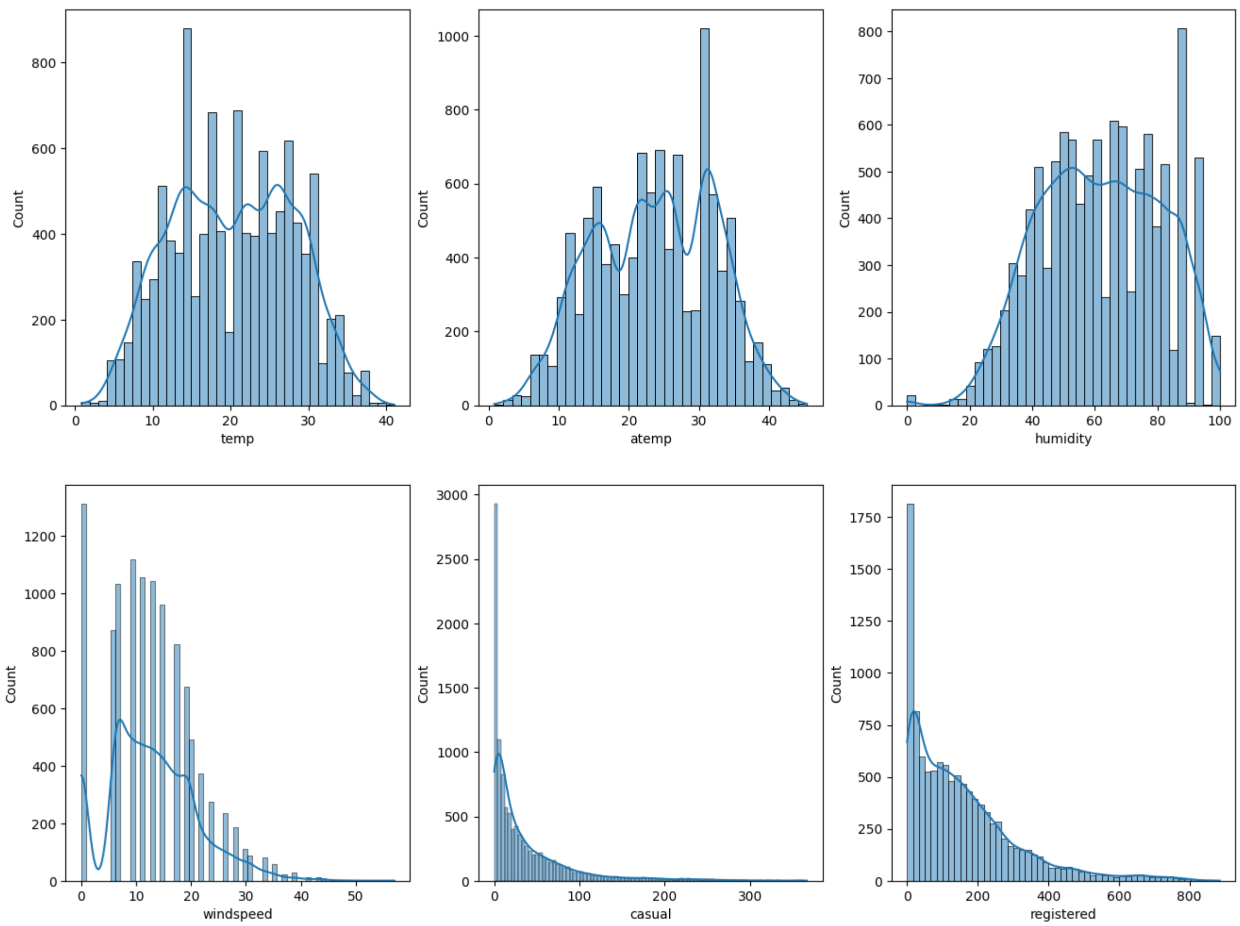
```
for i in df.columns:
 print(f'{i} has {df[i].nunique()} unique values')
 print("")
```

```
season has 4 unique values

holiday has 2 unique values

workingday has 2 unique values

weather has 4 unique values

temp has 49 unique values

atemp has 60 unique values

humidity has 89 unique values

windspeed has 28 unique values

casual has 309 unique values

registered has 731 unique values

count has 822 unique values
```

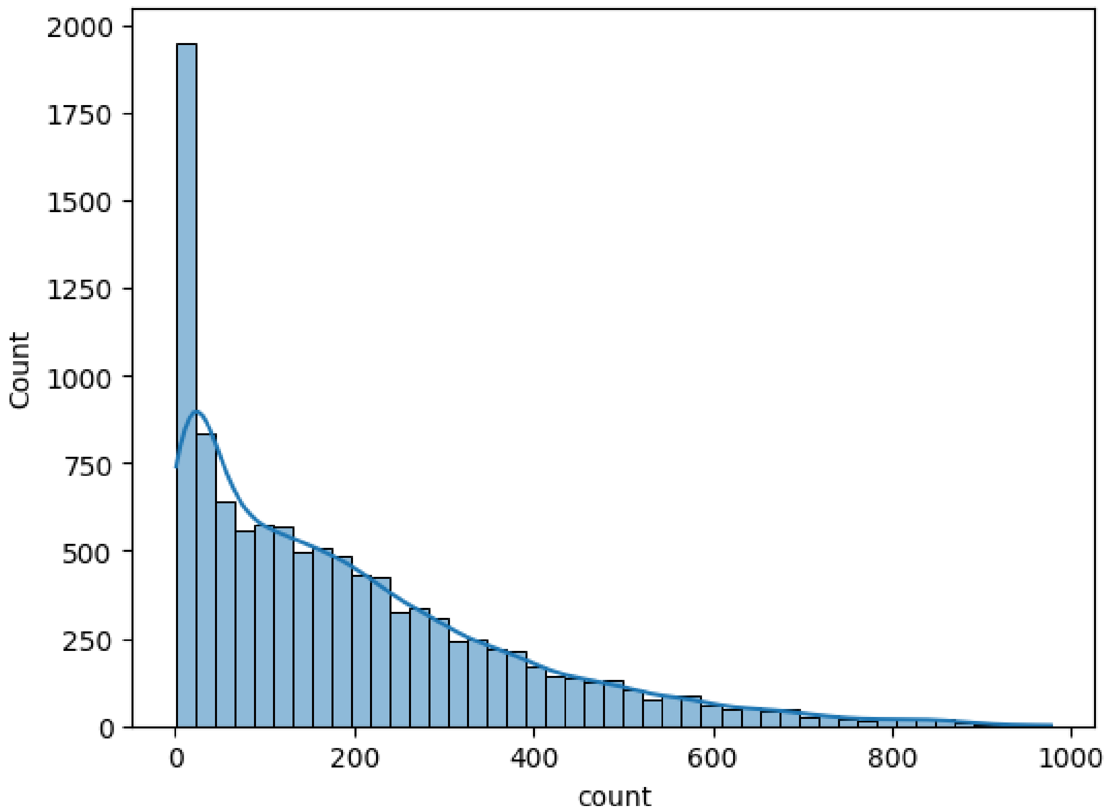
Univariate Analysis

Conducting univariate analysis on continuous variables including atemp, temp, humidity, windspeed, casual, registered, and count. Visualizing the distribution with displot for each variable.

```
#nume_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual',
'registered','count']
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))
index = 0
for row in range(2):
 for col in range(3):
  sns.histplot(df[nume_cols[index]], ax=axis[row, col], kde=True)
  index += 1

plt.show()
sns.histplot(df[nume_cols[-1]], kde=True)
plt.show()
```



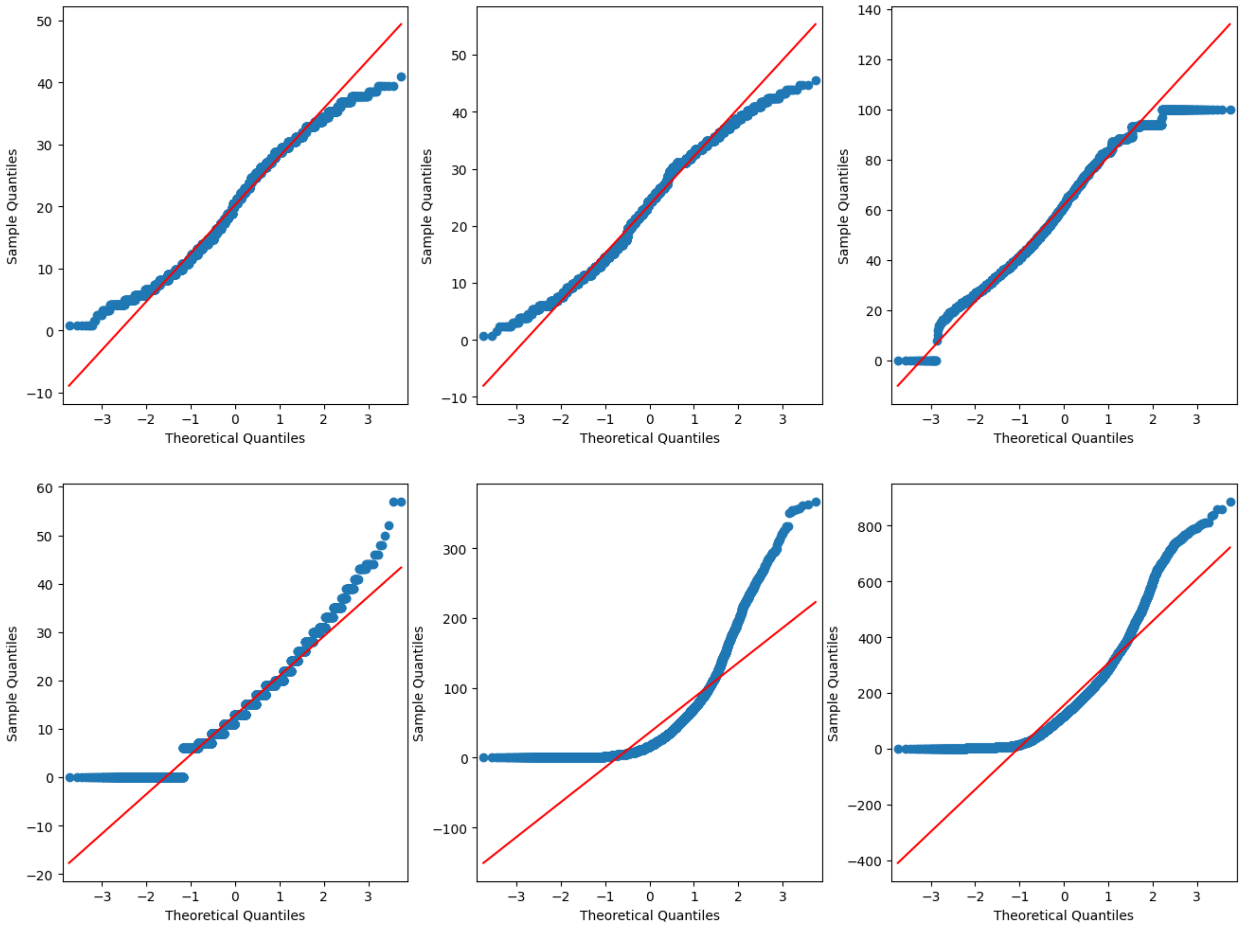
Count
temp
atemp
humidity
windspeed
casual
registered



