In [1]:
```
#Importing the Python libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Hypothesis Testing
from scipy.stats import f, f_oneway, shapiro, ttest_ind, levene, kruskal,chi2_co
```

In [2]:
```
df=pd.read_csv("yulu_dataset.csv")
```

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

Out[3]:

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

In [4]:
```
df.shape
```

Out[4]:
```
(10886, 12)
```

In [5]:
```
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 [6]:
```
df.dtypes
```

Out[6]:
```
datetime       object
season          int64
holiday         int64
workingday      int64
weather         int64
temp          float64
atemp         float64
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
dtype: object
```

Detect Null values

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

Check for Duplicates

In [8]:
```
df.duplicated().sum()
```

Out[8]:
```
0
```

Statistical summary

In [9]:
```
# Categorical variable
df.describe(include='object')
```

Out[9]:

	datetime
count	10886
unique	10886
top	2011-01-01 00:00:00
freq	1

In [10]:
```
#Numerical variable
df.describe()
```

Out[10]:

	season	holiday	workingday	weather	temp	ater
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.0000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.6550
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

Data Wrangling

In [11]:
```
df["datetime"] = pd.to_datetime(df['datetime'])
df = df.astype({'season': object, 'weather': object, 'holiday': object, 'working
df.info()
```

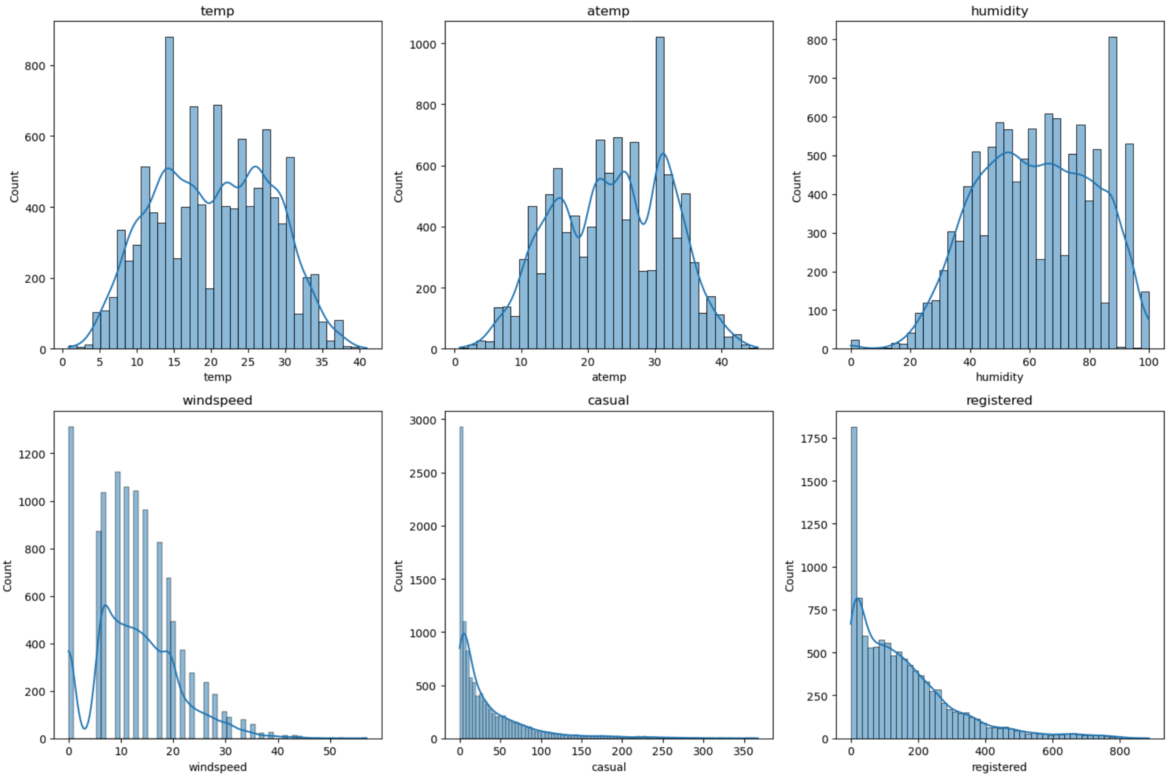
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  object        
 2   holiday     10886 non-null  object        
 3   workingday  10886 non-null  object        
 4   weather     10886 non-null  object        
 5   temp        10886 non-null  float64       
 6   atemp       10886 non-null  float64       
 7   humidity    10886 non-null  int64         
 8   windspeed   10886 non-null  float64       
 9   casual      10886 non-null  int64         
 10  registered  10886 non-null  int64         
 11  count       10886 non-null  int64         
dtypes: datetime64[ns](1), float64(3), int64(4), object(4)
memory usage: 1020.7+ KB
```

In [14]:
```
df['season'] = df['season'].map({
    1: 'Spring', 2: 'Summer', 3: 'Fall', 4: 'Winter'
})
```

Distribution of Numerical & Categorical variables

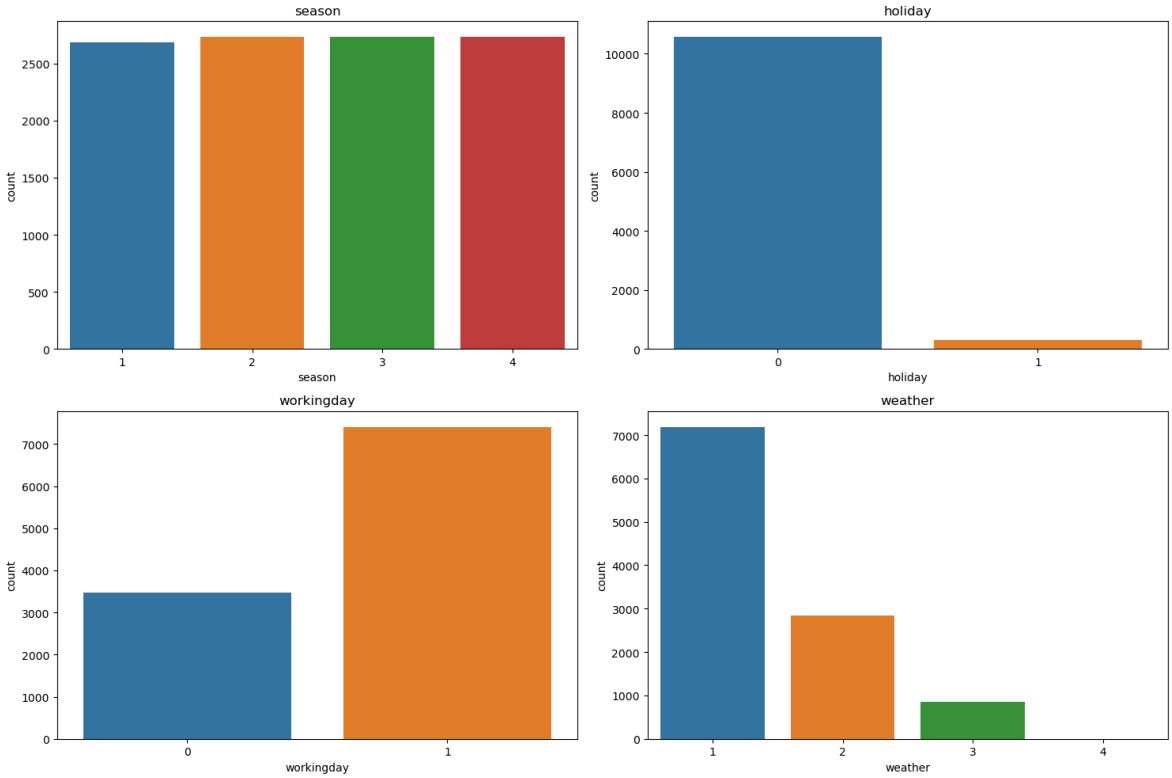
In [12]:
```
numerical = [ 'temp','atemp', 'humidity', 'windspeed', 'casual', 'registered']
# Create subplots
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
# Flatten the axes so that we can iterate over them easily
axes = axes.flatten()
# Iterate over numerical variables and plot histplots
for i, var in enumerate(numerical):
    sns.histplot(x=df[var], ax=axes[i],kde=True)
    axes[i].set_title(var)
# Adjust layout
plt.tight_layout()
plt.show()
```

```
C:\Users\Pawan Kumar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future versi
on. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Pawan Kumar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future versi
on. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Pawan Kumar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future versi
on. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Pawan Kumar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future versi
on. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Pawan Kumar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future versi
on. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Pawan Kumar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future versi
on. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



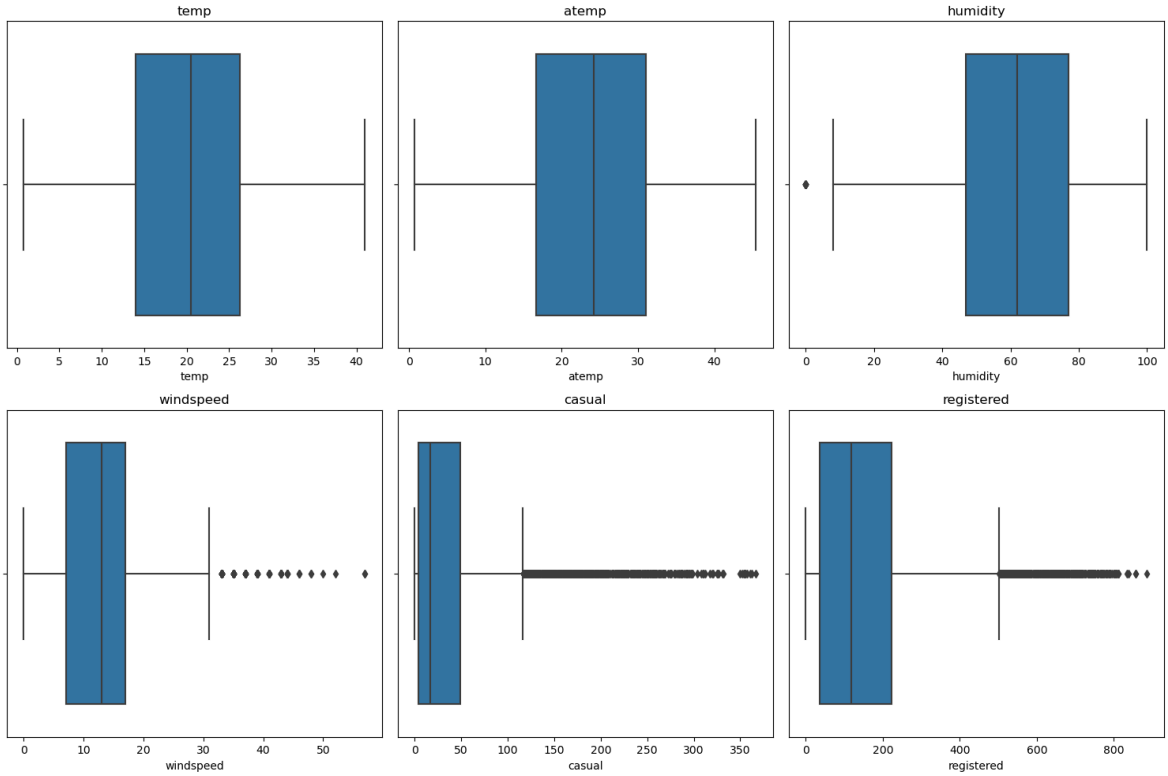
In [13]:
```python
categorical = [ 'season','holiday', 'workingday', 'weather']
# Create subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))
# Flatten the axes so that we can iterate over them easily
axes = axes.flatten()
# Iterate over categorical variables and plot countplots
for i, var in enumerate(categorical):
    sns.countplot(x=df[var], ax=axes[i])
    axes[i].set_title(var)
# Adjust layout
```

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



Detect and Handle Outliers

In [15]:
```
numerical = [ 'temp','atemp', 'humidity', 'windspeed', 'casual', 'registered']
# Create subplots
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
# Flatten the axes so that we can iterate over them easily
axes = axes.flatten()
# Iterate over numerical variables and plot boxplots
for i, var in enumerate(numerical):
    sns.boxplot(x=df[var], ax=axes[i])
    axes[i].set_title(var)
# Adjust layout
plt.tight_layout()
plt.show()
```



In [16]:
```
# Calculate 5th and 95th percentiles for each column
percentile_5 = df[numerical].quantile(0.05)
percentile_95 = df[numerical].quantile(0.95)
# Clip the data for each column between the 5th and 95th percentiles
df[numerical] = df[numerical].clip(percentile_5, percentile_95, axis=1)
# Output the clipped data
df[numerical]
```

Out[16]:

	temp	atemp	humidity	windspeed	casual	registered
0	9.84	14.395	81	0.0000	3	13
1	9.02	13.635	80	0.0000	8	32
2	9.02	13.635	80	0.0000	5	27
3	9.84	14.395	75	0.0000	3	10
4	9.84	14.395	75	0.0000	0	4
...	...	...	...	...	...	...
10881	15.58	19.695	50	26.0027	7	329
10882	14.76	17.425	57	15.0013	10	231
10883	13.94	15.910	61	15.0013	4	164
10884	13.94	17.425	61	6.0032	12	117
10885	13.12	16.665	66	8.9981	4	84

10886 rows × 6 columns

In [17]:
```
q1 = df['count'].quantile(0.25)
q3 = df['count'].quantile(0.75)
```

```
iqr = q3-q1
iqr
```

Out[17]: 242.0

In [18]:
```
df = df[(df['count']>(q1-1.5*iqr)) & (df['count'] < (q3 + 1.5 *iqr))]
df
```

Out[18]:

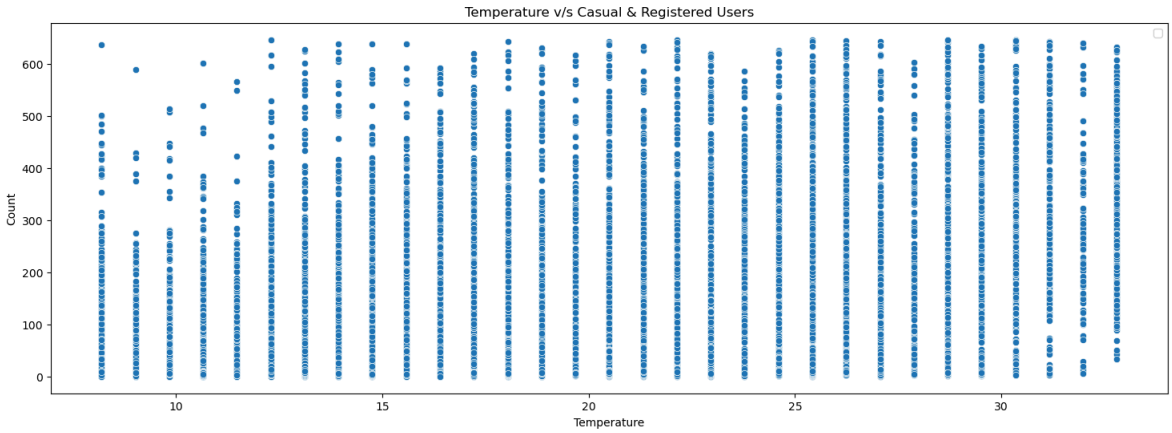
	datetime	season	holiday	workingday	weather	temp	atemp	humidity	winc
0	2011-01-01 00:00:00	Spring	0	0	1	9.84	14.395	81	
1	2011-01-01 01:00:00	Spring	0	0	1	9.02	13.635	80	
2	2011-01-01 02:00:00	Spring	0	0	1	9.02	13.635	80	
3	2011-01-01 03:00:00	Spring	0	0	1	9.84	14.395	75	
4	2011-01-01 04:00:00	Spring	0	0	1	9.84	14.395	75	
...	...	...	...	...	...	...	...	...	
10881	2012-12-19 19:00:00	Winter	0	1	1	15.58	19.695	50	2
10882	2012-12-19 20:00:00	Winter	0	1	1	14.76	17.425	57	1
10883	2012-12-19 21:00:00	Winter	0	1	1	13.94	15.910	61	1
10884	2012-12-19 22:00:00	Winter	0	1	1	13.94	17.425	61	
10885	2012-12-19 23:00:00	Winter	0	1	1	13.12	16.665	66	

10583 rows × 12 columns

Relationship between the Dependent and Independent Variables

In [19]:
```
# Plot visualizing difference between count of casual user and registered users
plt.figure(figsize=(18, 6))
sns.scatterplot(x ="temp",y = "count",data = df)
plt.xlabel('Temperature')
plt.ylabel('Count')
plt.title('Temperature v/s Casual & Registered Users')
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
```



No. of bike rides on Weekdays and Weekends

In [20]:
```
holiday_count=df[df['holiday'] == 1]['count']
non_holiday_count=df[df['holiday'] == 0]['count']
# Perform 2-sample t-test
t_stat, p_value = ttest_ind(holiday_count, non_holiday_count)
t_stat, p_value
```

Out[20]: (0.8002380245944458, 0.4235908688796304)

In [21]:
```
if p_value<0.05:
    print("Reject Ho")
    print("The mean number of electric cycles rented is different on holidays co
else:
    print("Fail to reject Ho")
    print("The mean number of electric cycles rented is the same on holidays and
```

```
Fail to reject Ho
The mean number of electric cycles rented is the same on holidays and non-holiday
s
```

Number of Bicycles Rented in Different Weather Conditions

In [22]:
```
# Performing Anova test for effect of Weather
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 [32]:
```
f_stat, p_value = f_oneway(weather_1, weather_2, weather_3,weather_4)

print("f_stat: ", f_stat)
print("p-value: ", p_value)

#Significance level is 5%
alpha = 0.05
if p_value < alpha:
    print("Reject Ho: The mean number of electric cycles rented differs across w
else:
    print("Failed to reject Ho: The mean number of electric cycles rented is the
```

```
f_stat:  64.38048872136727
p-value:  3.029209202309234e-41
Reject Ho: The mean number of electric cycles rented differs across weather condi
tions
```

In [41]:
```
#Normality test
sstat,pvalue=shapiro(df["count"].sample(4999))
print(pvalue)
if pvalue<0.05:
  print("Normally distributed")
else:
  print("Not normally distributed")
```

```
0.0
Normally distributed
```

In [44]:
```
from scipy.stats import kstest
kstat,pvalue=kstest(weather_1,weather_2,weather_3,weather_4)
print(pvalue)
if pvalue<0.05:
  print("Normally distributed")
else:
  print("Not normally distributed")
```

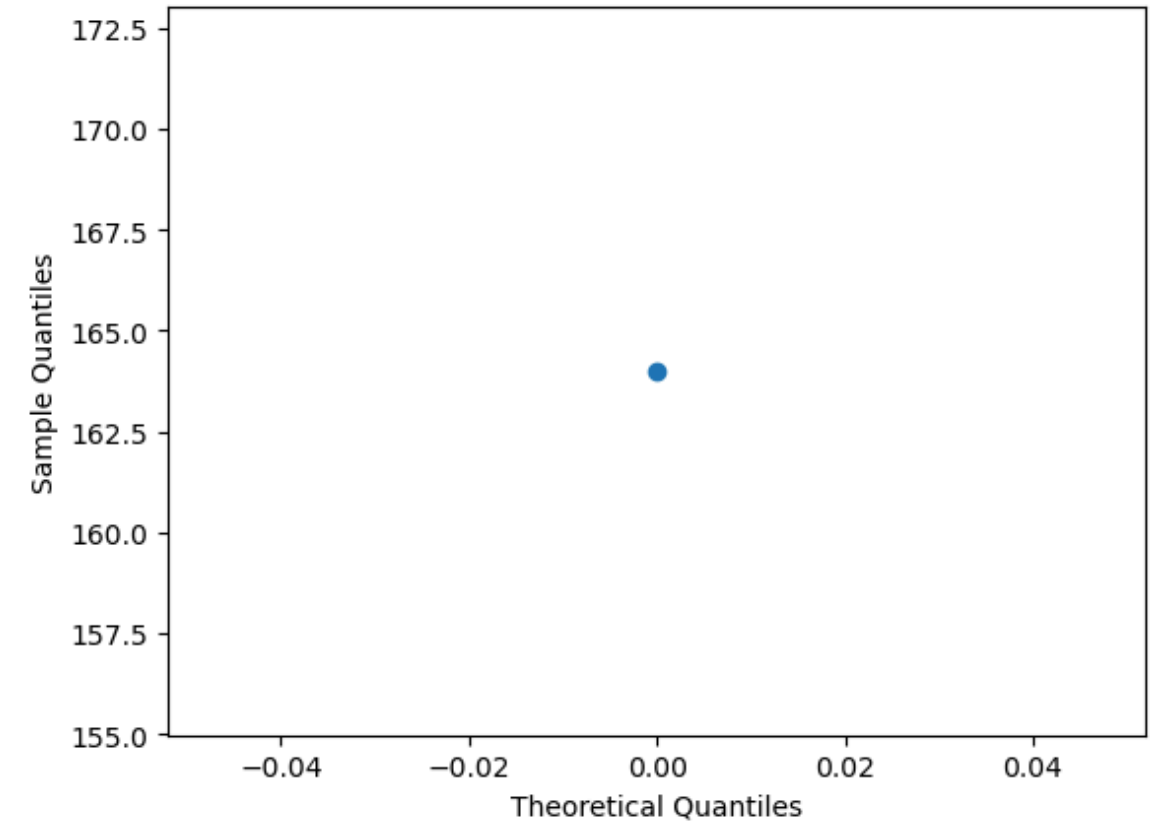
```
1.9541397976486484e-06
Normally distributed
```

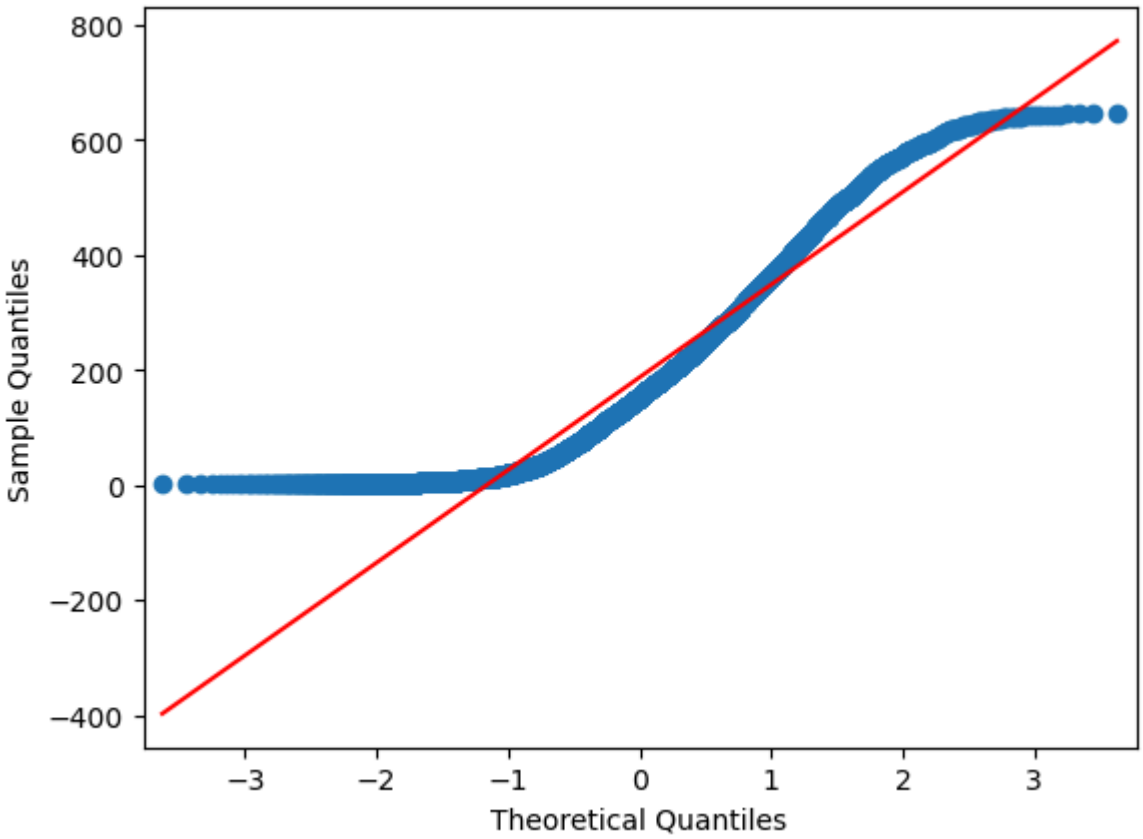
In [43]:
```
from scipy.stats import levene
lstat,pvalue=levene(weather_1,weather_2,weather_3,weather_4)
print(pvalue)
if pvalue<0.05:
  print("Reject Ho,variance is not equal ")
else:
  print("Fail to reject Ho,variance is equal ")
```

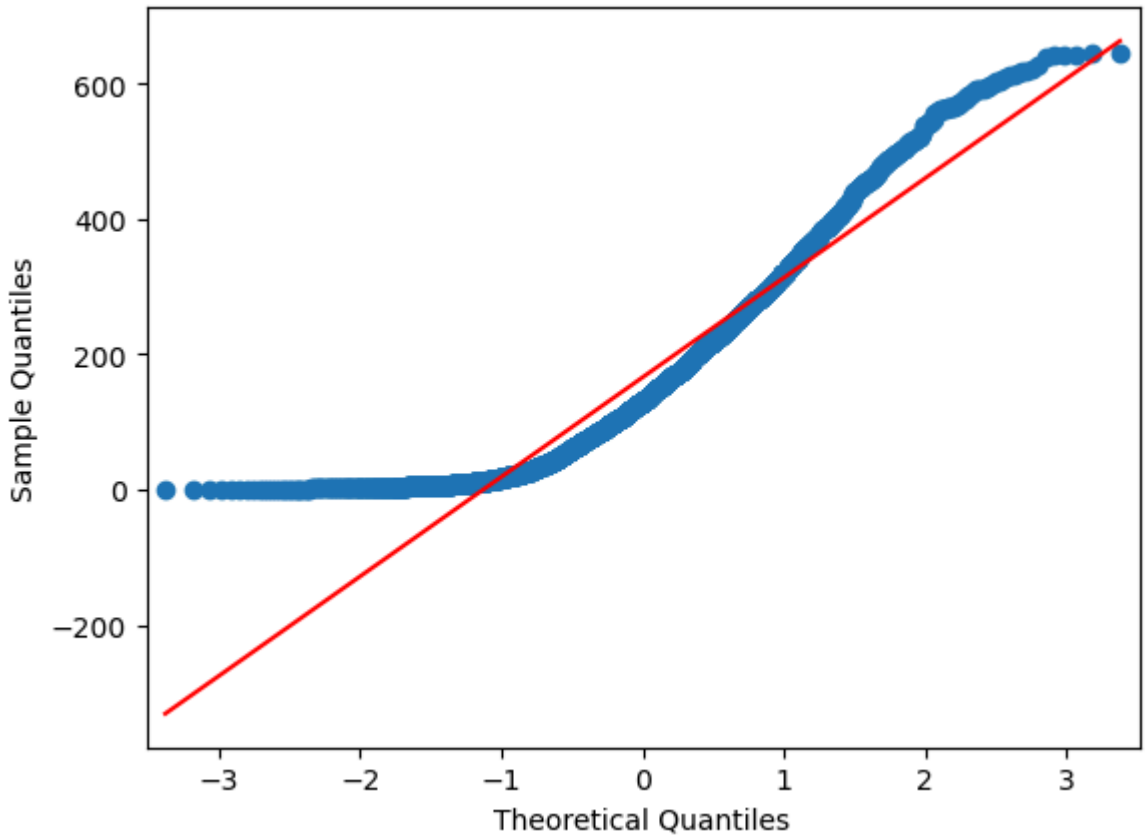
```
2.0385458926668884e-37
Reject Ho,variance is not equal
```

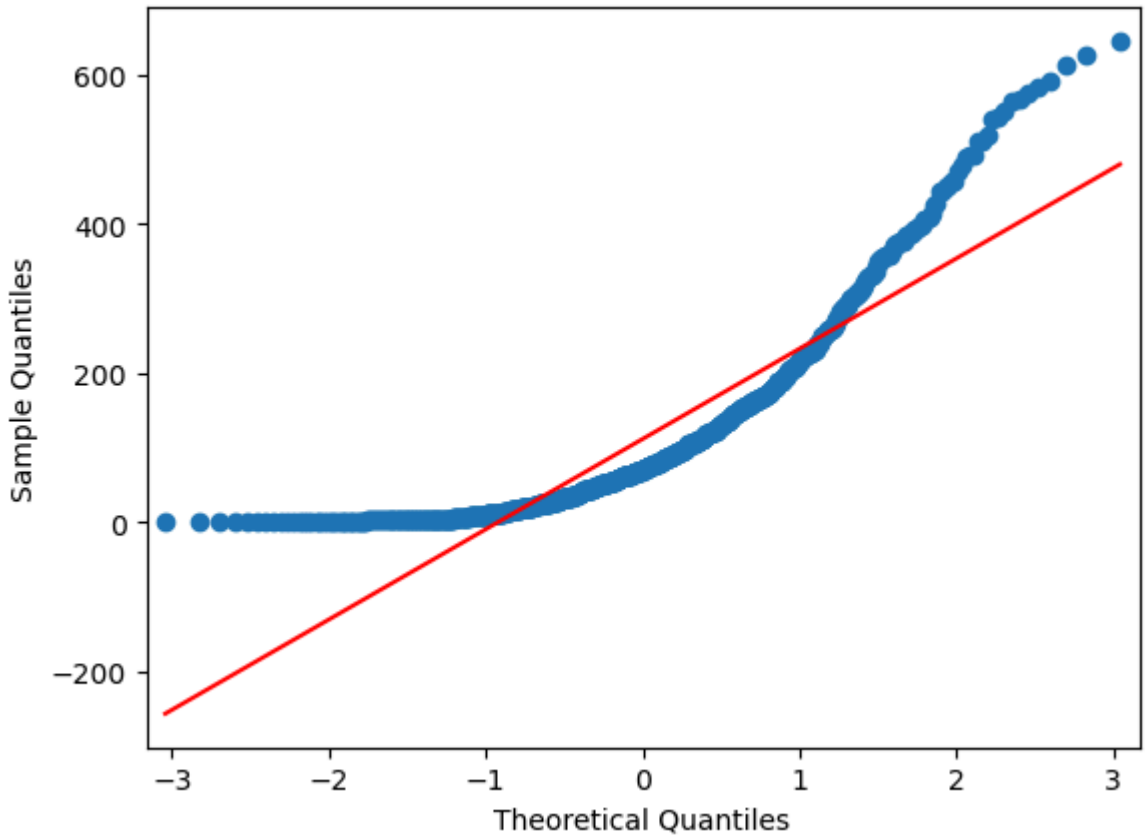
In [36]:
```
from statsmodels.graphics.gofplots import qqplot
qqplot(weather_1,line="s")
qqplot(weather_2,line="s")
qqplot(weather_3,line="s")
qqplot(weather_4,line="s")
```

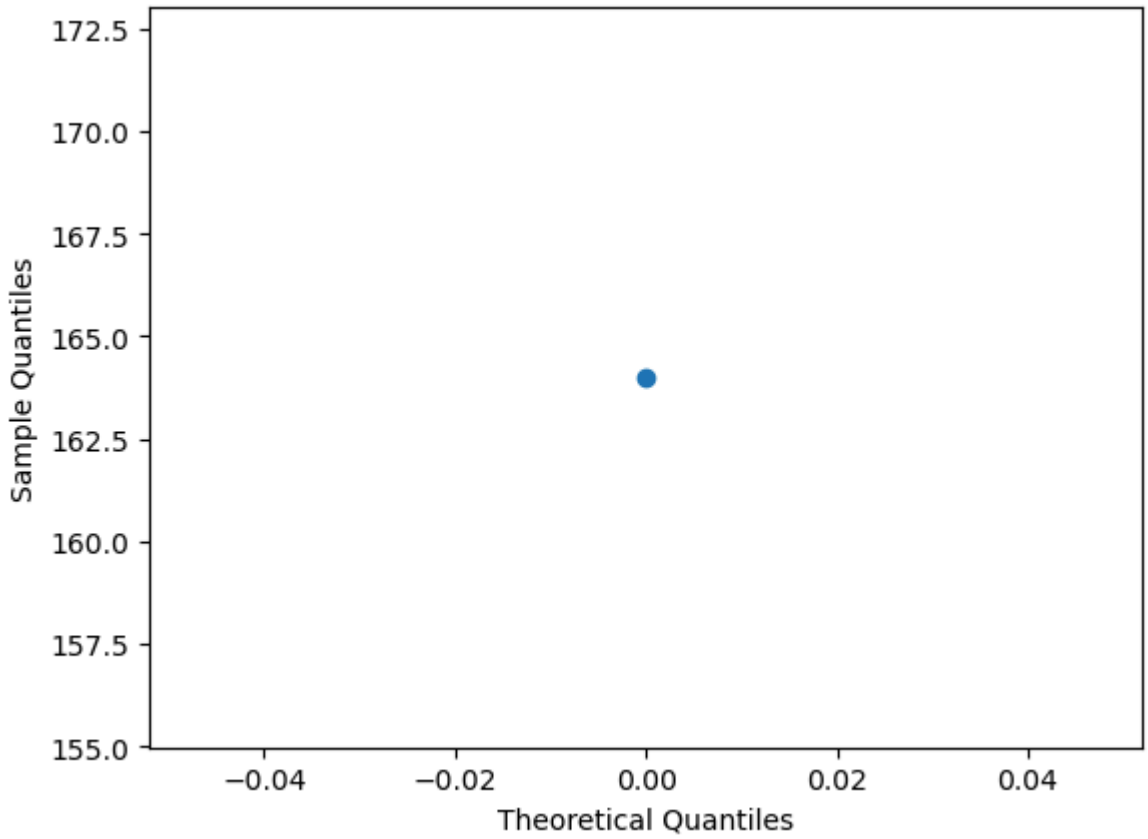
Out[36]:


172.5
170.0
167.5
165.0
162.5
160.0
157.5
155.0
Sample Quantiles
−0.04
−0.02
0.00
0.02
0.04
Theoretical Quantiles


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


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


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



Number of Bicycles Rented in Different Seasons

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

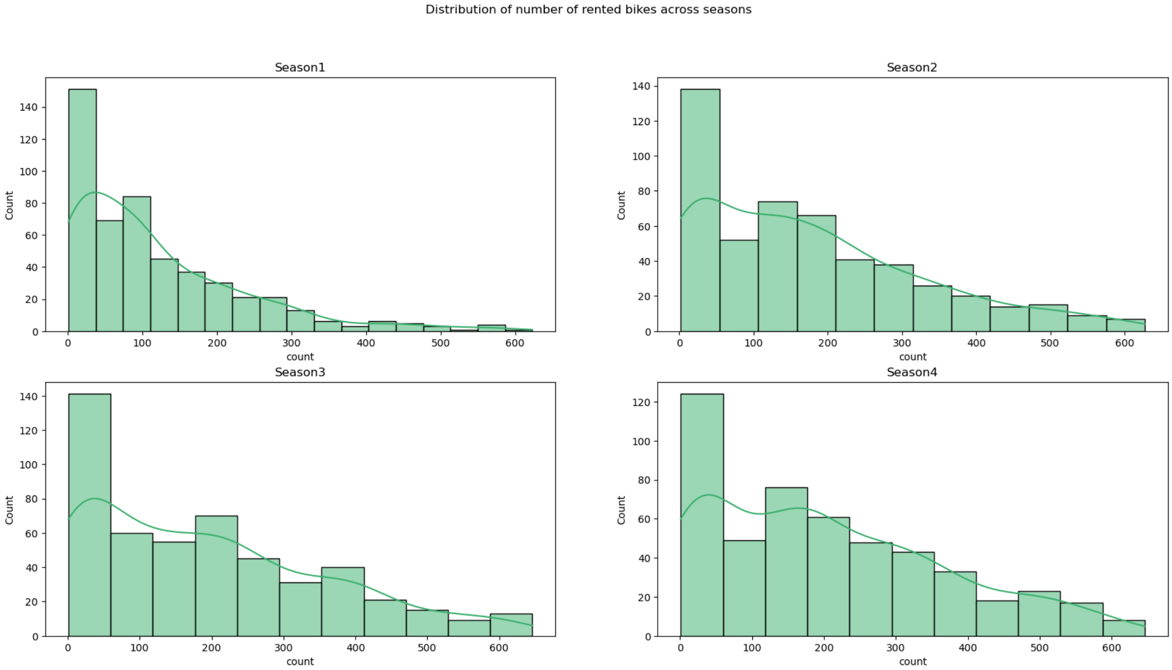
Out[25]:
```
season
Spring    2670
Winter    2664
Summer    2633
Fall      2616
Name: count, dtype: int64
```

In [45]:
```
season1 = df[df['season']=="Spring"]['count'].sample(500)
season2 = df[df['season']=="Winter"]['count'].sample(500)
season3 = df[df['season']=="Summer"]['count'].sample(500)
season4 = df[df['season']=="Fall"]['count'].sample(500)
```

In [46]:
```
plt.figure(figsize=(20,10))
#histogram for winter season
plt.subplot(2,2,1)
sns.histplot(season1,kde=True,color='mediumseagreen')
plt.title('Season1')
#histogram for fall season
plt.subplot(2,2,2)
sns.histplot(season2,kde=True,color='mediumseagreen')
plt.title('Season2')
#histogram for summer season
plt.subplot(2,2,3)
sns.histplot(season3,kde=True,color='mediumseagreen')
plt.title('Season3')
```

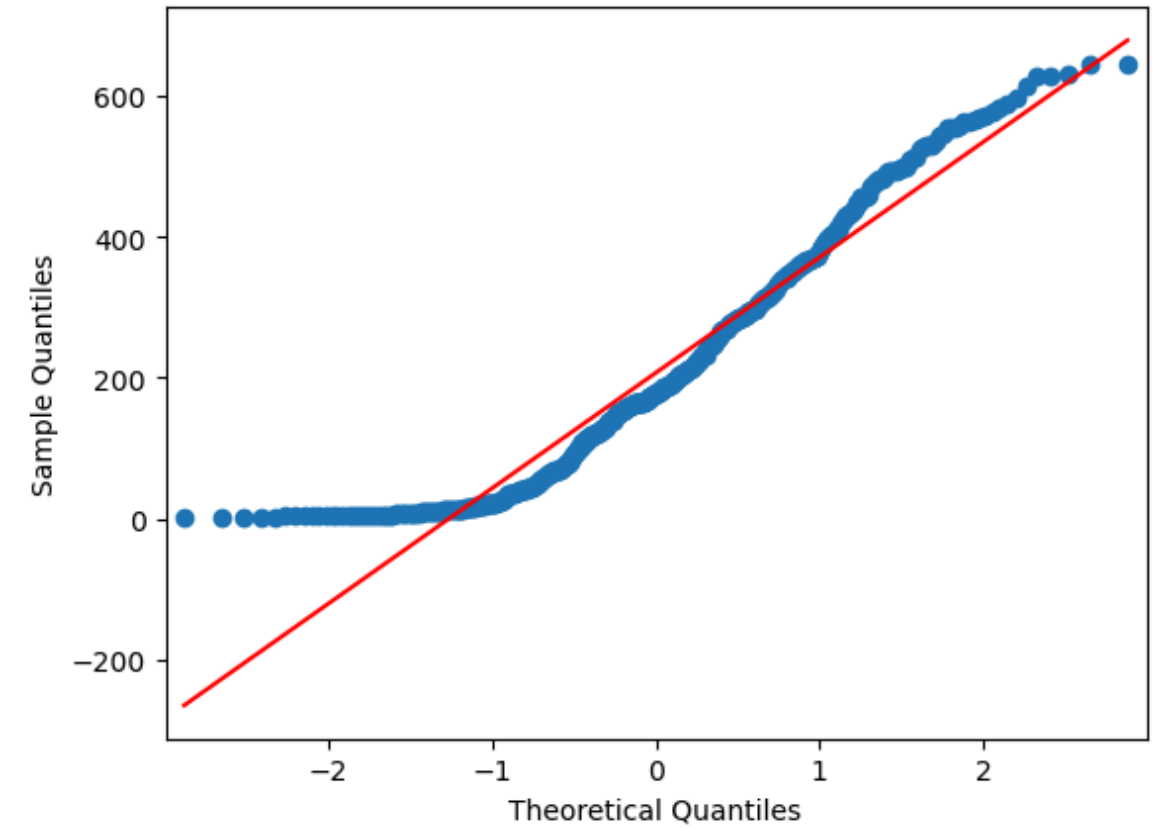
```
#histogram for spring season
plt.subplot(2,2,4)
sns.histplot(season4,kde=True,color='mediumseagreen')
plt.title('Season4')
plt.suptitle('Distribution of number of rented bikes across seasons')
plt.show()
```

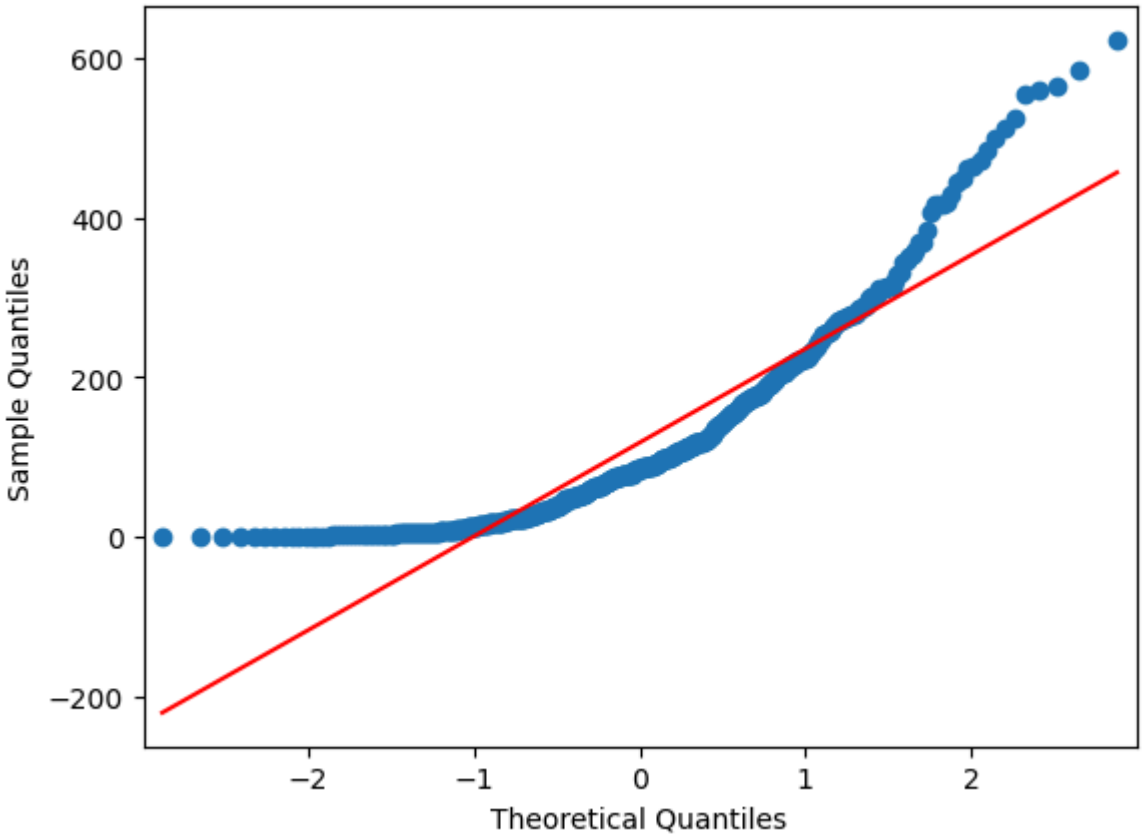
```
C:\Users\Pawan Kumar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future versi
on. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Pawan Kumar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future versi
on. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Pawan Kumar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future versi
on. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Pawan Kumar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future versi
on. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

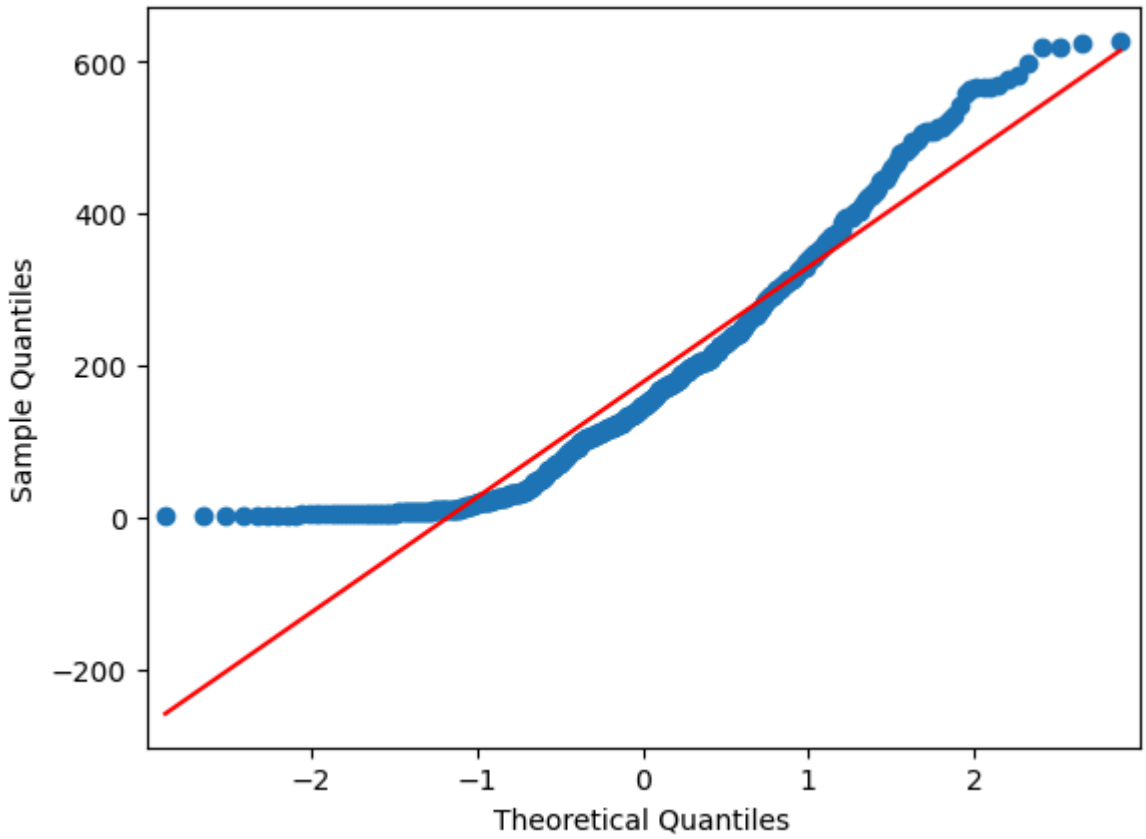


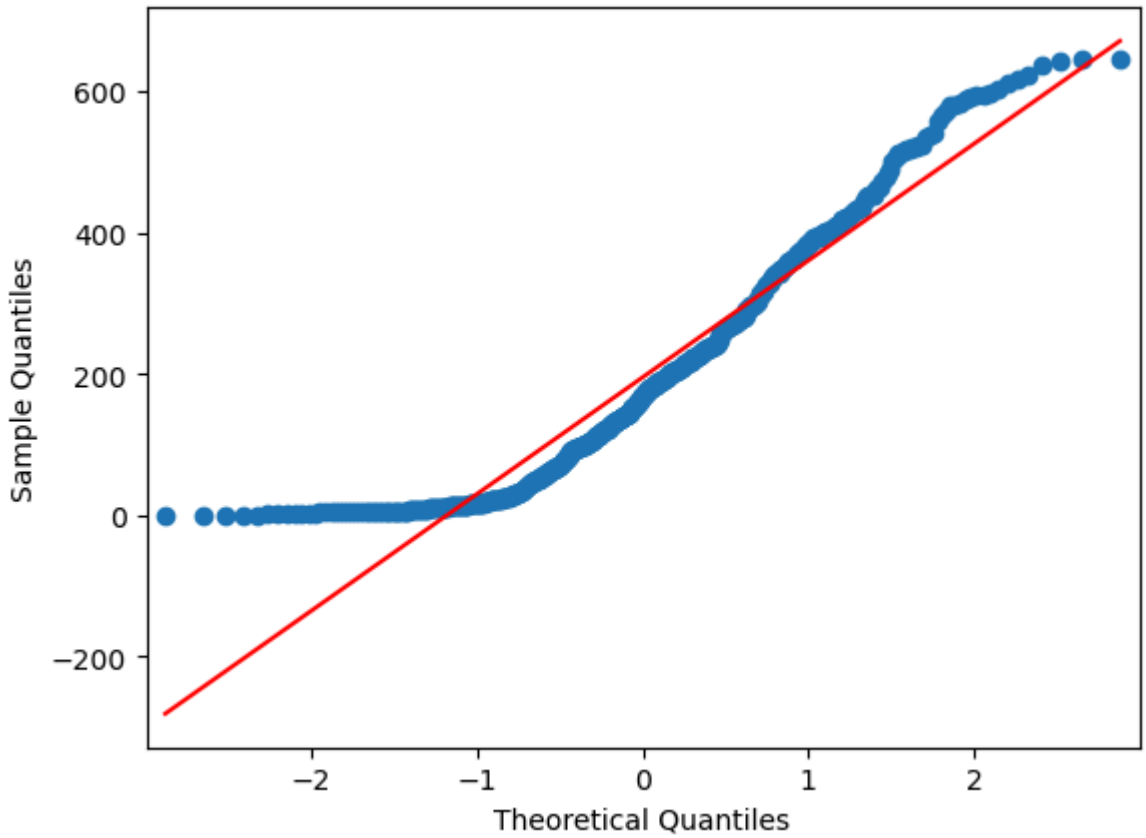
In [50]:
```
from statsmodels.graphics.gofplots import qqplot
qqplot(season1,line="s")
qqplot(season2,line="s")
qqplot(season3,line="s")
qqplot(season4,line="s")
```

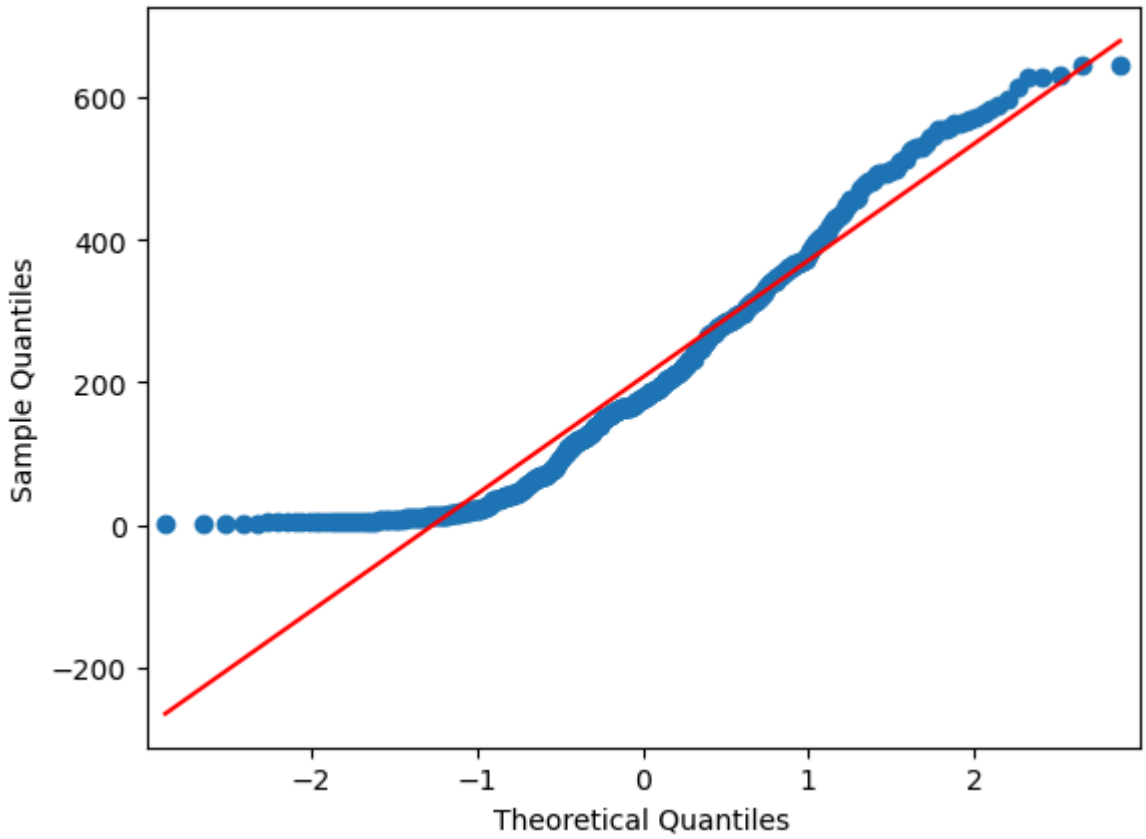
Out[50]:


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


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


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


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



In [51]:
```
h_stat, p_value = kruskal(season1,season2,season3,season4)
print("H statistic vale: ", h_stat)
print("P Value: ", p_value)
# Significance level is 5%
alpha = 0.05
if p_value < alpha:
    print("Reject Ho: The mean number of electric cycles rented differs across s
else:
    print("Failed to reject Ho: The mean number of electric cycles rented is the
```

```
H statistic vale:  92.58271938974767
P Value:  6.105765141635713e-20
Reject Ho: The mean number of electric cycles rented differs across seasons
```

In [52]:
```
# Performing Anova test for effect of Season
f_stat, p_value = f_oneway(season1,season2, season3,season4)
print("f_stat: ", f_stat)
print("p-value: ", p_value)

# Significance level is 5%
alpha = 0.05
if p_value < alpha:
    print("Reject Ho: The mean number of electric cycles rented differs across s
else:
    print("Failed to reject Ho: The mean number of electric cycles rented is the
```

```
f_stat:  34.32126380422523
p-value:  1.2617006587752654e-21
Reject Ho: The mean number of electric cycles rented differs across seasons
```

In [53]:
```
from scipy.stats import levene
lstat,pvalue=levene(season1,season2, season3,season4)
print(pvalue)
if pvalue<0.05:
```

```
  print("Reject Ho,variance is not equal ")
else:
  print("Fail to reject Ho,variance is equal ")
```

```
9.30264271270202e-19
Reject Ho,variance is not equal
```

Check if the Weather conditions are significantly different during different Seasons

In [56]:
```
# Creating a contingency table of weather conditions vs seasons
contingency_table = pd.crosstab(df['season'], df['weather'])
# Perform Chi-square test of independence
chi2, p_value, dof, expected = chi2_contingency(contingency_table)
chi2, p_value, dof, expected
```

Out[56]:
```
(47.16590591959627,
 3.6550317439064896e-07,
 9,
 array([[1.72092904e+03, 6.84713219e+02, 2.10110555e+02, 2.47188888e-01],
        [1.75645280e+03, 6.98847208e+02, 2.14447699e+02, 2.52291411e-01],
        [1.73211244e+03, 6.89162808e+02, 2.11475952e+02, 2.48795238e-01],
        [1.75250572e+03, 6.97276765e+02, 2.13965794e+02, 2.51724464e-01]]))
```

In [57]:
```
# Checking for significance level of 5%
alpha = 0.05
if p_value < alpha:
    print("Reject Ho - Weather is dependent on the season")
else:
    print("Failed to reject Ho- Weather is independent of the Season")
```

```
Reject Ho - Weather is dependent on the season
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: