YULU - HYPOTHESIS TESTING

```
Business Problem:

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

    - How well those variables describe the electric cycle demands

Column Profiling:

datetime: datetime
season: season (1: spring, 2: summer, 3: fall, 4: winter)
holiday: whether day is a holiday or not (extracted from
http://dchr.dc.gov/page/holiday-schedule)
workingday: if day is neither weekend nor holiday is 1, otherwise is
0.
weather:
    1: Clear, Few clouds, partly cloudy, partly cloudy
    2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light
Rain + Scattered clouds
    4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp: temperature in Celsius
atemp: feeling temperature in Celsius
humidity: humidity
windspeed: wind speed
casual: count of casual users
registered: count of registered users
count: count of total rental bikes including both casual and
registered
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import f_oneway, ttest_ind, shapiro, kruskal,
chi2_contingency, levene
from statsmodels.graphics.gofplots import qqplot
```

Loading yulu data

```
yulu_data = pd.read_csv('bike_sharing.txt')

yulu_data
```

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

        atemp  humidity  windspeed  casual  registered  count  
0      14.395        81     0.0000       3          13     16  
1      13.635        80     0.0000       8          32     40  
2      13.635        80     0.0000       5          27     32  
3      14.395        75     0.0000       3          10     13  
4      14.395        75     0.0000       0           1      1  
...       ...       ...        ...     ...         ...    ...  
10881  19.695        50    26.0027       7         329    336  
10882  17.425        57    15.0013      10         231    241  
10883  15.910        61    15.0013       4         164    168  
10884  17.425        61     6.0032      12         117    129  
10885  16.665        66     8.9981       4          84     88  

[10886 rows x 12 columns]
```

EDA

```
yulu_data.shape
```

```
(10886, 12)
```

Insights: Data set has 10886 rows and 12 columns.

```
yulu_data.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: No null values found.

```
yulu_data.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
```

```
print(list(yulu_data.columns))
```

```
['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']
```

```
print(yulu_data.dtypes)
```

```
datetime       object
season          int64
holiday         int64
```

```
workingday      int64
weather         int64
temp          float64
atemp         float64
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
dtype: object
```

Converting 'datetime' column to type datetime

```
yulu_data['datetime']= pd.to_datetime(yulu_data['datetime'])

yulu_data['datetime'].dtype

dtype('<M8[ns]')

print(yulu_data.dtypes)

datetime      datetime64[ns]
season                 int64
holiday                int64
workingday             int64
weather                int64
temp                 float64
atemp                float64
humidity               int64
windspeed            float64
casual                 int64
registered             int64
count                  int64
dtype: object
```

Checking value counts of each column

```
 for col in list(yulu_data.columns):
        print(col,'\n',yulu_data[col].value_counts(),'\n')

datetime
 2011-06-09 04:00:00    1
2012-06-06 21:00:00    1
2011-02-14 21:00:00    1
2012-08-02 05:00:00    1
2012-10-13 20:00:00    1
                      ..
```

```
2011-05-05 10:00:00    1
2011-06-15 11:00:00    1
2012-10-16 15:00:00    1
2011-05-14 09:00:00    1
2012-05-04 19:00:00    1
Name: datetime, Length: 10886, dtype: int64

season
 4    2734
3    2733
2    2733
1    2686
Name: season, dtype: int64

holiday
 0    10575
1      311
Name: holiday, dtype: int64

workingday
 1    7412
0    3474
Name: workingday, dtype: int64

weather
 1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64

temp
 14.76    467
26.24    453
28.70    427
13.94    413
18.86    406
22.14    403
25.42    403
16.40    400
22.96    395
27.06    394
24.60    390
12.30    385
21.32    362
13.12    356
17.22    356
29.52    353
10.66    332
18.04    328
```

```
20.50    327
30.34    299
9.84     294
15.58    255
9.02     248
31.16    242
8.20     229
27.88    224
23.78    203
32.80    202
11.48    181
19.68    170
6.56     146
33.62    130
5.74     107
7.38     106
31.98     98
34.44     80
35.26     76
4.92      60
36.90     46
4.10      44
37.72     34
36.08     23
3.28      11
38.54      7
0.82       7
39.36      6
2.46       5
1.64       2
41.00      1
Name: temp, dtype: int64

atemp
 31.060    671
25.760    423
22.725    406
20.455    400
26.515    395
16.665    381
25.000    365
33.335    364
21.210    356
30.305    350
15.150    338
21.970    328
24.240    327
17.425    314
31.820    299
```

```
34.850    283
27.275    282
32.575    272
11.365    271
14.395    269
29.545    257
19.695    255
15.910    254
12.880    247
13.635    237
34.090    224
12.120    195
28.790    175
23.485    170
10.605    166
35.605    159
9.850     127
18.180    123
36.365    123
37.120    118
9.090     107
37.880     97
28.030     80
7.575      75
38.635     74
6.060      73
39.395     67
6.820      63
8.335      63
18.940     45
40.150     45
40.910     39
5.305      25
42.425     24
41.665     23
3.790      16
4.545      11
43.940      7
43.180      7
2.275       7
3.030       7
44.695      3
0.760       2
1.515       1
45.455      1
Name: atemp, dtype: int64

humidity
 88    368
```

```
94    324
83    316
87    289
70    259
     ...
13      1
10      1
12      1
96      1
91      1
Name: humidity, Length: 89, dtype: int64

windspeed
 0.0000     1313
8.9981     1120
11.0014    1057
12.9980    1042
7.0015     1034
15.0013     961
6.0032      872
16.9979     824
19.0012     676
19.9995     492
22.0028     372
23.9994     274
26.0027     235
27.9993     187
30.0026     111
31.0009      89
32.9975      80
35.0008      58
39.0007      27
36.9974      22
43.0006      12
40.9973      11
43.9989       8
46.0022       3
56.9969       2
47.9988       2
50.0021       1
51.9987       1
Name: windspeed, dtype: int64

casual
 0      986
1      667
2      487
3      438
4      354
```

```
      ...
291      1
327      1
331      1
355      1
299      1
Name: casual, Length: 309, dtype: int64

registered
 3      195
4      190
5      177
6      155
2      150
      ...
577      1
561      1
537      1
521      1
839      1
Name: registered, Length: 731, dtype: int64

count
 5      169
4      149
3      144
6      135
2      132
      ...
667      1
603      1
587      1
970      1
843      1
Name: count, Length: 822, dtype: int64
```

Creating new column time and month from datetime

```
yulu_data['time']= yulu_data['datetime'].dt.hour.astype('str')
yulu_data['month']= yulu_data['datetime'].dt.month.astype('str')
yulu_data['year']= yulu_data['datetime'].dt.year.astype('str')
```

we can drop datetime column now, as we have slipt the datetime column to other columns

```
yulu_data=yulu_data.drop('datetime', axis=1)

yulu_data

       season  holiday  workingday  weather   temp   atemp  
humidity  \
0           1        0           0        1   9.84  14.395         81

1           1        0           0        1   9.02  13.635         80

2           1        0           0        1   9.02  13.635         80

3           1        0           0        1   9.84  14.395         75

4           1        0           0        1   9.84  14.395         75

...       ...      ...         ...      ...    ...     ...        ...

10881       4        0           1        1  15.58  19.695         50

10882       4        0           1        1  14.76  17.425         57

10883       4        0           1        1  13.94  15.910         61

10884       4        0           1        1  13.94  17.425         61

10885       4        0           1        1  13.12  16.665         66


       windspeed  casual  registered  count time month  year
0         0.0000       3          13     16    0     1  2011
1         0.0000       8          32     40    1     1  2011
2         0.0000       5          27     32    2     1  2011
3         0.0000       3          10     13    3     1  2011
4         0.0000       0           1      1    4     1  2011
...          ...     ...         ...    ...  ...   ...   ...
10881    26.0027       7         329    336   19    12  2012
10882    15.0013      10         231    241   20    12  2012
10883    15.0013       4         164    168   21    12  2012
10884     6.0032      12         117    129   22    12  2012
10885     8.9981       4          84     88   23    12  2012

[10886 rows x 14 columns]
```

Converting seasons, holiday, workingday and weather to categorical columns

```
yulu_data[['season','holiday','workingday','weather']]=
yulu_data[['season','holiday','workingday','weather']].astype('str')

yulu_data.dtypes
```

```
season          object
holiday         object
workingday      object
weather         object
temp           float64
atemp          float64
humidity         int64
windspeed      float64
casual           int64
registered       int64
count            int64
time            object
month           object
year            object
dtype: object
```

From the data columns, let us consider 'count' as north star metric and the target column, as at the end of the day metric which is measured for yulu business is total no of bikes rented per day.

```
target = yulu_data['count']

cat_cols = list(yulu_data.columns[yulu_data.dtypes=='object'])

cat_cols
```

```
['season', 'holiday', 'workingday', 'weather', 'time', 'month',
'year']
```

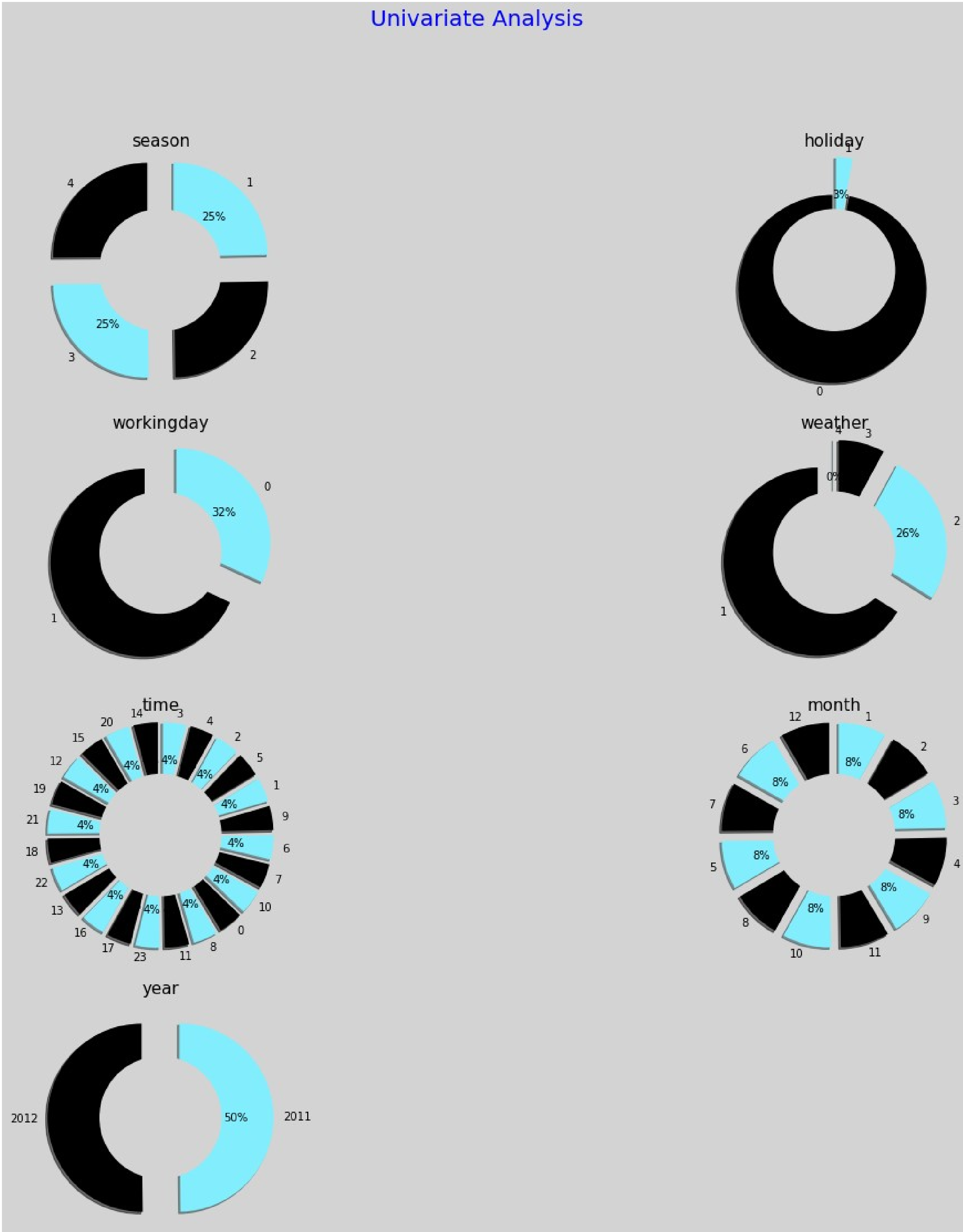
```
num_cols = list(yulu_data.columns[(yulu_data.dtypes!='object')&
(yulu_data.columns!='count')])
num_cols
```

```
['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered']
```

VISUAL ANALYSIS

```
yulu_color_palette = sns.color_palette(['black','#82EEFD'])
fig = plt.figure(figsize=(20,18))
fig.set_facecolor("lightgrey")
for n,col in enumerate(cat_cols):
```

```
    explode = yulu_data[col].nunique()*[0.2]
    plt.subplot(int(len(cat_cols)/2 +1), 2, n+1)
    plt.pie(list(yulu_data[col].value_counts().values),
            labels = list(yulu_data[col].value_counts().index),
autopct='%.0f%%', shadow= True,
            startangle = 90, colors = yulu_color_palette, radius= 1,
explode = explode )
    hole = plt.Circle((0, 0), 0.65, facecolor='lightgrey')
    plt.gcf().gca().add_artist(hole)
    plt.title(col, fontsize = 15)
    plt.xticks(rotation = 45)
fig.suptitle("Univariate Analysis", fontsize= 20, color = 'blue')
plt.show()
```

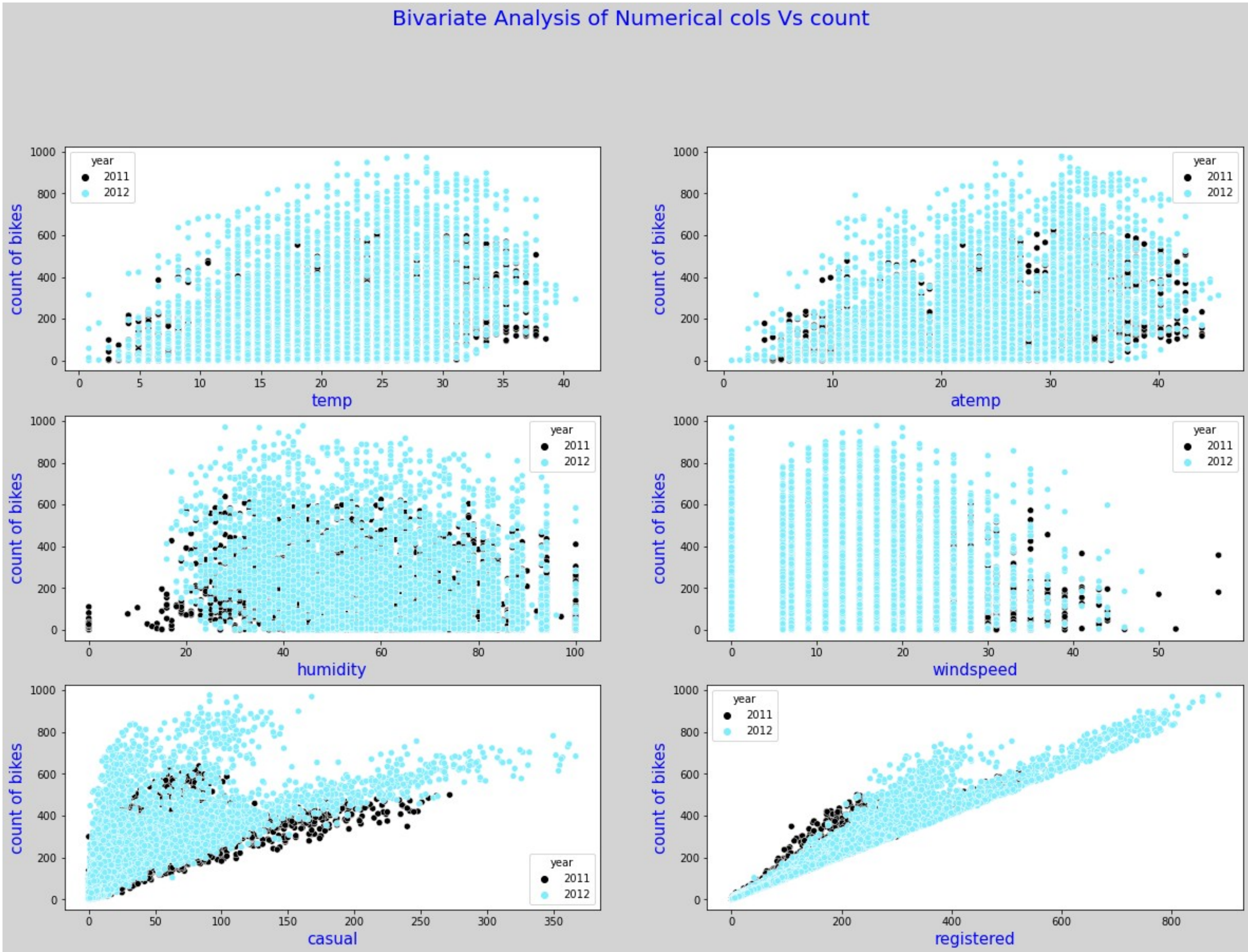

Univariate Analysis
season
4
1
25%
25%
3
2
holiday
1
3%
0
workingday
0
32%
1
weather
4
3
0%
2
26%
1
time
14
3
4
20
2
15
5
12
1
19
9
21
6
18
7
22
10
13
0
16
8
17
23
11
4%
month
12
1
6
2
7
3
5
4
8
9
10
11
8%
year
2012
50%
2011

Insights:

```
From the above plots we can observe that:
    - There are 4 categories in seasons which are equally distributed.
    - The holiday column has 2 categories which are 0 with 97%
datapoints and 1 with 3% datapoints.
    - The workingday has 2 categories which are 0 with 32% datapoints
and 1 with 68% datapoints.
    - The weather column has four categories, which are 1 with 65%
datapoints, 2 with 26% datapoints, 3 with 8% and 4 with 0% datapoints.
    - The time and month and year columns has equal distributions.
```

Bi-Variate Analysis

```
fig = plt.figure(figsize=(20,18))
fig.set_facecolor("lightgrey")
for i,col in enumerate(num_cols):
    plt.subplot(int(len(num_cols)/2 +1), 2, i+1)
    sns.scatterplot(x=yulu_data[col], y= target,hue =
yulu_data['year'], palette = yulu_color_palette)
    plt.xlabel(col,fontsize =15, color = 'blue')
    plt.ylabel("count of bikes", fontsize = 15, color = 'blue')
fig.suptitle("Bivariate Analysis of Numerical cols Vs count ",
fontsize= 20, color = 'blue')
plt.show()
```

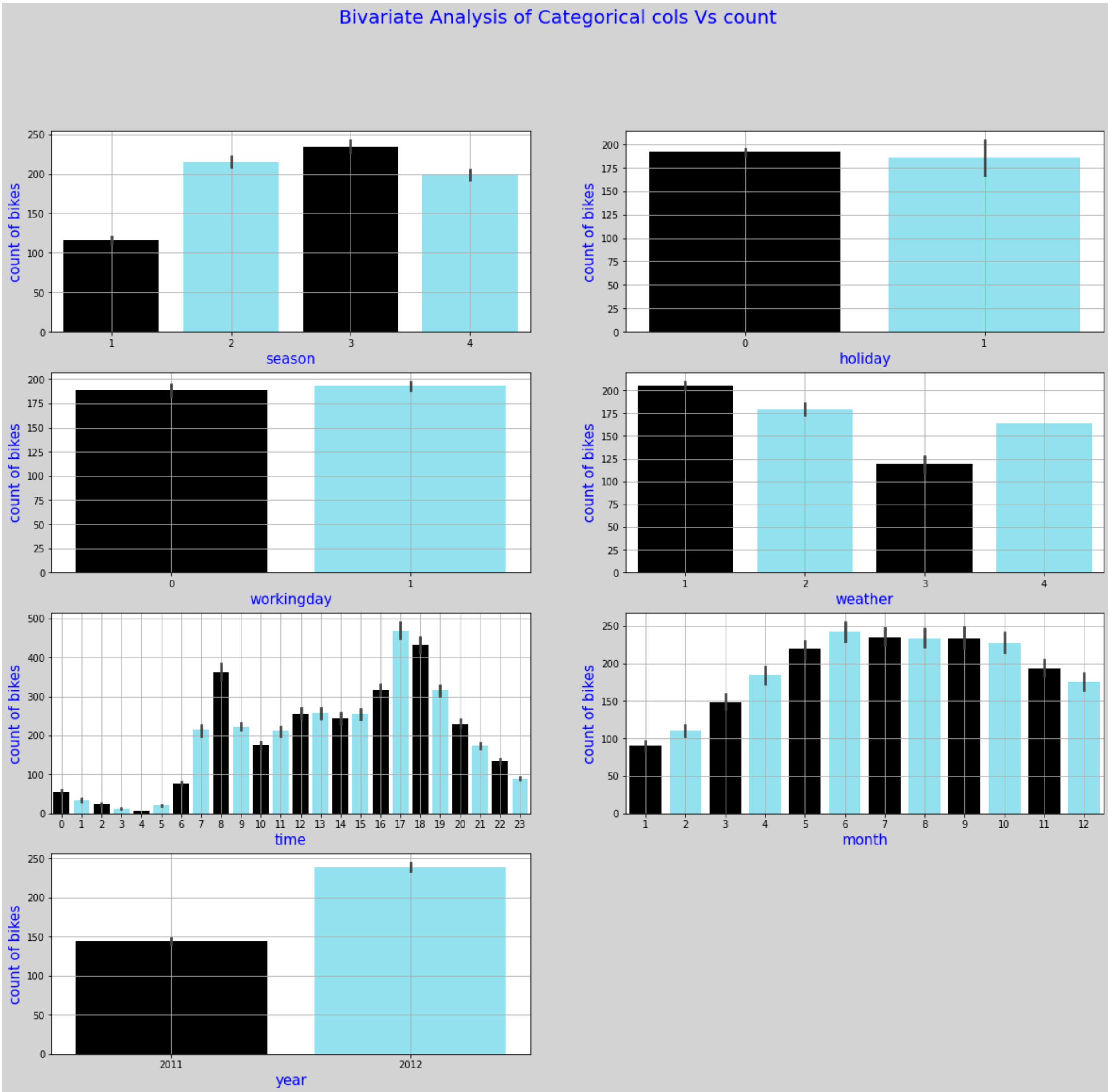


Insights:

```
From the above plot we can observe that :
    - Count of bikes is high when the temperature is between 20-30.
    - As count of casual and registered bikes increases total count
increases.
    - Wind speed and humidity has not much effect of count of bikes.
    - Overall count of bikes rented in 2012 is greater than 2011.
```

```
yulu_color_palette = sns.color_palette(['black','#82EEFD'])
fig = plt.figure(figsize=(20,18))
fig.set_facecolor("lightgrey")
for i,col in enumerate(cat_cols):
    plt.subplot(int(len(cat_cols)/2 +1), 2, i+1)
    sns.barplot(x=yulu_data[col], y= target, palette =
yulu_color_palette)
    plt.grid(True)
    plt.xlabel(col,fontsize =15, color = 'blue')
    plt.ylabel("count of bikes", fontsize = 15, color = 'blue')
fig.suptitle("Bivariate Analysis of Categorical cols Vs count ",
```

```
fontsize= 20, color = 'blue')
plt.show()
```



Insights:

```
From the above plots we can observe that:
    - count of bikes are more in season 3 and less in season 1.
    - count of bikes is more on non- holiday than holiday.
    - count of bikes nearly equal on both working and non working
days.
    - Number of bikes rented is high when weather is clear and low
when there is little rains.
```

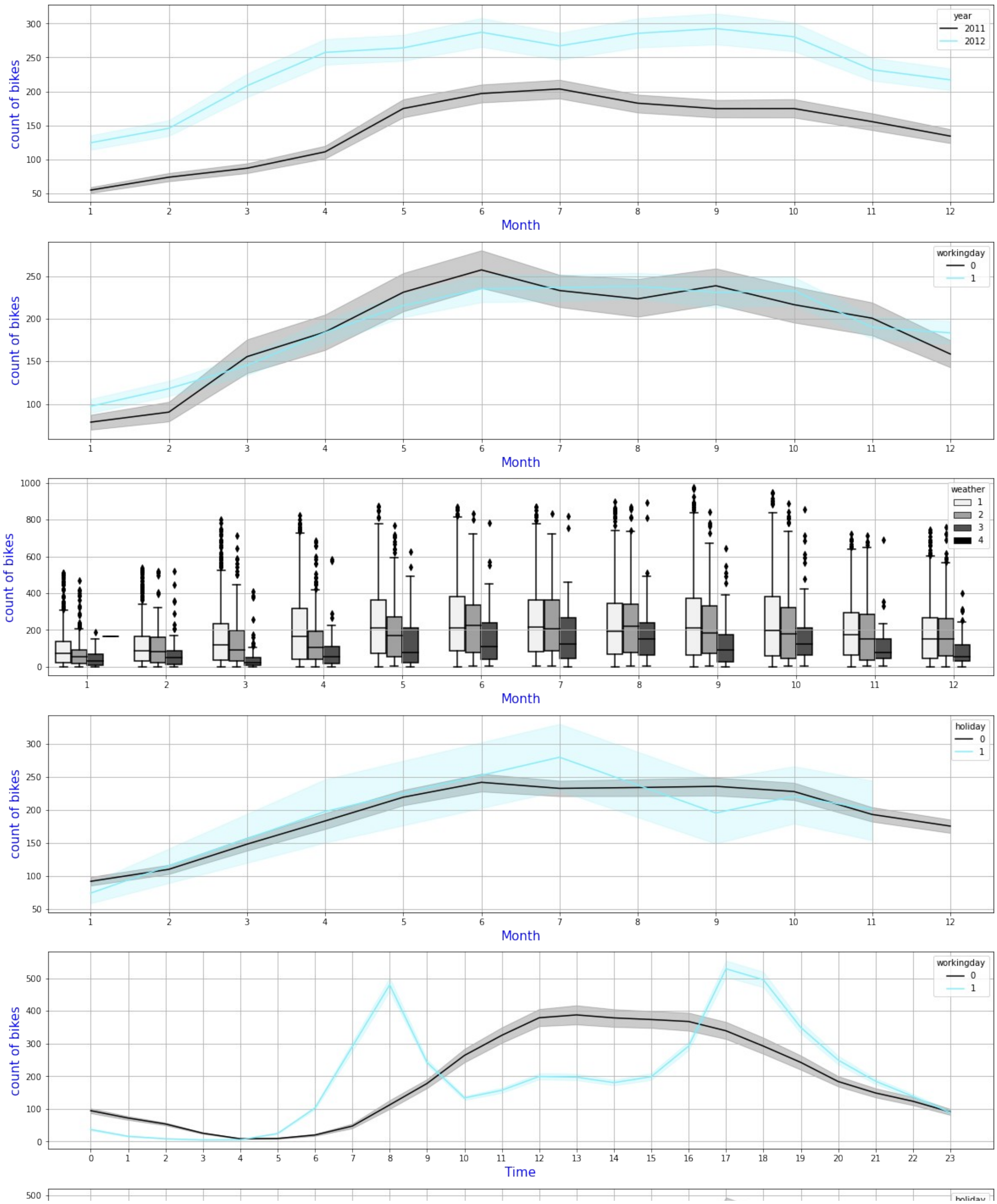
```
    - Number of bikes renteis high from the afternoon and gradually
decreased to night and low after 12am to 5am.
    - Least number of bikes are rented in the month of January and
maximum number in June.
    - Number of bikes rented in 2012 is more than 2011.
```

Multi-variate Analysis

```
plt.figure(figsize=(20,30))

plt.subplot(611)
sns.lineplot(x=yulu_data['month'],y=target, hue = yulu_data['year'],
palette = yulu_color_palette)
plt.xlabel('Month',fontsize =15, color = 'blue')
plt.ylabel("count of bikes", fontsize = 15, color = 'blue')
plt.grid(True)

plt.subplot(612)
sns.lineplot(x=yulu_data['month'],y=target, hue =
yulu_data['workingday'], palette = yulu_color_palette)
plt.xlabel('Month',fontsize =15, color = 'blue')
plt.ylabel("count of bikes", fontsize = 15, color = 'blue')
plt.grid(True)

plt.subplot(613)
sns.boxplot(x=yulu_data['month'],y=target, hue = yulu_data['weather'],
color='black')
plt.xlabel('Month',fontsize =15, color = 'blue')
plt.ylabel("count of bikes", fontsize = 15, color = 'blue')
plt.grid(True)

plt.subplot(614)
sns.lineplot(x=yulu_data['month'],y=target, hue =
yulu_data['holiday'], palette = yulu_color_palette)
plt.xlabel('Month',fontsize =15, color = 'blue')
plt.ylabel("count of bikes", fontsize = 15, color = 'blue')
plt.grid(True)

plt.subplot(615)
sns.lineplot(x=yulu_data['time'],y=target, hue =
yulu_data['workingday'], palette = yulu_color_palette)
plt.xlabel('Time',fontsize =15, color = 'blue')
plt.ylabel("count of bikes", fontsize = 15, color = 'blue')
plt.grid(True)

plt.subplot(616)
sns.lineplot(x=yulu_data['time'],y=target, hue = yulu_data['holiday'],
palette = yulu_color_palette)
```

```
plt.xlabel('Time',fontsize =15, color = 'blue')
plt.ylabel("count of bikes", fontsize = 15, color = 'blue')
plt.grid(True)

plt.suptitle("Multivariate Analysis", fontsize= 20, color = 'blue')
plt.show()
```

Multivariate Analysis

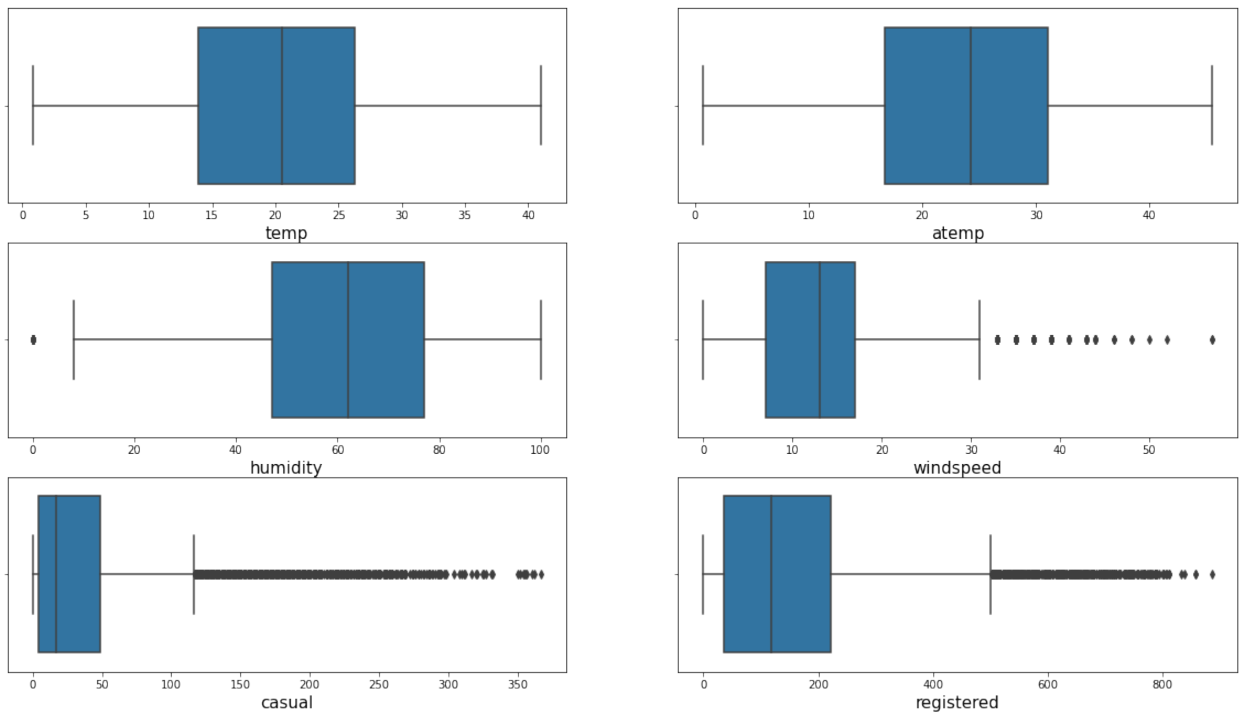


Insights:

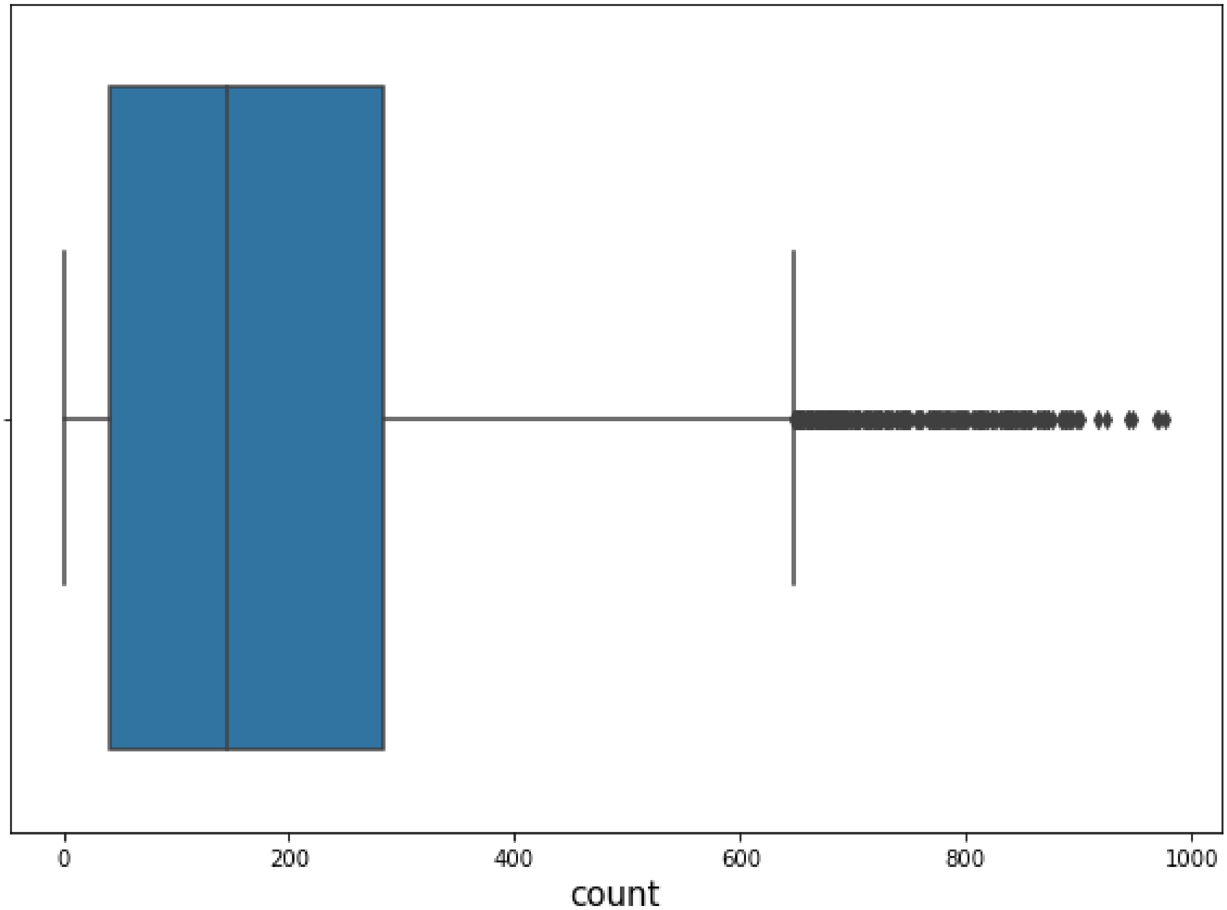
```
   From the above plots of multivariate analysis we observe that:
    - count of bikes rented in 2012 are higher than 2011 and overall
consitency is mainted between months may to october.
    - In any month the count of bikes rented on either working day or
non-working day are nearly equal.
    - Only three types of weathers are observed in all the months
except in jan, all the four types of weather were observed. Weather is
consistent from may to october.
    - Count of bikes on holidays are higher only in the month of july
and lower in the month of september. In the other months the count of
bikes is equal on holidys and non-holidays.
    - Count of bikes rented on working days raised from 5am to 8am and
then again raised in the eveing 5pm.
    - count of bikes rented on holidays and non-holidays follow nearly
same pattern.
```

Outliers

```
#outliers in all numerical columns
plt.figure(figsize=(20,15))
for i,col in enumerate(num_cols):
    plt.subplot(int(len(num_cols)/2)+1, 2, i+1)
    sns.boxplot(x= yulu_data[col])
    plt.xlabel(col, fontsize=15)
plt.show()
```




```python
#checking for outliers in count column using box plot method
plt.figure(figsize=(10,7))
sns.boxplot(x= target)
plt.xlabel('count', fontsize=15)
plt.show()
```



Insights:

```
Outliers are observed in count, casual and registered coulumns.

#checking for outliers in count col using IQR method

Q1 = target.quantile(0.25)
Q3 = target.quantile(0.75)

IQR = Q3-Q1

lower_bound=Q1-1.5*IQR
upper_bound=Q3+1.5*IQR

yulu_data[(yulu_data['count']< lower_bound) | (yulu_data['count'] >
upper_bound)]

      season holiday workingday weather   temp   atemp  humidity
windspeed  \
6611       1       0          1       2  24.60  31.060        43
12.9980
6634       1       0          1       1  28.70  31.820        37
```

```
7.0015
6635        1       0          1       1  28.70  31.820         34
19.9995
6649        1       0          1       1  18.04  21.970         82
0.0000
6658        1       0          1       1  28.70  31.820         28
6.0032
...       ...     ...        ...     ...    ...     ...        ...
...
10678       4       0          1       2  13.94  15.150         61
19.9995
10702       4       0          1       2  10.66  12.880         65
11.0014
10726       4       0          1       1   9.84  11.365         60
12.9980
10846       4       0          1       1  15.58  19.695         94
0.0000
10870       4       0          1       1   9.84  12.880         87
7.0015

       casual  registered  count  time  month  year
6611       89         623    712    18      3  2012
6634       62         614    676    17      3  2012
6635       96         638    734    18      3  2012
6649       34         628    662     8      3  2012
6658      140         642    782    17      3  2012
...       ...         ...    ...   ...    ...   ...
10678      16         708    724     8     12  2012
10702      18         670    688     8     12  2012
10726      24         655    679     8     12  2012
10846      10         652    662     8     12  2012
10870      13         665    678     8     12  2012

[300 rows x 14 columns]
```

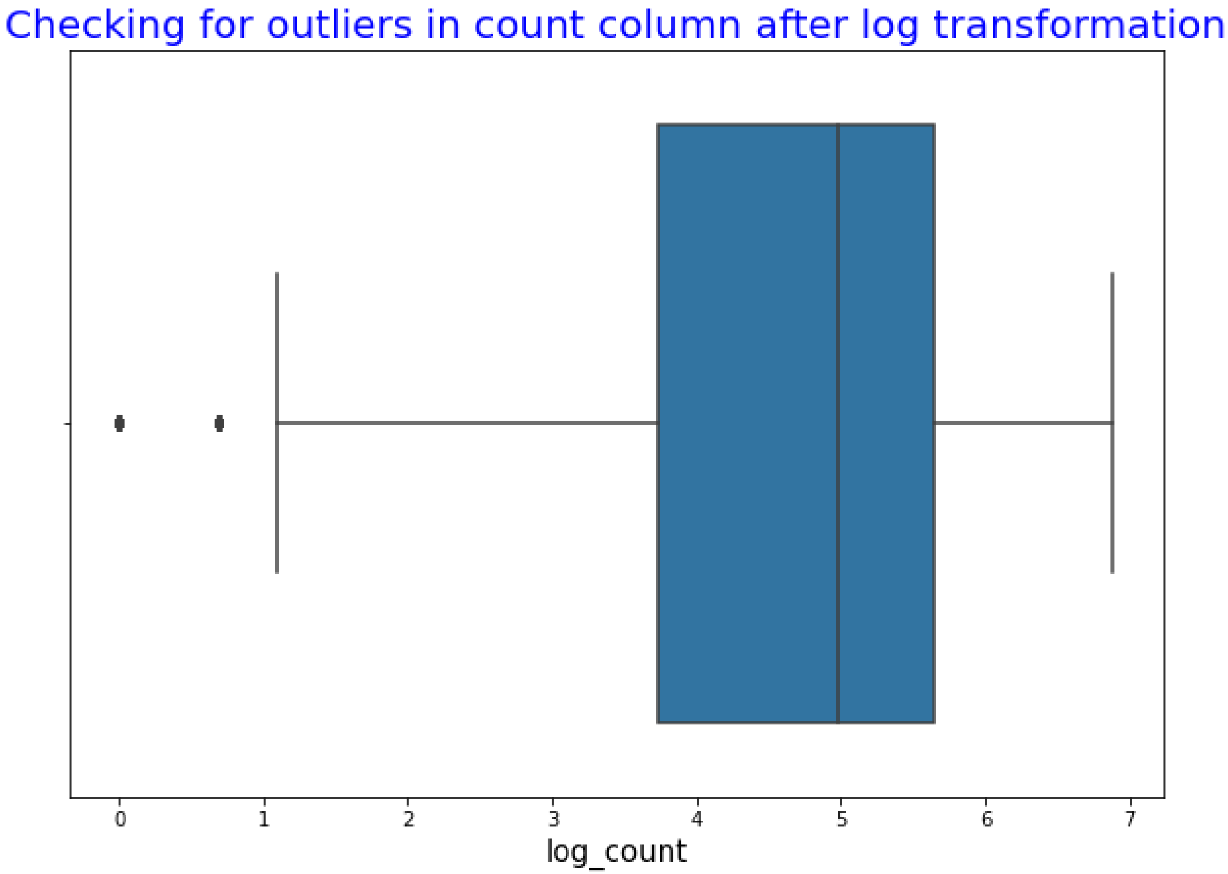
Insights: As count is the north star metric colum for analysis, we calculate outliers for count using IQR method and checked the outliers less than lowerbound and greater than upperbound and noticed that there are 300 ouliers.

Using Log trasformation for reducing the impact of outliers in count column

```python
yulu_data['log_count']= [np.log(x) for x in yulu_data['count']]

yulu_data
```

```
       season holiday workingday weather   temp   atemp  humidity 
windspeed  \
0           1       0          0       1   9.84  14.395        81 
0.0000
1           1       0          0       1   9.02  13.635        80 
0.0000
2           1       0          0       1   9.02  13.635        80 
0.0000
3           1       0          0       1   9.84  14.395        75 
0.0000
4           1       0          0       1   9.84  14.395        75 
0.0000
...       ...     ...        ...     ...    ...     ...       ... 
...
10881       4       0          1       1  15.58  19.695        50 
26.0027
10882       4       0          1       1  14.76  17.425        57 
15.0013
10883       4       0          1       1  13.94  15.910        61 
15.0013
10884       4       0          1       1  13.94  17.425        61 
6.0032
10885       4       0          1       1  13.12  16.665        66 
8.9981

       casual  registered  count time month  year  log_count
0           3          13     16    0     1  2011   2.772589
1           8          32     40    1     1  2011   3.688879
2           5          27     32    2     1  2011   3.465736
3           3          10     13    3     1  2011   2.564949
4           0           1      1    4     1  2011   0.000000
...       ...         ...    ...  ...   ...   ...        ...
10881       7         329    336   19    12  2012   5.817111
10882      10         231    241   20    12  2012   5.484797
10883       4         164    168   21    12  2012   5.123964
10884      12         117    129   22    12  2012   4.859812
10885       4          84     88   23    12  2012   4.477337

[10886 rows x 15 columns]
```

```
plt.figure(figsize=(10,7))
sns.boxplot(x= yulu_data['log_count'])
plt.xlabel("log_count", fontsize=15)
plt.title("Checking for outliers in count column after log 
transformation", fontsize=20, color='blue')
plt.show()
```



Insight: Outliers are low in log_count compared to count. So we proceed with analysis based on log_count.

```
yulu_data[num_cols].describe()

              temp         atemp      humidity     windspeed
casual  \
count  10886.00000  10886.000000  10886.000000  10886.000000
10886.000000
mean      20.23086     23.655084     61.886460     12.799395
36.021955
std        7.79159      8.474601     19.245033      8.164537
49.960477
min        0.82000      0.760000      0.000000      0.000000
0.000000
25%       13.94000     16.665000     47.000000      7.001500
4.000000
50%       20.50000     24.240000     62.000000     12.998000
17.000000
75%       26.24000     31.060000     77.000000     16.997900
49.000000
max       41.00000     45.455000    100.000000     56.996900
367.000000
```

```
           registered
count    10886.000000
mean       155.552177
std        151.039033
min          0.000000
25%         36.000000
50%        118.000000
75%        222.000000
max        886.000000
```

The above data gives the non-graphical analysis of numerical columns which gives the some measures like mean, standard deviation, min, max of each numeric column.

```
yulu_data[cat_cols].describe(include='object')
```

```
       season holiday workingday weather   time  month   year
count   10886   10886      10886   10886  10886  10886  10886
unique      4       2          2       4     24     12      2
top         4       0          1       1     14     12   2012
freq     2734   10575       7412    7192    456    912   5464
```

The above data gives the non-graphical analysis for the category columns such as no of categories in each category column, top category in each column, count, frequency of top category.

HYPOTHESIS TESTING

1. Checking if numerical columns have correlation with count

For checking correlation between numeric data we choose correlation Spearman correlation and calculate correlation coefficient by calculatiing rank of each data and then find the corrcoef.

```
- Null hypothesis is any two numeric columns are independent of each
other.
- Alternate hypothesis will be columns are dependent on each other.
- The correration coefficient gives the strength of relationship.
- The range of spearman corrcoef is [-1,1].
- The more +ve the coeff is the more +ve the strength will be.
- The more -ve the coeff is, the more -ve the strenght will be.
- If the corrcoeff is 0 then there is no correlation between two
numeric columns.
```

```python
for col in num_cols:
    val = np.corrcoef(yulu_data[col].rank(), 
yulu_data['log_count'].rank())[0,1]
    if val> 0 :
        print('There is +ve relation between count and', col,"-
corrcoef: ",np.round(val,2))
    if val == 0:
        print('There is no relation between count and ', col,"-
corrcoef: ",np.round(val,2))
    if val< 0:
        print('There is -ve relation between count and ', col,"-
corrcoef: ",np.round(val,2))
```

```
There is +ve relation between count and temp - corrcoef:  0.41
There is +ve relation between count and atemp - corrcoef:  0.41
There is -ve relation between count and  humidity - corrcoef:  -0.35
There is +ve relation between count and windspeed - corrcoef:  0.14
There is +ve relation between count and casual - corrcoef:  0.85
There is +ve relation between count and registered - corrcoef:  0.99
```

Insights:

From the above calculated data we can say that count of bikes rented depends on temp, windspeed and strongly dependent on casual and registered columns. count is negatively dependent on humidity. The less humid the weather is more the bikes are rented.

2. Checking if Working Day has an effect on the number of electric cycles rented

We are going to conduct 2-sample test as workingday has two categories.

```
-The two samples of count for 0 workingday and 1 workingday are 
independent.
- Null hypothesis will be "the mean values of the two sets of data are
equal".
- Alternate hypothesis will be "the mean values of the two sets of 
data are not equal".
- Let us conduct this test with 0.05 significance.
- If the p-value obtained from the test is less than the significance
value we say there are enough evidences to reject null hypothesis.
- Else we say that there are no enough evidences to reject null 
hyposthesis.

workingday_0_count = yulu_data[yulu_data['workingday']=='0']
['log_count']
workingday_1_count = yulu_data[yulu_data['workingday']=='1']
['log_count']
```

```
H0 = 'count of bikes rented doesnot depend on workingday' #says the
means are equal
Ha = 'count of bikes rented depend on workingday' # says means are not
equal

alpha = 0.05

_, p_val = ttest_ind(workingday_0_count,workingday_1_count,
alternative= 'less')

if p_val < alpha:
    print("p_value: ", np.round(p_val,2),'\n',"Reject H0: count of
bikes rented on non-working day is less than working day")
else:
     print("p_value: ", np.round(p_val,2),'\n',"Unable to Reject H0:",
H0)

p_value:  0.97
 Unable to Reject H0: count of bikes rented doesnot depend on
workingday
```

Insights:

```
  From the 2-sample ttest done the p-vaue is greater than alpha, so
there are no enough evidences to reject null hypothesis, which
concludes "The count of rented bikes doesnot depend on workingday."
```

3. Checking if count of bikes depends on season

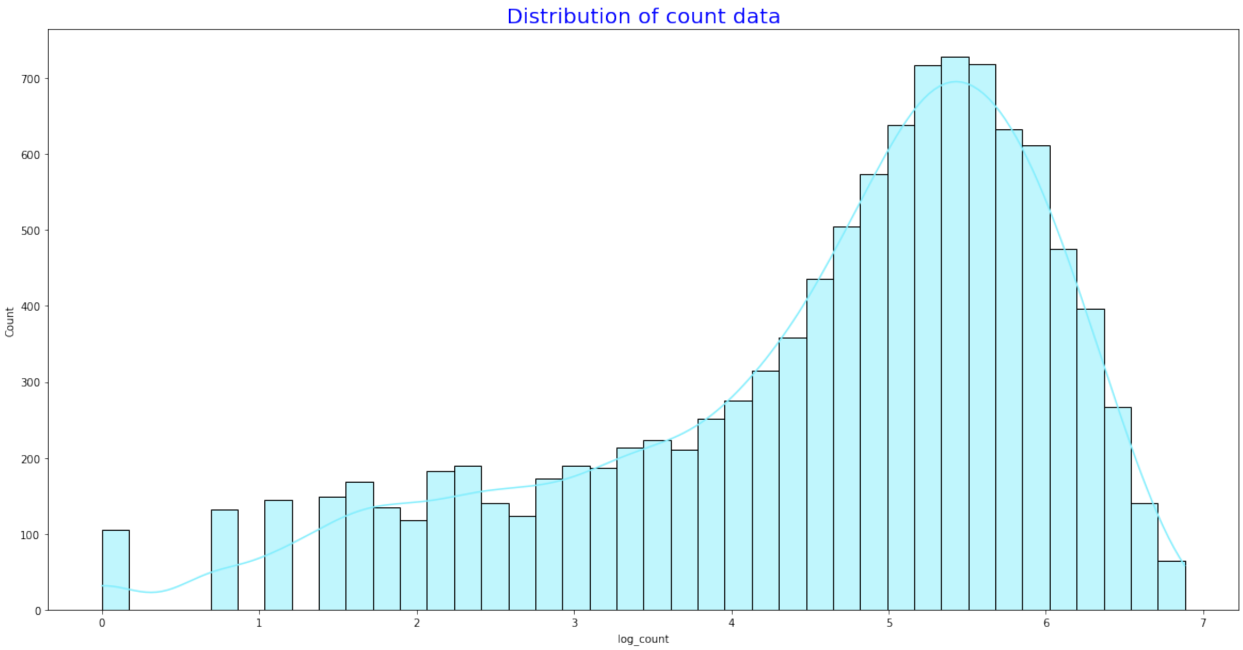
We are going to conduct ANOVA test on count and season as one is num col and the other is cat col with more than 2 categories. Checking for ANOVA test assumptions.

1. Check wheter data is normal using histplot, qqplot, shapiro test.
2. Checking of equal variance for different categories using levens test.
3. Checking if each category data is independent of each other.

If any of the abve assumptions fail go for kruskal test.

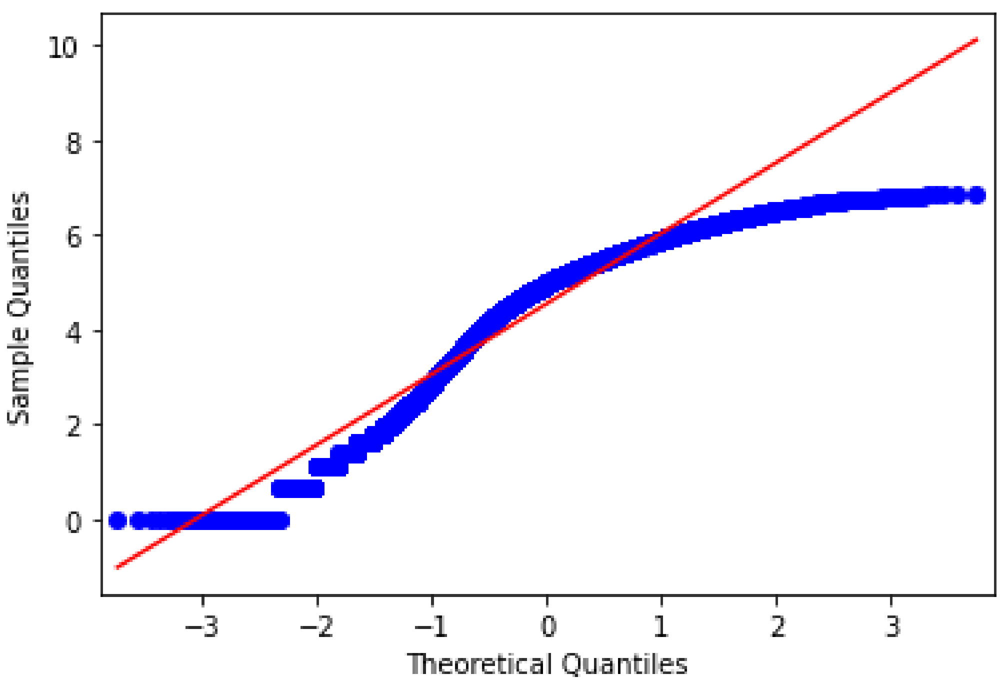
1. CHECKING IF DATA IS NORMALLY DISTRIBUTED.

```
# checking if the count data is normal using histplot

plt.figure(figsize=(20,10))
sns.histplot(yulu_data['log_count'], kde= True,color = '#82EEFD')
plt.title("Distribution of count data", fontsize = 20, color = 'blue')
plt.show()
```



count data is not normal from the above histplot.

```
#checking if the count data is normal using qqplot
qqplot(yulu_data['log_count'], line='s')
plt.show()
```



Even with qqplot the count data is not normal.

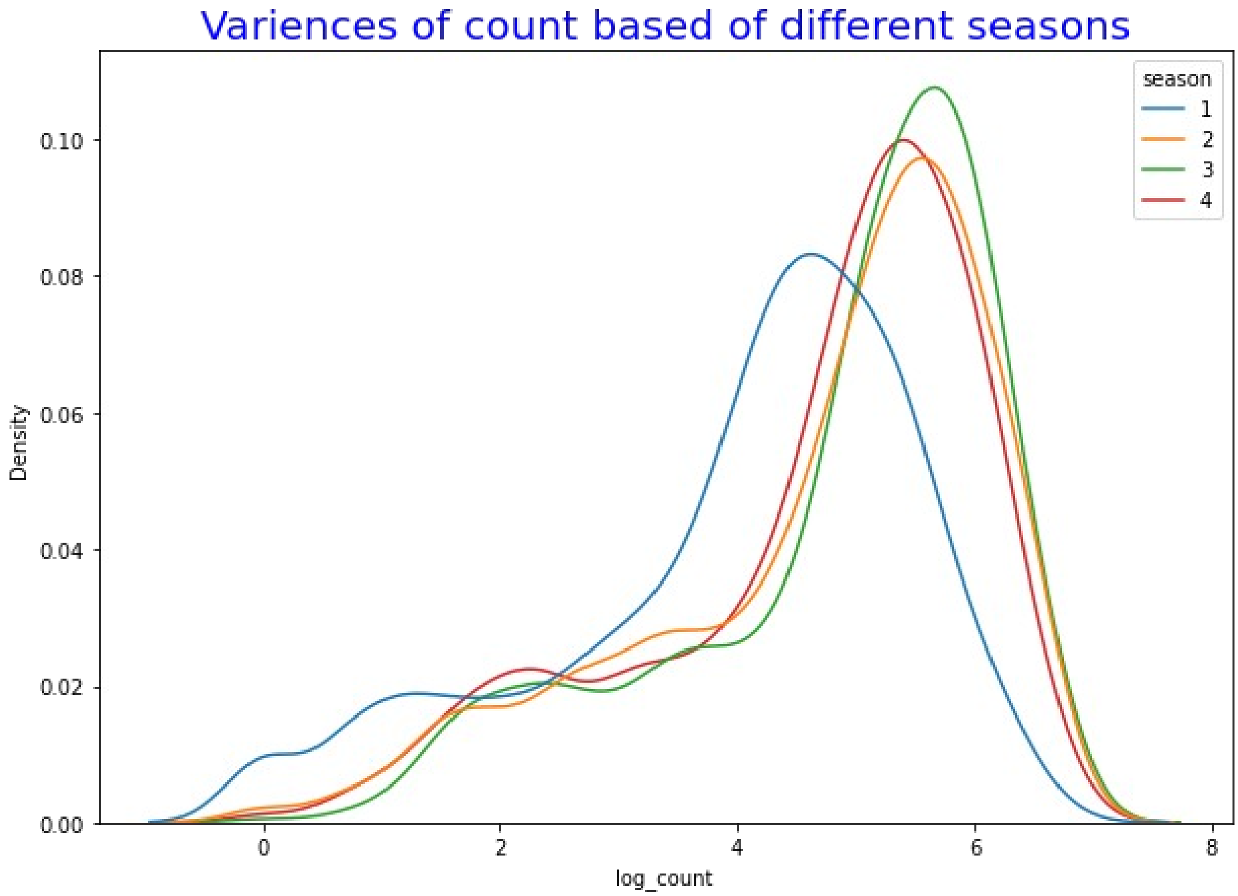
```
#checking if the count data is normal using shapiro test

H0 = 'count data is normal' # says count is normally distributed
Ha = 'count data is not normal' #says count is not normally
distributed

alpha = 0.05

_,p_val = shapiro(yulu_data['log_count'].sample(100))

if p_val < alpha:
    print("p_value: ", np.round(p_val,2),"Reject H0:", Ha)
else:
     print("p_value: ", np.round(p_val,2),"Unable to Reject H0:", H)

p_value:  0.0 Reject H0: count data is not normal
```

Insights:

```
From the histplot, qqplot and shapiro test results we conclude that
count is not normally distributed.
```

1. CHECKING FOR EQUAL VARIANCE FOR DIFFERENT CATEGORIES USING LEVENS TEST.

```
# dividing data based on seasons

season1 = yulu_data[yulu_data['season']== '1']['log_count']
season2 = yulu_data[yulu_data['season']== '2']['log_count']
season3 = yulu_data[yulu_data['season']== '3']['log_count']
season4 = yulu_data[yulu_data['season']== '4']['log_count']

plt.figure(figsize=(10,7))
sns.kdeplot(x= yulu_data['log_count'], hue = yulu_data['season'])
plt.title("Variences of count based of different seasons",
fontsize=20, color= 'blue')
plt.show()
```




```
H0 = 'Distributions have equal varience'
Ha = 'Distributions donot have equal varience'

alpha = 0.05

_,p_val = levene(season1,season2,season3,season4)

if p_val < alpha:
    print("p_value: ", np.round(p_val,2),"Reject H0:", Ha)
else:
     print("p_value: ", np.round(p_val,2),"Unable to Reject H0:", H0)

p_value:  0.0 Reject H0: Distributions donot have equal varience
```

Insights:

```
From the above kdeplot and Leven's test results we an conclude that
the groups are not equally distributed.
```

1. CHECKING IF EACH CATEGORY DATA IS INDEPENDENT OF EACH OTHER.

As the data of count is categorised based on season , we say that the data in the groups is independent of each other.

As from the first and second assumptions failed,ie the data is not normally distributed and doenot have equal variences, going for kruskal's test.

```
#kruskal test for checking if count of bikes depend on season

H0 = 'count of bikes rented does not vary with season' # says all groups have same mean
Ha = 'count of bikes rented vary with season' # says all atleat one or more groups have different mean

alpha = 0.05

_, p_val = kruskal(season1,season2, season3, season4)
if p_val < alpha:
    print("p_value: ", np.round(p_val,2),"Reject H0:", Ha)
else:
     print("p_value: ", np.round(p_val,2),"Unable to Reject H0:", H)
```

```
p_value:  0.0 Reject H0: count of bikes rented vary with season
```

Insights:

```
    From the Kruskal's test result we conclude that the count of bikes rented vary with season.
```

4. Checking if count depends on weather

```
# grouping count based of weather categories

weather1 = yulu_data[yulu_data['weather']== '1']['log_count']
weather2 = yulu_data[yulu_data['weather']== '2']['log_count']
weather3 = yulu_data[yulu_data['weather']== '3']['log_count']
weather4 = yulu_data[yulu_data['weather']== '4']['log_count']

# As from the shapiro test we know that count is not normally is not normally distributed we go with kruskal's test
# kruskal test for checking if count of bikes depend on weather

H0 = 'count of bikes rented does not vary with weather' # says all groups have same mean
Ha = 'count of bikes rented vary with weather' # says one or more groups have different mean

_, p_val = kruskal(season1,season2, season3, season4)

if p_val < alpha:
    print("p_value: ", np.round(p_val,2),"Reject H0:", Ha)
else:
     print("p_value: ", np.round(p_val,2),"Unable to Reject H0:", H0)
```

```
p_value:  0.0 Reject H0: count of bikes rented vary with weather
```

Insights:

```
  From the above kruskal's test on grouped count data based on season
we say that count of bikes rented vary with weather.
```

5. Checking if weather is dependent on the season

As weather and season are both categorial columns we use Chisquare test.

```
   Assumptions of chisquare test are:
    - Variables should be categorical.
    - Observations should be inepandent of each other.
    - Expected values in any cell should be greater than 5.

# Creating crosstab for season and weather.
season_weather_cross_tab = pd.crosstab(yulu_data.season,
yulu_data.weather)
season_weather_cross_tab
```

```
weather     1    2    3  4
season
1        1759  715  211  1
2        1801  708  224  0
3        1930  604  199  0
4        1702  807  225  0
```

```
H0= 'weather and season are independent'
Ha= 'weather and season are dependent'

alpha = 0.05

_,p_val,_,_ = chi2_contingency(season_weather_cross_tab)

if p_val < alpha:
    print("p_value: ", np.round(p_val,2),"Reject H0:", Ha)
else:
     print("p_value: ", np.round(p_val,2),"Unable to Reject H0:", H0)
```

```
p_value:  0.0 Reject H0: weather and season are dependent
```

Insights:

```
From the above chisquare contingency test result we conclude that
"weather and season are dependent"
```