exhabvcvu

March 4, 2023

Yulu Project

0.1 1.Define Problem Statement and perform Exploratory Data Analysis

The Goal of this project is to provide data driven insights so as to improve the descision making and to increase revenue and identify bottleneck by checking which factors affecting the count(revenue).

[188]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency, f_oneway, ttest_ind
from scipy.stats import chi2, shapiro, boxcox
from scipy.stats import chisquare, kruskal
import statsmodels.api as sm
import scipy.stats as stats
```

[189]:
```
df=pd.read_csv('C:/Users/rahul.kumar/Downloads/yulu.csv')
```

[190]:
```
df.shape
```

[190]:
```
(10886, 12)
```

[191]:
```
df.head()
```

[191]:

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

	humidity	windspeed	casual	registered	count
0	81	0.0	3	13	16
1	80	0.0	8	32	40
2	80	0.0	5	27	32
3	75	0.0	3	10	13
4	75	0.0	0	1	1

[192]:
```
df.describe(include='all')
```

[192]:

	datetime	season	holiday	workingday	\
count	10886	10886.000000	10886.000000	10886.000000	
unique	10886	NaN	NaN	NaN	
top	2011-01-01 00:00:00	NaN	NaN	NaN	
freq	1	NaN	NaN	NaN	
mean	NaN	2.506614	0.028569	0.680875	
std	NaN	1.116174	0.166599	0.466159	
min	NaN	1.000000	0.000000	0.000000	
25%	NaN	2.000000	0.000000	0.000000	
50%	NaN	3.000000	0.000000	1.000000	
75%	NaN	4.000000	0.000000	1.000000	
max	NaN	4.000000	1.000000	1.000000	

	weather	temp	atemp	humidity	windspeed	\
count	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	
unique	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	
mean	1.418427	20.23086	23.655084	61.886460	12.799395	
std	0.633839	7.79159	8.474601	19.245033	8.164537	
min	1.000000	0.82000	0.760000	0.000000	0.000000	
25%	1.000000	13.94000	16.665000	47.000000	7.001500	
50%	1.000000	20.50000	24.240000	62.000000	12.998000	
75%	2.000000	26.24000	31.060000	77.000000	16.997900	
max	4.000000	41.00000	45.455000	100.000000	56.996900	

	casual	registered	count
count	10886.000000	10886.000000	10886.000000
unique	NaN	NaN	NaN
top	NaN	NaN	NaN
freq	NaN	NaN	NaN
mean	36.021955	155.552177	191.574132
std	49.960477	151.039033	181.144454
min	0.000000	0.000000	1.000000
25%	4.000000	36.000000	42.000000
50%	17.000000	118.000000	145.000000
75%	49.000000	222.000000	284.000000
max	367.000000	886.000000	977.000000

[193]:
```
df.nunique()
```

[193]:
```
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
```

[194]:
```
columns=list(df)
for column in columns:
    print(column)
    print(df[column].value_counts())
    print('.....................................................................
 ↪.....................................')
```

```
datetime
2011-01-01 00:00:00    1
2012-05-01 21:00:00    1
2012-05-01 13:00:00    1
2012-05-01 14:00:00    1
2012-05-01 15:00:00    1
                      ..
2011-09-02 04:00:00    1
2011-09-02 05:00:00    1
2011-09-02 06:00:00    1
2011-09-02 07:00:00    1
2012-12-19 23:00:00    1
Name: datetime, Length: 10886, dtype: int64
…
…
season
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
…
…
holiday
0    10575
1      311
Name: holiday, dtype: int64
…
…
workingday
```

```
1    7412
0    3474
Name: workingday, dtype: int64
…
…
weather
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
…
…
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: temp, dtype: int64
…
…
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: atemp, dtype: int64
…
…
humidity
88    368
94    324
83    316
87    289
70    259
     …
8       1
10      1
97      1
96      1
91      1
Name: humidity, Length: 89, dtype: int64
…
```

```
…
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: windspeed, dtype: int64
…
…
casual
0      986
1      667
2      487
3      438
4      354
      …
332      1
361      1
356      1
331      1
304      1
Name: casual, Length: 309, dtype: int64
…
…
```

