In [1]:

```
# Import the required library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
data=pd.read_csv('hour.csv')
data
```

Out[2]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	win
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	
4	5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
17374	17375	2012-12-31	1	1	12	19	0	1	1	2	0.26	0.2576	0.60	
17375	17376	2012-12-31	1	1	12	20	0	1	1	2	0.26	0.2576	0.60	
17376	17377	2012-12-31	1	1	12	21	0	1	1	1	0.26	0.2576	0.60	
17377	17378	2012-12-31	1	1	12	22	0	1	1	1	0.26	0.2727	0.56	
17378	17379	2012-12-31	1	1	12	23	0	1	1	1	0.26	0.2727	0.65	

17379 rows × 17 columns

In [6]:

```
data.isnull().sum()
```

Out[6]:
```
instant       0
dteday        0
season        0
yr            0
mnth          0
hr            0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

In [3]:
```
data['registered']+data['casual']!=data['cnt']
```

Out[3]:
```
0        False
1        False
2        False
3        False
4        False
         ...
17374    False
17375    False
17376    False
17377    False
17378    False
Length: 17379, dtype: bool
```

In [8]:
```
duplicate=data[data.duplicated()]
duplicate
```

Out[8]:

instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed

In [9]:
```
#Sanity checks:
# 1. Check if registered + casual = cnt for all the records. If not, the row is junk and s
#dropped.

data['registered']+data['casual']!=data['cnt']
np.sum(data['registered']+data['casual']!=data['cnt'])
```

Out[9]:
```
0
```

In [11]:
```
# write the code to drop the rows where this is true
data.drop(data[data["registered"]+data["casual"]!=data["cnt"]].index,inplace=True)
```

In [12]:
```
# Month values should be 1-12 only
data['mnth'].unique()
```

Out[12]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12], dtype=int64)
```

In [13]:
```
#Hour values should be 0-23
data['hr'].unique()
```

Out[13]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23], dtype=int64)
```

In [ ]:
```
#The variables ‘casual’ and ‘registered’ are redundant and need to be dropped.
#‘Instant’ is the index and needs to be dropped too. The date column dteday will not be us
#in the model building, and therefore needs to be dropped.
#Create a new dataframe named inp1.
```

In [15]:
```
a = ['casual', 'registered', 'dteday', 'instant']
inp1 = data.drop(a, axis=1).copy()     # axis = 1 becoz we have remove colums otherwise 0
inp1.shape
```

Out[15]:
```
(17379, 13)
```

In [ ]:
```
#Univariate analysis:
```

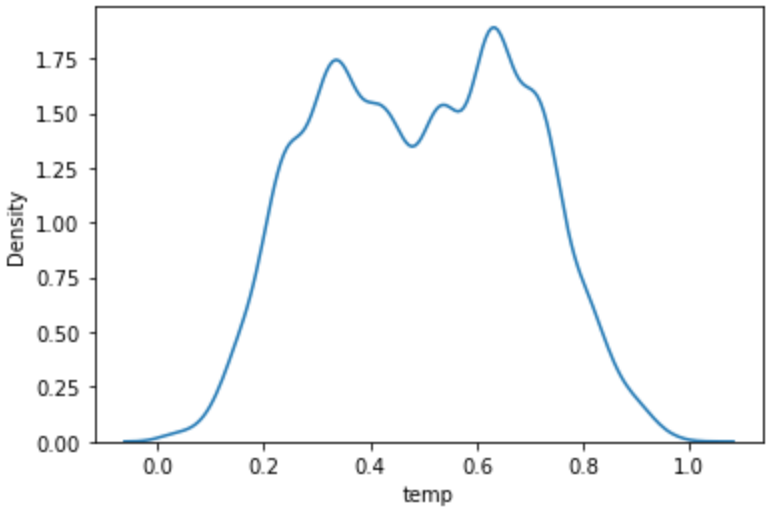
In [16]:
```
#Describe the numerical fields in the dataset using pandas describe method.
inp1.describe()
```

Out[16]:

	season	yr	mnth	hr	holiday	weekday	workingday	wea
count	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.
mean	2.501640	0.502561	6.537775	11.546752	0.028770	3.003683	0.682721	1.
std	1.106918	0.500008	3.438776	6.914405	0.167165	2.005771	0.465431	0.
min	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.
25%	2.000000	0.000000	4.000000	6.000000	0.000000	1.000000	0.000000	1.
50%	3.000000	1.000000	7.000000	12.000000	0.000000	3.000000	1.000000	1.
75%	3.000000	1.000000	10.000000	18.000000	0.000000	5.000000	1.000000	2.
max	4.000000	1.000000	12.000000	23.000000	1.000000	6.000000	1.000000	4.

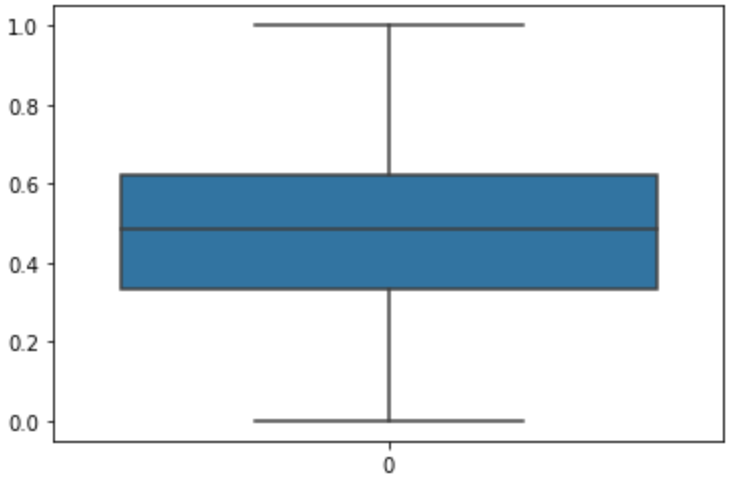
In [7]:
```
#Make density plot for temp. This would give a sense of the centrality and the spread of t
sns.kdeplot(data['temp'])
```

Out[7]:
```
<AxesSubplot:xlabel='temp', ylabel='Density'>
```



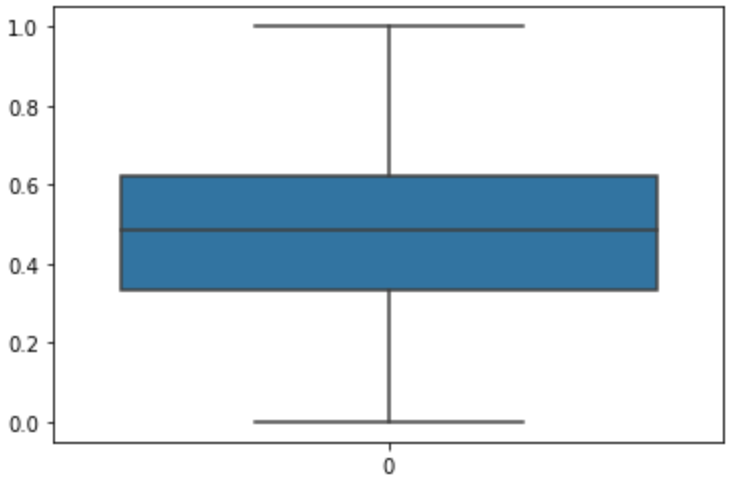
In [17]:

```
#Boxplot for atemp .Are there any outliers?
sns.boxplot(data=inp1.atemp)
plt.show()
import warnings
warnings.filterwarnings("ignore")
```



In [11]:

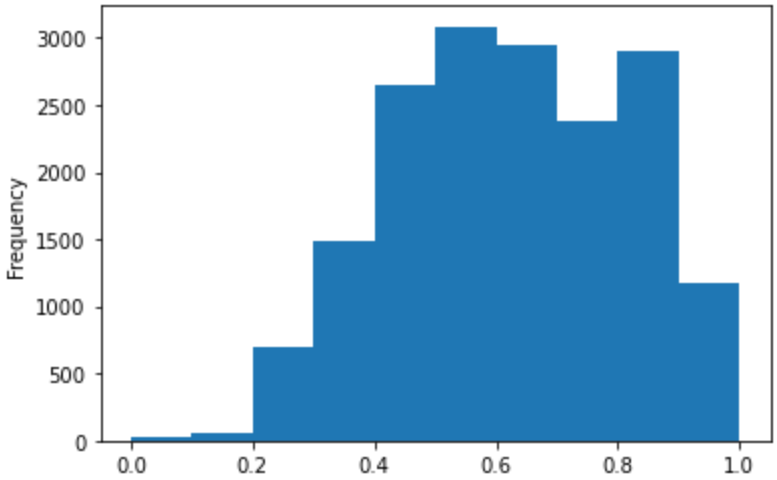
```
sns.boxplot(data=inp1.atemp)
plt.show()
```



In [12]:

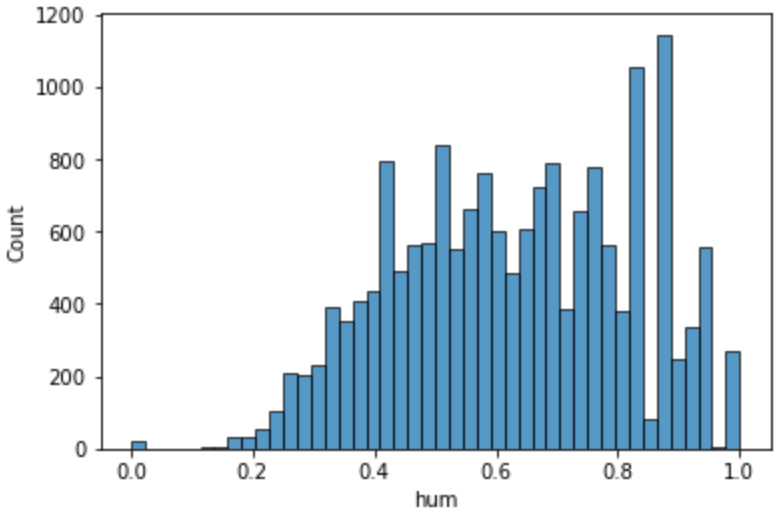
```
#Histogram for hum #Do you detect any abnormally high values?
inp1.hum.plot.hist()
```

```
plt.show()
```

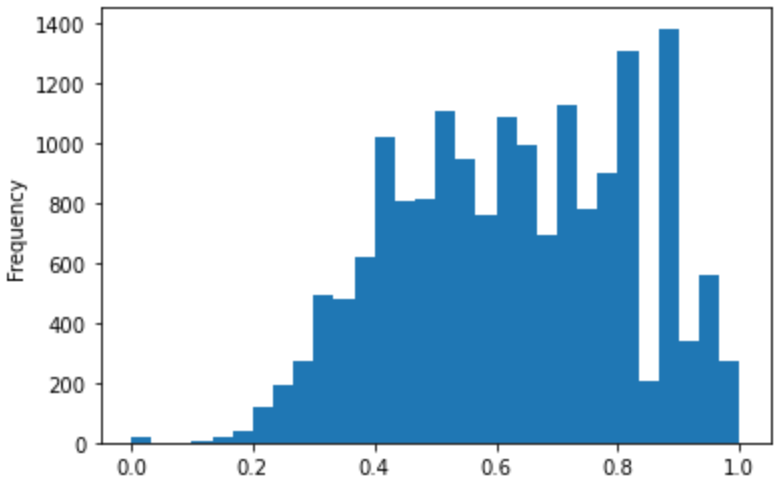


In [13]:
```
sns.histplot(data=inp1.hum)
```

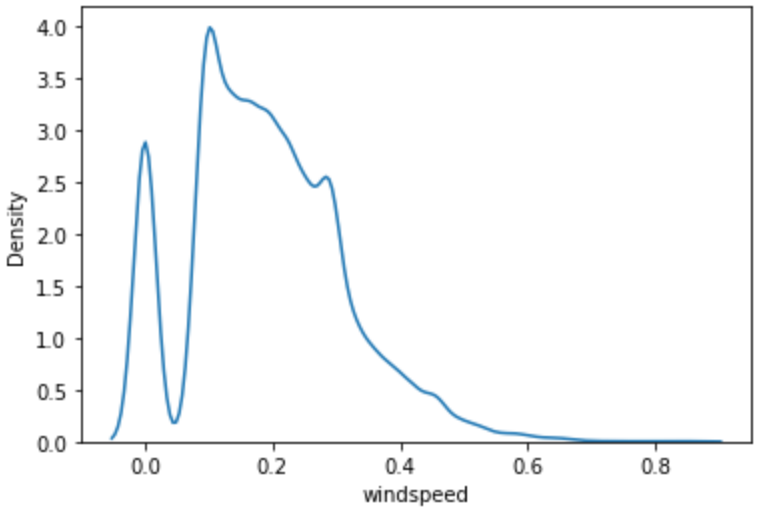
Out[13]:
```
<AxesSubplot:xlabel='hum', ylabel='Count'>
```



In [14]:
```
inp1.hum.plot.hist(bins=30)
plt.show()
```

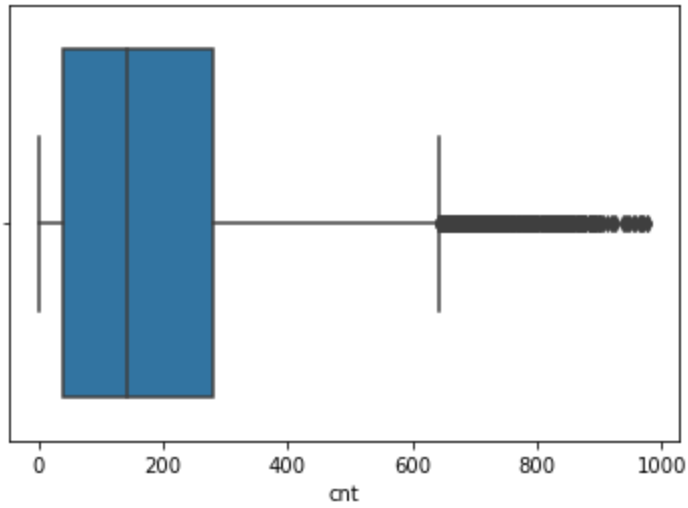


In [21]:
```
#Density plot for windspeed
sns.kdeplot(data['windspeed'])
plt.show()
```



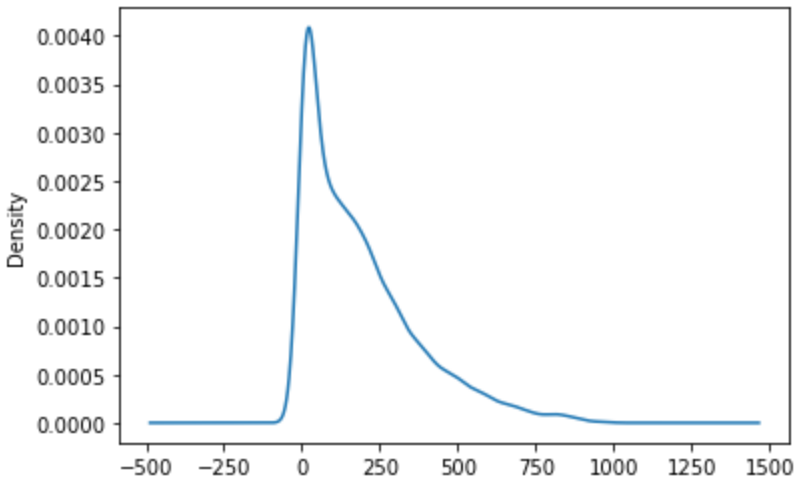
In [23]:

```
#Box and density plot for cnt – this is the variable of interest
#Do you see any outliers in the boxplot?
#Does the density plot provide a similar insight?
sns.boxplot(inp1.cnt)
plt.show()
```



In [25]:

```
inp1.cnt.plot.density()
plt.show()
```



In [26]:

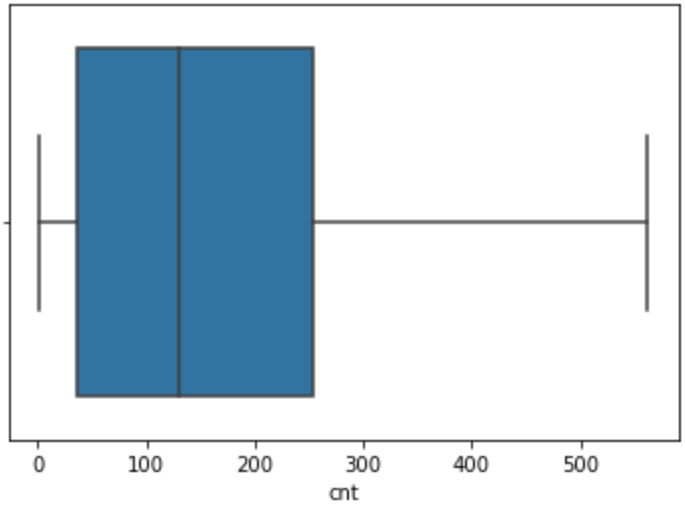
```
# Outlier treatment:
# 1.5 IQR rule
```

```
# Business -95%
inp1.cnt.quantile([0.1,0.25,0.50,0.75,0.90,0.95,0.99])
```

Out[26]:
```
0.10      9.00
0.25     40.00
0.50    142.00
0.75    281.00
0.90    451.20
0.95    563.10
0.99    782.22
Name: cnt, dtype: float64
```

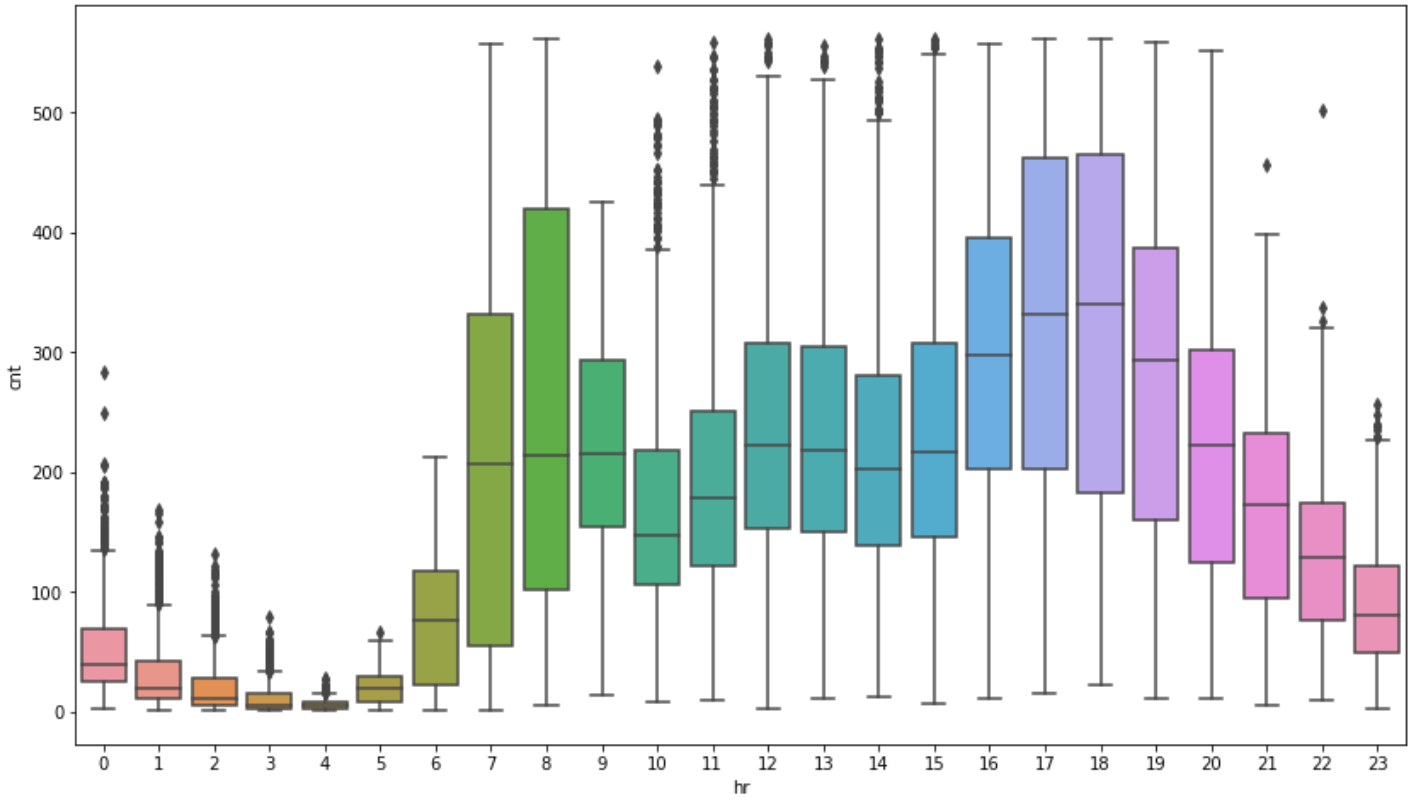
In [27]:
```
#Decide the cutoff percentile and drop records with values higher than the cutoff.
#Name the new dataframe as inp2.
inp2=inp1[inp1.cnt<563].copy()
```

In [28]:
```
sns.boxplot(inp2.cnt)
plt.show()
```



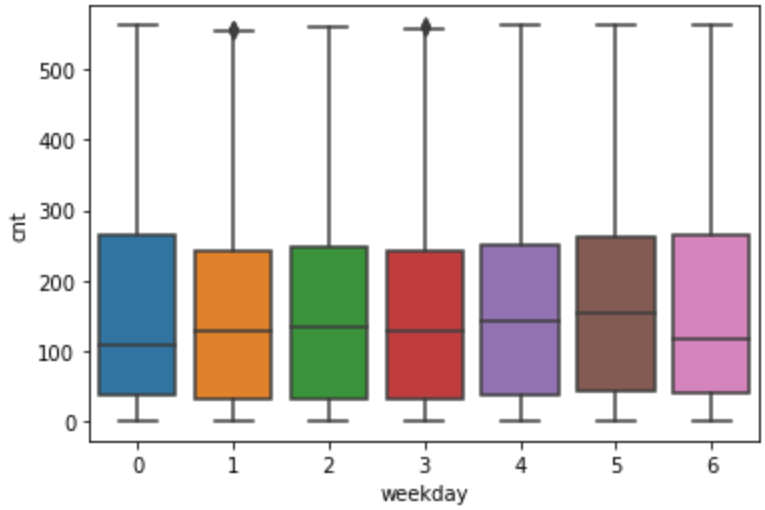
In [29]:
```
#Bivariate analysis
```

In [34]:
```
# Make boxplot for cnt vs. hour
plt.figure(figsize=[14,8])
sns.boxplot("hr","cnt",data=inp2)
plt.show()
```



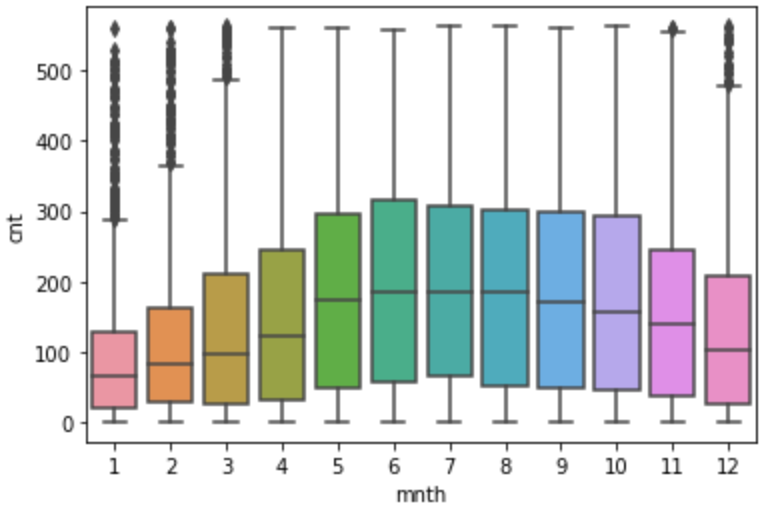
In [35]:

```
#Make boxplot for cnt vs. weekday
#Is there any difference in the rides by days of the week?
sns.boxplot("weekday","cnt",data=inp2)
plt.show()
```



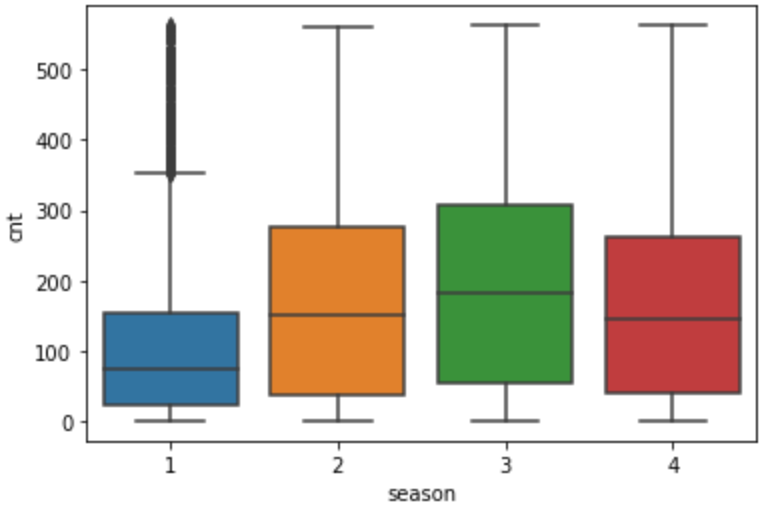
In [37]:

```
#Make boxplot for cnt vs. month

#Look at the median values. Any month(s) that stand out?
sns.boxplot("mnth","cnt",data=inp2)
plt.show()
```



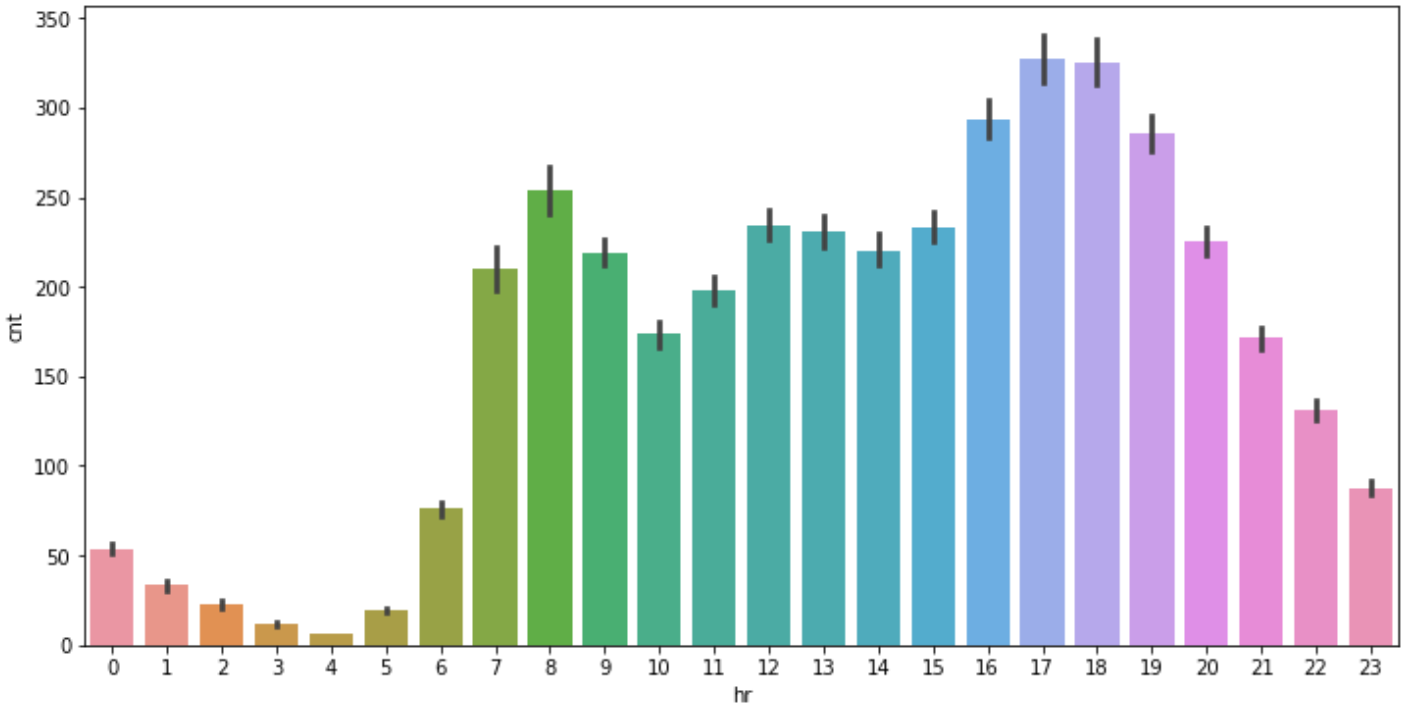
In [38]:

```
#Make boxplot for cnt vs. season

#Which season has the highest rides in general? Expected?
sns.boxplot("season","cnt",data=inp2)
plt.show()
```



In [40]:

```
#Make a bar plot with the median value of cnt for each hr

#Does this paint a different picture from the box plot?
plt.figure(figsize=[12,6])
sns.barplot("hr","cnt",data=inp2)
plt.show()
```



In [41]:

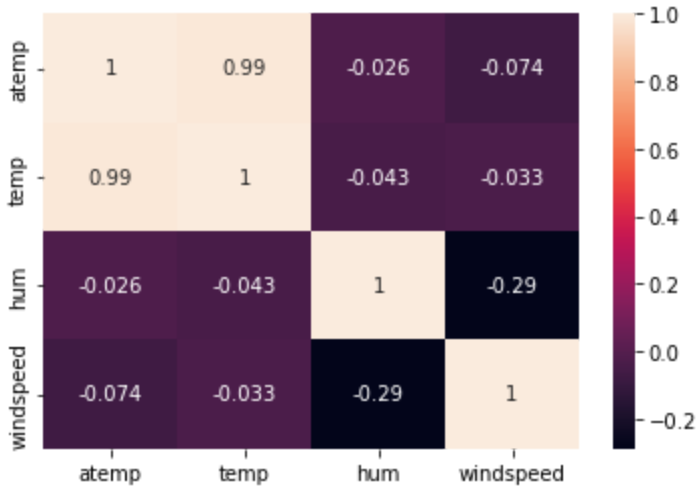
```
#Make a correlation matrix for variables atemp, temp, hum, and windspeed

#Which variables have the highest correlation?
a=['atemp','temp','hum','windspeed']
corrs=inp2[a].corr()
corrs
```

Out[41]:

	atemp	temp	hum	windspeed
atemp	1.000000	0.988218	-0.025747	-0.073985
temp	0.988218	1.000000	-0.042603	-0.033209
hum	-0.025747	-0.042603	1.000000	-0.288648
windspeed	-0.073985	-0.033209	-0.288648	1.000000

In [48]:

```
sns.heatmap(corrs,annot=True)
plt.show()
```



In [49]:

```
# Data preprocessing
```

In [51]:
```
#Treating mnth column

#For values 5,6,7,8,9,10, replace with a single value 5. This is because these have very
#similar values for cnt.

#Get dummies for the updated 6 mnth values
inp3=inp2.copy()
inp3.mnth[inp3.mnth.isin([5,6,7,8,9])] = 5
```

In [52]:
```
inp3['mnth'].value_counts()
# or you can use :  np.unique(inp3.mnth)
```

Out[52]:
```
5     6785
12    1455
1     1429
3     1412
11    1392
4     1349
10    1341
2     1339
Name: mnth, dtype: int64
```

In [58]:
```
#Treating hr column

#Create new mapping: 0-5: 0, 11-15: 11; other values are untouched. Again, the bucketing i
#a way that hr values with similar levels of cnt are treated the same.
inp3.hr[inp3.hr.isin([0,1,2,3,4,5])] = 0
inp3.hr[inp3.hr.isin([11,12,13,14,15])] = 11

inp3['hr'].value_counts()
```

Out[58]:
```
0     4276
11    3482
22     728
23     728
9      727
10     727
20     727
21     727
6      725
7      724
16     689
19     671
8      547
18     546
17     478
Name: hr, dtype: int64
```

In [59]:
```
#Get dummy columns for season, weathersit, weekday, mnth, and hr.
list=['season','weathersit','weekday','mnth','hr']
```

In [60]:
```
inp3=pd.get_dummies(inp3,columns=list)
```

In [62]:
```
inp3.head()
```

Out[62]:

	yr	holiday	workingday	temp	atemp	hum	windspeed	cnt	season_1	season_2	...	hr_10	hr_11	hr_16	hr
	0	0	0	0.24	0.2879	0.81	0.0	16	1	0	...	0	0	0	

	yr	holiday	workingday	temp	atemp	hum	windspeed	cnt	season_1	season_2	...	hr_10	hr_11	hr_16	hr
1	0	0	0	0.22	0.2727	0.80	0.0	40	1	0	...	0	0	0	
2	0	0	0	0.22	0.2727	0.80	0.0	32	1	0	...	0	0	0	
3	0	0	0	0.24	0.2879	0.75	0.0	13	1	0	...	0	0	0	
4	0	0	0	0.24	0.2879	0.75	0.0	1	1	0	...	0	0	0	

5 rows × 46 columns

In [76]:
```
#Train test split: Apply 70-30 split.

#call the new dataframes df_train and df_test
# Split
from sklearn.model_selection import train_test_split
df_train, df_test=train_test_split(inp3,test_size=0.3,random_state=5)
```

In [77]:
```
df_train.shape
```

Out[77]:
```
(11551, 46)
```

In [78]:
```
df_test.shape
```

Out[78]:
```
(4951, 46)
```

In [79]:
```
#Separate X and Y for df_train and df_test. For example, you should have X_train, y_train
#df_train. y_train should be the cnt column from inp3 and X_train should be all other colu
y_train = df_train.pop("cnt")
X_train = df_train
```

In [80]:
```
y_train
```

Out[80]:
```
10182    428
13824     17
3508      12
2649     196
5772      98
        ...
740       39
1032     206
5565     440
3048     146
2915     188
Name: cnt, Length: 11551, dtype: int64
```

In [81]:
```
X_train
```

Out[81]:

	yr	holiday	workingday	temp	atemp	hum	windspeed	season_1	season_2	season_3	...	hr_10	hr_11
10182	1	0	1	0.20	0.2273	0.59	0.1045	1	0	0	...	0	0
13824	1	0	0	0.70	0.6667	0.84	0.2239	0	0	1	...	0	0
3508	0	0	1	0.66	0.5909	0.94	0.1343	0	1	0	...	0	0
2649	0	0	1	0.52	0.5000	1.00	0.0896	0	1	0	...	0	0

	yr	holiday	workingday	temp	atemp	hum	windspeed	season_1	season_2	season_3	...	hr_10	hr_11
5772	0	0	1	0.64	0.6061	0.73	0.1642	0	0	1	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
740	0	0	1	0.16	0.1364	0.43	0.3582	1	0	0	...	0	0
1032	0	0	1	0.32	0.3030	0.22	0.2239	1	0	0	...	0	0
5565	0	0	1	0.74	0.6818	0.55	0.2985	0	0	1	...	0	0
3048	0	0	1	0.50	0.4848	0.72	0.2537	0	1	0	...	0	0
2915	0	0	1	0.46	0.4545	0.67	0.2836	0	1	0	...	0	0

11551 rows × 45 columns

In [82]:
```
y_test = df_test.pop("cnt")
X_test = df_test
```

In [83]:
```
# Model building

#Use linear regression as the technique

#Report the R2 on the train set

from sklearn.linear_model import LinearRegression
liner_reg=LinearRegression()
```

In [84]:
```
# fit() training
liner_reg.fit(X_train,y_train)
```

Out[84]:
```
LinearRegression()
```

In [85]:
```
y_pred=liner_reg.predict(X_test)
y_pred
```

Out[85]:
```
array([189.75 , 172.875,   9.125, ...,  -6.875,  79.375, -76.125])
```

In [86]:
```
y_test
```

Out[86]:
```
9946     163
4069     136
3747      23
9356     410
2780     371
        ...
7626     142
4265     381
17288     15
10773     59
7113      19
Name: cnt, Length: 4951, dtype: int64
```

In [87]:
```
# calculate r2 score
from sklearn.metrics import r2_score
print(r2_score(y_pred,y_test))
```

```
0.5146643160558204
```

In [88]:

```
# Cross_validation
from sklearn.metrics import r2_score
print(r2_score(liner_reg.predict(X_train),y_train))
```

```
0.5009561935077527
```

In [ ]:

```

```