**UNIVERSITY OF RUHUNA**

Faculty of Engineering

Mini Project – Semester 4: January 2024

Module Number: EE4202, EE4350 Module Name: Database Systems

**Deadline: - 2nd April 2024**

[15 Marks**]**

Instructions:

* You can select any schema of relational database as you wish. Some example topics are,
  + Airport
  + Harbour
  + Bank…
* You want to update the following google sheet once you finalize the topic with your team members to avoid any overlapping with other groups.
* <https://docs.google.com/spreadsheets/d/1Cak2QGfWpoJSPiEIz22wBX0ySqaNnt7ZAn7P-1xuqbY/edit?usp=sharing>
* The final report needs to contain all the details of creating the database.
* Do not crop the screenshots. Include the full screenshot.
* Use the centralized architecture for all designs.
* You must include all screenshots which you used to create the database and query results.
* Submit your MySQL schema folder and final report in PDF formats to the LMS.
* Your report name must be name as GP\_XX\_Schema\_name.PDF. Eg: GP\_01\_Airport.PDF
* One submission per group is necessary and sufficient. If you submit multiple reports per group; the last report submitted before the deadline will be considered as the proper submission unless you request by email to the module coordinator.
* You are given 3 months of time to complete the project and submit to LMS on or before the deadline: 02/04/2024

***Relational Database***

* Chapter 1 - Requirement Analysis

Functional requirements and data requirements of the schema must be identified, listed and described in First chapter.

* Chapter 2 - Conceptual Design

Draw the Entity - Relationship (ER) diagram of the whole schema using an appropriate tool. There must be at least 6 strong entities and 2 weak entities in your design. Further, ER diagram should consist of *at least* one recursive relationship, one multivalued attribute, one composite attribute and one derived attribute. There should be at least one entity having multiple candidate keys. Use the (min, max) notation in showing structural constraints and show the role names in relationships.

* Chapter 3 - Implementation

Use MySQL client for implementation of the 2NF of the representational data model. Your schema name must be as in the shared google sheet. Include the schema creation, table definitions, Insert, Update, Delete Operations with screen shots. You must populate all your tables with 6 rows of data, update 2 rows, delete 1 row.

* Chapter 4 - Transactions

Write 20 queries (7 simple queries and 13 complex queries) to retrieve data from the database. You should include screenshot of each query and result retrieved.

Simple queries - Out of the 7 simple queries; 1 query should demonstrate select operation, 1 for the project operation, 1 for the cartesian product operation, 1 for creating a user view, 1 for renaming operation, 1 query demonstrating the use of an aggregation function (Ex: Average, Maximum, minimum), 1 query to demonstrate the use of LIKE keyword

Complex queries - Use aliasing in all complex queries. Out of the 13 complex queries; 4 should demonstrate basic set operations (union, intersection, set difference, division) without user views and 6 should represent inner join, natural join, left outer join, right outer join, full outer join, **outer union** relational algebraic operations by using user views. There should be 3 nested queries in combination with any other relational algebraic operation.

* Chapter 5 - Database Tuning

Tune 10 selected complex queries that you have written in chapter 5 using any or all of the techniques that you have studied. Write the tuned queries and retrieve the data. Show screenshots of all tuned queries and retrievals.

Prove that your query is tuned by comparing how querying complexity has been reduced using the screenshots which compare the number of rows accessed in tuned queries and original queries retrieved using explain statement for select operation. Screenshots for each of the explain statement of original query and explain statement for tuned query must be included (Do not compare the querying time).

Hints: Here you can basically show how indexing can be used to speed up your queries. Use query tuning guidelines to identify cases where there is no speed improvement in queries even by using indices. Some guidelines may not be applicable to MySQL DBMS as the query optimizer does the query tuning itself.

Further, you can use intersect operation to speed up a correlated nested query.

Ex: select Depe\_name from dependent where Depe\_name IN (select Dept\_Head from Head);

can be replaced by select Depe\_name from dependent intersect select Dept\_Head from Head;