Winter 2015	**CSC/CPE 365:** Database Systems	Alexander Dekhtyar

Lab 3: Potpourri, Part 2

Due date: Tuesday, February 3, 11:59pm

Note: Lab 4 will be assigned on Tuesday, February 3, in class.

Lab Assignment

Assignment Preparation

This is an individual lab. Each student has to complete all work required in the lab, and submit all required materials **exactly as specified** in this assignment.

The lab has two sets of tasks:

- **Fix your databases.** You will receive the grades for your database creation scripts from **Lab 2**. You are required to fix all errors, and submit the final version of your databases.
- **Data manipulation tasks.** Similarly to **Lab 3-1**, there are four data manipulation tasks in this lab. The tasks use the four datasets, BAKERY, CSU, INN and AIRLINES, which were not used in **Lab 3**.

Fixing the Databases

You will receive your **Lab 2** grades (for the portion of the lab covering `CREATE TABLE` statements). The graded `<DATABASE>-setup.sql` scripts are given to you in hardcopy, with any errors clearly marked.

You need to do the following:

1. fix all errors and finalize the content of `<DATABASE>-setup.sql` files;
2. ensure that all insertions proceed correctly with the new `setup` scripts, fix any issues;

3. submit fixed script collections for ALL nine datasets (including the ones, for which there are no data manipulation assignments in this lab).

Submission filenames. You will submit the following files for each dataset:

- `<DATABASE>-setup.sql`: your `CREATE TABLE` statements.
- `<DATABASE>-build-<file>.sql`: one script per table that inserts all tuples in the table. Each tuple must be inserted using a separate INSERT INTO statement.
- `<DATABASE>-cleanup.sql`: the `DROP TABLE` script.
- `<DATABASE>-test.sql`: the database test script.

In addition, for the AIRLINES, BAKERY, CSU and INN datasets, you will include `<DATASET>-modify.sql` file that performs the actions specified below.

Data Manipulation Tasks

The assignments in this part are specific to individual databases you created in **Lab 2**. Please execute them only on the specified datasets. The assignments may ask you to change both the schemas and the instances of the databases.

[**AIRLINES dataset.**] Create an SQL script `AIRLINES-modify.sql` which performs the actions described below.

You are modeling corporate takeover of a number of routes from one airport in the database.

1. Remove from the flights database all flights except for those to and from `AOS` (that's the airport code).
2. You will be modeling corporate takeover by `Northwest` of all flights except those operated by `Virgin` and `Jet Blue` to and from `AOS`. For this assignment (and this assignment only), you will look up the numeric IDs for each airline, and substitute them for airline names in the commands you need to develop.
3. For all flights NOT operated by `Northwest`, `Jet Blue` or `Virgin`, increase the flight number by 2000 (this will ensure that after the corporate takeover, flight numbers are still unique).
4. All flights in the `Flights` table come in pairs. The first flight starts at Airport 1 and goes to Airport 2. The second flight goes from Airport

2 to Airport 1 (the return flight). The flight numbers of these two flights are different by 1 - one flight number is even, one is odd.

For all pairs of flights to/from `AOS` NOT operated by `Northwest`, `Jet Blue` or `Virgin`, you need to *flip* the flight numbers. That is, if a flight from `AOS` to some other airport had an even flight number N, it needs to be replaced by $N + 1$, while, the flight number for a return flight will change from $N + 1$ to N. Conversely, if a flight from `AOS` has an odd flight number N, it needs to be replaced by $N - 1$, while the flight number of the return flight needs to change from $N - 1$ to N.

(In other words, all even-numbered flights need to increase by 1, all odd-numbered flights need to decrease by 1.)

Write a sequence of SQL commands that achieve this result.

NOTE: You can use built-in MOD(N,D) function which returns the remainder of deleting N by D.

Also, you can perform any *temporary* operations with the `Flights` table you want to achieve this task, as long as you *clean up* after yourself - i.e., as long as after all the commands are completed, the only change in the `Flights` table is the flipped flight numbers.

5. Complete the corporate takeover. Replace the airline for all flights to and from `AOS` except for `Jet Blue` and `Virgin` with `Northwest`.

6. Output the contents of the `flights table` using the following SQL statement:

```
SELECT *
FROM <Flights-table>
ORDER BY <Airline-column>, <flight-no-column>;
```

replacing `<Flights-table>` with the name of the flights table in your database, and replacing `<Airline-column>` and `<flight-no-column>` with the name of the appropriate columns in that table.

[**BAKERY dataset.**] Create an SQL script `BAKERY-modify.sql` which performs the actions below.

1. Remove from the table containing the listing of the pastries, information about all pastries except for `Cake`s.

2. The bakery manager is looking to adjust the prices of the pastries to accommodate for seasonal changes in the prices of the ingredients. Fresh fruit is more expensive in the winter, so cakes that have fruit in them become more expensive. Some other cakes are not selling as well as expected, so the manager wants to lower some of the prices.

3. Increase the prices of the `Strawberry Cake` and the `Lemon Cake` by 20%.

4. Reduce the prices of `Chocolate Cake` and `Napoleon Cake` by $2.

5. Reduce the price of all other cakes by 10%.

6. Show the contents of the table using the following SQL statement:

```
SELECT *
FROM <Pastry-table>
ORDER BY GID;
```

   replacing `<Pastry-table>` with the name of the table containing the list of pastries in your database.

[**CSU dataset.**] Create an SQL script `CSU-modify.sql` which performs the actions below.

For this assignment (and this assignment only), you need to look up the campus ID number for each campus name mentioned below, and the numeric discipline enrollment ID for each discipline mentioned below, and use these Id numbers in the commands where campus identity and/or discipline identity is used.

1. Keep in the table documenting *campus enrollments by discipline* only the information about the following enrollments:

   - Undeclared majors from Cal State Fullerton and Cal State Long Beach.
   - Enrollments at Cal Poly San Luis Obispo in disciplines with non-zero graduate enrollment and the undergraduate enrollment under 500.
   - Engineering enrollments exceeding 2000 undergraduates at any CSU campus.

2. Output the remaining contents of the discipline enrollments table using the following SQL statement:

```
SELECT *
FROM <Discipline-Enrollments-Table>
ORDER BY <CampusID-column>, <DisciplineId-column>
```

   replacing `<Discipline-Enrollments-Table>` with the name of the discipline enrollments table in your database, and replacing `<Campus-ID-column>` and `<DisciplineId-column>` with the names of the appropriate attributes in that table.

3. Keep in the table documenting CSU fees only the information that matches ALL the conditions below:

   - The fee is greater than $2.500;

- The year is either 2002 or one of 2004—2006.
- The campus is either Cal Poly San Luis Obispo, Cal Poly Pomona or San Jose State.

4. Output the remaining contents of the fees table using the following SQL command:

```
SELECT *
FROM <Fees-table>
ORDER BY <CampusId-column>, <Year-column>
```

replacing `<Fees-table>` with the name of the CSU fees table in your database, and replacing `<CampusId-column>` and `<Year-column>` with the names of the appropriate columns in that table.

[INN dataset] Create an SQL script `INN-modify.sql` which performs the actions below.

- Add to the table storing information about the rooms an attribute to store the description of a room. The room description is a short text (about double the standard tweeter message length).
- Create a somewhat meaningful description for each room and place the description into the record for the respective room. The descriptions should be loosely based on the name of the room and other observable (from the table) room features. For example, for "Thrift and accolade" room, a description might be

  "Experience the luxury of our Bed & Breakfast for half the price! Cozy room overlooking picturesque garden."

- Output the contents of the rooms table using the following SQL command:

```
SELECT *
FROM <rooms-table>
ORDER BY <Room-Id> \G
```

replacing `<rooms-table>` with the name of your rooms table and `<Room-id>` with the name of the attribute for the three-letter room code. Note the
`G` at the end of the query. This will force MySQL to output results in a row-by-row fashion, rather than in a table.

Submission Instructions

Please, follow these instructions exactly. Up to 10% of the Lab 3 grade will be assigned for conformance to the assignment specifications, **including the subimssion instructions**.

Please, **name your files exactly as requested** (including capitalization), and submit all files **in a single archive**. Correct submission simplifies grading, and ensures its correctness.

Please include your name and Cal Poly email address in all files you are submitting. If you are submitting code/scripts, include, at the beginning of the file, a few comment lines with this information. Files that cannot be authentificated by observing their content will result in penalties assessed for your work.

Specific Instructions

You must submit all your files in a single archive. Accepted formats are `gzipped tar` (`.tar.gz`) or `zip` (`.zip`).

The file you are submitting must be named `lab3.zip` or `lab3.tar.gz`.

Inside it, the archive shall contain all nine dataset directories. `AIRLINES`, `CSU`, `BAKERY` and `INN` directories must contain the appropriate `<DATABASE>-modify.sql` files. All directories must contain the new versions of the database creation, deletion and test files.

In addition, the root of the directory must contain a `README` file, which should, at a minimum, contain your name, Cal Poly email, and any specific comments concerning your submission.

Submit your archive using the following `handin` command:

Section 01:

```
handin dekhtyar lab03-01 <file>
```

Section 04:

```
handin dekhtyar lab03-04 <file>
```

Testing

Your submission will be tested by running all scripts you supply and checking the produced output for correctness. I may also use some extra scripts to verify the correctness of the databases you have constructed.

If you are aware of any bugs, or incorrect behavior of your SQL scripts, I strongly suggest that you mention it in the README file.