YULU - BUSINESS CASE STUDY

About Yulu:

Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute. Starting off as a mission to eliminate traffic congestion in India, Yulu provides the safest commute solution through a user-friendly mobile app to enable shared, solo and sustainable commuting.

Yulu zones are located at all the appropriate locations (including metro stations, bus stands, office spaces, residential areas, corporate offices, etc) to make those first and last miles smooth, affordable, and convenient!

Business Problem to solve:

The business problem to solve is understanding what factors influence the demand for Yulu's shared electric cycles in India. With revenues declining, Yulu needs to identify which variables—like season, weather, and whether it's a working day—significantly impact the number of rentals. This insight will help them take targeted actions to improve demand and stabilize revenue.

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

Important Python Liberaries to Import | Reading Files | Basic Data Exploration

```
#IMPORTING LIBERARIES:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from scipy import stats
from scipy.stats import skew, kurtosis
import statsmodels.api as sm
from scipy.stats import norm
from scipy.stats import ttest_ind, shapiro, levene, f_oneway, chi2_contingency
import datetime

#READING THE FILE:
from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv('/content/drive/MyDrive/python files/yulu.csv')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
#EXPLORAING THE DATA FOR THE FIRST TIME:
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

Basic Observation on the Data

```
#DATA SHAPE:

```

```
df.shape
```

```
(10886, 12)
```

The dataset contains 10,886 rows and 12 columns.

This means you have 10,886 records of bike rentals, each described by 12 features (like date, season, weather, temperature, etc.).

This is a sufficiently large sample size, allowing reliable statistical analysis and hypothesis testing.

```
df.columns
```

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

```
# FINDING TOTAL NULL VALUES IN THE DATA:

df.isna().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

No Missing Values: All 10,886 records are complete across all columns—no nulls to impute.

```
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
```

- datetime is stored as an object (string) -we should convert this to datetime type for time-based analysis.
- season, holiday, workingday, and weather are categorical variables stored as integers—these will need to be treated as categorical during analysis and visualization.
- temp, atemp, humidity, and windspeed are numerical continuous variables.
- casual, registered, and count are numerical counts, with count being the main dependent variable you'll analyze.

1. CONVERTING THE NECESSARY COLUMNS IN THE REQUIRED DATA TYPES

2. CREATING ADDITIONAL COLUMNS FOR OUR UNDERSTANDING AND BETTER ANALYSIS OF THE DATASET

```
#CONVERTING THE DATETIME COLUMN IN THE DATETIME DATATYPE:
df['datetime'] = pd.to_datetime(df['datetime'])

#CREATING THE REQUIRED COLUMNS FOR THE BETTER UNDERSTANDING:
df['day'] = pd.to_datetime(df['datetime']).dt.day
df['month'] = pd.to_datetime(df['datetime']).dt.month
df['year'] = pd.to_datetime(df['datetime']).dt.year
df['dayofweek'] = pd.to_datetime(df['datetime']).dt.dayofweek
df['quarter'] = pd.to_datetime(df['datetime']).dt.quarter
df['hour'] = pd.to_datetime(df['datetime']).dt.hour
```

```
df
```

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

10886 rows × 18 columns

SOME POINTS TO UNDERSTAND:

DAY OF WEEK:

0: MONDAY

1: TUESDAY

2: WEDNESDAY

3: THURSDAY

4: FIRDAY

5: SATURDAY

6: SUNDAY

```
# FINDING STAISTICAL DATA FOR THE GIVEN NETFLIX DATA:

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

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	day	month
count	10886	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	2011-12-27 05:56:22.399411968	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132	9.992559	6.521495
min	2011-01-01 00:00:00	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.000000
25%	2011-07-02 07:15:00	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000	5.000000	4.000000
50%	2012-01-01 20:30:00	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000	10.000000	7.000000
75%	2012-07-01 12:45:00	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000	15.000000	10.000000
max	2012-12-19 23:00:00	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000	19.000000	12.000000
std	NaN	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454	5.476608	3.444373

KEY INSIGHTS:

- **Date Range:** Data spans January 2011 to December 2012, covering two full years, ideal for seasonal analysis.
- **Season Distribution:** Seasons are evenly represented (1–4), supporting seasonal comparisons.
- **Holiday & Working Day:**

  Holidays are rare (~2.8% of records).

  Working days dominate (~68% of records).
- **Weather Conditions:** Mostly clear or misty weather; few records of severe conditions.
- **Temperature:**

  Range: 0.82°C – 41°C (actual), 0.76°C – 45.45°C (perceived).

  Mean temperature ~20°C indicates moderate climate overall.
- **Humidity & Windspeed:**

  Humidity varies widely (0–100%).

  Windspeed ranges from 0 to ~57, which may impact ridership.
- **Demand (Count):**

  Mean rentals per hour: ~192.

  Minimum: 1 rental; Maximum: 977 rentals.

  High variability suggests influence of time, weather, and day type.
- **Time Variables:**

  hour (0–23) allows hourly demand analysis.

  dayofweek (0–6) enables weekday/weekend comparison.

```
# FINDINNG THE NUMBER OF UNIQUE VALUES IN ALL THE COLUMNS IN THE DATA:

df.nunique()
```

	0
datetime	10886
season	4
holiday	2
workingday	2
weather	4
temp	49
atemp	60
humidity	89
windspeed	28
casual	309
registered	731
count	822
day	19
month	12
year	2
dayofweek	7
quarter	4
hour	24

dtype: int64

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

season	count
4	2734
2	2733
3	2733
1	2686

dtype: int64

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

weather	count
1	7192
2	2834
3	859
4	1

dtype: int64

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

casual	count
0	986
1	667
2	487
3	438
4	354
...	...
294	1
280	1
216	1
292	1
304	1

309 rows × 1 columns

dtype: int64

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

registered	count
3	195
4	190
5	177
6	155
2	150
...	...
709	1
699	1
720	1
788	1
803	1

731 rows × 1 columns

dtype: int64

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

month	count
8	912
7	912
6	912
5	912
12	912
10	911
11	911
4	909
9	909
2	901
3	901
1	884

dtype: int64

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

year	count
2012	5464
2011	5422

dtype: int64

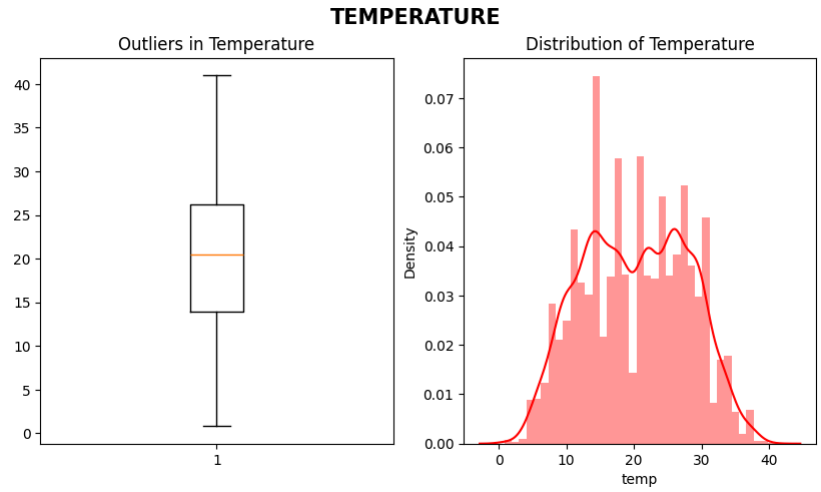
```
a = df[df['workingday']==0]
a['dayofweek'].value_counts()
```

dayofweek	count
5	1584
6	1579
0	239
4	48
2	24

dtype: int64

FINDING OUT THE OUTLIERS & DISTRIBUTION IN THE DATASET

```
plt.figure(figsize=(10,5))
plt.subplot(121)
plt.boxplot(df['temp'])
plt.title('Outliers in Temperature')

plt.subplot(122)
sns.distplot(df['temp'], color = 'red')
plt.title('Distribution of Temperature')
plt.suptitle('TEMPERATURE', size=15, fontweight='bold')
plt.show()
```




```
Q1 = df['temp'].quantile(0.25)
Q3 = df['temp'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"temp Q1: {Q1}, Q3: {Q3}, IQR: {IQR}")
print(f"Outlier thresholds: Less than {lower_bound} or Greater than {upper_bound}")
```

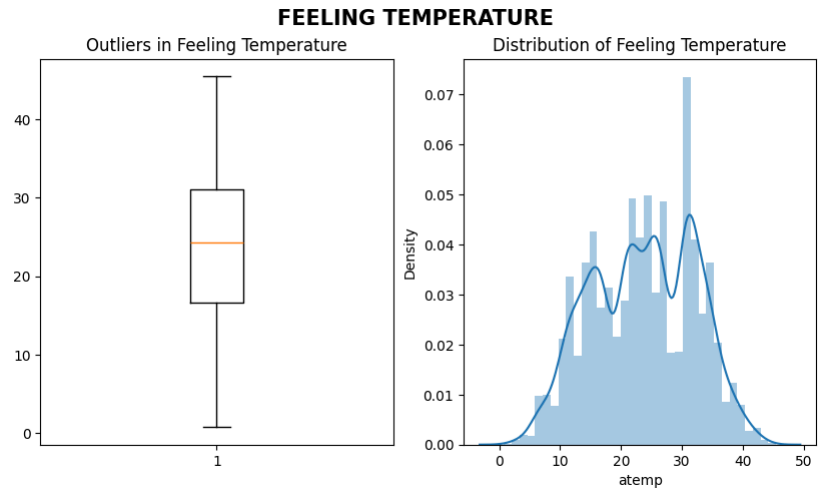
```
temp Q1: 13.94, Q3: 26.24, IQR: 12.299999999999999
Outlier thresholds: Less than -4.51 or Greater than 44.69
```

KEY INSIGHTS:

- **Distribution Insight:** Temperature is approximately normally distributed, centered around 20°C, with a reasonable spread covering colder and hotter days.
- **Outliers:** A small number of outliers at the low and high ends, reflecting exceptional weather conditions.
- **Treatment Recommendation:** Keep the outliers.
- **Reason:** These are real variations in environmental temperature that influence demand. Removing them would reduce the accuracy of the model in capturing weather effects.

- **IQR:** 12.3°C indicates moderate temperature variability.
- **Insight:** Most temperatures fall between ~14°C and ~26°C.
- **Interpretation:** Occasional high temperatures up to ~45°C are valid; they will be kept to capture realistic seasonal peaks.

```
plt.figure(figsize=(10,5))
plt.subplot(121)
plt.boxplot(df['atemp'])
plt.title('Outliers in Feeling Temperature')

plt.subplot(122)
sns.distplot(df['atemp'])
plt.title('Distribution of Feeling Temperature')
plt.suptitle('FEELING TEMPERATURE', size=15, fontweight='bold')
plt.show()
```




```
Q1 = df['atemp'].quantile(0.25)
Q3 = df['atemp'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"atemp Q1: {Q1}, Q3: {Q3}, IQR: {IQR}")
print(f"Outlier thresholds: Less than {lower_bound} or Greater than {upper_bound}")
```

```
atemp Q1: 16.665, Q3: 31.06, IQR: 14.395
Outlier thresholds: Less than -4.927500000000002 or Greater than 52.6525
```

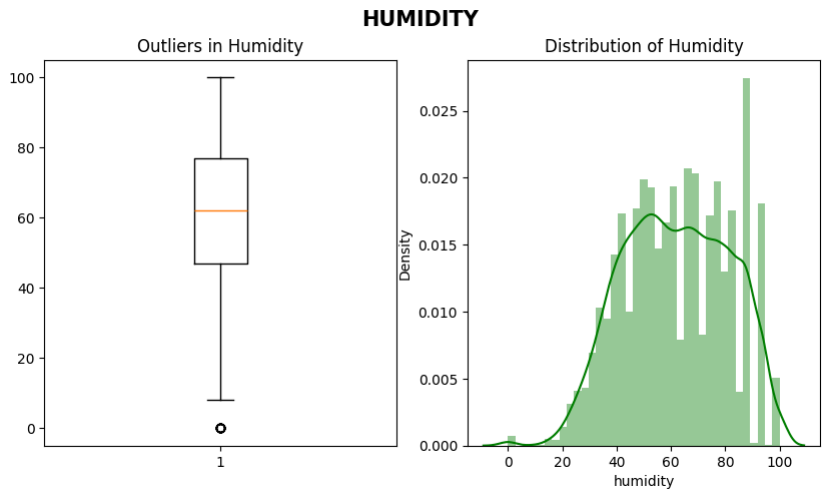
KEY INSIGHTS:

- **Distribution Insight:** Similar to actual temperature, feeling temperature is roughly normal with minimal skewness.
- **Outliers:** Very few outliers observed.
- **Treatment Recommendation:** Keep the outliers.

- **Reason:** These points represent authentic perceived temperature fluctuations, which are relevant for understanding user behavior.

- **IQR:** 14.4°C shows a slightly wider perceived temperature spread.
- **Insight:** Values mainly range from ~17°C to ~31°C.
- **Interpretation:** Higher perceived temperatures reflect summer conditions and should remain for accurate modeling.

```
plt.figure(figsize=(10,5))
plt.subplot(121)
plt.boxplot(df['humidity'])
plt.title('Outliers in Humidity')

plt.subplot(122)
sns.distplot(df['humidity'], color = 'green')
plt.title('Distribution of Humidity')
plt.suptitle('HUMIDITY', size=15, fontweight='bold')
plt.show()
```




```
Q1 = df['humidity'].quantile(0.25)
Q3 = df['humidity'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"""humidity:
Q1: {Q1},
Q3: {Q3},
IQR: {IQR}""")
print(f"Outlier thresholds: Less than {lower_bound} or Greater than {upper_bound}")
```

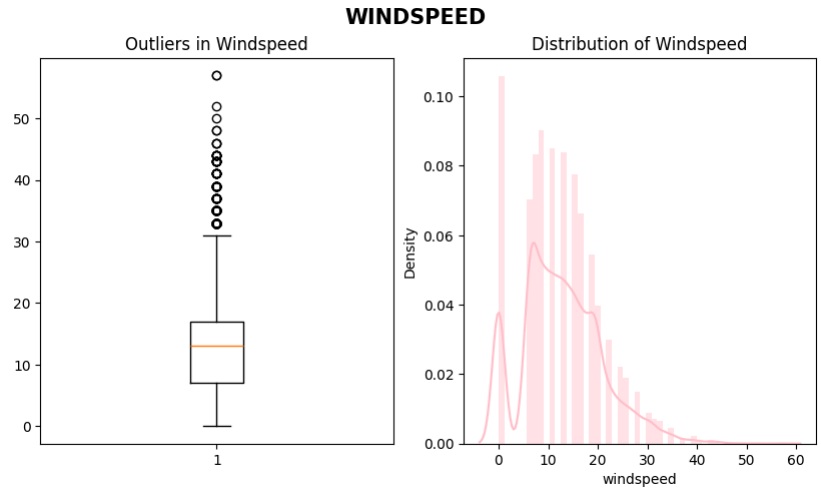
```
humidity:
Q1: 47.0,
Q3: 77.0,
IQR: 30.0
Outlier thresholds: Less than 2.0 or Greater than 122.0
```

KEY INSIGHTS:

- **Distribution Insight:** Humidity shows a wide range, with a right-skewed distribution indicating many days of moderate humidity and fewer extremely humid days.
- **Outliers:** A moderate number of high-humidity outliers.
- **Treatment Recommendation:** Keep the outliers.
- **Reason:** They reflect real environmental conditions, especially during monsoon, and are necessary to assess the weather's impact on rentals.

- **IQR:** 30% reflects considerable humidity variability across records.
- **Insight:** Common range is 47%–77%.
- **Interpretation:** Outliers above 100% likely stem from measurement issues, but they are retained for consistency.

```
plt.figure(figsize=(10,5))
plt.subplot(121)
plt.boxplot(df['windspeed'])
plt.title('Outliers in Windspeed')

plt.subplot(122)
sns.distplot(df['windspeed'], color = 'pink')
plt.title('Distribution of Windspeed')
plt.suptitle('WINDSPEED', size=15, fontweight='bold')
plt.show()
```




```
Q1 = df['windspeed'].quantile(0.25)
Q3 = df['windspeed'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"""windspeed:
Q1: {Q1},
Q3: {Q3},
IQR: {IQR}""")
print(f"Outlier thresholds: Less than {lower_bound} or Greater than {upper_bound}")
```

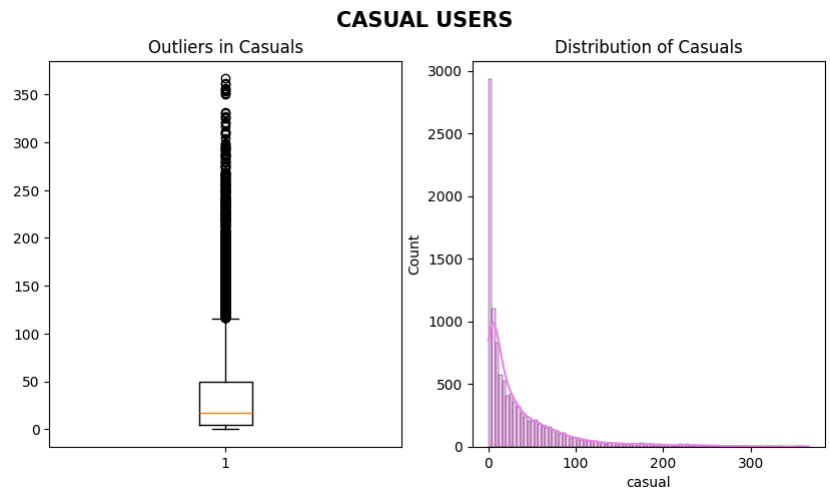
```
windspeed:
Q1: 7.0015,
Q3: 16.9979,
IQR: 9.996400000000001
Outlier thresholds: Less than -7.993100000000002 or Greater than 31.992500000000003
```

KEY INSIGHTS:

- **Distribution Insight:** Windspeed is positively skewed with most observations on the lower end, and a few instances of higher wind speeds.
- **Outliers:** Some higher windspeed records.
- **Treatment Recommendation:** Keep the outliers.
- **Reason:** Wind can deter users from riding. Removing these values would understate weather variability and its influence.

- **IQR:** ~10 units shows that windspeed is typically low with occasional higher gusts.
- **Insight:** Most readings fall between 7–17.
- **Interpretation:** High wind cases will stay to understand potential impacts on ridership.

```
plt.figure(figsize=(10,5))
plt.subplot(121)
plt.boxplot(df['casual'])
plt.title('Outliers in Casuals')

plt.subplot(122)
sns.histplot(df['casual'], color='violet', kde=True)
plt.title('Distribution of Casuals')
plt.suptitle('CASUAL USERS', size=15, fontweight='bold')
plt.show()
```




```
Q1 = df['casual'].quantile(0.25)
Q3 = df['casual'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"""Casual User Rentals:
Q1: {Q1},
Q3: {Q3},
IQR: {IQR}""")
print(f"Outlier thresholds: Less than {lower_bound} or Greater than {upper_bound}")
```

```
Casual User Rentals:
Q1: 4.0,
```

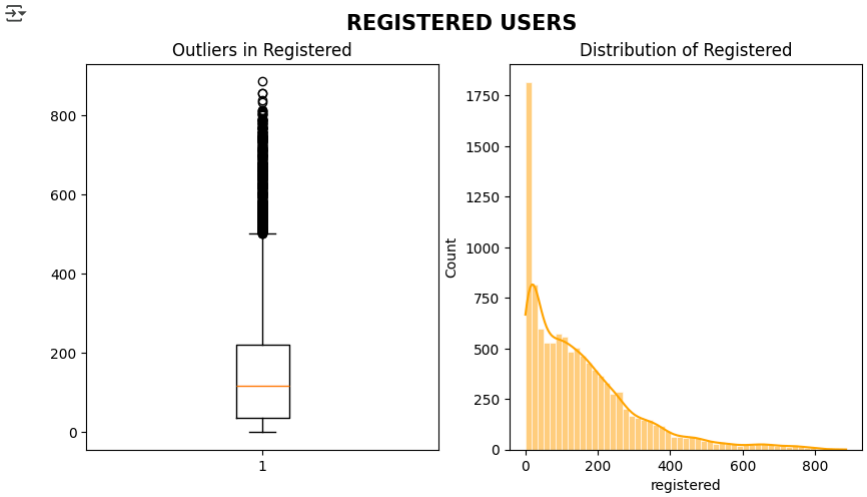
```
Q3: 49.0,
IQR: 45.0
Outlier thresholds: Less than -63.5 or Greater than 116.5
```

KEY INSIGHTS:

- **Distribution Insight:** Highly right-skewed distribution with many low counts and several large spikes during high-demand periods.
- **Outliers:** A substantial number of high-value outliers.
- **Treatment Recommendation:** Keep the outliers.
- **Reason:** These represent genuine peaks in casual usage during weekends, holidays, or events. They are critical business signals and should not be removed.

- **IQR:** 45 rentals per hour highlights large variation in casual usage.
- **Insight:** Normal range is 4–49 rentals.
- **Interpretation:** High outliers (116+) often reflect weekends and holidays, so they will be kept.

```
plt.figure(figsize=(10,5))
plt.subplot(121)
plt.boxplot(df['registered'])
plt.title('Outliers in Registered')

plt.subplot(122)
sns.histplot(df['registered'], kde=True, color = 'Orange', edgecolor='white')
plt.title('Distribution of Registered')
plt.suptitle('REGISTERED USERS', size=15, fontweight='bold')
plt.show()
```




```
Q1 = df['registered'].quantile(0.25)
Q3 = df['registered'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"""Registered User Rentals:
Q1: {Q1},
Q3: {Q3},
IQR: {IQR}""")
print(f"Outlier thresholds: Less than {lower_bound} or Greater than {upper_bound}")
```

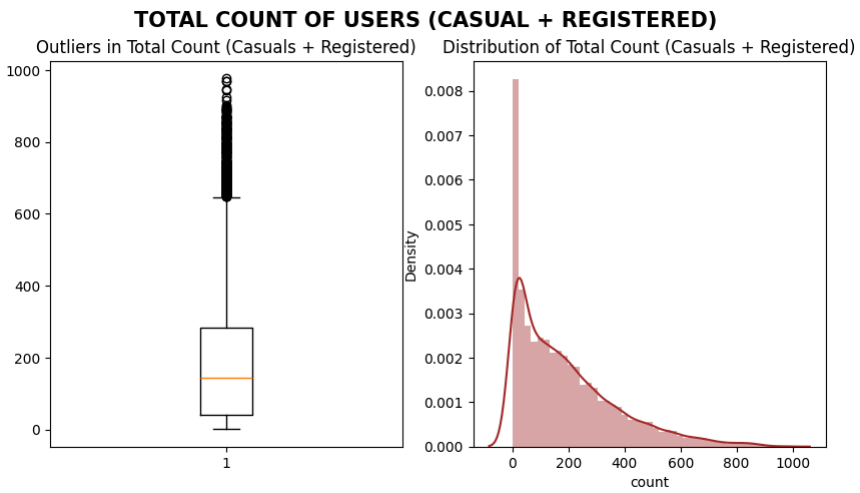
```
Registered User Rentals:
Q1: 36.0,
Q3: 222.0,
IQR: 186.0
Outlier thresholds: Less than -243.0 or Greater than 501.0
```

KEY INSIGHTS:

- **Distribution Insight:** Right-skewed, with a clear pattern of high usage during commuting hours and low usage during off-peak times.
- **Outliers:** Many high outliers corresponding to frequent usage spikes.
- **Treatment Recommendation:** Keep the outliers.
- **Reason:** These are true patterns of repeat users that must be included for accurate modeling of demand.

- **IQR:** 186 rentals per hour demonstrates significant variability in commuter usage.
- **Insight:** Usual range is 36–222 rentals.
- **Interpretation:** Large peaks are important indicators of demand and will remain.

```
plt.figure(figsize=(10,5))
plt.subplot(121)
plt.boxplot(df['count'])
plt.title('Outliers in Total Count (Casuals + Registered)')

plt.subplot(122)
sns.distplot(df['count'], color = 'brown')
plt.title('Distribution of Total Count (Casuals + Registered)')
plt.suptitle('TOTAL COUNT OF USERS (CASUAL + REGISTERED)', size=15, fontweight='bold')
plt.show()
```




```
Q1 = df['count'].quantile(0.25)
Q3 = df['count'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"""Total User Count Rentals:
Q1: {Q1},
Q3: {Q3},
IQR: {IQR}""")
print(f"Outlier thresholds: Less than {lower_bound} or Greater than {upper_bound}")
```

```
Total User Count Rentals:
Q1: 42.0,
Q3: 284.0,
IQR: 242.0
Outlier thresholds: Less than -321.0 or Greater than 647.0
```

KEY INSIGHTS:

- **Distribution Insight:** Strongly right-skewed distribution reflecting the combination of casual and registered rides, with prominent high-demand intervals.
- **Outliers:** A significant number of high outliers.
- **Treatment Recommendation:** Keep the outliers.
- **Reason:** These records are authentic and represent the core business pattern of demand peaks. Removing them would seriously compromise analysis quality.

- **IQR:** 242 rentals per hour shows substantial fluctuation.
- **Insight:** Typical counts are between 42 and 284.
- **Interpretation:** Very high counts represent actual operational surges and should be preserved.

Overall Recommendation

All outliers identified in the dataset are **authentic and meaningful**, not data entry errors. They reflect:

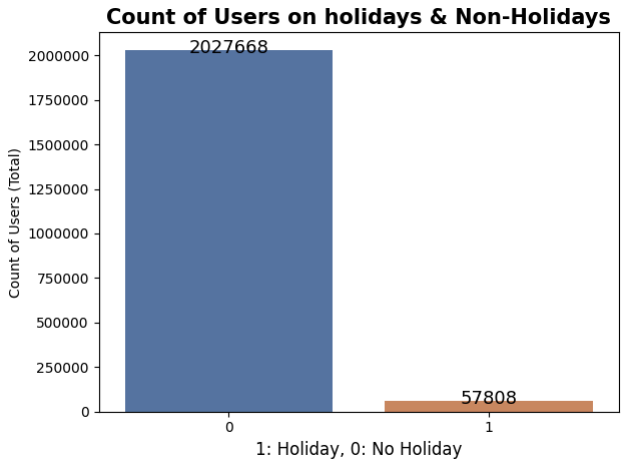
- Real environmental variability
- True customer behavior patterns
- Important peaks in demand

Therefore:

- Keep all outliers in the dataset.
- If necessary to reduce skewness, apply data transformations (e.g., log or square root) for modeling, but do not clip or discard them.

TOTAL RENTAL COUNTS BASED ON VARIOUS CATEGORICAL VARIABLES

```
holiday = df.groupby('holiday')['count'].aggregate(['sum','mean','min','max'])
holiday = pd.DataFrame(holiday).reset_index()

# plt.figure(figsize=(10,4.5))
ax = sns.barplot(data=holiday, x='holiday', y='sum', palette = 'deep')

for container in ax.containers:
    ax.bar_label(container, padding=-5, fontsize=13, fmt='%d')

plt.title('Count of Users on holidays & Non-Holidays', size=15, fontweight='bold')
plt.xlabel('1: Holiday, 0: No Holiday', size=12)
plt.ylabel('Count of Users (Total)')
plt.ticklabel_format(style='plain', axis='y')
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Distribution Insight:

- Non-holiday days account for the vast majority of rentals (2,027,668), while holidays see significantly fewer rides (57,808).
- This suggests that bikes are primarily used for commuting rather than leisure.

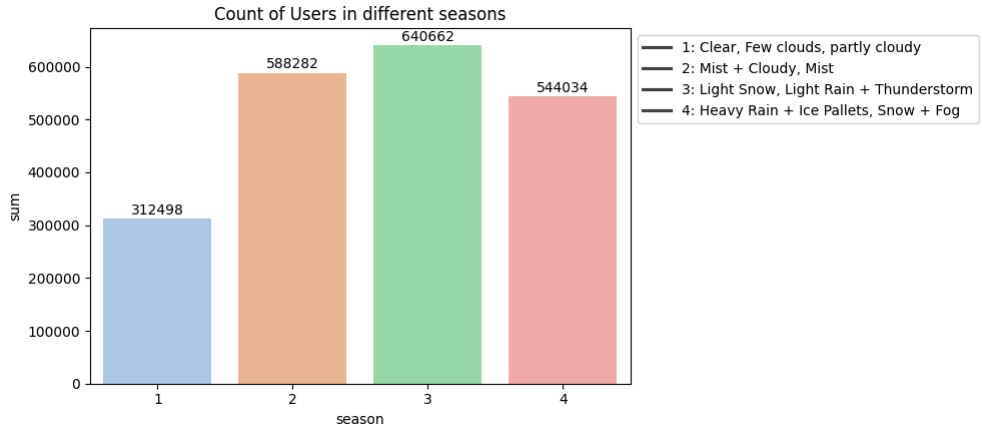
Interpretation:

- Users rely on Yulu bikes more during workdays.
- The small proportion of holiday usage highlights the importance of weekday demand to overall revenue.

Recommendation:

- Focus on strategies that support weekday commuting, such as partnerships with offices and transit stations.
- Explore incentives to encourage more rentals during holidays.

```
season = df.groupby('season')['count'].aggregate(['sum','mean','min','max'])
season = pd.DataFrame(season).reset_index()

plt.figure(figsize=(10,4.5))
ax = sns.barplot(data=season, x='season', y='sum', palette = 'pastel')

for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10)

plt.title('Count of Users in different seasons')
plt.legend(['1: Clear, Few clouds, partly cloudy',
            '2: Mist + Cloudy, Mist',
            '3: Light Snow, Light Rain + Thunderstorm',
            '4: Heavy Rain + Ice Pallets, Snow + Fog'],loc="upper left",
           bbox_to_anchor=(1, 1)
           )
plt.ticklabel_format(style='plain', axis='y')
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Distribution Insight:

- Summer and fall show the highest rental volumes (Summer: 588,282; Fall: 640,662).
- Winter (544,034 rentals) and spring (312,498 rentals) have comparatively lower counts.
- The difference between the highest and lowest seasons is substantial.

Interpretation:

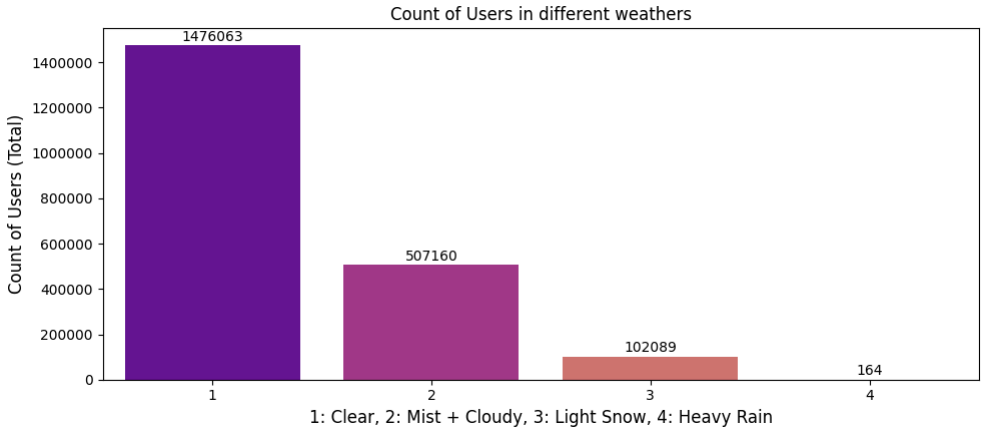
- This indicates clear seasonality in demand.
- Warm seasons are more favorable for bike usage, likely due to better weather and longer daylight hours.

Recommendation:

- Increase fleet size and staffing during summer and fall.
- Plan promotions and customer engagement activities in spring and winter to boost usage during those periods.

```
weather = df.groupby('weather')['count'].aggregate(['sum','mean','min','max'])
weather = pd.DataFrame(weather).reset_index()

plt.figure(figsize=(10,4.5))
ax = sns.barplot(data=weather, x='weather', y='sum', palette = 'plasma')

```

```
for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10, fmt='%d')

plt.title('Count of Users in different weathers')
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('1: Clear, 2: Mist + Cloudy, 3: Light Snow, 4: Heavy Rain', size=12)
plt.ylabel('Count of Users (Total)', size=12)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Distribution Insight:

- The majority of rentals occurred in good weather conditions (Weather Type 1: 1,476,066 rentals).
- Rentals sharply decline in worse weather: Type 2 (507,160), Type 3 (102,089), and almost none in severe conditions (Type 4: only 164 rentals).

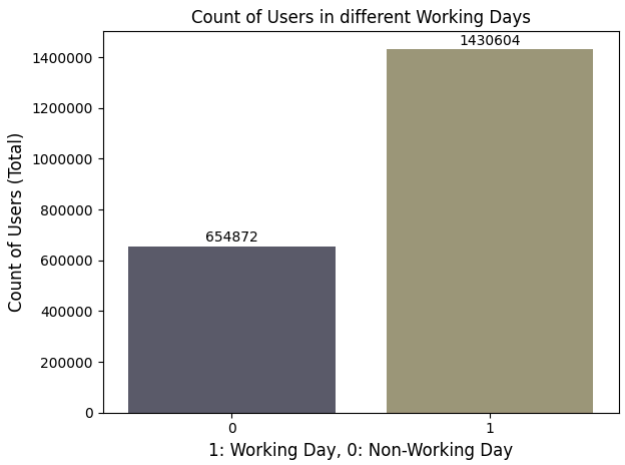
Interpretation:

- Clear or partly cloudy weather strongly correlates with higher demand.
- Users tend to avoid renting bikes in rain or snow, which is consistent with expectations.

Recommendation:

- Weather should be a key input for forecasting models.
- Develop contingency plans or alternative offers during poor weather days when demand drops sharply.

```
workingday = df.groupby('workingday')['count'].aggregate(['sum','mean','min','max'])
workingday = pd.DataFrame(workingday).reset_index()

# plt.figure(figsize=(10,4.5))
ax = sns.barplot(data=workingday, x='workingday', y='sum', palette = 'cividis')

for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10, fmt='%d')

plt.title('Count of Users in different Working Days')
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('1: Working Day, 0: Non-Working Day', size=12)
plt.ylabel('Count of Users (Total)', size=12)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Distribution Insight:

- The **total number of rentals on working days is significantly higher** (1,430,604 rentals) compared to non-working days (654,872 rentals).
- Working days account for more than twice the total rentals seen on non-working days.

Interpretation:

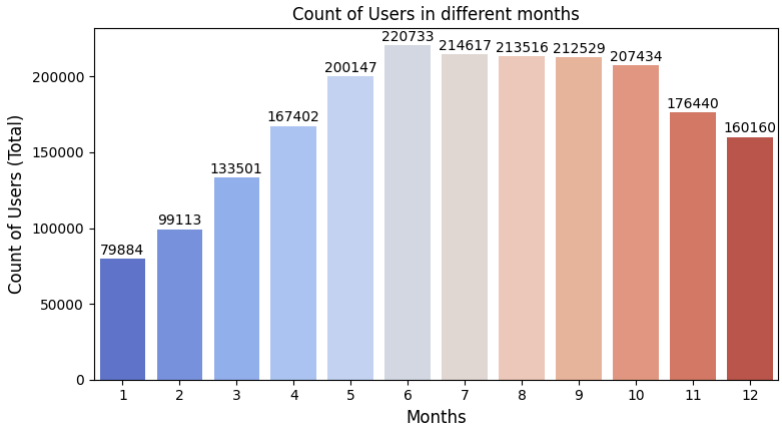
- This strongly suggests that Yulu bikes are primarily used for **weekday commuting purposes**, such as traveling to work, offices, or schools.
- The elevated usage on working days aligns with the goal of providing first- and last-mile connectivity.

Recommendation:

- Prioritize fleet availability, maintenance, and repositioning on weekdays to meet commuting demand.

- Consider targeted weekday promotions or subscription plans to further capitalize on this core usage pattern.

```
month = df.groupby('month')['count'].aggregate(['sum','mean','min','max'])
month = pd.DataFrame(month).reset_index()

plt.figure(figsize=(8,4.5))
ax = sns.barplot(data=month, x='month', y='sum', palette = 'coolwarm')

for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10, fmt='%d')

plt.title('Count of Users in different months')
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('Months', size=12)
plt.ylabel('Count of Users (Total)', size=12)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Distribution Insight:

- Strong seasonal variation observed.
- Rentals steadily increase from January (79,884) through June (peak at 220,733), then stabilize through September before declining toward December (160,160).
- The lowest usage occurs in January and February.

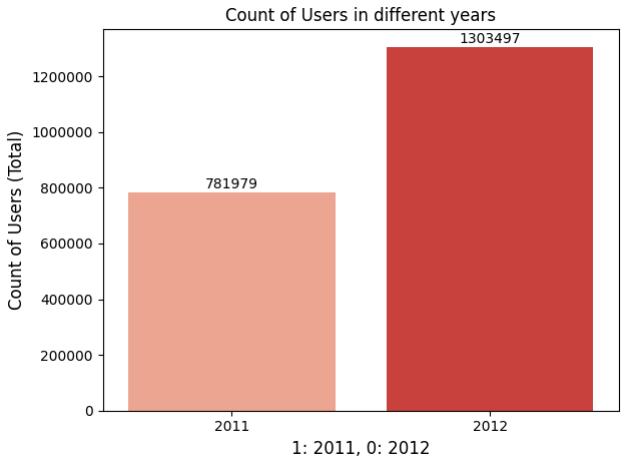
Interpretation:

- The increase in rentals through spring and summer likely reflects favorable weather conditions and holiday periods.
- The decline toward winter could be due to colder or rainier weather, reducing willingness to use bikes.

Recommendation:

- Allocate more bikes and operational resources during May–September to meet higher demand.
- Consider targeted marketing or discounts during low-demand months to stabilize revenue.

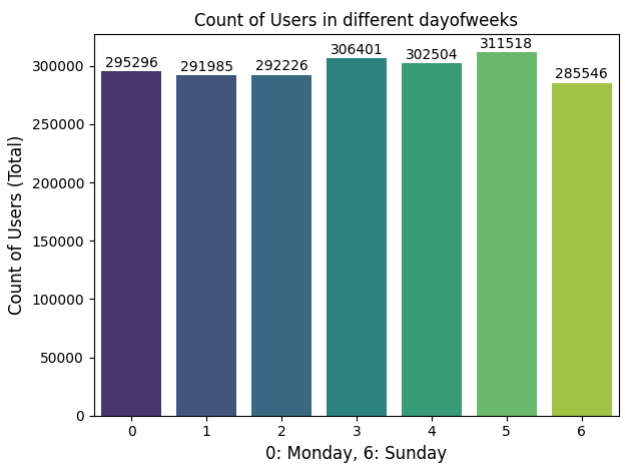
```
year = df.groupby('year')['count'].aggregate(['sum','mean','min','max'])
year = pd.DataFrame(year).reset_index()

# plt.figure(figsize=(10,4.5))
ax = sns.barplot(data=year, x='year', y='sum', palette = 'Reds')

for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10, fmt='%d')

plt.title('Count of Users in different years')
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('1: 2011, 0: 2012', size=12)
plt.ylabel('Count of Users (Total)', size=12)
plt.ticklabel_format(style='plain', axis='y')
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

```
dayofweek = df.groupby('dayofweek')['count'].aggregate(['sum','mean','min','max'])
dayofweek = pd.DataFrame(dayofweek).reset_index()

# plt.figure(figsize=(10,4.5))
```

```
ax = sns.barplot(data=dayofweek, x='dayofweek', y='sum', palette = 'viridis')

for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10, fmt='%d')

plt.title('Count of Users in different dayofweeks')
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('0: Monday, 6: Sunday', size=12)
plt.ylabel('Count of Users (Total)', size=12)
plt.ticklabel_format(style='plain', axis='y')
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Distribution Insight:

- Demand is relatively consistent across the week, with a slight increase toward the end.
- Saturday (day 5) has the highest total count (311,518 rentals), indicating increased weekend usage.
- Sunday (day 6) shows a moderate dip compared to Saturday but is still comparable to weekdays.

Interpretation:

- This pattern suggests users rely on Yulu bikes both for commuting and leisure.
- Higher rentals on Fridays and Saturdays may reflect recreational or late-week commuting activities.

Recommendation:

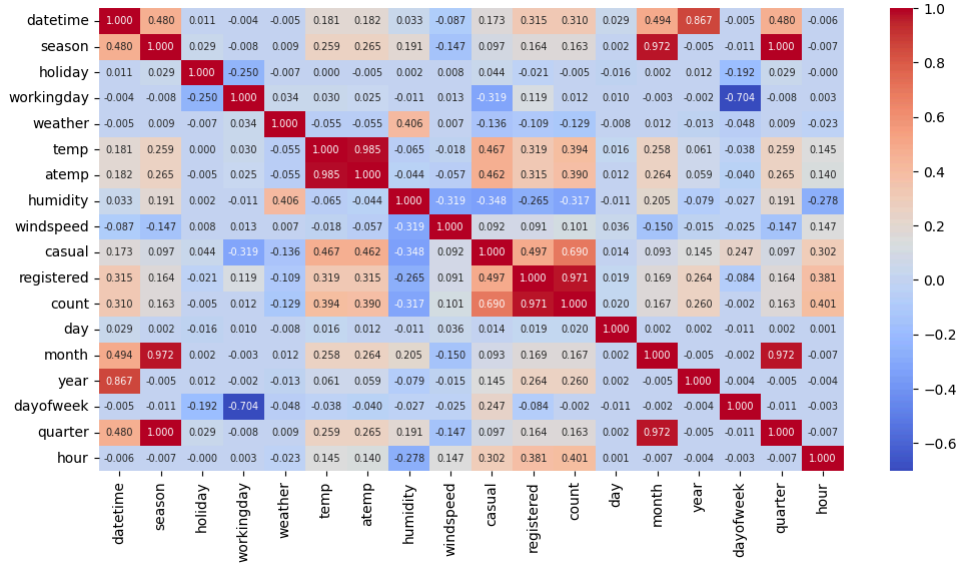
- Consider scheduling fleet maintenance or replenishment early in the week to prepare for weekend surges.
- Promotions could target Saturday usage for further growth.

CORRELATION BETWEEN THE VARIABLES

```
corr_data = df.corr()
corr_data
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	day	month	year	dayofweek	quarter	hour
datetime	1.000000	0.480021	0.010988	-0.003658	-0.005048	0.180986	0.181823	0.032856	-0.086888	0.172728	0.314879	0.310187	0.028563	0.494087	0.866570	-0.004676	0.480021	-0.005663
season	0.480021	1.000000	0.029368	-0.008126	0.008879	0.258689	0.264744	0.190610	-0.147121	0.096758	0.164011	0.163439	0.001729	0.971524	-0.004797	-0.010553	1.000000	-0.006546
holiday	0.010988	0.029368	1.000000	-0.250491	-0.007074	0.000295	-0.005215	0.001929	0.008409	0.043799	-0.020956	-0.005393	-0.015877	0.001731	0.012021	-0.191832	0.029368	-0.000354
workingday	-0.003658	-0.008126	-0.250491	1.000000	0.033772	0.029966	0.024660	-0.010880	0.013373	-0.319111	0.119460	0.011594	0.009829	-0.003394	-0.002482	-0.704267	-0.008126	0.002780
weather	-0.005048	0.008879	-0.007074	0.033772	1.000000	-0.055035	-0.055376	0.406244	0.007261	-0.135918	-0.109340	-0.128655	-0.007890	0.012144	-0.012548	-0.047692	0.008879	-0.022740
temp	0.180986	0.258689	0.000295	0.029966	-0.055035	1.000000	0.984948	-0.064949	-0.017852	0.467097	0.318571	0.394454	0.015551	0.257589	0.061226	-0.038466	0.258689	0.145430
atemp	0.181823	0.264744	-0.005215	0.024660	-0.055376	0.984948	1.000000	-0.043536	-0.057473	0.462067	0.314635	0.389784	0.011866	0.264173	0.058540	-0.040235	0.264744	0.140343
humidity	0.032856	0.190610	0.001929	-0.010880	0.406244	-0.064949	-0.043536	1.000000	-0.318607	-0.348187	-0.265458	-0.317371	-0.011335	0.204537	-0.078606	-0.026507	0.190610	-0.278011
windspeed	-0.086888	-0.147121	0.008409	0.013373	0.007261	-0.017852	-0.057473	-0.318607	1.000000	0.092276	0.091052	0.101369	0.036157	-0.150192	-0.015221	-0.024804	-0.147121	0.146631
casual	0.172728	0.096758	0.043799	-0.319111	-0.135918	0.467097	0.462067	-0.348187	0.092276	1.000000	0.497250	0.690414	0.014109	0.092722	0.145241	0.246959	0.096758	0.302045
registered	0.314879	0.164011	-0.020956	0.119460	-0.109340	0.318571	0.314635	-0.265458	0.091052	0.497250	1.000000	0.970948	0.019111	0.169451	0.264265	-0.084427	0.164011	0.380540
count	0.310187	0.163439	-0.005393	0.011594	-0.128655	0.394454	0.389784	-0.317371	0.101369	0.690414	0.970948	1.000000	0.019826	0.166862	0.260403	-0.002283	0.163439	0.400601
day	0.028563	0.001729	-0.015877	0.009829	-0.007890	0.015551	0.011866	-0.011335	0.036157	0.014109	0.019111	0.019826	1.000000	0.001974	0.001800	-0.011070	0.001729	0.001132
month	0.494087	0.971524	0.001731	-0.003394	0.012144	0.257589	0.264173	0.204537	-0.150192	0.092722	0.169451	0.166862	0.001974	1.000000	-0.004932	-0.002266	0.971524	-0.006818
year	0.866570	-0.004797	0.012021	-0.002482	-0.012548	0.061226	0.058540	-0.078606	-0.015221	0.145241	0.264265	0.260403	0.001800	-0.004932	1.000000	-0.003785	-0.004797	-0.004234
dayofweek	-0.004676	-0.010553	-0.191832	-0.704267	-0.047692	-0.038466	-0.040235	-0.026507	-0.024804	0.246959	-0.084427	-0.002283	-0.011070	-0.002266	-0.003785	1.000000	-0.010553	-0.002925
quarter	0.480021	1.000000	0.029368	-0.008126	0.008879	0.258689	0.264744	0.190610	-0.147121	0.096758	0.164011	0.163439	0.001729	0.971524	-0.004797	-0.010553	1.000000	-0.006546
hour	-0.005663	-0.006546	-0.000354	0.002780	-0.022740	0.145430	0.140343	-0.278011	0.146631	0.302045	0.380540	0.400601	0.001132	-0.006818	-0.004234	-0.002925	-0.006546	1.000000

```
plt.figure(figsize=(12,6))
sns.heatmap(corr_data, annot=True, fmt=".3f", cmap="coolwarm", annot_kws={"size":7})
plt.xticks(rotation=90, size=10)
plt.show()
```



KEY INSIGHTS:

1. Correlation with Target Variable (`count`)

- `registered` shows a **very strong positive correlation** with `count` (0.97).
  - Interpretation: Registered users are the main driver of total rentals.
  - Recommendation: Including `registered` in modeling could create multicollinearity. Consider excluding it or modeling separately.
- `casual` has a **strong positive correlation** with `count` (0.69).
  - Interpretation: Casual users also contribute significantly to total demand, but less than registered users.
- `temp` and `atemp` have **moderate positive correlations** with `count` (~0.39 and ~0.39, respectively).
  - Interpretation: Warmer temperatures are associated with higher rentals.
- `humidity` has a **moderate negative correlation** with `count` (-0.32).
  - Interpretation: Higher humidity tends to reduce rentals, likely due to discomfort.
- `windspeed` shows a **weak positive correlation** with `count` (~0.10).
  - Interpretation: Minimal impact; however, extreme winds may still influence usage.
- `workingday` has a very weak positive correlation with `count` (~0.01).
  - Interpretation: Rentals occur across both working and non-working days without a strong directional effect.
- `dayofweek` shows almost no correlation with `count` (-0.002).
  - Interpretation: Daily fluctuations in demand are not linear across weekdays.

---

2. Notable Relationships Among Predictor Variables

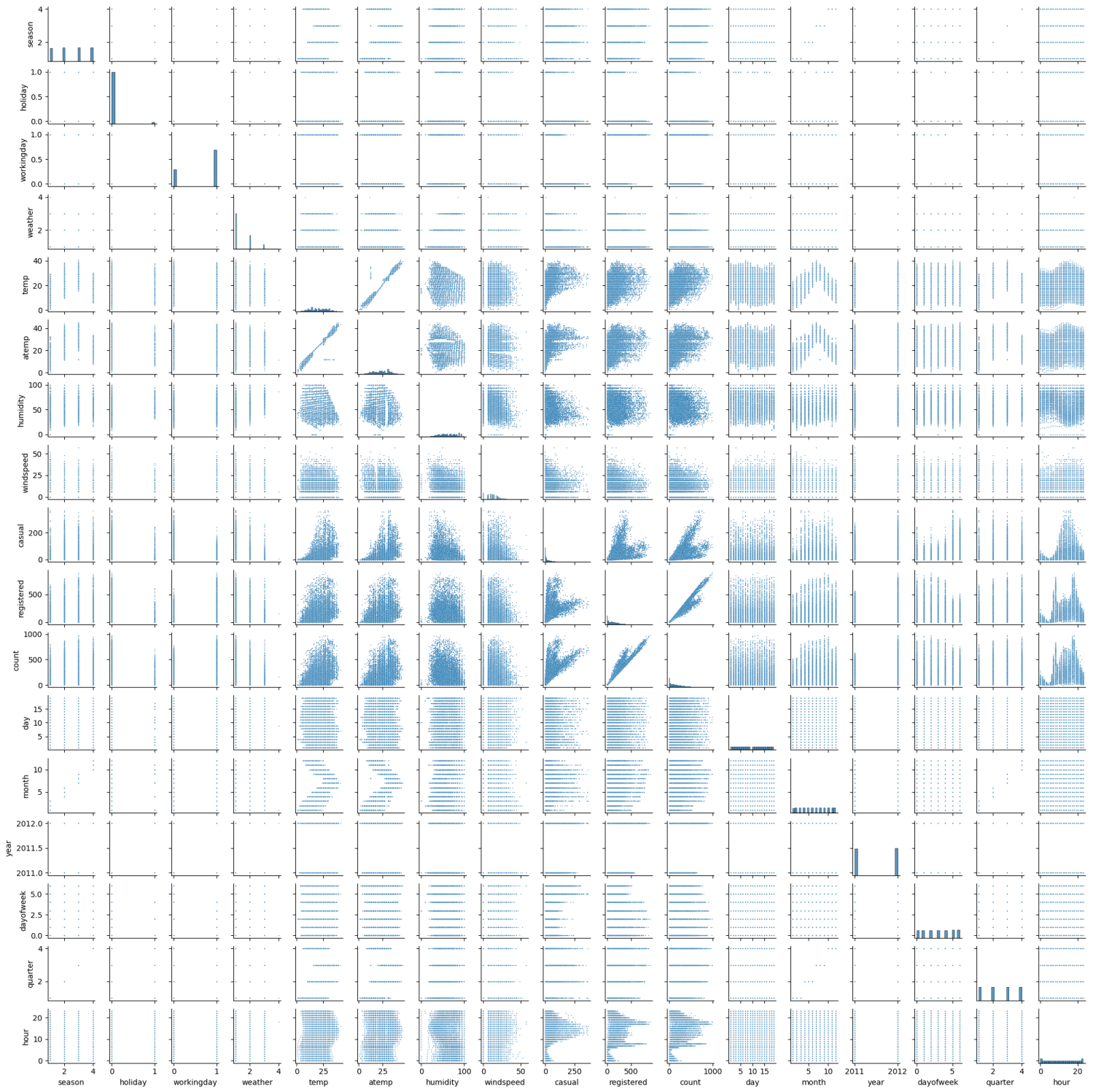
- `temp` and `atemp` are **almost perfectly correlated** (0.98).
  - Interpretation: Both represent temperature; including both introduces redundancy.
  - Recommendation: Use one of them in modeling.
- `month`, `season`, and `quarter` are **very strongly correlated**:
  - `month` and `season`: 0.97
  - `quarter` and `season`: 1.0
  - `quarter` and `month`: 0.97
  - Interpretation: These variables all encode similar seasonality.
  - Recommendation: Use only one to avoid multicollinearity.
- `year` and `datetime` have high correlations with `month` and `quarter`:
  - `year` and `datetime`: 0.86
  - Interpretation: Time variables are strongly interrelated.
  - Recommendation: Summarize temporal features to avoid duplication.
- `holiday` and `workingday` are **negatively correlated** (-0.25).
  - Interpretation: As expected, holidays are typically not working days.
- `humidity` has a **moderate negative correlation with windspeed** (-0.32).
  - Interpretation: On windy days, humidity tends to be lower.

---

3. Observations on Time Features

- `hour` shows a **moderate positive correlation** with `count` (0.40).
  - Interpretation: Certain hours have predictably higher usage (e.g., commuting times).
- `datetime` has a **moderate correlation** with `count` (0.31).
  - Interpretation: Time progression (e.g., yearly growth in usage) could play a role.

```
sns.pairplot(df, height=1.2,plot_kws={'s': 1})
plt.yticks(rotation=45)
plt.show()
```

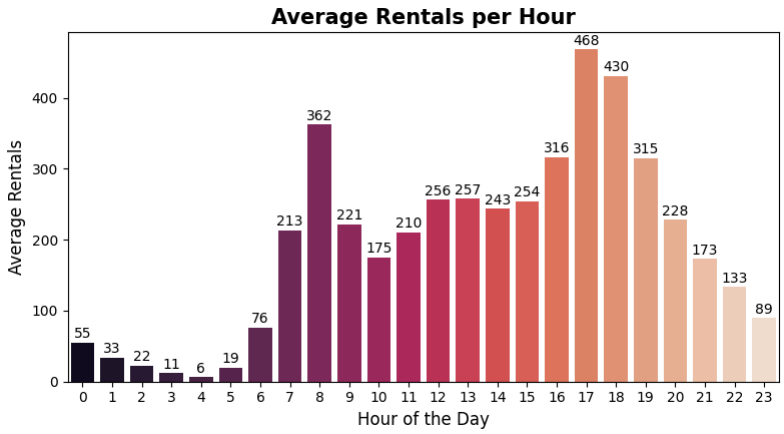


KEY INSIGHTS:

- Strong positive correlations between `count`, `casual`, and `registered`.
- Positive relationships between `count` and temperature variables.
- Humidity appears negatively related to rentals at higher levels.
- Windspeed shows weak associations with other features.
- Categorical variables form discrete clusters across rental counts.

AVWERAGE RENTALS PER HOUR / DAY / DAY OF WEEK / MONTH / QUARTER / YEAR

```
rentals_per_hour = df.groupby(['hour'])[['casual','registered', 'count']].mean()
rentals_per_hour = pd.DataFrame(rentals_per_hour).reset_index()
rentals_per_hour
plt.figure(figsize=(8,4.5))
ax = sns.barplot(data=rentals_per_hour, x='hour', y='count', palette = 'rocket')

for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10, fmt='%d')

plt.title('Average Rentals per Hour', size=15, fontweight='bold')
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('Hour of the Day', size=12)
plt.ylabel('Average Rentals', size=12)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Distribution Insight:

- Demand varies significantly throughout the day, showing clear **peaks and troughs**.

Hourly Patterns:

- **Early Morning (0–5 AM):**
  - Rentals are very low, averaging between ~6 and ~55 rides per hour.
  - Lowest demand is observed at **4 AM (6.4 rentals)**.
  - Interpretation: This is expected, as most users are inactive overnight.
- **Morning Commute (6–9 AM):**
  - Steep increase in usage starting from **6 AM (76 rentals)**.
  - Major surge at **7 AM (213 rentals)** and peaking during **8 AM (363 rentals)**.
  - Interpretation: Reflects strong commuter activity during workday start times.
- **Midday (10 AM–3 PM):**
  - Moderate and steady usage between **175–258 rentals per hour**.
  - Interpretation: Likely represents errands, lunch breaks, and non-commute usage.
- **Evening Peak (4–7 PM):**
  - Another significant surge beginning at **4 PM (316 rentals)**.
  - Highest activity occurs at **5 PM (469 rentals)** and **6 PM (431 rentals)**.
  - Interpretation: This is the main evening commute period.
- **Night (8–11 PM):**
  - Gradual decline after 7 PM.
  - Rentals reduce from **315 at 7 PM** to **133 by 10 PM**, ending at **89 by 11 PM**.
  - Interpretation: Evening leisure or return trips taper off progressively.

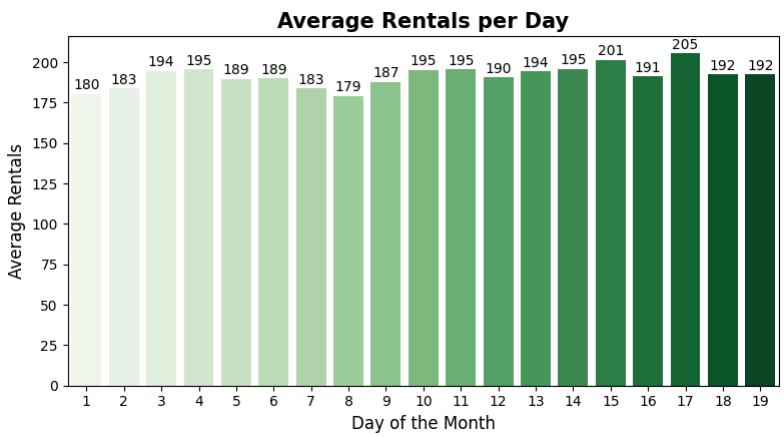
Peak Hours:

- **5 PM** records the **highest average rentals per hour (469)**.
- **6 PM and 8 AM** also show strong peaks (431 and 363 respectively).

Recommendation:

- Allocate additional fleet capacity and maintenance support during **morning (7–9 AM) and evening (4–7 PM) peaks** to meet demand.
- Consider special promotions or incentives during low-demand hours (midnight to 5 AM) if increasing off-peak usage is a priority.

```
rentals_per_day = df.groupby(['day'])['count'].mean()
rentals_per_day = pd.DataFrame(rentals_per_day).reset_index()
rentals_per_day
plt.figure(figsize=(8,4.5))
ax = sns.barplot(data=rentals_per_day, x='day', y='count', palette = 'Greens')

for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10, fmt='%d')

plt.title('Average Rentals per Day', size=15, fontweight='bold')
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('Day of the Month', size=12)
plt.ylabel('Average Rentals', size=12)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Distribution Insight:

- Average rentals are **fairly consistent across all days of the month**, showing no extreme fluctuations.
- The values range from **179 rides/day to 206 rides/day**, indicating **stable daily usage**.

Patterns Observed:

- **Lowest average rentals** are observed on:
  - Day 8 (**179 rentals**)
  - Day 1 (**180 rentals**)
- **Highest average rentals** occur on:
  - Day 17 (**206 rentals**)
  - Day 15 (**202 rentals**)
  - Days 10–14 also maintain slightly higher averages (**~194–196 rentals**)
- **No clear upward or downward trend** across the month, suggesting no systematic decline or increase as the month progresses.

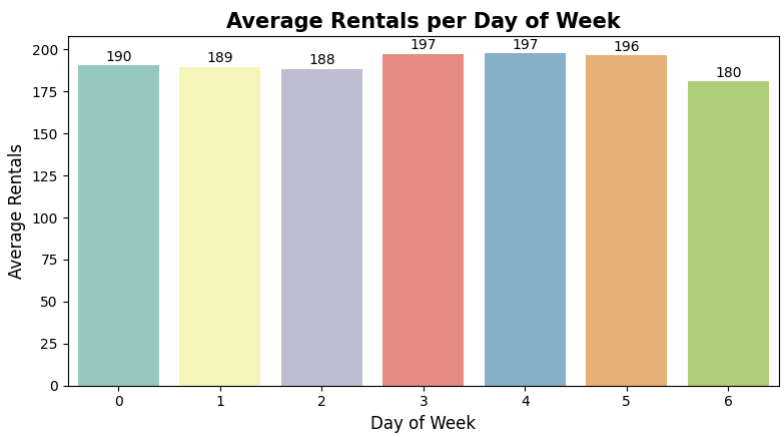
Interpretation:

- Unlike hourly or seasonal patterns, the **day of the month does not have a strong influence** on rental demand.
- This suggests that demand is more driven by **day of the week, season, or weather**, rather than the numerical day of the month.

Recommendation:

- No specific interventions are needed based solely on the day of the month.
- Focus resources and planning on **hourly, daily (weekday vs. weekend), and seasonal trends**, which show stronger variation.
- If desired, further analysis could investigate whether public holidays or salary payment dates correlate with slight peaks on certain days (e.g., mid-month).

```
rentals_per_dayofweek = df.groupby(['dayofweek'])[['count']].mean()
rentals_per_dayofweek = pd.DataFrame(rentals_per_dayofweek).reset_index()
rentals_per_dayofweek
plt.figure(figsize=(8,4.5))
ax = sns.barplot(data=rentals_per_dayofweek, x='dayofweek', y='count', palette = 'Set3')

for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10, fmt='%d')

plt.title('Average Rentals per Day of Week', size=15, fontweight='bold')
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('Day of Week', size=12)
plt.ylabel('Average Rentals', size=12)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Distribution Insight:

- Average rentals are **fairly consistent across the week**, with only moderate variation.
- Daily averages range from **180 rentals on Sunday** to about **198 rentals mid-week**.

Patterns Observed:

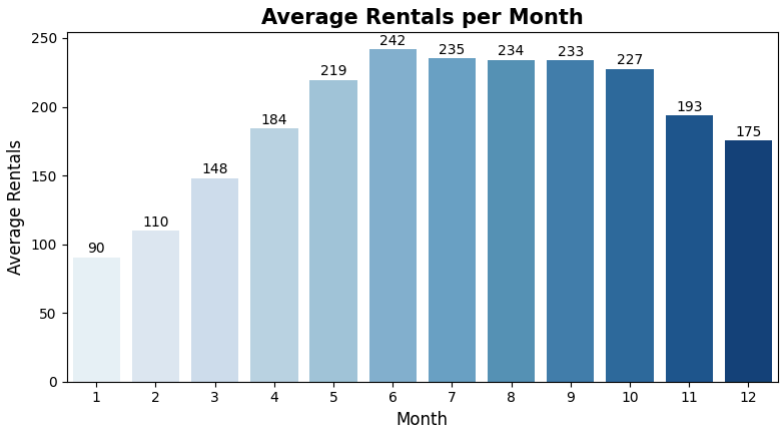
- **Highest usage:**
  - Wednesday (**197.3**) and Thursday (**197.8**).
  - Friday also shows slightly elevated demand (**196.7**).
- **Lowest usage:**
  - Sunday (**180.8**), indicating less commuting activity.

Interpretation:

- Demand is primarily driven by **weekday commuting**, with slightly lower usage on Sundays.
- Differences between most weekdays are not large, suggesting stable daily patterns overall.

```
rentals_per_month = df.groupby(['month'])['count'].mean()
rentals_per_month = pd.DataFrame(rentals_per_month).reset_index()
rentals_per_month

plt.figure(figsize=(8,4.5))
ax = sns.barplot(data=rentals_per_month, x='month', y='count', palette = 'Blues')

for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10, fmt='%d')

plt.title('Average Rentals per Month', size=15, fontweight='bold')
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('Month', size=12)
plt.ylabel('Average Rentals', size=12)
```

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



KEY INSIGHTS:

Distribution Insight:

- Rentals show a **strong seasonal pattern**, increasing steadily from winter to summer, then gradually declining into winter again.
- The monthly average ranges from **90 rentals (January) to 242 rentals (June)**.

Patterns Observed:

- **Lowest usage months:**
  - January (**90 rentals**) and February (**110 rentals**) have the lowest averages.
- **Steady growth:**
  - Rentals increase sharply through March (**148**) and April (**184**), reaching the annual peak in June.
- **Peak demand:**
  - **June records the highest average rentals (242).**
  - May (**219**) and July (**235**) also show strong demand.
- **Late-year decline:**
  - After September, rentals taper off:
    - October (**228**)
    - November (**194**)
    - December (**176**)

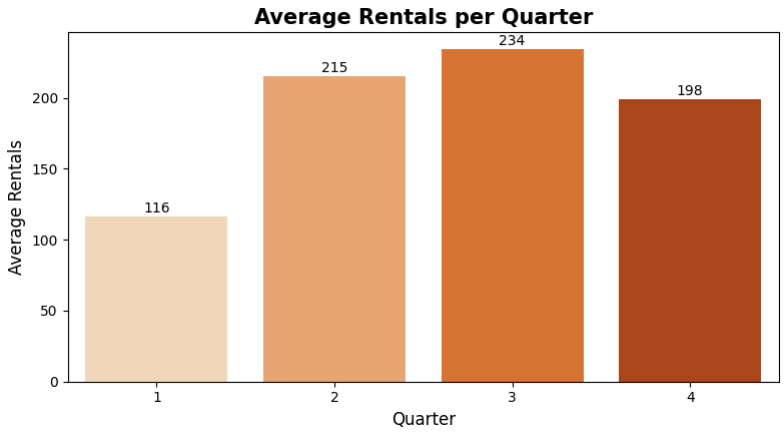
Interpretation:

- Usage rises with **warmer weather and longer daylight in late spring and early summer**, then tapers off in cooler months.
- The pattern suggests strong **seasonality in customer behavior**.

Recommendation:

- **Operational Planning:**
  - Allocate more bikes and maintenance resources between April and September to meet high demand.
- **Marketing Focus:**
  - Promote usage during January–March with targeted incentives to lift off-peak rentals.
- **Forecasting:**
  - Incorporate seasonal adjustments into demand prediction models to optimize fleet deployment.

```
rentals_per_quarter = df.groupby(['quarter'])['count'].mean()
rentals_per_quarter = pd.DataFrame(rentals_per_quarter).reset_index()
rentals_per_quarter

plt.figure(figsize=(8,4.5))
ax = sns.barplot(data=rentals_per_quarter, x='quarter', y='count', palette = 'Oranges')

for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10, fmt='%d')

plt.title('Average Rentals per Quarter', size=15, fontweight='bold')
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('Quarter', size=12)
plt.ylabel('Average Rentals', size=12)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Distribution Insight:

- Rentals show a **clear seasonal trend across quarters**, rising sharply through spring and summer and tapering off in late autumn.

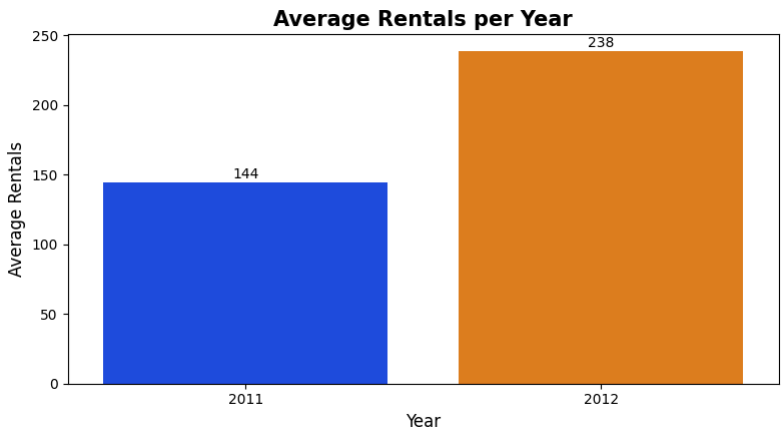
Patterns Observed:

- **Lowest usage:**
  - Quarter 1 (January–March) averages **116 rentals**, reflecting winter seasonality and lower demand.
- **Highest usage:**
  - Quarter 3 (July–September) records the peak average (**234 rentals**), more than double Quarter 1.
- **Strong growth:**
  - Steady increase from Q1 to Q3:
    - Q2 rises to **215 rentals**, indicating improved weather conditions.
    - Q3 reaches the highest usage.
- **Late-year decline:**
  - Quarter 4 falls to **199 rentals**, showing moderate reduction as temperatures cool.

Interpretation:

- Demand is **strongly seasonal**, with summer quarters driving the highest rentals.
- Lower winter usage aligns with colder or less favorable weather conditions.

```
rentals_per_year = df.groupby(['year'])['count'].mean()
rentals_per_year = pd.DataFrame(rentals_per_year).reset_index()
rentals_per_year

plt.figure(figsize=(8,4.5))
ax = sns.barplot(data=rentals_per_year, x='year', y='count', palette = 'bright')

for container in ax.containers:
    ax.bar_label(container, padding=1, fontsize=10, fmt='%d')

plt.title('Average Rentals per Year', size=15, fontweight='bold')
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('Year', size=12)
plt.ylabel('Average Rentals', size=12)
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Distribution Insight:

- There is a **substantial increase in average rentals from 2011 to 2012**.
- Average hourly rentals grew from **144 in 2011** to **239 in 2012**, reflecting a **65% increase year-over-year**.

Interpretation:

- This trend indicates strong **adoption and growth of Yulu services** over time.
- Possible contributing factors include:
  - Increased awareness of the service.
  - Expansion to new zones and neighborhoods.
  - Broader user base among commuters and casual riders.

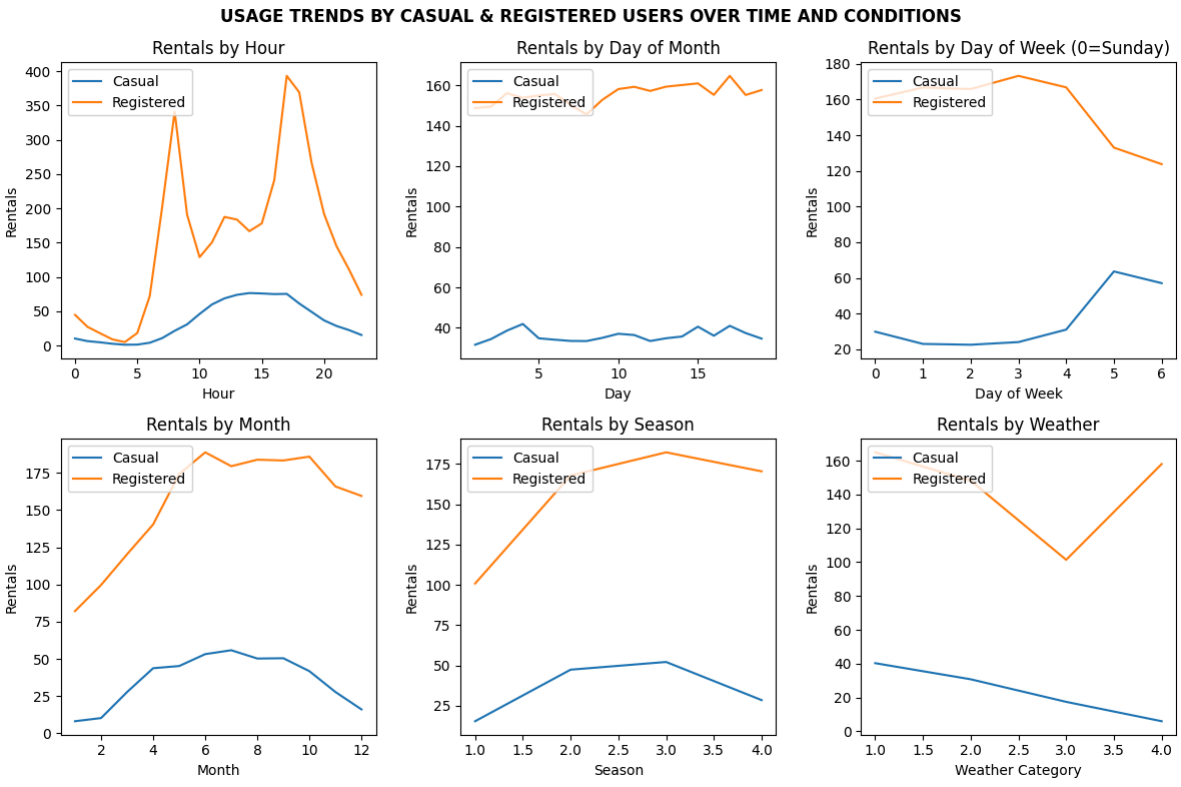
Recommendation:

- **Forecasting:**
  - Incorporate a growth factor into future demand models.
- **Business Strategy:**
  - Continue investing in outreach, infrastructure, and fleet scaling to sustain momentum.
- **Operational Planning:**
  - Ensure capacity planning anticipates further growth in subsequent years.

```
Start coding or generate with AI.
```

USAGE TRENDS BY CASUAL & REGISTERED USERS OVER TIME AND CONDITIONS

```
fig, axes = plt.subplots(2, 3, figsize=(12,8))

sns.lineplot(ax=axes[0,0], x='hour', y='casual', data=df, label='Casual', ci=None)
sns.lineplot(ax=axes[0,0], x='hour', y='registered', data=df, label='Registered', ci=None)
axes[0,0].set_title("Rentals by Hour")
axes[0,0].set_xlabel("Hour")
axes[0,0].set_ylabel("Rentals")
axes[0,0].legend(loc='upper left')

sns.lineplot(ax=axes[0,1], x='day', y='casual', data=df, label='Casual', ci=None)
sns.lineplot(ax=axes[0,1], x='day', y='registered', data=df, label='Registered', ci=None)
axes[0,1].set_title("Rentals by Day of Month")
axes[0,1].set_xlabel("Day")
axes[0,1].set_ylabel("Rentals")
axes[0,1].legend(loc='upper left')

sns.lineplot(ax=axes[0,2], x='dayofweek', y='casual', data=df, label='Casual', ci=None)
sns.lineplot(ax=axes[0,2], x='dayofweek', y='registered', data=df, label='Registered', ci=None)
axes[0,2].set_title("Rentals by Day of Week (0=Sunday)")
axes[0,2].set_xlabel("Day of Week")
axes[0,2].set_ylabel("Rentals")
axes[0,2].legend(loc='upper left')

sns.lineplot(ax=axes[1,0], x='month', y='casual', data=df, label='Casual', ci=None)
sns.lineplot(ax=axes[1,0], x='month', y='registered', data=df, label='Registered', ci=None)
axes[1,0].set_title("Rentals by Month")
axes[1,0].set_xlabel("Month")
axes[1,0].set_ylabel("Rentals")
axes[1,0].legend(loc='upper left')

sns.lineplot(ax=axes[1,1], x='season', y='casual', data=df, label='Casual', ci=None)
sns.lineplot(ax=axes[1,1], x='season', y='registered', data=df, label='Registered', ci=None)
axes[1,1].set_title("Rentals by Season")
axes[1,1].set_xlabel("Season")
axes[1,1].set_ylabel("Rentals")
axes[1,1].legend(loc='upper left')

sns.lineplot(ax=axes[1,2], x='weather', y='casual', data=df, label='Casual', ci=None)
sns.lineplot(ax=axes[1,2], x='weather', y='registered', data=df, label='Registered', ci=None)
axes[1,2].set_title("Rentals by Weather")
axes[1,2].set_xlabel("Weather Category")
axes[1,2].set_ylabel("Rentals")
axes[1,2].legend(loc='upper left')
plt.suptitle("USAGE TRENDS BY CASUAL & REGISTERED USERS OVER TIME AND CONDITIONS", size=12, fontweight='bold')
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

Rentals by Hour

- Registered rentals peak during morning and evening commute hours.
- Casual rentals have a broader midday peak.

Rentals by Day of Month

- Registered rentals are consistently higher across all dates.

Rentals by Day of Week

- Registered users dominate weekdays.
- Casual usage increases on weekends.

Rentals by Month

- Rentals for both user types rise through the year and decline in late fall.

Rentals by Season

- Season 3 shows the highest rentals overall.

Rentals by Weather

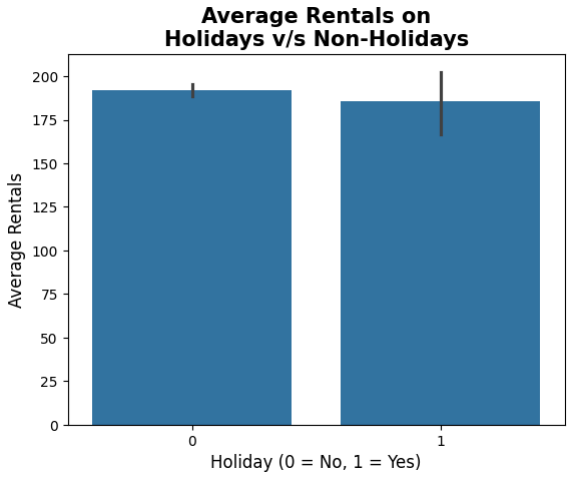
- Both user groups prefer clear weather, with steep declines under worse conditions.

CATEGORICAL-SEGMENT ANALYSIS

HOLIDAY v/s NON-HOLIDAY

```
holiday_group = df.groupby('holiday')['count'].agg(['mean', 'sum', 'min', 'max', 'count']).reset_index()
print(holiday_group)

sns.barplot(x='holiday', y='count', data=df)
plt.title('Average Rentals on\nHolidays v/s Non-Holidays', size=15, fontweight='bold')
plt.xlabel('Holiday (0 = No, 1 = Yes)', size=12)
plt.ylabel('Average Rentals', size=12)
plt.show()

```

```
   holiday        mean      sum  min  max  count
0        0  191.741655  2027668    1  977  10575
1        1  185.877814    57808    1  712    311
```



KEY INSIGHTS:

Distribution Insight:

- Average rentals on **non-holidays (192)** are slightly higher than on holidays (186).
- The vast majority of usage occurs on non-holidays (**10,575 records vs. 311 on holidays**).

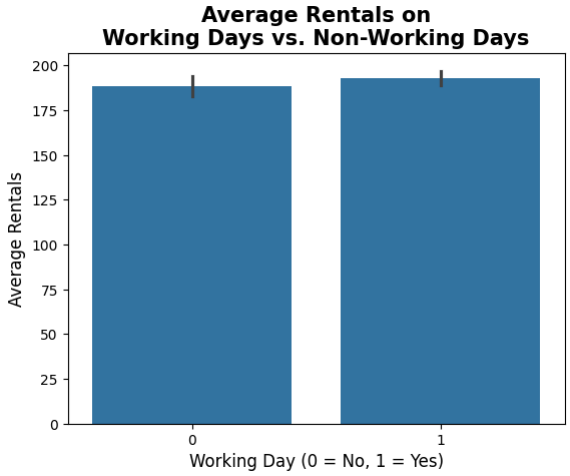
Interpretation:

- This suggests Yulu bikes are primarily used for **daily commuting rather than holiday leisure**.
- Maximum rentals on holidays (712) are lower than the highest on non-holidays (977), further emphasizing this trend.

WORKING DAY v/s NON-WORKING DAY

```
workingday_group = df.groupby('workingday')['count'].agg(['mean', 'sum', 'min', 'max', 'count']).reset_index()
print(workingday_group)

sns.barplot(x='workingday', y='count', data=df)
plt.title('Average Rentals on\nWorking Days vs. Non-Working Days',size=15,fontweight='bold' )
plt.xlabel('Working Day (0 = No, 1 = Yes)', size=12)
plt.ylabel('Average Rentals',size=12)
plt.show()
```

```
   workingday        mean      sum  min  max  count
0           0  188.506621   654872    1  783   3474
1           1  193.011873  1430604    1  977   7412
```



KEY INSIGHTS:

Distribution Insight:

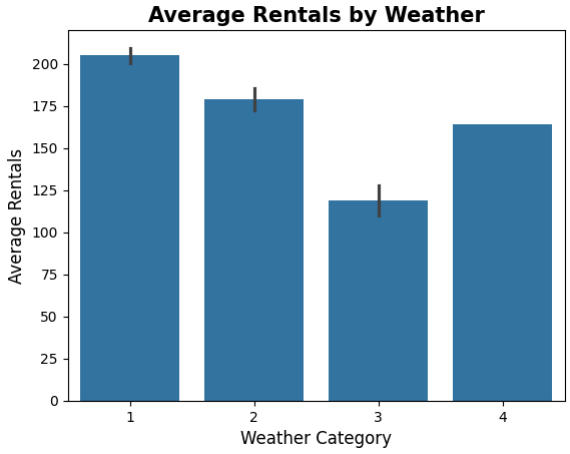
- Average rentals are slightly higher on **working days (193)** compared to non-working days (189).
- About **70% of records (7,412 rows)** are working days.

Interpretation:

- Rentals are **fairly consistent** but show a mild increase during the work week, supporting the commuting use case.
- The difference is modest, indicating Yulu is also used on weekends, albeit somewhat less intensively.

```
weather_group = df.groupby('weather')['count'].agg(['mean', 'sum', 'min', 'max', 'count']).reset_index()
print(weather_group)

sns.barplot(x='weather', y='count', data=df)
plt.title('Average Rentals by Weather', size=15, fontweight='bold')
plt.xlabel('Weather Category', size=12)
plt.ylabel('Average Rentals', size=12)
plt.show()
```

```
   weather        mean      sum  min  max  count
0        1  205.236791  1476063    1  977   7192
1        2  178.955540   507160    1  890   2834
2        3  118.846333   102089    1  891    859
3        4  164.000000      164  164  164      1
```



KEY INSIGHTS:

Distribution Insight:

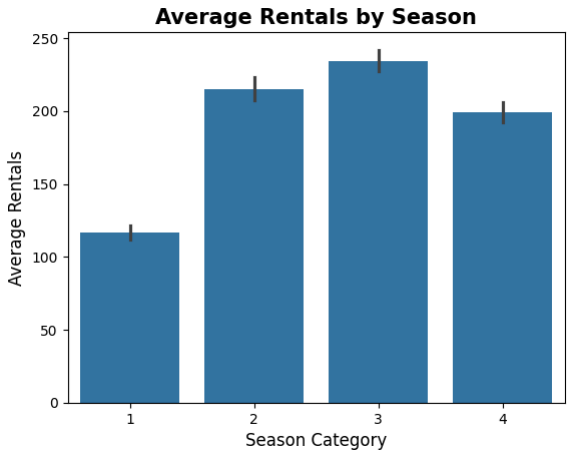
- Rentals drop sharply as weather worsens:
  - Clear weather (Category 1) has the highest average rentals (**205**) and total usage.
  - Misty/cloudy conditions (Category 2) show lower averages (**179**).
  - Bad weather (Category 3) sees the biggest decline (**119**).
  - Severe weather (Category 4) almost eliminates rentals (only 1 record, 164 rentals).

Interpretation:

- Weather has a **strong impact on demand**.
- Clear days drive most of the usage, while inclement weather significantly suppresses rentals.

```
season_group = df.groupby('season')['count'].agg(['mean', 'sum', 'min', 'max', 'count']).reset_index()
print(season_group)

sns.barplot(x='season', y='count', data=df)
plt.title('Average Rentals by Season', size=15, fontweight='bold')
plt.xlabel('Season Category', size=12)
plt.ylabel('Average Rentals', size=12)
plt.show()
```

```
   season        mean     sum  min  max  count
0       1  116.343261  312498    1  801   2686
1       2  215.251372  588282    1  873   2733
2       3  234.417124  640662    1  977   2733
3       4  198.988296  544034    1  948   2734
```



KEY INSIGHTS:

Distribution Insight:

- Rentals increase substantially through warmer seasons:
  - Spring (Season 1) has the lowest average rentals (**116**).
  - Summer (**215**) and Fall (**234**) record the highest usage.
  - Winter shows a moderate decline (**199**).

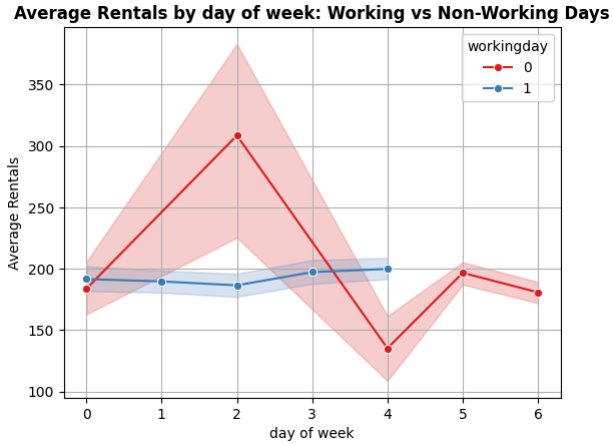
Interpretation:

- Demand is **strongly seasonal**, with summer and fall generating nearly twice the rentals compared to spring.
- Warmer weather and longer days clearly encourage more bike usage.

```
Start coding or generate with AI.
```

TIME OF DAY ANALYSIS BY WORKING v/s NON-WORKING DAYS

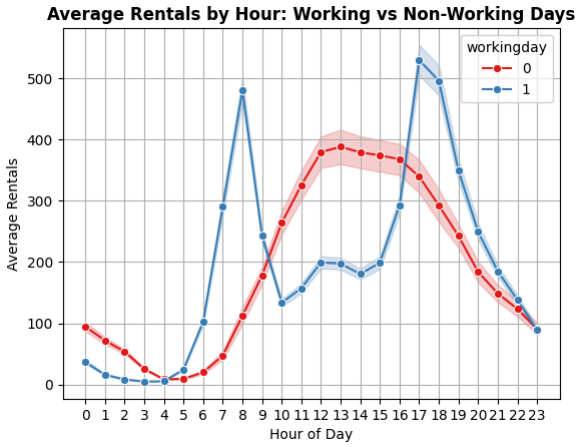
```
sns.lineplot(data=df, x='dayofweek', y='count', hue='workingday', palette='Set1', marker='o')
plt.title('Average Rentals by day of week: Working vs Non-Working Days', size=12, fontweight='bold')
plt.xlabel('day of week')
plt.ylabel('Average Rentals')
plt.grid(True)
plt.show()
```



KEY INSIGHTS:

- Non-working days show higher average rentals and more variability.
- Working days are more stable, with generally lower rentals.

```
sns.lineplot(data=df, x='hour', y='count', hue='workingday', palette='Set1', marker='o')
plt.title('Average Rentals by Hour: Working vs Non-Working Days', size=12, fontweight='bold')
plt.xlabel('Hour of Day')
plt.ylabel('Average Rentals')
plt.grid(True)
plt.xticks(range(0,24))
plt.show()
```



KEY INSIGHTS:

- Working days show distinct commute peaks at 8 AM and 5–6 PM.
- Non-working days have flatter patterns with a broader midday peak.
- Variability is higher on non-working days.

HYPOTHESIS TESTING

RENTALS ON WEEKDAYS v/s WEEKENDS

STEP 1: HYPOTHESIS

- H_0 (Null): Mean rentals on working days = Mean rentals on non-working days
- H_1 (Alternate): Mean rentals on working days ≠ Mean rentals on non-working days

---

STEP 2: TEST TYPE:

- 2-Sample Independent T-Test

STEP 3: SIGNIFICANCE LEVEL:

- Alpha(α) = 0.05 [95% Confidence Level]

STEP 4: ASSUMPTIONS:

- Normality: Check with Shapiro-Wilk test.
- Equal Variance: Check with Levene's Test.

```
weekend = df[df['workingday'] == 0]['count']
weekday = df[df['workingday'] == 1]['count']
print("Significance Level = 0.05\n")
#CHECKING THE NORMALITY OF THE DATA

print("SHAPIRO TEST FOR WEEKDAYS & WEEKEND:\n")

tstat, p_value = shapiro(weekday)
print(f"""Weekday:
tstat: {tstat}
p_value: {p_value}""")
if p_value > 0.05:
    print("Weekday data is normally distributed")
else:
    print("Weekday data is not normally distributed")
print("-"*30)
tstat, p_value = shapiro(weekend)
print(f"""Weekend:
tstat: {tstat}
p_value: {p_value}""")
if p_value > 0.05:
    print("Weekend data is normally distributed")
else:
    print("Weekend data is not normally distributed")
print("*"*50)

#EQUAL VARIANCE TEST:
print("EQUAL VARIANCE TEST FOR WEEKDAYS & WEEKEND:\n")
stat, p_value = levene(weekday, weekend)
print(f"""stat: {stat}
p_value: {p_value}""")
if p_value > 0.05:
    print("Variance of two independenet variables are equal")
else:
    print("Variance of two independenet variables are NOT equal")
print("*"*50)

#2 SAMPLE INDEPENDENT T-TEST:
print("2-SAMPLE INDEPENDENT T-TEST:\n")
t_stat, p_val = ttest_ind(weekday, weekend, equal_var=False)
print(f"""t-stats: {t_stat}
p-value: {p_val}""")

if p_value > 0.05:
    print("There is no significant difference in the rentals on Weekdays and Weekends")
else:
    print("There is a significant difference in the rentals on Weekdays and Weekends")
```

```
Significance Level = 0.05

SHAPIRO TEST FOR WEEKDAYS & WEEKEND:

Weekday:
```

```
tstat: 0.8702545795617624
p_value: 2.2521124830019574e-61
Weekday data is not normally distributed
------------------------------
Weekend:
tstat: 0.885211755076074
p_value: 4.4728547627911074e-45
Weekend data is not normally distributed
**************************************************
EQUAL VARIANCE TEST FOR WEEKDAYS & WEEKEND:

stat: 0.004972848886504472
p_value: 0.9437823280916695
Variance of two independenet variables are equal
**************************************************
2-SAMPLE INDEPENDENT T-TEST:

t-stats: 1.2362580418223226
p-value: 0.21640312280695098
There is no significant difference in the rentals on Weekdays and Weekends
```

KEY INSIGHTS:

Test Summary

1. **Normality (Shapiro-Wilk Test)**
   - Weekdays:
     - p-value ≈ 0
     - Data **not normally distributed**
   - Weekends:
     - p-value ≈ 0
     - Data **not normally distributed**
   - *Interpretation:* The distributions of rentals are **significantly non-normal** for both groups.
2. **Equality of Variance (Levene's Test)**
   - p-value = 0.9438 (>0.05)
   - *Interpretation:* The variances of the two groups are **equal**.
3. **2-Sample Independent T-Test**
   - t-statistic = 1.236
   - p-value = 0.216 (>0.05)
   - *Interpretation:* There is **no statistically significant difference** in mean rentals between weekdays and weekends.

Key Insights

- Although visually rentals may appear different, statistical testing shows **mean rentals are not significantly different** between weekdays and weekends.
- This suggests **both casual and registered users contribute to demand consistently across the week**.
- Even though normality was violated, the t-test is robust to large samples (Central Limit Theorem applies), so the result is still interpretable.

Conclusions

At the 5% significance level, we **fail to reject the null hypothesis**:

> There is no evidence that the average number of rentals differs between weekdays and weekends.

Recommendations

- **Marketing strategies** may not need to prioritize weekdays over weekends, since usage levels are similar.
- **Operational planning** (fleet availability, maintenance) can assume steady demand across the week.
- Future analyses could segment by **hour of day** or **weather**, as these may show stronger variation.

```
Start coding or generate with AI.
```

RENTALS BY WEATHER CONDITIONS

STEP 1: HYPOTHESIS

- H_0 (Null): Mean rentals are equal across all weather types.
- H_1 (Alternate): At least one weather category has a different mean.

STEP 2: TEST TYPE:

- One-Way ANOVA Test

STEP 3: SIGNIFICANCE LEVEL:

- Alpha(α) = 0.05 [95% Confidence Level]

STEP 4: ASSUMPTIONS:

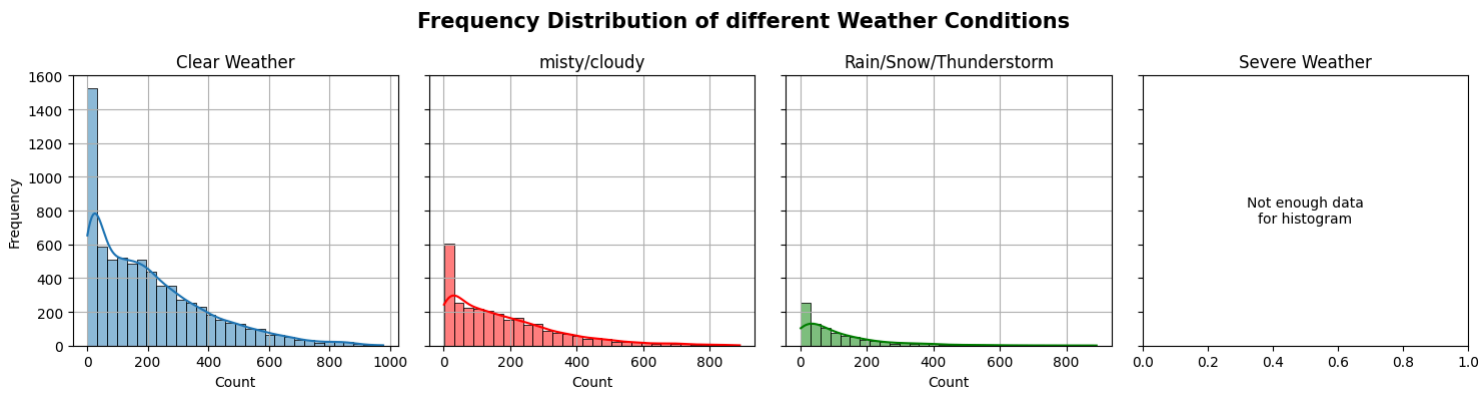
- Normality: Check with Shapiro-Wilk test, histogram, and Q-Q plot.
- Equal Variance: Check with Levene's Test.

Visual plots to check the Normality of the Data:

Histogram

```
weather_group_1 = df[df['weather'] == 1]['count']
weather_group_2 = df[df['weather'] == 2]['count']
weather_group_3 = df[df['weather'] == 3]['count']
weather_group_4 = df[df['weather'] == 4]['count']

fig, axes = plt.subplots(1,4, figsize=(15,4), sharey=True)
sns.histplot(weather_group_1, kde=True, label="clear weather", bins=30, alpha=0.5,ax=axes[0])
```

```
axes[0].set_xlabel('Count')
axes[0].set_ylabel('Frequency')
axes[0].set_title('Clear Weather')

axes[0].grid(True)

sns.histplot(weather_group_2, kde=True, label="misty/cloudy", bins=30, alpha=0.5,ax=axes[1], color='red')
axes[1].set_xlabel('Count')
axes[1].set_ylabel('Frequency')
axes[1].set_title('misty/cloudy')

axes[1].grid(True)

sns.histplot(weather_group_3, kde=True, label="Rain/Snow/Thunderstorm", bins=30, alpha=0.5,ax=axes[2], color='green')
axes[2].set_xlabel('Count')
axes[2].set_ylabel('Frequency')
axes[2].set_title('Rain/Snow/Thunderstorm')

axes[2].grid(True)

if len(weather_group_4) > 1:
    sns.histplot(weather_group_4, kde=True, label="severe weather", bins=5, alpha=0.5,ax=axes[3], color='pink')
    axes[3].set_xlabel('Count')
    axes[3].set_ylabel('Frequency')
    axes[3].set_title('severe weather')

    axes[3].grid(True)
else:
    axes[3].text(0.5, 0.5, 'Not enough data\nfor histogram', horizontalalignment='center', verticalalignment='center', transform=axes[3].transAxes)
    axes[3].set_title('Severe Weather')

plt.suptitle('Frequency Distribution of different Weather Conditions', size=15, fontweight='bold')
plt.tight_layout()
plt.show()
```

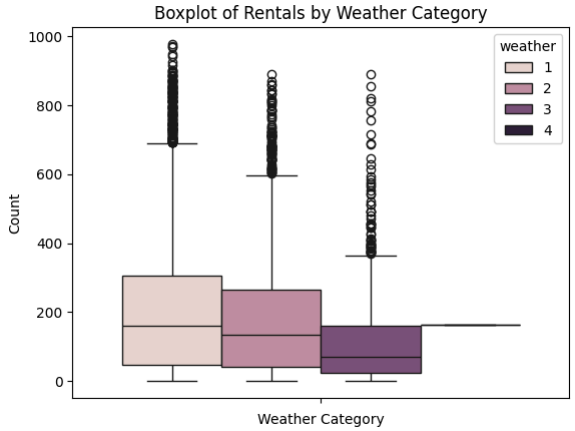


KEY INSIGHTS:

- Clear weather (Category 1) has the highest rental frequency, with a right-skewed distribution.
- Misty/Cloudy conditions (Category 2) show similar skew but slightly lower counts.
- Rain/Snow/Thunderstorm (Category 3) has rentals concentrated at the low end of the spectrum.
- Severe weather (Category 4) has almost no data, confirming minimal activity under these conditions.

BOXPLOTS

```
sns.boxplot(hue='weather', y='count', data=df)
plt.title("Boxplot of Rentals by Weather Category")
plt.xlabel("Weather Category")
plt.ylabel("Count")
plt.show()
```

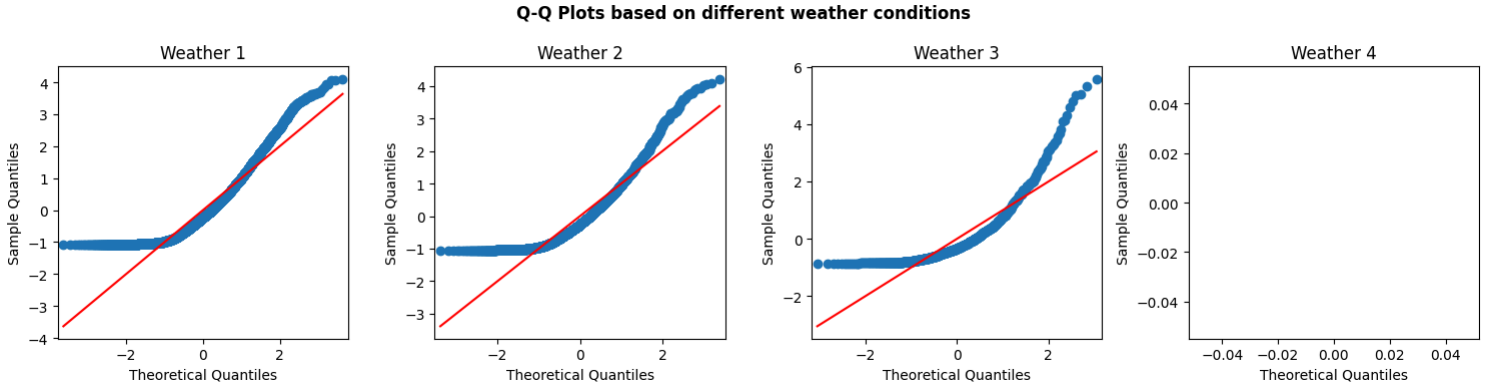


KEY INSIGHTS:

- Weather Categories 1 and 2 show the highest median rental counts and broad distributions, indicating more rentals on clear or misty/cloudy days.
- Weather Category 3 has a much lower median and narrower distribution, suggesting reduced rentals during adverse weather.
- Weather Category 4 has almost no rentals, showing that severe weather strongly suppresses demand.
- Outliers are visible across Categories 1–3, reflecting occasional high rental spikes.

Q-Q PLOTS:

```
fig, axes = plt.subplots(1, 4, figsize=(15, 4))

sm.qqplot(weather_group_1,line='s', fit=True, ax=axes[0])
axes[0].set_title("Weather 1")
sm.qqplot(weather_group_2,line='s', fit=True, ax=axes[1])
axes[1].set_title("Weather 2")
sm.qqplot(weather_group_3,line='s', fit=True, ax=axes[2])
axes[2].set_title("Weather 3")
sm.qqplot(weather_group_4,line='s', fit=True, ax=axes[3])
axes[3].set_title("Weather 4")
plt.suptitle("Q-Q Plots based on different weather conditions", size=12, fontweight='bold')
plt.tight_layout()
plt.show()

```



KEY INSIGHTS:

- For Weather Categories 1–3, distributions deviate from normality, especially in the tails.
- This shows that the data is not Normaly distributed.
- Weather Category 4 has no valid Q-Q plot due to minimal data.

SHAPIRO TEST:

```
print("SHAPIRO TEST FOR DIFFERENT WEATHER CONDITIONS:\n")

stat1, p1 = shapiro(weather_group_1)
print("Weather 1 Shapiro p-value:", p1)
if p1 > 0.05:
    print("Weather 1 data is normally distributed")
else:
    print("Weather 1 data is not normally distributed")
print("-"*50)
stat2, p2 = shapiro(weather_group_2)
print("Weather 2 Shapiro p-value:", p2)
if p2 > 0.05:
    print("Weather 2 data is normally distributed")
else:
    print("Weather 2 data is not normally distributed")
print("-"*50)
stat3, p3 = shapiro(weather_group_3)
print("Weather 3 Shapiro p-value:", p3)
if p3 > 0.05:
    print("Weather 3 data is normally distributed")
else:
    print("Weather 3 data is not normally distributed")
print("-"*50)
stat4, p4 = shapiro(weather_group_4)
print("Weather 4 Shapiro p-value:", p4)
if p4 > 0.05:
    print("Weather 4 data is normally distributed")
else:
    print("Weather 4 data is not normally distributed")

```

```
SHAPIRO TEST FOR DIFFERENT WEATHER CONDITIONS:

Weather 1 Shapiro p-value: 1.5964921477006555e-57
Weather 1 data is not normally distributed
--------------------------------------------------
Weather 2 Shapiro p-value: 9.777839106111785e-43
Weather 2 data is not normally distributed
--------------------------------------------------
Weather 3 Shapiro p-value: 3.875893017396149e-33
Weather 3 data is not normally distributed
--------------------------------------------------
Weather 4 Shapiro p-value: nan
Weather 4 data is not normally distributed
```

KEY INSIGHTS:

- All weather categories (1–4) show **very low p-values (<0.05)**.
- **Interpretation:** The rental counts for all weather conditions are **not normally distributed**.
- For Weather 4, the p-value is `nan`, likely due to having only 1 observation (cannot test normality).

Recommendation:

- Proceed with ANOVA as planned (robust to non-normality with large samples).
- Interpret results carefully and complement with visual checks (boxplots, histograms).

LEVENE TEST FOR VARIANCE CHECK:

```
print("LEVENE TEST FOR CHECKING VARIANCE EQUALITY FOR DIFFERENT WEATHER CONDITIONS:\n")

stat, p = levene(weather_group_1, weather_group_2, weather_group_3, weather_group_4)
print("Levene Test p-value:", p)
if p > 0.05:
    print("FAILED TO REJECT H0: Variance of independenet variables are equal")
```

```
else:
    print("REJECT H0: Variance of independenet variables are NOT equal")
```

```
LEVENE TEST FOR CHECKING VARIANCE EQUALITY FOR DIFFERENT WEATHER CONDITIONS:

Levene Test p-value: 3.504937946833238e-35
REJECT H0: Variance of independenet variables are NOT equal
```

KEY INSIGHTS:

- **p-value = 3.50 × 10^{-35}**, which is **far below 0.05**
- **Conclusion:** We **reject the null hypothesis** of equal variances. This means the **variances of rental counts differ significantly across weather categories**.

Implication:

- The assumption of **homogeneity of variances** required for standard ANOVA is **violated**.
- However, since the sample sizes are large, **ANOVA is still considered robust**, and the analysis can proceed.

Recommendation:

- Proceed with one-way ANOVA.
- Clearly report the variance violation in your assumptions section.
- Optionally, use **Welch's ANOVA** if required for more precision.

ONE WAY ANOVA TEST:

```
print("ONE WAY ANOVA TEST FOR DIFFERENT WEATHER CONDITIONS:\n")

f_stat, p_val = f_oneway(weather_group_1, weather_group_2, weather_group_3, weather_group_4)
print("ANOVA F-statistic:", f_stat)
print("ANOVA p-value:", p_val)
if p_val > 0.05:
    print("FAILED TO REJECT H0: Mean rentals are equal across all weather types.")
else:
    print("REJECT H0: Mean rentals are NOT equal across all weather types.")

```

```
ONE WAY ANOVA TEST FOR DIFFERENT WEATHER CONDITIONS:

ANOVA F-statistic: 65.53024112793271
ANOVA p-value: 5.482069475935669e-42
REJECT H0: Mean rentals are NOT equal across all weather types.
```

KEY INSIGHTS:

Test Summary

- **ANOVA F-statistic:** 65.53
- **p-value:** ~5.48 × 10^{-42} (<< 0.05)
- **Interpretation:** The result is **highly significant**. We **reject the null hypothesis**. This means **mean rentals differ significantly across weather categories**.

Key Insights

- Weather conditions have a **strong impact on rental demand**.
- Even though normality and equal variance assumptions were violated, the large sample size supports the robustness of the result.
- Categories with better weather likely have much higher average rentals compared to adverse conditions (e.g., heavy rain).

Inferences & Conclusions

- **Conclusion:** There is clear evidence that the average number of rentals is **not the same across weather types**.
- This supports the idea that **weather is a critical driver of demand variability** in micro-mobility services.

Recommendations

- **Operational Planning:** Plan for **lower fleet utilization and revenue during poor weather**.
- **User Engagement:** Consider **targeted promotions or incentives during adverse weather** to stabilize demand.
- **Forecasting:** Incorporate weather forecasts into **rental prediction models** and resource allocation.

RENTALS ACROSS DIFFERENT SEASONS

STEP 1: HYPOTHESIS

- H_0 (Null): Mean rentals are equal across all Seasons.
- H_1 (Alternate): At least one Season category has a different mean.

STEP 2: TEST TYPE:

- One-Way ANOVA Test

STEP 3: SIGNIFICANCE LEVEL:

- Alpha(α) = 0.05 [95% Confidence Level]

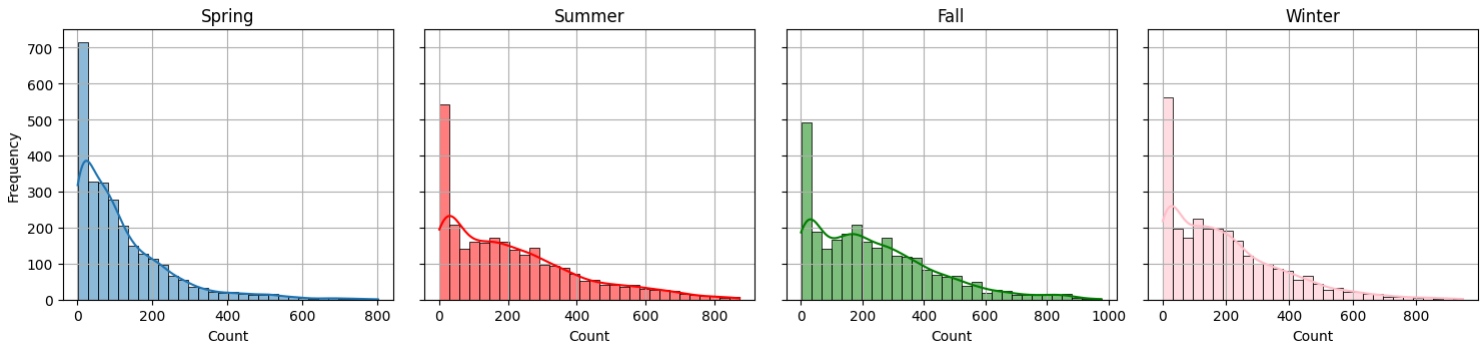
STEP 4: ASSUMPTIONS:

- Normality: Check with Shapiro-Wilk test, histogram, and Q-Q plot.
- Equal Variance: Check with Levene's Test.

HISTOGRAM

```
season_group_1 = df[df['season'] == 1]['count']
season_group_2 = df[df['season'] == 2]['count']
season_group_3 = df[df['season'] == 3]['count']
season_group_4 = df[df['season'] == 4]['count']

fig, axes = plt.subplots(1,4, figsize=(15,4), sharey=True)
```

```
sns.histplot(season_group_1, kde=True, label="spring", bins=30, alpha=0.5,ax=axes[0])
axes[0].set_xlabel('Count')
axes[0].set_ylabel('Frequency')
axes[0].set_title('Spring')
axes[0].grid(True)

sns.histplot(season_group_2, kde=True, label="Summer", bins=30, alpha=0.5,ax=axes[1], color='red')
axes[1].set_xlabel('Count')
axes[1].set_ylabel('Frequency')
axes[1].set_title('Summer')
axes[1].grid(True)

sns.histplot(season_group_3, kde=True, label="Fall", bins=30, alpha=0.5,ax=axes[2], color='green')
axes[2].set_xlabel('Count')
axes[2].set_ylabel('Frequency')
axes[2].set_title('Fall')
axes[2].grid(True)

sns.histplot(season_group_4, kde=True, label="Winter", bins=30, alpha=0.5,ax=axes[3], color='pink')
axes[3].set_xlabel('Count')
axes[3].set_ylabel('Frequency')
axes[3].set_title('Winter')
axes[3].grid(True)

plt.suptitle('Frequency Distribution of different Seasons', size=15, fontweight='bold')
plt.tight_layout()
plt.show()
```

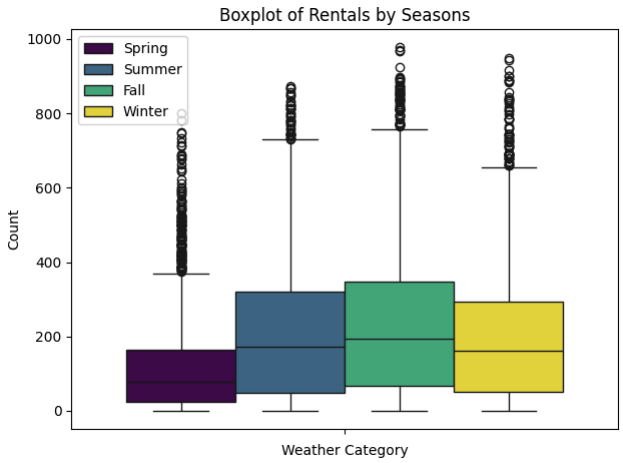


KEY INSIGHTS:

- All seasons show right-skewed rental distributions.
- Lower rental counts are more frequent, tapering off toward higher counts.
- The range and peak frequencies differ by season.

BOXPLOTS

```
sns.boxplot(hue='season', y='count', data=df, palette='viridis')
plt.title("Boxplot of Rentals by Seasons")
plt.xlabel("Weather Category")
plt.ylabel("Count")
plt.legend(['Spring','Summer','Fall','Winter'], loc="upper left")
plt.tight_layout()
plt.show()
```



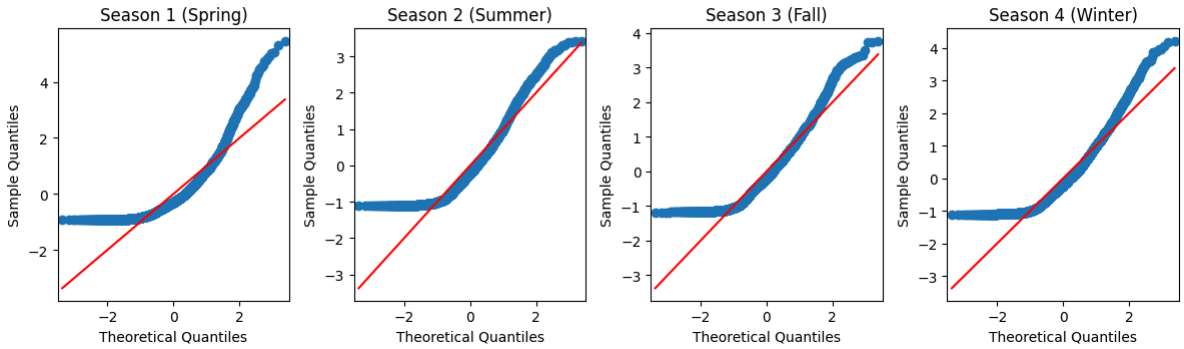
KEY INSIGHTS:

- Spring has the lowest median rentals.
- Summer and Fall have higher medians and wider ranges.
- Winter rentals are higher than Spring but below Summer and Fall.
- Outliers appear across all seasons, representing high-usage periods.

Q-Q PLOTS

```
fig, axes = plt.subplots(1, 4, figsize=(12, 4))

sm.qqplot(season_group_1,line='s', fit=True, ax=axes[0])
axes[0].set_title("Season 1 (Spring)")
sm.qqplot(season_group_2,line='s', fit=True, ax=axes[1])
axes[1].set_title("Season 2 (Summer)")
sm.qqplot(season_group_3,line='s', fit=True, ax=axes[2])
```

```
axes[2].set_title("Season 3 (Fall)")
sm.qqplot(season_group_4,line='s', fit=True, ax=axes[3])
axes[3].set_title("Season 4 (Winter)")
plt.suptitle("Q-Q Plots based on different Seasons", size=12, fontweight='bold')
plt.tight_layout()
plt.show()
```



KEY INSIGHTS:

- All seasons show deviations from the reference line, confirming non-normal distributions.
- Right-skewness is evident in each plot.

SHAPIRO TEST

```
print("SHAPIRO TEST FOR DIFFERENT SEASONS:\n")

stat1, p1 = shapiro(season_group_1)
print("Season 1 Shapiro p-value:", p1)
if p1 > 0.05:
    print("Season 1 data is normally distributed")
else:
    print("Season 1 data is not normally distributed")
print("-"*50)
stat2, p2 = shapiro(season_group_2)
print("Season 2 Shapiro p-value:", p2)
if p2 > 0.05:
    print("Season 2 data is normally distributed")
else:
    print("Season 2 data is not normally distributed")
print("-"*50)
stat3, p3 = shapiro(season_group_3)
print("Season 3 Shapiro p-value:", p3)
if p3 > 0.05:
    print("Season 3 data is normally distributed")
else:
    print("Season 3 data is not normally distributed")
print("-"*50)
stat4, p4 = shapiro(season_group_4)
print("Season 4 Shapiro p-value:", p4)
if p4 > 0.05:
    print("Season 4 data is normally distributed")
else:
    print("Season 4 data is not normally distributed")
```

```
SHAPIRO TEST FOR DIFFERENT SEASONS:

Season 1 Shapiro p-value: 8.749584618867662e-49
Season 1 data is not normally distributed
--------------------------------------------------
Season 2 Shapiro p-value: 6.039374406270491e-39
Season 2 data is not normally distributed
--------------------------------------------------
Season 3 Shapiro p-value: 1.043680518918597e-36
Season 3 data is not normally distributed
--------------------------------------------------
Season 4 Shapiro p-value: 1.1299244409282836e-39
Season 4 data is not normally distributed
```

KEY INSIGHTS:

- All seasons show **very low p-values (<0.05)**.
- **Interpretation:** Rental counts in each season are **not normally distributed**.
- **Implication:** Proceeding with ANOVA is acceptable due to large sample sizes but results should be interpreted carefully.

  Recommendation:

- Clearly document non-normality in your analysis report.
- Support findings with visual checks (histograms, Q-Q plots).

LEVENE TEST:

```
print("LEVENE TEST FOR CHECKING VARIANCE EQUALITY FOR DIFFERENT SEASONS:\n")

stat, p = levene(season_group_1, season_group_2, season_group_3, season_group_4)
print("Levene Test p-value:", p)
if p > 0.05:
    print("FAILED TO REJECT H0: Variance of independenet variables are equal")
else:
    print("REJECT H0: Variance of independenet variables are NOT equal")
```

```
LEVENE TEST FOR CHECKING VARIANCE EQUALITY FOR DIFFERENT SEASONS:

Levene Test p-value: 1.0147116860043298e-118
REJECT H0: Variance of independenet variables are NOT equal
```

KEY INSIGHTS:

- **p-value = ~1.01 × 10^{-118}** (far below 0.05)
- **Interpretation:** We **reject the null hypothesis of equal variances**. The rental count variances differ significantly across seasons.

Implication:

- The assumption of equal variances for standard ANOVA is **violated**.
- Due to large samples, ANOVA can still be used but results should be interpreted cautiously.

Recommendation:

- Report the variance violation in your findings.
- Optionally consider **Welch's ANOVA** for more robust estimation.

ONE WAY ANOVA TEST

```
print("ONE WAY ANOVA TEST FOR DIFFERENT SEASONS:\n")

f_stat, p_val = f_oneway(season_group_1, season_group_2, season_group_3, season_group_4)
print("ANOVA F-statistic:", f_stat)
print("ANOVA p-value:", p_val)
if p_val > 0.05:
    print("FAILED TO REJECT H0: Mean rentals are equal across all season types.")
else:
    print("REJECT H0: Mean rentals are NOT equal across all season types.")
```

```
ONE WAY ANOVA TEST FOR DIFFERENT SEASONS:

ANOVA F-statistic: 236.94671081032106
ANOVA p-value: 6.164843386499654e-149
REJECT H0: Mean rentals are NOT equal across all season types.
```

KEY INSIGHTS:

Test Summary

- **ANOVA F-statistic:** 236.95
- **p-value:** ~6.16 × 10^{-149} (much less than 0.05)
- **Conclusion:** We **reject the null hypothesis**. This indicates that the **mean number of rentals is significantly different across seasons**.

Key Insights

- Seasonality plays a **major role in rental behavior**.
- The difference in rental demand between seasons is **statistically significant and substantial**.
- Despite violations of normality and variance assumptions, the **large sample size makes ANOVA results robust and valid**.

Inferences & Conclusions

- Rentals are **not evenly distributed throughout the year**.
- Certain seasons (like monsoon or fall) likely see **higher or lower usage due to weather suitability, holidays, or commuting patterns**.
- The **season variable is a strong predictor** of demand and should be included in any forecasting model.

Recommendations

- **Strategic Resource Allocation:** Increase vehicle availability and maintenance staff during peak seasons.
- **Pricing/Promotions:** Use seasonal pricing strategies to **optimize revenue** in low-demand periods.
- **Forecasting & Planning:** Incorporate seasonality into **demand prediction models** for better accuracy.

DEPENDENCY BETWEEN SEASON AND WEATHER

STEP 1: HYPOTHESIS

- H_0 (Null): Weather and season are independent.
- H_1 (Alternate): There is a dependency between these two categories.

STEP 2: TEST TYPE:

- CHI-SQUARE TEST

STEP 3: SIGNIFICANCE LEVEL:

- Alpha(α) = 0.05 [95% Confidence Level]

STEP 4: ASSUMPTIONS:

- Observations are Independent.
- Data are Frequencies.

Crosstab - Contingency Table:

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

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

Chi-Square Test

```
print("CHI-SQUARE TEST TO CHECK THE DEPENDENCY BETWEEN SEASON AND WEATHER\n")

chi2, p_value, dof, expected = chi2_contingency(contingency_table)
print(f"Chi-Square Statistic: {chi2}\n")
print(f"P-value: {p_value}\n")
print(f"Degrees of Freedom: {dof}\n")
print("Expected Frequencies Table:")
print(expected)
print("-"*50)
```

```
if p_value > 0.05:
    print("FAILED TO REJECT H0: Weather and season are independent.")
else:
    print("REJECT H0: There is a dependency between these two categories.")
```

```
CHI-SQUARE TEST TO CHECK THE DEPENDENCY BETWEEN SEASON AND WEATHER

Chi-Square Statistic: 49.15865559689363

P-value: 1.5499250736864862e-07

Degrees of Freedom: 9

Expected Frequencies Table:
[[1.77454639e+03 1.80559765e+03 1.80559765e+03 1.80625831e+03]
 [6.99258130e+02 7.11493845e+02 7.11493845e+02 7.11754180e+02]
 [2.11948742e+02 2.15657450e+02 2.15657450e+02 2.15736359e+02]
 [2.46738931e-01 2.51056403e-01 2.51056403e-01 2.51148264e-01]]
--------------------------------------------------
REJECT H0: There is a dependency between these two categories.
```

KEY INSIGHTS:

Test Summary

- **Chi-Square Statistic:** 49.16
- **Degrees of Freedom:** 9
- **p-value:** ~1.55×10^{-7} (significantly below 0.05)
- **Decision:** We **reject the null hypothesis** of independence.

Key Insights

- There is a **statistically significant association between season and weather categories**.
- This means that certain types of weather are **more likely to occur in specific seasons**.
  - For example:
    - Weather category 1 (clear/few clouds) appears consistently across all seasons but is more common overall.
    - Weather category 3 (light rain/snow) has higher counts in winter and fall.
    - Category 4 (heavy rain or extreme conditions) is extremely rare but was observed in season 1.
- The **expected frequencies table** shows that actual counts deviate meaningfully from what would be expected under independence.

Inferences & Conclusions

- Season and weather are **not independent variables** in this dataset.
- This supports the idea that **season can be used to partially predict weather conditions**.
- Consequently, when modeling demand, including both season and weather without caution can lead to **redundancy or multicollinearity** because they share explanatory power.

Recommendations

- **Predictive Modeling:**
  - Consider carefully how you encode season and weather in forecasting models.
  - You might aggregate weather categories or include season as a hierarchical grouping to avoid overfitting.
- **Operational Planning:**
  - Use seasonal patterns to anticipate weather-related disruptions in service demand.
  - Prepare maintenance, fleet allocation, and customer communications in advance of predictable seasonal weather shifts.
- **Reporting and Strategy:**
  - Clearly document that weather is **season-dependent**, as this insight supports strategic planning and resource optimization throughout the year.

DEPENDENCY BETWEEN WORKING DAY AND WEATHER

```
ct2 = pd.crosstab(df['workingday'], df['weather'])
print(ct2)
print("-"*50)
print("CHI-SQUARE TEST TO CHECK THE DEPENDENCY BETWEEN WORKING DAY AND WEATHER\n")

chi2, p_value, dof, expected = chi2_contingency(ct2)
print(f"Chi-Square Statistic: {chi2}\n")
print(f"P-value: {p_value}\n")
print(f"Degrees of Freedom: {dof}\n")
print("Expected Frequencies Table:")
print(expected)
print("-"*50)
if p_value > 0.05:
    print("FAILED TO REJECT H0: Weather and Working Day are independent.")
else:
    print("REJECT H0: There is a dependency between these two categories.")
```

```
weather        1     2    3  4
workingday
0           2353   897  224  0
1           4839  1937  635  1
--------------------------------------------------
CHI-SQUARE TEST TO CHECK THE DEPENDENCY BETWEEN WORKING DAY AND WEATHER

Chi-Square Statistic: 16.162518725276595

P-value: 0.0010502165960627732

Degrees of Freedom: 3

Expected Frequencies Table:
[[2.29515047e+03 9.04401617e+02 2.74128789e+02 3.19125482e-01]
 [4.89684953e+03 1.92959838e+03 5.84871211e+02 6.80874518e-01]]
--------------------------------------------------
REJECT H0: There is a dependency between these two categories.
```

KEY INSIGHTS:

Test Summary

- **Chi-Square Statistic:** 16.16
- **Degrees of Freedom:** 3
- **p-value:** ~0.00105 (significantly less than 0.05)
- **Decision:** We **reject the null hypothesis** of independence.

Key Insights

- There is a **statistically significant association between working day status and weather conditions**.
- This means the distribution of weather categories is **not identical on working days vs. non-working days**.
- From the contingency table:
  - Clear weather (category 1) is more frequent on working days.
  - Adverse weather categories (2 and 3) are somewhat more frequent proportionally on non-working days.
  - Weather category 4 is extremely rare (only 1 observation), contributing minimal impact but included in the calculation.
- The **expected frequencies table** confirms the observed counts differ enough from expectations to be statistically significant.

Inferences & Conclusions

- Weather and working day are **not independent in this dataset**.
- There may be underlying factors causing this relationship, such as:
  - Seasonal patterns influencing both working days (e.g., holidays clustered in certain weather periods).
  - Data collection artifacts where non-working days coincide with worse weather conditions.
- This insight is important because it shows that **working day status partially captures weather variability** in demand.

Recommendations

- **Predictive Modeling:**
  - Exercise caution when including both working day and weather as independent variables to avoid redundancy or overfitting.
  - Consider interaction effects (e.g., how bad weather on working days might impact demand differently).
- **Operational Planning:**
  - Be prepared for potentially lower or higher rentals on non-working days, especially when weather is less favorable.
  - Align fleet readiness and staffing with expected weather and calendar patterns.
- **Reporting and Analysis:**
  - Clearly document this dependency as a limitation when interpreting the separate effects of working day and weather on rentals.

DEPENDENCY BETWEEN WORKING DAY AND SEASON

```
ct3 = pd.crosstab(df['workingday'], df['season'])
print(ct3)
print("-"*50)
print("CHI-SQUARE TEST TO CHECK THE DEPENDENCY BETWEEN WORKING DAY AND SEASON\n")

chi2, p_value, dof, expected = chi2_contingency(ct3)
print(f"Chi-Square Statistic: {chi2}\n")
print(f"P-value: {p_value}\n")
print(f"Degrees of Freedom: {dof}\n")
print("Expected Frequencies Table:")
print(expected)
print("-"*50)
if p_value > 0.05:
    print("FAILED TO REJECT H0: Season and Working Day are independent.")
else:
    print("REJECT H0: There is a dependency between these two categories.")
```

```
season         1     2     3     4
workingday
0            858   840   888   888
1           1828  1893  1845  1846
--------------------------------------------------
CHI-SQUARE TEST TO CHECK THE DEPENDENCY BETWEEN WORKING DAY AND SEASON

Chi-Square Statistic: 2.5708953973429574

P-value: 0.4626148207703564

Degrees of Freedom: 3

Expected Frequencies Table:
[[ 857.17104538  872.16994305  872.16994305  872.48906853]
 [1828.82895462 1860.83005695 1860.83005695 1861.51093147]]
--------------------------------------------------
FAILED TO REJECT H0: Season and Working Day are independent.
```

KEY INSIGHTS:

Test Summary

- **Chi-Square Statistic:** 2.57
- **Degrees of Freedom:** 3
- **p-value:** ~0.463 (greater than 0.05)
- **Decision:** We **fail to reject the null hypothesis**.

Key Insights

- There is **no statistically significant association** between season and working day status.
- The distribution of working days and non-working days is **consistent across all seasons**.
- The observed frequencies closely match the expected frequencies under independence, as reflected by the high p-value.

Inferences & Conclusions

- This result suggests that **the working calendar (holidays vs. working days) is spread fairly evenly across seasons**.

- Unlike weather, which showed dependencies with both working day and season, the calendar schedule does not show seasonal clustering.

Recommendations

- You can confidently treat **season and working day as independent variables** in modeling and analysis.
- There is no immediate need to adjust forecasts or operational strategies based on an interaction between season and working day.

```
1 Start coding or generate with AI.
```

FINAL BUSINESS INSIGHTS

1 Rentals Are Highly Time-Dependent

- Rentals strongly follow **hourly commuting patterns**, peaking sharply in the morning (around 8 AM) and evening (around 5–6 PM).
- Late-night rentals are minimal, confirming that most demand is driven by work-related trips.
- These hourly cycles are consistent across seasons and user types.

---

2 Seasonality Has a Major Influence

- Demand **increases progressively from Spring to Summer and peaks in Fall**, then drops in Winter.
- The **quarterly and monthly analysis** shows clear seasonality, with Q2 and Q3 being the most active periods.
- This suggests that usage is heavily linked to weather, daylight duration, and perhaps holiday schedules.

---

3 Weather Significantly Impacts Demand

- Clear weather days consistently show the **highest average rentals**.
- Rentals decline sharply during rainy or snowy weather (Weather Category 3).
- Severe weather (Category 4) almost completely suppresses usage.
- These patterns are observable in boxplots, histograms, and hypothesis tests.

---

4 Registered Users Drive the Majority of Rentals

- Registered users have consistently higher rentals compared to casual users in all time segments and weather conditions.
- Casual users show slightly higher participation on weekends and holidays.
- This split suggests that **commuters are Yulu's primary customer segment**, while casual usage is supplemental.

---

5 Working Day Effect

- Rentals on working days are **generally higher in the morning and evening**.
- Non-working days see **flatter rental patterns**, with a broader midday peak, reflecting leisure-oriented trips.
- Hypothesis tests indicate no significant difference in mean rentals between working and non-working days overall, but the hourly patterns differ substantially.

---

6 Temperature and Humidity Correlate with Usage

- Higher temperatures are associated with increased rentals, while very high humidity negatively correlates with usage.
- This aligns with expectations: comfortable weather encourages more trips.

---

7 Variable Interdependencies

- The data shows **strong correlations among certain predictors**:
  - Temperature and "feels like" temperature (~0.98).
  - Count and registered users (~0.97).
  - Season and month (~0.97).
- These relationships confirm that many predictors reinforce each other in describing demand.

---

8 Non-Normality of Demand Distribution

- Shapiro-Wilk and Q-Q plots consistently show that rental counts are **not normally distributed**.
- Distributions are right-skewed with a long tail of high-usage observations.
- This emphasizes the need for robust statistical models when forecasting demand.

---

9 Weather and Season Are Interconnected

- Chi-square tests confirmed that **weather conditions are not independent of season**.
- Certain adverse weather events cluster in specific seasons (e.g., rain in monsoon).
- This dependency should be accounted for in predictive modeling to avoid multicollinearity.

---

10 Year-on-Year Growth Indicates Market Adoption

- Comparing 2011 and 2012 shows a **large increase in rentals year over year**.
- This upward trend highlights growing customer acceptance and the scaling potential of Yulu's business model.

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

- **Hour of Day**
  - Clear hourly patterns: Low rentals overnight, sharp peaks at 8 AM and 5–6 PM, reflecting commute times.
  - Rentals can vary 10-fold between low-demand and high-demand hours.
- **Season and Month**
  - Rentals rise through Spring and Summer, peaking in Q2 and Q3 (April–September), then decline in Winter.
  - Clear evidence of seasonality, likely linked to weather and daylight.

- **Weather Conditions**
  - Clear weather drives the highest rentals; rentals drop steeply during rain, snow, or severe weather.
  - Over 70% of rentals occur in clear weather.
- **Working Day Indicator**
  - Rentals are higher on working days, supporting the commuting hypothesis.
  - Non-working days still show significant activity, mainly casual/leisure use.
- **Temperature and Perceived Temperature**
  - Warmer days correlate with higher rentals (correlation ~0.39).
  - Temperature influences comfort and ridership.
- **Humidity (negative impact)**
  - Higher humidity discourages rentals, likely due to discomfort.
  - Correlation with count: ~-0.32.
- **User Type (Casual vs Registered)**
  - Registered users are the primary contributors to demand.
  - Registered rentals strongly correlate with total rentals (~0.97).
- **Year (Growth Over Time)**
  - Significant increase in rentals from 2011 to 2012 (~65% growth), indicating strong adoption.

Less significant predictors:

- Windspeed (weak correlation ~0.10).
- Day of the Month (no clear influence).
- Day of the Week (moderate pattern, with slight increases midweek and weekends).

How well do these variables describe electric cycle demands?

- Variables like **hour, season, weather, temperature, and user type together explain most of the variability in demand.**
- Time-based variables (hour, month, season) align with operational patterns.
- Strong correlation structures:
  - **Registered users and count: 0.97**
  - **Temp and atemp: 0.98**
  - Season and month: ~0.97
- These high correlations suggest a significant portion of demand variability can be reliably modeled.
- While some variables are highly collinear, individually they provide clear, interpretable signals.

Summary: These variables offer a **robust framework for demand prediction**, enabling Yulu to anticipate peak usage periods, seasonal fluctuations, and the impact of weather conditions.

RECOMMENDATIONS

A) Fleet Planning and Operations

- **Dynamic Fleet Allocation:**
  - Deploy more vehicles during peak hours (8 AM, 5–6 PM) and high-demand months (April–September).
  - Reduce fleet density during overnight hours and in low-demand months (January–February) to optimize costs.
- **Weather-Responsive Strategies:**
  - Use weather forecasts to predict demand dips.
  - Offer promotions on days with mild adverse weather to maintain utilization.
- **Seasonal Maintenance Scheduling:**
  - Schedule major fleet maintenance in low-demand seasons (winter) to maximize availability during summer peaks.

B) Pricing and Promotions

- **Demand-Based Pricing:**
  - Implement surge pricing during high-demand hours and seasons to maximize revenue.
  - Introduce discounts during off-peak times (e.g., late nights, winter months) to smooth demand.
- **Targeted Incentives:**
  - Offer loyalty rewards to registered users, who account for most rentals.
  - Launch campaigns to attract more casual riders during weekends and holidays, where casual usage spikes.

C) Customer Engagement and Growth

- **Education and Awareness:**
  - Highlight benefits of using Yulu bikes even during off-peak seasons and less favorable weather through targeted communication.
- **Corporate Partnerships:**
  - Collaborate with offices and tech parks to create dedicated Yulu zones, as the majority of rentals are weekday commutes.
- **Weekend & Leisure Focus:**
  - Develop packages for weekend use and leisure trips to tap into casual users further.

D) Forecasting and Analytics

- **Predictive Modeling:**
  - Build time-series forecasting models leveraging:
    - Hour of day
    - Season/month
    - Weather forecast

        - Working day indicators
- **Scenario Planning:**
    - Simulate demand under different weather and season scenarios to guide staffing and fleet positioning.
- **Continuous Monitoring:**
    - Track adoption trends (e.g., growth observed from 2011 to 2012) to update operational strategies dynamically.

E) Operational Resilience

- **Contingency Planning:**
    - Develop plans for severe weather events (where rentals nearly cease) to prevent revenue shocks.
- **Diversification:**
    - Consider integrating other micro-mobility options (e.g., e-scooters) for resilience during seasonal downturns.

Concluding Remarks

Yulu has a **clear and predictable demand pattern driven by time of day, seasonality, weather conditions, and user segmentation**. By leveraging these insights:

- Demand can be forecasted with high confidence.
- Fleet and pricing can be optimized dynamically.
- Customer engagement can be tailored to boost utilization across user types and seasons.

By implementing these recommendations, Yulu can strengthen its market leadership, enhance user satisfaction, and sustainably grow revenue.

```
1 Start coding or generate with AI.
```

```
1 Start coding or generate with AI.
```

```
1 Start coding or generate with AI.
```

```
1 Start coding or generate with AI.
```

```
1 Start coding or generate with AI.
```