Yulu Case Study

In [110]:
```
import numpy as np
import pandas as pd
import seaborn as sns
```

In [66]:
```
df=pd.read_csv("bike_sharing.csv")
```

In [55]:
```
df.head()
```

Out[55]:

	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.shape
```

Out[4]: (10886, 12)

In [ ]:
```
# There are 10886 rows and 12 columns
```

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 [ ]:
```
# only datetime is object field while other fields are either int or float
```

In [107]:
```
df.describe(include='object')
```

Out[107]:

	datetime
count	10886
unique	10886
top	2012-09-05 17:00:00
freq	1

In [109]:
```
df.describe()
```

Out[109]:

	season	holiday	workingday	weather	temp	atemp	humidity
nt	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.00000
an	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.88646
td	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.24503
in	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.00000
%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.00000
%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.00000
%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.00000
ax	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.00000

In [ ]:
```
# The 50% value and mean differs mainly in casual, registered and count fields
```

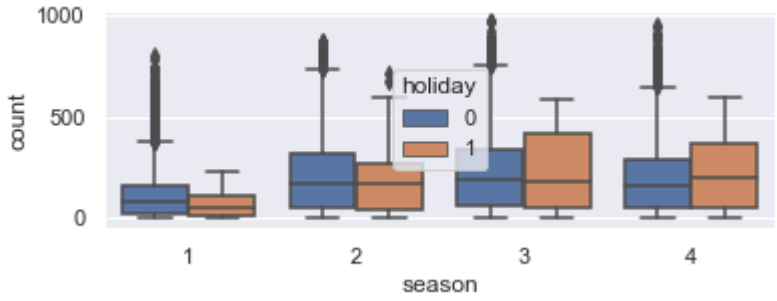
In [7]:
```
# finding unique values in each column
for i in df.columns:
    print(i,":", df[i].nunique())
```

```
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]:
```
# missing check
df.isna().sum()
# no missing values
```

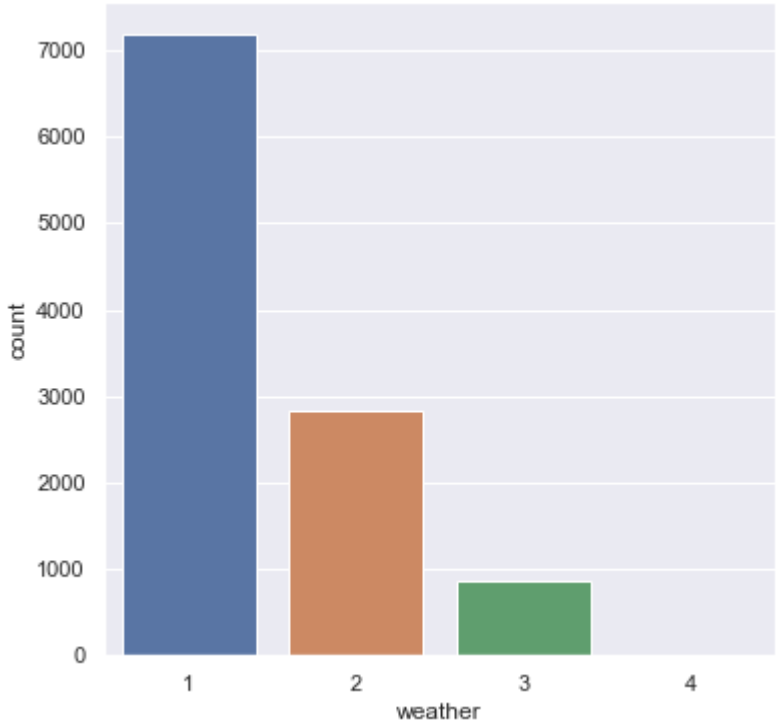
Out[8]:
```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

In [138]:
```
sns.boxplot(x=df['season'],y=df['count'],hue=df['holiday'])
sns.set(rc={"figure.figsize":(6,6)})
#clearly there are outliers in the dataset
```



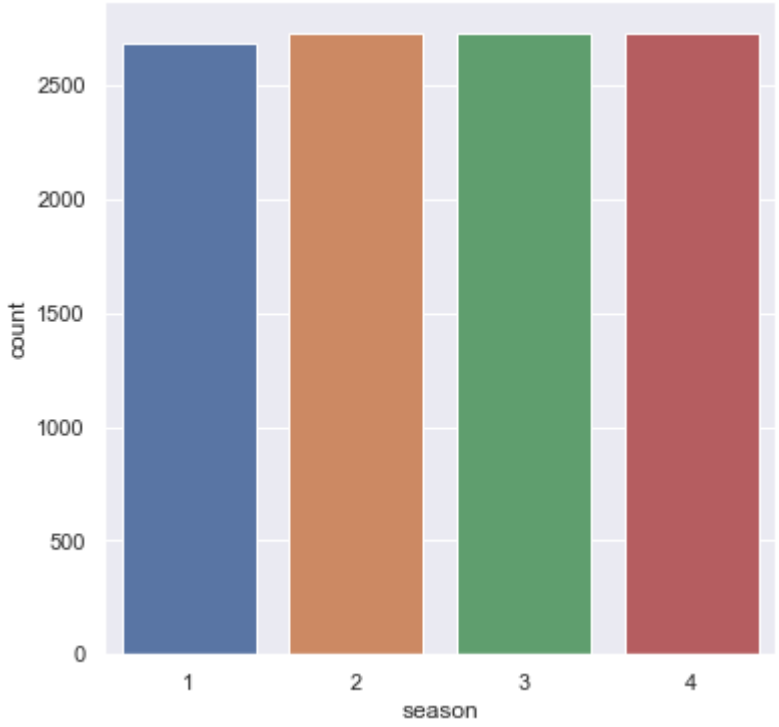
In [159]:
```
sns.countplot(data=df,x='weather')
```

Out[159]: <AxesSubplot:xlabel='weather', ylabel='count'>



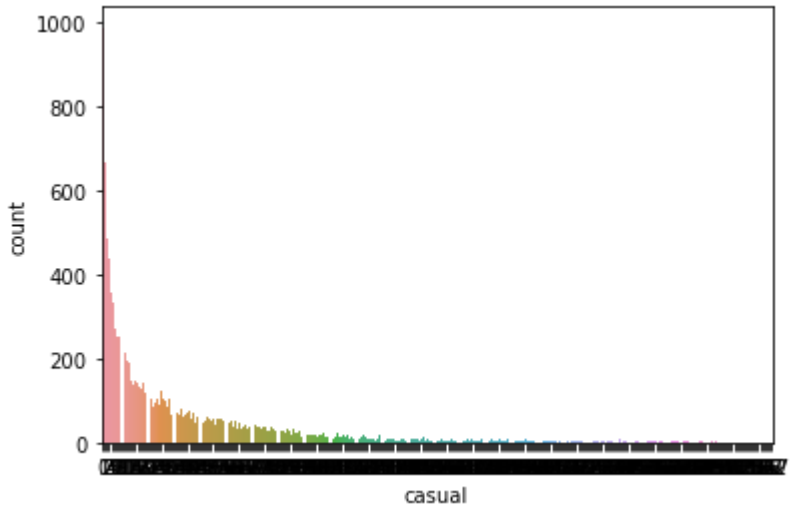
In [158]:
```
sns.countplot(data=df,x='season')
# count is somewhat similar for all the seasons
```

Out[158]: <AxesSubplot:xlabel='season', ylabel='count'>



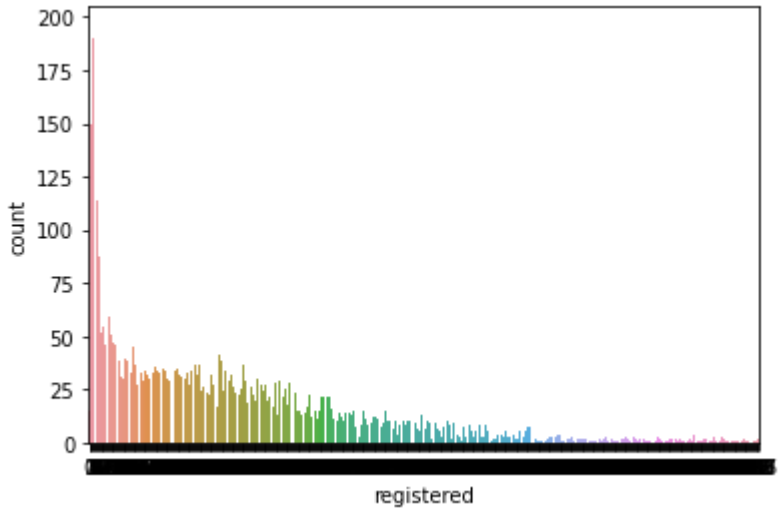
In [113]:
```
sns.countplot(data=df,x='casual')
# looks like exponentially decreasing graph
```

Out[113]: <AxesSubplot:xlabel='casual', ylabel='count'>



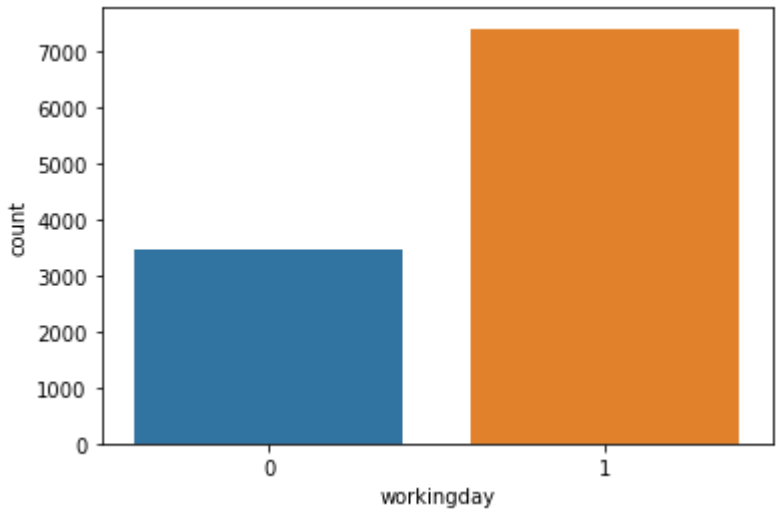
In [115]:
```
sns.countplot(data=df,x='registered')
# looks like exponentially decreasing graph, somewhat similar to casual
```

Out[115]: <AxesSubplot:xlabel='registered', ylabel='count'>



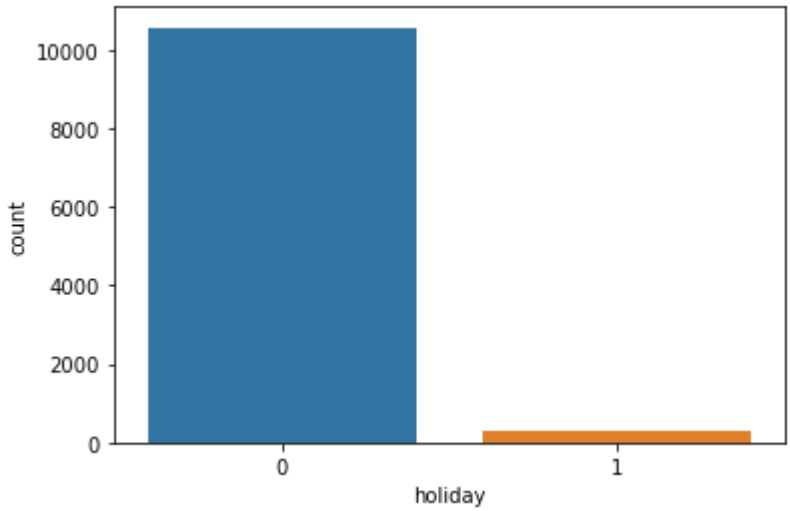
In [119]:
```
sns.countplot(data=df,x='workingday')
```

Out[119]: <AxesSubplot:xlabel='workingday', ylabel='count'>

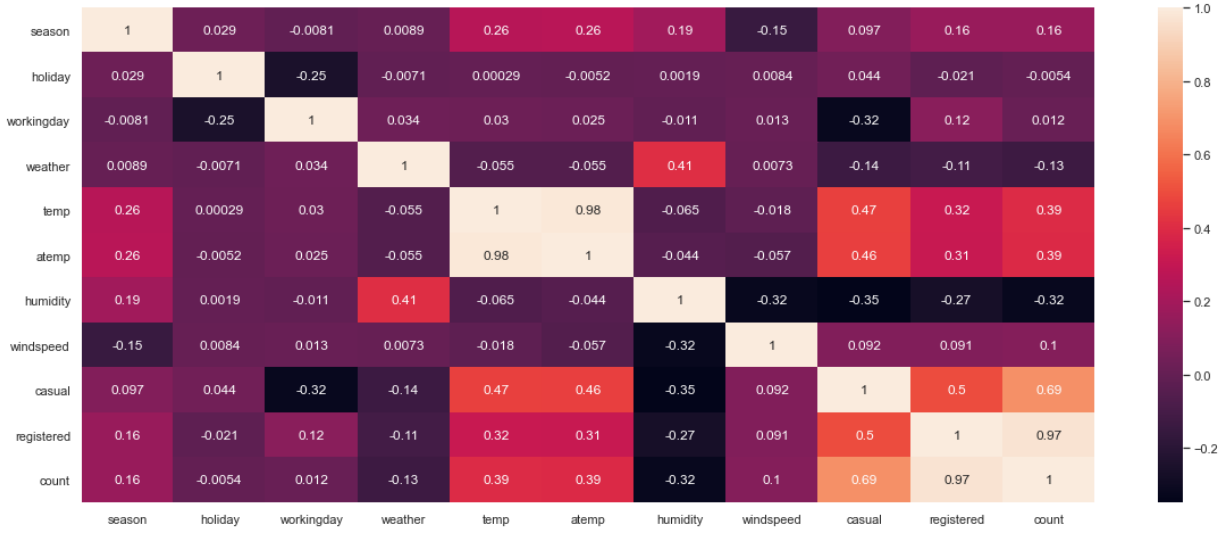


In [120]:
```
sns.countplot(data=df,x='holiday')
```

Out[120]: <AxesSubplot:xlabel='holiday', ylabel='count'>



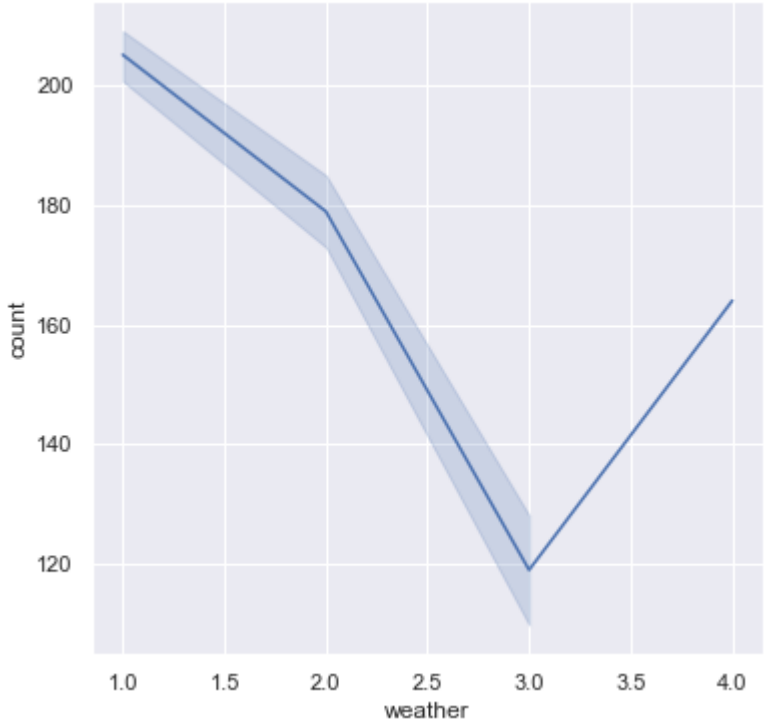
In [129]:
```
sns.heatmap(df.corr(),annot=True)
sns.set(rc={"figure.figsize":(6, 20)})
```



In [ ]:
```
# The above graph shows the correlation between various fields and how they are r
```

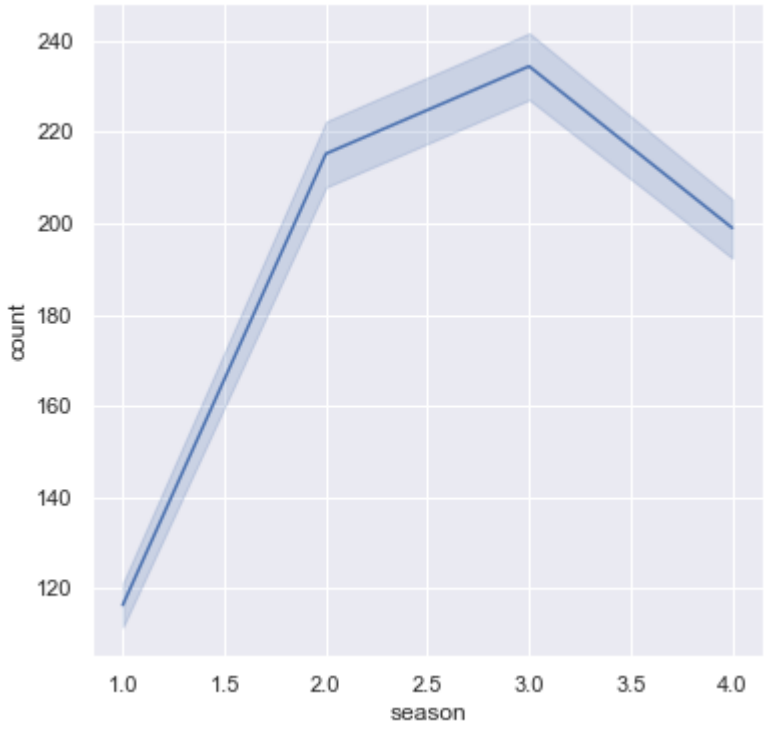
In [151]:
```
# Bi variate analysis
# Relation between workday and count
sns.lineplot(x='weather',y='count',data=df)
```

Out[151]: <AxesSubplot:xlabel='weather', ylabel='count'>



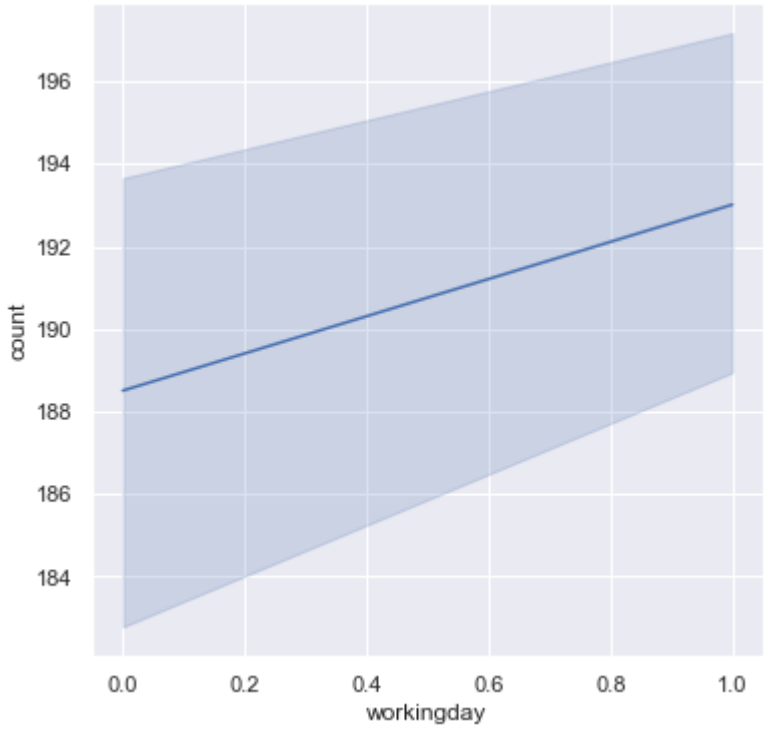
In [152]:
```
# Bi variate analysis
# Relation between season and count
sns.lineplot(x='season',y='count',data=df)
```

Out[152]: <AxesSubplot:xlabel='season', ylabel='count'>



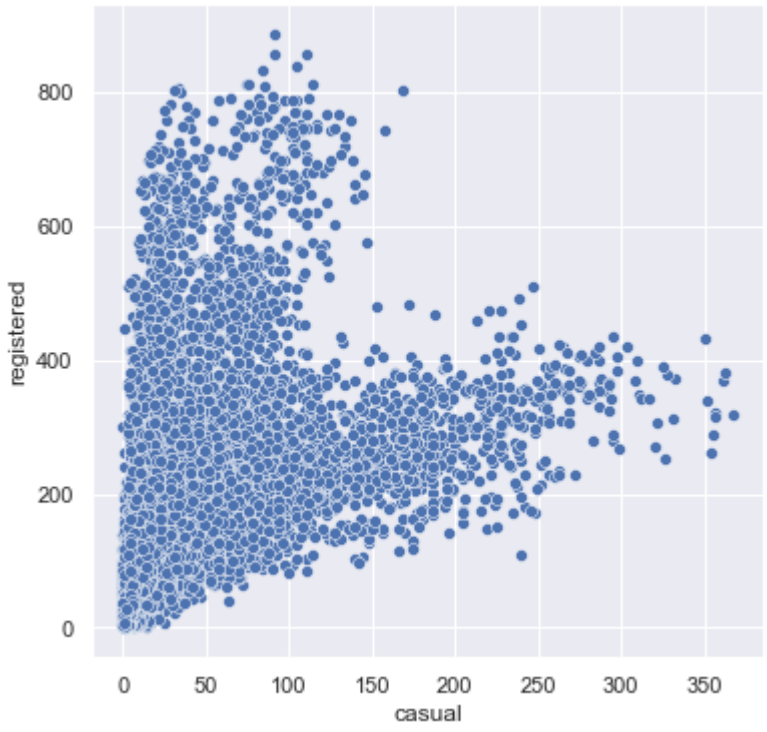
In [154]:
```
# Bi variate analysis
# Relation between season and count
sns.lineplot(x='workingday',y='count',data=df)
```

Out[154]: <AxesSubplot:xlabel='workingday', ylabel='count'>



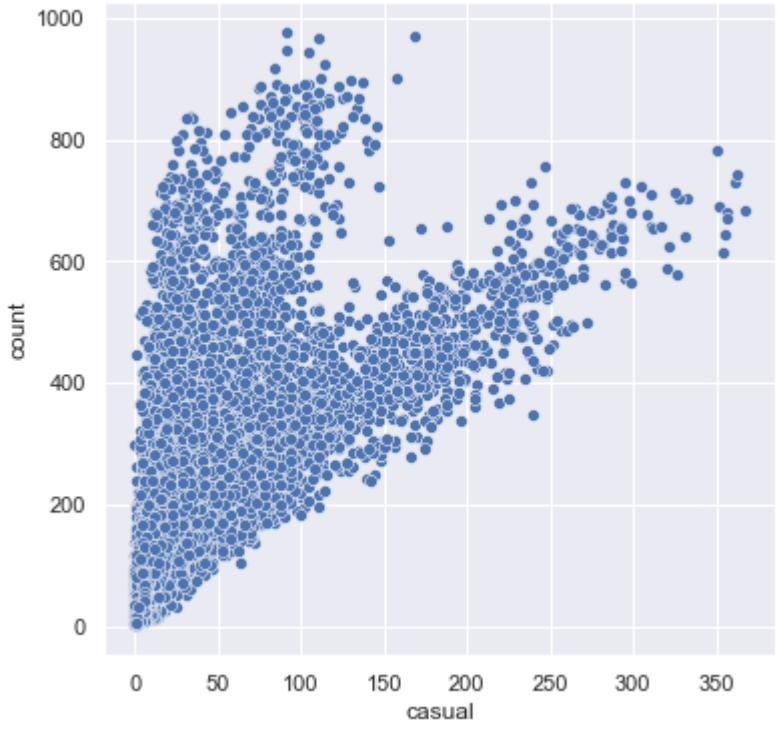
In [155]:
```
sns.scatterplot(x='casual',y='registered',data=df)
```

Out[155]: <AxesSubplot:xlabel='casual', ylabel='registered'>



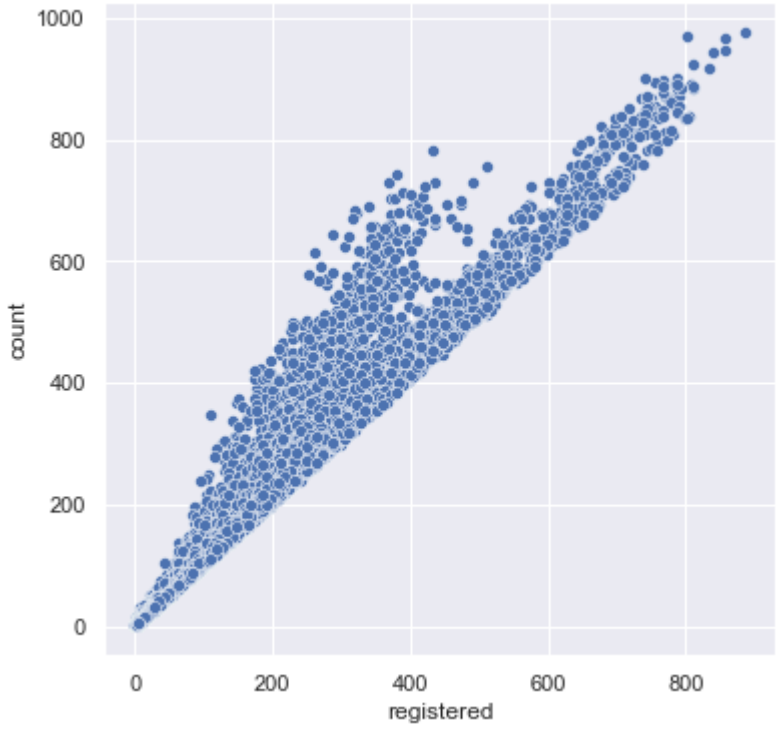
In [156]:
```
sns.scatterplot(x='casual',y='count',data=df)
```

Out[156]: <AxesSubplot:xlabel='casual', ylabel='count'>



In [157]:
```
sns.scatterplot(x='registered',y='count',data=df)
```

Out[157]: <AxesSubplot:xlabel='registered', ylabel='count'>

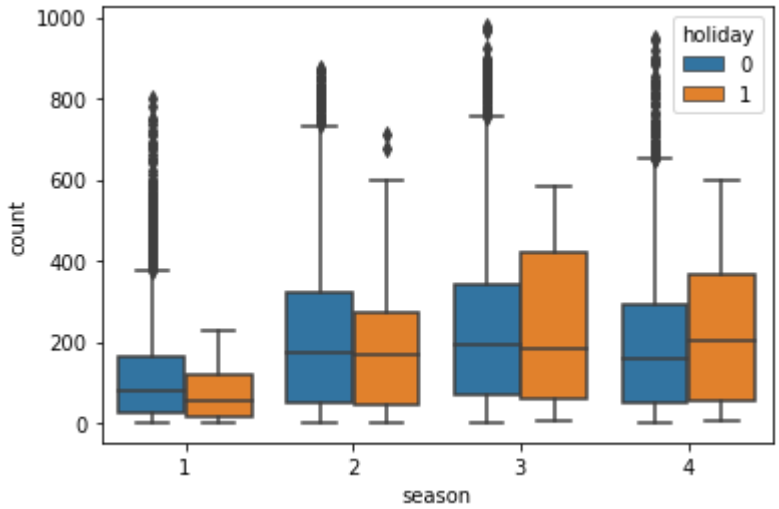


In [161]:
```
# conclusions from EDA
# 1. as the registered users increases the count of cycles increases
# 2. 1 to 3 means spring to fall season the count of cycles increases and then de
# 3. count decreaes as the weather changes from clear weather to thunderstorm and
# 4. The count of casual and registered is exponentially decresing
# 5. count of seasons is almost similar to each other
# 6. there are outliers in the dataset
```

In [ ]:
```
```

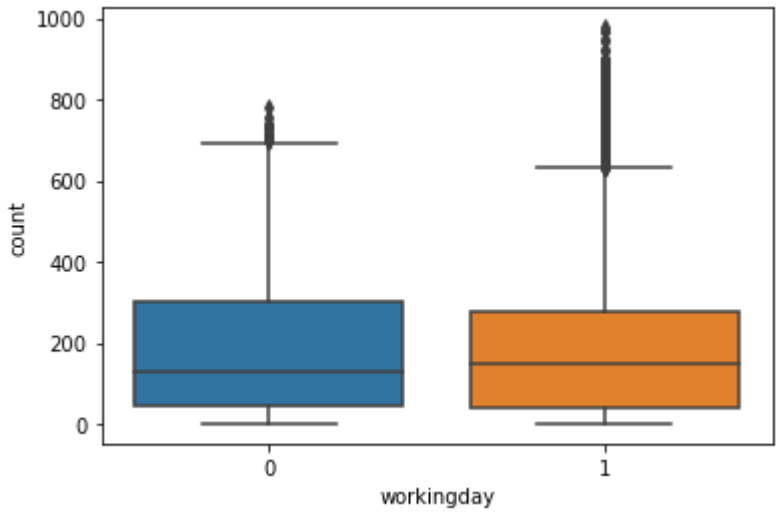
In [9]:
```
sns.boxplot(x=df['season'],y=df['count'],hue=df['holiday'])
```

Out[9]: <AxesSubplot:xlabel='season', ylabel='count'>



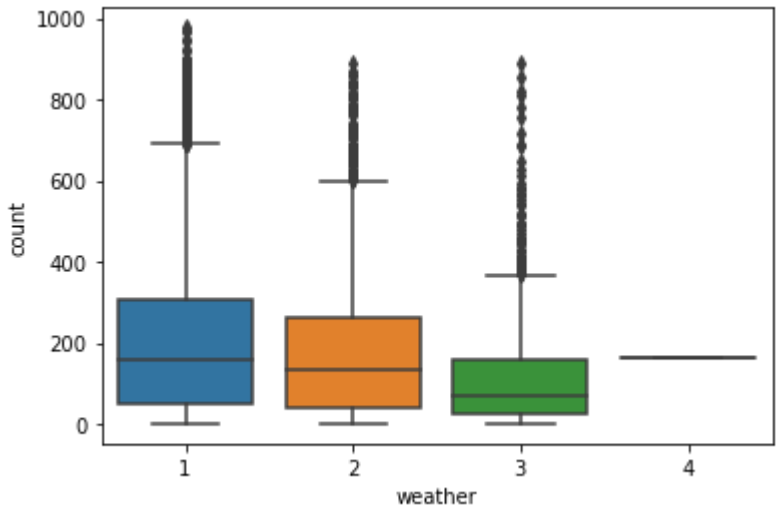
In [10]:
```
sns.boxplot(x=df['workingday'],y=df['count'])
```

Out[10]: <AxesSubplot:xlabel='workingday', ylabel='count'>



In [11]:
```
sns.boxplot(x=df['weather'],y=df['count'])
```

Out[11]:
```
<AxesSubplot:xlabel='weather', ylabel='count'>
```



In [ ]:
```

```

In [13]:
```
# 2- Sample T-Test to check if Working Day has an effect on the number of electri
# ANNOVA to check if No. of cycles rented is similar or different in different 1.
# Chi-square test to check if Weather is dependent on the season (10 points)
```

```
Two sample T test to check the effect of working day
```

In [ ]:
```
# assumptions of T test
#1. Independence: The observations in one sample are independent of the observati

#2. Normality: Both samples are approximately normally distributed.

#3. Homogeneity of Variances: Both samples have approximately the same variance.

#4. Random Sampling: Both samples were obtained using a random sampling method.
```

In [14]:
```
# experimenting 2 sample t test for working days.
df['workingday'].value_counts()
# so there are 7412 when its a working day, 3474 for a weekend or holiday.
```

Out[14]:
```
1    7412
0    3474
Name: workingday, dtype: int64
```

In [146]:
```
workday_count_array=np.array(df[df['workingday']==1]['count'])
holiday_count_array=np.array(df[df['workingday']==0]['count'])
```

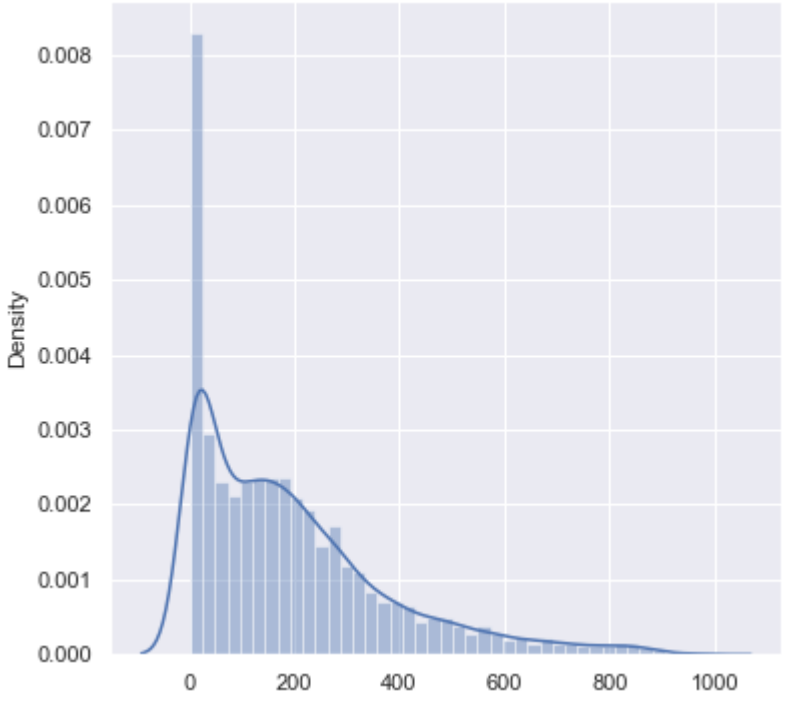
In [162]:
```
np.std(workday_count_array),np.std(holiday_count_array)
```

Out[162]:
```
(184.501211667422, 173.69901006897658)
```

In [149]:
```
sns.distplot(x=workday_count_array,kde=True)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

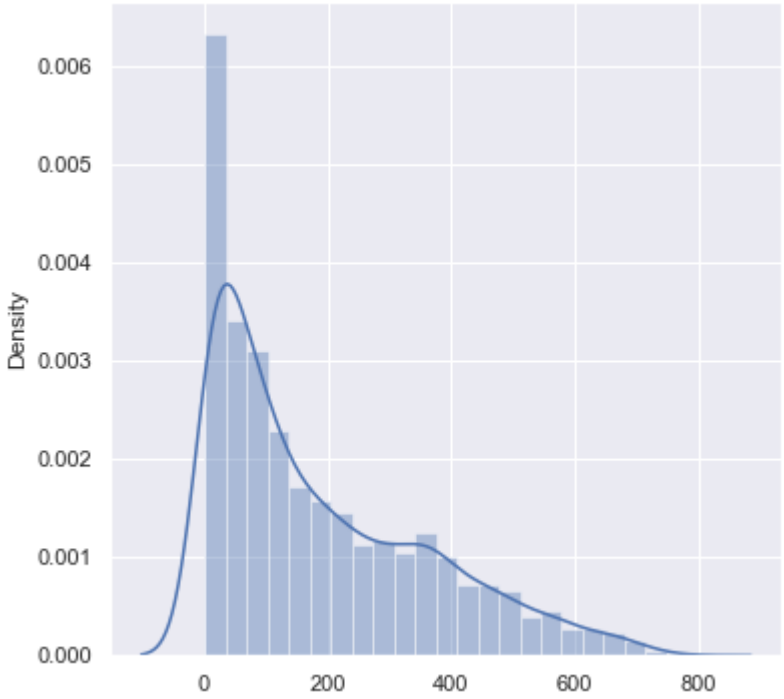
Out[149]:
```
<AxesSubplot:ylabel='Density'>
```



In [150]:
```
sns.distplot(x=holiday_count_array,kde=True)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[150]: <AxesSubplot:ylabel='Density'>



In [18]:
```
# Null hypothesis H0: working day and holiday have same effect on count of riders
# Alternate hypothesis Ha: working day and holiday does not have same effect on c
```

In [19]:
```
# calculating the P value
# considering 95% confidence interval
import scipy.stats as stats
stats.ttest_ind(workday_count_array,holiday_count_array)
# as the P value is very high we can say that there is eqaul effect of workday ar
```

Out[19]:
```
Ttest_indResult(statistic=1.2096277376026694, pvalue=0.22644804226361348)
```

```
Here p value is greater than significance level: 0.22>0.05
Conclusion based on the p-value  :
We fail to reject null hypothesis and conclude that there is no significant
difference comparing to holiday vs working day
```

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

In [ ]:
```
Assumption 1: Both variables are categorical.
Assumption 2: All observations are independent.
Assumption 3: Cells in the contingency table are mutually exclusive.
Assumption 4:Expected value of cells should be 5 or greater in at least 80% of ce
Considering 95% confidence interval
```

In [62]:
```
df['weather'].value_counts()
```

Out[62]:
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

In [63]:
```
df['season'].value_counts()
```

Out[63]:
```
4    2734
3    2733
2    2733
1    2686
Name: season, dtype: int64
```

In [23]:
```
observed=pd.crosstab(df['weather'], df['season'],margins=True)
```

Out[23]:

	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 [163]:
```
from scipy.stats import chi2_contingency
test_statistic,p_value,df,frq=chi2_contingency(observed)
test_statistic,p_value
```

Out[163]:
```
(49.15865559689363, 3.1185273325126814e-05)
```

```
 1.AS the p value is less than significance level
 2.p_value<0.05
 3.Reject the null hypothesis
 Conclusion based on the p-value  : weather is dependent on season
```

ANNOVA to check if No. of cycles rented is similar or different in different 1. weather 2. season

```
assumption 1: Normality
assumption 2: independent
assumption 3: Equal Variances
Considering 95% confidence interval
```

In [ ]:
```
# To check if the no of cycles rented is similar in different weather
```

In [67]:
```
df['weather'].value_counts()
```

Out[67]:
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

In [ ]:
```
# H0 : In all the weathers the count of cycles remains same
# HA : As the weather changes there is change in count of cycles
```

In [71]:
```
weather_1=np.array(df[df['weather']==1]['count'])
weather_2=np.array(df[df['weather']==2]['count'])
weather_3=np.array(df[df['weather']==3]['count'])
weather_4=np.array(df[df['weather']==4]['count'])
```

In [74]:
```
from scipy.stats import f_oneway
```

In [75]:
```
f_stat,p_value=f_oneway(weather_1,weather_2,weather_3,weather_4)
```

In [164]:
```
f_stat,p_value
```

Out[164]:
```
(236.94671081032106, 3.1185273325126814e-05)
```

In [ ]:
```
# As the P value is very small that is less than significant level(0.05) , we rej
# Conclusion based on the p-value  : weather has satistically significant affect
```

In [ ]:
```

```

In [ ]:
```
# To check if the no of cycles rented is similar in different season
```

In [99]:
```
season_1=np.array(df[df['season']==1]['count'])
season_2=np.array(df[df['season']==2]['count'])
season_3=np.array(df[df['season']==3]['count'])
season_4=np.array(df[df['season']==4]['count'])
```

In [ ]:
```
# H0: as the season changes there is no statistically significant difference in c
# HA: as the season changes there is statistically significant difference in cour
```

In [100]:
```
df['season'].value_counts()
```

Out[100]:
```
4    2734
3    2733
2    2733
1    2686
Name: season, dtype: int64
```

In [101]:
```
f_stat,p_value=f_oneway(season_1,season_2,season_3,season_4)
```

In [102]:
```
p_value
```

Out[102]:
```
6.164843386499654e-149
```

In [104]:
```
#AS the p_value is less than significance level(0.05) , we reject the null hypoth
#Conclusion based on the p-value  : The season has satistically significant affec
```

In [ ]:
```
# conclusions from EDA
# 1. as the registered users increases the count of cycles increases
# 2. 1 to 3 means spring to fall season the count of cycles increases and then de
# 3. count decreaes as the weather changes from clear weather to thunderstorm and
# 4. The count of casual and registered is exponentially decresing
# 5. count of seasons is almost similar to each other
# 6. there are outliers in the dataset
```

In [165]:
```
# conclusions from hypothesis testing
# T test result
#1.We fail to reject null hypothesis and conclude that there is no significant di

# Chi square test
# 1.Reject the null hypothesis and conlcude that weather is dependent on season

# Annova test
# 1. Reject the null hypothesis and conclude that weather has satistically signif
# 2. Reject the null hypothesis and conclude that  season has satistically signif
```

In [ ]:
```

```