

Part I: Exploratory Data Analysis (EDA)

This document explores the individual trips recorded by the Ford GoBike System in February 2019, following the **Question-Visualization-Observations (Q-V-O)** framework and the requirements of the project rubric

1. Introduction and Setup

Dataset Overview

This dataset contains information about individual rides made in the Ford GoBike System covering the greater San Francisco Bay area during February 2019. Key variables include trip duration, station locations, user type, gender, and member birth year.

Initial Questions

1. What is the typical user profile (e.g., age, subscriber status) of the system?
2. How does user type (Subscriber vs. Customer) affect trip duration and time-of-day riding patterns?
3. Where are the highest volume travel routes?

2. Preliminary Wrangling

We load libraries, clean the data, and engineer key features like `duration_min` and `member_age`

```
In [2]:  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# set a consistent style for professional plots  
sns.set_style('darkgrid')  
%matplotlib inline  
  
# Load the dataset  
df = pd.read_csv('201902-fordgobike-tripdata.csv')  
df_original = df.copy()  
  
def wrangle_gobike_data(df):  
    """ Cleans data and engineers time, duration, and age features. """  
    # 1. convert time and calculate derived features  
    df['start_time'] = pd.to_datetime(df['start_time'])  
    df['end_time'] = pd.to_datetime(df['end_time'])  
    df['start_hour'] = df['start_time'].dt.hour  
    df['start_day'] = df['start_time'].dt.day_name()  
  
    # order days correctly  
    day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']  
    df['start_day'] = pd.Categorical(df['start_day'], categories=day_order)  
  
    # calculate duration in minutes  
    df['duration_min'] = df['duration'].apply(lambda x: x / 60)  
  
    # calculate member age  
    df['member_age'] = df['birth_year'].apply(lambda x: 2019 - x)  
  
    # drop columns not needed for analysis  
    df.drop(['start_time', 'end_time', 'birth_year'], axis=1, inplace=True)  
  
    return df
```

```

df['start_day'] = pd.Categorical(df['start_day'], categories=day_order, order=True)

# duration in minutes
df['duration_min'] = df['duration_sec'] / 60

# 2. member age (data is from 2019)
CURRENT_YEAR = 2019
df['member_age'] = CURRENT_YEAR - df['member_birth_year'].fillna(0).astype(int)

# 3. filtering and type conversion
# filter out extreme outliers (e.g., age > 80 or < 18)
df = df.query('member_age >= 18 and member_age <= 80')
df.dropna(subset=['member_gender'], inplace=True) # Drop missing gender for demographic analysis

for col in ['user_type', 'member_gender']:
    df[col] = df[col].astype('category')

print(f'data cleaned. final shape: {df.shape}')
return df

```

```
df = wrangle_gobike_data(df)
```

data cleaned. final shape: (174955, 20)

C:\Users\Darawsheh\AppData\Local\Temp\ipykernel_29012\846345066.py:36: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

df.dropna(subset=['member_gender'], inplace=True) # Drop missing gender for demographic analysis

C:\Users\Darawsheh\AppData\Local\Temp\ipykernel_29012\846345066.py:39: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

df[col] = df[col].astype('category')

C:\Users\Darawsheh\AppData\Local\Temp\ipykernel_29012\846345066.py:39: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

df[col] = df[col].astype('category')

3. Univariate Exploration

A. Distribution of Trip Duration (Histogram)

Question: What is the typical duration of a bike trip?

In [31]: plt.figure(figsize=[14, 6])

```
# Use Log scale for x-axis due to extreme right skew
bins = 10**np.arange(0, np.log10(df['duration_sec'].max()) + 0.1, 0.1)
```

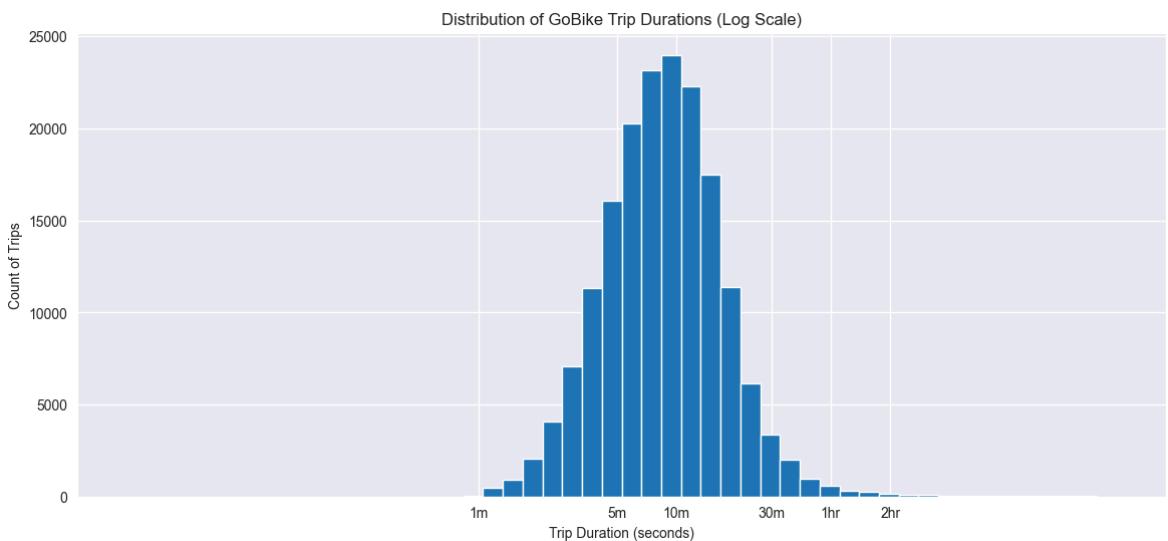
```

plt.hist(data=df, x='duration_sec', bins=bins)
plt.xscale('log')

# Set clear, interpretable tick labels
tick_locs = [60, 300, 600, 1800, 3600, 7200]
tick_labels = ['1m', '5m', '10m', '30m', '1hr', '2hr']
plt.xticks(tick_locs, tick_labels)

plt.title('Distribution of GoBike Trip Durations (Log Scale)')
plt.xlabel('Trip Duration (seconds)')
plt.ylabel('Count of Trips')
plt.show()

```



Observations: The distribution is unimodal and centered just under the 10-minute mark. When viewed on a log scale, the distribution is roughly normal, confirming that most trips are short commutes.

User Type Frequency (Count Plot)

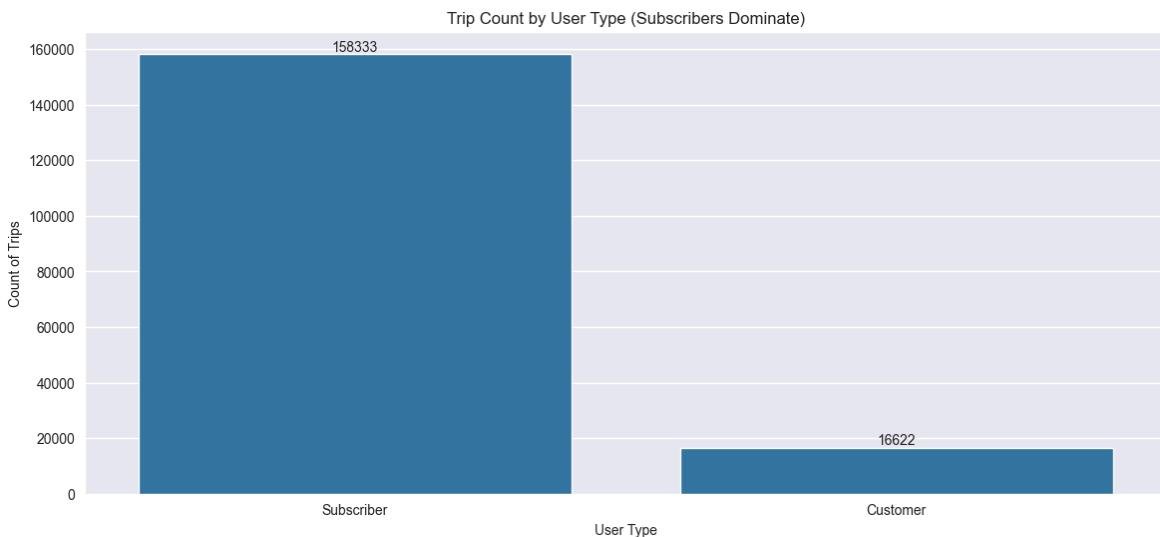
Question: Is the system used mostly by regular commuters or casual riders?

```

In [32]: plt.figure(figsize=[14, 6])
sns.countplot(data=df, x='user_type', color=sns.color_palette()[0], order=df['us
# Add annotations to show exact counts/proportions
for i, count in enumerate(df['user_type'].value_counts()):
    plt.text(i, count + 1000, str(count), ha='center')

plt.title('Trip Count by User Type (Subscribers Dominate)')
plt.xlabel('User Type')
plt.ylabel('Count of Trips')
plt.show()

```



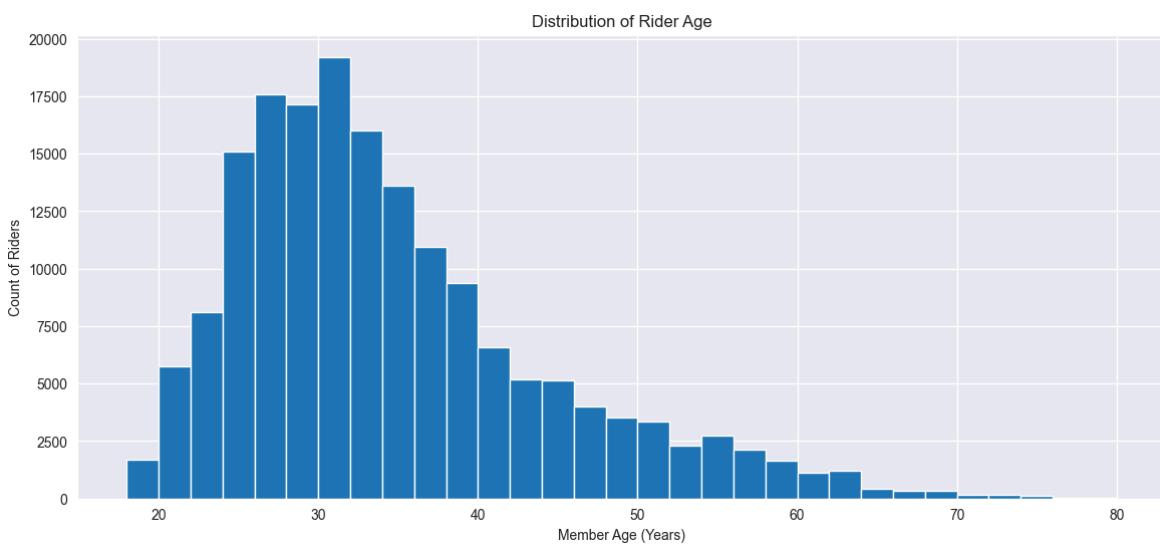
Observations: Subscribers (regular pass holders) make up the vast majority of trips (over 85%), indicating the system's primary function is supporting daily commuting.

C. Distribution of Member Age (Additional Plot: Histogram)

Question: What is the age profile of the riders?

```
In [33]: plt.figure(figsize=[14, 6])
# Using a fixed bin size for age
bins = np.arange(18, df['member_age'].max() + 2, 2)
plt.hist(data=df, x='member_age', bins=bins)

plt.title('Distribution of Rider Age')
plt.xlabel('Member Age (Years)')
plt.ylabel('Count of Riders')
plt.show()
```



Observations: The age distribution is roughly bell-shaped, peaking around 30 years old and heavily skewed towards younger to middle-aged adults, typical of a professional urban commuting service.

4. Bivariate Exploration

A. Age vs. Duration (Scatter Plot)

Question: Is there a relationship between a rider's age and how long their trip is?

```
In [34]: # Bivariate Plot: Age vs. Duration (Stripplot with Jitter)
# Question: What is the relationship between rider age and trip duration?

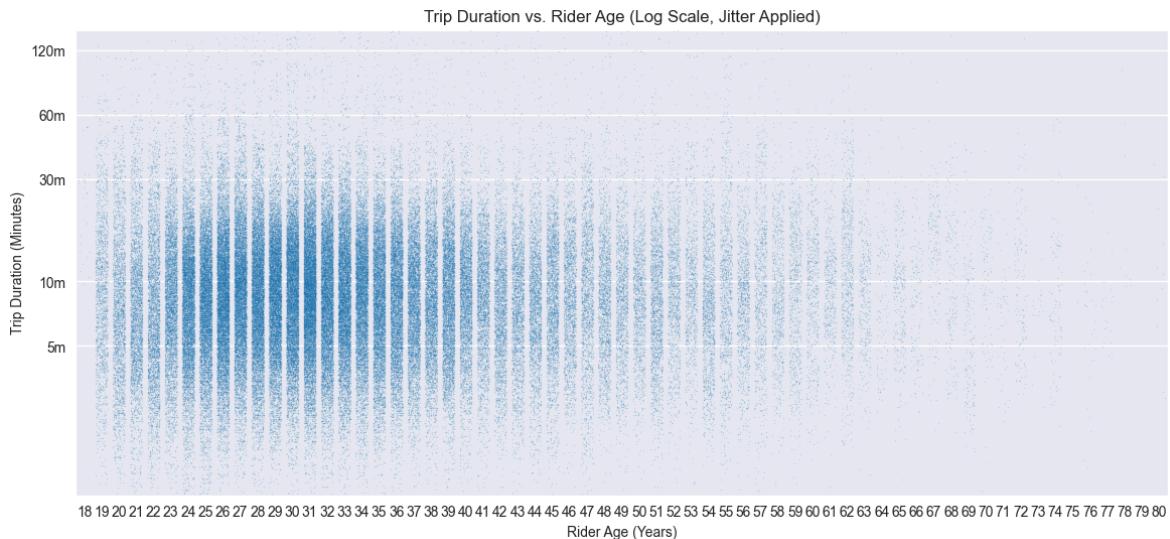
# Set fixed plot dimension (A good practice from the tips)
plt.figure(figsize=[14, 6])

# Use stripplot with jitter for the full dataset. 'y' axis is log-scaled.
sns.stripplot(data=df, x='member_age', y='duration_min',
               color=sns.color_palette()[0],
               jitter=0.35, # Apply jitter to prevent overplotting discrete x-val
               size=1, # Keep point size small
               alpha=0.2, # Low alpha to hint at density of the massive dataset
               order=np.arange(18, 81))

plt.yscale('log') # Log scale on axis only

# Set clear, interpretable tick Labels in MINUTES
tick_locs = [5, 10, 30, 60, 120]
plt.yticks(tick_locs, [f'{i}m' for i in tick_locs])
plt.ylim(1, 150) # Exclude extreme outliers for better visualization

plt.title('Trip Duration vs. Rider Age (Log Scale, Jitter Applied)')
plt.xlabel('Rider Age (Years)')
plt.ylabel('Trip Duration (Minutes)')
plt.show()
```



Observations: There is **no clear linear relationship** between age and duration. Most trips, regardless of age, are clustered at the bottom (under 10 minutes).

B. Duration by User Type (Box Plot)

Question: Does user type affect trip length?

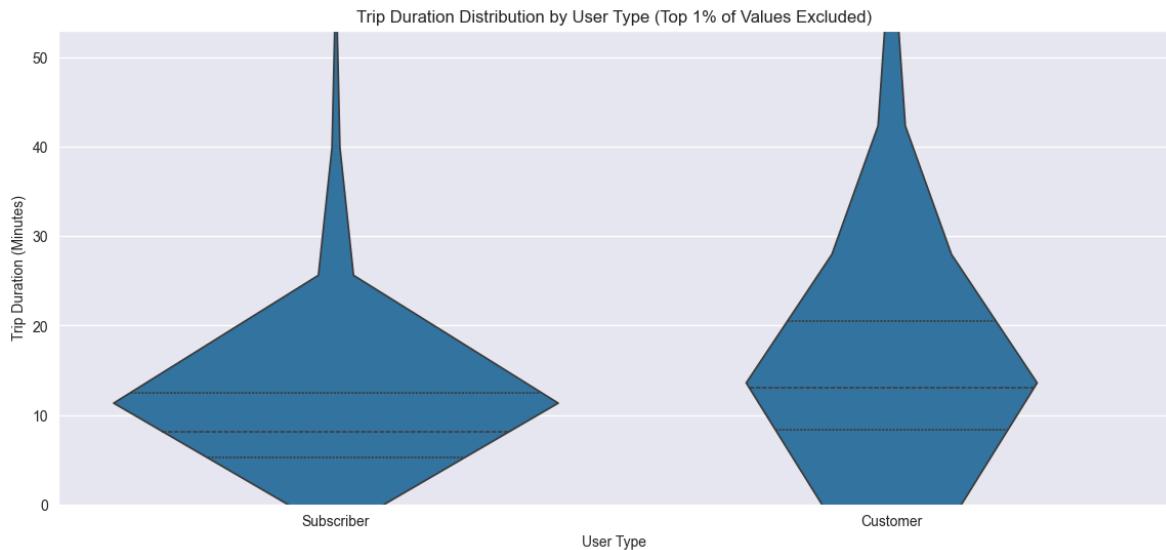
```
In [35]: # Additional Bivariate Plot: Duration by User Type (Outliers Excluded)
# Find the 99th percentile of duration_min to set the Y limit
quantile_99 = df['duration_min'].quantile(0.99)

# Set fixed plot dimension
plt.figure(figsize=[14, 6])

sns.violinplot(data=df, x='user_type', y='duration_min', inner='quartile',
                color=sns.color_palette()[0], order=['Subscriber', 'Customer'])

# Set the Y-axis limit to the 99th percentile to exclude extreme outliers
plt.ylim(0, quantile_99)

plt.title(f'Trip Duration Distribution by User Type (Top 1% of Values Excluded)')
plt.xlabel('User Type')
plt.ylabel('Trip Duration (Minutes)')
plt.show()
```



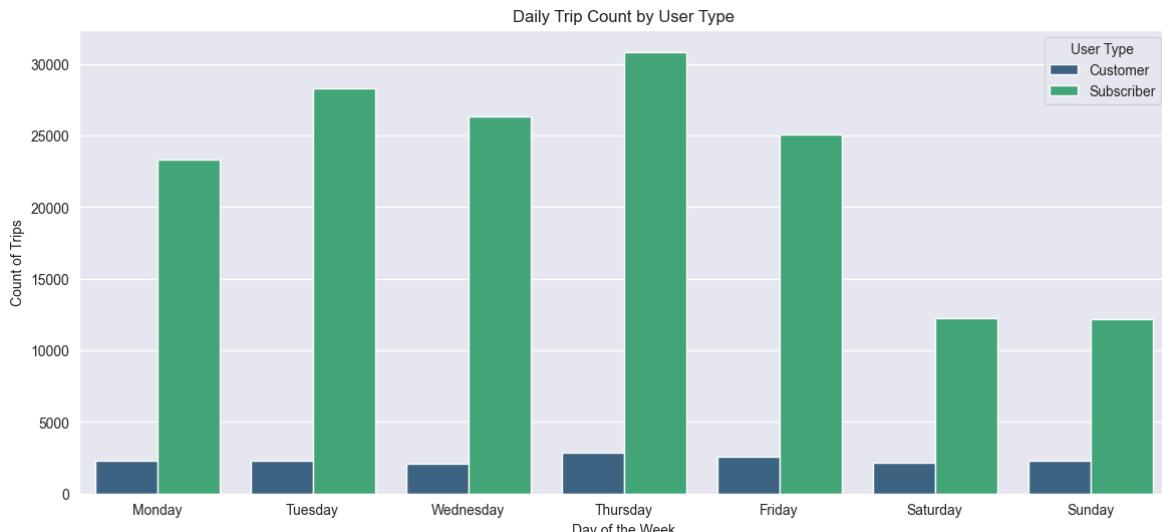
Observations: The **Customer** group has a **higher median duration** and a much broader distribution than the **Subscriber** group. This strongly suggests that Customers use the bikes for longer, less frequent recreational trips.