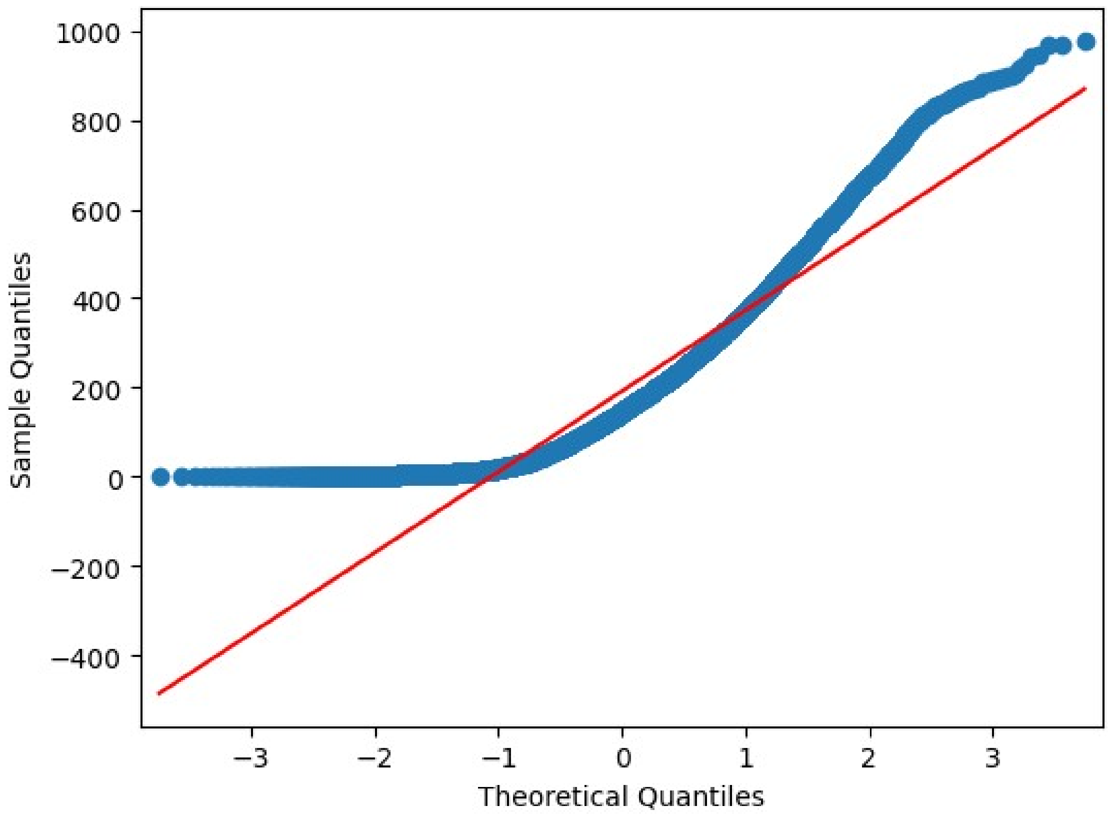
From the preceding histplots, several observations can be made:

1. The distributions of casual, registered, and count variables exhibit characteristics resembling a Log Normal Distribution.
2. The variables temp, atemp, and humidity display distributions that align with the Normal Distribution.
3. The distribution of windspeed can be approximated by a binomial distribution.

```
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))
index = 0
for row in range(2):
 for col in range(3):
  qqplot(df[nume_cols[index]], line="s", ax=axis[row, col])
  index += 1

qqplot(df[nume_cols[-1]], line = "s")
plt.show()
```



Sample Quantiles
Theoretical Quantiles
Sample Quantiles
Theoretical Quantiles
Sample Quantiles
Theoretical Quantiles
Sample Quantiles
Theoretical Quantiles
Sample Quantiles
Theoretical Quantiles
Sample Quantiles
Theoretical Quantiles

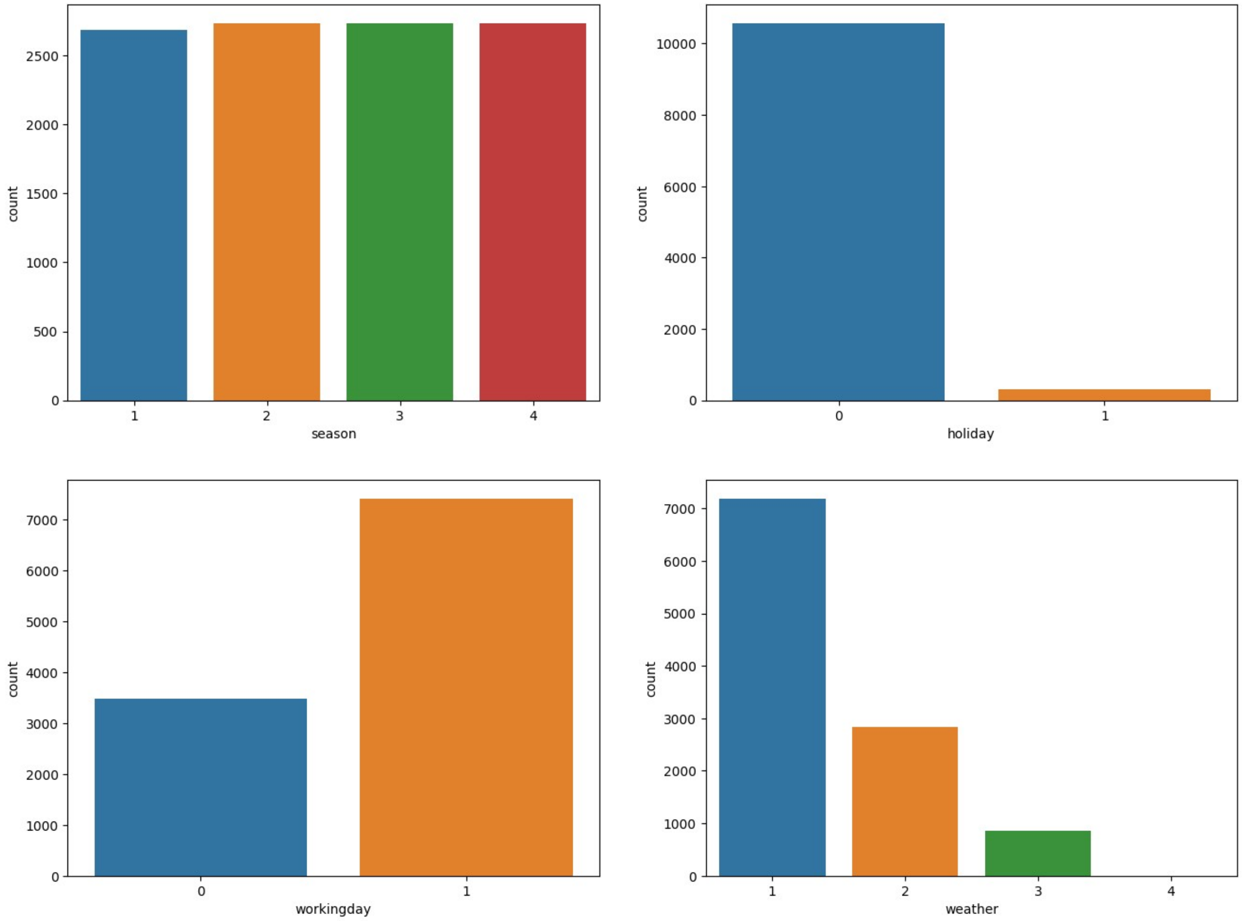


To assess the conformity with ANOVA assumptions, QQ plots of all the numerical attributes were plotted, yielding the following observations:

1. The QQ plots for casual, registered, and count variables, exhibiting characteristics resembling a Log Normal Distribution, do not align closely with the red "S" line.
2. QQ plots for temp, atemp, and humidity variables, which appear to follow the Normal Distribution, closely align with the red "S" line.
3. The QQ plot for windspeed, following a binomial distribution, does not closely align with the red "S" line.

Now countplots for categorical variables which are season holiday workingday and weather

```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))
index = 0
for row in range(2):
  for col in range(2):
    sns.countplot(data=df, x=cat_cols[index], ax=axis[row, col])
    index += 1
plt.show()
```

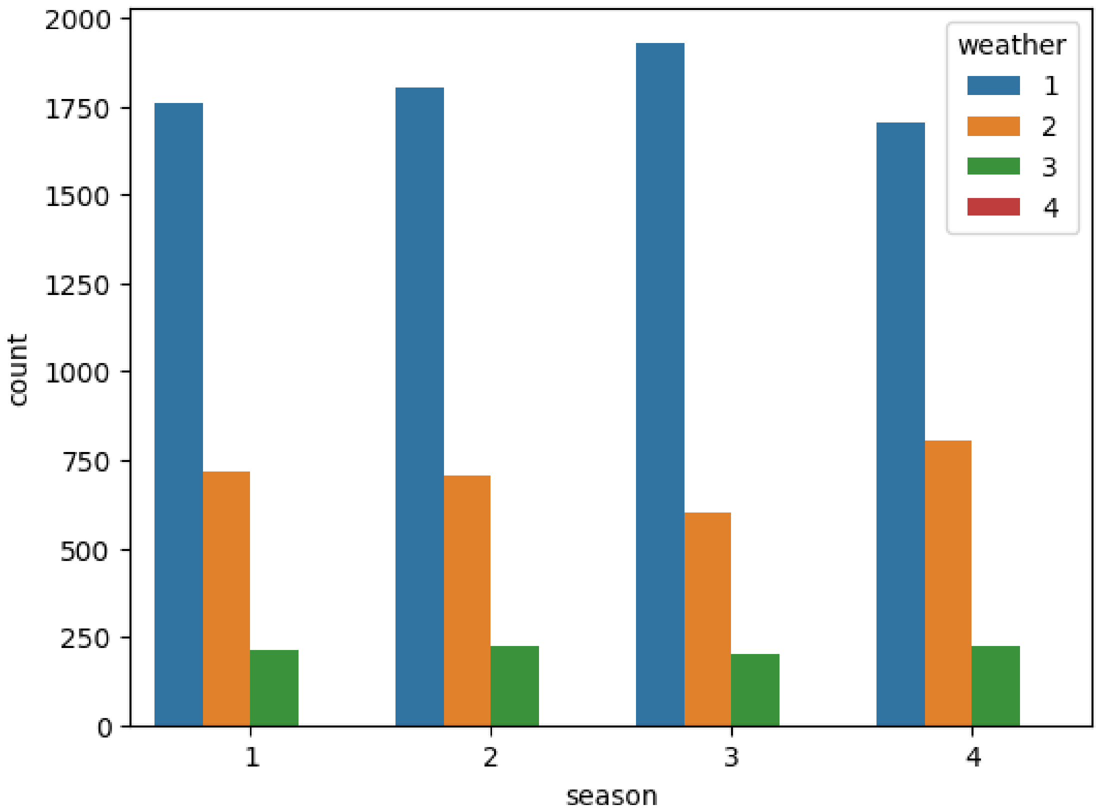


Based on the analysis of the four categorical variable countplots above, the following observations can be made:

