1. Importing Data & Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import math
import matplotlib.pyplot as plt
from scipy.stats import norm
import scipy.stats as stats

from scipy.stats import ttest_rel,ttest_ind,ttest_1samp
from scipy.stats import chi2_contingency, chisquare
from scipy.stats import f_oneway, kruskal, shapiro, levene
from scipy.stats import spearmanr
from statsmodels.graphics.gofplots import qqplot
```

```
!gdown 1T2ibpd2XtkI_kdd718uUqX_55aBL5CuK
```

```
Downloading...
From: https://drive.google.com/uc?id=1T2ibpd2XtkI_kdd718uUqX_55aBL5CuK
To: /content/bike_sharing.csv
100% 648k/648k [00:00<00:00, 39.0MB/s]
```

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

```
df_og = df.copy() # Saving the original dataset
```

```
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windsp
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	
2	2011-01-01	1	0	0	1	9.02	13.635	80	

Next steps: Generate code with df | View recommended plots

2 Checking the Dataset

```
df.shape
```

```
(10886, 12)
```

```
df.columns
```

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

```
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
```

```
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
```

Insight:

- There are 10886 rows and 12 features
- There are no null values in the dataset.

2.1 Changing Data type for columns

```
# Renaming season column

def season(s):
  if s==1:
    return 'Spring'
  if s==2:
    return 'Summer'
  if s==3:
    return 'Fall'
  if s==4:
    return 'Winter'

# Renaming weather column

def weather(s):
  if s==1:
    return 'Clear'
  if s==2:
    return 'Cloudy'
  if s==3:
    return 'Light Rain'
  if s==4:
    return 'Heavy Rain'
```

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

```
df['holiday'] = df.holiday.apply(lambda x: 'holiday' if x == 1 else 'no_holiday')
```

```
df['workingday'] = df.workingday.apply(lambda x: 'working_day' if x == 1 else 'weekend/holiday')
```

```
df['weather'] = df.weather.apply(weather)
```

```
# Converting 'datetime' column to datetime
df['datetime'] = pd.to_datetime(df['datetime'])

# Converting 'season', 'holiday', 'workingday' columns to category

col = ['season', 'holiday', 'workingday', 'weather']

for i in col:
  df[col] = df[col].astype('category')
```

```
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
```

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

```
0
```

2.2 Sanity Check for all Columns

```
for col in df.columns:
  print("Unique Values in: ", col)
  print(df[col].unique())
  print("-" * 75)
```

```
 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]
---------------------------------------------------------------------------
Unique Values in:  count
[ 16  40  32  13   1   2   3   8  14  36  56  84  94 106 110  93  67  35
  37  34  28  39  17   9   6  20  53  70  75  59  74  76  65  30  22  31
   5  64 154  88  44  51  61  77  72 157  52  12   4 179 100  42  57  78
  97  63  83 212 182 112  54  48  11  33 195 115  46  79  71  62  89 190
 169 132  43  19  95 219 122  45  86 172 163  69  23   7 210 134  73  50
  87 187 123  15  25  98 102  55  10  49  82  92  41  38 188  47 178 155
  24  18  27  99 217 130 136  29 128  81  68 139 137 202  60 162 144 158
 117  90 159 101 118 129  26 104  91 113 105  21  80 125 133 197 109 161
 135 116 176 168 108 103 175 147  96 220 127 205 174 121 230  66 114 216
 243 152 199  58 166 170 165 160 140 211 120 145 256 126 223  85 206 124
 255 222 285 146 274 272 185 191 232 327 224 107 119 196 171 214 242 148
 268 201 150 111 167 228 198 204 164 233 257 151 248 235 141 249 194 259
 156 153 244 213 181 221 250 304 241 271 282 225 253 237 299 142 313 310
 207 138 280 173 332 331 149 267 301 312 278 281 184 215 367 349 292 303
 339 143 189 366 386 273 325 356 314 343 333 226 203 177 263 297 288 236
 240 131 452 383 284 291 309 321 193 337 388 300 200 180 209 354 361 306
 277 428 362 286 351 192 411 421 276 264 238 266 371 269 537 518 218 265
 459 186 517 544 365 290 410 396 296 440 533 520 258 450 246 260 344 553
 470 298 347 373 436 378 342 289 340 382 390 358 385 239 374 598 524 384
 425 611 550 434 318 442 401 234 594 527 364 387 491 398 270 279 294 295
 322 456 437 392 231 394 453 308 604 480 283 565 489 487 183 302 547 513
 454 486 467 572 525 379 502 558 564 391 293 247 317 369 420 451 404 341
 251 335 417 363 357 438 579 556 407 336 334 477 539 551 424 346 353 481
 506 432 409 466 326 254 463 380 275 311 315 360 350 252 328 476 227 601
 586 423 330 569 538 370 498 638 607 416 261 355 552 208 468 449 381 377
 397 492 427 461 422 305 375 376 414 447 408 418 457 545 496 368 245 596
 563 443 562 229 316 402 287 372 514 472 511 488 419 595 578 400 348 587
 497 433 475 406 430 324 262 323 412 530 543 413 435 555 523 441 529 532
 585 399 584 559 307 582 571 426 516 465 329 483 600 570 628 531 455 389
 505 359 431 460 590 429 599 338 566 482 568 540 495 345 591 593 446 485
 393 500 473 352 320 479 444 462 405 620 499 625 395 528 319 519 445 512
 471 508 526 509 484 448 515 549 501 612 597 464 644 712 676 734 662 782
 749 623 713 746 651 686 690 679 685 648 560 503 521 554 541 721 801 561
 573 589 729 618 494 757 800 684 744 759 822 698 490 536 655 643 626 615
 567 617 632 646 692 704 624 656 610 738 671 678 660 658 635 681 616 522
 673 781 775 576 677 748 776 557 743 666 813 504 627 706 641 575 639 769
 680 546 717 710 458 622 705 630 732 770 439 779 659 602 478 733 650 873
 846 474 634 852 868 745 812 669 642 730 672 645 694 493 668 647 702 665
 834 850 790 415 724 869 700 793 723 534 831 613 653 857 719 867 823 403
 693 603 583 542 614 580 811 795 747 581 722 689 849 872 631 649 819 674
 830 814 633 825 629 835 667 755 794 661 772 657 771 777 837 891 652 739
 865 767 741 469 605 858 843 640 737 862 810 577 818 854 682 851 848 897
 832 791 654 856 839 725 863 808 792 696 701 871 968 750 970 877 925 977
 758 884 766 894 715 783 683 842 774 797 886 892 784 687 809 917 901 887
 785 900 761 806 507 948 844 798 827 670 637 619 592 943 838 817 888 890
 788 588 606 608 691 711 663 731 708 609 688 636]
---------------------------------------------------------------------------
```

There are no abnormal values in any columns

2.3 Statistical Summary

```
df.describe(include = 'category')
```

	season	holiday	workingday	weather
count	10886	10886	10886	10886
unique	4	2	2	4
top	Winter	no_holiday	working_day	Clear
freq	2734	10575	7412	7192

Insight

- **season:** There are 4 different seasons.
- **holiday:** The data is segregated into 2 categories holiday and no_holiday. Out of 10886, `97.14%` of the times it was not a holiday.
- **workingday:** The data is segregated into 2 categories working and holiday/ weekend. Out of 10886, `68.08%` of the times it was a working day.
- **weather:** There are 4 different weather conditions. 66.06% of the time the weather was Clear, Few clouds or partly cloudy

```
df.describe()
```

	datetime	temp	atemp	humidity	windspeed	casual	registered	count
count	10886	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	2011-12-27 05:56:22.399411968	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
min	2011-01-01 00:00:00	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	2011-07-02 07:15:00	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	2012-01-01 20:30:00	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	2012-07-01 12:45:00	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	2012-12-19 23:00:00	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000
std	NaN	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454