C. Daily Usage Patterns (Clustered Count Plot)

Question: How does daily usage differ between Subscribers and Customers?

```
In [36]: plt.figure(figsize=[14, 6])
sns.countplot(data=df, x='start_day', hue='user_type', palette='viridis')

plt.title('Daily Trip Count by User Type')
plt.xlabel('Day of the Week')
plt.ylabel('Count of Trips')
plt.legend(title='User Type')
plt.show()
```



Observations: Subscriber use drops significantly on weekends, confirming their reliance on the system for **weekday commuting**. Customer use remains relatively flat or slightly increases on weekends, confirming their non-commute, recreational use.

D. Top Routes (Heatmap)

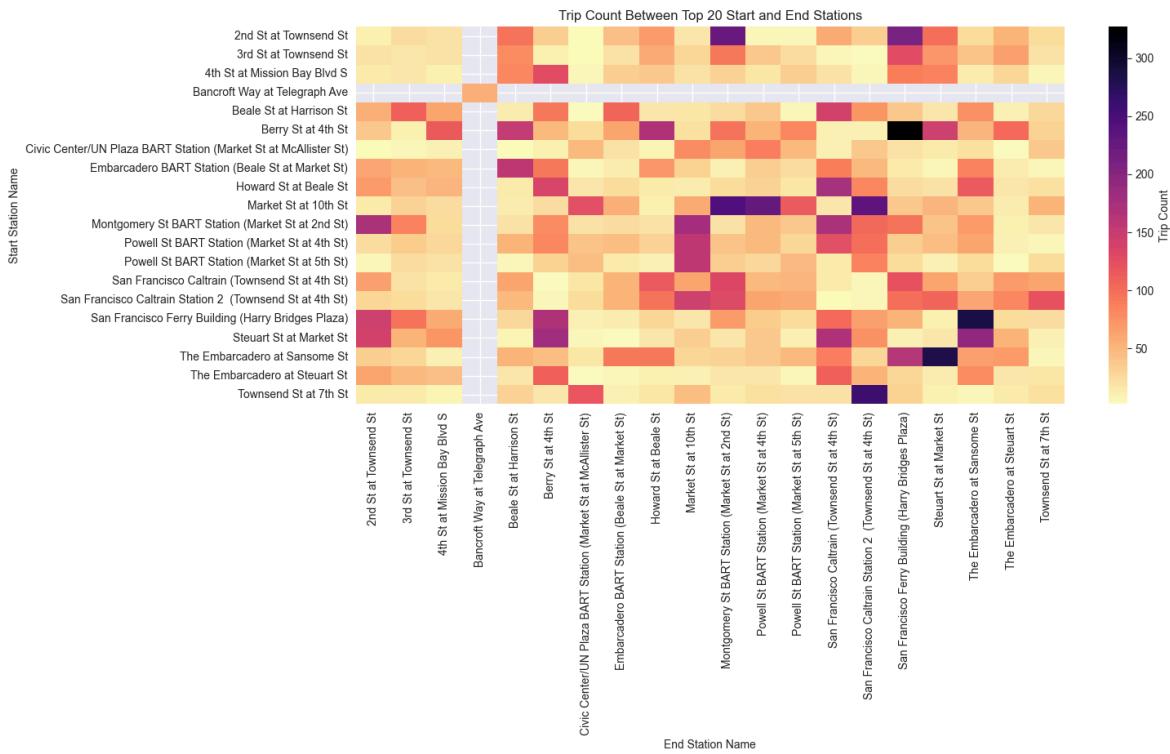
Question: What are the most frequent travel routes (start station to end station)?

```
In [37]: # 1. Identify the top 20 most frequent stations (start and end combined)
top_stations = pd.concat([df['start_station_name'], df['end_station_name']]).value_counts().nlargest(20).index

# 2. Filter the data to include only trips between these top stations
df_top = df[(df['start_station_name'].isin(top_stations)) & (df['end_station_name'].isin(top_stations))]

# 3. Create a contingency table (pivot table) for the heatmap counts
ct_counts = df_top.groupby(['start_station_name', 'end_station_name']).size()
ct_counts = ct_counts.reset_index(name='count')
ct_counts = ct_counts.pivot(index='start_station_name', columns='end_station_name', values='count')

plt.figure(figsize=[14, 6])
# Plot the heatmap
sns.heatmap(ct_counts, annot=False, fmt='d', cmap='magma_r', cbar_kws={'label': 'Trip Count Between Top 20 Start and End Stations'})
plt.title('Trip Count Between Top 20 Start and End Stations')
plt.xlabel('End Station Name')
plt.ylabel('Start Station Name')
plt.show()
```



Observations: This plot clearly identifies a few **very high-volume corridors** (darker squares) between major transit hubs and downtown employment centers. It also shows a strong diagonal pattern (trips starting and ending at the same station), indicating short local loops or returns.

5. Multivariate Exploration

Rush Hour Profile (Facet Plot)

Question: How do the rush-hour patterns look when gender is factored in?

```
In [38]: # Use a temporary DataFrame to avoid re-plotting all 175k points in the notebook
df_plot = df[['start_hour', 'user_type', 'member_gender']].copy()

# 1. Initialize the FacetGrid, specifying the data source and the column for facets
g = sns.FacetGrid(data=df_plot, col='user_type', height=5, aspect=1.5, col_wrap=2)

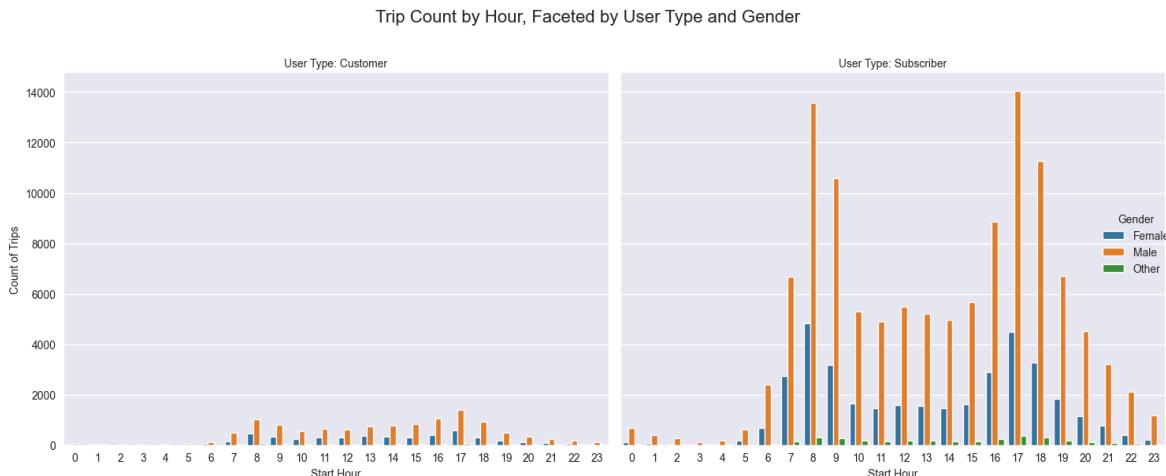
# After defining the FacetGrid (g)
g.fig.set_size_inches(14, 6) # Adjust height as needed, but maintain width for consistency

# 2. Use map_dataframe() to apply sns.countplot, passing all required arguments.
# Note: The 'order' for start_hour should be passed directly to countplot.
g.map_dataframe(sns.countplot, 'start_hour', hue='member_gender',
                palette='tab10',
                order=np.arange(0, 24))

# 3. Finalize plot formatting
g.set_axis_labels('Start Hour', 'Count of Trips')
g.set_titles('User Type: {col_name}')
g.add_legend(title='Gender')

plt.suptitle('Trip Count by Hour, Faceted by User Type and Gender', y=1.02, fontweight='bold')
```

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



Observations:

- **Subscribers** show clear **bimodal rush-hour peaks (8 AM & 5 PM)**, consistent across genders.
- **Customers** show a much flatter distribution with a midday concentration, reinforcing their recreational pattern.
- **Male** riders account for the overwhelming majority of trips across all hours and user types.

B. Age vs. Duration Explained by User Type (Multiple Encodings)

Question: Does the `user_type` variable explain the randomness observed in the initial age vs. duration scatter plot?

```
In [25]: # Multivariate Plot: Age vs. Duration (Stripplot with Jitter and Hue)
# Question: Does the User Type explain the relationship between age and duration

# Set fixed plot dimension
plt.figure(figsize=[14, 6])

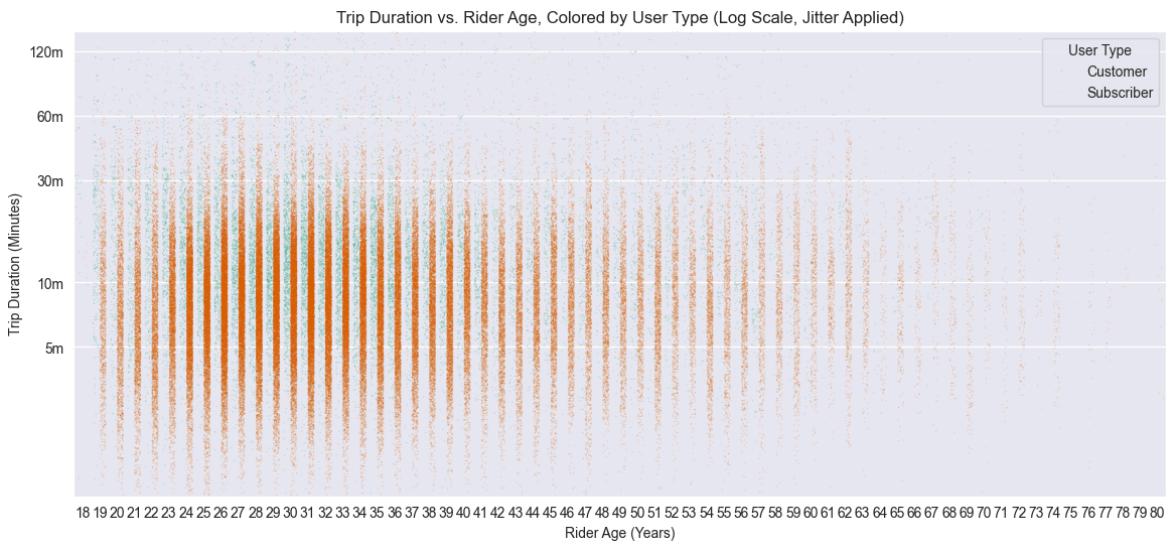
# Use stripplot with jitter and hue
sns.stripplot(data=df, x='member_age', y='duration_min', hue='user_type',
               jitter=0.35, # Apply jitter
               dodge=True, # Automatically separate categories
               size=1, alpha=0.2, # Keep points small and transparent
               palette='Dark2',
               order=np.arange(18, 81))

plt.yscale('log') # Log scale on axis only

# Set clear, interpretable tick Labels in MINUTES
tick_locs = [5, 10, 30, 60, 120]
plt.yticks(tick_locs, [f'{i}m' for i in tick_locs])
plt.ylim(1, 150) # Exclude extreme outliers for better visualization

plt.title('Trip Duration vs. Rider Age, Colored by User Type (Log Scale, Jitter')
```

```
plt.xlabel('Rider Age (Years)')
plt.ylabel('Trip Duration (Minutes)')
plt.legend(title='User Type')
plt.show()
```



Observations: The original scattered plot is now clearly separated. The vast majority of **Subscriber** (green) points are clustered tightly at the bottom (short trips). The **Customer** (purple) points are the ones scattered across the higher duration values. This confirms that **User Type is the primary factor** determining trip duration, not age.

Conclusion Summary

The exploration revealed a **dual-purpose system** defined by user type:

- **Subscribers (Commuters):** Dominate usage, taking short trips concentrated in clear **weekday rush-hour peaks** (8 AM / 5 PM).
- **Customers (Recreational):** Take significantly **longer trips** with flatter usage across the day, especially on weekends.

The variable `user_type` is the most critical factor explaining trip duration differences, confirmed after using logarithmic transformation to handle data skewness.