CS 325 - Homework 7

Deadline

11:59 pm on **Friday, October 29, 2021**.

Purpose

To start reading and thinking about converting ER models into database designs, to practice normalizing sets of relations into 1NF, 2NF, and 3NF, and to get more practice writing SQL statements, including statements using SQL `select` statement `order by` clauses, `group by` clauses, and `having` clauses.

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 `325hw7` 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 `7`.

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

Additional notes:

- **Reminder**: CS 325 course style for **relation-structure form** includes:
  - Write all attributes making up a relation's primary key in all-uppercase
  - For foreign keys, list their attributes as usual in the parentheses, but then also write a SQL-style foreign key clause after the closing parenthesis.
  - For example:

```
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
```

- You are required to use the HSU Oracle `student` database for **Problem 4** of this homework.
- **DB Reading Packet 6**, **DB Reading Packet 7** and **SQL Reading Packet 5**, on the course Canvas site, and the Week 9 Asynchronous Materials, along with the posted in-class projections from the public course web site, are useful references for this homework.
- Now that we have covered the `order by` clause, you are expected to use it appropriately when an *explicit* row ordering is specified. Queries for problems asking for *explicit* row ordering will be incorrect if they do not include a reasonable `order by` clause.
- Feel free to add additional `prompt` commands to your SQL scripts as desired to enhance the

readability of the resulting output.

- An example `325hw7-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 4. If your `325hw7-out.txt` matches this posted one, that doesn't guarantee that you wrote appropriate queries, but it is an encouraging sign.
- You are expected to follow **course style standards** for SQL `select` statements.
  - On the CS 325 public course web site, under "References", there are now some evolving lists of course style standards posted. There is also a link to these on the course Canvas home page.

Problem 1

Correctly complete the "HW 7 - Problem 1 - Reading Questions for DB Reading Packet 7 - Database Design, Part 1", on the course Canvas site.

Setup for Problems 2 onward

Use `ssh` to connect to `nrs-projects.humboldt.edu`, and create, protect, and go to a directory named `325hw7` on nrs-projects:

```
mkdir 325hw7
chmod 700 325hw7
cd 325hw7
```

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.)

Problem 2

There is now a posted set of CS 325 SQL style standards, (on the public course web, site near the bottom of the "References" section, and also on the course Canvas home page) This problem is an excuse to get you to look these over and hopefully remind you about these style standards along with their new additions.

In a file `325hw7-style.txt`, include your name, and then give answers for the following, preceding each answer with the number-and-part of the problem being answered.

2 part a

One of the posted SQL style standards specifically applies to `SELECT` statement `ORDER BY` clauses.

Give this style standard.

2 part b

One of the posted SQL style standards specifically applies to `SELECT` statement `GROUP BY` clauses.

Give this style standard.

2 part c

Two of the posted SQL style standards applies to nested `SELECT` statements.

Give either one of these style standards (you only need to give the main THOU SHALT bullet, not any sub-bullets).

2 part d

One of the posted SQL style standards applies specifically to the `EXISTS` and `NOT EXISTS` predicates - but it actually lists two style standards for these, in its two sub-bullets.

Give BOTH of these sub-bullets/style standards.

Submit your file `325hw7-style.txt`.

Problem 3

In a file `325hw7-norm.txt`, include your name, and then give answers for the following, preceding each answer with the number-and-part of the problem being answered.

Remember to indicate foreign keys, if any, using SQL foreign key syntax directly under the relation structure that has that foreign key.

3 part a

Consider this almost-relation, expressed in relation-structure form:

```
Choc_Bar_Type(BAR_ID, bar_brand, bar_name, bar_cacao_pct, bar_size_avlbl)
```

Why is it an *almost*-relation? ...because it turns out that a single chocolate bar type instance *can* have more than one value for the attribute `bar_size_avlbl`.

Convert this into first normal form (1NF), writing the resulting relation(s) in relation-structure form.

3 part b

Consider this relation, expressed in relation-structure form:

```
Stu_Course_Reg(STU_ID, COURSE_SECT_ID, stu_lname, stu_primary_email,
        course_num, course_name, course_num_units, reg_date, final_grade)
```

In addition to the functional dependencies *already* implied above (because this is a relation), the following functional dependencies *also* exist:

```
stu_id -> stu_lname, stu_primary_email

course_sect_id -> course_num, course_name, course_num_units
```

Convert this into second normal form (2NF), writing the resulting relation(s) in relation-structure form.

3 part c

Consider this relation, expressed in relation-structure form:

```
Symphony(SYMPH_ID, symph_title, symph_year_completed, composer_id,
    music_period_id, music_period_name, music_period_year_begins)

    foreign key (composer_id) references Composer
```

In addition to the functional dependencies *already* implied above (because this is a relation), the following functional dependency *also* exists:

```
music_period_id -> music_period_name, music_period_year_begins
```

Convert this into third normal form (3NF), writing the resulting relation(s) in relation-structure form.

Submit your file `325hw7-norm.txt`.

Problem 4

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` unless you have been experimenting with insertions or other table modifications.)

Use `nano` (or `vi` or `emacs`) to create a file named `325hw7.sql`:

```
nano 325hw7.sql
```

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

- your name
- `CS 325 - Homework 7 - Problem 4`
- the date this file was last modified

Then:

- use `spool` to start writing the results for this script's actions into a file `325hw7-out.txt`
- put in a `prompt` command printing `Homework 7 Problem 4`

- 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 `325hw7.sql`, add in SQL statements for the following, **PRECEDING** EACH with a SQL*Plus `prompt` command noting what problem part it is for.

Problem 4-1

Perform a relational selection of the rows of the `client` table, displaying the resulting rows in **increasing** order of client credit rating.

Then, perform another relational selection of the rows of the `client` table, but now displaying the resulting rows in **decreasing** order of client credit rating.

Problem 4-2

On a previous homework (Homework 6 - Problem 2-3), you wrote a query 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.

Now write a version of this query so that, now, its results are ordered by increasing/alphabetical order of movie rating, and for rows with the same movie rating, they should be ordered by increasing/alphabetical order of movie title.

Problem 4-3

Perform a projection of the *name* of a movie's category, the movie title, and the movie rating, for all movies, displaying the resulting rows in order of movie rating, and for movies with the same rating, in *reverse* alphabetical order of move category name, and for movies with the same rating and category name, in order of movie title.

Problem 4-4

Project the client's last name, telephone number, and credit rating for clients whose credit rating is less than or equal to than the average client credit rating, displaying the resulting rows in reverse order of credit rating.

Problem 4-5

From the video table, for each video format, project the video format, the number of videos with that format using the column alias `QTY`, and the average video rental price for videos with that format using the column alias `AVG RENTAL PRICE`. (Do not worry about the ugly formatting of the average video rental price.)

Problem 4-6

From the video table, for each video rental price, project the video rental price, and the number of videos with that rental price using the column alias `QUANTITY`, displaying the resulting rows in decreasing order of video rental price.

Problem 4-7

Rewrite Problem 6-7's query, except this time include **only** those video rental prices that are the prices of at least 5 (5 or more) videos. (That is, project the video rental price and number of videos with that rental price using the column alias `QUANTITY` ONLY for video rental prices that are the prices of 5 or more videos.)

Submit your files `325hw7.sql` and `325hw7-out.txt`.