Data Storage and Retrieval
Fall 2016

Assignment 3
September 7, 2016

Show all your work!

1. Create a database called hubway_<lastname> from the data in compsci01:/usr/share/databases/Hubway. Create two appropriately named tables and load the Hubway csv files into the tables. Include any reasonable integrity constraints in your "create table" commands. Please at least include a foreign key constraint. Read the README file!

2. Write these queries.

    a. Find the first 10 the station names whose status correspond to 'Removed', sorted by station, ascending.

    b. Find the first 10 the station names that are located inside the bounding box formed by two given (latitude, longitude) points, sorted by station, ascending.

    c. Find the first 10 trips' ids (hubway_id) that started or ended at stations within a bounding boxed formed by two given (latitude, longitude) points, sorted by id, ascending.

3. Write another query to find records in the hubway tables with suspect values of zip code or duration. Start by snooping around the database manually to see how data can be bad. Find any other bad data you can.

    Your query will create a table whose first column is called `problem`, a short description of the problem, and whose other columns are enough to identify the bad record and illustrate the problem with the data. Show five examples of each problem.

4. Create a database called fanfiction_<lastname>. Create a stories_orig table with url as the primary key.
    a. Use \copy to load Fanfiction/stories_orig.csv into the table. Despite what Postgres may say, the problem isn't that url is the primary key; the problem is in the data. Explain.
    b. Work around the data problem temporarily by adding a sequence column to the table, removing url as a primary key, and loading the table. (Consult the Postgres docs and/or StackOverflow.)
    c. Fix the data problem using a select statement that joins stories_orig with itself (!), checking for records with the same url. Try your sql with `explain` with and without an index, but run it with an index.

5. Write two python programs to fix the data problem. These are two strategies
    a. One program sorts the rows of stories_orig.csv and traverse rows in sorted order.
    b. The other program uses a *hash*:
        i. Find a way to map each row *r* to a number *h(r)* between 0 and N = 2^{20}-1. One suggestion (among many): represent `published` as mmddyy and `words` as nnn...n, and set *h(r)* to mmddyynnn...n mod N.
        ii. Create a python list of N empty lists.
        iii. Add each row *r* to the *h(r)*th list.
        iv. Traverse the list of lists. If any list contains more than one row, compare them.
    c. Which program runs faster? Explain.