CS 325 - Homework 1

Deadline:

Problem 1 -- answering the CS 325 Required Syllabus Confirmation and Reading Questions on Moodle -- had to be completed by 11:59 pm on Friday, August 26.

Problem 2 -- answering reading questions on Moodle for Reading Packet 1 -- had to be completed by 10:45 am on Tuesday, August 30.

Problem 3 -- answering reading questions on Moodle for Reading Packet 2 -- needs to be completed by 10:45 am on Tuesday, September 6.

The remaining problems are due by **11:59 pm** on **Friday, September 16, 2016**.

How to submit:

For Problem 4 onward:

Each time you wish to submit, within the directory `325hw1` 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 `1`.

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

Purpose:

To see if you are gleaning some important concepts from required class reading, and to practice a little with writing and running SQL scripts, creating (and dropping) a table, inserting rows into a table, and submitting files using the course submit tool.

Additional notes:

- You are required to use the HSU Oracle `student` database for this homework.
- SQL Reading Packet 1, on the course Moodle site, is a useful reference for this homework.
- For convenience, in a SQL script, sometimes one chooses to precede a `create table` command with a corresponding `drop table` command, so that when you re-run the script, the script drops and re-creates the table.
  - We'll often do this in this course -- in a production system one might be more wary of this! BUT if you are really re-creating a table, it has to be dropped first, anyway.
  - The `drop table` command will fail the very first time you run such a script, since the table is not there yet to be dropped. That's fine -- the subsequent `create table` command will still work, and you can always re-run the script a second time to ensure that the error message indeed goes away.

Problem 1

Had to correctly answer the CS 325 Required Syllabus Confirmation and Reading Questions, on the course Moodle site, by 11:59 pm on Friday, August 26.

Problem 2

Had to correctly answer the Reading Questions for Reading Packet 1, on the course Moodle site, by 10:45 am on Tuesday, August 30.

Problem 3

Had to correctly answer the Reading Questions for Reading Packet 2, on the course Moodle site, by 10:45 am on Tuesday, September 6.

Setup for Problems 4 onward

Use `ssh` to connect to `nrs-projects.humboldt.edu`, and create a directory named `325hw1` on nrs-projects:

```
mkdir 325hw1
```

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

```
chmod 700 325hw1
```

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

```
cd 325hw1
```

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 when you are done.)

Problem 4

Use `nano` (or `vi` or `emacs`) to create a file named `hw1-1.sql` within directory `325hw1`:

```
nano hw1-1.sql
```

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

- your name within a SQL comment
- `325 HW 1 - part 1` within a SQL comment
- the date this file was last modified within a SQL comment

This script is still not yet complete -- you have more to add to it.

Problem 5

Think of a table on some topic of your choice that you would like to create, and then add commands for the following to your SQL script `hw1-1.sql`:

- write a SQL `drop table` statement for that table
- write a SQL `create table` statement for that table, such that:
  - your new table has at least 3 columns
  - your table declares a primary key: one or more columns whose value must be unique for each row of your table
- insert at least 4 rows into your new table

IF you haven't already, this would be a good time to save your `hw1-1.sql` file, and go into `sqlplus` and see if:

```
start hw1-1.sql
```

...works -- is your new table dropped and created? (Remember that the `drop table` command will fail until you actually manage to create that table for the first time, since until then there is nothing to drop.)

This file `hw1-1.sql` is now complete.

Problem 6

You may have noticed that we didn't use `spool` in `hw1-1.sql`! That's because I want to make a point about how tables **persist** in a database --- once created, they STAY until they are dropped! We're going to display your table's contents using a separate SQL script, to show that we *can*.

Use `nano` (or `vi` or `emacs`) to create a file named `hw1-2.sql`:

```
nano hw1-2.sql
```

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

- your name within a SQL comment
- `325 HW 1 - part 2` 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 `hw1-out-1.txt`
- add a SQL `select` statement for the table you created in your `hw1-1.sql`, to show its contents
- then include a `spool off` command, at the BOTTOM/END of this file.

IF you haven't already, this would be a good time to save your `hw1-2.sql` file, and go into `sqlplus` and see if:

```
start hw1-2.sql
```

...works -- do you see the at-least-4-rows of your table?

(Do not worry about "ugliness" like chopped-off column headings, or too-long rows that wrap to the next line, or how values are formatted - we'll discuss how to change how these display later.)

This would also be a good time to look at the contents of `hw1-out-1.txt` --- at the nrs-projects prompt (the UNIX level, **NOT** in `sqlplus`!), type:

`more hw1-out-1.txt`

You should see that `hw1-out-1.txt` shows the file contents you just saw within `sqlplus`.

When you are satisfied with these, then `hw1-2.sql` and `hw1-out-1.txt` are complete.

Problem 7

Use `nano` (or `vi` or `emacs`) to create a file named `hw1-3.txt` within directory `325hw1`:

`nano hw1-3.txt`

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

- your name
- `CS 325 Homework 1-3`
- the date this file was last modified

Consider the following relations, written in **tabular form**. Also, note that, in this bizarre scenario, a client is only allowed to rent a particular video one time, ever. (That's an example of a **business rule**, by the way!)

the Client relation:

Cli_id	Cli_lname	Cli_fname	Cli_phone
000A	Alpha	Ann	000-0001
111B	Beta	Bob	111-1112
222B	Beta	Ann	222-2223
333C	Carlos	David	333-3334
444D	Delta	Edie	111-1112

the Video relation:

Vid_id	Vid_format	Vid_purchase_date	Vid_rental_price	Vid_length
00000D	DVD	11-JAN-2013	1.99	73
11111H	HD-DVD	22-FEB-2014	4.99	91
22222B	BluRay	03-MAR-2012	1.99	105
33333H	HD-DVD	22-FEB-2014	3.99	69
44444B	BluRay	04-APR-2010	0.99	91

the Rental relation:

Cli_id	Vid_id
111B	11111H

Cli_id	Vid_id
222B	00000D
222B	22222B
333C	22222B
333C	00000D
333C	11111H
000A	44444B

In `hw1-3.txt`, write these tables in **relation structure** form, writing the primary key attributes in all-uppercase. (Part of your task here is to determine what would be a **reasonable** primary key for each of these tables.)

(Remember: the primary key must **uniquely** identify a row, and it can consist of **one or more** attributes.)

Example relation structure:

```
Parts(PART_NUM, part_name, quant_on_hand, price, level_code,
      last_inspected)
```

...for table `Parts`,

...with attributes `part_num`, `part_name`, `quant_on_hand`, `price`, `level_code`, and `last_inspected`,

...with primary key consisting of attribute `part_num`, because `part_num` uniquely determines a row (every part has a unique part number).)

Do any of these tables have foreign keys (primary keys from other relations added to indicate relationships between this relation and other relations)? If so, indicate that in your relation structure for that relation by **following** the relation structure with a SQL `foreign key` clause for each foreign key it contains -- for example,

```
Empl(EMPL_NUM, empl_last_name, job_title, mgr, hiredate, salary,
     commission, dept_num)
foreign key (dept_num) references dept,
foreign key (mgr) references empl(empl_num)
```

...for table `Empl`,

...with attributes `empl_num`, `empl_last_name`, `job_title`, `mgr`, `hiredate`, `salary`, `commission`, and `dept_num`,

...with primary key consisting of attribute `empl_num`, because `empl_num` uniquely determines a row (every employee has a unique employee number),

...with a foreign key `dept_num` that references table `Dept` (whose primary key is `dept_num`),

...and also with another foreign key `mgr` that references table `Empl`'s own primary key `empl_num` (yes, a table can have a foreign key referencing itself...!).

Your file `hw1-3.txt` is now completed.

Problem 8

Use `nano` (or `vi` or `emacs`) to create a file named `hw1-4.sql`:

```
nano hw1-4.sql
```

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

- your name within a SQL comment
- `CS 325 Homework 1-4` within a SQL comment
- the date this file was last modified within a SQL comment

Now, within your file `hw1-4.sql`:

- write SQL `drop table` statements and `create table` statements for `Client`, `Video`, and `Rental`, being sure to:
  - include `cascade constraints` in these `drop table` statements
  - include all of the columns shown
  - give each column a reasonable, appropriate type, following the class style standards, noting that, in this scenario:
    * client ids are always intended to be exactly 4 characters long, and
    * video ids are always intended to be exactly 6 characters long
  - explicitly set an appropriate primary key for each table (note that you may NOT add additional columns to any of these tables)
  - note that you may **not** use types `char` or `varchar2` for `Video`'s `vid_purchase_date` nor `vid_rental_price` columns -- choose more appropriate types instead!
  - remember which type is more appropriate when a character string column's contents are of **different** lengths, and which is more appropriate when a character string column's contents are **always** the same length

This would be a good time to save your `hw1-4.sql` file, and go into `sqlplus` and see if:

```
start hw1-4.sql
```

...works -- are the 3 tables dropped and created? (Remember that the `drop table` commands will fail until you actually manage to create these tables for the first time, since until then there is nothing to drop.)

Note that this script is not yet complete -- you have more to add to it.

Problem 9

In `hw1-4.sql`, now add SQL `insert` statements to insert the rows shown in Problem 7 into these tables.

This would be a good time to save your `hw1-4.sql` file, and go into `sqlplus` and see if:

```
start hw1-4.sql
```

...still works -- are rows added to the 3 tables?

This script is still not yet complete -- you have more, still, to add to it.

Problem 10

In `hw1-4.sql`, now add SQL `insert` statements to insert:

- one additional row of your own design/choice into the `Client` table
- one additional row of your own design/choice into the `Video` table
- one additional row having your new client renting your new video into the `Rental` table

This would be a good time to save your `hw1-4.sql` file, and go into `sqlplus` and see if:

```
start hw1-4.sql
```

...still works -- are these new rows also added to the 3 tables?

This file `hw1-4.sql` is now complete.

Problem 11

We didn't use `spool` in `hw1-4.sql` to again underscore the point that once tables are created within a database, they **persist**, staying around until they are dropped -- and other SQL scripts can make use of them.

Use `nano` (or `vi` or `emacs`) to create a file named `hw1-5.sql`:

```
nano hw1-5.sql
```

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

- your name within a SQL comment
- `325 Homework 1-5` within a SQL comment
- the date this file was last modified within a SQL comment

Now, within your file `hw1-5.sql`:

- start spooling to a file `hw1-out-2.txt`
  - also put a `spool off` command at the BOTTOM/END of this file. Type your answers to the remainder of the parts below BEFORE this `spool off` command!
- put a `prompt` command that includes YOUR name
- put a `prompt` command that indicates that what follows are the current contents of the `client` table, and then put a `select` statement showing `client`'s contents
- put a `prompt` command that indicates that what follows are the current contents of the `video` table, and then put a `select` statement showing `video`'s contents

- put a `prompt` command that indicates that what follows are the current contents of the `rental` table, and then put a `select` statement showing `rental`'s contents
- double-check that your `spool off` command is AFTER these `prompt` and `select` statements!

(Do not worry about "ugliness" like chopped-off column headings, or too-long rows that wrap to the next line, repeated column headings, or how values are formatted - we'll discuss how to change how these display later.)

This would be a good time to save your `hw1-5.sql` file, and go into `sqlplus` and see if:

```
start hw1-5.sql
```

...works -- do you see the contents of these 3 tables?

This would also be a good time to look at the contents of `hw1-out-2.txt` --- at the nrs-projects prompt (the UNIX level, NOT in `sqlplus`!), type:

```
more hw1-out-2.txt
```

You should see that `hw1-out-2.txt` shows the file contents you just saw within `sqlplus`.

When you are satisfied with these, then `hw1-5.sql` and `hw1-out-2.txt` are completed.