Cal Poly Spring 2013	**CPE/CSC 365** Introduction to Database Systems	Alexander Dekhtyar Eriq Augustine

Lab 3: DML Potpourri

Due date: April 24th.

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.

Note on data. As of this lab, all students will be using the same datasest. These files are a collection of files similar to the ones that you created in **Lab 2**. The table schemas should be mostly the same, but make sure to review the specific table and attribute naming. You can get these files at: `http://users.csc.calpoly.edu/$\sim$eaugusti/365-files/misc/datasets.tar.gz`.

Tasks

You will have to perform a series of tasks for each dataset. The tasks may involve any combination of changing the table schema, modifying tuples, and deleting tuples. All the SQL commands necessary to perform each task will be in a single file (per dataset) named: <DATASET>`-modify.sql`.

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

You will modeling corporate takeover of a number of routes from one airport in the database. For this assignment only, you are allowed to look up and hard code Airline ids.

1. Remove 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.)

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`.

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

1. Drop the `Items` table. The **referential integrity** enforced by this table will only get in the way of this exercise.
2. Remove from the `Goods` table, information about all pastries except for `Cake`s.
3. 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.
   (a) Increase the prices of the `Strawberry Cake` and the `Lemon Cake` by 20%.
   (b) Reduce the prices of `Chocolate Cake` and `Napoleon Cake` by $2.
   (c) Reduce the price of all other cakes by 10%.

[**CARS dataset.**] Create a SQL script `CARS-modify.sql` which performs the following actions.

1. Keep in the `CarData` table **ONLY** the records about vehicles that:
   (a) were made in 1981 and after with accelerations between 14 and 15 (inclusive) OR
   (b) are heavier than 4900 lbs.
2. Only keep records in the `CarData` table that have a known mpg.
3. Convert the `CarData` table (`mpg`, `eDispl`, and `weight`) to metric! Rename any columns that no longer make sense. Use the following conversions:
   (a) 1 m/g = 0.4251 k/l
   (b) 1 cubic inch = 16.3871 cc
   (c) 1 lb = 0.4536 kg

[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 `DisciplineEnrollments` table 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. Keep in the `Fees` table 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.

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

1. Keep in the `Reservations` table only the reservations that start in the first half of September of 2010 (September 1 through September 15 inclusively) that involve no kids.
2. Keep in the `Reservations` table only the columns for the reservation code, room code, checkin date, last name on the reservation, and the number of adults.
3. The interior decorator has redone some of the rooms.
   (a) All `rustic` rooms with a `Queen` size bed have been converted to the `rusty` decor.
   (b) All `traditional` rooms with a `King` size bed have been converted to the `post-modern` decor and now have a `SuperKing` size bed.

WRNING: Check your MySQL warnings.

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

1. Remove from the `Marathon` table results for all runners except for the top four places.
2. Remove from the `Marathon` table all attributes except for the place, the total time running, and the pace of the run.
3. Add a new attribute to the `Marathon` table. The attribute should represent the total running time in seconds. Give it an appropriate name and type.
4. Update the `Marathon` table, computing the running time in seconds correctly based on the information extracted from the total running time attribute. (Note: Numeric pieces of temporal types (like the seconds from a TIME type) may be used in arithmetical expressions).

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

Extend the database structure to include the information about the GPA for each student.

Update the database as follows:

- All kindergarten students are assigned GPA of 3.8.
- All students in classroom 112 are assigned GPA of 2.8.
- All 1st graders who are not in classroom 102 are assigned GPA of 3.0.
- All other students are assigned GPA of 3.2.
- `ELTON FULVIO`, `ANIKA YUEN` and `JANEE DANESE` get a GPA of 4.0 (this should override whatever other rules for their GPA).

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

1. Remove the columns storing the appellation name and the name of the wine from the `Wine` table. **Hint:** You *may* have to drop the foreign key constraint before you can remove some columns.
2. Keep in the `Wine` table only the wines with scores of 97 or higher.
3. Modify the length of the attribute storing the winery name to be 14 characters long[1].
4. The remaining wines are some of the best wines in the world. Their price appreciates with time. Update the price of each wine using the following formula:

$$Price := Price + \frac{1000 - < \mathsf{Cases} >}{< Score >} \cdot 10,$$

where `<Cases>` is the number of cases of the wine produced and `<Score>` is the score of the wine.

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). 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 authenticated by observing their content will result in penalties assessed for your work.

[1] If you did everything right, all winery names in the remaining tuples will be shorter than 14 characters.

Specific Instructions

No archives are expected for this lab. You are expected to handin one file per dataset and an optional README.txt including any comments about this lab.

Submit your files using the following `handin` command:

$ handin eaugusti lab03 AIRLINES-modify.sql BAKERY-modify.sql CARS-modify.sql CSU-modify.sql INN-modify.sql MARATHON-modify.sql STUDENTS-modify.sql WINE-modify.sql [README.txt]

Tip: A command like this may be more convenient:

$ handin eaugusti lab03 *-modify.sql [README.txt]