CS 325 - Homework 6

Deadline

11:59 pm on **Friday, October 22, 2021**.

Purpose

To read and think about normalization, and to write more SQL queries, including queries with nested selects/sub-selects, queries projecting concatenated expressions, and queries using `&` to allow interactive input into a script.

How to submit

Problem 1 is completed on the course Canvas site.

For Problem 2 onward:

Each time you wish to submit, within the directory `325hw6` on nrs-projects.humboldt.edu (and at the nrs-projects UNIX prompt, **NOT inside** `sqlplus`!) type:

```
~st10/325submit
```

...to submit your current files, using a homework number of `6`.

(**Make sure** that the files you intend to submit are listed as having been submitted!)

Additional notes:

- You are required to use the HSU Oracle `student` database for **Problem 2** of this homework.
- **DB Reading Packet 6** and **SQL Reading Packet 4**, on the course Canvas site, and the Week 7 and Week 8 Asynchronous Materials, along with the posted in-class projections from the public course web site, are useful references for this homework.
- Feel free to add additional `prompt` commands to your SQL scripts as desired to enhance the readability of the resulting output.
- You are expected to follow **course style standards** for SQL `select` statements.

Problem 1

Correctly complete the "HW 6 - Problem 1 - Reading Questions for DB Reading Packet 6 - Intro to Normalization", on the course Canvas site.

Setup for Problem 2

Use `ssh` to connect to `nrs-projects.humboldt.edu`, and create, protect, and go to a directory named `325hw6` on nrs-projects:

```
mkdir 325hw6
chmod 700 325hw6
cd 325hw6
```

Put all of your files for Problem 2 in this directory. (And it is from this directory that you should type `~st10/325submit` each time you would like to submit your files for Problem 6.)

Problem 2

This problem again uses the tables created by the SQL script `movies-create.sql` and populated by `movies-pop.sql`. As a reminder, these tables can be described in relation structure form as:

```
Movie_category(CATEGORY_CODE, category_name)

Client(CLIENT_NUM, client_lname, client_fname, client_phone,
       client_credit_rtg, client_fave_cat)
   foreign key (client_fave_cat) references movie_category(category_code)

Movie(MOVIE_NUM, movie_title, movie_director_lname, movie_yr_released,
      movie_rating, category_code)
   foreign key(category_code) references movie_category

Video(VID_ID, vid_format, vid_purchase_date, vid_rental_price, movie_num)
   foreign key (movie_num) references movie

Rental(RENTAL_NUM, client_num, vid_id, date_out, date_due, date_returned)
   foreign key (client_num) references client,
   foreign key(vid_id) references video
```

And, again, for your convenience as a reference, a handout of these relation structures is posted along with this homework handout.

(These tables should **still exist** in your database from Homework 4, so you should **not** need to re-run `movies-create.sql` or `movies-pop.sql` unless you have been experimenting with insertions or other table modifications.)

Use `nano` (or `vi` or `emacs`) to create a file named `325hw6.sql`:

```
nano 325hw6.sql
```

While within `nano` (or whatever), type in the following within one or more SQL **comments**:

- your name
- `CS 325 - Homework 6 - Problem 2`
- the date this file was last modified

Then:

- use `spool` to start writing the results for this script's actions into a file `325hw6-out.txt`
- put in a `prompt` command printing `Homework 6 Problem 2`
- put in a `prompt` command printing your name
- include a `spool off` command, at the BOTTOM/END of this file. Type your answers to the problems below BEFORE this `spool off` command!

NOTE!!! READ THIS!!!

Now, within your file `325hw6.sql`, add in SQL statements for the following, **PRECEDING** EACH with a SQL*Plus `prompt` command noting what problem part it is for.

Problem 2-1

Using a **nested** `select` statement, **and using NO join or Cartesian product operations**, project just the **video ids** and **video formats** of all videos that have a rental price **less** than the average video rental price.

Problem 2-2

Using a **nested** `select` statement, **and using NO join or Cartesian product operations**, project the last names and first names only (do **NOT** project the date the video was due) for clients who have ever rented the video with ID `'130012'`.

Problem 2-3

Write a `select` statement which projects ONE column in its result: this column, with heading "`Movie: Rating`", shows, for each movie, the title for that movie, then a colon and a space, and then the rating for that movie.

Problem 2-4

Write a `select` statement that projects TWO columns in its result:

- its first column, with heading "`Movies`", shows, for each movie, the title for that movie, a space, and then, within a set of parentheses, the year that movie was released;
- and, its second column, with heading "`Directors`", shows the last name of the director of that movie

Problem 2-5

Write a `select` statement that projects the movie title(s) of the movie(s) whose video(s) have the earliest video purchase date.

Problem 2-6

Consider a row in the `rental` table. If a rental has not yet been returned, its `date_returned` attribute is `null`.

Using `EXISTS`, write a `select` statement that will project the last names, first names, and phone numbers of clients that have rented a video and not returned it yet (those who have **any** unreturned video rental -- we don't care, for this query, whether it happens to be overdue or not). That is, we want this information for clients for which such a rental exists.

(NOTE: This will not be accepted as correct unless it properly uses `EXISTS`.)

Problem 2-7

Using `NOT EXISTS`, write a `select` statement that will project the titles of movies for which there are no videos with the format Blu-Ray. That is, we want this information for movies for which no such video exists.

(NOTE: This will not be accepted as correct unless it properly uses `NOT EXISTS`.)

Problem 2-8

Using `&`, write a `select` statement that will project just the movie title and year released of movies whose director is that of the director last name entered by the user when prompted when this SQL script is run.

When you run `325hw6.sql` one last time before submitting your homework files, enter whatever director last name you like when this query is executed. I happened to enter `Spielberg` during the run that resulted in the posted example `325hw6-out.txt`.

Problem 2-9

Using `&`, AND using a **nested** `select` statement, **and using NO join or Cartesian product operations**, write a `select` statement that will project just the movie title and director last name of movies whose category CODE is that of the category NAME entered by the user when prompted when this SQL script is run.

When you run `325hw6.sql` one last time before submitting your homework files, enter whatever category name you like when this query is executed. I happened to enter `Classic` during the run that resulted in the posted example `325hw6-out.txt`.

Submit your files `325hw6.sql` and `325hw6-out.txt`.