```
registered
3      195
4      190
5      177
6      155
2      150
      …
570      1
422      1
678      1
565      1
636      1
Name: registered, Length: 731, dtype: int64
…
…
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
…
…
```

[195]:
```
df.isna().sum()
```

[195]:
```
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
```

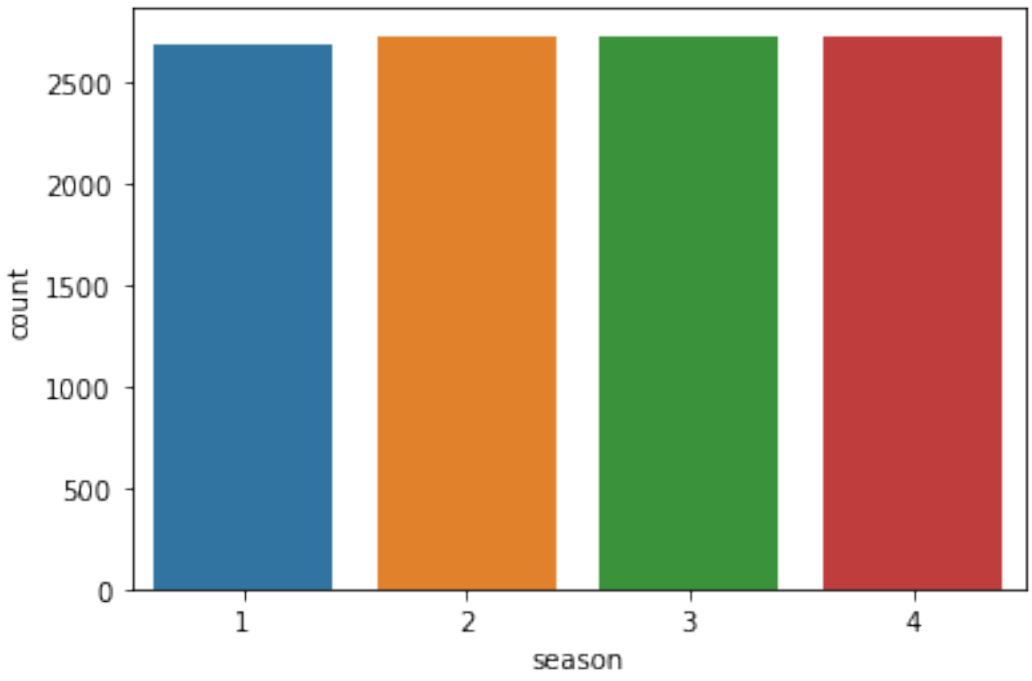
[196]:
```
df['datetime'] = df['datetime'].astype('datetime64[ns]')
```

[197]:
```
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
```

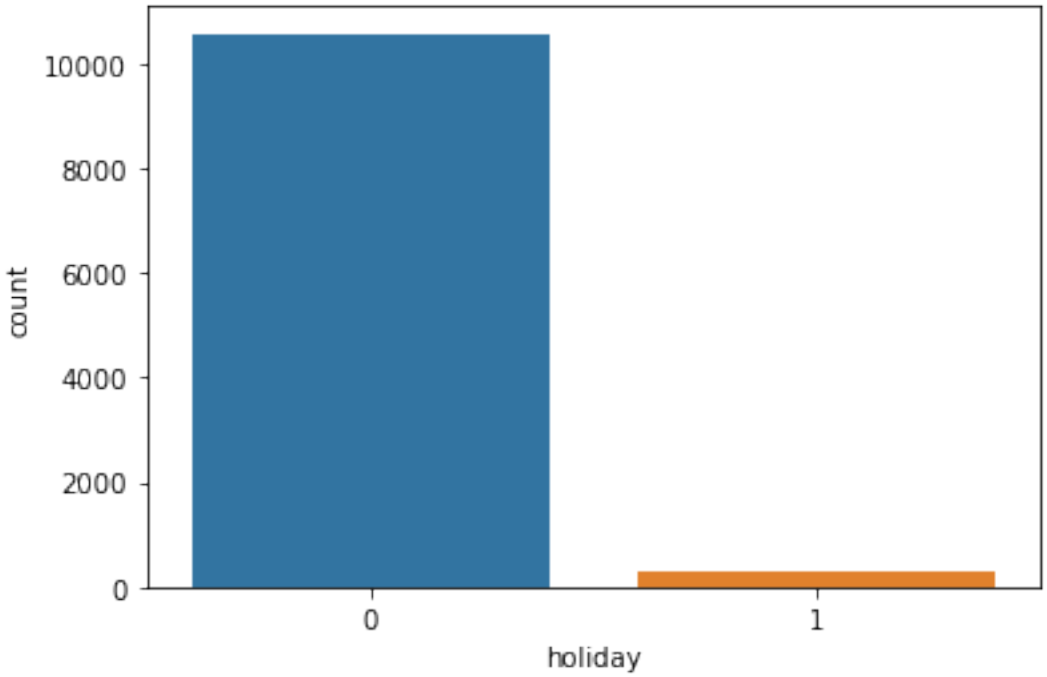
[198]:
```
sns.countplot(x='season',data=df)
```

[198]: <AxesSubplot:xlabel='season', ylabel='count'>



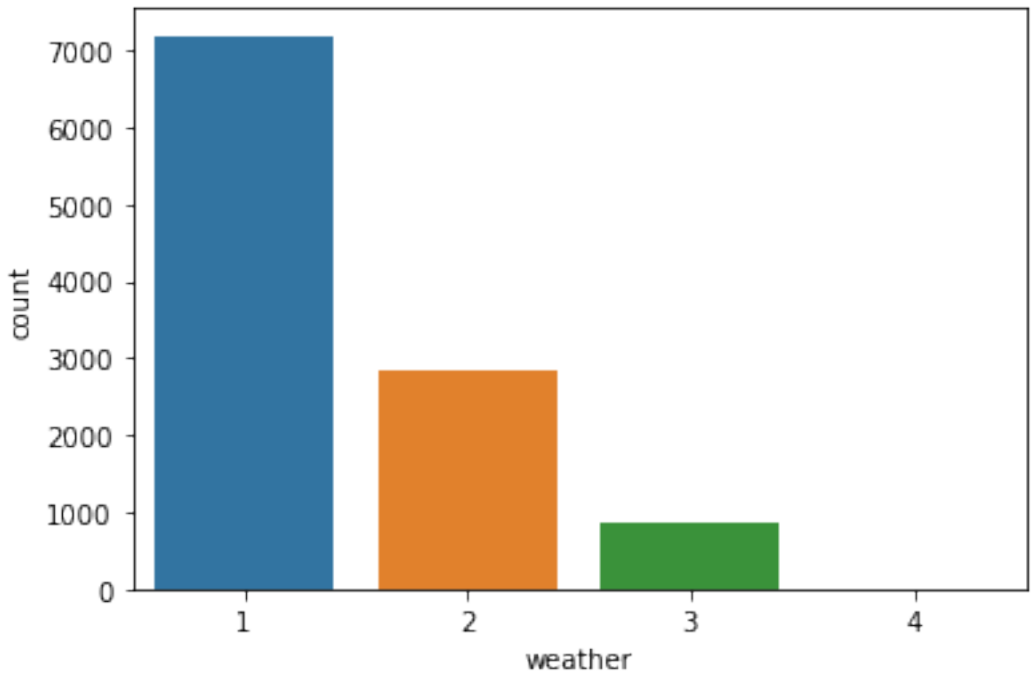
[199]:
```
sns.countplot(x='holiday',data=df)
```

[199]: <AxesSubplot:xlabel='holiday', ylabel='count'>



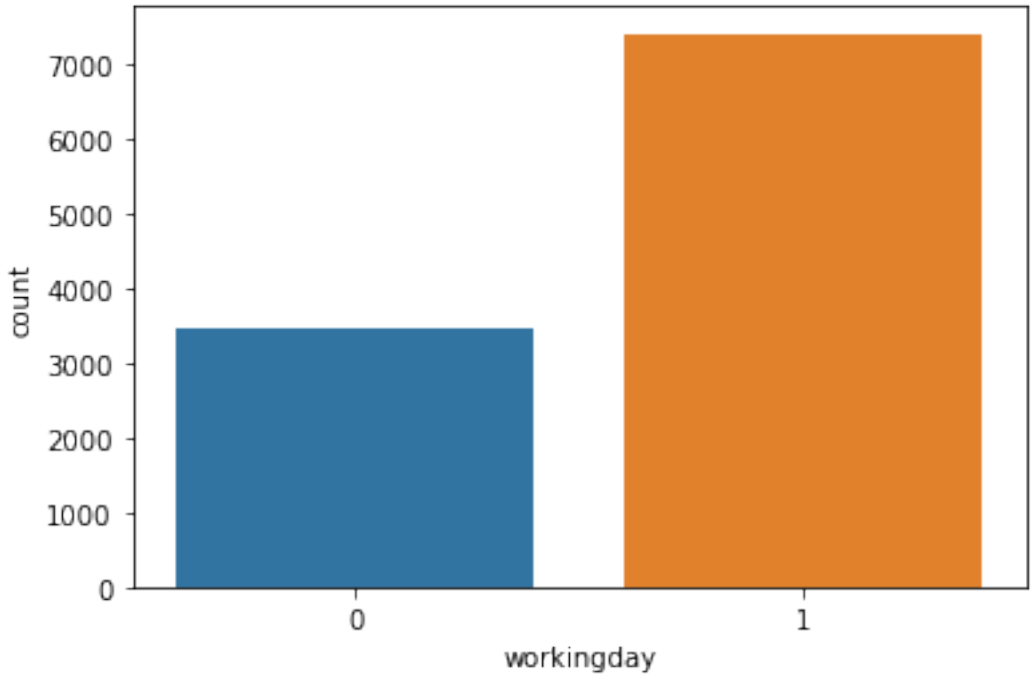
[200]:
```
sns.countplot(x='weather',data=df)
```

[200]: <AxesSubplot:xlabel='weather', ylabel='count'>



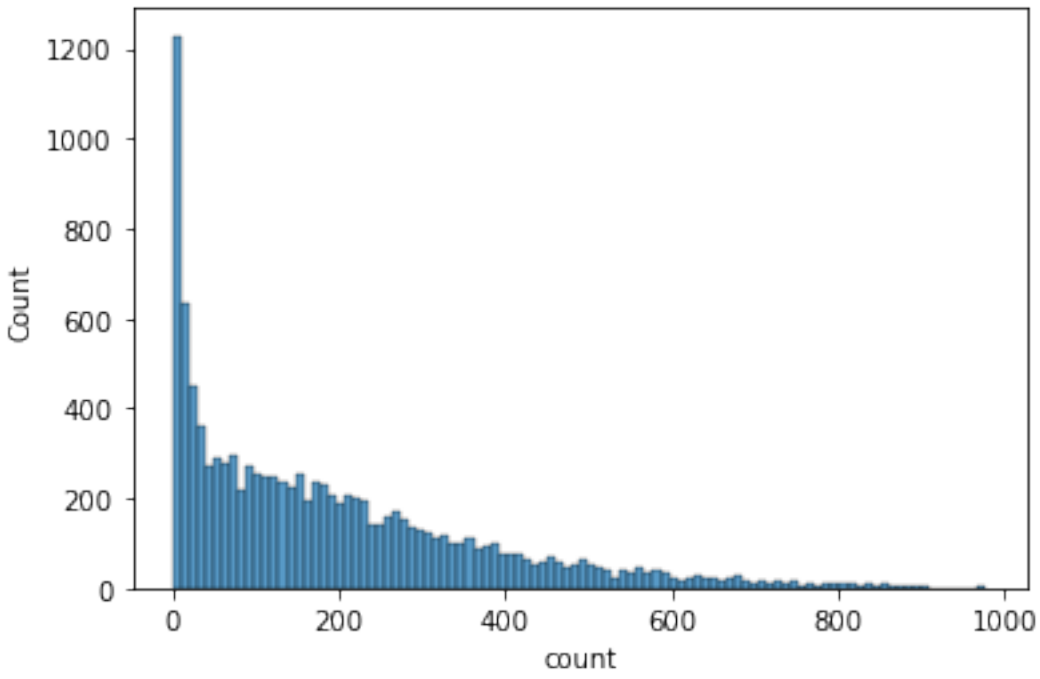
[201]:
```
sns.countplot(x='workingday',data=df)
```

[201]: <AxesSubplot:xlabel='workingday', ylabel='count'>



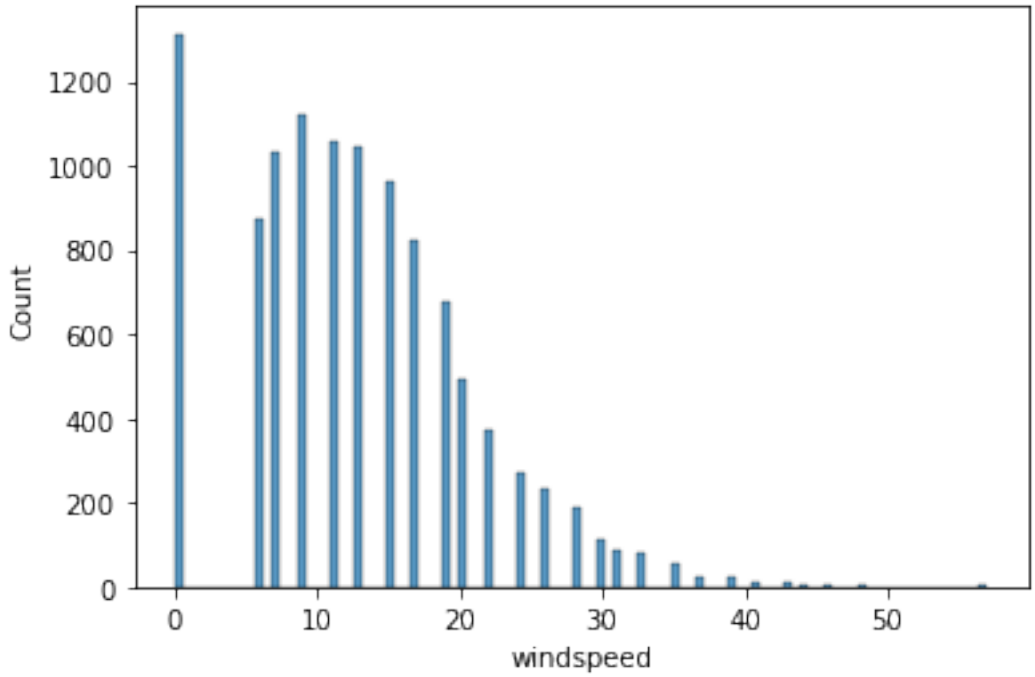
[202]:
```
sns.histplot(x='count',data=df,bins=100)
```

[202]:
```
<AxesSubplot:xlabel='count', ylabel='Count'>
```



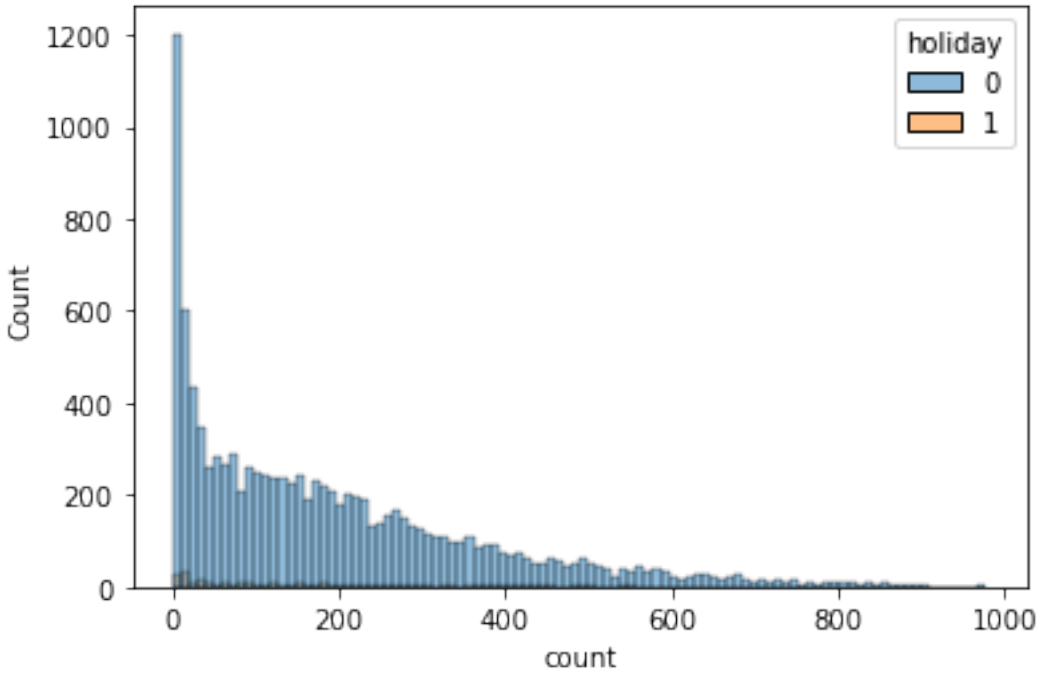
[203]:
```
sns.histplot(x='windspeed',data=df,bins=100)
```

[203]:
```
<AxesSubplot:xlabel='windspeed', ylabel='Count'>
```



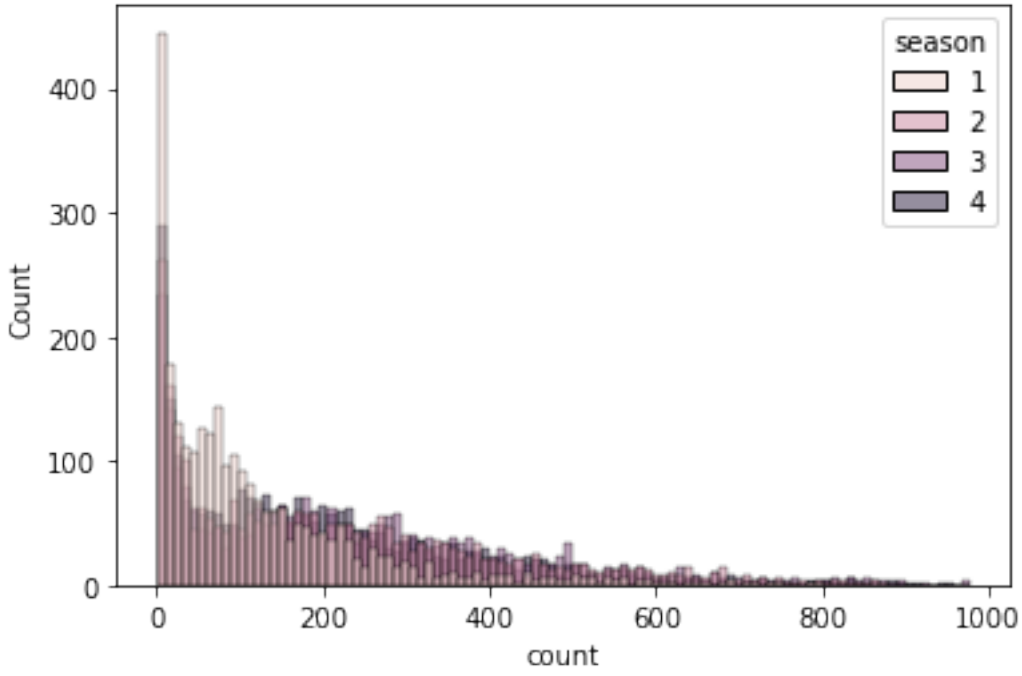
[204]:
```
sns.histplot(x='count',data=df,bins=100, hue='holiday')
```

[204]: <AxesSubplot:xlabel='count', ylabel='Count'>



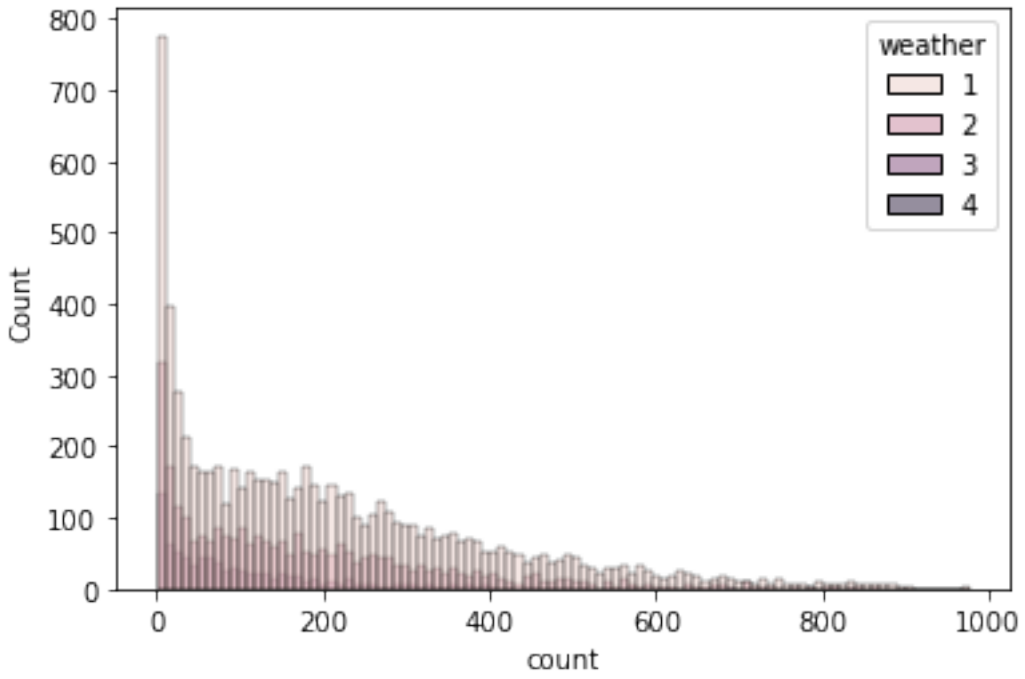
[205]:
```
sns.histplot(x='count',data=df,bins=100, hue='season')
```

[205]: <AxesSubplot:xlabel='count', ylabel='Count'>



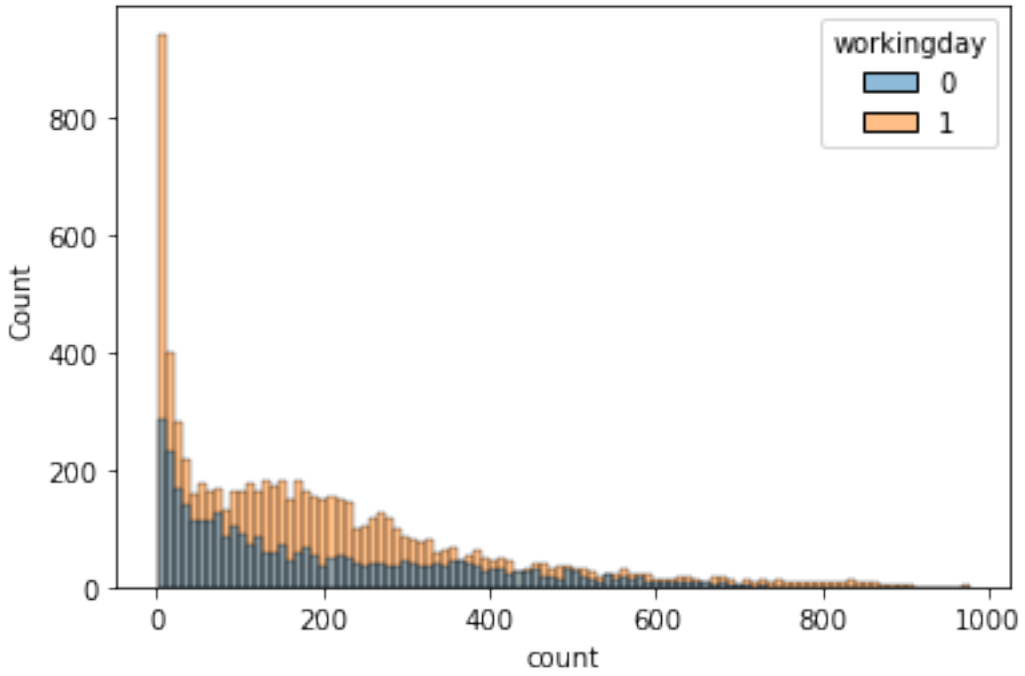
[206]:
```
sns.histplot(x='count',data=df,bins=100, hue='weather')
```

[206]: <AxesSubplot:xlabel='count', ylabel='Count'>



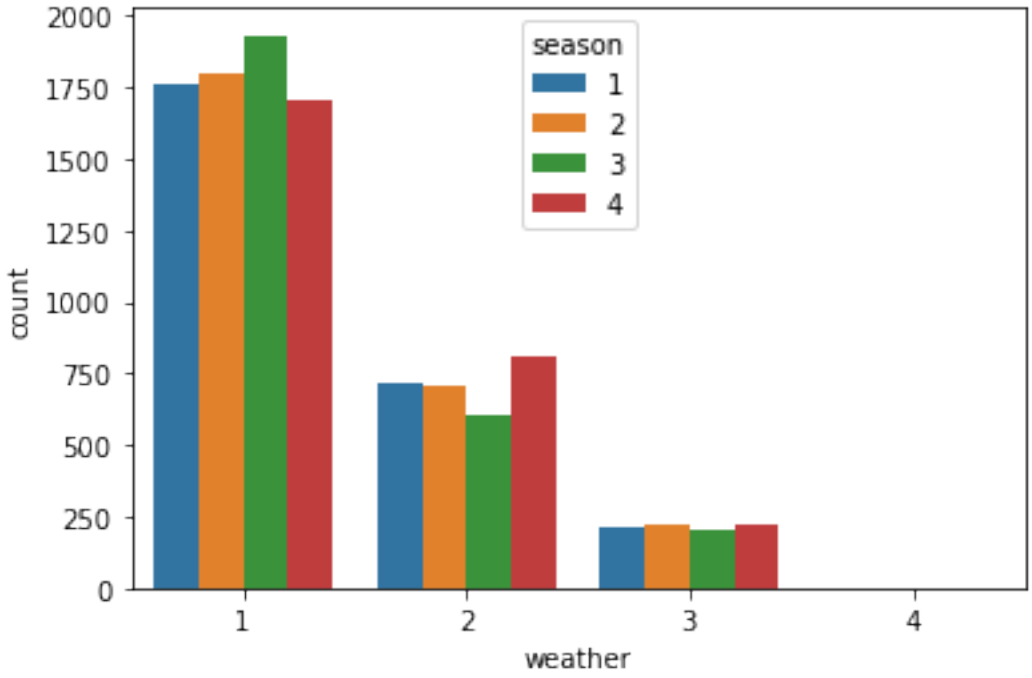
[207]:
```
sns.histplot(x='count',data=df,bins=100, hue='workingday')
```

[207]: <AxesSubplot:xlabel='count', ylabel='Count'>



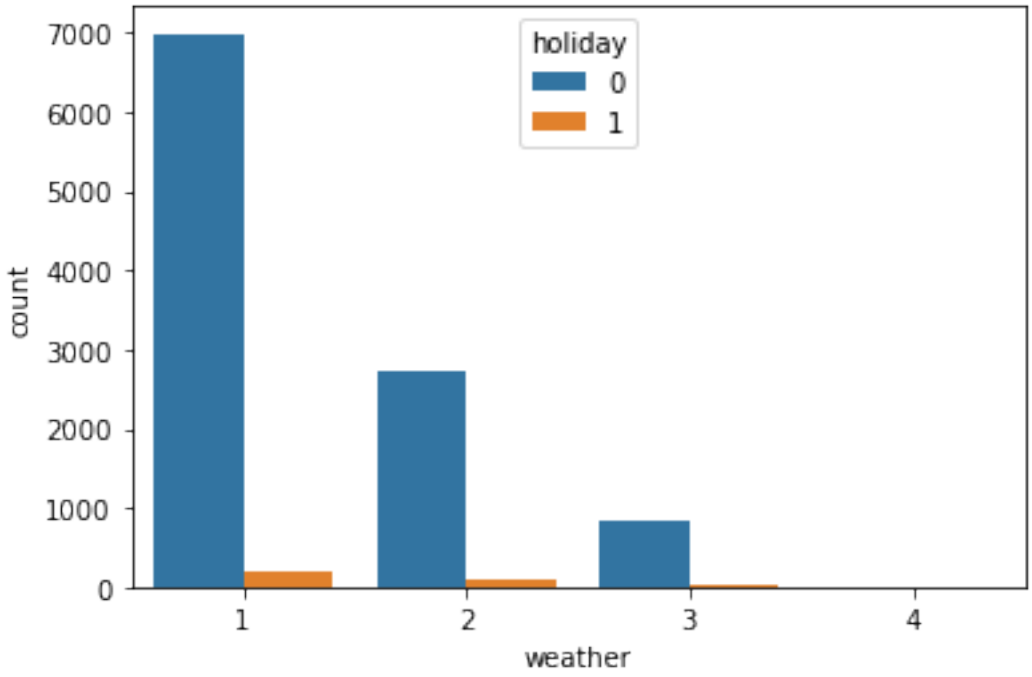
[208]:
```
sns.countplot(x='weather',data=df,hue='season')
```

[208]: <AxesSubplot:xlabel='weather', ylabel='count'>



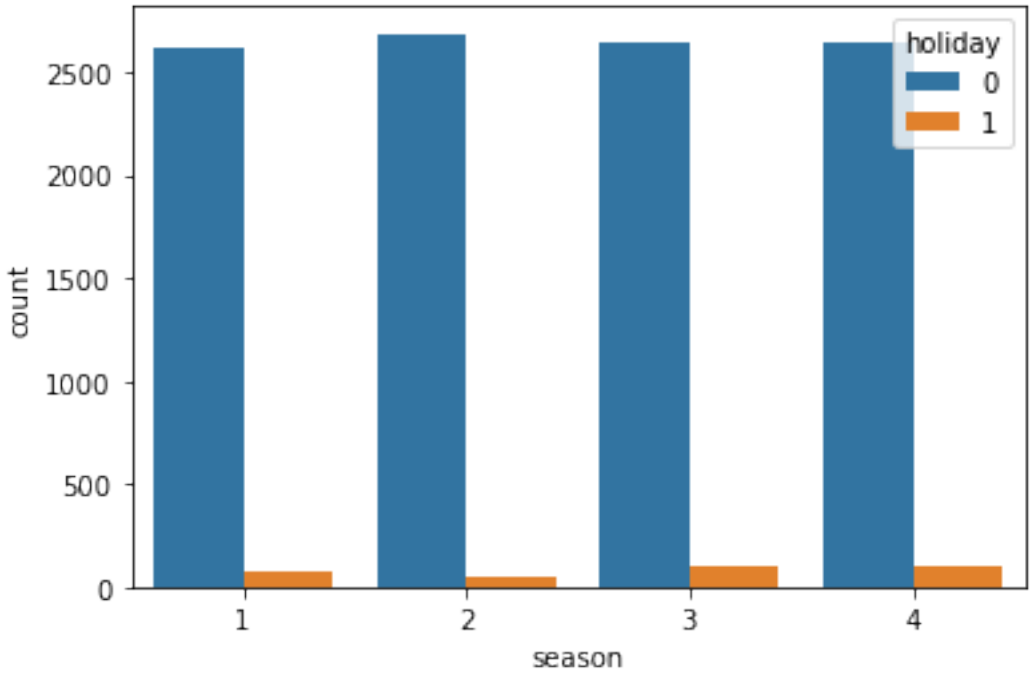
[209]:
```
sns.countplot(x='weather',data=df,hue='holiday')
```

[209]: <AxesSubplot:xlabel='weather', ylabel='count'>



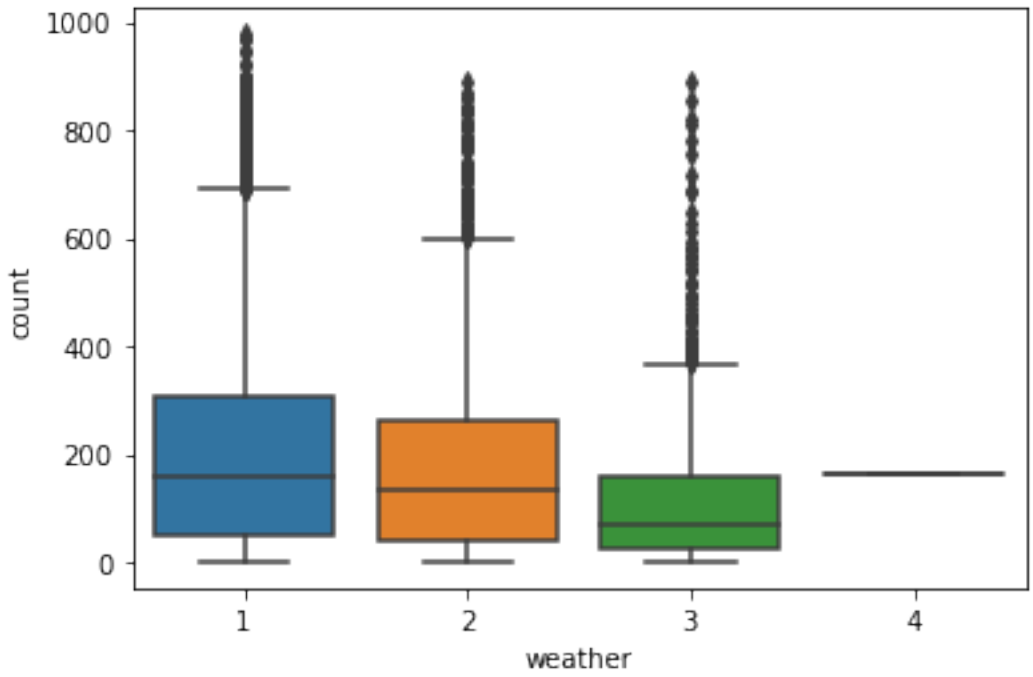
[210]:
```
sns.countplot(x='season',data=df,hue='holiday')
```

[210]: <AxesSubplot:xlabel='season', ylabel='count'>



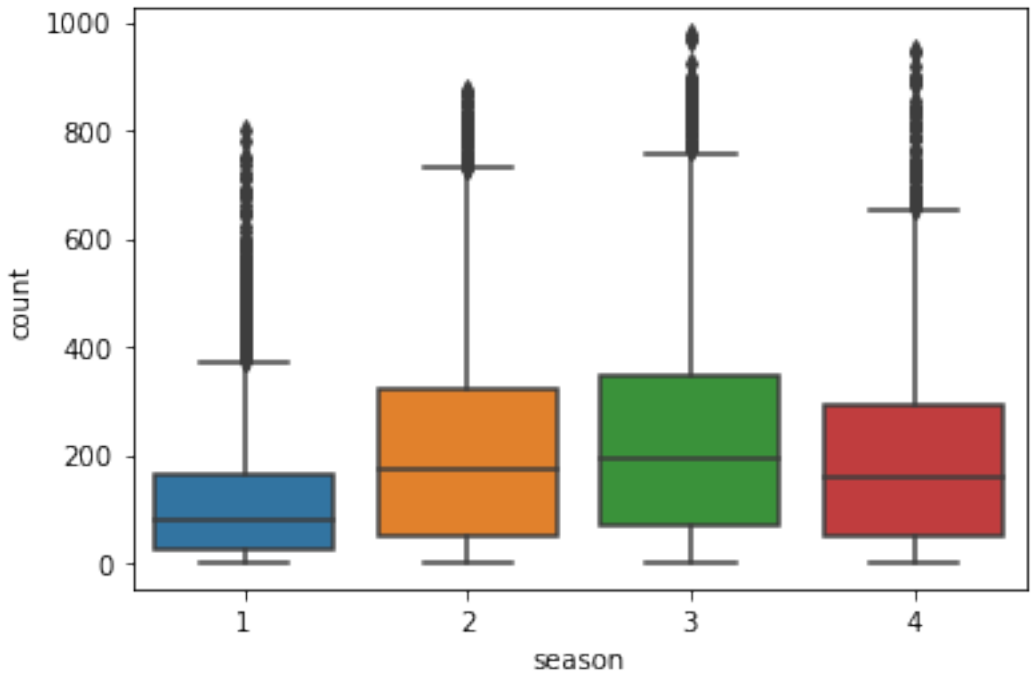
[211]:
```
sns.boxplot(data=df, y='count',x='weather')
```

[211]: <AxesSubplot:xlabel='weather', ylabel='count'>



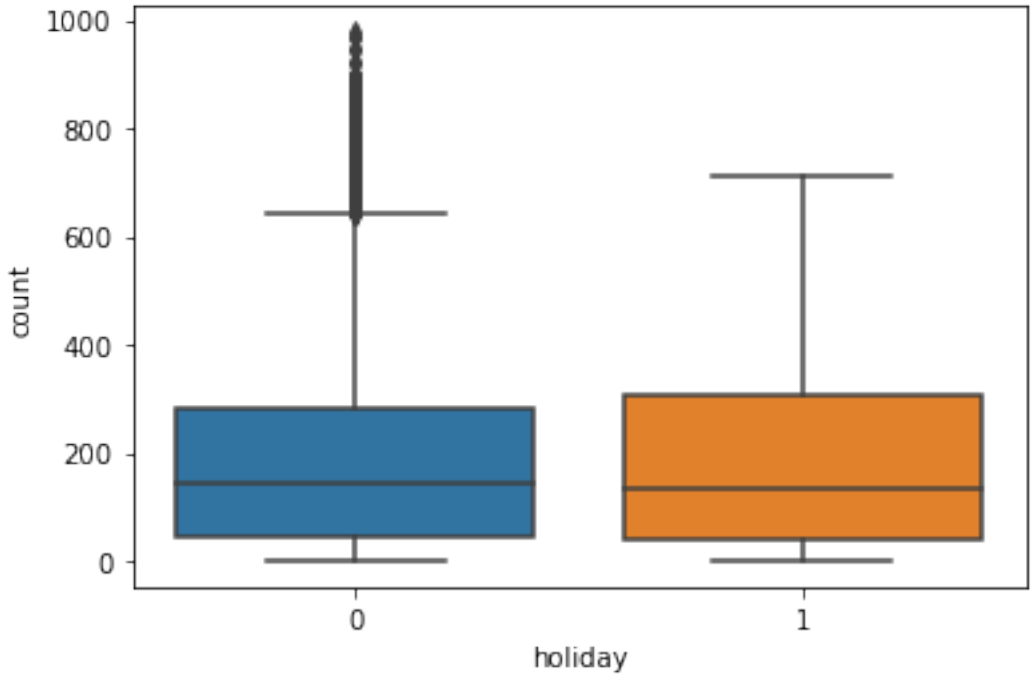
[212]:
```
sns.boxplot(data=df, y='count',x='season')
```

[212]: <AxesSubplot:xlabel='season', ylabel='count'>



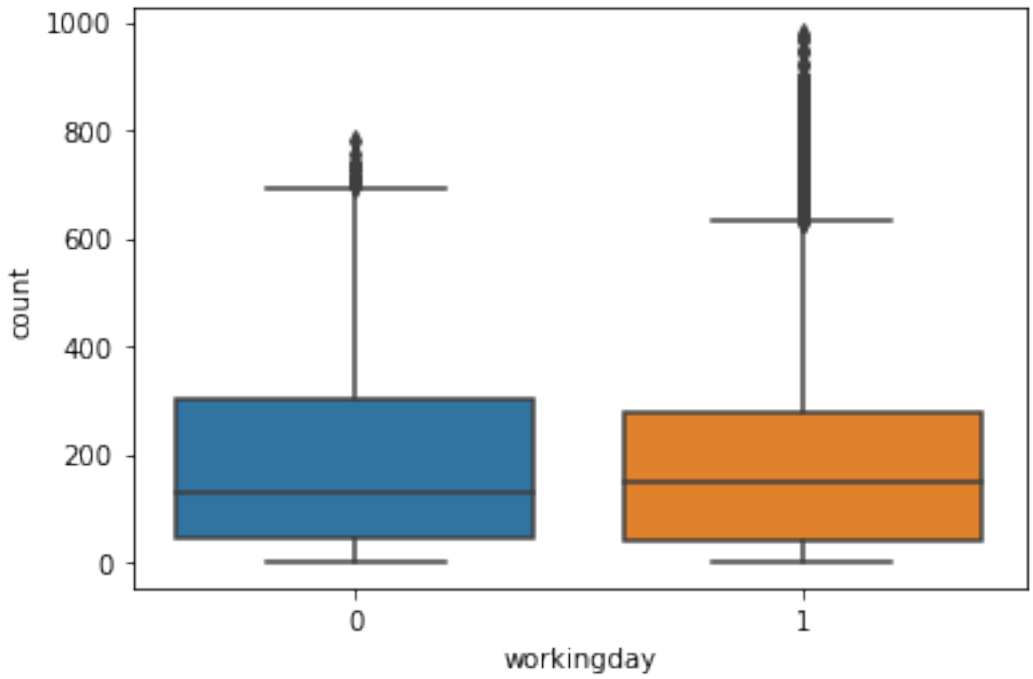
[213]:
```
sns.boxplot(data=df, y='count',x='holiday')
```

[213]:
```
<AxesSubplot:xlabel='holiday', ylabel='count'>
```



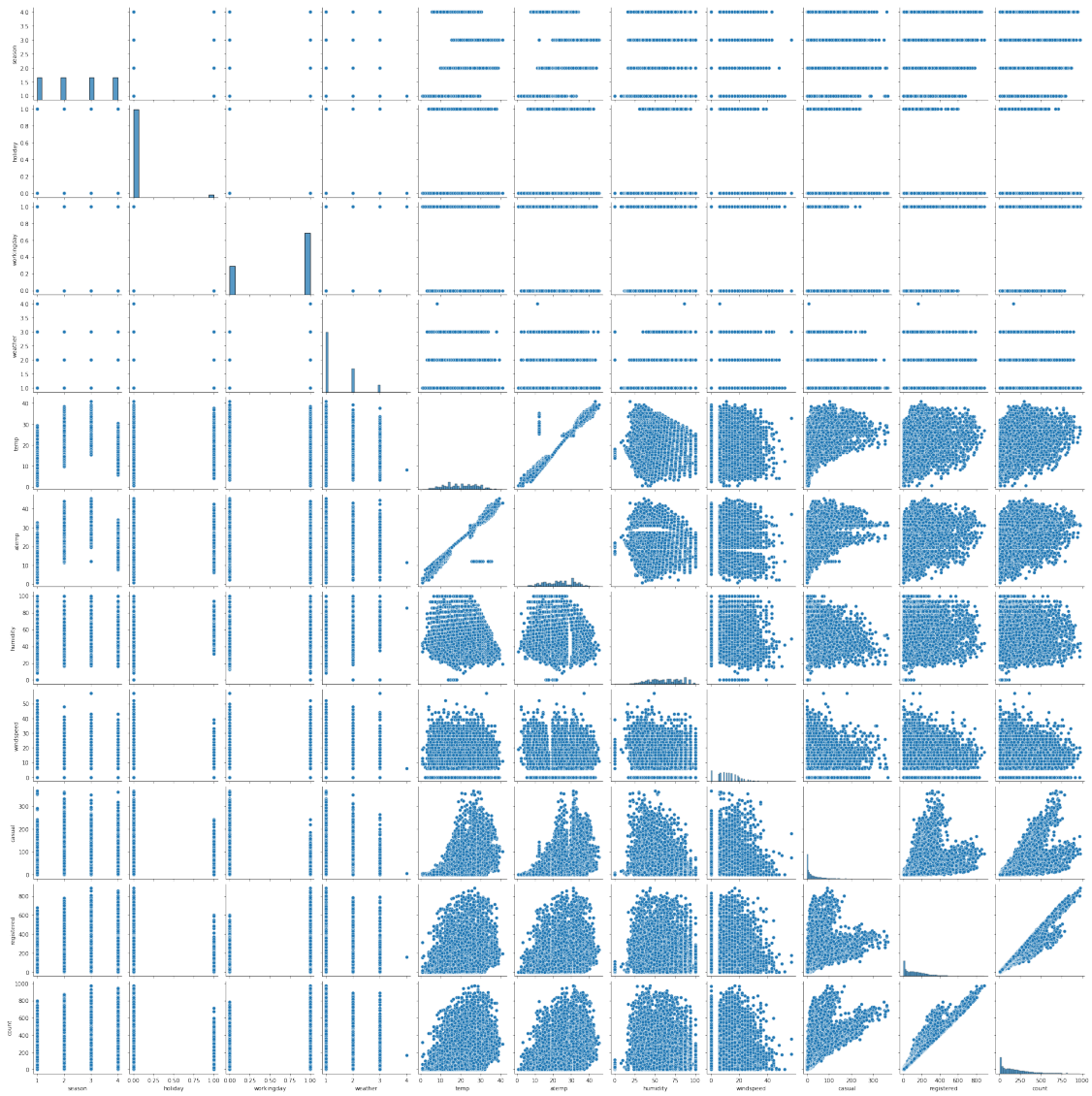
[214]:
```
sns.boxplot(data=df, y='count',x='workingday')
```

[214]: <AxesSubplot:xlabel='workingday', ylabel='count'>



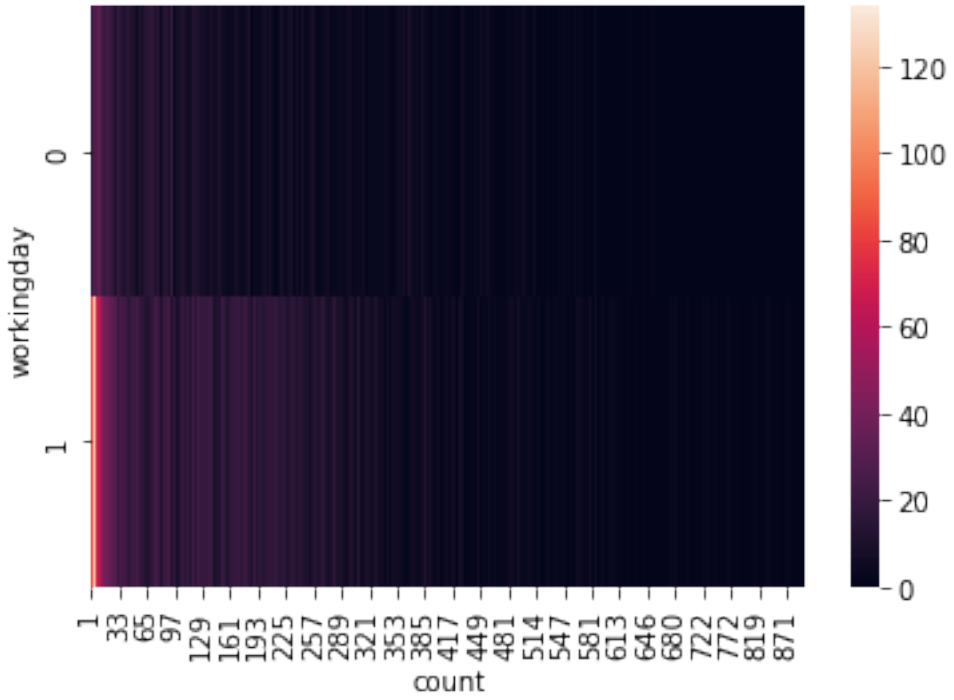
[215]:
```
sns.pairplot(data=df)
```

[215]: <seaborn.axisgrid.PairGrid at 0x1c227fb0d90>



