YULU HYPOTHESIS TESTING

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

from scipy.stats import ttest_ind,ttest_rel,norm,f_oneway

from scipy.stats import chi2_contingency

from sklearn.impute import SimpleImputer

from scipy import stats
import scipy
from scipy.stats import pearsonr,spearmanr
```

Problem statement

To identify, analyze, and understand the key factors affecting the demand for Yulu's shared electric cycles in the Indian market, in order to develop actionable strategies that can help reverse declining revenues and improve overall service adoption and customer retention.

```
data=pd.read_csv('/content/bike_sharing.txt')
data.head()
```

	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

Next steps: Generate code with data View recommended plots New interactive sheet

```
data.shape
```

```
(10886, 12)
```

```
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
```

We have to convert some variables into category

```
data['season']=data['season'].astype('category')
data['holiday']=data['holiday'].astype('category')
data['workingday']=data['workingday'].astype('category')
data['weather']=data['weather'].astype('category')
```

```
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  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), float64(3), int64(4), object(1)
memory usage: 723.7+ KB
```

missing values detection

```
data.isnull().sum()
```

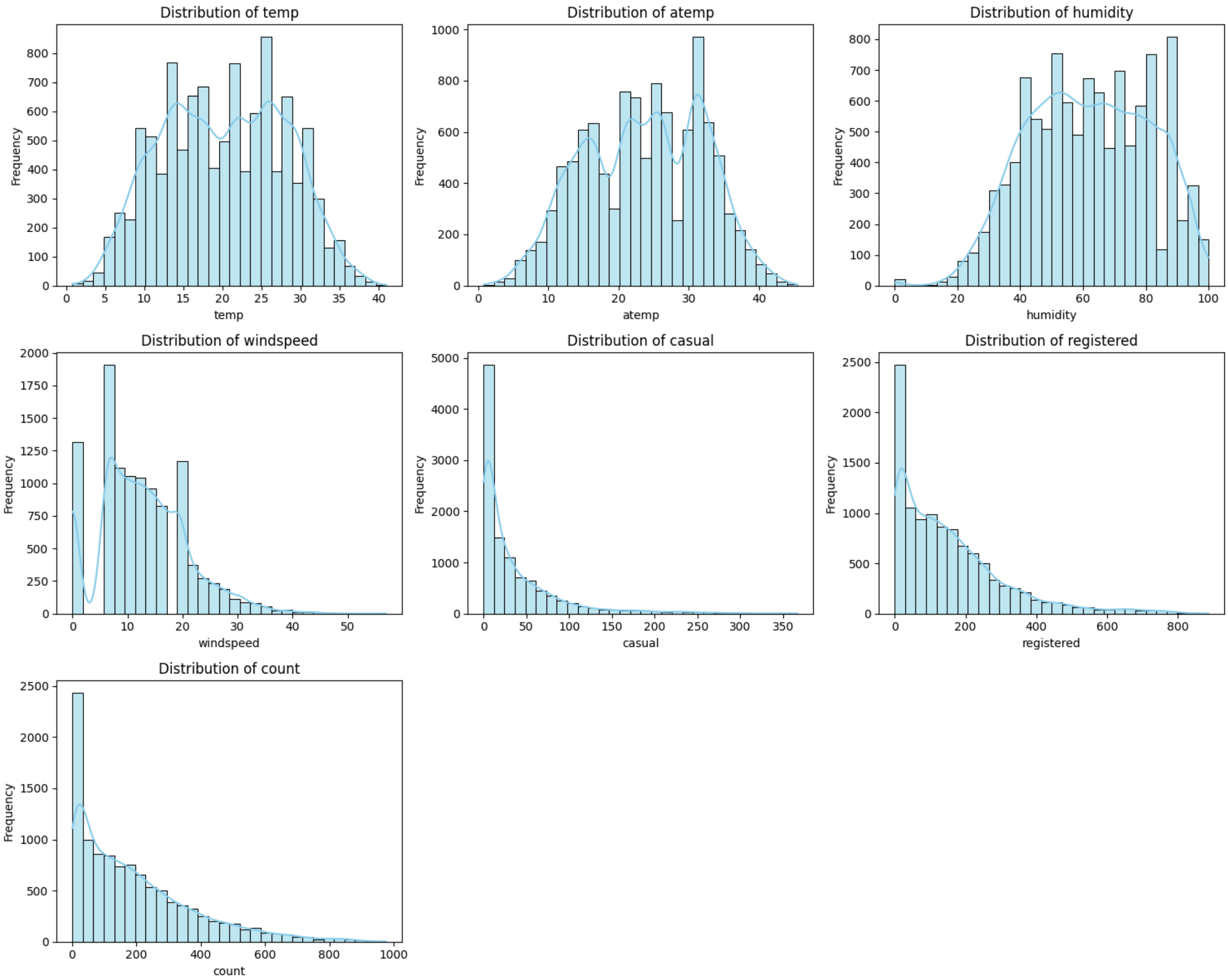
	0
datetime	0
season	0
holiday	0
workingday	0
weather	0
temp	0
atemp	0
humidity	0
windspeed	0
casual	0
registered	0
count	0

dtype: int64

so there are no missing values

Distribution plots for all the continuos variables

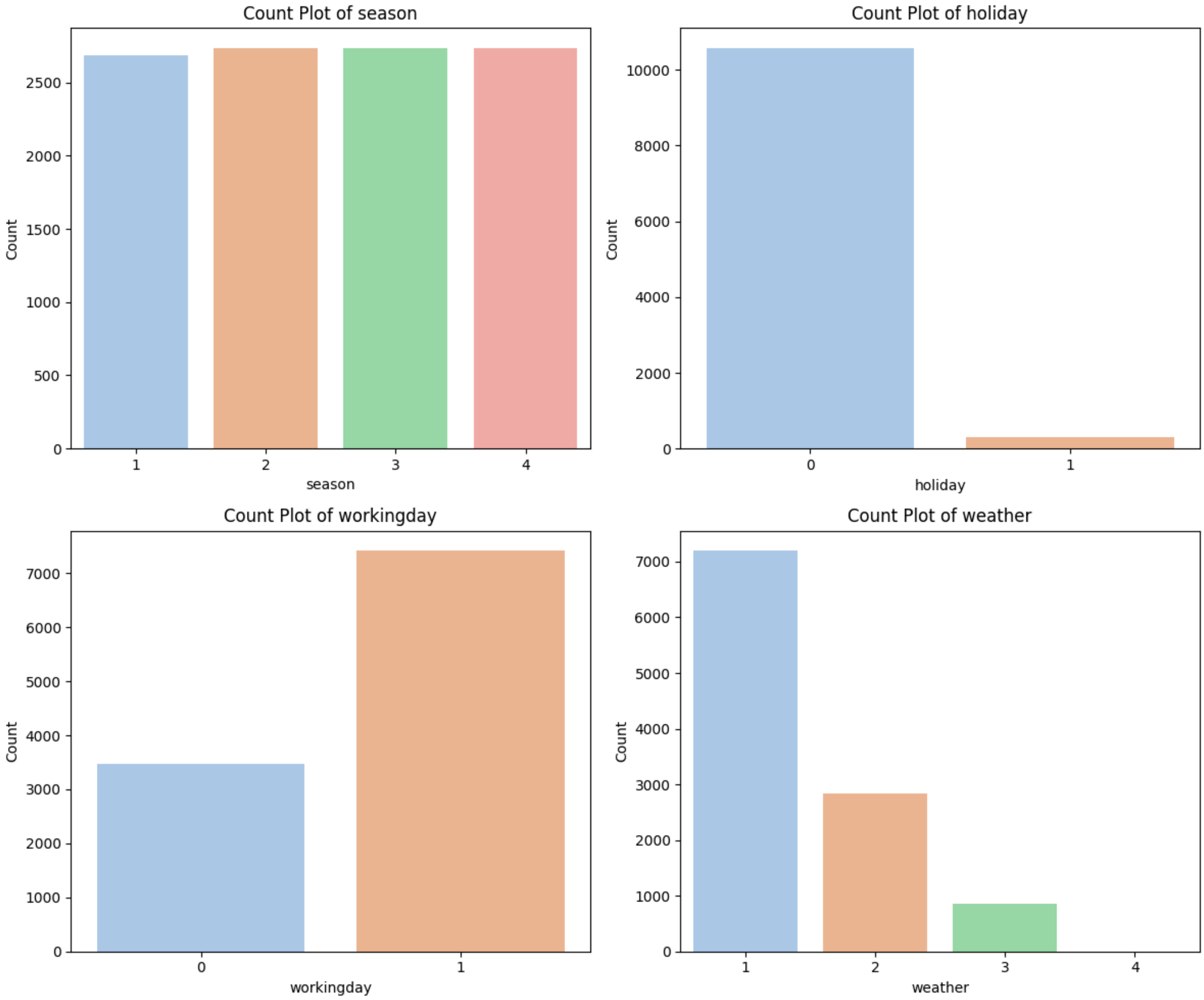
```
continuous_columns = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']

plt.figure(figsize=(15, 12))

for i, col in enumerate(continuous_columns, 1):
    plt.subplot(3, 3, i)
    sns.histplot(data[col], kde=True, bins=30, color='skyblue')
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



BAR PLOTS FOR ALL THE CATEGORICAL VARIABLES

```
categorical_columns = ['season', 'holiday', 'workingday', 'weather']
plt.figure(figsize=(12, 10))

for i, col in enumerate(categorical_columns, 1):
    plt.subplot(2, 2, i)
    sns.countplot(data=data, x=col, palette='pastel')
    plt.title(f'Count Plot of {col}')
    plt.xlabel(col)
    plt.ylabel('Count')

plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-9-3009635122.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(data=data, x=col, palette='pastel')
/tmp/ipython-input-9-3009635122.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(data=data, x=col, palette='pastel')
/tmp/ipython-input-9-3009635122.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(data=data, x=col, palette='pastel')
/tmp/ipython-input-9-3009635122.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(data=data, x=col, palette='pastel')
```



1] season: season (1: spring, 2: summer, 3: fall, 4: winter)

2] workingday: if day is neither weekend nor holiday is 1, otherwise is 0.

3] 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

Conlusion

Use of rental bikes is almost same in all the seasons.People use more number of rental bikes on working days rather than holidays.Usage of rental bikes is more when the weather is clear.

Two-Sample T-Test (Working Day vs Non-Working Day)

H0 (Null): Mean number of bikes rented is the same on working and non-working days.

H1 (Alt): Mean number of bikes rented is different on working vs non-working days

```
working = data[data['workingday'] == 1]['count']
non_working = data[data['workingday'] == 0]['count']

# Perform t-test
t_stat, p_val =ttest_ind(working, non_working)

print("T-Statistic:", t_stat)
print("P-Value:", p_val)
```

```
T-Statistic: 1.2096277376026694
P-Value: 0.22644804226361348
```

If $p < 0.05$, reject H0

If $p \geq 0.05$, fail to reject H0

Here, $p > 0.05$,so Mean number of bikes rented is the same on working and non-working days

ANOVA: Number of Rentals by Weather and Season

H0: All group means are equal.

H1: At least one group has a different mean.

```
#for weather
weather_groups = [group['count'].values for name, group in data.groupby('weather')]
f_stat_w, p_val_w =f_oneway(*weather_groups)

print("Weather ANOVA - F-statistic:", f_stat_w)
print("P-Value:", p_val_w)
```

```
Weather ANOVA - F-statistic: 65.53024112793271
P-Value: 5.482069475935669e-42
/tmp/ipython-input-18-1805890674.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a f
  weather_groups = [group['count'].values for name, group in data.groupby('weather')]
```

Here $p<0.05$,so we reject Ho,There is a diffrence in use of bike in each weather

```
#for season
season_groups = [group['count'].values for name, group in data.groupby('season')]
f_stat_s, p_val_s = stats.f_oneway(*season_groups)

print("Season ANOVA - F-statistic:", f_stat_s)
print("P-Value:", p_val_s)
```

```
Season ANOVA - F-statistic: 236.94671081032106
P-Value: 6.164843386499654e-149
/tmp/ipython-input-19-2323593191.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a f
  season_groups = [group['count'].values for name, group in data.groupby('season')]
```

Here $p<0.05$,so we reject Ho,There is a diffrence in use of bikes in each season

Chi-Square Test: Is Weather Dependent on Season?

H0: Weather is independent of season.

H1: Weather depends on season.

```
# Create contingency table
contingency = pd.crosstab(data['season'], data['weather'])

# Chi-square test
chi2, p_val_chi, dof, expected = stats.chi2_contingency(contingency)

print("Chi-Square Statistic:", chi2)
print("P-Value:", p_val_chi)
```

```
Chi-Square Statistic: 49.158655596893624
P-Value: 1.549925073686492e-07
```

Double-click (or enter) to edit

Here p<0.05,so we reject Ho,It tells us that weather is dependent on season

Recommendations

1] Allocate more bikes during weekdays, especially in spring/summer with clear weather.

2] Prepare for lower rentals during bad weather; consider protective gear or incentives.

3] Promote bike rentals more aggressively during favorable seasons to capitalize on natural demand.

4] Increase bike availability during weekdays, especially during commute hours.

5] Reduce fleet size or shift location focus during weekends and holidays.

6] Offer discounts or incentives on bad weather days to maintain usage.

7] Adjust pricing and promotions seasonally, with higher demand in spring and summer.

8] Deploy more bikes in recreational areas during weekends and tourist seasons.

9] Use weather forecasts to predict demand drops and adjust operations accordingly.

10] Implement targeted marketing for commuters, weekend riders, and casual users.

11] Optimize docking station placement based on seasonal and weather patterns.

12] Reduce operational costs by scaling down in low-demand weather or seasons.

13] Enhance user experience with real-time alerts and rewards for consistent use.