YULU Bike Sharing DATASET Day 5 Task

Name: Nikhil Kumar Nigam

Batch: DATA ANALYST INTERN AT ELEVATE LABS

Email: nikhilnigam@engineer.com

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

```
df=pd.read_csv('/content/Yulu Bike Sharing Dataset.txt')
df.head(5)
```

	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 df View recommended plots New interactive sheet

```
df.shape
```

```
(10886, 12)
```

No of Rows = 10886

No of Coloumn = 12

```
round(df.describe(),2)
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.00	10886.00	10886.00	10886.00	10886.00	10886.00	10886.00	10886.00	10886.00	10886.00	10886.00
mean	2.51	0.03	0.68	1.42	20.23	23.66	61.89	12.80	36.02	155.55	191.57
std	1.12	0.17	0.47	0.63	7.79	8.47	19.25	8.16	49.96	151.04	181.14
min	1.00	0.00	0.00	1.00	0.82	0.76	0.00	0.00	0.00	0.00	1.00
25%	2.00	0.00	0.00	1.00	13.94	16.66	47.00	7.00	4.00	36.00	42.00
50%	3.00	0.00	1.00	1.00	20.50	24.24	62.00	13.00	17.00	118.00	145.00
75%	4.00	0.00	1.00	2.00	26.24	31.06	77.00	17.00	49.00	222.00	284.00
max	4.00	1.00	1.00	4.00	41.00	45.46	100.00	57.00	367.00	886.00	977.00

```
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
```

Data does not have the null values

Categorical Variables are listed below:

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

holiday: (0 or 1)

workingday: (0 or 1)

weather: (1, 2, 3 or 4)

Numerical Variables are temp, atemp, humidity, windspeed, casual, registered, count

```
df['datetime'] =pd.to_datetime(df['datetime'])
#converting Datetime (object) into datetime datatype
```

```
duplicates = df.duplicated()
count_duplicates = duplicates.sum()
print(f'Count of Duplicates in the Yulu Data = {count_duplicates}')
```

```
Count of Duplicates in the Yulu Data = 0
```

No duplicates rows in the data

```
df.head(5)
```

	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 df View recommended plots New interactive sheet

```
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
```

Plotting Distribution of Numerical columns

```
df
```

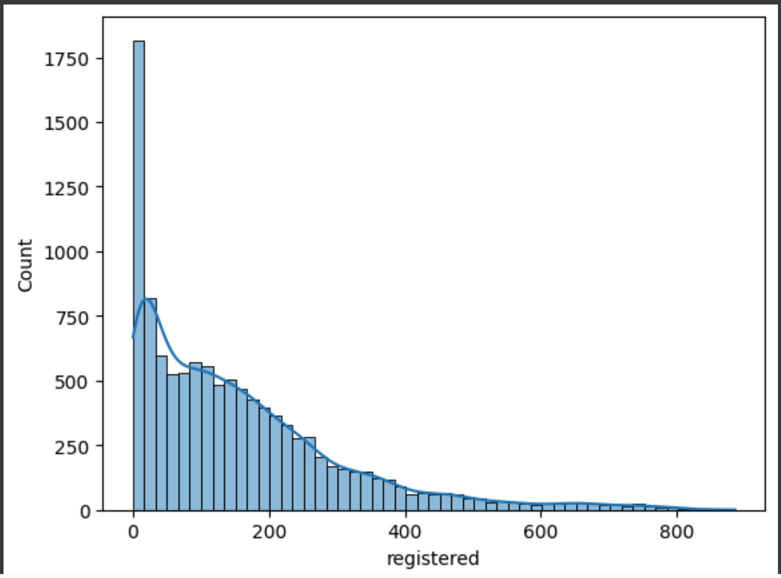
	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

10886 rows × 12 columns

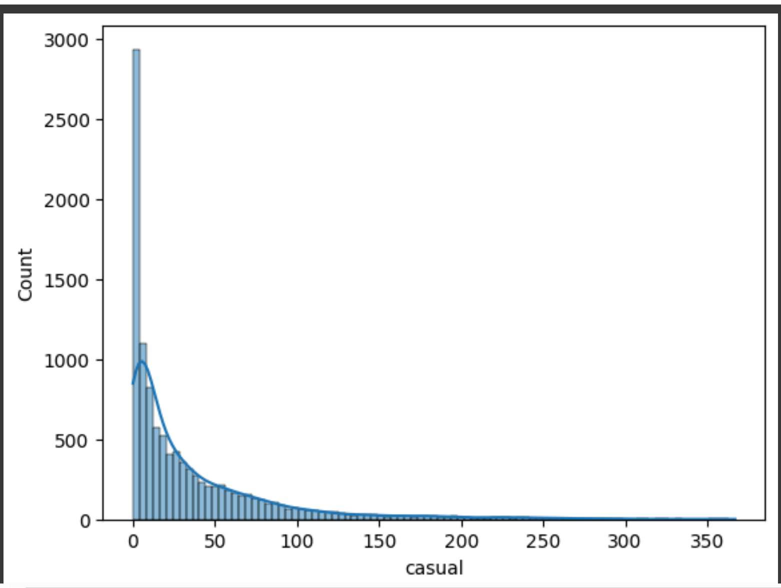
Next steps: Generate code with df | View recommended plots | New interactive sheet

Double-click (or enter) to edit

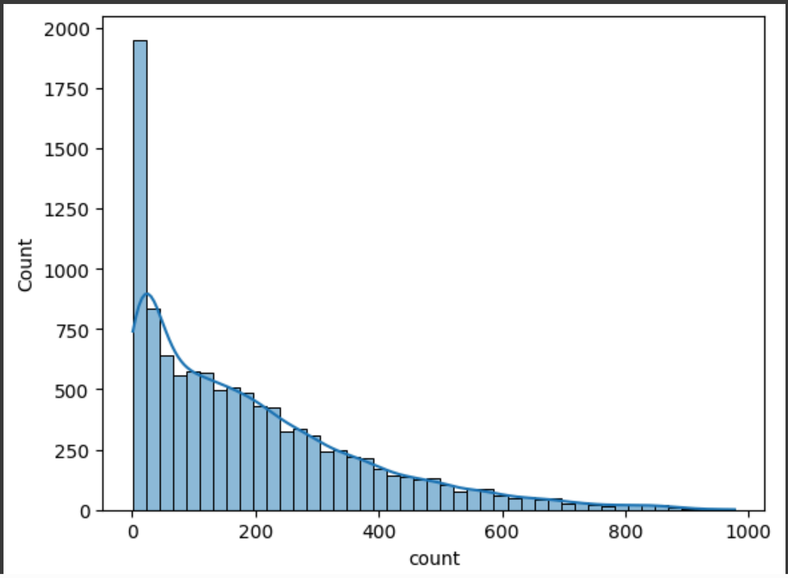
```
sns.histplot(df.registered   , kde = True)
plt.show()
```



```
sns.histplot(df.casual   , kde = True)
plt.show()
```



```
sns.histplot(df['count']     , kde = True)
plt.show()
```



Conclusions from the Histogram

1. **Most values are small (close to 0):**

   a. The highest bar is at the very low end (close to 0 registered users).

   b. This means most of the time, very few users were registered in the time period i.e. per hour.

2. **Distribution is right-skewed:**

   a. The histogram has a long tail towards the right (showing higher registered users).

   b. Right-skewed means that while most registration counts are low, some rare instances have very high registration numbers (outliers).

   c. There are small bars even near 600–800, meaning there are some rare periods where a very high number of users registered.

3. **Gradual decline as registration increases:**

   a. As the number of registered users increases (100, 200, 300, etc.), the count of such events decreases sharply.

   b. High registration numbers are rare events.

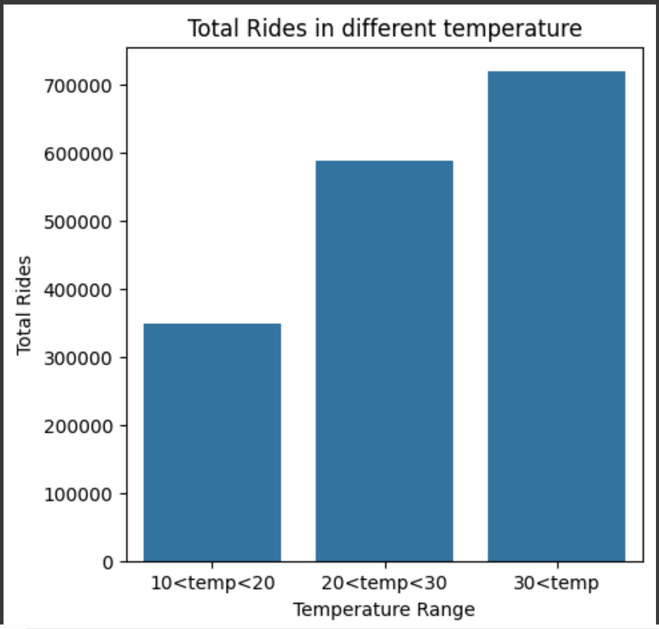
Summary: The registered user counts are mostly low, the data is right-skewed, and occasional spikes of very high registrations exist but are rare.

```
result = { 'Temperature_Range': ['10<temp<20', '20<temp<30', '30<temp'],
           'Total_Registered_Users': [ df[(df['atemp'] > 10) & (df['atemp'] < 20)]['registered'].sum(),
                                       df[(df['atemp'] > 20) & (df['atemp'] < 30)]['registered'].sum(),
                                       df[(df['atemp'] > 30)]['registered'].sum()]}
```

```
dd = pd.DataFrame(result)
print(dd)
```

```
  Temperature_Range  Total_Registered_Users
0         10<temp<20                  348823
1         20<temp<30                  587884
2            30<temp                  719620
```

```
plt.figure(figsize=(5, 5))
sns.barplot(x='Temperature_Range', y='Total_Registered_Users', data=dd)
plt.title('Total Rides in different temperature')
plt.xlabel('Temperature Range')
plt.ylabel('Total Rides')
plt.show()
```



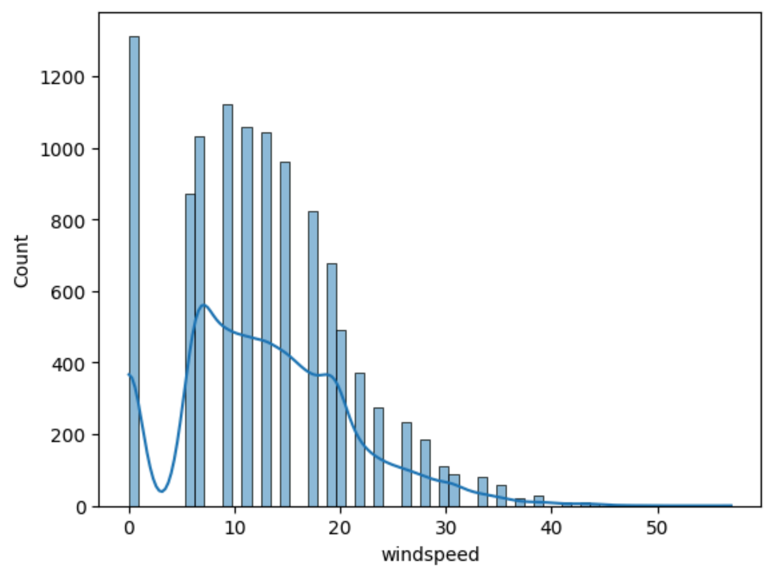
Conclusion:

```
1.  People prefer to ride more when it is warm to hot (above 20°C, especially above 30°C).

2. Colder weather discourages people from taking rides.
```

Business/Operational Suggestions for Yulu:

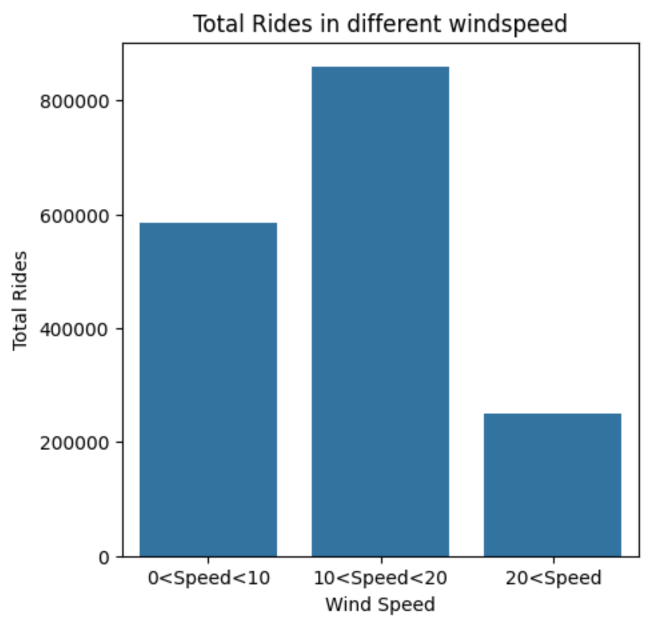
```
1. Offer special promotions, discount codes, or cashback offers during colder months to encourage more rides.
2. Launch winter-specific discounted monthly passes to lock in users during low-demand periods.
```

```
# windspeed Histogram plot to see the variation of windspeed in the region
sns.histplot(df.windspeed    , kde = True)
plt.show()
```



```
#plotting Effect on User numbers based on wind speed
print('--------------------------------------')
result_windspeed = { 'Wind_Speed': ['0<Speed<10', '10<Speed<20', '20<Speed'],
           'Total_Registered_Users': [ df[(df['windspeed'] >= 0) & (df['windspeed'] < 10)]['registered'].sum(),
                                       df[(df['windspeed'] >= 10) & (df['windspeed'] < 20)]['registered'].sum(),
                                       df[(df['windspeed'] >= 20)]['registered'].sum()]}

dw = pd.DataFrame(result_windspeed)
print(dw)
print('--------------------------------------')

print()
print()
plt.figure(figsize=(5, 5))
sns.barplot(x='Wind_Speed', y='Total_Registered_Users', data=dw)
plt.title('Total Rides in different windspeed')
plt.xlabel('Wind Speed')
plt.ylabel('Total Rides')
plt.show()
```

```
--------------------------------------
    Wind_Speed  Total_Registered_Users
0   0<Speed<10                  584708
1  10<Speed<20                  858827
2     20<Speed                  249806
--------------------------------------
```



Summary:

```
1. Mild to moderate wind is good for rides.
2. Very high winds discourage users significantly from taking rides.
```

Business/Operational Suggestions for Yulu:

```
1. Monitor weather forecasts — if strong winds are expected, expect a drop in rides.
2. Promote more rides when winds are mild (maybe send app notifications saying "Perfect weather to ride today!").
```

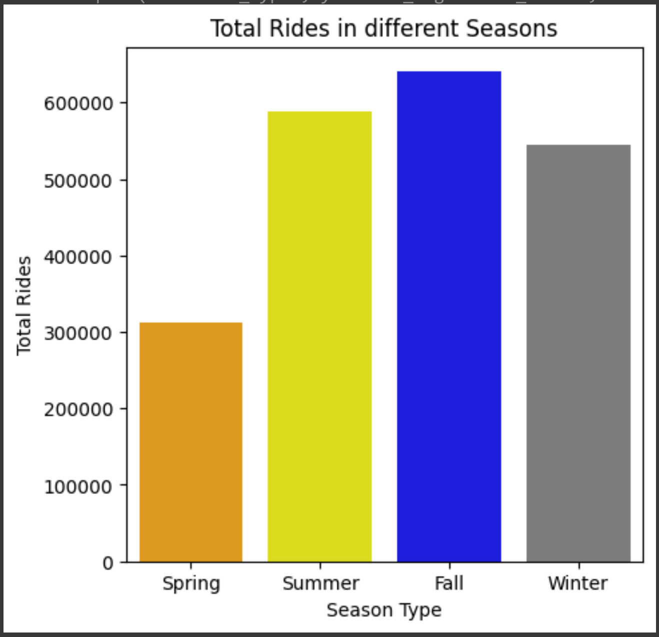
```
df.head()
```

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

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

```
#plotting Effect on User numbers based on Season
print('--------------------------------------')
result_season = { 'Season_Type': ['Spring', 'Summer', 'Fall', 'Winter'],
           'Total_Registered_Users': [ df[df['season'] == 1]['count'].sum(),
                                       df[df['season'] == 2]['count'].sum(),
                                       df[df['season'] == 3]['count'].sum(),
                                       df[df['season'] == 4]['count'].sum()]}
seasonal_ride_df = pd.DataFrame(result_season)
print(seasonal_ride_df)
print('--------------------------------------')

print()
print()
# Plotting the ride_df
plt.figure(figsize=(5, 5))
colors = ['orange', 'yellow', 'blue', 'grey']
sns.barplot(x='Season_Type', y='Total_Registered_Users', data=seasonal_ride_df, palette=colors)
plt.title('Total Rides in different Seasons')
plt.xlabel('Season Type')
plt.ylabel('Total Rides')
plt.show()
```

```
--------------------------------------
  Season_Type  Total_Registered_Users
0      Spring                  312498
1      Summer                  588282
2        Fall                  640662
3      Winter                  544034
--------------------------------------


<ipython-input-70-7691a6ea9893>:17: 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.barplot(x='Season_Type', y='Total_Registered_Users', data=seasonal_ride_df, palette=colors)
```



No Sharp variation is seen in the usage pattern with the changing seasons except Spring season.

Summary Insight:

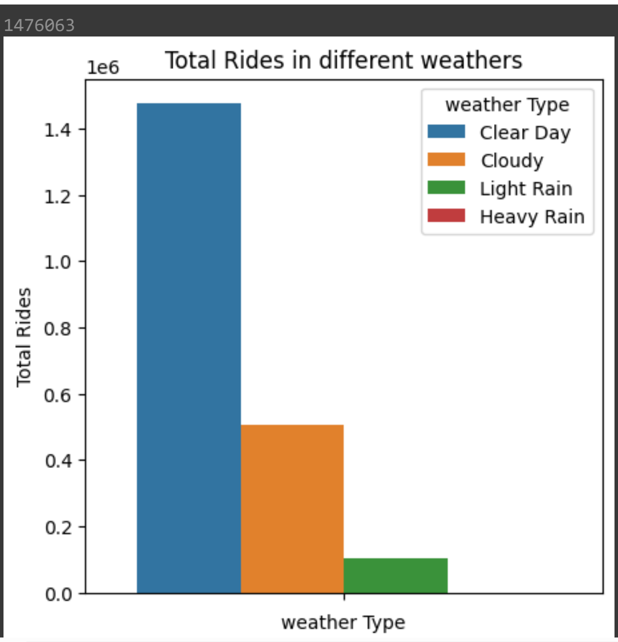
```
1. Mild to warm seasons (fall, summer) bring the highest engagement.
2. Spring and winter show reduced ridership — possibly due to unpredictable weather (spring rains, winter cold).
```

Business/Operational Suggestions for Yulu:

```
1. Encourage more rides in spring with limited-time offers, like "Ride 5 times this month, get 1 free!".
2. Implement dynamic pricing — lower prices during spring and winter to stimulate more demand.
3. Collaborate with festivals, marathons, public events that happen in spring/fall to drive rides.
4. Run a campaign around "Safe and Fun Riding in Spring Showers" to build confidence.
```

```
total_rides_weather1 = df[df['weather'] == 1]['count'].sum()
total_rides_weather2 = df[df['weather'] == 2]['count'].sum()
total_rides_weather3 = df[df['weather'] == 3]['count'].sum()
total_rides_weather4 = df[df['weather'] == 4]['count'].sum()

weather_ride_data = { 'weather Type': ['Clear Day', 'Cloudy', 'Light Rain', 'Heavy Rain'],
    'Total Rides': [total_rides_weather1, total_rides_weather2, total_rides_weather3, total_rides_weather4]}

weather_ride_df = pd.DataFrame(weather_ride_data)

# Plotting the ride_df
plt.figure(figsize=(5, 5))
sns.barplot(hue='weather Type', y='Total Rides', data=weather_ride_data)
plt.title('Total Rides in different weathers')
plt.xlabel('weather Type')
plt.ylabel('Total Rides')
plt.show()
```

```
1476063
```



Conclusion:

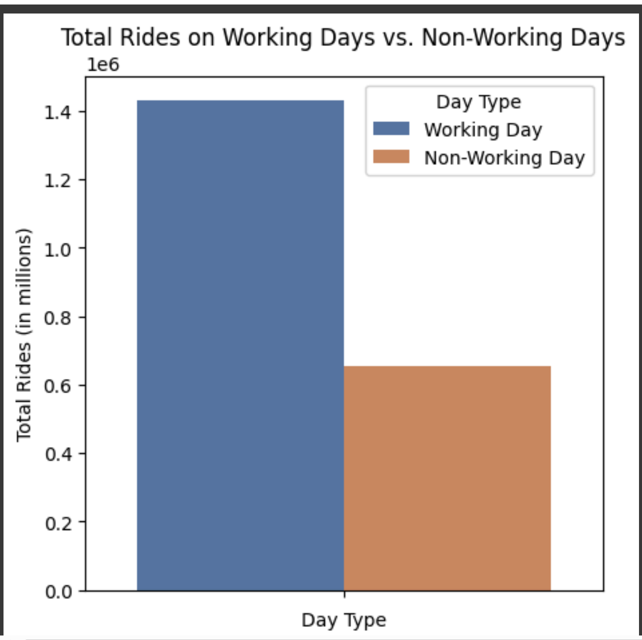
```
1. Clear Day weather has overwhelmingly the highest number of rides — way higher than Cloudy, Light Rain, or Heav
2. Cloudy days still maintain a decent number of rides.
3. Light Rain causes a sharp drop in total rides.
4. Heavy Rain almost results in zero rides — users avoid riding almost entirely.
```

Business Suggestions for Yulu:

```
1. Focus marketing and promotions on clear and slightly cloudy days:
2. Light Rain still has some riders. Push safety tips (raincoats, safe riding reminders) to encourage cautious us
3. Slight discounts during light rain to motivate some users without compromising safety.
4. Clearly mention ride-cancellation/refund policies for rain — improve trust.
5. Keep tracking real-time weather vs ride demand. Maybe in future, predict how many bikes should be active depen
```

The Graph shows the bike prefernce of the users is obviously declining in the raining days. People geenrally prefer Yulu's Bike on clear days.

```python
total_rides_workingday = df[df['workingday'] == 1]['count'].sum()
total_rides_non_workingday = df[df['workingday'] == 0]['count'].sum()

ride_data = { 'Day Type': ['Working Day', 'Non-Working Day'],
    'Total Rides': [total_rides_workingday, total_rides_non_workingday]}

ride_df = pd.DataFrame(ride_data)

# Plotting the ride_df
plt.figure(figsize=(5, 5))
sns.barplot(hue='Day Type', y='Total Rides', data=ride_df, palette = 'deep')
plt.title('Total Rides on Working Days vs. Non-Working Days')
plt.xlabel('Day Type')
plt.ylabel('Total Rides (in millions)')
plt.show()
```



Conclusion:

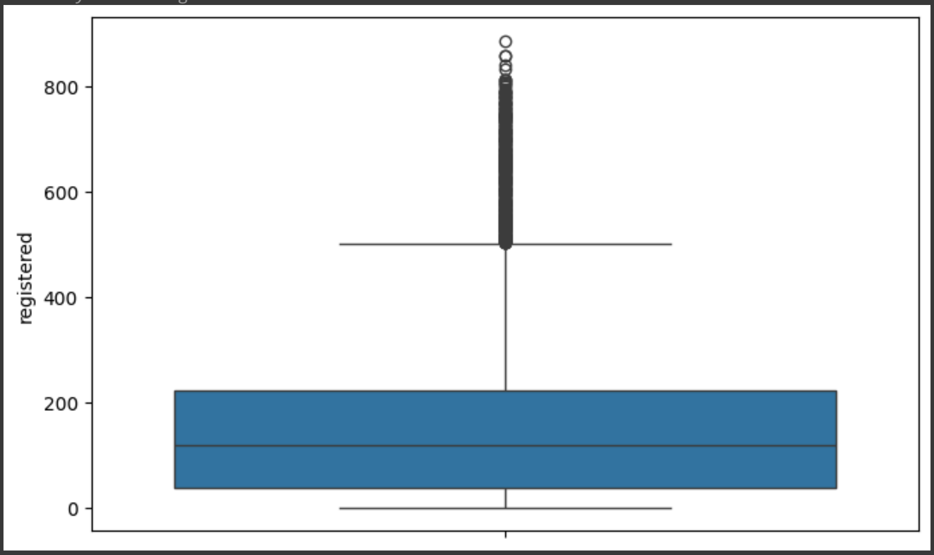
```
1. Working Days see a much higher number of rides (~14.5 Lakhs) compared to Non-Working Days (~6.5 Lakhs).
2. That means Yulu rides are heavily dependent on commuting needs (like office-goers, college students, etc.).
3. On Non-Working Days (weekends/holidays), ride volume drops by more than 50%.
```

Business Suggestions for Yulu:

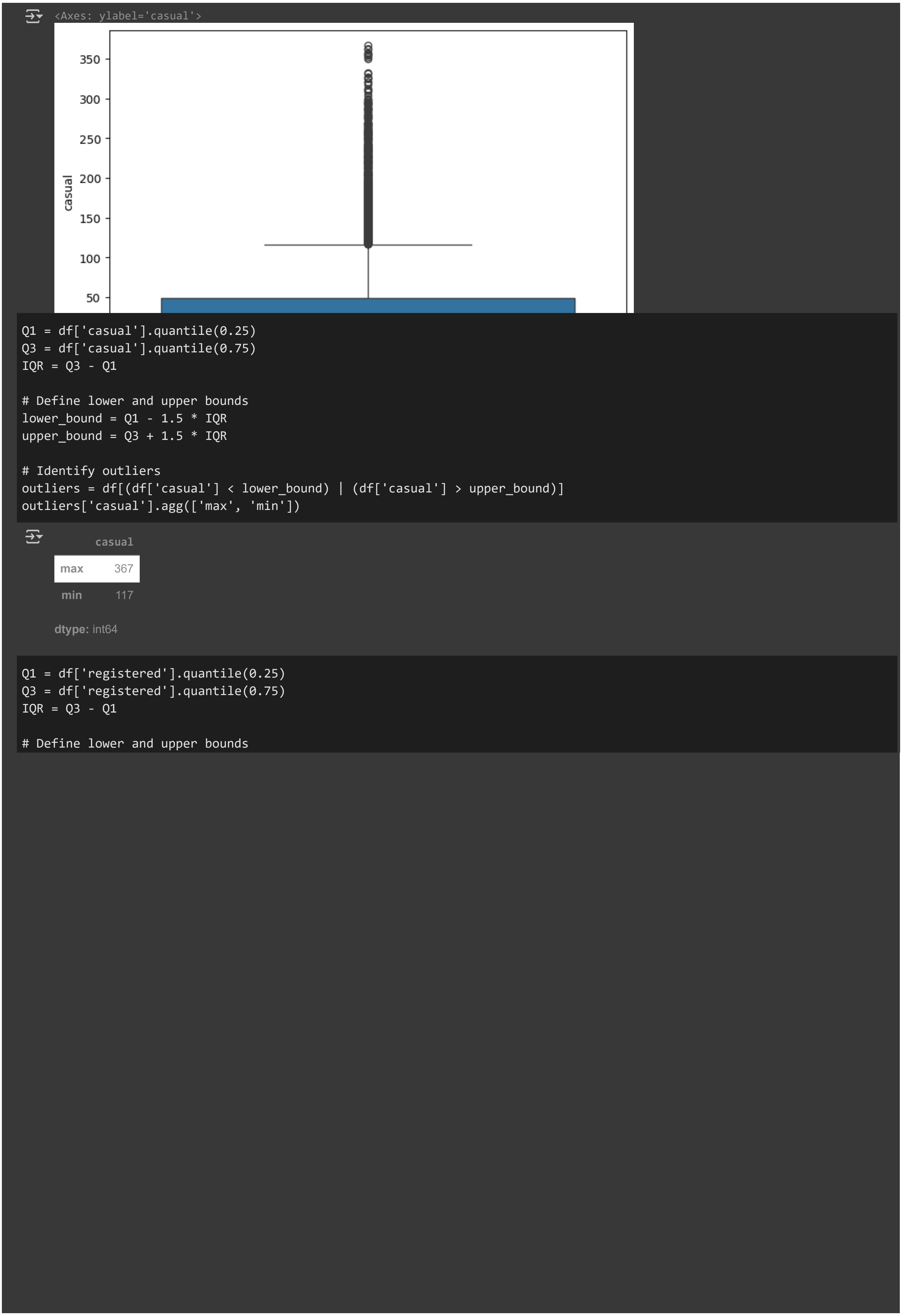
```
1. To boost weekend usage, launch offers for leisure rides, like: "Weekend Explorer Deals" – discount packages fo
2. Tie up with malls, parks, sports events during weekends to drive ridership when regular commute rides dip.
3. On weekends, people go to parks, shopping streets, picnic spots. Deploy more Yulu bikes in such areas.
```

```
# Boxplot of Casual and Registered Users
plt.figure(figsize=(8, 5))
sns.boxplot(df.registered)
```

```
<Axes: ylabel='registered'>
```



```
plt.figure(figsize=(8, 5))
sns.boxplot(df.casual)
```



```
<Axes: ylabel='casual'>
```

```
Q1 = df['casual'].quantile(0.25)
Q3 = df['casual'].quantile(0.75)
IQR = Q3 - Q1

# Define lower and upper bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = df[(df['casual'] < lower_bound) | (df['casual'] > upper_bound)]
outliers['casual'].agg(['max', 'min'])
```

```
     casual
max     367
min     117

dtype: int64
```

```
Q1 = df['registered'].quantile(0.25)
Q3 = df['registered'].quantile(0.75)
IQR = Q3 - Q1

# Define lower and upper bounds
```