Lab 3: Potpourri

Due date: Monday, January 28, 11:59pm

Note: Lab 4 will be assigned on Monday, January 28 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 eight 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. Due to the lab due date and the grading schedule, you won’t see your CREATE TABLE statements graded until the very end of Lab 3 timeline. Because of this, the following rules apply.

1. You will submit your <DATABASE>-setup.sql files. If you know of errors in them, you are allowed to correct them, i.e., you can submit improved versions of these and other Lab 2 files.

2. You WILL NOT receive deductions on your Lab 3 grades due to errors in your Lab 2 files, if the mistake in those files do not prevent correct operation of grading scripts.

That is, if we can create your database, populate your tables with all the tuples from the CSV files, run the Lab 3 scripts you submit and drop all tables successfully, you won’t see any deductions due to errors in how you created your tables.
3. You WILL receive deductions on your Lab 3 grades due to errors in your Lab 2 files if the proper work of grading scripts is impossible.

Examples of things for which you won't receive a deduction: forgetting to declare a foreign key, forgetting to declare a primary key, forgetting some UNIQUE constraints.

Examples of things for which you may receive a deduction, if they interfere with the work of the grading scripts: using incorrect key (and not being able to populate the table fully), specifying constraints that do not exist, which prevent tuples from being added to the tables, forgetting to declare a table, submitting a `<DATASET>-cleanup.sql` script that does not actually delete all tables.

**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.

One assignment asks you to show proficiency in learning *sqlplus* commands by properly formatting an output for a query.

**[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:

- All kindergarten students are assigned GPA of 4.0.
- 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 DANSE get a GPA of 4.0 (this should override whatever other rules for their 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 <student-lastname>;
```

query, replacing `<students-table>` with the name of your table containing the list of students and `<student-lastname>` with the name of the column storing the last names of the students.
[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 table storing the list of wines (we refer to this table as "the wine table" further).

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

   ```sql
   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 formula:

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

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

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

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

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

---

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

1. Remove from the marathon results table results for all runners except for the top four places.

2. Remove from the marathon results table all attributes except for the place, the total time running and the pace of the run.

3. Output the marathon results table using the following SQL query:

\(^{1}\text{If you did everything right, all winery names in the remaining tuples will be shorter than 14 characters.}\)
SELECT * FROM <marathon-table>
ORDER BY <place>;

4. Output the contents of the total running time column, formatting the time as <HH MI SS>. Use

```sql
SELECT <formatted-time-column> FROM <marathon-table>
ORDER BY <place>;
```

command, replacing `<formatted-time-column>` with the appropriate formatting expression.

5. Output the contents of the pace column, formatting the data as MI--->SS. Use

```sql
SELECT <formatted-pace-column> FROM <marathon-table>
ORDER BY <place>;
```

command, replacing `<formatted-pace-column>` with the appropriate formatting expression.

6. Add a new attribute to the marathon running time table. The attribute should represent the total running time in seconds. Give it an appropriate name and type.

7. Update the marathon table, computing the running time in seconds correctly based on the information extracted from the total running time attribute. (Note: any part of the DATE type that is extracted as a number can be used in arithmetical expressions).

8. Output the contents of the marathon table using the

```sql
SELECT * FROM <marathon-table>
ORDER BY <place>;
```

[CARS dataset] This assignment asks you to both manipulate the data in the database, and to output it using sqlplus formatting instructions.

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

1. Keeps in the table storing the technical characteristics about the cars (we refer to this table as "the car data table"), ONLY the following records:
   
   a) vehicles made in 1981 and after with accelerations between 14 and 15 (inclusive).
   
   b) vehicles that are heavier than 4900 lbs.
2. uses sqlplus formatting commands to set up the output format as described below.

   (a) Replace column names with the following column headings:
   i. 'Car' for the unique Id of each vehicle
   ii. 'Milage' for the column storing the milage of the vehicle
   iii. 'Engine Displacement' for the column storing the engine displacement of the vehicle
   iv. '0 to 60mph in...' for the column storing the 0 to 60mph acceleration
   v. 'Produced in' for the column storing the year the vehicle was manufactured

   (b) Set the length of line of output so that each output row of the SQL query specified below was printed on a single line. (Do not make the length of the line TOO long though, you will be penalized for that.)

   (c) Set the length of a page so that the entire output of the SQL query specified below fits on a single page.

   (d) Specify that the column separator in the output is ‘||’ (double vertical bar).

   (e) Specify that the character used to underline the header of the query is ‘.’ (dot).

   (f) Specify that in the output of the SQL query given below, each year of vehicle production should appear exactly once, and the set of records for a single year, shall be separated from other records by one empty line.

3. Runs the following SQL query:

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

   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.

   The result of the query formatted as required is available from the Lab 3 data page. Your script must produce character-for-character same output, except for the names of the columns storing number of cylinders, horsepower and weight of the vehicle.

**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 authenticated 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, MARATHON, 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 same 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.

Submit your archive using the following handin command:

Section 01:

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

Section 02:

```
handin dekhtyar lab03-01 <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.