Lab 3-1: Potpourri Part 1

Due date: Thursday, January 29, beginning of the lab period

Note: Lab 3-2 will be assigned during on January 29 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.

Note on data. This lab will require you to use the files you prepared in **Lab 2**. You will complete four assignments, using four of the nine course databases.

You will have to use (and resubmit as part of **Lab 3** submission) the files you submitted for Lab 2 to set up the appropriate databases. These files will be graded, and the results will be returned to you on Tuesday, January 27. As part of this assignment, you are required to submit the corrected versions of the CREATE TABLE statements for your databases that take into account the comments given to you.

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 ask you to change both the schemas and the instances of the databases.

[**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.

For the GPA, make certain that only GPAs in the range between 0.0 and 5.0 are allowed. Update the database as follows:

- Keep in the database only the students from grades 1 and 2.
- Set the GPA of first graders from room 102 to 3.0.
- Set the GPA of first graders from other rooms to 3.1.
- Set the GPA of all second graders to 3.2.
- The following instructions apply to individual students and override all prior GPA assignments.
  - Set the GPA of `ANIKA YUEN` to 3.5.
  - Set the GPA of `LANCE HOOSOCK` to 3.6
  - Set the GPA of `SUMMER LAPLANT` to be 0.7 higher than whatever it currently is.
  - Set the GPA of `KERI TRAYWICK` to be 25% higher than her current GPA.

Include all necessary SQL commands to achieve this result into the `STUDENTS-modify.sql` script. Complete the script with the

```
SELECT * FROM <students-table>
ORDER BY <GPA-column>, <grade-column>, <student-lastname-column>;
```

query, replacing `<students-table>` with the name of your table containing the list of students and `<GPA-column>`, `<grade-column>` and `<student-lastname-column>` with the names of the columns storing the GPA, the grade level of each student and their last names respectively.

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

1. Remove the columns storing the appelation name and the name of the wine from the table storing the list of wines (we refer to this table as "the wine table").
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].

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

4. Output the list of wines using the following SQL query:

```
SELECT * FROM <wine-table>
ORDER BY <wine-id-column>;
```

   (replace `<wine-table>` and `<wine-id-column>` with appropriate, in your database, names).

5. 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 idea: the price of wine increases by $X\%$ a year (a simple increase rate, not a compound one) for each year of vintage, where X is the difference between the score of the wine and the score 90 (that is, a 99 point wine increases in price by 9% a year). Assuming the price of wine in the table at the moment reflects the price during the production year. Update the price to what it should be in 2015 (part of the task is to figure out the correct update formula).

6. Output the list of wines using the following SQL query:

```
SELECT * FROM <wine-table>
ORDER BY <wine-id-column>;
```

   (replace `<wine-table>` and `<wine-id-column>` with appropriate, in your database, names).

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

1. Keep in the table storing the technical characteristics about the cars (we refer to this table as "the car data table"), ONLY the records that satisfy *at least one* of the following conditions:
   (a) vehicles made in 1981 and after with accelerations between 14 and 15 (inclusive).
   (b) vehicles that are heavier than 4900 lbs.

2. Run the following SQL query:

```
SELECT *
FROM <car-data-table>
ORDER BY <year-column>, <car-Id>;
```

   where `<car-data-table>` is the name of the car data table in your `CARS` database and `<year-column>` is the column in that table storing the year in which a vehicle was made and `<car-id>` is the unique Id of each tuple in the car data table.

3. Remove from the car data table all attributes except car id, car year, acceleration and weight.

4. Remove from the car data table information about any cars that weigh between 2000 and 3000 pounds.

5. Run the

```
SELECT *
FROM <car-data-table>
ORDER BY <year-column>, <car-Id>;
```

   query again.

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

1. In the table specifying which instruments the band members play on each song (we refer to it as the "instruments table" below), replace all occurrences of `'bass balalaika'` with `'awesome bass balalaika'`, and all occurrences of `'guitar'` with `'acoustic guitar'`. (Please note that you may need to change the length of the instrument name field if it is not long enough in your table - do it using a table schema modification command, rather than in the CREATE TABLE statement).

2. Keep in the instruments table only the information about `'acoustic guitar'` players and the information about all instruments `Turid` (her band member id is 4 - you can use it directly) played on all songs.

3. Run the following SQL query:

```
SELECT *
FROM <instruments-table>
ORDER BY <Song-id>, <Bandmate-Id>;
```

   where `<instruments-table>` is the name of your instruments table, and `<Song-id>` and `<Bandmate-Id>` are the columns in this table with the song and band member Ids respectively.

4. Add a new attribute to the table describing the albums released by Katzenjammer. The attribute should store the total number of songs on the album.

5. Based on information stored in the tracklists table (look up the CSV file if you have to), update each record in the albums table to show the correct number of tracks.

6. Run the following SQL query:

```
SELECT *
FROM <albums-table>
ORDER BY <Year>;
```

where `<albums-table>` is the name of your table of Katzenjammer albums, and `<Year>` is the name of the column in the albums table that stores the year of the album release.

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 submission 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 four directories named `CARS, KATZENJAMMER, STUDENTS` and `WINE` 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.

Each directory shall contain all SQL scripts built by you for the specific dataset in response to all parts of the lab. The Lab 2 scripts must be resubmitted, with the correct names. (these are the `<Dataset>-setup.sql`, `<Dataset>-build-<table>.sql` and `<Dataset>-cleanup.sql` files). New SQL scripts must be named as specified in the assignments above[2].

Submit your archive using the following `handin` command:

Section 01:

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

Section 04:

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

[2]After Lab 3-1 and Lab 3-2 we will consolidate your `<Dataset>-build-<table>.sql` files into a single one, but for now, keep all files separate.

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.