1. The counts for each season are nearly the same, with negligible variations.
2. The count is higher on holidays compared to working days.
3. Graph 2 illustrates a significant imbalance between holiday and working day counts, reflecting the tendency for fewer people to use vehicles on holidays.
4. In the weather category, it's noticeable that weather 1 (clear weather) has the highest demand for bike rentals. The demand gradually decreases as weather conditions change to mist, light snow, and becomes almost negligible in heavy rain. This decline is likely due to safety concerns associated with biking in adverse weather.
5. Another categorical variable has been created to categorize the count of rented bicycles as low, medium, high, etc. This reveals a lognormal distribution, with the majority of cases falling into the "Low" category, and different levels of "High" counts for various reasons.
6. The data appears to be consistent with expectations, including an equal number of days in each season, a higher count of working days, and predominantly clear to partially cloudy weather conditions.

```
sns.countplot(x='season', hue='weather', data=df)
```

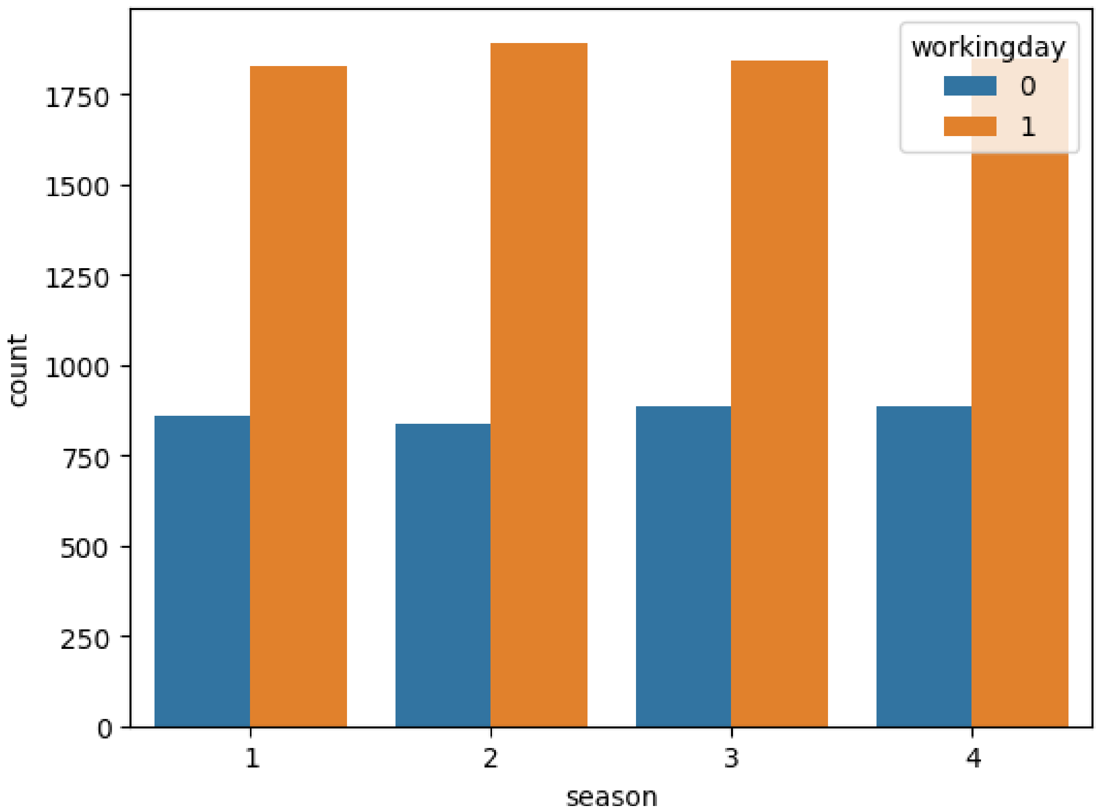
```
<Axes: xlabel='season', ylabel='count'>
```



The analysis of the plot indicates a noticeable influence of weather conditions on demand, regardless of the season. Specifically, the plot reveals a distinct pattern where clear weather conditions result in the highest demand, followed by mist and light snow conditions. Conversely, there is minimal to no demand observed during heavy rain conditions. This observation underscores the significant impact of weather on the demand for the shared electric cycles.

```
sns.countplot(x='season', hue='workingday', data=df)
```

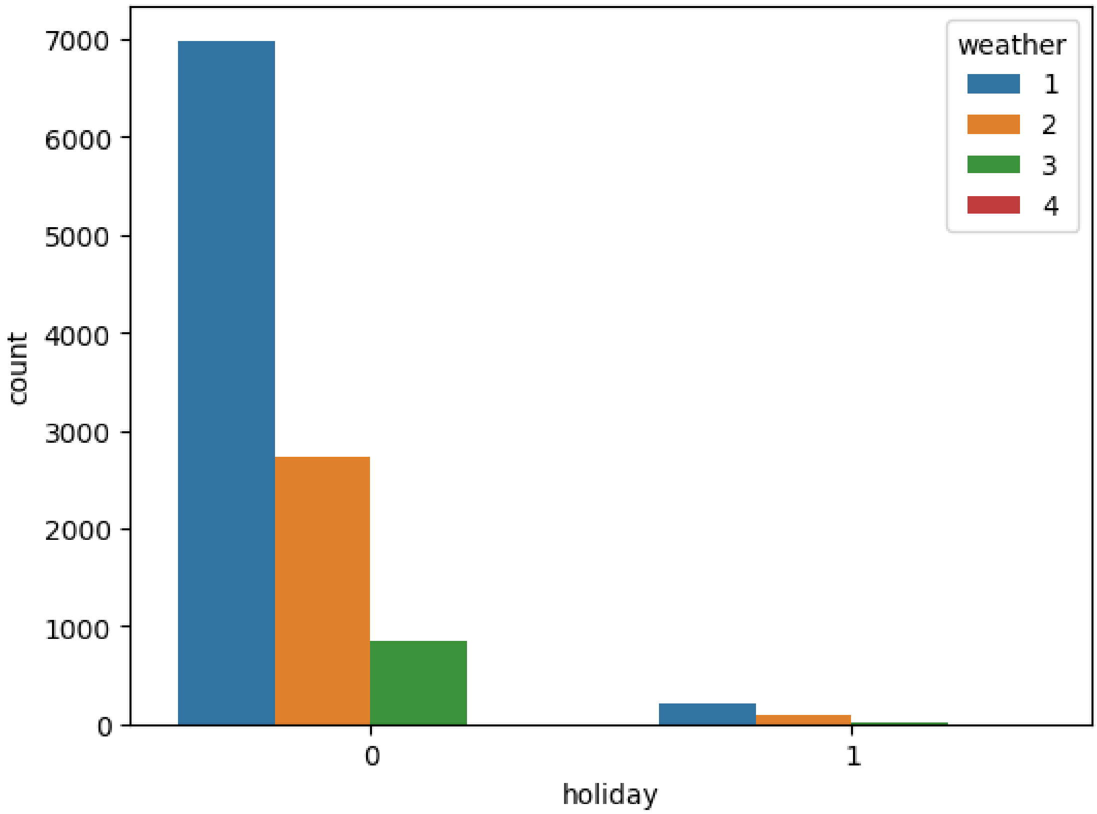
```
<Axes: xlabel='season', ylabel='count'>
```



Increased demand is observed on working days, likely due to employees utilizing the service for their commute to offices.

```
sns.countplot(x='holiday', hue='weather', data=df)
```

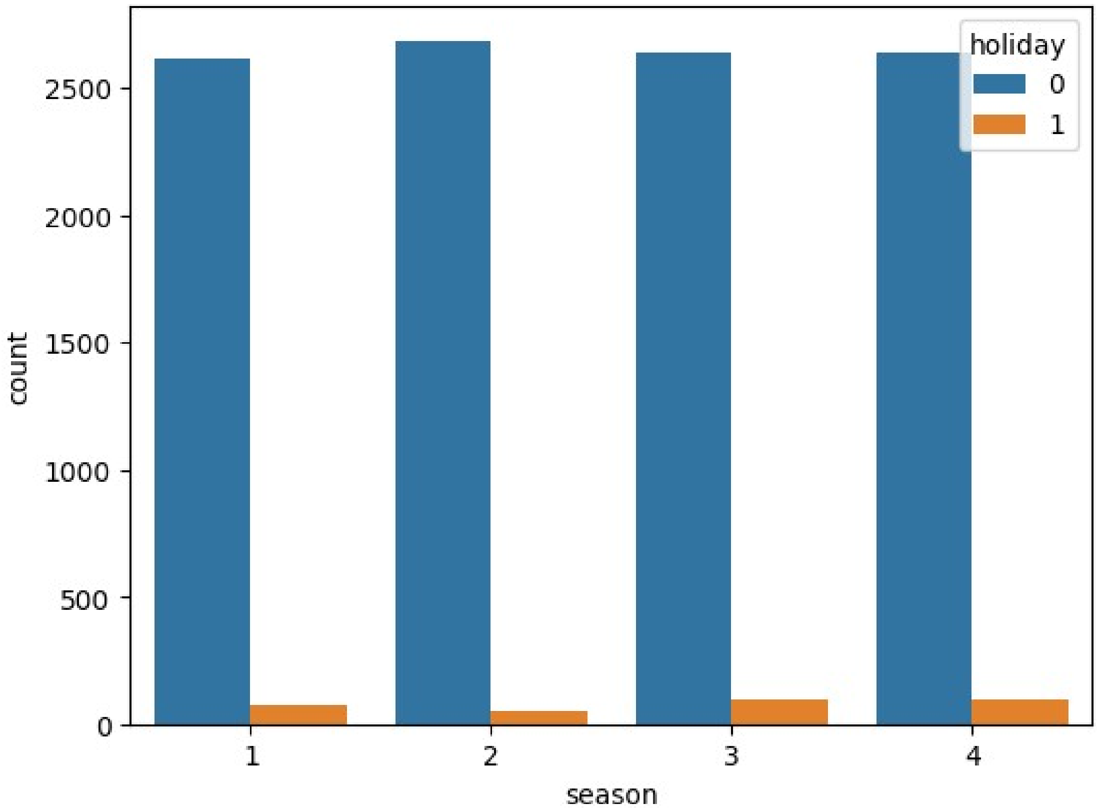
```
<Axes: xlabel='holiday', ylabel='count'>
```



Yulu bikes experience higher demand on weekdays, primarily because they serve as a convenient mode of transportation for individuals commuting to their workplaces.

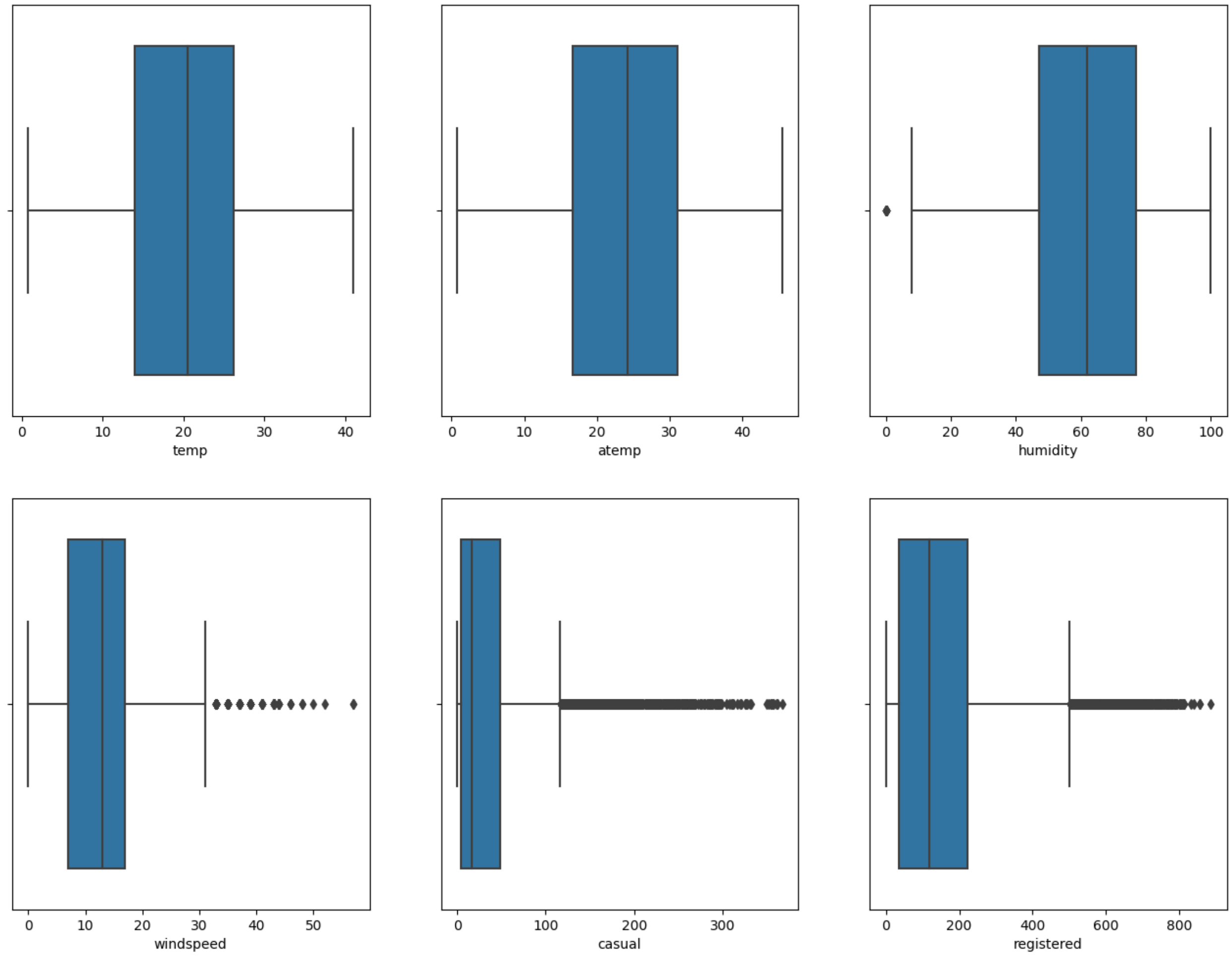
```
sns.countplot (x='season', hue='holiday', data=df)
```

```
<Axes: xlabel='season', ylabel='count'>
```

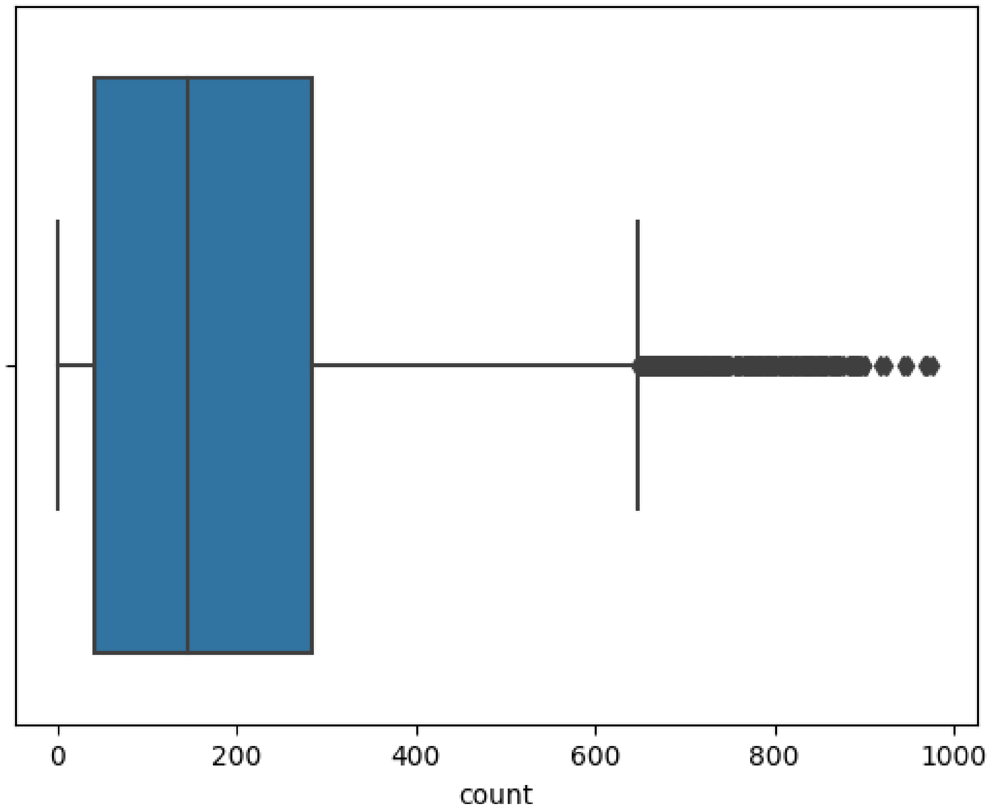


Throughout all seasons, the utilization of this service is predominantly observed on weekdays.

```
#Checking for outliers. Plotting boxplot for all the numerical 
columns.

fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))
index = 0
for row in range(2):
    for col in range(3):
        sns.boxplot(data=df, x=nume_cols[index], ax=axis[row, col])
        axis[row, col].set_xlabel(nume_cols[index])  # Set x-axis 
label
        index += 1

plt.show()
sns.boxplot(data=df, x=nume_cols[-1])
plt.show()
```



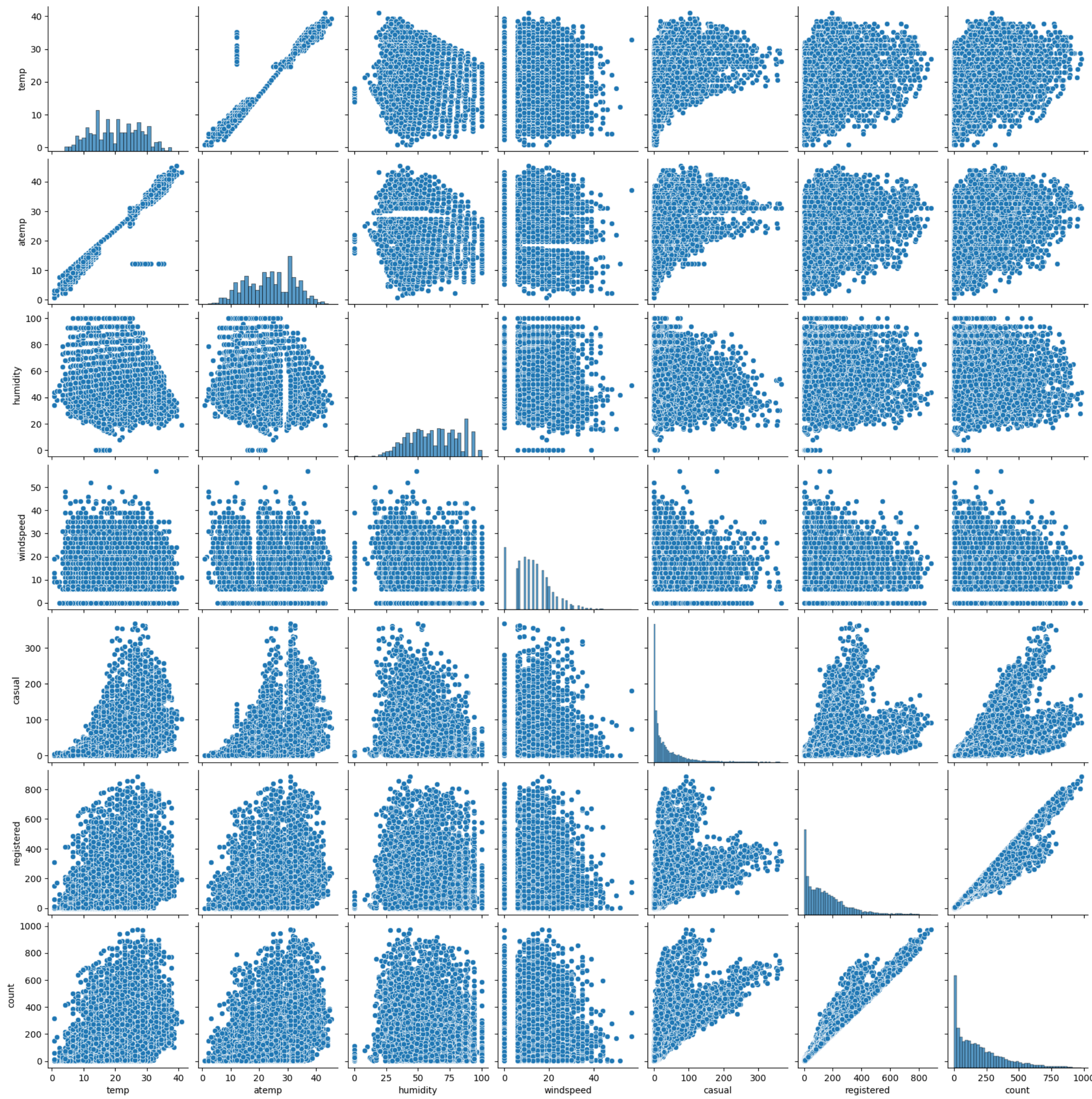
0 10 20 30 40
temp
0 10 20 30 40
atemp
0 20 40 60 80 100
humidity
0 10 20 30 40 50
windspeed
0 100 200 300
casual
0 200 400 600 800
registered



Here we observe be have outliers present for numerical columns such as count, windspeed, casual and registered.

