CSC 380/530 — Advanced Database
Project 2 (document version 1.1)
Data Clean-up Project using SQL and PL/SQL

- This project is due by 11:59:59 PM on Monday, October 26, 2015. Projects are to be submitted electronically.
- This project will count as 20% of your final course grade.
- This project is to be completed **individually**. Do not share your work with anyone else.

Overview

The focus of this assignment is on using SQL and Oracle PL/SQL to implement a data clean-up project. Datasets generated by various applications often contain incorrect or missing data. Your goal is to work with a relatively large dataset and process the data to form a "clean" and reorganized dataset.

Burger King Locations Dataset

Download the `bkloc.sql` file from the course website, then execute it within your Oracle environment. This will create a table called `burger_king_locations` and insert `6873` rows, each row corresponding to a single Burger King location.

If you do this from the SQL*Plus command interface, your output should look something like this:

```
SQL> @"/users/goldschmidt/documents/bkloc"
SQL> desc burger_king_locations
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------


 LONGITUDE                                          NUMBER
 LATITUDE                                           NUMBER
 NAME                                               VARCHAR2(200)
 ADDRESS                                            VARCHAR2(200)

SQL> select count(*) from burger_king_locations;

  COUNT(*)
----------
      6873

SQL>
```

The data in this table has been imported from a dataset sourced from `opendata.socrata.com`.

Burger King Data Clean-Up

Follow the steps below to complete this data clean-up project.

1. Create a new table called `burger_king_locations_clean` that contains the following fields: `longitude`, `latitude`, `street_address`, `city`, `state`, `zip`, and `phone`. Use the source dataset to determine the data types (and sizes) to use for each field.

2. Using SQL and PL/SQL, process each row from the original `burger_king_locations` table, extracting information as necessary to populate the new `burger_king_locations_clean` table. While the `longitude` and `latitude` fields likely require no changes, the original `address` field must be split up and parsed into the new `street_address`, `city`, `state`, and `phone` fields. Note that you can ignore the data source's `name` field (unless you find it useful), as well as the target `zip` field.

   Use any SQL and PL/SQL techniques that you would like to complete this step. If you detect rows that are undecipherable, store those problem rows in a newly created table called `burger_king_locations_unresolved`; this table consists of the same fields as the original table, as well as a new `problem_desc` field that contains a short string describing the problem encountered while trying to clean up the given row (e.g., missing city and state). Be sure your total row counts make sense (in other words, do not skip any rows from the original data).

3. The next task is to fill in the missing zip code values. To do so, first download the freely available zip code database from the link given on the course website, then view the `zipcode.csv` file to get familiar with the data. Ignore the other files included in the download.

4. Create a new `zipcodes` table that consists of the following fields (corresponding to the `zipcode.csv` file): `zip`, `city`, `state`, `longitude`, and `latitude`.

5. Write a program to convert the `zipcode.csv` file into a series of SQL `insert` statements. You may use any programming language you like to accomplish this (e.g., write a Java program, a Python script, etc.). More specifically, your program should read the `zipcode.csv` file and process each line to produce a single `zipdata.sql` file that contains all of the `insert` statements.

   Example output should look like this:

   ```
   ...
   insert into zipcodes ( zip, city, state, longitude, latitude )
    values ( 12865, 'Salem', 'NY', -73.3526, 43.20035 );
   insert into zipcodes ( zip, city, state, longitude, latitude )
    values ( 12866, 'Saratoga Springs', 'NY', -73.7704, 43.0804 );
   insert into zipcodes ( zip, city, state, longitude, latitude )
    values ( 12870, 'Schroon Lake', 'NY', -73.758, 43.83538 );
   insert into zipcodes ( zip, city, state, longitude, latitude )
    values ( 12871, 'Schuylerville', 'NY', -73.5963, 43.08892 );
   ...
   ```

6. Execute the generated script to insert all of the zip code data into the `zipcodes` table.

7. Write an Oracle PL/SQL stored function that takes as input `longitude` and `latitude` values. Using the distance formula, determine the closest match in the `zipcodes` table. Return the zip code (or `NULL` if no match could be determined).

8. Write a script that updates the `burger_king_locations_clean` table by filling in the `zip` field for each row that has valid `longitude` and `latitude` values (i.e., for which your stored function returns a non-`NULL` value).

9. Similar to the previous step, write a script that updates the `burger_king_locations_clean` table by filling in all missing `city` and `state` fields. In other words, given a `longitude` and `latitude` lookup, use the closest city and state value. To accomplish this, create another Oracle PL/SQL stored function similar to step 7 above.

10. As a final step, create a `finalreport.sql` script that shows all Burger King locations ordered by zip code.

Submission Instructions

To submit your work, create a single ZIP file (or compressed folder) containing all of your source files, including the code used to generate the SQL `insert` statements for the `zipcodes` table. Use your Saint Rose ID (e.g., `goldschmidtd168`) as the name of the ZIP file (i.e., `goldschmidtd168.zip`).

Though entirely optional, you can include a simple `README.txt` file with notes or instructions.

Email your ZIP file to `goldschmidt@gmail.com` (with a subject line of “`CSC 380/530 Project 2`”).