UNIVERSITY OF LONDON

CM2040

BSc EXAMINATION

COMPUTER SCIENCE

Databases, Networks and the Web

Release date: Monday 20 March 2023 at 12:00 midday Greenwich Mean Time

Submission date: Tuesday 21 March 2023 by 12:00 midday Greenwich Mean Time

Time allowed: 24 hours to submit

INSTRUCTIONS TO CANDIDATES:

Section A of this assessment paper consists of a set of **TEN** Multiple Choice Questions (MCQs) which you will take separately from this paper. You should attempt to answer **ALL** the questions in Section A. The maximum mark for Section A is **40**.

Section A will be completed online on the VLE. You may choose to access the MCQs at any time following the release of the paper, but once you have accessed the MCQs you must submit your answers before the deadline or within 4 hours of starting, whichever occurs first.

Section B of this assessment paper is an online assessment to be completed within the same 24-hour window as Section A. We anticipate that approximately **1 hour** is sufficient for you to answer Section B. Candidates must answer **TWO** out of the THREE questions in Section B. The maximum mark for Section B is **60**.

Calculators are not permitted in this examination. Credit will only be given if all workings are shown.

You should complete **Section B** of this paper and submit your answers as **one document**, if possible, in Microsoft Word or PDF to the appropriate area on the VLE. Each file uploaded must be accompanied by a coversheet containing your **candidate number** written clearly at the top of the page before you upload your work. Do not write your name anywhere in your answers.

UL23/0215

SECTION A

Candidates should answer the **TEN** Multiple Choice Questions (MCQs) quiz, **Question 1** in Section A on the VLE.

SECTION B

Candidates should answer any **TWO** questions in Section B.

Question 2

You are a backend developer for a Global Consulting Firm global_consult.com, and are responsible for implementing a MySQL database system. Your task is to design and develop a web application allowing the firm's employees to perform the following functions:

- Setup **client profiles** with standard information such as client-company name, company division, contact person, email, password.
  Note: **Clients** are Companies not individuals, although a contact person from a **Client Company** sets up and manages the **client profile** on behalf of that Company.
- Create **projects** defining the client company, the recommended end date, the knowledge domain required for phase A of the project, and knowledge domain for phase B. (the consulting firm specializes in a limited number of domains including IT, Finance, Change Management)
- Additional attributes are needed to update projects' status, which progresses through the following steps - "not started", "phaseA_1", "phaseA_2", "phaseB_1", "phaseB_2", "completed".
- **Consultants** are employees of the Global Consulting firms, who contribute to projects according to their competencies. One of the functionalities of the web application is creating **consultant_allocations** deal with the booking of consultants to projects, based on competency domains of consultants, and progress status of projects. At any point in time, a consultant can contribute to at most one project.
- A client can create several projects. A created project can then start at any time but must be completed by the recommended end date. Once a project has started, each one of steps "phaseA_1" through "phaseB_2" takes one month.
- Each consultant has one high-competency knowledge domain, and one medium-competency domain. Consultants are only booked on and contribute to projects during steps requiring their competencies, with a priority on high-skilled contributions.

(a) For the global_consult database, create an Entity Relationship Diagram (ERD), using crow’s foot notation. Entities, relationships between entities, and appropriate association types should all be included in your diagram. Use the minimum number of table that provide for the required database functionality.

[10]

(b) For each table of the database you designed in (a), list the primary and foreign keys.

[4]

(c) Create the tables you designed in your answer to part (a) with appropriate SQL code.

[14]

(d) Give scenarios how a Project Leader can benefit from the database you have created, when he selects a project's start date and contributors throughout project's life.

[2]

Question 3

A Company subscribes to part of the available range of cloud resources, offered by a Cloud Provider, and hires a Cloud Administrator. Access to subscribed cloud resources is granted according to job titles (roles) in the Company.

Use of resources is costed per hour. For each employee, there is a limit on accumulated daily cost. For each type of resource, there is a limit on accumulated daily cost by the Company. The Cloud Administrator is responsible for allocating resources to employees, and for monitoring that accumulated costs are within limits.

Consider the following tables, which the backend developers of cloud_provider.com include in a server-side database, in order to facilitate the client company's administrator performing the above tasks. The primary and foreign keys are underlined.

- *Role (role_id, role_title)*
  Roles table holding the roles (employment positions) in the Company.
- *Resource (resource_id, name, hourly_cost, company_cost_limit)*
  Resources table holding the name, **hourly** cost, and **daily** limit on the cost of using resources by the Company.
- *Permission (permission_id, role_id, resource_id)*
  Permissions table holding permissions granted to job titles in the company to access cloud resources. A job title may be granted access to several types of resources, and each resource may be accessed by several job titles.
- *Employee (employee_id, name, role_id, employee_cost_limit)*
  Employees table holding the name and role in company, and daily cost limit on using cloud resources.
- *Expenses (expenses_id, hour, employee_id, resource_id, accumulated_cost)*
  Expenses table holding dollar-value of expenses by employees for using resources, each **hour** of current day.

For each answer in this question you must include a short but complete explanation of how your SQL command works (NB. unexplained aspects of the command will not be awarded points).

Your answers must also be presented in a cut and pasteable format as we will be running your commands to mark them.

(a) Consider a role title '*somerole*'. Write SQL code that identifies role's total cost limit. Write SQL code that shows the number of resource the role is granted access to.

[5]

(b) The Expenses table for the current day is populated correctly in advance, except for the *accumulated_cost* field. At the end of each hour (1,24), that field is populated with accumulated so far expenses, by employee on resource. If employees reach their daily cost limit, the Administrator disallows their access till end of day.

Consider an employee called '*some employee*', who reaches his/her daily cost limit. Write SQL code that identifies the hour when the limit is reached.

Hint: MySQL requires that a derived table is given an alias, e.g. AS Subtable. Other SQL vendors do not require the alias.

[5]

(c) Use the identified *hour* (1,24) in point (b) above. Write SQL code that populates cells in the Expenses table that correspond to the next hour and to the employee '*some employee*'. How the Expenses table will be populated for the rest of the day for this employee?

Hint: Remember that to what extent the Expenses table is populated in advance. [5]

(d) Now focus on reaching the *company_cost_limit.* Write SQL code that identifies the hour when that limit is reached.

Hint: MySQL requires that a derived table is given an alias, e.g. AS Subtable. Other SQL vendors do not require the alias.

[5]

(e) Use the identified *hour* in point (d) above. Assume that hour happens after the one identified in point (b). Consider an employee '*another employee*' has not reached his/her *employee_cost_limit*. Write SQL code that populates table Expenses with the information relevant to '*some employee',* and information relevant to '*another employee*', in the hour after the one identified in (d).

[5]

(f) Review the JOIN operators, nested queries, and aggregated functions in points (a)-(e) above. Why have you made that choice? Which of the tables have you involved for the different tasks?

Can you suggest an alternative solution to some of points (b)-(e)? How do your two solutions compare? [5]

Question 4

You are a freelance backend developer responsible for implementing a relational database for a 'to do' list web application using Express and Node.js and the MySQL database management system. You have been tasked with designing and developing a web application that will allow an end-user to perform the following tasks:

- Create an account/user profile.
- Login using their credentials.
- Add a new 'to do' item to their list.
- List all items in their 'to do' list sorted by dates.

(a) Write a piece of middleware code, with a route named 'create_account' as follows:

- to store form data collected by a template file named account.ejs.
- account.ejs is already rendered in another route of the web application.
- Form data include *username* and *password*.
- Your code should access the database to store the account data.
- Your code should handle the error condition where the database query fails.
- Your code should display a message when an account is successfully created.

Please note, you are only required to write the middleware code and not the template file in this section of the question. Make sure you add comments for each section of the code.

[10]

(b) Write an EJS template file for creating a user account named account.ejs. Please note the 'title' of the page is passed as a parameter to this EJS file.

[7]

(c) Write a piece of code for another route called 'login' to let a user log in based on username and password already saved in the database as follows:

- To collect form data by a template file named login.ejs similar to account.ejs you have already written in part (b).
- To compare collected form data with data already saved in the database.
- The user is able to log in if and only if both username and password match with data saved in the database.
- Your code should handle the error condition where the database query fails.
- Your code should display appropriate error messages when the username is not found in the database or the password does not match.
- Please note each username saved in the database is unique.

[13]

END OF PAPER