1. Problem Statement

Yulu provides the safest commute solution through a mobile app to enable shared, solo and sustainable commuting. They have contracted a consulting company to understand the factors on which the shared electric cycles depends on in the Indian market. They want to know which variables are significant in predicting the demand and how strongly those variables describe these demands. We will take a look at the data for more than 10,000 entries of electric cycle usage, and see if the weather, temperature, humidity, day being working or a holiday can affect the cycle demands.

In [ ]:
```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_shar:
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_shari
ng.csv?1642089089
To: /content/bike_sharing.csv?1642089089
100% 648k/648k [00:00<00:00, 8.43MB/s]
```

In [ ]:
```
import numpy as np, seaborn as sns, pandas as pd, math, matplotlib.pyplot as plt, scipy.stats
from scipy.stats import chi2, chi2_contingency, ttest_ind, f_oneway
import statsmodels
from statsmodels.stats.weightstats import ztest
```

In [ ]:
```
bikes = pd.read_csv('/content/bike_sharing.csv?1642089089')
```

In [ ]:
```
bikes.head() # first 5 rows
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	cou
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	

In [ ]:
```
bikes.shape # shape of dataset
```

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

In [ ]:
```
bikes.info()  # information about columns, their data types, count
```

```
<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
```

There are 10886 rows, 12 columns. There are some numerical variables that can be converted to categorical columns by changing the numbers to categories such as in season, holiday, workingday, and weather.

In [ ]:
```
bikes.season.replace([1,2,3,4,],['spring','summer','fall','winter'],inplace=True)
```

In [ ]:
```
bikes.season.value_counts()
```

Out[ ]:
```
winter    2734
summer    2733
fall      2733
spring    2686
Name: season, dtype: int64
```

In [ ]:
```
bikes.holiday.replace([0,1],['no_holiday','holiday'],inplace=True) # converting numerical to cat
```

In [ ]:
```
bikes.holiday.value_counts()
```

Out[ ]:
```
no_holiday    10575
holiday         311
Name: holiday, dtype: int64
```

There are only 311 holiday entries in the dataset, most entries are from non holiday days.

In [ ]:
```
bikes.workingday.replace([0,1],['not_working','working'],inplace=True) # converting numerical to
```

In [ ]:
```
bikes.workingday.value_counts()
```

Out[ ]:
```
working        7412
not_working    3474
Name: workingday, dtype: int64
```

There are almost double working day entries compared to weekend and holiday entries.

In [ ]:
```
bikes.weather.replace([1,2,3,4],['clear','misty','light_rain/snow','heavy_rain/snow'],inplace=Tru
```

In [ ]:
```
bikes.weather.value_counts()
```

```
Out[ ]: clear              7192
        misty              2834
        light_rain/snow     859
        heavy_rain/snow       1
        Name: weather, dtype: int64
```

Let's look at the numerical variables

2. Univariate analysis

```
In [ ]: bikes.describe() # statistical summary of all numerical variables
```

Out[ ]:

	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

Mean number of users who used a bike in an hour on a day is 191.

Missing values and outlier detection

```
In [ ]: bikes.isnull().sum()
```

```
Out[ ]: 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
```

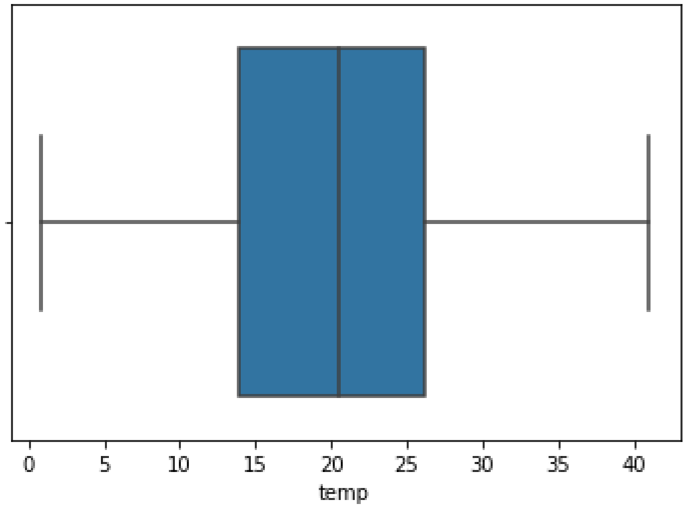
There are no missing values.

Let us look at any outliers present in the numerical variables like temp, atemp, humidity, windspeed, casual, registered.

```
In [ ]: sns.boxplot(bikes.temp) # looking at the spread of Purchase amount
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the follow
ing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will result in an error or misin
terpretation.
  FutureWarning
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfcfbe510>

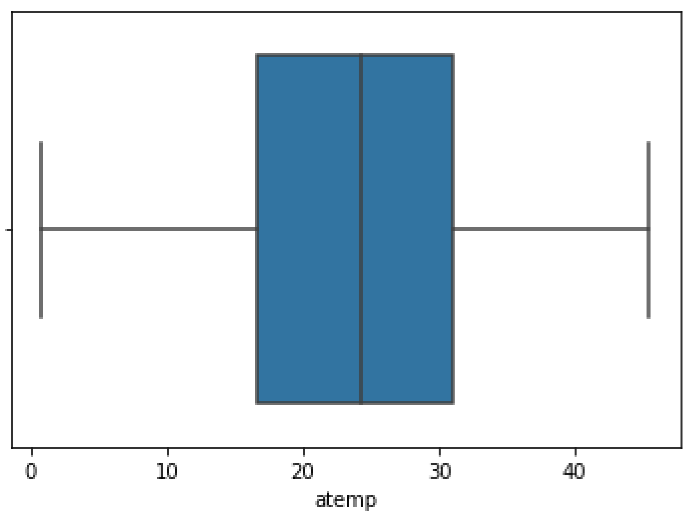


There are no outliers in temperature variable.

In [ ]:
```
sns.boxplot(bikes.atemp)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the follow
ing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will result in an error or misin
terpretation.
  FutureWarning
```

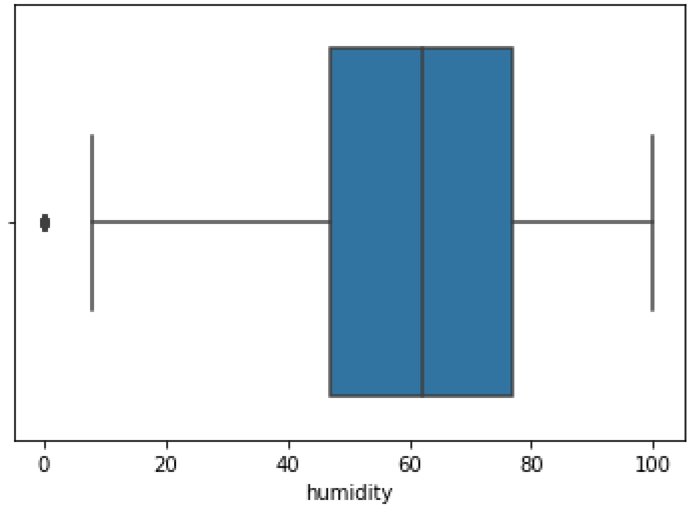
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfd4ca890>



In [ ]:
```
sns.boxplot(bikes.humidity)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the follow
ing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will result in an error or misin
terpretation.
  FutureWarning
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfceab790>

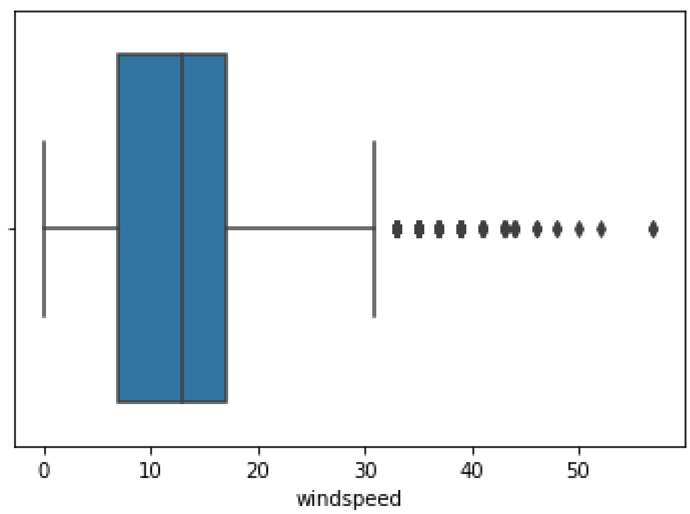


There is an outlier at 0 humidity, which is a reasonable number for a dry day so we would not discard it.

In [ ]:
```
sns.boxplot(bikes.windspeed)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the follow
ing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will result in an error or misin
terpretation.
  FutureWarning
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfce96e90>

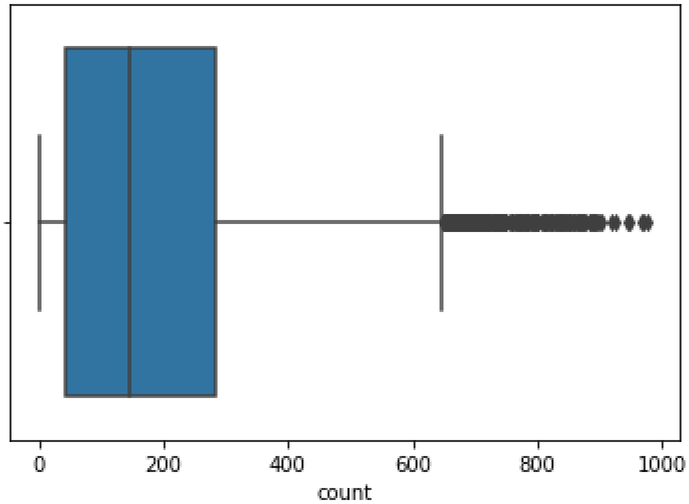


Windspeed seems to have a right skewed distribution which lots of large outliers, but they seem like high winds days and not errors so we would not discard them.

In [ ]:
```
sns.boxplot(bikes['count'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the follow
ing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will result in an error or misin
terpretation.
  FutureWarning
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfcdfd310>



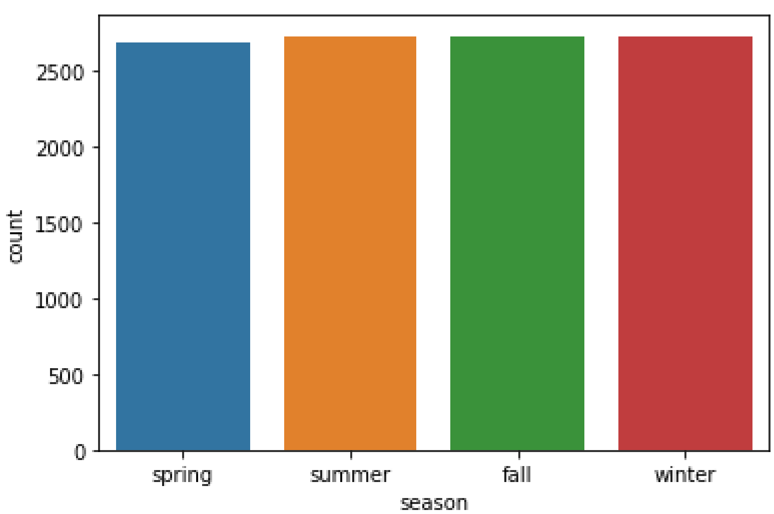
In [ ]:
```
iqr = scipy.stats.iqr(bikes['count']) # inter quartile range
q3 = np.percentile(bikes['count'],75) # third quartile
bikes['count'][bikes['count'] > (q3 +iqr*1.5)] # outlier points
```

Out[ ]:
```
6611     712
6634     676
6635     734
6649     662
6658     782
        ...
10678    724
10702    688
10726    679
10846    662
10870    678
Name: count, Length: 300, dtype: int64
```

There are 300 outliers all closely situated, and don't seem like error in measurement. So we would keep them.

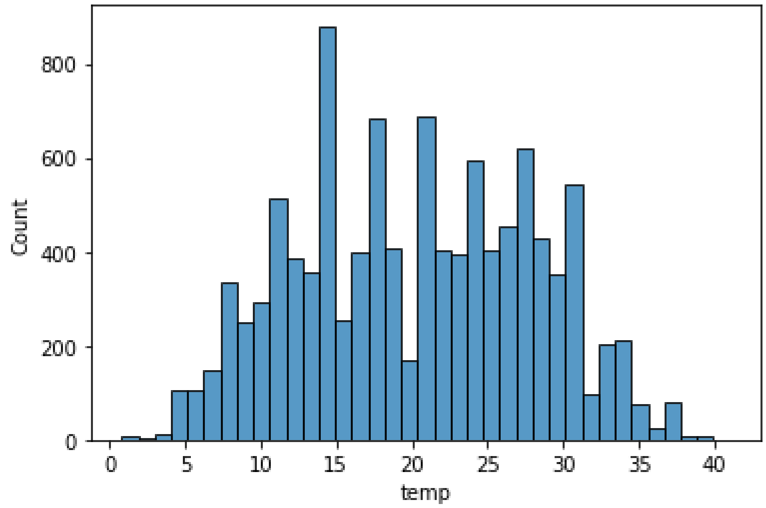
In [ ]:
```
sns.countplot(data=bikes, x = 'season') # all seasons have almost equal number of days
```

Out[ ]: `<matplotlib.axes._subplots.AxesSubplot at 0x7fbdfcd65a50>`



In [ ]:
```
sns.histplot(data=bikes, x='temp')
```

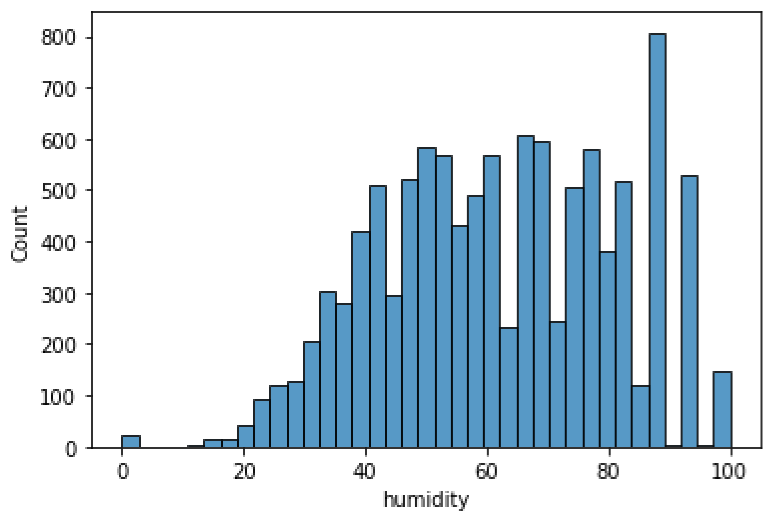
Out[ ]: `<matplotlib.axes._subplots.AxesSubplot at 0x7fbdfc5de710>`



This looks quite normally distributed.

In [ ]:
```
sns.histplot(data=bikes, x='humidity')
```

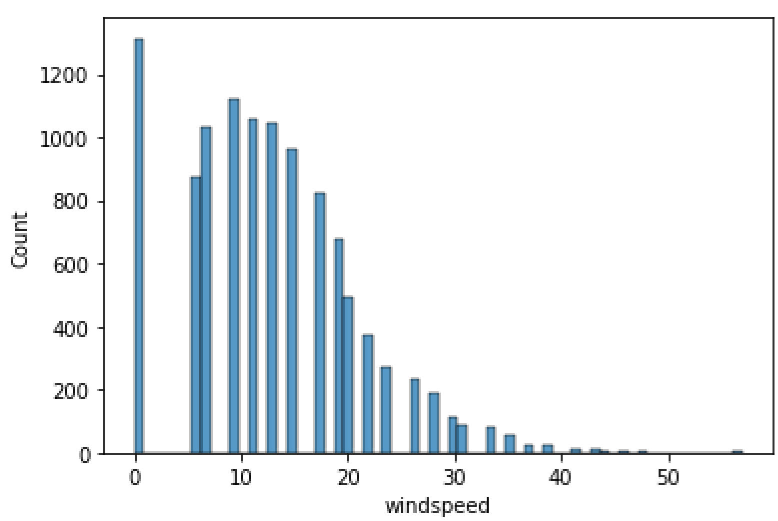
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfcd704d0>



Humidity looks slightly normal distribution but more higher humidity numbers are present.

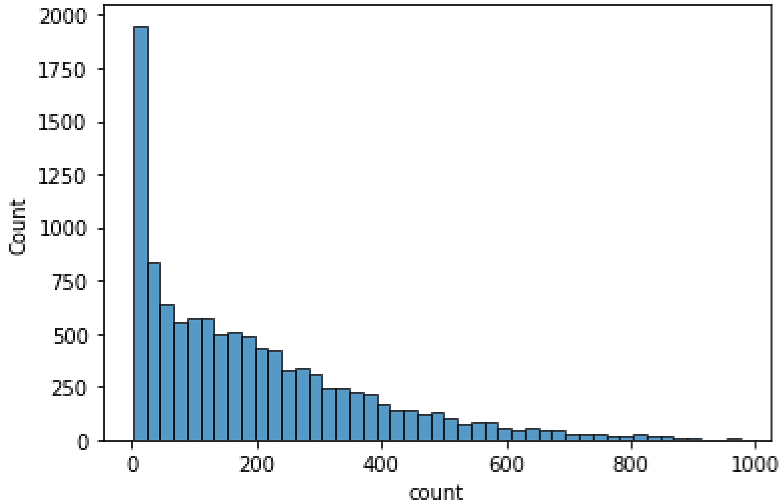
In [ ]:
```
sns.histplot(data=bikes, x='windspeed')
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfcc4f650>



In [ ]:
```
sns.histplot(data=bikes, x='count') # histogram of number of users per hour is almost linearly de
```

```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfcbd5490>
```

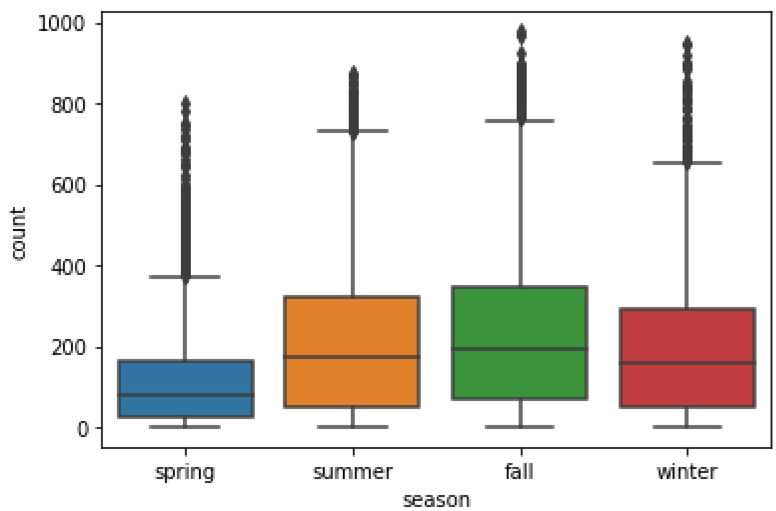


This looks slightly like an exponential distribution.

3. Bivariate analysis

```
In [ ]: sns.boxplot(data=bikes, x='season', y='count')
```

```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfcaa3650>
```

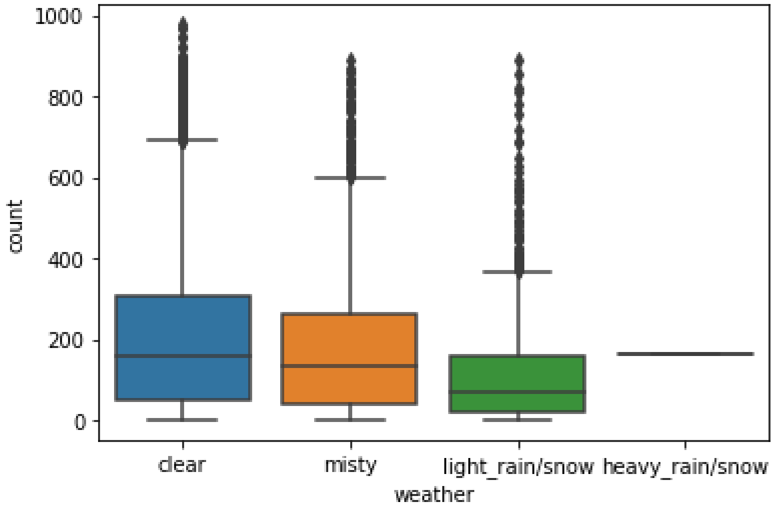


Mean number of users who used a bike in an hour on a day is 191.

The count distribution visually looks similar for summer, spring and highest for fall season for this sample, but less number of users per hour in spring than rest of the seasons in this sample dataset. For knowing how significantly different they are, we need to perform significance testing methods like ANOVA (analysis of variance).

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

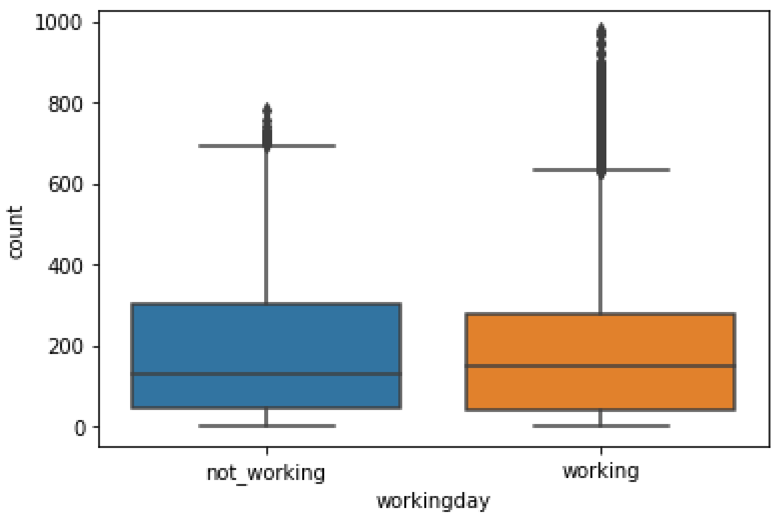
```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfca3fb50>
```



The median counts of users per hour is highest for clear sky, slightly less for misty weather and quite less for light rain/snow weather days. There are hardly any users for heavy rain/snow days. We would perform ANOVA to check statistical significance.

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

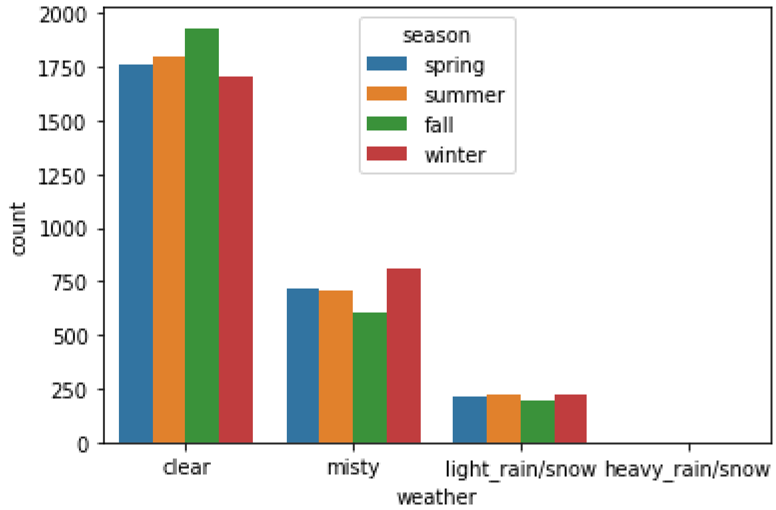
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfc992190>



The median number of users per hour on a working day and non working day are similar. To check statistical significance we need to perform 2-sample T-tests as the population standard deviation is unknown.

In [ ]:
```
sns.countplot(data=bikes, x='weather', hue='season')
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbdfd59ced0>



The number of hours recorded in every season is similar for light rain/snow weather and heavy rain/snow weather. However, they are different for clear sky weather, misty weather. We can check the statistical significance using chi-square to check the proportions.

4. Hypothesis testing

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

Some of the assumptions of the T-test are met such as random sampling, adequate number of samples. But some assumptions are not met such as Normality, Equal Variance.

We would continue doing the analysis even if some assumptions fail.

T-test Ho: the means for counts of users for working day and non working day entries are equal.

Ha: they are unequal

In [131…

```
a = bikes[bikes['workingday']=='not_working']['count']
b = bikes[bikes['workingday']=='working']['count']
print(np.array(a).var(),np.array(b).var()) # variances are different for the two groups
ttest_ind(a,b)
```

```
30171.346098942427 34040.69710674686
```

Out[131]:

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

T-test p-value 0.23 is larger than alpha=0.05. So we fail to reject the null hypothesis. Hence, similar to the visual analysis plots, working day has no statistically significant effect on the mean of number of cycles.

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

1. Weather

Ho : mean number of cycles rented per hour in different weather is similar

Ha : mean number of cycles rented are different for different weather

In [134…
```
a= bikes[bikes['weather']=='clear']['count']
b = bikes[bikes['weather']=='misty']['count']
c = bikes[bikes['weather']=='light_rain/snow']['count']
d = bikes[bikes['weather']=='heavy_rain/snow']['count']
print(np.array(a).var(),np.array(b).var(),np.array(c).var(),np.array(d).var()) # variances for th
f_oneway(a,b,c,d)
```

```
35323.8862270764 28337.246435435423 19182.418761290777 0.0
```

Out[134]:
```
F_onewayResult(statistic=65.53024112793271, pvalue=5.482069475935669e-42)
```

We assume that the variance for different population groups is same in order to perform F-test ANOVA, even though they differ largely.

As the p-value (5.5e-42) is very low, we reject the null hypothesis with a 95% confidence level (alpha=0.05). We can say that the mean number of cycles differs for different weather days, they are not independent.

2. Seasons

Ho : mean number of cycles rented per hour in different seasons is similar

Ha : mean number of cycles rented are different for different seasons

In [135…
```
a= bikes[bikes['season']=='spring']['count']
b = bikes[bikes['season']=='summer']['count']
c = bikes[bikes['season']=='fall']['count']
d = bikes[bikes['season']=='winter']['count']
print(np.array(a).var(),np.array(b).var(),np.array(c).var(),np.array(d).var()) # variances for th
f_oneway(a,b,c,d)
```

```
15687.725805298038 36853.522249306465 38854.295089130974 31538.180550642726
```

Out[135]:
```
F_onewayResult(statistic=236.94671081032106, pvalue=6.164843386499654e-149)
```

We assume that the variances being different doesn't affect the ANOVA F-test.

As the p-value is very low (6.2e-149), we reject the null hypothesis at 95% confidence level (alpha=0.05). We can say that the mean number of cycles differs for different seasons, they are not independent of each other.

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

In [ ]:
```
crosstable = pd.crosstab(index=bikes['weather'],columns=bikes['season'])
crosstable
```

Out[ ]:

season	fall	spring	summer	winter
weather				
clear	1930	1759	1801	1702
heavy_rain/snow	0	1	0	0
light_rain/snow	199	211	224	225
misty	604	715	708	807

There are counts/frequencies for different weather in different seasons. They can be assumed to be independent of each other. So, we create a null hypothesis for chi-square test.

Ho: the proportion of different weather days is equal for all seasons.

Ha: the proportions are not equal, weather days are dependent on seasons.

In [ ]:
```
chi2_contingency(crosstable)
```

Out[ ]:
```
(49.15865559689363,
 1.5499250736864862e-07,
 9,
 array([[1.80559765e+03, 1.77454639e+03, 1.80559765e+03, 1.80625831e+03],
        [2.51056403e-01, 2.46738931e-01, 2.51056403e-01, 2.51148264e-01],
        [2.15657450e+02, 2.11948742e+02, 2.15657450e+02, 2.15736359e+02],
        [7.11493845e+02, 6.99258130e+02, 7.11493845e+02, 7.11754180e+02]]))
```

As per the visual analysis, the number of hours recorded in every season is similar for light rain/snow weather and heavy rain/snow weather. However, they are different for clear sky weather, misty weather. The statistical test confirms this as p-value is very low (1.5e-7) so null hypothesis is rejected.

5. Inferences

Mean number of users who used a bike in an hour on a day is 191.

T-test p-value 0.23 is larger than alpha=0.05. So we fail to reject the null hypothesis. Hence, similar to the visual analysis plots, **working day has no statistically significant effect on the mean of number of cycles**.

The count distribution visually looks similar for summer, spring and highest for fall season for this sample, but less number of users per hour in spring than rest of the seasons in this sample dataset. For knowing how significantly different they are, we need to perform significance testing methods like ANOVA (analysis of variance).

As the ANOVA p-value is very low (6.2e-149), we reject the null hypothesis at 95% confidence level (alpha=0.05). We can say that the mean **number of cycles differs for different seasons**.

The median counts of users per hour is highest for clear sky, slightly less for misty weather and quite less for light rain/snow weather days. There are hardly any users for heavy rain/snow days. We would perform ANOVA to check statistical significance.

As the ANOVA p-value (4.9e-43) is very low, we reject the null hypothesis with a 95% confidence level (alpha=0.05). We can say that the mean **number of cycles differs for different weather days**.

As per the visual analysis, the number of hours recorded in every season is similar for light rain/snow weather and heavy rain/snow weather. However, they are different for clear sky weather, misty weather. The statistical chi-square test confirms this as p-value is very low (1.5e-7) so null hypothesis is rejected. **Weather and seasons are not independent.**

6. Recommendations

Weather and seasons are significant predictors for the demand for electric cycles as per this dataset

Clear sky and misty days have more demand so the fare can be increased for those days, and fare for light and heavy rain/snow days can be decreased.

Spring days can have reduced fares to increase number of users.

```
In [ ]:
```