In [2]:
```python
import pandas as pd
import numpy as np
from numpy import nan, NaN,NAN
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
import scipy
warnings.filterwarnings("ignore")
from scipy import stats
import statsmodels.api as sm
```

In [3]:
```python
yulu=pd.read_csv("bike_sharing.txt")
```

In [4]:
```python
df=yulu.copy()
df.head(5)
```

Out[4]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

In [5]:
```python
df.shape#The Dataset has 10,886 rows with 12 columns
```

Out[5]:
```
(10886, 12)
```

In [6]:
```
df.info()#There are no missing values in dataset
```

```
<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 [7]:
```
df.isnull().sum()/len(df)*100
```

Out[7]:
```
datetime      0.0
season        0.0
holiday       0.0
workingday    0.0
weather       0.0
temp          0.0
atemp         0.0
humidity      0.0
windspeed     0.0
casual        0.0
registered    0.0
count         0.0
dtype: float64
```

Observation-No missing/null values

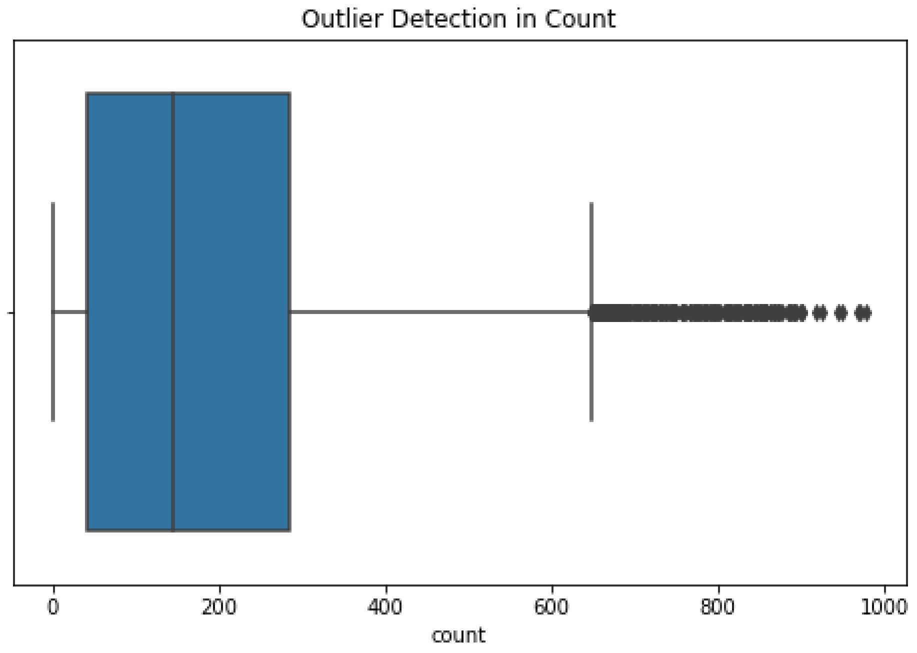
In [8]:
```
df.describe()
```

Out[8]:

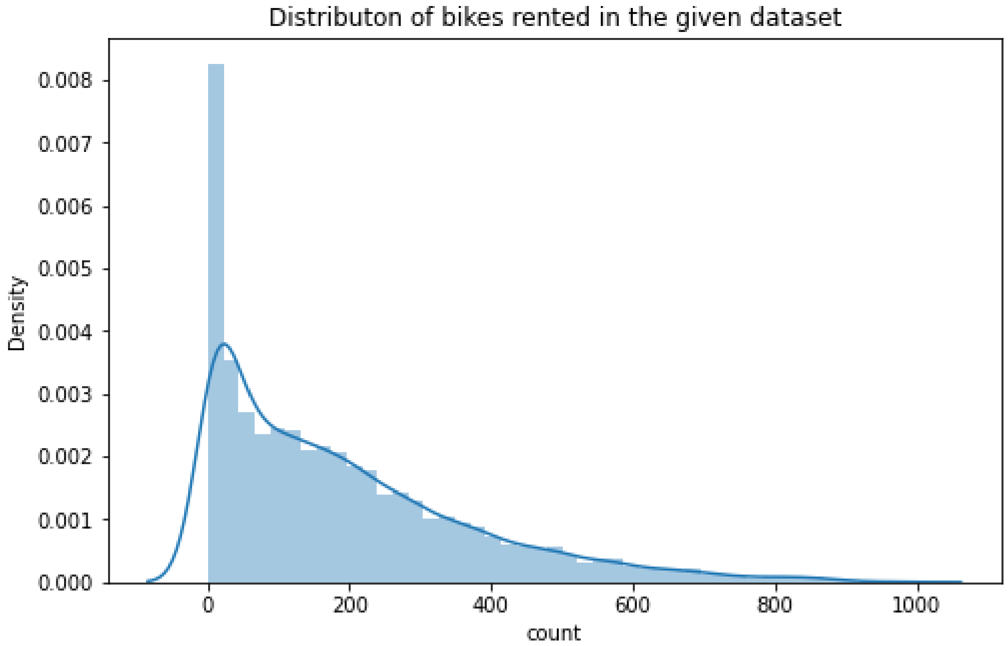
	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	
unt	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	108
an	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	1
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	1
nin	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	
5%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	
0%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	1
5%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	2
ax	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	9

In [16]:

```
plt.rcParams["figure.figsize"] = (8,5)
sns.boxplot(df["count"])
plt.title("Outlier Detection in Count")
plt.show()
```



In [18]:
```
plt.title("Distributon of bikes rented in the given dataset")
sns.distplot(df["count"])
plt.show()
```



The mean and median of the "count" is of the order in same magnitude.The outlier is

most likely due to the skewness in data.Hence lets not remove them.

In [47]:
```
#Categorical value conversions
#season
df["season"].replace({1:"Spring",2:"Summer",3:"Fall",4:"Winter"},inplace=True)
df["holiday"].replace({0:"No",1:"Yes"},inplace=True)
df["workingday"].replace({0:"No",1:"Yes"},inplace=True)
```

In [48]:
```
#Categorising weather into zones
"""Green-1: Clear, Few clouds, partly cloudy, partly cloudy
Yellow-2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
Orange-3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
Red-4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog"""
df.insert(12,"Zone",'')
```

In [49]:
```
df.loc[df["weather"]==1,"Zone"]="Green"
df.loc[df["weather"]==2,"Zone"]="Yellow"
df.loc[df["weather"]==3,"Zone"]="Orange"
df.loc[df["weather"]==4,"Zone"]="Red"
```

In [50]:
```python
#unique value
for col in ("season","holiday","workingday","Zone"):
    print(df[col].value_counts())
    print("-"*50)
```

```
Winter    2734
Summer    2733
Fall      2733
Spring    2686
Name: season, dtype: int64
--------------------------------------------------
No     10575
Yes      311
Name: holiday, dtype: int64
--------------------------------------------------
Yes    7412
No     3474
Name: workingday, dtype: int64
--------------------------------------------------
Green     7192
Yellow    2834
Orange     859
Red          1
Name: Zone, dtype: int64
--------------------------------------------------
```

In [51]:
```
#unique value
print("Percent Values")
print("*"*50)
for col in ("season","holiday","workingday","Zone"):
    print(df[col].value_counts(normalize=True)*100)
    print("-"*50)
```

```
Percent Values
**************************************************
Winter    25.114826
Summer    25.105640
Fall      25.105640
Spring    24.673893
Name: season, dtype: float64
--------------------------------------------------
No     97.14312
Yes     2.85688
Name: holiday, dtype: float64
--------------------------------------------------
Yes    68.087452
No     31.912548
Name: workingday, dtype: float64
--------------------------------------------------
Green     66.066507
Yellow    26.033437
Orange     7.890869
Red        0.009186
Name: Zone, dtype: float64
--------------------------------------------------
```

In [52]:
```
#Percent of rentals by casual and registered users
df.loc[:,"casual":"registered"].sum(axis=0)*100/df.loc[:,"count"].sum()
```

Out[52]:
```
casual        18.803141
registered    81.196859
dtype: float64
```

Observation- 81% of rentals are done by registered users and 19% by casual users.

In [53]:
```
#Datetime split two columns
#df.insert(1,"Date",'')
#df.insert(2,"Time","")
ts=pd.to_datetime(df["datetime"])
df["Date"]=ts.dt.date
df["Time"]=ts.dt.time
```

In [54]:
```
ts.dt.day.value_counts()#For what all days data is given
```

Out[54]:
```
1     575
9     575
17    575
5     575
16    574
15    574
14    574
13    574
19    574
8     574
7     574
4     574
2     573
12    573
3     573
6     572
10    572
11    568
18    563
Name: datetime, dtype: int64
```

In [55]:
```
ts.dt.year.unique()#for which years
```

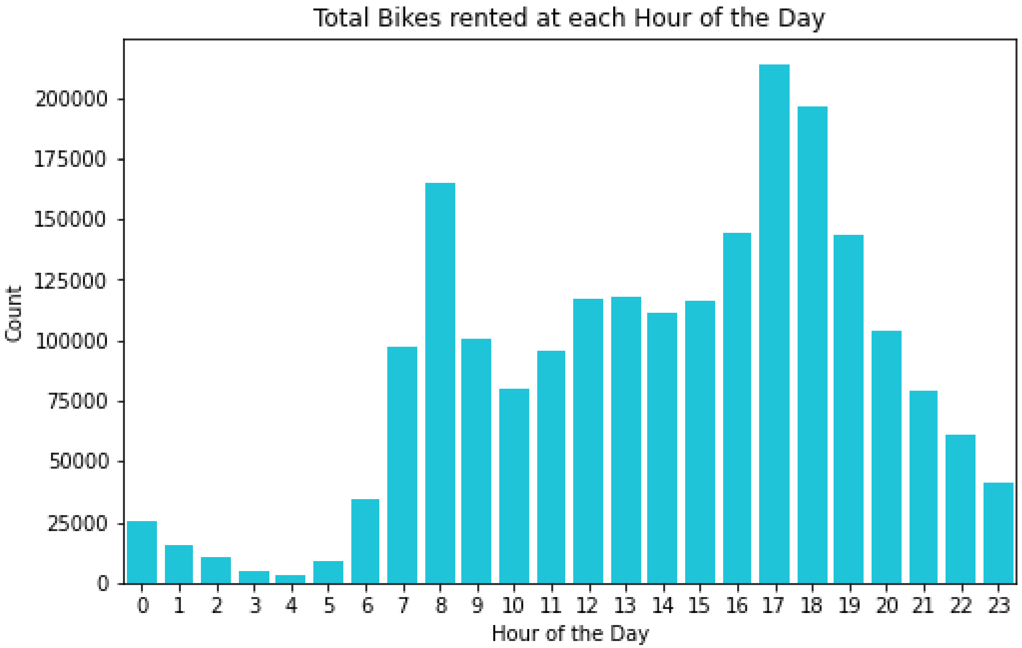
Out[55]:
```
array([2011, 2012], dtype=int64)
```

In [56]:
```
ts.dt.month.unique()#for which months
```

Out[56]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12], dtype=int64)
```

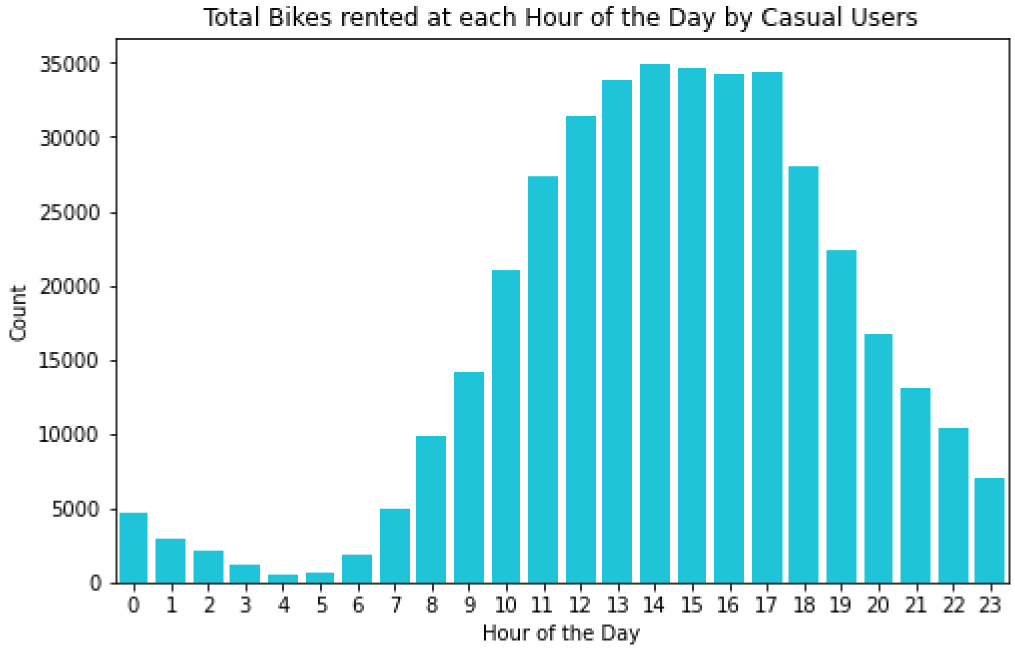
Observation-The data is given for the year 2011 and 2012 .Its for the first 19 days for each month .

In [57]:
```
y=df.groupby("Time")["count"].sum().to_list()
x=list(ts.dt.hour.unique())
plt.rcParams["figure.figsize"] = (8,5)
sns.barplot(x=x,y=y,color="#00DCF7")
plt.title("Total Bikes rented at each Hour of the Day")
plt.xlabel("Hour of the Day")
plt.ylabel("Count")
plt.show()
```

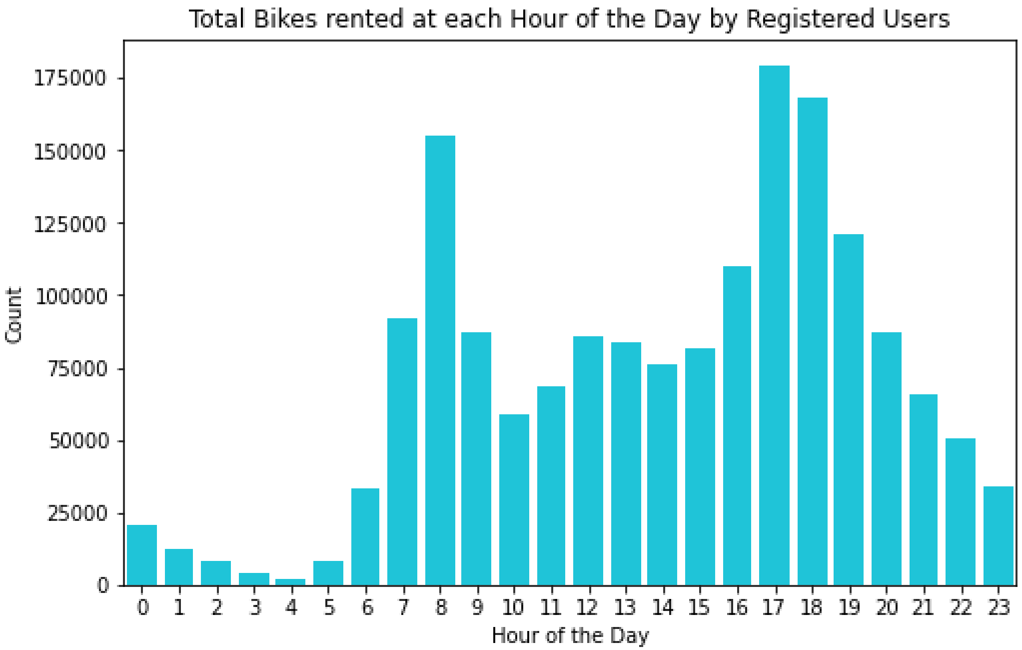


Inference-Significant number of bikes are rented between 4pm and 7pm.This is also observed in morning around 8pm.But an interesting fact is that morning hours people are not using Yulu bikes much as compared to evening.Lets check whether this is the trend followed by Casual and Registered Users

In [58]:
```
y=df.groupby("Time")["casual"].sum().to_list()
x=list(ts.dt.hour.unique())
plt.rcParams["figure.figsize"] = (8,5)
sns.barplot(x=x,y=y,color="#00DCF7")
plt.title("Total Bikes rented at each Hour of the Day by Casual Users")
plt.xlabel("Hour of the Day")
plt.ylabel("Count")
plt.show()
```



In [32]:
```
y=df.groupby("Time")["registered"].sum().to_list()
x=list(ts.dt.hour.unique())
plt.rcParams["figure.figsize"] = (8,5)
sns.barplot(x=x,y=y,color="#00DCF7")
plt.title("Total Bikes rented at each Hour of the Day by Registered Users")
plt.xlabel("Hour of the Day")
plt.ylabel("Count")
plt.show()
```

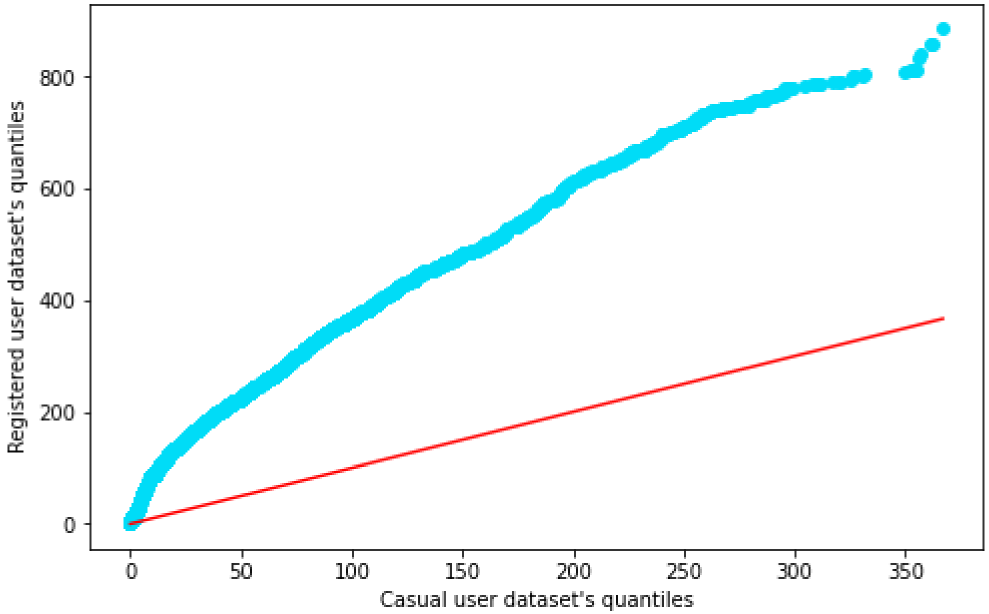


Inference-As per the plots the casual users tends to rent the bikes more between 12am and 5pm.Whereas for Registered users its in the morning between 7am and 9pm and also evening between 5pm and 7pm.Thus it can be concluded that a dip in usage of rented bikes in the morning when the total count is plottted is due to casual users

To check the Distribution of bikes rented by Registered and casual users .

In [60]:
```
y_cas=np.array(df["casual"])
y_reg=np.array(df["registered"])
```

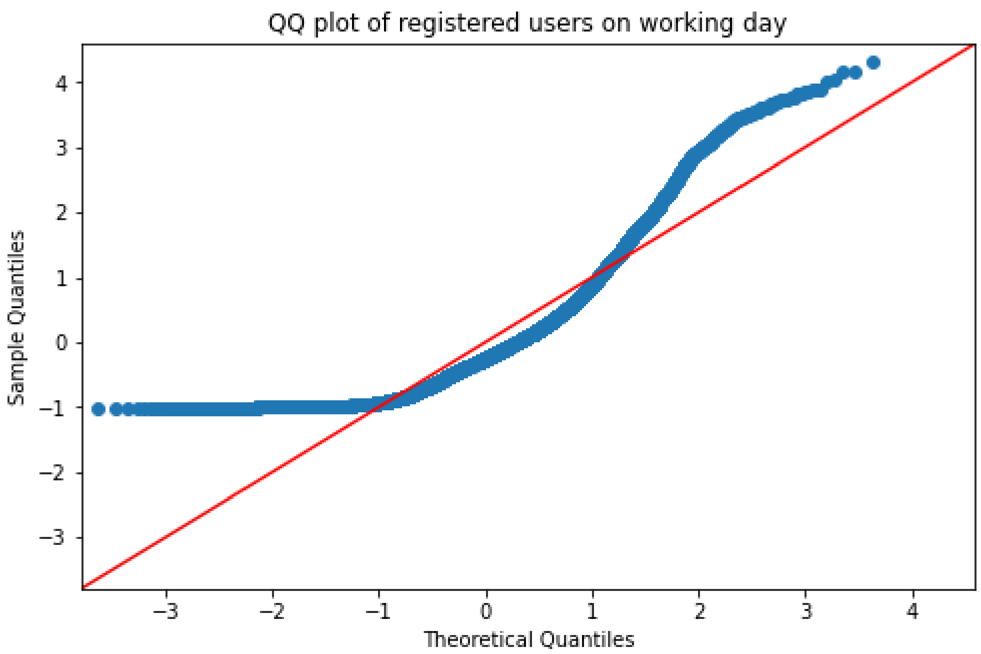
In [35]:
```
y_cas.sort()
y_reg.sort()
plt.scatter(y_cas,y_reg,color='#00DCF7')
plt.plot([min(y_cas),max(y_cas)],[min(y_cas),max(y_cas)],color="red")
plt.xlabel("Casual user dataset's quantiles")
plt.ylabel("Registered user dataset's quantiles")
plt.show()
```



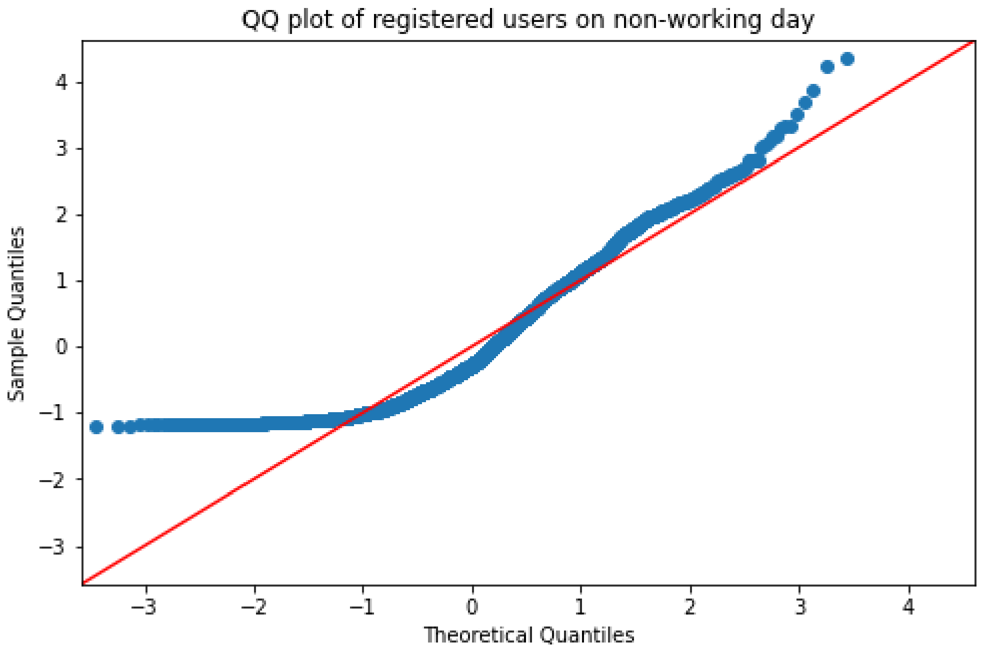
Inference-For Casual and Registered users the count of rented bikes follows different distribution.Hence need to do the Hypothesis test for these groups separately

In [62]:
```
#Split the registered users into two based on workingday
reg_workday_yes=np.array(df.loc[df["workingday"]=="Yes"]["registered"])
reg_workday_no=np.array(df.loc[df["workingday"]=="No"]["registered"])
```

In [38]:
```
fig=sm.qqplot(reg_workday_yes,line='45',fit=True,color='#00DCF7')
plt.title("QQ plot of registered users on working day")
plt.show()
```



In [39]:
```
fig=sm.qqplot(reg_workday_no,line='45',fit=True)
plt.title("QQ plot of registered users on non-working day")
plt.show()
```



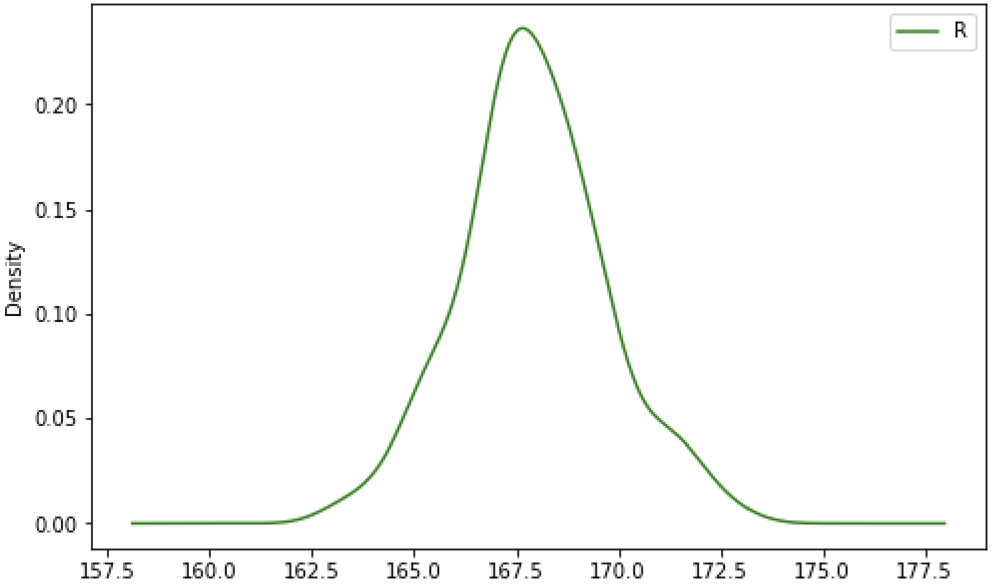
Both working and non working days the sample of registered users follows a non gaussian distribution

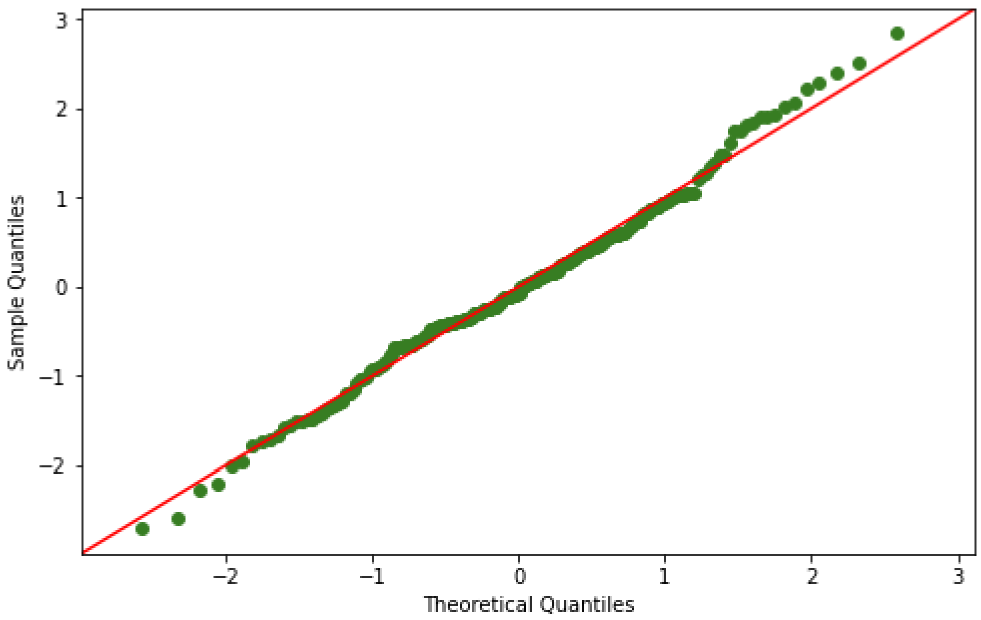
Below code to check CLT

In [176]:
```python
#Sampling the Registered users on working dayto a Normal Distribution
sample_mean_list=[]
number_of_times=200
for i in range (number_of_times):
    sample_data=np.random.choice(reg_workday_yes,size=len(reg_workday_yes),replace=True)
    sample_mean=np.mean(sample_data)
    sample_mean_list.append(sample_mean)
s_mean=round(np.mean(sample_mean_list),2)
s_std=round(np.std(sample_mean_list),2)
print("The mean of Distribution of sample means for Registered users on working day is ",s_mean,"with Standard Error",s_
print("Checking for Normality inorder to do T-test")
pd.DataFrame(sample_mean_list).plot(kind="density")
plt.legend('Reg users on working day')
fig=sm.qqplot(np.array(sample_mean_list),line='45',fit=True)

plt.show()
```

```
The mean of Distribution of sample means for Registered users on working day is  167.92 with Standard Error 1.79
Checking for Normality inorder to do T-test
```

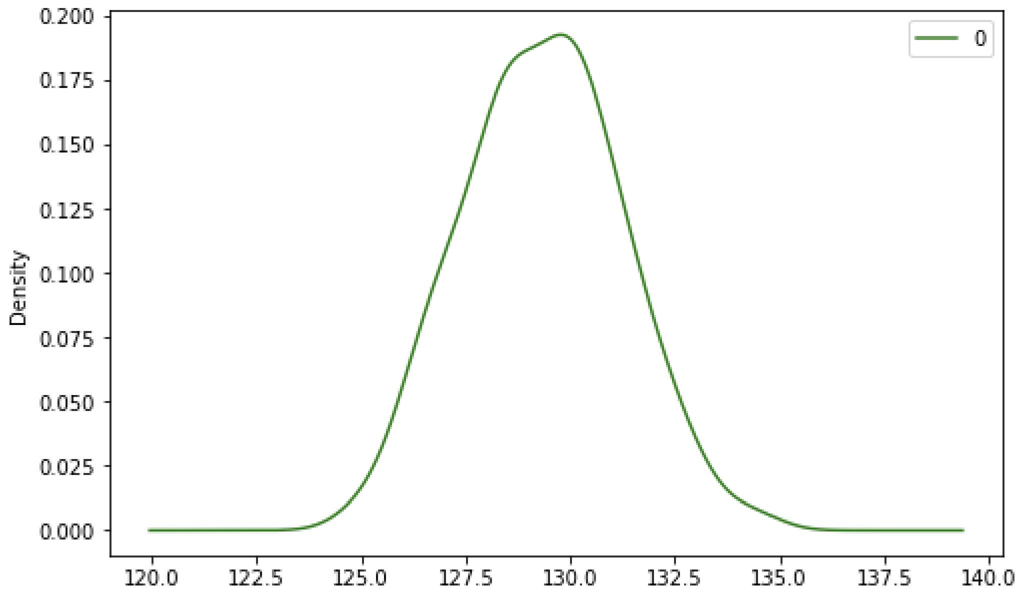


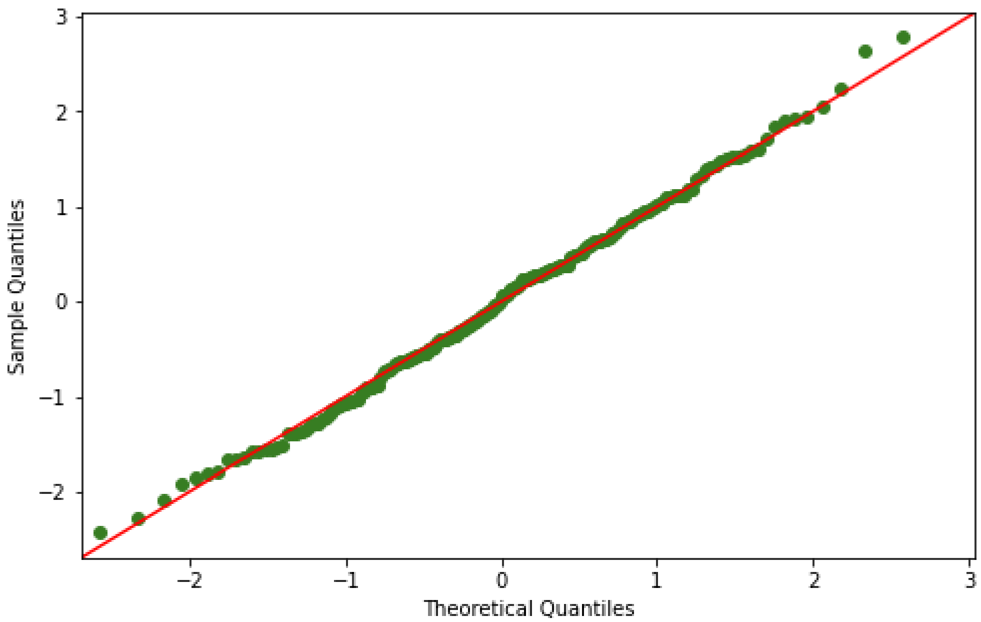

Sample Quantiles
Theoretical Quantiles
3
2
1
0
−1
−2
−2
−1
0
1
2
3

In [173]:
```
#Sampling the Registered users on NON working day to a Normal Distribution
sample_mean_list=[]
number_of_times=200
for i in range (number_of_times):
    sample_data=np.random.choice(reg_workday_no,size=len(reg_workday_no),replace=True)
    sample_mean=np.mean(sample_data)
    sample_mean_list.append(sample_mean)
s_mean=round(np.mean(sample_mean_list),2)
s_std=round(np.std(sample_mean_list),2)
print("The mean of Distribution of sample means for Registered users on NON-Working day is ",s_mean,"with Standard Error
print("Checking for Normality inorder to do T-test")
pd.DataFrame(sample_mean_list).plot(kind="density")
fig=sm.qqplot(np.array(sample_mean_list),line='45',fit=True)

plt.show()
```

```
The mean of Distribution of sample means for Registered users on NON-Working day is  129.34 with Standard Error 1.86
Checking for Normality inorder to do T-test
```





In [175]:
```
print("Number of sample for registered userd on working day", len(reg_workday_yes))
print("Number of sample for registered userd on NON working day", len(reg_workday_no))
print("Std deviation for registered userd on working day", np.std(reg_workday_yes))
print("Std deviation for registered userd on NON working day", np.std(reg_workday_no))
```

```
Number of sample for registered userd on working day 7412
Number of sample for registered userd on NON working day 3474
Std deviation for registered userd on working day 165.80677998119273
Std deviation for registered userd on NON working day 108.64170055329788
```

The CLT holds true for Registered users and the variances of the sample is also known hence do a 2 sample T test to check working day has an effect on the bikes rented by registered users.

Null Hypothesis Ho-Population mean of bikes rented by registered users are same on working and non working day

Alternate Hypothesis Ha--Population mean of bikes rented by registered users are not same on working and non working day

Do a 2 sided 2 sample T-test for the same.

Significance level alpha=5%

In [41]:
```
stats.ttest_ind(reg_workday_yes,reg_workday_no)
```

Out[41]:
```
Ttest_indResult(statistic=12.552707000266874, pvalue=6.806493719916074e-36)
```

Observation:Here Tobs=12.5 and p_val<<alpha.Hence reject Null Hypothesis.Thus it can be concluded that for Registered users working day do matter on Number of bikes rented

2sample T test to check whether the bikes rented on working day is more than non working day for registered users

Null Hypothesis Ho-Population mean of bikes rented by registered users are same on working and non working day

Alternate Hypothesis Ha--Population mean of bikes rented by registered users on working day is more than non working day

Do a Right tail 2 sample T-test for the same.

Significance level alpha=5%

In [42]:
```
import math
m1=np.mean(reg_workday_yes)
m2=np.mean(reg_workday_no)
s1=np.std(reg_workday_yes)
s2=np.std(reg_workday_no)
n1=len(reg_workday_yes)
n2=len(reg_workday_no)
df=n1+n2-2
den=math.sqrt(((s1**2)/n1)+((s2**2)/n2))
tobs=(m1-m2)/den

p_val=1-(stats.t.cdf(tobs,df))
print("Test statistic=",tobs,"p-val is ",p_val)
```

```
Test statistic= 14.519274677646957 p-val is  0.0
```

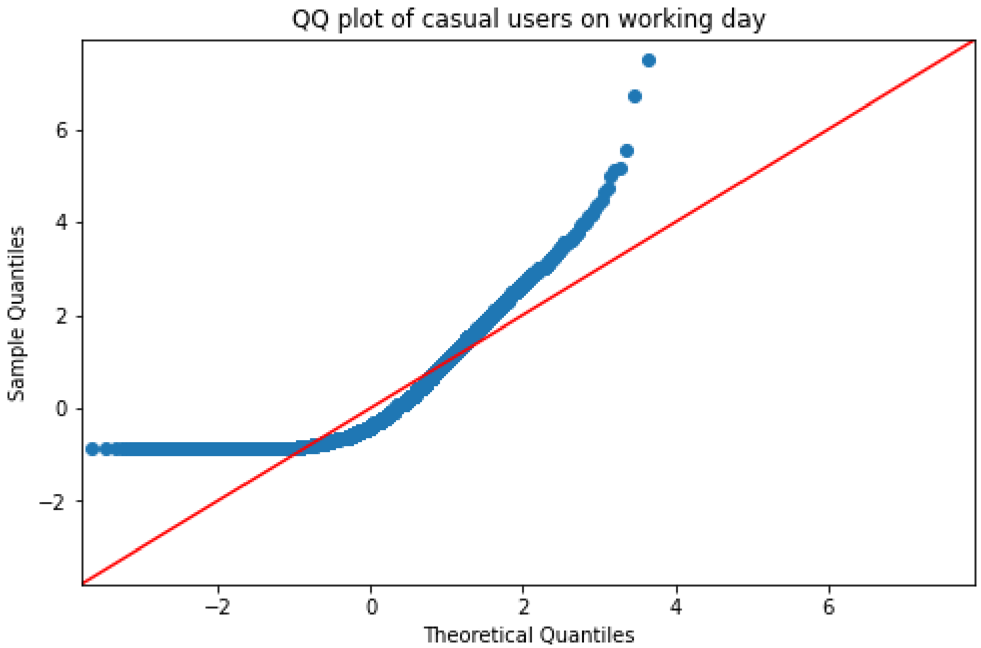
p_val<<alpha.Hence reject Null Hypothesis.Thus it can be concluded that for Registered users bike rented on working day is more than non working day

Similarly do T test on casual users

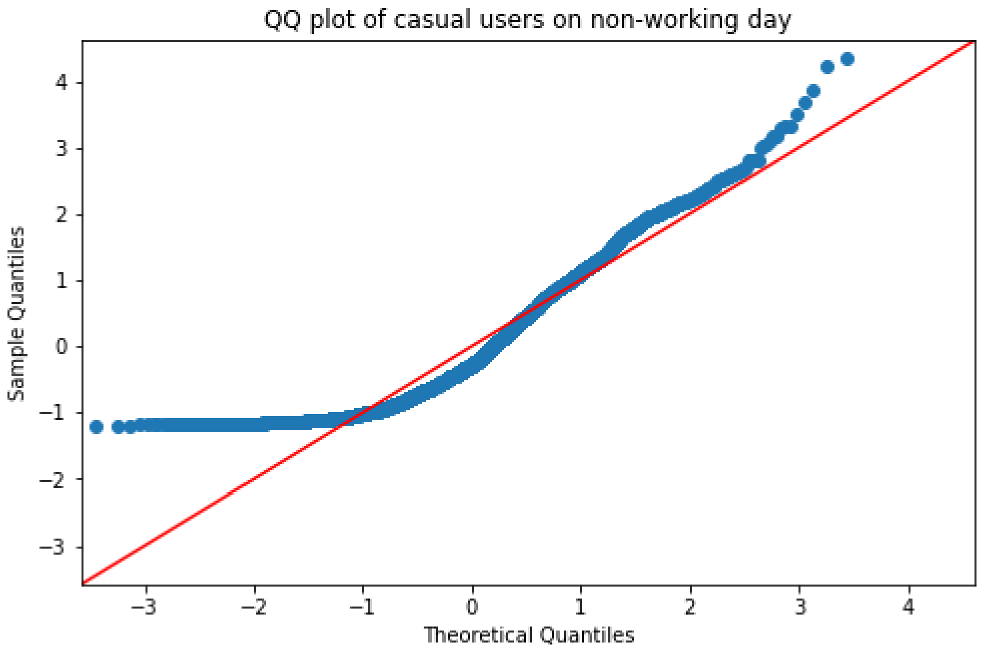
Check for CLT and Variances

In [67]:
```
cas_workday_yes=np.array(df.loc[df["workingday"]=="Yes"]["casual"])
cas_workday_no=np.array(df.loc[df["workingday"]=="No"]["casual"])
```

In [70]:
```
fig=sm.qqplot(cas_workday_yes,line='45',fit=True)
plt.title("QQ plot of casual users on working day")
plt.show()
```



In [71]:
```
fig=sm.qqplot(reg_workday_no,line='45',fit=True)
plt.title("QQ plot of casual users on non-working day")
plt.show()
```



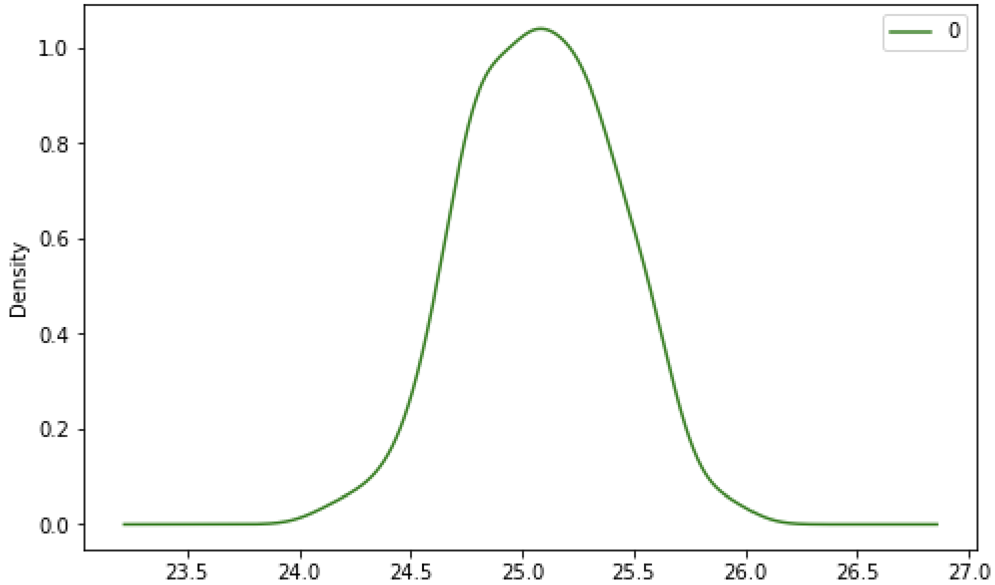
Observation-Casual users distribution also follows non gaussian irrespective of whether its a working day or not

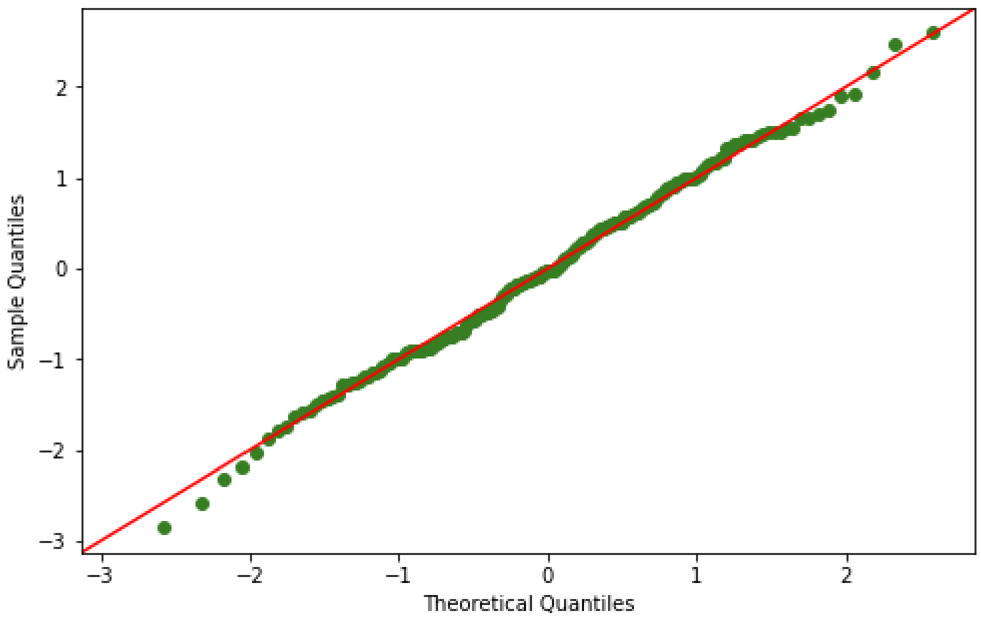
In [177]:

```
#Sampling the Caual users on working dayto a Normal Distribution
sample_mean_list=[]
number_of_times=200
for i in range (number_of_times):
    sample_data=np.random.choice(cas_workday_yes,size=len(cas_workday_yes),replace=True)
    sample_mean=np.mean(sample_data)
    sample_mean_list.append(sample_mean)
s_mean=round(np.mean(sample_mean_list),2)
s_std=round(np.std(sample_mean_list),2)
print("The mean of Distribution of sample means for Casual users on working day is ",s_mean,"with Standard Error",s_std)
print("Checking for Normality inorder to do T-test")
pd.DataFrame(sample_mean_list).plot(kind="density")
fig=sm.qqplot(np.array(sample_mean_list),line='45',fit=True)

plt.show()
```

```
The mean of Distribution of sample means for Casual users on working day is  25.08 with Standard Error 0.33
Checking for Normality inorder to do T-test
```





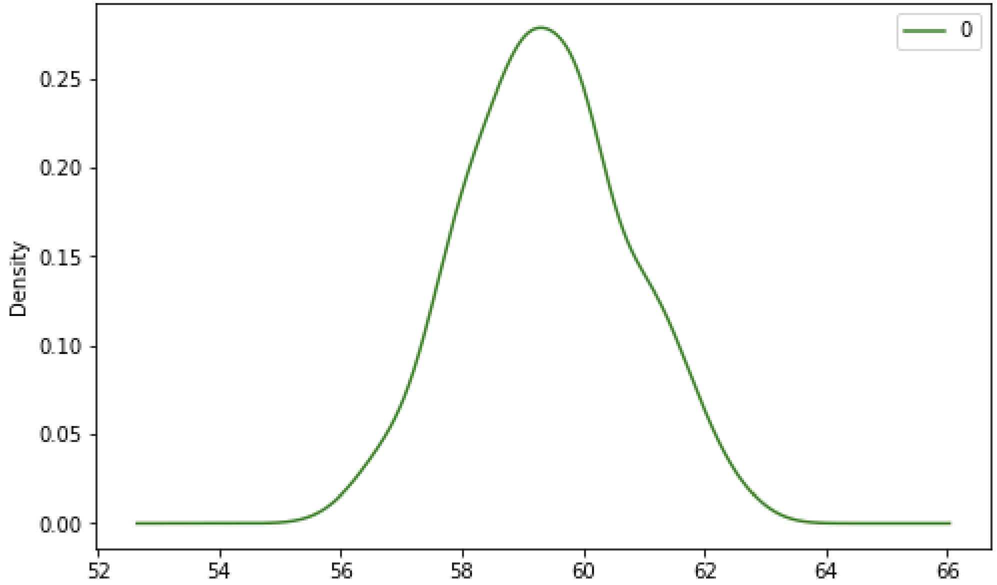
Sample Quantiles
Theoretical Quantiles
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2

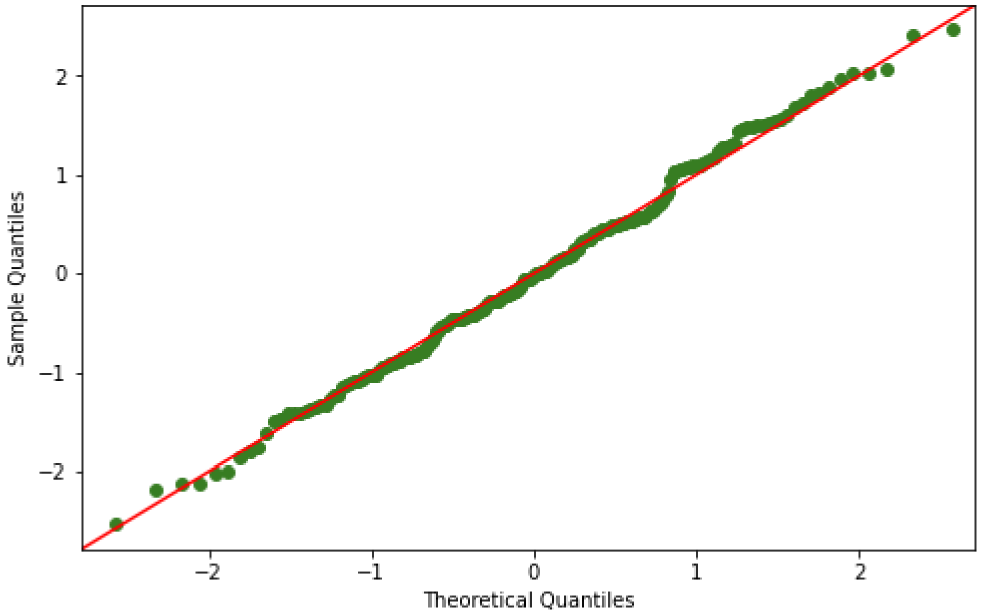
In [178]:

```
#Sampling the Caual users on NON working dayto a Normal Distribution
sample_mean_list=[]
number_of_times=200
for i in range (number_of_times):
    sample_data=np.random.choice(cas_workday_no,size=len(cas_workday_no),replace=True)
    sample_mean=np.mean(sample_data)
    sample_mean_list.append(sample_mean)
s_mean=round(np.mean(sample_mean_list),2)
s_std=round(np.std(sample_mean_list),2)
print("The mean of Distribution of sample means for Casual users on NON working day is ",s_mean,"with Standard Error",s_
print("Checking for Normality inorder to do T-test")
pd.DataFrame(sample_mean_list).plot(kind="density")
fig=sm.qqplot(np.array(sample_mean_list),line='45',fit=True)

plt.show()
```

```
The mean of Distribution of sample means for Casual users on NON working day is  59.39 with Standard Error 1.34
Checking for Normality inorder to do T-test
```




Sample Quantiles
Theoretical Quantiles
2
1
0
−1
−2
−2
−1
0
1
2

Assumption for T test are met

Null Hypothesis Ho-Population mean of bikes rented by casual users are same on working and non working day

Alternate Hypothesis Ha--Population mean of bikes rented by casual users are not same on working and non working day

Do a 2 sided 2 sample T-test for the same.

Significance level alpha=5%

In [72]:
```
stats.ttest_ind(cas_workday_yes,cas_workday_no)
```

Out[72]:
```
Ttest_indResult(statistic=-35.12830185964087, pvalue=3.5619674236054405e-256)
```

Observation:Here Tobs=35.12 and p_val<<alpha.Hence reject Null Hypothesis.Thus it can be concluded that for casual users working day do matter on Number of bikes rented

2sample T test to check whether the bikes rented on working day is more than non working day for casual users

Null Hypothesis Ho-Population mean of bikes rented by casual users are same on working and non working day

Alternate Hypothesis Ha--Population mean of bikes rented by casual users on working day is more than non working day

Do a Right tail 2 sample T-test for the same.

Significance level alpha=5%

In [79]:
```
df=len(cas_workday_yes)+len(cas_workday_no)-2

pval=(1-stats.t.cdf(35.12,df))
pval
```

Out[79]:
```
0.0
```

Since Pval is less than 5% Reject the null hypothesis.Thus Population mean of bikes rented by casual users on working day is more than non working day

To check the dependency of Weather on No:of cycles Rented

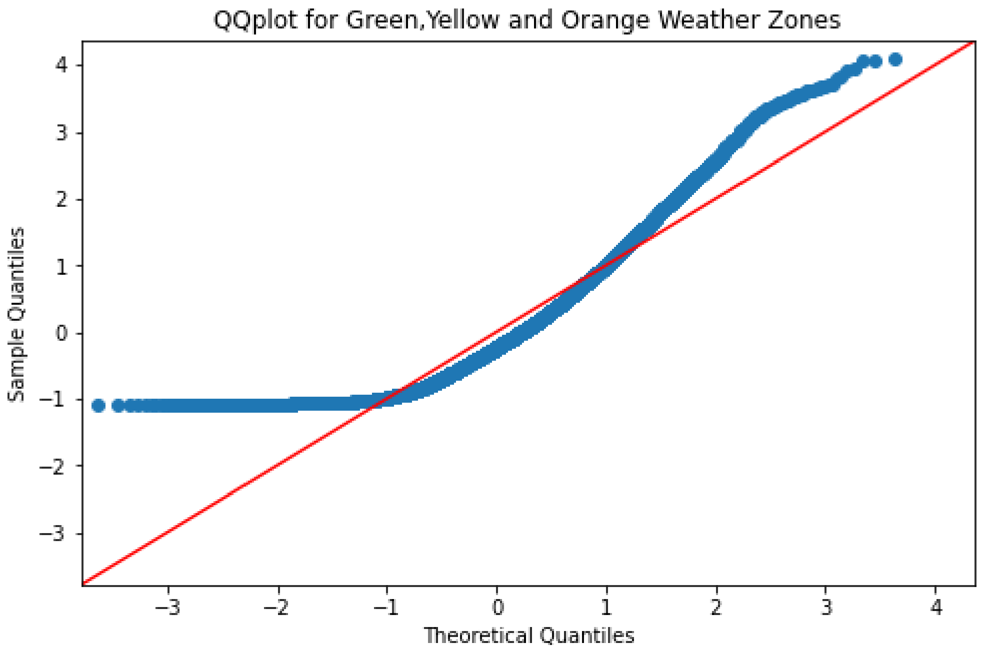
In [85]:
```
cnt_green=df.loc[df["Zone"]=="Green"]["count"]
cnt_yellow=df.loc[df["Zone"]=="Yellow"]["count"]
cnt_orange=df.loc[df["Zone"]=="Orange"]["count"]
cnt_red=df.loc[df["Zone"]=="Red"]["count"]
```

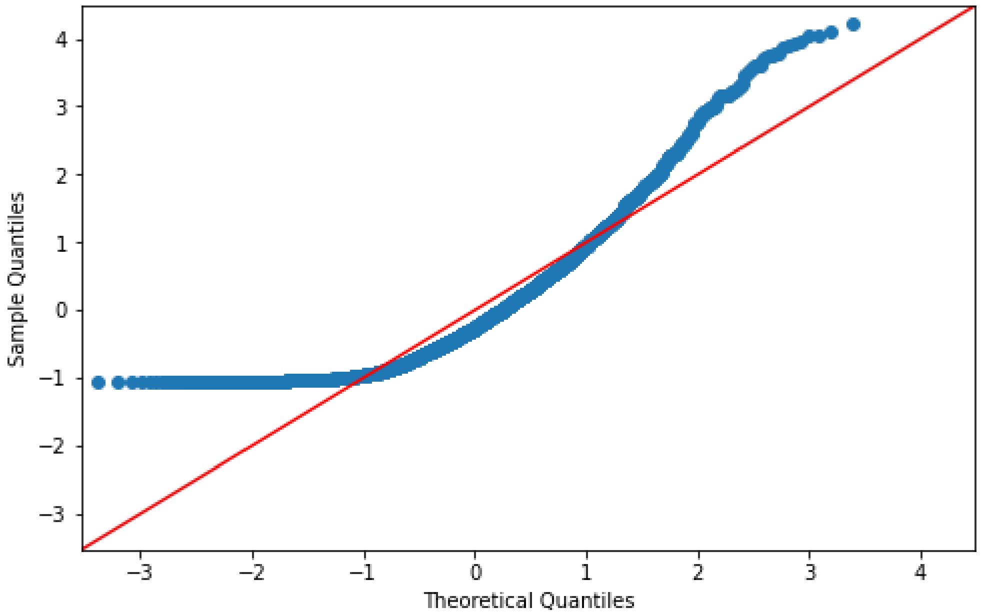
Checking Assumptions for Annova

In [86]:

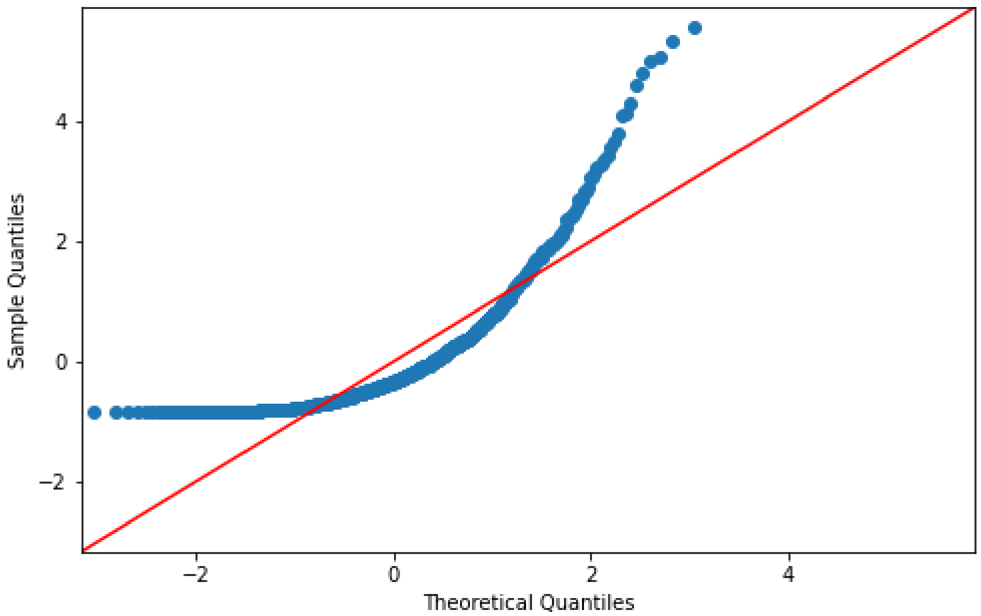
```
fig=sm.qqplot(cnt_green,line='45',fit=True)
plt.title("QQplot for Green,Yellow and Orange Weather Zones")
fig=sm.qqplot(cnt_yellow,line='45',fit=True)
fig=sm.qqplot(cnt_orange,line='45',fit=True)

plt.show()
```





Sample Quantiles
Theoretical Quantiles



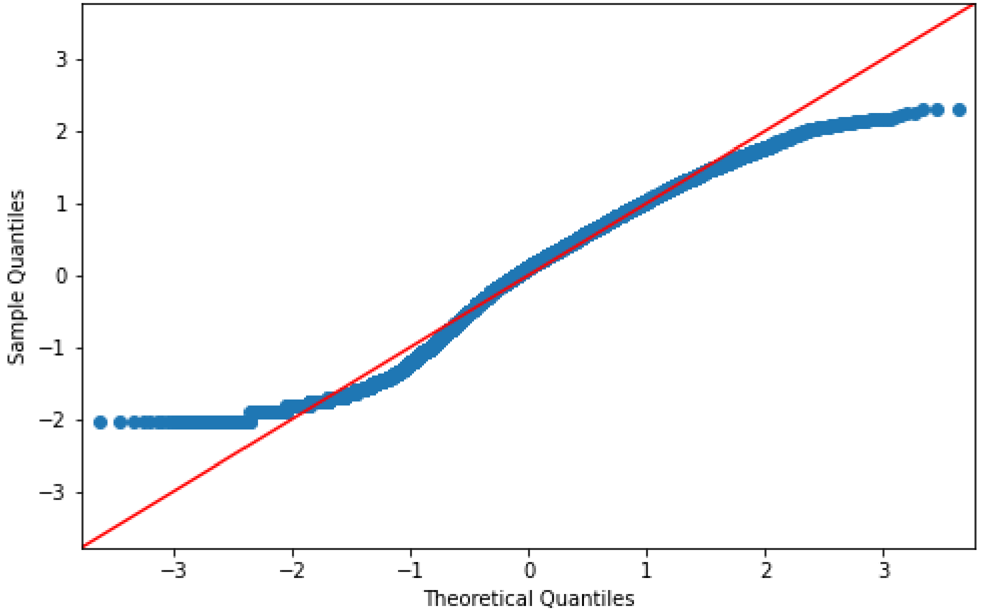
Sample Quantiles
Theoretical Quantiles

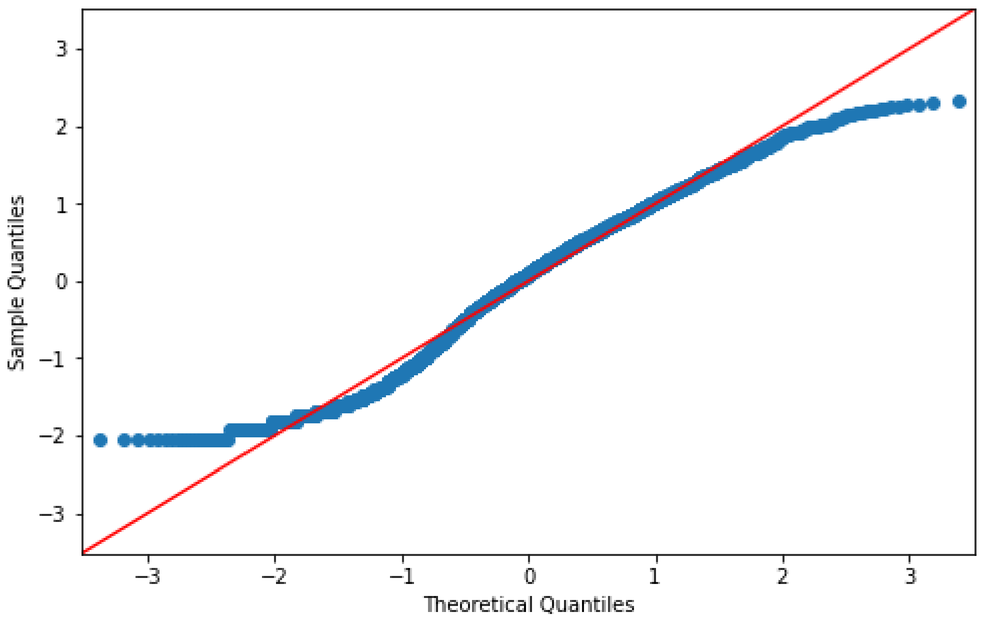
For the three weather zones ,its a non gaussian distribution,Hence need to do a Box cox transformation on the same.For the Red zone since it only have one observation,its not fair to draw conclusions based on it

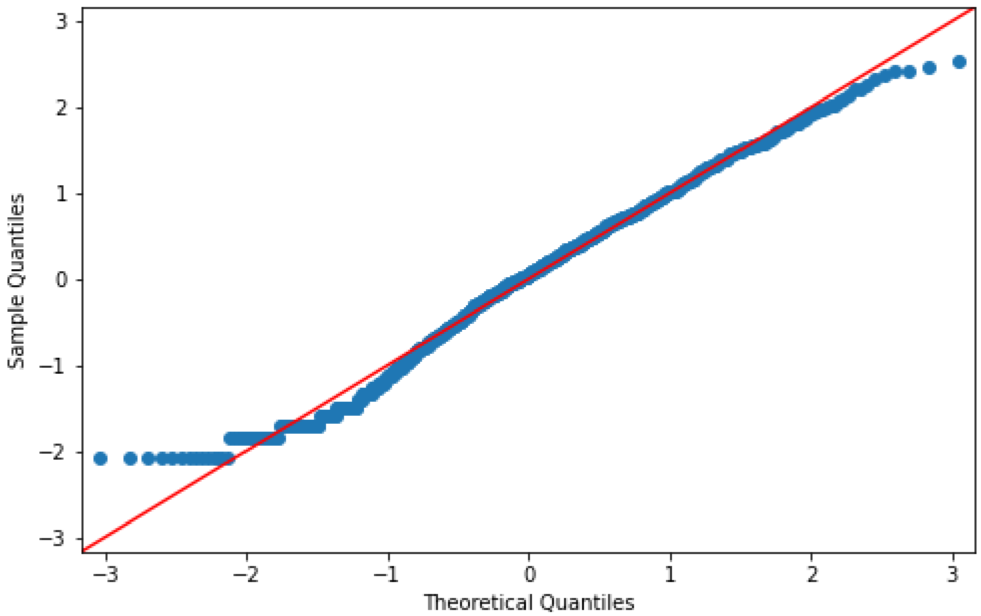
In [87]:
```
print("The Standard deviation for Green weather zone is",round(np.std(cnt_green),2))
print("The Standard deviation for Yellow weather zone is",round(np.std(cnt_yellow),2))
print("The Standard deviation for Orange weather zone is",round(np.std(cnt_orange),2))
```

```
The Standard deviation for Green weather zone is 187.95
The Standard deviation for Yellow weather zone is 168.34
The Standard deviation for Orange weather zone is 138.5
```

In [88]:
```
for x in (cnt_green,cnt_yellow,cnt_orange):
    x_list=np.array(x.to_list())
    x_trans,l=scipy.stats.boxcox(x_list)
    sm.qqplot(x_trans,line='45',fit=True)
```







Eventhough the variance for each of the three weather zones are roughly same ,the

distribution is clearly not gaussian even after a Box cox Transformation.Hence Opting for Kruskal Wallis Test to check the dependency of weather on bikes rented

Null Hypothesis Ho-The population median of groups (Green,Yellow,Orange) are equal

Alternate Hypothesis Ha-The population median of groups (Green,Yellow,Orange) are different

Kruskal Wallis test is done with significance level as 5%

In [89]:
```
scipy.stats.kruskal(cnt_green,cnt_yellow,cnt_orange)
```

Out[89]:
```
KruskalResult(statistic=204.95566833068537, pvalue=3.122066178659941e-45)
```

Observed Test Statistic is 205 with p_value very less compared to alpha.Hence concluding the weather do impact the bikes rented.Let's see how?

To Analyse data wrt to weather and holiday/workingday

In [90]:

```
#Create a new dataframe which conatins the total bikes rented by registered users wrt Weather/holiday/workingday
df_registered=df.groupby(["Zone","holiday","workingday"])["registered"].sum().to_frame().reset_index()
df_registered
```

Out[90]:

	Zone	holiday	workingday	registered
0	Green	No	No	296263
1	Green	No	Yes	861906
2	Green	Yes	No	27994
3	Orange	No	No	17927
4	Orange	No	Yes	67676
5	Orange	Yes	No	1503
6	Red	No	Yes	158
7	Yellow	No	No	92008
8	Yellow	No	Yes	314766
9	Yellow	Yes	No	13140

In [91]:
```
#Create a new dataframe which conatins the total bikes rented by casual users wrt Weather/holiday/workingday
df_casual=df.groupby(["Zone","holiday","workingday"])["casual"].sum().to_frame().reset_index()
df_casual
```

Out[91]:

	Zone	holiday	workingday	casual
0	Green	No	No	144793
1	Green	No	Yes	135684
2	Green	Yes	No	9423
3	Orange	No	No	7265
4	Orange	No	Yes	7391
5	Orange	Yes	No	327
6	Red	No	Yes	6
7	Yellow	No	No	38808
8	Yellow	No	Yes	43017
9	Yellow	Yes	No	5421

In [92]:
```
#Merging above two datasets
df1=df_registered.merge(df_casual,on=["Zone","holiday","workingday"])
df1
```

Out[92]:

	Zone	holiday	workingday	registered	casual
0	Green	No	No	296263	144793
1	Green	No	Yes	861906	135684
2	Green	Yes	No	27994	9423
3	Orange	No	No	17927	7265
4	Orange	No	Yes	67676	7391
5	Orange	Yes	No	1503	327
6	Red	No	Yes	158	6
7	Yellow	No	No	92008	38808
8	Yellow	No	Yes	314766	43017
9	Yellow	Yes	No	13140	5421

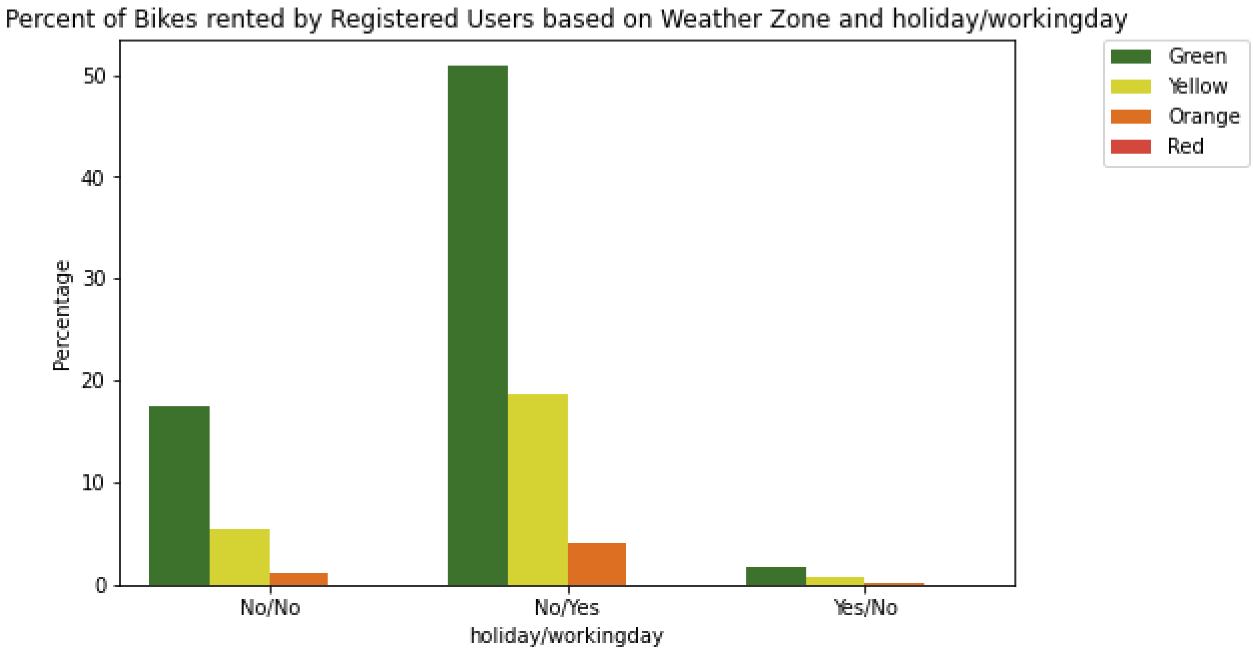
In [93]:
```
#Compute the conditional Probability for weather holiday/workingday criteria
df1["p_registered"]=df1["registered"]*100/(df1.loc[:,"registered"].sum())
df1["p_casual"]=df1["casual"]*100/((df1.loc[:,"casual"].sum()))
df1["holiday/workingday"]=df1["holiday"]+"/"+df1["workingday"]
df1
```

Out[93]:

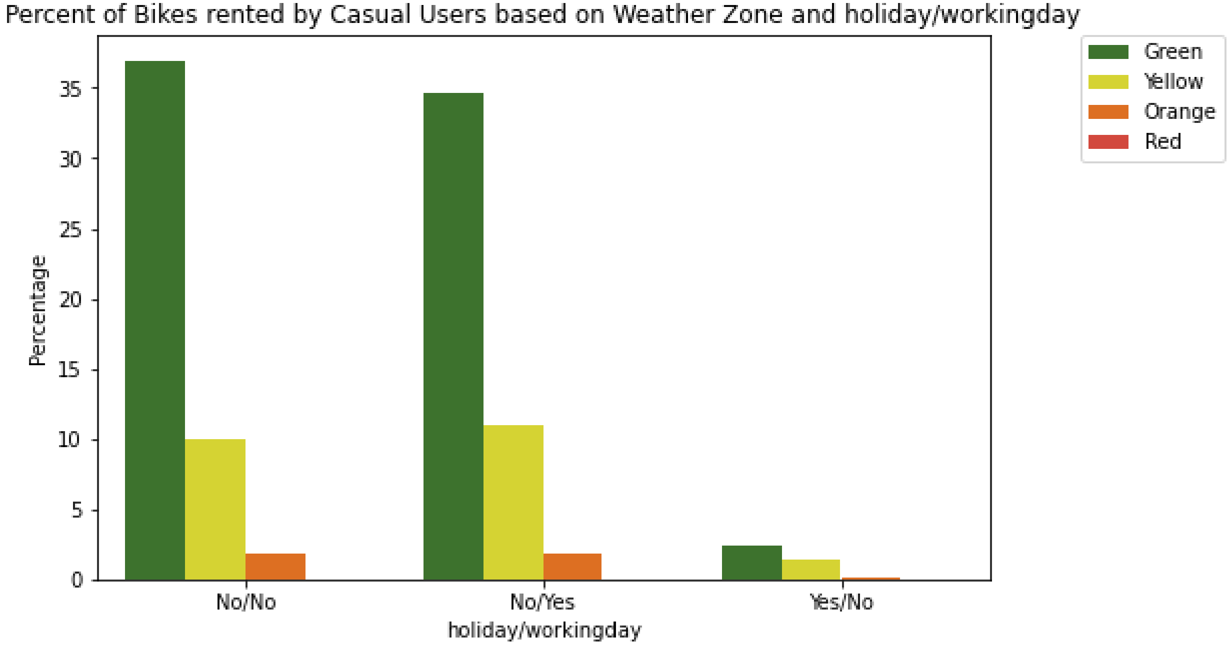
	Zone	holiday	workingday	registered	casual	p_registered	p_casual	holiday/workingday
0	Green	No	No	296263	144793	17.495767	36.924274	No/No
1	Green	No	Yes	861906	135684	50.899730	34.601349	No/Yes
2	Green	Yes	No	27994	9423	1.653181	2.402999	Yes/No
3	Orange	No	No	17927	7265	1.058676	1.852678	No/No
4	Orange	No	Yes	67676	7391	3.996596	1.884810	No/Yes
5	Orange	Yes	No	1503	327	0.088759	0.083390	Yes/No
6	Red	No	Yes	158	6	0.009331	0.001530	No/Yes
7	Yellow	No	No	92008	38808	5.433519	9.896592	No/No
8	Yellow	No	Yes	314766	43017	18.588459	10.969947	No/Yes
9	Yellow	Yes	No	13140	5421	0.775981	1.382432	Yes/No

In [94]:

```
colors=["#377D22","#F0ED18","#FC6A03","#EB3324"]
sns.set_palette(sns.color_palette(colors))
sns.barplot(x="holiday/workingday",y="p_registered",data=df1,hue_order=["Green","Yellow","Orange","Red"],hue="Zone")
plt.legend(bbox_to_anchor=(1.1 ,1), loc='upper left', borderaxespad=0)
plt.title("Percent of Bikes rented by Registered Users based on Weather Zone and holiday/workingday")
plt.ylabel("Percentage")
plt.show()
```



In [95]:
```
sns.barplot(x="holiday/workingday",y="p_casual",data=df1,hue="Zone",hue_order=["Green","Yellow","Orange","Red"])
plt.legend(bbox_to_anchor=(1.1 ,1), loc='upper left', borderaxespad=0)
plt.title("Percent of Bikes rented by Casual Users based on Weather Zone and holiday/workingday")
plt.ylabel("Percentage")
plt.show()
```



Inference-For both registered and casual users more bikes are rented when the whether is in Green zone ie Mostly clear followed by yellow ,orange and least in Red zones which is quite obvious.Working days has most bikes rented for registered users and for casual users this trend happen during weekend.Interesting to observe that for both registered and casual users the bikes rented are very less on holidays.

Through Hypothesis testing it was found that weather do impact the number of bikes rented.Let's check which weather zone has a higher population mean through Bootstrapping and 95% Confidence Intervals

In [114]:
```
"""introducing two custom functions to split the dataframe and to Bootstrap
1)Function to split the dataframe df based
on the column name and column value"""
def DataFrameSplit (df,column,value):
    name_dataframe="df_"+column+"_"+value
    x=df.loc[df[column]==value]
    x.reset_index(inplace=True)
    x.drop("index",axis=1,inplace=True)
    return name_dataframe,x
```

In [115]:
```
#Dataframe df is split based on column Zone and value=Green.The resultant dataframe is stored in dictionary dataframes
name,data=DataFrameSplit(df,"Zone","Green")
dataframes={}
dataframes[name] = data
```

In [116]:
```
#initialsing dictionaries to store the confidence intervals and bootstrap means for the dataframes post the split
dataframes_namelist_pos=0
ci_dict={}
bootstrap_mean_dict={}
```

In [117]:

```
"""2)This function is used to find the bootstrapped means of the dataframes post the spilt and then find their confidence
intervals
The bootstrapped means of each of the split dataframes are stored in dictionary bootstrap_mean_dict and the 95% CI's is
def BootStrapFunc(data):
    bootstap_mean_list=[]
    global dataframes_namelist_pos
    number_of_times=200
    for i in range (number_of_times):
        sample_data=data.sample(n=len(data),replace=True)
        bootstrap_mean=np.mean(sample_data["count"])
        bootstap_mean_list.append(bootstrap_mean)
    c_interval=[]
    global ci_dict
    global bootstrap_mean_dict
    bootstrap_mean=np.mean(bootstap_mean_list)

    ci_name=list(dataframes.keys())[dataframes_namelist_pos]
    ci_name=ci_name+'_CI'

    bs_name=list(dataframes.keys())[dataframes_namelist_pos]
    bs_name=bs_name+"_BS"

    ci=95
    lb=(100-ci)/2
    ub=ci+(100-ci)/2
    c_interval.append(np.percentile(bootstap_mean_list,[lb,ub]))
    print("Mean of the Sampling Distribution is",round(bootstrap_mean,2))
    print("95% Confidence Interval is [",round(c_interval[0][0],2),",",round(c_interval[0][1],2),"]")
    ci_dict[ci_name]=c_interval
    bootstrap_mean_dict[bs_name]=bootstap_mean_list

    dataframes_namelist_pos+=1
```

In [118]:

```
#BootStrapFunc function call on splitted dataframe to find CI
BootStrapFunc(dataframes['df_Zone_Green'])
```

```
Mean of the Sampling Distribution is 205.31
95% Confidence Interval is [ 201.36 , 210.03 ]
```

In [119]:
```
#Repeat the same function calls for weather Zone=Yellow
name,data=DataFrameSplit(df,"Zone","Yellow")
dataframes[name]=data
```

In [120]:
```
#BootStrapFunc function call on splitted dataframe to find CI
BootStrapFunc(dataframes['df_Zone_Yellow'])
```

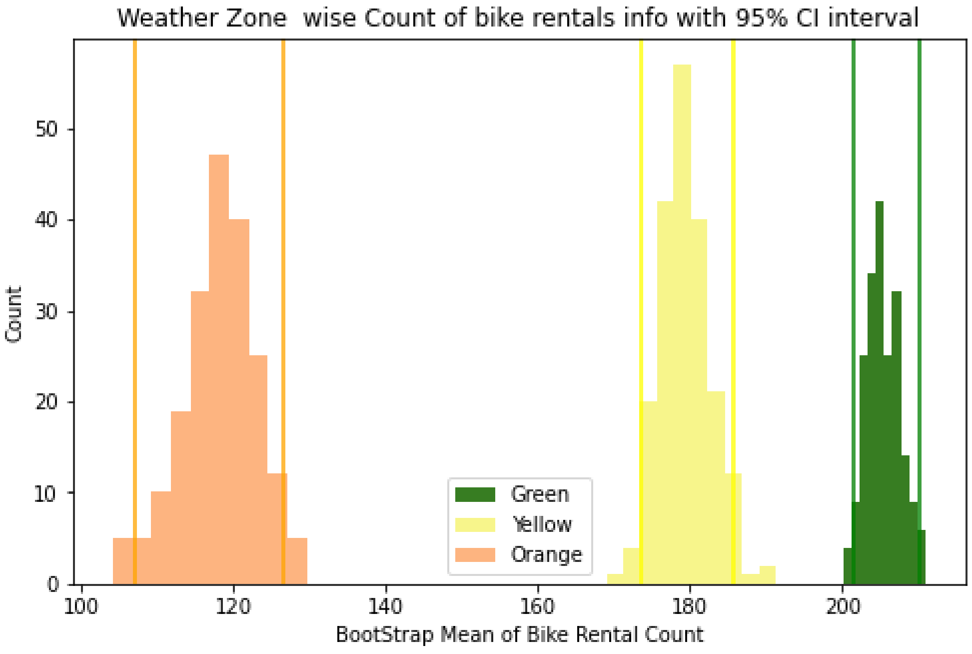
```
Mean of the Sampling Distribution is 179.41
95% Confidence Interval is [ 173.61 , 185.63 ]
```

In [121]:
```
#Repeat the same function calls for weather Zone=Orange
name,data=DataFrameSplit(df,"Zone","Orange")
dataframes[name]=data
```

In [122]:
```
#BootStrapFunc function call on splitted dataframe to find CI
BootStrapFunc(dataframes['df_Zone_Orange'])
```

```
Mean of the Sampling Distribution is 118.05
95% Confidence Interval is [ 106.95 , 126.48 ]
```

In [123]:
```
#Plot for 95% CI
plt.hist(bootstrap_mean_dict["df_Zone_Green_BS"],label="Green")
plt.hist(bootstrap_mean_dict["df_Zone_Yellow_BS"],label="Yellow",alpha=.5)
plt.hist(bootstrap_mean_dict["df_Zone_Orange_BS"],label="Orange",alpha=.5)
plt.axvline(ci_dict['df_Zone_Green_CI'][0][0],c='g')
plt.axvline(ci_dict['df_Zone_Green_CI'][0][1],c='g')
plt.axvline(ci_dict['df_Zone_Yellow_CI'][0][0],color='yellow')
plt.axvline(ci_dict['df_Zone_Yellow_CI'][0][1],color='yellow')
plt.axvline(ci_dict['df_Zone_Orange_CI'][0][0],color='orange')
plt.axvline(ci_dict['df_Zone_Orange_CI'][0][1],color='orange')
plt.title("Weather Zone  wise Count of bike rentals info with 95% CI interval")
plt.xlabel("BootStrap Mean of Bike Rental Count")
plt.ylabel("Count")
plt.legend()
plt.show()
```



Inference-The confidence intervals are distinct and non overlapping which implies the number of bikes rented significantly depends on the weather.Here when the weather is in Green zone the average number of bikes rented is between 201 and 210.In Yellow its between 173 and 185 and in Orange its between 107 and 126

To check the dependency of Season on No:of cycles Rented

In [124]:
```
cnt_spring=df.loc[df["season"]=="Spring"]["count"]
cnt_winter=df.loc[df["season"]=="Winter"]["count"]
cnt_summer=df.loc[df["season"]=="Summer"]["count"]
cnt_fall=df.loc[df["season"]=="Fall"]["count"]
```

Checking Assumptions for Annova

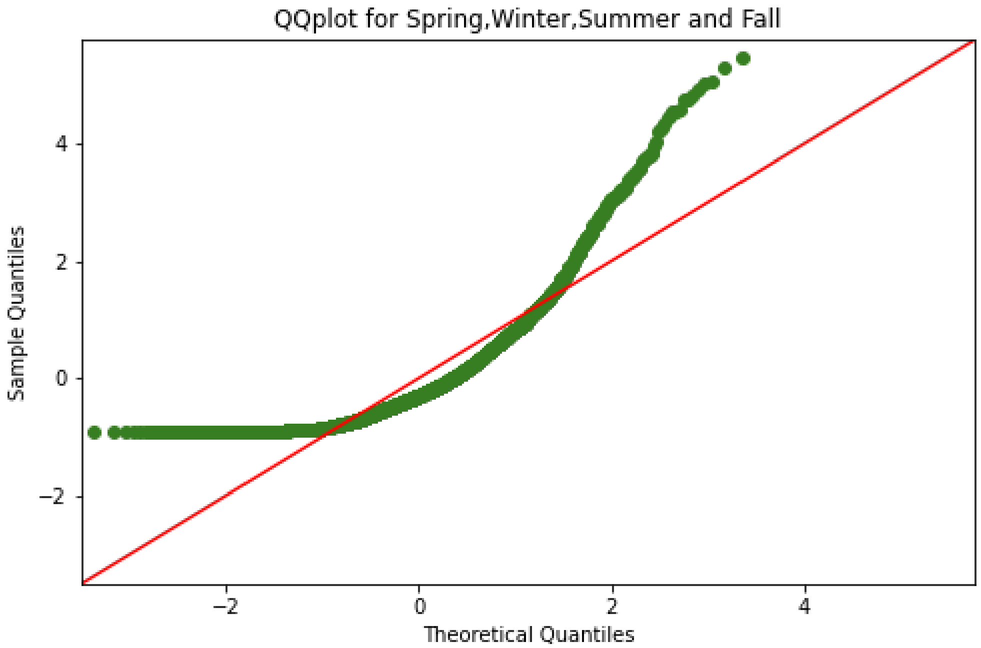
In [125]:

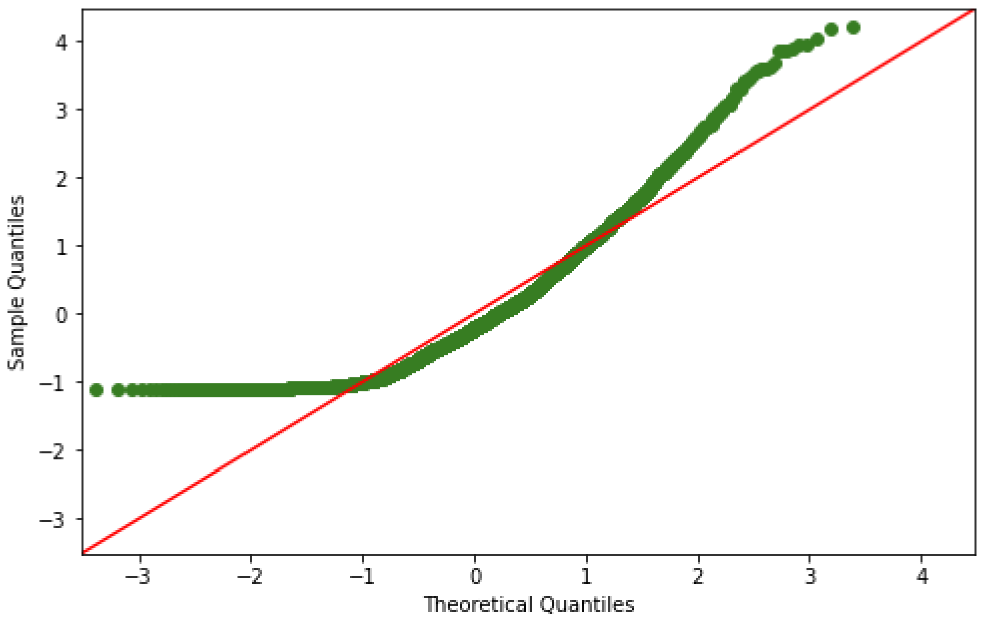
```
print("The Standard deviation for Spring is",round(np.std(cnt_spring),2))
print("The Standard deviation for Winter is",round(np.std(cnt_winter),2))
print("The Standard deviation for Summer is",round(np.std(cnt_summer),2))
print("The Standard deviation for Fall is",round(np.std(cnt_fall),2))
```

```
The Standard deviation for Spring is 125.25
The Standard deviation for Winter is 177.59
The Standard deviation for Summer is 191.97
The Standard deviation for Fall is 197.11
```

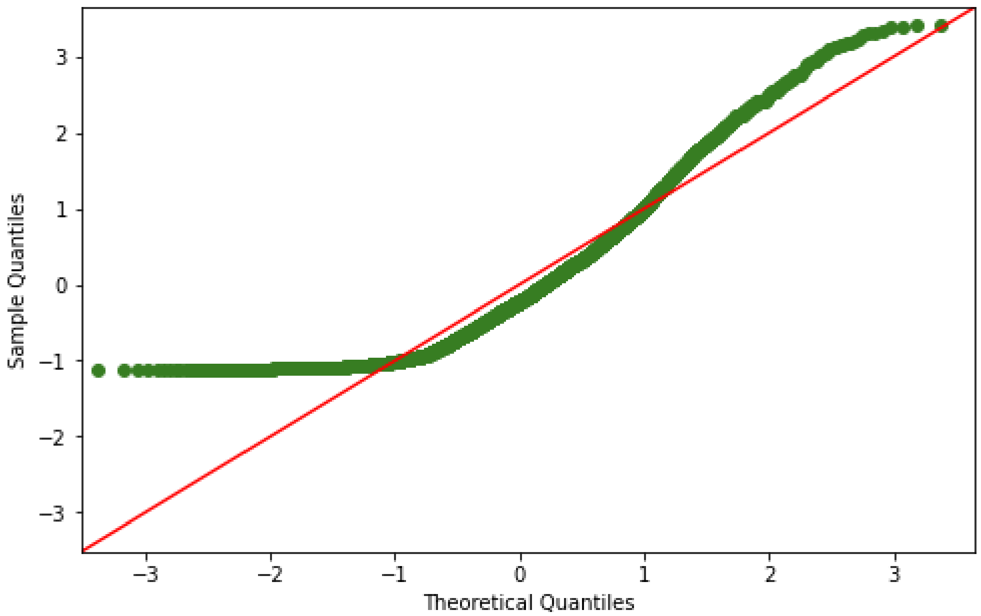
In [126]:
```
fig=sm.qqplot(cnt_spring,line='45',fit=True)
plt.title("QQplot for Spring,Winter,Summer and Fall")
fig=sm.qqplot(cnt_winter,line='45',fit=True)
fig=sm.qqplot(cnt_summer,line='45',fit=True)
fig=sm.qqplot(cnt_fall,line='45',fit=True)

plt.show()
```

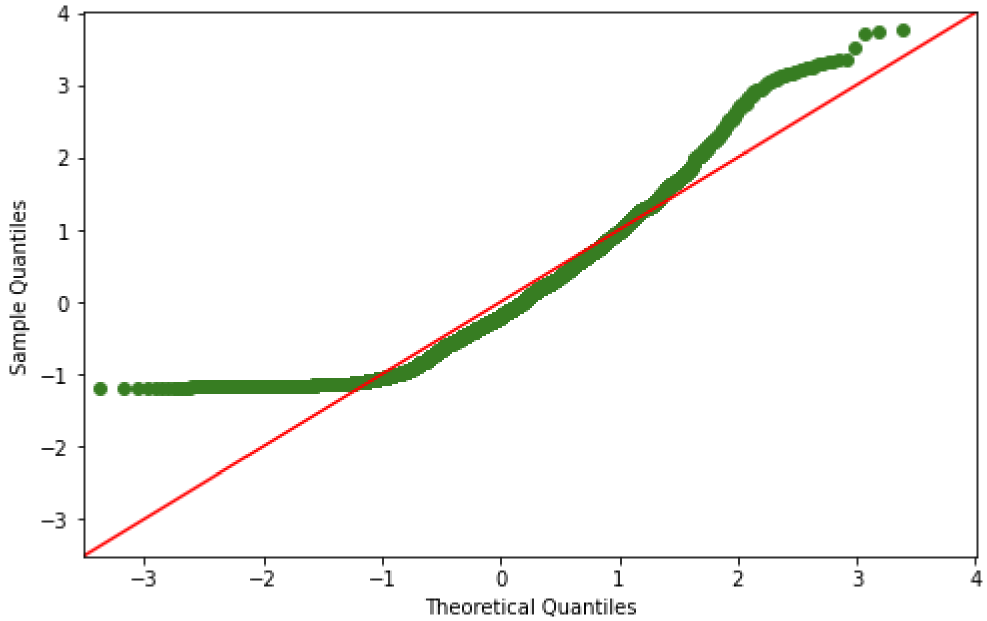




Sample Quantiles
Theoretical Quantiles



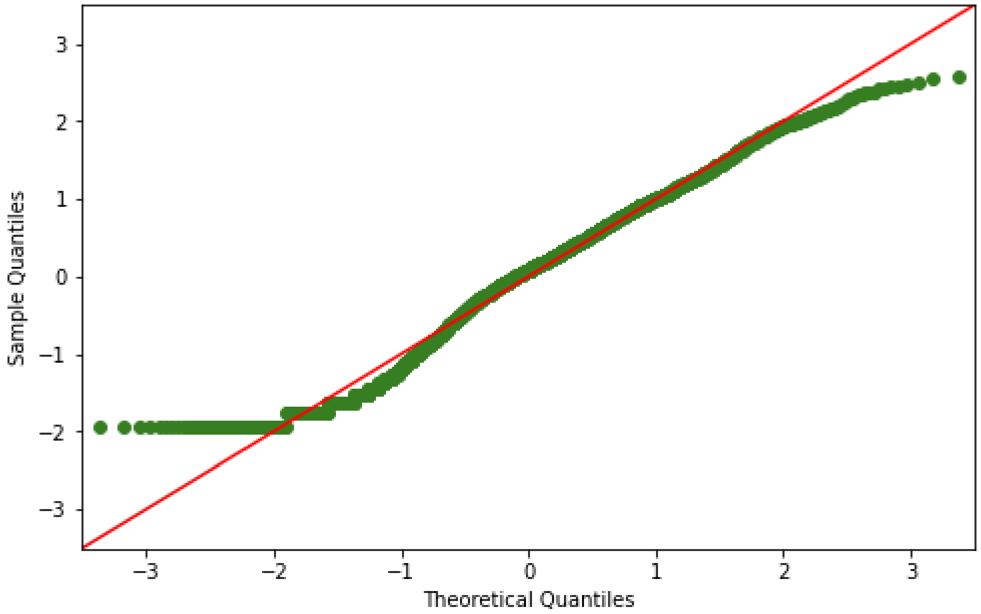
Sample Quantiles
Theoretical Quantiles

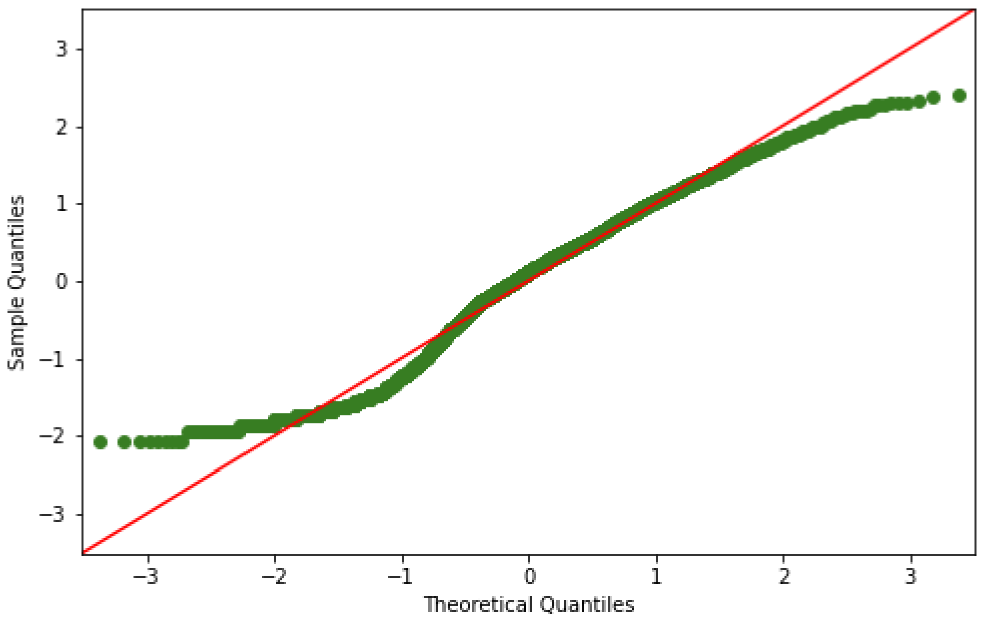


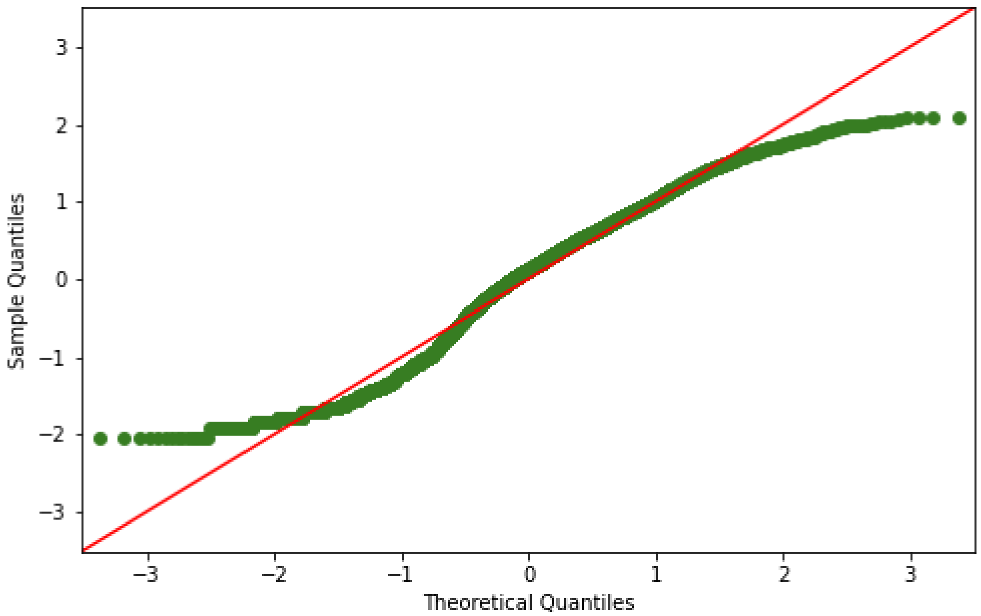
The above plots shows the distribution is not gaussian hence do BOX-COX transformation

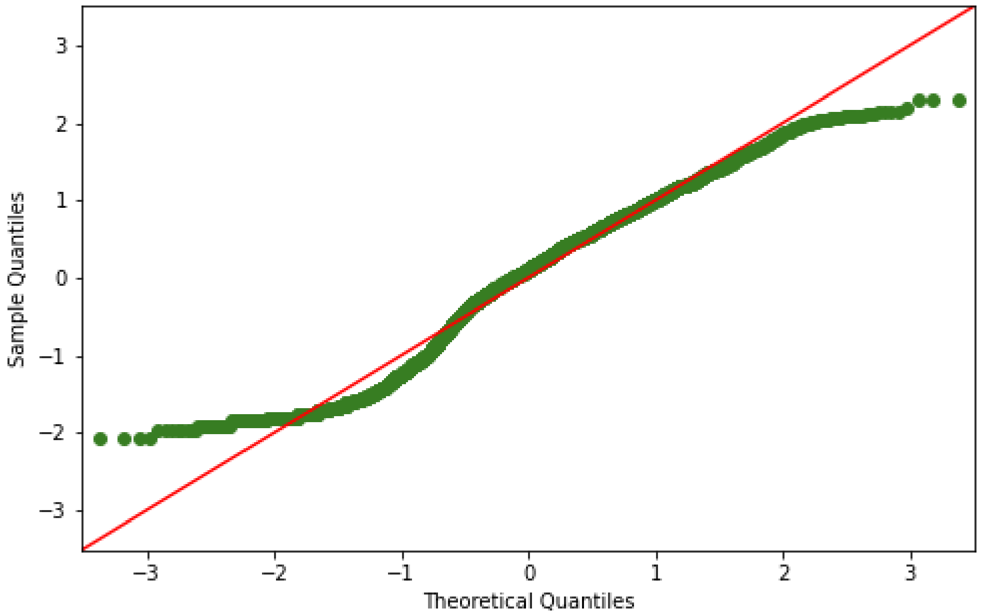
In [127]:
```
#BOX COX Transformation
#plt.title("After Transformation")
for x in (cnt_spring,cnt_winter,cnt_summer,cnt_fall):
    x_list=np.array(x.to_list())
    x_trans,l=scipy.stats.boxcox(x_list)

    sm.qqplot(x_trans,line='45',fit=True)
```




Sample Quantiles
Theoretical Quantiles
3
2
1
0
−1
−2
−3


Sample Quantiles
Theoretical Quantiles
3
2
1
0
−1
−2
−3



Eventhough the variance for each of the four seasons are roughly same ,the distribution is clearly not gaussian even after a Box cox Transformation.Hence Opting for Kruskal Wallis Test to check the dependency of season on bikes rented

Null Hypothesis Ho-The population median of all seasons are equal

Alternate Hypothesis Ha-The population median of the seasons are different

Kruskal Wallis test is done with significance level as 5%

In [128]:
```
scipy.stats.kruskal(cnt_spring,cnt_winter,cnt_summer,cnt_fall)
```

Out[128]:
```
KruskalResult(statistic=699.6668548181988, pvalue=2.479008372608633e-151)
```

Observed Test Statistic is 699.6 with p_value very less compared to alpha.Hence concluding the seasons do impact the bikes rented.Let's see how?

In [134]:
```
season_reg=df.groupby(["season"])["registered"].sum().to_frame().reset_index()
season_cas=df.groupby(["season"])["casual"].sum().to_frame().reset_index()
#Merging above two datasets
df2=season_reg.merge(season_cas,on=["season"])
```

In [135]:
```
df2#after Merging
```

Out[135]:

	season	registered	casual
0	Fall	497944	142718
1	Spring	270893	41605
2	Summer	458610	129672
3	Winter	465894	78140

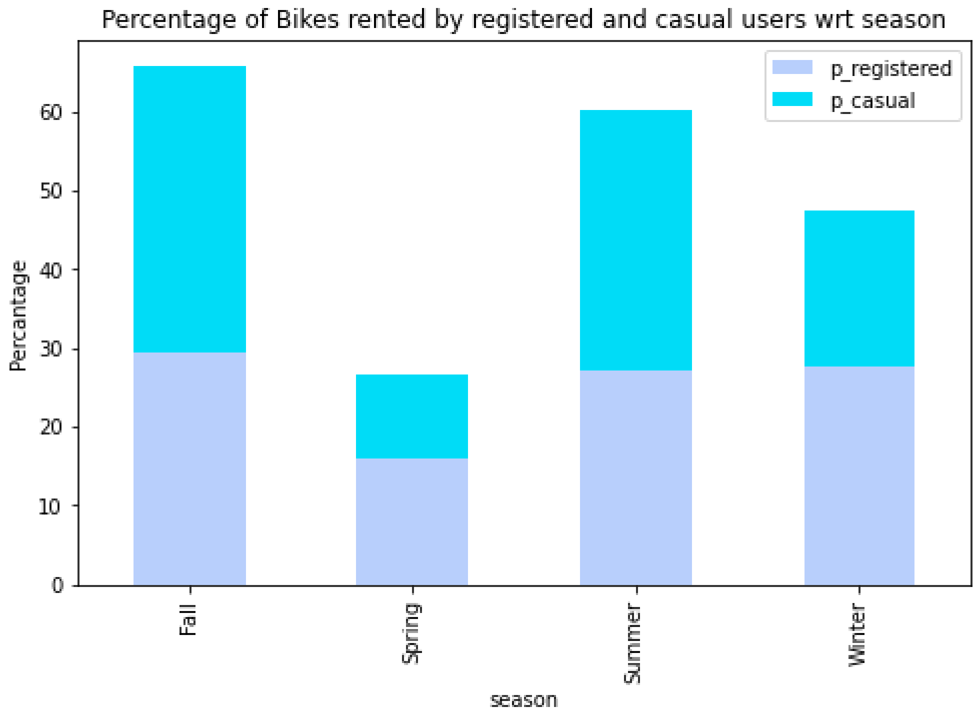
In [136]:

```
#Compute the conditional Probability for season criteria
df2["p_registered"]=df2["registered"]*100/(df2.loc[:,"registered"].sum())
df2["p_casual"]=df2["casual"]*100/((df2.loc[:,"casual"].sum()))
df2.drop(columns=["registered","casual"],inplace=True)
df2.set_index("season",inplace=True)
df2
```

Out[136]:

	p_registered	p_casual
season		
Fall	29.406009	36.395119
Spring	15.997546	10.609867
Summer	27.083145	33.068204
Winter	27.513301	19.926811

In [137]:
```python
df2.plot(kind='bar', stacked=True, color=['#B8CFFC', '#00DCF7'])
plt.ylabel("Percantage")
plt.title("Percentage of Bikes rented by registered and casual users wrt season")
plt.show()
```



Inference-For Registered users the percentage of rented bikes is almost same for all seasons except Spring.For Casual users its high during Fall and Summer and less during Spring and Winter

Through Hypothesis testing it was found that season do impact the number of bikes rented.Let's check which season has a higher population mean through Bootstrapping and 95% Confidence Intervals

In [138]:
```
#re-initialsing dictionaries to store the split dataframes,confidenceintervals,bootstrap mean
dataframes_namelist_pos=0
ci_dict={}
bootstrap_mean_dict={}
dataframes={}
```

In [139]:
```
name,data=DataFrameSplit(df,"season","Spring")
dataframes[name]=data
```

In [140]:
```
#BootStrapFunc function call on split dataframe to find CI
BootStrapFunc(dataframes['df_season_Spring'])
```

```
Mean of the Sampling Distribution is 116.43
95% Confidence Interval is [ 111.69 , 121.29 ]
```

In [141]:
```
name,data=DataFrameSplit(df,"season","Winter")
dataframes[name]=data
BootStrapFunc(dataframes['df_season_Winter'])
```

```
Mean of the Sampling Distribution is 199.06
95% Confidence Interval is [ 191.29 , 205.58 ]
```

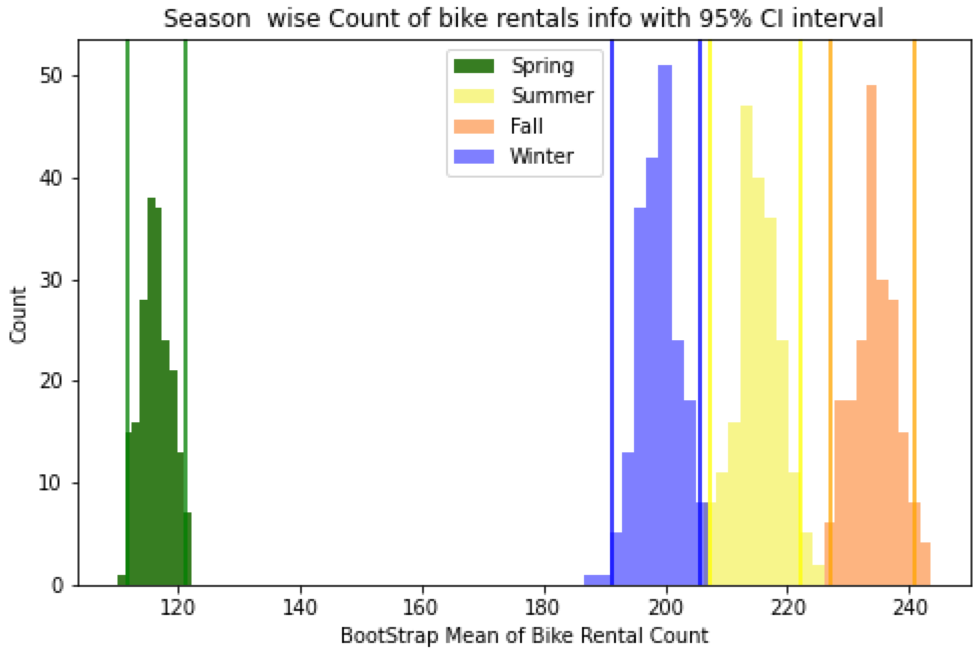
In [142]:
```
name,data=DataFrameSplit(df,"season","Summer")
dataframes[name]=data
BootStrapFunc(dataframes['df_season_Summer'])
```

```
Mean of the Sampling Distribution is 215.12
95% Confidence Interval is [ 207.36 , 222.1 ]
```

In [143]:
```
name,data=DataFrameSplit(df,"season","Fall")
dataframes[name]=data
BootStrapFunc(dataframes['df_season_Fall'])
```

```
Mean of the Sampling Distribution is 234.23
95% Confidence Interval is [ 227.1 , 240.99 ]
```

In [144]:
```
#Plot for 95% CI
plt.hist(bootstrap_mean_dict["df_season_Spring_BS"],label="Spring")
plt.hist(bootstrap_mean_dict["df_season_Summer_BS"],label="Summer",alpha=.5)
plt.hist(bootstrap_mean_dict["df_season_Fall_BS"],label="Fall",alpha=.5)
plt.hist(bootstrap_mean_dict["df_season_Winter_BS"],label="Winter",alpha=.5,color='blue')
plt.axvline(ci_dict['df_season_Spring_CI'][0][0],c='g')
plt.axvline(ci_dict['df_season_Spring_CI'][0][1],c='g')
plt.axvline(ci_dict['df_season_Summer_CI'][0][0],color='yellow')
plt.axvline(ci_dict['df_season_Summer_CI'][0][1],color='yellow')
plt.axvline(ci_dict['df_season_Fall_CI'][0][0],color='orange')
plt.axvline(ci_dict['df_season_Fall_CI'][0][1],color='orange')
plt.axvline(ci_dict['df_season_Winter_CI'][0][0],color='blue')
plt.axvline(ci_dict['df_season_Winter_CI'][0][1],color='blue')
plt.title("Season  wise Count of bike rentals info with 95% CI interval")
plt.xlabel("BootStrap Mean of Bike Rental Count")
plt.ylabel("Count")
plt.legend()
plt.show()
```



Inference-The bike rented is highest in fall and least during Spring.The population mean and CI's of the bikes rented for Summer,Winter and Fall are close to each other.

To check Weather dependent on season or not

Chi-square test is done

Since there is only one observation for Red Weather zone ,removing this row

Null Hypothesis Ho-Weather is independent on seasons

Alternate Hypothesis-Weather is dependent on seasons

Significance level alpha=5%

In [145]:
```python
#number of rows available for each season-weather combination
df.groupby(["season","Zone"])["count"].count()
```

Out[145]:
```
season  Zone
Fall    Green     1930
        Orange     199
        Yellow     604
Spring  Green     1759
        Orange     211
        Red          1
        Yellow     715
Summer  Green     1801
        Orange     224
        Yellow     708
Winter  Green     1702
        Orange     225
        Yellow     807
Name: count, dtype: int64
```

In [148]:
```python
df3=df.loc[df["Zone"]!="Red"]
```

In [149]:
```python
obs=pd.crosstab(index=df3["season"],columns=df3["Zone"],values=df3["count"],aggfunc="sum")
obs
```

Out[149]:

Zone season	Green	Orange	Yellow
Fall	470116	31160	139386
Spring	223009	12919	76406
Summer	426350	27755	134177
Winter	356588	30255	157191

Each cell has more than 5 frequency hence the pre-requisite of chi-square test is met

In [150]:
```
stats.chi2_contingency(obs)
```

Out[150]:
```
(10838.372332480216,
 0.0,
 6,
 array([[453484.88557396,  31364.39195574, 155812.72247031],
        [221081.86259035,  15290.69305984,  75961.44434981],
        [416408.3330293 ,  28800.06497733, 143073.60199337],
        [385087.91880639,  26633.8500071 , 132312.23118651]]))
```

Test statistic is 10838 with p value=0 less than significance level hence rejecting Ho,ie Weather is dependent on seasons

Inference

Through different Hypothesis tests found that weather ,season,working/non working day has significant influence on the bikes rented by the Users