Problem statement:

Yulu is India’s leading micro-mobility service provider, which offers unique vehicles for the daily commute.Now our role is identify which variables are significant in predicting the demand for shared electric cycles in the Indian market and how those variables describe the electric cycle demands.

Business Insights:

- Working day has clear effect on electric cycles rented
- The impact of working days is more when there is work .
- The number of cycles rented are dependent on the weather and different seasons
- The average number of cycles rented is more when the weather is clear
- The average number of cycles rented is more when it is fall
- weather is depenent on the seasons

Recommendations:

- Company should give coupons to the working professiols
- Company should make make campaigns near working offices
- Comapny should keep stock of vehicles more in fall and spring seasons
- Company should keep stock of vehicles on the clear days
- Comapny should keep track of good weather analysis on daily basis
- Company should come up with a way to make more cycles subscribed in holidays too.
- They can give more coupons on holidays

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
data=pd.read_csv('C:/Users/Ajith/Desktop/scaler case studiess/bike_sharing.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
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]:

```
data.shape
```

Out[4]:

```
(10886, 12)
```

In [5]:

```
data.info()
```

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

In [6]:

```
data.dtypes
```

Out[6]:

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

In [7]:

```
data.isna().sum()
```

Out[7]:

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

In [8]:

```
data.describe()
```

Out[8]:

	season	holiday	workingday	weather	temp	atemp	
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	108
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	1

In [9]:

```
data.describe(include='all')
```

Out[9]:

	datetime	season	holiday	workingday	weather	temp	
count	10886	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.(
unique	10886	NaN	NaN	NaN	NaN	NaN	
top	2011-03-19 21:00:00	NaN	NaN	NaN	NaN	NaN	
freq	1	NaN	NaN	NaN	NaN	NaN	
mean	NaN	2.506614	0.028569	0.680875	1.418427	20.23086	23.(
std	NaN	1.116174	0.166599	0.466159	0.633839	7.79159	8.4
min	NaN	1.000000	0.000000	0.000000	1.000000	0.82000	0.7
25%	NaN	2.000000	0.000000	0.000000	1.000000	13.94000	16.(
50%	NaN	3.000000	0.000000	1.000000	1.000000	20.50000	24.2
75%	NaN	4.000000	0.000000	1.000000	2.000000	26.24000	31.(
max	NaN	4.000000	1.000000	1.000000	4.000000	41.00000	45.4

Type *Markdown* and LaTeX: α^2

In [10]:

```
data2=data.copy()
```

In [11]:

```
data2.dtypes
```

Out[11]:

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

In [ ]:

```
```

Unique elements of each column

In [12]:

```
for i in data.columns:
    print(i,':',data[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
```

Conversion of Categorical variables to category type

In [13]:

```
data['season']=data['season'].astype('category')
data['holiday']=data['holiday'].astype('category')
data['workingday']=data['workingday'].astype('category')
```

In [14]:

```
data.dtypes
```

Out[14]:

```
datetime        object
season        category
holiday       category
workingday    category
weather          int64
temp           float64
atemp          float64
humidity         int64
windspeed      float64
casual           int64
registered       int64
count            int64
dtype: object
```

Non-Graphical Analysis

Checking the value_counts of each categorical variable

In [15]:

```
d=data.select_dtypes(include=['category'])
a=list(d.columns)
a
```

Out[15]:

```
['season', 'holiday', 'workingday']
```

In [16]:

```
for i in a:
    print(i)
    print(data[i].value_counts())
    print('\n')
```

```
season
4    2734
3    2733
2    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
```

In [17]:

```
data.groupby('season').agg({'count':['count','mean']})
```

Out[17]:

	count	
	count	mean
season		
1	2686	116.343261
2	2733	215.251372
3	2733	234.417124
4	2734	198.988296

Visual Analysis

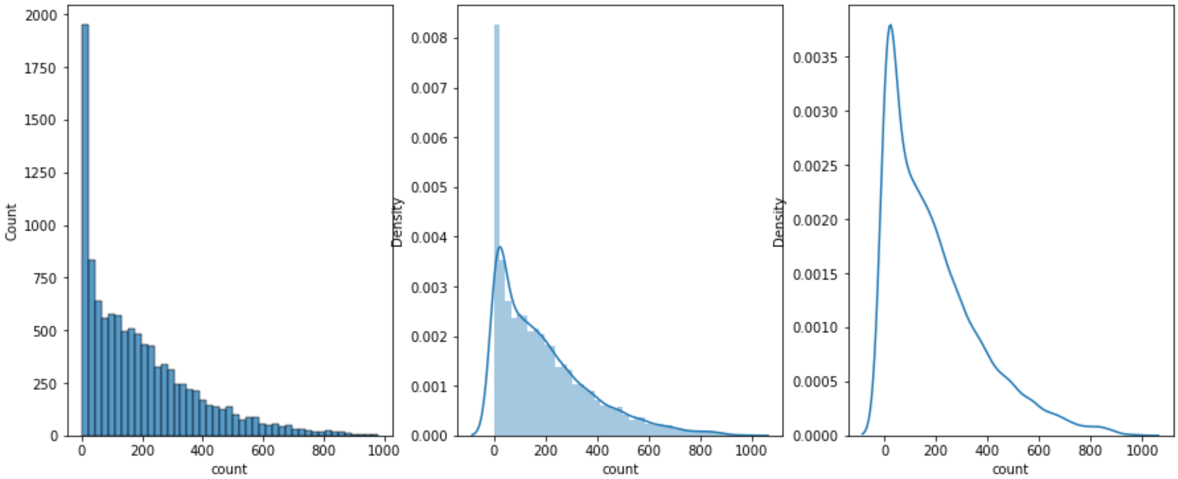
Univariate Analysis

In [18]:

```
# count
plt.figure(figsize=(15,6))
plt.subplot(131)
sns.histplot(data=data,x='count')
plt.subplot(132)
sns.distplot(data['count'])
plt.subplot(133)
sns.kdeplot(data['count'])
```

Out[18]:

```
<AxesSubplot:xlabel='count', ylabel='Density'>
```



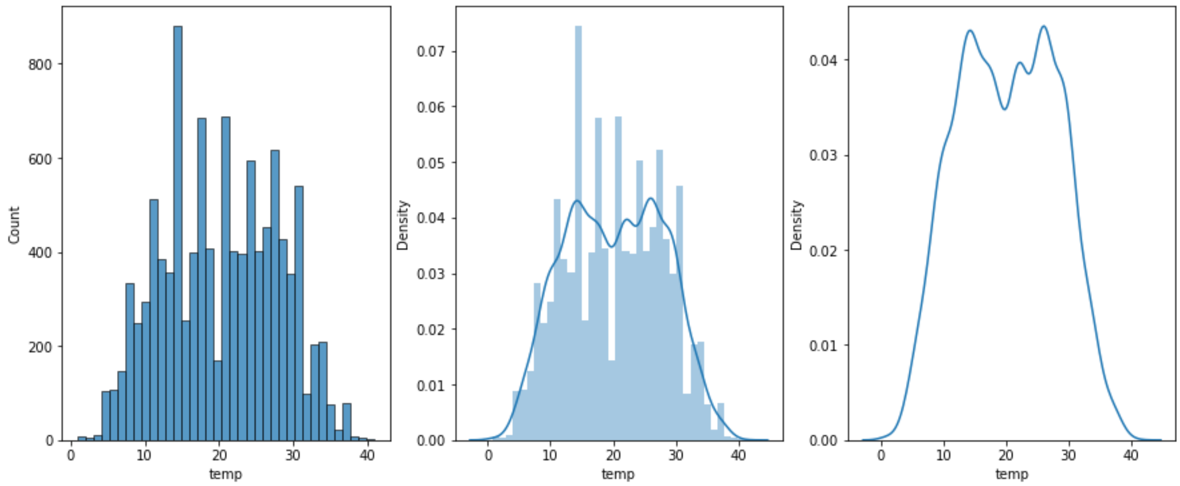
- It is following skewed distribution

In [19]:

```
# temp
plt.figure(figsize=(15,6))
plt.subplot(131)
sns.histplot(data=data,x='temp')
plt.subplot(132)
sns.distplot(data['temp'])
plt.subplot(133)
sns.kdeplot(data['temp'])
```

Out[19]:

```
<AxesSubplot:xlabel='temp', ylabel='Density'>
```



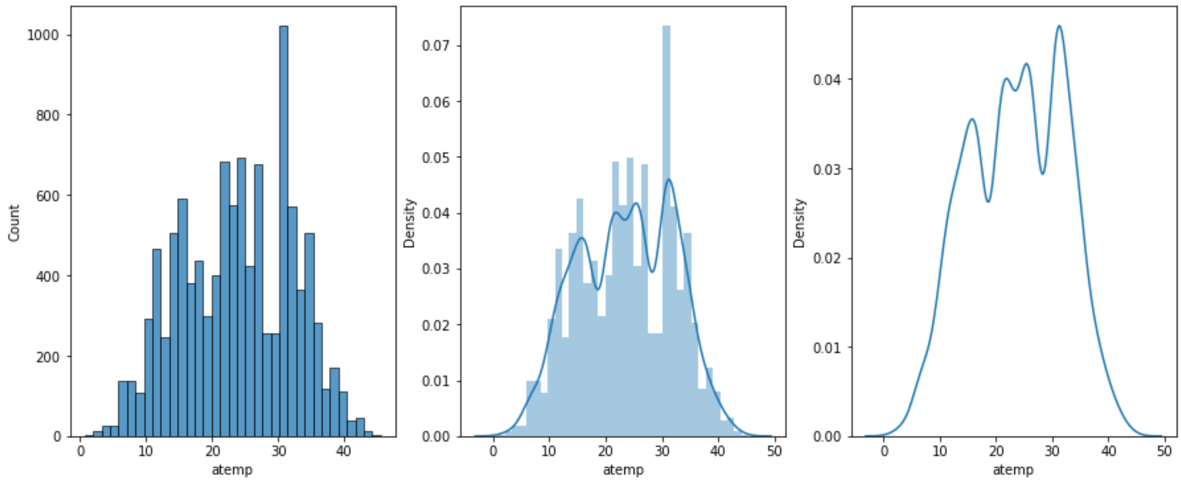
- It is kind of following normal distribution

In [20]:

```
#atemp
plt.figure(figsize=(15,6))
plt.subplot(131)
sns.histplot(data=data,x='atemp')
plt.subplot(132)
sns.distplot(data['atemp'])
plt.subplot(133)
sns.kdeplot(data['atemp'])
```

Out[20]:

```
<AxesSubplot:xlabel='atemp', ylabel='Density'>
```



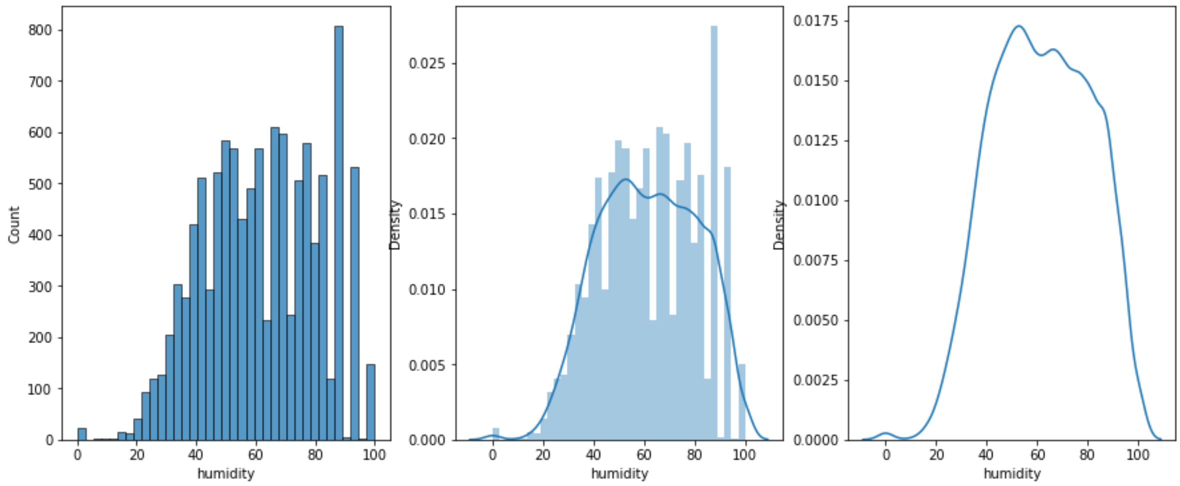
- It is also following Normal Distribution

In [21]:

```
#humidity
plt.figure(figsize=(15,6))
plt.subplot(131)
sns.histplot(data=data,x='humidity')
plt.subplot(132)
sns.distplot(data['humidity'])
plt.subplot(133)
sns.kdeplot(data['humidity'])
```

Out[21]:

```
<AxesSubplot:xlabel='humidity', ylabel='Density'>
```



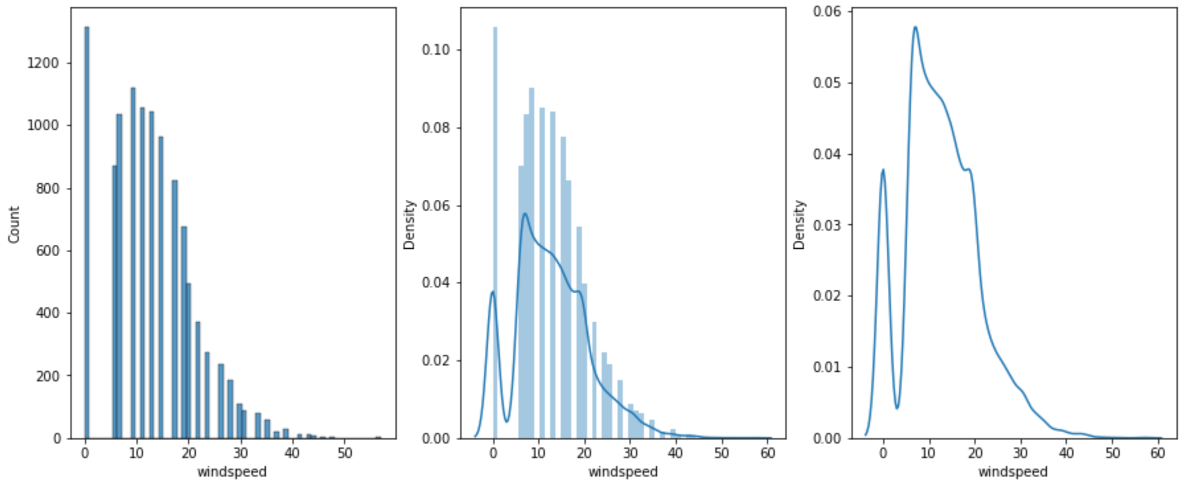
- It is following Normal Distribution

In [22]:

```
#windspeed
plt.figure(figsize=(15,6))
plt.subplot(131)
sns.histplot(data=data,x='windspeed')
plt.subplot(132)
sns.distplot(data['windspeed'])
plt.subplot(133)
sns.kdeplot(data['windspeed'])
```

Out[22]:

```
<AxesSubplot:xlabel='windspeed', ylabel='Density'>
```



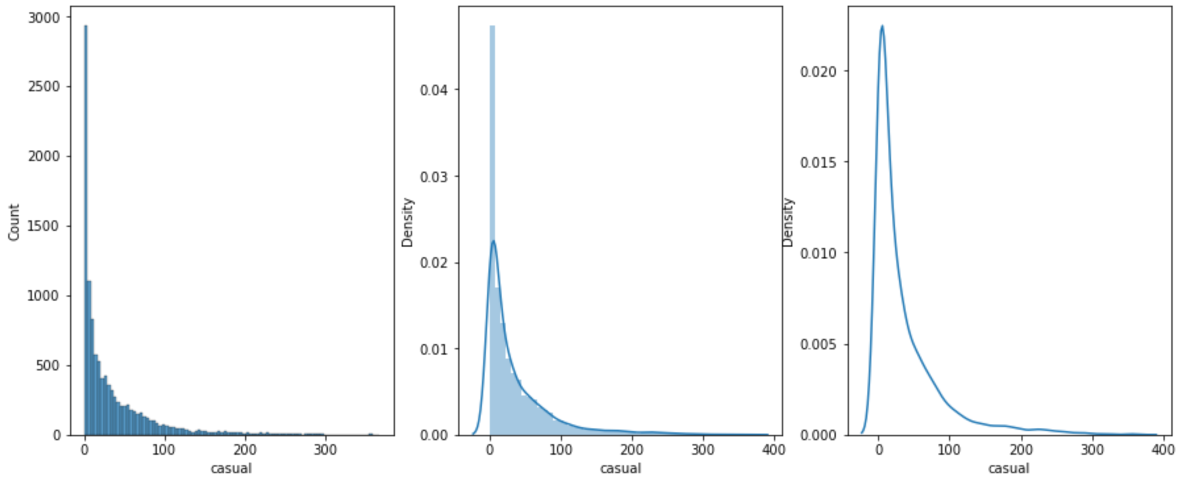
- It is following a skewed distribution

In [23]:

```
#casual
plt.figure(figsize=(15,6))
plt.subplot(131)
sns.histplot(data=data,x='casual')
plt.subplot(132)
sns.distplot(data['casual'])
plt.subplot(133)
sns.kdeplot(data['casual'])
```

Out[23]:

```
<AxesSubplot:xlabel='casual', ylabel='Density'>
```

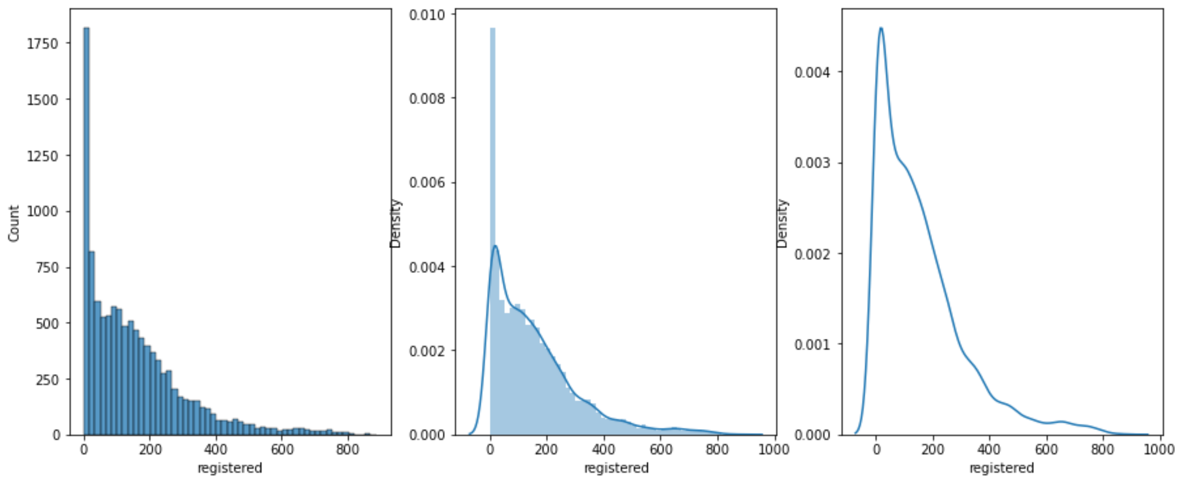


In [24]:

```
#registered
plt.figure(figsize=(15,6))
plt.subplot(131)
sns.histplot(data=data,x='registered')
plt.subplot(132)
sns.distplot(data['registered'])
plt.subplot(133)
sns.kdeplot(data['registered'])
```

Out[24]:

```
<AxesSubplot:xlabel='registered', ylabel='Density'>
```



- its distribution is right skewed

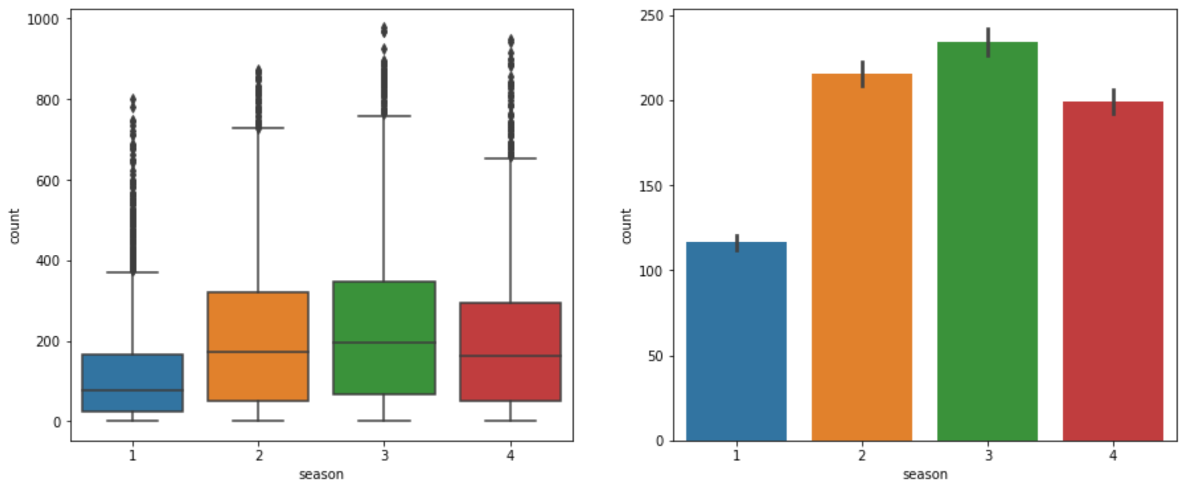
Bivariate analysis

In [25]:

```
#relation between season and count
plt.figure(figsize=(15,6))
plt.subplot(121)
sns.boxplot(x='season',y='count',data=data)
plt.subplot(122)
sns.barplot(x='season',y='count',data=data)
```

Out[25]:

```
<AxesSubplot:xlabel='season', ylabel='count'>
```



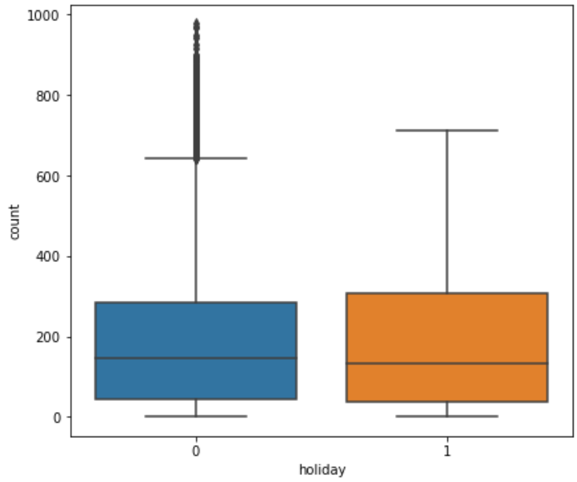
In [ ]:

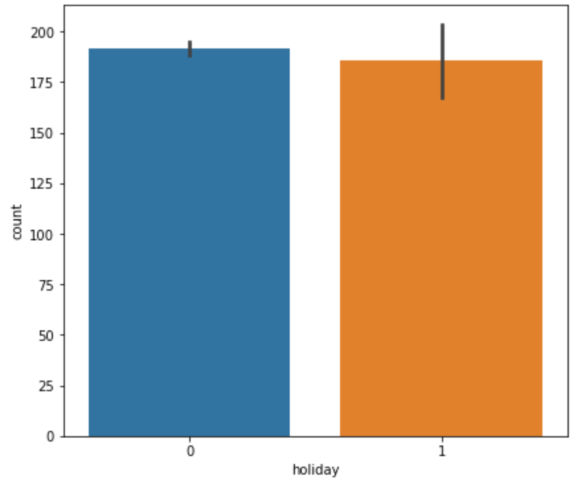
In [26]:

```
# relation between holiday and count
plt.figure(figsize=(15,6))
plt.subplot(121)
sns.boxplot(x='holiday',y='count',data=data)
plt.subplot(122)
sns.barplot(x='holiday',y='count',data=data)
```

Out[26]:

<AxesSubplot:xlabel='holiday', ylabel='count'>





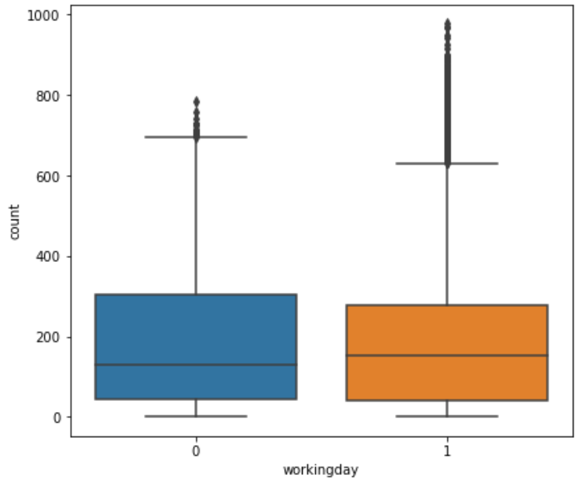
- There are outliers in case of no holidays but these are not outliers
- The holidays approx have equal no of values for each category

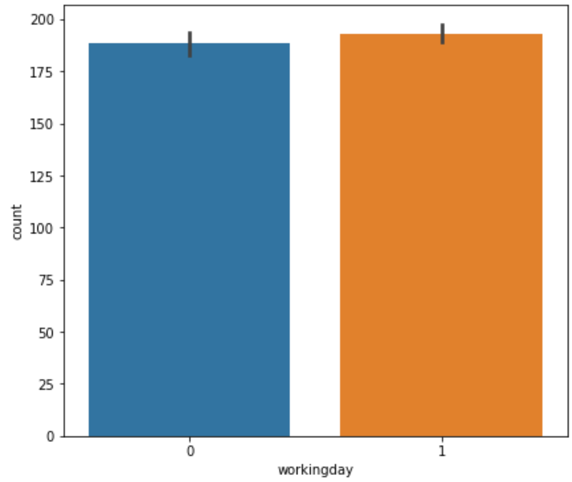
In [27]:

```
# relation between workingday and count
plt.figure(figsize=(15,6))
plt.subplot(121)
sns.boxplot(x='workingday',y='count',data=data)
plt.subplot(122)
sns.barplot(x='workingday',y='count',data=data)
```

Out[27]:

