Lab 3: Potpourri

Due date: Tuesday, April 29, **at the beginning of the lab period**

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 main assignments. First, you need to complete your database creation and population scripts and make certain that all errors found in their Lab 2 versions are fixed. You will be using these scripts for the rest of the quarter.

Second, you have to perform the tasks specified in the assignments below. The tasks establish your mastery of the SQL DDL and DML, the sql*plus environment and your ability to work effectively with DATE data in SQL.

Part 1: Finishing SQL scripts

Your task is to ensure that your scripts properly create all course datasets. If you were successful in doing so in Lab 2, then, you do not need to do anything else for this task.

If your Lab 2 submission yielded errors, you need to fix them, and ensure that your scripts create proper databases and insert correct data in them.

This part of the lab does not get graded, however, (a) you will need to complete this task in order to get the remainder of this lab to work and (b) you will be using the scripts you finalize in this lab in the labs that follow.

Part 2: Working with databases

For each database create an SQL script (or scripts, where necessary) performing specified tasks.

AIRLINES database

Write an script, which performs the following actions.

1. Delete from the list of `flights` all flights except those flown by `United` (`UAL`) and `Allegiant`.
2. Output the list of flights in the database.
3. Prepare `Allegiant` for "corporate takeover." First, "flip" the flight numbers on `Allegiant` flights between each pair of airports. For example if the database contains flight `400` from `ANA` to `AEL` and flight `401` from `AEL` to `ANA`, after the issued commands, flight from `ANA` to `AEL` will have number `401`, and flight from `AEL` to `ANA` will have number `400`.

   Note, that you may have to perform this activity in more than one step. Note also, that in the current dataset, all flight numbers are smaller than 9999.
4. Output the list of flights in the database.
5. Now, change the flight numbers on all `Allegiant` flights to be larger by 2000. (this is done to guarantee no overlapping flight numbers with United).
6. Finally, reassign all `Allegiant` flights to `United`.
7. Remove `Allegiant` from the list of airlines.
8. Output the list of flights in the database.
9. Output the list of airlines in the database.

Note. In this assignment you are allowed to use the unique Id values for `United Airlines (UAL)` and `Allegiant Air` when manipulating data in the flights table.

BAKERY database

Do the following.

1. Re-write (if needed) your `CREATE TABLE` statements for the `BAKERY` dataset so that each constraint defined in the database is explicitly named.

2. Write an SQL script that does the following:

   (a) Keeps in the database information only about the pastry items (goods), whose price is greater than \$10.

   (b) Keeps in the database information only about the purchases (receipts) that occurred on October 2, 2007.

   (c) Increases the prices of all pastries in the database by 10%.

   (d) Outputs the list of pastries in the database.

   (e) Outputs the list of receipts in the database.

Hint. There is a reason why each constraint in this database needs to have a name.

CARS database

Write an SQL script that does the following:

1. In the table of car parameters (`cars-data`), keeps only cars with MPG better than 20 (records with `null` MPG shall be excluded from the database) produced in 1970.

2. Outputs the list of cars in the `cars-data` table.

3. Changes the schema of the car parameters table to keep only the information about the model id, milage per gallon and acceleration.

4. Outputs the list of cars in the `cars-data` table.

5. Changes the schema of the car parameters (`cars-data`) table to incoporate information about the price of the car and number of cars sold in the US.

6. Sets the price of the car to be equal to its MPG times acceleration times 10.[1]

7. For cars with acceleration under 15 seconds, sets the number of cars sold in the US to be equal to the square of the price times the square of the milage.

8. For other cars, sets the number of cars sold in the US to be equal to the square of the price times the square of the acceleration.

9. Outputs the list of cars in the `cars-data` table.

[1] Yes, this is quite unrealistic.

CSU database

Write an SQL script which does for following.

1. In the tables of student fees, degrees and faculty information keep only the information about Cal Poly (you may use Cal Poly's unique id.)
2. Outputs the contents of the students fees, degrees and faculty information tables.
3. In the table of enrollements, keeps only the information about (a) Cal Poly enrollments in the years when it was over 15,000; (b) enrollments for any campus for any year prior to 1936; (c) enrollments for any campus except for `San Diego State University` for years when the enrollment exceeded 30,000 and (d) enrollments for any campus except for `California State Univeristy - Channel Islands` for any year when the total enrollment was less then 300 students.
4. Outputs the contents of the enrollments table.

MARATHON database

Write an SQL script which does the following. Note, the results of all actions are **cumulative**.

1. Keeps in the results table information only about runners from the state of Massachussets.
2. Outputs the contents of the results table formatted so that each record fits a single line, and the output fits one page. Time should be formatted as `HH:MI:SS`, pace should be formatted as `MI:SS`.
3. Keeps in the results table information only about runners in the `20-39` age group.
4. Outputs the contents of the results table formatted so that each record fits a single line, and the output fits one page. Time should be formatted as `HH:MI:SS`, pace should be formatted as `MI:SS`.
5. Keeps in the results table only the results of runners from the town of `HALIFAX`.
6. Keeps in the results table only the columns for the overall place of the runner, his/her time, pace and name (first and last).
7. Outputs the contents of the results table formatted so that each record fits a single line, and the output fits one page. Time should be formatted as `HH:MI::::(SS)`, pace should be formatted as `__MI::SS___`.

STUDENTS database

Write an SQL script that does the following.

1. Replaces the names `ROBBY`, `BRITT`, `HILMA`, `LANCE` and `BENNIE` with the names `ROBERT`, `ROBERT`, `WILMA`, `LENNY` and `BERNARD` respectively in the table of students.

2. Outputs the list of students.

3. Converts all first and last names of students into lowercase, keeping the first letter capitalized.

4. Outputs the list of students.

5. Adds a new column to the table of students, representing the full name of the student. Sets the full name of the student to be his/her lastname, followed by the first name, separated by a comma. E.g., the full name of the student with first name `'Ray'` and last name `'Madlock'` is `'Madlock, Ray'`.

6. Removes first name and last name columns from the students table.

7. Outputs the list of students.

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.

From now on, 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-ilastname.ext`, where *i* stands for the initial of your first name, and *lastname* is your last name. E.g., if I were submitting this file, the name would be `lab2-adekhtyar.zip` or `lab2-adekhtyar.tar.gz`.

Inside it, the archive shall contain six directories named `AIRLINES, CSU, CARS, BAKERY, MARATHON` and `STUDENTS` after the datset names. In addition, the root of the directory must contain a `README` file, which should, at a minimum, contain your name, Cal Poly email, 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 scripts shall be named as follows:

- Lab 2 scripts. All scripts from Lab 2 should be resubmitted, with the same names.
- Lab 3 scripts. Name each script `<Dataset>-lab3.sql`. For example, for the `MARATHON` dataset, your script name will be `MARATHON-lab3.sql`.

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.

I will use one of my Oracle accounts to test your scripts - not your personal Oracle account.

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.