In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

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

In [3]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17379 entries, 0 to 17378
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   instant     17379 non-null  int64  
 1   dteday      17379 non-null  object 
 2   season      17379 non-null  int64  
 3   yr          17379 non-null  int64  
 4   mnth        17379 non-null  int64  
 5   hr          17379 non-null  int64  
 6   holiday     17379 non-null  int64  
 7   weekday     17379 non-null  int64  
 8   workingday  17379 non-null  int64  
 9   weathersit  17379 non-null  int64  
 10  temp        17379 non-null  float64
 11  atemp       17379 non-null  float64
 12  hum         17379 non-null  float64
 13  windspeed   17379 non-null  float64
 14  casual      17379 non-null  int64  
 15  registered  17379 non-null  int64  
 16  cnt         17379 non-null  int64  
dtypes: float64(4), int64(12), object(1)
memory usage: 2.3+ MB
```

In [4]:

```
data.head()
```

Out[4]:

| | instant | dteday | season | yr | mnth | hr | holiday | weekday | workingday | weathersit | temp | ate |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2011-01-01 | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2 |
| **1** | 2 | 2011-01-01 | 1 | 0 | 1 | 1 | 0 | 6 | 0 | 1 | 0.22 | 0.2 |
| **2** | 3 | 2011-01-01 | 1 | 0 | 1 | 2 | 0 | 6 | 0 | 1 | 0.22 | 0.2 |
| **3** | 4 | 2011-01-01 | 1 | 0 | 1 | 3 | 0 | 6 | 0 | 1 | 0.24 | 0.2 |
| **4** | 5 | 2011-01-01 | 1 | 0 | 1 | 4 | 0 | 6 | 0 | 1 | 0.24 | 0.2 |

In [5]:

```
# Check for null values in the data and drop records with NAs.


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

Out[5]:

```
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 [6]:

```
# if data is missing
data.dropna(inplace=True)
```

In [7]:

```
# Sanity checks:
# Check if registered + casual = cnt for all the records. If not, the row is junk and shoul
```

In [8]:

```
data['registered']+data['casual']!=data['cnt']
```

Out[8]:

```
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 [9]:

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

Out[9]:

```
0
```

In [10]:

```
# Suppose i have to drop rows based on some condition, in this condition there is no

# drop function

data.drop(data[data['registered']+data['casual']!=data['cnt']].index,inplace=True)
```

In [11]:

```
# Month values should be 1-12 only

data['mnth'].unique()
```

Out[11]:

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

In [12]:

```
# Hour values should be 0-23

data['hr'].unique()
```

Out[12]:

```
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)
```

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 used in the model building, and therefore needs to be dropped. Create a new dataframe named inp1

In [13]:

```
col_drop=['casual','registered','dteday','instant']
inp1=data.drop(col_drop,axis=1).copy()   # use copy method
```

In [14]:

```
inp1.head()
```

Out[14]:

| | season | yr | mnth | hr | holiday | weekday | workingday | weathersit | temp | atemp | hum | winds |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.81 | |
| **1** | 1 | 0 | 1 | 1 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | |
| **2** | 1 | 0 | 1 | 2 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | |
| **3** | 1 | 0 | 1 | 3 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | |
| **4** | 1 | 0 | 1 | 4 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | |

In [15]:

```
inp1.shape
```

Out[15]:

```
(17379, 13)
```

5. Univariate analysis:

Describe the numerical fields in the dataset using pandas describe method.

Make density plot for temp. This would give a sense of the centrality and the spread of the distribution.

Boxplot for atemp

Are there any outliers?

Histogram for hum

Do you detect any abnormally high values?

Density plot for windspeed

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?

In [16]:

```
# Describe the numerical fields in the dataset using pandas describe method.

inp1.describe()
```

Out[16]:

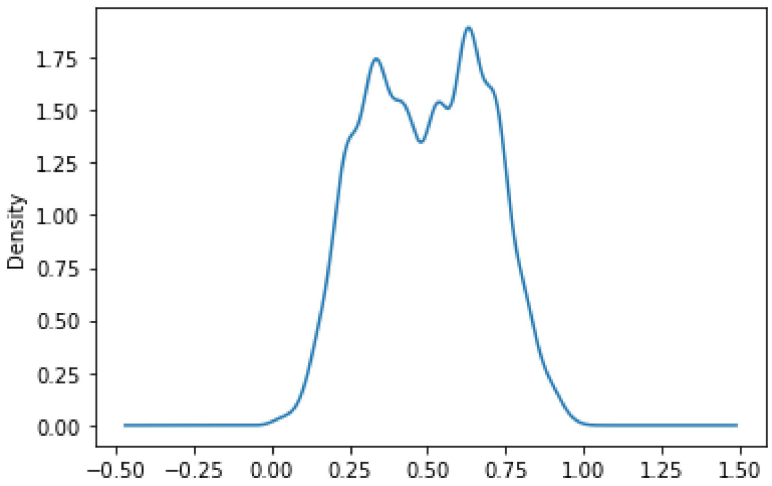
| | season | yr | mnth | hr | holiday | weekday | |
|---|---|---|---|---|---|---|---|
| **count** | 17379.000000 | 17379.000000 | 17379.000000 | 17379.000000 | 17379.000000 | 17379.000000 | 17 |
| **mean** | 2.501640 | 0.502561 | 6.537775 | 11.546752 | 0.028770 | 3.003683 | |
| **std** | 1.106918 | 0.500008 | 3.438776 | 6.914405 | 0.167165 | 2.005771 | |
| **min** | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | |
| **25%** | 2.000000 | 0.000000 | 4.000000 | 6.000000 | 0.000000 | 1.000000 | |
| **50%** | 3.000000 | 1.000000 | 7.000000 | 12.000000 | 0.000000 | 3.000000 | |
| **75%** | 3.000000 | 1.000000 | 10.000000 | 18.000000 | 0.000000 | 5.000000 | |
| **max** | 4.000000 | 1.000000 | 12.000000 | 23.000000 | 1.000000 | 6.000000 | |

In [17]:

```
# Make density plot for temp. This would give a sense of the centrality and the spread of t

inp1.temp.plot.density()
```

Out[17]:

```
<AxesSubplot:ylabel='Density'>
```



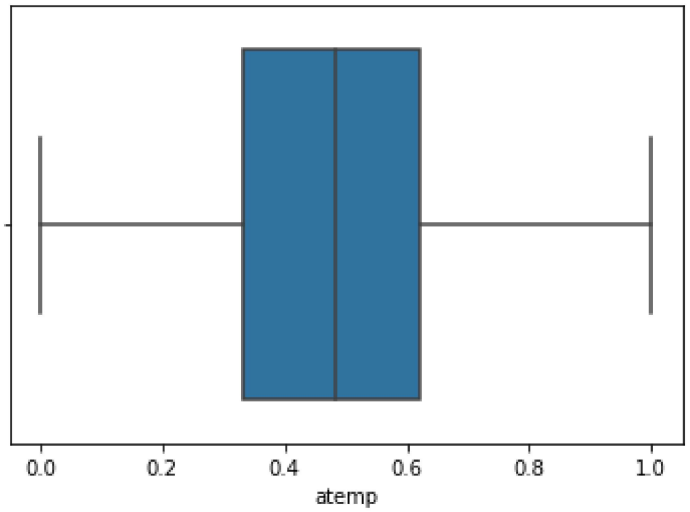
In [18]:

```
# Boxplot for atemp

import seaborn as sns
sns.boxplot(inp1['atemp'])

# we dont have any outlier
```

```
C:\Users\akhilesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variable as a keyword arg: x. From version 0.
12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```

Out[18]:

```
<AxesSubplot:xlabel='atemp'>
```

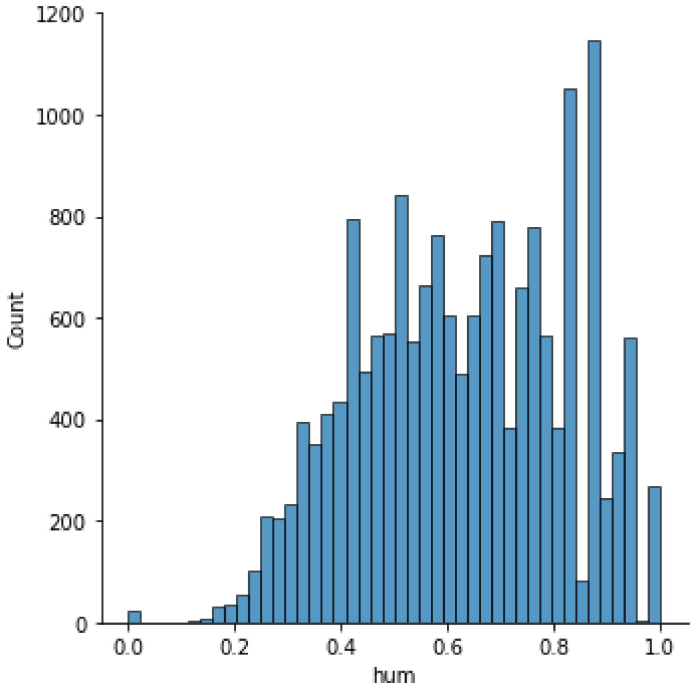


In [19]:

```
# Histogram for hum ??

