Part I - Ford GoBike System Data Exploration

by Joud Hijaz

Table of Content

Introduction

> The `Ford GoBike System` dataset includes information about individual rides made in a bike-sharing system covering the greater San Francisco Bay area. Data columns such as ride durations, start and end station names, station locations and the customer infomration are provided in the dataset.

```
# import all packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme(sns.set_style('dark'))
sns.set_palette('Paired')
%matplotlib inline
plt.style.use('ggplot')
```

```
%ls ./
```

```
201902-fordgobike-tripdata.csv  sample_data/
```

```
# read data
df = pd.read_csv('201902-fordgobike-tripdata.csv')
```

```
df.head()
```

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_
0	52185	2019-02-28 17:32:10.1450	2019-03-01 08:01:55.9750	21.0	Montgomery St BART Station (Market St at 2nd St)	
1	42521	2019-02-28 18:53:21.7890	2019-03-01 06:42:03.0560	23.0	The Embarcadero at Steuart St	
2	61854	2019-02-28 12:13:13.2180	2019-03-01 05:24:08.1460	86.0	Market St at Dolores St	
3	36490	2019-02-28 17:54:26.0100	2019-03-01 04:02:36.8420	375.0	Grove St at Masonic Ave	
4	1585	2019-02-28 23:54:18.5490	2019-03-01 00:20:44.0740	7.0	Frank H Ogawa Plaza	

```
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
```

Preliminary Wrangling

Data Cleaning

> - The data set is small, so I cleaned on the original data table.
> - The data contains one month rides from 2019-02-01 to 2019-02-28, with total 183,413 rows.
> - There are 329 unique stations and 4,646 unique bikes.
> - 90% of the users are subscribers, the rest are customers.
> - 70% of the users are males, 20% are females, and there are 10% for others.
> - The average duration for the rides are 12mins.
> - The youngest user is born in 2001, while the oldest in 1878.

```
# change start and end time from string to datetime object
df['start_time'] = pd.to_datetime(df['start_time'])
df['end_time'] = pd.to_datetime(df['end_time'])
```

```
print('The Earliest Date', df['start_time'].min())
print('The Latest Date: ', df['start_time'].max())
```

```
The Earliest Date 2019-02-01 00:00:20.636000
The Latest Date:  2019-02-28 23:59:18.548000
```

```
# change ids from integer/float to string type
df['start_station_id'] = df['start_station_id'].astype('string')
df['end_station_id'] = df['end_station_id'].astype('string')
df['bike_id'] = df['bike_id'].astype('string')
```

```
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  datetime64[ns]
 2   end_time                 183412 non-null  datetime64[ns]
 3   start_station_id         183215 non-null  string
 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  string
 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  string
 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: datetime64[ns](2), float64(5), int64(1), object(5), string(3)
memory usage: 22.4+ MB
```

```
print('No. of Unique Start Stations:', df.start_station_id.nunique())
print('No. of Unique End Stations:', df.end_station_id.nunique())
print('No. of Unique Bikes:', df.bike_id.nunique())
```

```
No. of Unique Start Stations: 329
No. of Unique End Stations: 329
No. of Unique Bikes: 4646
```

```
# check for duplications
sum(df.duplicated())
```

```
0
```

```
# user info
print(df.user_type.value_counts())
print(df.member_gender.value_counts())
print(df.bike_share_for_all_trip.value_counts())
```

```
user_type
Subscriber    163544
Customer       19868
Name: count, dtype: int64
member_gender
Male      130651
Female     40844
Other       3652
Name: count, dtype: int64
bike_share_for_all_trip
No     166053
Yes     17359
Name: count, dtype: int64
```

```
# numeric values description
df[['duration_sec', 'member_birth_year']].describe()
```

	duration_sec	member_birth_year
count	183412.000000	175147.000000
mean	726.078435	1984.806437
std	1794.389780	10.116689
min	61.000000	1878.000000
25%	325.000000	1980.000000
50%	514.000000	1987.000000
75%	796.000000	1992.000000
max	85444.000000	2001.000000

Feature Engineering

> For a clear exploratory and explanatory analysis, I created features based on the current variables.
>
> 1. `dow` : day of the week for start date
> 2. `hour` : hour of the start date
> 3. `duration_min` : duration by minutes
> 4. `distance` : the direct distance based on start and end longitude, latitude
> 5. `member_age` : rider age of the riding based on their dob and ride start date

```
# add dow
df['dow'] = df['start_time'].dt.day_name()


# add moy
df['hour'] = df['start_time'].dt.hour


# add duration_min
df['duration_min'] = round(df['duration_sec']/60,2)


# add distance
# reference: https://kanoki.org/2019/12/27/how-to-calculate-distance-in-python-and-pandas-us
def haversine_vectorize(lon1, lat1, lon2, lat2):

    lon1, lat1, lon2, lat2 = map(np.radians, [lon1, lat1, lon2, lat2])

    newlon = lon2 - lon1
    newlat = lat2 - lat1
```

```
    haver_formula = np.sin(newlat/2.0)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(newlon/2.0)

    dist = 2 * np.arcsin(np.sqrt(haver_formula ))
    km = 6367 * dist #6367 for distance in KM for miles use 3958
    return round(km,2)

df['distance'] = haversine_vectorize(df['start_station_longitude'],
                                     df['start_station_latitude'],
                                     df['end_station_longitude'],
                                     df['end_station_latitude'])


# df.head()


# add member_age
df['member_age'] = df['start_time'].dt.year - df['member_birth_year']


df.head()
```

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_st
0	52185	2019-02-28 17:32:10.145	2019-03-01 08:01:55.975	21.0	Montgomery St BART Station (Market St at 2nd St)	
1	42521	2019-02-28 18:53:21.789	2019-03-01 06:42:03.056	23.0	The Embarcadero at Steuart St	
2	61854	2019-02-28 12:13:13.218	2019-03-01 05:24:08.146	86.0	Market St at Dolores St	
3	36490	2019-02-28 17:54:26.010	2019-03-01 04:02:36.842	375.0	Grove St at Masonic Ave	
4	1585	2019-02-28 23:54:18.549	2019-03-01 00:20:44.074	7.0	Frank H Ogawa Plaza	

5 rows × 21 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 21 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   duration_sec             183412 non-null  int64
 1   start_time               183412 non-null  datetime64[ns]
 2   end_time                 183412 non-null  datetime64[ns]
```

```
 3   start_station_id         183215 non-null  string
 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  string
 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  string
 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
 16  dow                      183412 non-null  object
 17  hour                     183412 non-null  int32
 18  duration_min             183412 non-null  float64
 19  distance                 183412 non-null  float64
 20  member_age               175147 non-null  float64
dtypes: datetime64[ns](2), float64(8), int32(1), int64(1), object(6), string(3)
memory usage: 28.7+ MB
```

```
# save cleaned df to a file
df.to_csv('fordgobike_clean.csv', index=False)
```

What is the structure of your dataset?

> There are 183,412 rides in the dataset with 21 features.
>
> - duration_sec
> - start_time
> - end_time
> - start_station_id
> - start_station_name
> - start_station_latitude
> - start_station_longitude
> - end_station_id
> - end_station_name
> - end_station_latitude
> - end_station_longitude
> - bike_id
> - user_type
> - member_birth_year
> - member_gender
> - bike_share_for_all_trip
> - dow

> - hour
> - duration_min
> - distance
> - member_age

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

> I'm most interested in the spatial and temporal features of the rides to understand the riding habit of people in San Francisco Bay area.

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

> I expect time, duration, distance and location information will be the most important in the investigation.

Univariate Exploration

Temporal Analysis

Question #1. which day of the week has the most rides?

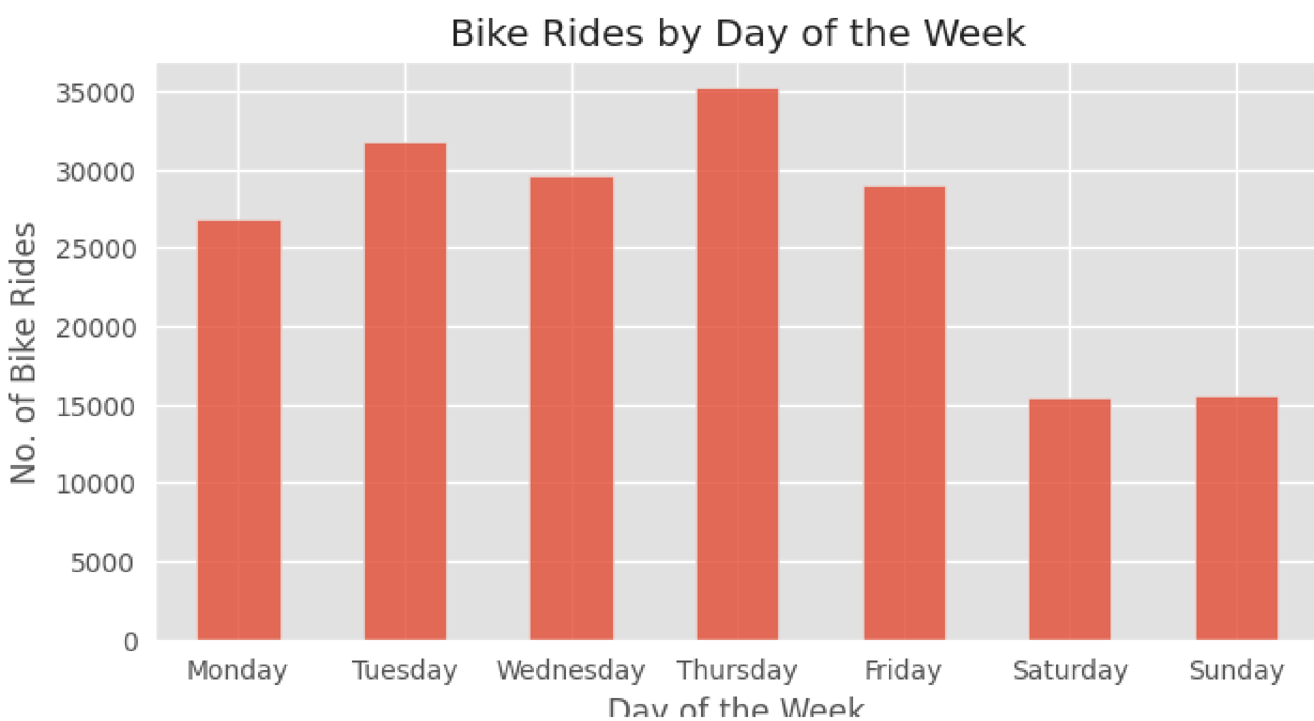
```
df['dow'].value_counts()
```

dow	count
Thursday	35197
Tuesday	31813
Wednesday	29641
Friday	28981
Monday	26852
Sunday	15523
Saturday	15405

dtype: int64

Visualization #1. day of the week bar plot

```
# barplot for dow
# order day of the week
ordinal_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'
dow_cnt = df['dow'].value_counts().reindex(ordinal_week)

plt.figure(figsize=(8,4))

dow_cnt.plot(kind='bar', alpha=0.8)
plt.title('Bike Rides by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('No. of Bike Rides')
plt.xticks(rotation=360);
```



Observation #1.

> There are more rides during weekdays, especially on Thursday, the No. of rides reaches 35k in total. I guess in San Francsisco Bay area, people use shared bikes for work quite often, while during weekends, the No. of rides clearly decreased.

Question #2. which hour of the day is the most busy hour for bike riding?

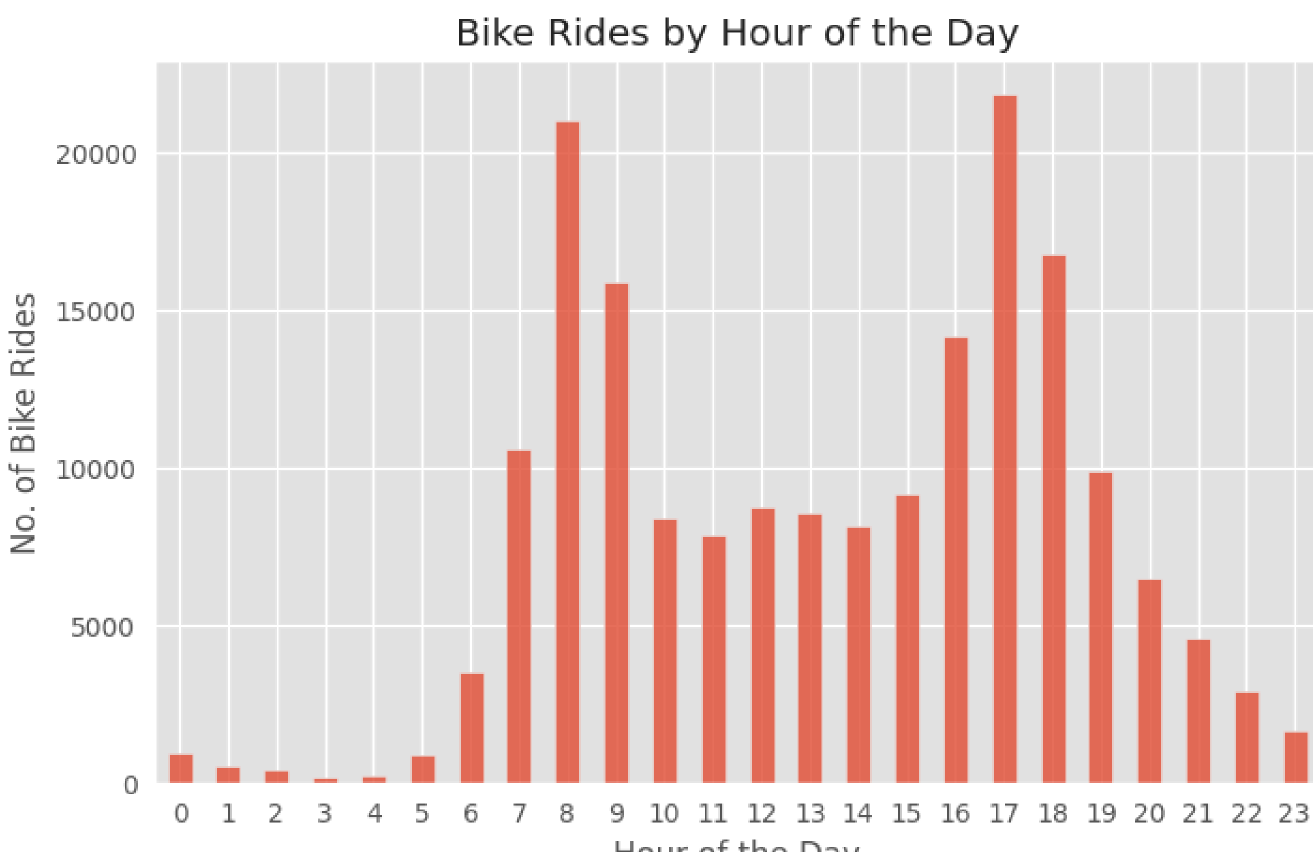
```
df['hour'].value_counts().sort_index()
```

hour	count
0	925
1	548
2	381
3	174
4	235
5	896
6	3485
7	10614
8	21056
9	15903
10	8364
11	7884
12	8724
13	8551
14	8152
15	9174
16	14169
17	21864
18	16827
19	9881
20	6482
21	4561
22	2916
23	1646

dtype: int64

Visualization #2. hour of the day bar plot

```
#barplot for hour of the day
hour_cnt = df['hour'].value_counts().sort_index()

plt.figure(figsize=(8,5))

hour_cnt.plot(kind='bar', alpha=0.8)
plt.title('Bike Rides by Hour of the Day')
plt.xlabel('Hour of the Day')
plt.ylabel('No. of Bike Rides')
plt.xticks(rotation=360);
```



Observation #2.

> The second visualization supports my assumption that people use bikes mainly for commuting to work. The peak hours are during morning office time 7am to 9am, and after work time 4pm to 6pm, at 8am and 5pm the numbers reach the highest points. Between 10am to 3pm, the no. of bike rides are very steady, and after 6pm, the number decreased hour by hour till midnight.

Question #3. how long dose people ride?

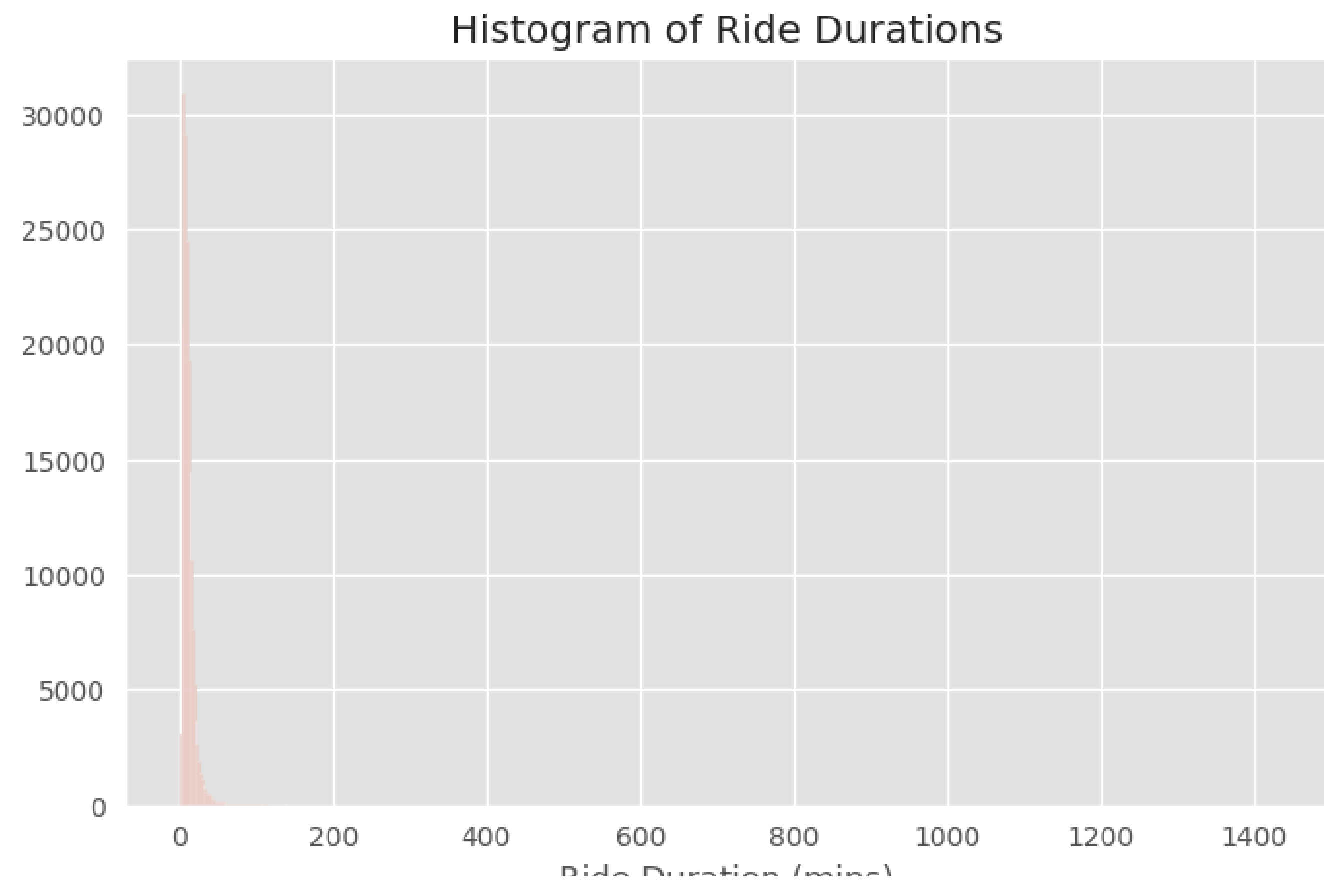
```
df['duration_min'].describe()
```

	duration_min
count	183412.000000
mean	12.101301
std	29.906501
min	1.020000
25%	5.420000
50%	8.570000
75%	13.270000
max	1424.070000

dtype: float64

Visualization #3. ride durations (mins) histogram

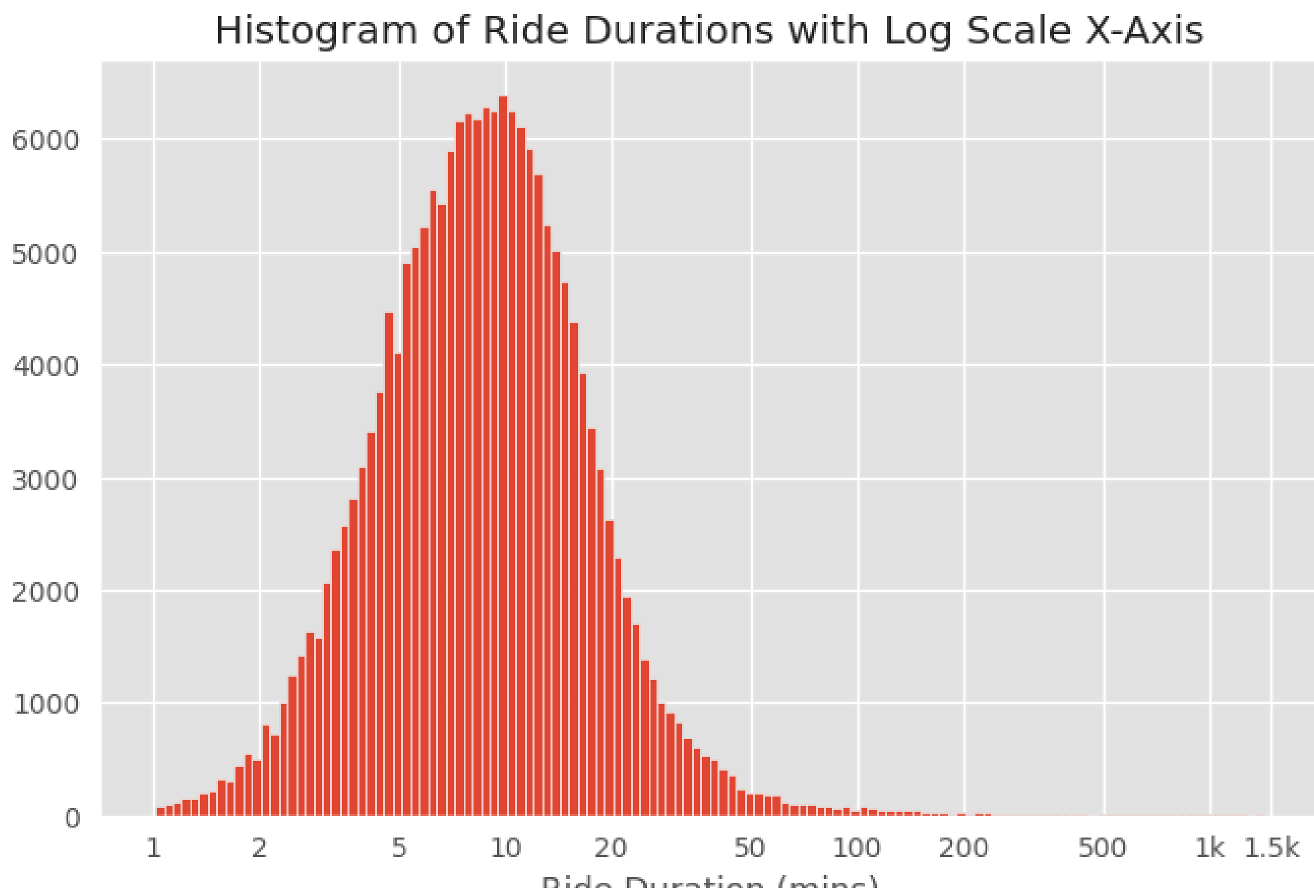
```
#set bins
binsize = 2
bins = np.arange(0, df['duration_min'].max()+binsize, binsize)

plt.figure(figsize=[8, 5])

plt.hist(data = df, x = 'duration_min', bins = bins)
plt.xlabel('Ride Duration (mins)')
plt.title('Histogram of Ride Durations')
plt.show()
```




```
np.log10(df['duration_min'].min())
```

```
0.008600171761917567
```

```
# there's a long tail in the distribution, so let's put it on a log scale instead

log_binsize = 0.025
bins = 10 ** np.arange(0.005, np.log10(df['duration_min'].max())+log_binsize, log_binsize)

plt.figure(figsize=[8, 5])

plt.hist(data = df, x = 'duration_min', bins = bins)
plt.xscale('log')
plt.title('Histogram of Ride Durations with Log Scale X-Axis')
plt.xticks([1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 1500], [1, 2, 5, 10, 20, 50, 100, 200,
plt.xlabel('Ride Duration (mins)')
plt.show()
```




```
df.loc[df['duration_min']>=1400]
```

	duration_sec	start_time	end_time	start_station_id	start_station_name	sta
85465	84548	2019-02-16 15:48:25.029	2019-02-17 15:17:33.080	3.0	Powell St BART Station (Market St at 4th St)	
101361	85444	2019-02-13 17:59:55.124	2019-02-14 17:43:59.954	5.0	Powell St BART Station (Market St at 5th St)	

2 rows × 21 columns

```
df.loc[df['duration_min']>=1000].head(10)
```

	duration_sec	start_time	end_time	start_station_id	start_station_name	sta
2	61854	2019-02-28 12:13:13.218	2019-03-01 05:24:08.146	86.0	Market St at Dolores St	
3401	62452	2019-02-28 00:04:01.344	2019-02-28 17:24:54.137	154.0	Doyle St at 59th St	
5203	83195	2019-02-27 14:47:23.181	2019-02-28 13:53:58.433	243.0	Bancroft Way at College Ave	
7268	66065	2019-02-27 15:00:20.639	2019-02-28 09:21:26.336	349.0	Howard St at Mary St	
8631	81549	2019-02-27 09:41:38.552	2019-02-28 08:20:48.386	138.0	Jersey St at Church St	
14381	70211	2019-02-26 17:08:16.897	2019-02-27 12:38:28.436	80.0	Townsend St at 5th St	
29922	70925	2019-02-24 07:08:31.270	2019-02-25 02:50:36.590	375.0	Grove St at Masonic Ave	
31295	70050	2019-02-23 21:46:00.982	2019-02-24 17:13:31.412	34.0	Father Alfred E Boeddeker Park	
32098	69980	2019-02-23 19:52:25.335	2019-02-24 15:18:46.072	21.0	Montgomery St BART Station (Market St at 2nd St)	
33431	69620	2019-02-23 16:33:41.580	2019-02-24 11:54:02.408	90.0	Townsend St at 7th St	

10 rows × 21 columns

Observation #3.

> The histogram for ride duration is a long-tailed distribution, after plotting the X axis on log scale, it is clear that the majority of the rides are within 50mins, however, there are some rides have long durations, such as 100mins, 500mins, and the highest reaches around 1500mins, that is 25h. The long duration mostly due to people keep the bike overnight.

Spatial Analysis

Question #4. how far do people travel?

```
df['distance'].describe()
```

	distance
count	183412.000000
mean	1.689620
std	1.096911
min	0.000000
25%	0.910000
50%	1.430000
75%	2.220000
max	69.430000

dtype: float64

```
df.loc[df['distance']==69.43]
```

	duration_sec	start_time	end_time	start_station_id	start_station_name	sta
112038	6945	2019-02-12 14:28:44.402	2019-02-12 16:24:30.158	21.0	Montgomery St BART Station (Market St at 2nd St)	

1 rows × 21 columns

```
# the maximum of 70km riding distance is an outlier
df.loc[df['distance']>=10]
```

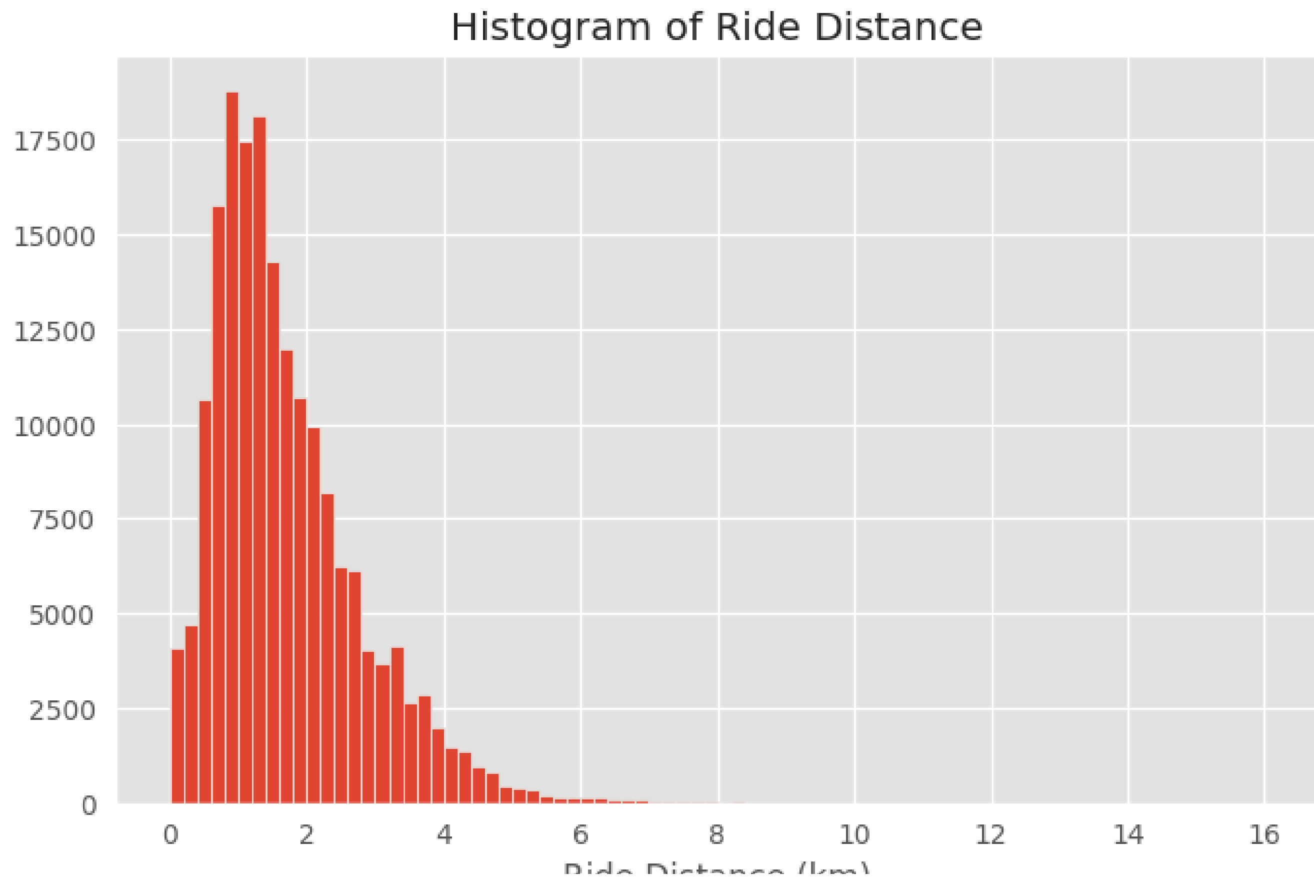
	duration_sec	start_time	end_time	start_station_id	start_station_name	sta
19827	2229	2019-02-26 15:11:44.523	2019-02-26 15:48:54.373	227.0	Foothill Blvd at Fruitvale Ave	
50859	3225	2019-02-21 17:51:18.986	2019-02-21 18:45:04.085	167.0	College Ave at Harwood Ave	
84701	16022	2019-02-17 12:39:48.765	2019-02-17 17:06:51.472	163.0	Lake Merritt BART Station	
85529	8957	2019-02-17 12:38:50.477	2019-02-17 15:08:08.352	163.0	Lake Merritt BART Station	
87602	4378	2019-02-17 00:27:13.613	2019-02-17 01:40:11.883	9.0	Broadway at Battery St	
89787	1800	2019-02-16 14:15:06.336	2019-02-16 14:45:06.488	201.0	10th St at Fallon St	
112038	6945	2019-02-12 14:28:44.402	2019-02-12 16:24:30.158	21.0	Montgomery St BART Station (Market St at 2nd St)	
121514	1792	2019-02-11 14:39:16.299	2019-02-11 15:09:09.130	230.0	14th St at Mandela Pkwy	
138857	57059	2019-02-07 12:17:12.295	2019-02-08 04:08:11.319	7.0	Frank H Ogawa Plaza	
153112	2216	2019-02-06 13:05:00.691	2019-02-06 13:41:57.678	219.0	Marston Campbell Park	
161775	2357	2019-02-05 13:14:18.246	2019-02-05 13:53:35.665	201.0	10th St at Fallon St	

11 rows × 21 columns

Visualization #4. ride distance histogram

```
# set bins and remove the ourlier by setting the maximum bin edge to 16+binsize
binsize = 0.2
bins = np.arange(0, 16+binsize, binsize)

plt.figure(figsize=[8, 5])

plt.hist(data = df, x = 'distance', bins = bins)
plt.xlabel('Ride Distance (km)')
```

```
plt.title('Histogram of Ride Distance')
plt.show()
```



Observation #4.

> The histogram of ride distance is right skewed with most of the riding less than 8km. However, there are some rides with long distances, such as 10km and even 15km.

Questions #5. which is the busiest start/end station throught the month?

```
df['start_station_name'].value_counts().to_frame().head(10)
```

start_station_name	count
Market St at 10th St	3904
San Francisco Caltrain Station 2 (Townsend St at 4th St)	3544
Berry St at 4th St	3052
Montgomery St BART Station (Market St at 2nd St)	2895
Powell St BART Station (Market St at 4th St)	2760
San Francisco Ferry Building (Harry Bridges Plaza)	2710
San Francisco Caltrain (Townsend St at 4th St)	2703
Powell St BART Station (Market St at 5th St)	2327
Howard St at Beale St	2293
Steuart St at Market St	2283

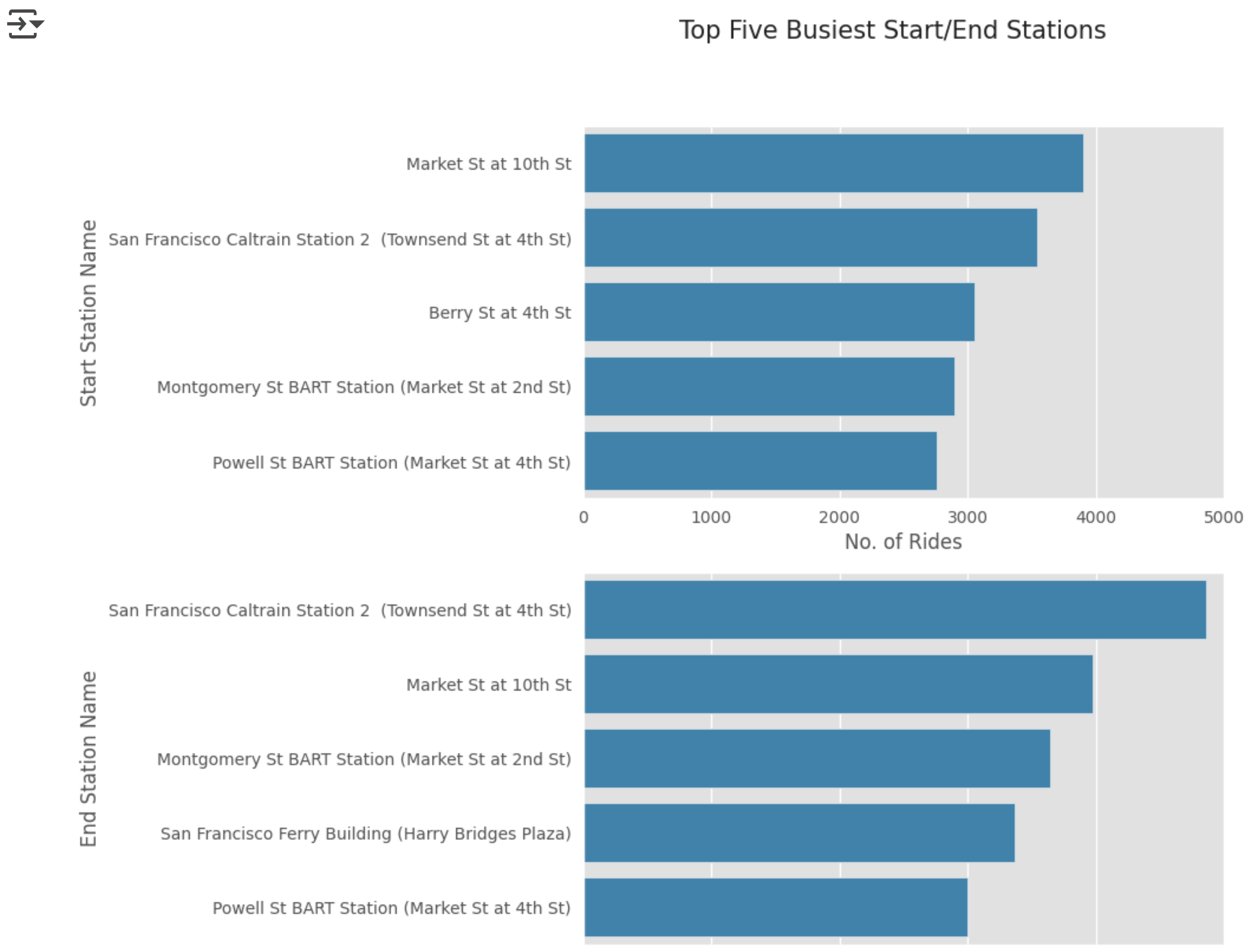
```
df['end_station_name'].value_counts().to_frame().head(10)
```

end_station_name	count
San Francisco Caltrain Station 2 (Townsend St at 4th St)	4857
Market St at 10th St	3973
Montgomery St BART Station (Market St at 2nd St)	3647
San Francisco Ferry Building (Harry Bridges Plaza)	3368
Powell St BART Station (Market St at 4th St)	2997
San Francisco Caltrain (Townsend St at 4th St)	2947
Berry St at 4th St	2872
The Embarcadero at Sansome St	2512
Powell St BART Station (Market St at 5th St)	2353
Steuart St at Market St	2338

Visualization #5. bar plot for the busiest stations

```
start_st = df['start_station_name'].value_counts()
end_st = df['end_station_name'].value_counts()
```

```
start_order = start_st.index.to_list()[0:5]
end_order = end_st.index.to_list()[0:5]


# countplot with seaborn
fig, ax = plt.subplots(nrows=2, figsize=(7, 9))
base_color = sns.color_palette()[1]
sns.countplot(y='start_station_name', data=df, order=start_order, color=base_color, ax=ax[0]
sns.countplot(y='end_station_name', data=df, order=end_order, color=base_color, ax=ax[1])
ax[0].set_xlabel('No. of Rides')
ax[1].set_xlabel('No. of Rides')
ax[0].set_ylabel('Start Station Name')
ax[1].set_ylabel('End Station Name')
ax[0].set_xlim((0, 5000))
ax[1].set_xlim((0, 5000))

fig.suptitle('Top Five Busiest Start/End Stations', fontsize=15);
```



Observation #5.

> `Market St at 10th St` and `San Francisco Caltrain Station 2` are two busiest stations for both start and end stations. During the whole month, there are around 5000 rides ended at San Francisco Caltrain Station 2. Besides there two stations, Montgomery St BART Station, Powell St BART Station, Berry St at 4th St and San Francisco Ferry Building are busy riding stations as well.

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

> Both the riding duration and distance distributions are right skewed with a long tail.

> The duration variable is quite spread out, so I looked at the data using a log transform.

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?

> The distance variable has an outlier that is way larger than normal, so I removed the outlier in the visualization.

Bivariate Exploration

Question #6. what's the difference between customer and subscribers in terms of travel time?

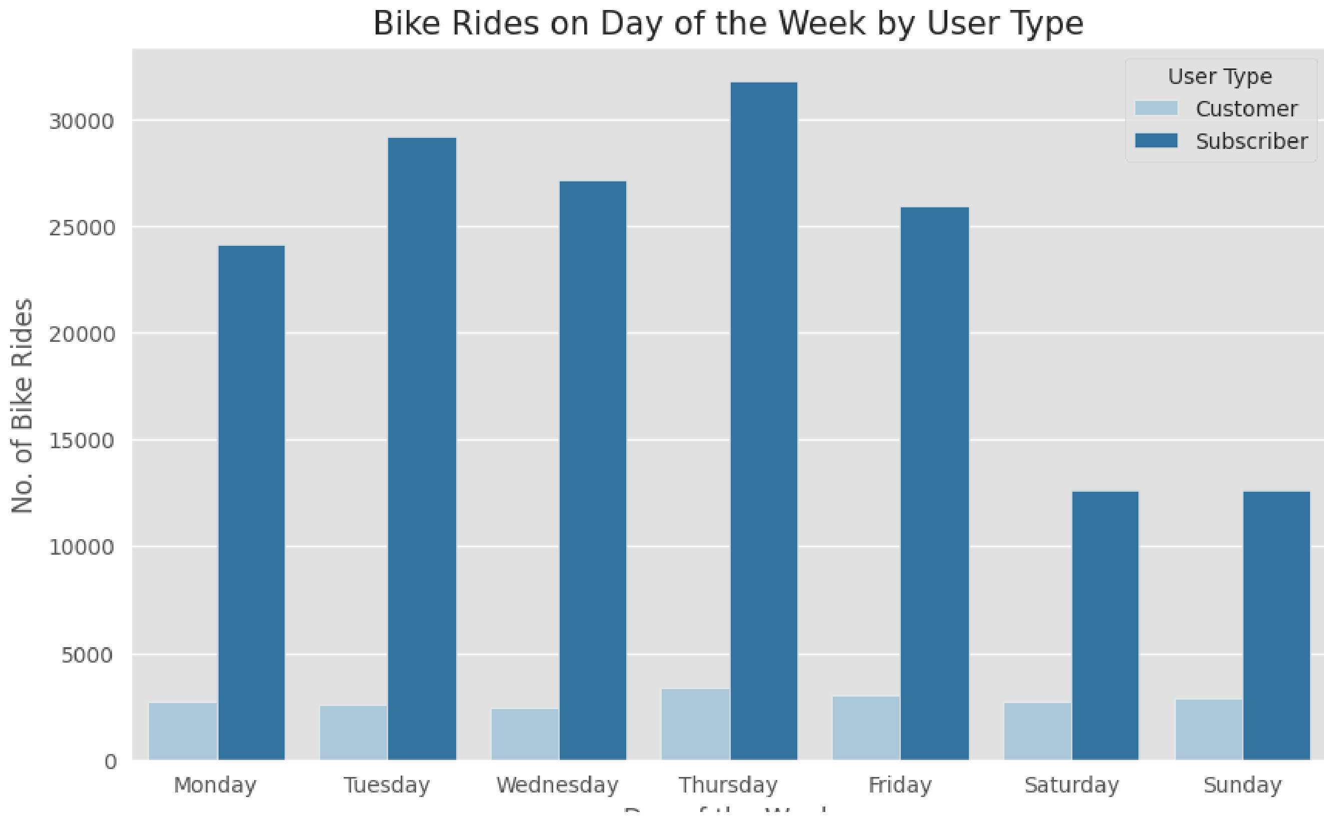
```
df.groupby('user_type')['dow'].value_counts()
```

		count
user_type	**dow**	
Customer	**Thursday**	3390
	Friday	3030
	Sunday	2896
	Monday	2741
	Saturday	2739
	Tuesday	2606
	Wednesday	2466
Subscriber	**Thursday**	31807
	Tuesday	29207
	Wednesday	27175
	Friday	25951
	Monday	24111
	Saturday	12666
	Sunday	12627

dtype: int64

Visualization #6. day of the week bar plot by user type

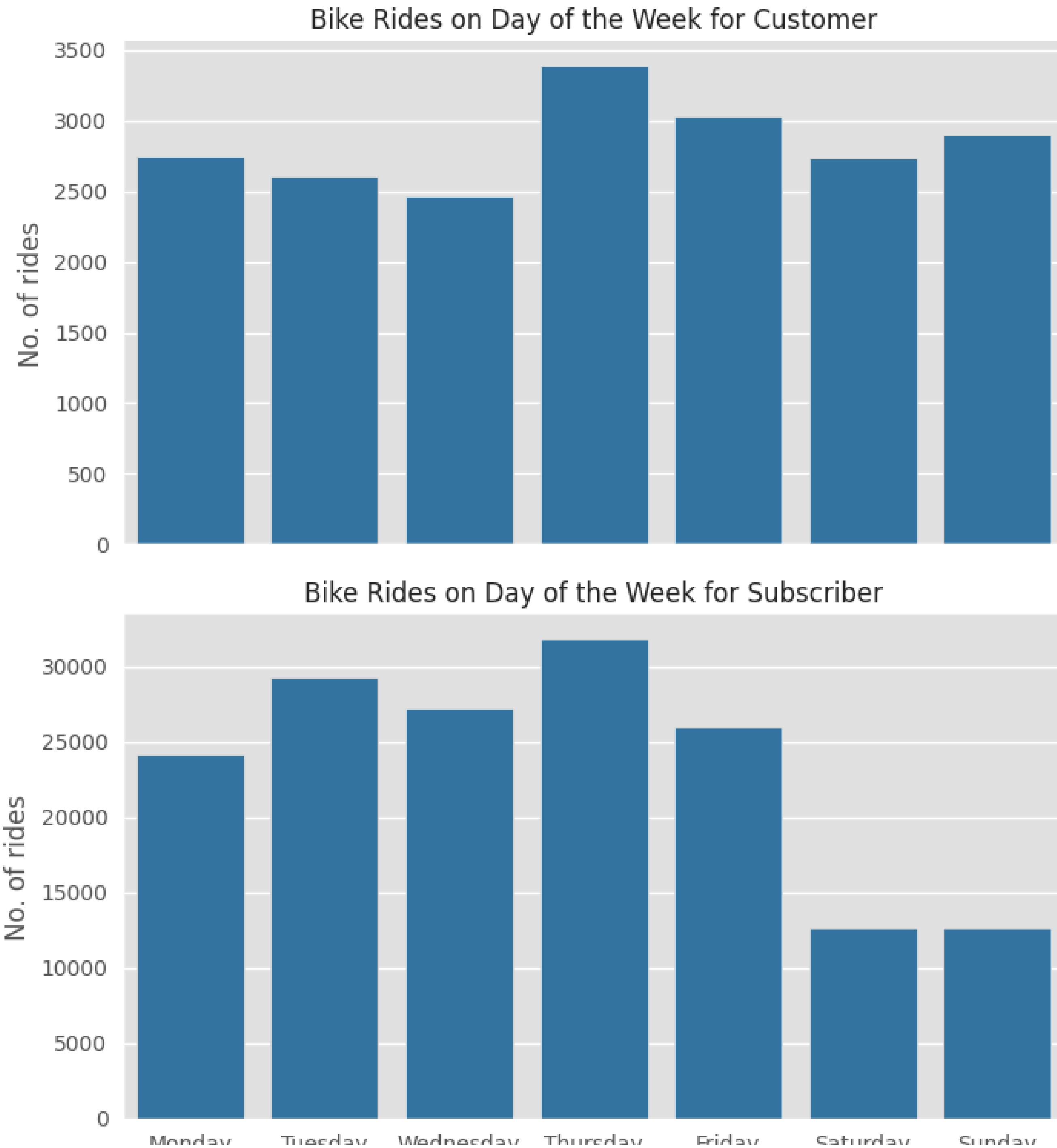
```
ordinal_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'
plt.figure(figsize=[10, 6])
sns.set_palette('Paired')
sns.countplot(data=df, x='dow', hue='user_type', order=ordinal_week)
plt.xlabel('Day of the Week', fontsize=12)
plt.ylabel('No. of Bike Rides', fontsize=12)
plt.title('Bike Rides on Day of the Week by User Type', fontsize=15)
plt.legend(title = 'User Type');
```



> It is a bit difficult to see the trend for customers due to the small total number of rides. I think two seperated plots will be better in this case.

```
g = sns.FacetGrid(df, row='user_type', height=4, aspect=1.8, sharey=False)
base_color = sns.color_palette()[1]
g.map(sns.countplot, 'dow', order=ordinal_week, color=base_color)
g.set_titles('Bike Rides on Day of the Week for {row_name}')
g.set_axis_labels(x_var="Day of the Week", y_var="No. of rides")
```

```
<seaborn.axisgrid.FacetGrid at 0x7c7cdbd25810>
```



Observation #6.

> Subscribers ride almost 10 times more than normal customers. The trend that more users ride bikes during workdays are quite clear for subscribers. However, the normal customers don't share the same trend. Users use shared bikes for commuting purpose are mostly bike subscribers.

Question #7. are there any difference in riding durations for male and female?

```
# df.info()
```

```
member_info = df[['user_type', 'duration_min', 'distance', 'member_gender', 'member_age']]
```

```
member_info.head()
```

	user_type	duration_min	distance	member_gender	member_age
0	Customer	869.75	0.54	Male	35.0
1	Customer	708.68	1.74	NaN	NaN
2	Customer	1030.90	2.70	Male	47.0
3	Subscriber	608.17	0.26	Other	30.0
4	Subscriber	26.42	2.41	Male	45.0

```
member_info.groupby('member_gender').agg({'duration_min':['size', np.mean]})
```

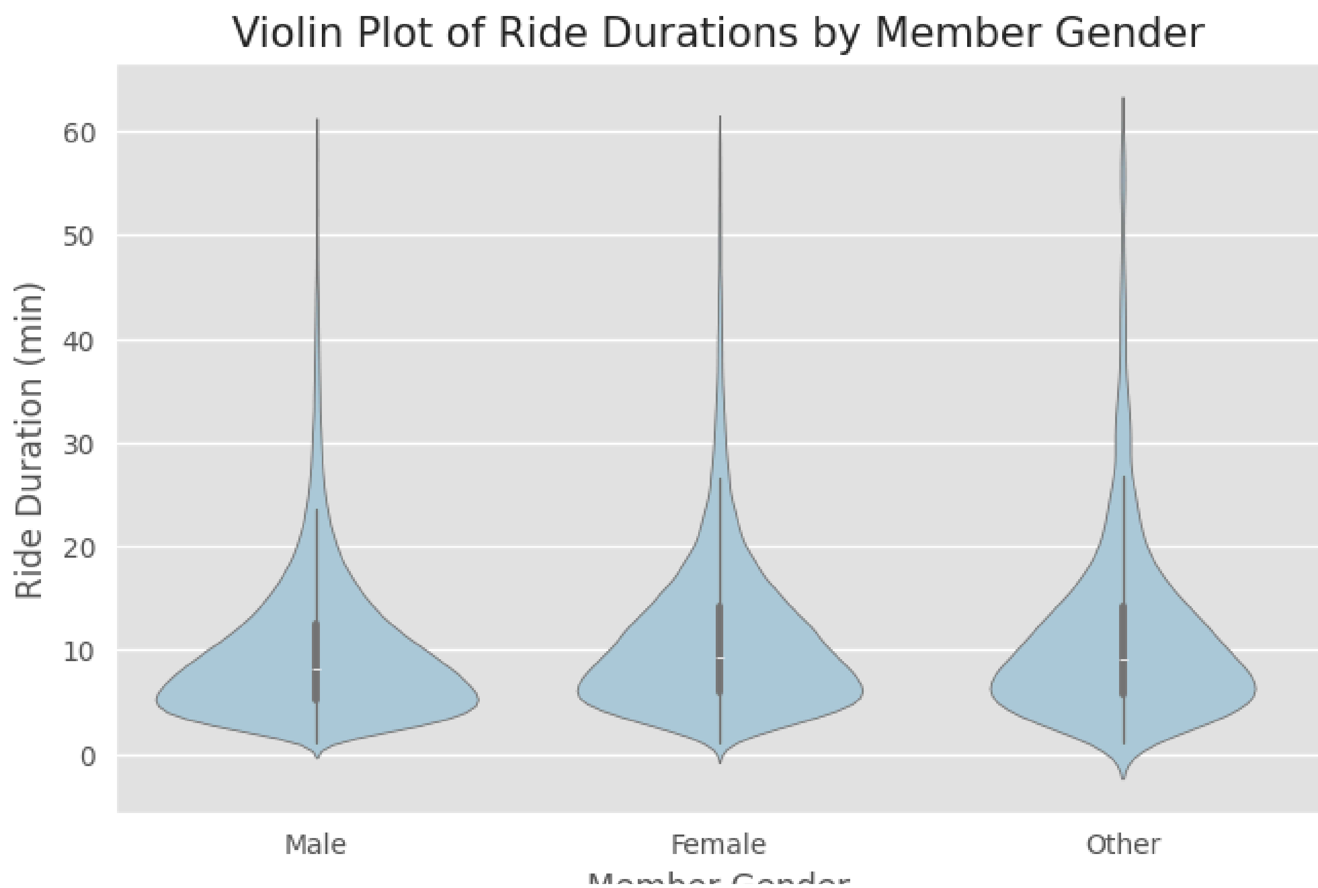
```
<ipython-input-64-990f0c72cddd>:1: FutureWarning: The provided callable <function mean a
  member_info.groupby('member_gender').agg({'duration_min':['size', np.mean]})
```

	duration_min	
	size	mean
member_gender		
Female	40844	12.984508
Male	130651	11.210645
Other	3652	16.608535

Visualization #7. violin plot and histogram of ride durations for different member genders

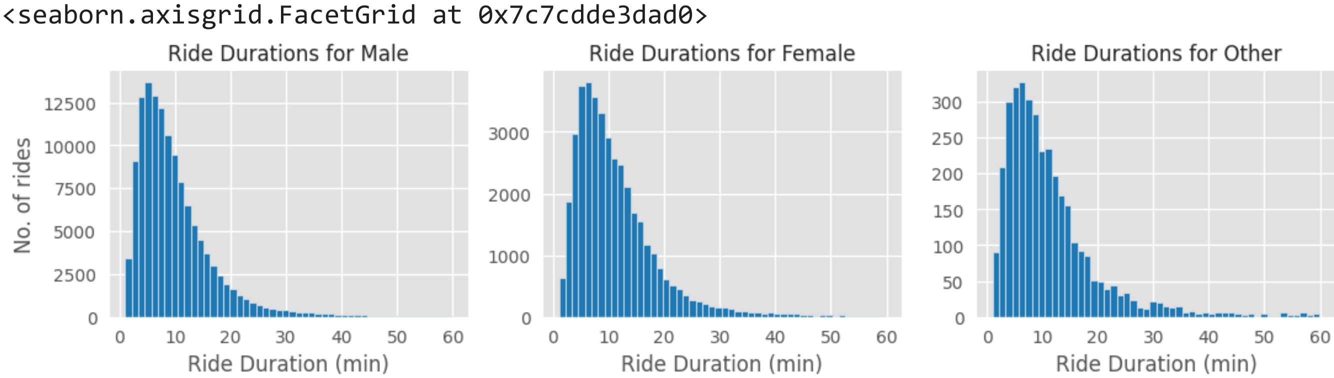
```
plt.figure(figsize=[8, 5])

base_color = sns.color_palette()[0]
sns.violinplot(data=member_info.loc[member_info['duration_min']<=60], x='member_gender',
               y='duration_min', color=base_color)
plt.xlabel('Member Gender', fontsize=12)
```

```
plt.ylabel('Ride Duration (min)', fontsize=12)
plt.title('Violin Plot of Ride Durations by Member Gender', fontsize=15);
```




```
g = sns.FacetGrid(member_info.loc[member_info['duration_min']<=60],
                  col='member_gender', height=3, aspect=1.2, sharey=False, col_wrap=3)
base_color = sns.color_palette()[1]
g.map(plt.hist, 'duration_min', color=base_color, bins=50)
g.set_titles('Ride Durations for {col_name}')
g.set_axis_labels(x_var='Ride Duration (min)', y_var='No. of rides')
```

```
<seaborn.axisgrid.FacetGrid at 0x7c7cdde3dad0>
```



Observation #7.

> The duration distribution (consider only rides within 60mins) for diffirent genders are in similar shapes. Male riders have more rides within 10mins, and their average riding

> duration is around 8mins. While female riders also have most of the rides within 10mins, they have more rides in 10-20mins range compared with male riders, and their average riding duration is around 10mins.

> Male riders took much more rides compared with female riders.

Question #8. are there any difference in riding durations for members indifferent age groups?

```
member_info['member_age'].describe()
```

	member_age
count	175147.000000
mean	34.193563
std	10.116689
min	18.000000
25%	27.000000
50%	32.000000
75%	39.000000
max	141.000000

dtype: float64

```
# there are some suspicious member age data
(df['member_age']>=90).sum()
```

```
77
```

```
# df.loc[df['member_birth_year']<=1920]
```

```
member_info.loc[member_info['member_age']<90].describe()
```

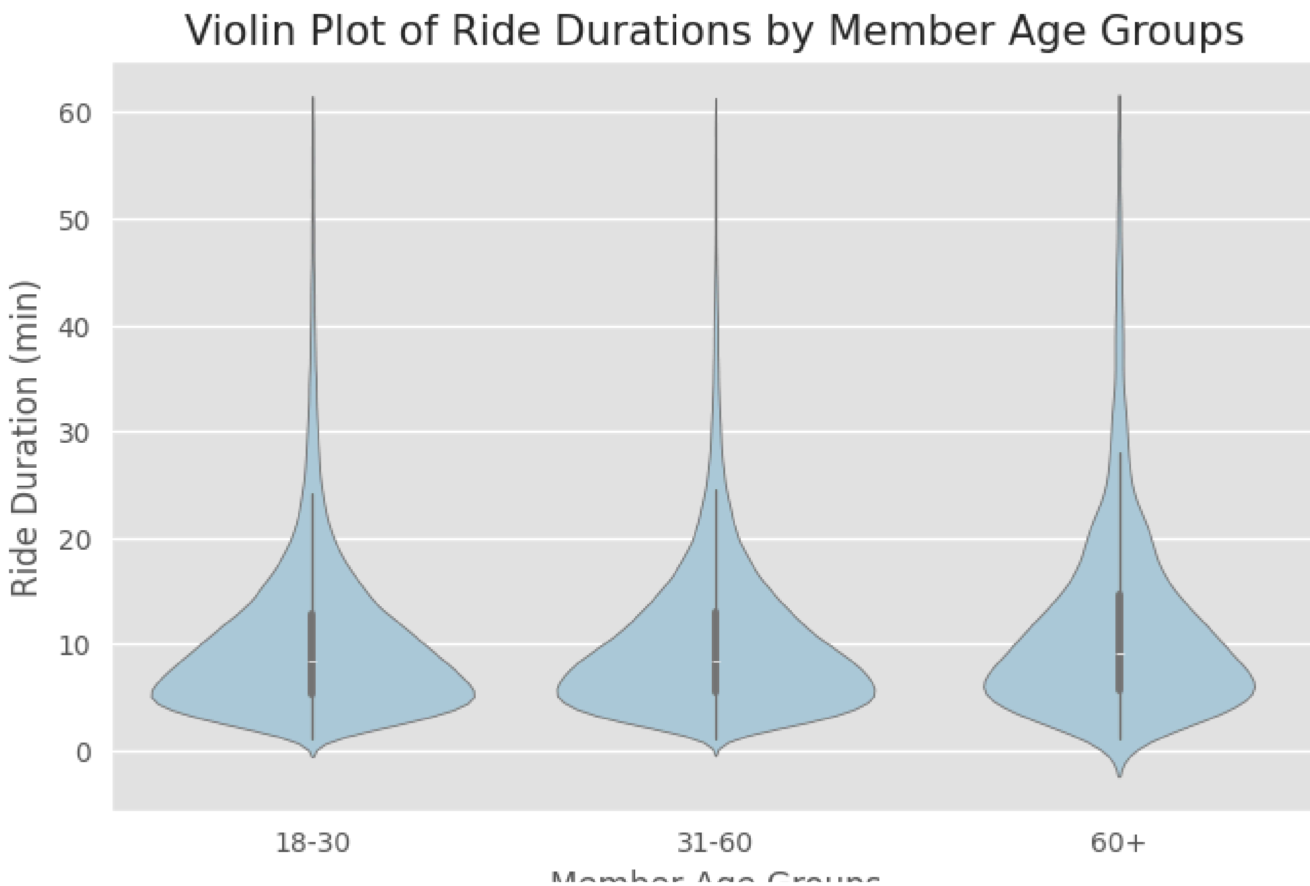
	duration_min	distance	member_age
count	175070.000000	175070.000000	175070.000000
mean	11.737280	1.688140	34.157011
std	27.365684	1.095992	9.966723
min	1.020000	0.000000	18.000000
25%	5.380000	0.910000	27.000000
50%	8.500000	1.430000	32.000000
75%	13.150000	2.220000	39.000000
max	1409.130000	69.430000	89.000000

```
# bin edges that will be used to define member age group
bin_edges = [18, 30, 60, 90]
# labels
bin_names = ['18-30', '31-60', '60+']
pd.options.mode.chained_assignment = None  # default='warn'
member_info['age_group'] = pd.cut(member_info['member_age'], bin_edges, labels=bin_names)
member_info.head()
```

	user_type	duration_min	distance	member_gender	member_age	age_group
0	Customer	869.75	0.54	Male	35.0	31-60
1	Customer	708.68	1.74	NaN	NaN	NaN
2	Customer	1030.90	2.70	Male	47.0	31-60
3	Subscriber	608.17	0.26	Other	30.0	18-30
4	Subscriber	26.42	2.41	Male	45.0	31-60

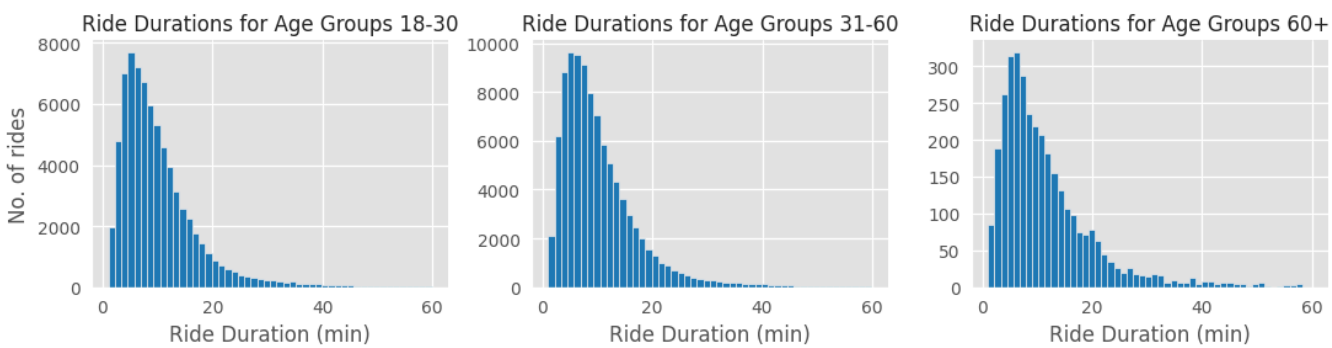
Visualization #8. violin plot and histogram of ride durations for different age groups

```
plt.figure(figsize=[8, 5])

base_color = sns.color_palette()[0]
sns.violinplot(data=member_info.loc[(member_info['member_age']<90) & (member_info['duration_
               x='age_group',
               y='duration_min',
               color=base_color)
plt.xlabel('Member Age Groups', fontsize=12)
plt.ylabel('Ride Duration (min)', fontsize=12)
plt.title('Violin Plot of Ride Durations by Member Age Groups', fontsize=15);
```




```
g = sns.FacetGrid(member_info.loc[(member_info['member_age']<90) & (member_info['duration_mi
                  col='age_group', height=3, aspect=1.2, sharey=False, col_wrap=3)
base_color = sns.color_palette()[1]
g.map(plt.hist, 'duration_min', color=base_color, bins=50)
g.set_titles('Ride Durations for Age Groups {col_name}')
g.set_axis_labels(x_var='Ride Duration (min)', y_var='No. of rides')
```

<seaborn.axisgrid.FacetGrid at 0x7c7cdb972ad0>



Observation #8.

> All three age groups have an average ride duration of 10mins. Compared with 60+ riders, members in 18 to 60 age groups have more rides within 10mins.

> The most rides are taken by members in 31 to 60 age group, rides for the elders are less. The ride durations distribution for all age groups are very much the same, with one peak poin at around 10mins, and then gradualy decreased with a long tail.

Question #9. what's the difference between customer and subscriber in terms of riding distance?

```
df.groupby('user_type')['distance'].describe()
```

	count	mean	std	min	25%	50%	75%	max
user_type								
Customer	19868.0	1.866098	1.169476	0.0	1.04	1.68	2.53	13.58
Subscriber	163544.0	1.668180	1.085817	0.0	0.90	1.41	2.18	69.43

```
df.loc[(df.user_type=='Customer') & (df.distance==0)]
```

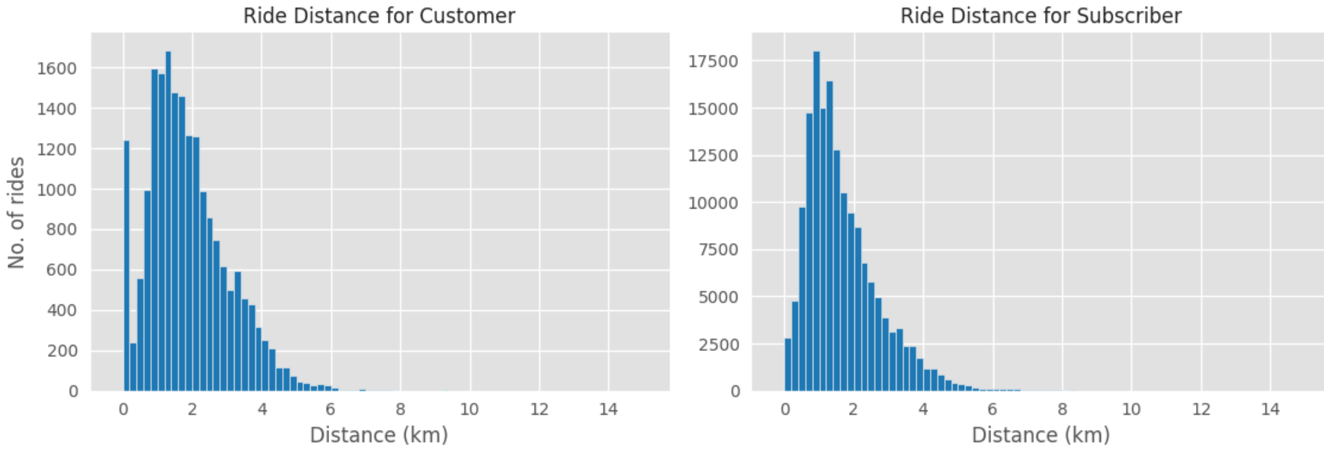
	duration_sec	start_time	end_time	start_station_id	start_station_name	sta
19	874	2019-02-28 23:43:05.183	2019-02-28 23:57:39.796	180.0	Telegraph Ave at 23rd St	
53	3418	2019-02-28 22:41:16.362	2019-02-28 23:38:14.363	11.0	Davis St at Jackson St	
1197	3200	2019-02-28 18:46:44.597	2019-02-28 19:40:04.775	186.0	Lakeside Dr at 14th St	
1198	3175	2019-02-28 18:47:04.953	2019-02-28 19:40:00.440	186.0	Lakeside Dr at 14th St	
1541	4222	2019-02-28 17:56:48.285	2019-02-28 19:07:10.497	381.0	20th St at Dolores St	
...	...	...	...	...	...	
180366	1778	2019-02-01 10:56:08.587	2019-02-01 11:25:47.311	377.0	Fell St at Stanyan St	
180523	2121	2019-02-01 10:21:35.740	2019-02-01 10:56:57.049	200.0	2nd Ave at E 18th St	
180709	901	2019-02-01 10:08:17.199	2019-02-01 10:23:18.421	370.0	Jones St at Post St	
180710	878	2019-02-01 10:08:37.063	2019-02-01 10:23:15.278	370.0	Jones St at Post St	
180848	2844	2019-02-01 09:17:14.446	2019-02-01 10:04:38.811	186.0	Lakeside Dr at 14th St	

1225 rows × 21 columns

Visualization #9. histogram of ride distance for different user types

```
g = sns.FacetGrid(df.loc[df['distance']<=15], col='user_type', height=4, aspect=1.4, sharey=
base_color = sns.color_palette()[1]

binsize = 0.2
bins = np.arange(-0.2, 15+binsize, binsize)
g.map(plt.hist, 'distance', bins=bins, color=base_color)
g.set_titles('Ride Distance for {col_name}')
g.set_axis_labels(x_var='Distance (km)', y_var="No. of rides")
```

```
<seaborn.axisgrid.FacetGrid at 0x7c7cdbd05a10>
```



Observation #9.

> The ride distance for customers have two peaks. The first one is zero kilometer, meaning the users return bikes to the start stations of the trip. The second one is around 2km, which is also the peak point for subscriber users.

> For subscribers, not as many subscribers will return bikes to their start station, echoing with the fact that they use bikes maninly for commuting.

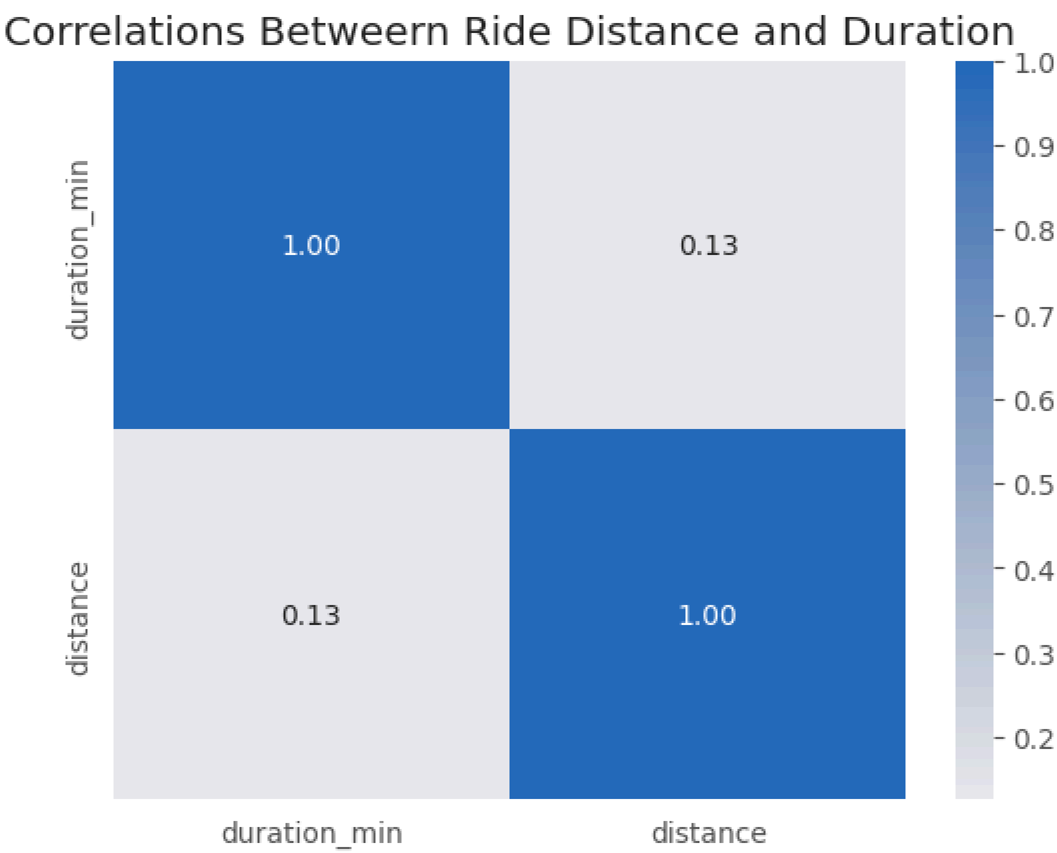
Question #10. what's the relation for distance and duration?

```
df[['duration_min','distance']].describe()
```

	duration_min	distance
count	183412.000000	183412.000000
mean	12.101301	1.689620
std	29.906501	1.096911
min	1.020000	0.000000
25%	5.420000	0.910000
50%	8.570000	1.430000
75%	13.270000	2.220000
max	1424.070000	69.430000

Visualization #10. plot of ride distance and duration

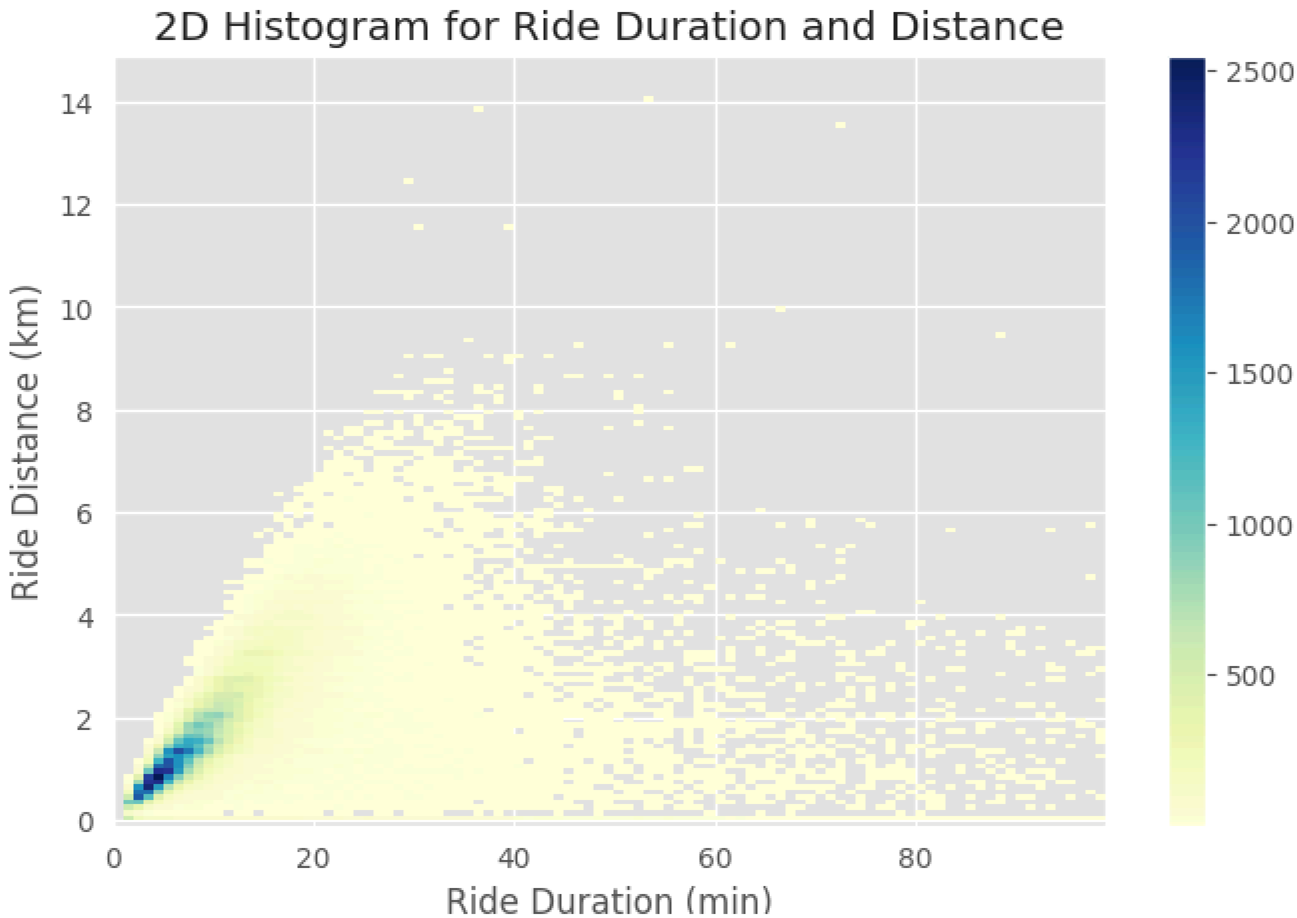
```
# the correlation heatmap
sns.heatmap(df[['duration_min','distance']].corr(),
            annot = True,
            fmt = '.2f',
            cmap = 'vlag_r',
            center = 0)
plt.title('Correlations Betweern Ride Distance and Duration');
```



> The correlation between distance and duration are positive but small (0.13), not suggesting a strong relation between the two variables.

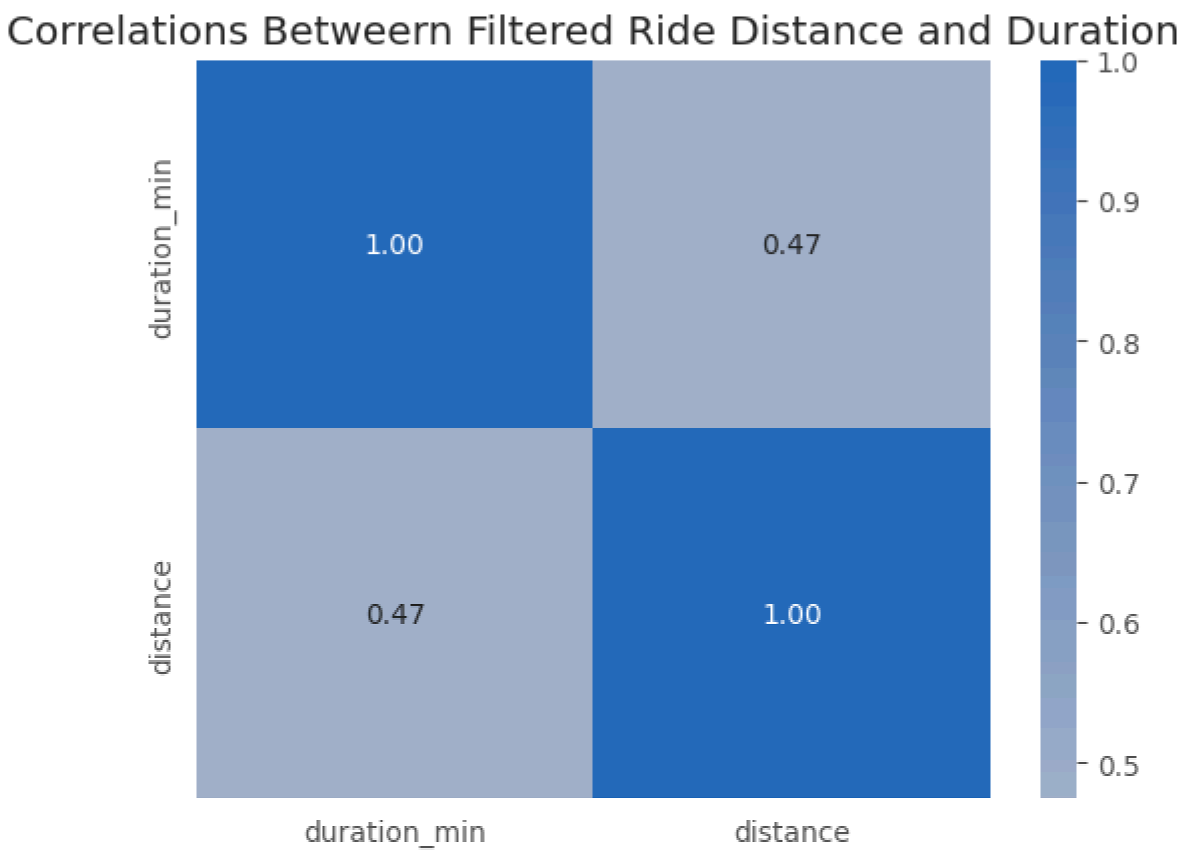
```
plt.figure(figsize=[8,5])

df_dd = df.query('distance < 15 and duration_min < 100')
bins_dur = np.arange(0, 100, 1)
bins_dis = np.arange(-0.1, 15, 0.1)
plt.hist2d(data = df_dd, x = 'duration_min', y = 'distance',
           bins = [bins_dur, bins_dis],
           cmap='YlGnBu', cmin = 0.3)
plt.colorbar();
plt.xlabel('Ride Duration (min)')
plt.ylabel('Ride Distance (km)')
plt.title('2D Histogram for Ride Duration and Distance')
```

```
Text(0.5, 1.0, '2D Histogram for Ride Duration and Distance')
```



> After I filtered out the outliers, from the 2D histogram it is clear that there is a strong relation for duration and distance when duration is within 20mins and distance is within 3km.

```
# the correlation heatmap for filtered duration and distance
sns.heatmap(df_dd[['duration_min','distance']].corr(),
            annot = True,
            fmt = '.2f',
            cmap = 'vlag_r',
            center = 0)
plt.title('Correlations Betweern Filtered Ride Distance and Duration');
```



> The correlation for distance and duration is 0.47 for durations less than 100mins and distance less than 15km, that is a stronger positive relation compared with that without filters.

Observation #10.

> The correlation for duration and distance is small if no filter is applied to the distribution. From the 2D histogram map, I noticed a strong positive linear relation between the two variables when their values are within 20mins and 3km respectively.

> After applying filters to limit maximum duration to 100mins and maximum distance to 15km, the correlation between these two variables increased from 0.13 to 0.47.

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?

> I noticed riding durations and distance have a strong positive relation when duration is within 20mins and distance is within 3km. When duration and distance increase, this relation is affected largely by outliers, so the correlation coefficient is decreased.

> The subscribers use shared bikes for commuting purpose, therefore there is a clear increasing of bike rides during weekdays, while the normal customers don't share the same time pattern.

> There is no huge differences of riding durations for users in different gender and age groups.

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

> I noticed that there is a large portion of the rides completed by normal customers have the same start and end stations.

Multivariate Exploration

Question #11. how do customers and cubsribers ride bikes throught the days and weeks?

```
df_temporal = df.groupby(['user_type', 'dow', 'hour'])['bike_id'].size().to_frame
```

```
ordinal_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday
df_temporal['dow'] = pd.Categorical(df_temporal['dow'],
                                    categories=ordinal_week,
                                    ordered=True)
```

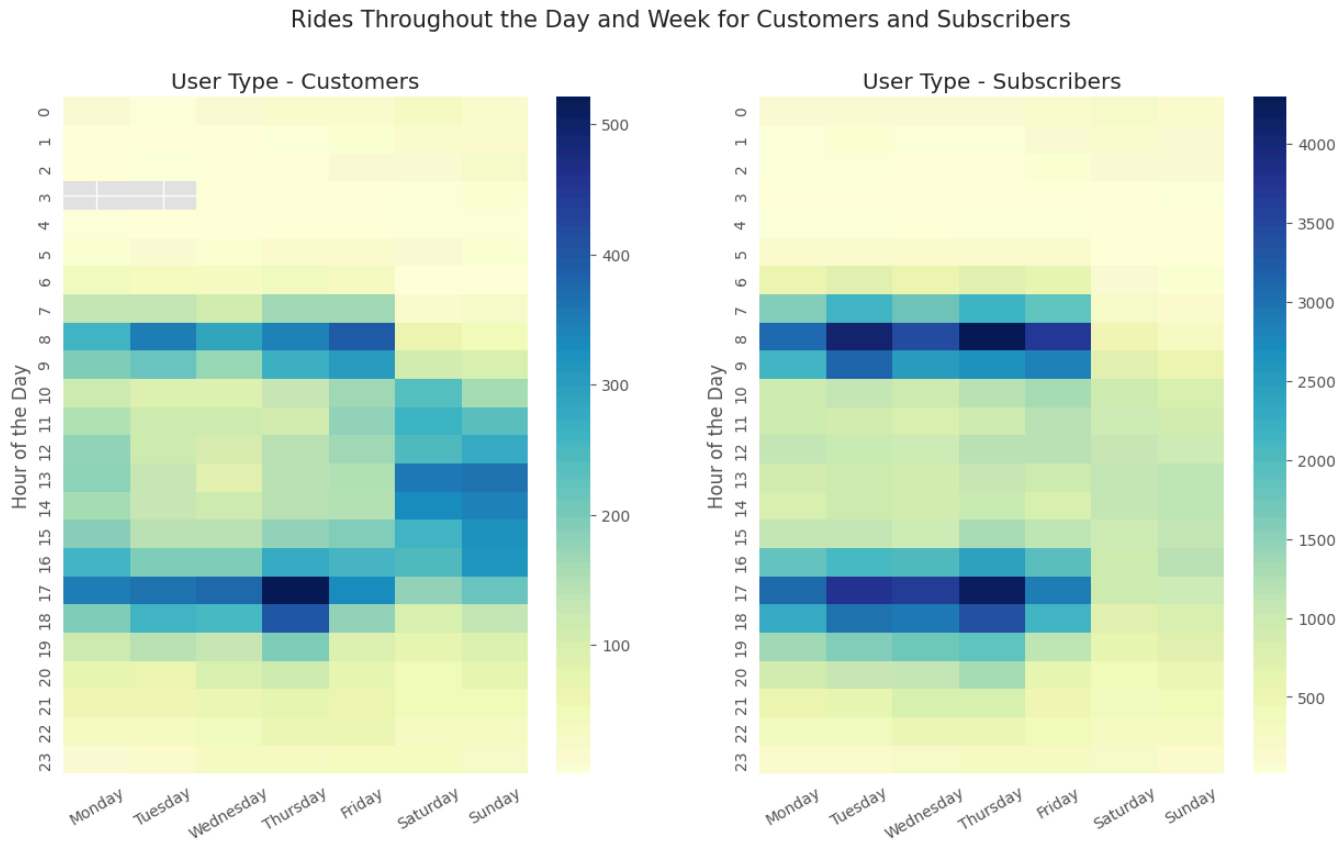
```
df_temporal.head()
```

	user_type	dow	hour	bike_id
0	Customer	Friday	0	15
1	Customer	Friday	1	7
2	Customer	Friday	2	9
3	Customer	Friday	3	2
4	Customer	Friday	4	2

Next steps: Generate code with df_temporal View recommended plots New interactive sheet

Visualization #11. heatmap for rides during the day and weeks for customers and subscribers

```
import matplotlib.pyplot as plt
import seaborn as sns

# Creating subplots for customer and subscriber ride heatmaps
fig, ax = plt.subplots(ncols=2, figsize=(15, 8))

# Heatmap for Customers
sns.heatmap(data=df_temporal.loc[df_temporal.user_type == 'Customer']
                     .pivot(index='hour', columns='dow', values='bike_id'),
            cmap='YlGnBu', ax=ax[0])

# Heatmap for Subscribers
sns.heatmap(data=df_temporal.loc[df_temporal.user_type == 'Subscriber']
                     .pivot(index='hour', columns='dow', values='bike_id'),
            cmap='YlGnBu', ax=ax[1])

# Labels and titles
ax[0].set_xlabel('Day of the Week')
ax[1].set_xlabel('Day of the Week')
ax[0].set_ylabel('Hour of the Day')
ax[1].set_ylabel('Hour of the Day')
ax[0].set_title('User Type - Customers')
ax[1].set_title('User Type - Subscribers')

# Rotate x-axis ticks
ax[0].tick_params(axis='x', rotation=30)
ax[1].tick_params(axis='x', rotation=30)

# Figure title
fig.suptitle('Rides Throughout the Day and Week for Customers and Subscribers', fontsize=15)

plt.show()
```



Observation #11.

> The subscribers mainly use bikes for commuting, there are few rides during off-peak hours on weekdays and the whole day during weekends.

> Normal customers use bikes for commuting purposes as well, more of the ride during off-peak hours on weekdays compared with subscribers, while during weekends, there are quite a lot of customers ride shared bikes.

Question #12. how's the riding distance and durations related for users in different user groups?

```
df_user = member_info.groupby(['member_gender', 'age_group'])[['duration_min', 'distance']].
```

```
<ipython-input-86-4d152218f2d2>:1: FutureWarning: The default of observed=False is depre
  df_user = member_info.groupby(['member_gender', 'age_group'])[['duration_min', 'distar
```

```
df_user
```

member_gender	age_group	duration_min	distance
Female	**18-30**	13.239727	1.716540
	31-60	12.723427	1.810560
	60+	13.691276	1.621193
Male	**18-30**	11.045689	1.600511
	31-60	11.288922	1.710476
	60+	12.111504	1.550469
Other	**18-30**	23.096714	1.812994
	31-60	13.374790	1.779480
	60+	16.670833	2.060625

Next steps: Generate code with df_user | View recommended plots | New interactive sheet

```
member_dd = member_info.query('distance < 5 and duration_min < 10')
```

```
member_dd.head()
```

	user_type	duration_min	distance	member_gender	member_age	age_group
10	Subscriber	7.63	0.98	Female	23.0	18-30
11	Subscriber	8.43	1.61	Male	26.0	18-30
14	Subscriber	6.58	1.21	Male	31.0	31-60
15	Subscriber	3.47	0.79	Male	26.0	18-30
16	Subscriber	9.13	2.32	Male	38.0	31-60

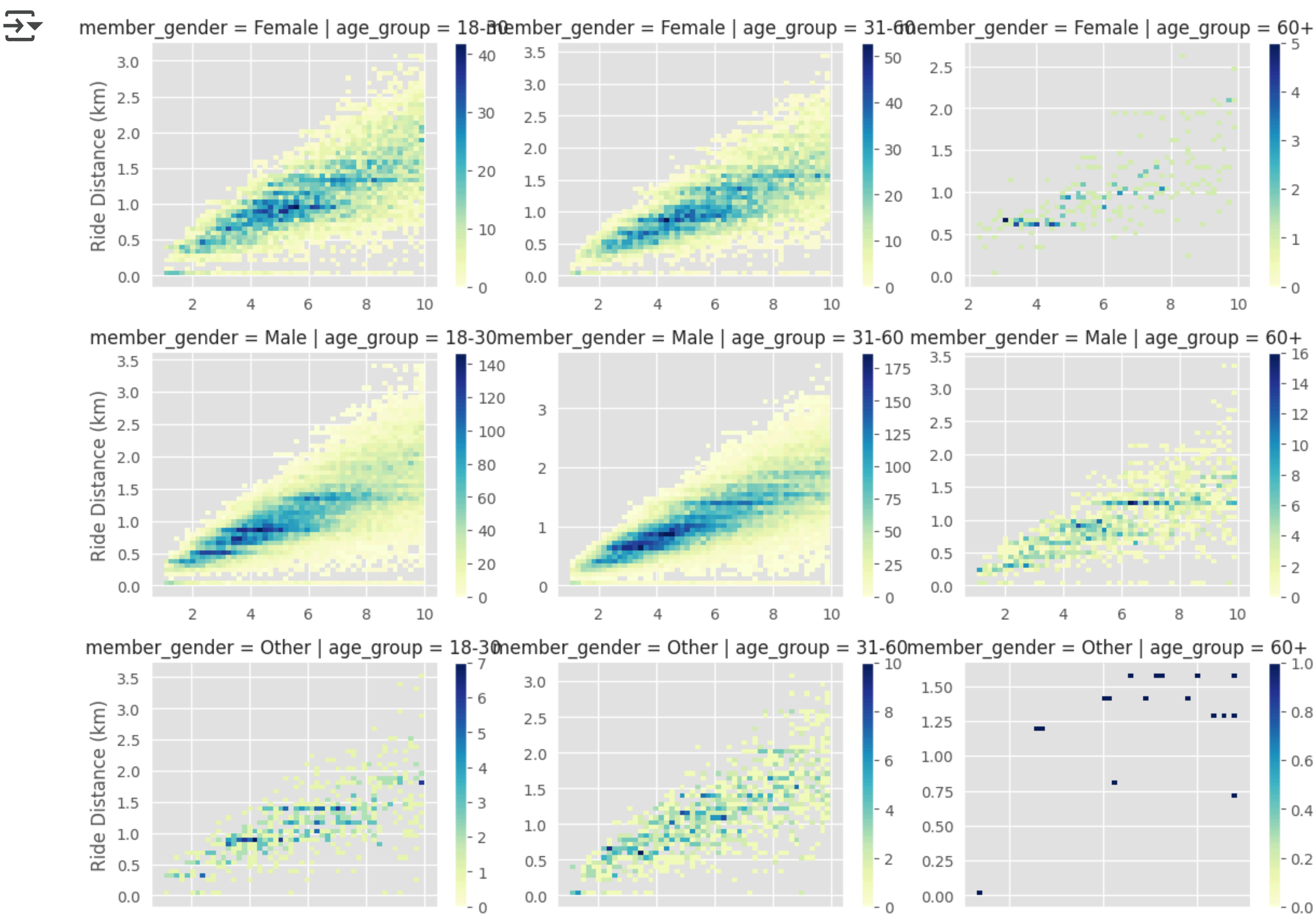
```
member_dd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 108471 entries, 10 to 183411
Data columns (total 6 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   user_type      108471 non-null  object
 1   duration_min   108471 non-null  float64
 2   distance       108471 non-null  float64
 3   member_gender  104399 non-null  object
 4   member_age     104399 non-null  float64
 5   age_group      104348 non-null  category
```

```
dtypes: category(1), float64(3), object(2)
memory usage: 5.1+ MB
```

Visualization #12. histogram of distance and duration for different user groups

```
g = sns.FacetGrid(data=member_dd, col='age_group', row='member_gender',
                  height=3, aspect=1.3, sharey=False, sharex=False)
# g.map_dataframe(sns.scatterplot, x='duration_min', y='distance')
# bins_dur = np.arange(0, 50, 1)
# bins_dis = np.arange(-0.1, 10, 0.1)
g.map_dataframe(sns.histplot, x='duration_min', y='distance',
#                  bins = [bins_dur, bins_dis],
                bins=50,
                cmap='YlGnBu', cbar=True)

g.set_xlabels('Ride Duration (min)')
g.set_ylabels('Ride Distance (km)');
```



Observation #12.

> For all the user groups except the 'Other' gender and age 60+ group, there is a positive linear relation for riding distance and duration.

> When users are female or male and in age groups 18 to 30 or 31 to 60, the relation for riding distance and duration are quite clear. There are less rides for users whose gender labeled as 'Other' or users age larger than 60, therefore, for these groups, the linear relation is not as clear as other groups.

Question #13. which is the most busiest station at different time of the day and day of the week?

```
df.hour.describe()
```

	hour
count	183412.000000
mean	13.458421
std	4.724978
min	0.000000
25%	9.000000
50%	14.000000
75%	17.000000
max	23.000000

dtype: float64

```
# bin edges that will be used to define member age group
bin_edges = [-1, 12, 18, 23]
# labels
bin_names = ['Morning', 'Afternoon', 'Evening']
pd.options.mode.chained_assignment = None  # default='warn'
df['hour_label'] = pd.cut(df['hour'], bin_edges, labels=bin_names)
df.head()
```

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_st
0	52185	2019-02-28 17:32:10.145	2019-03-01 08:01:55.975	21.0	Montgomery St BART Station (Market St at 2nd St)	
1	42521	2019-02-28 18:53:21.789	2019-03-01 06:42:03.056	23.0	The Embarcadero at Steuart St	
2	61854	2019-02-28 12:13:13.218	2019-03-01 05:24:08.146	86.0	Market St at Dolores St	

```
station_df = df.groupby(['dow', 'hour_label', 'start_station_name']).size()
```

```
<ipython-input-94-d49a1e5a4d48>:1: FutureWarning: The default of observed=False is depre
  station_df = df.groupby(['dow', 'hour_label', 'start_station_name']).size()
```

5 rows × 22 columns

```
station_df.unstack(level=[0,1]).loc[start_order,:]
```

dow	Friday			Monday			Saturday	
hour_label	**Morning**	**Afternoon**	**Evening**	**Morning**	**Afternoon**	**Evening**	**Morning**	**A**
start_station_name								
Market St at 10th St	352	282	81	211	241	78	129	
San Francisco Caltrain Station 2	[illegible]	117	[illegible]	[illegible]	[illegible]	[illegible]	51	