Part_I_Ford_GoBike_System

November 13, 2022

1 Part I - Ford GoBike System Data Analysis

1.1 by Sandra Kamikazi

1.2 Introduction

Dataset Description The larger San Francisco Bay area's bike-sharing program's data collection includes information on each ride. The Ford GoBike was first released in the San Francisco Bay Area in 2013 as Bay Area Bike Share, and it was then revived in 2017 as the Ford GoBike. Additionally, the name Bay Wheels has been used for the system since June 2019. The dataset contains the three Bay Wheels pricing tiers, so you must know them. In addition, two payment choices are available: paying per trip as a non-user or paying monthly, yearly, or for Bike Share for All as a subscriber.

You can find the dataset here

The Dataset has 16 columns which are listed below:

- Duration_sec
- Start_time
- End_time
- Start_station_ID
- Start_station_name
- Start_station_latitude
- Start_tation_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

Preliminary Wrangling

```
In [1]: # 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 sb
import datetime as dt
```

In [2]:
```
#magic word to help in plotting the visualization
%matplotlib inline
```

1.2.1 1. Assessing Data Ford GoBike System Data

Visual Assessment

In [3]:
```
#loading dataset
df = pd.read_csv('201902-fordgobike-tripdata.csv')
df.head()
```

Out[3]:
```
   duration_sec                start_time                  end_time  \
0         52185  2019-02-28 17:32:10.1450  2019-03-01 08:01:55.9750
1         42521  2019-02-28 18:53:21.7890  2019-03-01 06:42:03.0560
2         61854  2019-02-28 12:13:13.2180  2019-03-01 05:24:08.1460
3         36490  2019-02-28 17:54:26.0100  2019-03-01 04:02:36.8420
4          1585  2019-02-28 23:54:18.5490  2019-03-01 00:20:44.0740

   start_station_id                                start_station_name  \
0              21.0  Montgomery St BART Station (Market St at 2nd St)
1              23.0                     The Embarcadero at Steuart St
2              86.0                           Market St at Dolores St
3             375.0                           Grove St at Masonic Ave
4               7.0                               Frank H Ogawa Plaza

   start_station_latitude  start_station_longitude  end_station_id  \
0               37.789625              -122.400811            13.0
1               37.791464              -122.391034            81.0
2               37.769305              -122.426826             3.0
3               37.774836              -122.446546            70.0
4               37.804562              -122.271738           222.0

                               end_station_name  end_station_latitude  \
0                Commercial St at Montgomery St             37.794231
1                            Berry St at 4th St             37.775880
2  Powell St BART Station (Market St at 4th St)             37.786375
3                        Central Ave at Fell St             37.773311
4                         10th Ave at E 15th St             37.792714

   end_station_longitude  bike_id   user_type  member_birth_year  \
0            -122.402923     4902    Customer             1984.0
1            -122.393170     2535    Customer                NaN
2            -122.404904     5905    Customer             1972.0
3            -122.444293     6638  Subscriber             1989.0
```

```
4           -122.248780     4898   Subscriber             1974.0

  member_gender bike_share_for_all_trip
0          Male                      No
1           NaN                      No
2          Male                      No
3         Other                      No
4          Male                     Yes
```

```
In [4]: #check the size of our dataset
        df.shape
```

```
Out[4]: (183412, 16)
```

As we see, our dataset has 16 columns.

```
In [5]: #print columns names
        list(df.columns)
```

```
Out[5]: ['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']
```

Programmatic Assessement Create a function that can help me in Programmatic assessement without repetition

```
In [6]: #extract dataset information
        def information(df):
            print(" Our dataset has the following number of Columns and Rows",df.shape)
            print("\n")
            print("We have the following columns",df.columns)
            print("\n")
            print("My attributes have the following data types ",df.dtypes)
            print("\n")
            print("Here is the brief summary of my dataset", df.info())
            print("\n")
```

```
    print("My dataset have the following number of following attributes",df.nunique())
    print("\n")
    print("My dataset have the following missing values",df.isnull().sum())
    print("\n")
    print("My dataset have this number of duplicates",sum(df.duplicated()))
```

In [7]:
```
#describing df
df.describe()
```

Out[7]:

	duration_sec	start_station_id	start_station_latitude
count	183412.000000	183215.000000	183412.000000
mean	726.078435	138.590427	37.771223
std	1794.389780	111.778864	0.099581
min	61.000000	3.000000	37.317298
25%	325.000000	47.000000	37.770083
50%	514.000000	104.000000	37.780760
75%	796.000000	239.000000	37.797280
max	85444.000000	398.000000	37.880222

	start_station_longitude	end_station_id	end_station_latitude
count	183412.000000	183215.000000	183412.000000
mean	-122.352664	136.249123	37.771427
std	0.117097	111.515131	0.099490
min	-122.453704	3.000000	37.317298
25%	-122.412408	44.000000	37.770407
50%	-122.398285	100.000000	37.781010
75%	-122.286533	235.000000	37.797320
max	-121.874119	398.000000	37.880222

	end_station_longitude	bike_id	member_birth_year
count	183412.000000	183412.000000	175147.000000
mean	-122.352250	4472.906375	1984.806437
std	0.116673	1664.383394	10.116689
min	-122.453704	11.000000	1878.000000
25%	-122.411726	3777.000000	1980.000000
50%	-122.398279	4958.000000	1987.000000
75%	-122.288045	5502.000000	1992.000000
max	-121.874119	6645.000000	2001.000000

In [8]:
```
# Read information
information(df)
```

```
 Our dataset has the following number of Columns and Rows (183412, 16)


We have the following columns Index(['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'],
      dtype='object')
```

```
My attributes have the following data types  duration_sec                 int64
start_time                  object
end_time                    object
start_station_id           float64
start_station_name          object
start_station_latitude     float64
start_station_longitude    float64
end_station_id             float64
end_station_name            object
end_station_latitude       float64
end_station_longitude      float64
bike_id                      int64
user_type                   object
member_birth_year          float64
member_gender               object
bike_share_for_all_trip     object
dtype: object
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 16 columns):
duration_sec               183412 non-null int64
start_time                 183412 non-null object
end_time                   183412 non-null object
start_station_id           183215 non-null float64
start_station_name         183215 non-null object
start_station_latitude     183412 non-null float64
start_station_longitude    183412 non-null float64
end_station_id             183215 non-null float64
end_station_name           183215 non-null object
end_station_latitude       183412 non-null float64
end_station_longitude      183412 non-null float64
bike_id                    183412 non-null int64
user_type                  183412 non-null object
member_birth_year          175147 non-null float64
member_gender              175147 non-null object
bike_share_for_all_trip    183412 non-null object
dtypes: float64(7), int64(2), object(7)
memory usage: 22.4+ MB
Here is the brief summary of my dataset None
```

```
My dataset have the following number of following attributes duration_sec                 4752
```

```
start_time                183401
end_time                  183397
start_station_id             329
start_station_name           329
start_station_latitude       334
start_station_longitude      335
end_station_id               329
end_station_name             329
end_station_latitude         335
end_station_longitude        335
bike_id                     4646
user_type                      2
member_birth_year             75
member_gender                  3
bike_share_for_all_trip        2
dtype: int64


My dataset have the following missing values 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


My dataset have this number of duplicates 0
```

The general summary of our dataset

Our dataset has 16 columns and 183412 rows. The attributes have different data types, like objects, floats, and int. Some features have small missing values, and others have many missing values, like member birth year and member gender. Finally, our dataset doesn't have duplicates.

1.2.2 Quality issues

- Missing Values for some attributes like member birth year, member gender, etc

- Some features have datatypes that are difficult to analyze.

1.2.3 Tidiness Issues

- The names of the days the bike was rented and returned are not displayed
- Underserved columns like start_station_latitude, end_station_longitude, etc

1.3 Assessing Data Conclusion

I was able to detect and document at least eight (2) quality issues and two (2) tidiness issue using both visual assessment and programmatic assessement.

1.3.1 Cleaning Data

In [9]:
```
# Make copies of original piece of data
df1 = df.copy()
```

In [10]:
```
df1.head()
```

Out[10]:
```
   duration_sec                start_time                  end_time  \
0         52185  2019-02-28 17:32:10.1450  2019-03-01 08:01:55.9750
1         42521  2019-02-28 18:53:21.7890  2019-03-01 06:42:03.0560
2         61854  2019-02-28 12:13:13.2180  2019-03-01 05:24:08.1460
3         36490  2019-02-28 17:54:26.0100  2019-03-01 04:02:36.8420
4          1585  2019-02-28 23:54:18.5490  2019-03-01 00:20:44.0740

   start_station_id                                start_station_name  \
0              21.0  Montgomery St BART Station (Market St at 2nd St)
1              23.0                     The Embarcadero at Steuart St
2              86.0                           Market St at Dolores St
3             375.0                           Grove St at Masonic Ave
4               7.0                               Frank H Ogawa Plaza

   start_station_latitude  start_station_longitude  end_station_id  \
0               37.789625              -122.400811            13.0
1               37.791464              -122.391034            81.0
2               37.769305              -122.426826             3.0
3               37.774836              -122.446546            70.0
4               37.804562              -122.271738           222.0

                               end_station_name  end_station_latitude  \
0                Commercial St at Montgomery St             37.794231
1                            Berry St at 4th St             37.775880
2  Powell St BART Station (Market St at 4th St)             37.786375
3                        Central Ave at Fell St             37.773311
4                            10th Ave at E 15th St             37.792714

   end_station_longitude  bike_id   user_type  member_birth_year  \
0            -122.402923     4902    Customer             1984.0
```

```
1             -122.393170     2535    Customer                   NaN
2             -122.404904     5905    Customer                1972.0
3             -122.444293     6638  Subscriber                1989.0
4             -122.248780     4898  Subscriber                1974.0

  member_gender bike_share_for_all_trip
0          Male                      No
1           NaN                      No
2          Male                      No
3         Other                      No
4          Male                     Yes
```

1.3.2 Issue #1:

Missing Values for some attributes like member birth year, member gender, etc

Define: Delete all empty rows in our table

Code

In [11]:
```
df1.dropna(inplace=True)
```

Test

In [12]:
```
df1.isna().sum()
```

Out[12]:
```
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
```

1.3.3 Issue #2:

Some features have datatypes that are difficult to analyze

Define: Change to columns to its appropriate datatypes

Code

```
In [13]: # Change from object to dataetime datatype
         df1['start_time'] = pd.to_datetime(df1['start_time'])
         df1['end_time'] = pd.to_datetime(df1['end_time'])
```

Test

```
In [14]: df1.dtypes
```

```
Out[14]: duration_sec                        int64
         start_time                 datetime64[ns]
         end_time                   datetime64[ns]
         start_station_id                  float64
         start_station_name                 object
         start_station_latitude            float64
         start_station_longitude           float64
         end_station_id                    float64
         end_station_name                   object
         end_station_latitude              float64
         end_station_longitude             float64
         bike_id                             int64
         user_type                          object
         member_birth_year                 float64
         member_gender                      object
         bike_share_for_all_trip            object
         dtype: object
```

1.4 Tideness

1.4.1 Issue #1:

The names of the days the bike was rented and returned are not displayed

Define: We to add them both to see when the bike was rented and when it was returned

Code

```
In [15]: df1['start_hour'] = df1.start_time.dt.hour
         df1['end_hour'] = df1.end_time.dt.hour
```

Test

```
In [16]: information(df1)
```

```
 Our dataset has the following number of Columns and Rows (174952, 18)


We have the following columns Index(['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',
       'start_hour', 'end_hour'],
      dtype='object')
```

```
My attributes have the following data types  duration_sec                        int64
start_time                 datetime64[ns]
end_time                   datetime64[ns]
start_station_id                  float64
start_station_name                 object
start_station_latitude            float64
start_station_longitude           float64
end_station_id                    float64
end_station_name                   object
end_station_latitude              float64
end_station_longitude             float64
bike_id                             int64
user_type                          object
member_birth_year                 float64
member_gender                      object
bike_share_for_all_trip            object
start_hour                          int64
end_hour                            int64
dtype: object
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 174952 entries, 0 to 183411
Data columns (total 18 columns):
duration_sec               174952 non-null int64
start_time                 174952 non-null datetime64[ns]
end_time                   174952 non-null datetime64[ns]
start_station_id           174952 non-null float64
start_station_name         174952 non-null object
start_station_latitude     174952 non-null float64
start_station_longitude    174952 non-null float64
end_station_id             174952 non-null float64
end_station_name           174952 non-null object
end_station_latitude       174952 non-null float64
end_station_longitude      174952 non-null float64
bike_id                    174952 non-null int64
user_type                  174952 non-null object
member_birth_year          174952 non-null float64
member_gender              174952 non-null object
bike_share_for_all_trip    174952 non-null object
```

```
start_hour                 174952 non-null int64
end_hour                   174952 non-null int64
dtypes: datetime64[ns](2), float64(7), int64(4), object(5)
memory usage: 25.4+ MB
Here is the brief summary of my dataset None


My dataset have the following number of following attributes duration_sec                 4429
start_time                 174941
end_time                   174939
start_station_id              329
start_station_name            329
start_station_latitude        329
start_station_longitude       329
end_station_id                329
end_station_name              329
end_station_latitude          329
end_station_longitude         329
bike_id                      4607
user_type                       2
member_birth_year              75
member_gender                   3
bike_share_for_all_trip         2
start_hour                     24
end_hour                       24
dtype: int64


My dataset have the following missing values 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
start_hour                 0
end_hour                   0
dtype: int64
```

```
My dataset have this number of duplicates 0
```

1.4.2 Issue #2:

Underserved columns like start_station_latitude, end_station_longitude, etc

Define: Delete all those columns

Code

```
In [17]: df1 = df1.drop(['start_station_latitude', 'start_station_longitude',
                      'end_station_latitude', 'end_station_longitude', 'bike_id'],1)
```

Test

```
In [18]: df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 174952 entries, 0 to 183411
Data columns (total 13 columns):
duration_sec               174952 non-null int64
start_time                 174952 non-null datetime64[ns]
end_time                   174952 non-null datetime64[ns]
start_station_id           174952 non-null float64
start_station_name         174952 non-null object
end_station_id             174952 non-null float64
end_station_name           174952 non-null object
user_type                  174952 non-null object
member_birth_year          174952 non-null float64
member_gender              174952 non-null object
bike_share_for_all_trip    174952 non-null object
start_hour                 174952 non-null int64
end_hour                   174952 non-null int64
dtypes: datetime64[ns](2), float64(3), int64(3), object(5)
memory usage: 18.7+ MB
```

```
In [19]: information(df1)
```

```
 Our dataset has the following number of Columns and Rows (174952, 13)


We have the following columns Index(['duration_sec', 'start_time', 'end_time', 'start_station_id
       'start_station_name', 'end_station_id', 'end_station_name', 'user_type',
       'member_birth_year', 'member_gender', 'bike_share_for_all_trip',
       'start_hour', 'end_hour'],
```

```
      dtype='object')


My attributes have the following data types  duration_sec                        int64
start_time                 datetime64[ns]
end_time                   datetime64[ns]
start_station_id                  float64
start_station_name                 object
end_station_id                    float64
end_station_name                   object
user_type                          object
member_birth_year                 float64
member_gender                      object
bike_share_for_all_trip            object
start_hour                          int64
end_hour                            int64
dtype: object


<class 'pandas.core.frame.DataFrame'>
Int64Index: 174952 entries, 0 to 183411
Data columns (total 13 columns):
duration_sec               174952 non-null int64
start_time                 174952 non-null datetime64[ns]
end_time                   174952 non-null datetime64[ns]
start_station_id           174952 non-null float64
start_station_name         174952 non-null object
end_station_id             174952 non-null float64
end_station_name           174952 non-null object
user_type                  174952 non-null object
member_birth_year          174952 non-null float64
member_gender              174952 non-null object
bike_share_for_all_trip    174952 non-null object
start_hour                 174952 non-null int64
end_hour                   174952 non-null int64
dtypes: datetime64[ns](2), float64(3), int64(3), object(5)
memory usage: 18.7+ MB
Here is the brief summary of my dataset None


My dataset have the following number of following attributes duration_sec                4429
start_time                 174941
end_time                   174939
start_station_id              329
start_station_name            329
end_station_id                329
end_station_name              329
user_type                       2
```

```
member_birth_year              75
member_gender                   3
bike_share_for_all_trip         2
start_hour                     24
end_hour                       24
dtype: int64


My dataset have the following missing values duration_sec               0
start_time                 0
end_time                   0
start_station_id           0
start_station_name         0
end_station_id             0
end_station_name           0
user_type                  0
member_birth_year          0
member_gender              0
bike_share_for_all_trip    0
start_hour                 0
end_hour                   0
dtype: int64


My dataset have this number of duplicates 0
```

1.4.3 What is the structure of your dataset?

We now have 174952 rows and 13 columns in our dataset. There are various organized data types for the attributes. None of the values are missing. Finally, there are no duplicates in our dataset.

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

My main features are time and gender

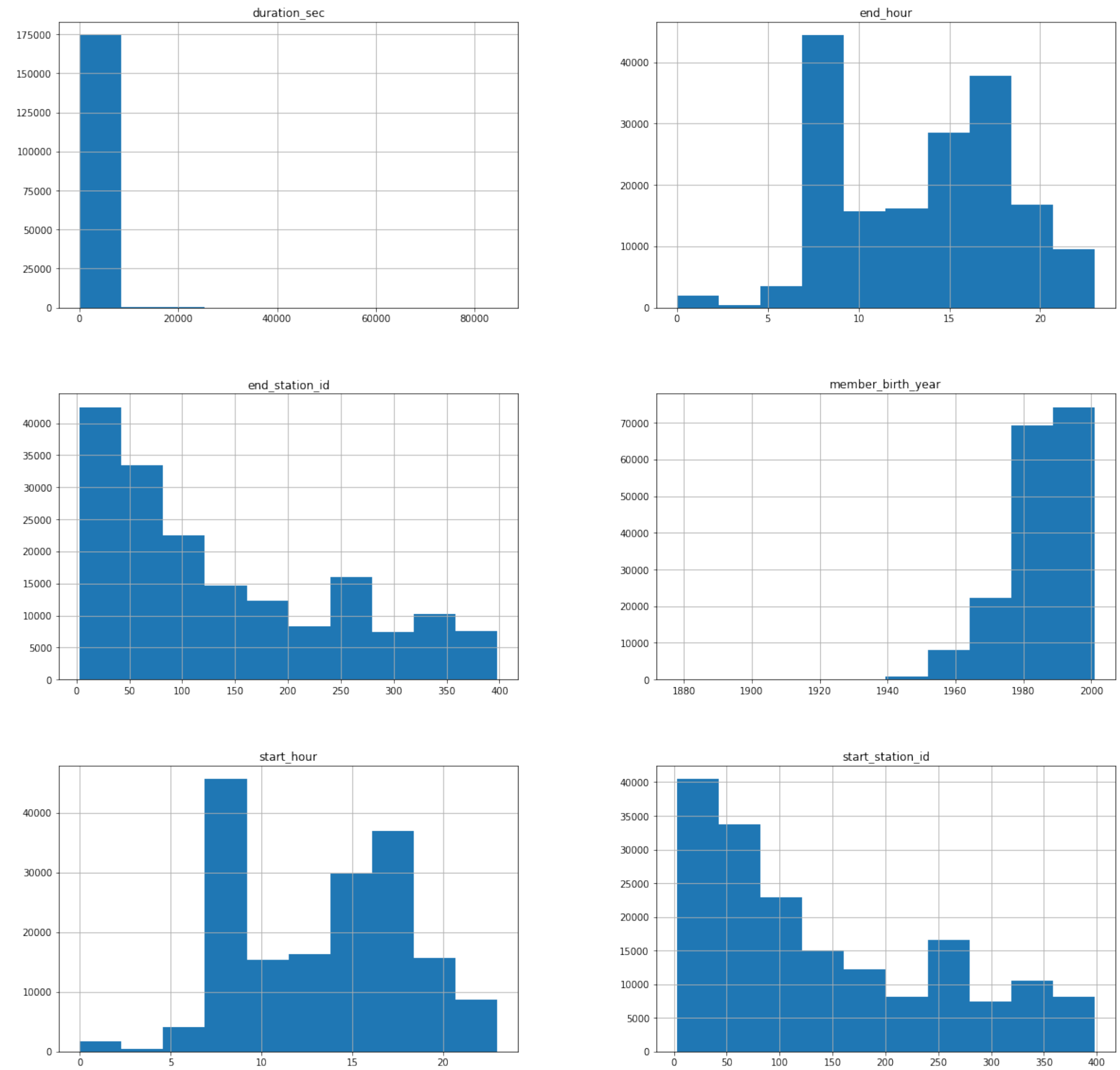
1.4.5 What features in the dataset do you think will help support your investigation into your feature(s) of interest?

The features that I think will support my inverstigation are start_hour, end_hour, member_gender, user_types and maybe others that might help to understand those ones.

1.5 Univariate Exploration

1. Data visualization for numerical data

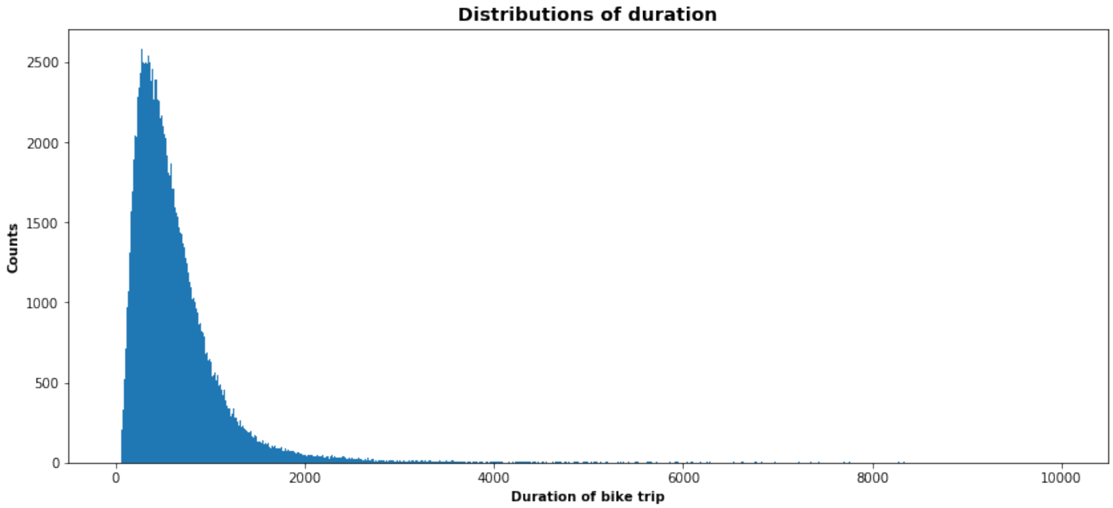
```
In [21]: #histogram to visualize all numerical data in ourdata set at ones
         df1.hist(figsize=(20,20));
```



Some of the interesting outcomes from the graphs are: - The duration seems to be in one group. It will be necessary to view it alone and see why. And it looks like it is between 0 and 10000 - The end hour is skewed to the right, and between 7 and 8, people seem to finnsh at that time. If not, then between the range of 7 and 20. - The start hour is more or less similar to the end hour.

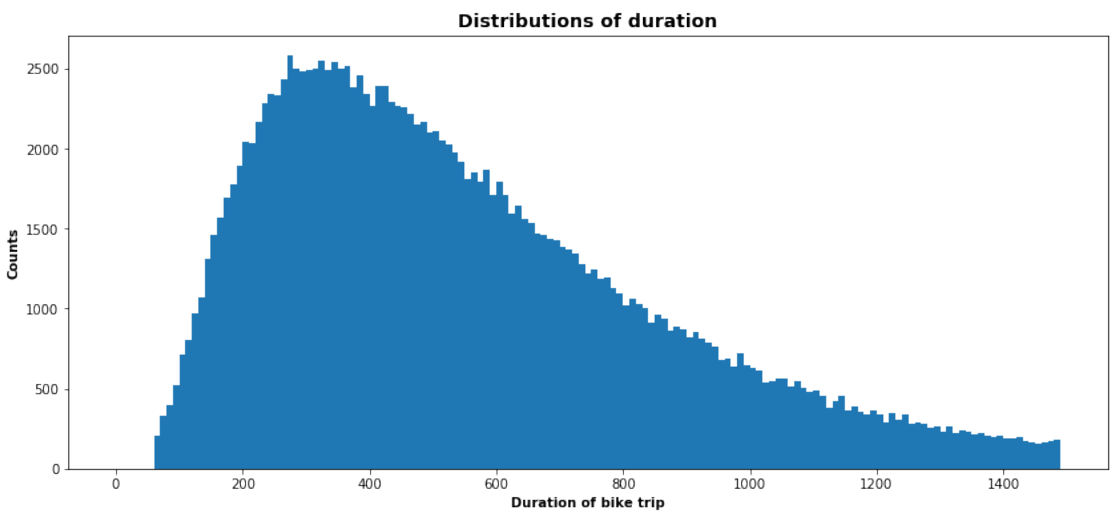
2. Visualizing the duration

```
In [24]: #duration histogram
         bins = np.arange(0, 10000, 10)
         plt.figure(figsize=(14, 6))
         plt.hist(df1.duration_sec, bins=bins)
         plt.title('Distributions of duration', fontsize=14, weight='bold')
         plt.ylabel('Counts', fontsize=10, weight='bold')
         plt.xlabel('Duration of bike trip', fontsize=10, weight='bold');
```



- It knows it shows that it is skewed to the right, but more data points this time are between 0 and 1500.
- To see it clearly, we can slice it more.

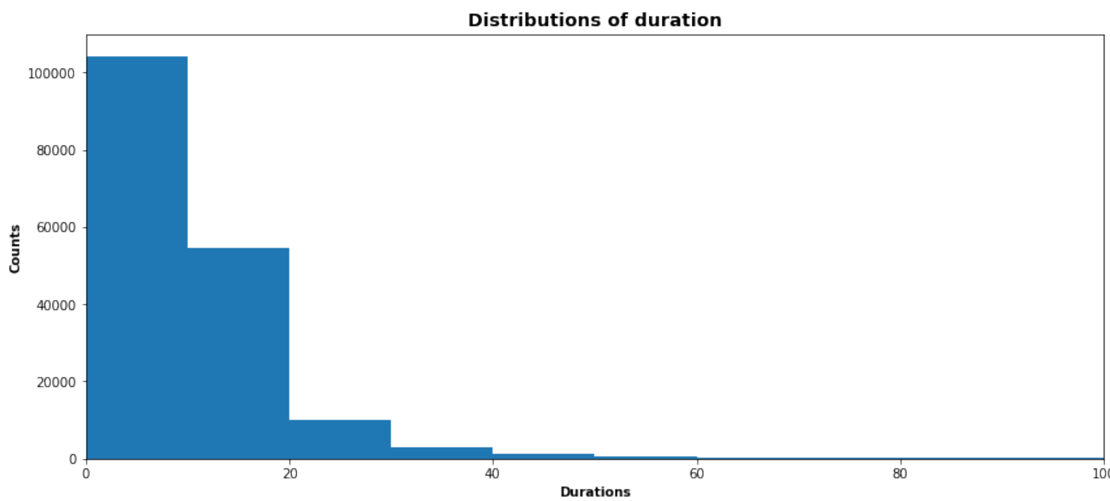
```
In [25]: bins = np.arange(0, 1500, 10)
         plt.figure(figsize=(14, 6))
         plt.hist(df1.duration_sec, bins=bins)
         plt.title('Distributions of duration', fontsize=14, weight='bold')
         plt.ylabel('Counts', fontsize=10, weight='bold')
         plt.xlabel('Duration of bike trip', fontsize=10, weight='bold');
```



This makes it obvious that the pick is close to 400.
We may now attempt to plot the first 100 minutes in 10 minute intervals to examine the results.

In [27]:
```
df1['duration_min'] =df1['duration_sec']/60
bins = np.arange(0, df1.duration_min.max(0)+1, 10)
plt.figure(figsize=(14, 6))
plt.hist(data=df1, x='duration_min', bins=bins)
plt.title('Distributions of duration', fontsize=14, weight='bold')
plt.ylabel('Counts', fontsize=10, weight='bold')
plt.xlabel('Durations', fontsize=10, weight='bold')
plt.xlim((0,100))
plt.ylim((0,110000))
```

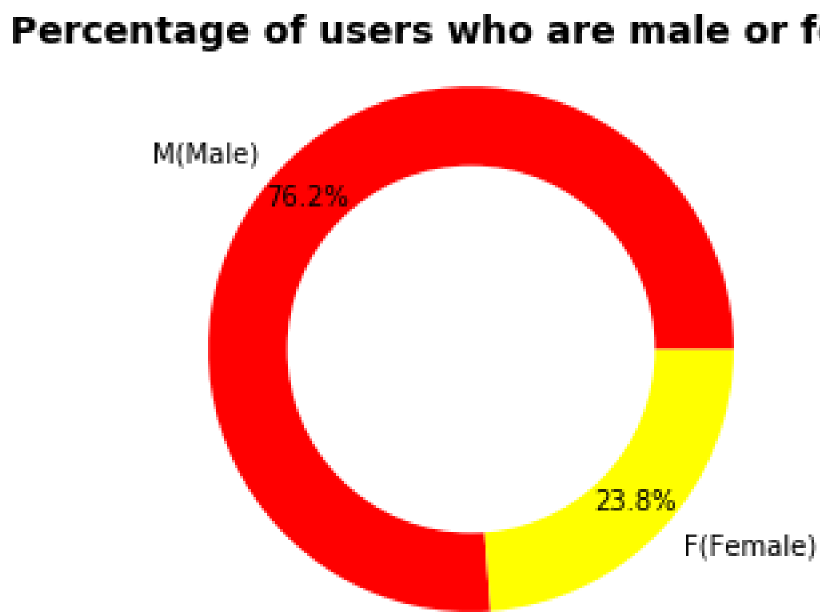
Out[27]: (0, 110000)



Considering the intervals between 0 and 100 minutes. Most rides, as can be seen, occur within the first 10 minutes.

3. Percentage of users who are male and female

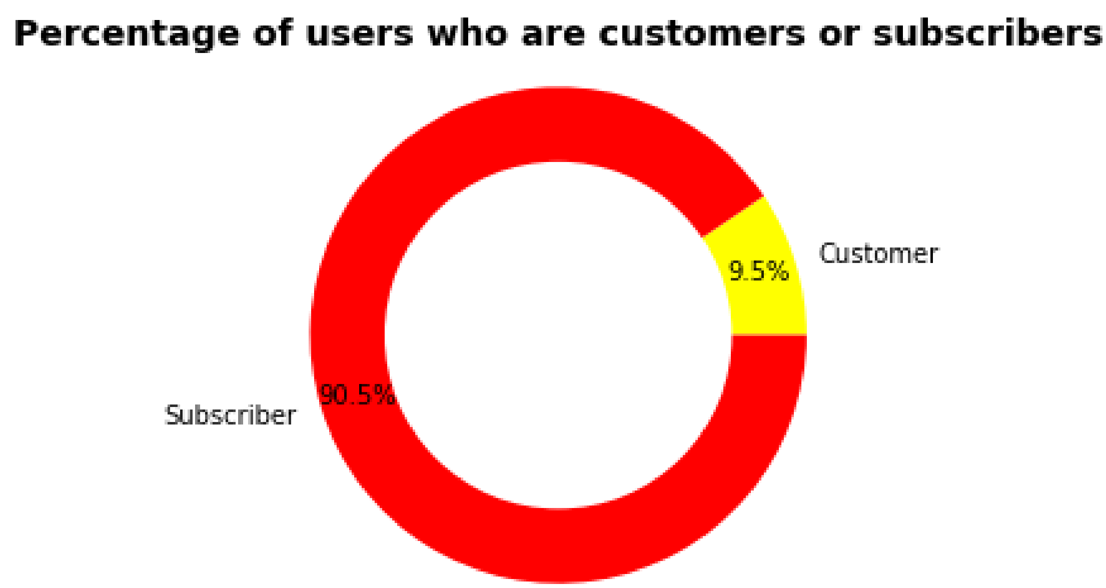
In [28]:
```
#Donut chart for gender
M = df1.query("member_gender == 'Male'")["member_gender"].count()
F = df1.query("member_gender == 'Female'")["member_gender"].count()
gender = [M, F]
labels = 'M(Male)', 'F(Female)'
colors = ['#FF0000', '#FFFF00']
plt.pie(gender, colors=colors, labels=labels, autopct='%1.1f%%', pctdistance=0.85)
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.axis('square');
plt.title(" Percentage of users who are male or female", fontsize=14, weight='bold')
plt.show()
```



The gender split is 76.2% men to 23.8% women.

4. Number of User Types Split

```
In [29]: #Donut chart for number of user types split
         customer = df1.query("user_type == 'Customer'")["user_type"].count()
         subscriber = df1.query("user_type == 'Subscriber'")["user_type"].count()
         labels = 'Customer', 'Subscriber'
         colors = ['#FFFF00', '#FF0000']
         userType = [customer, subscriber]
         plt.pie(userType, colors=colors, labels=labels, autopct='%1.1f%%',pctdistance=0.85)
         centre_circle = plt.Circle((0, 0), 0.70, fc='white')
         fig = plt.gcf()
         plt.axis('square');
         fig.gca().add_artist(centre_circle)
         plt.title("Percentage of users who are customers or subscribers", fontsize=14, weight='
         plt.show()
```



90.5% of users fall under the subscriber category, and 9.5% fall under the customer category.

4.The number of bike rides rented on the various days of the week

```
In [30]: df1
```

```
Out[30]:        duration_sec              start_time                end_time  \
         0             52185 2019-02-28 17:32:10.145 2019-03-01 08:01:55.975
         2             61854 2019-02-28 12:13:13.218 2019-03-01 05:24:08.146
         3             36490 2019-02-28 17:54:26.010 2019-03-01 04:02:36.842
         4              1585 2019-02-28 23:54:18.549 2019-03-01 00:20:44.074
         5              1793 2019-02-28 23:49:58.632 2019-03-01 00:19:51.760
         6              1147 2019-02-28 23:55:35.104 2019-03-01 00:14:42.588
         7              1615 2019-02-28 23:41:06.766 2019-03-01 00:08:02.756
         8              1570 2019-02-28 23:41:48.790 2019-03-01 00:07:59.715
         9              1049 2019-02-28 23:49:47.699 2019-03-01 00:07:17.025
         10              458 2019-02-28 23:57:57.211 2019-03-01 00:05:35.435
         11              506 2019-02-28 23:56:55.540 2019-03-01 00:05:21.733
         12             1176 2019-02-28 23:45:12.651 2019-03-01 00:04:49.184
         14              395 2019-02-28 23:56:26.848 2019-03-01 00:03:01.947
         15              208 2019-02-28 23:59:18.548 2019-03-01 00:02:47.228
         16              548 2019-02-28 23:50:41.607 2019-02-28 23:59:49.953
         17              674 2019-02-28 23:48:25.095 2019-02-28 23:59:40.092
         18              557 2019-02-28 23:49:01.851 2019-02-28 23:58:19.809
         19              874 2019-02-28 23:43:05.183 2019-02-28 23:57:39.796
         20              417 2019-02-28 23:50:38.239 2019-02-28 23:57:35.852
         21              414 2019-02-28 23:50:26.879 2019-02-28 23:57:21.130
         22              743 2019-02-28 23:44:56.439 2019-02-28 23:57:20.212
```

```
23              367 2019-02-28 23:51:06.014 2019-02-28 23:57:13.312
24              252 2019-02-28 23:52:51.164 2019-02-28 23:57:03.976
25              360 2019-02-28 23:50:31.431 2019-02-28 23:56:31.891
26              385 2019-02-28 23:49:24.399 2019-02-28 23:55:50.284
27              408 2019-02-28 23:48:08.282 2019-02-28 23:54:56.930
29              629 2019-02-28 23:43:48.658 2019-02-28 23:54:18.254
30              163 2019-02-28 23:50:45.698 2019-02-28 23:53:29.569
31              223 2019-02-28 23:49:27.027 2019-02-28 23:53:10.535
32              405 2019-02-28 23:45:39.234 2019-02-28 23:52:24.850
...             ...                     ...                     ...
183381          426 2019-02-01 00:48:54.159 2019-02-01 00:56:00.474
183382          961 2019-02-01 00:38:29.904 2019-02-01 00:54:31.732
183383          434 2019-02-01 00:47:11.653 2019-02-01 00:54:26.305
183384          184 2019-02-01 00:50:41.579 2019-02-01 00:53:46.124
183385          400 2019-02-01 00:46:47.276 2019-02-01 00:53:27.596
183386          425 2019-02-01 00:42:20.472 2019-02-01 00:49:25.515
183387          598 2019-02-01 00:39:12.684 2019-02-01 00:49:10.791
183388          490 2019-02-01 00:39:53.112 2019-02-01 00:48:03.338
183389          184 2019-02-01 00:43:56.556 2019-02-01 00:47:01.009
183390          232 2019-02-01 00:40:00.035 2019-02-01 00:43:52.880
183391          269 2019-02-01 00:37:47.527 2019-02-01 00:42:17.060
183392         1289 2019-02-01 00:19:45.641 2019-02-01 00:41:15.558
183393          155 2019-02-01 00:37:26.368 2019-02-01 00:40:01.576
183394          720 2019-02-01 00:27:33.834 2019-02-01 00:39:34.233
183395           95 2019-02-01 00:37:23.115 2019-02-01 00:38:58.346
183396          576 2019-02-01 00:27:06.503 2019-02-01 00:36:43.452
183397          438 2019-02-01 00:28:56.101 2019-02-01 00:36:14.534
183398         1019 2019-02-01 00:16:59.155 2019-02-01 00:33:58.590
183399          958 2019-02-01 00:12:24.247 2019-02-01 00:28:22.738
183400          250 2019-02-01 00:23:52.611 2019-02-01 00:28:02.679
183401          383 2019-02-01 00:16:48.062 2019-02-01 00:23:11.201
183403          249 2019-02-01 00:15:12.067 2019-02-01 00:19:21.699
183404          256 2019-02-01 00:12:50.554 2019-02-01 00:17:07.362
183405          111 2019-02-01 00:14:49.874 2019-02-01 00:16:41.301
183406          706 2019-02-01 00:04:40.616 2019-02-01 00:16:27.080
183407          480 2019-02-01 00:04:49.724 2019-02-01 00:12:50.034
183408          313 2019-02-01 00:05:34.744 2019-02-01 00:10:48.502
183409          141 2019-02-01 00:06:05.549 2019-02-01 00:08:27.220
183410          139 2019-02-01 00:05:34.360 2019-02-01 00:07:54.287
183411          271 2019-02-01 00:00:20.636 2019-02-01 00:04:52.058

        start_station_id                                   start_station_name  \
0                   21.0   Montgomery St BART Station (Market St at 2nd St)
2                   86.0                            Market St at Dolores St
3                  375.0                            Grove St at Masonic Ave
4                    7.0                                Frank H Ogawa Plaza
5                   93.0                       4th St at Mission Bay Blvd S
6                  300.0                               Palm St at Willow St
```

7	10.0	Washington St at Kearny St
8	10.0	Washington St at Kearny St
9	19.0	Post St at Kearny St
10	370.0	Jones St at Post St
11	44.0	Civic Center/UN Plaza BART Station (Market St ...
12	127.0	Valencia St at 21st St
14	243.0	Bancroft Way at College Ave
15	349.0	Howard St at Mary St
16	131.0	22nd St at Dolores St
17	74.0	Laguna St at Hayes St
18	321.0	5th St at Folsom
19	180.0	Telegraph Ave at 23rd St
20	72.0	Page St at Scott St
21	163.0	Lake Merritt BART Station
22	370.0	Jones St at Post St
23	243.0	Bancroft Way at College Ave
24	190.0	West St at 40th St
25	163.0	Lake Merritt BART Station
26	6.0	The Embarcadero at Sansome St
27	78.0	Folsom St at 9th St
29	258.0	University Ave at Oxford St
30	238.0	MLK Jr Way at University Ave
31	28.0	The Embarcadero at Bryant St
32	109.0	17th St at Valencia St
...	...	...
183381	230.0	14th St at Mandela Pkwy
183382	95.0	Sanchez St at 15th St
183383	274.0	Oregon St at Adeline St
183384	316.0	San Salvador St at 1st St
183385	220.0	San Pablo Ave at MLK Jr Way
183386	239.0	Bancroft Way at Telegraph Ave
183387	239.0	Bancroft Way at Telegraph Ave
183388	61.0	Howard St at 8th St
183389	66.0	3rd St at Townsend St
183390	239.0	Bancroft Way at Telegraph Ave
183391	119.0	18th St at Noe St
183392	8.0	The Embarcadero at Vallejo St
183393	116.0	Mississippi St at 17th St
183394	26.0	1st St at Folsom St
183395	276.0	Julian St at The Alameda
183396	181.0	Grand Ave at Webster St
183397	62.0	Victoria Manalo Draves Park
183398	339.0	Jackson St at 11th St
183399	67.0	San Francisco Caltrain Station 2 (Townsend St...
183400	356.0	Valencia St at Clinton Park
183401	186.0	Lakeside Dr at 14th St
183403	256.0	Hearst Ave at Euclid Ave
183404	241.0	Ashby BART Station

```
183405           324.0             Union Square (Powell St at Post St)
183406           138.0                         Jersey St at Church St
183407            27.0                         Beale St at Harrison St
183408            21.0  Montgomery St BART Station (Market St at 2nd St)
183409           278.0                         The Alameda at Bush St
183410           220.0                      San Pablo Ave at MLK Jr Way
183411            24.0                           Spear St at Folsom St

        end_station_id                                end_station_name  \
0                 13.0                  Commercial St at Montgomery St
2                  3.0     Powell St BART Station (Market St at 4th St)
3                 70.0                          Central Ave at Fell St
4                222.0                          10th Ave at E 15th St
5                323.0                              Broadway at Kearny
6                312.0                        San Jose Diridon Station
7                127.0                          Valencia St at 21st St
8                127.0                          Valencia St at 21st St
9                121.0                              Mission Playground
10                43.0  San Francisco Public Library (Grove St at Hyde...
11               343.0                             Bryant St at 2nd St
12               323.0                              Broadway at Kearny
14               252.0                    Channing Way at Shattuck Ave
15                60.0                            8th St at Ringold St
16                71.0                          Broderick St at Oak St
17               336.0                     Potrero Ave and Mariposa St
18                75.0                        Market St at Franklin St
19               180.0                        Telegraph Ave at 23rd St
20               107.0                          17th St at Dolores St
21               221.0         6th Ave at E 12th St (Temporary Location)
22                52.0                        McAllister St at Baker St
23               269.0                    Telegraph Ave at Carleton St
24               189.0                            Genoa St at 55th St
25               196.0                           Grand Ave at Perkins St
26                15.0  San Francisco Ferry Building (Harry Bridges Pl...
27                78.0                             Folsom St at 9th St
29               263.0                   Channing Way at San Pablo Ave
30               244.0                      Shattuck Ave at Hearst Ave
31                50.0                          2nd St at Townsend St
32                73.0                          Pierce St at Haight St
...                ...                                             ...
183381           213.0                          32nd St at Adeline St
183382           324.0             Union Square (Powell St at Post St)
183383           244.0                      Shattuck Ave at Hearst Ave
183384           298.0                                Oak St at 1st St
183385           337.0                          Webster St at 19th St
183386           245.0                         Downtown Berkeley BART
183387           245.0                         Downtown Berkeley BART
183388            81.0                              Berry St at 4th St
```

```
183389          47.0                    4th St at Harrison St
183390         266.0                    Parker St at Fulton St
183391          85.0                   Church St at Duboce Ave
183392         350.0                     8th St at Brannan St
183393          93.0              4th St at Mission Bay Blvd S
183394          96.0                    Dolores St at 15th St
183395         277.0                 Morrison Ave at Julian St
183396         212.0                             Mosswood Park
183397          59.0               S Van Ness Ave at Market St
183398          46.0                         San Antonio Park
183399          58.0                     Market St at 10th St
183400          58.0                     Market St at 10th St
183401         181.0                   Grand Ave at Webster St
183403         247.0                 Fulton St at Bancroft Way
183404         248.0                Telegraph Ave at Ashby Ave
183405          19.0                     Post St at Kearny St
183406          78.0                      Folsom St at 9th St
183407         324.0        Union Square (Powell St at Post St)
183408          66.0                     3rd St at Townsend St
183409         277.0                 Morrison Ave at Julian St
183410         216.0                  San Pablo Ave at 27th St
183411          37.0                      2nd St at Folsom St

         user_type  member_birth_year member_gender bike_share_for_all_trip  \
0         Customer             1984.0          Male                      No
2         Customer             1972.0          Male                      No
3       Subscriber             1989.0         Other                      No
4       Subscriber             1974.0          Male                     Yes
5       Subscriber             1959.0          Male                      No
6       Subscriber             1983.0        Female                      No
7       Subscriber             1989.0          Male                      No
8       Subscriber             1988.0         Other                      No
9       Subscriber             1992.0          Male                      No
10      Subscriber             1996.0        Female                     Yes
11      Subscriber             1993.0          Male                      No
12        Customer             1990.0          Male                      No
14      Subscriber             1988.0          Male                      No
15      Subscriber             1993.0          Male                     Yes
16      Subscriber             1981.0          Male                      No
17      Subscriber             1975.0          Male                      No
18      Subscriber             1990.0          Male                      No
19        Customer             1978.0          Male                      No
20      Subscriber             1983.0          Male                      No
21      Subscriber             1984.0          Male                     Yes
22      Subscriber             1991.0        Female                      No
23      Subscriber             1997.0        Female                      No
24      Subscriber             1975.0          Male                      No
25      Subscriber             1986.0          Male                      No
```

```
26        Customer        2000.0      Male     No
27      Subscriber        1982.0      Male     No
29      Subscriber        1995.0      Male     No
30      Subscriber        1996.0      Male    Yes
31        Customer        1993.0      Male     No
32      Subscriber        1980.0    Female     No
...            ...           ...       ...    ...
183381  Subscriber        1997.0     Other    Yes
183382  Subscriber        1988.0      Male     No
183383    Customer        1997.0      Male     No
183384  Subscriber        1991.0      Male     No
183385  Subscriber        1945.0      Male    Yes
183386  Subscriber        1998.0      Male    Yes
183387  Subscriber        1999.0      Male    Yes
183388  Subscriber        1927.0      Male     No
183389  Subscriber        1985.0     Other     No
183390  Subscriber        1999.0      Male     No
183391  Subscriber        1980.0      Male    Yes
183392  Subscriber        1993.0      Male     No
183393  Subscriber        1985.0      Male     No
183394  Subscriber        1975.0      Male     No
183395  Subscriber        1993.0      Male    Yes
183396  Subscriber        1991.0      Male    Yes
183397  Subscriber        1988.0      Male     No
183398  Subscriber        1982.0      Male     No
183399  Subscriber        1993.0      Male     No
183400  Subscriber        1984.0      Male     No
183401  Subscriber        1991.0      Male    Yes
183403  Subscriber        2000.0      Male     No
183404  Subscriber        1980.0      Male    Yes
183405  Subscriber        1984.0      Male     No
183406  Subscriber        1988.0      Male     No
183407  Subscriber        1996.0      Male     No
183408  Subscriber        1984.0      Male     No
183409  Subscriber        1990.0      Male    Yes
183410  Subscriber        1988.0      Male     No
183411  Subscriber        1989.0      Male     No

        start_hour  end_hour  duration_min
0               17         8    869.750000
2               12         5   1030.900000
3               17         4    608.166667
4               23         0     26.416667
5               23         0     29.883333
6               23         0     19.116667
7               23         0     26.916667
8               23         0     26.166667
9               23         0     17.483333
```

10	23	0	7.633333
11	23	0	8.433333
12	23	0	19.600000
14	23	0	6.583333
15	23	0	3.466667
16	23	23	9.133333
17	23	23	11.233333
18	23	23	9.283333
19	23	23	14.566667
20	23	23	6.950000
21	23	23	6.900000
22	23	23	12.383333
23	23	23	6.116667
24	23	23	4.200000
25	23	23	6.000000
26	23	23	6.416667
27	23	23	6.800000
29	23	23	10.483333
30	23	23	2.716667
31	23	23	3.716667
32	23	23	6.750000
...	...	...	...
183381	0	0	7.100000
183382	0	0	16.016667
183383	0	0	7.233333
183384	0	0	3.066667
183385	0	0	6.666667
183386	0	0	7.083333
183387	0	0	9.966667
183388	0	0	8.166667
183389	0	0	3.066667
183390	0	0	3.866667
183391	0	0	4.483333
183392	0	0	21.483333
183393	0	0	2.583333
183394	0	0	12.000000
183395	0	0	1.583333
183396	0	0	9.600000
183397	0	0	7.300000
183398	0	0	16.983333
183399	0	0	15.966667
183400	0	0	4.166667
183401	0	0	6.383333
183403	0	0	4.150000
183404	0	0	4.266667
183405	0	0	1.850000
183406	0	0	11.766667
183407	0	0	8.000000

```
183408          0          0     5.216667
183409          0          0     2.350000
183410          0          0     2.316667
183411          0          0     4.516667

[174952 rows x 14 columns]
```

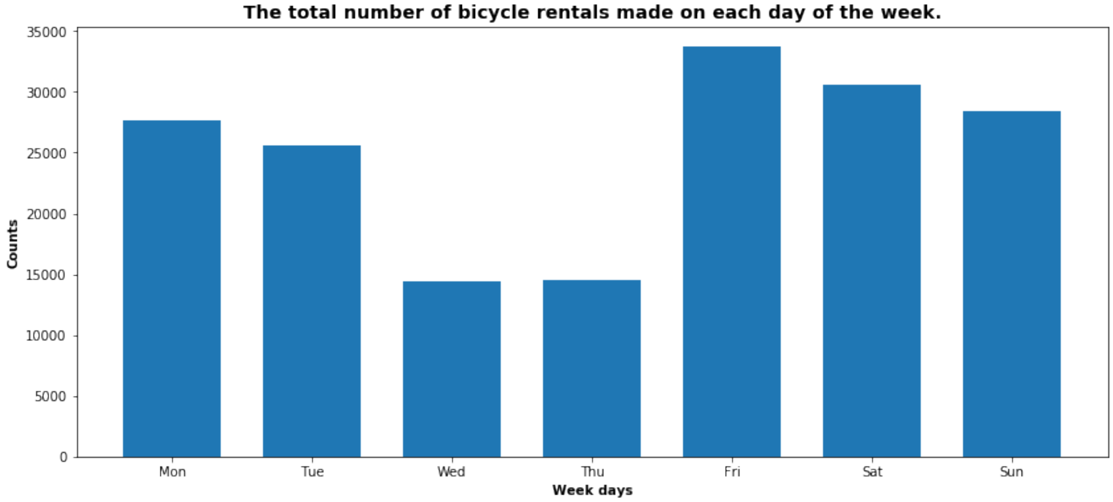
In [31]:
```
# Produce and start day and an end day column for the start time and end time columns
df1['start_day'] = df1['start_time'].dt.day_name()
df1['end_day'] = df1['end_time'].dt.day_name()
```

In [32]:
```
df1.head()
```

Out[32]:
```
   duration_sec              start_time                end_time  \
0         52185 2019-02-28 17:32:10.145 2019-03-01 08:01:55.975
2         61854 2019-02-28 12:13:13.218 2019-03-01 05:24:08.146
3         36490 2019-02-28 17:54:26.010 2019-03-01 04:02:36.842
4          1585 2019-02-28 23:54:18.549 2019-03-01 00:20:44.074
5          1793 2019-02-28 23:49:58.632 2019-03-01 00:19:51.760

   start_station_id                                start_station_name  \
0              21.0  Montgomery St BART Station (Market St at 2nd St)
2              86.0                           Market St at Dolores St
3             375.0                           Grove St at Masonic Ave
4               7.0                               Frank H Ogawa Plaza
5              93.0                          4th St at Mission Bay Blvd S

   end_station_id                              end_station_name   user_type  \
0            13.0                Commercial St at Montgomery St    Customer
2             3.0  Powell St BART Station (Market St at 4th St)    Customer
3            70.0                        Central Ave at Fell St  Subscriber
4           222.0                         10th Ave at E 15th St  Subscriber
5           323.0                           Broadway at Kearny St  Subscriber

   member_birth_year member_gender bike_share_for_all_trip  start_hour  \
0             1984.0          Male                      No          17
2             1972.0          Male                      No          12
3             1989.0         Other                      No          17
4             1974.0          Male                     Yes          23
5             1959.0          Male                      No          23

   end_hour  duration_min start_day end_day
0         8    869.750000  Thursday  Friday
2         5   1030.900000  Thursday  Friday
3         4    608.166667  Thursday  Friday
4         0     26.416667  Thursday  Friday
5         0     29.883333  Thursday  Friday
```

In [34]:
```
#Producing a bar chart for start day
days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
```

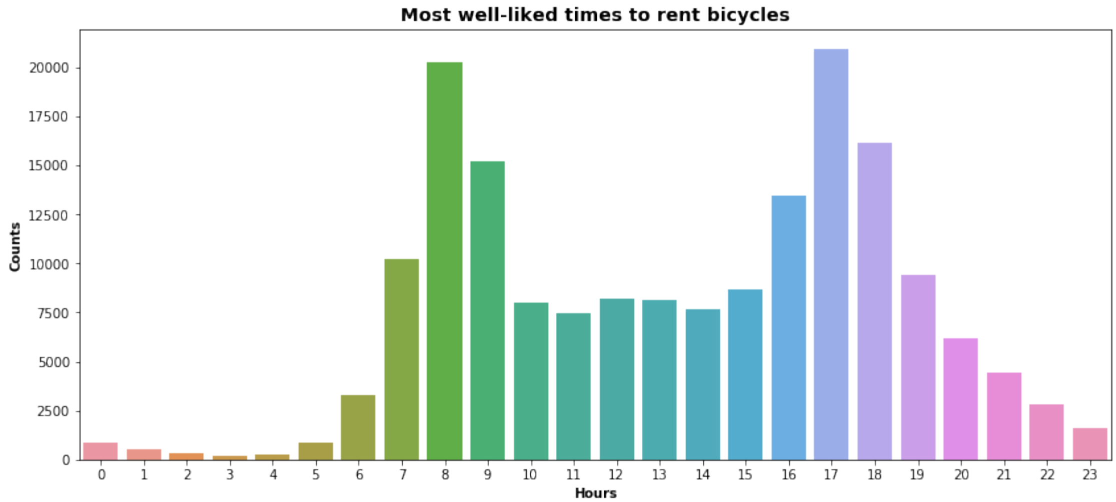
```
plt.figure(figsize=[14,6])
bin_edges = np.arange(-0.5, 6.5+1, 1)
plt.hist(data = df1, x = 'start_day', bins = bin_edges, rwidth = 0.7)
plt.xticks(np.arange(0, 6+1, 1), days)
plt.title('The total number of bicycle rentals made on each day of the week.', fontsize
plt.xlabel('Week days' , fontsize=10, weight='bold')
plt.ylabel('Counts' , fontsize=10, weight='bold');
```



- It is interesting how many bicycles are rented on Friday, and I think it is because people may go for a walk with their bike to relax after work. And also on the weeks. It is also interesting that people rent bicycles on Mondays and Tuesdays. Maybe relax for five days and work for two days, lol. People don't rent much on Wednesday and Thursday, and I presume many people are working here.
- It will be interesting to check which are the most popular hours to rent a bike

6.Periods when renting bicycles is most common

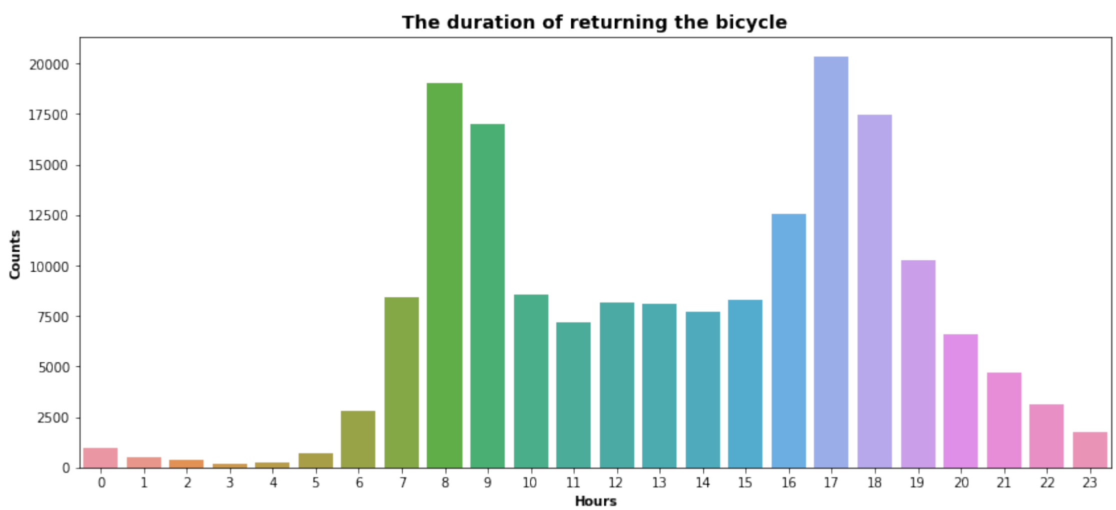
```
In [35]: order = np.arange(0,24)
         plt.figure(figsize=(14, 6))
         plt.title('Most well-liked times to rent bicycles', fontsize=14, weight='bold')
         ax = sb.countplot(data=df1, x='start_hour', order=order)
         plt.ylabel('Counts', fontsize=10, weight='bold')
         plt.xlabel('Hours', fontsize=10, weight='bold');
```



From 7 am to 7 pm is when most people want to rent bikes. Because humans are active from dawn until evening and sleep at night, I believe this to be the case.

7. The time the bike was delivered back

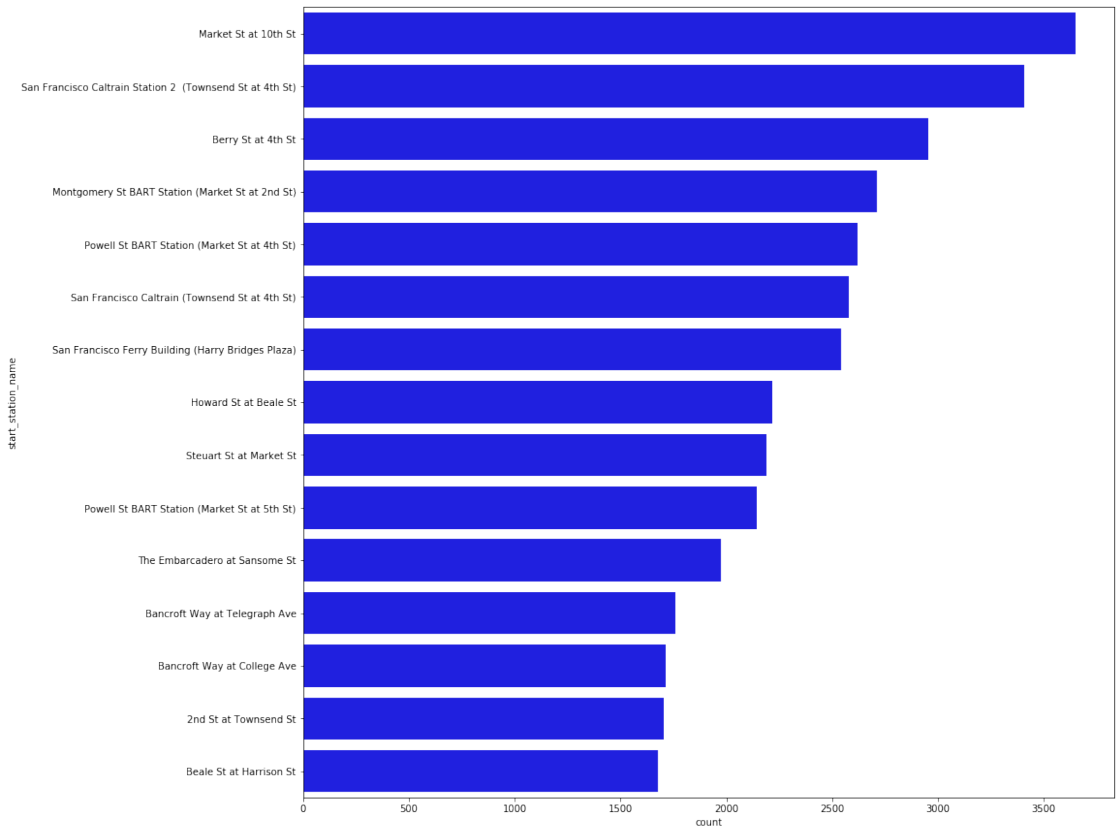
In [37]:
```
hour_order = np.arange(0,24)
plt.figure(figsize=(14, 6))
plt.title('The duration of returning the bicycle', fontsize=14, weight='bold')
ax = sb.countplot(data=df1, x='end_hour', order=hour_order)
plt.ylabel('Counts', fontsize=10, weight='bold')
plt.xlabel('Hours', fontsize=10, weight='bold');
```



It appears that the return window, which was from 7 am to 7 pm, was almost the same as the ranting window.

8. The Leading ten starting stations

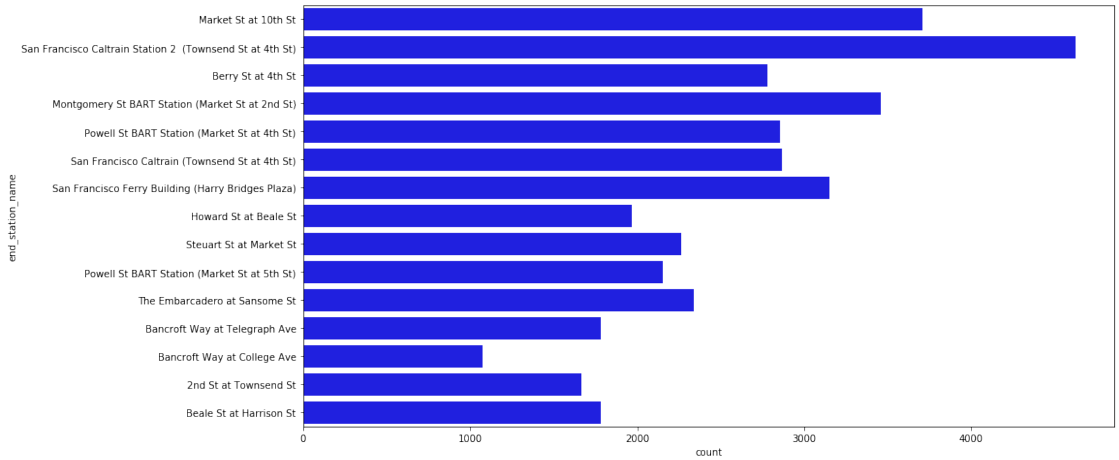
```
In [38]: station = df1['start_station_name'].value_counts().index[:15]
         plt.figure(figsize=[15,15])
         sb.countplot(data = df1, y = 'start_station_name', order = station, color = "blue");
```



The best three starting situations are Market st, San Francisco Caltrain, and Berry st, respectively.

9. The Leading ten ending stations

```
In [39]: end = df1['end_station_name'].value_counts().index[:10]
         plt.figure(figsize=[15,8])
         sb.countplot(data = df1, y = 'end_station_name', order = station, color = "blue");
```



Again the best three ending situations are Market st, San Francisco Caltrain, and Berry st, respectively.

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

No changes were made, although, for many charts, the duration of seconds was changed to the period of minutes because it is simpler to read than to calculate how long. Around 7 am and 7 pm are when most bicycling ends and starts. Men primarily use bikes, and 90.5% of users are subscribers. In addition to Monday and Tuesday, Friday, Saturday, and Sunday are the busiest days.

1.5.2 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?

Yes, I did perform operations like changing datatypes for some features and creating new columns.

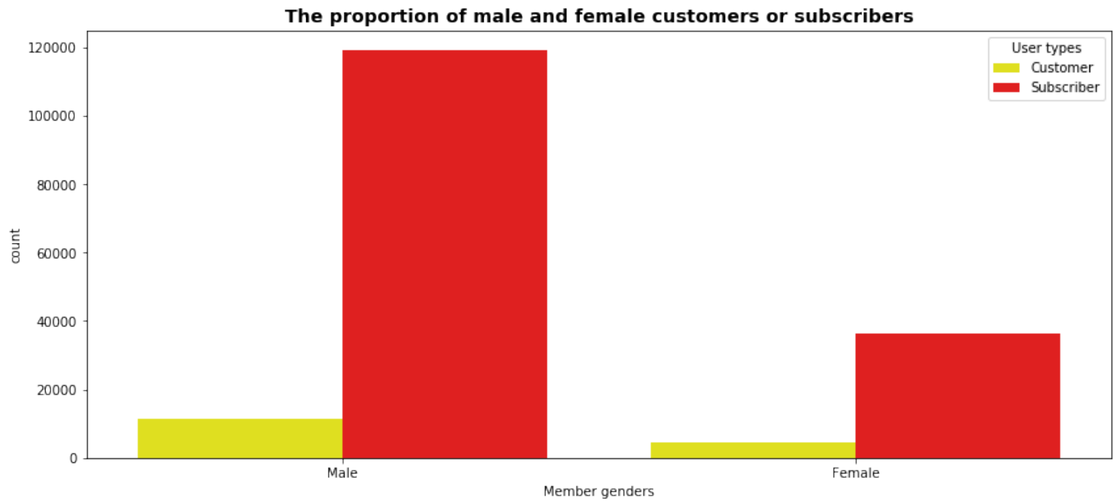
1.6 Bivariate Exploration

10. What connection exists between user types and gender? *One issue to solve for use to perform a better analysis on gender is removing other genders, not male and female.*

```
In [40]: #remove other gender
         #gender = df1.groupby(['member_gender'])['member_gender'].count()
         df2 = df1.loc[df1["member_gender"] != 'Other']
```

```
In [42]: plt.figure(figsize=(14,6))
         ax = sb.countplot(x=df2.member_gender, hue=df2.user_type, palette=['#FFFF00', '#FF0000'
         ax.set_title("The proportion of male and female customers or subscribers" , fontsize=14
         x_ticks_labels=['Male', 'Female']
         ax.set_xticklabels(x_ticks_labels)
         ax.set_xlabel('Member genders')
```

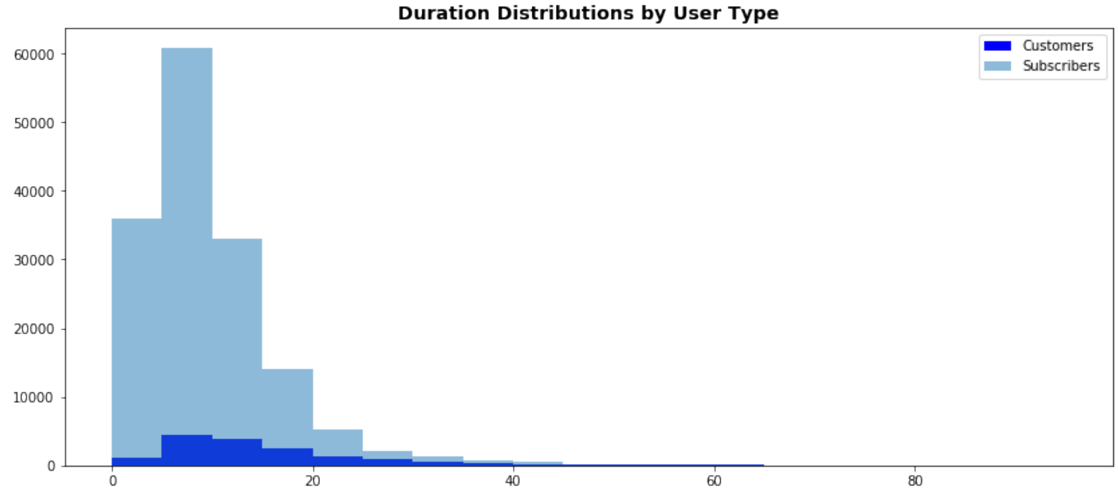
```
plt.legend(title='User types', loc='upper right', labels=['Customer', 'Subscriber'])
plt.show()
```



Although there are far more subscribers than customers, there are significantly more male subscribers than female subscribers. In terms of clients, males outnumber females.

11. What is the relationship between gender and user types?

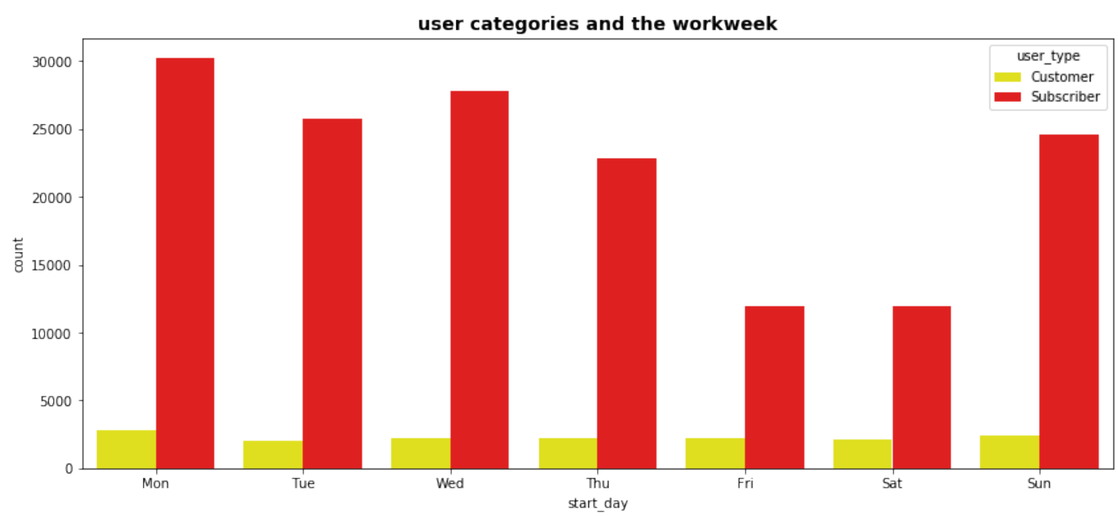
```
In [44]: #showing customers and subscribers duration distributions in one graph

         customer = df2['user_type']=='Customer'
         subscriber = df2['user_type']=='Subscriber'
         plt.figure(figsize=(14, 6))
         bins = np.arange(0, 100, 5)
         plt.hist(df2[customer].duration_min, bins, alpha=1, label='Customers', color = 'blue')
         plt.hist(df2[subscriber].duration_min, bins, alpha=0.5, label='Subscribers')
         plt.legend(loc='upper right')
         plt.title('Duration Distributions by User Type', fontsize=14, weight='bold')
         plt.show()
```



The histogram that shows the duration of subscribers' and customers' users' sessions. Perhaps due to the higher rates per minute, customers make up a considerably smaller portion of the duration distribution. The bulk of subscribers, however, used the bike for 0 to 20 minutes.

12. What connection exists between different user categories and the workweek?

```
In [45]: days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
         plt.figure(figsize = [14,6])
         sb.countplot(data = df2, x = 'start_day', hue = 'user_type', palette=['#FFFF00', '#FF00
         ax.legend(ncol = 2)
         plt.title('user categories and the workweek' , fontsize=14, weight='bold')
         plt.xticks(np.arange(0, 6+1, 1), days)
         plt.show()
```



Assessing the days of the week that bike service users and subscribers use the service the most. Interestingly, subscribers are most active on Monday, Tuesday, and Wednesday, whereas customers are most active on Monday and Sunday.

1.6.1 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?

It is challenging to compare subscribers and customers side by side without considering proportions because there is a significant variation between them. For example, men are more prevalent in both user types than females.

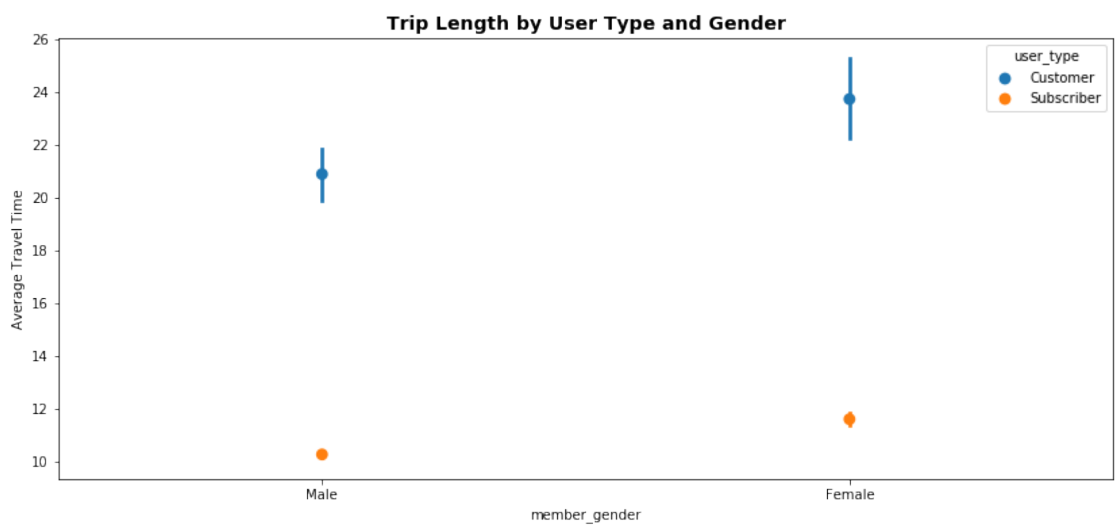
1.6.2 Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

When I include the additional variable of user types, the univariate plot of the count of bike rides during the week looks different.

1.7 Multivariate Exploration

13. Average journey length for many people, both male and female

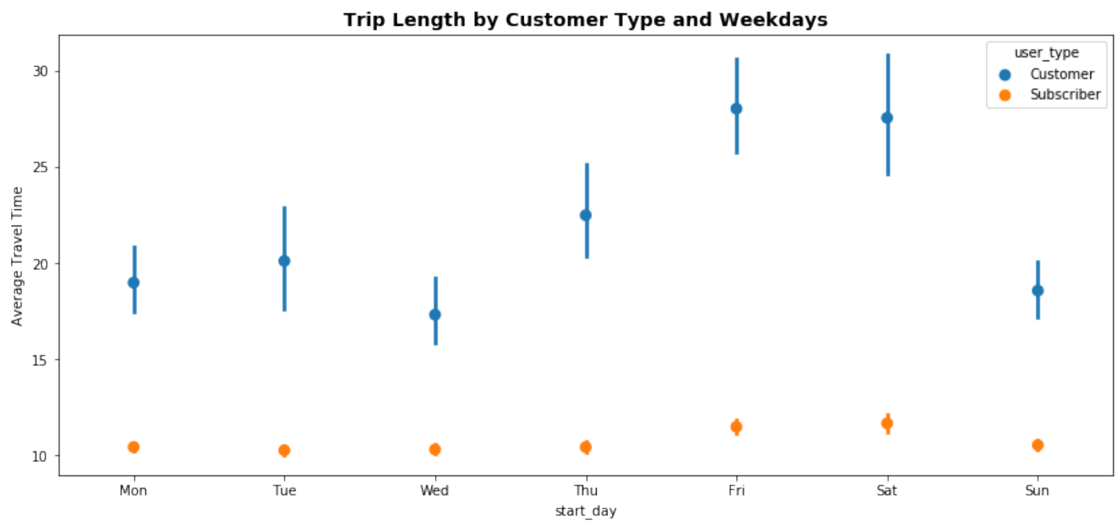
In [47]:
```
fig = plt.figure(figsize = [14,6])
ax = sb.pointplot(data = df2, x ='member_gender', y = 'duration_min', hue = 'user_type'
plt.title('Trip Length by User Type and Gender', fontsize=14, weight='bold')
plt.ylabel('Average Travel Time')
plt.show();
```



More women than men often use the bikes over time. Additionally, they have a wider range of averages.

13. Average journey length for various users on different days of the week

```
In [48]: fig = plt.figure(figsize = [14,6])
         ax = sb.pointplot(data = df2, x ='start_day', y = 'duration_min', hue = 'user_type', li
         plt.title('Trip Length by Customer Type and Weekdays', fontsize=14, weight='bold')
         plt.ylabel('Average Travel Time')
         plt.xticks(np.arange(0, 6+1, 1), days)
         ax.set_yticklabels([],minor = True)
         plt.show();
```



For both user types, Friday and Saturday are the busiest days. Sunday has the third-highest number of subscribers, while Thursday has the third-highest number of customers.

1.7.1 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?

Customers' average ride times are longer than subscribers', and females ride for longer on average.

1.7.2 Were there any interesting or surprising interactions between features?

Yeah! The average ride time for customers appears to be longer than for subscribers.

1.8 Conclusions

- Each of these graphs supports the basic idea that while subscribers frequently use them to commute to work or school, clients are more likely to be travelers, tourists, or occasional users.
- The usual hours were skewed to 7 am and 7 pm because subscribers have a significantly bigger user base. It would be fascinating to observe the difference if a study were done on the number of hours users have utilized them.

- Customers' bike trips typically last longer than Subscribers' do.
- About 90.5% of the trips were taken by subscribers.
- Men make up about three-quarters of the users.