Project: Ford GoBike System Data Exploration

Yu Tao 2020 12/28

Table of Contents

Introduction

In this project, I explored the Ford GoBike system data. This data set includes information about individual rides made in a bike-sharing system covering the greater San Francisco Bay area. The data set was directly downloaded from the Udacity Data Analyst Nanodegree project website, the file is named 201902-Fordgobike-Tripdata.csv.

Questions to Answer

To help better explore the data, I listed several questions I'd like to answer:

- **When are most trips taken in terms of time of day or day of the week?**
- **How long does the average trip take?**
- **Does the above depend on if a user is a subscriber or customer?**

It is almost certain that after the exploration, we will come up with more questions and interesting findings, therefore, a summary will be given at the end of this project.

Preliminary Wrangling

First, let's import the necessary libraries for exploration.

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 sns

%matplotlib inline

import geopy.distance # find distance from longitudes and latitudes

from sklearn.linear_model import LinearRegression # regression model

import warnings
warnings.filterwarnings('ignore')
```

Gather data

Next, I will load the GoBike data set to a pandas dataframe called df_bike.

In [2]:

```
# load the csv data
df_bike = pd.read_csv('201902-fordgobike-tripdata.csv')
```

Assess data

In [3]:

```
# check the head
df_bike.head()
```

Out[3]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_station_longitude
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)	37.789625	-122.400811
1	42521	2019-02-28 18:53:21.7890	2019-03-01 06:42:03.0560	23.0	The Embarcadero at Steuart St	37.791464	-122.391034
2	61854	2019-02-28 12:13:13.2180	2019-03-01 05:24:08.1460	86.0	Market St at Dolores St	37.769305	-122.426826
3	36490	2019-02-28 17:54:26.0100	2019-03-01 04:02:36.8420	375.0	Grove St at Masonic Ave	37.774836	-122.446546
4	1585	2019-02-28 23:54:18.5490	2019-03-01 00:20:44.0740	7.0	Frank H Ogawa Plaza	37.804562	-122.271738

In [4]:

```
# find the time range of our bike data set
print (df_bike['start_time'].min(), df_bike['start_time'].max())
print (df_bike['end_time'].min(), df_bike['end_time'].max())
```

```
2019-02-01 00:00:20.6360 2019-02-28 23:59:18.5480
2019-02-01 00:04:52.0580 2019-03-01 08:01:55.9750
```

As we can see, each observation contains the duration, start/end time, start/end station name/position of the ride, and basic infomation about the rider (gender, birthyear, customer/subscriber). The data set records all the renting activities in February 2019.

In [5]:

```
# check the info
df_bike.info()
```

```
<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
```

memory usage: 22.4+ MB

In this data set, there are 183412 observations and 16 features. There are some missing values in the columns of start/end station id and name, member birthyear and gender . However, in the birthyear and gender columns, we don't necessary need to have values since the riders have the rights not to share their personal information, therefore, missing values in these columns should be fine.

In [6]:

```
# check for duplicate observations
df_bike.duplicated().sum()
```

Out[6]:

```
0
```

It's good we don't have duplicate entries in our data set. Now let's focus on the missing start/end station id and name, first I queried on all null start station id.

In [7]:

```
# check null start_station_id
df_bike[df_bike['start_station_id'].isnull()]
```

Out[7]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_station_long
475	1709	2019-02-28 20:55:53.9320	2019-02-28 21:24:23.7380	NaN	NaN	37.40	-1
1733	1272	2019-02-28 18:32:34.2730	2019-02-28 18:53:46.7270	NaN	NaN	37.40	-1
3625	142	2019-02-28 17:10:46.5290	2019-02-28 17:13:09.4310	NaN	NaN	37.41	-1
4070	585	2019-02-28 16:28:45.9340	2019-02-28 16:38:31.3320	NaN	NaN	37.39	-1
5654	509	2019-02-28 12:30:17.1310	2019-02-28 12:38:46.3290	NaN	NaN	37.40	-1
6214	1334	2019-02-28 10:32:47.9300	2019-02-28 10:55:02.0280	NaN	NaN	37.40	-1
8499	240	2019-02-28 08:23:07.0920	2019-02-28 08:27:07.2890	NaN	NaN	37.41	-1
8783	883	2019-02-28 07:58:07.3720	2019-02-28 08:12:51.2760	NaN	NaN	37.41	-1
10967	116	2019-02-27 19:25:57.0360	2019-02-27 19:27:54.0260	NaN	NaN	37.41	-1
11071	828	2019-02-27 19:03:14.6380	2019-02-27 19:17:03.4340	NaN	NaN	37.40	-1
13945	399	2019-02-27 14:06:51.7720	2019-02-27 14:13:31.6720	NaN	NaN	37.41	-1
14073	402	2019-02-27 13:37:01.8640	2019-02-27 13:43:44.3600	NaN	NaN	37.41	-1
14115	870	2019-02-27 13:17:28.6630	2019-02-27 13:31:59.1710	NaN	NaN	37.40	-1
14276	856	2019-02-27 12:42:38.9310	2019-02-27 12:56:55.0260	NaN	NaN	37.41	-1
15298	890	2019-02-27 09:33:32.5720	2019-02-27 09:48:23.0750	NaN	NaN	37.41	-1
15932	186	2019-02-27 09:01:57.4740	2019-02-27 09:05:03.9760	NaN	NaN	37.41	-1
16746	1005	2019-02-27 08:00:53.5280	2019-02-27 08:17:38.5340	NaN	NaN	37.41	-1

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_station_long
17749	316	2019-02-26 20:50:30.7760	2019-02-26 20:55:47.0760	NaN	NaN	37.41	-1
17908	393	2019-02-26 20:11:35.0800	2019-02-26 20:18:08.6700	NaN	NaN	37.40	-1
18573	142	2019-02-26 18:22:48.7720	2019-02-26 18:25:10.7740	NaN	NaN	37.41	-1
18639	954	2019-02-26 18:02:37.2290	2019-02-26 18:18:32.0650	NaN	NaN	37.40	-1
19033	913	2019-02-26 17:29:26.5680	2019-02-26 17:44:39.7700	NaN	NaN	37.40	-1
19932	327	2019-02-26 15:00:30.5340	2019-02-26 15:05:58.0320	NaN	NaN	37.41	-1
21233	170	2019-02-26 09:30:48.1350	2019-02-26 09:33:38.6190	NaN	NaN	37.41	-1
22306	1123	2019-02-26 08:05:08.6320	2019-02-26 08:23:52.0330	NaN	NaN	37.41	-1
22893	301	2019-02-26 02:10:14.2240	2019-02-26 02:15:15.7340	NaN	NaN	37.41	-1
23515	352	2019-02-25 20:13:00.9730	2019-02-25 20:18:53.0700	NaN	NaN	37.41	-1
24102	150	2019-02-25 18:53:35.2670	2019-02-25 18:56:05.3730	NaN	NaN	37.41	-1
24429	806	2019-02-25 18:11:33.0340	2019-02-25 18:24:59.5310	NaN	NaN	37.41	-1
24539	997	2019-02-25 17:57:40.5330	2019-02-25 18:14:18.3300	NaN	NaN	37.40	-1
...	...	...	...	...	...	...	
130572	1200	2019-02-09 16:36:30.3070	2019-02-09 16:56:31.2460	NaN	NaN	37.40	-1
130846	481	2019-02-09 15:09:44.3840	2019-02-09 15:17:45.7220	NaN	NaN	37.42	-1
130944	339	2019-02-09 14:40:39.5040	2019-02-09 14:46:18.8620	NaN	NaN	37.41	-1
131475	527	2019-02-09 12:24:39.8020	2019-02-09 12:33:26.9450	NaN	NaN	37.42	-1
131502	795	2019-02-09 12:10:29.1640	2019-02-09 12:23:44.7060	NaN	NaN	37.40	-1
131575	2233	2019-02-09 11:26:46.8750	2019-02-09 12:04:00.0710	NaN	NaN	37.41	-1
131645	303	2019-02-09 11:28:57.1210	2019-02-09 11:34:01.0220	NaN	NaN	37.42	-1
131750	276	2019-02-09 11:02:12.2670	2019-02-09 11:06:49.1040	NaN	NaN	37.41	-1
132593	439	2019-02-08 23:52:25.8470	2019-02-08 23:59:44.9270	NaN	NaN	37.42	-1
135455	6725	2019-02-08 11:06:42.2950	2019-02-08 12:58:48.2570	NaN	NaN	37.41	-1
136317	1927	2019-02-08 09:41:14.3240	2019-02-08 10:13:22.0890	NaN	NaN	37.41	-1
137030	838	2019-02-08 08:57:23.1240	2019-02-08 09:11:21.5430	NaN	NaN	37.41	-1
142719	2031	2019-02-07 16:23:38.5680	2019-02-07 16:57:29.7450	NaN	NaN	37.41	-1
142864	3064	2019-02-07 15:53:33.2350	2019-02-07 16:44:37.8200	NaN	NaN	37.41	-1
144928	349	2019-02-07	2019-02-07	NaN	NaN	37.42	-1

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_station_long
145154	1503	2019-02-07 09:49:31.9250	2019-02-07 10:14:35.8680	NaN	NaN	37.41	-1
152178	366	2019-02-06 16:20:37.1020	2019-02-06 16:26:43.1650	NaN	NaN	37.41	-1
152648	660	2019-02-06 15:07:33.9600	2019-02-06 15:18:34.5620	NaN	NaN	37.40	-1
153088	789	2019-02-06 13:32:34.6330	2019-02-06 13:45:44.3340	NaN	NaN	37.41	-1
154695	968	2019-02-06 09:19:56.8780	2019-02-06 09:36:05.2380	NaN	NaN	37.41	-1
155470	984	2019-02-06 08:39:02.6480	2019-02-06 08:55:26.6830	NaN	NaN	37.41	-1
161400	1159	2019-02-05 14:53:30.4310	2019-02-05 15:12:49.4460	NaN	NaN	37.40	-1
161844	823	2019-02-05 13:28:05.5940	2019-02-05 13:41:49.3090	NaN	NaN	37.41	-1
172481	3791	2019-02-03 14:53:55.9470	2019-02-03 15:57:07.0860	NaN	NaN	37.40	-1
174807	8209	2019-02-02 14:26:59.9650	2019-02-02 16:43:49.5180	NaN	NaN	37.40	-1
176154	1447	2019-02-02 12:03:04.5440	2019-02-02 12:27:12.2670	NaN	NaN	37.40	-1
179730	309	2019-02-01 12:59:45.9690	2019-02-01 13:04:55.4260	NaN	NaN	37.40	-1
179970	659	2019-02-01 12:17:37.6750	2019-02-01 12:28:37.0140	NaN	NaN	37.41	-1
180106	2013	2019-02-01 11:33:55.1470	2019-02-01 12:07:28.9400	NaN	NaN	37.40	-1
181201	312	2019-02-01 09:26:34.8030	2019-02-01 09:31:46.9210	NaN	NaN	37.40	-1

197 rows × 16 columns

In [8]:

```
# count on the latitude of the null start station id
df_bike[df_bike['start_station_id'].isnull()]['start_station_latitude'].value_counts()
```

Out[8]:

```
37.41    101
37.40     63
37.42     21
37.39     11
37.38      1
Name: start_station_latitude, dtype: int64
```

In [9]:

```
# count on the longitude of the null start station id
df_bike[df_bike['start_station_id'].isnull()]['start_station_longitude'].value_counts()
```

Out[9]:

```
-121.94    57
-121.96    46
-121.93    46
-121.95    37
-121.92    10
-121.98     1
Name: start_station_longitude, dtype: int64
```

The missing start station ids seem to have similar latitudes (37.38 to 37.42) and longitudes (-121.98 to -121.92). Compared with those with start station id, these values are less accurate as well as they have less digits after the decimal point.

In [10]:

```
# check the null end_station_id
df_bike[df_bike['end_station_id'].isnull()]
```

Out[10]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_station_long
475	1709	2019-02-28 20:55:53.9320	2019-02-28 21:24:23.7380	NaN	NaN	37.40	-1
1733	1272	2019-02-28 18:32:34.2730	2019-02-28 18:53:46.7270	NaN	NaN	37.40	-1
3625	142	2019-02-28 17:10:46.5290	2019-02-28 17:13:09.4310	NaN	NaN	37.41	-1
4070	585	2019-02-28 16:28:45.9340	2019-02-28 16:38:31.3320	NaN	NaN	37.39	-1
5654	509	2019-02-28 12:30:17.1310	2019-02-28 12:38:46.3290	NaN	NaN	37.40	-1
6214	1334	2019-02-28 10:32:47.9300	2019-02-28 10:55:02.0280	NaN	NaN	37.40	-1
8499	240	2019-02-28 08:23:07.0920	2019-02-28 08:27:07.2890	NaN	NaN	37.41	-1
8783	883	2019-02-28 07:58:07.3720	2019-02-28 08:12:51.2760	NaN	NaN	37.41	-1
10967	116	2019-02-27 19:25:57.0360	2019-02-27 19:27:54.0260	NaN	NaN	37.41	-1
11071	828	2019-02-27 19:03:14.6380	2019-02-27 19:17:03.4340	NaN	NaN	37.40	-1
13945	399	2019-02-27 14:06:51.7720	2019-02-27 14:13:31.6720	NaN	NaN	37.41	-1
14073	402	2019-02-27 13:37:01.8640	2019-02-27 13:43:44.3600	NaN	NaN	37.41	-1
14115	870	2019-02-27 13:17:28.6630	2019-02-27 13:31:59.1710	NaN	NaN	37.40	-1
14276	856	2019-02-27 12:42:38.9310	2019-02-27 12:56:55.0260	NaN	NaN	37.41	-1
15298	890	2019-02-27 09:33:32.5720	2019-02-27 09:48:23.0750	NaN	NaN	37.41	-1
15932	186	2019-02-27 09:01:57.4740	2019-02-27 09:05:03.9760	NaN	NaN	37.41	-1
16746	1005	2019-02-27 08:00:53.5280	2019-02-27 08:17:38.5340	NaN	NaN	37.41	-1
17749	316	2019-02-26 20:50:30.7760	2019-02-26 20:55:47.0760	NaN	NaN	37.41	-1
17908	393	2019-02-26 20:11:35.0800	2019-02-26 20:18:08.6700	NaN	NaN	37.40	-1
18573	142	2019-02-26 18:22:48.7720	2019-02-26 18:25:10.7740	NaN	NaN	37.41	-1
18639	954	2019-02-26 18:02:37.2290	2019-02-26 18:18:32.0650	NaN	NaN	37.40	-1
19033	913	2019-02-26 17:29:26.5680	2019-02-26 17:44:39.7700	NaN	NaN	37.40	-1
19932	327	2019-02-26 15:00:30.5340	2019-02-26 15:05:58.0320	NaN	NaN	37.41	-1

21233	**duration_sec**	**start_time**	**end_time**	**start_station_id**	**start_station_name**	**start_station_latitude**	**start_station_long**
	170	2019-02-26 09:30:48.1350	2019-02-26 09:33:38.6190	NaN	NaN	37.41	-1
22306	1123	2019-02-26 08:05:08.6320	2019-02-26 08:23:52.0330	NaN	NaN	37.41	-1
22893	301	2019-02-26 02:10:14.2240	2019-02-26 02:15:15.7340	NaN	NaN	37.41	-1
23515	352	2019-02-25 20:13:00.9730	2019-02-25 20:18:53.0700	NaN	NaN	37.41	-1
24102	150	2019-02-25 18:53:35.2670	2019-02-25 18:56:05.3730	NaN	NaN	37.41	-1
24429	806	2019-02-25 18:11:33.0340	2019-02-25 18:24:59.5310	NaN	NaN	37.41	-1
24539	997	2019-02-25 17:57:40.5330	2019-02-25 18:14:18.3300	NaN	NaN	37.40	-1
...	...	...	...	...	...	...	
130572	1200	2019-02-09 16:36:30.3070	2019-02-09 16:56:31.2460	NaN	NaN	37.40	-1
130846	481	2019-02-09 15:09:44.3840	2019-02-09 15:17:45.7220	NaN	NaN	37.42	-1
130944	339	2019-02-09 14:40:39.5040	2019-02-09 14:46:18.8620	NaN	NaN	37.41	-1
131475	527	2019-02-09 12:24:39.8020	2019-02-09 12:33:26.9450	NaN	NaN	37.42	-1
131502	795	2019-02-09 12:10:29.1640	2019-02-09 12:23:44.7060	NaN	NaN	37.40	-1
131575	2233	2019-02-09 11:26:46.8750	2019-02-09 12:04:00.0710	NaN	NaN	37.41	-1
131645	303	2019-02-09 11:28:57.1210	2019-02-09 11:34:01.0220	NaN	NaN	37.42	-1
131750	276	2019-02-09 11:02:12.2670	2019-02-09 11:06:49.1040	NaN	NaN	37.41	-1
132593	439	2019-02-08 23:52:25.8470	2019-02-08 23:59:44.9270	NaN	NaN	37.42	-1
135455	6725	2019-02-08 11:06:42.2950	2019-02-08 12:58:48.2570	NaN	NaN	37.41	-1
136317	1927	2019-02-08 09:41:14.3240	2019-02-08 10:13:22.0890	NaN	NaN	37.41	-1
137030	838	2019-02-08 08:57:23.1240	2019-02-08 09:11:21.5430	NaN	NaN	37.41	-1
142719	2031	2019-02-07 16:23:38.5680	2019-02-07 16:57:29.7450	NaN	NaN	37.41	-1
142864	3064	2019-02-07 15:53:33.2350	2019-02-07 16:44:37.8200	NaN	NaN	37.41	-1
144928	349	2019-02-07 10:39:21.0240	2019-02-07 10:45:10.3850	NaN	NaN	37.42	-1
145154	1503	2019-02-07 09:49:31.9250	2019-02-07 10:14:35.8680	NaN	NaN	37.41	-1
152178	366	2019-02-06 16:20:37.1020	2019-02-06 16:26:43.1650	NaN	NaN	37.41	-1
152648	660	2019-02-06 15:07:33.9600	2019-02-06 15:18:34.5620	NaN	NaN	37.40	-1
153088	789	2019-02-06 13:32:34.6330	2019-02-06 13:45:44.3340	NaN	NaN	37.41	-1
154695	968	2019-02-06 09:19:56.8780	2019-02-06 09:36:05.2380	NaN	NaN	37.41	-1
155470	984	2019-02-06 08:39:02.6480	2019-02-06 08:55:26.6830	NaN	NaN	37.41	-1

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_station_long
161400	1159	2019-02-05 14:53:30.4310	2019-02-05 15:12:49.4460	NaN	NaN	37.40	-1
161844	823	2019-02-05 13:28:05.5940	2019-02-05 13:41:49.3090	NaN	NaN	37.41	-1
172481	3791	2019-02-03 14:53:55.9470	2019-02-03 15:57:07.0860	NaN	NaN	37.40	-1
174807	8209	2019-02-02 14:26:59.9650	2019-02-02 16:43:49.5180	NaN	NaN	37.40	-1
176154	1447	2019-02-02 12:03:04.5440	2019-02-02 12:27:12.2670	NaN	NaN	37.40	-1
179730	309	2019-02-01 12:59:45.9690	2019-02-01 13:04:55.4260	NaN	NaN	37.40	-1
179970	659	2019-02-01 12:17:37.6750	2019-02-01 12:28:37.0140	NaN	NaN	37.41	-1
180106	2013	2019-02-01 11:33:55.1470	2019-02-01 12:07:28.9400	NaN	NaN	37.40	-1
181201	312	2019-02-01 09:26:34.8030	2019-02-01 09:31:46.9210	NaN	NaN	37.40	-1

197 rows × 16 columns

In [11]:

