In [24]:
```
import pandas as pd
import numpy as np
import seaborn as sbn
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import ttest_ind # T-test for independent samples
from scipy.stats import shapiro # Shapiro-Wilk’s test for Normality
from scipy.stats import levene # Levene's test for Equality of Variance
from scipy.stats import f_oneway # One-way ANOVA
from scipy.stats import chi2_contingency # Chi-square test of independence
```

In [2]:
```
df = pd.read_excel("C:/Users/Hp/Desktop/yulu.xlsx")
```

In [3]:
```
df.head()
```

Out[3]:

	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

In [4]:
```
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  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: datetime64[ns](1), float64(3), int64(8)
memory usage: 1020.7 KB
```

In [6]:
```
df.shape
```

Out[6]:
```
(10886, 12)
```

In [7]:
```
#checking for null values
df.isna().sum()
```

Out[7]:
```
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 [8]:
```
#There are no null values
```

In [9]:
```
# Checking for duplicate rows -
dup_rows = df[df.duplicated()]
print("No. of duplicate rows: ", dup_rows.shape[0])
```

```
No. of duplicate rows:  0
```

In [25]:
```
#1: spring, 2: summer, 3: fall, 4: winter
df['season'].value_counts()
```

Out[25]:
```
season
4    2734
2    2733
3    2733
1    2686
Name: count, dtype: int64
```

In [14]:
```
df['holiday'].value_counts()
```

Out[14]:
```
holiday
0    10575
1      311
Name: count, dtype: int64
```

In [15]:
```
#if day is neither weekend nor holiday is 1, otherwise is 0.
df['workingday'].value_counts()
```

Out[15]:
```
workingday
1    7412
0    3474
Name: count, dtype: int64
```

In [26]:
```
#1: Clear, Few clouds, partly cloudy, partly cloudy
#2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
#3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
#4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
df['weather'].value_counts()
```

Out[26]:
```
weather
1    7192
2    2834
3     859
4       1
Name: count, dtype: int64
```

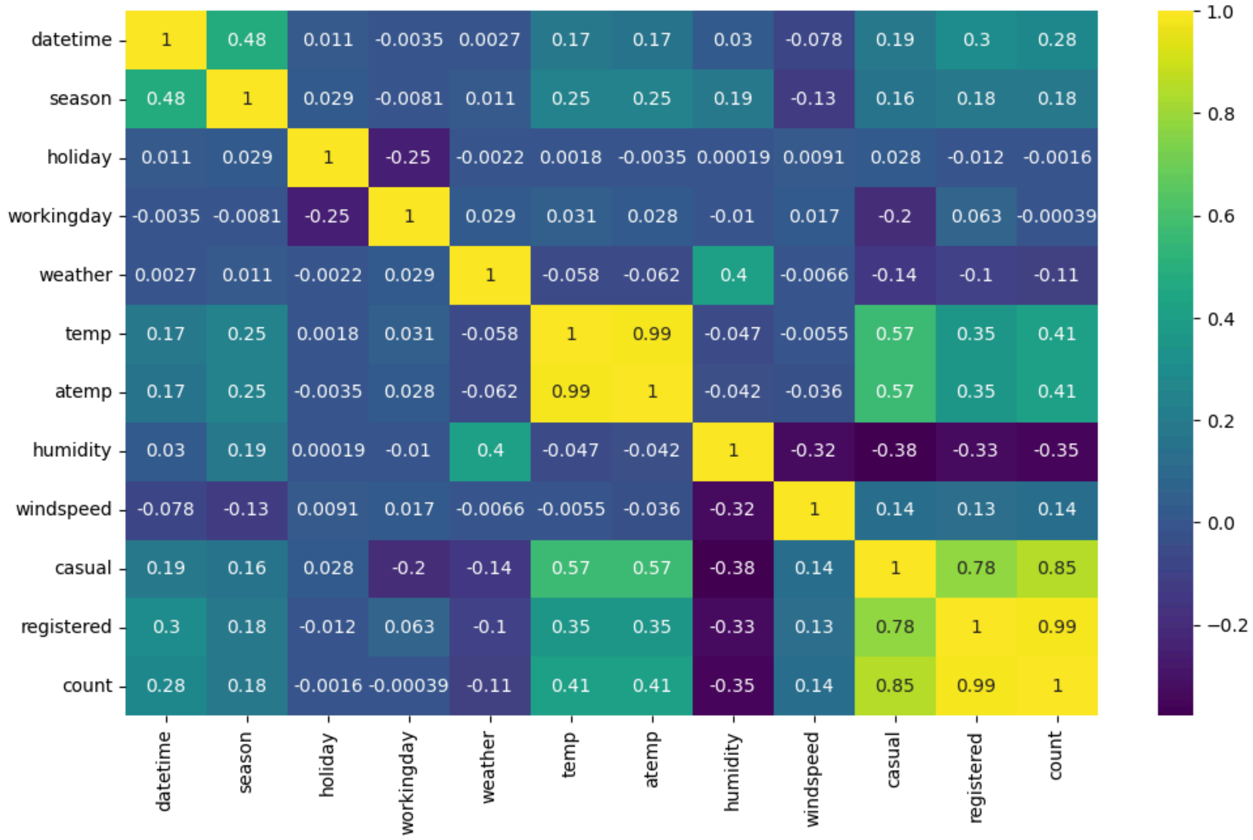
Converting season,holidays,workingday,weather to categorical

In [31]:
```
df['season'] = pd.Categorical(df.season)
df['holiday'] = pd.Categorical(df.holiday)
df['workingday'] = pd.Categorical(df.workingday)
df['weather'] = pd.Categorical(df.weather)
```

In [32]:
```
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  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), datetime64[ns](1), float64(3), int64(4)
memory usage: 723.7 KB
```

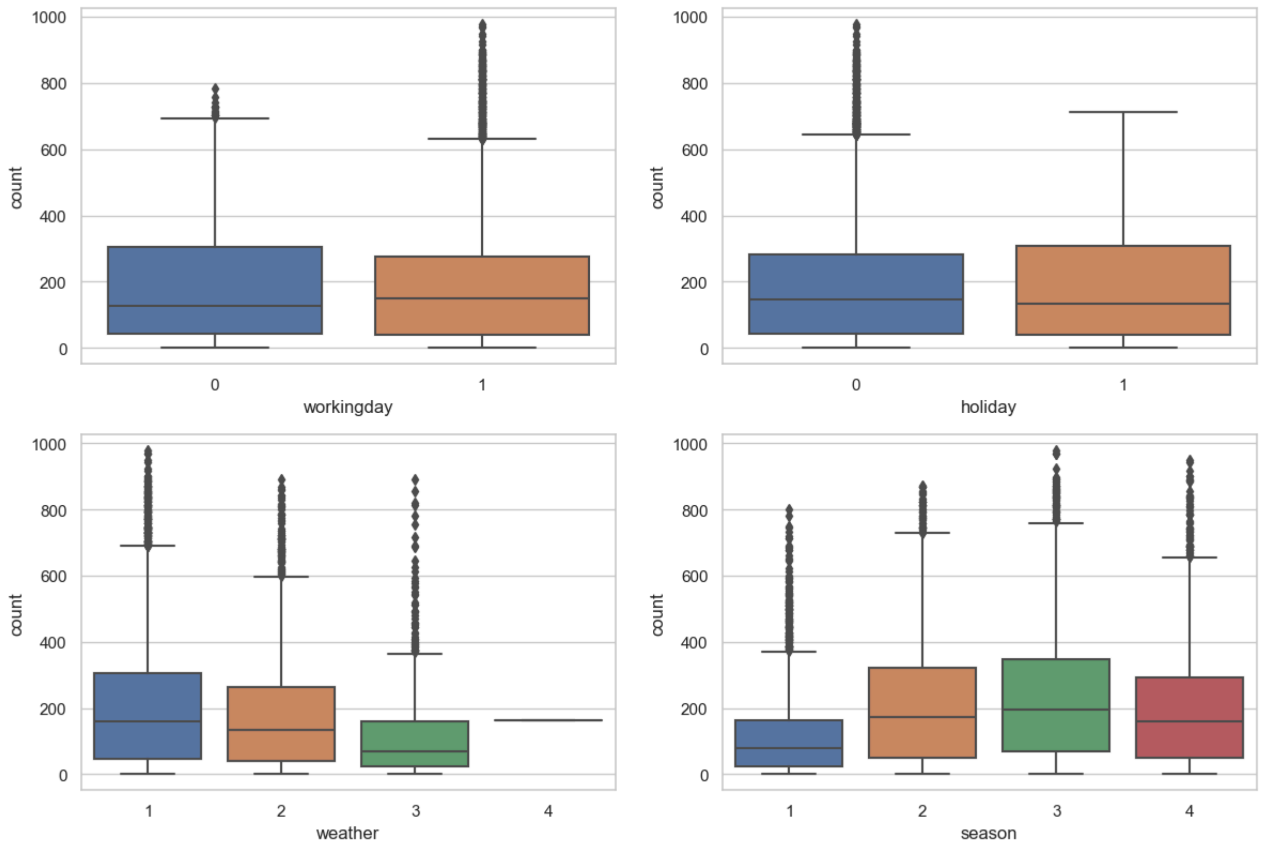
In [33]:
```
# Correlation Heatmap -

plt.figure(figsize=(12, 7))
sbn.heatmap(df.corr(method='spearman'),
            annot=True, cmap='viridis')
plt.show()
```



from the correlation we can verify some logical points:

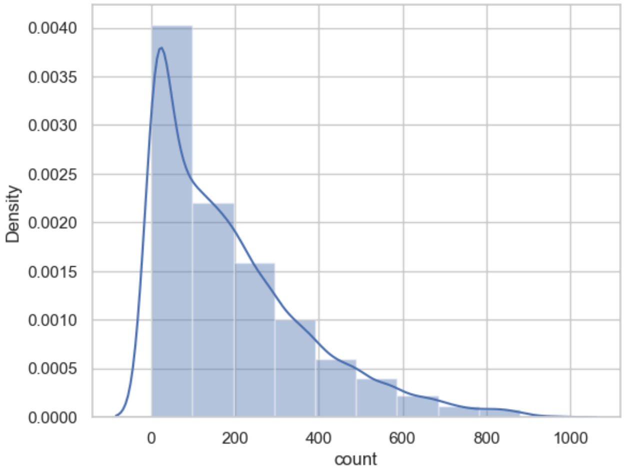
- feeling temperature or aparent temprature and temp are highly correlated, because they are most of the times approximately the same have a very small diffrerence
- count, causal, registered are all correlated to each other because all of them are linked as per: causal + registered = count

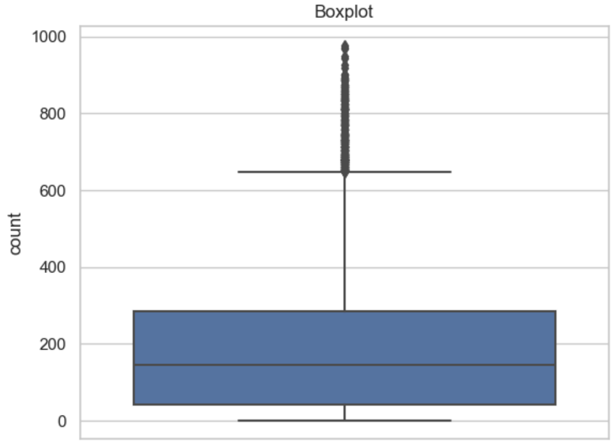
In [36]:
```
# Outlier Detection using Boxplots -
col_list = ['workingday',  'holiday',  'weather', 'season']
sns.set(style="whitegrid")
fig = plt.figure(figsize=(8, 25))
fig.subplots_adjust(right=1.5)

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

plt.show()
```



In [37]:
```
# Checking distribution of 'count' column -
plt.figure(figsize=(14, 5))

#Histogram
plt.subplot(1, 2, 1)
sbn.distplot(df['count'], bins=10)

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

plt.show()
```

```
C:\Users\Hp\AppData\Local\Temp\ipykernel_4136\2347940702.py:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sbn.distplot(df['count'], bins=10)
```





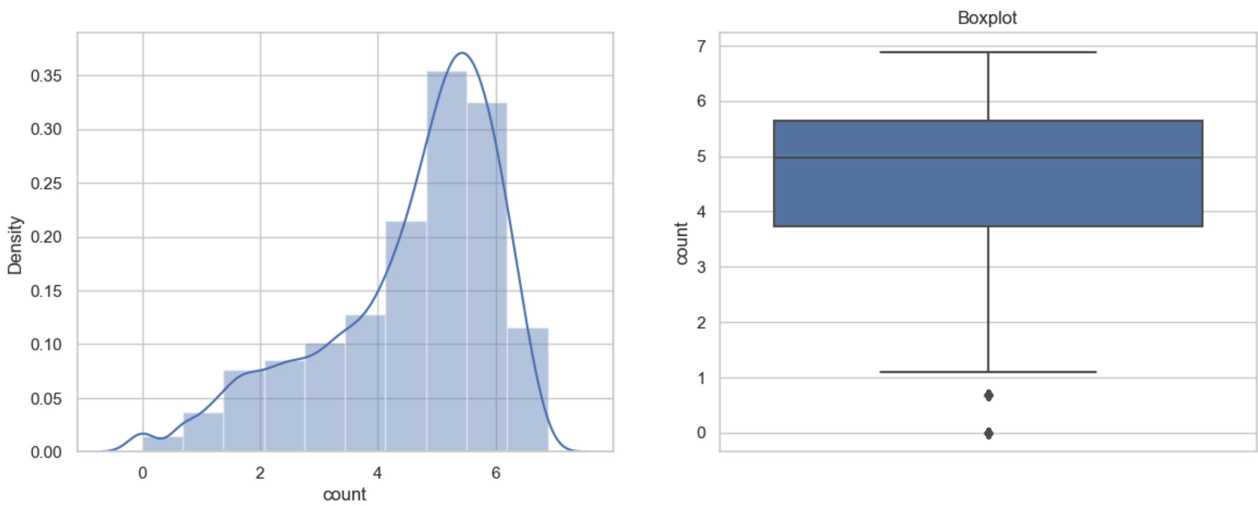
We can see that outliers are present in the given columns. We need to figure out a way to deal with them before starting with the tests.

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 [38]:
```
# Checking distribution after applying log transformation -
plt.figure(figsize=(14, 5))

#Histogram
plt.subplot(1, 2, 1)
sns.distplot(np.log(df['count']), bins=10)

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

plt.show()
```

```
C:\Users\Hp\AppData\Local\Temp\ipykernel_4136\3791880922.py:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750
bbe5751)

  sns.distplot(np.log(df['count']), bins=10)
```

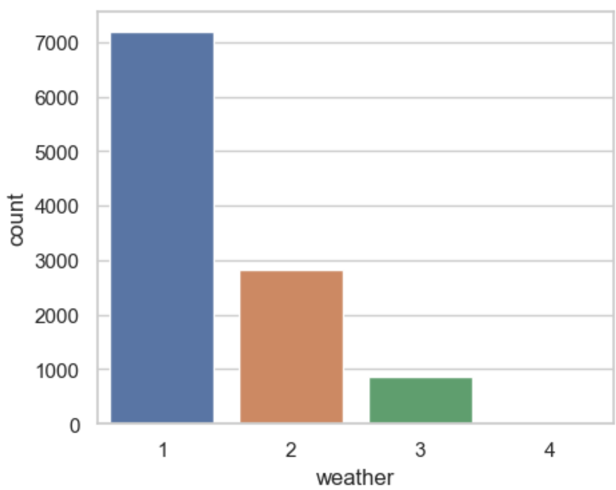


Relation between the dependent and independent variable (Dependent “Count” & Independent: Workingday, Weather, Season etc)

In [ ]:

In [44]:
```
plt.figure(figsize=(5,4))
sns.countplot(x='weather',data=df)
plt.ylabel('count')
```

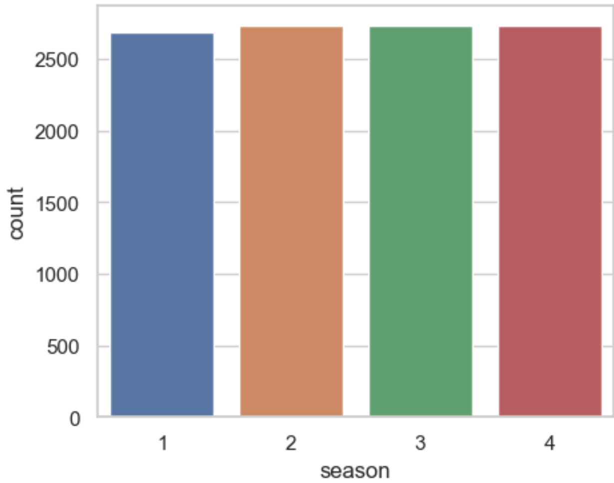
Out[44]: `Text(0, 0.5, 'count')`



In [46]:
```
#1: Clear, Few clouds, partly cloudy, partly cloudy
#2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
#3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
#4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

#Above graph shows that the number of rides are more when it is weather 1. and no bookings in 4.
```

In [47]:
```
plt.figure(figsize=(5,4))
sns.countplot(x='season',data=df)
plt.ylabel('count')
```

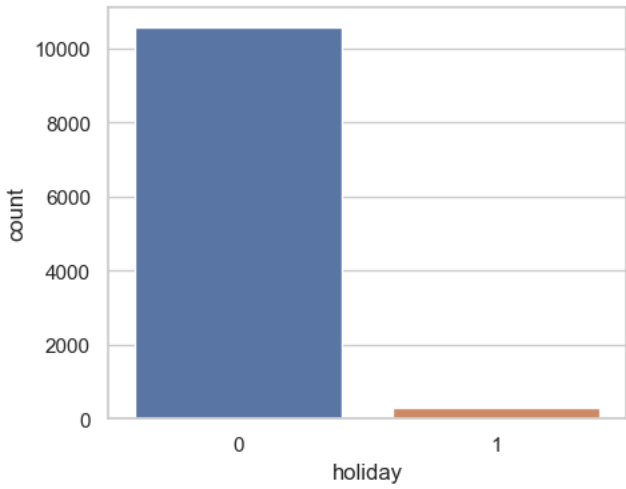
Out[47]: Text(0, 0.5, 'count')



In all the season there is almost equla number of bookings

In [48]:
```
plt.figure(figsize=(5,4))
sns.countplot(x='holiday',data=df)
plt.ylabel('count')
```

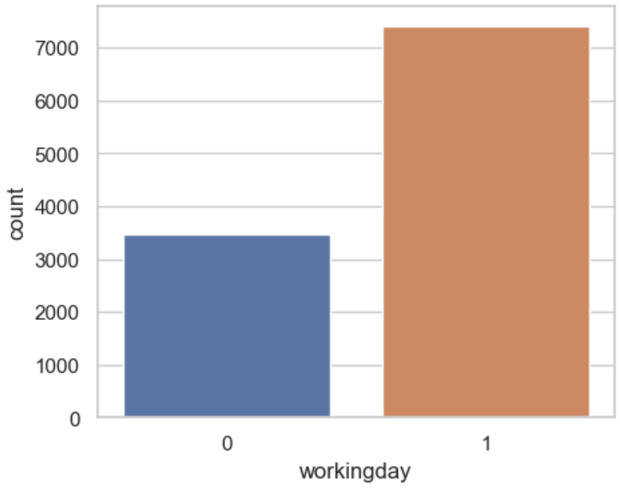
Out[48]: Text(0, 0.5, 'count')



The count of booking is more during no holiday and hence booking is done to commute to work

In [49]:
```
plt.figure(figsize=(5,4))
sns.countplot(x='workingday',data=df)
plt.ylabel('count')
```

Out[49]: Text(0, 0.5, 'count')



In [50]:
```
#if day is neither weekend nor holiday is 1, otherwise is 0
# hence booking  is done to commute to work
```

Select an appropriate test to check whether:

Working Day has effect on number of electric cycles rented

Step 1: Define the null and alternate hypothesis

H0: The demand of bikes on weekdays is greater or similar to the demand of bikes on weekend.

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

Let $\mu 1$ and $\mu 2$ be the average no. of bikes rented on weekdays and weekends respectively.

Mathematically, the above formulated hypothesis can be written as:

$H0: \mu 1 >= \mu 2$

$Ha: \mu 1 < \mu 2$

Step 2: Since the standard deviation of the poulation is not known we use T test

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

In [52]:
```
alpha = 0.05
```

Step 4: Calculate the p-value

In [55]:
```
weekday = df[df['workingday'] == 1]['count']
weekend = df[df['workingday'] == 0]['count']
```

In [57]:
```
test_stat, p_value = ttest_ind(weekday, weekend, equal_var=False, alternative='less')
print('The p-value is : ', p_value)

print(p_value, alpha)
```

```
The p-value is :  0.8917984385965245
0.8917984385965245 0.05
```

As the p-value 0.891 is greater than the level of significance, 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.

No. of cycles rented similar or different in different seasons

Step 1: Define the null and alternate hypothesis

H0: The demand of bikes on regular days is greater or similar to the demand of bikes on holidays.

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

Let μ1 and μ2 be the average no. of bikes rented on regular days and holidays respectively.

Mathematically, the above formulated hypothesis can be written as:

H0:μ1>=μ2

Ha:μ1<μ2

Step 2: Select an appropriate test

the standard deviation of the population is not known

In [59]:
```
holiday = df[df['holiday'] == 1]['count']
regular = df[df['holiday'] == 0]['count']
```

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 [60]:
```
test_stat, p_value = ttest_ind(regular, holiday, equal_var=False, alternative='less')
print('The p-value is : ', p_value)

print(p_value, alpha)
```

```
The p-value is :  0.7269345033197261
0.7269345033197261 0.05
```

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.

No. of cycles rented similar or different in different weather

Step 1: Define the null and alternate hypothesis H0: The average no. of bike rides in different weather conditions are equal.

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

Let μ1 and μ2 be the average no. of bikes rented on weekdays and weekends respectively.

Step 2: Select an appropriate test

In [61]:
```
w1 = df[df['weather'] == 1]['count']
w2 = df[df['weather'] == 2]['count']
w3 = df[df['weather'] == 3]['count']
```

In [62]:
```
df.groupby(['weather'])['count'].describe()
```

Out[62]:

weather	count	mean	std	min	25%	50%	75%	max
1	7192.0	205.236791	187.959566	1.0	48.0	161.0	305.0	977.0
2	2834.0	178.955540	168.366413	1.0	41.0	134.0	264.0	890.0
3	859.0	118.846333	138.581297	1.0	23.0	71.0	161.0	891.0
4	1.0	164.000000	NaN	164.0	164.0	164.0	164.0	164.0

we can drop weather 4 as there is no bike rides during rainy season

There are three independent population means.Hence 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.

Test to check the normality

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 [66]:
```
# Assumption 1: Normality

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

print(p_value, alpha)
```

```
The p-value is :  0.0
0.0 0.05
```

As the p-value 0.0 is less than the level of significance, we reject the null hypothesis.

Equality of variance of the data set

Levene's test - We will test the null hypothesis

H0 : All the count variances are equal

against the alternative hypothesis

Ha : At least one variance is different from the rest

In [67]:
```
#Assumption 2: Homogeneity of Variance

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

print(p_value, alpha)
```

```
The p-value is :  6.198278710731511e-36
6.198278710731511e-36 0.05
```

As the p-value 6.1915385128947197e-36is less than the level of significance, we reject the null hypothesis.

Both the assumptions are true, hence ANOVA is used

In [68]:
```
test_stat, p_value = f_oneway(w1, w2, w3)
print('The p-value is : ', p_value)

print(p_value, alpha)
```

```
The p-value is :  4.976448509904196e-43
4.976448509904196e-43 0.05
```

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.

Weather is dependent on season (check between 2 predictor variable)

Step 1: Define the null and alternate hypothesis H0: The average no. of bike rides in different seasons are equal.

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

Step 2: Select an appropriate test

In [69]:
```
s1 = df[df['season'] == 1]['count'].sample(2399)
s2 = df[df['season'] == 2]['count'].sample(2399)
s3 = df[df['season'] == 3]['count'].sample(2399)
s4 = df[df['season'] == 4]['count'].sample(2399)
```

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 [70]:
```
test_stat, p_value = f_oneway(s1, s2, s3, s4)
print('The p-value is : ', p_value)

print(p_value, alpha)
```

```
The p-value is :  2.1836998545489677e-135
2.1836998545489677e-135 0.05
```

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.

Chi-square test to check if Weather is dependent on the season (10 points)

Step 1: Define the null and alternate hypothesis H0: Weather conditions are independent of the season.

Ha: Weather condition depends on the ongoing season.

Step 2: Select an appropriate test

In [77]:
```
dict1 = {1: 'Sunny',
         2: 'Cloudy',
         3: 'Rainy'}
df['weather_enc'] = df['weather'].map(dict1)
```

In [78]:
```
dict2 = {1: 'Summer',
         2: 'Monsoon',
         3: 'Winter',
         4: 'Autumn'}
df['season_enc'] = df['season'].map(dict2)
```

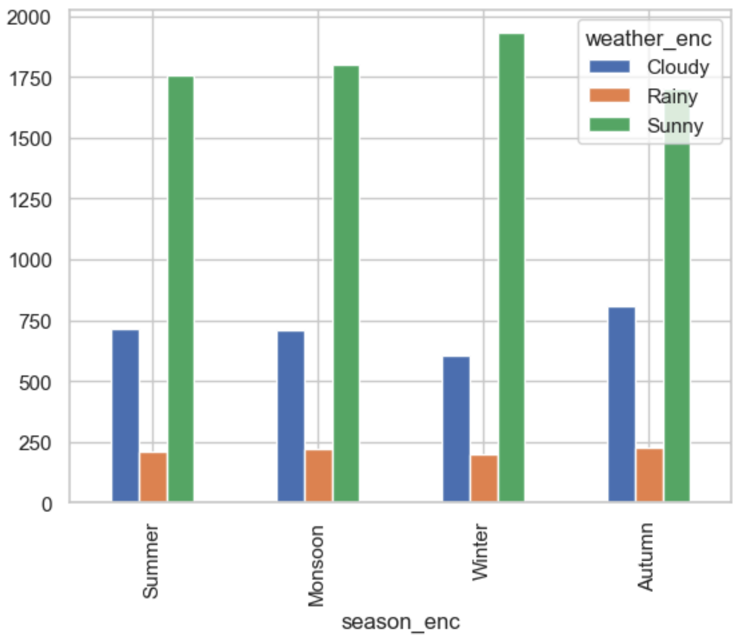
In [79]:
```
contigency= pd.crosstab(df.season_enc, df.weather_enc)
contigency
```

Out[79]:

weather_enc	Cloudy	Rainy	Sunny
season_enc			
Summer	715	211	1759
Monsoon	708	224	1801
Winter	604	199	1930
Autumn	807	225	1702

In [80]:
```
contigency.plot(kind='bar')
```

Out[80]: <Axes: xlabel='season_enc'>



Step 3: Decide the significance level

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

In [81]:
```
pval, dof, exp_freq = chi2_contingency(contigency, correction=False)
'Chi-square Statistic: {} \n P-value: {} \n Degree of Freedom: {} \n Expected Frequencies: {}'.format(chi2, pval, dof, exp_fr
```

```
Chi-square Statistic: 46.1014573107325
 P-value: 2.826001450992924e-08
 Degree of Freedom: 6
 Expected Frequencies: [[ 699.06201194  211.8892972  1774.04869086]
 [ 711.55920992  215.67726229 1805.76352779]
 [ 711.55920992  215.67726229 1805.76352779]
 [ 711.81956821  215.75617823 1806.42425356]]
```

In [82]:
```
print(pval, alpha)
```

```
2.826001450992924e-08 0.05
```

As the p-value 2.8260014509929343e-08 is less than the level of significance, we reject the null hypothesis.

Insights and Recommendations

-Total 10,886 rows were present in the data set. -Neither missing values, nor duplicate rows were found. -'count', 'casual' and 'registered' columns were highly correlated. -Outlier values were found in the 'count' column.

Insights from hypothesis testing - The no. of bikes rented on weekdays is comparatively higher than on weekends. The no. of bikes rented on regular days is comparatively higher than on holidays. The demand of bicycles on rent differs under different weather conditions. The demand of bicycles on rent is different during different seasons. 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 [87]:
```
conda install nbconvert
```

```
  Cell In[87], line 1
    conda install nbconvert
          ^
SyntaxError: invalid syntax
```

In [ ]:
```

```