CS 325 - Homework 5

Deadline

11:59 pm on **Friday, October 8, 2021**.

Purpose

To read and think some more about, and practice some more, database modeling (including involving supertype/subtype entity classes), and to write more SQL queries, including queries with column aliases, computed columns, table aliases, and aggregate functions.

How to submit

Problem 1 will be completed on the course Canvas site.

Problem 2 will be submitted on nrs-projects.

The E-R diagrams for Problems 3 and 4 will each be submitted to Canvas.

For Problem 2:

Each time you wish to submit Problem 2, 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!)

Additional notes:

- You are required to use the HSU Oracle `student` database for **Problem 2** of this homework.
- **DB Reading Packet 5** and **SQL Reading Packet 3**, on the course Canvas site, 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 entity-relationship diagrams and SQL `select` statements.

Problem 1

Correctly complete the "HW 5 - Problem 1 - Reading Questions for DB Reading Packet 5 - Entity-Relationship Modeling, Part 2", 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 `325hw5` on nrs-projects:

```
mkdir 325hw5
chmod 700 325hw5
cd 325hw5
```

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 2.)

Problem 2

YOU ARE USING ORACLE and SQL FOR THIS PROBLEM.

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 `325hw5.sql`:

```
nano 325hw5.sql
```

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

- your name
- `CS 325 - Homework 5 - 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 `325hw5-out.txt`
- put in a `prompt` command printing `Homework 5 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 `325hw5.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

Write a `select` statement that projects, just for clients whose credit rating is less than or equal to 4.5, just the client last name and what their credit rating would be if it were increased by 0.5, giving this second projected column a column heading of `IF_BUMPED_UP`.

Problem 2-2

Write a `select` statement that projects, just for rentals that have not yet been returned (so, whose `date_returned` is `null`), just the rental number, the `date_due`, and what the `date_due` would be if it were extended by 5 days, giving this second projected column a column heading of `IF_GRACE`

Problem 2-3

Write a `select` statement that projects, just for videos whose format is *not* Blu-Ray, just the video ID, the video format, and what the rental price would be if it were decreased by 20%, giving this third projected column a column heading of `IF_DISCT`. (Do not worry about the number of decimal places in the result!)

Problem 2-4

Write a `select` statement that projects, based on an equi-join of `movie` and `movie_category`, just the movie title, the category code, and the category name for that category code. For full credit, use appropriate table aliases in this `select` statement.

Problem 2-5

Write a `select` statement that projects just the earliest video purchase date for any video and the latest video purchase date for any video, with column headings of `OLDEST` and `NEWEST`.

Problem 2-6

Write a `select` statement that projects the current total number of videos and the average video rental price for videos, with column headings of "`Total Videos`" and "`Avg Rental`" (with the blanks and mixed case as shown). (Do not worry about the many decimal places in the result!)

Problem 2-7

Write a `select` statement that projects, based on an equi-join of `video` and `rental`, the total number of rentals and the sum of the video rental prices for all rentals, with column headings of "`# Rentals`"

and "`Total Rental $`" (with the blanks and mixed case as shown). For full credit, use appropriate table aliases in this `select` statement.

Submit your files `325hw5.sql` and `325hw5-out.txt`.

Problems 3 and 4 - NOTE! These do NOT use Oracle or SQL!

For these problems, you will be creating entity-relationship diagrams including entity attribute lists for two scenarios.

Be sure to meet the class style standards for ERDs, as shown in the class reading packets.

How are you going to create these? You have several choices:

- you may use Word's or OpenOffice's or NeoOffice's or LibreOffice's drawing tools to draw them,
- you can use other drawing software if you have it,
- you can draw them by hand,
- you can produce part of them using software, and then finish them by hand; etc.

Then, you need to convert your result into PDF format using some means, into files named `325hw5-erd-1.pdf` and `325hw5-erd-2.pdf`.

- you might be using software with an option to save or export them as PDF,
- you can scan them and save them as PDF,
- you can take a (legible!) photo of them and save them as or convert them to PDF, etc.

Including a set of **business rules** is **NOT required** for these problems, unless you would like to to justify/explain some of your choices.

- Note that you may **not** change anything in either scenario, however.

NOTE -- because these files `325hw5-erd-1.pdf` and `325hw5-erd-2.pdf` are being created on a computer other than nrs-projects, you will submit them to Canvas.

Problem 3

Model the following, creating a file `325hw5-erd-1.pdf` with its entity-relationship diagram including entity attribute lists.

Consider the following. In a particular on-line financial institution, clients have a unique client number, and the institution keeps their one preferred e-mail address, and whichever operating systems (yes, there might be more than one) that they like to use to access their account(s). Clients may be individuals or corporations -- individual clients also need their last name, first name, and social security number on-record, while corporate clients need their one preferred doing-business-as (DBA) name and their tax ID number on-record.

A client can have 1 or more accounts at this institution; they must have at least one account. Accounts are related to at least one client, but may be related to more than one client as well (that is, joint or multiple-owner accounts are possible); each account has a unique account number, the date it was opened, its current balance, and its current interest rate. An account is identified by its account number. Finally, transactions are made on accounts, and the institution cares about the following information from each

transaction: the transaction number, the date of the transaction, and the amount of the transaction (which is a single monetary amount, and which may be positive or negative). Each transaction is uniquely identified by a transaction number, and each transaction is only on a single account. An account may have many transactions, but it does not have to have any transactions.

Problem 4

Model the following, creating a file `325hw5-erd-2.pdf` with its entity-relationship diagram including entity attribute lists.

A local high-school has an active and enthusiastic booster society. Its purpose is to encourage interest in, as well as donations to, the school. For each donation, they keep track of a donation's unique donation number, as well as the date and the amount of the donation. Not all donations are from booster society members, but if one is, they want to know which single booster society member is to be credited for that donation. They also keep track of each booster society members' unique booster number, their last name, their first name, and a single preferred contact e-mail address. If the booster is a parent/guardian of a current student at the school, they also want to keep track of their volunteer hours and their single preferred contact phone number. If the booster is an alumni of the school, they keep track of the year they left the school (that is, the year they graduated or transferred). And, if the booster is a friend of the school, they keep track of the year they joined the booster society. Note that some boosters may be alumni and parents, or might have started as a friend of the school and later became a parent, or vice-versa.