Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute. Starting off as a mission to eliminate traffic congestion in India, Yulu provides the safest commute solution through a user-friendly mobile app to enable shared, solo and sustainable commuting.

Yulu zones are located at all the appropriate locations (including metro stations, bus stands, office spaces, residential areas, corporate offices, etc) to make those first and last miles smooth, affordable, and convenient!

Yulu has recently suffered considerable dips in its revenues. They have contracted a consulting company to understand the factors on which the demand for these shared electric cycles depends. Specifically, they want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

How you can help here?

The company wants to know:

Which variables are significant in predicting the demand for shared electric cycles in the Indian market?

How well those variables describe the electric cycle demands

importing python libraries

In [24]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [25]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 180)
pd.set_option('display.max_colwidth', 120)
```

In [26]:
```
df = pd.read_csv('D:\\Learning\\scaler data science\\Testing\\business case\\Yulu_data.csv')
print(df.head())
print(df.tail())
```

```
              datetime  season  holiday  workingday  weather  temp   atemp  humidity  windspeed  casual  registered  count
0  2011-01-01 00:00:00       1        0           0        1  9.84  14.395        81        0.0       3          13     16
1  2011-01-01 01:00:00       1        0           0        1  9.02  13.635        80        0.0       8          32     40
2  2011-01-01 02:00:00       1        0           0        1  9.02  13.635        80        0.0       5          27     32
3  2011-01-01 03:00:00       1        0           0        1  9.84  14.395        75        0.0       3          10     13
4  2011-01-01 04:00:00       1        0           0        1  9.84  14.395        75        0.0       0           1      1
                  datetime  season  holiday  workingday  weather   temp   atemp  humidity  windspeed  casual  registered  count
10881  2012-12-19 19:00:00       4        0           1        1  15.58  19.695        50    26.0027       7         329    336
10882  2012-12-19 20:00:00       4        0           1        1  14.76  17.425        57    15.0013      10         231    241
10883  2012-12-19 21:00:00       4        0           1        1  13.94  15.910        61    15.0013       4         164    168
10884  2012-12-19 22:00:00       4        0           1        1  13.94  17.425        61     6.0032      12         117    129
10885  2012-12-19 23:00:00       4        0           1        1  13.12  16.665        66     8.9981       4          84     88
```

datetime: datetime

season: season (1: spring, 2: summer, 3: fall, 4: winter)

holiday: whether day is a holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule)

workingday: if day is neither weekend nor holiday is 1, otherwise is 0.

weather:

1: Clear, Few clouds, partly cloudy, 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 temp: temperature in Celsius

atemp: feeling temperature in Celsius

humidity: humidity

windspeed: wind speed

casual: count of casual users

registered: count of registered users

count: count of total rental bikes including both casual and registered

In [27]:
```
print(df.columns)
print(df.shape)
```

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'], dtype='object')
(10886, 12)
```

In [28]:
```
print(df.info())
print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   datetime    10886 non-null  object 
 1   season      10886 non-null  int64  
 2   holiday     10886 non-null  int64  
 3   workingday  10886 non-null  int64  
 4   weather     10886 non-null  int64  
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64  
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64  
 10  registered  10886 non-null  int64  
 11  count       10886 non-null  int64  
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
None
             season       holiday    workingday       weather         temp         atemp      humidity     windspeed        casual    registered         count
count  10886.000000  10886.000000  10886.000000  10886.000000  10886.00000  10886.000000  10886.000000  10886.000000  10886.000000  10886.000000  10886.000000
mean       2.506614      0.028569      0.680875      1.418427     20.23086     23.655084     61.886460     12.799395     36.021955    155.552177    191.574132
std        1.116174      0.166599      0.466159      0.633839      7.79159      8.474601     19.245033      8.164537     49.960477    151.039033    181.144454
min        1.000000      0.000000      0.000000      1.000000      0.82000      0.760000      0.000000      0.000000      0.000000      0.000000      1.000000
25%        2.000000      0.000000      0.000000      1.000000     13.94000     16.665000     47.000000      7.001500      4.000000     36.000000     42.000000
50%        3.000000      0.000000      1.000000      1.000000     20.50000     24.240000     62.000000     12.998000     17.000000    118.000000    145.000000
75%        4.000000      0.000000      1.000000      2.000000     26.24000     31.060000     77.000000     16.997900     49.000000    222.000000    284.000000
max        4.000000      1.000000      1.000000      4.000000     41.00000     45.455000    100.000000     56.996900    367.000000    886.000000    977.000000
```

In [29]:
```
print(df.isnull().sum())
print(df.nunique())
print(df.head())
```

```
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
datetime      10886
season            4
holiday           2
workingday        2
weather           4
temp             49
atemp            60
humidity         89
windspeed        28
casual          309
registered      731
count           822
dtype: int64
              datetime  season  holiday  workingday  weather  temp   atemp  humidity  windspeed  casual  registered  count
0  2011-01-01 00:00:00       1        0           0        1  9.84  14.395        81        0.0       3          13     16
1  2011-01-01 01:00:00       1        0           0        1  9.02  13.635        80        0.0       8          32     40
2  2011-01-01 02:00:00       1        0           0        1  9.02  13.635        80        0.0       5          27     32
3  2011-01-01 03:00:00       1        0           0        1  9.84  14.395        75        0.0       3          10     13
4  2011-01-01 04:00:00       1        0           0        1  9.84  14.395        75        0.0       0           1      1
```

In [30]:
```
df.skew(numeric_only = True)
```

Out[30]:
```
season       -0.007076
holiday       5.660517
workingday   -0.776163
weather       1.243484
temp          0.003691
atemp        -0.102560
humidity     -0.086335
windspeed     0.588767
casual        2.495748
registered    1.524805
count         1.242066
dtype: float64
```

The majority of the variables, including 'season' and 'temp', exhibit skewness values close to zero, suggesting relatively symmetrical distributions.

Variables such as 'holiday', 'weather', 'windspeed', 'casual', 'registered', and 'count' demonstrate positive skewness, pointing to a concentration of lower values and a rightward skew in their distributions.

In contrast, 'workingday', 'atemp', and 'humidity' exhibit negative skewness, implying a concentration of higher values and a leftward skew in their distributions.

In [31]:
```
df['datetime'] = pd.to_datetime(df['datetime'])
df['datetime'].dtype

df['year'] = df['datetime'].dt.year
df['month'] = df['datetime'].dt.month
df['day'] = df['datetime'].dt.day
df['hour'] = df['datetime'].dt.hour
print(df.head())
print(df.tail())
```

```
             datetime  season  holiday  workingday  weather  temp   atemp  humidity  windspeed  casual  registered  count  year  month  day  hour
0 2011-01-01 00:00:00       1        0           0        1  9.84  14.395        81        0.0       3          13     16  2011      1    1     0
1 2011-01-01 01:00:00       1        0           0        1  9.02  13.635        80        0.0       8          32     40  2011      1    1     1
2 2011-01-01 02:00:00       1        0           0        1  9.02  13.635        80        0.0       5          27     32  2011      1    1     2
3 2011-01-01 03:00:00       1        0           0        1  9.84  14.395        75        0.0       3          10     13  2011      1    1     3
4 2011-01-01 04:00:00       1        0           0        1  9.84  14.395        75        0.0       0           1      1  2011      1    1     4
                 datetime  season  holiday  workingday  weather   temp   atemp  humidity  windspeed  casual  registered  count  year  month  day  hour
10881 2012-12-19 19:00:00       4        0           1        1  15.58  19.695        50    26.0027       7         329    336  2012     12   19    19
10882 2012-12-19 20:00:00       4        0           1        1  14.76  17.425        57    15.0013      10         231    241  2012     12   19    20
10883 2012-12-19 21:00:00       4        0           1        1  13.94  15.910        61    15.0013       4         164    168  2012     12   19    21
10884 2012-12-19 22:00:00       4        0           1        1  13.94  17.425        61     6.0032      12         117    129  2012     12   19    22
10885 2012-12-19 23:00:00       4        0           1        1  13.12  16.665        66     8.9981       4          84     88  2012     12   19    23
```

converting the date time format to year month day and hour to check the time where booking is made more and on which day the customers where more.

In [35]:

```
time_span = df['datetime'].max() - df['datetime'].min()
time_span
```

Out[35]:

```
Timedelta('718 days 23:00:00')
```

In [32]:

```
cat_col = ['season', 'holiday', 'workingday', 'weather', 'year', 'month' , 'day', 'hour']

for cat in cat_col:
    df[cat] = df[cat].astype('category')

print(df.info())
print(print(df.nunique()))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  category
 2   holiday     10886 non-null  category
 3   workingday  10886 non-null  category
 4   weather     10886 non-null  category
 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
 12  year        10886 non-null  category
 13  month       10886 non-null  category
 14  day         10886 non-null  category
 15  hour        10886 non-null  category
dtypes: category(8), datetime64[ns](1), float64(3), int64(4)
memory usage: 767.6 KB
None
datetime      10886
season            4
holiday           2
workingday        2
weather           4
temp             49
atemp            60
humidity         89
windspeed        28
casual          309
registered      731
count           822
year              2
month            12
day              19
hour             24
dtype: int64
None
```

In [33]:

```
columns = df.columns
for column in columns:
    print(f"{column}: ", df[column].value_counts())
```

```
datetime:  datetime
2011-01-01 00:00:00    1
2012-05-01 21:00:00    1
2012-05-01 13:00:00    1
2012-05-01 14:00:00    1
2012-05-01 15:00:00    1
                      ..
2011-09-02 04:00:00    1
2011-09-02 05:00:00    1
2011-09-02 06:00:00    1
2011-09-02 07:00:00    1
2012-12-19 23:00:00    1
Name: count, Length: 10886, dtype: int64
season:  season
4    2734
2    2733
3    2733
1    2686
Name: count, dtype: int64
holiday:  holiday
0    10575
1      311
Name: count, dtype: int64
workingday:  workingday
1    7412
0    3474
Name: count, dtype: int64
weather:  weather
1    7192
2    2834
3     859
4       1
Name: count, dtype: int64
temp:  temp
14.76    467
26.24    453
28.70    427
13.94    413
18.86    406
22.14    403
25.42    403
16.40    400
22.96    395
27.06    394
24.60    390
12.30    385
21.32    362
17.22    356
13.12    356
29.52    353
10.66    332
18.04    328
20.50    327
30.34    299
9.84     294
15.58    255
9.02     248
31.16    242
8.20     229
27.88    224
23.78    203
32.80    202
11.48    181
19.68    170
6.56     146
33.62    130
```

```
5.74     107
7.38     106
31.98     98
34.44     80
35.26     76
4.92      60
36.90     46
4.10      44
37.72     34
36.08     23
3.28      11
0.82       7
38.54      7
39.36      6
2.46       5
1.64       2
41.00      1
Name: count, dtype: int64
atemp:  atemp
31.060    671
25.760    423
22.725    406
20.455    400
26.515    395
16.665    381
25.000    365
33.335    364
21.210    356
30.305    350
15.150    338
21.970    328
24.240    327
17.425    314
31.820    299
34.850    283
27.275    282
32.575    272
11.365    271
14.395    269
29.545    257
19.695    255
15.910    254
12.880    247
13.635    237
34.090    224
12.120    195
28.790    175
23.485    170
10.605    166
35.605    159
9.850     127
18.180    123
36.365    123
37.120    118
9.090     107
37.880     97
28.030     80
7.575      75
38.635     74
6.060      73
39.395     67
6.820      63
8.335      63
18.940     45
40.150     45
40.910     39
```

```
5.305      25
42.425     24
41.665     23
3.790      16
4.545      11
3.030       7
43.940      7
2.275       7
43.180      7
44.695      3
0.760       2
1.515       1
45.455      1
Name: count, dtype: int64
humidity:  humidity
88    368
94    324
83    316
87    289
70    259
     ... 
8       1
10      1
97      1
96      1
91      1
Name: count, Length: 89, dtype: int64
windspeed:  windspeed
0.0000     1313
8.9981     1120
11.0014    1057
12.9980    1042
7.0015     1034
15.0013     961
6.0032      872
16.9979     824
19.0012     676
19.9995     492
22.0028     372
23.9994     274
26.0027     235
27.9993     187
30.0026     111
31.0009      89
32.9975      80
35.0008      58
39.0007      27
36.9974      22
43.0006      12
40.9973      11
43.9989       8
46.0022       3
56.9969       2
47.9988       2
51.9987       1
50.0021       1
Name: count, dtype: int64
casual:  casual
0      986
1      667
2      487
3      438
4      354
      ... 
332      1
361      1
```

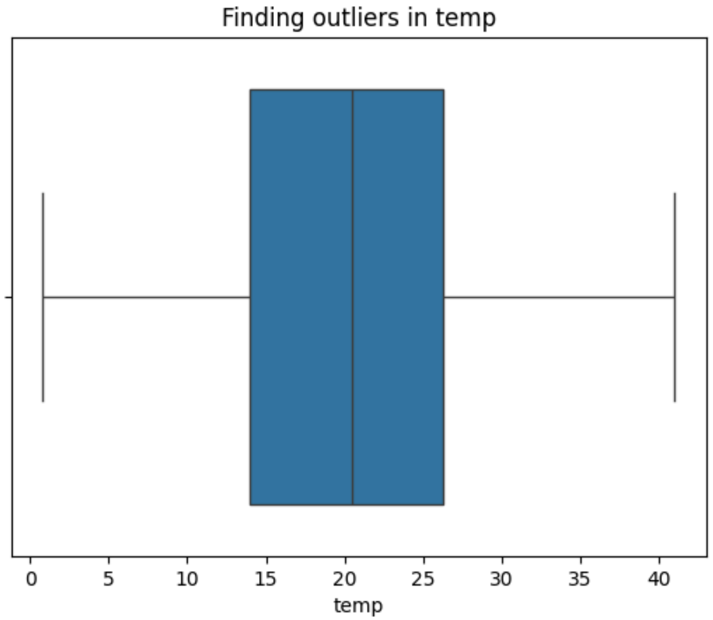
```
356      1
331      1
304      1
Name: count, Length: 309, dtype: int64
registered:  registered
3      195
4      190
5      177
6      155
2      150
      ...
570      1
422      1
678      1
565      1
636      1
Name: count, Length: 731, dtype: int64
count:  count
5      169
4      149
3      144
6      135
2      132
      ...
801      1
629      1
825      1
589      1
636      1
Name: count, Length: 822, dtype: int64
year:  year
2012    5464
2011    5422
Name: count, dtype: int64
month:  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
day:  day
1     575
9     575
17    575
5     575
16    574
15    574
14    574
13    574
19    574
8     574
7     574
4     574
2     573
12    573
3     573
6     572
10    572
```

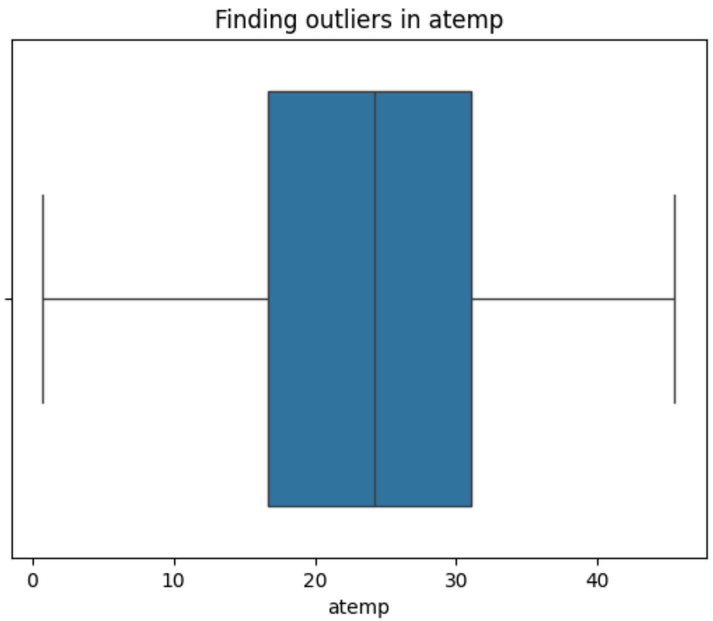
```
11    568
18    563
Name: count, dtype: int64
hour:  hour
12    456
13    456
22    456
21    456
20    456
19    456
18    456
17    456
16    456
15    456
14    456
23    456
11    455
10    455
9     455
8     455
7     455
6     455
0     455
1     454
5     452
2     448
4     442
3     433
Name: count, dtype: int64
```

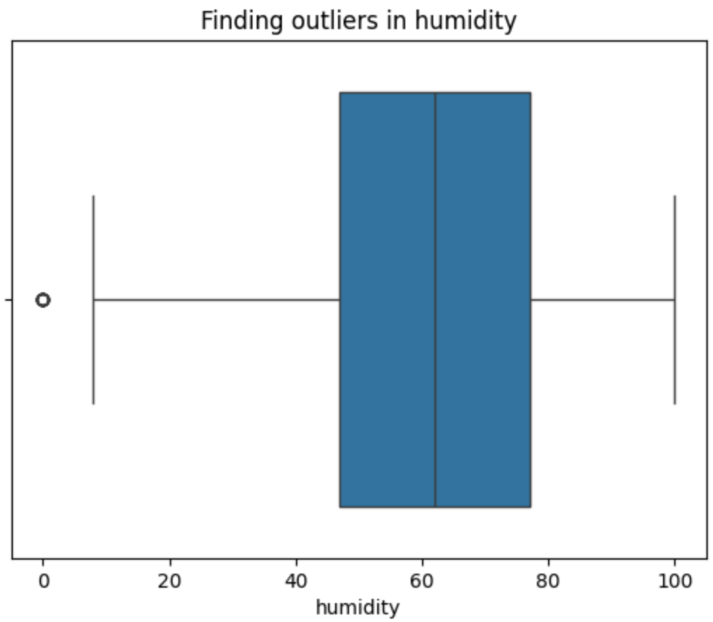
Outlier Detection

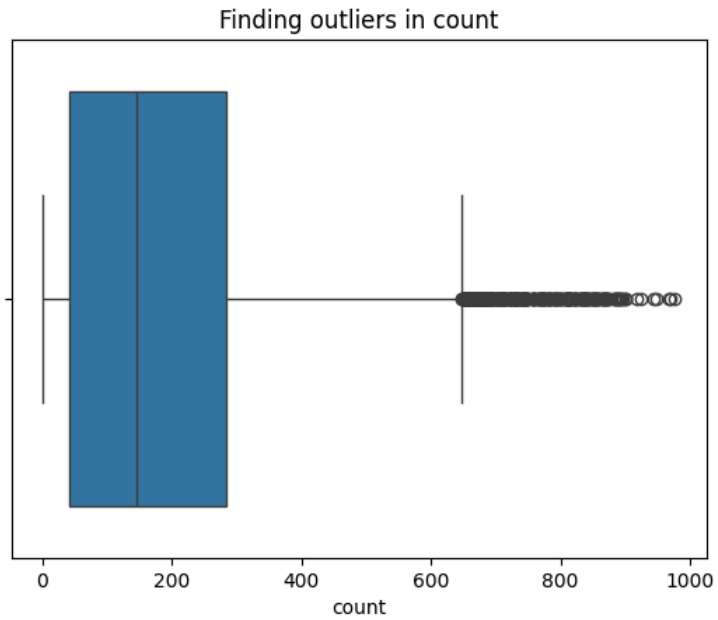
In [34]:

```
columns = [ 'temp', 'atemp', 'humidity', 'count']
for column in columns:
    sns.boxplot(x=column, data=df)
    plt.title(f'Finding outliers in {column}')
    plt.show()
```


Finding outliers in temp
0
5
10
15
20
25
30
35
40
temp


Finding outliers in atemp
0
10
20
30
40
atemp





In [93]:
```
count_25 = df["count"].quantile(0.25)
count_75 = df["count"].quantile(0.75)
count_median = df["count"].median()

lower_limit = count_25 - 1.5*(count_75 - count_25)
upper_limit = count_75 + 1.5*(count_75 - count_25)

print(f'Lower limit: {lower_limit}\nUpper limit: {upper_limit}\nMedian: {count_median}')
# print(len(df[df['Income']>upper_limit])
print(f"Outliers: {round((len(df[df['count']>upper_limit])/len(df))*100,2)}%")
```

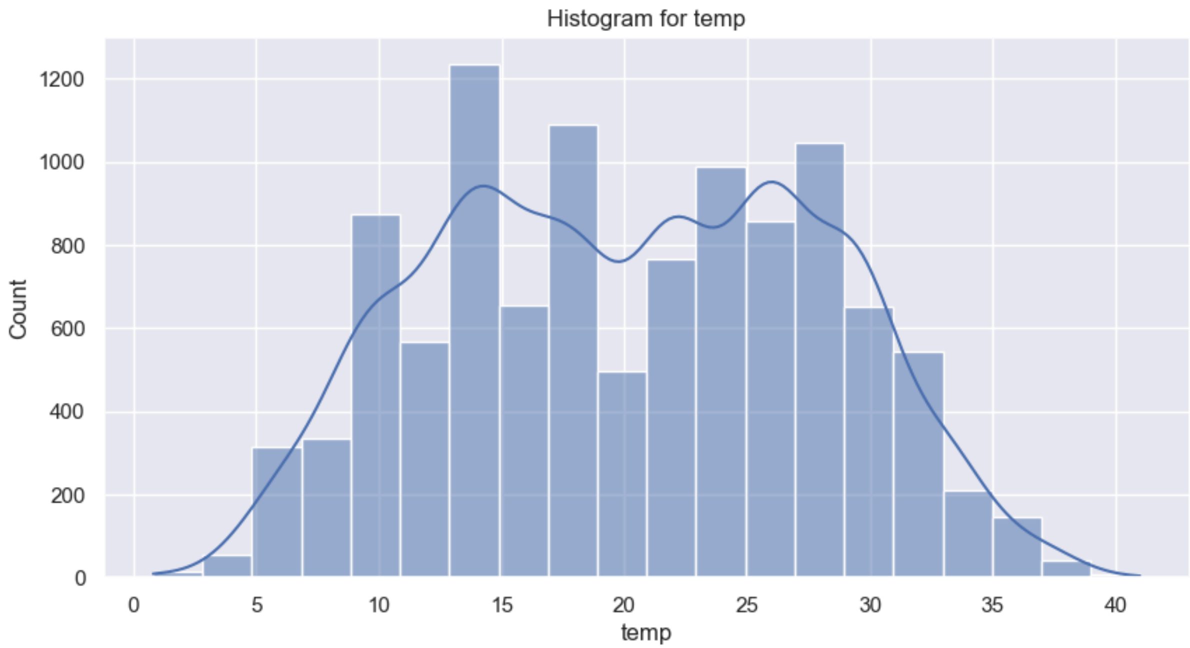
```
Lower limit: -321.0
Upper limit: 647.0
Median: 145.0
Outliers: 2.76%
```

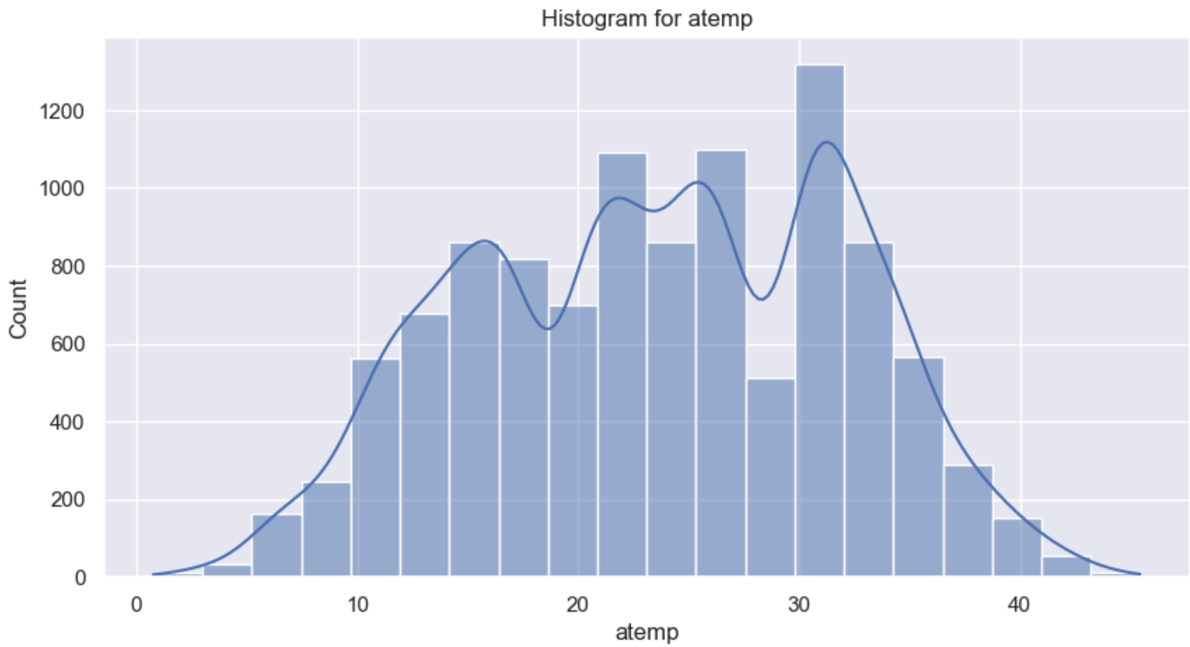
Numerical column like temp, atemp and humidity doesnt have so much outlier

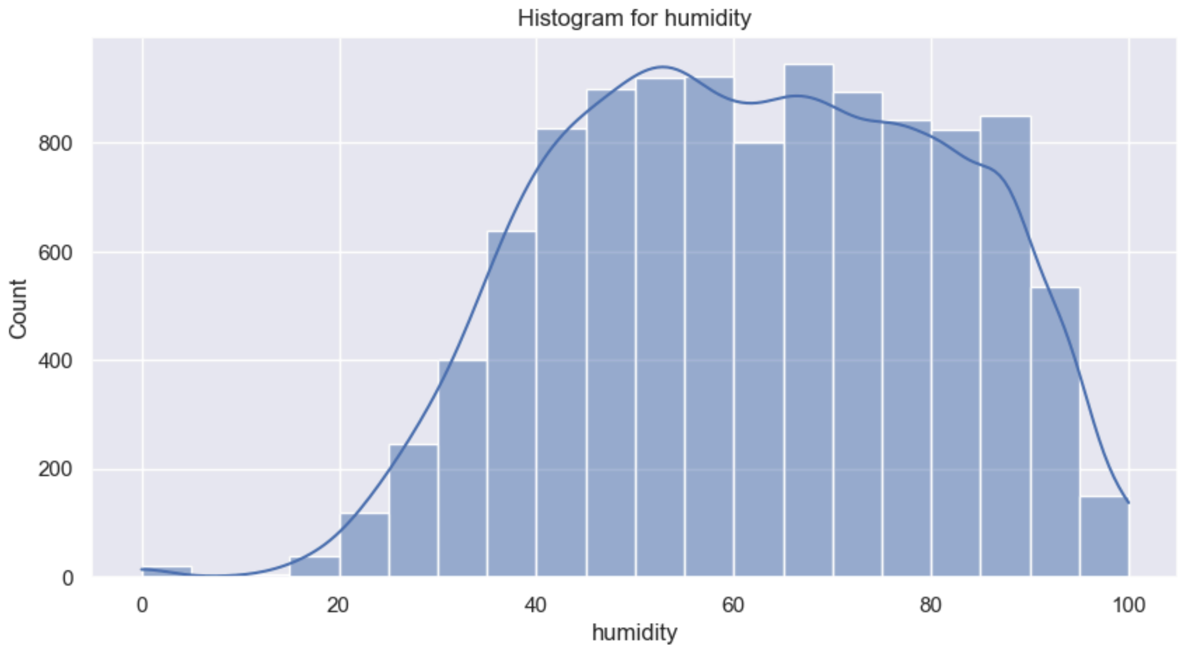
count column has 2.76% value in total as outliers, but it cannot be avoided to check which day lot of bookings happened.

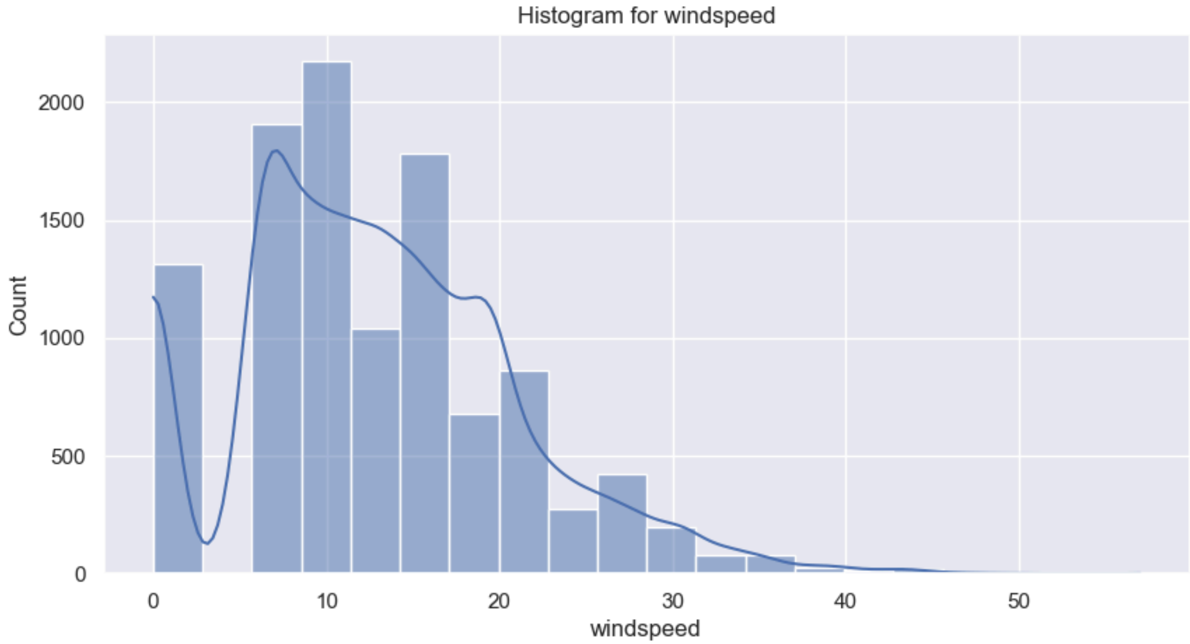
Univaria te Analysis

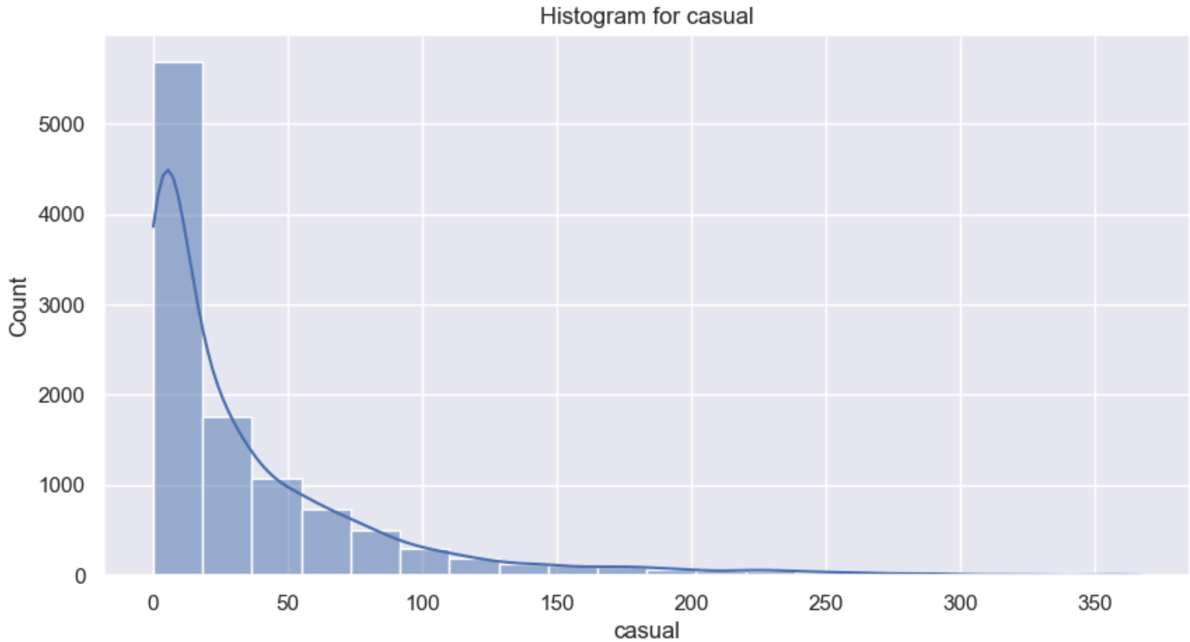
In [ ]:
```
num_col = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']

for column in num_col:
    # plt.figure(figsize=(10, 5))
    # value_counts = df[column].value_counts().sort_index()
    # plt.bar(x=value_counts.index, height=value_counts.values, color='skyblue')
    # plt.xlabel(column)
    # plt.ylabel('Count')
    # plt.title(f'Count of {column}')
    # plt.show()
    plt.figure(figsize=(10, 5))
    sns.set(style="darkgrid")

    sns.histplot(df[column], bins=20, kde=True)
    plt.title(f'Histogram for {column}')
```

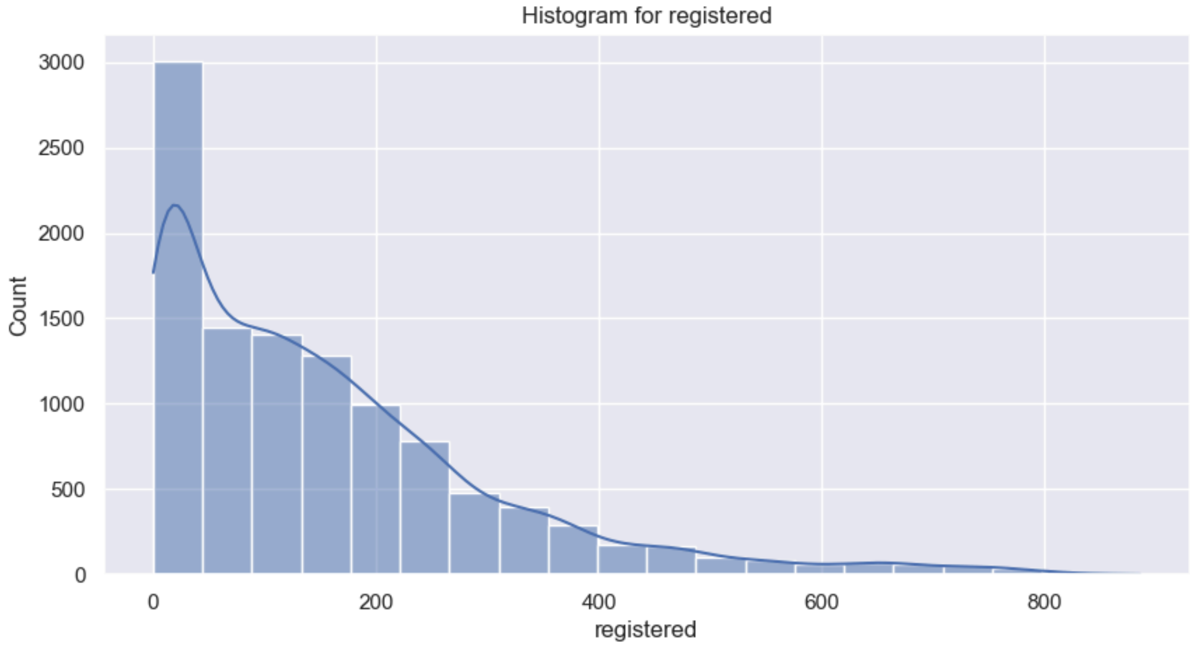


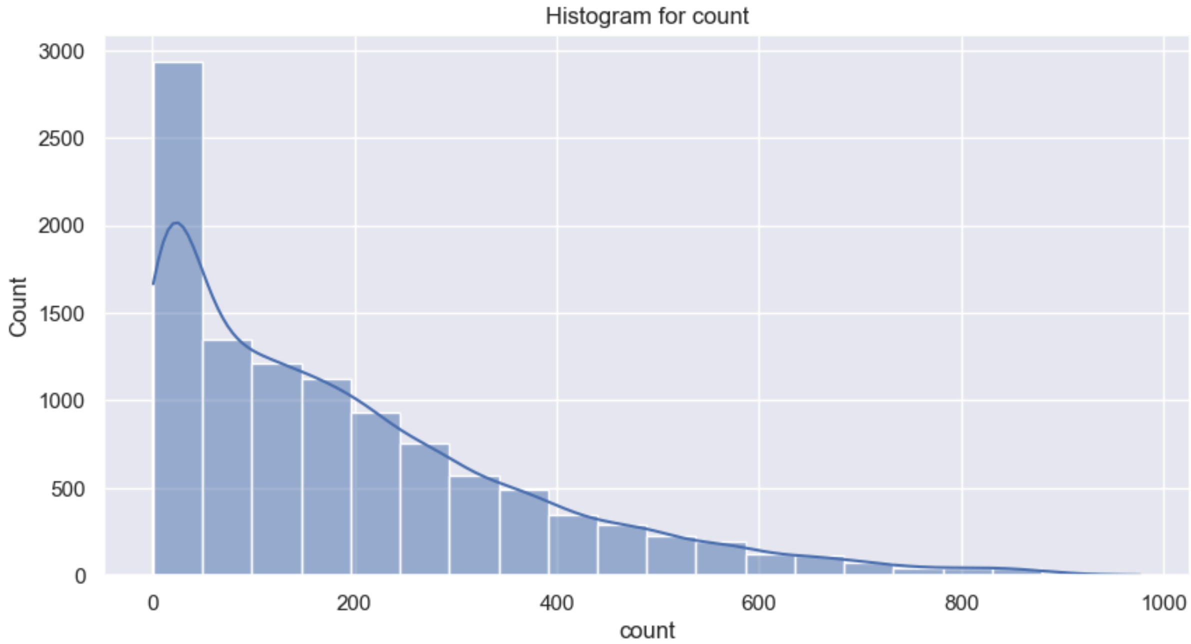

Histogram for atemp
1200
1000
800
600
400
200
0
Count
0
10
20
30
40
atemp


Histogram for humidity
800
600
400
200
0
Count
0
20
40
60
80
100
humidity


Histogram for windspeed
Count
windspeed
0
500
1000
1500
2000
0
10
20
30
40
50

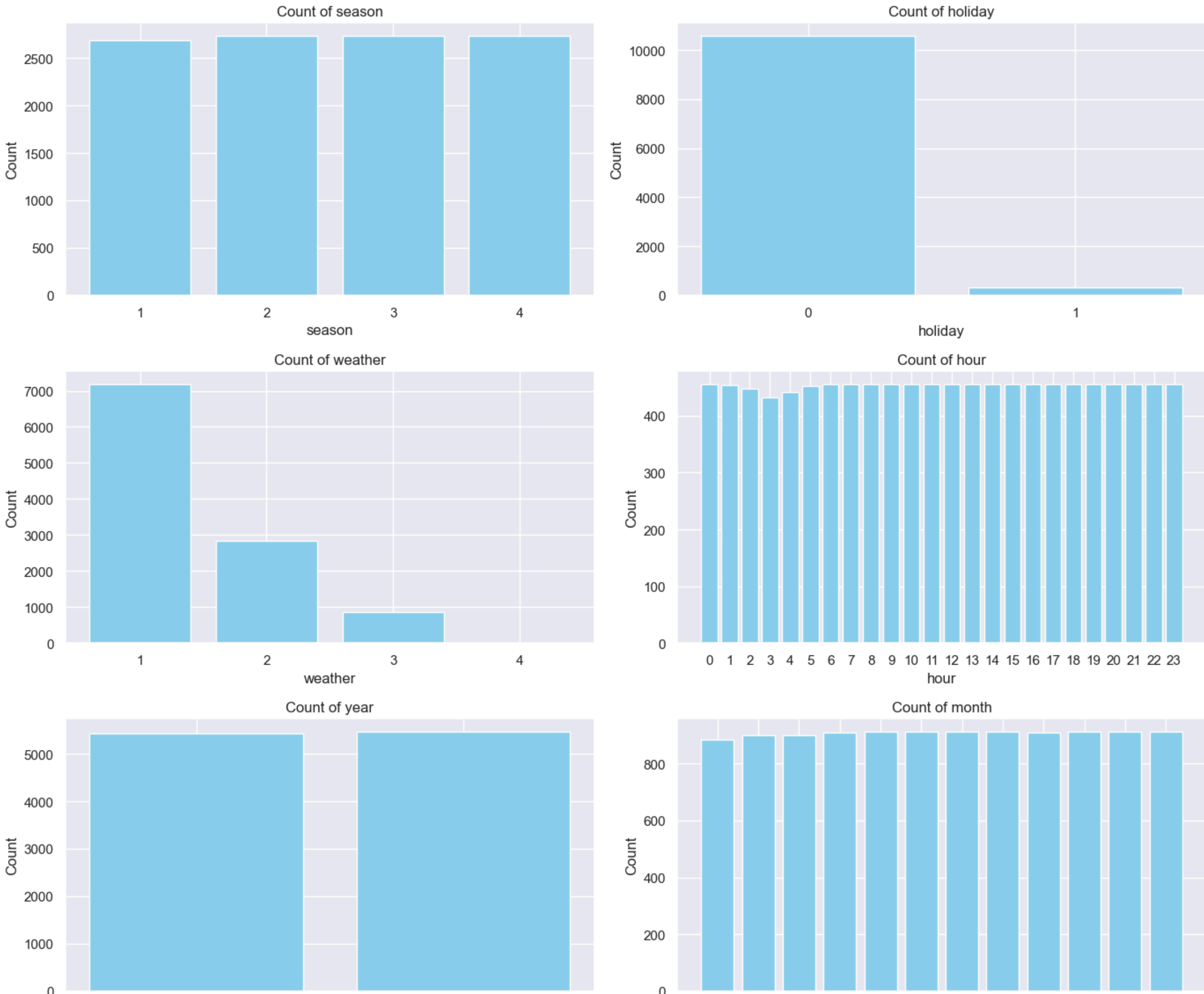

Histogram for casual
Count
casual
0
1000
2000
3000
4000
5000
0
50
100
150
200
250
300
350

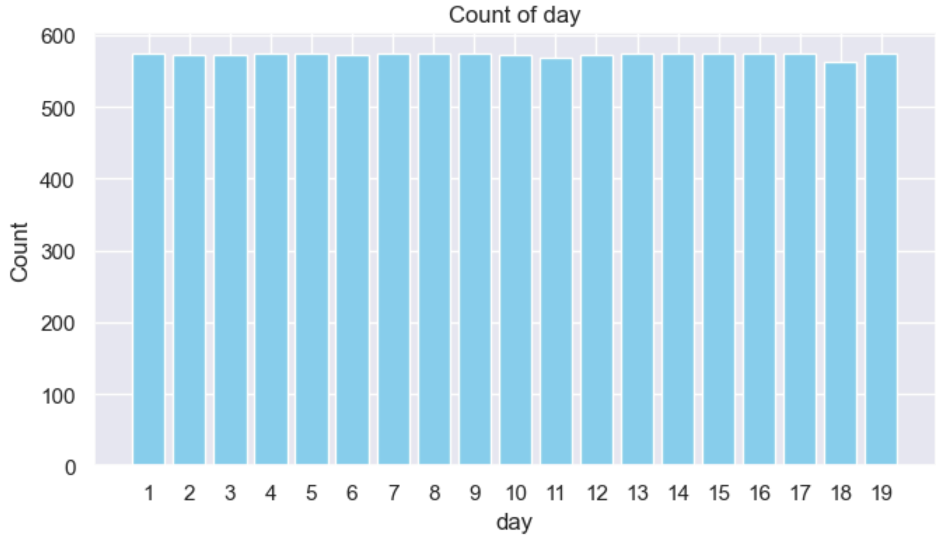


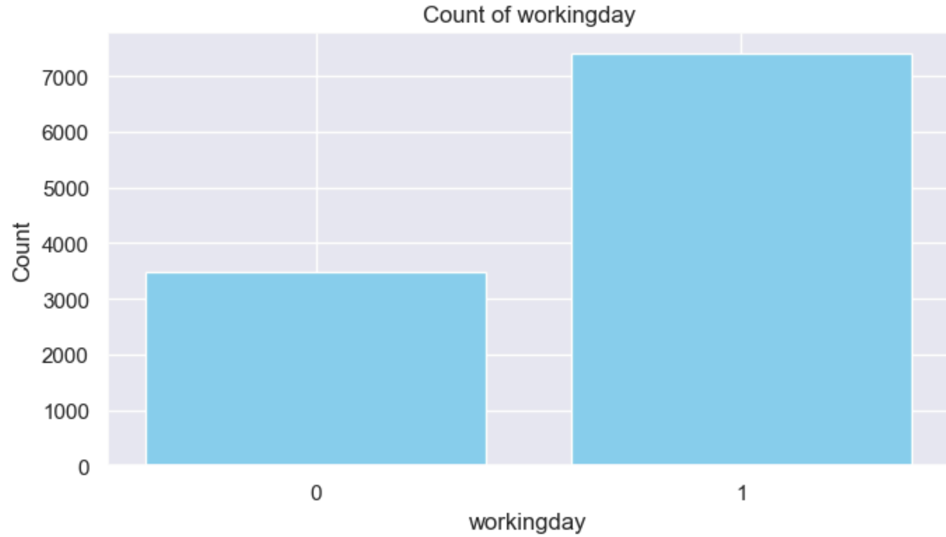


In [50]:
```
cat_col = ['season', 'holiday', 'weather', 'hour', 'year', 'month', 'day', 'workingday']

# Create a 4x2 grid of subplots
fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(14, 16))
axes = axes.flatten()

for idx, column in enumerate(cat_col):
    value_counts = df[column].value_counts().sort_index()
```

```python
    # Convert index to string to force categorical x-axis
    x_labels = value_counts.index.astype(str)

    axes[idx].bar(x=x_labels, height=value_counts.values, color='skyblue')
    axes[idx].set_xlabel(column)
    axes[idx].set_ylabel('Count')
    axes[idx].set_title(f'Count of {column}')

# Remove any unused axes (if fewer plots than subplots)
for i in range(len(cat_col), len(axes)):
    fig.delaxes(axes[i])

plt.tight_layout()
plt.show()
```


Count of season
Count
season
1
2
3
4
Count of holiday
Count
holiday
0
1
Count of weather
Count
weather
1
2
3
4
Count of hour
Count
hour
Count of year
Count
Count of month
Count

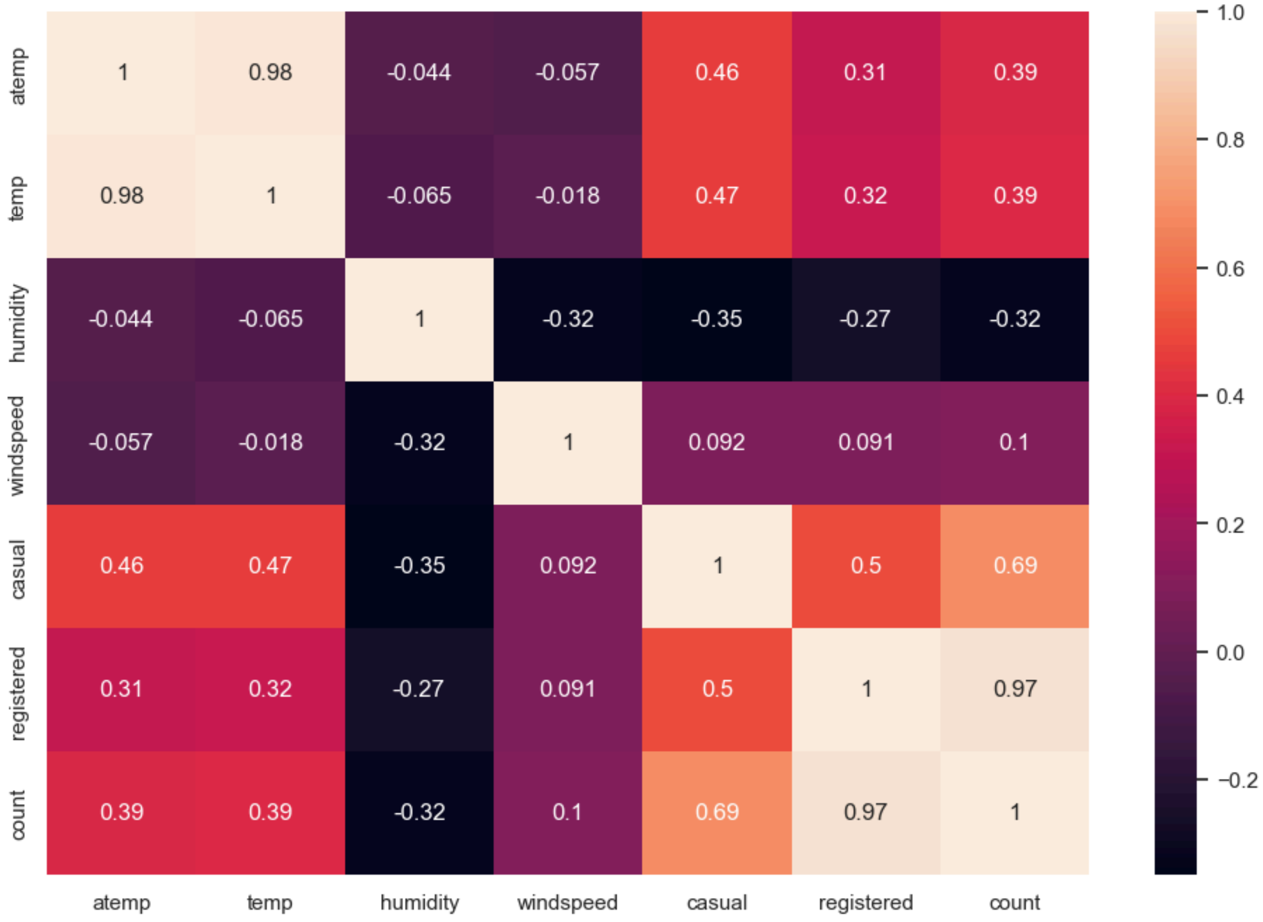




In [54]:

```python
correlation_matrix = df[["atemp", "temp", "humidity", "windspeed", "casual", "registered", "count"]].corr()
correlation_df = pd.DataFrame(correlation_matrix)
print(correlation_df)

plt.figure(figsize = (12, 8))
sns.heatmap(correlation_matrix, annot = True)
plt.show()
```

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



Correlation Analysis

Atemp:

Strong positive correlation with 'temp' (0.98), indicating a close relationship. Moderate positive correlation with 'casual' (0.46) and 'registered' (0.31). Positive correlation with 'count' (0.39), suggesting a relationship with overall bike rentals.

Temp (Temperature):

Highly correlated with 'atemp' (0.98), indicating a strong connection. Moderate positive correlation with 'casual' (0.47) and 'registered' (0.32). Positive correlation with 'count' (0.39), showing a relationship with overall bike rentals.

Humidity:

Weak negative correlation with 'atemp' (-0.04) and 'temp' (-0.06). Moderate negative correlation with 'casual' (-0.35), 'registered' (-0.27), and 'count' (-0.32). Indicates a tendency for fewer bike rentals during higher humidity.

Windspeed:

Weak negative correlation with 'atemp' (-0.06) and 'temp' (-0.02). Weak positive correlation with 'casual' (0.09), 'registered' (0.09), and 'count' (0.10). Suggests a subtle influence on bike rentals with increasing wind speed.

Casual (Casual Bike Rentals):

Strong positive correlation with 'atemp' (0.46) and 'temp' (0.47). Moderate negative correlation with 'humidity' (-0.35) and positive correlation with 'windspeed' (0.09). Highly correlated with 'registered' (0.50) and 'count' (0.69), indicating a significant impact on overall rentals.

Registered (Registered Bike Rentals):

Positive correlation with 'atemp' (0.31) and 'temp' (0.32). Negative correlation with 'humidity' (-0.27) and positive correlation with 'windspeed' (0.09). Highly correlated with 'casual' (0.50) and 'count' (0.97), emphasizing a substantial impact on overall rentals.

Count (Total Bike Rentals):

Positive correlation with 'atemp' (0.39), 'temp' (0.39), and 'casual' (0.69). Negative correlation with 'humidity' (-0.32). Highly correlated with 'registered' (0.97), emphasizing the joint impact of casual and registered rentals on the overall count.

Hypothesis Testing

In [67]:
```
from scipy.stats import norm,t
from scipy.stats import poisson, expon,geom, ttest_1samp, ttest_ind,ttest_ind_from_stats
from scipy.stats import shapiro, levene, kruskal, chi2, chi2_contingency
from statsmodels.graphics.gofplots import qqplot
```

Demand of bicycles on rent is the same on Weekdays & Weekends

Since we have two independent samples, we can go with Two Sample Independent T-Test.

Assumptions of Two Sample Independent T-Test :

The data should be normall distributed

variances of the two groups are equal

Let the Confidence interval be 95%, so siginificance (alpha) is 0.05

In [55]:
```
#Filtering count based on working day
working_day_count= df.loc[df["workingday"]==1,"count"]
non_working_day_count=df.loc[df["workingday"]==0,"count"]
```

In [56]:
```
working_day_count.mean(), working_day_count.std()
```

Out[56]:
```
(193.01187263896384, 184.5136590421481)
```

In [57]:
```
non_working_day_count.mean(), non_working_day_count.std()
```

Out[57]:
```
(188.50662061024755, 173.7240153250003)
```

Ho : mean of working day and non working day is same : mu1 = mu2

Ha : mean of working day is higher than non working day : mu1 > mu2

In [59]:
```
#Let us set siginificance level 0.05, confidence level 95%
alpha=0.05

#Let we do t-test for 2 samples and find test_statistics and p-value
test_statistic, p_value = ttest_ind(working_day_count,non_working_day_count, alternative="greater")
test_statistic, p_value
```

Out[59]:
```
(1.2096277376026694, 0.11322402113180674)
```

In [60]:
```
if p_value < alpha:
    print("Reject Null Hypothesis Ho")
```

```
else:
    print("Fail to Reject Null Hypothesis Ho")
```

```
Fail to Reject Null Hypothesis Ho
```

We have considered a confidence level of 95% in the Test.

The 2 Sample T-Test between the count attributes of the working day and the non-working day has been carried out and We found from the 2 Sample T-test that the means of both samples have no statistically significant difference.

In [63]:
```
weather_1 = df.loc[df["weather"]==1,"count"]
weather_2 = df.loc[df["weather"]==2,"count"]
weather_3 = df.loc[df["weather"]==3,"count"]
weather_4 = df.loc[df["weather"]==4,"count"]
print(weather_4)
```

```
5631    164
Name: count, dtype: int64
```

Only single value is there with weather category 4 so, We will not consider this category for ANOVA Test

We will do shapiro Test for checking whether our sample follows Gaussian Distribution or not

Null and Alternate Hypothesis for Shapiro Test

H0: The sample follows Gaussian Distribution

Ha: The sample does not follow Gaussian Distribution

In [64]:
```
#Let us set siginificance level 0.05, confidence level 95%
alpha=0.05
test_statistics, p_value = shapiro(weather_1)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

```
p-value: 0.0
Reject Null Hypotheis, Sample does not follow Gaussian Distribution
```

```
C:\Users\salsa\AppData\Roaming\Python\Python311\site-packages\scipy\stats\_axis_nan_policy.py:573: UserWarning: scipy.stats.shapiro: For N > 5000, computed p-value may not be accurate. Current N is 7192.
  res = hypotest_fun_out(*samples, **kwds)
```

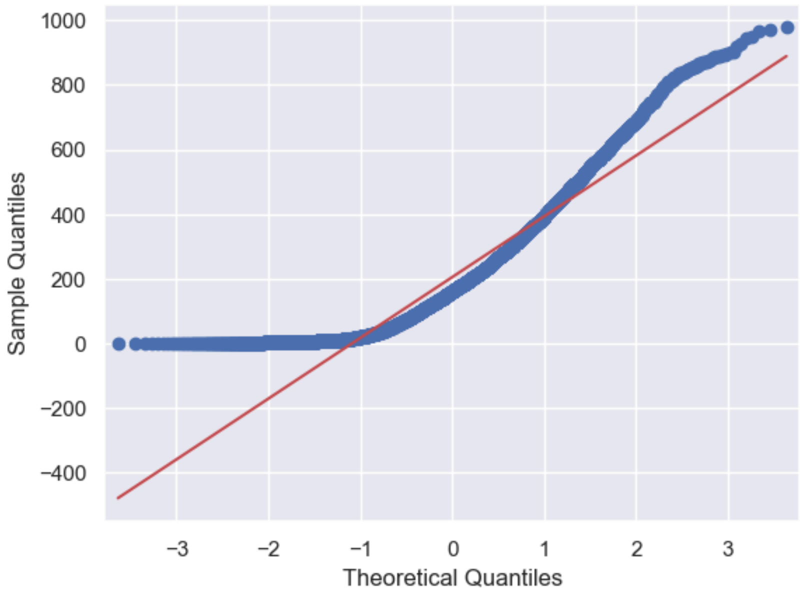
In [65]:
```
test_statistics, p_value = shapiro(weather_2)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

```
p-value: 0.0
Reject Null Hypotheis, Sample does not follow Gaussian Distribution
```

In [66]:
```
test_statistics, p_value = shapiro(weather_3)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

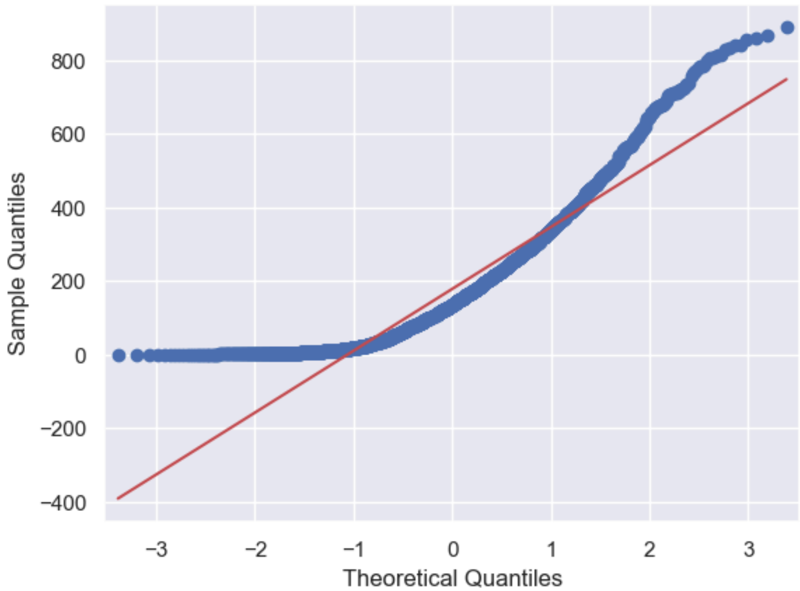
```
p-value: 0.0
Reject Null Hypotheis, Sample does not follow Gaussian Distribution
```

In [68]:
```
#Let's check for normality based on q-q plot
qqplot(weather_1,line="s")
plt.show()
```

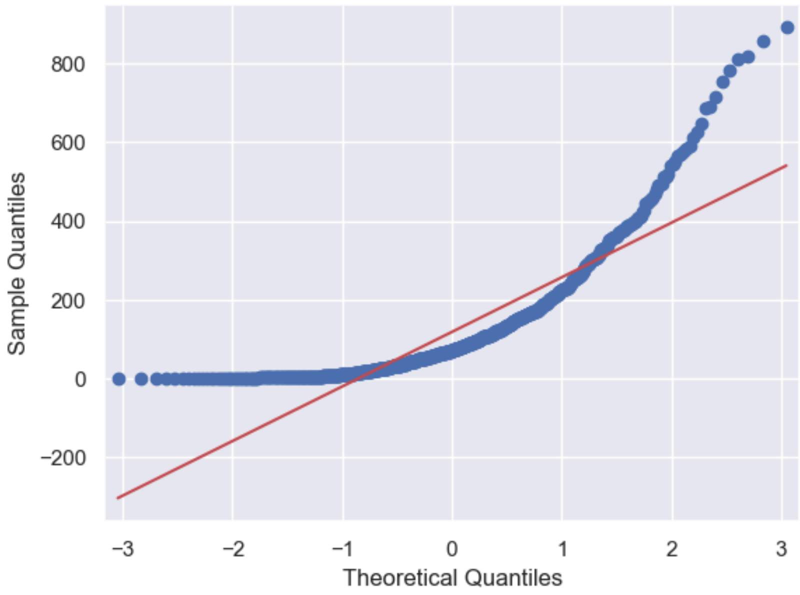


Here Plot not matching with straight line so based on that we can say that sample does not follow normal distribution

In [69]:
```
qqplot(weather_2,line="s")
plt.show()
```



In [70]:
```
qqplot(weather_3,line="s")
plt.show()
```



We will do levene test to check whether variance of the samples are same or not

Null Hypothesis and Alternate Hypothesis for Levene Test

H0: Variances of the samples are same

Ha: Variances of the samples are not same

In [71]:
```
#Let us set siginificance level 0.05, confidence level 95%
alpha=0.05

#p-value calculation
test_statistics, p_value=levene(weather_1,weather_2, weather_3)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, Variances of the samples are not same")
else:
    print("Fail to Reject Null Hypothesis, Variances of the samples are same")
```

```
p-value: 0.0
Reject Null Hypotheis, Variances of the samples are not same
```

p-value: 0.0

Reject Null Hypotheis, Variances of the samples are not same

As we have done shapiro and Q-Q Plot for checking Normality and Levene Test for checking Variance.

We have found that Samples do not follow Gaussian Distribution and do not have similar variance. So we will go for Kruskal-Wallis Test

Null and Alternate Hypothesis for Kruskal Wallis Test

H0: mean of total rental bikes of different weathers are same

Ha: mean of total rental bikes of different weathers are not same

In [72]:
```
#Let us set siginificance level 0.05, confidence level 95%
alpha=0.05
```

```
#p-value calculation
test_statistics,p_value=kruskal(weather_1,weather_2,weather_3)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, mean of total rental bikes of different weathers are not same")
else:
    print("Fail to Reject Null Hypothesis, mean of total rental bikes of different weathers are same")
```

```
p-value: 0.0
Reject Null Hypotheis, mean of total rental bikes of different weathers are not same
```

In [74]:
```
#Filtering count based on weather category
season_1 = df.loc[df["season"]==1,"count"]
season_2 = df.loc[df["season"]==2,"count"]
season_3 = df.loc[df["season"]==3,"count"]
season_4 = df.loc[df["season"]==4,"count"]
```

We will do shapiro Test for checking whether our sample follows Gaussian Distribution or not

Null and Alternate Hypothesis for Shapiro Test

H0: The sample follows Gaussian Distribution

Ha: The sample does not follow Gaussian Distribution

In [75]:
```
#Let us set siginificance level 0.05, confidence level 95%
alpha=0.05

#p-value calculation
test_statistics, p_value = shapiro(season_1)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

```
p-value: 0.0
Reject Null Hypotheis, Sample does not follow Gaussian Distribution
```

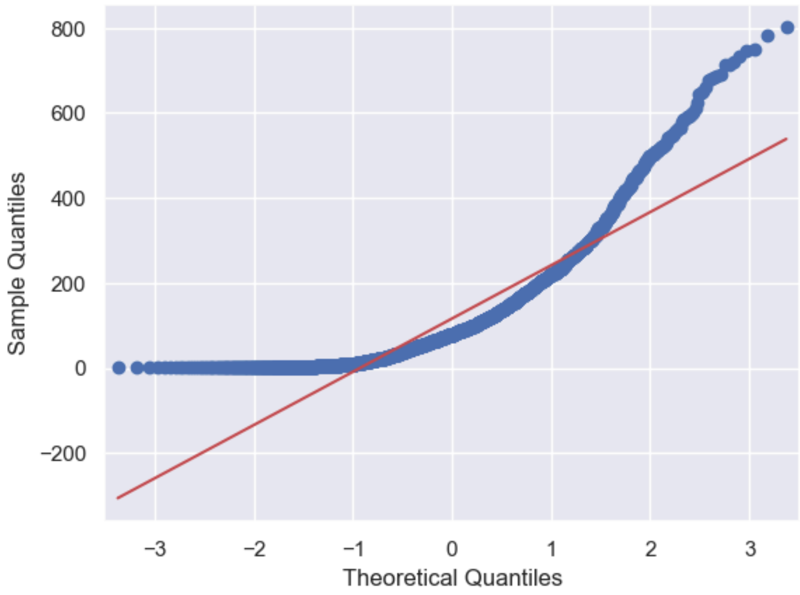
In [76]:
```
test_statistics, p_value = shapiro(season_2)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

```
p-value: 0.0
Reject Null Hypotheis, Sample does not follow Gaussian Distribution
```

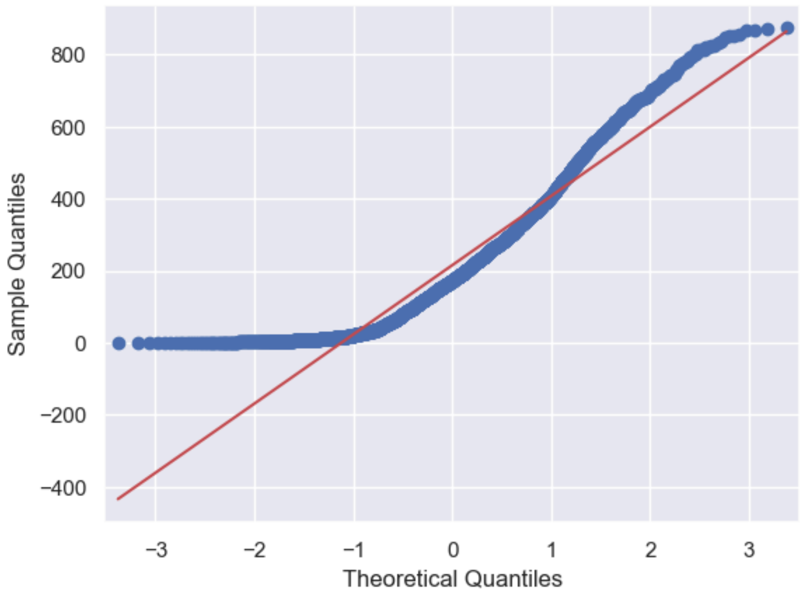
In [77]:
```
test_statistics, p_value = shapiro(season_3)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

```
p-value: 0.0
Reject Null Hypotheis, Sample does not follow Gaussian Distribution
```

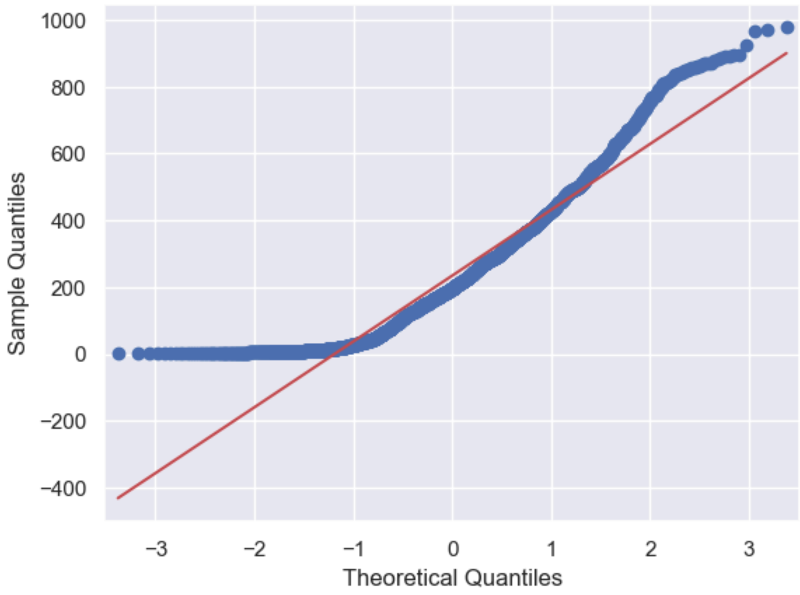
In [78]:
```
#Let's check for normality based on q-q plot
qqplot(season_1,line="s")
plt.show()
```



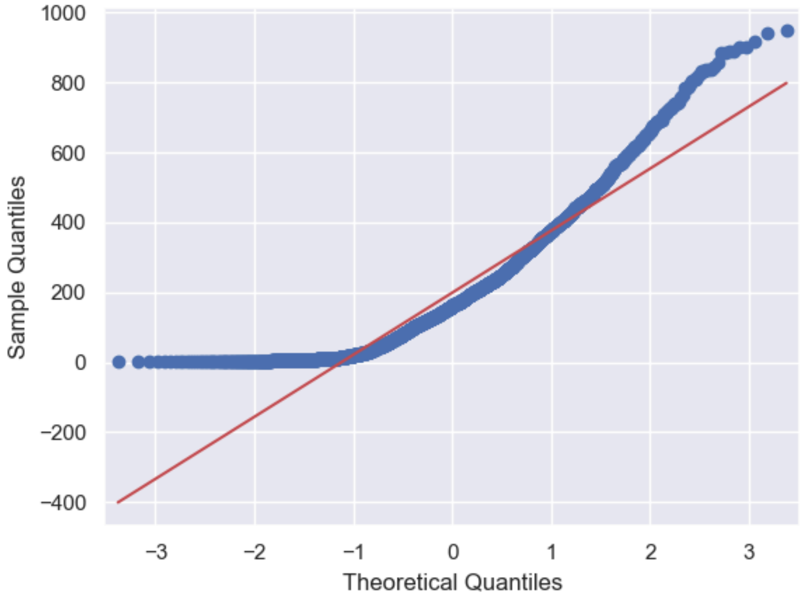
In [79]:
```
qqplot(season_2,line="s")
plt.show()
```



In [80]:
```
qqplot(season_3,line="s")
plt.show()
```



In [81]:
```
qqplot(season_4,line="s")
plt.show()
```



We will do levene test to check whether variance of the samples are same or not

Null Hypothesis and Alternate Hypothesis for Levene Test

H0: Variances of the samples are same

Ha: Variances of the samples are not same

In [82]:
```
#Let us set siginificance level 0.05, confidence level 95%
alpha=0.05

#p-value calculation
test_statistics, p_value=levene(season_1, season_2, season_3, season_4)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, Variances of the samples are not same")
else:
    print("Fail to Reject Null Hypothesis, Variances of the samples are same")
```

```
p-value: 0.0
Reject Null Hypotheis, Variances of the samples are not same
```

As we have done shapiro and Q-Q Plot for checking Normality and Levene Test for checking Variance.

We have found that Samples do not follow Gaussian Distribution and do not have similar variance. So we will go for Kruskal-Wallis Test

Null and Alternate Hypothesis for Kruskal Wallis Test

H0: mean of total rental bikes of different seasons are same

Ha: mean of total rental bikes of different seasons are not same

In [83]:
```
test_statistics,p_value=kruskal(season_1, season_2, season_3, season_4)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, mean of total rental bikes of different seasons are not same")
else:
    print("Fail to Reject Null Hypothesis, mean of total rental bikes of different seasons are same")
```

```
p-value: 0.0
Reject Null Hypotheis, mean of total rental bikes of different seasons are not same
```

We have considered a confidence level of 95% in the Test.

For the assumptions testing like the shapiro-wilk test, q-q plot, and levene test has also been done in the Jupyter Notebook.

As samples fail for normality tests and variance tests, we have carried out Kruskal Wallis Test.

From the Kruskal Walis Test, It can be said that the Means of total rental bikes for different weathers has a statistically significant difference.

From the Kruskal Walis Test, It can be said that the Means of total rental bikes for different seasons has a statistically significant difference.

Chi-square Test

In [84]:
```
#Creating Contingency table between categorical attributes weather and season
ws= pd.crosstab(df["weather"], df["season"])
ws
```

Out[84]:

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

In [85]:
```
#we can not do chi-square test as minimum frequency to run chi-square test is 5
ws.loc[1:3,:]
```

Out[85]:

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

Here For Chi-Square Test between weather and Season

Null and Alternate Hypothesis

H0: Seasons and weather are independent

Ha: Seasons and weather are dependent on each other

In [87]:
```
test_statistics,p_value, dof, exp=chi2_contingency(ws.loc[1:3,:])
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, Seasons and weather are dependent on each other")
else:
    print("Fail to Reject Null Hypothesis, Seasons and weather are independent")
```

```
p-value: 0.0
Reject Null Hypotheis, Seasons and weather are dependent on each other
```

We have considered a confidence level of 95% in the Test.

From the Chi-Square Test, We can say that weather and season are depended on each other.

Business Insights

Weather and seasons are dependent on each other.

Total rental bikes are depended on the weather. the mean value for the total rental bikes for the weather 1st category is high compared to others.

Total rental bikes are also depended on the seasons. the mean value for total rental bikes for fall is higher compared to other, during spring there is the lowest number of users.

There is no statistical difference in the mean of the total rental bikes on working days and non-working days

Most days in the city are of the weather of 1st category.

Temperature and total rental bikes are correlated and humidity and total rental bikes are negatively correlated.

casual users and total rental bikes are less correlated compared to registered users and total rental bikes.

Recommendations

During spring, Yulu should provide some discounts and offers to increase the use of rental bikes.
During weather of rain, The mean of total rental bikes is lower than others. As Yulu provides bike services, customers can't use it in rainy times. so Yulu should provide some roofs or cab services during this weather.

As humidity increases the total number of rental bikes decreases, so, Yulu should provide benefits during these humid days.

Yulu can increase the use of rental bikes by providing some city tour offers, events, or campaigns during non-working days.

Yulu can convert its casual users to registered users by providing some discounts or registration offers to convert casual users to registered users.

As mostly there is clear weather, Yulu should focus on the increase in total rental bikes during clear weather days.