Avesh Raza Nagauri

aveshnagauri5@gmail.com (mailto:aveshnagauri5@gmail.com) 9082425683

StatisticalRide Dynamics

In [2]:
```
# Importing Libraries

import numpy as np,pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from scipy.stats import f_oneway, ttest_ind
from statsmodels.graphics.gofplots import qqplot
from scipy.stats import shapiro
from scipy.stats import levene
from scipy.stats import kruskal
from scipy.stats import chi2_contingency
```

In [3]:
```
df = pd.read_csv("bike_sharing.csv") # Importing Dataset
```

In [105]:
```
df.head()
```

Out[105]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	ca:
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 [9]:
```
df.shape
```

Out[9]:
```
(10886, 12)
```

There are 10886 rows and 12 columns

In [7]:
```
df.describe()
```

Out[7]:

	season	holiday	workingday	weather	temp	aten
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.0000(
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.6550ε
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.4746(
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.7600(
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.6650(
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.2400(
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.0600(
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.4550(

Some columns might have outliers like temp,atemp,humidity,windpeed,casual,registered and count as their is a significant difference between their mean and max value.

In [8]:
```
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 [16]:
```
df.nunique() # Unique values for each columns
```

Out[16]:
```
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
```

In [5]:
```
df.isna().sum() # Checking for null values
```

Out[5]:
```
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
```

There are no null values in the dataset

In [3]:
```
df['Date'] = pd.to_datetime(df['datetime']).dt.date # extrcating date fr
```

In [4]:
```
df.drop('datetime',axis = 1,inplace = True) #Deleting datetime column
```

In [5]:
```
df.head()
```

Out[5]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regis
0	1	0	0	1	9.84	14.395	81	0.0	3	
1	1	0	0	1	9.02	13.635	80	0.0	8	
2	1	0	0	1	9.02	13.635	80	0.0	5	
3	1	0	0	1	9.84	14.395	75	0.0	3	
4	1	0	0	1	9.84	14.395	75	0.0	0	

In [25]:
```
df['season'].value_counts()
```

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

In [26]:
```
df['holiday'].value_counts()
```

Out[26]:
```
holiday
0    10575
1      311
Name: count, dtype: int64
```

In [27]:
```
df['workingday'].value_counts()
```

Out[27]:
```
workingday
1    7412
0    3474
Name: count, dtype: int64
```

In [28]:
```
df['weather'].value_counts()
```

Out[28]:
```
weather
1    7192
2    2834
3     859
4       1
Name: count, dtype: int64
```

In [29]:
```
df['temp'].value_counts()
```

Out[29]:
```
temp
14.76    467
26.24    453
28.70    427
13.94    413
18.86    406
22.14    403
25.42    403
16.40    400
22.96    395
27.06    394
24.60    390
12.30    385
21.32    362
17.22    356
13.12    356
29.52    353
10.66    332
18.04    328
20.50    327
30.34    299
9.84     294
15.58    255
9.02     248
31.16    242
8.20     229
27.88    224
23.78    203
32.80    202
11.48    181
19.68    170
6.56     146
33.62    130
5.74     107
7.38     106
31.98     98
34.44     80
35.26     76
4.92      60
36.90     46
4.10      44
37.72     34
36.08     23
3.28      11
0.82       7
38.54      7
39.36      6
2.46       5
1.64       2
41.00      1
Name: count, dtype: int64
```

In [30]:
```
df['atemp'].value_counts()
```

Out[30]:
```
atemp
31.060    671
25.760    423
22.725    406
20.455    400
26.515    395
16.665    381
25.000    365
33.335    364
21.210    356
30.305    350
15.150    338
21.970    328
24.240    327
17.425    314
31.820    299
34.850    283
27.275    282
32.575    272
11.365    271
14.395    269
29.545    257
19.695    255
15.910    254
12.880    247
13.635    237
34.090    224
12.120    195
28.790    175
23.485    170
10.605    166
35.605    159
9.850     127
18.180    123
36.365    123
37.120    118
9.090     107
37.880     97
28.030     80
7.575      75
38.635     74
6.060      73
39.395     67
6.820      63
8.335      63
18.940     45
40.150     45
40.910     39
5.305      25
42.425     24
41.665     23
3.790      16
4.545      11
3.030       7
43.940      7
2.275       7
43.180      7
44.695      3
0.760       2
1.515       1
```

```
45.455      1
Name: count, dtype: int64
```

In [4]:
```
df['humidity'].value_counts()
```

Out[4]:
```
humidity
88    368
94    324
83    316
87    289
70    259
     ...
8       1
10      1
97      1
96      1
91      1
Name: count, Length: 89, dtype: int64
```

In [5]:
```
df['windspeed'].value_counts()
```

Out[5]:
```
windspeed
0.0000     1313
8.9981     1120
11.0014    1057
12.9980    1042
7.0015     1034
15.0013     961
6.0032      872
16.9979     824
19.0012     676
19.9995     492
22.0028     372
23.9994     274
26.0027     235
27.9993     187
30.0026     111
31.0009      89
32.9975      80
35.0008      58
39.0007      27
36.9974      22
43.0006      12
40.9973      11
43.9989       8
46.0022       3
56.9969       2
47.9988       2
51.9987       1
50.0021       1
Name: count, dtype: int64
```

In [6]:
```
df['casual'].value_counts()
```

Out[6]:
```
casual
0      986
1      667
2      487
3      438
4      354
      ...
332      1
361      1
356      1
331      1
304      1
Name: count, Length: 309, dtype: int64
```

In [7]:
```
df['registered'].value_counts()
```

Out[7]:
```
registered
3      195
4      190
5      177
6      155
2      150
      ...
570      1
422      1
678      1
565      1
636      1
Name: count, Length: 731, dtype: int64
```

In [8]:
```
df['count'].value_counts()
```

Out[8]:
```
count
5      169
4      149
3      144
6      135
2      132
      ...
801      1
629      1
825      1
589      1
636      1
Name: count, Length: 822, dtype: int64
```

In [6]:
```
arr = df.copy() # making a copy of data
```

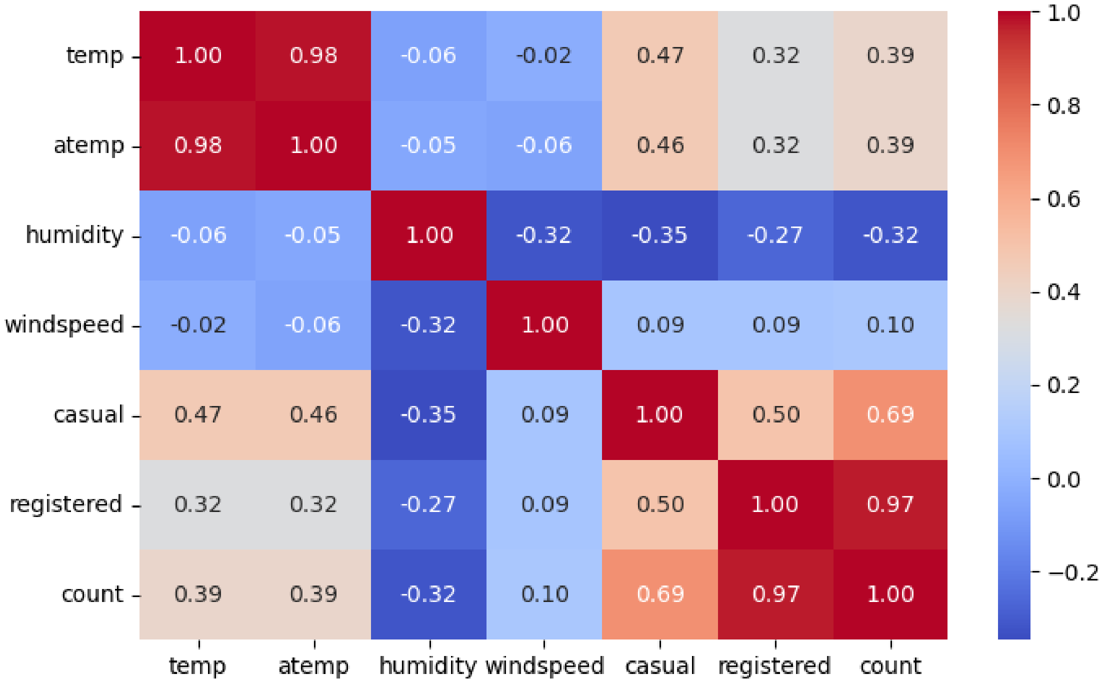
In [7]:
```
arr.head()
```

Out[7]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regis
0	1	0	0	1	9.84	14.395	81	0.0	3	
1	1	0	0	1	9.02	13.635	80	0.0	8	
2	1	0	0	1	9.02	13.635	80	0.0	5	
3	1	0	0	1	9.84	14.395	75	0.0	3	
4	1	0	0	1	9.84	14.395	75	0.0	0	

In [8]:
```
arr.drop(columns = ['Date','season','holiday','weather','workingday'], i
arr[['temp', 'atemp', 'windspeed']] = arr[['temp', 'atemp', 'windspeed']
```

In [10]:
```
# Heatmap for co relation
correlation_matrix = arr.corr()

plt.figure(figsize=(8,5))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.show()
```



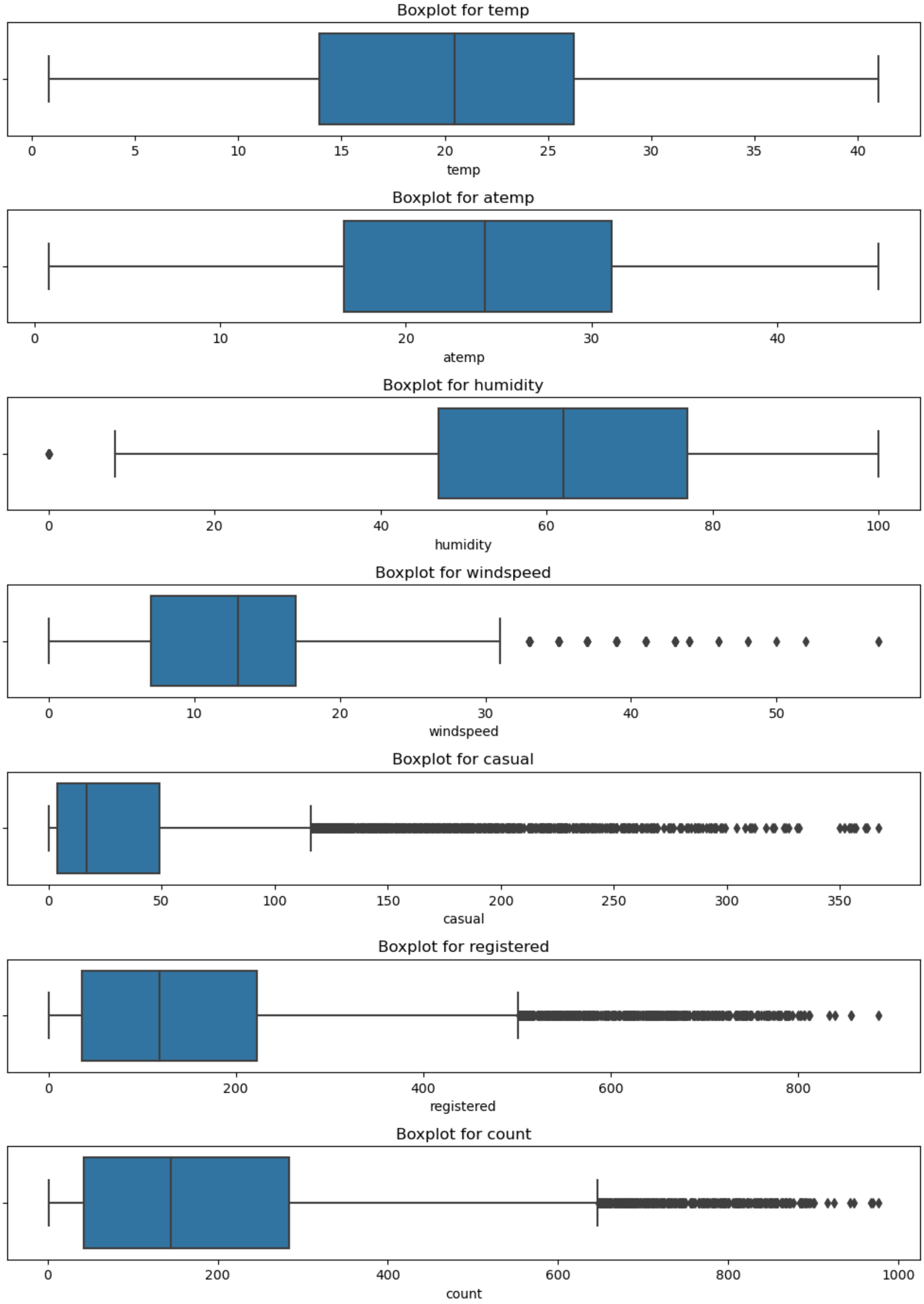
Noteworthy points

1. Temperature ('temp') and apparent temperature ('atemp') show an extremely positive correlation, as expected. Humidity and weather also exhibit a positive relation.
2. Casual, registered, and overall user count have a positive relation with temperature ('temp') and apparent temperature ('atemp'), suggesting an influence of temperature on bike usage.
3. Casual, registered, and overall user count have an extremely positive relationship with each other, indicating a strong correlation among these user-related metrics.
4. Windspeed demonstrates a positive correlation with casual, registered, and overall user count, suggesting potential impacts on bike usage.
5. Humidity displays a negative correlation with every other column, indicating lower bike usage during higher humidity levels.

In [11]:
```
df['humidity'].value_counts()
```

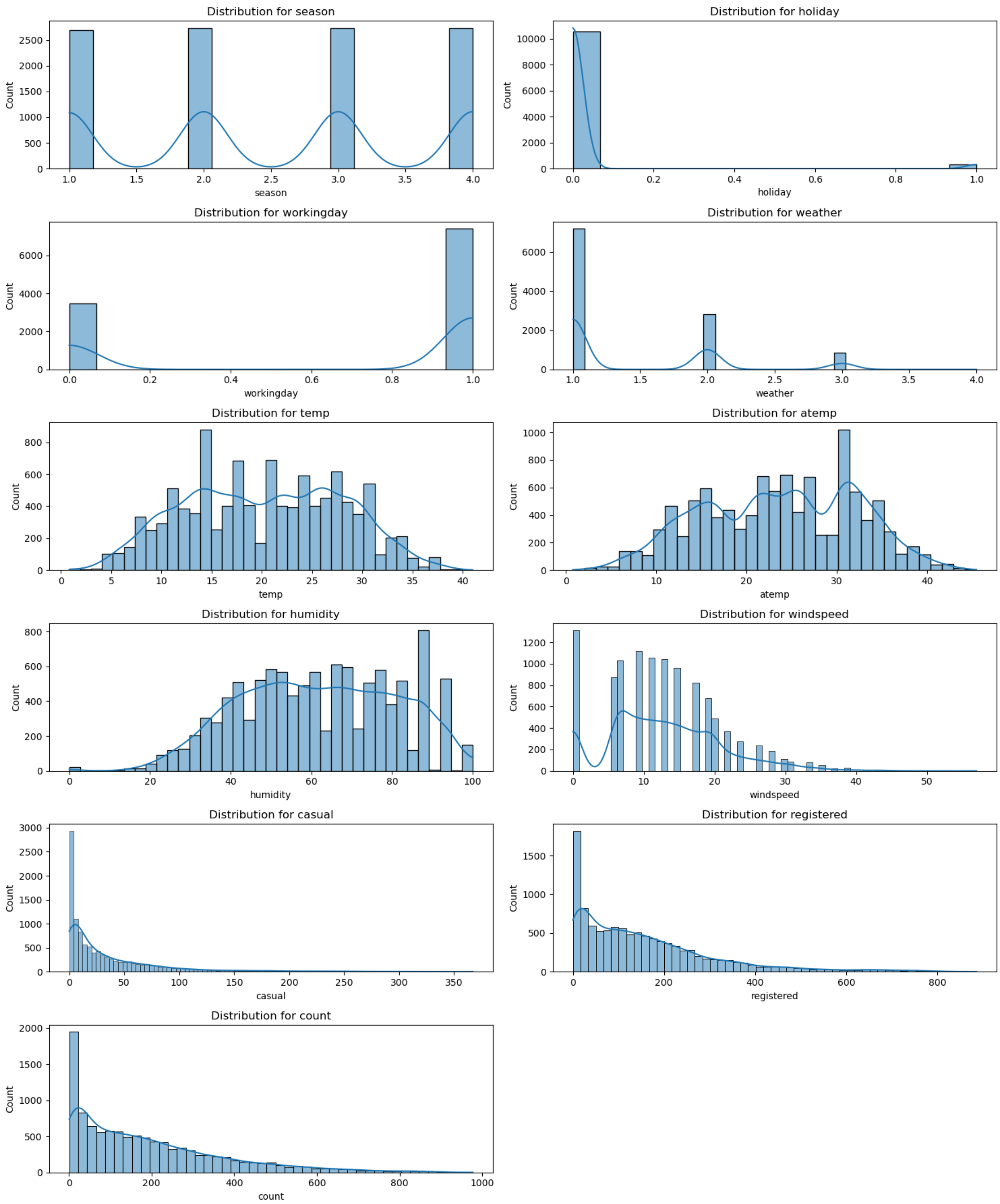
Out[11]:
```
humidity
88    368
94    324
83    316
87    289
70    259
     ...
8       1
10      1
97      1
96      1
91      1
Name: count, Length: 89, dtype: int64
```

In [11]:
```
# Boxplot for continous columns

columns = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'register

# Create subplots
fig, axes = plt.subplots(nrows=len(columns), figsize=(10, 2 * len(column

# Plot boxplots for each numeric column
for i, column in enumerate(columns):
    sns.boxplot(x=df[column], ax=axes[i])
    axes[i].set_title(f'Boxplot for {column}')

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



Outliers are evident in the dataset, with humidity showing one outlier, windspeed exhibiting 12 outliers, and a notable presence of outliers in the casual, count, and registered variables.

In [17]:
```
# Plot distribution for each column

columns = ['season', 'holiday', 'workingday', 'weather', 'temp', 'atemp


num_rows = len(columns) // 2 + len(columns) % 2
fig, axes = plt.subplots(num_rows, 2, figsize=(15, 3 * num_rows))
axes = axes.flatten()


for i, column in enumerate(columns):
    sns.histplot(df[column], kde=True, ax=axes[i])
    axes[i].set_title(f'Distribution for {column}')

for j in range(len(columns), len(axes)):
    fig.delaxes(axes[j])

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

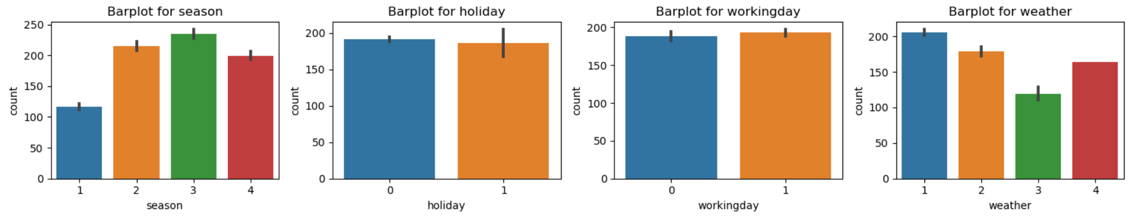


Insight:

The distributions depicted in the above plot indicate right-skewness for the 'atemp', 'casual', 'registered', and 'count' columns. None of the columns exhibit a normal distribution.

Univariate Analysis

In [46]:
```
# Bar plots for each categorical column

columns = ['season', 'holiday', 'workingday', 'weather']

fig, axes = plt.subplots(1, len(columns), figsize=(15, 3))

for i, num in enumerate(columns):
    sns.barplot(x=num,y = 'count', data=df, ax=axes[i])
    axes[i].set_title(f'Barplot for {num}')

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



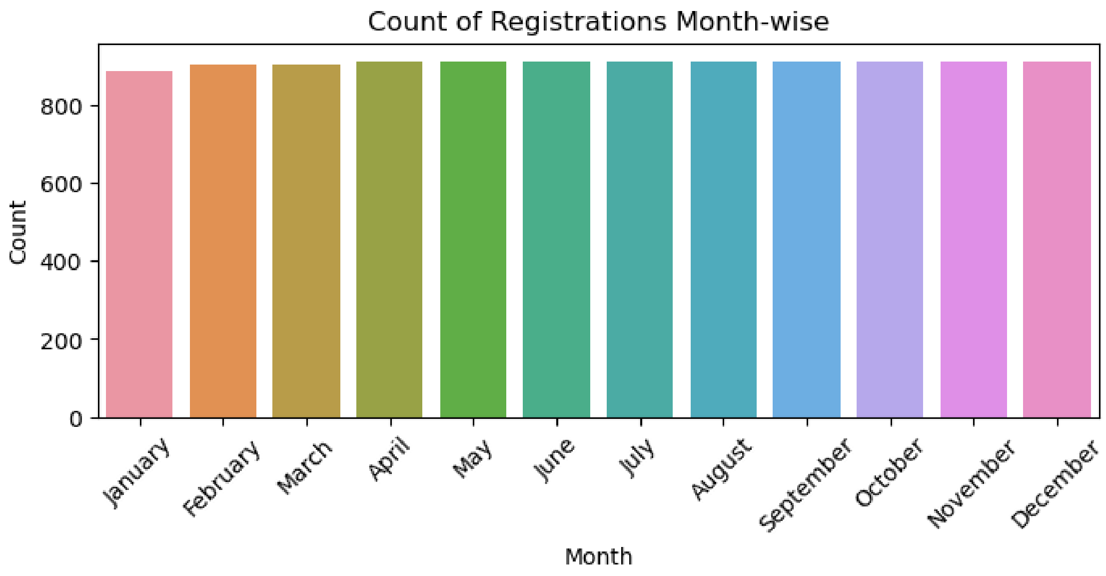
Insight:

1.Seasons 3, 2, and 4 exhibit nearly equal traffic distribution with slight variations, but Season 1 experiences a noticeable dip.

2&3.The usage of bikes is comparable between working days and weekends, indicating a similar level of user engagement on both types of days.

4.Seasons 1, 2, and 3 show minor differences in distribution, but Season 3 is adversely affected, possibly due to the occurrence of storms or other impactful events during this season.

In [44]:
```
# Barplot for Count of Registrations Month-wise

df['Date'] = pd.to_datetime(df['Date'])  # Converting 'Date' to datetime
df['Month'] = df['Date'].dt.month_name()

# Group by Month
filter = df.groupby('Month')['count'].size().reset_index(name='count_per


plt.figure(figsize=(8, 3))
sns.barplot(x='Month', y='count_per_month', data=filter, order=['January
plt.xticks(rotation=45)

plt.title('Count of Registrations Month-wise')
plt.xlabel('Month')
plt.ylabel('Count')
plt.show()
```

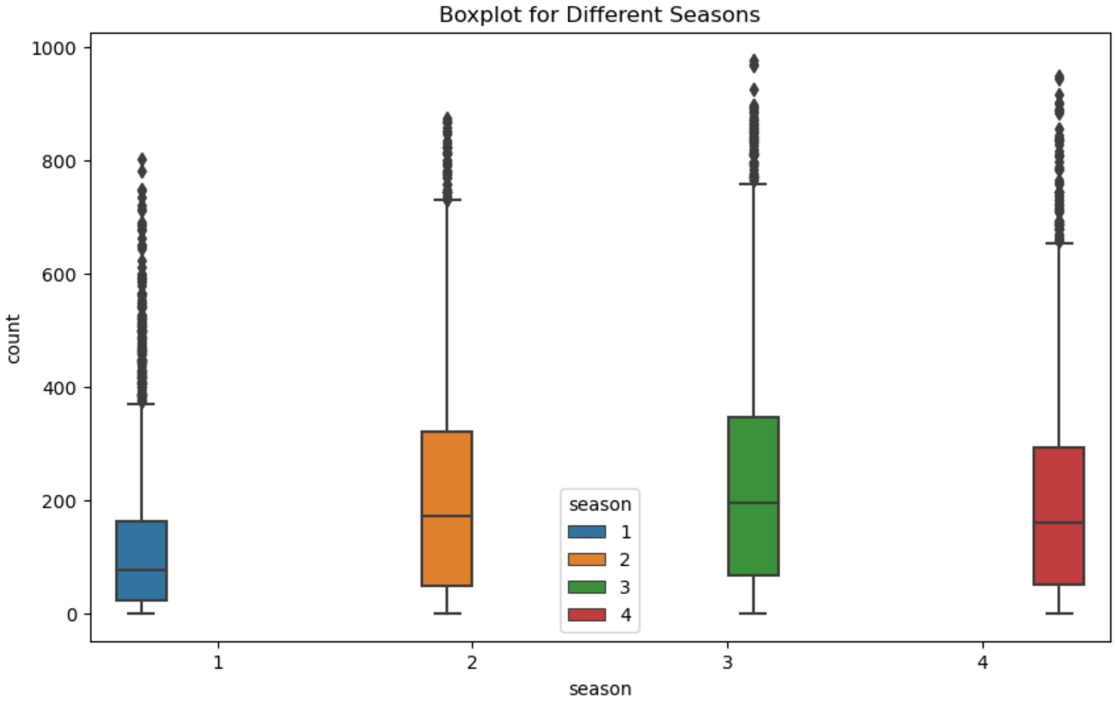


Insight:

There is similar distribution of bikes rented among all the months.

Bivariate Analysis

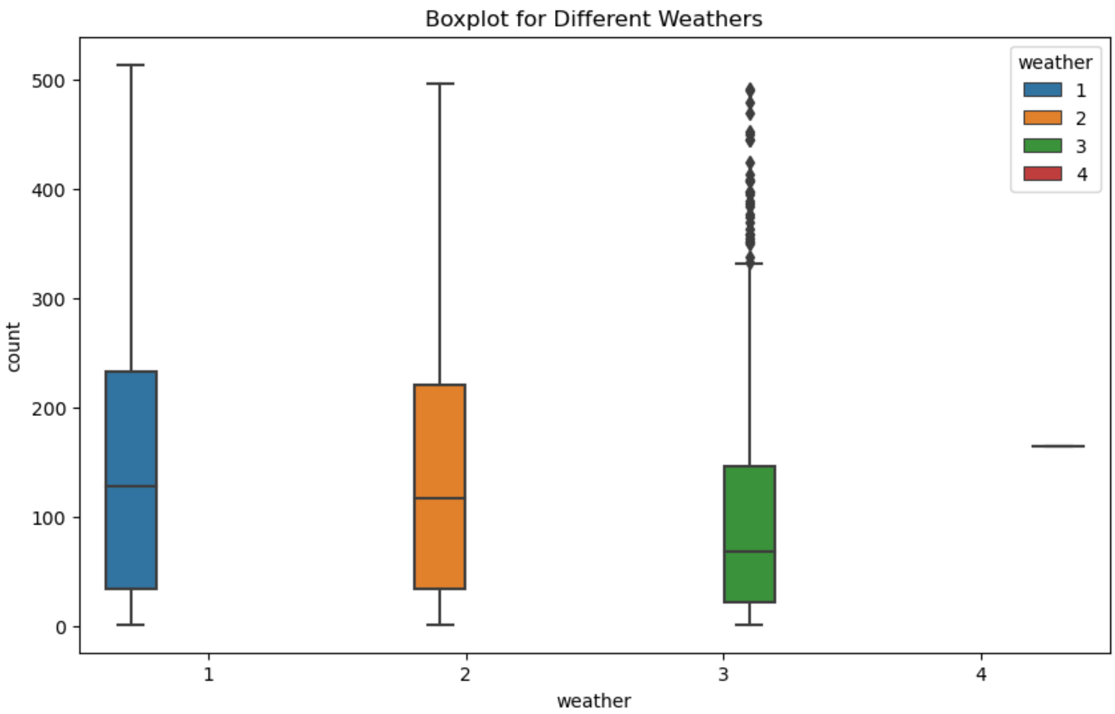
In [113]:
```
# Boxplot for Different Seasons

plt.figure(figsize = (10,6))

sns.boxplot(x = 'season', y = 'count', data = df, hue = 'season')

plt.title('Boxplot for Different Seasons')
plt.show()
```



Insight:

Bike rentals exhibit higher demand during the summer and fall compared to the winter and spring seasons.

In [100]:
```
# Boxplot for Different Weathers

plt.figure(figsize = (10,6))

sns.boxplot(x = 'weather', y = 'count', data = df, hue = 'weather')

plt.title('Boxplot for Different Weathers')
plt.show()
```



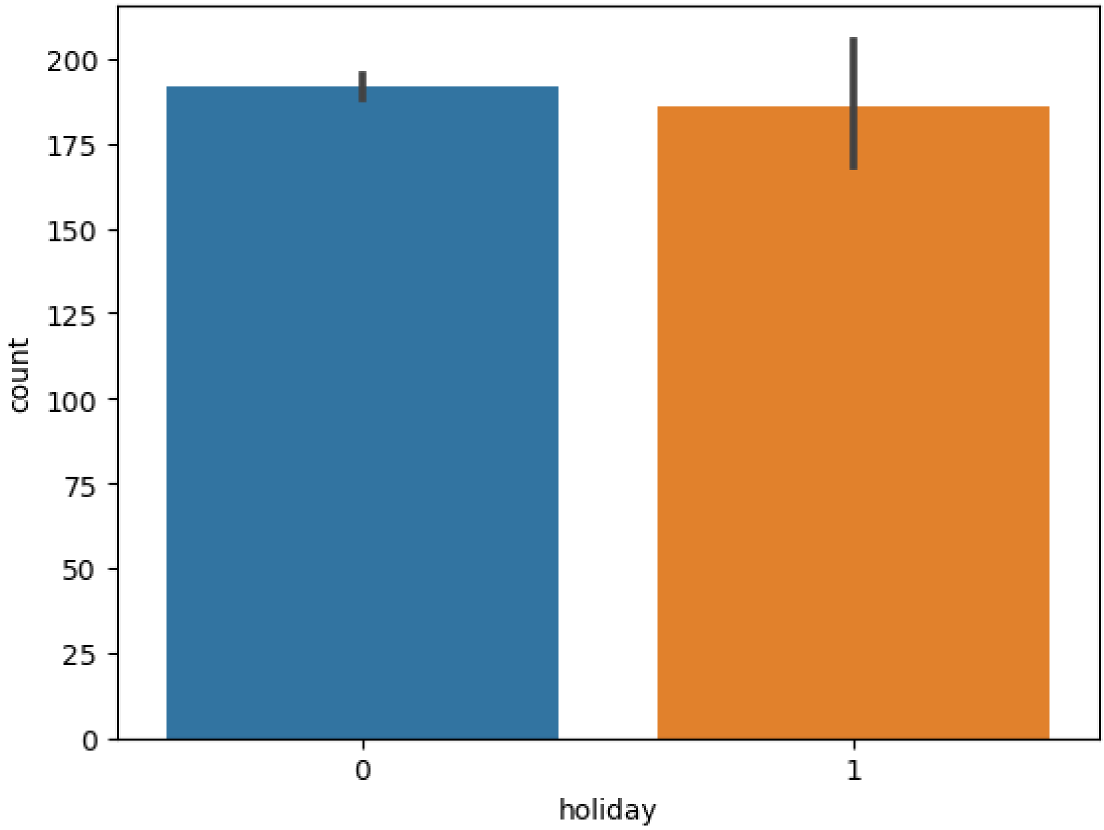
Insight:

1: The preference for bike rentals is evident in favorable weather conditions such as Clear, Few clouds, partly cloudy, Mist + Cloudy, Mist + Broken clouds, and Mist + Few clouds. The data reflects the region's inclination towards these weather types for biking activities.

2: Weather conditions categorized as 3 (Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds) result in decreased bike rentals, suggesting that these weather patterns are less conducive to biking. Additionally, weather conditions classified as 4 (Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog) does not contribute in bike usage, indicating unfavorable conditions for biking activities.

In [17]:
```
sns.barplot(x = 'holiday', y = 'count', data = df)
```

Out[17]: <Axes: xlabel='holiday', ylabel='count'>



Using IQR to handle outliers

In [41]:
```
# Using IQR method to deal with outliers

columns = ['windspeed', 'casual', 'registered', 'count']

for col in columns:
    Q1 = np.percentile(df[col], 25)
    Q3 = np.percentile(df[col], 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - IQR * 1.5
    upper_bound = Q3 + IQR * 1.5
    outliers = (df[col] < lower_bound) | (df[col] > upper_bound)
    df = df[~outliers]
```

In [43]:
```
df.shape
```

Out[43]: (9381, 12)

Insight:

A total of over 1500 rows were removed from the dataset due to their outlier status. This decision was made to mitigate potential impacts on our statistical analysis.

Is there effect of working day on cycles being rented?

Null Hypothesis(H0): There is no effect of working day on the number of electric cycles being rented.

Alternate Hypothesis(H1): There is a significant effect of working day on the number of electric cycles being rented.

Significance level = 0.05

Performing 2 sample independent T-test to prove this.

In [18]:
```
working_day = df[df['workingday'] == 1]['count'].values
non_working_day = df[df['workingday'] == 0]['count'].values
```

In [21]:
```
# 2 Sample Independent T-test

t_stat, p_value = stats.ttest_ind(working_day, non_working_day,alternat:

print(f'T-statistic: {t_stat}')
print(f'P-value: {p_value}')

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant effect of
else:
    print("Fail to reject the null hypothesis: There is no significant e
```

```
T-statistic: 1.2096277376026694
P-value: 0.11322402113180674
Fail to reject the null hypothesis: There is no significant effect of
working day on the number of electric cycles rented.
```

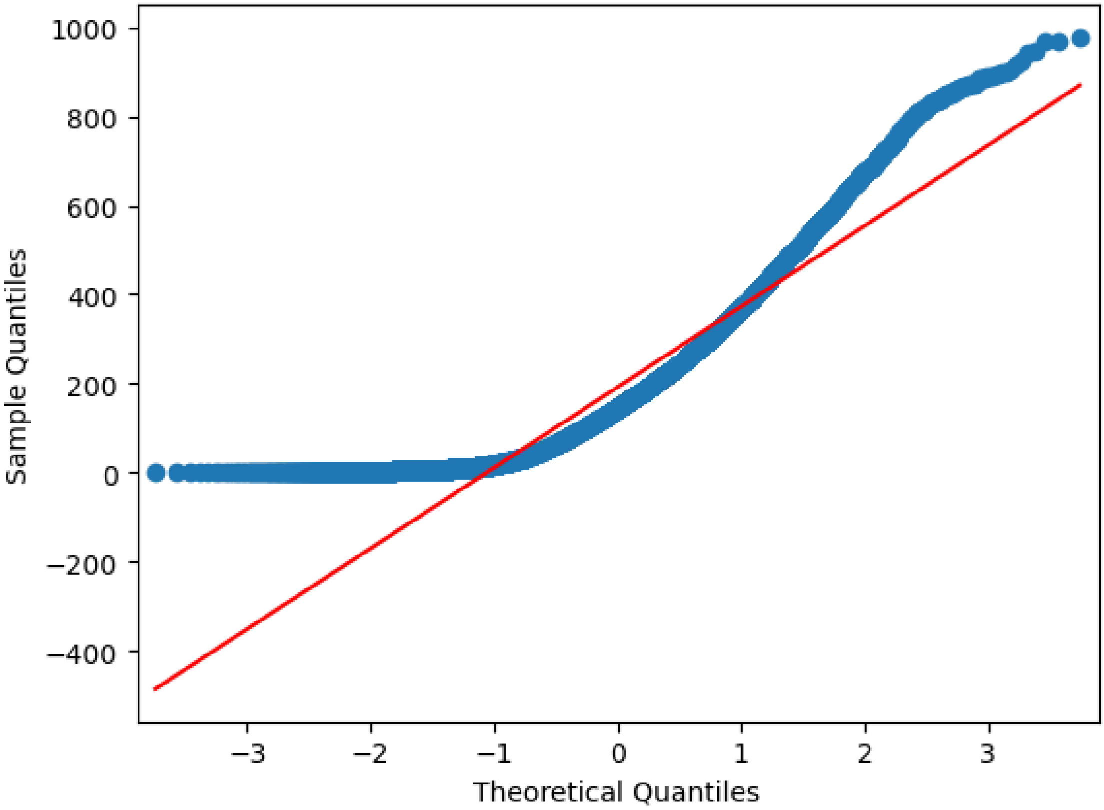
Insight:

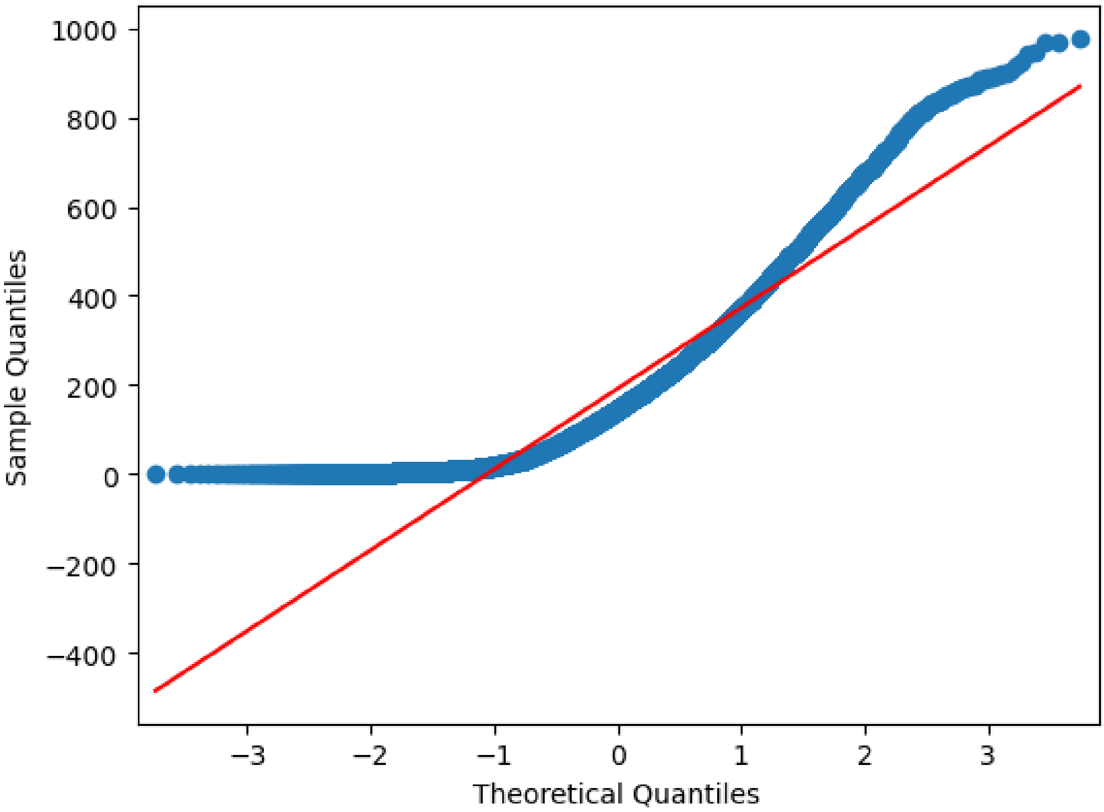
Our test proves that,There is no significant effect of working day on the number of electric cycles rented.

Are no. of cycles rented similar or different in different seasons?

In [25]:
```
qqplot(df["count"], line = "s")
```

Out[25]:





Shapiro Wilk Test

Null Hypothesis(H0): Data is gaussian

Alternate Hypothesis(H1): Data is not gaussian

In [52]:
```
count = df['count']

count_subset  = count.sample(4000)

t_stat, p_value = shapiro(count_subset)

print(f'T-statistic: {t_stat}')
print(f'P-value: {p_value}')

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: Data is not gaussian.")
else:
    print("Fail to reject the null hypothesis: Data is gaussian")
```

```
T-statistic: 0.9114214777946472
P-value: 2.2420775429197073e-43
Reject the null hypothesis: Data is not gaussian.
```

LEVENE TEST

Helps us determine whether the variances within the groups are roughly the same or not.

Null Hypothesis(H0): Variances are equal

Alternate Hypothesis(Ha): Variances are not equal

In [55]:
```
season_1 = df[df['season']==1]['count']
season_2 = df[df['season']==2]['count']
season_3 = df[df['season']==3]['count']
season_4 = df[df['season']==4]['count']
```

In [61]:
```
levene_stat, p_value = levene(season_1, season_2, season_3, season_4)

print(f'Levene-value: {levene_stat}')
print(f'P-value: {p_value}')

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: Variances are not Equal.")
else:
    print("Fail to reject the null hypothesis: Variances are Equal")
```

```
Levene-value: 138.09834574958816
P-value: 1.479015830333895e-87
Reject the null hypothesis: Variances are not Equal.
```

Since we can see that data does not follow assumptions of One Way ANOVA, we will need to perform Kruskal-Wallis test in order to make conclusions.

Null Hypothesis (H0): No. of cycles rented across different seasons are similar.

Alternative Hypothesis (H1): At least in one of the seasons No. of cycles rented are different.

Significance level = 0.05

In [64]:
```
t_stat, p_value = kruskal(season_1, season_2, season_3, season_4)

print(f'T-statistic: {t_stat}')
print(f'P-value: {p_value}')

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: At least in one of the seasons No
else:
    print("Fail to reject the null hypothesis: No. of cycles rented acro
```

```
T-statistic: 414.73793455525293
P-value: 1.4212554003308481e-89
Reject the null hypothesis: At least in one of the seasons No. of cycl
es rented are different.
```

Insight:

Our test proves that, in atleast one of the seasons No. of cycles rented are different.

Does the No. of cycles rented similar or different in different weather?

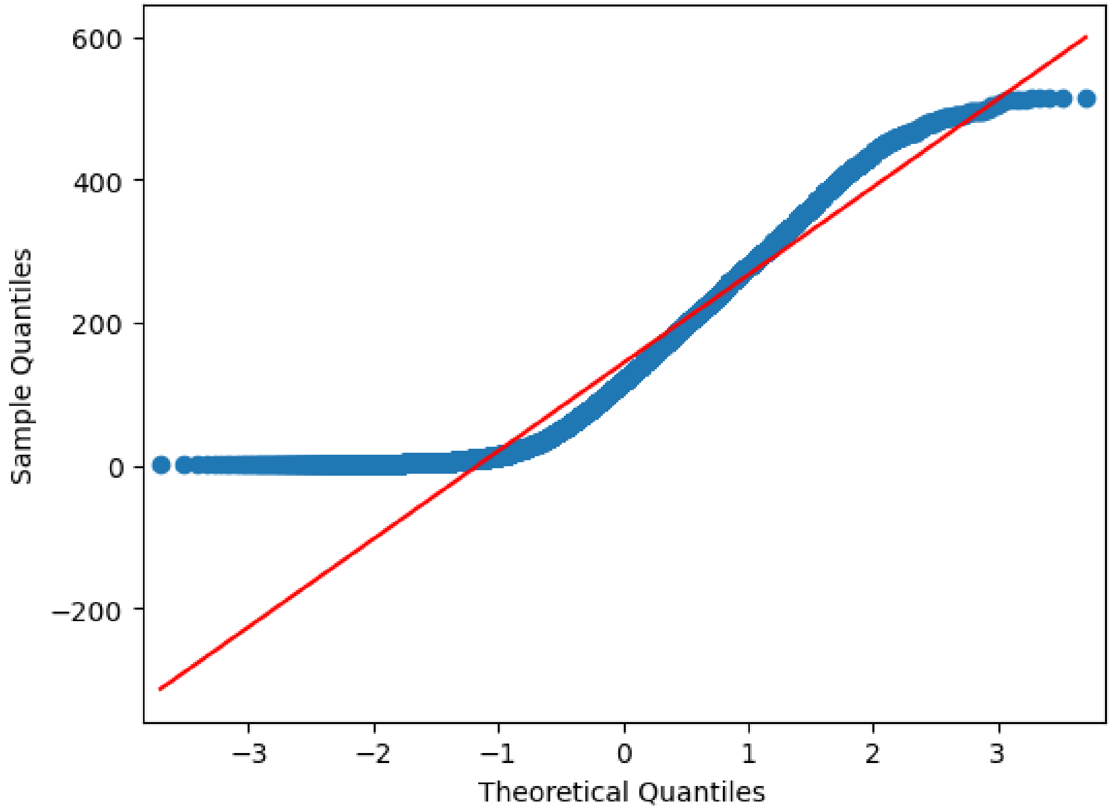
Null Hypothesis(H0): There is no difference in no. of cycles being rented for different weather conditions.

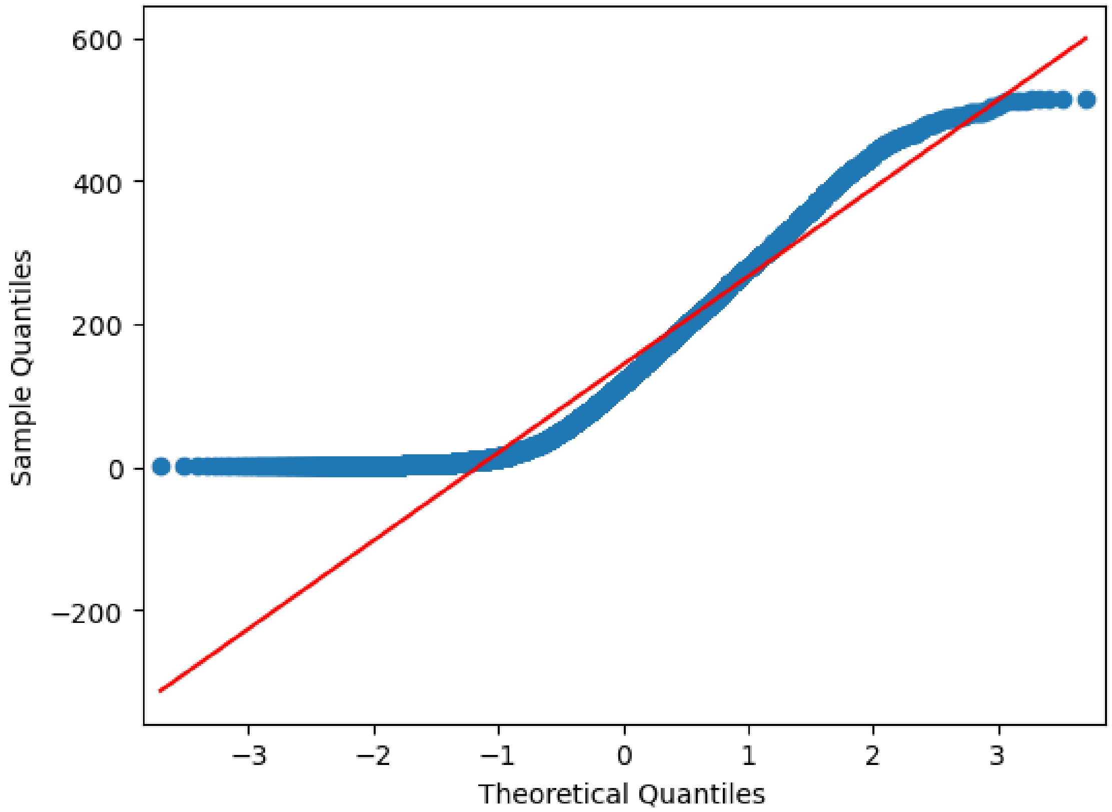
Alternate Hypothesis(H1): There is significant difference in no. of cycles being rented for different weather conditions.

significance level = 0.05

In [69]:
```
qqplot(df["count"], line = "s")
```

Out[69]:





Based on the Q-Q plot above, it is evident that the distribution of the 'count' column deviates from normality. As a preliminary step, we will assess the assumptions required for an ANOVA test to determine its feasibility.

Shapiro Wilk Test

Null Hypothesis(H0): Data is gaussian

Alternate Hypothesis(H1): Data is not gaussian

In [71]:
```
count = df['count']

count_subset  = count.sample(4000)

t_stat, p_value = shapiro(count_subset)

print(f'T-statistic: {t_stat}')
print(f'P-value: {p_value}')

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: Data is not gaussian.")
else:
    print("Fail to reject the null hypothesis: Data is gaussian")
```

```
T-statistic: 0.9127504229545593
P-value: 3.853570776893247e-43
Reject the null hypothesis: Data is not gaussian.
```

LEVENE TEST

Helps us determine whether the variances within the groups are roughly the same or not.

Null Hypothesis(H0): Variances are equal

Alternate Hypothesis(Ha): Variances are not equal

In [5]:
```
weather_1 = df[df['weather']==1]['count']
weather_2 = df[df['weather']==2]['count']
weather_3 = df[df['weather']==3]['count']
weather_4 = df[df['weather']==4]['count']
```

In [6]:
```
# Levene's Test

levene_stat, p_value = levene(weather_1, weather_2, weather_3, weather_4

print(f'Levene-value: {levene_stat}')
print(f'P-value: {p_value}')

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: Variances are not Equal.")
else:
    print("Fail to reject the null hypothesis: Variances are Equal")
```

```
Levene-value: 54.85106195954556
P-value: 3.504937946833238e-35
Reject the null hypothesis: Variances are not Equal.
```

Since we can see that data does not follow assumptions of One Way ANOVA, we will need to perform Kruskal-Wallis test in order to make conclusions.

In [ ]:
```
Null Hypothesis (H0): No. of cycles rented across different weathers are
Alternative Hypothesis (H1): At least in one of the weathers No. of cycl
Significance level = 0.05
```

In [7]:
```
# Kruksal Wallis Test

t_stat, p_value = kruskal(weather_1, weather_2, weather_3, weather_4)

print(f'T-statistic: {t_stat}')
print(f'P-value: {p_value}')

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: At least in one of the weathers N
else:
    print("Fail to reject the null hypothesis: No. of cycles rented acro
```

```
T-statistic: 205.00216514479087
P-value: 3.501611300708679e-44
Reject the null hypothesis: At least in one of the weathers No. of cyc
les rented are different.
```

Insight

Our test proves that, in atleast one of the weathers No. of cycles rented are different.

Are Weather conditions significantly different during different Seasons?

In [79]:
```
Weather = df['weather']
Season = df['season']
```

In [81]:
```
observed = pd.crosstab(Weather,Season)
```

Performing Chi-Square test

Null Hypothesis(H0): Weather conditions are independent of Seasons.

Alternate Hypothesis(H1): Weather conditions are dependent of Seasons.

Significance level : 0.05

In [85]:
```
# Chi-Square test

chi2_stat, p_value, dof, expected = chi2_contingency(observed)

print(f"Chi-square Statistic: {chi2_stat}")
print(f"P-value: {p_value}")
print(f"Degrees of Freedom: {dof}")
print("Expected Frequencies:")
print(expected)


alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: Weather conditions are dependent
else:
    print("Fail to reject the null hypothesis: Weather conditions are Ir
```

```
Chi-square Statistic: 49.11735823421667
P-value: 1.5778137220034608e-07
Degrees of Freedom: 9
Expected Frequencies:
[[1.59350560e+03 1.45210457e+03 1.43919124e+03 1.57219859e+03]
 [6.66394201e+02 6.07261166e+02 6.01860889e+02 6.57483744e+02]
 [2.07837118e+02 1.89394521e+02 1.87710265e+02 2.05058096e+02]
 [2.63084959e-01 2.39739900e-01 2.37607931e-01 2.59567210e-01]]
Reject the null hypothesis: Weather conditions are dependent on Season
s .
```

Insight:

Our test proves that, Weather conditions are dependent on Seasons.

Recommendations based on Insights

1.Implement a dynamic bike distribution strategy that aligns with weather conditions. Increase bike availability during favorable seasons (summer and fall) when demand is high, and consider reducing fleet size in extreme weather conditions to optimize costs.

2.Prioritize bike maintenance efforts during the summer and fall seasons, ensuring that electric cycles are in top condition to meet the heightened demand. This can enhance user experience and contribute to positive word-of-mouth.

3.Optimize resource allocation by strategically managing the number of bikes in locations. During rush seasons, ensure ample availability, while considering cost-effective measures, such as reducing bike availability during less-demanding periods like winter and fall.

Recommendations

1.Launch targeted marketing campaigns and promotions during peak demand seasons (summer and fall) to engage users and encourage increased bike usage. This can include special offers, rewards, or events to attract more riders.

2.Consider implementing dynamic pricing models that respond to weather conditions. For example, offer discounts during extreme weather conditions to encourage bike usage and adjust prices during peak demand to reflect increased operational costs.

3.Educate users about how weather conditions can affect bike availability and usage. Provide tips or notifications on the app to inform users about the impact of weather on their riding experience, managing expectations during extreme conditions.

In [ ]:

```
```