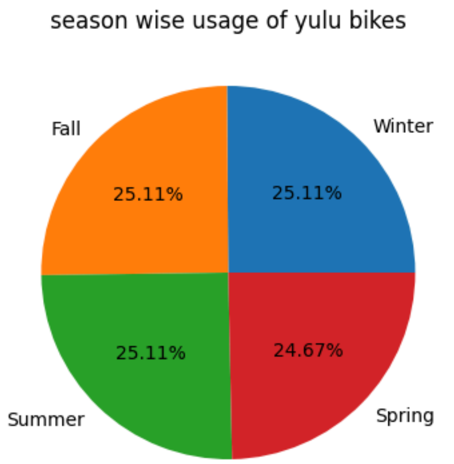
Insight

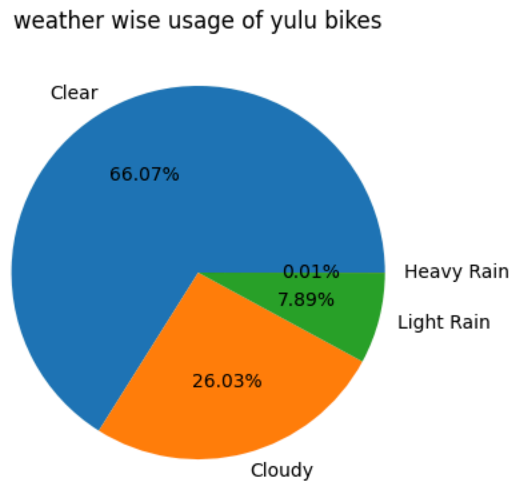
- **temp:** The range for temperature varies from 0.82 Celsius to 41 Celsius.
- **atemp:** The range for feeling temperature varies from 0.76 Celsius to 45.45 Celsius.
- **humidity:** 50% of the times was between 47 and 77.
- **windspeed:** 50% of the times was between 7.001500 and 16.997900. Max windspeed recorded was 56.99 indicating that there are outliers.
- **users:** [casual, registered, count] The median values for all type of users is less than mean values. We can expect a right skewed data distribution.

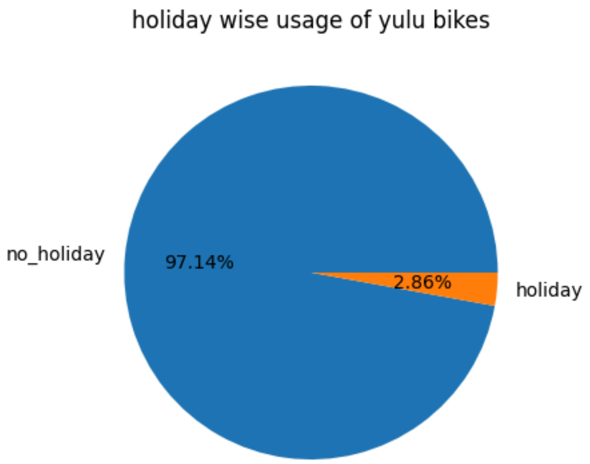
2.4 Graphical Analysis

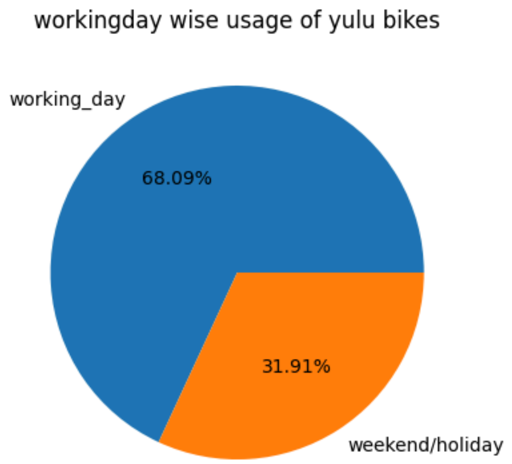
2.4.1 Analysing Categorical Columns

```
col = ['season', 'weather', 'holiday', 'workingday']

plt.figure(figsize=(12,10))
for i in range(len(col)):
  plt.subplot(2,2,i+1)
  plt.pie(df[col[i]].value_counts().values, labels = df[col[i]].value_counts().index, radius=1, autopct='%1.2f%%')
  plt.title(f'{col[i]} wise usage of yulu bikes')
```









Insight

- **season:** All four seasons share equal proportion except Spring season is slightly low at 24.67%
- **weather:** 92.10% of the times the weather was clear or cloudy while it rained only 7.90% of the times.
- **holiday:** 97.4% of the times there was no holiday.
- **working day:** Weekends or holidays combined contributes to 32%.

2.4.1 Converting Temperature to Categorical Column

```
temp_bin = [0,13,21,30,50]
temp_label = ['low', 'moderate', 'moderate-high', 'high']

df['temp_group'] = pd.cut(df['temp'], bins = temp_bin, labels = temp_label)
df['temp_group'] = df['temp_group'].astype('category')

df['temp_group'].value_counts()
```

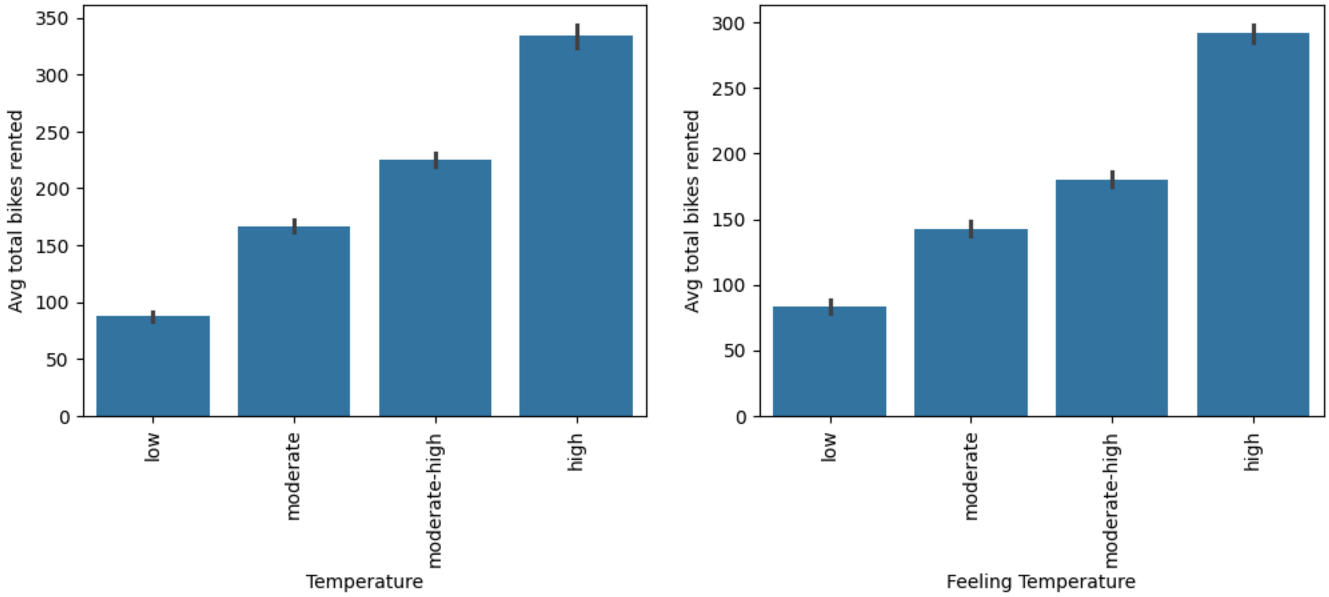
```
temp_group
moderate-high    4007
moderate         3478
low              2157
high             1244
Name: count, dtype: int64
```

```
df['atemp_group'] = pd.cut(df['atemp'], bins = temp_bin, labels = temp_label)
df['atemp_group'] = df['atemp_group'].astype('category')

df['atemp_group'].value_counts()
```

```
atemp_group
moderate-high    3564
high             3250
moderate         2616
```

```
low              1456
Name: count, dtype: int64
```

```
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
sns.barplot(data=df, x='temp_group', y='count', estimator='mean')
plt.ylabel('Avg total bikes rented')
plt.xlabel('Temperature')
plt.xticks(rotation=90)

plt.subplot(1,2,2)
sns.barplot(data=df, x='atemp_group', y='count', estimator='mean')
plt.xticks(rotation=90)
plt.ylabel('Avg total bikes rented')
plt.xlabel('Feeling Temperature')
plt.show()
```

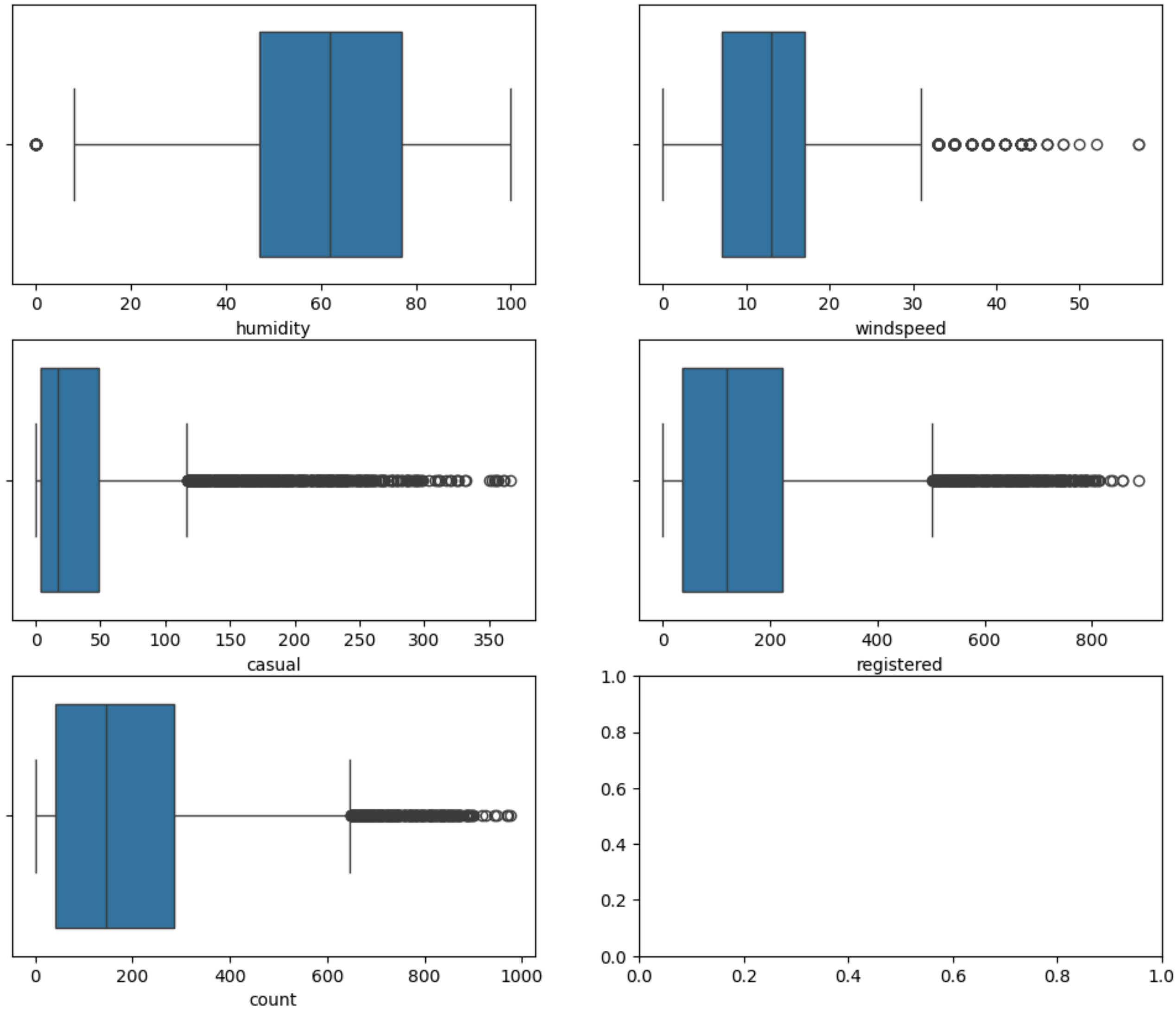


Insight

- People prefer using electric bike when the temperature is moderately high or high i.e when the **temperature is above 21 Celcius**

2.4.2 Checking for Outliers

```
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(12, 10))

sns.boxplot(data = df, x = "humidity", ax=axis[0,0])

sns.boxplot(data = df, x = 'windspeed', ax=axis[0,1])

sns.boxplot(data = df, x = 'casual', ax=axis[1,0])

sns.boxplot(data = df, x = 'registered', ax=axis[1,1])

sns.boxplot(data = df, x = 'count', ax=axis[2,0])

plt.show()
```




```
# Get Numerical columns

num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']


def Get_Numerical_Outlier_indices(df, cols):
    out_ind = []
    for col in cols:
        q1 = df[col].quantile(0.25)
        q2 = df[col].quantile(0.75)
        iqr = q2-q1
        rare_ind = df[((df[col]<(q1-(1.5*iqr)))|(df[col]>(q2+(1.5*iqr))))].index
        out_ind.extend(rare_ind)

    out_ind = set(out_ind)
    return out_ind

numerical_outlier_indices = Get_Numerical_Outlier_indices(df, num_cols)



outlier_len = len(numerical_outlier_indices)   #number of outliers in dataset
orig_len =  len(df)
print(f'original length of data: {orig_len}')
print(f'outliers length: {outlier_len}')
```

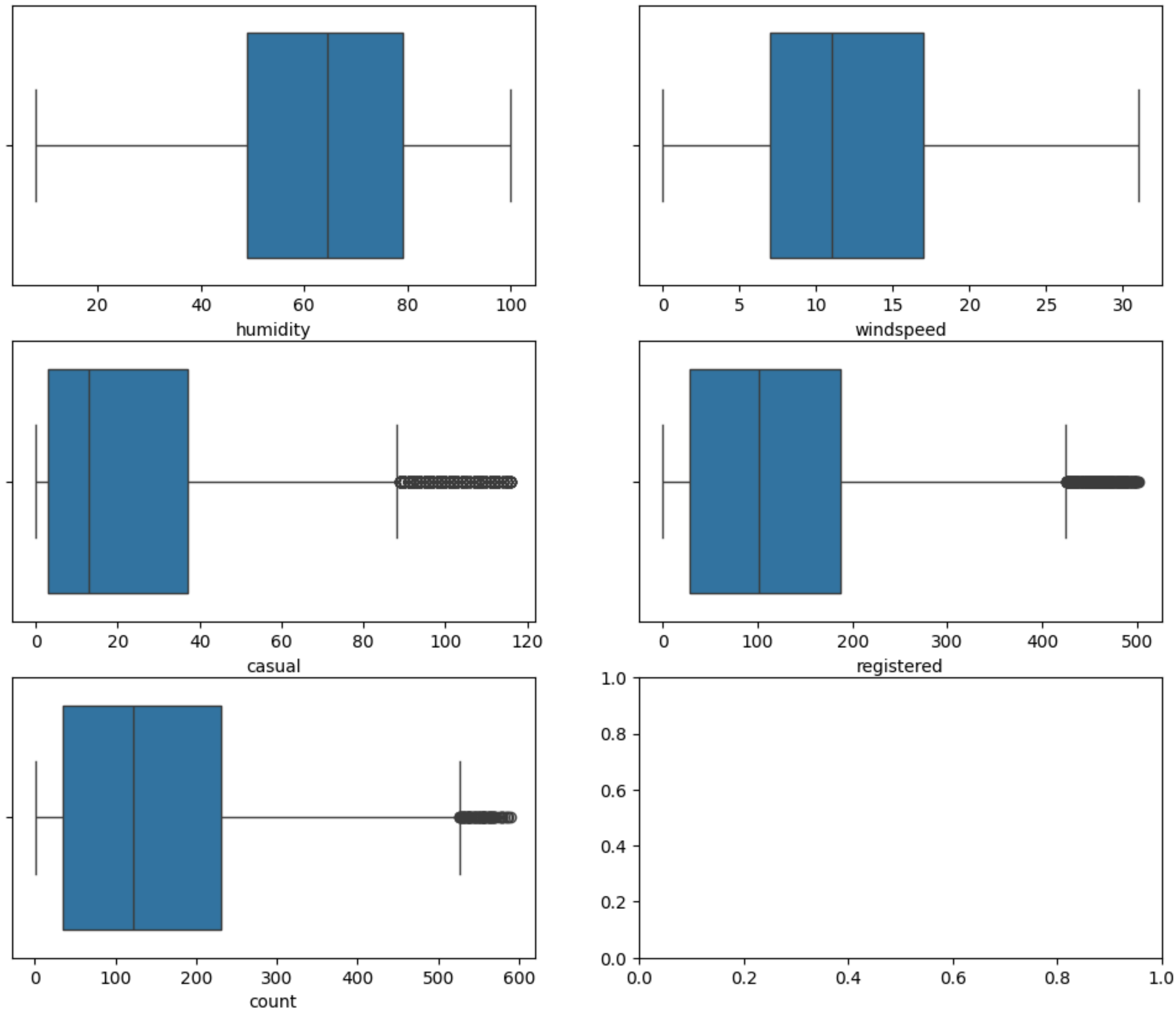
```
     original length of data: 10886
     outliers length: 1368
```

```
df = df.drop(numerical_outlier_indices)

df.shape
```

```
     (9518, 14)
```

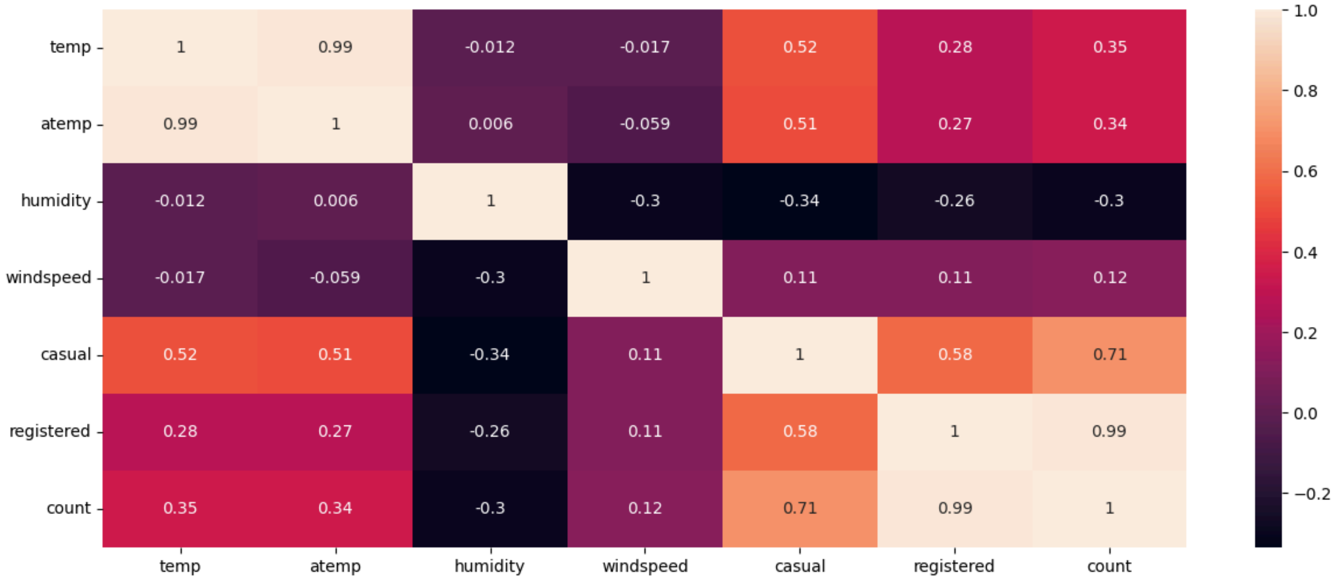
2.4.3 After Removing the Outliers

```
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(12, 10))

sns.boxplot(data = df, x = "humidity", ax=axis[0,0])

sns.boxplot(data = df, x = 'windspeed', ax=axis[0,1])

sns.boxplot(data = df, x = 'casual', ax=axis[1,0])

sns.boxplot(data = df, x = 'registered', ax=axis[1,1])

sns.boxplot(data = df, x = 'count', ax=axis[2,0])

plt.show()
```



3. Relationship between Dependent and Independent Variables

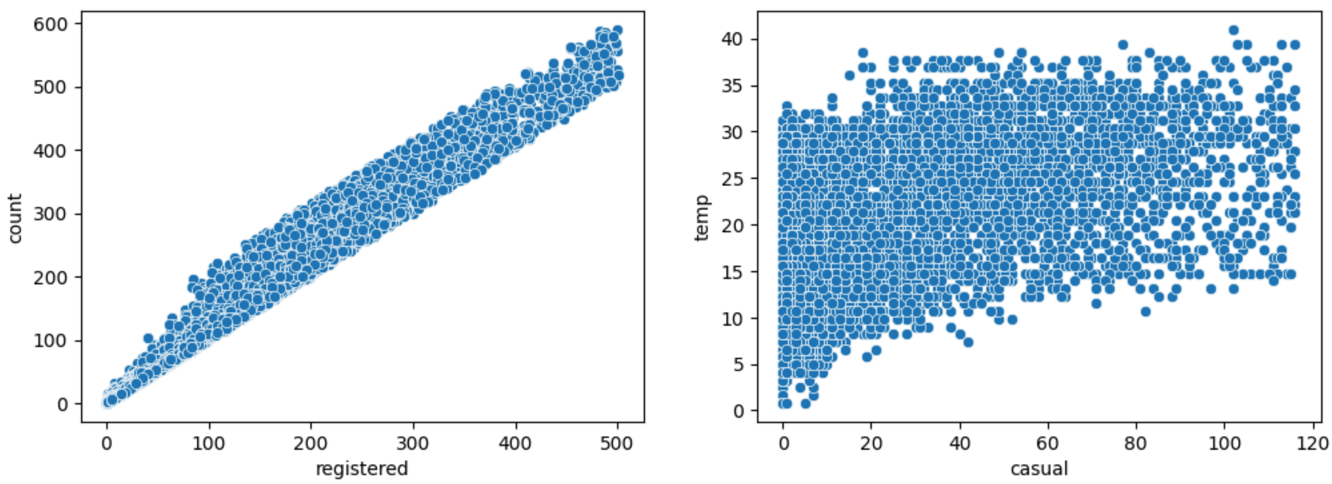
```
a = df.corr(numeric_only= True)

plt.figure(figsize=(15,6))
sns.heatmap(data = a, annot = True)

plt.show()
```



Insight

- **count X registered:** There is a high positive correlation between count total users and count of registered users. As the number of total users increase the number of registered users also grow.
- **casual X temp:** We can see a moderately positive correlation between count of casual users and temperature. As the temperature increases number of casual users grow.

```
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))

sns.scatterplot(x = df['registered'], y = df['count'], ax=axis[0] )

sns.scatterplot(x = df['casual'], y = df['temp'], ax=axis[1])

plt.show()
```



4. Hypothesis Testing

4.1 Weekdays vs Weekends

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

- **H0:** Working day has no effect on number of bike rides
- **H1:** Working day has significant effect on number of bike rides

Statistical Summary

```
df.groupby('workingday')['count'].agg(['mean','sum', 'std'])
```

workingday	mean	sum	std
weekend/holiday	120.681085	329218	106.747811
working_day	161.970103	1099777	138.588572

```
working = df[df['workingday']=='working_day']
nonworking = df[df['workingday']=='weekend/holiday']

alpha = 0.05

stats, p = ttest_ind(working['count'], nonworking['count'], alternative = "greater") # 2-Sample Independent t-test

print(f'p-value: {p}')

if p < alpha:
    print('Reject Null hypothesis: Working day has significant effect on number of bike rides')
else:
    print('Fail to reject Null hypothesis: Working day has no effect on number of bike rides')
```

```
p-value: 2.6924480901178837e-44
Reject Null hypothesis: Working day has significant effect on number of bike rides
```

Insight

- As per 2 sample independent T-test we can conclude that usage of e-bike is dependent on whether it is a working day or non working day (weekend/ holiday)
- The average number of users is higher on a working day than on a non working day.

Recommendation

- **Increase usage during non working day:**
  - Yulu should promote commuting on e-bike for daily activities like grocery shopping, cafe hopping etc and not just for commuting to work.
  - Yulu can offer discounts and offer during non-working days.
- **Brand positioning:**
  - Yulu can position itself as a smart and eco-friendly commuting options.
  - Yulu can target towards the non-working population like older and younger people.

4.2 Seasons vs Demand for Yulu

Checking if the demand of e-bikes on rent is the same for different Weather conditions

- **H0:** Demand is same for all seasons
- **H1:** Demand is different same for all seasons

```
df.groupby('season')[['count', 'registered', 'casual']].agg(['sum', 'mean'])
```

	count		registered		casual	
season	sum	mean	sum	mean	sum	mean
Fall	405323	177.151661	324740	141.931818	80583	35.219843
Spring	254093	103.164028	225283	91.466910	28810	11.697117
Summer	367547	160.360820	298696	130.321117	68851	30.039703
Winter	402032	162.437172	352272	142.332121	49760	20.105051

```
alpha = 0.05
```

Checking Assumptions for the Test

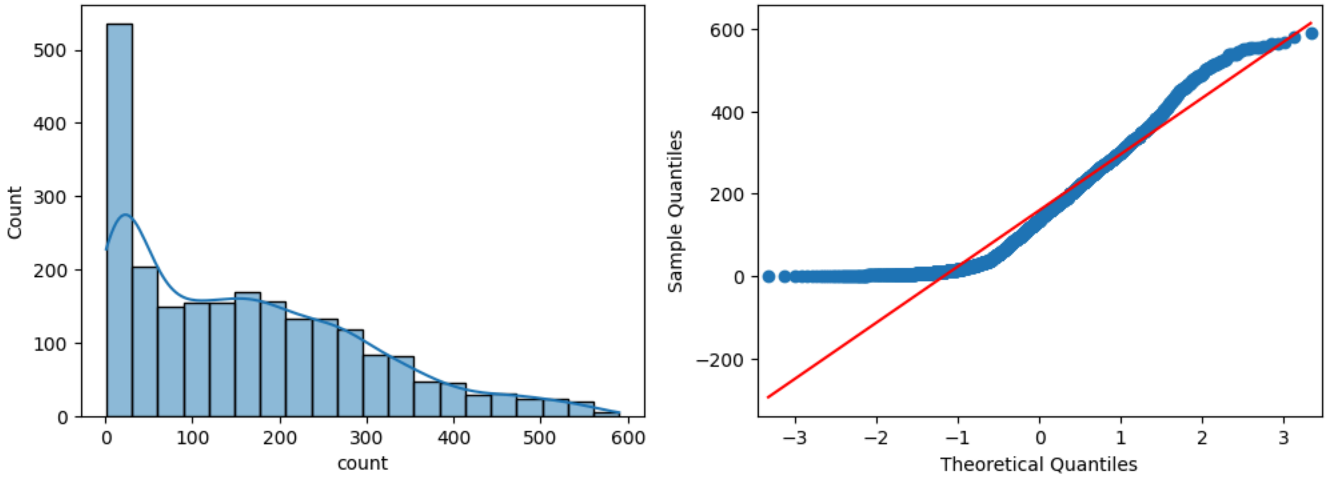
```
# Check if data is normally distributed

# h0: data is normally distributed
# ha: data is not normally distributed

summer = df[df['season']=='Summer']
winter = df[df['season']=='Winter']
fall = df[df['season']=='Fall']
spring = df[df['season']=='Spring']

shapiro(summer['count']), shapiro(winter['count']), shapiro(fall['count']), shapiro(spring['count'])
```

```
(ShapiroResult(statistic=0.9176210165023804, pvalue=1.2426929547549821e-33),
 ShapiroResult(statistic=0.9272552728652954, pvalue=4.5287389233367154e-33),
 ShapiroResult(statistic=0.9323311448097229, pvalue=5.115096899524057e-31),
 ShapiroResult(statistic=0.8594179153442383, pvalue=2.0725204287364044e-42))
```

Since p-value is less than 0.05 we can say that the data is not normally distributed

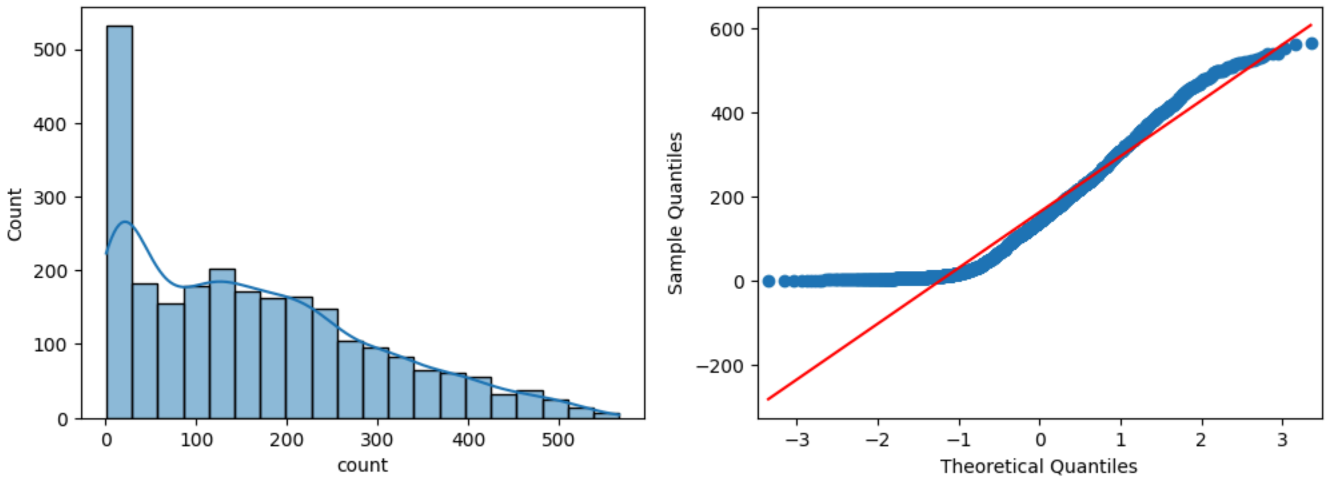
Histogram, Q-Q Plot, Skewness & Kurtosis

```
summer = df[df['season']=='Summer']['count']
winter = df[df['season']=='Winter']['count']
fall = df[df['season']=='Fall']['count']
spring = df[df['season']=='Spring']['count']


# Defininng Normality Plot Function

def normality_plot(df):
  fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))
  sns.histplot(df, ax=axis[0], bins = 20, kde = True)

  qqplot(df, line="s", ax=axis[1])

  plt.show()



# Defining Kurtosis-Skew funtion

def kurtosis_skew(df):
  k = stats.kurtosis(df)
  if k > 3:
    print(f'The distribution is tall and thin (Kurtosis: {k} > 3)')
  if k < 3:
    print(f'The distribution is flat and moderately spread out (Kurtosis: {k} < 3)')
  if k == 3:
    print(f'The distribution is normal (Kurtosis: {k} = 3)')

  s = stats.skew(df)
  if -0.5 < s < 0.5:
    print(f'The distribution is normal (Skew: {s}')
  if -1 < s <= -0.5:
    print(f'The distribution is moderately skewed (Skew: {s})')
  if 0.5 <= s < 1:
    print(f'The distribution is moderately skewed (Skew: {s})')
  if s > 1 or s < -1:
    print(f'The distribution is highly skewed (Skew: {s})')



normality_plot(summer)
kurtosis_skew(summer)
```



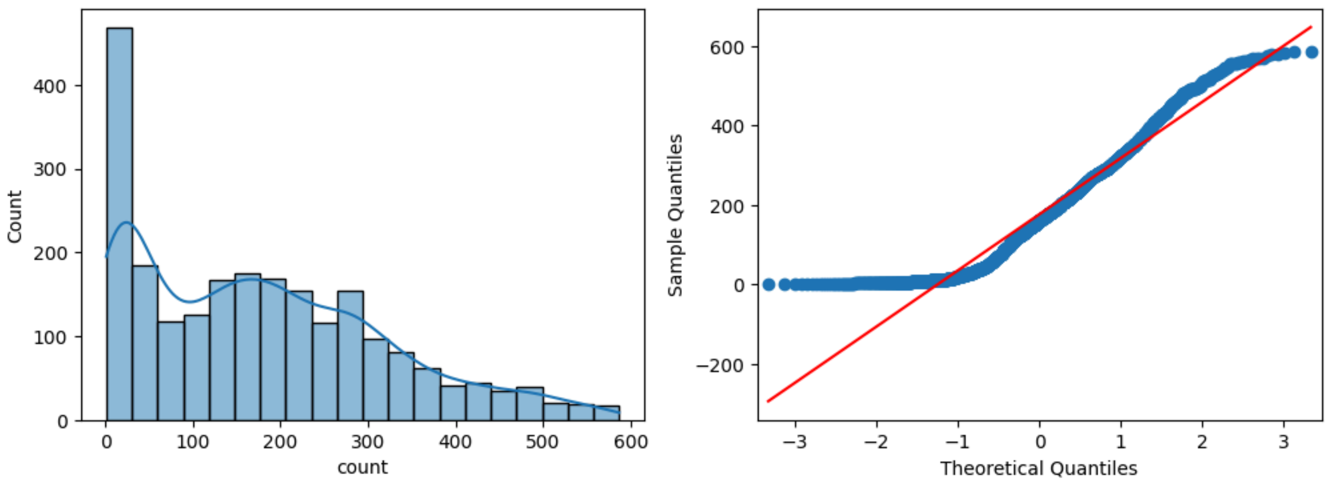

```
The distribution is flat and moderately spread out (Kurtosis: -0.10868680504449912 < 3)
The distribution is moderately skewed (Skew: 0.779796604548657)
```

```
normality_plot(winter)
kurtosis_skew(winter)
```



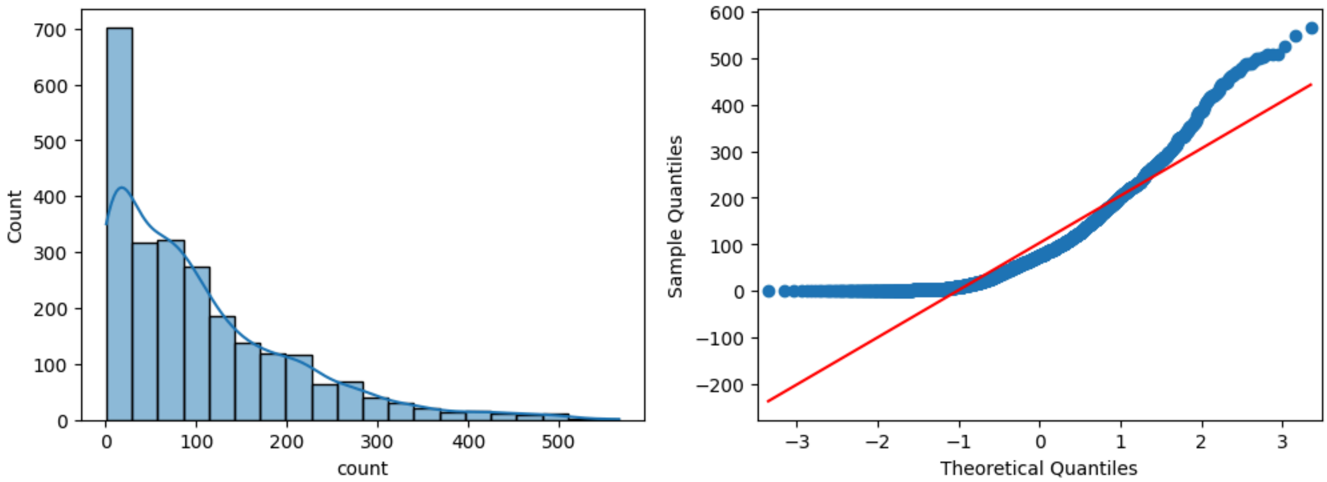

```
The distribution is flat and moderately spread out (Kurtosis: -0.3091287011796875 < 3)
The distribution is moderately skewed (Skew: 0.702651812931851)
```

```
normality_plot(fall)
kurtosis_skew(fall)
```




```
The distribution is flat and moderately spread out (Kurtosis: -0.29916266297451477 < 3)
The distribution is moderately skewed (Skew: 0.6604715346579652)
```

```
normality_plot(spring)
kurtosis_skew(spring)
```




```
The distribution is flat and moderately spread out (Kurtosis: 2.051304237496603 < 3)
The distribution is highly skewed (Skew: 1.4256800374639509)
```

Levene Test

```
# H0: Variances are equal
# Ha: Variances are not equal
levene_stat, p_value = levene(summer, winter, fall, spring)
if p_value < alpha:
    print("Variances are not equal")
else:
  print("Variances are equal")
```

```
Variances are not equal
```

One way ANOVA test

```
summer = df[df['season']=='Summer']['count']
winter = df[df['season']=='Winter']['count']
fall = df[df['season']=='Fall']['count']
spring = df[df['season']=='Spring']['count']


f_stats, p_value = f_oneway(summer, winter, fall, spring)

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

```
test statistic: 155.83821650550502
p_value: 1.328514170995064e-98
```

```
if p_value < alpha:
    print('Reject Null hypothesis: Bike usage depends on season')
else:
    print('Fail to Reject Null hypothesis: Bike usage is independent of season ')
```

```
Reject Null hypothesis: Bike usage depends on season
```

Insight

- **Shapiro Test:** The count of users for all the seasons is not normally distributed.
- **Histogram & Q-Q Plot:** Looking at the histogram and qq plot we can say that the values are not normally distributed. The distribution appears to be right skewed.
- **Skewness & Kurtosis:** After calculating the Kurtosis and Skewness values we can say that the distribution for summer, winter and fall is flat, moderately spread out and moderately skewed. While the distribution for Spring season is flat and highly skewed.
- **Levene Test:** We performed Levene Test to check the equality of variance between seasons and we can say that the variances are not equal.
- **One-way ANOVA Test:** We used One-way ANOVA test to conclude that usage of Yulu bikes is dependent of season.

Recommendation

- **Increase usage during Spring season:** Create seasonal marketing campaigns and promotions that align with the weather and outdoor activities.
- **Increase fleet size during peak season:** Yulu can increase no of bike during peak season like summer and fall to meet the demand.

4.3 Weather vs Demand for Yulu

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

- **H0:** Demand is same for all weather
- **H1:** Demand is different same for all weather

```
df.groupby('weather')[['count', 'registered', 'casual']].agg(['sum', 'mean'])
```

	count		registered		casual	
	sum	mean	sum	mean	sum	mean
weather						
Clear	972856	157.522021	812903	131.622895	159953	25.899126
Cloudy	376997	146.805685	319241	124.315031	57756	22.490654
Heavy Rain	164	164.000000	158	158.000000	6	6.000000
Light Rain	78978	102.170763	68689	88.860285	10289	13.310479

```
alpha = 0.05
```

Checking Assumptions for the Test

```
# Check if data is normally distributed

# h0: data is normally distributed
# ha: data is not normally distributed

clear = df[df['weather']=='Clear']
cloudy = df[df['weather']=='Cloudy']
lightRain = df[df['weather']=='Light Rain']
heavyRain = df[df['weather']=='Heavy Rain']
```

```
shapiro(clear['count']), shapiro(cloudy['count']), shapiro(lightRain['count'])
```

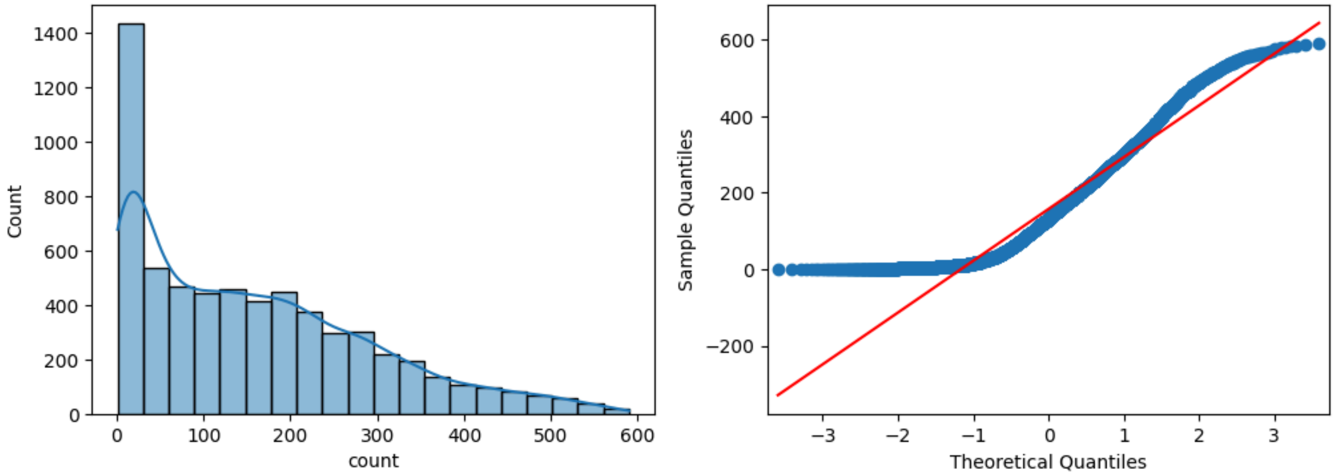
```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_morestats.py:1882: UserWarning: p-value may not be accurate for N >
  warnings.warn("p-value may not be accurate for N > 5000.")
(ShapiroResult(statistic=0.9150597453117371, pvalue=0.0),
 ShapiroResult(statistic=0.91085284948349, pvalue=2.1035535621065664e-36),
 ShapiroResult(statistic=0.8443405628204346, pvalue=7.518643302497174e-27))
```

Since p-value is less than 0.05 we can say that the data is not normally distributed

Histogram, Q-Q Plot, Skewness & Kurtosis

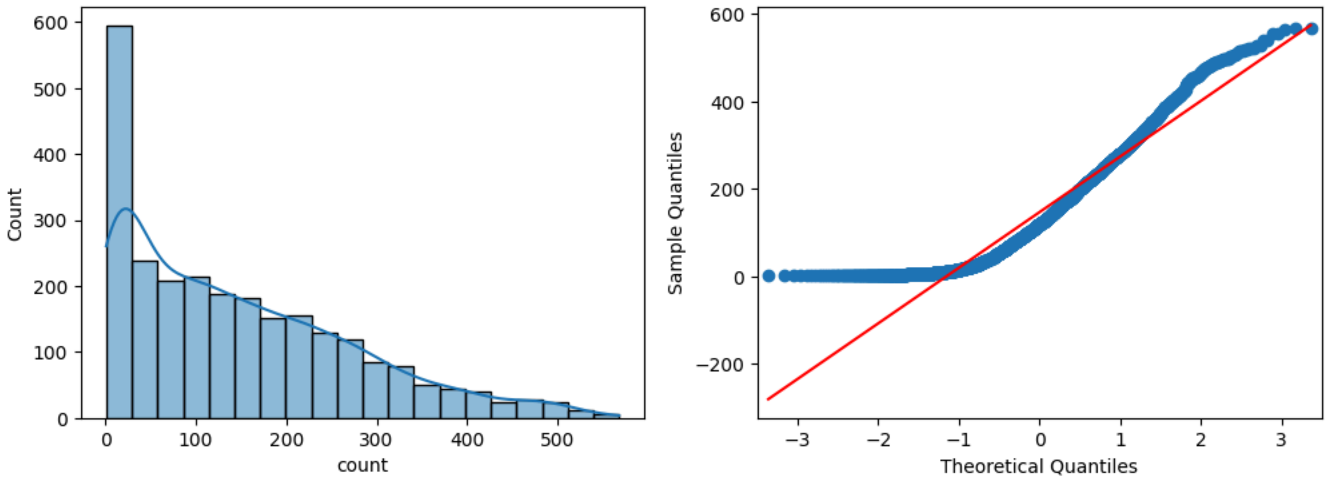
```
clear = df[df['weather']=='Clear']['count']
cloudy = df[df['weather']=='Cloudy']['count']
lightRain = df[df['weather']=='Light Rain']['count']
heavyRain = df[df['weather']=='Heavy Rain']['count']
```

```
normality_plot(clear)
kurtosis_skew(clear)
```



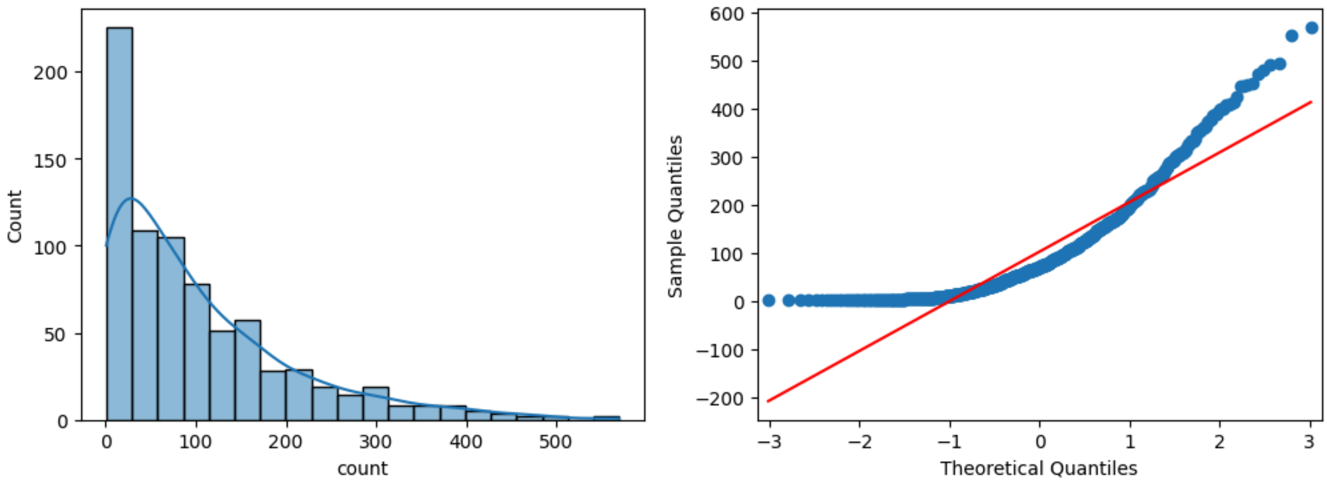

```
The distribution is flat and moderately spread out (Kurtosis: -0.02630509771840117 < 3)
The distribution is moderately skewed (Skew: 0.8256728710085347)
```

```
normality_plot(cloudy)
kurtosis_skew(cloudy)
```




```
The distribution is flat and moderately spread out (Kurtosis: 0.08997334454087769 < 3)
The distribution is moderately skewed (Skew: 0.8782720514882909)
```

```
normality_plot(lightRain)
kurtosis_skew(lightRain)
```




```
The distribution is flat and moderately spread out (Kurtosis: 2.262161515931262 < 3)
The distribution is highly skewed (Skew: 1.5118437612190248)
```

Levene Test

```
# H0: Variances are equal
# Ha: Variances are not equal
levene_stat, p_value = levene(clear, cloudy, lightRain)
if p_value < alpha:
    print("Variances are not equal")
else:
  print("Variances are equal")
```

```
Variances are not equal
```

One way ANOVA test

```
f_stats, p_value = f_oneway(clear, cloudy, lightRain)

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

```
test statistic: 62.70255115766609
p_value: 8.841710069607572e-28
```

```
if p_value < alpha:
    print('Reject Null hypothesis: Bike usage depends on weather condition')
else:
    print('Fail to Reject Null hypothesis: Bike usage is independent of weather condition')
```

```
Reject Null hypothesis: Bike usage depends on weather condition
```

Insight

- **Shapiro Test:** The count of users for all kinds weather is not normally distributed.
- **Histogram & Q-Q Plot:** Looking at the histogram and qq plot we can say that the values are not normally distributed. The distribution appears to be right skewed.
- **Skewness & Kurtosis:** After calculating the Kurtosis and Skewness value we can say that the distribution for clear and cloudy condition is flat, moderately spread out and moderately skewed. While the distribution for light rain weather the data is flat and highly skewed.
- **Levene Test:** We performed Levene Test to check the equality of variance between different weather conditions and we can say that the variances are not equal.
- **One-way ANOVA Test:** We used One-way ANOVA test to conclude that usage of Yulu bikes is dependent on weather.

Recommendation

- **Weather Alerts and Notifications:** Implement a system that sends weather alerts and notifications to riders. When the weather is clear and suitable for biking, send notifications to riders.

4.4 Weather vs Seasons

Checking if the Weather conditions are significantly different during different Seasons

- **H0:** Weather conditions are same for all season
- **H1:** Weather conditions are different for all season

Contingency Table against ‘Weather’ & ‘Season’

```
df2 = df.drop(df[df['weather'] == 'Heavy Rain'].index) # Droping heavy rain weather condition due to limited data
```

```
contingency_table = pd.crosstab(index = df2['weather'], columns = df2['season'], margins = True, normalize = True).round(3)
```

```
contingency_table
```

season	Fall	Spring	Summer	Winter	All
weather					
Clear	0.168	0.168	0.155	0.159	0.649
Cloudy	0.054	0.072	0.065	0.079	0.270
Light Rain	0.018	0.019	0.022	0.022	0.081
All	0.240	0.259	0.241	0.260	1.000

Next steps: Generate code with `contingency_table` View recommended plots

Insight

1. Probability of weather being clear is **64.9%**
   - The **conditional probability** of weather being clear given that the season is:
     - Fall is 16.8%
     - Spring is 16.8%
     - Summer is 15.5%
     - Winter is 15.9%
2. Probability of weather being cloudy is **27%**
   - The **conditional probability** of weather being cloudy given that the season is:
     - Fall is 5.4%
     - Spring is 7.2%
     - Summer is 6.5%
     - Winter is 7.9%
3. Probability of weather being light rainy is **8.1%**
   - The **conditional probability** of weather being light rainy given that the season is:
     - Fall is 1.8%
     - Spring is 1.9%
     - Summer is 2.2%
     - Winter is 2.2%

```
contingency_table2 = pd.crosstab(df['season'], df['weather'])

stats, p, dof, e = chi2_contingency(contingency_table2)

print(f'p-value: {p}')

if p < 0.05:
    print('reject null hypothesis: Weather conditions are different for all season')
else:
    print('fail to reject null hypothesis: Weather conditions are same for all season')
```

```
p-value: 1.0976664201931212e-07
reject null hypothesis: Weather conditions are different for all season
```