<AxesSubplot:xlabel='workingday', ylabel='count'>





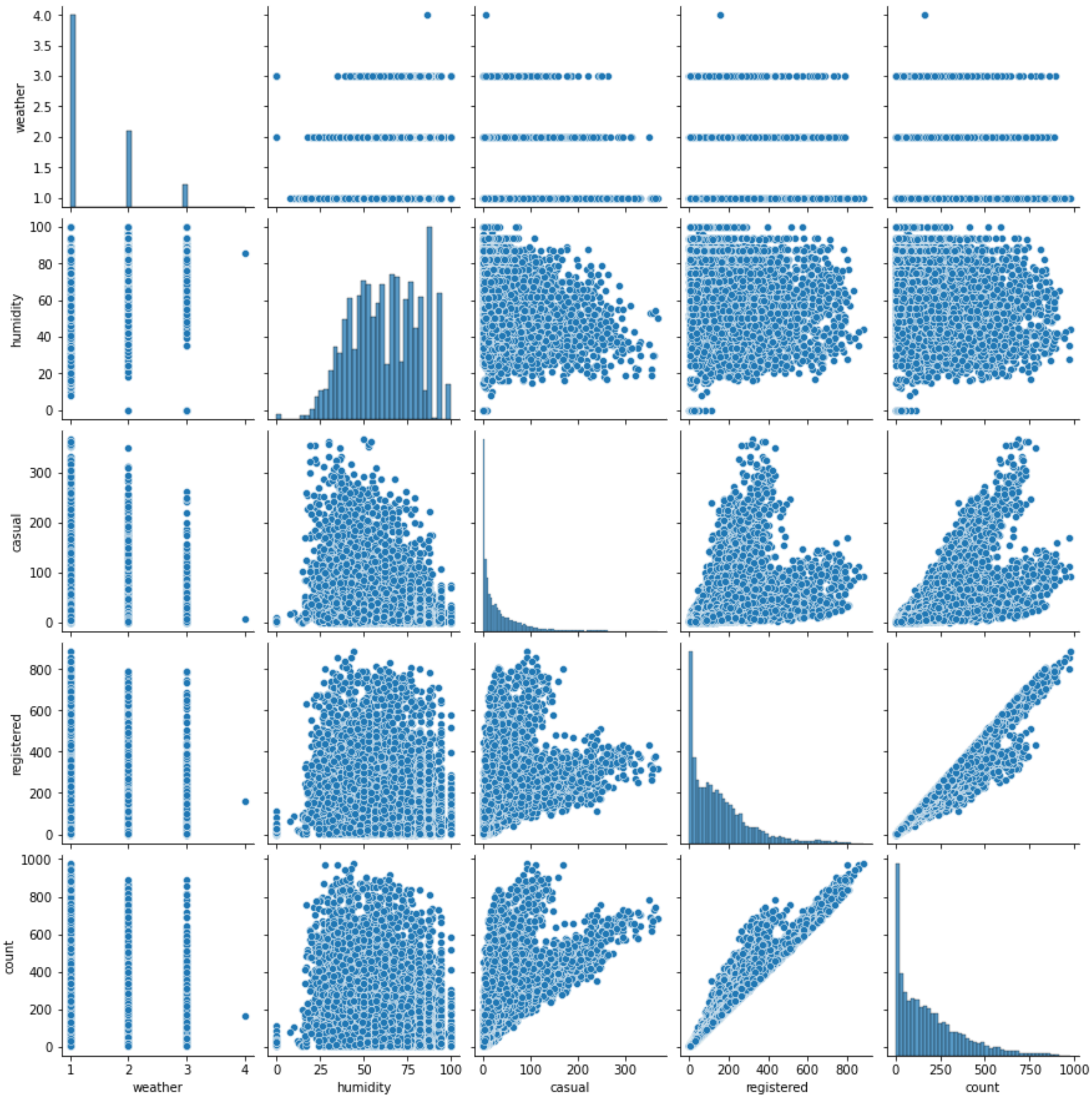
In [ ]:

In [28]:

```
# relation between
sns.pairplot(data.select_dtypes(include='int64'))
```

Out[28]:

<seaborn.axisgrid.PairGrid at 0x1a6e5f70d90>



In [29]:

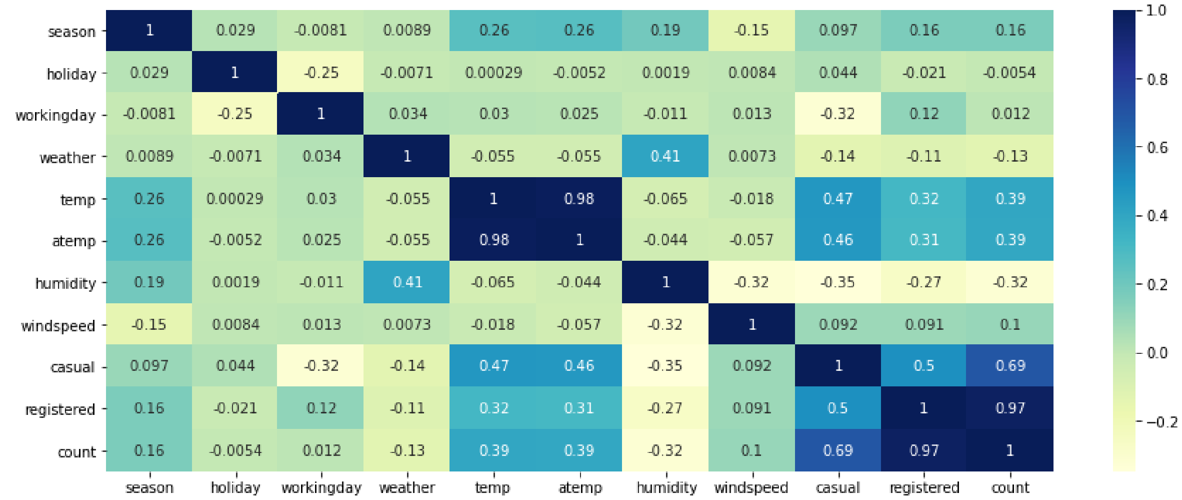
```
x=data2.corr()
x
```

Out[29]:

	season	holiday	workingday	weather	temp	atemp	humidity	winds
season	1.000000	0.029368	-0.008126	0.008879	0.258689	0.264744	0.190610	-0.1
holiday	0.029368	1.000000	-0.250491	-0.007074	0.000295	-0.005215	0.001929	0.0
workingday	-0.008126	-0.250491	1.000000	0.033772	0.029966	0.024660	-0.010880	0.0
weather	0.008879	-0.007074	0.033772	1.000000	-0.055035	-0.055376	0.406244	0.0
temp	0.258689	0.000295	0.029966	-0.055035	1.000000	0.984948	-0.064949	-0.0
atemp	0.264744	-0.005215	0.024660	-0.055376	0.984948	1.000000	-0.043536	-0.0
humidity	0.190610	0.001929	-0.010880	0.406244	-0.064949	-0.043536	1.000000	-0.3
windspeed	-0.147121	0.008409	0.013373	0.007261	-0.017852	-0.057473	-0.318607	1.0
casual	0.096758	0.043799	-0.319111	-0.135918	0.467097	0.462067	-0.348187	0.0
registered	0.164011	-0.020956	0.119460	-0.109340	0.318571	0.314635	-0.265458	0.0
count	0.163439	-0.005393	0.011594	-0.128655	0.394454	0.389784	-0.317371	0.1

In [30]:

```
plt.figure(figsize=(15,6))
sns.heatmap(data2.corr(), cmap="YlGnBu", annot=True)
plt.show()
```



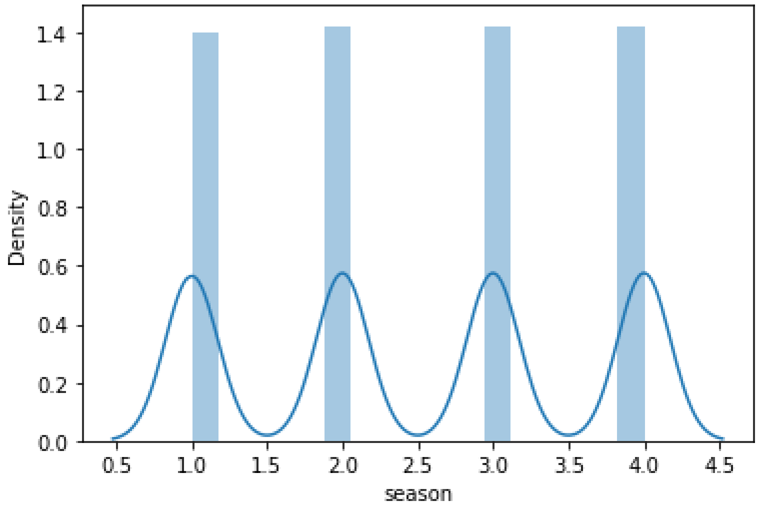
- From heatmap we can see that count is dependent on registerd and casual features
- it is dependent on temp and atemp but not as much as registered and casual
- Temp and atemp are highly correlated to eachother

In [31]:

```
sns.distplot(data['season'])
```

Out[31]:

<AxesSubplot:xlabel='season', ylabel='Density'>

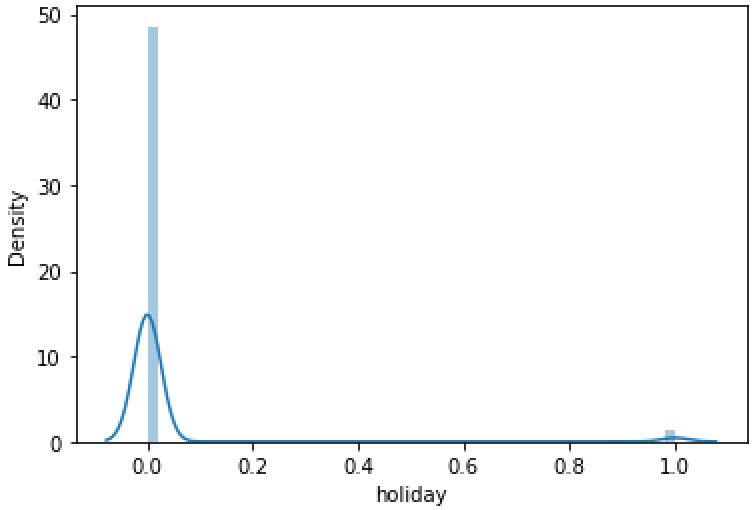


In [32]:

```
sns.distplot(data['holiday'])
```

Out[32]:

<AxesSubplot:xlabel='holiday', ylabel='Density'>

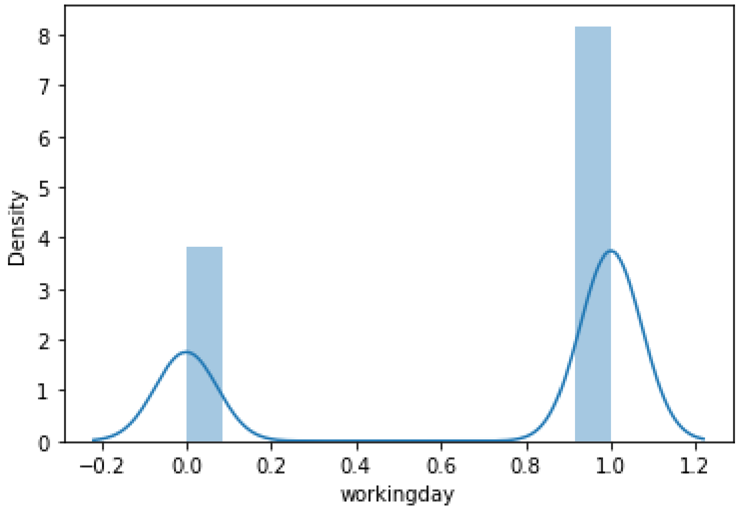


In [33]:

```
sns.distplot(data['workingday'])
```

Out[33]:

```
<AxesSubplot:xlabel='workingday', ylabel='Density'>
```



1)Selecting test for Working Day and no of electric cycles rented

In [34]:

```
# Here we use T-test because we dont know the population variances
#H0:no of bikes rented on working day and non working day are same
#Ha:no of bikes rented on working day and non working day are not same
#alpha=0.05
```

In [36]:

```
s1=data[data['workingday']==0]
s2=data[data['workingday']==1]
dof=s1.shape[0]+s2.shape[0]-2
```

In [37]:

```
# Checking the assumptions of T-Test:
    # 1) Normality check-Shapiro Wilk test
    # 2) population variances are equal-Levene's test
```

In [41]:

```
from scipy import stats
```

In [42]:

```
#1) Shapiro test
#H0: Data is in Normal Distribution
#Ha: Data is not normally distributed
stats.shapiro(s1['count'])
```

Out[42]:

```
ShapiroResult(statistic=0.8852126598358154, pvalue=4.203895392974451e-45)
```

In [43]:

```

#so the data is in normal distribution
```

In [54]:

```
#1) Shapiro test
#H0: Data is in Normal Distribution
#Ha: Data is not normally distributed
stats.shapiro(s2['count'])
```

Out[54]:

```
ShapiroResult(statistic=0.8702576160430908, pvalue=0.0)
```

In [51]:

```

#so the data is in normal distribution
```

In [44]:

```
#2) Levenes test
#H0: Both variances are equal
# Ha: Both variances are not equal
stats.levene(s1['count'],s2['count'])
```

Out[44]:

```
LeveneResult(statistic=0.004972848886504472, pvalue=0.9437823280916695)
```

In [45]:

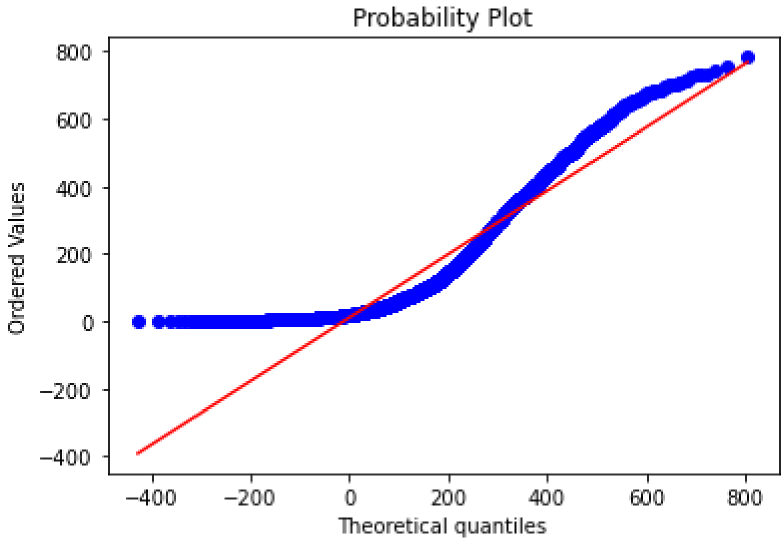
```
m1=s1['count'].mean()
st1=s1['count'].std()
m2=s2['count'].mean()
st2=s2['count'].std()
```

In [47]:

```
# Here we are continuing the analysis even levenes test failed
# QQPLOT
stats.probplot(s1['count'],dist='norm',sparams=(m1,st1),plot=plt)
```

Out[47]:

```
((array([-426.60470873, -384.69828364, -361.54747191, ...,  738.56071313,
          761.71152486,  803.61794995]),
  array([  1,   1,   1, ..., 743, 757, 783], dtype=int64)),
 (0.9418069773575181, 10.969770041429882, 0.941072405242598))
```

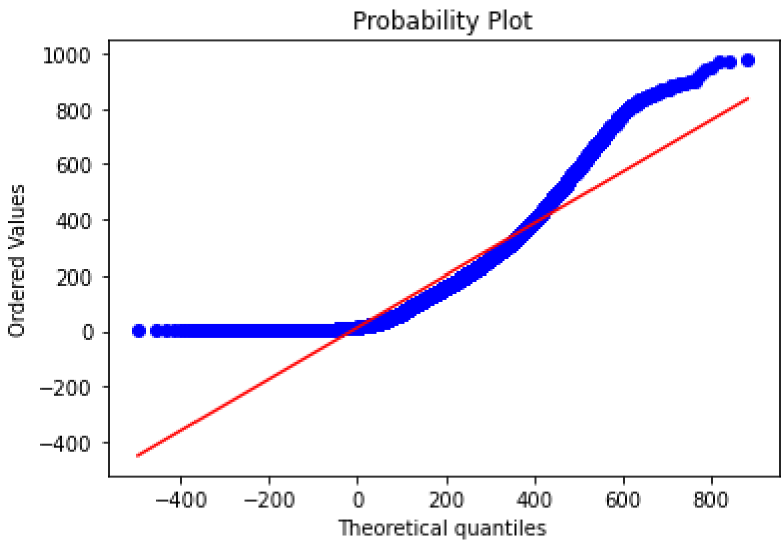


In [48]:

```
stats.probplot(s2['count'],dist='norm',sparams=(m2,st2),plot=plt)
```

Out[48]:

```
((array([-496.31766087, -453.99287291, -430.71904597, ...,  816.74279124,
          840.01661819,  882.34140615]),
  array([  1,   1,   1, ..., 968, 970, 977], dtype=int64)),
 (0.9333254460569756, 12.868980513910742, 0.9329510592827182))
```



In [56]:

```
# T-Test
stats.ttest_ind(s1['count'],s2['count'])
```

Out[56]:

```
Ttest_indResult(statistic=-1.2096277376026694, pvalue=0.22644804226361348)
```

Here the p-value 0.22 > alpha=0.05 so we fail to reject the null hypothesis

2) No. of cycles rented similar or different in different weather conditions

In [59]:

```
data['weather'].value_counts()
```

Out[59]:

```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

In [ ]:

```
#H0: No of cycles rented doesnt depend on weather
#H1: No of cycles rented does depend on weather
```

In [103]:

```

```

In [105]:

```
#There are 4 catgories in weather
w1=data[data['weather']==1]
w2=data[data['weather']==2]
w3=data[data['weather']==3]
w4=data[data['weather']==4]
```

In [129]:

```
mw1=w1['count'].mean()
stw1=w1['count'].std()
mw2=w2['count'].mean()
stw2=w2['count'].std()
mw3=w3['count'].mean()
stw3=w3['count'].std()
mw4=w4['count'].mean()
stw4=w4['count'].std()
print(mw1)
print(mw2)
print(mw3)
print(mw4)
```

```
205.23679087875416
178.95553987297106
118.84633294528521
164.0
```

In [116]:

```
# Shapiro test
# H0: data is noramlly distributed
# h1: data is normally distributed
stats.shapiro(w1['count'])
```

Out[116]:

```
ShapiroResult(statistic=0.8909225463867188, pvalue=0.0)
```

In [117]:

```
stats.shapiro(w2['count'])
```

Out[117]:

```
ShapiroResult(statistic=0.8767688274383545, pvalue=9.781063280987223e-43)
```

In [118]:

```
stats.shapiro(w3['count'])
```

Out[118]:

```
ShapiroResult(statistic=0.7674333453178406, pvalue=3.876134581802921e-33)
```

In [119]:

```
# Levene test
# H0:Both variances are equal
# Ha: Both variances are not equal
stats.levene(w1['count'],w2['count'],w3['count'],w4['count'])
```

Out[119]:

```
LeveneResult(statistic=54.85106195954556, pvalue=3.504937946833238e-35)
```

In [ ]:

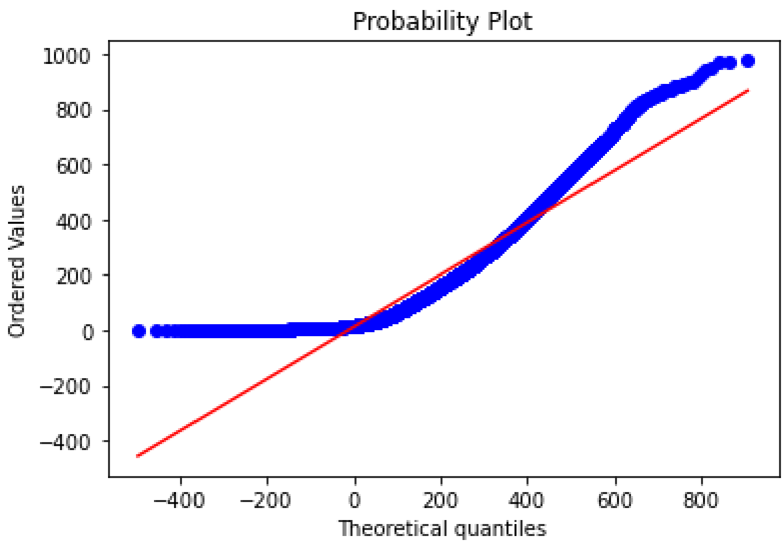
```

```

In [120]:

```
stats.probplot(w1['count'],dist='norm',sparams=(mw1,stw1),plot=plt)
```

Out[120]:

```
((array([-495.54024335, -452.34252071, -428.58469377, ...,  839.05827552,
          862.81610246,  906.01382511]),
  array([  1,   1,   1, ..., 968, 970, 977], dtype=int64)),
 (0.9443656760745699, 11.418210105164377, 0.9439766522111211))
```

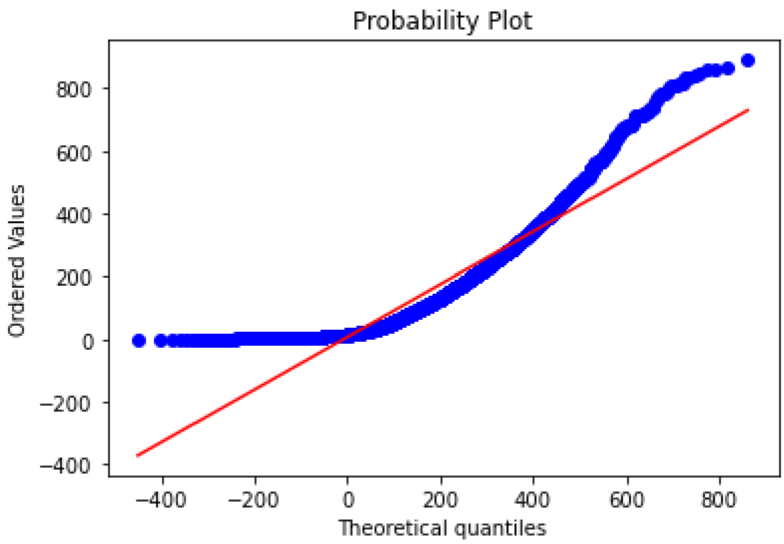


In [121]:

```
stats.probplot(w2['count'],dist='norm',sparams=(mw2,stw2),plot=plt)
```

Out[121]:

```
((array([-450.11300143, -404.11573316, -378.66907991, ...,  789.14266167,
          814.58931492,  860.58658319]),
  array([  1,   1,   1, ..., 862, 868, 890], dtype=int64)),
 (0.8396698568244296, 6.624393060702118, 0.9365111227539784))
```



In [122]:

```
stats.probplot(w3['count'],dist='norm',sparams=(mw3,stw3),plot=plt)
```

```
2,
         2.40861421e+02,  2.41458147e+02,  2.42057156e+02,  2.42658475e+0
2,
         2.43262135e+02,  2.43868165e+02,  2.44476595e+02,  2.45087456e+0
2,
         2.45700781e+02,  2.46316600e+02,  2.46934946e+02,  2.47555854e+0
2,
         2.48179356e+02,  2.48805487e+02,  2.49434282e+02,  2.50065777e+0
2,
         2.50700009e+02,  2.51337015e+02,  2.51976832e+02,  2.52619500e+0
2,
         2.53265058e+02,  2.53913546e+02,  2.54565005e+02,  2.55219478e+0
2,
         2.55877006e+02,  2.56537633e+02,  2.57201405e+02,  2.57868366e+0
2,
         2.58538563e+02,  2.59212042e+02,  2.59888854e+02,  2.60569046e+0
2,
         2.61252670e+02,  2.61939777e+02,  2.62630420e+02,  2.63324653e+0
2,
```

Here we should apply anova test

In [66]:

```
from scipy.stats import f_oneway
f_oneway(w1['count'],w2['count'],w3['count'],w4['count'])
```

Out[66]:

```
F_onewayResult(statistic=65.53024112793271, pvalue=5.482069475935669e-42)
```

In [67]:

```
# Here  p-value is less than alpha so we reject the null hypothesis
```

- therefore No of cycles rented does depent on Weather on that particular day

2) No. of cycles rented similar or different in different seasons

In [69]:

```
data['season'].value_counts()
```

Out[69]:

```
4    2734
3    2733
2    2733
1    2686
Name: season, dtype: int64
```

In [87]:

```
se1=data[data['season']==1]
se2=data[data['season']==2]
se3=data[data['season']==3]
se4=data[data['season']==4]
```

In [88]:

```
#H0: No of cycles rented doesnt depend on season
#H1: No of cycles rented does depend on season
```

In [89]:

```
# Shapiro test
# H0: data is noramlly distributed
# h1: data is normally distributed
stats.shapiro(se1['count'])
```

Out[89]:

```
ShapiroResult(statistic=0.8087379336357117, pvalue=0.0)
```

In [90]:

```
stats.shapiro(se2['count'])
```

Out[90]:

```
ShapiroResult(statistic=0.9004815220832825, pvalue=6.038716365804366e-39)
```

In [91]:

```
stats.shapiro(se3['count'])
```

Out[91]:

```
ShapiroResult(statistic=0.9148167371749878, pvalue=1.0437229694698105e-36)
```

In [92]:

```
stats.shapiro(se4['count'])
```

Out[92]:

```
ShapiroResult(statistic=0.8954642415046692, pvalue=1.130082751748606e-39)
```

In [93]:

```
# from above we can say that we reject null hypo but we have proofs that given data is norm
```

In [94]:

```
# Levene test
# H0:Both variances are equal
# Ha: Both variances are not equal
stats.levene(se1['count'],se2['count'],se3['count'],se4['count'])
```

Out[94]:

```
LeveneResult(statistic=187.7706624026276, pvalue=1.0147116860043298e-118)
```

In [95]:

```
f_oneway(se1['count'],se2['count'],se3['count'],se4['count'])
```

Out[95]:

```
F_onewayResult(statistic=236.94671081032106, pvalue=6.164843386499654e-149)
```

- From p-value we can say that it is lower than alpha so we reject the null hypothesis.
- Therefore the season feature effects the no of cycles rented

4)Weather is dependent on season (check between 2 predictor variable)

For this we use chisquare test of independence

In [ ]:

```
# Here we need to check whether there is any relation between Weather and season
# H0: There is no relation between weather and season
# Ha: Weather is dependent on season
```

In [101]:

```
# Creating the observed value table or contingency table
t=pd.crosstab(data['weather'],data['season'])
```

In [97]:

```
t
```

Out[97]:

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 [99]:

```
chi2_stat,p,dof,expected = stats.chi2_contingency(t)
```

In [102]:

```
print('chi2_statistic :',chi2_stat)
print('p-value :',p)
print('Degrees Of Freedom',dof)
print('Expected',expected)
```

```
chi2_statistic : 49.15865559689363
p-value : 1.5499250736864862e-07
Degrees Of Freedom 9
Expected [[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 [136]:

```
x3=data[data['season']==3].mean()
x3['count'].mean()
```

Out[136]:

```
234.417124039517
```

- here as p value is less than alpha we reject null hypothesis
- Therfore we can conclude that weather is dependent on season

Business Insights:

```
* Working day has clear effect on electric cycles rented
* The impact of working days is more when there is work .
* The number of cycles rented are dependent on the weather and different seasons
* The average number of cycles rented is more when the weather is clear
* The average number of cycles rented is more when it is fall
* weather is depenent on the seasons
```

Recommendations:

```
* Company should give coupons to the working professiols
* Company should make make campaigns near working offices
* Comapny should keep stock of vehicles more in fall and spring seasons
* Company should keep stock of vehicles on the clear days
* Comapny should keep track of good weather analysis on daily basis
* Company should come up with a way to make more cycles subscribed in holidays to
o.
* They can give more coupons on holidays
```

In [ ]: