CS 325 - Homework 2

Deadline

11:59 pm on **Friday, September 10, 2021**.

Purpose

To see if you are gleaning some important concepts from required class reading, and to get more practice writing and running SQL scripts, creating (and dropping) tables, and inserting rows into tables.

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 `325hw2` 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 `2`.

(**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 this homework.
- **SQL Reading Packet 1**, on the course Moodle site, is a useful reference for this homework.
- Feel free to add additional `prompt` commands to your SQL scripts as desired to enhance the readability of the resulting output.

Problem 1

Problem 1 is correctly answering the "HW 2 - Problem 1 - Reading Questions for DB Reading Packet 2 - More Database Fundamentals" on the course Canvas site.

Setup for Problems 2 onward

Use `ssh` to connect to `nrs-projects.humboldt.edu`, and create a directory named `325hw2` on nrs-projects:

```
mkdir 325hw2
```

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

```
chmod 700 325hw2
```

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

`cd 325hw2`

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 would like to submit your work-so-far.)

Problem 2

Use `nano` (or `vi` or `emacs`) to create a file named `325hw2-create.sql`:

`nano 325hw2-create.sql`

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

- your name
- `CS 325 Homework 2 - create tables`
- the date this file was last modified

Now, within your file `325hw2-create.sql`:

Consider the following tables/relations, each written in **tabular form** and including additional requirements/constraints.

the Client table/relation:

- NOTE: client ids are always intended to be **exactly** 4 characters long

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 table/relation:

- NOTE: video ids are always intended to be **exactly** 6 characters long

Vid_id	Vid_format	Vid_purchase_date	Vid_rental_price	Vid_length
00000D	DVD	10-JAN-2020	1.99	73
11111H	HD-DVD	20-FEB-2021	4.99	91
22222B	BluRay	30-MAR-2019	1.99	105
33333H	HD-DVD	20-FEB-2021	3.99	69
44444B	BluRay	04-APR-2017	0.99	91

the Rental table/relation:

- NOTE: 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!)
  - (That is, a client can rent different videos over time, but they can only rent a particular video at most once. And a video can be rented by more than one client over time, but it can only be rented by a particular client at most once.)

Cli_id	Vid_id
111B	11111H
222B	00000D
222B	22222B
333C	22222B
333C	00000D
333C	11111H
000A	44444B

Keeping the above in mind:

- write at least one `prompt` command saying that what follows is for `Problem 2`, and that you are creating tables
  - (even though there are no `spool` commands in *this* SQL script, the `prompt` commands will still make your within-`sqlplus` output more readable)
- 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, and satisfy all of the requirements given
  - give each column a reasonable, appropriate type, following the class style standards and the stated requirements given above
    - 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
    - note that you may **NOT** use types `char` or `varchar2` for the video purchase date nor the video rental price -- choose more appropriate types instead!
  - explicitly set an appropriate **primary key** for each table (note that you may **NOT** add additional columns to any of these tables)
  - be sure to declare **foreign keys** as appropriate

Make sure that, when run in `sqlplus`:

```
start 325hw2-create.sql
```

successfully drops and creates these three tables (although 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.)

Submit your script `325hw2-create.sql`.

Problem 3

Use `nano` (or `vi` or `emacs`) to create a file named `325hw2-pop.sql`:

```
nano 325hw2-pop.sql
```

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

- your name
- `CS 325 Homework 2 - populate tables`
- the date this file was last modified

Now -- for this homework, we are **separating** table creation and initial table population into separate SQL scripts, because that can be a useful practice.

BUT, for this to work smoothly during re-population (in the case of debugging, or future modifications, etc.), the population script needs to delete the tables' current rows when it is re-run, to avoid trying to insert the same rows more than once. Which would be fine, except we have not yet discussed the SQL `delete` command!

SO: you are being provided with these commands, this time!

After your opening comment(s) in `325hw2-pop.sql`, add the following comment and `delete` commands:

```
/*========
    in case this is re-run (to get a "fresh" set of initial
    contents), delete any current contents
========*/

delete from rental;
delete from video;
delete from client;
```

Follow the above with at least one `prompt` command noting that what follows is for `Problem 3`, and that you are inserting rows.

Then, add SQL `insert` statements to insert the rows shown in Problem 2's tables into your versions of these tables.

Make sure that, when run in `sqlplus`:

```
start 325hw2-pop.sql
```

successfully adds the rows shown in Problem 2's tables to these three tables.

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

Problem 4

In `325hw2-pop.sql`, now add at least one `prompt` command noting that what follows is for `Problem 4`, and that you are inserting YOUR additional rows.

Then, add SQL `insert` statements to insert:

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

Make sure that, when run in `sqlplus`:

```
start 325hw2-pop.sql
```

successfully adds the rows shown in Problem 2's tables AND your additional rows to these three tables.

Submit your script `325hw2-pop.sql`.

Problem 5

We didn't use `spool` in `325hw2-create.sql` or `325hw2-pop.sql` to again underscore the point that once tables (and their rows) 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 `325hw2-use.sql`:

```
nano 325hw2-use.sql
```

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

- your name
- `CS 325 Homework 2 - use tables`
- the date this file was last modified

Now, within your file `325hw2-use.sql`:

- start spooling to a file `325hw2-out.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 state of the `client` table,
  - and then put a `describe` command showing `client`'s structure,
  - and then put a `select` statement showing `client`'s contents
- put a `prompt` command that indicates that what follows are the current state of the `video` table,
  - and then put a `describe` command showing `video`'s structure,

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

Make sure that, when run in `sqlplus`:

```
start 325hw2-use.sql
```

successfully displays the structure and contents of these three tables.

Also make sure that, when run at the `nrs-projects` prompt (that is, at the UNIX level, NOT in `sqlplus`!):

```
more 325hw2-out.txt
```

has as its contents the output from running the script `325hw2-use.sql` that you just saw within `sqlplus`.

Submit your files `325hw2-use.sql` and `325hw2-out.txt`.