Data Storage and Retrieval

Fall 2018

Assignment 3

Prateek Shaw and Diana Sujanto

1. Create a postgres 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!

We created hubway\_shaw\_sujanto database on compsci01. Then, start creating our tables : Stations and Trips

#Table 1

CREATE TABLE Stations (

id int primary key,

terminal character varying(6),

station character varying(100),

municipal character varying(30),

lat real,

lng real,

status character varying(20)

);

Import hubway\_stations.csv to Stations table:

\COPY Stations( id, terminal, station, municipal,lat, lng, status) FROM '/usr/share/databases/Hubway/hubway\_stations.csv' delimiter ',' csv header;

#Table 2

Trips table

CREATE TABLE Trips (

seq\_id serial primary key,

hubway\_id bigint,

status character varying(10),

duration integer,

start\_date timestamp without time zone,

strt\_statn integer,

end\_date timestamp without time zone,

end\_statn integer,

bike\_nr character varying(20),

subsc\_type character varying(20),

zip\_code character varying(6),

birth\_date integer,

gender character varying(10)

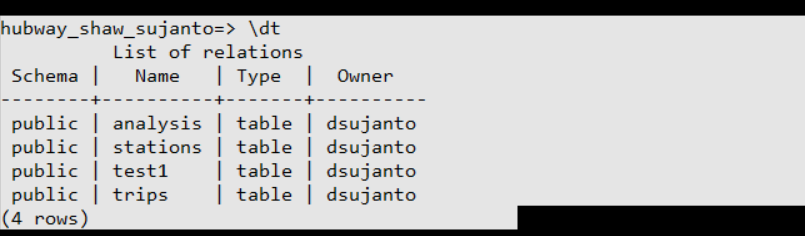
);

Adding constraints to the Trips table:

ALTER TABLE Trips ADD CONSTRAINT constraint\_fk FOREIGN KEY (strt\_statn) REFERENCES Stations(id) ON DELETE CASCADE;

ALTER TABLE Trips ADD CONSTRAINT constraint1\_fk FOREIGN KEY (end\_statn) REFERENCES Stations(id) ON DELETE CASCADE;

Check to make sure that the tables exists: \dt



Because there is a sequence datatype in Trips table, we cannot just import the hubway\_trips.csv into the table. Our solution is to create a temp table and import the file to the temp table. Then, we will insert all records into Trips table.

This is our temp table:

CREATE TABLE test1 (seq\_id integer,

hubway\_id bigint,

status character varying(10),

duration integer,

start\_date timestamp without time zone,

strt\_statn integer,

end\_date timestamp without time zone,

end\_statn integer,

bike\_nr character varying(20),

subsc\_type character varying(20),

zip\_code character varying(6),

birth\_date integer,

gender character varying(10));

Import the .csv into test1:

\COPY test1( seq\_id, hubway\_id, status, duration,start\_date,strt\_statn,end\_date,end\_statn,bike\_nr,subsc\_type,zip\_code,birth\_date,gender) FROM '/usr/share/databases/Hubway/hubway\_trips.csv' delimiter ',' csv header;

Insert into Trips table:

INSERT INTO Trips (hubway\_id,

status,

duration,

start\_date,

strt\_statn,

end\_date,

end\_statn,

bike\_nr,

subsc\_type,

zip\_code,

birth\_date,

gender) select hubway\_id,

status,

duration,

start\_date,

strt\_statn,

end\_date,

end\_statn,

bike\_nr,

subsc\_type,

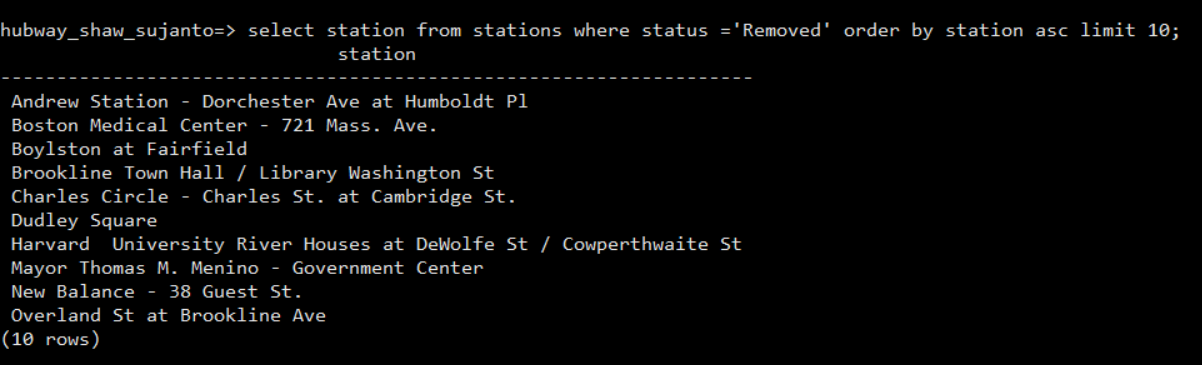
zip\_code,

birth\_date,

gender FROM test1;

1. Write these queries.
   1. Find the first 10 the station names whose status correspond to 'Removed', sorted by station, ascending.

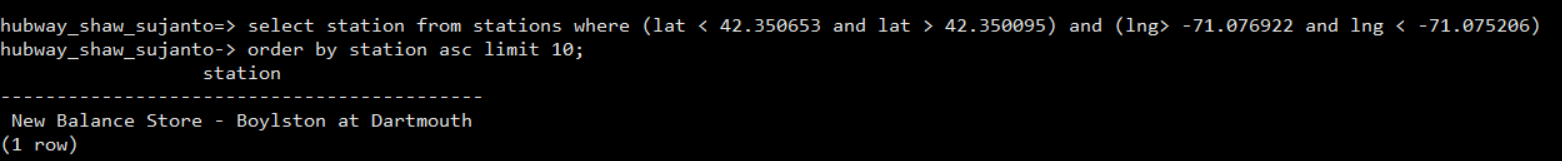
SELECT station from Stations WHERE status ='Removed' ORDER BY station asc limit 10;



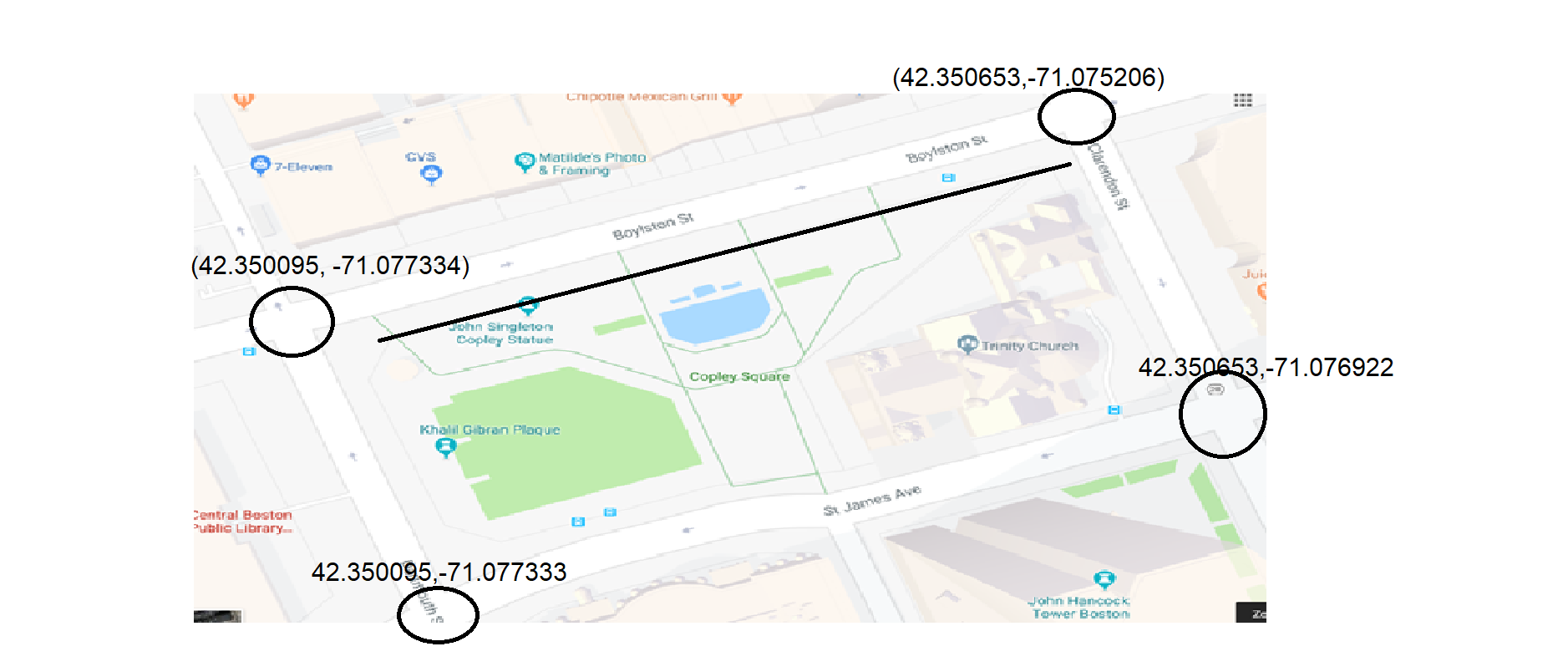
* 1. Find the first 10 the station names that are located inside the bounding box formed by two given (latitude, longitude) points, sorted by station, ascending.

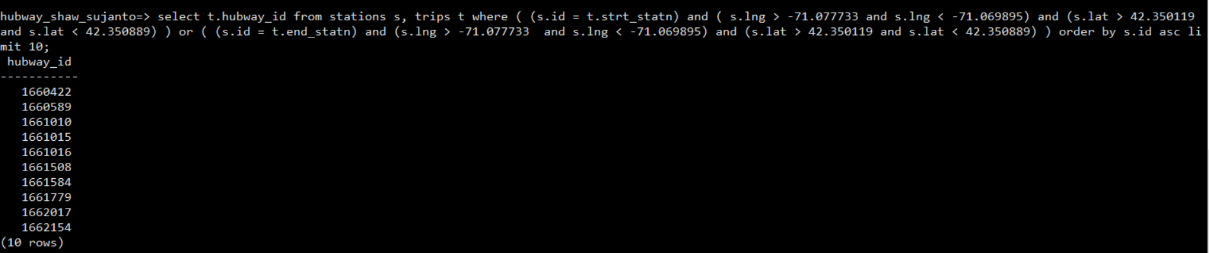
SELECT station FROM stations WHERE (lat < 42.350653 AND lat > 42.350095) AND (lng> -71.076922 AND lng < -71.075206)

ORDER BY station asc LIMIT 10;



* 1. 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.





1. 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.

SELECT zip\_code, duration FROM Trips LIMIT 10;

We found zip codes with single quotes in front of them.

Your query will create a table whose first column is called problem, a short text 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.

CREATE TABLE Analysis (

problem character varying(30),

description character varying(255)

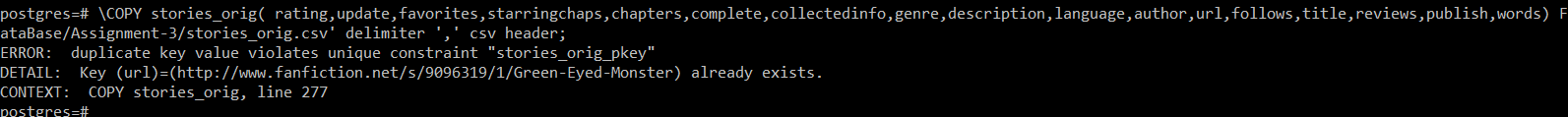
);

INSERT INTO Analysis(problem, description) VALUES ('Bad Zip\_codes','Zip\_codes in Trips table have single quotes in front of it:''02140 ''02143 ''02139 ''02215');

At first, we could not insert the record because bad zipcodes have single quotes. We had to put another quote in front of each zipcode.

1. Create a postgres database called fanfiction\_<lastname> from the data in compsci01:/usr/share/databases/Fanfiction. Create a stories\_orig table with url as the primary key.
   1. Use \copy to load Fanfiction/stories\_orig.csv into the table. Despite what Postgres may say, url really should be the primary key; the problem is in the data. Explain.

Our data have some duplicate URLs, which cause problem for URL to be the primary key. By definition, primary key should be unique. This is the reason of the following error:



To prove this, we created one temporary table where URL is not a primary key. Here, we did not have any troubles importing data since there was no primary key constraint.



1. select count(\*) from stories\_orig\_temp;

result is 613705 and if you open csv file you can see the same number 613706(one extra row because of row header).

1. Then we ran below query to know count of distinct rows.  
   SELECT COUNT (DISTINCT url) FROM stories\_orig\_temp

Result is **613653** , which mean there are total number (613706-613653=53) has duplicate records.

* 1. Fix the data problem using SQL. You may want to create temporary tables and indexes. Explain your solution step by step.

Solution: to solve this issue we use this approach:

1. First we have created temp table named stories\_orig\_temp where there is no primary key.

CREATE TEMP TABLE stories\_orig\_temp (

rating varchar(50),

update varchar(50),

favorites int,

starringchaps varchar(50),

chapters int,

complete varchar(10),

collectedinfo varchar(500),

genre varchar(500),

description varchar(500),

language varchar(100),

author varchar(250),

url varchar(500),

follows int,

title varchar(100),

reviews int,

publish varchar(100),

words int

);

1. Then we have created index on column url .  
   CREATE INDEX url ON stories\_orig\_temp
2. Then we have created correct table with primary key.

CREATE TABLE stories\_orig (

rating varchar(50),

update varchar(50),

favorites int,

starringchaps varchar(50),

chapters int,

complete varchar(10),

collectedinfo varchar(500),

genre varchar(500),

description varchar(500),

language varchar(100),

author varchar(250),

url varchar(500) **primary key**,

follows int,

title varchar(100),

reviews int,

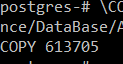
publish varchar(100),

words int

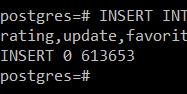
);

1. Also created index on url column
2. Then we have imported data from csv file to temp table.

\COPY stories\_orig\_temp( rating,update,favorites,starringchaps,chapters,complete,collectedinfo,genre,description,language,author,url,follows,title,reviews,publish,words) FROM '/mnt/c/Learning/DataScience/DataBase/Assignment-3/stories\_orig.csv' delimiter ',' csv header;



1. Then we have ran below sql query to populate unique URL rows from temp table to main table.  
   **INSERT** **INTO** stories\_orig (url,rating,update,favorites,starringchaps,chapters,complete,collectedinfo,genre,description,language,author,follows,title,reviews,publish,words) **SELECT** **DISTINCT** on (url) url,rating,update,favorites,starringchaps,chapters,complete,collectedinfo,genre,description,language,author,follows,title,reviews,publish,words **FROM** stories\_orig\_temp



The number of records being inserted into the table matched with the one in Question #4a.2, which is 613653 records. That means the duplicate records problem had been solved.

1. Fix the data problem in (4) using a python program. You may use the following strategy:
   * 1. Parse each line in the file using a csv parser.
     2. Add each url as a key to a python dictionary.

We have written below python code to address this.

#!/opt/rh/rh-python36/root/usr/bin/python  
import csv  
import sys  
  
inputfile = "/usr/share/databases/Fanfiction/stories\_orig.csv"  
  
  
  
  
# Function to read stories\_orig.csv and then populate dictionary  
# Dictionary is compose of two things one key and second is value.  
# From this file URL is key and row are value.  
# You can not have duplicate key. if it find same key then it will override value.  
def populateurldictionary():  
 with open(inputfile) as csvfile:  
 recordreader = csv.reader(csvfile, dialect='unix')  
 dicts = {}  
 next(recordreader)  
 i = 0  
 for line in recordreader:  
 # line[11] is the URL column  
 # i am only adding if key is not present  
 dicts[line[11]] = line  
 print(len(dicts.keys()))  
  
  
populateurldictionary()

If you run the above code it will print number of keys 613653 , this prove that our code is working as expected , because numbers rows are same.

1. Fix the data problem in (4) on the Linux command line using sort, uniq, and maybe cut.

sort stories\_orig.csv | uniq -c | wc -l