## Do you detect any abnormally high values ??

sns.displot(inp1["hum"])

# left skewed data--- definately outlier is there
```

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0x223a6395c10>
```



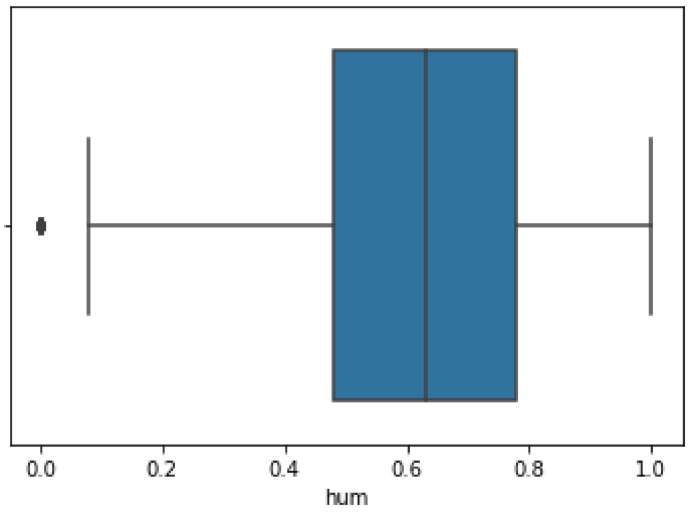
In [20]:

```
sns.boxplot(inp1['hum'])
```

```
C:\Users\akhilesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variable as a keyword arg: x. From version 0.
12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```

Out[20]:

```
<AxesSubplot:xlabel='hum'>
```



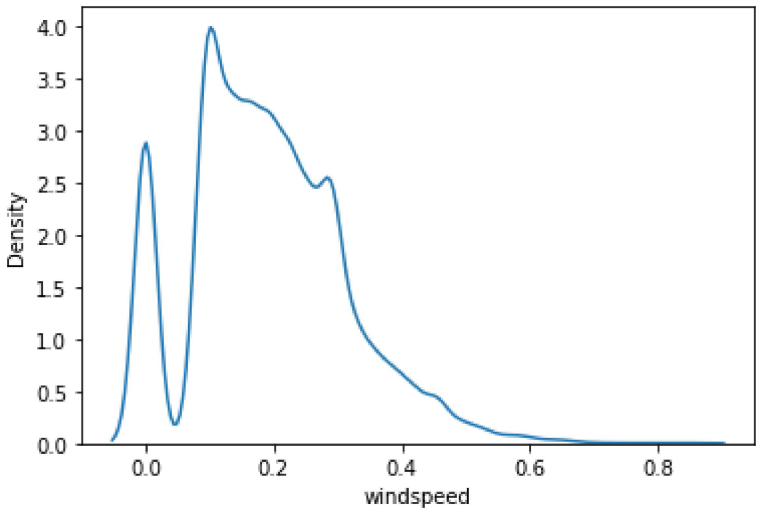
In [21]:

```
# Density plot for windspeed

sns.distplot(inp1['windspeed'], hist=False)
```

```
C:\Users\akhilesh\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `kdeplot` (an axes-level function f
or kernel density plots).
  warnings.warn(msg, FutureWarning)
```

Out[21]:

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



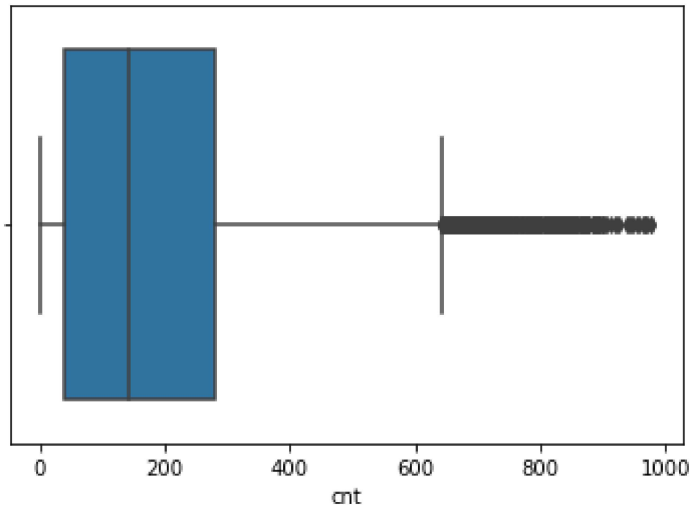
In [22]:

```
# 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)

# lot of outliers are there
```

```
C:\Users\akhilesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variable as a keyword arg: x. From version 0.
12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```

Out[22]:

```
<AxesSubplot:xlabel='cnt'>
```




6. Outlier treatment:

Cnt looks like some hours have rather high values. You'll need to treat these outliers so that they don't skew the analysis and the model.

Find out the following percentiles: 10, 25, 50, 75, 90, 95, 99

Decide the cutoff percentile and drop records with values higher than the cutoff. Name the new dataframe as inp2.

In [23]:

```
inp1.cnt.quantile([0.10,0.25,0.50,0.75,0.90,0.95,0.99])

# maximum outlier are comming after 600 in boxplot. so, selection will be 95 percentile
```

Out[23]:

```
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 [24]:

