UNIVERSITY OF LONDON

MSc Computer Science
Coursework Brief

Module: CSM040 Data Management
Coursework: July to September 2023 study session

Submission Deadline: Monday 2nd October 2023: 13:00

- You may submit the majority of your answers to the assessment questions in a Word document or PDF. Some of the questions you can optionally submit hand written solutions, which can be photographed or scanned and inserted into your submission file. The questions where you can submit hand written solutions are highlighted.
- Please Note: You are permitted to upload your Coursework in the final submission area as many times as you like before the deadline. You will receive a similarity/originality score which represents what the Turnitin system identifies as work similar to another source. The originality score can take over 24 hours to generate, especially at busy times e.g., submission deadline.
- If you upload the wrong version of your Coursework, you can upload the correct version of your Coursework via the same submission area. You simply need to click on the 'submit paper' button again and submit your new version before the deadline.

  In doing so, this will delete the previous version which you submitted, and your new updated version will replace it. Therefore, your Turnitin similarity score should not be affected. If there is a change in your Turnitin similarity score, it will be due to any changes you may have made to your Coursework.
- Please Note: When the due date is reached, the version you have submitted last, will be considered as your final submission and it will be the version that is marked.
- **Once the due date has passed, it will not be possible for you to upload a different version of your assessment. Therefore, you must ensure you have submitted the correct version of your assessment which you wish to be marked, by the due date.**

Coursework is weighted at 100% of final mark for the module.

Answer all **four** questions.

A total of 150 marks is available. The breakdown of marks per question is as follows:

Question	1	2	3	4
Marks	40	30	40	40

You are encouraged to test your solutions for **Question 1**, **Question 2** and **Question 4** by creating your own MySQL/MariaDB schemas and test datasets in codio.

Question 1 [40 marks in total]

A MySQL relational database is used to record information about an online auction site where members can sell items and bid for items.

The database contains the following tables:

```
MEMBER (MEMBER_ID, MEMBER_NAME, DOB, MEMBER_TYPE)

ITEM (ITEM_ID, SELLER_ID, DESCRIPTION,
      RESERVE_PRICE, START_DATE, AUCTION_LENGTH)

BID (BID_ID, ITEM_ID, BIDDER_ID, BID_DATE,
     BID_AMOUNT)

REVIEW (ITEM_ID, RATING)
```

Each member of the site, who either is selling an item or bidding for an item is identified by a unique identifier `MEMBER_ID`.

Each item and bid has a unique identifier `ITEM_ID` and `BID_ID` respectively.

A row in `MEMBER` records the unique identifier, member name, member date of birth and membership type of each member which can only be one of `'STANDARD'`, `'SILVER'` or `'GOLD'`. For example:

```
('M5', 'Abul', '1982-02-27', 'SILVER')
('M6', 'Martyn', '1961-07-11', 'STANDARD')
```

A row in `ITEM` records the unique item identifier, id of member selling the item, description of item, reserve price of item, start date of auction and length of the auction in days for each item being auctioned, for example:

```
('I95','M5','armchair', 120.00, '2022-03-01', 21)
```

A row in `BID` records the unique bid identifier, id of item, id of member making the bid, date of bid and amount of bid for each bid. For example:

```
('B53', 'I95', 'M6', '2022-03-03', 75.50)
```

A row in `REVIEW` records the id of the item and the rating given by a winning bidder which can be an integer between 1 and 10. For example:

```
('I95', 7)
```

The winning bid for an item is the highest bid received by the time the auction closes for that item. Any bid received after the auction has closed is not recorded. A winning bid must be no lower than the reserve price for an item. If more than one bidder makes the same highest bid for a lot, the earliest bid will be successful.

a) Write both **relational algebra** and **SQL** statements to answer the following queries:

i. Find the name, and date of birth of each member who has put an item up for auction where the item has received any bid that is greater than the reserve price.

(7 marks)

ii. Find the description of each item which has received bids from members of both `'SILVER'` and `'GOLD'` membership types.

(7 marks)

iii. Find the name of each member who has received a review rating of 10 for all of the items they have auctioned where a review has been made.

(8 marks)

iv. Find the description of each item that has received bids from every member who has a `'GOLD'` membership type.

(8 marks)

b) Write **SQL** statements to answer the following queries:

i. Find the name and the average review rating for each member who put more than 10 items up for auction where the auction start date was in the calendar year 2022.
(Note that in SQL you can assume you can use the comparison operators `>, >=, =, <, <=` to compare two attributes of type DATE).

(3 marks)

ii. Find the name and membership type of any member who made successful bids in more than 50% of the items they bid for.

(5 marks)

c) Write down the SQL statement to delete rows from the `REVIEW` table where the start date of the auction for the item being reviewed is earlier than `'2019-12-31'`.

(Note that in SQL you can assume you can use the comparison operators `>, >=, =, <, <=` to compare two attributes of type DATE).

(2 marks)

Question 2 [30 marks in total]

a) Identify with explanation appropriate primary key and foreign key constraints for the tables in Question 1, making clear which table each constraint relates to. State any assumptions made.

(10 marks)

b) There is concern that the tables described in Question 1 could hold incomplete or inconsistent data.

Explain to what extent the problems identified as (i) to (iv) below could be avoided if appropriate SQL constraints were defined on the tables of the database described in Question 1. Write down the SQL clause for the constraint in each case, making clear which table each constraint relates to. State any assumptions you make.

i. Some reviews may refer to non-existent items.

(3 marks)

ii. Some bids may be recorded as having non-existent item and bidder identifiers.

(3 marks)

iii. Some members are recorded more than once with different membership types.

(4 marks)

iv. Each member should be recorded as having a membership type of `'STANDARD'`, `'SILVER'` or `'GOLD'`, but this may not have been recorded for all members.

(4 marks)

c) Give two examples with explanations of specific indexes which you think might improve the performance of your queries you wrote in Question 1.

(6 marks)

Question 3 [40 marks in total]

The following schedule shows three transactions TA, TB and TC with time increasing from top to bottom. The schedule assumes the existence of a READ statement to read a record from the database and a WRITE statement to write the record back to the database. It is assumed that a record must be read before it is written. The transactions terminate in the order TA, TC, TB with a COMMIT in each case.

TA	TB	TC
READ R1		
	READ R2	
		READ R3
	WRITE R2	
READ R3		
READ R2		
		WRITE R3
		READ R2
	READ R1	
	READ R4	
		READ R4
		READ R5
		WRITE R4
COMMIT		
		WRITE R5
		COMMIT
	COMMIT	

a) Explain how this schedule would be modified if the following database concurrency algorithms described in the module notes were in operation. Assume that any transaction which goes into a wait state proceeds as soon as possible once locks are released. In each case state the order in which the transactions terminate and give the revised schedule annotated to show the effect of the concurrency algorithm.

i. locking with S (share) and X (exclusive) locks.

(7 marks)

ii. timestamping algorithm assuming transactions `TA`, `TB` and `TC` are in increasing timestamp order.

(9 marks)

iii. multiversion-based control of read-only transactions together with locking based control of read-write transactions.

(9 marks)

b) Annotate the amended schedule given in your answer to part a) (iii) of this question to show which version of each record (`R1`, `R2`, `R3`, `R4` and `R4`) is read for each `READ` operation in the schedule.

(3 marks)

c) Considering just the successful transactions in each amended schedule in your answer to part (a), what serial order would each amended schedule of successful transactions be equivalent to? Note the reasons for your answers.

(3 marks)

d) Briefly explain whether or not the original schedule given in the question is conflict serializable and if it is what the equivalent serial schedule would be.

(6 marks)

e) Explain whether or not the locking algorithm you have shown in answer to part (a) (i) of this question is two-phase.

(3 marks)

Question 4 [40 marks in total]

A MySQL relational database is used to record responses of students to questionnaires rating aspects of modules.

Students complete a questionnaire for each module they take in a year. In any year questionnaires have the same questions for every module.

The database includes tables created as follows:

```
MODULE (MOD_CODE, LEADER)

MOD_ENROL (YEAR, MOD_CODE, STUDENT)

MOD_QUESTIONNAIRE (YEAR, Q_NO, Q_TEXT)

MOD_RESPONSE (YEAR, MOD_CODE, STUDENT, Q_NO,
              SCORE)
```

A row is stored in `MODULE` for each module recording the module code and leader, for example `('Python', 'RICHARD')`.

A row is stored in `MOD_ENROL` for each module enrolment recording the year, module and student, for example `(2023, 'Python', 'MARY01')`.

Rows in `MOD_QUESTIONNAIRE` record the questions in questionnaires each year. So, for example, assuming that for questionnaires in 2023 question 1 asks students to rate whether the workload for a module was appropriate, the following row would be stored:

```
(2023, 1, 'the workload for the module was
appropriate')
```

Rows in `MOD_RESPONSE` record the responses of students to the questions using a scale from 1 to 9. So, for example, assuming that `MARY01` responds to question 1 of the 2023 Python questionnaire with a rating of 6, the following row would be stored.

```
(2023, 'Python', 'MARY01', 1, 6)
```

Values stored for `LEADER` and `STUDENT` are the MySQL usernames of module leaders and students respectively.

Views are an additional mechanism which can help support secure access to table data when used with appropriate privileges. The following views are proposed to control the data visible to different database users when logged on to MySQL.

Note that in MySQL you may reference the function `USER()` in an SQL statement, for example in a `SELECT` or `WHERE` clause. It returns the username of the person executing the SQL statement. To return the username without the hostname included, you can use the `SUBSTRING` function as follows:

```
SUBSTRING_INDEX(user(),'@',1)
```

Also, a function `ROUND(n,1)` returns n rounded to one decimal place.

Give the SQL statements for the creation of views in MySQL to support these requirements:

i. One single view is required with the same columns as `MOD_ENROL` which gives access to all rows to module leaders but gives students access to only the rows for their own module enrolments.

(10 marks)

ii. One single view is required giving access to the responses of students to the questionnaires:

```
MOD_QUEST_RESPONSE (YEAR, MOD_CODE, Q_NO, Q_TEXT,
                    SCORE)
```

The view contains a row for each response of a student to a question on a questionnaire. For example, a row `(2023, 'Python', 1, 'the workload for the module was appropriate', 6)` records that a response to question 1 of the 2023 Python questionnaire gave a score of 6.

`MOD_QUEST_RESPONSE` differs in which rows it gives access to by students and module leaders. The view should give students access

to rows recording their own responses to questions while giving module leaders access to rows recording responses to questions for the modules they lead.

(12 marks)

iii. Two views are required summarising the questionnaire results in each year:

```
MOD_QUEST_SUMMARY(YEAR, MOD_CODE, Q_NO, Q_TEXT,
                  MIN_SCORE, MAX_SCORE,
                  AVG_SCORE)
MOD_OVERALL_SUMMARY (YEAR, MOD_CODE, AVG_SCORE)
```

A row in `MOD_QUEST_SUMMARY` records the lowest, highest and average scores for each question of a module questionnaire in a year. For example, if for question 1 of the 2023 `Python` questionnaire the lowest score was 3, the highest score was 8 and the average score was 5.7, these scores would be recorded in the row for that question. The view should give a module leader access to rows summarising the results for questions relating to his or her own modules.

A row in `MOD_OVERALL_SUMMARY` records the average score for all questions of a module questionnaire in a year. For example, if the average score for all questions of the 2023 `Python` questionnaire was 5.5, this score would be recorded in the row for that module. The view should give a module leader access to rows summarising the results for all modules.

(18 marks)

Note: Your solutions to Question 4 should be extensible and not limited to any particular sample dataset. Therefore using the GRANT/REVOKE access control support in SQL is not a correct approach. The access control should be solely in the definition of the views.

Assessment Criteria

Please refer to Appendix C of the Programme Regulations for detailed Assessment Criteria.

Plagiarism

This is cheating. Do not be tempted and certainly do not succumb to temptation. Plagiarised copies are invariably rooted out and severe penalties apply. All assignment submissions are electronically tested for plagiarism. More information may be accessed via: https://learn.london.ac.uk/mod/page/view.php?id=3214