CS 325 - Homework 5

Deadline:

Problem 1 -- answering reading questions on Moodle for Reading Packet 6 -- needs to be completed by 10:45 am on Tuesday, October 18.

The remaining problems are due by **11:59 pm** on **Friday, October 21, 2016**.

How to submit:

For Problem 2 onward:

Each time you wish to submit, within the directory `325hw5` 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 `5`.

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

Purpose:

To write more SQL queries (including queries with nested selects/subselects and with joins of more than 2 tables), and to practice a bit with `&` to allow interactive input into a script.

Additional notes:

- NOTE the following course style standards for SQL `select` statements:
  - In a SQL script, put a blank line BEFORE and AFTER each `select` statement, for better readability.
  - A `select` statement's `from` clause should ALWAYS start on a new line.
  - If a `select` statement has a `where` clause, it should always start on a new line.
  - If a clause is longer than one line, INDENT the continuation on the next by at least three spaces (so it is clear which clause it "belongs" to).
  - Nested selects (sub-selects) should be indented within their outer select, STILL with their `from` and `where` clauses each on their own line -- for example, both of the following meet class style standards:

```
select empl_last_name, salary
from   empl
where  dept_num IN
       (select dept_num
        from   dept
        where  dept_loc = 'Dallas');
```

-- continued on next page

```
    select empl_last_name, salary
    from   empl
    where  dept_num IN (select dept_num
                        from   dept
                        where  dept_loc = 'Dallas');
```

- You are required to use the HSU Oracle `student` database for this homework.
- SQL Reading Packet 4 on the course Moodle site and the Week 8 Lecture 2 posted examples on the public course web site are useful references for this homework.
- An example `hw5-out.txt` has been posted along with this homework handout, to help you see if you are on the right track with your queries for Problem 2. If your `hw5-out.txt` matches this posted one, that doesn't guarantee that you wrote appropriate queries, but it is an encouraging sign.
    - (I added a few extra `prompt` commands near the beginning of this script to output some blank lines for slightly-prettier output.)
- Feel free to add additional `prompt` commands to your SQL scripts as desired to enhance the readability of the resulting spooled output.

Problem 1

Have to correctly answer the Reading Questions for Reading Packet 6 - Normalization, on the course Moodle site, by 10:45 am on Tuesday, October 18.

Setup for Problems 2 onward

Use `ssh` to connect to `nrs-projects.humboldt.edu`, and create a directory named `325hw5` on nrs-projects:

```
mkdir 325hw5
```

...and change this directory's permissions so that only you can read it:

```
chmod 700 325hw5
```

...and change your current directory to that directory (go to that new directory) to do the rest of the problems for this homework:

```
cd 325hw5
```

Put all of your files for this homework in this directory. (And it is from this directory that you should type `~st10/325submit` to submit your files each time you want to submit the work you have done so far.)

This homework again uses the tables created by the SQL script `hw4-create.sql` and populated by `hw4-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 5, so you should **not** need to re-run `hw4-create.sql` or `hw4-pop.sql` unless you have been experimenting with insertions or other table modifications.)

Use `nano` (or `vi` or `emacs`) to create a file named `hw5.sql`:

```
nano hw5.sql
```

While within `nano` (or `vi` or `emacs`), type in the following:

- your name within a SQL comment
- `CS 325 Homework 5` within a SQL comment
- the date this file was last modified within a SQL comment
- use `spool` to start writing the results for this script's actions into a file `hw5-out.txt`
- put in a `prompt` command printing `Homework 5`
- 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 `hw5.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

Write a query that projects just the average video rental price.

Problem 3

Using a **nested** select statement, **and using NO join or Cartesian product operations**, project the **video id's only** of all videos that have a rental price less than the average video rental price.

Problem 4

Using a **nested** select statement, **and using NO join or Cartesian product operations**, project the client numbers **only** for clients involved in rentals of videos that have a rental price less than the average video rental price. Do not worry, in this case, about whether or not there are duplicate rows in the result.

Problem 5

Using a **nested** select statement, **and using NO join or Cartesian product operations**, project the last names and the client credit ratings of clients involved in rentals of videos that have a rental price less than the average video rental price. (If done correctly, there is no way that you can get duplicate rows in this query's results, even without the keyword that prevents them -- do you understand why? You do not have to answer as part of this homework, but you should know...)

Problem 6

Write a query which will project the **average rental price** and **number of such videos** for videos of Classic movies -- BUT! You must write this without explicitly using the category code for Classic movies; you must use the name `'Classic'` in your query instead. Rename the columns to be "Avg Cost - Classic" and "# Videos - Classic", respectively, using precisely these spaces and case.

HINTS:

- in an equi-join of *n* tables, make sure you have *n-1* join conditions
- look carefully at the foreign keys of tables involved to determine what those join conditions should be, looking for columns with common domains where their being equal in the Cartesian product has useful meaning

Problem 7

Using a **join**, **and NOT using ANY nesting or sub-selects**, project the last names, first names, and date the video was due for clients who have ever rented the video with ID `'130012'`.

Problem 8

Using a **nested** select, **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 this time) for clients who have ever rented the video with ID `'130012'`.

Problem 9

Project the last names, favorite movie category **names**, and credit ratings for clients who have credit ratings higher than the average credit rating for all clients. (**NOTE**: I am not asking you to project the `client_fave_cat` --- I am asking you to project the **name** of the category corresponding to the `client_fave_cat`.)

HINTS:

- a single query can definitely use both equi-joins AND sub-selects
- a `select` clause can only project attributes from relation(s) in its corresponding `from` clause

Problem 10

Using `&`, write a query that will project just the movie title 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 `hw5.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 `Lasseter` during the run that resulted in the posted example `hw5-out.txt`.

Problem 11

Using `&`, write a query that will project just the movie title 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 `hw5.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 `hw5-out.txt`.

When you think the results of all of these queries look correct, this would also be a good time to look at the contents of `hw5-out.txt` -- at the nrs-projects prompt (the UNIX level, NOT in `sqlplus`!), type:

```
more hw5-out.txt
```

You should see that `hw5-out.txt` contains the query results you just saw within `sqlplus`.

When you are satisfied with these, then `hw5.sql` and `hw5-out.txt` are completed.