Yulu Bikes Case Study

About Yulu:

Yulu, India's pioneering micro-mobility service provider, has embarked on a mission to revolutionize daily commutes by offering unique, sustainable transportation solutions.

Problem Statement

Yulu's recent revenue decline is a pressing concern. Specifically, they want to understand the factors affecting the demand for these shared electric cycles in the Indian market. As a Data scientist we are looking at the dataset to present a positive suggestion to regain profitability in the market.

1.0 Importing Libraries and loading the Dataset

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

```
!gdown 'https://drive.google.com/uc?id=1o94fXnmvrx6jRgI6S-SeZ3tfnKjCDY0i' -O
'bike_sharing.csv'
```

```
Downloading...
From: https://drive.google.com/uc?id=1o94fXnmvrx6jRgI6S-SeZ3tfnKjCDY0i
To: /content/bike_sharing.csv
  0% 0.00/648k [00:00<?, ?B/s]100% 648k/648k [00:00<00:00, 88.2MB/s]
```

```
df = pd.read_csv('/content/bike_sharing.csv')
```

#2.0 Exploratory Data Analysis

```
df.sample(10, random_state= 42)
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
3133	2011-07-19 11:00:00	3	0	1	1	33.62	40.150	59	0.0000
5786	2012-01-16 06:00:00	1	1	0	1	4.10	6.820	54	6.0032
5224	2011-12-11 18:00:00	4	0	0	1	9.84	11.365	48	12.9980
8953	2012-08-15 10:00:00	3	0	1	2	29.52	34.090	62	12.9980
8054	2012-06-15 23:00:00	2	0	1	1	25.42	31.060	53	16.9979
10044	2012-11-03 21:00:00	4	0	0	1	13.94	17.425	53	7.0015
5337	2011-12-16 11:00:00	4	0	1	2	13.94	15.150	42	19.9995
2753	2011-07-03 15:00:00	3	0	0	1	34.44	40.150	53	19.9995
10127	2012-11-07 08:00:00	4	0	1	2	10.66	12.120	60	19.0012
33	2011-01-02 10:00:00	1	0	0	2	14.76	17.425	81	15.0013

Insights - It is clear that certain categorical column's value are numbered instead of strings. - It is a good practice to replace them according to have better visual representation aspect.

```
df_unique_values = df.copy()
df.shape
```

```
(10886, 12)
```

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

```
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
```

Insights - There are no null values in our dataset.

```
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  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
```

Insight - There are totally 10886 rows and 12 columns. - The data type of columns are wrongfully assigned and needed to be assigned appropriately.

Further Action- - Change the data type of the columns to fit the analysis.

2.1 Changing the data types of columns

- datetime –> to datetime
- season –> to categorical variable
- holiday –> to categorical variable
- workingday –> to categorical variable
- weather –> to categorical variable

```python
df['datetime'] = pd.to_datetime(df['datetime'])
df['season'] = df['season'].astype('category')
df['holiday'] = df['holiday'].astype('category')
df['workingday'] = df['workingday'].astype('category')
df['weather'] = df['weather'].astype('category')
```

```python
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  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
dtypes: category(4), datetime64[ns](1), float64(3), int64(4)
memory usage: 723.7 KB
```

- All the datatypes are changed perfectly.

2.2 Handling the datetime column

2.2.1 Splitting the datetime column

```
df['day'] = df['datetime'].dt.day
df['month'] = df['datetime'].dt.month_name().str[:3]
df['year'] = df['datetime'].dt.year
df['month']
```

```
0        Jan
1        Jan
2        Jan
3        Jan
4        Jan
        ...
10881    Dec
10882    Dec
10883    Dec
10884    Dec
10885    Dec
Name: month, Length: 10886, dtype: object
```

2.2.2 Assessing the column

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

```
2011-01-01 00:00:00
2012-12-19 23:00:00
718 days 23:00:00
```

Insights - The dataset have records for 718 days and it starts from 01/01/2011 and goes upto 19/12/2012.

2.3 Checking for nested vaues

```python
def nested_values_check():
  df_check = df.copy() # Creating a copy to have the main dataframe unaffected.
  for i in df_check.columns:
    df_check[i] = df_check[i].astype('str')
    if df_check[i].str.contains(', ').any():
      print(f'{i} column --> Have nested values')
    else:
      print(f'{i} column --> No nested values')

nested_values_check()
```

```
datetime column --> No nested values
season column --> No nested values
holiday column --> No nested values
workingday column --> No nested values
weather column --> No nested values
temp column --> No nested values
atemp column --> No nested values
humidity column --> No nested values
windspeed column --> No nested values
casual column --> No nested values
registered column --> No nested values
count column --> No nested values
day column --> No nested values
month column --> No nested values
year column --> No nested values
```

- We can confidently say that there are no nested values in our column.

2.4 Accessing the unique values in each column

```python
df.nunique()
```

```
datetime      10886
season            4
holiday           2
workingday        2
weather           4
temp             49
atemp            60
humidity         89
windspeed        28
casual          309
registered      731
count           822
day              19
month            12
year              2
dtype: int64
```

```
for i in df.columns:
  if df[i].dtype == 'int64' or df[i].dtype == 'float64':
    print(i)
    print(np.sort(df[i].unique()))
    print('\n')
  else:
    print(i)
    print(df[i].unique())
    print('\n')
```

```
datetime
<DatetimeArray>
['2011-01-01 00:00:00', '2011-01-01 01:00:00', '2011-01-01 02:00:00',
 '2011-01-01 03:00:00', '2011-01-01 04:00:00', '2011-01-01 05:00:00',
 '2011-01-01 06:00:00', '2011-01-01 07:00:00', '2011-01-01 08:00:00',
 '2011-01-01 09:00:00',
 ...
 '2012-12-19 14:00:00', '2012-12-19 15:00:00', '2012-12-19 16:00:00',
 '2012-12-19 17:00:00', '2012-12-19 18:00:00', '2012-12-19 19:00:00',
 '2012-12-19 20:00:00', '2012-12-19 21:00:00', '2012-12-19 22:00:00',
 '2012-12-19 23:00:00']
Length: 10886, dtype: datetime64[ns]


season
[1, 2, 3, 4]
Categories (4, int64): [1, 2, 3, 4]


holiday
```

```
[0, 1]
Categories (2, int64): [0, 1]


workingday
[0, 1]
Categories (2, int64): [0, 1]


weather
[1, 2, 3, 4]
Categories (4, int64): [1, 2, 3, 4]


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


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


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


windspeed
[ 0.      6.0032  7.0015  8.9981 11.0014 12.998  15.0013 16.9979 19.0012
 19.9995 22.0028 23.9994 26.0027 27.9993 30.0026 31.0009 32.9975 35.0008
 36.9974 39.0007 40.9973 43.0006 43.9989 46.0022 47.9988 50.0021 51.9987
 56.9969]


casual
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
```

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


registered
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449
 450 451 452 453 454 455 456 458 459 460 461 462 463 464 465 466 467 468
 469 470 471 472 473 474 475 477 478 479 480 481 483 484 485 486 487 488
 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506
```

```
 507 508 509 510 511 512 513 514 515 516 517 518 521 522 523 525 527 529
 530 531 532 533 534 536 537 538 539 540 541 542 543 544 545 546 547 548
 549 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567
 568 570 571 572 573 575 576 577 578 579 580 581 582 583 584 586 589 591
 593 594 595 596 597 598 601 602 603 604 605 608 609 610 613 614 615 616
 617 618 619 620 621 622 623 624 625 626 628 629 631 633 634 636 637 638
 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 655 656 658
 659 660 661 662 663 664 665 666 667 668 669 670 672 673 675 677 678 679
 680 681 682 684 688 689 690 692 693 694 696 697 698 699 700 702 703 704
 706 708 709 711 712 713 715 716 718 719 720 723 725 726 727 733 734 735
 737 739 740 741 742 743 744 745 746 749 750 751 756 757 758 761 764 766
 767 768 769 770 773 775 778 779 780 781 782 786 787 788 789 790 791 794
 800 802 803 806 807 811 812 833 839 857 886]
```

```
count
[  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414
 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450
 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468
 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486
 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504
 505 506 507 508 509 511 512 513 514 515 516 517 518 519 520 521 522 523
 524 525 526 527 528 529 530 531 532 533 534 536 537 538 539 540 541 542
 543 544 545 546 547 549 550 551 552 553 554 555 556 557 558 559 560 561
 562 563 564 565 566 567 568 569 570 571 572 573 575 576 577 578 579 580
```

```
 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598
 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616
 617 618 619 620 622 623 624 625 626 627 628 629 630 631 632 633 634 635
 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653
 654 655 656 657 658 659 660 661 662 663 665 666 667 668 669 670 671 672
 673 674 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691
 692 693 694 696 698 700 701 702 704 705 706 708 710 711 712 713 715 717
 719 721 722 723 724 725 729 730 731 732 733 734 737 738 739 741 743 744
 745 746 747 748 749 750 755 757 758 759 761 766 767 769 770 771 772 774
 775 776 777 779 781 782 783 784 785 788 790 791 792 793 794 795 797 798
 800 801 806 808 809 810 811 812 813 814 817 818 819 822 823 825 827 830
 831 832 834 835 837 838 839 842 843 844 846 848 849 850 851 852 854 856
 857 858 862 863 865 867 868 869 871 872 873 877 884 886 887 888 890 891
 892 894 897 900 901 917 925 943 948 968 970 977]
```

```
day
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

```
month
['Jan' 'Feb' 'Mar' 'Apr' 'May' 'Jun' 'Jul' 'Aug' 'Sep' 'Oct' 'Nov' 'Dec']
```

```
year
[2011 2012]
```

Insights - The following columns - season, holiday, workingday, weather needed their values to suit better analysis. - Also the days recorded are from 1 to 19.

2.5 Replacing the appropriate values for respective column

```
cols =  ['season', 'holiday', 'workingday', 'weather']
df['season'] = df['season'].replace({1: 'spring', 2: 'summer', 3: 'fall', 4:
'winter'})
df['holiday'] = df['holiday'].replace({0: 'No', 1: 'Yes'})
df['workingday'] = df['workingday'].replace({0: 'No', 1: 'Yes'})
df['weather'] = df['weather'].replace({1: 'Clear', 2: 'Mist', 3: 'Light Rain', 4:
'Heavy Rain'})
```

2.6 Checking for duplicate values

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

```
0
```

Insights - There are no duplicate values.

2.7 Comprehensive understanding of the dataset

```
df.describe().T
```

	count	mean	min	25%	50%
datetime	10886	2011-12-27 05:56:22.399411968	2011-01-01 00:00:00	2011-07-02 07:15:00	2012-01-01 20
temp	10886.0	20.23086	0.82	13.94	20.5
atemp	10886.0	23.655084	0.76	16.665	24.24
humidity	10886.0	61.88646	0.0	47.0	62.0
windspeed	10886.0	12.799395	0.0	7.0015	12.998
casual	10886.0	36.021955	0.0	4.0	17.0
registered	10886.0	155.552177	0.0	36.0	118.0
count	10886.0	191.574132	1.0	42.0	145.0
day	10886.0	9.992559	1.0	5.0	10.0
year	10886.0	2011.501929	2011.0	2011.0	2012.0

```
df.describe(include=['category']).T
```

	count	unique	top	freq
season	10886	4	winter	2734
holiday	10886	2	No	10575
workingday	10886	2	Yes	7412
weather	10886	4	Clear	7192

Insights - We have records for 2 years. - Numerical features casual and registered bike rentals have high standard deviations and small mean indicating outliers and lots of diverse ranges and distributions. - This can be an indication that the rental will be based on lots of conditions and will be different with each given condition.

3.0 Univariate Analysis

```
categorical = ['season', 'holiday', 'workingday', 'weather', 'month', 'year']
numerical = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered',
'count']
```

3.1 Numerical Variables

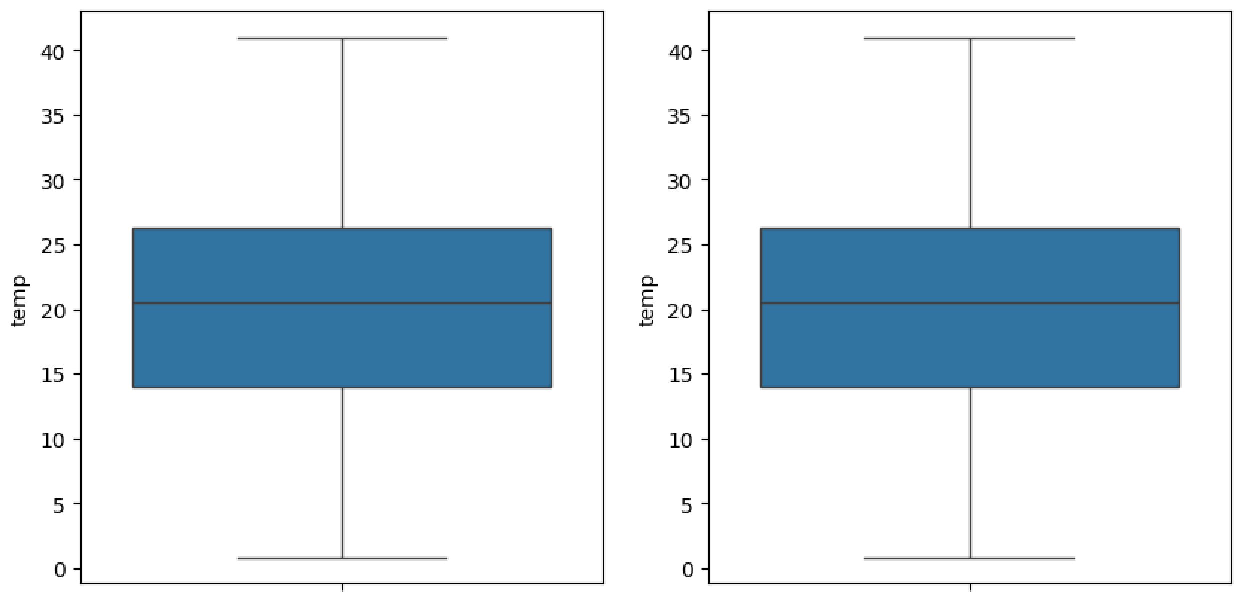
3.1.1Detecting Outliers

```
def det_out():
  for i in numerical:
    fig, ax = plt.subplots(1, 2, figsize=(10, 5))
    sns.boxplot(df[i], ax=ax[0])
    plt.suptitle(f'Box plot of {i} column: Full data vs IQR')

    # Calculating the Quantiles
    Q1 = df[i].quantile(0.25)
    Q3 = df[i].quantile(0.75)

    # Calculating the IQR
    IQR = Q3 - Q1

    # Setting lower and upper limit to seperate the outliers
    lower_limit = Q1 - 1.5*IQR
    upper_limit = Q3 + 1.5*IQR

    # Calulating the Total outliers
    lower_outliers = df[df[i] < lower_limit]
    upper_outliers = df[df[i] > upper_limit]
    tot_out = len(lower_outliers) + len(upper_outliers)
    print(f'Total number of the outliers in the {i} column is {tot_out}')

    # Removing Outliers
    remove_outliers =  df[(df[i] > lower_limit) & (df[i] < upper_limit)]

    # Boxplot with Outliers removed
    sns.boxplot(data = remove_outliers[i], ax = ax[1])

det_out()
```

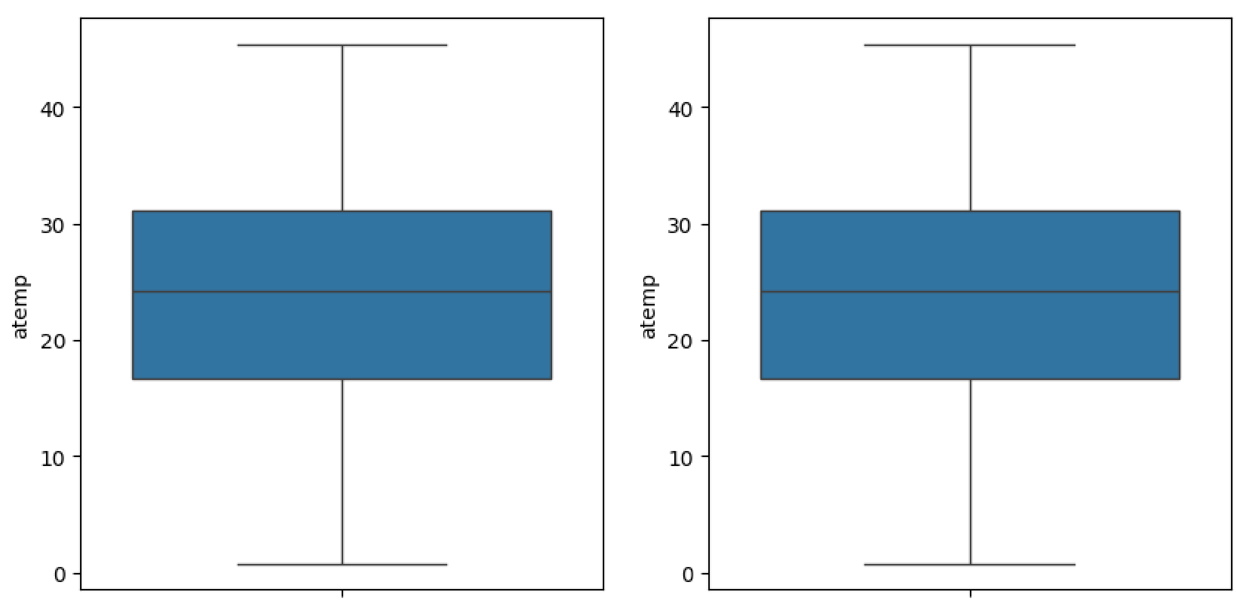
```
Total number of the outliers in the temp column is 0
Total number of the outliers in the atemp column is 0
Total number of the outliers in the humidity column is 22
Total number of the outliers in the windspeed column is 227
Total number of the outliers in the casual column is 749
```

```
Total number of the outliers in the registered column is 423
Total number of the outliers in the count column is 300
```

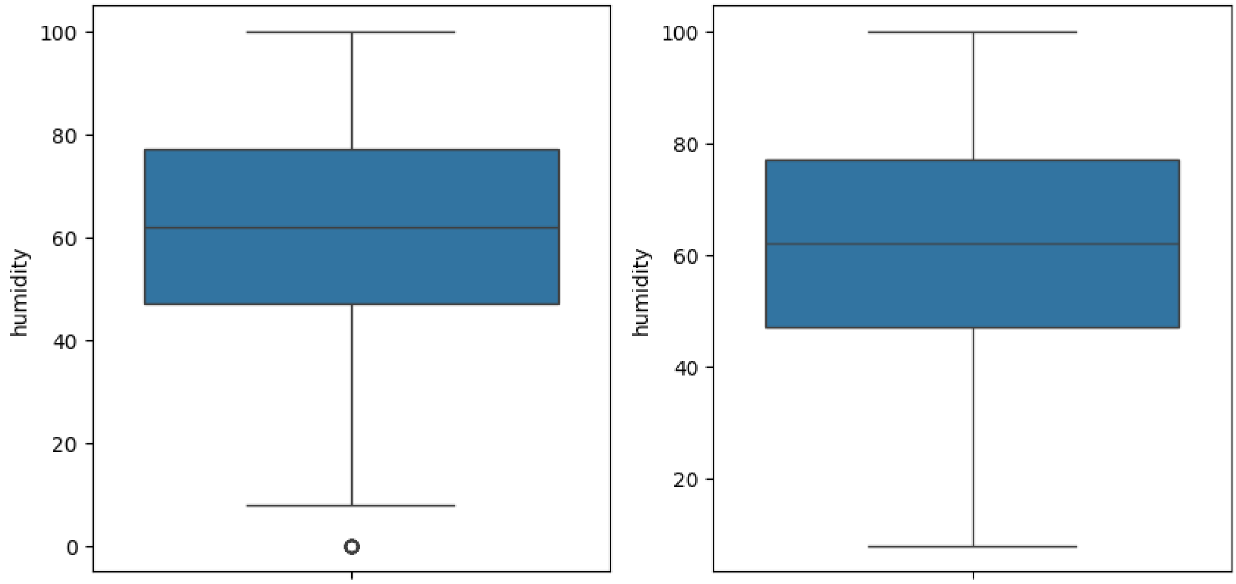
Box plot of temp column: Full data vs IQR



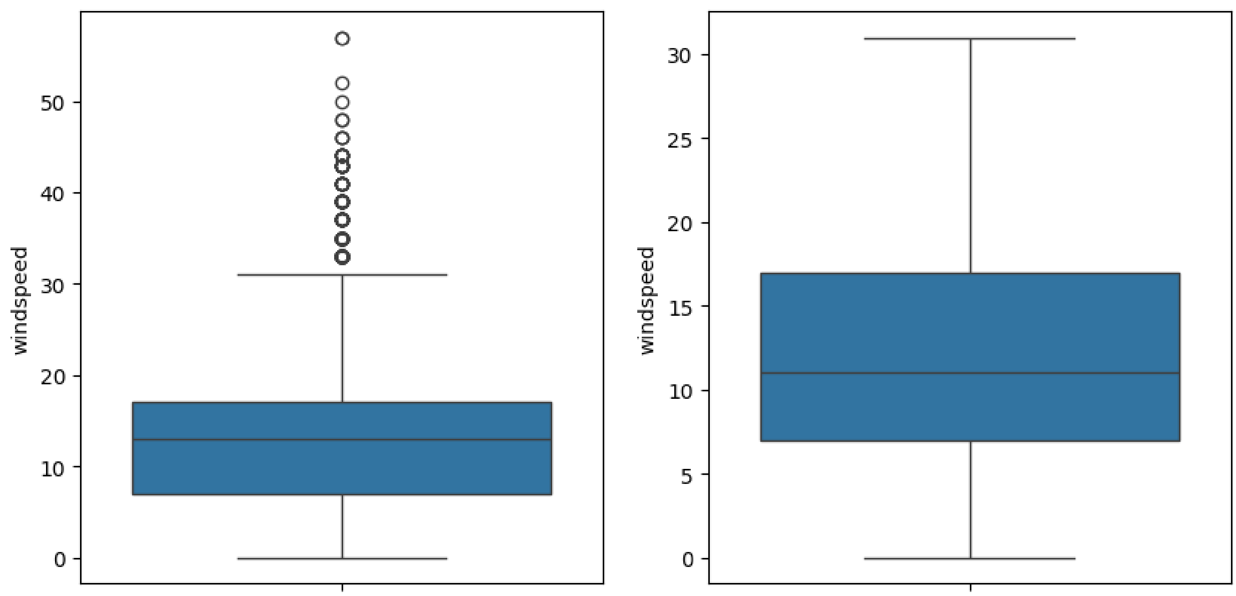
Box plot of atemp column: Full data vs IQR



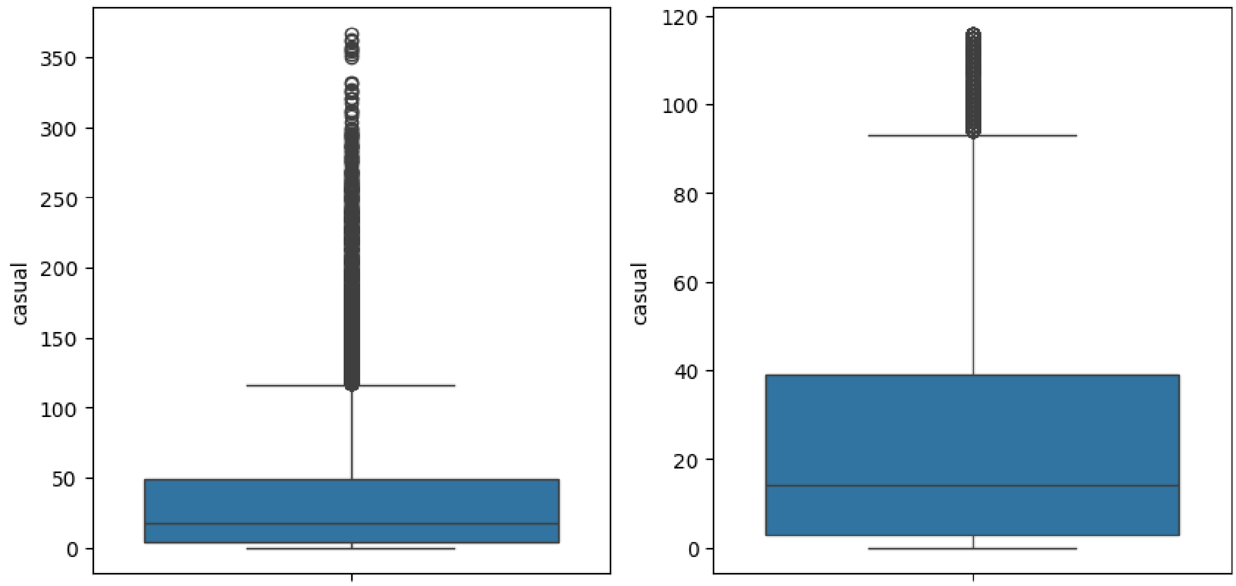
Box plot of humidity column: Full data vs IQR



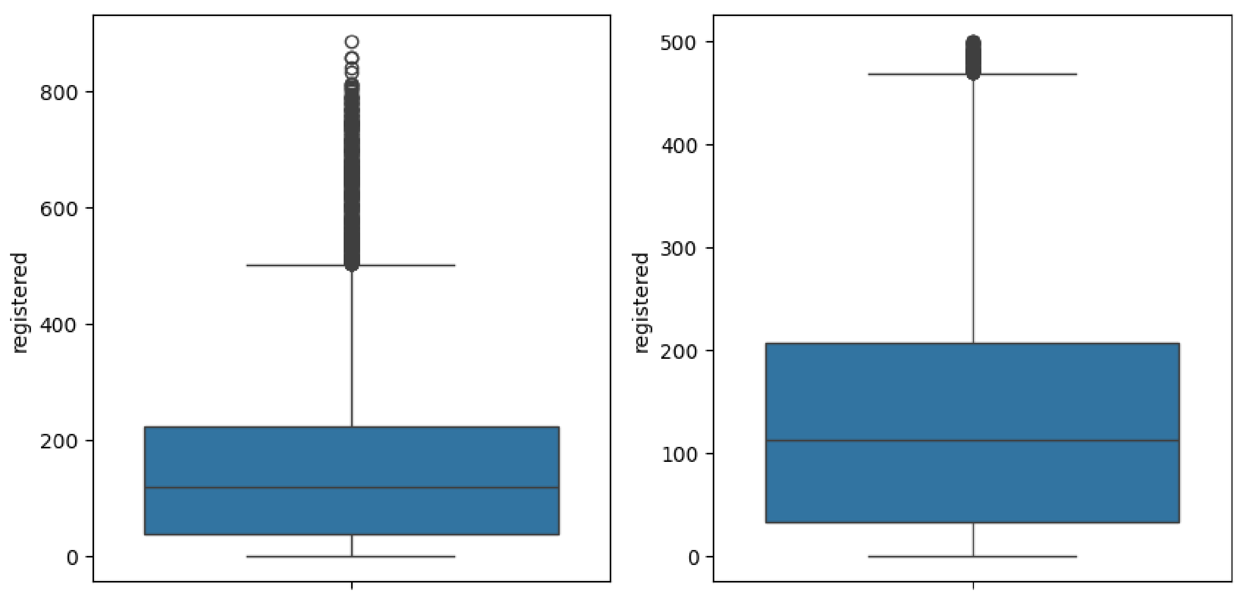
Box plot of windspeed column: Full data vs IQR

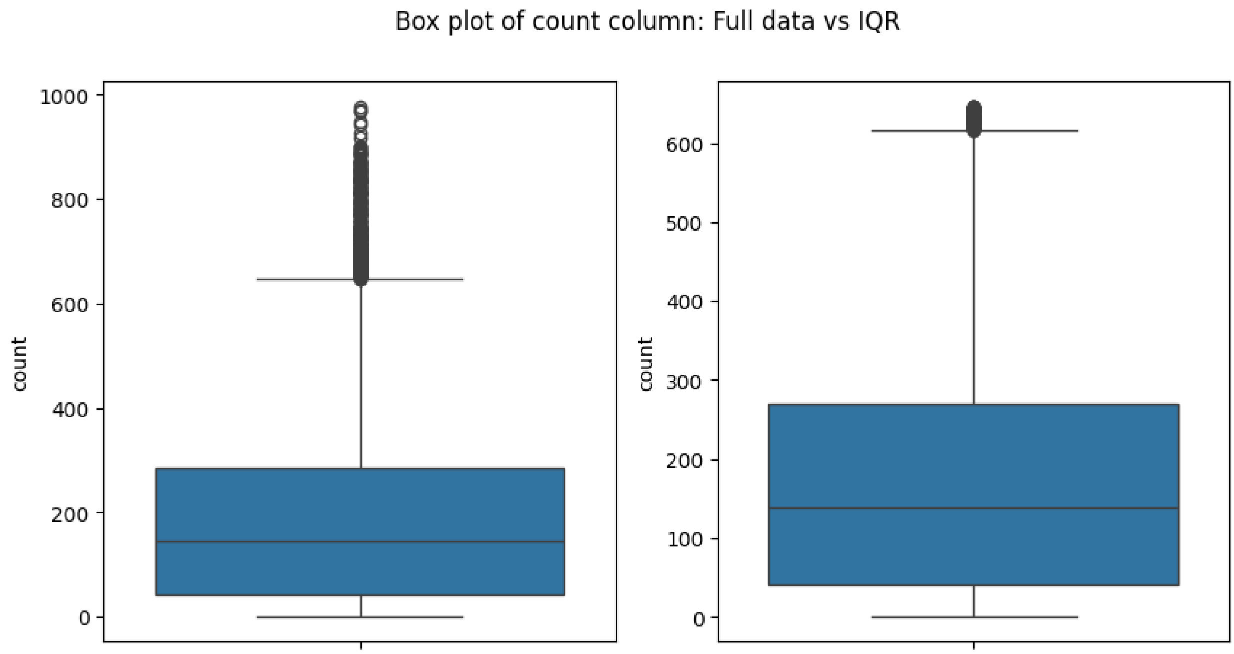


Box plot of casual column: Full data vs IQR



Box plot of registered column: Full data vs IQR

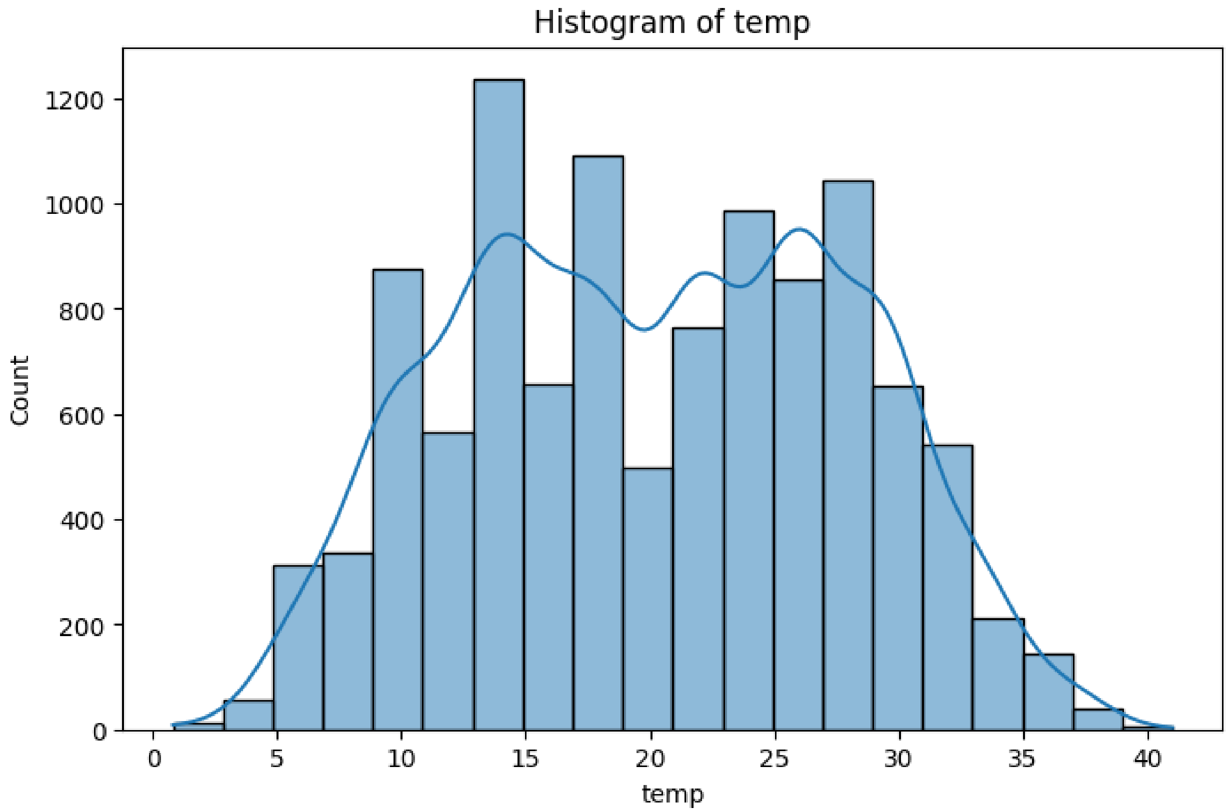




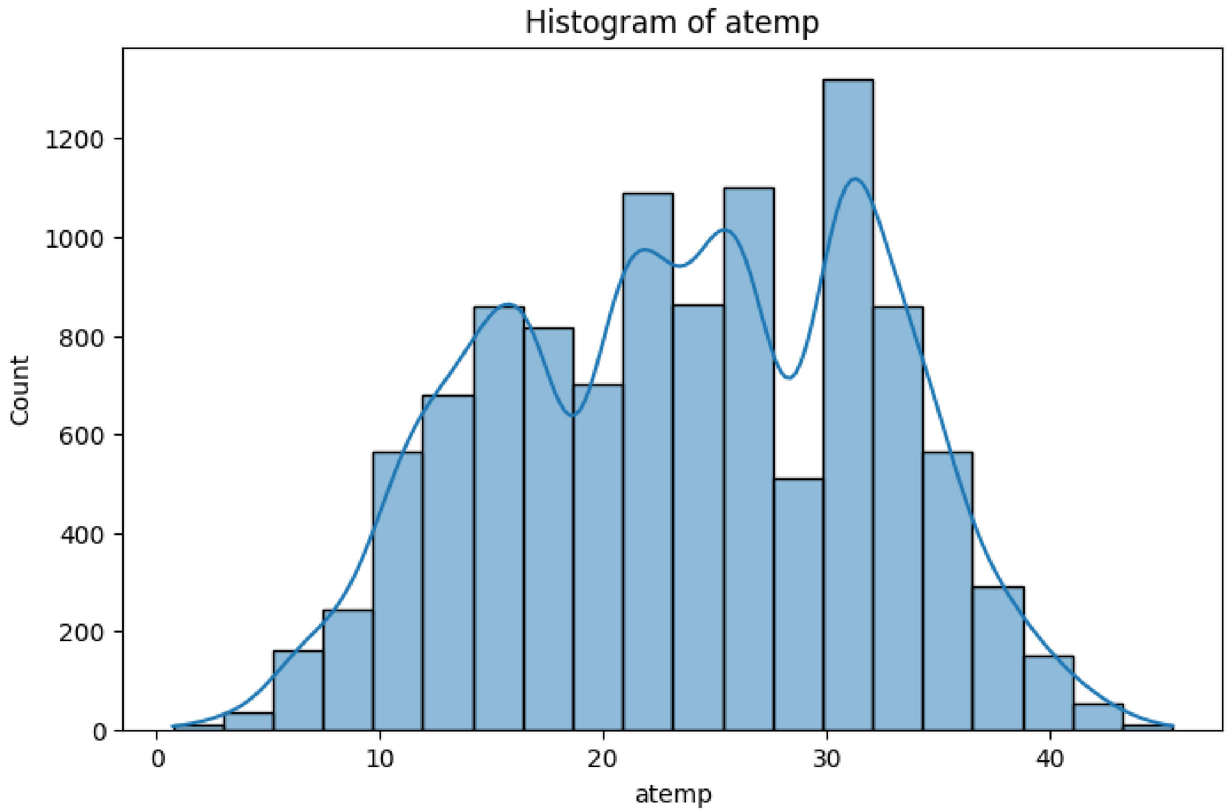
Insight- - As per our previous inference there are certainly lots of ouliers and they are removed using IQR. - Now we can further look at the dependencies for these outliers.

3.1.2 Analysis of numerical variable through Histplot

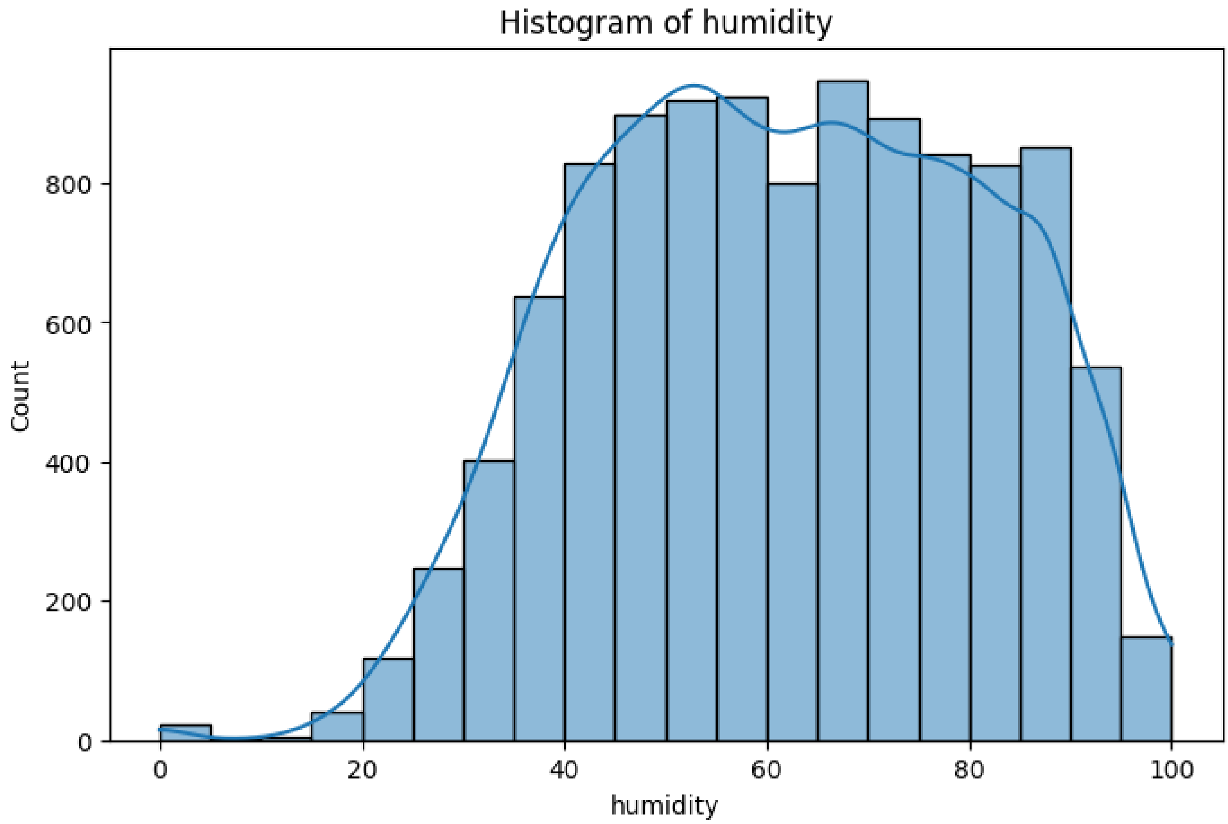
```
for i in numerical:
  fig, ax = plt.subplots(1, 1, figsize = (8,5))
  sns.histplot(df[i], bins=20, kde=True, ax = ax)
  plt.title(f'Histogram of {i}')
  plt.show()

  tabular_data = df[i].describe().reset_index()
  tabular_data.columns = ['Statistic', 'Value']
  display(tabular_data)
```



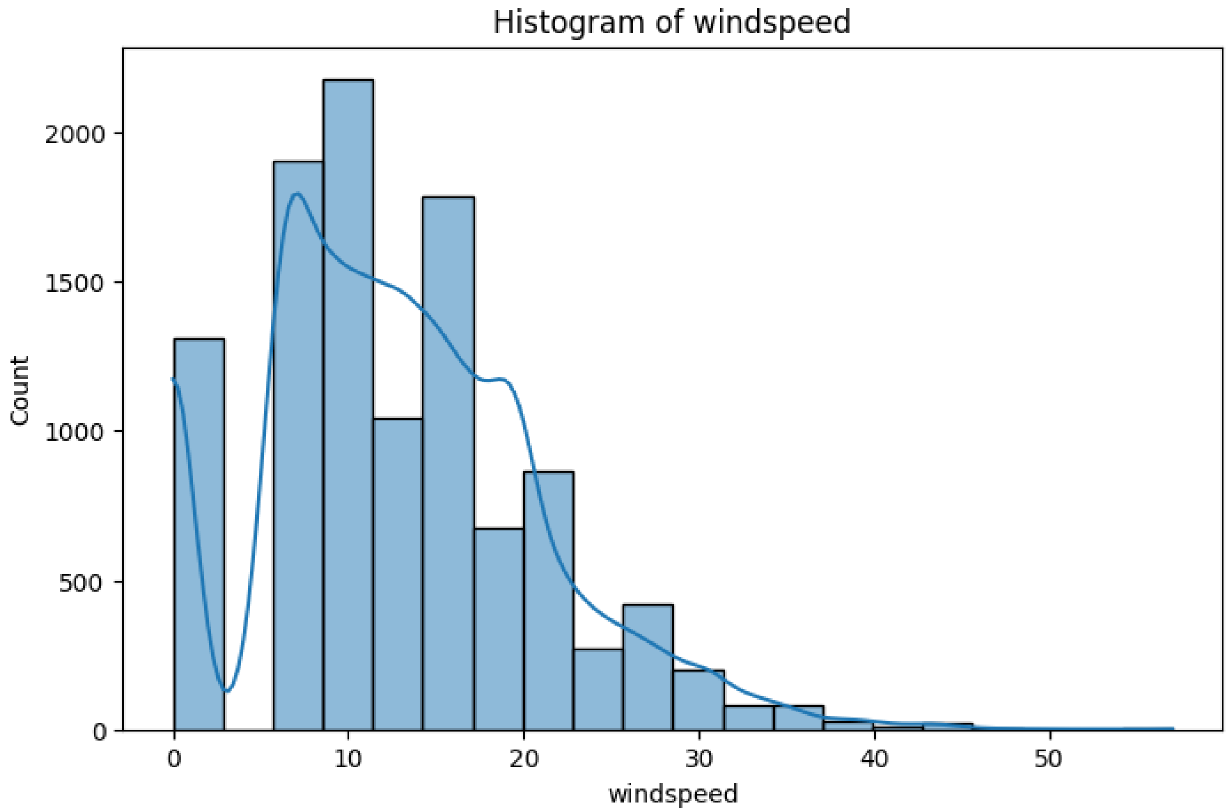
	Statistic	Value
0	count	10886.00000
1	mean	20.23086
2	std	7.79159
3	min	0.82000
4	25%	13.94000
5	50%	20.50000
6	75%	26.24000
7	max	41.00000



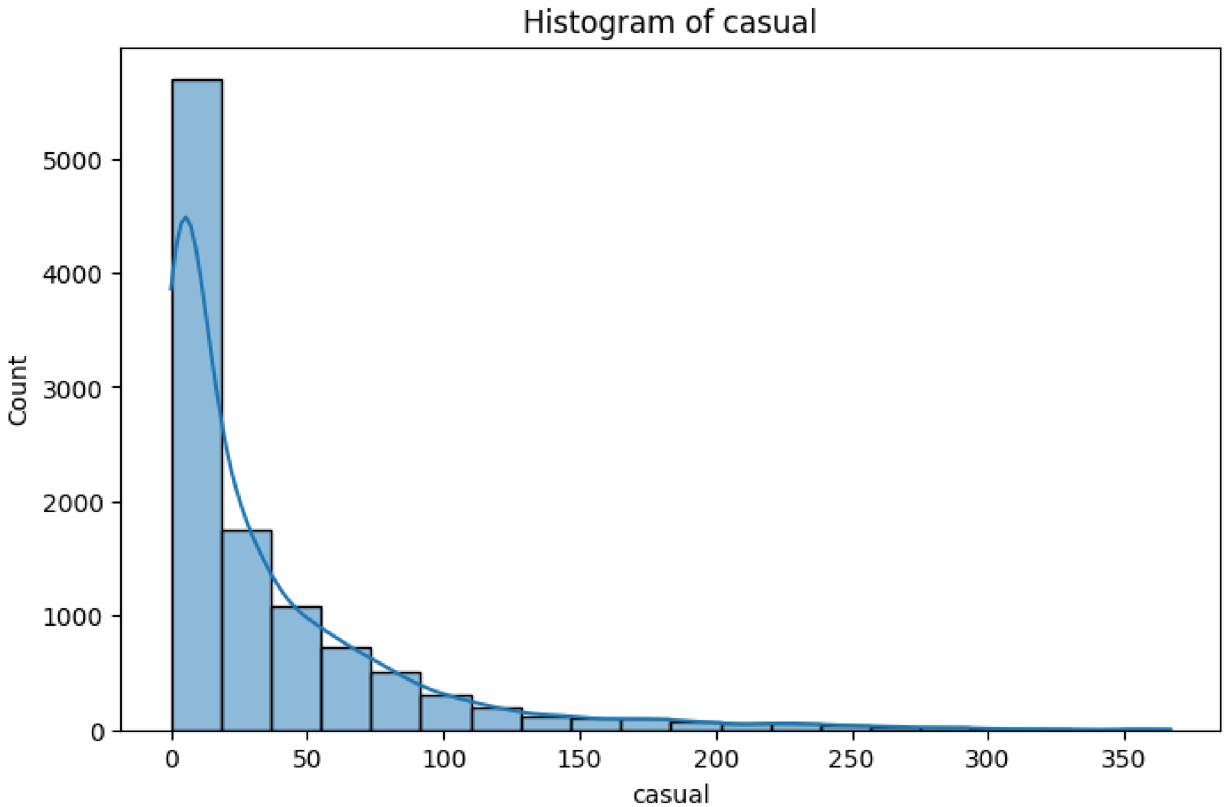
	Statistic	Value
0	count	10886.000000
1	mean	23.655084
2	std	8.474601
3	min	0.760000
4	25%	16.665000
5	50%	24.240000
6	75%	31.060000
7	max	45.455000



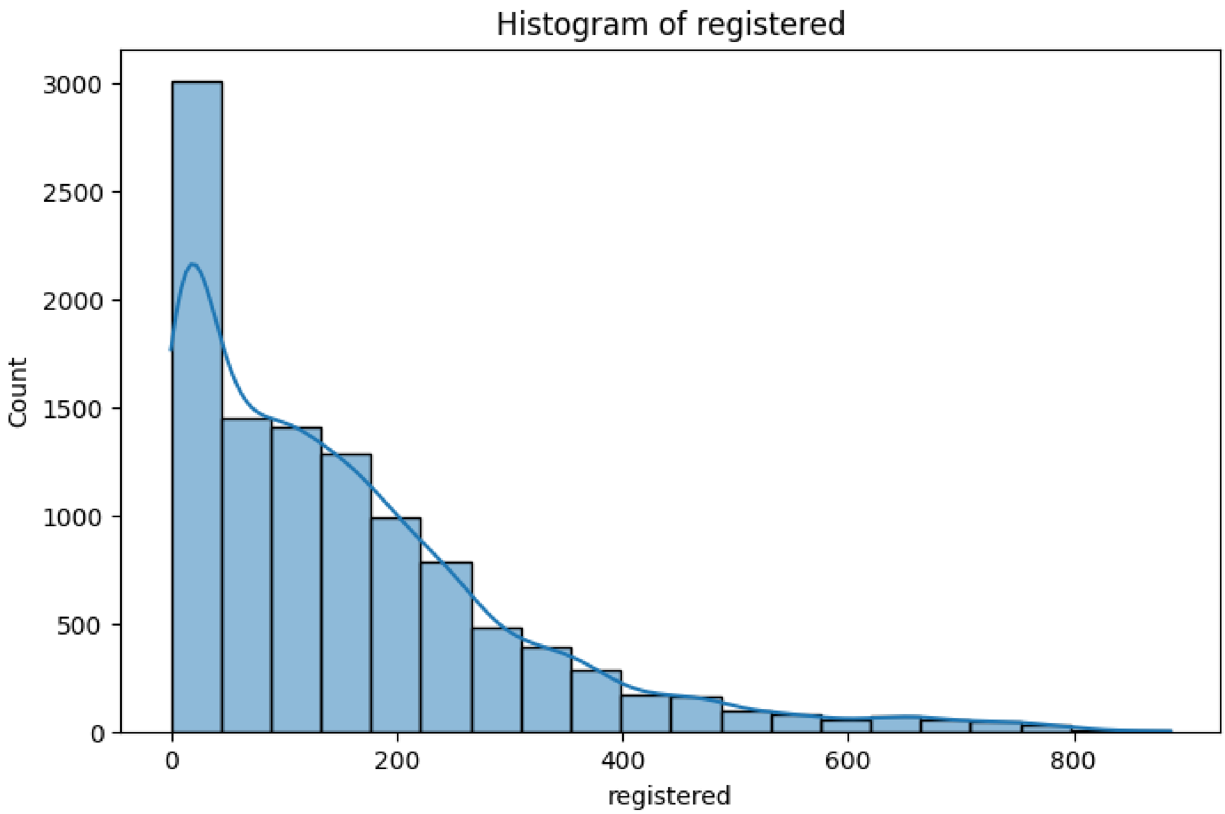
	Statistic	Value
0	count	10886.000000
1	mean	61.886460
2	std	19.245033
3	min	0.000000
4	25%	47.000000
5	50%	62.000000
6	75%	77.000000
7	max	100.000000



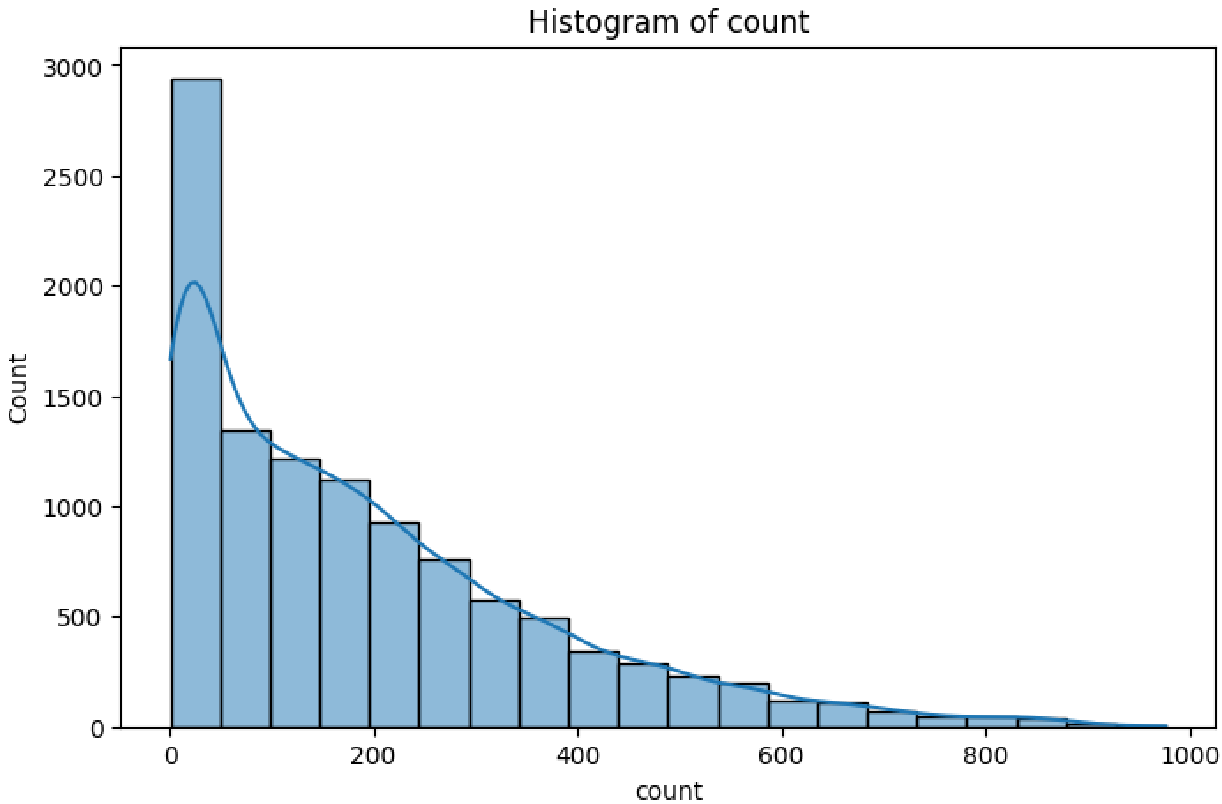
	Statistic	Value
0	count	10886.000000
1	mean	12.799395
2	std	8.164537
3	min	0.000000
4	25%	7.001500
5	50%	12.998000
6	75%	16.997900
7	max	56.996900



	Statistic	Value
0	count	10886.000000
1	mean	36.021955
2	std	49.960477
3	min	0.000000
4	25%	4.000000
5	50%	17.000000
6	75%	49.000000
7	max	367.000000



	Statistic	Value
0	count	10886.000000
1	mean	155.552177
2	std	151.039033
3	min	0.000000
4	25%	36.000000
5	50%	118.000000
6	75%	222.000000
7	max	886.000000



	Statistic	Value
0	count	10886.000000
1	mean	191.574132
2	std	181.144454
3	min	1.000000
4	25%	42.000000
5	50%	145.000000
6	75%	284.000000
7	max	977.000000

Insight - The columns such as casual, registered and count are right skewed. - Since log-normal distribution is a right skewed continuous probability distribution, we can safely assume it follow a log normal distribution. Therefore CLT could be an option. - It is safe to assume the natural featues almost looks like they follow a normal distribution.

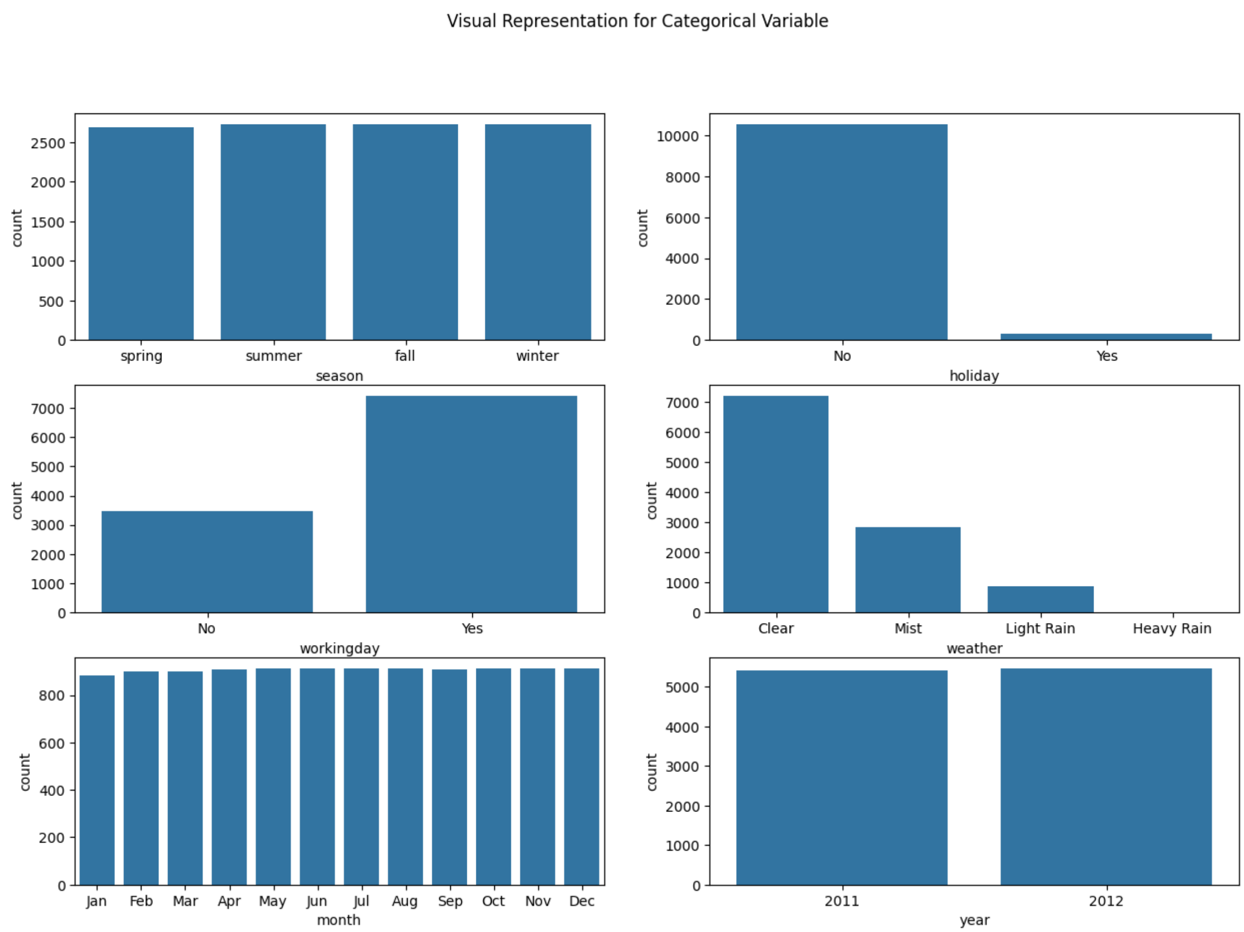
```
df['temp'].describe()
```

```
count    10886.00000
mean        20.23086
std          7.79159
min          0.82000
25%         13.94000
```

```
50%          20.50000
75%          26.24000
max          41.00000
Name: temp, dtype: float64
```

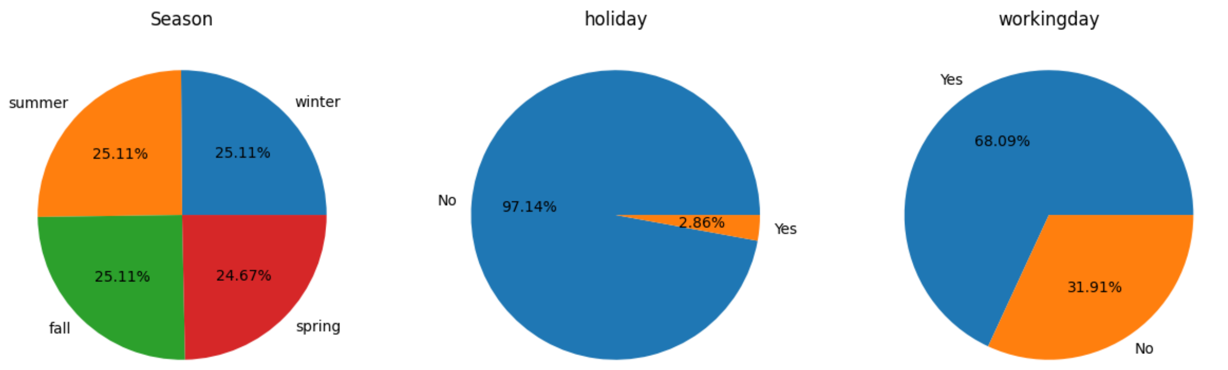
3.2 Categorical Variable

```python
# categorical = ['season', 'holiday', 'workingday', 'weather', 'month', 'year']
fig, axs= plt.subplots(3,2 , figsize=(15,10))
sns.countplot(data=df,x='season',ax=axs[0,0])
sns.countplot(data=df,x='holiday',ax=axs[0,1])
sns.countplot(data=df,x='workingday',ax=axs[1,0])
sns.countplot(data=df,x='weather',ax=axs[1,1])
sns.countplot(data=df,x='month',ax=axs[2,0])
sns.countplot(data=df,x='year',ax=axs[2,1])
plt.suptitle('Visual Representation for Categorical Variable')
plt.show()
```

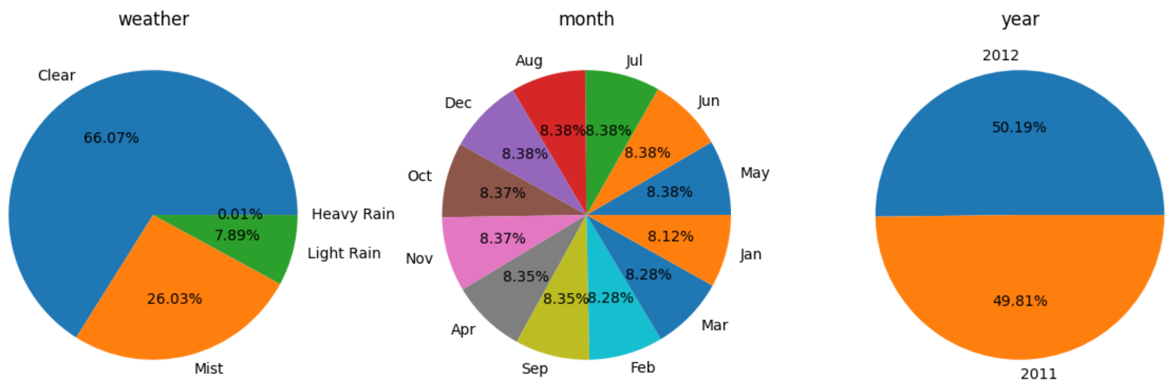


```python
fig, axis = plt.subplots(2, 3, figsize=(15,7))

plt.subplot(1,3,1)
pie = df['season'].value_counts().to_frame()
labels = pie.index
values = pie['count'].to_list()
plt.pie(values, labels = labels, autopct= '%0.2f%%')
plt.title('Season')

plt.subplot(1,3,2)
pie = df['holiday'].value_counts().to_frame()
labels = pie.index
values = pie['count'].to_list()
plt.pie(values, labels = labels, autopct= '%0.2f%%')
plt.title('holiday')


plt.subplot(1,3,3)
pie = df['workingday'].value_counts().to_frame()
labels = pie.index
values = pie['count'].to_list()
plt.pie(values, labels = labels, autopct= '%0.2f%%')
plt.title('workingday')


plt.show()
```

```
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
ax.remove() as needed.
  plt.subplot(1,3,1)
<ipython-input-589-df1dcd13d9b1>:10: MatplotlibDeprecationWarning: Auto-removal
of overlapping axes is deprecated since 3.6 and will be removed two minor
releases later; explicitly call ax.remove() as needed.
  plt.subplot(1,3,2)
<ipython-input-589-df1dcd13d9b1>:18: MatplotlibDeprecationWarning: Auto-removal
of overlapping axes is deprecated since 3.6 and will be removed two minor
releases later; explicitly call ax.remove() as needed.
  plt.subplot(1,3,3)
```




```
fig, axis = plt.subplots(1, 3, figsize=(15,7))

plt.subplot(1,3,1)
pie = df['weather'].value_counts().to_frame()
labels = pie.index
values = pie['count'].to_list()
plt.pie(values, labels = labels, autopct= '%0.2f%%')
plt.title('weather')


plt.subplot(1,3,2)
pie = df['month'].value_counts().to_frame()
labels = pie.index
values = pie['count'].to_list()
plt.pie(values, labels = labels, autopct= '%0.2f%%')
plt.title('month')


plt.subplot(1,3,3)
pie = df['year'].value_counts().to_frame()
labels = pie.index
values = pie['count'].to_list()
plt.pie(values, labels = labels, autopct= '%0.2f%%')
plt.title('year')


plt.show()
```



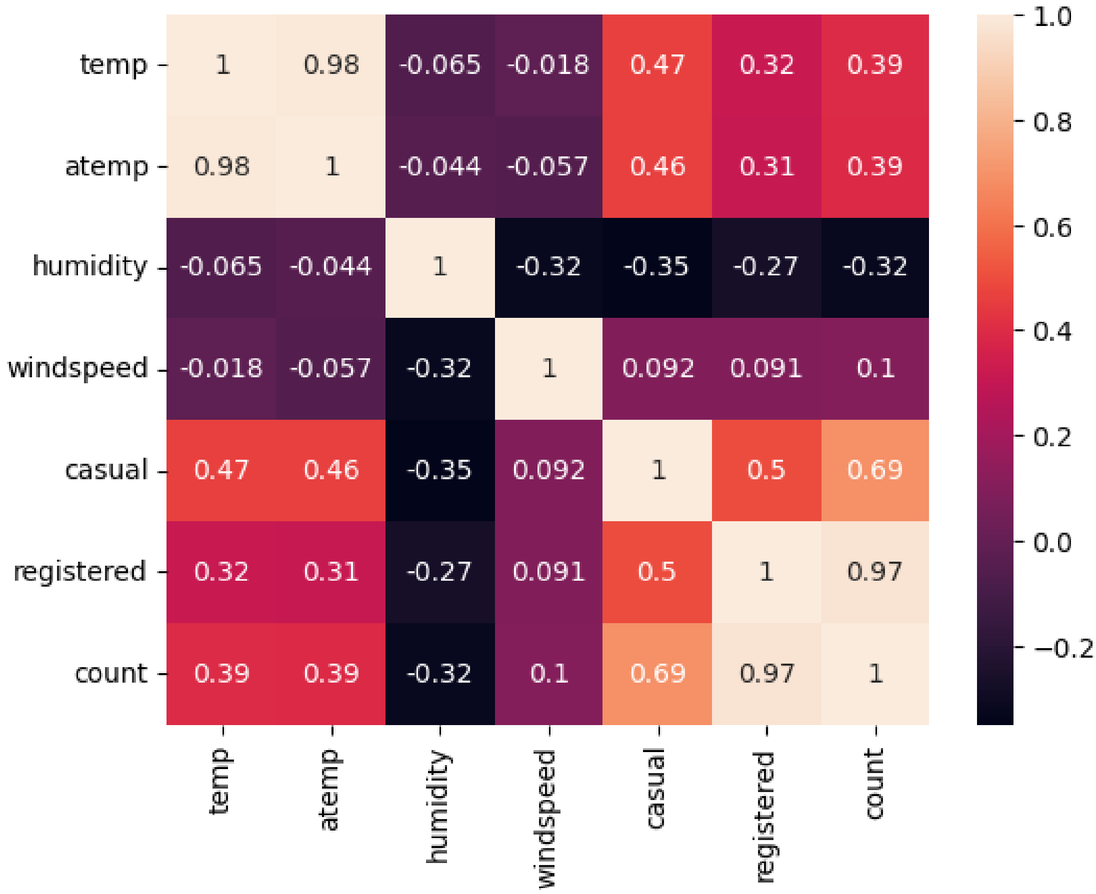
4.0 Relationship between the Dependent and Independent Variables.

4.1 Correlation Plot

```
corr_data = df[['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered',
'count']]
corr_data.corr()
```

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

```
sns.heatmap(corr_data.corr(), annot = True)

plt.show()
```



5.0 Hypothesis Testing

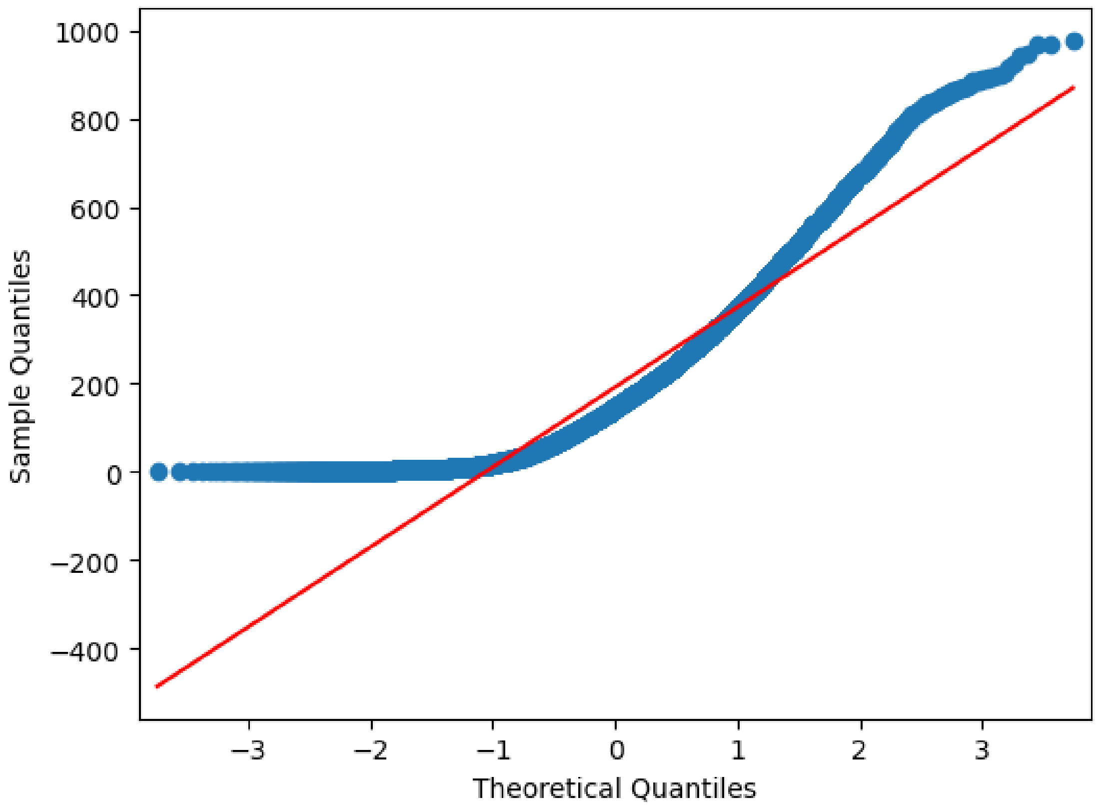
5.1 Check if there any significant difference between the no. of bike rides on Weekdays and Weekends?

We can consider Two Sample Independent T-Test. Meanwhile we have to check for normality before going ahead with the Two Sample Independent T-Test.

5.1.1 Test for Normality

QQ-Plot

```
from statsmodels.graphics.gofplots import qqplot

qqplot(df['count'], line = 's')
plt.show()
```



Insight- - It is clearly seen that it doesn't follow normal distribution but we will do an another test called Shapiro-Wilk to be more accurate.

Shapiro-Wilk test

```
from scipy.stats import shapiro
count_subset = df.sample(100, random_state= 42)['count']
test_stat, p_value = shapiro(count_subset)

print(p_value)

if p_value < 0.05:
    print("Reject H0")
    print("Data is not Gaussian")
else:
    print("Fail to reject H0")
    print("Data is Gaussian")
```

```
3.6810121173402877e-08
Reject H0
Data is not Gaussian
```

5.1.2 Test of Variance

Levene's Test - Since the it is not normally distributed, we can use Levene's Test to see whether this difference in variance is significant or not. - Then we can make decision on going ahead with Two sample independent T-Test or not.

```
from scipy.stats import levene

# Ho:  Variance across the group is similar
# Ha: Variance is not the same.

working_day = df[df['workingday'] == 'Yes']['count']
holiday = df[df['workingday'] == 'No']['count']

levene_stat, p_value = levene(working_day, holiday)

print(p_value)
if p_value < 0.05:
  print("Variances are not equal")
else:
  print("Variances are equal")
```

```
0.9437823280916695
Variances are equal
```

5.1.3 Two Sample Independent T-Test

```
from scipy.stats import ttest_ind

# Formulating Null Hypothesis (H0) and Alternate Hypothesis (Ha)

# Ho: Number of bike rides are same on weekdays and weekends.
# Ha: There is a significant difference between the number of bike rides on
weekdays and weekend.
# Two-Tailed

alpha = 0.05

ttest_stat, p_value = ttest_ind(working_day, holiday)
print(p_value)
if p_value < 0.05:
  print("Reject H0")
else:
  print("Fail to reject H0")
```

```
0.22644804226361348
Fail to reject H0
```

Insights - Therefore, There is no significant difference on bike rentals between working and non-working days.

5.2 Checking if the demand of bicycles on rent is the same for different Weather conditions.

5.2.1 Test for Normality

QQ-Plot - The test is done and verified already which shows the data is **not normally distributed**.

Shapiro-Wilk Test - The test is done and verified already which shows the data is **not normally distributed**.

Insights- - Previous test proves that the dataset doesn't follow a gaussian ditribution. So therefore we have to conduct other before deciding on One way Anova.

5.2.2 Skewness of weather w.r.t count

```python
df.groupby('weather')['count'].skew()
```

```
weather
Clear          1.139857
Mist           1.294444
Light Rain     2.187137
Heavy Rain          NaN
Name: count, dtype: float64
```

Insights- - The weather such as clear, mist and light rain are moderately right skewed. - Whereas there is no enough data for heavy rain to make an assumption.

5.2.3 Kurtosis test on different season.

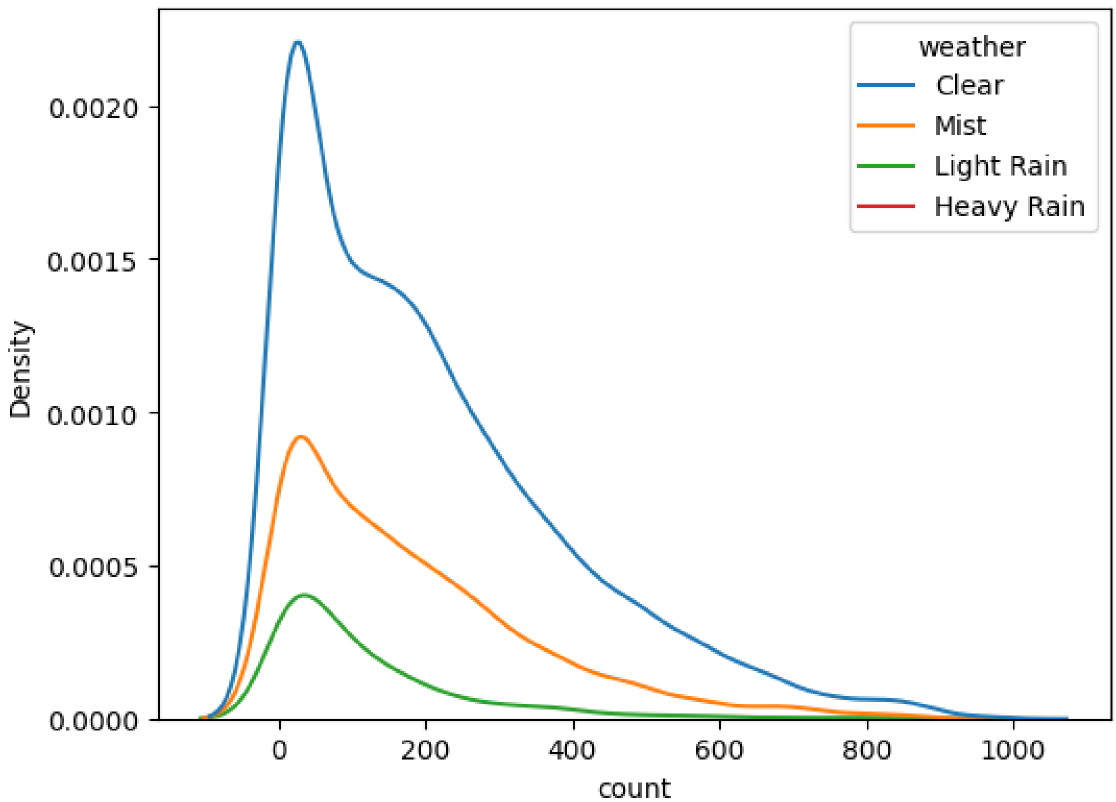
```python
wether_kurt = df.groupby('weather')['count'].apply(lambda x: x.kurtosis())
wether_kurt
```

```
weather
Clear          0.964720
Mist           1.588430
Light Rain     6.003054
Heavy Rain          NaN
Name: count, dtype: float64
```

Insights - Clear weather has a nearly normal distribution of bike counts with moderate outliers. - Mist weather shows more variability and outliers in bike counts. - Light Rain weather shows significant variability and many extreme values in bike counts. - Heavy Rain lacks sufficient data for analysis.

```
sns.kdeplot(data = df, x = 'count', hue = 'weather')
plt.show()
```

```
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
  sns.kdeplot(data = df, x = 'count', hue = 'weather')
```



Insight - This proves our above inference from kurtosis and skew test.

5.2.4 Levene's Test

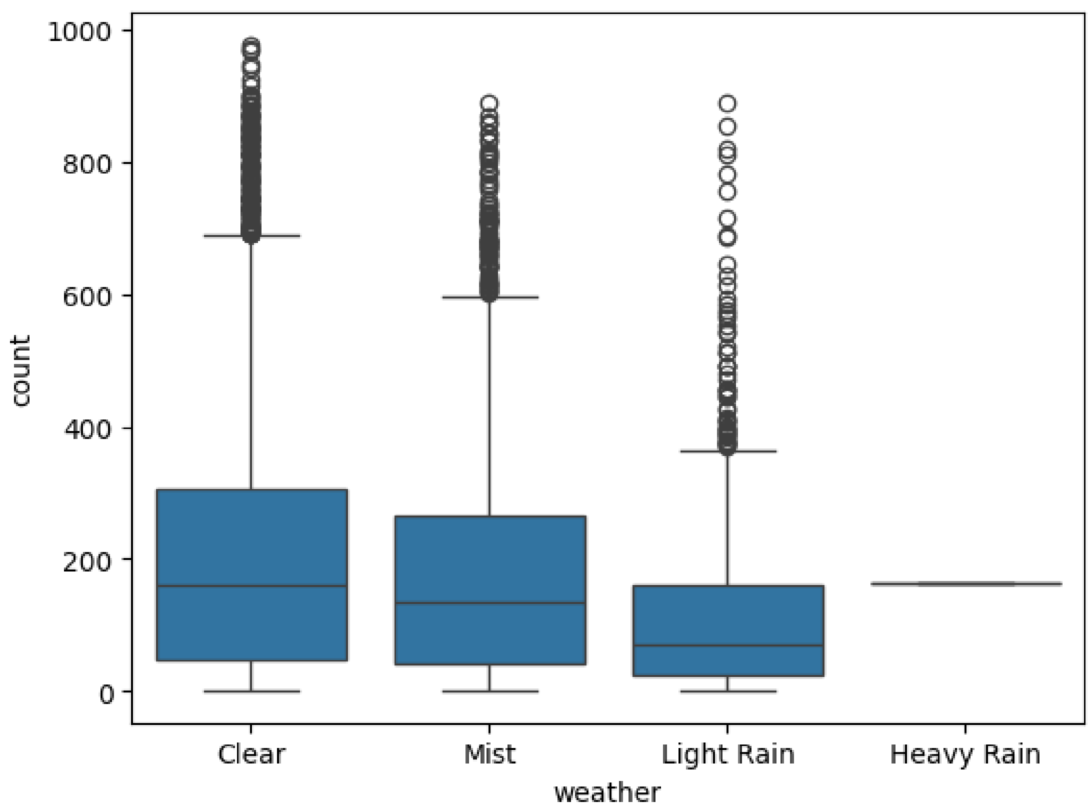
```
from scipy.stats import levene

# Ho:  Variance across the group is similar
# Ha: Variance is not the same.

clear = df[df['weather'] == 'Clear']['count']
```

```
mist = df[df['weather'] == 'Mist']['count']
light_rain = df[df['weather'] == 'Light Rain']['count']
heavy_rain = df[df['weather'] == 'Heavy Rain']['count']

levene_stat, p_value = levene(clear, mist, light_rain, heavy_rain)

print(p_value)
if p_value < 0.05:
  print("Variances are not equal")
else:
  print("Variances are equal")
```

```
3.504937946833238e-35
Variances are not equal
```

5.2.5 One-way ANOVA Test

```
sns.boxplot(x='weather', y='count', data=df)
plt.show()
```



Insight - This proves our above inference from kurtosis and skew test.

```
from scipy.stats import f_oneway
# H0: All groups have the same mean
# Ha: One or more groups have different mean
f_stats, p_value = f_oneway(clear, mist, light_rain, heavy_rain)

print("test statistic:",f_stats)
print("p_value:",p_value)

if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean")
else:
    print("Fail to reject H0")
    print("All groups have same mean")
```

```
test statistic: 65.53024112793271
p_value: 5.482069475935669e-42
Reject H0
Atleast one group have different mean
```

Insight- - After all the test we can say that, there is a significant difference between demand of bicycles for different Weather conditions.

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

5.3.1 Test for Normality

QQ-Plot - The test is done and verified already which shows the data is **not normally distributed**.

Shapiro-Wilk Test - The test is done and verified already which shows the data is **not normally distributed**.

Insights- - Previous test proves that the dataset doesn't follow a gaussian ditribution. So therefore we have conduct other before deciding on One way Anova.

5.3.2 Skewness of season w.r.t count

```
df.groupby('season')['count'].skew()
```

```
season
spring    1.888056
summer    1.003264
fall      0.991495
winter    1.172117
Name: count, dtype: float64
```

Insight - Spring shows the highest positive skewness, indicating a distribution with a significant number of high counts. - Summer, Fall, and Winter all have positive skewness values around 1, indicating moderately skewed distributions with a tendency towards higher counts, but not as extreme as spring.

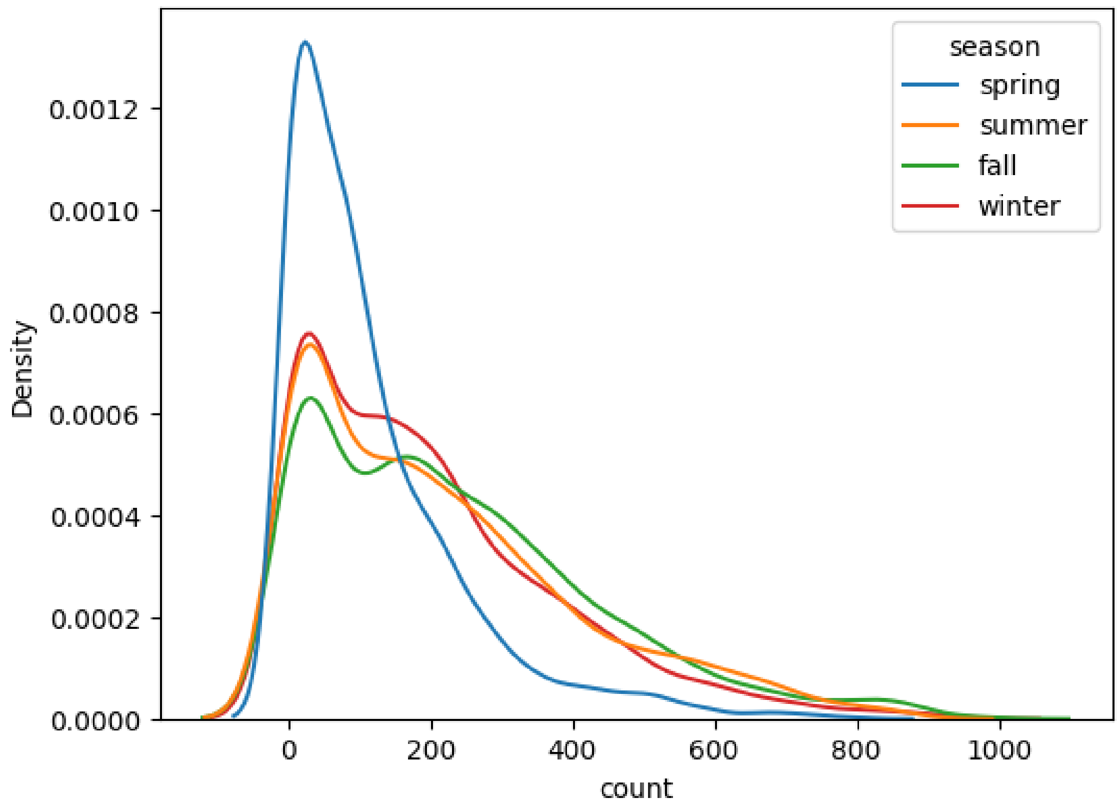
5.3.3 Kurtosis test on different season.

```
wether_kurt = df.groupby('season')['count'].apply(lambda x: x.kurtosis())
wether_kurt
```

```
season
spring    4.314757
summer    0.425213
fall      0.699383
winter    1.273485
Name: count, dtype: float64
```

Insights - Spring has the highest kurtosis, indicating a distribution with a significant number of extreme values or outliers and a sharper peak. - Winter has moderate kurtosis, showing some presence of extreme values but less pronounced than spring. - Summer and Fall have the lowest kurtosis, indicating flatter distributions with fewer extreme values or outliers.

```
sns.kdeplot(data = df, x = 'count', hue = 'season')
plt.show()
```



Insight - This proves our above inference from kurtosis and skew test.

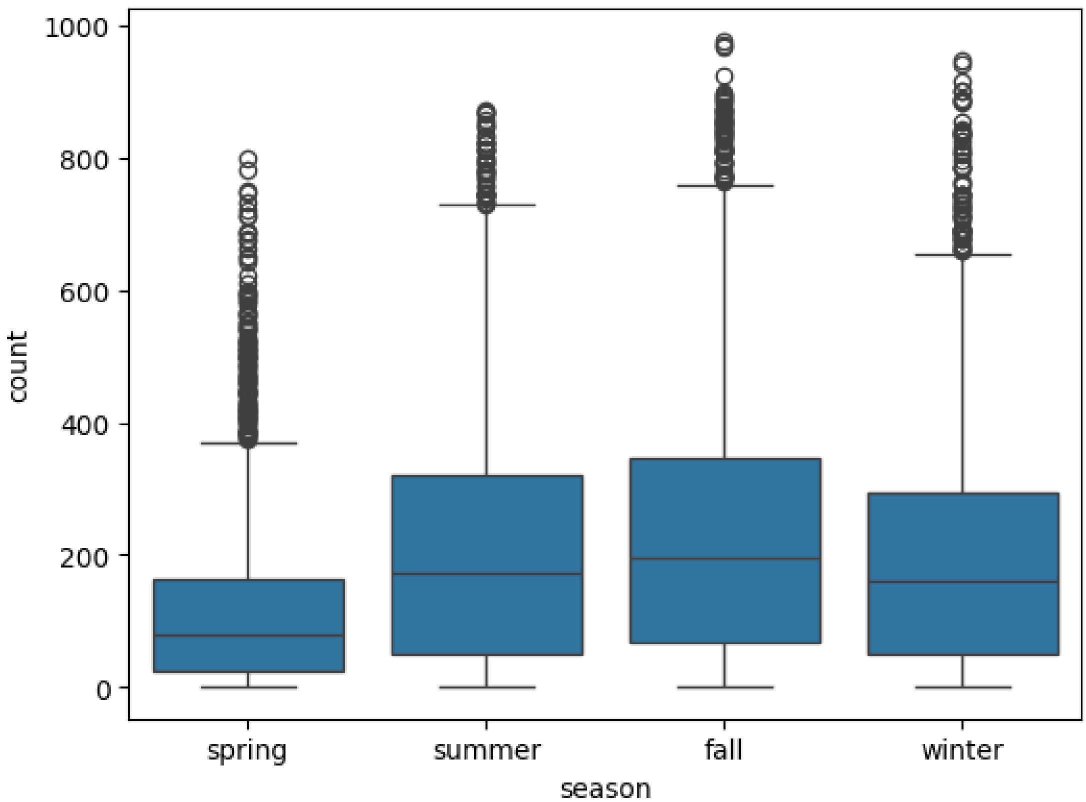
5.3.4 Levene's Test

```
from scipy.stats import levene

# Ho:  Variance across the group is similar
# Ha: Variance is not the same.

spring = df[df['season'] == 'spring']['count']
summer = df[df['season'] == 'summer']['count']
fall = df[df['season'] == 'fall']['count']
winter = df[df['season'] == 'winter']['count']

levene_stat, p_value = levene(spring, summer, fall, winter)

print(p_value)
if p_value < 0.05:
  print("Variances are not equal")
else:
  print("Variances are equal")
```

```
1.0147116860043298e-118
Variances are not equal
```

5.3.5 One-way ANOVA Test

```
sns.boxplot(x='season', y='count', data=df)
plt.show()
```



Insight - This proves our above inference from kurtosis and skew test.

```
from scipy.stats import f_oneway
# H0: All groups have the same mean
# Ha: One or more groups have different mean
f_stats, p_value = f_oneway(spring, summer, fall, winter)

print("test statistic:",f_stats)
print("p_value:",p_value)

if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean")
```

```
else:
    print("Fail to reject H0")
    print("All groups have same mean")
```

```
test statistic: 236.94671081032106
p_value: 6.164843386499654e-149
Reject H0
Atleast one group have different mean
```

Insight- - After all the test we can say that, There is a significant difference between demand of bicycles for different Seasons.

5.4 Checking if the Weather conditions are significantly different during different Seasons?

```
from scipy.stats import chi2_contingency

observed = pd.crosstab(df['weather'], df['season'])
print(observed)
```

```
season      spring  summer  fall  winter
weather
Clear         1759    1801  1930    1702
Mist           715     708   604     807
Light Rain     211     224   199     225
Heavy Rain       1       0     0       0
```

```
chi_stat, p_value, df, exp_freq = chi2_contingency(observed) # chi_stat, p_value,
df, expected values
print("chi_stat:",chi_stat)
print("p_value:",p_value)
print("df:",df)
print("exp_freq:",exp_freq)

alpha = 0.05

if p_value < alpha:
    print("Reject H0")
    print("Weather condition and seasons are not independent")
else:
    print("Fail to reject H0")
    print("Weather condition and seasons are independent")
```

```
chi_stat: 49.158655596893634
p_value: 1.549925073686486e-07
df: 9
exp_freq: [[1.77454639e+03 1.80559765e+03 1.80559765e+03 1.80625831e+03]
 [6.99258130e+02 7.11493845e+02 7.11493845e+02 7.11754180e+02]
 [2.11948742e+02 2.15657450e+02 2.15657450e+02 2.15736359e+02]
 [2.46738931e-01 2.51056403e-01 2.51056403e-01 2.51148264e-01]]
Reject H0
Weather condition and seasons are not independent
```

Insight - Based on the chi-squared test we performed, there is statistically significant dependency of weather and season based on the number of bikes rented.