Importing libraries

Definition - Imorting Pandas, Numpy, Matplotlib, Seaborn, Glob and OS, for accessing, assesing, cleaning, Engineering, Analysing and Visualising Given Data.

In [88]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import glob
import os

%matplotlib inline
```

Gathering Data

Definition - Reading file from cloud stored csv file using file drive path . **Function Used** : pandas.read_csv()

In [89]:

```
df = pd.read_csv('/content/drive/MyDrive/Utkarsh doc/201902-fordgobike-tripdata.
csv')
```

Assessing Data

Definition - Assessing given Raw database using functions such as:

- dataframe.head()
- dataframe.tail()
- dataframe.shape - It is an attribute not a function.
- dataframe.info()
- dataframe.dtypes - It is an attribute not a function.
- dataframe.describe() - It is an attribute not a function.

In [90]:

```
df.head()
```

Out[90]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_
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)	37
1	42521	2019-02-28 18:53:21.7890	2019-03-01 06:42:03.0560	23.0	The Embarcadero at Steuart St	37
2	61854	2019-02-28 12:13:13.2180	2019-03-01 05:24:08.1460	86.0	Market St at Dolores St	37
3	36490	2019-02-28 17:54:26.0100	2019-03-01 04:02:36.8420	375.0	Grove St at Masonic Ave	37
4	1585	2019-02-28 23:54:18.5490	2019-03-01 00:20:44.0740	7.0	Frank H Ogawa Plaza	37

In [91]:

```
df.tail()
```

Out[91]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_s
183407	480	2019-02-01 00:04:49.7240	2019-02-01 00:12:50.0340	27.0	Beale St at Harrison St	
183408	313	2019-02-01 00:05:34.7440	2019-02-01 00:10:48.5020	21.0	Montgomery St BART Station (Market St at 2nd St)	
183409	141	2019-02-01 00:06:05.5490	2019-02-01 00:08:27.2200	278.0	The Alameda at Bush St	
183410	139	2019-02-01 00:05:34.3600	2019-02-01 00:07:54.2870	220.0	San Pablo Ave at MLK Jr Way	
183411	271	2019-02-01 00:00:20.6360	2019-02-01 00:04:52.0580	24.0	Spear St at Folsom St	

In [92]:

```
df.shape
```

Out[92]:

```
(183412, 16)
```

In [93]:

```
df.dtypes
```

Out[93]:

```
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
```

In [94]:

```
df.describe()
```

Out[94]:

	duration_sec	start_station_id	start_station_latitude	start_station_longitude	end_statio
count	183412.000000	183215.000000	183412.000000	183412.000000	183215.00(
mean	726.078435	138.590427	37.771223	-122.352664	136.24(
std	1794.389780	111.778864	0.099581	0.117097	111.51!
min	61.000000	3.000000	37.317298	-122.453704	3.00(
25%	325.000000	47.000000	37.770083	-122.412408	44.00(
50%	514.000000	104.000000	37.780760	-122.398285	100.00(
75%	796.000000	239.000000	37.797280	-122.286533	235.00(
max	85444.000000	398.000000	37.880222	-121.874119	398.00(

What is the structure of your dataset?

The dataset includes 183,412 trips 'rows' with 16 features 'columns'. Out of 16 features, seven are float64, two are int64, and seven objects. Also, start and end time have the wrong datatype(string instead of date-time object).

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

According to me, the main features are the duration of the trip, origination and conclusion of trips, the user-type and most used stations.

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

- duration_sec
- start_time
- end_time
- user_type
- start_station_name
- end_station_name

Cleaning Data

1.Definition - Making a copy of provided dataframe to retain original values for future comparision.

Code

In [95]:

```
df_copy = df.copy()
```

2. Definition - Start and End times of trips are given in String format and thus need to be changed into Date-time object as it would further facilitate the calsulations and visualisation process.

Code

In [96]:

```
#changing datatype of 'start_time' & 'end_time' into datetime object
df.start_time = pd.to_datetime(df.start_time)
df.end_time = pd.to_datetime(df.end_time)
```

testing acquired Changes

In [97]:

```
df.dtypes
```

Out[97]:

```
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
```

3.Definition - Ahead, Checking for data redundancy and duplicacy in particular.

Code

In [98]:

```
#check if any rows are duplicated
sum(df.duplicated())
```

Out[98]:

0

Test Result -

No Duplicacy Found

4.Definition - Extracting Hours and minutes from Date-Time object("start_time" and "end_time") into new added respectible columns for further extraction of data into subpart for reducing complexity and better visualization.

Code

In [99]:

```
#extract the hours from start time
df['start_time_hours']=df['start_time'].dt.hour
#extract the minute from start time
df['start_time_minutes']=df['start_time'].dt.minute + df['start_time_hours']*60
```

In [100]:

```
#extract the hours from end time
df['end_time_hours']=df['end_time'].dt.hour
#extract the minute from end time
df['end_time_minutes']=df['end_time'].dt.minute + df['start_time_hours']*60
```

Testing above made changes

In [101]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 183412 entries, 0 to 183411
Data columns (total 20 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  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
 16  start_time_hours         183412 non-null  int64
 17  start_time_minutes       183412 non-null  int64
 18  end_time_hours           183412 non-null  int64
 19  end_time_minutes         183412 non-null  int64
dtypes: datetime64[ns](2), float64(7), int64(6), object(5)
memory usage: 28.0+ MB
```

5.Definition - Finally Extracting Month Number from given timestamps.

(We do not need to extract Year as this data in collected from 2019 only.)

Code

In [102]:

```
df['month']=df.start_time.dt.month
```

Testing above made changes

In [103]:

```
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  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
 16  start_time_hours         183412 non-null  int64
 17  start_time_minutes       183412 non-null  int64
 18  end_time_hours           183412 non-null  int64
 19  end_time_minutes         183412 non-null  int64
 20  month                    183412 non-null  int64
dtypes: datetime64[ns](2), float64(7), int64(7), object(5)
memory usage: 29.4+ MB
```

6.Definition -

- Dropping irrelevant columns from our Data Base.
- Dropping Rows with Null values for removing all NAN values from our database.

Code

In [104]:

```
#dropping irrelevant columns
df.drop(['start_station_latitude', 'start_station_longitude','end_station_latitu
de', 'end_station_longitude'] ,axis =1 , inplace = True)
```

In [105]:

```
df=df.dropna()
```

Testing above made alteration.

In [106]:

```
df.isna().sum()
```

Out[106]:

```
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
bike_id                    0
user_type                  0
member_birth_year          0
member_gender              0
bike_share_for_all_trip    0
start_time_hours           0
start_time_minutes         0
end_time_hours             0
end_time_minutes           0
month                      0
dtype: int64
```

Storing Data

Saving Engineered Database in a csv file

In [107]:

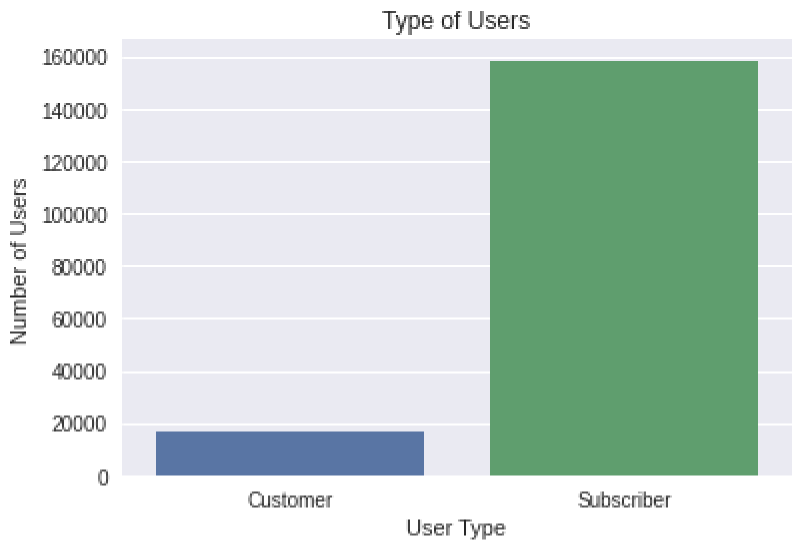
```
df.to_csv("fordbike_engineered.csv")
```

Analyzing Data

Univariate Exploration

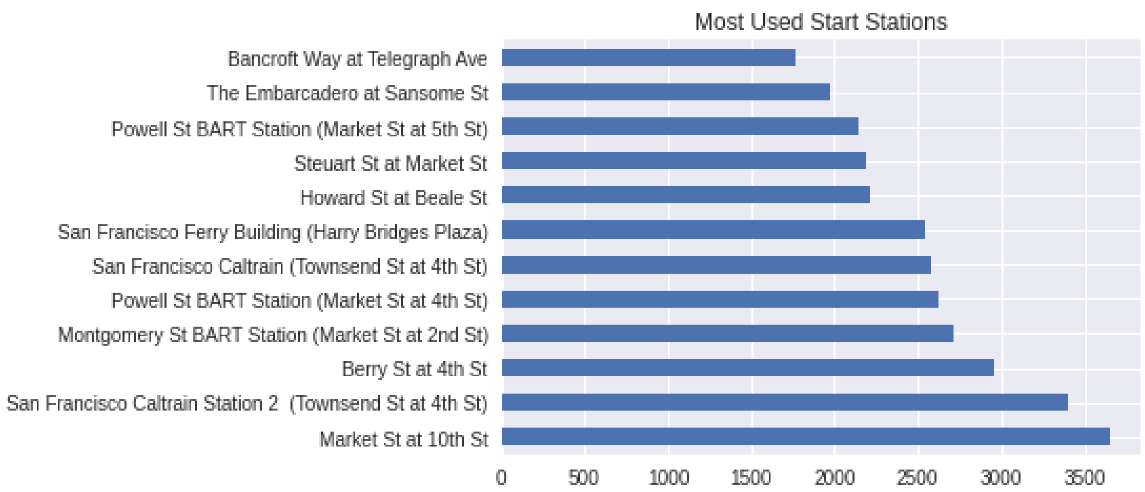
In [108]:

```
#BoxPlot representing User Type : Customer/Subscriber
sb.countplot(data=df,x='user_type')
plt.title("Type of Users")
plt.xlabel('User Type')
plt.ylabel('Number of Users')
plt.show()
```



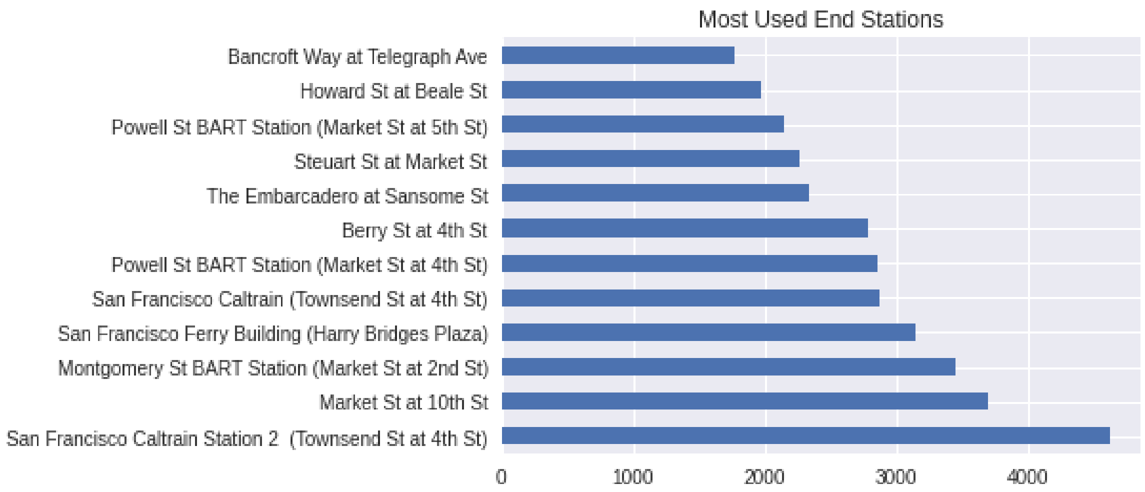
In [109]:

```
#A Barchart for depicting journey origination frquency of all stations.
df.start_station_name.value_counts()[:12].plot(kind='barh')
plt.title('Most Used Start Stations')
plt.show()
```



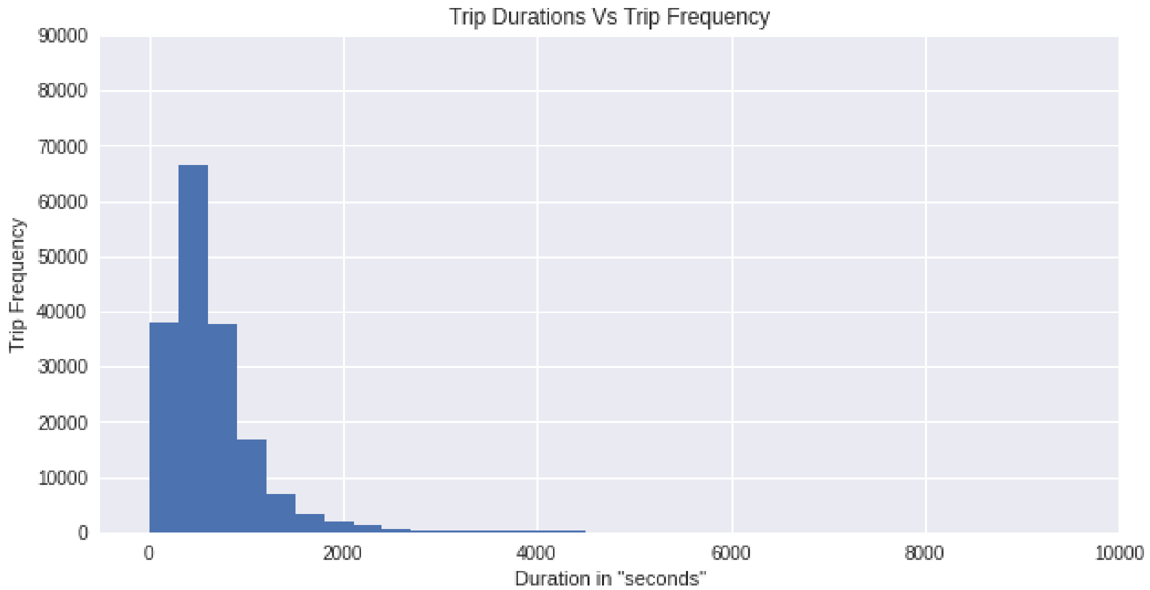
In [110]:

```
#A Barchart for depicting journey conclusion frquency of all stations.
df.end_station_name.value_counts()[:12].plot(kind='barh')
plt.title('Most Used End Stations')
plt.show()
```



In [111]:

```
#Histogram plot for plotting Frequency Vs time duration in seconds
plt.figure(figsize=(10,5))
duration_bins = np.arange(1,df.duration_sec.max()+300,300)
plt.hist(data=df,x='duration_sec',bins=duration_bins)
plt.title('Trip Durations Vs Trip Frequency')
plt.xlabel('Duration in "seconds"')
plt.ylabel('Trip Frequency')
plt.axis([-500, 10000, 0, 90000])
plt.show()
```



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

The graph shows the most common trip durationz take 0-2000 seconds. The number of subscribers are significantly higher than customers in User-types. also, Market st at 10th street is the most used as trip origination station. And, San Francisco Caltrain Station 2 is most used as trip conclusion station.

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?

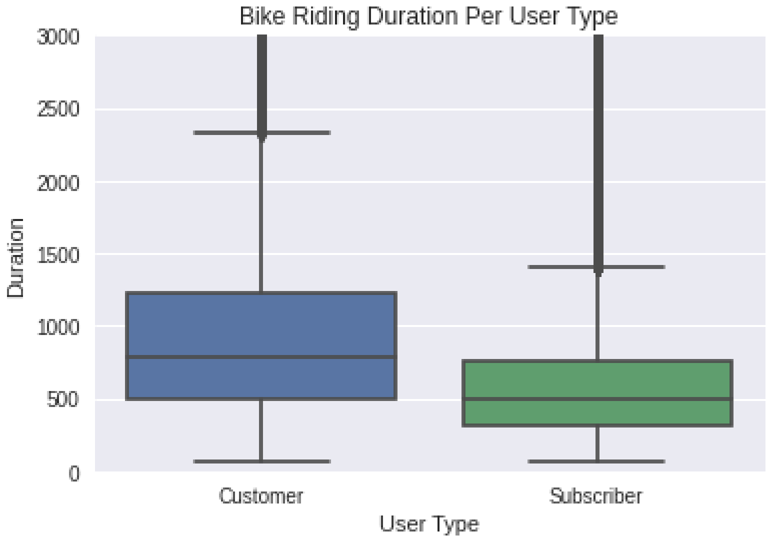
YES! There were some data redundancy and distribution issues in the used dataset and needed some cleaning. The problems adressed are as follows:

- Start and End times of trips are given in String format and thus need to be changed into Date-time object as it would further facilitate the calsulations and visualisation process.
- Extracting Hours and minutes from Date-Time object("start_time" and "end_time") into new added respectible columns for further extraction of data into subpart for reducing complexity and better visualization.
- Extracting Month Number from given timestamps. (We do not need to extract Year as this data in collected from 2019 only.)
- Dropping irrelevant columns from our Data Base.
- Dropping Rows with Null values for removing all NAN values from our database.

Bivariate Exploration

In [112]:

```
# Boxplot showing trip duration per user type, ie; Customer/Subscriber
sb.boxplot(data=df, x='user_type', y='duration_sec')
plt.ylim(0,3000)
plt.title("Bike Riding Duration Per User Type")
plt.xlabel('User Type')
plt.ylabel('Duration')
plt.show()
```



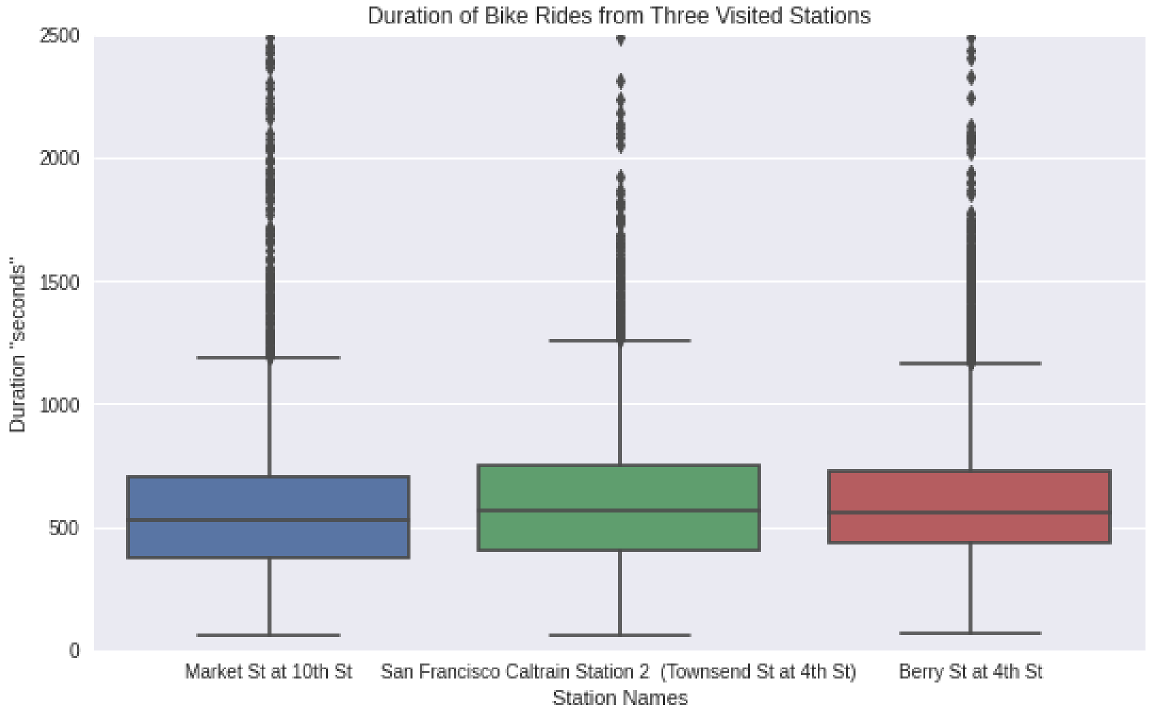
In [113]:

```
#boxplot for trip duration analysis of journeys from top three most used station
s.
start=df['start_station_name'].value_counts().index[:3]
s_stations = df.loc[df['start_station_name'].isin(start)]

plt.figure(figsize=(10,6))
sb.boxplot(data = s_stations, x='start_station_name', y='duration_sec')
plt.ylim(0, 2500)
plt.style.use('seaborn')
plt.title('Duration of Bike Rides from Three Visited Stations')
plt.xlabel('Station Names')
plt.ylabel('Duration "seconds"')
```

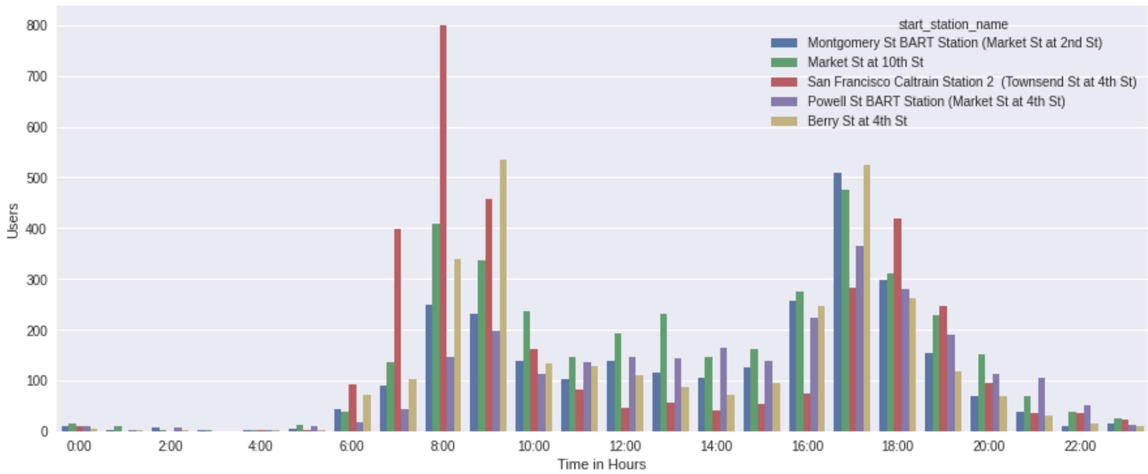
Out[113]:

```
Text(0, 0.5, 'Duration "seconds"')
```



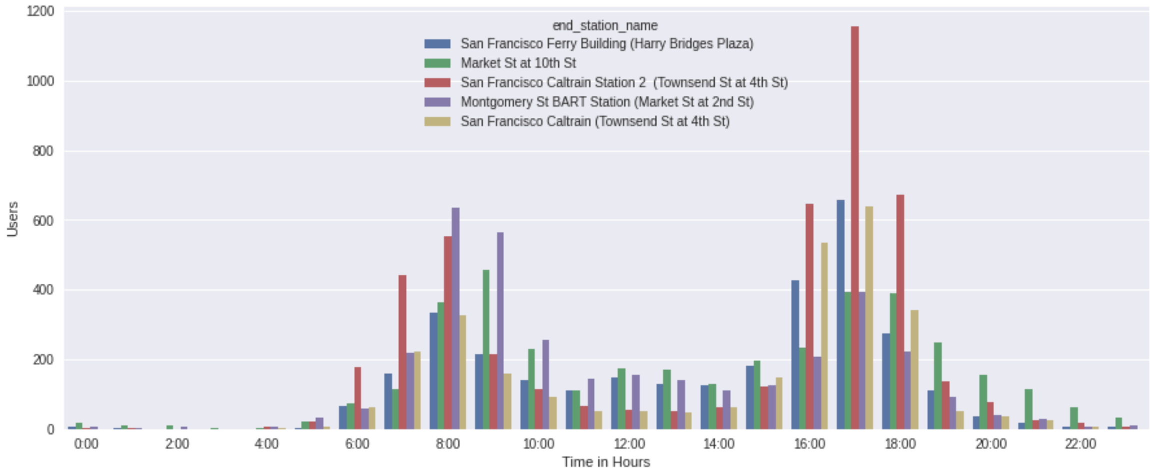
In [114]:

```
start=df['start_station_name'].value_counts().index[:5]
s_stations = df.loc[df['start_station_name'].isin(start)]
plt.figure(figsize = (15,6))
sb.countplot(data=s_stations, x='start_time_hours', hue='start_station_name')
x_tick= np.arange(0,24,2)
x_label= [str(x)+":00" for x in x_tick]
plt.xticks(x_tick, x_label)
plt.xlabel('Time in Hours')
plt.ylabel('Users')
plt.show()
```



In [115]:

```
end= df['end_station_name'].value_counts().index[:5]
e_stations = df.loc[df['end_station_name'].isin(end)]
plt.figure(figsize = (15,6))
sb.countplot(data=e_stations, x='end_time_hours', hue='end_station_name')
x_tick= np.arange(0,24,2)
x_label= [str(x)+":00" for x in x_tick]
plt.xticks(x_tick, x_label)
plt.xlabel('Time in Hours')
plt.ylabel('Users')
plt.show()
```



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?

First, the duration per type of user shows that the customers average duration is higher than the subscribers. While as seen in the univariate plot the subscribers bike ride count is higher than the customers bike ride count. Also, the start station and end station does not much determine the duration. It suggests that some starting stations are having higher visited as the starting point and some end stations are having higher visited as the ending point.

Were there any interesting or surprising interactions between features?

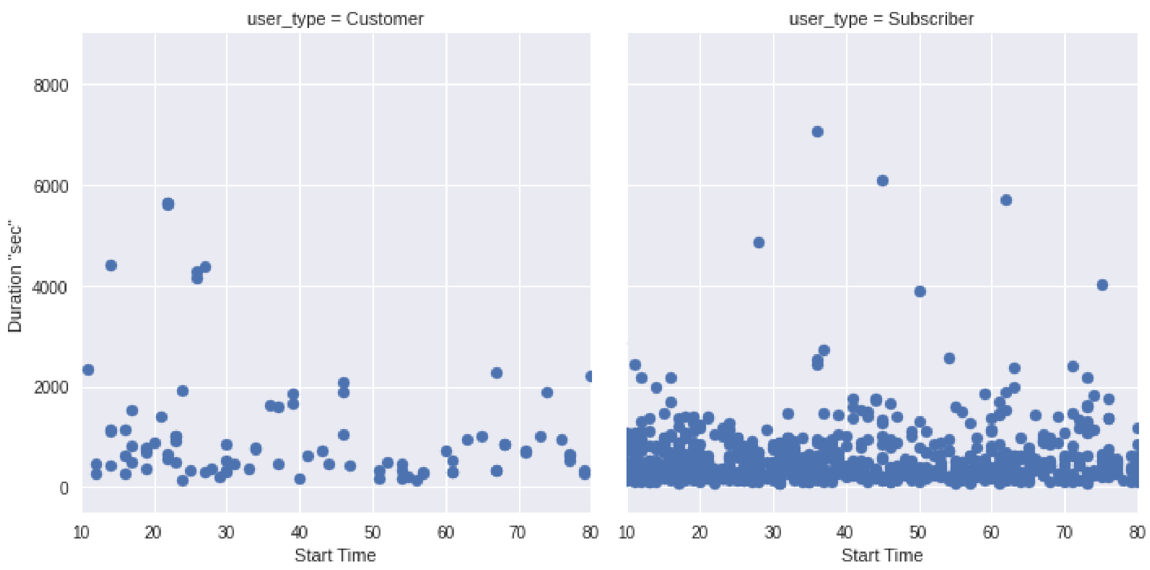
Graphs suggest, bigger number of users prefer ride bike in the morning at 8. On the other hand, the users end the trip usually at 5 pm.

Multivariate Exploration

In [116]:

```
# Scatterplot depicting time duration trends for both user types.
ut=sb.FacetGrid(data = df, col = 'user_type', col_wrap = 2, size = 5,
                 xlim = [10, 80], ylim = [-500, 9000])
ut.map(plt.scatter, 'start_time_minutes', 'duration_sec')
ut.set_xlabels('Start Time')
ut.set_ylabels('Duration "sec"')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:337: User
Warning: The `size` parameter has been renamed to `height`; please u
pdate your code.
  warnings.warn(msg, UserWarning)
```



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

The graph shows the subscribers generally ride for a shorter time than customers who ride for a longer time during the period from 8 am. **We can clearly see few outlier values in both cases.**