Department of Information and Communication Technology

Faculty of Technology

University of Ruhuna

Database Management Systems Practicum

ICT1222

Assignment 02 – Mini Project

Group 21

Submitted to : Mr. P.H.P. Nuwan Laksiri

Submitted by: TG/2023/1755 : K.R.M.D.N. Nimshan

TG/2023/1760 : D.G.B.N. Dilshan

TG/2023/1726 :V.C.L. Jayasuriya

TG/2023/1751 : V.C.H. Kulathilaka

Table of Contents

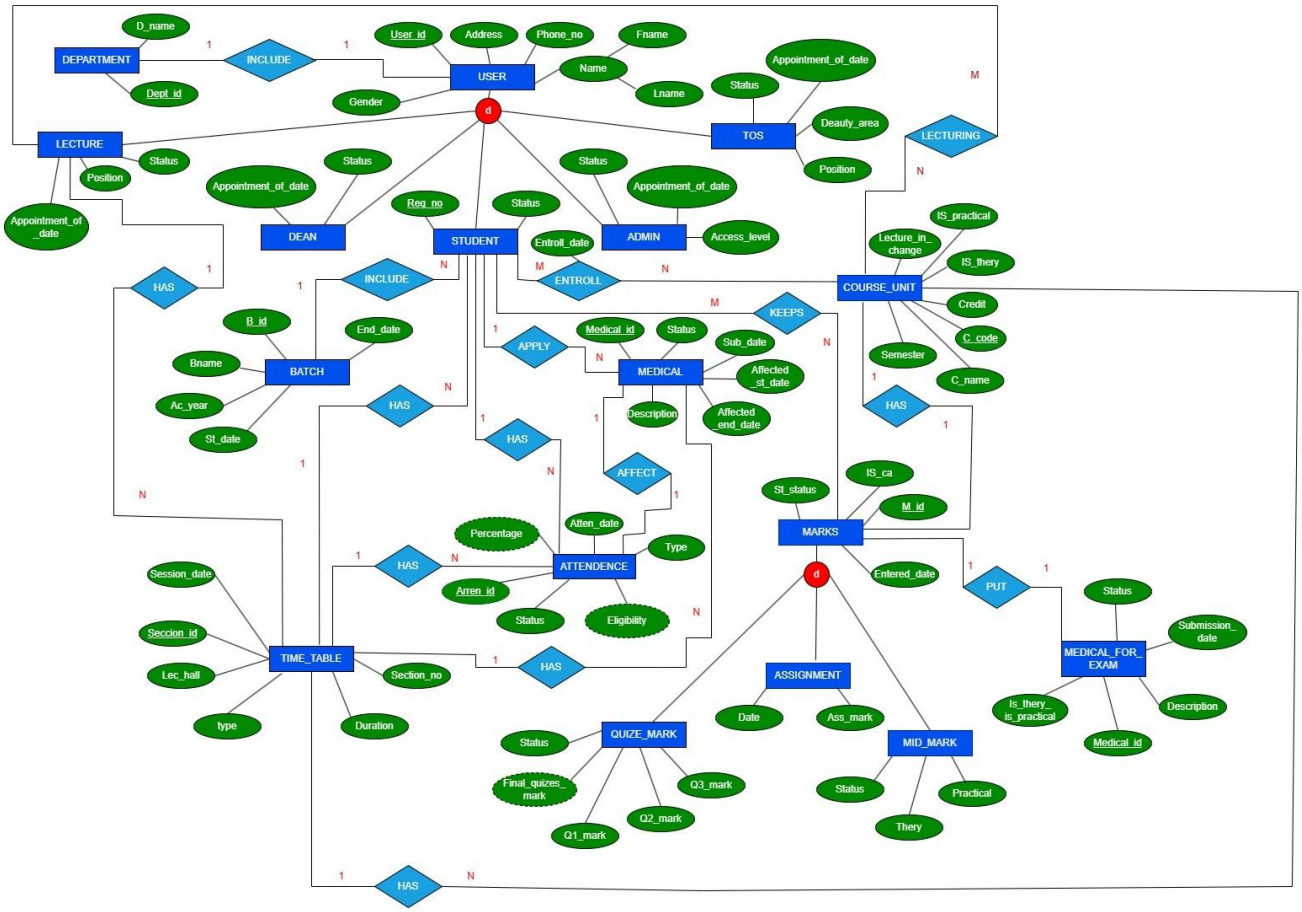
1. Introduction

This project was developed as part of the ICT1222 - Database Management Systems Practicum course. The main objective of this project was to design and implement a database system for managing student information, attendance records, and examination results for the Faculty of Technology, University of Ruhuna. The system helps lecturers, technical officers, and administrators to efficiently manage and retrieve student-related data.

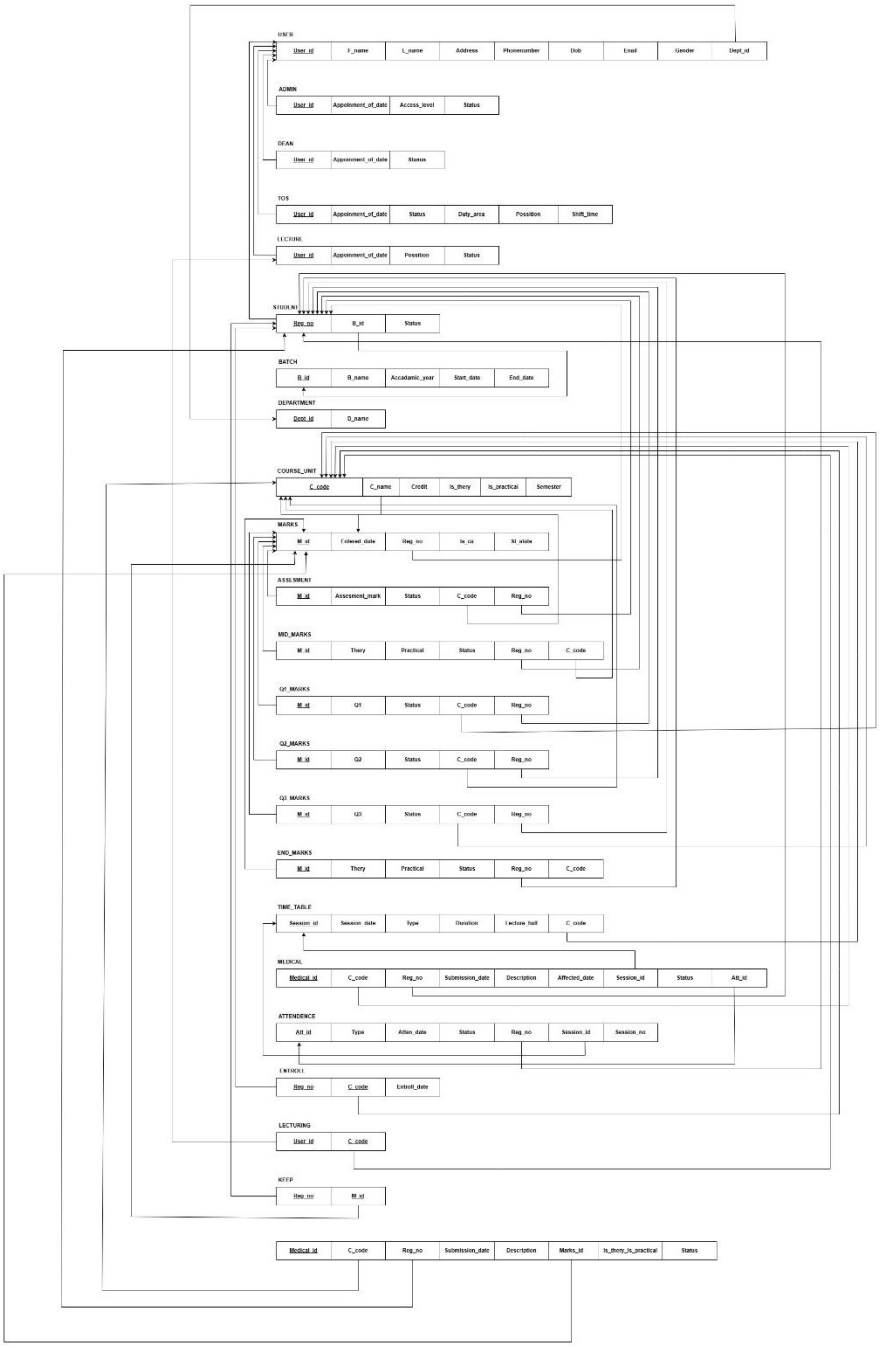
2. Solution Overview

Our solution is a MySQL-based database management system that stores and manages student details, course information, attendance, and examination marks. The database includes stored procedures and views to calculate Continuous Assessment (CA) marks, End Exam marks, and GPA values. These automated features help to reduce manual work, ensure accuracy, and improve efficiency.

3. ER/EER Diagram



4. Relational Mapping Diagram



5. Table Structures

```
desc medical_for_exam;
+----------------------+-----------------------------------------------------------------------------------------+------+-----+---------+-------+
| Field                | Type                                                                                    | Null | Key | Default | Extra |
+----------------------+-----------------------------------------------------------------------------------------+------+-----+---------+-------+
| medical_id           | varchar(5)                                                                              | NO   | PRI | NULL    |       |
| c_code               | varchar(20)                                                                             | NO   |     | NULL    |       |
| reg_no               | varchar(20)                                                                             | NO   |     | NULL    |       |
| submission_date      | date                                                                                    | NO   |     | NULL    |       |
| description          | text                                                                                    | YES  |     | NULL    |       |
| marks_type           | enum('q1_marks','q2_marks','q3_marks','assesment_marks','mid_masks','end_marks')        | NO   |     | NULL    |       |
| is_practical_or_theory | enum('practical','theory','no')                                                       | NO   |     | no      |       |
| status               | enum('pending','approved','not approved')                                               | NO   |     | pending |       |
+----------------------+-----------------------------------------------------------------------------------------+------+-----+---------+-------+
```

```
desc keep;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| Reg_no | varchar(8) | NO   | MUL | NULL    |       |
| m_id   | varchar(8) | NO   | MUL | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

```
desc lecturing;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| user_id | varchar(8) | NO   | MUL | NULL    |       |
| c_code  | varchar(8) | NO   | MUL | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

```
DESC entroll;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Reg_no       | varchar(8) | NO   | MUL | NULL    |       |
| c_code       | varchar(8) | NO   | MUL | NULL    |       |
| entroll_date | date        | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

```
DESC attendence;
+-------------------+-----------------------------+------+-----+---------+-------+
| Field             | Type                        | Null | Key | Default | Extra |
+-------------------+-----------------------------+------+-----+---------+-------+
| att_id            | varchar(8)                  | NO   | PRI | NULL    |       |
| type              | enum('theory','practical')  | NO   |     | theory  |       |
| atten_date        | date                        | NO   |     | NULL    |       |
| status            | enum('absent','Present')    | NO   |     | Present |       |
| change_status     | enum('absent','Present')    | NO   |     | Present |       |
| Is_medical_affect | enum('yes','no')            | NO   |     | no      |       |
| Reg_no            | varchar(8)                  | YES  | MUL | NULL    |       |
| session_id        | varchar(8)                  | YES  | MUL | NULL    |       |
| session_no        | varchar(20)                 | YES  |     | NULL    |       |
+-------------------+-----------------------------+------+-----+---------+-------+
```

```
DESC medical;
+-----------------+-----------------------------------------+------+-----+---------+-------+
| Field           | Type                                    | Null | Key | Default | Extra |
+-----------------+-----------------------------------------+------+-----+---------+-------+
| medical_id      | varchar(8)                              | NO   | PRI | NULL    |       |
| c_code          | char(8)                                 | NO   | MUL | NULL    |       |
| reg_no          | varchar(8)                              | YES  | MUL | NULL    |       |
| submission_date | date                                    | NO   |     | NULL    |       |
| description     | text                                    | NO   |     | NULL    |       |
| affectted_date  | date                                    | YES  |     | NULL    |       |
| session_id      | varchar(8)                              | YES  | MUL | NULL    |       |
| status          | enum('approved','not approved','pending') | NO   |     | pending |       |
| att_id          | varchar(8)                              | YES  | MUL | NULL    |       |
+-----------------+-----------------------------------------+------+-----+---------+-------+
```

```
DESC timetable;
+--------------+-----------------------------------------------------------+------+-----+---------+-------+
| Field        | Type                                                      | Null | Key | Default | Extra |
+--------------+-----------------------------------------------------------+------+-----+---------+-------+
| session_id   | varchar(8)                                                | NO   | PRI | NULL    |       |
| session_date | enum('Monday','Tuesday','Wednesday','Thursday','Friday')  | NO   |     | NULL    |       |
| type         | enum('PRACTICAL','THEORY')                                | NO   |     | THEORY  |       |
| duration     | time                                                      | NO   |     | NULL    |       |
| lec_hall     | varchar(10)                                               | NO   |     | NULL    |       |
| c_code       | varchar(8)                                                | YES  | MUL | NULL    |       |
+--------------+-----------------------------------------------------------+------+-----+---------+-------+
```

```
DESC end_marks;
+-----------+--------------------------------------+------+-----+---------+-------+
| Field     | Type                                 | Null | Key | Default | Extra |
+-----------+--------------------------------------+------+-----+---------+-------+
| m_id      | varchar(8)                           | NO   | PRI | NULL    |       |
| theory    | varchar(10)                          | YES  |     | NULL    |       |
| practical | varchar(10)                          | YES  |     | NULL    |       |
| c_code    | varchar(8)                           | YES  | MUL | NULL    |       |
| reg_no    | varchar(8)                           | YES  | MUL | NULL    |       |
| status    | enum('proper','repeat','suspended')  | NO   |     | proper  |       |
+-----------+--------------------------------------+------+-----+---------+-------+
```

```
DESC mid_masks;
+-----------+-------------------------------------+------+-----+---------+-------+
| Field     | Type                                | Null | Key | Default | Extra |
+-----------+-------------------------------------+------+-----+---------+-------+
| m_id      | varchar(8)                          | NO   | PRI | NULL    |       |
| theory    | varchar(10)                         | YES  |     | NULL    |       |
| practical | varchar(10)                         | YES  |     | NULL    |       |
| status    | enum('proper','repeat','suspended') | NO   |     | proper  |       |
| reg_no    | varchar(8)                          | YES  | MUL | NULL    |       |
| c_code    | varchar(8)                          | YES  | MUL | NULL    |       |
+-----------+-------------------------------------+------+-----+---------+-------+
```

```
DESC assesment_marks;
+-----------------+-------------------------------------+------+-----+---------+-------+
| Field           | Type                                | Null | Key | Default | Extra |
+-----------------+-------------------------------------+------+-----+---------+-------+
| m_id            | varchar(8)                          | NO   | PRI | NULL    |       |
| Assesment_marks | varchar(10)                         | YES  |     | NULL    |       |
| status          | enum('proper','repeat','suspended') | NO   |     | proper  |       |
| c_code          | varchar(8)                          | YES  | MUL | NULL    |       |
| reg_no          | varchar(8)                          | YES  | MUL | NULL    |       |
+-----------------+-------------------------------------+------+-----+---------+-------+
```

```
 DESC q3_marks;
+--------+-------------------------------------+------+-----+---------+-------+
| Field  | Type                                | Null | Key | Default | Extra |
+--------+-------------------------------------+------+-----+---------+-------+
| m_id   | varchar(8)                          | NO   | PRI | NULL    |       |
| q3     | varchar(10)                         | YES  |     | NULL    |       |
| status | enum('proper','repeat','suspended') | NO   |     | proper  |       |
| c_code | varchar(8)                          | YES  | MUL | NULL    |       |
| reg_no | varchar(8)                          | YES  | MUL | NULL    |       |
+--------+-------------------------------------+------+-----+---------+-------+
```

```
DESC q2_marks;
+--------+-------------------------------------+------+-----+---------+-------+
| Field  | Type                                | Null | Key | Default | Extra |
+--------+-------------------------------------+------+-----+---------+-------+
| m_id   | varchar(8)                          | NO   | PRI | NULL    |       |
| q2     | varchar(10)                         | YES  |     | NULL    |       |
| status | enum('proper','repeat','suspended') | NO   |     | proper  |       |
| c_code | varchar(8)                          | YES  | MUL | NULL    |       |
| reg_no | varchar(8)                          | YES  | MUL | NULL    |       |
+--------+-------------------------------------+------+-----+---------+-------+
```

```
DESC q1_marks;
+--------+---------------------------------------+------+-----+---------+-------+
| Field  | Type                                  | Null | Key | Default | Extra |
+--------+---------------------------------------+------+-----+---------+-------+
| m_id   | varchar(8)                            | NO   | PRI | NULL    |       |
| q1     | varchar(10)                           | YES  |     | NULL    |       |
| status | enum('proper','repeat','suspended')   | NO   |     | proper  |       |
| c_code | varchar(8)                            | YES  | MUL | NULL    |       |
| reg_no | varchar(8)                            | YES  | MUL | NULL    |       |
+--------+---------------------------------------+------+-----+---------+-------+
```

```
MariaDB [fot]> DESC marks;
+--------------+-----------------------------------------------------------------------+------+-----+---------+-------+
| Field        | Type                                                                  | Null | Key | Default | Extra |
+--------------+-----------------------------------------------------------------------+------+-----+---------+-------+
| m_id         | varchar(8)                                                            | NO   | PRI | NULL    |       |
| Entered_date | date                                                                  | NO   |     | NULL    |       |
| Reg_no       | char(8)                                                               | YES  | MUL | NULL    |       |
| is_CA        | enum('YES','NO')                                                      | NO   |     | YES     |       |
| st_status    | enum('proper','CA repeat','End Repeat','Both repeat','suspended')     | NO   |     | NULL    |       |
+--------------+-----------------------------------------------------------------------+------+-----+---------+-------+
```

```
DESC course_unit;
+--------------+------------------+------+-----+---------+-------+
| Field        | Type             | Null | Key | Default | Extra |
+--------------+------------------+------+-----+---------+-------+
| c_code       | char(8)          | NO   | PRI | NULL    |       |
| c_name       | varchar(50)      | NO   |     | NULL    |       |
| credit       | int(11)          | NO   |     | NULL    |       |
| is_theory    | enum('YES','NO') | NO   |     | YES     |       |
| is_practicel | enum('YES','NO') | NO   |     | YES     |       |
| semester     | char(4)          | YES  |     | NULL    |       |
+--------------+------------------+------+-----+---------+-------+
```

```
DESC department;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| dept_id | char(8)      | NO   | PRI | NULL    |       |
| d_name  | varchar(100) | NO   |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
```

```
DESC batch;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| b_id          | char(8)      | NO   | PRI | NULL    |       |
| b_name        | varchar(100) | NO   |     | NULL    |       |
| acedemic_year | year(4)      | NO   |     | NULL    |       |
| start_date    | date         | NO   |     | NULL    |       |
| end_date      | date         | NO   |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
```

```
DESC student;
+--------+--------------------------------------+------+-----+---------+-------+
| Field  | Type                                 | Null | Key | Default | Extra |
+--------+--------------------------------------+------+-----+---------+-------+
| Reg_no | char(8)                              | NO   | PRI | NULL    |       |
| status | enum('PROPER','REPEAT','SUSPENDED') | NO   |     | PROPER  |       |
| b_id   | varchar(8)                           | YES  | MUL | NULL    |       |
+--------+--------------------------------------+------+-----+---------+-------+
```

```
DESC lecture;
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
| Field              | Type                                                            | Null | Key | Default | Extra |
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
| user_id            | char(8)                                                         | NO   | PRI | NULL    |       |
| appointment_of_date | date                                                           | NO   |     | NULL    |       |
| status             | enum('ACTIVE','INACTIVE','RETIRED','TRANSFERRED','TERMINATED') | NO   |     | ACTIVE  |       |
| position           | varchar(100)                                                    | NO   |     | NULL    |       |
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
```

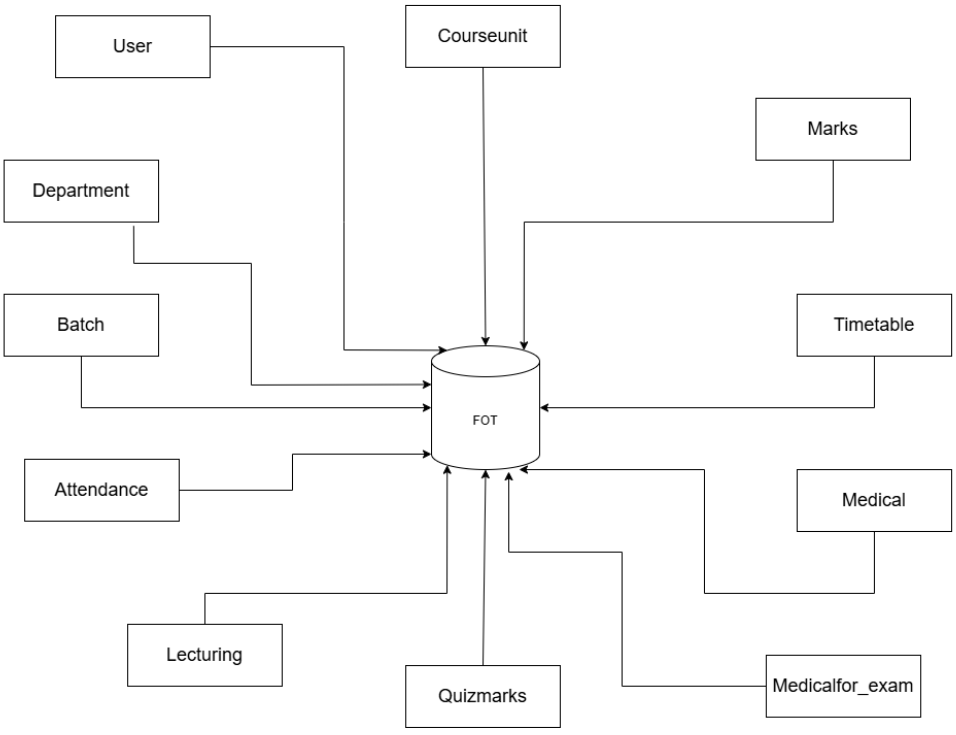
```
DESC tos;
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
| Field              | Type                                                            | Null | Key | Default | Extra |
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
| user_id            | char(8)                                                         | NO   | PRI | NULL    |       |
| appointment_of_date | date                                                           | NO   |     | NULL    |       |
| status             | enum('ACTIVE','INACTIVE','RETIRED','TRANSFERRED','TERMINATED') | NO   |     | ACTIVE  |       |
| Duty_of_Area       | varchar(100)                                                    | NO   |     | NULL    |       |
| position           | varchar(100)                                                    | NO   |     | NULL    |       |
| shift_time         | time                                                            | NO   |     | NULL    |       |
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
```

```
DESC dean;
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
| Field              | Type                                                            | Null | Key | Default | Extra |
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
| user_id            | char(8)                                                         | NO   | PRI | NULL    |       |
| appointment_of_date | date                                                           | NO   |     | NULL    |       |
| status             | enum('ACTIVE','INACTIVE','RETIRED','TRANSFERRED','TERMINATED') | NO   |     | ACTIVE  |       |
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
```

```
DESC admin;
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
| Field              | Type                                                            | Null | Key | Default | Extra |
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
| user_id            | char(8)                                                         | NO   | PRI | NULL    |       |
| appointment_of_date | date                                                           | NO   |     | NULL    |       |
| Access_level       | varchar(50)                                                     | NO   |     | NULL    |       |
| status             | enum('ACTIVE','INACTIVE','RETIRED','TRANSFERRED','TERMINATED') | NO   |     | ACTIVE  |       |
+--------------------+-----------------------------------------------------------------+------+-----+---------+-------+
```

```
DESC user;
+--------------+----------------+------+-----+---------+-------+
| Field        | Type           | Null | Key | Default | Extra |
+--------------+----------------+------+-----+---------+-------+
| user_id      | char(8)        | NO   | PRI | NULL    |       |
| F_name       | varchar(100)   | NO   |     | NULL    |       |
| L_name       | varchar(100)   | NO   |     | NULL    |       |
| Address      | varchar(100)   | NO   |     | NULL    |       |
| phone_number | int(11)        | NO   |     | NULL    |       |
| dob          | date           | NO   |     | NULL    |       |
| email        | varchar(100)   | NO   |     | NULL    |       |
| gender       | enum('M','F') | NO   |     | NULL    |       |
| dept_id      | varchar(8)     | YES  | MUL | NULL    |       |
+--------------+----------------+------+-----+---------+-------+
```

6. Architecture of Solution



7. Tools and Technologies

- MySQL
- XAMPP
- Visual Code & Notepad ++

8. Security measures that we have taken to protect our DB

- **Data Type Validation (varchar, INT**) : Prevents storage errors and type-based injection vulnerabilities.
- **Fixed Length for IDs (CHAR 8)** : Ensures consistency, which can aid in validation logic and prevent truncation attacks.
- **Validate Text Length (VARCHAR 100**) : Prevents buffer overflow issues and overly large, unusable data inputs.
- **Use Triggers** : Can enforce complex business rules and automate auditing (e.g., logging every update to a sensitive table).
- **Mark Validation using CHECK** : Enforces critical business constraints (e.g., a score must be between 0 and 100).\

9. Brief Description about our FOT database accounts, users, and the reasons for creating

In our Faculty of Technology (FoT) database system, several MySQL user accounts were created to ensure proper access control, maintain data integrity, and provide secure, role-based access to students, staff, and administrators. Each account is assigned specific privileges according to its role within the university system:

- **Admin**
  Privileges: Full access to all tables, views, and the ability to create, modify, or delete users (GRANT OPTION).

  Purpose: The Admin oversees the entire database, manages user accounts, and maintains the system. This role ensures centralized control and security.

- **Dean**
  Privileges: All table operations without the ability to create or grant users.

  Purpose: The Dean can review and manage all academic records, attendance, and marks, ensuring that decision-making is informed by accurate data while limiting administrative privileges.

- **Lecturer**
  Privileges: Full access to all tables for operational purposes but cannot create users or grant privileges.

  Purpose: Lecturers can enter and update attendance records, grades, and other student information required for their courses, without compromising system security.

- **Technical Officer**
  Privileges: Read, write, and update access limited to attendance-related tables and views.

  Purpose: Technical officers manage attendance data, verify submissions, and assist in maintaining accurate records without access to sensitive exam marks or grades.

- **Student**
  Privileges: Read-only access to final attendance and final marks/grades tables and views.

  Purpose: Students can view their attendance percentages, medical submissions, and final grades, promoting transparency and academic awareness without altering records.

  Reason for Creating Separate Accounts:

- Role-based access ensures security, data integrity, and controlled access to sensitive academic information.

- Prevents unauthorized changes to attendance or exam records.

- Allows system automation and auditing while ensuring that each user can perform only the actions relevant to their responsibilities.

10. Code snippets to Support our work

//Dinesha's procedure views and trigers

//Views Here

```
create view all_attendence_with_precentage AS
select
a.reg_no,u.F_name,u.L_name,t.c_code,c.c_name,a.type,round(avg(a.status='Present')
*100) AS att_pecentage
from attendence a
INNER JOIN timetable t on a.session_id=t.session_id
INNER JOIN course_unit c on t.c_code=c.c_code
INNER JOIN user u ON a.reg_no=u.user_id
group by t.c_code,a.reg_no,a.type;

create view attendance_above_80 AS
select
a.reg_no,u.F_name,u.L_name,t.c_code,c.c_name,a.type,round(avg(a.status='Present')
*100) AS att_pecentage
from attendence a
INNER JOIN timetable t on a.session_id=t.session_id
INNER JOIN course_unit c on t.c_code=c.c_code
INNER JOIN user u ON a.reg_no=u.user_id
group by t.c_code,a.reg_no,a.type
having att_pecentage>=80;

create view attendance_below_80 AS
select
a.reg_no,u.F_name,u.L_name,t.c_code,c.c_name,a.type,round(avg(a.status='Present')
*100) AS att_pecentage
from attendence a
INNER JOIN timetable t on a.session_id=t.session_id
INNER JOIN course_unit c on t.c_code=c.c_code
INNER JOIN user u ON a.reg_no=u.user_id
group by t.c_code,a.reg_no,a.type
having att_pecentage<80;
```

//3 Views

```
create view medical_percentage_below_80 AS
select
a.reg_no,u.F_name,u.L_name,t.c_code,c.c_name,a.type,round(avg(a.change_status='P
resent')*100) AS attendence_percentage
from attendence a
```

```
INNER JOIN timetable t on a.session_id=t.session_id
INNER JOIN course_unit c on t.c_code=c.c_code
INNER JOIN user u ON a.reg_no=u.user_id
group by t.c_code,a.reg_no,a.type
having attendence_percentage<80;

create view medical_percentage_above_80 AS
select
a.reg_no,u.F_name,u.L_name,t.c_code,c.c_name,a.type,round(avg(a.change_status='P
resent')*100) AS attendence_percentage
from attendence a
INNER JOIN timetable t on a.session_id=t.session_id
INNER JOIN course_unit c on t.c_code=c.c_code
INNER JOIN user u ON a.reg_no=u.user_id
group by t.c_code,a.reg_no,a.type
having attendence_percentage>=80;

create view all_attendence_with_medical AS
select
a.reg_no,u.F_name,u.L_name,t.c_code,c.c_name,a.type,round(avg(a.status='Present')
*100) AS attendence_percentage,round(avg(a.change_status='Present')*100) AS
attendence_percentage_with_medical
from attendence a
INNER JOIN timetable t on a.session_id=t.session_id
INNER JOIN course_unit c on t.c_code=c.c_code
INNER JOIN user u ON a.reg_no=u.user_id
group by t.c_code,a.reg_no,a.type
ORDER BY a.reg_no asc;



//trigers for 2 limit

DELIMITER //
CREATE TRIGGER trg_medical_limit_insert
BEFORE INSERT ON medical
FOR EACH ROW
BEGIN
   DECLARE med_count INT;
   IF NEW.status!='not approved' THEN
      SELECT COUNT(*)
      INTO med_count
      FROM medical
      WHERE reg_no = NEW.reg_no
      AND session_id = NEW.session_id
      AND c_code=NEW.c_code
```

```
        AND status IN ('approved', 'pending');
        IF med_count >= 2 THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Error: Student already has 2 approved or pending
medicals for this session';
        END IF;
    END IF;
END//

DELIMITER ;

//triger for change status

DELIMITER //

CREATE TRIGGER AFFECT_MEDICAL_for_attendence
AFTER INSERT ON medical
FOR EACH ROW
BEGIN
    IF NEW.status = 'approved' THEN
        UPDATE attendence
        SET change_status = 'Present',
            Is_medical_affect = 'yes'
        WHERE reg_no = NEW.reg_no
          AND session_id = NEW.session_id
          AND atten_date = NEW.affectted_date;
    END IF;
END//
DELIMITER ;



//update trigers

DELIMITER //

CREATE TRIGGER AFFECT_MEDICAL_for_attendence_update_table
AFTER update ON medical
FOR EACH ROW
BEGIN
    IF NEW.status = 'approved' AND OLD.status <> 'approved' THEN
        UPDATE attendence
        SET change_status = 'Present',
            Is_medical_affect = 'yes'
        WHERE reg_no = NEW.reg_no
          AND session_id = NEW.session_id
```

```
        AND atten_date = NEW.affectted_date;
    END IF;
END//
DELIMITER ;
```

```
//medical_for_exam

DELIMITER //
CREATE TRIGGER update_marks_base_on_medical
AFTER INSERT ON medical_for_exam
FOR EACH ROW
BEGIN

    IF NEW.status='approved' THEN
        IF NEW.marks_type = 'q1_marks' THEN
            UPDATE q1_marks
            SET q1 = 'MC'
            WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;

        ELSEIF NEW.marks_type = 'q2_marks' THEN
            UPDATE q2_marks
            SET q2 = 'MC'
            WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;

        ELSEIF NEW.marks_type = 'q3_marks' THEN
            UPDATE q3_marks
            SET q3 = 'MC'
            WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;

        ELSEIF NEW.marks_type = 'assesment_marks' THEN
            UPDATE assesment_marks
            SET Assesment_marks = 'MC'
            WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;

        ELSEIF NEW.marks_type = 'mid_masks' THEN
            IF NEW.is_practical_or_theory = 'practical' THEN
                UPDATE mid_masks
                SET practical = 'MC'
                WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;
            ELSEIF NEW.is_practical_or_theory = 'theory' THEN
                UPDATE mid_masks
                SET theory = 'MC'
```

```
            WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;
        END IF;

      ELSEIF NEW.marks_type = 'end_marks' THEN
        IF NEW.is_practical_or_theory = 'practical' THEN
          UPDATE end_marks
          SET practical = 'MC'
          WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;
        ELSEIF NEW.is_practical_or_theory = 'theory' THEN
          UPDATE end_marks
          SET theory = 'MC'
          WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;
        END IF;
      END IF;
  END IF;
END //
DELIMITER ;


//tiger throw
DELIMITER //
CREATE TRIGGER update_marks_base_on_medical_update
AFTER UPDATE ON medical_for_exam
FOR EACH ROW
BEGIN
  -- Run only when status becomes "approved"
  IF NEW.status = 'approved' AND OLD.status != 'approved' THEN

    IF NEW.marks_type = 'q1_marks' THEN
      UPDATE q1_marks
      SET q1 = 'MC'
      WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;

    ELSEIF NEW.marks_type = 'q2_marks' THEN
      UPDATE q2_marks
      SET q2 = 'MC'
      WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;

    ELSEIF NEW.marks_type = 'q3_marks' THEN
      UPDATE q3_marks
      SET q3 = 'MC'
      WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;

    ELSEIF NEW.marks_type = 'assesment_marks' THEN
      UPDATE assesment_marks
      SET Assesment_marks = 'MC'
```

```
            WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;

        ELSEIF NEW.marks_type = 'mid_masks' THEN
            IF NEW.is_practical_or_theory = 'practical' THEN
                UPDATE mid_masks
                SET practical = 'MC'
                WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;
            ELSEIF NEW.is_practical_or_theory = 'theory' THEN
                UPDATE mid_masks
                SET theory = 'MC'
                WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;
            END IF;

        ELSEIF NEW.marks_type = 'end_marks' THEN
            IF NEW.is_practical_or_theory = 'practical' THEN
                UPDATE end_marks
                SET practical = 'MC'
                WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;
            ELSEIF NEW.is_practical_or_theory = 'theory' THEN
                UPDATE end_marks
                SET theory = 'MC'
                WHERE reg_no = NEW.reg_no AND c_code = NEW.c_code;
            END IF;
        END IF;
    END IF;
END //
DELIMITER ;


//procedure for attendence eligibility

DELIMITER //

CREATE PROCEDURE GetStudentAttendance(IN p_reg_no VARCHAR(8))
BEGIN
    SELECT
        a.reg_no,
        u.F_name,
        u.L_name,
        t.c_code,
        c.c_name,
        a.session_id,
        a.type,
        ROUND(AVG(a.change_status='Present')*100) AS attendence_percentage,
        ROUND(AVG(a.change_status='Present')*100) AS
attendence_percentage_with_medical,
```

```
        CASE
            WHEN ROUND(AVG(a.change_status='Present')*100) >= 80 THEN 'eligible'
            ELSE 'not eligible'
        END AS is_eligible
    FROM attendence a
    INNER JOIN timetable t ON a.session_id = t.session_id
    INNER JOIN course_unit c ON t.c_code = c.c_code
    INNER JOIN user u ON a.reg_no = u.user_id
    WHERE a.reg_no = p_reg_no
    GROUP BY a.reg_no, t.c_code, a.type;
END //

DELIMITER ;

call GetStudentAttendance("TCH/1010");

DELIMITER //

CREATE PROCEDURE GetStudentAttendance(IN p_c_code VARCHAR(8))
BEGIN
    SELECT
        t.c_code,
        a.reg_no,
        count(a.change_status = 'Present') AS dates,
        ROUND(AVG(a.change_status = 'Present') * 100) AS attendance_percentage,
        CASE
            WHEN ROUND(AVG(a.change_status = 'Present') * 100) >= 80 THEN
'eligible'
            ELSE 'not eligible'
        END AS is_eligible
    FROM attendence a
    INNER JOIN timetable t ON a.session_id = t.session_id
    WHERE t.c_code = p_c_code
    GROUP BY a.reg_no,t.c_code;
END //

DELIMITER ;

drop procedure GetStudentAttendance;

CALL GetStudentAttendance('TMS1233');

select * from attendence
where session_no like("ict1253%");
```

```
DELIMITER //

CREATE OR REPLACE PROCEDURE GetStudentAttendance(IN p_c_code
VARCHAR(8))
BEGIN
   SELECT
      t.c_code,
      a.reg_no,

      -- Present days (rounded to whole number)
      CASE
         WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
         THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
         ELSE SUM(a.change_status = 'Present')
      END AS present_days,

      -- Total sessions (fixed properly)
      CASE
         WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
         THEN 15
         ELSE 15
      END AS total_sessions,


      ROUND(
         (
            CASE
               WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
               THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
               ELSE SUM(a.change_status = 'Present')
            END
            /
            CASE
               WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
               THEN 15
               ELSE 15
            END
         ) * 100,
      0) AS attendance_percentage,

      -- Eligibility
      CASE
         WHEN
            ROUND(
               (
                  CASE
```

```
                    WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
                    THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
                    ELSE SUM(a.change_status = 'Present')
                END
                /
                CASE
                    WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
                    THEN 15
                    ELSE 15
                END
            ) * 100,
        0) >= 80
        THEN 'eligible'
        ELSE 'not eligible'
    END AS is_eligible

  FROM attendence a
  INNER JOIN timetable t ON a.session_id = t.session_id
  WHERE t.c_code = p_c_code
  GROUP BY a.reg_no, t.c_code;
END //

DELIMITER ;


create view All_subject_with_Eligibility AS
SELECT
  t.c_code,
  c.c_name,
  a.type,
  a.reg_no,
  u.F_name,
  u.L_name,
  ROUND(AVG(a.status = 'Present') * 100) AS attendence_percentage,
  ROUND(AVG(a.change_status = 'Present') * 100) AS
attendence_percentage_with_medical,

  CASE
    WHEN ROUND(AVG(a.change_status = 'Present') * 100) >= 80
    THEN 'eligible'
    ELSE 'not eligible'
  END AS eligibility

FROM attendence a
INNER JOIN timetable t ON a.session_id = t.session_id
INNER JOIN course_unit c ON t.c_code = c.c_code
```

```
INNER JOIN user u ON a.reg_no = u.user_id
GROUP BY t.c_code, a.reg_no, a.type
ORDER BY t.c_code ASC;


DELIMITER //

CREATE OR REPLACE PROCEDURE GetStudentAttendanceIndividuals(IN
p_reg_no VARCHAR(8))
BEGIN
   SELECT
      t.c_code,
      a.reg_no,

      -- Present days (rounded to whole number)
      CASE
         WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
         THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
         ELSE SUM(a.change_status = 'Present')
      END AS present_days,

      -- Total sessions (fixed properly)
      CASE
         WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
         THEN 15
         ELSE 15
      END AS total_sessions,

      ROUND(
         (
            CASE
               WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
               THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
               ELSE SUM(a.change_status = 'Present')
            END
            /
            CASE
               WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
               THEN 15
               ELSE 15
            END
         ) * 100,
      0) AS attendance_percentage,

      -- Eligibility
      CASE
```

```
        WHEN
          ROUND(
            (
              CASE
                WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
                THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
                ELSE SUM(a.change_status = 'Present')
              END
              /
              CASE
                WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
                THEN 15
                ELSE 15
              END
            ) * 100,
          0) >= 80
        THEN 'eligible'
        ELSE 'not eligible'
      END AS is_eligible

  FROM attendence a
  INNER JOIN timetable t ON a.session_id = t.session_id
  WHERE a.reg_no = p_reg_no
  GROUP BY a.reg_no, t.c_code;
END //

DELIMITER ;



DELIMITER //

CREATE OR REPLACE PROCEDURE
GetAttendanceWith_Reg_no_and_c_code(IN p_reg_no VARCHAR(8),IN p_c_code
varchar(8))
BEGIN
  SELECT
    t.c_code,
    a.reg_no,

    -- Present days (rounded to whole number)
    CASE
      WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
      THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
      ELSE SUM(a.change_status = 'Present')
    END AS present_days,
```

```
-- Total sessions (fixed properly)
CASE
   WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
   THEN 15
   ELSE 15
END AS total_sessions,


ROUND(
   (
      CASE
         WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
         THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
         ELSE SUM(a.change_status = 'Present')
      END
      /
      CASE
         WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
         THEN 15
         ELSE 15
      END
   ) * 100,
0) AS attendance_percentage,
-- Eligibility
CASE
   WHEN
      ROUND(
         (
            CASE
               WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
               THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
               ELSE SUM(a.change_status = 'Present')
            END
            /
            CASE
               WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
               THEN 15
               ELSE 15
            END
         ) * 100,
      0) >= 80
   THEN 'eligible'
   ELSE 'not eligible'
END AS is_eligible
```

```
    FROM attendence a
    INNER JOIN timetable t ON a.session_id = t.session_id
    WHERE a.reg_no = p_reg_no and where t.c_code=p_c_code
    GROUP BY a.reg_no, t.c_code;
END //

DELIMITER ;


DELIMITER //

CREATE OR REPLACE PROCEDURE GetStudentAttendance(IN p_reg_no
VARCHAR(8),IN p_c_code VARCHAR(8),IN is_what VARCHAR(20))
BEGIN
    SELECT
        t.c_code,
        a.reg_no,

        -- Present days (rounded to whole number)
        CASE
            WHEN t.c_code IN ('ICT1233', 'ICT1253')
            THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
            ELSE SUM(a.change_status = 'Present')
        END AS present_days,

        -- Total sessions (fixed properly)
        CASE
            WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
            THEN 15
            ELSE 15
        END AS total_sessions,


        ROUND(
            (
                CASE
                    WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
                    THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
                    ELSE SUM(a.change_status = 'Present')
                END
                /
                CASE
                    WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
                    THEN 15
                    ELSE 15
                END
```

```
        ) * 100,
    0) AS attendance_percentage,

    -- Eligibility
    CASE
      WHEN
        ROUND(
          (
            CASE
              WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
              THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
              ELSE SUM(a.change_status = 'Present')
            END
            /
            CASE
              WHEN t.c_code IN ('ICT1233', 'ICT1253','ICT1222')
              THEN 15
              ELSE 15
            END
          ) * 100,
        0) >= 80
      THEN 'eligible'
      ELSE 'not eligible'
    END AS is_eligible

  FROM attendence a
  INNER JOIN timetable t ON a.session_id = t.session_id
  WHERE t.c_code = p_c_code
  GROUP BY a.reg_no, t.c_code;
END //

DELIMITER ;

DELIMITER //

CREATE OR REPLACE PROCEDURE theory_and_practical_subject_attendence(
  IN p_reg_no VARCHAR(8),
  IN p_c_code VARCHAR(8),
  IN is_what VARCHAR(20)
)
BEGIN
  SELECT
    t.c_code,
    a.reg_no,

    -- Present days
```

```
CASE
   WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
   THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
   ELSE SUM(a.change_status = 'Present')
END AS present_days,

-- Total sessions
CASE
   WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
   THEN 15
   ELSE 15
END AS total_sessions,

-- Attendance percentage (rounded)
ROUND(
   (
      CASE
         WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
         THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
         ELSE SUM(a.change_status = 'Present')
      END
      /
      CASE
         WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
         THEN 15
         ELSE 15
      END
   ) * 100,
0) AS attendance_percentage,

-- Eligibility
CASE
   WHEN
      ROUND(
         (
            CASE
               WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
               THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
               ELSE SUM(a.change_status = 'Present')
            END
            /
            CASE
               WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
               THEN 15
               ELSE 15
            END
```

```
            ) * 100,
          0) >= 80
        THEN 'eligible'
        ELSE 'not eligible'
    END AS is_eligible

  FROM attendence a
  INNER JOIN timetable t ON a.session_id = t.session_id
  WHERE a.reg_no = p_reg_no
    AND t.c_code = p_c_code
    AND (is_what = 'all' OR a.type = is_what)  -- Filter by theory/practical/all
  GROUP BY a.reg_no, t.c_code;
END //

DELIMITER ;



DELIMITER //

CREATE OR REPLACE PROCEDURE theory_and_practical_by_course_code(
  IN p_c_code VARCHAR(8),
  IN is_what VARCHAR(20)
)
BEGIN
  SELECT
    t.c_code,
    a.reg_no,

    -- Present days
    CASE
      WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
      THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
      ELSE SUM(a.change_status = 'Present')
    END AS present_days,

    -- Total sessions
    CASE
      WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
      THEN 15
      ELSE 15
    END AS total_sessions,

    -- Attendance percentage (rounded)
    ROUND(
      (
```

```
            CASE
               WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
               THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
               ELSE SUM(a.change_status = 'Present')
            END
            /
            CASE
               WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
               THEN 15
               ELSE 15
            END
         ) * 100,
      0) AS attendance_percentage,

      -- Eligibility
      CASE
         WHEN
            ROUND(
               (
                  CASE
                     WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
                     THEN ROUND(SUM(a.change_status = 'Present') / 2, 0)
                     ELSE SUM(a.change_status = 'Present')
                  END
                  /
                  CASE
                     WHEN t.c_code IN ('ICT1233', 'ICT1253') AND is_what = 'all'
                     THEN 15
                     ELSE 15
                  END
               ) * 100,
            0) >= 80
         THEN 'eligible'
         ELSE 'not eligible'
      END AS is_eligible

   FROM attendence a
   INNER JOIN timetable t ON a.session_id = t.session_id
   WHERE t.c_code = p_c_code
     AND (is_what = 'all' OR a.type = is_what)  -- Filter by theory/practical/all
   GROUP BY a.reg_no, t.c_code;
END //

DELIMITER ;
```

```
// Bashtha’s code here
CREATE OR REPLACE VIEW final_grades AS
SELECT
   sub.*,

   CASE
      WHEN sub.q1 = 'WH' OR sub.q2 = 'WH' OR sub.q3 = 'WH'
         OR sub.mid_theory = 'WH' OR sub.mid_practical = 'WH'
         OR sub.assessment = 'WH' THEN 'WH'

      WHEN sub.q1 = 'MC' OR sub.q2 = 'MC' OR sub.q3 = 'MC'
         OR sub.mid_theory = 'MC' OR sub.mid_practical = 'MC'
         OR sub.assessment = 'MC' OR sub.end_theory = 'MC' OR sub.end_practical = 'MC'
THEN 'MC'

      WHEN LOWER(sub.status) = 'both repeat' THEN
        CASE
           WHEN sub.final_marks >= 45 THEN 'C'
           WHEN sub.final_marks < 45 AND sub.final_marks >= 40 THEN 'C-'
           WHEN sub.final_marks < 40 AND sub.final_marks >= 35 THEN 'D'
           ELSE 'E'
        END

      ELSE -- proper students or suspended
        CASE
           WHEN sub.final_marks >= 85 THEN 'A+'
           WHEN sub.final_marks >= 75 THEN 'A'
           WHEN sub.final_marks >= 70 THEN 'A-'
           WHEN sub.final_marks >= 65 THEN 'B+'
           WHEN sub.final_marks >= 60 THEN 'B'
           WHEN sub.final_marks >= 55 THEN 'B-'
```

```
            WHEN sub.final_marks >= 50 THEN 'C+'
            WHEN sub.final_marks >= 45 THEN 'C'
            WHEN sub.final_marks >= 40 THEN 'C-'
            WHEN sub.final_marks >= 35 THEN 'D'
            ELSE 'E'
        END
    END AS grade

FROM
(
    SELECT
        q1.reg_no,
        q1.c_code,

        CASE
            WHEN q1.q1 = 'WH' OR q2.q2 = 'WH' OR q3.q3 = 'WH'
                OR m.theory = 'WH' OR m.practical = 'WH'
                OR a.Assesment_marks = 'WH' THEN 'suspended'
            ELSE mk.st_status
        END AS status,

        q1.q1,
        q2.q2,
        q3.q3,
        a.Assesment_marks AS assessment,
        m.theory AS mid_theory,
        m.practical AS mid_practical,
        e.theory AS end_theory,
        e.practical AS end_practical,
```

```
    ROUND(
    CASE
      WHEN q1.c_code IN ('ICT1242','TCS1212','ICT1212','ENG1222',
                         'TMS1233','ICT1222','ICT1233','ICT1253') THEN
        (((q1.q1 + q2.q2 + q3.q3) - LEAST(q1.q1, q2.q2, q3.q3)) / 2) * 0.1
      ELSE 0
    END, 2) AS quiz_component,

    ROUND(
    CASE
      WHEN q1.c_code IN ('ICT1242','TCS1212','ICT1212','ENG1222') THEN m.theory *
0.15
      WHEN q1.c_code = 'TMS1233' THEN m.theory * 0.25
      WHEN q1.c_code = 'ICT1222' THEN m.practical * 0.25
      WHEN q1.c_code IN ('ICT1233','ICT1253') THEN (m.theory + m.practical) * 0.25 /
2
      ELSE 0
    END, 2) AS mid_component,

    ROUND(a.Assesment_marks * 0.05, 2) AS assessment_component,

    ROUND(
    CASE
      WHEN q1.c_code IN ('ICT1242','TCS1212','ICT1212','ENG1222') THEN e.theory *
0.7
      WHEN q1.c_code = 'TMS1233' THEN e.theory * 0.6
      WHEN q1.c_code = 'ICT1222' THEN e.practical * 0.6
      WHEN q1.c_code IN ('ICT1233','ICT1253') THEN e.theory * 0.4 + e.practical * 0.2
      ELSE 0
    END, 2) AS end_component,
```

```
    ROUND(
      (((q1.q1 + q2.q2 + q3.q3) - LEAST(q1.q1, q2.q2, q3.q3)) / 2) * 0.1
      + CASE
        WHEN q1.c_code IN ('ICT1242','TCS1212','ICT1212','ENG1222') THEN m.theory
* 0.15
        WHEN q1.c_code = 'TMS1233' THEN m.theory * 0.25
        WHEN q1.c_code = 'ICT1222' THEN m.practical * 0.25
        WHEN q1.c_code IN ('ICT1233','ICT1253') THEN (m.theory + m.practical) * 0.25
/ 2
        ELSE 0
      END
      + a.Assesment_marks * 0.05
      + CASE
        WHEN q1.c_code IN ('ICT1242','TCS1212','ICT1212','ENG1222') THEN e.theory
* 0.7
        WHEN q1.c_code = 'TMS1233' THEN e.theory * 0.6
        WHEN q1.c_code = 'ICT1222' THEN e.practical * 0.6
        WHEN q1.c_code IN ('ICT1233','ICT1253') THEN e.theory * 0.4 + e.practical *
0.2
        ELSE 0
      END
    , 2) AS final_marks

  FROM q1_marks q1
  LEFT JOIN (
    SELECT reg_no, c_code, MAX(q2) AS q2 FROM q2_marks GROUP BY reg_no,
c_code
  ) q2 ON q1.reg_no = q2.reg_no AND q1.c_code = q2.c_code
  LEFT JOIN (
    SELECT reg_no, c_code, MAX(q3) AS q3 FROM q3_marks GROUP BY reg_no,
c_code
  ) q3 ON q1.reg_no = q3.reg_no AND q1.c_code = q3.c_code
```

```
    LEFT JOIN (
        SELECT reg_no, c_code, MAX(theory) AS theory, MAX(practical) AS practical
        FROM mid_masks GROUP BY reg_no, c_code
    ) m ON q1.reg_no = m.reg_no AND q1.c_code = m.c_code
    LEFT JOIN (
        SELECT reg_no, c_code, MAX(Assesment_marks) AS Assesment_marks
        FROM assesment_marks GROUP BY reg_no, c_code
    ) a ON q1.reg_no = a.reg_no AND q1.c_code = a.c_code
    LEFT JOIN (
        SELECT reg_no, c_code, MAX(theory) AS theory, MAX(practical) AS practical
        FROM end_marks GROUP BY reg_no, c_code
    ) e ON q1.reg_no = e.reg_no AND q1.c_code = e.c_code
    LEFT JOIN (
        SELECT Reg_no, MAX(st_status) AS st_status
        FROM marks GROUP BY Reg_no
    ) mk ON q1.reg_no = mk.Reg_no
) AS sub

WHERE LOWER(sub.status) IN ('proper','both repeat','suspended');



//3 column extracet

CREATE OR REPLACE VIEW final_grades_Only AS
SELECT
    reg_no,
    c_code,
    grade
FROM final_grades;
```

//3 column extracet prodedure

```
DELIMITER $$

CREATE PROCEDURE get_student_grades(IN p_reg_no CHAR(8))
BEGIN
   SELECT reg_no, c_code, grade
   FROM final_grades
   WHERE reg_no = p_reg_no;
END $$

DELIMITER ;
```

//3 column extracet prodedure

```
DELIMITER $$

CREATE PROCEDURE get_course_grades(IN p_c_code VARCHAR(10))
BEGIN
   SELECT reg_no, c_code, grade
   FROM final_grades
   WHERE c_code = p_c_code;
END $$

DELIMITER ;
```

/3 column extracet

```
CREATE OR REPLACE VIEW final_marks_Only AS
SELECT
   reg_no,
   c_code,
   final_marks
FROM final_grades;



/3 column extracet prodedure


DELIMITER $$


CREATE OR REPLACE PROCEDURE get_course_finalmarks(IN p_reg_no
VARCHAR(10))
BEGIN
   SELECT reg_no, c_code, grade
   FROM final_grades
   WHERE reg_no = p_reg_no;
END $$


DELIMITER ;


//3 column extracet prodedure
DELIMITER $$


CREATE OR REPLACE PROCEDURE get_course_finalmarks(IN p_c_code
VARCHAR(10))
BEGIN
   SELECT reg_no, c_code, grade
   FROM final_grades
   WHERE c_code = p_c_code;
```

```
END $$

DELIMITER ;
```

Final procedure in calculating ca results - TG/2023/1726

```
DELIMITER //

CREATE PROCEDURE calculate_ca_marks_by_student(
    IN p_reg_no VARCHAR(8),
    IN p_c_code VARCHAR(8)
)
BEGIN
    DECLARE q1 DECIMAL(5,2) DEFAULT 0;
    DECLARE q2 DECIMAL(5,2) DEFAULT 0;
    DECLARE q3 DECIMAL(5,2) DEFAULT 0;
    DECLARE mid_theory DECIMAL(5,2) DEFAULT 0;
    DECLARE mid_practical DECIMAL(5,2) DEFAULT 0;
    DECLARE mid_total DECIMAL(5,2) DEFAULT 0;
    DECLARE assess DECIMAL(5,2) DEFAULT 0;
    DECLARE best_quiz_sum DECIMAL(5,2);
    DECLARE ca_total DECIMAL(5,2);
```

```
DECLARE grade VARCHAR(10);

-- Get quiz, mid (theory & practical), and assessment marks
SELECT
   IFNULL(q1.Q1,0),
   IFNULL(q2.Q2,0),
   IFNULL(q3.Q3,0),
   IFNULL(m.theory,0),
   IFNULL(m.practical,0),
   IFNULL(a.Assesment_marks,0)
INTO q1, q2, q3, mid_theory, mid_practical, assess
FROM q1_marks q1
LEFT JOIN q2_marks q2 ON q1.reg_no = q2.reg_no AND q1.c_code = q2.c_code
LEFT JOIN q3_marks q3 ON q1.reg_no = q3.reg_no AND q1.c_code = q3.c_code
LEFT JOIN mid_masks m ON q1.reg_no = m.reg_no AND q1.c_code = m.c_code
LEFT JOIN assesment_marks a ON q1.reg_no = a.reg_no AND q1.c_code = a.c_code
WHERE q1.reg_no = p_reg_no AND q1.c_code = p_c_code;

-- Calculate mid_total based on presence of theory and practical
IF mid_theory > 0 AND mid_practical > 0 THEN
   SET mid_total = (mid_theory + mid_practical) / 2;
ELSE
   SET mid_total = mid_theory + mid_practical;
END IF;

-- Calculate best 2 quiz sum
SET best_quiz_sum = q1 + q2 + q3 - LEAST(q1, q2, q3);

-- Calculate total CA marks
SET ca_total = best_quiz_sum + mid_total + assess;
```

```
    -- Determine grade
    IF ca_total >= 160 THEN
        SET grade = 'CA Pass';
    ELSE
        SET grade = 'CA Fail';
    END IF;

    -- Output
    SELECT p_reg_no AS Student_ID,
           p_c_code AS Course_Code,
           best_quiz_sum AS Best_2_Quizzes,
           mid_total AS Mid_Marks,
           assess AS Assessment,
           ca_total AS CA_Total,
           grade AS CA_Grade;
END //

DELIMITER ;

CALL calculate_ca_marks_by_student('TCH/1012' , 'TMS1233') ;
```

Final procedure in Displaying marks by student - TG/2023/1726

```
-- Change delimiter to //
DELIMITER //

-- Create procedure
CREATE PROCEDURE display_student_marks(
    IN p_reg_no VARCHAR(8),
    IN p_c_code VARCHAR(8)
)
BEGIN
    SELECT
        tq.reg_no,
        tq.c_code,
        q1.Q1,
        q2.Q2,
        q3.Q3,
        am.Assesment_marks AS Assessment,
        mm.theory AS Mid_Theory,
        mm.practical AS Mid_Practical,
        em.theory AS End_Theory,
        em.practical AS End_Practical
    FROM top2_quiz_view tq
    LEFT JOIN q1_marks q1
        ON tq.reg_no = q1.reg_no AND tq.c_code = q1.c_code
    LEFT JOIN q2_marks q2
        ON tq.reg_no = q2.reg_no AND tq.c_code = q2.c_code
    LEFT JOIN q3_marks q3
```

```
        ON tq.reg_no = q3.reg_no AND tq.c_code = q3.c_code
    LEFT JOIN assesment_marks am
        ON tq.reg_no = am.reg_no AND tq.c_code = am.c_code
    LEFT JOIN mid_masks mm
        ON tq.reg_no = mm.reg_no AND tq.c_code = mm.c_code
    LEFT JOIN end_marks em
        ON tq.reg_no = em.reg_no AND tq.c_code = em.c_code
    WHERE tq.reg_no = p_reg_no
      AND tq.c_code = p_c_code;
END //

-- Restore delimiter back to default
DELIMITER ;

CALL display_student_marks('TCH/1001', 'ICT1233');
```

Final VIEW in calculating ca results for proper students - TG/2023/1726

```
CREATE OR REPLACE VIEW ca_results_for_proper_student AS
SELECT
    q1.reg_no AS Student_ID,
    q1.c_code AS Course_Code,

    -- Quiz marks
    ROUND(q1.Q1, 2) AS Q1,
    ROUND(q2.Q2, 2) AS Q2,
    ROUND(q3.Q3, 2) AS Q3,

    -- Best 2 quizzes sum
    ROUND((q1.Q1 + q2.Q2 + q3.Q3 - LEAST(q1.Q1, q2.Q2, q3.Q3)), 2) AS
Best_2_Quizzes,

    -- Mid marks (calculate based on presence of theory and practical)
    ROUND(
        CASE
            WHEN mid.theory > 0 AND mid.practical > 0 THEN (mid.theory + mid.practical)/2
            ELSE mid.theory + mid.practical
        END, 2
    ) AS Mid_Marks,

    -- Assessment marks
    ROUND(asses.Assesment_marks, 2) AS Assessment,

    -- Total CA
    ROUND(
```

```
        (q1.Q1 + q2.Q2 + q3.Q3 - LEAST(q1.Q1, q2.Q2, q3.Q3)
         + CASE
              WHEN mid.theory > 0 AND mid.practical > 0 THEN (mid.theory + mid.practical)/2
              ELSE mid.theory + mid.practical
           END
         + asses.Assesment_marks), 2
    ) AS CA_Total,

    -- CA Pass/Fail
    CASE
        WHEN (q1.Q1 + q2.Q2 + q3.Q3 - LEAST(q1.Q1, q2.Q2, q3.Q3)
            + CASE
                 WHEN mid.theory > 0 AND mid.practical > 0 THEN (mid.theory +
mid.practical)/2
                 ELSE mid.theory + mid.practical
              END
            + asses.Assesment_marks) >= 160
        THEN 'CA Pass'
        ELSE 'CA Fail'
    END AS CA_Grade

FROM q1_marks q1
JOIN q2_marks q2 ON q1.reg_no = q2.reg_no AND q1.c_code = q2.c_code
JOIN q3_marks q3 ON q1.reg_no = q3.reg_no AND q1.c_code = q3.c_code
JOIN mid_masks mid ON q1.reg_no = mid.reg_no AND q1.c_code = mid.c_code
JOIN assesment_marks asses ON q1.reg_no = asses.reg_no AND q1.c_code = asses.c_code
WHERE mid.status = "proper";



SELECT * FROM ca_results_for_proper_student ;
```

Final VIEW in calculating ca results_ca_repeters - TG/2023/1726

```
CREATE OR REPLACE VIEW ca_results_for_repeat_student AS
SELECT
    q1.reg_no AS Student_ID,
    q1.c_code AS Course_Code,

    -- Quiz marks
    ROUND(q1.Q1, 2) AS Q1,
    ROUND(q2.Q2, 2) AS Q2,
    ROUND(q3.Q3, 2) AS Q3,

    -- Best 2 quizzes sum
    ROUND((q1.Q1 + q2.Q2 + q3.Q3 - LEAST(q1.Q1, q2.Q2, q3.Q3)), 2) AS
Best_2_Quizzes,

    -- Mid marks (calculate based on presence of theory and practical)
    ROUND(
        CASE
            WHEN mid.theory > 0 AND mid.practical > 0 THEN (mid.theory + mid.practical)/2
            ELSE mid.theory + mid.practical
        END, 2
    ) AS Mid_Marks,

    -- Assessment marks
    ROUND(asses.Assesment_marks, 2) AS Assessment,

    -- Total CA
```

```
    ROUND(
        (q1.Q1 + q2.Q2 + q3.Q3 - LEAST(q1.Q1, q2.Q2, q3.Q3)
         + CASE
              WHEN mid.theory > 0 AND mid.practical > 0 THEN (mid.theory + mid.practical)/2
              ELSE mid.theory + mid.practical
           END
         + asses.Assesment_marks), 2
    ) AS CA_Total,

    -- CA Pass/Fail
    CASE
        WHEN (q1.Q1 + q2.Q2 + q3.Q3 - LEAST(q1.Q1, q2.Q2, q3.Q3)
              + CASE
                   WHEN mid.theory > 0 AND mid.practical > 0 THEN (mid.theory +
mid.practical)/2
                   ELSE mid.theory + mid.practical
                END
              + asses.Assesment_marks) >= 160
        THEN 'CA Pass'
        ELSE 'CA Fail'
    END AS CA_Grade

FROM q1_marks q1
JOIN q2_marks q2 ON q1.reg_no = q2.reg_no AND q1.c_code = q2.c_code
JOIN q3_marks q3 ON q1.reg_no = q3.reg_no AND q1.c_code = q3.c_code
JOIN mid_masks mid ON q1.reg_no = mid.reg_no AND q1.c_code = mid.c_code
JOIN assesment_marks asses ON q1.reg_no = asses.reg_no AND q1.c_code = asses.c_code
WHERE mid.status = "repeat";

SELECT * FROM ca_results_for_repeat_student ;
```

//chenuka's code here

---

get_end_marks procedure

```
DELIMITER $$

CREATE OR REPLACE PROCEDURE get_end_marks(
   IN p_reg_no VARCHAR(8),
   IN p_c_code VARCHAR(8)
)
BEGIN
   SELECT
      reg_no,
      c_code,
      CASE
         WHEN c_code = 'ENG1222' AND theory > 35 THEN 'PASS'
         WHEN c_code = 'ICT1212' AND theory > 35 THEN 'PASS'
         WHEN c_code = 'ICT1222' AND practical > 35 THEN 'PASS'
         WHEN c_code = 'ICT1233' AND ((theory + practical)/2) > 35 THEN 'PASS'
         WHEN c_code = 'ICT1242' AND theory > 35 THEN 'PASS'
         WHEN c_code = 'ICT1253' AND ((theory + practical)/2) > 35 THEN 'PASS'
         WHEN c_code = 'TCS1212' AND theory > 35 THEN 'PASS'
         WHEN c_code = 'TMS1233' AND theory > 35 THEN 'PASS'
         ELSE 'FAIL'
      END AS end_result
   FROM end_marks
   WHERE reg_no = p_reg_no
     AND c_code = p_c_code;
END$$
DELIMITER ;
```

---

proper_end_marks_view

```
CREATE OR REPLACE VIEW proper_end_marks_view AS
SELECT
   reg_no,
   c_code,
   CASE
      WHEN c_code = 'ENG1222' AND theory > 35 THEN 'PASS'
      WHEN c_code = 'ICT1212' AND theory > 35 THEN 'PASS'
      WHEN c_code = 'ICT1222' AND practical > 35 THEN 'PASS'
      WHEN c_code = 'ICT1233' AND ((theory + practical)/2) > 35 THEN 'PASS'
      WHEN c_code = 'ICT1242' AND theory > 35 THEN 'PASS'
      WHEN c_code = 'ICT1253' AND ((theory + practical)/2) > 35 THEN 'PASS'
      WHEN c_code = 'TCS1212' AND theory > 35 THEN 'PASS'
      WHEN c_code = 'TMS1233' AND theory > 35 THEN 'PASS'
      ELSE 'FAIL'
   END AS end_result
FROM end_marks
WHERE status='proper';
```

---

repeat_end_marks_view

```
CREATE OR REPLACE VIEW repeat_end_marks_view AS
SELECT
   reg_no,
   c_code,
   CASE
      WHEN c_code = 'ENG1222' AND theory > 35 THEN 'PASS'
      WHEN c_code = 'ICT1212' AND theory > 35 THEN 'PASS'
      WHEN c_code = 'ICT1222' AND practical > 35 THEN 'PASS'
      WHEN c_code = 'ICT1233' AND ((theory + practical)/2) > 35 THEN 'PASS'
      WHEN c_code = 'ICT1242' AND theory > 35 THEN 'PASS'
      WHEN c_code = 'ICT1253' AND ((theory + practical)/2) > 35 THEN 'PASS'
      WHEN c_code = 'TCS1212' AND theory > 35 THEN 'PASS'
      WHEN c_code = 'TMS1233' AND theory > 35 THEN 'PASS'
      ELSE 'FAIL'
   END AS end_result
FROM end_marks
WHERE status='repeat';
```

//sgpa

```
CREATE OR REPLACE VIEW student_sgpa AS
SELECT
   fg.reg_no,
   ROUND(SUM(c.credit *
```

```
        CASE fg.grade
            WHEN 'A+' THEN 4.0
            WHEN 'A'  THEN 4.0
            WHEN 'A-' THEN 3.7
            WHEN 'B+' THEN 3.3
            WHEN 'B'  THEN 3.0
            WHEN 'B-' THEN 2.7
            WHEN 'C+' THEN 2.3
            WHEN 'C'  THEN 2.0
            WHEN 'C-' THEN 1.7
            WHEN 'D'  THEN 1.3
            WHEN 'E'  THEN 0.0
            ELSE 0
        END
    ) / SUM(c.credit), 2) AS sgpa
FROM final_grades fg
JOIN course_unit c ON fg.c_code = c.c_code
WHERE fg.status IN ('proper', 'both repeat', 'suspended') -- only valid students
GROUP BY fg.reg_no;



//cgpa
CREATE OR REPLACE VIEW student_cgpa AS
SELECT
    fg.reg_no,
    ROUND(
        SUM(c.credit *
            CASE fg.grade
                WHEN 'A+' THEN 4.0
                WHEN 'A'  THEN 4.0
```

```
            WHEN 'A-' THEN 3.7
            WHEN 'B+' THEN 3.3
            WHEN 'B'  THEN 3.0
            WHEN 'B-' THEN 2.7
            WHEN 'C+' THEN 2.3
            WHEN 'C'  THEN 2.0
            WHEN 'C-' THEN 1.7
            WHEN 'D'  THEN 1.3
            WHEN 'E'  THEN 0.0
            ELSE 0
        END
    ) / SUM(c.credit), 2
  ) AS CGPA
FROM final_grades fg
JOIN course_unit c ON fg.c_code = c.c_code
WHERE fg.status IN ('proper','both repeat','suspended')
  AND c.c_code <> 'ENG1222' -- exclude ENG1222
GROUP BY fg.reg_no;

//procedure sand cgpa

DELIMITER $$

CREATE PROCEDURE get_student_sgpa_cgpa(IN p_reg_no CHAR(8))
BEGIN
  SELECT
    fg.reg_no,

    -- SGPA (all subjects included)
    ROUND(SUM(c.credit *
```

```
    CASE fg.grade
        WHEN 'A+' THEN 4.0
        WHEN 'A'  THEN 4.0
        WHEN 'A-' THEN 3.7
        WHEN 'B+' THEN 3.3
        WHEN 'B'  THEN 3.0
        WHEN 'B-' THEN 2.7
        WHEN 'C+' THEN 2.3
        WHEN 'C'  THEN 2.0
        WHEN 'C-' THEN 1.7
        WHEN 'D'  THEN 1.3
        WHEN 'E'  THEN 0.0
        ELSE 0
    END
) / SUM(c.credit), 2) AS SGPA,

-- CGPA (excluding ENG1222)
ROUND(SUM(
    CASE
        WHEN c.c_code <> 'ENG1222' THEN c.credit *
            CASE fg.grade
                WHEN 'A+' THEN 4.0
                WHEN 'A'  THEN 4.0
                WHEN 'A-' THEN 3.7
                WHEN 'B+' THEN 3.3
                WHEN 'B'  THEN 3.0
                WHEN 'B-' THEN 2.7
                WHEN 'C+' THEN 2.3
                WHEN 'C'  THEN 2.0
                WHEN 'C-' THEN 1.7
```

```
                WHEN 'D'  THEN 1.3
                WHEN 'E'  THEN 0.0
                ELSE 0
            END
        END
    ) / SUM(
        CASE WHEN c.c_code <> 'ENG1222' THEN c.credit ELSE 0 END
    ), 2) AS CGPA

  FROM final_grades fg
  JOIN course_unit c ON fg.c_code = c.c_code
  WHERE fg.reg_no = p_reg_no
    AND fg.status IN ('proper','both repeat','suspended');
END $$

DELIMITER ;
```

11. Problems that we faced during the development of the solution

- **Managing Multiple Medical Submissions**:
  Students occasionally tried to submit more medicals than permitted for a subject. Without proper validation, this could have caused inconsistencies in attendance and exam eligibility.

- **Maintaining Consistency Between Attendance and Exam Marks:**
  Updating attendance and exam marks simultaneously when a medical was applied was challenging. Any miscalculation could have led to incorrect final grades or attendance percentages.

- **Handling Missing or Partial Data:**
  Some students missed only theory or only practical sessions, making it difficult to calculate averages, total marks, and eligibility accurately.

- **Data Integrity Across Multiple Tables:**
  Attendance, medical for attendance, and medical for exam tables are interconnected. Updates in one table could affect others, risking inconsistent or invalid records.

- **Calculating GPA and SGPA:**
  GPA and SGPA calculations were complex because they required combining multiple exam marks (quizzes, assessments, midterm, final) with varying credit weights and applying the new 2024 grading criteria.

- **Storing and Managing Various Exam Marks:**
  Organizing and retrieving different exam components (quizzes, assessments, theory, practical, mid, and final) consistently across students and courses was challenging.
- **Joining Multiple Tables for Final Marks:**
  Generating final marks required joining attendance, medicals, and exam tables, applying medical considerations, and ensuring accurate calculation for each student.

- **Implementing 2024 Evaluation Criteria:**
  Applying the updated marking rules for quizzes, assessments, midterms, and final exams in stored procedures and triggers required careful planning to ensure compliance.

- **Ensuring Automation and Accuracy:**
  Triggers, views, and procedures had to work flawlessly to automatically update attendance, medicals, and exam marks. Any error could lead to incorrect GPA, eligibility, or final marks.

12. Solution how we have overcome the above identified problems

- **Restricting Medical Submissions:**
  Implemented triggers that limited each student to two medicals per subject and generated error messages if exceeded, preventing abuse of the system.

- **Synchronizing Attendance and Exam Updates:**
  Created triggers that automatically updated attendance status and exam marks whenever a medical was approved, ensuring consistency across all related tables.

- **Handling Partial or Missing Data:**
  Used conditional logic in stored procedures to compute attendance percentages and marks correctly even when some components were missing.

- **Maintaining Data Integrity:**
  Applied foreign keys, constraints, and carefully designed triggers to ensure that updates in one table did not break consistency in other tables.

- **Automating GPA and SGPA Calculations:**
  Developed stored procedures that pulled data from all relevant exam tables, applied course credits, and computed GPA/SGPA according to the 2024 grading criteria.

- **Organizing Multiple Exam Components:**
  Created separate tables for different exam types (quizzes, assessments, midterm, final, theory, practical) and linked them to students and courses for structured, consistent storage.

- **Joining Multiple Tables Efficiently:**
  Designed stored procedures to join attendance, medical, and exam tables seamlessly, ensuring that final marks reflected medical adjustments accurately.

- **Implementing 2024 Evaluation Rules:**
  Applied weighted logic in stored procedures to calculate CA, total marks, and final grades according to the updated 2024 criteria.

- **Ensuring Automation and Accuracy:**
  Used triggers, views, and stored procedures in combination to automatically update attendance, medicals, exam marks, and eligibility, reducing manual errors and maintaining accuracy throughout the system.

Name: K. R. M. D. N. Nimshan

Registration No: TG/2023/1755

Introduction

The main objective of this project was to design and implement a relational database system for the Faculty of Technology (FOT) to effectively manage student attendance and academic performance. The system was developed to streamline the process of recording attendance, managing medical submissions, and determining eligibility for examinations based on attendance and medical considerations.

My individual responsibility in this project was to design and implement the attendance and medical management modules. These components automate key administrative tasks, ensuring that students' attendance percentages and medical submissions are handled fairly, transparently, and efficiently. The system eliminates manual calculations and leverages triggers and stored procedures to maintain real-time data consistency.

Database Design and Implementation

To support the attendance and medical management processes, I designed the following three key tables:

- Attendance
- Medical for Attendance
- Medical for Exam

Each table serves a specific role in tracking attendance, managing medical submissions, and automating updates through triggers.

Attendance Table

The Attendance table is used to record every attendance entry for all students throughout the semester. The table includes the following key fields:

- **att_id** – Unique identifier for each attendance record
- **type** – Specifies whether the session is *theory* or *practical*
- **atten_date** – Date of the class session
- **status** – Indicates whether the student was *present* or *absent*
- **change_status** – Reflects any automatic status updates caused by medical submissions
- **is_medical_affect** – Indicates whether a medical submission has impacted that record

This table maintains a complete record of student attendance data and forms the basis for calculating attendance percentages and eligibility for examinations.

Medical for Attendance Table

The medical_for_attendance table was designed to manage medical submissions related to lecture or practical attendance. Its structure is as follows:

Field	Description
medical_id	Unique identifier for each medical submission
c_code	Course code to which the medical applies
reg_no	Student's registration number
submission_date	Date the medical was submitted
description	Details or reason for the medical
affected_date	Date of the missed class
session_id	Session identifier for the missed class
status	Indicates approval stage (*pending*, *approved*, or *not approved*)
att_id	Links to the related attendance record

This table allows students to submit medical documents for missed classes. A student may submit up to two medicals per subject per semester, which is enforced through triggers to prevent excessive submissions.

Medical for Exam Table

The medical_for_exam table manages medical submissions related to quizzes, assessments, mid-semester exams, and end-semester exams. Its structure includes:

Field	Description
medical_id	Unique identifier for each exam-related medical
c_code	Course code of the affected exam
reg_no	Student's registration number
submission_date	Date of submission
description	Explanation of the medical reason
marks_type	Specifies the affected component (*q1_marks*, *q2_marks*, *q3_marks*, *assessment_marks*, *mid_marks*, *end_marks*)
is_practical_or_theory	Indicates whether it applies to the *practical* or *theory* component
status	Tracks the medical's approval stage (*pending*, *approved*, *not approved*)

This table helps manage and track medical excuses for examination components and ensures that student marks are correctly adjusted based on approval status.

Triggers and Automation

To maintain data consistency and automate system behavior, several MySQL triggers were implemented for both medical tables.

1. **Trigger for Attendance Medical Insertion:**
   When a student submits a medical record for attendance, this trigger automatically updates the related change_status field in the Attendance table, marking that session as affected by medical leave.

2. **Trigger for Medical Status Updates (Attendance):**
   When an administrator changes a medical record's status from *pending* to *approved* or *not approved*, the trigger updates the corresponding Attendance record. This ensures that once the medical is approved, the student's attendance status is appropriately modified.

3. **Trigger to Restrict Medical Submissions:**
   A validation trigger ensures that a student cannot submit more than two medicals per subject. If a student attempts to exceed this limit, the system raises a user-defined error message, notifying them that they have already submitted the maximum number of medicals allowed.

4. **Trigger for Medical for Exam Submissions:**
   This trigger activates when a student submits a medical for any exam component. It automatically updates the related marks record by replacing the score with "MC," indicating that the student is under medical consideration.

5. **Trigger for Medical for Exam Status Updates:**
   Once the administrator reviews the exam-related medical and updates its status, the trigger ensures that the marks table reflects the decision accurately—either maintaining "MC" for approved cases or reverting the marks if not approved.

These triggers ensure automation, consistency, and fairness across the system by minimizing manual administrative actions.

Views and Stored Procedures

To simplify data access and reporting, I developed several MySQL views and stored procedures:

- **Views:**
    - To display all student attendance details.
    - To calculate attendance percentages for each student.
    - To identify students above or below the 80% attendance eligibility threshold.
    - To generate final eligibility lists for examination participation.
- **Stored Procedures:**
    - To extract attendance details for an individual student.
    - To retrieve attendance data for a specific course.
    - To generate attendance summaries for entire student batches.

These tools enable lecturers and administrators to easily monitor attendance performance and make eligibility decisions quickly and accurately.

Conclusion

In conclusion, this project successfully implemented an automated database system for managing student attendance and medical records within the Faculty of Technology. My contribution focused on developing the attendance and medical management modules, including the related tables, triggers, views, and stored procedures.

These components work together to ensure that attendance tracking is accurate, medical submissions are handled transparently, and eligibility for examinations is determined automatically according to university policy. Through this process, I gained hands-on experience in relational database design, trigger-based automation, data validation, and procedural SQL programming.

Overall, this system provides a practical and efficient solution that enhances administrative accuracy, supports fair student evaluation, and contributes to the digital modernization of academic record management at the Faculty of Technology.

Name : D.G.B.N. Dilshan

Index Number : TG/2023/1760

1. Introduction

My part of the project focused on creating procedures and views in MySQL to manage and display student grades and final marks.The goal was to make it easier for students and teachers to view grades using either a student registration number or a course code**.**
I developed automated calculations for grades**,** SGPA, and CGPA, ensuring accuracy and consistency.
The procedures also simplified how data is retrieved and presented from multiple tables.
This approach reduces manual work and minimizes calculation errors.
Overall, my contribution improved efficiency, transparency, and accessibility in managing academic results.

2. Objectives

main objectives of my work were:

Show student grades

- View grades for a specific student using their registration number.
- View grades for all students in a specific course using the course code.

Show final marks

- View final marks for a specific student using their registration number.
- View final marks for all students in a specific course using the course code.

Create views for the whole batch

- A view showing all grades for all students.
- A view showing final marks for all students.

Ensure correct calculations

- Combine quiz, mid-term, assessment, and final exam marks.
- Apply grading rules correctly

3. Implementation

Main View (final_grades)

- Combines all marks from different tables.
- Calculates **final marks** for each student.
- Assigns a **grade** based on final marks and student status.

Simplified Views

- final_grades_Only: Shows only **student ID, course code, and grade**.
- final_marks_Only: Shows only **student ID, course code, and final marks**.

Stored Procedures

- get_student_grades: Shows grades for a student.
- get_course_grades: Shows grades for a course.
- get_course_finalmarks (by student): Shows final marks for a student.
- get_course_finalmarks (by course): Shows final marks for a course.

How it works

- Used CREATE VIEW to make views up-to-date.
- Used DELIMITER to create procedures with multiple SQL statements.
- Used CASE to handle special cases like missing marks or medical certificates.
- Joined multiple tables to get all marks in one place.

4. Conclusion

My work makes it easy to view grades and final marks for both individual students and entire courses. The views and procedures I created help make the database system more simple, fast, and accurate. With these features, students and teachers can quickly access the information they need by using either a registration number or a course code**.**

This approach reduces manual effort, improves efficiency, and ensures that academic results are always clear and reliable.

NAME - V.C.L. JAYASURIYA.

REG_NO - TG/2023/1726

1. Introduction

In our university database project, my main goal was to create a system that can manage and calculate Continuous Assessment (CA) marks for students .My part of the project was to design and create MySQL stored procedures and views to make this process automatic. With my code, the database can calculate total CA marks, decide if the student passed or failed, and show all marks in one place.

2. My Main Responsibilities

1. **Procedure to calculate CA marks**
   Calculates total CA marks using quizzes, mid exam, and assessment.
2. **Procedure to display all student marks**
   Shows all marks of a student when entering registration number and course code.
3. **View for proper students' CA results**
   Shows CA details and status for students marked as "proper."
4. **View for repeat students' CA results**
   Shows CA details and status for students marked as "repeat."

Procedure to Display All Marks

Name: display_student_marks

This procedure is used to display **all marks** of a student for a selected course.
It takes the registration number and course code as input and shows:

- Quiz 1, Quiz 2, Quiz 3 marks
- Assessment marks
- Mid exam marks (theory and practical)
- End exam marks (theory and practical)

It collects these details from multiple tables by joining them together, so everything appears in one place.

Example use: CALL display_student_marks('TCH/1001', 'ICT1233');

Procedure to Calculate CA Marks

Name: calculate_ca_marks_by_student

This procedure is used to find the total CA marks of a student.
When we enter a student's registration number and course code, it does the following steps automatically:

1. Takes marks from several tables:
    - Quiz 1 (q1_marks)
    - Quiz 2 (q2_marks)
    - Quiz 3 (q3_marks)
    - Mid exam marks (mid_masks)
    - Assessment marks (assesment_marks)
2. Finds the best 2 quizzes:
   From the 3 quiz marks, it removes the lowest one and adds the best two.
3. Calculates mid exam marks:
   If both theory and practical marks exist, it takes their average.
   If only one is available, it uses that mark.
4. Adds all the marks together:
   Best 2 quizzes + Mid marks + Assessment = CA Total
5. Checks pass or fail:
    - If CA total ≥ 160 → "CA Pass"
    - Else → "CA Fail"
6. Displays the final result:
   It shows all the marks, the total CA, and the CA grade.

Example use: CALL calculate_ca_marks_by_student('TCH/1012', 'TMS1233');

View for Proper Students

View name: ca_results_for_proper_student

This view displays only students whose status = "proper".
It shows:

- Student ID and Course Code
- Quiz 1, Quiz 2, Quiz 3 marks
- Best two quiz total
- Mid marks (calculated automatically)
- Assessment marks
- Total CA marks
- CA grade (Pass or Fail)

Example : SELECT * FROM ca_results_for_proper_student;

View for Repeat Students

View name: ca_results_for_repeat_student

This view is almost the same as the one above but shows data for students whose status = "repeat".
It helps the lecturer or coordinator to separately check repeat students' performance.

Example : SELECT * FROM ca_results_for_repeat_student;

3. Conclusion

In this project, I successfully created stored procedures and views to handle the Continuous Assessment (CA) process automatically.
These database components save time, improve accuracy, and make the system more efficient.

My work ensures that teachers and administrators can easily calculate, view, and manage student marks without doing it manually.
This contribution made our database project more complete and practical for real university use.

NAME – Chenuka Hasith

REG – TG/2023/1751

1. Introduction

Our group project was about developing a Database Management System to manage student marks and performance in a university environment. It includes stored procedures and views to calculate and display students' results, such as end exam pass/fail status, SGPA (Semester Grade Point Average), and CGPA (Cumulative Grade Point Average). The system ensures accurate grade calculations using standard grade points and applies specific rules like excluding certain subjects from CGPA.

2. My Responsibilities in the Project

In this project, my main task was to create:

I. View – Student_sgpa
II. View – Student_cgpa
III. View – Proper_end_marks_view
IV. View – Repeat_end_marks_view
V. Procedure – Get_end_mark
VI. Procedure – Get_student_sgpa_cgpa

3. View: Student_sgpa

Purpose

This view calculates each student's Semester Grade Point Average (SGPA) by combining their course grades and course credits for one semester.

How It Works

- It joins two tables:
    - final_grades → contains each student's grade for every subject.
    - course_unit → contains the credit value for each subject.
- Each letter grade (A+, A, B+, etc.) is converted into a grade point value (for example, A = 4.0, B+ = 3.3).
- Each grade point is multiplied by the subject's credit to get "credit × grade point."

- Then the sum of all “credit × grade point” values is divided by the total number of credits the student took.
- Finally, the result is rounded to two decimal places and displayed as the student’s SGPA.

4. View: Student_cgpa

Purpose

This view calculates the Cumulative Grade Point Average (CGPA) for each student across all semesters.
It works similarly to the SGPA view but excludes the ENG1222 subject (English course) from the calculation because it is not counted for CGPA.

How It Works

- It joins the same two tables (final_grades and course_unit).
- Converts grades to grade points.
- Multiplies credits by grade points.
- Sums up all the “credit × grade point” values and divides them by the total credits (excluding ENG1222).
- The result is rounded to two decimals and shown as **CGPA**.

5. View – Proper_end_marks_view

Purpose

- This view displays the end exam results of proper students only (students who are currently taking the course normally).
- How It Works
- It selects data from the end_marks table where the status is 'proper'.
  Then it uses the same logic as the procedure to check if each student passed or failed based on their marks.

6. View – Repeat_end_marks_view

Purpose

- This view shows the end exam results of repeat students (students who are retaking the subject).
- How It Works
- It is similar to the proper_end_marks_view, but it filters only students whose status is 'repeat'.
  It then calculates their pass/fail result based on their latest exam marks.

7. Procedure: Get_student_sgpa_cgpa

Purpose

The procedure is used to display both **SGPA and CGPA** of a specific student using their registration number.

How It Works

- Input: **Student Registration Number (p_reg_no)**
- It calculates:
    - **SGPA:** Includes all courses.
    - **CGPA:** Excludes "ENG1222."
- It uses the same grade-to-point conversion logic.
- Both values are displayed in one result table.

8. Stored Procedure – Get_end_marks

Purpose

The get_end_marks procedure is used to check a student's end exam result for a specific course.
It takes two inputs:

- **p_reg_no:** The registration number of the student.
- **p_c_code:** The course code.

How It Works

The procedure uses a **CASE** statement to check the student's marks for the selected course.

- If the **theory mark** or **average of theory and practical** is greater than **35**, the student **passes**.
- If the marks are **below 35**, the student **fails**.

This rule changes slightly depending on the subject — for example, some subjects consider only theory marks, while others use both theory and practical marks.

Example Use

CALL get_end_marks('TCH/1012', 'ICT1233');

9. Conclusion

In this project, I developed four views, (student_sgpa,student_cgpa,proper_end_marks_view,repeat_end_marks_view,) and stored procedure (get_student_sgpa_cgpa, get_end_mark) using MySQL.These parts are responsible for automatically calculating and displaying students' SGPA and CGPA based on their grades and credit values.

These components are responsible for automatically calculating and displaying each student's Semester Grade Point Average (SGPA) and Cumulative Grade Point Average (CGPA) based on their grades and corresponding credit values from the course_unit table. The CGPA calculation also accounts for special cases, such as excluding specific subjects like ENG1222, to ensure fairness and accuracy.

15.References

- UGC Commission circular No 12-2024
- YouTube
- Stack Overflow
- Fundamentals of Database Systems (7th Edition) By Remez Elmasri & Shamkant B. Navathe