Assignments

Assignment 2 - Fall 2023 - Not Started

Assignment Details

Title	Assignment 2 - Fall 2023
Student	Yulun Feng
Grade Scale	Points (max 100.00)

Instructions

Assignment 2

Purpose:

To give you practice:

- drawing more ER diagrams
- inserting, modifying and deleting data from tables using SQL
- querying a database using SQL

NOTE: You MUST set up your Virtual Machine (do the assignment worth 1% called *Setting up your VM*) BEFORE you can complete this assignment.

The Problem:

The database you design should be for the following scenario:

The Computer Science instructors at Western are having a hard time keeping their teaching assistants (usually the departments grad students) straight and would like to have a small application that keeps track of the teaching assistants name and image and the course he/she has been has been a teaching assistant for. In assignment 3 you will write the application to solve the above problem. In this assignment you will be given the tables that create a database for the scenario below. That database will be used as the backend database for assignment 3. The database will need to keep track of the following information:

For the teaching assistants, store the:

- first name (up to 30 characters, must have at least 1)
- last name (up to 30 characters, must have at least 1)
- student number (unique - 9 digits) not null
- Western user id (which will be unique, 2 to 8 characters, make this the primary key)
- whether the t.a. is a PhD student or a Masters student
- an image of the t.a. (NOTE: you don't have to store the images in the database, you could store a path to a file on your virtual machine, or a url location, or you could store the image as a blob in the database (if you want), this is up to you, can be null initially).

For the course, store the:

- course number, it will always be the characters CS and then 4 digits (this field must be unique).
- the course name, (up to 30 characters), cannot be empty
- the level of the course (first year, second year, third year or fourth year so it will be one of the following integers: 1, 2, 3 or 4 or it could be null)
- the year the course was created.

For the course offering (an instance of the course above):

- courseOfferId (unique key that is 4 digits long)
- number of students in the course that term (integer greater than 0)
- the term (Fall, Spring or Summer) not null
- the year it was offered (integer greater than 1964) not null

You will need to keep track of:

- which course offerings that t.a.s have worked on and how many hours they worked on that course
- which course is associated with which offering. Eg. The course cs1033 would have an offering id of 7777 that was offered in the Fall of 2023 and had 600 students in it.
- which courses a t.a. loves and which courses a t.a. hates

NOTE

- Some courses are not offered the first time they are created.
- Some courses are not loved by anyone and some courses are not hated by anyone
- T.A.s don't have to love or hate any course. They only decide that for a course if they want to.
- Every T.A. must be a teaching assistant for at least one course offering
- Some course offerings do not get assigned a teaching assistant.

Instructions:

There are 2 parts for this assignment. In part 1 you will draw an ER diagram and in part 2 you will convert your ER diagram into a database and create the database on your virtual machine using MySQL

Part 1:

For the above scenario, draw an ER diagram using MS Visio or draw.io. Then save your ER diagram as last2digits_ERDiagram.jpg (or .pdf, or .gif or .png). You should show the total or partial participation using the single/double lines. You do NOT need to include the (min,max) this time. Do NOT use Crows Feet Notation. Make sure you indicate the primary key and the cardinality. If I haven't mentioned something, then there is a "Common Sense" answer.

Part 2:

The given script below will create the database and insert most of the data for the scenario above. You will write your own script file with some updates, queries, inserts and deletes on the data to demonstrate your SQL knowledge. Make sure the SQL commands are UPPER case and the table names and column names are lowercase.

IMPORTANT:

Get onto your virtual machine using your terminal window or putty (use vm???@cs3319.gaul.csd.uwo.ca, then ssh rocky@vm??? where ??? is the number for the virtual machine you were assigned) and do the following:

- Make sure that you have set up your Bit Bucket Account properly and put your files in a folder that you can push to BitBucket (repo.csd.uwo.ca). You should have done this for the VM Workshop.
- Make sure that you FREQUENTLY push your script file up to Bit Bucket so that you have backups of your work. MAKE SURE YOU PUSH YOUR SCRIPT FILE OFTEN!

Creation of Database and Tables and some Data:

I have supplied you with a file that creates the database and inserts some data. Here is a link to it: https://www.csd.uwo.ca/~lreid2/cs3319/assignments/assignment2/startscriptscriptfall2023.sql

Copy everything in the above link to your virtual machine (in the link above, do ctrl-a and then ctrl-c and then in your virtual machine, within the assignment2 folder, do *nano script1.sql* and press the right mouse key to paste it into script1.sql on your vm and hit ctrl-x to save it and exit) OR find some other way to get it in a file on your VM called *script1.sql* and use it to create your database and data that you will need for your assignment. To run it on your virtual machine, type this command: (HINT: DO NOT COPY AND PASTE THIS COMMAND, type it in yourself, it took me about an hour to figure out that I had copied and pasted it and some weird characters got embedded, TYPE IT, do NOT copy it)

- ***sudo mysql -nvvf --verbose -pyourpassword < script1.sql > outputscript1.txt 2>&1***
  - NOTES:
    - make sure you turn -pyourpassword to be whatever your MySQL password is. Likely you will put: ***-pcs3319***
    - < means pipe every command in the script1.sql INTO mysql to execute each command in order.
    - > means pipe all the output from the commands in the input script to outscript1.txt
    - 2>&1 means also pipe (>) any errors (errors go to 2) into the same location as the stdout(&1) as the output script file as well rather than to the terminal
    - To see what happened when the script was run, just type:
      - *sudo more outputscript1.txt* (NOTE: you likely will not need the sudo part, I put it just incase)

After running the above yellow command, the lines in the script1.sql file will have created an MySQL database and put some data in it. You can go into mysql and check it out within mysql by typing USE assign2db (or you can just type more outputscript1.txt and see what the commands did)

Your script file - Updates, Queries, Inserts and Deletes

Create a simple text file using Notepad or TextEdit or Nano called *last2digitsofyourstudentnumber_script2.sql* that you run after the script1.sql above. Your new script should have the following SQL commands:

1. Write the SQL commands to update the tables as follows:

- Make sure you start each of the 4 parts below with a comment (-- hyphen hyphen space is a comment). For this part put:
  -- Part 1 SQL Updates
  - NOTE: when doing comments in script files you MUST put two hyphens and then you MUST put a space, if you put 3 or more hyphens in a row or 2 hyphens with no space, the comment does NOT work. For example:
    - These 2 comment lines will NOT work:

```
----------
--Part 1 SQL Updates
UPDATE hasworkedon ...
```

    - These 2 comment lines WILL work:

```
-- ---------
-- Part 1 SQL Updates
UPDATE hasworkedon ...
```

- Write the command to show all the data in any table you modify BEFORE you modify it.

- Write the command to update any courses in the course table that have a title of *Multimedia* to now have a title of *Multimedia and Communications*
- Write the updates to make anyone with a first name starting with R to now have worked 200 hours on any course offering they have worked on. Your answer must reference both the ta table and the hasworkedon table (you must look up the first name in the ta table and use that in your update where clause);
- Write the command to show all the data in the course table and hasworkedon table AFTER to prove that your two UPDATE commands worked.

2. Write the SQL commands to insert some new data:

- Make sure you start with a comment (-- hyphen hyphen space is a comment), and put -- *Part 2 SQL Inserts*
- Insert your favourite CS course (any course of your choice) that we offer at Western in the Computer Science Department. Just guess at the year that we started offering it.
- Insert 3 offerings of the course you just added, you can pick when it was offered (the year and the term and make up a random offering id)
- Insert a ta whose name is your favourite actor or actress. Just make up the student number and user id and put either Masters or PhD.
- Insert the data that will show that your new t.a. loves being a t.a. for the new course you just added.
- Write the required statements to prove that all the data above was added (prove that all your INSERT commands worked)

3. Write the SQL commands to query the data.

- Make sure you start with a comment (-- hyphen hyphen space is a comment), and put -- *Part 3 SQL Queries*
- Before each of the below queries, put an MySQL comment that tells the query number to help figure out which queries are answering each of the questions above. . In MySQL comments start with -- , so do something like:
  -- Query 1
  SELECT lastname FROM ...
  -- Query 2
  SELECT ...

  Here are the SQL queries you must write:

1. Show the last names of all the tas
2. Show the last names of all the tas with no repeats
3. Show all the data in the ta table, but show them in order of their first names from A to Z
4. Show the first name, last name and user id for any t.a.s who are completing a Masters degree.
5. List the course offering id (the key), the term, the year and the course number of any course offering of a Database course (use the coursename as the look up, do not just type in the course numbers)
6. Check for the weird situation where a course offering happened BEFORE we even created the course. In this case show all fields from both tables.
7. List the course name and number of the course that is loved by anyone with the last name of Geller
8. Find the total number of students who took the course CS1033. Display the total value, the course name and the course number.
9. List the first name and last name of all the t.a.s who have been assigned to be a t.a. for the first year courses (level = 1). Also show the course number. (do not show repeated identical rows in the results, ie. use DISTINCT)
10. Display the first and last name and hours and the course number of the t.a. who has worked the most hours during a course offering on just one offering of the course (if they worked on 2 or more offerings, do not add

them all together). If there is more than 1 tas which the same max hours, show all the tas. (This one is tricky, you might have to do it in steps using views, i figured out how to do it in one line but it is okay to use views if you want).

11. Find the course name and numbers of any courses that are neither loved nor hated by any t.a.s (HINT: you might want to try using UNION or INTERSECT with this one)
12. Find all tas (last name and first name and number of courses they have been a t.a. for) who have been assigned to MORE than one course offering (HInt: you will have to use the key words *Group By* and *Having*)
13. List the first name and last name of any t.a. and the course number and name worked where the t.a. has been a t.a. for a course they love. HINT: I had to use DISTINCT in order to not show the course offerings each time a ta. was a ta for the same course.
14. List which course has been offered the most in the Fall. Display the course number, course name and the number of times it was offered. You might want to create a view and use the view to do this one (I had to do this and I named the count column so i could refer to it in the next select query).
15. Think of your own query that might be useful for someone interested in using our database. Include a comment to say what query it answers and then put the actual SQL command to answer that query, for example (NOTE: if it uses more than one table it will get a the better the mark, i picked a very silly example):
    -- Query 15-My Query - Display first name of all the tas in alphabetical order by first name if their name contains the letter e
    SELECT firstname FROM ta WHERE firstname LIKE '%e%' ORDER BY firstname;

4. Write the SQL commands to delete some data and to create a view.

- Make sure you start with a comment (-- hyphen hyphen space is a comment), and put -- *Part 4 SQL Views/Deletes*
- Create a view (give it the name of your choice) that lists the ta first and last name and ta user id and the course number and name of each of the courses that the ta hates. Make sure the view is in alphabetical order by the course level (i.e. first year courses should go first)
- Prove that it works by selecting all the rows from it
- Write a query using your view and any other tables you need to show the first and last name and course number of any ta. who got stuck having to work on a course that they hate. Do not show repeats.
- Write a query to show all the ta table information.
- Write a query to show all the hates table information.
- Delete the ta with the userid of *pbing*
- Prove that the ta was deleted.
- Write a query to show all the hates information again
- Try to delete the ta with the userid of *mgeller*. It shouldn't delete, put a comment right after this command to BRIEFLY but CLEARLY explain why pbing got deleted from the hates table as well as the ta table but mgeller didnt get deleted at all.
- Alter the ta table so that it has image column (call the new column *image*). Make this column be VARCHAR (200) (Google the mysql ALTER command to see how to do this)
- Select all the columns from the ta table again to make the sure the column is there but it will be empty right now for all the t.a.s.
- For ta named mgeller, update the value for the image column to be this value: *https://i.pinimg.com/originals/bf/85/8d/bf858d262ce992754e2b78042c9e0fe8.gif*
- Select from the ta table again to make sure that it worked.
- After you have the script working, save the output from it using the following command:
  sudo mysql -nvvv --verbose -pyourpassword < last2digits_script2.sql > last2digits_outputscript2.txt 2>&1

Notes:

- Here is a sample script file: [http://www.csd.uwo.ca/~lreid/cs3319/assignments/assignment2/samplescriptmysql.txt]
- Do the steps (inserts, deletes, etc...) in the order given
- Put MySQL comments (comments start with --) anywhere that you think might help the reader figure out what you are doing.
- Remember: Best practice is for SQL keywords to be uppercase and the tables and columns to be lowercase (or camelCase)

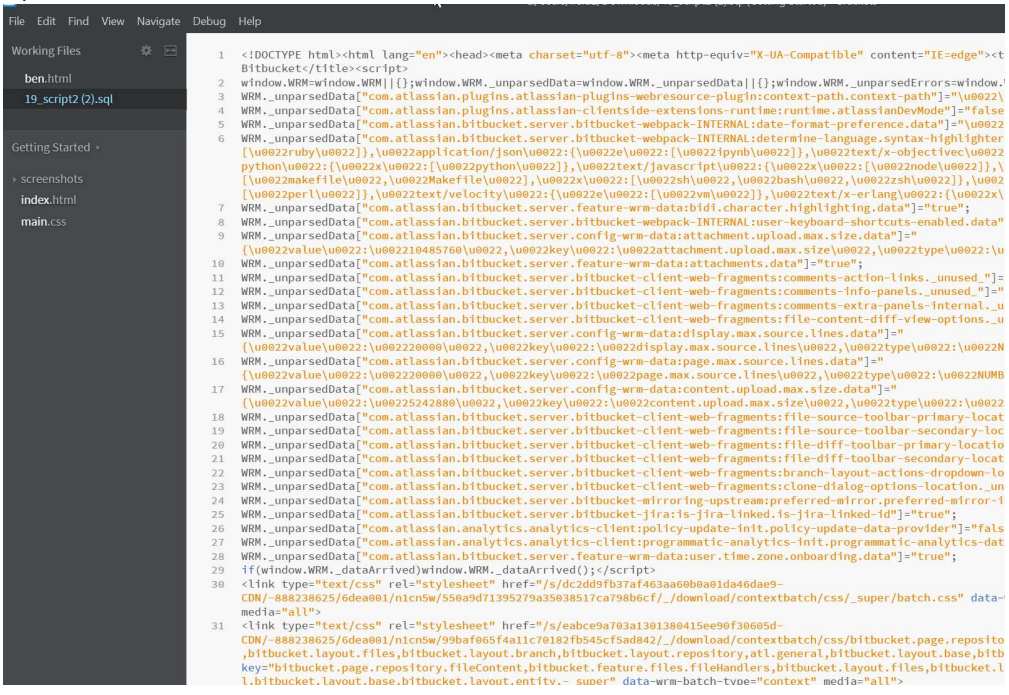
Handing in the Assignment

You are required to submit the following files via **kritik.io:**

- your ER diagram (named last2digitsofyourstudentnumber_ERDiagram.jpg/.pdf/.gif/.png)
- your input script file --> last2digits_script2.sql
- your output script file --> last2digits_outputscript2.txt

IF YOU LOOK IN KRITIK.IO YOU CAN SEE THE RUBRIC FOR THIS ASSIGNMENT!

IMPORTANT WARNING: After you have handed in your files to kritik.io, PLEASE PLEASE PLEASE double check again in Kritik.io (by clicking on your .sql file and .txt file in order to download them yourself and then open those files up again in something like Notepad) to make sure that you uploaded the correct file. If you didn't download your files properly from repo.csd.uwo.ca, I noticed that some students had handed in the wrong file and the file contained garbage from bitbucket. Here is an example of what I saw when I downloaded the student's .sql file from kritik.io and it was INCORRECTLY handed in: PLEASE DOUBLE CHECK BY DOWNLOADING YOUR FILES FROM KRITIK AND OPENING THEM AGAIN (just to make sure you did upload the correct files with no errors like below)



HINT:

To make your scripts, the easiest way to do it is to create a .sql file on your local machine (in Notepad for example) with the commands in it, test those commands by copying them from the .sql file and then pasting them into your virtual machine window while running mysql on your VM, make any necessary changes for commands that don't work in the Notepad .sql file, then save the .sql file every so often. When you are ready to do your final

version, copy the code from the .sql local file and paste it into the virtual machine window (and occasionally use Push with Bit Bucket). **THIS METHOD WILL AVOID TYPING ERRORS AND LOSING YOUR FILES!!!**

This assignment does not accept online submissions. Contact your instructor for additional instructions.

Back to list