In [50]:

```
#importing the required packages for the case study visualisation and analysis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm, binom, ttest_1samp, ttest_ind, ttest_rel, f_oneway, c
from scipy.stats import chisquare, shapiro, levene, boxcox, kruskal
from statsmodels.stats.proportion import proportions_ztest
from statsmodels.graphics.gofplots import qqplot
```

In [51]:

```
#reading the dataset to the platform
data = pd.read_csv('bike_sharing.csv')
```

In [52]:

```
#dataset snippet - to understand the general style of data / column Names
data.head()
```

Out[52]:

| | 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 [49]:

```
#get the details of the columnNames along with their datatypes
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 13 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  
 12  log_count   10886 non-null  float64
dtypes: float64(4), int64(8), object(1)
memory usage: 1.1+ MB
```

In [53]:

```
#check whether there is any null data in the dataset
data.isnull().sum()
```

Out[53]:

| | **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

In [54]:

```
#shape of the dataset
data.shape
```

Out[54]:

```
(10886, 12)
```

In [55]:
```
#Does the dataset has any duplicate records ?
data.duplicated().value_counts()
```

Out[55]:

| | count |
|---|---|
| **False** | 10886 |

**dtype:** int64

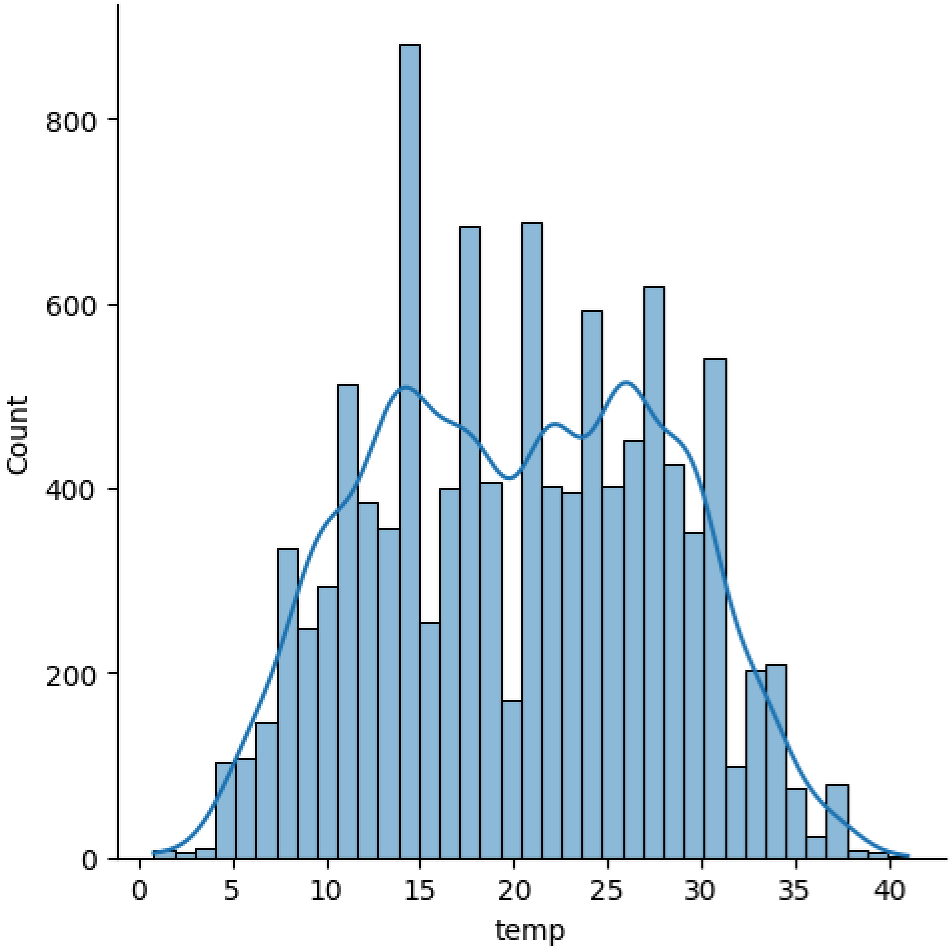
In [56]:
```
#what are the population parameters for the numerical variables?
data.loc[:,'temp':'count'].describe()
```

Out[56]:

| | temp | atemp | humidity | windspeed | casual | registered |
|---|---|---|---|---|---|---|
| **count** | 10886.00000 | 10886.000000 | 10886.000000 | 10886.000000 | 10886.000000 | 10886.000000 |
| **mean** | 20.23086 | 23.655084 | 61.886460 | 12.799395 | 36.021955 | 155.552177 |
| **std** | 7.79159 | 8.474601 | 19.245033 | 8.164537 | 49.960477 | 151.039033 |
| **min** | 0.82000 | 0.760000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 13.94000 | 16.665000 | 47.000000 | 7.001500 | 4.000000 | 36.000000 |
| **50%** | 20.50000 | 24.240000 | 62.000000 | 12.998000 | 17.000000 | 118.000000 |
| **75%** | 26.24000 | 31.060000 | 77.000000 | 16.997900 | 49.000000 | 222.000000 |
| **max** | 41.00000 | 45.455000 | 100.000000 | 56.996900 | 367.000000 | 886.000000 |

In [58]:
```
#analyse the distribution of numerical variable - temperature
sns.displot(data=data, x='temp', kde=True)
```

Out[58]:
```
<seaborn.axisgrid.FacetGrid at 0x7db58b6a7310>
```

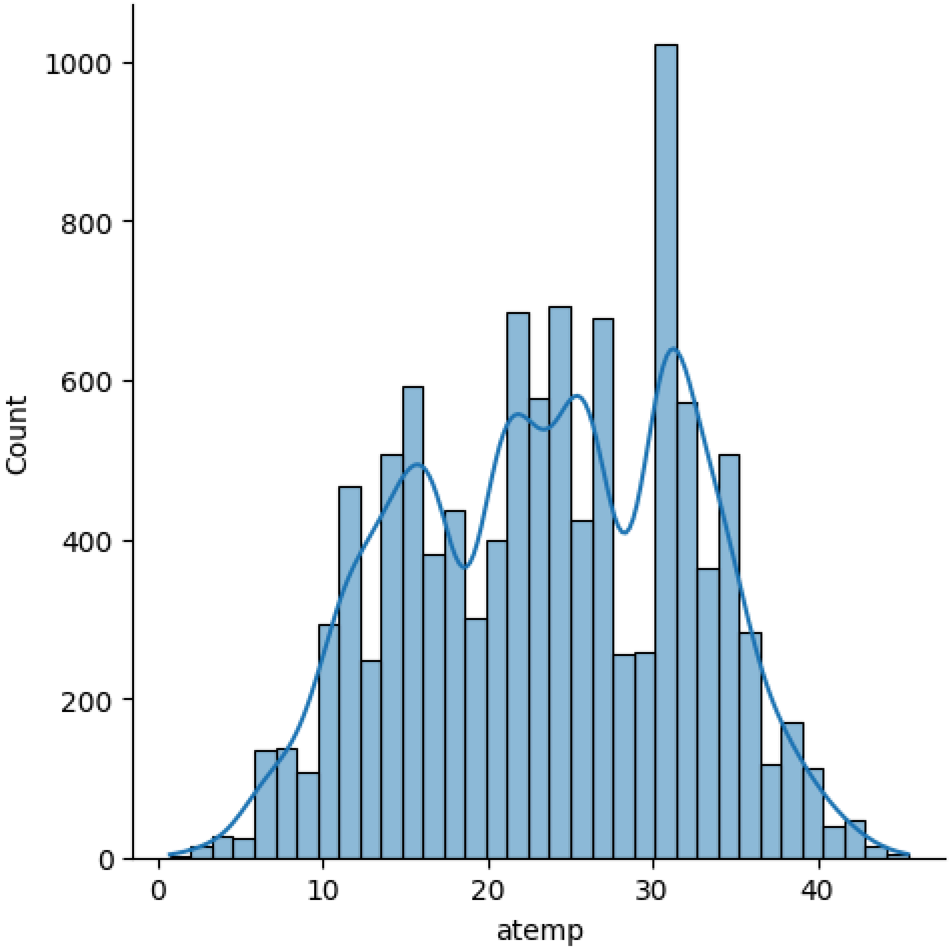


In [57]:

```
#analyse the distribution of numerical variable - feel factor temperature
  sns.displot(data=data, x='atemp', kde=True)
```

