Data Exploration – Ford GoBike Tripdata

Gabriela Trindade

Table of Contents

1. Introduction

This project was written for Udacity's Data Analyst nanodegree program. The primary goal is to demonstrate the importance and value of data visualization techniques in the data analysis process. For this purpose, the Ford GoBike dataset was chosen and wrangled. This dataset contains basic information about individual rides made in this system.

Ford GoBike is a regional public bicycle sharing system in San Francisco Bay Area. The system was originally launched as Bay Area Bike Share in August 2013, but it was re-launched in 2017 after the sponsorship of Ford. Like other bike share systems, it consists of a fleet of bikes that is locked into a network of docking stations throughout the city. The bikes can be unlocked in one station and returned to any other station in the system, making them ideal for one-way trips. After becoming a member or purchasing a pass, riders will have access to all bikes in the network.

In [35]:
```
#imports required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from datetime import date

%matplotlib inline
%config InlineBackend.figure_format = 'svg'
```

2. Gathering Data

In [36]:
```
#loads the ".csv" file into a dataframe
df = pd.read_csv('201902-fordgobike-tripdata.csv')
```

3. Assessing Data

In [37]:
```
#displays two decimal places in python pandas
pd.set_option('display.float_format', '{:.2f}'.format)
#sets the maximum number of rows and columns to display to unlimited
pd.set_option("display.max_rows", None, "display.max_columns", None)
```

In [38]:
```
#returns the first n rows of the dataframe
df.head()
```

Out[38]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitud
0	52185	2019-02-28 17:32:10.1450	2019-03-01 08:01:55.9750	21.00	Montgomery St BART Station (Market St at 2nd St)	37.7
1	42521	2019-02-28 18:53:21.7890	2019-03-01 06:42:03.0560	23.00	The Embarcadero at Steuart St	37.7
2	61854	2019-02-28 12:13:13.2180	2019-03-01 05:24:08.1460	86.00	Market St at Dolores St	37.7
3	36490	2019-02-28 17:54:26.0100	2019-03-01 04:02:36.8420	375.00	Grove St at Masonic Ave	37.7
4	1585	2019-02-28 23:54:18.5490	2019-03-01 00:20:44.0740	7.00	Frank H Ogawa Plaza	37.8

In [39]:
```
#returns the last n rows of the dataframe
df.tail()
```

Out[39]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_l
183407	480	2019-02-01 00:04:49.7240	2019-02-01 00:12:50.0340	27.00	Beale St at Harrison St	
183408	313	2019-02-01 00:05:34.7440	2019-02-01 00:10:48.5020	21.00	Montgomery St BART Station (Market St at 2nd St)	
183409	141	2019-02-01 00:06:05.5490	2019-02-01 00:08:27.2200	278.00	The Alameda at Bush St	
183410	139	2019-02-01 00:05:34.3600	2019-02-01 00:07:54.2870	220.00	San Pablo Ave at MLK Jr Way	
183411	271	2019-02-01 00:00:20.6360	2019-02-01 00:04:52.0580	24.00	Spear St at Folsom St	

In [40]:
```
#gets a concise summary of the dataframe
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 16 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   duration_sec             183412 non-null  int64  
 1   start_time               183412 non-null  object 
 2   end_time                 183412 non-null  object 
 3   start_station_id         183215 non-null  float64
 4   start_station_name       183215 non-null  object 
 5   start_station_latitude   183412 non-null  float64
 6   start_station_longitude  183412 non-null  float64
 7   end_station_id           183215 non-null  float64
 8   end_station_name         183215 non-null  object 
 9   end_station_latitude     183412 non-null  float64
 10  end_station_longitude    183412 non-null  float64
 11  bike_id                  183412 non-null  int64  
 12  user_type                183412 non-null  object 
 13  member_birth_year        175147 non-null  float64
 14  member_gender            175147 non-null  object 
 15  bike_share_for_all_trip  183412 non-null  object 
dtypes: float64(7), int64(2), object(7)
memory usage: 22.4+ MB
```

In [41]:
```
#looks for NA values in the dataframe
df.isnull().sum()
```

Out[41]:
```
duration_sec                 0
start_time                   0
end_time                     0
start_station_id           197
start_station_name         197
start_station_latitude       0
start_station_longitude      0
end_station_id             197
end_station_name           197
end_station_latitude         0
end_station_longitude        0
bike_id                      0
user_type                    0
member_birth_year         8265
member_gender             8265
bike_share_for_all_trip      0
dtype: int64
```

In [42]:
```
#checks for duplicate  values
df.duplicated().sum()
```

Out[42]:
```
0
```

In [43]:
```
#shows basic statistical details
df.describe()
```

Out[43]:

	duration_sec	start_station_id	start_station_latitude	start_station_longitude	end_station_id	e
count	183412.00	183215.00	183412.00	183412.00	183215.00	
mean	726.08	138.59	37.77	-122.35	136.25	
std	1794.39	111.78	0.10	0.12	111.52	
min	61.00	3.00	37.32	-122.45	3.00	
25%	325.00	47.00	37.77	-122.41	44.00	
50%	514.00	104.00	37.78	-122.40	100.00	

	duration_sec	start_station_id	start_station_latitude	start_station_longitude	end_station_id	e
75%	796.00	239.00	37.80	-122.29	235.00	
max	85444.00	398.00	37.88	-121.87	398.00	

4. Issues Summary

i) "Start_time" and "end_time" columns should be converted to datetime64 datatype;

ii) "Bike_id", "start_station_id" and " end_station_id" columns should be converted to object datatype;

iii) "Start_station_id", "start_station_name", "end_station_id", "end_station_name", "member_birth_year" and "member_gender" columns have null values;

iv) "Member_birth_year" should be converted to int64 datatype;

v) It would be nice to calculate the age of the users and converte this column to int64 datatype;

vi) It would be interesting to extract month, day, hour and weekday from "start_time" and "end_time";

vii) It would be nice to convert the duration in seconds to minutes for proper analysis.

5. Cleaning Data

In [44]:
```
#creates a copy of the original dataset
df_cleaned = df.copy(deep=True)
```

i) Define

"Start_time" and "end_time" columns should be converted to datetime64 datatype.

i) Code

In [45]:
```
#converts columns to datetime datatype
df_cleaned['start_time'] = pd.to_datetime(df_cleaned['start_time'])
```

In [46]:
```
df_cleaned['end_time'] = pd.to_datetime(df_cleaned['end_time'])
```

i) Test

In [47]:
```
#checks the datatypes
df_cleaned['start_time'].head(0)
```

Out[47]:
```
Series([], Name: start_time, dtype: datetime64[ns])
```

In [48]:
```
df_cleaned['end_time'].head(0)
```

Out[48]:
```
Series([], Name: end_time, dtype: datetime64[ns])
```

ii) Define

"Bike_id", "start_station_id" and " end_station_id" columns should be converted to object datatype.

ii) Code

In [49]:
```
#converts columns to object datatype
df_cleaned['bike_id'] = df_cleaned['bike_id'].astype('object')
```

In [50]:
```
df_cleaned['start_station_id'] = df_cleaned['start_station_id'].astype('object')
```

In [51]:
```
df_cleaned['end_station_id'] = df_cleaned['end_station_id'].astype('object')
```

ii) Test

In [52]:
```
#checks the datatypes
df_cleaned['bike_id'].head(0)
```

Out[52]:
```
Series([], Name: bike_id, dtype: object)
```

In [53]:
```
df_cleaned['start_station_id'].head(0)
```

Out[53]:
```
Series([], Name: start_station_id, dtype: object)
```

In [54]:
```
df_cleaned['end_station_id'].head(0)
```

Out[54]:
```
Series([], Name: end_station_id, dtype: object)
```

iii) Define

"Start_station_id", "start_station_name", "end_station_id", "end_station_name", "member_birth_year" and "member_gender" columns have null values.

> **Note:**
>
> "Member_birth_year" and "member_gender" columns have the same number of missing values (8,265), which is the highest total in the dataset. Since this represents only 4.5% of the total values in these columns, and even less in the others, it was decided to drop all the missing values.

iii) Code

In [55]:
```
df_cleaned = df_cleaned.dropna()
```

iii) Test

In [56]:
```
#Looks for NA values in the dataframe
df_cleaned.isnull().sum()
```

Out[56]:
```
duration_sec               0
start_time                 0
end_time                   0
start_station_id           0
start_station_name         0
start_station_latitude     0
start_station_longitude    0
end_station_id             0
end_station_name           0
end_station_latitude       0
end_station_longitude      0
bike_id                    0
```

```
user_type                0
member_birth_year        0
member_gender            0
bike_share_for_all_trip  0
dtype: int64
```

iv) Define

"Member_birth_year" should be converted to int64 datatype.

iv) Code

In [57]:
```
#converts columns to int64 datatype
df_cleaned['member_birth_year'] = df_cleaned['member_birth_year'].astype(np.int64)
```

iv) Test

In [58]:
```
#checks the datatype
df_cleaned['member_birth_year'].head(0)
```

Out[58]:
```
Series([], Name: member_birth_year, dtype: int64)
```

v) Define

It would be nice to calculate the age of the users.

v) Code

In [59]:
```
#gets today's date
today = date.today()
#subtracts the birth year from the current year and inserts into the dataset the "me
df_cleaned.insert(14, 'member_age', (today.year - df_cleaned['member_birth_year']))
```

In [60]:
```
#converts columns to int64 datatype
df_cleaned['member_age'] = df_cleaned['member_age'].astype(np.int64)
```

v) Test

In [61]:
```
#returns the first n rows of the dataframe
df_cleaned.head()
```

Out[61]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude
0	52185	2019-02-28 17:32:10.145	2019-03-01 08:01:55.975	21.00	Montgomery St BART Station (Market St at 2nd St)	37.79
2	61854	2019-02-28 12:13:13.218	2019-03-01 05:24:08.146	86.00	Market St at Dolores St	37.77
3	36490	2019-02-28 17:54:26.010	2019-03-01 04:02:36.842	375.00	Grove St at Masonic Ave	37.77
4	1585	2019-02-28 23:54:18.549	2019-03-01 00:20:44.074	7.00	Frank H Ogawa Plaza	37.80
5	1793	2019-02-28 23:49:58.632	2019-03-01 00:19:51.760	93.00	4th St at Mission Bay Blvd S	37.77

In [62]:
```
#checks changes
df_cleaned['member_age'].head(0)
```

Out[62]:
```
Series([], Name: member_age, dtype: int64)
```

vi) Define

It would be interesting to extract month, day, hour and weekday from "start_time" and "end_time".

vi) Code

In [63]:
```
#inserts into the dataset the following columns (related to month, day and hour) at
df_cleaned.insert(3, 'start_month', (df_cleaned.start_time.dt.strftime('%b')))
df_cleaned.insert(4, 'end_month', (df_cleaned.end_time.dt.strftime('%b')))
df_cleaned.insert(5, 'start_day', (df_cleaned.start_time.dt.strftime('%a')))
df_cleaned.insert(6, 'end_day', (df_cleaned.end_time.dt.strftime('%a')))
df_cleaned.insert(7, 'start_hour', (df_cleaned.start_time.dt.hour))
df_cleaned.insert(8, 'end_hour', (df_cleaned.end_time.dt.hour))
```

In [64]:
```
#change both to category type
df_cleaned['start_month'] = df_cleaned['start_month'].astype('object')
df_cleaned['end_month'] = df_cleaned['end_month'].astype('object')
df_cleaned['start_day'] = df_cleaned['start_day'].astype('object')
df_cleaned['end_day'] = df_cleaned['end_day'].astype('object')
df_cleaned['start_hour'] = df_cleaned['start_hour'].astype('object')
df_cleaned['end_hour'] = df_cleaned['end_hour'].astype('object')
```

vi) Test

In [65]:
```
#returns the first n rows of the dataframe
df_cleaned.head(1)
```

Out[65]:

	duration_sec	start_time	end_time	start_month	end_month	start_day	end_day	start_hour
0	52185	2019-02-28 17:32:10.145	2019-03-01 08:01:55.975	Feb	Mar	Thu	Fri	17

vii) Define

It would be nice to convert the duration in seconds to minutes for proper analysis.

vii) Code

In [66]:
```
#inserts into the dataset the "duration_min" column at a specific column index
df_cleaned.insert(1, 'duration_min', (df_cleaned.duration_sec/60))
```

vii) Test

In [67]:
```
#returns the first n rows of the dataframe
df_cleaned.head()
```

Out[67]:

	duration_sec	duration_min	start_time	end_time	start_month	end_month	start_day	end_d
0	52185	869.75	2019-02-28 17:32:10.145	2019-03-01 08:01:55.975	Feb	Mar	Thu	

	duration_sec	duration_min	start_time	end_time	start_month	end_month	start_day	end_d
2	61854	1030.90	2019-02-28 12:13:13.218	2019-03-01 05:24:08.146	Feb	Mar	Thu	
3	36490	608.17	2019-02-28 17:54:26.010	2019-03-01 04:02:36.842	Feb	Mar	Thu	
4	1585	26.42	2019-02-28 23:54:18.549	2019-03-01 00:20:44.074	Feb	Mar	Thu	
5	1793	29.88	2019-02-28 23:49:58.632	2019-03-01 00:19:51.760	Feb	Mar	Thu	

5.1. Data storage

In [68]:
```
#writes dataframe to ".csv" file
df_cleaned.to_csv('df_cleaned.csv', index=False)
```

In [69]:
```
#loads ".csv" file into a dataframe
df_cleaned = pd.read_csv('df_cleaned.csv')
```

6. Guiding Questions

What is the structure of the dataset?

In [70]:
```
#gets the current shape of the dataset
df_cleaned.shape
```

Out[70]: (174952, 24)

> After the dataset was cleaned, there are 174,952 rows and 24 columns. In summary, this dataset contains variables about:
>
> - trip: start/end date (month, day and hour) and its duration in seconds and minutes;
> - stations: start/end stations, their names and geolocations (latitude/longitude);
> - anonymized customer data: gender, birth date, age and user type.

What is/are the main feature(s) of interest in the dataset?

> I'm most interested in figuring out how genders ("member_gender") differ from each other in the use of this bike service, in terms of the users' age, the trip duration, and the day and hour that this trip starts.

What features in the dataset do you think will help support your investigation into your feature(s) of interest?

> The following features in the dataset will help to support this investigation: "duration_min", "member_age", "start_day" and "start_hour".

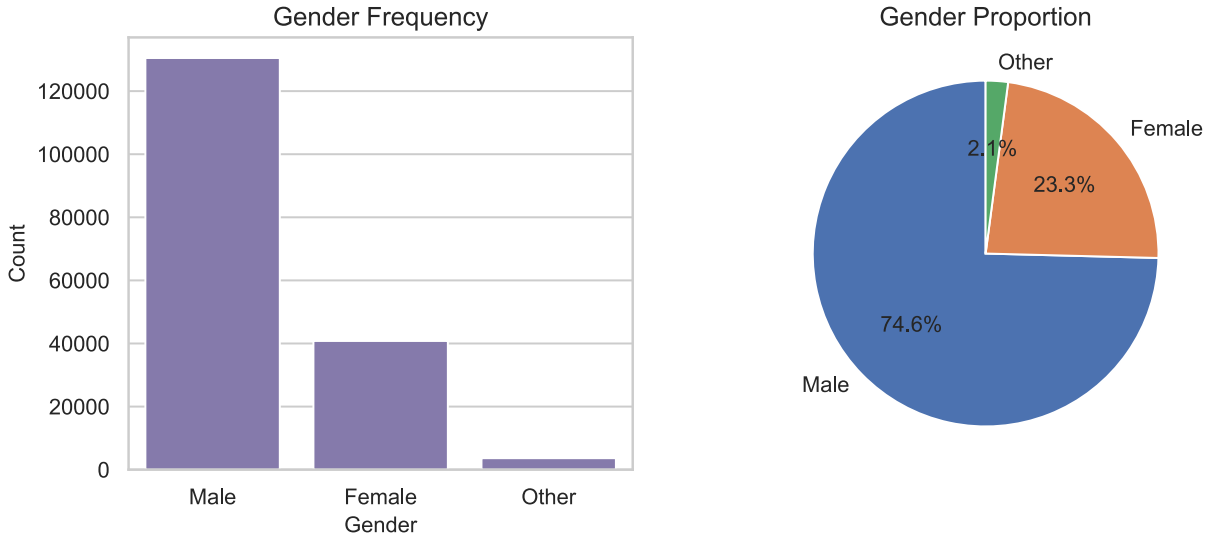
7. Exploring and Visualizing Data

7.1.1. Univariate Exploration

- This exploration will start by checking the frequency and proportion of each gender.

In [71]:
```
##sets the figure size
plt.figure(figsize = [10, 4])
#sets 1 row, 2 cols, subplot 1
plt.subplot(1, 2, 1)
#chooses the tuple of RGB colors
base_color = sb.color_palette()[4]
#counts the frequency of each unique value
gender_count = df_cleaned['member_gender'].value_counts()
#gets the indexes of the series
gender_order = gender_count.index
#plots the bar chart
sb.countplot(data = df_cleaned, x = 'member_gender', color=base_color, order = gende
#sets title and labels
plt.title('Gender Frequency', fontsize = 13)
plt.xlabel('Gender', fontsize = 11)
plt.ylabel('Count',  fontsize = 11);

#sets 1 row, 2 cols, subplot 2
plt.subplot(1, 2, 2)
#creates the pie chart
gender_counts = df_cleaned['member_gender'].value_counts()
gender_counts.plot.pie(autopct='%1.1f%%', textprops={'fontsize': 11}, startangle= 90
#sets title and labels
plt.title('Gender Proportion', fontsize=13)
plt.ylabel('')
plt.show();
```



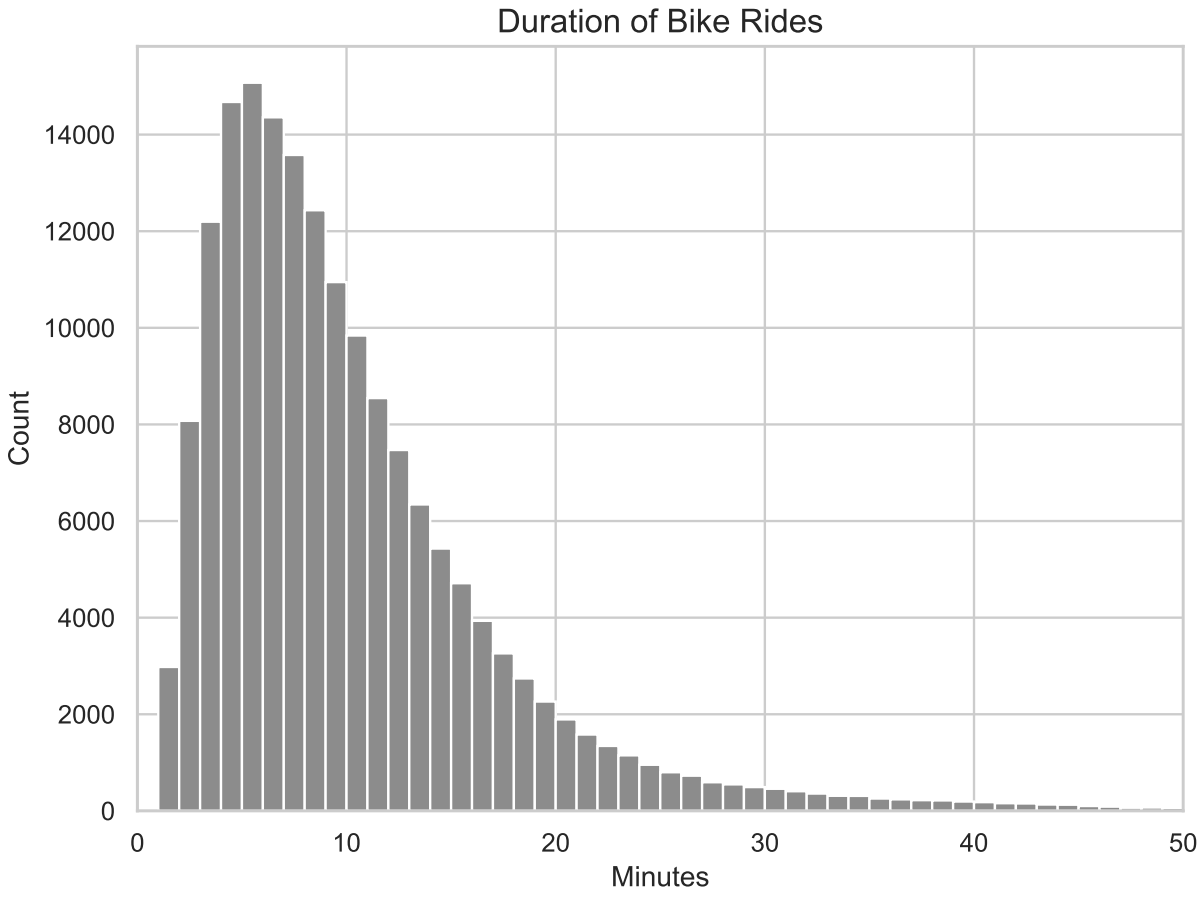
- From these vizualizations, it is possible to notice that the majority of users are males (74.6%). Following that, a histogram of the duration of bike rides will be created.

In [72]:
```
df_cleaned['duration_min'].describe()
```

Out[72]:
```
count   174952.00
mean        11.73
std         27.37
min          1.02
```

```
25%          5.38
50%          8.50
75%         13.15
max       1409.13
Name: duration_min, dtype: float64
```

In [73]:
```
#sets the figure size
plt.figure(figsize = [8, 6])
#chooses the tuple of RGB colors
base_color = sb.color_palette()[7]
#creates bins
bins = np.arange(0, df_cleaned['duration_min'].max()+1, 1)
#plots the histogram
plt.hist(data = df_cleaned, x = 'duration_min', bins = bins, color = base_color)
#sets title and labels
plt.title('Duration of Bike Rides', fontsize = 14)
plt.xlabel('Minutes', fontsize = 12)
plt.ylabel('Count', fontsize = 12)
#sets the upper and lower bounds of the bins that are displayed in the plot
plt.xlim((0,50));
```



- This histogram and the summarized descriptive statistics of "duration_min" column indicates that the majority of rides last less than 20 minutes. The distribution is skewed to the right. Next, a boxplot of the age distribution of bike users will be created.

In [74]:
```
df_cleaned['member_age'].describe()
```

Out[74]:
```
count   174952.00
mean        36.20
std         10.12
min         20.00
25%         29.00
50%         34.00
75%         41.00
```

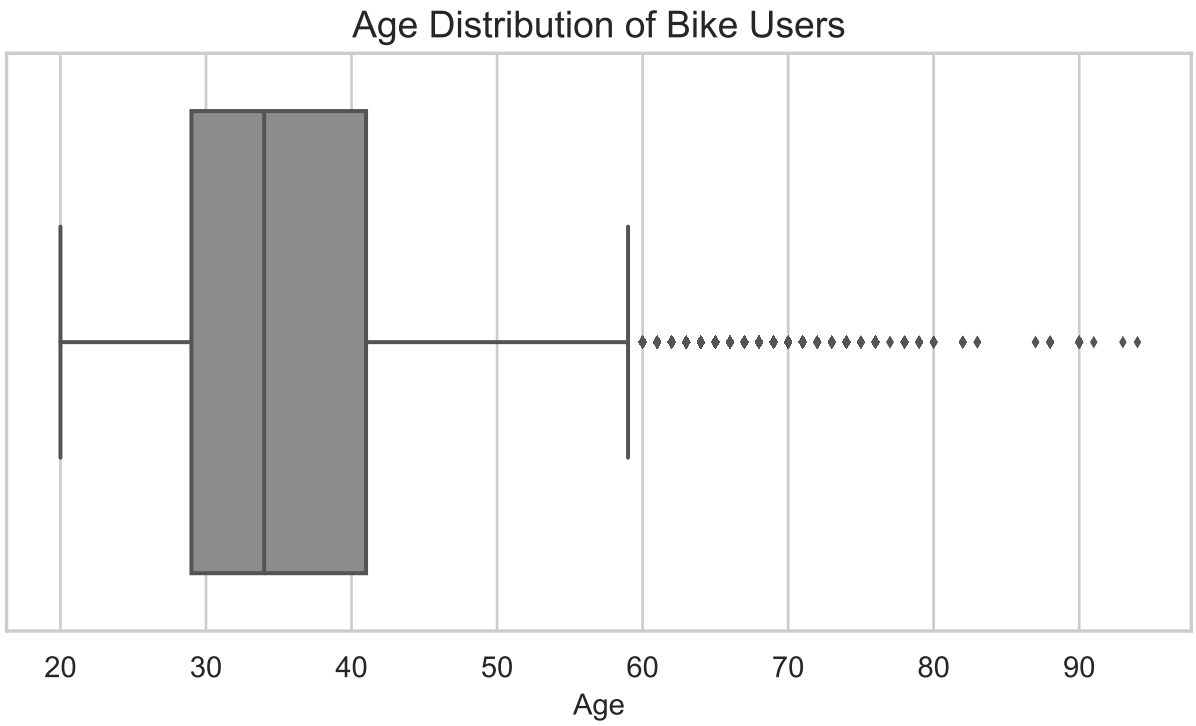
```
max        143.00
Name: member_age, dtype: float64
```

In [75]:
```
df_age = df_cleaned[df_cleaned['member_age']<100]
```

In [76]:
```
df_age['member_age'].describe()
```

Out[76]:
```
count   174877.00
mean        36.16
std          9.97
min         20.00
25%         29.00
50%         34.00
75%         41.00
max         94.00
Name: member_age, dtype: float64
```

In [77]:
```
#chooses the tuple of RGB colors
base_color = sb.color_palette()[7]
#sets style
sb.set_theme(style="whitegrid")
#sets the figure size
plt.figure(figsize=[8, 4])
#plots the boxplot
sb.boxplot(data = df_age, x = 'member_age', color=base_color, fliersize=2)
#sets title and labels
plt.title("Age Distribution of Bike Users", fontsize = 14)
plt.xlabel("Age", fontsize = 11);
```

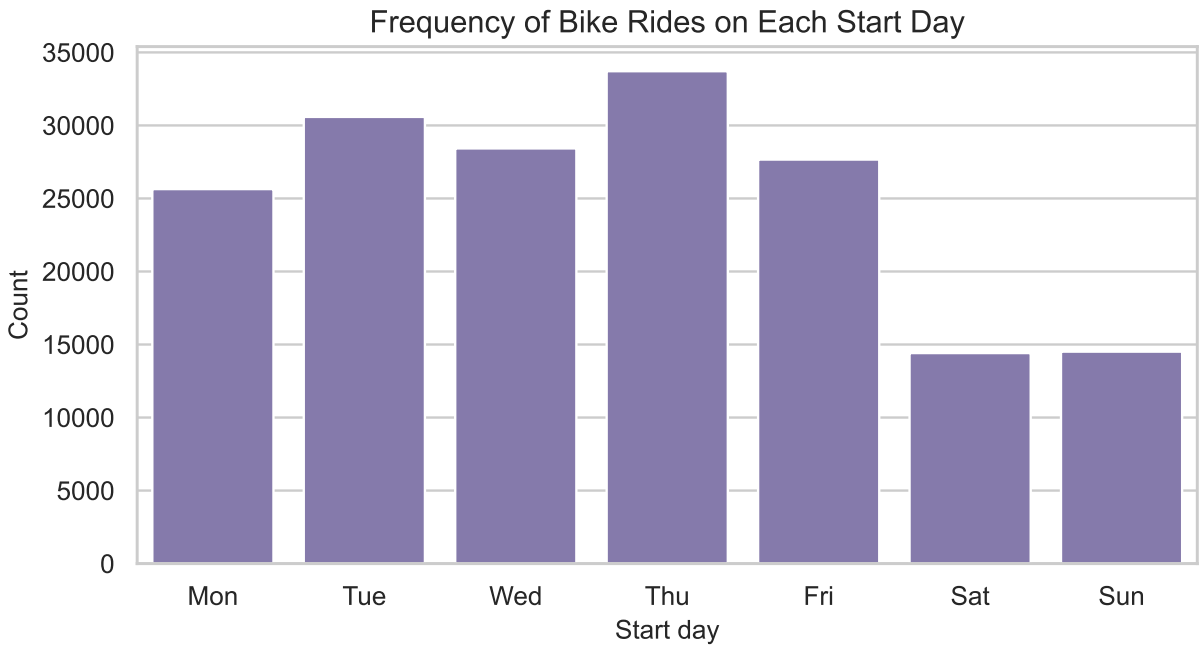


- From this boxplot and the summarized descriptive statistics of "member_age" column, it can be noticed that the mean age of bike users is about 36 years old. Also it appears that there are users with 100 years and more. Now, it would be nice to check the frequency of bike rides on each weekday.

In [78]:
```
df_cleaned.start_month.value_counts()
```

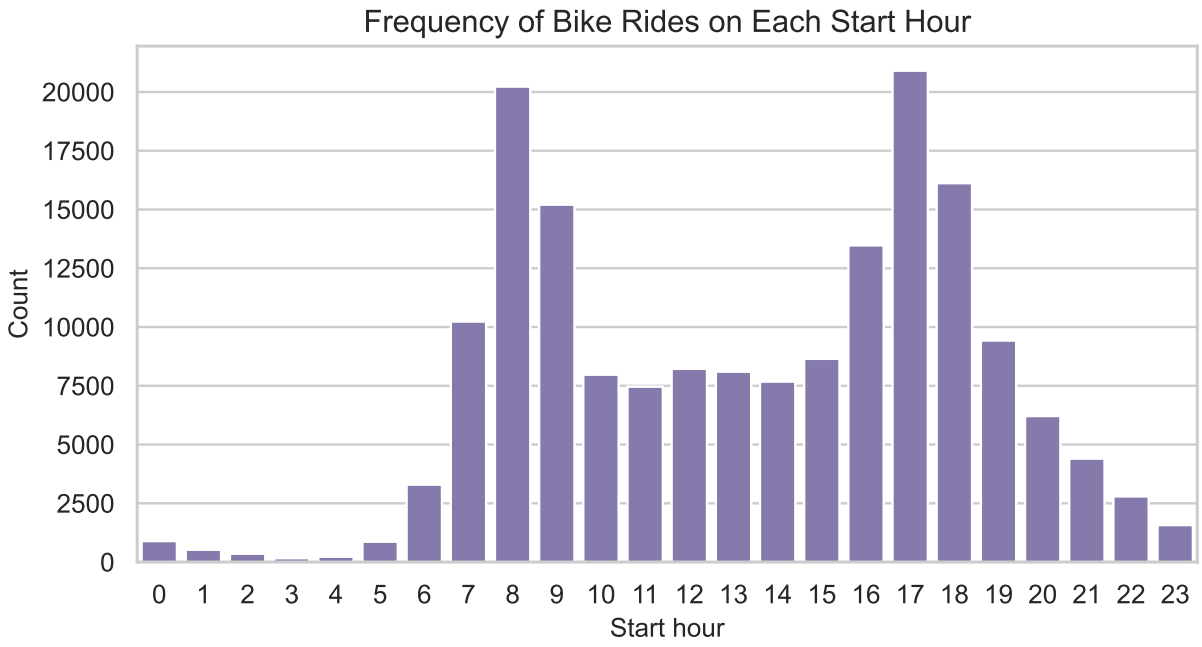
Out[78]:
```
Feb    174952
Name: start_month, dtype: int64
```

In [79]:
```
#sets the figure size
plt.figure(figsize=[8, 4])
#chooses the tuple of RGB colors
base_color = sb.color_palette()[4]
#defines de corder of the categories
cat_order = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
#plots the bar chart
sb.countplot(data = df_cleaned, x = 'start_day', order = cat_order, color = base_col
#sets title and labels
plt.title('Frequency of Bike Rides on Each Start Day', fontsize = 13)
plt.xlabel('Start day', fontsize = 11)
plt.ylabel('Count', fontsize = 11);
```



- This chart indicates that more rides are made on weekdays than on weekends.The next chart will check the frequency of bike ride starts for each hour of the day.

In [80]:
```
#sets the figure size
plt.figure(figsize=[8, 4])
#chooses the tuple of RGB colors
base_color = sb.color_palette()[4]
#plots the bar chart
sb.countplot(data = df_cleaned, x = 'start_hour', color=base_color)
#sets title and labels
plt.title('Frequency of Bike Rides on Each Start Hour', fontsize = 13)
plt.xlabel('Start hour', fontsize = 11)
plt.ylabel('Count', fontsize = 11);
```



- This chart shows that the peak hours of rentals are from 7:00 to 9:00 in the morning and from 4:00 to 6:00 in the evening.

7.1.2. Univariate Questions

Discuss the distribution(s) of your variable(s) of interest. Were there any unusual points? Did you need to perform any transformations?

> The main variable of interest, member gender, had no unusual points and requeired no transformartions to be performed. Some might believe that the "Other" category should be disconsidered, but it was decided that these identities different from male and female are part of the individual experience of gender and, therefore, should remain in the analysis.

Of the features you investigated, were there any unusual distributions? Did you perform any operations on the data to tidy, adjust, or change the form of the data? If so, why did you do this?

> Operations were made to assure better vizualizations in two cases. First, regarding the trip distribution. As the majoity of the trips lasted less than 13.15 minutes (Q3), only the rides that lasted up to 50 minutes were selected, leaving aside outliers, which lasted almost 24h. The second case was about the user age. There were some users with ages greater than 100 years (143 years maximum). This was most likely due to an error. So, it was decided to select only the age values below 100 years old, excluding these outliers from the analisys.

7.2.1. Bivariate Exploration

- In this second section, firstly, the boxplot of the duration distribution of bike rides per gender will be created.

In [81]:
```
df_f = df_cleaned.query('member_gender == "Female"')
df_f.duration_min.describe()
```

Out[81]:
```
count   40805.00
mean       12.98
std        29.78
min         1.02
25%         6.02
50%         9.45
75%        14.40
max      1386.58
Name: duration_min, dtype: float64
```

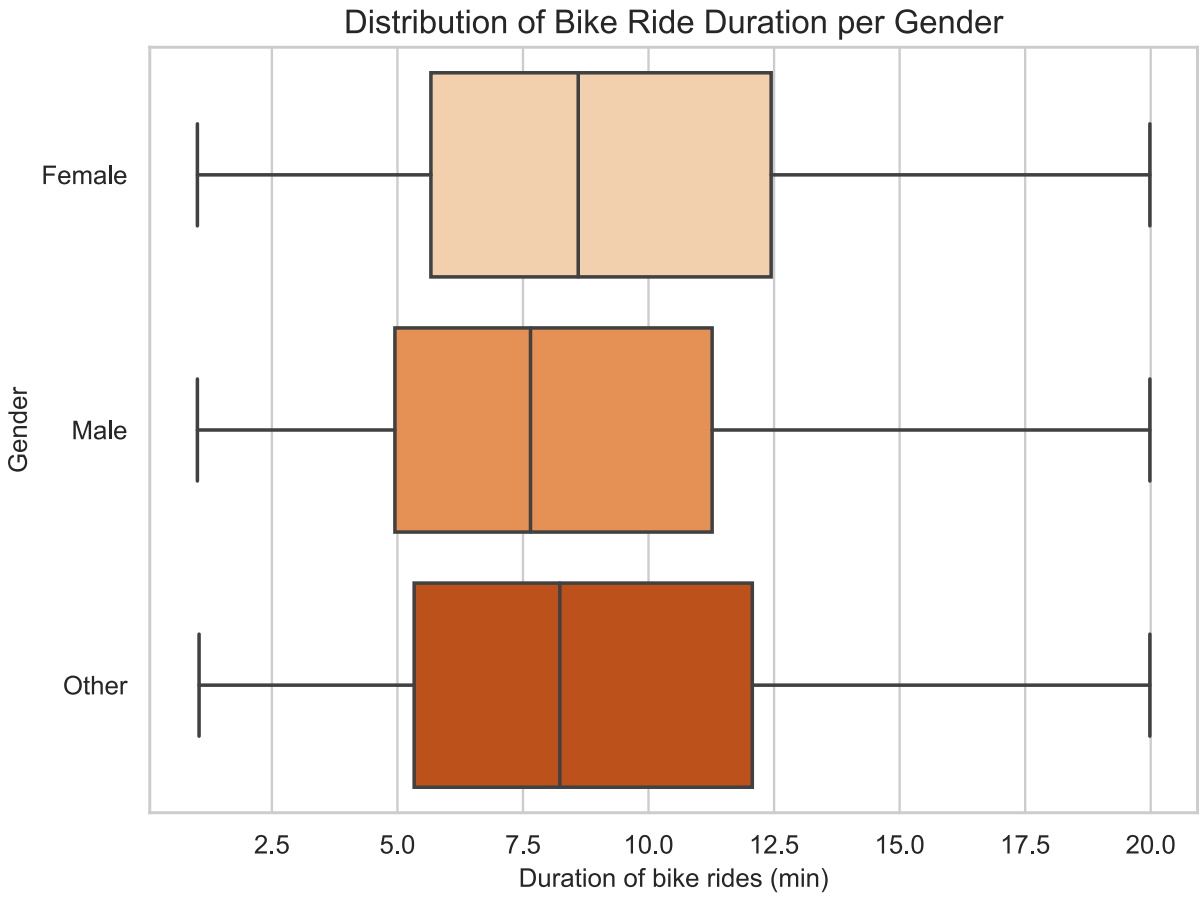
In [82]:
```
df_m = df_cleaned.query('member_gender == "Male"')
df_m.duration_min.describe()
```

Out[82]:
```
count   130500.00
mean        11.21
std         25.08
min          1.02
25%          5.18
50%          8.22
75%         12.68
max       1409.13
Name: duration_min, dtype: float64
```

In [83]:
```
df_o = df_cleaned.query('member_gender == "Other"')
df_o.duration_min.describe()
```

Out[83]:
```
count   3647.00
mean      16.62
std       58.77
min        1.05
25%        5.73
50%        9.27
75%       14.57
max     1375.20
Name: duration_min, dtype: float64
```

In [84]:
```
#sets the figure size
plt.figure(figsize=[8, 6])
#sets data do rides with less than 20 min of duration to avoid outliers
data = df_cleaned[df_cleaned.duration_min < 20]
#plots the boxplots
sb.boxplot(data = data, x = 'duration_min', y = 'member_gender', palette = 'Oranges'
#sets title and labels
plt.title("Distribution of Bike Ride Duration per Gender", fontsize = 14)
plt.xlabel("Duration of bike rides (min)", fontsize = 11)
plt.ylabel("Gender", fontsize = 11);
```



- These boxplots and the summarized statistics of "member_gender" for each of the categories shows that the distributions are very similar. Despite that, the male user category has the lowest median, interquartile range and mean. So, it is possible to say that male users usually take shorter rides, compared to other genders. Now, it would be nice to take a look at the frequency of users in different age groups per gender.

In [85]:
```
df_cleaned['member_age']. describe()
```

Out[85]:
```
count   174952.00
mean        36.20
std         10.12
min         20.00
25%         29.00
50%         34.00
75%         41.00
max        143.00
Name: member_age, dtype: float64
```

In [86]:
```
def age_group (x):
    """Returns the age categorie"""
    if (x >= 20 and x < 30):
        return "20-30"
    if (x >= 30 and x < 40):
        return "30-40"
    if (x >= 40 and x < 50):
        return "40-50"
    elif (x >= 50 and x < 60):
        return "50-60"
    else:
        return "60+"
```

In [87]:
```
df_cleaned['age_group'] = df_cleaned['member_age'].apply(lambda x: age_group(x))
```

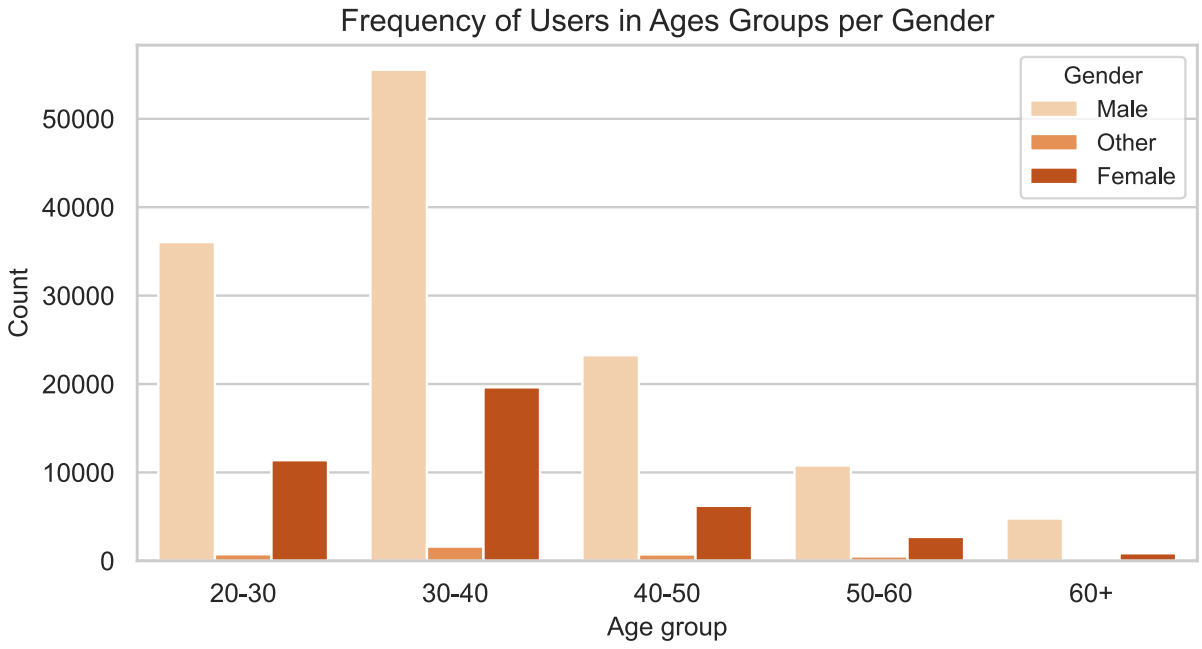
In [88]:
```
df_cleaned.head()
```

Out[88]:

	duration_sec	duration_min	start_time	end_time	start_month	end_month	start_day	end_d
0	52185	869.75	2019-02-28 17:32:10.145	2019-03-01 08:01:55.975	Feb	Mar	Thu	
1	61854	1030.90	2019-02-28 12:13:13.218	2019-03-01 05:24:08.146	Feb	Mar	Thu	
2	36490	608.17	2019-02-28 17:54:26.010	2019-03-01 04:02:36.842	Feb	Mar	Thu	
3	1585	26.42	2019-02-28 23:54:18.549	2019-03-01 00:20:44.074	Feb	Mar	Thu	
4	1793	29.88	2019-02-28 23:49:58.632	2019-03-01 00:19:51.760	Feb	Mar	Thu	

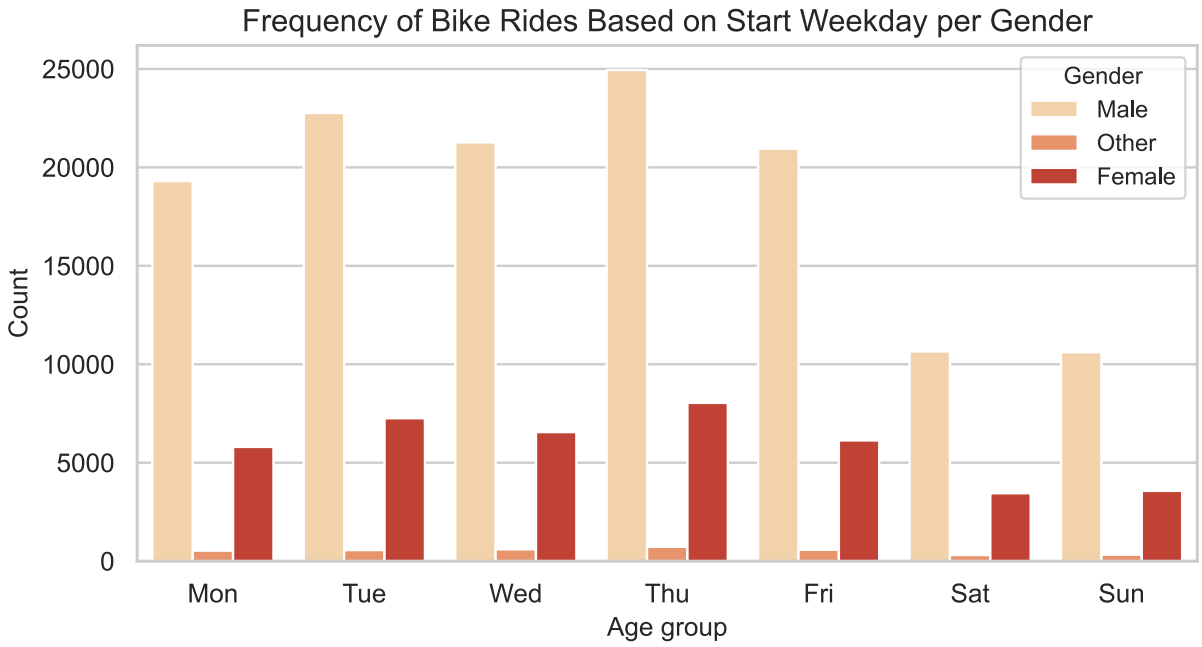
In [89]:
```
df_age2 = df_cleaned[df_cleaned['member_age']<100]
```

In [90]:
```
#sets the figure size
plt.figure(figsize=[8, 4])
#defines the order of the categories
cat_order = ['20-30','30-40','40-50','50-60','60+']
#plots the bar chart
sb.countplot(data = df_age2, x = 'age_group', order = cat_order, palette = 'Oranges'
#sets legend, title and label
plt.legend(title = 'Gender', title_fontsize = 10, prop ={'size': 10})
plt.title('Frequency of Users in Ages Groups per Gender', fontsize = 13)
plt.xlabel('Age group', fontsize = 11)
plt.ylabel('Count', fontsize = 11);
```



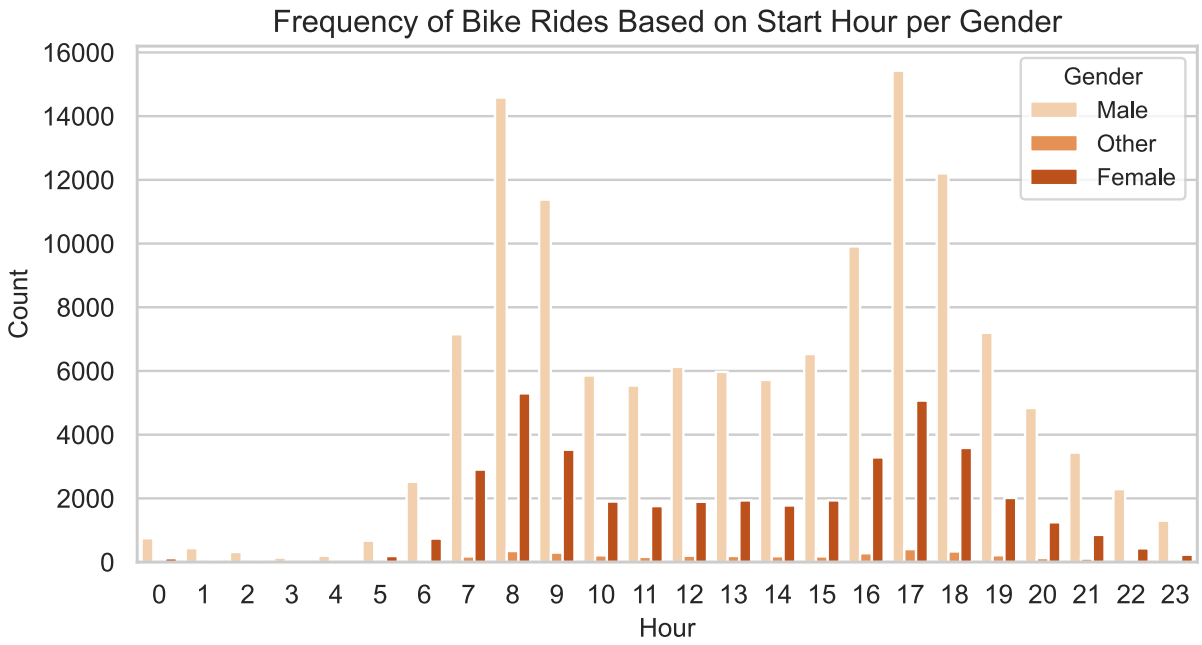
- From this chart, it is possible to notice a very similar trend for the three genders in all age categories, especially, for female and male. All genders have more users in the 30-40 years old group. Therefore, this bike service is usually used by a more mature public. Then, the frequency of bike rides based on start weekday per genderwill be checked.

In [91]:
```
#sets the figure size
plt.figure(figsize=[8, 4])
#defines the weekday order
cat_order = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
#plots the bar chart
sb.countplot(data = df_cleaned, x = 'start_day', order = cat_order, palette = 'OrRd'
#sets legend, title and label
plt.legend(title = 'Gender', title_fontsize = 10, prop ={'size': 10})
plt.title('Frequency of Bike Rides Based on Start Weekday per Gender', fontsize = 13
plt.xlabel('Age group', fontsize = 11)
plt.ylabel('Count', fontsize = 11);
```



- From this chart, it can be seen that there are not important differences between genders concearning the start day of the ride. For all gender categories, most rides are done during weekdays. Next, the frequency of bike rides based on start hour per gender will be checked.

In [92]:
```
#sets the figure size
plt.figure(figsize=[8, 4])
#plots the bar chart
sb.countplot(data = df_cleaned, x = 'start_hour', hue = 'member_gender', palette = '
#sets legend, title and label
plt.legend(title = 'Gender', title_fontsize = 10, prop ={'size': 10})
plt.title('Frequency of Bike Rides Based on Start Hour per Gender', fontsize = 13)
plt.xlabel('Hour', fontsize = 11)
plt.ylabel('Count', fontsize = 11);
```



- Once again, all genders appear to follow the same trend, although this is less perceptive for the "other" category. The users in all genders take trips mostly between 7:00 and 9:00 in the morning and between 4:00 and 6:00 in the evening. These periods correspond to working hours in most companies. It might suggest that all users, regardless of their gender, take bike rides to go to work and to come back home.

7.2.2. Bivariate Questions

Talk about some of the relationships you observed in this part of the investigation. How did the feature(s) of interest vary with other features in the dataset?

> There were no important differences observed in gender categories use of the bicycle sharing system, in terms of the users' age, the trip duration, and the day and hour that this trip starts. All genders show similar distributions of bike ride duration and frequency in age groups, with greater values in the 30-40-year-old group. Regardless of their gender, users usually take rides during weekdays, with peak hours of rentals in periods when employees are expected to travel to their jobs and to travel back home.

Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

> No interesting relationships between the other features, except for the main feature of interest, were observed.

7.3.1. Multivariate Exploration

- In this last section, a further look will be taken on the relation between the gender of users and the hourly usage of the bike service during weekdays. For this purpose, a heat map will

be created.

In [93]:

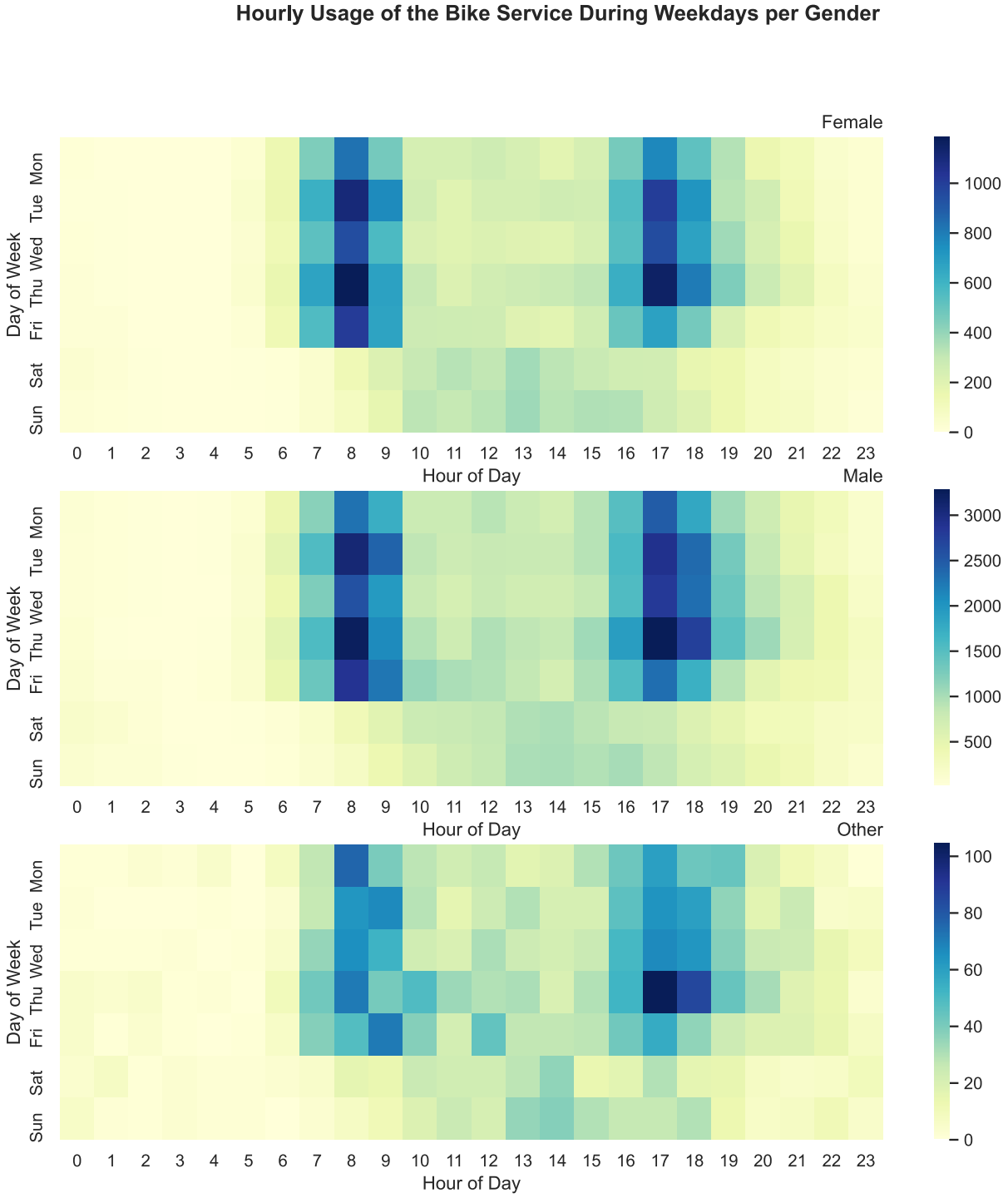
```
##sets the figure size
plt.figure(figsize=(12,12))
#sets title
plt.suptitle('Hourly Usage of the Bike Service During Weekdays per Gender', fontsize

df_cleaned['start_day'] = pd.Categorical(df_cleaned['start_day'],
                                         categories=['Mon','Tue','Wed','Thu','Fri','
                                         ordered=True)

#sets 3 rows, 1 col, subplot 1
plt.subplot(3, 1, 1)
#counts number of points in each "bin"
female = df_cleaned.query('member_gender == "Female"')
f_counts = female.groupby(['start_day', 'start_hour']).size()
f_counts = f_counts.reset_index(name='count')
f_counts = f_counts.pivot(index='start_day', columns='start_hour', values='count')
#plots heatmap
sb.heatmap(f_counts, cmap='YlGnBu')
#sets title and label
plt.title('Female', loc='right')
plt.xlabel('Hour of Day')
plt.ylabel('Day of Week')

#sets 3 rows, 1 col, subplot 2
plt.subplot(3, 1, 2)
#counts number of points in each "bin"
male = df_cleaned.query('member_gender == "Male"')
m_counts = male.groupby(['start_day', 'start_hour']).size()
m_counts = m_counts.reset_index(name='count')
#plots heatmap
m_counts = m_counts.pivot(index='start_day', columns='start_hour', values='count')
sb.heatmap(m_counts, cmap='YlGnBu')
#sets title and label
plt.title('Male', loc='right')
plt.xlabel('Hour of Day')
plt.ylabel('Day of Week')

#sets 3 rows, 1 col, subplot 3
plt.subplot(3, 1, 3)
#counts number of points in each "bin"
other = df_cleaned.query('member_gender == "Other"')
o_counts = other.groupby(['start_day', 'start_hour']).size()
o_counts = o_counts.reset_index(name='count')
o_counts = o_counts.pivot(index='start_day', columns='start_hour', values='count')
#plots heatmap
sb.heatmap(o_counts, cmap='YlGnBu')
#sets title and label
plt.title('Other', loc='right')
plt.xlabel('Hour of Day')
plt.ylabel('Day of Week');
```



- The above heat map shows clearly that the bike rides taken by female, male and others start mostly on weekdays and between 7:00 and 9:00 in the morning and between 4:00 and 6:00 in the evening.

7.3.2. Multivariate Questions

Talk about some of the relationships you observed in this part of the investigation. Were there features that strengthened each other in terms of looking at your feature(s) of interest?

> The multivariate vizualization reinforced the earlier relationships that were observed. All genders show a similar trend in hourly usage of the bike service

> during weekdays. Most of the rides start during the working days and their highest concentration corresponds to rush hours.

Were there any interesting or surprising interactions between features?

> No interesting relationships between the other features, except for the main feature of interest, were observed.

8. Main Findings

> In this analysis, I was most interested in figuring out how genders ("member_gender") differ from each other in the use of this bike service, concerning the users age, the trip duration, and the day and hour that this trip starts.
>
> The data exploration demonstrated that there were no important differences observed in the way the differnt genders use the bicycle sharing system, in terms of the selected variables. All genders show similar distributions of bike ride duration, although male users usually take slightly shorter rides. The three genders frequency per age group also follows a similar trend, with greater values in the 30-40-year-old group. Therefore, this bike service appears to be more frequently used by a more mature public. Finally, regardless of their gender, users usually take rides during the weekdays, with peak hours of rentals corresponding to rush hours periods, when employees are expected to travel to and from work.

9. References

- https://www.geeksforgeeks.org/convert-birth-date-to-age-in-pandas/
- https://stackoverflow.com/questions/43956335/convert-float64-column-to-int64-in-pandas
- https://stackoverflow.com/questions/51603690/extract-day-and-month-from-a-datetime-object
- https://stackoverflow.com/questions/18674064/how-do-i-insert-a-column-at-a-specific-column-index-in-pandas
- https://stackoverflow.com/questions/17582137/ipython-notebook-svg-figures-by-default
- https://stackoverflow.com/questions/36519086/how-to-get-rid-of-unnamed-0-column-in-a-pandas-dataframe
- https://medium.com/@morganjonesartist/color-guide-to-seaborn-palettes-da849406d44f
- https://www.dataforeverybody.com/seaborn-legend-change-location-size/