Bin Count

```
bins=[0,40,100,200, 300, 500, 700, 900, 1000]
group=['Low','Average','medium', 'H1', 'H2', 'H3', 'H4' , 'Very high']

df['Rent_count']= pd.cut(df['count'],bins,labels=group) # Create new categorical column

sns.pairplot(df)

<seaborn.axisgrid.PairGrid at 0x7c72f4e3e080>
```

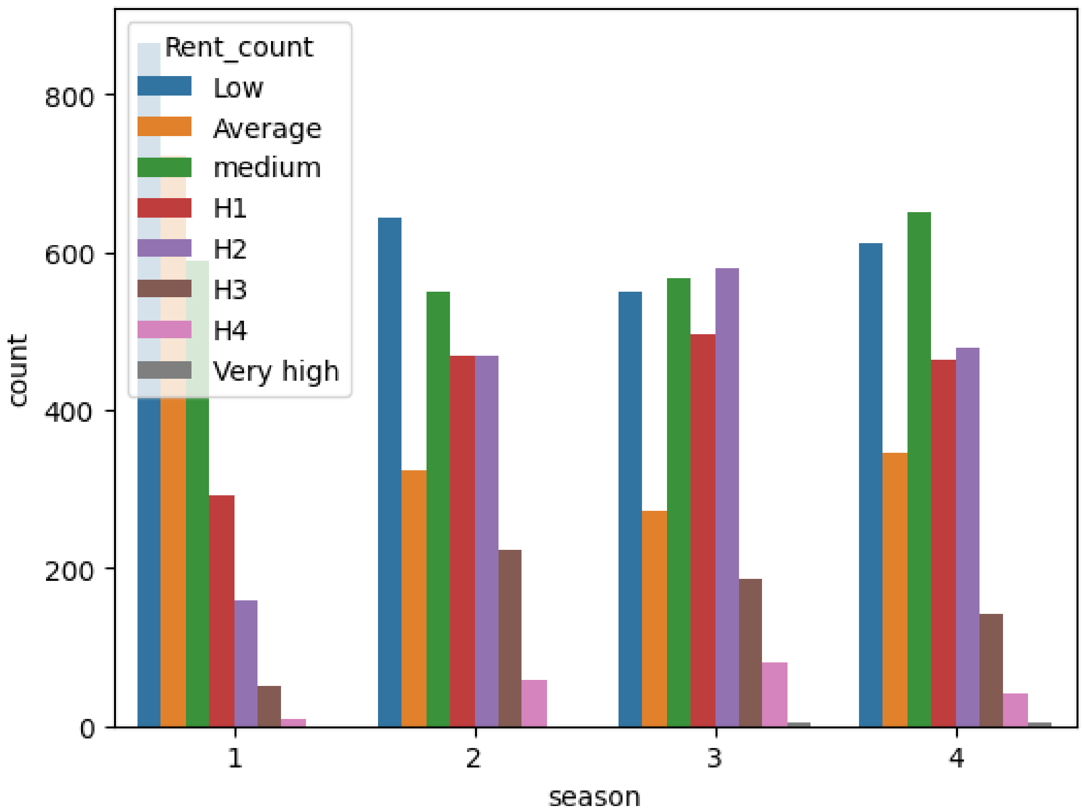


Bivariate Analysis

Examine the associations between key variables, including workday and count, season and count, as well as weather and count. As the "count" variable is continuous, it has been categorized into distinct groups such as "Rent_count Low," "Average," "Medium," "H1," "H2," "H3," "H4," and "Very High."

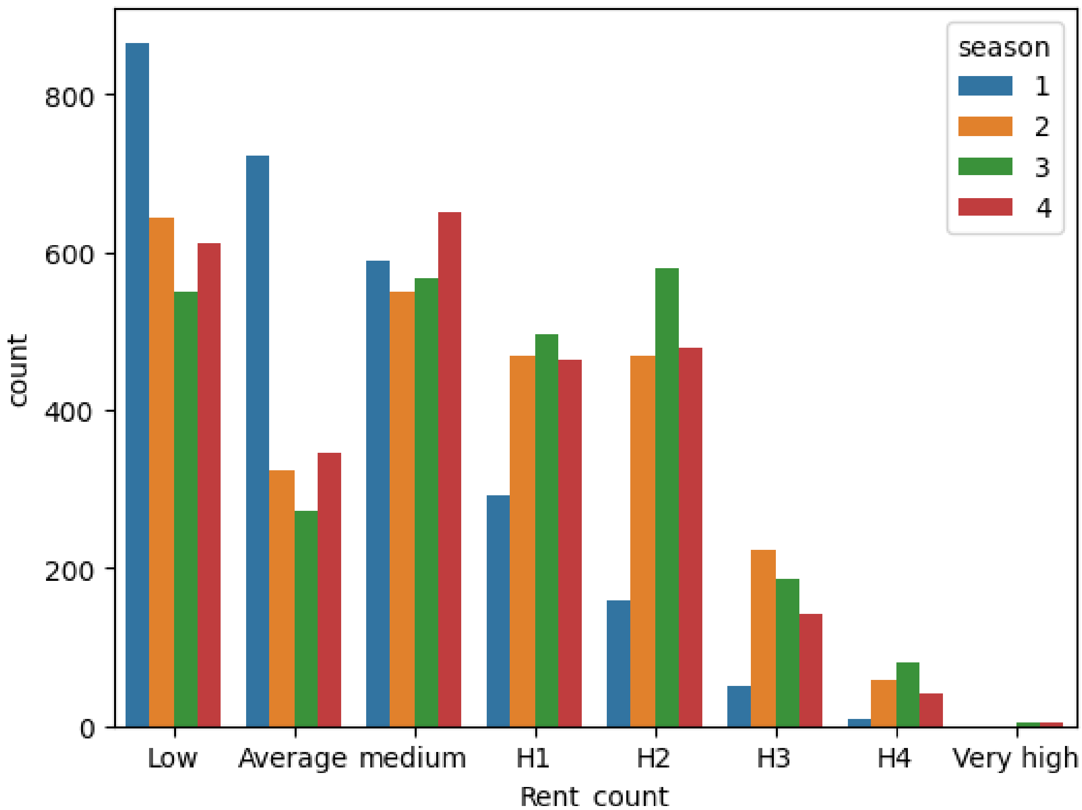
```
sns.countplot(x='season', hue='Rent_count', data=df)
```

```
<Axes: xlabel='season', ylabel='count'>
```



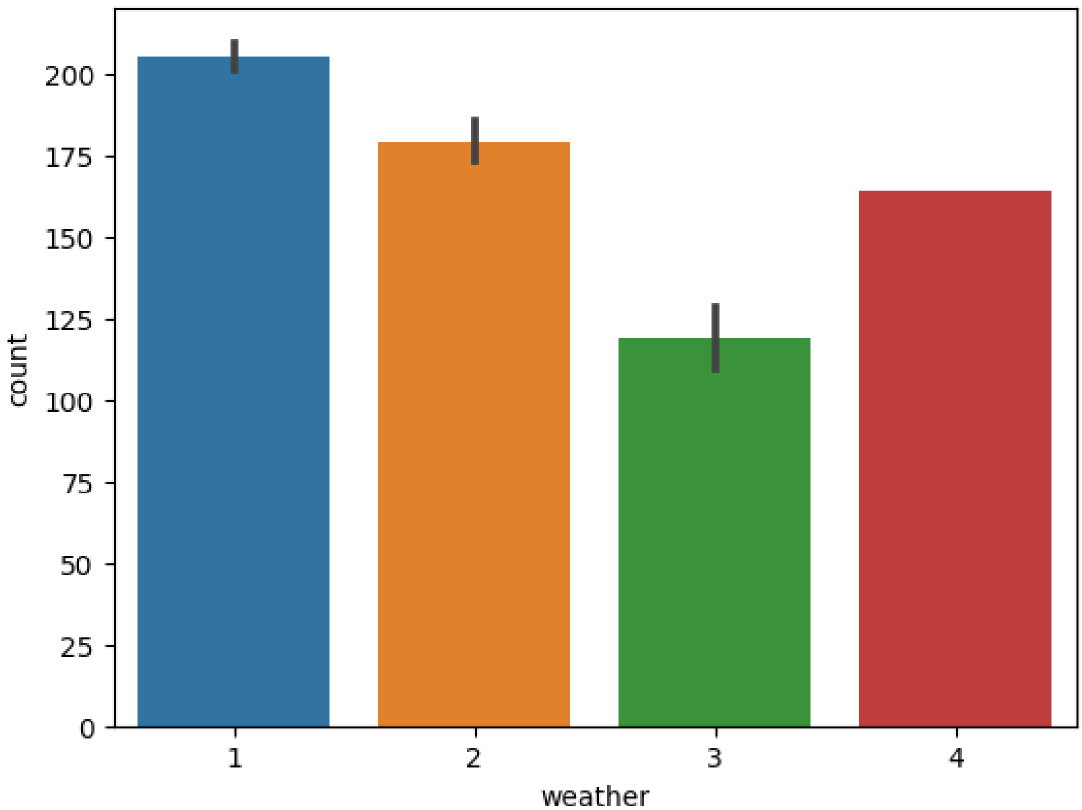

```
sns.countplot(x='Rent_count', hue='season', data = df)
```

```
<Axes: xlabel='Rent_count', ylabel='count'>
```



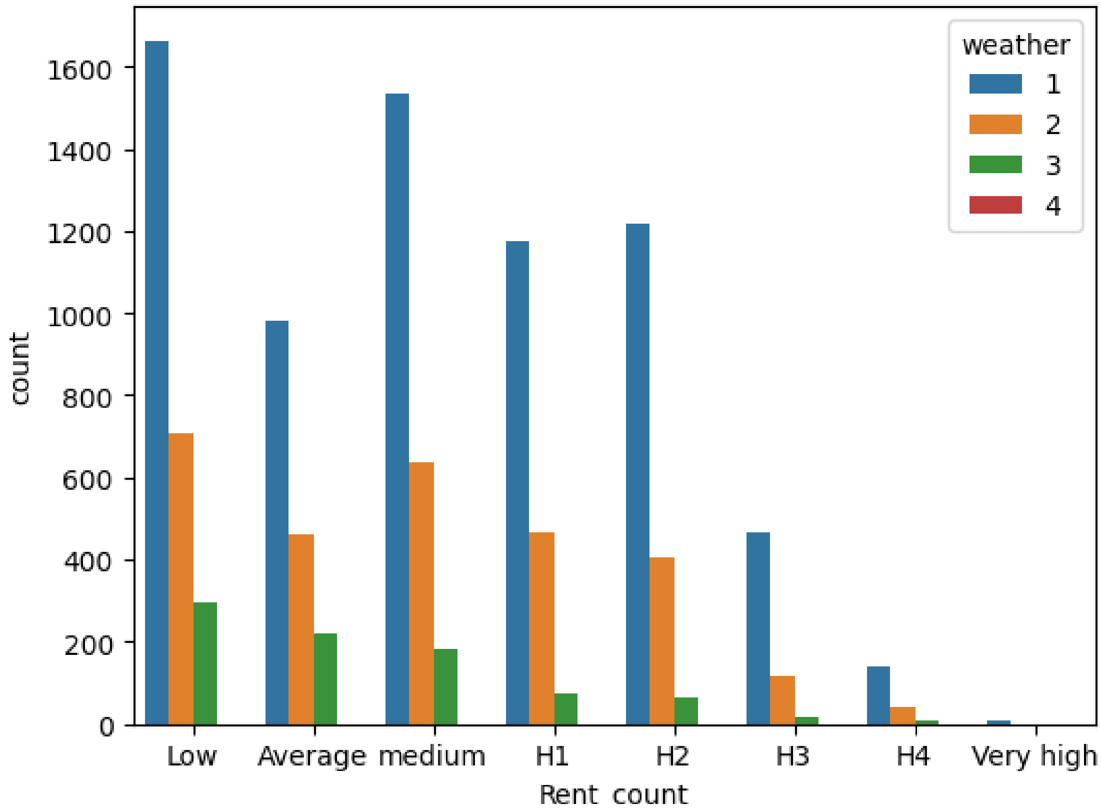

```
sns.barplot(x='weather', y='count',  data= df)
```

```
<Axes: xlabel='weather', ylabel='count'>
```



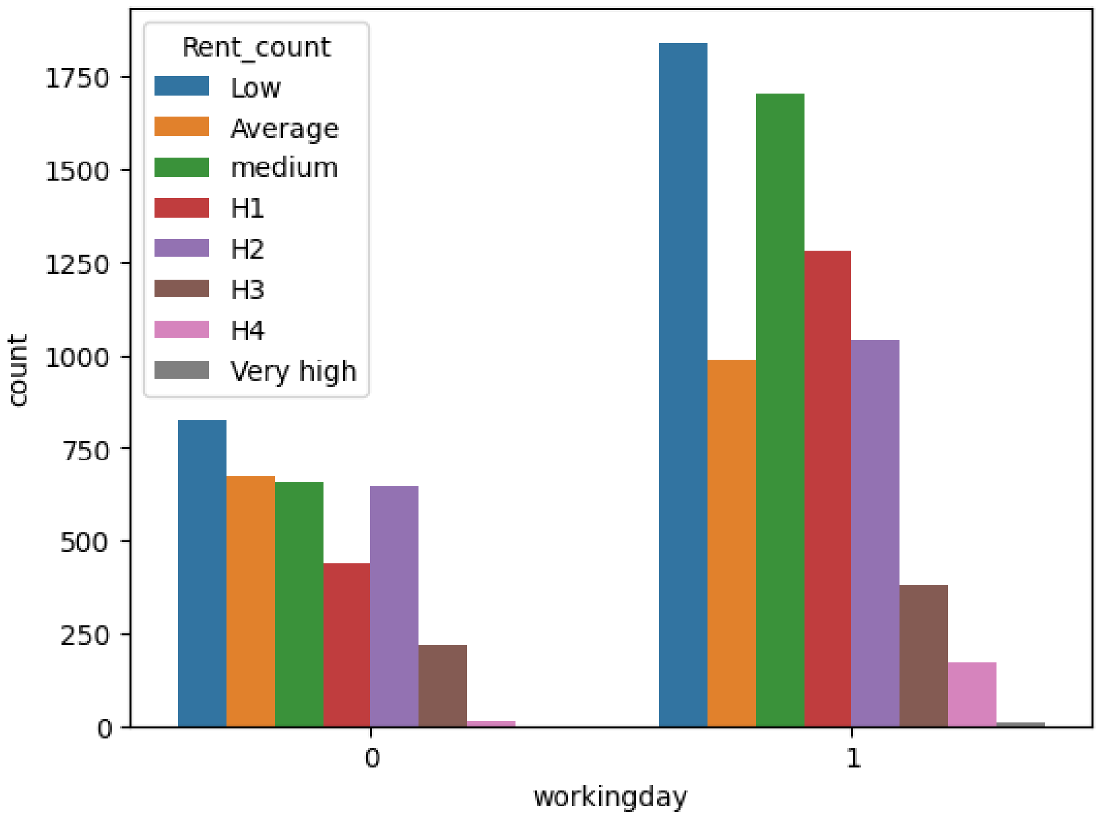

```
sns.countplot(x = 'Rent_count', hue='weather', data = df)
```

```
<Axes: xlabel='Rent_count', ylabel='count'>
```



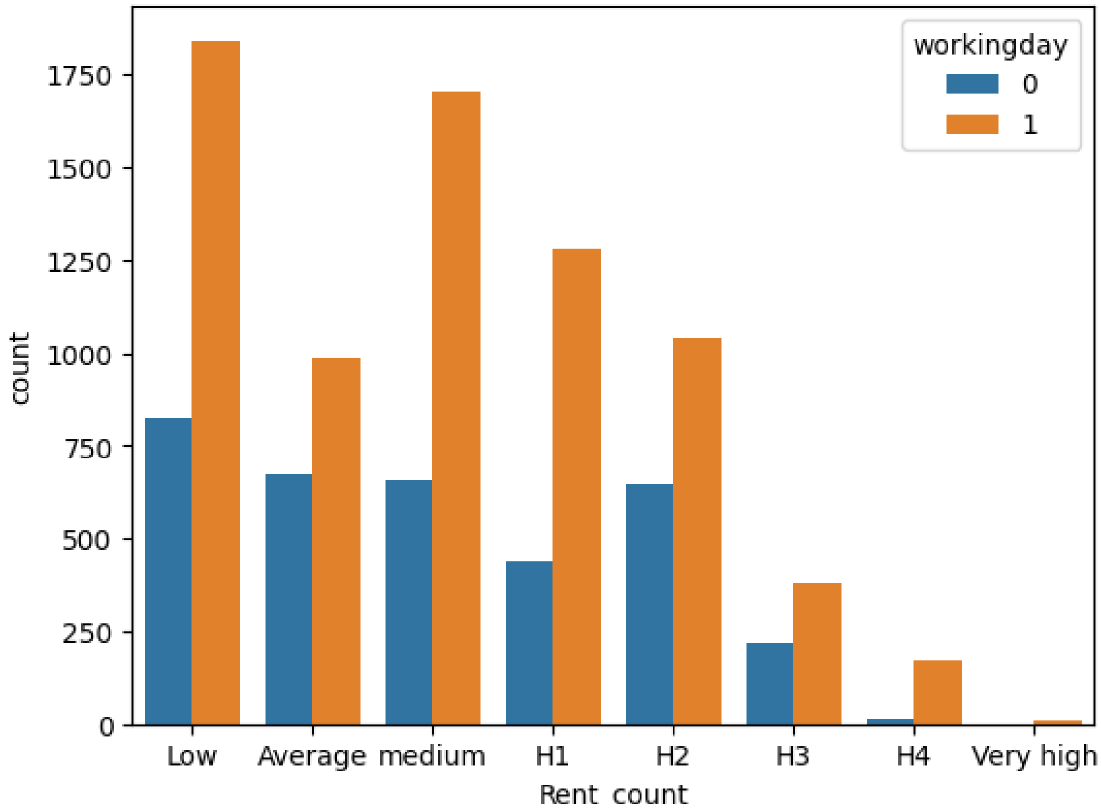

```
sns.countplot(x='workingday', hue='Rent_count', data= df)
```

```
<Axes: xlabel='workingday', ylabel='count'>
```



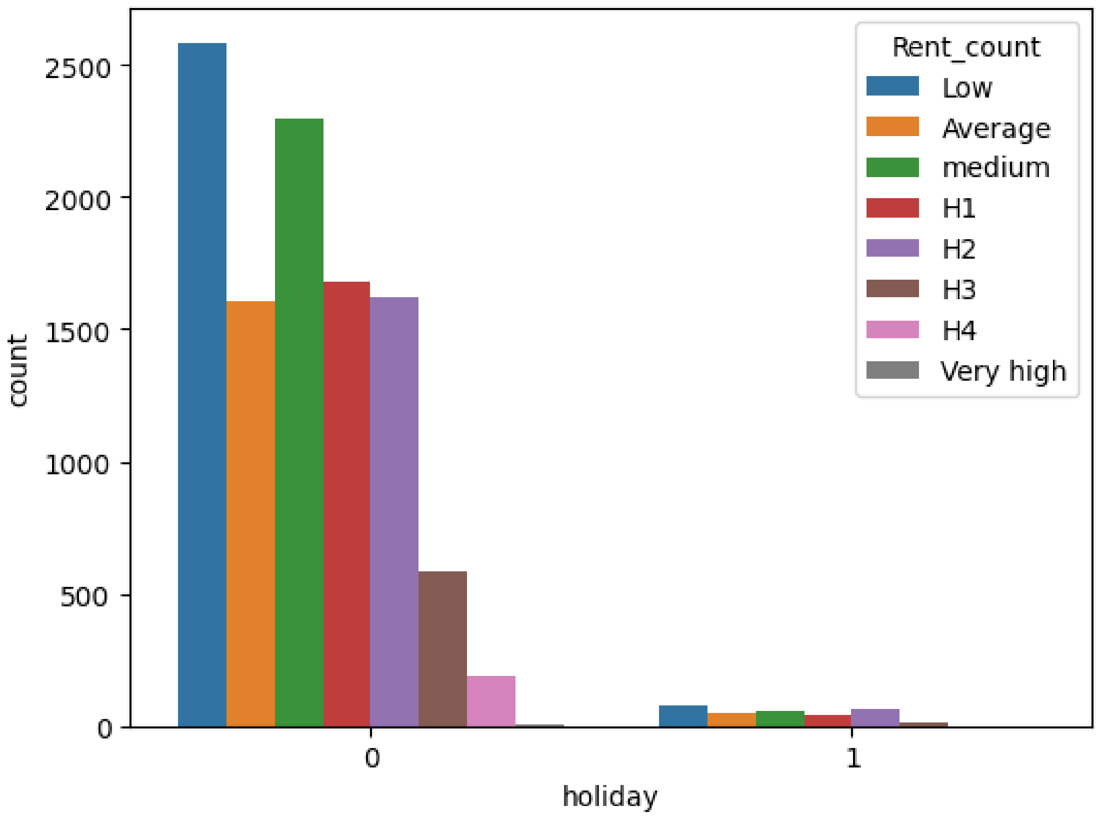

```
sns.countplot(x = 'Rent_count', hue='workingday',  data= df)
```

```
<Axes: xlabel='Rent_count', ylabel='count'>
```



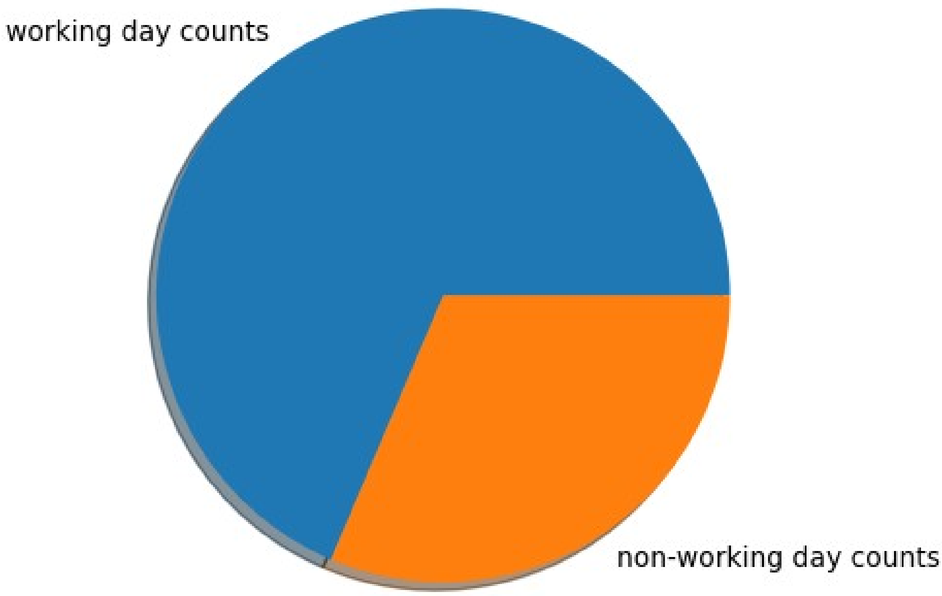

```
sns.countplot(x='holiday', hue='Rent_count', data=df)
```

```
<Axes: xlabel='holiday', ylabel='count'>
```




```
plt.pie([df.loc[df['workingday']==1]
['count'].sum(),df.loc[df['workingday']==0]['count'].sum()],
 labels=['working day counts','non-working day counts'],
 shadow=True
 )
```

```
([<matplotlib.patches.Wedge at 0x7c72f3601960>,
  <matplotlib.patches.Wedge at 0x7c72f3601870>],
 [Text(-0.6067654144600506, 0.9175160662435963, 'working day counts'),
  Text(0.6067653285559936, -0.9175161230530706, 'non-working day
counts')])
```



Present the findings derived from Exploratory Data Analysis (EDA) in terms of attribute ranges, outlier observations, variable distributions, and inter-variable relationships. Provide commentary on each univariate and bivariate plot with the following insights:

1. During the summer and fall seasons, there is a noticeable increase in the rental of bikes in comparison to other seasons.
2. Instances of increased bike rentals are evident during holidays.
3. The analysis of working days also indicates a slight rise in bike rentals during holidays or weekends.
4. Reduced bike rentals are observed during adverse weather conditions such as rain, thunderstorms, snow, or fog.

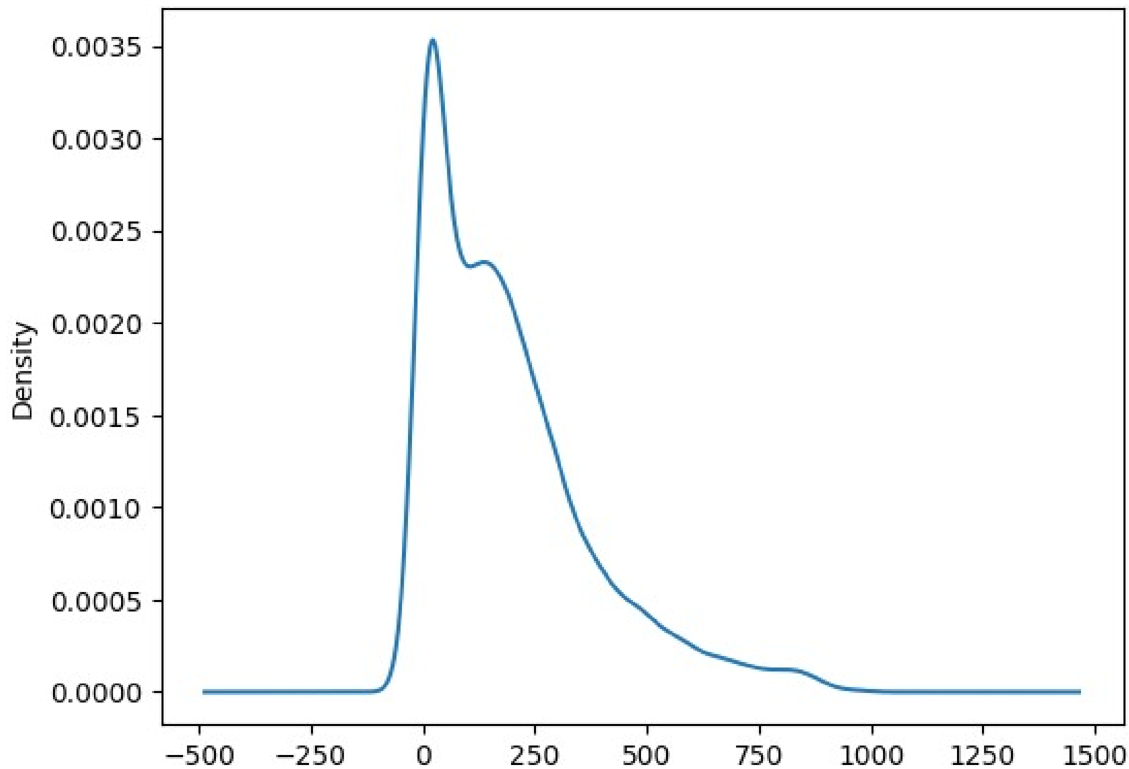
Evaluate the assumptions of the statistical test, encompassing normality and equal variance. Employ tools such as Histograms, Q-Q plots, or statistical assessments like Levene's test and optionally, the Shapiro-Wilk test. Continue the analysis even if certain assumptions are not met according to Levene's or Shapiro-Wilk tests, but ensure to corroborate with visual scrutiny and provide corresponding observations as needed.

2 sample t test

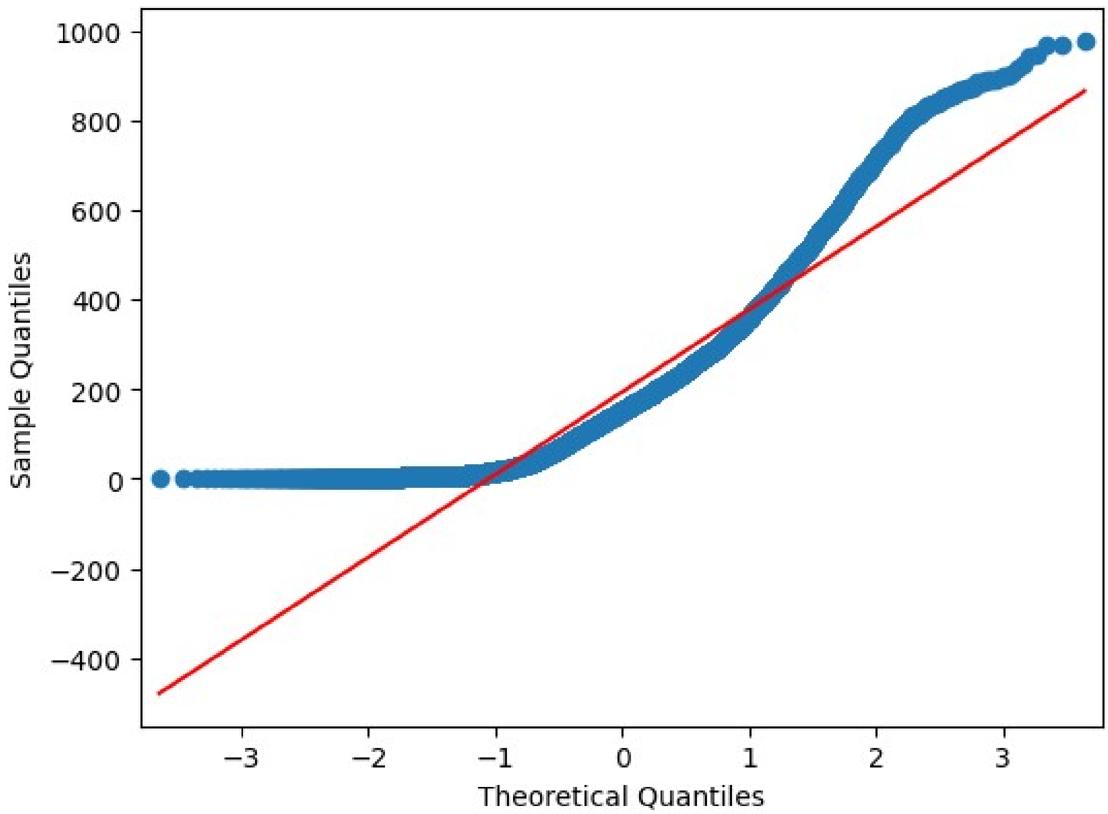
Perfoming 2 sample t test on working day and non working day counts. Taking significant level(alpha) as 0.05 for all test. considreing: Null hypothesis Ho = mean of count of bike on non working day is equal to mean of counts of bike on working day. Alternate hypothesis Hn = mean of count of bike on non working day is not equal to mean of counts of bike on working day.

```
df.loc[df['workingday']==1]['count'].plot(kind='kde')
```

```
<Axes: ylabel='Density'>
```




```
x=df.loc[df['workingday']==1]['count']
sm.qqplot(x, dist=stats.norm, line='s');
```




```
#The distribution does not follows normal distribution
df1=df.loc[df['workingday']==1]['count'].reset_index()
df1.drop(['index'], axis=1, inplace=True)
df2=df.loc[df['workingday']==0]['count'].reset_index()
df2.drop(['index'], axis=1, inplace=True)
ttest,p_value=ttest_ind(df1,df2)
print("p_value = ",p_value)

p_value =  [0.22644804]
```

As the calculated p-value exceeds the threshold of 0.05, the null hypothesis cannot be rejected. Consequently, it can be concluded that non-working days do not exert a significant impact on bike counts.

Hypothesis Testing

2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented ANNOVA to check if No. of cycles rented is similar or different in different 1. weather 2. season Chi-square test to check if Weather is dependent on the season

```
t_stat, p_value = levene(df["count"],df["workingday"])
p_value
alpha = 0.5
```

2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented H0 = There is no effect of Working Day on the number of electic cycles rented. Ha = There is an effect of WorkingDay on the number of electric cycles rented. Right/Left/Two_tailed Test_statistic Using ttest_ind

```
ttest_ind(df["count"], df["workingday"])
```

```
Ttest_indResult(statistic=109.95076974934595, pvalue=0.0)
```

```
population_mean_count = df["count"].mean()
population_mean_count
```

```
191.57413191254824
```

Select an appropriate test to check whether:

1. Working Day has effect on number of electric cycles r of cycles rented similar or different ented
2. No.in different seasons
3. No. of cycles rented similar or different in different weather
4. Weather is dependent on season (check between 2 predictor variable) First 3 statements to chk are having one Numerical variable i.e. Count and one Categorical_variable as working Day or seasons or Weather. So For these type of questions we use ttest or Anova i.e (Numeric, catagorical) 4th one is both the categorical variables so use Chisquare or chi2_contingency test

```
#1.Working Day has effect on number of electric cycles rented
population_mean_count = df["count"].mean()
population_mean_count
```

```
191.57413191254824
```

```
df_workingday_count = df[df["workingday"] == 1]["count"]
df_workingday_count.mean()
```

```
193.01187263896384
```

```
df_non_workingday_count = df[df["workingday"] == 0]["count"]
df_non_workingday_count.mean()
```

```
188.50662061024755
```

Using ANOVA

```
#H0 = Working day does not have any effect on number of cycles rented.
#HA = Working day has an positive effect on number of cycles rented.
i.e. mu1 > mu2
# We consider it to be Right Tailed
#Test Statistic and p_value
#We will consider alpha as 0.01 significance value. i.e 99% confidence
alpha = 0.01
```

```
f_stat, p_value =
f_oneway(df_workingday_count,df_non_workingday_count)
print(f"Test statistic = {f_stat} pvalue = {p_value}")
if (p_value < alpha):
 print("Reject Null Hypothesis")
else:
 print("Fail to reject Null Hypothesis")

Test statistic = 1.4631992635777575 pvalue = 0.22644804226428558
Fail to reject Null Hypothesis
```

Using ttest

```
#H0 = Working day does not have any effect on number of cycles rented.
#HA = Working day has an effect on number of cycles rented. mu1 > m2
# We consider it to be Righ Tailed.
#Test Statistic and p_value
#We will consider alpha as 0.01 significance value. i.e 99% confidence
alpha = 0.01
t_stat, p_value =
ttest_ind(df_workingday_count,df_non_workingday_count, alternative =
"greater")
print(f"Test statistic = {t_stat} pvalue = {p_value}")
if (p_value < alpha):
 print("Reject Null Hypothesis")
else:
 print("Fail to reject Null Hypothesis")

Test statistic = 1.2096277376026694 pvalue = 0.11322402113180674
Fail to reject Null Hypothesis
```

2.No. of cycles rented similar or different in different seasons

Considering the presence of four distinct seasons, a t-test is not applicable in this scenario. Instead, an analysis of variance (ANOVA) should be employed for conducting the statistical comparison.

```
df_season1_spring = df[df["season"] == 1]["count"]
df_season1_spring_subset = df_season1_spring.sample(100)

df_season2_summer =df[df["season"] == 2]["count"]
df_season2_summer_subset = df_season2_summer.sample(100)

df_season3_fall = df[df["season"] == 3]["count"]
df_season3_fall_subset = df_season3_fall.sample(100)

df_season4_winter = df[df["season"] == 4]["count"]
df_season4_winter_subset = df_season4_winter.sample(100)
```

We have extracted samples from each dataframe in order to subject them to the Shapiro-Wilk test.

```
#checking for assumptions:
#Levene's Test

#H0 = All samples have equal variance
#HA = At least one sample will have different variance
t_stat, p_value = levene(df_season1_spring, df_season2_summer,
df_season3_fall, df_season4_winter)
p_value
```

```
1.0147116860043298e-118
```

The Shapiro-Wilk test is employed to assess the normality of the data. In order to conduct this test effectively, we have extracted subsets of each dataset, each containing 100 values. This range of values (50 to 200) is considered suitable for the Shapiro-Wilk test.

```
#H0 = Sample is drawn from NormalDistribution
#HA = Sample is not from Normal Distribution
##Here we are considering alpha (significance value as ) 0.05
t_stat, pvalue = shapiro(df_season1_spring_subset)
if pvalue < 0.05:
 print("Reject H0 Data is not Gaussian")
else:
 print("Fail to reject Data is Gaussian")
```

```
Reject H0 Data is not Gaussian
```

```
t_stat, pvalue = shapiro(df_season2_summer_subset)
if pvalue < 0.05:
 print("Reject H0 Data is not Gaussian")
else:
 print("Fail to reject Data is Gaussian")
```

```
Reject H0 Data is not Gaussian
```

```
t_stat, pvalue = shapiro(df_season2_summer_subset)
if pvalue < 0.05:
 print("Reject H0 Data is not Gaussian")
else:
 print("Fail to reject Data is Gaussian")
```

```
Reject H0 Data is not Gaussian
```

```
t_stat, pvalue = shapiro(df_season3_fall_subset)
if pvalue < 0.05:
 print("Reject H0 Data is not Gaussian")
else:
 print("Fail to reject Data is Gaussian")
```

```
Reject H0 Data is not Gaussian

t_stat, pvalue = shapiro(df_season4_winter_subset)
if pvalue < 0.05:
 print("Reject H0 Data is not Gaussian")
else:
 print("Fail to reject Data is Gaussian")

Reject H0 Data is not Gaussian
```

In all four of the aforementioned tests, the obtained p-values are extremely close to zero (approximately 10^-6 or similar), which is significantly lower than the designated alpha level. Consequently, we reject the Null Hypothesis for these sample distributions, indicating that they do not conform to a normal distribution.

Despite the non-compliance with the normal distribution assumption, we will proceed with the ANOVA test, adhering to the specifications outlined in the problem statement.

```
#H0 = season does not have any effect on number of cycles rented.
#HA = At least one season out of four (1:spring, 2:summer,3:fall,
4:winter) has an effect on number of cycles rented.
#Righ Tailed /Left/Two
#Test Statistic and p_value
#We will consider alpha as 0.01 significance value. i.e 99% confidence
alpha = 0.01
f_stat, p_value = f_oneway(df_season1_spring, df_season2_summer,
df_season3_fall, df_season4_winter)
print(f"Test statistic = {f_stat} pvalue = {p_value}")
if (p_value < alpha):
 print("Reject Null Hypothesis")
else:
 print("Fail to reject Null Hypothesis")

Test statistic = 236.94671081032106 pvalue = 6.164843386499654e-149
Reject Null Hypothesis
```

No. of cycles rented similar or different in different weather

Considering the presence of four distinct seasons, a t-test is not applicable in this scenario. Instead, an analysis of variance (ANOVA) should be employed for conducting the statistical comparison.

```
df_weather1_clear = df[df["weather"] == 1]["count"]
df_weather1_clear.mean()

205.23679087875416

df_weather2_Mist = df[df["weather"] == 2]["count"]
df_weather2_Mist.mean()
```

```
178.95553987297106
```

```
df_weather3_LightSnow = df[df["weather"] == 3]["count"]
df_weather3_LightSnow.mean()
```

```
118.84633294528521
```

```
df_weather4_HeavyRain = df[df["weather"] == 4]["count"]
df_weather4_HeavyRain.mean()
```

```
164.0
```

```
#checking for assumptions
#levene's Test = chexking for variance
```

```
#H0 = All samples have equal variance
#HA = At least one sample will have different variance
t_stat, p_value = levene(df_weather1_clear, df_weather2_Mist,
df_weather3_LightSnow, df_weather4_HeavyRain)
p_value
```

```
3.504937946833238e-35
```

Shapiro Test for normality

```
#H0 = Sample is drawn from NormalDistribution
#HA = Sample is not from Normal Distribution
##Here we are considering alpha (significance value as ) 0.05
shapiro(df_weather1_clear)
```

```
ShapiroResult(statistic=0.8909230828285217, pvalue=0.0)
```

```
shapiro(df_weather2_Mist)
```

```
ShapiroResult(statistic=0.8767687082290649, pvalue=9.781063280987223e-
43)
```

```
shapiro(df_weather3_LightSnow)
```

```
ShapiroResult(statistic=0.7674332857131958, pvalue=3.876090133422781e-
33)
```

```
#shapiro(df_weather4_HeavyRain)
df_weather4_HeavyRain
```

```
5631    164
Name: count, dtype: int64
```

using ANOVA

```
#H0 = weather does not have any effect on number of cycles rented.
#HA = At least one weather out of four (1: clear, 2: Mist, 3:Light
```

```
snow, 4:Heavy Rain) has an effect on number of cycles re
#Righ Tailed /Left/Two
#Test Statistic and p_value
#We will consider alpha as 0.01 significance value. i.e 99% confidence
alpha = 0.01
f_stat, p_value =
f_oneway(df_weather1_clear,df_weather2_Mist,df_weather3_LightSnow,df_w
eather4_HeavyRain)
print(f"Test statistic = {f_stat} pvalue = {p_value}")
if (p_value < alpha):
 print("Reject Null Hypothesis")
else:
 print("Fail to reject Null Hypothesis")
```

```
Test statistic = 65.53024112793271 pvalue = 5.482069475935669e-42
Reject Null Hypothesis
```

```
#H0 = weather does not have any effect on number of cycles rented.
#HA = At least one weather out of four (1: clear, 2: Mist, 3:Light
snow, 4:Heavy Rain) has an effect on number of cycles re
#Righ Tailed /Left/Two
#Test Statistic and p_value
#We will consider alpha as 0.01 significance value. i.e 99% confidence
alpha = 0.01
f_stat, p_value =
f_oneway(df_weather1_clear,df_weather2_Mist,df_weather3_LightSnow)
print(f"Test statistic = {f_stat} pvalue = {p_value}")
if (p_value < alpha):
 print("Reject Null Hypothesis")
else:
 print("Fail to reject Null Hypothesis")
```

```
Test statistic = 98.28356881946706 pvalue = 4.976448509904196e-43
Reject Null Hypothesis
```

Conclusion

The observed p-value is exceedingly minuscule, and we are opting to reject the Null Hypothesis due to the discernible differences in the rent count for various weather conditions. Specifically, the "clear" and "lightsnow" weather conditions exhibit substantial bike rentals, whereas the "weather 4" condition showcases negligible rentals. This pattern indicates a discernible impact of weather conditions on bike rentals, reinforcing the notion that these conditions are not uniformly similar.

```
#Weather is dependent on season (checking between 2 predictor
variable)
#Using chisquare_test
```

```
val = pd.crosstab(index = df["weather"], columns = df["season"])
print(val)
chisquare(val)
```

```
season      1     2     3     4
weather
1        1759  1801  1930  1702
2         715   708   604   807
3         211   224   199   225
4           1     0     0     0
```

```
Power_divergenceResult(statistic=array([2749.33581534, 2821.39590194,
3310.63995609, 2531.07388442]), pvalue=array([0., 0., 0., 0.]))
```

Using chi2_contigency test

```
#H0 = Weather is not dependent (Independent) on season.
#HA = Weather is dependent on Season
#Righ Tailed /Left Tailed/Two tailed
#Test Statistic and p_value
#We will consider alpha as 0.01 significance value. i.e 99% confidence
alpha = 0.01
val = pd.crosstab(index = df["weather"], columns = df["season"])
#print(val)
chi_stat, p_value, df, confusion_matrix = chi2_contingency(val)
print(f"Test statistic = {chi_stat} pvalue = {p_value}") #degree of
freedom (df) = {df}")
#print("The confusion matrix is :")
#print(confusion_matrix)
if (p_value < alpha):
 print("Reject Null Hypothesis")
else:
 print("Fail to reject Null Hypothesis")
```

```
Test statistic = 49.15865559689363 pvalue = 1.5499250736864862e-07
Reject Null Hypothesis
```

Conclusion:
Based on the results of the chi-squared test for independence conducted at a significance level of 0.01, we reject the null hypothesis. The obtained p-value is considerably low, indicating a strong statistical dependency between the attributes "Weather" and "Season." This implies that the weather conditions and the season are closely associated with each other in the given dataset.

Insights:

1. A 2-sample T-test conducted on working and non-working days in terms of the count variable indicates that the mean population count remains statistically similar for both categories.
2. An ANOVA test performed on distinct seasons concerning the count variable suggests that the mean population counts differ significantly across various seasons, implying a notable variation in Yulu bike usage.
3. Upon applying an ANOVA test on different weather conditions (excluding condition 4) concerning the count variable, it can be inferred that the mean population counts across diverse weather conditions remain statistically similar, implying that Yulu bike usage demonstrates consistency across various weather scenarios.
4. The results of a Chi-squared test conducted on the categorical variables of season and weather indicate a correlation between weather and season, implying a dependency between these two factors.
5. The fall and winter seasons exhibit the highest number of holidays among all the seasons.
6. A positive correlation exists between counts and temperature.
7. Conversely, a negative correlation is observed between counts and humidity.
8. The ANOVA hypothesis test supports the observation that a larger count occurs during clear weather conditions with minimal cloud cover.

Recommendations:

1. To attract more casual users, Yulu should consider implementing effective marketing strategies such as offering first-time user discounts, introducing friends and family discounts, and providing referral bonuses.
2. On non-working days when the count is considerably low, Yulu could leverage promotional activities like organizing city exploration competitions or health campaigns to boost user engagement and rentals.
3. To address the issue of low rent counts during heavy rainfall, Yulu might explore introducing alternative vehicles like covered cars or vehicles with protective features against rain.
4. During the summer and fall seasons, Yulu should ensure a substantial inventory of bikes for rent, as these periods witness a higher demand compared to other seasons.
5. Based on the analysis, with a significance level of 0.05, it can be concluded that working days have no significant impact on the number of bikes rented.
6. To optimize bike inventory management, Yulu should consider reducing bike availability on days with extremely low humidity.
7. During days with a temperature less than 10°C or in very cold weather conditions, Yulu may choose to reduce the number of available bikes for rent.
8. On days when the wind speed exceeds 35 units or during thunderstorms, Yulu should consider limiting bike availability due to safety concerns. *VRM*