```
inp2=inp1[inp1.cnt<563].copy()
inp2
```

Out[24]:

| | season | yr | mnth | hr | holiday | weekday | workingday | weathersit | temp | atemp | hum |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.81 |
| **1** | 1 | 0 | 1 | 1 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 |
| **2** | 1 | 0 | 1 | 2 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 |
| **3** | 1 | 0 | 1 | 3 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 |
| **4** | 1 | 0 | 1 | 4 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **17374** | 1 | 1 | 12 | 19 | 0 | 1 | 1 | 2 | 0.26 | 0.2576 | 0.60 |
| **17375** | 1 | 1 | 12 | 20 | 0 | 1 | 1 | 2 | 0.26 | 0.2576 | 0.60 |
| **17376** | 1 | 1 | 12 | 21 | 0 | 1 | 1 | 1 | 0.26 | 0.2576 | 0.60 |
| **17377** | 1 | 1 | 12 | 22 | 0 | 1 | 1 | 1 | 0.26 | 0.2727 | 0.56 |
| **17378** | 1 | 1 | 12 | 23 | 0 | 1 | 1 | 1 | 0.26 | 0.2727 | 0.65 |

16502 rows × 13 columns

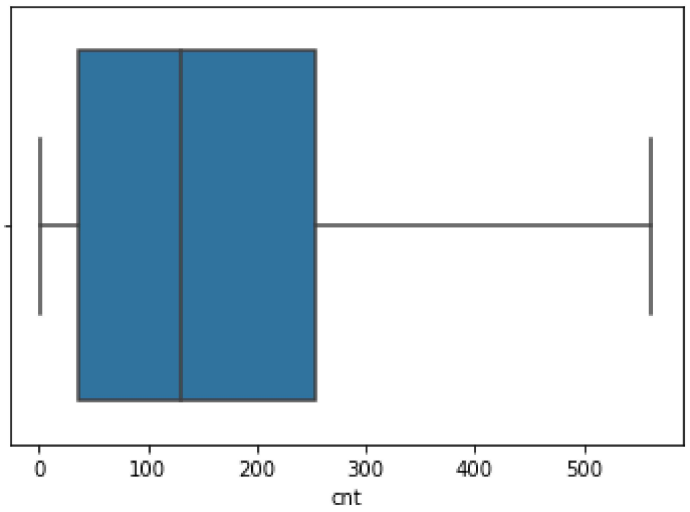
In [25]:

```
sns.boxplot(inp2.cnt)

# Now no outliers are there
```

```
C:\Users\akhilesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variable as a keyword arg: x. From version 0.
12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```

Out[25]:

```
<AxesSubplot:xlabel='cnt'>
```




7. Bivariate analysis

1.Make boxplot for cnt vs. hour

What kind of pattern do you see?

2.Make boxplot for cnt vs. weekday

Is there any difference in the rides by days of the week?

3.Make boxplot for cnt vs. month

Look at the median values. Any month(s) that stand out?

4.Make boxplot for cnt vs. season

Which season has the highest rides in general? Expected?

5.Make a bar plot with the median value of cnt for each hr

Does this paint a different picture from the box plot?

6.Make a correlation matrix for variables atemp, temp, hum, and windspeed

Which variables have the highest correlation?

In [26]:

```
import matplotlib.pyplot as plt
```

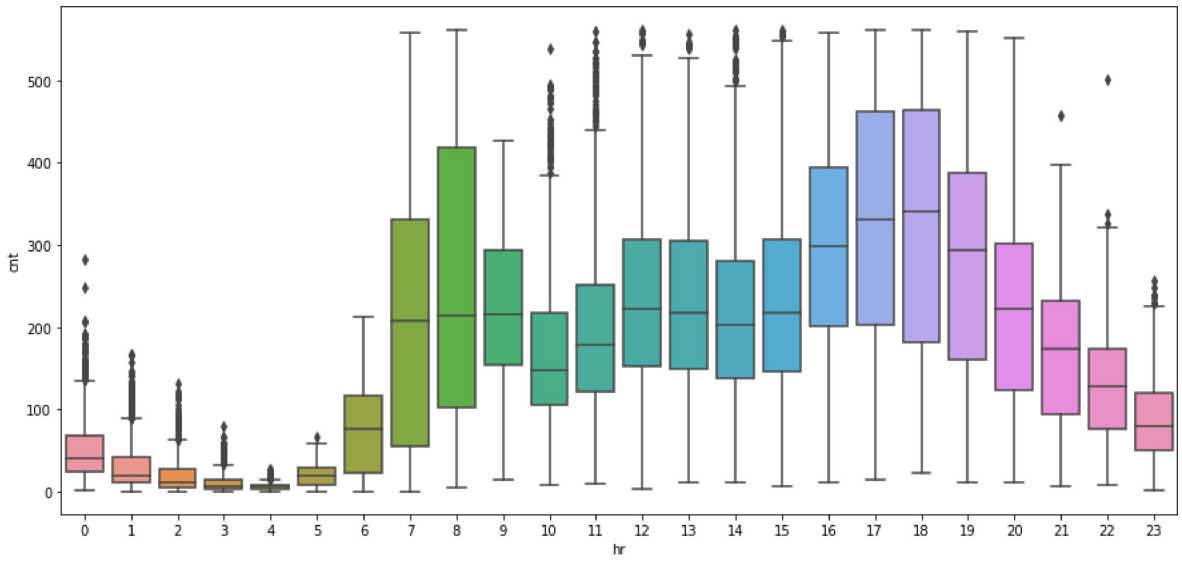
In [27]:

```
# 1.Make boxplot for cnt vs. hour. What kind of pattern do you see?


plt.figure(figsize=[15,7])
sns.boxplot("hr","cnt",data=inp2)
```

```
C:\Users\akhilesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpret
ation.
  warnings.warn(
```

