In [151…
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import ttest_ind
import warnings
warnings.filterwarnings('ignore')
```

In [152…
```
df=pd.read_csv("bike_sharing.txt")
df.head()
```

Out[152]:

	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 [153…
```
df.shape
```

Out[153]:
```
(10886, 12)
```

In [154…
```
df.ndim
```

Out[154]:
```
2
```

In [155…
```
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 [156…

```
df.describe()
```

Out[156]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	1088
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	19
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	18
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	4
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	14
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	28
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	97

In [157…

```
# NULL
```

In [158…

```
df.isna().sum()
```

Out[158]:
```
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 is no null in the data set.

In [159…
```
df.duplicated().value_counts()
```

Out[159]:
```
False    10886
Name: count, dtype: int64
```

- There is no duplicate values

In [160…
```
df.head()
```

Out[160]:

	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 [161…
```
for col in df.columns:
    print("Value counts for column", col)
    print(df[col].value_counts().head())
    print()
```

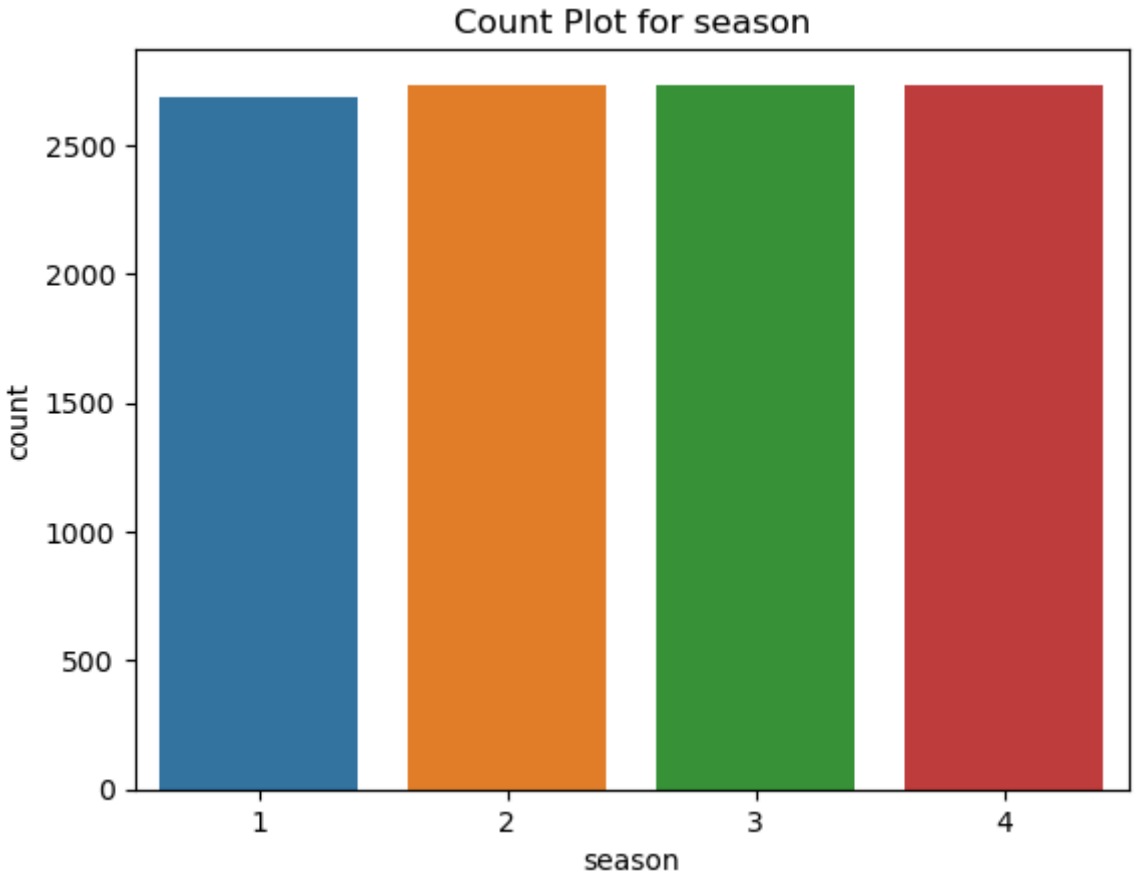
```
Value counts for column datetime
datetime
2011-01-01 00:00:00    1
2012-05-01 21:00:00    1
2012-05-01 13:00:00    1
2012-05-01 14:00:00    1
2012-05-01 15:00:00    1
Name: count, dtype: int64

Value counts for column season
season
4    2734
2    2733
3    2733
1    2686
Name: count, dtype: int64

Value counts for column holiday
holiday
0    10575
1      311
Name: count, dtype: int64

Value counts for column workingday
workingday
1    7412
0    3474
Name: count, dtype: int64

Value counts for column weather
weather
1    7192
2    2834
3     859
4       1
Name: count, dtype: int64

Value counts for column temp
temp
14.76    467
26.24    453
28.70    427
13.94    413
18.86    406
```

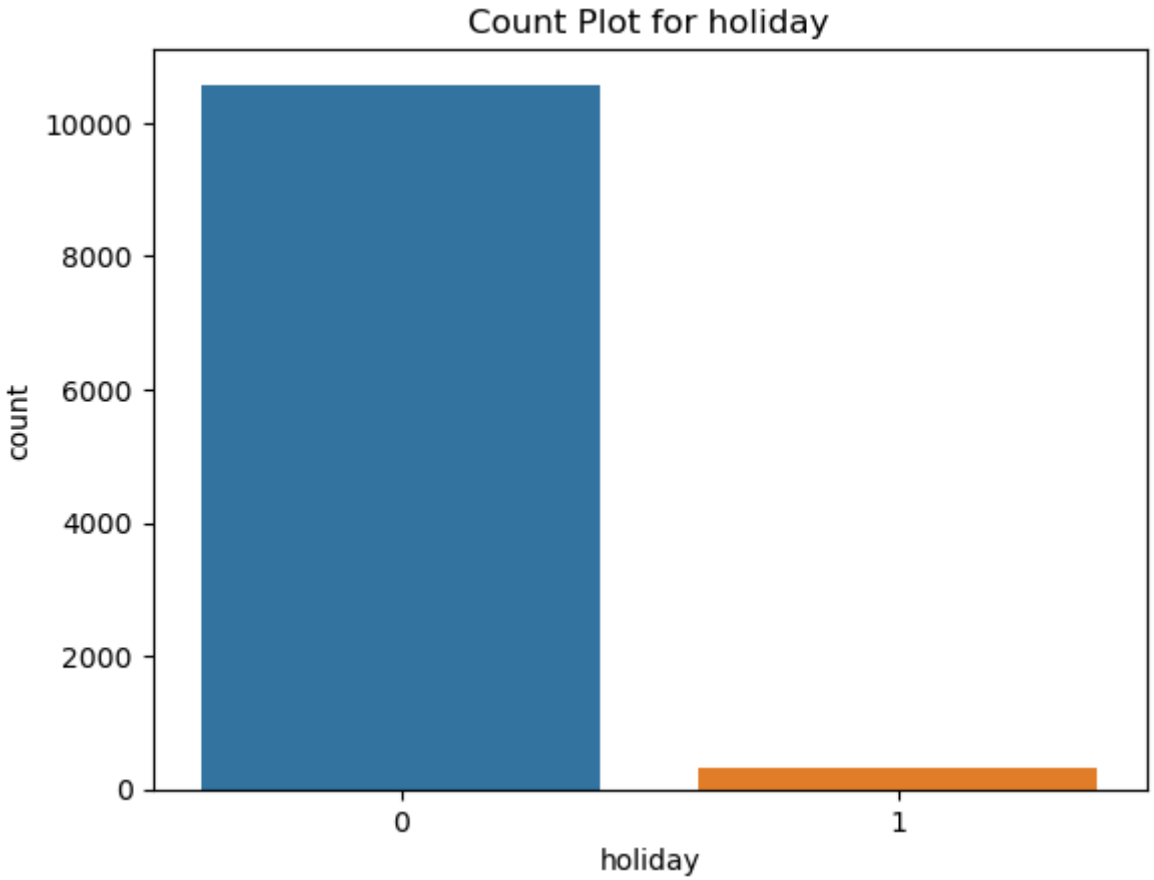
```
Name: count, dtype: int64

Value counts for column atemp
atemp
31.060    671
25.760    423
22.725    406
20.455    400
26.515    395
Name: count, dtype: int64

Value counts for column humidity
humidity
88    368
94    324
83    316
87    289
70    259
Name: count, dtype: int64

Value counts for column windspeed
windspeed
0.0000     1313
8.9981     1120
11.0014    1057
12.9980    1042
7.0015     1034
Name: count, dtype: int64

Value counts for column casual
casual
0    986
1    667
2    487
3    438
4    354
Name: count, dtype: int64

Value counts for column registered
registered
3    195
4    190
5    177
6    155
```

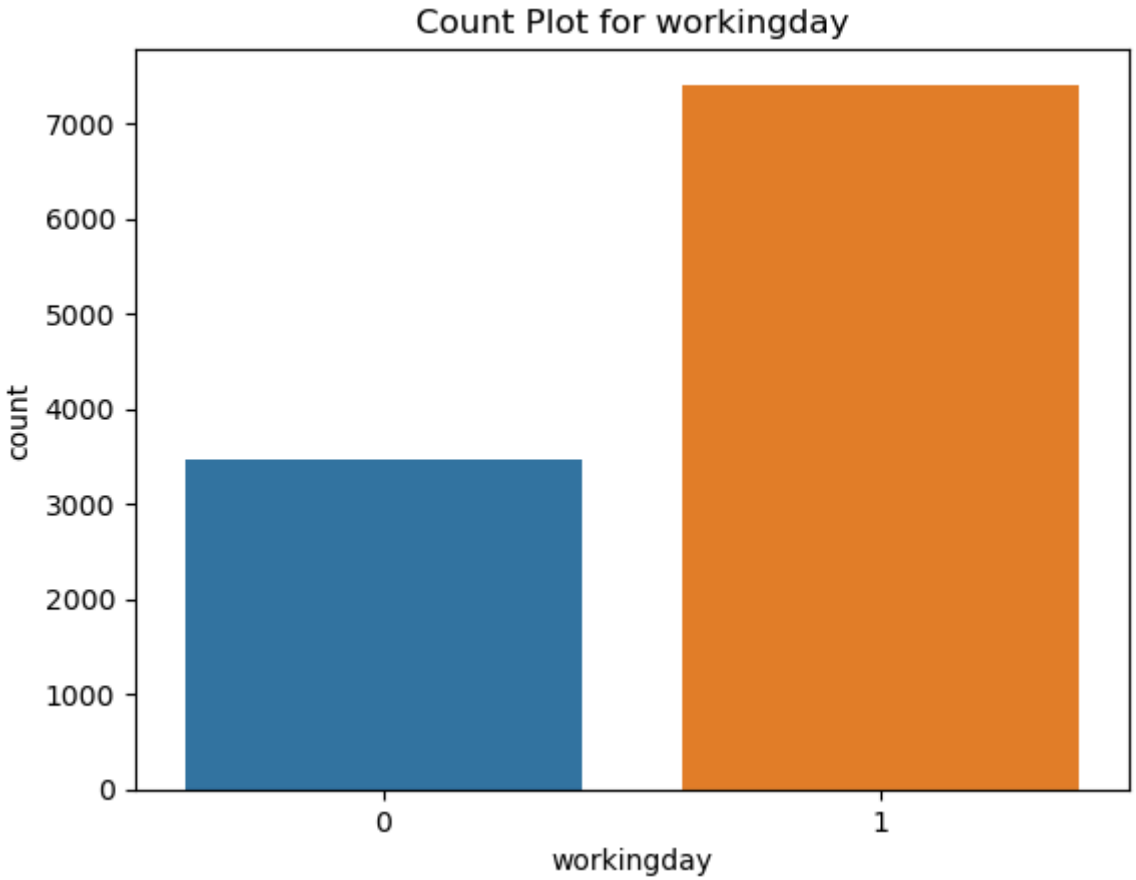
```
2    150
Name: count, dtype: int64

Value counts for column count
count
5    169
4    149
3    144
6    135
2    132
Name: count, dtype: int64
```

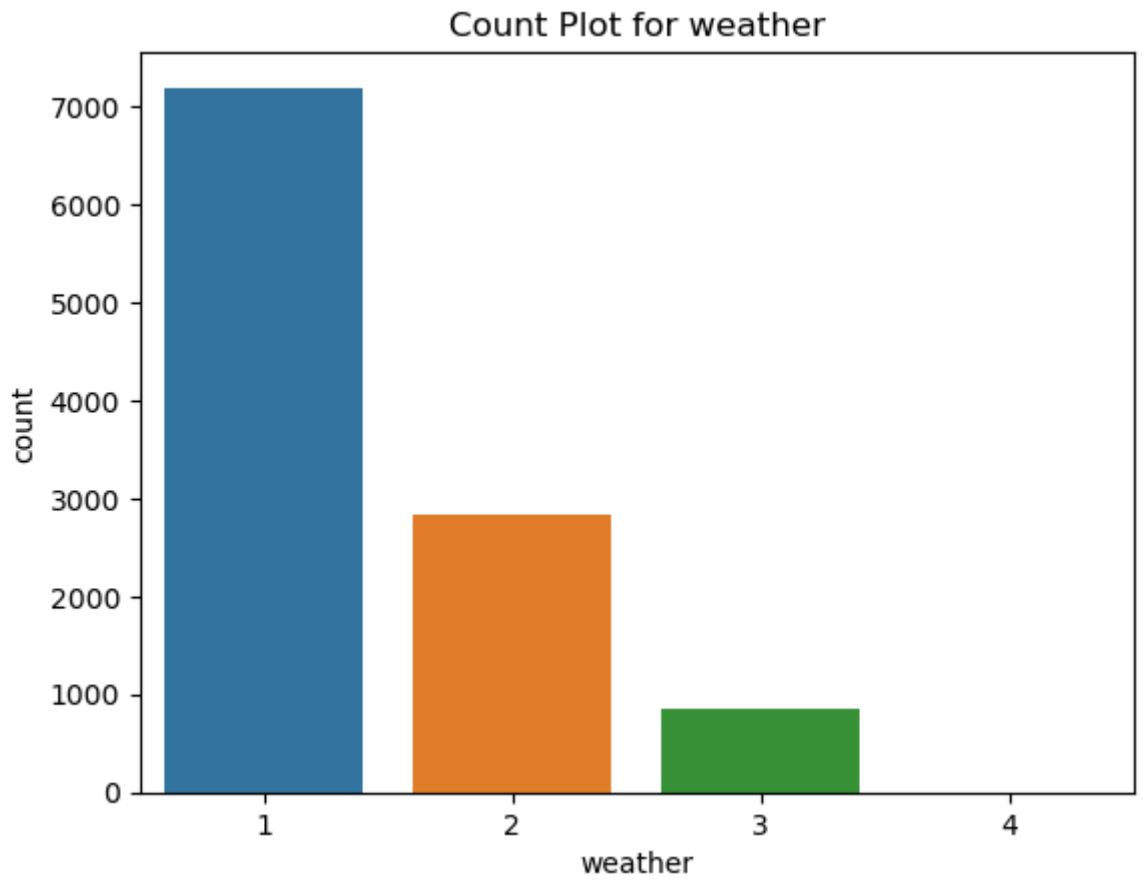
In [162…

```python
# for categorical
col=['season','holiday','workingday','weather']
for i in col:
    plt.figure()
    sns.countplot(x=i, data=df)
    plt.title("Count Plot for " + i)
    plt.show()
```


Count Plot for season
count
2500
2000
1500
1000
500
0
1
2
3
4
season


Count Plot for holiday
10000
8000
6000
4000
2000
0
count
0
1
holiday

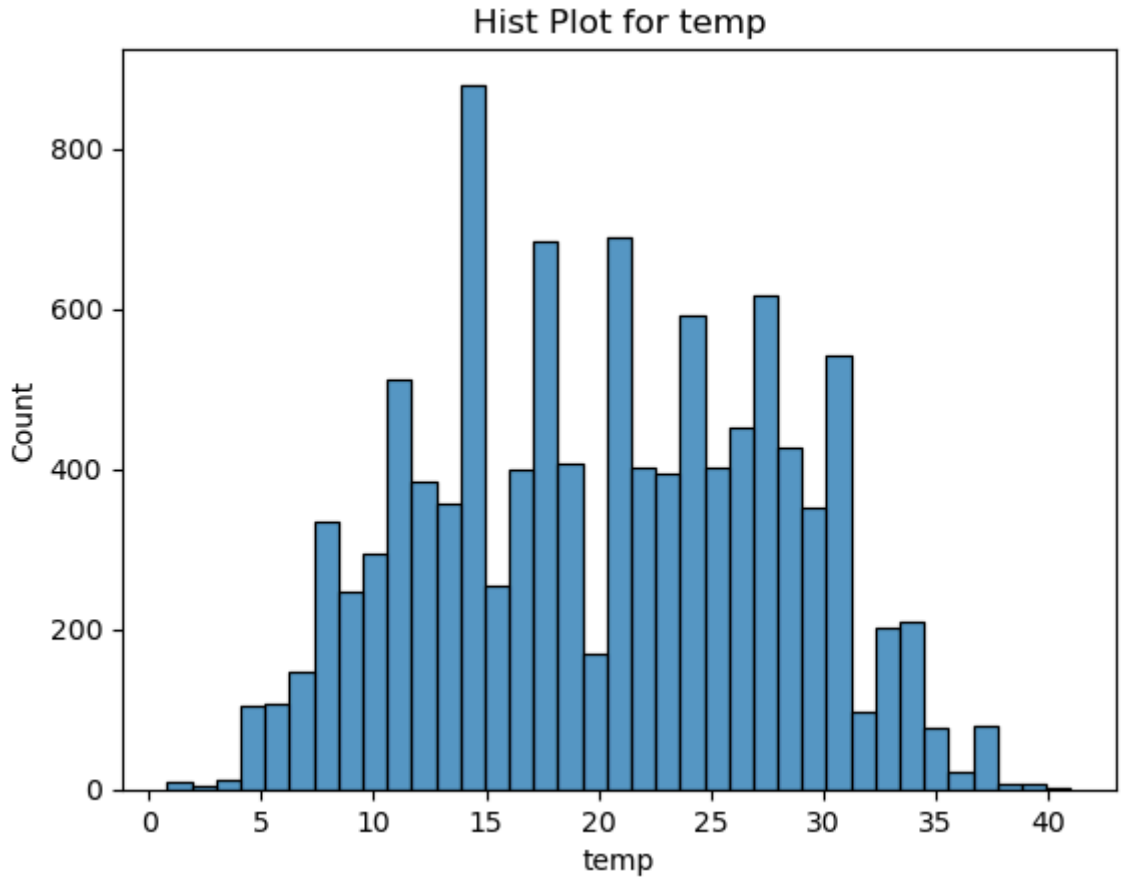

Count Plot for workingday
7000
6000
5000
4000
3000
2000
1000
0
count
0
1
workingday

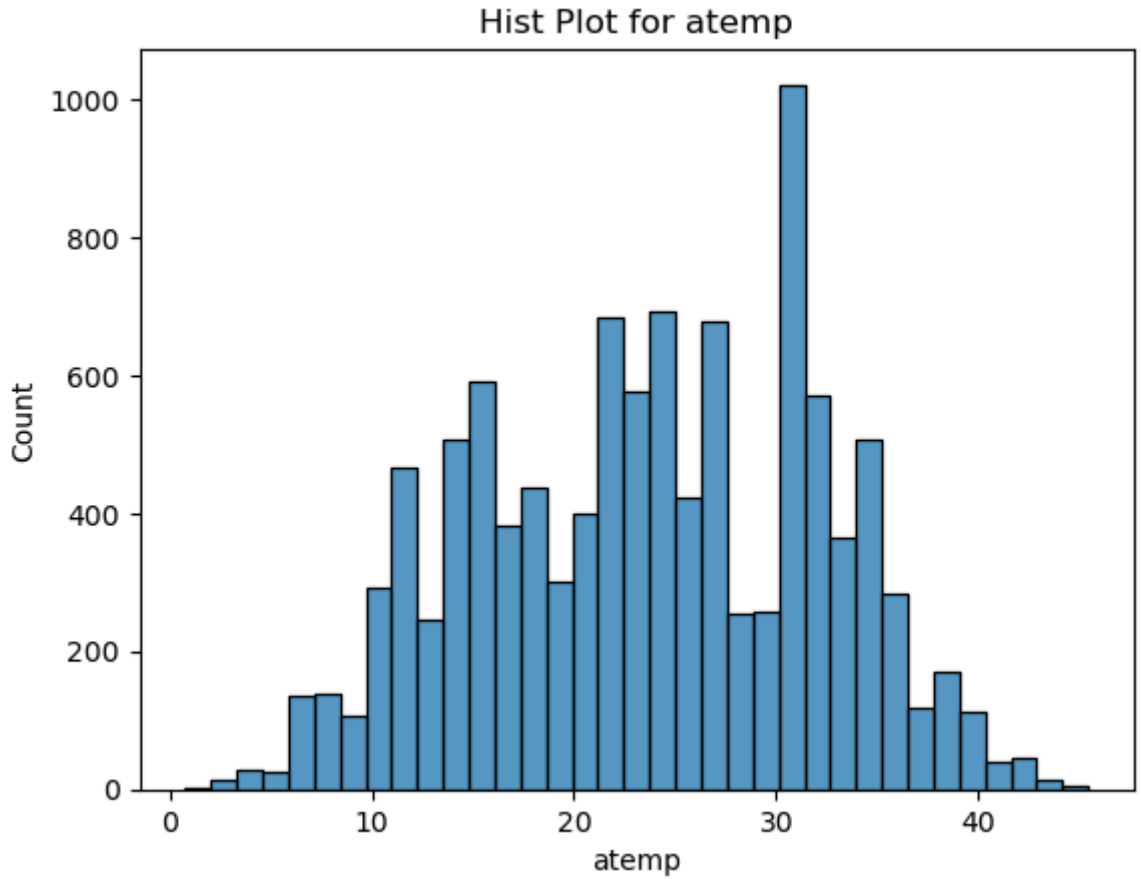


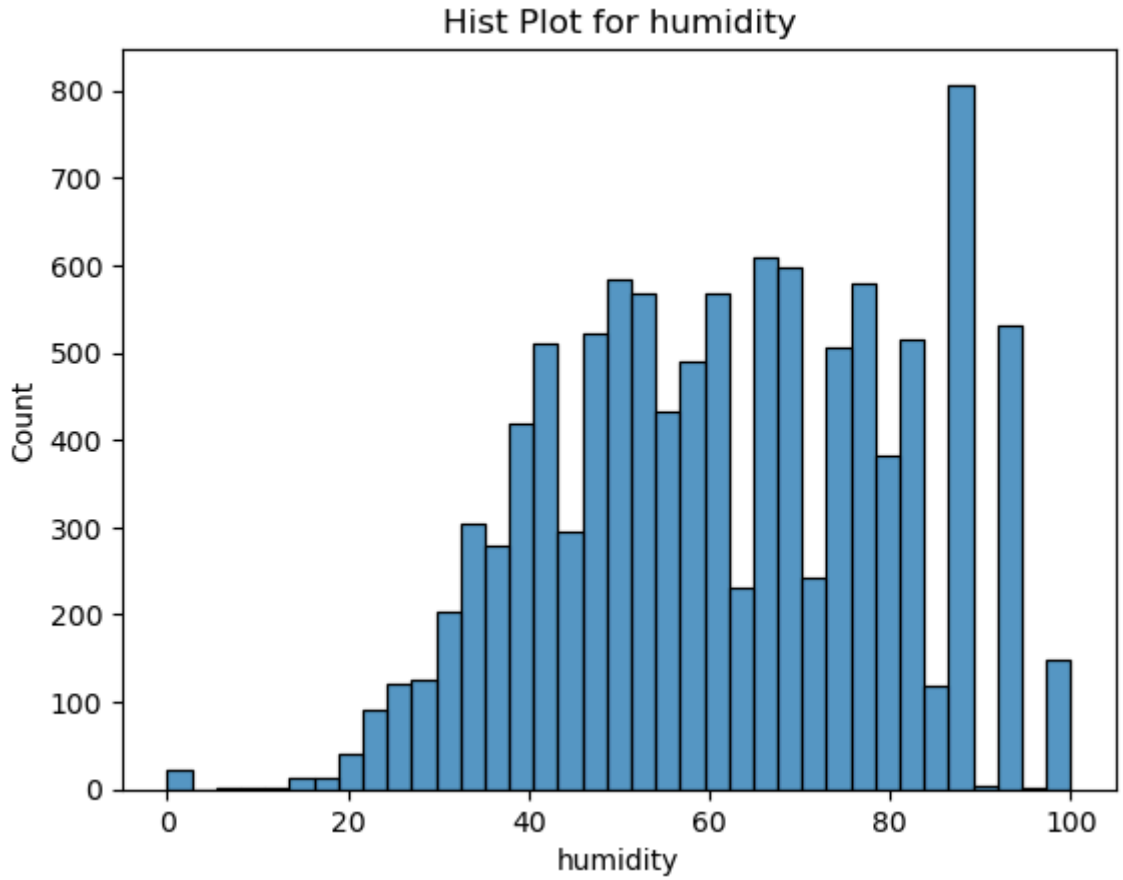
In [163…
```
# Numerical
```

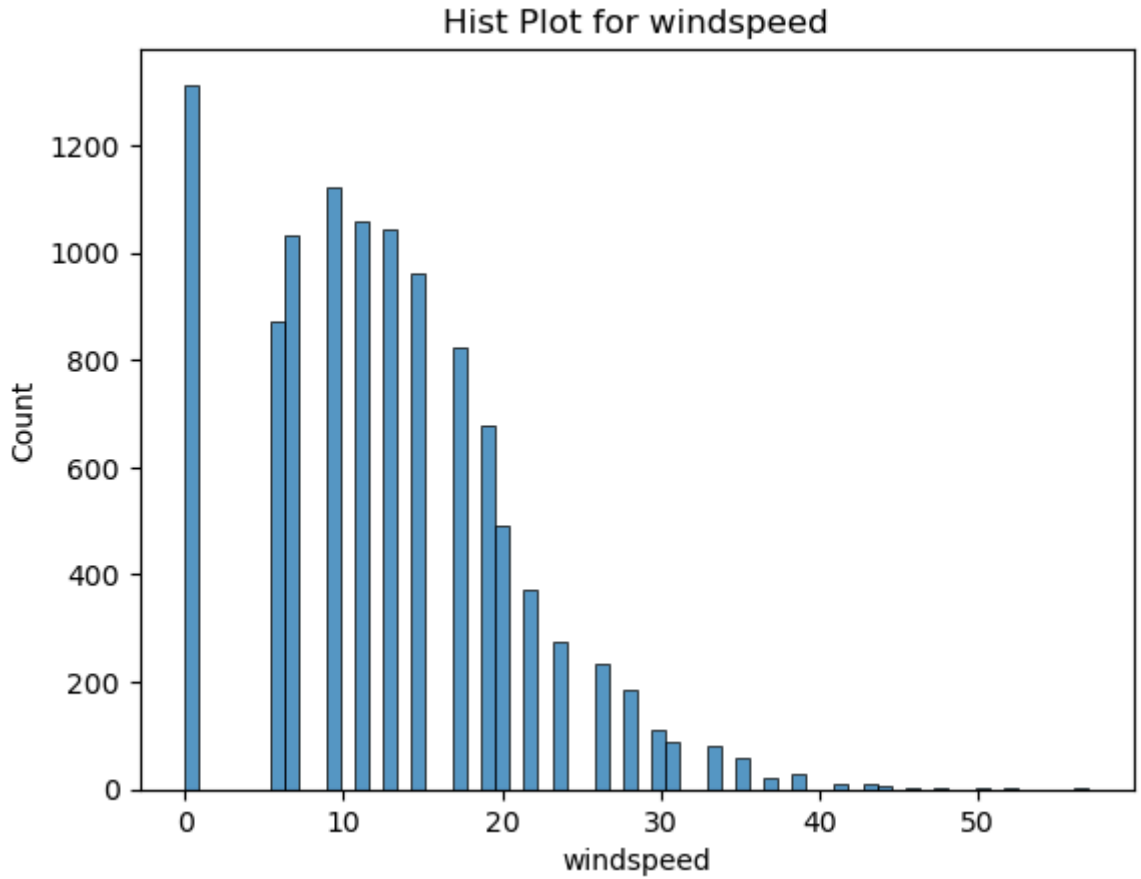
In [164…
```
num_col=['temp','atemp','humidity','windspeed','casual','registered','count']
```

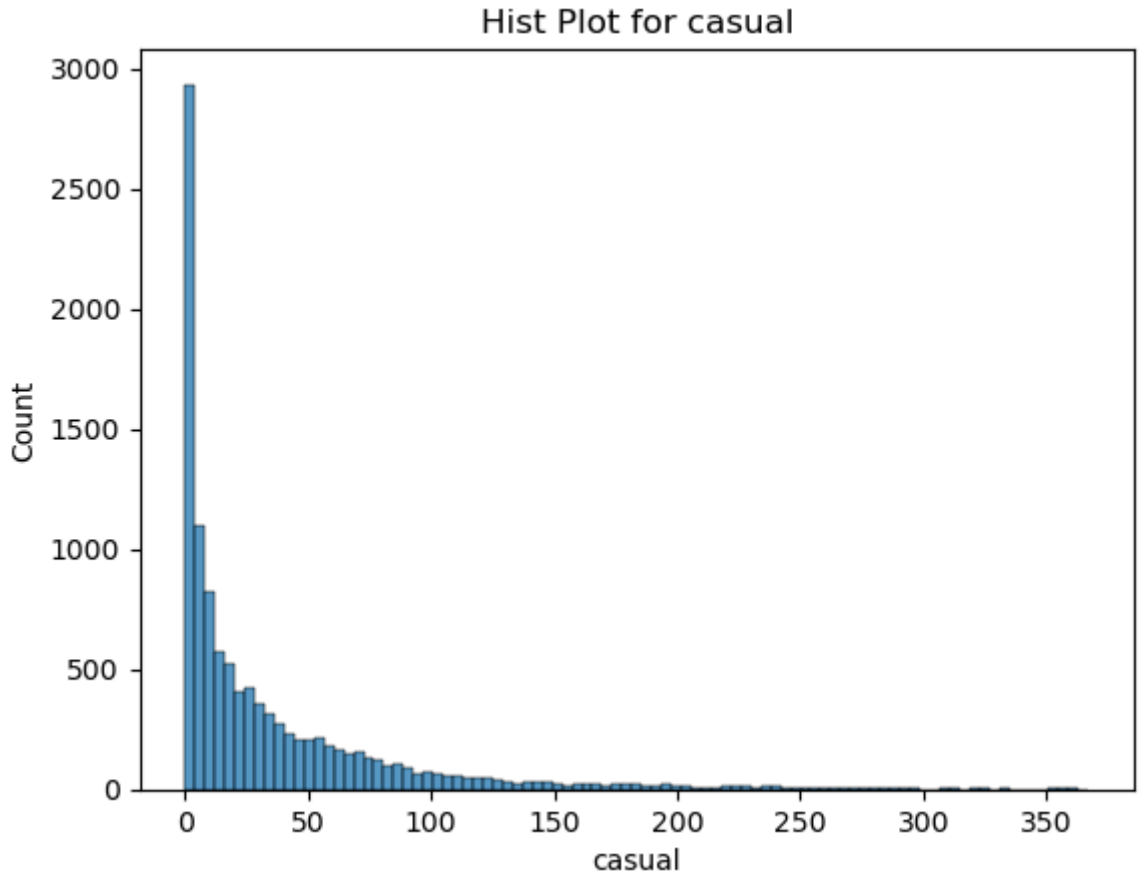
In [165…
```
# for Numerical
col=['season','holiday','workingday','weather']
for i in num_col:
    plt.figure()
    sns.histplot(x=i, data=df)
    plt.title("Hist Plot for " + i)
    plt.show()
```

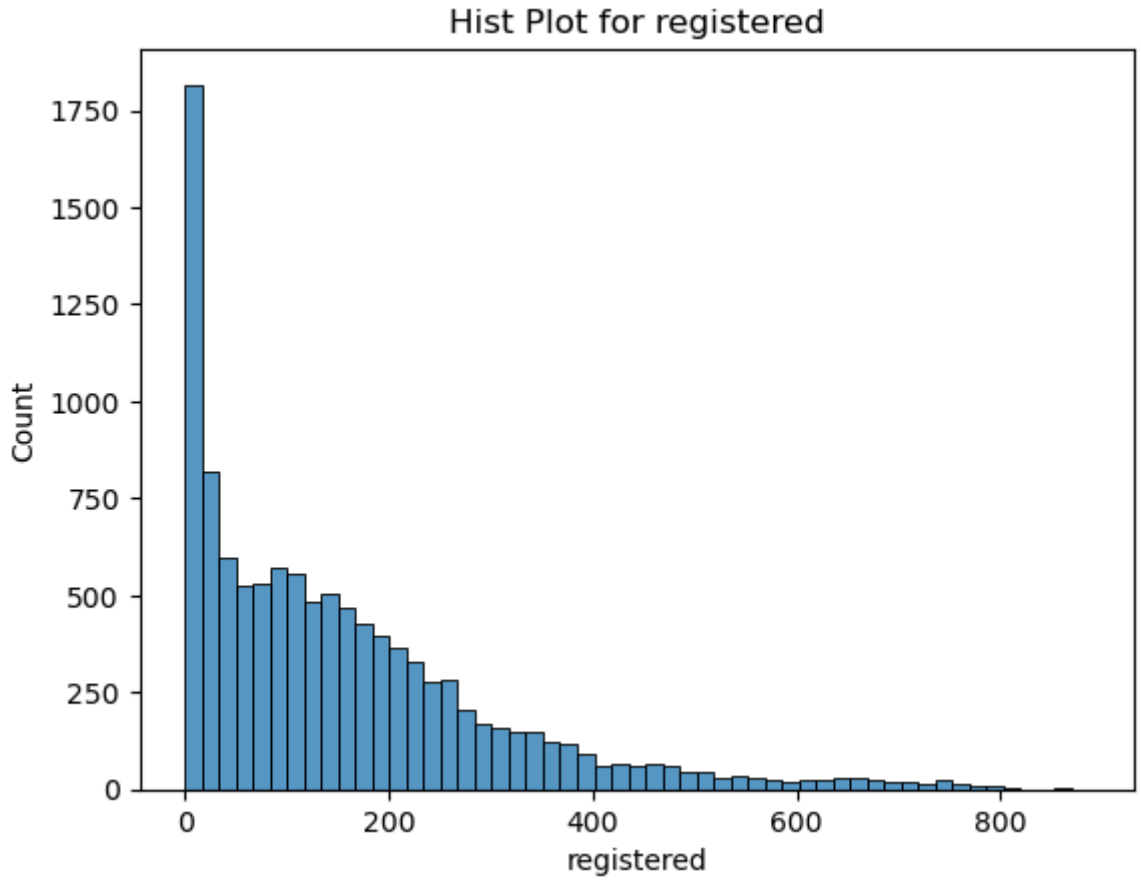

Hist Plot for temp
Count
0
200
400
600
800
0
5
10
15
20
25
30
35
40
temp

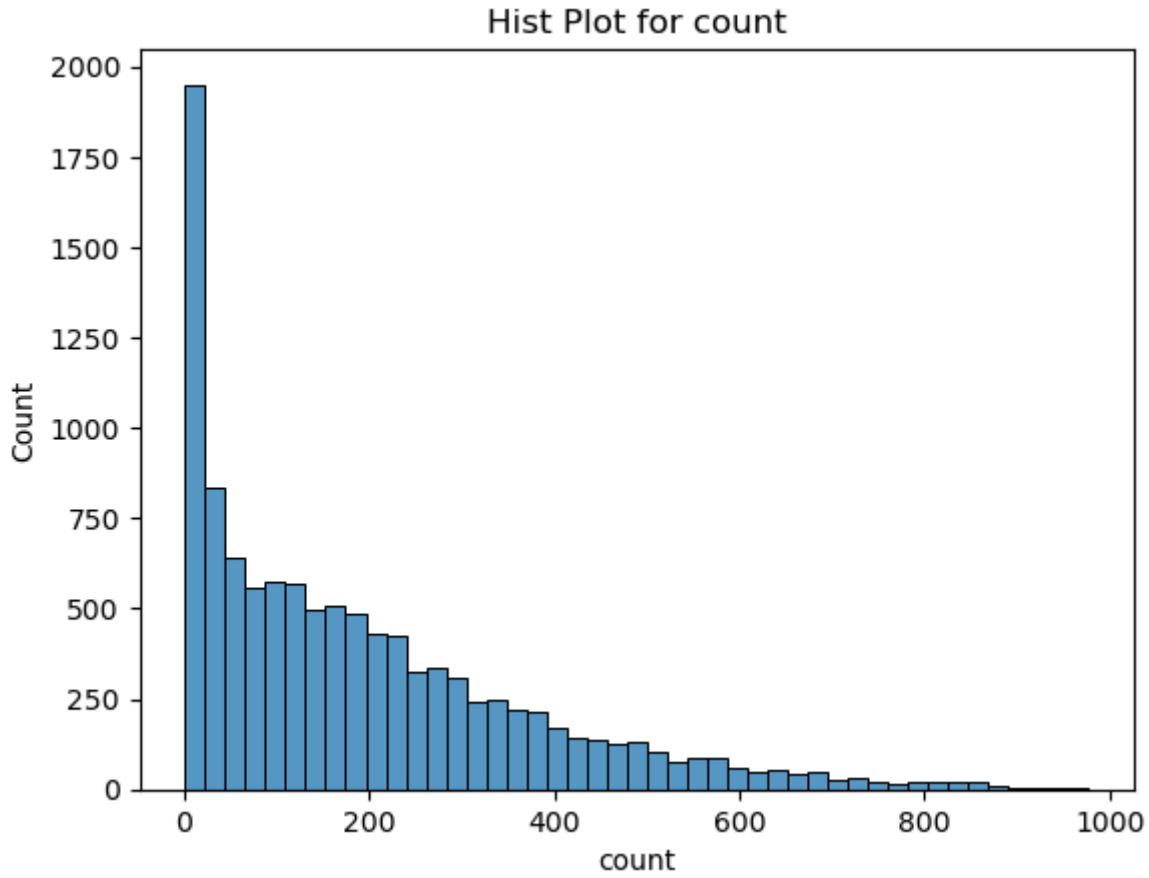

Hist Plot for atemp
Count
1000
800
600
400
200
0
0
10
20
30
40
atemp


Hist Plot for humidity
Count
humidity
0
100
200
300
400
500
600
700
800
0
20
40
60
80
100


Hist Plot for windspeed
Count
0
200
400
600
800
1000
1200
0
10
20
30
40
50
windspeed


Hist Plot for casual
Count
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250
300
350
casual


Hist Plot for registered
Count
1750
1500
1250
1000
750
500
250
0
0
200
400
600
800
registered

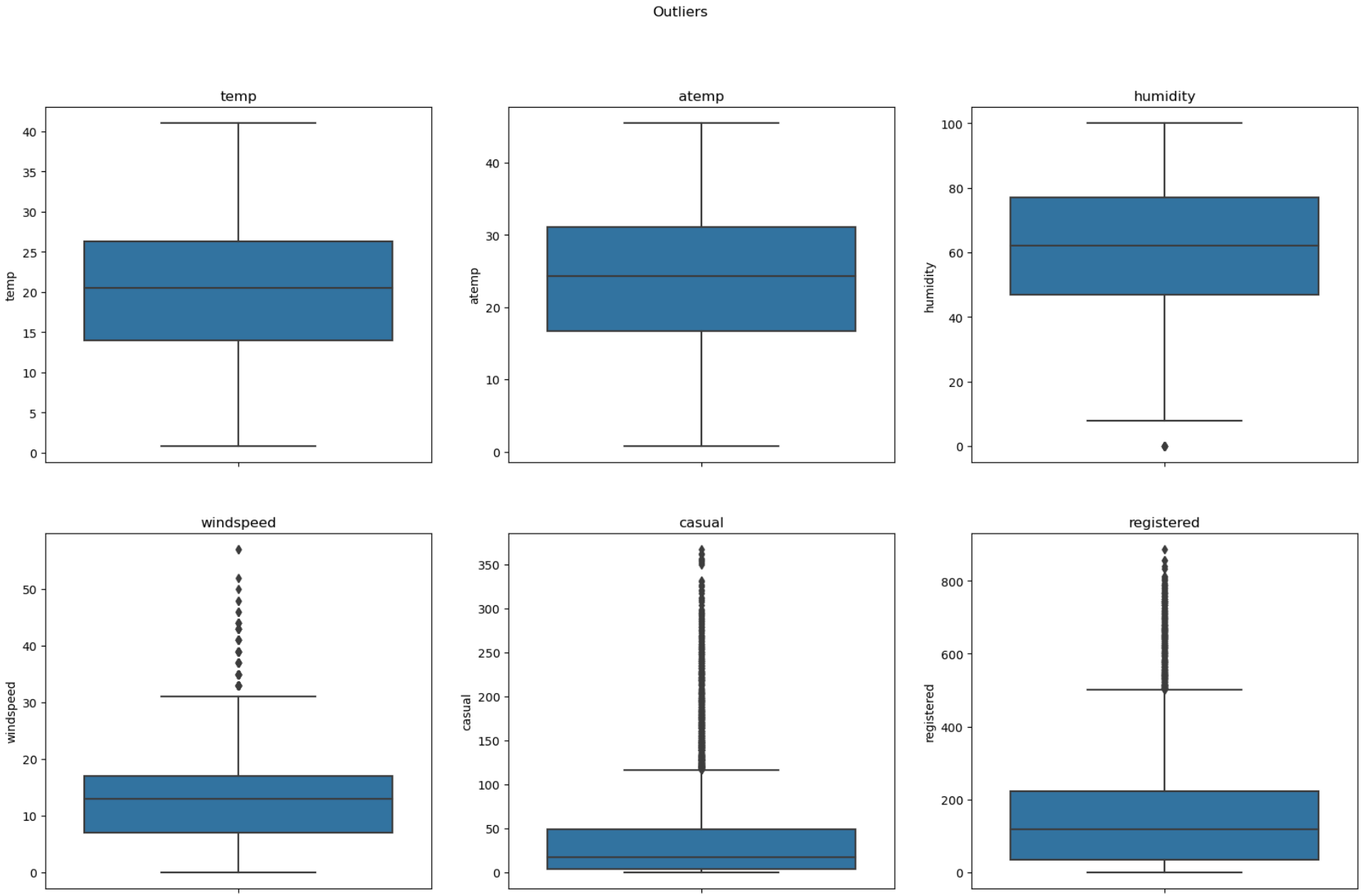


outlier

In [166…

```
col=['temp','atemp','humidity','windspeed','casual','registered']
fig, axes = plt.subplots(2, 3, figsize=(20, 12))

for i in range(2):
    for j in range(3):
        variable = col[i * 3 + j]
        sns.boxplot(ax=axes[i, j], data=df, y=variable)
        axes[i, j].set_title(variable)
```

```
plt.suptitle("Outliers")
plt.show();
```



In [167…

```
# fro humidity
```

```
q1=df['humidity'].quantile(0.25)
q3=df['humidity'].quantile(0.75)
iqr=q3-q1
iqr
df[(df['humidity']<(q1-1.5*iqr)) | (df['humidity']>(q3+1.5*iqr))]['humidity'].reset_index().head()
```

Out[167]:

	index	humidity
0	1091	0
1	1092	0
2	1093	0
3	1094	0
4	1095	0

- These are the outier for humidity

In [168…

```
# for windspeed

q1=df['windspeed'].quantile(0.25)
q3=df['windspeed'].quantile(0.75)
iqr=q3-q1
iqr
df[(df['windspeed']<(q1-1.5*iqr)) | (df['windspeed']>(q3+1.5*iqr))]['windspeed'].reset_index().head()
```

Out[168]:

	index	windspeed
0	175	32.9975
1	178	36.9974
2	194	35.0008
3	196	35.0008
4	265	39.0007

- These are the outier for windspeed

In [169…

```
# fro casual

q1=df['casual'].quantile(0.25)
q3=df['casual'].quantile(0.75)
iqr=q3-q1
iqr
df[(df['casual']<(q1-1.5*iqr)) | (df['casual']>(q3+1.5*iqr))]['casual'].reset_index().head()
```

Out[169]:

	index	casual
0	1173	144
1	1174	149
2	1175	124
3	1311	126
4	1312	174

- These are the outier for casual

In [170…

```
# fro registered
q1=df['registered'].quantile(0.25)
q3=df['registered'].quantile(0.75)
iqr=q3-q1
iqr
df[(df['registered']<(q1-1.5*iqr)) | (df['registered']>(q3+1.5*iqr))]['registered'].reset_index().head()
```

Out[170]:

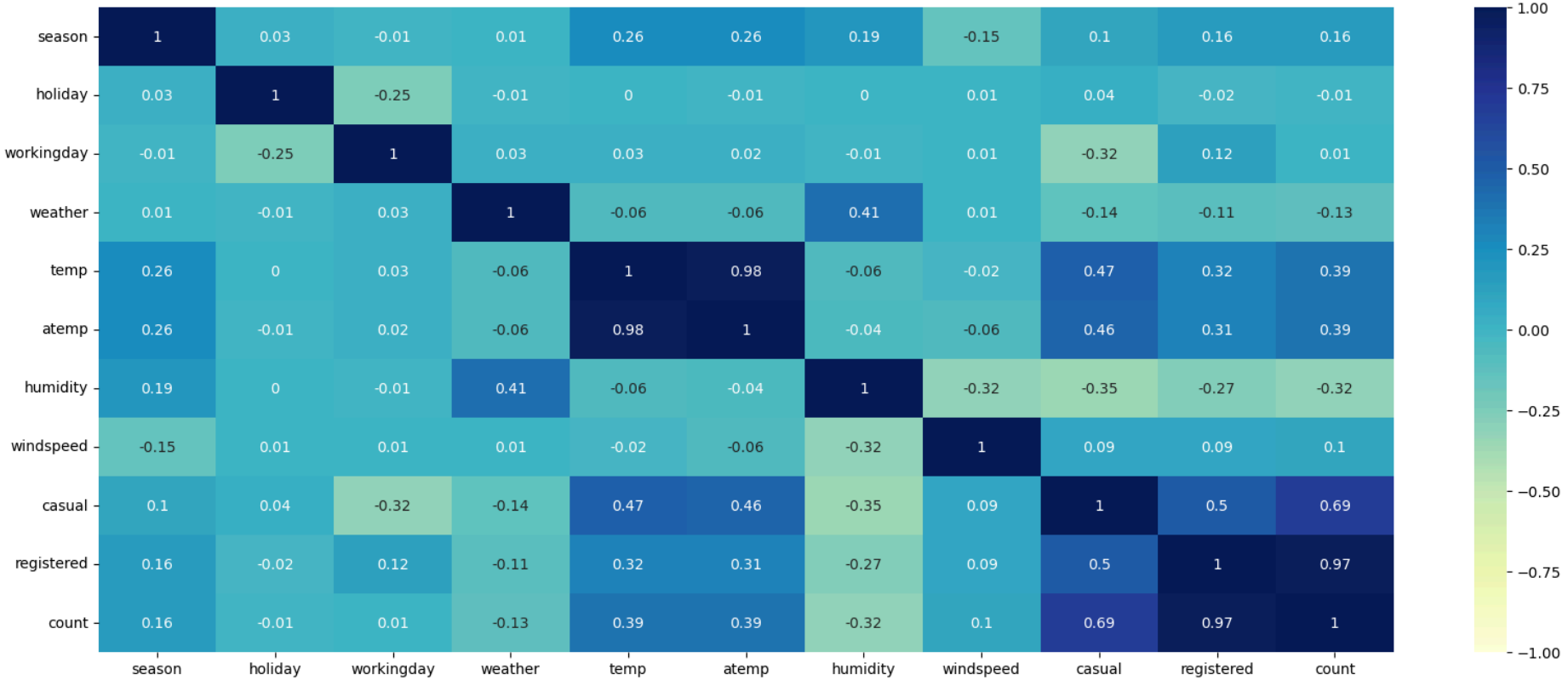
	index	registered
0	1987	539
1	2011	532
2	2059	540
3	2179	521
4	2371	516

- These are the outier for registered

Relationship

In [171…

```
data=df.copy()
data.drop('datetime',axis=1,inplace=True)
cor=data.corr()
plt.figure(figsize=(20,8))
sns.heatmap(cor.round(2),annot=True,vmin=-1,vmax=1,center=0,cmap='YlGnBu')
plt.show()
```



- "atemp" is very positively correlated with "temp".

- "temp" and "atemp" are both positively correlated with "season".
- "Registered" is also very positively correlated with "count".
- "Casual" is also positively correlated with "count".
- "atemp" and "temp" are also positively correlated with "casual".
- "Humidity" is correlated with "weather".

Test 1

- Check if there any significant difference between the no. of bike rides on Weekdays and Weekends?

Null Hypothesis (H0):There is no significant difference between the no. of bike rides on Weekdays

Alternate Hypothesis (H1):There is significant difference between the no. of bike rides on Weekdays

- Test : Sample Independent T-test ##### Significance level: 5%

In [172…

```
from scipy.stats import ttest_ind
weekdays=df[df['workingday']==1]['count']
weekend=df[df['workingday']==0]['count']
statistic,pvalue=ttest_ind(weekdays,weekend,alternative='two-sided')
print("Statistic value ",statistic)
print("P-value",pvalue)
if pvalue <= 0.05 :
    print("Reject null ,There is significant difference between the no. of bike rides on Weekdays")
else:
    print("Failed to reject null,There no significant difference between the no. of bike rides on Weekdays")
```

```
Statistic value  1.2096277376026694
P-value 0.22644804226361348
Failed to reject null,There no significant difference between the no. of bike rides on Weekdays
```

Test 2

- Check if the demand of bicycles on rent is the same for different Weather conditions?

Null Hypothesis (H0): The demand of bicycles on rent is the same for different Weather conditions

Alternate Hypothesis (H1): The demand of bicycles on rent is the not same for different Weather conditions

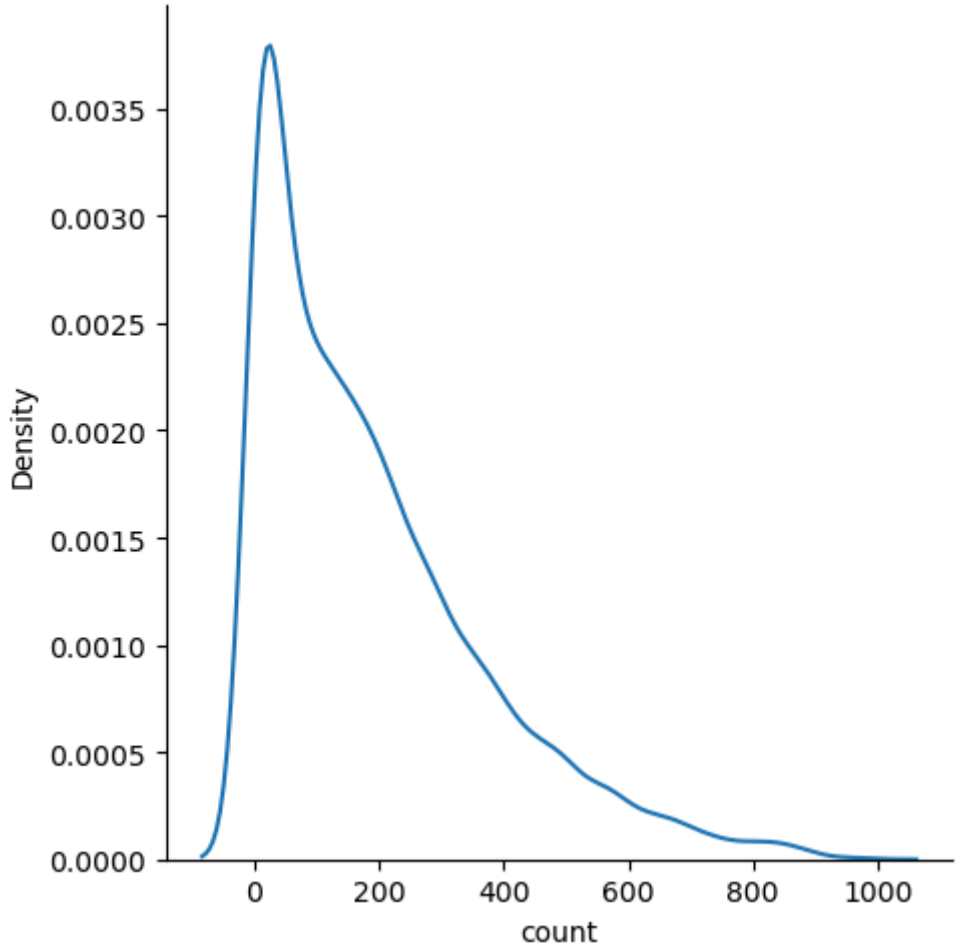
- Test : One-way ANOVA test ##### Significance level: 5%

Check assumptions of the test

- Normality

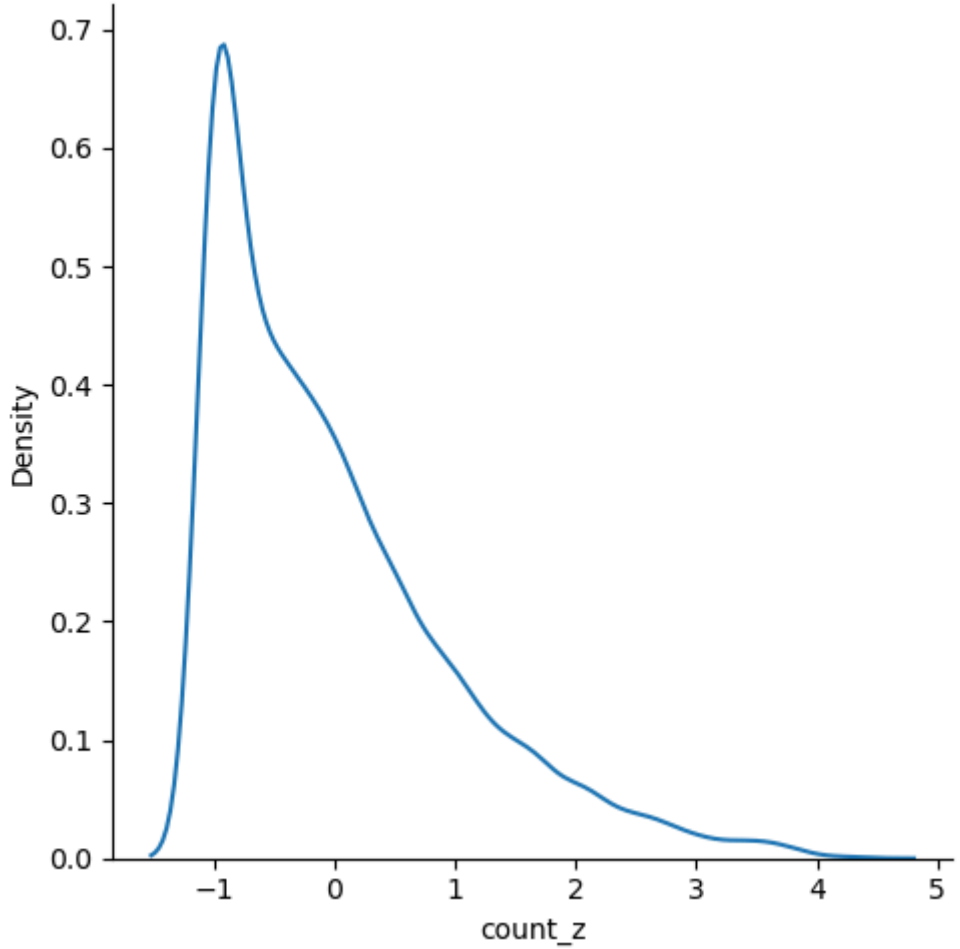
In [173…

```
sns.displot(data=df,x='count',kind='kde')
plt.show()
```



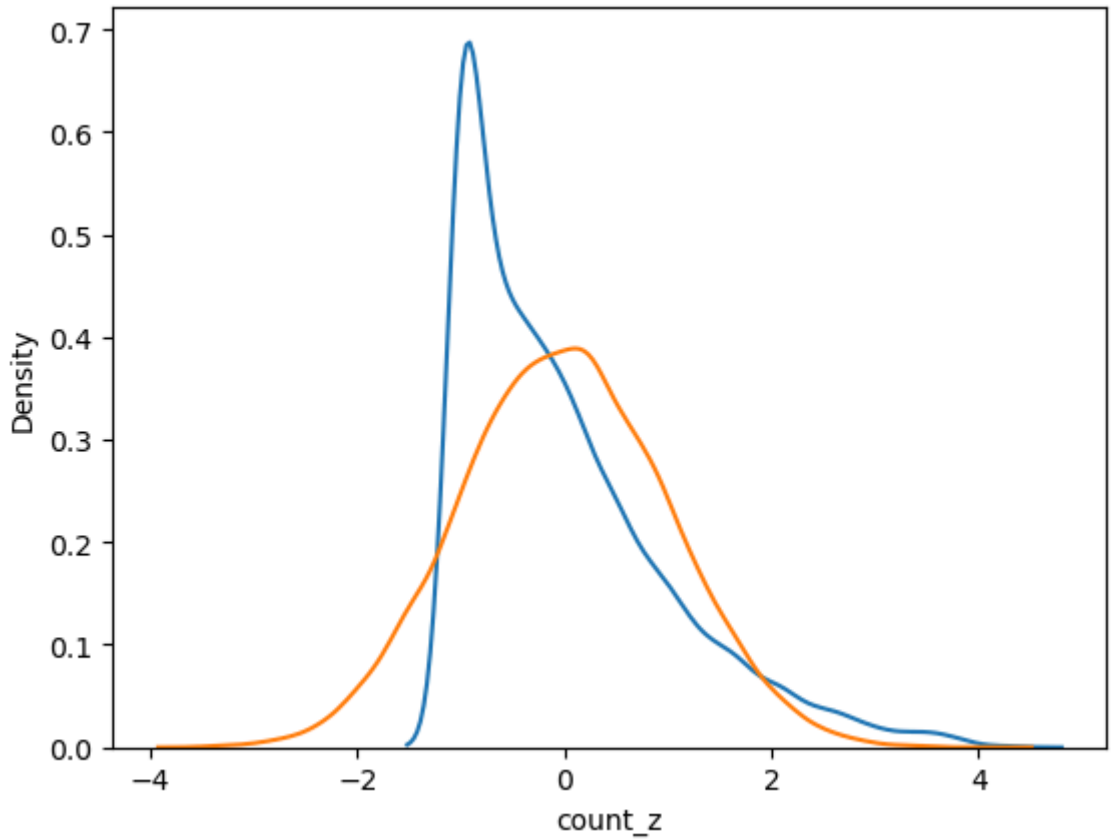
In [174…
```
df['count_z']=(df['count']-df['count'].mean())/df['count'].std()
```

In [175…
```
sns.displot(data=df,x='count_z',kind='kde')
plt.show()
```



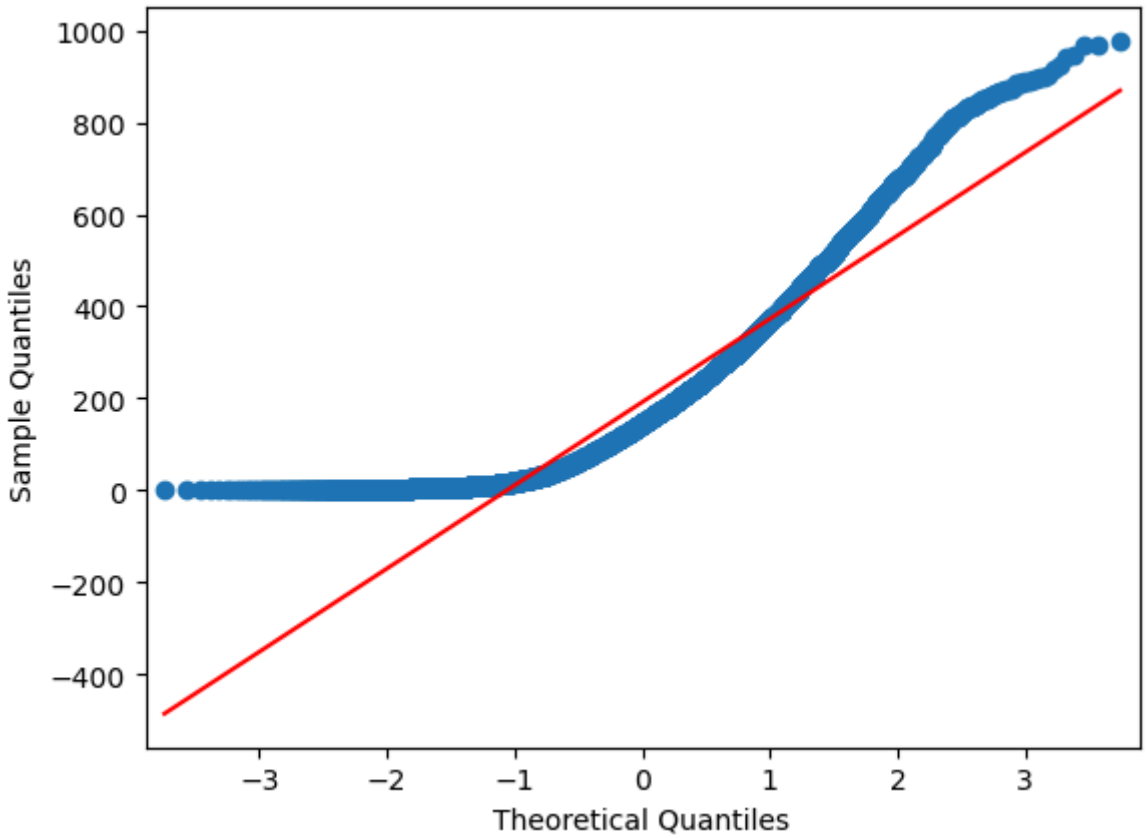
In [176…

```
stz=np.random.normal(0,1,10000)
sns.distplot(df['count_z'],hist=False)
sns.distplot(stz,hist=False)
plt.show()
```



In [177…

```
# qq plot
from statsmodels.graphics.gofplots import qqplot
qqplot(df['count'],line='s')
plt.show()
```



In [178…

```
# shapiro test
count_subset=df['count'].sample(100)
from scipy.stats import shapiro
static,pvalue=shapiro(count_subset)
print("static value :",static)
print("P-value:",pvalue)
if pvalue<0.05:
    print("Reject null, Data follows non-normal distribution ")
else:
    print('Failed to reject null, it follows normal distribution')
```

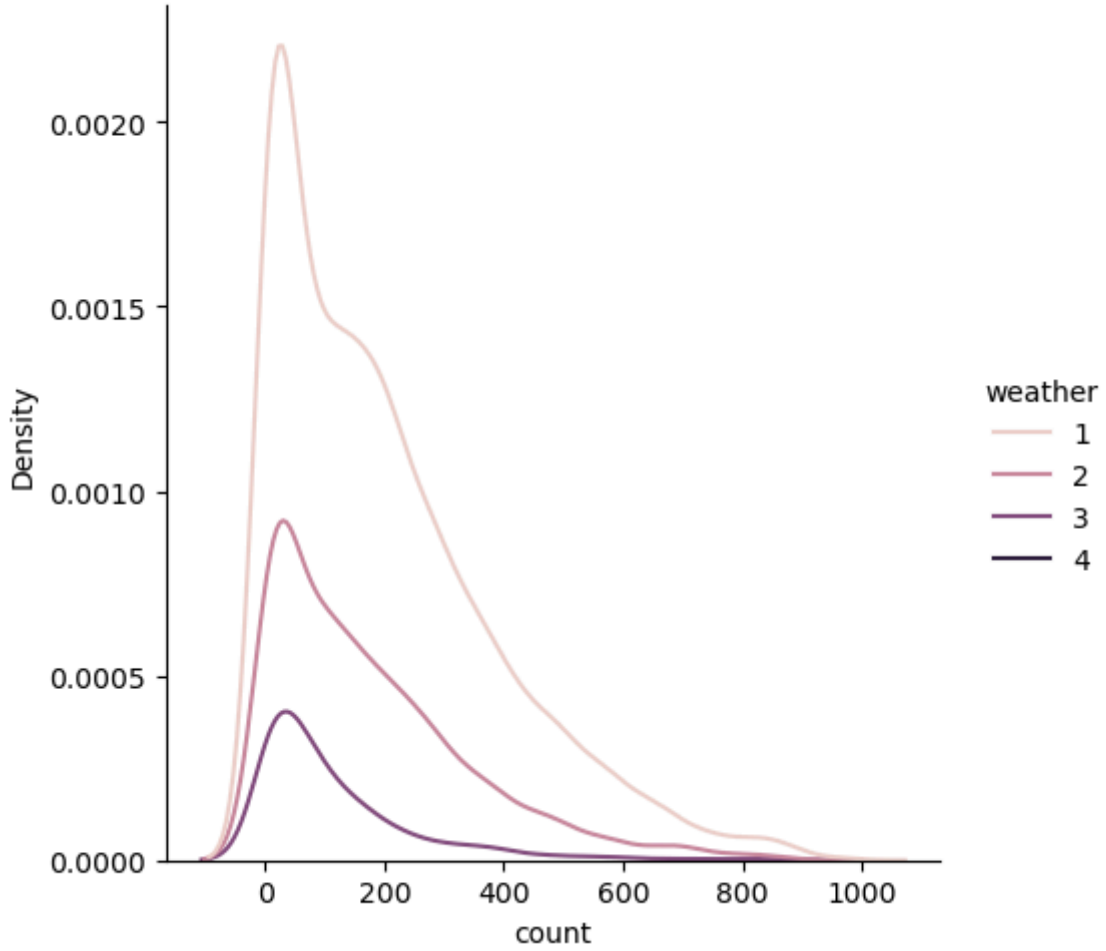
```
static value : 0.8054764866828918
P-value: 3.6596600749838615e-10
Reject null, Data follows non-normal distribution
```

The data is not normally distrubuted

Test for variance

In [179…

```
sns.displot(data=df,x='count',hue='weather',kind='kde')
plt.show()
```



In [180…

```
# Levene
from scipy.stats import levene

#h0: variance same
#h1 : variance is not same
```

```
w1=df[df['season']==1]['count']
w2=df[df['season']==2]['count']
w3=df[df['season']==3]['count']
w4=df[df['season']==4]['count']

s,p=levene(w1,w2,w3,w4)

if p<0.05:
    print('variance is not same')
else:
    print("variance is  same")
```

```
variance is not same
```

The assumption not hold true, so we use hear kruskal

In [181…

```
from scipy.stats import kruskal
k_stat,pvalue=kruskal(w1,w2,w3,w4)
print('k_stat',k_stat)
print('pvalue',pvalue)
if pvalue < 0.05:
    print("Reject null, The demand of bicycles on rent is the not same for different Weather conditions")
else:
    print('Falied to reject, The demand of bicycles on rent is the  same for different Weather conditions')
```

```
k_stat 699.6668548181988
pvalue 2.479008372608633e-151
Reject null, The demand of bicycles on rent is the not same for different Weather conditions
```

Test 3

- Check if the demand of bicycles on rent is the same for different Seasons?

Null Hypothesis (H0): The demand of bicycles on rent is the same for different Seasons

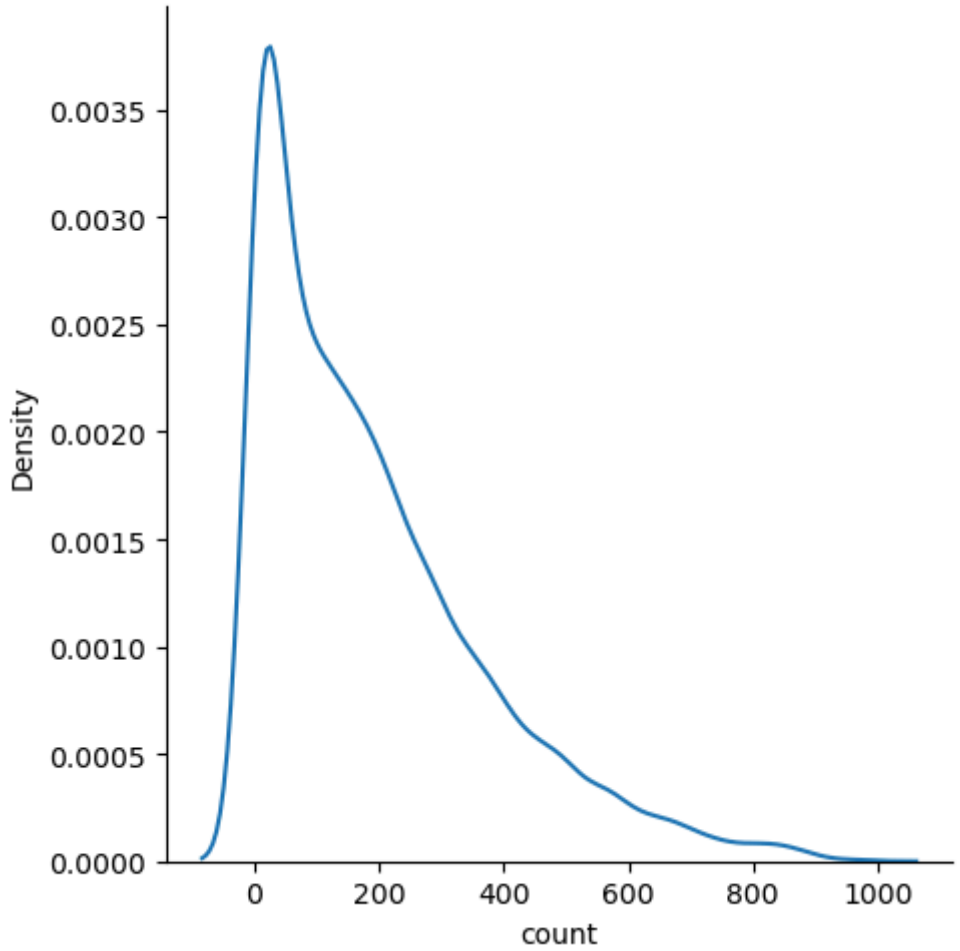
Alternate Hypothesis (H1): The demand of bicycles on rent is the not same for different Seasons

- Test : One-way ANOVA test ##### Significance level: 5%

Check assumptions of the test

- Normality

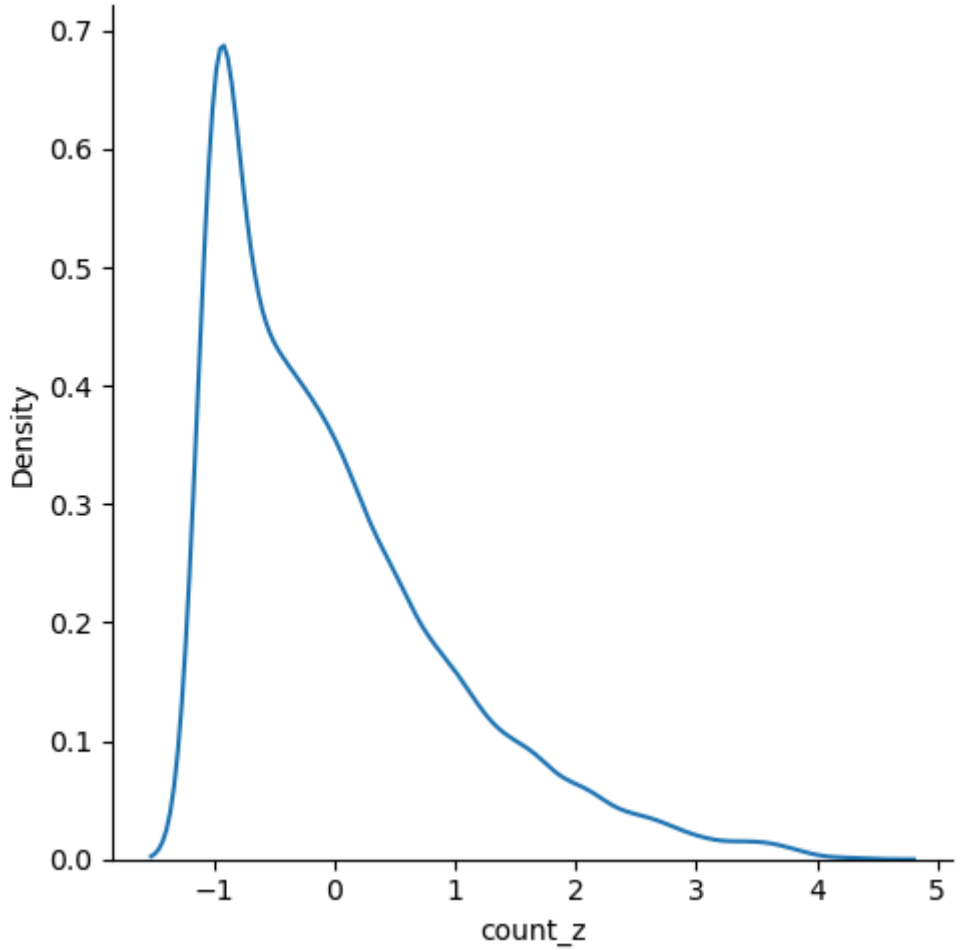
In [182…]
```
sns.displot(data=df,x='count',kind='kde')
plt.show()
```



In [183…]
```
df['count_z']=(df['count']-df['count'].mean())/df['count'].std()
```

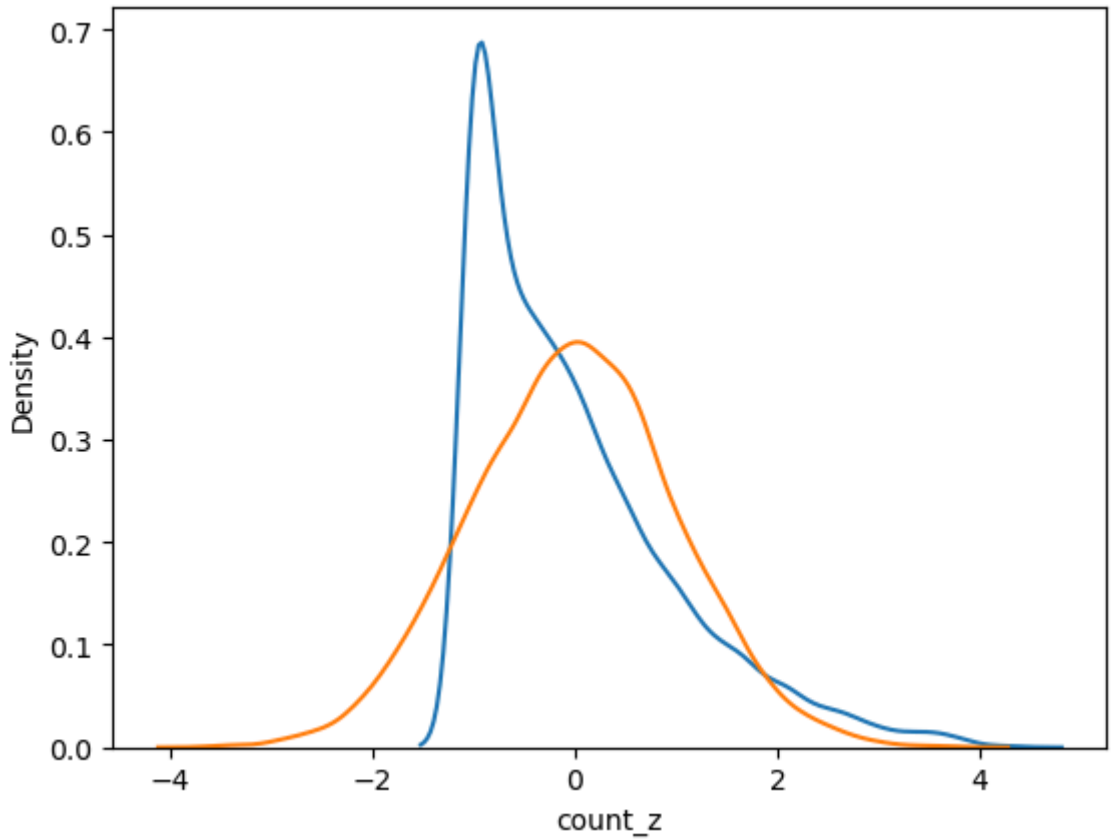
In [184…

```
sns.displot(data=df,x='count_z',kind='kde')
plt.show()
```



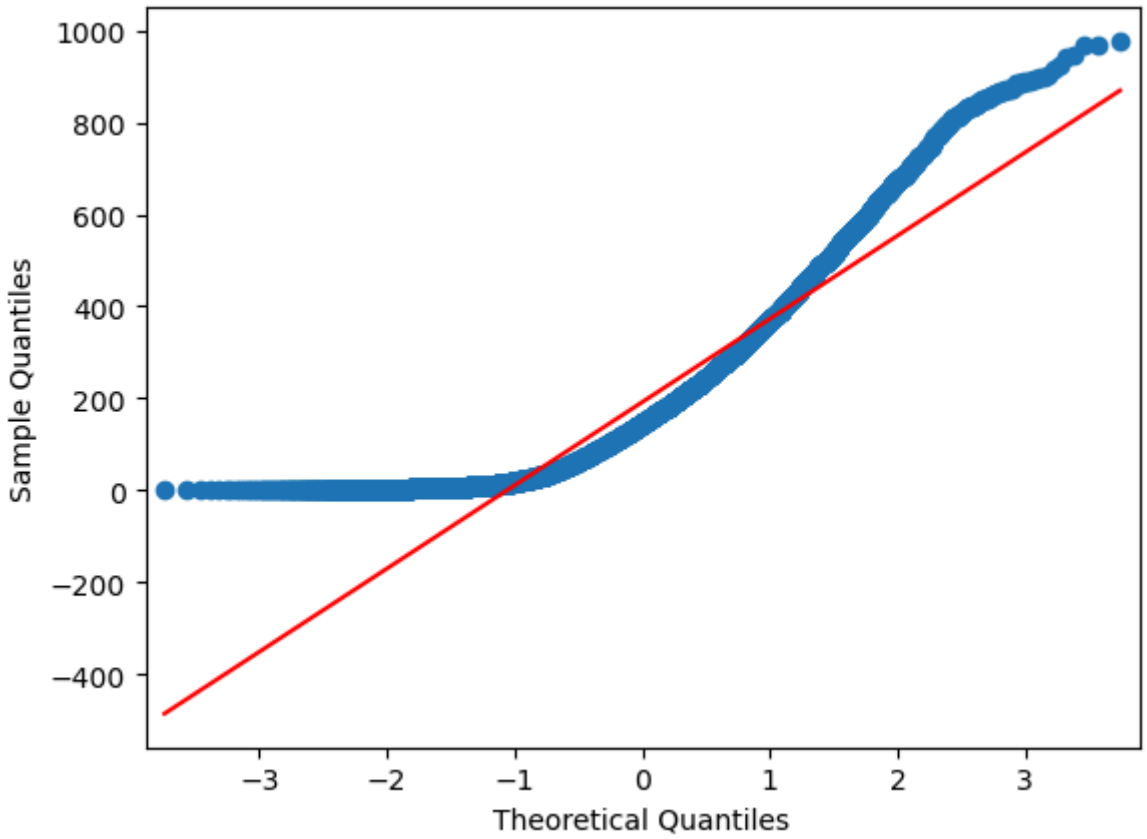
In [185…

```
stz=np.random.normal(0,1,10000)
sns.distplot(df['count_z'],hist=False)
sns.distplot(stz,hist=False)
plt.show()
```



In [186…

```
# qq plot
from statsmodels.graphics.gofplots import qqplot
qqplot(df['count'],line='s')
plt.show()
```



In [187…

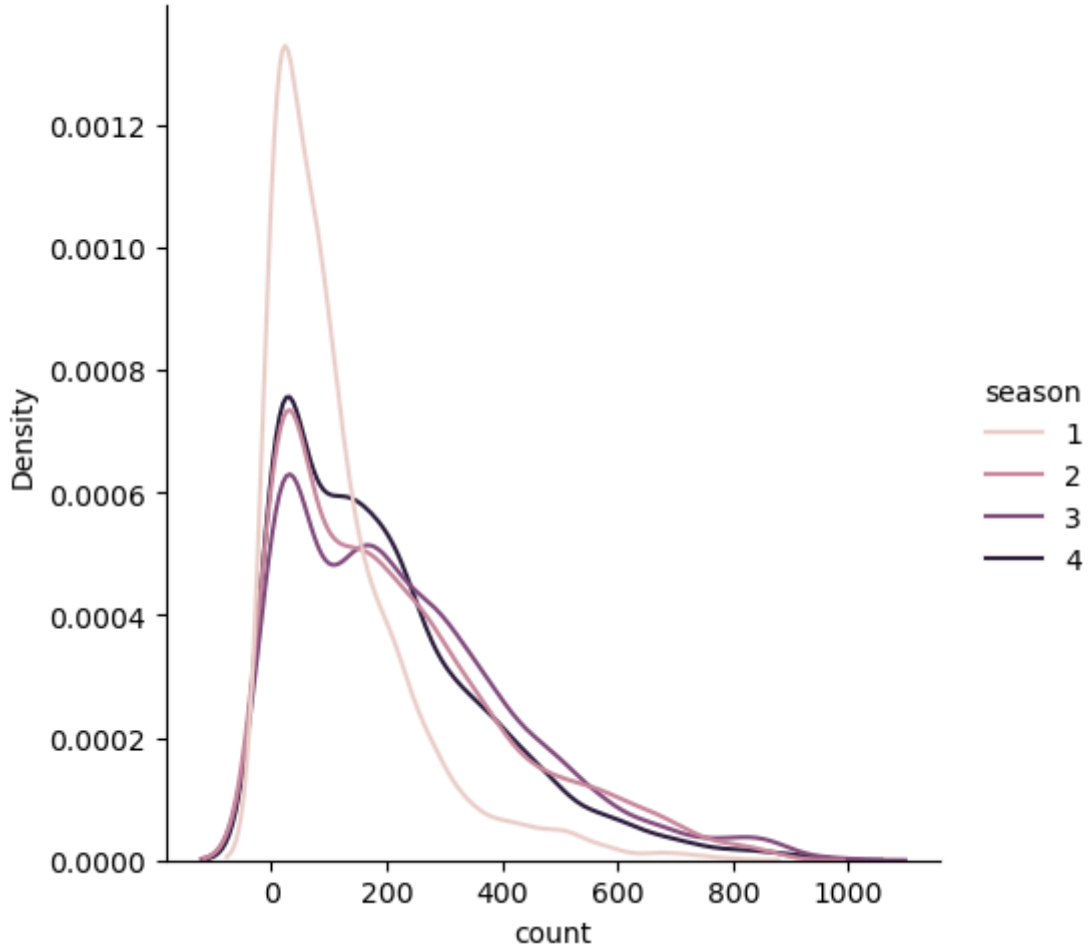
```
# shapiro test
count_subset=df['count'].sample(100)
from scipy.stats import shapiro
static,pvalue=shapiro(count_subset)
print("static value :",static)
print("P-value:",pvalue)
if pvalue<0.05:
    print("Reject null, Data follows non-normal distribution ")
else:
    print('Failed to reject null, it follows normal distribution')
```

```
static value : 0.8473027944564819
P-value: 9.439723669402156e-09
Reject null, Data follows non-normal distribution
```

The data is not normally distrubuted

Test for variance

In [188…
```
sns.displot(data=df,x='count',hue='season',kind='kde')
plt.show()
```



In [189…
```
# Levene
s1=df[df['season']==1]['count']
s2=df[df['season']==2]['count']
s3=df[df['season']==3]['count']
s4=df[df['season']==4]['count']
```

```
levene(s1,s2,s3,s4)
p=1.0147116860043298e-118

if p < 0.05 :
    print("variance is not same")
else:
    print("variance is same")
```

```
variance is not same
```

assumptions of ANOVA don't hold, we need Kruskal Wallis

In [190…

```
kruskal_stat, p_value = kruskal(s1,s2,s3,s4)
print('kruskal_stat :',kruskal_stat)
print("p_value :",p_value)
if p_value<0.05:
  print("Reject null,The demand of bicycles on rent is the not same for different Seasons")
else:
    print("failed to reject null,The demand of bicycles on rent is the  same for different Seasons")
```

```
kruskal_stat : 699.6668548181988
p_value : 2.479008372608633e-151
Reject null,The demand of bicycles on rent is the not same for different Seasons
```

Test 4:

- Check if the Weather conditions are significantly different during different Seasons

Null Hypothesis (H0): Weather is independent on season

Alternate Hypothesis (H1): Weather is not independent on season

- Test : Chi-square test ##### Significance level: 5%

In [191…

```
# Contingency Table
ct=pd.crosstab(df['weather'],df['season'])
ct
```

Out[191]:

season	1	2	3	4
weather				
1	1759	1801	1930	1702
2	715	708	604	807
3	211	224	199	225
4	1	0	0	0

In [192…

```
from scipy.stats import chi2_contingency
chi2_contingency(ct)
```

Out[192]:

```
Chi2ContingencyResult(statistic=49.15865559689363, pvalue=1.5499250736864862e-07, dof=9, expected_freq=array([[1.77454639e+03,
1.80559765e+03, 1.80559765e+03, 1.80625831e+03],
       [6.99258130e+02, 7.11493845e+02, 7.11493845e+02, 7.11754180e+02],
       [2.11948742e+02, 2.15657450e+02, 2.15657450e+02, 2.15736359e+02],
       [2.46738931e-01, 2.51056403e-01, 2.51056403e-01, 2.51148264e-01]]))
```

In [193…

```
pvalue=1.5499250736864862e-07
if pvalue < 0.05:
    print("Reject null,Weather is not  independent  on season ")
else:
    print("Failed to reject null,Weather is  independent  on season")
```

```
Reject null,Weather is not  independent  on season
```

In [ ]:

Insights

- During the summer and fall seasons, bike rentals surge compared to other seasons.
- Bike rentals are higher on holidays.
- The data also indicates that more bikes are rented on weekends or holidays, suggesting a correlation with non working days.
- Conversely, bike rentals decrease during adverse weather conditions such as rain, thunderstorms, snow, or fog.
- There is a noticeable drop in bike rentals when humidity levels fall below 20%.
- Similarly, when the temperature drops below 10 degrees Celsius, bike rentals decrease.

- Bike rentals also decline when wind speeds exceed 35 km/h.

In [ ]:

Recommendations

- During summer and fall, it's advisable for the company to increase its bike inventory since demand peaks during these seasons compared to others.
- Based on a significance level of 0.05, it appears that whether it's a working day or not doesn't significantly impact bike rentals.
- On days with extremely low humidity, it's recommended to reduce the number of bikes available for rent, as demand tends to be lower during such conditions.
- During very cold days, especially when the temperature drops below 10 degrees Celsius, it's wise for the company to decrease bike availability.
- Similarly, during thunderstorms or when wind speeds exceed 35 km/h, reducing the number of bikes in stock for rental is advisable due to decreased demand during such weather conditions.