Bike-Sharing Business Analysis (Python Version)

Date: September 18, Thursday to September 23, Tuesday, 2025
Aims & Goals: This .ipynb file serves as a supplement to the Bike-Sharing Business Analysis project, showcasing a fundamental analytical workflow using Python. Additionally, a comparative analysis between using Python and Google Sheets for this bike-sharing business analysis is conducted during the project.
Self-directed professional: Qi Zhou
Email: qiqizhou1996@gmail.com
LinkedIn: www.linkedin.com/in/qi-zhou-1996to2096
Github: https://github.com/QiZhou1996/a-bike-sharing-business-analysis-project

Introduction:

This Bike-Sharing Business Analysis (Python Version) consists of three sequential steps: first, importing packages and loading the dataset; second, understanding the dataset through descriptive statistics and data cleaning; and lastly, performing exploratory data analysis (EDA) and data visualizations. For the insight report of this analysis, please see the above *LinkedIn* or *Github* link.
From data cleaning, including removing missing values, identifying duplicates, and handling outliers, to visualizations, comprising boxplots, bar charts, pie charts, countplots and scatterplots, key findings are summarized and compared (with those in Google Sheets).

Data Analysis using Python:

Step 1: Import Packages and Load Dataset

In [1]:
```
# Import packages

# For data manipulation
import numpy as np
import pandas as pd

# For data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset into a dataframe
raw_data = pd.read_csv("raw-data.csv")

# Display the first five rows of the raw data and check if the data is lo
raw_data.head()
```

Out[1]:

	Start Time	Stop Time	Start Station ID	Start Station Name	End Station ID	End Station Name	Bike ID	User Type	Birth Year	Ag
0	01-01-17 00:38	1-1-17 01:03	3194	McGinley Square	3271	Danforth Light Rail	24668	Subscriber	1961	6
1	01-01-17 01:47	01-01-17 01:58	3183	Exchange Place	3203	Hamilton Park	26167	Subscriber	1993	2
2	01-01-17 01:47	01-01-17 01:58	3183	Exchange Place	3203	Hamilton Park	26167	Subscriber	1993	2
3	01-01-17 01:56	01-01-17 02:00	3186	Grove St PATH	3270	Jersey & 6th St	24604	Subscriber	1970	5
4	1-1-17 02:12	01-01-17 02:23	3270	Jersey & 6th St	3206	Hilltop	24641	Subscriber	1978	4

Step 2: Understand the Dataset through Descriptive Statistics and Data Cleaning

In [2]:

```
# Gather the basic information about the raw data
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20400 entries, 0 to 20399
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Start Time           20400 non-null  object
 1   Stop Time            20400 non-null  object
 2   Start Station ID     20400 non-null  int64 
 3   Start Station Name   20400 non-null  object
 4   End Station ID       20400 non-null  int64 
 5   End Station Name     20399 non-null  object
 6   Bike ID              20400 non-null  int64 
 7   User Type            20400 non-null  object
 8   Birth Year           20400 non-null  int64 
 9   Age                  20400 non-null  int64 
 10  Age Groups           20400 non-null  object
 11  Trip Duration        20400 non-null  int64 
 12  Trip_Duration_in_min 20400 non-null  object
 13  Month                20400 non-null  int64 
 14  Season               20400 non-null  object
 15  Temperature          20400 non-null  int64 
 16  Weekday              20400 non-null  object
dtypes: int64(8), object(9)
memory usage: 2.6+ MB
```

In [3]:

```
# Gather the descriptive statistics about the raw data
raw_data.describe(include='all')
```

Out[3]:

	Start Time	Stop Time	Start Station ID	Start Station Name	End Station ID	End Station Name	Bike
count	20400	20400	20400.000000	20400	20400.000000	20399	20400.0000
unique	15746	16039	NaN	50	NaN	56	Na
top	20-03-17 17:39	09-03-17 08:20	NaN	Grove St PATH	NaN	Grove St PATH	Na
freq	7	7	NaN	2544	NaN	3313	Na
mean	NaN	NaN	3215.863627	NaN	3211.439510	NaN	25301.7326
std	NaN	NaN	34.563120	NaN	82.707121	NaN	989.9742
min	NaN	NaN	3183.000000	NaN	152.000000	NaN	15084.0000
25%	NaN	NaN	3186.000000	NaN	3186.000000	NaN	24523.0000
50%	NaN	NaN	3203.000000	NaN	3202.000000	NaN	24679.0000
75%	NaN	NaN	3267.000000	NaN	3220.000000	NaN	26220.0000
max	NaN	NaN	3281.000000	NaN	3442.000000	NaN	29296.0000

In [4]:

```
# Check missing values
raw_data.isna().sum()
```

Out[4]:

```
Start Time              0
Stop Time               0
Start Station ID        0
Start Station Name      0
End Station ID          0
End Station Name        1
Bike ID                 0
User Type               0
Birth Year              0
Age                     0
Age Groups              0
Trip Duration           0
Trip_Duration_in_min    0
Month                   0
Season                  0
Temperature             0
Weekday                 0
dtype: int64
```

Finding 1:

There is one missing value that can be removed, matching the way missing values are handled in Google Sheets.

In [5]:

```
# Remove missing values
raw_dropna = raw_data.dropna()
#raw_dropna.info()
```

In [6]:

```
# Check duplicates
raw_dropna.duplicated().sum()
```

```python
# Check some rows containing duplicates as needed
#raw_dropna[raw_dropna.duplicated()].head()
```

Out[6]:

```
np.int64(1950)
```

Finding 2:
Note that Pandas here identifies 1,950 duplicates, whereas Google Sheets identifies 3,555 duplicates.

On the one hand, it is critically important to pay attention to whether different methods of handling duplicates affect the analysis results; on the other hand, from the perspective of de-duplicating techniques, it is worth further study.

In [7]:

```python
# Remove duplicates
data_deduplicated = raw_dropna.drop_duplicates(keep='first')
#data_deduplicated.info()
```

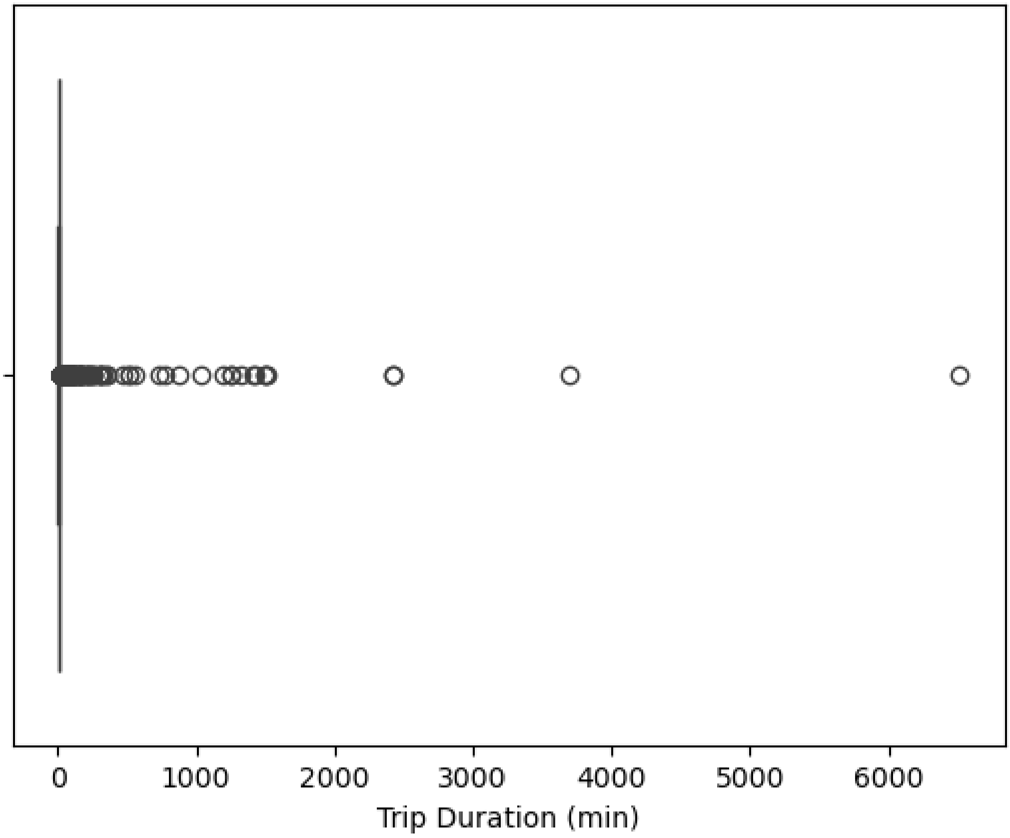
In [21]:

```python
# Check outliers of the "Trip_Duration_in_min" variable
data_deduplicated.loc[:,'Trip Duration (min)'] = data_deduplicated.loc[:,
data_deduplicated.info()
sns.boxplot(x=data_deduplicated['Trip Duration (min)'])
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18449 entries, 0 to 20399
Data columns (total 18 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Start Time            18449 non-null  object
 1   Stop Time             18449 non-null  object
 2   Start Station ID      18449 non-null  int64 
 3   Start Station Name    18449 non-null  object
 4   End Station ID        18449 non-null  int64 
 5   End Station Name      18449 non-null  object
 6   Bike ID               18449 non-null  int64 
 7   User Type             18449 non-null  object
 8   Birth Year            18449 non-null  int64 
 9   Age                   18449 non-null  int64 
 10  Age Groups            18449 non-null  object
 11  Trip Duration         18449 non-null  int64 
 12  Trip_Duration_in_min  18449 non-null  object
 13  Month                 18449 non-null  int64 
 14  Season                18449 non-null  object
 15  Temperature           18449 non-null  int64 
 16  Weekday               18449 non-null  object
 17  Trip Duration (min)   18449 non-null  int64 
dtypes: int64(9), object(9)
memory usage: 2.7+ MB
```

Out[21]:

```
<Axes: xlabel='Trip Duration (min)'>
```



Finding 3:
Note that converting the *Object* variable of "Trip_Duration_in_min" to the *Int* variable of "Trip Duration (min)" leads to no box being generated in the boxplot for identifying outliers; interestingly, the original *Object* variable of "Trip_Duration_in_min" can produce a box in the boxplot, which is worth further research.

In addition, based on practical significance, a maximum of over 6,000 is handled as an outlier in Google Sheets. Regarding outliers, therefore, the same treatment is done here.

In [9]:
```
# Remove outliers through Boolean Masking
mask_for_non_outliers = data_deduplicated['Trip Duration (min)'] < 6000
data_without_outliers = data_deduplicated[mask_for_non_outliers]
data_without_outliers.info()

# After data cleaning, a new dataframe named "data_cleaned" is created fo
data_cleaned = data_without_outliers.reset_index(drop=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18448 entries, 0 to 20399
Data columns (total 18 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Start Time            18448 non-null  object
 1   Stop Time             18448 non-null  object
 2   Start Station ID      18448 non-null  int64 
 3   Start Station Name    18448 non-null  object
 4   End Station ID        18448 non-null  int64 
 5   End Station Name      18448 non-null  object
 6   Bike ID               18448 non-null  int64 
 7   User Type             18448 non-null  object
 8   Birth Year            18448 non-null  int64 
 9   Age                   18448 non-null  int64 
 10  Age Groups            18448 non-null  object
 11  Trip Duration         18448 non-null  int64 
 12  Trip_Duration_in_min  18448 non-null  object
 13  Month                 18448 non-null  int64 
 14  Season                18448 non-null  object
 15  Temperature           18448 non-null  int64 
 16  Weekday               18448 non-null  object
 17  Trip Duration (min)   18448 non-null  int64 
dtypes: int64(9), object(9)
memory usage: 2.7+ MB
```

Step 3: Perform Exploratory Data Analysis (EDA) and Data Visualizations

In [10]:
```
# Gather the basic information about the cleaned data
data_cleaned.info()
```

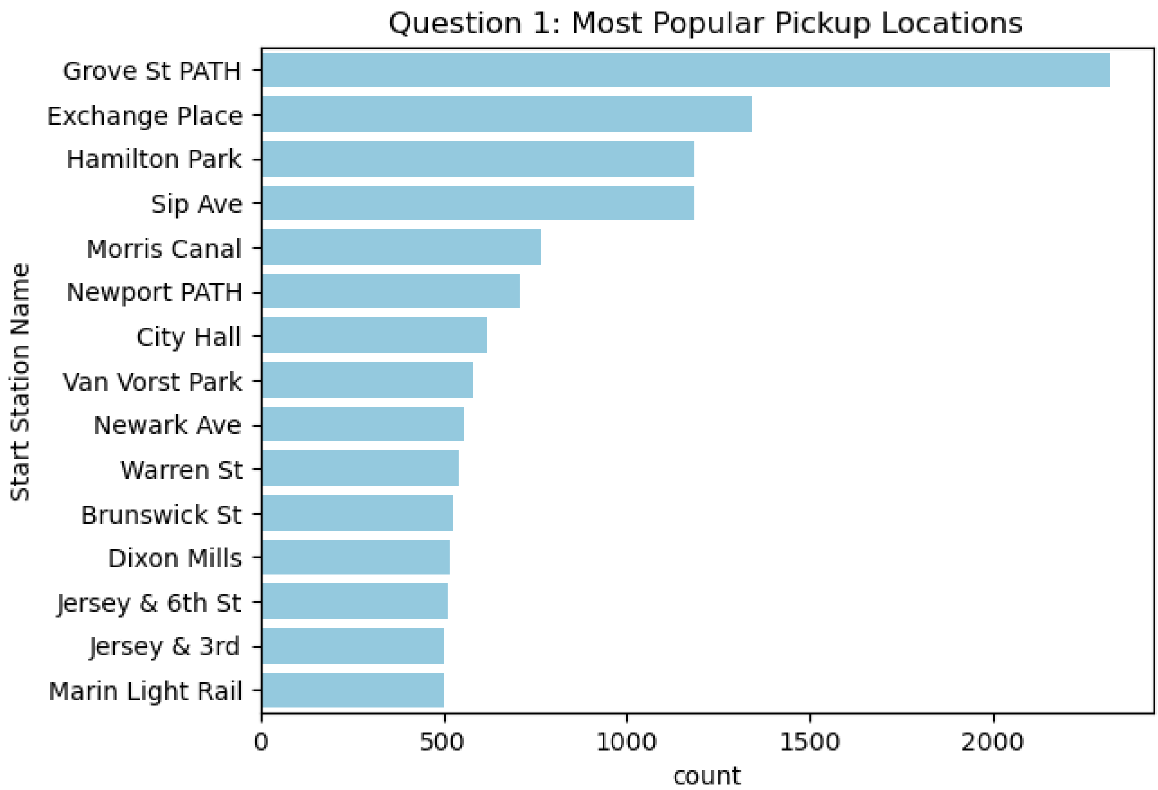
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18448 entries, 0 to 18447
Data columns (total 18 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Start Time            18448 non-null  object
 1   Stop Time             18448 non-null  object
 2   Start Station ID      18448 non-null  int64 
 3   Start Station Name    18448 non-null  object
 4   End Station ID        18448 non-null  int64 
 5   End Station Name      18448 non-null  object
 6   Bike ID               18448 non-null  int64 
 7   User Type             18448 non-null  object
 8   Birth Year            18448 non-null  int64 
 9   Age                   18448 non-null  int64 
 10  Age Groups            18448 non-null  object
 11  Trip Duration         18448 non-null  int64 
 12  Trip_Duration_in_min  18448 non-null  object
 13  Month                 18448 non-null  int64 
 14  Season                18448 non-null  object
 15  Temperature           18448 non-null  int64 
 16  Weekday               18448 non-null  object
 17  Trip Duration (min)   18448 non-null  int64 
dtypes: int64(9), object(9)
memory usage: 2.5+ MB
```

Question 1: What are the most popular pick-up locations across the city for NY Citi Bike rental?

In [11]:
```
# Conduct Analysis of Question 1
df1 = data_cleaned['Start Station Name'].value_counts()
df1_normalized = data_cleaned['Start Station Name'].value_counts(normaliz

# Display the result of analysis
print("Analysis:")
print(f"There are {data_cleaned['Start Station Name'].nunique()} starting
print()
print("For the first question, the top 15 starting stations and their res
print()
print(df1[:15])
print('----- Proportions -----')
# Quickly grasp what proportions of the top 15 starting stations are of t
print(df1_normalized[:15])
```

```
Analysis:
There are 50 starting stations in total; also see the 'descriptive statist
ics' above.

For the first question, the top 15 starting stations and their respective
proportions are as follows.

Start Station Name
Grove St PATH       2319
Exchange Place      1341
Hamilton Park       1185
Sip Ave             1184
Morris Canal         768
Newport PATH         707
City Hall            622
Van Vorst Park       580
Newark Ave           554
Warren St            540
Brunswick St         525
Dixon Mills          515
Jersey & 6th St      512
Jersey & 3rd         502
Marin Light Rail     500
Name: count, dtype: int64
----- Proportions -----
Start Station Name
Grove St PATH       0.125705
Exchange Place      0.072691
Hamilton Park       0.064235
Sip Ave             0.064180
Morris Canal        0.041631
Newport PATH        0.038324
City Hall           0.033716
Van Vorst Park      0.031440
Newark Ave          0.030030
Warren St           0.029271
Brunswick St        0.028458
Dixon Mills         0.027916
Jersey & 6th St     0.027754
Jersey & 3rd        0.027212
Marin Light Rail    0.027103
Name: proportion, dtype: float64
```

In [12]:
```
# Conduct Visualization for Question 1
viz1 = sns.barplot(data=df1[:15], orient="h", color='skyblue')
viz1.set_title("Question 1: Most Popular Pickup Locations")
plt.show()
```



Finding 4:
Although the de-duplicating techniques of Pandas and Google Sheets differ, the analysis results for Question 1 are the same (except for the third and fourth place rankings as well as exact numbers).

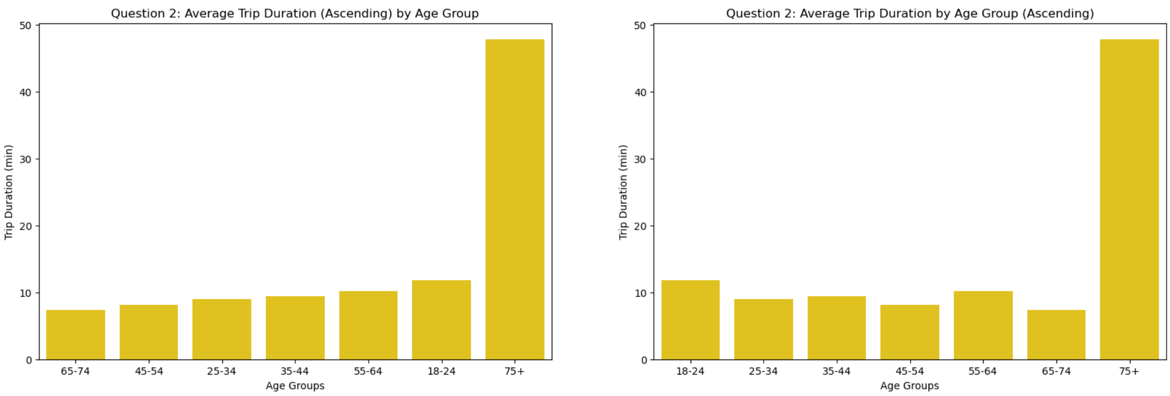
Question 2: How does the average trip duration vary across different age groups?

In [13]:
```
# Conduct Analysis of Question 2
df2 = data_cleaned.groupby(['Age Groups'])['Trip Duration (min)'].mean()
df2_sorted = data_cleaned.groupby(['Age Groups'])['Trip Duration (min)'].

# Display the result of analysis
print("Analysis:")
print("For the second question, the average trip duration by the age grou
print()
print(df2_sorted)
```

```
Analysis:
For the second question, the average trip duration by the age group is as
follows.

Age Groups
65-74     7.350801
45-54     8.085855
25-34     9.028642
35-44     9.476480
55-64    10.200000
18-24    11.857143
75+      47.877193
Name: Trip Duration (min), dtype: float64
```

In [14]:

```python
# Conduct Visualization for Question 2
fig, viz2 = plt.subplots(1, 2, figsize = (20,6))
viz2[0] = sns.barplot(data=df2_sorted, ax=viz2[0], color='Gold')
viz2[0].set_title("Question 2: Average Trip Duration (Ascending) by Age G
viz2[1] = sns.barplot(data=df2, ax=viz2[1], color='Gold')
viz2[1].set_title("Question 2: Average Trip Duration by Age Group (Ascend
plt.show()
```



Finding 5:
Although the de-duplicating techniques of Pandas and Google Sheets differ, the analysis results for Question 2 are the same (except for exact numbers).

Moreover, in order to quickly gain insights, an additional bar graph is generated to depict the main features when the age groups are arranged in ascending order.

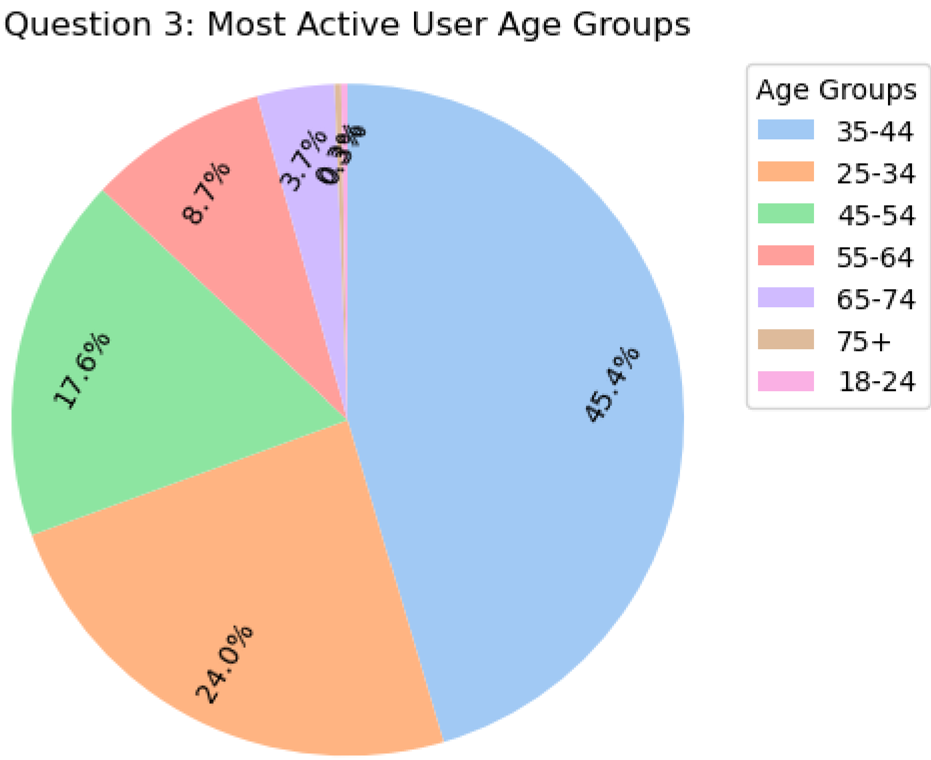
Question 3: Which age group rents the most bikes?

In [15]:

```python
# Conduct Analysis of Question 3
df3 = data_cleaned['Age Groups'].value_counts()

# Display the result of analysis
print("Analysis:")
print("For the third question, the most active user age groups are as fol
print()
print(df3)
```

```
Analysis:
For the third question, the most active user age groups are as follows.

Age Groups
35-44    8376
25-34    4434
45-54    3238
55-64    1600
65-74     687
75+        57
18-24      56
Name: count, dtype: int64
```

In [16]:
```
#### Conduct Visualization for Question 3
fig, viz3 = plt.subplots()
viz3_colors = sns.color_palette('pastel')
patches, texts, autotexts = viz3.pie(df3, colors=viz3_colors, autopct='%.
viz3.legend(patches, df3.index, title="Age Groups", loc="upper right", bb
#viz3.pie(df3, labels=df3.index, colors=viz3_colors, autopct='%.1f%%', co
viz3.set_title("Question 3: Most Active User Age Groups")
plt.setp(autotexts, size=10, rotation=60)
plt.axis('equal')
plt.show()
```



Finding 6:
Although the de-duplicating techniques of Pandas and Google Sheets differ, the analysis results for Question 3 are the same (except for the exact numbers).

Meanwhile, it is worth mentioning that Google Sheets makes it much easier to create a pie chart than the *matplotlib.pyplot* and *Seaborn* libraries, considering that *Seaborn* currently does not have a direct "pieplot" function (similar to *barplot*).

Question 4: How does bike rental vary across the two user groups (one-time users vs. long-term subscribers) on different days of the week?

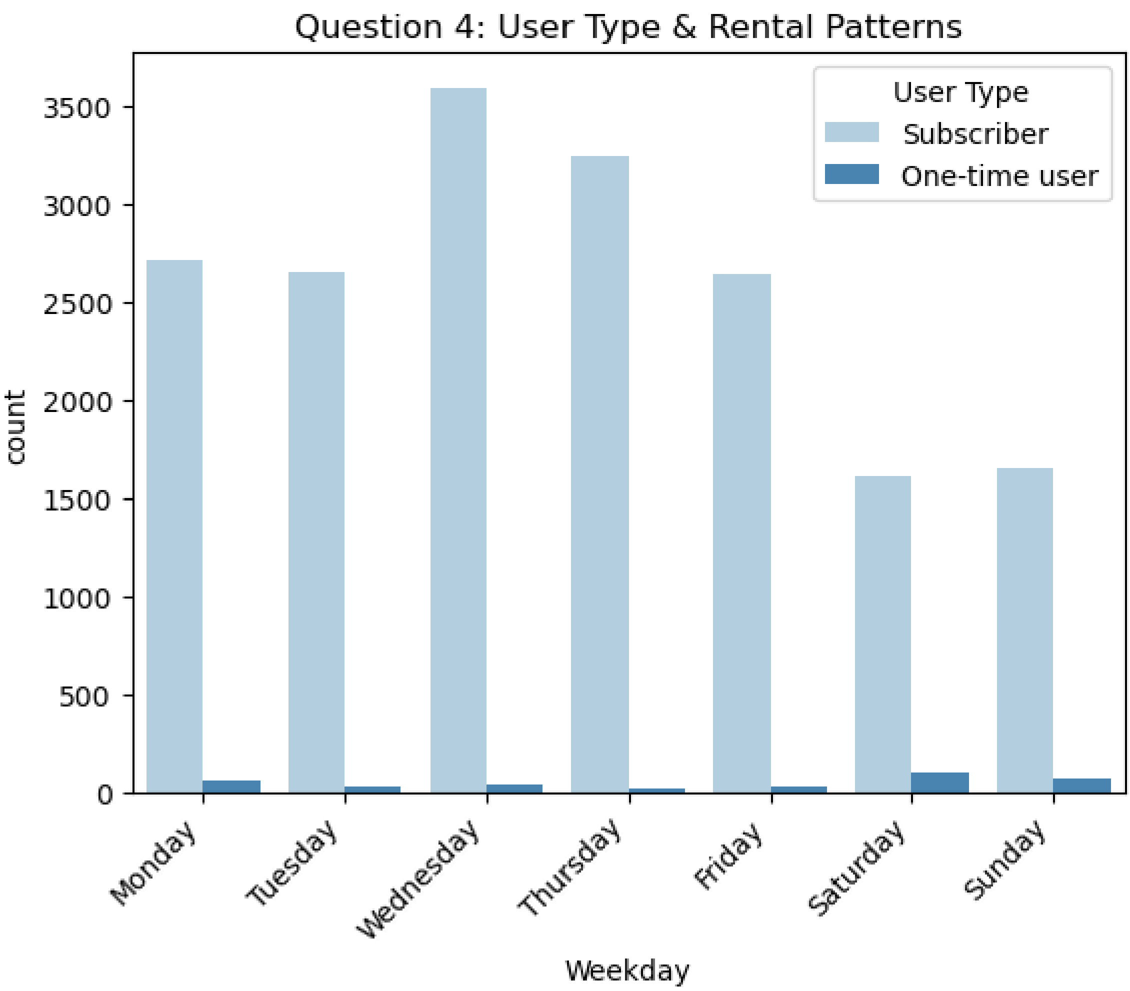
In [17]:

```
# Conduct Analysis of Question 4
df4 = data_cleaned.groupby(['Weekday', 'User Type']).size().reset_index(n
#df4['Count'].sum() # Check if the sum of the new "Count" variable is equ

# Define the desired order of weekdays
weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
df4['Weekday'] = pd.Categorical(df4['Weekday'], categories=weekday_order,
df4_sorted_weekday = df4.sort_values(by='Weekday').reset_index(drop=True)

# Display the result of analysis
print("Analysis:")
print("For the fourth question, the user types and their respective renta
print()
print(df4_sorted_weekday)
```

```
Analysis:
For the fourth question, the user types and their respective rental patter
ns are as follows.

      Weekday     User Type  Count
0      Monday  One-time user     57
1      Monday     Subscriber   2715
2     Tuesday  One-time user     30
3     Tuesday     Subscriber   2647
4   Wednesday  One-time user     43
5   Wednesday     Subscriber   3590
6    Thursday  One-time user     20
7    Thursday     Subscriber   3239
8      Friday  One-time user     29
9      Friday     Subscriber   2645
10   Saturday  One-time user    102
11   Saturday     Subscriber   1612
12     Sunday  One-time user     68
13     Sunday     Subscriber   1651
```

In [18]:

```
# Conduct Visualization for Question 4
# Seaborn's Countplot for two Categorical columns
df4_countplot = data_cleaned.loc[:,['Weekday', 'User Type']]
df4_countplot['Weekday'] = pd.Categorical(df4_countplot['Weekday'], categ
df4_countplot_sorted_weekday = df4_countplot.sort_values(by='Weekday').re
viz4 = sns.countplot(x='Weekday', hue='User Type', data=df4_countplot_sor
viz4.set_title("Question 4: User Type & Rental Patterns")
plt.xticks(rotation=45, ha='right')
plt.show()
```



Finding 7:
Although the de-duplicating techniques of Pandas and Google Sheets differ, the analysis results for Question 4 are the same (except for the exact numbers).

Similar to the stacked stepped area chart in Google Sheets, the *Seaborn* countplot has an excellent ability to capture the primary features of two categorical variables, but in a more efficient manner.

Furthermore, regarding two categorical variables, *One-hot Encoding*, *Seaborn Catplot* (with "kind='count'", making it similar to *Seaborn Countplot*) and *Proportions.plot* (which may take a long time to produce results) are also useful techniques, especially for complex datasets.

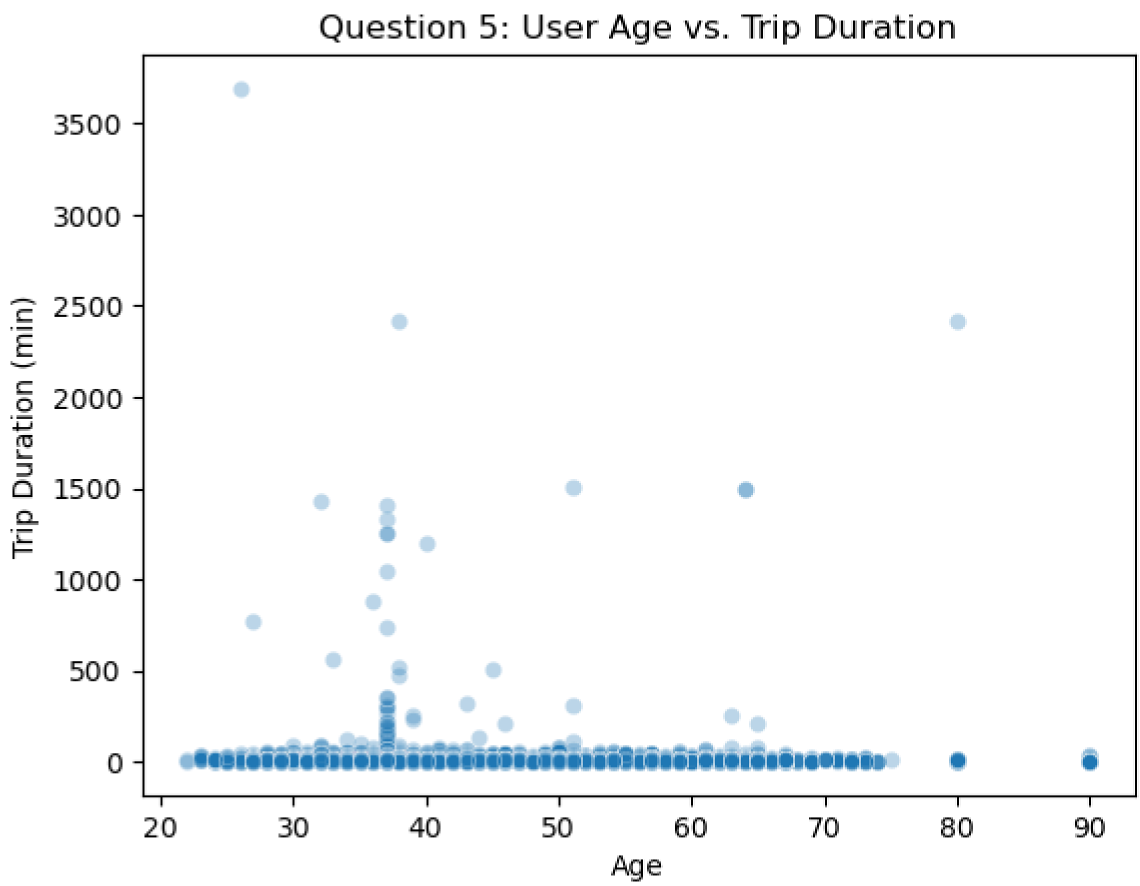
Question 5: Does user age impact the average bike trip duration?

In [19]:
```
# Conduct Analysis of Question 5
df5 = data_cleaned.loc[:,['Age', 'Trip Duration (min)']]

# Compute the Median of the 'Age' and 'Trip Duration (min)' variables
age_median = df5['Age'].median()
trip_median = df5['Trip Duration (min)'].median()
median = pd.DataFrame({'Age': [age_median], 'Trip Duration (min)': [trip_

df5_reorganized = pd.concat([df5.describe().loc[['mean', 'min', 'max'],:]

# Display the result of analysis
```

```
print("Analysis:")
print("Regarding the last question, the basic descriptive statistics of u
print()
print(df5_reorganized)
```

```
Analysis:
Regarding the last question, the basic descriptive statistics of user age
and trip duration are as follows.

              Age  Trip Duration (min)
mean    41.705713             9.234226
min     22.000000             1.000000
max     90.000000          3693.000000
median  39.000000             5.000000
```

In [20]:
```
# Conduct Visualization for Question 5
viz5 = sns.scatterplot(data=df5, x='Age', y='Trip Duration (min)', alpha=
viz5.set_title("Question 5: User Age vs. Trip Duration")
plt.show()
```



Finding 8:
Although the de-duplicating techniques of Pandas and Google Sheets differ, the analysis results for Question 5 are the same (except for the exact mean values).

Moreover, the *alpha* parameter in *Seaborn Scatterplot* helps visualize many overlapping data points, thus revealing data density and improving clarity. Therefore, the same treatment is applied in Google Sheets.

Conclusions:

This project achieves an end-to-end analytical journey, beginning with a comprehensive business analysis of a bike-sharing program and culminating in a demonstration of advanced data manipulation and visualization skills.

Initially, I leveraged Google Sheets to meticulously store, clean, and analyze a dataset of over 20,400 raw records. Through exploratory data analysis using pivot tables and various visualizations, I successfully uncovered key user behavior patterns and rental trends. This analysis yielded actionable, data-driven insights that could directly help optimize station supply, target core demographics, and enhance subscriber benefits to boost weekend rentals.

To validate these findings and demonstrate my expanded technical capabilities, I recreated the entire analysis using Python. This supplementary project involved robust data cleaning, including the identification and handling of outliers, duplicates, and missing values. Utilizing Python's powerful libraries, I generated a diverse range of visualizations — from scatter plots to bar charts — which confirmed the core business findings while showcasing a professional and reproducible analytical workflow.

Ultimately, this dual-methodology project demonstrates my ability to not only translate raw data into strategic business recommendations but also to proficiently apply industry-standard analytical tools, ensuring the accuracy and integrity of every insight delivered.