In [19]:
```
# Importing Required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import f_oneway # one-way ANOVA
from scipy.stats import chi2_contingency # Chi-square test of independence
from scipy.stats import levene # Levene's test for Equality of Variance
from scipy.stats import shapiro # Shapiro-wilk's test for sample
from scipy.stats import ttest_ind # T-test for independent sample
```

Import and Analyze The Dataset

Import the Dataset:

In [20]:
```
df = pd.read_csv("yulu_data.csv")
```

checking the Structure and Characteristics

- Display the first few rows: `df.head()`
- Summary of the dataset: `df.info()`
- Descriptive statistics: `df.describe()`

In [57]:
```
df.head()
```

Out[57]:

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

In [34]:
```
df.shape
```

Out[34]: (10886, 12)

In [21]:
```
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
```

In [22]:
```
#### The dataset contains 10886 rows and 12 columns.
#  All columns have non-null values, indicating a complete dataset with no missing
# 'season', 'holiday', 'workingday', 'weather', 'temp', 'humidity', 'casual', 'regi
# datetime is categorical (object).
# 'temp','atemp', windspeed  are numerical(float64)
```

In [23]:
```
df.describe()
```

Out[23]:

	season	holiday	workingday	weather	temp	atemp
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000

In [24]:
```
df.describe(include="all")
```

Out[24]:

	datetime	season	holiday	workingday	weather	temp
count	10886	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000
unique	10886	NaN	NaN	NaN	NaN	NaN
top	2011-01-01 00:00:00	NaN	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN	NaN
mean	NaN	2.506614	0.028569	0.680875	1.418427	20.23086
std	NaN	1.116174	0.166599	0.466159	0.633839	7.79159
min	NaN	1.000000	0.000000	0.000000	1.000000	0.82000
25%	NaN	2.000000	0.000000	0.000000	1.000000	13.94000
50%	NaN	3.000000	0.000000	1.000000	1.000000	20.50000
75%	NaN	4.000000	0.000000	1.000000	2.000000	26.24000
max	NaN	4.000000	1.000000	1.000000	4.000000	41.00000

Detect Null Values , Duplicate and Outliers

In [28]:

```
df.isna().sum()
```

Out[28]:

```
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 [35]:

```
# checking null in propation format
df.isna().sum()/len(df)*100
```

```
Out[35]: datetime      0.0
         season        0.0
         holiday       0.0
         workingday    0.0
         weather       0.0
         temp          0.0
         atemp         0.0
         humidity      0.0
         windspeed     0.0
         casual        0.0
         registered    0.0
         count         0.0
         dtype: float64
```

```
In [37]: # checking Duplicated Record in Datasets
         df[df.duplicated()].size
```

```
Out[37]: 0
```

```
In [42]: # checking Number of Unqiue Vlaue in All Column
         for col in df.keys():
             print(col," : ",df[col].nunique())
```

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

```
In [47]: # finding Unqiue Vlaue of categorical object and find out what is distribution thos
         for col in ["season","holiday","workingday","weather"]:
             print(col,":\n\n",df[col].value_counts())
```

```
season :

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

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

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

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

In [49]:

```
# i found only one data point of weather 4 type
# so it's not matter when it's record deleted
```

In [58]:

```
# correlation Heatmap :
df.iloc[:,1:]
```

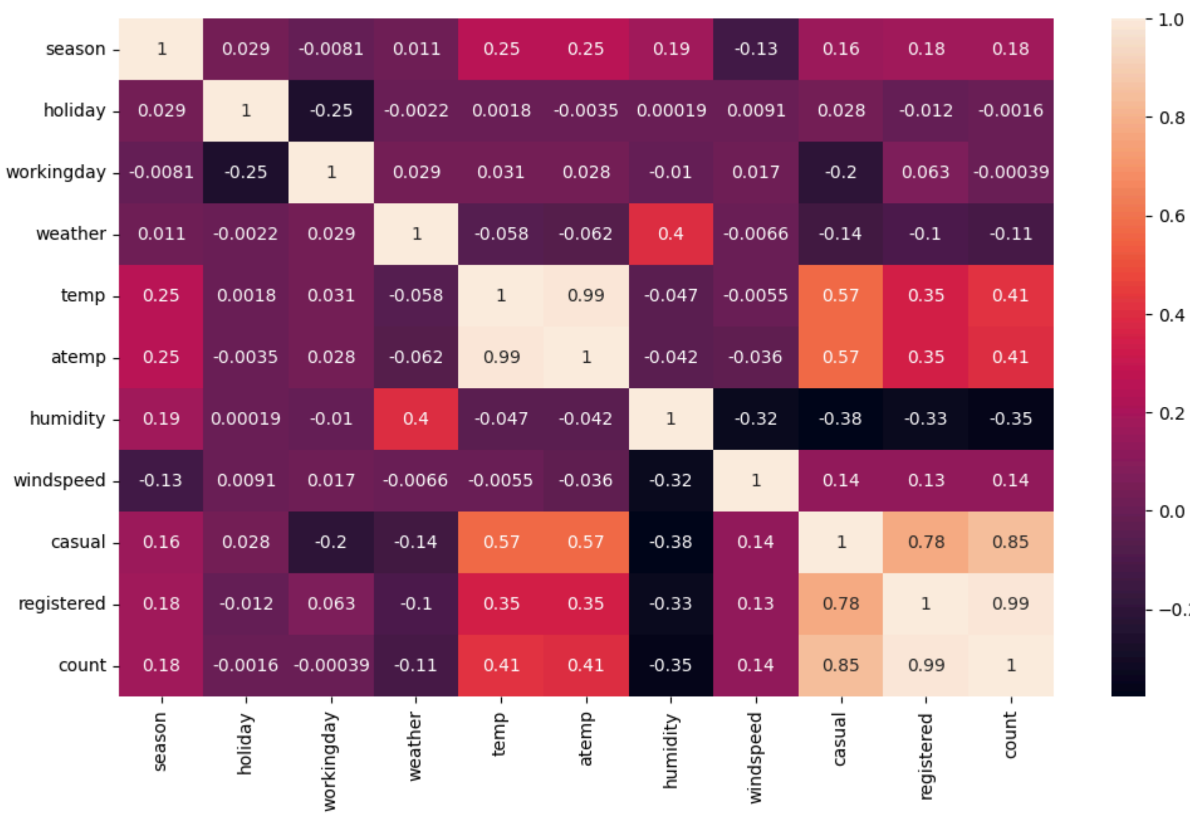
Out[58]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
0	1	0	0	1	9.84	14.395	81	0.0000	3
1	1	0	0	1	9.02	13.635	80	0.0000	8
2	1	0	0	1	9.02	13.635	80	0.0000	5
3	1	0	0	1	9.84	14.395	75	0.0000	3
4	1	0	0	1	9.84	14.395	75	0.0000	0
...	...	...	...	...	...	...	...	...	...
10881	4	0	1	1	15.58	19.695	50	26.0027	7
10882	4	0	1	1	14.76	17.425	57	15.0013	10
10883	4	0	1	1	13.94	15.910	61	15.0013	4
10884	4	0	1	1	13.94	17.425	61	6.0032	12
10885	4	0	1	1	13.12	16.665	66	8.9981	4

10886 rows × 11 columns

In [61]:

```
# heatmap not working on object datatype
plt.figure(figsize=(12, 7))
sns.heatmap(df.iloc[:,1:].corr(method='spearman'),
            annot=True)
plt.show()
```



	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
season	1	0.029	-0.0081	0.011	0.25	0.25	0.19	-0.13	0.16	0.18	0.18
holiday	0.029	1	-0.25	-0.0022	0.0018	-0.0035	0.00019	0.0091	0.028	-0.012	-0.0016
workingday	-0.0081	-0.25	1	0.029	0.031	0.028	-0.01	0.017	-0.2	0.063	-0.00039
weather	0.011	-0.0022	0.029	1	-0.058	-0.062	0.4	-0.0066	-0.14	-0.1	-0.11
temp	0.25	0.0018	0.031	-0.058	1	0.99	-0.047	-0.0055	0.57	0.35	0.41
atemp	0.25	-0.0035	0.028	-0.062	0.99	1	-0.042	-0.036	0.57	0.35	0.41
humidity	0.19	0.00019	-0.01	0.4	-0.047	-0.042	1	-0.32	-0.38	-0.33	-0.35
windspeed	-0.13	0.0091	0.017	-0.0066	-0.0055	-0.036	-0.32	1	0.14	0.13	0.14
casual	0.16	0.028	-0.2	-0.14	0.57	0.57	-0.38	0.14	1	0.78	0.85
registered	0.18	-0.012	0.063	-0.1	0.35	0.35	-0.33	0.13	0.78	1	0.99
count	0.18	-0.0016	-0.00039	-0.11	0.41	0.41	-0.35	0.14	0.85	0.99	1

by default corr use pearson merthod,i'm using spearman that give more undirectal relationship as well

from the correlation we can verify some logical points:

- feeling temperature or aparent temprature and temp are highly correlated, - because they are most of the times approximately the same have a very small diffrerence
- count, causal, registered are all correlated to each other because all of them

In [62]:
```
col_list = ['workingday',       'holiday',       'weather', 'season']
```

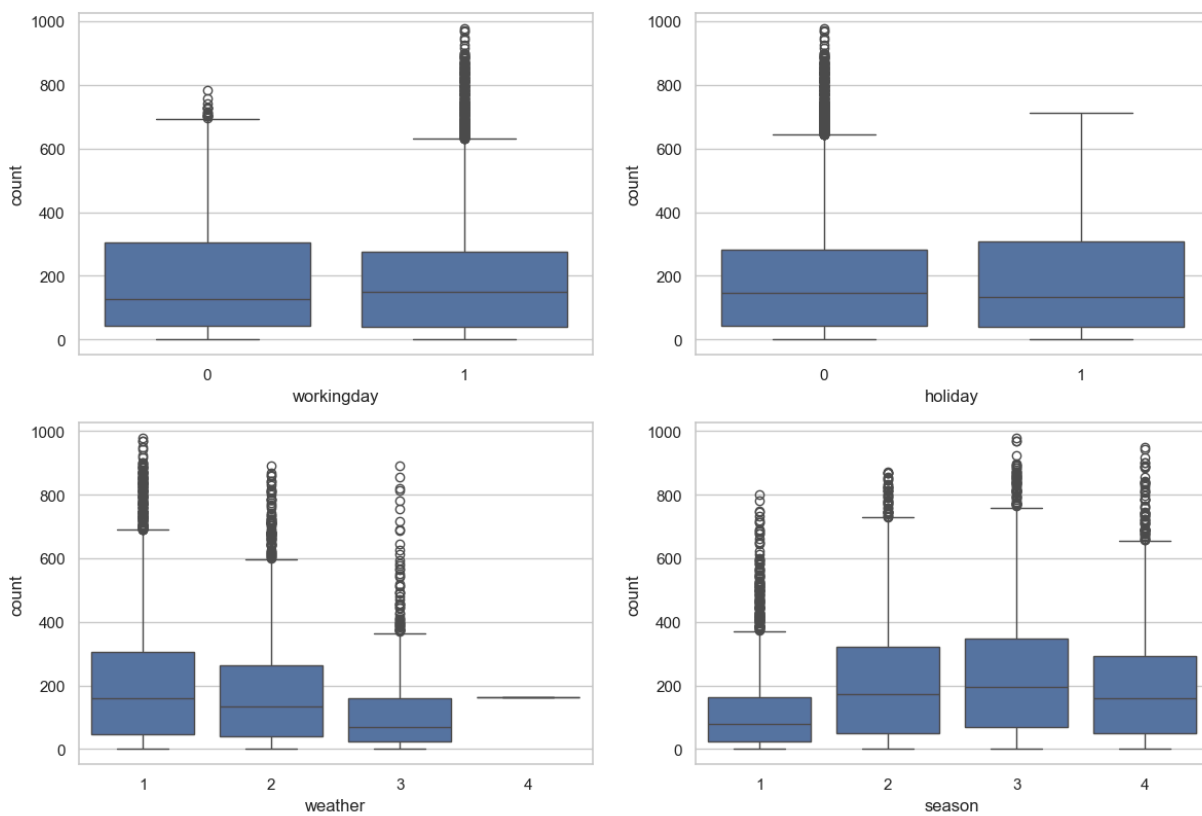
In [63]:
```
# Dropping highly correlated columns -
dfn = df.drop(columns=['casual', 'registered', 'atemp'])
```

In [74]:
```
# Outlier Detection using Boxplots -

sns.set(style="whitegrid")
fig = plt.figure(figsize=(8, 25))
fig.subplots_adjust(right=1.5)
sns.color_palette("hls", 8)

for plot in range(1, len(col_list)+1):
    plt.subplot(5, 2, plot)
    sns.boxplot(x=dfn[col_list[plot-1]], y=dfn['count'])

plt.show()
```



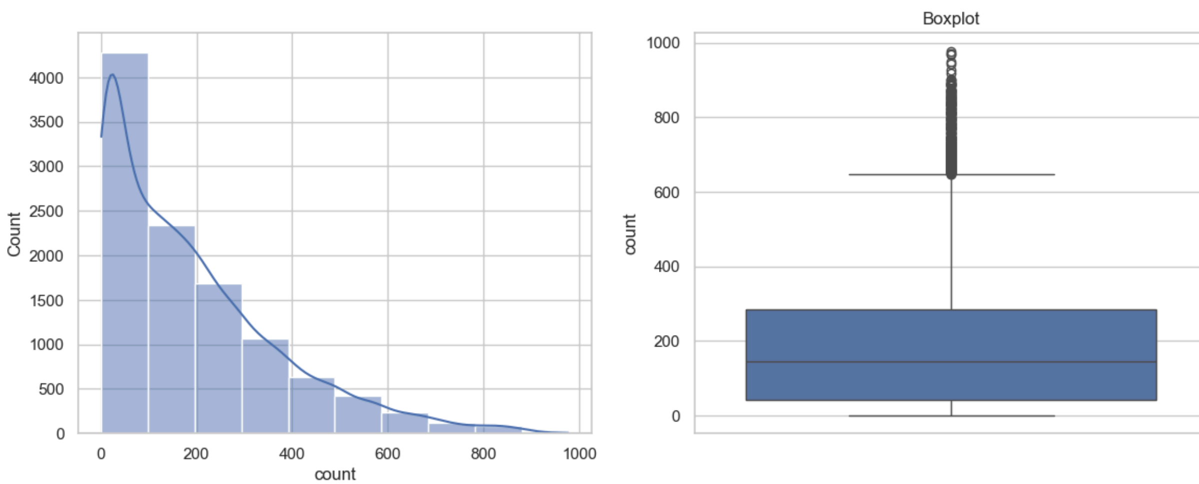
In [81]:

```
# Checking distribution of 'count' column -
plt.figure(figsize=(14, 5))

#Histogram
plt.subplot(1, 2, 1)
sns.histplot(dfn['count'], bins=10,kde=True)

#Boxplot
plt.subplot(1, 2, 2)
sns.boxplot(y=dfn['count'])
plt.title('Boxplot')

plt.show()
```



We can see that outliers are present in the given columns. We need to figure out a way to deal with them before starting with the tests.

We have multiple options available on how to proceed with these outlier values.

1. Try to understand if these values make any sense according to the business problem. If yes, then we can keep them as it is.
2. In case these outliers are some invalid values which do not make much sense, we can remove them using the IQR.
3. Or we can apply a log transformation on the data to reduce the effect of these outliers.
4. •

- The outliers in the given data set are the no. of bike rides per session/day. These values could sometimes be higher than expected due to increase in the crowd on certain days/occasions.
- These data values are important for capturing variations in the data. Hence, in this case, the ideal approach of dealing with outliers would be to leave them as it is.
- But since the tests that we are going to apply are based on the assumption that the dataset is normal or near normal, we will drop those outlier values using the IQR

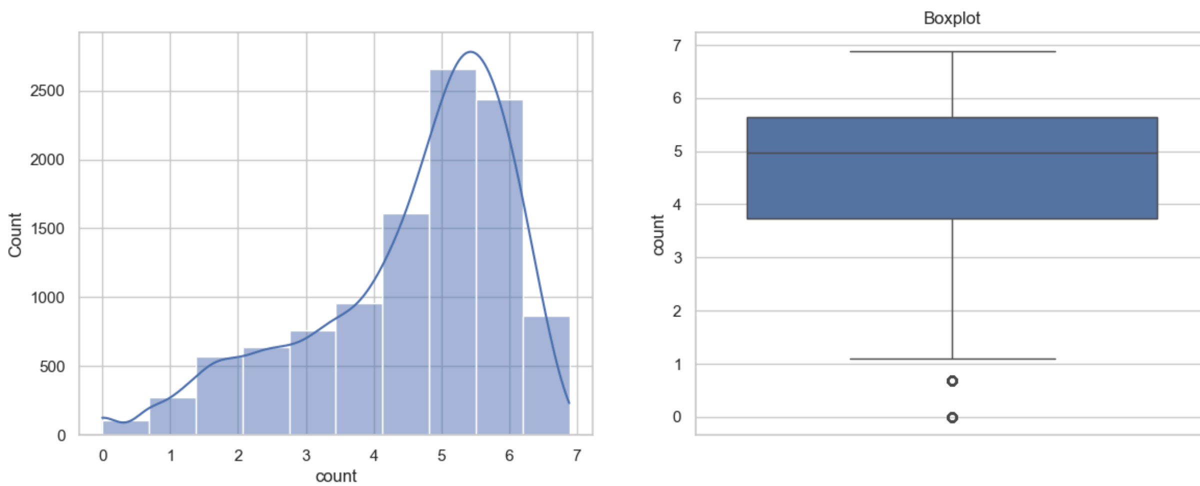
method.

In [83]:
```
# 2.
# Checking distribution after applying log transformation -
plt.figure(figsize=(14, 5))

#Histogram
plt.subplot(1, 2, 1)
sns.histplot(np.log(dfn['count']), bins=10,kde=True)

#Boxplot
plt.subplot(1, 2, 2)
sns.boxplot(y=np.log(dfn['count']))
plt.title('Boxplot')

plt.show()
```



In [84]:
```
# # 3.
# # Outlier Treatment using IQR (not needed but, we can do it) -

q1 = dfn['count'].quantile(0.25)
q3 = dfn['count'].quantile(0.75)
iqr = q3-q1

dfn = dfn[(dfn['count']>(q1-1.5*iqr) ) & (dfn['count']<(q3+1.5*iqr))]

print("No. of rows : ", dfn.shape[0])
```

```
No. of rows :  10583
```

Aggregating the total no. of bike rides based on the given factors -

In [85]:
```
# 1. Workingday -
pd.DataFrame(dfn.groupby('workingday')['count'].describe())
```

Out[85]:

workingday	count	mean	std	min	25%	50%	75%	max
0	3422.0	180.965517	163.782166	1.0	43.0	124.0	295.75	645.0
1	7161.0	173.011591	152.358993	1.0	38.0	143.0	262.00	646.0

In [86]:

```
# 2. Holiday -
pd.DataFrame(dfn.groupby('holiday')['count'].describe())
```

Out[86]:

holiday	count	mean	std	min	25%	50%	75%	max
0	10274.0	175.372786	155.950275	1.0	40.0	138.0	269.0	646.0
1	309.0	182.588997	163.766590	1.0	38.0	127.0	304.0	597.0

In [87]:

```
# 3. Season -
pd.DataFrame(dfn.groupby('season')['count'].describe())
```

Out[87]:

season	count	mean	std	min	25%	50%	75%	max
1	2670.0	112.795131	116.884929	1.0	24.00	78.0	161.00	644.0
2	2633.0	195.653627	166.170802	1.0	45.00	165.0	299.00	646.0
3	2616.0	210.484327	164.055532	1.0	59.75	185.0	323.25	646.0
4	2664.0	184.404655	154.563069	1.0	48.75	154.0	276.25	646.0

In [88]:

```
# 4. Weather -
pd.DataFrame(dfn.groupby('weather')['count'].describe())
```

Out[88]:

weather	count	mean	std	min	25%	50%	75%	max
1	6962.0	187.131140	161.333785	1.0	45.0	153.0	286.0	646.0
2	2770.0	166.117690	146.992422	1.0	39.0	130.0	254.0	646.0
3	850.0	111.862353	121.233389	1.0	23.0	70.5	157.0	646.0
4	1.0	164.000000	NaN	164.0	164.0	164.0	164.0	164.0

Ques. 1 - Is there any significant difference between the no. of bike rides on working and non-working days?

Step 1: Define the null and alternate hypothesis

H_0 : The demand of bikes on weekdays is greater or similar to the demand of bikes on weekend.

H_a : The demand of bikes on weekdays is less than the demand of bikes on weekend.

Let μ_1 and μ_2 be the average no. of bikes rented on weekdays and weekends respectively.

Mathematically, the above formulated hypothesis can be written as:

$H_0 : \mu_1 >= \mu_2$

$H_a : \mu_1 < \mu_2$

Ques. What is the difference between a t-test and a z-test?

Ans.

- A t-test looks at two sets of data that are different from each other, with no standard deviation or variance.
- A z-test views the averages of data sets that are different from each other but have the standard deviation or variance given.
- The t test as compared with z test has its advantage for small sample comparison. As n increases, t approaches to z. The advantage of t test disappears, and t distribution simply becomes z distribution.
- In other words, with large n, t test is just close to z test and one doen't lose anything to continue to use t test.
- In the past, for convenience, we use z table when n > 30. We don't have to do it anymore.
- In fact, all statistical packages use t test even n is large. This is easy, convenience with computer programming, and is correct. All statistical packages are good references.

italicized text#### **Step 2:** Select an appropriate test

Note that the standard deviation of the population is not known.

In [90]:
```
weekday = dfn[dfn['workingday'] == 1]['count'].sample(2999)
weekend = dfn[dfn['workingday'] == 0]['count'].sample(2999)
```

Ques. Why do we take same no. of samples from two different populations for conducting the tests?

Ans.

- Unequal sample sizes can lead to unequal variances between samples, which affects the assumption of equal variances in tests like t-test, ANOVA, etc.
- Having both unequal sample sizes and variances dramatically affects the statistical power of a test.

In [91]:
```
print('The sample standard deviation of the bike rides on weekday is:', round(weekd
print('The sample standard deviation of the bike rides on weekend is:', round(weeke
```

```
The sample standard deviation of the bike rides on weekday is: 154.18
The sample standard deviation of the bike rides on weekend is: 162.95
```

As the sample standard deviations are different, the population standard deviations can be assumed to be different.

This is a one-tailed test concerning two population means from two independent populations. As the population standard deviations are unknown, the two sample independent t-test will be the appropriate test for this problem.

Step 3: Decide the significance level

As given in the problem statement, we select α = 0.05.

In [92]:
```
alpha = 0.05
```

Step 4: Calculate the p-value

In [95]:
```
def result(p_value, alpha):
  if p_value < alpha:
    print(f'As the p-value {p_value} is less than the level of significance, we rej
  else:
    print(f'As the p-value {p_value} is greater than the level of significance, we
```

In [96]:
```
test_stat, p_value = ttest_ind(weekday, weekend, equal_var=False, alternative='less
print('The p-value is : ', p_value)

result(p_value, alpha)
```

```
The p-value is :  0.1031550159978245
As the p-value 0.1031550159978245 is greater than the level of significance, we fail
to reject the null hypothesis.
```

Observation: Since the p-value is greater than the 5% significance level, we fail to reject the null hypothesis. Hence, we have enough statistical evidence to say that the average no. of bike rides during weekdays is greater than or equal to those on weekends.

Ques. 2 - Is there any significant difference between the no. of bike rides on regular days and holidays?

Step 1: Define the null and alternate hypothesis

H_0 : The demand of bikes on regular days is greater or similar to the demand of bikes on holidays.

H_a : The demand of bikes on regular days is less than the demand of bikes on holidays.

Let μ_1 and μ_2 be the average no. of bikes rented on regular days and holidays respectively.

Mathematically, the above formulated hypothesis can be written as:

$H_0 : \mu_1 >= \mu_2$

$H_a : \mu_1 < \mu_2$

Step 2: Select an appropriate test

Again the standard deviation of the population is not known.

In [97]:
```
holiday = dfn[dfn['holiday'] == 1]['count'].sample(299)
regular = dfn[dfn['holiday'] == 0]['count'].sample(299)
```

In [98]:
```
print('The sample standard deviation of the bike rides on holidays is:', round(holi
print('The sample standard deviation of the bike rides on regular days is:', round(
```

```
The sample standard deviation of the bike rides on holidays is: 164.03
The sample standard deviation of the bike rides on regular days is: 155.55
```

As the sample standard deviations are different, the population standard deviations can be assumed to be different.

This is also a one-tailed test concerning two population means from two independent populations. As the population standard deviations are unknown, the two sample independent t-test will be the appropriate test for this problem.

Step 3: Decide the significance level

The significance level (α) is already set to 5% i.e., 0.05

Step 4: Calculate the p-value

In [100…
```
test_stat, p_value = ttest_ind(regular, holiday, equal_var=False, alternative='less
print('The p-value is : ', p_value)

result(p_value, alpha)
```

```
The p-value is :  0.10125786757406595
As the p-value 0.10125786757406595 is greater than the level of significance, we fai
l to reject the null hypothesis.
```

Observation: Since the p-value is greater than the 5% significance level, we fail to reject the null hypothesis. Hence, we have enough statistical evidence to say that the average no. of bike rides during regular days is greater than or equal to those on holidays.

Ques. 3 - Is the demand of bicycles on rent same for different weather conditions?

Step 1: Define the null and alternate hypothesis

H_0 : The average no. of bike rides in different weather conditions are equal.

H_a : The average no. of bike rides in different weather conditions are not equal.

Let μ_1 and μ_2 be the average no. of bikes rented on weekdays and weekends respectively.

Step 2: Select an appropriate test

In [102…
```
dfn = dfn[~(dfn['weather']==4)]
```

In [103…
```
w1 = dfn[dfn['weather'] == 1]['count'].sample(750)
w2 = dfn[dfn['weather'] == 2]['count'].sample(750)
w3 = dfn[dfn['weather'] == 3]['count'].sample(750)
```

In [104…
```
dfn.groupby(['weather'])['count'].describe()
```

Out[104…

weather	count	mean	std	min	25%	50%	75%	max
1	6962.0	187.131140	161.333785	1.0	45.0	153.0	286.0	646.0
2	2770.0	166.117690	146.992422	1.0	39.0	130.0	254.0	646.0
3	850.0	111.862353	121.233389	1.0	23.0	70.5	157.0	646.0

This is a problem, concerning three independent population means. **One-way ANOVA** could be the appropriate test here provided normality and equality of variance assumptions are verified.

The ANOVA test has important assumptions that must be satisfied in order for the associated p-value to be valid.

- The samples are independent.
- Each sample is from a normally distributed population.
- The population variance of the groups are all equal.

Now, we will be using the following statistical tests to check the normality and euality of variance of the data set -

- For testing of normality, Shapiro-Wilk's test is applied to the response variable.
- For equality of variance, Levene test is applied to the response variable.

Shapiro-Wilk's test -

We will test the null hypothesis

> H_0 : Count follows normal distribution

against the alternative hypothesis

> H_a : Count doesn't follow normal distribution

In [106…
```
# Assumption 1: Normality

w, p_value = shapiro(dfn['count'].sample(4999))
print('The p-value is : ', p_value)

result(p_value, alpha)
```

```
The p-value is :  3.5849313376827864e-49
As the p-value 3.5849313376827864e-49 is less than the level of significance, we rej
ect the null hypothesis.
```

Levene's test -

We will test the null hypothesis

> H_0: All the count variances are equal

against the alternative hypothesis

> H_a: At least one variance is different from the rest

In [107…
```
#Assumption 2: Homogeneity of Variance

stat, p_value = levene(w1, w2, w3)
print('The p-value is : ', p_value)

result(p_value, alpha)
```

```
The p-value is :  1.5073745994321196e-18
As the p-value 1.5073745994321196e-18 is less than the level of significance, we rej
ect the null hypothesis.
```

Note: If these assumptions are not true for a given set of data (like in this case), it may still be possible to use the **Kruskal-Wallis H-test** or the **Alexander-Govern test** although with some loss of power.

Central Limit Theorem -

You all must have studies about the CLT in previous classes.

- According to this theorem, the distribution of sample means approximates a normal distribution as the sample size gets larger, regardless of the population's distribution.
- In other words, if we find the mean of a large number of independent random variables, the mean will follow a normal distribution, irrespective of the distribution of the original variables.
- In practice, sample sizes equal to or greater than 30-40 are often considered sufficient for the CLT to hold.

Hence, the sample size being large enough, we don't need to worry about the non-normality of distribution of the data set in hand before applying the tests.

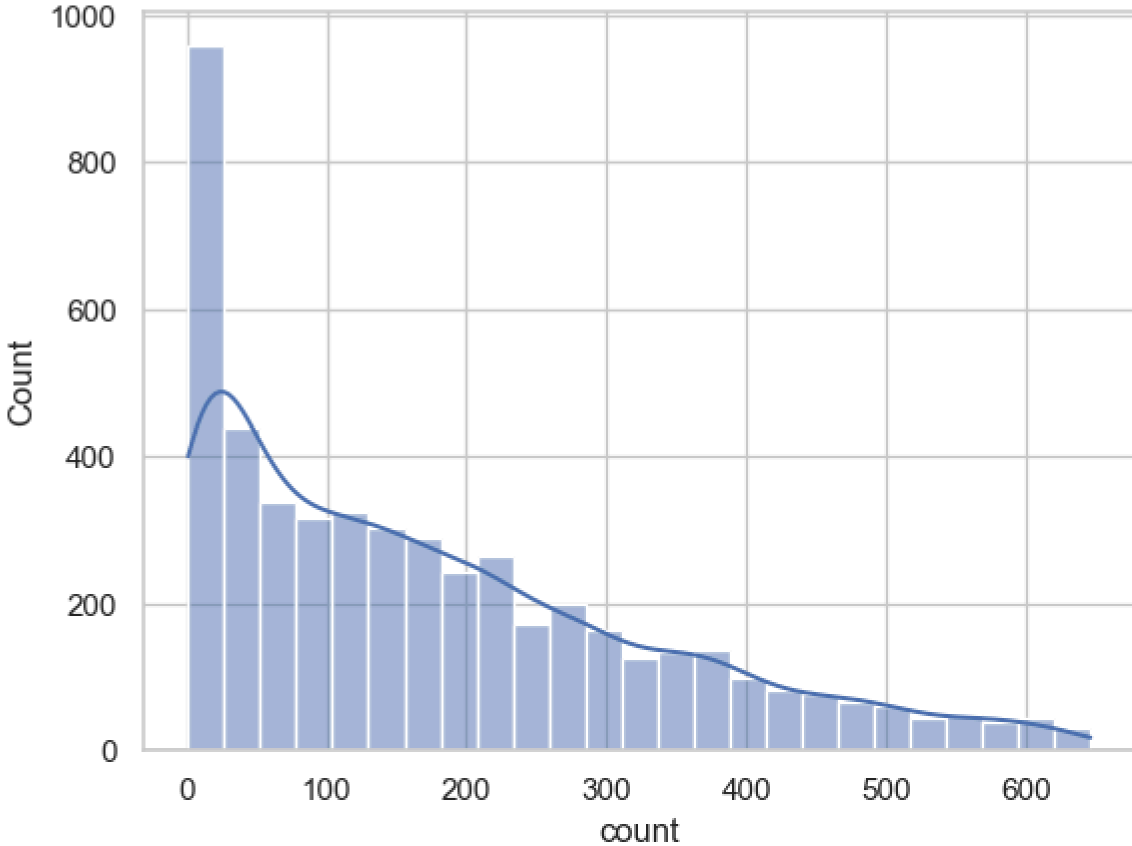
Eventually, as the sample size gets larger, the distribution of sample means will fall into a normal or near normal shape.

Ques. What are some of the basic methods (other than statistical tests) to test the normality & homogeneity of variance?

A. To check for **Normality -**

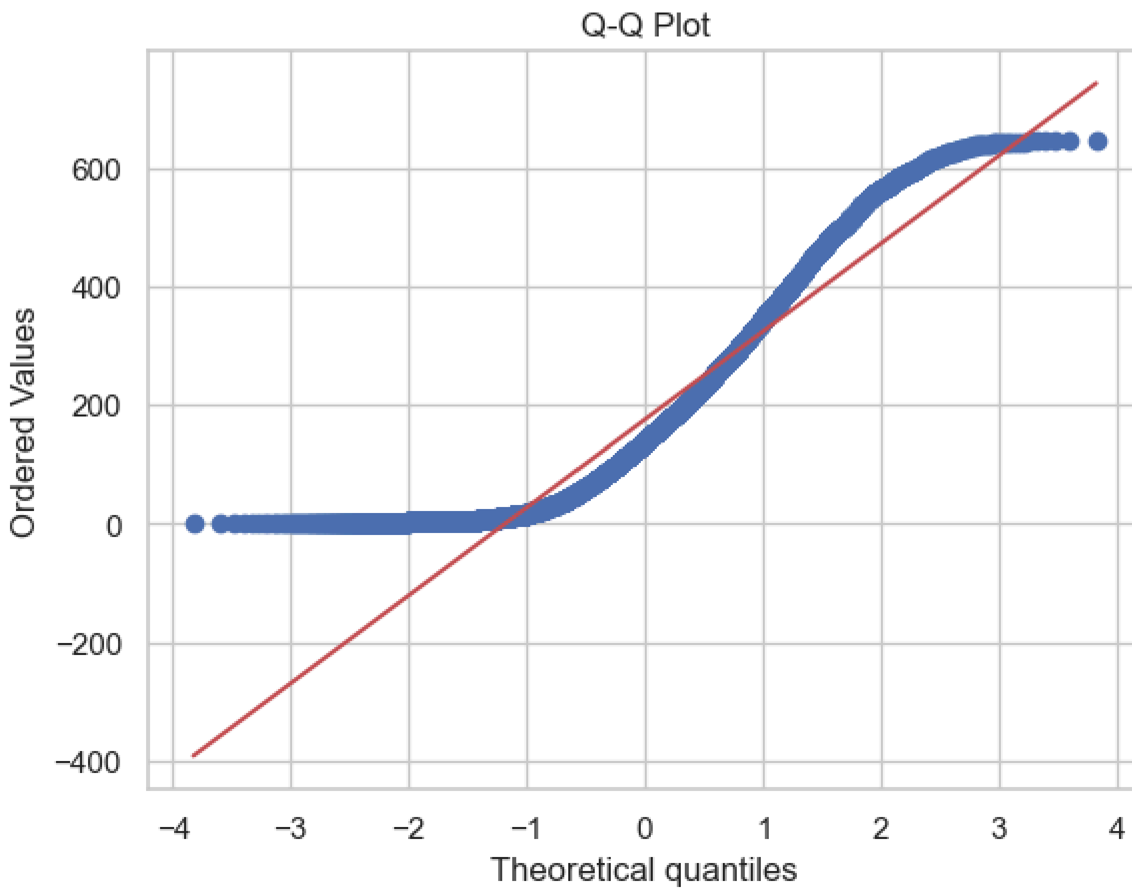
In [111…

```
# Method 1: Using a Histogram
# You should see a 'Bell' shaped curve.
sns.histplot(dfn['count'].sample(4999),kde=True)
plt.show()
```



In [116…
```
from scipy import stats
```

In [117…
```
# Method 2: Using a Q-Q plot
# The linearity of points suggests that the data is normally distributed.
stats.probplot(dfn['count'], dist='norm', fit=True, plot=plt)
plt.title('Q-Q Plot')
plt.show()
```



In [118…

```
# Method 3: Check skewness & kurtosis
# Skewness should be close to 0 and Kurtosis close to 3.

print("Skewness : ", df['count'].skew())
print("Kurtosis : ", df['count'].kurt())
```

```
Skewness :  1.2420662117180776
Kurtosis :  1.3000929518398334
```

In [119…

```
## Method 4: Using KS Test to compare with the Gaussian CDF
zs = (dfn["count"] - dfn["count"].mean())/dfn["count"].std()
stats.kstest(zs, stats.norm.cdf)
```

Out[119…

```
KstestResult(statistic=0.13182946815041796, pvalue=8.049661990629187e-161, statist
ic_location=-1.1177847497853437, statistic_sign=-1)
```

B. To check for **Homogeneity of Variance -**

In [120…

```
# Method 1:
print(w1.var(), w2.var(), w3.var())
```

```
26252.44592790387 20343.58471918113 14716.515512238542
```

Step 3: Decide the significance level

The significance level (α) is already set to 5% i.e., 0.05

Step 4: Calculate the p-value

One-way ANOVA -

In [121...

```
test_stat, p_value = f_oneway(w1, w2, w3)
print('The p-value is : ', p_value)

result(p_value, alpha)
```

```
The p-value is :  5.769128153160212e-25
As the p-value 5.769128153160212e-25 is less than the level of significance, we reje
ct the null hypothesis.
```

Observation: Since the p-value is less than the 5% significance level, we reject the null hypothesis. Hence, we have enough statistical evidence to say that the average no. of bike rides in different weather conditions are not equal.

Ques. 4 - Is the demand of bicycles on rent same for different seasons?

Step 1: Define the null and alternate hypothesis

H_0 : The average no. of bike rides in different seasons are equal.

H_a : The average no. of bike rides in different seasons are not equal.

Step 2: Select an appropriate test

In [122...

```
s1 = dfn[dfn['season'] == 1]['count'].sample(2399)
s2 = dfn[dfn['season'] == 2]['count'].sample(2399)
s3 = dfn[dfn['season'] == 3]['count'].sample(2399)
s4 = dfn[dfn['season'] == 3]['count'].sample(2399)
```

In [123...

```
dfn.groupby(['season'])['count'].describe()
```

Out[123...

	count	mean	std	min	25%	50%	75%	max
season								
1	2669.0	112.775946	116.902627	1.0	24.00	78.0	161.00	644.0
2	2633.0	195.653627	166.170802	1.0	45.00	165.0	299.00	646.0
3	2616.0	210.484327	164.055532	1.0	59.75	185.0	323.25	646.0
4	2664.0	184.404655	154.563069	1.0	48.75	154.0	276.25	646.0

Step 3: Decide the significance level

The significance level (α) is already set to 5% i.e., 0.05

Step 4: Calculate the p-value

We have already performed tests for normality and homogeneity of variance. So we will be directly moving onto the One-way ANOVA test.

In [124...

```
test_stat, p_value = f_oneway(s1, s2, s3, s4)
print('The p-value is : ', p_value)

result(p_value, alpha)
```

```
The p-value is :  2.1425369128418175e-143
As the p-value 2.1425369128418175e-143 is less than the level of significance, we re
ject the null hypothesis.
```

Observation: Since the p-value is less than the 5% significance level, we reject the null hypothesis. Hence, we have enough statistical evidence to say that the average no. of bike rides in different seasons are not equal.

Ques. How does the increase in sample size affect hypothesis testing?

Ans. Increasing sample size makes the hypothesis test more sensitive, more likely to reject the null hypothesis when it is, in fact, false. Thus, it increases the power of the test.

Ques. 5 - Are the weather conditions significantly different during different seasons?

Step 1: Define the null and alternate hypothesis

H_0 : Weather conditions are independent of the season.

H_a : Weather condition depends on the ongoing season.

Although the data values in 'season' and 'weather' columns are numerical, as per our intuition, they still represent different catgories. Hence, we will encode them accordingly before moving onto the tests.

In [125...

```
dict1 = {1: 'Sunny',
         2: 'Cloudy',
         3: 'Rainy'}
dfn['weather_enc'] = dfn['weather'].map(dict1)
```

In [126…

```
dict2 = {1: 'Summer',
         2: 'Monsoon',
         3: 'Winter',
         4: 'Autumn'}
dfn['season_enc'] = dfn['season'].map(dict2)
```

Here we will be comparing two different categorical variables, 'season' and 'weather'. So will perform a **Chi-square test**.

In [127…

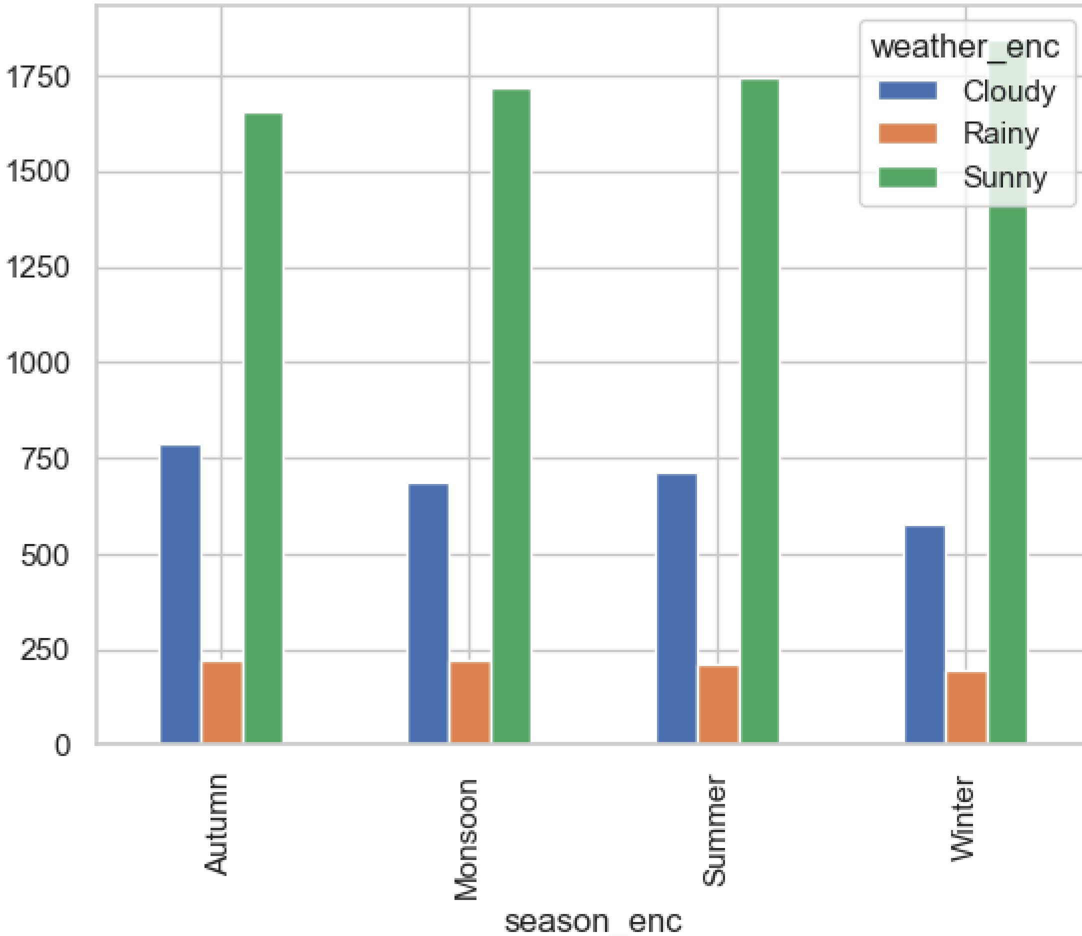
```
contigency= pd.crosstab(dfn.season_enc, dfn.weather_enc)
contigency
```

Out[127…

weather_enc	Cloudy	Rainy	Sunny
season_enc			
Autumn	787	221	1656
Monsoon	690	223	1720
Summer	714	211	1744
Winter	579	195	1842

In [129…

```
contigency.plot(kind='bar')
plt.show()
```



Step 3: Decide the significance level

The significance level (α) is already set to 5% i.e., 0.05

Step 4: Calculate the p-value

In [130…

```
chi2, pval, dof, exp_freq = chi2_contingency(contigency, correction=False)
print('Chi-square Statistic: {} \n P-value: {} \n Degree of Freedom: {} \n Expected
```

```
Chi-square Statistic: 44.1979555965044
 P-value: 6.753122128664597e-08
 Degree of Freedom: 6
 Expected Frequencies: [[ 697.34265734  213.98601399 1752.67132867]
 [ 689.22793423  211.4959365  1732.27612928]
 [ 698.65148365  214.38763939 1755.96087696]
 [ 684.77792478  210.13041013 1721.09166509]]
```

In [131…

```
result(pval, alpha)
```

```
As the p-value 6.753122128664597e-08 is less than the level of significance, we reje
ct the null hypothesis.
```

Observation: Since the p-value is less than the 5% significance level, we reject the null hypothesis. Hence, we have enough statistical evidence to say that the weather conditions are dependent on the ongoing season.

italicized text### Insights and Recommendations

EDA based insights -

1. Total 10,886 rows were present in the data set.
2. Neither missing values, nor duplicate rows were found.
3. 'temp' and 'atemp' columns were found to be highly correlated. Dropping one of them (atemp) to avoid multicollinearity.
4. 'count', 'casual' and 'registered' columns were highly correlated. Dropping casual & registered columns to avoid multicollinearity.
5. Outlier values were found in the 'count' column.

Insights from hypothesis testing -

1. The no. of bikes rented on weekdays is comparatively higher than on weekends.
2. The no. of bikes rented on regular days is comparatively higher than on holidays.
3. The demand of bicycles on rent differs under different weather conditions.
4. The demand of bicycles on rent is different during different seasons.
5. The weather conditions are surely dependent upon the ongoing season.

Miscellaneous observations -

The distribution of 'count' column wasn't actually normal or near normal. Infact the column's distribution is found to be a bit skewed towards right.

Generic recommendations -

- The demand of bikes on rent are usually higher during Weekdays.
- The demand of bikes on rent are usually higher during Regular days.
- The chances of person renting a bike are usually higher during Season 3.
- The chances of person renting a bike are usually higher during Weather condition 1.

We recommend the company to maintain the bike stocks accordingly.

In [ ]: