CSCU9B3: Database Principles and Applications Assignment

50% of Module Grade

Due Date: 4pm Monday 18th November

This assignment is designed to test your knowledge of ER modelling, normalisation and use of the SQL and PHP languages.

In addition to working in your own time, you will have a full practical session to work on your assignment. You should use this opportunity to ask demonstrators about any theoretical or practical issues you are not sure about (but note, they will **NOT** be able to help with formulating solutions to the questions in this assignment!)

The scenario and associated data

For several decades, starting in the late 1950s, there was a vigorous pantomime tradition in Stirling, in the old Gaiety Theatre (sadly, now demolished). Each year a pantomime would be put on, directed by one of a small group of directors. Each director was a local person, and the records give the town in which each director was born. There was a pool of actors (called "stars" in the profession) who appeared in the pantomimes: each star was associated with a highly stereotyped performance role, such as "Slapstick" or "Romantic lead." Stars never changed their role from year to year. The stars were professionals, and each had an agent. Agents were associated with the towns in which they had their offices: they never moved their offices from one town to another. Stars could (and did) change their agents from year to year. Agents were able to act for more than one star in any given year.

Fortunately, some records of these pantomimes have survived. A devoted local historian has made some progress with the surviving records and has established a single table covering the material discussed above. This has been stored in a simple CSV file. The aim here is to turn this into a usable relational database. Each row of the table in the file gives information about the pantomime that was mounted in a particular year, its director, one of the stars that appeared in it, and that star's current agent. It is clear that the table is in First Normal Form (1NF) but not 2NF or 3NF, so it should be decomposed into a number of smaller tables to implement it as a relational database.

Assignment Requirements

Your assignment is to complete the following tasks and present your results in a written report. The instructions are detailed and following them properly should ensure you get good marks.

It is your job to turn this data into a relational database. The data is stored in the **panto.csv** file which you can download from the module's Canvas assignment page. Note that a version without header information, for loading into the database, is in **loadpanto.csv**, on this page.

Task A: Design your database using E-R modelling and normalisation (40% of marks)

From the scenario given above, and by examining the data in **panto.csv**, use E-R modelling to design a database for this data. In your **written report**, fully document your design process: this should include your reasoning behind choosing entities and their relationships, leading to one or more

Entity-Relationship (ER) diagrams that model the information requirements of the theatre group. Be sure to state your design assumptions and show the cardinality and optionality of each relationship that you identify.

Your final design should be in Third Normal Form (3NF).

Either in your final ER diagram, or written as a set of relations, show all attributes of each entity and identify the primary key and any foreign keys used to form relationships.

Perhaps by giving the predicates for each relation and/or by giving explicit examples from data, justify that each of your relations is in 3NF. Also, with the aid of explicit examples, discuss in detail why 3NF is desirable for the maintenance and use of your database.

Task B: MySQL database implementation (15% of marks)

To implement your relational data model created in Task A, set up the new tables in your MySQL database by writing and executing, via phpMyAdmin, appropriate SQL CREATE statements that choose appropriate data types for your fields (attributes) and set the primary and foreign keys that will be needed.

Using SQL, create a suitable table to hold the data in **loadpanto.csv** and then upload the data using the "import" facility of phpMyAdmin.

Load the tables of your ER design with data drawn from this data-holding table using appropriate SQL INSERT .. SELECT commands.

Include all the SQL for creating and loading the tables in your **report**.

Task C: Querying your database (25% of marks)

Write and execute **single** SQL statements to provide answers to each of the following queries. Show your SQL and the query results in your **report**:

1) List the names of all directors from Bannockburn.
2) Display a list, with suitable column headings, of how often each pantomime has been performed, shown in ascending order of the number of performances.
3) Display the total number of performances ever given by each type of star role.
4) Display a list of all directors who have directed the pantomime "Aladdin" and in what year.
5) List all the stars and their roles from final performance (determined as a part of this query) of the pantomime "Aladdin".

Task D: Use PHP to interface to your database (15% of marks)

Starting with the template file, **assignment.php** (available from Canvas), and using the PHP **mysqli** package (either the procedural or object-oriented version), complete the PHP and SQL required to take whatever text is entered in the form box and query the tables corresponding to your ER design to do the following:

1) Search for any stars whose names (forename or surname) contain the text entered.

   a. Make sure your code is robust against any mistakes or malicious intent in the text entered in the form box by a user of your webpage. During marking, your webpage will be trialled and this will be checked.
2) Display neatly in the web page the following characteristics of all matching stars found:
   a. ID,
   b. complete name,
   c. role,
   d. each agent (show the agent name) they have had and the year(s) of that agent.

Try out your code by placing this file (DO NOT rename it) in your web folder on wamp0, as you did in practical exercises (\\wamp0.cs.stir.ac.uk\www\xxx where xxx is your username).

In your **report**, show one **screenshot** of a query result, and include a copy of your **code** (please **REMOVE** your password and any other sensitive information) and give its URL ie http://wamp0.cs.stir.ac.uk/xxx/assignment.php.

Submitting your work and assessment procedures

The assignment will be submitted as an electronic type-written report uploaded (via Turnitin) to Canvas by 4pm on Monday 18th November. DO NOT put your name in the report, only your registration number.

Summary of marking breakdown and criteria

Your work will be marked out of 100, according to the following breakdown:

1) Database design and its justification: 40%
2) Database implementation: 15%
3) SQL queries: 25%
4) PHP (assignment.php): 15%
5) Overall quality of your report: 5%

Marks will be awarded both for the technical correctness of what you have done and also for the clarity and organisation of how you describe it in the report.

The assignment counts for **50% of the total module grade**.

Late submission

It is possible for the co-ordinator to vary the dates for a student. If you must submit your work late (perhaps for medical reasons) then this can be arranged, but it must be discussed with the co-ordinator as soon as possible after the problem becomes known. Coursework that is submitted late will be accepted up to seven days after the published deadline (or expiry of any agreed extension) but the mark will be lowered by three marks per day or part thereof. After seven days the piece of work will be deemed a non-submission and you will receive a mark of 0 for the coursework and a **Fail grade** for the whole module.

Plagiarism

Work which is submitted for assessment must be your own work. All students should note that the University has a formal policy on plagiarism which can be found at http://www.quality.stir.ac.uk/ac-policy/assessment.php.