Yulu - Hypothesis Testing

July 21, 2024

1 Business Case: Yulu - Hypothesis Testing

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!

Yulu has recently suffered considerable dips in its revenues. They have contracted a consulting company to understand the factors on which the demand for these shared electric cycles depends. Specifically, they want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

Column Profiling:

- datetime: datetime
- season: season (1: spring, 2: summer, 3: fall, 4: winter)
- holiday: whether day is a holiday or not
- 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

[136]:
```
import pandas as pd
import numpy as np
```

```
from scipy.stats import stats
import matplotlib.pyplot as plt
import seaborn as sns
```

[137]:
```
data = pd.read_csv(r"C:\Users\n.rahman\OneDrive - BALADNA\Desktop\BALADNA\Ex␣
↪Docs\SCALER-DSML\Module 7 -Statistics\bike_sharing.csv")
data.sample(5)
```

[137]:
```
                 datetime  season  holiday  workingday  weather   temp  \
3717  2011-09-05 19:00:00       3        1           0        3  27.06
1645  2011-04-14 11:00:00       2        0           1        1  21.32
2297  2011-06-03 15:00:00       2        0           1        1  28.70
9513  2012-09-19 18:00:00       3        0           1        1  23.78
8921  2012-08-14 02:00:00       3        0           1        2  27.88

       atemp  humidity  windspeed  casual  registered  count
3717  29.545        94     7.0015      52         123    175
1645  25.000        45    11.0014      28          87    115
2297  31.820        28    19.0012      56         156    212
9513  27.275        40    19.0012      85         807    892
8921  31.820        83    12.9980       2           9     11
```

1.1 Exploratory Data Analysis

[138]:
```
data.shape
```

[138]:
```
(10886, 12)
```

[139]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  object
 1   season      10886 non-null  int64
 2   holiday     10886 non-null  int64
 3   workingday  10886 non-null  int64
 4   weather     10886 non-null  int64
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
```

```
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

[140]:
```
data.describe()
```

[140]:

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

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

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

1.1.1 Missing Values & Duplicates Check

[141]:
```
missing_value = pd.DataFrame({"Missing_Values":data.isnull().sum(),"Percentage":
↪(data.isnull().sum()/len(data))*100})
missing_value
```

[141]:

	Missing_Values	Percentage
datetime	0	0.0
season	0	0.0
holiday	0	0.0
workingday	0	0.0
weather	0	0.0

```
temp                        0          0.0
atemp                       0          0.0
humidity                    0          0.0
windspeed                   0          0.0
casual                      0          0.0
registered                  0          0.0
count                       0          0.0
```

[269]:
```python
data.duplicated().sum() #no duplicates
```

[269]:
```
0
```

1.1.2 Datatype conversions of attributes

[142]:
```python
data["season"].value_counts()
```

[142]:
```
season
4    2734
2    2733
3    2733
1    2686
Name: count, dtype: int64
```

[143]:
```python
data["holiday"].value_counts()
```

[143]:
```
holiday
0    10575
1      311
Name: count, dtype: int64
```

[144]:
```python
data["workingday"].value_counts()
```

[144]:
```
workingday
1    7412
0    3474
Name: count, dtype: int64
```

[145]:
```python
data["weather"].value_counts()
```

[145]:
```
weather
1    7192
2    2834
3     859
4       1
Name: count, dtype: int64
```

[146]:
```python
data["datetime"] = pd.to_datetime(data["datetime"])
data["season"] = data["season"].astype("category")
```

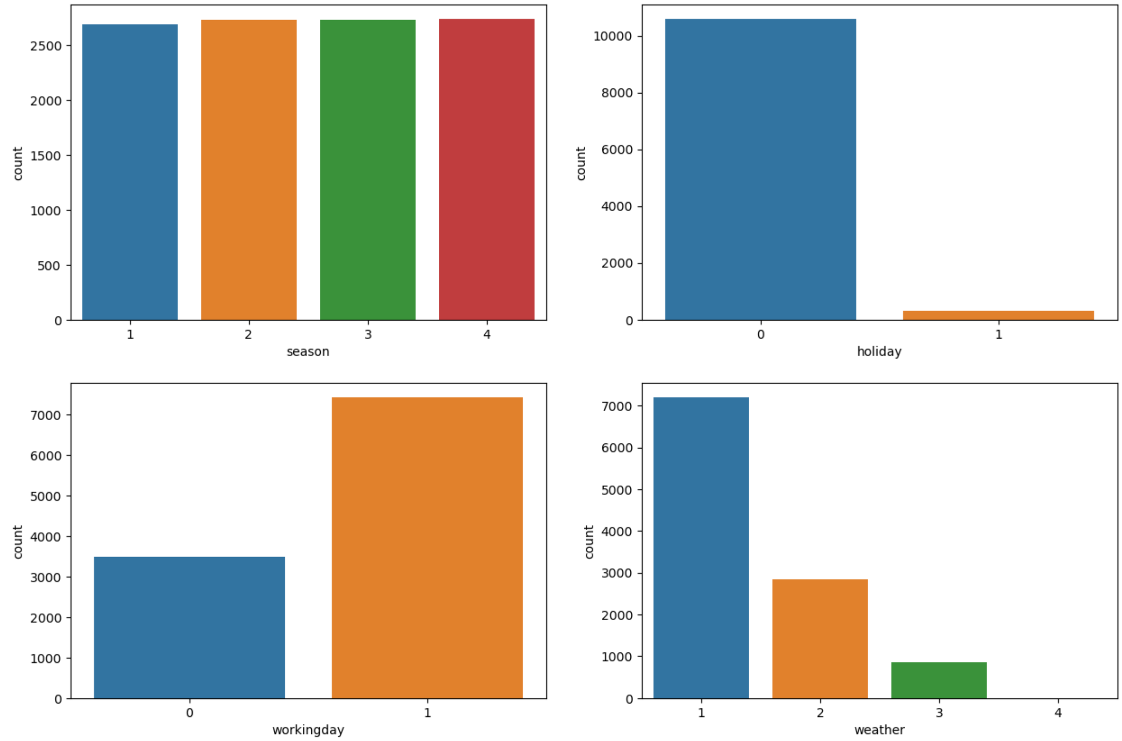
```
data["holiday"] = data["holiday"].astype("category")
data["workingday"] = data["workingday"].astype("category")
data["weather"] = data["weather"].astype("category")
```

[147]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  category
 2   holiday     10886 non-null  category
 3   workingday  10886 non-null  category
 4   weather     10886 non-null  category
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
dtypes: category(4), datetime64[ns](1), float64(3), int64(4)
memory usage: 723.7 KB
```

1.1.3 Univariate Analysis

[148]:
```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))
sns.countplot(data=data,x="season", ax=axs[0,0])
sns.countplot(data=data,x="holiday",ax = axs[0,1])
sns.countplot(data=data,x="workingday",ax=axs[1,0])
sns.countplot(data=data,x="weather",ax=axs[1,1])
plt.show()
```



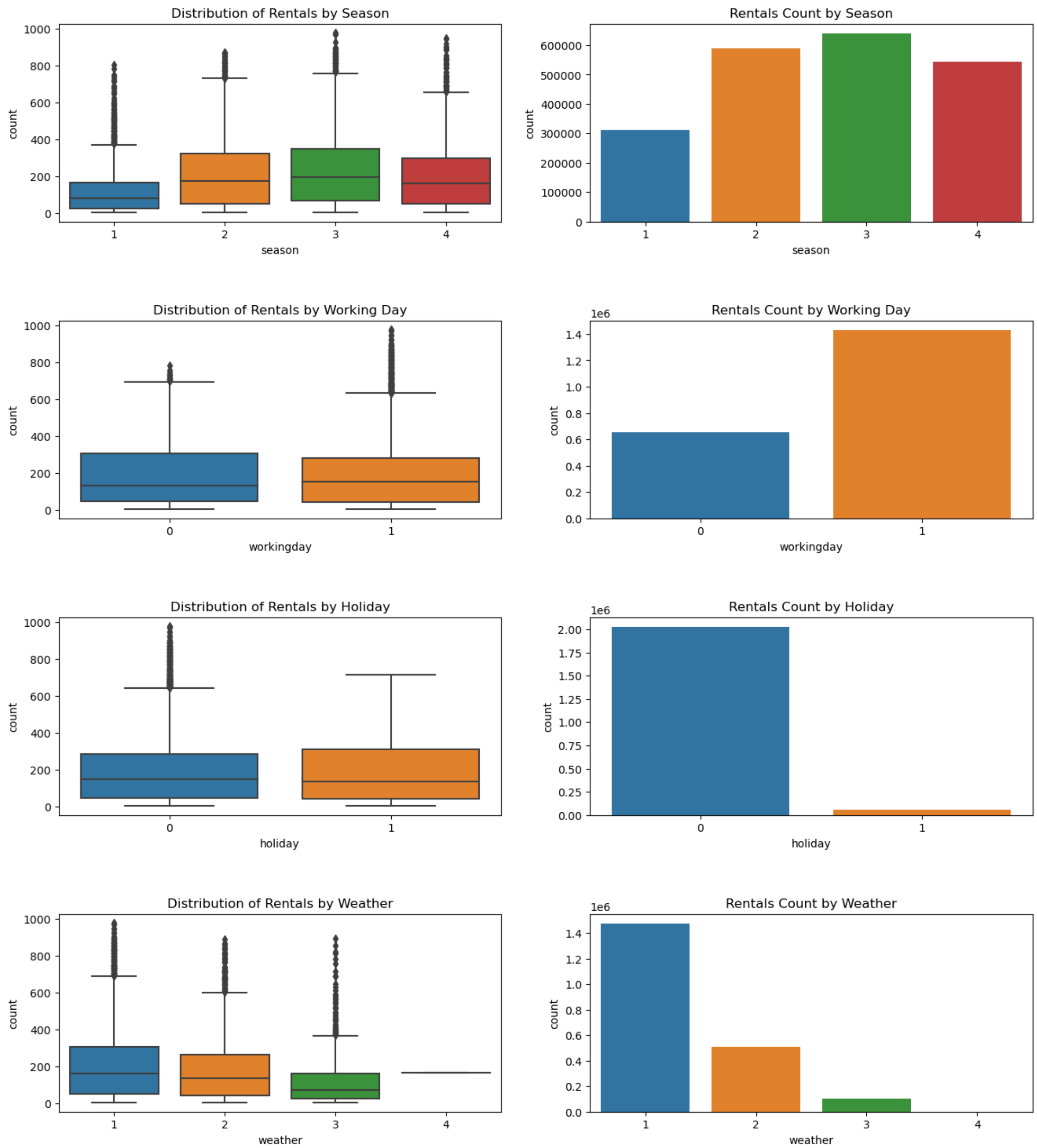
Observations

- For all four seasons, the number of transactions seems to be nearly equal; however, we need to examine the rental counts and impacts.
- During holidays, rentals are minimal compared to non-holidays, which is expected.
- On working days, rental transactions are higher compared to non-holidays, which is logical.
- Weather count plots indicate that most rentals occur during clear and cloudy weather (1) compared to other weather periods, while during heavy rain (4), only one rental occurred. So definitely there seems to be some patterns in rentals during weather.

1.1.4 Bivariate Analysis

[150]:
```
fig, axs = plt.subplots(nrows=4, ncols=2, figsize=(16,18))

#first row
sns.boxplot(x=data["season"],y=data["count"], ax = axs[0,0]).
  ↪set_title("Distribution of Rentals by Season")
sns.barplot(x="season",y="count",data=data.groupby("season")["count"].sum().
  ↪reset_index(), ax=axs[0,1]).set_title("Rentals Count by Season")

#second row
sns.boxplot(x=data["workingday"],y=data["count"], ax=axs[1,0]).
  ↪set_title("Distribution of Rentals by Working Day")
```

```
sns.barplot(x="workingday",y="count",data=data.groupby("workingday")["count"].
    ↪sum().reset_index(), ax=axs[1,1]).set_title("Rentals Count by Working Day")


#third row
sns.boxplot(x=data["holiday"],y=data["count"], ax=axs[2,0]).
    ↪set_title("Distribution of Rentals by Holiday")
sns.barplot(x="holiday",y="count",data=data.groupby("holiday")["count"].sum().
    ↪reset_index(), ax=axs[2,1]).set_title("Rentals Count by Holiday")

#fourth row
sns.boxplot(x=data["weather"],y=data["count"], ax=axs[3,0]).
    ↪set_title("Distribution of Rentals by Weather")
sns.barplot(x="weather",y="count",data=data.groupby("weather")["count"].sum().
    ↪reset_index(), ax=axs[3,1]).set_title("Rentals Count by Weather")

fig.subplots_adjust(hspace=0.5)
plt.show()
```



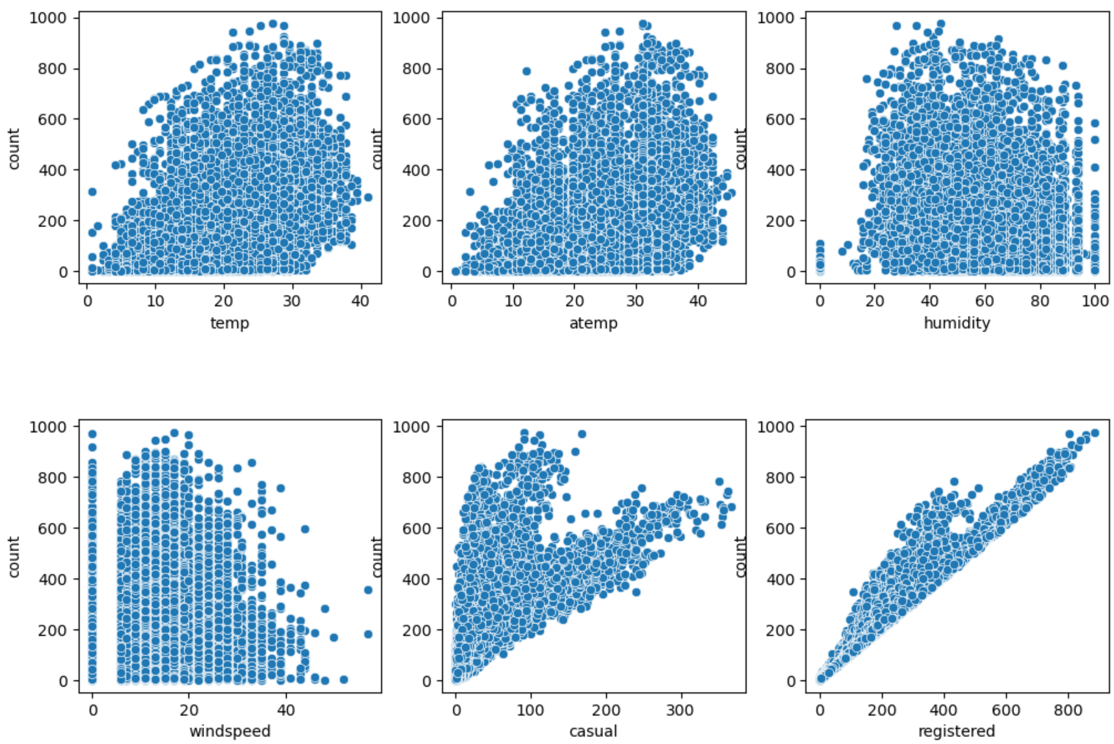
Observations

- From the season boxplot and bar chart, it is evident that rental counts peak during summer and the fall season and starts drop during winter with a notable decline in spring as well. Each season exhibits outliers that requires further investigation.
- The working day plot clearly indicates higher rental activity on working days compared to non-working days. The boxplot shows a greater number of outliers on working days.
- Rental activity is significantly higher on non-holidays which is expected and the data contains numerous outliers that need further examination.
- Analysis of weather conditions reveals that rentals are absent during heavy rain, with outliers

present in all other weather conditions. The majority of rentals occur during clear and cloudy weather, highlighting the substantial influence of weather and season on Yulu rental's sales and revenue.

1.1.5 Correlation between features

[375]:
```
fig, axs = plt.subplots(nrows=2, ncols=3, figsize=(12,8))

#first row
sns.scatterplot(data=data,x = data["temp"],y= data["count"],ax=axs[0,0])
sns.scatterplot(data=data,x = data["atemp"],y= data["count"],ax=axs[0,1])
sns.scatterplot(data=data,x = data["humidity"],y= data["count"],ax=axs[0,2])

#second row
sns.scatterplot(data=data,x = data["windspeed"],y= data["count"],ax=axs[1,0])
sns.scatterplot(data=data,x = data["casual"],y= data["count"],ax=axs[1,1])
sns.scatterplot(data=data,x = data["registered"],y= data["count"],ax=axs[1,2])

fig.subplots_adjust(hspace=0.5)
plt.show()
```



[391]:
```
cr = data[["temp","atemp","humidity","windspeed","casual","registered","count"]]
cr.corr(method = "spearman")
```

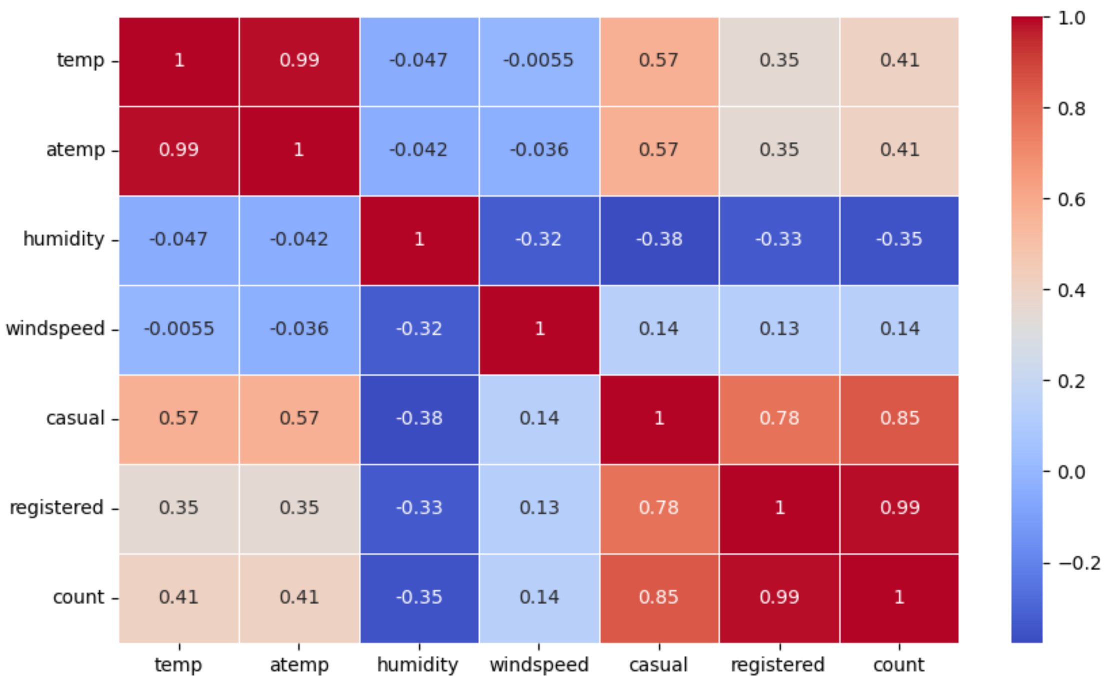
[391]:

```
                temp     atemp  humidity  windspeed    casual  registered  \
temp        1.000000  0.987128 -0.046854  -0.005535  0.573034    0.352174
atemp       0.987128  1.000000 -0.042028  -0.036350  0.571588    0.350577
humidity   -0.046854 -0.042028  1.000000  -0.324447 -0.378254   -0.332785
windspeed  -0.005535 -0.036350 -0.324447   1.000000  0.135040    0.131011
casual      0.573034  0.571588 -0.378254   0.135040  1.000000    0.775785
registered  0.352174  0.350577 -0.332785   0.131011  0.775785    1.000000
count       0.407989  0.406562 -0.354049   0.135777  0.847378    0.988901

               count
temp        0.407989
atemp       0.406562
humidity   -0.354049
windspeed   0.135777
casual      0.847378
registered  0.988901
count       1.000000
```

[398]:

```
plt.figure(figsize=(10,6))
sns.heatmap(cr.corr(method="spearman"),annot=True, cmap="coolwarm",linewidth=0.
 ↪5)
```

[398]: <Axes: >



Observations

- Positive Correlation: the correlation between rental_count and registered users is 0.99, it means that as the registered useres increases, the rental count also tends to increase.they follow a linear relation.
- Negative Correlation: the correlation between rental_count and humidity is slighlty negative correlated -0.35, it means that as the humidity increases, the rental count tends to decrease.
- Weak/No Correlation: Other features suggests that there's no strong linear relationship with rental counts

1.1.6 Rentals by month/ year

[151]:
```
data["datetime"].min()
```

[151]:
```
Timestamp('2011-01-01 00:00:00')
```

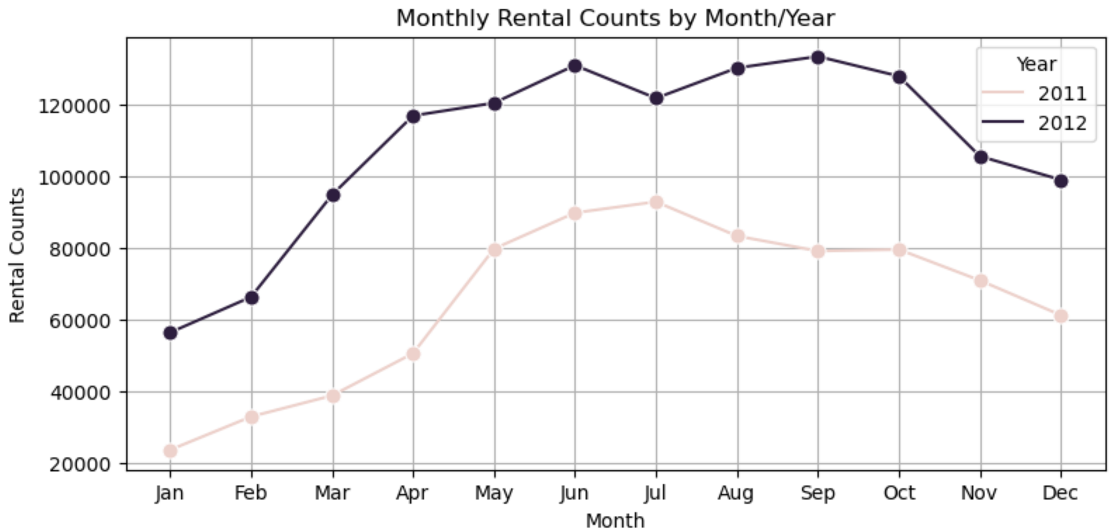
[152]:
```
data["datetime"].max()
```

[152]:
```
Timestamp('2012-12-19 23:00:00')
```

[153]:
```
data["year"] = data["datetime"].dt.year
data["month"] = data["datetime"].dt.month
```

[154]:
```
monthly_rentals = data.groupby(["year","month"])["count"].sum().reset_index()
```

[155]:
```
plt.figure(figsize=(9,4))
sns.lineplot(data=monthly_rentals,x="month",y="count",hue="year",marker='o',␣
 ↪markersize=8)
plt.title('Monthly Rental Counts by Month/Year')
plt.xlabel('Month')
plt.ylabel('Rental Counts')
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul',␣
 ↪'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
plt.grid(True)
plt.legend(title='Year')
plt.show()
```



Observation

- The timeline line chart reveals a clear monthly rental pattern over the two-year period.
- Although the number of rentals varies significantly, we focus on the overall trend in rental activity over the months rather than the exact counts.
- For both 2011 and 2012, the rental trends follow a consistent pattern from November to June, despite differences in rental numbers. Rentals decreased from July to November in 2011, whereas they increased during the same period in 2012, with a slight drop in July.May be more users have rented during this period picking up the same. Requires more data or further analysis.
- The months with rental drops correspond possibly to winter, with rentals starting to increase in spring and peaking in summer.

1.2 Outliers Detection

[156]:
```
data.describe(include="all")
```

[156]:
```
                             datetime   season   holiday  workingday  weather  \
count                           10886  10886.0   10886.0     10886.0  10886.0
unique                            NaN      4.0       2.0         2.0      4.0
top                               NaN      4.0       0.0         1.0      1.0
freq                              NaN   2734.0   10575.0      7412.0   7192.0
mean    2011-12-27 05:56:22.399411968      NaN       NaN         NaN      NaN
min               2011-01-01 00:00:00      NaN       NaN         NaN      NaN
25%               2011-07-02 07:15:00      NaN       NaN         NaN      NaN
50%               2012-01-01 20:30:00      NaN       NaN         NaN      NaN
75%               2012-07-01 12:45:00      NaN       NaN         NaN      NaN
max               2012-12-19 23:00:00      NaN       NaN         NaN      NaN
std                               NaN      NaN       NaN         NaN      NaN
```

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

          registered         count          year         month
count   10886.000000  10886.000000  10886.000000  10886.000000
unique           NaN           NaN           NaN           NaN
top              NaN           NaN           NaN           NaN
freq             NaN           NaN           NaN           NaN
mean      155.552177    191.574132   2011.501929      6.521495
min         0.000000      1.000000   2011.000000      1.000000
25%        36.000000     42.000000   2011.000000      4.000000
50%       118.000000    145.000000   2012.000000      7.000000
75%       222.000000    284.000000   2012.000000     10.000000
max       886.000000    977.000000   2012.000000     12.000000
std       151.039033    181.144454      0.500019      3.444373
```

Season

[157]:
```
data["season"].unique()
```

[157]:
```
[1, 2, 3, 4]
Categories (4, int64): [1, 2, 3, 4]
```

[248]:
```
for i in data["season"].unique():
    season_data = data.loc[data["season"] == i]["count"].reset_index()
    Q1 = np.percentile(season_data["count"],25)
    Q3 = np.percentile(season_data["count"],75)
    IQR = Q3-Q1
    upper_bound = Q3 + 1.5 * IQR

    outliers = np.round(len(season_data.loc[season_data["count"]>upper_bound])/
 ↪len(season_data)*100,2)

    print("Season-",i,"25th Percentile=",Q1 ,"75th␣
 ↪Percentile=",Q3,"IQR=",IQR,"Upper_Whisker=",upper_bound,"% of␣
 ↪Outliers=",outliers)
```

```
Season- 1 25th Percentile= 24.0 75th Percentile= 164.0 IQR= 140.0 Upper_Whisker=
374.0 % of Outliers= 5.17
Season- 2 25th Percentile= 49.0 75th Percentile= 321.0 IQR= 272.0 Upper_Whisker=
729.0 % of Outliers= 1.54
Season- 3 25th Percentile= 68.0 75th Percentile= 347.0 IQR= 279.0 Upper_Whisker=
765.5 % of Outliers= 2.23
Season- 4 25th Percentile= 51.0 75th Percentile= 294.0 IQR= 243.0 Upper_Whisker=
658.5 % of Outliers= 2.34
```

Working Day

[250]:
```
data["workingday"].unique()
```

[250]:
```
[0, 1]
Categories (2, int64): [0, 1]
```

[252]:
```
for i in data["workingday"].unique():
    wd_data = data.loc[data["workingday"] == i]["count"].reset_index()
    Q1 = np.percentile(wd_data["count"],25)
    Q3 = np.percentile(wd_data["count"],75)
    IQR = Q3-Q1
    upper_bound = Q3 + 1.5 * IQR

    outliers = np.round(len(wd_data.loc[wd_data["count"]>upper_bound])/
 ↪len(wd_data)*100,2)

    print("Working Day-",i, "25th Percentile=",Q1 ,"75th␣
 ↪Percentile=",Q3,"IQR=",IQR,"Upper_Whisker=",upper_bound,"% of␣
 ↪Outliers=",outliers)
```

```
Working Day- 0 25th Percentile= 44.0 75th Percentile= 304.0 IQR= 260.0
Upper_Whisker= 694.0 % of Outliers= 0.46
Working Day- 1 25th Percentile= 41.0 75th Percentile= 277.0 IQR= 236.0
Upper_Whisker= 631.0 % of Outliers= 3.75
```

Holiday

[254]:
```
data["holiday"].unique()
```

[254]:
```
[0, 1]
Categories (2, int64): [0, 1]
```

[255]:
```
for i in data["holiday"].unique():
    hd_data = data.loc[data["holiday"] == i]["count"].reset_index()
    Q1 = np.percentile(hd_data["count"],25)
    Q3 = np.percentile(hd_data["count"],75)
    IQR = Q3-Q1
    upper_bound = Q3 + 1.5 * IQR
```

```
    outliers = np.round(len(hd_data.loc[hd_data["count"]>upper_bound])/
↪len(hd_data)*100,2)

    print("Holiday-",i, "25th Percentile=",Q1 ,"75th␣
↪Percentile=",Q3,"IQR=",IQR,"Upper_Whisker=",upper_bound,"% of␣
↪Outliers=",outliers)
```

```
Holiday- 0 25th Percentile= 43.0 75th Percentile= 283.0 IQR= 240.0
Upper_Whisker= 643.0 % of Outliers= 2.94
Holiday- 1 25th Percentile= 38.5 75th Percentile= 308.0 IQR= 269.5
Upper_Whisker= 712.25 % of Outliers= 0.0
```

Weather

[257]:
```
data["weather"].unique()
```

[257]:
```
[1, 2, 3, 4]
Categories (4, int64): [1, 2, 3, 4]
```

[259]:
```
for i in data["weather"].unique():
    we_data = data.loc[data["weather"] == i]["count"].reset_index()
    Q1 = np.percentile(we_data["count"],25)
    Q3 = np.percentile(we_data["count"],75)
    IQR = Q3-Q1
    upper_bound = Q3 + 1.5 * IQR

    outliers = np.round(len(we_data.loc[we_data["count"]>upper_bound])/
↪len(we_data)*100,2)

    print("Weather-",i, "25th Percentile=",Q1 ,"75th␣
↪Percentile=",Q3,"IQR=",IQR,"Upper_Whisker=",upper_bound,"% of␣
↪Outliers=",outliers)
```

```
Weather- 1 25th Percentile= 48.0 75th Percentile= 305.0 IQR= 257.0
Upper_Whisker= 690.5 % of Outliers= 2.22
Weather- 2 25th Percentile= 41.0 75th Percentile= 264.0 IQR= 223.0
Upper_Whisker= 598.5 % of Outliers= 2.89
Weather- 3 25th Percentile= 23.0 75th Percentile= 161.0 IQR= 138.0
Upper_Whisker= 368.0 % of Outliers= 6.52
Weather- 4 25th Percentile= 164.0 75th Percentile= 164.0 IQR= 0.0 Upper_Whisker=
164.0 % of Outliers= 0.0
```

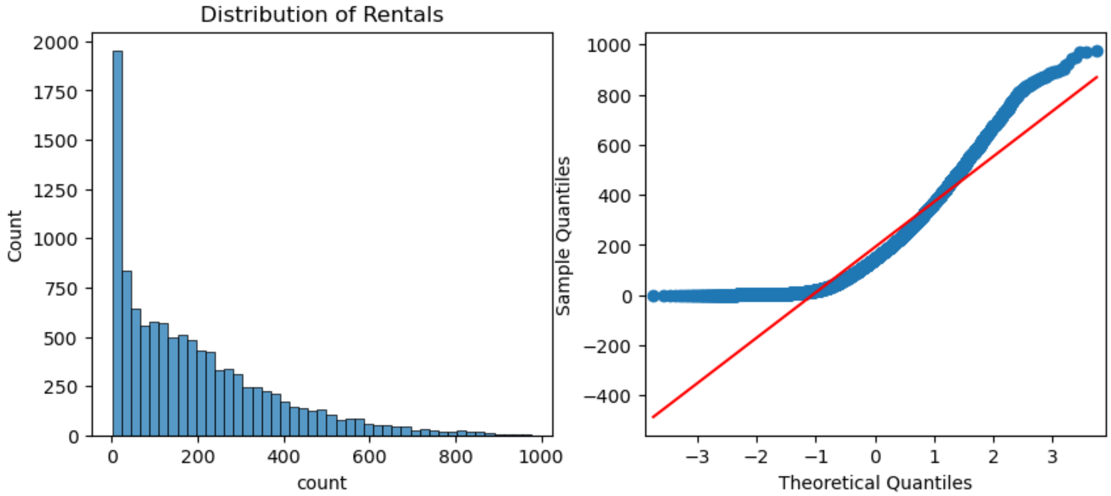
Observations

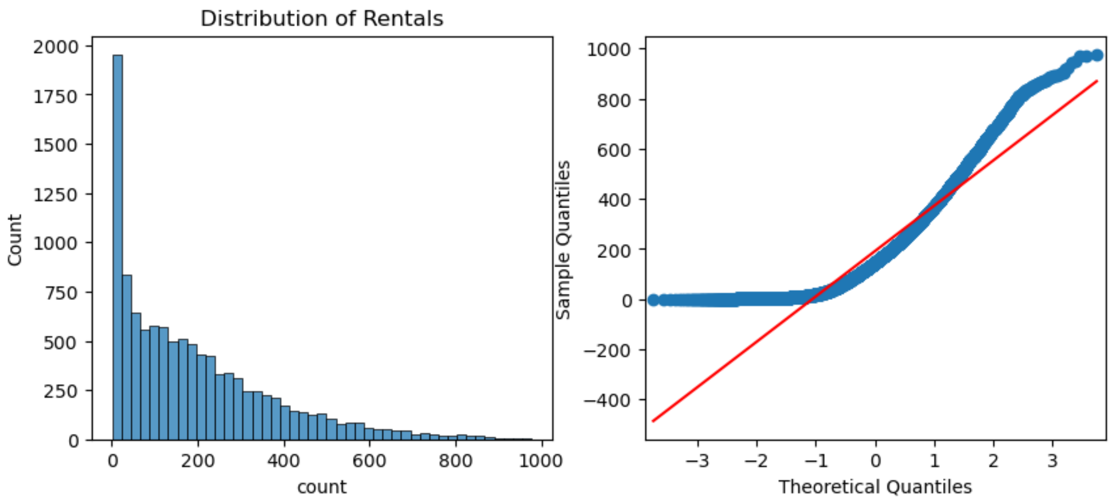
- For this dataset I will retain all the outliers among all the attributes as outliers can contain valuable information and their removal can sometimes lead to misleading or inaccurate results in analysis.

1.2.1 Test for Normality

[356]:
```python
from statsmodels.graphics.gofplots import qqplot
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(10,4))

#first row
sns.histplot(data["count"], ax = axs[0]).set_title("Distribution of Rentals")
qqplot(data["count"],line="s", ax=axs[1])
```

[356]:





Observation - In short, rental counts are not normally distributed, its right skewed or positve skewed. Evident from QQplot as well its non-normally distributed.

2 Hypothesis Testing

[400]:
```
#importing libraries

from scipy.stats import␣
↪ttest_ind,chisquare,chi2,f_oneway,mannwhitneyu,chi2_contingency
```

2.0.1 To check if Working Day has an effect on the number of electric cycles rented

Using 2 Sample T-test

[401]:
```
df = data[["workingday","count"]]
df.groupby("workingday")["count"].mean()
```

[401]:
```
workingday
0    188.506621
1    193.011873
Name: count, dtype: float64
```

#framing the hypothesis

- H0: There is no significant difference in rentals between the working & non working days
- Ha: There is significant difference in rentals between the working & non working days

[402]:
```
df_working = df.loc[df["workingday"]==1]["count"]
df_notworking = df.loc[df["workingday"]==0]["count"]
```

[430]:
```
alpha =0.05 #significance value
```

[431]:
```
#appylying 2 sample t-test
t_stat, pvalue = ttest_ind(df_working, df_notworking, alternative = "two-sided")
t_stat,pvalue
```

[431]:
```
(1.2096277376026694, 0.22644804226361348)
```

[433]:
```
if pvalue<alpha:
    print("Reject the null hypothesis, There is significant difference in mean␣
↪rentals between working and non-working days")
else:
    print("Fail to Reject the null hypothesis, There is no significant␣
↪difference in mean rentals between working and non-working days")
    print("We dont have sufficient evidence to say that working day had impact␣
↪on bike rentals")
```

```
Fail to Reject the null hypothesis, There is no significant difference in mean
rentals between working and non-working days
We dont have sufficient evidence to say that working day had impact on bike
rentals
```

Using Mann-Whitney U test as the assumption of normality is violated to use 2 sample t-test

[406]:
```
stat, p_value = mannwhitneyu(df_working,df_notworking)
stat,p_value
```

[406]:
```
(12868495.5, 0.9679139953914079)
```

[407]:
```
if p_value<alpha:
    print("Reject the null hypothesis, There is significant difference in␣
 ↪rentals between working and non-working days")
else:
    print("Fail to Reject the null hypothesis, There is no significant␣
 ↪difference in rentals between working and non-working days")
    print("We dont have sufficient evidence to say that working day had impact␣
 ↪on bike rentals")
```

```
Fail to Reject the null hypothesis, There is no significant difference in
rentals between working and non-working days
We dont have sufficient evidence to say that working day had impact on bike
rentals
```

[302]:
```
data.groupby("workingday")["count"].sum()
```

[302]:
```
workingday
0     654872
1    1430604
Name: count, dtype: int64
```

[303]:
```
data.groupby("workingday")["count"].mean()
```

[303]:
```
workingday
0    188.506621
1    193.011873
Name: count, dtype: float64
```

2.0.2 To check if No. of cycles rented is similar or different in different weather conditions using One Way ANOVA test

#framing the hypothesis

- H0: No difference in cycle rentals during different weather conditions
- Ha: There is differrnce in cycle rentals during difference weather conditions

[307]:
```
data["weather"].value_counts()
```

[307]:
```
weather
1    7192
2    2834
3     859
```

```
4        1
Name: count, dtype: int64
```

[319]:
```
w1 = data.loc[data["weather"]==1]["count"]
w2 = data.loc[data["weather"]==2]["count"]
w3 = data.loc[data["weather"]==3]["count"]
w4 = data.loc[data["weather"]==4]["count"]
```

[323]:
```
f_stats, p_value = f_oneway(w1,w2,w3,w4)
f_stats,p_value
```

[323]:
```
(65.53024112793271, 5.482069475935669e-42)
```

[326]:
```
alpha =0.05
if p_value<alpha:
    print("Reject the null hypothesis, There is significant difference in␣
 ↪rentals on different weather conditions")
else:
    print("Fail to Reject the null hypothesis, There is no significant␣
 ↪difference in rentals on different weather conditions")
```

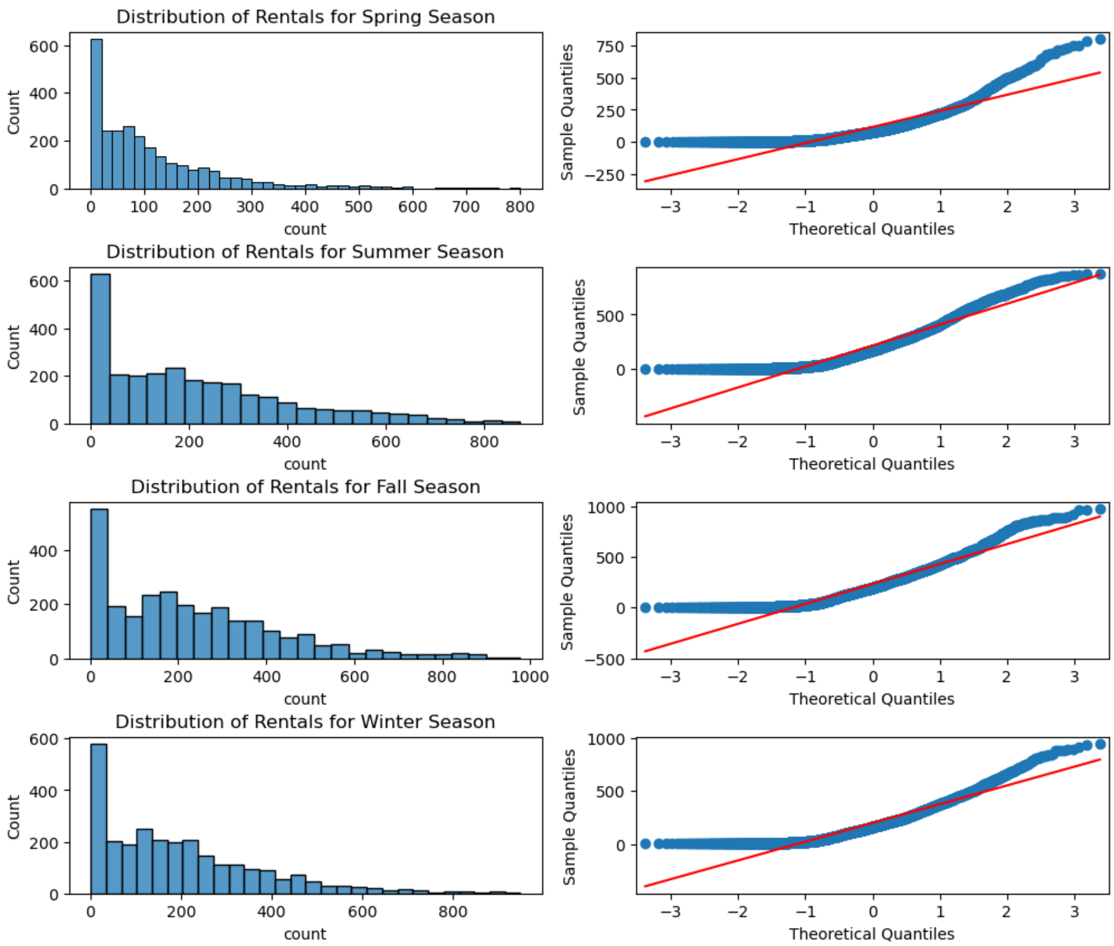
```
Reject the null hypothesis, There is significant difference in rentals on
different weather conditions
```

2.0.3 To check if No. of cycles rented is similar or different in different seasons using One Way ANOVA test

[327]:
```
s1 = data.loc[data["season"]==1]["count"]
s2 = data.loc[data["season"]==2]["count"]
s3 = data.loc[data["season"]==3]["count"]
s4 = data.loc[data["season"]==4]["count"]
```

[342]:
```
#checking the assumptions

fig, axs = plt.subplots(nrows=4, ncols=2, figsize=(12,10))

#first row
sns.histplot(s1, ax = axs[0,0]).set_title("Distribution of Rentals for Spring␣
 ↪Season")
qqplot(s1,line="s", ax=axs[0,1])


#second row
sns.histplot(s2, ax = axs[1,0]).set_title("Distribution of Rentals for Summer␣
 ↪Season")
qqplot(s2,line="s", ax=axs[1,1])
```

```
#third row
sns.histplot(s3, ax = axs[2,0]).set_title("Distribution of Rentals for Fall␣
↪Season")
qqplot(s3,line="s", ax=axs[2,1])

#fourth row
sns.histplot(s4, ax = axs[3,0]).set_title("Distribution of Rentals for Winter␣
↪Season")
qqplot(s4,line="s", ax=axs[3,1])

fig.subplots_adjust(hspace=0.5)
plt.show()
```



[328]:
```
f_stats, p_value = f_oneway(s1,s2,s3,s4)
f_stats,p_value
```

[328]: (236.94671081032106, 6.164843386499654e-149)

[329]:
```
alpha =0.05
if p_value<alpha:
    print("Reject the null hypothesis, There is significant difference in␣
 ↪rentals on different seasons")
else:
    print("Fail to Reject the null hypothesis, There is no significant␣
 ↪difference in rentals on different seasons")
```

```
Reject the null hypothesis, There is significant difference in rentals on
different seasons
```

2.0.4 Check if the Weather conditions are significantly different during different Seasons?

Using Chi-Square Contingency Test #framing the hypothesis

- H0: Weather and Season are independent
- Ha: Weather and Season are not independent

[361]:
```
crosstab = pd.crosstab(data["weather"],data["season"])
crosstab
```

[361]:

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

[364]:
```
chi_stat, p_value, df, exp_freq = chi2_contingency(crosstab) # chi_stat,␣
 ↪p_value, df, expected value

print("chi_stat:",chi_stat)
print("p_value:",p_value)
print("df:",df)
print("exp_freq:",exp_freq)
```

```
chi_stat: 49.15865559689363
p_value: 1.5499250736864862e-07
df: 9
exp_freq: [[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]]
```

[365]:
```
alpha = 0.05

if p_value < alpha:
    print("Reject H0")
```

```
    print("Weather and Season are dependent")
else:
    print("Fail to reject H0")
    print("Weather and Season are not dependent")
```

```
Reject H0
Weather and Season are dependent
```

Recommendations

- Introduce special promotions and discounts during summer and fall when rentals peak to capitalize on high demand and further increase revenue.
- Develop targeted marketing campaigns for winter months to mitigate the drop in rentals. Offer winter deals such as discounted long term rentals and accessories like disposible rain coats.
- Given the higher rental activity on working days, consider introducing weekday specific packages or subscriptions for daily commuters to boost regular usage.
- Make dynamic pricing that adjusts based on weather conditions, offering discounts on clear and cloudy days to maximize rentals.
- Will have to conduct a detailed analysis of outliers to understand unusual rental behaviors and identify opportunities service improvements. Need more data like demographics, city etc.
- Enhance the customer experience by ensuring bikes are well-maintained, easily accessible, and equipped with necessary accessories e.g. helmets, phone holders, phone charger.
- Use the insights from the timeline line chart to plan and allocate resources effectively throughout the year, ensuring adequate bike availability and operational support during peak and off-peak months.
- Develop and deploy a machine learning model to predict rental demand based on historical data, weather conditions, seasonal trends, holidays, and other relevant factors.
- Make deal with food delivery partners to make use of Yulu bikes equipped with all necessary accessories to preserve the food quality and delivery on time.

[ ]:

```

```