Out[27]:

```
<AxesSubplot:xlabel='hr', ylabel='cnt'>
```



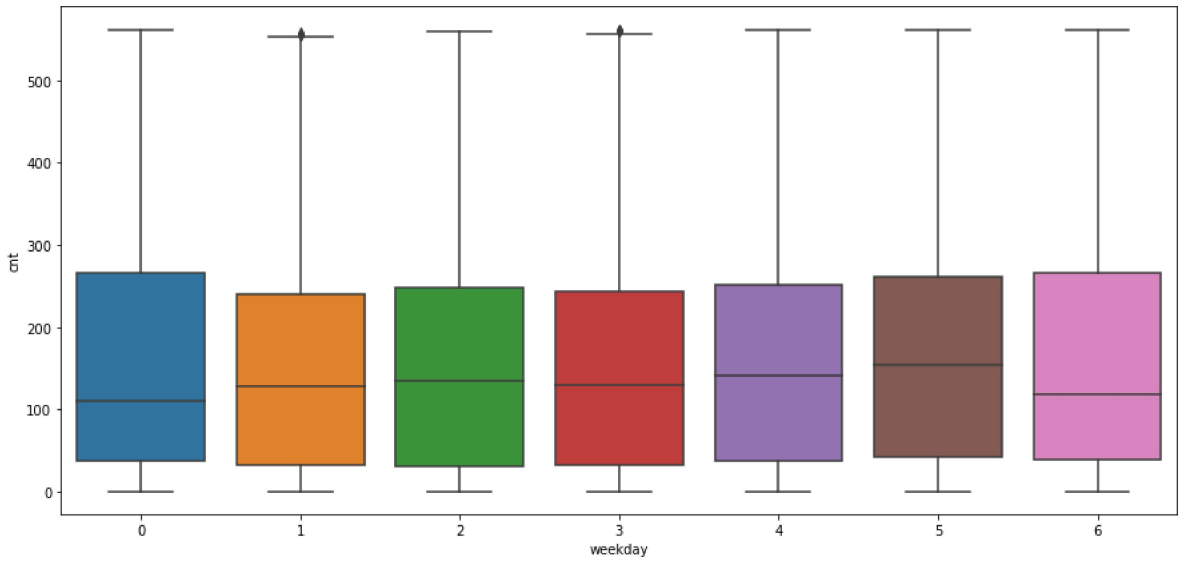
In [28]:

```
# 2.Make boxplot for cnt vs. weekday. Is there any difference in the rides by days of the w

plt.figure(figsize=[15,7])
sns.boxplot("weekday","cnt",data=inp2)
```

```
C:\Users\akhilesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpret
ation.
  warnings.warn(
```

Out[28]:

```
<AxesSubplot:xlabel='weekday', ylabel='cnt'>
```



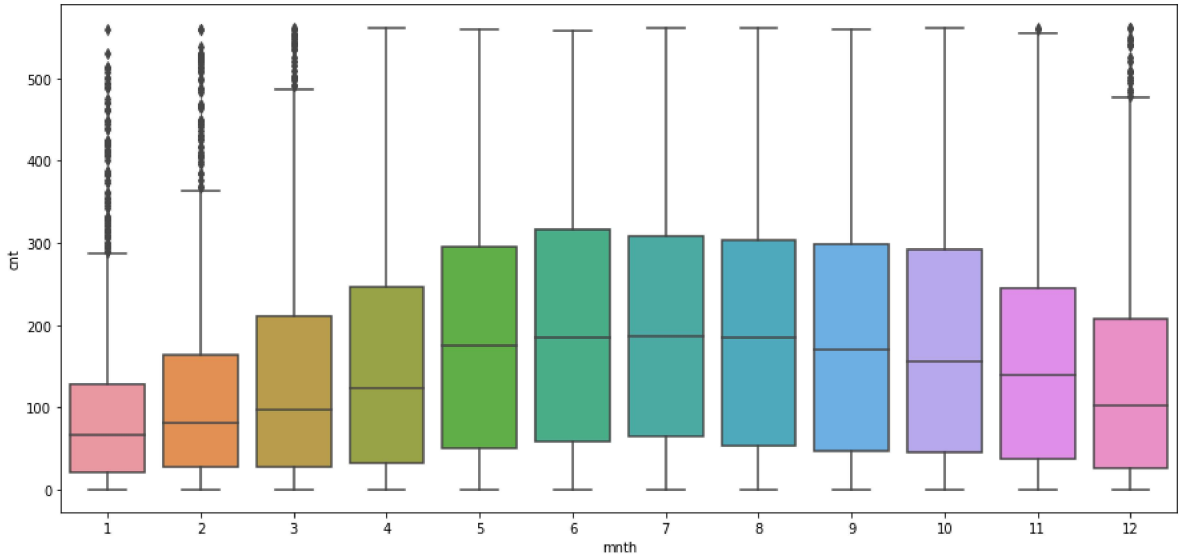
In [29]:

```
3.Make boxplot for cnt vs. month. Look at the median values. Any month(s) that stand out?

plt.figure(figsize=[15,7])
sns.boxplot("mnth","cnt",data=inp2)
```

```
Object `out` not found.

C:\Users\akhilesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpret
ation.
  warnings.warn(
```

