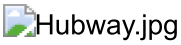
Hubway: Bike Sharing Service

In this project, we will conduct data analysis on a dataset from the bike-sharing service Hubway, which includes data on over 1.5 million trips made with the service.This project was first done by James Coe on Dataquest blogs. Here the project is done again for practice.



Analysis on 'Trips' Table

Let's start by taking a look at our database. The database has two tables, **trips** and **stations**. To begin with, we'll just look at the **trips** table. It contains the following columns:

- id — A unique integer that serves as a reference for each trip
- duration — The duration of the trip, measured in seconds
- start_date — The date and time the trip began
- start_station — An integer that corresponds to the id column in the stations table for the station the trip started at
- end_date — The date and time the trip ended
- end_station — The 'id' of the station the trip ended at
- bike_number — Hubway's unique identifier for the bike used on the trip
- sub_type — The subscription type of the user. "Registered" for users with a membership, "Casual" for users without a membership
- zip_code — The zip code of the user (only available for registered members)
- birth_date — The birth year of the user (only available for registered members)
- gender — The gender of the user (only available for registered members)

We will now perform SQL analysis on the *trips* table to answer the following questions-

- What was the duration of the longest trip?
- How many trips were taken by 'registered' users?
- What was the average trip duration?
- Do registered or casual users take longer trips?
- Which bike was used for the most trips?
- What is the average duration of trips by users over the age of 30?

In [1]:
```
import sqlite3
import pandas as pd
db = sqlite3.connect('hubway.db')
def run_query(query):
    return pd.read_sql_query(query,db)
```

Let's see the first 5 rows of trips table to get some idea about the data inserted in this table.

In [2]:
```
query='''
SELECT *
FROM trips
LIMIT 5;
'''
run_query(query)
```

Out[2]:

	id	duration	start_date	start_station	end_date	end_station	bike_number	sub_type	zip_co
0	1	9	2011-07-28 10:12:00	23	2011-07-28 10:12:00	23	B00468	Registered	'972
1	2	220	2011-07-28 10:21:00	23	2011-07-28 10:25:00	23	B00554	Registered	'022
2	3	56	2011-07-28 10:33:00	23	2011-07-28 10:34:00	23	B00456	Registered	'021
3	4	64	2011-07-28 10:35:00	23	2011-07-28 10:36:00	23	B00554	Registered	'021
4	5	12	2011-07-28 10:37:00	23	2011-07-28 10:37:00	23	B00554	Registered	'972

In [3]:
```
query='''
SELECT duration AS 'Duration of trips'
FROM trips
ORDER BY  duration  DESC
LIMIT 5;
'''
run_query(query)
```

Out[3]:

	Duration of trips
0	9999
1	9998
2	9998
3	9997
4	9996

The duration of the longest trip is 9999 sec. But this number actually doesn't represent duration. It is representing the biggest possible 4 digit number. Hubway charges additional fees for rides over 30 minutes (somebody keeping a bike for 9999 seconds would have to pay an extra 25 in fees) so it's plausible that they decided 4 digits would be sufficient to track the majority of rides.

In [4]:
```
query='''
SELECT count(*) AS 'Total Trips by Registered Users'
FROM trips
WHERE sub_type='Registered'
'''
run_query(query)
```

Out[4]:

	Total Trips by Registered Users
0	1105192

So, there are about 1105192 registered users.

In [5]:
```
query='''
SELECT AVG(duration) AS 'Average Trip Duration'
FROM trips ;
'''
run_query(query)
```

Out[5]:

	Average Trip Duration
0	912.409682

It turns out that the average trip duration is 912 seconds, which is about 15 minutes. This makes some sense, since we know that Hubway charges extra fees for trips over 30 minutes. The service is designed for riders to take short, one-way trips.

In [6]:
```
query='''
SELECT sub_type 'Subscription Type', AVG(duration)  'Average Trip Duration'
FROM trips
GROUP BY sub_type ;
'''
run_query(query)
```

Out[6]:

	Subscription Type	Average Trip Duration
0	Casual	1519.643897
1	Registered	657.026067

That's quite a difference! On average, registered users take trips that last around 11 minutes whereas casual users are spending almost 25 minutes per ride. Registered users are likely taking shorter, more frequent trips, possibly as part of their commute to work. Casual users, on the other hand, are spending around twice as long per trip.

It's possible that casual users tend to come from demographics (tourists, for example) that are more inclined to take longer trips to get around and see all the sights. Once we've discovered this difference in the data, there are many ways the company might be able to investigate it to better understand what's causing it.

In [7]:
```
query='''
SELECT count(DISTINCT bike_number)  'Bike Number'
FROM trips;
'''
run_query(query)
```

Out[7]:

	Bike Number
0	1164

In [8]:
```
query='''
SELECT bike_number  'Bike Number', count(*)  'Number of Trips'
FROM trips
GROUP BY bike_number
ORDER BY count(*) DESC
LIMIT 1;
'''
run_query(query)
```

Out[8]:

	Bike Number	Number of Trips
0	B00490	2120

*As we can see from the above analysis, there are total 1164 bikes. The Bike **B00490** has the maximum number of trips*

Analysis on 'Stations' Table

Now, let's look at the other table **Stations** table. It contains the following columns:

- id — A unique identifier for each station (corresponds to the start_station and end_station columns in the trips table)
- station — The station name
- municipality — The municipality that the station is in (Boston, Brookline, Cambridge or Somerville)
- lat — The latitude of the station
- lng — The longitude of the station

We will now perform SQL analysis on the stations table to answer the following questions-

- which station is the most frequent starting point?
- Which stations are most frequently used for round trips?
- How many trips start and end in different municipalities?

Let's see the first 5 rows of stations table to get some idea about the data inserted in this table.

In [9]:
```
query='''
SELECT *
FROM stations
LIMIT 5;
'''
run_query(query)
```

Out[9]:

	id	station	municipality	lat	lng
0	3	Colleges of the Fenway	Boston	42.340021	-71.100812
1	4	Tremont St. at Berkeley St.	Boston	42.345392	-71.069616
2	5	Northeastern U / North Parking Lot	Boston	42.341814	-71.090179
3	6	Cambridge St. at Joy St.	Boston	42.361284999999995	-71.06514
4	7	Fan Pier	Boston	42.353412	-71.044624

In [10]:
```
query='''
SELECT t.start_station 'Start Station Id', s.station 'Station name', count(*)
 'Number of Trips'
FROM trips t
JOIN stations s
ON t.start_station = s.id
GROUP BY s.station
ORDER BY count(*) DESC
LIMIT 5;
'''
run_query(query)
```

Out[10]:

	Start Station Id	Station name	Number of Trips
0	22	South Station - 700 Atlantic Ave.	56123
1	36	Boston Public Library - 700 Boylston St.	41994
2	60	Charles Circle - Charles St. at Cambridge St.	35984
3	53	Beacon St / Mass Ave	35275
4	67	MIT at Mass Ave / Amherst St	33644

South Station, the most frequent starting point, is one of the main commuter rail stations in the city, Charles Street runs along the river close to some nice scenic routes, and Boylston and Beacon streets are right downtown near a number of office buildings.

In [11]:
```
query='''
SELECT count(*)
FROM trips
WHERE start_station=end_station ;
'''
run_query(query)
```

Out[11]:

	count(*)
0	73068

In [12]:
```
query='''
SELECT t.start_station 'Start Station Id', s.station 'Station name', count(*)
 'Number of Trips'
FROM trips t
JOIN stations s
ON t.start_station = s.id
WHERE t.start_station=t.end_station
GROUP BY s.station;
'''
run_query(query)
```

Out[12]:

	Start Station Id	Station name	Number of Trips
0	135	359 Broadway - Broadway at Fayette Street	88
1	9	Agganis Arena - 925 Comm Ave.	771
2	66	Allston Green District - Commonwealth Ave & Gr...	344
3	85	Andrew Station - Dorchester Ave at Humboldt Pl	99
4	20	Aquarium Station - 200 Atlantic Ave.	1280
...	...	...	...
132	39	Washington St. at Rutland St.	727
133	26	Washington St. at Waltham St.	555
134	106	West Broadway at Dorchester St	34
135	138	Wilson Square	101
136	45	Yawkey Way at Boylston St.	771

137 rows × 3 columns

In [13]:
```
query='''
SELECT t.start_station 'Start Station Id', s.station 'Station name', count(*)
 'Number of Trips'
FROM trips t
JOIN stations s
ON t.start_station = s.id
WHERE t.start_station=t.end_station
GROUP BY s.station
ORDER BY count(*) DESC
LIMIT 5;
'''
run_query(query)
```

Out[13]:

	Start Station Id	Station name	Number of Trips
0	58	The Esplanade - Beacon St. at Arlington St.	3064
1	60	Charles Circle - Charles St. at Cambridge St.	2739
2	36	Boston Public Library - 700 Boylston St.	2548
3	42	Boylston St. at Arlington St.	2163
4	53	Beacon St / Mass Ave	2144

*From the above analysis, we can conclude that, there are total 73068 round trips and these trips take place through 137 stations. Among these stations, the station **The Esplanade - Beacon St. at Arlington St** is most freuenty used or round trips and around 3064 trips occur at this station.*

As we can see, a number of these stations are the same as the previous question but the amounts are much lower. The busiest stations are still the busiest stations, but the lower numbers overall suggest that people are typically using Hubway bikes to get from point A to point B rather than cycling around for a while before returning to where they started. There is one significant difference here — the Esplande, which was not one of the overall busiest stations from our first query, appears to be the busiest for round trips. Why? Well, a picture is worth a thousand words. This certainly looks like a nice spot for a bike ride:


Esplanade.jpg

In [14]:
```
query='''
SELECT  s.municipality 'Municipality', count(*) 'Number of Stations'
FROM stations s
GROUP BY s.municipality;
'''
run_query(query)
```

Out[14]:

	Municipality	Number of Stations
0	Boston	97
1	Brookline	5
2	Cambridge	28
3	Somerville	12

There are 142 stations distributed in 4 different municipalities. Lets see how may trips start and end in different municipalities. To achieve this, we need to JOIN the trips table to the stations table twice. Once ON the start_station column and then ON the end_station column.

In [15]:
```
query='''
SELECT s_start.municipality 'Start Station Municipality', s_end.municipality
 'End Station Municipality'
FROM trips t
JOIN stations s_start
ON t.start_station = s_start.id
JOIN stations s_end
ON t.end_station = s_end.id
WHERE s_start.municipality!=s_end.municipality
'''
run_query(query)
```

Out[15]:

	Start Station Municipality	End Station Municipality
0	Boston	Cambridge
1	Boston	Cambridge
2	Boston	Cambridge
3	Boston	Cambridge
4	Cambridge	Boston
...	...	...
309743	Cambridge	Somerville
309744	Cambridge	Somerville
309745	Cambridge	Somerville
309746	Cambridge	Boston
309747	Cambridge	Somerville

309748 rows × 2 columns

This shows that about 300,000 out of 1.5 million trips (or 20%) ended in a different municipality than they started — further evidence that people mostly use Hubway bicycles for relatively short journeys rather than longer trips between towns.

Next Steps

Here are a few more questions for further analysis-

- How many trips incurred additional fees (lasted longer than 30 minutes)?
- Which bike was used for the longest total time?
- Did registered or casual users take more round trips?
- Which municipality had the longest average duration?