Problem statement- To analyze the factors on which the demand of electric cycles depend on

Insights:

1. [Cell 3]: # Data has 10886 rows and 12 columns
2. [Cell 4]: # There are no null values in the data frame
3. [Cell 6]: # Below table summarizes the min, max etc for each numerical column
4. [Cell 7]: # Below table shows the data type of each column a. # No Null values
5. [Cell 9 to 19]: # Converting the values to appropriate values
6. [Cell 22]: # Removing the value for 'Heavy rain' as we have only one row of value
7. [Cell 25]: # Only 2.7 % data are outliners, hence removed
8. [Cell 27]:# Table below shows the min, max, mean, median, count for season variable
9. [Cell 28]: # Table below shows the min, max, mean, median, count for weather variable
10. [Cell 29]: # Table below shows the min, max, mean, median, count for season and weather variable
11. [Cell 30]: # Table below shows the min, max, mean, median, count for season and weather variable for temp, humidity and windspeed
12. [Cell 31 to 32]: # Unique values in season and weather column
13. [Cell 33]: # Below is the pairplot for all numerical variables with hue as season Temperature increases in the following order Spring, Winter and Fall
14. [Cell 34]: # Below is the pairplot for all numerical variables with hue as weather 'Clear' kde plot overlaps 'Mist' and 'Light Rain'
15. [Cell 38]: # Below graph shows the Count plot at each hour. We have almost the same count at each hour. Slightly less count at 8th,17th and 18th hour
16. [Cell 40]: # Bar plot shows the summ of 'count' at each hour Higher number of booking take place between 17th hour and 20th hour. Very booking between 0th hour to 6th hour
17. [Cell 41]: # Graph shows the total 'count' of booking in each year We almost same number of booking on each year
18. [Cell 42]: # Below graph shows the count of booking on each month and they are roughly the same
19. [Cell 43]: # Below graph shows the count of booking on each date and they are roughly the same
20. [Cell 44]: # Below table shows the conditional probality of 'count' across weather and season
21. [Cell 45]: # Below is the barplot for 'count' across weather. There are lot of outliners in clear weather and season spring
22. [Cell 46]: # Below plot concludes that clear weather has highest median
23. [Cell 47]: Barplot # Medians of count on Holiday and non holiday days are almost the same
24. [Cell 48]: Barplot # Temperature is Fall are higher followed by summer, spring and winter
25. [Cell 49]: Barplot # Humidity is higher when weather is Light rain followed by Mist and clear
26. [Cell 51]: # Below plot shows that more booking are taken on 'Non-Holiday','Working day' and when weather is clean
27. [Cell 52]: # Below bar plot concludes that there is no difference between the 4 season for the total 'count' of bookings
28. [Cell 53]: # Below kde plot concludes that Casual count and registered count are more in spring and winter
29. [Cell 54]: # Below kde plot concludes that Casual count and registered count are more between 0 to 100
30. [Cell 55]: # Below is the kde plot for all numerical variables with hue as workingday and non-working day
31. [Cell 56]: # Below table shows the sum of count on each season Season Fall has the highest count followed by summer, spring and winter
32. [Cell 57]: # Below table shows the sum of count on each weather Season Fall has the highest count followed by clear, mist and light rain
33. [Cell 58]: # Below is the barplot for count with respect to weather
34. [Cell 59]: # Below table shows the sum of count on each holiday More booking happen on non-holiday
35. [Cell 60]: # Below is the bar plot for count with respect to holiday

36. [Cell 61]: # 'Count' is split with respect to percentiles In spring season the count of cycle takers are more for <26, '26,96' when compared with other seasons But it is the least when count is [183,306] and [306,647]
37. [Cell 62]: # 'Count' is split with respect to percentiles In mist weather the count of cycle takers are almost the same across different count range
38. [Cell 63]: # 'Count' is split with respect to percentiles Below is the count plot for 'holiday'
39. [Cell 64]: # 'Count' is split with respect to percentiles Below is the count plot for 'Working day'
40. [Cell 65]: # Below table shows the sum of count on each working day Bookings are doubled on a workingday when compared with non-workingday
41. [Cell 66]: # Below table shows the sum of 'count' with respect to season and weather
42. [Cell 67]: # Below is the bar plot for weather and season
43. [Cell 68]: # Below is the pair plot for variables humidity, count, windspeed and temp.With hue as season
44. [Cell 69]: # Below is the pair plot for variables humidity, count, windspeed and temp. With hue as weather
45. [Cell 70]: # Below table shows the correlation for numerical variables
46. 42 correlation between count and hour
47. 388096 correlation between count and temp
48. [Cell 71-73]: # spearman rank correlation between temp and count is 0.392 spearman rank correlation between humidity and count is -0.137 spearman rank correlation between windspeed and count is 0.13734
49. [Cell 74]: #Below is the heatmap for coefficient correlation for numerical variables
50. [Cell 75,76]: # Below is the heat map for count for season and weather Below is the heat map for count for holiday and working day
51. [Cell 77]: # Below is the distribution plot for 'count'. This distribution is right skewed and not a normal distribution 2 Sample T Test • ## Assumptions • # 1. Sample mean and variance should be finite • # 2. Sample follow normal dist Null and alternate hypothesis for KS test • #Ho= Sample follow normal distribution • #Ha= Sample does not follow normal distribution, • Significance value= 5% • p-value should be greater than 0.05 for the distribution to be normal T Test for Holiday and count

++ • Null hypothesis: The mean of count is equal on a Non-holiday and on a holiday • Alternate hypothesus:The mean of count not equal on a Non-holiday and on a holiday • The Significance value= 5% • p-value <= 0.05, Then we have enough evidence to reject Null hypothesis • p-value > 0.05, We fail to reject null hypothesis • Sample mean and standard deviation are finite and follow closely normal distribution • Result: As pvalue > 0.05 we conclude the mean of count is equal on a Non-holiday and on a holiday

T Test for Working day and count • Null hypothesis: The mean of count is equal on a Non-working day and on a working day • Alternate hypothesis:The mean of count not equal on a Non-working day and on a working day • Significance value= 5% • p-value <= 0.05, Then we have enough evidance to reject Null hypothesis • p-value > 0.05, We fail to reject null hypothesis • Sample mean and standard deviation are finite and closely follow normal distribution • Result: As pvalue > 0.05 we conclude the mean of count is equal on a Non-working day and on a working day ANOVA test for testing means of count for different seasons

o #Sample sizes are almost equal o [Cell 81]:# Outliners are removed as they are only 4.7 % and 0.52 % o [Cell 85]: Above plots are not normal distribution o [Cell 86]: # Converting data to normal distribution with help of boxcox transformations o [Cell 87]: # Means of the above data are almost equal o [Cell 88]:# Variances of the above data are almost equal o [Cell 89]: # Chi2 test to check if count is independent with respect to season o [Cell 90]: # As p-Value is > 0.05 we conclude that the H0 is true. (Values are independent) o [Cell 91]: # KS test to check if they are normal # P-values are not close to 0.05 and are lesser and they are not gaussian distribution, #but still looks like normal distribution o [Cell 94]: # Below is the QQ plot for the data. Plot looks like normal distribution o [Cell 94]: F_onewayResult(statistic=524.6205973018104, pvalue=1.4785115164657819e-279) o [Cell 94]:
++
Null hypothesis: The mean of count is equal for spring ,winter, fall and summer Alternate hypothesus:The mean of count not equal for spring ,winter, fall and summer Significance value= 5% p-value <= 0.05, Then we have enough evidence to reject Null hypothesis p-value > 0.05, We fail to reject null hypothesis Sample

mean and variance are finite and closely follow normal distribution Result: As pvalue < 0.05 we conclude the mean of count not equal for spring ,winter, fall and summer T stat is shown above ANOVA test for testing means of count for differents weathers

[cell 98]: # Sample sizes are not equal hence we do a random choice of 840 [Cell 101] : # Outliners are removed as they are 1.08 % and 5.88 % [cell 102]: Above plots are not normal distribution and are right skewed [Cell 103]: # Applying boxcox transformations [Cell 104]: # Variances of the above data are almost equal [Cell 105]: # Means of the above data are almost equal [Cell 106]: # KS test to check if they are normal P-values are not close to 0.05 and are lesser and they are not gaussian distribution, #but still looks like normal distribution [Cell 108]: # Chi2 test to check if count is independent with respect to weather [Cell 110]: # As p-Value is > 0.05 we conclude that the H0 is true. (Values are independent) [Cell 111]: # Below is the QQ plot for the data. Plot looks like normal distribution Cell 112
(++
F_onewayResult(statistic=165.43626508856718, pvalue=3.143539828592585e-68)

• Null hypothesis: The mean of count is equal for clear, Mist and Light rain • Alternate hypothesis: The mean of count not equal for clear, Mist and Light rain • Significance value= 5% • p-value <= 0.05, Then we have enough evidence to reject Null hypothesis • p-value > 0.05, We fail to reject null hypothesis • Sample mean and variance are finite and closely follow normal distribution • Result: As pvalue < 0.05 we conclude the mean of count not equal for clear, Mist and Light rain • T stat is shown above Chi square (Test of Independence)

[Cell 113]: # To check the total count of weather and season are independent [Cell 118]: chi2, p, dof, ex= (8409.547609716874, 0.0, 6, array([[386711.04478819, 28181.4160831 , 136381.53912871], [211146.61270112, 15387.22678668, 74465.1605122], [361828.6315605 , 26368.11995471, 127606.24848479], [345061.71095019, 25146.23717551, 121693.0518743]])) [Cell 118]:
++
• Null hypothesis: The count is independent of season and weather • Alternate hypothesis: The count is not independent of season and weather • Significance value= 5% • p-value <= 0.05, Then we have enough evidence to reject Null hypothesis • p-value > 0.05, We fail to reject null hypothesis • Result: As p value < 0.05 we conclude that count is not independent of season and weather • T stat is shown above

ANOVA 2-way Test

• Weather: p-value = 4.503016e-45 • Season: p-value = 3.448572e-138 • Weather*Season: p-value = 4.338083e-02

• Hypothesis for Weather: • Ho : Weather has no statistically significant effect on eletrical cycles count • Ha : Weather has statistically significant effect on eletrical cycles count

• As p-Value <0.05, we conclude that Weather has statistically significant effect on eletrical cycles count

• Hypothesis for Season: • Ho : Season has no statistically significant effect on eletrical cycles count • Ha : Season has statistically significant effect on eletrical cycles count

• As p-Value <0.05, we conclude that season has statistically significant effect on eletrical cycles count

[Cell 122]: Confidence interval using Bootstrap : Confidence interval for Spring season group is [108.30759647808168,117.31071562382914] with 95.0% confidence Confidence interval for Summer season group is [189.48500379650721,202.2044608959757] with 95.0% confidence Confidence interval for Fall season group is [204.40768055024836,216.98775315246465] with 95.0% confidence Confidence interval for Winter season group is [178.67913696060037,190.383574108818] with 95.0% confidence

[Cell 123]: Hist plot for each season group. Spring confidence interval does not coincide with any other group Summer, winter and Fall confidence interval are overlapping.

[Cell 124]: Confidence interval using Bootstrap : Confidence interval for Clear weather group is [183.52671213208902,191.0960624551328] with 95.0% confidence Confidence interval for Mist weather group is [160.68084837545126,171.54016245487367] with 95.0% confidence Confidence interval for Light Rain weather group is [103.76114705882354,120.22708823529412] with 95.0% confidence

[Cell 125]: Hist plot for each weather group. clear ,mist and light rain confidence interval are not overlapping.

[Cell 126]: Confidence interval using Bootstrap : Confidence interval for Non-Holiday holiday group is [172.4988589918256,178.53687232386142] with 95.0% confidence Confidence interval for Holiday holiday group is [164.44959546925566,200.85833333333332] with 95.0% confidence

[Cell 127]: Hist plot for each holiday group. Holiday and non-holiday confidence interval are overlapping completely.

[Cell 128]: Confidence interval using Bootstrap : Confidence interval for Non-Workingday workingday group is [175.85114598540147,186.84342335766422] with 95.0% confidence Confidence interval for Workingday workingday group is [169.48717877094973,176.50942737430168] with 95.0% confidence

[Cell 129]: Hist plot for each working day group. working day and non-working day confidence interval are not overlapping.¬¬

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from  scipy import stats
from statsmodels.stats.weightstats import ztest as ztest
from statsmodels.formula.api import ols
import statsmodels.api as sm
import pingouin as pg
import datetime
import math
import scipy
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df=pd.read_csv('bike_sharing.csv')
```

In [3]:

```
df.shape
# Data has 10886 rows and 12 columns
```

Out[3]:

```
(10886, 12)
```

In [4]:

```
df.isnull().sum()
# There are no null values in the data frame
```

Out[4]:

```
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 [5]:

```
df.describe(include=object)
```

Out[5]:

	datetime
count	10886
unique	10886
top	2011-01-01 00:00:00
freq	1

In [6]:

```
df.describe()
# Below table summarizes the min, max etc for each numerical column
```

Out[6]:

	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 [7]:

```
df.info()
# Below table showss the data type of each column
# No Null values
```

```
<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 [8]:

```
df
```

Out[8]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	c
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	
...	...	...	...	...	...	...	...	...	...	
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	

10886 rows × 12 columns

In [9]:

```
df['time']=pd.to_datetime(df['datetime']).dt.time
df['date']=pd.to_datetime(df['datetime']).dt.date
# Converting to date time object
```

In [10]:

```
df.drop('datetime',axis=1,inplace=True)
```

In [11]:

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

Out[11]:

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

In [12]:

```
df.head(10)
```

Out[12]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registere
0	1	0	0	1	9.84	14.395	81	0.0000	3	
1	1	0	0	1	9.02	13.635	80	0.0000	8	
2	1	0	0	1	9.02	13.635	80	0.0000	5	
3	1	0	0	1	9.84	14.395	75	0.0000	3	
4	1	0	0	1	9.84	14.395	75	0.0000	0	
5	1	0	0	2	9.84	12.880	75	6.0032	0	
6	1	0	0	1	9.02	13.635	80	0.0000	2	
7	1	0	0	1	8.20	12.880	86	0.0000	1	
8	1	0	0	1	9.84	14.395	75	0.0000	1	
9	1	0	0	1	13.12	17.425	76	0.0000	8	

In [13]:

```
df['season']=['Spring' if i==1 else i for i in df['season']]
df['season']=['Summer' if i==2 else i for i in df['season']]
df['season']=['Fall' if i==3 else i for i in df['season']]
df['season']=['Winter' if i==4 else i for i in df['season']]
# Converting the values to appropriate values
```

In [14]:

```
df['holiday'].value_counts()
```

Out[14]:

```
0    10575
1      311
Name: holiday, dtype: int64
```

In [15]:

```
df['holiday']=['Holiday' if i==1 else i for i in df['holiday']]
df['holiday']=['Non-Holiday' if i==0 else i for i in df['holiday']]
#Converting the values to appropriate values
```

In [16]:

```
df['holiday'].value_counts()
```

Out[16]:

```
Non-Holiday    10575
Holiday          311
Name: holiday, dtype: int64
```

In [17]:

```
df['workingday']=['Workingday' if i==1 else i for i in df['workingday']]
df['workingday']=['Non-Workingday' if i==0 else i for i in df['workingday']]
# Converting the values to appropriate values
```

In [18]:

```
df['weather']=['Clear' if i==1 else i for i in df['weather']]
df['weather']=['Mist' if i==2 else i for i in df['weather']]
df['weather']=['Light Rain' if i==3 else i for i in df['weather']]
df['weather']=['Heavy Rain' if i==4 else i for i in df['weather']]
# Converting the values to appropriate values
```

In [19]:

```
df
```

Out[19]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	reg
0	Spring	Non-Holiday	Non-Workingday	Clear	9.84	14.395	81	0.0000	3	
1	Spring	Non-Holiday	Non-Workingday	Clear	9.02	13.635	80	0.0000	8	
2	Spring	Non-Holiday	Non-Workingday	Clear	9.02	13.635	80	0.0000	5	
3	Spring	Non-Holiday	Non-Workingday	Clear	9.84	14.395	75	0.0000	3	
4	Spring	Non-Holiday	Non-Workingday	Clear	9.84	14.395	75	0.0000	0	
...	...	...	...	...	...	...	...	...	...	...
10881	Winter	Non-Holiday	Workingday	Clear	15.58	19.695	50	26.0027	7	
10882	Winter	Non-Holiday	Workingday	Clear	14.76	17.425	57	15.0013	10	
10883	Winter	Non-Holiday	Workingday	Clear	13.94	15.910	61	15.0013	4	
10884	Winter	Non-Holiday	Workingday	Clear	13.94	17.425	61	6.0032	12	
10885	Winter	Non-Holiday	Workingday	Clear	13.12	16.665	66	8.9981	4	

10886 rows × 13 columns

In [20]:

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

Out[20]:

```
Clear         7192
Mist          2834
Light Rain     859
Heavy Rain       1
Name: weather, dtype: int64
```

In [21]:

```
df.loc[df['weather']=='Heavy Rain'].index
```

Out[21]:

```
Int64Index([5631], dtype='int64')
```

In [22]:

```
df.drop(5631,axis=0,inplace=True)
# Removing the value for 'Heavy rain' as we have only one row of value
```

In [23]:

```
df
```

Out[23]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	reg
0	Spring	Non-Holiday	Non-Workingday	Clear	9.84	14.395	81	0.0000	3	
1	Spring	Non-Holiday	Non-Workingday	Clear	9.02	13.635	80	0.0000	8	
2	Spring	Non-Holiday	Non-Workingday	Clear	9.02	13.635	80	0.0000	5	
3	Spring	Non-Holiday	Non-Workingday	Clear	9.84	14.395	75	0.0000	3	
4	Spring	Non-Holiday	Non-Workingday	Clear	9.84	14.395	75	0.0000	0	
...	...	...	...	...	...	...	...	...	...	...
10881	Winter	Non-Holiday	Workingday	Clear	15.58	19.695	50	26.0027	7	
10882	Winter	Non-Holiday	Workingday	Clear	14.76	17.425	57	15.0013	10	
10883	Winter	Non-Holiday	Workingday	Clear	13.94	15.910	61	15.0013	4	
10884	Winter	Non-Holiday	Workingday	Clear	13.94	17.425	61	6.0032	12	
10885	Winter	Non-Holiday	Workingday	Clear	13.12	16.665	66	8.9981	4	

10885 rows × 13 columns

In [24]:

```
def outliners(x,col):
    Q1 = np.percentile(x[col], 25)
    Q3 = np.percentile(x[col], 75)
    IQR = Q3 - Q1
    upper = Q3 +1.5*IQR
    lower = Q1 - 1.5*IQR
    #print(upper,lower)
    ls=list(x.iloc[((x[col]<lower) | (x[col]>upper)).values].index)
    return ls
```

In [25]:

```
len(outliners(df,'count'))*100/len(df)
# Only 2.7 % data are outliners, hence removed
```

Out[25]:

```
2.756086357372531
```

In [26]:

```
df.drop(outliners(df,'count'),axis=0,inplace=True)
# Removing the outliners for 'count' column
```

In [27]:

```
df.groupby('season')[['count','casual','registered']].aggregate({'count':['min','max','mean
# Table below shows the min, max, mean, median, count for season variable
```

Out[27]:

	count					casual				
	min	max	mean	median	count	min	max	mean	median	count	min	m
season												
Fall	1	647	210.651127	185.0	2617	0	289	49.779519	34.0	2617	0	6
Spring	1	644	112.775946	78.0	2669	0	240	14.512552	5.0	2669	0	6
Summer	1	647	195.824981	165.0	2634	0	355	44.830296	27.0	2634	0	6
Winter	1	647	184.578236	154.0	2665	0	294	27.221013	13.0	2665	1	6

In [28]:

```
df.groupby('weather')[['count','casual','registered']].aggregate({'count':['min','max','mea
# Table below shows the min, max, mean, median, count for weather variable
```

Out[28]:

	count					casual				
	min	max	mean	median	count	min	max	mean	median	count	min	m
weather												
Clear	1	647	187.329218	153.0	6965	0	355	37.787365	19.0	6965	0	6
Light Rain	1	646	111.862353	70.5	850	0	263	16.896471	5.5	850	0	6
Mist	1	646	166.117690	130.0	2770	0	283	29.632852	15.0	2770	0	6

In [29]:

```
df.groupby(['season','weather'])[['count','casual','registered']].aggregate({'count':['min'
# Table below shows the min, max, mean, median, count for season and weather variable
```

Out[29]:

		count					casua				
		min	max	mean	median	count	min	max	mean	median	count
season	weather										
Fall	Clear	1	647	218.449267	196.0	1843	0	289	52.404775	39.0	1843
	Light Rain	1	646	142.989744	110.0	195	0	263	31.861538	13.0	195
	Mist	2	643	208.616580	187.0	579	0	257	47.457686	31.0	579
Spring	Clear	1	623	121.780963	88.0	1744	0	240	16.209862	6.0	1744
	Light Rain	1	520	61.227488	34.0	211	0	31	3.526066	1.0	211
	Mist	1	644	106.014006	76.0	714	0	226	13.613445	5.0	714
Summer	Clear	1	647	213.482859	185.0	1721	0	355	49.798373	31.0	1721
	Light Rain	1	627	120.955157	70.0	223	0	241	20.260090	8.0	223
	Mist	1	642	175.979710	151.0	690	0	283	40.379710	25.5	690
Winter	Clear	1	647	194.541943	168.0	1657	0	294	31.764635	16.0	1657
	Light Rain	1	613	123.565611	94.0	221	0	131	13.063348	6.0	221
	Mist	1	646	180.733164	153.0	787	0	269	21.630241	12.0	787

In [30]:

```
df.groupby(['season','weather'])[['temp','humidity','windspeed']].aggregate({'temp':['min',
# Table below shows the min, max, mean, median, count for season and weather variable for t
```

Out[30]:

		temp					humidi				
		min	max	mean	median	count	min	max	mean	median	cou
season	weather										
Fall	Clear	15.58	41.00	29.188974	29.52	1843	17	100	61.039609	62.0	184
	Light Rain	19.68	37.72	26.710974	26.24	195	35	100	82.400000	84.0	19
	Mist	18.86	39.36	27.963558	27.88	579	24	100	70.037997	70.0	57
Spring	Clear	0.82	29.52	12.421778	11.48	1744	8	100	50.122133	49.0	174
	Light Rain	3.28	22.96	12.152322	12.30	211	0	100	76.473934	87.0	21
	Mist	3.28	29.52	12.610084	12.30	714	0	100	65.543417	65.0	71
Summer	Clear	9.84	38.54	23.033376	22.96	1721	16	100	55.522952	55.0	172
	Light Rain	9.84	33.62	20.985381	21.32	223	35	100	81.273543	83.0	22
	Mist	9.84	34.44	22.422841	22.14	690	21	100	69.639130	73.0	69
Winter	Clear	5.74	30.34	16.126337	15.58	1657	16	100	61.178636	60.0	165
	Light Rain	9.84	26.24	18.555747	18.86	221	48	100	85.660633	88.0	22
	Mist	6.56	29.52	16.928259	15.58	787	28	100	71.787802	72.0	78

In [31]:

```
df['season'].value_counts()
# Unique values in season column
```

Out[31]:

```
Spring    2669
Winter    2665
Summer    2634
Fall      2617
Name: season, dtype: int64
```

In [32]:

```
df['weather'].value_counts()
# Unique values in weather column
```

Out[32]:

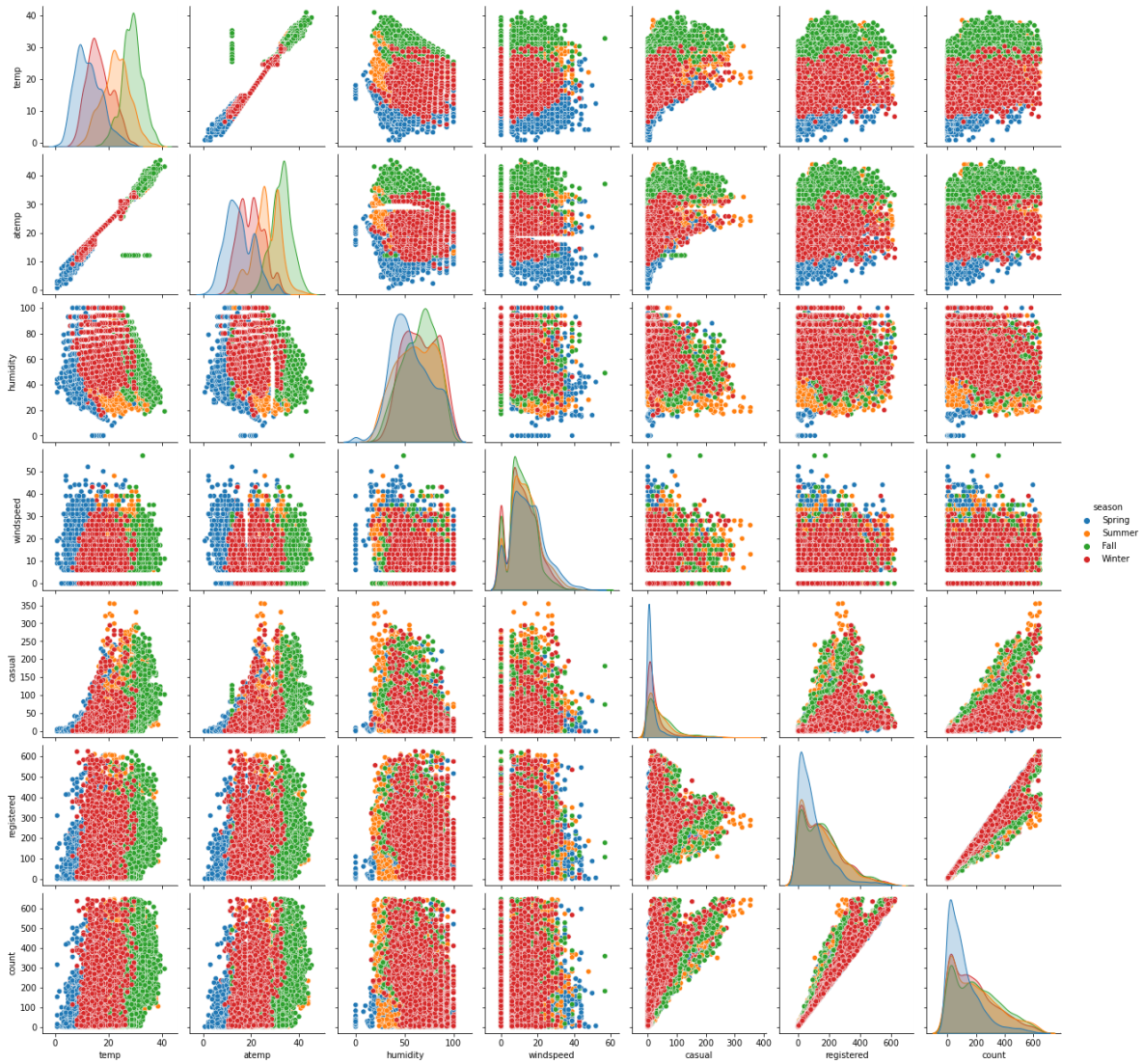
```
Clear         6965
Mist          2770
Light Rain     850
Name: weather, dtype: int64
```

In [33]:

```
sns.pairplot(data=df,hue='season')
# Below is the pairplot for all numerical variables with hue as season
# Temperature increases in the following order Spring, Winter and Fall
```

Out[33]:

```
<seaborn.axisgrid.PairGrid at 0x2253fe97430>
```

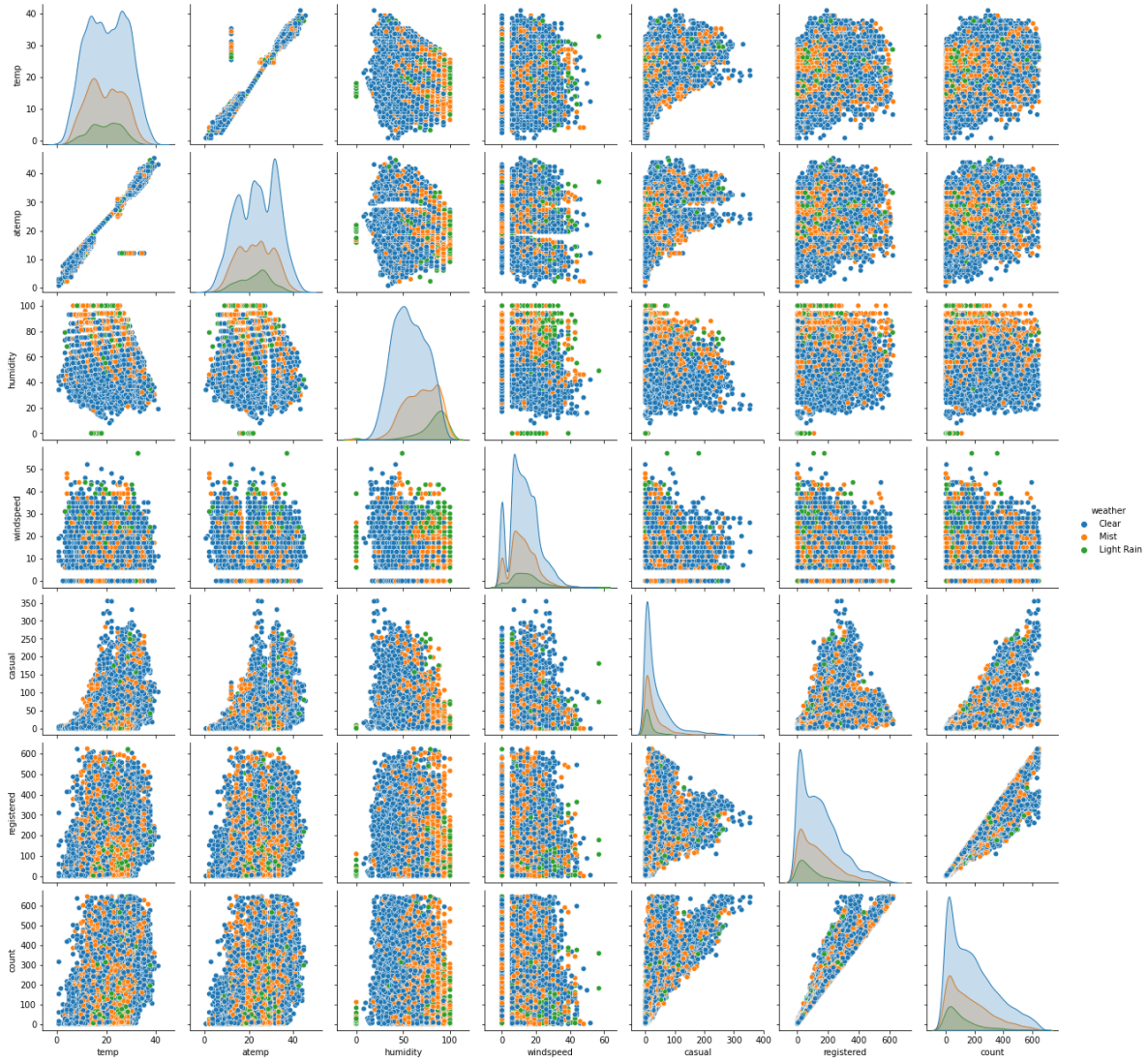


In [34]:

```
sns.pairplot(data=df,hue='weather')
# Below is the pairplot for all numerical variables with hue as weather
# 'Clear' kde plot overlaps 'Mist' and 'Light Rain'
```

Out[34]:

```
<seaborn.axisgrid.PairGrid at 0x22545780a90>
```

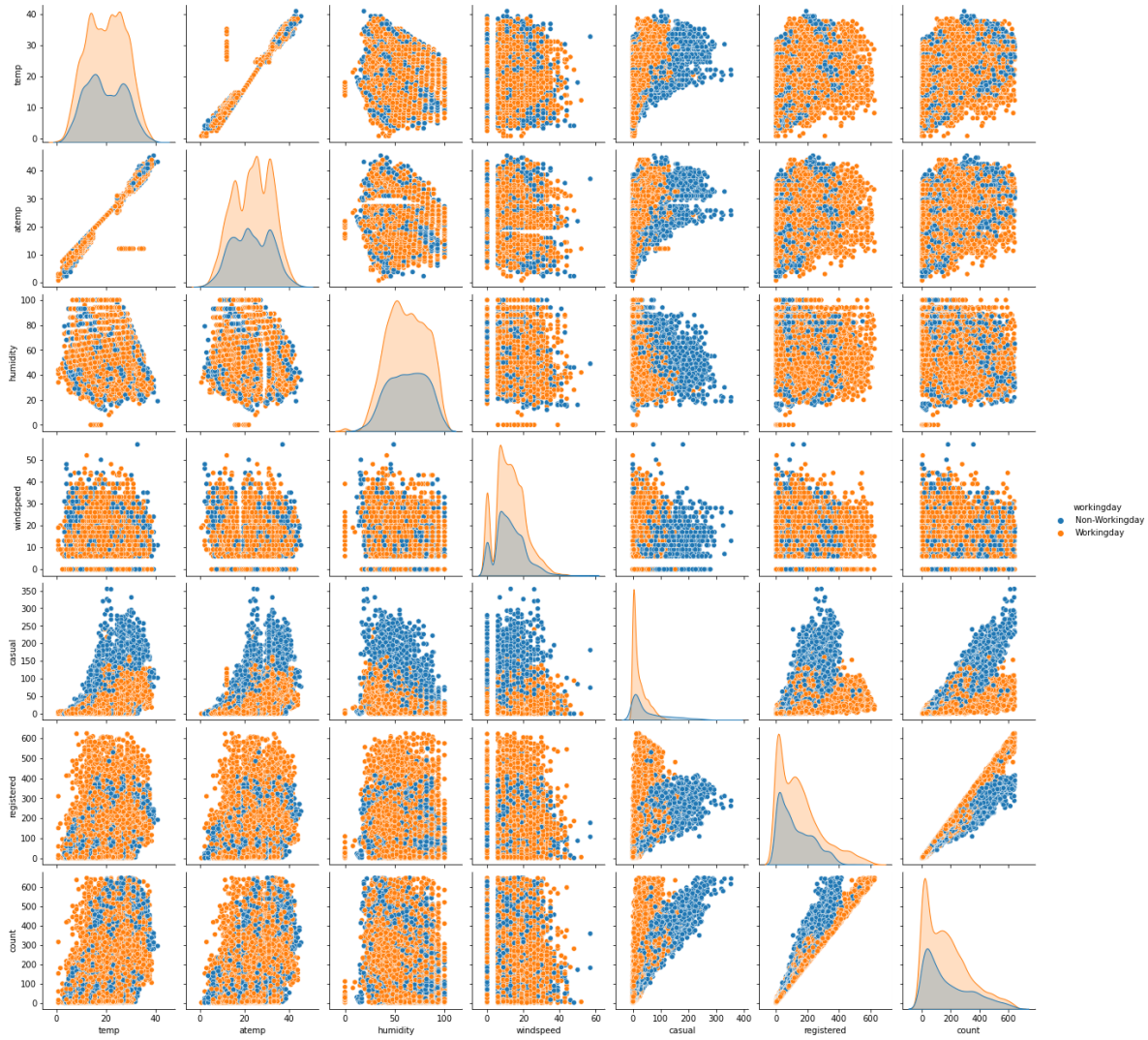


In [35]:

```
sns.pairplot(data=df,hue='workingday')
```

Out[35]:

```
<seaborn.axisgrid.PairGrid at 0x22549f7fb20>
```



In [36]:

```
ls=[]
for i in df['time']:
    ls.append(i.hour)
```

In [37]:

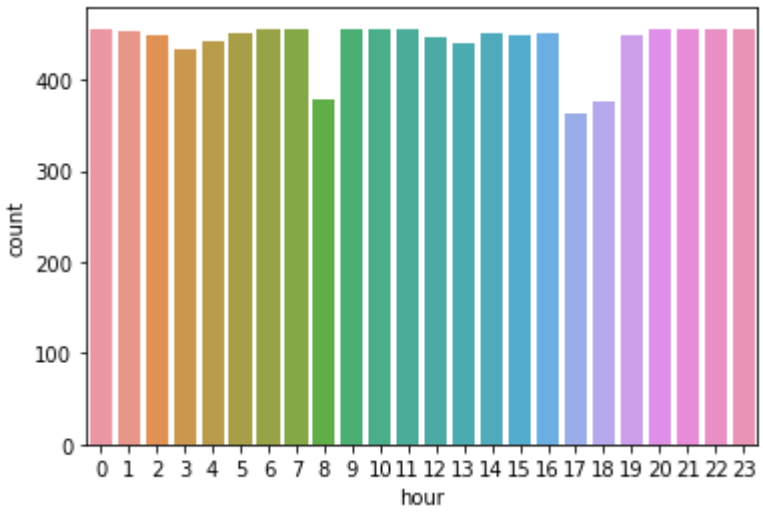
```
df['hour']=ls
```

In [38]:

```
sns.countplot(df['hour'])
# Below graph shows the Count plot at each hour.
# We have almost the same count at each hour. Slightly less count at 8th,17th and 18th hour
```

Out[38]:

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



In [39]:

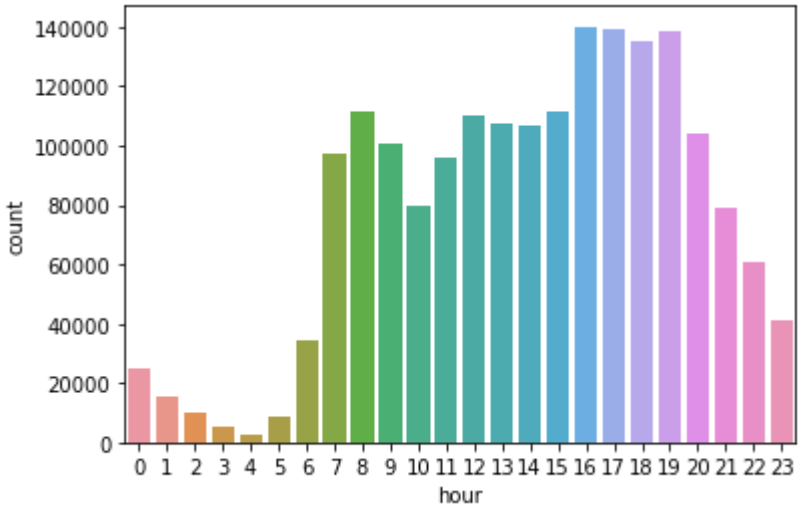
```
ans=df.groupby(['hour'])['count'].sum().reset_index()
```

In [40]:

```
sns.barplot(x=ans['hour'],y=ans['count'])
# Bar plot shows the summ of 'count' at each hour
# Higher number of booking take place between 17th hour and 20th hour. Very booking between
```

Out[40]:

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

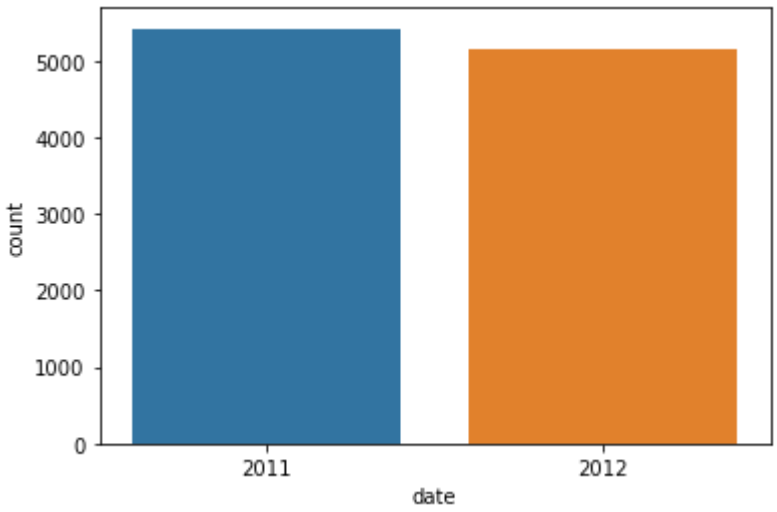


In [41]:

```
sns.countplot(pd.to_datetime(df['date']).dt.year)
# Graph shows the total 'count' of booking in each year
# We almost same number of booking on each year
```

Out[41]:

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

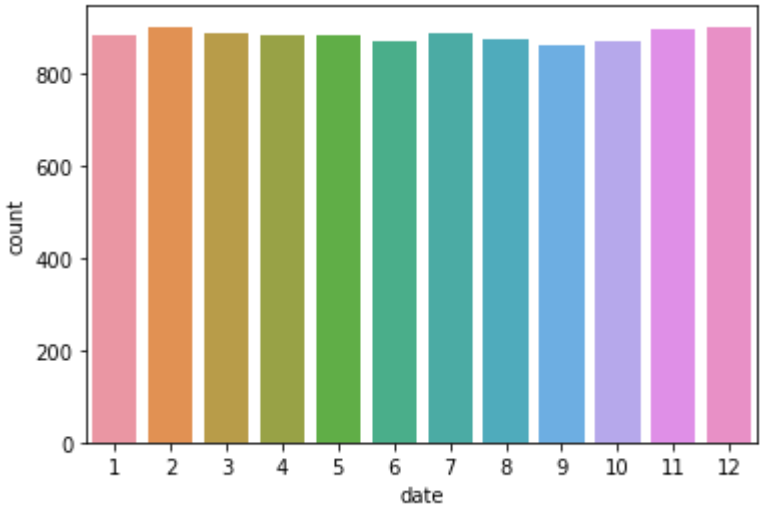


In [42]:

```
sns.countplot(pd.to_datetime(df['date']).dt.month)
# Below graph shows the count of booking on each month and they are roughly the same
```

Out[42]:

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

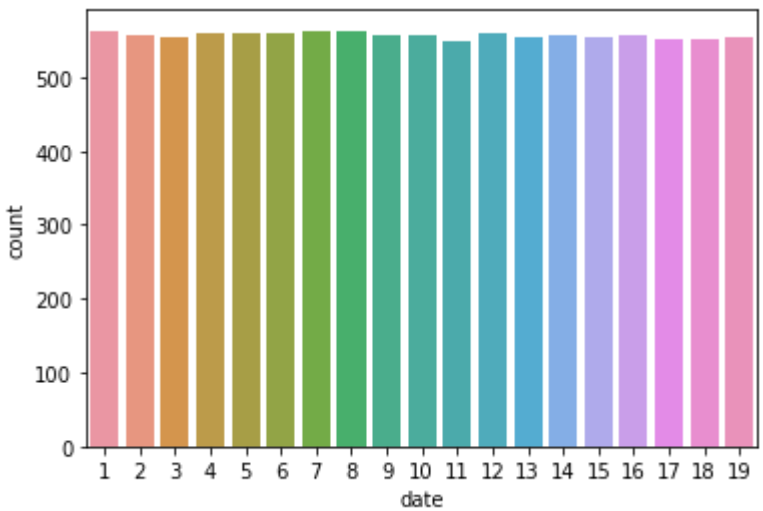


In [43]:

```
sns.countplot(pd.to_datetime(df['date']).dt.day)
# Below graph shows the count of booking on each date and they are roughly the same
```

Out[43]:

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



In [44]:

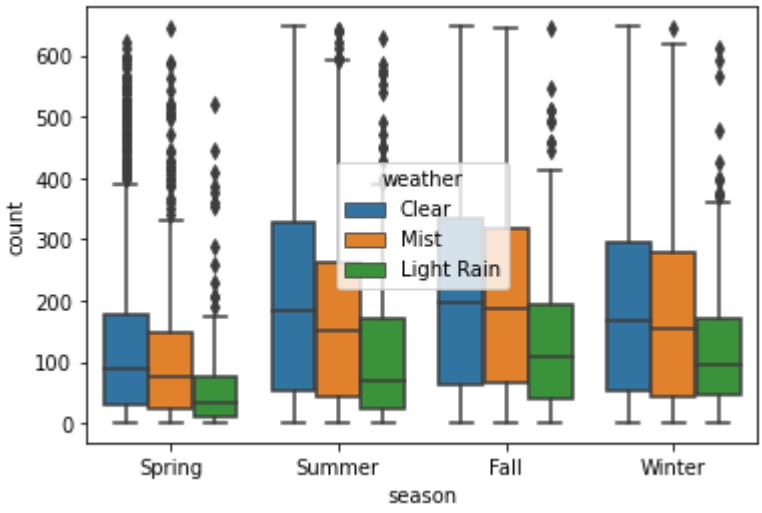
```
pd.crosstab(index=df['season'], columns=df['weather'], values=df['count'],  aggfunc=sum, no
# Below table shows the conditional probality of 'count' across weather and season
```

Out[44]:

weather	Clear	Light Rain	Mist
season			
Fall	0.730312	0.050579	0.219109
Spring	0.705604	0.042920	0.251476
Summer	0.712295	0.052293	0.235412
Winter	0.655327	0.055515	0.289158
All	0.701486	0.051121	0.247393

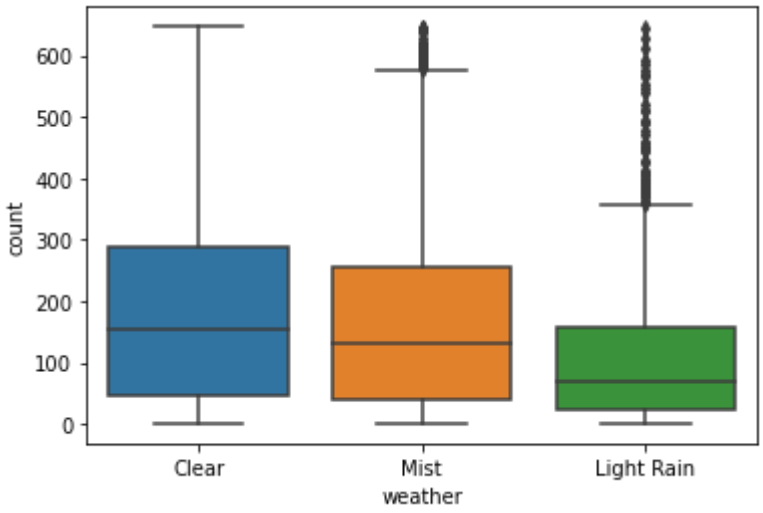
In [45]:

```
sns.boxplot(x=df['season'],y=df['count'],hue=df['weather'])
plt.show()
# Below is the barplot for 'count' across weather.
# There are lot of outliners in clear weather and season spring
```



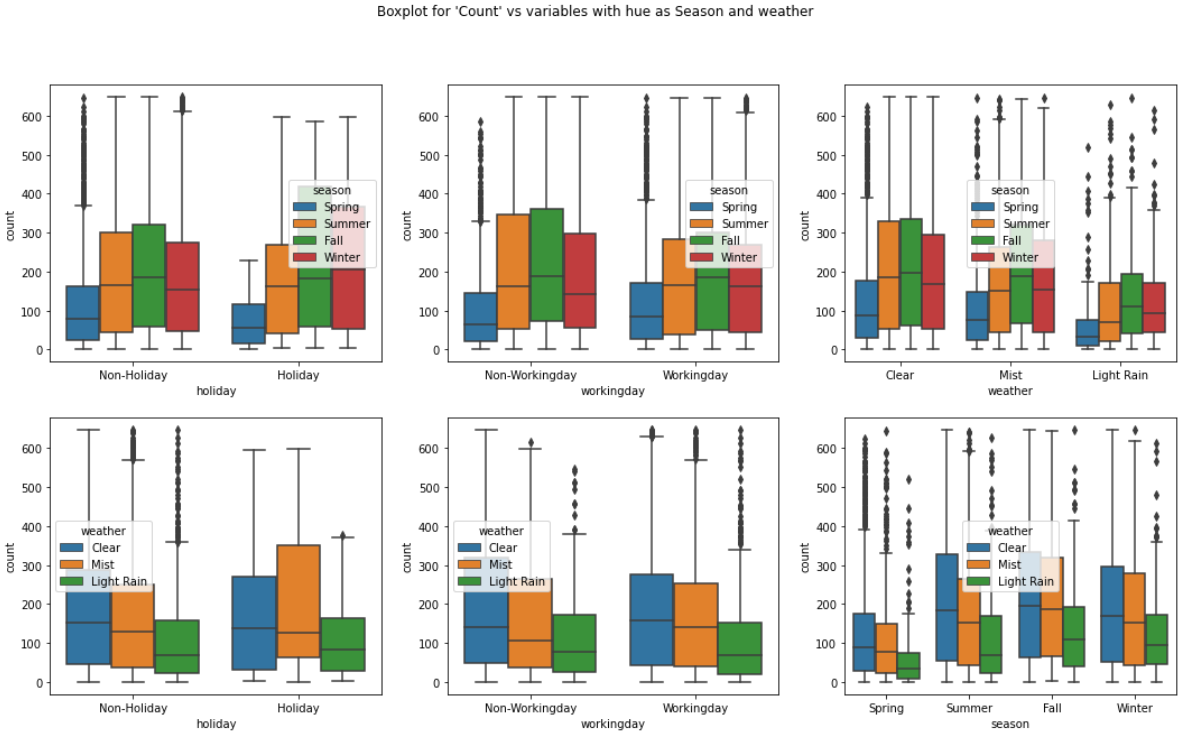
In [46]:

```
sns.boxplot(x=df['weather'],y=df['count'])
plt.show()
# Below plot concludes that clear weather has highest median
```



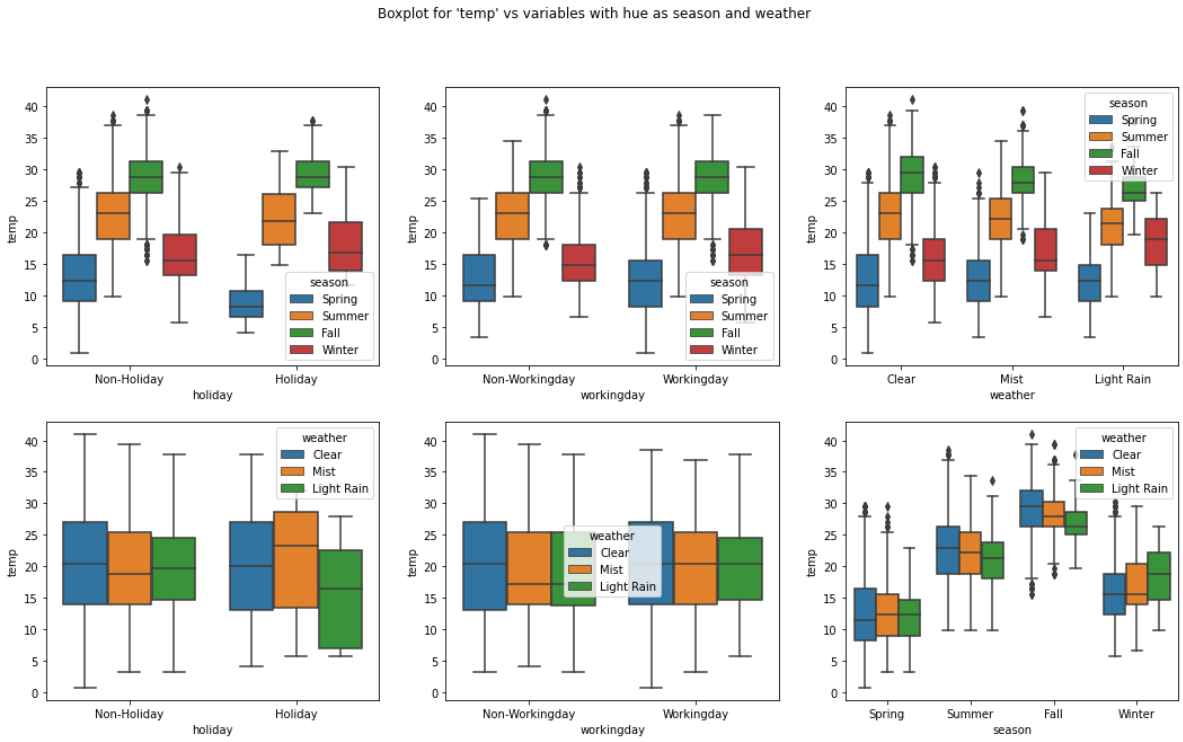
In [47]:

```
fig, axes = plt.subplots(2, 3, figsize=(18, 10))

fig.suptitle('Boxplot for \'Count\' vs variables with hue as Season and weather')
sns.boxplot(ax=axes[0, 0], data=df, x='holiday', y='count',hue='season' )
sns.boxplot(ax=axes[0, 1], data=df, x='workingday', y='count',hue='season')
sns.boxplot(ax=axes[0, 2], data=df, x='weather', y='count',hue='season')
sns.boxplot(ax=axes[1, 0], data=df, x='holiday', y='count',hue='weather')
sns.boxplot(ax=axes[1, 1], data=df, x='workingday', y='count',hue='weather')
sns.boxplot(ax=axes[1, 2], data=df, x='season', y='count',hue='weather')
plt.show()
# Medians of count on Holiday and non holiday days are almost the same
```



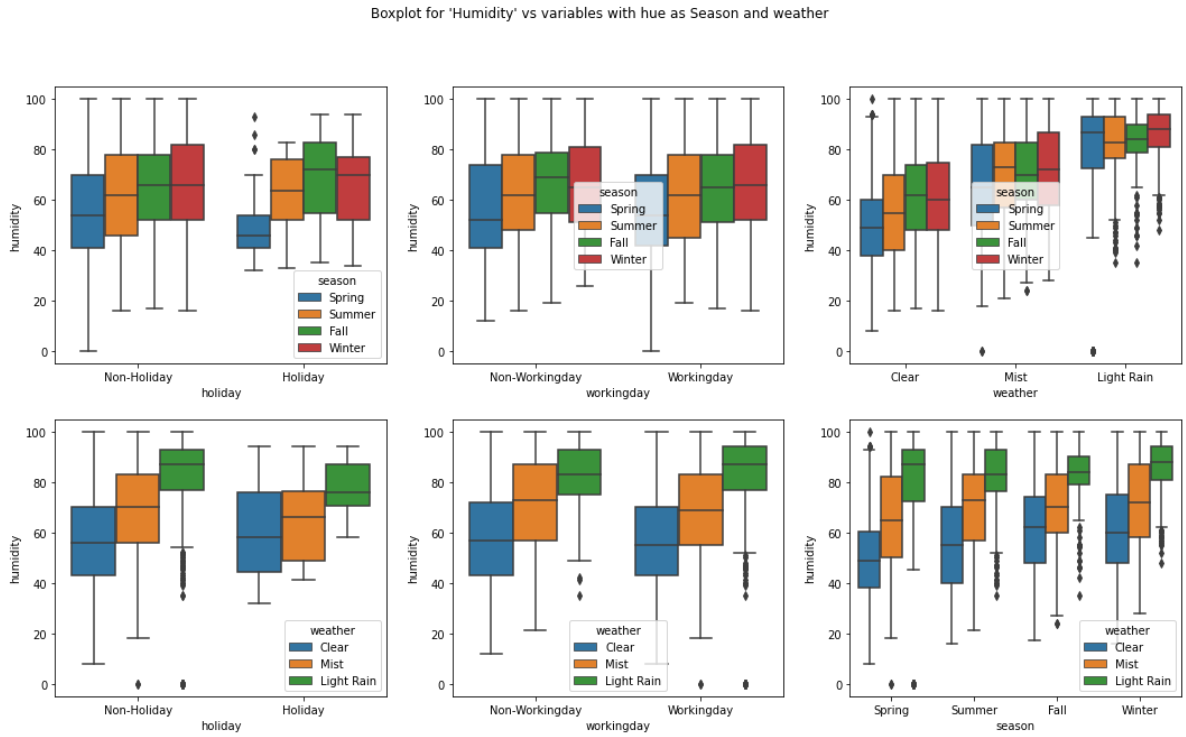
In [48]:

```
fig, axes = plt.subplots(2, 3, figsize=(18, 10))

fig.suptitle('Boxplot for \'temp\' vs variables with hue as season and weather')
sns.boxplot(ax=axes[0, 0], data=df, x='holiday', y='temp',hue='season' )
sns.boxplot(ax=axes[0, 1], data=df, x='workingday', y='temp',hue='season')
sns.boxplot(ax=axes[0, 2], data=df, x='weather', y='temp',hue='season')
sns.boxplot(ax=axes[1, 0], data=df, x='holiday', y='temp',hue='weather')
sns.boxplot(ax=axes[1, 1], data=df, x='workingday', y='temp',hue='weather')
sns.boxplot(ax=axes[1, 2], data=df, x='season', y='temp',hue='weather')
plt.show()
# Temperature is Fall are higher followed by summer, spring and winter
```



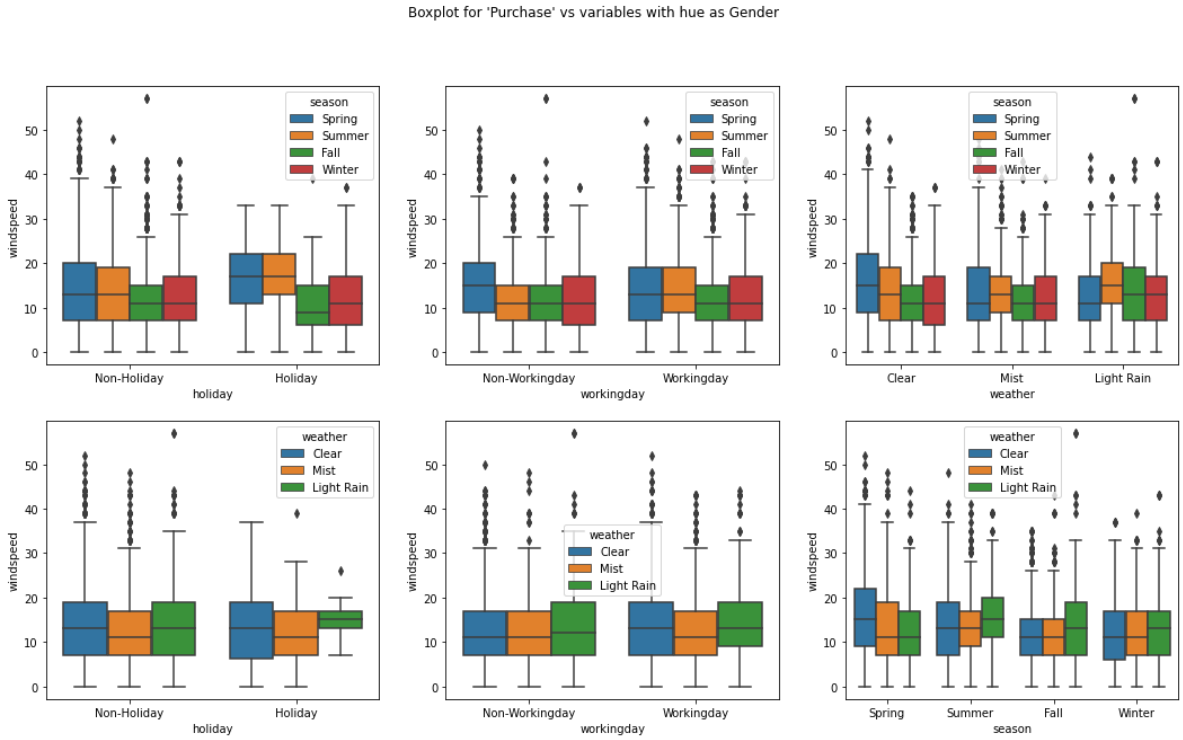
In [49]:

```
fig, axes = plt.subplots(2, 3, figsize=(18, 10))

fig.suptitle('Boxplot for \'Humidity\' vs variables with hue as Season and weather')
sns.boxplot(ax=axes[0, 0], data=df, x='holiday', y='humidity',hue='season' )
sns.boxplot(ax=axes[0, 1], data=df, x='workingday', y='humidity',hue='season')
sns.boxplot(ax=axes[0, 2], data=df, x='weather', y='humidity',hue='season')
sns.boxplot(ax=axes[1, 0], data=df, x='holiday', y='humidity',hue='weather')
sns.boxplot(ax=axes[1, 1], data=df, x='workingday', y='humidity',hue='weather')
sns.boxplot(ax=axes[1, 2], data=df, x='season', y='humidity',hue='weather')
plt.show()
# Humidity is higher when weather is Light rain followed by Mist and clear
```



In [50]:

```
fig, axes = plt.subplots(2, 3, figsize=(18, 10))

fig.suptitle('Boxplot for \'Purchase\' vs variables with hue as Gender')
sns.boxplot(ax=axes[0, 0], data=df, x='holiday', y='windspeed',hue='season' )
sns.boxplot(ax=axes[0, 1], data=df, x='workingday', y='windspeed',hue='season')
sns.boxplot(ax=axes[0, 2], data=df, x='weather', y='windspeed',hue='season')
sns.boxplot(ax=axes[1, 0], data=df, x='holiday', y='windspeed',hue='weather')
sns.boxplot(ax=axes[1, 1], data=df, x='workingday', y='windspeed',hue='weather')
sns.boxplot(ax=axes[1, 2], data=df, x='season', y='windspeed',hue='weather')
plt.show()
```



In [51]:

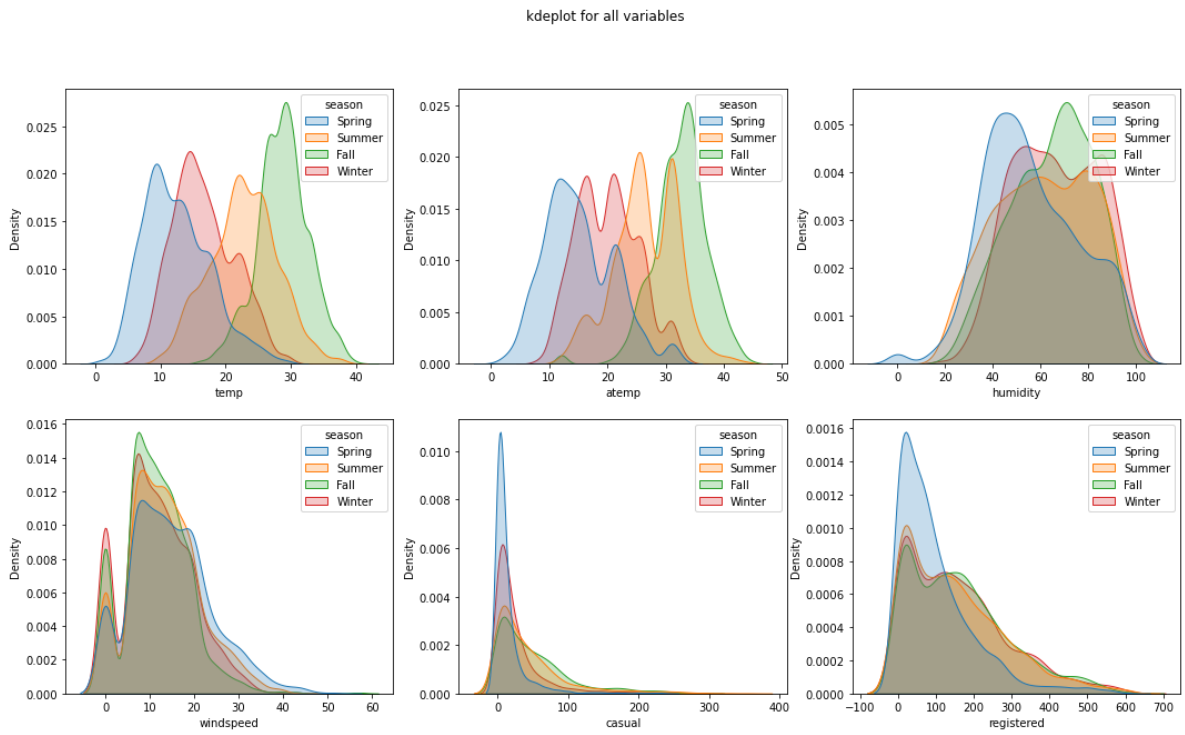
```
fig, axes = plt.subplots(3, figsize=(18, 15))

fig.suptitle('Histplot for all variables')
#sns.histplot(ax=axes[0, 0], data=df, x='season',multiple='season')
sns.histplot(ax=axes[0], data=df, x='holiday',hue='season',multiple='stack')
sns.histplot(ax=axes[1], data=df, x='workingday',hue='season',multiple='stack')
sns.histplot(ax=axes[2], data=df, x='weather',hue='season',multiple='stack')
plt.show()
# Below plot shows that more booking are taken on 'Non-Holiday','Working day' and when weat
```

In [52]:

```
fig, axes = plt.subplots(3, figsize=(18, 15))

fig.suptitle('Histplot for all variables')
#sns.histplot(ax=axes[0, 0], data=df, x='season',multiple='season')
sns.histplot(ax=axes[0], data=df, x='holiday',hue='weather',multiple='stack')
sns.histplot(ax=axes[1], data=df, x='workingday',hue='weather',multiple='stack')
sns.histplot(ax=axes[2], data=df, x='season',hue='weather',multiple='stack')
plt.show()

# Below bar plot concludes that there is no difference between the 4 season for the total '
```

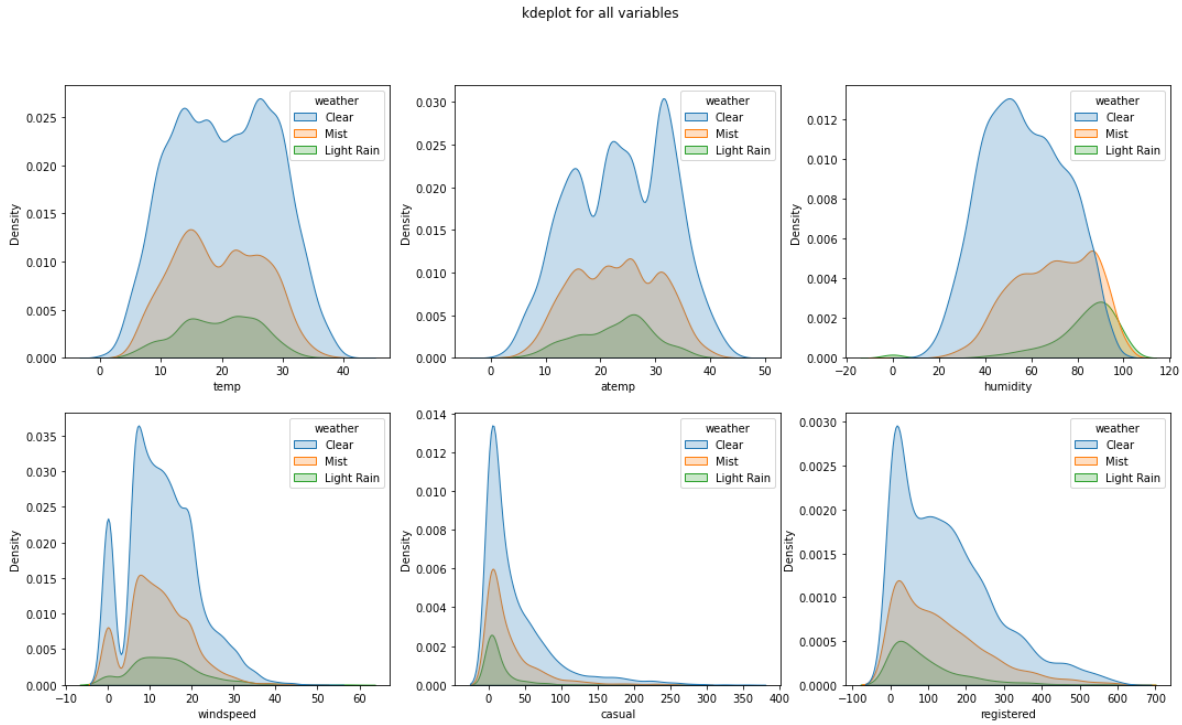
In [53]:

```
fig, axes = plt.subplots(2, 3, figsize=(18, 10))

fig.suptitle('kdeplot for all variables')
sns.kdeplot(ax=axes[0, 0], data=df, x='temp',hue='season',shade=True)
sns.kdeplot(ax=axes[0, 1], data=df, x='atemp',hue='season',shade=True)
sns.kdeplot(ax=axes[0, 2], data=df, x='humidity',hue='season',shade=True)
sns.kdeplot(ax=axes[1, 0], data=df, x='windspeed',hue='season',shade=True)
sns.kdeplot(ax=axes[1, 1], data=df, x='casual',hue='season',shade=True)
sns.kdeplot(ax=axes[1, 2], data=df, x='registered',hue='season',shade=True)
plt.show()
# Below kde plot concludes that Casual count and registered count are more in spring and wi
```



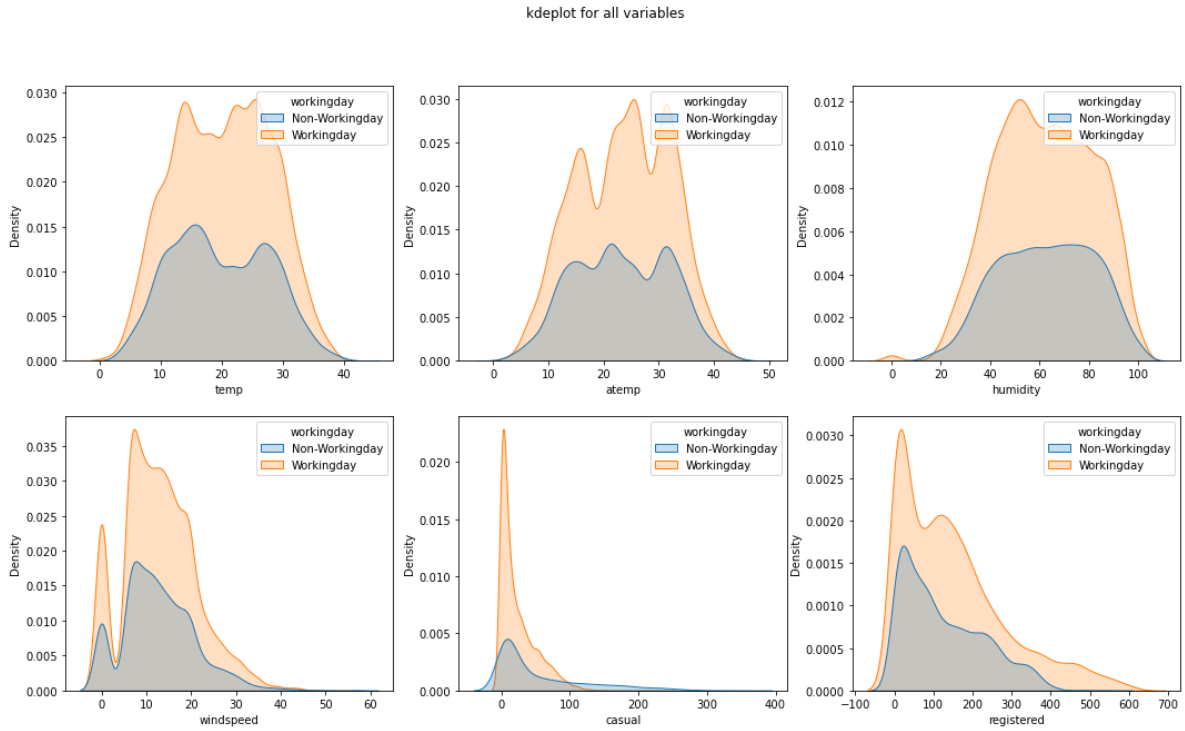
In [54]:

```
fig, axes = plt.subplots(2, 3, figsize=(18, 10))

fig.suptitle('kdeplot for all variables')
sns.kdeplot(ax=axes[0, 0], data=df, x='temp',hue='weather',shade=True)
sns.kdeplot(ax=axes[0, 1], data=df, x='atemp',hue='weather',shade=True)
sns.kdeplot(ax=axes[0, 2], data=df, x='humidity',hue='weather',shade=True)
sns.kdeplot(ax=axes[1, 0], data=df, x='windspeed',hue='weather',shade=True)
sns.kdeplot(ax=axes[1, 1], data=df, x='casual',hue='weather',shade=True)
sns.kdeplot(ax=axes[1, 2], data=df, x='registered',hue='weather',shade=True)
plt.show()
# Below kde plot concludes that Casual count and registered count are more between 0 to 100
```



In [55]:

```
fig, axes = plt.subplots(2, 3, figsize=(18, 10))

fig.suptitle('kdeplot for all variables')
sns.kdeplot(ax=axes[0, 0], data=df, x='temp',hue='workingday',shade=True)
sns.kdeplot(ax=axes[0, 1], data=df, x='atemp',hue='workingday',shade=True)
sns.kdeplot(ax=axes[0, 2], data=df, x='humidity',hue='workingday',shade=True)
sns.kdeplot(ax=axes[1, 0], data=df, x='windspeed',hue='workingday',shade=True)
sns.kdeplot(ax=axes[1, 1], data=df, x='casual',hue='workingday',shade=True)
sns.kdeplot(ax=axes[1, 2], data=df, x='registered',hue='workingday',shade=True)
plt.show()
# Below is the kde plot for all numerical variables with hue as workingday and non-working
```



In [56]:

```
df.groupby(['season'])['count'].sum().reset_index()
# Below table shows the sum of count on each season
# Season Fall has the highest count followed by summer,spring and winter
```

Out[56]:

	season	count
0	Fall	551274
1	Spring	300999
2	Summer	515803
3	Winter	491901

In [57]:

```
df.groupby(['weather'])['count'].sum().reset_index()
# Below table shows the sum of count on each weather
# Season Fall has the highest count followed by clear, mist and light rain
```

Out[57]:

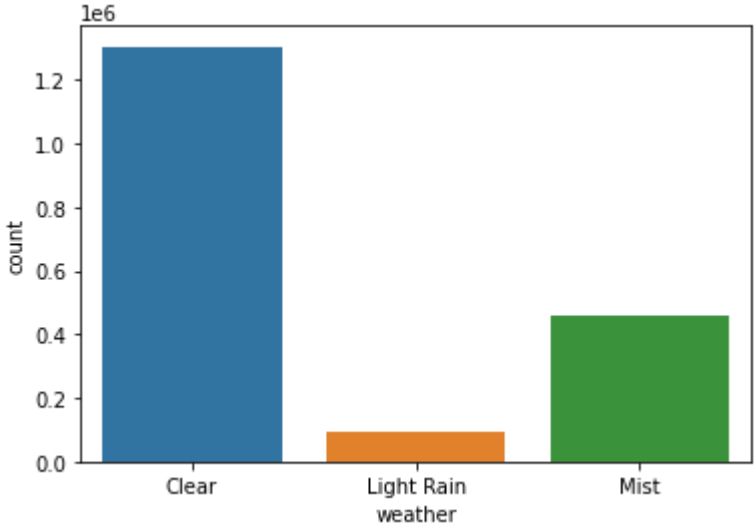
	weather	count
0	Clear	1304748
1	Light Rain	95083
2	Mist	460146

In [58]:

```
ddf=df.groupby(['weather'])['count'].sum().reset_index()
sns.barplot(x=ddf['weather'],y=ddf['count'])
# Below is the barplot for count withrespect to weather
```

Out[58]:

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



In [59]:

```
df.groupby(['holiday'])['count'].sum().reset_index()
# Below table shows the sum of count on each holiday
# More booking happen on non-holiday
```

Out[59]:

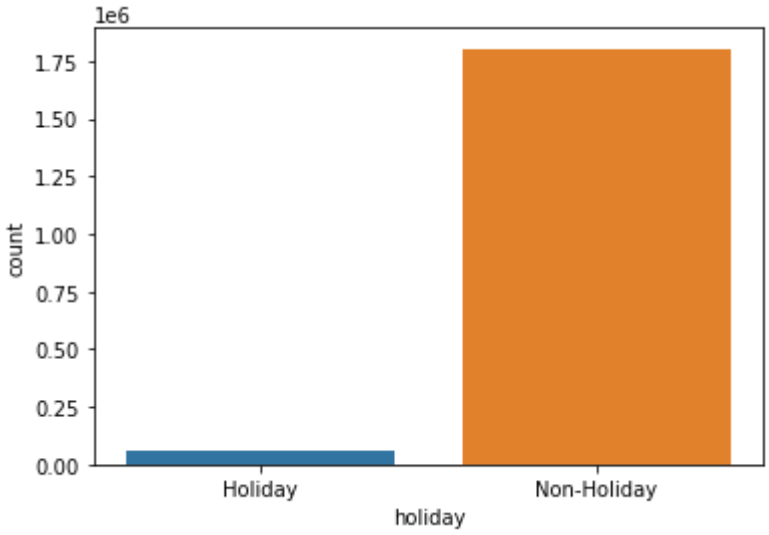
	holiday	count
0	Holiday	56420
1	Non-Holiday	1803557

In [60]:

```
ddf=df.groupby(['holiday'])['count'].sum().reset_index()
sns.barplot(x=ddf['holiday'],y=ddf['count'])
# Below is the barplot for count withrespect to holiday
```

Out[60]:

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

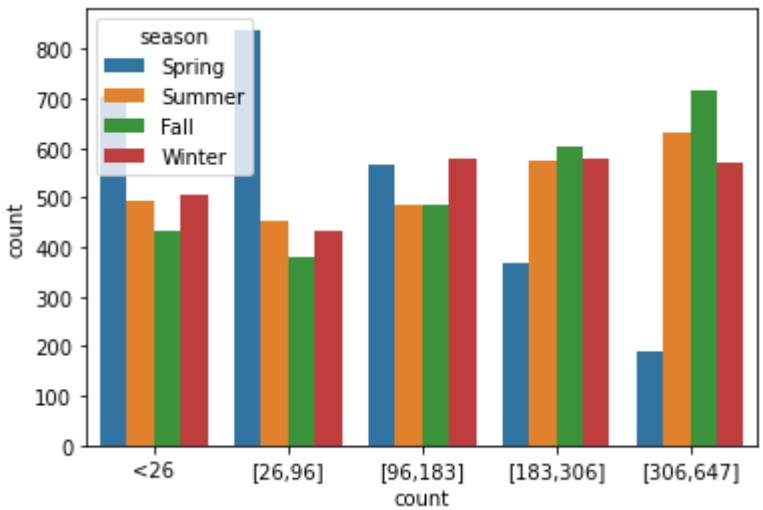


In [61]:

```
sns.countplot(pd.qcut(df['count'], q=[0,0.2,0.4,0.6,0.8,1], labels=['<26','[26,96]','[96,18
# 'Count' is split withrespect to percentiles
#  In spring season the count of cycle takers are more for <26, '26,96' when compared with
# But it is the least when count is [183,306] and [306,647]
```

Out[61]:

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

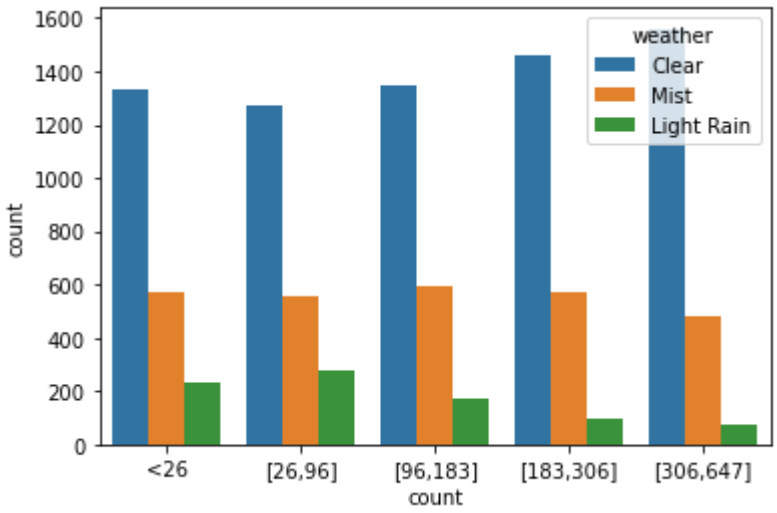


In [62]:

```
sns.countplot(pd.qcut(df['count'], q=[0,0.2,0.4,0.6,0.8,1], labels=['<26','[26,96]','[96,18
# 'Count' is split with respect to percentiles
#  In mist weather the count of cycle takers are almost the same across different count ran
```

Out[62]:

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

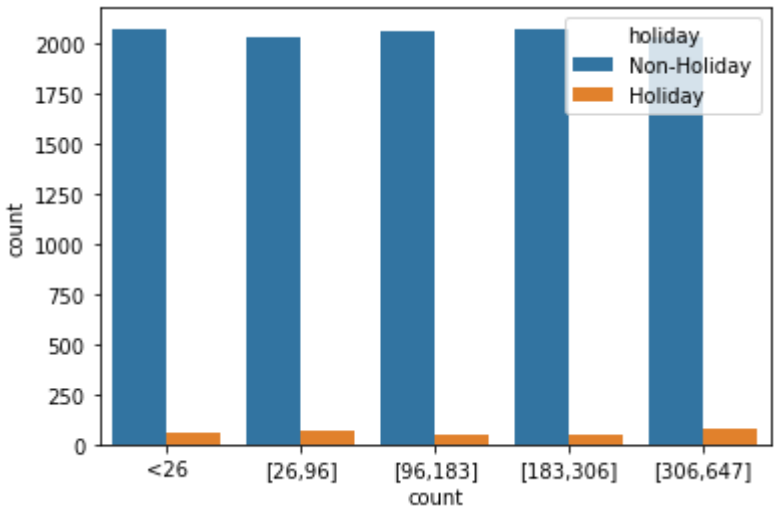


In [63]:

```
sns.countplot(pd.qcut(df['count'], q=[0,0.2,0.4,0.6,0.8,1], labels=['<26','[26,96]','[96,18
# 'Count' is split with respect to percentiles
#  Below is the count plot for 'holiday'
```

Out[63]:

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

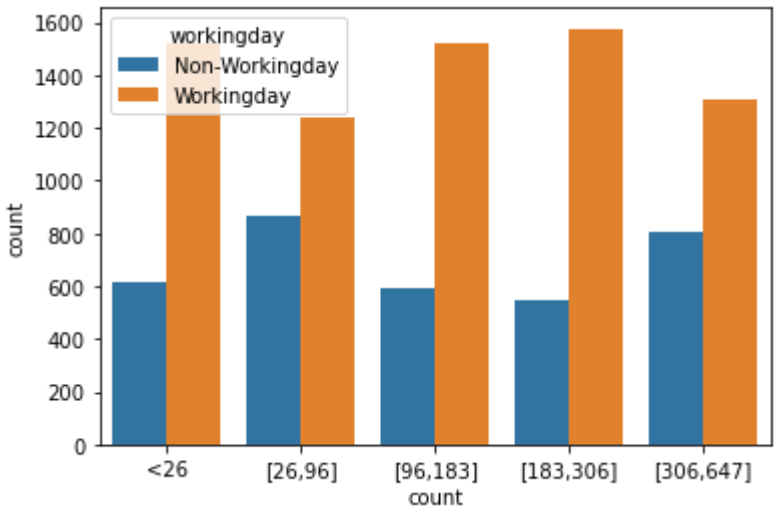


In [64]:

```
sns.countplot(pd.qcut(df['count'], q=[0,0.2,0.4,0.6,0.8,1], labels=['<26','[26,96]','[96,18
# 'Count' is split with respect to percentiles
#  Below is the count plot for 'Workingday'
```

Out[64]:

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



In [65]:

```
df.groupby(['workingday'])['count'].sum().reset_index()
# Below table shows the sum of count on each working day
# Bookings are doubled on a workingday when compared with non-workingday
```

Out[65]:

	workingday	count
0	Non-Workingday	621205
1	Workingday	1238772

In [66]:

```
df.groupby(['season','weather'])[['count']].aggregate({'count':['sum']})
# Below table shows the sum of 'count' with respect to season and weather
```

Out[66]:

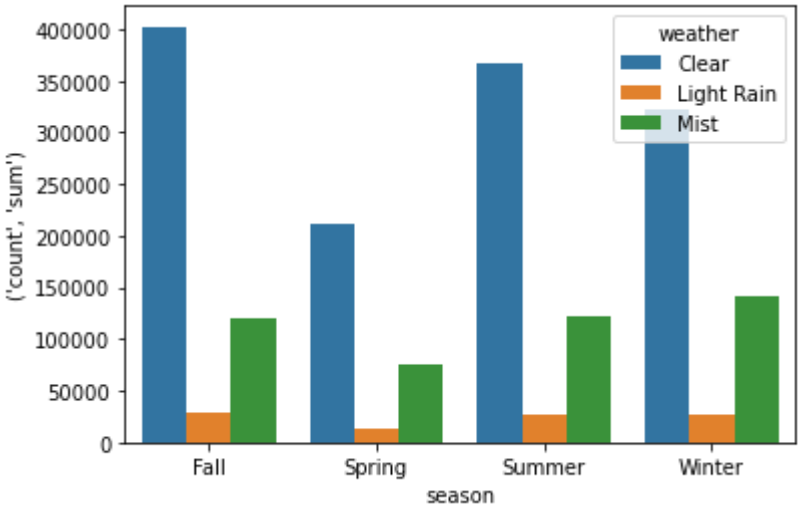
		count
		sum
season	weather	
	Clear	402602
Fall	Light Rain	27883
	Mist	120789
	Clear	212386
Spring	Light Rain	12919
	Mist	75694
	Clear	367404
Summer	Light Rain	26973
	Mist	121426
	Clear	322356
Winter	Light Rain	27308
	Mist	142237

In [67]:

```
ddf=df.groupby(['season','weather'])[['count']].aggregate({'count':['sum']}).reset_index()
sns.barplot(x=ddf['season'],y=ddf[('count', 'sum')],hue=ddf['weather'])
# Below isthe bar plot for weather ans season
```

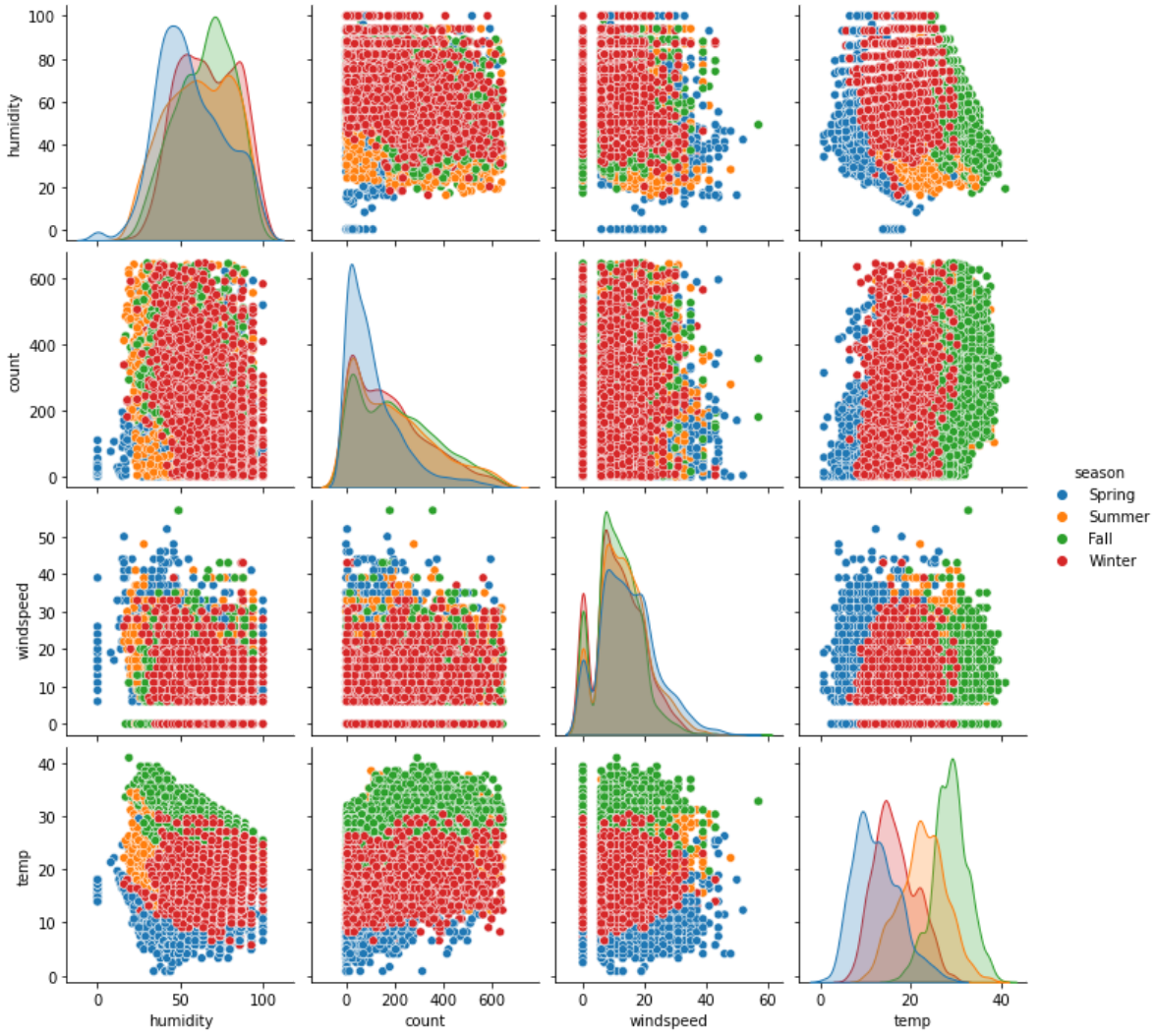
Out[67]:

```
<AxesSubplot:xlabel='season', ylabel="('count', 'sum')">
```



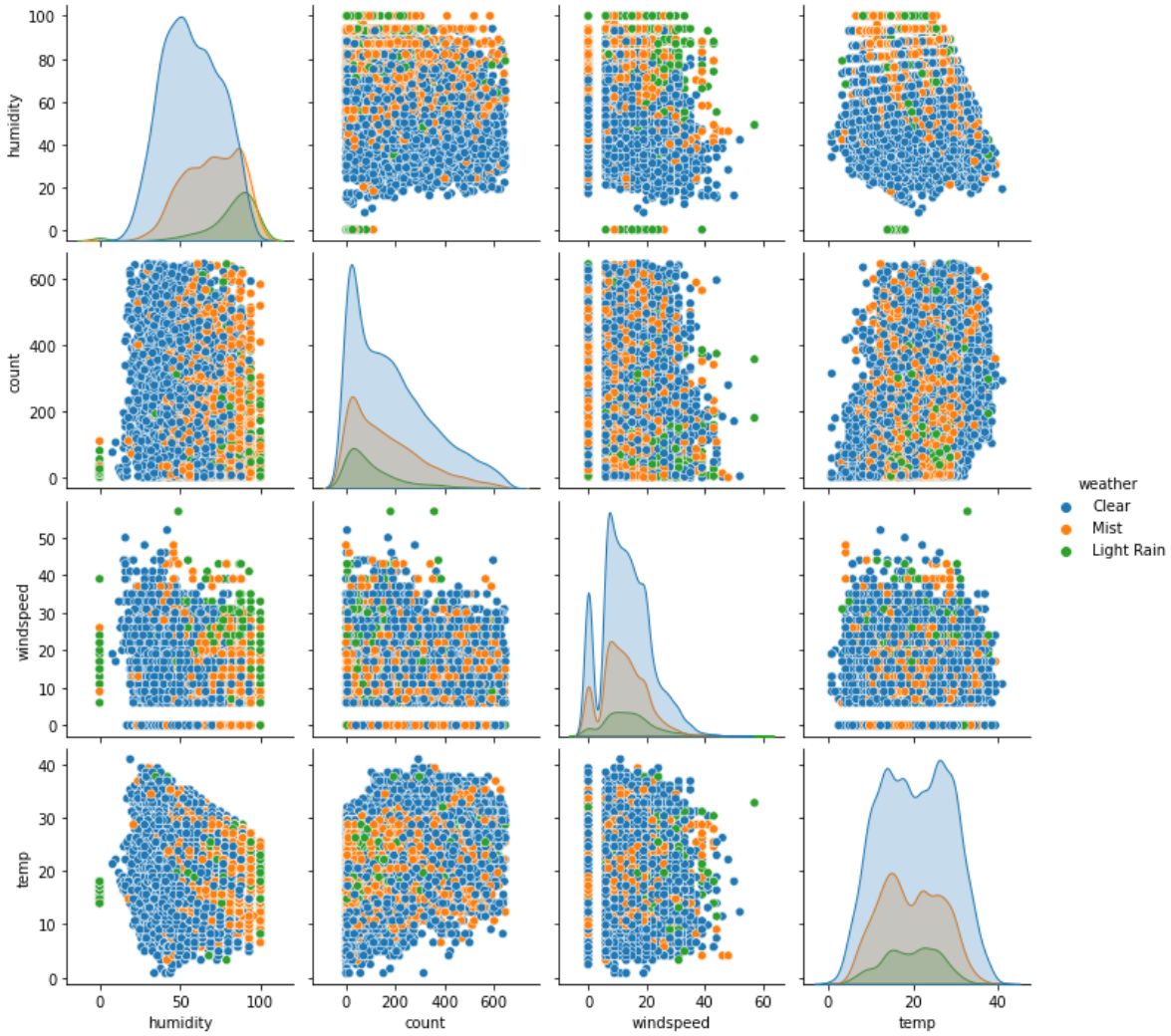
In [68]:

```
sns.pairplot(df, vars=['humidity', 'count','windspeed','temp'], hue = "season")
plt.show()
# Below is the pair plot for variables humidity, count, windspeed and temp.With hue as seas
```



In [69]:

```
sns.pairplot(df, vars=['humidity', 'count','windspeed','temp'], hue = "weather")
plt.show()
# Below is the pair plot for variables humidity, count, windspeed and temp. With hue as wea
```



In [70]:

```
df.corr()
# Below table shows the correlation for numerical variables
# 0.42 correlation between count and hour
# 0.388096 correlation between count and temp
```

Out[70]:

	temp	atemp	humidity	windspeed	casual	registered	count	h
temp	1.000000	0.985884	-0.051047	-0.022034	0.468588	0.304638	0.388096	0.134
atemp	0.985884	1.000000	-0.030210	-0.062519	0.463596	0.302310	0.384707	0.129
humidity	-0.051047	-0.030210	1.000000	-0.319656	-0.335256	-0.274258	-0.323378	-0.270
windspeed	-0.022034	-0.062519	-0.319656	1.000000	0.088682	0.102894	0.110179	0.145
casual	0.468588	0.463596	-0.335256	0.088682	1.000000	0.513760	0.717520	0.301
registered	0.304638	0.302310	-0.274258	0.102894	0.513760	1.000000	0.966217	0.412
count	0.388096	0.384707	-0.323378	0.110179	0.717520	0.966217	1.000000	0.425
hour	0.134000	0.129333	-0.270886	0.145230	0.301674	0.412867	0.425824	1.000

In [71]:

```
stats.spearmanr(df['temp'], df['count'])[0]
# spearman rank correlation between temp and count is 0.392
```

Out[71]:

```
0.39238080924519303
```

In [72]:

```
stats.spearmanr(df['humidity'], df['count'])[0]
# spearman rank correlation between humidity and count is -0.137
```

Out[72]:

```
-0.3487468591029361
```

In [73]:

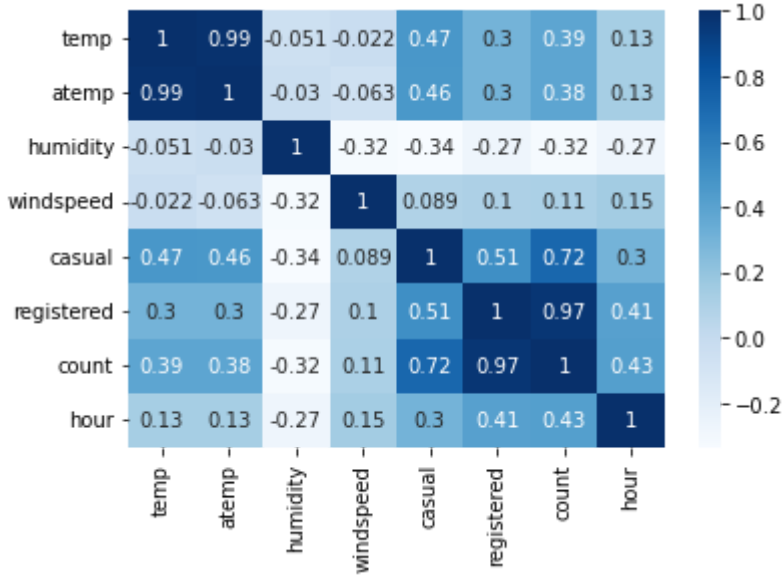
```
stats.spearmanr(df['windspeed'], df['count'])[0]
# spearman rank correlation between windspeed and count is 0.13734
```

Out[73]:

```
0.137348161531927
```

In [74]:

```
sns.heatmap(df.corr(),cmap='Blues',annot=True)
plt.show()
#Below is the heatmap for coefficient correlation for numerical variables
```

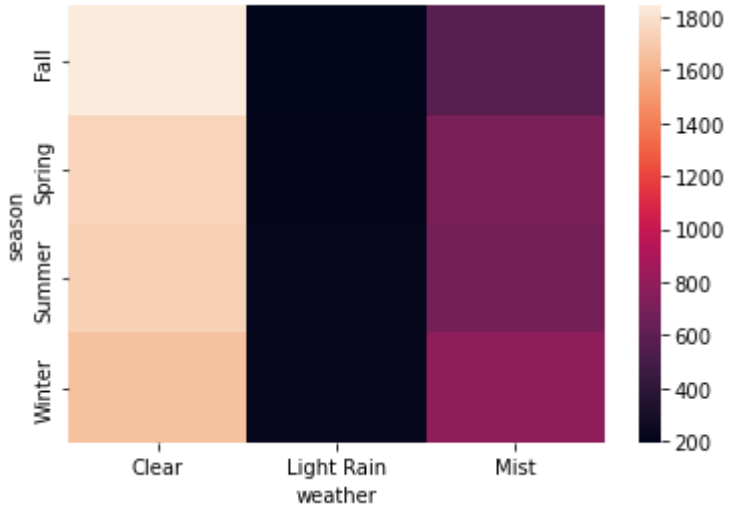


In [75]:

```
sns.heatmap(pd.crosstab(df['season'],df['weather']))
# Below is the heat map for count for season and weather
```

Out[75]:

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



In [76]:

```
sns.heatmap(pd.crosstab(df['holiday'],df['workingday']))
# Below is the heat map for count for holiday and working day
```

Out[76]:

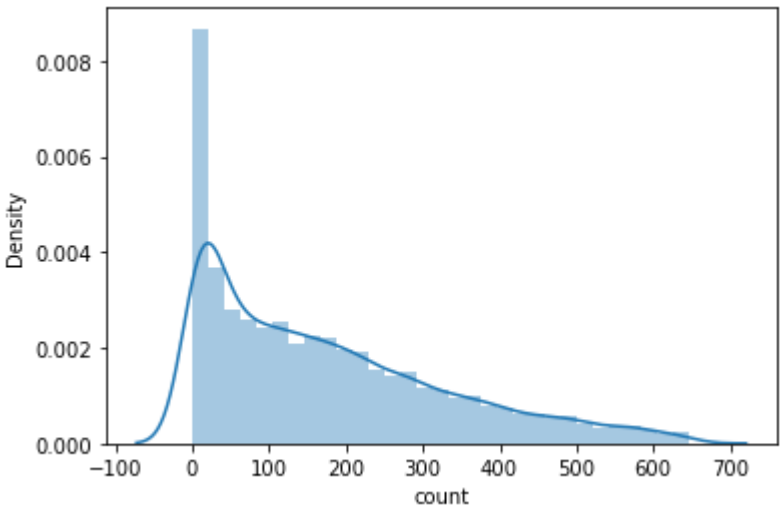
```
<AxesSubplot:xlabel='workingday', ylabel='holiday'>
```

In [77]:

```
sns.distplot(df['count'])
# Below is the distribution plot for 'count'.
# This distribution is right skewed and not a normal distribution
```

Out[77]:

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



2 Sample T Test

Assumptions

1. Sample mean and variance should be finite

2. Sample follow normal dist

Null and alternate hypothesis for KS test

#Ho= Sample follow normal distribution,
#Ha= Sample does not follow normal distribution,
Significance value= 5%
p-value should be greater than 0.05 for the distribution to be normal

In [78]:

```
#Ho= Sample follow normal distribution
#Ha= Sample does not follow normal distribution
def norm(x,var):
    ls=[]
    p_values=[]
    for i in range(1000):
        ans=np.random.choice(x[var],30,replace=False)
        p_values.append(stats.ks_2samp(ans,stats.norm.rvs(loc=np.mean(ans), scale=np.std(an
    p_values=np.array(p_values)
    ans=len(p_values[p_values>=0.05])/1000
    if ans>0.93:
        print('Sample distribution follow Normal distribution')
    else:
        print('Sample distribution does not follow Normal distribution')
```

T Test for Holiday and count

In [79]:

```
# T Test for Holiday and count
NH=df.loc[df['holiday']=='Non-Holiday']
H=df.loc[df['holiday']=='Holiday']
print(np.var(NH['count']),np.var(H['count']),np.mean(NH['count']),np.mean(H['count']))
print(norm(NH,'count'),norm(H,'count'))
statistic,pvalue=stats.ttest_ind(np.random.choice(NH['count'],30,replace=False),np.random.c
if pvalue > 0.05:
    print('We fail to reject Null hypothesis',',p_value is,',pvalue,',T_ stat is,',statisti
    print('Sample means are equal')
else:
    print('We have enough evidance to reject null hypothesis')
    print('Sample means are not equal')
```

```
24378.293657253183 26732.701626501595 175.5115803814714 182.58899676375404
Sample distribution follow Normal distribution
Sample distribution follow Normal distribution
None None
We fail to reject Null hypothesis ,p_value is, 0.19561393396280863 ,T_ stat
is, -1.309246853714329
Sample means are equal
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++<br>
Null hypothesis: The mean of count is equal on a Non-holiday and on a holiday <br>
```

```
Alternate hypothesus:The mean of count not equal on a Non-holiday and on a holiday
The<br>
Significance value= 5% <br>
p-value <= 0.05, Then we have enough evidance to reject Null hypothesis<br>
p-value > 0.05, We fail to reject null hypothesis <br>
Sample mean and standard deviation are finite and follow  closely normal distribution
<br>
Result: As pvalue > 0.05 we conclude the mean of count is equal on a Non-holiday and on a
holiday <br>
```

T Test for Working day and count

In [80]:

```
# T Test for Working Day and count
NH=df.loc[df['workingday']=='Workingday']
H=df.loc[df['workingday']=='Non-Workingday']
print(np.var(NH['count']),np.var(H['count']),np.mean(NH['count']),np.mean(H['count']))
print(norm(NH,'count'),norm(H,'count'))
statistic,pvalue=stats.ttest_ind(np.random.choice(NH['count'],30,replace=False),np.random.c
if pvalue > 0.05:
    print('We fail to reject Null hypothesis',',p_value is,',pvalue,',T_ stat is,',statisti
    print('Sample means are equal')
else:
    print('We have enough evidance to reject null hypothesis','p_value is',pvalue,'T_ stat'
    print('Sample means are not equal')
```

```
23213.25151087672 26983.341207310033 173.01284916201118 181.37372262773724
Sample distribution follow Normal distribution
Sample distribution follow Normal distribution
None None
We fail to reject Null hypothesis ,p_value is, 0.8136432572102805 ,T_ stat i
s, 0.2368039622192249
Sample means are equal
```

++

Null hypothesis: The mean of count is equal on a Non-working day and on a working day
Alternate hypothesus:The mean of count not equal on a Non-working day and on a working day
Significance value= 5%
p-value <= 0.05, Then we have enough evidance to reject Null hypothesis
p-value > 0.05, We fail to reject null hypothesis
Sample mean and standard deviation are finite and closely follow normal distribution
Result: As pvalue > 0.05 we conclude the mean of count is equal on a Non-working day and on a working day

++

""" ANOVA assumptions:

1. Each group assumptions is gaussian
2. Eah group variance is roughly the same
3. Each observation is independent
4. Different groups must have equal sample size """

ANOVA test for testing means of count for differents seasons

In [81]:

```
#ANOVA
df['season'].value_counts()
#Sample sizes are almost equal
```

Out[81]:

```
Spring    2669
Winter    2665
Summer    2634
Fall      2617
Name: season, dtype: int64
```

In [82]:

```
spri=df.loc[df['season']=='Spring']
wint=df.loc[df['season']=='Winter']
summ=df.loc[df['season']=='Summer']
fall=df.loc[df['season']=='Fall']
```

In [83]:

```
len(outliners(spri,'count'))*100/len(spri),len(outliners(wint,'count'))*100/len(wint),len(o
```

Out[83]:

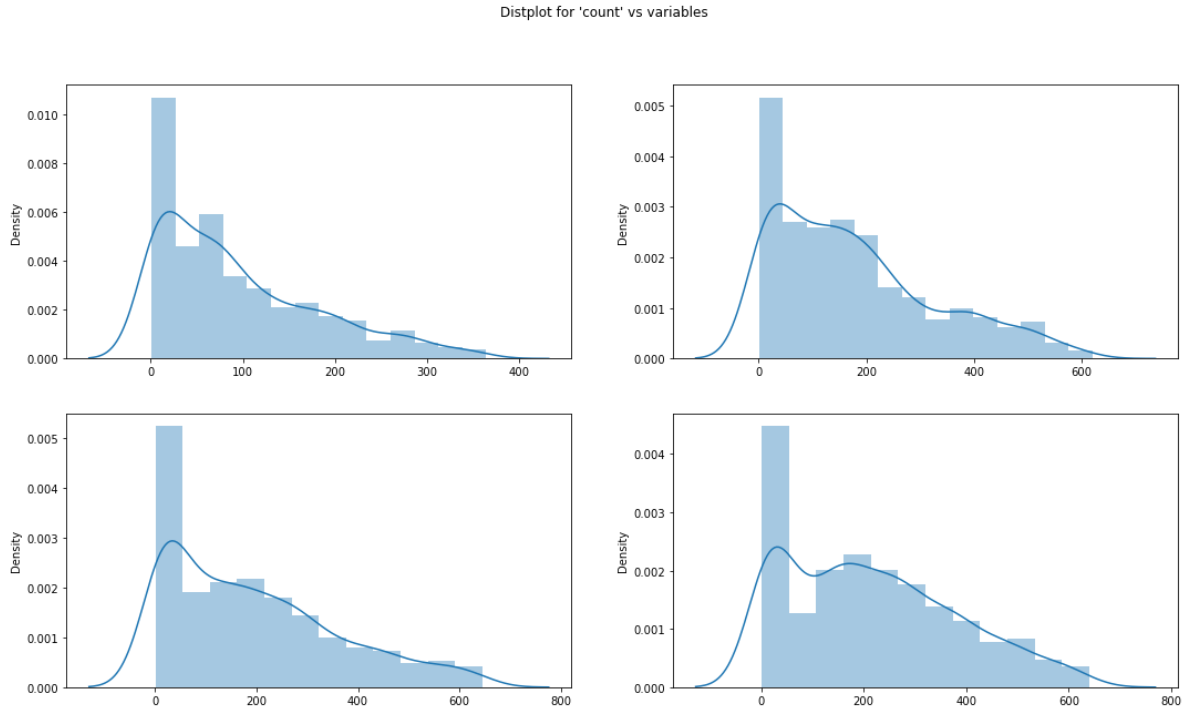
```
(4.720869239415512, 0.525328330206379, 0.0, 0.0)
```

In [84]:

```
spri.drop(outliners(spri,'count'),axis=0,inplace=True)
wint.drop(outliners(wint,'count'),axis=0,inplace=True)
# Outliners are removed as they are only 4.7 % and 0.52 %
```

In [85]:

```
fig, axes = plt.subplots(2, 2, figsize=(18, 10))

fig.suptitle('Distplot for \'count\' vs variables')
sns.distplot(np.random.choice(spri['count'],840),ax=axes[0, 0])
sns.distplot(np.random.choice(wint['count'],840),ax=axes[0, 1])
sns.distplot(np.random.choice(summ['count'],840),ax=axes[1, 0])
sns.distplot(np.random.choice(fall['count'],840),ax=axes[1, 1])
plt.show()
```



Above plots are not normal distribution

In [86]:

```
fitted_data1, fitted_lambda1 = stats.boxcox(np.random.choice(spri['count'],840))
fitted_data2, fitted_lambda2 = stats.boxcox(np.random.choice(wint['count'],840))
fitted_data3, fitted_lambda3 = stats.boxcox(np.random.choice(summ['count'],840))
fitted_data4, fitted_lambda4 = stats.boxcox(np.random.choice(fall['count'],840))
# Converting data to normal distribution with help of boxcox transformations
```

In [87]:

```
np.mean(fitted_data1),np.mean(fitted_data2),np.mean(fitted_data3),np.mean(fitted_data4)
# Means of the above data are almost equal
```

Out[87]:

```
(8.377222224059773, 16.180830048053117, 11.348862419170347, 22.4326176043920
46)
```

In [88]:

```
np.var(fitted_data1),np.var(fitted_data2),np.var(fitted_data3),np.var(fitted_data4)
# Variances of the above data are almost equal
```

Out[88]:

```
(20.3521147148289, 62.95998372783924, 31.961658873295573, 125.9804311409545
2)
```

In [89]:

```
ans=df.groupby([df['season']])[['count']].sum().reset_index()
```

In [90]:

```
chi2, p, dof, ex=stats.chi2_contingency(ans['count'])
# Chi2 test to check if count is independent with respect to season
```

In [91]:

```
chi2, p, dof, ex
```

Out[91]:

```
(0.0, 1.0, 0, array([551274., 300999., 515803., 491901.]))
```

In [92]:

```
# As p-Value is > 0.05 we conclude that the H0 is true. (Values are independent)
```

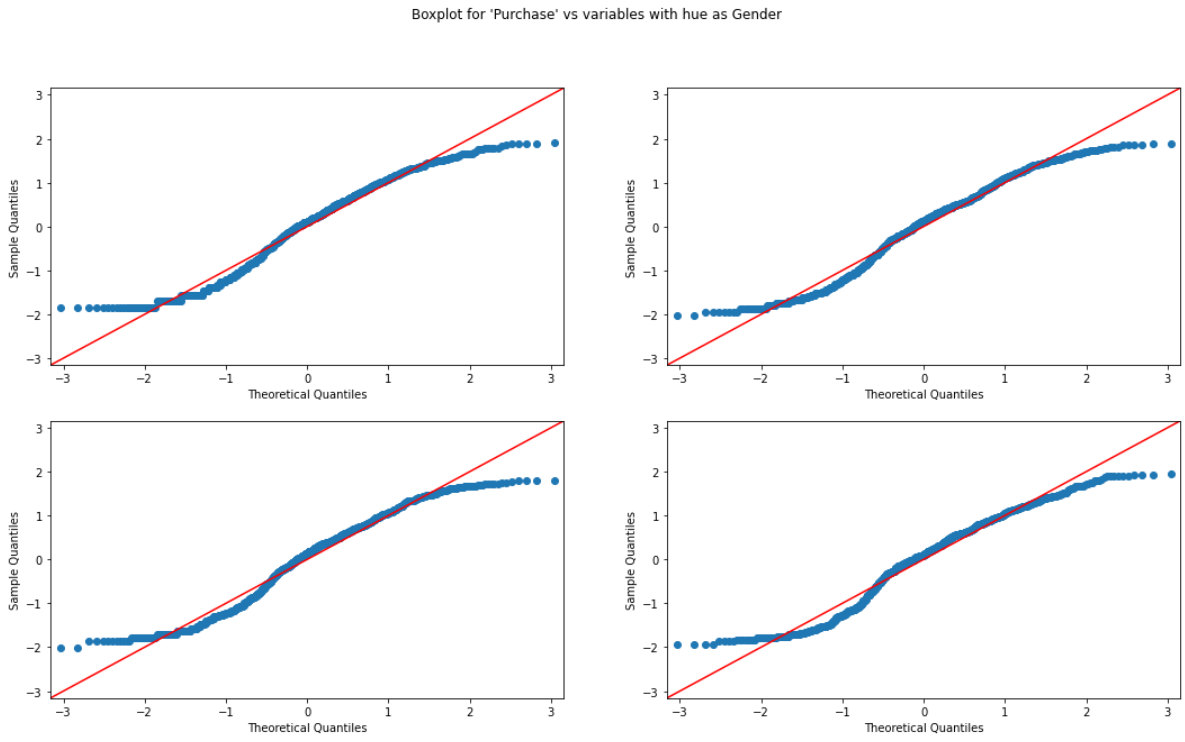
In [93]:

```
print(stats.ks_2samp(fitted_data1,stats.norm.rvs(loc=np.mean(fitted_data1),scale=np.std(fit
print(stats.ks_2samp(fitted_data2,stats.norm.rvs(loc=np.mean(fitted_data2),scale=np.std(fit
print(stats.ks_2samp(fitted_data3,stats.norm.rvs(loc=np.mean(fitted_data3),scale=np.std(fit
print(stats.ks_2samp(fitted_data4,stats.norm.rvs(loc=np.mean(fitted_data4),scale=np.std(fit
# KS test to check if they are normal
# P-values are not close to 0.05 and are lesser and they are not gaussian distribution,
#but still looks like normal distribution
```

```
KstestResult(statistic=0.06428571428571428, pvalue=0.06212320295340358)
KstestResult(statistic=0.075, pvalue=0.017712691814970632)
KstestResult(statistic=0.0869047619047619, pvalue=0.003499549166076452)
KstestResult(statistic=0.08452380952380953, pvalue=0.004933897825377105)
```

In [94]:

```
fig, axes = plt.subplots(2, 2, figsize=(18, 10))

fig.suptitle('Boxplot for \'Purchase\' vs variables with hue as Gender')
sm.qqplot(fitted_data1,ax=axes[0, 0],line='45',fit=True)
sm.qqplot(fitted_data2,ax=axes[0, 1],line='45',fit=True)
sm.qqplot(fitted_data3,ax=axes[1, 0],line='45',fit=True)
sm.qqplot(fitted_data4,ax=axes[1, 1],line='45',fit=True)
plt.show()
# Below is the QQ plot for the data. Plot looks like normal distribution
```



In [95]:

```
stats.f_oneway(fitted_data1,fitted_data2,fitted_data3,fitted_data4)
```

Out[95]:

```
F_onewayResult(statistic=524.6205973018104, pvalue=1.4785115164657819e-279)
```

In [96]:

```
stats.f_oneway(spri['count'],wint['count'],summ['count'],fall['count'])
```

Out[96]:

```
F_onewayResult(statistic=322.6531333835799, pvalue=2.3242495315066624e-200)
```

++

Null hypothesis: The mean of count is equal for spring ,winter, fall and summer
Alternate hypothesus:The mean of count not equal for spring ,winter, fall and summer
Significance value= 5%
p-value <= 0.05, Then we have enough evidance to reject Null hypothesis
p-value > 0.05, We fail to reject null hypothesis
Sample mean and variance are finite and closely follow normal distribution
Result: As pvalue < 0.05 we conclude the mean of count not equal for spring ,winter, fall and summer
T stat is shown above

ANOVA test for testing means of count for differents weathers

In [97]:

```
# ANOVA
```

In [98]:

```
df['weather'].value_counts()
# Sample sizes are not equal hence we do a randon choice of 840
```

Out[98]:

```
Clear         6965
Mist          2770
Light Rain     850
Name: weather, dtype: int64
```

In [99]:

```
cle=df.loc[df['weather']=='Clear']
mis=df.loc[df['weather']=='Mist']
lig=df.loc[df['weather']=='Light Rain']
```

In [100]:

```
len(outliners(cle,'count'))*100/len(cle),len(outliners(mis,'count'))*100/len(mis),len(outli
```

Out[100]:

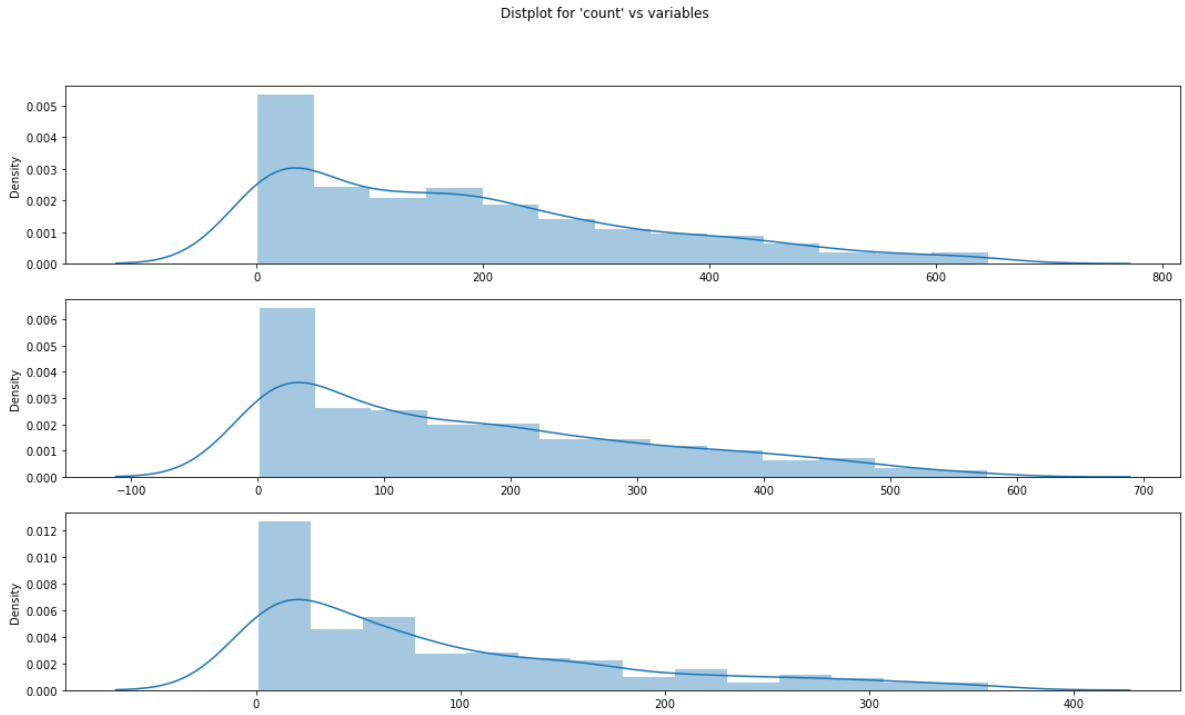
```
(0.0, 1.0830324909747293, 5.882352941176471)
```

In [101]:

```
cle.drop(outliners(cle,'count'),axis=0,inplace=True)
mis.drop(outliners(mis,'count'),axis=0,inplace=True)
lig.drop(outliners(lig,'count'),axis=0,inplace=True)
# Outliners are removed as they are 1.08 % and 5.88 %
```

In [102]:

```
fig, axes = plt.subplots(3, figsize=(18, 10))

fig.suptitle('Distplot for \'count\' vs variables')
sns.distplot(np.random.choice(cle['count'],840),ax=axes[0])
sns.distplot(np.random.choice(mis['count'],840),ax=axes[1])
sns.distplot(np.random.choice(lig['count'],840),ax=axes[2])

plt.show()
```



Above plots are not normal distribution and are right skewed

In [103]:

```
fitted_data1, fitted_lambda1 = stats.boxcox(np.random.choice(cle['count'],840))
fitted_data2, fitted_lambda2 = stats.boxcox(np.random.choice(mis['count'],840))
fitted_data3, fitted_lambda3 = stats.boxcox(np.random.choice(lig['count'],840))
# Applying boxcox transformations
```

In [104]:

```
np.var(fitted_data1),np.var(fitted_data2),np.var(fitted_data3),np.var(fitted_data4)
# Variances of the above data are almost equal
```

Out[104]:

```
(38.58678469841408, 35.92017230887363, 17.012770882559284, 125.9804311409545
2)
```

In [105]:

```
np.mean(fitted_data1),np.mean(fitted_data2),np.mean(fitted_data3)
# Means of the above data are almost equal
```

Out[105]:

```
(12.282000557438709, 12.13586369000345, 7.962747029325681)
```

In [106]:

```
print(stats.ks_2samp(fitted_data1,stats.norm.rvs(loc=np.mean(fitted_data1),scale=np.std(fit
print(stats.ks_2samp(fitted_data2,stats.norm.rvs(loc=np.mean(fitted_data2),scale=np.std(fit
print(stats.ks_2samp(fitted_data3,stats.norm.rvs(loc=np.mean(fitted_data3),scale=np.std(fit
# KS test to check if they are normal
# P-values are not close to 0.05 and are lesser and they are not gaussian distribution,
#but still looks like normal distribution
```

```
KstestResult(statistic=0.07738095238095238, pvalue=0.013054206218996531)
KstestResult(statistic=0.06904761904761905, pvalue=0.03642790227440366)
KstestResult(statistic=0.06547619047619048, pvalue=0.054557875353374426)
```

In [107]:

```
ans=df.groupby([df['weather']])[['count']].sum().reset_index()
```

In [108]:

```
chi2, p, dof, ex=stats.chi2_contingency(ans['count'])
# Chi2 test to check if count is independent with respect to weather
```

In [109]:

```
chi2, p, dof, ex
```

Out[109]:

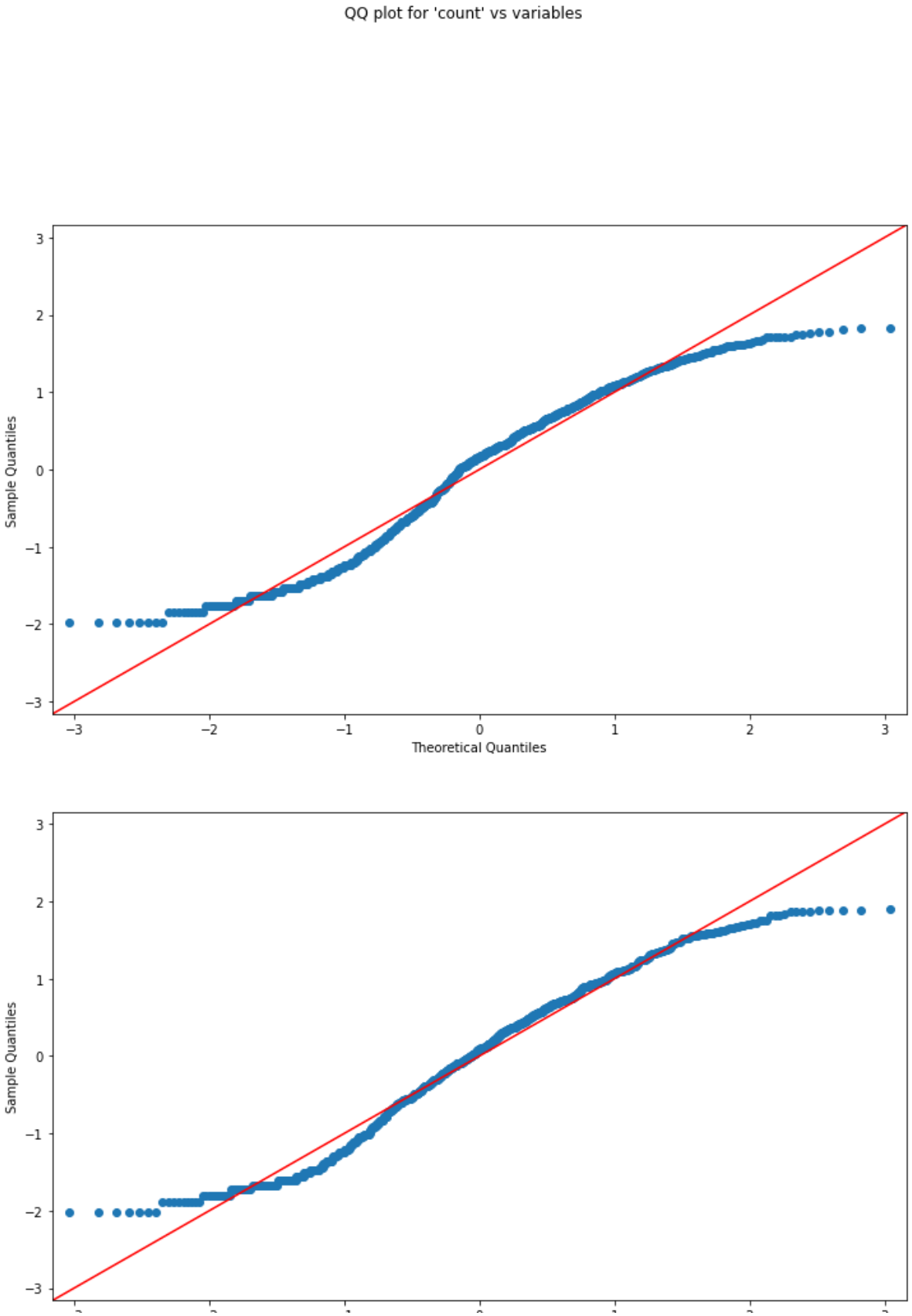
```
(0.0, 1.0, 0, array([1304748.,   95083.,  460146.]))
```

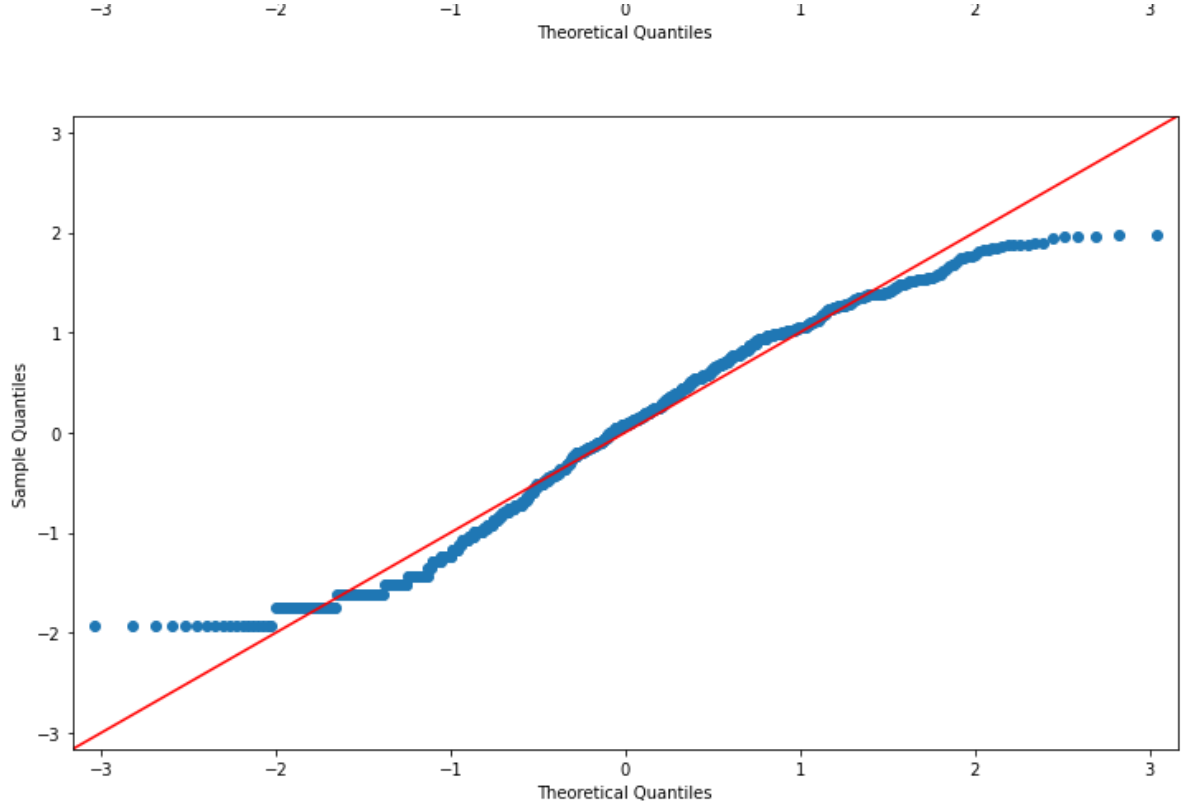
In [110]:

```
# As p-Value is > 0.05 we conclude that the H0 is true. (Values are independent)
```

In [111]:

```
fig, axes = plt.subplots(3, figsize=(12, 24))

fig.suptitle('QQ plot for \'count\' vs variables ')
sm.qqplot(fitted_data1,ax=axes[0],line='45',fit=True)
sm.qqplot(fitted_data2,ax=axes[1],line='45',fit=True)
sm.qqplot(fitted_data3,ax=axes[2],line='45',fit=True)
plt.show()
# Below is the QQ plot for the data. Plot looks like normal distribution
```





In [112]:

```
stats.f_oneway(fitted_data1,fitted_data2,fitted_data3)
```

Out[112]:

```
F_onewayResult(statistic=165.43626508856718, pvalue=3.143539828592585e-68)
```

++
Null hypothesis: The mean of count is equal for clear, Mist and Light rain
Alternate hypothesus:The mean of count not equal for clear, Mist and Light rain
Significance value= 5%
p-value <= 0.05, Then we have enough evidance to reject Null hypothesis
p-value > 0.05, We fail to reject null hypothesis
Sample mean and variance are finite and closely follow normal distribution
Result: As pvalue < 0.05 we conclude the mean of count not equal for clear, Mist and Light rain
T stat is shown above

Chi square (Test of Independence)

In [113]:

```
# To check the total count of weather and season are independent
```

In [114]:

```
dff=df.groupby([df['season'],df['weather']])[['count']].sum().reset_index()
```

In [115]:

```
ans=dff.pivot(index='season', columns='weather')
```

In [116]:

```
ans
```

Out[116]:

	count		
weather	Clear	Light Rain	Mist
season			
Fall	402602	27883	120789
Spring	212386	12919	75694
Summer	367404	26973	121426
Winter	322356	27308	142237

In [117]:

```
chi2, p, dof, ex=stats.chi2_contingency(ans)
```

In [118]:

```
chi2, p, dof, ex
```

Out[118]:

```
(8409.547609716874,
 0.0,
 6,
 array([[386711.04478819,  28181.4160831 , 136381.53912871],
        [211146.61270112,  15387.22678668,  74465.1605122 ],
        [361828.6315605 ,  26368.11995471, 127606.24848479],
        [345061.71095019,  25146.23717551, 121693.0518743 ]]))
```

++

Null hypothesis: The count is independent of season and weather
Alternate hypothesus:The count is not independent of season and weather
Significance value= 5%
p-value <= 0.05, Then we have enough evidance to reject Null hypothesis
p-value > 0.05, We fail to reject null hypothesis
Result: As pvalue < 0.05 we conclude that count is not independent of season and weather
T stat is shown above

2-Way ANOVA

In [119]:

```
model = ols('count ~ C(weather) + C(season) +C(weather):C(season)',data=df).fit()
result = sm.stats.anova_lm(model, type=2)
```

In [120]:

```
result
```

Out[120]:

	df	sum_sq	mean_sq	F	PR(>F)
C(weather)	2.0	4.660236e+06	2.330118e+06	103.104132	4.503016e-45
C(season)	3.0	1.488695e+07	4.962318e+06	219.574923	3.448572e-138
C(weather):C(season)	6.0	2.934311e+05	4.890518e+04	2.163979	4.338083e-02
Residual	10573.0	2.389462e+08	2.259966e+04	NaN	NaN

Weather: p-value = 4.503016e-45
Season: p-value = 3.448572e-138
Weather*Season: p-value = 4.338083e-02

Hypothesis for Weather: Ho : Weather has no statistically significant effect on eletrical cycles count Ha : Weather has statistically significant effect on eletrical cycles count

As p-Value <0.05, we conclude that Weather has statistically significant effect on eletrical cycles count

Hypothesis for Season: Ho : Season has no statistically significant effect on eletrical cycles count Ha : Season has statistically significant effect on eletrical cycles count

As p-Value <0.05, we conclude that season has statistically significant effect on eletrical cycles count

In [121]:

```
def conf(para):
    print('Confidence interval using Bootstrap : ')
    per=[0.05,0.025,0.005]
    for a in df[para].unique():
        data=df[df[para]==a]
        ls=[]
        for i in range(10000):
            ans=np.random.choice(data['count'],len(data),replace=True)
            l1=np.mean(ans)
            ls.append(l1)
        #print('Confidence interval using Bootstrap : ')
        #print('Confidence interval for {} {} group is [{},{}] with {}% confidence'.format(
        print('Confidence interval for {} {} group is [{},{}] with {}% confidence'.format(a
        #print('Confidence interval for {} {} group is [{},{}] with {}% confidence'.format(
```

In [122]:

```
conf('season')
```

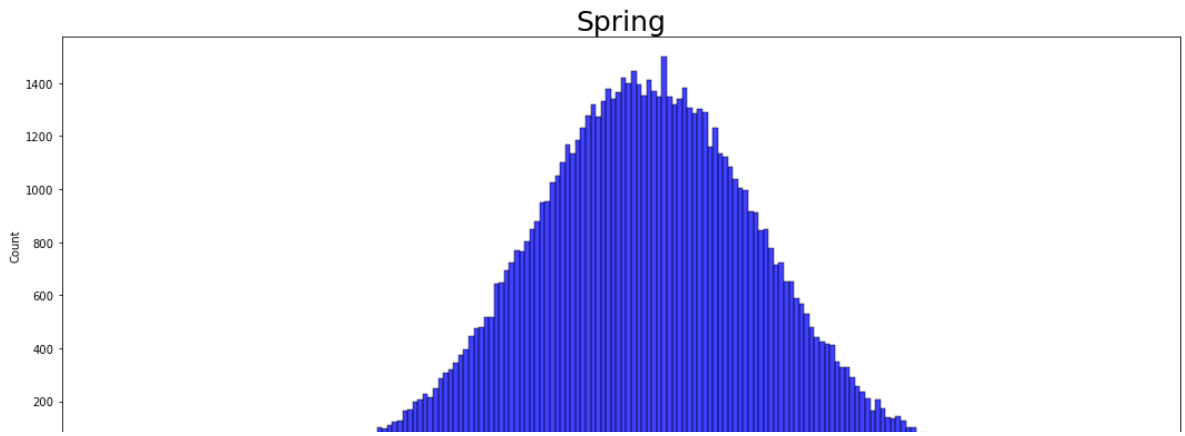
```
Confidence interval using Bootstrap :
Confidence interval for Spring season group is [108.30759647808168,117.31071
562382914] with 95.0% confidence
Confidence interval for Summer season group is [189.48500379650721,202.20446
08959757] with 95.0% confidence
Confidence interval for Fall season group is [204.40768055024836,216.9877531
5246465] with 95.0% confidence
Confidence interval for Winter season group is [178.67913696060037,190.38357
4108818] with 95.0% confidence
```

In [123]:

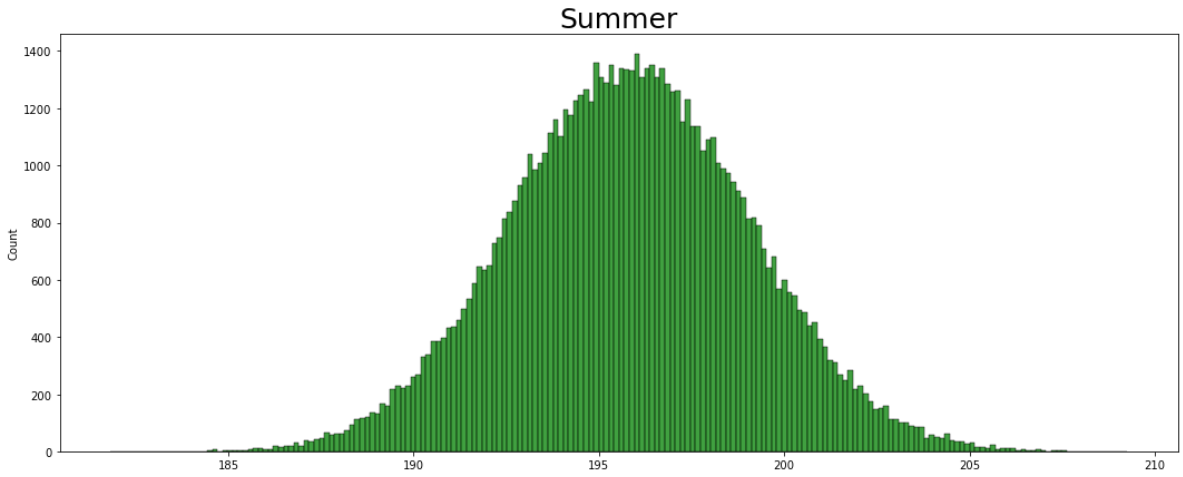
```
data=df[df['season']=='Spring']
ls=[]
for i in range(80000):
    ans=np.random.choice(data['count'],len(data),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
fig, axes = plt.subplots(5,1, figsize=(18, 40))
plt.subplot(5,1,1)
sns.histplot(ls,color='b',ax=axes[0],bins=200)
sns.histplot(ls,color='b',ax=axes[4],alpha=0.5,bins=200)
axes[0].set_title("Spring",fontsize=25)
data=df[df['season']=='Summer']
ls=[]
for i in range(80000):
    ans=np.random.choice(data['count'],len(data),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(5,1,2)
sns.histplot(ls,color='g',ax=axes[1],bins=200)
sns.histplot(ls,color='g',ax=axes[4],alpha=0.7,bins=200)
axes[1].set_title("Summer",fontsize=25)
data=df[df['season']=='Fall']
ls=[]
for i in range(80000):
    ans=np.random.choice(data['count'],len(data),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(5,1,3)
sns.histplot(ls,color='r',ax=axes[2],bins=200)
sns.histplot(ls,color='r',ax=axes[4],alpha=0.9,bins=200)
axes[2].set_title("Fall",fontsize=25)
data=df[df['season']=='Winter']
ls=[]
for i in range(80000):
    ans=np.random.choice(data['count'],len(data),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(5,1,4)
sns.histplot(ls,color='c',ax=axes[3],bins=200)
sns.histplot(ls,color='c',ax=axes[4],alpha=0.9,bins=200)
axes[3].set_title("Winter",fontsize=25)
axes[4].set_title("Combined",fontsize=25)
```

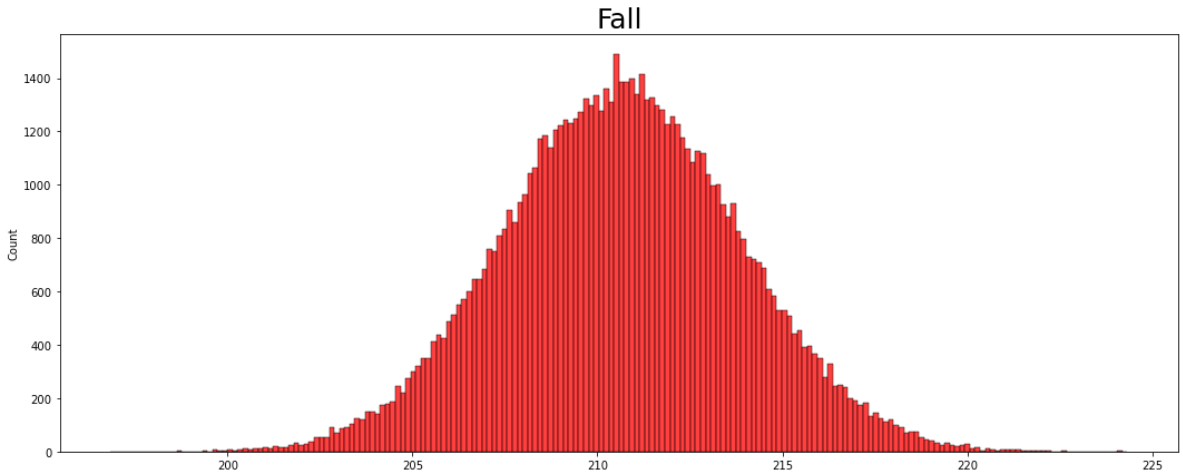
Out[123]:

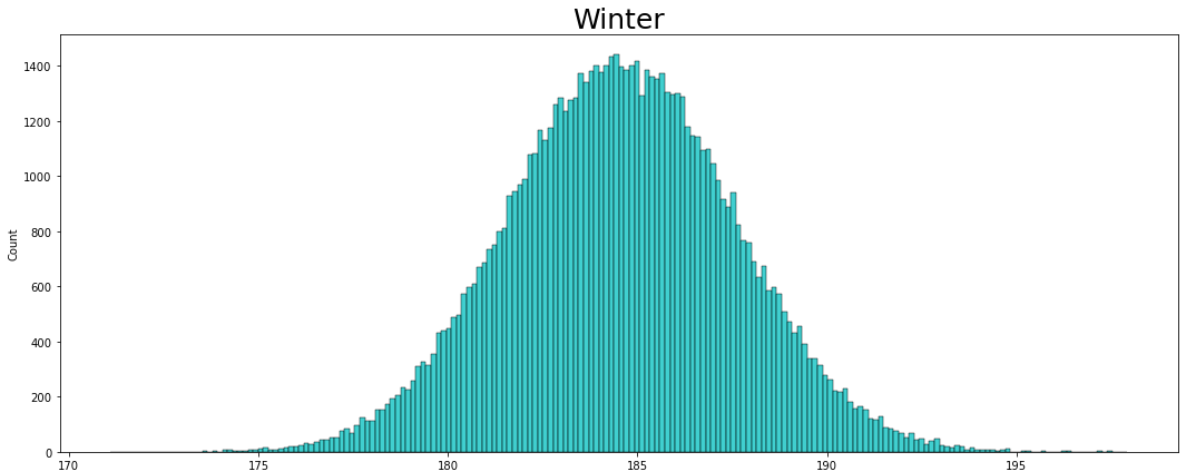
```
Text(0.5, 1.0, 'Combined')
```

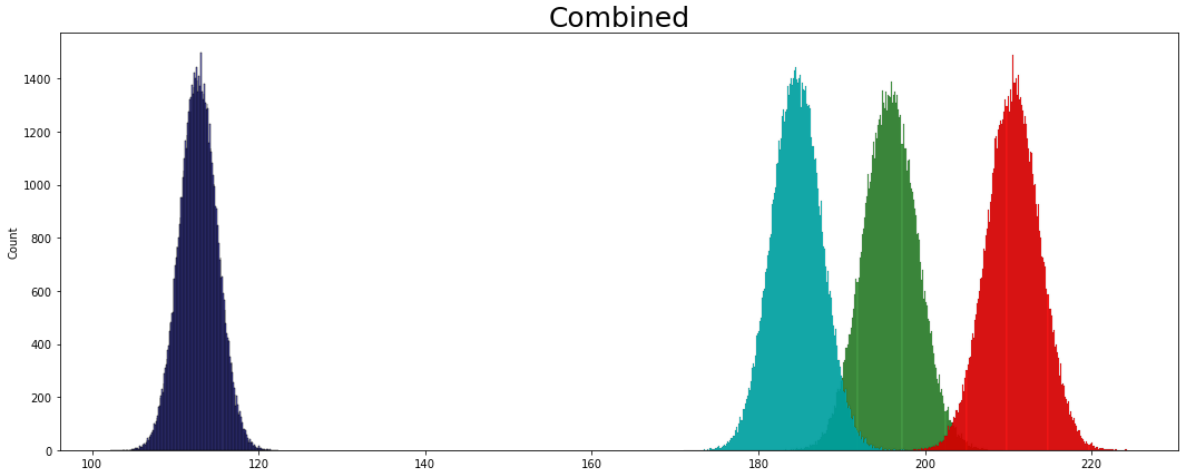



0
102.5
105.0
107.5
110.0
112.5
115.0
117.5
120.0
122.5


Summer
Count
0
200
400
600
800
1000
1200
1400
185
190
195
200
205
210


Fall
Count
0
200
400
600
800
1000
1200
1400
200
205
210
215
220
225


Winter
Count
0
200
400
600
800
1000
1200
1400
170
175
180
185
190
195


Combined
Count
0
200
400
600
800
1000
1200
1400
100
120
140
160
180
200
220

Hist plot for each season group.
Spring confidence interval does not coincide with any other group
Summer, winter and Fall confidence interval are overlapping.

In [124]:

```
conf('weather')
```

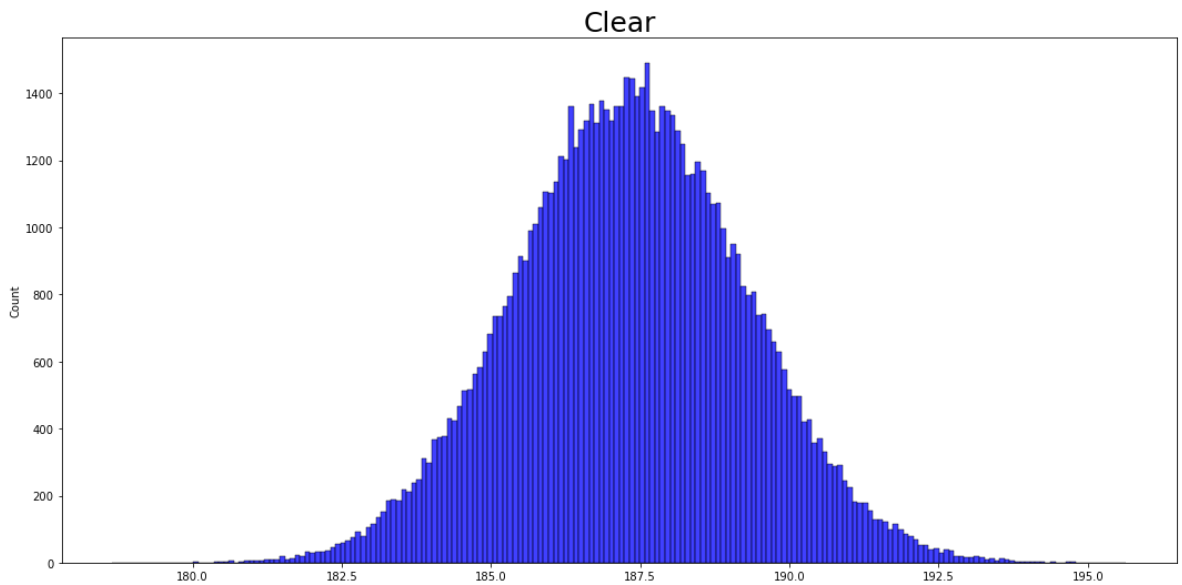
```
Confidence interval using Bootstrap :
Confidence interval for Clear weather group is [183.52671213208902,191.09606
24551328] with 95.0% confidence
Confidence interval for Mist weather group is [160.68084837545126,171.540162
45487367] with 95.0% confidence
Confidence interval for Light Rain weather group is [103.76114705882354,120.
22708823529412] with 95.0% confidence
```

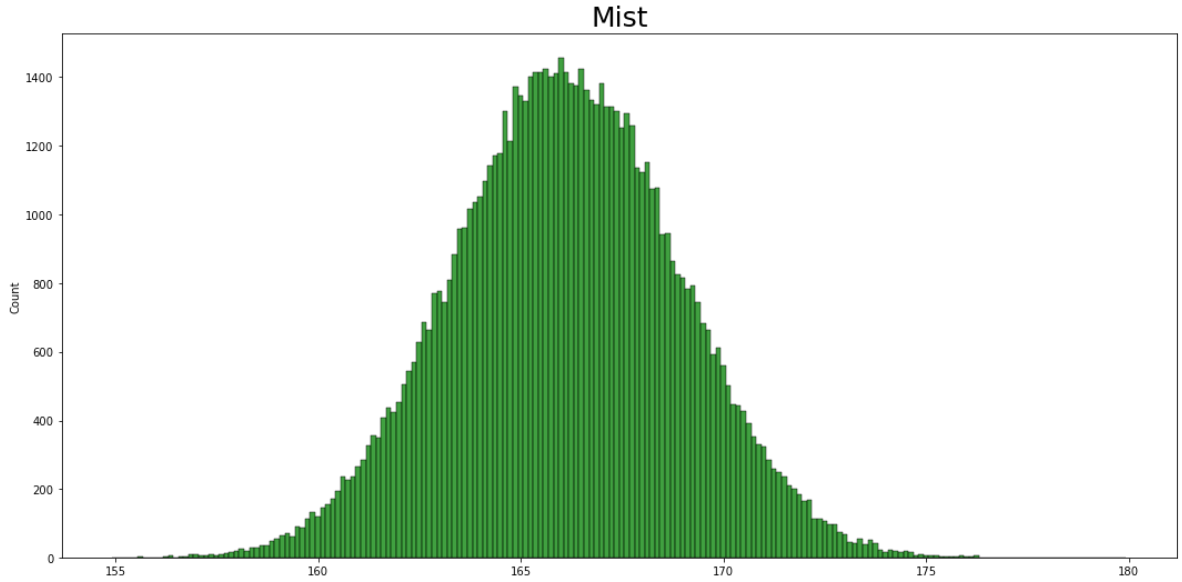
In [125]:

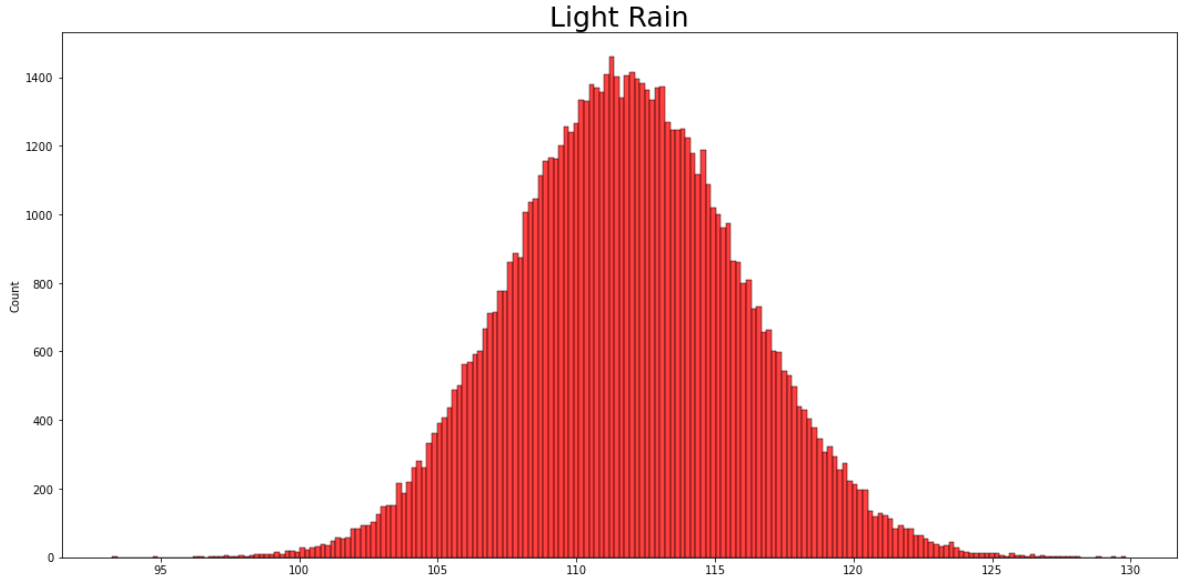
```
data=df[df['weather']=='Clear']
ls=[]
for i in range(80000):
    ans=np.random.choice(data['count'],len(data),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
fig, axes = plt.subplots(4,1, figsize=(18, 40))
plt.subplot(4,1,1)
sns.histplot(ls,color='b',ax=axes[0],bins=200)
sns.histplot(ls,color='b',ax=axes[3],alpha=0.5,bins=200)
axes[0].set_title("Clear",fontsize=25)


data=df[df['weather']=='Mist']
ls=[]
for i in range(80000):
    ans=np.random.choice(data['count'],len(data),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(4,1,2)
sns.histplot(ls,color='g',ax=axes[1],bins=200)
sns.histplot(ls,color='g',ax=axes[3],alpha=0.7,bins=200)
axes[1].set_title("Mist",fontsize=25)


data=df[df['weather']=='Light Rain']
ls=[]
for i in range(80000):
    ans=np.random.choice(data['count'],len(data),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(4,1,3)
sns.histplot(ls,color='r',ax=axes[2],bins=200)
sns.histplot(ls,color='r',ax=axes[3],alpha=0.9,bins=200)
axes[2].set_title("Light Rain",fontsize=25)
axes[3].set_title("Combined",fontsize=25)
```

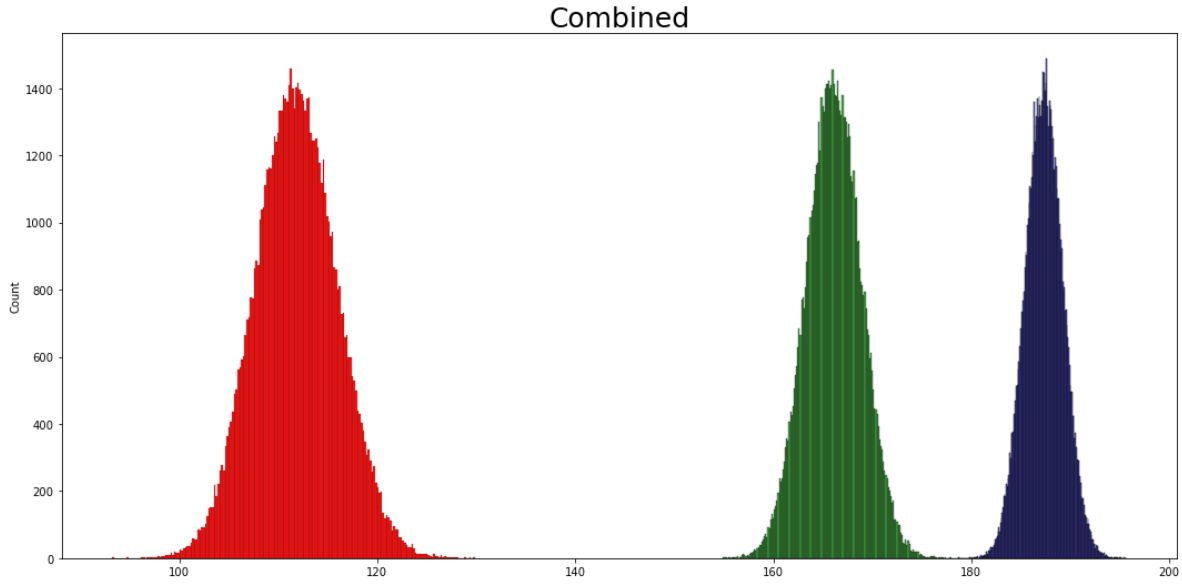
Out[125]:

```
Text(0.5, 1.0, 'Combined')
```









Hist plot for each weather group.

clear ,mist and light rain confidence interval are not overlapping.

In [126]:

```
conf('holiday')
```

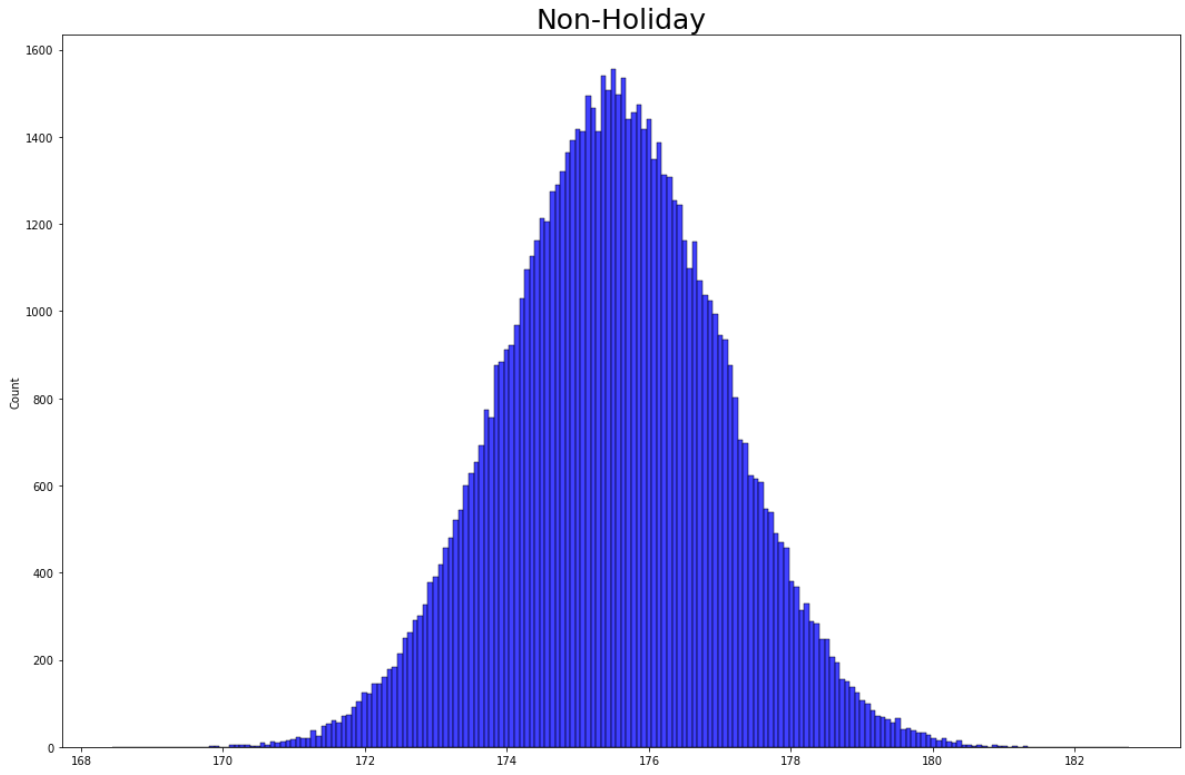
```
Confidence interval using Bootstrap :
Confidence interval for Non-Holiday holiday group is [172.4988589918256,178.
53687232386142] with 95.0% confidence
Confidence interval for Holiday holiday group is [164.44959546925566,200.858
33333333332] with 95.0% confidence
```

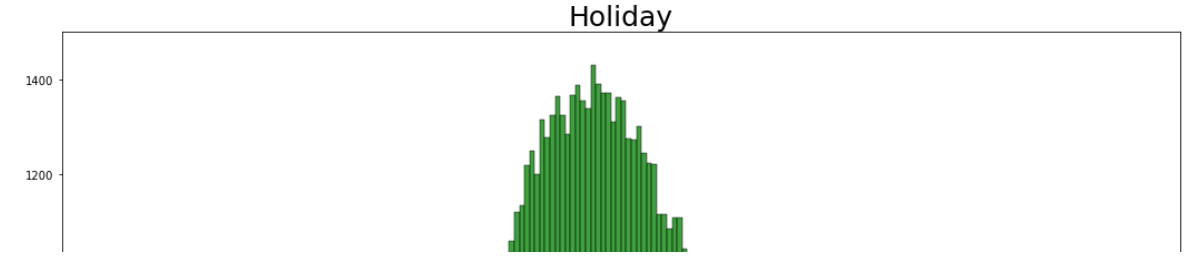
In [127]:

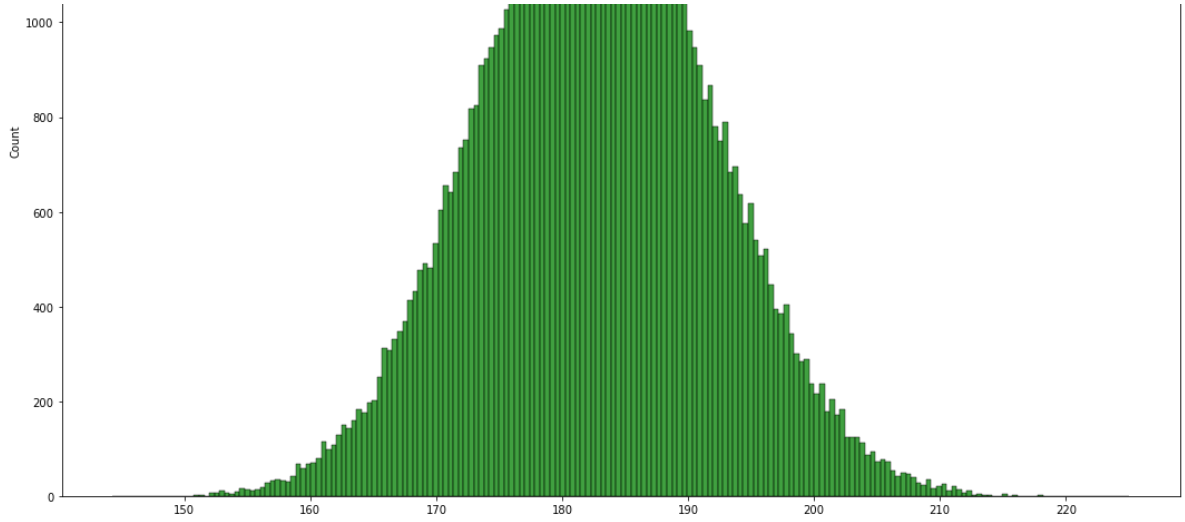
```
data=df[df['holiday']=='Non-Holiday']
ls=[]
for i in range(80000):
    ans=np.random.choice(data['count'],len(data),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
fig, axes = plt.subplots(3,1, figsize=(18, 40))
plt.subplot(3,1,1)
sns.histplot(ls,color='b',ax=axes[0],bins=200)
sns.histplot(ls,color='b',ax=axes[2],alpha=0.5,bins=200)
axes[0].set_title("Non-Holiday",fontsize=25)
data=df[df['holiday']=='Holiday']
ls=[]
for i in range(80000):
    ans=np.random.choice(data['count'],len(data),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(3,1,2)
sns.histplot(ls,color='g',ax=axes[1],bins=200)
sns.histplot(ls,color='g',ax=axes[2],alpha=0.7,bins=200)
axes[1].set_title("Holiday",fontsize=25)
axes[2].set_title("Combined",fontsize=25)
```

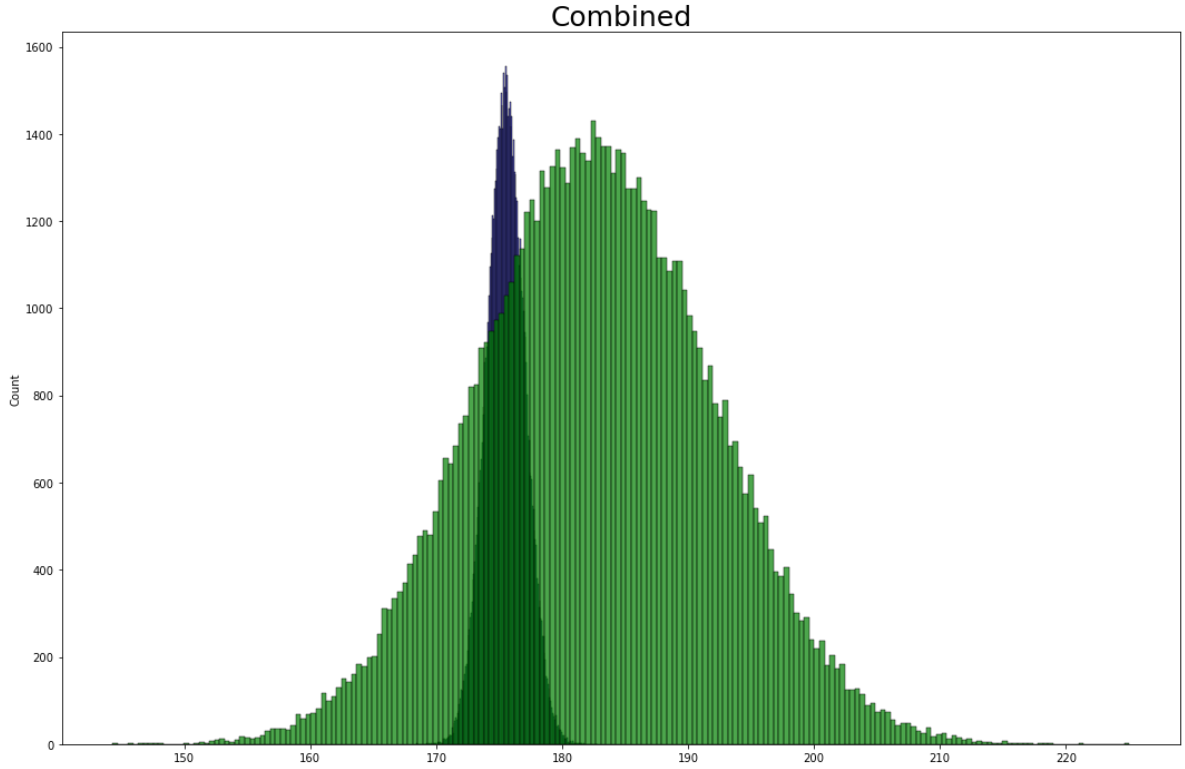
Out[127]:

```
Text(0.5, 1.0, 'Combined')
```









Hist plot for each holiday group.

Holiday and non-holiday confidence interval are overlapping completely.

In [128]:

```
conf('workingday')
```

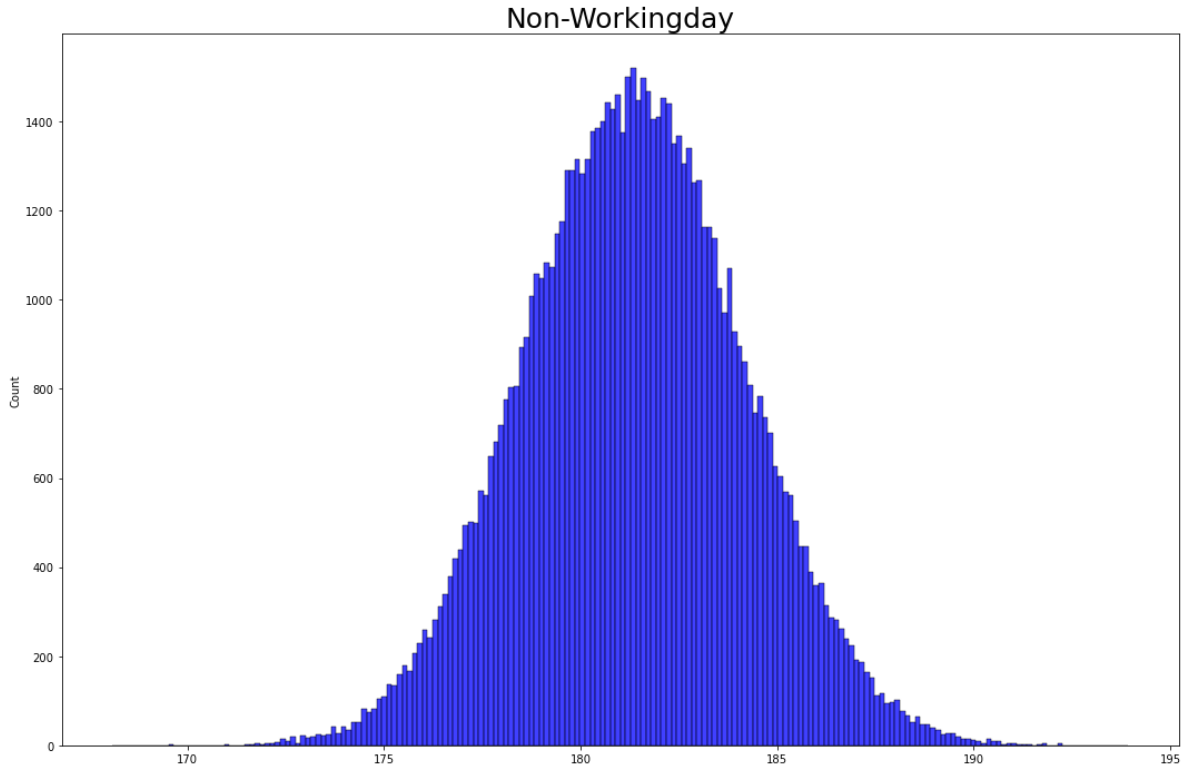
```
Confidence interval using Bootstrap :
Confidence interval for Non-Workingday workingday group is [175.851145985401
47,186.84342335766422] with 95.0% confidence
Confidence interval for Workingday workingday group is [169.48717877094973,1
76.50942737430168] with 95.0% confidence
```

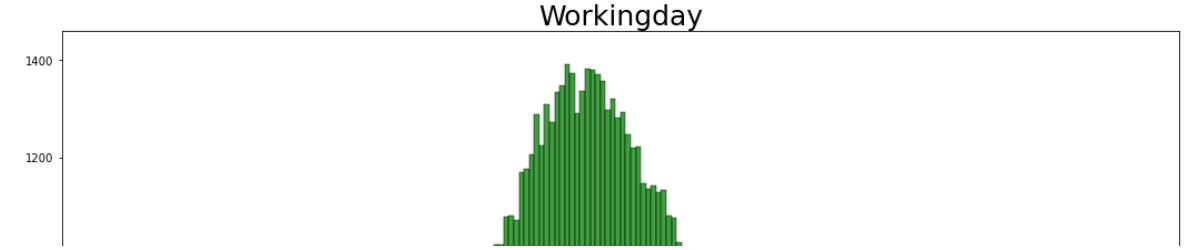
In [129]:

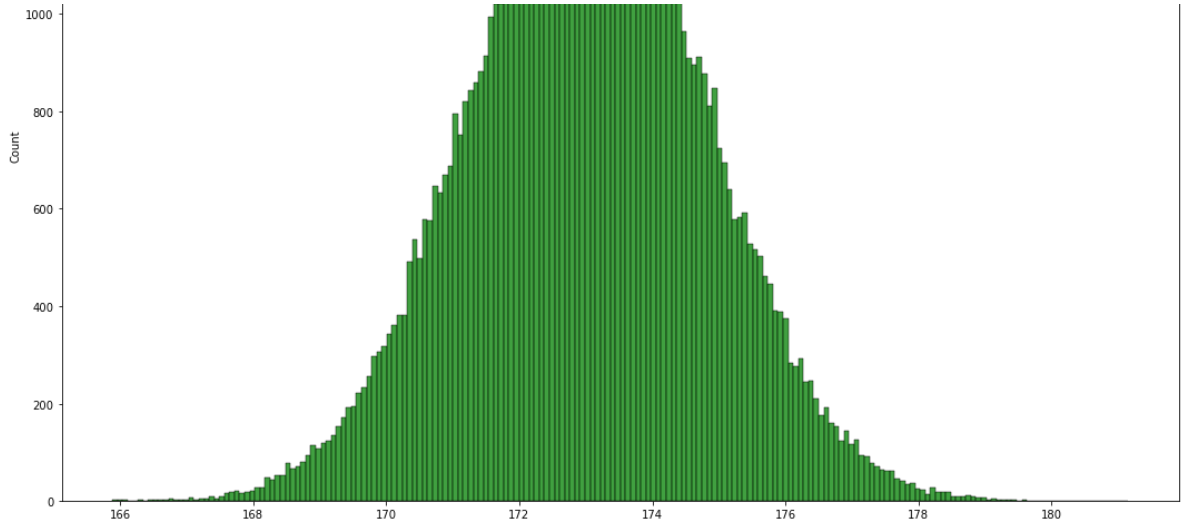
```
data=df[df['workingday']=='Non-Workingday']
ls=[]
for i in range(80000):
    ans=np.random.choice(data['count'],len(data),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
fig, axes = plt.subplots(3,1, figsize=(18, 40))
plt.subplot(3,1,1)
sns.histplot(ls,color='b',ax=axes[0],bins=200)
sns.histplot(ls,color='b',ax=axes[2],alpha=0.5,bins=200)
axes[0].set_title("Non-Workingday",fontsize=25)
data=df[df['workingday']=='Workingday']
ls=[]
for i in range(80000):
    ans=np.random.choice(data['count'],len(data),replace=True)
    l1=np.mean(ans)
    ls.append(l1)
plt.subplot(3,1,2)
sns.histplot(ls,color='g',ax=axes[1],bins=200)
sns.histplot(ls,color='g',ax=axes[2],alpha=0.7,bins=200)
axes[1].set_title("Workingday",fontsize=25)
axes[2].set_title("Combined",fontsize=25)
```

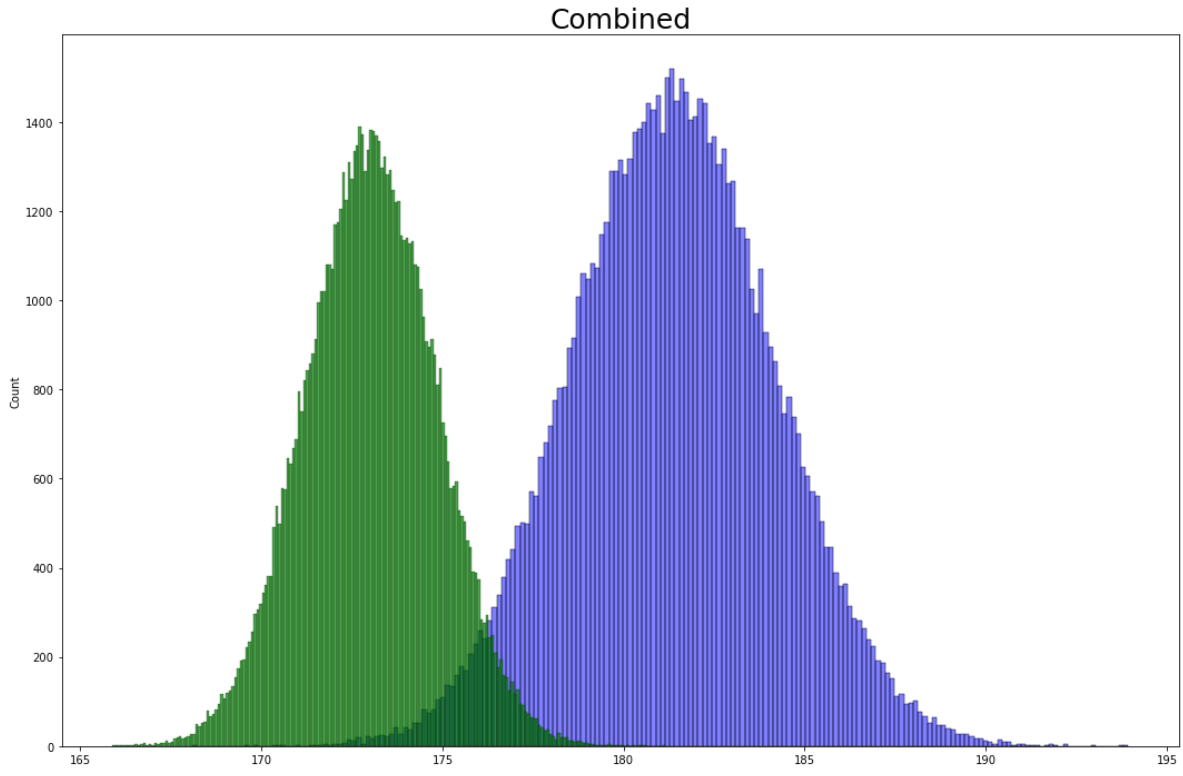
Out[129]:

```
Text(0.5, 1.0, 'Combined')
```









Hist plot for each working day group.
working day and non-working day confidence interval are not overlapping.