Lab 2: Database Creation

Due dates:
Part 1: Tuesday, October 15, 12:00pm
Part 2: Thusrday, October 17, 12:00pm

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.

For this assignment, you will be using your MySQL account.

You will be using the MySQL server hosted at `mysql.labthreesixfive.com` and the MySQL client available on the CSL machines. Please refer to your MySQL handouts distributed in class in order to gain access to the MySQL server as well as perform most of the work for this assignment.

LabThreeSixFive.com sandbox environment is available for you to use for development of parts of this assignment, but I did not create a LabThreeSixFive.com lab for you to work on it - primarily due to the complex structure of your submission.

For the purposes of this, and future assignments, an SQL script is a text file that contains a sequence of SQL statements and MySQL commands, as well as SQL comments. Typically, these files should receive a .sql extension. You will be required to prepare and submit a number of SQL scripts for this lab. You can develop individual statements using LabThreeSixFive interactive functionality, or using direct access to `mysql.labthreesixfive.com` (or both interchangeably), but you must eventually place all statements you develop in SQL scripts described in this assignment, and submit your files **precisely following the submission instructions** provided below.

Lab Assignment Parts

We separate Lab 2 into two parts, and set two due dates.

Part 1 involves submission of only one type of files: the `<DATABASE>-setup.sql` files for each of the nine datasets described below (these files will contain your **CREATE TABLE** statements). This part of the assignment will be graded by the instructor manually.

Part 2 involves submission of all required lab deliverables (including the `<DATABASE>-setup.sql` files that you submitted in Part 1). Please follow detailed submission instructions for this part, as correctly organized submission and correctly named files speed up our grading efforts (while incorrectly named files, and errors in submission yield penalties). This part of the lab will be graded by the course grader by actually running your scripts.

The Datasets

Starting with this lab, and for most of the remaining labs in the course you will be working with several datasets created specifically for this course. These datasets are not large, but they are sufficiently diverse in content, scope and structure.

All datasets are available from the course web page:

http://users.csc.calpoly.edu/~dekhtyar/365-Fall2024

Each dataset comes with a README file, which contains the exact specifications of all data files included in the dataset, and briefly explains the meaning of the dataset. Before starting your work on a dataset, **please study carefully the README file and make sure you understand the structure of the dataset!** All data files in each dataset are stored in the CSV (comma-separated values) format. All text values are enclosed in single quotes. A .zip file for each dataset is avilable.

Brief descriptions of each of the course datasets are given below.

CSU dataset. Type: multidimensional statistical data. This dataset contains various statistics about the California State University system. Historic information, such as annual enrollments and graduations is included, as well as information about enrollments by discipline at all campuses in a single year, and information about faculty lines and campus fees.

CARS dataset. Type: normalized[1] dataset. This database stores information about the properties (such as number of cylinders, milage per gallon, engine displacement, etc) for over 400 domestic and import cars produced between 1970 and 1982. Information is split into several files: starting with lists of continents and countries, going onto the lists of automakers and the models and makes they were producing.

BAKERY dataset. Type: OLTP (on-line transaction processing) dataset. This dataset records information about one month of sales from a small bakery to a list of its dedicated customers. The dataset captures the notions of a transaction (a single purchase) and market baskets (each purchase may contain more than one item).

MARATHON dataset. Type: universal table. This dataset consists of a single CSV file documenting the performance of participants of a half-marathon race. The performance is tracked for the entire race, as well as within each gender/age category.

STUDENTS dataset. Type: simple normalized. This is a variation of the dataset you have encountered (different names, no buses, no GPA) in Lab 1. The dataset consists of a list of students assigned to grades and classrooms, and a separate list of teachers assigned to classroms. This is the simplest dataset and can be used as "staging grounds" for most of the activities in this and future labs.

AIRLINES dataset. Type: graph. The dataset stores information about a number of airlines, and the flights these airlines have between 100 different airports. It can be viewed as a multicolored graph with 100 nodes representing airports, edges representing direct flights and edge colors representing the airlines running the flights.

WINE dataset. Type: normalized (somewhat). The dataset lists ratings of a variety of single-grape California wines of different vintages as given by the Wine Spectator magazine. The dataset consists of a list of wines complete with their ratings on a 100 point scale, their reported sales prices and the production volumes. Two additional lists: appelation/American Viticultural Areas (AVAs) and grape varieties are available as well.

INN dataset. Type: simple OLTP. The dataset contains information about one year's (2010) worth of completed hotel reservations for a small Bed & Breakfast inn. The inn contains 10 uniquely named rooms - each with its own set of features. The dataset lists all reservations that *commenced* 2010[2] and specifies for each room reservation the checkin and checkout dates, the name of the person making the reservation and the number of adults and kids staying in the room.

[1] In CS 366 you will get an opportunity to study the formal meaning of the term "normalized" when applied to databases. An informal explanation: a "normalized" database is a database where information is split into a large number of tables, reducing redundancy in data.

[2] Some reservations made at the end of the year end in 2011.

KATZENJAMMER dataset. Type: normalized[3]. The dataset follows the recording and performing career of an all-female Norwegian band Katzenjammer[4]. The four members of the band, Sloveig, Marianne, Anne-Marit and Turid turn their live shows into a cascade of instrument changes. The dataset documents the albums the band released, the songs on the albums, and the typical ways in which the songs are performed live: the instruments each band member plays, the vocals, and their position on the stage.

Examples of the band's performances can be found in abundance on Youtube.

Your Lab 2 assignment should be performed for each of the datasets in the list above. We separate the datasets into two tiers:

1. Tier 1. Datasets that result in tables with only `INTEGER`, `FLOAT` (or `DOUBLE`) and `VARCHAR` attributes. These datsets are:
   - `STUDENTS`
   - `CSU`
   - `CARS`
   - `AIRLINES`
   - `WINE`
   - `KATZENJAMMER`
2. Tier 2. Datasets in this tier have at least one table with `DATE` attributes. Handling dates and times in MySQL requires some care, primarily because the datasets we have use a date format that is different from MySQL's default `DATE` format. Because of it, you will first work on the Tier 1 datasets. Tier 2 datasets are:
   - `BAKERY`
   - `MARATHON`
   - `INN`

The Task

For this lab you will design a relational database for each of the datasets, and will instantiate it with the data provided to you.

Database design and creation.

All tasks outlined in this section must be performed on all datasets.

You need to do the following:

1. For each dataset, create a collection of tables to store its data (we refer to such a collection as a "database", but your code should NOT contain `CREATE DATABASE` commands). The following rules must be followed:
   (a) The tables of the database must match the files of the dataset one for one.
   (b) You are allowed to choose any (hopefully meaningful and non-offensive) names for all relational tables and columns in them.

[3] In the course we use the Version 2.0 of this dataset, created in September of 2016.

[4] http://www.katzenjammer.com

(c) You must properly detect and declare all constraints, including primary key, candidate key (SQL's UNIQUE), and referential integrity/foreign key constraints. Correct detection of these constraints is mandatory. Each table **must have** an explicity declared primary key constraint. There are additional types of constraints (`NOT NULL`, `CHECK`, and a few more) that you are allowed to declare, but these are left to your sensibilities.

2. Write and test a SQL script for creation of each database (one script per database). This script *shall only contain* the `CREATE TABLE` statements and name these scripts `<DATABASE>-setup.sql` where `<DATABASE>` is the name of the specific database (in ALLCAPS). For example, `BAKERY-setup.sql`, `INN-setup.sql`, etc....

3. Write and test a SQL script for deleting all tables from each database (one script per database). Name the scripts `<DATABASE>-cleanup.sql` (e.g., `CSU-cleanup.sql` or `CARS-cleanup.sql`).

4. Prepare and test SQL scripts for populating each database with the data avilable from the .csv files. (Hint: use any available programming/scripting language to convert the .csv file into a list of SQL statements for adding records to the database. There is a relatively simple way to solve this problem (almost) forever with a single Python script, for example). You should have one SQL script per database table. Each script you produce *must contain only* `INSERT` statements. For this lab you must use *one `INSERT` statement per tuple*. (This is to make sure that if you have an error in the script, you can debug it efficiently).

5. **Note:** While your table/column names can be any of your choice, the names of all scripts you must prepare and submit are defined in this assignment. Please refer to the **Submission Instructions** section for the specs for proper script naming. **Be aware that incorrect names for submitted files, EVEN incorrect capitalization will lead to significant deductions in your lab score!**

   You will have one script per table in each database. Name the scripts `<DATASET>-build-<table>.sql`, where `<DATASET>` is the name of your database, and `<table>` is the name of the CSV file (w/o the `.csv` extension) whose table you are created. For example, for the file `continents.csv` from the `CARS` dataset, the filename will be `CARS-build-continents.csv`. Use the correct spelling an punctuation: the script inserting data from `csu-fees.csv` shall be called `CSU-build-csu-fees.sql`, while the script inserting data from `Albums.csv` shall be called `KATZENJAMMER-build-Albums.csv`.

6. Write and test the scripts for checking the contents of each database. A simple statement to check the contents of a database table is

   ```
   SELECT * FROM <Table>;
   ```

   This statement will result in all records in a table printed in their entirety. The query

   ```
   SELECT COUNT(*) FROM <Table>;
   ```

   will output the number of tuples in the table.

   Name your test scripts `<DATASET>-test.sql`. For example, `BAKERY-test.sql` or `AIRLINES-test.sql`.

WINE dataset: Special Notes

To make your life a bit more interesting, when creating the database for the `WINE` dataset, please, take into account the following information:

1. The list of appelations/AVAs contained in the `appelations.csv` file, lists a state for each appelation/AVA entry[5]. Similarly, there is a "State" column in the file `wine.csv`, describing the state of

[5] In the current dataset all appelations/AVAs are from California, but state information still shall be recorded for each appelation/AVA.

the origin of the wine. We note, that `wine.csv` does record the appelation/AVA for each wine (the "Appelation" column), and that the state of origin of the wine is essentially the state of the wine's appelation. Therefore, there is a duplication of this information in the CSV files in the dataset.

2. The last column of the `wine.csv` file, "Drink" is designed to contain the advice of the wine raters on when the wine is best consumed. Most of the entries contain a text value `'now'`. Some entires contain a number representing the year when the wine will start being at its best.

For your `WINE` database you should do as following:

- The State information should be included as part of the appelation/AVA information.
- State information SHOULD NOT be included with the wine score (including appelation information will suffice).
- the drink advice column SHOULD NOT be included with the wine score information (just skip it).

Note 1: There is a number of other interesting features in this dataset regarding the primary/foreign key identification. Be careful!

Note 2: There is some missing data from this dataset - some wines do not have production volume information. You have to deal with this properly.

Note 3: In winemaking, an "appellation" is essentially a region from which the grapes used in making the wine originate. The most broad appelation in our dataset is the entire state of California. According to US law, county names are legal/valid descriptions of grape origins. US recognizes other regions as legal/valid specifications of grape origins. Such regions are called AVAs, a.k.a., American Viticultural Areas. In California, AVAs can cross county borders and also be embedded into one another. For example, "Russian River Valley" AVA is located inside "Sonoma Valley" AVA, while "Lodi" AVA spans multiple counties. For simplicity, in the latter cases, `appelations.csv` file lists the main county of an AVA. Some AVAs are larger than a county. E.g., "Central Coast" and "North Coast" AVAs span multiple counties. For such AVAs, `appelations.csv` lists the value `'N/A'` in the "County" column.

File `appelations.csv` lists all grape origin specifications that can appear on wine labels. The last column specifies whether the origin specification is an AVA or not.

Submission Instructions

Part 1 submission

In **Part 1** of the lab, you are submitting **only** the nine `<DATABASE>-create.sql` files. Put these files into a single directory, and either zip them, or tar and then gz them to create

`lab2-1.zip` or `lab2-1.tar.gz` file. Submit this file using the following `handin` command:

```
$ handin dekhtyar 365-lab02-1 <file>
```

Part 2: General Instructions

In **Part 2** of the lab, you are submitting **all files** described in the assignment. This **includes** the `<DATABASE>-setup.sql>` files you submitted in Part 1. **However**, your files shall be organized differently. **Please, follow the instructions below instructions exactly.** Up to 25% of the Lab 2 Part 2 grade will be assigned for conformance to the assignment specifications, **including the subimssion instructions**.

Please, **name your files exactly as requested** (including capitalization and any typos present in instructor's filenames), 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 creating for your final submission must be named `lab2.ext`, i.e., either `lab2.tar.gz` or `lab2.zip`

Inside it, the archive shall contain nine directories named `AIRLINES, CSU, CARS, BAKERY, KATZENJAMMER, MARATHON, STUDENTS, WINE` and `INN` (same as the dataset names). 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.

INCORRECT SUBMISSION. A submission that unpacks a single directory (e.g., `lab02/`) which, in turn contains the nine directories described above in it is considered to be **INCORRECT**. Also, please make sure the file name capitalization and spelling for all files is correct!

Each directory shall contain all SQL scripts built by you for the specific dataset. The scripts shall be named as follows.

Filenaming conventions (repeating from above)

- database creation scripts. The filename shall be `<DATASET>-setup.sql`. Here, `<DATASET>` is the name of the directory in ALL CAPS. E.g., for the CARS dataset, the filename will be `CARS-setup.sql`.
- table deletion scripts. The filename shall be `<DATASET>-cleanup.sql`. E.g., for the CARS dataset, the filename will be `CARS-cleanup.sql`.
- table population scripts. The filename shall be `<DATASET>-build-<table>.sql`. Here `<table>` is the name of the `.csv` file (not the name of the table that you are building).

  For example, for the table populating the list of car makers in the CARS database, the filename will be `CARS-build-car-makers.sql`. Use **the same capitalization** as the filename in the dataset: e.g., `CSU-build-Campuses.sql` and `INN-build-Rooms.sql`, `CARS-build-cars-data.csv` and `BAKERY-build-items.s`
- testing scripts. These are scripts described in item 6 of the assignment description. The filename shall be `<DATASET>-test.sql`. E.g., `CARS-test.sql`

Submission of the main assignment

Submit the full Lab 2 Part 2 assignment using the following `handin` command:

```
$ handin dekhtyar 365-lab02-2 <file>
```

Testing

Your Part 2 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.

A test script will be released to you before the due date for the lab. **Please**, do not submit your work without first running the test script on it.

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.