Business Case: Yulu - Hypothesis Testing

About Yulu

Yulu is India’s leading micro-mobility service provider, which offers unique vehicles for the daily commute. StarƟng off as a mission to eliminate traffic congesƟon in India, Yulu provides the safest commute soluƟon through a user-friendly mobile app to enable shared, solo and sustainable commuƟng. Yulu zones are located at all the appropriate locaƟons (including metro staƟons, bus stands, office spaces, residenƟal areas, corporate offices, etc) to make those first and last miles smooth, affordable, and convenient! Yulu has recently suffered considerable dips in its revenues. They have contracted a consulƟng company to understand the factors on which the demand for these shared electric cycles depends. Specifically, they want to understand the factors affecƟng the demand for these shared electric cycles in the Indian market.

A) Import the dataset and do usual exploratory data analysis steps like checking the structure & characteristics of the dataset.

In [409…
```
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
```

In [410…
```
pip install scipy
```

```
Requirement already satisfied: scipy in c:\users\vishal\desktop\scaler projects\y
ulu hypothesis testing\yulu-hypothesis\.venv\lib\site-packages (1.15.3)
Requirement already satisfied: numpy<2.5,>=1.23.5 in c:\users\vishal\desktop\scal
er projects\yulu hypothesis testing\yulu-hypothesis\.venv\lib\site-packages (from
scipy) (2.2.6)
Note: you may need to restart the kernel to use updated packages.
```

In [411…
```
df = pd.read_csv("Yulu.csv");
```

In [412…
```
df.head()
```

Out[412…

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

In [413…

```
# no of rows amd columns in dataset
print(f"# rows: {df.shape[0]} \n# columns: {df.shape[1]}")
```

```
# rows: 10886
# columns: 12
```

In [414…

```
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
```

🔄 Converting Data Types of Columns

- `datetime` → `datetime`
- `season` → `category`
- `holiday` → `category`
- `workingday` → `category`
- `weather` → `category`

In [415…

```
df['datetime'] = pd.to_datetime(df['datetime'])
cat_cols= ['season', 'holiday', 'workingday', 'weather']
for col in cat_cols:
    df[col] = df[col].astype('object')
```

In [416…

```
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  datetime64[ns]
 1   season      10886 non-null  object
 2   holiday     10886 non-null  object
 3   workingday  10886 non-null  object
 4   weather     10886 non-null  object
 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: datetime64[ns](1), float64(3), int64(4), object(4)
memory usage: 1020.7+ KB
```

In [417…

```
df.iloc[:,1:].describe(include='all')
```

Out[417…

	season	holiday	workingday	weather	temp	atemp	humidit
count	10886.0	10886.0	10886.0	10886.0	10886.00000	10886.000000	10886.00000
unique	4.0	2.0	2.0	4.0	NaN	NaN	Na
top	4.0	0.0	1.0	1.0	NaN	NaN	Na
freq	2734.0	10575.0	7412.0	7192.0	NaN	NaN	Na
mean	NaN	NaN	NaN	NaN	20.23086	23.655084	61.88646
std	NaN	NaN	NaN	NaN	7.79159	8.474601	19.24503
min	NaN	NaN	NaN	NaN	0.82000	0.760000	0.00000
25%	NaN	NaN	NaN	NaN	13.94000	16.665000	47.00000
50%	NaN	NaN	NaN	NaN	20.50000	24.240000	62.00000
75%	NaN	NaN	NaN	NaN	26.24000	31.060000	77.00000
max	NaN	NaN	NaN	NaN	41.00000	45.455000	100.00000

- There are no missing values in the dataset.
- casual and registered attributes might have outliers because their mean and median are very far away to one another and the value of standard deviation is also high which tells us that there is high variance in the data of these attributes.

In [418…

```
df.isnull().sum()
```

Out[418…

```
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
```

In [419…

```
df.season.value_counts()
```

Out[419…

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

In [420…

```
df.weather.value_counts()
```

Out[420…

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

In [421…

```
df.workingday.value_counts()
```

Out[421…

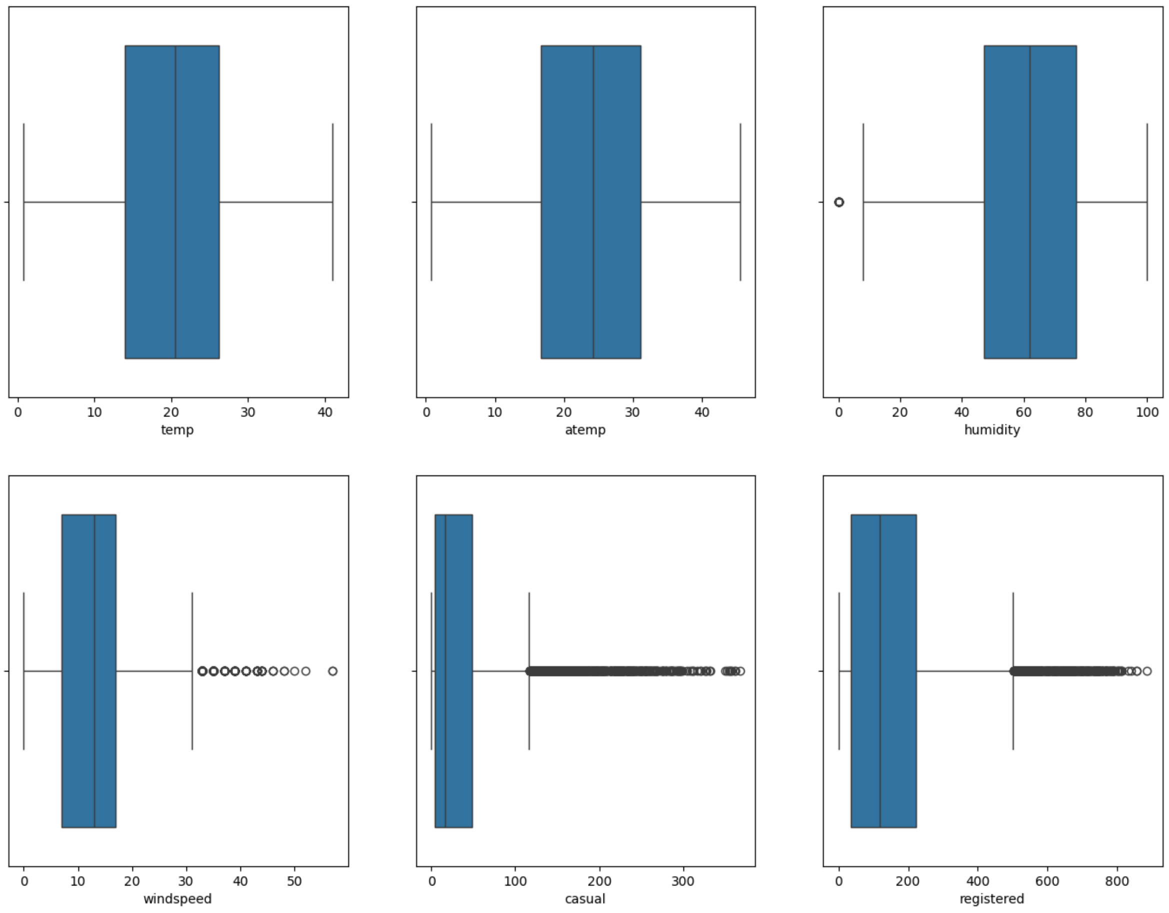
```
workingday
1    7412
0    3474
Name: count, dtype: int64
```

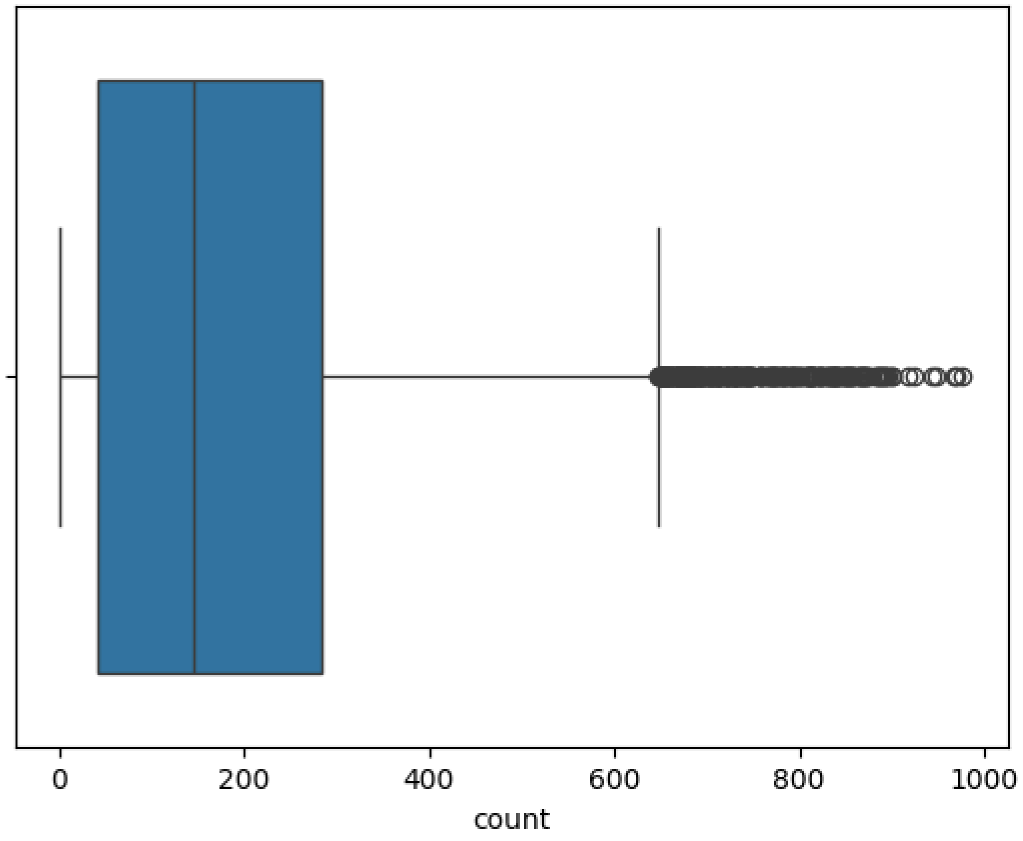
Try establishing a relation between the dependent and independent variable (Dependent “Count” & Independent: Workingday, Weather, Season etc)

Univariate Analysis:

In [422…

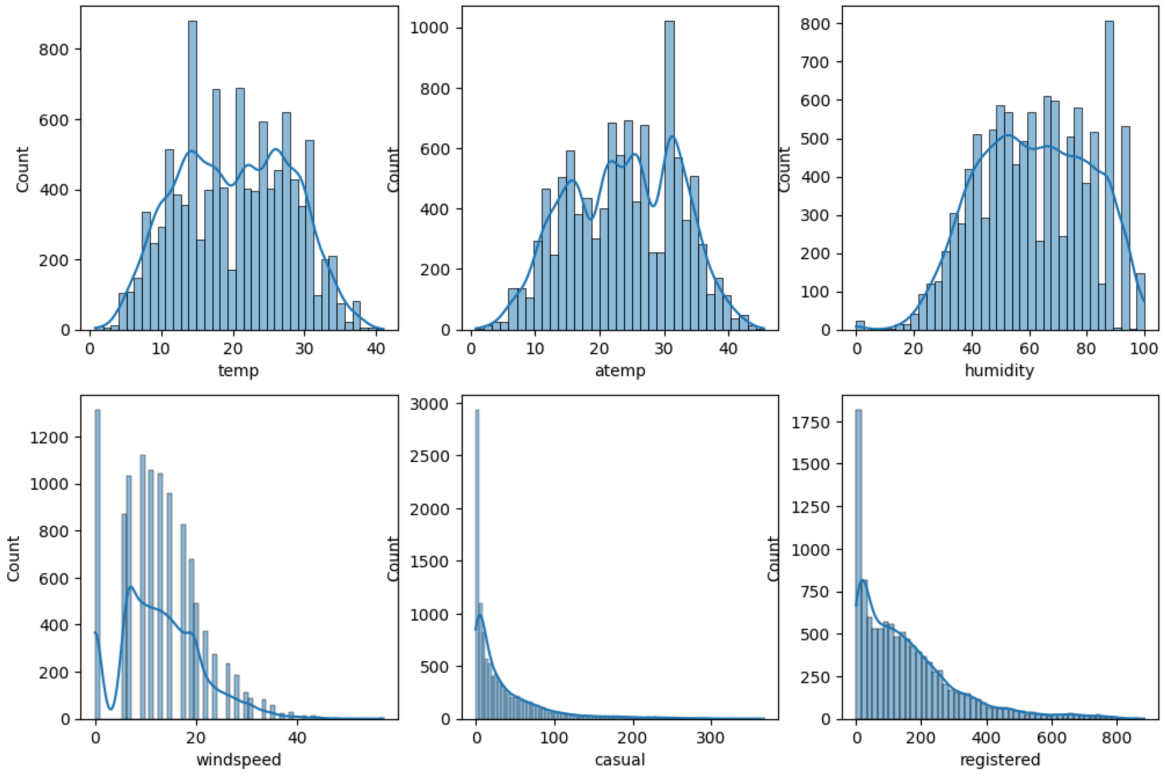
```
# plotting box plots to detect outliers in the data
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))
index = 0
for row in range(2):
    for col in range(3):
        sns.boxplot(x=df[num_cols[index]], ax=axis[row, col])
        index += 1
plt.show()
sns.boxplot(x=df[num_cols[-1]])
plt.show()
```

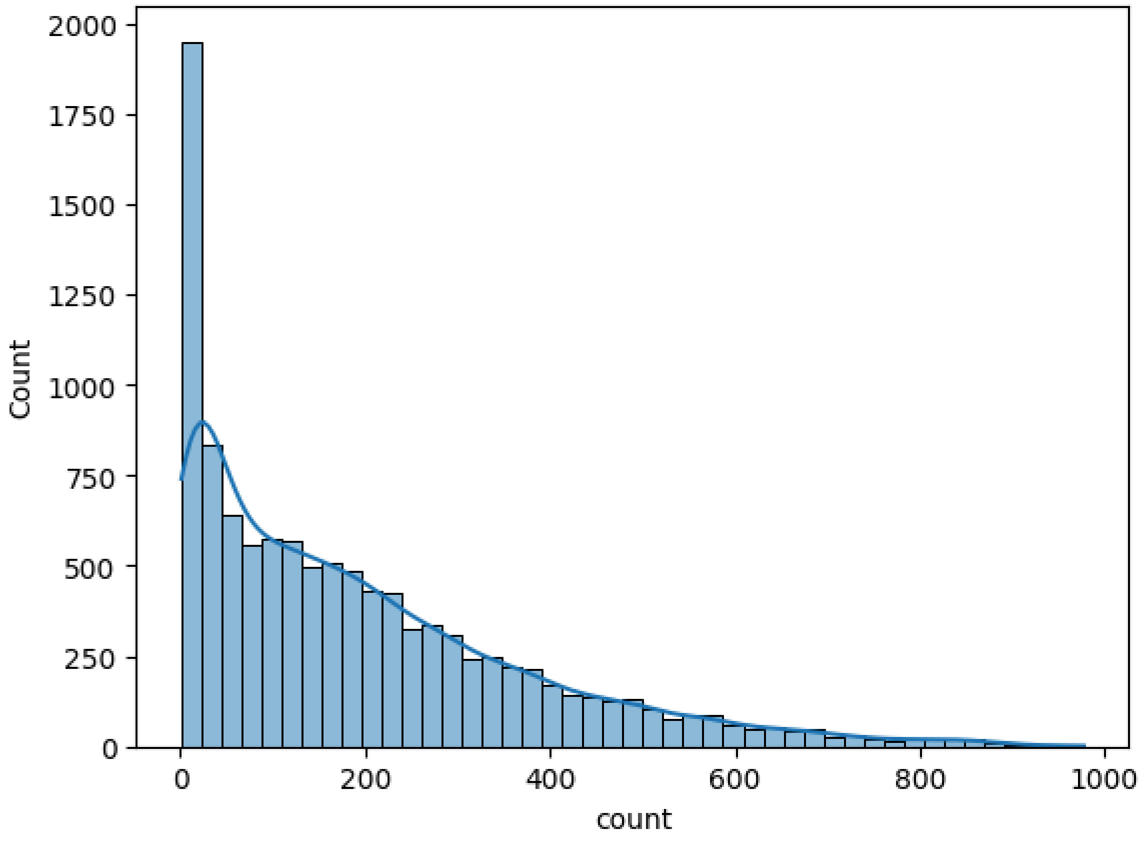




In [423…

```
# understanding the distribution for numerical variables
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual',
'registered','count']
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(12, 8))
index = 0
for row in range(2):
    for col in range(3):
        sns.histplot(df[num_cols[index]], ax=axis[row, col], kde=True)
        index += 1
plt.show()
sns.histplot(df[num_cols[-1]], kde=True)
plt.show()
```



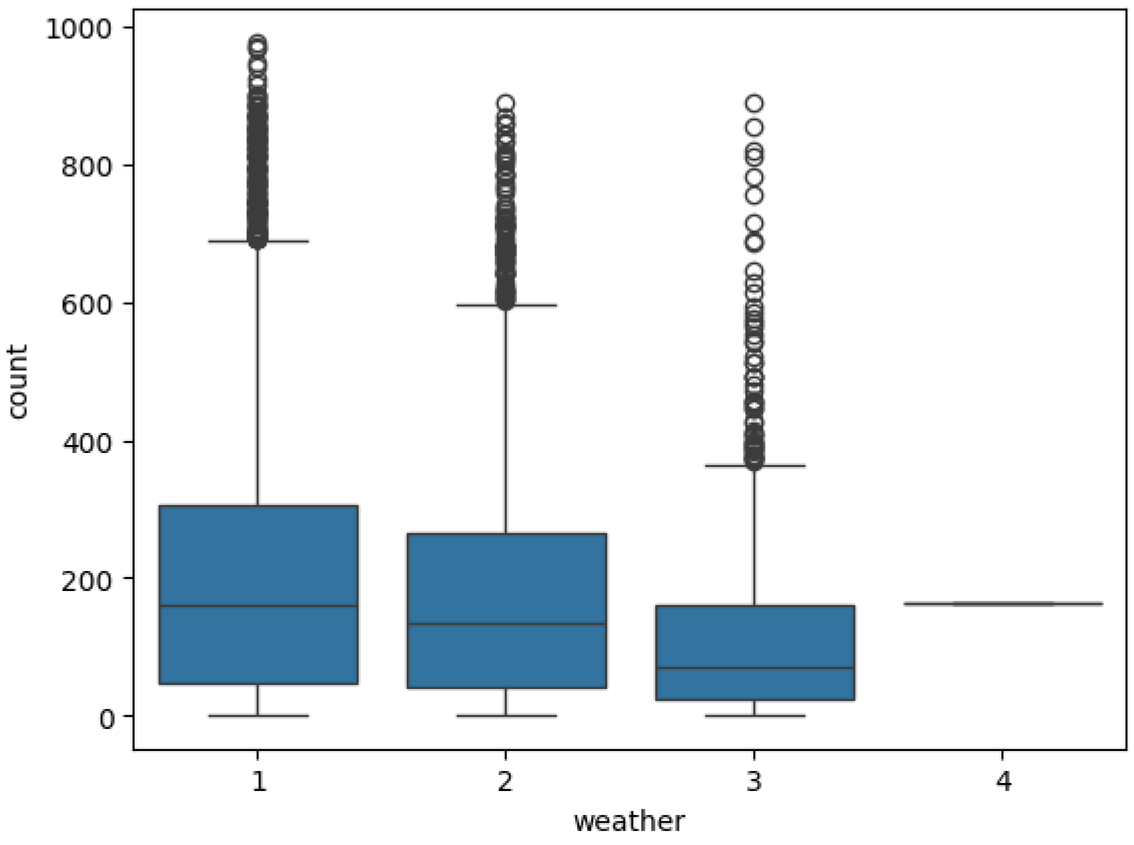


- 'casual', 'registered', and 'count' somewhat look like Log Normal Distribution
- 'temp', 'atemp', and 'humidity' appear to follow a Normal Distribution

- 'windspeed' seems to follow a Binomial Distribution

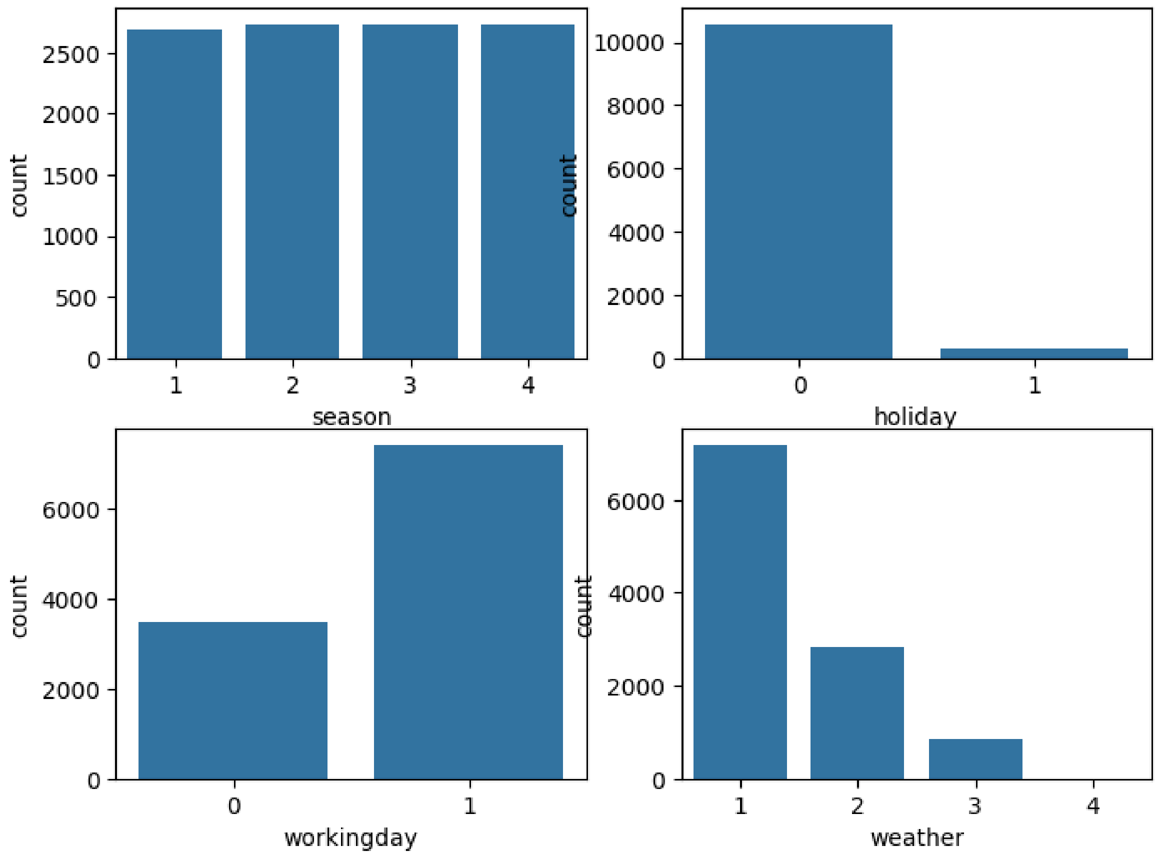
In [424…
```
sns.boxplot(x='weather',y='count', data=df)
```

Out[424…
```
<Axes: xlabel='weather', ylabel='count'>
```



Looks like humidity, casual, registered and count have outliers in the data.

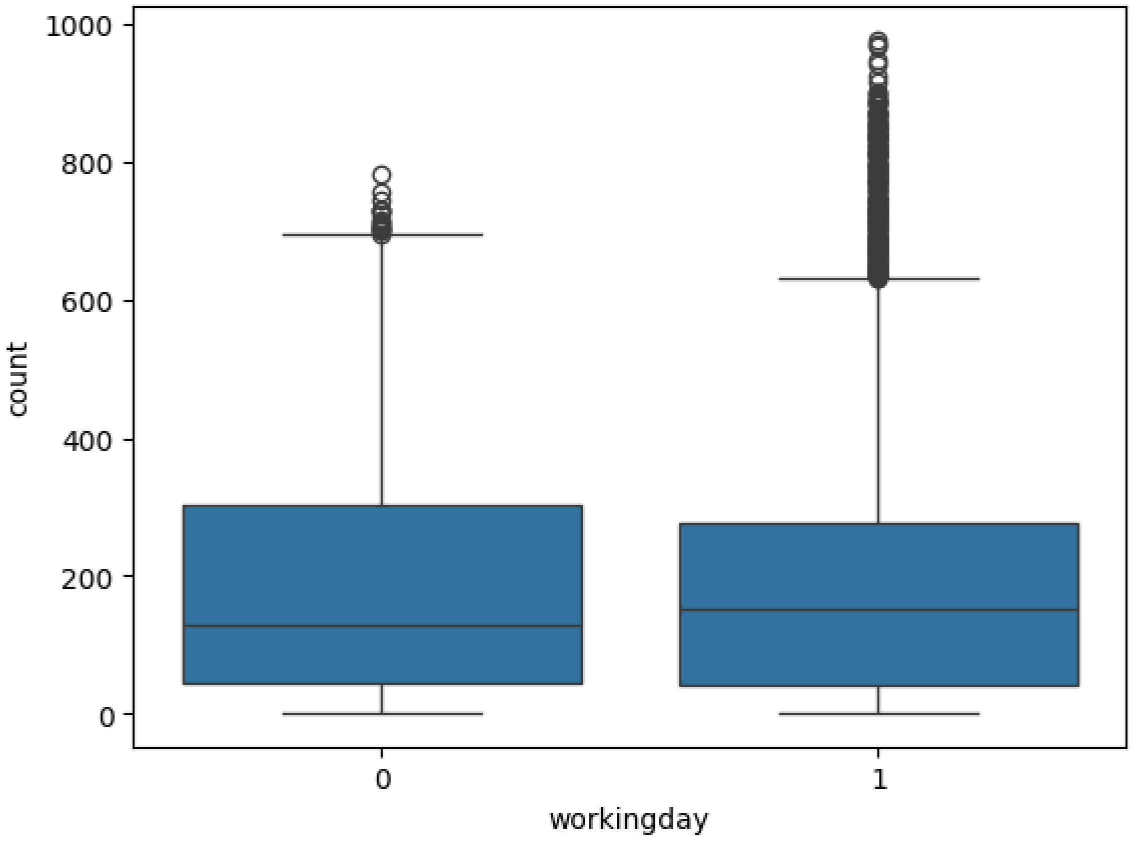
In [425…
```
# countplot of each categorical column
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(8, 6))
index = 0
for row in range(2):
    for col in range(2):
        sns.countplot(data=df, x=cat_cols[index], ax=axis[row, col])
        index += 1
plt.show()
```



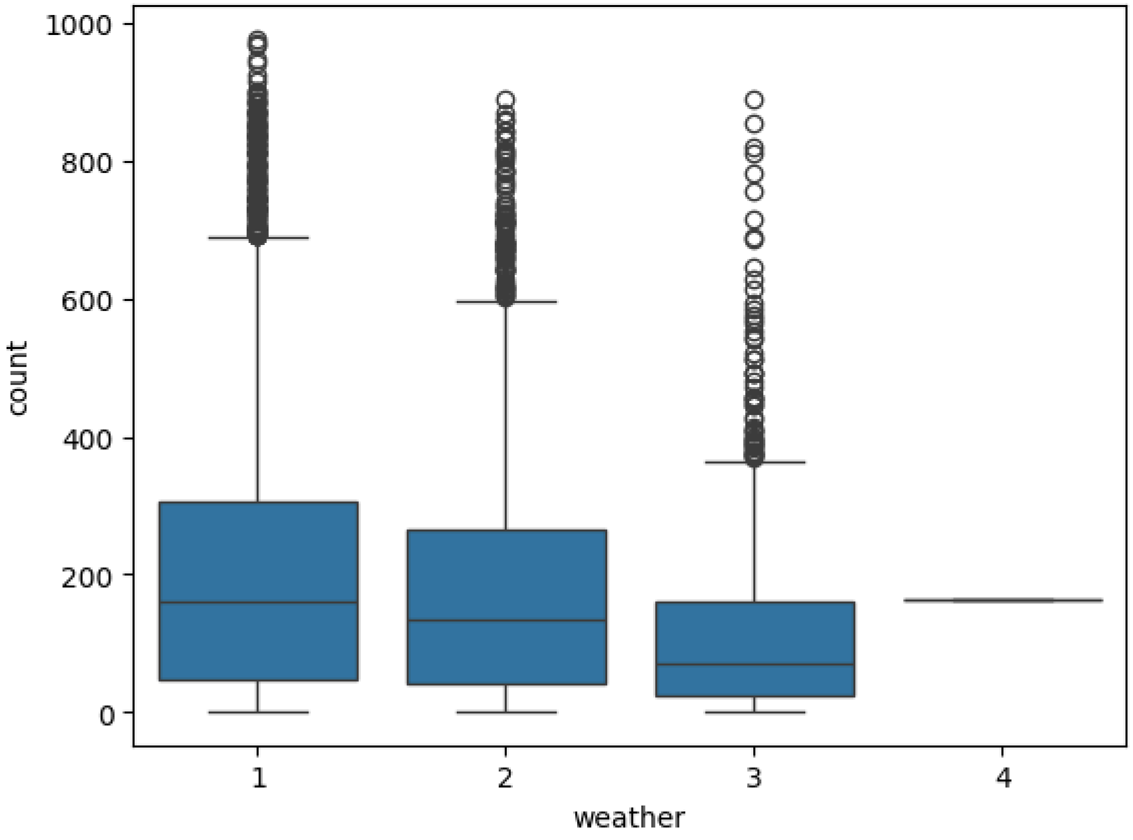
Data looks common as it should be like equal number of days in each season, more working days and weather is mostly Clear, Few clouds, partly cloudy, partly cloudy.

```
In [426…  sns.boxplot(x='workingday',y='count', data=df)
```

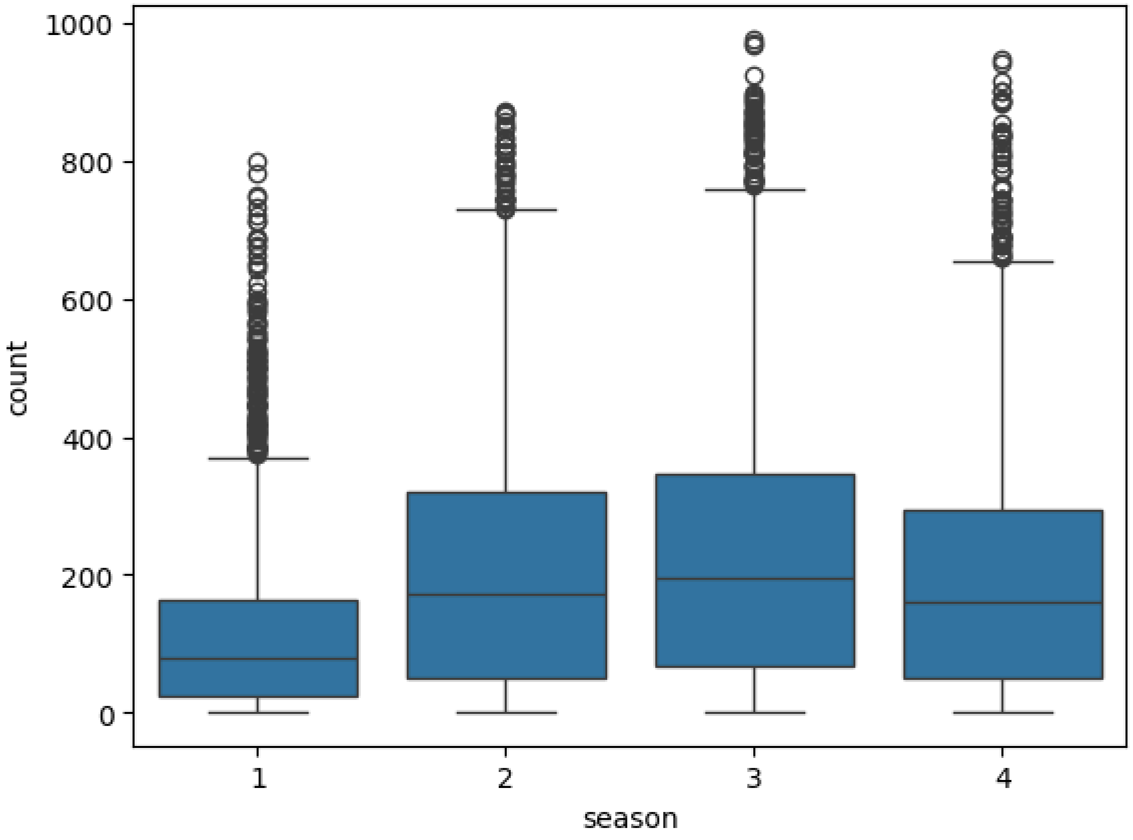
```
Out[426…  <Axes: xlabel='workingday', ylabel='count'>
```



In [427…
```
sns.boxplot(x='weather',y='count', data=df)
```
Out[427…
```
<Axes: xlabel='weather', ylabel='count'>
```



In [428…
```
sns.boxplot(x='season',y='count', data=df)
```
Out[428…
```
<Axes: xlabel='season', ylabel='count'>
```

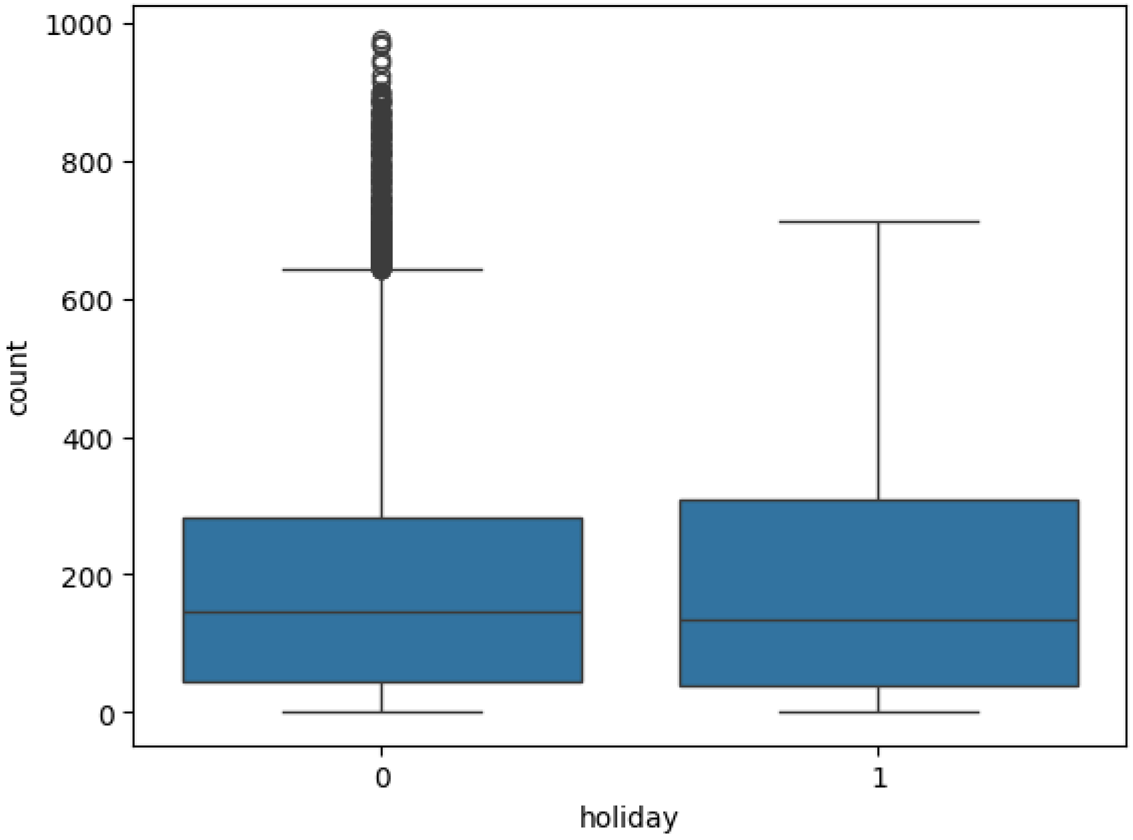


In [429…

```
sns.boxplot(x='holiday',y='count', data=df)
```

Out[429…

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



• In summer and fall seasons, more bikes are rented as compared to other seasons.

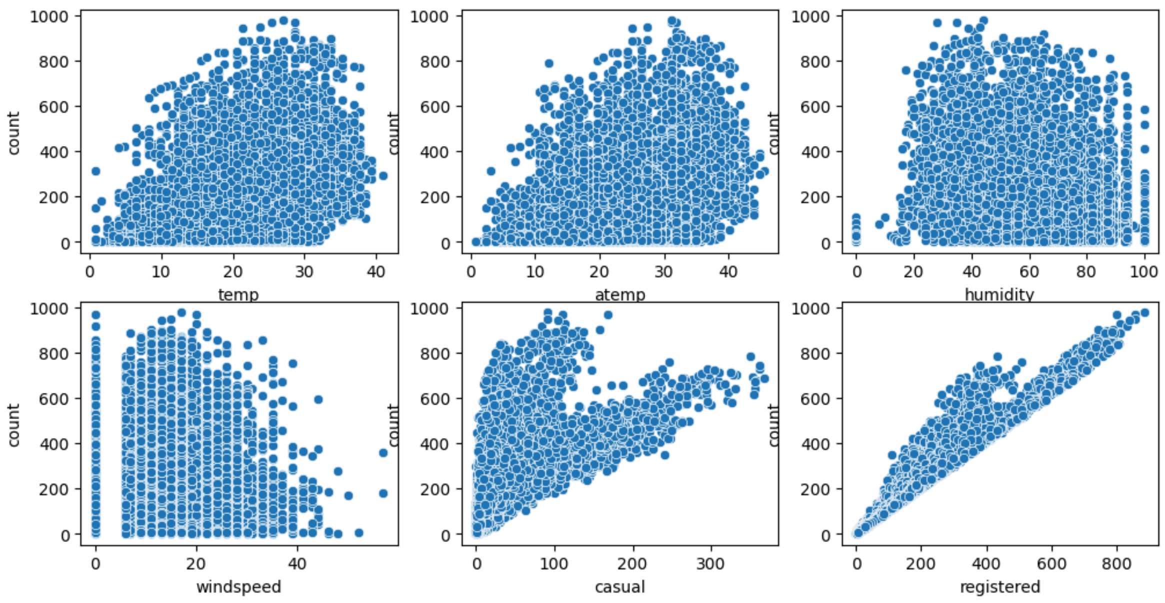
• Whenever it's a holiday, more bikes are rented.

• It is also clear from the workingday column that when the day is a holiday or weekend, slightly more bikes are rented.

• Whenever there is rain, thunderstorm, snow, or fog, fewer bikes are rented.

In [430…

```
# plotting numerical variables againt count using scatterplot
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(12, 6))
index = 0
for row in range(2):
    for col in range(3):
        sns.scatterplot(data=df, x=num_cols[index], y='count',
        ax=axis[row, col])
        index += 1
plt.show()
```



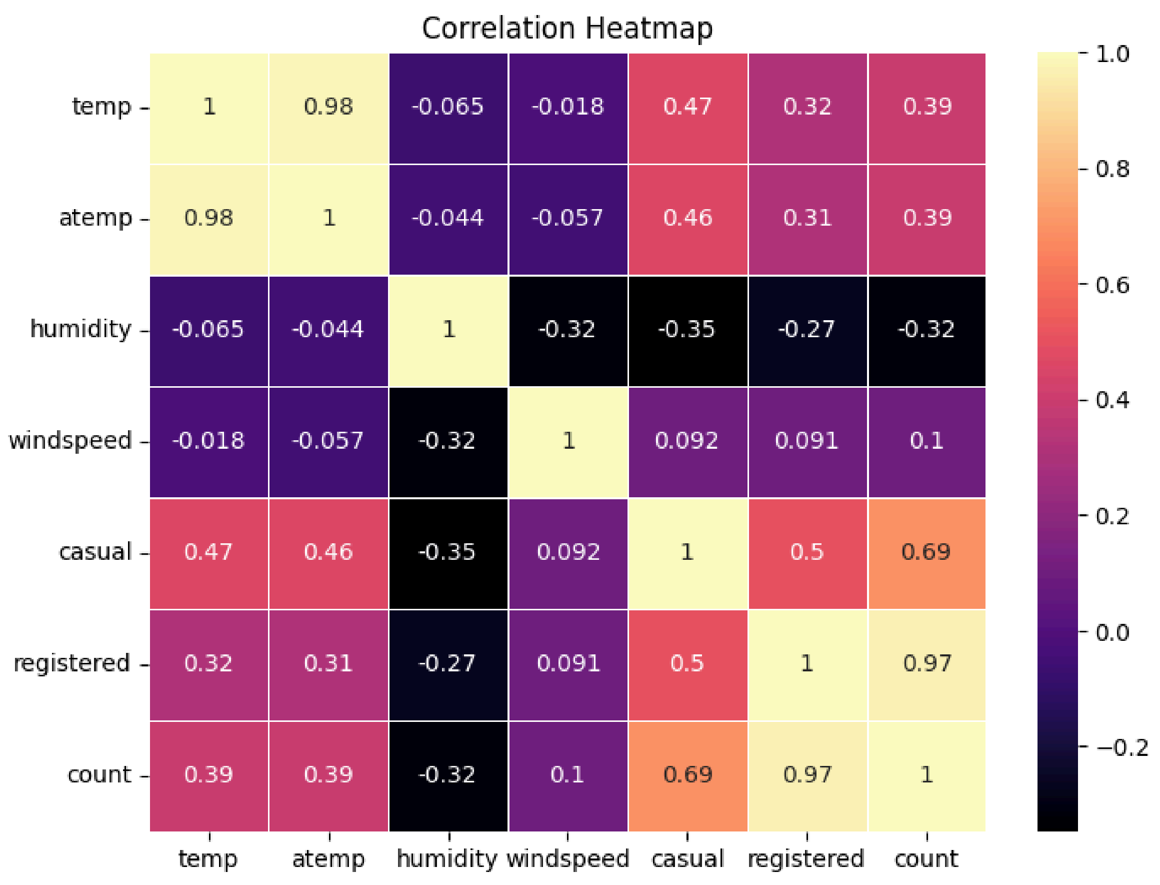
- Whenever the humidity is less than 20, number of bikes rented is very very low.
- Whenever the temperature is less than 10, number of bikes rented is less.
- Whenever the windspeed is greater than 35, number of bikes rented is less.

In [431…

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Select only relevant numerical columns
numerical_features = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'regis

# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(df[numerical_features].corr(), annot=True, cmap='magma', linewidths=
plt.title("Correlation Heatmap")
plt.show()
```



	temp	atemp	humidity	windspeed	casual	registered	count
temp	1	0.98	-0.065	-0.018	0.47	0.32	0.39
atemp	0.98	1	-0.044	-0.057	0.46	0.31	0.39
humidity	-0.065	-0.044	1	-0.32	-0.35	-0.27	-0.32
windspeed	-0.018	-0.057	-0.32	1	0.092	0.091	0.1
casual	0.47	0.46	-0.35	0.092	1	0.5	0.69
registered	0.32	0.31	-0.27	0.091	0.5	1	0.97
count	0.39	0.39	-0.32	0.1	0.69	0.97	1

2: Hypothesis Testing

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

Null Hypothesis (H_0): Weather is independent of the season
Alternate Hypothesis (H_1): Weather is not independent of the season
Significance Level (α): 0.05

In [432…

```
data_table = pd.crosstab(df['season'], df['weather'])
print("Observed values:")
data_table
```

```
Observed values:
```

Out[432…

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

In [433…

```
val = stats.chi2_contingency(data_table)
print(val)
expected_values = val[3]
print(expected_values)
```

```
nrows, ncols = 4, 4
dof = (nrows-1)*(ncols-1)
print("degrees of freedom: ", dof)
alpha = 0.05

chi_sqr = sum([(o-e)**2/e for o, e in zip(data_table.values,
expected_values)])
chi_sqr_statistic = chi_sqr[0] + chi_sqr[1]
print("chi-square test statistic: ", chi_sqr_statistic)
critical_val = stats.chi2.ppf(q=1-alpha, df=dof)
print(f"critical value: {critical_val}")
p_val = 1-stats.chi2.cdf(x=chi_sqr_statistic, df=dof)
print(f"p-value: {p_val}")
if p_val <= alpha:
    print("\nSince p-value is less than the alpha 0.05,\nWe reject the Null Hypo
else:
    print("Since p-value is greater than the alpha 0.05, We do not reject the Nu
```

```
Chi2ContingencyResult(statistic=np.float64(49.158655596893624), pvalue=np.float64
(1.549925073686492e-07), dof=9, expected_freq=array([[1.77454639e+03, 6.99258130e
+02, 2.11948742e+02, 2.46738931e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80625831e+03, 7.11754180e+02, 2.15736359e+02, 2.51148264e-01]]))
[[1.77454639e+03 6.99258130e+02 2.11948742e+02 2.46738931e-01]
 [1.80559765e+03 7.11493845e+02 2.15657450e+02 2.51056403e-01]
 [1.80559765e+03 7.11493845e+02 2.15657450e+02 2.51056403e-01]
 [1.80625831e+03 7.11754180e+02 2.15736359e+02 2.51148264e-01]]
degrees of freedom:  9
chi-square test statistic:  44.09441248632364
critical value: 16.918977604620448
p-value: 1.3560001579371317e-06

Since p-value is less than the alpha 0.05,
We reject the Null Hypothesis.Meaning that Weather is dependent on the season.
```

2-Sample T-Test to Check Effect of Working Day on Number of Electric Cycles Rented

Null Hypothesis (H_0):
Working day has **no effect** on the number of cycles being rented.

Alternate Hypothesis (H_1):
Working day **has an effect** on the number of cycles being rented.

Significance Level (α): 0.05

We will use the **2-Sample T-Test** to test the hypothesis defined above.

Before conducting the two-sample T-Test we need to find if the given data groups have the same variance. If the ratio of the larger data groups to the small data group is less than 4:1 then we can consider that the given data groups have equal variance.

In [434…

```
data_group1 = df[df['workingday']==0]['count'].values
data_group2 = df[df['workingday']==1]['count'].values
print(np.var(data_group1), np.var(data_group2))
np.var(data_group2)// np.var(data_group1)
```

```
30171.346098942427 34040.69710674686
```

Out[434…
```
np.float64(1.0)
```

Here, the ratio is 34040.70 / 30171.35 which is less than 4:1

In [435…
```
stats.ttest_ind(a=data_group1, b=data_group2, equal_var=True)
```

Out[435…
```
TtestResult(statistic=np.float64(-1.2096277376026694), pvalue=np.float64(0.2264
4804226361348), df=np.float64(10884.0))
```

Since pvalue is greater than 0.05 so we cannot reject the Null hypothesis. We don't have the sufficient evidence to say that working day has effect on the number of cycles being rented.

ANOVA to Check if Number of Cycles Rented is Similar or Different Across

1. Weather
2. Season

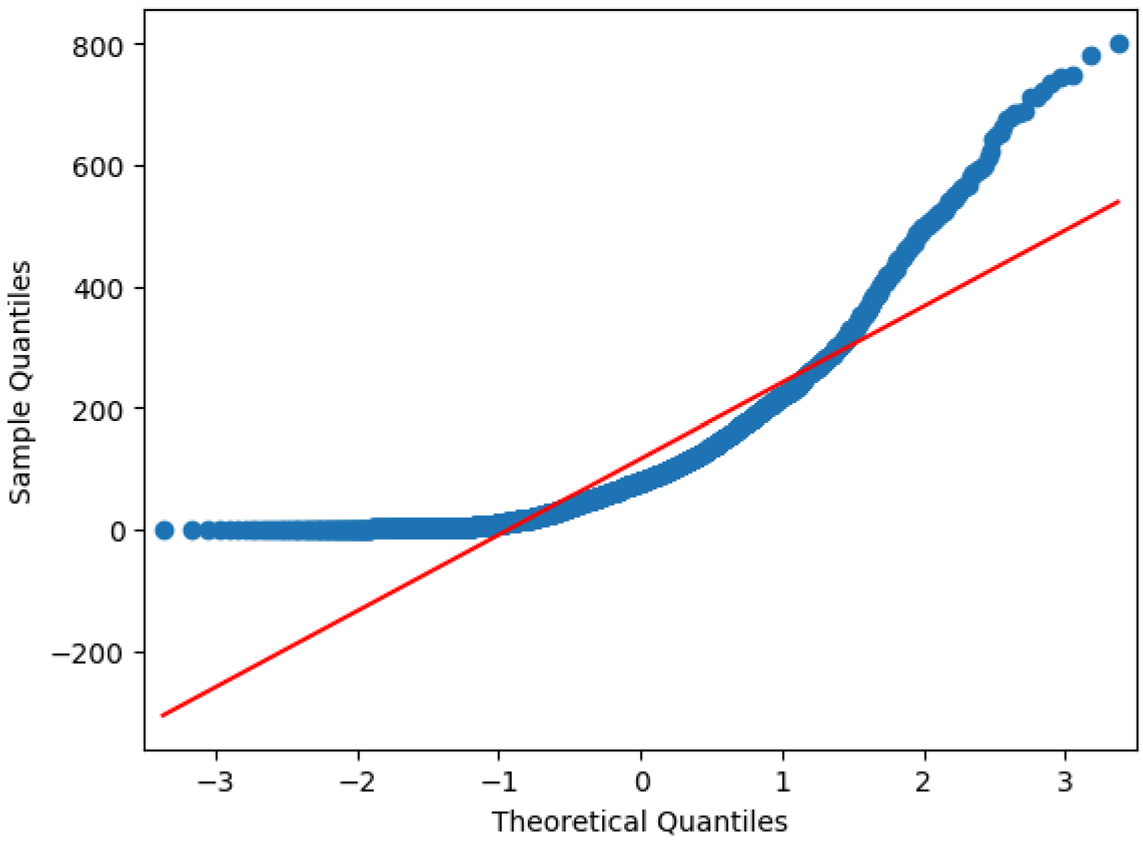
Null Hypothesis (H_0):
Number of cycles rented is **similar** in different weather conditions and seasons.

Alternate Hypothesis (H_1):
Number of cycles rented is **not similar** in different weather conditions and seasons.

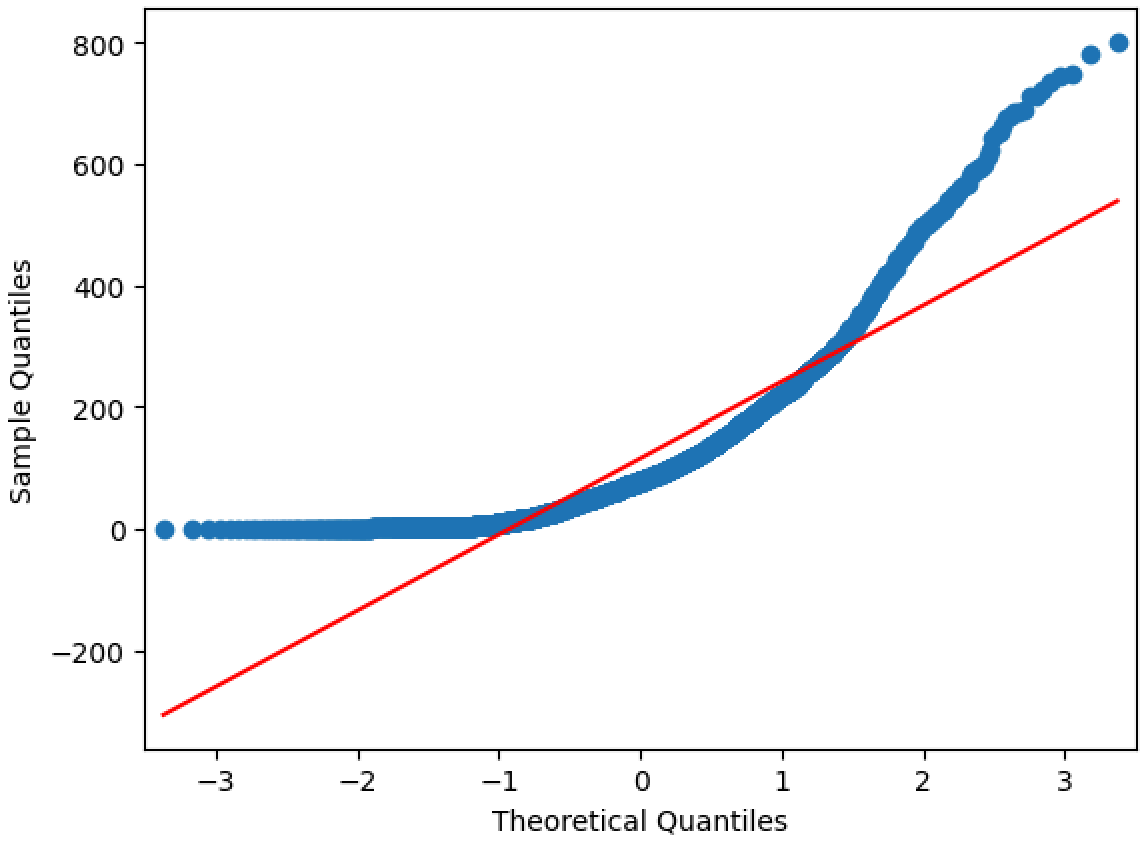
Significance Level (α): 0.05

In [436…
```
from statsmodels.graphics.gofplots import qqplot
import matplotlib.pyplot as plt
```

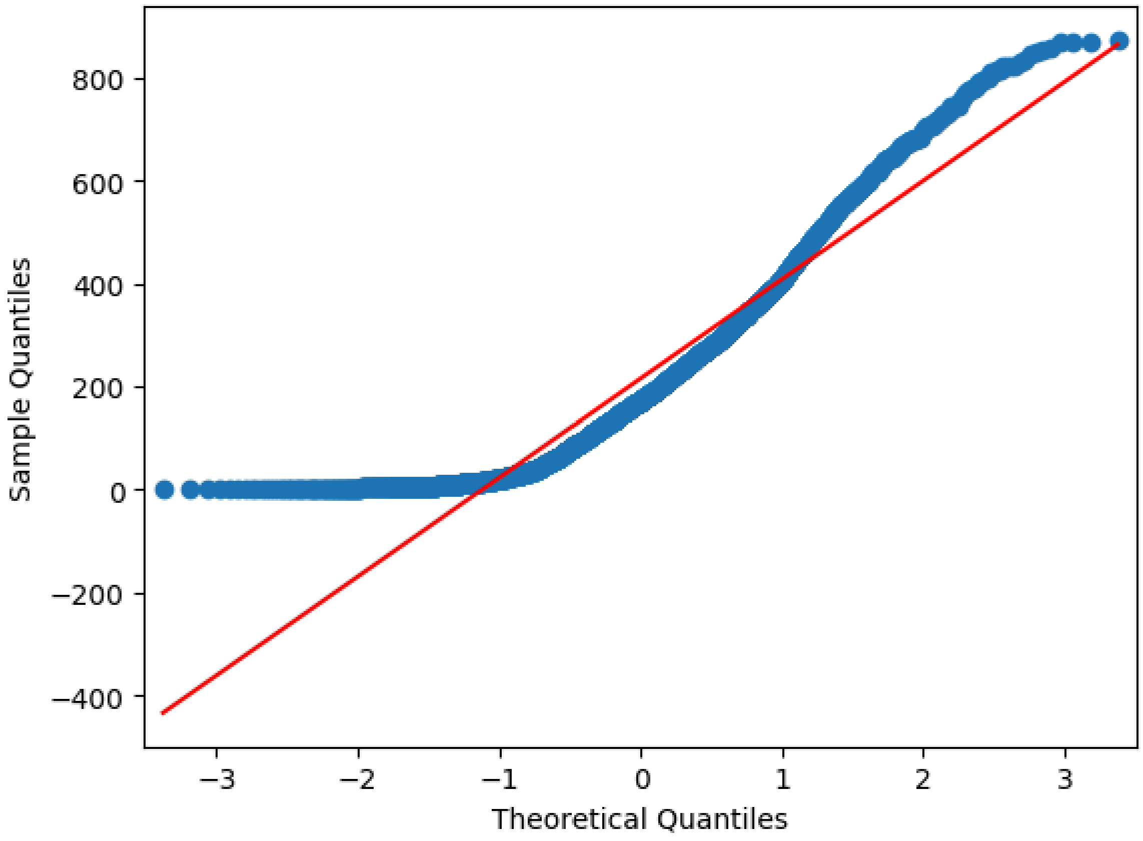
In [449…
```
index = 0
for row in range(4):
    for col in range(2):
        qqplot(groups[index], line="s")
    index += 1
plt.show()
fig, axis = plt.subplots(nrows=4, ncols=2, figsize=(8, 8))
index = 0
for row in range(4):
    for col in range(2):
        sns.histplot(groups[index], ax=axis[row, col], kde=True)
    index += 1
plt.show()
```

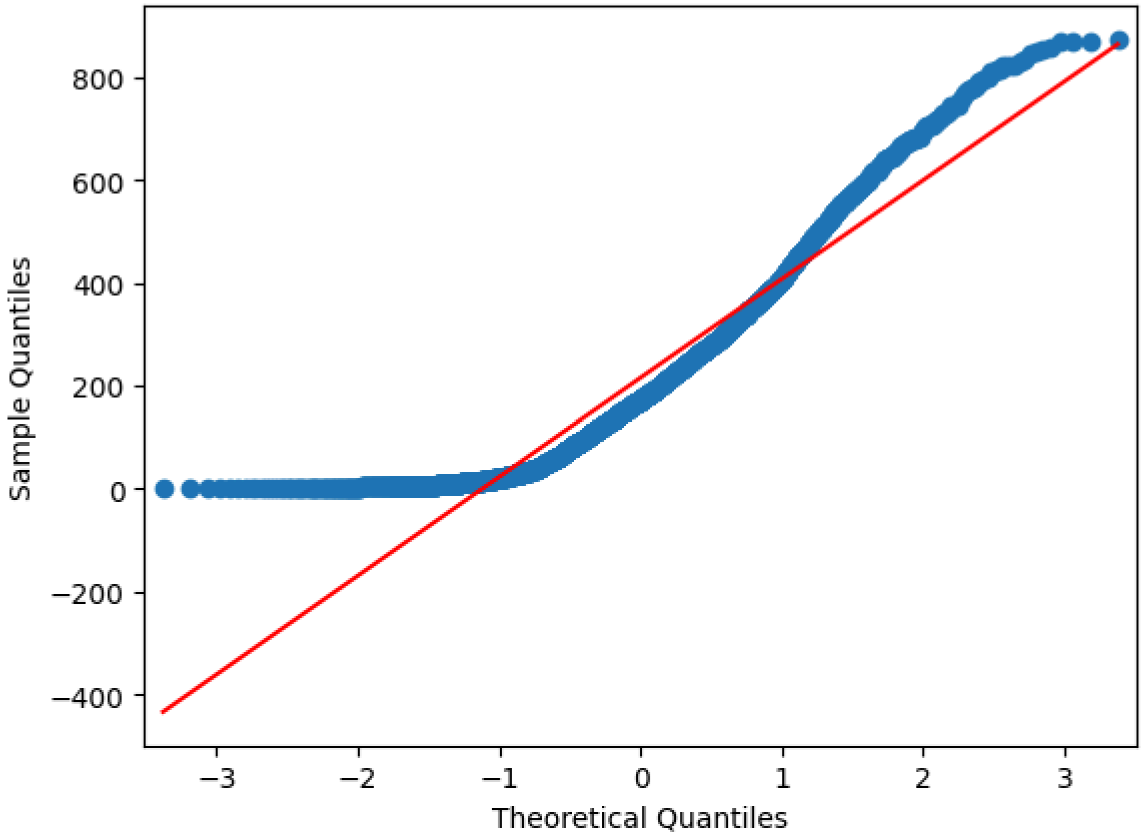


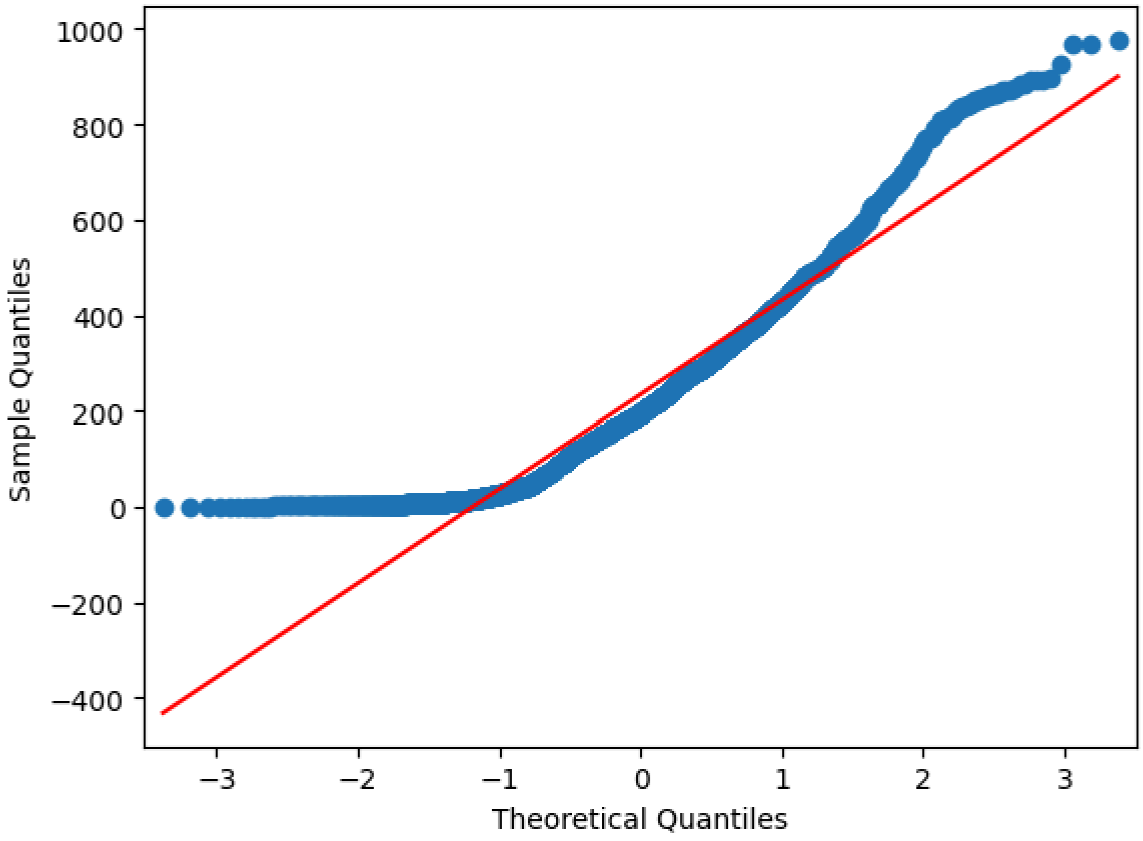
Sample Quantiles
Theoretical Quantiles
800
600
400
200
0
−200
−3
−2
−1
0
1
2
3

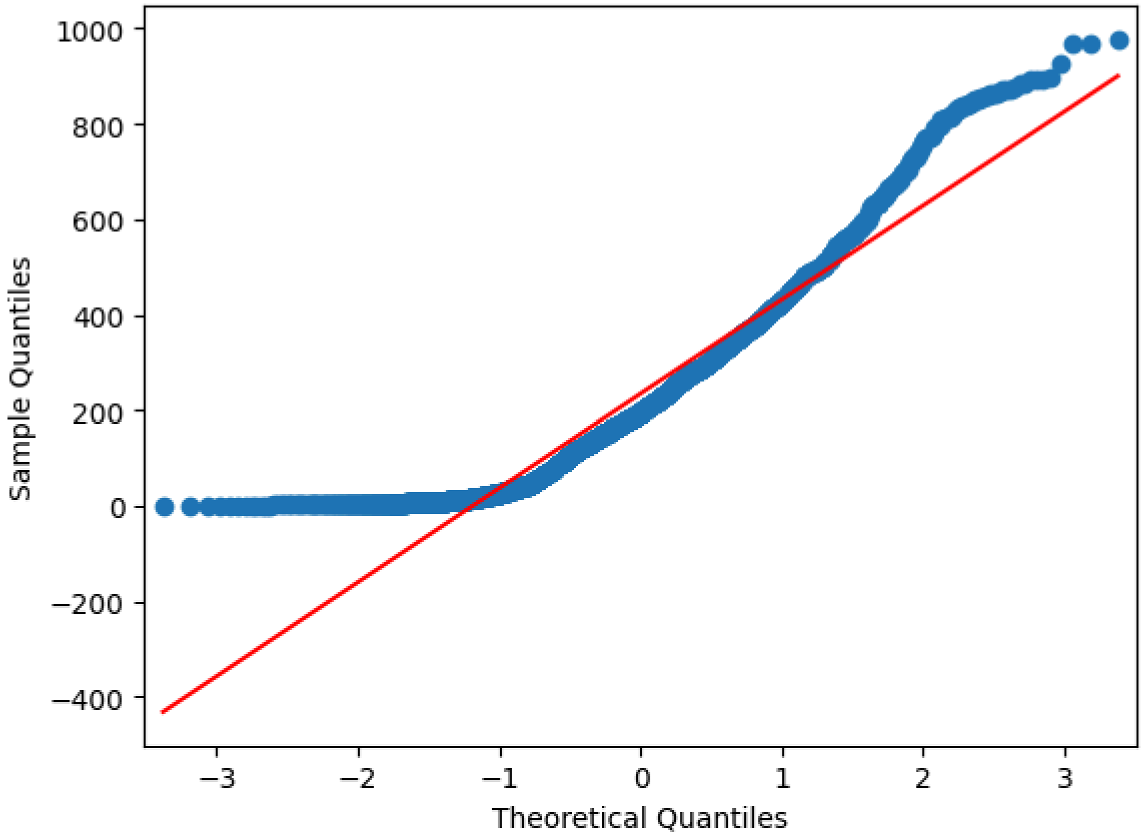


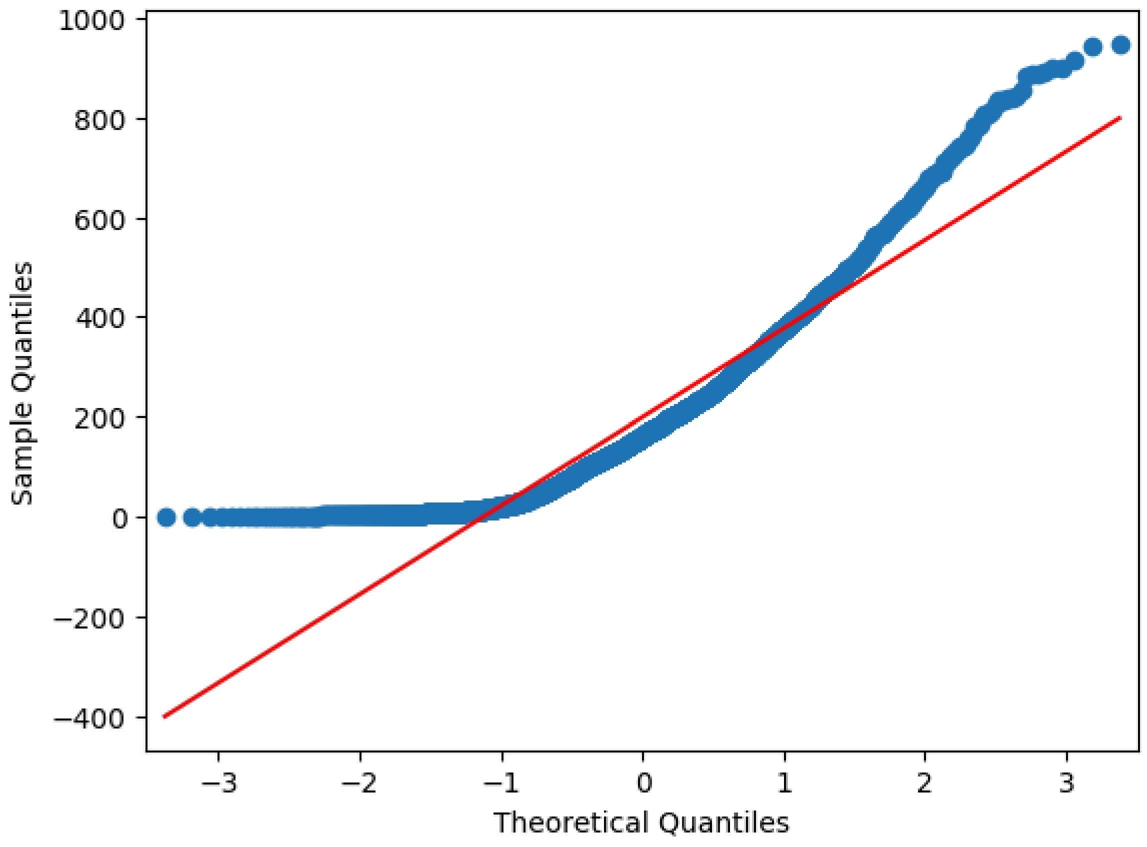
Sample Quantiles
Theoretical Quantiles
800
600
400
200
0
−200
−3
−2
−1
0
1
2
3


800
600
400
200
0
−200
−400
Sample Quantiles
−3
−2
−1
0
1
2
3
Theoretical Quantiles

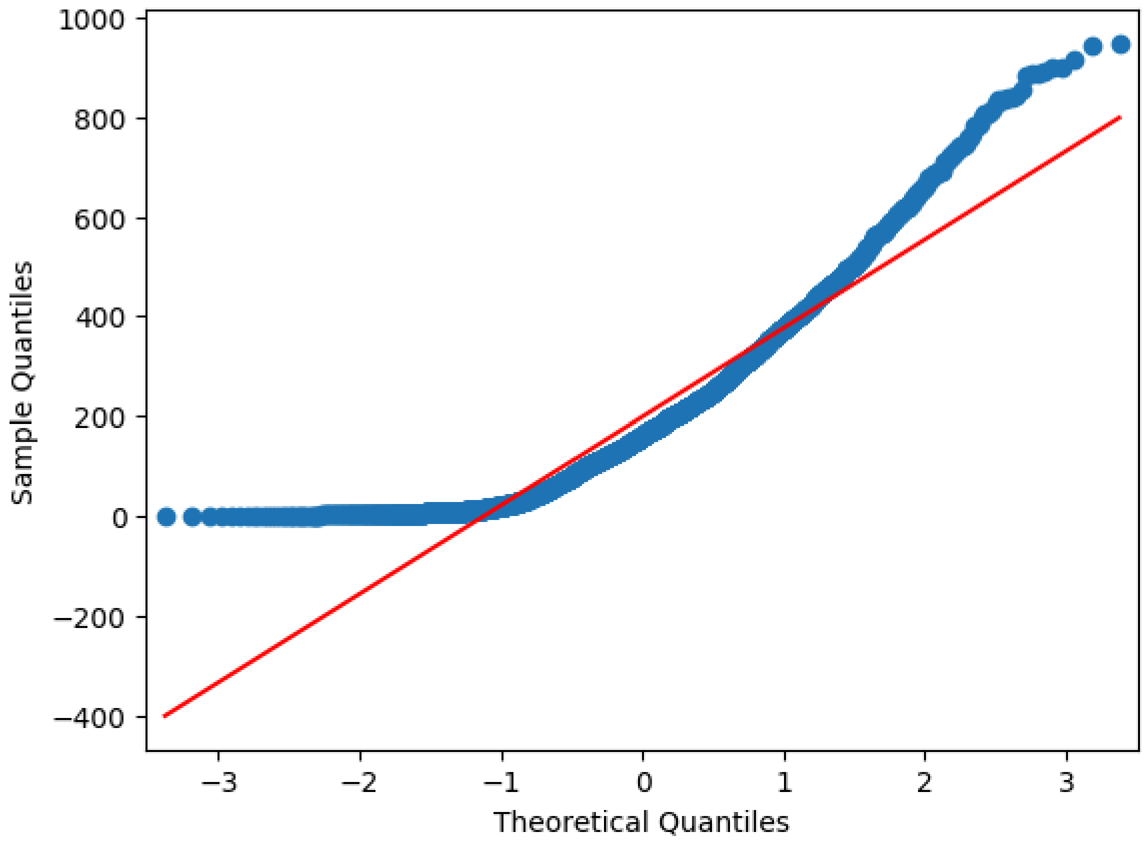

800
600
400
200
0
−200
−400
Sample Quantiles
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Sample Quantiles
Theoretical Quantiles
1000
800
600
400
200
0
−200
−400
−3
−2
−1
0
1
2
3

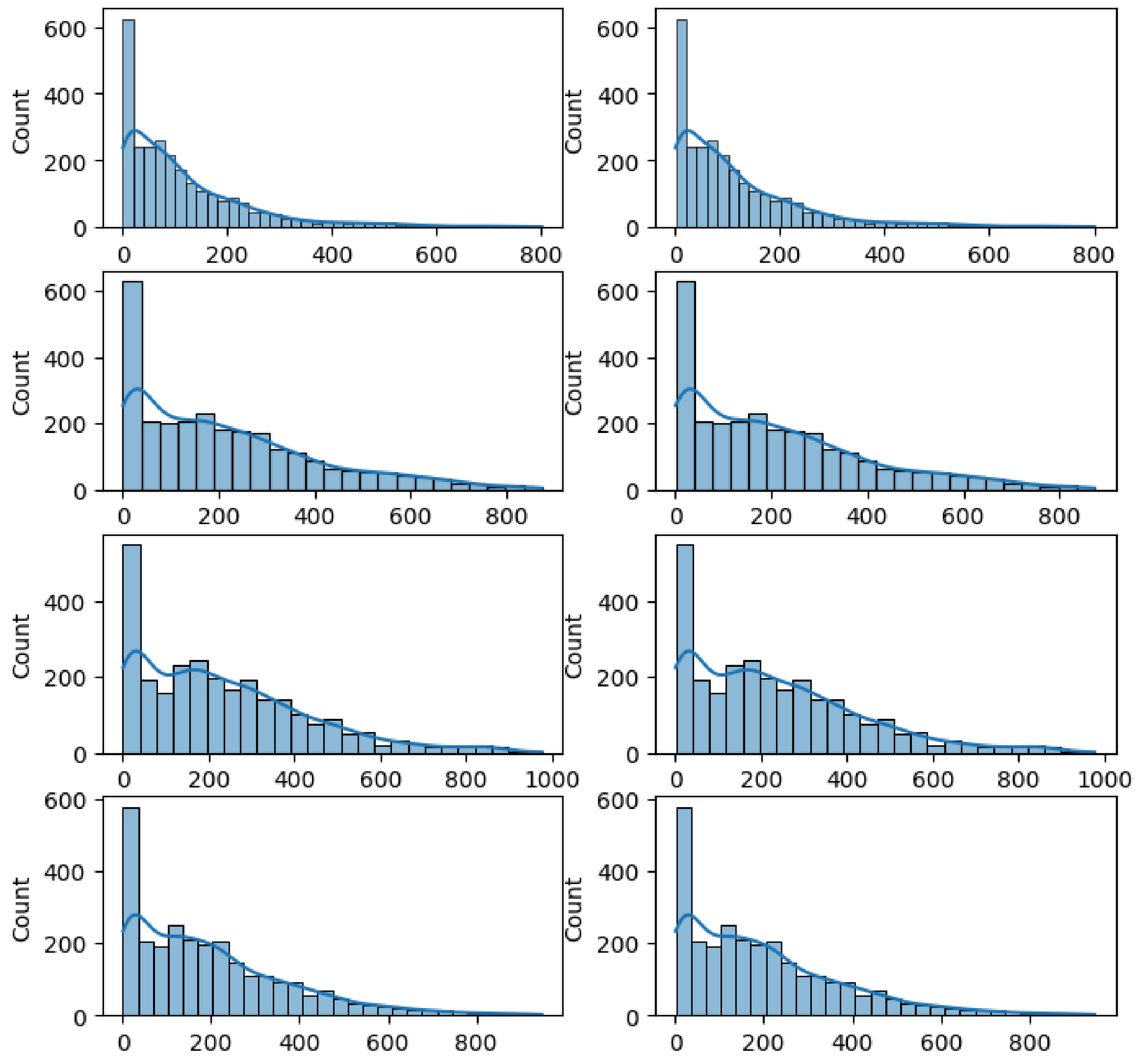

Sample Quantiles
Theoretical Quantiles
1000
800
600
400
200
0
−200
−400
−3
−2
−1
0
1
2
3



Sample Quantiles
Theoretical Quantiles
1000
800
600
400
200
0
−200
−400
−3
−2
−1
0
1
2
3



Sample Quantiles
Theoretical Quantiles
1000
800
600
400
200
0
−200
−400
−3
−2
−1
0
1
2
3



- As per above graphs, all groups are not following Gaussian distribution
- 2: Data is Independent
- 3: Equal variance: Levene's Test

In [ ]:
```python
# Null Hypothesis: Variances are similar across different seasons.
# Alternate Hypothesis: Variances are not similar across different seasons.
# Significance level (alpha): 0.05

# Group 'count' values by 'season'
groups = [group["count"].values for name, group in df.groupby("season")]

# Levene's Test
levene_stat, p_value = stats.levene(*groups)
print("p_value ===", p_value)

if p_value < 0.05:
    print("Reject the Null hypothesis. Variances are not equal.")
else:
    print("Fail to Reject the Null hypothesis. Variances are equal.")
```

```
p_value === 1.0147116860043298e-118
Reject the Null hypothesis. Variances are not equal.
```

- **p-value:** 1.0147116860043298e-118
- **Decision:** Reject the null hypothesis — variances are **not** equal.

Conclusion:

- Based on the **Q-Q plots** and **Levene's Test**, the assumptions of ANOVA (normality and equal variance) are violated.
- ❌ **ANOVA cannot be used.**
- ✅ **Use the Kruskal-Wallis H-test** as a non-parametric alternative.

Conclusion:

- Based on the **Q-Q plots** and **Levene's Test**, the assumptions of ANOVA (normality and equal variance) are violated.
- ❌ **ANOVA cannot be used.**
- ✅ **Use the Kruskal-Wallis H-test** as a non-parametric alternative.

In [ ]:
```python
# Grouping by 'season' and extracting the 'count' values for each group
groups = [group["count"].values for name, group in df.groupby("season")]

# Apply Kruskal-Wallis test
kruskal_stat, p_value = stats.kruskal(*groups)
print("p_value ===", p_value)

if p_value < 0.05:
    print("Since p-value is less than 0.05, we reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
p_value === 2.479008372608633e-151
Since p-value is less than 0.05, we reject the null hypothesis
```

p_value === 2.479008372608633e-151 Since p-value is less than 0.05, we reject the null hypothesis

Since p-value is less than 0.05, we reject the null hypothesis. This implies that Number of cycles rented is not similar in different weather and season conditions

📊 Insights

- In **summer** and **fall** seasons, more bikes are rented compared to other seasons.
- More bikes are rented on **holidays**.
- On **non-working days** (holidays or weekends), slightly more bikes are rented.
- During **adverse weather** (rain, thunderstorm, snow, or fog), fewer bikes are rented.
- When **humidity is less than 20**, the number of bikes rented is **very low**.
- When **temperature is below 10°C**, bike rentals **decrease**.
- When **windspeed exceeds 35 km/h**, fewer bikes are rented.

💡 Recommendations

- During **summer and fall**, increase the number of bikes in stock due to **higher demand**.
- At a **0.05 significance level**, `workingday` has **no significant effect** on bike rentals.
- On **very low humidity** days, reduce the number of bikes available for rent.
- On **very cold days** (temperature < 10°C), maintain a **smaller rental fleet**.
- During **high winds** or **thunderstorms** (windspeed > 35 km/h), reduce bike availability.