CS 155A | Spring 2014 Assignment 13

Due Date: Tuesday , April 22, 2014 11:00 PM
Points: 35 points max
Turn In: The script and spool files turned in via the assignment drop box

General Directions

Use the a_testbed database. These tasks focus on the use of data modification techniques. This can cause frustrations while you are working out the queries. Suppose I asked you to insert a new row for an animal and we have a PK on the animal table. If you run your script, it will insert that new row- but the second time you run the script the insert will fail due to a pk violation. For these queries to work properly, you **will need to rerun the supplied scripts that populate the tables** as you work and as part of the script.(see task 01)

1. You are not allowed to run alter table queries or change the change the structure or contents of the provided tables in any way. Run the create table statements once only before you start working on the assignment.
2. Since this assignment will ask you to change data in tables, we are using a new set of tables. This way you can make changes in these tables without affecting future assignments. The assignment falls into two parts Tasks 2- 3 use one set of tables and tasks 4 - 6 use a different set of tables. I have split the create table and inserts statements into two sets of scripts. That sql is posted. Open these scripts and read the sql; you should be able to understand those scripts.
3. At various points, the task tells you to include select statements to display table data. Do **not** include other table displays in your final script.
4. **Read the directions carefully**; if a task says that a task uses three sql statements, then you need to use exactly three sql statement. If the task says that you use a single sql statement to create and populate a table, then you must use exactly one sql statement for that.

Tasks

Some of these tasks may require more than one sql statement.

Task 01: For this assignment in copy and paste in the sql for A13_part1_populate.sql and A13_part2_populate.sql. Follow with these select statements.

That will make this task rather long but this assignment is difficult to grade if you do not have the correct starting data. **Failure to do this correct will cost 15 points (That is a lot of the assignment points- this is important.)**

```
select * from upd_animals;
select * from upd_exam_details;
select * from upd_exam_headers;
select * from upd_services;
```

Task 02: **This task uses two sql statements:**
An animal is considered to be non-current if it has not had an exam in the past year and a half. Update the animals table to set the status to NC for animals with no exams in the past year and a half. Then do

```
select * from upd_animals order by an_status, an_id;
```

this
/* For this task you need to understand how to use the temporal functions to handle the concept of the past year and a half. You need to use the system date for this calculation.

You also need to know how to do an exclusion query. You want to work with animals that do not have some characteristic. This means you need to understand the difference between a NOT IN subquery and a <> test. Examine your output carefully.*/

Task 03: **This task uses three sql statements:**
In this task you are going to create a table. Since you cannot create a table if it already exists, drop the table upd_nc_felines; use the same syntax as in the scripts.(This is the first sql command.)
Create a table named upd_nc_felines and copy the following data for cats with a status of NC from the current set of data to the upd_nc_felines table. This does not remove any rows from the upd_animals table. The upd_nc_felines table should include
the animal ID,
the animal name,
the date of their most recent exam if they have any exams, and
the total of all of their exam fees.
The last two columns will be null if the animal has no exams.
There is one row per animal in this table.(This is the second sql command.)
Display the table upd_nc_felines; order by the animal id(This is the third sql command.).

This task uses three sql commands:
1. **a single sql statement** to drop the table that you are create
2. **a single sql statement** to create and populate the table upd_nc_felines
3. **a single sql statement** to display the table as indicated above

The next queries do replace actions. They are each changing the upd_services table. Each of these tasks will consist of:

- (1) A single replace statement .
- (2) A select statement that displays the upd_services table after the merge. Display all columns and sort by the srv_id.

Sometime people add additional data to the tables for testing. That is ok because your final script will reset these tables. These tables (services and changes) do not have any relationships set to other tables.
The tables used for the merge are:
upd_services,
upd_services_changes_1
upd_services_changes_2
upd_services_changes_3
The changes tables are modeled on the services table but have attribute name that start with item, rather than srv. The attributes srv_desc, item_desc and item_list price are nullable. (Read the Create table statements!)

Task 04: Replace_1: **This task uses two sql statements:**
Use a single Replace statement to update the upd_services table with the data in the table upd_services_changes_1
The changes file includes rows that may have a new id value or an id value that matches an existing service id. Do *inserts only*; ignore any rows in changes that match an existing service id value.

Task 05: Replace_2: **This task uses two sql statements:**
Use a single Replace statement to update the upd_services table with the data in the table upd_services_changes_2.

The changes file includes rows that may have a new id value or an id value that matches an existing service id. Do *replacements only*;

Ignore any rows in changes that do not match an existing service id value.

If the description field in the changes table is null, then do not change the description in the services table.

If the price field in the changes table is null, then do not change the price in the services table.

/* For this task you need to know how to handle null columns. A zls is not a null.*/

Task 06: Replace_3: **This task uses two sql statements:**
Use a single Replace statement to update the upd_services table with the data in the table upd_services_changes_3.
The changes file includes rows that may have a new id value or an id value that matches an existing service id. This task will include *inserts and replacements.* But the changes (inserts and updates) should be applied only for changes where the price is between 0 and 100. If the description field in the changes table is null, then do not change the description in the services table on updates.