

Yulu - Hypothesis Testing



About Yulu

Yulu, India's pioneering micro-mobility service provider, has embarked on a mission to revolutionize daily commutes by offering unique, sustainable transportation solutions.

However, recent revenue setbacks have prompted Yulu to seek the expertise of a consulting company to delve into the factors influencing the demand for their shared electric cycle specifically in the Indian market.

Analysis goal

- Which variables play a key role in predicting the demand for shared electric cycles in India?
- How effectively these variables explain the demand for electric cycles.

Exploratory Data Analysis

```
# Importing the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import matplotlib.cm as cm
from scipy.stats import ttest_ind, f_oneway, levene, kruskal, shapiro, chi2_contingency
from statsmodels.stats.weightstats import ztest

import warnings
warnings.filterwarnings("ignore")
```

```
#loading the dataset
df = pd.read_csv('bike_sharing.csv')
```

Analysing Basic Metrics

```
# Top 5 rows of the dataframe
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

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

```
# Bottom 5 rows of the dataframe
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

```
#Checking Shape of dataset
df.shape
```

```
(10886, 12)
```

```
#columns in the dataset
df.columns
```

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

```
#Checking dataset structure and features
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
```

Missing Value Analysis

```
#Checking missing values
df.isnull().sum()
```

```
0
datetime    0
season      0
holiday     0
workingday  0
weather     0
temp        0
atemp       0
humidity    0
windspeed   0
casual      0
registered  0
count       0
```

```
dtype: int64
```

- **Data Overview:** The dataset contains **10886 rows** and **12 columns**.
- **Mixed Data Types:** The dataset includes **3 float** attributes and **8 integer** attributes. The **datetime** column is currently an **object** type and needs to be converted to datetime.
- **Numerical Attributes:**
 - **3 float attributes:** temp, atemp, windspeed.
 - **8 integer attributes:** season, holiday, workingday, weather, casual, registered, count, humidity.
- **Categorical Attributes:** The **season**, **holiday**, **workingday**, and **weather** columns are represented as integers and need to be converted to more meaningful categorical labels.
- **No Missing Values:** There are no null or missing values in any row or column.

👁️ checking for duplicates

```
#Checking for duplicates
df.duplicated().sum()
```

0

💡 Insights:

- No duplicates found in dataset

✅ Sanity Check: Unique Values per Column

```
# checking Uniques values of each columns
df.nunique()
```

	0
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

🔄 Changing the Datatype of Columns

```
# List of categorical columns to be converted
categorical_columns = ['season', 'holiday', 'workingday', 'weather']

# Convert categorical columns to 'category' type
for column in categorical_columns:
    df[column] = df[column].astype('category')

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  category
```

```
2 holiday      10886 non-null category
3 workingday   10886 non-null category
4 weather      10886 non-null category
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: category(4), float64(3), int64(4), object(1)
memory usage: 723.7+ KB
```

Datetime Extraction and Categorical Mapping

```
# Converting datetime column into date time format
df['datetime'] = pd.to_datetime(df['datetime'])
df['datetime'].dtype
```

```
# adding new columns for better understanding
df['hour'] = df['datetime'].dt.hour
df['day'] = df['datetime'].dt.day
df['month'] = df['datetime'].dt.month
df['year'] = df['datetime'].dt.year
df['weekday'] = df['datetime'].dt.weekday
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	hour	day	month
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	0	1	1
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	1	1	1
	2011-01-														

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

```
# Mapping numerical values to categorical labels
df['season'] = df['season'].replace({1: 'Spring', 2: 'Summer', 3: 'Fall', 4: 'Winter'})
df['holiday'] = df['holiday'].replace({0: 'No', 1: 'Yes'})
df['workingday'] = df['workingday'].replace({0: 'No', 1: 'Yes'})
df['weather'] = df['weather'].replace({1: 'Clear/Partly Cloudy', 2: 'Mist', 3: 'Light Snow/Rain', 4: 'Heavy Rain/Snow'})
df['month'] = df['month'].replace({1: 'January', 2: 'February',
                                   3: 'March', 4: 'April', 5: 'May', 6: 'June', 7: 'July', 8: 'August',
                                   9: 'September', 10: 'October', 11: 'November', 12: 'December'})
df['weekday'] = df['weekday'].replace({0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4: 'Friday', 5: 'Saturday', 6: 'Su
```

Statistical Summary

```
df.describe(include = 'category')
```

	season	holiday	workingday	weather
count	10886	10886	10886	10886
unique	4	2	2	4
top	Winter	No	Yes	Clear/Partly Cloudy
freq	2734	10575	7412	7192

Insights:

- 1) **Season:** Winter has the highest frequency, suggesting that bike rentals are more common during colder months.
 - 4 unique values:** Winter, Spring, Summer, Fall.
 - Most frequent:** Winter (2,734 occurrences).
- 2) **Holiday:** Most data is from non-holiday days, highlighting the dominance of regular rental patterns.
 - 2 unique values:** Yes, No.
 - Most frequent:** No (10,575 occurrences).

- 3) **Workingday:** Rentals are more common on working days, showing typical weekday rental trends.
 - **2 unique values:** Yes, No.
 - **Most frequent:** Yes (7,412 occurrences).
- 4) **Weather:** Clear/Partly Cloudy weather is the most frequent, indicating that bike rentals peak during pleasant weather conditions.
 - **4 unique values:** Clear/Partly Cloudy, Mist, Light Snow/Rain, Heavy Rain/Snow.
 - **Most frequent:** Clear/Partly Cloudy(7,192 occurrences).

```
df.describe().transpose()
```

	count	mean	min	25%	50%	75%	max	std
datetime	10886	2011-12-27 05:56:22.399411968	2011-01-01 00:00:00	2011-07-02 07:15:00	2012-01-01 20:30:00	2012-07-01 12:45:00	2012-12-19 23:00:00	NaN
temp	10886.0	20.23086	0.82	13.94	20.5	26.24	41.0	7.79159
atemp	10886.0	23.655084	0.76	16.665	24.24	31.06	45.455	8.474601
humidity	10886.0	61.88646	0.0	47.0	62.0	77.0	100.0	19.245033
windspeed	10886.0	12.799395	0.0	7.0015	12.998	16.9979	56.9969	8.164537
casual	10886.0	36.021955	0.0	4.0	17.0	49.0	367.0	49.960477
registered	10886.0	155.552177	0.0	36.0	118.0	222.0	886.0	151.039033
count	10886.0	191.574132	1.0	42.0	145.0	284.0	977.0	181.144454
hour	10886.0	11.541613	0.0	6.0	12.0	18.0	23.0	6.915838
day	10886.0	9.992559	1.0	5.0	10.0	15.0	19.0	5.476608

💡 Insights:

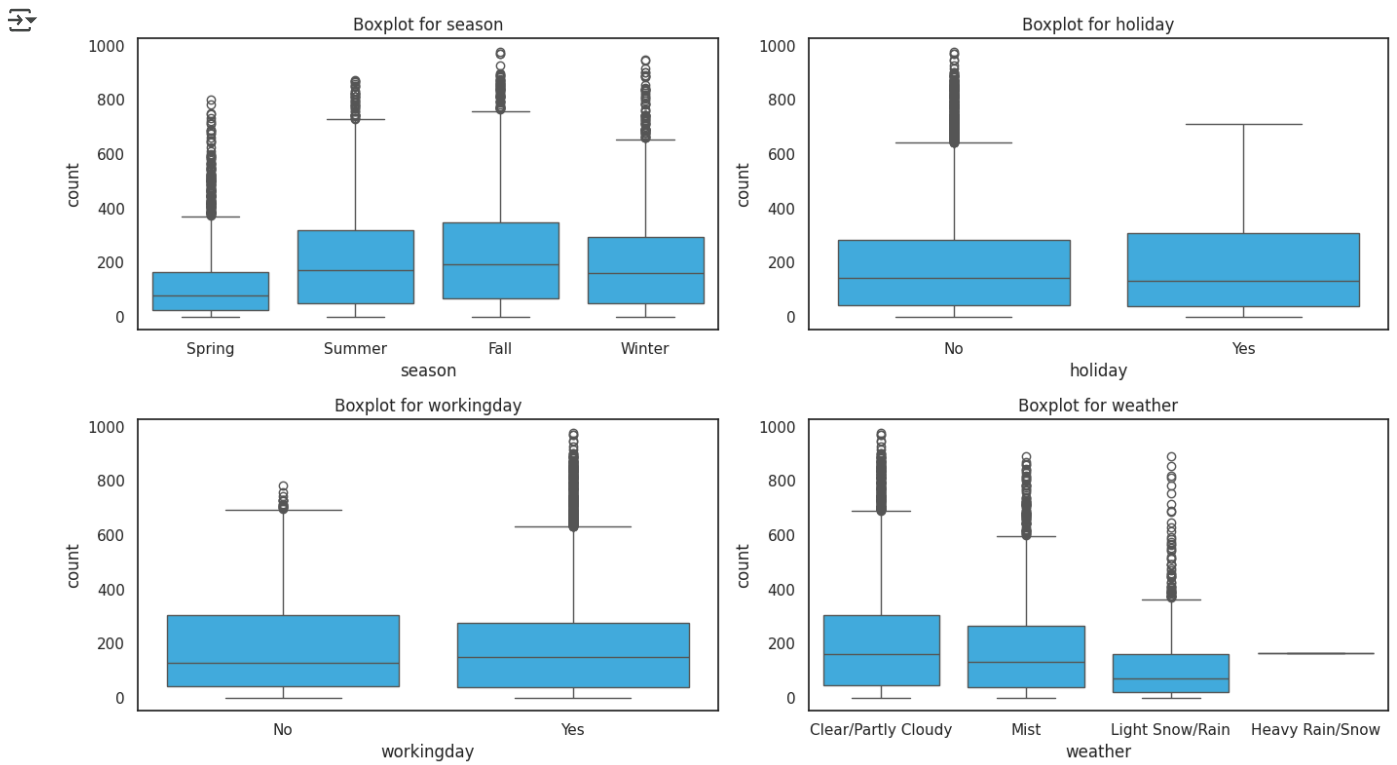
- **Weather & Rentals:** Rentals are higher on sunny days, and temperature and weather conditions (humidity, wind) influence demand.
- **User Behavior:** Registered users dominate rental activity, with casual rentals showing more variation.
- **Skewness:**
 - Rental count is right-skewed, with a few days having significantly higher rentals.
 - Temperature and windspeed distributions may also show slight skewness.

👁️ Detect Outliers

```
# Plotting boxplots for categorical columns
plt.figure(figsize=(14, 8))
sns.set(style="white")

# Create a boxplot for each categorical column
for i, column in enumerate(categorical_columns, 1):
    plt.subplot(2, 2, i) # 2 rows, 2 columns grid
    sns.boxplot(x=column, y='count', data=df, color="#29B6F6")
    plt.title(f'Boxplot for {column}')

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



💡 Insights:

- **Seasonal Outliers:**
 - **Spring** and **Winter** show **more outliers** compared to Summer and Fall, suggesting unusual patterns in these seasons.
- **Weather-Related Outliers:**
 - **Light Drizzle** has **more outliers**, while Heavy Rain/Snow has none, indicating stable rental patterns during heavy weather.
- **Workday vs. Holiday Outliers:**
 - **More outliers** are found **on workdays** than holidays, suggesting irregular rental behavior on regular workdays.

✓ To Calculate Interquartile Range (IQR) to Identify Outliers

```
# Function to calculate IQR and identify outliers
def find_outliers(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data[(data < lower_bound) | (data > upper_bound)]

# Identify outliers in 'count' column
outliers = find_outliers(df['count'])
print(f'Outliers in count:\n', outliers)
```

```
Outliers in count:
6611    712
6634    676
6635    734
6649    662
6658    782
...
10678   724
10702   688
10726   679
10846   662
```

10870 678
Name: count, Length: 300, dtype: int64

```
# Function to clip outliers
def clip_outliers(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data.clip(lower=lower_bound, upper=upper_bound)

df['count'] = clip_outliers(df['count'])
```

Non graphical analysis

```
# Count of unique combinations of the specified columns
df[['season', 'holiday', 'workingday', 'weather', 'month', 'day', 'hour']].value_counts()
```

								count
season	holiday	workingday	weather	month	day	hour		
Spring	No	Yes	Clear/Partly Cloudy	March	8	22		2
				January	12	14		2
Fall	No	Yes	Mist	August	3	11		2
			Clear/Partly Cloudy	July	12	22		2
						21		2
...
Summer	No	Yes	Clear/Partly Cloudy	June	3	10		1
						9		1
						7		1
						6		1
Winter	Yes	No	Light Snow/Rain	October	8	22		1

9540 rows × 1 columns

dtype: int64

- Insights:
- **Seasonal Peaks:** **Spring** and **Summer**, particularly with clear/partly cloudy weather, see high bike rental activity.
 - **Workday Influence:** Rentals are more common **on working days** compared to holidays.
 - **Weather Impact:** **Clear weather** conditions significantly boost rentals.
 - **Monthly Patterns:** **March** and **June** show higher rental counts, indicating peak usage during these months.
 - **Hourly Trends:** Higher rental counts are observed in the **late evening** hours around 22:00.

```
# Checking the unique values for columns
for column in df.columns:
    unique_values = df[column].unique()
    print(f'Unique Values in {column} column are :-')
    print(unique_values[:7])
    print('_' * 70)
```

<DatetimeArray>

['2011-01-01 00:00:00', '2011-01-01 01:00:00', '2011-01-01 02:00:00', '2011-01-01 03:00:00', '2011-01-01 04:00:00', '2011-01-01 05:00:00', '2011-01-01 06:00:00']

Length: 7, dtype: datetime64[ns]

Unique Values in season column are :-

['Spring', 'Summer', 'Fall', 'Winter']

Categories (4, object): ['Spring', 'Summer', 'Fall', 'Winter']

```
[ 'No', 'Yes' ]
Categories (2, object): ['No', 'Yes']

Unique Values in workingday column are :-
['No', 'Yes']
Categories (2, object): ['No', 'Yes']

Unique Values in weather column are :-
['Clear/Partly Cloudy', 'Mist', 'Light Snow/Rain', 'Heavy Rain/Snow']
Categories (4, object): ['Clear/Partly Cloudy', 'Mist', 'Light Snow/Rain', 'Heavy Rain/Snow']

Unique Values in temp column are :-
[ 9.84  9.02  8.2  13.12 15.58 14.76 17.22]

Unique Values in atemp column are :-
[14.395 13.635 12.88  17.425 19.695 16.665 21.21 ]

Unique Values in humidity column are :-
[81 80 75 86 76 77 72]

Unique Values in windspeed column are :-
[ 0.          6.0032 16.9979 19.0012 19.9995 12.998  15.0013]

Unique Values in casual column are :-
[ 3  8  5  0  2  1 12]

Unique Values in registered column are :-
[13 32 27 10  1  0  2]

Unique Values in count column are :-
[16 40 32 13  1  2  3]

Unique Values in hour column are :-
[0 1 2 3 4 5 6]

Unique Values in day column are :-
[1 2 3 4 5 6 7]

Unique Values in month column are :-
['January' 'February' 'March' 'April' 'May' 'June' 'July']

Unique Values in year column are :-
[2011 2012]

Unique Values in weekday column are :-
['Saturday' 'Sunday' 'Monday' 'Tuesday' 'Wednesday' 'Thursday' 'Friday']
```

Visual Analysis-Univariate & Bivariate

Univariate Analysis

For Continuous Variables

```
# List of continuous variables
group1_variables = ['temp', 'atemp', 'humidity', 'windspeed']

# subplot
fig1, axes1 = plt.subplots(nrows=2, ncols=2, figsize=(12, 8))

# Plotting histograms with KDE for each continuous variable
sns.histplot(df['temp'], bins=30, kde=True, color='#FF6347', ax=axes1[0, 0])
axes1[0, 0].set_title('Distribution of temp')
axes1[0, 0].set_xlabel('temp')
axes1[0, 0].set_ylabel('Frequency')

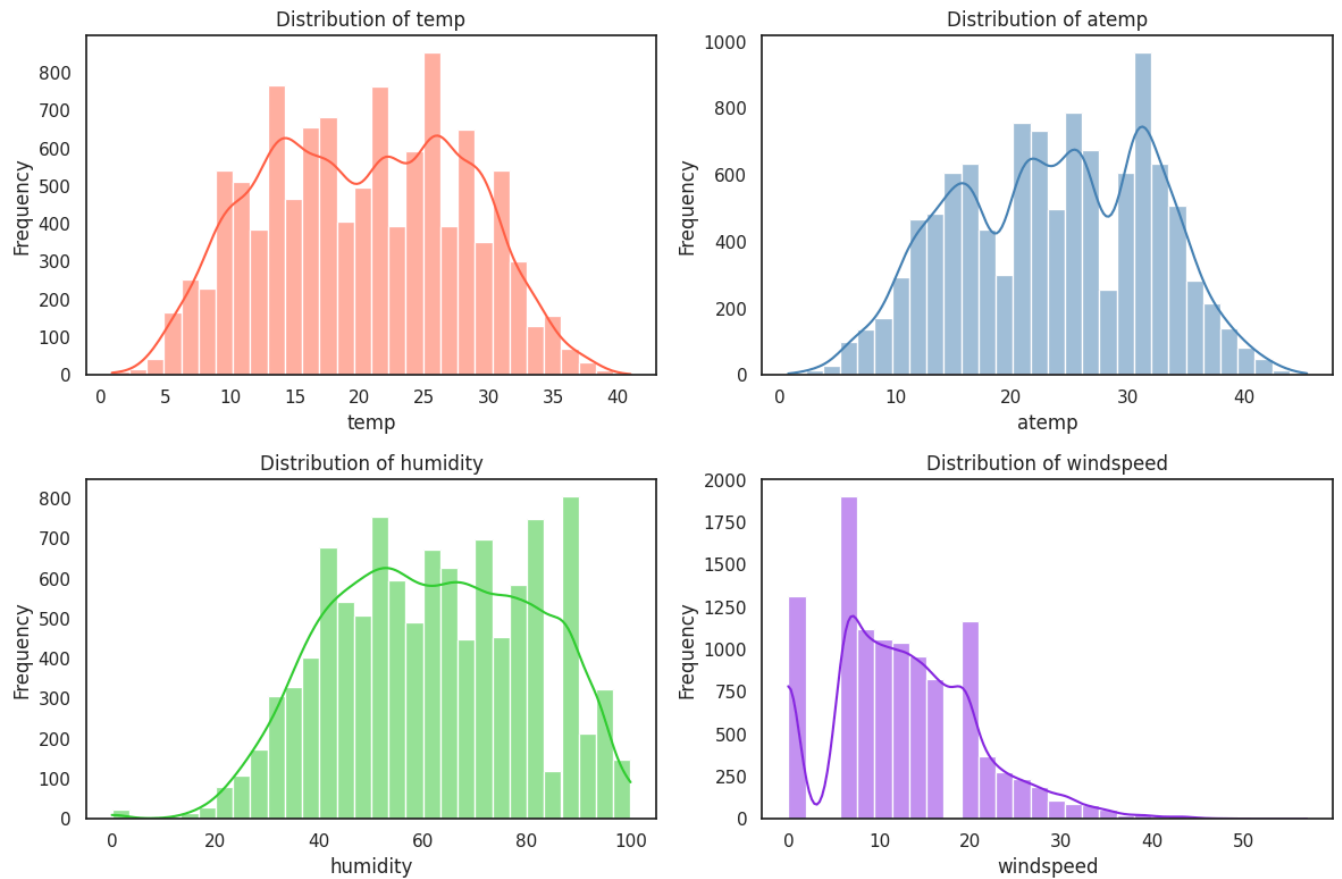
sns.histplot(df['atemp'], bins=30, kde=True, color='#4682B4', ax=axes1[0, 1])
axes1[0, 1].set_title('Distribution of atemp')
axes1[0, 1].set_xlabel('atemp')
axes1[0, 1].set_ylabel('Frequency')

sns.histplot(df['humidity'], bins=30, kde=True, color='#32CD32', ax=axes1[1, 0])
axes1[1, 0].set_title('Distribution of humidity')
axes1[1, 0].set_xlabel('humidity')
axes1[1, 0].set_ylabel('Frequency')

sns.histplot(df['windspeed'], bins=30, kde=True, color='#8A2BE2', ax=axes1[1, 1])
axes1[1, 1].set_title('Distribution of windspeed')
axes1[1, 1].set_xlabel('windspeed')
axes1[1, 1].set_ylabel('Frequency')
```



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



💡 Insights:

- **Temp:** Temperature values are centered around 25°C with a slight skew towards higher temperatures.
- **Atemp:** Feels-like temperature is mostly around 30, indicating perceived temperatures are generally higher.
- **Humidity:** Most humidity values fall between 60-70%, showing a tendency towards higher humidity.
- **Windspeed:** Wind speeds are mainly low, around 10, with a right-skewed distribution.

```
# List of continuous variables
group2_variables = ['casual', 'registered', 'count']

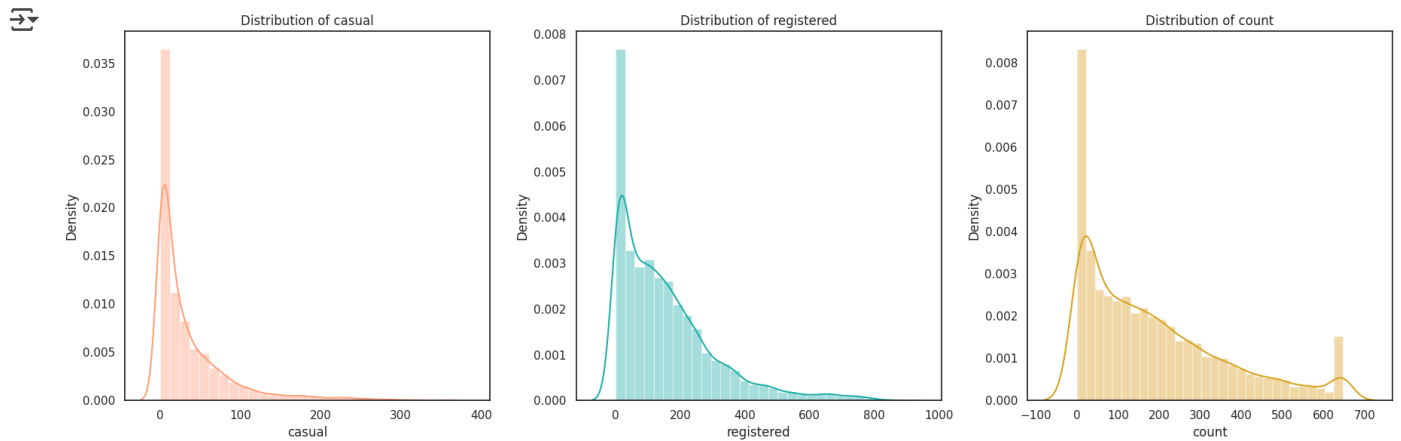
# subplot grid
fig2, axes2 = plt.subplots(nrows=1, ncols=3, figsize=(18, 6))

# Plotting distribution plots for each continuous variable
sns.distplot(df['casual'], bins=30, kde=True, color='#FFA07A', ax=axes2[0])
axes2[0].set_title('Distribution of casual')
axes2[0].set_xlabel('casual')
axes2[0].set_ylabel('Density')

sns.distplot(df['registered'], bins=30, kde=True, color='#20B2AA', ax=axes2[1])
axes2[1].set_title('Distribution of registered')
axes2[1].set_xlabel('registered')
axes2[1].set_ylabel('Density')

sns.distplot(df['count'], bins=30, kde=True, color='#DAA520', ax=axes2[2])
axes2[2].set_title('Distribution of count')
axes2[2].set_xlabel('count')
axes2[2].set_ylabel('Density')
```

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



💡 Insights:

- **Casual Users:** Most counts are low, peaking sharply near zero, indicating fewer casual users.
- **Registered Users:** Counts are higher with a strong central peak, showing consistent usage.
- **Total Rentals:** Mirrors registered users' pattern, highlighting their key contribution to overall rentals.

For Categorical Variables

```
# List of categorical variables
categorical_columns = ['season', 'holiday', 'workingday', 'weather']

# Setting up the subplot grid for count plots
fig1, axes1 = plt.subplots(nrows=2, ncols=2, figsize=(12, 8))
axes1 = axes1.flatten() # Flattening the array of axes for easy iteration

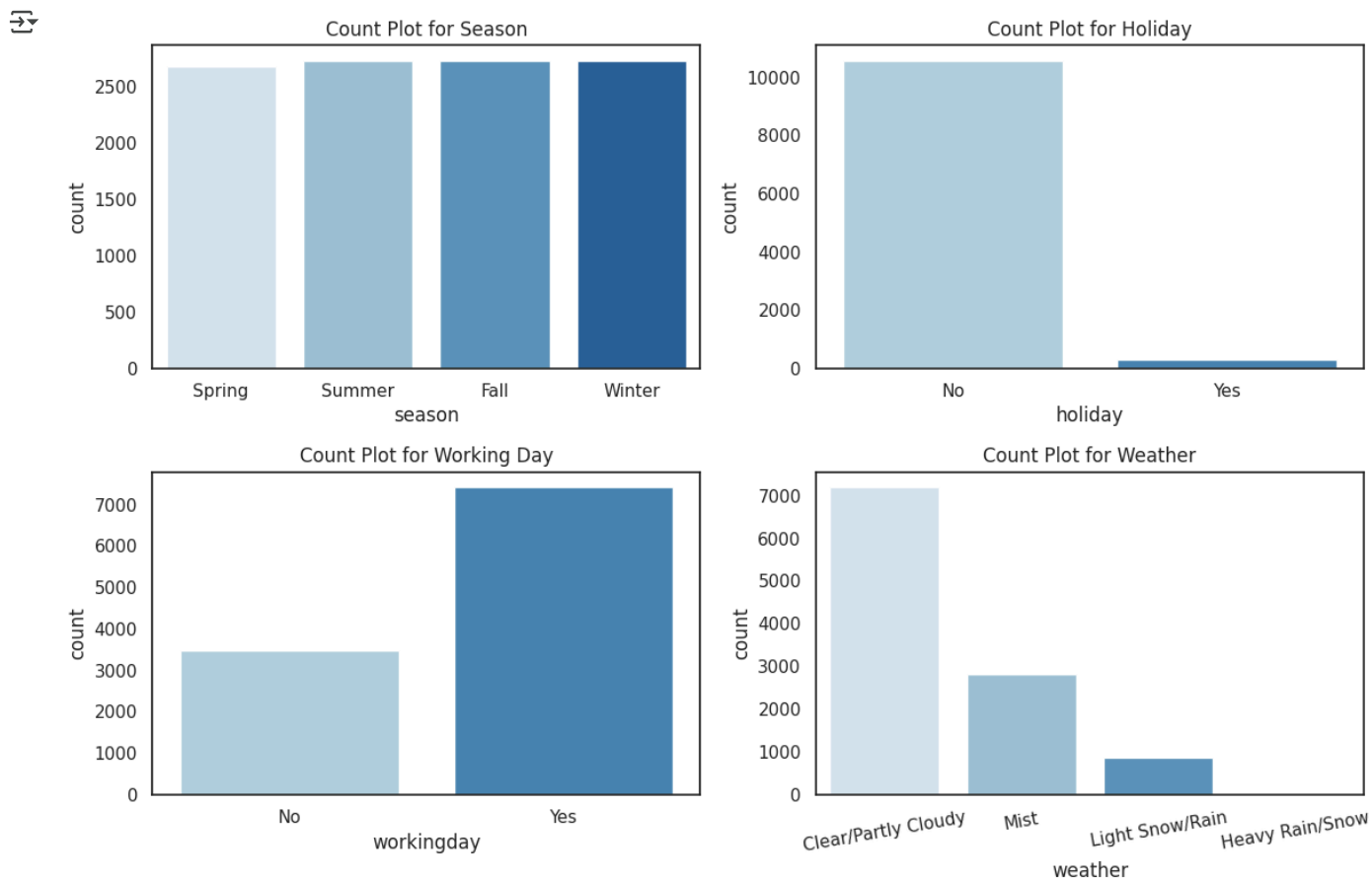
# Plotting count plots for each categorical variable
sns.countplot(x='season', data=df, palette='Blues', ax=axes1[0])
axes1[0].set_title('Count Plot for Season')

sns.countplot(x='holiday', data=df, palette='Blues', ax=axes1[1])
axes1[1].set_title('Count Plot for Holiday')

sns.countplot(x='workingday', data=df, palette='Blues', ax=axes1[2])
axes1[2].set_title('Count Plot for Working Day')

sns.countplot(x='weather', data=df, palette='Blues', ax=axes1[3])
axes1[3].set_title('Count Plot for Weather')
plt.xticks(rotation=10)

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



```
pastel_colors = sns.color_palette("pastel")

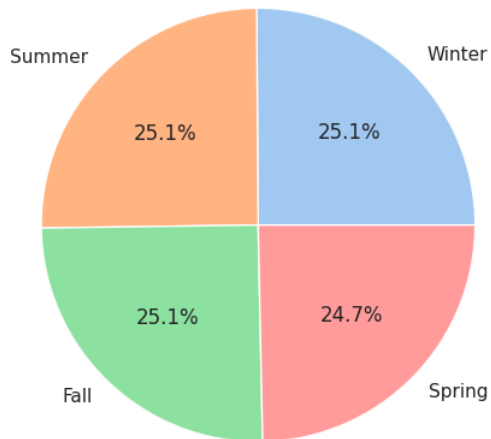
# subplot grid
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))
axes = axes.flatten()

# Plotting pie charts for each categorical variable
for i, column in enumerate(categorical_columns):
    df[column].value_counts().plot.pie(autopct='%1.1f%%', colors=pastel_colors, ax=axes[i])
    axes[i].set_title(f'Pie Chart for {column}')
    axes[i].set_ylabel('')

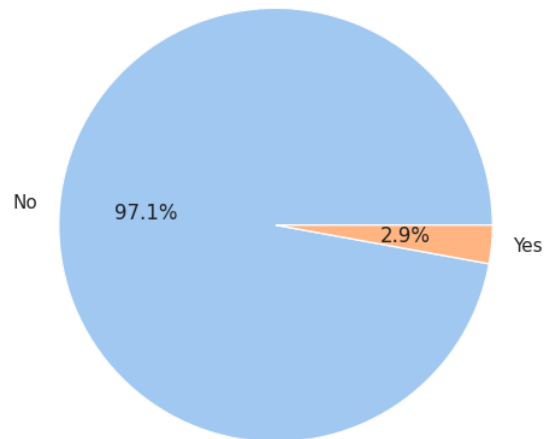
plt.tight_layout()
plt.show()
```



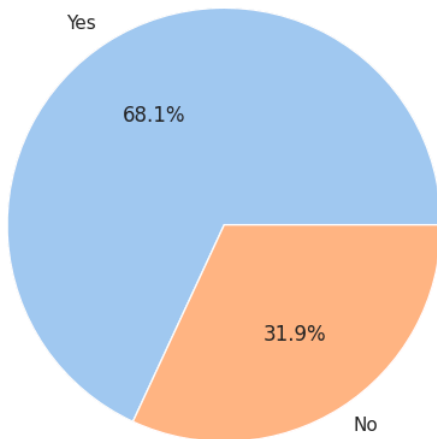
Pie Chart for season



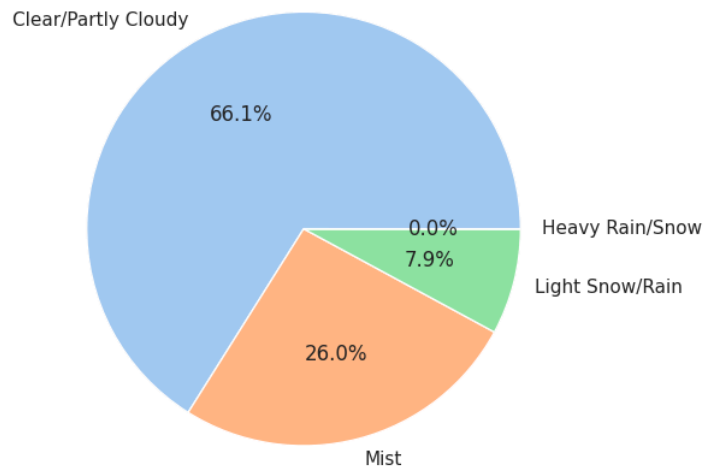
Pie Chart for holiday



Pie Chart for workingday



Pie Chart for weather

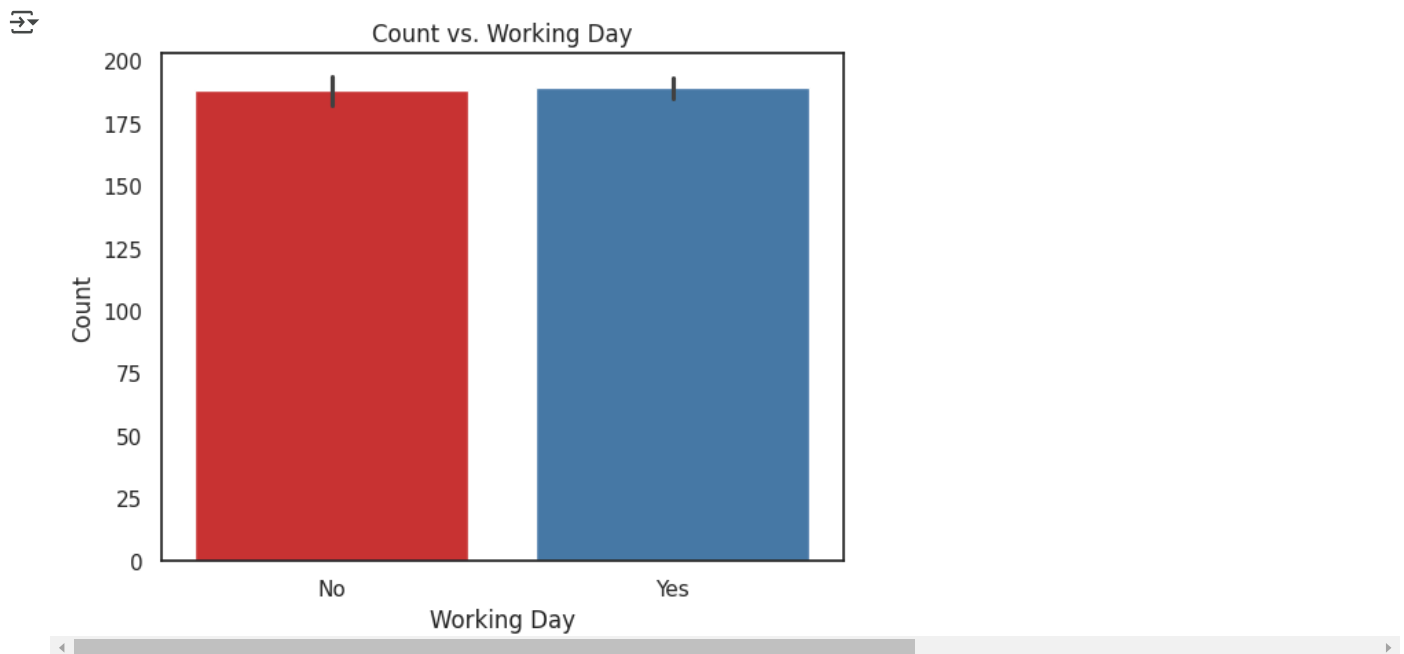


💡 Insights:

- **Season:** Rentals are fairly evenly distributed across seasons, indicating consistent year-round usage.
- **Holiday:** 97.1% of rentals occur on non-holidays, showing primary use for commuting.
- **Workday:** 68.1% of rentals are on working days, reinforcing commuting patterns.
- **Weather:** 66.1% of rentals happen in clear/partly cloudy conditions, highlighting the impact of favorable weather.

✓ Bivariate Analysis

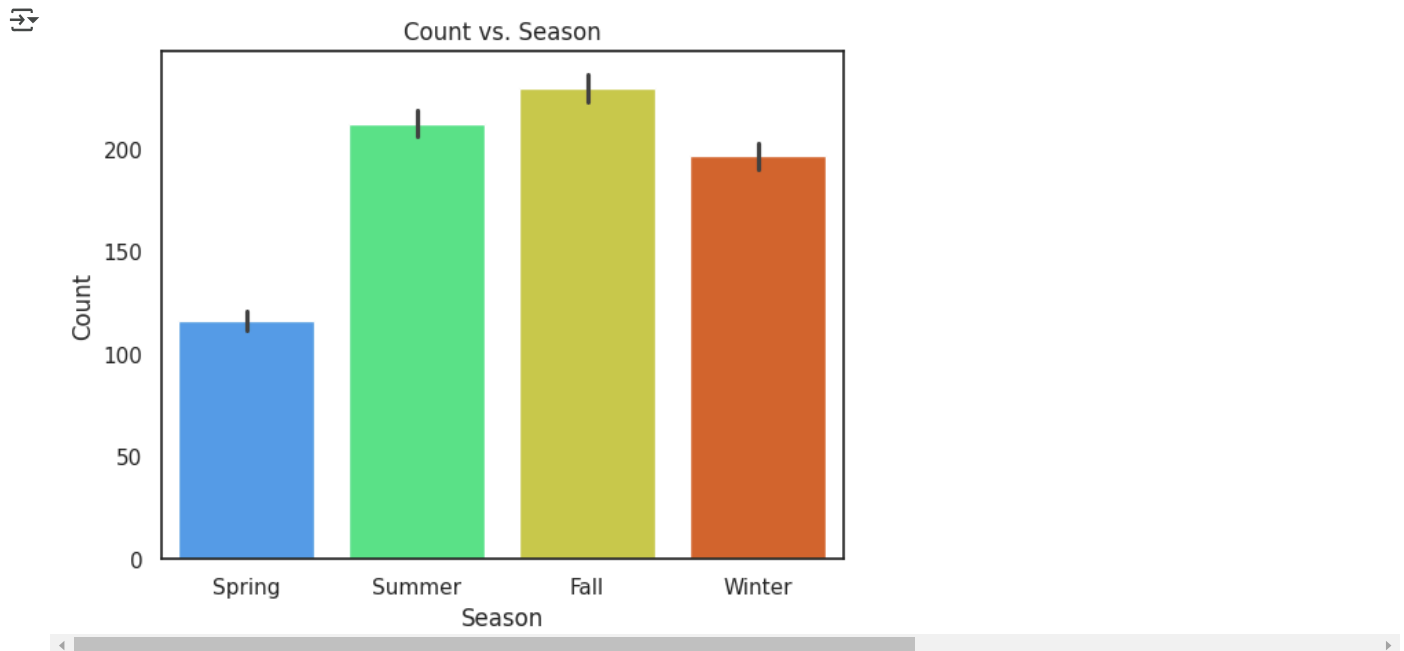
```
# Relationship between working day and count
sns.barplot(x='workingday', y='count', data=df, palette='Set1')
plt.title('Count vs. Working Day')
plt.xlabel('Working Day')
plt.ylabel('Count')
plt.show()
```



💡 **Insights:**

- **Higher Rentals on Working Days:** The count of bike rentals is higher on working days compared to non-working days. This suggests that people are more likely to rent bikes for commuting purposes during the workweek.

```
# Relationship between season and count
sns.barplot(x='season', y='count', data=df, palette='turbo')
plt.title('Count vs. Season')
plt.xlabel('Season')
plt.ylabel('Count')
plt.show()
```

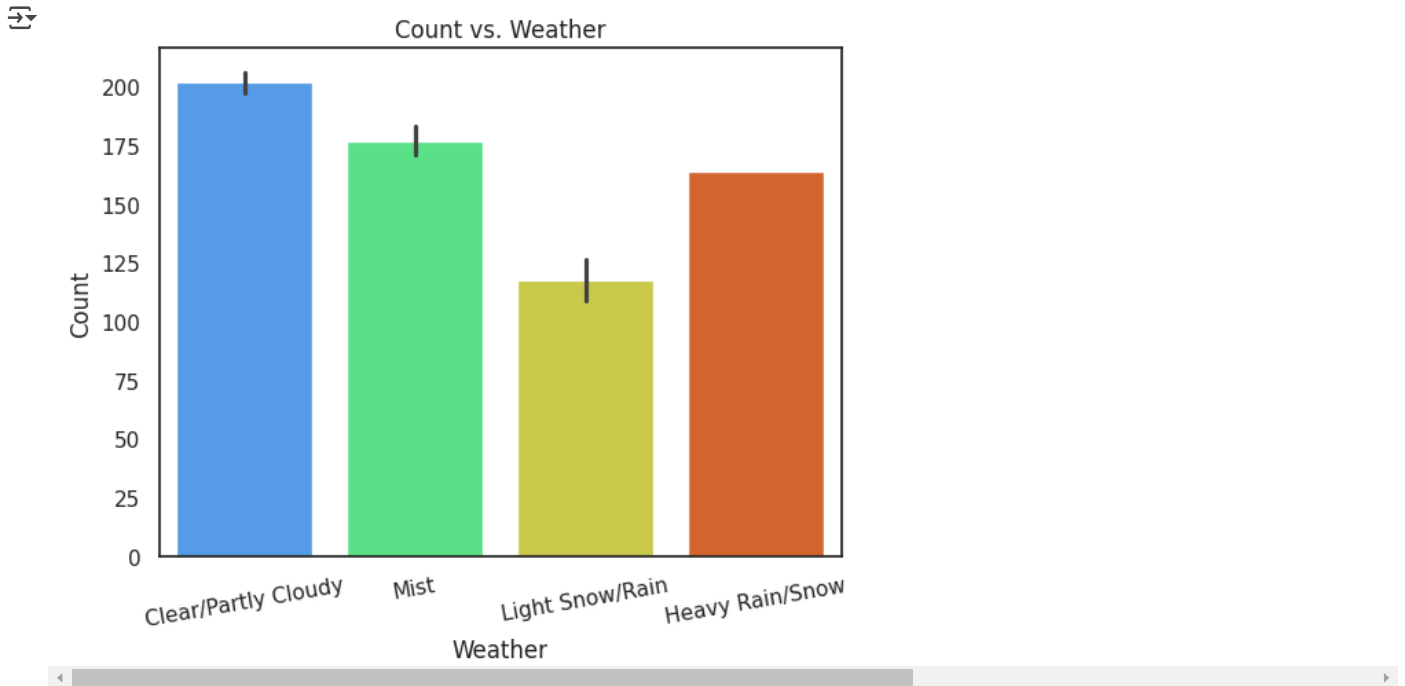


💡 **Insights:**

- **Fall:** Highest bike rental counts, indicating peak usage during this season.
- **Summer:** High rental counts, suggesting favorable conditions for biking.
- **Winter:** Moderate rental counts, possibly due to milder weather compared to other regions.
- **Spring:** Lowest rental counts, which might be influenced by transitional weather conditions.

```
# Relationship between weather and count
sns.barplot(x='weather', y='count', data=df, palette='turbo')
plt.title('Count vs. Weather')
```

```
plt.xlabel('Weather')
plt.ylabel('Count')
plt.xticks(rotation=10)
plt.show()
```

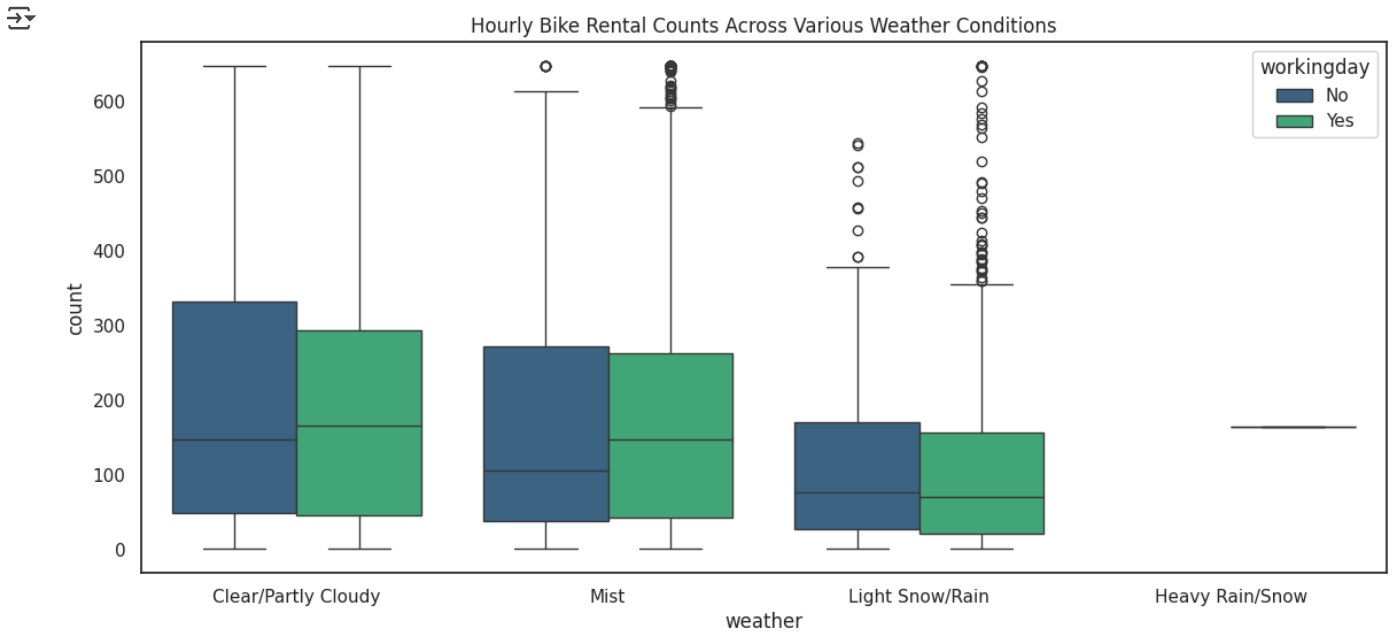


💡 Insights:

- **Clear/Partly Cloudy:** The highest rental counts, suggesting favorable weather encourages bike usage.
- **Mist:** High rental counts, indicating moderate weather still supports bike rentals.
- **Light Snow/Rain:** Lower rental counts, showing some decrease in usage during mild adverse weather.
- **Heavy Rain/Snow:** The lowest rental counts, indicating that severe weather discourages bike rentals.

Hourly Bike Rental Counts Across Various Weather Conditions

```
# Hourly Bike Rental Counts Across Various Weather Conditions
plt.figure(figsize=(14, 6))
plt.title('Hourly Bike Rental Counts Across Various Weather Conditions')
sns.boxplot(data=df, x='weather', y='count', hue='workingday', palette='viridis')
plt.show()
```

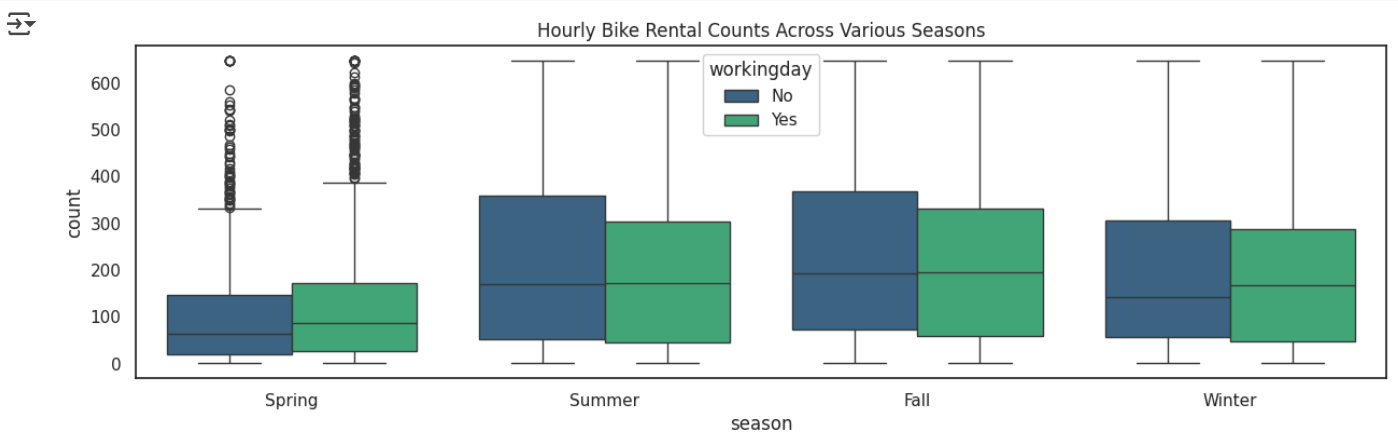


💡 Insights:

- **Clear/Partly Cloudy:** Higher rentals on non-working days.
- **Mist:** Higher rentals on non-working days.
- **Light Snow/Rain:** Lower rentals overall, more on non-working days.
- **Heavy Rain/Snow:** Very few rentals regardless of the day type.

Hourly Bike Rental Counts Across Various Seasons

```
# Hourly Bike Rental Counts Across Various Seasons
plt.figure(figsize=(15, 4))
plt.title('Hourly Bike Rental Counts Across Various Seasons')
sns.boxplot(data=df, x='season', y='count', hue='workingday', palette='viridis')
plt.show()
```



💡 Insights:

- **Spring:** Higher rentals on non-working days.
- **Summer:** Increased rentals, especially on non-working days.

- **Fall:** Moderate rentals with a slight increase on non-working days.
- **Winter:** Lower rentals overall, with little difference between working and non-working days.

✚ Relationship between the Dependent and Independent Variables.

```
# correlation
correlation_matrix = df[["atemp", "temp", "humidity", "windspeed", "casual", "registered", "count"]].corr()
correlation_table = pd.DataFrame(correlation_matrix)
correlation_table
```

	atemp	temp	humidity	windspeed	casual	registered	count
atemp	1.000000	0.984948	-0.043536	-0.057473	0.462067	0.314635	0.395062
temp	0.984948	1.000000	-0.064949	-0.017852	0.467097	0.318571	0.399567
humidity	-0.043536	-0.064949	1.000000	-0.318607	-0.348187	-0.265458	-0.323456
windspeed	-0.057473	-0.017852	-0.318607	1.000000	0.092276	0.091052	0.104662
casual	0.462067	0.467097	-0.348187	0.092276	1.000000	0.497250	0.703930
registered	0.314635	0.318571	-0.265458	0.091052	0.497250	1.000000	0.959190
count	0.395062	0.399567	-0.323456	0.104662	0.703930	0.959190	1.000000

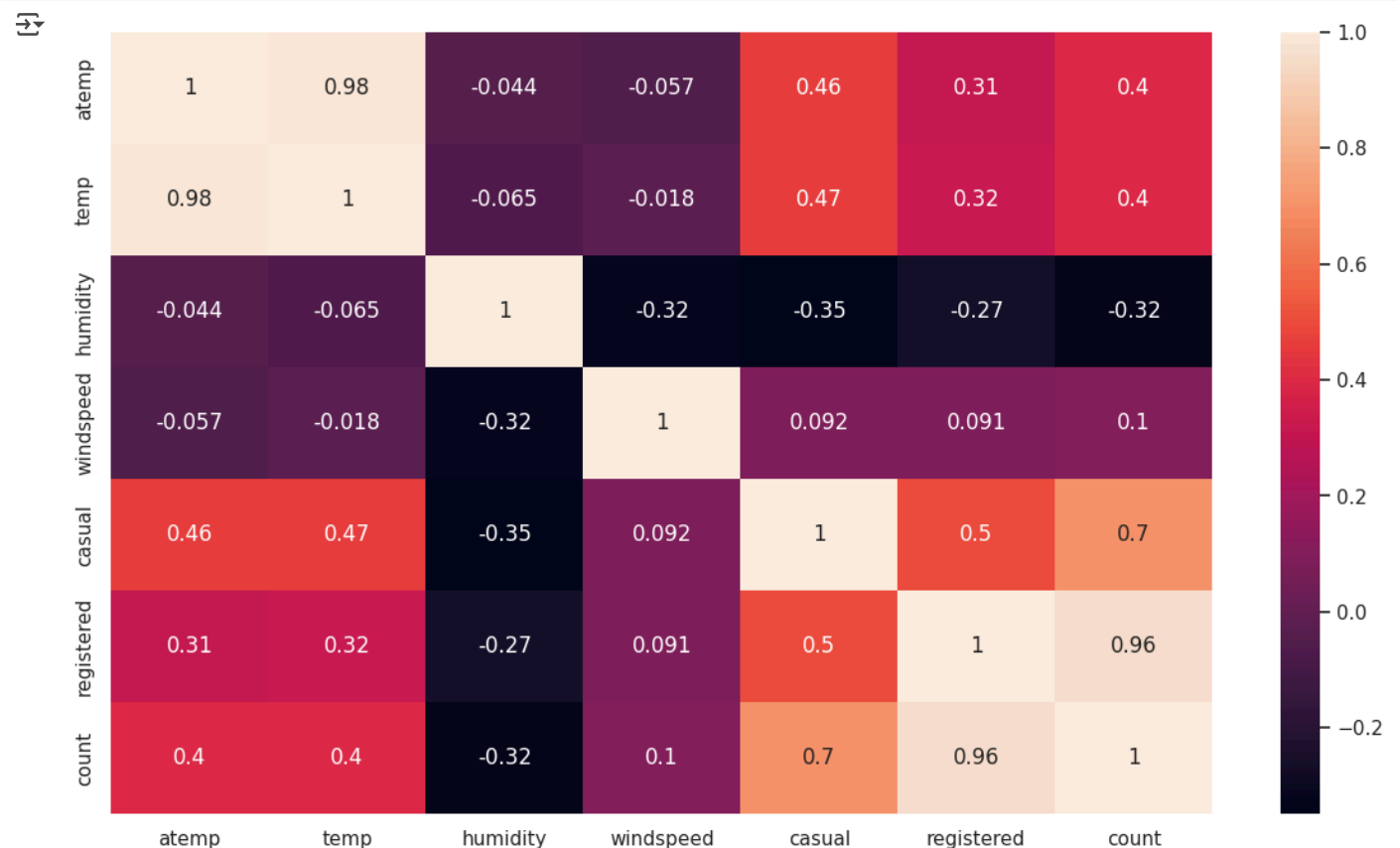
Next steps:

[Generate code with correlation_table](#)

[View recommended plots](#)

[New interactive sheet](#)

```
# correlation plotting
plt.figure(figsize = (14, 8))
sns.heatmap(correlation_matrix, annot = True)
plt.show()
```



💡 Insights:


- **Strong Positive Correlations:**
 - **atemp and temp (0.98):** Indicates that as the apparent temperature (atemp) increases, the actual temperature (temp) increases almost proportionally.

- **registered and count (0.96)**: Suggests that the number of registered users significantly contributes to the total count.
- **casual and count (0.70)**: Indicates that casual users also contribute significantly to the total count.
- **Moderate Positive Correlations:**
 - **casual and registered (0.50)**: Shows some relationship between the number of casual and registered users.
 - **temp and casual (0.47)**: Indicates that higher temperatures might encourage more casual users.
 - **temp and registered (0.32)**: Suggests that higher temperatures might also encourage more registered users.
- **Negative Correlations:**
 - **humidity and casual (-0.35)**: Indicates that higher humidity levels might reduce the number of casual users.
 - **humidity and count (-0.32)**: Suggests that higher humidity levels might reduce the total count.
- **Other Observations:**
 - **humidity and windspeed (-0.32)**: Suggests that higher humidity is associated with lower wind speeds.

✓ Hypothesis Testing

✓ Hypothesis Testing for Bike Rentals on Weekdays vs Weekends


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



count	
workingday	
Yes	7412
No	3474

dtype: int64

```
df.groupby('workingday')['count'].mean()
```



count	
workingday	
No	187.941278
Yes	189.062736

dtype: float64

✓ a) Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

Hypothesis Formulation

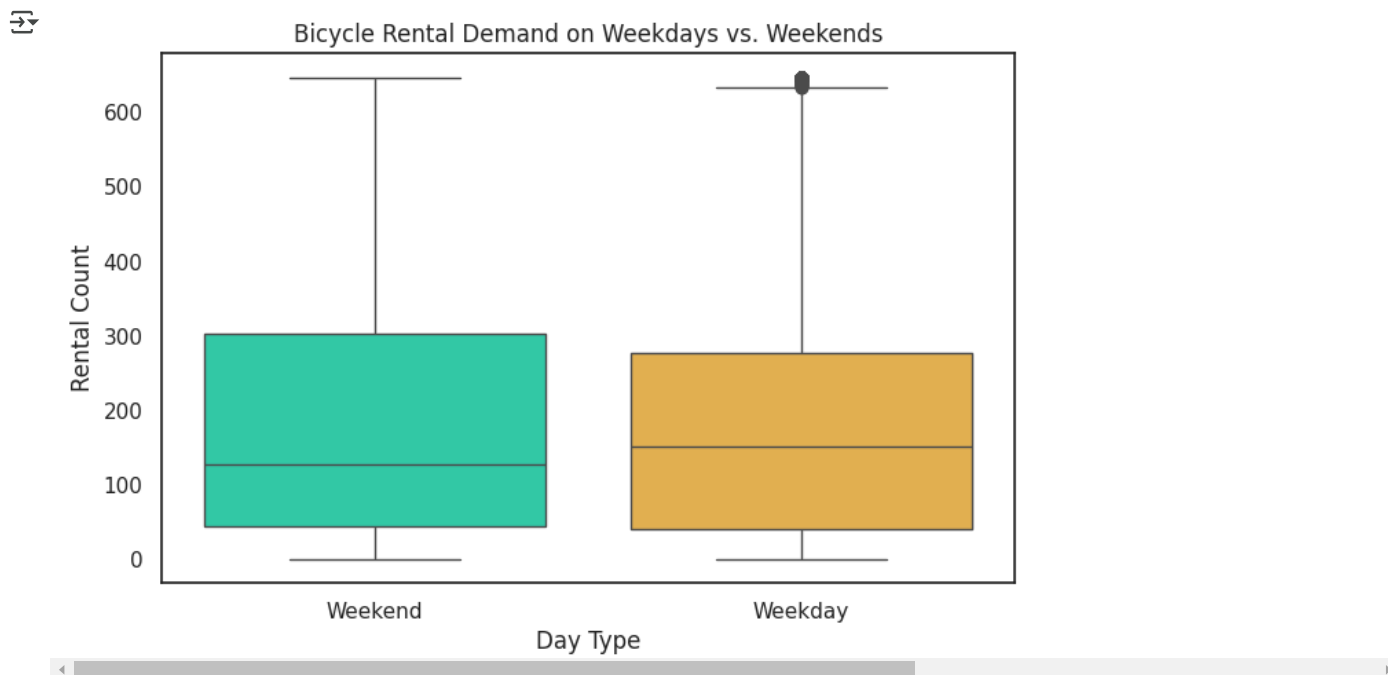
Null Hypothesis (H0): The number of electric cycle rentals is not influenced by whether it is a working day or not.

Alternative Hypothesis (H1): The number of electric cycle rentals is affected by whether it is a working day or not.

Visualizing the Data:

```
# Create 'day_type' column to categorize weekday and weekend
df['day_type'] = df['weekday'].apply(lambda x: 'Weekend' if x in ['Saturday', 'Sunday'] else 'Weekday')
avg_rentals = df.groupby('day_type')['count'].mean().reset_index()

# Box plot to visualize demand on weekdays vs. weekends
plt.figure(figsize=(8, 5))
sns.boxplot(x="day_type", y="count", data=df, palette="turbo")
plt.title("Bicycle Rental Demand on Weekdays vs. Weekends")
plt.xlabel("Day Type")
plt.ylabel("Rental Count")
plt.show()
```

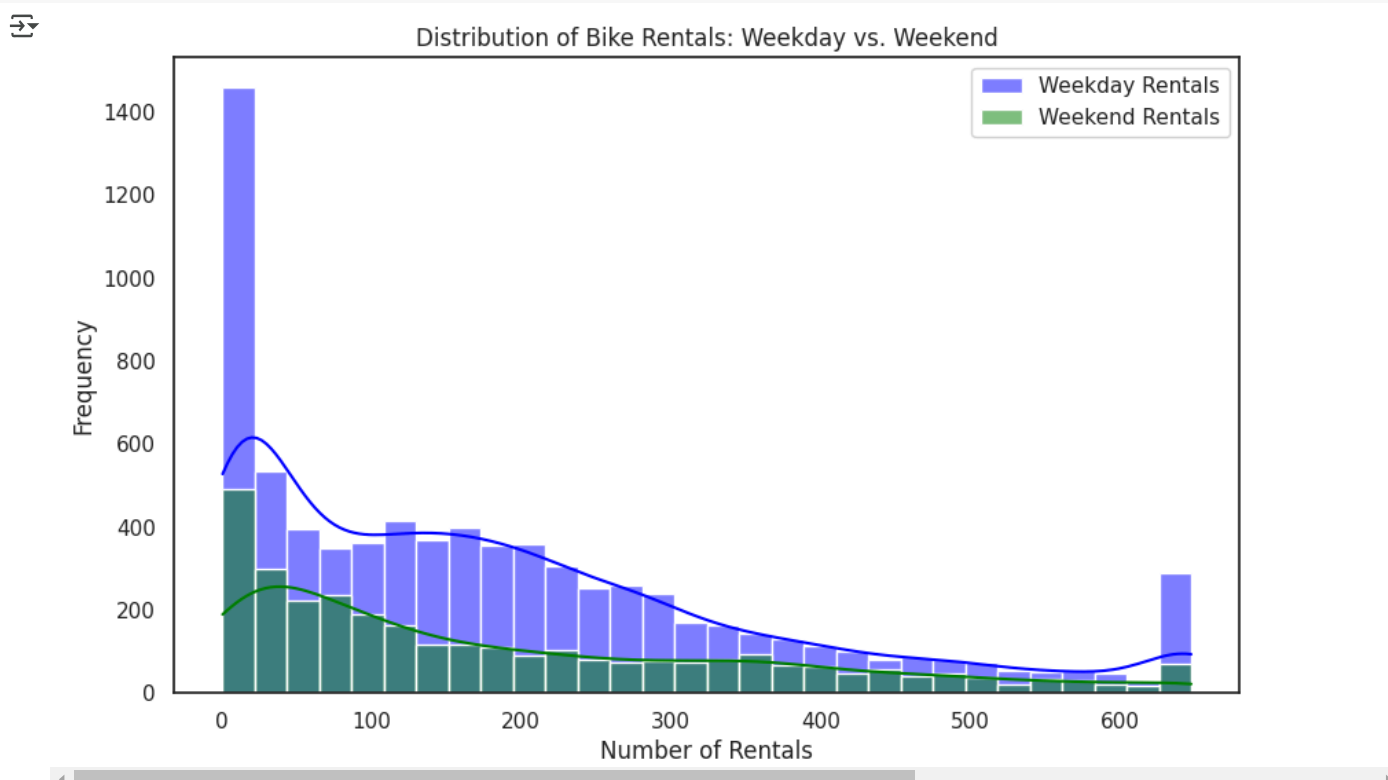


💡 **Insights:**

- **Higher Median on Weekends:** Bicycle rentals are higher on weekends compared to weekdays.
- **Greater Variability on Weekends:** Rental demand shows more variation on weekends.
- **Consistent Weekday Rentals:** Rentals are more consistent and predictable on weekdays.

```
# Define Weekday and Weekend groups
weekday_counts = df[df['weekday'].isin(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'])['count']
weekend_counts = df[df['weekday'].isin(['Saturday', 'Sunday'])['count']

# Plot histograms for weekday and weekend rentals
plt.figure(figsize=(10, 6))
sns.histplot(weekday_counts, label="Weekday Rentals", kde=True, color='blue', bins=30)
sns.histplot(weekend_counts, label="Weekend Rentals", kde=True, color='green', bins=30)
plt.title('Distribution of Bike Rentals: Weekday vs. Weekend')
plt.xlabel('Number of Rentals')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```



✓ b) Performing the Two-Sample Independent T-Test


```
# Since we have two independent groups, so Two Sample Independent T-Test is choosen.
t_stat, p_value = stats.ttest_ind(weekday_counts, weekend_counts)
```

✓ c) Setting the Significance Level for Hypothesis Testing

```
# The Confidence interval be 95%, so significance (alpha) is 0.05
alpha = 0.05
```

✓ d) Calculate test Statistics / p-value


```
#The test statistics and p-value
print(f'T-Test Results:\nT-statistic = {t_stat}, P-value = {p_value}')
```

 T-Test Results:
T-statistic = 0.2056168596222793, P-value = 0.8370940523093207

✓ e) Decision on Null Hypothesis: Accept or Reject Based on P-Value

```
# Decision based on p-value

if p_value <= alpha:
    conclusion = "Reject the Null Hypothesis (H0): There is a significant difference between the number of bike rides on weekdays a
else:
    conclusion = "Do not reject the Null Hypothesis (H0): There is no significant difference between the number of bike rides on we
print(conclusion)
```

 Do not reject the Null Hypothesis (H0): There is no significant difference between the number of bike rides on weekdays and wee

✓ e) Inferences & Conclusions from the Analysis

Inferences:

- The analysis suggests that there is no statistically significant difference in the number of bike rides between weekdays and weekends.
- The rental patterns are similar for both weekdays and weekends, indicating consistent behavior across different days of the week.

Conclusions:

- The Null Hypothesis (H0) is accepted, indicating that any observed difference in bike rentals between weekdays and weekends is not significant enough to warrant a different strategy.

Recommendations:

- **Uniform Strategy:** Use the same strategy for both weekdays and weekends.
- **Consistent Promotions:** Apply consistent promotions throughout the week.
- **Resource Planning:** Plan resources and staff uniformly across the week.

✓ Analyzing the Impact of Weather Conditions on Bicycle Rental Demand

✓ a) Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

Hypotheses Formulation:

- **Null Hypothesis (H0):** The demand for bicycles on rent is the same across different weather conditions.
- **Alternate Hypothesis (H1):** The demand for bicycles on rent is different across different weather conditions.

✓ b) Selection of Appropriate Test

Test Selection:

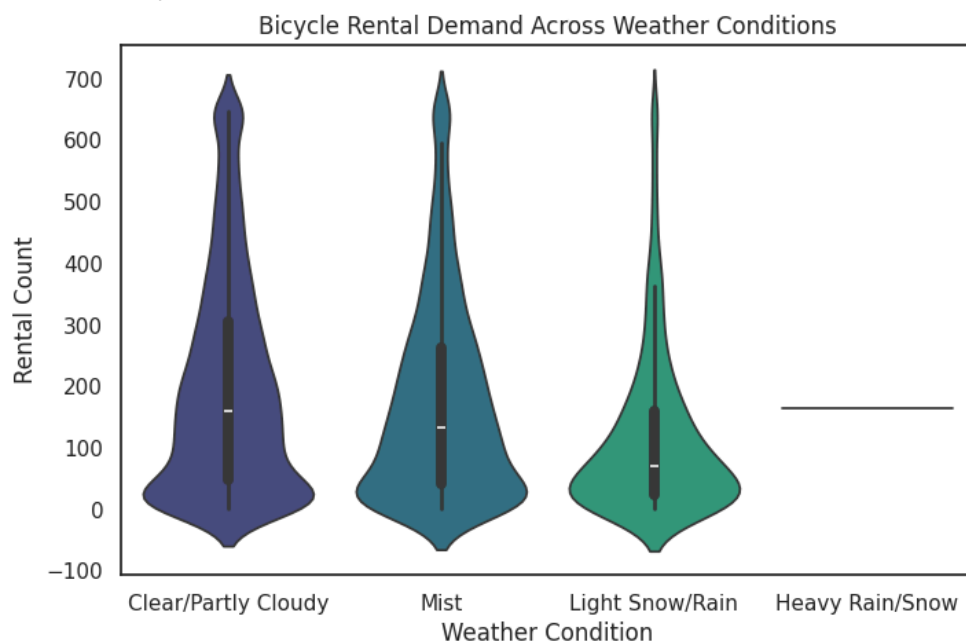
Since we have more than two categories now, we will use **ANOVA** here

```
#Data preparation for anova
# Separate the data by weather condition
weather_conditions = df['weather'].unique()
rental_counts_by_weather = [df[df['weather'] == condition]['count'] for condition in weather_conditions]

# Print counts of each weather condition
print(df['weather'].value_counts())

# Violin plot to visualize demand across weather conditions
plt.figure(figsize=(8, 5))
sns.violinplot(x="weather", y="count", data=df, palette="viridis")
plt.title("Bicycle Rental Demand Across Weather Conditions")
plt.xlabel("Weather Condition")
plt.ylabel("Rental Count")
plt.show()
```

```
weather
Clear/Partly Cloudy    7192
Mist                   2834
Light Snow/Rain        859
Heavy Rain/Snow         1
Name: count, dtype: int64
```



💡 Insights:

- **Highest Rental Demand:** Bicycle rentals are highest under "Clear/Partly Cloudy" conditions, indicating that people prefer renting bikes when the weather is good.
- **Lowest Rental Demand:** Rentals are lowest under "Heavy Rain/Snow" conditions, suggesting adverse weather significantly reduces the demand for bicycle rentals.
- **Moderate Demand:** "Mist" and "Light Snow/Rain" conditions show moderate rental demand, with a more significant density around the median.

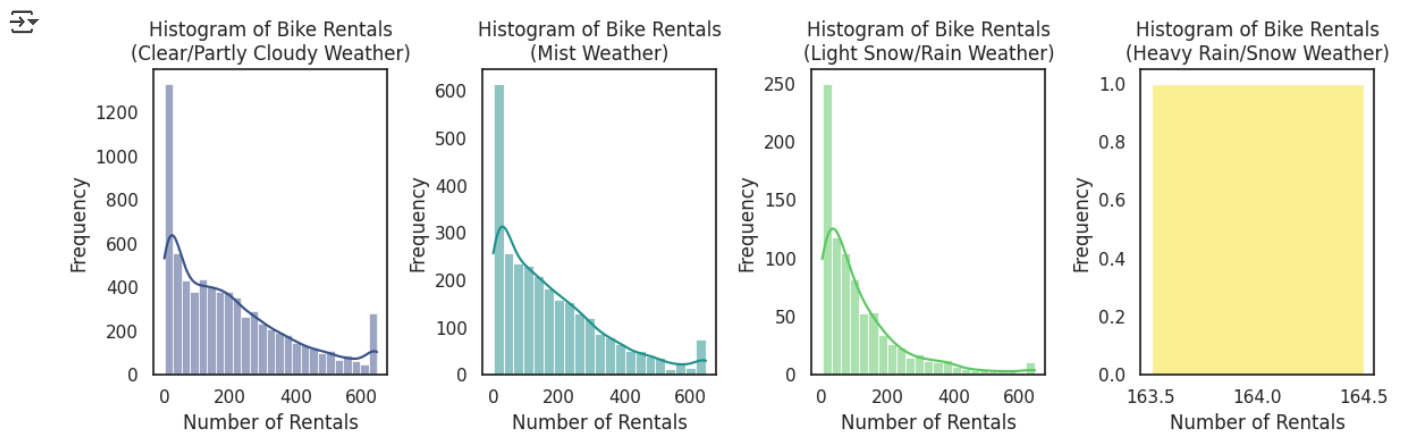
✓ c. Check assumptions of the ANOVA test

i. Check for Normality:

```
# Create subplots to display histograms for each weather condition
plt.figure(figsize=(12, 4))

for i, (condition, rental_counts) in enumerate(zip(weather_conditions, rental_counts_by_weather), 1):
    plt.subplot(1, len(weather_conditions), i)
    color = cm.viridis(i / len(weather_conditions))
    sns.histplot(rental_counts, kde=True, color=color)
    plt.title(f'Histogram of Bike Rentals\n({condition} Weather)')
    plt.xlabel('Number of Rentals')
    plt.ylabel('Frequency')
```

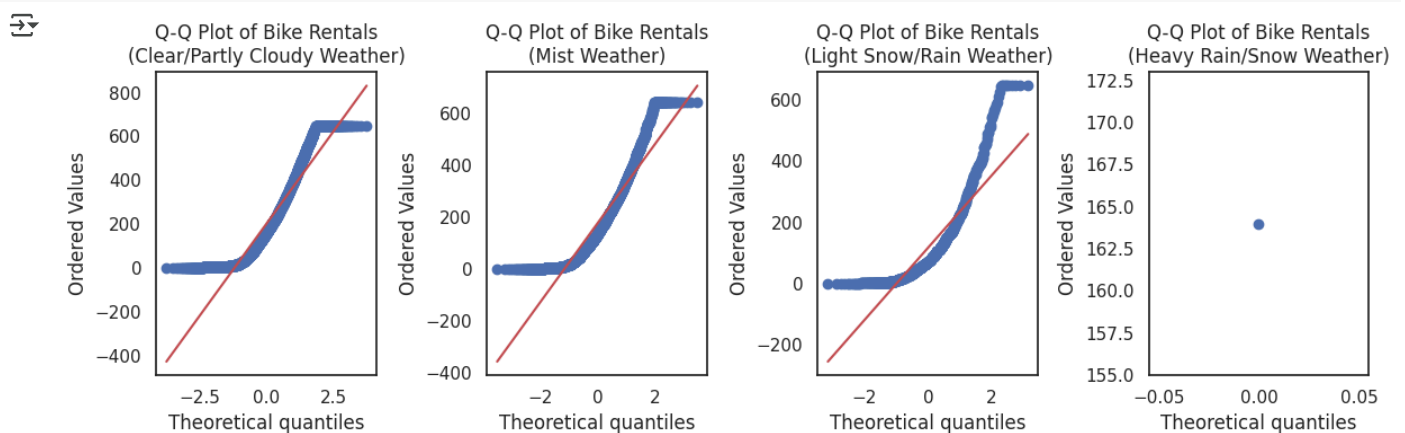
```
plt.tight_layout()
plt.show()
```



```
# Create subplots to display Q-Q plots for each weather condition
plt.figure(figsize=(12, 4))

for i, (condition, rental_counts) in enumerate(zip(weather_conditions, rental_counts_by_weather), 1):
    plt.subplot(1, len(weather_conditions), i)
    stats.probplot(rental_counts, dist="norm", plot=plt)
    plt.title(f'Q-Q Plot of Bike Rentals\n({condition} Weather)')

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



Skewness, Kurtosis, and Shapiro-Wilk's test

```
# Check skewness and kurtosis
for condition in weather_conditions:
    data = df[df['weather'] == condition]['count']
    print(f"Weather: {condition}")
    print(f"Skewness: {stats.skew(data)}")
    print(f"Kurtosis: {stats.kurtosis(data)}\n")
```

```
Weather: Clear/Partly Cloudy
Skewness: 0.8895324985274121
Kurtosis: -0.07753836444599749
```

```
Weather: Mist
Skewness: 1.0860241450099524
Kurtosis: 0.5878010604066195
```

Weather: Light Snow/Rain
Skewness: 1.863255584487065
Kurtosis: 3.6090968062204993

Weather: Heavy Rain/Snow
Skewness: nan
Kurtosis: nan

```
# Perform Shapiro-Wilk Test for Normality
for condition in weather_conditions:
    data = df[df['weather'] == condition]['count']

    # Check if the number of data points is greater than or equal to 3
    if len(data) >= 3:
        stat, p_value = stats.shapiro(data)
        print(f"Shapiro-Wilk Test for Weather {condition} - p-value: {p_value}")
        if p_value <= 0.05:
            print(f>Note: {condition} failed normality assumption (Shapiro-Wilk Test p-value <= 0.05)")
        else:
            print(f"{condition} appears to be normally distributed.")
    else:
        print(f>Note: {condition} has fewer than 3 observations, cannot perform Shapiro-Wilk test.")
```

➦ Shapiro-Wilk Test for Weather Clear/Partly Cloudy - p-value: 3.723807000853401e-56
Note: Clear/Partly Cloudy failed normality assumption (Shapiro-Wilk Test p-value <= 0.05)
Shapiro-Wilk Test for Weather Mist - p-value: 1.7719335162413008e-41
Note: Mist failed normality assumption (Shapiro-Wilk Test p-value <= 0.05)
Shapiro-Wilk Test for Weather Light Snow/Rain - p-value: 6.40167565957445e-32
Note: Light Snow/Rain failed normality assumption (Shapiro-Wilk Test p-value <= 0.05)
Note: Heavy Rain/Snow has fewer than 3 observations, cannot perform Shapiro-Wilk test.

ii. Equality of Variance (Levene's Test)

```
# Perform Levene's Test for Equality of Variance
stat, p_value = stats.levene(*rental_counts_by_weather)
print(f"Levene's Test p-value: {p_value}")
if p_value <= 0.05:
    print(f>Note: Variances are unequal (Levene's Test p-value <= 0.05)")
else:
    print(f"Variances appear to be equal across groups.")
```

➦ Levene's Test p-value: 2.499984328437755e-38
Note: Variances are unequal (Levene's Test p-value <= 0.05)

✓ d) Setting a significance level and calculating the test statistics / p-value

```
# Setting the significance level (alpha)
alpha = 0.05

# Perform One-Way ANOVA
f_stat, p_value = stats.f_oneway(*rental_counts_by_weather)

print(f"F-Statistic: {f_stat}")
print(f"P-Value: {p_value}")
```

➦ F-Statistic: 68.4116520342703
P-Value: 8.034967610817961e-44

✓ e) Decision on Null Hypothesis: Accept or Reject Based on P-Value

```
# Decision rule based on p-value and significance level (alpha)
if p_value <= alpha:
    result = "Reject the null hypothesis. There is a significant difference in rental demand across weather conditions."
else:
    result = "Fail to reject the null hypothesis. There is no significant difference in rental demand across weather conditions."

print(result)
```

➦ Reject the null hypothesis. There is a significant difference in rental demand across weather conditions.

✓ f) Inferences & Conclusions from the Analysis

Inferences:

- Weather significantly impacts bicycle rental demand.
- People rent bicycles more on clear days compared to rainy or snowy days.

Conclusions:

- The Null Hypothesis (H0) is rejected.
- This confirms that different weather conditions result in varying rental demands.

Recommendations:

- **Weather-Based Pricing:** Offer discounts on rainy or snowy days to increase rentals.
- **Targeted Marketing:** Promote specific features for different weather conditions, like rain covers.
- **Resource Management:** Adjust the number of bikes and staff based on the weather forecast.
- **Customer Engagement:** Provide weather updates and riding tips to keep customers informed.

✓ 🌤️🌧️❄️ Analyzing Bicycle Rental Demand Across Seasons 🚴

✓ a) Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

Hypotheses Formulation:

- **Null Hypothesis (H0):** The demand for bicycles on rent is the same across different seasons.
- **Alternate Hypothesis (H1):** The demand for bicycles on rent is different across different seasons.

✓ b) Selection of Appropriate Test

Test Selection:

Since we have more than two categories now, we will use **ANOVA** here

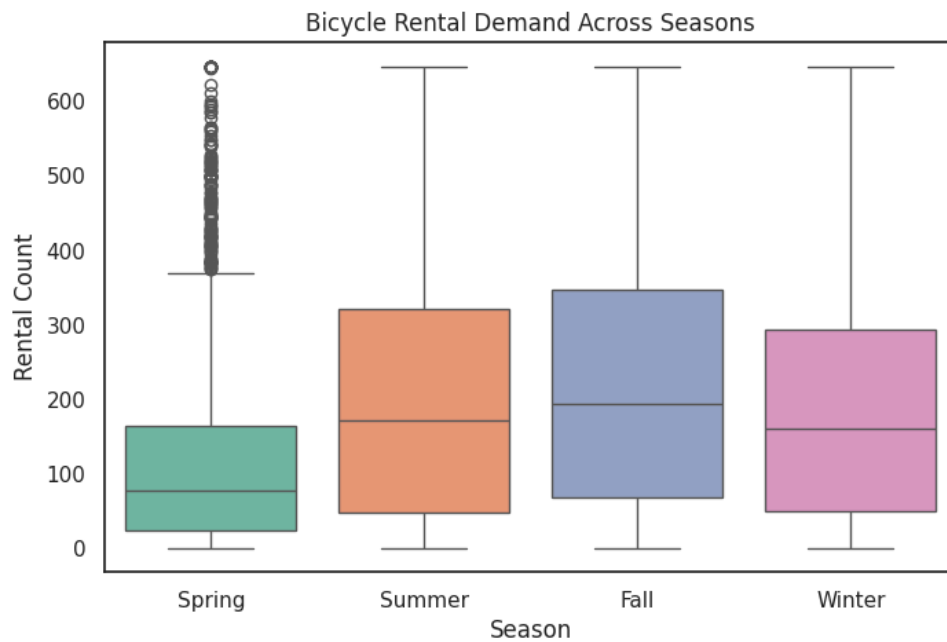
```
#Data preparation for anova
# Print counts of each season
print(df['season'].value_counts())

# Boxplot to visualize demand across seasons
plt.figure(figsize=(8, 5))
sns.boxplot(x="season", y="count", data=df, palette="Set2")
plt.title("Bicycle Rental Demand Across Seasons")
plt.xlabel("Season")
plt.ylabel("Rental Count")
plt.show()
```

```

season
Winter    2734
Summer    2733
Fall      2733
Spring    2686
Name: count, dtype: int64

```



💡 Insights:

- **Higher Demand in Summer and Fall:** Bicycle rentals are highest in Summer and Fall.
- **Lower Demand in Spring:** Rentals are lowest in Spring.
- **Moderate Demand in Winter:** Winter has higher demand than Spring but lower than Summer and Fall.

✓ c. Check assumptions of the ANOVA test

i. Check for Normality:

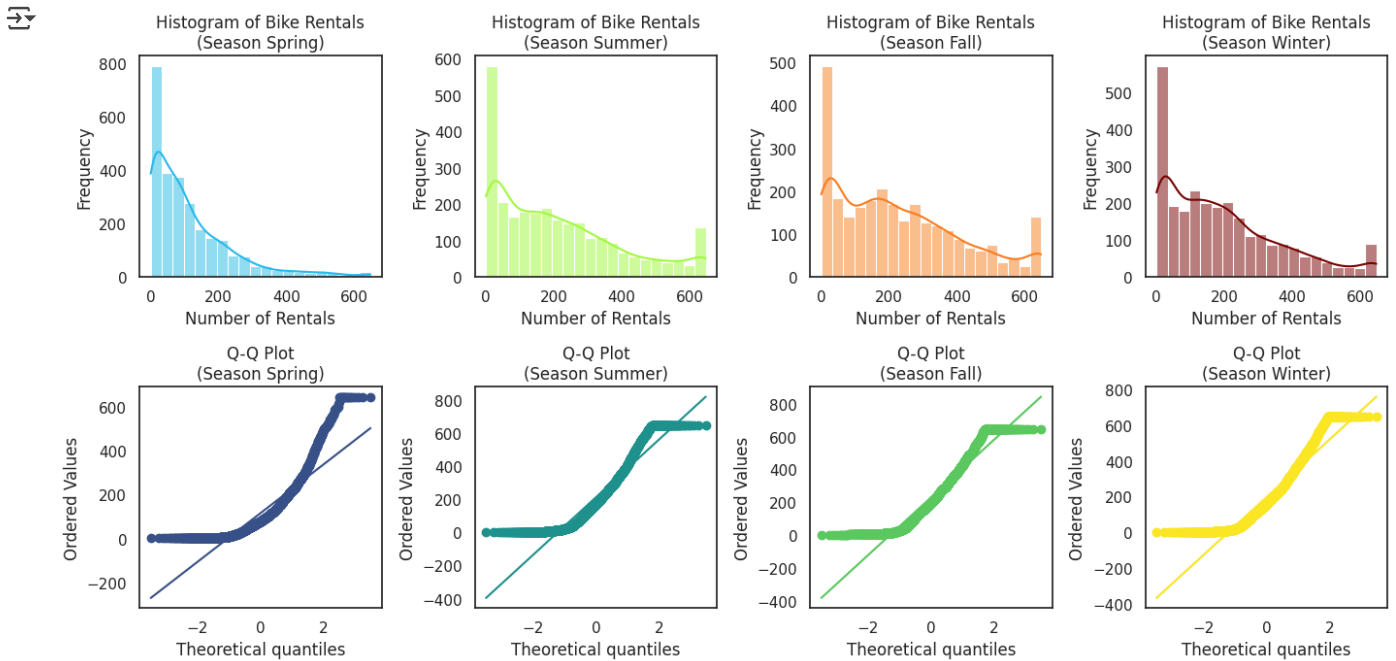
```

plt.figure(figsize=(14, 7))
# Loop through each unique season, using the categorical values directly
for i, season in enumerate(df["season"].cat.categories, 1):
    rental_counts = df[df["season"] == season]["count"]

    # histogram for each season
    plt.subplot(2, len(df["season"].cat.categories), i)
    sns.histplot(rental_counts, kde=True, color=cm.turbo(i / len(df["season"].cat.categories)), bins=20)
    plt.title(f'Histogram of Bike Rentals\n(Season {season})')
    plt.xlabel('Number of Rentals')
    plt.ylabel('Frequency')

    # Create the Q-Q plot for each season
    plt.subplot(2, len(df["season"].cat.categories), len(df["season"].cat.categories) + i)
    res = stats.probplot(rental_counts, dist="norm", plot=plt)
    plt.gca().get_lines()[0].set_color(cm.viridis(i / len(df["season"].cat.categories)))
    plt.gca().get_lines()[1].set_color(cm.viridis(i / len(df["season"].cat.categories)))
    plt.title(f'Q-Q Plot\n(Season {season})')
plt.tight_layout()
plt.show()

```

Skewness, Kurtosis, and Shapiro-Wilk's test

```
# Skewness & Kurtosis
for season in df["season"].unique():
    season_data = df[df["season"] == season]["count"]
    skewness = stats.skew(season_data)
    kurtosis = stats.kurtosis(season_data)
    print(f"Season {season} - Skewness: {skewness:.2f}, Kurtosis: {kurtosis:.2f}")
```

```
Season Spring - Skewness: 1.78, Kurtosis: 3.52
Season Summer - Skewness: 0.82, Kurtosis: -0.25
Season Fall - Skewness: 0.66, Kurtosis: -0.46
Season Winter - Skewness: 0.91, Kurtosis: 0.12
```

```
# Shapiro-Wilk Test for Normality
seasons = list(df["season"].cat.categories)
for season in seasons:
    rental_counts_season = df[df["season"] == season]["count"]

    # Check if the group has enough data points
    if len(rental_counts_season) >= 3:
        # Perform the Shapiro-Wilk test for normality
        stat, p_value = stats.shapiro(rental_counts_season)
        print(f'Shapiro-Wilk Test for Season {season}:')
        print(f'Test Statistic: {stat}')
        print(f'P-value: {p_value}')

        # Interpret the result
        if p_value > 0.05:
            print(f"Season {season}: The data appears to follow a normal distribution (Fail to Reject H0).")
        else:
            print(f"Season {season}: The data does not follow a normal distribution (Reject H0).")
    else:
        print(f"Season {season}: Not enough data points (fewer than 3) for Shapiro-Wilk test.")
```

```
Shapiro-Wilk Test for Season Spring:
Test Statistic: 0.9057489057257019
P-value: 3.6766046654389127e-38
Season Spring: The data does not follow a normal distribution (Reject H0).
Shapiro-Wilk Test for Season Summer:
Test Statistic: 0.9057489057257019
P-value: 3.6766046654389127e-38
Season Summer: The data does not follow a normal distribution (Reject H0).
Shapiro-Wilk Test for Season Fall:
```

```
Test Statistic: 0.9057489057257019
P-value: 3.6766046654389127e-38
Season Fall: The data does not follow a normal distribution (Reject H0).
Shapiro-Wilk Test for Season Winter:
Test Statistic: 0.9057489057257019
P-value: 3.6766046654389127e-38
Season Winter: The data does not follow a normal distribution (Reject H0).
```

ii. Equality of Variance (Levene's Test)

```
# Performing Levene's Test for equality of variances across different seasons
stat, p_value = levene(*season_data)
print(f"Levene's Test p-value: {p_value:.4f}")
# Interpret the result based on p-value
if p_value <= 0.05:
    print(f"Note: Variances are unequal (Levene's Test p-value <= 0.05)")
else:
    print(f"Variances appear to be equal across groups.")
```

🔗 Levene's Test p-value: nan
Variances appear to be equal across groups.

✓ d) Setting a significance level and calculating the test statistics / p-value

```
# Setting the significance level (alpha)
alpha = 0.05

# Perform One-Way ANOVA directly using groupby
f_stat, p_value = stats.f_oneway(*(df[df["season"] == season]["count"] for season in seasons))
print(f"F-Statistic: {f_stat}")
print(f"P-Value: {p_value}")
```

🔗 F-Statistic: 243.33766355201303
P-Value: 7.771506553957677e-153

✓ e) Decision on Null Hypothesis: Accept or Reject Based on P-Value

```
if p_value <= alpha:
    result = "Reject the null hypothesis. There is a significant difference in rental demand across seasons."
else:
    result = "Fail to reject the null hypothesis. There is no significant difference in rental demand across seasons."
print(result)
```

🔗 Reject the null hypothesis. There is a significant difference in rental demand across seasons.

f) Inferences & Conclusions from the Analysis

• Inferences:

- There is a **significant difference** in bicycle rental demand across seasons.
- **Weather conditions** and outdoor activities likely drive demand in different seasons, with **Spring and Summer** seeing **higher** demand, and **Winter** seeing **lower** demand.

• Conclusions:

- **Seasonal variations** in demand must be considered for fleet management and marketing strategies.
- Adjusting the **number of bicycles** and promotions based on seasonality can help optimize rentals.

• Recommendations:

- **Increase the number of bicycles** in **Spring and Summer**; consider dynamic pricing to match demand.
- **Reduce the number of bicycles** in **Winter**, offering discounts to maintain steady rentals.
- Implement **season-specific promotions** and improve customer engagement during low-demand periods.

✓ 🌤️🌧️🌸 Analysis of Weather Conditions Across Seasons using Chi-square Test

✓ a) Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

Hypotheses Formulation:

- **H0 (Null Hypothesis):** There is no significant association between weather conditions and seasons.
- **H1 (Alternate Hypothesis):** There is a significant association between weather conditions and seasons.

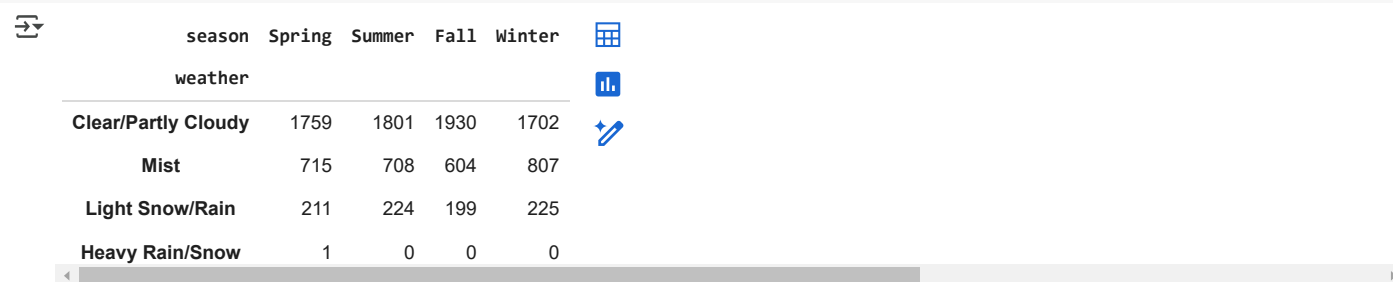
✓ b) Selection of Appropriate Test

Test Selection:

Since we have two categorical variables, the Chi-square test is used to check if there is a significant relationship between the weather conditions and the seasons.

✓ c) Create a Contingency Table for 'Weather' and 'Season'

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



season	Spring	Summer	Fall	Winter
weather				
Clear/Partly Cloudy	1759	1801	1930	1702
Mist	715	708	604	807
Light Snow/Rain	211	224	199	225
Heavy Rain/Snow	1	0	0	0

Next steps: [Generate code with contingency_table](#) [View recommended plots](#) [New interactive sheet](#)

```
chi2_contingency(contingency_table)
```

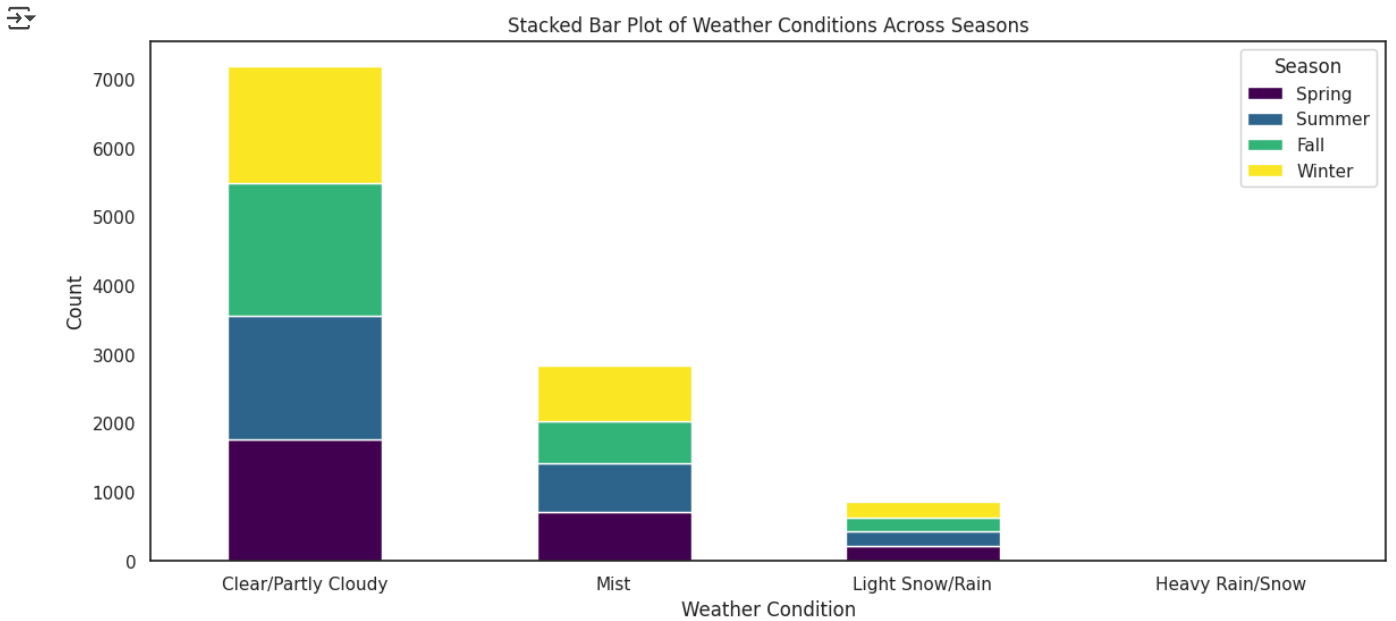
```
Chi2ContingencyResult(statistic=49.15865559689363, pvalue=1.5499250736864862e-07, dof=9, expected_freq=array([[1.77454639e+03,
1.80559765e+03, 1.80559765e+03, 1.80625831e+03],
[6.99258130e+02, 7.11493845e+02, 7.11493845e+02, 7.11754180e+02],
[2.11948742e+02, 2.15657450e+02, 2.15657450e+02, 2.15736359e+02],
[2.46738931e-01, 2.51056403e-01, 2.51056403e-01, 2.51148264e-01]]))
```

💡 Insights:

- Therefore, it can be concluded that **Season** and **Weather** are **dependent** on each other.

📊 Visualizing the Data:

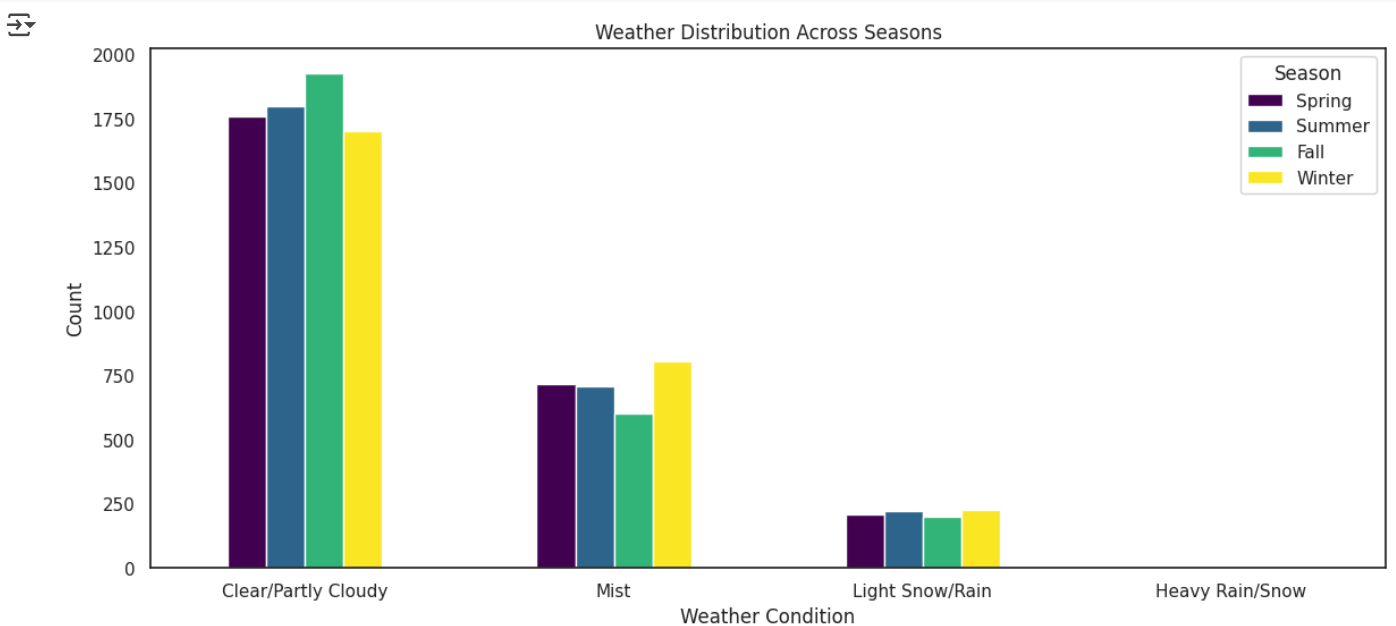
```
# Stacked Bar Plot of the Contingency Table
contingency_table.plot(kind='bar', stacked=True, colormap='viridis', figsize=(12, 7))
plt.title('Stacked Bar Plot of Weather Conditions Across Seasons')
plt.xlabel('Weather Condition')
plt.ylabel('Count')
plt.legend(title='Season')
plt.tight_layout()
plt.xticks(rotation=0)
plt.show()
```



💡 Insights:

- The plot shows that clear or partly cloudy weather is the most common in all seasons, especially in Winter and Fall. Misty weather is the second most common, mainly happening in Winter. Light snow or rain occurs less often, with Winter having the most. Heavy rain or snow is the least common across all seasons.

```
# Weather Distribution Across Seasons
contingency_table.plot(kind='bar', stacked=False, colormap='viridis', figsize=(12, 7))
plt.title('Weather Distribution Across Seasons')
plt.xlabel('Weather Condition')
plt.ylabel('Count')
plt.legend(title='Season')
plt.tight_layout()
plt.xticks(rotation=0)
plt.show()
```



💡 Insights:

- **Clear/Partly Cloudy Weather:**
 - Most common across all seasons.
 - Each season has high counts of around 2000 for this weather condition.
- **Mist:**
 - Second most common weather condition.
 - Winter has the highest count, followed by Spring, Summer, and Fall.
- **Light Snow/Rain:**
 - Less common than Clear/Partly Cloudy and Mist.
 - Winter has the highest count, followed by Spring, Summer, and Fall.
- **Heavy Rain/Snow:**
 - Least common weather condition.
 - Very low counts across all seasons.

✓ d) Setting a significance level and calculating the test statistics / p-value

```
# Setting the significance level (alpha)
alpha = 0.05

# Perform Chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Test Statistic:", chi2)
print("p-value:", p)
```

```
🔄 Chi-square Test Statistic: 49.15865559689363
p-value: 1.5499250736864862e-07
```

✓ e) Decision on Null Hypothesis: Accept or Reject Based on P-Value

```
if p <= alpha:
    print("\nReject the Null Hypothesis (H0). There is a significant relationship between Weather and Season.")
else:
    print("\nFail to Reject the Null Hypothesis (H0). There is no significant relationship between Weather and Season.")
```

```
🔄
Reject the Null Hypothesis (H0). There is a significant relationship between Weather and Season.
```

💡 Insights:

- Since the p-value is much less than the significance level (alpha = 0.05).
- This indicates a significant relationship between weather conditions and seasons.

✓ f) Inferences & Conclusions from the Analysis

- **Inferences: Weather Patterns**
 - **Summer:** Clear/Partly Cloudy conditions are more prevalent.
 - **Winter:** Heavy Rain/Snow conditions are more frequent.
 - **Spring and Fall:** Transitional weather conditions like Mist and Light Rain/Snow are common.
- **Recommendations:**
 - **Seasonal Planning:**
 - Adjust marketing strategies based on prevalent weather conditions in each season.
 - For instance, promote outdoor activities and rentals during summer when clear weather is more common.
 - **Resource Allocation:**
 - Allocate more resources (e.g., bicycles, staff) during seasons with favorable weather conditions to meet higher demand.
 - **Weather-Based Promotions:**

- Offer special promotions or discounts during less favorable weather conditions to encourage rentals.

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

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