yulu-case-study

May 8, 2025

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


from scipy.stats import ttest_ind
from scipy.stats import chisquare # Statistical test (chistat, pvalue)
from scipy.stats import chi2
from scipy.stats import chi2_contingency
from scipy.stats import f_oneway # Numeric Vs categorical for many categories

from scipy.stats import kruskal
from scipy.stats import shapiro # Test Gaussian (50 to 200 samples)

from scipy.stats import levene # Test variance
import statsmodels.api as sm
from statsmodels.formula.api import ols
from scipy import stats
```

[2]:
```
df = pd.read_csv('bike_sharing.csv')
df.head()
```

[2]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

[3]:
```
df.shape
```

[3]:
```
(10886, 12)
```

[4]:
```
df.describe()
```

[4]:

	season	holiday	workingday	weather	temp	\
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	
mean	2.506614	0.028569	0.680875	1.418427	20.23086	
std	1.116174	0.166599	0.466159	0.633839	7.79159	
min	1.000000	0.000000	0.000000	1.000000	0.82000	
25%	2.000000	0.000000	0.000000	1.000000	13.94000	
50%	3.000000	0.000000	1.000000	1.000000	20.50000	
75%	4.000000	0.000000	1.000000	2.000000	26.24000	
max	4.000000	1.000000	1.000000	4.000000	41.00000	

	atemp	humidity	windspeed	casual	registered	\
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	
mean	23.655084	61.886460	12.799395	36.021955	155.552177	
std	8.474601	19.245033	8.164537	49.960477	151.039033	
min	0.760000	0.000000	0.000000	0.000000	0.000000	
25%	16.665000	47.000000	7.001500	4.000000	36.000000	
50%	24.240000	62.000000	12.998000	17.000000	118.000000	
75%	31.060000	77.000000	16.997900	49.000000	222.000000	
max	45.455000	100.000000	56.996900	367.000000	886.000000	

	count
count	10886.000000
mean	191.574132
std	181.144454
min	1.000000
25%	42.000000
50%	145.000000
75%	284.000000
max	977.000000

[5]:
```
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
```

[6]:
```
df['datetime'] = pd.to_datetime(df['datetime'])
```

[7]:
```
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  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: datetime64[ns](1), float64(3), int64(8)
memory usage: 1020.7 KB
```

[8]:
```
df.isnull().sum()
```

[8]:
```
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
```

1 #There are No missing values here::

---

[8]:
```
```

[9]:
```
print(df['season'].value_counts())
print('-----------------------------------')
print(df['holiday'].value_counts())
print('0 :  Working day ')
print('1 :  Holiday')
print('-----------------------------------')

print(df['workingday'].value_counts())
print('0 :  Holiday ')
print('1 :  Working day')
print('-----------------------------------')

print(df['weather'].value_counts())
```

```
season
4    2734
2    2733
3    2733
1    2686
Name: count, dtype: int64
-----------------------------------
holiday
0    10575
1      311
Name: count, dtype: int64
0 :  Working day
1 :  Holiday
-----------------------------------
workingday
1    7412
0    3474
Name: count, dtype: int64
0 :  Holiday
1 :  Working day
-----------------------------------
weather
1    7192
```

```
2    2834
3     859
4       1
Name: count, dtype: int64
```

[10]:
```
season_map = {
    1: 'Spring',
    2: 'Summer',
    3: 'Fall',
    4: 'Winter'
}
df['season_name'] = df['season'].map(season_map)
```

We will apply this mapping for whether for clear understanding

1 Clear, Few clouds, partly cloudy, partly cloudy — Clear/Cloudy

2 Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist — Misty

3 Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain — Light Precip

4 Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog — Severe Weather

[11]:
```
weather_map = {
    1: 'Clear/Cloudy',
    2: 'Misty',
    3: 'Light Precip',
    4: 'Severe Weather'
}
df['weather_label'] = df['weather'].map(weather_map)
```

#We Will convert Datetime feature to Hour Day Month Year

[12]:
```
df['hour'] = df['datetime'].dt.hour
df['day'] = df['datetime'].dt.day
df['monthname'] = df['datetime'].dt.month_name().str[:3]
df['year'] = df['datetime'].dt.year
```

[13]:
```
df.head()
```

[13]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	season_name	weather_label
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	Spring	Clear/Cloudy
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	Spring	Clear/Cloudy
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635							
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395							
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395							

```
2         80        0.0       5           27      32       Spring  Clear/Cloudy
3         75        0.0       3           10      13       Spring  Clear/Cloudy
4         75        0.0       0            1       1       Spring  Clear/Cloudy

   hour  day monthname  year
0     0    1       Jan  2011
1     1    1       Jan  2011
2     2    1       Jan  2011
3     3    1       Jan  2011
4     4    1       Jan  2011
```

[14]:
```
print(f'Data lies from {min(df.monthname)} {min(df.year)} to {max(df.
↪monthname)} {max(df.year)} ')
```

```
Data lies from Apr 2011 to Sep 2012
```

[14]:
```
```

2 Univariate Analysis

[15]:
```
df.head()
```

[15]:
```
             datetime  season  holiday  workingday  weather  temp   atemp  \
0 2011-01-01 00:00:00       1        0           0        1  9.84  14.395
1 2011-01-01 01:00:00       1        0           0        1  9.02  13.635
2 2011-01-01 02:00:00       1        0           0        1  9.02  13.635
3 2011-01-01 03:00:00       1        0           0        1  9.84  14.395
4 2011-01-01 04:00:00       1        0           0        1  9.84  14.395

   humidity  windspeed  casual  registered  count season_name weather_label  \
0        81        0.0       3          13     16      Spring  Clear/Cloudy
1        80        0.0       8          32     40      Spring  Clear/Cloudy
2        80        0.0       5          27     32      Spring  Clear/Cloudy
3        75        0.0       3          10     13      Spring  Clear/Cloudy
4        75        0.0       0           1      1      Spring  Clear/Cloudy

   hour  day monthname  year
0     0    1       Jan  2011
1     1    1       Jan  2011
2     2    1       Jan  2011
3     3    1       Jan  2011
4     4    1       Jan  2011
```

[16]:
```
col_to_move = 'count'

# Reorder columns
```

```
df = df[[col for col in df.columns if col != col_to_move] + [col_to_move]]
df.head()
```

[16]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	season_name	weather_label	hour	day	monthname	year	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	Spring	Clear/Cloudy	0	1	Jan	2011	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	Spring	Clear/Cloudy	1	1	Jan	2011	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	Spring	Clear/Cloudy	2	1	Jan	2011	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	Spring	Clear/Cloudy	3	1	Jan	2011	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	Spring	Clear/Cloudy	4	1	Jan	2011	1

[17]:
```
df.drop(columns= ['season','weather','datetime'], inplace = True)
df.head()
```

```
<ipython-input-17-4f41ae2e68bf>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.drop(columns= ['season','weather','datetime'], inplace = True)
```

[17]:

	holiday	workingday	temp	atemp	humidity	windspeed	casual	registered	season_name	weather_label	hour	day	monthname	year	count
0	0	0	9.84	14.395	81	0.0	3	13	Spring	Clear/Cloudy	0	1	Jan	2011	16
1	0	0	9.02	13.635	80	0.0	8	32	Spring	Clear/Cloudy	1	1	Jan	2011	40
2	0	0	9.02	13.635	80	0.0	5	27	Spring	Clear/Cloudy	2	1	Jan	2011	32
3	0	0	9.84	14.395	75	0.0	3	10	Spring	Clear/Cloudy	3	1	Jan	2011	13
4	0	0	9.84	14.395	75	0.0	0	1	Spring	Clear/Cloudy	4	1	Jan	2011	1

[21]:
```
df.describe()
```

[21]:

	holiday	workingday	temp	atemp	humidity
count	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000
mean	0.028569	0.680875	20.23086	23.655084	61.886460
std	0.166599	0.466159	7.79159	8.474601	19.245033
min	0.000000	0.000000	0.82000	0.760000	0.000000
25%	0.000000	0.000000	13.94000	16.665000	47.000000
50%	0.000000	1.000000	20.50000	24.240000	62.000000
75%	0.000000	1.000000	26.24000	31.060000	77.000000
max	1.000000	1.000000	41.00000	45.455000	100.000000

	windspeed	casual	registered	hour	day
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	12.799395	36.021955	155.552177	11.541613	9.992559
std	8.164537	49.960477	151.039033	6.915838	5.476608
min	0.000000	0.000000	0.000000	0.000000	1.000000
25%	7.001500	4.000000	36.000000	6.000000	5.000000
50%	12.998000	17.000000	118.000000	12.000000	10.000000
75%	16.997900	49.000000	222.000000	18.000000	15.000000
max	56.996900	367.000000	886.000000	23.000000	19.000000

	year	count
count	10886.000000	10886.000000
mean	2011.501929	191.574132
std	0.500019	181.144454
min	2011.000000	1.000000
25%	2011.000000	42.000000
50%	2012.000000	145.000000
75%	2012.000000	284.000000
max	2012.000000	977.000000

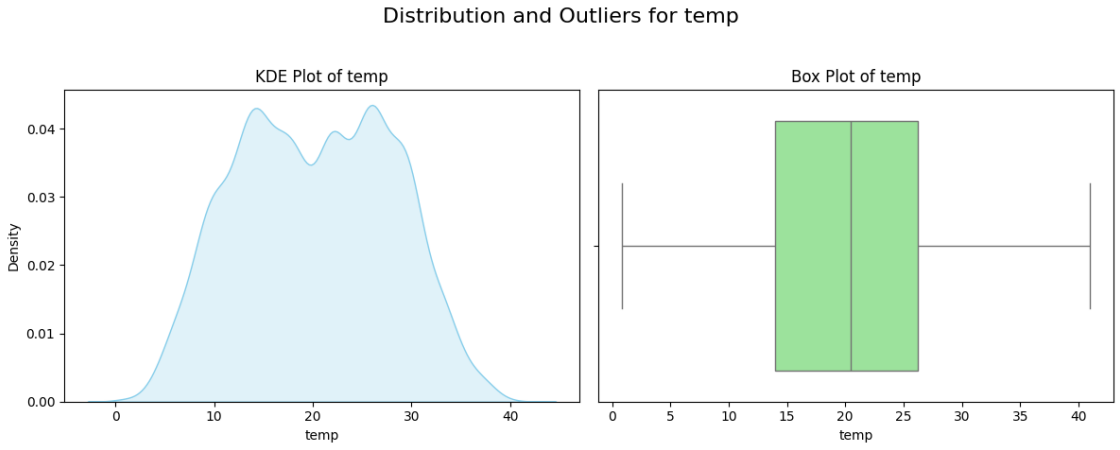
[17]:
```
```

3 Univariate Analysis

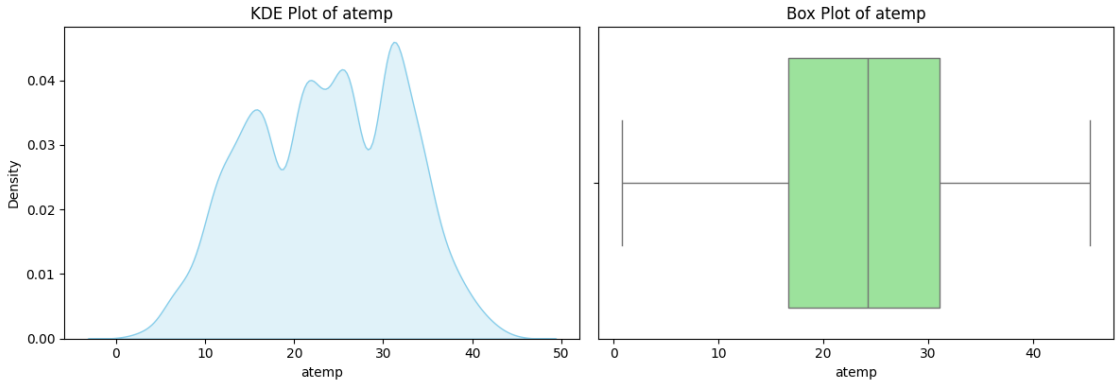
[17]:
```
num_features = ['temp', 'atemp','humidity', 'windspeed','casual','registered']
```

[22]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats

# Assuming your dataframe is called df
num_features = ['temp', 'atemp', 'humidity', 'windspeed', 'casual',␣
 ↪'registered']
```

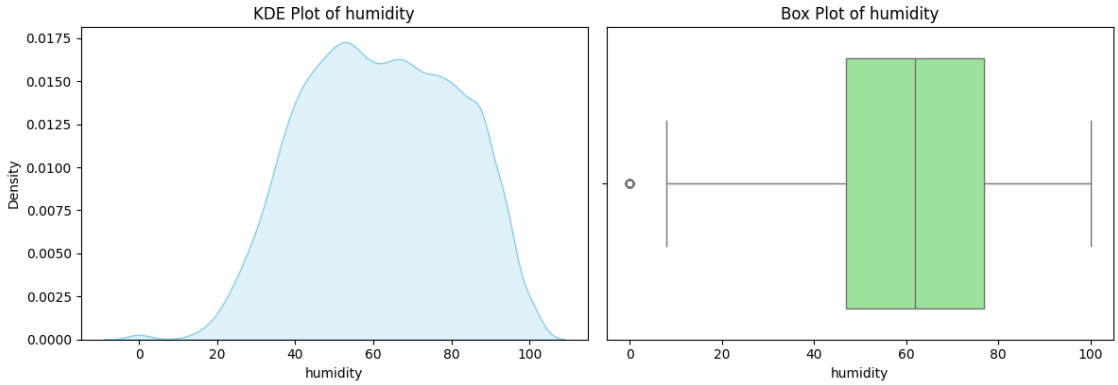
```python
for col in num_features:
    plt.figure(figsize=(12,5))

    # First subplot: KDE Plot
    plt.subplot(1,2,1)
    sns.kdeplot(df[col], fill=True, color='skyblue')
    plt.title(f'KDE Plot of {col}')
    plt.xlabel(col)

    # Second subplot: Boxplot
    plt.subplot(1,2,2)
    sns.boxplot(x=df[col], color='lightgreen')
    plt.title(f'Box Plot of {col}')
    plt.xlabel(col)

    plt.suptitle(f'Distribution and Outliers for {col}', fontsize=16)
    plt.tight_layout(rect=[0, 0.03, 1, 0.95])
    plt.show()

    # Skewness and Kurtosis
    skewness = df[col].skew()
    kurtosis = df[col].kurt()

    print(f"Feature: {col}")
    print(f"Skewness: {skewness:.3f}")
    print(f"Kurtosis: {kurtosis:.3f}")
    print('-'*50)
```



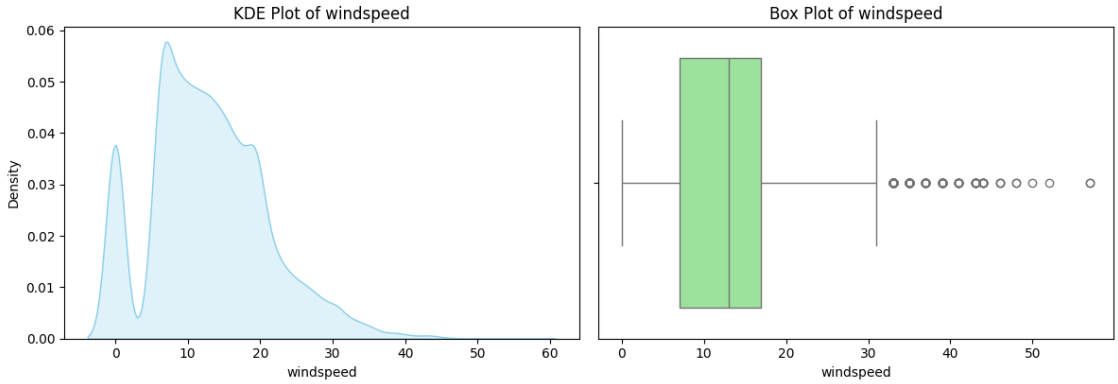

```
Feature: temp
Skewness: 0.004
Kurtosis: -0.915
--------------------------------------------------
```



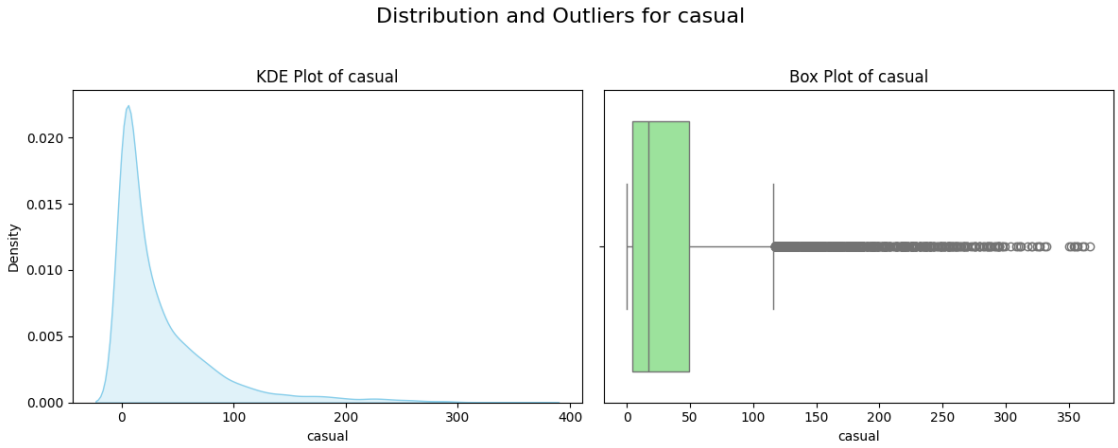

```
Feature: atemp
Skewness: -0.103
Kurtosis: -0.850
--------------------------------------------------
```



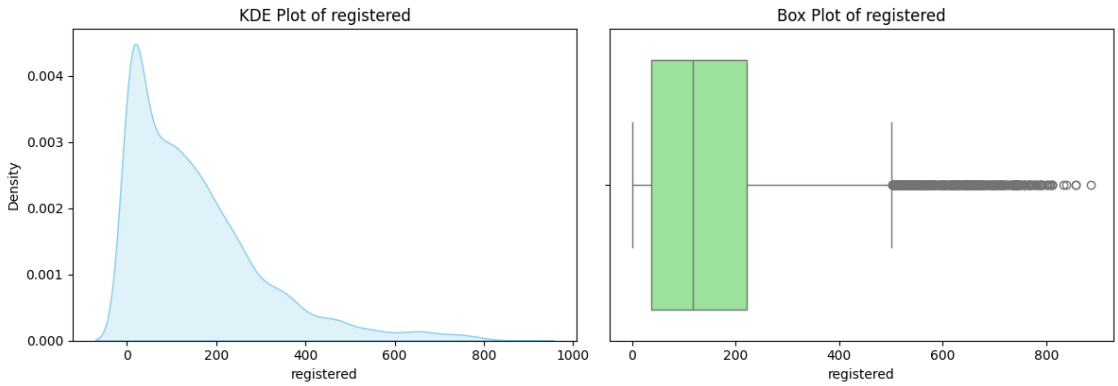

```
Feature: humidity
Skewness: -0.086
Kurtosis: -0.760
--------------------------------------------------
```




```
Feature: windspeed
Skewness: 0.589
Kurtosis: 0.630
-------------------------------------------------
```




```
Feature: casual
Skewness: 2.496
Kurtosis: 7.552
-------------------------------------------------
```




```
Feature: registered
Skewness: 1.525
Kurtosis: 2.626
-------------------------------------------------
```

4 Checking For outliers using IQR Method:

[28]:
```
def detect_outliers(data, column):
  q1 = data[column].quantile(0.25)
  q3 = data[column].quantile(0.75)
  iqr = q3-q1
  lowerbound = q1 - (1.5*iqr)
  upperbound = q3 + (1.5*iqr)
  outliers = data[(data[column] < lowerbound) | (data[column] > upperbound)]
  return outliers
```

[29]:
```
print(detect_outliers(df,'casual'))
```

```
       holiday  workingday   temp   atemp  humidity  windspeed  casual  \
1173         0           0  18.86  22.725        41    19.9995     144
1174         0           0  19.68  23.485        39    22.0028     149
1175         0           0  18.86  22.725        41    26.0027     124
1311         0           0  18.86  22.725        33    27.9993     126
1312         0           0  20.50  24.240        34    31.0009     174
...        ...         ...    ...     ...       ...         ...     ...
10610        0           0  16.40  20.455        87    15.0013     122
10611        0           0  16.40  20.455        87    11.0014     148
10612        0           0  16.40  20.455        87    19.0012     164
10613        0           0  17.22  21.210        82    11.0014     167
10614        0           0  17.22  21.210        82    11.0014     139
```

```
      registered season_name weather_label  hour  day monthname  year  count
1173         106      Spring  Clear/Cloudy    14   13       Mar  2011    250
1174         155      Spring  Clear/Cloudy    15   13       Mar  2011    304
1175         132      Spring  Clear/Cloudy    16   13       Mar  2011    256
1311         141      Spring  Clear/Cloudy    12   19       Mar  2011    267
1312         127      Spring  Clear/Cloudy    13   19       Mar  2011    301
…            …           …             …    …  …       …    …    …
10610        364      Winter         Misty    12    8       Dec  2012    486
10611        399      Winter         Misty    13    8       Dec  2012    547
10612        378      Winter         Misty    14    8       Dec  2012    542
10613        374      Winter  Clear/Cloudy    15    8       Dec  2012    541
10614        368      Winter  Clear/Cloudy    16    8       Dec  2012    507

[749 rows x 15 columns]
```

[31]:
```
print(detect_outliers(df,'registered'))
```

```
      holiday  workingday   temp   atemp  humidity  windspeed  casual  \
1987        0           1  25.42  31.060        38    16.9979      59
2011        0           1  26.24  31.060        33     0.0000      79
2059        0           1  26.24  31.060        57    12.9980      54
2179        0           1  25.42  30.305        65    27.9993      83
2371        0           1  31.98  34.090        33    19.0012      63
…         …           …     …      …         …         …      …
10855       0           1  16.40  20.455        47    30.0026      39
10856       0           1  15.58  19.695        46    22.0028      13
10870       0           1   9.84  12.880        87     7.0015      13
10879       0           1  16.40  20.455        50    26.0027      26
10880       0           1  15.58  19.695        50    23.9994      23

      registered season_name weather_label  hour  day monthname  year  count
1987         539      Summer  Clear/Cloudy    17    9       May  2011    598
2011         532      Summer  Clear/Cloudy    17   10       May  2011    611
2059         540      Summer         Misty    17   12       May  2011    594
2179         521      Summer  Clear/Cloudy    17   17       May  2011    604
2371         516      Summer  Clear/Cloudy    17    6       Jun  2011    579
…            …           …             …    …  …       …    …    …
10855        533      Winter  Clear/Cloudy    17   18       Dec  2012    572
10856        512      Winter  Clear/Cloudy    18   18       Dec  2012    525
10870        665      Winter  Clear/Cloudy     8   19       Dec  2012    678
10879        536      Winter  Clear/Cloudy    17   19       Dec  2012    562
10880        546      Winter  Clear/Cloudy    18   19       Dec  2012    569

[423 rows x 15 columns]
```

[ ]:
```

```

5 Observation:

After applying simple KDE plots and Box plots to See outliers for now we can say

1.Temp, Atemp, Humidity is approximately normally distributed.

2.It Seems there are multiple outliers in Casaul and Registered Fields and are Right Skewed,

However considering the business context of the Yulu Bike case study, these extreme values are likely to represent valid scenarios, such as sudden surges in bike rentals due to events, weekends, or weather conditions. So we are keeping the same as of now.

[32]:
```
df.head()
```

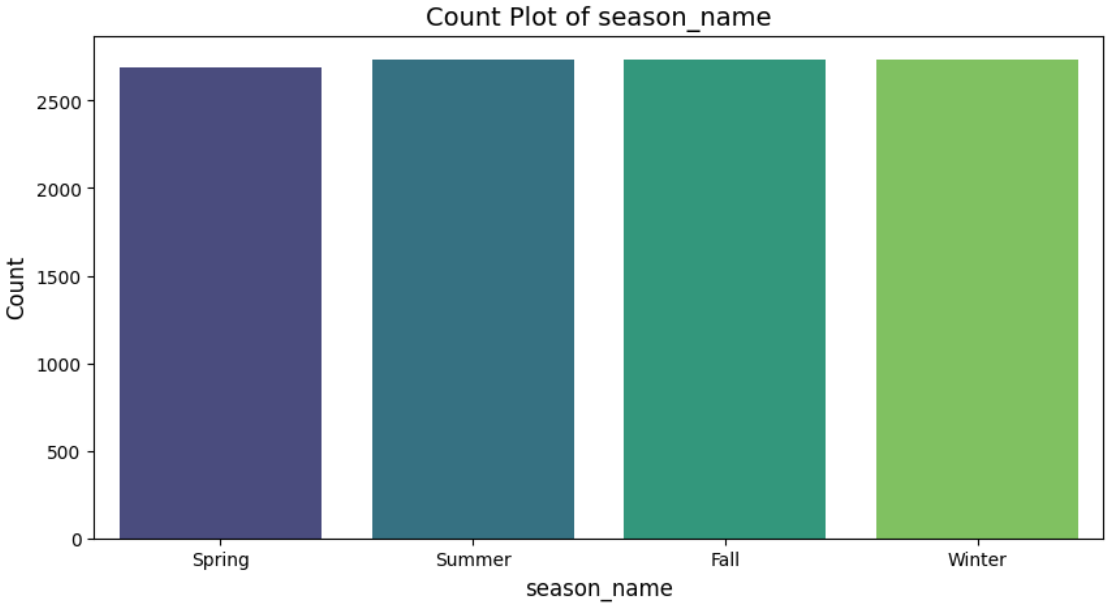
[32]:

	holiday	workingday	temp	atemp	humidity	windspeed	casual	registered
0	0	0	9.84	14.395	81	0.0	3	13
1	0	0	9.02	13.635	80	0.0	8	32
2	0	0	9.02	13.635	80	0.0	5	27
3	0	0	9.84	14.395	75	0.0	3	10
4	0	0	9.84	14.395	75	0.0	0	1

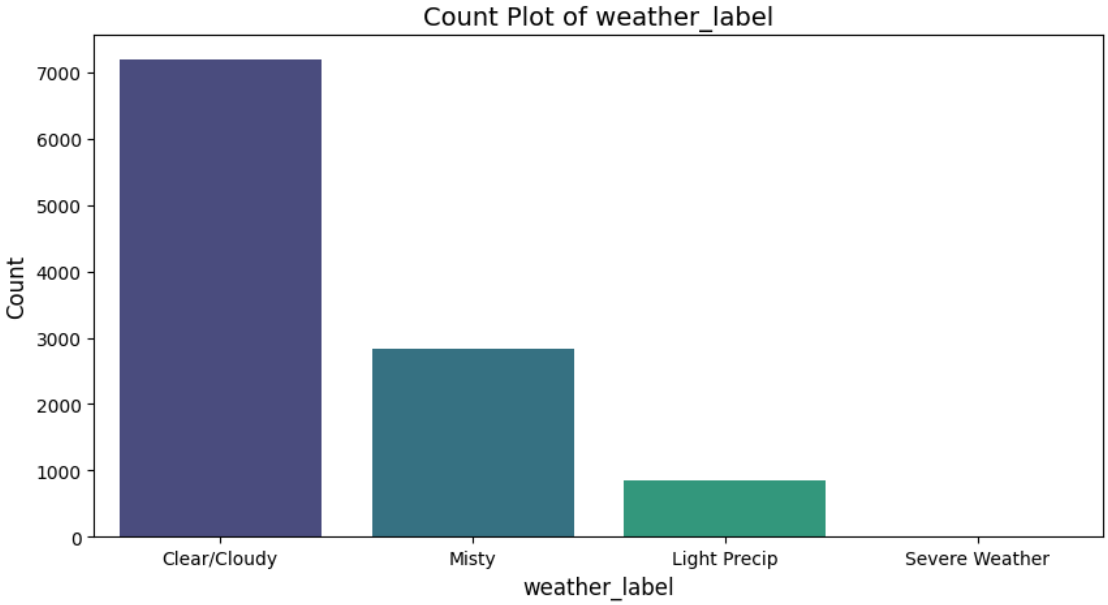
	season_name	weather_label	hour	day	monthname	year	count
0	Spring	Clear/Cloudy	0	1	Jan	2011	16
1	Spring	Clear/Cloudy	1	1	Jan	2011	40
2	Spring	Clear/Cloudy	2	1	Jan	2011	32
3	Spring	Clear/Cloudy	3	1	Jan	2011	13
4	Spring	Clear/Cloudy	4	1	Jan	2011	1

[36]:
```
cols = ['season_name', 'weather_label', 'monthname', 'year']

for col in cols:
    plt.figure(figsize=(10,5))
    sns.countplot(x=df[col], palette='viridis')  # you can change palette if␣
 ↪you want
    plt.title(f'Count Plot of {col}', fontsize=14)
    plt.xlabel(col, fontsize=12)
    plt.ylabel('Count', fontsize=12)
    plt.show()
```

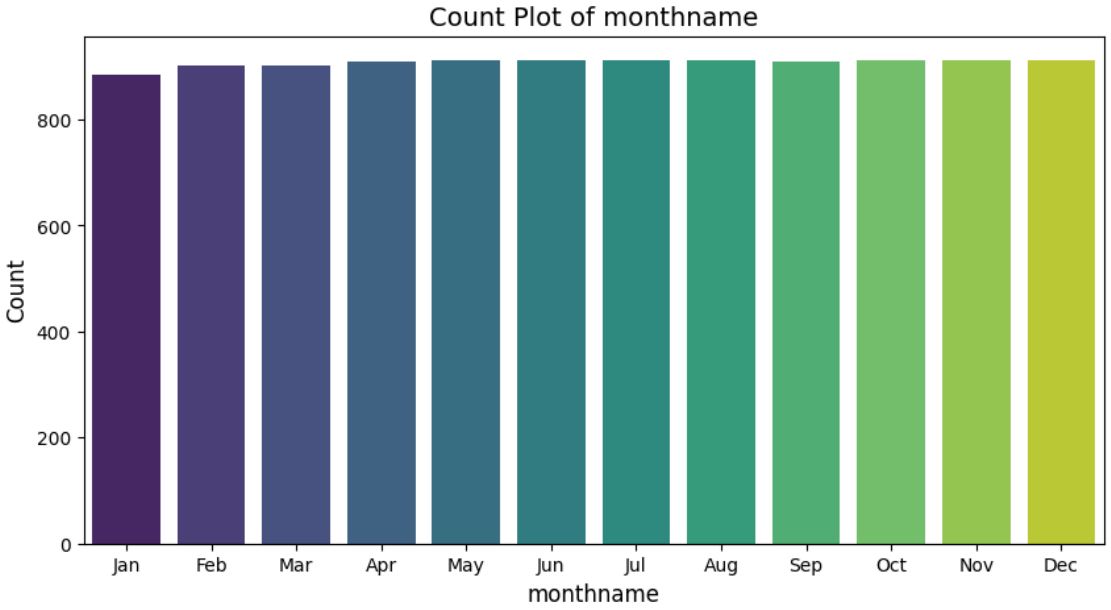
```
<ipython-input-36-f23bf137deb2>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x=df[col], palette='viridis')  # you can change palette if you
want
```



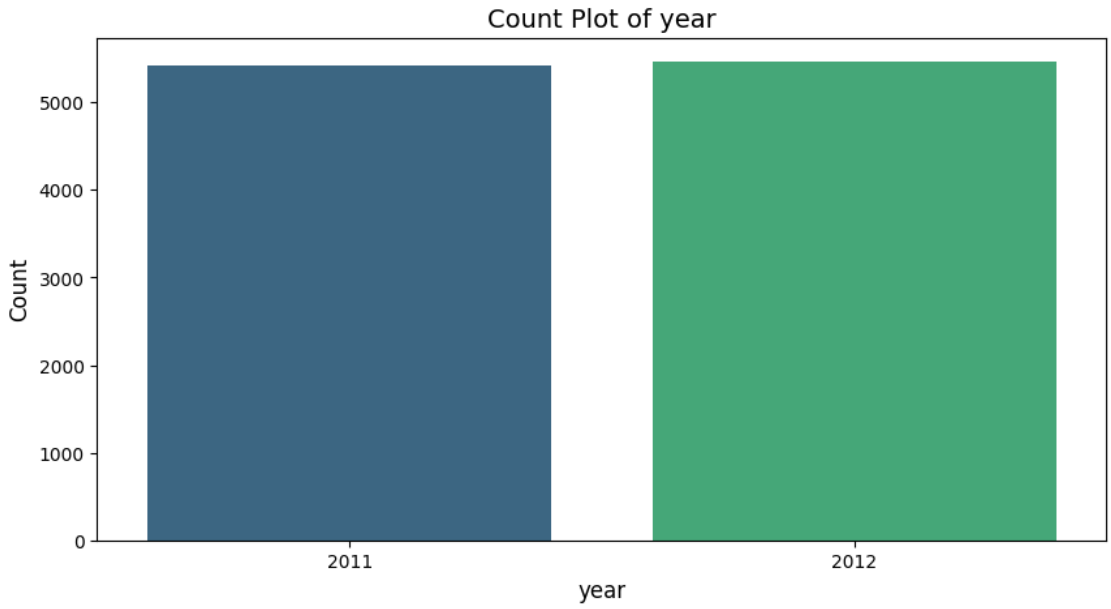

```
<ipython-input-36-f23bf137deb2>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x=df[col], palette='viridis')  # you can change palette if you
want
```



```
<ipython-input-36-f23bf137deb2>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x=df[col], palette='viridis')  # you can change palette if you
want
```




```
<ipython-input-36-f23bf137deb2>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x=df[col], palette='viridis')  # you can change palette if you
want
```



6 Observation:

1.For Season name almost equal data is avalibale for all seasons.Similarly for Monthname

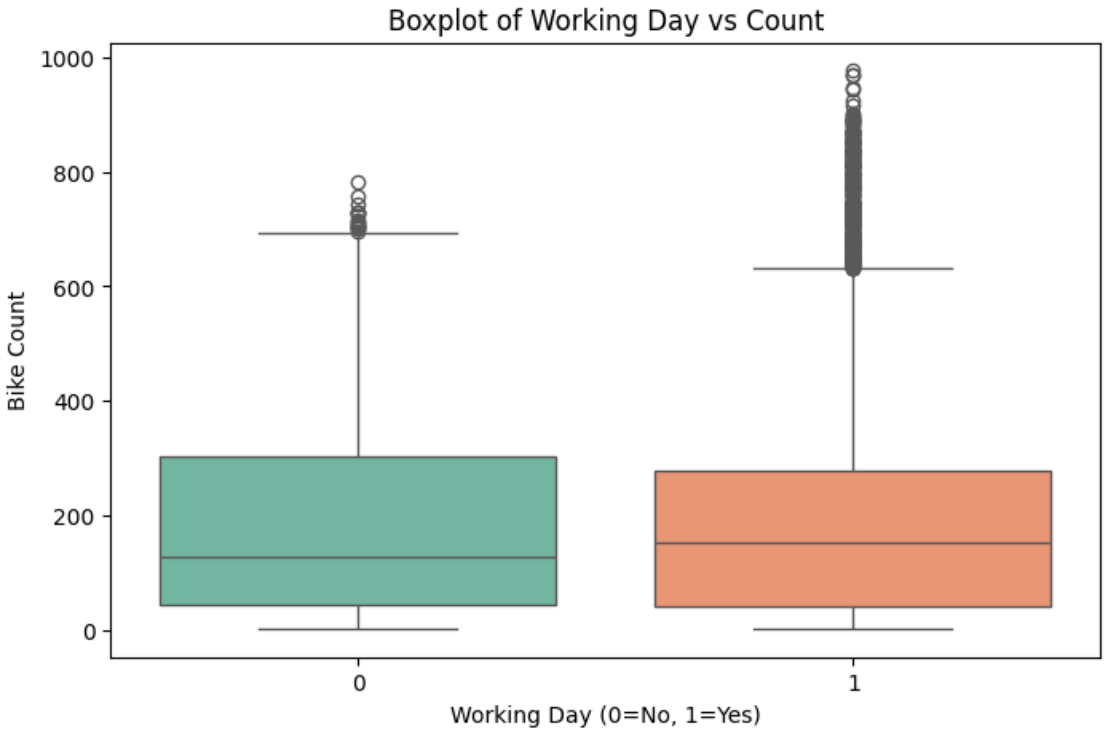
2.For Weather conditions,Majority of rides were recorded during Clear/Cloudy weather conditions, indicating that users are most likely to use Yulu bikes when the weather is pleasant.

[18]:

7 Bi-Variate Analysis

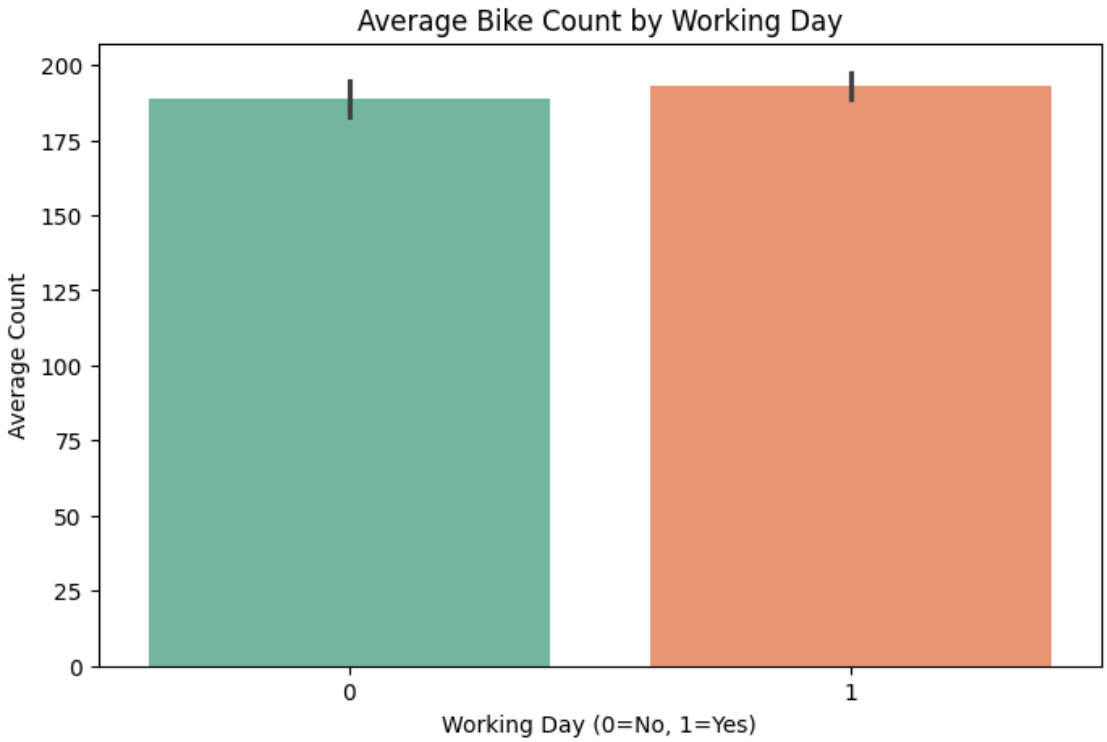
[37]:
```
plt.figure(figsize=(8,5))
sns.boxplot(x='workingday', y='count', data=df, palette='Set2')
plt.title('Boxplot of Working Day vs Count')
plt.xlabel('Working Day (0=No, 1=Yes)')
plt.ylabel('Bike Count')
plt.show()
```

```
<ipython-input-37-a34c47ddc3fc>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='workingday', y='count', data=df, palette='Set2')
```



[38]:
```
plt.figure(figsize=(8,5))
sns.barplot(x='workingday', y='count', data=df, estimator='mean',␣
 ↪palette='Set2')
plt.title('Average Bike Count by Working Day')
plt.xlabel('Working Day (0=No, 1=Yes)')
plt.ylabel('Average Count')
plt.show()
```

```
<ipython-input-38-d0930dc09cd4>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='workingday', y='count', data=df, estimator='mean',
palette='Set2')
```



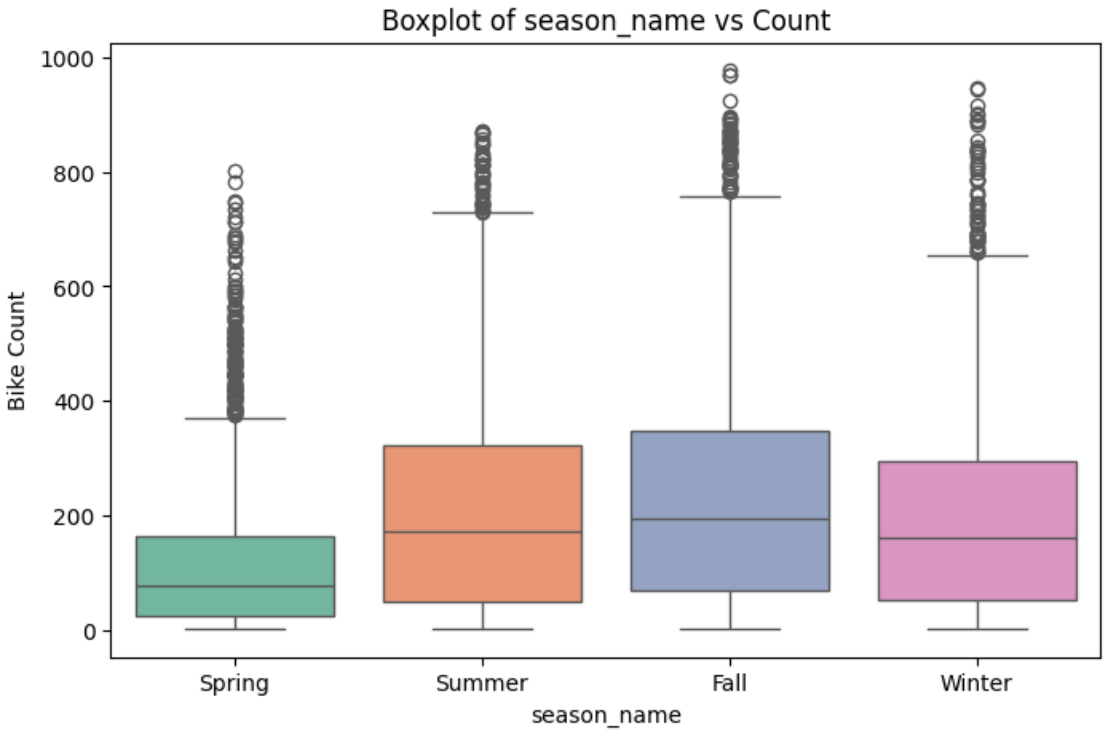
8 Observations

1.Usage of YULU Bikes is slightly higher in working days. But indicating regular use of YULU bikes for computation.

[ ]:

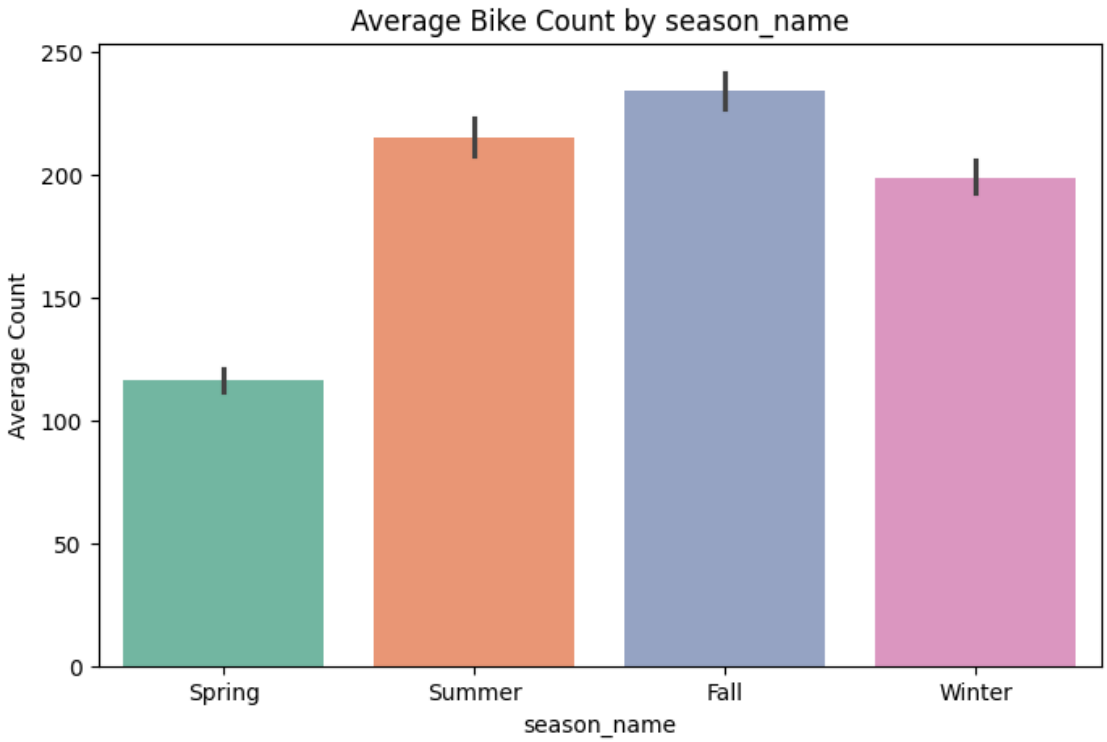
[40]:
```
plt.figure(figsize=(8,5))
sns.boxplot(x='season_name', y='count', data=df, palette='Set2')
plt.title('Boxplot of season_name vs Count')
plt.xlabel('season_name')
plt.ylabel('Bike Count')
plt.show()
```

```
<ipython-input-40-1e23531c41d5>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='season_name', y='count', data=df, palette='Set2')
```



[41]:
```
plt.figure(figsize=(8,5))
sns.barplot(x='season_name', y='count', data=df, estimator='mean',␣
 ↪palette='Set2')
plt.title('Average Bike Count by season_name')
plt.xlabel('season_name')
plt.ylabel('Average Count')
plt.show()
```

```
<ipython-input-41-29366039a5f9>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='season_name', y='count', data=df, estimator='mean',
palette='Set2')
```

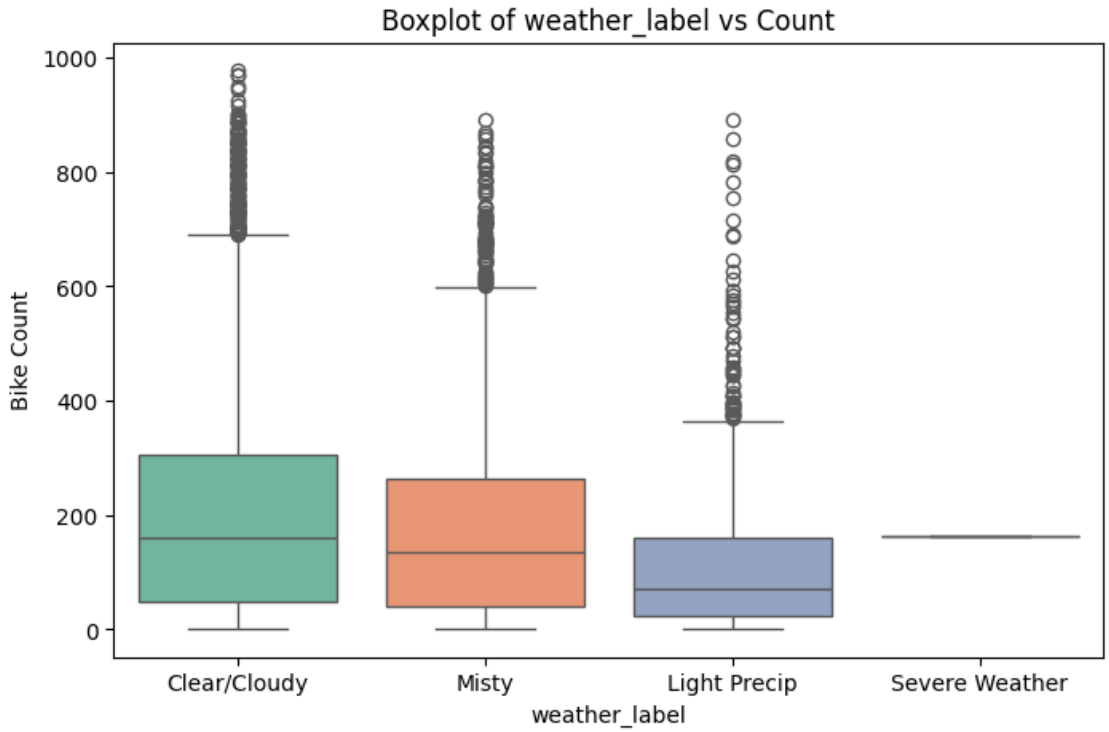


9 Observation:

1.Bike usage peaks during the Summer and Fall seasons, while it is relatively lower during Winter, suggesting seasonal influence on biking patterns.

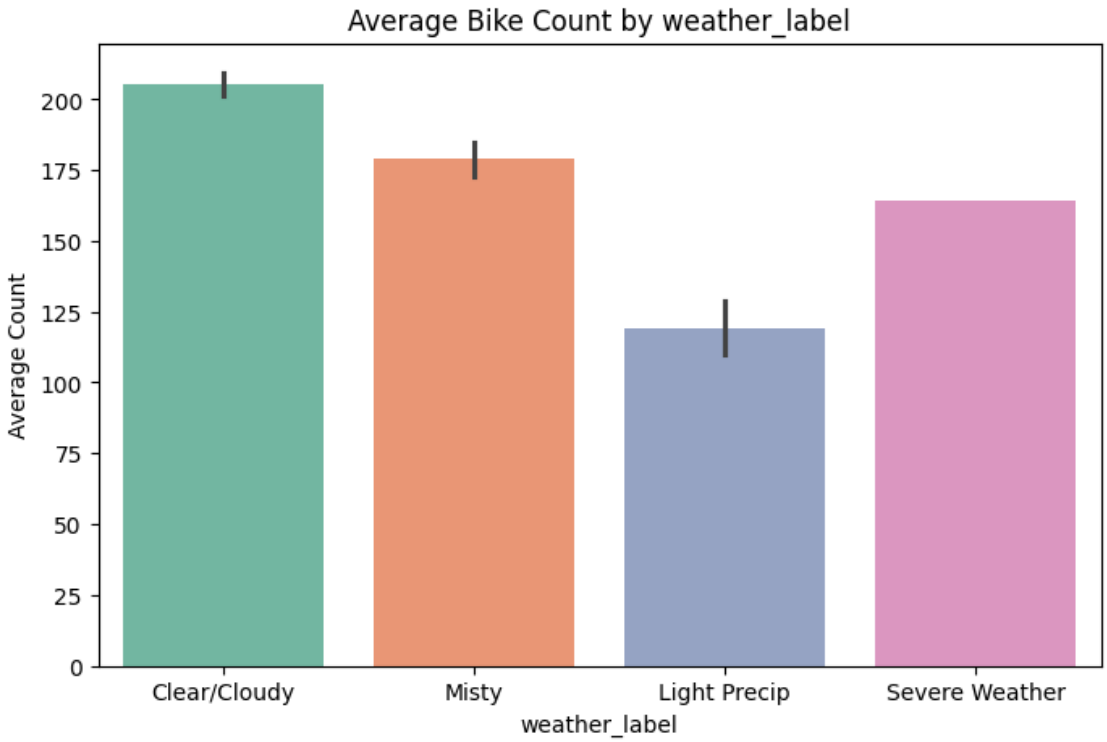
[43]:
```
plt.figure(figsize=(8,5))
sns.boxplot(x='weather_label', y='count', data=df, palette='Set2')
plt.title('Boxplot of weather_label vs Count')
plt.xlabel('weather_label')
plt.ylabel('Bike Count')
plt.show()
```

```
<ipython-input-43-e8b5f2bdd616>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='weather_label', y='count', data=df, palette='Set2')
```



[44]:
```
plt.figure(figsize=(8,5))
sns.barplot(x='weather_label', y='count', data=df, estimator='mean',␣
 ↪palette='Set2')
plt.title('Average Bike Count by weather_label')
plt.xlabel('weather_label')
plt.ylabel('Average Count')
plt.show()
```

```
<ipython-input-44-2315874d15c5>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='weather_label', y='count', data=df, estimator='mean',
palette='Set2')
```



[46]:
```
df[df['weather_label'] == 'Severe Weather']
```

[46]:

	holiday	workingday	temp	atemp	humidity	windspeed	casual	registered	season_name	weather_label	hour	day	monthname	year	count
5631	0	1	8.2	11.365	86	6.0032	6	158	Spring	Severe Weather	18	9	Jan	2012	164

10 Observation

1.Most bike rentals are during clear/Cloudy weather, Giving significant drop in Misty and rainy season. Showing that badweather negatively impacts bike users due to safety reasons.

2.We can aviod Server weather point from graph,As there is only single point for the same which may be misleading. We would be deleting the same further as it would create complexity in further hypothesis testing results.

[47]:
```
df = df[df['weather_label'] != 'Severe Weather']
```

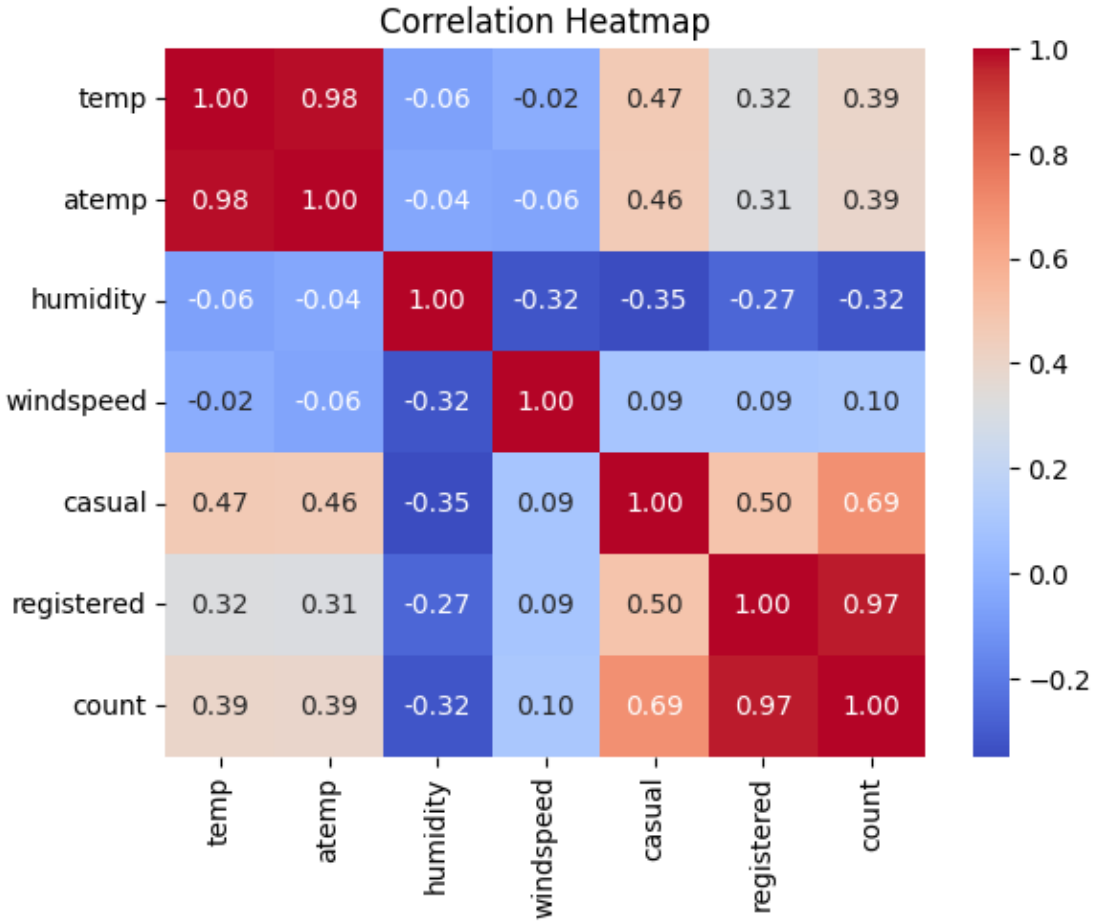
[48]:
```
df['weather_label'].value_counts()
```

[48]:
```
weather_label
Clear/Cloudy    7192
```

```
Misty           2834
Light Precip     859
Name: count, dtype: int64
```

[ ]:

[49]:
```
corr_matrix = df[['temp', 'atemp', 'humidity', 'windspeed', 'casual',␣
 ↪'registered', 'count']].corr()

sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



11 Observation

1.As we know the Final 'Count' is addition of Casual and registered so it is showing highly positive corelation.

2.Temp and Atemp feature shows are very high correlation.We can think to remove 1 of the feature from this as these are highly correlated and will give exactly same input the ML models.(Although Tree based models does not face Multi collinearity problem).

[ ]:

[50]:
```
df.head()
```

[50]:

	holiday	workingday	temp	atemp	humidity	windspeed	casual	registered	\
0	0	0	9.84	14.395	81	0.0	3	13	
1	0	0	9.02	13.635	80	0.0	8	32	
2	0	0	9.02	13.635	80	0.0	5	27	
3	0	0	9.84	14.395	75	0.0	3	10	
4	0	0	9.84	14.395	75	0.0	0	1	

	season_name	weather_label	hour	day	monthname	year	count
0	Spring	Clear/Cloudy	0	1	Jan	2011	16
1	Spring	Clear/Cloudy	1	1	Jan	2011	40
2	Spring	Clear/Cloudy	2	1	Jan	2011	32
3	Spring	Clear/Cloudy	3	1	Jan	2011	13
4	Spring	Clear/Cloudy	4	1	Jan	2011	1

[ ]:

[ ]:

12 Begin with Hypothesis Testings

[ ]:

13 Scenarrio 1 :

We will apply 2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented.

We know that T-test is used when you want to compare the means of two groups. and since we are dealing with Numeric (bike rentals) vs Categorical (Workingday (0 = No, 1 = Yes)). Perfect case for Independent 2 sample T-test.

[ ]:

Null Hypothesis (H0): The mean number of bike rentals on working days is equal to the mean number of bike rentals on non-working days.

H0 : M1 = M2

Alternate Hypothesis (H1): The mean number of bike rentals on working days is not equal to the mean number of bike rentals on non-working days.

H1 : M1 != M2

Assuming Significane level to be of 0.05

[53]:
```
from scipy.stats import ttest_ind
```

[56]:
```
workingday_yes = df[df['workingday'] == 1]['count']
workingday_no = df[df['workingday'] == 0]['count']
```

[57]:
```
tstat, p_value = ttest_ind(workingday_yes,workingday_no)
print('tstats',tstat)
print('p_value',p_value)
```

```
tstats 1.2105985511265596
p_value 0.22607559007082925
```

[58]:
```
if p_value < 0.05:
  print('Reject Null Hypothesis')
  print('Workingday has a significant effect on count.')
else:
  print('Fail to Reject Null Hypothesis')
  print('Workingday has no significant effect on count.')
```

```
Fail to Reject Null Hypothesis
Workingday has no significant effect on count.
```

14 Observation:

So since P value is greater than Significance level.We would Fail to Reject Null Hypothesis. Stating that Workingday has no significant effect on count.

[ ]:
```

```

15 Scenario 2

We will apply 2 sample ANNOVA test to check if No. of cycles rented is similar or different in different 1. weather 2. season

We would be testing 3 things here

Main Effect 1 (weather):

H0 : Mean number of bikes rented is same across all weathers

H1 : There is significane difference between mean number of bikes rented between atleast 1 weather

---

Main Effect 2 (season):

H0 : Mean number of bikes rented is same across all seasons

H1 : There is significane difference between mean number of bikes rented between atleast 1 season

---

Interaction Effect (Weather × Season):

H0 : There is no interaction effect between weather and season on bike rentals.

H1 : There is interaction between weather and season affecting bike rentals.

[60]:
```
pd.set_option('display.width', 200)          # widen the display
pd.set_option('display.max_columns', None)
```

[61]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols


model = ols('count ~ C(weather_label) + C(season_name) + C(weather_label):
 ↪C(season_name)', data=df).fit()

anova_table = sm.stats.anova_lm(model, typ=2)

print(anova_table)
```

```
                                    sum_sq       df           F
PR(>F)
C(weather_label)                 6.022043e+06      2.0   99.604322
1.361093e-43
C(season_name)                   2.158708e+07      3.0  238.032851
1.350921e-149
C(weather_label):C(season_name)  5.588352e+05      6.0    3.081036
5.150817e-03
Residual                         3.286889e+08  10873.0         NaN
NaN
```

Because all 3 p values are less than 0.05 we will Reject All 3 Null Hypothesis (H0).

And we can conclude :

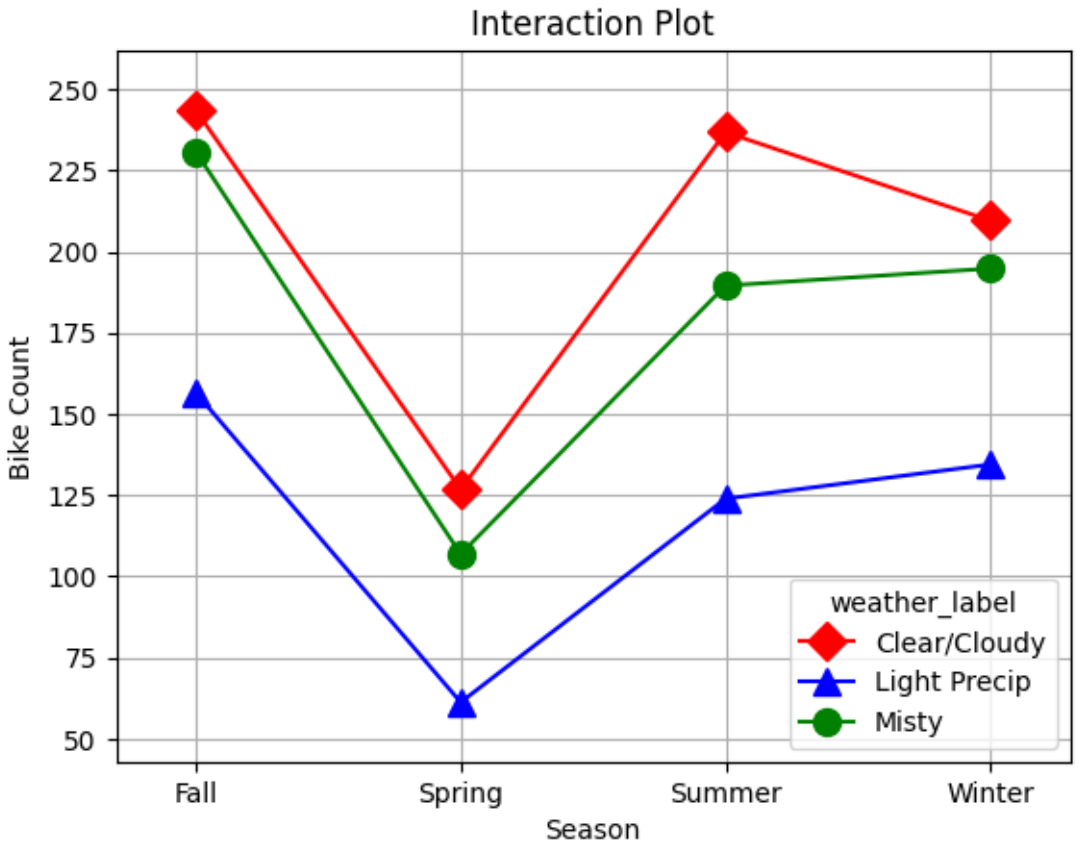
1.Both Weather and Season individually have significant effect on bike rentals

2.But the interaction of Weather and Season also have significant effect on bike rentals.

[ ]:

[ ]:

[62]:
```
#Help was taken from ChatGPT for below code
from statsmodels.graphics.factorplots import interaction_plot

interaction_plot(
    df['season_name'],
    df['weather_label'],
    df['count'],
    colors=['red','blue','green'],
    markers=['D','^','o'],
    ms=10
)
plt.title('Interaction Plot')
plt.xlabel('Season')
plt.ylabel('Bike Count')
plt.grid(True)
plt.show()
```



16 Observations

1.Across all seasons, Clear/Cloudy weather shows the highest bike rentals

2.Since the lines are not parallel it shows interaction exists(weather effect is changing across seasons.

[ ]:

17 Scenario 3

We would use Chi-square test to check if Weather is dependent on the season

[ ]:

Null Hypothesis (H0): Weathers and Seasons independet

Alternate Hypothesis (Ha) : Weather and Seasons are dependent on each other.

We would run the test at 5% Significane Level

[63]:
```
from scipy.stats import chi2_contingency
```

[64]:
```
contingency_table = pd.crosstab(df['season_name'], df['weather_label'])
print(contingency_table)
```

```
weather_label  Clear/Cloudy  Light Precip  Misty
season_name
Fall                   1930           199    604
Spring                 1759           211    715
Summer                 1801           224    708
Winter                 1702           225    807
```

[65]:
```
chi2_stat, p_val, dof, expected = chi2_contingency(contingency_table)
print(p_val)
```

```
2.8260014509929343e-08
```

[66]:
```
if p_val < 0.05:
  print('Reject Null Hypothesis')
  print('Weather and Seasons are dependent on each other.')
else:
  print('Fail to Reject Null Hypothesis')
  print('Weather and Seasons are independent.')
```

```
Reject Null Hypothesis
Weather and Seasons are dependent on each other.
```

18 Observation :

Since P value is less than Significane level we would Reject Null Hypothesis and State that Weather and Season fields are dependent on each other.

[ ]:

19 Final Observation:

1.Distribution of Data:

i.Most numerical features (like temperature, humidity, etc.) are slightly skewed but within a reasonable range.

ii.“Casual” and “Registered” users have a few outliers, but they seem genuine based on the business context.

2.T-test

So since P value is greater than Significance level.We would Fail to Reject Null Hypothesis. Stating that Workingday has no significant effect on count.

3.Interaction Effect

A Two-Way ANOVA test showed: There is also an interaction between season and weather on Bike rental counts.

4.A Chi-Square test showed that weather conditions depend on seasons.

[ ]:

20 Recommendations :

1.Plan offers and advertising mostly on Clear/Cloudy days because rentals are highest then.

2.Expect low rentals in spring, Can perform Campaings or special discounts here to attract customers.

3.During Fall and Summer when rentals are high, Make sure to keep high availablity of well maintained bikes for better customer experienced.

4.Can provide Specific office packages for working peoples in working days(Kind of memberships).

[ ]: