https://drive.google.com/file/d/1mG9Wl87Le0EH3cvmEwkGb1HYMsm44QgI/view?usp=sharing (https://drive.google.com/file/d/1mG9Wl87Le0EH3cvmEwkGb1HYMsm44QgI/view?usp=sharing)

In [1]:
```
!gdown 1mG9Wl87Le0EH3cvmEwkGb1HYMsm44QgI
```

```
Downloading...
From: https://drive.google.com/uc?id=1mG9Wl87Le0EH3cvmEwkGb1HYMsm44QgI (https://drive.google.com/uc?id=1mG9Wl87Le0EH3cvmEwkGb1HYMsm44QgI)
To: C:\Users\91944\yulu.csv

  0%|          | 0.00/648k [00:00<?, ?B/s]
100%|##########| 648k/648k [00:00<00:00, 11.1MB/s]
```

In [10]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


yu=pd.read_csv("yulu.csv")
yu
```

Out[10]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	cas
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	
...	...	...	...	...	...	...	...	...	...	...
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	

10886 rows × 12 columns

In [5]:
```
yu.shape
```

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

In [6]:
```
yu.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  int64  
 2   holiday     10886 non-null  int64  
 3   workingday  10886 non-null  int64  
 4   weather     10886 non-null  int64  
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64  
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64  
 10  registered  10886 non-null  int64  
 11  count       10886 non-null  int64  
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

In [29]:
```
yu.describe()
```

Out[29]:

	season	holiday	workingday	weather	temp	atemp	
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	1088
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	6
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	1
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	4
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	6
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	7
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	1(

In [6]:
```
yu.describe(include=["object"])
```

Out[6]:

	datetime
count	10886
unique	10886
top	2011-01-01 00:00:00
freq	1

UNIQUE COLUMN VALUES

In [14]:
```
yu.season.unique()
```

Out[14]:
```
array([1, 2, 3, 4], dtype=int64)
```

In [15]:
```
yu.holiday.unique()
```

Out[15]:
```
array([0, 1], dtype=int64)
```

In [16]:
```
yu.workingday.unique()
```

Out[16]:
```
array([0, 1], dtype=int64)
```

In [17]:
```
yu.weather.unique()
```

Out[17]:
```
array([1, 2, 3, 4], dtype=int64)
```

In [19]:
```
yu.temp.unique()
```

Out[19]:
```
array([ 9.84,  9.02,  8.2 , 13.12, 15.58, 14.76, 17.22, 18.86, 18.04,
       16.4 , 13.94, 12.3 , 10.66,  6.56,  5.74,  7.38,  4.92, 11.48,
        4.1 ,  3.28,  2.46, 21.32, 22.96, 23.78, 24.6 , 19.68, 22.14,
       20.5 , 27.06, 26.24, 25.42, 27.88, 28.7 , 30.34, 31.16, 29.52,
       33.62, 35.26, 36.9 , 32.8 , 31.98, 34.44, 36.08, 37.72, 38.54,
        1.64,  0.82, 39.36, 41.  ])
```

In [20]:
```
yu.atemp.unique()
```

Out[20]:
```
array([14.395, 13.635, 12.88 , 17.425, 19.695, 16.665, 21.21 , 22.725,
       21.97 , 20.455, 11.365, 10.605,  9.85 ,  8.335,  6.82 ,  5.305,
        6.06 ,  9.09 , 12.12 ,  7.575, 15.91 ,  3.03 ,  3.79 ,  4.545,
       15.15 , 18.18 , 25.   , 26.515, 27.275, 29.545, 23.485, 25.76 ,
       31.06 , 30.305, 24.24 , 18.94 , 31.82 , 32.575, 33.335, 28.79 ,
       34.85 , 35.605, 37.12 , 40.15 , 41.665, 40.91 , 39.395, 34.09 ,
       28.03 , 36.365, 37.88 , 42.425, 43.94 , 38.635,  1.515,  0.76 ,
        2.275, 43.18 , 44.695, 45.455])
```

In [21]:
```
yu.humidity.unique()
```

Out[21]:
```
array([ 81,  80,  75,  86,  76,  77,  72,  82,  88,  87,  94, 100,  71,
        66,  57,  46,  42,  39,  44,  47,  50,  43,  40,  35,  30,  32,
        64,  69,  55,  59,  63,  68,  74,  51,  56,  52,  49,  48,  37,
        33,  28,  38,  36,  93,  29,  53,  34,  54,  41,  45,  92,  62,
        58,  61,  60,  65,  70,  27,  25,  26,  31,  73,  21,  24,  23,
        22,  19,  15,  67,  10,   8,  12,  14,  13,  17,  16,  18,  20,
        85,   0,  83,  84,  78,  79,  89,  97,  90,  96,  91], dtype=int64)
```

In [22]:
```
yu.windspeed.unique()
```

Out[22]:
```
array([ 0.    ,  6.0032, 16.9979, 19.0012, 19.9995, 12.998 , 15.0013,
        8.9981, 11.0014, 22.0028, 30.0026, 23.9994, 27.9993, 26.0027,
        7.0015, 32.9975, 36.9974, 31.0009, 35.0008, 39.0007, 43.9989,
       40.9973, 51.9987, 46.0022, 50.0021, 43.0006, 56.9969, 47.9988])
```

In [23]:
```
yu.casual.unique()
```

Out[23]:
```
array([  3,   8,   5,   0,   2,   1,  12,  26,  29,  47,  35,  40,  41,
        15,   9,   6,  11,   4,   7,  16,  20,  19,  10,  13,  14,  18,
        17,  21,  33,  23,  22,  28,  48,  52,  42,  24,  30,  27,  32,
        58,  62,  51,  25,  31,  59,  45,  73,  55,  68,  34,  38, 102,
        84,  39,  36,  43,  46,  60,  80,  83,  74,  37,  70,  81, 100,
        99,  54,  88,  97, 144, 149, 124,  98,  50,  72,  57,  71,  67,
        95,  90, 126, 174, 168, 170, 175, 138,  92,  56, 111,  89,  69,
       139, 166, 219, 240, 147, 148,  78,  53,  63,  79, 114,  94,  85,
       128,  93, 121, 156, 135, 103,  44,  49,  64,  91, 119, 167, 181,
       179, 161, 143,  75,  66, 109, 123, 113,  65,  86,  82, 132, 129,
       196, 142, 122, 106,  61, 107, 120, 195, 183, 206, 158, 137,  76,
       115, 150, 188, 193, 180, 127, 154, 108,  96, 110, 112, 169, 131,
       176, 134, 162, 153, 210, 118, 141, 146, 159, 178, 177, 136, 215,
       198, 248, 225, 194, 237, 242, 235, 224, 236, 222,  77,  87, 101,
       145, 182, 171, 160, 133, 105, 104, 187, 221, 201, 205, 234, 185,
       164, 200, 130, 155, 116, 125, 204, 186, 214, 245, 218, 217, 152,
       191, 256, 251, 262, 189, 212, 272, 223, 208, 165, 229, 151, 117,
       199, 140, 226, 286, 352, 357, 367, 291, 233, 190, 283, 295, 232,
       173, 184, 172, 320, 355, 326, 321, 354, 299, 227, 254, 260, 207,
       274, 308, 288, 311, 253, 197, 163, 275, 298, 282, 266, 220, 241,
       230, 157, 293, 257, 269, 255, 228, 276, 332, 361, 356, 331, 279,
       203, 250, 259, 297, 265, 267, 192, 239, 238, 213, 264, 244, 243,
       246, 289, 287, 209, 263, 249, 247, 284, 327, 325, 312, 350, 258,
       362, 310, 317, 268, 202, 294, 280, 216, 292, 304], dtype=int64)
```

In [24]:
```
yu.registered.unique()
```

Out[24]:
```
array([ 13,  32,  27,  10,   1,   0,   2,   7,   6,  24,  30,  55,  47,
        71,  70,  52,  26,  31,  25,  17,  16,   8,   4,  19,  46,  54,
        73,  64,  67,  58,  43,  29,  20,   9,   5,   3,  63, 153,  81,
        33,  41,  48,  53,  66, 146, 148, 102,  49,  11,  36,  92, 177,
        98,  37,  50,  79,  68, 202, 179, 110,  34,  87, 192, 109,  74,
        65,  85, 186, 166, 127,  82,  40,  18,  95, 216, 116,  42,  57,
        78,  59, 163, 158,  51,  76, 190, 125, 178,  39,  14,  15,  56,
        60,  90,  83,  69,  28,  35,  22,  12,  77,  44,  38,  75, 184,
       174, 154,  97, 214,  45,  72, 130,  94, 139, 135, 197, 137, 141,
       156, 117, 155, 134,  89,  80, 108,  61, 124, 132, 196, 107, 114,
       172, 165, 105, 119, 183, 175,  88,  62,  86, 170, 145, 217,  91,
       195, 152,  21, 126, 115, 223, 207, 123, 236, 128, 151, 100, 198,
       157, 168,  84,  99, 173, 121, 159,  93,  23, 212, 111, 193, 103,
       113, 122, 106,  96, 249, 218, 194, 213, 191, 142, 224, 244, 143,
       267, 256, 211, 161, 131, 246, 118, 164, 275, 204, 230, 243, 112,
       238, 144, 185, 101, 222, 138, 206, 104, 200, 129, 247, 140, 209,
       136, 176, 120, 229, 210, 133, 259, 147, 227, 150, 282, 162, 265,
       260, 189, 237, 245, 205, 308, 283, 248, 303, 291, 280, 208, 286,
       352, 290, 262, 203, 284, 293, 160, 182, 316, 338, 279, 187, 277,
       362, 321, 331, 372, 377, 350, 220, 472, 450, 268, 435, 169, 225,
       464, 485, 323, 388, 367, 266, 255, 415, 233, 467, 456, 305, 171,
       470, 385, 253, 215, 240, 235, 263, 221, 351, 539, 458, 339, 301,
       397, 271, 532, 480, 365, 241, 421, 242, 234, 341, 394, 540, 463,
       361, 429, 359, 180, 188, 261, 254, 366, 181, 398, 272, 167, 149,
       325, 521, 426, 298, 428, 487, 431, 288, 239, 453, 454, 345, 417,
       434, 278, 285, 442, 484, 451, 252, 471, 488, 270, 258, 264, 281,
       410, 516, 500, 343, 311, 432, 475, 479, 355, 329, 199, 400, 414,
       423, 232, 219, 302, 529, 510, 348, 346, 441, 473, 335, 445, 555,
       527, 273, 364, 299, 269, 257, 342, 324, 226, 391, 466, 297, 517,
       486, 489, 492, 228, 289, 455, 382, 380, 295, 251, 418, 412, 340,
       433, 231, 333, 514, 483, 276, 478, 287, 381, 334, 347, 320, 493,
       491, 369, 201, 408, 378, 443, 460, 465, 313, 513, 292, 497, 376,
       326, 413, 328, 525, 296, 452, 506, 393, 368, 337, 567, 462, 349,
       319, 300, 515, 373, 399, 507, 396, 512, 503, 386, 427, 312, 384,
       530, 310, 536, 437, 505, 371, 375, 534, 469, 474, 553, 402, 274,
       523, 448, 409, 387, 438, 407, 250, 459, 425, 422, 379, 392, 430,
       401, 306, 370, 449, 363, 389, 374, 436, 356, 317, 446, 294, 508,
       315, 522, 494, 327, 495, 404, 447, 504, 318, 579, 551, 498, 533,
       332, 554, 509, 573, 545, 395, 440, 547, 557, 623, 571, 614, 638,
       628, 642, 647, 602, 634, 648, 353, 322, 357, 314, 563, 615, 681,
       601, 543, 577, 354, 661, 653, 304, 645, 646, 419, 610, 677, 618,
       595, 565, 586, 670, 656, 626, 581, 546, 604, 596, 383, 621, 564,
       309, 360, 330, 549, 589, 461, 631, 673, 358, 651, 663, 538, 616,
       662, 344, 640, 659, 770, 608, 617, 584, 307, 667, 605, 641, 594,
       629, 603, 518, 665, 769, 749, 499, 719, 734, 696, 688, 570, 675,
       405, 411, 643, 733, 390, 680, 764, 679, 531, 637, 652, 778, 703,
       537, 576, 613, 715, 726, 598, 625, 444, 672, 782, 548, 682, 750,
       716, 609, 698, 572, 669, 633, 725, 704, 658, 620, 542, 575, 511,
       741, 790, 644, 740, 735, 560, 739, 439, 660, 697, 336, 619, 712,
       624, 580, 678, 684, 468, 649, 786, 718, 775, 636, 578, 746, 743,
       481, 664, 711, 689, 751, 745, 424, 699, 552, 709, 591, 757, 768,
       767, 723, 558, 561, 403, 502, 692, 780, 622, 761, 690, 744, 857,
       562, 702, 802, 727, 811, 886, 406, 787, 496, 708, 758, 812, 807,
       791, 639, 781, 833, 756, 544, 789, 742, 655, 416, 806, 773, 737,
       706, 566, 713, 800, 839, 779, 766, 794, 803, 788, 720, 668, 490,
```

```
       568, 597, 477, 583, 501, 556, 593, 420, 541, 694, 650, 559, 666,
       700, 693, 582], dtype=int64)
```

VALUE_COUNTS

In [30]:
```
yu["season"].value_counts()
```

Out[30]:
```
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
```

In [31]:
```
yu["holiday"].value_counts()
```

Out[31]:
```
0    10575
1      311
Name: holiday, dtype: int64
```

In [32]:
```
yu["workingday"].value_counts()
```

Out[32]:
```
1    7412
0    3474
Name: workingday, dtype: int64
```

In [33]:
```
yu["weather"].value_counts()
```

Out[33]:
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

In [36]:
```
yu["temp"].value_counts().head(10)
```

Out[36]:
```
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
Name: temp, dtype: int64
```

In [37]:
```
yu["atemp"].value_counts().head(10)
```

Out[37]:
```
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
Name: atemp, dtype: int64
```

In [38]:
```
yu["humidity"].value_counts().head(10)
```

Out[38]:
```
88    368
94    324
83    316
87    289
70    259
65    253
46    247
66    246
77    244
49    234
Name: humidity, dtype: int64
```

In [40]:
```
yu["windspeed"].value_counts().head(10)
```

Out[40]:
```
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
Name: windspeed, dtype: int64
```

In [41]:
```
yu["casual"].value_counts().head(10)
```

Out[41]:
```
0    986
1    667
2    487
3    438
4    354
5    332
6    269
8    250
7    250
9    230
Name: casual, dtype: int64
```

In [16]:
```
yu["registered"].value_counts().head(10)
```

Out[16]:
```
3     195
4     190
5     177
6     155
2     150
1     135
7     126
9     114
8     114
11     87
Name: registered, dtype: int64
```

In [21]:
```
yu["count"].value_counts().head(10)
```

Out[21]:
```
5     169
4     149
3     144
6     135
2     132
7     118
1     105
8      99
10     95
11     95
Name: count, dtype: int64
```

In [50]:
```
yu["date"].value_counts().head(10)
```

Out[50]:
```
2011-01-01    24
2012-04-18    24
2012-05-10    24
2012-05-09    24
2012-05-08    24
2012-05-07    24
2012-05-06    24
2012-05-05    24
2012-05-04    24
2012-05-03    24
Name: date, dtype: int64
```

In [49]:
```
yu["time"].value_counts().head(10)
```

Out[49]:
```
12:00:00    456
13:00:00    456
22:00:00    456
21:00:00    456
20:00:00    456
19:00:00    456
18:00:00    456
17:00:00    456
16:00:00    456
15:00:00    456
Name: time, dtype: int64
```

In [9]:
```
yu.isna().sum()
```

Out[9]:
```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

In [4]:
```
#converting object to datetime
yu["datetime"]=yu["datetime"].astype("datetime64[ns]")
```

In [15]:
```
yu.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 14 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         
 12  date        10886 non-null  datetime64[ns]
 13  time        10886 non-null  datetime64[ns]
dtypes: datetime64[ns](2), float64(3), int64(8), object(1)
memory usage: 1.2+ MB
```

In [11]:
```
yu[["date","time"]]=yu["datetime"].str.split(" ",expand=True)
```

In [13]:
```
yu["date"]=yu["date"].astype("datetime64[ns]")
```

In [14]:
```
yu["time"]=yu["time"].astype("datetime64[ns]")
```

In [17]:
```
yu.drop(columns=["datetime"],inplace=True)
```

In [48]:
```
yu[["workingday","weather","season","count","temp","atemp"]].corr()
```

Out[48]:

	workingday	weather	season	count	temp	atemp
workingday	1.000000	0.033772	-0.008126	0.011594	0.029966	0.024660
weather	0.033772	1.000000	0.008879	-0.128655	-0.055035	-0.055376
season	-0.008126	0.008879	1.000000	0.163439	0.258689	0.264744
count	0.011594	-0.128655	0.163439	1.000000	0.394454	0.389784
temp	0.029966	-0.055035	0.258689	0.394454	1.000000	0.984948
atemp	0.024660	-0.055376	0.264744	0.389784	0.984948	1.000000

In [49]:
```
yu.corr()
```

```
C:\Users\91944\AppData\Local\Temp\ipykernel_16848\3518043383.py:1: FutureWarn
ing: 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 specif
y the value of numeric_only to silence this warning.
  yu.corr()
```

Out[49]:

	season	holiday	workingday	weather	temp	atemp	humidity	windsp
season	1.000000	0.029368	-0.008126	0.008879	0.258689	0.264744	0.190610	-0.147
holiday	0.029368	1.000000	-0.250491	-0.007074	0.000295	-0.005215	0.001929	0.008
workingday	-0.008126	-0.250491	1.000000	0.033772	0.029966	0.024660	-0.010880	0.013
weather	0.008879	-0.007074	0.033772	1.000000	-0.055035	-0.055376	0.406244	0.007
temp	0.258689	0.000295	0.029966	-0.055035	1.000000	0.984948	-0.064949	-0.017
atemp	0.264744	-0.005215	0.024660	-0.055376	0.984948	1.000000	-0.043536	-0.057
humidity	0.190610	0.001929	-0.010880	0.406244	-0.064949	-0.043536	1.000000	-0.318
windspeed	-0.147121	0.008409	0.013373	0.007261	-0.017852	-0.057473	-0.318607	1.000
casual	0.096758	0.043799	-0.319111	-0.135918	0.467097	0.462067	-0.348187	0.092
registered	0.164011	-0.020956	0.119460	-0.109340	0.318571	0.314635	-0.265458	0.091
count	0.163439	-0.005393	0.011594	-0.128655	0.394454	0.389784	-0.317371	0.101

In [33]:
```
#convert numerical to categorical
cat_cols=["weather","season"]

for i in cat_cols:
    yu[i]=yu[i].astype("object")
```

In [34]:
```
yu.dtypes
```

Out[34]:
```
season         object
holiday         int64
workingday     object
weather        object
temp          float64
atemp         float64
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
date           object
time           object
dtype: object
```

In [14]:
```
#a relation between the dependent and independent variable (Dependent “Count”

from scipy.stats import ttest_ind

ttest_ind(yu["workingday"],yu["count"])
```

Out[14]:
```
Ttest_indResult(statistic=-109.95076974934595, pvalue=0.0)
```

In [40]:
```
ttest_ind(yu["weather"],yu["count"])
```

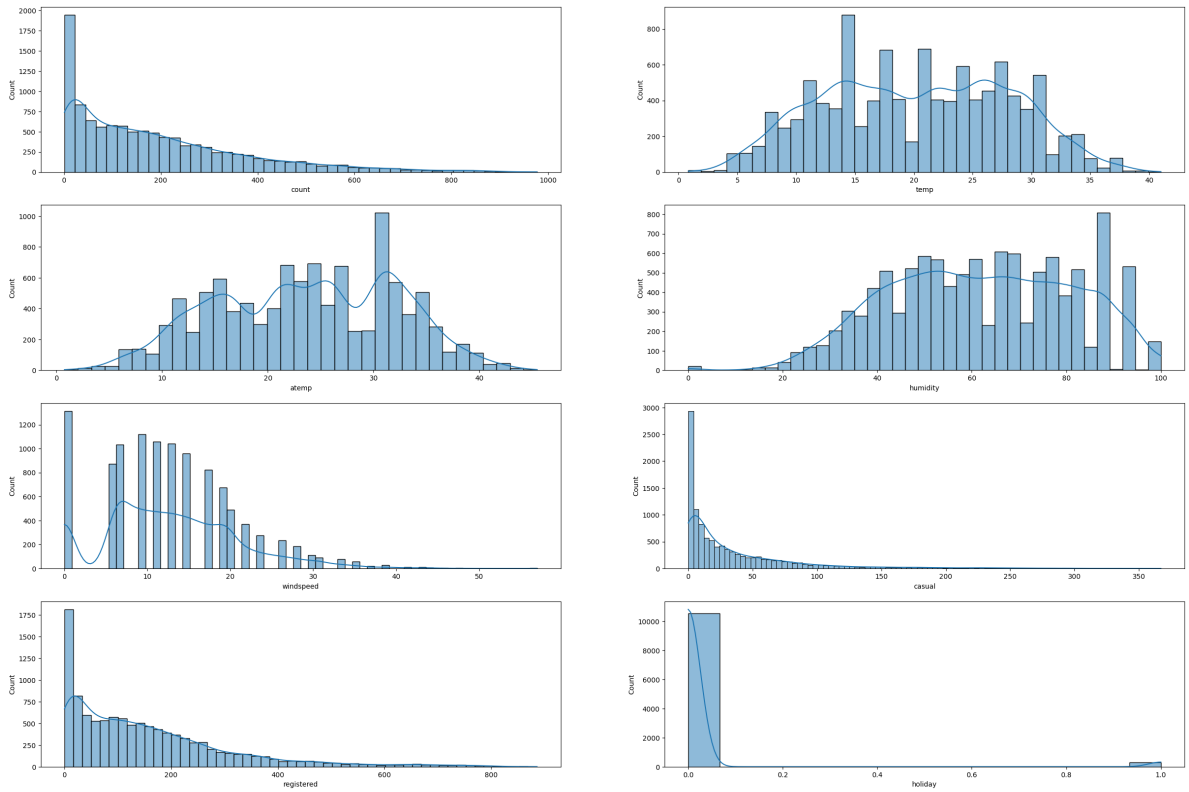
Out[40]:
```
Ttest_indResult(statistic=-109.5256459753639, pvalue=0.0)
```

In [43]:
```
ttest_ind(yu["season"],yu["count"])
```

Out[43]:
```
Ttest_indResult(statistic=-108.89747295682916, pvalue=0.0)
```

Univariate Analysis

In [20]:
```
hist=["count","temp","atemp","humidity","windspeed","casual","registered","hol

fig,axs=plt.subplots(nrows=4,ncols=2,figsize=(30,20))
c=0

for i in range(4):
    for j in range(2):
        sns.histplot(data=yu,x=hist[c],kde=True,ax=axs[i,j])
        c+=1
plt.show()
```

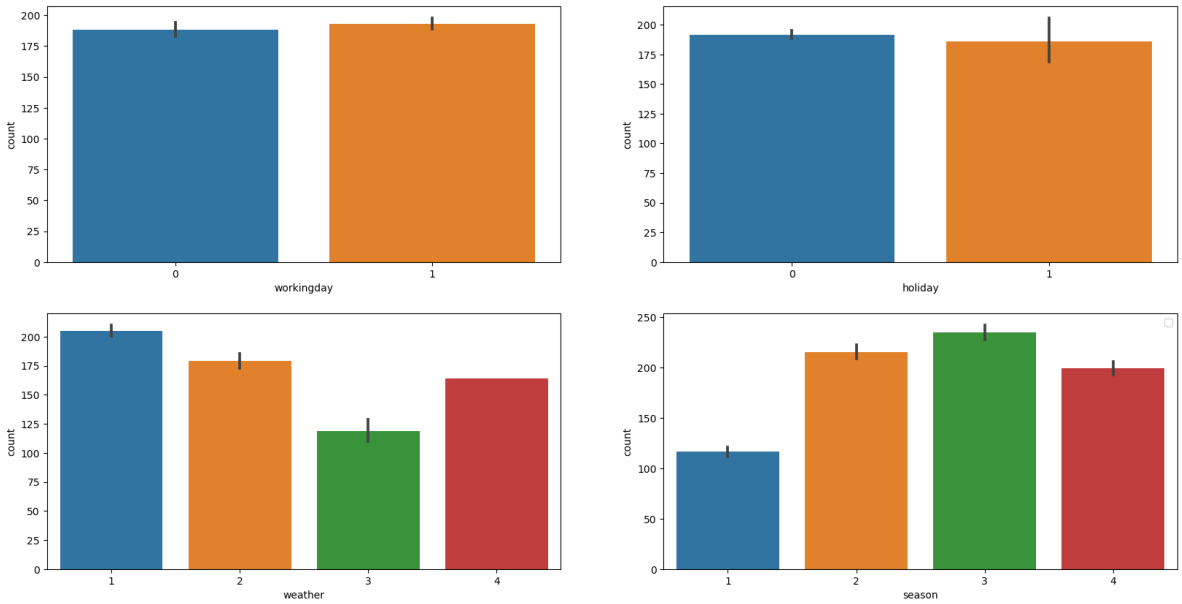


Bivariate Analysis:

(Relationships between important variables such as workday and count, season and count, weather and count.

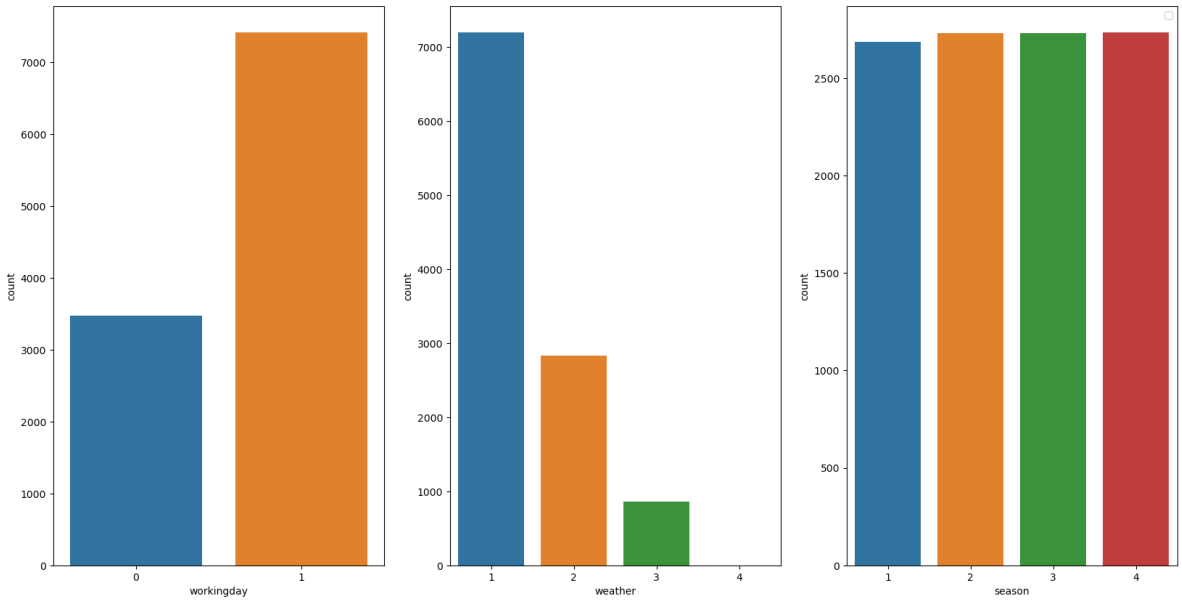
In [19]:
```
hist=["workingday","holiday","weather","season"]

fig,axs=plt.subplots(nrows=2,ncols=2,figsize=(20,10))
c=0

for i in range(2):
    for j in range(2):
        sns.barplot(data=yu,x=hist[c],y=yu["count"],ax=axs[i,j])
        plt.legend(loc="upper right")
        c+=1
plt.show()


#sns.countplot(data=yu,x="workingday")
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
```



In [28]:
```
hist=["workingday","weather","season"]

fig,axs=plt.subplots(nrows=1,ncols=3,figsize=(20,10))
c=0

for i in range(3):
        sns.countplot(data=yu,x=hist[c],ax=axs[i])
        plt.legend(loc="upper right")
        c+=1
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argumen
t.
```

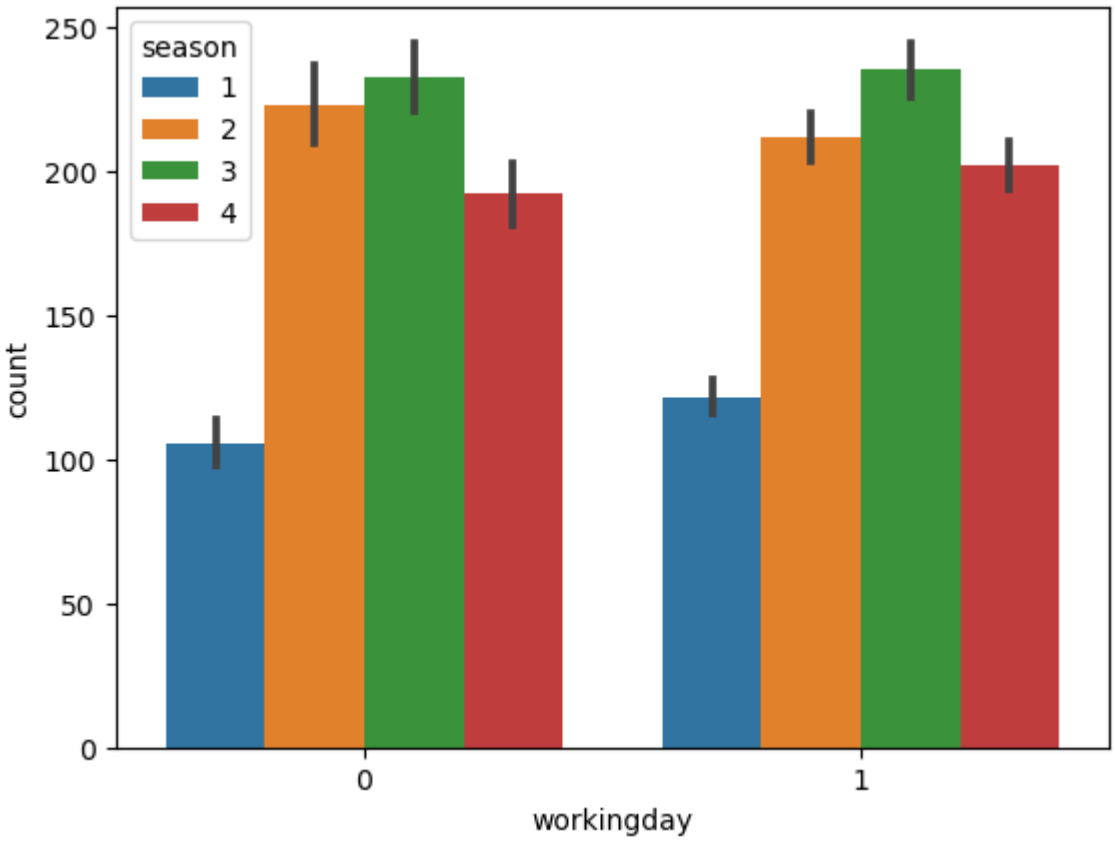


In [ ]:

Bivariate analysis

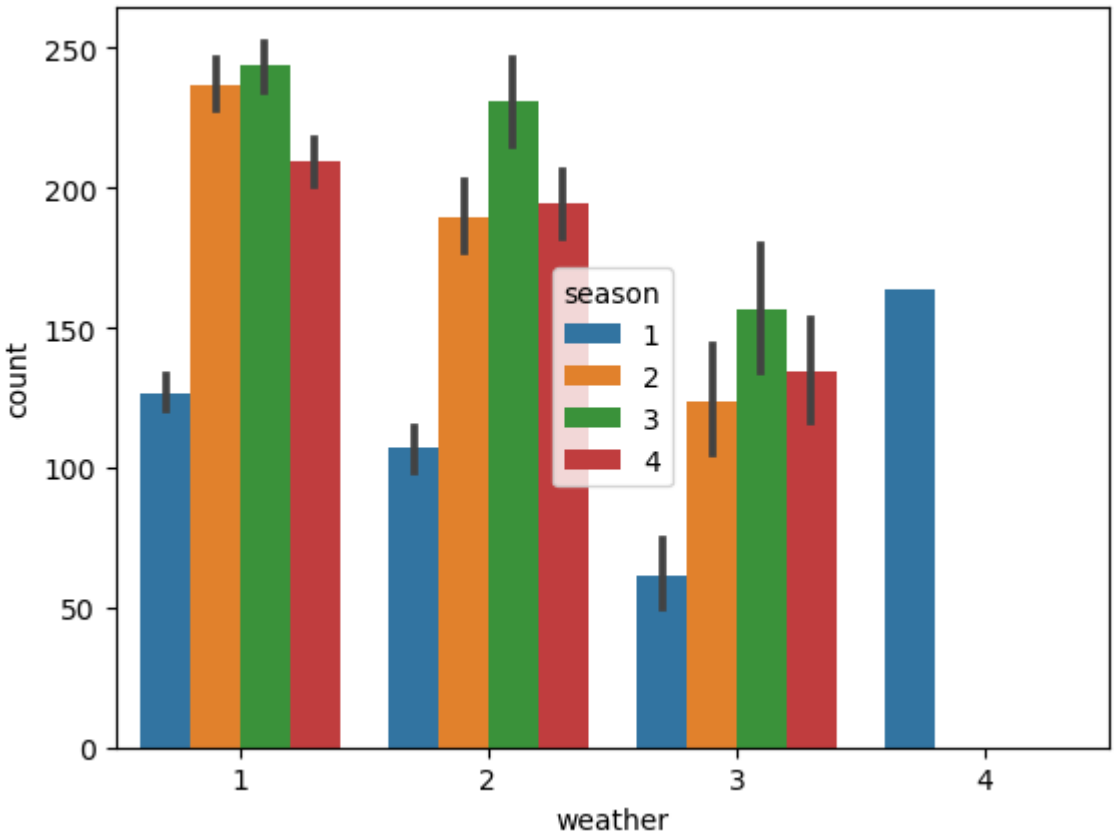
In [51]:
```
sns.barplot(x=yu["workingday"],y=yu["count"],hue=yu["season"])
```

Out[51]:
```
<Axes: xlabel='workingday', ylabel='count'>
```



In [52]:
```
sns.barplot(x=yu["weather"],y=yu["count"],hue=yu["season"])
```

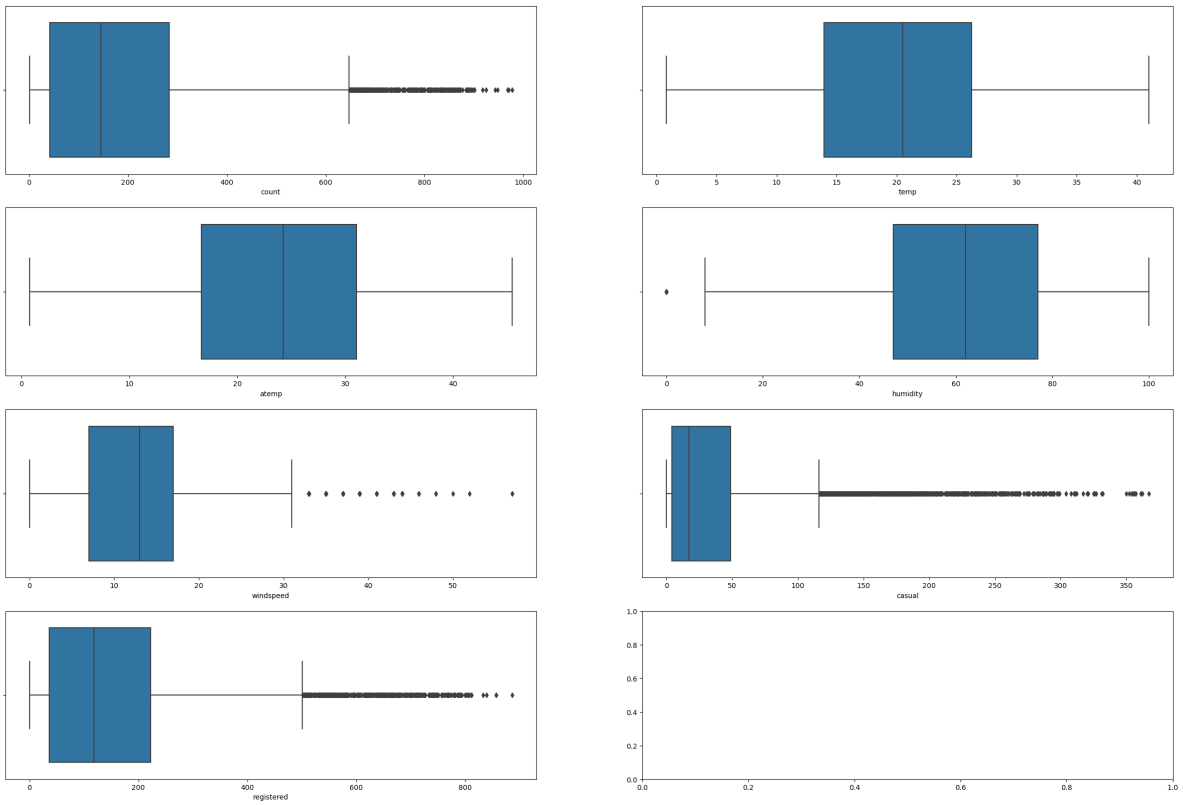
Out[52]:
```
<Axes: xlabel='weather', ylabel='count'>
```



In [51]:

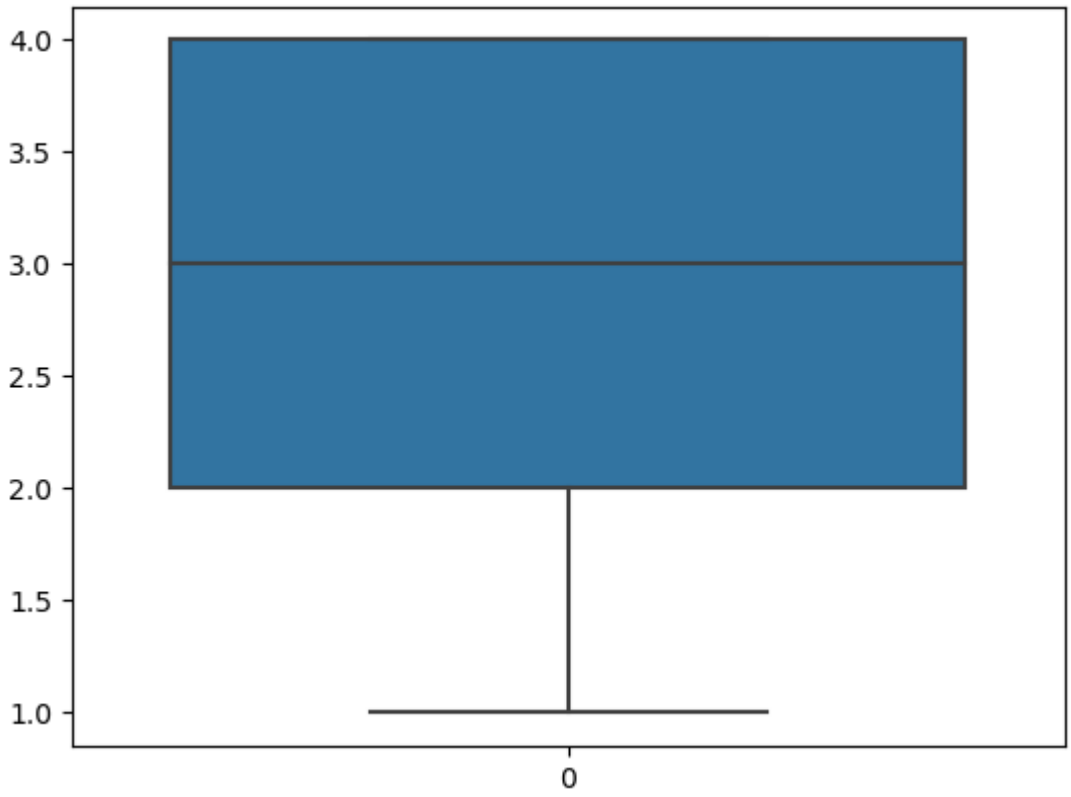
```
hist=["count","temp","atemp","humidity","windspeed","casual","registered"]

fig,axs=plt.subplots(nrows=4,ncols=2,figsize=(30,20))
c=0

for i in range(4):
    for j in range(2):
        sns.boxplot(data=yu,x=hist[c],ax=axs[i,j])
        c+=1
plt.show()
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[51], line 8
      6 for i in range(4):
      7     for j in range(2):
----> 8         sns.boxplot(data=yu,x=hist[c],ax=axs[i,j])
      9         c+=1
     10 plt.show()

IndexError: list index out of range
```



In [53]:
```
sns.boxplot(yu["season"])
```

Out[53]: <Axes: >

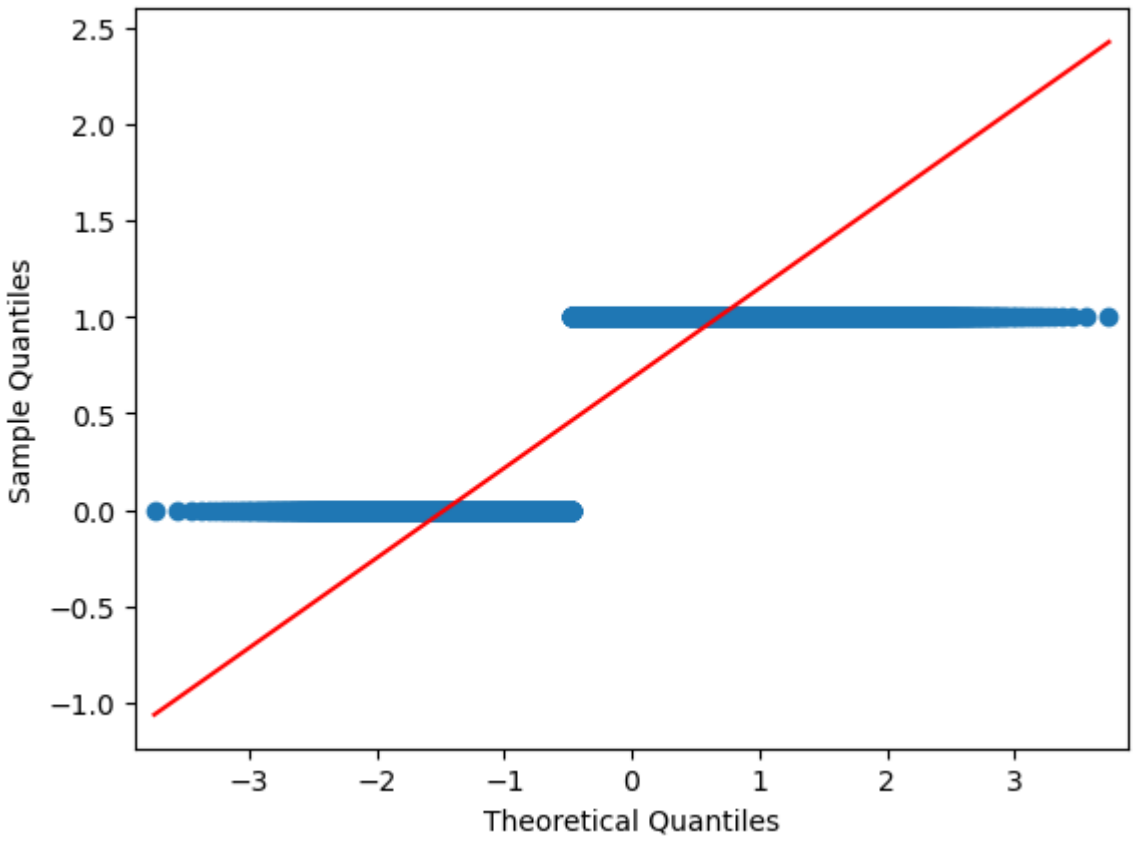


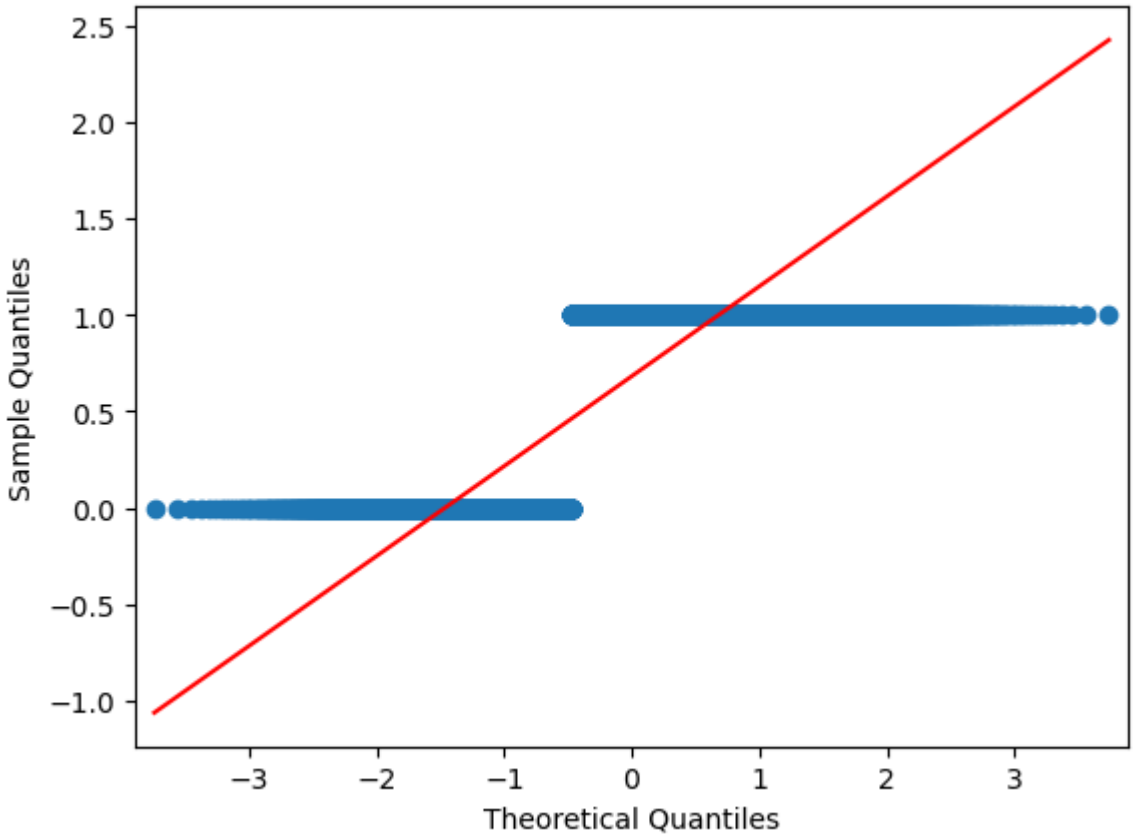
In [ ]:
```

```

Hypothesis Testing

In [20]:
```
#Test for gaussian
from statsmodels.graphics.gofplots import qqplot
qqplot(yu["workingday"],line="s")
```

Out[20]:





In [37]:
```
#checking variances of 2 groups

levene(yu["workingday"],yu["count"])
```

Out[37]:
```
LeveneResult(statistic=12954.75107188816, pvalue=0.0)
```

levene test shows that variances are not equal between both groups and there is significant difference.

2- Sample T-Test

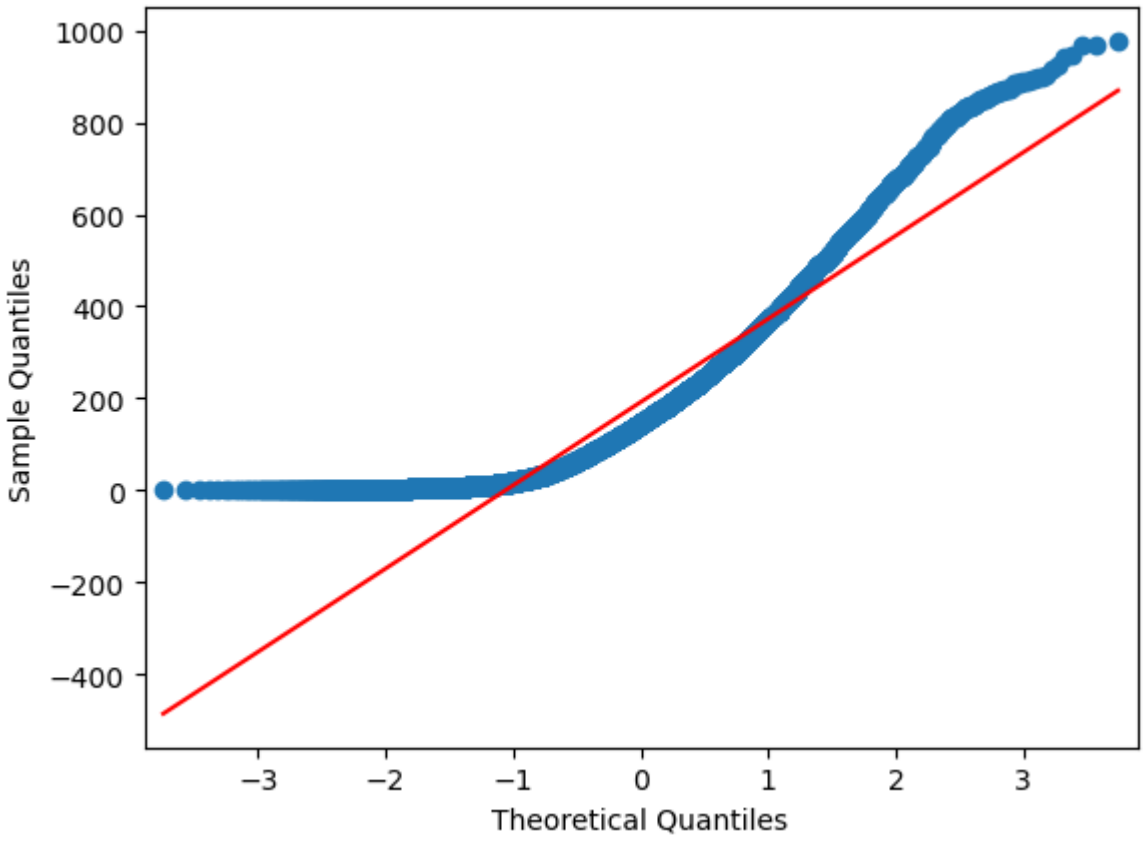
In [56]:
```
# with assumption of 95% CI ,we consider significance level to be 5%.
#H0:Working Day has no effect on the number of electric cycles rented
#Ha:Working Day has an effect on the number of electric cycles rented

from scipy.stats import ttest_ind,levene,shapiro,f_oneway,chisquare,chi2,chi2_

t_stat,p_value= ttest_ind(list(yu["workingday"]),list(yu["count"]))


if p_value < 0.05:
    print("t_stat",t_stat)
    print("p_value",p_value)

    print("Working Day has an effect on the number of electric cycles rented.

else:
    print("Working Day has no effect on the number of electric cycles rented."
```

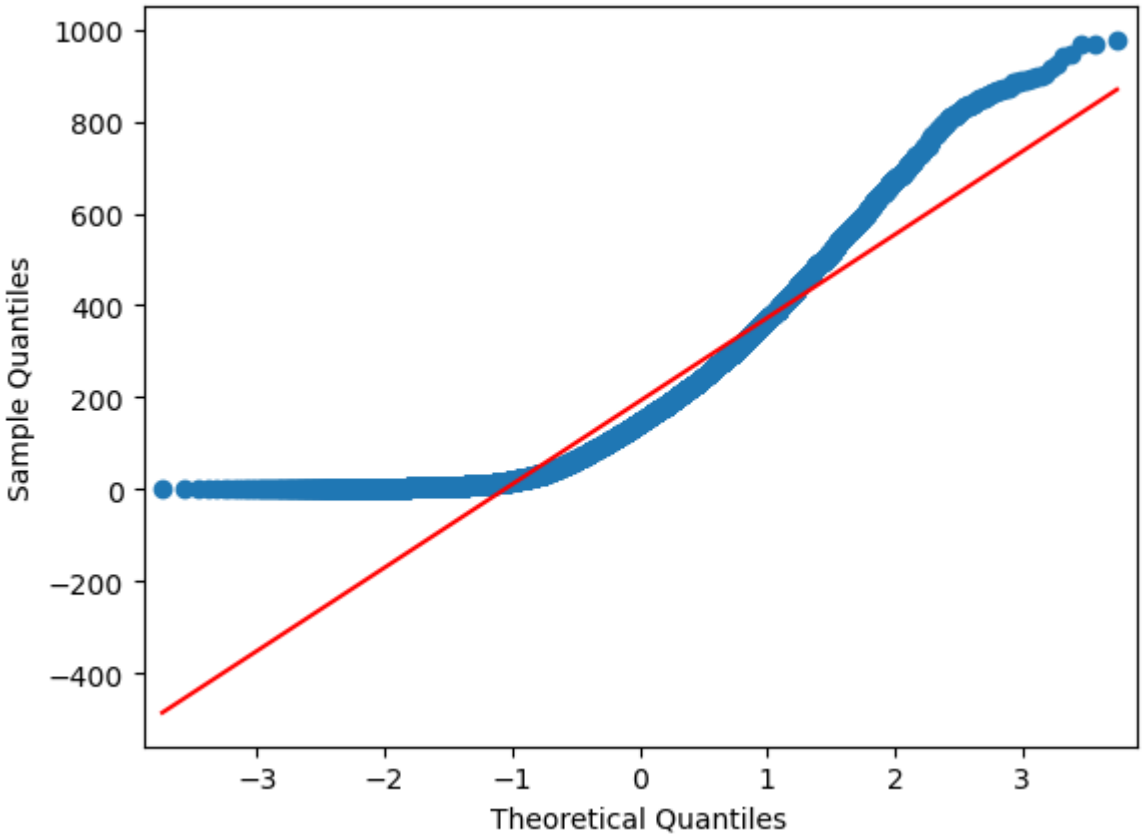
```
t_stat -109.95076974934595
p_value 0.0
Working Day has an effect on the number of electric cycles rented.
```

ANNOVA

In [40]:
```
#Anova

#1. test gaussian distribution

qqplot(yu["count"],line="s")
```

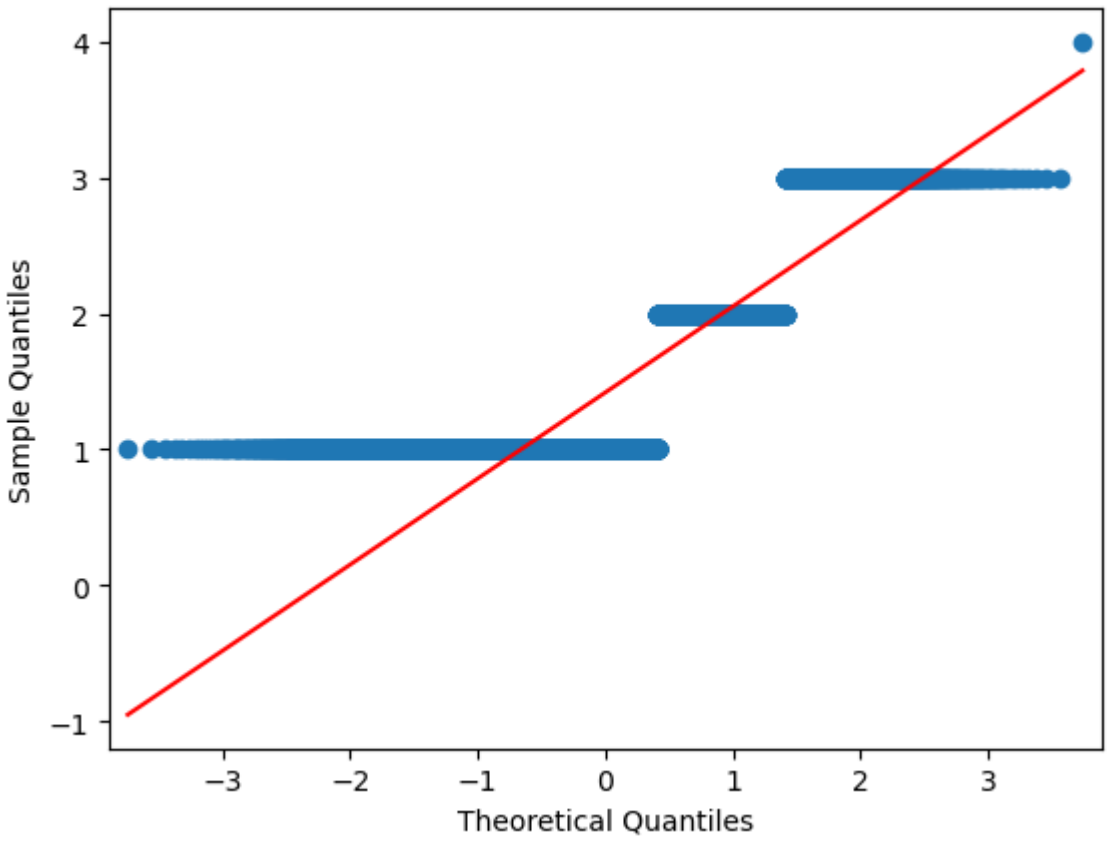
Out[40]:

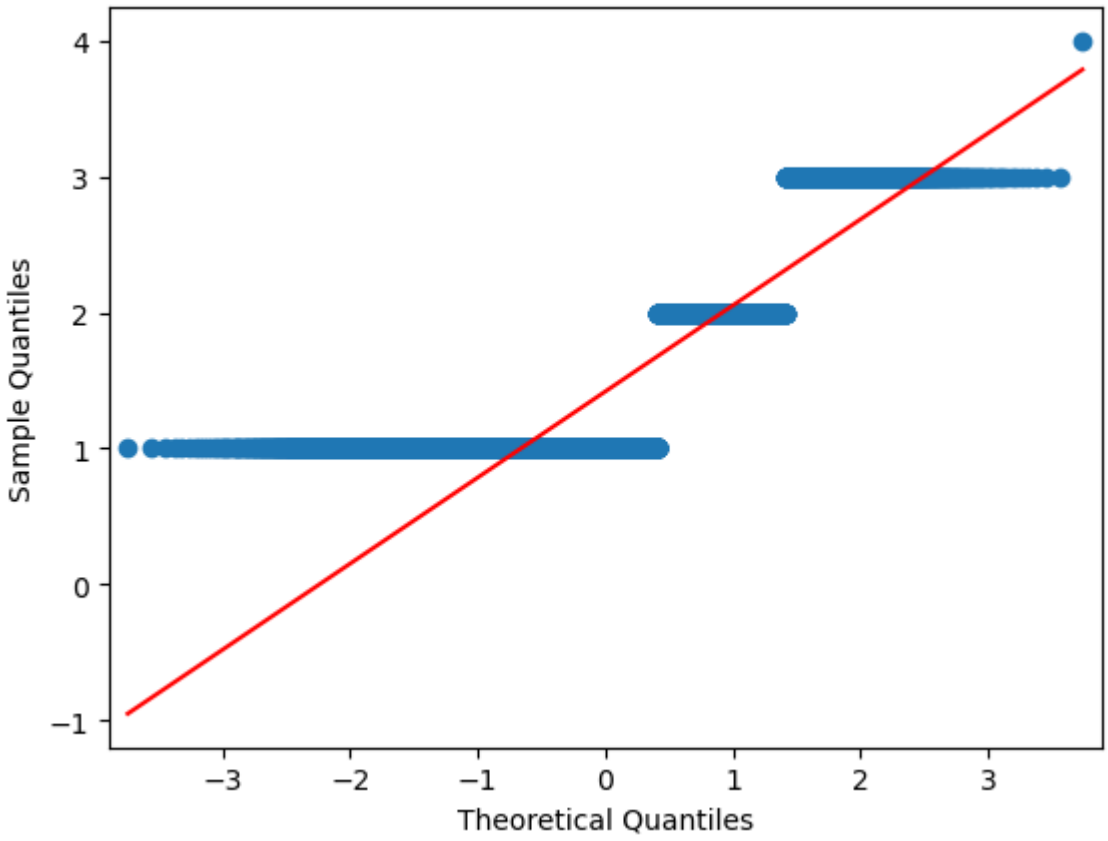



1000
800
600
400
200
0
−200
−400
Sample Quantiles
−3
−2
−1
0
1
2
3
Theoretical Quantiles

In [41]:
```
qqplot(yu["weather"],line="s")
```

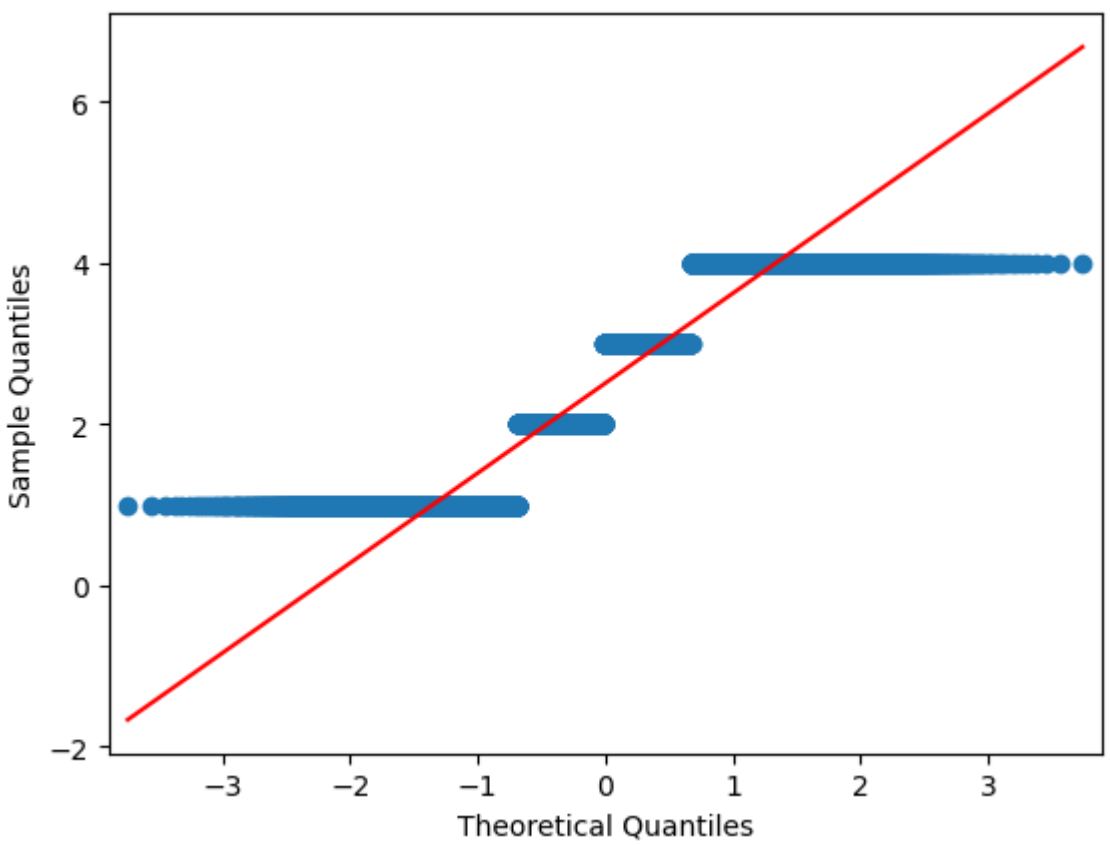
Out[41]:

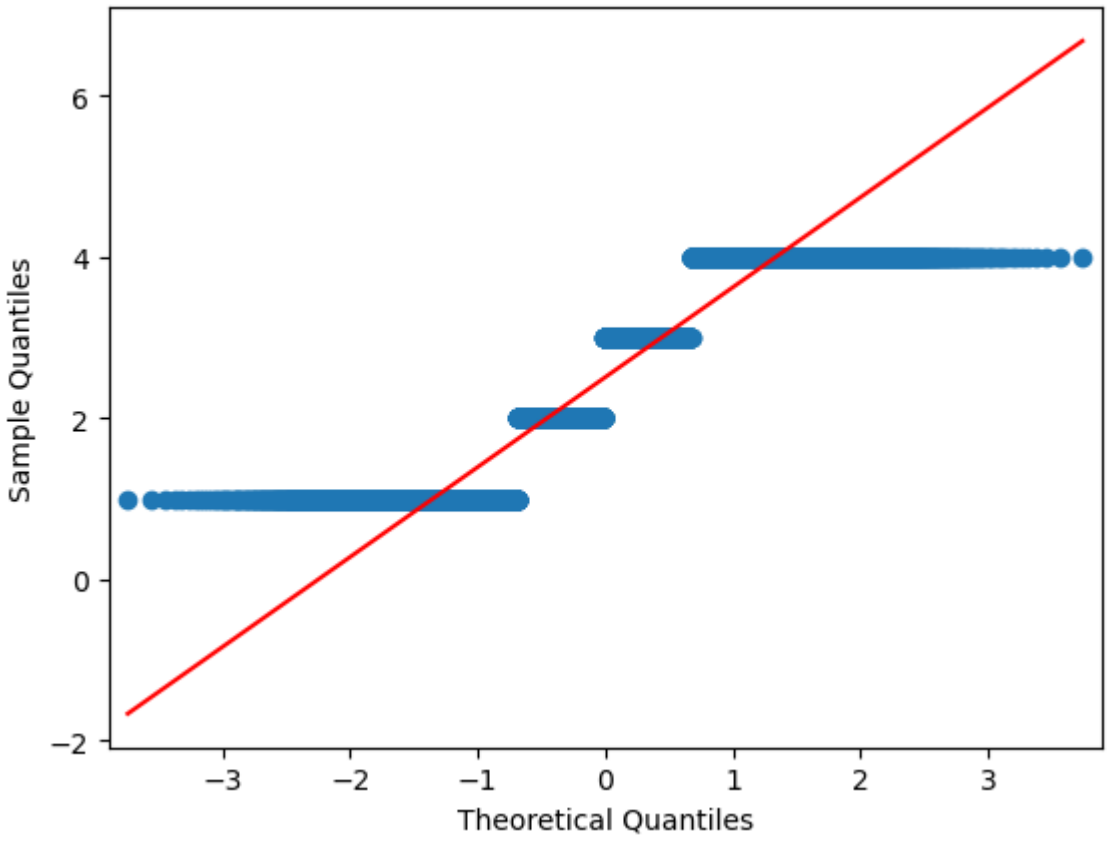




In [42]:
```
qqplot(yu["season"],line="s")
```

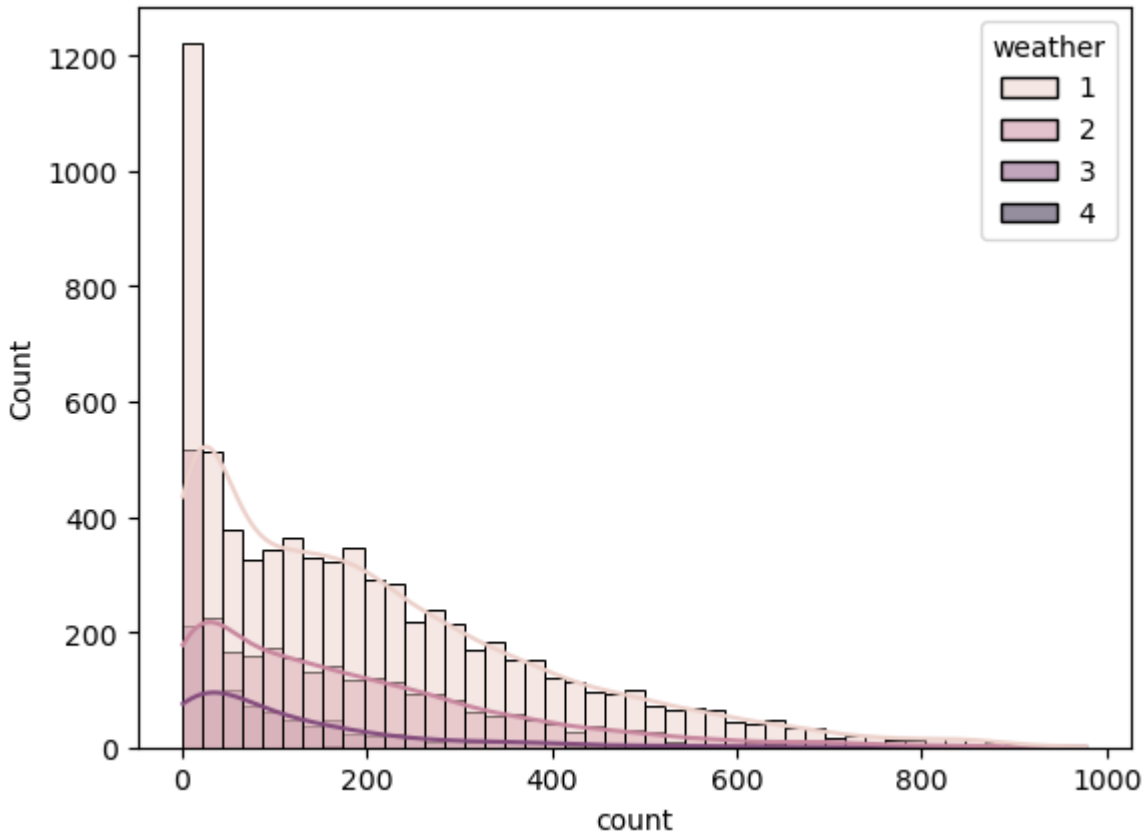
Out[42]:





In [22]:
```
#2.Levene test

sns.histplot(data=yu,x="count",hue="weather",color="g",kde=True)
```

Out[22]:
```
<Axes: xlabel='count', ylabel='Count'>
```



Distribution is right-skewed and is not uniformly distributed.people prefering yulu decreases or is seemingly distributed only until 600 total bikes.

In [52]:
```
#H0:weather & count means are equal
#Ha: weather & count means are not equal

ttest_ind(yu["weather"],yu["count"])
```

Out[52]:
```
Ttest_indResult(statistic=-109.5256459753639, pvalue=0.0)
```

```
# p_value is less than 5% alpha value .Thus reject H0, both groups has
different means and are drwan from different population.
```

In [53]:
```
#H0:weather & count variances are equal
#Ha: weather & count variances are not equal
levene(yu["weather"],yu["count"])
```

Out[53]:
```
LeveneResult(statistic=12935.92310973606, pvalue=0.0)
```

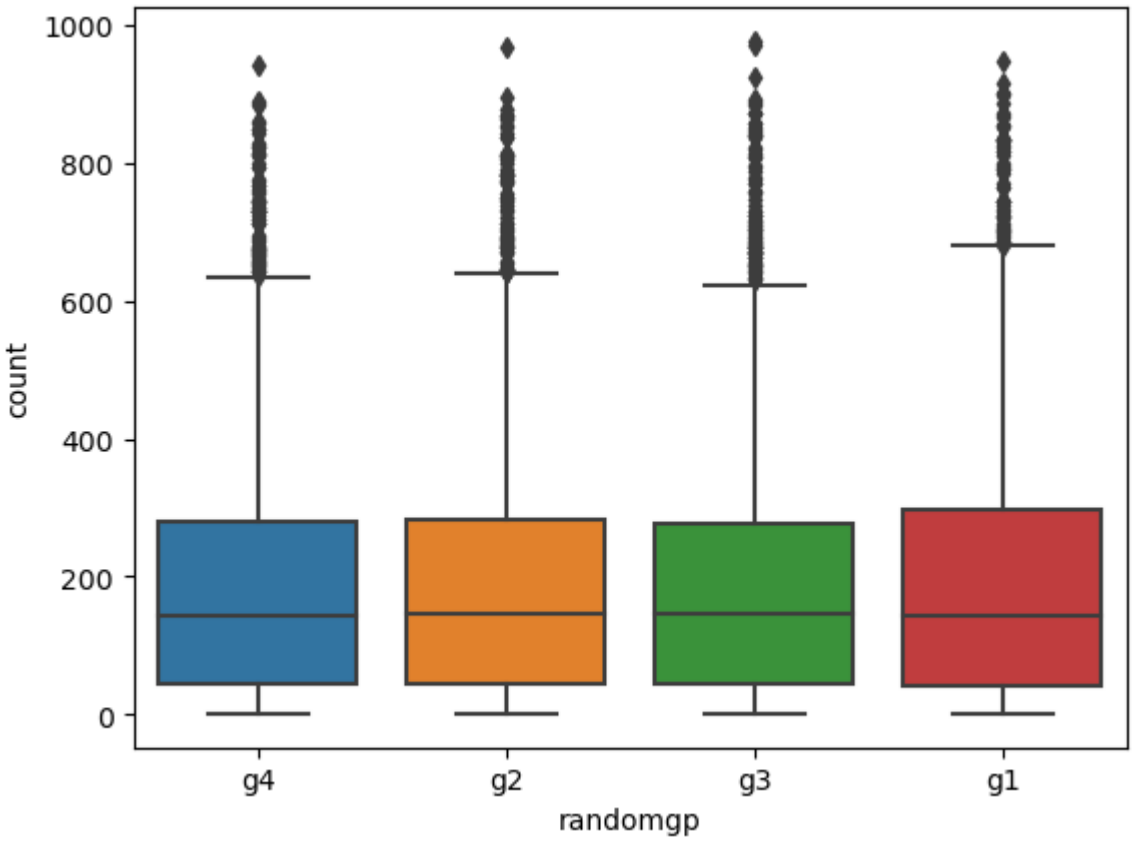
levene test also confirms that variances are not equal.

In [156]:
```
yu["randomgp"]=np.random.choice(["g1","g2","g3","g4"],size=len(yu))

g1=yu[yu["randomgp"]=="g1"]["weather"]
g2=yu[yu["randomgp"]=="g2"]["weather"]
g3=yu[yu["randomgp"]=="g3"]["weather"]
g4=yu[yu["randomgp"]=="g4"]["weather"]


f_stat,p_value=f_oneway(g1,g2,g3,g4)

if p_value < 0.05:
    print("f_stat",f_stat)
    print("p_value",p_value)

    print("weather has an effect on the number of electric cycles rented. \n")

else:
    print("f_stat",f_stat)
    print("p_value",p_value)
    print("weather has no effect on the number of electric cycles rented.")
```

```
f_stat 0.3380193046682795
p_value 0.7978505677100386
weather has no effect on the number of electric cycles rented.
```

In [157]:
```
sns.boxplot(x="randomgp",y="count",data=yu)
```

Out[157]:
```
<Axes: xlabel='randomgp', ylabel='count'>
```



High p_value greater than 5% signioficance level which tells us that means of all the groups are near to each other.Thus the difference is by chance and not significantly different.

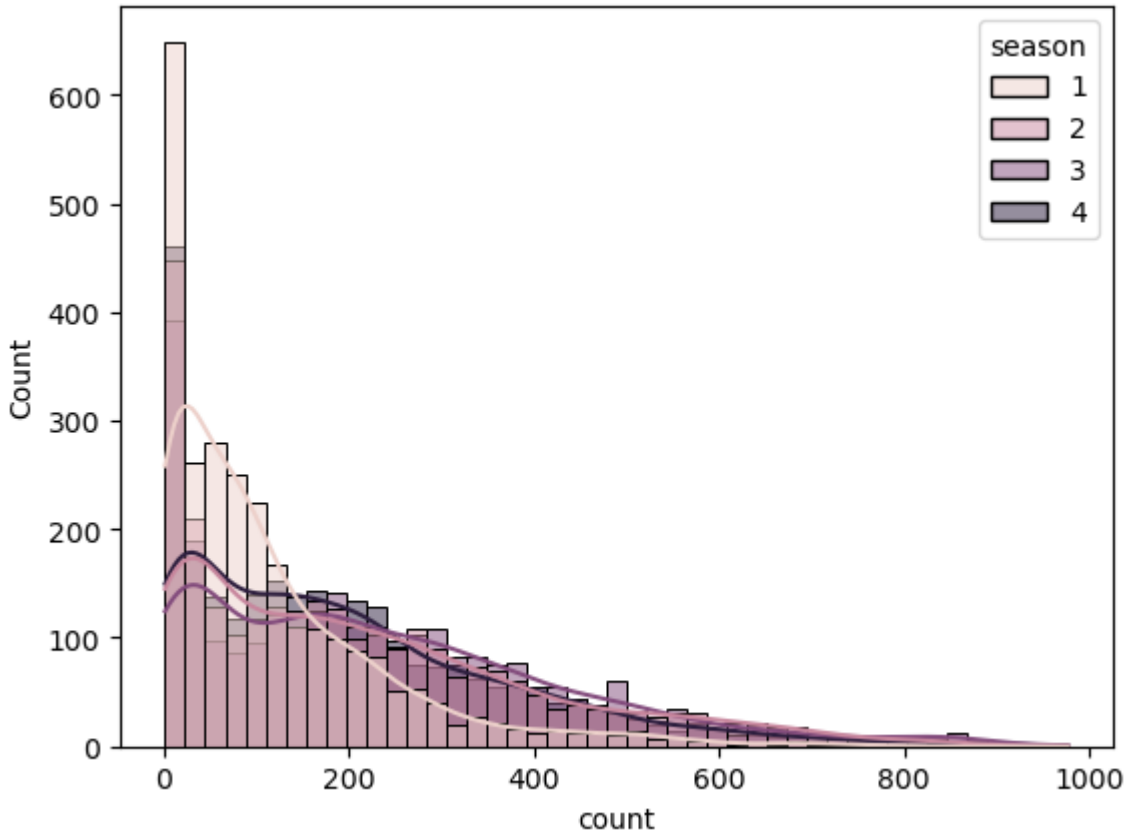
In [160]:
```
import warnings
warnings.filterwarnings("ignore")


g1=yu[yu["weather"]==1]["count"]
g2=yu[yu["weather"]==2]["count"]
g3=yu[yu["weather"]==3]["count"]
g4=yu[yu["weather"]==4]["count"]


f_stat,p_value=f_oneway(g1,g2,g3,g4)

if p_value < 0.05:
    print("f_stat",f_stat)
    print("p_value",p_value)

    print("weather has an effect on the number of electric cycles rented.\n")

else:
    print("f_stat",f_stat)
    print("p_value",p_value)
    print("weather has no effect on the number of electric cycles rented.")
```

```
f_stat 65.53024112793271
p_value 5.482069475935669e-42
weather has an effect on the number of electric cycles rented.
```

Low p_value greater than 5% significance level which tells us that means of all the groups are not near to each other.Thus they are significantly different.

In [161]:
```
# season is not gaussian shown in above qqplot

#2.Levene test

sns.histplot(data=yu,x="count",hue="season",color="g",kde=True)
```

Out[161]: <Axes: xlabel='count', ylabel='Count'>



Distribution is right-skewed and is not uniformly distributed.people prefering yulu decreases or is seemingly distributed only until range of 600 total bikes.

In [163]:
```
#H0:weather & count means are equal
#Ha: weather & count means are not equal

ttest_ind(yu["season"],yu["count"])
```

Out[163]: Ttest_indResult(statistic=-108.89747295682916, pvalue=0.0)

In [ ]:
```
# p_value is less than 5% alpha value .Thus reject H0, both groups has differe
```

In [23]:
```
#H0:season & count variances are equal
#Ha: season & count variances are not equal
levene(yu["season"],yu["count"])

if p_value < 0.05:
    print("Reject H0, season & count variances are not equal")
else:
    print("season & count variances are equal")
```

```
Reject H0, season & count variances are not equal
```

levene test also confirms that variances are not equal.

In [29]:
```
#Creating random groups g1,g2,g3,g4 to test if season has an effect on the num

yu["randomgp"]=np.random.choice(["g1","g2","g3","g4"],size=len(yu))

g1=yu[yu["randomgp"]=="g1"]["count"]
g2=yu[yu["randomgp"]=="g2"]["count"]
g3=yu[yu["randomgp"]=="g3"]["count"]
g4=yu[yu["randomgp"]=="g4"]["count"]


f_stat,p_value=f_oneway(g1,g2,g3,g4)

if p_value < 0.05:
    print("f_stat",f_stat)
    print("p_value",p_value)

    print("season has an effect on the number of electric cycles rented. \n")

else:
    print("f_stat",f_stat)
    print("p_value",p_value)
    print("season has no effect on the number of electric cycles rented.")
```

```
f_stat 0.01617854024669315
p_value 0.9971970003671882
season has no effect on the number of electric cycles rented.
```

As shown above High p_value greater than 5% significance level which tells us that means of all the groups are near to each other.Thus the difference is by chance and not significantly different.

In [328]:
```
#Creating groups based on actual values of season 1,2,3,4.
import warnings
warnings.filterwarnings("ignore")


g1=yu[yu["season"]==1]["count"]
g2=yu[yu["season"]==2]["count"]
g3=yu[yu["season"]==3]["count"]
g4=yu[yu["season"]==4]["count"]


f_stat,p_value=f_oneway(g1,g2,g3,g4)

if p_value < 0.05:
    print("f_stat",f_stat)
    print("p_value",p_value)

    print("season has an effect on the number of electric cycles rented.\n")

else:
    print("f_stat",f_stat)
    print("p_value",p_value)
    print("season has no effect on the number of electric cycles rented.")
```

```
f_stat 236.94671081032106
p_value 6.164843386499654e-149
season has an effect on the number of electric cycles rented.
```

Low p_value greater than 5% significance level which tells us that means of all the groups are not near to each other.Thus they are significantly different.

Chi-square test

In [31]:
```
val=pd.crosstab(yu["weather"],yu["season"])
val
```

Out[31]:

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

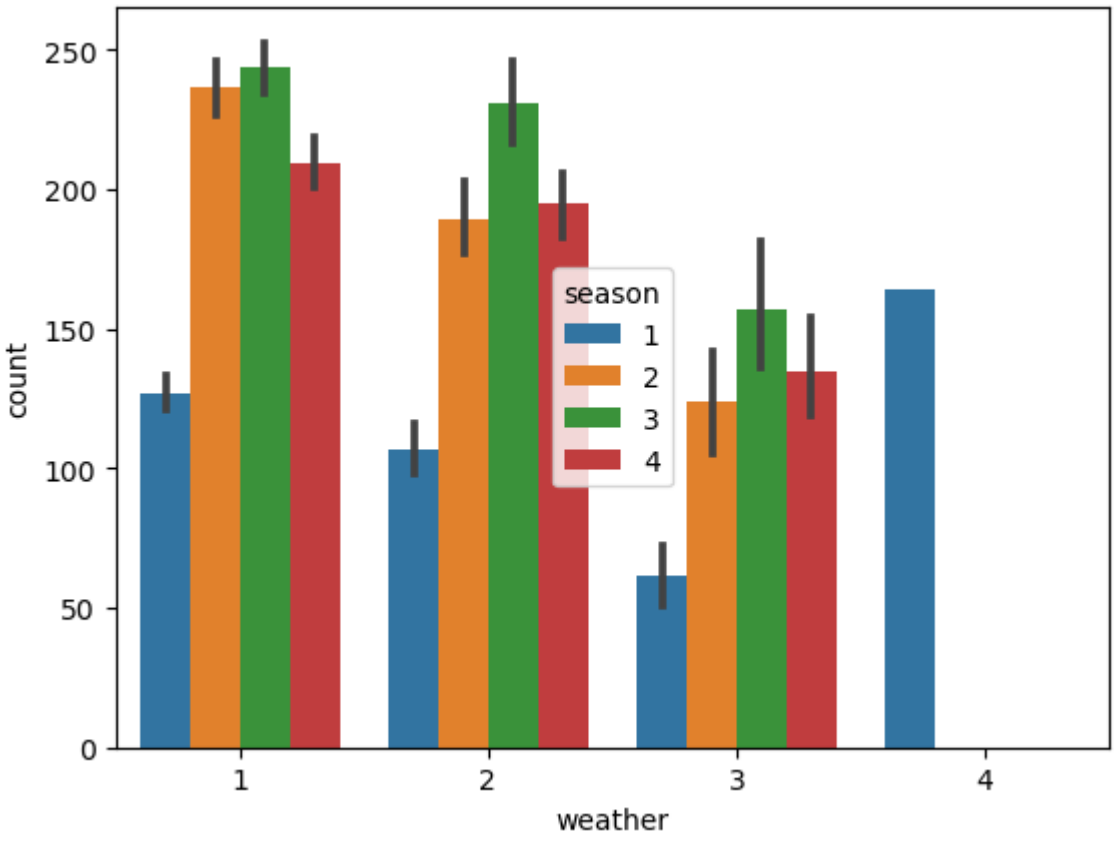
In [37]:
```
#H0:Weather & season are both independent
#Ha:Weather & season are both dependent

chi2_contingency(val)
```

Out[37]:
```
Chi2ContingencyResult(statistic=49.15865559689363, pvalue=1.5499250736864862e
-07, dof=9, expected_freq=array([[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]]))
```

 # Since pvalue is very low nearly 0 then we can reject H0 i.e.,Weather & season are both independent.

In [47]:
```
sns.barplot(x=yu["weather"],y=yu["count"],hue=yu["season"])
```

Out[47]:
```
<Axes: xlabel='weather', ylabel='count'>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: