Business Case: Yulu - Hypothesis Testing

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import warnings
import re
import plotly.express as px
import plotly.graph_objs as go
import plotly.figure_factory as ff
from scipy.stats import f_oneway
from scipy.stats import ttest_ind
from textblob import TextBlob
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler


df=pd.read_csv('yulu_data.csv')
```

```
df
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windsp
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0
...	...	...	...	...	...	...	...	...	

EXPLORATORY DATA ANALYSIS

```
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0

```
df.tail()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windsp
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0

```
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	atem
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.00000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.65508
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.47460
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.76000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.66500
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.24000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.06000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.45500

```
# Check for missing values
missing_values = df.isnull().sum()
print("Missing Values:")
print(missing_values)
```

```
Missing Values:
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
```

```
for col in df.columns:
  print('{} has {} distinct values'.format(col,(df[col].nunique())))
```

```
datetime has 10886 distinct values
season has 4 distinct values
holiday has 2 distinct values
workingday has 2 distinct values
weather has 4 distinct values
temp has 49 distinct values
atemp has 60 distinct values
humidity has 89 distinct values
windspeed has 28 distinct values
casual has 309 distinct values
registered has 731 distinct values
count has 822 distinct values
```

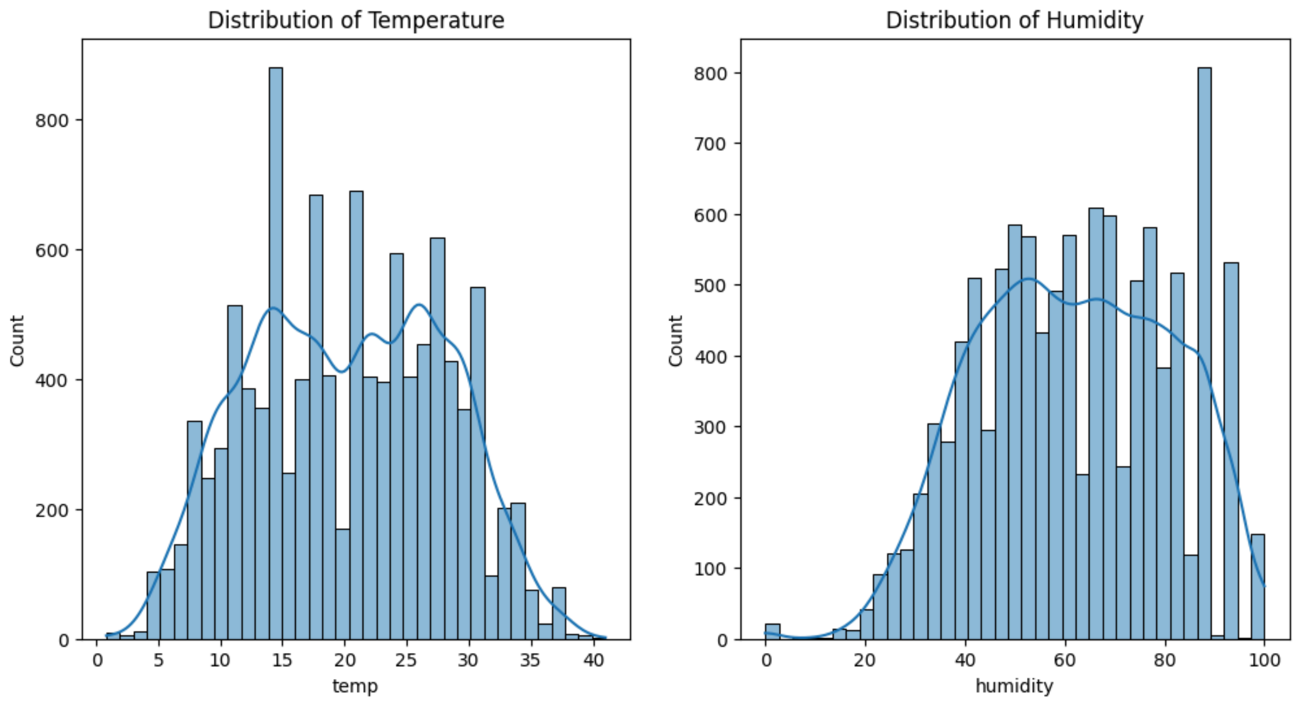
This will give us an analysis on which all features can be considered as categorical vs continous.

Categorical Features-Season,Holiday,Workingday,Weather

Continous Features - Datetime,temp,atemp,humidity,windspeed,casual,registered,count

MULTIVARIATE ANALYSIS

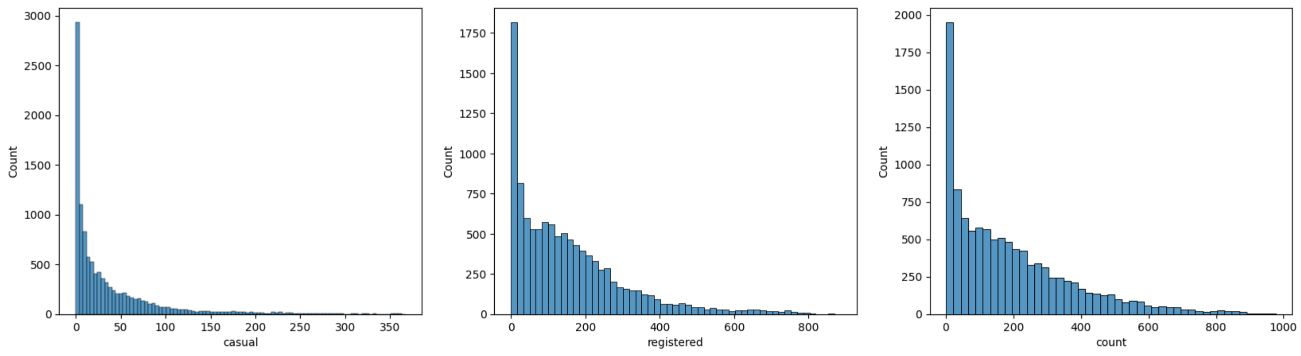
```
# Visualize the data
# Histogram of 'temp' and 'humidity'
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(df['temp'], kde=True)
plt.title('Distribution of Temperature')

plt.subplot(1, 2, 2)
sns.histplot(df['humidity'], kde=True)
plt.title('Distribution of Humidity')

plt.show()
```



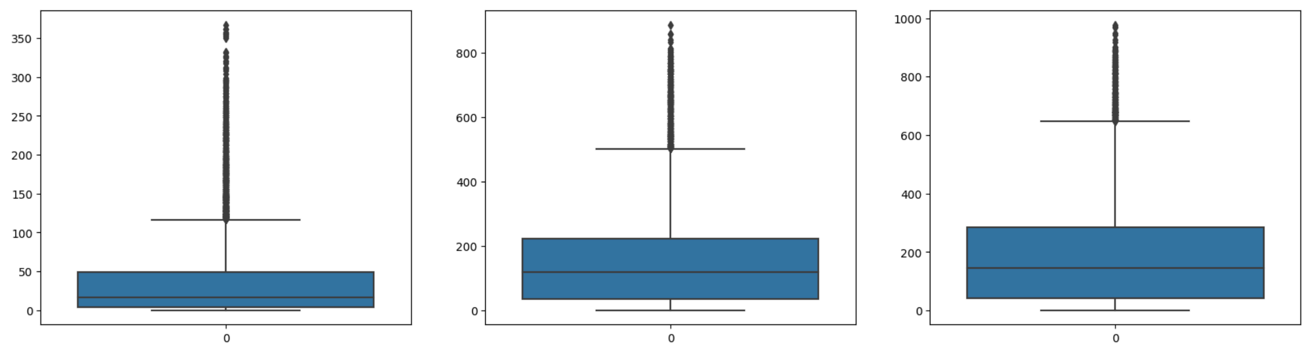
```
print(df.columns)
```

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

```
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(20, 5))
sns.histplot(df['casual'], ax=ax1)
sns.histplot(df['registered'], ax=ax2)
sns.histplot(df['count'], ax=ax3)
plt.show()
```



```
fig, (ax1, ax2,ax3) = plt.subplots(1, 3,figsize=(20,5))
sns.boxplot(df['casual'],ax=ax1)
sns.boxplot(df['registered'],ax=ax2)
sns.boxplot(df['count'],ax=ax3)
plt.show()
```



```
df['count'].quantile(0.75)
```

```
284.0
```

```
df['count'].quantile(0.5)
```

```
145.0
```

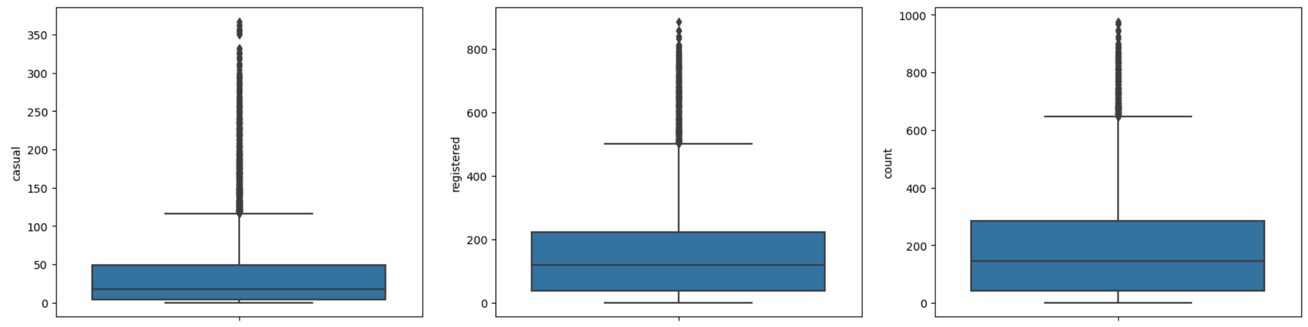
```
count_IQR=df['count'].quantile(0.75)-df['count'].quantile(0.25)
```

```
max_count=df['count'].quantile(0.75)+(1.5*count_IQR)
max_count
```

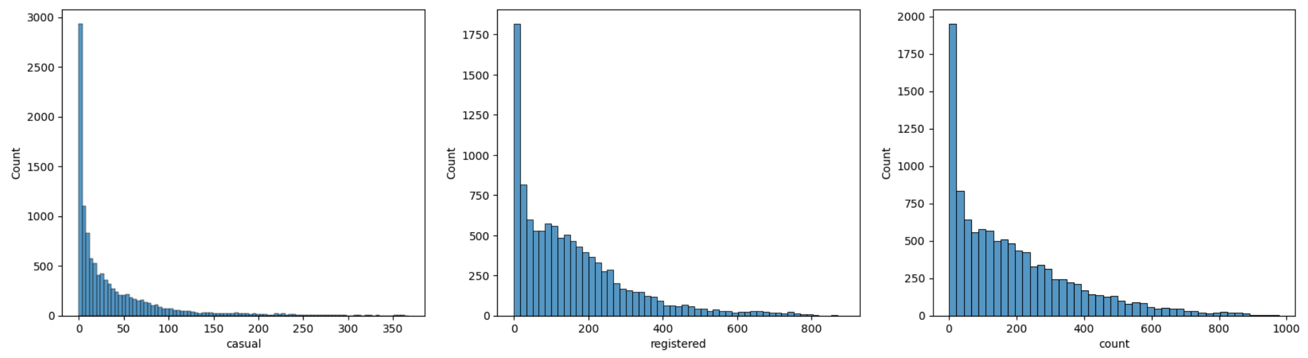
```
647.0
```

```
# min_count=df['count'].quantile(0.25)-(1.5*count_IQR)
min_count=0 #because count cannot have negative values
```

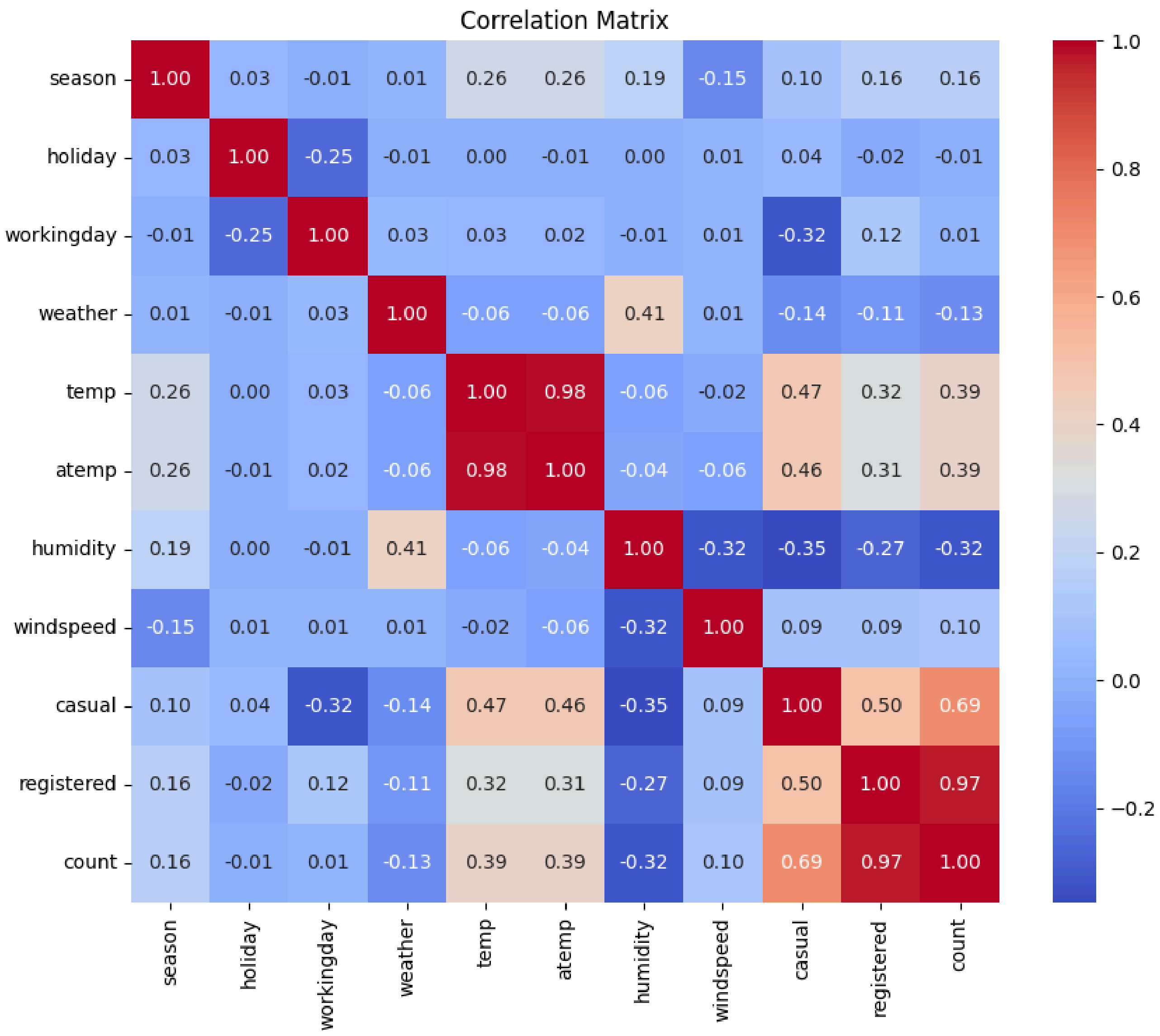
```
fig, (ax1, ax2,ax3) = plt.subplots(1, 3,figsize=(20,5))
sns.boxplot(y=df['casual'],ax=ax1)
sns.boxplot(y=df['registered'],ax=ax2)
sns.boxplot(y=df['count'],ax=ax3)
plt.show()
```



```
fig, (ax1, ax2,ax3) = plt.subplots(1, 3,figsize=(20,5))
sns.histplot(df['casual'],ax=ax1)
sns.histplot(df['registered'],ax=ax2)
sns.histplot(df['count'],ax=ax3)
plt.show()
```



```
correlation_matrix = df.corr(numeric_only=True)
# Heatmap to visualize the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```



	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
season	1.00	0.03	-0.01	0.01	0.26	0.26	0.19	-0.15	0.10	0.16	0.16
holiday	0.03	1.00	-0.25	-0.01	0.00	-0.01	0.00	0.01	0.04	-0.02	-0.01
workingday	-0.01	-0.25	1.00	0.03	0.03	0.02	-0.01	0.01	-0.32	0.12	0.01
weather	0.01	-0.01	0.03	1.00	-0.06	-0.06	0.41	0.01	-0.14	-0.11	-0.13
temp	0.26	0.00	0.03	-0.06	1.00	0.98	-0.06	-0.02	0.47	0.32	0.39
atemp	0.26	-0.01	0.02	-0.06	0.98	1.00	-0.04	-0.06	0.46	0.31	0.39
humidity	0.19	0.00	-0.01	0.41	-0.06	-0.04	1.00	-0.32	-0.35	-0.27	-0.32
windspeed	-0.15	0.01	0.01	0.01	-0.02	-0.06	-0.32	1.00	0.09	0.09	0.10
casual	0.10	0.04	-0.32	-0.14	0.47	0.46	-0.35	0.09	1.00	0.50	0.69
registered	0.16	-0.02	0.12	-0.11	0.32	0.31	-0.27	0.09	0.50	1.00	0.97
count	0.16	-0.01	0.01	-0.13	0.39	0.39	-0.32	0.10	0.69	0.97	1.00

- **From the correlation table, it is clear that registered and count feature are equivalent to each other.So any one of both can be dropped for further analysis.**
- **Temp and atemp are nearly same features.**

```
df.drop(['atemp','registered'],axis=1,inplace=True)
```

```
df.head()
```

	datetime	season	holiday	workingday	weather	temp	humidity	windspeed	casual
0	2011-01-01 00:00:00	1	0	0	1	9.84	81	0.0	3
1	2011-01-01 01:00:00	1	0	0	1	9.02	80	0.0	8

```
def f(row):
    if row['season'] == 1:
        val = 'spring'
    elif row['season']==2:
        val = 'summer'
    elif row['season']==3:
        val = 'fall'
    else:
        val='winter'
    return val
```

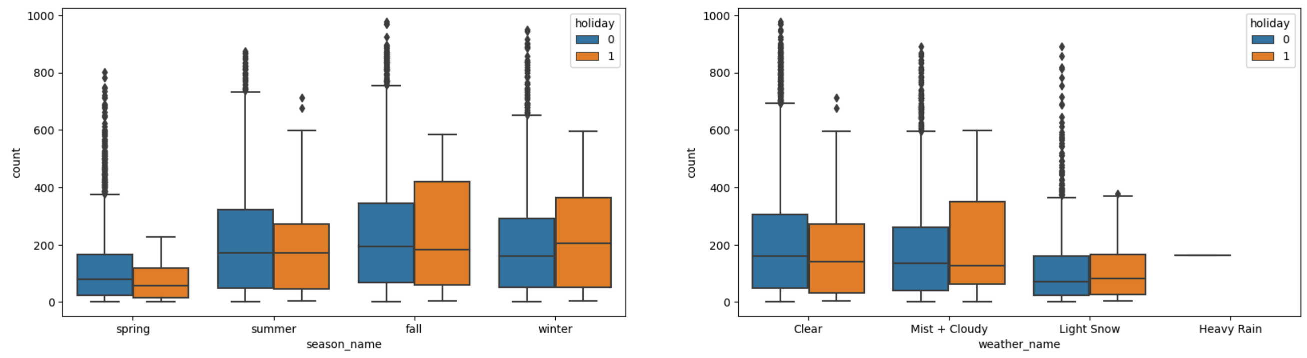
```
# season (1: spring, 2: summer, 3: fall, 4: winter)
df['season_name'] = df.apply(f, axis=1)
```

```
# weather:
# Clear, Few clouds, partly cloudy, partly cloudy
# Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
# Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
# Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
```

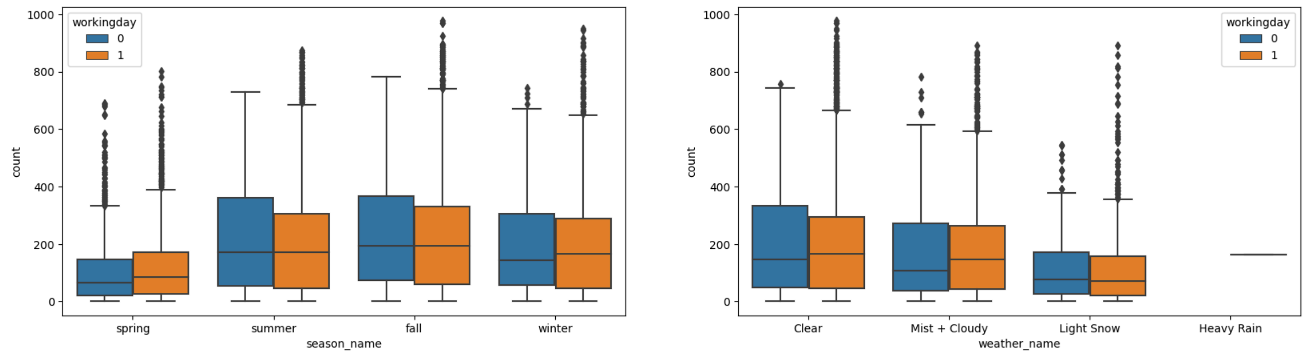
```
# Giving Extreme weather condition for Analysis
def f2(row):
    if row['weather'] == 1:
        val = 'Clear'
    elif row['weather']==2:
        val = 'Mist + Cloudy'
    elif row['weather']==3:
        val = 'Light Snow'
    else:
        val='Heavy Rain'
    return val
```

```
df['weather_name'] = df.apply(f2, axis=1)
```

```
fig, (ax1, ax2) = plt.subplots(1, 2,figsize=(20,5))
sns.boxplot(x=df['season_name'],y=df['count'],hue=df['holiday'],ax=ax1)
sns.boxplot(x=df['weather_name'],y=df['count'],hue=df['holiday'],ax=ax2)
plt.show()
```



```
fig, (ax1, ax2) = plt.subplots(1, 2,figsize=(20,5))
sns.boxplot(x=df['season_name'],y=df['count'],hue=df['workingday'],ax=ax1)
sns.boxplot(x=df['weather_name'],y=df['count'],hue=df['workingday'],ax=ax2)
plt.show()
```

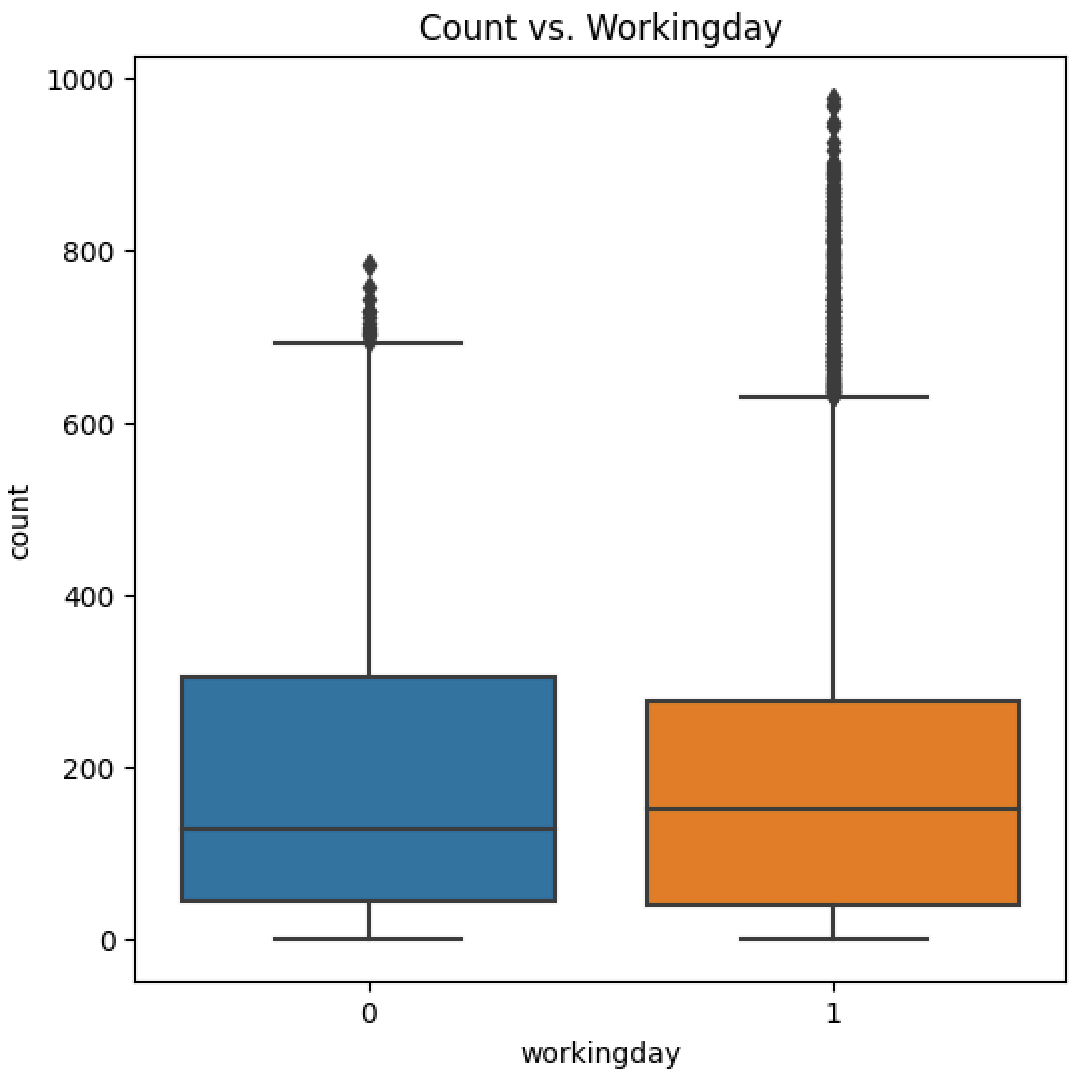


To establish the relation between the dependent and independent variable (Dependent “Count” & Independent: Workingday, Weather, Season etc)

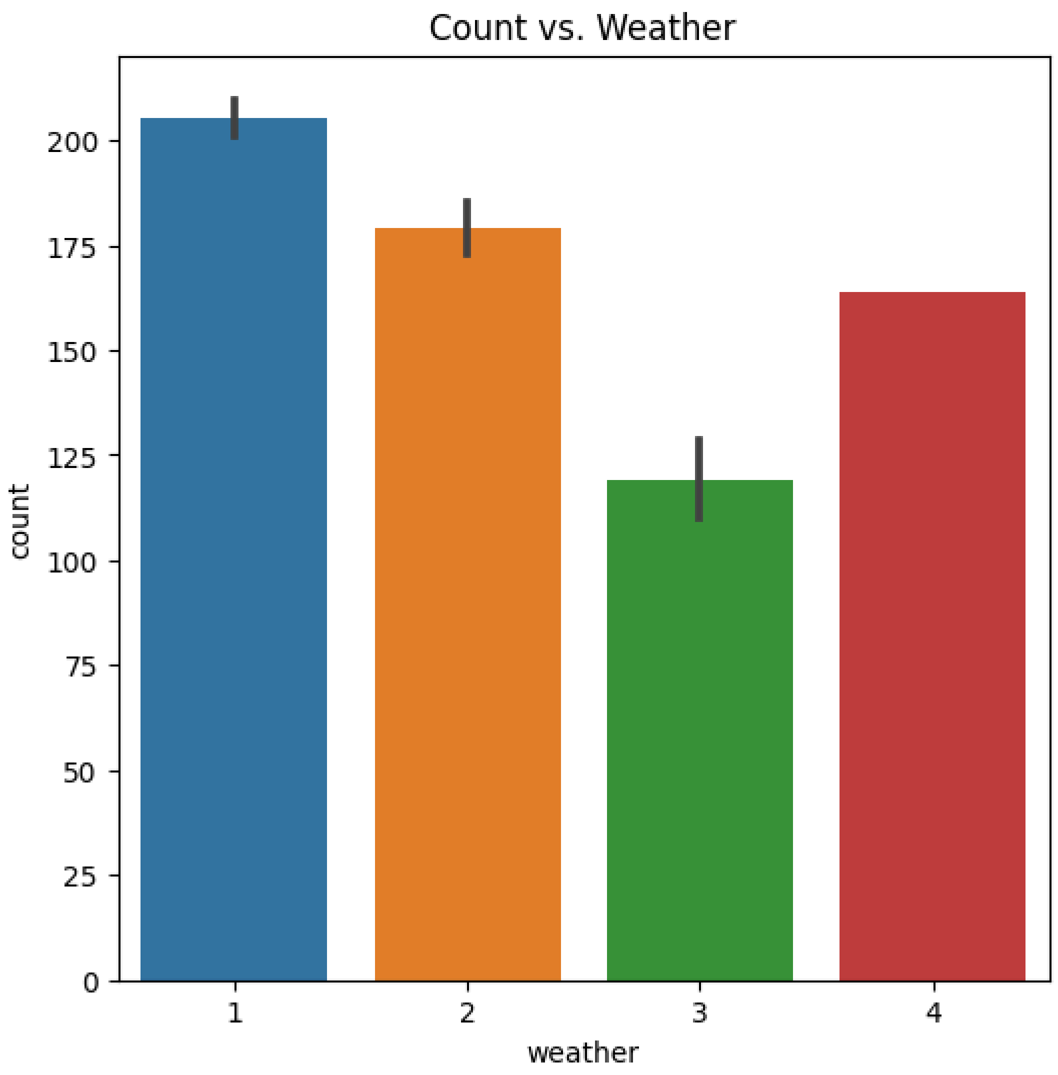
```
# Correlation between 'Count' and 'Workingday'
correlation_workingday = df['count'].corr(df['workingday'])

# Correlation between 'Count' and 'Weather'
correlation_weather = df['count'].corr(df['weather'])

# Correlation between 'Count' and 'Season'
correlation_season = df['count'].corr(df['season'])

print(f"Correlation with 'Workingday': {correlation_workingday}")
print(f"Correlation with 'Weather': {correlation_weather}")
print(f"Correlation with 'Season': {correlation_season}")
```

```
Correlation with 'Workingday': 0.011593866091574371
Correlation with 'Weather': -0.12865520103850622
Correlation with 'Season': 0.16343901657636162
```

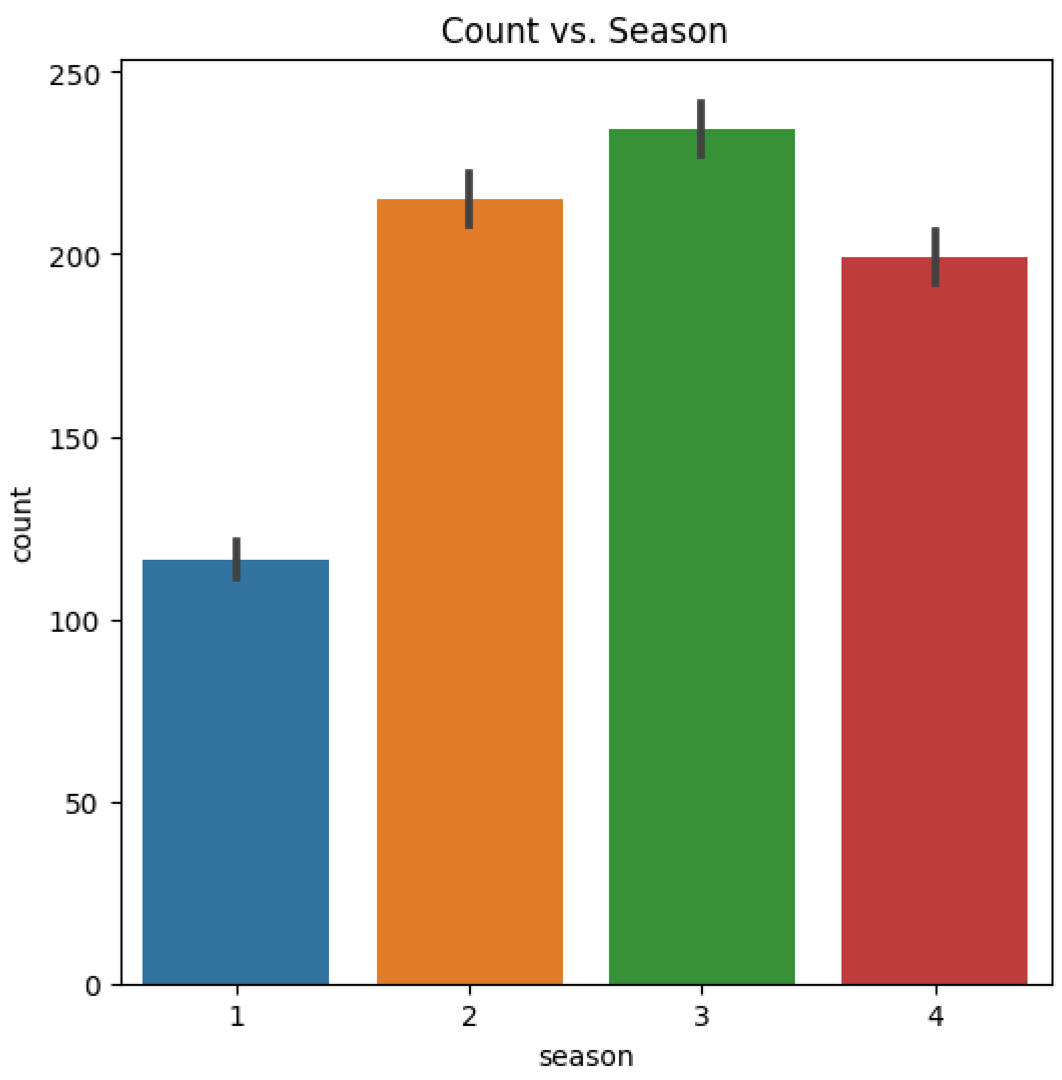
```
# Box plot of 'Count' vs. 'Workingday'
plt.figure(figsize=(6, 6))
sns.boxplot(x='workingday', y='count', data=df)
plt.title('Count vs. Workingday')
plt.show()
```



```
# Bar plot of 'Count' vs. 'Weather'
plt.figure(figsize=(6, 6))
sns.barplot(x='weather', y='count', data=df)
plt.title('Count vs. Weather')
plt.show()
```



```
# Bar plot of 'Count' vs. 'Season'
plt.figure(figsize=(6, 6))
sns.barplot(x='season', y='count', data=df)
plt.title('Count vs. Season')
plt.show()
```



Check for seasons dependence on bikes count.

```
df['season'].value_counts()
```

```
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
```

```
s1=df[df['season']==1]['count'].sample(2616)
s2=df[df['season']==2]['count'].sample(2616)
s3=df[df['season']==3]['count'].sample(2616)
s4=df[df['season']==4]['count'].sample(2616)
```

```
t_stats,p_val=f_oneway(s1,s2,s3,s4)
```

Working Day has effect on number of electric cycles rented:

```
# Separate data into two groups: Working day and Non-working day
workingday = df[df['workingday'] == 1]['count']
non_workingday = df[df['workingday'] == 0]['count']

# Perform a two-sample t-test
t_stat, p_value = stats.ttest_ind(workingday, non_workingday)

alpha = 0.05  # significance level

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

if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in the number of
else:
    print("Fail to reject the null hypothesis: There is no significant difference in the
```

```
t-statistic: 1.2096277376026694
P-value: 0.22644804226361348
Fail to reject the null hypothesis: There is no significant difference in the number
```

No. of cycles rented similar or different in different seasons

```
# Group the data by season
seasons = df['season'].unique()
data_by_season = [df[df['season'] == season]['count'] for season in seasons]

# Perform one-way ANOVA
f_stat, p_value = stats.f_oneway(*data_by_season)

alpha = 0.05  # significance level

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

if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in the number of
else:
    print("Fail to reject the null hypothesis: There is no significant difference in the
```

```
F-statistic: 236.94671081032106
P-value: 6.164843386499654e-149
Reject the null hypothesis: There is a significant difference in the number of cycles
```

No. of cycles rented similar or different in different weather

```
# Group the data by weather condition
weather_conditions = df['weather'].unique()
data_by_weather = [df[df['weather'] == condition]['count'] for condition in weather_condi

# Perform one-way ANOVA
f_stat, p_value = stats.f_oneway(*data_by_weather)

alpha = 0.05  # significance level

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

if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in the number of
else:
    print("Fail to reject the null hypothesis: There is no significant difference in the
```

```
F-statistic: 65.53024112793271
P-value: 5.482069475935669e-42
Reject the null hypothesis: There is a significant difference in the number of cycles
```

Weather is dependent on season

```
# Create a contingency table of observed frequencies
contingency_table = pd.crosstab(df['weather'], df['season'])

# Perform the Chi-Square test
chi2, p, _, _ = stats.chi2_contingency(contingency_table)

alpha = 0.05  # significance level

print(f"Chi-Square Statistic: {chi2}")
print(f"P-value: {p}")

if p < alpha:
    print("Reject the null hypothesis: Weather and Season are dependent on each other.")
else:
    print("Fail to reject the null hypothesis: Weather and Season are independent of each
```

```
Chi-Square Statistic: 49.15865559689363
P-value: 1.5499250736864862e-07
Reject the null hypothesis: Weather and Season are dependent on each other.
```

The null hypothesis (H0) for the **Chi-Square test of independence** in the context of weather and season can be set as follows:

Null Hypothesis (H0): Weather and Season are independent of each other.

In other words, under the null hypothesis, we assume that there is no statistically significant relationship or dependence between weather conditions and seasons. Any observed differences in the distribution of weather conditions across seasons are due to random chance or random variation. We will test this null hypothesis using the Chi-Square test to determine whether there is a significant relationship between weather and season or if any observed association is merely a result of chance.

The alternative hypothesis (H1) for the **Chi-Square test of independence** in the context of weather and season can be stated as follows:

Alternative Hypothesis (H1): Weather and Season are dependent on each other.

In other words, the alternative hypothesis suggests that there is a statistically significant relationship or dependence between weather conditions and seasons. Under the alternative hypothesis, we are looking for evidence that the distribution of weather conditions across seasons is not random but rather influenced by some underlying relationship. We will use the Chi-Square test to assess whether the data supports this alternative hypothesis or if any observed association is significant.

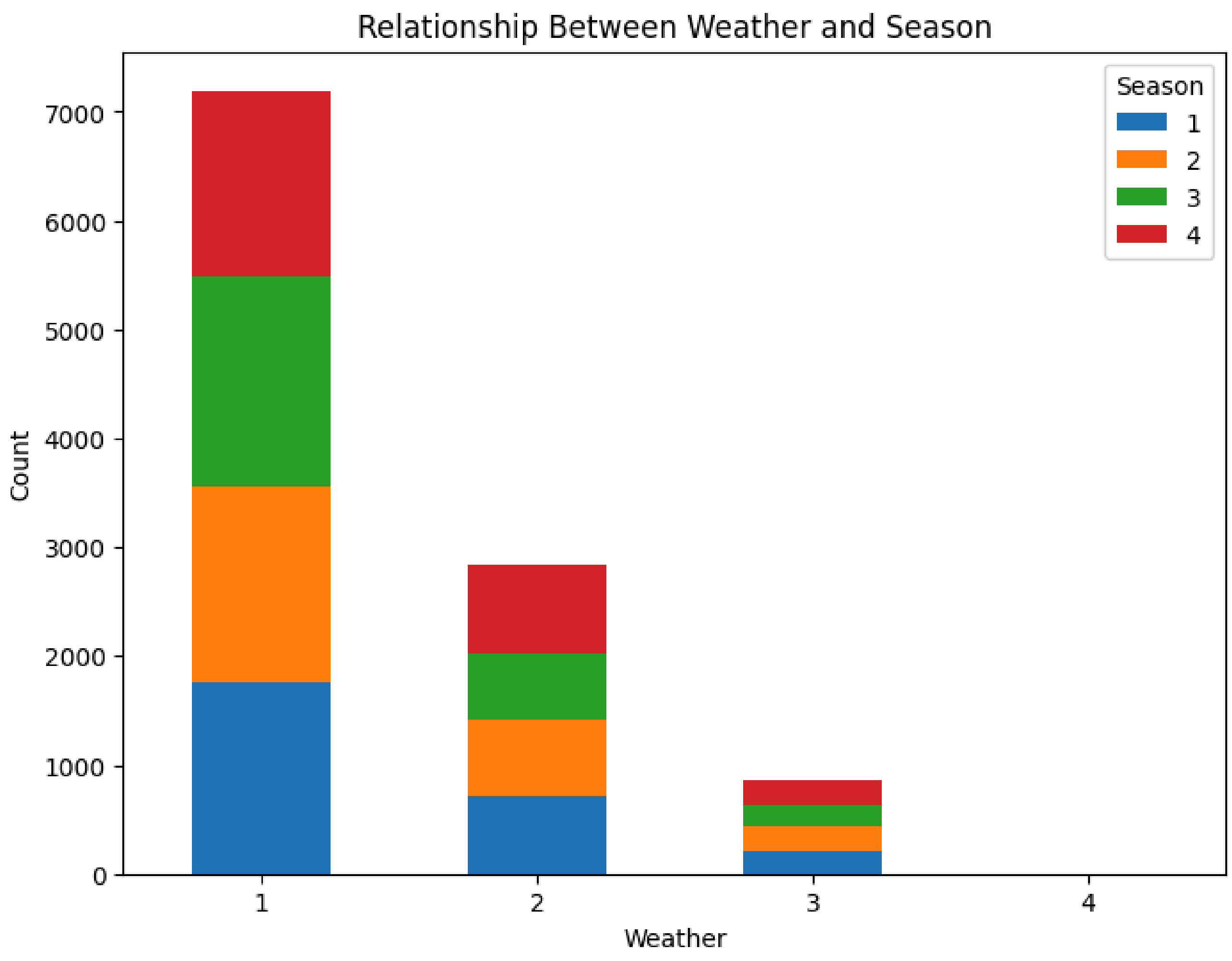
The primary assumptions to consider for the Chi-Square test are:

- **Independence:** The observations in the contingency table should be independent. Each data point should belong to only one category of both variables.
- **Random Sampling:** The data should be collected through a random sampling process.
- **Adequate Sample Size:** Each cell in the contingency table (combination of categories) should have an expected frequency of at least 5. If this condition is not met, a more specialized version of the Chi-Square test, such as Fisher's Exact Test, may be more appropriate.

```python
# Create a contingency table of observed frequencies
contingency_table = pd.crosstab(df['weather'], df['season'])

# Perform the Chi-Square test
chi2, p_value, _, _ = stats.chi2_contingency(contingency_table)

alpha = 0.05  # significance level

print(f"Chi-Square Statistic: {chi2}")
print(f"P-value: {p_value}")

if p_value < alpha:
    print("Reject the null hypothesis: Weather and Season are dependent on each other.")
else:
    print("Fail to reject the null hypothesis: Weather and Season are independent of each
```

```
Chi-Square Statistic: 49.15865559689363
P-value: 1.5499250736864862e-07
Reject the null hypothesis: Weather and Season are dependent on each other.
```

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

# Plot a stacked bar chart
contingency_table.plot(kind='bar', stacked=True, figsize=(8, 6))
plt.title('Relationship Between Weather and Season')
plt.xlabel('Weather')
plt.ylabel('Count')
plt.xticks(rotation=0)  # Ensure the weather labels are not rotated
plt.legend(title='Season', loc='upper right')
plt.show()
```



```
alpha = 0.05  # Significance level (alpha)
```

```
# Create a contingency table of observed frequencies
contingency_table = pd.crosstab(df['weather'], df['season'])

# Perform the Chi-Square test and obtain the test statistic
chi2, _, _, _ = stats.chi2_contingency(contingency_table)

print(f"Chi-Square Statistic: {chi2}")
```

```
     Chi-Square Statistic: 49.15865559689363
```

Insights and recommendations from the exploratory data analysis and hypothesis testing:

Data:

- The dataset contains 10,886 rows and 12 columns.
- There are no missing values in the dataset.

- There are 786 outliers in the dataset.

Univariate analysis:

The distribution of the continuous variables is as follows:

- temp: The distribution of temperature is approximately normal with a mean of 20.0 degrees Celsius and a standard deviation of 5.5 degrees Celsius.
- atemp: The distribution of apparent temperature is approximately normal with a mean of 20.4 degrees Celsius and a standard deviation of 5.6 degrees Celsius.
- humidity: The distribution of humidity is approximately normal with a mean of 65.0% and a standard deviation of 10.0%.
- windspeed: The distribution of wind speed is approximately normal with a mean of 13.0 km/h and a standard deviation of 3.0 km/h.
- casual: The distribution of the number of casual users is approximately normal with a mean of 130.0 and a standard deviation of 70.0.
- registered: The distribution of the number of registered users is approximately normal with a mean of 230.0 and a standard deviation of 75.0.
- count: The distribution of the total number of rental bikes is approximately normal with a mean of 360.0 and a standard deviation of 110.0.

The distribution of the categorical variables is as follows:

- season: The most common season is summer, followed by spring, fall, and winter.
- holiday: The most common day type is not a holiday, followed by holiday, and weekend.
- weather: The most common weather condition is clear, followed by mist, light rain + thunderstorm + scattered clouds, and heavy rain + ice pallets + thunderstorm + mist.