Out[29]:

```
<AxesSubplot:xlabel='mnth', ylabel='cnt'>
```



In [30]:

```
inp2
```

Out[30]:

| | season | yr | mnth | hr | holiday | weekday | workingday | weathersit | temp | atemp | hum |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.81 |
| **1** | 1 | 0 | 1 | 1 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 |
| **2** | 1 | 0 | 1 | 2 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 |
| **3** | 1 | 0 | 1 | 3 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 |
| **4** | 1 | 0 | 1 | 4 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **17374** | 1 | 1 | 12 | 19 | 0 | 1 | 1 | 2 | 0.26 | 0.2576 | 0.60 |
| **17375** | 1 | 1 | 12 | 20 | 0 | 1 | 1 | 2 | 0.26 | 0.2576 | 0.60 |
| **17376** | 1 | 1 | 12 | 21 | 0 | 1 | 1 | 1 | 0.26 | 0.2576 | 0.60 |
| **17377** | 1 | 1 | 12 | 22 | 0 | 1 | 1 | 1 | 0.26 | 0.2727 | 0.56 |
| **17378** | 1 | 1 | 12 | 23 | 0 | 1 | 1 | 1 | 0.26 | 0.2727 | 0.65 |

16502 rows × 13 columns

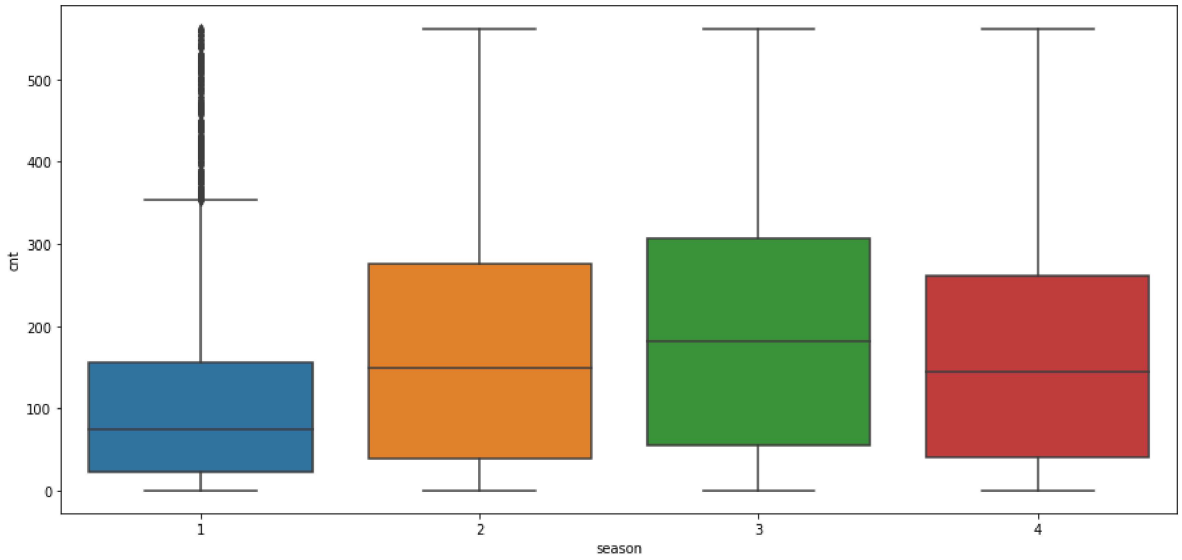
In [31]:

```
# 4.Make boxplot for cnt vs. season. Which season has the highest rides in general? Expecte

plt.figure(figsize=[15,7])
sns.boxplot("season","cnt",data=inp2)
```

```
C:\Users\akhilesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other a
rguments without an explicit keyword will result in an error or misinterpret
ation.
  warnings.warn(
```

Out[31]:

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

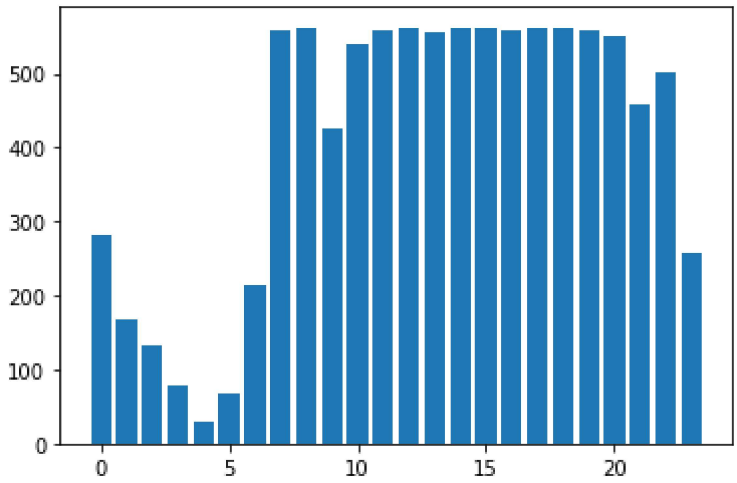


In [32]:

```
# 5.Make a bar plot with the median value of cnt for each hr. Does this paint a different p

plt.bar(x='hr',height='cnt',data=inp2)
```

Out[32]:

```
<BarContainer object of 16502 artists>
```




In [33]:

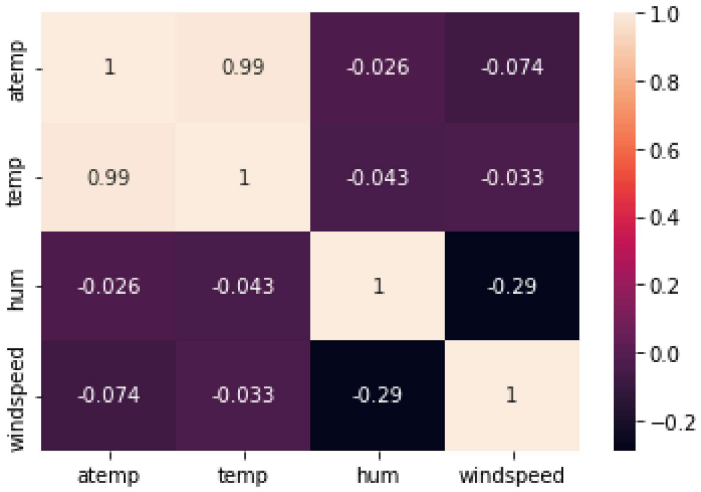
```
# 6.Make a correlation matrix for variables atemp, temp, hum, and windspeed. Which variable

col=['atemp', 'temp', 'hum', 'windspeed']
output=inp2[col].corr()
```

In [34]:

```
sns.heatmap(output,annot=True)
```

Out[34]:

```
<AxesSubplot:>
```



8. Data preprocessing

1.A few key considerations for the preprocessing:

There are plenty of categorical features. Since these categorical features can't be used in the predictive model, you need to convert to a suitable numerical representation. Instead of creating dozens of new dummy variables, try to club levels of categorical features wherever possible. For a feature with high number of categorical levels, you can club the values that are very similar in value for the target variable.

2.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

3.Treating hr column

Create new mapping: 0-5: 0, 11-15: 11; other values are untouched. Again, the bucketing is done in a way that hr values with similar levels of cnt are treated the same.

Get dummy columns for season, weathersit, weekday, mnth, and hr. You needn't club these further as the levels seem to have different values for the median cnt, when seen from the box plots.

In [35]:

```
# treating month column

inp2['mnth'] = inp2['mnth'].replace([5,6,7,8,9,10],5)
inp2['mnth'].unique()
```

Out[35]:

```
array([ 1,  2,  3,  4,  5, 11, 12], dtype=int64)
```

In [36]:

```
# treating hr column

inp2['hr'] = inp2['hr'].replace([0,1,2,3,4,5],0)
inp2['hr'].unique()
```

Out[36]:

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

In [37]:

```
inp2['hr'] = inp2['hr'].replace([11,12,13,14,15],11)
inp2['hr'].unique()
```

Out[37]:

```
array([ 0,  6,  7,  8,  9, 10, 11, 16, 17, 18, 19, 20, 21, 22, 23],
      dtype=int64)
```

In [38]:

```
inp2.head()
```

Out[38]:

| | season | yr | mnth | hr | holiday | weekday | workingday | weathersit | temp | atemp | hum | winds |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.81 | |
| **1** | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | |
| **2** | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | |
| **3** | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | |
| **4** | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | |

In [39]:

```
# dummy columns for season, weathersit, weekday, mnth, and hr

list1=['season', 'weathersit', 'weekday', 'mnth', 'hr']
inp2=pd.get_dummies(inp2,columns=list1)
```

In [40]:

```
inp2.shape
```

Out[40]:

```
(16502, 45)
```

In [41]:

```
inp2.columns
```

Out[41]:

```
Index(['yr', 'holiday', 'workingday', 'temp', 'atemp', 'hum', 'windspeed',
       'cnt', 'season_1', 'season_2', 'season_3', 'season_4', 'weathersit_
1',
       'weathersit_2', 'weathersit_3', 'weathersit_4', 'weekday_0',
       'weekday_1', 'weekday_2', 'weekday_3', 'weekday_4', 'weekday_5',
       'weekday_6', 'mnth_1', 'mnth_2', 'mnth_3', 'mnth_4', 'mnth_5',
       'mnth_11', 'mnth_12', 'hr_0', 'hr_6', 'hr_7', 'hr_8', 'hr_9', 'hr_1
0',
       'hr_11', 'hr_16', 'hr_17', 'hr_18', 'hr_19', 'hr_20', 'hr_21', 'hr_2
2',
       'hr_23'],
      dtype='object')
```

In [42]:

```
inp2.head()
```

Out[42]:

| | yr | holiday | workingday | temp | atemp | hum | windspeed | cnt | season_1 | season_2 | ... | hr_1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0.24 | 0.2879 | 0.81 | 0.0 | 16 | 1 | 0 | ... | |
| **1** | 0 | 0 | 0 | 0.22 | 0.2727 | 0.80 | 0.0 | 40 | 1 | 0 | ... | |
| **2** | 0 | 0 | 0 | 0.22 | 0.2727 | 0.80 | 0.0 | 32 | 1 | 0 | ... | |
| **3** | 0 | 0 | 0 | 0.24 | 0.2879 | 0.75 | 0.0 | 13 | 1 | 0 | ... | |
| **4** | 0 | 0 | 0 | 0.24 | 0.2879 | 0.75 | 0.0 | 1 | 1 | 0 | ... | |

5 rows × 45 columns

#9. Train test split: apply 70:30 split--

call the new dataframes df_train and df_test.

In [43]:

```
from sklearn.model_selection import train_test_split
df_train, df_test=train_test_split(inp2,test_size=0.30,random_state=32)
```

In [44]:

```
df_train.shape
```

Out[44]:

```
(11551, 45)
```

In [45]:

```
df_test.shape
```

Out[45]:

```
(4951, 45)
```

In [ ]:

```
# 10. Separate X and Y for df_train and df_test. For example, you should have X_train, y_tr
##     y_train should be the cnt column from inp3 and X_train should be all other columns.
```

In [46]:

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11551 entries, 1473 to 11140
Data columns (total 45 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   yr            11551 non-null  int64  
 1   holiday       11551 non-null  int64  
 2   workingday    11551 non-null  int64  
 3   temp          11551 non-null  float64
 4   atemp         11551 non-null  float64
 5   hum           11551 non-null  float64
 6   windspeed     11551 non-null  float64
 7   cnt           11551 non-null  int64  
 8   season_1      11551 non-null  uint8  
 9   season_2      11551 non-null  uint8  
 10  season_3      11551 non-null  uint8  
 11  season_4      11551 non-null  uint8  
 12  weathersit_1  11551 non-null  uint8  
 13  weathersit_2  11551 non-null  uint8  
 14  weathersit_3  11551 non-null  uint8  
 15  weathersit_4  11551 non-null  uint8  
 16  weekday_0     11551 non-null  uint8  
 17  weekday_1     11551 non-null  uint8  
 18  weekday_2     11551 non-null  uint8  
 19  weekday_3     11551 non-null  uint8  
 20  weekday_4     11551 non-null  uint8  
 21  weekday_5     11551 non-null  uint8  
 22  weekday_6     11551 non-null  uint8  
 23  mnth_1        11551 non-null  uint8  
 24  mnth_2        11551 non-null  uint8  
 25  mnth_3        11551 non-null  uint8  
 26  mnth_4        11551 non-null  uint8  
 27  mnth_5        11551 non-null  uint8  
 28  mnth_11       11551 non-null  uint8  
 29  mnth_12       11551 non-null  uint8  
 30  hr_0          11551 non-null  uint8  
 31  hr_6          11551 non-null  uint8  
 32  hr_7          11551 non-null  uint8  
 33  hr_8          11551 non-null  uint8  
 34  hr_9          11551 non-null  uint8  
 35  hr_10         11551 non-null  uint8  
 36  hr_11         11551 non-null  uint8  
 37  hr_16         11551 non-null  uint8  
 38  hr_17         11551 non-null  uint8  
 39  hr_18         11551 non-null  uint8  
 40  hr_19         11551 non-null  uint8  
 41  hr_20         11551 non-null  uint8  
 42  hr_21         11551 non-null  uint8  
 43  hr_22         11551 non-null  uint8  
 44  hr_23         11551 non-null  uint8  
dtypes: float64(4), int64(4), uint8(37)
memory usage: 1.2 MB
```

In [47]:

```
y_train=df_train.pop('cnt')

x_train=df_train
```

In [48]:

```
y_test=df_test.pop('cnt')

x_test=df_test
```

In [49]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(11551, 44)
(4951, 44)
(11551,)
(4951,)
```

In [50]:

```
# Apply linear regression project

from sklearn.linear_model import LinearRegression
regressor=LinearRegression()

regressor.fit(x_train, y_train)
```

Out[50]:

```
LinearRegression()
```

In [51]:

```
y_pred=regressor.predict(x_test)
y_pred
```

Out[51]:

```
array([219.75, 139.5 , -73.25, ..., 168.  , 220.5 ,  79.25])
```

In [52]:

```
# test score r2 score

from sklearn import metrics
print(metrics.r2_score(y_test,y_pred))
```

```
0.6612720880277326
```

In [ ]:

In [ ]:

In [ ]: