CIS 315 - Homework 3

Deadline:

1:00 pm (beginning of lab) on Wednesday, September 29th

How to submit:

When you are ready, within the directory `315hw3` on `nrs-labs.humboldt.edu` (and at the nrs-labs UNIX prompt, NOT inside `sqlplus`!) type:

```
~st10/315submit
```

...to submit these `.sql`, `.txt`, and `.pdf` files, using a homework number of `3`. (**Make sure** that you see the names of all of the files you wished to submit!)

Purpose:

To practice with basic SQL select statements, and to practice with database modeling (creating an ERD meeting the required notation for this course).

Additional notes:

- You are required to use the HSU Oracle `student` database for Problem 2.
- An example `hw3-results.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 `hw3-results.txt` matches this posted one, that doesn't guarantee that you wrote appropriate queries, but it is an encouraging sign... (note that your results for Problem 2-14 are very likely to be different, given the interactive nature of that query.)

Problem 1:

NOTE: THIS PROBLEM DOES NOT USE ORACLE AT ALL!!

For this part, you will be creating an Entity-Relationship Diagram (ERD) for a scenario.

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

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

Then, you need to convert your result into PDF format using some means, into a file named `hw3erd.pdf`; see the handout "Saving a file in PDF format", posted along with this homework handout, for a description of some of the ways in which you can convert something into PDF format.

You will submit your resulting `hw3erd.pdf` for this part. (Make sure to put it/them in the same directory as your `.sql` and `.txt` files on `nrs-labs.humboldt.edu`)

The Scenario:

Consider the following. In a particular club, would-be members are asked to give their last name and all of their significant e-mail addresses. When accepted as a member, the new member is given a unique membership number, and the date that the member joined the club is recorded, along with their last name and those significant e-mail addresses.

This club offers seminars frequently. To keep track of them, the club gives each seminar a title, a unique seminar number, its date of occurrence, the time that it begins on that date, and the time that it ends on that date. (No seminar lasts more than a day.)

Members may sign up to attend one or more seminars, and do not have to sign up for any. Members may also be in charge of one or more seminars, but do not have to be in charge of any. A seminar must have precisely one member in charge of that seminar, and at least one member must agree to sign up to attend that seminar before it is approved for offering; many members may sign up for each seminar, of course.

Finally, this club has assigned parking spaces in several nearby parking garages. The club has assigned its own unique parking space ID number for each such space; it also keeps track of the name of the garage that that space is in, what section number it is in within that garage, and what space number it is marked with within that section.

A member may be assigned one of these parking spaces, but does not have to be assigned one. No member may have more than one assigned parking space, and a parking space may not be assigned to more than one member at a time (and, of course, some parking spaces may not be assigned to anyone at any given time).

Your Task:

Develop and draw an appropriate ERD for the above scenario. Create a PDF of your final ERD, save a copy of that `.pdf` file to `nrs-labs.humboldt.edu`, and submit it using `~st10/315submit` from there.

Be sure to meet the following style standards:

- each **entity class** is represented by a **rectangle**, with the name of the entity class within the rectangle. There should be ONLY 1 rectangle per entity class! (You can have multiple relationship class lines connected to a single entity class rectangle.)
- each **relationship class** is represented by a diamond on a line connecting the associated entity classes, labeled on or above or below the diamond with a descriptive (and preferably unique) name for that relationship class.
  - (that is, you draw a line between related entity classes, and the diamond for the relationship class appears on that line…)
- the **maximum** cardinality of each relationship class must be depicted by writing the 1, M, or N, whichever is appropriate, near each end of the relationship line near the appropriate entity class rectangle (as depicted in the Week 5 Lecture reading packet, the PDF "slightly different version of

Week 5 lecture material").

- the **minimum** cardinality of each relationship class must be depicted by drawing either an oval or a line, whichever is appropriate, on each end of the relationship line (near the entity class rectangle) (as depicted in the Week 5 Lecture reading packet, the PDF "slightly different version of Week 5 lecture material"). (You draw an oval on the relationship line near the entity class rectangle if the relationship is optional at that end, and you draw a line across the relationship line near the entity class rectangle if the relationship is mandatory on that end.)
- under the finished ERD, or on the next page, should be listed the name of each entity class, and underneath it:
  - the attributes for that entity class
  - for any attribute that can be multi-valued, write `(MV)` after that attribute
  - underline **or** write in all capital letters the attribute(s), if any, that users in the scenario use to identify/distinguish between different instances of that entity (keeping in mind that not all entity classes have identifying attributes)

  Do NOT write these lists of entity class attributes as relation structures -- these are **not** relations yet! They are **entities**, and each entity will eventually be transformed into **one or more** relations during the database design phase, which FOLLOWS the modeling phase.

  Also remember: in the modeling phase, the attributes in these lists are attributes of that entity alone -- they do NOT indicate relationships between entities. The relationship lines in the ERD do that! A useful rule of thumb: each attribute should only appear once, TOTAL, in this list of entities' attributes.
- Including a set of **business rules** is not required for this homework problem, unless you would like to to justify/explain some of your choices (for example, for whether attributes can be multi-valued). Note that you may **not** change anything in the scenario, however.

For example, the following meets the above standards:


```
Professor                    Course-section
-----------------            ------------------
PROF_SSN                     Course_name
Prof_lname                   COURSE_ID
Prof_email (MV)              Course_length
```

In the above example, given the maximum and minimum cardinalities shown, a professor does not HAVE to teach any course-sections, but may teach many course-sections. A course-section MUST have an associated professor, and may have at most one professor associated with it --- that is, there is exactly one professor associated with each course-session. Note that the Professor entity attributes do not include any indication of which courses that professor teaches, nor do the Course-section entity attributes include any indication of which professor teaches that course-section --- this is NOT an error.

This is proper for the database modeling phase --- the relationship between professors and courses is indicated at this phase by the line between their rectangles and by the minimum and maximum cardinalities shown on that line.

Problem 2:

On the public course web page, along with this assignment, you will find a SQL script `hw3-setup.sql`. It creates and populates a collection of tables that 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

Use `ssh` to connect to `nrs-labs.humboldt.edu`, and create a directory named `315hw3` on nrs-labs:

```
mkdir 315hw3
```

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

```
chmod 700 315hw3
```

...and change your current directory to that directory (go to that new directory) to do this homework:

```
cd 315hw3
```

Put all of your files for this homework in this directory. (And it is from this directory that you should type `~st10/315submit` to submit your files when you are done.)

Create a copy of the script `hw3-setup.sql`, either by pasting it from the public course web page, or by using this command at the nrs-labs UNIX prompt while in the directory `315hw3`:

```
cp  ~st10/hw3-setup.sql  hw3-setup.sql
```

Run this SQL script in `sqlplus`; these 5 tables should be created and populated.

Then, use `nano` (or `vi` or `emacs`) to create a file named `hw3-2.sql`:

```
nano hw3-2.sql
```

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

- your name within a SQL comment

- `315 Homework 3-2` as a SQL comment
- the date this file was last modified as a SQL comment
- use `spool` to start writing the results for this script's actions into a file `hw3-results.txt`
- include a `spool off` command, at the BOTTOM/END of this file. Type your answers to the problems below BEFORE this `spool off` command!

Now add in SQL statements for the following, **preceding** each with a SQL comment noting what question it is for.

Problem 2-1

Perform a **relational selection** of rows of the `client` table for clients with a client credit rating less than 3.4.

Problem 2-2

Perform a **"pure" relational projection** of the movie rating and year released columns only from the `movie` table.

Problem 2-3

Perform a **Cartesian product** of the `client` and `movie_category` tables.

Problem 2-4

Perform an **equi-join** of the `client` and `movie_category` tables. (Consider: what would be a suitable join condition for such an equi-join?)

Problem 2-5

Project just the video ID's and their date due for rentals that have not yet been returned. (Note that a rental's date returned attribute is empty/null until it has been returned.)

Problem 2-6

Project just the video ID's, formats, and rental prices for videos that do not have the format DVD.

Problem 2-7

Project just the movie category NAME (not code!), client's last name, and client's credit rating from the equi-join of the `movie_category` and `client` tables.

Problem 2-8

Perform a relational selection of videos with a purchase date between July 15, 2002 and December 1, 2005 (inclusive). For full credit, appropriately use the `between` operation in your query.

Problem 2-9

Perform a relational selection of videos which have a rental price greater than or equal to $3.99 and have a purchase date on or after January 1, 2005.

Problem 2-10

Project only the movie title and the movie rating for all movies containing the string `'the'` anywhere in their title (where the case IS significant -- only all-lowercase `'the'` is desired). For full credit, appropriately use the `like` operation in your query.

Problem 2-11

Project only the video ID, format and then what the rental price would be if it were decreased by 20%, for videos whose format is not DVD. (That is, you will have three columns in your result --- yes, you should project a computed column for that 3rd column. You are not actually changing the video table, you are simply seeing what the rental prices would look like IF such a change were made.)

Problem 2-12

Project only the movie title and movie rating for all movies with ratings of PG-13 or R or A only. For full credit, appropriately use the `in` operation in your query.

Problem 2-13

Project the current total number of videos and the average rental price for a video. (Do not worry about the many decimal places in the result...) Rename the column heading so that they appear as "`Num Vids`" and "`Avg Rental`" (with the blanks and mixed case as shown).

Problem 2-14

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 `hw3-2.sql` one last time before submitting your homework files, enter whatever category name you like when this query is executed.

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 `hw3-results.txt` -- at the nrs-labs prompt (the UNIX level, NOT in `sqlplus`!), type:

```
more hw3-results.txt
```

You should see that `hw3-results.txt` contains the query results you just saw within `sqlplus`.

When you are satisfied with these, then `hw3-2.sql` and `hw3-results.txt` are ready to submit.