```
# count on the latitude of the null end station id
df_bike[df_bike['end_station_id'].isnull()]['end_station_latitude'].value_counts()
```

Out[11]:

```
37.41    98
37.40    65
37.42    18
37.39    13
37.38     2
37.43     1
Name: end_station_latitude, dtype: int64
```

In [12]:

```
# count on the longitude of the null end station id
df_bike[df_bike['end_station_id'].isnull()]['end_station_longitude'].value_counts()
```

Out[12]:

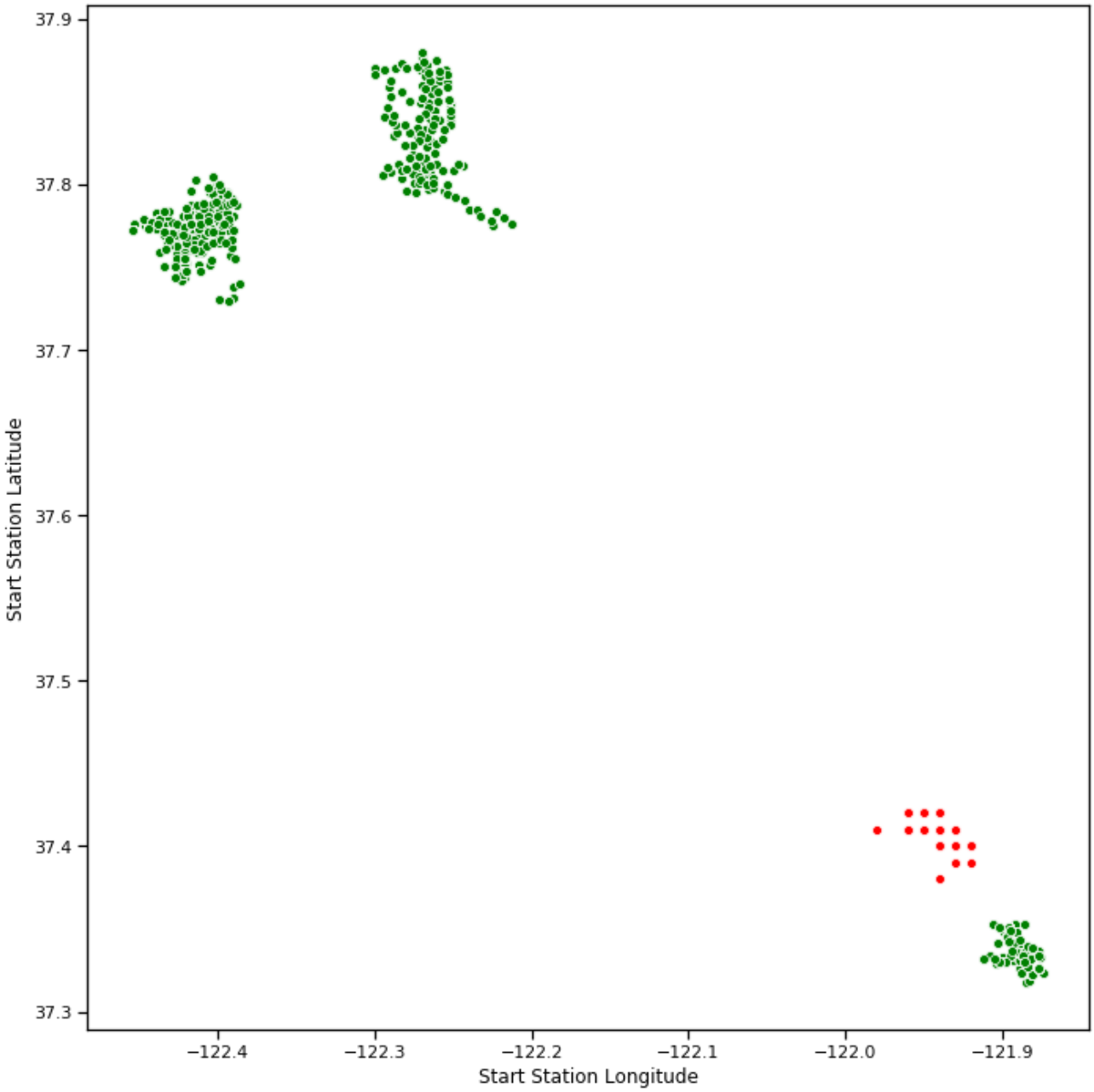
```
-121.94    55
-121.93    50
-121.96    46
-121.95    33
-121.92    11
-121.98     2
Name: end_station_longitude, dtype: int64
```

Similar issue with the null end station ids, they all have similar latitudes (37.38 to 37.42) and longitudes (-121.98 to -121.92). I decided to plot the latitudes and longitudes of all the observations and see the pattern.

In [13]:

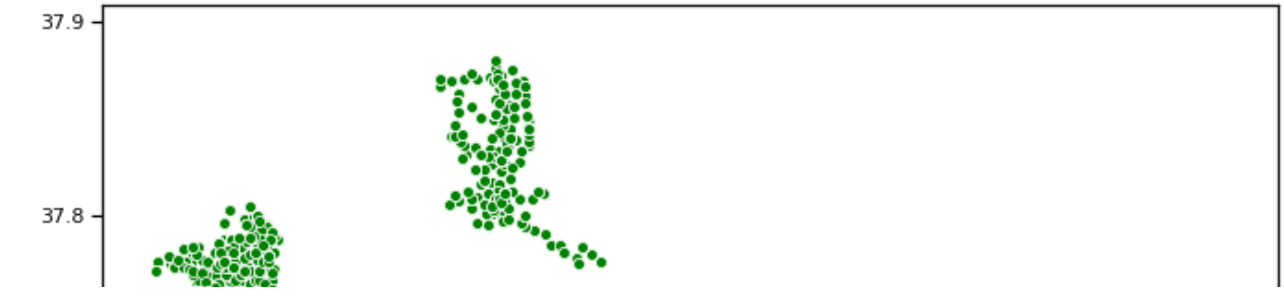
```
# plot all latitudes and longitudes, those with missing ids in red
# first on start stations
with sns.plotting_context("notebook"):
    # create a matplotlib figure
    fig, ax = plt.subplots(figsize = (10,10))
    sns.scatterplot(y = "start_station_latitude", x = "start_station_longitude", data =
df_bike[df_bike['start_station_id'].isnull()], color = 'r')
    sns.scatterplot(y = "start_station_latitude", x = "start_station_longitude", data =
df_bike.dropna(subset=["start_station_id"]), color = 'g')
```

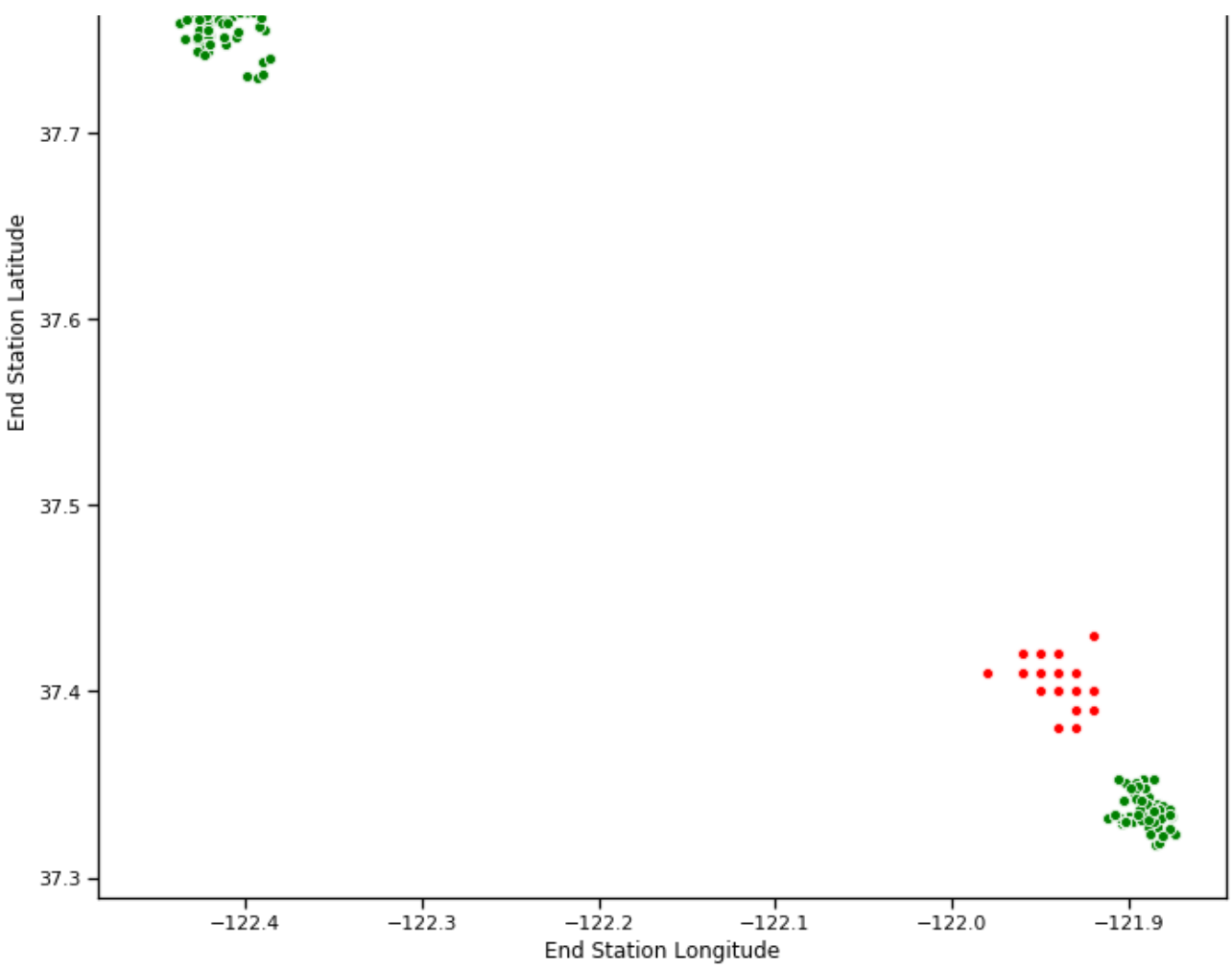
```
    plt.ylabel('Start Station Latitude')
    plt.xlabel('Start Station Longitude')
    plt.tight_layout()
```



In [14]:

```
# plot all latitudes and longitudes, those with missing ids in red
# second on end stations
with sns.plotting_context("notebook"):
    # create a matplotlib figure
    fig, ax = plt.subplots(figsize = (10,10))
    sns.scatterplot(y = "end_station_latitude", x = "end_station_longitude", data = df_b
ike[df_bike['end_station_id'].isnull()], color = 'r')
    sns.scatterplot(y = "end_station_latitude", x = "end_station_longitude", data = df_b
ike.dropna(subset=["end_station_id"]), color = 'g')
    plt.ylabel('End Station Latitude')
    plt.xlabel('End Station Longitude')
    plt.tight_layout()
```





We can see 4 clusters in both plots, one of them contains all the start/end stations with missing ids and the rest three looks fine. As we discussed before, siince these points are less accurate (with fewer decimal digits), we will drop them in the cleaning section.

In [15]:

```
# check the statistics of the numerical columns
df_bike.describe()
```

Out[15]:

	duration_sec	start_station_id	start_station_latitude	start_station_longitude	end_station_id	end_station_latitude	end_s
count	183412.000000	183215.000000	183412.000000	183412.000000	183215.000000	183412.000000	
mean	726.078435	138.590427	37.771223	-122.352664	136.249123	37.771427	
std	1794.389780	111.778864	0.099581	0.117097	111.515131	0.099490	
min	61.000000	3.000000	37.317298	-122.453704	3.000000	37.317298	
25%	325.000000	47.000000	37.770083	-122.412408	44.000000	37.770407	
50%	514.000000	104.000000	37.780760	-122.398285	100.000000	37.781010	
75%	796.000000	239.000000	37.797280	-122.286533	235.000000	37.797320	
max	85444.000000	398.000000	37.880222	-121.874119	398.000000	37.880222	

In the duration_sec column, the max duration is 85444 seconds (23.73 hours), this seems to be unrealistic, let's check on that.

In [16]:

```
# check the max duration_sec
df_bike.query("duration_sec == 85444")
```

Out[16]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_station_long
101361	85444	2019-02-13 17:59:55.1240	2019-02-14 17:43:59.9540	5.0	Powell St BART Station (Market St at 5th St)	37.783899	-122.4

It seems to me the rider might just forgot to return the bike properly. Now, I will assess all the columns related to the start station.

In [17]:

```
# extract all the start station columns
df_start_station = df_bike[["start_station_id", "start_station_name", "start_station_lati
tude", "start_station_longitude"]]
# drop duplicates and null values
df_start_station.drop_duplicates(inplace = True)
df_start_station.dropna(inplace = True)
df_start_station.sort_values("start_station_id", inplace = True)
df_start_station.reset_index(inplace = True, drop = True)
```

In [18]:

```
# df_start_station includes all the info of the start stations
df_start_station
```

Out[18]:

	start_station_id	start_station_name	start_station_latitude	start_station_longitude
0	3.0	Powell St BART Station (Market St at 4th St)	37.786375	-122.404904
1	4.0	Cyril Magnin St at Ellis St	37.785881	-122.408915
2	5.0	Powell St BART Station (Market St at 5th St)	37.783899	-122.408445
3	6.0	The Embarcadero at Sansome St	37.804770	-122.403234
4	7.0	Frank H Ogawa Plaza	37.804562	-122.271738
5	8.0	The Embarcadero at Vallejo St	37.799953	-122.398525
6	9.0	Broadway at Battery St	37.798572	-122.400869
7	10.0	Washington St at Kearny St	37.795393	-122.404770
8	11.0	Davis St at Jackson St	37.797280	-122.398436
9	13.0	Commercial St at Montgomery St	37.794231	-122.402923
10	14.0	Clay St at Battery St	37.795001	-122.399970
11	15.0	San Francisco Ferry Building (Harry Bridges Pl...	37.795392	-122.394203
12	16.0	Steuart St at Market St	37.794130	-122.394430
13	17.0	Embarcadero BART Station (Beale St at Market St)	37.792251	-122.397086
14	18.0	Telegraph Ave at Alcatraz Ave	37.850222	-122.260172
15	19.0	Post St at Kearny St	37.788975	-122.403452
16	20.0	Mechanics Monument Plaza (Market St at Bush St)	37.791300	-122.399051
17	21.0	Montgomery St BART Station (Market St at 2nd St)	37.789625	-122.400811
18	22.0	Howard St at Beale St	37.789756	-122.394643
19	23.0	The Embarcadero at Steuart St	37.791464	-122.391034
20	24.0	Spear St at Folsom St	37.789677	-122.390428
21	25.0	Howard St at 2nd St	37.787522	-122.397405
22	26.0	1st St at Folsom St	37.787290	-122.394380
23	27.0	Beale St at Harrison St	37.788059	-122.391865

	start_station_id	start_station_name	start_station_latitude	start_station_longitude
24	28.0	The Embarcadero at Bryant St	37.787168	-122.388098
25	29.0	O'Farrell St at Divisadero St	37.782405	-122.439446
26	30.0	San Francisco Caltrain (Townsend St at 4th St)	37.776598	-122.395282
27	31.0	Raymond Kimbell Playground	37.783813	-122.434559
28	33.0	Golden Gate Ave at Hyde St	37.781650	-122.415408
29	34.0	Father Alfred E Boeddeker Park	37.783988	-122.412408
...	...	...	...	...
299	350.0	8th St at Brannan St	37.771431	-122.405787
300	351.0	10th St at University Ave	37.869060	-122.293400
301	355.0	23rd St at Tennessee St	37.755367	-122.388795
302	356.0	Valencia St at Clinton Park	37.769188	-122.422285
303	357.0	2nd St at Julian St	37.341132	-121.892844
304	358.0	Williams Ave at 3rd St	37.729279	-122.392896
305	359.0	Williams Ave at Apollo St	37.730168	-122.398963
306	360.0	Newhall St at 3rd St	37.738572	-122.389618
307	361.0	Mendell St at Fairfax Ave	37.739853	-122.385655
308	362.0	Lane St at Revere Ave	37.731727	-122.390056
309	363.0	Salesforce Transit Center (Natoma St at 2nd St)	37.787492	-122.398285
310	364.0	China Basin St at 3rd St	37.772000	-122.389970
311	365.0	Turk St at Fillmore St	37.780450	-122.431946
312	368.0	Myrtle St at Polk St	37.785434	-122.419622
313	369.0	Hyde St at Post St	37.787349	-122.416651
314	370.0	Jones St at Post St	37.787327	-122.413278
315	371.0	Lombard St at Columbus Ave	37.802746	-122.413579
316	372.0	Madison St at 17th St	37.804037	-122.262409
317	373.0	Potrero del Sol Park (25th St at Utah St)	37.751792	-122.405216
318	375.0	Grove St at Masonic Ave	37.774836	-122.446546
319	377.0	Fell St at Stanyan St	37.771917	-122.453704
320	378.0	Empire St at 7th St	37.347745	-121.890800
321	380.0	Masonic Ave at Turk St	37.779047	-122.447291
322	381.0	20th St at Dolores St	37.758238	-122.426094
323	383.0	Golden Gate Ave at Franklin St	37.780787	-122.421934
324	385.0	Woolsey St at Sacramento St	37.850578	-122.278175
325	386.0	24th St at Bartlett St	37.752105	-122.419724
326	388.0	Backesto Park (Jackson St at 13th St)	37.352887	-121.886050
327	389.0	Taylor St at 9th St	37.353062	-121.891937
328	398.0	Leavenworth St at Broadway	37.796471	-122.416858

329 rows × 4 columns

In [19]:

```
# double check there should not be duplicates
df_start_station[df_start_station.duplicated("start_station_name")]
```

Out[19]:

start_station_id	start_station_name	start_station_latitude	start_station_longitude

As a result, there are 329 different start stations in ourt data set.

In [20]:

```
# similarly we find all the end station list
df_end_station = df_bike[["end_station_id", "end_station_name", "end_station_latitude",
"end_station_longitude"]]
# get rid of duplicates and nulls
df_end_station.drop_duplicates(inplace = True)
df_end_station.dropna(inplace = True)
df_end_station.sort_values("end_station_id", inplace = True)
df_end_station.reset_index(inplace = True, drop = True)
```

In [21]:

```
# all the end stations info
df_end_station
```

Out[21]:

	end_station_id	end_station_name	end_station_latitude	end_station_longitude
0	3.0	Powell St BART Station (Market St at 4th St)	37.786375	-122.404904
1	4.0	Cyril Magnin St at Ellis St	37.785881	-122.408915
2	5.0	Powell St BART Station (Market St at 5th St)	37.783899	-122.408445
3	6.0	The Embarcadero at Sansome St	37.804770	-122.403234
4	7.0	Frank H Ogawa Plaza	37.804562	-122.271738
5	8.0	The Embarcadero at Vallejo St	37.799953	-122.398525
6	9.0	Broadway at Battery St	37.798572	-122.400869
7	10.0	Washington St at Kearny St	37.795393	-122.404770
8	11.0	Davis St at Jackson St	37.797280	-122.398436
9	13.0	Commercial St at Montgomery St	37.794231	-122.402923
10	14.0	Clay St at Battery St	37.795001	-122.399970
11	15.0	San Francisco Ferry Building (Harry Bridges Pl...	37.795392	-122.394203
12	16.0	Steuart St at Market St	37.794130	-122.394430
13	17.0	Embarcadero BART Station (Beale St at Market St)	37.792251	-122.397086
14	18.0	Telegraph Ave at Alcatraz Ave	37.850222	-122.260172
15	19.0	Post St at Kearny St	37.788975	-122.403452
16	20.0	Mechanics Monument Plaza (Market St at Bush St)	37.791300	-122.399051
17	21.0	Montgomery St BART Station (Market St at 2nd St)	37.789625	-122.400811
18	22.0	Howard St at Beale St	37.789756	-122.394643
19	23.0	The Embarcadero at Steuart St	37.791464	-122.391034
20	24.0	Spear St at Folsom St	37.789677	-122.390428
21	25.0	Howard St at 2nd St	37.787522	-122.397405
22	26.0	1st St at Folsom St	37.787290	-122.394380
23	27.0	Beale St at Harrison St	37.788059	-122.391865
24	28.0	The Embarcadero at Bryant St	37.787168	-122.388098
25	29.0	O'Farrell St at Divisadero St	37.782405	-122.439446
26	30.0	San Francisco Caltrain (Townsend St at 4th St)	37.776598	-122.395282
27	31.0	Raymond Kimbell Playground	37.783813	-122.434559
28	33.0	Golden Gate Ave at Hyde St	37.781650	-122.415408
29	34.0	Father Alfred E Boeddeker Park	37.783988	-122.412408

...	end_station_id	end_station_name	end_station_latitude	end_station_longitude
299	350.0	8th St at Brannan St	37.771431	-122.405787
300	351.0	10th St at University Ave	37.869060	-122.293400
301	355.0	23rd St at Tennessee St	37.755367	-122.388795
302	356.0	Valencia St at Clinton Park	37.769188	-122.422285
303	357.0	2nd St at Julian St	37.341132	-121.892844
304	358.0	Williams Ave at 3rd St	37.729279	-122.392896
305	359.0	Williams Ave at Apollo St	37.730168	-122.398963
306	360.0	Newhall St at 3rd St	37.738572	-122.389618
307	361.0	Mendell St at Fairfax Ave	37.739853	-122.385655
308	362.0	Lane St at Revere Ave	37.731727	-122.390056
309	363.0	Salesforce Transit Center (Natoma St at 2nd St)	37.787492	-122.398285
310	364.0	China Basin St at 3rd St	37.772000	-122.389970
311	365.0	Turk St at Fillmore St	37.780450	-122.431946
312	368.0	Myrtle St at Polk St	37.785434	-122.419622
313	369.0	Hyde St at Post St	37.787349	-122.416651
314	370.0	Jones St at Post St	37.787327	-122.413278
315	371.0	Lombard St at Columbus Ave	37.802746	-122.413579
316	372.0	Madison St at 17th St	37.804037	-122.262409
317	373.0	Potrero del Sol Park (25th St at Utah St)	37.751792	-122.405216
318	375.0	Grove St at Masonic Ave	37.774836	-122.446546
319	377.0	Fell St at Stanyan St	37.771917	-122.453704
320	378.0	Empire St at 7th St	37.347745	-121.890800
321	380.0	Masonic Ave at Turk St	37.779047	-122.447291
322	381.0	20th St at Dolores St	37.758238	-122.426094
323	383.0	Golden Gate Ave at Franklin St	37.780787	-122.421934
324	385.0	Woolsey St at Sacramento St	37.850578	-122.278175
325	386.0	24th St at Bartlett St	37.752105	-122.419724
326	388.0	Backesto Park (Jackson St at 13th St)	37.352887	-121.886050
327	389.0	Taylor St at 9th St	37.353062	-121.891937
328	398.0	Leavenworth St at Broadway	37.796471	-122.416858

329 rows × 4 columns

In [22]:

```
# double check no duplicates in end stations
df_end_station[df_end_station.duplicated("end_station_name")]
```

Out[22]:

end_station_id	end_station_name	end_station_latitude	end_station_longitude

Once again, we have 329 different end stations, I am wondering whether there are stations that only allow rent (can't return) or return (can't rent).

In [23]:

```
# check the start station list is the same as the end station list
start_station_list = list(df_start_station['start_station_name'])
end_station_list = list(df_end_station['end_station_name'])
start_station_list == end_station_list
```

Out[23]:

```
True
```

Since the start station list is the same as the end station list, we know all the 329 stations allow both bike rent and return.

Data Assessing Summary:

- **We should change the data type of start_time and end_time to datetime.**
- **There are missing values in the start/end station_id and start/end station_name, we should drop them since their latitude and longitude values are less accurate.**

Clean data

Define:

From the data assessing summary, there are two cleaning steps to do:

- **Change data type of start_time and end_time to datetime.**
- **Drop rows with missing start/end station_id and start/end station_name.**

Code:

In [24]:

```
# make a copy of the original data
df_bike_copy = df_bike.copy()
```

In [25]:

```
# drop rows with missing station id and name
df_bike_copy.dropna(subset = ["start_station_id"], inplace = True)
df_bike_copy.dropna(subset = ["end_station_id"], inplace = True)
```

In [26]:

```
# change data type to datetime
df_bike_copy['start_time'] = pd.to_datetime(df_bike_copy['start_time'])
df_bike_copy['end_time'] = pd.to_datetime(df_bike_copy['end_time'])
```

Test:

In [27]:

```
df_bike_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 183215 entries, 0 to 183411
Data columns (total 16 columns):
duration_sec               183215 non-null int64
start_time                 183215 non-null datetime64[ns]
end_time                   183215 non-null datetime64[ns]
start_station_id           183215 non-null float64
start_station_name         183215 non-null object
start_station_latitude     183215 non-null float64
start_station_longitude    183215 non-null float64
end_station_id             183215 non-null float64
end_station_name           183215 non-null object
end_station_latitude       183215 non-null float64
end_station_longitude      183215 non-null float64
bike_id                    183215 non-null int64
user_type                  183215 non-null object
member_birth_year          174952 non-null float64
member_gender              174952 non-null object
bike_share_for_all_trip    183215 non-null object
```

```
dtypes: datetime64[ns](2), float64(7), int64(2), object(5)
memory usage: 23.8+ MB
```

Now that we have the cleaned data, I will save it as a csv file called GoBike_Cleaned.csv for future use.

In [28]:

```
# save the cleaned data
df_bike_copy.to_csv("GoBike_Cleaned.csv", index = False)
```

Univariate Exploration

In this section, I investigated the distributions of individual variables, and they include:

- **The distribution of trip duration.**
- **The distribution of trip distance.**
- **The distribution of user's age.**
- **The distribution of user's gender.**
- **The distribution of user's type.**
- **Renting counts on day of week.**
- **Renting counts on hour of day.**

In [29]:

```
# check on dftaframe head
df_bike_copy.head()
```

Out[29]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_station_longitude	e
0	52185	2019-02-28 17:32:10.145	2019-03-01 08:01:55.975	21.0	Montgomery St BART Station (Market St at 2nd St)	37.789625	-122.400811	
1	42521	2019-02-28 18:53:21.789	2019-03-01 06:42:03.056	23.0	The Embarcadero at Steuart St	37.791464	-122.391034	
2	61854	2019-02-28 12:13:13.218	2019-03-01 05:24:08.146	86.0	Market St at Dolores St	37.769305	-122.426826	
3	36490	2019-02-28 17:54:26.010	2019-03-01 04:02:36.842	375.0	Grove St at Masonic Ave	37.774836	-122.446546	
4	1585	2019-02-28 23:54:18.549	2019-03-01 00:20:44.074	7.0	Frank H Ogawa Plaza	37.804562	-122.271738	

(1) The distribution of trip duration

In [30]:

```
# duration in second statistics
df_bike_copy['duration_sec'].describe()
```

Out[30]:

```
count    183215.000000
mean        725.902017
std        1795.078654
min          61.000000
25%         325.000000
50%         514.000000
75%         796.000000
max       85444.000000
Name: duration_sec, dtype: float64
```

We see from here and also from the wrangling section that, the max duration is unrealistically long (85444 seconds), and this point is an outlier in the duration_sec column. In order not to include outliers in our distribution plot, I decided to first look at the .99 percentile.

In [31]:

```
df_bike_copy['duration_sec'].quantile(.99)
```

Out[31]:

```
3456.859999999986
```

The .99 percentile has the value of 3457 seconds (~58 min), which is reasonable, given one might indeed use the bike for an hour. Therefore for the duration distribution plot, I will cut off at 3457 second.

In [32]:

```
# new statistics after cutting off at .99 quantile
df_bike_copy.query('duration_sec < 3457')['duration_sec'].describe()
```

Out[32]:

```
count    181382.000000
mean        621.487672
std         442.762656
min          61.000000
25%         323.000000
50%         510.000000
75%         784.000000
max        3456.000000
Name: duration_sec, dtype: float64
```

In [33]:

```
# for subscriber
df_bike_copy.query('(duration_sec < 3457)&(user_type == "Subscriber")')['duration_sec'].d
escribe()
```

Out[33]:

```
count    162645.000000
mean        588.859313
std         410.871196
min          61.000000
25%         312.000000
50%         489.000000
75%         745.000000
max        3456.000000
Name: duration_sec, dtype: float64
```

In [34]:

```
# for customer
df_bike_copy.query('duration_sec < 3457&(user_type == "Customer")')['duration_sec'].descr
ibe()
```

Out[34]:

```
count    18737.000000
mean       904.715483
std        585.584207
min         62.000000
25%        490.000000
50%        757.000000
75%       1154.000000
max       3456.000000
Name: duration_sec, dtype: float64
```

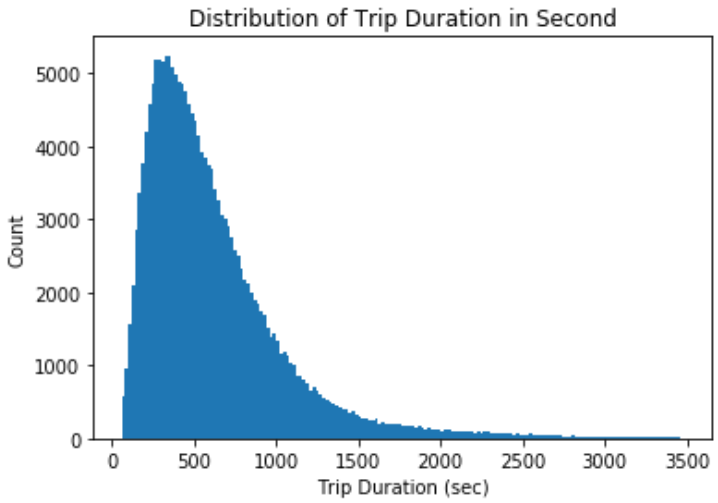
In [35]:

```
# distribution of trip duration
```

```
plt.hist(data = df_bike_copy.query('duration_sec < 3457'), x = 'duration_sec', bins = np
.arange(60, 3500, 20))
plt.xlabel('Trip Duration (sec)')
plt.ylabel('Count')
plt.title('Distribution of Trip Duration in Second')
```

Out[35]:

```
Text(0.5, 1.0, 'Distribution of Trip Duration in Second')
```



Finding: After cutting the trip duration data at .99 percentile, the distribution of trip duration is positively(right)-skewed, with an average value of 621 seconds (10.35 min) on the 181382 observations. 75% of the riders finished their rides within 784 seconds (13.07 min).

(2) The distribution of trip distance

To find each trip's distance (not given directly in the data set), I need to first define a function that calculate distances based on two locations' latitudes and longitudes, and I found a package call geopy.distance that works for our case.

In [36]:

```
# function to calculate distance (in meter) give 2 latitudes and longitudes
def geo_distance(list):
    loc1 = (list[0], list[1])
    loc2 = (list[2], list[3])

    return geopy.distance.distance(loc1, loc2).m
```

In [37]:

```
# apply the dustance function and add distance column to each row
attributes = ['start_station_latitude', 'start_station_longitude', 'end_station_latitude'
, 'end_station_longitude']
df_bike_copy['distance'] = df_bike_copy[attributes].apply(geo_distance, axis = 1)
```

In [38]:

```
# statistics on trip distance
df_bike_copy['distance'].describe()
```

Out[38]:

```
count    183215.000000
mean       1691.247186
std        1097.178647
min           0.000000
25%         909.777251
50%        1431.357962
75%        2227.240841
max       69465.977135
Name: distance, dtype: float64
```

Similar to the trip duration, we also have the issue that the max distance is unrealistically long (69.465 km), I decided to cut off at .99 percentile as well.

In [39]:

```
df_bike_copy['distance'].quantile(.99)
```

Out[39]:

```
5063.37242497863
```

There is one more issue, that if the distance is, e.g. 0 m, it might indicate that the rider eventually didn't use the bike for various reasons (such as failed to unlock the bike, etc.), so I would also like to cut off if the distance is below 1 m.

In [40]:

```
# new statistics after cutting off above 1 meter and below .99 quantile
df_bike_copy.query('(distance > 1) & (distance < 5070)')['distance'].describe()
```

Out[40]:

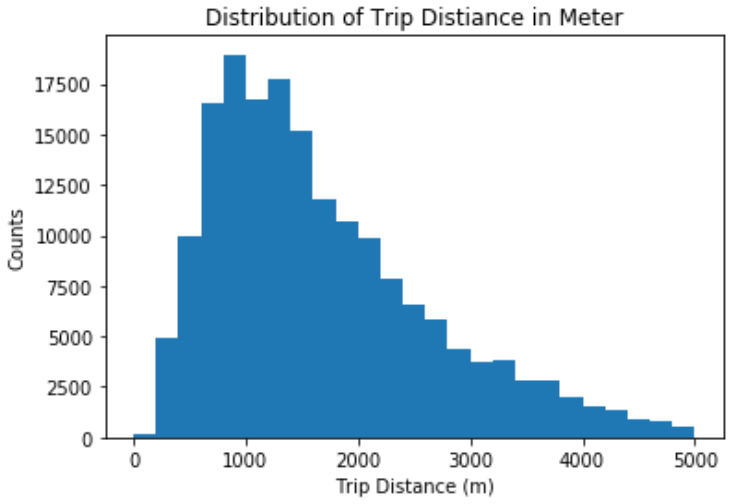
```
count    177564.000000
mean       1682.915003
std         976.655662
min          13.307064
25%         936.749271
50%        1445.277458
75%        2215.371960
max        5067.299998
Name: distance, dtype: float64
```

In [41]:

```
# distribution of trip distance
plt.hist(data = df_bike_copy.query('(distance > 1) & (distance < 5070)'), x = 'distance'
, bins = np.arange(0, 5070, 200))

plt.xlabel('Trip Distance (m)')
plt.ylabel('Counts')
plt.title('Distribution of Trip Distiance in Meter')
```

Out[41]:

```
Text(0.5, 1.0, 'Distribution of Trip Distiance in Meter')
```



Finding: the distribution of the trip distances is also positively(right)-skewed. The average distance of each trip is ~1.68 km. 75% of the trips are within 2.22 km.

I would like to do some extra analysis, focusing on the observations where the trip distances are less than 1 m.

In [42]:

```
# number of observations where distance < 1 m
df_bike_copy.query('distance < 1').shape[0]
```

Out[42]:

```
3822
```

In [43]:

```
# percentage of within-1-meter trip in all the observations
df_bike_copy.query('distance < 1').shape[0]/df_bike_copy.shape[0]
```

Out[43]:

```
0.020860737385039435
```

In [44]:

```
# the distance is < 1 m but the duration is > 10 min
df_bike_copy.query('(distance < 1) & (duration_sec > 600)').shape[0]/df_bike_copy.query(
'distance < 1').shape[0]
```

Out[44]:

```
0.6355311355311355
```

Finding: There are 3822 rides with recorded distance less than 1 m, which accounts for 2.09% of the total observations. However 63.5% of these trips have durations more than 10 min. This could indicate that the riders brought back the bikes to the station where they started their rides.

(3) The distribution of user's age

In [45]:

```
# calculate age from birthyear
df_bike_copy['age'] = 2019 - df_bike_copy['member_birth_year']
```

In [46]:

```
# statistics on age
df_bike_copy['age'].describe()
```

Out[46]:

```
count    174952.000000
mean         34.196865
std          10.118731
min          18.000000
25%          27.000000
50%          32.000000
75%          39.000000
max         141.000000
Name: age, dtype: float64
```

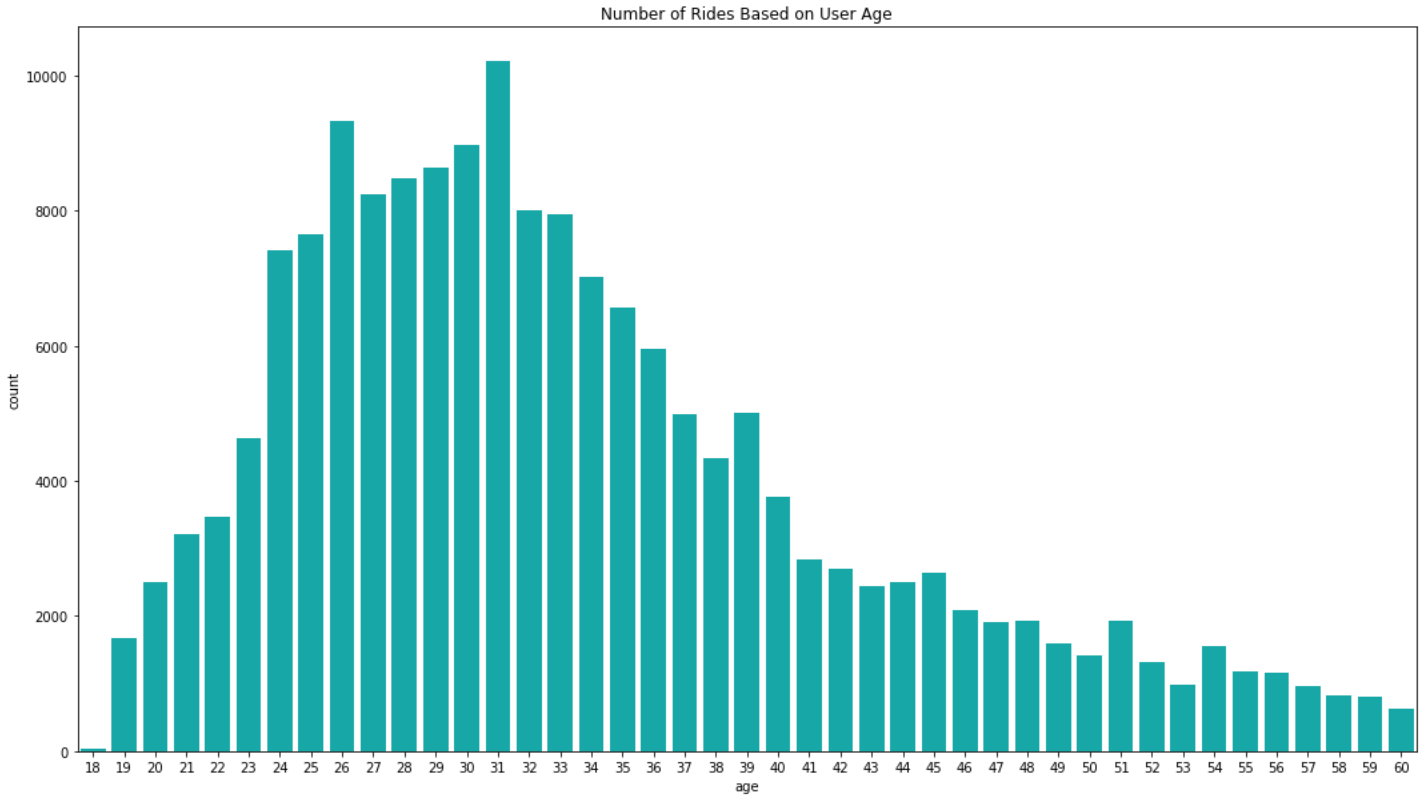
We can see that the user needs to be at least 18 to rent a bike. There are ages, such as 141, that is obviously wrong (the user just entered a random number I guess), so I set the plotting upper limit to 60 years old to see the distribution.

In [47]:

```
# user age distribution
fig, ax = plt.subplots(figsize = (18,10))
ax = sns.countplot(x = "age", data = df_bike_copy.query('age <= 60'), color = 'c')
ax.set_xticklabels(range(18, 61, 1))
ax.set_title('Number of Rides Based on User Age')
```

Out[47]:

```
Text(0.5, 1.0, 'Number of Rides Based on User Age')
```



Finding: From the 174952 non-empty entries on birthyear, we extracted the ages of users. To rent a bike, the user needs to be at least 18 years old, 75% of the users are younger than 39 years old. The highest renting count (more than 10000 times) is from 31-year-old age group, The GoBike is much more popular among young people.

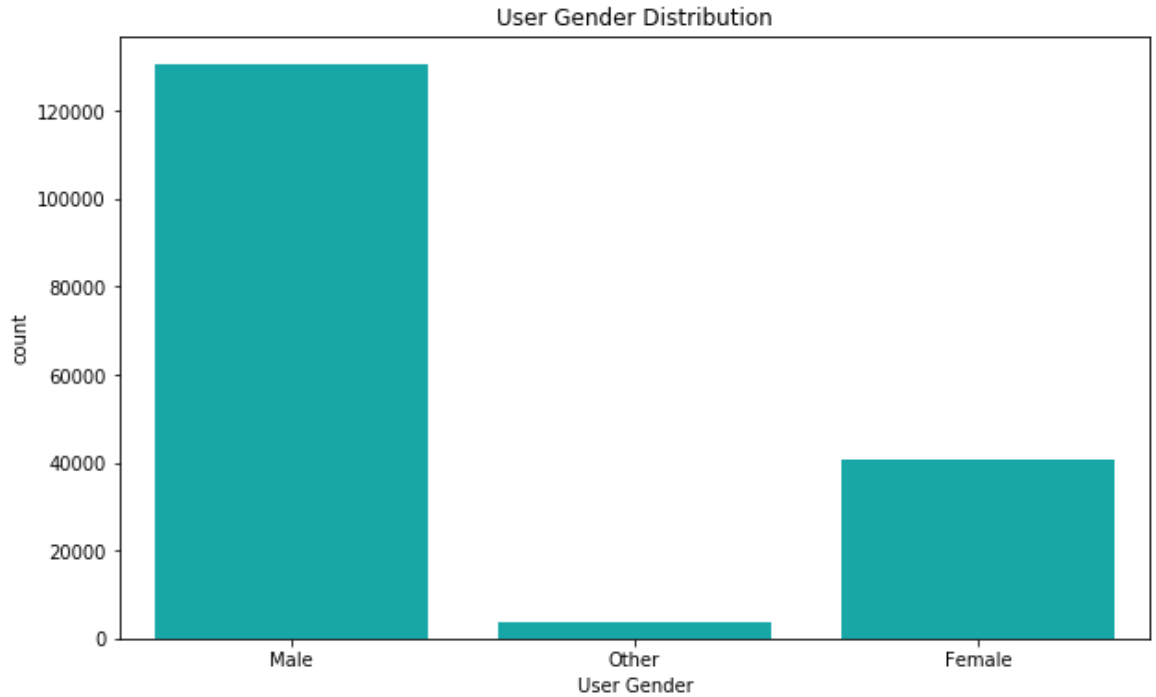
(4) The distribution of user's gender

In [48]:

```
# plot ride counts on genders
fig, ax = plt.subplots(figsize = (10,6))
ax = sns.countplot(x = "member_gender", data = df_bike_copy, color = 'c')
ax.set_title('User Gender Distribution')
ax.set_xlabel('User Gender')
```

Out[48]:

```
Text(0.5, 0, 'User Gender')
```



Finding: From all the observations, male users have used the GoBike much more frequently, and they record

Finding: From all the observations, male users have used the GoBike much more frequently, and they recored ~130000 times of rides. In comparison, female users rided with GoBike for ~40000 times. A couple of thousands rides go to users with other genders.

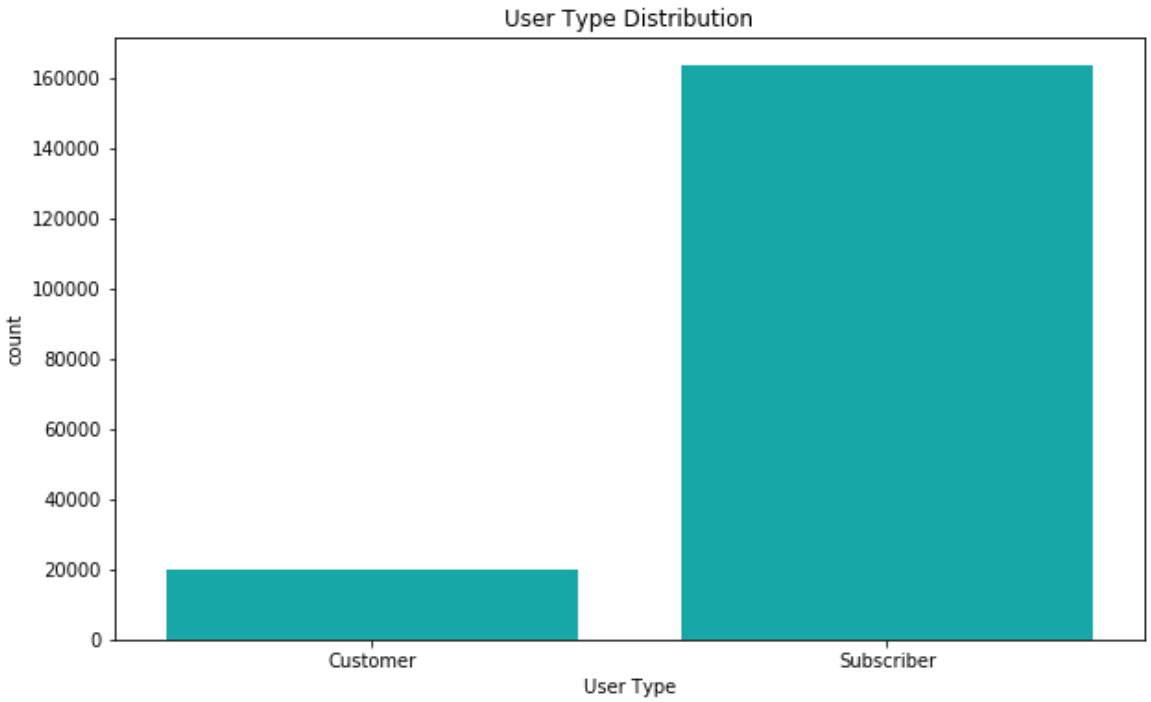
(5) The distribution of user's type

In [49]:

```
# plot ride counts on user type
fig, ax = plt.subplots(figsize = (10,6))
ax = sns.countplot(x = "user_type", data = df_bike_copy, color = 'c')
ax.set_title('User Type Distribution')
ax.set_xlabel('User Type')
```

Out[49]:

```
Text(0.5, 0, 'User Type')
```



In [50]:

```
# user type ratio
customer_ratio = df_bike_copy.query('user_type == "Customer"').shape[0] / df_bike_copy.s
hape[0]
subscriber_ratio = df_bike_copy.query('user_type == "Subscriber"').shape[0] / df_bike_co
py.shape[0]
print (customer_ratio, subscriber_ratio)
```

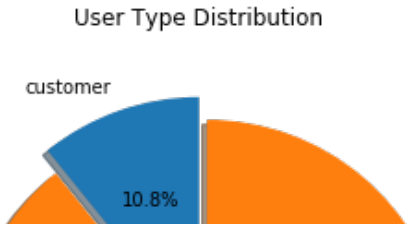
```
0.10807521218240865 0.8919247878175913
```

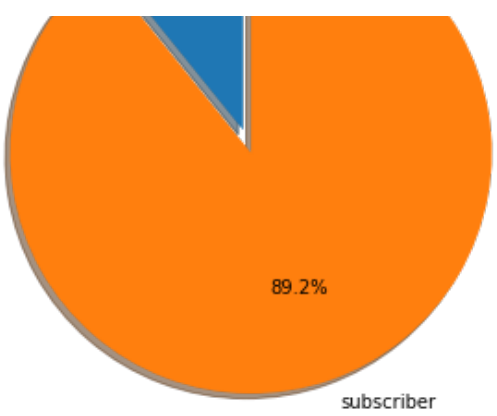
In [51]:

```
pie_data = [customer_ratio, subscriber_ratio]
fig, ax = plt.subplots(figsize = (10,6))
plt.pie(pie_data, explode=[0, 0.1], labels=['customer', 'subscriber'], startangle=90, au
topct='%1.1f%%', shadow=True)
plt.title('User Type Distribution')
```

Out[51]:

```
Text(0.5, 1.0, 'User Type Distribution')
```





Finding: There are two types of users in the data set. From some research, the users can either become a "Subscriber" by paying monthly or yearly membership fees, or become a "Customer" by paying for single trip or for single day. It turns out that more than 89.2% of rental bike records are made by subscribers, 10.9% of rental bike records are made by customers.

(6) Renting counts on day of week

In [52]:

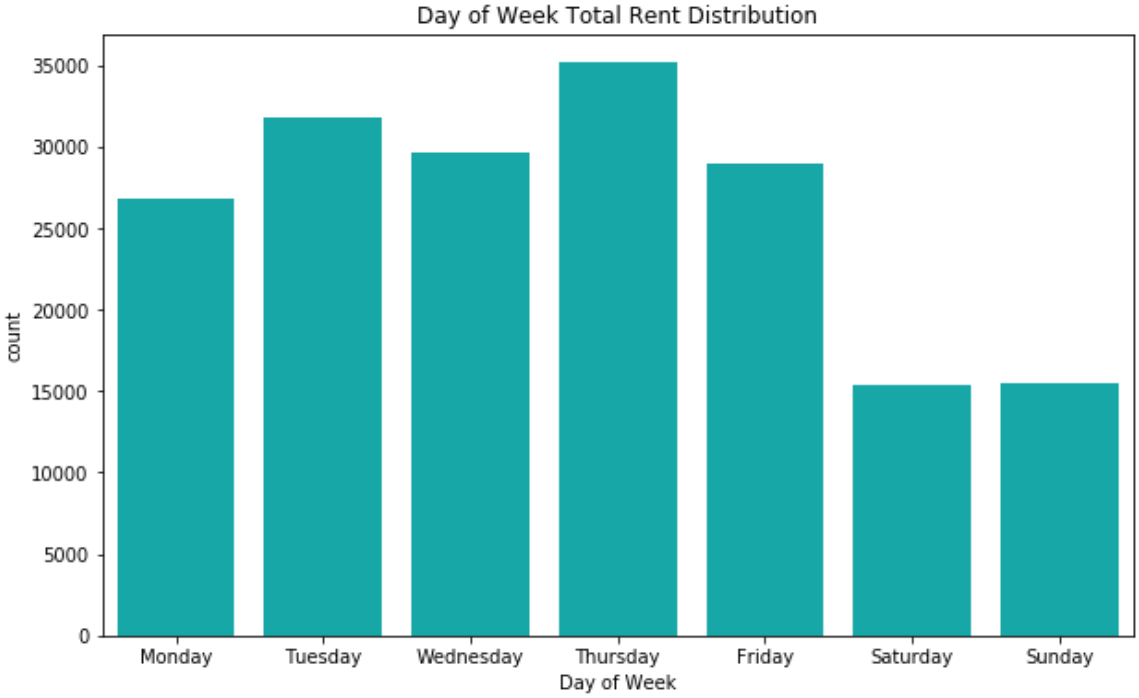
```
# extract day of week from starting time column
df_bike_copy['day_of_week'] = df_bike_copy['start_time'].dt.day_name()
```

In [53]:

```
# distribution of rentals from day of week
fig, ax = plt.subplots(figsize = (10,6))
ax = sns.countplot(x = "day_of_week", data = df_bike_copy, color = 'c',
                   order=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','
Sunday'])
ax.set_title('Day of Week Total Rent Distribution')
ax.set_xlabel('Day of Week')
```

Out[53]:

```
Text(0.5, 0, 'Day of Week')
```



Finding: There are 28 days in Feb 2019, therefore, each day of the week appears 4 times equally. The number of rides is much higher in weekdays than weekend. The plot shows that total rental number is ~15000 times on both Saturday and Sunday, and more than 25000 times in any of the weekdays. Therefore, it seems that the majoriy of bike rides are for work commutions.

(7) Renting counts on hour of day

In [54]:

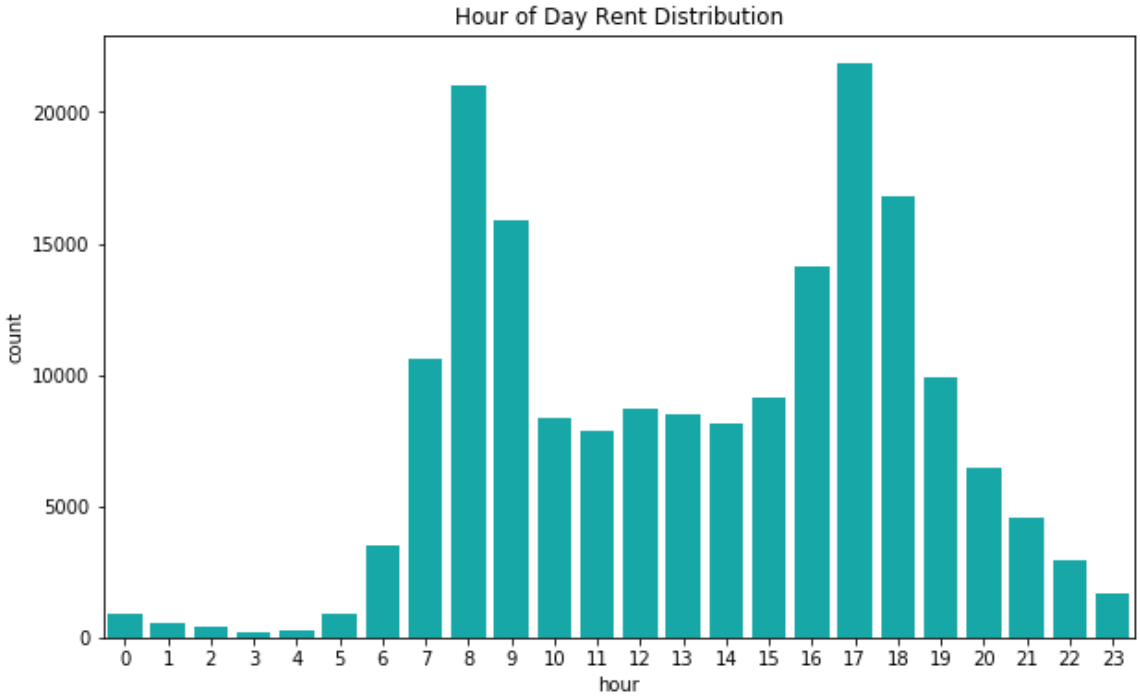
```
# extract hour from start time
df_bike_copy['hour'] = df_bike_copy['start_time'].dt.hour
```

In [55]:

```
# distribution of rentals from hour of day
fig, ax = plt.subplots(figsize = (10,6))
ax = sns.countplot(x = "hour", data = df_bike_copy, color = 'c')
ax.set_title('Hour of Day Rent Distribution')
```

Out[55]:

```
Text(0.5, 1.0, 'Hour of Day Rent Distribution')
```



Finding: More people used GoBike during the rush hours. The plot showsa bimodel distribution, with modes at 8:00 am and 17:00 pm. Therefore, It is clear that most users take the rental bikes for work commuting.

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

- The distribution of **trip duration** is positively(right)-skewed, with an average value of 621 seconds (10.35 min) on the 181382 observations. 75% of the riders finished their rides within 784 seconds (13.07 min).
- the distribution of the **trip distance** is also positively(right)-skewed. The average distance of each trip is ~1.68 km. 75% of the trips are within 2.22 km.
- The distribution of **user age**: to rent a bike, the user needs to be at least 18 years old, 75% of the users are younger than 39 years old. The highest renting count (more than 10000 times) is from 31-year-old age group, The GoBike is much more popular among young people.
- The distribution of **user gender**: from all the observations, male users have used the GoBike much more frequently, and they recored ~130000 times of rides. In comparison, female users rided with GoBike for ~40000 times. A couple of thousands rides go to users with other genders.
- The distribution of **user type**: there are two types of users in the data set. From some research, the users can either become a "Subscriber" by paying monthly or yearly membership fees, or become a "Customer" by paying for single trip or for single day. It turns out that more than 89.2% of rental bike records are made by subscribers, 10.9% of rental bike records are made by customers.
- Rental counts **day of week**: the number of rides is much higher in weekdays than weekend. The plot shows that total rental number is ~15000 times on both Saturday and Sunday, and more than 25000 times in any of the weekdays. Therefore, it seems that the majoriy of bike rides are for work commutions.

- **Rental counts hour of day: more people used GoBike during the rush hours. The plot showsa bimodel distribution, with modes at 8:00 am and 17:00 pm. Therefore, It is clear that most users take the rental bikes for work commuting.**

There are unusual rows where the 'duration_sec' and 'distance' columns have several outliers. I took data up to .99 percentile, and for the distance data excluded points below 1m, I removed the outliers. Because our dataset has a large amount of data points, removing 1% of the data points do not heavily affect the results.

To investigate the day of week and hour of day distribution, I changed the data type of start_time, end_time to datetime, and used the to_datetime method to get day of week and hour information.

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?

I assessed the original data set in various ways to find unusual distributions and parts that need cleaning:

- **First, I have to change the data type of start_time and end_time to datetime in order to get day and hour information.**
- **There are unusual points in the start/end station_id and start/end station_name because the latitudes and latitudes associated with these observations are less accurate than other rows with station id and names, and these rows don't belong to any of the three region clusters that we are interested. Therefore I dropped these rows.**
- **I extracted the trip distance from the geometry locations of the start and end stations using geopy.distance, because I'm interested in the distribution of trip distance which is not given directly.**
- **There are 3822 rides with recorded distance less than 1 m, which accounts for 2.09% of the total observations. Since they won't affect the general results on trip distance, I drop these rows as well. Nevertheless, I did analysis on these rows and found that 63.5% of these trips have durations more than 10 min. This could indicate that the riders brought back the bikes to the station where they started their rides.**
- **I also used the 174952 non-empty entries of birthyear and extracted the ages of users to obtain age distribution.**

Bivariate Exploration

Based on the univariate exploration, I looked further at some relations between different features and they include:

- **Distribution of rentals on day of week by user type**
- **Distribution of rentals on day of week by user gender**
- **Average ride distance by day of month**
- **Average ride duration by day of month**
- **Ride duration distribution at the 5 most visited stations**
- **Relation between ride distance and duration**

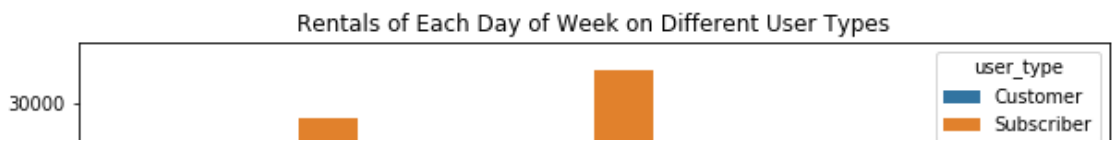
(1) Distribution of rentals on day of week by user type

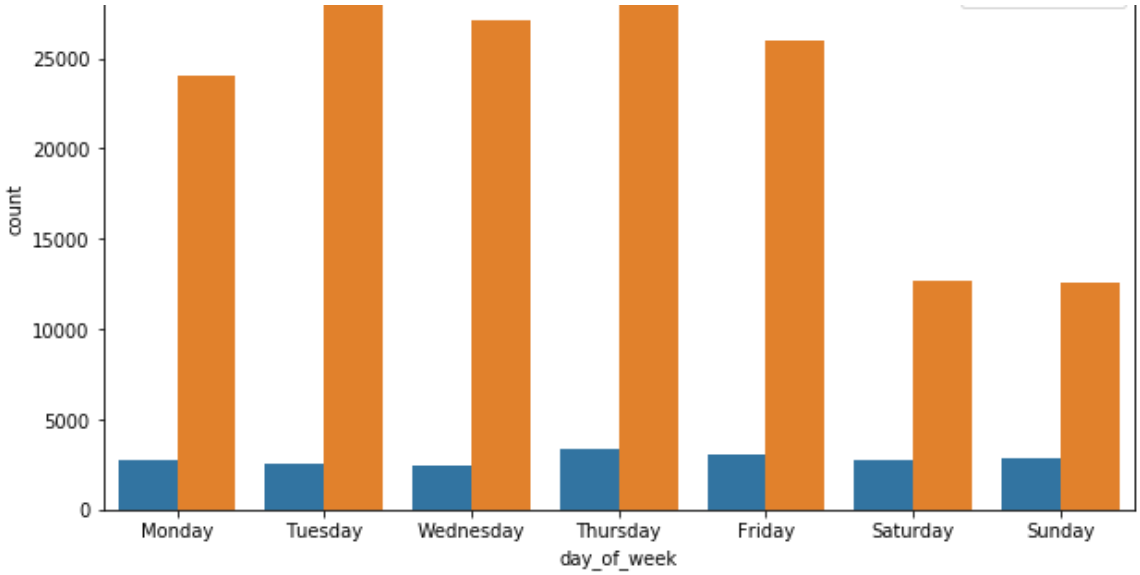
In [56]:

```
# rentals of each day of week on different user types
fig, ax = plt.subplots(figsize = (10,6))
ax = sns.countplot(x = "day_of_week", hue = 'user_type', data = df_bike_copy,
                   order = ['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'
,'Sunday'])
ax.set_title('Rentals of Each Day of Week on Different User Types')
```

Out[56]:

```
Text(0.5, 1.0, 'Rentals of Each Day of Week on Different User Types')
```





Finding: Most of the rentals are made by subscribers compared to customers on each day of the week. For subscribers, we see an interesting pattern that many more rentals were made during weekdays than weekends (>10000 more rentals in total on Weekdays), it's likely that they use GoBike for commuting. Meanwhile for customers, they have fewer (3000 to 4000 in total From Monday to Sunday) rental counts throughout the week than subscribers, and the numbers are pretty stable on each day, e.g., these rentals might partly come from tourists in the Bay area.

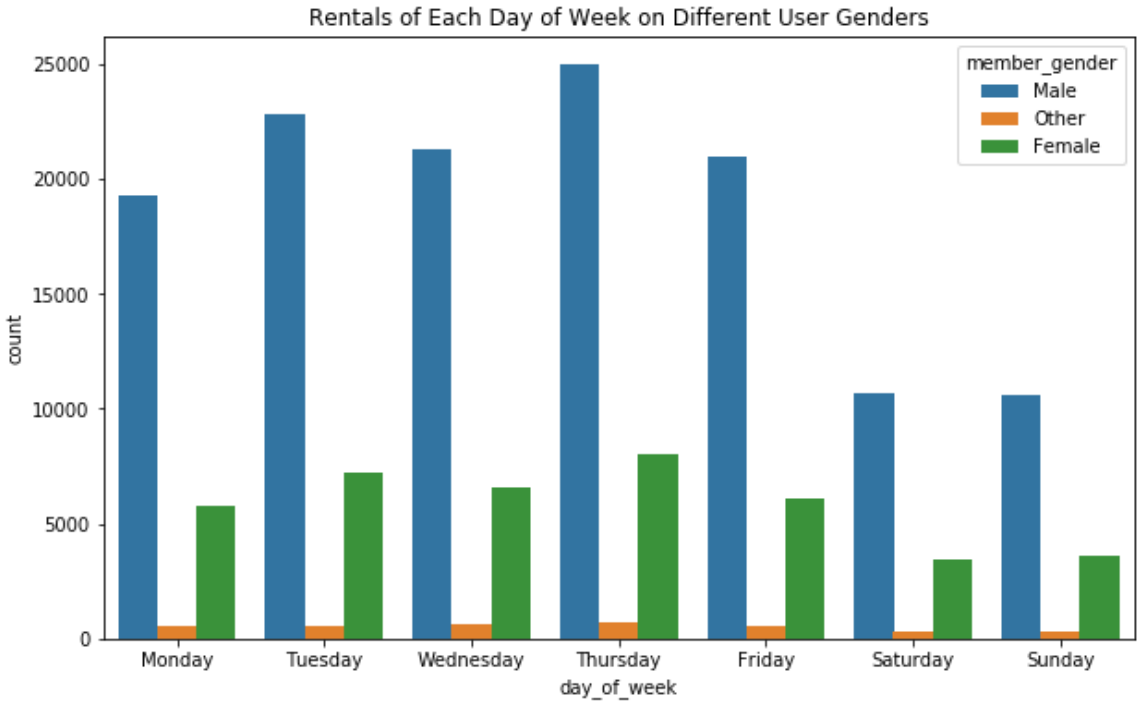
(2) Distribution of rentals on day of week by user gender

In [57]:

```
# rentals of each day of week on different user genders
fig, ax = plt.subplots(figsize = (10,6))
ax = sns.countplot(x = "day_of_week", hue = 'member_gender', data = df_bike_copy,
                   order = ['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'
,'Sunday'])
ax.set_title('Rentals of Each Day of Week on Different User Genders')
```

Out[57]:

```
Text(0.5, 1.0, 'Rentals of Each Day of Week on Different User Genders')
```



Finding: We see similar patterns on day-of-week rentals throughout genders (male/female/other), more uses are recorded on weekdays than weekends (for commuting purpose).

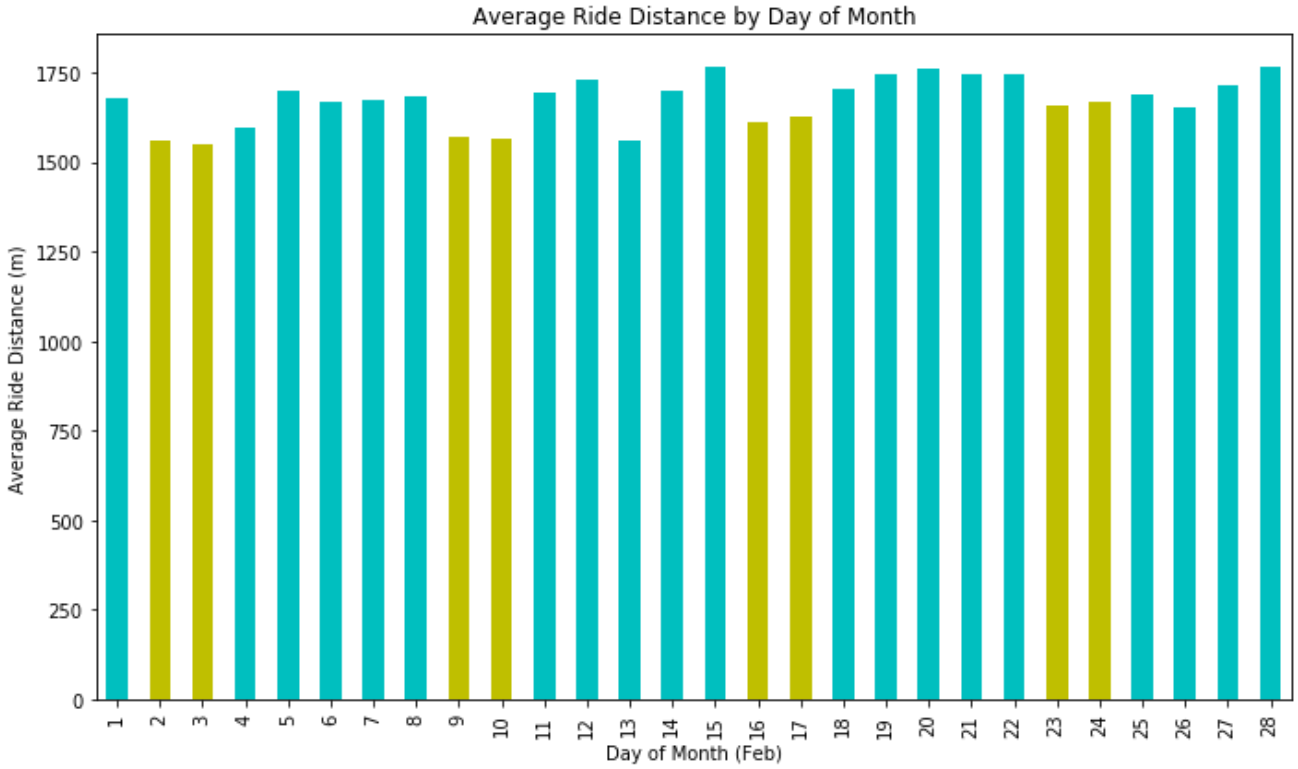
(3) Average ride distance by day of month

In [58]:

```
# add a day column using start_time
df_bike_copy['day'] = df_bike_copy['start_time'].dt.day
```

In [59]:

```
# plotting color: 'y' on weekends, 'c'on weekdays for Feb 2019
colors = ('c','y','y','c','c','c','c','c', 'y','y','c','c','c','c','c',
          'y','y','c','c','c','c','c', 'y','y','c','c','c','c')

# groupby data by day and find the average value
df_avg_distance = df_bike_copy.groupby('day')['distance'].mean()

# visualize using a bar plot
df_avg_distance.plot.bar(figsize = (10, 6), color = colors)
plt.title('Average Ride Distance by Day of Month')
plt.xlabel('Day of Month (Feb)')
plt.ylabel('Average Ride Distance (m)')
plt.tight_layout()
```

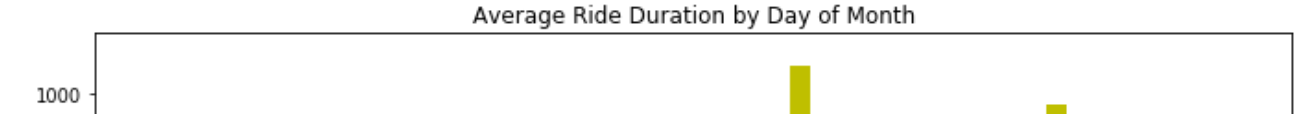


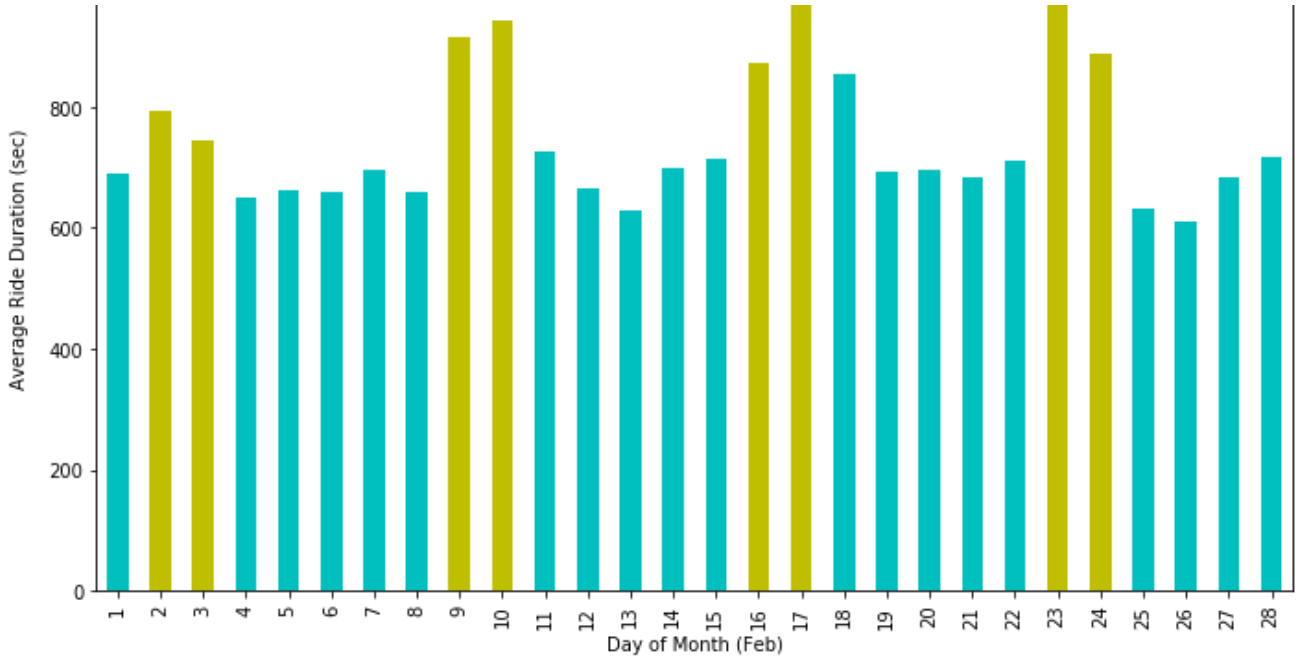
Finding: From the distance vs. day of month graph, on average, the trip distance is longer on weekedays (averaged between 1.7 to 1.8 km) than weekends (averaged between 1.6 to 1.7 km).

(4) Average ride duration by day of month

In [60]:

```
# groupby data by day and find the average value
df_avg_distance = df_bike_copy.groupby('day')['duration_sec'].mean()

# visualize using a bar plot
df_avg_distance.plot.bar(figsize = (10, 6), color = colors)
plt.title('Average Ride Duration by Day of Month')
plt.xlabel('Day of Month (Feb)')
plt.ylabel('Average Ride Duration (sec)')
plt.tight_layout()
```





Finding: We can see that on weekdays the users are going on shorter trips that takes 10 to 12 min, while the average duration on the weekend rises to 13 min to 18 min.

(5) Ride duration distribution at the 5 most visited stations

In [61]:

```
# first count rentals at each start station and find the top 5
top_station = df_bike_copy['start_station_name'].value_counts().index[:5]

# I cut off the duration at 3457 s (.99 percentile) same as before
df_top5_stations = df_bike_copy.query('duration_sec < 3457').loc[df_bike_copy['start_sta
tion_name'].isin(top_station)]
```

In [62]:

```
# show the top stations
top_station
```

Out[62]:

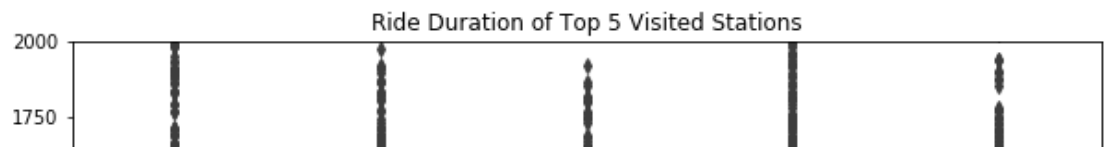
```
Index(['Market St at 10th St',
       'San Francisco Caltrain Station 2  (Townsend St at 4th St)',
       'Berry St at 4th St',
       'Montgomery St BART Station (Market St at 2nd St)',
       'Powell St BART Station (Market St at 4th St)'],
      dtype='object')
```

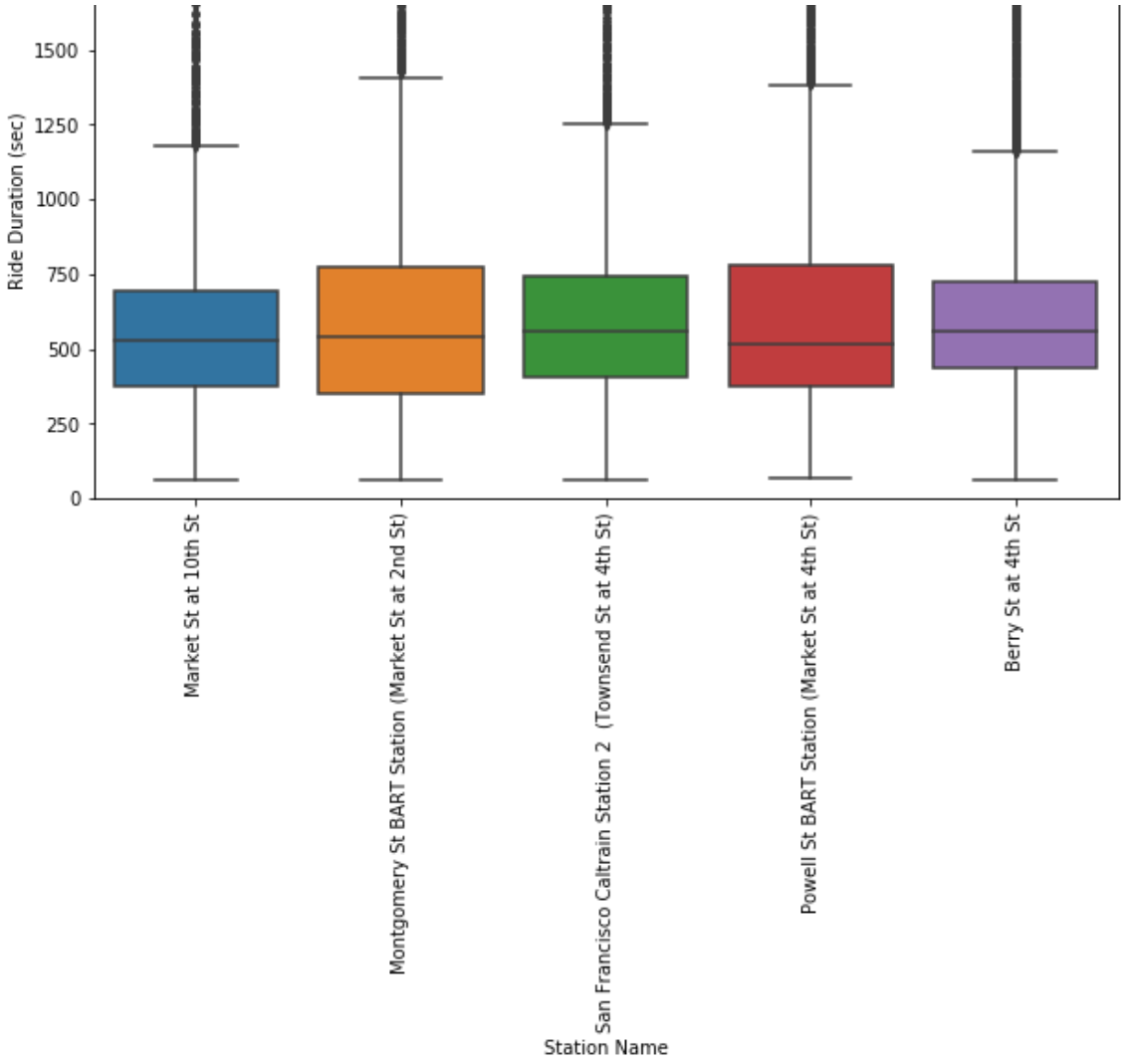
In [63]:

```
# for top 5 starting stations, I use box plot to find the duration distribution
fig, ax = plt.subplots(figsize = (10,6))
ax = sns.boxplot(x = 'start_station_name', y = 'duration_sec', data = df_top5_stations)

plt.title('Ride Duration of Top 5 Visited Stations')
plt.xlabel('Station Name')
plt.ylabel('Ride Duration (sec)')
plt.xticks(rotation = 90)
plt.ylim(0, 2000)
```

Out[63]:

```
(0, 2000)
```





Finding: The top 5 start stations overall have similar statistics in ride duration. They have median numbers in duration ~500 seconds, and the interquartile ranges are all between 250 sec to 400 sec, I also notice some outliers in the data set of each station.

(6) Relation between ride distance and duration

In [64]:

```
# I will use the 10 & 90 percentile of duration sec to tidy the data
print (df_bike_copy['duration_sec'].quantile(.9))
print (df_bike_copy['duration_sec'].quantile(.1))
```

```
1178.0
214.0
```

In [65]:

```
# query the data between .1 and .9 quantile to remove outliers
df_quantiled_bike_copy = df_bike_copy.query('(duration_sec < 1178.0) & (duration_sec > 21
4.0)')

# calculate the correlation coefficient between duration and distance
correlation = df_quantiled_bike_copy['duration_sec'].corr(df_quantiled_bike_copy['distanc
e'])
print (correlation)
```

```
0.7459935605464151
```

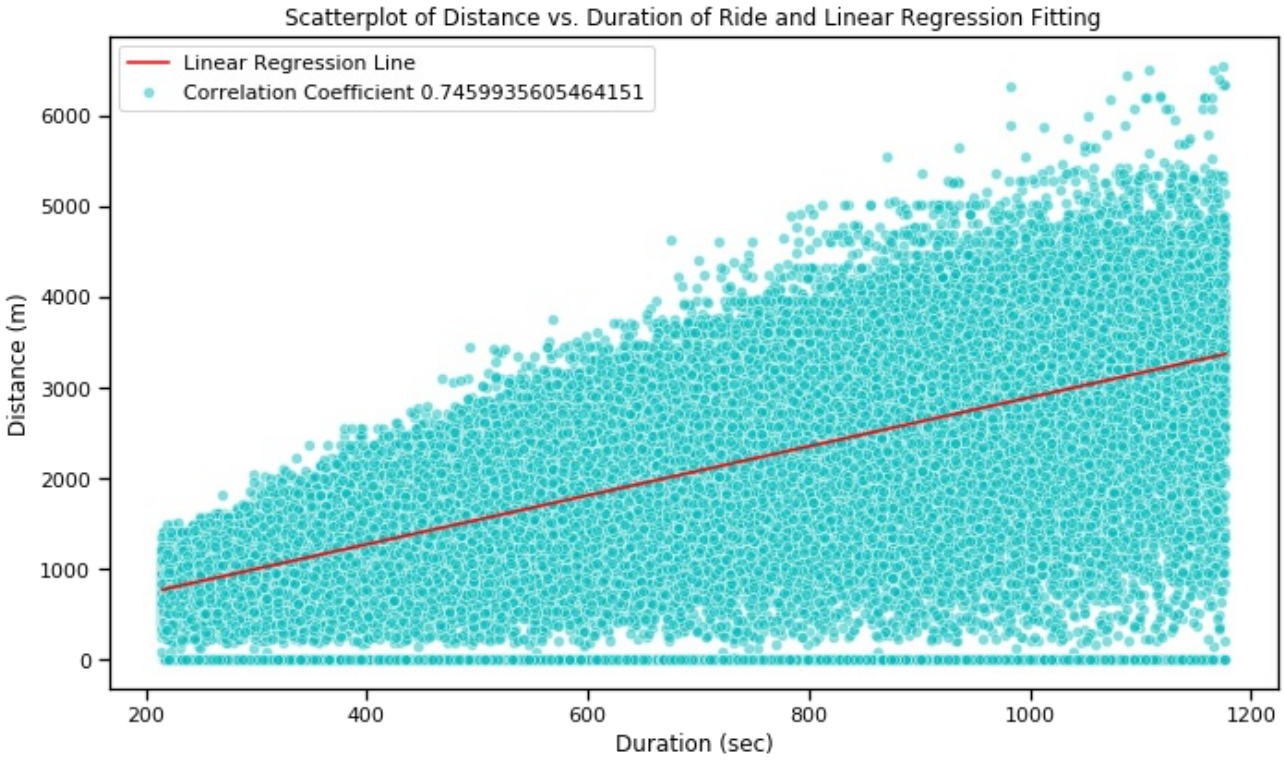
In [66]:

```
# use a linear regression on distance vs duration
regression_model = LinearRegression()
x = df_quantiled_bike_copy[['duration_sec']]
y = df_quantiled_bike_copy[['distance']]
```

```
# fit the data
regression_model.fit(x, y)
# predict the values with the x data to get the trend line
df_quantiled_bike_copy["trend_line"] = regression_model.predict(x)
```

In [67]:

```
# visualize the data as scatterplot and the trend line
with sns.plotting_context("notebook"):
# create a matplotlib figure
    fig, ax = plt.subplots(figsize = (10,6))
    # plot the scatterplot of distance vs duration
    sns.scatterplot(x = "duration_sec", y = "distance", data = df_quantiled_bike_copy,
                    label="Correlation Coefficient {}".format(correlation), alpha = 0.5,
color = "c")
    # plot the trend line
    sns.lineplot(x = "duration_sec", y = "trend_line", data = df_quantiled_bike_copy,
                 label = "Linear Regression Line", color = "r")
    # set the title, legend, x & y labels
    ax.set_title('Scatterplot of Distance vs. Duration of Ride and Linear Regression Fitt
ing')
    ax.legend(loc='best')
    ax.set_xlabel('Duration (sec)')
    ax.set_ylabel('Distance (m)')
    plt.tight_layout()
```



Finding: For the trimmed GoBike data set (use observations between the .1 and .9 quantile based on diration sec), we can see that the trip distance and duration are positively highly correlated with a coefficient of 0.75. It's as expected, the longer the distance, the more time it takes.

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?

After the exploration between variable, I have the following observations:

- **By comparing day-of-week rentals between subscribers and consumers, most of the rentals are made by subscribers on each day of the week.**
- **We see similar patterns on day-of-week rentals throughout genders (male/female/other), more uses are recorded on weekdays than weekends (for commuting purpose).**
- **From the distance vs. day of month graph, on average, the trip distance is longer on weekedays (averaged between 1.7 to 1.8 km) than weekends (averaged between 1.6 to 1.7 km).**
- **We can see that on weekdays the users are going on shorter trips that takes 10 to 12 min, while the average**

duration on the **weekend** rises to 13 min to 18 min.

- The **top 5 start stations** overall have similar statistics in ride **duration**. They have median numbers in duration ~500 seconds, and the interquartile ranges are all between 250 sec to 400 sec, I also notice some outliers in the data set of each station.
- For the trimmed GoBike data set (use observations between the .1 and .9 quantile based on diration sec), we can see that the **trip distance** and **duration** are positively highly correlated with a coefficient of 0.75. It's as expected, the longer the distance, the more time it takes.

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

There are a few interesting relationships between features that I'd like to point out:

- For subscribers, we see an interesting pattern when we look at their rentals in each day of week, that many more rentals were made during weekdays than weekends (>10000 more rentals in total on Weekdays), it's likely that they use GoBike for commuting. Meanwhile for customers, though they have fewer (3000 to 4000 in total From Monday to Sunday) rental counts throughout the week than subscribers, and the numbers are pretty stable on each day, e.g., these rentals might partly come from tourists in the Bay area.
- If we compare the average trip distance in weekdays and weekends, the trip distance is longer on weekedays (averaged between 1.7 to 1.8 km) than weekends (averaged between 1.6 to 1.7 km). However, the duration behaves opposite, the users are going on shorter trips that takes 10 to 12 min on weekdays, while the average duration on the weekend rises to 13 min to 18 min.

Multivariate Exploration

Extended from univariate and bivariate explorations, now we investigate multi-variables, I'm interested in the following:

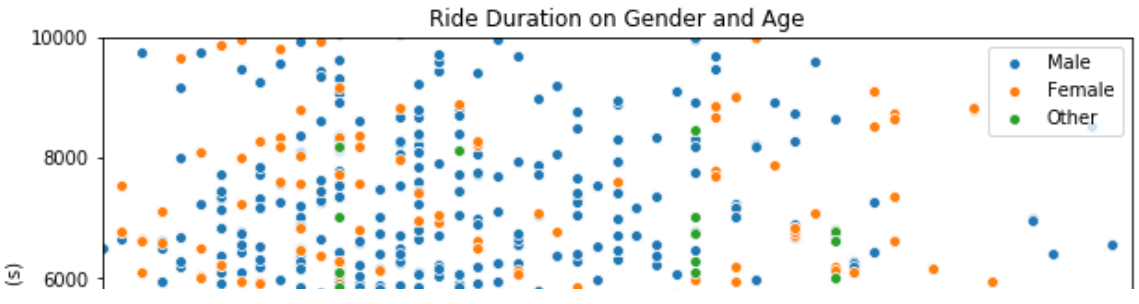
- Distribution of ride duration based on gender and age
- Number of rides during day of week per age group
- Average trip distance during each hour of day in different regions (I will define regions later in this section)
- Average trip duration during each hour of day in different regions

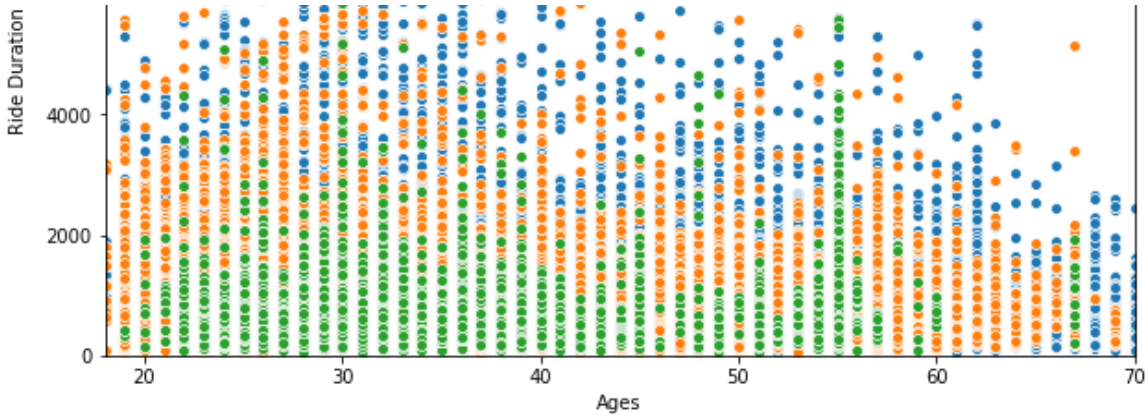
Note that in the univariate exploration section, when we plotted the latitudes and longitudes of all observations, we obsered 4 clusters (one deleted due to missing values), I will define each cluster more carefully in this section and provide more analysis.

(1) Distribution of ride duration based on gender and age

In [68]:

```
# scatterplot of duration on gender and age
plt.figure(figsize = (10,6))

# query each gender subset and plot separately
for gender in ['Male', 'Female', 'Other']:
    df_gender = df_bike_copy[df_bike_copy['member_gender'] == gender]
    sns.scatterplot(x = "age", y = "duration_sec", data = df_gender)
    plt.legend(['Male', 'Female', 'Other'])
    plt.xlim(18, 70)
    plt.ylim(0, 10000)
    plt.title('Ride Duration on Gender and Age')
    plt.xlabel('Ages')
    plt.ylabel('Ride Duration (s)')
```





Finding: It seems that ride durations from other genders are less than ride durations from females and males on average throughout different ages. For data points that have ride duration values larger than 4000s (67 min, likely to be outliers), most of them come from male users (blue), which indicates males might be "less careful" than females to return the bikes.

(2) Number of rides during day of week per age group

In [69]:

```
# first add age group column

age_bins = [0, 19, 29, 39, 49, 59, 69, 79, 89, 99]
age_labels = ['10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80-89', '9(
-99']

df_bike_copy['age_group'] = pd.cut(df_bike_copy['age'], bins = age_bins, labels = age_la
bels, right = False)
df_bike_copy[['age','age_group']].head()
```

Out[69]:

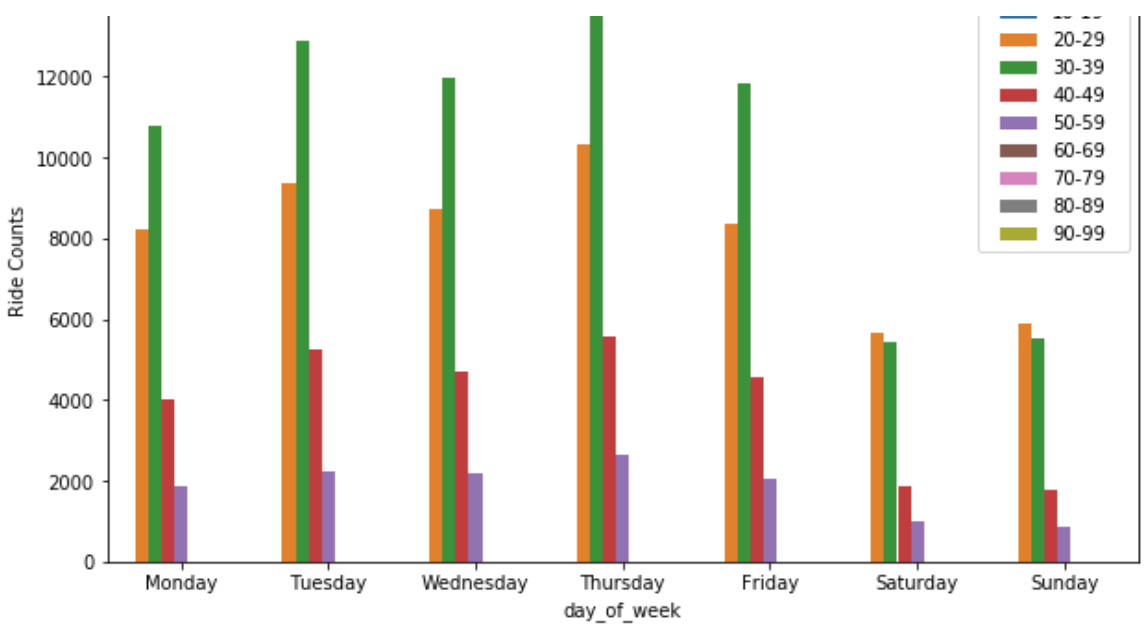
	age	age_group
0	35.0	30-39
1	NaN	NaN
2	47.0	40-49
3	30.0	30-39
4	45.0	40-49

In [70]:

```
# rides counts on age group and day of week, save in df_age_dow
df_age_dow = df_bike_copy.loc[df_bike_copy['age_group'].isin(['10-19', '20-29', '30-39',
'40-49', '50-59'])]

# visualization
plt.figure(figsize = (10, 6))

ax = sns.countplot(data = df_age_dow, x = 'day_of_week', hue = 'age_group',
                   order=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','
Sunday'])

plt.title('Number of Rides during Day of Week per Age Group')
plt.legend(title = 'Age Groupings', loc = 'upper right')
plt.ylabel('Ride Counts')
```

Out[70]:

```
Text(0, 0.5, 'Ride Counts')
```



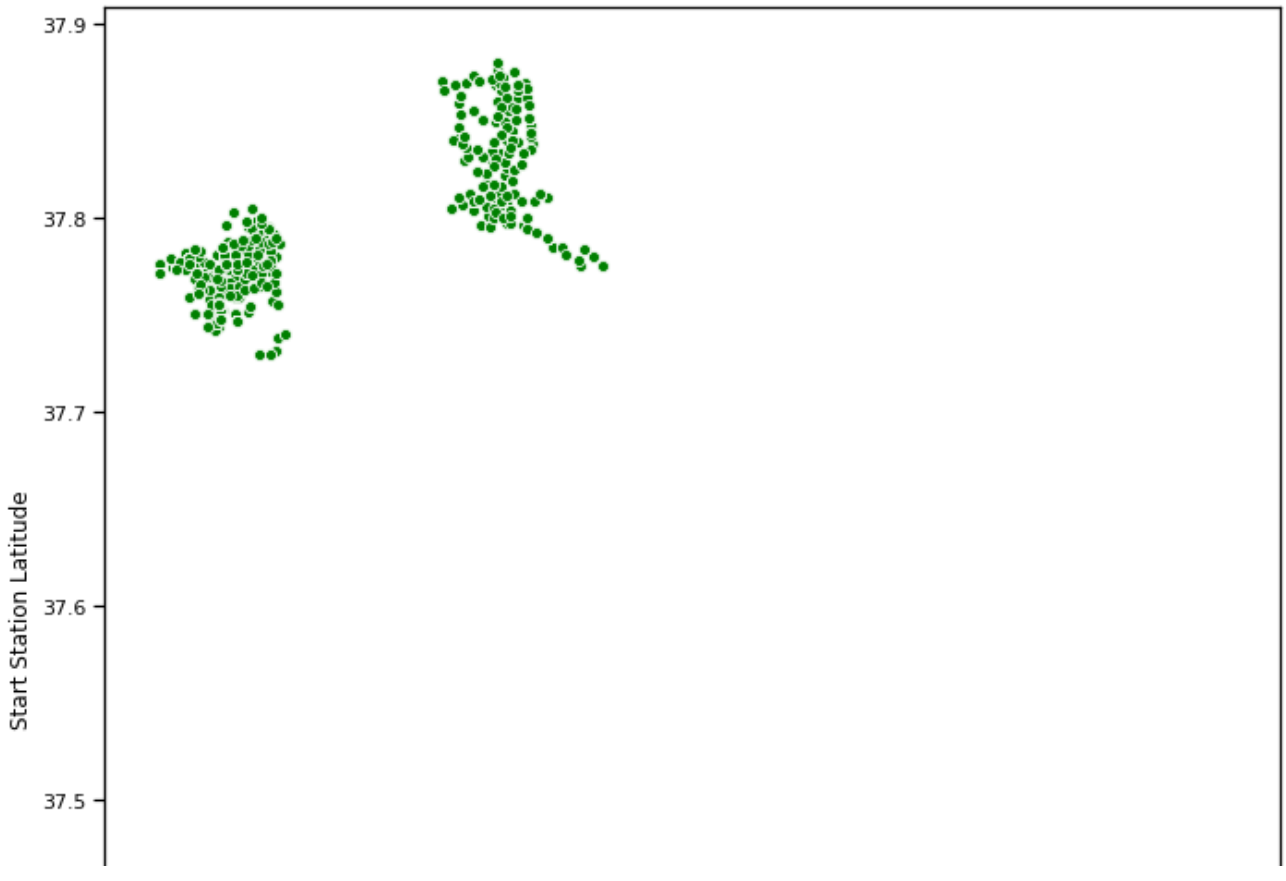
Finding: During weekdays, the rentals come from people between 30-39 the most, followed by people between 20-29 who rented 2000-4000 times fewer. Whereas in the weekend, 20-29 group takes over and makes the most rentals, they only lead 30-39 group by a few hundred.

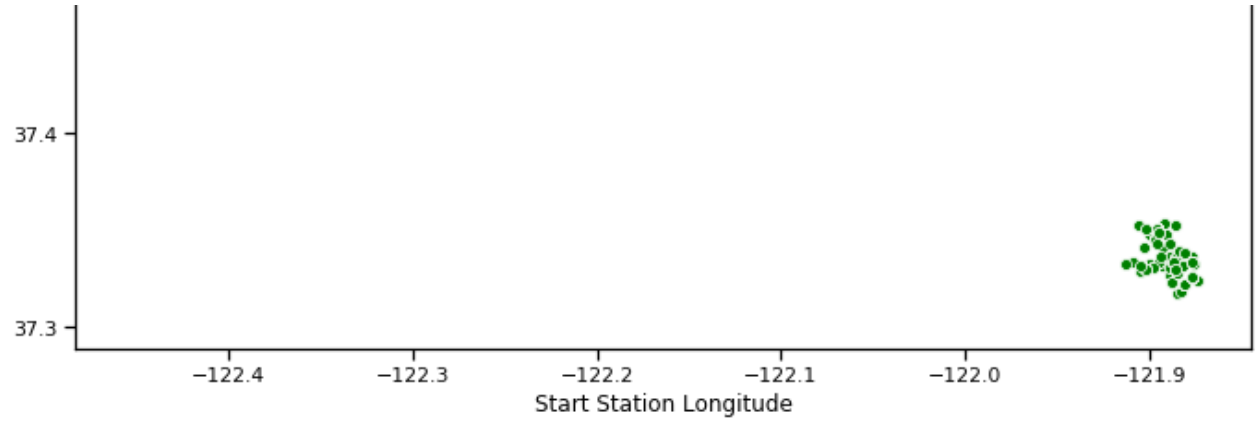
(3) Average trip distance distribution during each hour of day in different regions

First, let's review the locations of the start stations.

In [71]:

```
# plot all latitudes and longitudes of start stations
with sns.plotting_context("notebook"):
    # create a matplotlib figure
    fig, ax = plt.subplots(figsize = (10,10))
    sns.scatterplot(y = "start_station_latitude", x = "start_station_longitude", data =
df_bike_copy, color = 'g')
    plt.ylabel('Start Station Latitude')
    plt.xlabel('Start Station Longitude')
    plt.tight_layout()
```





I searched in Google map these locations and found they (from left to right) correspond to San Francisco, Oakland and San Jose. Now, I will add a column in the df_bike_copy data set that records the region infomation based on start stations.

In [72]:

```
# function that returns region based on longitudes
def region(x):
    if x < -122.35:
        return 'San Francisco'
    elif x < -122.10:
        return 'Oakland'
    else:
        return 'San Jose'
```

In [73]:

```
# apply the dustance function and add distance column to each row
df_bike_copy['region'] = df_bike_copy['start_station_longitude'].apply(region)
```

In [74]:

```
# data head after adding region info
df_bike_copy.head()
```

Out[74]:

	duration_sec	start_time	end_time	start_station_id	start_station_name	start_station_latitude	start_station_longitude	e
0	52185	2019-02-28 17:32:10.145	2019-03-01 08:01:55.975	21.0	Montgomery St BART Station (Market St at 2nd St)	37.789625	-122.400811	
1	42521	2019-02-28 18:53:21.789	2019-03-01 06:42:03.056	23.0	The Embarcadero at Steuart St	37.791464	-122.391034	
2	61854	2019-02-28 12:13:13.218	2019-03-01 05:24:08.146	86.0	Market St at Dolores St	37.769305	-122.426826	
3	36490	2019-02-28 17:54:26.010	2019-03-01 04:02:36.842	375.0	Grove St at Masonic Ave	37.774836	-122.446546	
4	1585	2019-02-28 23:54:18.549	2019-03-01 00:20:44.074	7.0	Frank H Ogawa Plaza	37.804562	-122.271738	

5 rows × 23 columns

In [75]:

```
# observation counts in different regions
df_bike_copy['region'].value_counts()
```

Out[75]:

```
San Francisco    133708
```

```
Oakland            41434
San Jose            8073
Name: region, dtype: int64
```

We have 133708 rentals from San Francisco area, 41434 rentals from Oakland and 8073 rentals from San Jose. I will replot the locations of start stations of all the rentals, using different colors to represent different regions.

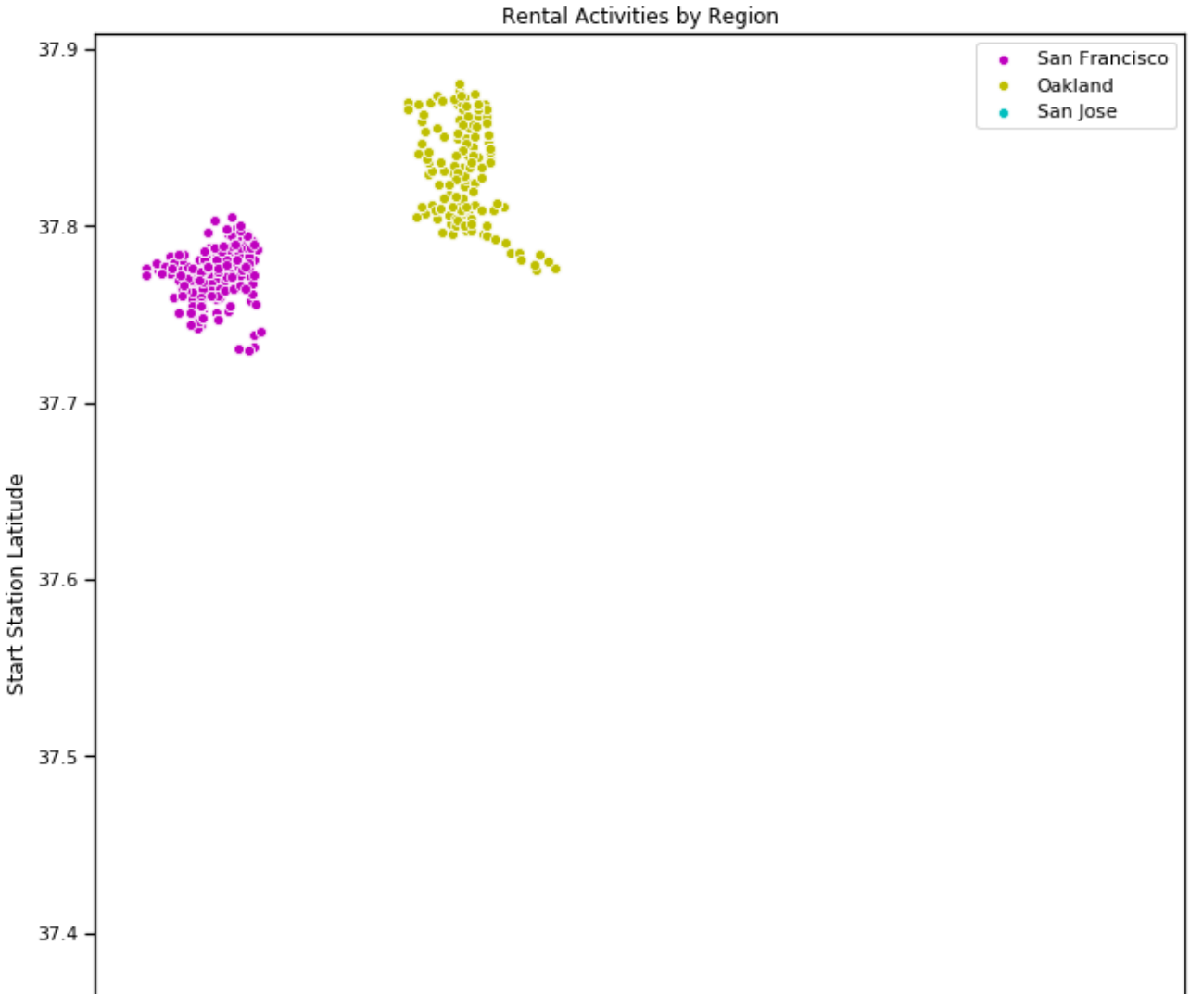
I would like to overwrite the saved clean file since now we have an extra region column.

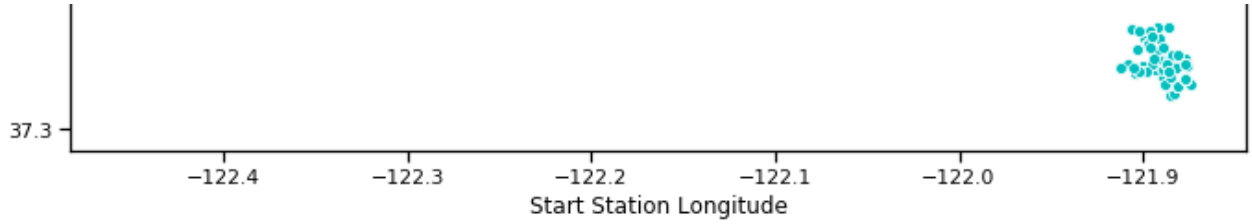
In [76]:

```
# save the cleaned data
df_bike_copy.to_csv("GoBike_Cleaned.csv", index = False)
```

In [77]:

```
# visualize different regions
with sns.plotting_context("notebook"):
    # create a matplotlib figure
    fig, ax = plt.subplots(figsize = (10,10))
    sns.scatterplot(y = "start_station_latitude", x = "start_station_longitude", data =
df_bike_copy.query("region == 'San Francisco'"),
                    label = 'San Francisco', color = 'm',)
    sns.scatterplot(y = "start_station_latitude", x = "start_station_longitude", data =
df_bike_copy.query("region == 'Oakland'"),
                    label = 'Oakland', color = 'y')
    sns.scatterplot(y = "start_station_latitude", x = "start_station_longitude", data =
df_bike_copy.query("region == 'San Jose'"),
                    label = 'San Jose', color = 'c')
    plt.title('Rental Activities by Region')
    plt.ylabel('Start Station Latitude')
    plt.xlabel('Start Station Longitude')
    plt.legend()
    plt.tight_layout()
```





In [78]:

```
# plot trip distance on each hour of day in different regions
# cut off at .99 percentile, 5070 meter
fig, ax = plt.subplots(figsize = (20,10))
sns.barplot(x = 'hour', y = 'distance', data = df_bike_copy.query('distance < 5070').gro
upby(['hour', 'region'], as_index = False).mean(), hue = 'region')
plt.title('Trip Distance on each Hour of Day in Different Regions')
```

Out[78]:

```
Text(0.5, 1.0, 'Trip Distance on each Hour of Day in Different Regions')
```

Finding: Compared between the regions of San Francisco, Oakland and San Jose, we can see that on average the ride distance over hour in each region follows similar rush-hour pattern as peaks are observed at ~8 am and ~5 pm. San Francisco riders took the longest trip, averaged over 1.5 km, Oakland, San Jose are the second (~1.4 km) and the third (~1.2 km) (we expect this because SF is a larger city with more populations).

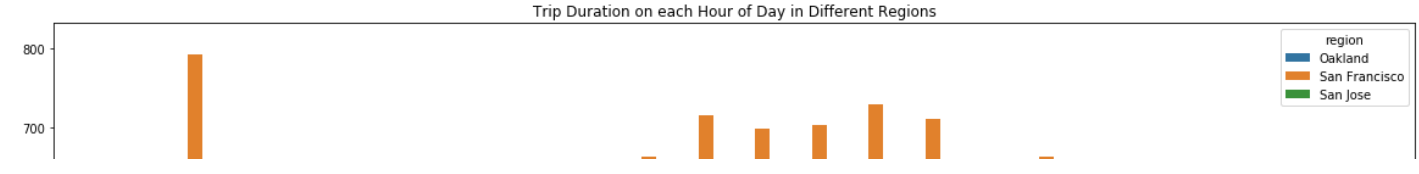
(4) Average trip duration during each hour of day in different regions

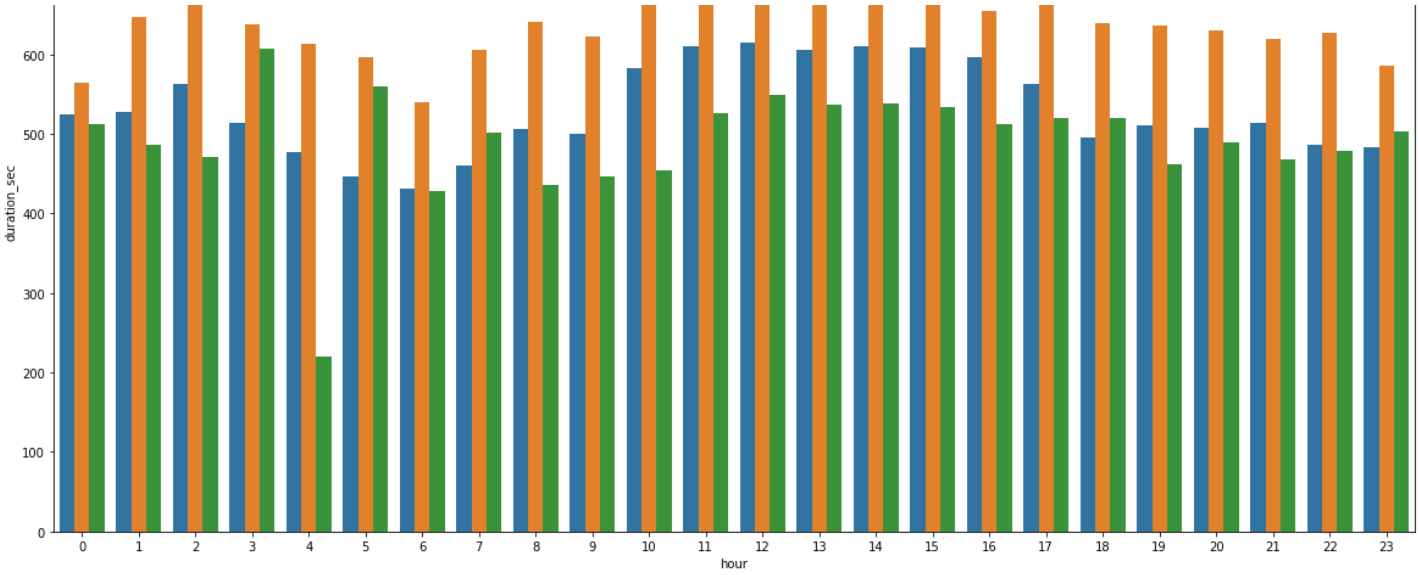
In [79]:

```
# plot trip duration on each hour of day in different regions
# cut off at .99 percentile, 3457 sec
fig, ax = plt.subplots(figsize = (20,10))
sns.barplot(x = 'hour', y = 'duration_sec', data = df_bike_copy.query('duration_sec < 34
57').groupby(['hour', 'region'], as_index = False).mean(), hue = 'region')
plt.title('Trip Duration on each Hour of Day in Different Regions')
```

Out[79]:

```
Text(0.5, 1.0, 'Trip Duration on each Hour of Day in Different Regions')
```





Finding: Compared between the regions of San Francisco, Oakland and San Jose, on average the ride duration over hour in each region follows the similar pattern and in contrast to the ride distance, peak is located at noon. San Francisco riders took the longest trip (over 10 min), Oakland, San Jose are the second (~9.2 min) and the third (~8.3 min) (again we expect this because SF is a larger city with more populations).

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?

From the multivariate exploration, I have the follow findings:

- Ride durations from other genders are less than ride durations from females and males on average throughout different ages. For data points that have ride duration values larger than 4000s (67 min, likely to be outliers), most of them come from male users (blue), which indicates males might be "less careful" than females to return the bikes.
- During weekdays, the rentals come from people between 30-39 the most, followed by people between 20-29 who rented 2000-4000 times fewer. Whereas in the weekend, 20-29 group takes over and makes the most rentals, they only lead 30-39 group by a few hundred.
- We have 133708 rentals from San Francisco area, 41434 rentals from Oakland and 8073 rentals from San Jose.
- Compared between the regions of San Francisco, Oakland and San Jose, we can see that on average the ride distance over hour in each region follows similar rush-hour pattern as peaks are observed at ~8 am and ~5 pm. San Francisco riders took the longest trip, averaged over 1.5 km, Oakland, San Jose are the second (~1.4 km) and the third (~1.2 km) (we expect this because SF is a larger city with more populations).
- Compared between the regions of San Francisco, Oakland and San Jose, on average the ride duration over hour in each region follows the similar pattern and in contrast to the ride distance, peak is located at noon. San Francisco riders took the longest trip (over 10 min), Oakland, San Jose are the second (~9.2 min) and the third (~8.3 min) (again we expect this because SF is a larger city with more populations).

From the point that 30-39 age group used far more rentals during weekday, it strengthens the point that **the use of GoBike during weekdays is majorly for work commuting**.

Were there any interesting or surprising interactions between features?

This is the interesting thing I found:

- On average the ride distance over hour in each region follows similar rush-hour pattern as peaks are observed at ~8 am and ~5 pm. however, the ride duration over hour in each region follows a different pattern, in contrast to the ride distance, peak is located at noon. The explanation might be at noon people rent the bike to get lunch nearby, therefore, it takes longer time but shorter distance, whereas during rush hours, people rides for a longer distance (from company to metro station for example), and it takes less time.

Conclusion

In this project, I explored the Ford GoBike system data. We have made 7 univariate explorations, 6 bivariate explorations and 4 multivariate explorations. At the beginning I gave three questions (actually we explored questions far more than these), I will provide some insights for them here:

- **When are most trips taken in terms of time of day or day of the week? (In terms of time of day, most trips are taken at 8 am and 5 pm, in terms of day of week, most trips are taken on weekdays.)**
- **How long does the average trip take? (On average the trip takes 621 seconds (10.35 min))**
- **Does the above depend on if a user is a subscriber or customer? (For subscriber it takes on average 589 seconds, whereas it takes 904 seconds for customer.)**

In [ ]: