EEET2485 - Research Methods for Engineers

Group Assignment

E-Scooter Stations Analysis

Students:

Tran Khai Minh (s3818343)

Nguyen Huy Hoang (s3764704)

Nguyen Nhat Tan (s3818559)

Tong Son Tung (s3818153)

Nguyen Thanh Loan (s3821185)

Group Number: 2

Assigned Dataset: 2

Lecturer: Dr. Arthur Tang

Due Date: January 13, 2023

Table of Content

---

1. Data Preparation

1.1 Introduction

The dataset is a record from 24 e-scooter rental stations from Winter 2017 to Autumn 2018. Each entry in the record includes the number of different classes of users, together with the temperature of each day. The analysis of this dataset will give insights into e-scooter rentees' behavior as well as the factors affecting the operation of the e-scooter stations.

1.2 Research Questions (RQs)

> - ***RQ1: Which weather factor(s) most likely affect the number of e-scooter rentees?***
> - ***RQ2: Is there a relationship between temperature and dew point temperature?***
> - ***RQ3: Which season are people most/least likely to rent an e-scooter?***
> - ***RQ4: Which station has the most/least e-scooter rent in a single day?***
> - ***RQ5: Is season a factor for e-scooter station's closure?***
> - ***RQ6: Are people more likely to rent an e-scooter when the temperature is above 0°C?***
> - ***RQ7: Are people more likely to rent an e-scooter on completely dry days (no rain, no snow)?***
> - ***RQ8: Has the percentage of registered/newly registered user increase after 6 months (from Dec 2017 to Jun 2018)?***
> - ***RQ9: Do unregistered rentees prefer to register or stay casual?***

1.3 Importing neccessary libraries

In [339…

```
# Turning off warninng
from IPython.display import HTML
HTML('''<script>
code_show_err=false;
function code_toggle_err() {
 if (code_show_err){
 $('div.output_stderr').hide();
 } else {
 $('div.output_stderr').show();
 }
 code_show_err = !code_show_err
}
$( document ).ready(code_toggle_err);
```

```
</script>
To toggle on/off output_stderr, click <a href="javascript:code_toggle_err()">here</a>.''
```

Out[339]: To toggle on/off output_stderr, click here.

In [340…

```
import sys
!{sys.executable} -m pip -q install pingouin

# Import pandas and numpy libraries
import pandas as pd
import numpy as np

# Scipy stats library for statistical tests (Pearson R, t-test, ANOVA, chi-square, Leven
import scipy
import scipy.stats as stats

# Library for Welch's ANOVA and Games-Howell post-hoc tests
import pingouin as pg

# Libraries for plotting
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.patches as mpatches

# Setting the figure size
plt.rcParams["figure.figsize"]= 5, 5
# Setting theme in seaborn
sns.set_theme(style="ticks", color_codes=True)

import warnings ## importing warnings library.
warnings.filterwarnings('ignore') ## Ignore warning
```

Checking software version

In [341…

```
# check the version of the packages
! python --version
print("Numpy version: ", np.__version__)
print("Pandas version: ",pd.__version__)
print("Scipy version: ", scipy.__version__)
```

```
Python 3.9.13
Numpy version:  1.21.5
Pandas version:  1.4.4
Scipy version:  1.9.1
```

1.4 Importing the dataset

In [342…

```
df = pd.read_excel("dataset2.xlsx")

df.columns = df.columns.str.replace(' ', '') # Strip whitespaces

print("The shape of the ORIGINAL data is (row, column):", str(df.shape))

df.head()
```

```
The shape of the ORIGINAL data is (row, column): (8760, 23)
```

Out[342]:

	StationNumber	Date	CasualUser	RegisteredUser	Newregistereduser	Temperature(°C)	Humidity(
0	1	2017-01-	80.0	254	5	-5.2	

		12 00:00:00				
1	2	2017-01-12 00:00:00	79.0	204	6	-5.5
2	3	2017-01-12 00:00:00	81.0	173	8	-6.0
3	4	2017-01-12 00:00:00	48.0	107	3	-6.2
4	5	2017-01-12 00:00:00	30.0	78	3	-6.0

5 rows × 23 columns

1.5 Data Information

Get the general information about the dataset.

In [343…

```
print ("The shape of the dataset is (row, column):"+ str(df.shape))
df.info()
```

```
The shape of the dataset is (row, column):(8760, 23)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 23 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   StationNumber            8760 non-null   int64  
 1   Date                     8760 non-null   object 
 2   CasualUser               8760 non-null   float64
 3   RegisteredUser           8760 non-null   int64  
 4   Newregistereduser        8760 non-null   int64  
 5   Temperature(°C)          8760 non-null   float64
 6   Humidity(%)              8760 non-null   int64  
 7   Windspeed(m/s)           8760 non-null   float64
 8   Visibility(10m)          8760 non-null   int64  
 9   Dewpointtemperature(°C)  8760 non-null   float64
 10  SolarRadiation(MJ/m2)    8760 non-null   float64
 11  Rainfall(mm)             8760 non-null   float64
 12  Snowfall(cm)             8760 non-null   float64
 13  Seasons                  8760 non-null   object 
 14  OperationDay             8760 non-null   object 
 15  TotalUser                8760 non-null   float64
 16  CasualPercent            8760 non-null   float64
 17  RegisteredPercent        8760 non-null   float64
 18  NewRegisteredPercent     8760 non-null   float64
 19  CompletelyDry            8760 non-null   object 
 20  Below0                   8760 non-null   object 
 21  IsDec2017                8760 non-null   object 
 22  IsJun2018                8760 non-null   object 
dtypes: float64(11), int64(5), object(7)
memory usage: 1.5+ MB
```

The dataset has a total of 23 columns (15 original + 8 added in Excel file) and 8760 rows with no missing value.

The original 15 columns of the dataset are of type:

Categorical:

- **Nominal** (variables that have two or more categories, but which do not have an intrinsic order.)

> - **Station Number:** The number of the station to which the record belongs (Station 1 - 24).
> - **Date:** The date when the data was recorded.
> - **Seasons:** The current season on the day of the record.
> - **OperationDay:** Whether the station is open on the day of the record.

Numeric:

- **Continuous**

> - **Casual User:** The number of casual (non-registered) users on the day of the record.
> - **Registered User:** The number of registered users on the day of the record.
> - **New registered user:** The number of users who registered right on the day of the record.
> - **Temperature (°C):** Average temperature at the station on the day of the record.
> - **Humidity (%):** Average humidity at the station on the day of the record.
> - **Windspeed (m/s):** Average wind speed at the station on the day of the record.
> - **Visibility (10m):** Average visibility at the station on the day of the record.
> - **Dew point temperature (°C):** Average dew point temperature at the station on the day of the record.
> - **SolarRadiation (MJ/m2):** Average solar radiation at the station on the day of the record.
> - **Rainfall (mm):** Average rainfall rate at the station on the day of the record.
> - **Snowfall (cm):** Average snowfall rate at the station on the day of the record.

Additionally, there are 8 columns that are added based on the original data to aid with the analysis:

Categorical:

- **Nominal** (variables that have two or more categories, but which do not have an intrinsic order.)

> - **CompletelyDry:** Whether day of the record has no rain and snow.
> - **Below0:** Whether the average temperature were below 0°C on the day of the record.
> - **IsDec2017:** Whether the record was logged on December 2017.
> - **IsJun2018:** Whether the record was logged on June 2018.

Numeric:

- **Continuous**

> - **TotalUser:** Total number of users on the day of the record, including non-registered and registered.
> - **CasualPercent (%):** Percentage of casual users on the day of the record.
> - **RegisteredPercent (%):** Percentage of registered users on the day of the record.
> - **NewRegisteredPercent (%):** Percentage of newly registered users on the day of the record.

2. Data Cleaning and Pre-processing

2.1. Drop duplicate

In [344…

```
print ("The shape of the dataset before dropping duplicate entries:"+ str(df.shape))

df = df.drop_duplicates()

print ("The shape of the dataset after dropping duplicate entries:"+ str(df.shape))
```

```
The shape of the dataset before dropping duplicate entries:(8760, 23)
The shape of the dataset after dropping duplicate entries:(8760, 23)
```

Discussion:

There is no duplicate entry in this dataset.

2.2 Outliers

Detect and Drop regulation:

Outliers are indentified using the interquartile range (IQR) rule: Any value fall outside of the Q1 - 1.5 x IQR - Q3 + 1.5 x IQR is considered as outlier. Q1 and Q3 are the first and third quartiles of the data, respectively. The IQR measures how the data is spread about the median. Therefore it is useful in detecting outliers.

Descriptive Statistics

In [345…

```python
# Descriptive statistics of all numerical fields
df.describe().T
```

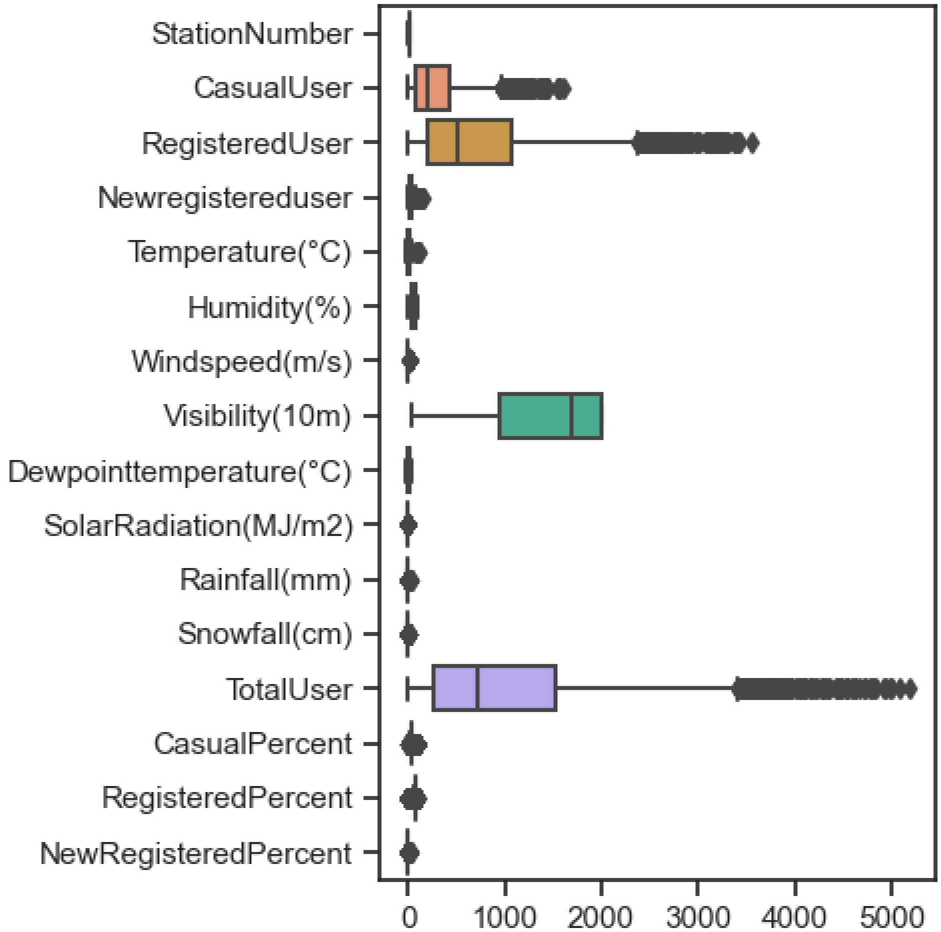
Out[345]:

	count	mean	std	min	25%	50%	75%
StationNumber	8760.0	12.500000	6.922582	1.0	6.750000	12.500000	18.25000
CasualUser	8760.0	279.777523	266.546813	0.0	69.000000	195.000000	424.00000
RegisteredUser	8760.0	704.602055	644.997468	0.0	191.000000	504.500000	1065.25000
Newregistereduser	8760.0	22.454566	22.155487	0.0	5.000000	15.000000	34.00000
Temperature(°C)	8760.0	12.993653	12.271382	-17.8	3.500000	13.800000	22.60000
Humidity(%)	8760.0	58.226256	20.362413	0.0	42.000000	57.000000	74.00000
Windspeed(m/s)	8760.0	1.724909	1.036300	0.0	0.900000	1.500000	2.30000
Visibility(10m)	8760.0	1436.825799	608.298712	27.0	940.000000	1698.000000	2000.00000
Dewpointtemperature(°C)	8760.0	4.073813	13.060369	-30.6	-4.700000	5.100000	14.80000
SolarRadiation(MJ/m2)	8760.0	0.569111	0.868746	0.0	0.000000	0.010000	0.93000
Rainfall(mm)	8760.0	0.148687	1.128193	0.0	0.000000	0.000000	0.00000
Snowfall(cm)	8760.0	0.075068	0.436746	0.0	0.000000	0.000000	0.00000
TotalUser	8760.0	1006.834144	930.154714	1.0	267.000000	715.500000	1522.25000
CasualPercent	8760.0	29.139219	13.772225	0.0	25.026670	27.066892	29.48051
RegisteredPercent	8760.0	68.838421	13.498710	0.0	68.213764	70.805825	73.01587
NewRegisteredPercent	8760.0	2.022360	0.955831	0.0	1.764608	2.066116	2.40000

In [346…

```python
# Box plot for all numerical fields
sns.boxplot(data=df,orient="h")
```

Out[346]: <AxesSubplot:>



There are some unrealistic values in `Temperature` and `Humidity`, which is outside the highest and lowest values ever recorded in the world.

Temperature

IQR

In [347…

```
# Find Q1, Q3, and IQR of Temperature
q1_Temperature = df['Temperature(°C)'].quantile(.25)
q3_Temperature = df['Temperature(°C)'].quantile(.75)
iqr_Temperature = q3_Temperature - q1_Temperature

print("q1_Temperature:", q1_Temperature, "\n")
print("q3_Temperature:", q3_Temperature, "\n")
print("iqr_Temperature:", iqr_Temperature)
```

```
q1_Temperature: 3.5

q3_Temperature: 22.6

iqr_Temperature: 19.1
```

Variability

In [348…

```
# Temperature Mean
Temperature_mean = df['Temperature(°C)'].mean()
print("Temperature_mean:", Temperature_mean)
# Temperature Median
Temperature_median = df['Temperature(°C)'].median()
print("Temperature_median:", Temperature_median)
```

```
# Temperature Mode
Temperature_mode = df['Temperature(°C)'].mode().values[0]
print("Temperature_mode:", Temperature_mode)
```

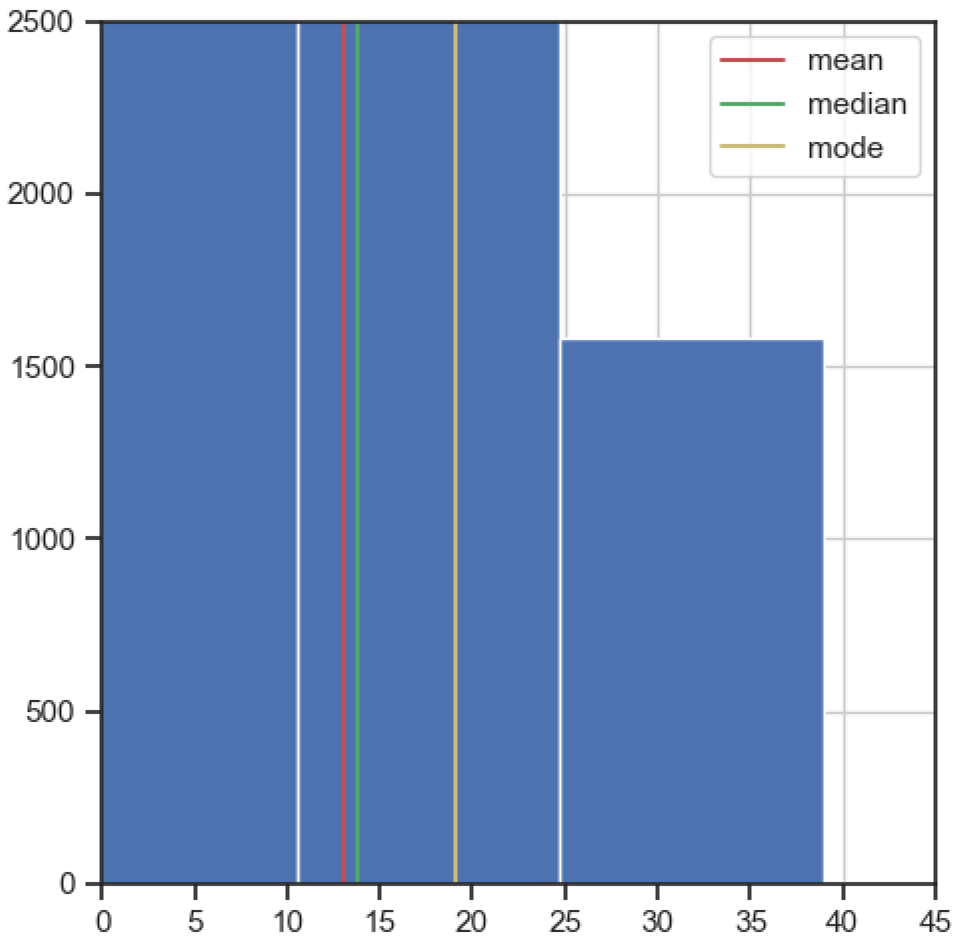
```
Temperature_mean: 12.99365296803654
Temperature_median: 13.8
Temperature_mode: 19.1
```

In [349…

```
# Plot the histogram of Temperature with mean, median, and mode
df['Temperature(°C)'].hist()

plt.axvline(Temperature_mean, color='r', label='mean')
plt.axvline(Temperature_median, color='g', label='median')
plt.axvline(Temperature_mode, color='y', label='mode')

plt.legend()

plt.xlim(0, 45)
plt.ylim(0, 2500)
```

Out[349]:

```
(0.0, 2500.0)
```



Discussion:

> - The mean lower than the median indicates the data is skewed to the right.
> - Moreover, it is impossible for the `Temperature(°C)` to have the value is higher than the world highest record for Temperature [1]. Therefore, there are outliers in this field.

Humidity

IQR

In [350…

```
# Find Q1, Q3, and IQR of Humidity
q1_Humidity = df['Humidity(%)'].quantile(.25)
q3_Humidity = df['Humidity(%)'].quantile(.75)
iqr_Humidity = q3_Humidity - q1_Humidity

print("q1_Humidity:", q1_Humidity, "\n")
print("q3_Humidity:", q3_Humidity, "\n")
print("iqr_Humidity:", iqr_Humidity)
```

```
q1_Humidity: 42.0

q3_Humidity: 74.0

iqr_Humidity: 32.0
```

Variability

In [351…

```
# Humidity Mean
Humidity_mean = df['Humidity(%)'].mean()
print("Humidity_mean:", Humidity_mean)
# Humidity Median
Humidity_median = df['Humidity(%)'].median()
print("Humidity_median:", Humidity_median)
# Humidity Mode
Humidity_mode = df['Humidity(%)'].mode().values[0]
print("Humidity_mode:", Humidity_mode)
```

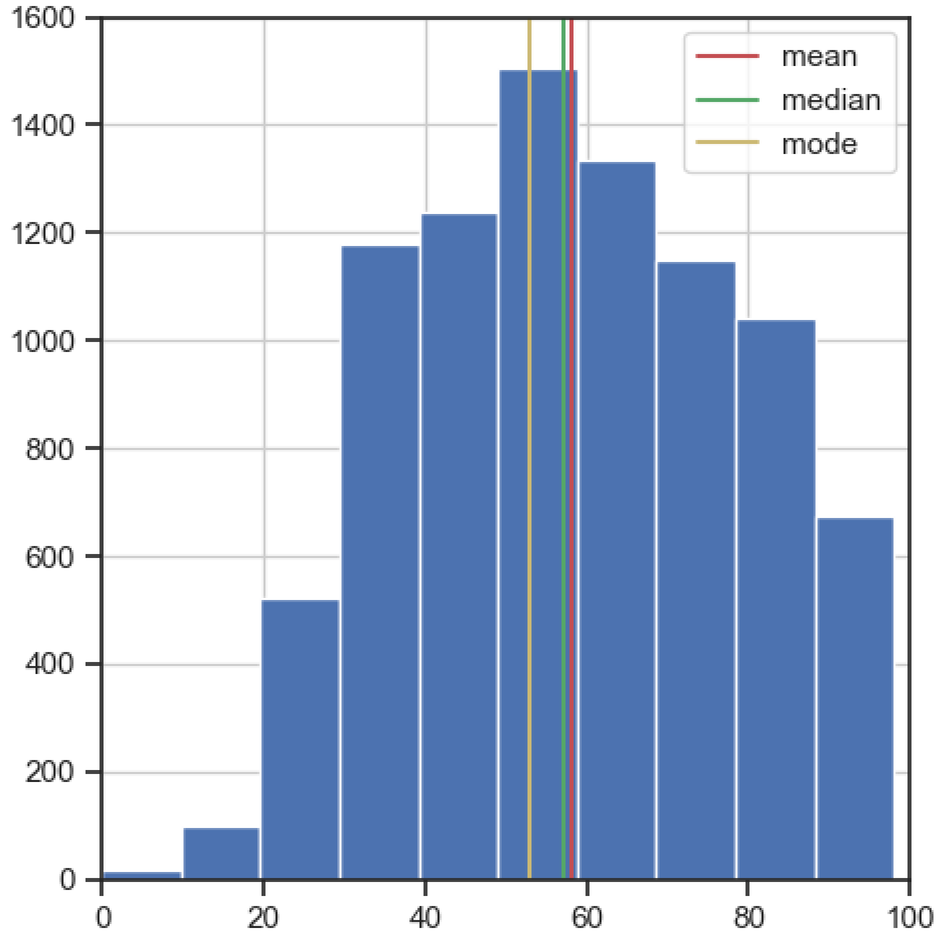
```
Humidity_mean: 58.226255707762554
Humidity_median: 57.0
Humidity_mode: 53
```

In [352…

```
# Plot the histogram of Humidity with mean, median, and mode
df['Humidity(%)'].hist()

plt.axvline(Humidity_mean, color='r', label='mean')
plt.axvline(Humidity_median, color='g', label='median')
plt.axvline(Humidity_mode, color='y', label='mode')

plt.legend()

plt.xlim(0, 100)
plt.ylim(0, 1600)
```

Out[352]:

```
(0.0, 1600.0)
```



Discussion:

> - The mean larger than the median indicates that the data is skewed to the left.
> - Moreover, it is impossible for the `Humidity(%)` to have the value is higher than the world highest record for Humidity [2]. Therefore, there are outliers in this field.

Detecting and Dealing with outliers

In [353…

```
def detect_outliers_IQR(df):
    # Find Q1:
    Q1 = np.percentile(df, 25)
    # Find Q3:
    Q3 = np.percentile(df, 75)
    # Find the IQR:
    IQR = Q3 - Q1
    # Upper bound
    upper = np.where(df >= (Q3 + 1.5*IQR))
    # Lower bound
    lower = np.where(df <= (Q1 - 1.5*IQR))
    # Outliers
    outliers = df[((df < (Q1 - 1.5*IQR)) | (df > (Q3 + 1.5*IQR)))]
    return outliers, upper, lower
```

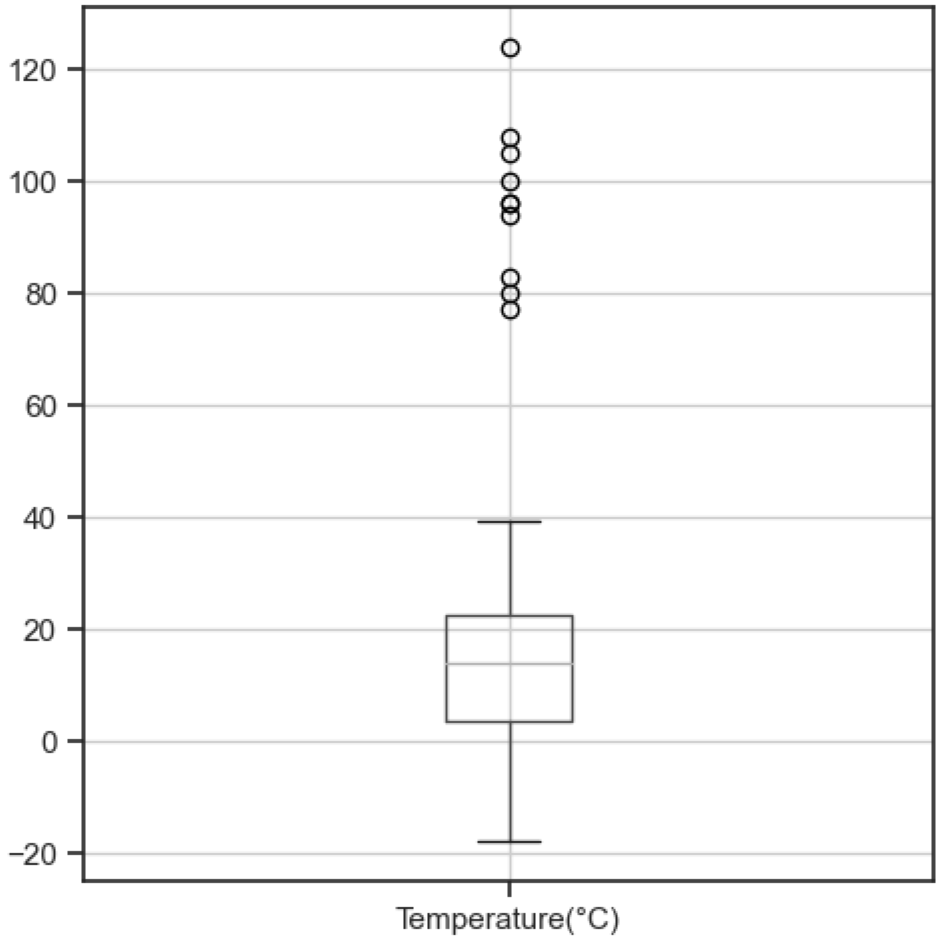
`Temperature(°C)` column

In [354…

```
df.boxplot(column="Temperature(°C)")
```

Out[354]:

```
<AxesSubplot:>
```



In [355…

```
outliers, upper, lower = detect_outliers_IQR(df['Temperature(°C)'])

print("number of outliers: "+ str(len(outliers)))

print("max outlier value: "+ str(outliers.max()))

print("min of outliers: "+ str(outliers.min()))

print("Percentage of outliers: "+ str(len(outliers)/len(df) * 100))
```

```
number of outliers: 10
max outlier value: 124.0
min of outliers: 77.0
Percentage of outliers: 0.1141552511415525
```

Discussion:

The percentage of outliers in `Temperature(°C)` is extremely small so all entries with outliers are dropped.

In [356…

```
df.drop(upper[0], inplace=True)
```

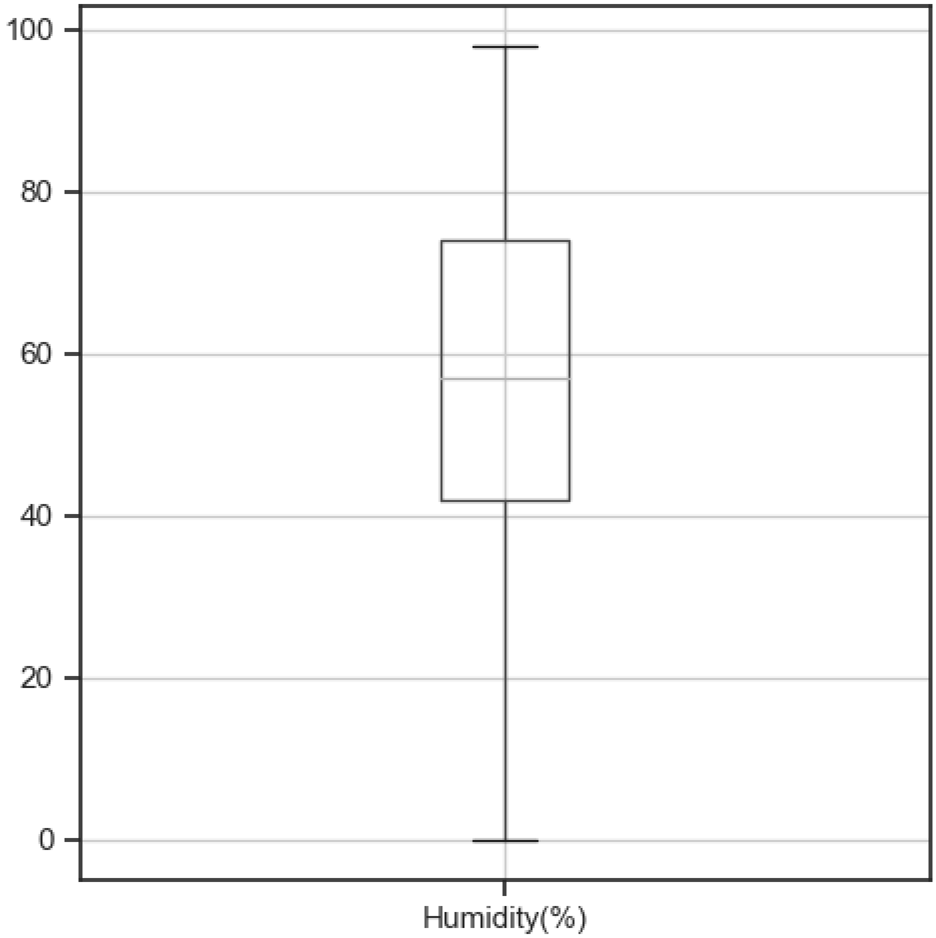
> **`Humidity(%)` column**

In [357…

```
df.boxplot(column= "Humidity(%)")
```

Out[357]:

```
<AxesSubplot:>
```



In [358…

```
outliers, upper, lower = detect_outliers_IQR(df['Humidity(%)'])

print("number of outliers: "+ str(len(outliers)))

print("max outlier value: "+ str(outliers.max()))

print("min of outliers: "+ str(outliers.min()))

print("Percentage of outliers: "+ str(len(outliers)/len(df) * 100))
```

```
number of outliers: 0
max outlier value: nan
min of outliers: nan
Percentage of outliers: 0.0
```

Discussion:

`Humidity` has no outliers.

CasualUser

In [359…

```
outliers, upper, lower = detect_outliers_IQR(df['CasualUser'])

print("number of outliers: "+ str(len(outliers)))

print("max outlier value: "+ str(outliers.max()))

print("min of outliers: "+ str(outliers.min()))

print("Percentage of outliers: "+ str(len(outliers)/len(df) * 100))
```

```
number of outliers: 200
max outlier value: 1599.0
min of outliers: 958.0
Percentage of outliers: 2.2857142857142856
```

In [360…

```
extreme_temp = df[(df['CasualUser'] < outliers.min()) | (df['CasualUser'] > outliers.max
extreme_temp.head()
```

Out[360]:

	StationNumber	Date	CasualUser	RegisteredUser	Newregistereduser	Temperature(°C)	Humidity(
0	1	2017-01-12 00:00:00	80.0	254	5	-5.2	
1	2	2017-01-12 00:00:00	79.0	204	6	-5.5	
2	3	2017-01-12 00:00:00	81.0	173	8	-6.0	
3	4	2017-01-12 00:00:00	48.0	107	3	-6.2	
4	5	2017-01-12 00:00:00	30.0	78	3	-6.0	

5 rows × 23 columns

Discussion:

In case the weather is in the extreme mode, the number of user may drop. There might be some correlation between them.

RegisteredUser

In [361…

```
outliers, upper, lower = detect_outliers_IQR(df['RegisteredUser'])

print("number of outliers: "+ str(len(outliers)))

print("max outlier value: "+ str(outliers.max()))

print("min of outliers: "+ str(outliers.min()))

print("Percentage of outliers: "+ str(len(outliers)/len(df) * 100))
```

```
number of outliers: 155
max outlier value: 3556
min of outliers: 2379
Percentage of outliers: 1.7714285714285714
```

In [362…

```
extreme_temp = df[(df['RegisteredUser'] < outliers.min()) | (df['RegisteredUser'] > outl
extreme_temp.head()
```

Out[362]:

	StationNumber	Date	CasualUser	RegisteredUser	Newregistereduser	Temperature(°C)	Humidity(
0	1	2017-01-12 00:00:00	80.0	254	5	-5.2	

	StationNumber	Date	CasualUser	RegisteredUser	Newregistereduser	Temperature(°C)
1	2	2017-01-12 00:00:00	79.0	204	6	-5.5
2	3	2017-01-12 00:00:00	81.0	173	8	-6.0
3	4	2017-01-12 00:00:00	48.0	107	3	-6.2
4	5	2017-01-12 00:00:00	30.0	78	3	-6.0

5 rows × 23 columns

Discussion:

In case the weather is in the extreme mode, the number of user may drop. There might be some correlation between them.

Newregistereduser

In [363…

```
outliers, upper, lower = detect_outliers_IQR(df['Newregistereduser'])

print("number of outliers: "+ str(len(outliers)))

print("max outlier value: "+ str(outliers.max()))

print("min of outliers: "+ str(outliers.min()))

print("Percentage of outliers: "+ str(len(outliers)/len(df) * 100))
```

```
number of outliers: 256
max outlier value: 159
min of outliers: 78
Percentage of outliers: 2.9257142857142857
```

In [364…

```
extreme_temp = df[(df['Newregistereduser'] < outliers.min()) | (df['Newregistereduser']
extreme_temp.head()
```

Out[364]:

	StationNumber	Date	CasualUser	RegisteredUser	Newregistereduser	Temperature(°C)	Humidity(
0	1	2017-01-12 00:00:00	80.0	254	5	-5.2	
1	2	2017-01-12 00:00:00	79.0	204	6	-5.5	
2	3	2017-01-12 00:00:00	81.0	173	8	-6.0	
3	4	2017-01-12 00:00:00	48.0	107	3	-6.2	
4	5	2017-01-12 00:00:00	30.0	78	3	-6.0	

5 rows × 23 columns

Discussion:

In case the weather is in the extreme mode, the number of user may drop. There might be some correlation between them.

Windspeed

In [365…

```
df_Windspeed_q_low = df["Windspeed(m/s)"].quantile(0.02)
df_Windspeed_q_hi  = df["Windspeed(m/s)"].quantile(0.99)

df_filtered = df[(df["Windspeed(m/s)"] > df_Windspeed_q_hi) | (df["Windspeed(m/s)"] < df
print(len(df_filtered)/ len(df) * 100)
df_filtered
```

```
2.262857142857143
```

Out[365]:

	StationNumber	Date	CasualUser	RegisteredUser	Newregistereduser	Temperature(°C)	Hum
84	13	2017-04-12 00:00:00	89.318182	393	11	-0.3	
85	14	2017-04-12 00:00:00	97.750000	391	11	0.0	
87	16	2017-04-12 00:00:00	87.435897	341	9	-0.1	
89	18	2017-04-12 00:00:00	128.750000	515	11	-1.3	
107	12	2017-05-12 00:00:00	87.894737	334	6	-3.9	
...	...	...	...	...	...	...	
8330	3	13/11/2018	128.000000	330	12	5.6	
8331	4	13/11/2018	64.000000	205	5	5.3	
8332	5	13/11/2018	51.000000	133	4	4.9	
8333	6	13/11/2018	67.000000	162	5	4.7	
8410	11	16/11/2018	316.000000	699	26	9.4	

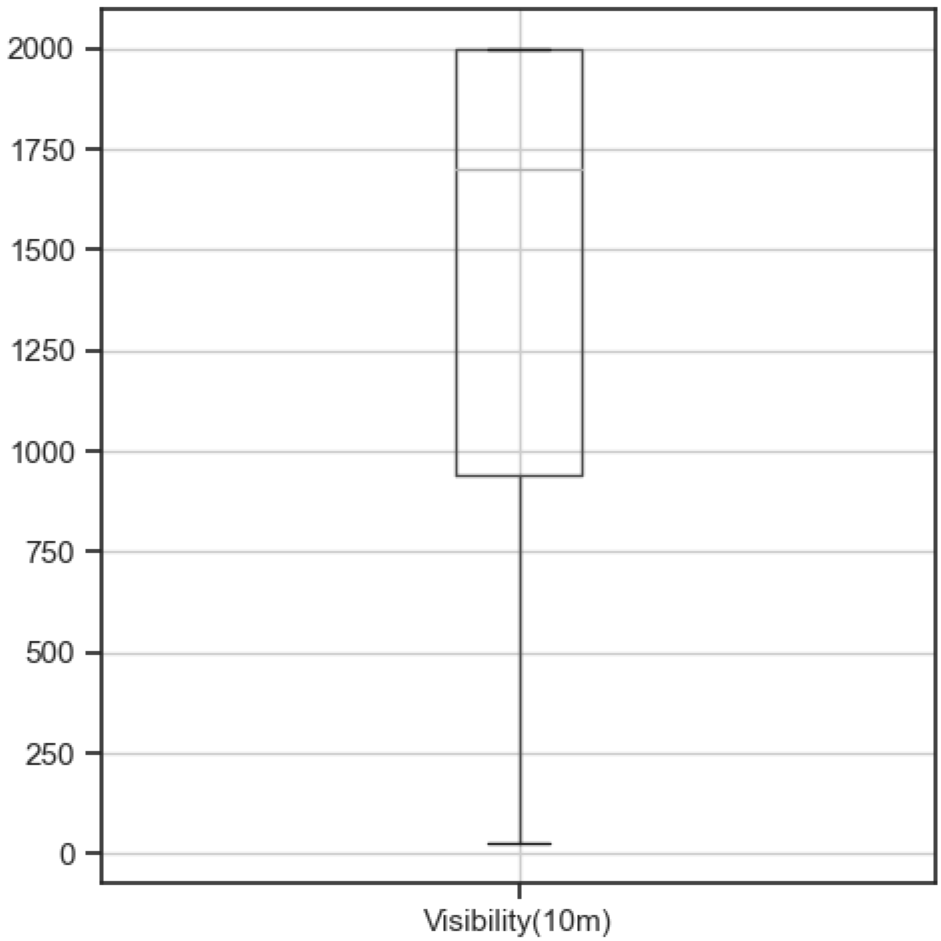
198 rows × 23 columns

Discussion:

`Windspeed` seem to have some influences on the number of user since when `Windspeed` = 0 then the number of users are higher than the `Windspeed` is high.

Visibility

In [366…
```
df.boxplot(column= "Visibility(10m)")
```
Out[366]:
```
<AxesSubplot:>
```



In [367…
```
outliers, upper, lower = detect_outliers_IQR(df['Visibility(10m)'])

print("number of outliers: "+ str(len(outliers)))

print("max outlier value: "+ str(outliers.max()))

print("min of outliers: "+ str(outliers.min()))

print("Percentage of outliers: "+ str(len(outliers)/len(df) * 100))
```
```
number of outliers: 0
max outlier value: nan
min of outliers: nan
Percentage of outliers: 0.0
```

Discussion:

There is so outlier in `Visibility`

Dewpointtemperature

In [368…
```
df_Dewpointtemperature_q_low = df["Dewpointtemperature(°C)"].quantile(0.02)
df_Dewpointtemperature_q_hi  = df["Dewpointtemperature(°C)"].quantile(0.99)
```

```
df_filtered = df[(df["Dewpointtemperature(°C)"] > df_Dewpointtemperature_q_hi) |
                 (df["Dewpointtemperature(°C)"] < df_Dewpointtemperature_q_low)]
print(len(df_filtered)/ len(df) * 100)
df_filtered
```

2.9485714285714284

Out[368]:

	StationNumber	Date	CasualUser	RegisteredUser	Newregistereduser	Temperature(°C)	Hum
613	14	26/12/2017	114.0	262	10	-2.0	
615	16	26/12/2017	88.0	246	7	-1.9	
616	17	26/12/2017	133.0	282	9	-1.9	
617	18	26/12/2017	138.0	350	10	-3.6	
633	10	27/12/2017	133.0	279	10	-9.8	
...	...	...	...	...	...	...	
6121	2	13/08/2018	75.0	231	5	28.2	
6123	4	13/08/2018	76.0	196	5	27.7	
6124	5	13/08/2018	49.0	151	4	27.1	
6125	6	13/08/2018	72.0	230	6	26.8	
6186	19	15/08/2018	328.0	925	21	32.8	

258 rows × 23 columns

Discussion:

The `Dewpointtemperature` seem to have some influences on the number of user since when the number of `Dewpointtemperature` < 0 then the number of users are lower than the `Dewpointtemperature` is high.

SolarRadiation

In [369…

```
df_SolarRadiation_q_low = df["SolarRadiation(MJ/m2)"].quantile(0.02)
df_SolarRadiation_q_hi  = df["SolarRadiation(MJ/m2)"].quantile(0.99)

df_filtered = df[(df["SolarRadiation(MJ/m2)"] > df_SolarRadiation_q_hi) |
                 (df["SolarRadiation(MJ/m2)"] < df_SolarRadiation_q_low)]
print(len(df_filtered)/ len(df) * 100)
df_filtered
```

0.9714285714285713

Out[369]:

	StationNumber	Date	CasualUser	RegisteredUser	Newregistereduser	Temperature(°C)	Hum
2989	14	2018-04-04 00:00:00	339.0	951	22	16.7	
3157	14	2018-11-04 00:00:00	1.0	0	0	15.6	
3181	14	2018-12-04 00:00:00	440.0	1029	44	18.1	
3277	14	16/04/2018	370.0	973	26	16.6	

3325	14	18/04/2018	459.0	975	32	17.7
...	...	...	...	...	...	...
6204	13	16/08/2018	242.0	668	17	34.5
6205	14	16/08/2018	293.0	652	26	35.1
6229	14	17/08/2018	339.0	820	30	31.3
6230	15	17/08/2018	358.0	791	32	32.7
6253	14	18/08/2018	364.0	1003	30	31.1

85 rows × 23 columns

Discussion:

There is no clear relationship between the `SolarRadiation` and number of users

Rainfall

In [370…

```
df_SolarRadiation_q_low = df["Rainfall(mm)"].quantile(0.02)
df_SolarRadiation_q_hi  = df["Rainfall(mm)"].quantile(0.99)

df_filtered = df[(df["Rainfall(mm)"] > df_SolarRadiation_q_hi) |
                 (df["Rainfall(mm)"] < df_SolarRadiation_q_low)]
print(len(df_filtered)/ len(df) * 100)
df_filtered
```

```
0.9028571428571429
```

Out[370]:

	StationNumber	Date	CasualUser	RegisteredUser	Newregistereduser	Temperature(°C)	Hun
561	10	24/12/2017	4.0	3	0	4.6	
564	13	24/12/2017	6.0	4	0	4.1	
2151	16	28/02/2018	4.0	7	0	4.8	
2154	19	28/02/2018	2.0	11	0	3.6	
2157	22	28/02/2018	3.0	10	0	2.4	
...	...	...	...	...	...	...	
8223	16	2018-08-11 00:00:00	25.0	56	1	11.4	
8226	19	2018-08-11 00:00:00	7.0	40	0	12.9	
8229	22	2018-08-11 00:00:00	6.0	21	0	14.0	
8232	1	2018-09-11 00:00:00	7.0	0	0	12.0	
8601	10	24/11/2018	4.0	24	0	0.3	

79 rows × 23 columns

Discussion:

There is no clear relationship between the `Rainfall` and number of users

Snowfall

In [371…

```
df_SolarRadiation_q_low = df["Snowfall(cm)"].quantile(0.02)
df_SolarRadiation_q_hi  = df["Snowfall(cm)"].quantile(0.99)

df_filtered = df[(df["Snowfall(cm)"] > df_SolarRadiation_q_hi) |
                 (df["Snowfall(cm)"] < df_SolarRadiation_q_low)]
print(len(df_filtered)/ len(df) * 100)
df_filtered
```

```
0.9942857142857142
```

Out[371]:

	StationNumber	Date	CasualUser	RegisteredUser	Newregistereduser	Temperature(°C)	Hum
222	7	2017-10-12 00:00:00	1.956522	9	5	-0.5	
223	8	2017-10-12 00:00:00	4.878049	20	1	-0.4	
224	9	2017-10-12 00:00:00	8.974359	35	2	-0.2	
225	10	2017-10-12 00:00:00	9.117647	31	4	0.2	
226	11	2017-10-12 00:00:00	5.937500	19	3	0.5	
...	...	...	...	...	...	...	
8621	6	25/11/2018	34.000000	88	2	2.1	
8622	7	25/11/2018	24.000000	75	1	1.7	
8623	8	25/11/2018	61.000000	142	5	1.3	
8624	9	25/11/2018	117.000000	250	10	1.4	
8625	10	25/11/2018	119.000000	355	9	2.3	

87 rows × 23 columns

Discussion:

There is no clear relationship between the `Snowfall` and number of users

3. Data Analysis

3.1. RQ1. Which weather factor(s) most likely affect the number of e-scooter rentees?

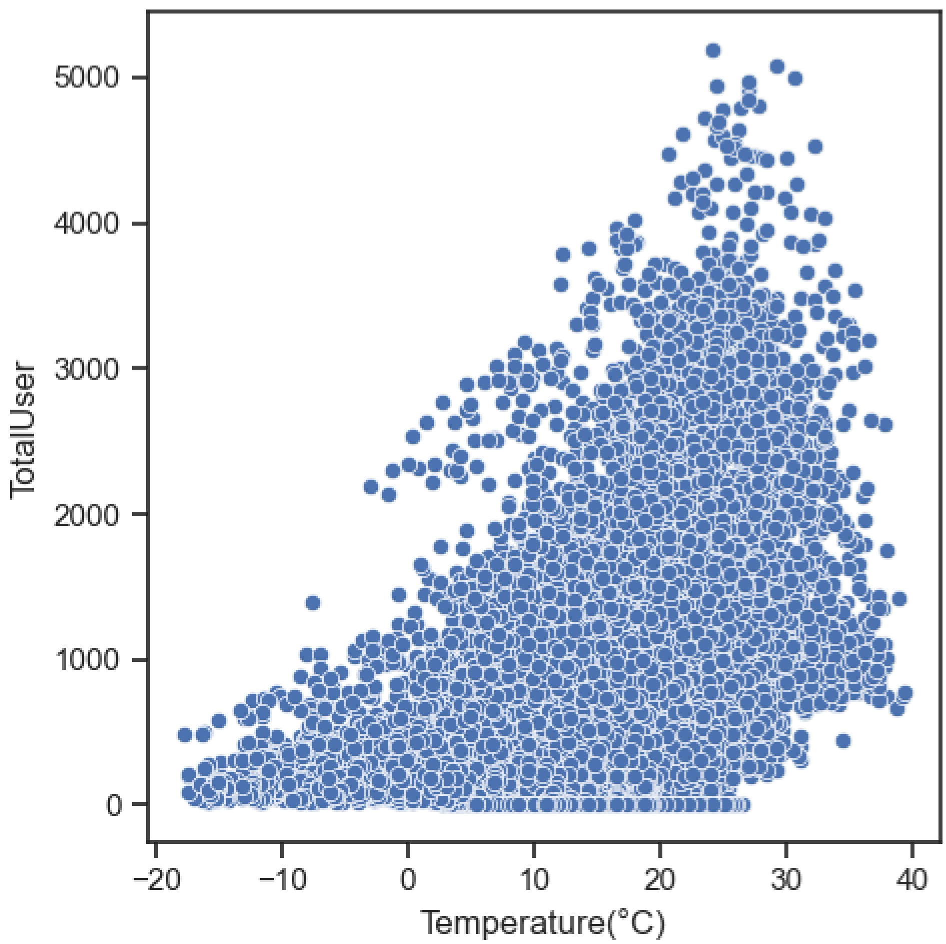
Correlation between `Temperature` and `TotalUser`:

Hypotheses:

> - **Null hypothesis (H_0)** : `Temperature` and `TotalUser` are not correlated.
> - **Alternative hypothesis (H_1)**: `Temperature` and `TotalUser` are correlated.

Significance level: 0.05

In [372…
```
sns.scatterplot(x="Temperature(°C)", y="TotalUser", data=df);
```



In [373…
```
stats.pearsonr(df['TotalUser'], df['Temperature(°C)'])
```

Out[373]:
```
PearsonRResult(statistic=0.5392749424031467, pvalue=0.0)
```

Discussion:

`Temperature` and `TotalUser` have a strong positive correlation because:

> - p-value is lower than the ***significance level (0.05)***, therfore H_0 ***is REJECTED***.
> - The Pearson correlation coefficient (r) is high: 0.539.

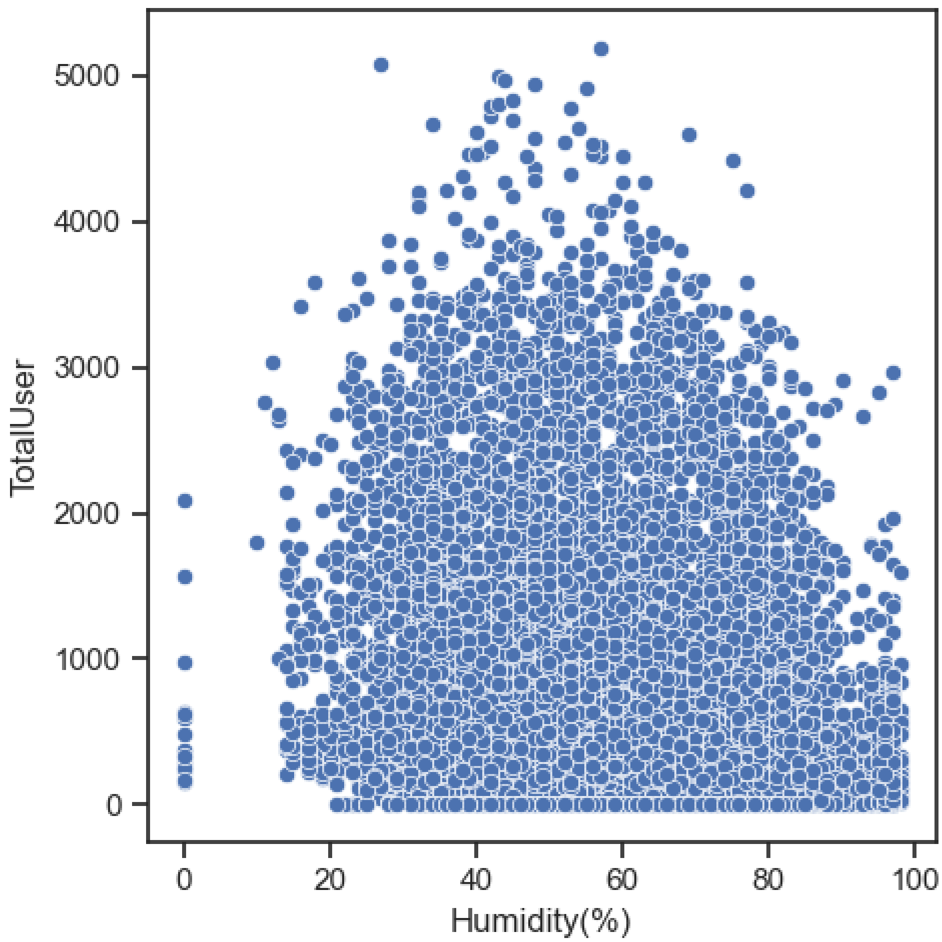
Correlation between `Humidity` and `TotalUser`:

Hypotheses:

> - **Null hypothesis (H_0)** : `Humidity` and `TotalUser` are not correlated.
> - **Alternative hypothesis (H_1)**: `Humidity` and `TotalUser` are correlated.

Significance level: 0.05

In [374…
```
sns.scatterplot(x="Humidity(%)", y="TotalUser", data=df);
```



In [375…
```
stats.pearsonr(df['TotalUser'], df['Humidity(%)'])
```

Out[375]:
```
PearsonRResult(statistic=-0.19779007923186678, pvalue=6.708536037979973e-78)
```

Discussion:

`Humidity` and `TotalUser` have a weak negative correlation because:

> - p-value is lower than the ***significance level (0.05)***, therefore H_0 ***is REJECTED***.
> - The Pearson correlation coefficient (r) is low: -0.198.

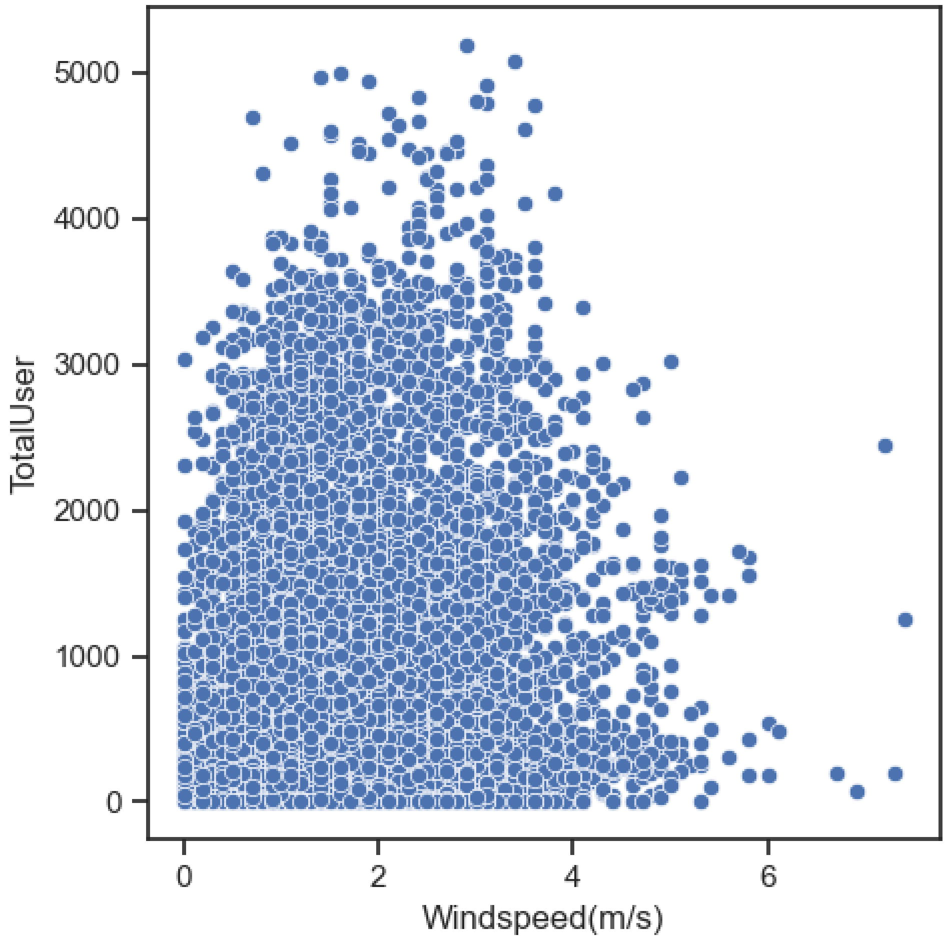
Correlation between `Windspeed` and `TotalUser`:

Hypotheses:

> - **Null hypothesis (H_0)** : `Windspeed` and `TotalUser` are not correlated.
> - **Alternative hypothesis (H_1)**: `Windspeed` and `TotalUser` are correlated.

Significance level: 0.05

In [376…
```
sns.scatterplot(x="Windspeed(m/s)", y="TotalUser", data=df);
```



In [377…
```
stats.pearsonr(df['TotalUser'], df['Windspeed(m/s)'])
```

Out[377]:
```
PearsonRResult(statistic=0.12017919733501264, pvalue=1.628191773492102e-29)
```

Discussion:

`Windspeed` and `CasualUser` have a weak positive correlation because:

> - p-value is lower than the ***significance level (0.05)***, therefore H_0 ***is REJECTED***.
> - The Pearson correlation coefficient (r) is low: 0.120.

Correlation between `Visibility` and `TotalUser`:

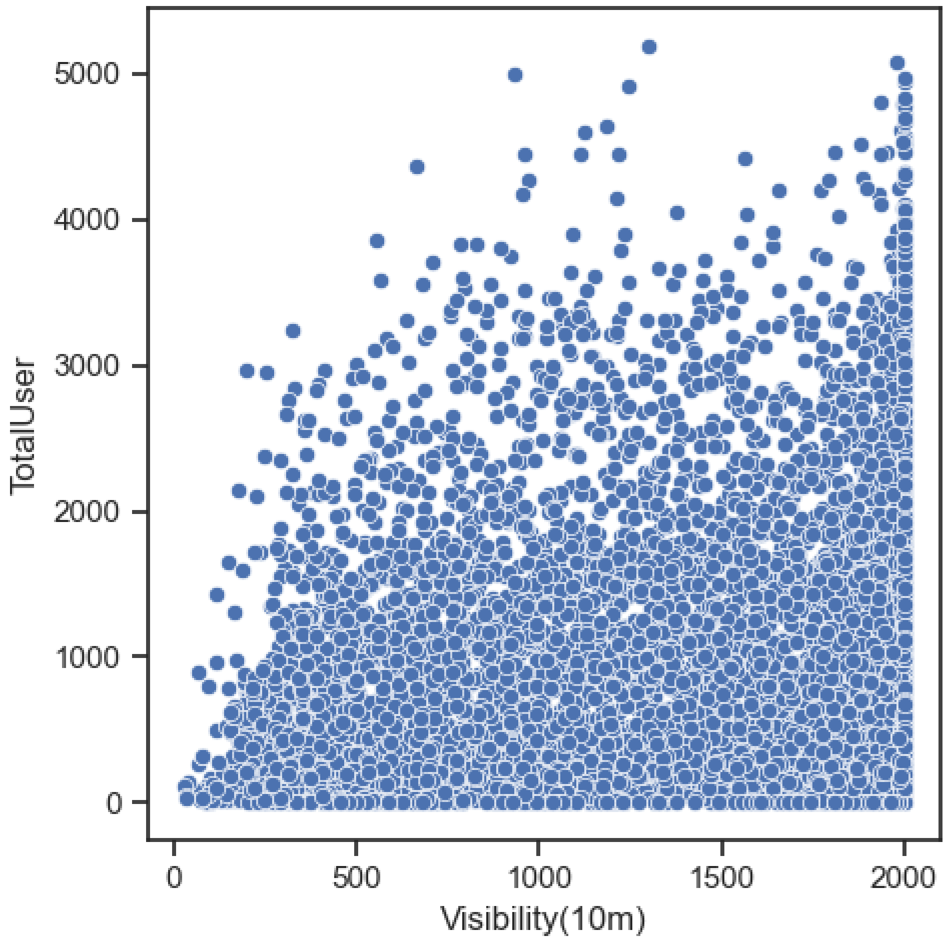
Hypotheses:

> - **Null hypothesis (H_0)** : `Visibility` and `TotalUser` are not correlated.

- **Alternative hypothesis (H_1):** `Visibility` and `TotalUser` are correlated.

Significance level: 0.05

In [378…
```
sns.scatterplot(x="Visibility(10m)", y="TotalUser", data=df);
```



In [379…
```
stats.pearsonr(df['TotalUser'], df['Visibility(10m)'])
```

Out[379]:
```
PearsonRResult(statistic=0.1966548193062197, pvalue=5.1781840308660615e-77)
```

Discussion:

`Visibility` and `CasualUser` have a weak positive correlation because:

> - p-value is lower than the ***significance level (0.05)***, therefore H_0 ***is REJECTED***.
> - The Pearson correlation coefficient (r) is low: 0.197.

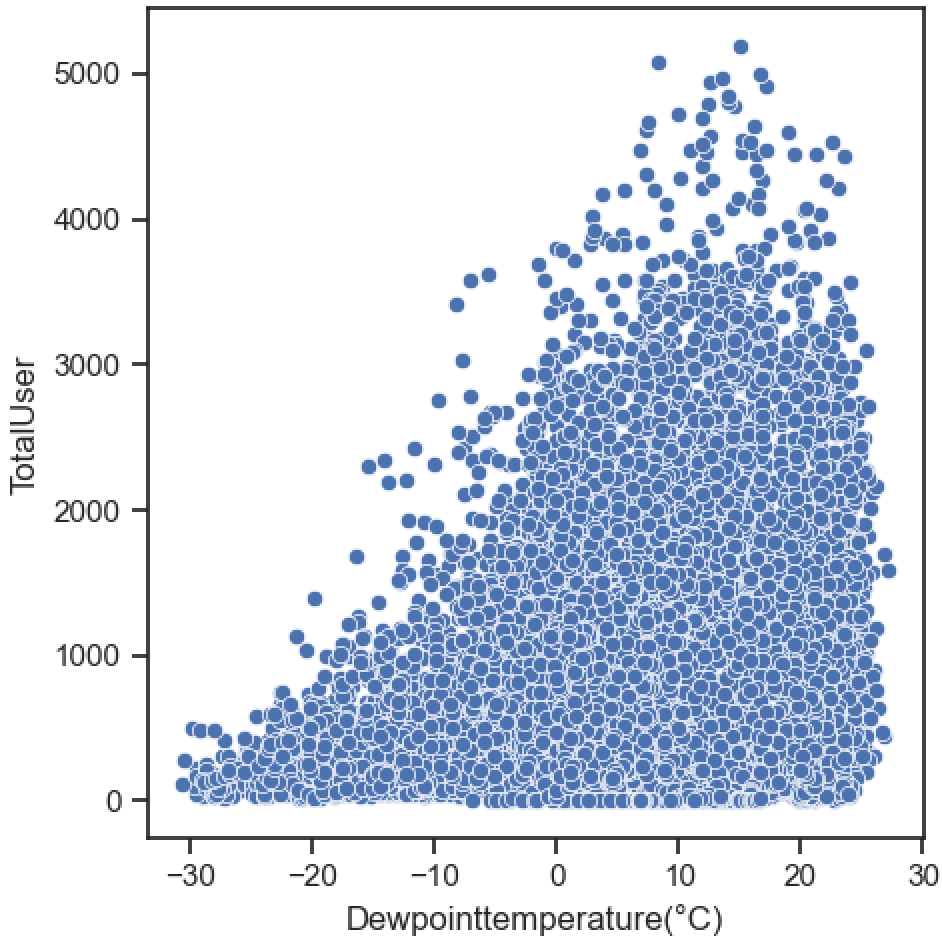
Correlation between `Dewpointtemperature` and `TotalUser`:

Hypotheses:

> - **Null hypothesis (H_0)** : `DewPointTemperature` and `TotalUser` are not correlated.
> - **Alternative hypothesis (H_1):** `DewPointTemperature` and `TotalUser` are correlated.

Significance level: 0.05

In [380…
```
sns.scatterplot(x="Dewpointtemperature(°C)", y="TotalUser", data=df);
```



In [381…
```
stats.pearsonr(df['TotalUser'], df['Dewpointtemperature(°C)'])
```

Out[381]:
```
PearsonRResult(statistic=0.38090845431375475, pvalue=3.6889172092605124e-300)
```

Discussion:

`Dewpointtemperature` and `TotalUser` have a moderate positive correlation because:

> - p-value is lower than the ***significance level (0.05)***, therefore H_0 ***is REJECTED***.
> - The Pearson correlation coefficient (r) is medium: 0.381.

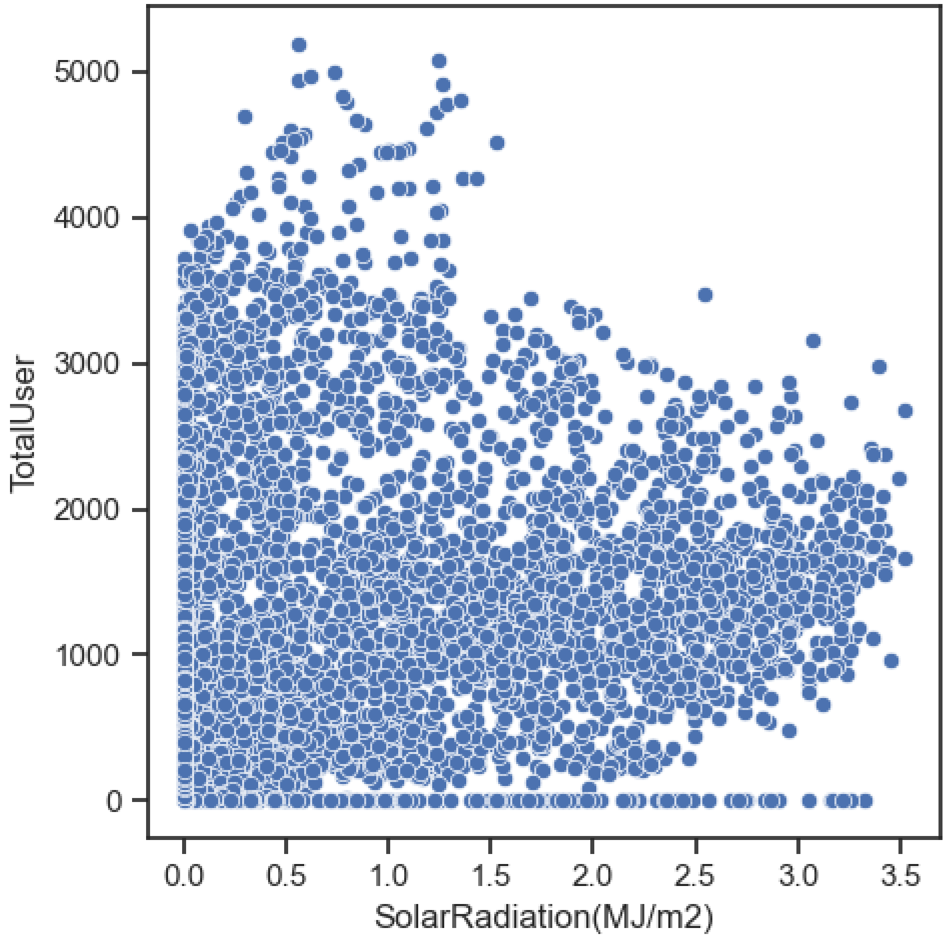
Correlation between `SolarRadiation` and `TotalUser`:

Hypotheses:

> - **Null hypothesis (H_0)** : `SolarRadiation` and `TotalUser` are not correlated.
> - **Alternative hypothesis (H_1)**: `SolarRadiation` and `TotalUser` are correlated.

Significance level: 0.05

In [382…
```
sns.scatterplot(x="SolarRadiation(MJ/m2)", y="TotalUser", data=df);
```



In [383…
```
stats.pearsonr(df['TotalUser'], df['SolarRadiation(MJ/m2)'])
```

Out[383]:
```
PearsonRResult(statistic=0.26143375780127354, pvalue=1.0703045741383986e-136)
```

Discussion:

`SolarRadiation` and `TotalUser` have a weak positive correlation because:

> - p-value is lower than the ***significance level (0.05)***, therefore H_0 ***is REJECTED***.
> - The Pearson correlation coefficient (r) is low: 0.261.

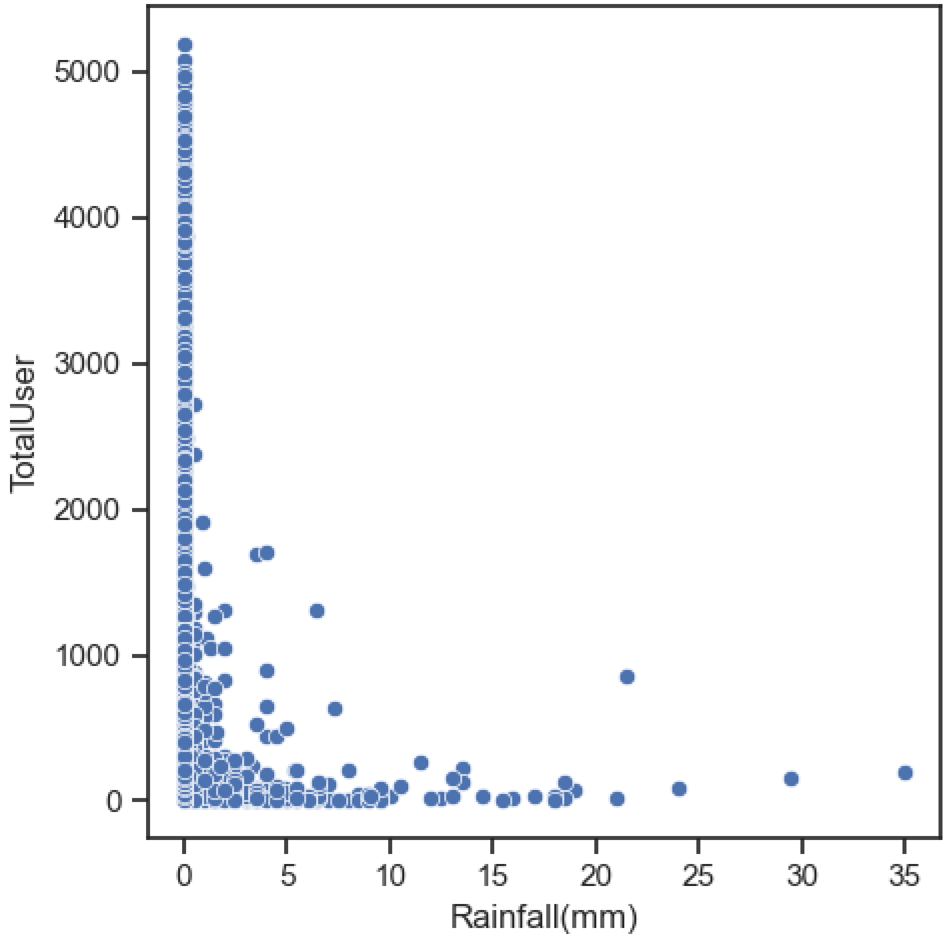
Correlation between `Rainfall` and `TotalUser`:

Hypotheses:

> - **Null hypothesis (H_0)** : `Rainfall` and `TotalUser` are not correlated.
> - **Alternative hypothesis (H_1)**: `Rainfall` and `TotalUser` are correlated.

Significance level: 0.05

In [384…
```
sns.scatterplot(x="Rainfall(mm)", y="TotalUser", data=df);
```



In [385…
```
stats.pearsonr(df['TotalUser'], df['Rainfall(mm)'])
```

Out[385]:
```
PearsonRResult(statistic=-0.12248067401187565, pvalue=1.3392301488629682e-30)
```

Discussion:

`Rainfall` and `TotalUser` have a weak negative correlation because:

> - p-value is lower than the ***significance level (0.05)***, therefore H_0 ***is REJECTED***.
> - The Pearson correlation coefficient (r) is low: -0.122.

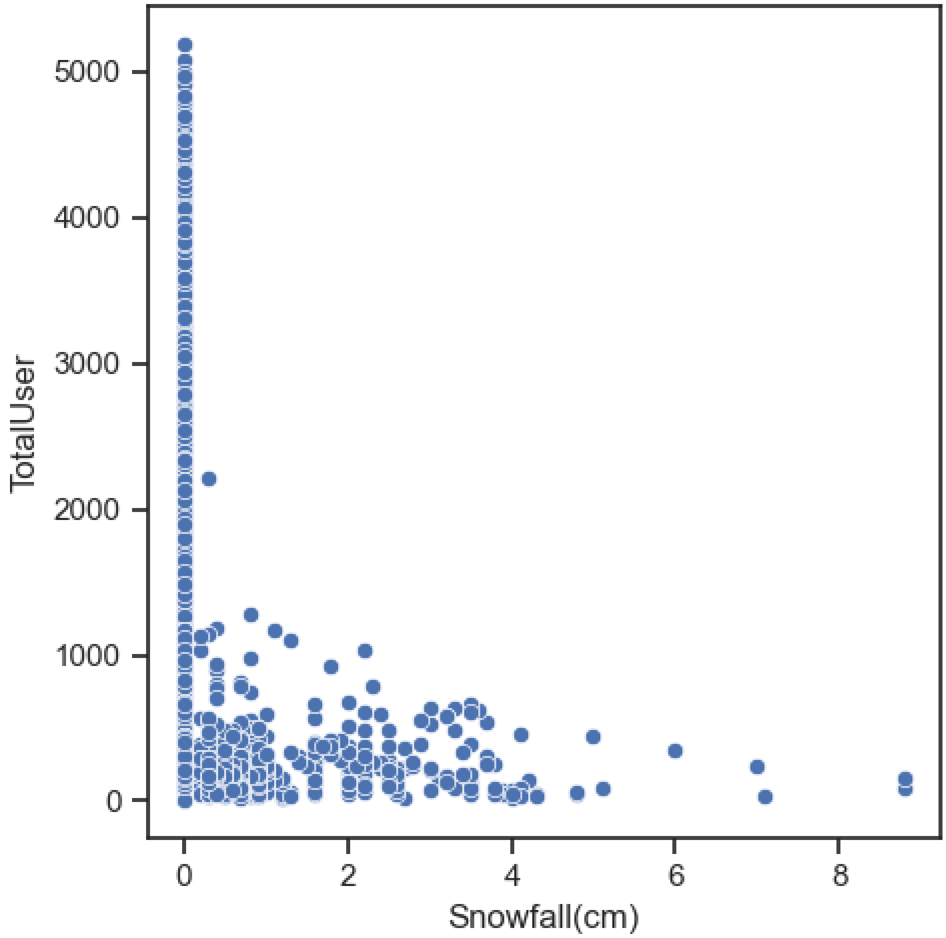
Correlation between `Snowfall` and `TotalUser`:

Hypotheses:

> - **Null hypothesis (H_0)** : `Snowfall` and `TotalUser` are not correlated.
> - **Alternative hypothesis (H_1)**: `Snowfall` and `TotalUser` are correlated.

Significance level: 0.05

In [386…
```
sns.scatterplot(x="Snowfall(cm)", y="TotalUser", data=df);
```



In [387…
```
stats.pearsonr(df['TotalUser'], df['Snowfall(cm)'])
```

Out[387]:
```
PearsonRResult(statistic=-0.140354018552346, pvalue=9.706049292735327e-40)
```

Discussion:

`Snowfall` and `TotalUser` have a weak negative correlation because:

> - p-value is lower than the ***significance level (0.05)***, therefore H_0 ***is REJECTED***.
> - The Pearson correlation coefficient (r) is low: -0.140.

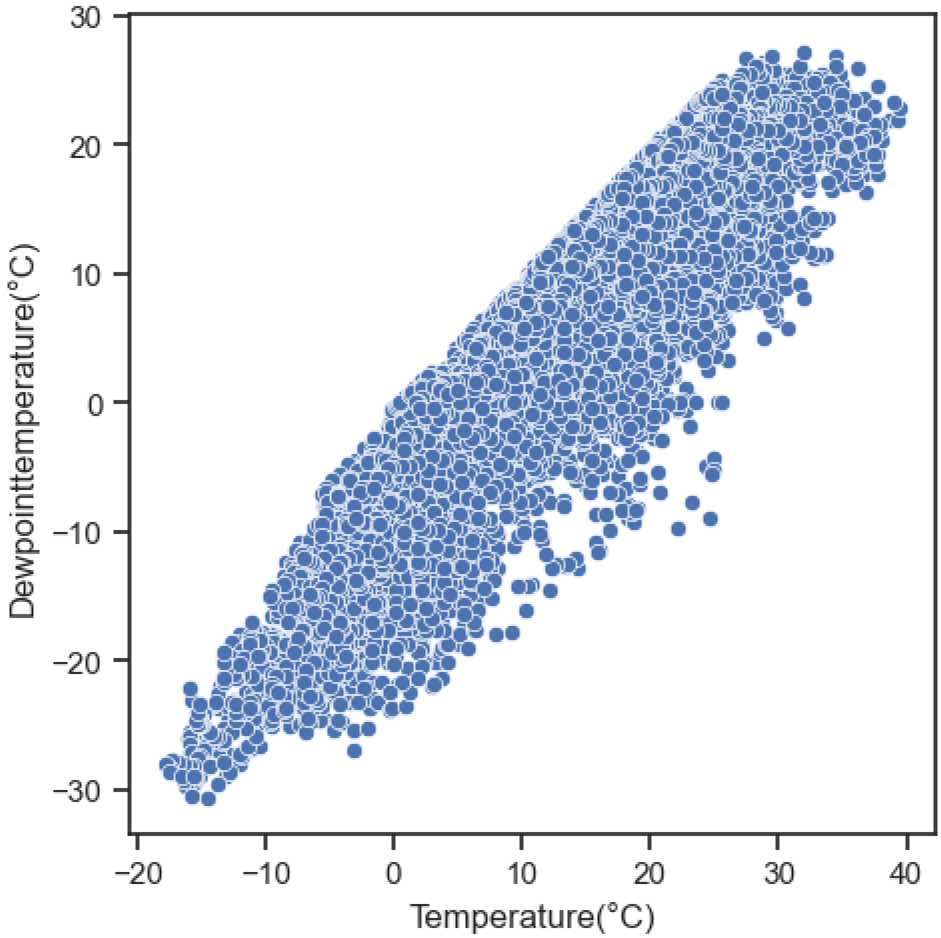
3.2. RQ2: Is there a relationship between Temperature and Dew Point Temperature?

Hypotheses:

> - **Null hypothesis (H_0)** : `Temperature` and `Dewpointtemperature` are not correlated.
> - **Alternative hypothesis (H_1)**: `Temperature` and `Dewpointtemperature` are correlated.

Significance level: 0.05

In [388…
```
sns.scatterplot(x="Temperature(°C)", y="Dewpointtemperature(°C)", data=df);
```



In [389…
```
stats.pearsonr(df['Temperature(°C)'], df['Dewpointtemperature(°C)'])
```
Out[389]:
```
PearsonRResult(statistic=0.9127976489794506, pvalue=0.0)
```

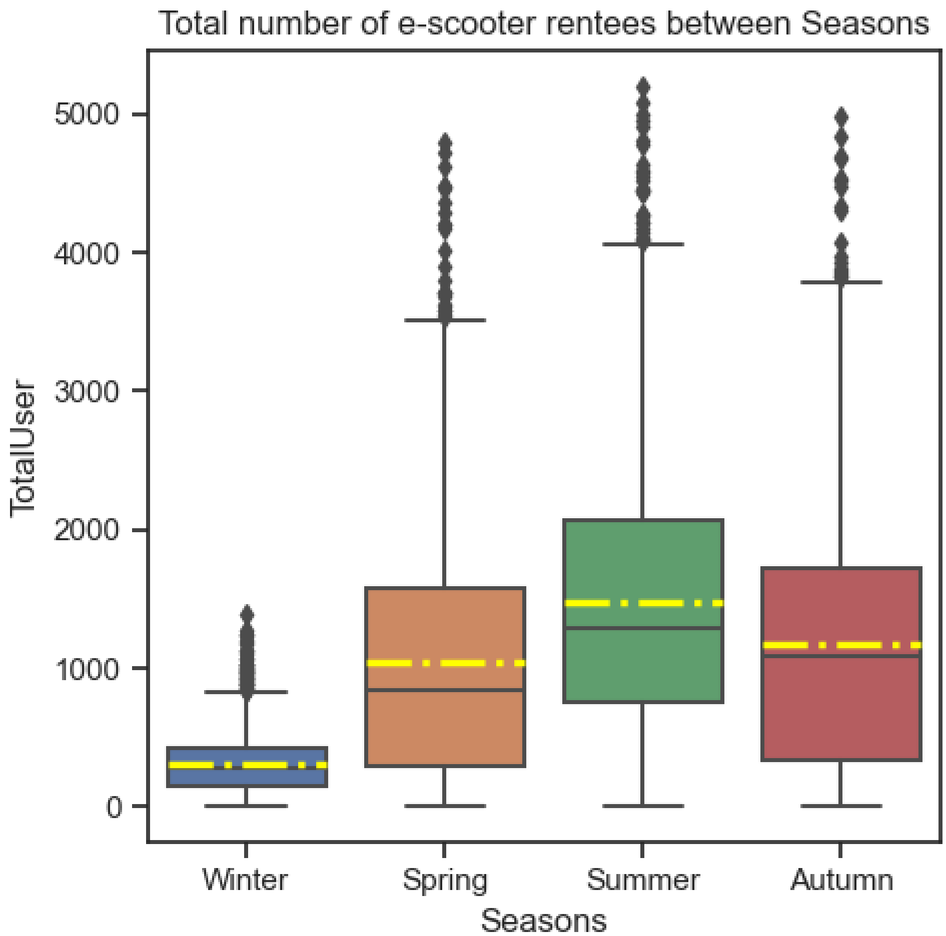
Discussion:

`Temperature` and `Dewpointtemperature` have a strong positive correlation because:

> - p-value is lower than the ***significance level (0.05)***, therefore H_0 ***is REJECTED***.
> - The Pearson correlation coefficient (r) is high: 0.913.

3.3. RQ3: Which season are people most/least likely to rent an e-scooter?

In [390…
```
meanlineprops = dict(linestyle='-.', linewidth=2.5, color='yellow')

sns.boxplot(data = df, x='Seasons', y='TotalUser', showmeans=True, meanprops = meanlinep

plt.title('Total number of e-scooter rentees between Seasons')
```
Out[390]:
```
Text(0.5, 1.0, 'Total number of e-scooter rentees between Seasons')
```



Levene's Test for Homogeneity of Variance

In [391…
```
stats.levene(df['TotalUser'][df['Seasons'] == 'Autumn'],
             df['TotalUser'][df['Seasons'] == 'Spring'],
             df['TotalUser'][df['Seasons'] == 'Summer'],
             df['TotalUser'][df['Seasons'] == 'Winter'],
            )
```

Out[391]:
```
LeveneResult(statistic=696.4275477222299, pvalue=0.0)
```

The Levene's test on four groups of `Seasons` shows significance. The null hypothesis for the Levene's test is ***REJECTED*** since the p-value is less than the significance level (0.05) and infer that at least one pair of groups has uneven variance. Therefore, the traditional ANOVA method to compare the means of four `Seasons` groups cannot be used. A non-parametric version of ANOVA, the Welch's ANOVA will be used instead, since Welch's ANOVA does not assume homogeneity of variance between groups. The same reason for why Games-Howell post-hoc test is used in place of of Tukey HSD.

Hypotheses:

> - **Null hypothesis (H_0)** : The number of `TotalUser` in four seasons have the same mean.
> - **Alternative hypothesis (H_1)**: The number of `TotalUser` in four seasons have different means.

Significance level: 0.05

In [392…
```
pg.welch_anova(dv='TotalUser', between='Seasons', data=df)
```

Out[392]:

	Source	ddof1	ddof2	F	p-unc	np2
0	Seasons	3	3985.332533	1854.376926	0.0	0.211406

Discussion:

The p-value from the Welch's ANOVA test is less than the significance level (0.05), therefore H_0 that says all seasons have the same mean is ***REJECTED***. However, the test did not point out which season is different from the others. The Games-Howell post-hoc test is conducted to see which season has a different mean.

In [393…

```
pg.pairwise_gameshowell(dv='TotalUser', between='Seasons', data=df)
```

Out[393]:

	A	B	mean(A)	mean(B)	diff	se	T	df	
0	Autumn	Spring	1174.162088	1046.014040	128.148048	27.664993	4.632137	4377.461281	2.209
1	Autumn	Summer	1174.162088	1481.834239	-307.672151	29.164210	-10.549648	4378.403369	0.000
2	Autumn	Winter	1174.162088	311.889928	862.272160	20.531358	41.997815	2405.184500	0.000
3	Spring	Summer	1046.014040	1481.834239	-435.820199	28.526730	-15.277608	4366.141196	1.779
4	Spring	Winter	1046.014040	311.889928	734.124111	19.615296	37.426103	2454.069786	0.000
5	Summer	Winter	1481.834239	311.889928	1169.944311	21.678541	53.967852	2407.489459	2.958

Discussion:

From the Games-Howell post-hoc test, the pair-wise comparison between four groups all give p-values less than the significance level (0.05). Hence, the H_0 is ***REJECTED***. The number of e-scooter rentees differs in all four seasons. Summer has the highest mean of `TotalUser`, and Winter has the lowest mean of `TotalUser`. Therefore, most people would rent an e-scooter in Summer, and the least amount of people would rent an e-scooter in Winter.

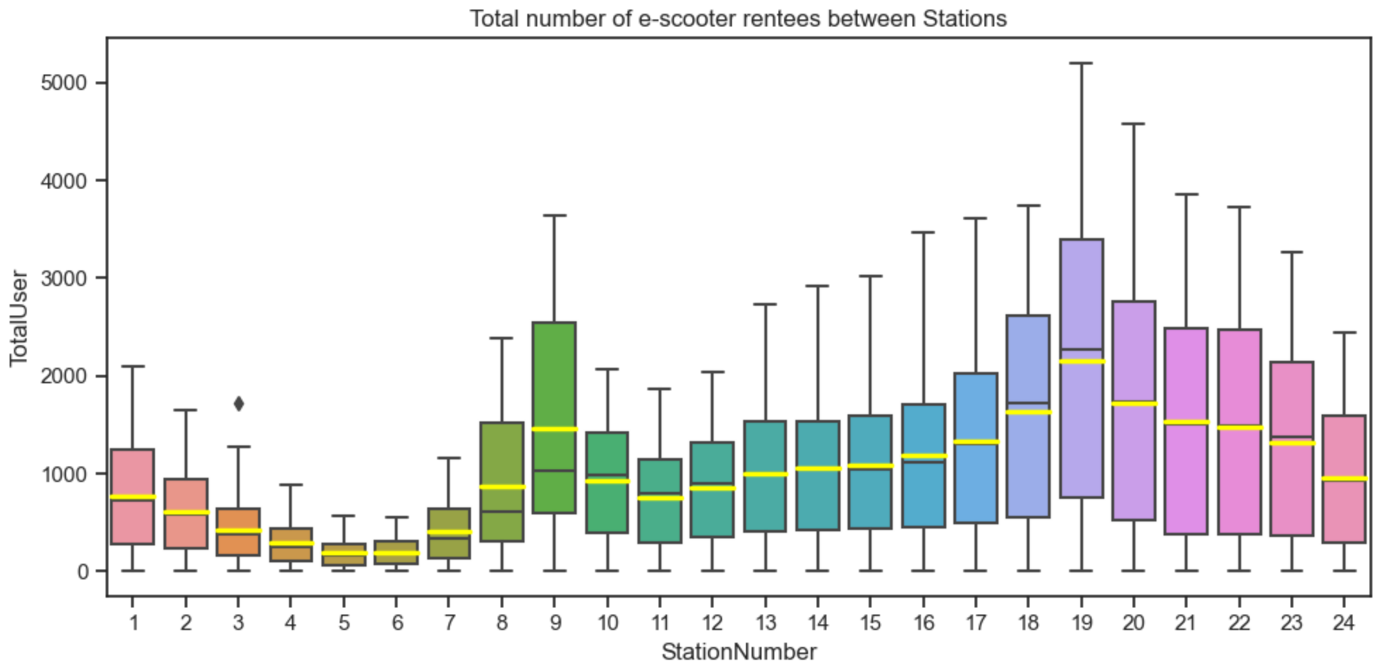
3.4. RQ4: Which station has the most/least e-scooter rent in a single day?

In [394…

```
meanlineprops = dict(linestyle='-', linewidth=2.5, color='yellow')

plt.figure(figsize=(10, 5))
sns.boxplot(data = df, x='StationNumber', y='TotalUser', showmeans=True, meanprops = mea

plt.title('Total number of e-scooter rentees between Stations')
```

Out[394]:

```
Text(0.5, 1.0, 'Total number of e-scooter rentees between Stations')
```



Levene's Test for Homogeneity of Variance

In [395…

```
stats.levene(df['TotalUser'][df['StationNumber'] == 1],
             df['TotalUser'][df['StationNumber'] == 2],
             df['TotalUser'][df['StationNumber'] == 3],
             df['TotalUser'][df['StationNumber'] == 4],
             df['TotalUser'][df['StationNumber'] == 5],
             df['TotalUser'][df['StationNumber'] == 6],
             df['TotalUser'][df['StationNumber'] == 7],
             df['TotalUser'][df['StationNumber'] == 8],
             df['TotalUser'][df['StationNumber'] == 9],
             df['TotalUser'][df['StationNumber'] == 10],
             df['TotalUser'][df['StationNumber'] == 11],
             df['TotalUser'][df['StationNumber'] == 12],
             df['TotalUser'][df['StationNumber'] == 13],
             df['TotalUser'][df['StationNumber'] == 14],
             df['TotalUser'][df['StationNumber'] == 15],
             df['TotalUser'][df['StationNumber'] == 16],
             df['TotalUser'][df['StationNumber'] == 17],
             df['TotalUser'][df['StationNumber'] == 18],
             df['TotalUser'][df['StationNumber'] == 19],
             df['TotalUser'][df['StationNumber'] == 20],
             df['TotalUser'][df['StationNumber'] == 21],
             df['TotalUser'][df['StationNumber'] == 22],
             df['TotalUser'][df['StationNumber'] == 23],
             df['TotalUser'][df['StationNumber'] == 24],
            )
```

Out[395]:

```
LeveneResult(statistic=253.70970410605364, pvalue=0.0)
```

Discussion:

The Levene's test for on 24 groups of `StationNumber` shows significant. Hence, the homogeneity of variance assumption for ANOVA is violated. The Welch's ANOVA method with Games-Howell post-hoc test will be used instead.

Hypotheses:

- **Null hypothesis (H_0)** : The number of `TotalUser` at all stations has the same mean.
- **Alternative hypothesis (H_1)**: The number of `TotalUser` at different stations has different means.

Significance level: 0.05

In [396…
```
pg.welch_anova(dv='TotalUser', between='StationNumber', data=df)
```

Out[396]:

	Source	ddof1	ddof2	F	p-unc	np2
0	StationNumber	23	3140.720638	344.400455	0.0	0.291106

Discussion:

There is a mean difference of `TotalUser` between 24 stations because:

> - The p-value is smaller than the ***significance level (0.05)***, therefore H_0 is ***REJECTED***.

The Games-Howell post-hoc test is conducted too observe the mean difference between pairs of stations:

In [397…
```
post_hoc = pg.pairwise_gameshowell(dv='TotalUser', between='StationNumber', data=df)
post_hoc
```

Out[397]:

	A	B	mean(A)	mean(B)	diff	se	T	df	pval
0	1	2	775.367322	607.760789	167.606533	35.093000	4.776067	683.899915	5.511823e-04
1	1	3	775.367322	427.545645	347.821677	31.883108	10.909278	575.449159	0.000000e+00
2	1	4	775.367322	286.924196	488.443127	29.693542	16.449473	468.127072	1.140199e-13
3	1	5	775.367322	185.375139	589.992183	28.510890	20.693573	405.312457	1.176836e-13
4	1	6	775.367322	195.062918	580.304404	28.609392	20.283703	410.562484	0.000000e+00
...	...	...	...	...	...	...	...	...	...
271	21	23	1535.310835	1316.807743	218.503091	77.998775	2.801366	702.509488	4.378205e-01
272	21	24	1535.310835	956.590393	578.720442	70.087838	8.257074	594.260075	2.371436e-13
273	22	23	1477.202369	1316.807743	160.394626	75.481552	2.124951	714.857019	9.032775e-01
274	22	24	1477.202369	956.590393	520.611976	67.275262	7.738535	614.606957	1.148903e-11
275	23	24	1316.807743	956.590393	360.217350	61.265197	5.879641	663.376272	1.772030e-06

276 rows × 10 columns

In [398…
```
max_a = post_hoc['mean(A)'].max() # Find station with highest mean in column A
max_b = post_hoc['mean(B)'].max() # Find station with highest mean in column B
post_hoc[post_hoc['mean(A)'] == max_a]
```

Out[398]:

	A	B	mean(A)	mean(B)	diff	se	T	df	pval
261	19	20	2159.936204	1721.145623	438.790581	101.711866	4.314055	702.729743	4.228518e-03
262	19	21	2159.936204	1535.310835	624.625369	98.600027	6.334941	682.156573	1.182269e-07

263	19	22	2159.936204	1477.202369	682.733835	96.621017	7.066101	663.885607	1.112933e-09
264	19	23	2159.936204	1316.807743	843.128461	92.536936	9.111264	615.225016	5.706546e-14
265	19	24	2159.936204	956.590393	1203.345811	85.974303	13.996575	510.105284	0.000000e+00

In [399…

```python
post_hoc[post_hoc['mean(B)'] == max_b]
```

Out[399]:

	A	B	mean(A)	mean(B)	diff	se	T	df	pva
17	1	19	775.367322	2159.936204	-1384.568882	82.870135	-16.707694	452.818205	1.233458e-1
39	2	19	607.760789	2159.936204	-1552.175415	81.026541	-19.156383	417.986536	1.464384e-1
60	3	19	427.545645	2159.936204	-1732.390559	79.688859	-21.739432	392.666736	7.283063e-1
80	4	19	286.924196	2159.936204	-1873.012009	78.838367	-23.757621	376.665264	1.476597e-1
99	5	19	185.375139	2159.936204	-1974.561065	78.400590	-25.185538	368.477757	0.000000e+0
117	6	19	195.062918	2159.936204	-1964.873286	78.436464	-25.050508	369.147213	2.396972e-1
134	7	19	407.941231	2159.936204	-1751.994974	79.885975	-21.931196	396.389274	1.757483e-1
150	8	19	869.292658	2159.936204	-1290.643546	86.270119	-14.960493	515.323404	2.950973e-1
165	9	19	1461.331858	2159.936204	-698.604347	97.212641	-7.186353	669.286744	4.911852e-1
179	10	19	924.582997	2159.936204	-1235.353207	83.696159	-14.759975	468.315955	1.488809e-1
192	11	19	750.043349	2159.936204	-1409.892855	81.791550	-17.237635	432.484626	1.632028e-1
204	12	19	855.438433	2159.936204	-1304.497771	82.707977	-15.772333	449.802906	1.511014e-1
215	13	19	999.812587	2159.936204	-1160.123617	84.666249	-13.702315	486.290428	0.000000e+0
225	14	19	1054.097037	2159.936204	-1105.839167	85.568838	-12.923386	502.750732	0.000000e+0
234	15	19	1083.948455	2159.936204	-1075.987749	86.395179	-12.454257	517.610797	1.496581e-1
242	16	19	1180.375499	2159.936204	-979.560705	88.211045	-11.104740	548.994770	1.288969e-1
249	17	19	1328.181609	2159.936204	-831.754595	90.954346	-9.144748	592.588453	0.000000e+0
255	18	19	1628.649563	2159.936204	-531.286641	96.334119	-5.515041	660.947183	1.343739e-0

Station with highest `TotalUser`:

Station 19 has the highest mean of total e-scooter rentees (mean = 2160). The pair-wise comparison (Games-Howell post-hoc test) shows that Station 19's mean of `TotalUser` if different from all other stations (p-value < 0.05). Hence, it can be concluded that Station 19 has the most e-scooter rentees.

In [400…

```python
min_a = post_hoc['mean(A)'].min()  # Find station with lowest mean in column A
min_b = post_hoc['mean(B)'].min()  # Find station with lowest mean in column B
post_hoc[post_hoc['mean(A)'] == min_a]
```

Out[400]:

	A	B	mean(A)	mean(B)	diff	se	T	df	pval
86	5	6	185.375139	195.062918	-9.687779	9.883852	-0.980162	725.593113	9.999990e-01
87	5	7	185.375139	407.941231	-222.566092	18.088132	-12.304537	480.082586	0.000000e+00
88	5	8	185.375139	869.292658	-683.917519	37.255134	-18.357672	387.876179	0.000000e+00
89	5	9	185.375139	1461.331858	-1275.956719	58.272713	-21.896299	372.975172	5.118128e-14
90	5	10	185.375139	924.582997	-739.207858	30.829509	-23.977283	399.858143	1.760814e-13
91	5	11	185.375139	750.043349	-564.668211	25.204549	-22.403424	420.527404	3.017586e-13

92	5	12	185.375139	855.438433	-670.063294	28.036066	-23.900047	409.111690	1.629807e-13
93	5	13	185.375139	999.812587	-814.437448	33.373423	-24.403774	394.242623	6.217249e-15
94	5	14	185.375139	1054.097037	-868.721898	35.601091	-24.401553	390.324029	1.485478e-13
95	5	15	185.375139	1083.948455	-898.573316	37.543822	-23.933986	388.549336	2.160494e-13
96	5	16	185.375139	1180.375499	-995.000360	41.552377	-23.945691	383.925529	1.465494e-13
97	5	17	185.375139	1328.181609	-1142.806470	47.096757	-24.265078	379.421348	6.161738e-14
98	5	18	185.375139	1628.649563	-1443.274424	56.795018	-25.411990	374.537132	9.914292e-14
99	5	19	185.375139	2159.936204	-1974.561065	78.400590	-25.185538	368.477757	0.000000e+00
100	5	20	185.375139	1721.145623	-1535.770484	65.503552	-23.445606	369.850832	0.000000e+00
101	5	21	185.375139	1535.310835	-1349.935696	60.558871	-22.291296	373.252341	2.756684e-13
102	5	22	185.375139	1477.202369	-1291.827230	57.280299	-22.552732	374.356890	0.000000e+00
103	5	23	185.375139	1316.807743	-1131.432604	50.084889	-22.590299	377.604106	1.341149e-13
104	5	24	185.375139	956.590393	-771.215254	36.564906	-21.091679	389.925489	4.696243e-14

In [401…
```
post_hoc[post_hoc['mean(B)'] == min_b]
```

Out[401]:

	A	B	mean(A)	mean(B)	diff	se	T	df	pval	hedg
3	1	5	775.367322	185.375139	589.992183	28.510890	20.693573	405.312457	1.176836e-13	1.5360
25	2	5	607.760789	185.375139	422.385650	22.598935	18.690511	435.416206	5.917489e-14	1.3821
46	3	5	427.545645	185.375139	242.170506	17.196680	14.082399	493.798347	0.000000e+00	1.0413
66	4	5	286.924196	185.375139	101.549057	12.684624	8.005681	615.370449	1.944667e-12	0.5919

Stations with lowest `TotalUser`:

Station 5 has the lowest mean of `TotalUser` (mean = 185). The pair-wise comparison (Games-Howell post-hoc test) shows that Station 5's mean of `TotalUser` is different from other stations (p-value < 0.05), except Station 6 (p-value = 0.99). Hence, it can be concluded that Station 5 and Station 6 has the lowest number of e-scooter rentees.

3.5. RQ5: Does `Seasons` affect e-scooter stations' closure?

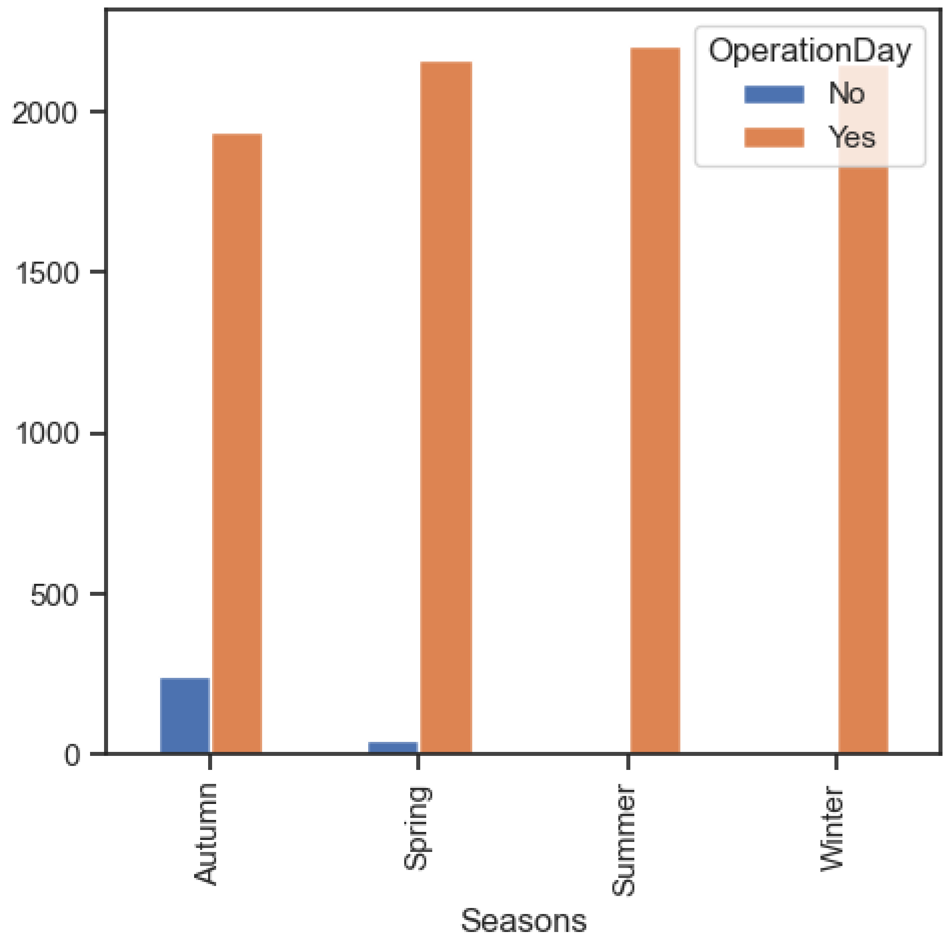
In [402…
```
chi2table = pd.crosstab(df['Seasons'], df['OperationDay'])
chi2table
```

Out[402]:

OperationDay	No	Yes
Seasons		
Autumn	247	1937
Spring	48	2160
Summer	0	2208
Winter	0	2150

In [403…
```
chi2table.plot.bar()
```

Out[403]: <AxesSubplot:xlabel='Seasons'>



Hypotheses:

> - **Null hypothesis (H_0)** : `OperationDay` is independent of `Seasons`.
> - **Alternative hypothesis (H_1)**: `OperationDay` is dependent on `Seasons`.

Significance level: 0.05

In [404…
```
result = stats.chi2_contingency(chi2table)

print('statistic: ', result[0])
print('p-value: ', result[1])
print('dof: ', result[2])
print('Expected frequency:\n', result[3])
```

```
statistic:  584.2133462356013
p-value:  2.664716557182926e-126
dof:  3
Expected frequency:
 [[  73.632       2110.368     ]
 [  74.44114286 2133.55885714]
 [  74.44114286 2133.55885714]
 [  72.48571429 2077.51428571]]
```

Discussion:

`Seasons` does affect whether a station is open or not because:

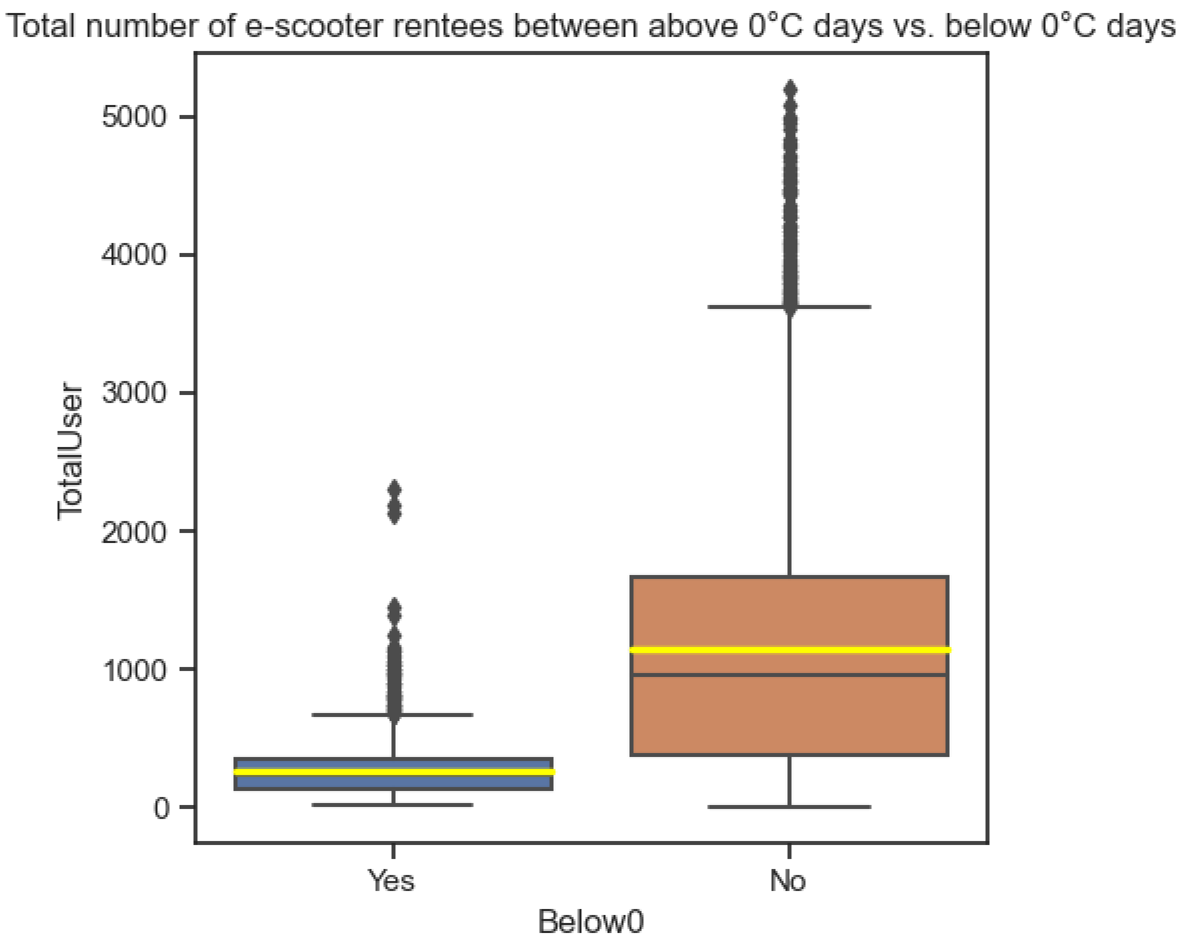
- p-value is lower than the ***significance level (0.05)***, therefore H_0 ***is REJECTED***. Most of the time the stations were closed during Autumn.

The stations were most likely closed in Autumn than any other seasons.

3.6. RQ6: Are people more likely to rent an e-scooter when the temperature is above 0°C?

In [405…
```
meanlineprops = dict(linestyle='-', linewidth=2.5, color='yellow')

sns.boxplot(data = df, x='Below0', y='TotalUser', showmeans=True, meanprops = meanlinepr

plt.title('Total number of e-scooter rentees between above 0°C days vs. below 0°C days')
```

Out[405]:
```
Text(0.5, 1.0, 'Total number of e-scooter rentees between above 0°C days vs. below 0°C
days')
```



Levene's Test for Homogeneity of Variance

In [406…
```
stats.levene(df['TotalUser'][df['Below0'] == 'Yes'],
             df['TotalUser'][df['Below0'] == 'No'], )
```

Out[406]:
```
LeveneResult(statistic=1375.4131268371218, pvalue=9.396357590649263e-280)
```

Discussion:

The p-value of the Levene's test is less than the significance level of 0.05, which means the variance between two testing groups are not equal. Hence, the Welch's t-test is conducted because it does not assume homogeneity of variance. This t-test is also one-tailed to compare if one condition has higher mean than the other.

Hypotheses:

> - **Null hypothesis (H_0)** : The mean of `TotalUser` when the temperature is below 0°C is NOT less than the mean of `TotalUser` when the temperature is above 0°C.
> - **Alternative hypothesis (H_1)**: The mean of `TotalUser` when the temperature is below 0°C is less than the mean of `TotalUser` when the temperature is above 0°C.

Significance level: 0.05

In [407…
```
stats.ttest_ind(df['TotalUser'][df['Below0'] == 'Yes'],
                df['TotalUser'][df['Below0'] == 'No'],
                equal_var=False,
                alternative='less')
```

Out[407]:
```
Ttest_indResult(statistic=-70.40315736414858, pvalue=0.0)
```

Discussion:

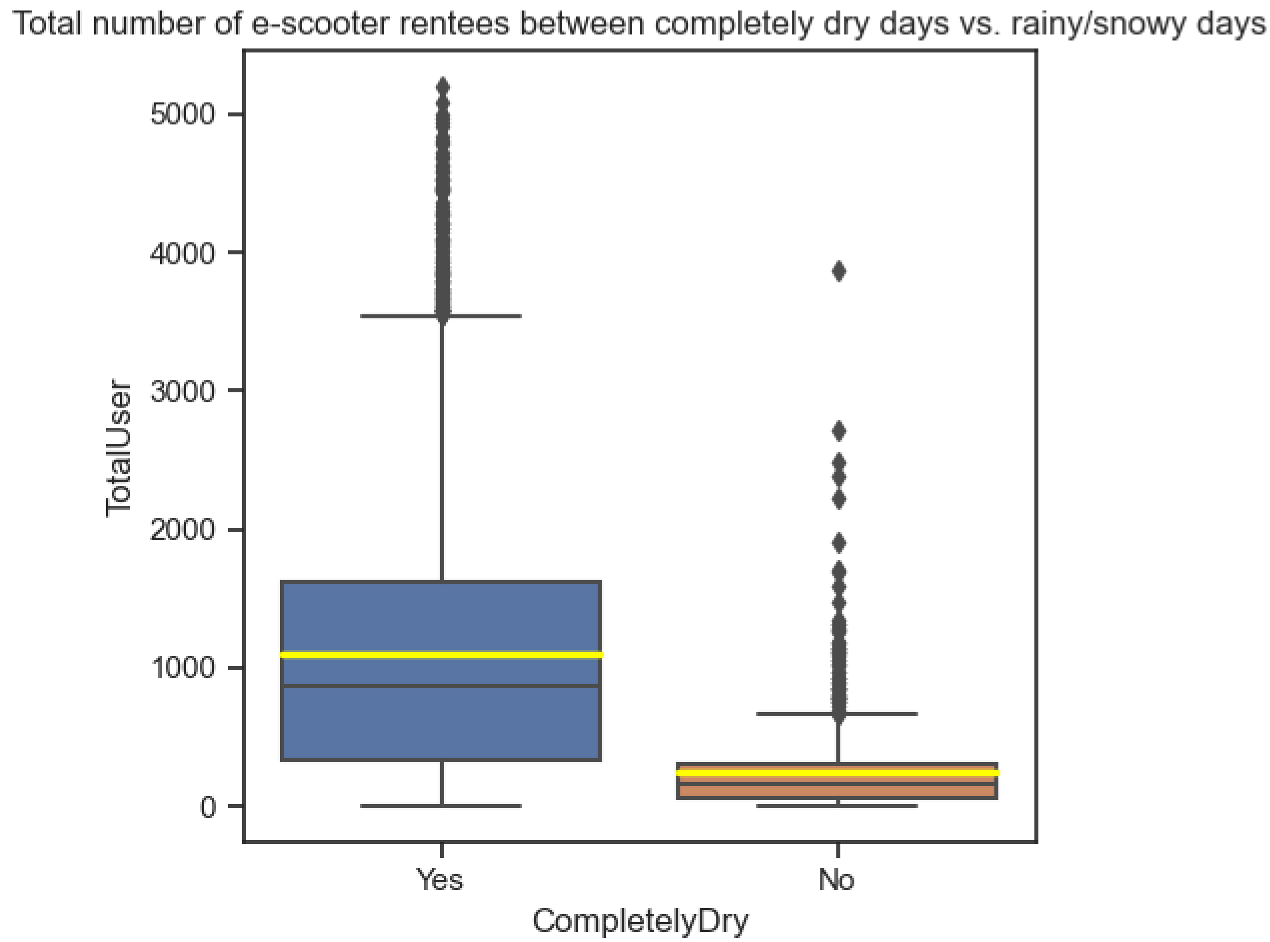
People tend to rent an e-scooter when the temperature is above 0°C because:

> - p-value is lower than the ***significance level (0.05)***, therefore H_0 ***is REJECTED***. The mean of `TotalUser` when the temperature is below 0°C is less than the mean of `TotalUser` when the temperature is above 0°C.

3.7. RQ7: Are people more likely to rent an e-scooter on completely dry days (no rain, no snow)?

In [408…
```
meanlineprops = dict(linestyle='-', linewidth=2.5, color='yellow') # Mean line propertie

sns.boxplot(data = df,
            x='CompletelyDry',
            y='TotalUser',
            showmeans=True,
            meanprops = meanlineprops,
            meanline=True)

plt.title('Total number of e-scooter rentees between completely dry days vs. rainy/snowy
```

Out[408]:
```
Text(0.5, 1.0, 'Total number of e-scooter rentees between completely dry days vs. rain
y/snowy days')
```



From the scatterplots in RQ1, 2, and 3, it looks like the number of e-scooter rentees decreases as `Rainfall` or `Snowfall` increases, and the amount of rentees seems extremely high there is no rain or snow. However, the correlation is not linear, therefore the Pearson correlation tests show very weak correlation between `Rainfall` or `Snowfall` and the number of users. Hence, this time, an independent t-test is carried out to see if people are more likely to rent an e-scooter on completely no rain and no snow days.

Levene's Test for Homogeneity of Variance

A Levene's test must be conducted before the t-test to check the homogeneity of variance of the two samples: `TotalUser` on days without rain/snow and `TotalUser` on days with rain/snow.

In [409…
```
stats.levene(df['TotalUser'][df['CompletelyDry'] == 'Yes'],
             df['TotalUser'][df['CompletelyDry'] == 'No']
            )
```

Out[409]:
```
LeveneResult(statistic=719.4165738038259, pvalue=2.303843517131762e-152)
```

The p-value from the Levene's test is less than the significance level (0.05), hence the test yields significance, which means that the two samples do not have equal variance. Hence, the independent t-test must not assume equal variance between groups, and the Welch's one-tailed t-test is conducted.

Hypotheses:

> - **Null hypothesis (H_0)** : The mean of `TotalUser` on dry (no rain, no snow) days is NOT less than the mean of `TotalUser` on days with rain or snow.

- **Alternative hypothesis (H_1)**: The mean of `TotalUser` on dry (no rain, no snow) days is less than the mean of `TotalUser` on days with rain or snow.

Significance level: 0.05

In [410…

```
stats.ttest_ind(df['TotalUser'][df['CompletelyDry'] == 'Yes'],
                df['TotalUser'][df['CompletelyDry'] == 'No'],
                equal_var=False,
                alternative='less')
```

Out[410]:

```
Ttest_indResult(statistic=56.84253060284808, pvalue=1.0)
```

Discussion:

The mean of `TotalUser` who rented e-scooter is higher when there is no rain and snow on a given day because:

> - p-value is higher than the ***significance level (0.05)***, therefore H_0 ***is NOT rejected***. People tends to rent more e-scooters on dry days without rain and snow.

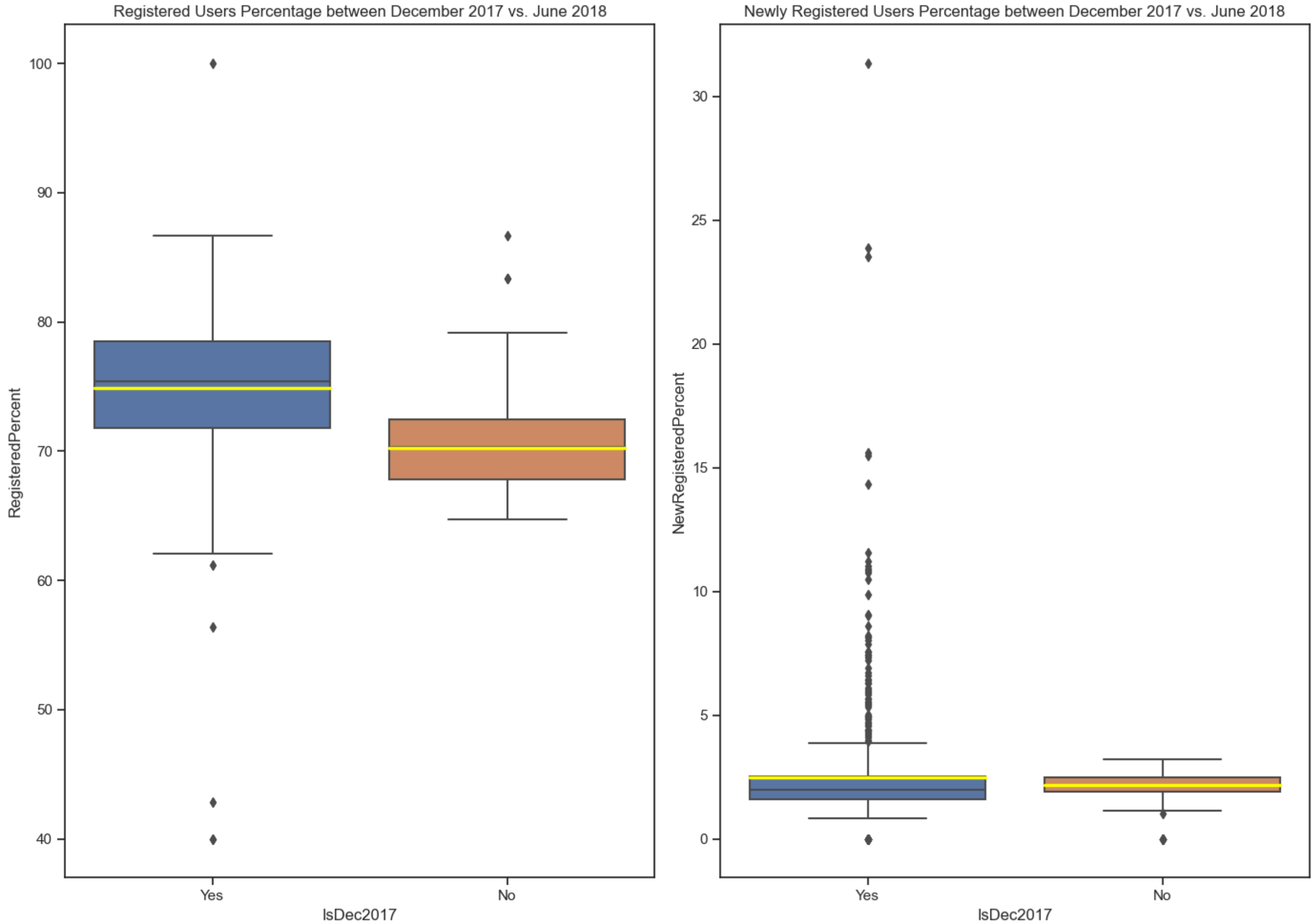
3.8. RQ8: Has the percentage of registered/newly registered user increase after 6 months (from Dec 2017 to Jun 2018)?

In [411…

```
test_groups = df[(df['IsDec2017'] == 'Yes') | (df['IsJun2018'] == 'Yes')] # Extract rele

meanlineprops = dict(linestyle='-', linewidth=2.5, color='yellow') # Mean line propertie
fig, axes = plt.subplots(1, 2, figsize=(14, 10))   # Subplot 1x2

sns.boxplot(data = test_groups,
            x='IsDec2017',
            y='RegisteredPercent',
            showmeans=True,
            meanprops = meanlineprops,
            meanline=True,
            ax=axes[0])

sns.boxplot(data=test_groups,
            x='IsDec2017',
            y='NewRegisteredPercent',
            showmeans=True,
            meanprops = meanlineprops,
            meanline=True,
            ax=axes[1])

axes[0].set_title('Registered Users Percentage between December 2017 vs. June 2018')
axes[1].set_title('Newly Registered Users Percentage between December 2017 vs. June 2018
```

Out[411]:

```
Text(0.5, 1.0, 'Newly Registered Users Percentage between December 2017 vs. June 2018')
```



Registered Users:

Levene's Test for Homogeneity of Variance

In [412…

```
stats.levene(df['RegisteredPercent'][df['IsDec2017'] == 'Yes'],
             df['RegisteredPercent'][df['IsJun2018'] == 'Yes'])
```

Out[412]:

```
LeveneResult(statistic=113.46522187510439, pvalue=1.457700574426362e-25)
```

The p-value of the Levene's test is less than the significance level of 0.05, which means the two groups do not have equal variances. Therefore, the Welch's one-tailed t-test is conducted to test if the mean of registered user percentage in December 2017 is higher than the mean of registered user percentage in June 2018.

Hypotheses:

> - **Null hypothesis (H_0)** : The mean of `RegisteredPercent` in December 2017 days is NOT less than the mean of `RegisteredPercent` in June 2018.
> - **Alternative hypothesis (H_1)**: The mean of `RegisteredPercent` in December 2017 days is less than the mean of `RegisteredPercent` in June 2018.

Significance level: 0.05

In [413…

```
stats.ttest_ind(df['RegisteredPercent'][df['IsDec2017'] == 'Yes'],
                df['RegisteredPercent'][df['IsJun2018'] == 'Yes'],
                equal_var=False,
                alternative='less')
```

Out[413]: Ttest_indResult(statistic=21.93894375694393, pvalue=1.0)

Newly Registered Users:

Levene's Test for Homogeneity of Variance

In [414…
```
stats.levene(df['NewRegisteredPercent'][df['IsDec2017'] == 'Yes'],
             df['NewRegisteredPercent'][df['IsJun2018'] == 'Yes'])
```

Out[414]: LeveneResult(statistic=67.69986737060182, pvalue=4.186617216533479e-16)

The p-value of the Levene's test is less than the significance level of 0.05, which means the two groups do not have equal variances. Therefore, the Welch's one-tailed t-test is conducted to test if the mean of newly registered user percentage in December 2017 is higher than the mean of newly registered user percentage in June 2018.

Hypotheses:

> - **Null hypothesis (H_0)** : The mean of `NewRegisteredPercent` in December 2017 days is NOT less than the mean of `NewRegisteredPercent` in June 2018.
> - **Alternative hypothesis (H_1)**: The mean of `NewRegisteredPercent` in December 2017 days is less than the mean of `NewRegisteredPercent` in June 2018.

Significance level: 0.05

In [415…
```
stats.ttest_ind(df['NewRegisteredPercent'][df['IsDec2017'] == 'Yes'],
                df['NewRegisteredPercent'][df['IsJun2018'] == 'Yes'],
                equal_var=False,
                alternative='less')
```

Out[415]: Ttest_indResult(statistic=3.2406307544919377, pvalue=0.9993785949748367)

Discussion:

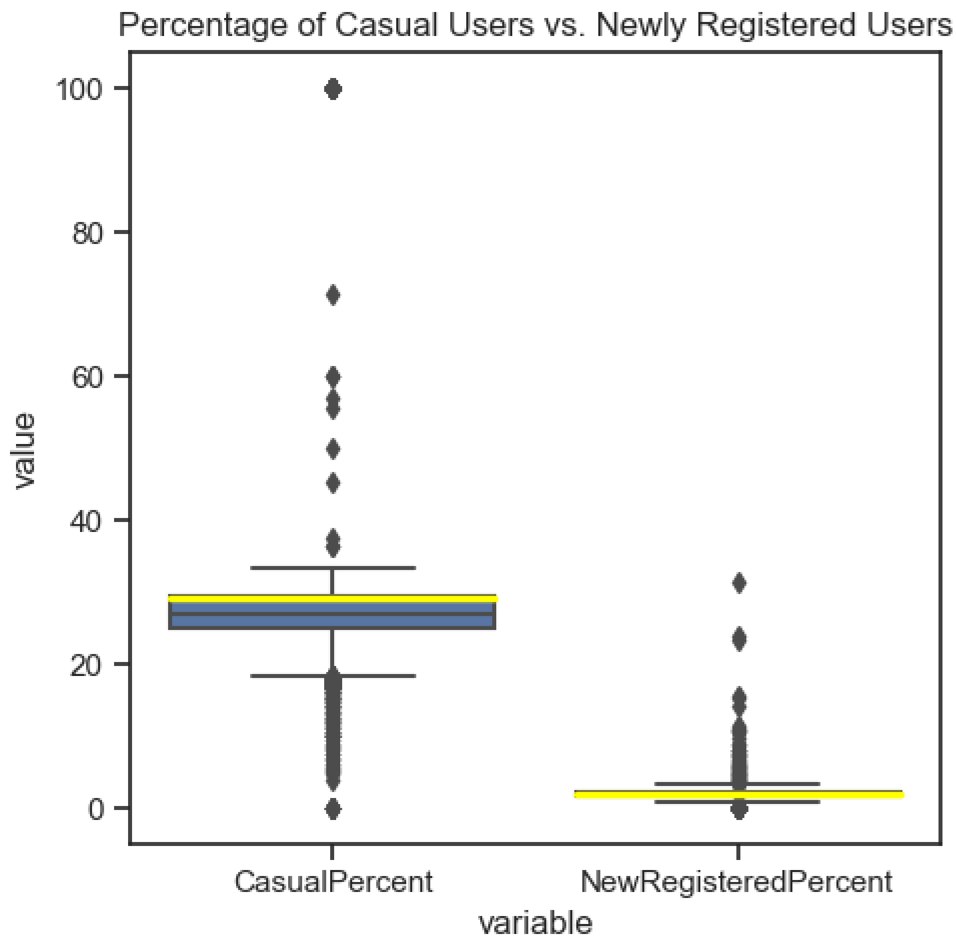
The mean of both `RegisteredPercent` and `NewRegisteredPercent` in December 2017 were both higher than that of June 2018 because:

> - p-value is higher than the ***significance level (0.05)***, therefore H_0 ***is NOT rejected***. The percentage of registered users and newly registered users had dropped over the course of 6 months (Dec 2017 - Jun 2018).

3.9. RQ9: Do unregistered rentees prefer to register or stay casual?

In [416…
```
test_groups = df[['CasualPercent', 'NewRegisteredPercent']] # Extract 2 relevant fields

meanlineprops = dict(linestyle='-', linewidth=2.5, color='yellow') # Properties of the m

sns.boxplot(data=test_groups.melt(),
            x='variable',
            y='value',
            showmeans=True,
```

```
            meanprops = meanlineprops,
            meanline=True,);

plt.title('Percentage of Casual Users vs. Newly Registered Users');
```



Levene's Test for Homogeneity of Variance

In [417…

```
stats.levene(df['CasualPercent'],
             df['NewRegisteredPercent'])
```

Out[417]:

```
LeveneResult(statistic=1081.1149972574658, pvalue=4.0115193350344095e-230)
```

The p-value of the Levene's test is less than the significance level of 0.05, which means the two groups do not have equal variances. Therefore, the Welch's one-tailed t-test is conducted to test if the mean of `CasualUser` is higher than the mean of `Newregistereduser`.

Hypotheses:

> - **Null hypothesis (H_0)** : The mean of `CasualPercent` is NOT less than the mean of `NewRegisteredPercent`.
> - **Alternative hypothesis (H_1)**: The mean of `CasualPercent` is less than the mean of `NewRegisteredPercent`.

Significance level: 0.05

In [418…

```
stats.ttest_ind(df['CasualPercent'],
                df['NewRegisteredPercent'],
                equal_var=False,
                alternative='less')
```

```
Out[418]: Ttest_indResult(statistic=183.74693693041166, pvalue=1.0)
```

Discussion:

The mean of `CasualPercent` higher than the mean of `NewRegisteredPercent` because:

> - p-value is higher than the ***significance level (0.05)***, therefore H_0 ***is NOT rejected***. Unregistered users were not keen on registering to e-scooter stations.

4. References

- [1] World: Highest Temperature

- [2] The Most Humid Cities In The World, Mapped