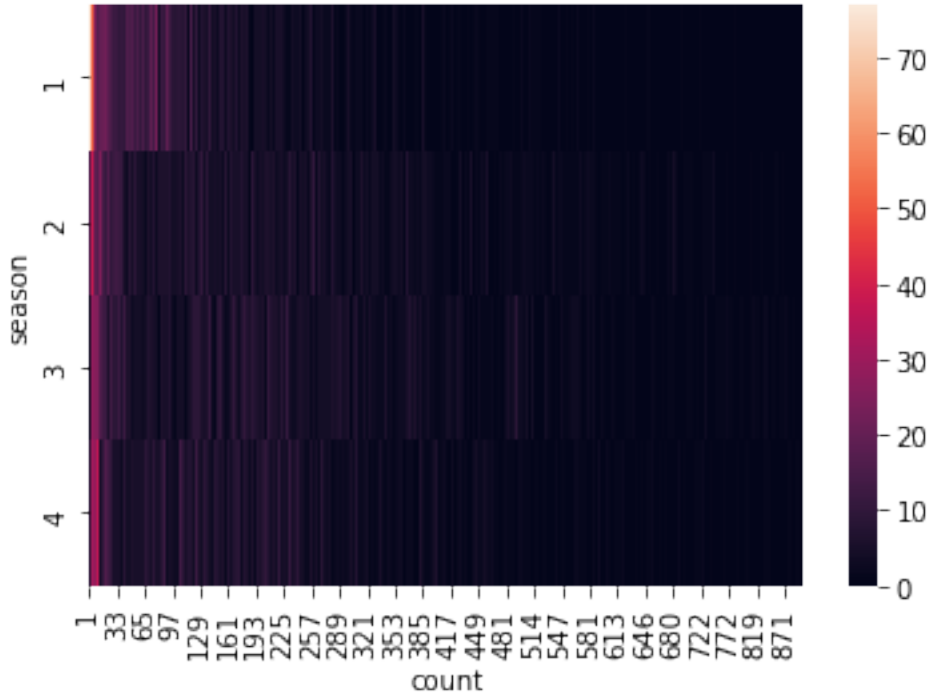
[216]:
```
sns.heatmap(pd.crosstab(index=df['workingday'], columns=df['count']))
```

[216]: <AxesSubplot:xlabel='count', ylabel='workingday'>



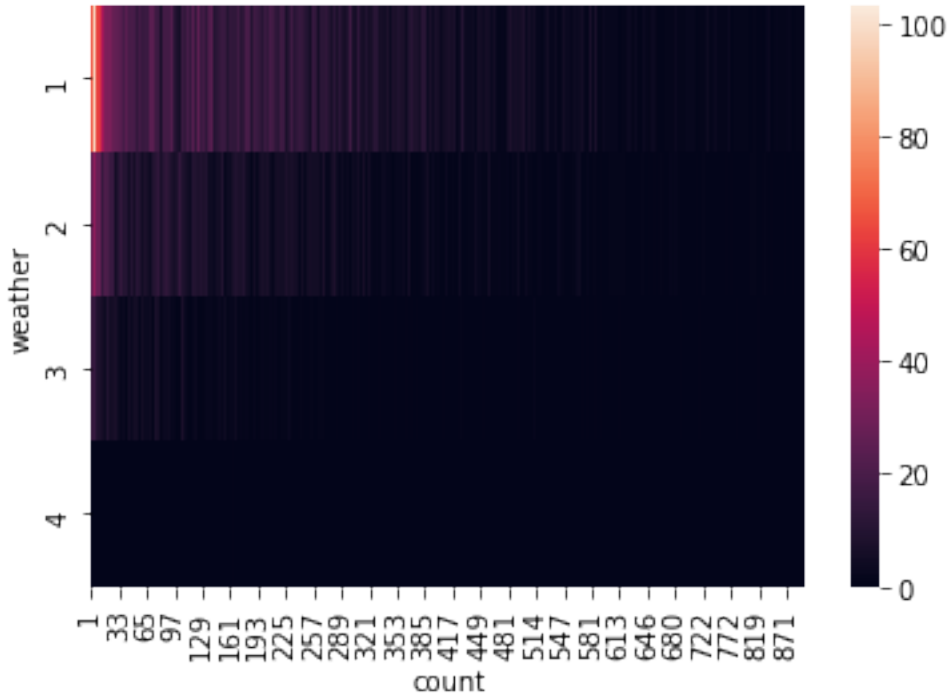
[217]:
```
sns.heatmap(pd.crosstab(index=df['season'], columns=df['count']))
```

[217]: <AxesSubplot:xlabel='count', ylabel='season'>



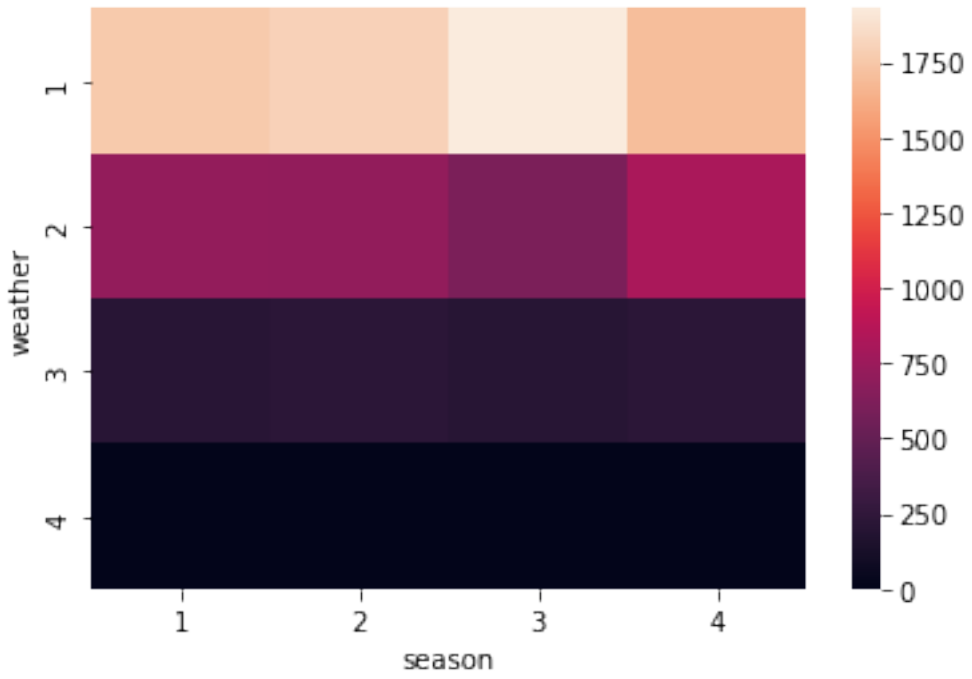
[218]:
```
sns.heatmap(pd.crosstab(index=df['weather'], columns=df['count']))
```

[218]: <AxesSubplot:xlabel='count', ylabel='weather'>



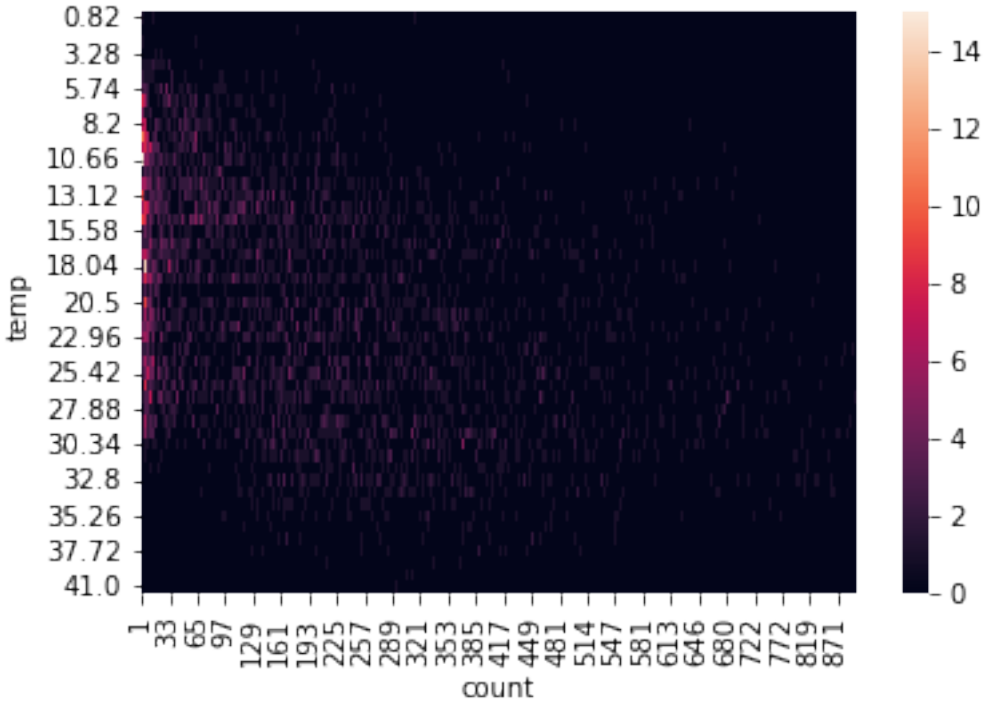
[219]:
```
sns.heatmap(pd.crosstab(index=df['weather'], columns=df['season']))
```

[219]: <AxesSubplot:xlabel='season', ylabel='weather'>



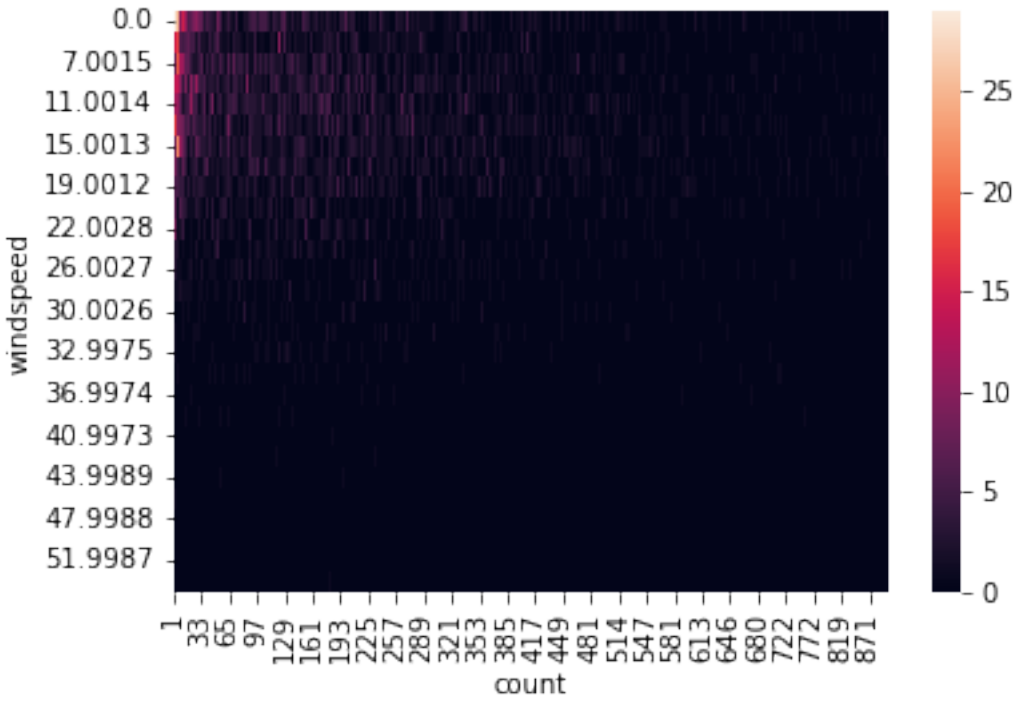
[234]:
```
sns.heatmap(pd.crosstab(index=df['temp'], columns=df['count']))
```

[234]: <AxesSubplot:xlabel='count', ylabel='temp'>



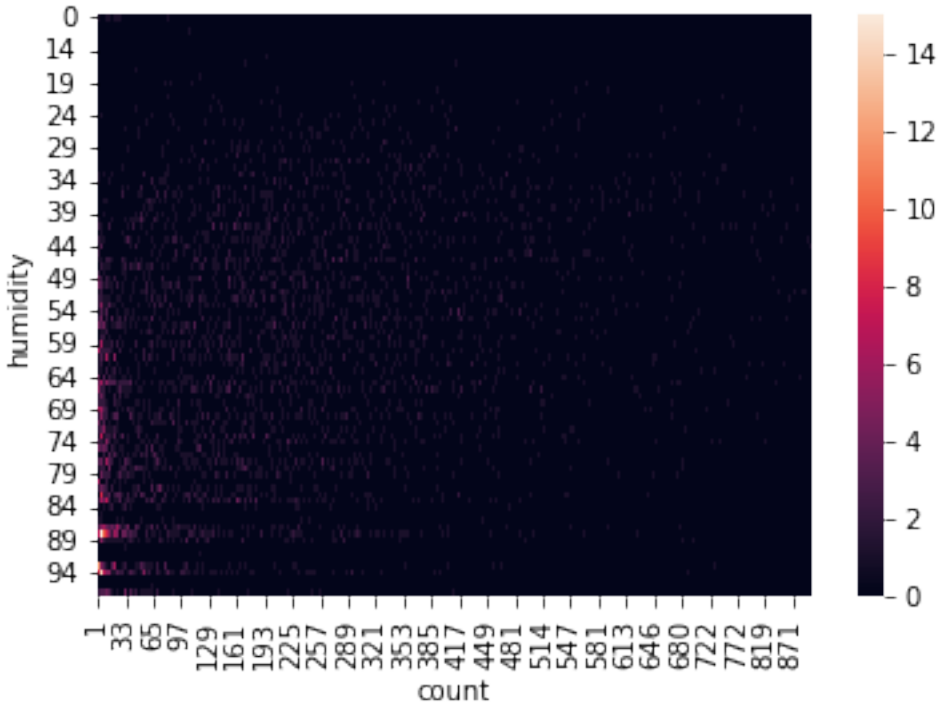
[235]:
```
sns.heatmap(pd.crosstab(index=df['windspeed'], columns=df['count']))
```

[235]: <AxesSubplot:xlabel='count', ylabel='windspeed'>



[236]:
```
sns.heatmap(pd.crosstab(index=df['humidity'], columns=df['count']))
```

[236]: <AxesSubplot:xlabel='count', ylabel='humidity'>



1. The data set contains equal no. of days of 4 seasons.
2. Weather 1 & 2 (with little or no rain) are favourable for cycle rent.
3. Season 2 & 3 (summer and fall) are favourable for cycle rent.
4. People prefer to rent cycle with temperature range 13-27 Celsius.
5. Rent count is more when windspeed is less than 19.
6. For humdity less than 20, count of rent is very very low.

0.2 2.Hypothesis Testing

0.2.1 2.1.Sample T-Test to check if Working Day has an effect on the number of electric cycles rented

Null Hypothesis (Ho) : Working Day has no effect on the number of electric cycles rented

Alternate Hypothesis (Ha) : Working Day has an effect on the number of electric cycles rented

As we need to compare two data set & it is catagorial-continuous, ttest_ind is prefered

[220]:
```
Arr_1=df[df['workingday']==0]['count'].to_numpy()
Arr_2=df[df['workingday']==1]['count'].to_numpy()
```

[221]:
```
print(np.var(Arr_1))
print(np.var(Arr_2))
print((np.var(Arr_2))/(np.var(Arr_1)))
```

```
30171.346098942427
```

```
34040.69710674686
1.1282458858519429
```

As ratio of variance is less than 4, we can proceed with T_test

[222]:
```
# t_test_ind
# alpha = 0.05
test_stat,p_value = ttest_ind(Arr_1,Arr_2)
print(p_value)
if p_value<0.05:
    print("Working Day has an effect on the number of electric cycles rented")
else:
    print("Working Day has no effect on the number of electric cycles rented")
```

```
0.22644804226361348
Working Day has no effect on the number of electric cycles rented
```

0.2.2 2.2.1.ANNOVA to check if No. of cycles rented is similar or different in different weather

Null Hypothesis (Ho) : weather has no effect on the number of electric cycles rented

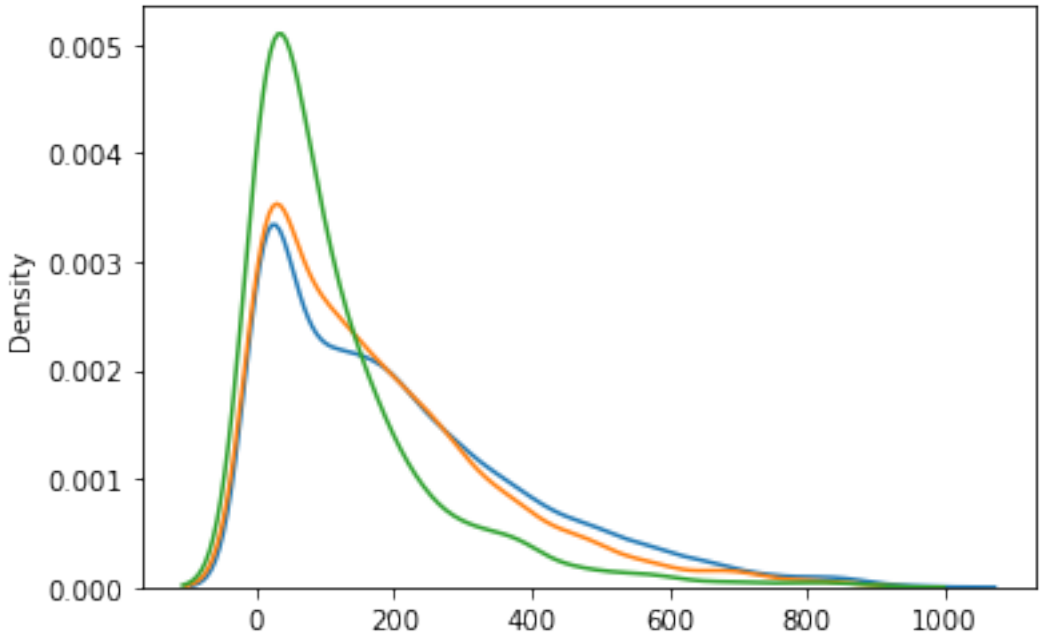
Alternate Hypothesis (Ha) : weather has an effect on the number of electric cycles rented

As we need to compare 4 data set & it is catagorial-continuous, ANNOVA is prefered

[224]:
```
ar_1=df[df['weather']==1]['count'].to_numpy()
ar_2=df[df['weather']==2]['count'].to_numpy()
ar_3=df[df['weather']==3]['count'].to_numpy()
ar_4=df[df['weather']==4]['count'].to_numpy()
```

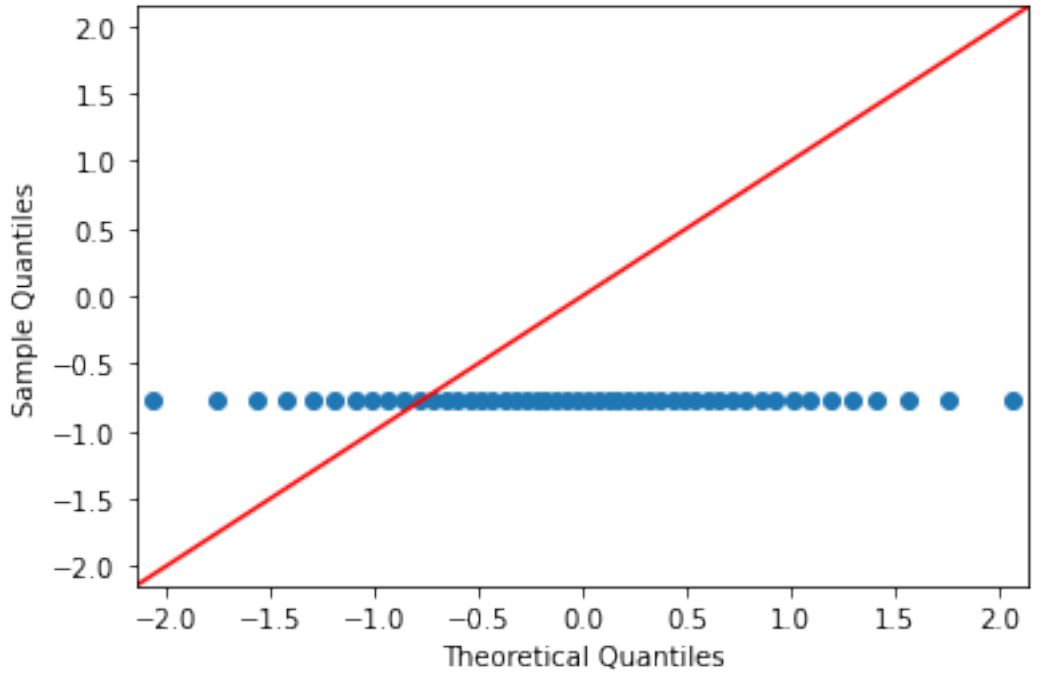
[225]:
```
sns.kdeplot(ar_1)
sns.kdeplot(ar_2)
sns.kdeplot(ar_3)
sns.kdeplot(ar_4)
plt.show()
```

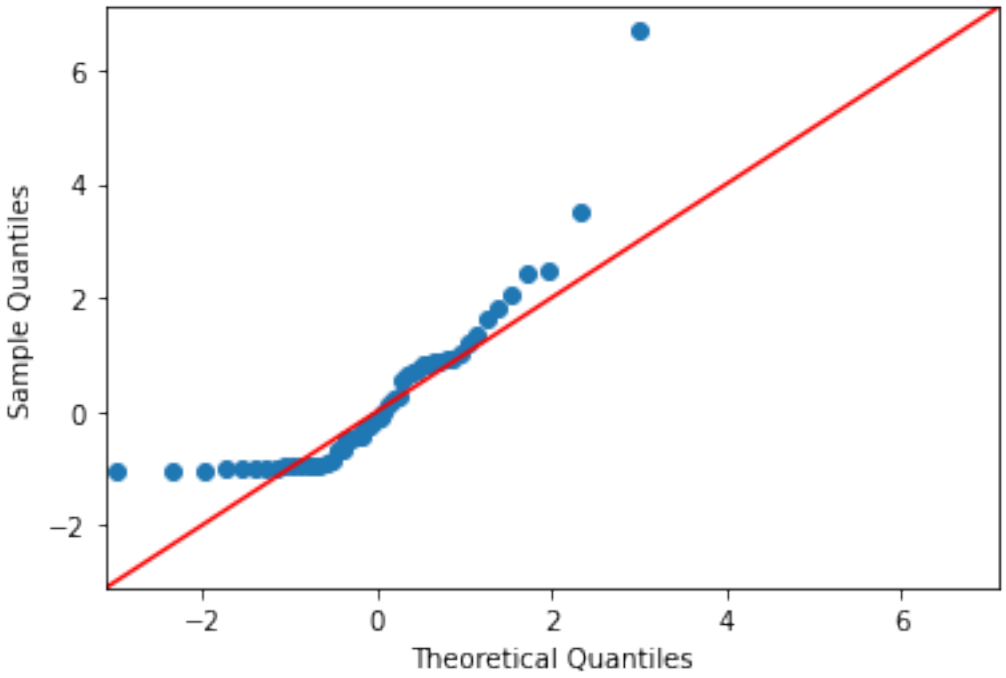
```
C:\Users\rahul.kumar\Anaconda3\lib\site-packages\seaborn\distributions.py:316:
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
  warnings.warn(msg, UserWarning)
```

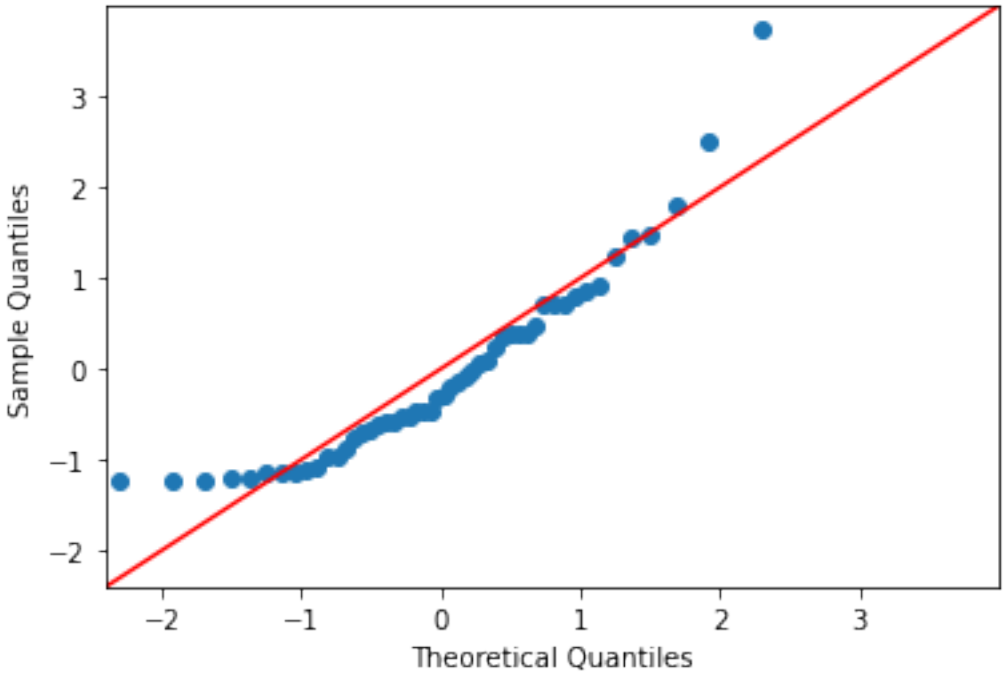


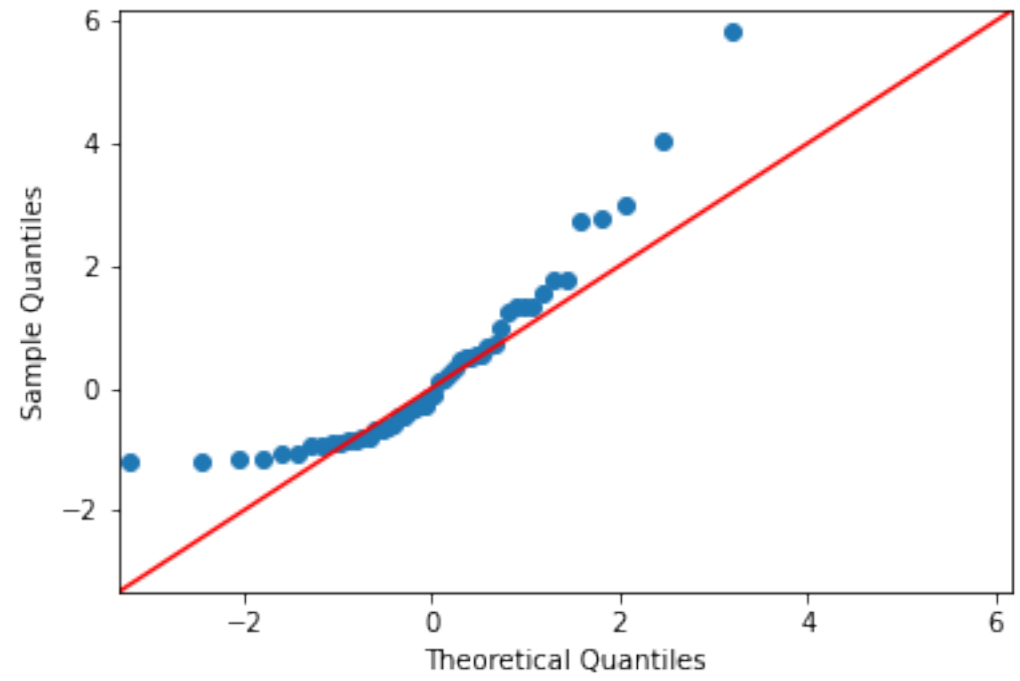
[226]:
```
sm.qqplot(np.random.choice(ar_1, size=50), stats.t, fit=True, line="45")
sm.qqplot(np.random.choice(ar_2, size=50), stats.t, fit=True, line="45")
sm.qqplot(np.random.choice(ar_3, size=50), stats.t, fit=True, line="45")
sm.qqplot(np.random.choice(ar_4, size=50), stats.t, fit=True, line="45")
```

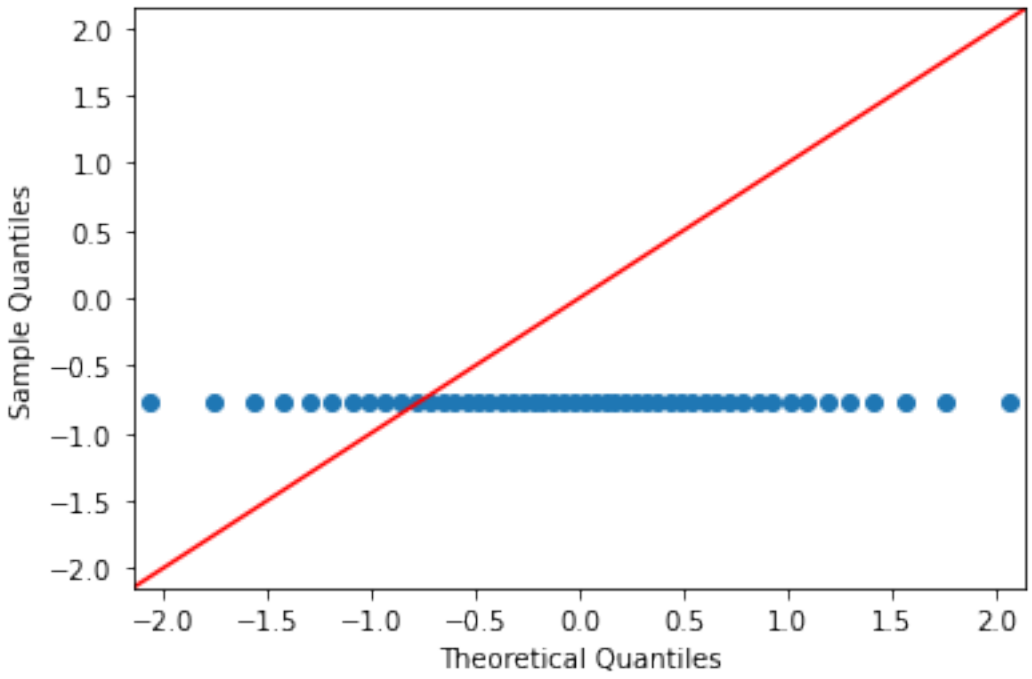
[226]:




Sample Quantiles
Theoretical Quantiles
6
4
2
0
−2
−2
0
2
4
6


Sample Quantiles
Theoretical Quantiles
3
2
1
0
−1
−2
−2
−1
0
1
2
3





As the datasets are not close to gaussian, we are going for kruskal test

[227]:
```
# kruskal
# alpha = 0.05
test_stat,p_value = kruskal(ar_1,ar_2,ar_3,ar_4)
print(p_value)
if p_value<0.05:
    print("weather has an effect on the number of electric cycles rented")
else:
    print("weather has no effect on the number of electric cycles rented")
```

```
3.501611300708679e-44
weather has an effect on the number of electric cycles rented
```

0.2.3 2.2.2.ANNOVA to check if No. of cycles rented is similar or different in different season

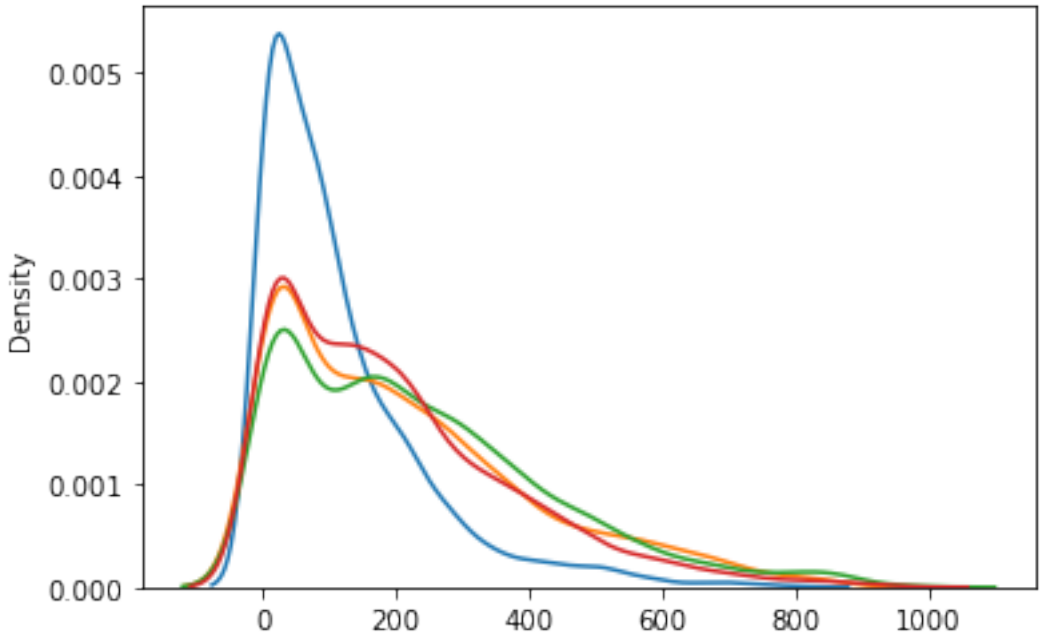
Null Hypothesis (Ho) : season has no effect on the number of electric cycles rented

Alternate Hypothesis (Ha) : season has an effect on the number of electric cycles rented

As we need to compare 4 data set & it is catagorial-continuous, ANNOVA is prefered

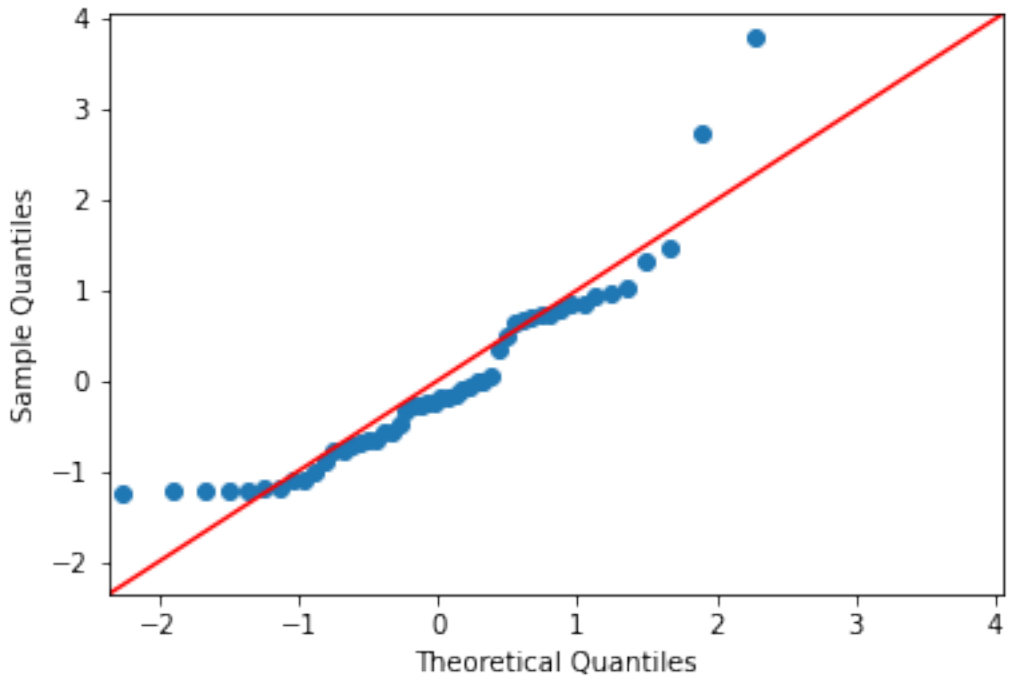
[228]:
```
a_1=df[df['season']==1]['count'].to_numpy()
a_2=df[df['season']==2]['count'].to_numpy()
a_3=df[df['season']==3]['count'].to_numpy()
a_4=df[df['season']==4]['count'].to_numpy()
```

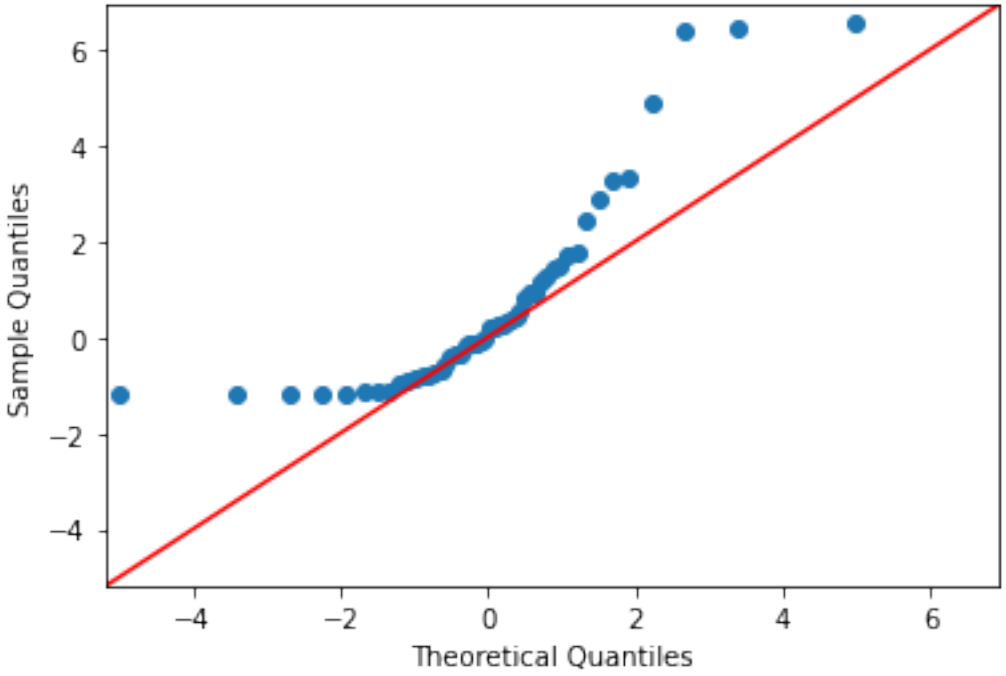
[229]:
```
sns.kdeplot(a_1)
sns.kdeplot(a_2)
sns.kdeplot(a_3)
sns.kdeplot(a_4)
plt.show()
```

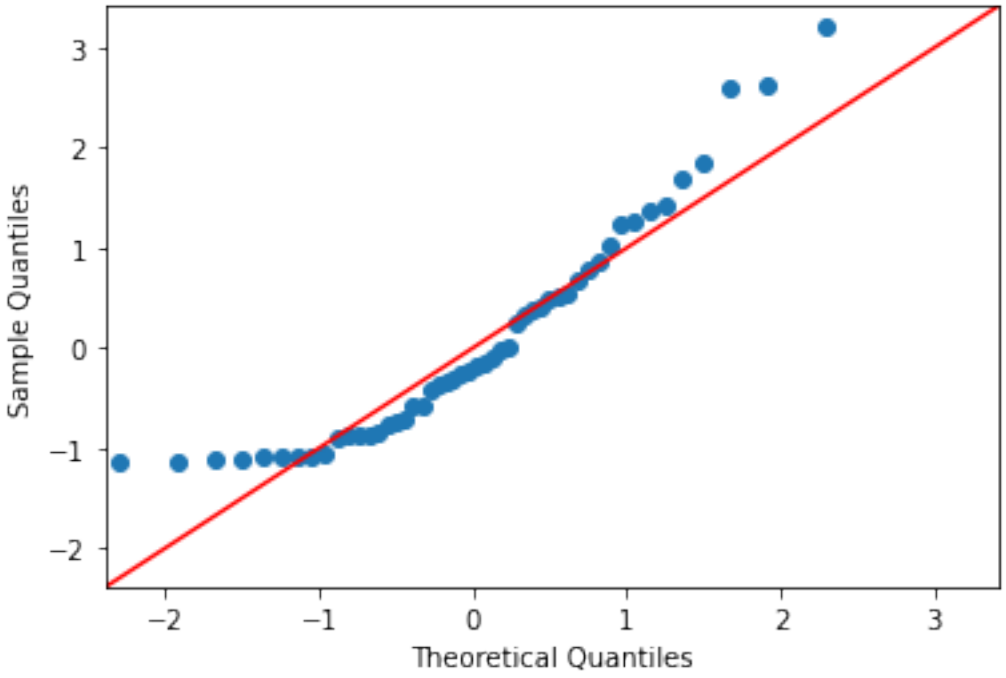


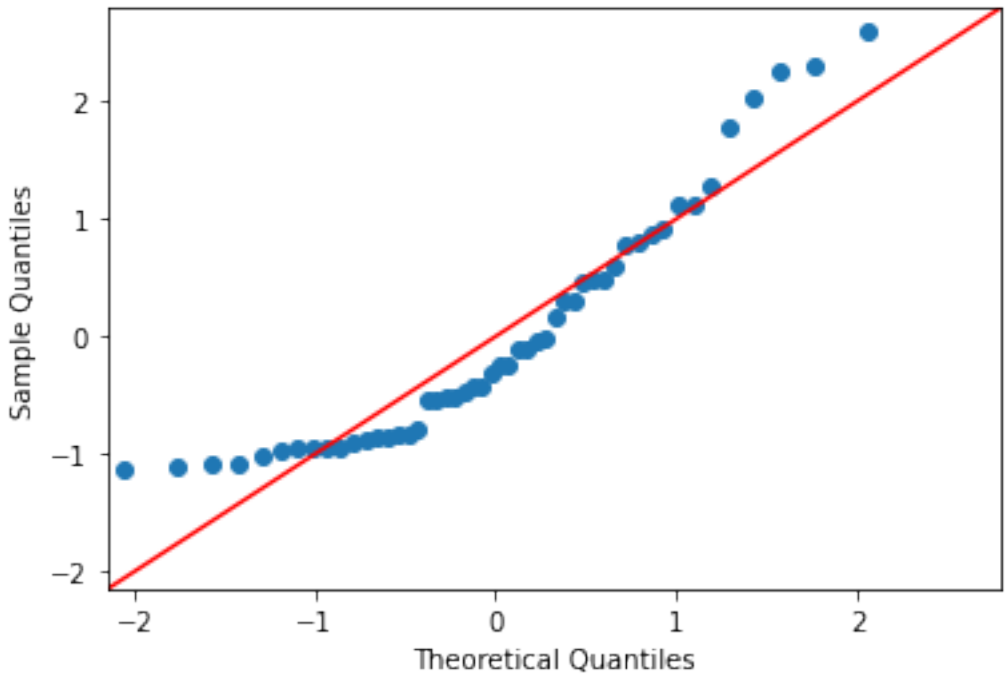
[230]:
```
sm.qqplot(np.random.choice(a_1, size=50), stats.t, fit=True, line="45")
sm.qqplot(np.random.choice(a_2, size=50), stats.t, fit=True, line="45")
sm.qqplot(np.random.choice(a_3, size=50), stats.t, fit=True, line="45")
sm.qqplot(np.random.choice(a_4, size=50), stats.t, fit=True, line="45")
```

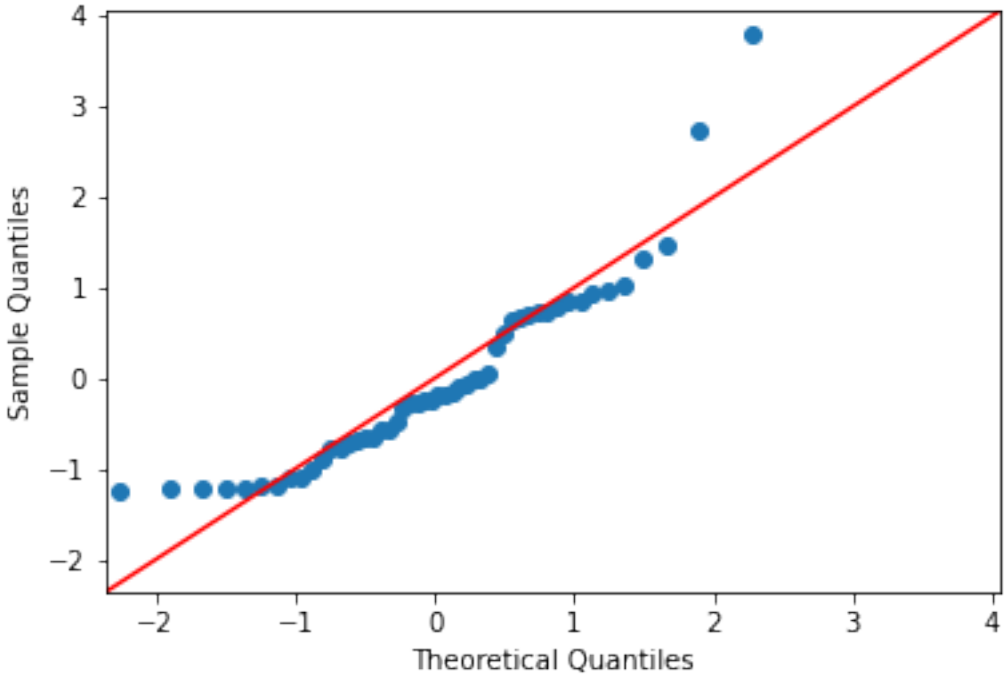
[230]:




Sample Quantiles
Theoretical Quantiles
6
4
2
0
−2
−4
−4
−2
0
2
4
6


Sample Quantiles
Theoretical Quantiles
3
2
1
0
−1
−2
−2
−1
0
1
2
3





As the datasets are not close to gaussian, we are going for kruskal test

[231]:
```
# kruskal
# alpha = 0.05
test_stat,p_value = kruskal(a_1,a_2,a_3,a_4)
print(p_value)
if p_value<0.05:
    print("season has an effect on the number of electric cycles rented")
else:
    print("season has no effect on the number of electric cycles rented")
```

```
2.479008372608633e-151
season has an effect on the number of electric cycles rented
```

0.2.4 2.3.Chi-square test to check if Weather is dependent on the season

Null Hypothesis (Ho) : Weather is independent on the season

Alternate Hypothesis (Ha) : Weather is dependent on the season

As we need to compare 2 data set & it is catagorial-catagorial, Chi-square is prefered

[232]:
```
data_table = pd.crosstab(df['season'], df['weather'])
print("Observed values:")
data_table
```

```
Observed values:
```

[232]:

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

[233]:
```
# Chi-Sqaure test
# alpha = 0.05
test_stat,p_value,dof,expecrted = chi2_contingency(data_table)
print(p_value)
if p_value<0.05:
    print("Weather is dependent on the season")
else:
    print("Weather is independent on the season")
```

```
1.549925073686492e-07
Weather is dependent on the season
```