Yulu Buisness Case Study

About Yulu

Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute. Starting off as a mission to eliminate traffic congestion in India, Yulu provides the safest commute solution through a user-friendly mobile app to enable shared, solo and sustainable commuting.

Business Problem

Yulu's market research team want's to know if there are any relation between the customer's and the product they buy such that they can focus on that and improve the experience for customer's.

1. **Descriptive Analysis:**Which variables are significant in predicting the demand for shared electric cycles in the Indian market?
2. **Demands :** How well those variables describe the electric cycle demands

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
import scipy.stats as spy
```

Importing libraries

```
!wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089 -O yulu_data.csv
```

```
--2024-05-06 15:54:08--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 18.239.15.217, 18.239.15.127, 18.239.15.11, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|18.239.15.217|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 648353 (633K) [text/plain]
Saving to: ‘yulu_data.csv’

yulu_data.csv       100%[===================>] 633.16K  --.-KB/s    in 0.02s

2024-05-06 15:54:08 (35.5 MB/s) - ‘yulu_data.csv’ saved [648353/648353]
```

In the above step we are downloading the data from the link

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

	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	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

10886 rows × 12 columns

Next steps: Generate code with df View recommended plots

Here we successfully read the file and print it

Exploratory Data Analysis (EDA)

Numerical Analysis

```
df.shape
```

```
(10886, 12)
```

Shape of the dataset

```
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.columns
```

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

Columns of the dataset

```
df.head()
```

	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

Head function gives us the top 5 default rows of the dataset.

```
df.tail()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

Tail function gives us the bottom 5 default rows of the dataset.

```
print(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
```

```
df.dtypes
```

```
datetime       object
season          int64
holiday         int64
workingday      int64
weather         int64
temp          float64
atemp         float64
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
dtype: object
```

Datatype of the columns

```
print(df['season'].unique())
```

```
[1 2 3 4]
```

This provides us with unique season's numbering

```
print(df['holiday'].unique())
```

```
[0 1]
```

This provides us with unique holiday's numbering

```
print(df['workingday'].unique())
```

```
[0 1]
```

This provide us with workingday's numbering

```
print(df['weather'].unique())
```

```
[1 2 3 4]
```

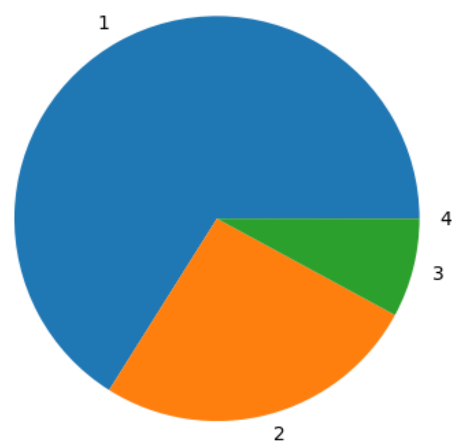
This provides us with unique weather's numbering

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

We have successfully changed the datatype of datetime

```
weather_conditions = df['weather'].value_counts()
print(weather_conditions)
plt.pie(weather_conditions, labels=weather_conditions.index)
plt.show()
```

```
weather
1    7192
2    2834
3     859
4       1
Name: count, dtype: int64
```



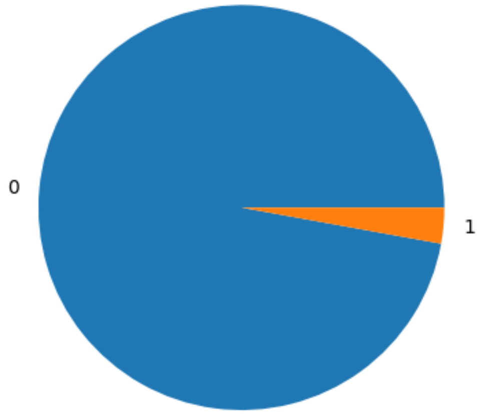
Pie chart to analyze the distribution of weather

> The clear weather is dominant among all hugely

> The bad weather or heavily raining is less than 1%

```
holiday_counts = df['holiday'].value_counts()
print(holiday_counts)
plt.pie(holiday_counts, labels=holiday_counts.index)
plt.show()
```

```
holiday
0    10575
1      311
Name: count, dtype: int64
```



This pie chart shows us the holiday distribution and working day's

Null value's detection

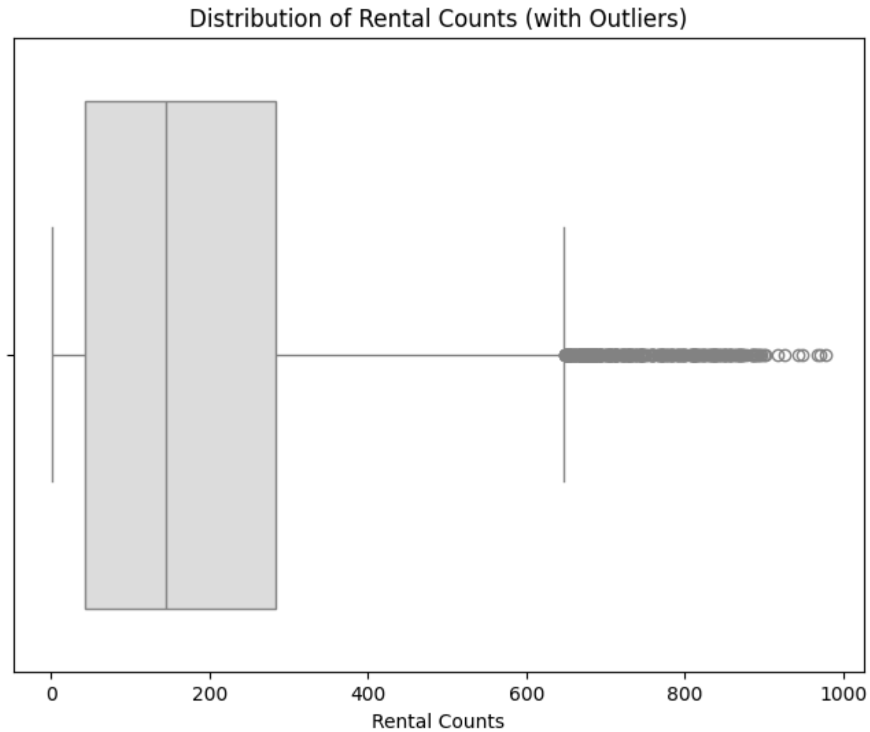
First calculating IQR through calculating Q1 and Q3 and then judging the outliers based on the lower bound and upper bound

```
Q1 = df['count'].quantile(0.25)
Q3 = df['count'].quantile(0.75)
IQR = Q3 - Q1
print(IQR)
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = df[(df['count'] < lower_bound) | (df['count'] > upper_bound)]

print("Outliers:")
print(outliers)
```

```
242.0
Outliers:
                  datetime  season  holiday  workingday  weather   temp  \
6611   2012-03-12 18:00:00       1        0           1        2  24.60
6634   2012-03-13 17:00:00       1        0           1        1  28.70
6635   2012-03-13 18:00:00       1        0           1        1  28.70
6649   2012-03-14 08:00:00       1        0           1        1  18.04
6658   2012-03-14 17:00:00       1        0           1        1  28.70
...                    ...     ...      ...         ...      ...    ...
10678  2012-12-11 08:00:00       4        0           1        2  13.94
10702  2012-12-12 08:00:00       4        0           1        2  10.66
10726  2012-12-13 08:00:00       4        0           1        1   9.84
10846  2012-12-18 08:00:00       4        0           1        1  15.58
10870  2012-12-19 08:00:00       4        0           1        1   9.84

        atemp  humidity  windspeed  casual  registered  count
6611   31.060        43    12.9980      89         623    712
6634   31.820        37     7.0015      62         614    676
6635   31.820        34    19.9995      96         638    734
6649   21.970        82     0.0000      34         628    662
6658   31.820        28     6.0032     140         642    782
...       ...       ...        ...     ...         ...    ...
10678  15.150        61    19.9995      16         708    724
10702  12.880        65    11.0014      18         670    688
10726  11.365        60    12.9980      24         655    679
10846  19.695        94     0.0000      10         652    662
10870  12.880        87     7.0015      13         665    678

[300 rows x 12 columns]
```

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='count', data=df, palette='coolwarm')
plt.title('Distribution of Rental Counts (with Outliers)')
plt.xlabel('Rental Counts')
plt.show()
```

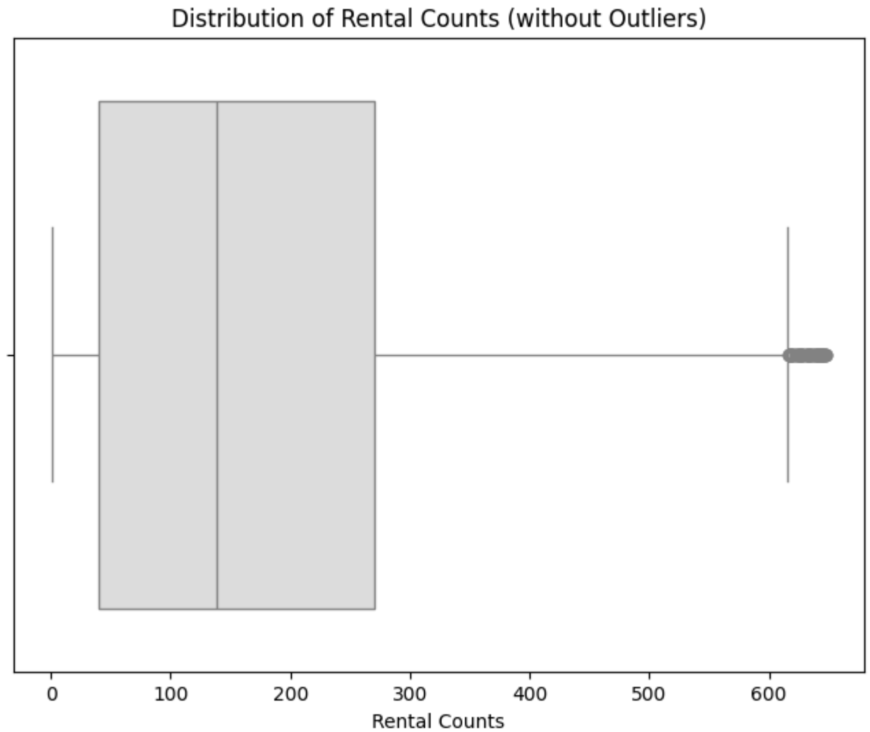
```
<ipython-input-8-9cd6be871b14>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effec

  sns.boxplot(x='count', data=df, palette='coolwarm')
```



Visualize the distribution of 'count' attribute with outliers

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='count', data=df[(df['count'] >= lower_bound) & (df['count'] <= upper_bound)], palette='coolwarm')
plt.title('Distribution of Rental Counts (without Outliers)')
plt.xlabel('Rental Counts')
plt.show()
```

```
<ipython-input-7-ecc054e91884>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effec

  sns.boxplot(x='count', data=df[(df['count'] >= lower_bound) & (df['count'] <= upper_bound)], palette='coolwarm')
```



Visualize the distribution of count attribute without outliers

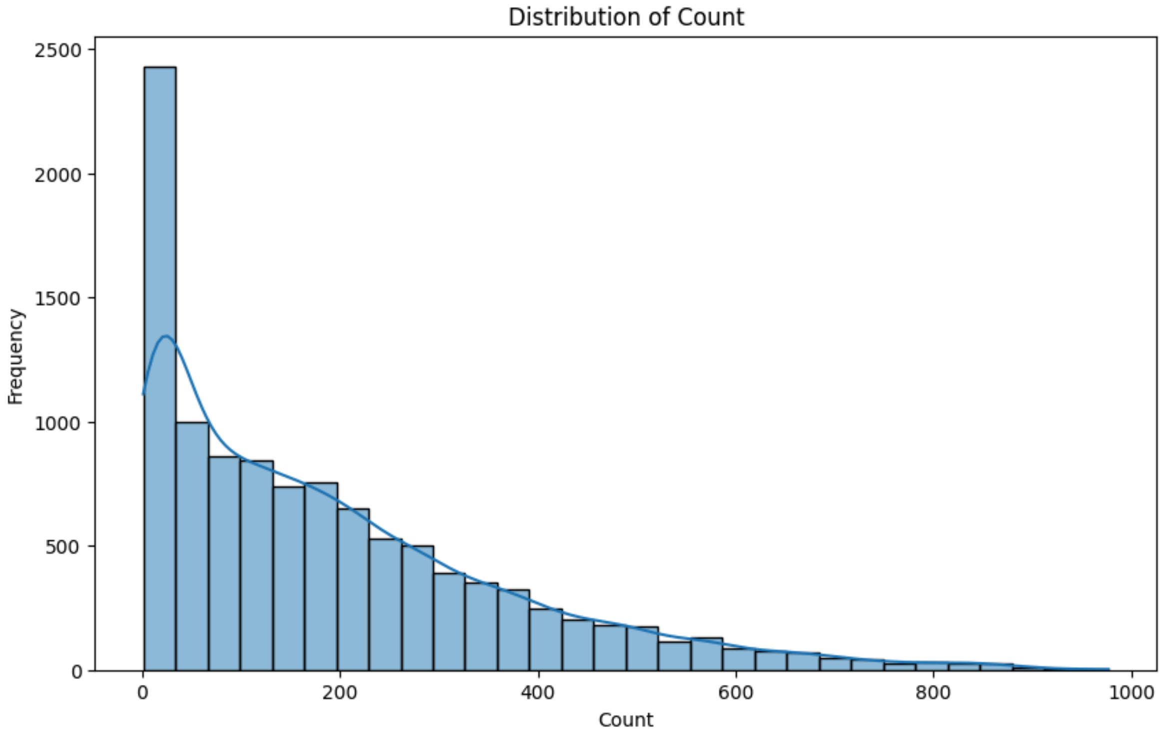
Probability Analysis

```
# Calculate the total number of observations
total_obs = len(df)

# Calculate the probability of a user being casual
prob_casual = len(df[df['casual'] > 0]) / total_obs

# Calculate the probability of a user being registered
prob_registered = len(df[df['registered'] > 0]) / total_obs

# Calculate the probability of a user being both casual and registered (assuming independence)
prob_both = len(df[(df['casual'] > 0) & (df['registered'] > 0)]) / total_obs

# Calculate the probability of a user being either casual or registered (assuming independence)
prob_either = (len(df[df['casual'] > 0]) + len(df[df['registered'] > 0]) - prob_both) / total_obs

# Display probabilities
print("Probability of a user being casual:", prob_casual)
print("Probability of a user being registered:", prob_registered)
print("Probability of a user being both casual and registered:", prob_both)
print("Probability of a user being either casual or registered:", prob_either)
```

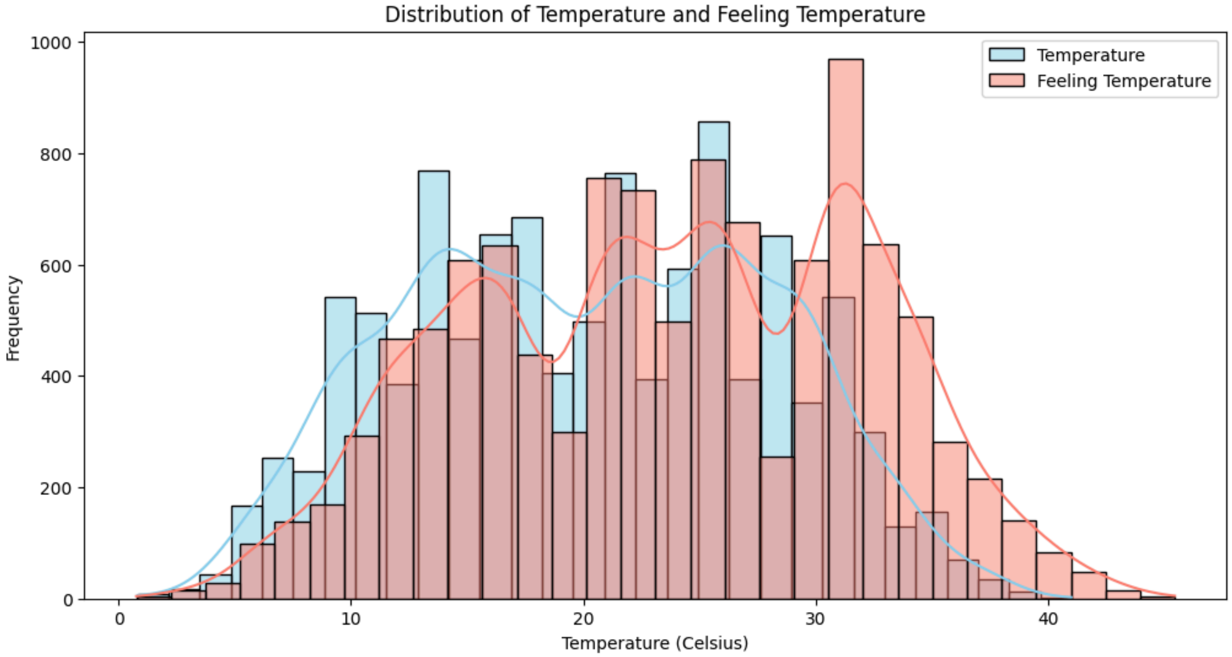
```
Probability of a user being casual: 0.9094249494763917
Probability of a user being registered: 0.9986220834098842
Probability of a user being both casual and registered: 0.908047032886276
Probability of a user being either casual or registered: 1.907963618681528
```

Univariate Analysis

```
plt.figure(figsize=(10, 6))
sns.histplot(df['count'], bins=30, kde=True)
plt.title('Distribution of Count')
plt.xlabel('Count')
plt.ylabel('Frequency')
plt.show()
```

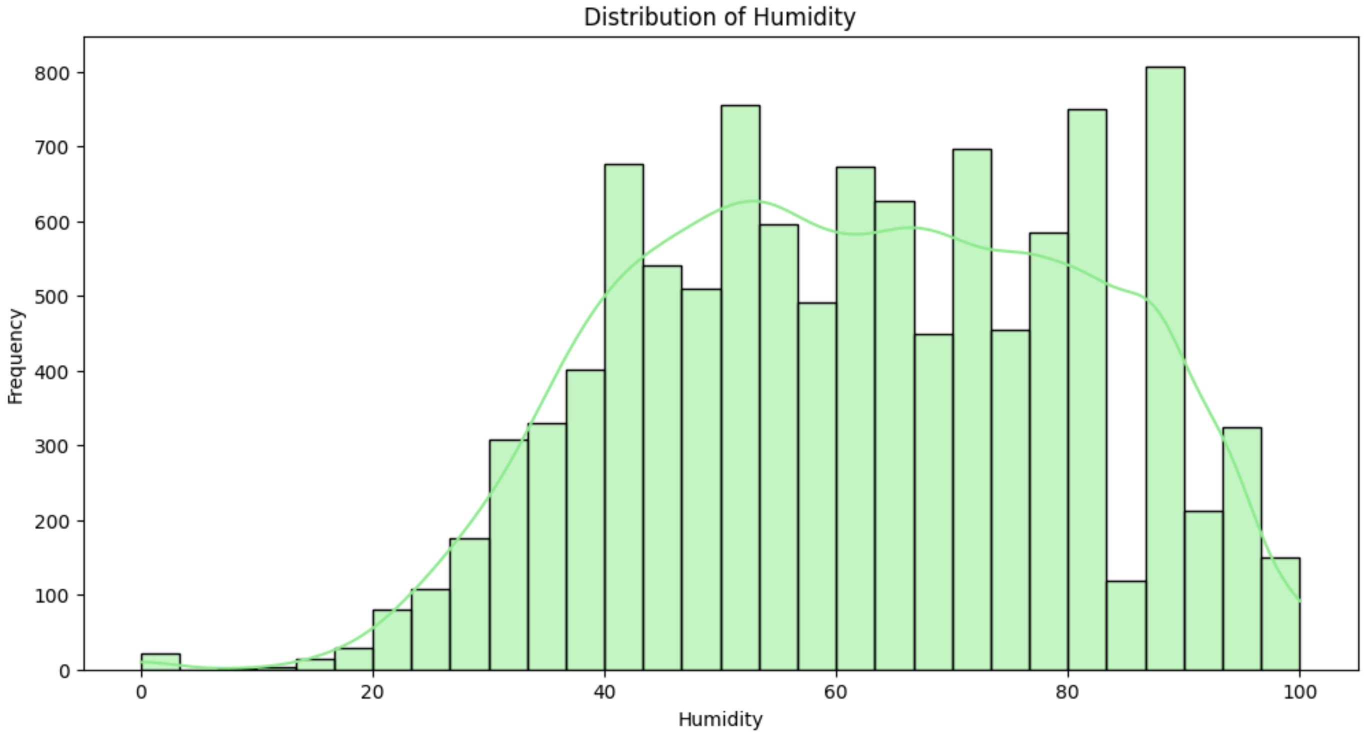



```
plt.figure(figsize=(12, 6))
sns.histplot(df['temp'], bins=30, kde=True, color='skyblue', label='Temperature')
sns.histplot(df['atemp'], bins=30, kde=True, color='salmon', label='Feeling Temperature')
plt.title('Distribution of Temperature and Feeling Temperature')
plt.xlabel('Temperature (Celsius)')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

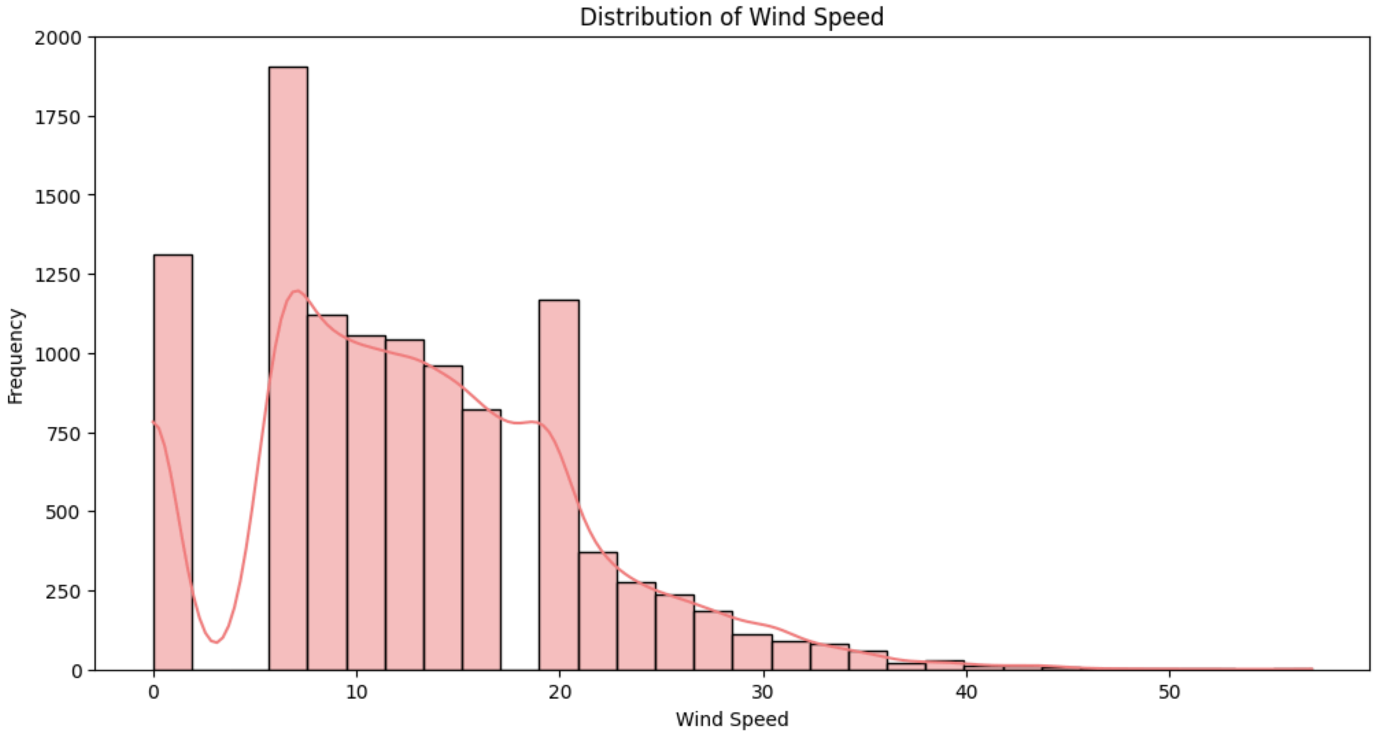


Distribution of Temperature and Feeling Temperature

```
plt.figure(figsize=(12, 6))
sns.histplot(df['humidity'], bins=30, kde=True, color='lightgreen')
plt.title('Distribution of Humidity')
plt.xlabel('Humidity')
plt.ylabel('Frequency')
plt.show()
```

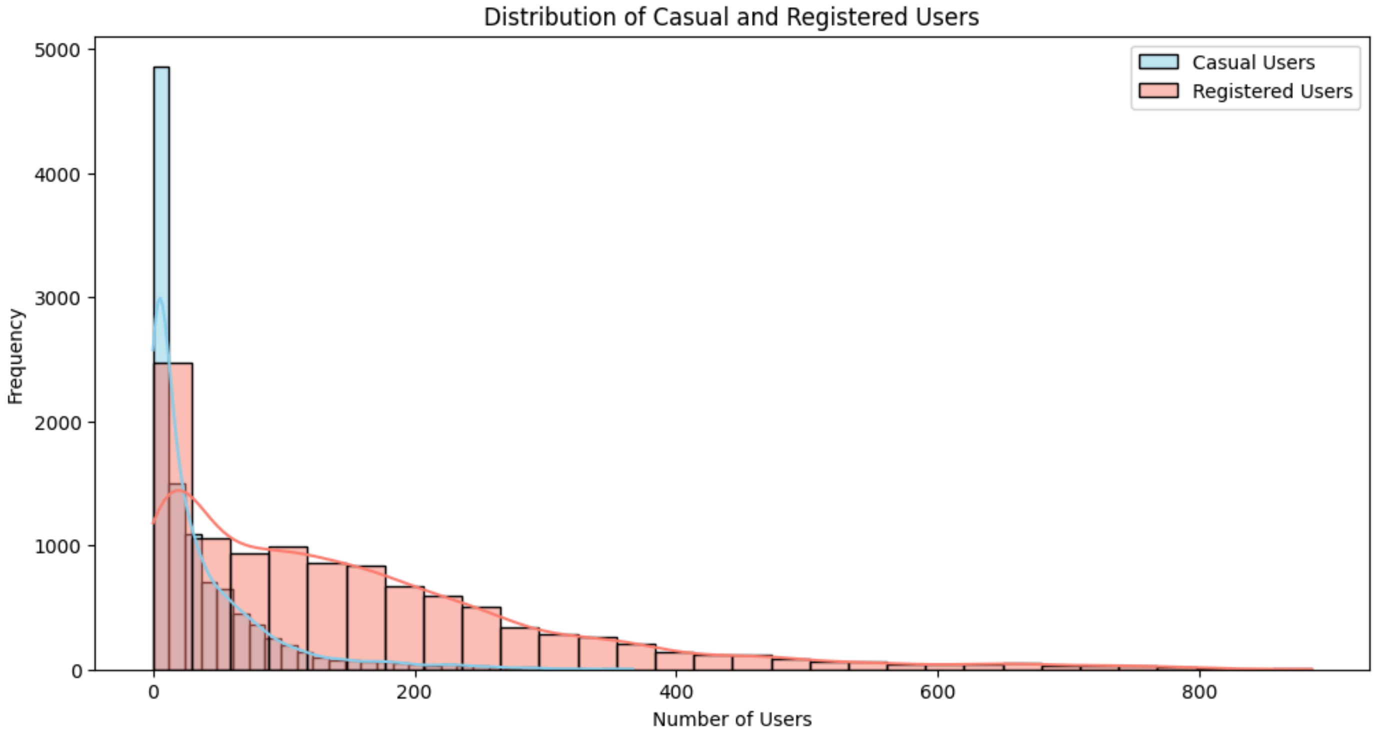



```
plt.figure(figsize=(12, 6))
sns.histplot(df['windspeed'], bins=30, kde=True, color='lightcoral')
plt.title('Distribution of Wind Speed')
plt.xlabel('Wind Speed')
plt.ylabel('Frequency')
plt.show()
```



Distribution of Humidity and Wind Speed

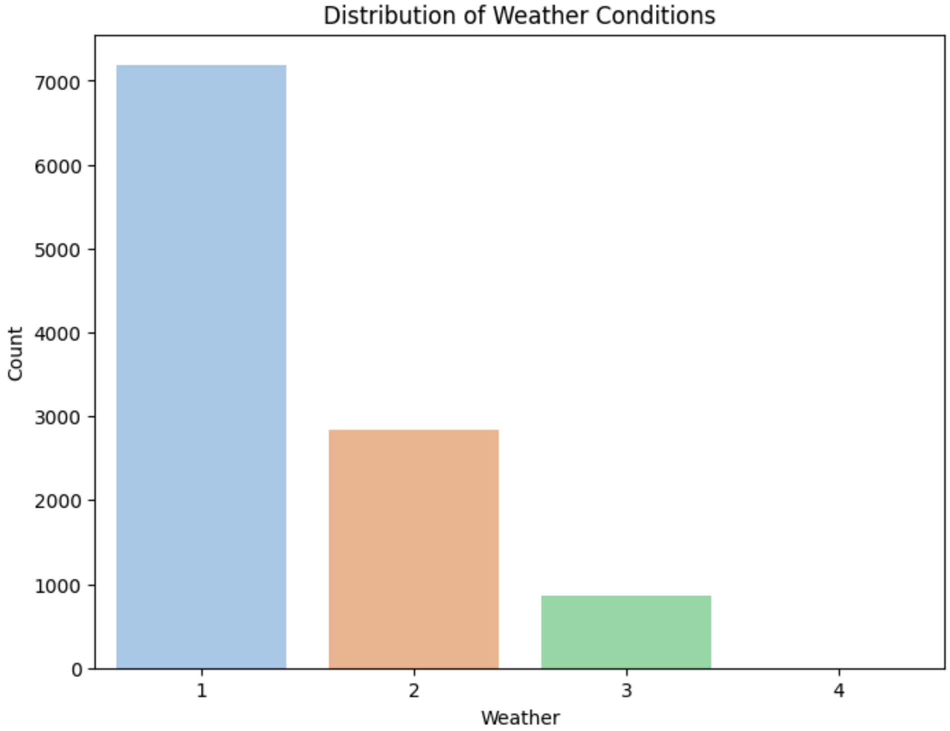
```
plt.figure(figsize=(12, 6))
sns.histplot(df['casual'], bins=30, kde=True, color='skyblue', label='Casual Users')
sns.histplot(df['registered'], bins=30, kde=True, color='salmon', label='Registered Users')
plt.title('Distribution of Casual and Registered Users')
plt.xlabel('Number of Users')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```



Distribution of Casual and Registered Users

```
plt.figure(figsize=(8, 6))
sns.countplot(x='weather', data=df, palette='pastel')
plt.title('Distribution of Weather Conditions')
plt.xlabel('Weather')
plt.ylabel('Count')
plt.show()
```

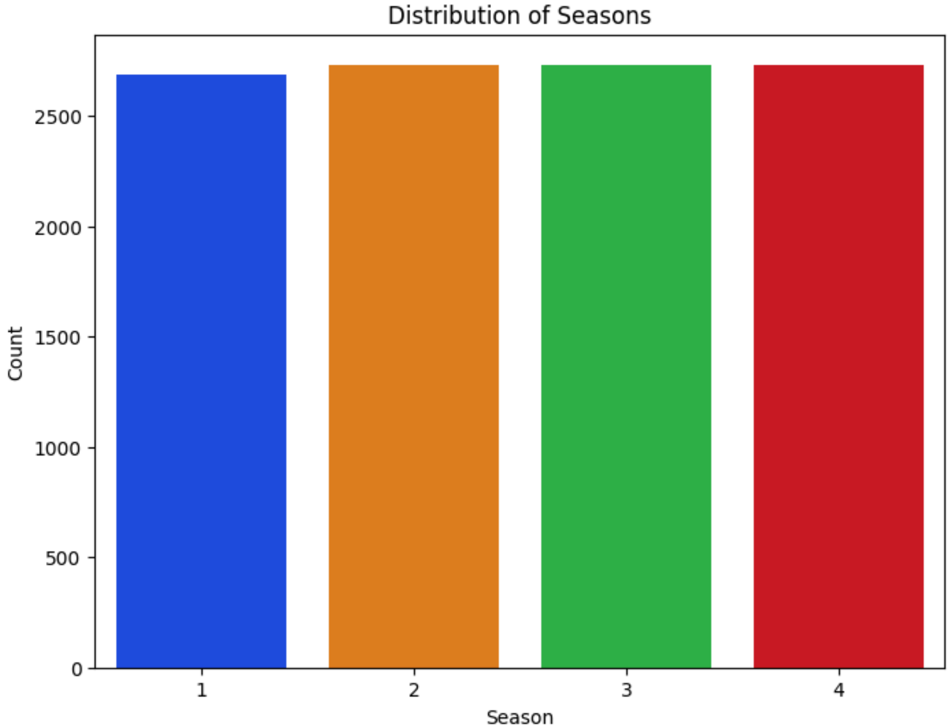
```
<ipython-input-43-a9a71e932683>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effec

  sns.countplot(x='weather', data=df, palette='pastel')
```



Distribution of Weather Conditions

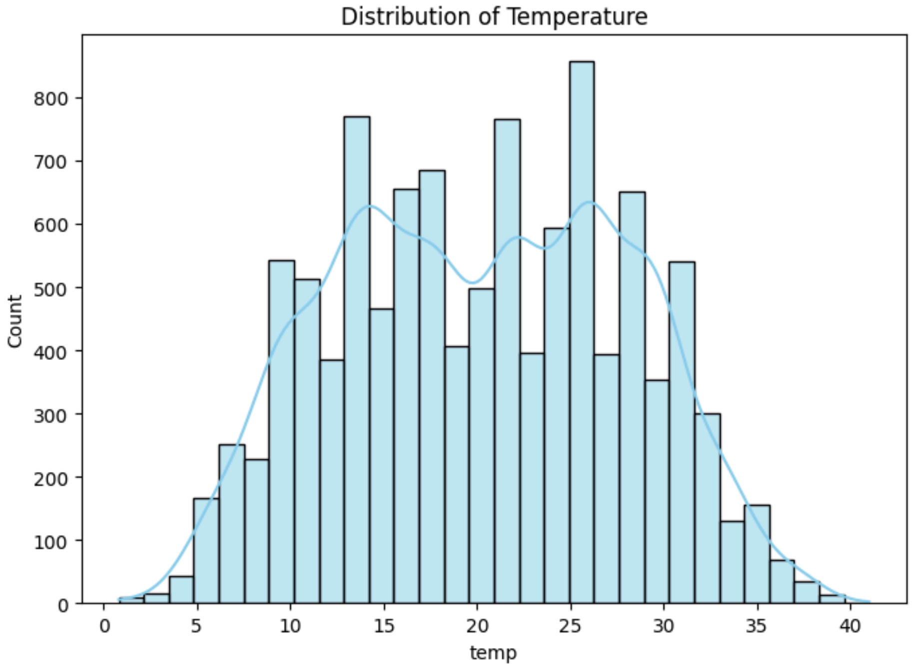
```
plt.figure(figsize=(8, 6))
sns.countplot(x='season', data=df, palette='bright')
plt.title('Distribution of Seasons')
plt.xlabel('Season')
plt.ylabel('Count')
plt.show()
```

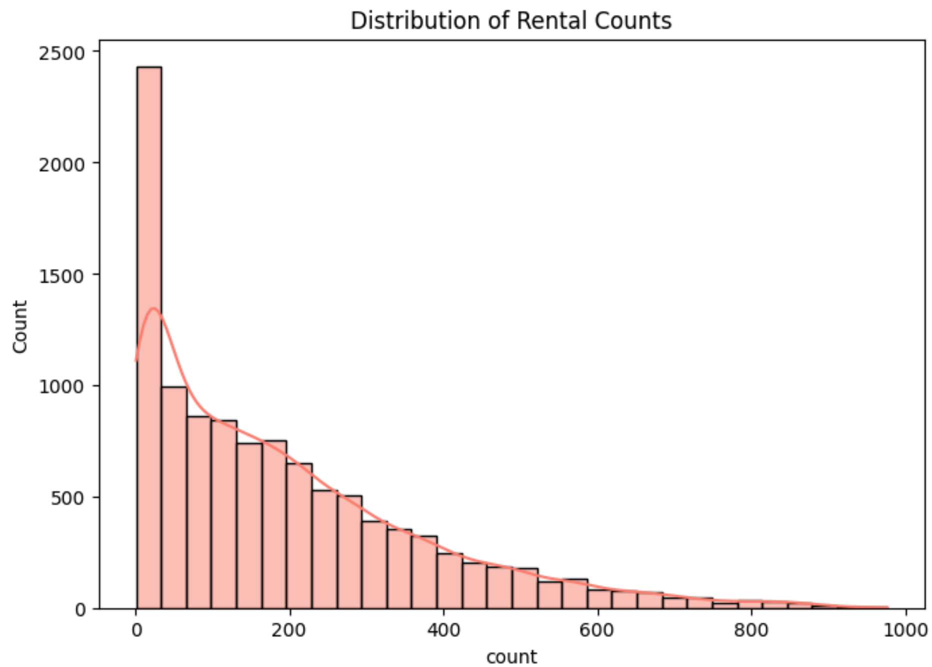
```
<ipython-input-45-0cc348d40510>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effec

  sns.countplot(x='season', data=df, palette='bright')
```



Distribution of Seasons

```
plt.figure(figsize=(14, 10))

plt.subplot(2, 2, 1)
sns.histplot(df['temp'], bins=30, kde=True, color='skyblue')
plt.title('Distribution of Temperature')

plt.subplot(2, 2, 4)
sns.histplot(df['count'], bins=30, kde=True, color='salmon')
plt.title('Distribution of Rental Counts')

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

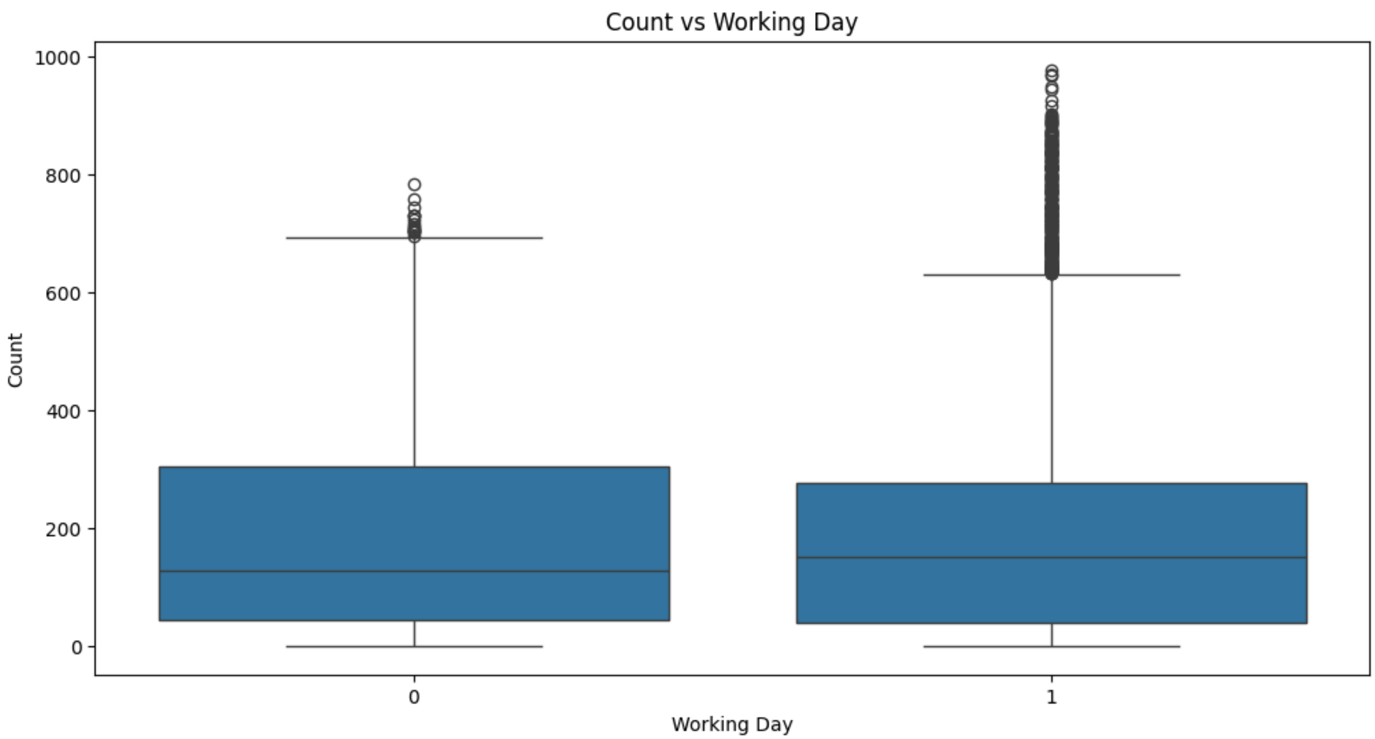




Distribution plots of continuous variables

Bivariate Analysis

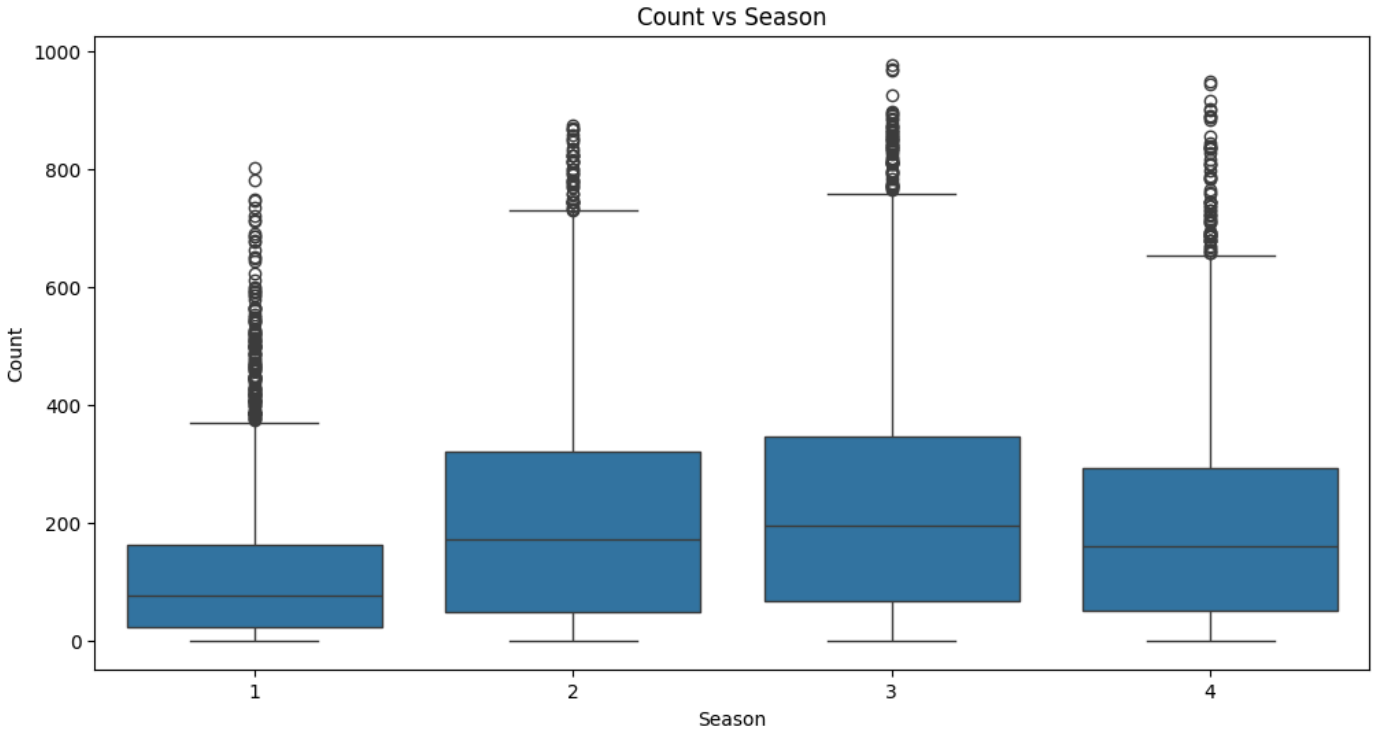
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='workingday', y='count', data=df)
plt.title('Count vs Working Day')
plt.xlabel('Working Day')
plt.ylabel('Count')
plt.show()
```



Boxplot for count vs working day.

The box plot suggests that the count of rental bikes may be slightly higher on working days compared to non-working days.

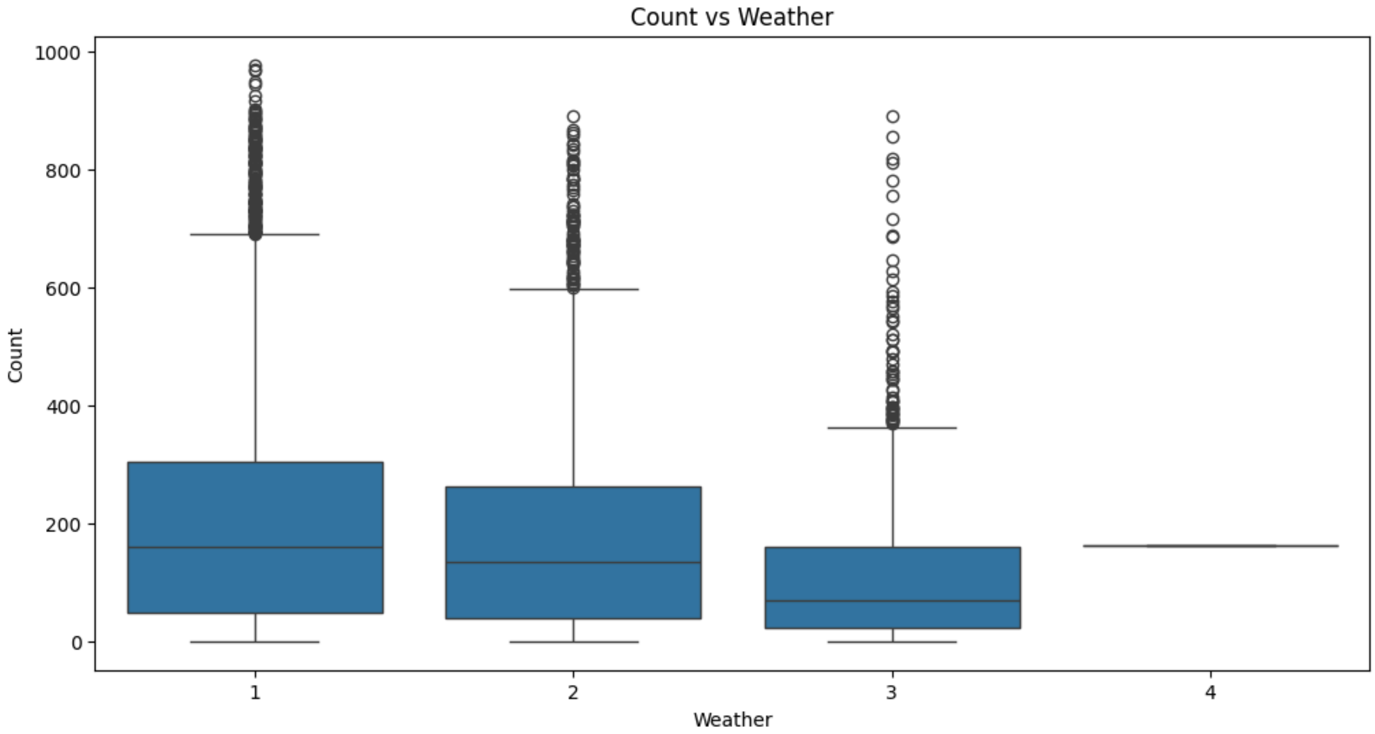
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='season', y='count', data=df)
plt.title('Count vs Season')
plt.xlabel('Season')
plt.ylabel('Count')
plt.show()
```



Boxplot for count vs season.

> Count vs Season: There are variations in the count of rental bikes across different seasons, with potentially higher counts during certain seasons.

```
plt.figure(figsize=(12, 6))
sns.boxplot(x='weather', y='count', data=df)
plt.title('Count vs Weather')
plt.xlabel('Weather')
plt.ylabel('Count')
plt.show()
```



Boxplot for count vs weather

> Count vs Weather: The box plot shows variations in the count of rental bikes based on weather conditions, with fewer rentals during adverse weather conditions.

Analysis on EDA

Comments on EDA :

- Shape of data: The dataset contains 10886 rows and 12 columns.
- Data types: All attributes are of appropriate data types.
- Missing values: There are no missing values in the dataset.
- Statistical summary: Provides summary statistics for numerical attributes.

Univariate Analysis :

- Temperature: Approximately normal distribution.
- Humidity: Slightly right-skewed distribution.
- Windspeed: Right-skewed distribution with some outliers.
- Rental Counts: Right-skewed distribution with potential outliers.

- Weather Conditions: Majority of days have weather condition 1.
- Seasons: Dataset contains roughly equal observations across all four seasons.

Bivariate Analysis :

- Rental counts tend to be slightly higher on working days compared to non-working days.
- Rental counts vary across different seasons and weather conditions.

Hypothesis Testing

2- Sample T-Test

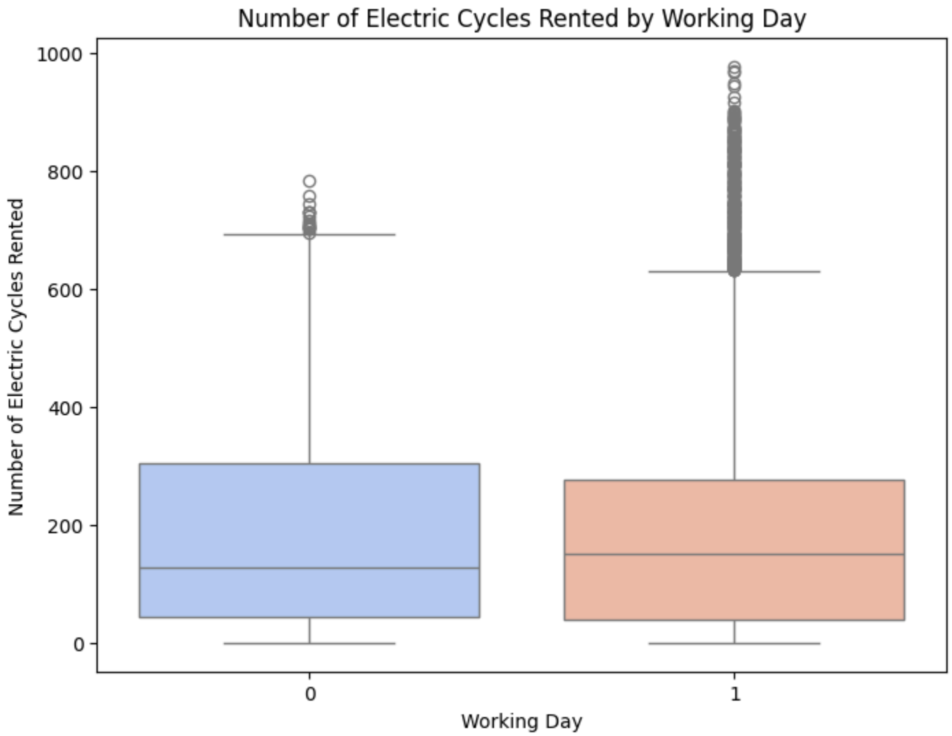
Problem Statement: To determine if there is a significant difference in the number of electric cycles rented between working days and non-working days.

- Null Hypothesis (H0): There is no significant difference in the number of electric cycles rented between working days and non-working days.
- Alternative Hypothesis (H1): There is a significant difference in the number of electric cycles rented between working days and non-working days.

Visual analysis according to the test

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='workingday', y='count', data=df, palette='coolwarm')
plt.title('Number of Electric Cycles Rented by Working Day')
plt.xlabel('Working Day')
plt.ylabel('Number of Electric Cycles Rented')
plt.show()
```

```
<ipython-input-57-4c8634bb20a2>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effec

  sns.boxplot(x='workingday', y='count', data=df, palette='coolwarm')
```



Selecting appropriate test and assumptions

> Selecting the appropriate test

- We will use a 2-sample t-test to compare the means of the number of electric cycles rented on working days and non-working days.

> Check test assumptions

> Assumption 1: Independence of observations

> Add blockquote

- We assume that the observations of the number of electric cycles rented on working days are independent from the observations on non-working days.

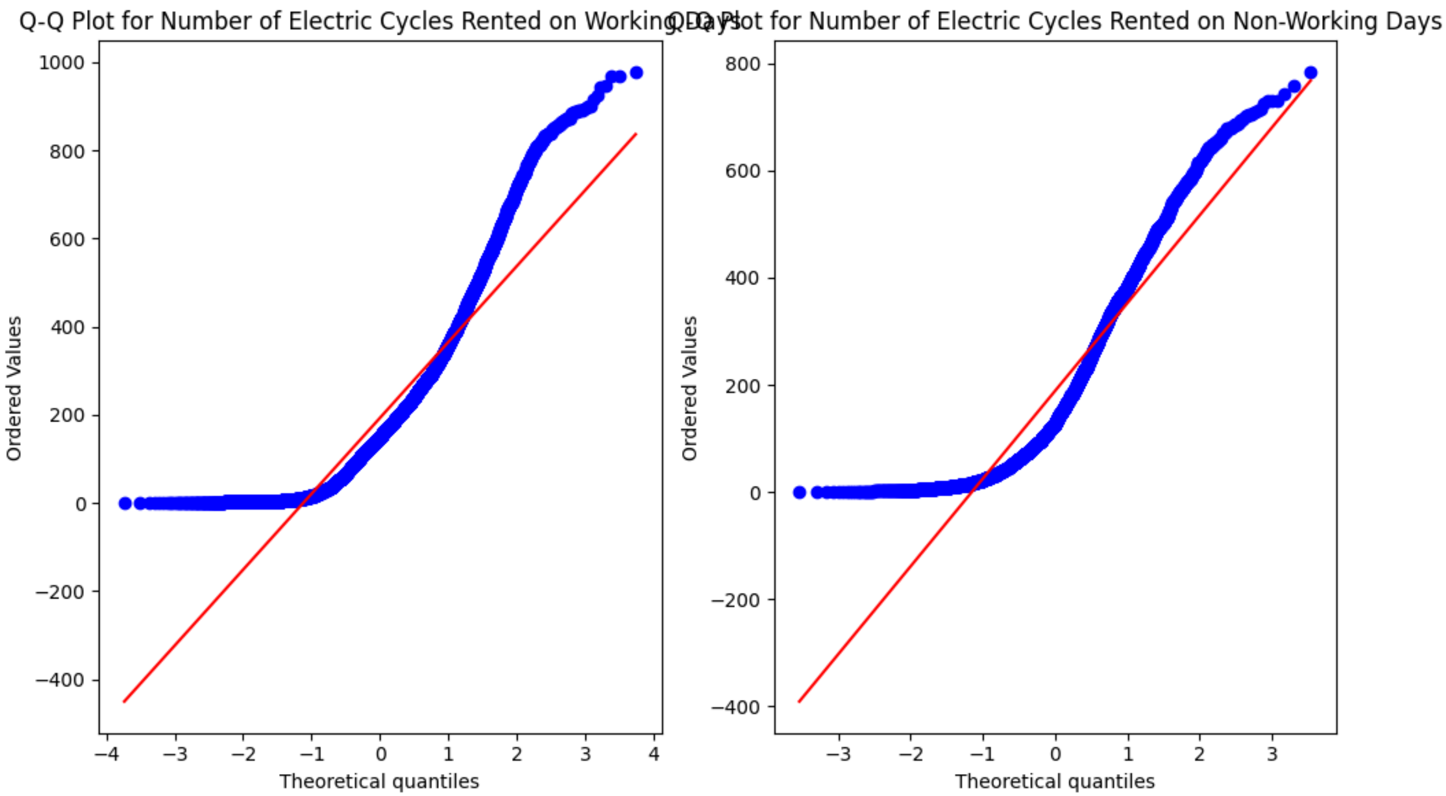
> Assumption 2: Normality of data

- We need to check if the data for the number of electric cycles rented on working days and non-working days are approximately normally distributed.

- We can visually inspect the distributions using histograms or Q-Q plots, and we can also perform normality tests such as Shapiro-Wilk test.

Let's check the normality assumption using Q-Q plots

```
plt.figure(figsize=(10, 6))

plt.subplot(1, 2, 1)
spy.probplot(df[df['workingday'] == 1]['count'], dist="norm", plot=plt)
plt.title('Q-Q Plot for Number of Electric Cycles Rented on Working Days')

plt.subplot(1, 2, 2)
spy.probplot(df[df['workingday'] == 0]['count'], dist="norm", plot=plt)
plt.title('Q-Q Plot for Number of Electric Cycles Rented on Non-Working Days')
plt.tight_layout()
plt.show()
```




```
workingday_yes = df[df['workingday'] == 1]['count']
workingday_no = df[df['workingday'] == 0]['count']

t_stat, p_value = spy.ttest_ind(workingday_yes, workingday_no)
```

```
print("2-Sample T-Test for Working Day:")
print("Test Statistic:", t_stat)
print("P-Value:", p_value)

alpha = 0.05

if p_value < alpha:
    print("Conclusion: Reject the null hypothesis. There is a significant difference in the number of electric cycles rented between working days and non-working days.
else:
    print("Conclusion: Fail to reject the null hypothesis. There is no significant difference in the number of electric cycles rented between working days and non-work
```

```
2-Sample T-Test for Working Day:
Test Statistic: 1.2096277376026694
P-Value: 0.22644804226361348
Conclusion: Fail to reject the null hypothesis. There is no significant difference in the number of electric cycles rented between working days and non-working da
```

Sample T-test to check if Working Day has an effect on the number of electric cycles rented.

After succesfully testing the hypothesis we conclude that There is no significant difference in the number of electric cycles rented between working days and non-working days

ANNOVA

ANNOVA to check if No. of cycles rented is similar or different in different 1. weather 2. season

- Null Hypothesis (H0): The mean number of cycles rented is equal across all weather conditions or seasons.
- Alternative Hypothesis (H1): The mean number of cycles rented is not equal across all weather conditions or seasons.
- Significance Level (alpha): 0.05.

```
f_stat_weather, p_value_weather = spy.f_oneway(df['count'][df['weather'] == 1],
                                               df['count'][df['weather'] == 2],
                                               df['count'][df['weather'] == 3],
                                               df['count'][df['weather'] == 4])

print("\nANOVA for Weather:")
print("F-Statistic:", f_stat_weather)
print("P-Value:", p_value_weather)

if p_value_weather < alpha:
    print("Conclusion: Reject the null hypothesis. The mean number of cycles rented is different across different weather conditions.")
else:
    print("Conclusion: Fail to reject the null hypothesis. The mean number of cycles rented is similar across different weather conditions.")
```

```
ANOVA for Weather:
F-Statistic: 65.53024112793271
P-Value: 5.482069475935669e-42
Conclusion: Reject the null hypothesis. The mean number of cycles rented is different across different weather conditions.
```

Anova results for Weather after testing the hypothesis we conclude that the mean number of cycles rented is different across different weather

```
f_stat_season, p_value_season = spy.f_oneway(df['count'][df['season'] == 1],
                                             df['count'][df['season'] == 2],
                                             df['count'][df['season'] == 3],
                                             df['count'][df['season'] == 4])

print("\nANOVA for Season:")
print("F-Statistic:", f_stat_season)
print("P-Value:", p_value_season)

if p_value_season < alpha:
    print("Conclusion: Reject the null hypothesis. The mean number of cycles rented is different across different seasons.")
else:
    print("Conclusion: Fail to reject the null hypothesis. The mean number of cycles rented is similar across different seasons.")
```

```
ANOVA for Season:
F-Statistic: 236.94671081032106
P-Value: 6.164843386499654e-149
Conclusion: Reject the null hypothesis. The mean number of cycles rented is different across different seasons.
```

Chi-square test

Chi-square test to check if Weather is dependent on the season

- Null Hypothesis (H0): Weather and season are independent of each other.
- Alternative Hypothesis (H1): Weather and season are dependent on each other.
- Significance Level (alpha): 0.05.

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

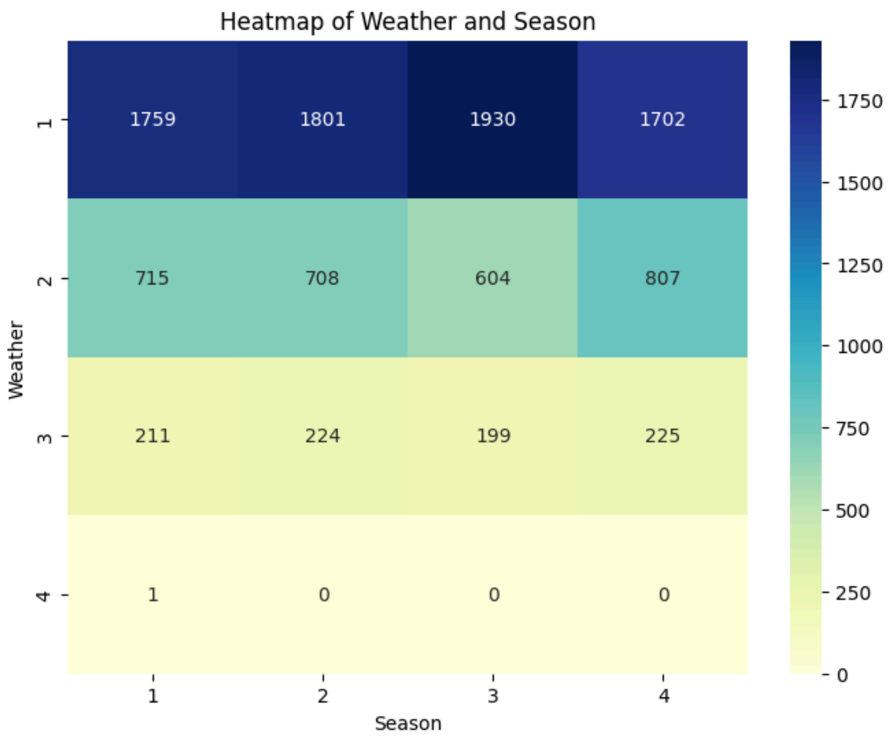
Hypothesis Testing

Chi-square test

Chi-square test to check if Weather is dependent on the season

- Null Hypothesis (H0): Weather and season are independent of each other.
- Alternative Hypothesis (H1): Weather and season are dependent on each other.
- Significance Level (alpha): 0.05.

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

plt.figure(figsize=(8, 6))
sns.heatmap(weather_season_cross, annot=True, cmap='YlGnBu', fmt='d')
plt.title('Heatmap of Weather and Season')
plt.xlabel('Season')
plt.ylabel('Weather')
plt.show()
```



```
chi2_stat, p_value_chi2, dof, expected = spy.chi2_contingency(weather_season_cross)

print("\nChi-Square Test for Weather and Season:")
print("Chi-Square Statistic:", chi2_stat)
print("P-Value:", p_value_chi2)

if p_value_chi2 < alpha:
    print("Conclusion: Reject the null hypothesis. Weather and season are dependent on each other.")
else:
    print("Conclusion: Fail to reject the null hypothesis. Weather and season are independent of each other.")
```

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

Conclusion

Insights and recommendation's

Insights based on the analysis and hypothesis testing done

> **1. Optimizing Service Availability :**

- Since there is no significant difference in rental counts between working days and non-working days, Yulu can consider adjusting its operational hours to better cater to customer demand patterns.
- For instance, if there are specific time slots during non-working days where demand is consistently high, Yulu could extend its operating hours during those periods to accommodate more riders.

> **2. Promotional Strategies :**

- Given the significant variation in rental counts across different weather conditions and seasons, Yulu can develop targeted promotional campaigns to incentivize ridership during less favorable conditions.
- For example, offering discounts or rewards for rides taken during rainy or colder seasons can encourage riders to choose Yulu's electric cycles over other modes of transportation.

> **3. Consistency in Rental Demand :**

- There is consistent demand for Yulu's shared electric cycles across both working and non-working days, indicating that the service is utilized regularly by users regardless of the day of the week.

> **4. Weather-Specific Service Enhancements :**

- Yulu can invest in weather-specific enhancements to improve the user experience and ensure rider safety during adverse weather conditions. For instance, providing rain covers or waterproof accessories for electric cycles during monsoon seasons can make riding more comfortable and appealing to users.

> **5. Dependency between Weather and Season :**

- There is a dependency between weather conditions and seasons, suggesting that certain weather patterns are more prevalent during specific seasons.
- This highlights the importance of considering both weather and seasonality factors when planning operational activities and promotional strategies.

> **6. Seasonal Fleet Management :**

- Based on the dependency between weather and season, Yulu can adopt a dynamic fleet management approach to optimize resource allocation. This could involve adjusting the distribution of electric cycles across different zones or neighborhoods based on anticipated changes in weather patterns and seasonal demand fluctuations.

> **7. Customer Communication and Education :**

- Yulu can proactively communicate with users about how weather conditions may impact their riding experience and provide tips for riding safely in different conditions.
- By educating users on the benefits of using Yulu's service regardless of weather or season, the company can foster greater loyalty and engagement among its customer base.

> **8. Collaboration with Local Authorities :**

- Yulu can collaborate with local authorities to implement infrastructure improvements that support safe and convenient riding experiences year-round. This could include initiatives such as expanding dedicated bike lanes, installing sheltered bike parking facilities, or improving road conditions in areas with high ridership.
- By implementing these recommendations, Yulu can further enhance its service offerings, attract more users, and establish itself as a reliable and sustainable micro-mobility solution in the Indian market.

> **9. Effect of Working Day on Rental Counts :**

- The 2-sample t-test results suggest that there is no significant difference in the number of electric cycles rented between working days and non-working days.
- This implies that Yulu's service is utilized consistently across both working and non-working days.
- Recommendation: Yulu's R&D team should focus on maintaining a consistent level of service availability and promotion strategies across all days of the week.

> **10. Effect of Weather and Season on Rental Counts :**

- The ANOVA tests indicate that the mean number of electric cycles rented varies significantly across different weather conditions and seasons. This suggests that weather and seasonal factors play a role in influencing customer demand.
- Recommendation: Yulu should adapt its operations and marketing strategies based on weather forecasts and seasonal trends. For instance, offering promotions during favorable weather conditions or seasonal events can help boost ridership.

> **11. Dependency between Weather and Season :**

- The Chi-square test results show that weather and season are dependent on each other.
- This implies that certain weather conditions are more prevalent during specific seasons.
- Recommendation: Yulu should consider the interplay between weather and season when planning operational activities, such as maintenance schedules or fleet adjustments.

Final Analysis

Overall Insights :

- The data-driven analysis provides valuable insights into the factors influencing the demand for Yulu's shared electric cycles. Understanding these factors can help Yulu optimize its operations, enhance user experience, and drive business growth.
- The main profit that lie's for yulu is in accomodating close to perfect number of bikes on specific location's according to predicted need for user's such that the whole fleet is used efficiently and the potential is not wasted.
- Yulu's R&D team should continue monitoring and analyzing rental patterns to identify emerging trends and opportunities for improvement.

Future Directions :

- Further analysis could explore additional variables such as time of day, geographical location, or promotional activities to gain deeper insights into rental patterns.
- Yulu could also consider implementing predictive analytics models to forecast demand and optimize resource allocation.
- Predictive analaytics model are a way to sustain and make the best out of yulu's bike fleet rather than increasing the fleet size they could first find out if they can optimize and make user's availabillity for bikes in heavy usage areas.