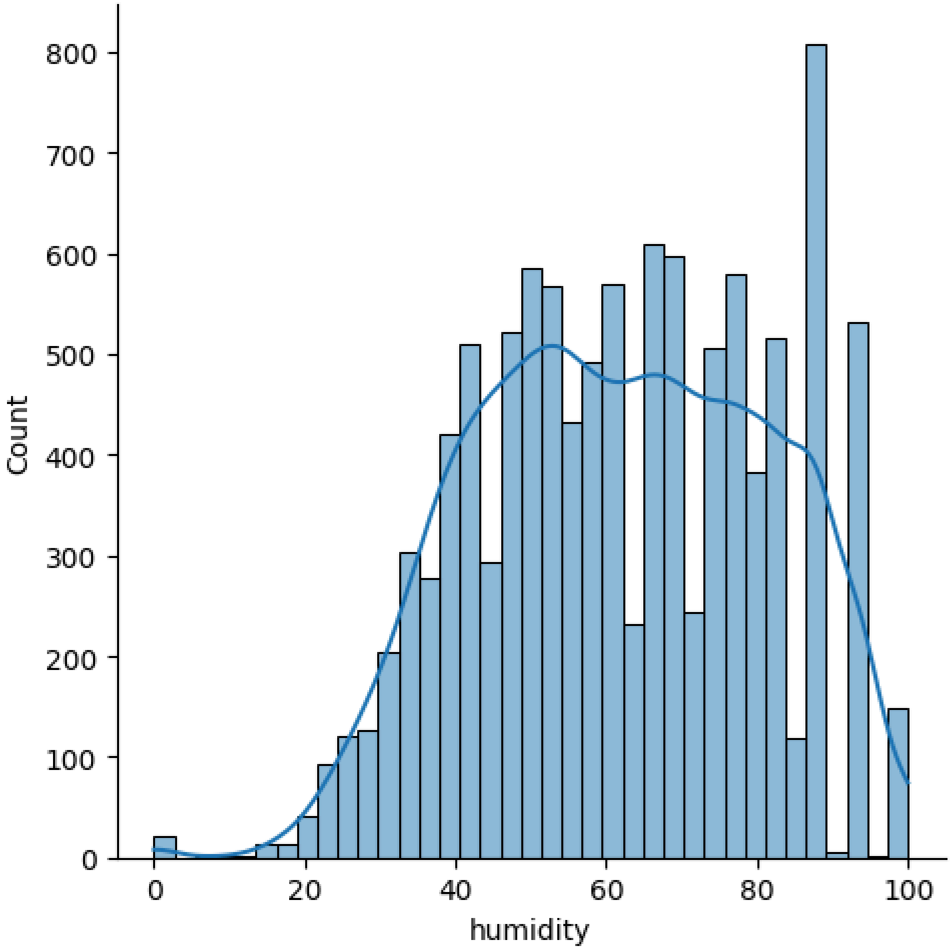
Out[57]:

```
<seaborn.axisgrid.FacetGrid at 0x7db5862c6fe0>
```



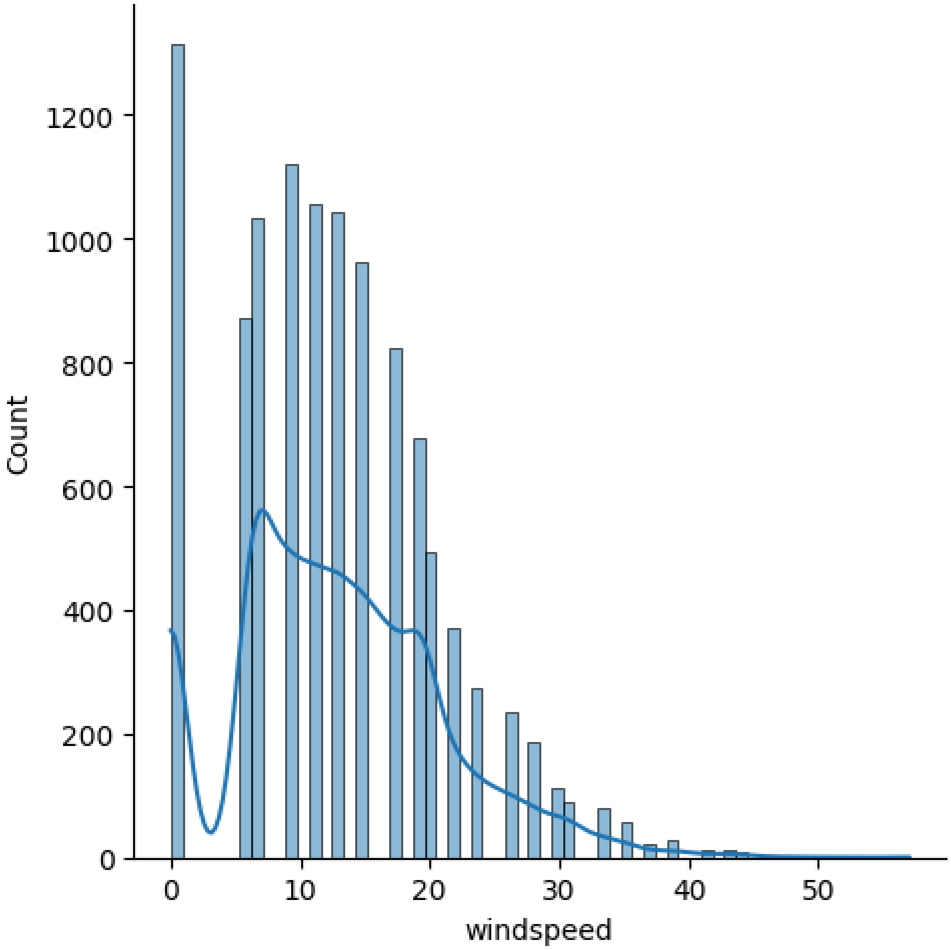
```
In [59]: #analyse the distribution of numerical variable - humidity
           sns.displot(data=data, x='humidity', kde=True)
```

```
Out[59]: <seaborn.axisgrid.FacetGrid at 0x7db58476c580>
```



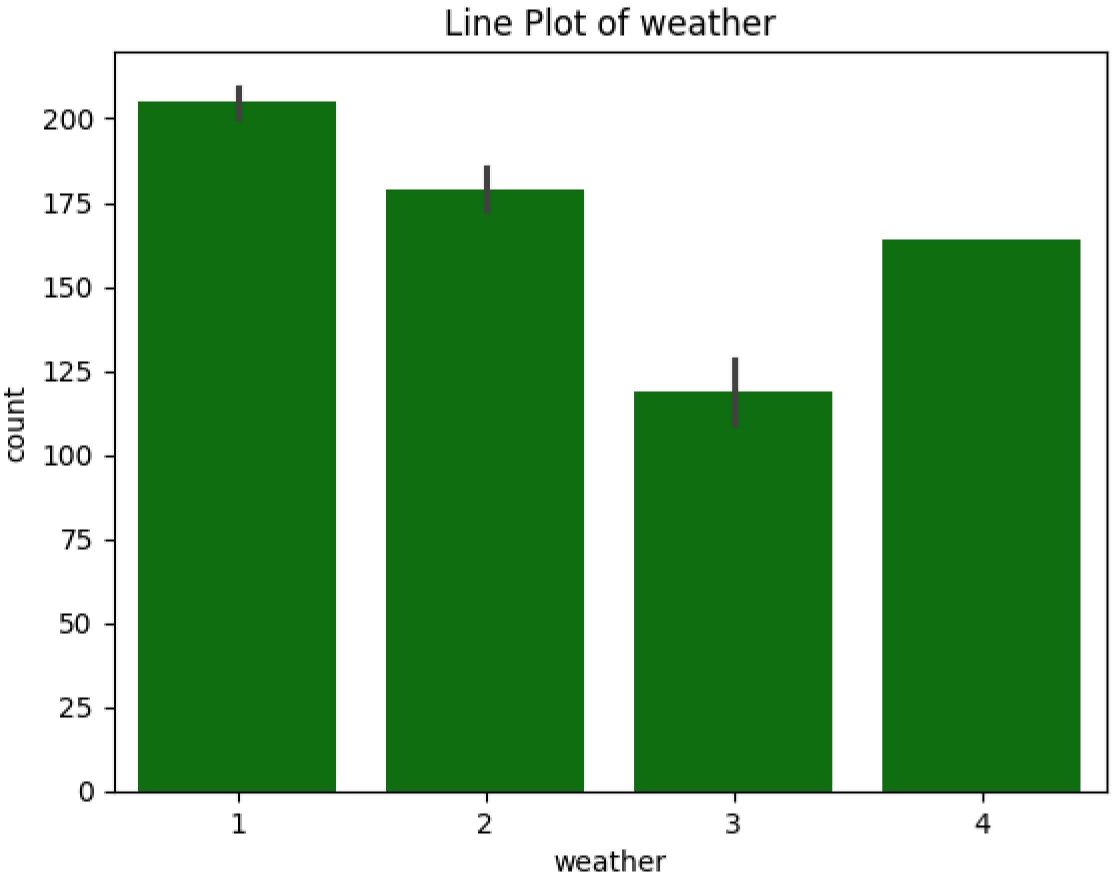
In [60]:
```
#analyse the distribution of numerical variable - windspeed
  sns.displot(data=data, x='windspeed', kde=True)
```

Out[60]:
```
<seaborn.axisgrid.FacetGrid at 0x7db5854d4250>
```

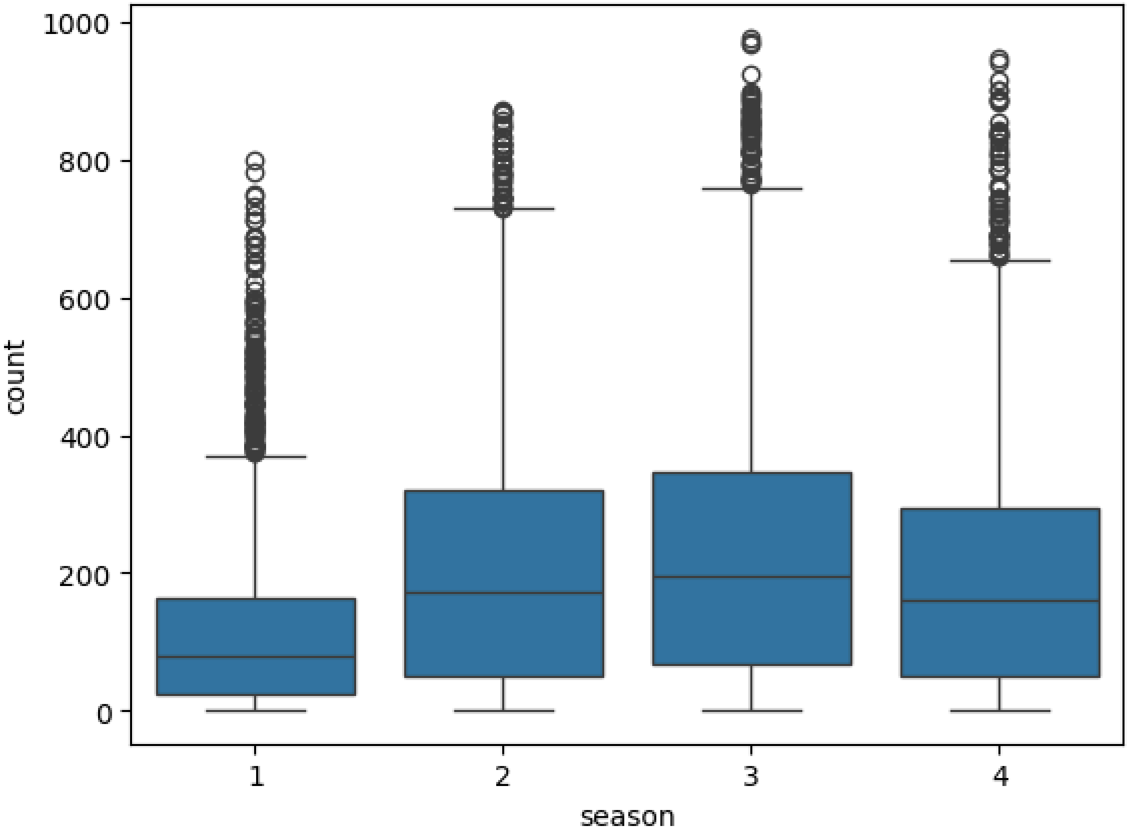


In [ ]:

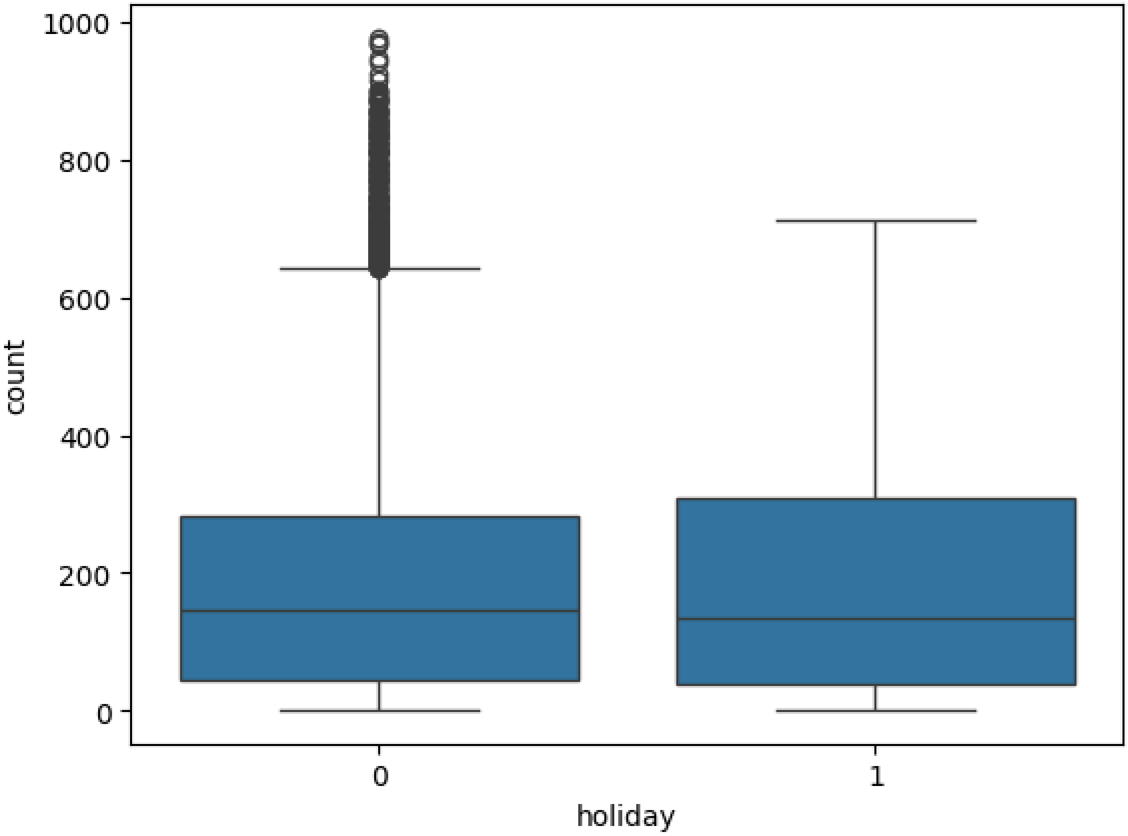
In [61]:
```
sns.barplot(x='weather', y='count', data=data, color='g')

plt.title("Line Plot of weather")
plt.show()
```



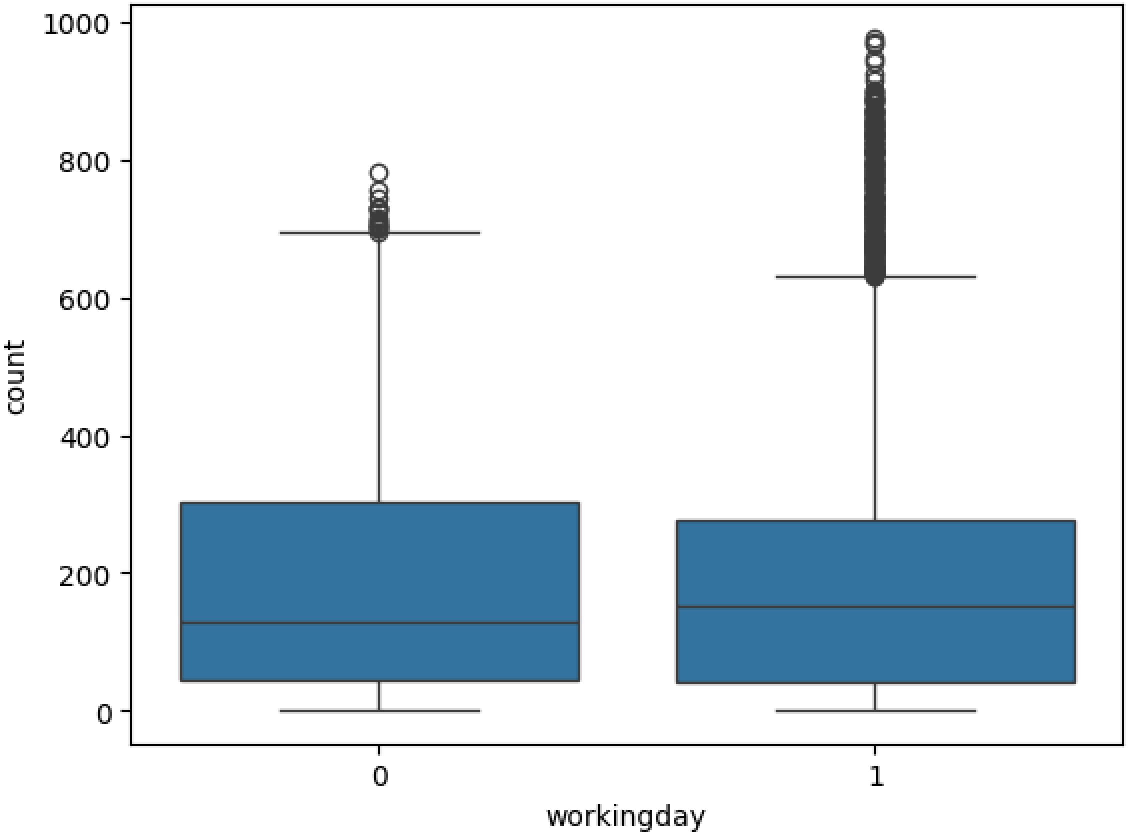
In [62]:
```
#Visualisation of relationship of categorical variable - Season
sns.boxplot(data=data, x='season', y='count')
plt.show()
```



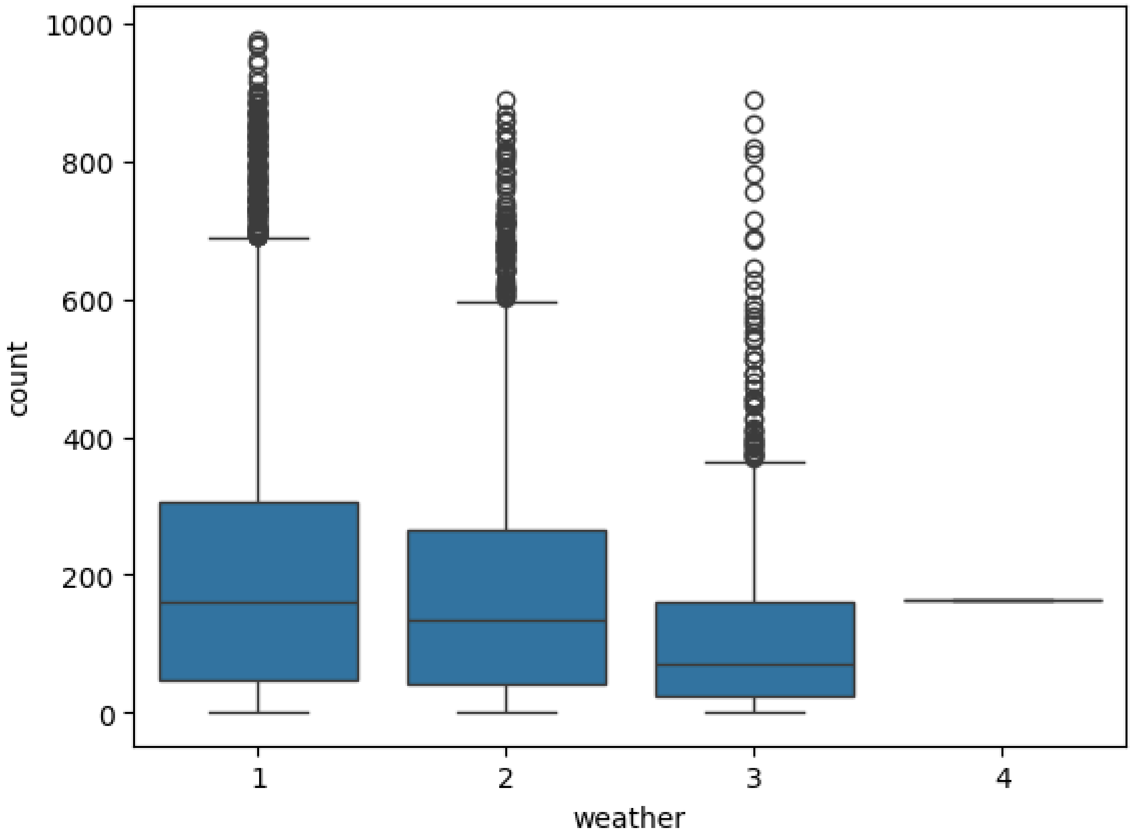
In [63]:
```
#Visualisation of relationship of categorical variable - Holiday
sns.boxplot(x='holiday', data=data, y='count')
plt.show()
```



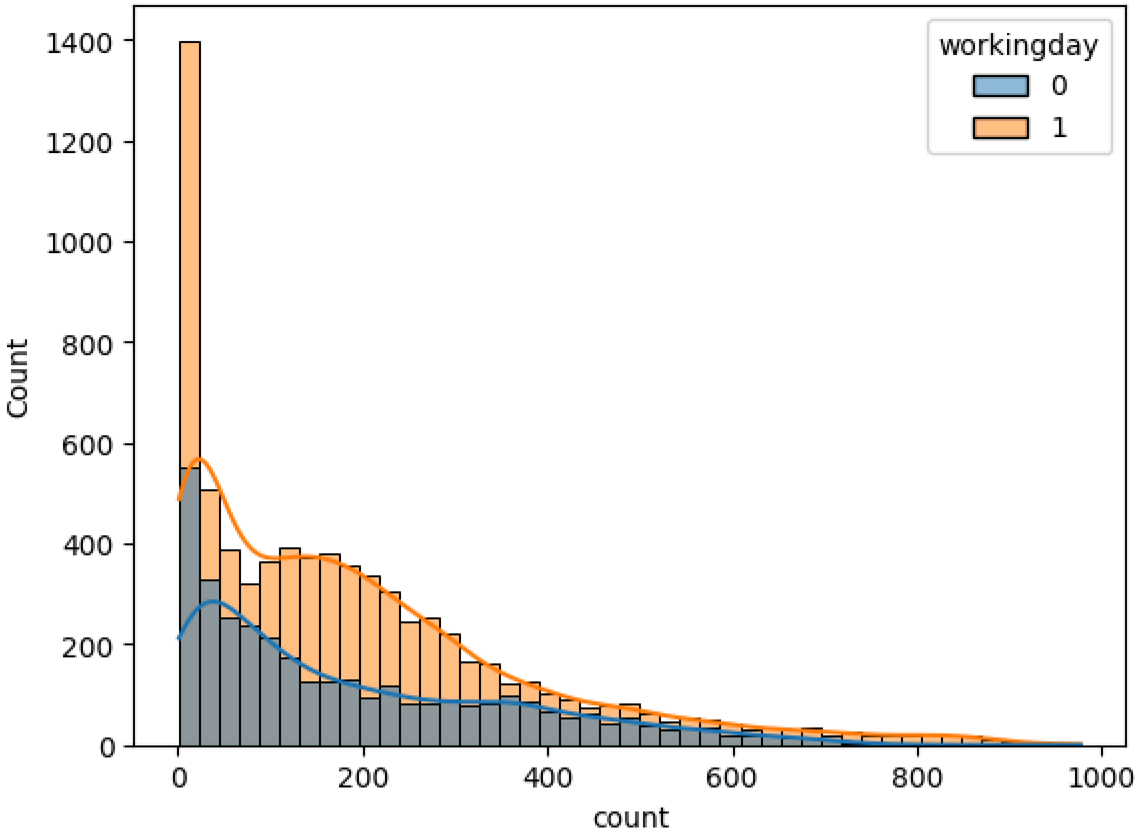
In [64]:
```
#Visualisation of relationship of categorical variable - Working day
sns.boxplot(x='workingday', data=data, y='count')
plt.show()
```



In [66]:
```
#Visualisation of relationship of categorical variable - weather
sns.boxplot(x='weather', data=data, y='count')
plt.show()
```

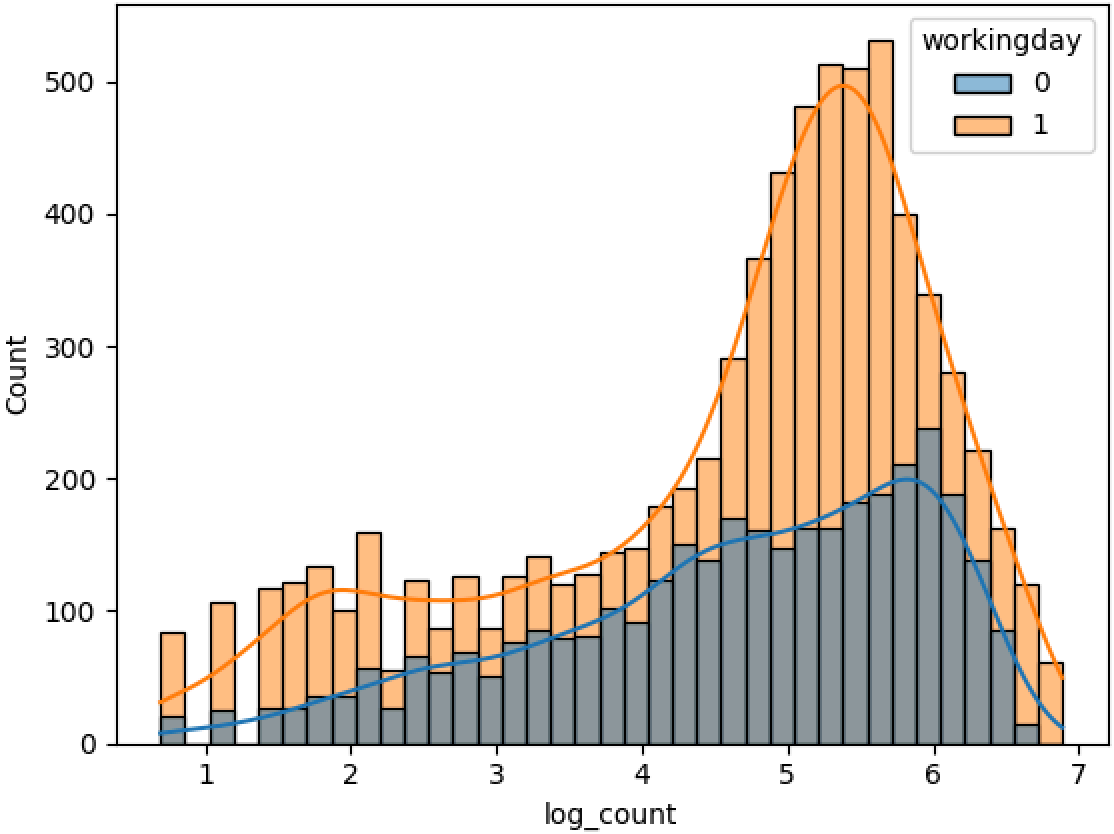


In [65]:
```
#Check if there is any significant difference between the number of bike rides on w
sns.histplot(data=data, x='count', hue='workingday', kde=True)
plt.show()
```



In [41]:

```
#making the above distribution as log normal distribution
data['log_count'] = np.log(data['count'] + 1) #adding 1 to avoid log(0)

#Create the histogram with log-transformed data
sns.histplot(data=data, x='log_count', hue='workingday', kde=True)
plt.show()
```



In [42]:

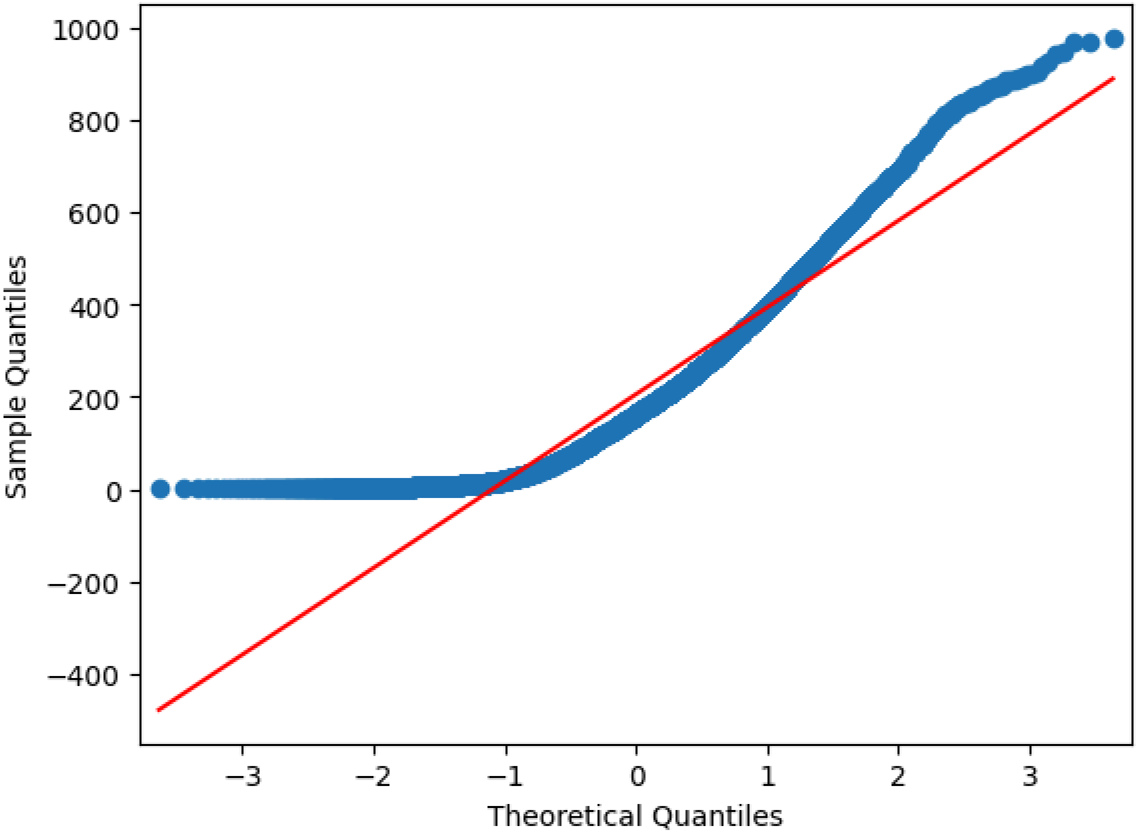
```
#- Finding t statistics and p-value as to reject or accept null hypothesis using tw
alpha = 0.05
working_day_count = data[data['workingday'] == 1]['log_count']
non_working_day_count = data[data['workingday'] == 0]['log_count']

t_stat, p_value = ttest_ind(working_day_count, non_working_day_count)

print("T-statistic:", t_stat)
print("P-value:", p_value)

if alpha > p_value:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

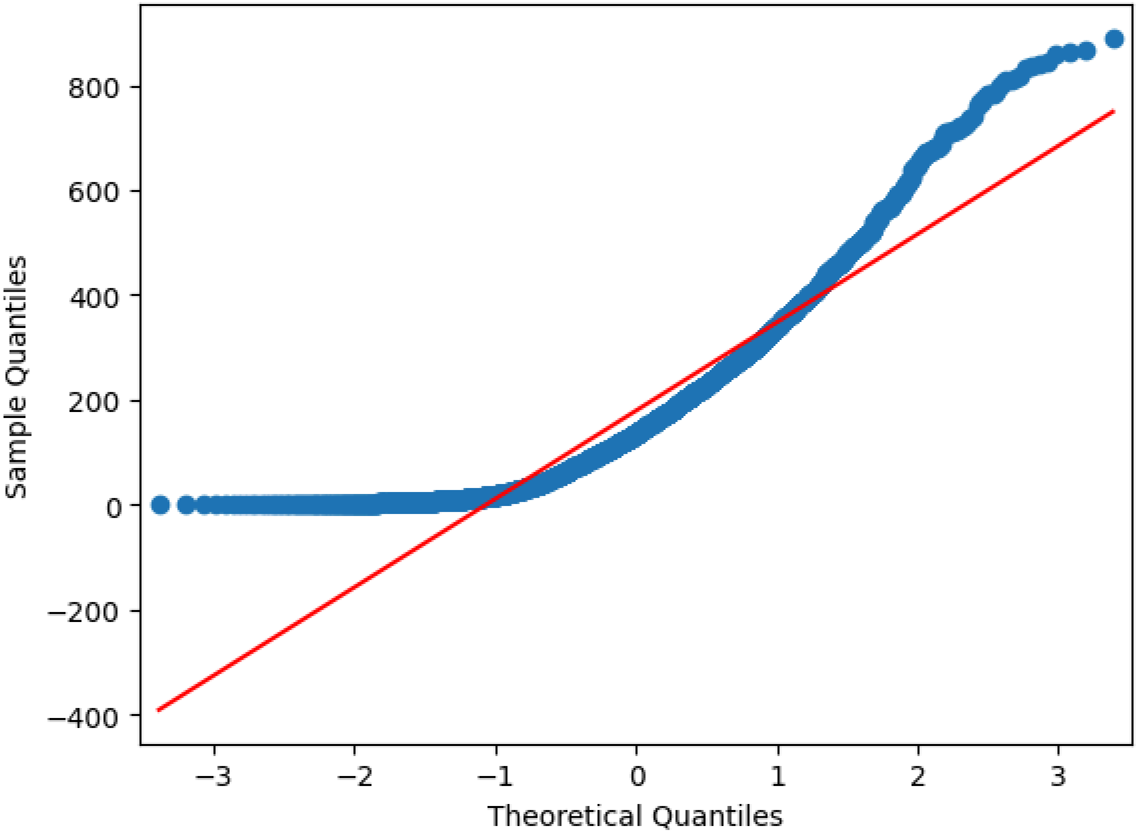
```
T-statistic: -1.5991402409331144
P-value: 0.10981847102316886
Fail to reject the null hypothesis
```

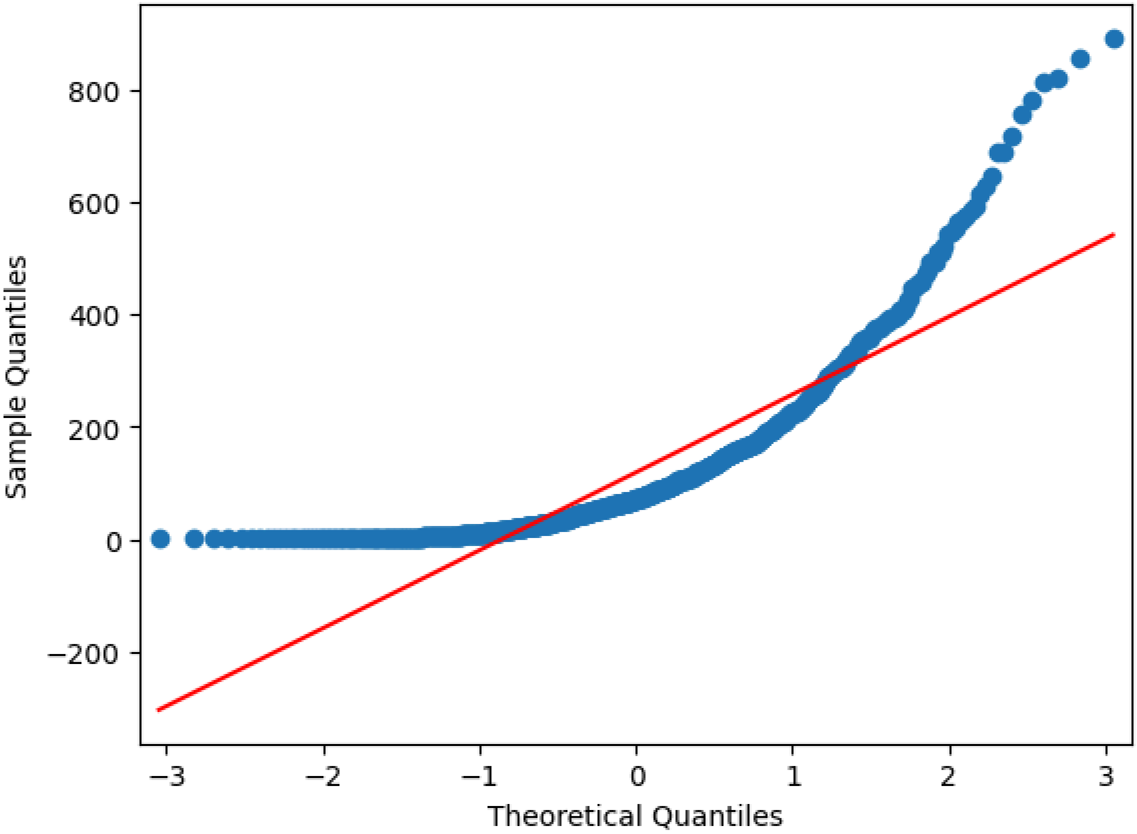
In [43]:

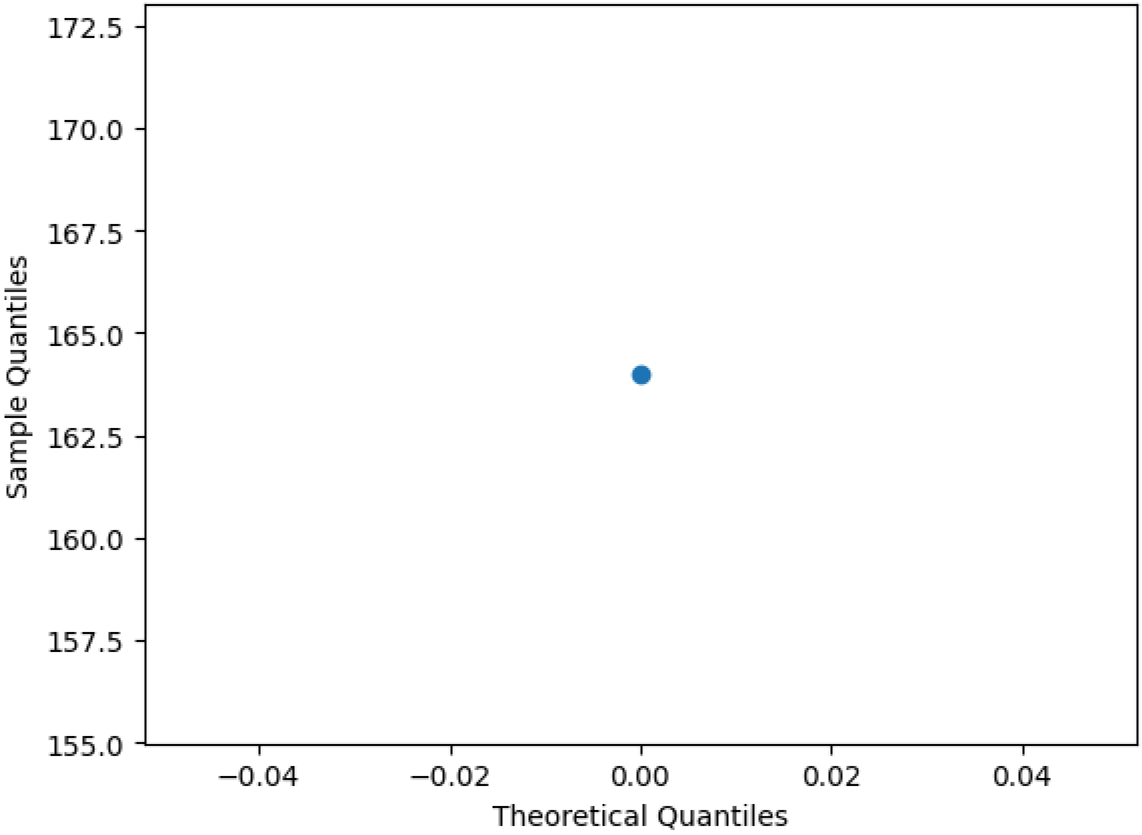
```
# Check if the demand of bicycles on rent is the same for different weather conditi

#Check for QQPlot for normal distribution
```

```
qqplot(data[data['weather'] == 1 ]['count'], line='s')
qqplot(data[data['weather'] == 2 ]['count'], line='s')
qqplot(data[data['weather'] == 3 ]['count'], line='s')
qqplot(data[data['weather'] == 4 ]['count'], line='s')


plt.show()
```




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


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



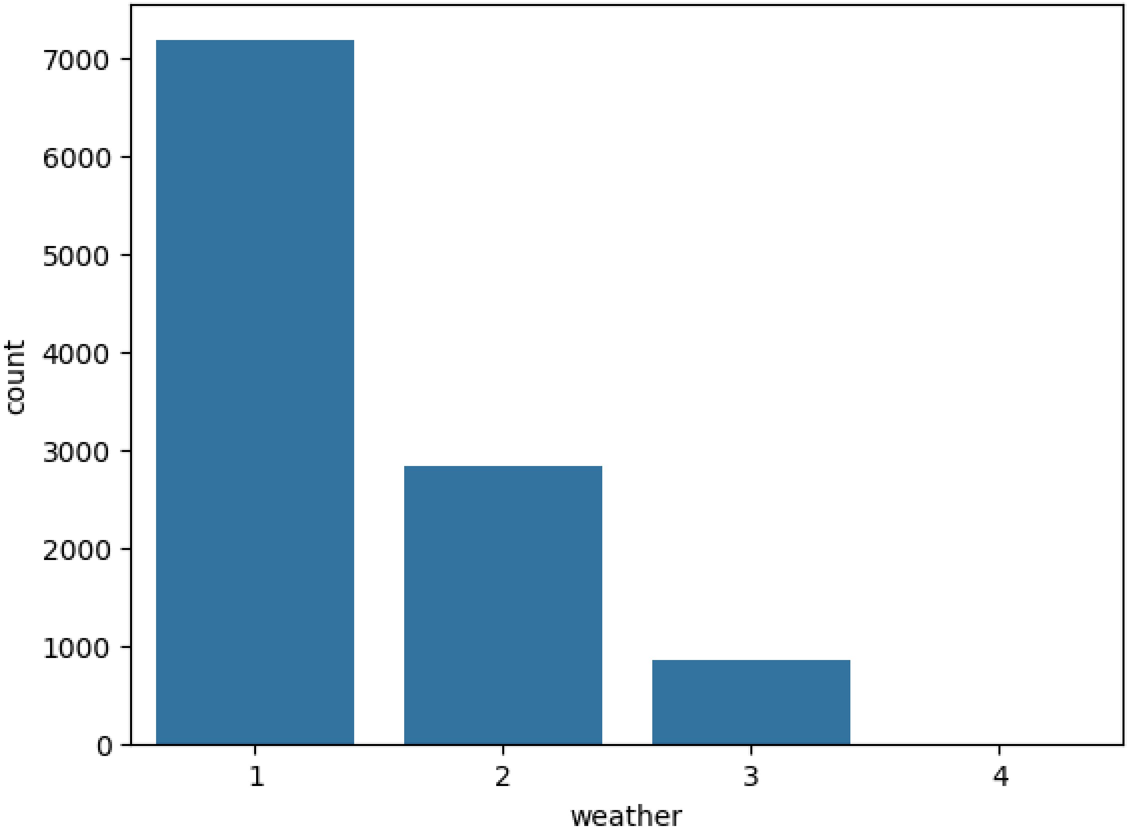
In [44]:
```
# check whether the variance is same across the different groups / weather
group1 = data[data['weather'] == 1]['log_count']
group2 = data[data['weather'] == 2]['log_count']
group3 = data[data['weather'] == 3]['log_count']
group4 = data[data['weather'] == 4]['log_count']
levene_stat, p_value = levene(group1, group2, group3, group4)

print("Levene's statistic:", levene_stat)
print("p-value:", p_value)

if p_value < alpha:
    print("Reject null hypothesis: There is a significant variance among various gr
else:
    print("Fail to reject null hypothesis: There is no significant dvariance among
```

```
Levene's statistic: 0.7152137286881602
p-value: 0.5427571379087562
Fail to reject null hypothesis: There is no significant dvariance among various grou
ps
```

In [45]:
```
#using kruskal wallis test to find the result since ANOVA cannot be used.
stat,p_value = kruskal(group1, group2, group3, group4)

print("p-value:", p_value)

if p_value < alpha:
    print("Reject null hypothesis: There is a significant difference in bike rental
else:
    print("Fail to reject null hypothesis: There is no significant difference in bi
p-value: 3.501611300708679e-44
Reject null hypothesis: There is a significant difference in bike rental demand acro
ss weather conditions.
```

In [76]:

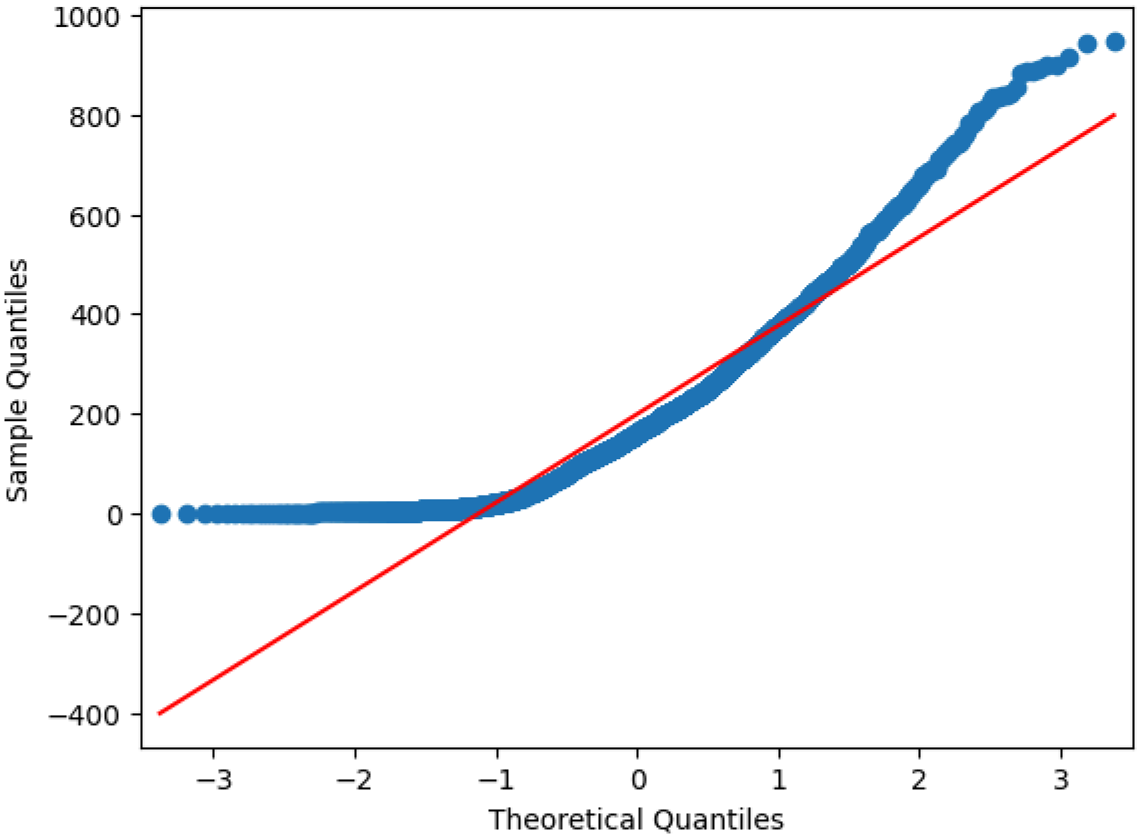
```
#Which weather type has the most number of bike rentals ?
sns.countplot(data=data, x='weather')
plt.show()
```

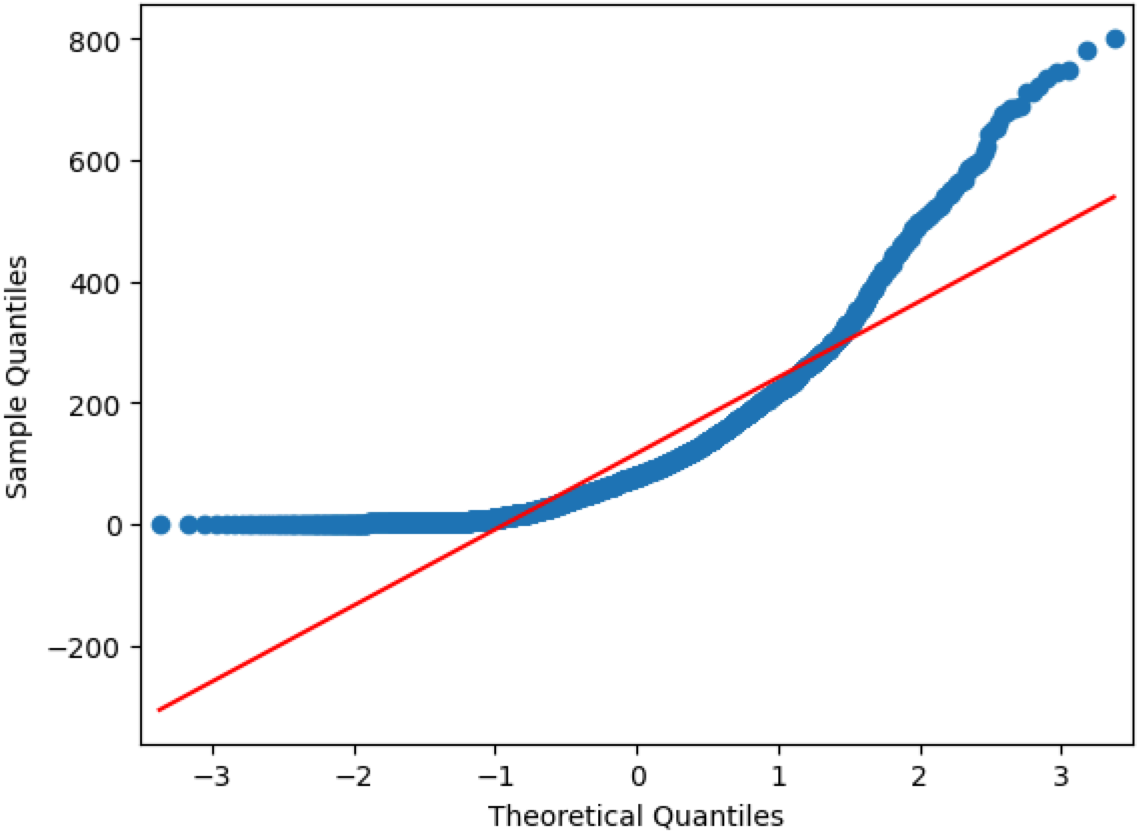


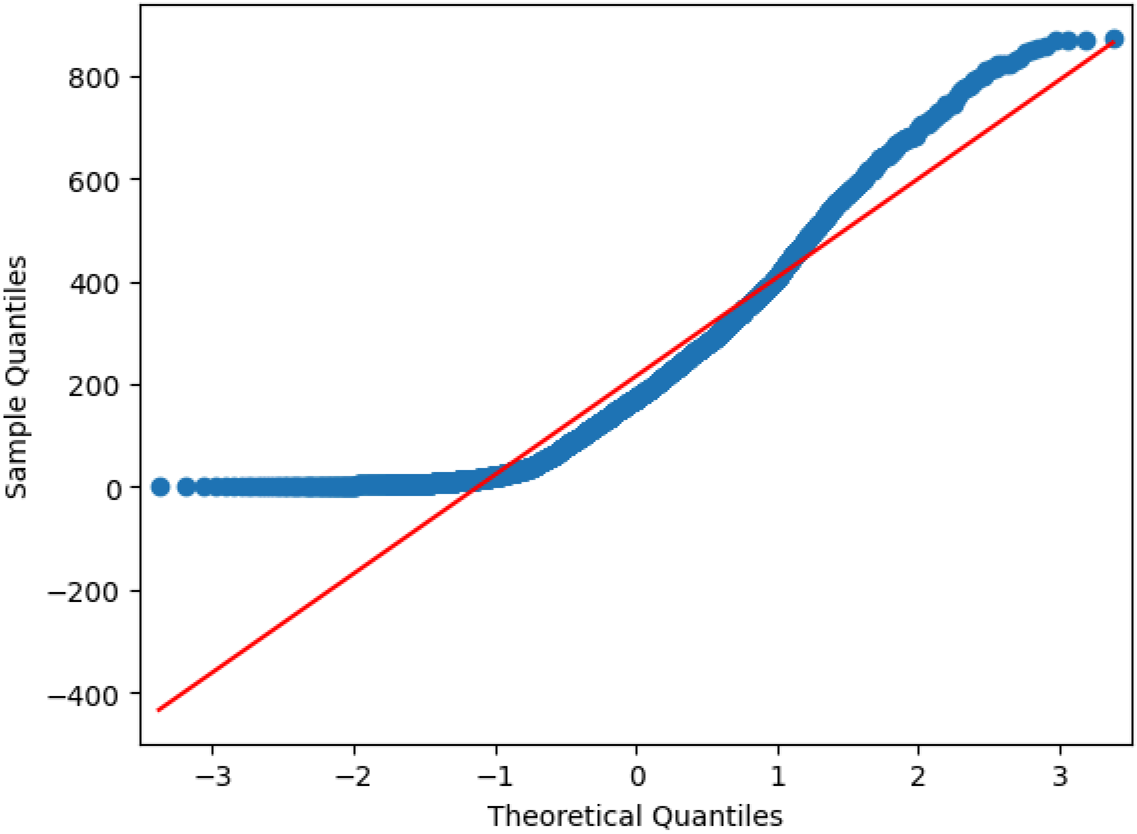
In [46]:

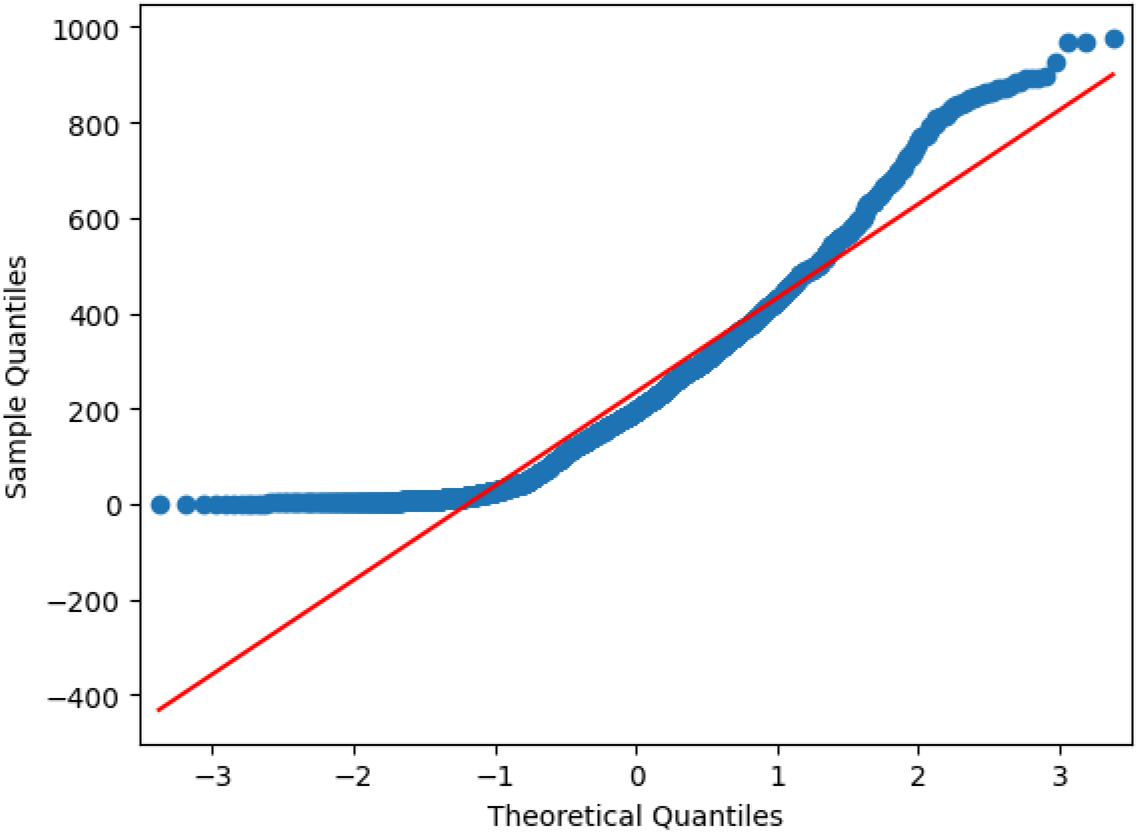
```
# Q) Check if the demand of bicycles on rent is the same for different seasons ?

#Check for QQPlot for normal distribution
qqplot(data[data['season'] == 1 ]['count'], line='s')
qqplot(data[data['season'] == 2 ]['count'], line='s')
qqplot(data[data['season'] == 3 ]['count'], line='s')
qqplot(data[data['season'] == 4 ]['count'], line='s')
```

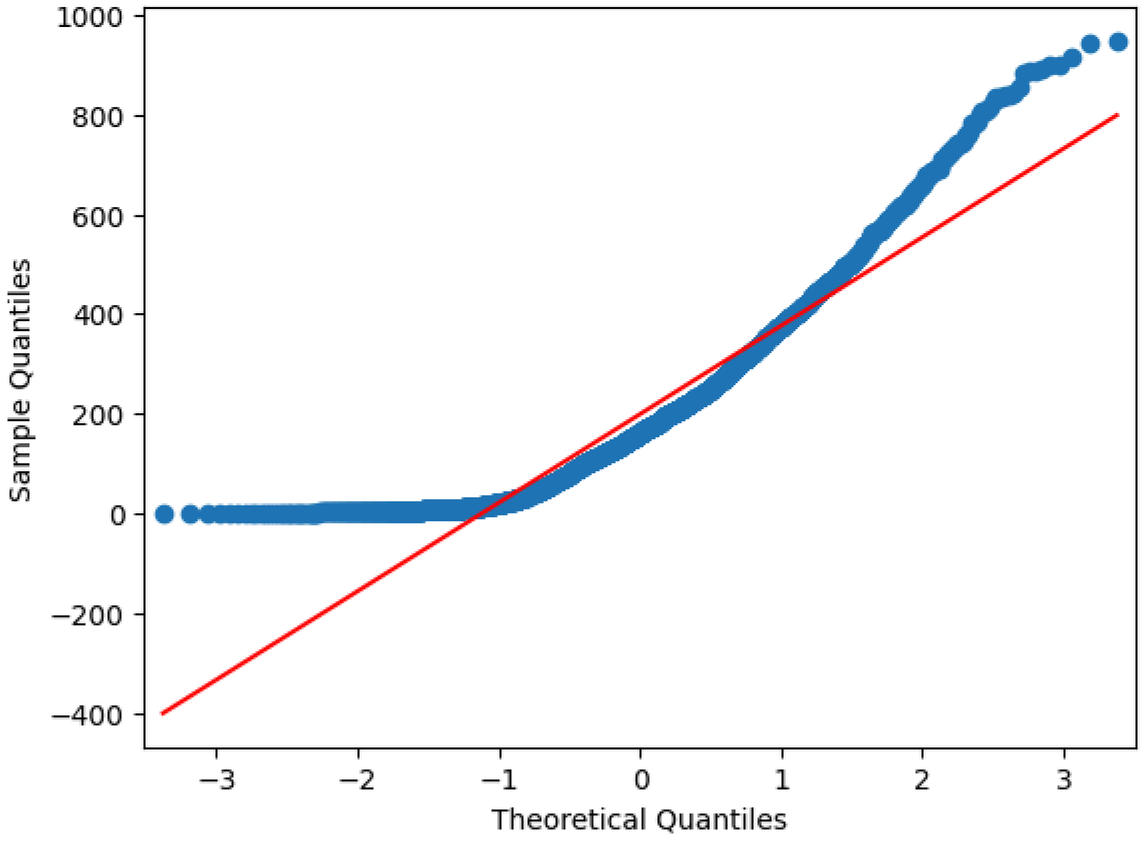
Out[46]:


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


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


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


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



In [67]:
```
#convert the distribution to normal using boxcox transformation
transformed_data, lambda_value = boxcox(data['count'])
data['transformed_count'] = transformed_data
```

In [70]:
```
# check whether the variance is same across the different groups / seasons
group1 = data[data['season'] == 1]['transformed_count']
group2 = data[data['season'] == 2]['transformed_count']
group3 = data[data['season'] == 3]['transformed_count']
group4 = data[data['season'] == 4]['transformed_count']
levene_stat, p_value = levene(group1, group2, group3, group4)

print("Levene's statistic:", levene_stat)
print("p-value:", p_value)

if p_value < alpha:
    print("Reject null hypothesis: There is a significant variance among various gr
else:
    print("Fail to reject null hypothesis: There is no significant dvariance among
```

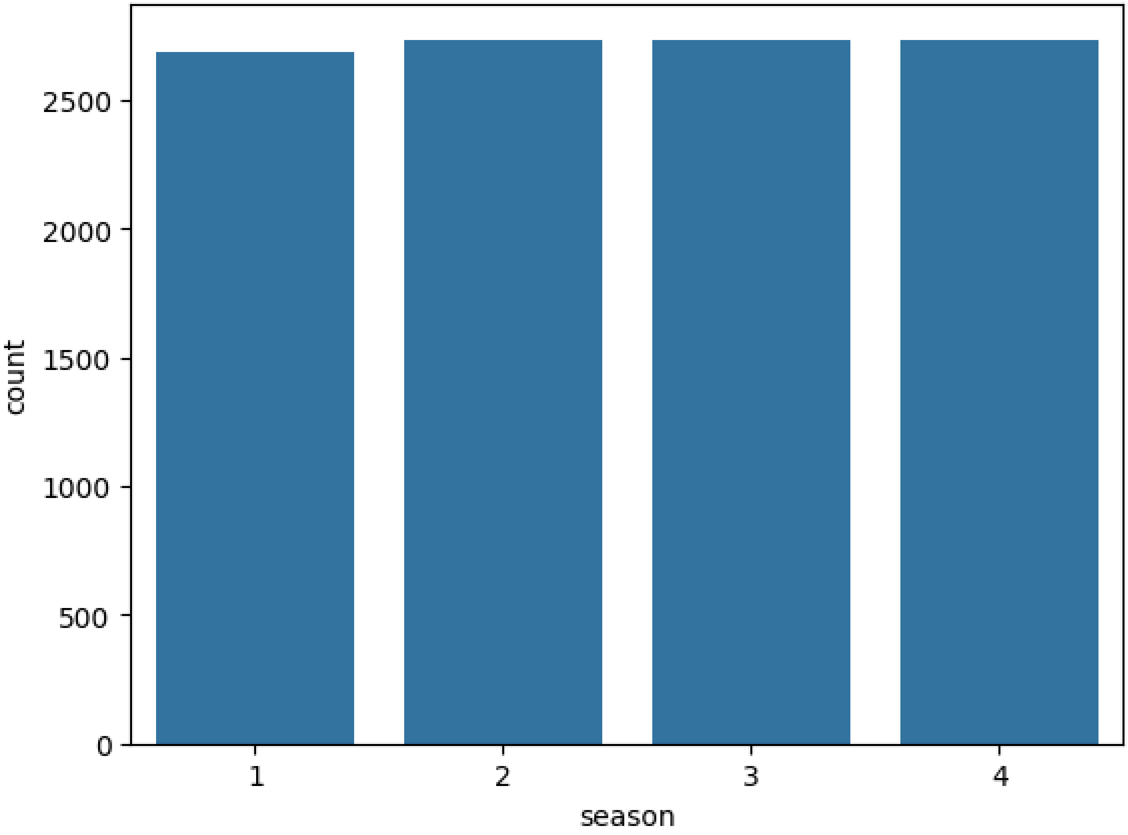
```
Levene's statistic: 21.910242580189703
p-value: 3.852090330184752e-14
Reject null hypothesis: There is a significant variance among various groups.
```

In [69]:
```
#using kruskal wallis test to find the result since ANOVA cannot be used. Seasons -
stat,p_value = kruskal(group1, group2, group3, group4)

print("p-value:", p_value)

if p_value < alpha:
    print("Reject null hypothesis: There is a significant difference in bike rental
else:
    print("Fail to reject null hypothesis: There is no significant difference in bi
```

```
p-value: 2.479008372608633e-151
Reject null hypothesis: There is a significant difference in bike rental demand acro
ss differnent seasons.
```

In [75]:

```
#Which seasons has the most number of bike rentals ?
sns.countplot(data=data, x='season')
plt.show()
```



In [72]:

```
#  Q) Check if weather conditions are significantly different during different seas

#create cross tab for the two variables
observed = pd.crosstab(data['season'], data['weather'])
observed
```

Out[72]:

| 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 [78]:

```
# apply the chi sqaure test
chistat, p_value,df,exp_freq = chi2_contingency(observed)

print("Chi-square statistic:", chistat)
print("p-value:", p_value)
print("Degrees of freedom:", df)

if p_value < 0.05:
  print("Reject null hypothesis: Weather conditions are significantly different dur
else :
  print("Fail to reject null hypothesis: Weather conditions are not significantly d
```

```
Chi-square statistic: 49.158655596893624
p-value: 1.549925073686492e-07
Degrees of freedom: 9
Reject null hypothesis: Weather conditions are significantly different during differ
ent seasons.
```

In [ ]: