Assignment 4. Creating a Transit Bus App Utilizing Streaming Open Data

Due March 25 11:59 PM Worth 25 points

In your teams, build a bus finder "app" for a city of your choice. A user will click on a point, indicating his/her location. Your bus finder will inform the user whether or not he/she has sufficient time to get to the bus stop.

GOALS

After you've completed this assignment, you should better understand how to use:

- Use JS to parse tagged XML data
- Use SQL (structured query language) with a cloud database--this year it's Google Fusion tables
- Create "Classes," import libraries and work with the cloud
- Gain a working knowledge of how to work with open data and open data standards
- Finally, gain experience in working with streaming data.

Resources

1. Transit feeds of all kinds are aggregated at http://transitfeeds.com and discussed in https://www.nextbus.com/xmlFeedDocs/NextBusXMLFeed.pdf. You're looking for real time feeds of buses. A close-by source can be found in the City of Laval or Ottawa.

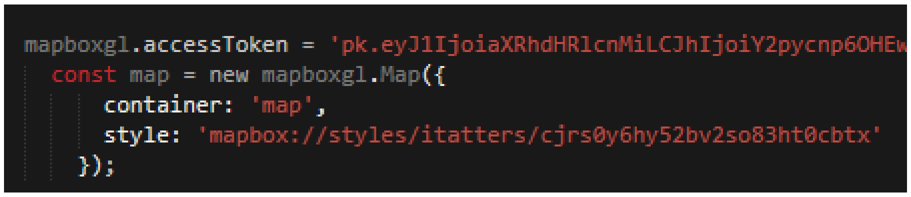
2. Bus data is streamed in the most common transit standard, the General Transit Feed Specification (GTFS) standard. More information on GTFS can be found at https://gtfs.org/. [If you want to use train/subway/ferry data, contact us.]

3. If you haven't already used them, learn about Google Fusion tables: https://support.google.com/fusiontables/answer/184641.

4. You will need to query data and also interact with the cloud. Learn more about Structured Query Language (SQL) as a method to extract information from a database: https://developers.google.com/fusiontables/docs/v2/sql-reference. Similar to previous assignments, you may wish to use jquery, which combines callbacks and queries.

5. You need to use the Mapbox API for this one.

6. OPTIONAL: Feel free to customize the look and feel of your map using Mapbox Studio. The style must be public and added to your new map object like shown below.



```
mapboxgl.accessToken = 'pk.eyJ1IjoiaXRhdHRlcnMiLCJhIjoiY2pycnp6OHEw
  const map = new mapboxgl.Map({
      container: 'map',
      style: 'mapbox://styles/itatters/cjrs0y6hy52bv2so83ht0cbtx'
    });
```

Tasks

1. Build your own bus "app" (i.e., another website)
    a. You will need to map the locations of each bus stop. You will get these from one of the GTFS datasets. Depending on the data (e.g., bus stop locations), you may need to geocode data so features like bus stops appear in your app. Use Mapbox for the actual mapping.
    b. Allow the user to click on the map to identify his/her current location. You will need to obtain the coordinates of the on click.
    c. Query the fusion table using JS to get the bus stop data. You may find that you can refine the query to get a subset of the data.
    d. Then calculate the distance from the clicked on point to all the bus stops to find the nearest bus stops (see below).
    e. From the live bus feed data find the distance of all buses to the chosen bus stop.
    f. Based on the nearest (temporally) bus, inform the user at what time the next bus arrives and whether he/she has sufficient time to get to the stop based on the location of the user click.
    g. Highlight the eventually selected bus stop.

2. Write a function that calculates the distance between bus stops and the user-identified point.
    a. Choose whichever equation or algorithm best fits your app.
    b. Your function also needs to translate distance into walking time (in minutes). The suggested speed is from 0.066km/min to 0.1km/min.

3. You will need to store your city's bus data in a database in the cloud. We are using Google Fusion Tables for the cloud db. You may need to create several tables because bus data contains static and dynamic components. Recall from GIS that tables should be well defined and give the reader a sense of the primary keys and the tuple structure. Provide screenshot(s) of your Fusion tables.

4. Your app should be presented in a user-friendly interface. A single page is best (no navigating away) and you should be able to do the query and display without

doing a refresh. Your app should explain its purpose and provide any necessary information to the user. You will not be graded on the aesthetics of the app.

5. You can do development on your local host but your final app must be hosted in the cloud and function server side, which for us means running from your group directory of our site, https://neogeoweb.ca.

6. As usual, document your code. Also structure your code. When you are working with features like bus stops, these should be defined as classes. As much as possible you should structure your code with classes and with functions.

7. Your report should do the following
   a. It should describe the purpose of your app.
   b. It should justify the city you chose. Compare your city's transit feed to the specification for GTFS. Discuss the strengths and weaknesses of that city's feeds.
   c. It should document the structure of your app. Build a flowchart (think of it as a cartographic model) to explain the step-by-step logic of your app. Make sure it highlights the functions you have created and decisions you have made about table structure(s).

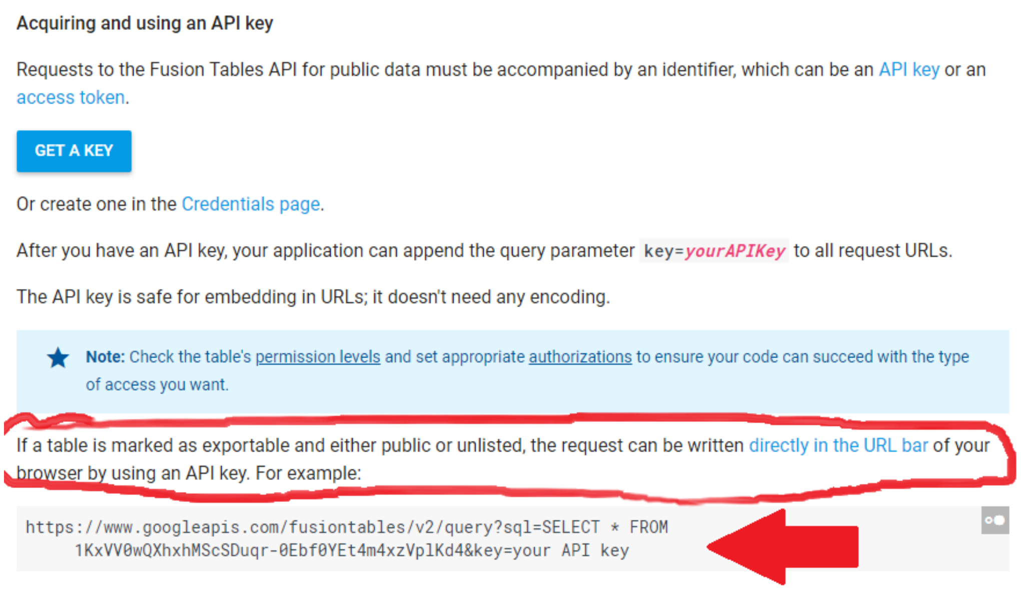
8. For each of you in the team, reflect on the assignment (approximately one page each)
   a. What was the hardest part of this assignment? What was the easiest?
   b. What information sources did you use to complete your portion of the assignment? For example, did you view any Youtube videos? Which ones? Did you utilize stackoverflow? How?
   c. How did you structure your Fusion Tables data (at the level of data like keys, data records, and datasets)? Comment on the static and dynamic natures of the datasets and how this effected your structure and the code you wrote. If you had more time, would you have structured them differently? How? Is there any missing data?
   d. Make sure that all the individual reflections for your team are appended to the bottom of your report.

Submission

1) Email the code, the URL to your working app, and your report describing the process (including individual reflections) to renee.sieber@mcgill.ca and sam.lumley@mail.mcgill.ca.
2) Share your finished fusion table(s) with resieber@gmail.com and samfredlumley@gmail.com.

IMPORTANT: Procedural Advice

DO NOT use OAuth to authenticate your request. Use the following method highlighted below. This is described on https://developers.google.com/fusiontables/docs/v2/using



Acquiring and using an API key

Requests to the Fusion Tables API for public data must be accompanied by an identifier, which can be an API key or an access token.

GET A KEY

Or create one in the Credentials page.

After you have an API key, your application can append the query parameter `key=yourAPIKey` to all request URLs.

The API key is safe for embedding in URLs; it doesn't need any encoding.

★ **Note:** Check the table's permission levels and set appropriate authorizations to ensure your code can succeed with the type of access you want.

If a table is marked as exportable and either public or unlisted, the request can be written directly in the URL bar of your browser by using an API key. For example:

```
https://www.googleapis.com/fusiontables/v2/query?sql=SELECT * FROM
    1KxVV0wQXhxhMScSDuqr-0Ebf0YEt4m4xzVplKd4&key=your API key
```