kuntalkole-yulu-hypothesis-test

May 26, 2024

[89]:
```
# importing necessary library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[90]:
```
# uploading the file in colab
!wget 'https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/
  ↪original/bike_sharing.csv?1642089089'
```

```
--2024-05-26 05:27:16--  https://d2beiqkhq929f0.cloudfront.net/public_assets/ass
ets/000/001/428/original/bike_sharing.csv?1642089089
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)…
108.157.172.10, 108.157.172.176, 108.157.172.173, …
Connecting to d2beiqkhq929f0.cloudfront.net
(d2beiqkhq929f0.cloudfront.net)|108.157.172.10|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 648353 (633K) [text/plain]
Saving to: ‘bike_sharing.csv?1642089089.1’

bike_sharing.csv?16 100%[===================>] 633.16K  --.-KB/s    in 0.1s

2024-05-26 05:27:16 (6.47 MB/s) - ‘bike_sharing.csv?1642089089.1’ saved
[648353/648353]
```

[91]:
```
# load the csv file
data=pd.read_csv('bike_sharing.csv?1642089089')
data.head()
```

[91]:

	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					
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635					
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635					
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395					
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395					

0	81	0.0	3	13	16
1	80	0.0	8	32	40
2	80	0.0	5	27	32
3	75	0.0	3	10	13
4	75	0.0	0	1	1

0.1 Data Observation

[ ]:
```
# get the total no of rows and columns
data.shape
```

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

[ ]:
```
# get the data types of all columns
data.dtypes
```

[ ]:
```
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
```

[92]:
```
# changing the datetime column datatype to 'datetime'
data['datetime'] = pd.to_datetime(data['datetime'])
```

[ ]:
```
# checking the datatype of columns
data.dtypes
```

[ ]:
```
datetime      datetime64[ns]
season                 int64
holiday                int64
workingday             int64
weather                int64
temp                 float64
atemp                float64
humidity               int64
windspeed            float64
casual                 int64
registered             int64
```

```
count                    int64
dtype: object
```

[ ]:
```
# getting the 5 rows of data
data.head()
```

[ ]:
```
             datetime  season  holiday  workingday  weather  temp   atemp  \
0 2011-01-01 00:00:00       1        0           0        1  9.84  14.395
1 2011-01-01 01:00:00       1        0           0        1  9.02  13.635
2 2011-01-01 02:00:00       1        0           0        1  9.02  13.635
3 2011-01-01 03:00:00       1        0           0        1  9.84  14.395
4 2011-01-01 04:00:00       1        0           0        1  9.84  14.395

   humidity  windspeed  casual  registered  count
0        81        0.0       3          13     16
1        80        0.0       8          32     40
2        80        0.0       5          27     32
3        75        0.0       3          10     13
4        75        0.0       0           1      1
```

[93]:
```
# creating 2 separate columns from datetime column
data['date']=data['datetime'].dt.date
data['year']=data['datetime'].dt.year
data['month'] = data['datetime'].dt.month
data['time']=data['datetime'].dt.time
```

[94]:
```
# removing the datetime column
data = data.drop('datetime', axis = 1)
```

[ ]:
```
# again checking the data
data.head()
```

[ ]:
```
   season  holiday  workingday  weather  temp   atemp  humidity  windspeed  \
0       1        0           0        1  9.84  14.395        81        0.0
1       1        0           0        1  9.02  13.635        80        0.0
2       1        0           0        1  9.02  13.635        80        0.0
3       1        0           0        1  9.84  14.395        75        0.0
4       1        0           0        1  9.84  14.395        75        0.0

   casual  registered  count        date  year  month      time
0       3          13     16  2011-01-01  2011      1  00:00:00
1       8          32     40  2011-01-01  2011      1  01:00:00
2       5          27     32  2011-01-01  2011      1  02:00:00
3       3          10     13  2011-01-01  2011      1  03:00:00
4       0           1      1  2011-01-01  2011      1  04:00:00
```

[95]:
```
# renaming 3 columns name
data= data.rename(columns = {'count':'total_count', 'casual':
 ↪'casual_user','registered': 'registered_user'})
```

[ ]:
```
data.head(3)
```

[ ]:
```
   season  holiday  workingday  weather  temp   atemp  humidity  windspeed  \
0       1        0           0        1  9.84  14.395        81        0.0
1       1        0           0        1  9.02  13.635        80        0.0
2       1        0           0        1  9.02  13.635        80        0.0

   casual_user  registered_user  total_count        date  year  month  \
0            3               13           16  2011-01-01  2011      1
1            8               32           40  2011-01-01  2011      1
2            5               27           32  2011-01-01  2011      1

       time
0  00:00:00
1  01:00:00
2  02:00:00
```

[ ]:
```
# checking unique values in date column and time columns
data[['date','time','month','year']].nunique()
```

[ ]:
```
date     456
time      24
month     12
year       2
dtype: int64
```

[ ]:
```
# checking if there is any null values in any columns
data.isnull().sum()
```

[ ]:
```
season            0
holiday           0
workingday        0
weather           0
temp              0
atemp             0
humidity          0
windspeed         0
casual_user       0
registered_user   0
total_count       0
date              0
year              0
month             0
```

```
time                0
dtype: int64
```

[ ]:
```
# gettin the no of unique values of every columns
data.nunique()
```

[ ]:
```
season               4
holiday              2
workingday           2
weather              4
temp                49
atemp               60
humidity            89
windspeed           28
casual_user        309
registered_user    731
total_count        822
date               456
year                 2
month               12
time                24
dtype: int64
```

0.2 Univariate Analysis

0.2.1 continuous variables

[ ]:
```
data.head()
```

[ ]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	1	0	0	1	9.84	14.395	81	0.0
1	1	0	0	1	9.02	13.635	80	0.0
2	1	0	0	1	9.02	13.635	80	0.0
3	1	0	0	1	9.84	14.395	75	0.0
4	1	0	0	1	9.84	14.395	75	0.0

	casual_user	registered_user	total_count	date	year	month
0	3	13	16	2011-01-01	2011	1
1	8	32	40	2011-01-01	2011	1
2	5	27	32	2011-01-01	2011	1
3	3	10	13	2011-01-01	2011	1
4	0	1	1	2011-01-01	2011	1

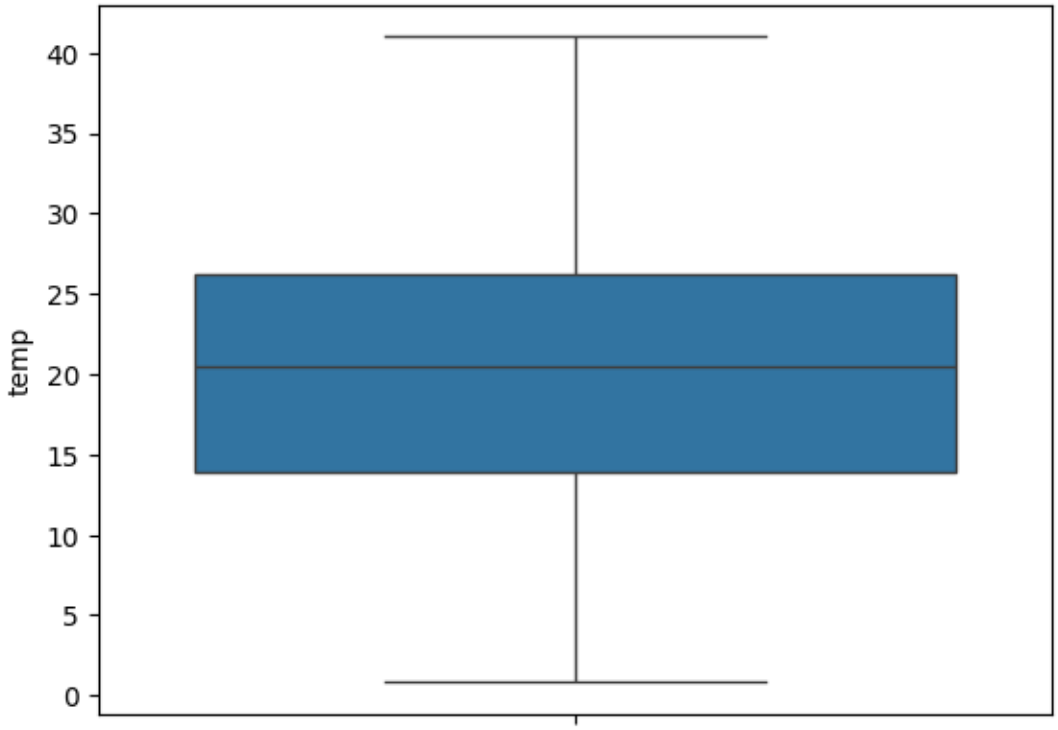
	time
0	00:00:00
1	01:00:00
2	02:00:00

```
3  03:00:00
4  04:00:00
```

[ ]:
```
# to get the mean, standard deviation, max of temp column
data['temp'].describe()
```

[ ]:
```
count    10886.00000
mean        20.23086
std          7.79159
min          0.82000
25%         13.94000
50%         20.50000
75%         26.24000
max         41.00000
Name: temp, dtype: float64
```

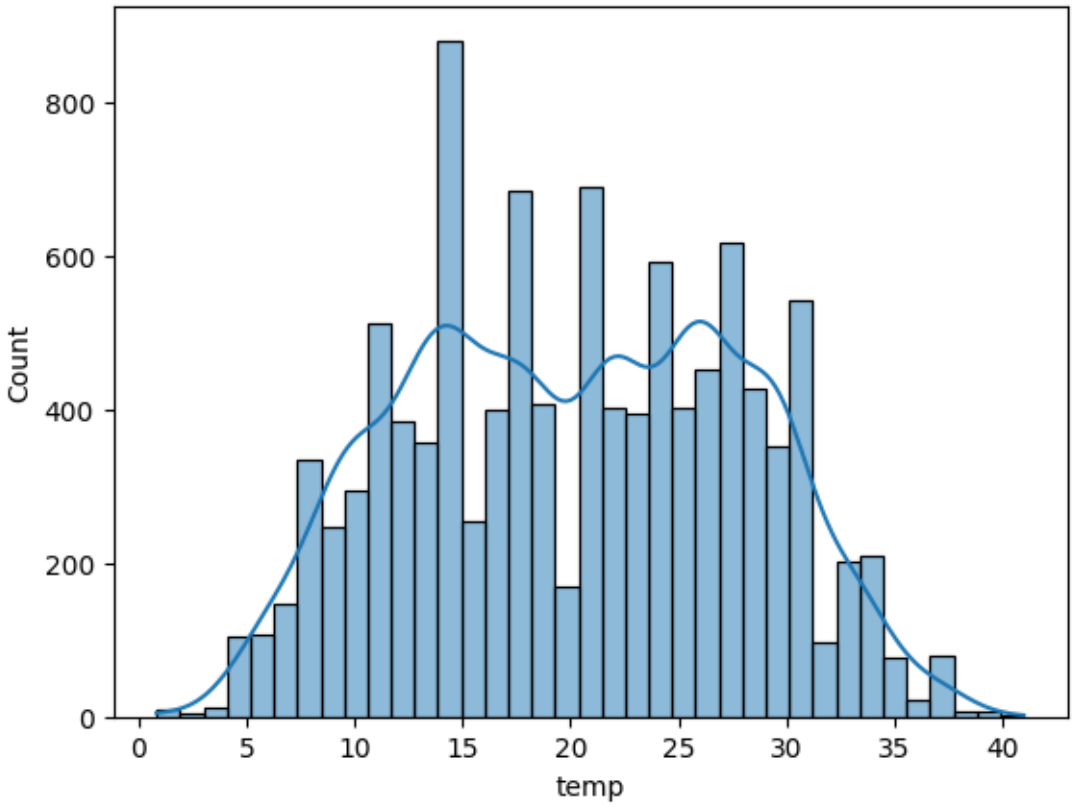
[ ]:
```
sns.boxplot(data['temp'])
plt.show()
```



from the above chart,
we can see that there is **no OUTLIERS in 'temp' column.**
and the temperature range is 0.82 to 41 'C .

25% to 75% of the data is in range of 13.94 to 26.24 'C temarature.

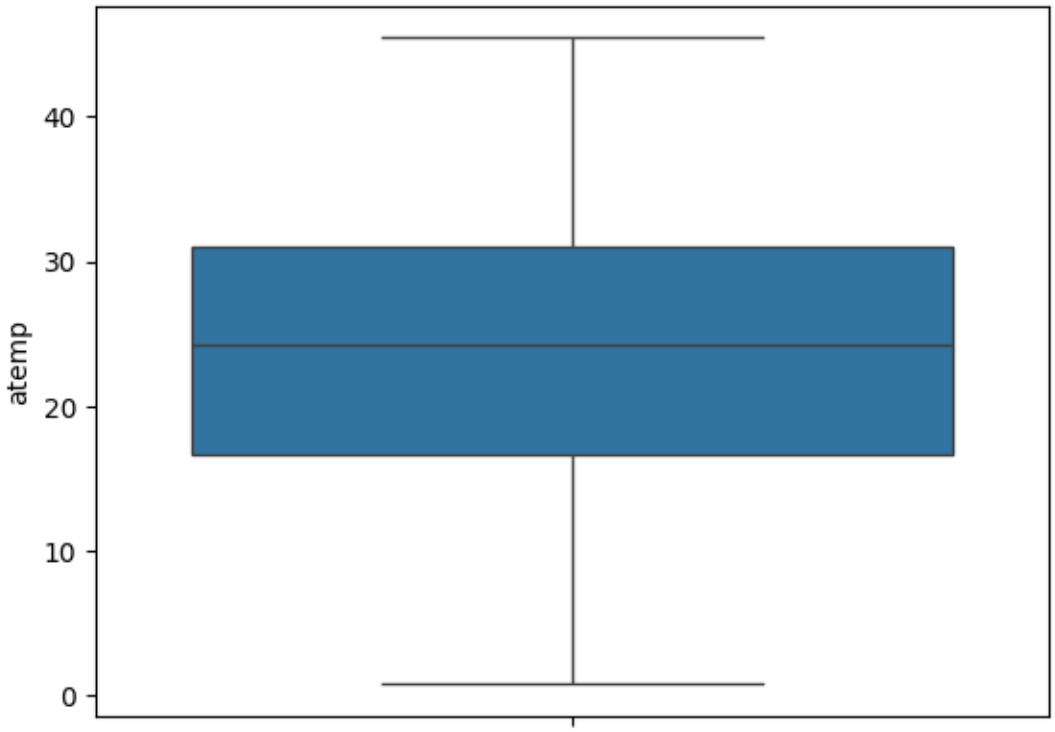
[ ]:
```
# to see the frequency distribution of 'temp'
sns.histplot(data['temp'], kde = True)
plt.show()
```



[ ]:
```
# to get the mean, standard deviation, max of atemp column
data['atemp'].describe()
```

[ ]:
```
count    10886.000000
mean        23.655084
std          8.474601
min          0.760000
25%         16.665000
50%         24.240000
75%         31.060000
max         45.455000
Name: atemp, dtype: float64
```

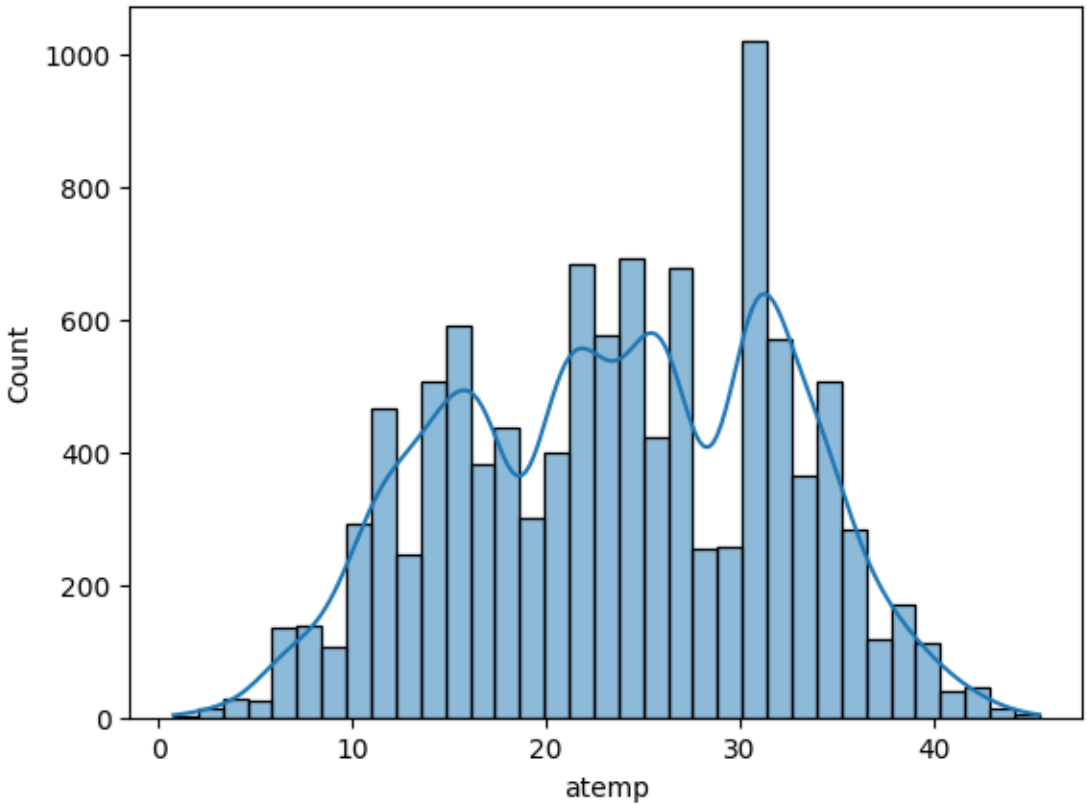
[ ]:
```
sns.boxplot(data['atemp'])
plt.show()
```



from the above chart,
we can see that there is **no OUTLIERS in 'atemp' column.**
and the feel temperature range is 0.76 to 45.455 'C .

25% to 75% of the data is in range of 16.665 to 31.06 'C temarature.

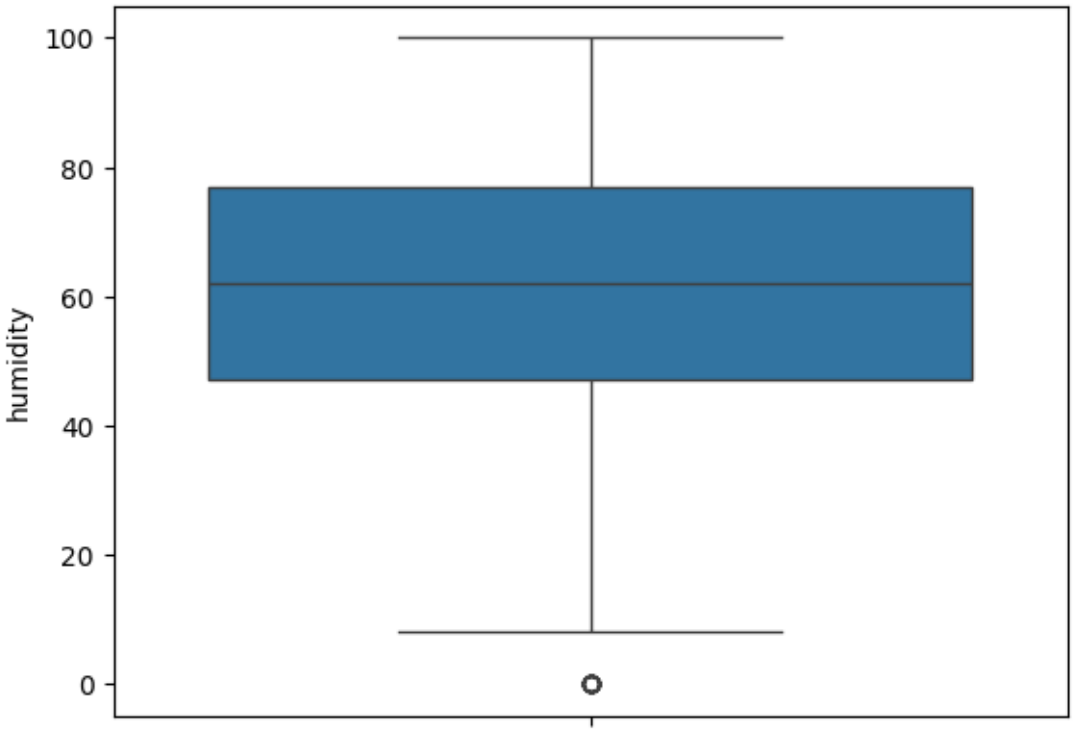
[ ]:
```
# to see the frequency distribution of 'atemp'
sns.histplot(data['atemp'], kde = True)
plt.show()
```



[ ]:
```
# to get the mean, standard deviation, max of humidity column
data['humidity'].describe()
```

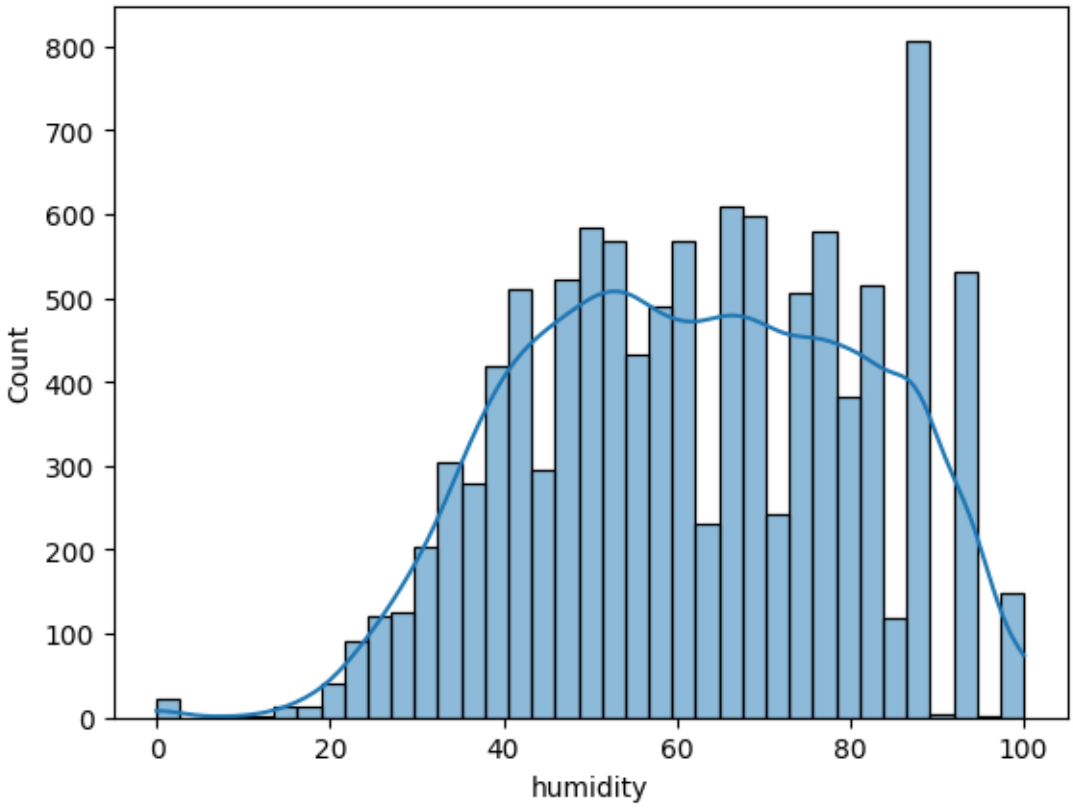
[ ]:
```
count    10886.000000
mean        61.886460
std         19.245033
min          0.000000
25%         47.000000
50%         62.000000
75%         77.000000
max        100.000000
Name: humidity, dtype: float64
```

[ ]:
```
sns.boxplot(data['humidity'])
plt.show()
```



From the above chart, we can see that there is ONE OUTLIERS in 'humidity' column at 0 point. 25% to 75% of the data is in range of 47 to 77 humidity.

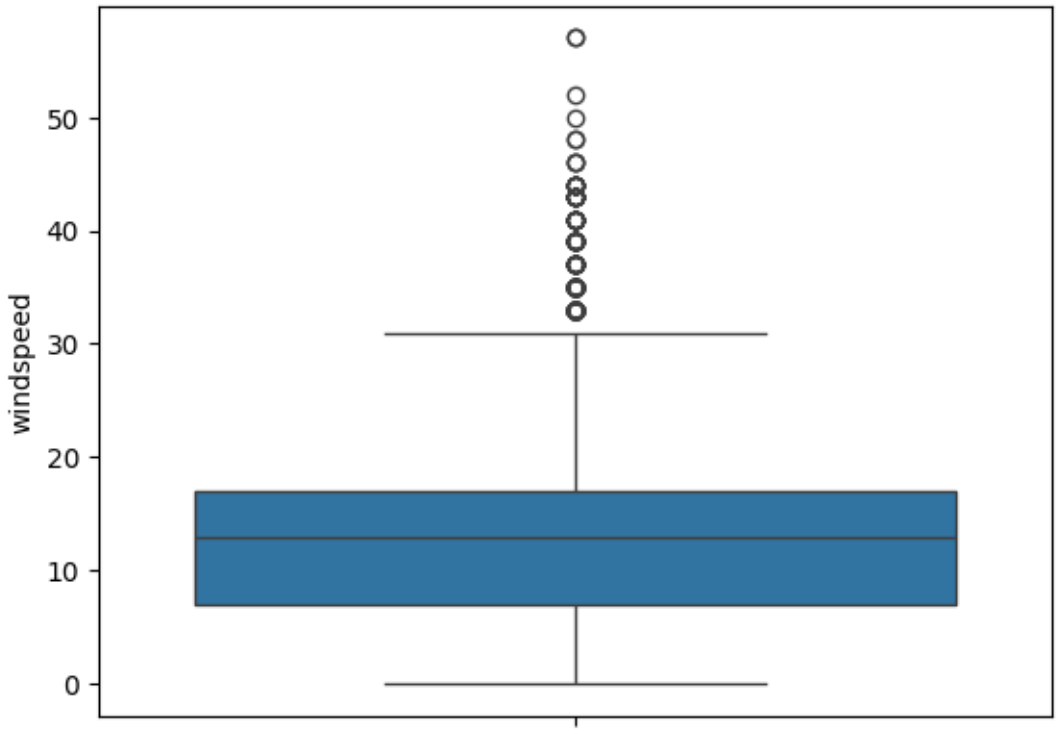
[ ]:
```
# to see the frequency distribution of 'humidity'
sns.histplot(data['humidity'], kde = True)
plt.show()
```



[ ]:
```
# to get the mean, standard deviation, max of windspeed column
data['windspeed'].describe()
```

[ ]:
```
count    10886.000000
mean        12.799395
std          8.164537
min          0.000000
25%          7.001500
50%         12.998000
75%         16.997900
max         56.996900
Name: windspeed, dtype: float64
```

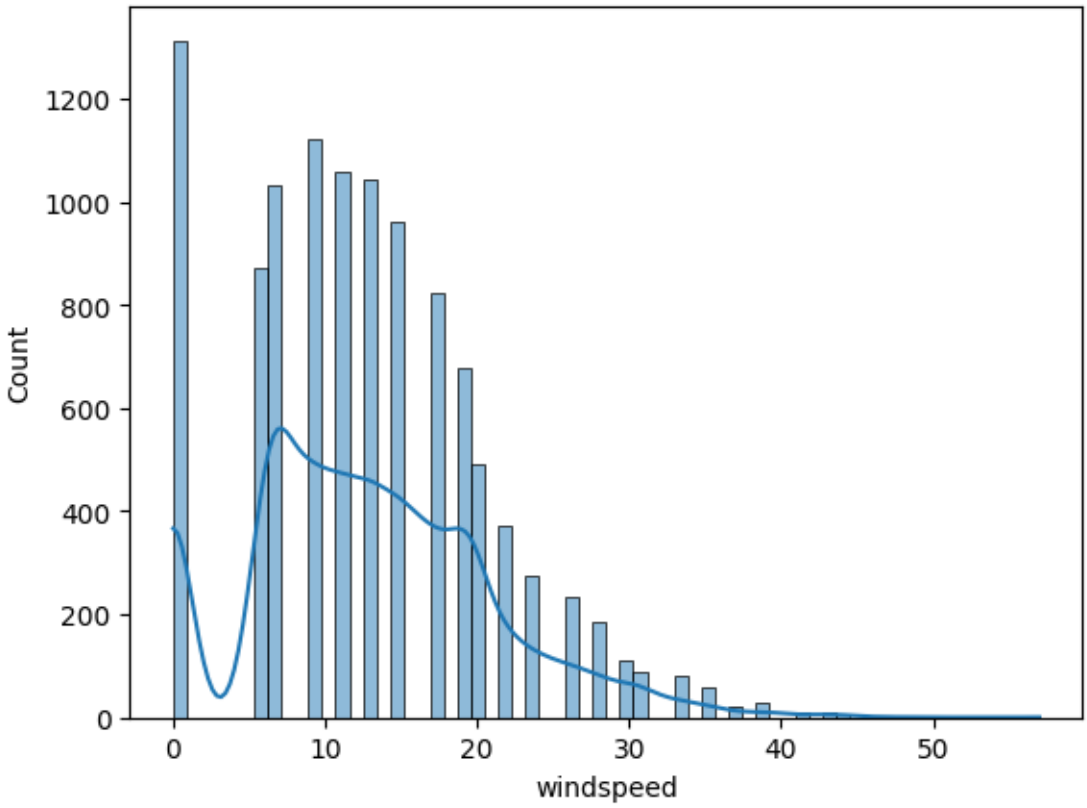
[ ]:
```
sns.boxplot(data['windspeed'])
plt.show()
```



From the above chart, we can see that there is alot of OUTLIERS in 'windspeed' column.

25% to 75% of the data is in range of 7.0015 to 16.9979 speed.

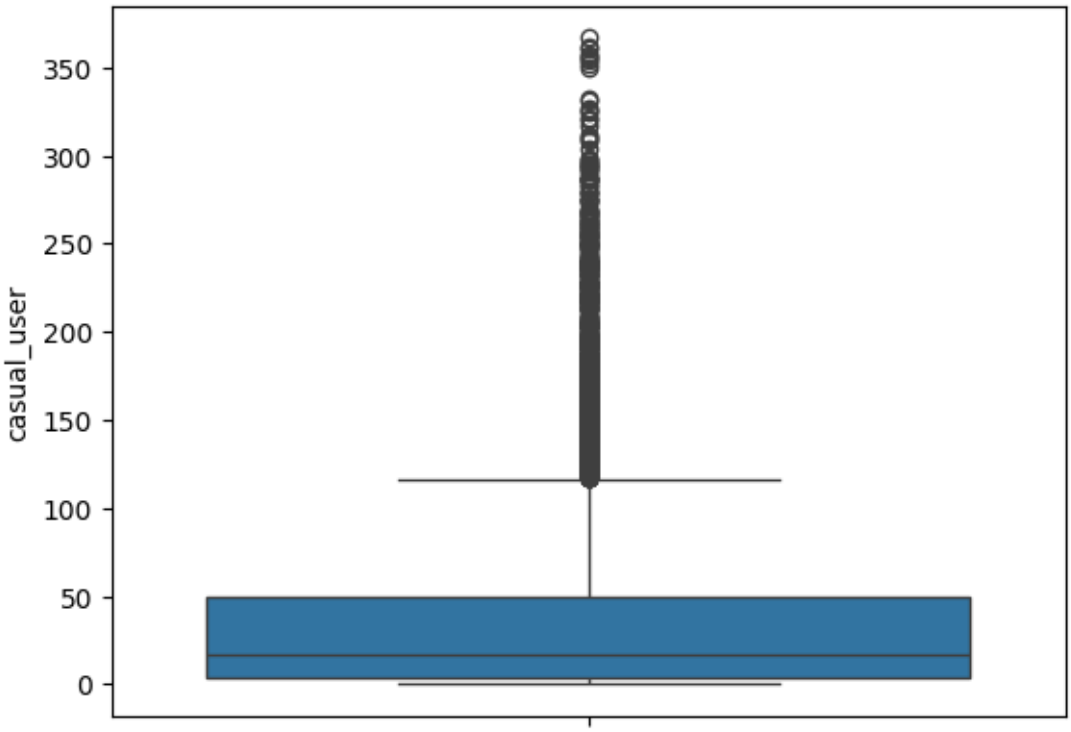
[ ]:
```
# to see the frequency distribution of 'windspeed'
sns.histplot(data['windspeed'], kde = True)
plt.show()
```



[ ]:
```
# to get the mean, standard deviation, max of casual_user column
data['casual_user'].describe()
```

[ ]:
```
count    10886.000000
mean        36.021955
std         49.960477
min          0.000000
25%          4.000000
50%         17.000000
75%         49.000000
max        367.000000
Name: casual_user, dtype: float64
```

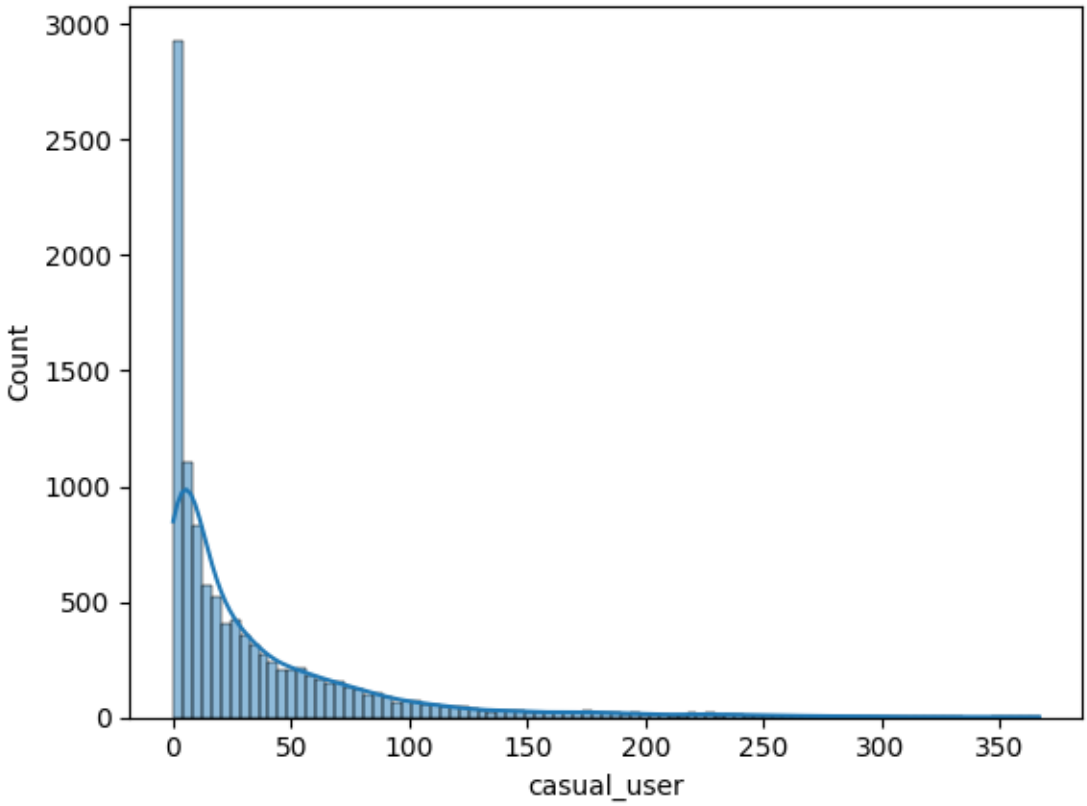
[ ]:
```
sns.boxplot(data['casual_user'])
plt.show()
```



From the above chart, we can see that there is alot of OUTLIERS in 'casual_user' column.

25% to 75% of the data is in range of 4 to 49 .

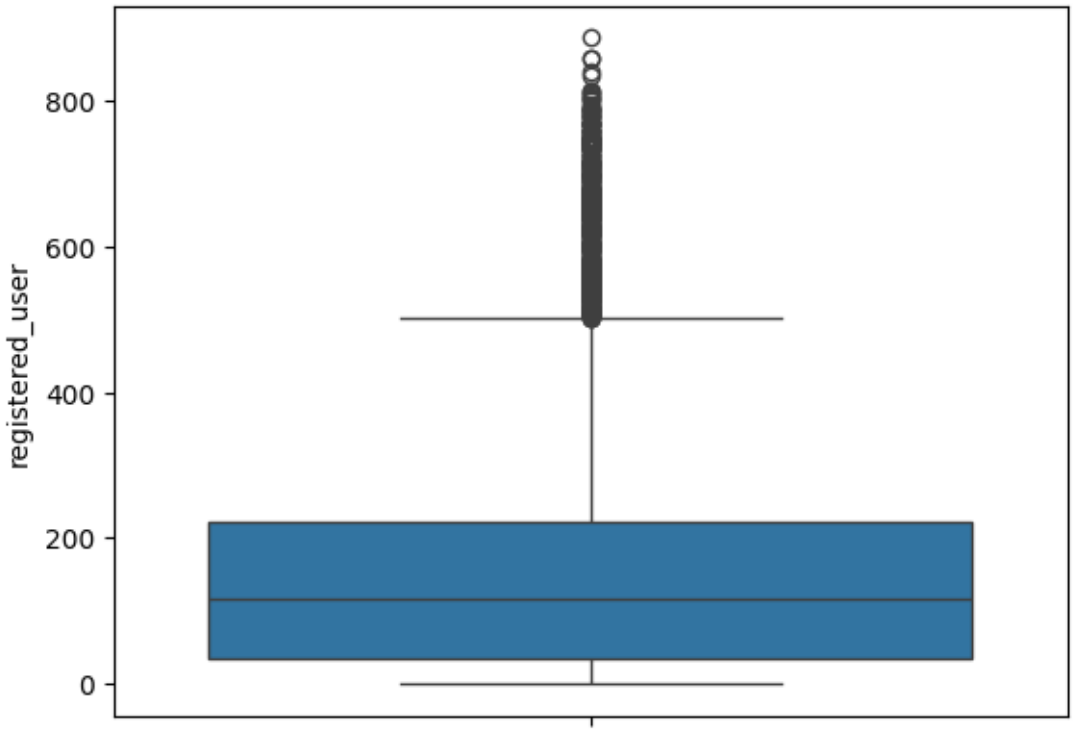
[ ]:
```
# to see the frequency distribution of 'casual_user'
sns.histplot(data['casual_user'], kde = True)
plt.show()
```



[ ]:
```
# to get the mean, standard deviation, max of registered_user column
data['registered_user'].describe()
```

[ ]:
```
count    10886.000000
mean       155.552177
std        151.039033
min          0.000000
25%         36.000000
50%        118.000000
75%        222.000000
max        886.000000
Name: registered_user, dtype: float64
```

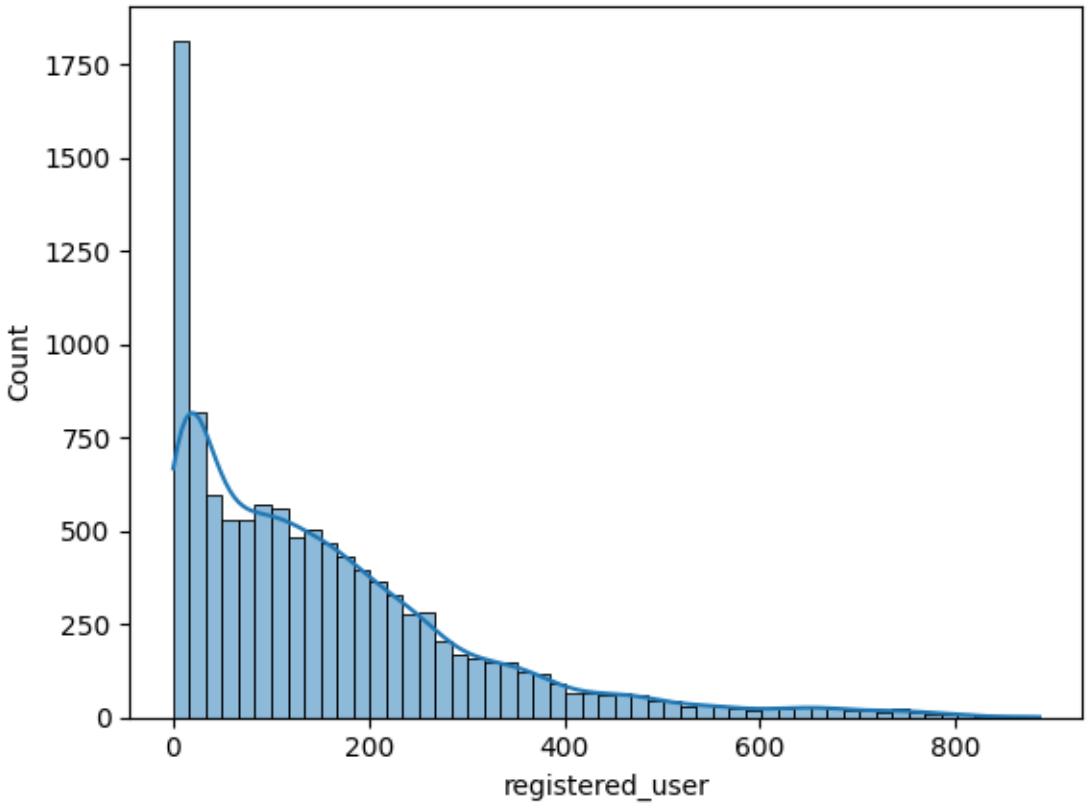
[ ]:
```
sns.boxplot(data['registered_user'])
plt.show()
```



From the above chart, we can see that there is alot of OUTLIERS in 'registered_user' column.

25% to 75% of the data is in range of 36 to 222 .

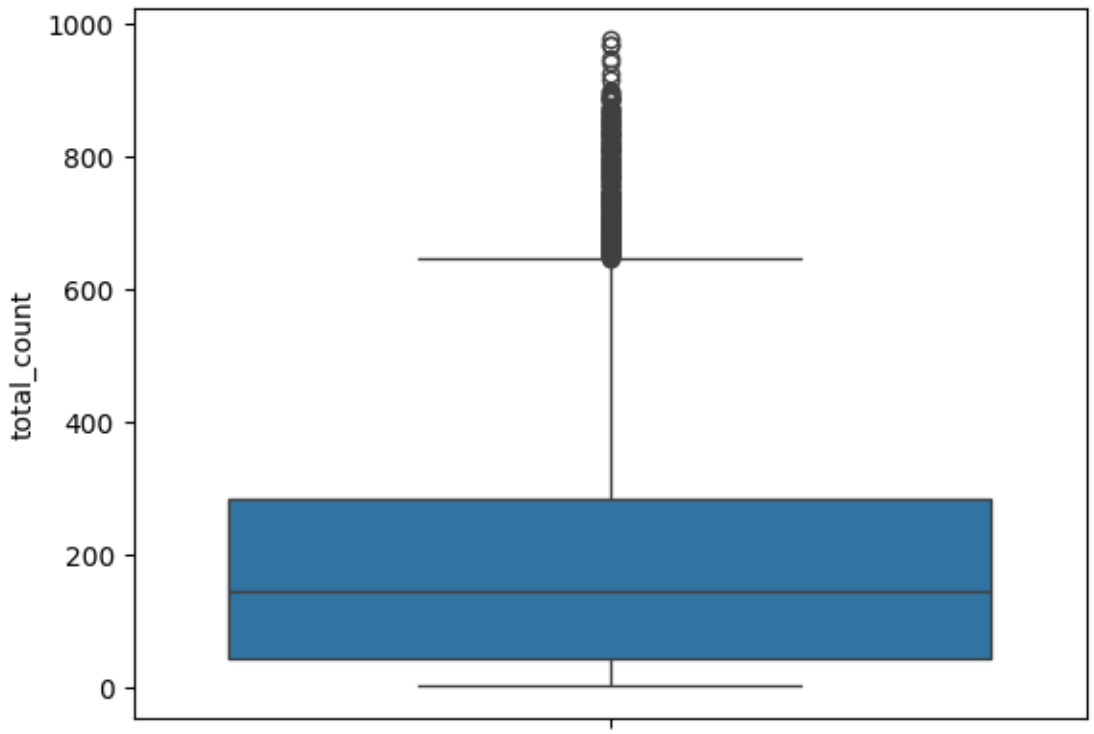
[ ]:
```
# to see the frequency distribution of 'registered_user'
sns.histplot(data['registered_user'], kde = True)
plt.show()
```



[ ]:
```
# to get the mean, standard deviation, max of total_count column
data['total_count'].describe()
```

[ ]:
```
count    10886.000000
mean       191.574132
std        181.144454
min          1.000000
25%         42.000000
50%        145.000000
75%        284.000000
max        977.000000
Name: total_count, dtype: float64
```

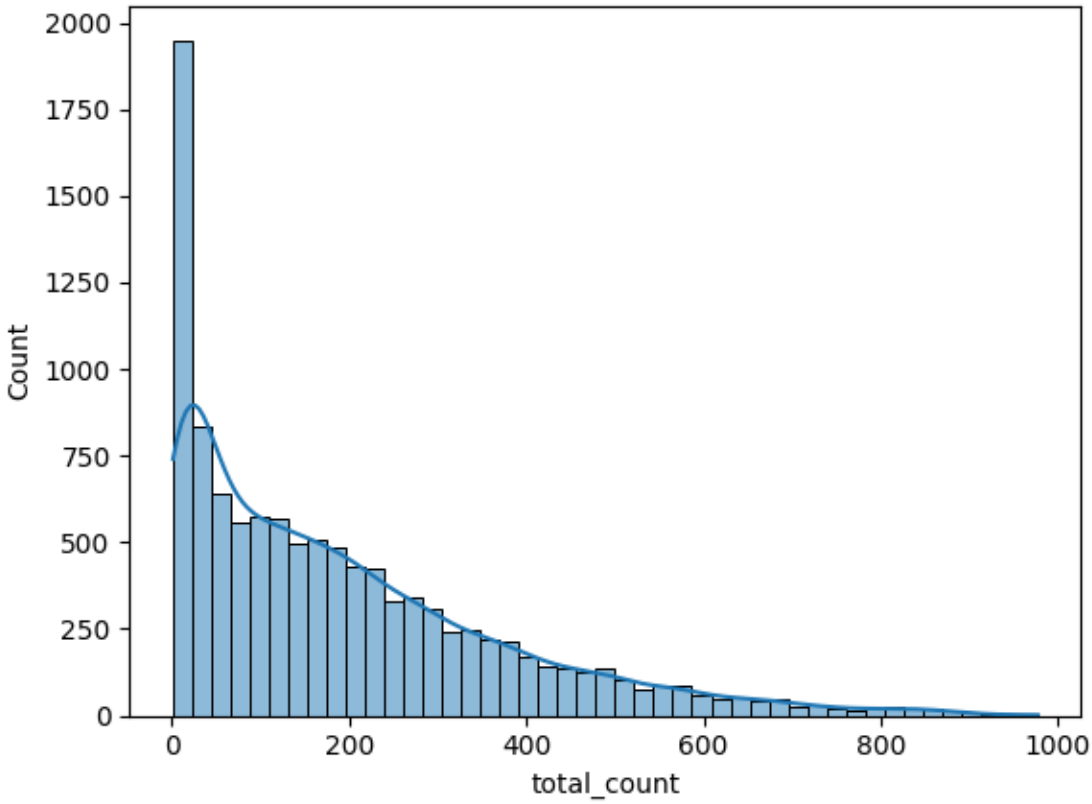
[ ]:
```
sns.boxplot(data['total_count'])
plt.show()
```



From the above chart, we can see that there is alot of OUTLIERS in 'total_count' column.

25% to 75% of the data is in range of 42 to 284 .

[ ]:
```
# to see the frequency distribution of 'total_count'
sns.histplot(data['total_count'], kde = True)
plt.show()
```



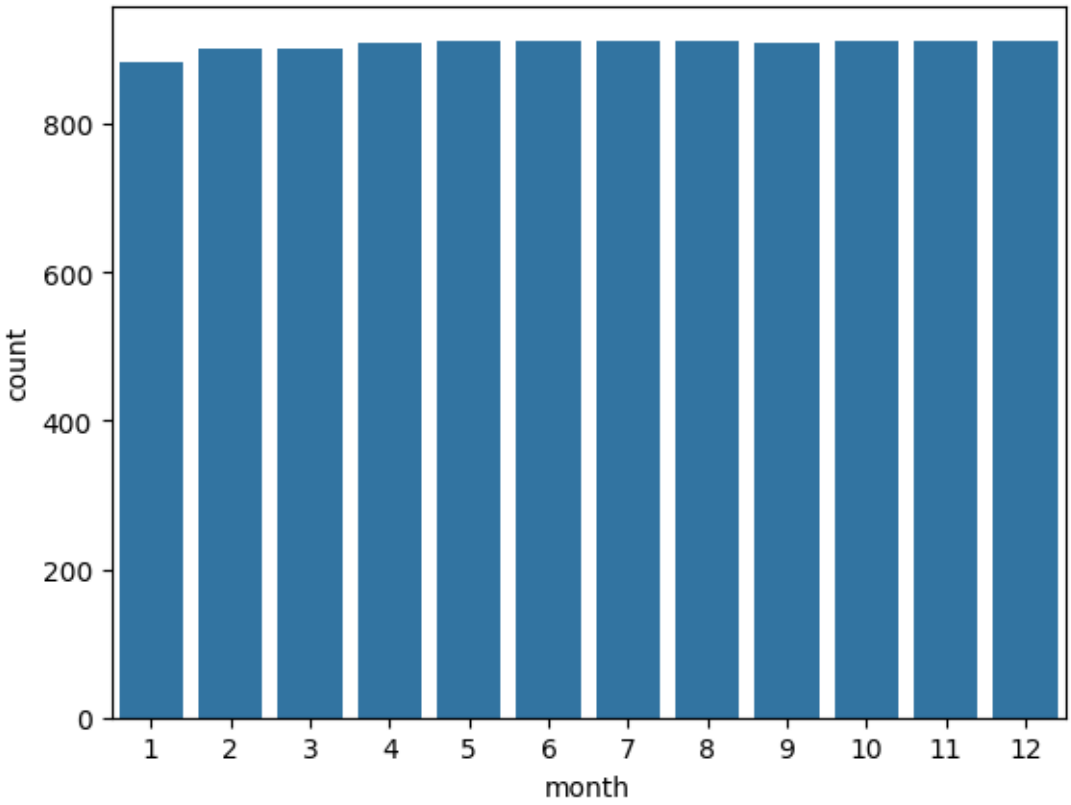
0.2.2 Above plot of 'total_count' of rented cycle is right skewed , not normally distributed

0.2.3 cateorical variable

[ ]:
```
# to see Frequency of the data across different years
d5 = data['year'].value_counts()
d5
```

[ ]:
```
year
2012    5464
2011    5422
Name: count, dtype: int64
```

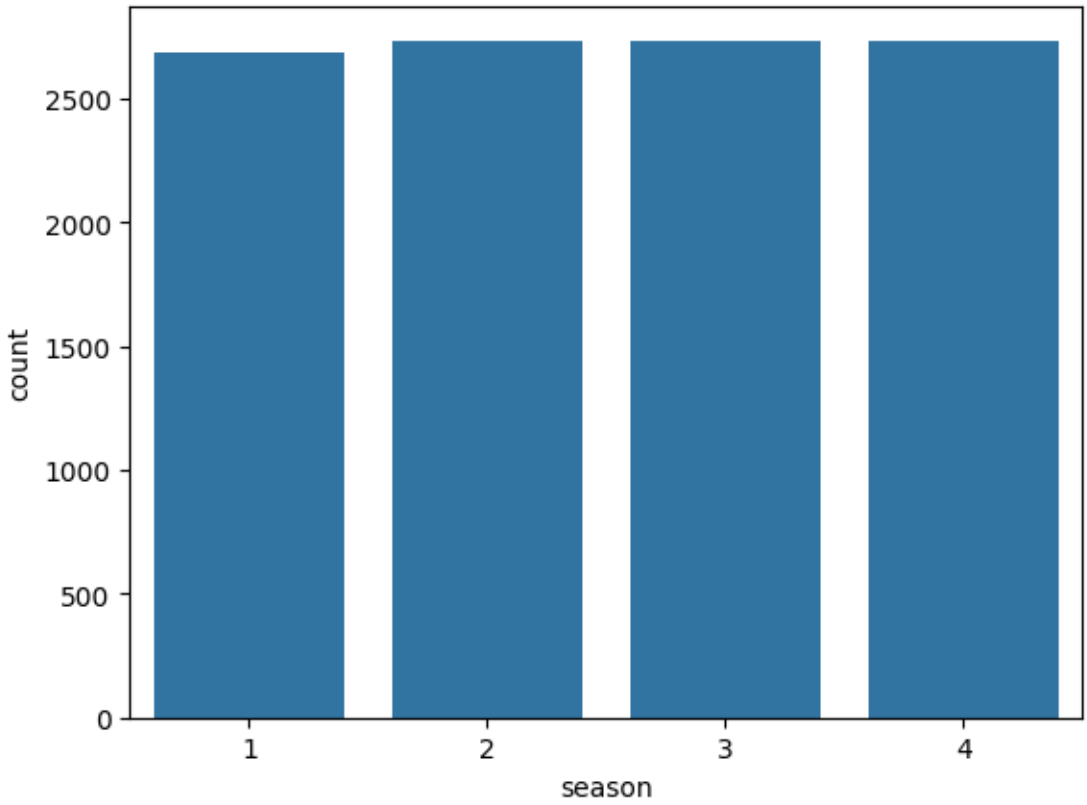
[ ]:
```
sns.barplot(d5)
plt.show()
```



[ ]:
```
# to see Frequency of the data across different month
d6 = data['month'].value_counts()
d6
```

[ ]:
```
month
5     912
6     912
7     912
8     912
12    912
10    911
11    911
4     909
9     909
2     901
3     901
1     884
Name: count, dtype: int64
```

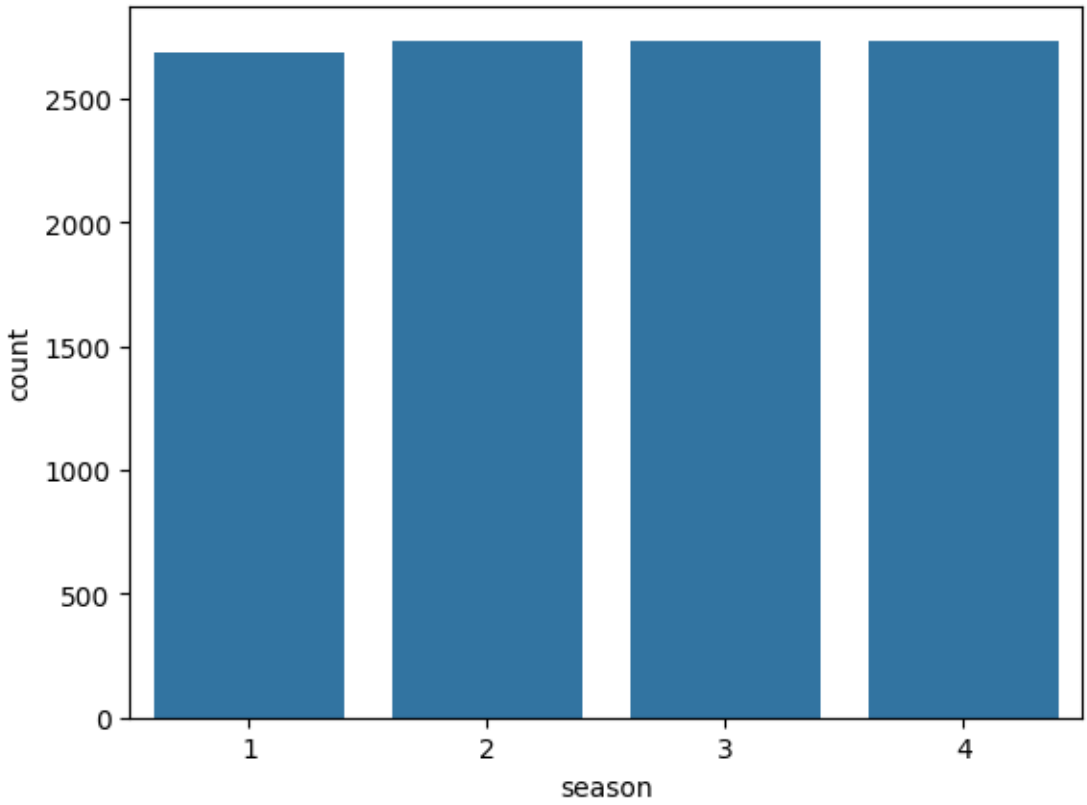
[ ]:
```
sns.barplot(d6)
plt.show()
```



[ ]:
```
# to see Frequency of the data across different season
d7 = data['season'].value_counts()
d7
```

[ ]:
```
season
4    2734
2    2733
3    2733
1    2686
Name: count, dtype: int64
```

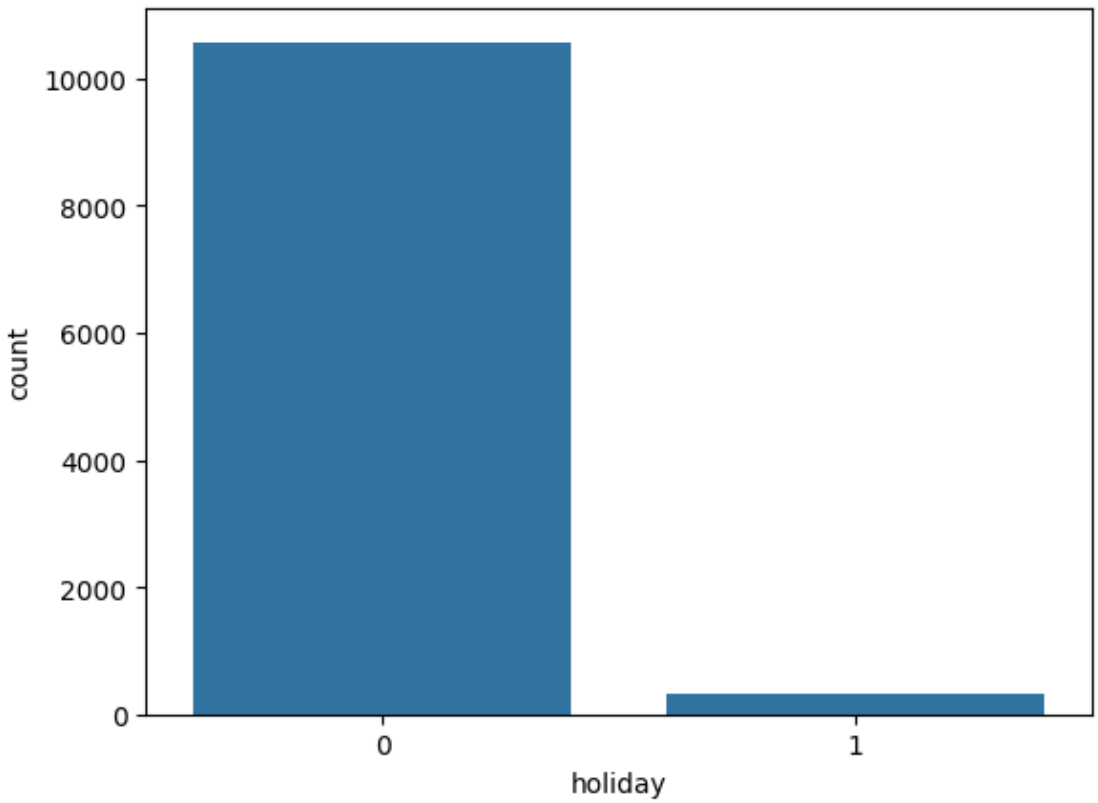
[ ]:
```
sns.barplot(d7)
plt.show()
```



[ ]:
```
# to see Frequency of the data across different holiday
d8 = data['holiday'].value_counts()
d8
```

[ ]:
```
holiday
0    10575
1      311
Name: count, dtype: int64
```

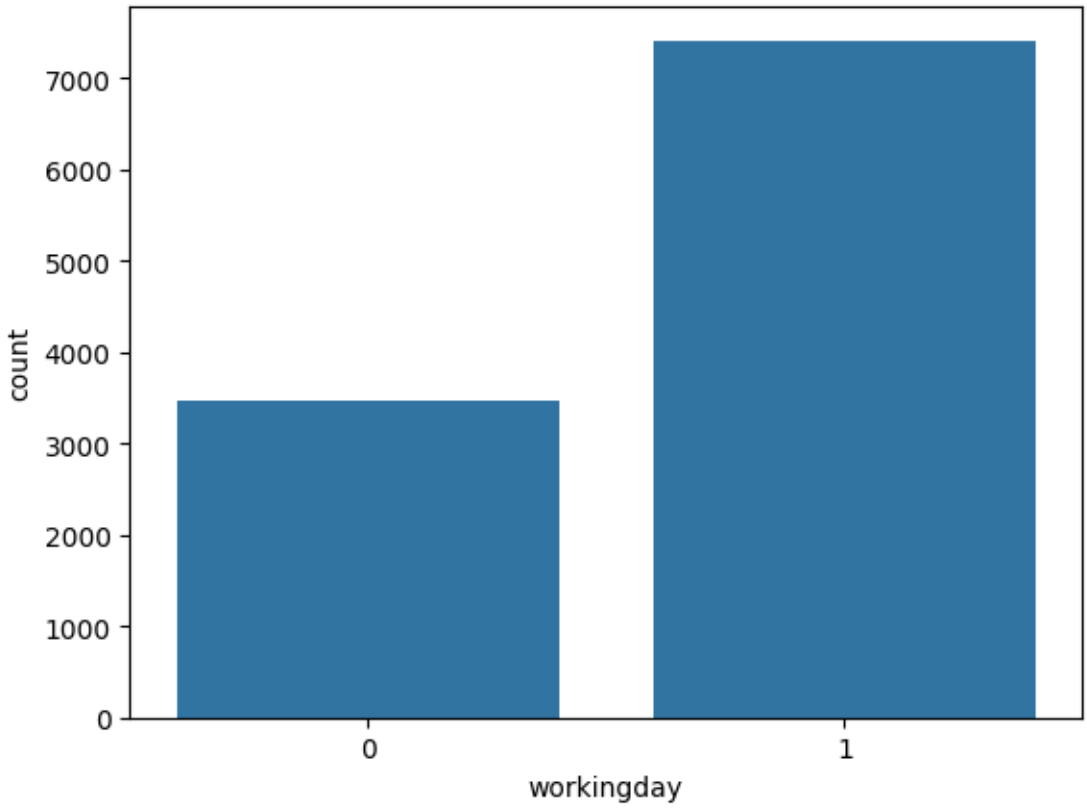
[ ]:
```
sns.barplot(d8)
plt.show()
```



[ ]:
```
# to see Frequency of the data across different workingday
d9 = data['workingday'].value_counts()
d9
```

[ ]:
```
workingday
1    7412
0    3474
Name: count, dtype: int64
```

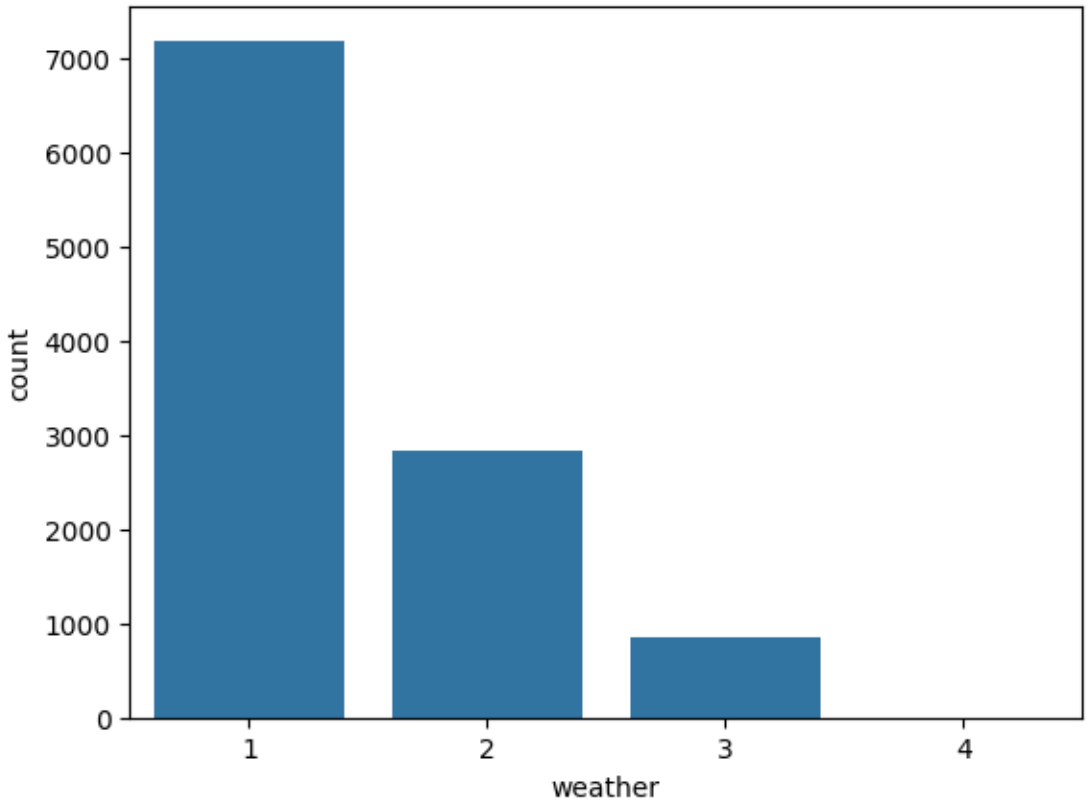
[ ]:
```
sns.barplot(d9)
plt.show()
```



[ ]:
```
# to see Frequency of the data across different weather
d10 = data['weather'].value_counts()
d10
```

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

[ ]:
```
sns.barplot(d10)
plt.show()
```



0.3 Bivariate Analysis in terms of counts of column and data visualization

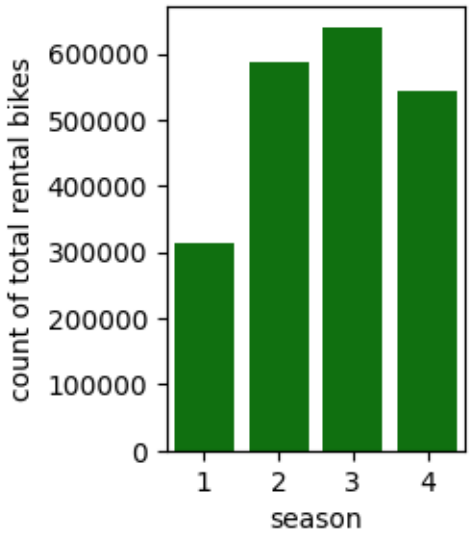
[ ]:
```
data.head(4)
```

[ ]:
```
   season  holiday  workingday  weather  temp   atemp  humidity  windspeed  \
0       1        0           0        1  9.84  14.395        81        0.0
1       1        0           0        1  9.02  13.635        80        0.0
2       1        0           0        1  9.02  13.635        80        0.0
3       1        0           0        1  9.84  14.395        75        0.0

   casual_user  registered_user  total_count        date  year  month  \
0            3               13           16  2011-01-01  2011      1
1            8               32           40  2011-01-01  2011      1
2            5               27           32  2011-01-01  2011      1
3            3               10           13  2011-01-01  2011      1

       time
0  00:00:00
1  01:00:00
2  02:00:00
3  03:00:00
```

[ ]:
```
#  (1: spring, 2: summer, 3: fall, 4: winter)
# We can conseider these 4 columns 'season','holiday','workingday','weather' have the categorical values.
# total count of rental bikes according to the 4 different seasons
d1 = data.groupby('season').agg({'total_count':'sum'})
d1
```

[ ]:
```
        total_count
season
1            312498
2            588282
3            640662
4            544034
```

[ ]:
```
# visualising above table
plt.figure(figsize = (2,3))
sns.barplot(data=d1, x=d1.index, y=d1['total_count'].values, color = 'green')
plt.ylabel('count of total rental bikes')
plt.show()
```



From above plot,
we can see that almost half of the no if bikes got ranted in spring season..

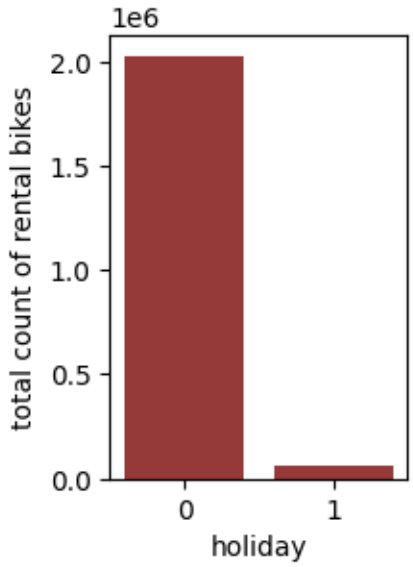
In autumn, people rented maximun no of bikes.
1: spring, 2: summer, 3: fall/ autumn , 4: winter

[ ]:
```
# if day is holiday 1, otherwise is 0
# total count of rental bikes according to the holiday
d2 = data.groupby('holiday').agg({'total_count':'sum'})
```

```
d2
```

[ ]:
```
         total_count
holiday
0            2027668
1              57808
```

[ ]:
```
# visualising above table
plt.figure(figsize = (2,3))
sns.barplot(data = d2, x = d2.index, y = d2['total_count'].values ,color =␣
↪'brown')
plt.ylabel('total count of rental bikes')
plt.yticks(fontsize=10)
plt.show()
```

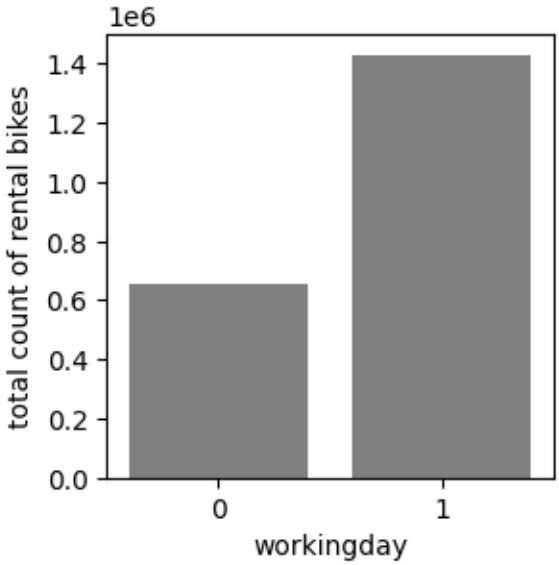


as from above chart,
if day is holiday 1, otherwise is 0 ,
people ranted the bike on other days way more than holidays

[ ]:
```
# if day is neither weekend nor holiday is 1, otherwise is 0
# total count of rental bikes according to the working day
d3 = data.groupby('workingday').agg({'total_count':'sum'})
d3
```

[ ]:
```
            total_count
workingday
0                654872
1               1430604
```

[ ]:
```
# visualising above table
plt.figure(figsize = (3,3))
sns.barplot(data = d3, x = d3.index, y = d3['total_count'].values ,color =␣
 ↪'grey')
plt.ylabel('total count of rental bikes')
plt.show()
```



From above plot,

if day is neither weekend nor holiday is 1, otherwise is 0

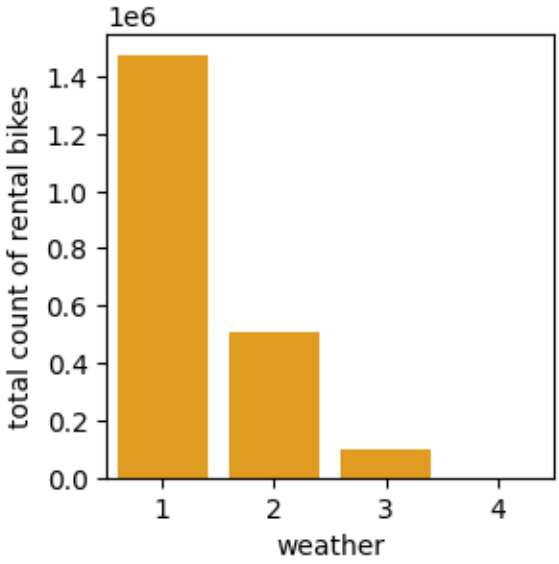
we can see that on working days people rented bikes more.

[ ]:
```
'''
1: Clear, Few clouds, partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain +␣
 ↪Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
'''
d4 = data.groupby('weather').agg({'total_count':'sum'})
d4
```

[ ]:

weather	total_count
1	1476063
2	507160
3	102089
4	164

[ ]:
```
# visualising above table
plt.figure(figsize = (3,3))
sns.barplot(data = d4, x = d4.index, y = d4['total_count'].values ,color =␣
 ↪'orange')
plt.ylabel('total count of rental bikes')
plt.show()
```



from above plot,
we can see that on Clear, Few clouds, partly cloudy day,
people rented maximun no of bikes

[ ]:
```
data.head()
```

[ ]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	\
0	1	0	0	1	9.84	14.395	81	0.0	
1	1	0	0	1	9.02	13.635	80	0.0	
2	1	0	0	1	9.02	13.635	80	0.0	
3	1	0	0	1	9.84	14.395	75	0.0	
4	1	0	0	1	9.84	14.395	75	0.0	

	casual_user	registered_user	total_count	date	year	month	\
0	3	13	16	2011-01-01	2011	1	
1	8	32	40	2011-01-01	2011	1	
2	5	27	32	2011-01-01	2011	1	
3	3	10	13	2011-01-01	2011	1	
4	0	1	1	2011-01-01	2011	1	

	time

```
0  00:00:00
1  01:00:00
2  02:00:00
3  03:00:00
4  04:00:00
```

1 Questions to be Explored Now for Recommendations

1. Working Day has effect on number of electric cycles rented or not?
2. No. of cycles rented similar or different in different seasons?
3. No. of cycles rented similar or different in different weather?
4. Weather is dependent on the season or not?

2 Hypothesis Testing

3 Q1 : Working Day has effect on number of electric cycles rented or not?

[96]:
```
#  if day is neither weekend nor holiday is 1, otherwise is 0.
# getting the mean of total no of cycle rented on working days and not workin␣
 ↪days
data.groupby('workingday')['total_count'].mean()
```

[96]:
```
workingday
0    188.506621
1    193.011873
Name: total_count, dtype: float64
```

Setting the Significance level = 5%
setting the hypothesis:
null hypothesis(H0)
alternate hypothesis(H1)
Option 1:
H0 : Working Day has no effect on number of electric cycles rented. 1= 0
H1 : Working Day has effect on number of electric cycles rented. 1 0

1 : mean no of cycle got rented on working days
0 : mean no of cycle got rented on not working days(holidays and weekends)

[97]:
```
# First, let's store the no of cycle in separate variables.

workingdays = data[data['workingday'] == 1]['total_count']
not_workingdays = data[data['workingday']== 0]['total_count']
```

[98]:
```
# no of cycles got rented on each working day
workingdays
```

[98]:
```
47          5
48          2
49          1
50          3
51         30
          …
10881     336
10882     241
10883     168
10884     129
10885      88
Name: total_count, Length: 7412, dtype: int64
```

[ ]:
```
# no of cycles got rented on each working day
not_workingdays
```

[ ]:
```
0          16
1          40
2          32
3          13
4           1
          …
10809     109
10810     122
10811     106
10812      89
10813      33
Name: total_count, Length: 3474, dtype: int64
```

Performing the 2 sample T-test:

As the population standard deviation is not provided.

[87]:
```
# limporting the Library
from scipy.stats import ttest_ind
```

[99]:
```
t_stats, p_value  = ttest_ind(workingdays,not_workingdays)
t_stats, p_value
```

[99]:
```
(1.2096277376026694, 0.22644804226361348)
```

p-Value is 0.22644804226361348

[101]:
```
alpha = 0.05 # 95% confidence

if p_value < alpha:
  print('Reject H0')
  print(' Working Day has effect on number of electric cycles rented.')
```

```
else:
  print ('Fail to Reject H0')
  print('mean of total no of cycle rented on working day and on not working day␣
 ↪is same')
  print('WorkingDays has no effect on number of electric cycles rented')
```

```
Fail to Reject H0
mean of total no of cycle rented on working day and on not working day is same
WorkingDays has no effect on number of electric cycles rented
```

4 Q2 : No. of cycles rented similar or different in different seasons?

[ ]:
```
# get the data again
data.head()
```

[ ]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	\
0	1	0	0	1	9.84	14.395	81	0.0	
1	1	0	0	1	9.02	13.635	80	0.0	
2	1	0	0	1	9.02	13.635	80	0.0	
3	1	0	0	1	9.84	14.395	75	0.0	
4	1	0	0	1	9.84	14.395	75	0.0	

	casual_user	registered_user	total_count	date	year	month	\
0	3	13	16	2011-01-01	2011	1	
1	8	32	40	2011-01-01	2011	1	
2	5	27	32	2011-01-01	2011	1	
3	3	10	13	2011-01-01	2011	1	
4	0	1	1	2011-01-01	2011	1	

	time
0	00:00:00
1	01:00:00
2	02:00:00
3	03:00:00
4	04:00:00

[ ]:
```
# to get the unique seasons
# season (1: spring, 2: summer, 3: fall, 4: winter)
data['season'].unique()
```

[ ]:
```
array([1, 2, 3, 4])
```

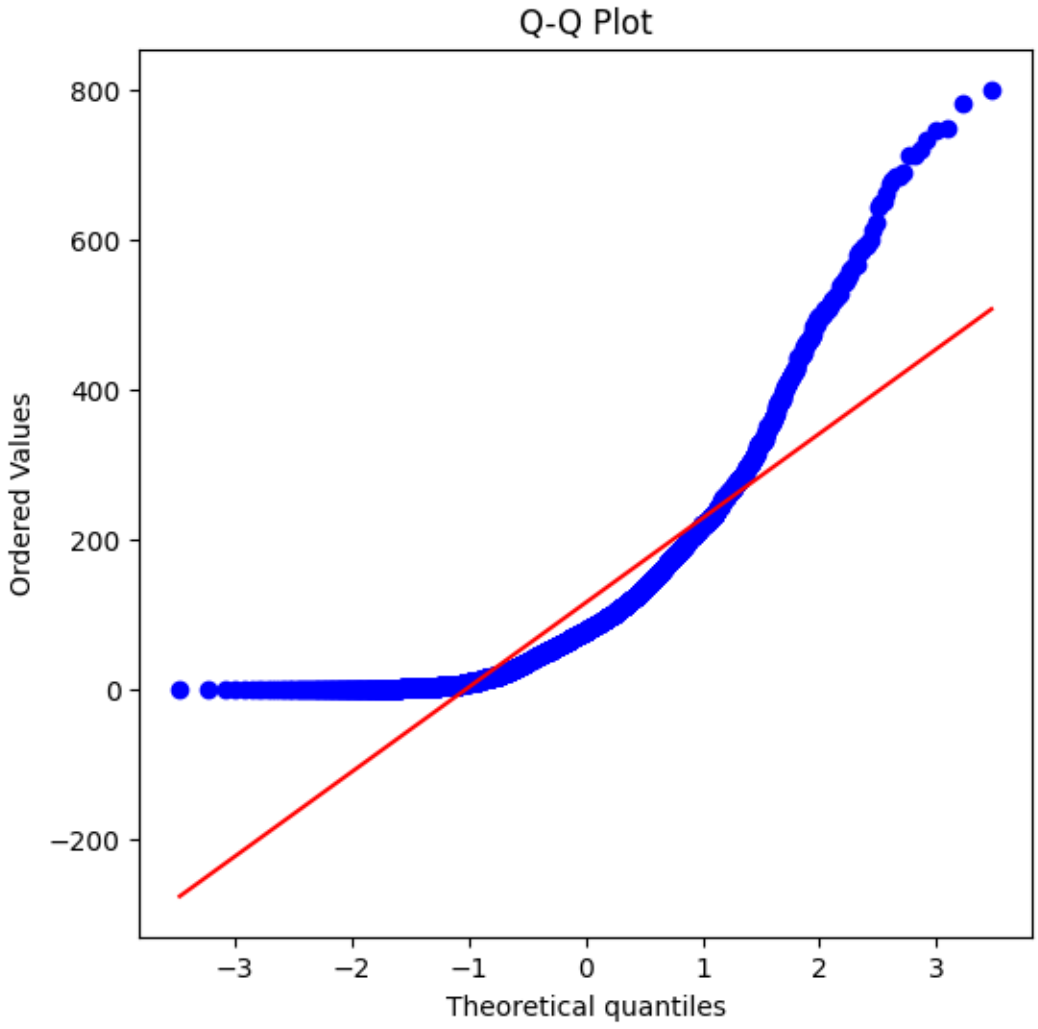
[ ]:
```
# Import the necessary modules
import scipy.stats as stats
import matplotlib.pyplot as plt
```

```
# Filter the data based on season
spring_count = data[data['season']==1]['total_count']
summer_count = data[data['season']==2]['total_count']
fall_count = data[data['season']==3]['total_count']
winter_count = data[data['season']==4]['total_count']
```

4.1 checking the assumptions for anova testing

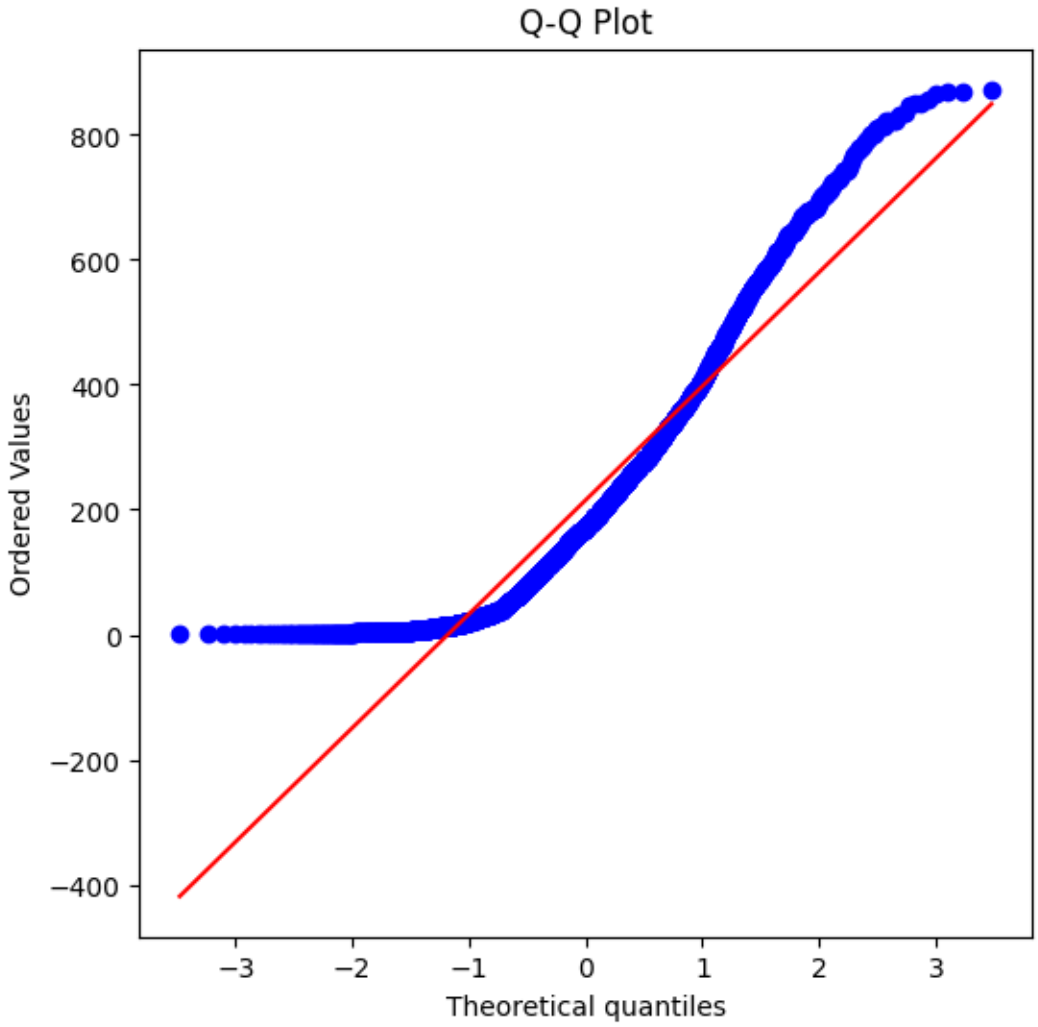
4.1.1 Q-Q *plot*

[ ]:
```
# Create a Q-Q plot for the spring season data
plt.figure(figsize=(6, 6))
stats.probplot(spring_count, dist="norm", plot=plt)
plt.title('Q-Q Plot')
plt.show()
```



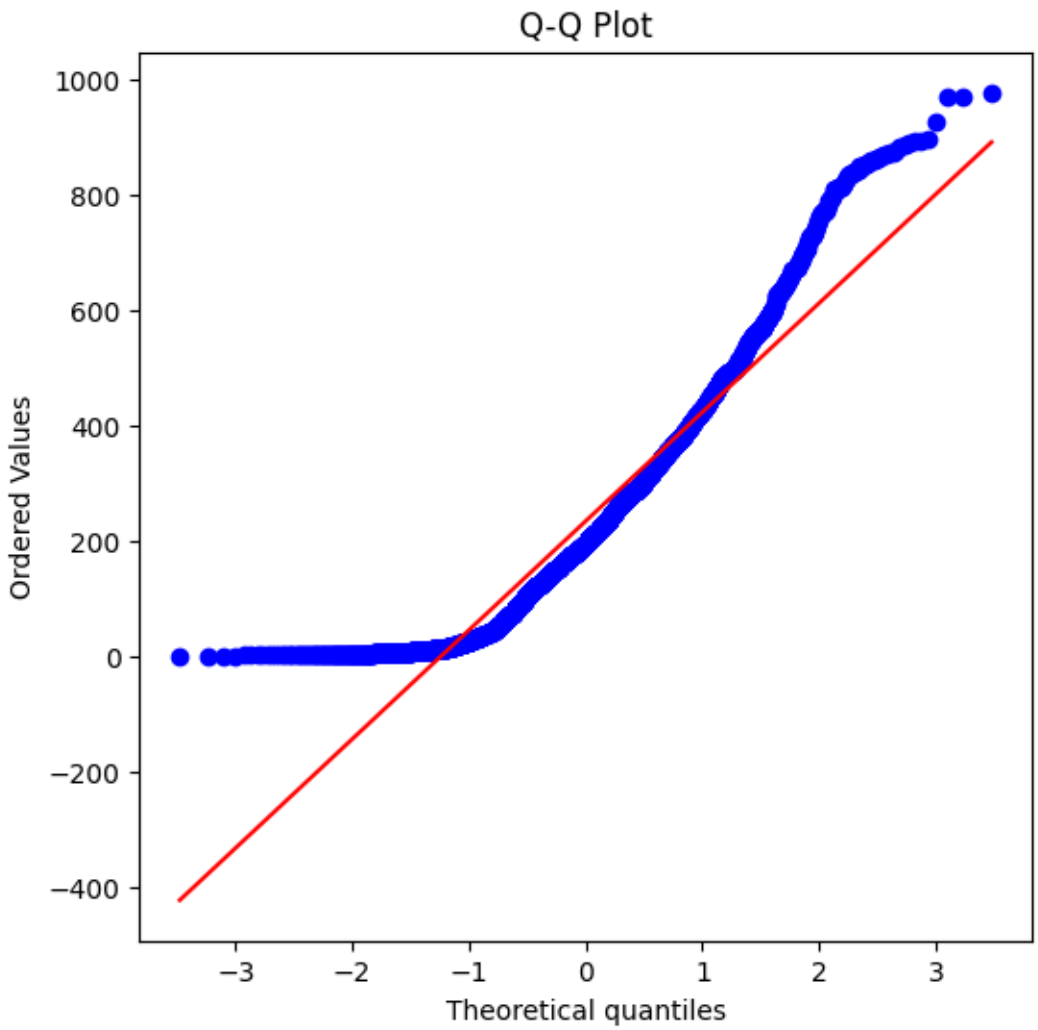
from the above plot, we can see that the data is not normal distribution.

[ ]:
```
# Create a Q-Q plot for the summer season data
plt.figure(figsize=(6, 6))
stats.probplot(summer_count, dist="norm", plot=plt)
plt.title('Q-Q Plot')
plt.show()
```



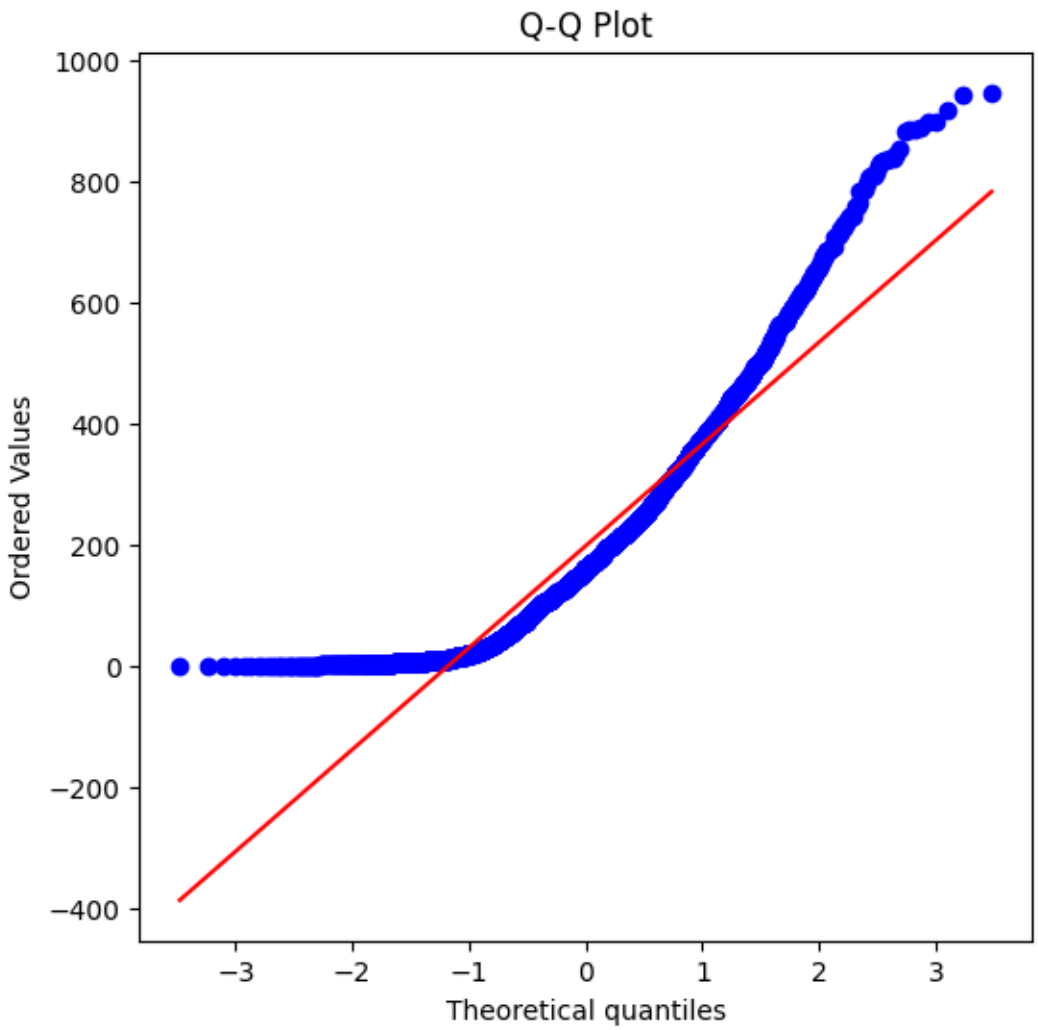
from the above plot, we can see that the data is not normal distribution.

[ ]:
```
# Create a Q-Q plot for the fall season data
plt.figure(figsize=(6, 6))
stats.probplot(fall_count, dist="norm", plot=plt)
plt.title('Q-Q Plot')
plt.show()
```



from the above plot, we can see that the data is not normal distribution.

[ ]:
```
# Create a Q-Q plot for the fall season data
plt.figure(figsize=(6, 6))
stats.probplot(winter_count, dist="norm", plot=plt)
plt.title('Q-Q Plot')
plt.show()
```



from the above plot, we can see that the data is not normal distribution.

4.1.2 Shapiro-Wilk test to check normality satistically

[ ]:
```
# H0: data is normal distribution
# Ha: data is not normal distribution

# Perform the Shapiro-Wilk test for spring_count
stat, p_value = stats.shapiro(spring_count)

print(f"Shapiro-Wilk Statistic of spring_count: {stat}")
print(f"P-value of spring_count : {p_value}")
print('\n')
```

```
# Perform the Shapiro-Wilk test for summer_count
stat, p_value = stats.shapiro(summer_count)

print(f"Shapiro-Wilk Statistic of summer_count: {stat}")
print(f"P-value of summer_count : {p_value}")
print('\n')


# Perform the Shapiro-Wilk test for fall_count
stat, p_value = stats.shapiro(fall_count)

print(f"Shapiro-Wilk Statistic of fall_count: {stat}")
print(f"P-value of fall_count : {p_value}")
print('\n')


# Perform the Shapiro-Wilk test for winter_count
stat, p_value = stats.shapiro(winter_count)

print(f"Shapiro-Wilk Statistic of winter_count: {stat}")
print(f"P-value of winter_count : {p_value}")
print('\n')
```

```
Shapiro-Wilk Statistic of spring_count: 0.8087388873100281
P-value of spring_count : 0.0


Shapiro-Wilk Statistic of summer_count: 0.900481641292572
P-value of summer_count : 6.039093315091269e-39


Shapiro-Wilk Statistic of fall_count: 0.9148160815238953
P-value of fall_count : 1.043458045587339e-36


Shapiro-Wilk Statistic of winter_count: 0.8954644799232483
P-value of winter_count : 1.1301682309549298e-39
```

as, the significant level = 0.05 all the 4 groups p-values are less than 0.05. so , statistically we can see that the groups are not nomal distribution

5 Perform the Levene test

to test equality of varience of 4 groups

[ ]:
```
# H0 : groups have the equal varience
# Ha: groups varience are not equal
# Perform the Levene test
stat, p_value = stats.levene(spring_count, summer_count, fall_count,␣
 ↪winter_count)

print(f"Levene Statistic: {stat}")
print(f"P-value: {p_value}")

alpha = 0.05 # 95% confidence
print('\n')
if p_value < alpha:
  print('Reject H0')
  print(' groups varience are not equal.')
else:
  print ('Fail to Reject H0')
  print('groups have the equal varience')
```

```
Levene Statistic: 187.7706624026276
P-value: 1.0147116860043298e-118


Reject H0
 groups varience are not equal.
```

Hence, we won't able to perform ANOVA test because of all anova asumptions have failes

5.1 We will use kruskal-wallis test

[ ]:
```
# importing necessry library
from scipy.stats import kruskal
```

[ ]:
```
# H0: no of cycles rented is similar in different seasons
# Ha: no of cycles rented is different alleast in one different season
# significance level = 0.05

stat, p_value = kruskal(spring_count, summer_count, fall_count, winter_count)

print("test statistic:",stat)
print("p_value:",p_value)
print('\n')

if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different median")
    print('no of cycles rented is different alleast in one different season')
else:
    print("Fail to reject H0")
```

```
    print("All groups have same median")
    print('no of cycles rented is similar in different seasons')
```

```
test statistic: 699.6668548181988
p_value: 2.479008372608633e-151


Reject H0
Atleast one group have different median
no of cycles rented is different alleast in one different season
```

6 Q3. No. of cycles rented similar or different in different weather?

[ ]:
```
# get the data again
data.head(3)
```

[ ]:
```
   season  holiday  workingday  weather  temp   atemp  humidity  windspeed  \
0       1        0           0        1  9.84  14.395        81        0.0
1       1        0           0        1  9.02  13.635        80        0.0
2       1        0           0        1  9.02  13.635        80        0.0

   casual_user  registered_user  total_count        date  year  month  \
0            3               13           16  2011-01-01  2011      1
1            8               32           40  2011-01-01  2011      1
2            5               27           32  2011-01-01  2011      1

       time
0  00:00:00
1  01:00:00
2  02:00:00
```

[ ]:
```
'''
weather:
1: Clear, Few clouds, partly cloudy,
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain +␣
 ↪Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

'''
# get the unique weathers
data['weather'].unique()
```

[ ]:
```
array([1, 2, 3, 4])
```

[ ]:
```
# Import the necessary modules
import scipy.stats as stats
import matplotlib.pyplot as plt

# Filter the data based on weather
clear = data[data['weather']==1]['total_count']
cloudy = data[data['weather']==2]['total_count']
rain = data[data['weather']==3]['total_count']
heavy_Rain = data[data['weather']==4]['total_count']
```

[ ]:
```
# heavy rain has only one data point
data[data['weather']==4]
```

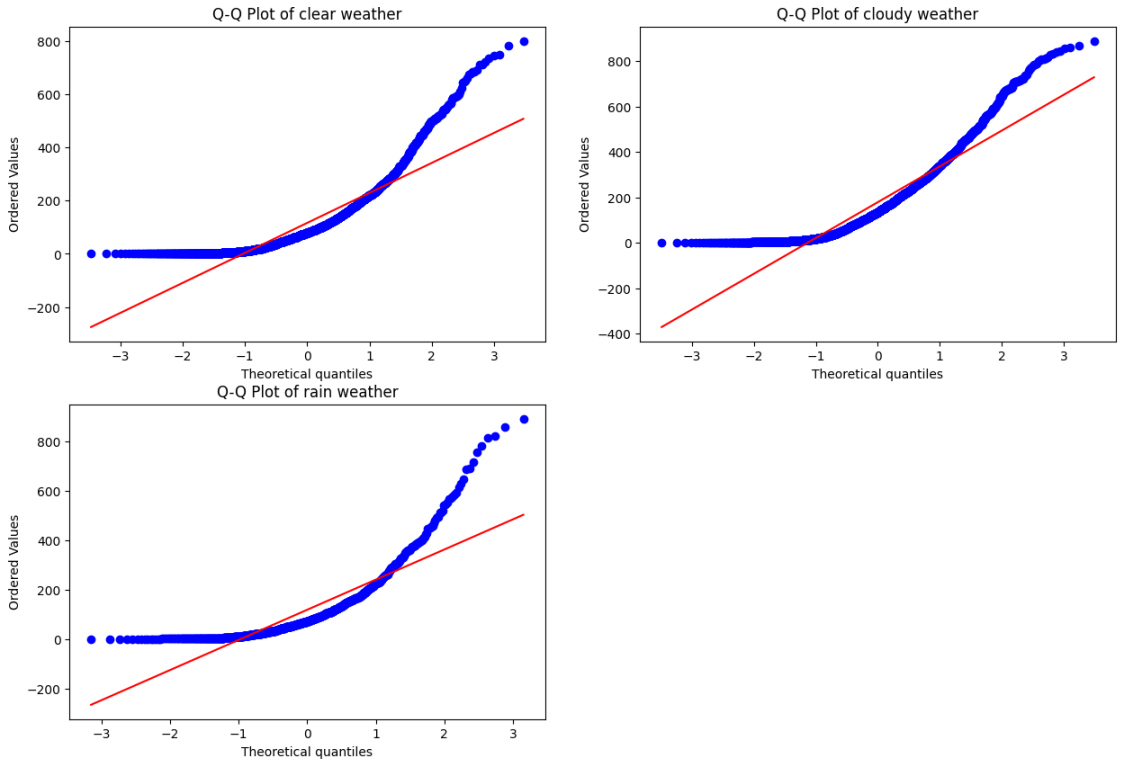
[ ]:
```
      season  holiday  workingday  weather  temp   atemp  humidity  windspeed  \
5631       1        0           1        4   8.2  11.365        86     6.0032

      casual_user  registered_user  total_count        date  year  month  \
5631            6              158          164  2012-01-09  2012      1

          time
5631  18:00:00
```

6.1 checking the assumptions for anova testing

6.1.1 Q-Q *plot*

[ ]:
```
fig = plt.figure(figsize=(15,10))

plt.subplot(2,2, 1)
# Create a Q-Q plot for the clear weather data
stats.probplot(spring_count, dist="norm", plot=plt)
plt.title('Q-Q Plot of clear weather')
plt.subplot(2,2, 2)
# Create a Q-Q plot for the cloudy weather data
stats.probplot(cloudy, dist="norm", plot=plt)
plt.title('Q-Q Plot of cloudy weather')
plt.subplot(2,2, 3)
# Create a Q-Q plot for the rain weather data
stats.probplot(rain, dist="norm", plot=plt)
plt.title('Q-Q Plot of rain weather')
#plt.subplot(2,2, 4)
# Create a Q-Q plot for the heavy_rain weather data
#stats.probplot(heavy_Rain, dist="norm", plot=plt)
#plt.title('Q-Q Plot of heavy_Rain weather')
```

[ ]:
```
Text(0.5, 1.0, 'Q-Q Plot of rain weather')
```



from the above plot we get to see that groups are not normally distributed.

6.2 Shapiro-Wilk test to check normality satistically

[ ]:
```
# H0: data is normal distribution
# H1: data is not normal distribution

# Perform the Shapiro-Wilk test for clear weather
stat, p_value = stats.shapiro(clear)

print(f"Shapiro-Wilk Statistic of clear weather: {stat}")
print(f"P-value of clear weather : {p_value}")
print('\n')


# Perform the Shapiro-Wilk test for cloudy weather
stat, p_value = stats.shapiro(cloudy)

print(f"Shapiro-Wilk Statistic of cloudy weather: {stat}")
print(f"P-value of cloudy weather : {p_value}")
print('\n')
```

```
# Perform the Shapiro-Wilk test for rain weather
stat, p_value = stats.shapiro(rain)

print(f"Shapiro-Wilk Statistic of rain weather: {stat}")
print(f"P-value of rain weather : {p_value}")
print('\n')


# Perform the Shapiro-Wilk test for heavy_Rain weather
# stat, p_value = stats.shapiro(heavy_Rain)

#print(f"Shapiro-Wilk Statistic of heavy_Rain weather: {stat}")
#print(f"P-value of heavy_Rain weather : {p_value}")
#print('\n')
```

```
Shapiro-Wilk Statistic of clear weather: 0.8909230828285217
P-value of clear weather : 0.0


Shapiro-Wilk Statistic of cloudy weather: 0.8767687082290649
P-value of cloudy weather : 9.781063280987223e-43


Shapiro-Wilk Statistic of rain weather: 0.7674332857131958
P-value of rain weather : 3.876090133422781e-33
```

P_values are lower than significance level 0.05.
hence, statistially checked data in not normal distribution.

7 Perform the Levene test

to test equality of varience of 3 weather groups

[ ]:
```
# H0 : groups have the equal varience
# H1: groups varience are not equal
# Perform the Levene test
stat, p_value = stats.levene(clear, cloudy, rain)

print(f"Levene Statistic: {stat}")
print(f"P-value: {p_value}")

alpha = 0.05 # 95% confidence
print('\n')
if p_value < alpha:
  print('Reject H0')
```

```
  print(' groups varience are not equal.')
else:
  print ('Fail to Reject H0')
  print('groups have the equal varience')
```

```
Levene Statistic: 81.67574924435011
P-value: 6.198278710731511e-36


Reject H0
 groups varience are not equal.
```

Hence, we won't able to perform ANOVA test because of all anova asumptions have failes

7.1 We will use kruskal-wallis test

[ ]:
```
# H0: no of cycles rented is similar in different weather
# H1: no of cycles rented is different alleast in one different weather
# significance level = 0.05

stat, p_value = kruskal(clear, cloudy, rain, heavy_Rain)

print("test statistic:",stat)
print("p_value:",p_value)
print('\n')

if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different median")
    print('no of cycles rented is different alleast in one different weather')
else:
    print("Fail to reject H0")
    print("All groups have same median")
    print('no of cycles rented is similar in different weather')
```

```
test statistic: 205.00216514479087
p_value: 3.501611300708679e-44


Reject H0
Atleast one group have different median
no of cycles rented is different alleast in one different weather
```

8 Q4: Weather is dependent on season or not?

[79]:
```
# importing ilbraries
from scipy.stats import chisquare # Statistical test (chistat, pvalue)
from scipy.stats import chi2
from scipy.stats import chi2_contingency
```

[81]:
```
season_weather = pd.crosstab(index=data['weather'],columns=data['season'])
season_weather # This will give the count of each season for each weather
```

[81]:

season	1	2	3	4
weather				
1	1759	1801	1930	1702
2	715	708	604	807
3	211	224	199	225
4	1	0	0	0

[84]:
```
# H0: weather and season are independent
# H1: weather and season are not independent
chi_stat, p_value, df, exp_freq = chi2_contingency(season_weather) # chi_stat,␣
 ↪p_value, df, expected value

print("chi_stat:",chi_stat)
print("p_value:",p_value)
print("df:",df)
print("exp_freq:",exp_freq)
print('\n')
if p_value < 0.05:
    print("Reject H0")
    print('weather and season are not independent')
else:
    print("Fail to reject H0")
    print('weather and season are independent')
```

```
chi_stat: 49.15865559689363
p_value: 1.5499250736864862e-07
df: 9
exp_freq: [[1.77454639e+03 1.80559765e+03 1.80559765e+03 1.80625831e+03]
 [6.99258130e+02 7.11493845e+02 7.11493845e+02 7.11754180e+02]
 [2.11948742e+02 2.15657450e+02 2.15657450e+02 2.15736359e+02]
 [2.46738931e-01 2.51056403e-01 2.51056403e-01 2.51148264e-01]]


Reject H0
weather and season are not independent
```

9 Inference

1. WorkingDays has no effect on number of electric cycles rented.
2. no of cycles rented is different alleast in one different seasons.
3. no of cycles rented is different alleast in one different weathers.
4. weather and season are not independent.