CS 325 - Week 2 Lab Exercise

Deadline

Due by the end of lab on 2021-09-03.

How to submit

JUST this "driver" for each pair should use `~st10/325submit` to submit the pair's copy of `325lab2.sql` and `325lab2-out.txt`, with a lab number of **82**

Important notes

- You may find the following useful for this lab exercise:
  - NOTE that on the course Canvas site, under Modules, in the "Class Recordings" section, the FIRST link in that section leads to the public course web site's "In-class Examples" section.
  - SQL Reading Packet 1 - Intro to Oracle SQL at HSU
    - (posted on the course Canvas site, under Modules, in the "SQL-related CS 325 Reading Packets" section,

      and on the public course web site, under "In-class Examples", in the "Week 2 Asynchronous Materials" section)
  - `325lect02-2.sql`
    - (posted on the public course web site, under "In-class Examples", in the "Week 2 Asynchronous Materials" section)
- **RECOMMENDATION:** RUN your script-in-progress FREQUENTLY as you are developing it -- do not create the entire script before running it for the first time.
  - save your script-file-in-progress `325lab2.sql`
  - run it in `sqlplus` using:

    ```
    SQL> start 325lab2.sql
    ```

    (or using one of the variants of this discussed in SQL Reading Packet 1)
- You are required to work in **pairs** for this lab exercise. If you are not pair-programming, then you may not receive full credit for your lab exercise.
  - If there are an odd number of students attending lab, or too many students with connectivity issues, some teams may have 3 students.
  - **TYPE BOTH (all) OF YOUR NAMES** in the beginning comment of the file `325lab2.sql`.

Your tasks

- Begin a SQL script **`325lab2.sql`** with comment(s) including at least **BOTH (all)** of your **names** and

today's date. Add commands for the following into this SQL script.

- Start spooling to a file **`325lab2-out.txt`**.
- Use one or more **`prompt`** commands to print a message to the screen containing **both (all)** of your names
- **THINK ABOUT FIRST**: Decide on a table for a type of thing of your choice, that someone might like to **borrow**, with an appropriate name, and at least **FOUR** appropriate columns such that:
  - at least 1 column is character data (fixed or varying, whichever is appropriate for that column)
  - at least 1 column is of a numeric type,
  - (the other at-least-two column(s) can be of appropriate type(s) of your choice)

  CREATE this in SQL:
  - Write a **`drop table`** statement for your table, *including* a **`cascade constraints`** clause.
  - Write a **`create table`** statement for your table.
  - Be sure to include an appropriate **`primary key`** clause for your table!
- Insert at least **6** appropriate rows into your table.
- Write a **`describe`** command for this table (it simply outputs a listing of the table's columns).
- Write a **`select`** statement that will display your table's contents.
- **NEXT**: Think about another table, one for LOANS of an instance of a thing from your first table.
  - IMPORTANT: In this odd-and-small scenario, each loan is for a SINGLE instance of a thing, and you don't care WHO or WHAT you loaned that thing to...!

  Decide on a name and appropriate columns for this second table, such that:
  - there is a column for a unique loan id for each loan instance,
  - there is a column for the date that the loan was begun,
  - there is a column for the date that the thing loaned in this loan was returned,
  - there is a column/columns whose name and domain are those of the primary key of the FIRST table (in this case, they represent the thing that was loaned in this loan).
  - and you may include additional columns if you wish.

  CREATE this second table in SQL:
  - Write a **`drop table`** statement for this second table.
  - Write a **`create table`** statement for this second table.
  - Be sure to include an appropriate **`primary key`** clause for this second table!
  - Be sure to include the appropriate **`foreign key`** clause for this second table, referencing your first table!
- Insert at least **4** appropriate rows into this second table (for at least **4** loan instances).
  - At least one of these loan instances should be for a **completed** loan.
  - At least one of these loan instances should be for a **not-yet-completed** loan (the item has NOT been

returned yet) -- using the version of **`insert`** in which you only insert values into SOME of the columns.

- Write a **`describe`** command for this second table.
- Write a **`select`** statement that will display this second table's contents.
- Turn **off** spooling.
- When you believe your SQL script is working properly (or at the end of lab, whichever comes first), submit your `325lab2.sql` and `325lab2-out.txt` files using `~st10/325submit` with a homework number of `82`.
  - (Once you have submitted your lab exercise files, you may leave lab if you wish. Or, you can ask questions, work on the CS 325 homework, etc.)
- When you believe your SQL script is working properly (or at the end of lab, whichever comes first), the "driver" submit the pair's/trio's `325lab2.sql` and `325lab2-out.txt` files using `~st10/325submit` with a homework number of **82**.
  - (Once your pair's/trio's lab exercise files have been submitted, you may leave lab if you wish. Or, you can ask questions, read SQL Reading Packet 1 and/or DB Reading Packet 2, etc. But note that questions about today's lab exercise will get first priority.)