In [1]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

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

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

Out[3]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	re
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	

In [4]:
```
df.tail()
```

Out[4]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casua
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	1
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	1
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	

In [5]:
```
df.shape
```

Out[5]:
```
(10886, 12)
```

In [6]:
```
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 [7]:
```
#checking unique values in the dataframe
for i in df.columns :
    print(i,":",df[i].nunique())
```

```
datetime : 10886
season : 4
holiday : 2
workingday : 2
weather : 4
temp : 49
atemp : 60
humidity : 89
windspeed : 28
casual : 309
registered : 731
count : 822
```

In [8]:
```
df["datetime"]
```

Out[8]:
```
0        2011-01-01 00:00:00
1        2011-01-01 01:00:00
2        2011-01-01 02:00:00
3        2011-01-01 03:00:00
4        2011-01-01 04:00:00
                 ...
10881    2012-12-19 19:00:00
10882    2012-12-19 20:00:00
10883    2012-12-19 21:00:00
10884    2012-12-19 22:00:00
10885    2012-12-19 23:00:00
Name: datetime, Length: 10886, dtype: object
```

In [9]:
```
#convert the date time object to date time format
df['datetime']=pd.to_datetime(df['datetime'])
```

In [10]:
```
#From above , we can say that Season,holiday,workingday,weather are 4 categorical
#7 are numerical vairables
```

In [11]:
```
#Cheking the span of data given,it is about 2 years
df["datetime"].min(),df["datetime"].max()
```

Out[11]:
```
(Timestamp('2011-01-01 00:00:00'), Timestamp('2012-12-19 23:00:00'))
```

In [12]:
```
#making a shallow copy of dataframe and modyfing datetime
df1=df.copy()
df1["year"]=df1["datetime"].dt.year
df1["month"]=df1["datetime"].dt.month
df1["day"]=df1["datetime"].dt.day
```

In [13]:
```
df.describe()
```

Out[13]:

	season	holiday	workingday	weather	temp	atemp	hu
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.0
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.8
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.2
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.0
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.0
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.0
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.0
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.0

In [14]:
```
#checking the null values in the dataframe
df.isnull().sum()
```

Out[14]:
```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

In [15]:
```
#checking for duplicates
df.duplicated().sum()
```

Out[15]:
```
0
```

In [16]:
```
#rental count as per seasons
df.groupby("season")["count"].sum()
```

Out[16]:
```
season
1    312498
2    588282
3    640662
4    544034
Name: count, dtype: int64
```

In [17]:
```
#count as per holidays
df.groupby("holiday")["count"].sum()
```

Out[17]:
```
holiday
0    2027668
1      57808
Name: count, dtype: int64
```

In [18]:
```
#count as per working day
df.groupby("workingday")["count"].sum()
```

Out[18]:
```
workingday
0     654872
1    1430604
Name: count, dtype: int64
```

In [19]:
```
#count as per wheather conditions
df.groupby("weather")["count"].sum()
```

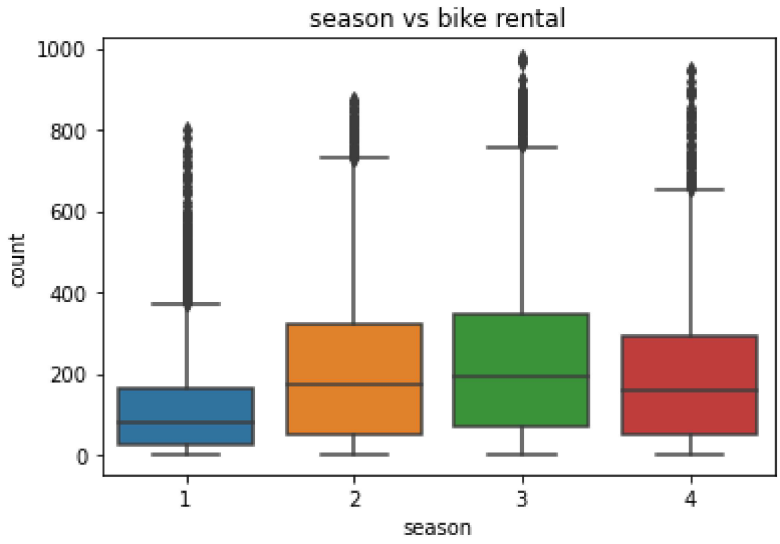
Out[19]:
```
weather
1    1476063
2     507160
3     102089
4        164
Name: count, dtype: int64
```

Cheking outliers

In [20]:
```
import seaborn as sns
```

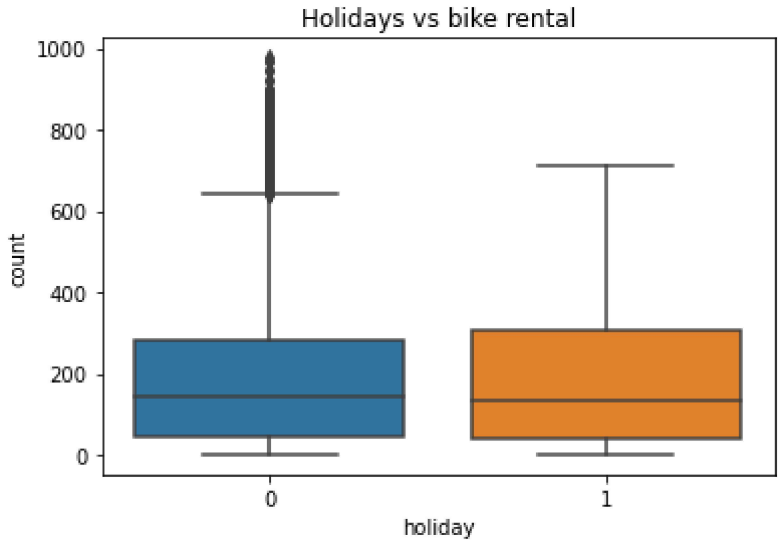
In [21]:
```
sns.boxplot(x="season",y='count',data=df)
plt.title("season vs bike rental")
```

Out[21]:
```
Text(0.5, 1.0, 'season vs bike rental')
```



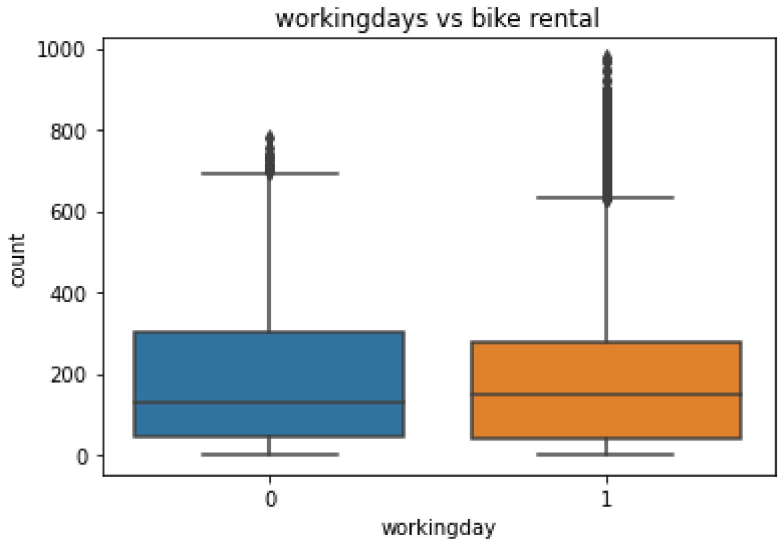
In [22]:
```
sns.boxplot(x="holiday",y="count",data=df)
plt.title("Holidays vs bike rental")
```

Out[22]:
```
Text(0.5, 1.0, 'Holidays vs bike rental')
```



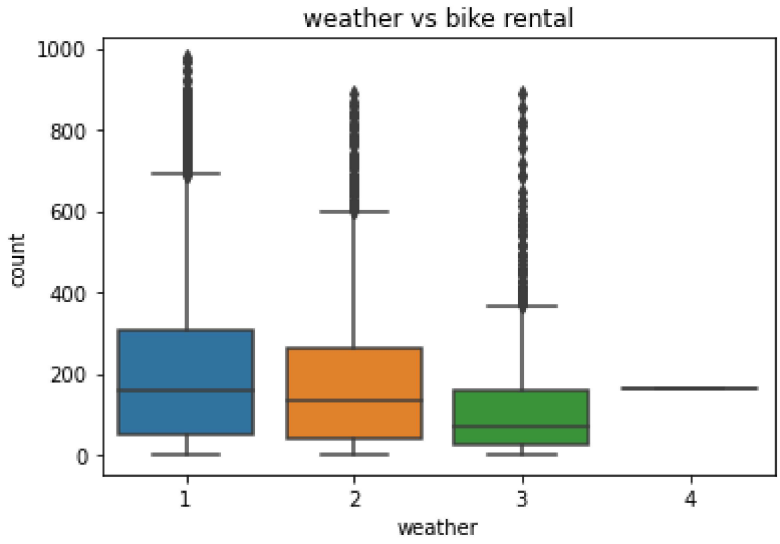
In [23]:
```
sns.boxplot(x="workingday",y="count",data=df)
plt.title("workingdays vs bike rental")
```

Out[23]: Text(0.5, 1.0, 'workingdays vs bike rental')



In [24]:
```
sns.boxplot(x="weather",y="count",data=df)
plt.title("weather vs bike rental")
```

Out[24]: Text(0.5, 1.0, 'weather vs bike rental')



Removing the outliers

In [25]:
```
Q1 = df["count"].quantile(0.25)
Q3 = df["count"].quantile(0.75)
IQR = Q3 - Q1

df = df[((df["count"] >= (Q1 - 1.5 * IQR)) &(df["count"] <= (Q3 + 1.5 * IQR)))]
```

In [26]:
```
df.head()
```

Out[26]:

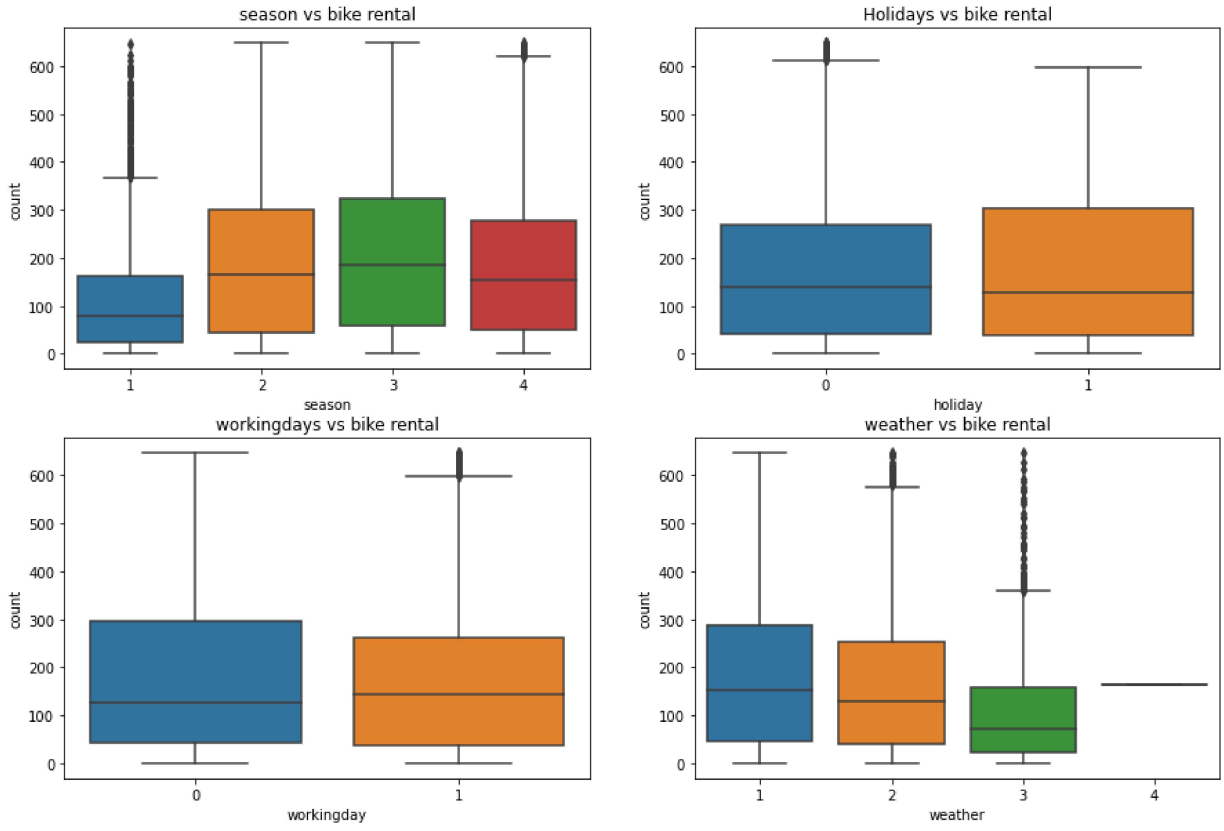
	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	re
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	

In [27]:
```
df.shape
```

Out[27]: (10586, 12)

boxplot after outlier removed

In [28]:
```
fig=plt.figure(figsize=(15,10))

plt.subplot(2,2,1)
sns.boxplot(x="season",y='count',data=df)
plt.title("season vs bike rental")

plt.subplot(2,2,2)
sns.boxplot(x="holiday",y="count",data=df)
plt.title("Holidays vs bike rental")

plt.subplot(2,2,3)
sns.boxplot(x="workingday",y="count",data=df)
plt.title("workingdays vs bike rental")

plt.subplot(2,2,4)
sns.boxplot(x="weather",y="count",data=df)
plt.title("weather vs bike rental")

plt.show()
```



In [29]:
```
df.shape
```

Out[29]: (10586, 12)

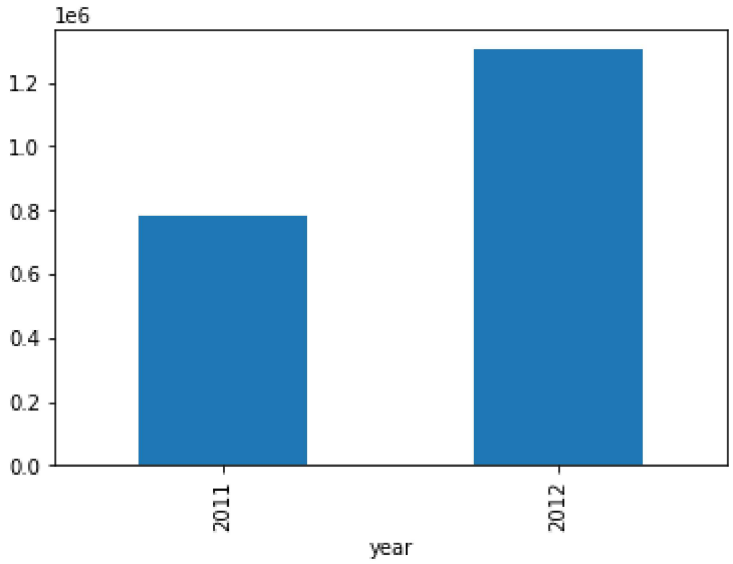
In [30]:
```
df1.shape[0]-df.shape[0]
```

Out[30]: 300

In [31]:
```
#as we observe 300 rows are removed from the dataframe
```

In [32]:
```
#Year wise rental
df1.groupby("year")["count"].sum().plot(kind="bar")
```

Out[32]: <AxesSubplot:xlabel='year'>



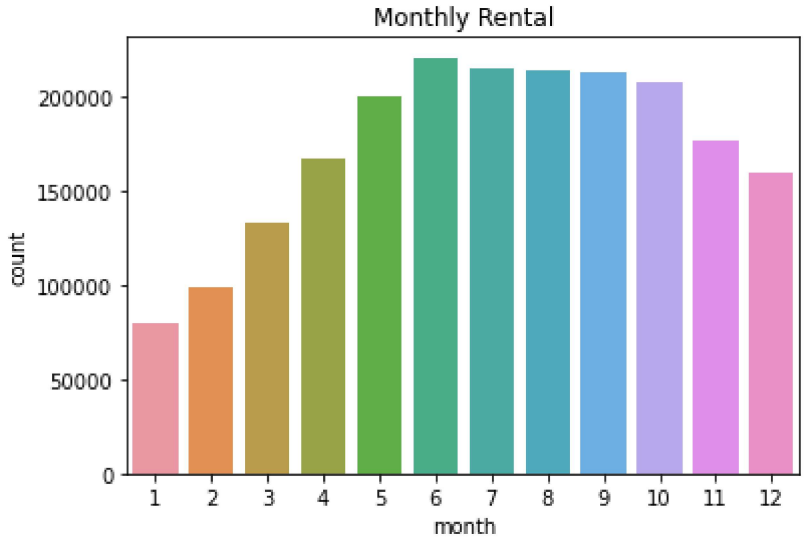
In [33]:
```
monthly_rental=df1.groupby("month")["count"].sum().reset_index()
```

In [34]:
```
monthly_rental
```

Out[34]:

	month	count
0	1	79884
1	2	99113
2	3	133501
3	4	167402
4	5	200147
5	6	220733
6	7	214617
7	8	213516
8	9	212529
9	10	207434
10	11	176440
11	12	160160

In [35]:
```
plt.title("Monthly Rental")
sns.barplot(x='month',y='count',data=monthly_rental)
plt.show()
```

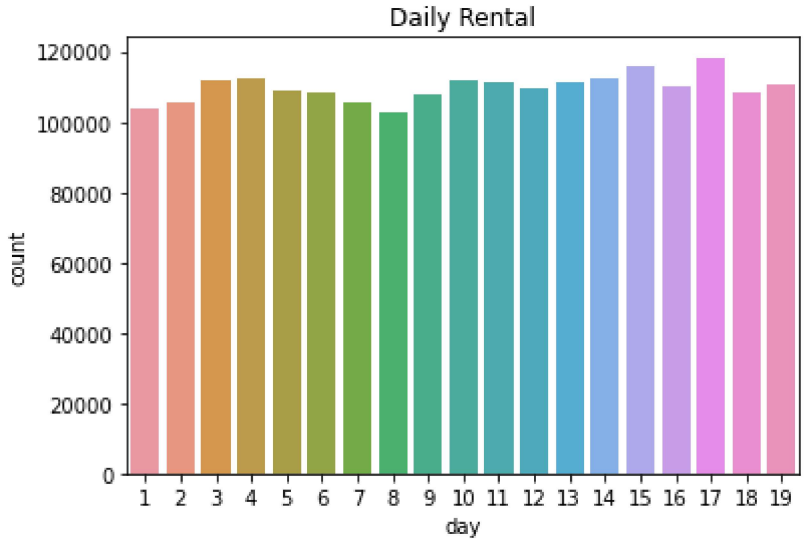


In [36]:
```
daily_rental=df1.groupby("day")["count"].sum().reset_index()
daily_rental
```

Out[36]:

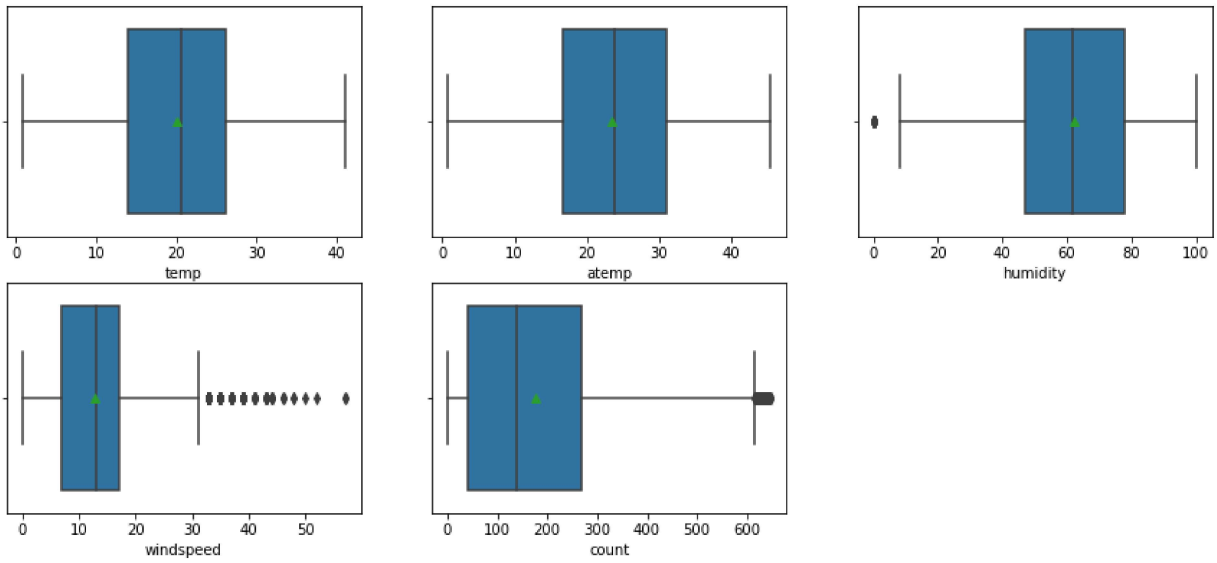
	day	count
0	1	103692
1	2	105381
2	3	111561
3	4	112335
4	5	109115
5	6	108600
6	7	105486
7	8	102770
8	9	108041
9	10	111645
10	11	111146
11	12	109257
12	13	111448
13	14	112406
14	15	115677
15	16	109837
16	17	118255
17	18	108437
18	19	110387

In [37]:
```
plt.title("Daily Rental")
sns.barplot(x='day',y='count',data=daily_rental)
plt.show()
```



Univariate anlaysis on numerical variables

In [38]:
```
plt.figure(figsize=(15,10))

#for temperature
plt.subplot(3,3,1)
sns.boxplot(x="temp",showmeans=True,data=df)


#for temperature feel
plt.subplot(3,3,2)
sns.boxplot(x="atemp",showmeans=True,data=df)


#for humidity
plt.subplot(3,3,3)
sns.boxplot(x="humidity",showmeans=True,data=df)

#for windspeed
plt.subplot(3,3,4)
sns.boxplot(x="windspeed",showmeans=True,data=df)

#for count
plt.subplot(3,3,5)
sns.boxplot(x="count",showmeans=True,data=df)

plt.show()
```

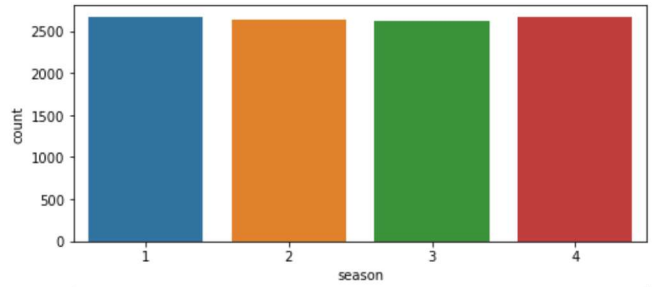


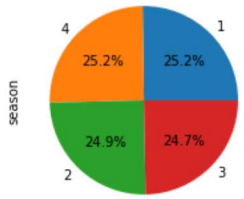
from above we can observe the mean of all the numerical attributes

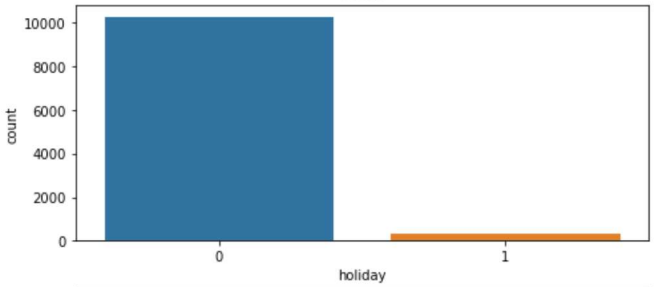
Univariate anlaysis of Categorical variables

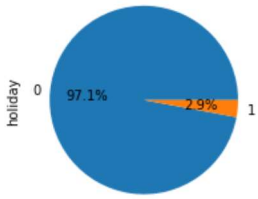
In [39]:
```
plt.figure(figsize=(17,15))



#for seasons
plt.subplot(4,2,1)
sns.countplot(x="season",data=df)

plt.subplot(4,2,2)
df["season"].value_counts().plot.pie(autopct='%1.1f%%')


#for holidays
plt.subplot(4,2,3)
sns.countplot(x="holiday",data=df)

plt.subplot(4,2,4)
df["holiday"].value_counts().plot.pie(autopct='%1.1f%%')


#for workingday
plt.subplot(4,2,5)
sns.countplot(x="workingday",data=df)

plt.subplot(4,2,6)
df["workingday"].value_counts().plot.pie(autopct='%1.1f%%')

#for weather

plt.subplot(4,2,7)
sns.countplot(x="weather",data=df)

plt.subplot(4,2,8)
df["weather"].value_counts().plot.pie(autopct='%1.1f%%')
```

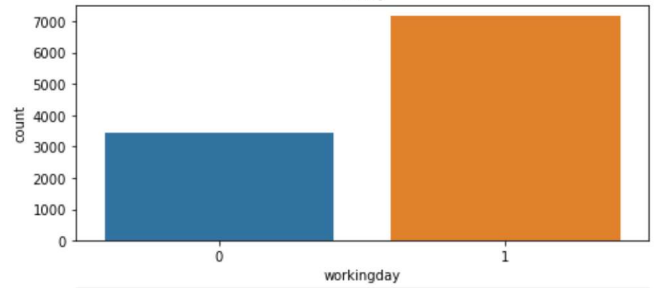
Out[39]:
```
<AxesSubplot:ylabel='weather'>
```

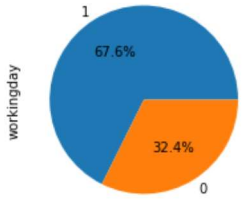


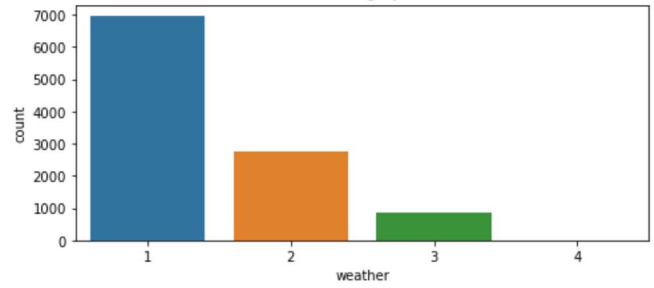


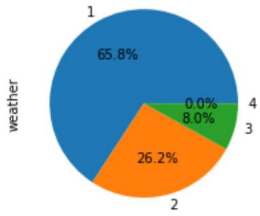






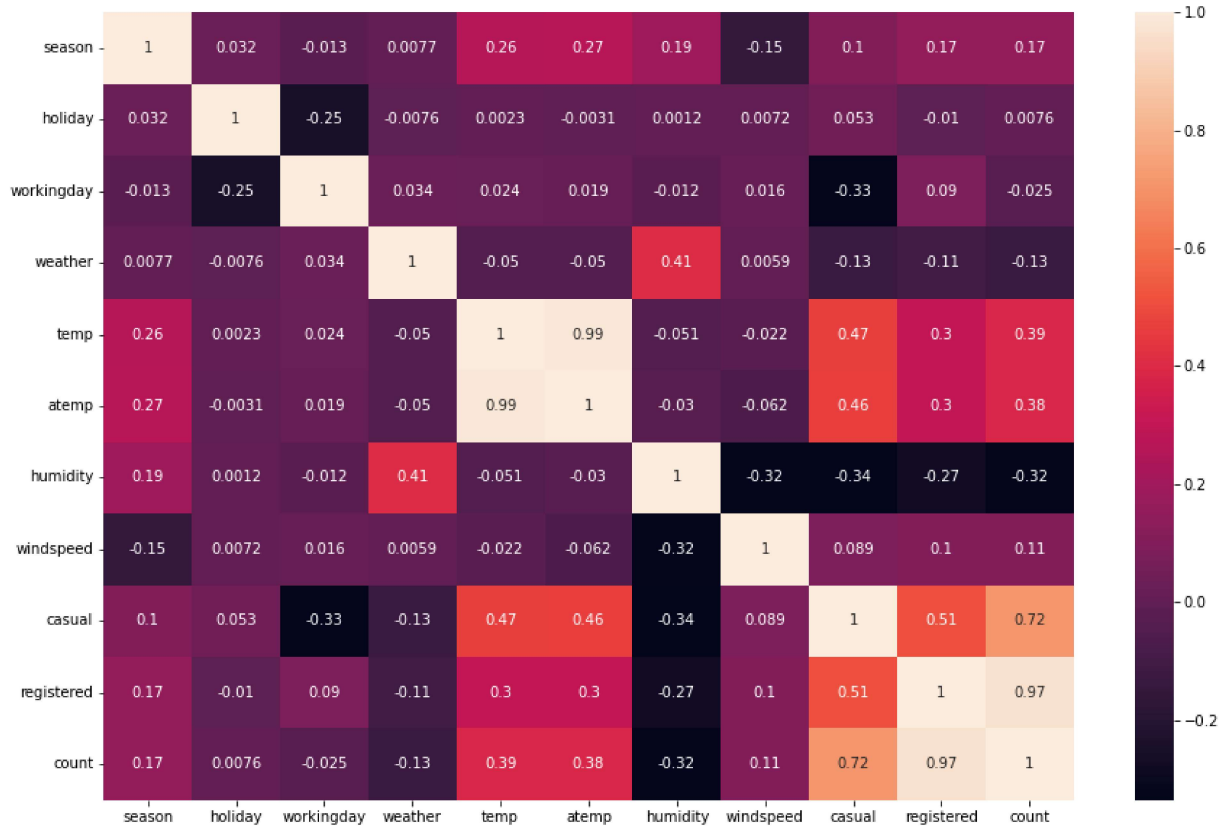






Multivariate analysis

In [40]:
```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



Hypothesis testing

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

In [41]:
```
#### defining the hypothesis
 #H0: The electrical vehicals rental count on working day and non working day are
 #Ha: The electrical vehicals rental count on working day and non working day are
alpha = 0.5
```

In [42]:
```
df2=df.copy()
```

In [43]:
```
df2["year"]=df2["datetime"].dt.year
df2["month"]=df2["datetime"].dt.month
df2["day"]=df2["datetime"].dt.day
#df2 is the copy of df with extra columns added --- year,month,day
```

In [ ]:
```

```

In [44]:
```
df.head()
```

Out[44]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	re
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	

In [45]:
```
df["workingday"].value_counts()
```

Out[45]:
```
1    7161
0    3425
Name: workingday, dtype: int64
```

In [46]:
```
workingday=df[df["workingday"]==1].sample(3000,replace = False)
workingday
```

Out[46]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
7456	2012-05-10 01:00:00	2	0	1	2	19.68	23.485	88	8.9981	2
7251	2012-05-01 12:00:00	2	0	1	2	28.70	32.575	58	15.0013	41
5663	2012-01-11 03:00:00	1	0	1	1	9.02	13.635	80	0.0000	0
4708	2011-11-09 06:00:00	4	0	1	1	12.30	16.665	87	0.0000	0
9149	2012-09-04 14:00:00	3	0	1	1	32.80	37.880	55	27.9993	72
...	...	...	...	...	...	...	...	...	...	...
3345	2011-08-09 07:00:00	3	0	1	1	29.52	34.850	74	7.0015	30
3369	2011-08-10 07:00:00	3	0	1	1	28.70	32.575	58	6.0032	16
3134	2011-07-19 12:00:00	3	0	1	1	34.44	40.910	56	6.0032	26
9373	2012-09-13 22:00:00	3	0	1	1	24.60	29.545	69	8.9981	28
6814	2012-04-02 06:00:00	2	0	1	1	14.76	15.910	71	31.0009	6

3000 rows × 12 columns

In [47]:
```
Nonworkingday=df[df["workingday"]==0].sample(3000,replace = False)
Nonworkingday
```

Out[47]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casua
9062	2012-08-19 23:00:00	3	0	0	2	25.42	29.545	73	0.0000	
4797	2011-11-12 23:00:00	4	0	0	1	16.40	20.455	58	11.0014	2
10434	2012-12-01 04:00:00	4	0	0	1	10.66	14.395	81	6.0032	
6721	2012-03-17 08:00:00	1	0	0	2	17.22	21.210	94	7.0015	6
8316	2012-07-07 21:00:00	3	0	0	1	36.08	39.395	37	12.9980	5
...	...	...	...	...	...	...	...	...	...	.
5430	2012-01-01 08:00:00	1	0	0	1	10.66	13.635	87	7.0015	
4270	2011-10-09 23:00:00	4	0	0	1	20.50	24.240	88	7.0015	3
2490	2011-06-11 16:00:00	2	0	0	2	32.80	36.365	46	12.9980	16
7145	2012-04-16 02:00:00	2	1	0	1	24.60	30.305	64	16.9979	
5426	2012-01-01 04:00:00	1	0	0	1	11.48	15.150	81	6.0032	

3000 rows × 12 columns

In [48]:
```
#means of sample
workingday["count"].mean(),Nonworkingday["count"].mean()
```

Out[48]:
```
(172.881, 180.88833333333332)
```

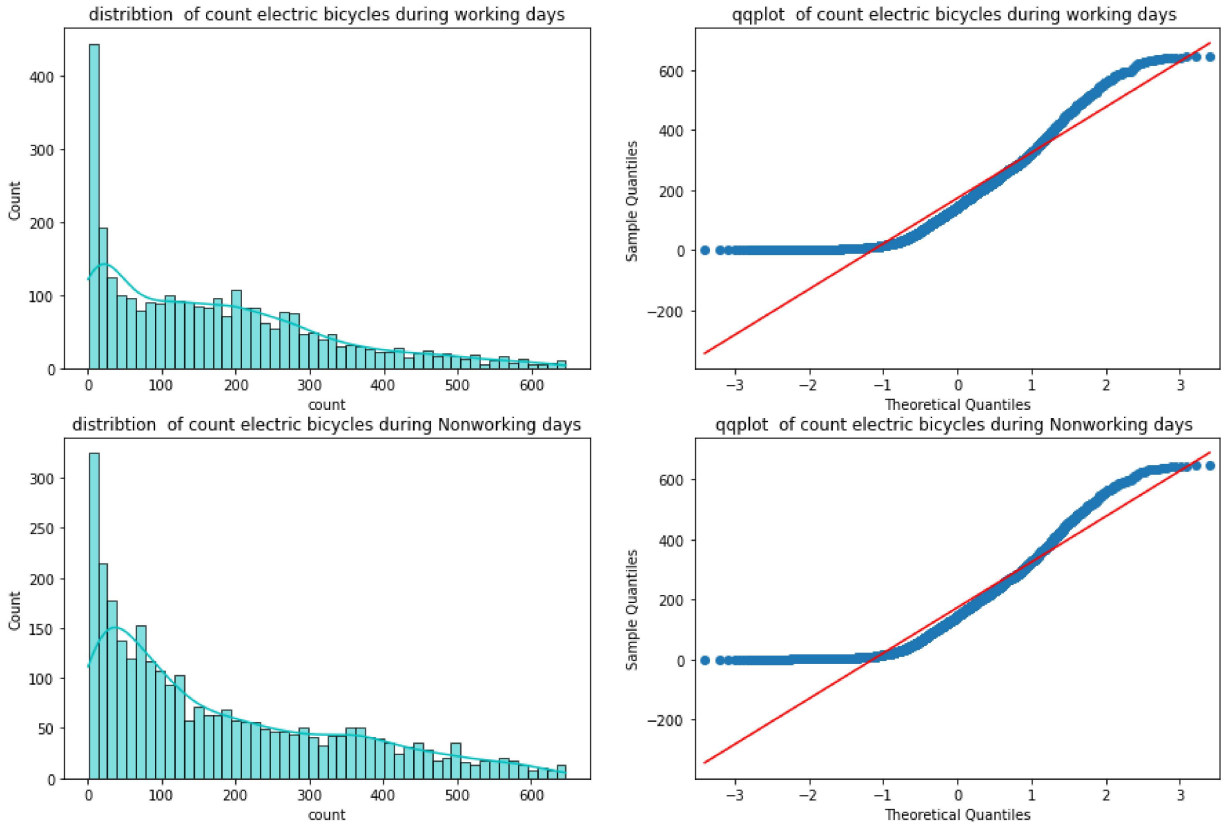
In [49]:
```
#standard deviation of sample
round(workingday["count"].std()**2,2),round(Nonworkingday["count"].std()**2,2)
```

Out[49]:
```
(22961.03, 27002.02)
```

variance is not equal for both samples

Cheking for noramality for working day

In [50]:
```
fig = plt.figure(figsize=(15,10))

ax1 = fig.add_subplot(221)
sns.histplot(data = workingday,x ="count",bins=50,kde= True , ax = ax1,color='c')
ax1.set_title("distribtion  of count electric bicycles during working days")

ax2= fig.add_subplot(222)
sm.qqplot(workingday["count"],line ='s',ax = ax2 )
ax2.set_title("qqplot  of count electric bicycles during working days ")


ax3 = fig.add_subplot(223)
sns.histplot(data = Nonworkingday,x ="count",bins=50,kde= True , ax = ax3,color='
ax3.set_title("distribtion  of count electric bicycles during Nonworking days")


ax4= fig.add_subplot(224)
sm.qqplot(workingday["count"],line ='s',ax = ax4 )
ax4.set_title("qqplot  of count electric bicycles during Nonworking days ")

plt.show()
```



In [51]:
```
#calucalting the p-value and test statistic
#setting alerantive as greater as the distriburion is right skewed
t_test,p_value=stats.ttest_ind(workingday["count"],Nonworkingday['count'],alterna
print("t_test :",t_test," P_value: ",p_value)
```

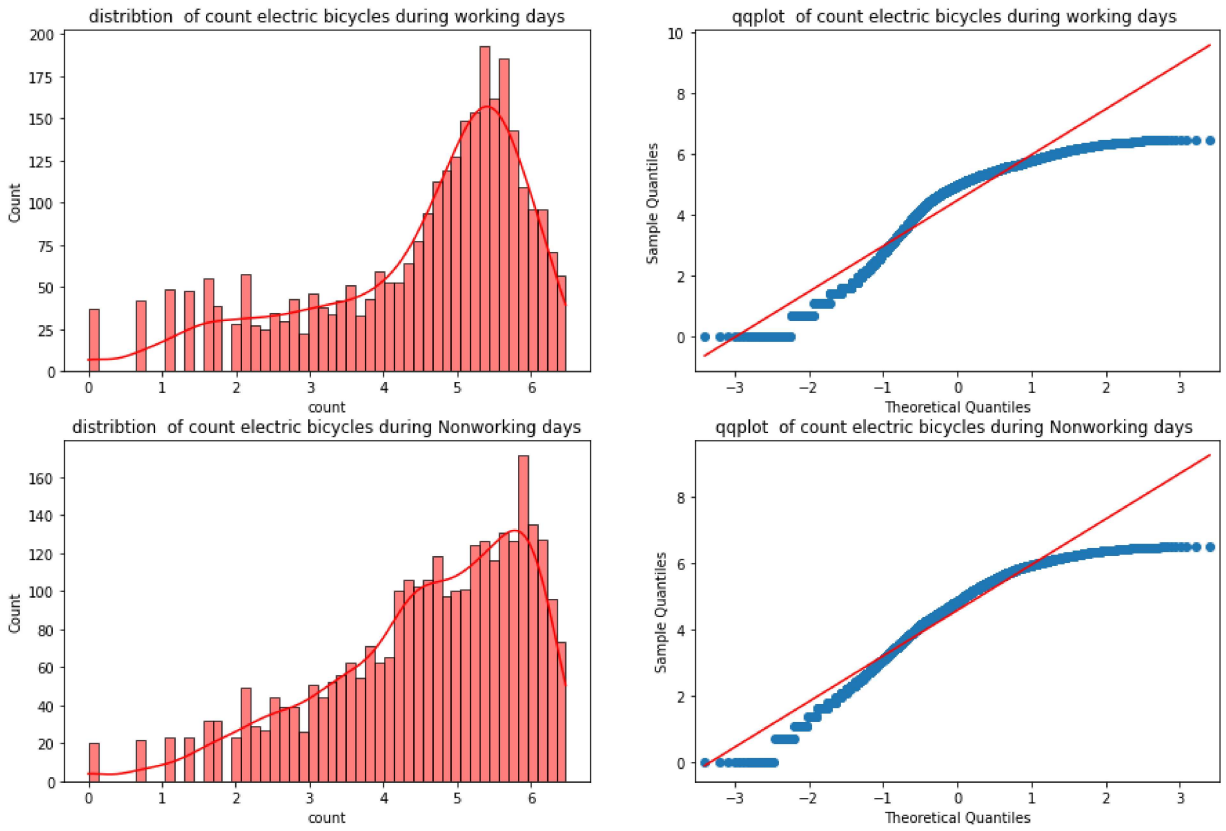
```
t_test : -1.9621132532314876  P_value:  0.9751022645938946
```

#1.From above p value is greater than alpha so we fail to reject the null hypothesis. #2.Since the

distribution is right is skewed and the and the variance is not equal will do a log transformation

log normal distribution

In [52]:
```
fig = plt.figure(figsize=(15,10))

ax1 = fig.add_subplot(221)
sns.histplot(data = np.log(workingday["count"]),bins=50,kde= True , ax = ax1,colo
ax1.set_title("distribtion  of count electric bicycles during working days")

ax2= fig.add_subplot(222)
sm.qqplot(np.log(workingday["count"]),line ='s',ax = ax2 )
ax2.set_title("qqplot  of count electric bicycles during working days ")


ax3 = fig.add_subplot(223)
sns.histplot(data = np.log(Nonworkingday["count"]),bins=50,kde= True , ax = ax3,c
ax3.set_title("distribtion  of count electric bicycles during Nonworking days")


ax4= fig.add_subplot(224)
sm.qqplot(np.log(Nonworkingday["count"]),line ='s',ax = ax4 )
ax4.set_title("qqplot  of count electric bicycles during Nonworking days ")

plt.show()
```



In [53]:
```
round(np.log(workingday["count"]).std()**2,2),round(np.log(Nonworkingday["count"]
```

Out[53]: (2.26, 1.88)

In [54]:
```
log_workingday=np.log(workingday['count']).sample(3000)
log_Nonworkingday=np.log(Nonworkingday['count']).sample(3000)
```

In [55]:
```
t_test1,p_value1=stats.ttest_ind(workingday["count"],Nonworkingday['count'],alter
print("t_test1 :",t_test1," P_value1 : ",p_value1)
```

```
t_test1 : -1.9621132532314876  P_value1 :  0.9751022645938946
```

In [56]:
```
if p_value1 < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject null hypothesis")
```

```
Fail to reject null hypothesis
```

After doing log transformation on sample population the distribution looks almost like normal so we can calculate the test statistic and the p value.

```
Conclusion : As the P value > alpha , we fail to reject the null hypothesis
and we can say that count of bicycles rental is same for both working day and
non working day for the population.
```

Cheking for number of cycles rented is simliar for different seasons or not

H0: No.of cycles rented is similar in different seasons

H1: No.of cycles rented is different in different seasons

In [57]:
```
alpha =0.05
```

In [58]:
```
df["season"].value_counts()
```

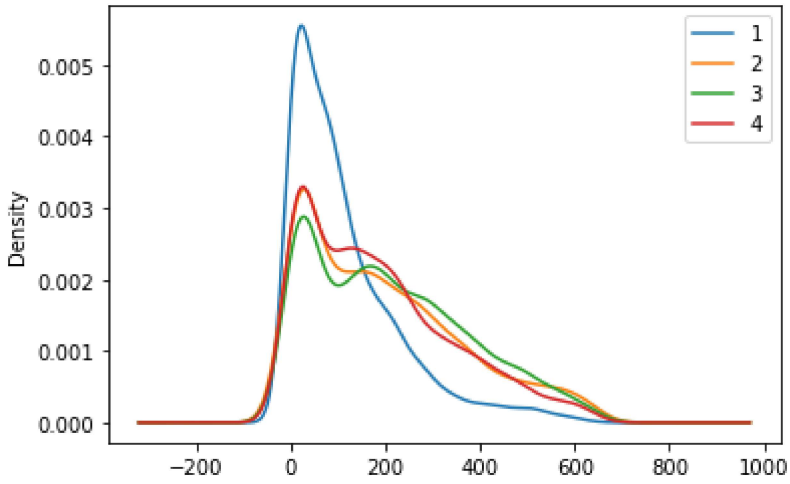
Out[58]:
```
1    2670
4    2665
2    2634
3    2617
Name: season, dtype: int64
```

In [59]:
```
#count as per seasons
df.groupby("season")["count"].sum()
```

Out[59]:
```
season
1    301163
2    515803
3    551274
4    491901
Name: count, dtype: int64
```

In [60]:
```
#kde plot for seasons
df.groupby('season')['count'].plot(kind='kde')
plt.legend()
```

Out[60]:
```
<matplotlib.legend.Legend at 0x275fa248910>
```



In [61]:
```
df.groupby("season")["count"].describe()
```

Out[61]:

season	count	mean	std	min	25%	50%	75%	max
1	2670.0	112.795131	116.884929	1.0	24.0	78.0	161.0	644.0
2	2634.0	195.824981	166.371838	1.0	45.0	165.0	299.0	647.0
3	2617.0	210.651127	164.245975	1.0	60.0	185.0	324.0	647.0
4	2665.0	184.578236	154.793646	1.0	49.0	154.0	277.0	647.0

From above the means are different with different standard deviations

In [62]:
```
a = df.loc[df['season'] == 1]['count'].sample(2000)
b = df.loc[df['season'] == 2]['count'].sample(2000)
c = df.loc[df['season'] == 3]['count'].sample(2000)
d = df.loc[df['season'] == 4]['count'].sample(2000)
stats.f_oneway(a, b, c, d)
```

Out[62]:
```
F_onewayResult(statistic=156.14533980049487, pvalue=2.3678438875369206e-98)
```

As the P value < alpha ,We can reject the null hypothesis

```
Conclusion: We can conclude that bicycle rental is different for different
seasons
```

Cheking for number of cycles rented is simliar for different weather conditions or not

H0: No.of cycles rented is similar in different weather conditions

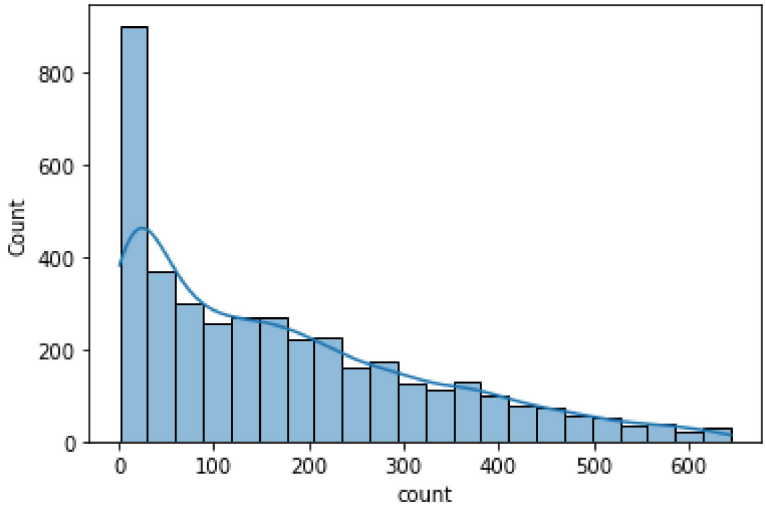
H1: No.of cycles rented is different in different weather conditions

In [63]:
```
alpha = 0.05
```

In [64]:
```
df["weather"].value_counts()
```

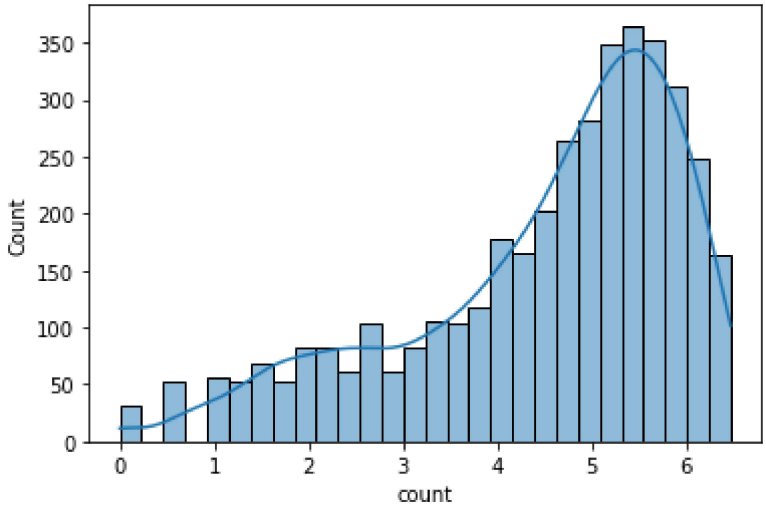
Out[64]:
```
1    6965
2    2770
3     850
4       1
Name: weather, dtype: int64
```

In [65]:
```
#cheking for normality
sns.histplot(df['count'].sample(4000),kde =True)
plt.show()
```



In [66]:
```
#using Log transformation as above distibuition is not normal
sns.histplot(np.log(df['count'].sample(4000)),kde =True)
```

Out[66]: <AxesSubplot:xlabel='count', ylabel='Count'>



In [67]:
```
df.groupby("weather")["count"].describe()
```

Out[67]:

weather	count	mean	std	min	25%	50%	75%	max
1	6965.0	187.329218	161.581066	1.0	45.0	153.0	287.0	647.0
2	2770.0	166.117690	146.992422	1.0	39.0	130.0	254.0	646.0
3	850.0	111.862353	121.233389	1.0	23.0	70.5	157.0	646.0
4	1.0	164.000000	NaN	164.0	164.0	164.0	164.0	164.0

In [68]:
```
x = df.loc[df['season'] == 1]['count'].sample(500)
y = df.loc[df['season'] == 2]['count'].sample(500)
z = df.loc[df['season'] == 3]['count'].sample(500)

stats.f_oneway(x,y,z)
```

Out[68]:
```
F_onewayResult(statistic=66.97232602385965, pvalue=1.3891819428544483e-28)
```

In [69]:
```
#skipping solving with log as it is not making much difference
```

As the P value < alpha ,We can reject the null hypothesis

Conclusion: We can conclude that bicycle rental is different for different weather condtions

Chi square test to decide if weather and season are dependent on each other

H0:Both weather and season are indepedent of each other

H1:Both weather and season are depedent on each other

In [70]:
```
df["weather"].value_counts()
```

Out[70]:
```
1    6965
2    2770
3     850
4       1
Name: weather, dtype: int64
```

In [71]:
```
df["season"].value_counts()
```

Out[71]:
```
1    2670
4    2665
2    2634
3    2617
Name: season, dtype: int64
```

In [72]:
```
table=pd.crosstab(df["weather"],df["season"])
table
```

Out[72]:

weather \ season	1	2	3	4
1	1744	1721	1843	1657
2	714	690	579	787
3	211	223	195	221
4	1	0	0	0

In [73]:
```
#since there is only 1 count of weather kind 4 ,we can remove it for easy calculc
df=df[df["weather"]!=4]
```

In [74]:
```
table_new=pd.crosstab(df["weather"],df["season"])
table_new
```

Out[74]:

weather \ season	1	2	3	4
1	1744	1721	1843	1657
2	714	690	579	787
3	211	223	195	221

In [75]:
```
stat,p_value,dof,expected= stats.chi2_contingency(table_new)
print(stat)
print(p_value)
print(dof)
print(expected)
```

```
44.20402460567892
6.734426550686341e-08
6
[[1756.21965045 1733.18941899 1722.00330657 1753.587624  ]
 [ 698.45347189  689.29428436  684.84553614  697.40670761]
 [ 214.32687766  211.51629665  210.1511573   214.0056684 ]]
```

since P value < alpha ,We reject the null hypothesis

```
Conclusion : We can conclude that weather and season are depedent on each
other
```

In [ ]: