Part I - (Ford GoBike System Findings)

by (**PURITY NJOKI NG'ANG'A**)

Introduction This data set includes information about individual rides made in a bike-sharing system covering the greater San Francisco Bay area.

Preliminary Wrangling

In [1]:
```
# importing necessary libraries

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

In [2]:
```
# unzip the files to access the datasets needed for analysis
with zipfile.ZipFile('jan-baywheels.zip', 'r') as myzip1:
    myzip1.extractall()

with zipfile.ZipFile('feb-baywheels.zip', 'r') as myzip2:
    myzip2.extractall()

with zipfile.ZipFile('march-baywheels.zip', 'r') as myzip3:
    myzip3.extractall()
```

In [3]:
```
#load datasets into pandas dataframe

jan_baywhls = pd.read_csv('202001-baywheels-tripdata.csv')
feb_baywhls = pd.read_csv('202002-baywheels-tripdata.csv')
mar_baywhls = pd.read_csv('202003-baywheels-tripdata.csv')
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_4872\429148203.py:3: DtypeWarning: Columns (1
3) have mixed types. Specify dtype option on import or set low_memory=False.
  jan_baywhls = pd.read_csv('202001-baywheels-tripdata.csv')
C:\Users\User\AppData\Local\Temp\ipykernel_4872\429148203.py:4: DtypeWarning: Columns (1
3) have mixed types. Specify dtype option on import or set low_memory=False.
  feb_baywhls = pd.read_csv('202002-baywheels-tripdata.csv')
```

In [4]:
```
#save datasets to csv files and;
#load the saved dataset into pandas dataframe

jan_baywhls.to_csv('ford_gobike1_2020.csv', index = False)
df1 = pd.read_csv('ford_gobike1_2020.csv', dtype={"rental_access_method" : "object"})

feb_baywhls.to_csv('ford_gobike2_2020.csv', index = False)
df2 = pd.read_csv('ford_gobike2_2020.csv', dtype={"rental_access_method" : "object"})

mar_baywhls.to_csv('ford_gobike3_2020.csv', index = False)
df3 = pd.read_csv('ford_gobike3_2020.csv', dtype={"rental_access_method" : "object"})
```

In [5]:
```
df1.head(5)
```

Out[5]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_statio
0	83118	2020-01-31 15:23:47.7330	2020-02-01 14:29:06.2630	400.0	Buchanan St at North Point St	37.804272	

1	68239	2020-01-31 15:40:31.6160	2020-02-01 10:37:51.0000	99.0	Folsom St at 15th St	37.767037
2	55407	2020-01-31 17:48:16.6350	2020-02-01 09:11:44.3170	197.0	El Embarcadero at Grand Ave	37.808848
3	54929	2020-01-31 17:53:03.4130	2020-02-01 09:08:32.6340	197.0	El Embarcadero at Grand Ave	37.808848
4	55700	2020-01-31 17:12:33.4600	2020-02-01 08:40:53.6460	12.0	Pier 1/2 at The Embarcadero	37.796389

In [6]:
```
df1.shape
```

Out[6]:
```
(295854, 14)
```

In [7]:
```
df1.sample(10)
```

Out[7]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_statior
194909	406	2020-01-30 16:21:42	2020-01-30 16:28:28	NaN	NaN	37.775023	-
198687	538	2020-01-06 09:07:03	2020-01-06 09:16:01	81.0	Berry St at 4th St	37.775880	-
123787	735	2020-01-04 19:34:04	2020-01-04 19:46:19	NaN	NaN	37.338660	-
178857	996	2020-01-10 17:07:42	2020-01-10 17:24:19	93.0	4th St at Mission Bay Blvd S	37.770407	-
190203	1144	2020-01-26 15:02:04	2020-01-26 15:21:08	NaN	NaN	37.773572	-
284139	720	2020-01-24 17:04:25	2020-01-24 17:16:26	NaN	NaN	37.798697	-
118925	289	2020-01-20 08:03:19	2020-01-20 08:08:09	NaN	NaN	37.334211	-
178730	477	2020-01-31 13:00:12	2020-01-31 13:08:09	93.0	4th St at Mission Bay Blvd S	37.770407	-
188541	621	2020-01-17 15:46:13	2020-01-17 15:56:34	NaN	NaN	37.773230	-
252955	655	2020-01-27 05:32:50	2020-01-27 05:43:45	NaN	NaN	37.788512	-

In [8]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 295854 entries, 0 to 295853
Data columns (total 14 columns):
 #   Column                   Non-Null Count   Dtype
```

```
---  ------                    --------------   -----
 0   duration_sec              295854 non-null  int64
 1   start_time                295854 non-null  object
 2   end_time                  295854 non-null  object
 3   start_station_id          146286 non-null  float64
 4   start_station_name        146866 non-null  object
 5   start_station_latitude    295854 non-null  float64
 6   start_station_longitude   295854 non-null  float64
 7   end_station_id            145934 non-null  float64
 8   end_station_name          146511 non-null  object
 9   end_station_latitude      295854 non-null  float64
 10  end_station_longitude     295854 non-null  float64
 11  bike_id                   295854 non-null  int64
 12  user_type                 295854 non-null  object
 13  rental_access_method      185746 non-null  object
dtypes: float64(6), int64(2), object(6)
memory usage: 31.6+ MB
```

In [9]:
```
df1.describe() # to get descriptives stats
```

Out[9]:

	duration_sec	start_station_id	start_station_latitude	start_station_longitude	end_station_id	end_station_lat
count	295854.000000	146286.000000	295854.000000	295854.000000	145934.000000	295854.0
mean	780.473193	168.900879	37.751277	-122.357866	161.061788	37.7
std	2037.786317	141.956388	0.228337	0.650796	140.766154	0.2
min	60.000000	3.000000	0.000000	-122.513871	3.000000	0.0
25%	369.000000	53.000000	37.766214	-122.419481	43.000000	37.7
50%	588.000000	120.000000	37.777158	-122.403234	113.000000	37.7
75%	912.000000	263.000000	37.790102	-122.392450	250.000000	37.7
max	811077.000000	506.000000	37.880222	0.000000	506.000000	37.9

In [10]:
```
df1.isna().sum() #finding if there are any missing values
```

Out[10]:
```
duration_sec                  0
start_time                    0
end_time                      0
start_station_id         149568
start_station_name       148988
start_station_latitude        0
start_station_longitude       0
end_station_id           149920
end_station_name         149343
end_station_latitude          0
end_station_longitude         0
bike_id                       0
user_type                     0
rental_access_method     110108
dtype: int64
```

In [11]:
```
df1.nunique() #number of unique entries
```

Out[11]:
```
duration_sec                5574
start_time                282787
end_time                  282613
start_station_id             429
start_station_name           430
start_station_latitude    122218
start_station_longitude   123962
end_station_id               428
end_station_name             429
```

```
end_station_latitude       122941
end_station_longitude      125091
bike_id                      8016
user_type                       2
rental_access_method            2
dtype: int64
```

In [12]:
```
print(df1['user_type'].unique())
df1['user_type'].value_counts()
```

Out[12]:
```
['Customer' 'Subscriber']
Subscriber    170988
Customer      124866
Name: user_type, dtype: int64
```

In [13]:
```
#getting unique categories of the rental_access_method
print(df1['rental_access_method'].unique())
df1.rental_access_method.value_counts()
```

Out[13]:
```
[nan 'app' 'clipper']
app        171751
clipper     13995
Name: rental_access_method, dtype: int64
```

In [14]:
```
df1[df1.duration_sec <= 60].count() # getting the minimum number of duration
```

Out[14]:
```
duration_sec               54
start_time                 54
end_time                   54
start_station_id           12
start_station_name         12
start_station_latitude     54
start_station_longitude    54
end_station_id             10
end_station_name           11
end_station_latitude       54
end_station_longitude      54
bike_id                    54
user_type                  54
rental_access_method       54
dtype: int64
```

In [15]:
```
df1.nlargest(5, 'duration_sec') # extracting 5 largest values of the time duration
# seems like there's one outlier
```

Out[15]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_
121168	811077	2020-01-09 16:23:21	2020-01-19 01:41:19	NaN	NaN	37.336035	
49926	86221	2020-01-16 07:51:37.7090	2020-01-17 07:48:39.5150	417.0	Park Ave at Race St	37.326011	
63677	85994	2020-01-12 17:08:54.1100	2020-01-13 17:02:08.5300	284.0	Yerba Buena Center for the Arts (Howard St at ...	37.784872	
27024	85450	2020-01-23 14:29:54.6580	2020-01-24 14:14:05.0000	14.0	Clay St at Battery St	37.795001	
50984	85268	2020-01-15 18:13:47.2190	2020-01-16 17:54:55.2520	258.0	University Ave at Oxford St	37.872355	

What is the structure of your dataset? There are 295854 observations and 14 variables(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 and rental_access_method

What is/are the main feature(s) of interest in your dataset? Most interest is in figuring out features that best determine least time taken for each trip in the dataset.

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

I expect that duration will have strongest effect on in determining the time spent on each trip taken. Other features that influence time taken could be day of the week, month or user type.

Assesing the second dataframe

In [16]:
```
df2.head(10)
```

Out[16]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_statio
0	62083	2020-02-29 18:32:30.5750	2020-03-01 11:47:14.0850	176.0	MacArthur BART Station	37.828410	
1	1364	2020-02-29 23:53:53.7190	2020-03-01 00:16:37.9720	375.0	Grove St at Masonic Ave	37.774836	
2	836	2020-02-29 23:54:03.1970	2020-03-01 00:07:59.8490	375.0	Grove St at Masonic Ave	37.774836	
3	1004	2020-02-29 23:48:34.6480	2020-03-01 00:05:19.2020	179.0	Telegraph Ave at 27th St	37.816073	
4	1007	2020-02-29 23:48:25.9000	2020-03-01 00:05:13.4490	179.0	Telegraph Ave at 27th St	37.816073	
5	338	2020-02-29 23:57:43.5250	2020-03-01 00:03:22.4400	182.0	19th Street BART Station	37.809369	
6	570	2020-02-29 23:52:37.7100	2020-03-01 00:02:08.3470	252.0	Channing Way at Shattuck Ave	37.865847	
7	1001	2020-02-29 23:36:02.9230	2020-02-29 23:52:44.6450	5.0	Powell St BART Station (Market St at 5th St)	37.783899	
8	3247	2020-02-29 22:58:00.6150	2020-02-29 23:52:08.3760	246.0	Berkeley Civic Center	37.869060	
9	898	2020-02-29 23:35:54.7540	2020-02-29 23:50:53.7240	95.0	Sanchez St at 15th St	37.766219	

In [17]:
```
df2.shape
```

Out[17]:
```
(432354, 14)
```

In [18]:
```
df1.sample(10)
```

Out[18]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_
208696	355	2020-01-10 23:55:09	2020-01-11 00:01:05	NaN	NaN	37.776756	
195483	424	2020-01-07	2020-01-07	NaN	NaN	37.775175	

		20:05:49	20:12:53			
49029	985	2020-01-17 09:40:40.2820	2020-01-17 09:57:05.9780	121.0	Mission Playground	37.759210
242633	511	2020-01-27 09:17:33	2020-01-27 09:26:04	NaN	NaN	37.786279
37767	310	2020-01-21 18:16:31.8420	2020-01-21 18:21:42.5470	81.0	Berry St at 4th St	37.775880
68528	428	2020-01-12 11:31:02.2490	2020-01-12 11:38:10.7820	170.0	Telegraph Ave at 58th St	37.844493
26040	375	2020-01-24 17:50:06.5220	2020-01-24 17:56:21.5540	114.0	Rhode Island St at 17th St	37.764478
102157	442	2020-01-03 17:41:09.4920	2020-01-03 17:48:32.1130	182.0	19th Street BART Station	37.809369
122522	650	2020-01-06 14:12:50	2020-01-06 14:23:41	NaN	NaN	37.337091
271663	171	2020-01-27 18:48:34	2020-01-27 18:51:26	NaN	NaN	37.793551

In [19]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 432354 entries, 0 to 432353
Data columns (total 14 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   duration_sec             432354 non-null  int64  
 1   start_time               432354 non-null  object 
 2   end_time                 432354 non-null  object 
 3   start_station_id         161723 non-null  float64
 4   start_station_name       161723 non-null  object 
 5   start_station_latitude   432354 non-null  float64
 6   start_station_longitude  432354 non-null  float64
 7   end_station_id           162194 non-null  float64
 8   end_station_name         162194 non-null  object 
 9   end_station_latitude     432354 non-null  float64
 10  end_station_longitude    432354 non-null  float64
 11  bike_id                  432354 non-null  int64  
 12  user_type                432354 non-null  object 
 13  rental_access_method     317843 non-null  object 
dtypes: float64(6), int64(2), object(6)
memory usage: 46.2+ MB
```

In [20]:
```
df2.describe() # getting descriptive statistics of the dataset
```

Out[20]:

	duration_sec	start_station_id	start_station_latitude	start_station_longitude	end_station_id	end_station_lat
count	432354.000000	161723.000000	432354.000000	432354.000000	162194.000000	432354.0
mean	802.375502	174.885601	37.752000	-122.363239	167.831301	37.7
std	1383.128099	144.087616	0.249644	0.733128	143.653096	0.2
min	60.000000	3.000000	0.000000	-122.514230	3.000000	0.0
25%	369.000000	56.000000	37.765910	-122.421264	44.000000	37.7
50%	596.000000	126.000000	37.777288	-122.405528	120.000000	37.7

75%	934.750000	268.000000	37.790102	-122.393572	258.000000	37.7
max	86317.000000	512.000000	37.880222	0.000000	512.000000	37.8

In [21]:
```
df2[df2.duration_sec <= 60].count() #getting th number of entries with duration points l
```

Out[21]:
```
duration_sec               100
start_time                 100
end_time                   100
start_station_id            16
start_station_name          16
start_station_latitude     100
start_station_longitude    100
end_station_id              14
end_station_name            14
end_station_latitude       100
end_station_longitude      100
bike_id                    100
user_type                  100
rental_access_method       100
dtype: int64
```

In [22]:
```
df2.nsmallest(5, 'duration_sec') #extracting smallest duration number to find if there a
```

Out[22]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_statior
119217	60	2020-02-26 11:12:31	2020-02-26 11:13:31	296.0	5th St at Virginia St	37.325998	-
122497	60	2020-02-07 15:19:12	2020-02-07 15:20:13	NaN	NaN	37.331160	-
124418	60	2020-02-09 18:03:10	2020-02-09 18:04:11	NaN	NaN	37.332897	-
124746	60	2020-02-29 17:21:04	2020-02-29 17:22:05	NaN	NaN	37.333003	-
124857	60	2020-02-18 20:59:04	2020-02-18 21:00:04	NaN	NaN	37.333054	-

In [23]:
```
df2.nlargest(10, 'duration_sec') #finding if there're any outliers
```

Out[23]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_s
51618	86317	2020-02-17 11:42:26.0590	2020-02-18 11:41:03.7410	295.0	William St at 10th St	37.332794	
91130	86239	2020-02-06 17:03:46.4380	2020-02-07 17:01:05.7030	486.0	Arguello Blvd at Edward St	37.778487	
82154	84805	2020-02-09 16:37:18.0310	2020-02-10 16:10:43.9410	115.0	Jackson Playground	37.765026	
38157	84627	2020-02-20 10:40:59.1810	2020-02-21 10:11:26.2040	87.0	Folsom St at 13th St	37.769757	
66401	84252	2020-02-12 18:50:07.3260	2020-02-13 18:14:19.5050	8.0	The Embarcadero at Vallejo St	37.799953	
57284	84008	2020-02-15	2020-02-16	246.0	Berkeley Civic	37.869060	

		15:39:15.3380	14:59:24.1610		Center	
67124	83889	2020-02-12 17:34:09.6320	2020-02-13 16:52:19.3130	467.0	Brannan St at Colin P Kelly Jr St	37.782386
99658	83845	2020-02-04 17:41:24.5800	2020-02-05 16:58:50.3570	58.0	Market St at 10th St	37.776619
45960	83318	2020-02-18 18:07:42.7780	2020-02-19 17:16:21.2950	284.0	Yerba Buena Center for the Arts (Howard St at ...	37.784872
77832	83101	2020-02-10 13:08:06.4540	2020-02-11 12:13:08.1690	263.0	Channing Way at San Pablo Ave	37.862827

In [24]:
```
print(df2.user_type.unique()) #finding unique categories in the user_type var
df2.user_type.value_counts()
```

Out[24]:
```
['Customer' 'Subscriber']
Subscriber    277446
Customer      154908
Name: user_type, dtype: int64
```

In [25]:
```
print(df2.rental_access_method.unique())

df2.rental_access_method.value_counts()
```

Out[25]:
```
[nan 'app' 'clipper']
app        293205
clipper     24638
Name: rental_access_method, dtype: int64
```

What is the structure of your dataset?

There are 432354 observations and 14 variables(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 and rental_access_method

What is/are the main feature(s) of interest in your dataset? Most interest is in figuring out features that best determine least time taken for each trip in the dataset.

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

I expect that duration will have strongest effect in determining the time spent on each trip taken. Other features that influence time taken could be day of the week, month or user type.

(iii) Assessing the third dataset

In [26]:
```
df3.shape
```

Out[26]:
```
(176799, 14)
```

In [27]:
```
df3.head()
```

Out[27]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_statio
0	35187	2020-03-31 20:42:10.0790	2020-04-01 06:28:37.8440	462.0	Cruise Terminal at Pier 27	37.804648	

1	14568	2020-03-31 22:45:25.5010	2020-04-01 02:48:13.7730	42.0	San Francisco City Hall (Polk St at Grove St)	37.778650
2	35990	2020-03-31 15:08:22.3310	2020-04-01 01:08:12.9900	391.0	1st St at Younger Ave	37.355030
3	1068	2020-03-31 23:55:00.4260	2020-04-01 00:12:49.0200	456.0	Arguello Blvd at Geary Blvd	37.781468
4	3300	2020-03-31 23:00:55.6410	2020-03-31 23:55:56.6110	6.0	The Embarcadero at Sansome St	37.804770

In [28]:
```
df3.sample(5)
```

Out[28]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_
24569	1101	2020-03-10 10:34:02.5990	2020-03-10 10:52:24.4980	21.0	Montgomery St BART Station (Market St at 2nd St)	37.789625	
19256	810	2020-03-11 18:09:48.3900	2020-03-11 18:23:18.6650	104.0	4th St at 16th St	37.767045	
176383	536	2020-03-07 16:49:31	2020-03-07 16:58:27	NaN	NaN	37.807410	
100092	286	2020-03-11 19:09:13	2020-03-11 19:14:00	223.0	16th St Mission BART Station 2	37.764765	
103170	3033	2020-03-23 18:47:33	2020-03-23 19:38:06	95.0	Sanchez St at 15th St	37.766219	

In [29]:
```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 176799 entries, 0 to 176798
Data columns (total 14 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   duration_sec             176799 non-null  int64  
 1   start_time               176799 non-null  object 
 2   end_time                 176799 non-null  object 
 3   start_station_id         110696 non-null  float64
 4   start_station_name       111349 non-null  object 
 5   start_station_latitude   176799 non-null  float64
 6   start_station_longitude  176799 non-null  float64
 7   end_station_id           111097 non-null  float64
 8   end_station_name         111794 non-null  object 
 9   end_station_latitude     176799 non-null  float64
 10  end_station_longitude    176799 non-null  float64
 11  bike_id                  176799 non-null  int64  
 12  user_type                176799 non-null  object 
 13  rental_access_method     114269 non-null  object 
dtypes: float64(6), int64(2), object(6)
memory usage: 18.9+ MB
```

In [30]:
```
df3.isna().sum()
```

Out[30]:

```
duration_sec                 0
start_time                   0
end_time                     0
start_station_id         66103
start_station_name       65450
start_station_latitude       0
```

```
start_station_longitude       0
end_station_id              65702
end_station_name            65005
end_station_latitude            0
end_station_longitude           0
bike_id                         0
user_type                       0
rental_access_method        62530
dtype: int64
```

In [31]:
```
df3.describe()
```

Out[31]:

	duration_sec	start_station_id	start_station_latitude	start_station_longitude	end_station_id	end_station_lat
count	176799.000000	110696.000000	176799.000000	176799.000000	111097.000000	176799.0
mean	861.590156	185.082261	37.732506	-122.339595	179.244228	37.7
std	1528.143090	154.647521	0.170476	0.336403	153.858017	0.2
min	60.000000	3.000000	0.000000	-122.513814	3.000000	0.0
25%	374.000000	58.000000	37.763708	-122.421339	53.000000	37.7
50%	602.000000	125.000000	37.776598	-122.403969	121.000000	37.7
75%	963.000000	321.000000	37.789677	-122.390648	309.000000	37.7
max	84450.000000	521.000000	37.880222	0.000000	521.000000	37.9

In [32]:
```
df3[df3.duration_sec <= 60].count()
```

Out[32]:
```
duration_sec                29
start_time                  29
end_time                    29
start_station_id            10
start_station_name          10
start_station_latitude      29
start_station_longitude     29
end_station_id               6
end_station_name             6
end_station_latitude        29
end_station_longitude       29
bike_id                     29
user_type                   29
rental_access_method        29
dtype: int64
```

In [33]:
```
df3.nlargest(5, 'duration_sec') # extracting maximum durations
```

Out[33]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_s
7463	84450	2020-03-20 10:07:18.0910	2020-03-21 09:34:48.1470	84.0	Duboce Park	37.769200	
11311	82608	2020-03-15 13:14:31.1470	2020-03-16 12:11:19.8850	3.0	Powell St BART Station (Market St at 4th St)	37.786375	
43844	82449	2020-03-03 19:53:05.7190	2020-03-04 18:47:15.4680	508.0	St. Joseph's Ave at Geary Blvd	37.782476	
4392	81351	2020-03-24 11:45:56.7950	2020-03-25 10:21:47.8420	178.0	Broadway at 30th St	37.819381	
23142	80980	2020-03-09 19:19:48.0250	2020-03-10 17:49:28.1160	246.0	Berkeley Civic Center	37.869060	

In [34]:
```
print(df3.user_type.unique())
df3.user_type.value_counts()
```

Out[34]:
```
['Customer' 'Subscriber']
Subscriber    96262
Customer      80537
Name: user_type, dtype: int64
```

In [35]:
```
print(df3.rental_access_method.unique())
df3.rental_access_method.value_counts()
```

Out[35]:
```
[nan 'app' 'clipper']
app        105515
clipper      8754
Name: rental_access_method, dtype: int64
```

What is the structure of your dataset? There are 176799 observations and 14 variables(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 and rental_access_method

What is/are the main feature(s) of interest in your dataset? Most interest is in figuring out features that best determine least time taken for each trip in the dataset.

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

I expect that duration will have strongest effect in determining the time spent on each trip taken. Other features that influence time taken could be day of the week, month or user type.

Findings

(i) Tidiness Issues

1. Data is divided into 3 datasets.
2. 'Day' column is missing from the datasets
3. Time is displayed in term of seconds, instead of minutes or hours.
4. Unnecessary columns for analysis.
5. Month column is missing.

Quality

1. Wrong data types (start_time, end_time)
2. Null records present
3. Jan dataset has one outlier

Cleaning Data

In [36]:
```
# Creating dataframe copies

df1_clean = df1.copy()
df2_clean = df2.copy()
df3_clean = df3.copy()
```

T5. Month column is missing

Define:

Add month column. In each dataset add the month column, with the name of the month, to tell apart each dataset for ease of analysis.

Code:

In [105…
```
#creating the new column 'month'

df1_clean['month'] = 'Jan'
df2_clean['month'] = 'Feb'
df3_clean['month'] = 'Mar'
```

Test:

In [38]:
```
print(df1_clean.shape)
df1_clean.head()
```

```
(295854, 15)
```

Out[38]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_statio
0	83118	2020-01-31 15:23:47.7330	2020-02-01 14:29:06.2630	400.0	Buchanan St at North Point St	37.804272	
1	68239	2020-01-31 15:40:31.6160	2020-02-01 10:37:51.0000	99.0	Folsom St at 15th St	37.767037	
2	55407	2020-01-31 17:48:16.6350	2020-02-01 09:11:44.3170	197.0	El Embarcadero at Grand Ave	37.808848	
3	54929	2020-01-31 17:53:03.4130	2020-02-01 09:08:32.6340	197.0	El Embarcadero at Grand Ave	37.808848	
4	55700	2020-01-31 17:12:33.4600	2020-02-01 08:40:53.6460	12.0	Pier 1/2 at The Embarcadero	37.796389	

In [39]:
```
print(df2_clean.shape)
df2_clean.head()
```

```
(432354, 15)
```

Out[39]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_statio
0	62083	2020-02-29 18:32:30.5750	2020-03-01 11:47:14.0850	176.0	MacArthur BART Station	37.828410	
1	1364	2020-02-29 23:53:53.7190	2020-03-01 00:16:37.9720	375.0	Grove St at Masonic Ave	37.774836	

2	836	2020-02-29 23:54:03.1970	2020-03-01 00:07:59.8490	375.0	Grove St at Masonic Ave	37.774836	
3	1004	2020-02-29 23:48:34.6480	2020-03-01 00:05:19.2020	179.0	Telegraph Ave at 27th St	37.816073	
4	1007	2020-02-29 23:48:25.9000	2020-03-01 00:05:13.4490	179.0	Telegraph Ave at 27th St	37.816073	

In [40]:
```
print(df3_clean.shape) # checking if the new column has been added
df3_clean.head()
```

```
(176799, 15)
```

Out[40]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_statio
0	35187	2020-03-31 20:42:10.0790	2020-04-01 06:28:37.8440	462.0	Cruise Terminal at Pier 27	37.804648	
1	14568	2020-03-31 22:45:25.5010	2020-04-01 02:48:13.7730	42.0	San Francisco City Hall (Polk St at Grove St)	37.778650	
2	35990	2020-03-31 15:08:22.3310	2020-04-01 01:08:12.9900	391.0	1st St at Younger Ave	37.355030	
3	1068	2020-03-31 23:55:00.4260	2020-04-01 00:12:49.0200	456.0	Arguello Blvd at Geary Blvd	37.781468	
4	3300	2020-03-31 23:00:55.6410	2020-03-31 23:55:56.6110	6.0	The Embarcadero at Sansome St	37.804770	

T2. Data is divided into 3 datasets.

Define

Append the clean datasets to form one dataset.

Code

In [41]:
```
df = df1_clean.append(df2_clean, ignore_index = True)

df= df.append(df3_clean, ignore_index = True)
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_4872\3831234053.py:1: FutureWarning: The fram
e.append method is deprecated and will be removed from pandas in a future version. Use p
andas.concat instead.
  df = df1_clean.append(df2_clean, ignore_index = True)
C:\Users\User\AppData\Local\Temp\ipykernel_4872\3831234053.py:3: FutureWarning: The fram
e.append method is deprecated and will be removed from pandas in a future version. Use p
andas.concat instead.
  df= df.append(df3_clean, ignore_index = True)
```

Test

In [42]:
```
df.shape
```

Out[42]:
```
(905007, 15)
```

merged datadrame has the total number of rows from the indvidual datasets combined.

T3. Unnecessary columns

Define

Delete the unrequired columns from the dataset

Code

In [43]:
```
df.head(10)
```

Out[43]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_statio
0	83118	2020-01-31 15:23:47.7330	2020-02-01 14:29:06.2630	400.0	Buchanan St at North Point St	37.804272	
1	68239	2020-01-31 15:40:31.6160	2020-02-01 10:37:51.0000	99.0	Folsom St at 15th St	37.767037	
2	55407	2020-01-31 17:48:16.6350	2020-02-01 09:11:44.3170	197.0	El Embarcadero at Grand Ave	37.808848	
3	54929	2020-01-31 17:53:03.4130	2020-02-01 09:08:32.6340	197.0	El Embarcadero at Grand Ave	37.808848	
4	55700	2020-01-31 17:12:33.4600	2020-02-01 08:40:53.6460	12.0	Pier 1/2 at The Embarcadero	37.796389	
5	11333	2020-01-31 22:48:22.6390	2020-02-01 01:57:15.7160	60.0	8th St at Ringold St	37.774520	
6	11341	2020-01-31 22:48:12.9890	2020-02-01 01:57:14.1650	60.0	8th St at Ringold St	37.774520	
7	4038	2020-01-31 23:32:03.9070	2020-02-01 00:39:22.0210	450.0	Funston Ave at Irving St	37.763934	
8	4059	2020-01-31 23:31:01.1610	2020-02-01 00:38:40.8570	450.0	Funston Ave at Irving St	37.763934	
9	1980	2020-01-31 23:49:09.2300	2020-02-01 00:22:09.7540	238.0	MLK Jr Way at University Ave	37.871719	

In [44]:
```
#dropping unnecessary columns
columns = ["start_station_latitude", "start_station_longitude", "end_station_latitude",
df.drop(columns, axis = 1, inplace = True)
```

Test

In [45]:
```
df.shape
```

Out[45]:
```
(905007, 11)
```

T4 Time is displayed in term of seconds, instead of minutes or hours.

Define

Add two more columns to display time in minutes and hours spent on each ride

Code

In [46]:
```
# converting seconds to minutes and hours

minutes = df['duration_sec']/60
hours =  df['duration_sec']/3600
```

In [47]:
```
df.insert(1, 'duration_mins', minutes)
df.insert(2, 'duration_hrs', hours)
```

In [48]:
```
days = df['duration_hrs']/24

df.insert(3, 'duration_days', days)
```

Test

In [49]:
```
df.head()
```

Out[49]:

	duration_sec	duration_mins	duration_hrs	duration_days	start_time	end_time	start_station_id	start_sta
0	83118	1385.300000	23.088333	0.962014	2020-01-31 15:23:47.7330	2020-02-01 14:29:06.2630	400.0	Buc No
1	68239	1137.316667	18.955278	0.789803	2020-01-31 15:40:31.6160	2020-02-01 10:37:51.0000	99.0	Folsom S
2	55407	923.450000	15.390833	0.641285	2020-01-31 17:48:16.6350	2020-02-01 09:11:44.3170	197.0	El Emb
3	54929	915.483333	15.258056	0.635752	2020-01-31 17:53:03.4130	2020-02-01 09:08:32.6340	197.0	El Emb
4	55700	928.333333	15.472222	0.644676	2020-01-31 17:12:33.4600	2020-02-01 08:40:53.6460	12.0	Pier En

T5. 'Day' column is missing from the datasets

Define

Add 'day' column to the dataframe

Code

In [50]:
```
df[['start_time', 'end_time']].info() # finding the data type
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 905007 entries, 0 to 905006
Data columns (total 2 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   start_time  905007 non-null  object
 1   end_time    905007 non-null  object
dtypes: object(2)
memory usage: 13.8+ MB
```

In [51]:
```
# convert the two data types from string to datetime data type
df[['start_time', 'end_time']] = df[['start_time', 'end_time']].apply(pd.to_datetime)

# create two columns, 'start_day' and 'end_day'
start_day = df['start_time'].dt.day_name()    #defining start_day and end_day
end_day = df['end_time'].dt.day_name()

df.insert(4, 'start_day', start_day)
df.insert(6, 'end_day', end_day)
```

In [52]:
```
df.head()
```

Out[52]:

	duration_sec	duration_mins	duration_hrs	duration_days	start_day	start_time	end_day	end_time	start_s
0	83118	1385.300000	23.088333	0.962014	Friday	2020-01-31 15:23:47.733	Saturday	2020-02-01 14:29:06.263	
1	68239	1137.316667	18.955278	0.789803	Friday	2020-01-31 15:40:31.616	Saturday	2020-02-01 10:37:51.000	
2	55407	923.450000	15.390833	0.641285	Friday	2020-01-31 17:48:16.635	Saturday	2020-02-01 09:11:44.317	
3	54929	915.483333	15.258056	0.635752	Friday	2020-01-31 17:53:03.413	Saturday	2020-02-01 09:08:32.634	
4	55700	928.333333	15.472222	0.644676	Friday	2020-01-31 17:12:33.460	Saturday	2020-02-01 08:40:53.646	

In [53]:
```
df.shape
```

Out[53]:
```
(905007, 16)
```

Cleaning: Quality Issues

(i) Wrong Data types

Define

Correct the wrong data types, by converting 'start_station_id', and 'end_station_id' into strings since no analysis will be done on its values.

Code

In [54]:
```
df[['start_station_id', 'end_station_id']] = df[['start_station_id', 'end_station_id']].
```

Test

In [55]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 905007 entries, 0 to 905006
Data columns (total 16 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   duration_sec          905007 non-null  int64
```

```
 1   duration_mins          905007 non-null  float64
 2   duration_hrs           905007 non-null  float64
 3   duration_days          905007 non-null  float64
 4   start_day              905007 non-null  object
 5   start_time             905007 non-null  datetime64[ns]
 6   end_day                905007 non-null  object
 7   end_time               905007 non-null  datetime64[ns]
 8   start_station_id       905007 non-null  object
 9   start_station_name     419938 non-null  object
 10  end_station_id         905007 non-null  object
 11  end_station_name       420499 non-null  object
 12  bike_id                905007 non-null  int64
 13  user_type              905007 non-null  object
 14  rental_access_method   617858 non-null  object
 15  month                  905007 non-null  object
dtypes: datetime64[ns](2), float64(3), int64(2), object(9)
memory usage: 110.5+ MB
```

Q2. Outlier in Jan dataset

Define

Drop the row containing the outlier

Code

In [56]:
```
df.nlargest(5, 'duration_sec')
```

Out[56]:

	duration_sec	duration_mins	duration_hrs	duration_days	start_day	start_time	end_day	end_time
121168	811077	13517.950000	225.299167	9.387465	Thursday	2020-01-09 16:23:21.000	Sunday	2020-01-19 01:41:19.000
347472	86317	1438.616667	23.976944	0.999039	Monday	2020-02-17 11:42:26.059	Tuesday	2020-02-18 11:41:03.741
386984	86239	1437.316667	23.955278	0.998137	Thursday	2020-02-06 17:03:46.438	Friday	2020-02-07 17:01:05.703
49926	86221	1437.016667	23.950278	0.997928	Thursday	2020-01-16 07:51:37.709	Friday	2020-01-17 07:48:39.515
63677	85994	1433.233333	23.887222	0.995301	Sunday	2020-01-12 17:08:54.110	Monday	2020-01-13 17:02:08.530

In [57]:
```
df.drop(index = 121168, inplace = True)
```

Test

In [58]:
```
df.describe()
```

Out[58]:

	duration_sec	duration_mins	duration_hrs	duration_days	bike_id
count	905006.000000	905006.000000	905006.000000	905006.000000	905006.000000
mean	805.888110	13.431469	0.223858	0.009327	345257.291351
std	1415.275537	23.587926	0.393132	0.016381	305503.268537

min	60.000000	1.000000	0.016667	0.000694	12.000000
25%	370.000000	6.166667	0.102778	0.004282	12114.000000
50%	595.000000	9.916667	0.165278	0.006887	326439.000000
75%	932.000000	15.533333	0.258889	0.010787	557460.000000
max	86317.000000	1438.616667	23.976944	0.999039	999960.000000

In [59]:
```
df.nlargest(5, 'duration_sec')
```

Out[59]:

	duration_sec	duration_mins	duration_hrs	duration_days	start_day	start_time	end_day	end_time
347472	86317	1438.616667	23.976944	0.999039	Monday	2020-02-17 11:42:26.059	Tuesday	2020-02-18 11:41:03.741
386984	86239	1437.316667	23.955278	0.998137	Thursday	2020-02-06 17:03:46.438	Friday	2020-02-07 17:01:05.703
49926	86221	1437.016667	23.950278	0.997928	Thursday	2020-01-16 07:51:37.709	Friday	2020-01-17 07:48:39.515
63677	85994	1433.233333	23.887222	0.995301	Sunday	2020-01-12 17:08:54.110	Monday	2020-01-13 17:02:08.530
27024	85450	1424.166667	23.736111	0.989005	Thursday	2020-01-23 14:29:54.658	Friday	2020-01-24 14:14:05.000

Maximum number of duration in seconds is now 86317

Storing clean data

In [60]:
```
df.to_csv('clean_df.csv', index = False) # storing clean data
```

Exploratory Data Analysis

(i) Univariate Exploration

In [61]:
```
df_2020 = pd.read_csv('clean_df.csv', dtype = {'rental_access_method' : 'object'}) # rea
```

In [62]:
```
df_2020.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 905006 entries, 0 to 905005
Data columns (total 16 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   duration_sec          905006 non-null  int64  
 1   duration_mins         905006 non-null  float64
 2   duration_hrs          905006 non-null  float64
 3   duration_days         905006 non-null  float64
 4   start_day             905006 non-null  object 
 5   start_time            905006 non-null  object 
 6   end_day               905006 non-null  object 
 7   end_time              905006 non-null  object 
 8   start_station_id      418705 non-null  float64
 9   start_station_name    419938 non-null  object 
```

```
 10  end_station_id        419225 non-null  float64
 11  end_station_name      420499 non-null  object
 12  bike_id               905006 non-null  int64
 13  user_type             905006 non-null  object
 14  rental_access_method  617857 non-null  object
 15  month                 905006 non-null  object
dtypes: float64(5), int64(2), object(9)
memory usage: 110.5+ MB
```

I'll begin by looking at the checking the distribution of the main variable of interest; duration in minutes

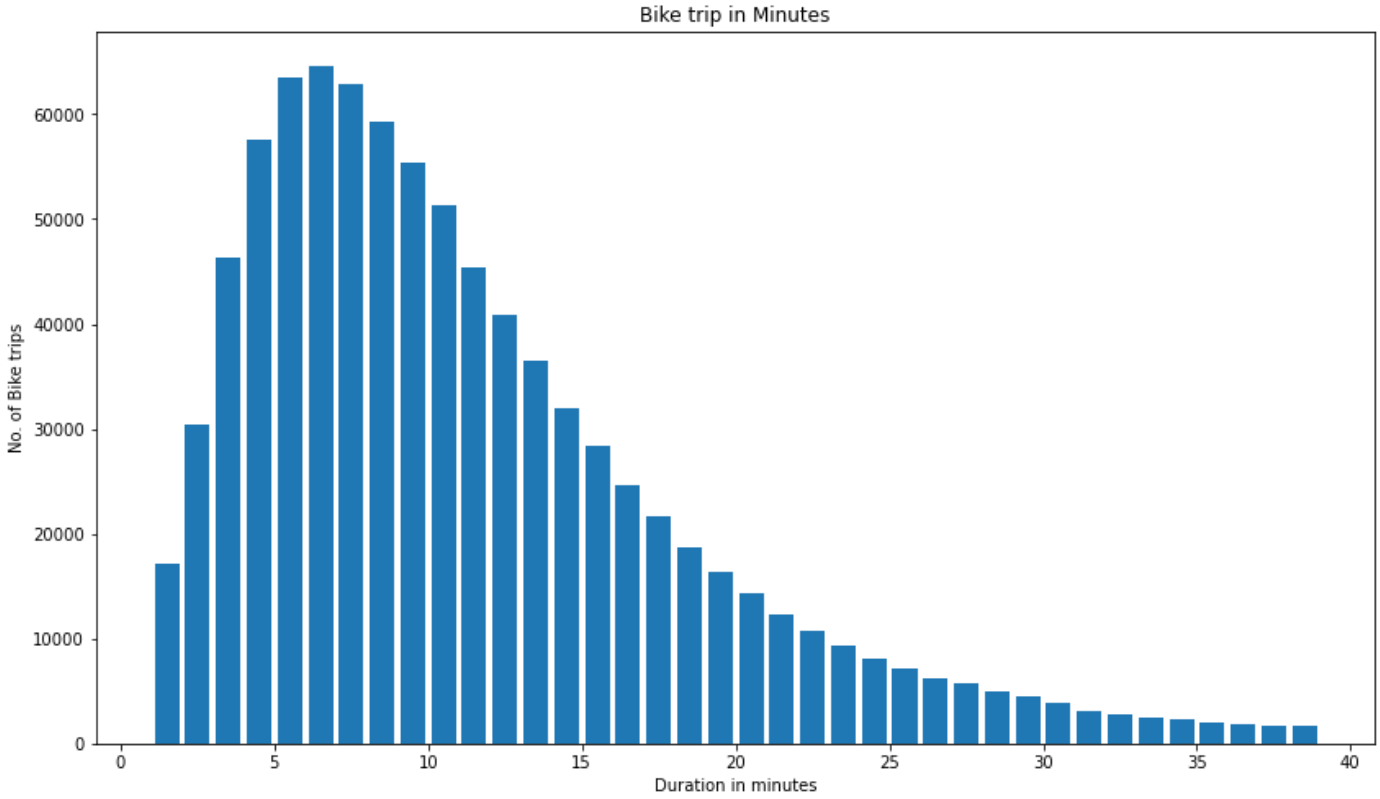
In [63]:
```
df_2020['duration_mins'].describe() # descriptive statistics
```

Out[63]:
```
count    905006.000000
mean         13.431469
std          23.587926
min           1.000000
25%           6.166667
50%           9.916667
75%          15.533333
max        1438.616667
Name: duration_mins, dtype: float64
```

- Overall bike trip duration is 13.431469 minutes.

In [109…
```
#defining a function to assist set x  and y labels and the title of plots
def function(xL, yL, title):
    plt.title(title)
    plt.xlabel(xL)
    plt.ylabel(yL)
```

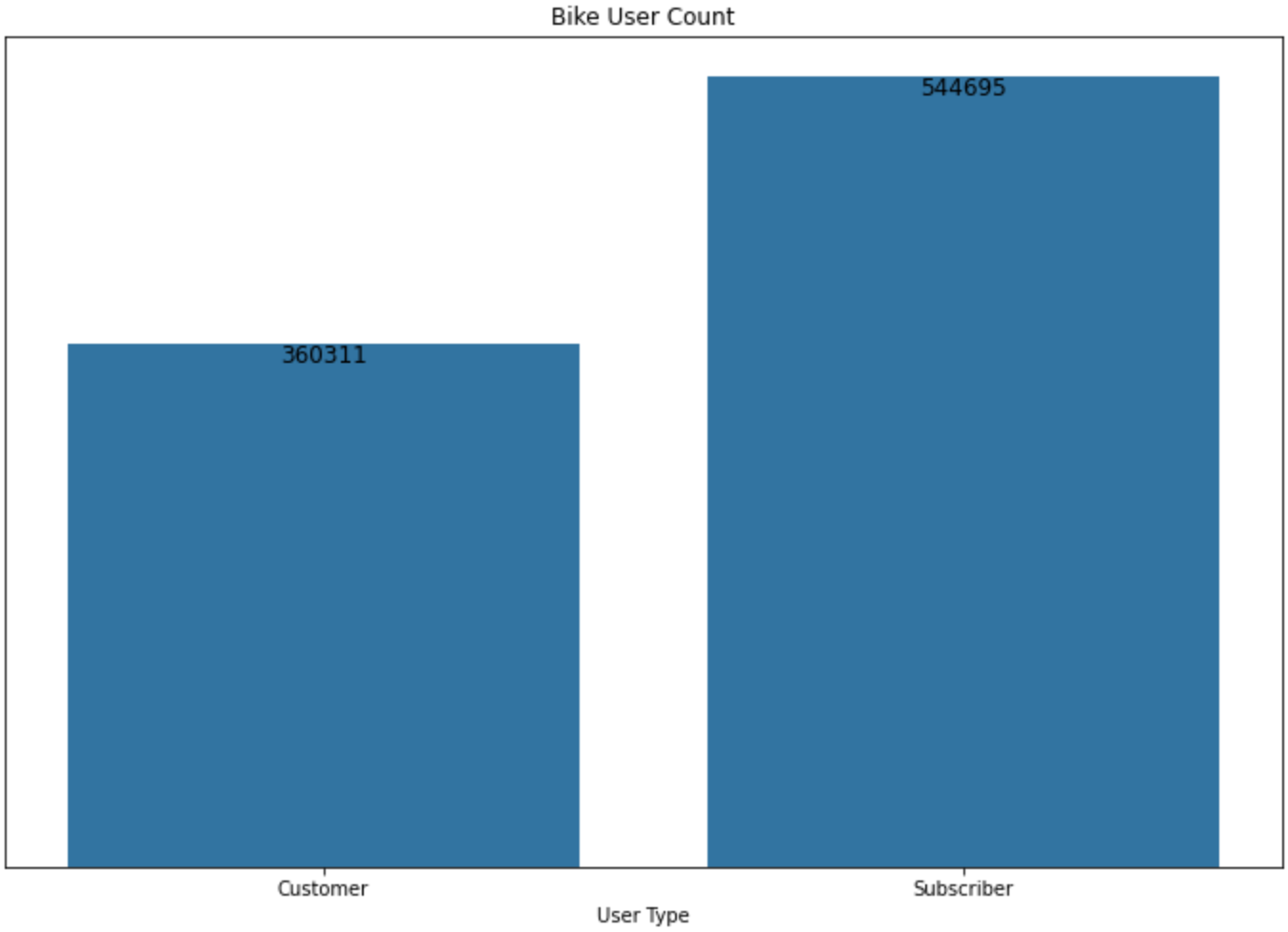
In [107…
```
# Duration of trips in minutes

# settings of the plot
plt.figure(figsize = [14, 8])
bin_edges = np.arange(1, 40, 1)
ticks = [0, 5, 10, 15, 20, 25, 30, 35, 40]
labels = ['{}'.format(val) for val in ticks]

# creating a histogram
plt.hist(data = df_2020, x = 'duration_mins', bins = bin_edges, rwidth = 0.8);
plt.xticks(ticks, labels)
function('Duration in minutes', 'No. of Bike trips', 'Bike trip in Minutes' )
```



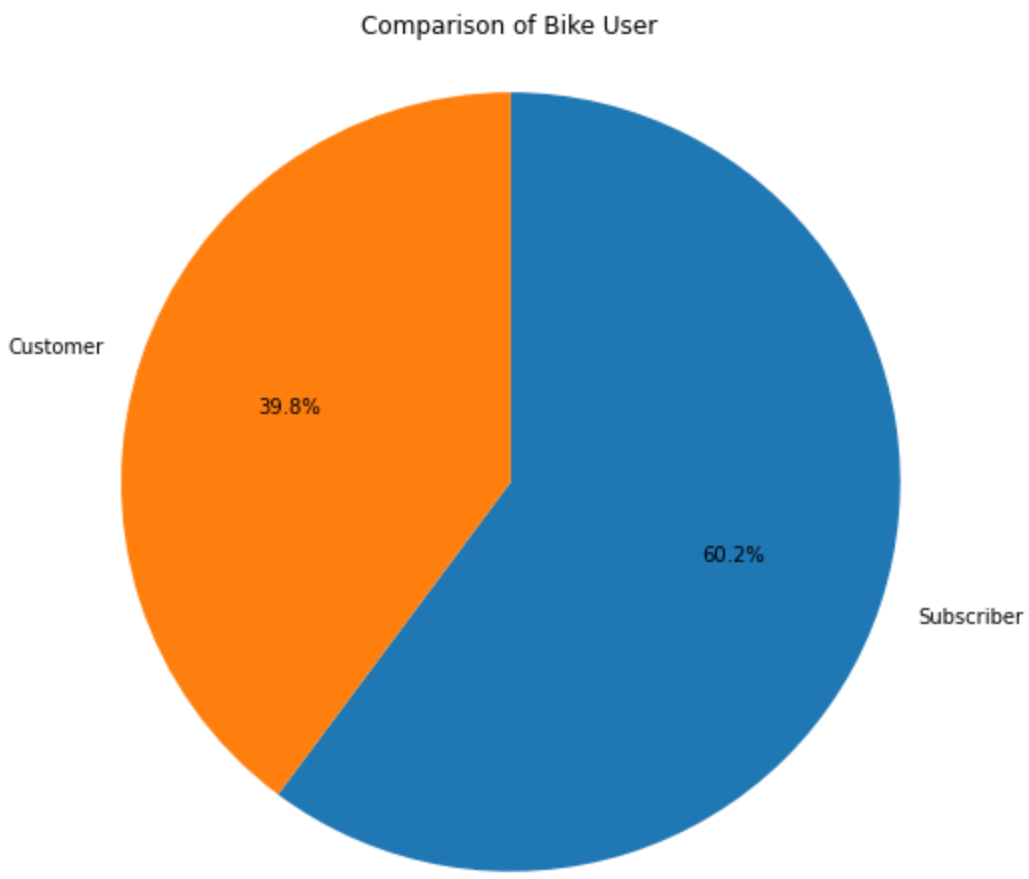
In [ ]:
```
- The distribution is right skewed.
- It is observed that the average duration of bike trip lasts between 3-13 minutes
```

Next would be to plot pie charts to determine Number of and rental access methods used

In [67]:
```
plt.figure(figsize=(12,8)) # setting plot size
base_color = sb.color_palette()[0] #setting the color

#creating a barplot
ax = sb.countplot(data=df_2020, x ='user_type', color=base_color)

type_counts = df_2020['user_type'].value_counts()
locs, labels = plt.xticks()

# to loop through each set of loc and labels
for loc, label in zip(locs, labels):
    count = type_counts[label.get_text()]
    pct_string = '{}'.format(count)

    plt.text(loc, count-8, pct_string, va='top', ha='center', fontsize=12)#to create ann

plt.yticks([])
function('User Type', '', 'Bike User Count');
```



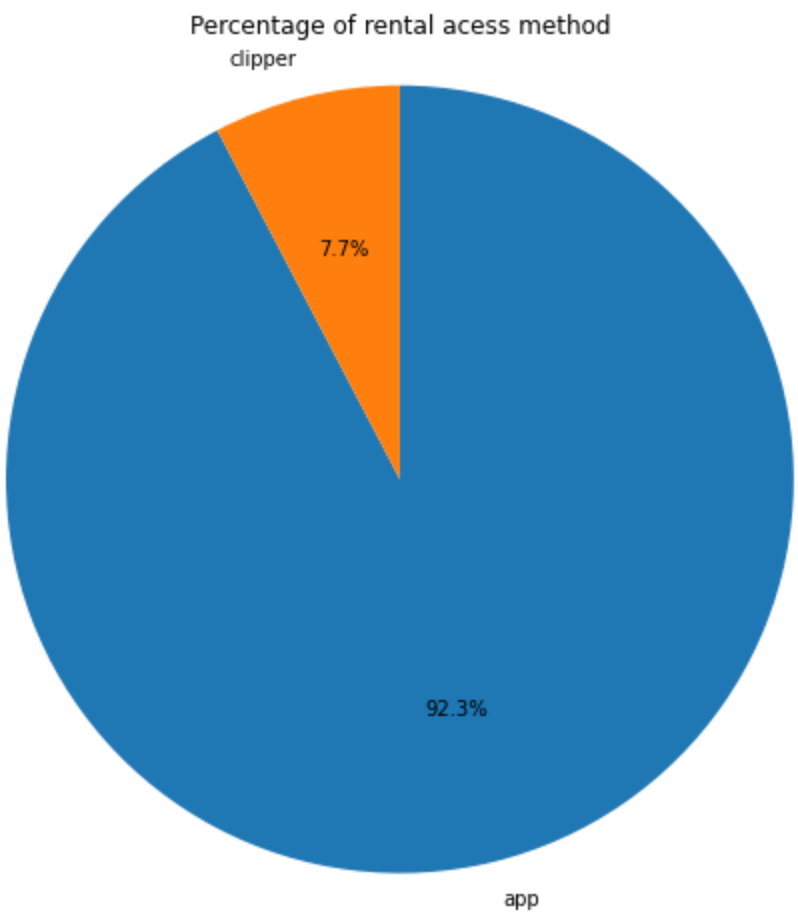
In [68]:
```
plt.figure(figsize = [14,8]) #setting figure size

#creating a pie chart
counts = df_2020['user_type'].value_counts()
sorted_counts= counts.index
plt.pie(counts, labels = sorted_counts, startangle = 90, autopct ='%1.1f%%', countercloc
plt.axis('square');
function('', '', 'Comparison of Bike User')
```



- Subscribers take up larger proportions of the bike rides taken as compared to the customers, who only take up about 40%.

To show proportions of rental access method employed by users

In [69]:
```
plt.figure(figsize = [14,8]) #setting figure size
rental_counts = df_2020['rental_access_method'].value_counts()

#creating pie chart to show proportions
sorted_rental_counts = df_2020['rental_access_method'].value_counts().index
plt.pie(rental_counts, labels = sorted_rental_counts, startangle = 90, autopct ='%1.1f%%
plt.axis('square');

function('', '', 'Percentage of rental acess method')
```



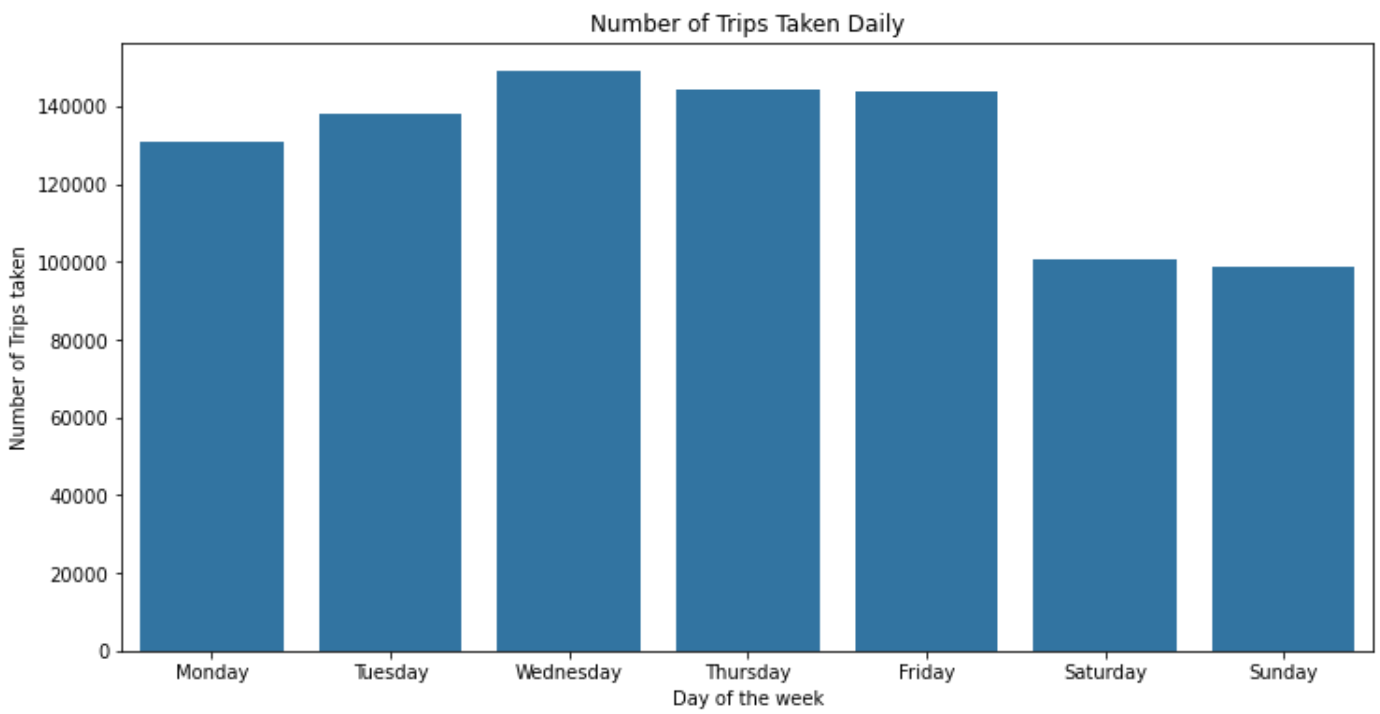
- Most users use the app to rent bikes. with only a few using the dipper method

2. Finding out which month most trips are taken

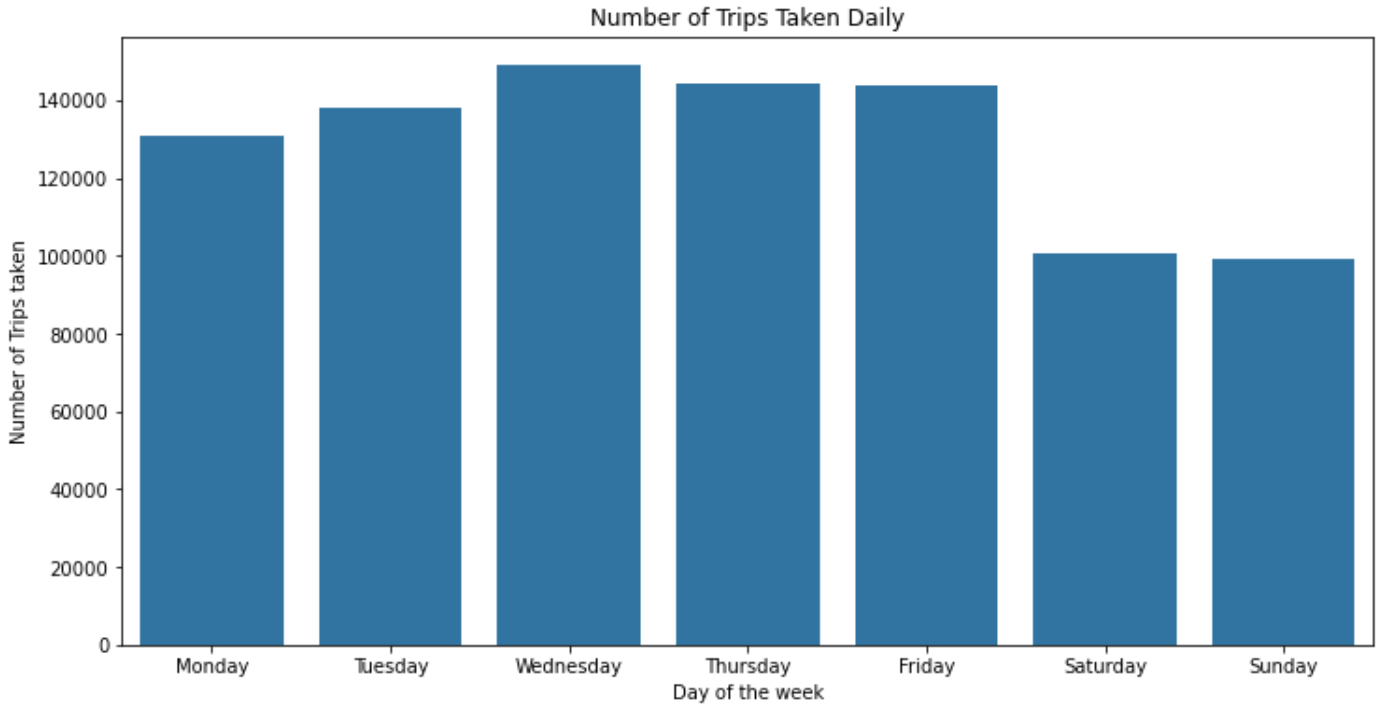
I'll now move to the next step of determining number of trips taken each month

(i). Weekly trips

In [97]:
```
plt.figure(figsize = [12, 6]) #figure size setting

base_color = sb.color_palette()[0] #setting the color
day_order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday


#creating a barchart
sb.countplot(data = df_2020, x = 'start_day', color = base_color, order = day_order);
function('Day of the week', 'Number of Trips taken', 'Number of Trips Taken Daily')
```



In [111…
```
plt.figure(figsize = [12, 6]) #setting figure size

sb.countplot(data = df_2020, x = 'end_day', color = base_color,order = day_order)
function('Day of the week','Number of Trips taken', 'Number of Trips Taken Daily');
```



- Most trips are taken on wednesday. Seems like less trips are taken during the weekend.
- Most trips also end on wednesdays

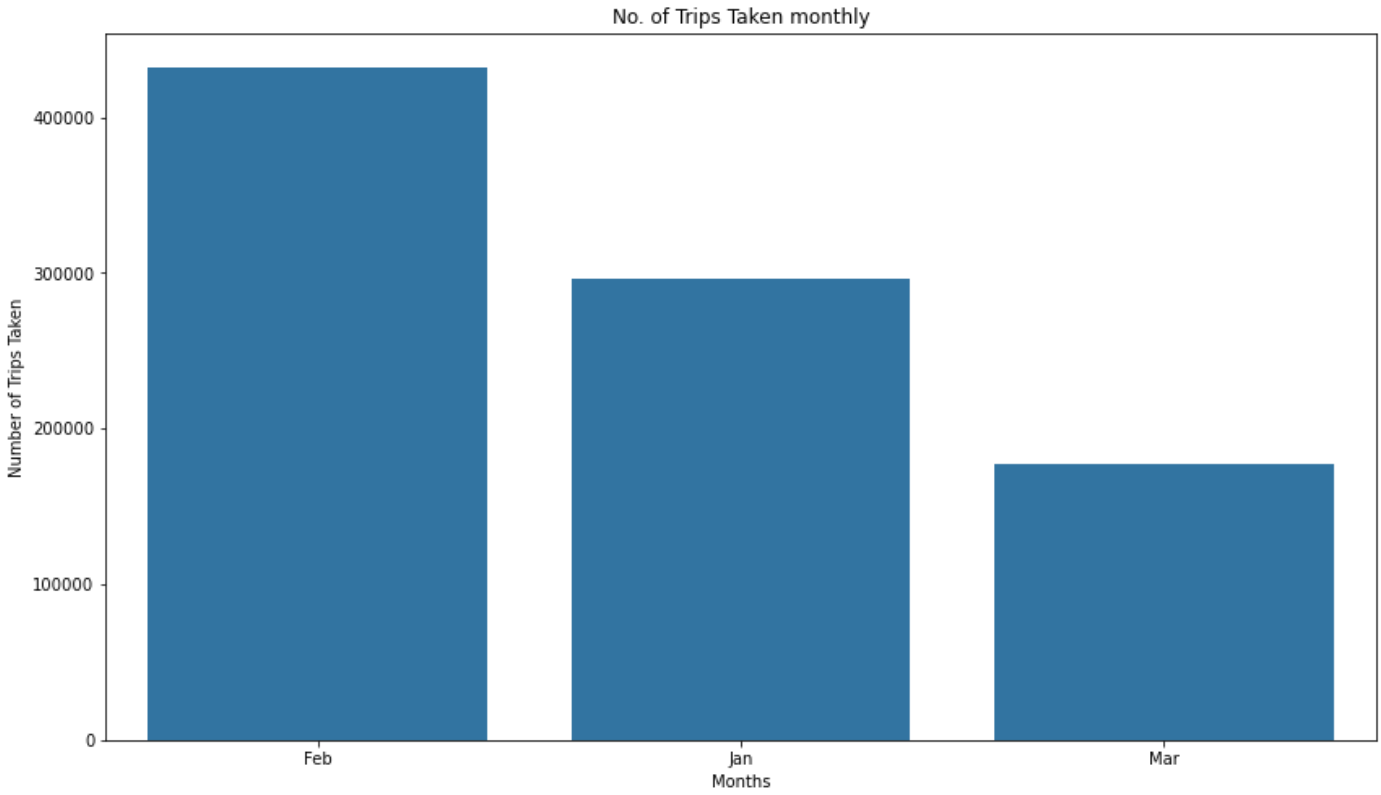
(ii) Trips taken; Monthly count

In [72]:
```
df_2020['month'].value_counts() #to get a count of every month
```

Out[72]:
```
Feb    432354
Jan    295853
```

```
Mar    176799
Name: month, dtype: int64
```

In [112…
```
plt.figure(figsize = [14, 8]) #setting the figure size
order = df_2020['month'].value_counts().index

#creating a barchart
sb.countplot(data = df_2020, x = 'month', color = base_color, order = order)
function('Months', 'Number of Trips Taken', 'No. of Trips Taken monthly');
```



Most trips are taken in February

(iii) Hourly count

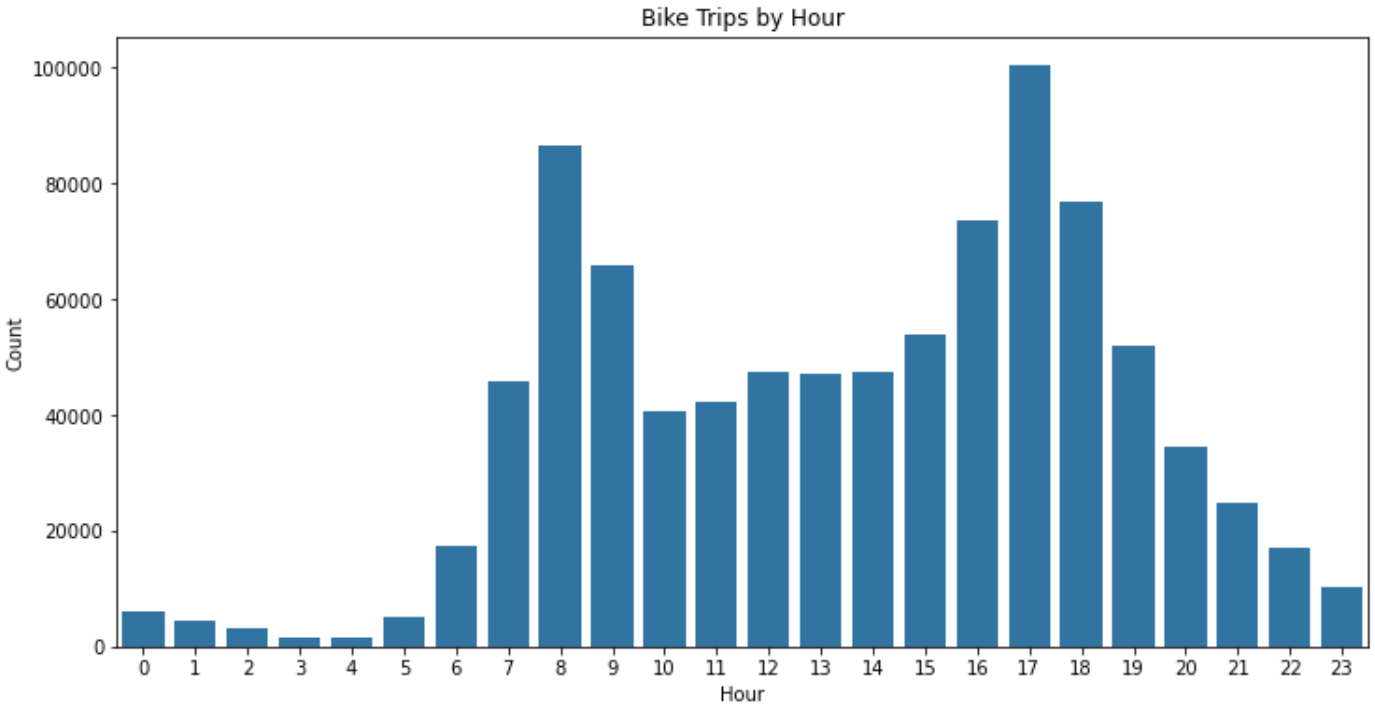
In [74]:
```
# create a new column, 'hour'
df_2020['start_time'] = pd.to_datetime(df_2020['start_time'])

df_2020['hour'] = df_2020['start_time'].dt.hour
```

In [75]:
```
df_2020.info() #to proof that the column was created
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 905006 entries, 0 to 905005
Data columns (total 17 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   duration_sec          905006 non-null  int64
 1   duration_mins         905006 non-null  float64
 2   duration_hrs          905006 non-null  float64
 3   duration_days         905006 non-null  float64
 4   start_day             905006 non-null  object
 5   start_time            905006 non-null  datetime64[ns]
 6   end_day               905006 non-null  object
 7   end_time              905006 non-null  object
 8   start_station_id      418705 non-null  float64
 9   start_station_name    419938 non-null  object
 10  end_station_id        419225 non-null  float64
 11  end_station_name      420499 non-null  object
```

```
 12  bike_id                 905006 non-null  int64
 13  user_type               905006 non-null  object
 14  rental_access_method    617857 non-null  object
 15  month                   905006 non-null  object
 16  hour                    905006 non-null  int64
dtypes: datetime64[ns](1), float64(5), int64(3), object(8)
memory usage: 117.4+ MB
```

In [76]:
```
order = np.arange(0,24) # to get results for the first 24 hours

base_color = sb.color_palette()[0] #setting up the color of the bars

plt.figure(figsize = [12,6]) #setting figure size

#creating a barplot
sb.countplot(data = df_2020, x = 'hour', order = order, color = base_color)
function('Hour', 'Count', 'Bike Trips by Hour');
```



Accordding the plot above, bike services are taken during peak hours.

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

The distribution is mostly skewed to the right(long tail to the right) with most points lying between 1 to 40 minutes of each bike ride

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 datasets collected had some quality issues. There were erroneous data types that had to be canged inoredr to conduct analysis on them, some columns were also added.
- The histogram generated on the original data had few points from 40 minutes going onwards, the data was trimmed to focus on the first 1 to 40 minutes of the dataset

(b) Bivariate Exploration

i'll start by exploring the relationship between duration of trips taken and the user type, to determine how the user type influence time taken on each bike ride

(i) Trip duration depending on the User type

In [77]:
```
df2 = df_2020[df_2020['duration_mins'] <= 120] #creating a dataframe of duration points
df2
```

Out[77]:

	duration_sec	duration_mins	duration_hrs	duration_days	start_day	start_time	end_day	end_tim
7	4038	67.300000	1.121667	0.046736	Friday	2020-01-31 23:32:03.907	Saturday	2020-02- 00:39:22.0
8	4059	67.650000	1.127500	0.046979	Friday	2020-01-31 23:31:01.161	Saturday	2020-02- 00:38:40.8
9	1980	33.000000	0.550000	0.022917	Friday	2020-01-31 23:49:09.230	Saturday	2020-02- 00:22:09.7
10	1969	32.816667	0.546944	0.022789	Friday	2020-01-31 23:49:03.972	Saturday	2020-02- 00:21:53.5
11	1611	26.850000	0.447500	0.018646	Friday	2020-01-31 23:49:10.931	Saturday	2020-02- 00:16:02.4
...	...	...	...	...	...	...	...	
905001	61	1.016667	0.016944	0.000706	Wednesday	2020-04-01 10:42:35.000	Wednesday	2020-04- 10:43:
905002	78	1.300000	0.021667	0.000903	Wednesday	2020-03-18 17:10:42.000	Wednesday	2020-03- 17:12:
905003	122	2.033333	0.033889	0.001412	Wednesday	2020-04-01 13:32:28.000	Wednesday	2020-04- 13:34:
905004	340	5.666667	0.094444	0.003935	Wednesday	2020-04-01 13:07:34.000	Wednesday	2020-04- 13:13:
905005	333	5.550000	0.092500	0.003854	Monday	2020-03-23 16:39:57.000	Monday	2020-03- 16:45:

902387 rows × 17 columns

In [78]:
```
df1 = df_2020[df_2020['duration_mins'] <= 60] # creating a dataframe of durations less t
df1
```

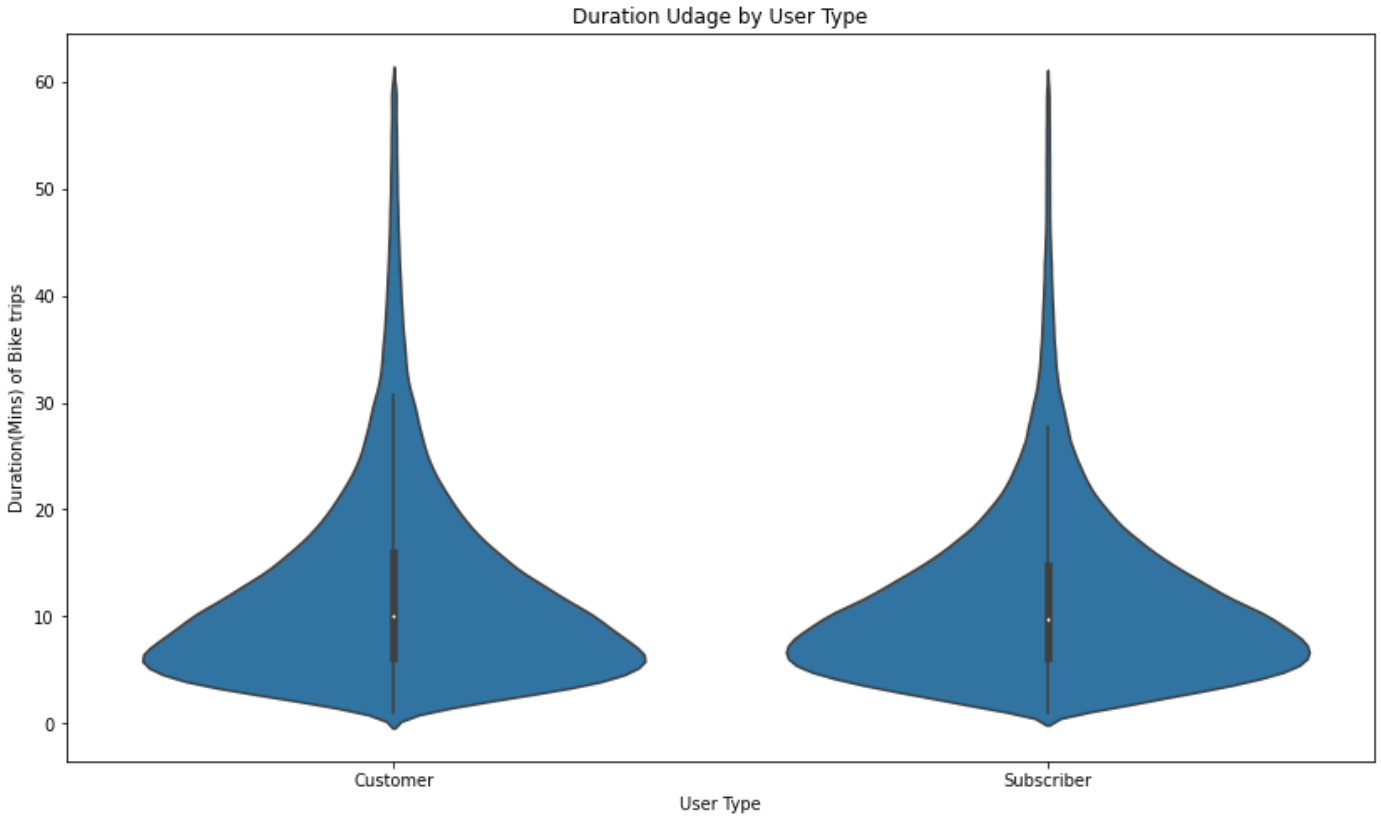
Out[78]:

	duration_sec	duration_mins	duration_hrs	duration_days	start_day	start_time	end_day	end_tim
9	1980	33.000000	0.550000	0.022917	Friday	2020-01-31 23:49:09.230	Saturday	2020-02- 00:22:09.7
10	1969	32.816667	0.546944	0.022789	Friday	2020-01-31 23:49:03.972	Saturday	2020-02- 00:21:53.5
11	1611	26.850000	0.447500	0.018646	Friday	2020-01-31 23:49:10.931	Saturday	2020-02- 00:16:02.4
12	1133	18.883333	0.314722	0.013113	Friday	2020-01-31 23:56:49.475	Saturday	2020-02- 00:15:42.9

13	1119	18.650000	0.310833	0.012951	Friday	2020-01-31 23:57:02.284	Saturday	2020-02- 00:15:42.0
...	...	...	...	...	...	...	...	
905001	61	1.016667	0.016944	0.000706	Wednesday	2020-04-01 10:42:35.000	Wednesday	2020-04- 10:43:
905002	78	1.300000	0.021667	0.000903	Wednesday	2020-03-18 17:10:42.000	Wednesday	2020-03- 17:12:
905003	122	2.033333	0.033889	0.001412	Wednesday	2020-04-01 13:32:28.000	Wednesday	2020-04- 13:34:
905004	340	5.666667	0.094444	0.003935	Wednesday	2020-04-01 13:07:34.000	Wednesday	2020-04- 13:13:
905005	333	5.550000	0.092500	0.003854	Monday	2020-03-23 16:39:57.000	Monday	2020-03- 16:45:

893066 rows × 17 columns

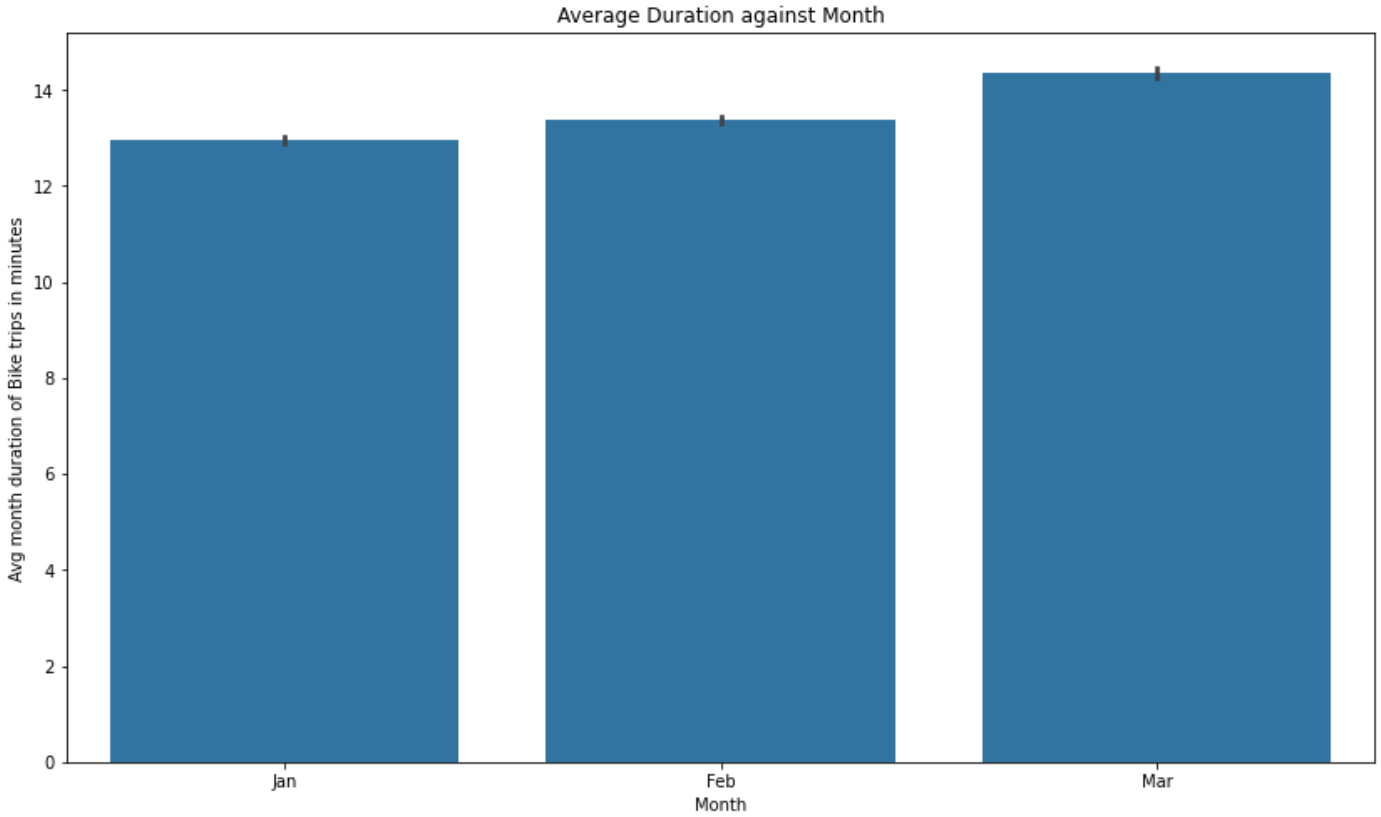
In [79]:
```
#creating the size of the plot
plt.figure(figsize= [14,8])

base_color = sb.color_palette()[0] # setting the color of the vusuals

#creating a violin plot to display average duration relationship to user types
sb.violinplot(data = df1, x = 'user_type', y = 'duration_mins', color = base_color )
function('User Type', 'Duration(Mins) of Bike trips', 'Duration Udage by User Type');
```



At duration less than or equal to 60 minutes, both user types(Casual) and subscribers get almost equal amount of time of using the bikes.

(ii) duration against the month

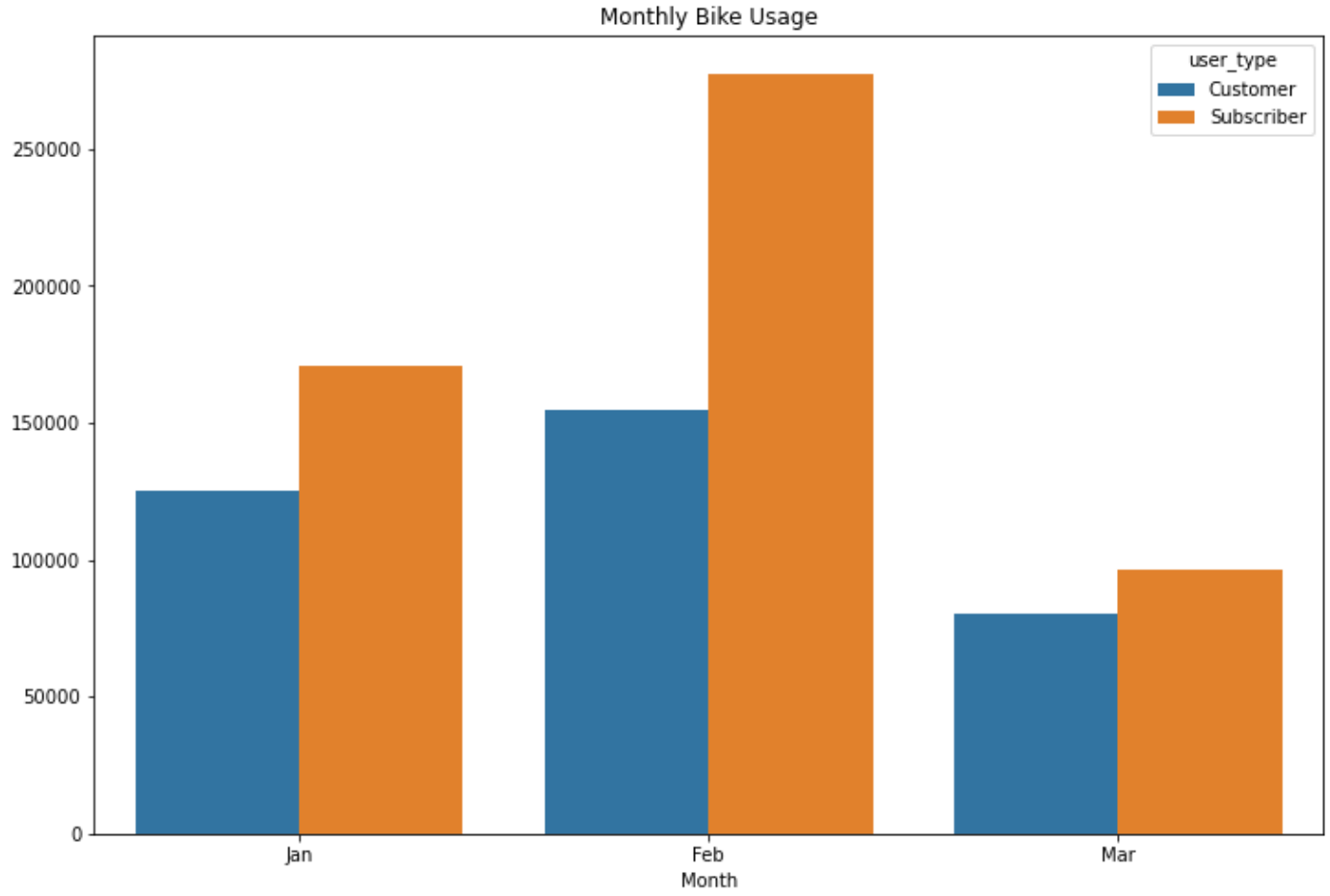
In [80]:
```
plt.figure(figsize= [14,8]) #setting figure size

base_color = sb.color_palette()[0] # setting the color of the bars

#creating a barplot
sb.barplot(data = df_2020, x = 'month', y = 'duration_mins', color = base_color)
function('Month', 'Avg month duration of Bike trips in minutes', 'Average Duration again
```



- The average duration of each bike ride has been increasing. It reached a peak in march. This is to imply that the idea of bike service has been succesful due to the increase in number of duration on bike trips.

Monthly Usage Analysis

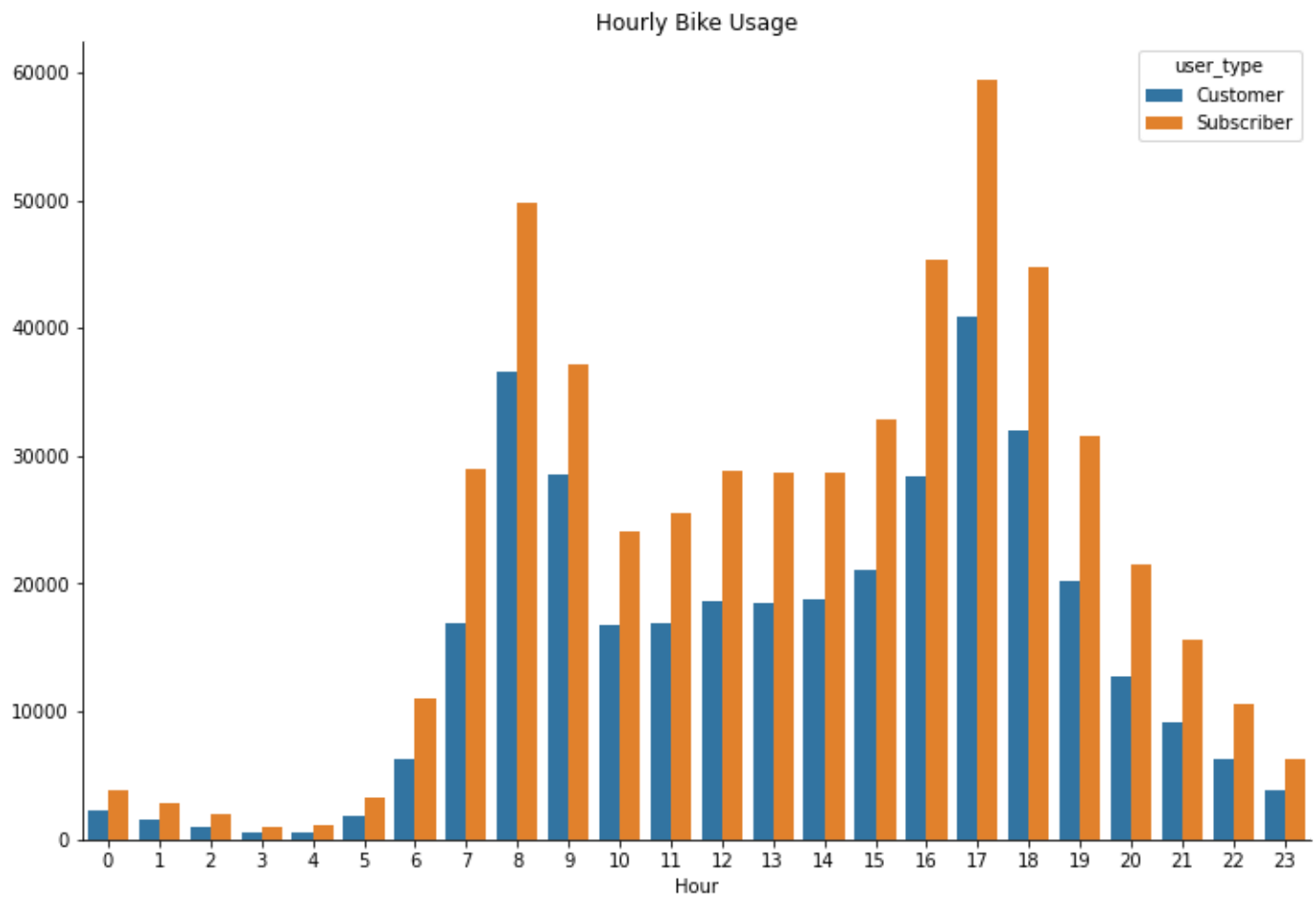
In [104…
```
order = ["Jan", "Feb", "Mar"]

plt.figure(figsize=(12,8)) # setting the figure size

#creating a barchart to display bike usage between the months
ax=sb.countplot(data= df_2020, x = 'month', order = order, hue = 'user_type')
function('Month', '', 'Monthly Bike Usage');
```



- Viewed by month, The plot depicts that, most of the trips were taken by subscribers.

Hourly Usage Analysis

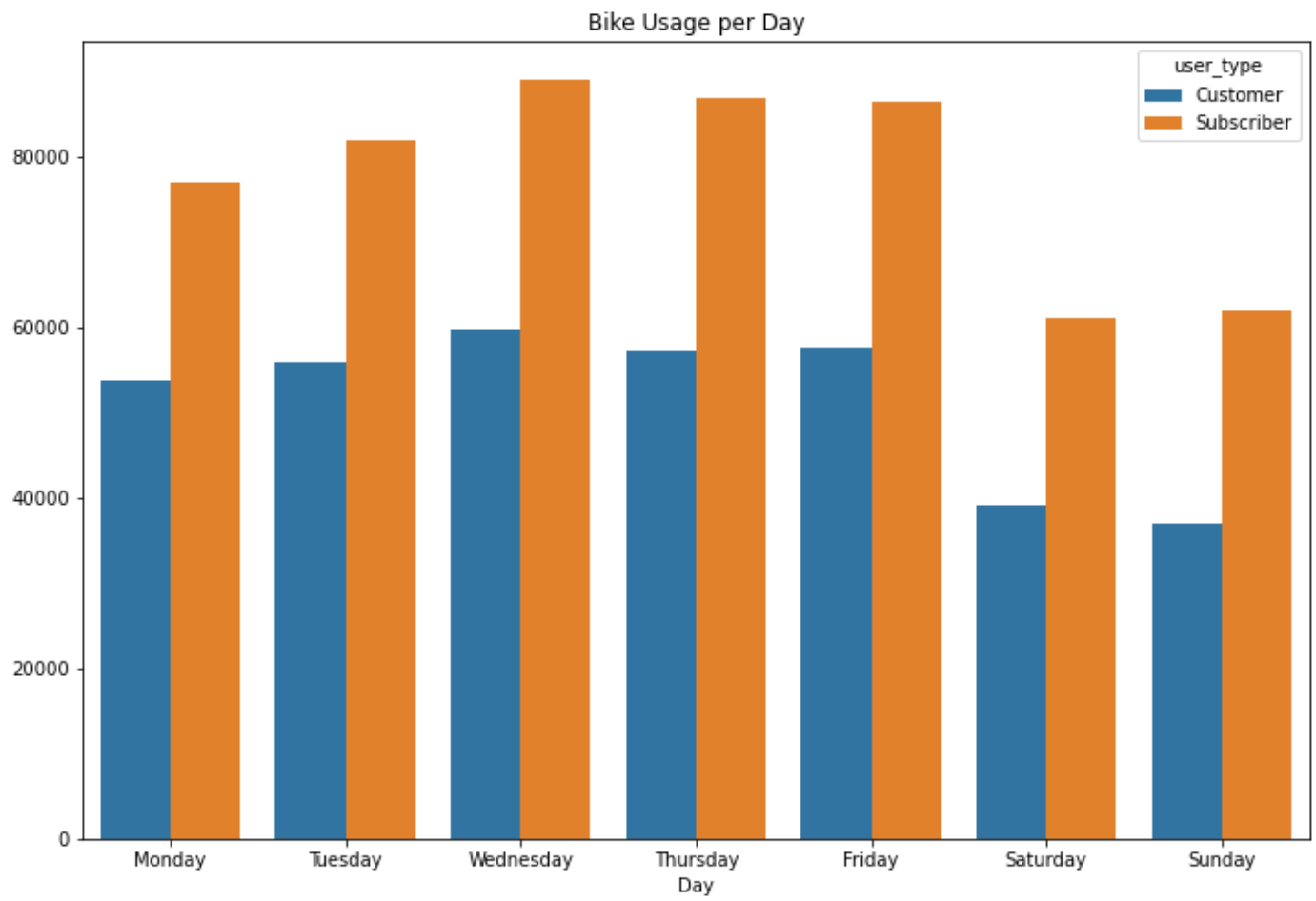
In [82]:
```
#setting the figure size
plt.figure(figsize=(12,8))

#creating a barchart to show how bike usage vary hourly
sb.countplot(data= df_2020, x = 'hour', hue = 'user_type')
function('Hour', '', 'Hourly Bike Usage')
sb.despine(fig=None, top=True, right=True, left=False, bottom=False, offset=None, trim=F
```



- Both the customers and subscribers use the service more during pesk hours, but the usage is more by the subscribers during thses hours, as comapred to customers whose schedule tend to be more flexilbe.

Daily Usage Analysis

In [99]:
```
#creating a bar chart to depict how bike trip usage vary between days of the week
plt.figure(figsize=[12,8])
sb.countplot(data=df_2020, x='start_day', order=day_order, hue='user_type')
function('Day', '', 'Bike Usage per Day');
```



- Usage of Bike services is high on weekdays and low on weeknds
- Subsribers tend to use the service less on weekends and more on weekdays, there's a steady pattern notable during weekdays

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

There seems to have a relationship between duration and user type, where customers tend to have longer durations than subscribers

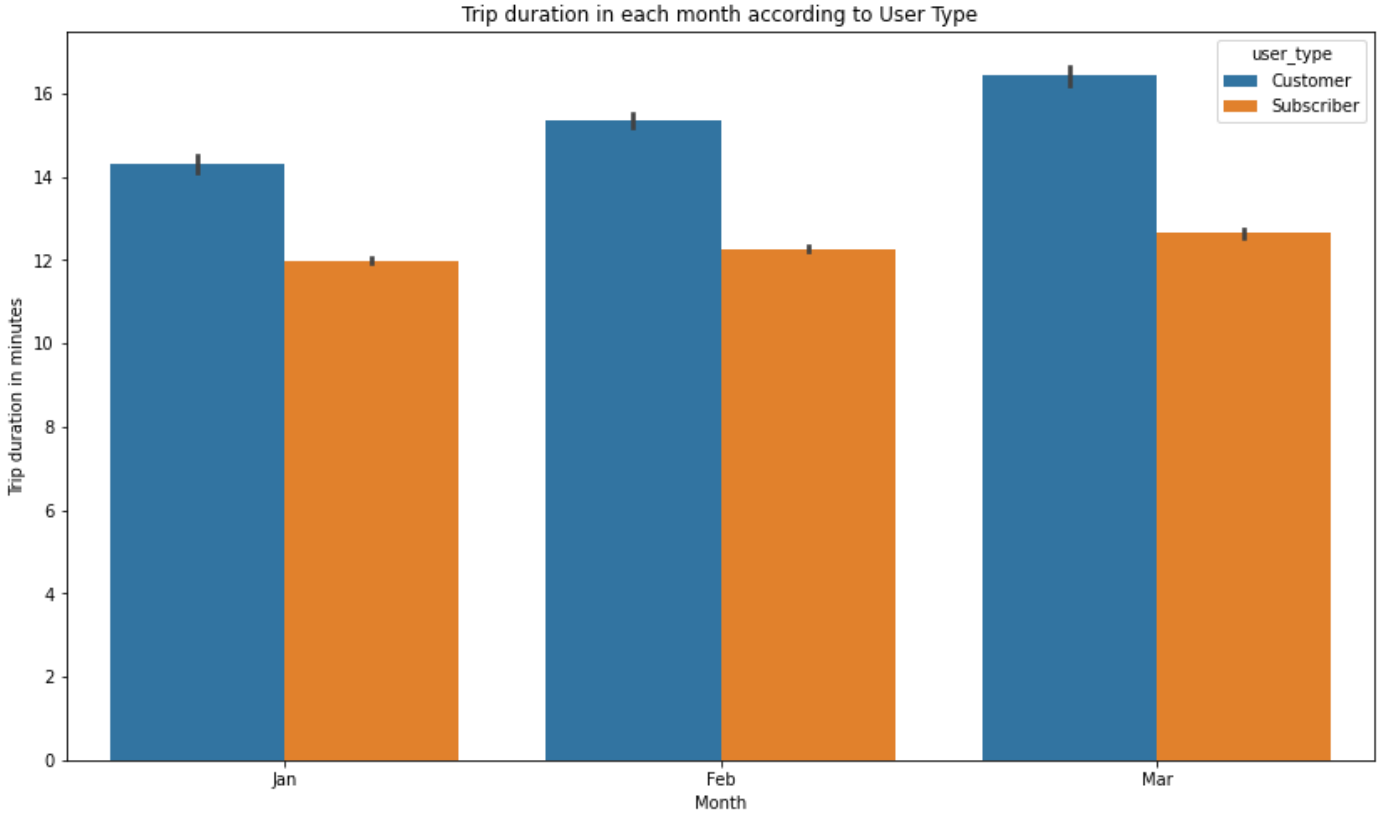
Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

- Bike service tend to be more in usage during peak hours, especially by subscribers. Customers however have a more flexible pattern compare to subscribers.

3.Multivariate Exploration

In [114…

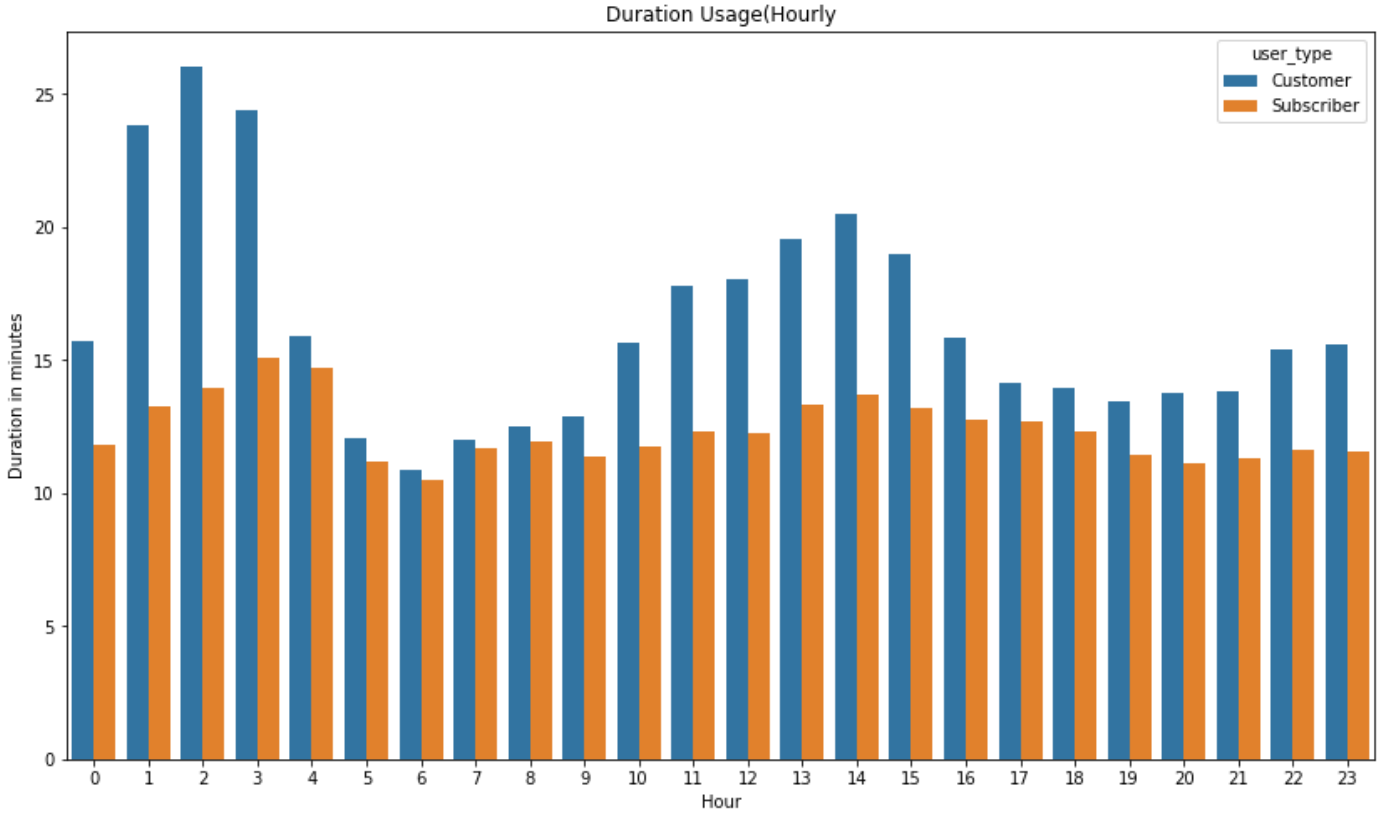
```
plt.figure(figsize = [14, 8]) #setting the figure size

# creating  a bar plot to display trip durations between months vary with the user types
sb.barplot(data = df_2020, x = 'month', y = 'duration_mins', hue = 'user_type')
function('Month', 'Trip duration in minutes', 'Trip duration in each month according to
```



- Viewed by month, according to the plot above, customers had longer durations than subscribers

In [116…

```
plt.figure(figsize=(14,8)) #setting the figure size

#creating a barplot to display relationship between duration, user type and  hour variab

sb.barplot(data=df_2020, x='hour', y='duration_mins', ci=None, hue='user_type');
function('Hour', 'Duration in minutes', 'Duration Usage(Hourly');
```



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?

When viewed according to the month, by comparing user types, customers took longer durations than subscribers.

- Type of user has some influence on the usage of bike service. For instance on the duration of use, compared hourly, subscribers tend to have a stable pattern than customers. Customer usage vary greatly by the hour.
- Second, Suscribers tend to use the service mostly on weekdays and less on weeknds, compared to customers who seem to be more flexible. Although in general bike service seems to be in use less on weekends.
- On hour Usage, customers seem to have more flexibility as compared to subscribers who seem to have more intense usage at peak hours.

Were there any interesting or surprising interactions between features

- Between 2 AM and 3AM, there's average high duration, especially by customers, which is not notable in any other hour.

CONCLUSION

- The idea has been successful, since the average trip duration has been increasing since january, through to february and reached its peak in march. This leads to conclusion that more people are depending on bikes to travel long distances.
- During weekdays, most trips are preffered on wednesdays.
- Comparing the three months, most trips are preffered in February.
- Most trips are taken on shorter durations, but as for longer durations less and less trips are taken.