In [ ]:

```
!gdown "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/origi
```

In [163]:

```
#importing libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [6]:

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

In [23]:

```
dd.head()
```

Out[23]:

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

In [8]:

```
dd.shape
#we have 10886 row and 12 columns
```

Out[8]:

```
(10886, 12)
```

In [9]:

```
dd.describe()
```

Out[9]:

	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	6
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	1
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	4
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	6
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	7
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	10

In [11]:

```
dd.isnull().sum()
# we do not have null values
```

Out[11]:

```
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 [25]:

```
#checking datatype
dd.dtypes
```

Out[25]:

```
datetime       object
season          int64
holiday         int64
workingday      int64
weather         int64
temp          float64
atemp         float64
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
dtype: object
```

In [27]:

```
dd.head(1)
```

Out[27]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3

In [63]:

```
(dd.value_counts(['season'])/len(dd))*100
#checking season
```

Out[63]:

```
season
4         25.114826
2         25.105640
3         25.105640
1         24.673893
dtype: float64
```

In [64]:

```
(dd.value_counts(['holiday'])/len(dd))*100
#checking percentage of holidays
```

Out[64]:

```
holiday
0          97.14312
1           2.85688
dtype: float64
```

In [78]:

```
(dd.value_counts(['workingday'])/len(dd))*100
#checking percentage of workingday
```

Out[78]:

```
workingday
1             68.087452
0             31.912548
dtype: float64
```

In [115]:

```
(dd.value_counts(['weather'])/len(dd))*100
#checking percentage of weather
```

Out[115]:

```
weather
1          66.066507
2          26.033437
3           7.890869
4           0.009186
dtype: float64
```

In [171]:

```
pd.pivot_table(dd, values=['count'], index=['season'],
                    aggfunc={'count': np.sum}).reset_index()
# calculating total bike rented in particular season
# most number of bike rented in season 3
```

Out[171]:

	season	count
0	1	312498
1	2	588282
2	3	640662
3	4	544034

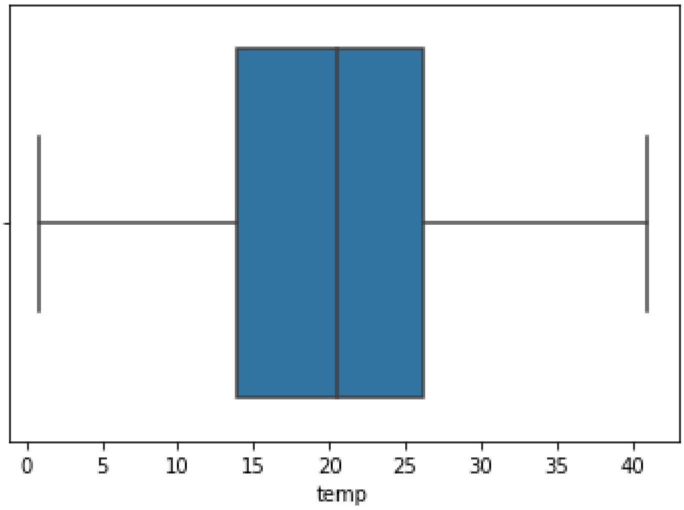
In [328]:

```
sns.boxplot(dd['temp'])
# temperature boxplot for outlier
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decora
tors.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will res
ult in an error or misinterpretation.
  warnings.warn(
```

Out[328]:

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



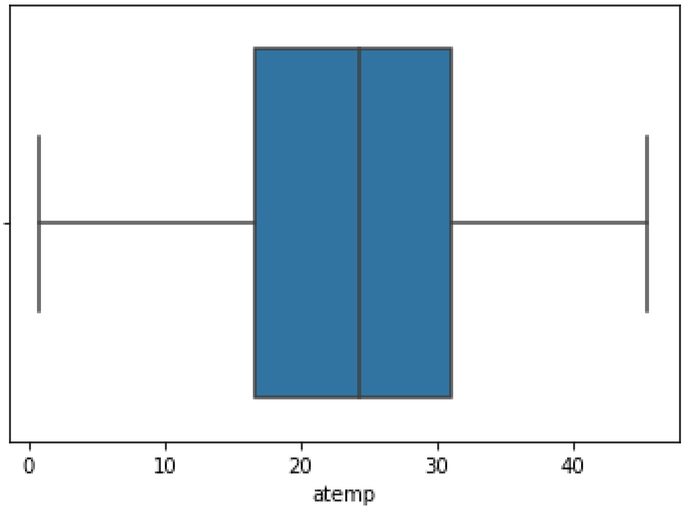
In [218]:

```
sns.boxplot(dd['atemp'])
# atemperature boxplot for outlier
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decora
tors.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will res
ult in an error or misinterpretation.
  warnings.warn(
```

Out[218]:

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



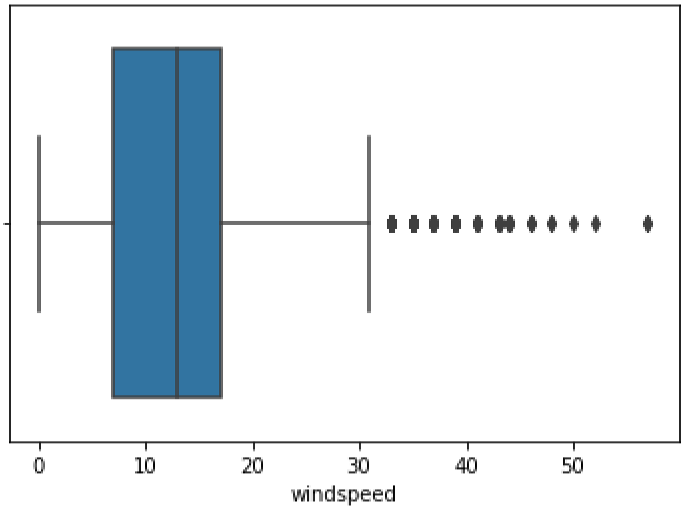
In [329]:

```
sns.boxplot(dd['windspeed'])
# windspeed boxplot for outlier
#we have outlier in windspeed
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decora
tors.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will res
ult in an error or misinterpretation.
  warnings.warn(
```

Out[329]:

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



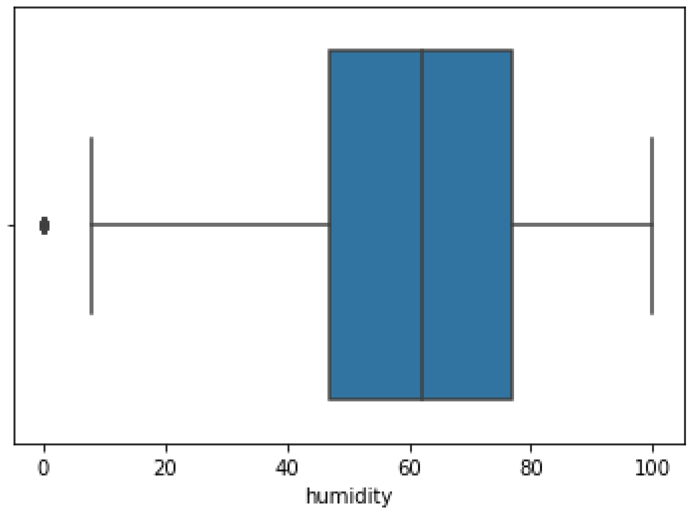
In [220]:

```
sns.boxplot(dd['humidity'])
# humidity boxplot for outlier
```

```
/Users/apple/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decora
tors.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will res
ult in an error or misinterpretation.
  warnings.warn(
```

Out[220]:

<AxesSubplot:xlabel='humidity'>

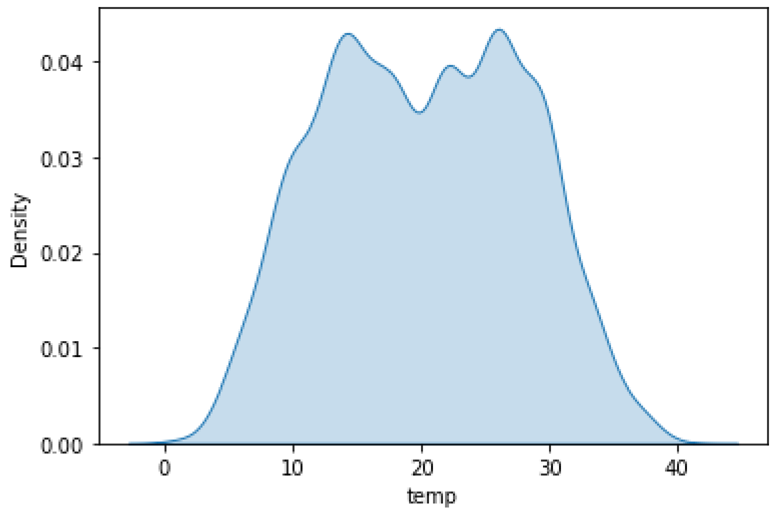


In [222]:

```
sns.kdeplot(dd['temp'],shade=True)
```

Out[222]:

<AxesSubplot:xlabel='temp', ylabel='Density'>

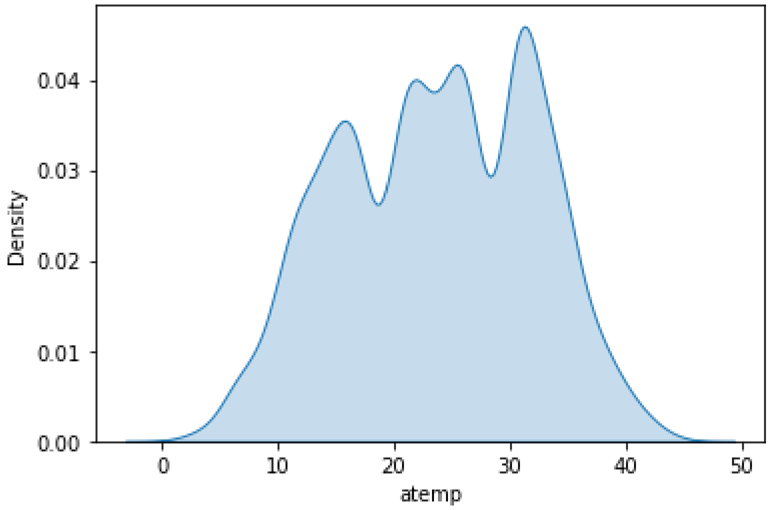


In [223]:

```
sns.kdeplot(dd['atemp'],shade=True)
```

Out[223]:

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

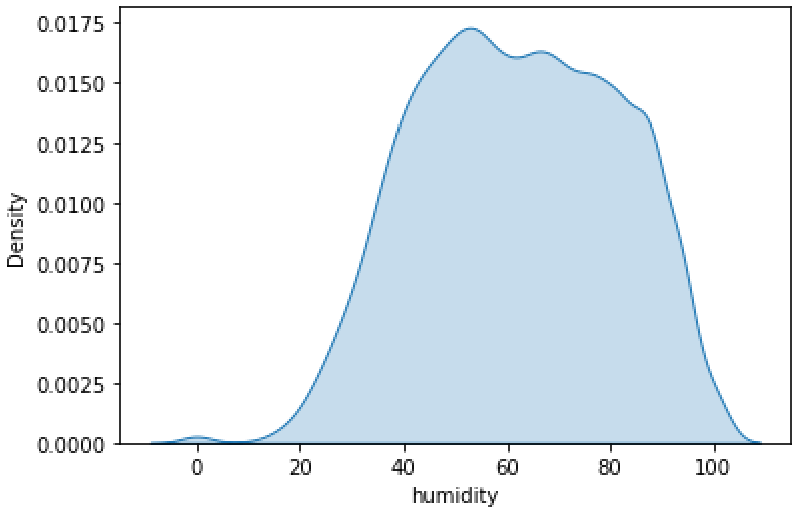


In [224]:

```
sns.kdeplot(dd['humidity'],shade=True)
```

Out[224]:

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

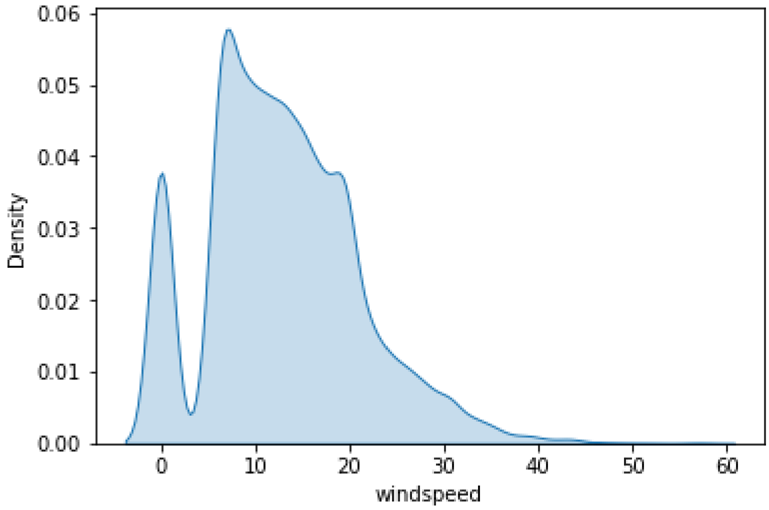


In [225]:

```
sns.kdeplot(dd['windspeed'],shade=True)
#exponential distribution
```

Out[225]:

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

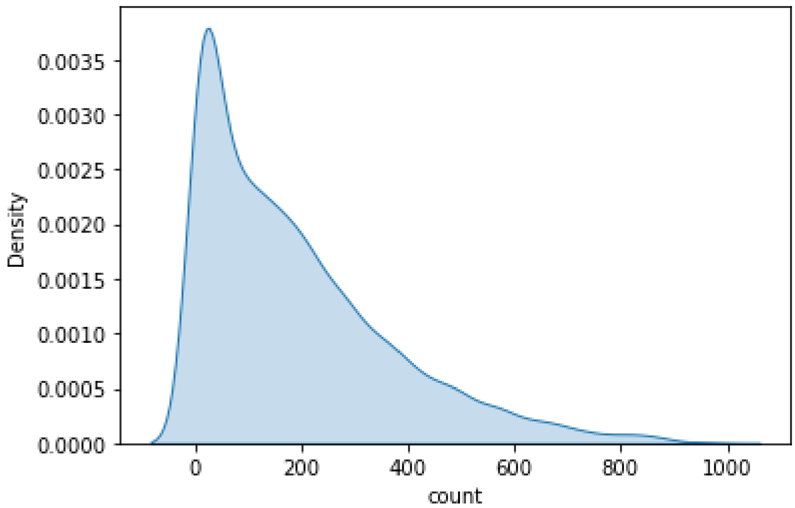


In [230]:

```
sns.kdeplot(dd['count'],shade=True)
```

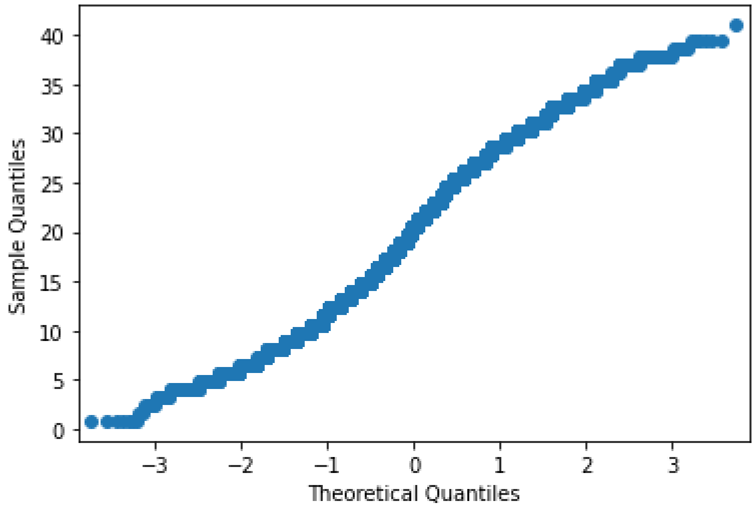
Out[230]:

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



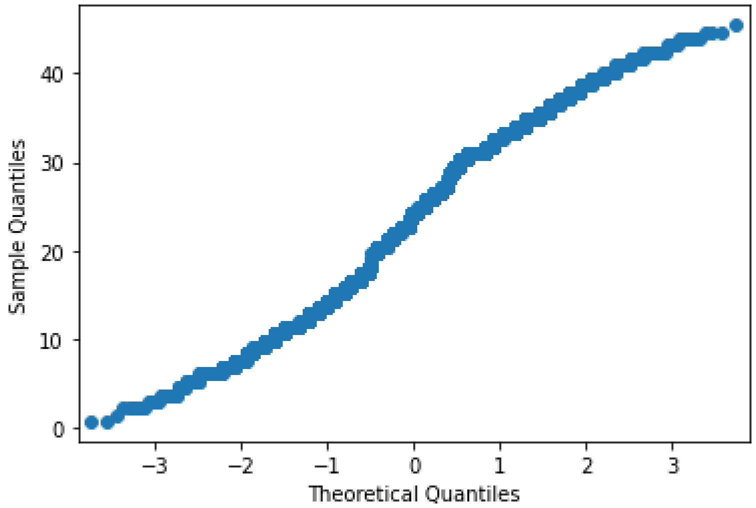
In [210]:

```
import statsmodels.api as sm
sm.qqplot(dd['temp'])
plt.show()
```



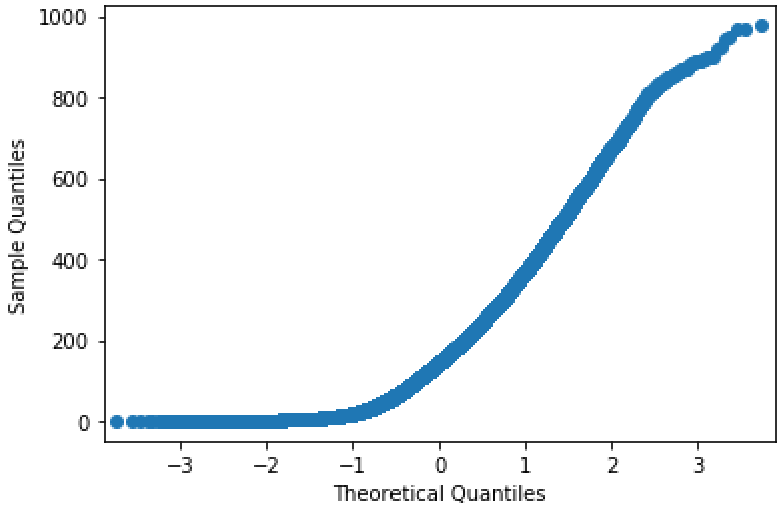
In [212]:

```
sm.qqplot(dd['atemp'])
plt.show()
```



In [231]:

```
sm.qqplot(dd['count'])
plt.show()
```

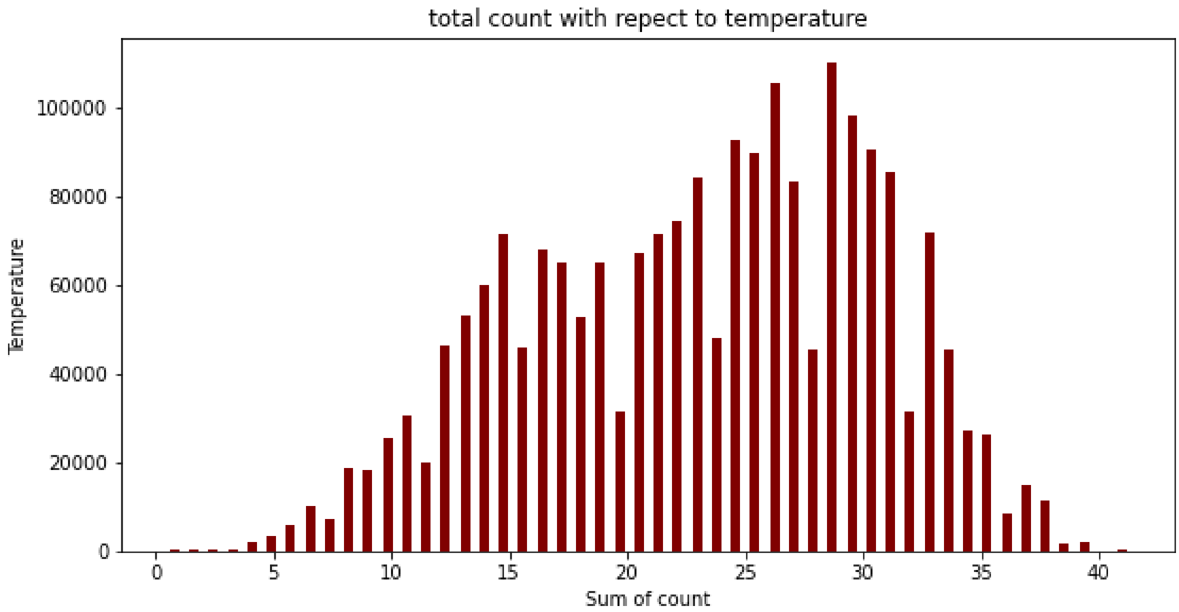


In [242]:

```
temperature = list((dd.groupby('temp').sum('count').reset_index())['temp'])
counts = list((dd.groupby('temp').sum('count').reset_index())['count'])

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(temperature, counts, color ='maroon',
        width = 0.4)
plt.xlabel("Sum of count")
plt.ylabel("Temperature")
plt.title("total count with repect to temperature")
plt.show()
```

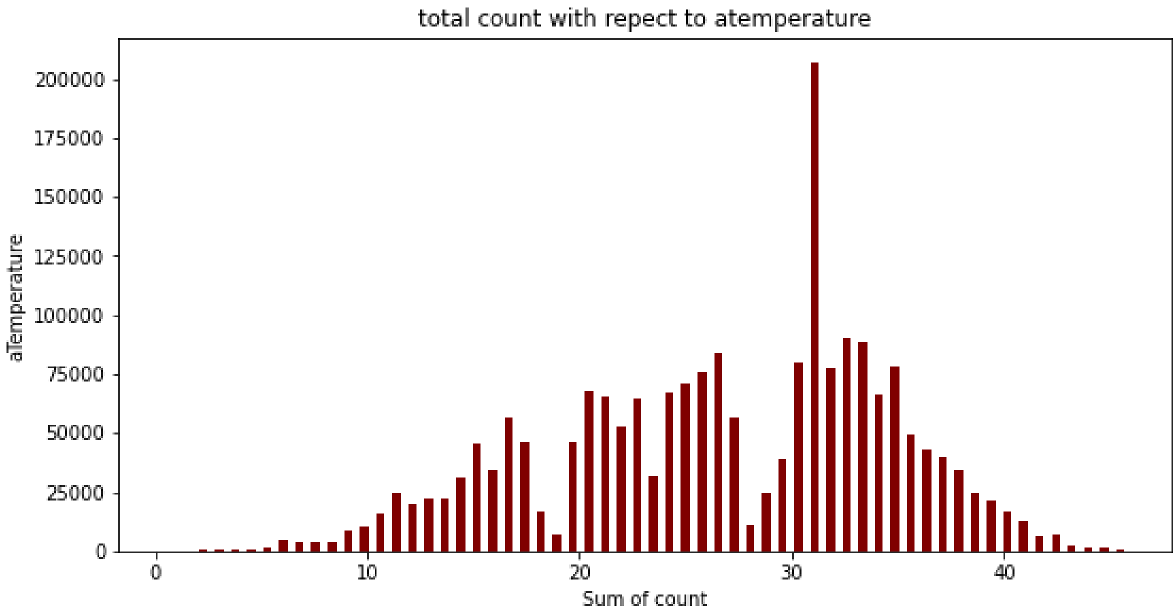


In [245]:

```
temperature = list((dd.groupby('atemp').sum('count').reset_index())['atemp'])
counts = list((dd.groupby('atemp').sum('count').reset_index())['count'])

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(temperature, counts, color ='maroon',
        width = 0.4)
plt.xlabel("Sum of count")
plt.ylabel("aTemperature")
plt.title("total count with repect to atemperature")
plt.show()
```

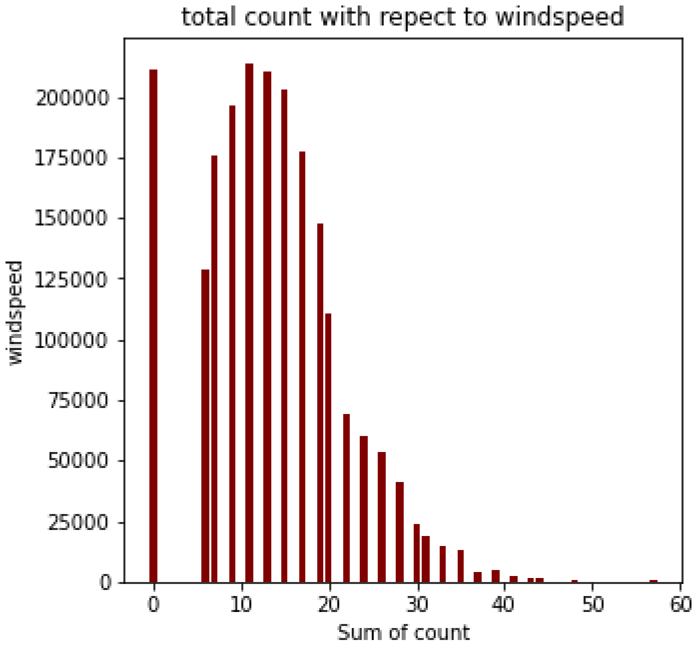


In [250]:

```
temperature = list((dd.groupby('windspeed').sum('count').reset_index())['windspeed']
counts = list((dd.groupby('windspeed').sum('count').reset_index())['count'])

fig = plt.figure(figsize = (5, 5))

# creating the bar plot
plt.bar(temperature, counts, color ='maroon',
        width = 0.8)
plt.xlabel("Sum of count")
plt.ylabel("windspeed")
plt.title("total count with repect to windspeed")
plt.show()
```



In [ ]:

```
temperature = list((dd.groupby('humidity').sum('count').reset_index())['humidity'])
counts = list((dd.groupby('humidity').sum('count').reset_index())['count'])

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(temperature, counts, color ='maroon',
        width = 0.4)
plt.xlabel("Sum of count")
plt.ylabel("humidity")
plt.title("total count with repect to humidity")
plt.show()
```

In [172]:

```
pd.pivot_table(dd, values=['count'], index=['weather'],
                    aggfunc={'count': np.sum}).reset_index()
```

Out[172]:

	weather	count
0	1	1476063
1	2	507160
2	3	102089
3	4	164

In [173]:

```
pd.pivot_table(dd, values=['count'], index=['holiday'],
                    aggfunc={'count': np.sum}).reset_index()
```

Out[173]:

	holiday	count
0	0	2027668
1	1	57808

In [177]:

```
pd.pivot_table(dd, values=['count'], index=['workingday'],
                    aggfunc='sum').reset_index()
```

Out[177]:

	workingday	count
0	0	654872
1	1	1430604

In [252]:

```
pd.crosstab([dd['season']], [dd['weather']], dd['holiday'], aggfunc='sum')
```

Out[252]:

weather	**1**	**2**	**3**	**4**
season				
1	42.0	23.0	6.0	0.0
2	41.0	7.0	0.0	NaN
3	52.0	40.0	4.0	NaN
4	69.0	22.0	5.0	NaN

In [253]:

```
pd.crosstab([dd['season']], [dd['weather']], dd['workingday'], aggfunc='sum')
```

Out[253]:

weather	1	2	3	4
season				
1	1213.0	474.0	140.0	1.0
2	1244.0	468.0	181.0	NaN
3	1306.0	404.0	135.0	NaN
4	1076.0	591.0	179.0	NaN

In [193]:

```
pd.crosstab([dd['season'],dd['weather']], [dd['holiday']], dd['count'], aggfunc='sum
```

Out[193]:

	holiday	0	1
season	**weather**		
1	**1**	219219.0	3790.0
	2	75112.0	1294.0
	3	12754.0	165.0
	4	164.0	NaN
2	**1**	418409.0	7941.0
	2	132622.0	1555.0
	3	27755.0	NaN
3	**1**	456366.0	13750.0
	2	130734.0	8652.0
	3	30731.0	429.0
4	**1**	344652.0	11936.0
	2	150131.0	7060.0
	3	29019.0	1236.0

In [194]:

```
pd.crosstab([dd['season'],dd['weather']], [dd['workingday']], dd['count'], aggfunc='
```

Out[194]:

	workingday	0	1
season	**weather**		
1	1	66706.0	156303.0
	2	21144.0	55262.0
	3	2864.0	10055.0
	4	NaN	164.0
2	1	135007.0	291343.0
	2	46489.0	87688.0
	3	5566.0	22189.0
3	1	153036.0	317080.0
	2	41402.0	97984.0
	3	12040.0	19120.0
4	1	123724.0	232864.0
	2	40342.0	116849.0
	3	6552.0	23703.0

In [196]:

```
pd.crosstab([dd['season'],dd['weather']], [dd['workingday'], dd['holiday']], dd['cou
```

Out[196]:

	workingday	0		1
	holiday	0	1	0
season	weather			
1	1	62916.0	3790.0	156303.0
	2	19850.0	1294.0	55262.0
	3	2699.0	165.0	10055.0
	4	NaN	NaN	164.0
2	1	127066.0	7941.0	291343.0
	2	44934.0	1555.0	87688.0
	3	5566.0	NaN	22189.0
3	1	139286.0	13750.0	317080.0
	2	32750.0	8652.0	97984.0
	3	11611.0	429.0	19120.0
4	1	111788.0	11936.0	232864.0
	2	33282.0	7060.0	116849.0
	3	5316.0	1236.0	23703.0

In [200]:

```
pd.crosstab([dd['season'],dd['weather']], [dd['workingday'], dd['holiday']], dd['cas
```

Out[200]:

	workingday	0		1
	holiday	0	1	0
season	weather			
1	1	17163.0	410.0	13473.0
	2	4791.0	158.0	4860.0
	3	304.0	10.0	430.0
	4	NaN	NaN	6.0
2	1	47220.0	1606.0	46858.0
	2	16165.0	234.0	12977.0
	3	1864.0	NaN	2748.0
3	1	48810.0	5138.0	52149.0
	2	11213.0	3546.0	15310.0
	3	4186.0	144.0	2222.0
4	1	31600.0	2269.0	23204.0
	2	6639.0	1483.0	9870.0
	3	911.0	173.0	1991.0

In [201]:

```
pd.pivot_table(dd, values=['count','registered', 'casual'], index=['season','weather
                        aggfunc={'registered': np.sum,'casual':np.sum,'count': np.sum})
```

Out[201]:

season	weather	casual	count	registered
1	1	31046	223009	191963
	2	9809	76406	66597
	3	744	12919	12175
	4	6	164	158
2	1	95684	426350	330666
	2	29376	134177	104801
	3	4612	27755	23143
3	1	106097	470116	364019
	2	30069	139386	109317
	3	6552	31160	24608
4	1	57073	356588	299515
	2	17992	157191	139199
	3	3075	30255	27180

In [161]:

```
pd.crosstab(dd['season'], dd['weather'],values=dd['count'], aggfunc='sum')
```

Out[161]:

weather season	1	2	3	4
1	223009.0	76406.0	12919.0	164.0
2	426350.0	134177.0	27755.0	NaN
3	470116.0	139386.0	31160.0	NaN
4	356588.0	157191.0	30255.0	NaN

2 - Hypothesis Testing

--------------------Two Sample T-test---------------------

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

We have total 654872 bike rented on non working day and 1430604 bike rented on working day. For t-test we will randomly select sample(<30) from respective populations.

In [301]:

```
# here we are creating two empty lists and  list of working day bike
# rented count and non working day bike rented count.

Working_count=[]
Non_working_count=[]

for i in range(len(dd)):
    if dd['workingday'][i]==0:
        Non_working_count.append(dd['count'][i])
    else:
        Working_count.append(dd['count'][i])
```

In [302]:

```
# randomly selecting sample of size 25 of rented bike count on working day and non w
import random
random.seed(10)
Working_count_sample=random.sample(Working_count, 25)
Non_Working_count_sample=random.sample(Non_working_count, 25)
```

In [303]:

```
X1_bar=np.mean(Working_count_sample)
X2_bar=np.mean(Non_Working_count_sample)
std1=np.std(Working_count_sample)
std2=np.std(Non_Working_count_sample)
#We have sample size n1=25, n2=25
#we have sample means as X1_bar and X2_bar and standard deviation as std1 and std2
```

In [304]:

```
# Suppose mu1 and mu2 be the respective means of count
# Null hypothesis, H0: mu1-mu2=0   mens are equal
# Alternative hypothesis, H1: mu1-mu2>0
# level of signigicance: alpha=0.05
# test statistics: t
# Decision rule: if t(calculated)> t(critical(1.677)), then reject H0
```

In [305]:

```
Sp=24*((std1)**2)+24*((std2)**2)/48
```

In [306]:

```
Sp
```

Out[306]:

```
954124.0144
```

In [308]:

```
t=(X1_bar-X2_bar)/np.sqrt(Sp*((1/25)+(1/25)))
t
```

Out[308]:

```
0.28623270429975184
```

In [310]:

```
#since t value(.2862) less than 1.677 we can not reject the null hypothesis
```

ANNOVA

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

In [313]:

```
#Let first calculate F-statistics for weather
weather1=[]
weather2=[]
weather3=[]
weather4=[]
for i in range(len(dd)):
    if dd['weather'][i]==1:
        weather1.append(dd['count'][i])
    if dd['weather'][i]==2:
        weather2.append(dd['count'][i])
    if dd['weather'][i]==3:
        weather3.append(dd['count'][i])
    if dd['weather'][i]==1:
        weather4.append(dd['count'][i])
```

In [ ]:

```
#mu1, mu2, mu3, mu4 are the means of count in waether1, waether2, waether3, waether4
# Null Hypothesis mu1=mu2=mu3=mu4
# alternative hypothesis : Not all the means are equal
# level of significance 0.05
# test statistics F-test
```

In [315]:

```
from scipy.stats import f_oneway



# Conduct the one-way ANOVA
f_oneway(weather1, weather2, weather3, weather4)
```

Out[315]:

```
F_onewayResult(statistic=71.16449949906588, pvalue=9.512123371458035e-
46)
```

In [319]:

```
# we have P value close to zero so we reject the null hypothesis
# we can say that there is some impact of weather on count in our data
```

In [320]:

```
# #calculate F-statistics for season
```

In [323]:

```
season1=[]
season2=[]
season3=[]
season4=[]
for i in range(len(dd)):
    if dd['season'][i]==1:
        season1.append(dd['count'][i])
    if dd['season'][i]==2:
        season2.append(dd['count'][i])
    if dd['season'][i]==3:
        season3.append(dd['count'][i])
    if dd['season'][i]==1:
        season4.append(dd['count'][i])
```

In [324]:

```
#mu1, mu2, mu3, mu4 are the means of count in season1, season2, season3, season4 res
# Null Hypothesis mu1=mu2=mu3=mu4
# alternative hypothesis : Not all the means are equal
# level of significance 0.05
# test statistics F-test
```

In [325]:

```
# Conduct the one-way ANOVA
f_oneway(season1, season2, season3, season4)
```

Out[325]:

```
F_onewayResult(statistic=401.7230336505781, pvalue=1.9042750553991286e
-247)
```

In [326]:

```
# we have P value close to zero so we reject the null hypothesis
# we can say that there is some impact of season on count in our data
```

Chi-square

```
INFO:  to check weather dependency on season, we can not apply chi square test
```