yulu-casestudy

October 5, 2024

1 Yulu Case study

Importing the required libraries for the dataset analysis

[188]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

''' creating a df from the csv file and parsing the data colum i.e changing the␣
↪datatype from object to 'datetime' '''

df = pd.read_csv('yulu.csv',parse_dates= [1],dayfirst = True,na_values = 'NA')
df
```

[188]:
```
                 datetime  season  holiday  workingday  weather   temp  \
0     2011-01-01 00:00:00       1        0           0        1   9.84
1     2011-01-01 01:00:00       1        0           0        1   9.02
2     2011-01-01 02:00:00       1        0           0        1   9.02
3     2011-01-01 03:00:00       1        0           0        1   9.84
4     2011-01-01 04:00:00       1        0           0        1   9.84
…                     …     …       …           …        …      …
10881 2012-12-19 19:00:00       4        0           1        1  15.58
10882 2012-12-19 20:00:00       4        0           1        1  14.76
10883 2012-12-19 21:00:00       4        0           1        1  13.94
10884 2012-12-19 22:00:00       4        0           1        1  13.94
10885 2012-12-19 23:00:00       4        0           1        1  13.12

        atemp  humidity  windspeed  casual  registered  count
0      14.395        81     0.0000       3          13     16
1      13.635        80     0.0000       8          32     40
2      13.635        80     0.0000       5          27     32
3      14.395        75     0.0000       3          10     13
4      14.395        75     0.0000       0           1      1
…         …         …        …       …         …
10881  19.695        50    26.0027       7         329    336
10882  17.425        57    15.0013      10         231    241
10883  15.910        61    15.0013       4         164    168
```

```
10884  17.425        61     6.0032       12          117     129
10885  16.665        66     8.9981        4           84      88

[10886 rows x 12 columns]
```

[189]:
```python
''' checking the size of the dataframe '''

df.shape
```

[189]:
```
(10886, 12)
```

[190]:
```python
''' information regarding the columns and their datatypes '''

df.info()
```

```
<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  object
 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(7), object(2)
memory usage: 1020.7+ KB
```

[191]:
```python
''' checking for the null values acros the columns '''

df.isnull().sum()
```

[191]:
```
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
```

[192]:
```
'''' getiing statistical information of the df '''

df.describe()
```

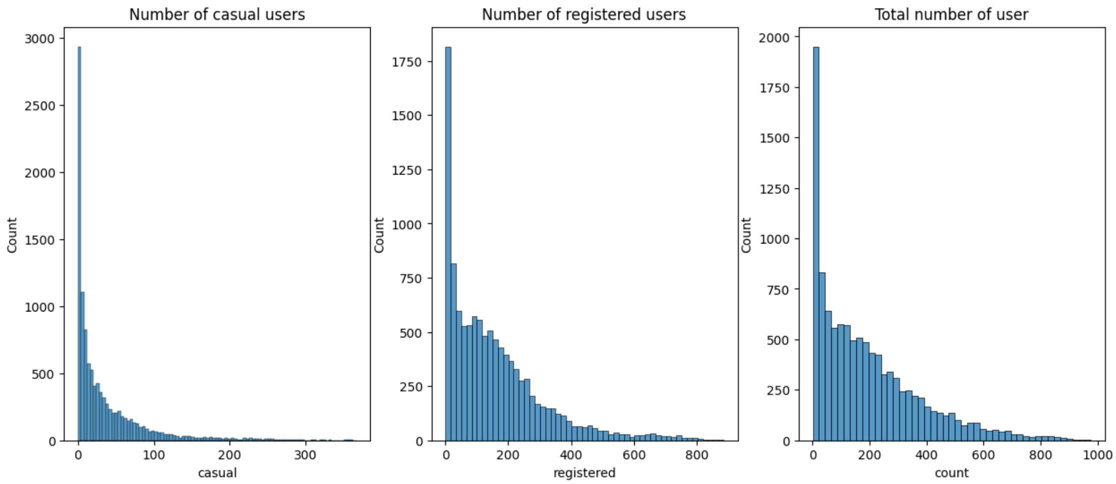
[192]:
```
            holiday    workingday       weather         temp         atemp  \
count  10886.000000  10886.000000  10886.000000  10886.00000  10886.000000
mean       0.028569      0.680875      1.418427     20.23086     23.655084
std        0.166599      0.466159      0.633839      7.79159      8.474601
min        0.000000      0.000000      1.000000      0.82000      0.760000
25%        0.000000      0.000000      1.000000     13.94000     16.665000
50%        0.000000      1.000000      1.000000     20.50000     24.240000
75%        0.000000      1.000000      2.000000     26.24000     31.060000
max        1.000000      1.000000      4.000000     41.00000     45.455000

           humidity     windspeed        casual    registered         count
count  10886.000000  10886.000000  10886.000000  10886.000000  10886.000000
mean      61.886460     12.799395     36.021955    155.552177    191.574132
std       19.245033      8.164537     49.960477    151.039033    181.144454
min        0.000000      0.000000      0.000000      0.000000      1.000000
25%       47.000000      7.001500      4.000000     36.000000     42.000000
50%       62.000000     12.998000     17.000000    118.000000    145.000000
75%       77.000000     16.997900     49.000000    222.000000    284.000000
max      100.000000     56.996900    367.000000    886.000000    977.000000
```

[193]:
```
''' checking for the total duplicates in the df '''

df.duplicated().sum()
```

[193]: 0

[194]:
```
''' Histogram for  the casual,registered users along with total number of users␣
 ↪'''
plt.figure(figsize = (15,6))
plt.subplot(1,3,1)
plt.title('Number of casual users')
sns.histplot(df['casual'])
plt.subplot(1,3,2)
plt.title('Number of registered users')
sns.histplot(df['registered'])
plt.subplot(1,3,3)
plt.title('Total number of user')
```

```
sns.histplot(df['count'])
plt.show()
```



[195]:
```
''' filtering the top 5 rows odf the df '''

df.head()
```

[195]:
```
              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
```

[196]:
```
''' creating new colums Day by merging the two columns holiday and woringday 1␣
↪and 0 by replacing 'holiday' and 'workingday' '''

df['day'] = df['workingday'].apply(lambda x: 'workingday' if x == 1 else(
                                                                         ␣
↪'holiday'))
```

[197]:
```
'Encoding the numericl values in the weather colum to appropriate weather␣
↪condition in new column is_weather '''
```

```
df['is_weather'] = df['weather'].apply(lambda x : 'Clear or partly cloudy' if x␣
↪== 1 else(
                                                                                ␣
↪          'Mist or Few cloud' if x == 2 else(
                                                                                ␣
↪                                           'Light Rain' if x == 3 else(
                                                                                ␣
↪                                                                              ␣
↪'Heavy Rain or Thunderstorm' ))))
```

[198]:
```
''' checking the newly created is_weather column '''

df[df['weather'] > 1]
```

[198]:
```
                 datetime  season  holiday  workingday  weather   temp  \
5     2011-01-01 05:00:00       1        0           0        2   9.84
13    2011-01-01 13:00:00       1        0           0        2  18.86
14    2011-01-01 14:00:00       1        0           0        2  18.86
15    2011-01-01 15:00:00       1        0           0        2  18.04
16    2011-01-01 16:00:00       1        0           0        2  17.22
…                         …     …       …           …        …     …
10837 2012-12-17 23:00:00       4        0           1        3  17.22
10838 2012-12-18 00:00:00       4        0           1        2  18.04
10839 2012-12-18 01:00:00       4        0           1        2  18.04
10840 2012-12-18 02:00:00       4        0           1        2  18.04
10850 2012-12-18 12:00:00       4        0           1        3  19.68

        atemp  humidity  windspeed  casual  registered  count         day  \
5      12.880        75     6.0032       0           1      1     holiday
13     22.725        72    19.9995      47          47     94     holiday
14     22.725        72    19.0012      35          71    106     holiday
15     21.970        77    19.9995      40          70    110     holiday
16     21.210        82    19.9995      41          52     93     holiday
…           …         …         …       …           …      …           …
10837  21.210        94    15.0013       6          41     47  workingday
10838  21.970        94     8.9981       0          18     18  workingday
10839  21.970        94     8.9981       0          15     15  workingday
10840  21.970        88    15.0013       2           5      7  workingday
10850  23.485        48    16.9979      49         264    313  workingday

               is_weather
5       Mist or Few cloud
13      Mist or Few cloud
14      Mist or Few cloud
15      Mist or Few cloud
16      Mist or Few cloud
```

```
…                    …
10837         Light Rain
10838  Mist or Few cloud
10839  Mist or Few cloud
10840  Mist or Few cloud
10850         Light Rain

[3694 rows x 14 columns]
```

[199]:

```
''' creating the new column is_season by given label encoding in the data of␣
↪season column '''

def season_fun(x):
  res = ''
  if x == '1':
    res = 'spring'
  elif x == '2':
    res = 'summer'
  elif x == '3':
    res = 'fall'
  elif x == '4':
    res = 'winter'
  return res


df['is_season'] = df['season'].apply(season_fun)
```

[200]:

```
''' dropping the un necessary columns '''

df.drop(['holiday','workingday','weather','season'],axis = 1,inplace = True)
```

[201]:

```
df
```

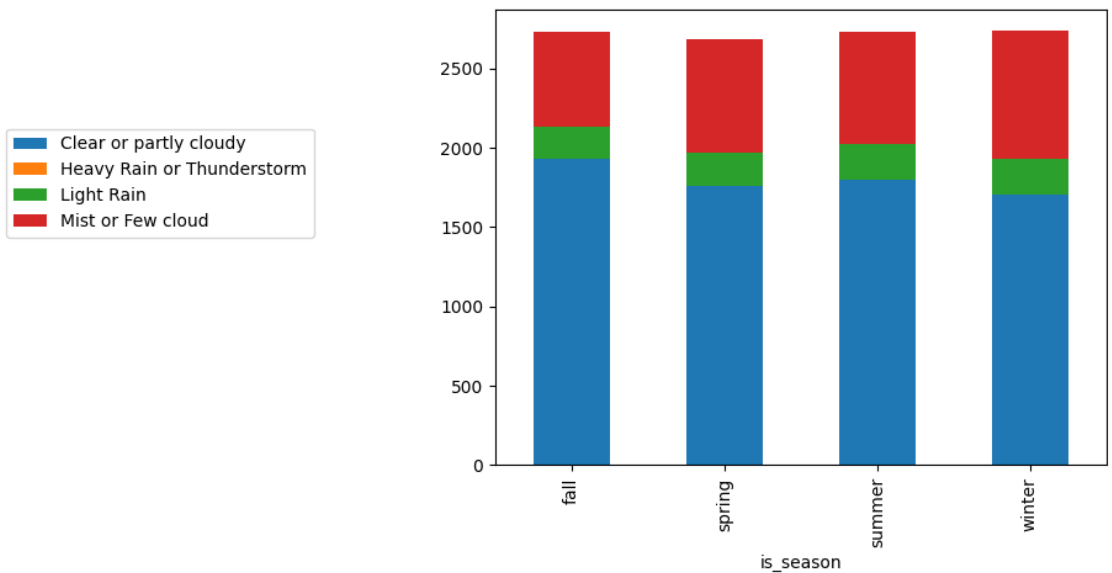
[201]:

```
                  datetime   temp   atemp  humidity  windspeed  casual  \
0      2011-01-01 00:00:00   9.84  14.395        81     0.0000       3
1      2011-01-01 01:00:00   9.02  13.635        80     0.0000       8
2      2011-01-01 02:00:00   9.02  13.635        80     0.0000       5
3      2011-01-01 03:00:00   9.84  14.395        75     0.0000       3
4      2011-01-01 04:00:00   9.84  14.395        75     0.0000       0
…                        …     …      …        …         …
10881  2012-12-19 19:00:00  15.58  19.695        50    26.0027       7
10882  2012-12-19 20:00:00  14.76  17.425        57    15.0013      10
10883  2012-12-19 21:00:00  13.94  15.910        61    15.0013       4
10884  2012-12-19 22:00:00  13.94  17.425        61     6.0032      12
10885  2012-12-19 23:00:00  13.12  16.665        66     8.9981       4

       registered  count        day              is_weather is_season
```

```
0               13     16     holiday  Clear or partly cloudy    spring
1               32     40     holiday  Clear or partly cloudy    spring
2               27     32     holiday  Clear or partly cloudy    spring
3               10     13     holiday  Clear or partly cloudy    spring
4                1      1     holiday  Clear or partly cloudy    spring
…              …     …          …                    …     …
10881          329    336  workingday  Clear or partly cloudy    winter
10882          231    241  workingday  Clear or partly cloudy    winter
10883          164    168  workingday  Clear or partly cloudy    winter
10884          117    129  workingday  Clear or partly cloudy    winter
10885           84     88  workingday  Clear or partly cloudy    winter

[10886 rows x 11 columns]
```

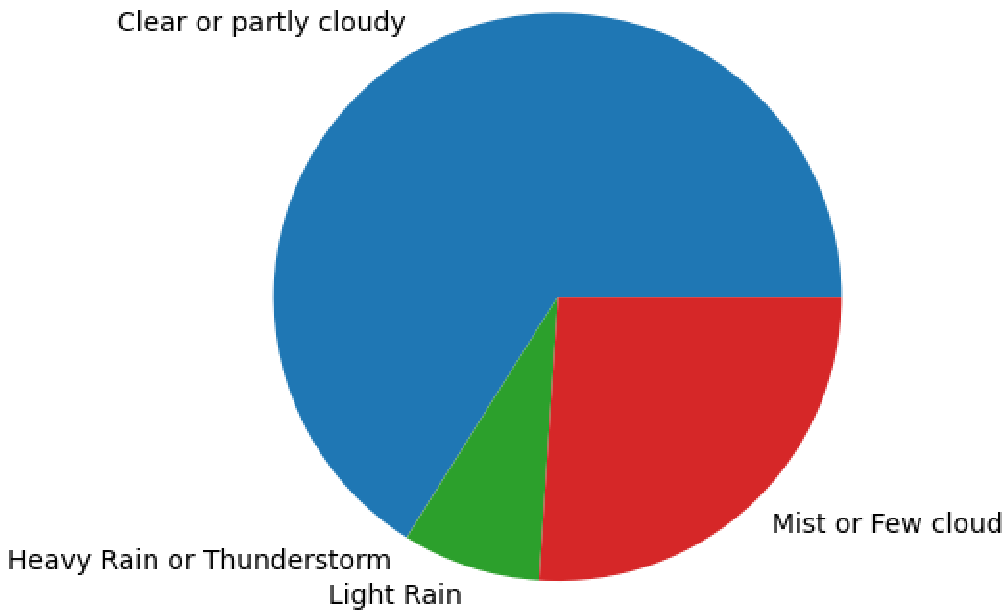
[202]:
```
''' graphical representation of weather condition for every season given '''

df.groupby(['is_season','is_weather']).size().unstack().plot(kind =
 ↪'bar',stacked = True)
plt.legend(loc = (-0.8,0.5))
```

[202]: <matplotlib.legend.Legend at 0x7d406ec665f0>



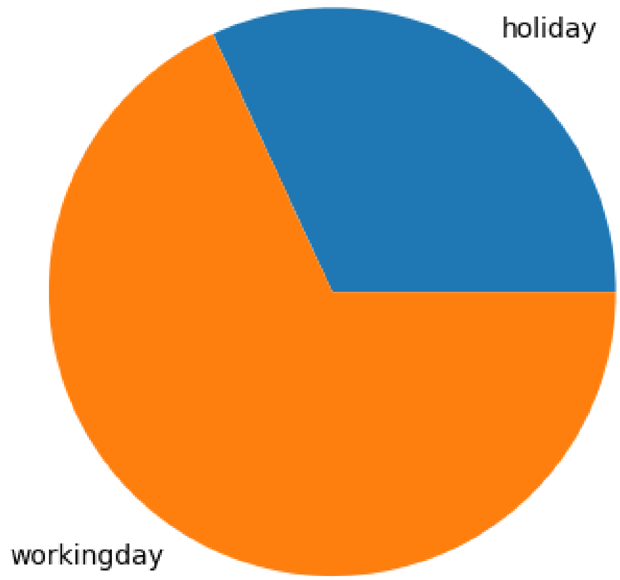
[203]:
```
''' Composition  of weather conditions in the data '''

df.groupby(['is_weather']).size().plot(kind = 'pie',stacked = True)
```

[203]: <Axes: >



[204]:
```
''' composition of day according to data '''

df.groupby(['day']).size().plot(kind = 'pie',stacked = True)
```

[204]: <Axes: >



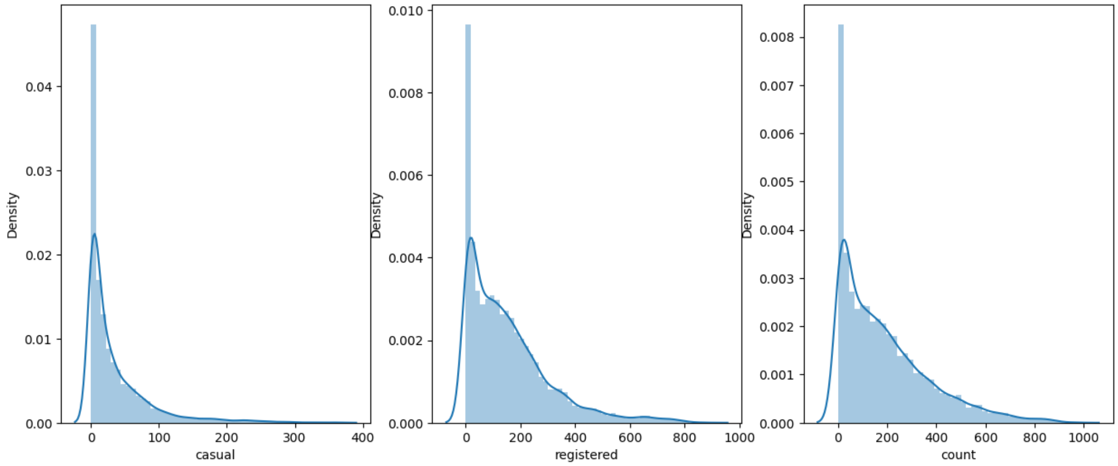
[205]:
```
df.head()
```

[205]:
```
              datetime  temp   atemp  humidity  windspeed  casual  registered  \
0  2011-01-01 00:00:00  9.84  14.395        81        0.0       3          13
1  2011-01-01 01:00:00  9.02  13.635        80        0.0       8          32
2  2011-01-01 02:00:00  9.02  13.635        80        0.0       5          27
3  2011-01-01 03:00:00  9.84  14.395        75        0.0       3          10
4  2011-01-01 04:00:00  9.84  14.395        75        0.0       0           1

   count      day             is_weather is_season
0     16  holiday  Clear or partly cloudy    spring
1     40  holiday  Clear or partly cloudy    spring
2     32  holiday  Clear or partly cloudy    spring
3     13  holiday  Clear or partly cloudy    spring
4      1  holiday  Clear or partly cloudy    spring
```

[206]:
```
''' Distpot for the casual,registered users and the total users plot '''

plt.figure(figsize = (15,6))
plt.subplot(1,3,1)
sns.distplot(df['casual'])
plt.subplot(1,3,2)
sns.distplot(df['registered'])
```

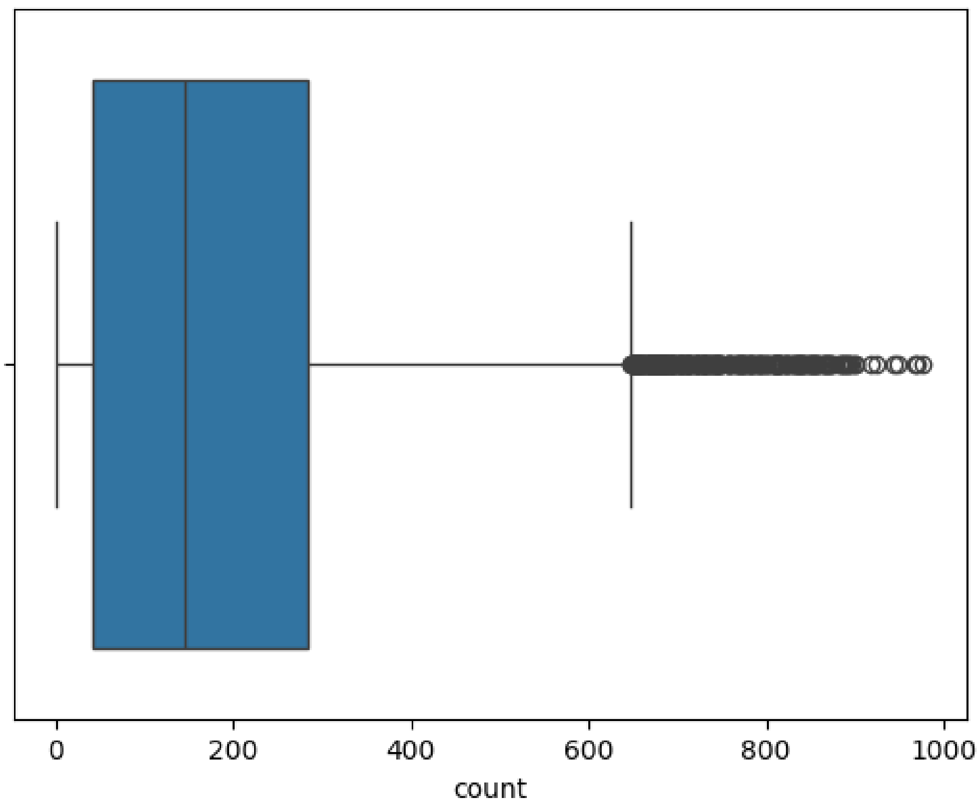
```python
plt.subplot(1,3,3)
sns.distplot(df['count'])

plt.show()
```

```
<ipython-input-206-72467a08eb09>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['casual'])
<ipython-input-206-72467a08eb09>:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['registered'])
<ipython-input-206-72467a08eb09>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['count'])
```



[207]:
```python
''' Determining the outliers in the data '''

sns.boxplot(x = df['count'])
```

[207]: <Axes: xlabel='count'>



[208]:
```python
''' calculating the 25 percentile of total users '''

q1 = df['count'].quantile(0.25)
q1
```

[208]: 42.0

[209]:
```python
''' calculating the 75 percentile of mtotal users '''

q3 = df['count'].quantile(0.75)
q3
```

[209]: 284.0

[210]:
```python
''' calculating the inter-quartiel range for the total users '''

iqr = q3-q1

upper_cut = q3 + 1.5 * iqr
lower_cut = q1 - 1.5 * iqr
```

[211]:
```python
''' calculating the interval where the 95 percent of data lies  for total users␣
↪'''

upper_cut,lower_cut
```

[211]: (647.0, -321.0)

[212]:
```python
''' filtering the data within the range of uper and lower boundaries '''

df[(df['count']< lower_cut) | (df['count']> upper_cut)]
```

[212]:
```
                  datetime   temp   atemp  humidity  windspeed  casual  \
6611   2012-03-12 18:00:00  24.60  31.060        43    12.9980      89
6634   2012-03-13 17:00:00  28.70  31.820        37     7.0015      62
6635   2012-03-13 18:00:00  28.70  31.820        34    19.9995      96
6649   2012-03-14 08:00:00  18.04  21.970        82     0.0000      34
6658   2012-03-14 17:00:00  28.70  31.820        28     6.0032     140
…                        …      …       …         …          …       …
10678  2012-12-11 08:00:00  13.94  15.150        61    19.9995      16
10702  2012-12-12 08:00:00  10.66  12.880        65    11.0014      18
10726  2012-12-13 08:00:00   9.84  11.365        60    12.9980      24
10846  2012-12-18 08:00:00  15.58  19.695        94     0.0000      10
10870  2012-12-19 08:00:00   9.84  12.880        87     7.0015      13

       registered  count         day           is_weather is_season
6611          623    712  workingday       Mist or Few cloud    spring
```

```
6634          614   676  workingday  Clear or partly cloudy    spring
6635          638   734  workingday  Clear or partly cloudy    spring
6649          628   662  workingday  Clear or partly cloudy    spring
6658          642   782  workingday  Clear or partly cloudy    spring
…           …    …          …                      …       …
10678         708   724  workingday       Mist or Few cloud    winter
10702         670   688  workingday       Mist or Few cloud    winter
10726         655   679  workingday  Clear or partly cloudy    winter
10846         652   662  workingday  Clear or partly cloudy    winter
10870         665   678  workingday  Clear or partly cloudy    winter

[300 rows x 11 columns]
```

[213]:
```
''' Standardising and Noramlizing techniques for total users '''

from sklearn.preprocessing import StandardScaler,MinMaxScaler

ss = StandardScaler()

df['count_z'] = ss.fit_transform(df[['count']]) # standardise the data base␣
↪value 0

mm = MinMaxScaler() # normlze the data base value 0.5

df['count_minmax'] = mm.fit_transform(df[['count']])
```

[214]:
```
df[['count','count_z','count_minmax']]
```

[214]:
```
       count   count_z  count_minmax
0         16 -0.969294      0.015369
1         40 -0.836797      0.039959
2         32 -0.880962      0.031762
3         13 -0.985856      0.012295
4          1 -1.052104      0.000000
…        …        …            …
10881    336  0.797333      0.343238
10882    241  0.272866      0.245902
10883    168 -0.130146      0.171107
10884    129 -0.345454      0.131148
10885     88 -0.571803      0.089139

[10886 rows x 3 columns]
```

[215]:
```
df[(df['count_z'] < -3) | (df['count_z'] > 3)␣
↪][['count','count_z','count_minmax']]
```

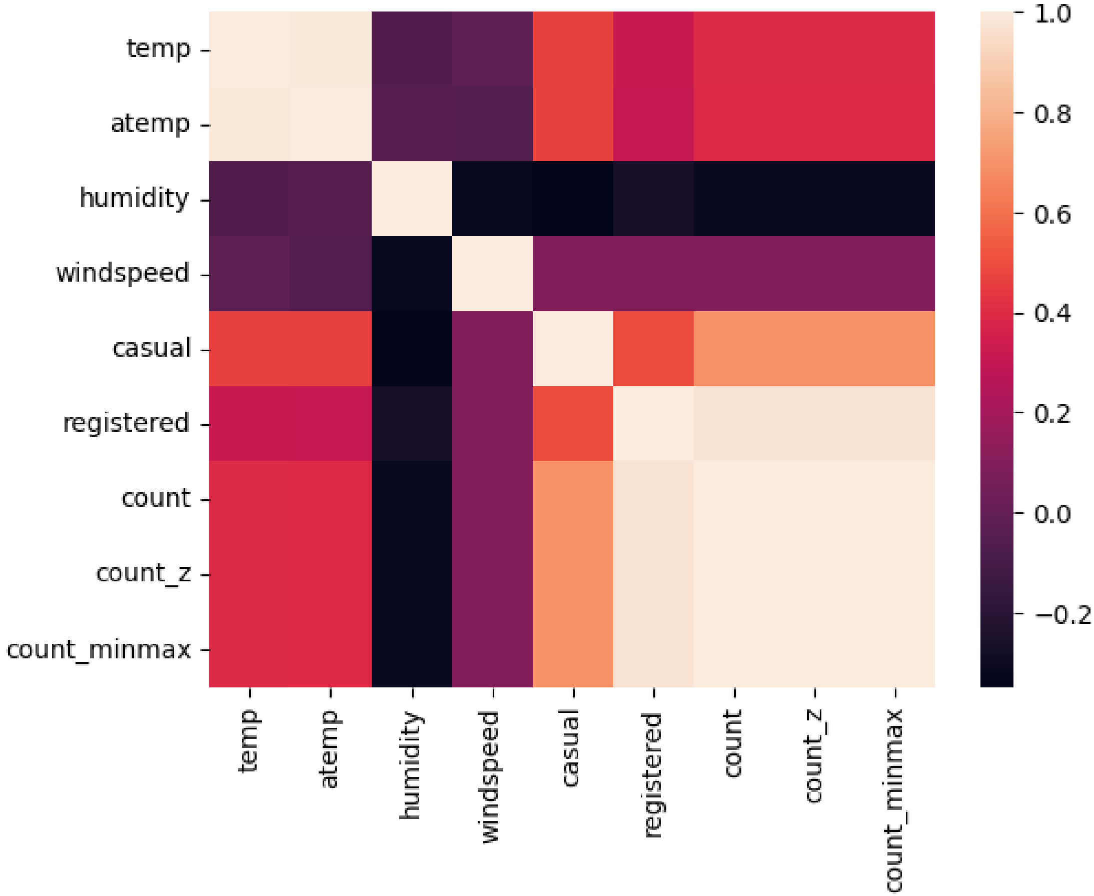
[215]:
```
       count   count_z  count_minmax
6658     782  3.259570      0.800205
6659     749  3.077386      0.766393
6683     746  3.060824      0.763320
6779     801  3.364463      0.819672
6849     757  3.121552      0.774590
…        …         …            …
9935     834  3.546647      0.853484
9944     890  3.855806      0.910861
9945     788  3.292694      0.806352
10519    743  3.044262      0.760246
10534    759  3.132593      0.776639

[147 rows x 3 columns]
```

Plot a Correlation Heatmap and draw insights.

[216]:
```
''' Finding the co-relation between teh columns in the df and plotting a␣
↪heatmap '''

df_cor = df.corr()

sns.heatmap(df_cor)
```

```
<ipython-input-216-adc3370a68be>:3: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df_cor = df.corr()
```

[216]: `<Axes: >`



[217]:
```
df_cor
```

[217]:
```
                 temp     atemp  humidity  windspeed    casual  registered  \
temp         1.000000  0.984948 -0.064949  -0.017852  0.467097    0.318571
atemp        0.984948  1.000000 -0.043536  -0.057473  0.462067    0.314635
humidity    -0.064949 -0.043536  1.000000  -0.318607 -0.348187   -0.265458
windspeed   -0.017852 -0.057473 -0.318607   1.000000  0.092276    0.091052
casual       0.467097  0.462067 -0.348187   0.092276  1.000000    0.497250
registered   0.318571  0.314635 -0.265458   0.091052  0.497250    1.000000
count        0.394454  0.389784 -0.317371   0.101369  0.690414    0.970948
count_z      0.394454  0.389784 -0.317371   0.101369  0.690414    0.970948
count_minmax 0.394454  0.389784 -0.317371   0.101369  0.690414    0.970948

                count   count_z  count_minmax
temp         0.394454  0.394454      0.394454
atemp        0.389784  0.389784      0.389784
humidity    -0.317371 -0.317371     -0.317371
windspeed    0.101369  0.101369      0.101369
```

```
casual         0.690414  0.690414      0.690414
registered     0.970948  0.970948      0.970948
count          1.000000  1.000000      1.000000
count_z        1.000000  1.000000      1.000000
count_minmax   1.000000  1.000000      1.000000
```

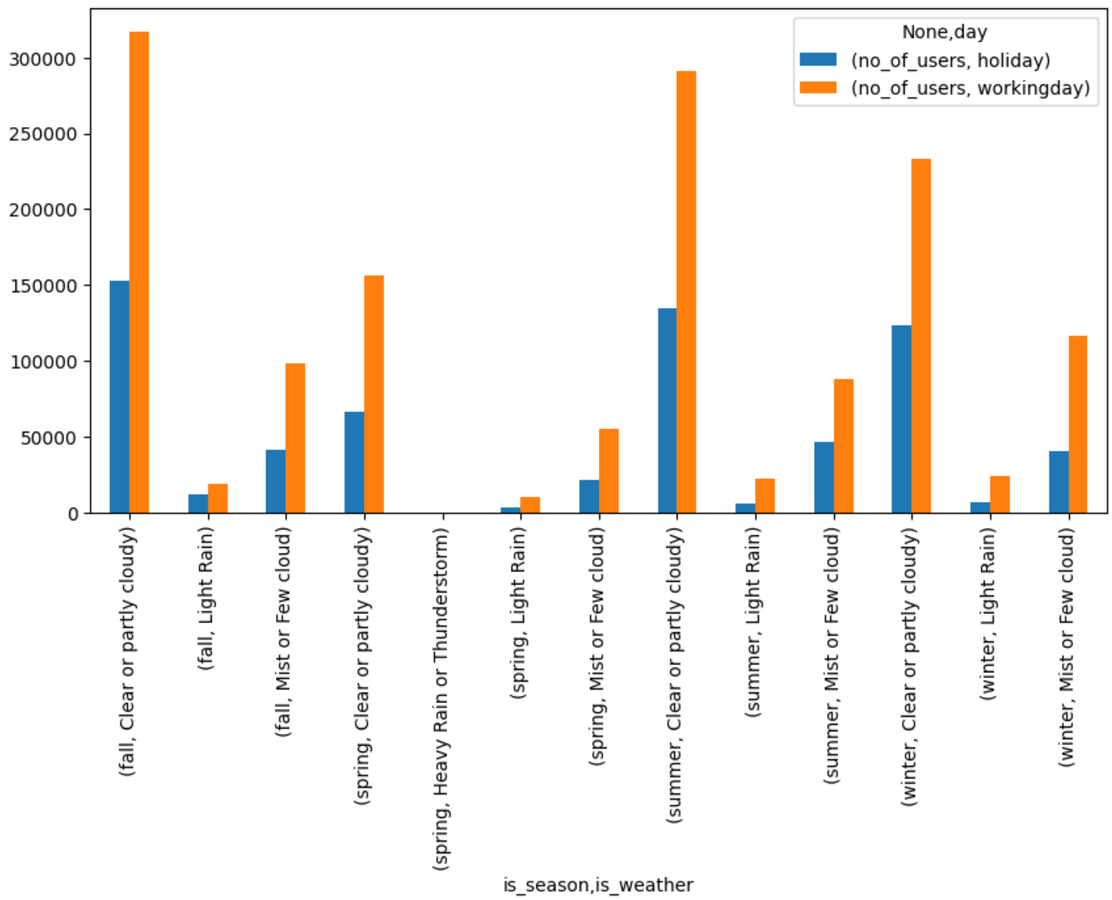
[218]:
```
df.head()
```

[218]:
```
              datetime  temp   atemp  humidity  windspeed  casual  registered  \
0  2011-01-01 00:00:00  9.84  14.395        81        0.0       3          13
1  2011-01-01 01:00:00  9.02  13.635        80        0.0       8          32
2  2011-01-01 02:00:00  9.02  13.635        80        0.0       5          27
3  2011-01-01 03:00:00  9.84  14.395        75        0.0       3          10
4  2011-01-01 04:00:00  9.84  14.395        75        0.0       0           1

   count      day              is_weather is_season   count_z  count_minmax
0     16  holiday  Clear or partly cloudy    spring -0.969294      0.015369
1     40  holiday  Clear or partly cloudy    spring -0.836797      0.039959
2     32  holiday  Clear or partly cloudy    spring -0.880962      0.031762
3     13  holiday  Clear or partly cloudy    spring -0.985856      0.012295
4      1  holiday  Clear or partly cloudy    spring -1.052104      0.000000
```

Working Day has effect on number of electric cycles rented

[219]:
```
'''
1. Here we group the data by season and wether columns and sum the number of␣
↪users in the df.
2. Now we plot the stacked bar plot for the visual refernece of users on for␣
↪season particular weather day. '''


df.groupby(['is_season','is_weather','day']).aggregate(no_of_users =␣
↪('count','sum')).unstack().plot(kind = 'bar',stacked = False,figsize =␣
↪(10,5))
```

[219]:
```
<Axes: xlabel='is_season,is_weather'>
```



[220]:
```
''' Filtering the data by day as 'working day' '''

df1 = df[df['day'] == 'workingday']
df1
```

[220]:
```
                  datetime   temp   atemp  humidity  windspeed  casual  \
47     2011-01-03 00:00:00   9.02   9.850        44    23.9994       0
48     2011-01-03 01:00:00   8.20   8.335        44    27.9993       0
49     2011-01-03 04:00:00   6.56   6.820        47    26.0027       0
50     2011-01-03 05:00:00   6.56   6.820        47    19.0012       0
51     2011-01-03 06:00:00   5.74   5.305        50    26.0027       0
…                        …     …      …         …        …
10881  2012-12-19 19:00:00  15.58  19.695        50    26.0027       7
10882  2012-12-19 20:00:00  14.76  17.425        57    15.0013      10
10883  2012-12-19 21:00:00  13.94  15.910        61    15.0013       4
10884  2012-12-19 22:00:00  13.94  17.425        61     6.0032      12
10885  2012-12-19 23:00:00  13.12  16.665        66     8.9981       4

       registered  count         day              is_weather is_season  \
```

```
47              5      5  workingday  Clear or partly cloudy    spring
48              2      2  workingday  Clear or partly cloudy    spring
49              1      1  workingday  Clear or partly cloudy    spring
50              3      3  workingday  Clear or partly cloudy    spring
51             30     30  workingday  Clear or partly cloudy    spring
…             …     …          …                      …       …
10881         329    336  workingday  Clear or partly cloudy    winter
10882         231    241  workingday  Clear or partly cloudy    winter
10883         164    168  workingday  Clear or partly cloudy    winter
10884         117    129  workingday  Clear or partly cloudy    winter
10885          84     88  workingday  Clear or partly cloudy    winter

        count_z  count_minmax
47    -1.030022      0.004098
48    -1.046584      0.001025
49    -1.052104      0.000000
50    -1.041063      0.002049
51    -0.892004      0.029713
…           …           …
10881  0.797333      0.343238
10882  0.272866      0.245902
10883 -0.130146      0.171107
10884 -0.345454      0.131148
10885 -0.571803      0.089139

[7412 rows x 13 columns]
```

[221]:

```
''' Filtering the data by day as 'Holiday' '''

df2 = df[df['day'] == 'holiday']
df2
```

[221]:

```
                  datetime   temp   atemp  humidity  windspeed  casual  \
0      2011-01-01 00:00:00   9.84  14.395        81     0.0000       3
1      2011-01-01 01:00:00   9.02  13.635        80     0.0000       8
2      2011-01-01 02:00:00   9.02  13.635        80     0.0000       5
3      2011-01-01 03:00:00   9.84  14.395        75     0.0000       3
4      2011-01-01 04:00:00   9.84  14.395        75     0.0000       0
…                      …     …       …       …         …       …
10809  2012-12-16 19:00:00  14.76  17.425        93     8.9981      10
10810  2012-12-16 20:00:00  15.58  19.695        82     0.0000      14
10811  2012-12-16 21:00:00  14.76  18.940        93     0.0000      14
10812  2012-12-16 22:00:00  16.40  20.455        82    12.9980       6
10813  2012-12-16 23:00:00  14.76  17.425        93     8.9981       4

       registered  count      day              is_weather is_season   count_z  \
0              13     16  holiday  Clear or partly cloudy    spring -0.969294
```

```
1              32    40  holiday  Clear or partly cloudy    spring -0.836797
2              27    32  holiday  Clear or partly cloudy    spring -0.880962
3              10    13  holiday  Clear or partly cloudy    spring -0.985856
4               1     1  holiday  Clear or partly cloudy    spring -1.052104
…             …   …    …                   …       …     …
10809          99   109  holiday  Clear or partly cloudy    winter -0.455868
10810         108   122  holiday       Mist or Few cloud    winter -0.384099
10811          92   106  holiday       Mist or Few cloud    winter -0.472430
10812          83    89  holiday       Mist or Few cloud    winter -0.566282
10813          29    33  holiday       Mist or Few cloud    winter -0.875442

       count_minmax
0          0.015369
1          0.039959
2          0.031762
3          0.012295
4          0.000000
…            …
10809      0.110656
10810      0.123975
10811      0.107582
10812      0.090164
10813      0.032787

[3474 rows x 13 columns]
```

[222]:
```python
'''
 Taking the sample of 20 randomly from the two nelwy created dataframes of df1␣
↪and df2
'''


working20 = df1['count'].sample(20)


holiday20 = df2['count'].sample(20)
```

1. Formulating the null and Alternative Hypothesis for the data of bike rentals and the type of day.
2. 'Marking the significance level of 0.05 (confidence of 95%)
3. h0 : day has no effect on bike rentals.
4. ha : day has effect on bike rentals.
5. alpha = 0.05 – 95% confidence.

[223]:
```
''' Conducting the T test individual for bothe the acquired data '''

x1 = working20
x2 = holiday20

from scipy.stats import ttest_ind

ttest_ind(x1,x2)

# hence the p_value > alpha we cant reject null hypothesis and there is no␣
↪effect of days on bike rentals
```

[223]:
```
TtestResult(statistic=1.1826836792249074, pvalue=0.24428177337195203, df=38.0)
```

1. The p_val we observed in the T test individual is 0.58(varies) which is greater than the alpha(significance level) of 0.05.
2. Hence we cannot reject the null and the workingday and holiday has no significant affect on the bike rentals.

[224]:
```
df.groupby(['is_weather']).sum('count')
```

[224]:
```
                                temp       atemp  humidity   windspeed  \
is_weather
Clear or partly cloudy     147846.82  172565.755    407907  92723.1626
Heavy Rain or Thunderstorm      8.20      11.365        86      6.0032
Light Rain                  16790.32   19544.905     69872  12087.2020
Mist or Few cloud           55587.80   65387.220    195831  34517.8506

                            casual  registered    count     count_z  \
is_weather
Clear or partly cloudy      289900     1186163  1476063  542.475106
Heavy Rain or Thunderstorm       6         158      164   -0.152229
Light Rain                   14983       87106   102089 -344.896284
Mist or Few cloud            87246      419914   507160 -197.426594

                            count_minmax
is_weather
Clear or partly cloudy       1504.990779
Heavy Rain or Thunderstorm      0.167008
Light Rain                    103.719262
Mist or Few cloud             516.727459
```

Check if the demand of bicycles on rent is the same for different Weather conditions?

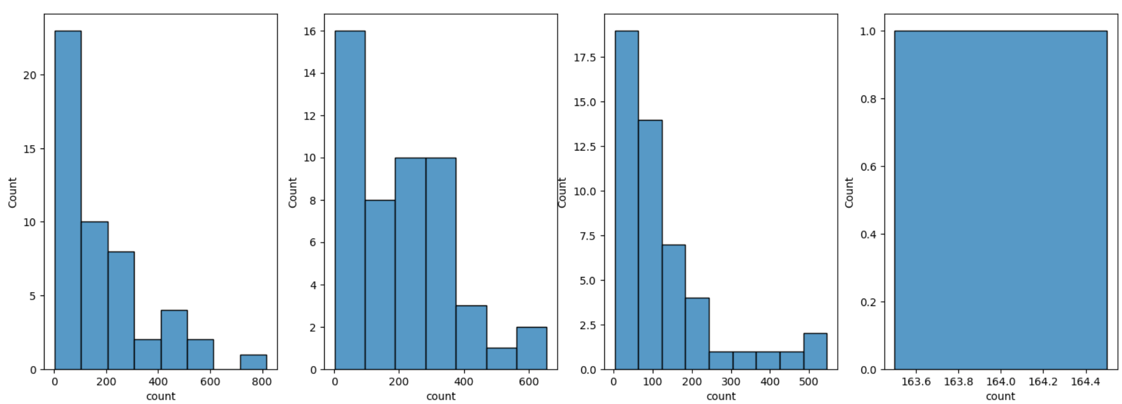
[225]:
```
''' Taking the sample data of 20 records for each weather type.'''


mist50 = df[df['is_weather']== 'Mist or Few cloud']['count'].sample(50)
```

```
clear50 = df[df['is_weather'] == 'Clear or partly cloudy']['count'].sample(50)

lite50 = df[df['is_weather'] == 'Light Rain']['count'].sample(50)

heavyrain50 = df[df['is_weather'] == 'Heavy Rain or Thunderstorm']['count']
```

1. Formulating the null and alternqative hypothesis for data of bike rentals and weather conditions.
2. Marking the significance level of 0.05(95 % confidence)
3. H0 : Weather conditions has effect on bike rentals.
4. Ha : Weather conditions has no effect on tbike rentals.
5. alpha : 0.05 – 95% confidence

[226]:
```
''' plotting the histogram for the sample data for each weather type '''

plt.figure(figsize = (18,6))
plt.subplot(1,4,1)
sns.histplot(mist50)
plt.subplot(1,4,2)
sns.histplot(clear50)
plt.subplot(1,4,3)
sns.histplot(lite50)
plt.subplot(1,4,4)
sns.histplot(heavyrain50)
```

[226]: <Axes: xlabel='count', ylabel='Count'>

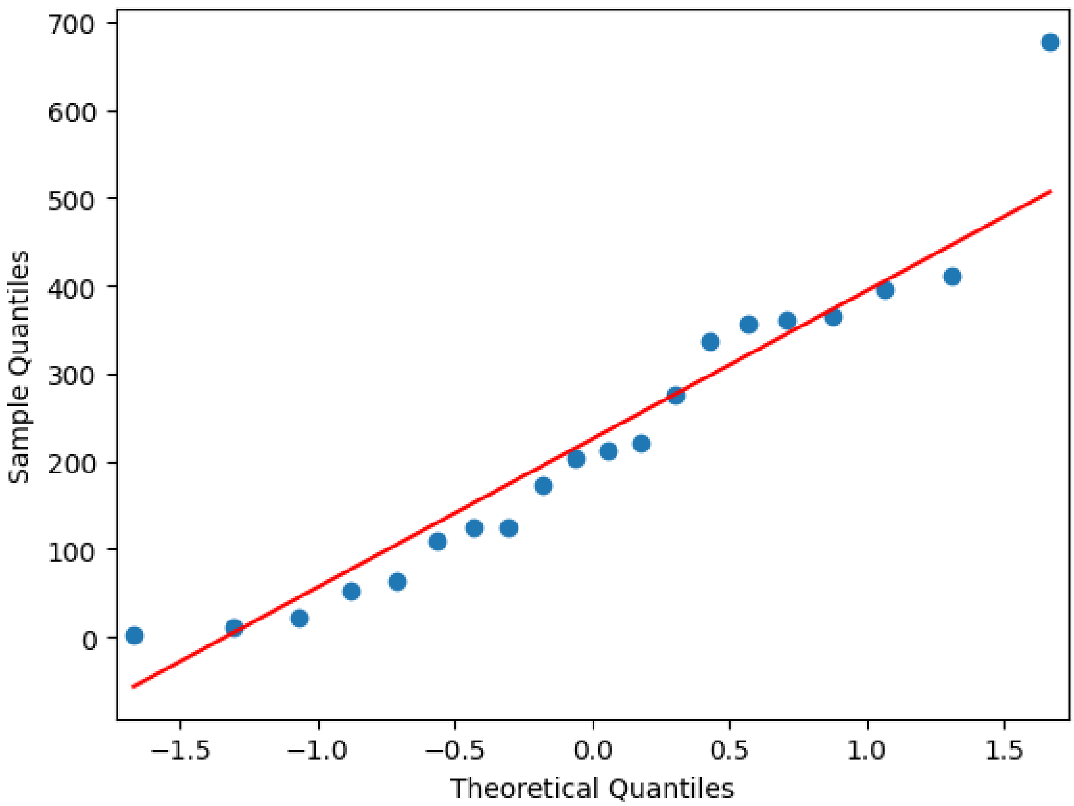


[227]:
```
mist20 = df[df['is_weather']== 'Mist or Few cloud']['count'].sample(20)

clear20 = df[df['is_weather'] == 'Clear or partly cloudy']['count'].sample(20)

lite20 = df[df['is_weather'] == 'Light Rain']['count'].sample(20)
```

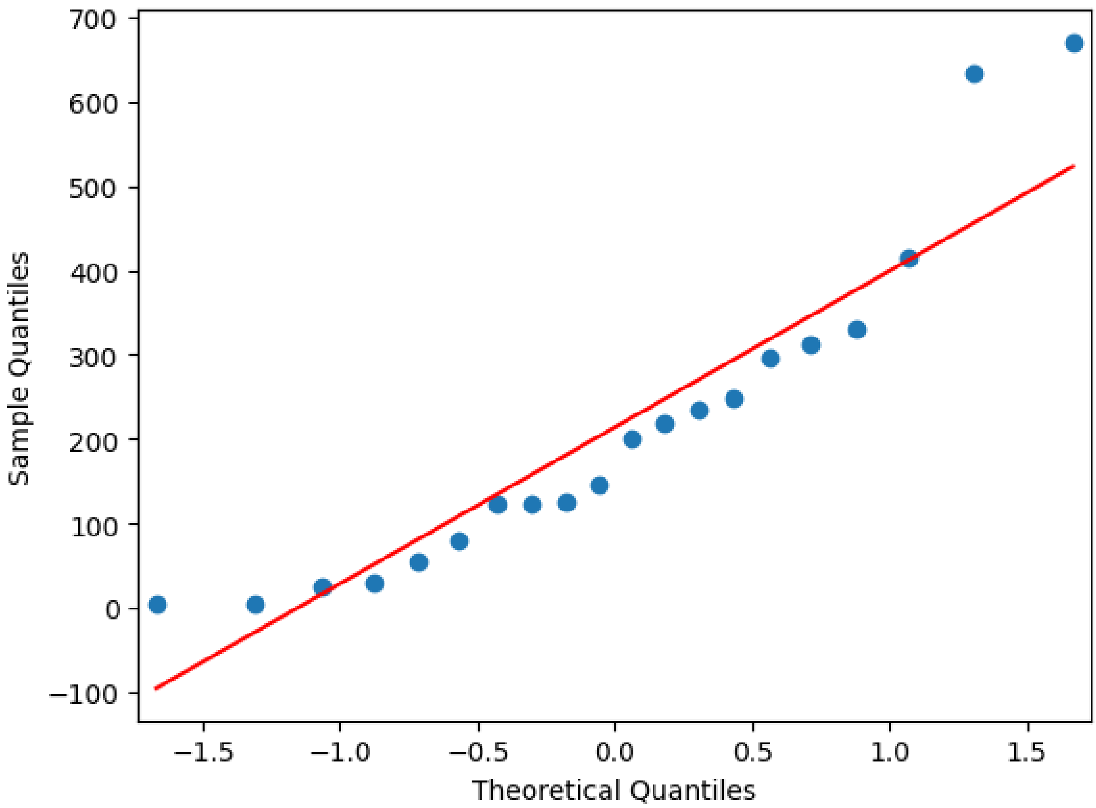
```
heavyrain20 = df[df['is_weather'] == 'Heavy Rain or Thunderstorm']['count']
```

[228]:
```
''' Plotting the QQ-plot for the data mist20 '''

import statsmodels.api as sm
plt.figure(figsize = (15,6))

sm.qqplot(mist20,line = 's')
plt.show()
```

<Figure size 1500x600 with 0 Axes>

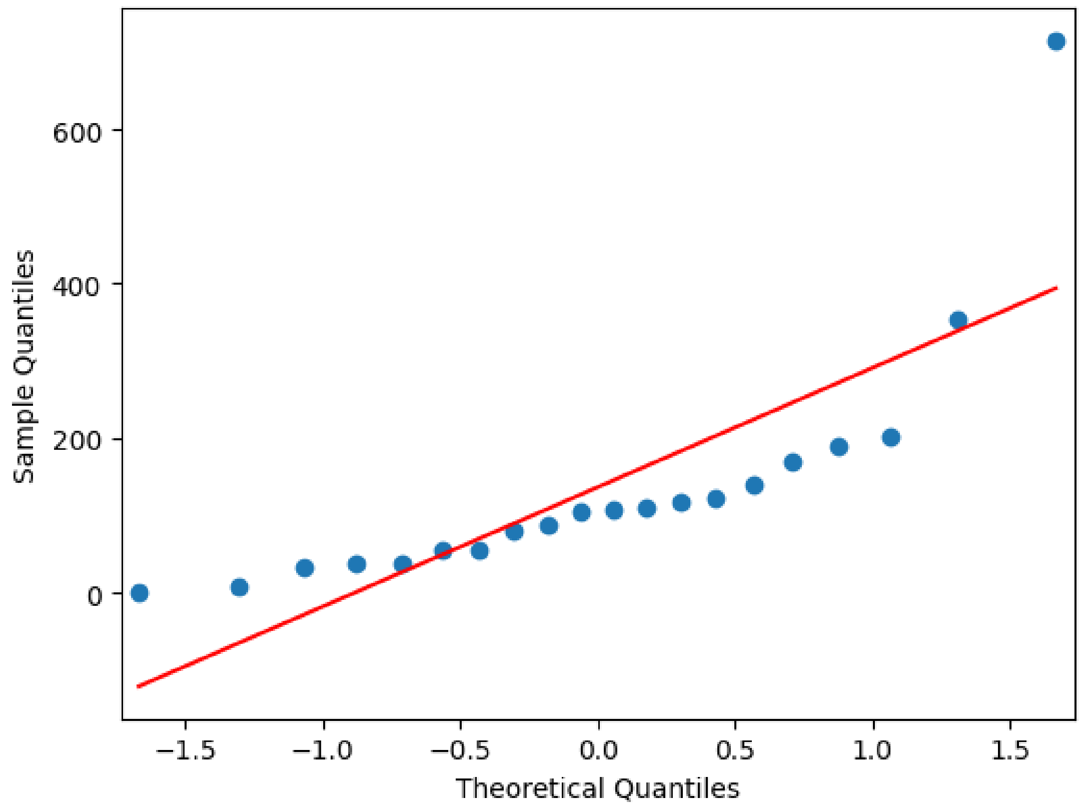


[229]:
```
''' Plotting the QQ-plot for the data clear20 '''

sm.qqplot(clear20,line = 's')
plt.show()
```

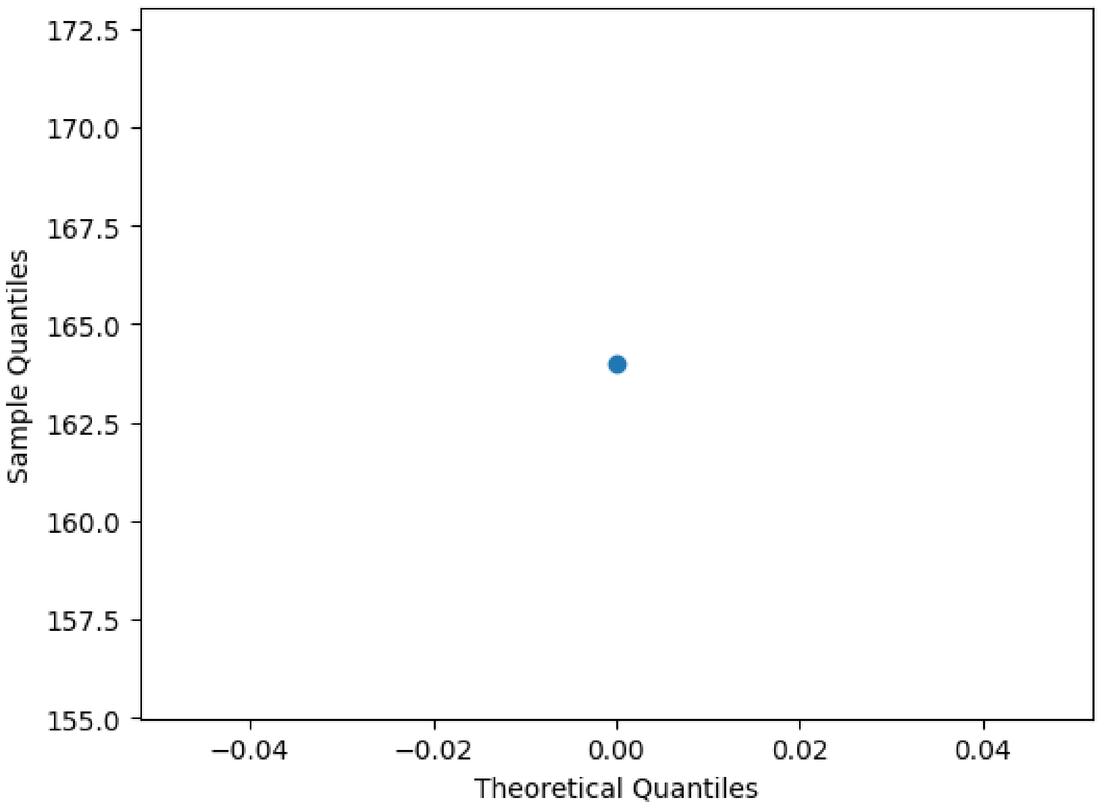


[230]:

```
''' Plotting the QQ-plot for the data lite20 '''

sm.qqplot(lite20,line = 's')
plt.show()
```



[231]:
```
''' Plotting the QQ-plot for the data heavyrain20 '''

sm.qqplot(heavyrain20,line = 's')
plt.show()

''' We can ignore this data as there is only one data point for this filter of␣
↪heavyrain in the df '''
```



[231]: ' We can ignore this data as there is only one data point for this filter of heavyrain in the df '

[232]:
```
''' Performing the levene,kruskal and one_way anova tests for the data of␣
↪weather conditions'''

from scipy.stats import levene,kruskal,f_oneway

levene(mist50,clear50,lite50,heavyrain50)
```

[232]: LeveneResult(statistic=1.8405200404644044, pvalue=0.14235047295888245)

[233]:
```
''' Kruskal test for weather conditions of mist20,clear20,lite20,heavyrain20 '''

kruskal(mist50,clear50,lite50,heavyrain50)
```

[233]: KruskalResult(statistic=8.862939039863937, pvalue=0.031169797575285513)

[234]:
```
''' One way Anova test for weather conditions of␣
↪mist20,clear20,lite20,heavyrain20 '''
```

```
f_oneway(mist50,clear50,lite50,heavyrain50)
```

[234]: F_onewayResult(statistic=2.548582152275388, pvalue=0.05810771832902403)

1. The p _val is less than alpha value(0.05).
2. We can reject null hypothesis and coclude bike rentals depends on weather conditions.

[235]:
```
''' Performing the Shapiro- Wallis test for the weather sample mist20 '''

from scipy.stats import shapiro
t_test,p_val = shapiro(mist20)
p_val
```

[235]: 0.14603275060653687

As the p_value is very smaller and away from 1, we can conclude that the distribution is not normal

[236]:
```
''' Performing the Shapiro- Wallis test for the weather sample clear20 '''

t_test,p_val = shapiro(clear20)
p_val
```

[236]: 0.018626874312758446

As the p_value is very smaller and away from 1, we can conclude that the distribution is not normal

[237]:
```
''' Performing the Shapiro- Wallis test for the weather sample lite20 '''

t_test,p_val = shapiro(lite20)
p_val
```

[237]: 2.068323919957038e-05

As the p_value is very smaller and away from 1, we can conclude that the distribution is not normal

.

Check if the demand of bicycles on rent is the same for different Seasons?

[238]:
```
df
```

[238]:
```
                  datetime   temp   atemp  humidity  windspeed  casual  \
0      2011-01-01 00:00:00   9.84  14.395        81     0.0000       3
1      2011-01-01 01:00:00   9.02  13.635        80     0.0000       8
2      2011-01-01 02:00:00   9.02  13.635        80     0.0000       5
3      2011-01-01 03:00:00   9.84  14.395        75     0.0000       3
4      2011-01-01 04:00:00   9.84  14.395        75     0.0000       0
…                        …     …      …        …       …      …
10881  2012-12-19 19:00:00  15.58  19.695        50    26.0027       7
```

```
10882  2012-12-19 20:00:00  14.76  17.425        57     15.0013       10
10883  2012-12-19 21:00:00  13.94  15.910        61     15.0013        4
10884  2012-12-19 22:00:00  13.94  17.425        61      6.0032       12
10885  2012-12-19 23:00:00  13.12  16.665        66      8.9981        4

       registered  count         day              is_weather is_season  \
0              13     16     holiday  Clear or partly cloudy    spring
1              32     40     holiday  Clear or partly cloudy    spring
2              27     32     holiday  Clear or partly cloudy    spring
3              10     13     holiday  Clear or partly cloudy    spring
4               1      1     holiday  Clear or partly cloudy    spring
…             …    …         …                    …       …
10881         329    336  workingday  Clear or partly cloudy    winter
10882         231    241  workingday  Clear or partly cloudy    winter
10883         164    168  workingday  Clear or partly cloudy    winter
10884         117    129  workingday  Clear or partly cloudy    winter
10885          84     88  workingday  Clear or partly cloudy    winter

        count_z  count_minmax
0     -0.969294      0.015369
1     -0.836797      0.039959
2     -0.880962      0.031762
3     -0.985856      0.012295
4     -1.052104      0.000000
…           …           …
10881  0.797333      0.343238
10882  0.272866      0.245902
10883 -0.130146      0.171107
10884 -0.345454      0.131148
10885 -0.571803      0.089139

[10886 rows x 13 columns]
```

[239]:

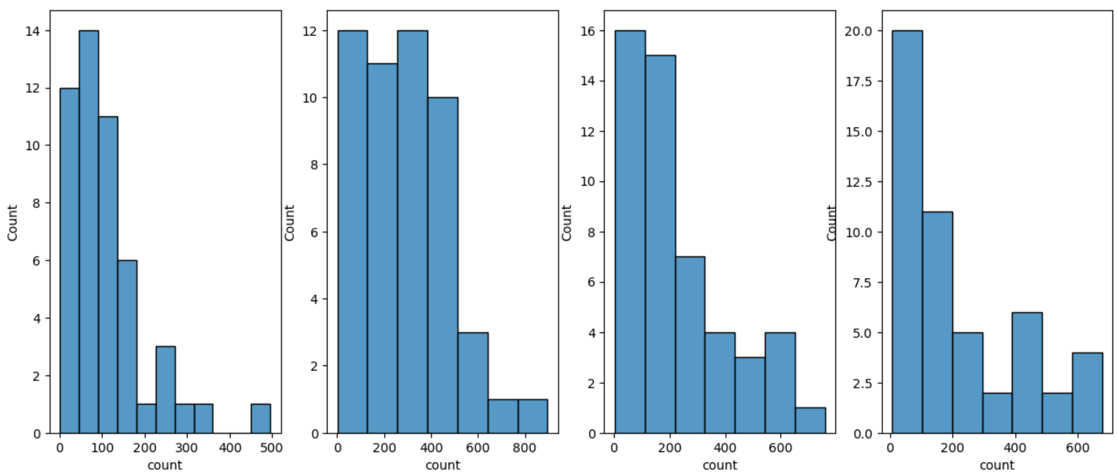
```
'''' Creating the data for the different season by taking sample of 20 random␣
↪records. '''


spring50 = df[df['is_season'] == 'spring']['count'].sample(50)

fall50 = df[df['is_season'] == 'fall']['count'].sample(50)

summer50 = df[df['is_season'] == 'summer']['count'].sample(50)

winter50 = df[df['is_season'] == 'winter']['count'].sample(50)
```

[240]:

```
''' Histogram for the data for different seasons '''
```

```
plt.figure(figsize = (15,6))
plt.subplot(1,4,1)
sns.histplot(spring50)
plt.subplot(1,4,2)
sns.histplot(fall50)
plt.subplot(1,4,3)
sns.histplot(summer50)
plt.subplot(1,4,4)
sns.histplot(winter50)
```

[240]: <Axes: xlabel='count', ylabel='Count'>



1. Formulating the NUll and Alternative Hypothesis for the Seasos data.
2. H0 : bike rentals are equal in all seasons.
3. Ha : bike rentals are not equal in all seasons.

[241]:
```
''' Performing the levene,kruskal,One-way Anova for the following seasons data
↪'''

from scipy.stats import levene,kruskal,f_oneway


levene(spring50,summer50,fall50,winter50)
```

[241]: LeveneResult(statistic=6.97282490618, pvalue=0.00017584915575216578)

[242]:
```
''' kruskal test for seasons data '''

kruskal(spring50,summer50,fall50,winter50)
```

[242]: KruskalResult(statistic=20.675205459827282, pvalue=0.00012295873453147527)

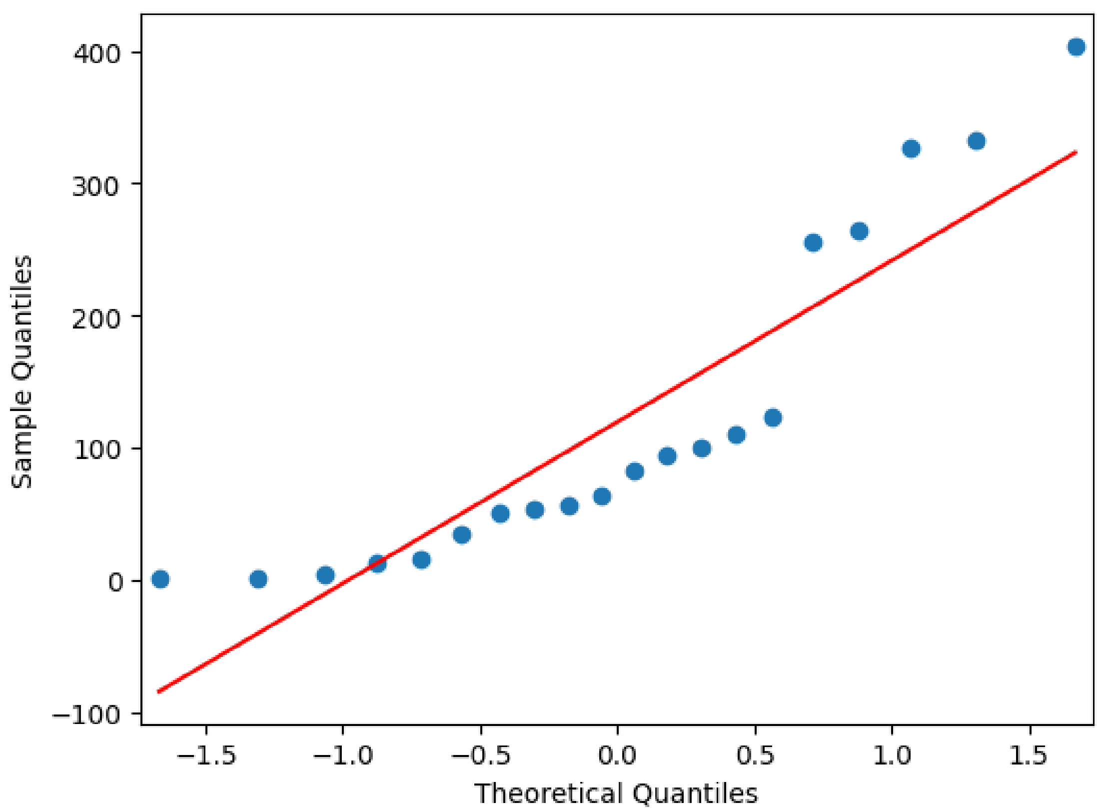
[243]:
```
''' One way Anova test for season data '''

f_oneway(spring50,summer50,fall50,winter50)
```

[243]: F_onewayResult(statistic=7.945570705995613, pvalue=5.007287903641841e-05)

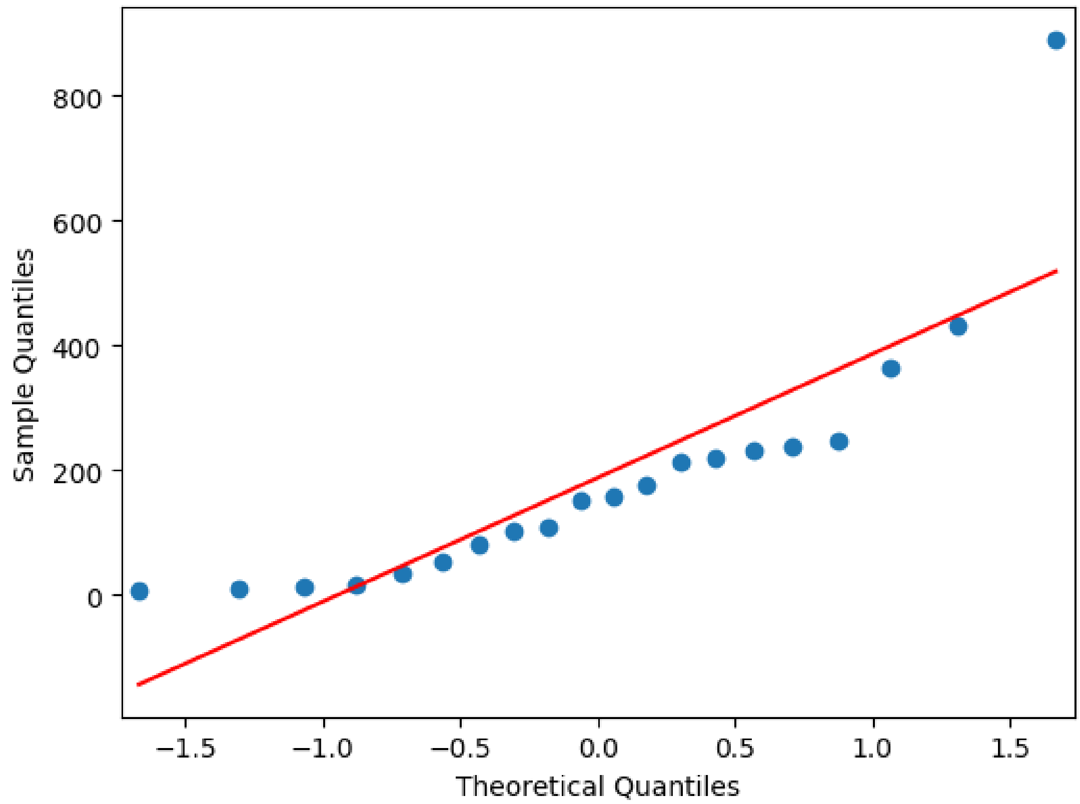
as p_val is lesser than alpha(0.05) we can reject null and conclude that bike rentals are effected by seasons.

[244]:
```
spring20 = df[df['is_season'] == 'spring']['count'].sample(20)

fall20 = df[df['is_season'] == 'fall']['count'].sample(20)

summer20 = df[df['is_season'] == 'summer']['count'].sample(20)

winter20 = df[df['is_season'] == 'winter']['count'].sample(20)
```

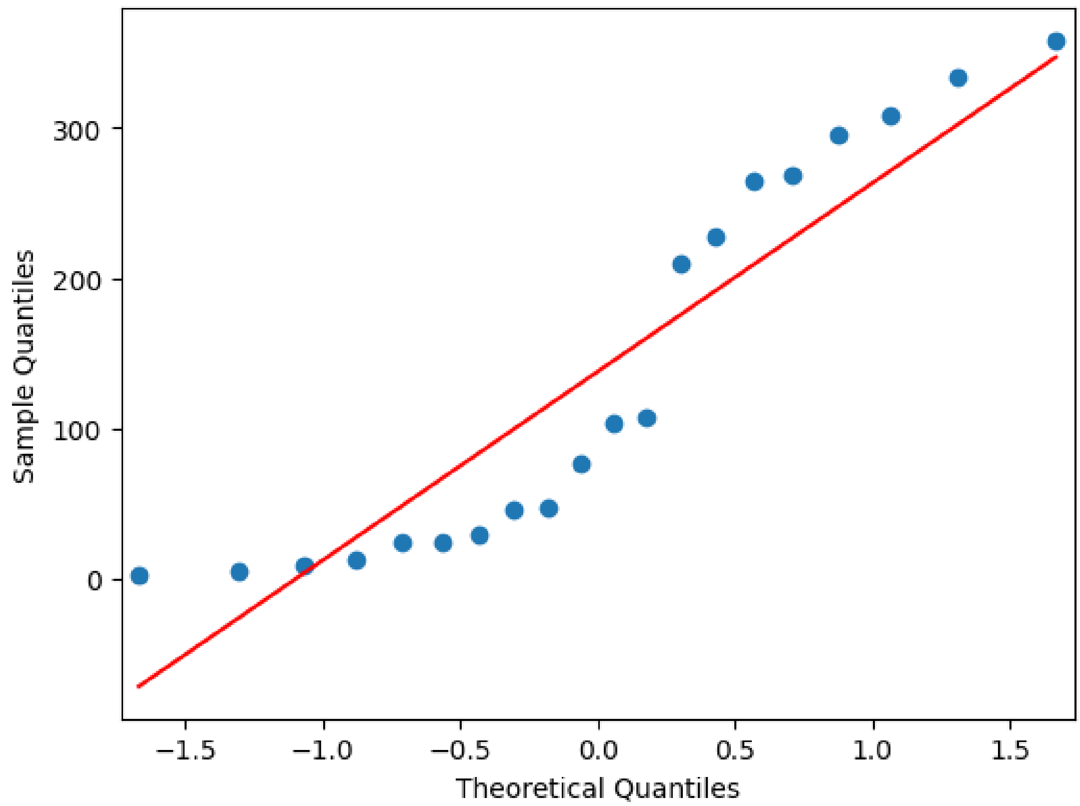
[245]:
```
sm.qqplot(spring20,line = 's')
plt.show()
```



[246]:
```
sm.qqplot(fall20,line = 's')
plt.show()
```

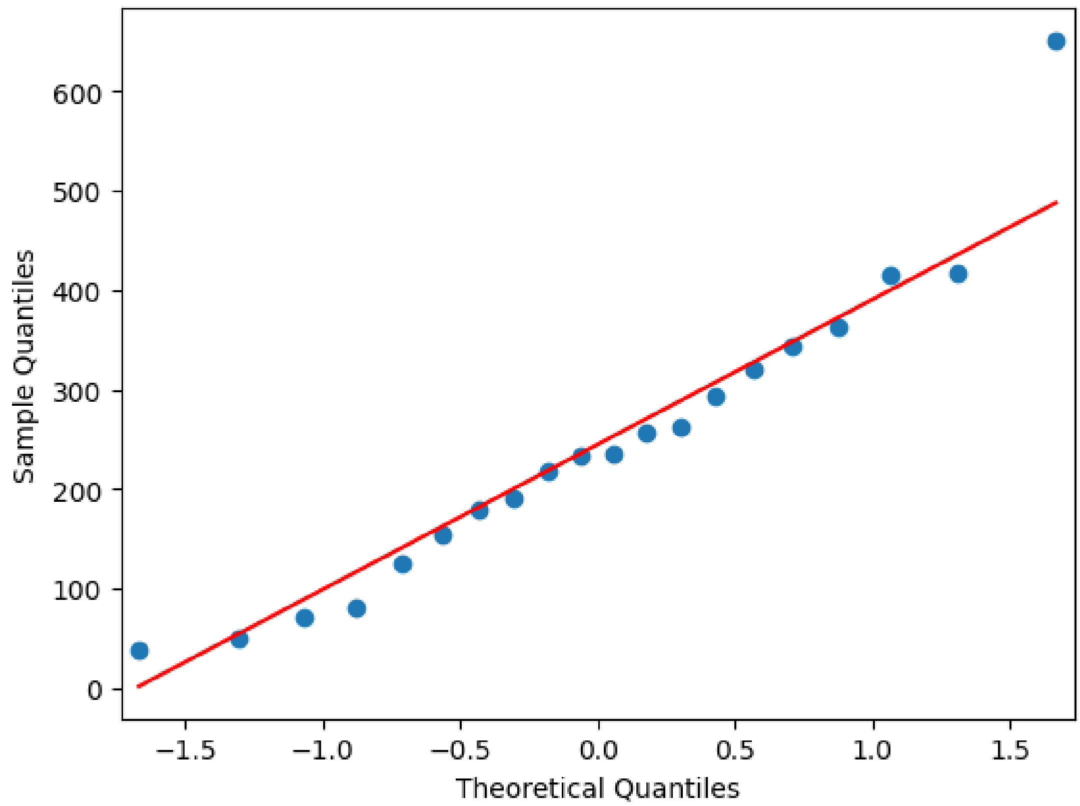


[247]:
```
sm.qqplot(summer20,line = 's')
plt.show()
```



[248]:

```
sm.qqplot(winter20,line = 's')
plt.show()
```



[248]:

```

```

[249]:

```
from scipy.stats import shapiro

t_test,p_val = shapiro(spring20)
p_val
```

[249]: 0.0019803964532911777

[250]:

```
t_test,p_val = shapiro(fall20)
p_val
```

[250]: 0.00024950012448243797

[251]:

```
t_test,p_val = shapiro(summer20)
p_val
```

[251]: 0.005241308361291885

[252]:

```
t_test,p_val = shapiro(winter20)
p_val
```

[252]: 0.25809207558631897

As the p_ values for each seasons for saamples of 20 each is very small i.e far from 1 that states the distribution is not normal.

.

Check if the demand of bicycles on rent is the same for different Seasons?

[253]:
```
''' Taking the categorical columns of weather and seasons in the data and␣
↪checking their dependecies over ecah other '''

pd.crosstab(df['is_season'],df['is_weather'],margins = 'index')
```

[253]:
```
is_weather  Clear or partly cloudy  Heavy Rain or Thunderstorm  Light Rain  \
is_season
fall                          1930                           0         199
spring                        1759                           1         211
summer                        1801                           0         224
winter                        1702                           0         225
All                           7192                           1         859

is_weather  Mist or Few cloud    All
is_season
fall                      604   2733
spring                    715   2686
summer                    708   2733
winter                    807   2734
All                      2834  10886
```

1. Formulating the Null and Alternative Hypothesis for the data of weather and season categorical columns.
2. H0 : Weather conditions are not significantly different over each season.
3. Ha : Weather conditions are significantly different over each season.
4. sigificance level – 0.05 (95 % confidence)

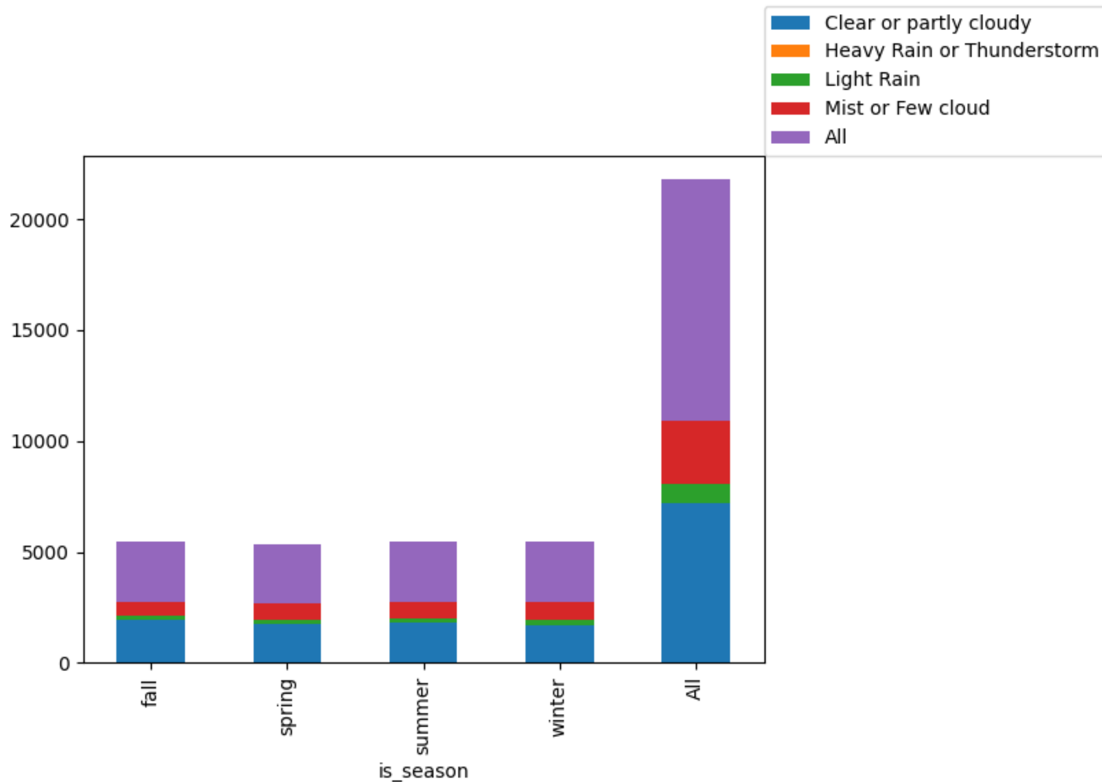
[254]:
```
''' Conducting the chisquare test for contingency  over the season and weather␣
↪columns'''

from scipy.stats import chi2_contingency

chi2_contingency([[1930,199,604],[1759,211,715],[1801,224,708],[1702,225,807]])
```

[254]:
```
Chi2ContingencyResult(statistic=46.10145731073249,
pvalue=2.8260014509929343e-08, dof=6, expected_freq=array([[1805.76352779,
215.67726229,  711.55920992],
       [1774.04869086,  211.8892972 ,  699.06201194],
       [1805.76352779,  215.67726229,  711.55920992],
       [1806.42425356,  215.75617823,  711.81956821]]))
```

1. The p_val obtained is less then the alpha(0.05) value.
2. We can reject the null hypothesis and suport the alternative hypothesis. – There is a siginificant change in weather conditions over each season.

[255]:
```
''' Visual representation of weather conditions over each season '''

pd.crosstab(df['is_season'],df['is_weather'],margins = 'index').plot(kind =␣
↪'bar',stacked = True)
plt.legend(loc = (1,1))
```

[255]: <matplotlib.legend.Legend at 0x7d406e743310>



2 Recommendations

1. The season plays a vital role in the bike rentals and number of users.
2. There should be siginificant bike avalability on the holidays as far as at the pickup and drop points.
3. The weather conditons also has major affect on the users of bike rentals.
4. The registered users are also not much greater then the csual users, there should be necessary steps taken to attarct the users for registering into the app by offering discount rides, weather forecasting and other customer gaining techniques.

5. Most of the people won't get the bike for rentqals at peal hours of morning,afternoon and evening of working days and holidays, there should be effort to resolve this issues,
6. The seasonal weather conditions are very important for rental of the bike.
7. The temperature and humidity also effects the bike rentals, there shuld be enough forecasting data over a period for specific region of every peak point of the region inorder to help the customers for the weather conditions and other predictions of peak hour,free hours.
8. Making the customer satisfaction is key for the this bussiness as it runs on the service offered to the customers.
9. Having a right and timed deartments for every aspect of the company lead to the success fro long years of the company.
10. These are few recomendations from my side. Thank you.