Department of Information and Communication Technology

Faculty of Technology

University of Ruhuna

Database Management Systems

Practicum ICT 1222

Assignment 02 – Mini Project

Group 10

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

Submitted by: TG/2023/1699
TG/2023/1734
TG/2023/1747
TG/2023/1772

Content

1.1 Brief Introductions about the Project

The purpose of this project to design and implement a Database Management System that automates and centralizes the handling of student and other users data.The system will allow administrators,lectures,technical officers and student to perform their respective functins through controlled database access.It will manage data related to student profiles, attendance, course units and exam marks while ensuring data integrity, security and accessibility.

The project follows the Database Development Life Cycle (DDLC) to design a structured and efficient database solution that meets the faculty's academic and administrative requirements.

This system will also incorporate user roles with specific privileges to ensure secure and efficient operations for all stakeholders.

1.2 Brief Introductions about The Solution

- Storing basic student details, subject units and examination data.

  This feature is responsible for maintaining and managing all essential student information, including personal details, enrolled course units and examination records. It ensures that data is stored securely and can be easily accessed or updated when needed, providing a centralized database for academic management.

- Preparing attendance reports and calculating centuries

  The system records students' daily attendance and automatically generates attendance reports. It calculates attendance percentages accurately, helping lecturers and administrators monitor student participation and identify irregular attendance patterns efficiently.

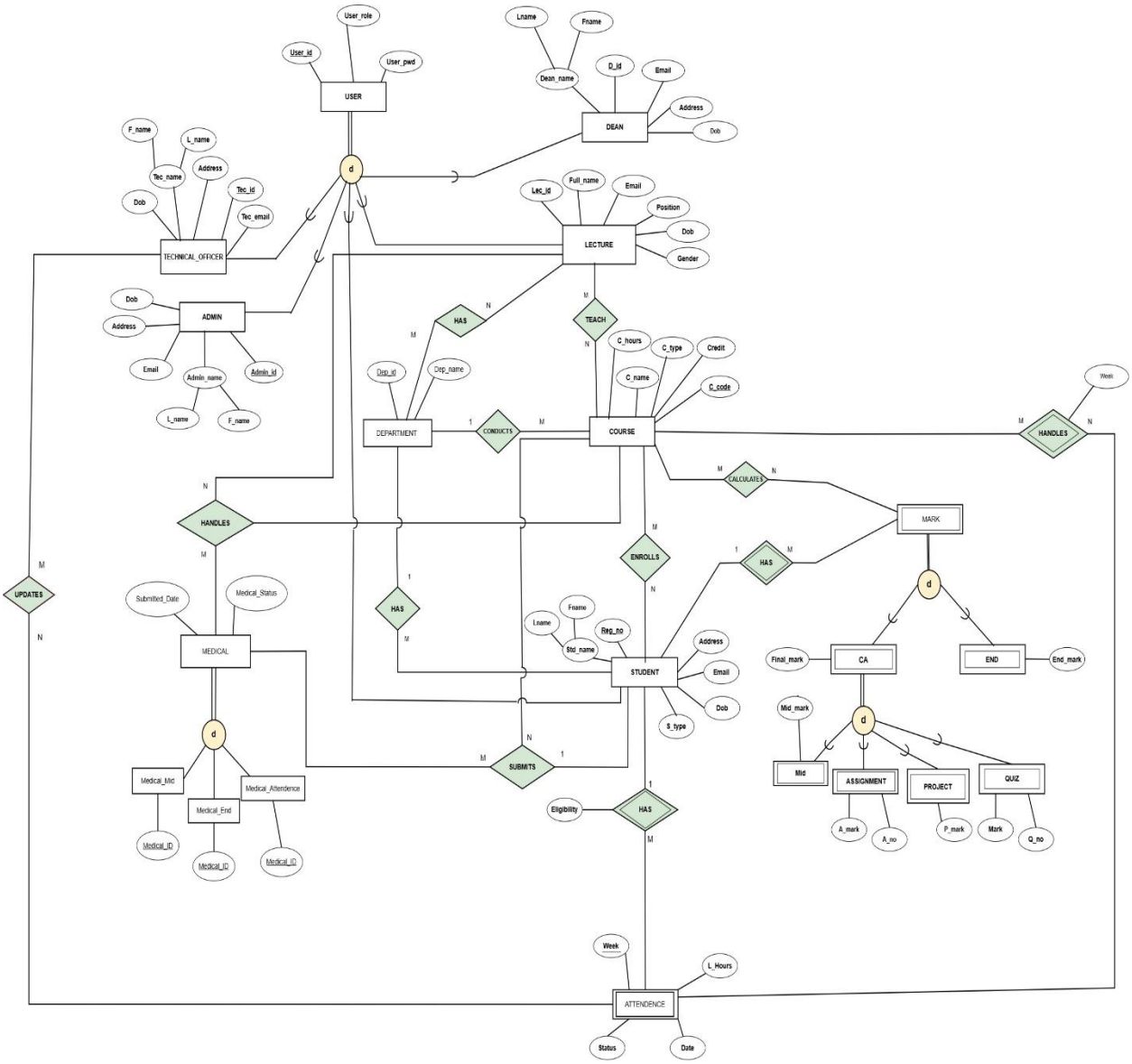
- Systematization of midterm, final and practical marks

  This component organizes and manages the entry and processing of marks from various assessments such as midterm examination, final examination and practical sessions. It helps standardize the grading process and ensure that all evaluations are recorded in a consistent and transparent manner.

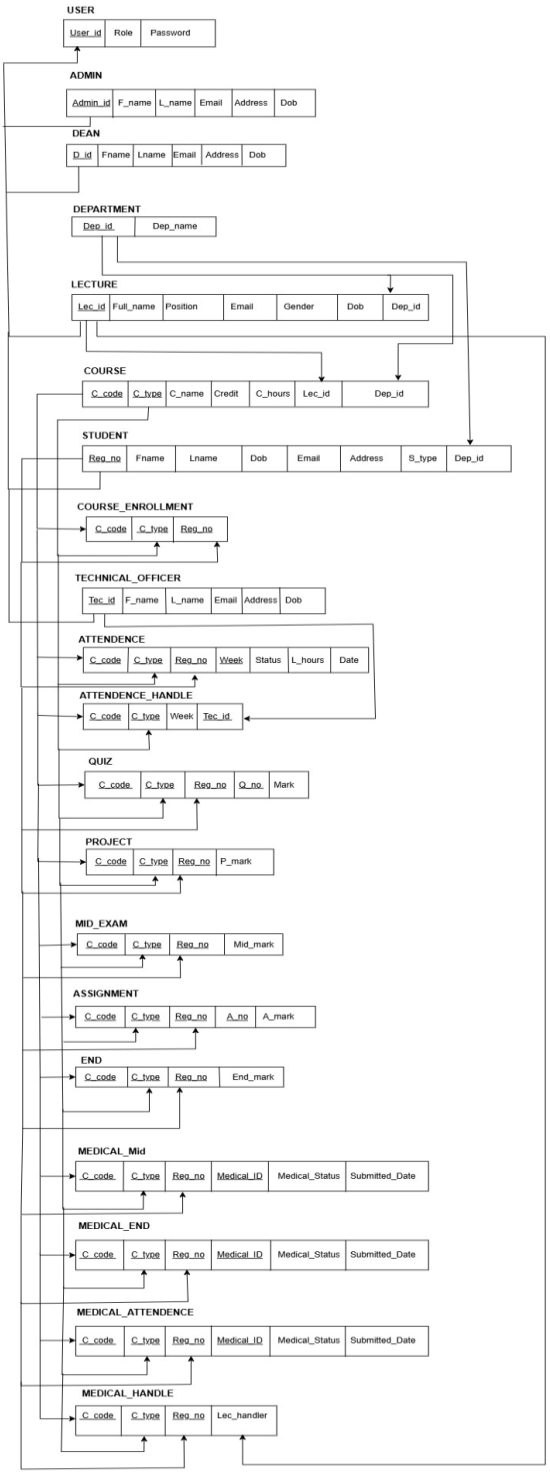
- Automatic calculation of results, grades, SGPA and CGPA

  The system automatically computes students' final results, convert marks into grades and calculates semester-wise performance indicators such as SGPA (Semester Grade Point Average) and CGPA (Cumulative Grade Point Average). This automation minimizes human error and provides fast, accurate academic evaluations.

2.1 ER/EER Diagram



2.2 Relational Mapping



3.1 Table Structure of Solution

USER

```
CREATE TABLE User
(
  User_id CHAR(12)  NOT NULL,
  Role VARCHAR(30) NOT NULL,
  Password VARCHAR(50) NOT NULL,
  PRIMARY KEY(User_id)
);
```

```
mysql> desc user;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| User_id  | char(12)    | NO   | PRI | NULL    |       |
| Role     | varchar(30) | NO   |     | NULL    |       |
| Password | varchar(50) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

DEAN

```
CREATE TABLE Dean(
   D_id CHAR(12) NOT NULL
PRIMARY KEY,
   Fname VARCHAR(50) NOT NULL,
   Lname VARCHAR(50) NOT NULL,
   Email VARCHAR(30) NOT NULL,
   Address VARCHAR(100),
   Dob DATE,

   -- Foreign key 01 : Related to User
 FOREIGN KEY (D_id) REFERENCES User(User_id)
      ON UPDATE CASCADE
);
```

```
mysql> desc dean;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| D_id    | char(12)     | NO   | PRI | NULL    |       |
| Fname   | varchar(50)  | NO   |     | NULL    |       |
| Lname   | varchar(50)  | NO   |     | NULL    |       |
| Email   | varchar(30)  | NO   |     | NULL    |       |
| Address | varchar(100) | YES  |     | NULL    |       |
| Dob     | date         | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

ADMIN

```
CREATE TABLE Admin(
        Admin_id CHAR(12)  NOT NULL,
        F_name VARCHAR(50) NOT NULL,
        L_name VARCHAR(50) NOT NULL,
        Email VARCHAR(30) NOT NULL,
        Address VARCHAR(100) NOT NULL,
        Dob DATE ,

        PRIMARY KEY(Admin_id),

   -- Foreign key 01 : Related to User
 FOREIGN KEY (Admin_id)
REFERENCES User(User_id)
      ON UPDATE CASCADE
);
```

```
mysql> desc admin;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| Admin_id | char(12)     | NO   | PRI | NULL    |       |
| F_name   | varchar(50)  | NO   |     | NULL    |       |
| L_name   | varchar(50)  | NO   |     | NULL    |       |
| Email    | varchar(30)  | NO   |     | NULL    |       |
| Address  | varchar(100) | NO   |     | NULL    |       |
| Dob      | date         | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

TECHNICAL_OFFICER

```
CREATE TABLE Technical_officer(
   Tec_id CHAR(12) NOT NULL,
   F_Name VARCHAR(50) NOT NULL,
   L_Name VARCHAR(50) NOT NULL,
   Email VARCHAR(30) NOT NULL,
   Address VARCHAR(100) NOT NULL,
   Dob DATE,

   PRIMARY KEY(Tec_id),

   -- Foreign key 01 : Related to User
   FOREIGN KEY (Tec_id) REFERENCES User(User_id)
      ON UPDATE CASCADE
);
```

```
mysql> desc Technical_officer;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| Tec_id  | char(12)     | NO   | PRI | NULL    |       |
| F_Name  | varchar(50)  | NO   |     | NULL    |       |
| L_Name  | varchar(50)  | NO   |     | NULL    |       |
| Email   | varchar(30)  | NO   |     | NULL    |       |
| Address | varchar(100) | NO   |     | NULL    |       |
| Dob     | date         | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

LECTURER

```
CREATE TABLE Lecturer(
  Lec_id CHAR(10)  NOT NULL,
  Fullname VARCHAR(100),
  position VARCHAR(30),
  email VARCHAR(30)  NOT NULL,
  gender ENUM('Male','Female','Other'),
  dob DATE,
  Dep_id CHAR(5),

  PRIMARY KEY(Lec_id),

  -- Foreign key 01 : Related to Department
  CONSTRAINT fk_lec_dep
  FOREIGN KEY (Dep_id) REFERENCES Department(Dep_id)
  ON UPDATE CASCADE,

 -- Foreign key 02 : Related to User
  FOREIGN KEY (Lec_id) REFERENCES User(User_id)
      ON UPDATE CASCADE
);
```

```
mysql> desc Lecturer;
+----------+-------------------------------+------+-----+---------+-------+
| Field    | Type                          | Null | Key | Default | Extra |
+----------+-------------------------------+------+-----+---------+-------+
| Lec_id   | char(10)                      | NO   | PRI | NULL    |       |
| Fullname | varchar(100)                  | YES  |     | NULL    |       |
| position | varchar(30)                   | YES  |     | NULL    |       |
| email    | varchar(30)                   | NO   |     | NULL    |       |
| gender   | enum('Male','Female','Other') | YES  |     | NULL    |       |
| dob      | date                          | YES  |     | NULL    |       |
| Dep_id   | char(5)                       | YES  | MUL | NULL    |       |
+----------+-------------------------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

DEPARTMENT

```
CREATE TABLE Department(
          Dep_id CHAR(5)  NOT NULL,
          Dep_name VARCHAR(50)
           NOT NULL,
          PRIMARY KEY(Dep_id)
);
```

```
mysql> desc Department;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| Dep_id   | char(5)     | NO   | PRI | NULL    |       |
| Dep_name | varchar(50) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

STUDENT

```
mysql> desc Student;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| Reg_no  | char(12)     | NO   | PRI | NULL    |       |
| Fname   | varchar(50)  | YES  |     | NULL    |       |
| Lname   | varchar(50)  | YES  |     | NULL    |       |
| dob     | date         | YES  |     | NULL    |       |
| email   | varchar(100) | NO   |     | NULL    |       |
| address | varchar(150) | YES  |     | NULL    |       |
| S_type  | varchar(10)  | NO   |     | NULL    |       |
| Dep_id  | char(5)      | YES  | MUL | NULL    |       |
+---------+--------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

```
CREATE TABLE Student (
  Reg_no CHAR(12) NOT NULL,
  Fname VARCHAR(50),
  Lname VARCHAR(50),
  dob DATE,
  email VARCHAR(100) NOT NULL,
  address VARCHAR(150),
  S_type VARCHAR(10) NOT NULL,
  Dep_id CHAR(5),

  PRIMARY KEY (Reg_no),

  CONSTRAINT fk_student_dep FOREIGN KEY (Dep_id) REFERENCES Department(Dep_id)
    ON UPDATE CASCADE ,

  CONSTRAINT fk_student_user FOREIGN KEY (Reg_no) REFERENCES User(User_id)
    ON UPDATE CASCADE ON DELETE CASCADE
);
```

COURSE

```
mysql> desc Course;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| C_code  | char(8)       | NO   | PRI | NULL    |       |
| C_type  | enum('P','T') | NO   | PRI | NULL    |       |
| C_name  | varchar(50)   | NO   |     | NULL    |       |
| Credit  | int           | NO   |     | NULL    |       |
| C_hours | int           | YES  |     | NULL    |       |
| Lec_id  | char(10)      | NO   | MUL | NULL    |       |
| Dep_id  | char(5)       | YES  | MUL | NULL    |       |
+---------+---------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

```
CREATE TABLE Course(
  C_code CHAR(8) NOT NULL,
  C_type ENUM('P','T') NOT NULL,
  C_name VARCHAR(50) NOT
NULL,
  Credit INT NOT NULL,
  C_hours INT,
  Lec_id CHAR(10) NOT NULL,
  Dep_id CHAR(5),
  PRIMARY KEY(C_code, C_type),

  -- Foreign key 01 : Related to Lecturer (use Lecturer(Lec_id))
  CONSTRAINT fk_course_lec
    FOREIGN KEY (Lec_id) REFERENCES Lecturer(Lec_id)
    ON UPDATE CASCADE,

  -- Foreign key 02 : Related to Department
  CONSTRAINT fk_course_dep
    FOREIGN KEY (Dep_id) REFERENCES Department(Dep_id)
    ON UPDATE CASCADE
);
```

COURSE ENROLLMENT

```
mysql> desc Course_Enrollment;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| C_code  | char(8)       | NO   | PRI | NULL    |       |
| C_type  | enum('P','T') | NO   | PRI | NULL    |       |
| Regno   | char(12)      | NO   | PRI | NULL    |       |
+---------+---------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
CREATE TABLE Course_Enrollment(
        C_code CHAR(8) NOT NULL,
        C_type ENUM('P','T') NOT
NULL,
        Regno CHAR(12) NOT NULL,
        PRIMARY KEY(C_code,C_type,Regno),

        -- Foreign key 1:Relate to the Course
        CONSTRAINT fk_EnrollCourse FOREIGN KEY(C_code,C_type) REFERENCES
Course(C_code,C_type)
  ON UPDATE CASCADE,

        -- Foreign key 2:Relate to the Student
        CONSTRAINT fk_EnrollStudent FOREIGN KEY(Regno) REFERENCES
User(User_id)
  ON UPDATE CASCADE ON DELETE CASCADE
);
```

ATTENDENCE

```
mysql> desc Attendence;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| C_code  | char(8)       | NO   | PRI | NULL    |       |
| C_type  | enum('P','T') | NO   | PRI | NULL    |       |
| Regno   | char(12)      | NO   | PRI | NULL    |       |
| Week    | int           | NO   | PRI | NULL    |       |
| Date    | date          | YES  |     | NULL    |       |
| Status  | varchar(7)    | NO   |     | NULL    |       |
| L_Hours | int           | NO   |     | NULL    |       |
+---------+---------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

```
CREATE TABLE Attendence(
        C_code CHAR(8) NOT NULL,
        C_type ENUM('P','T') NOT
NULL,
        Regno CHAR(12) NOT NULL,
        Week INT NOT NULL,
  Date Date,
        Status VARCHAR(7) NOT NULL,
        L_Hours INT NOT NULL,

        PRIMARY KEY(Regno,C_code,C_type,week),

        -- Foreign key 1:Relate to the Course
        CONSTRAINT fk_CourseAttend FOREIGN KEY(C_code,C_type) REFERENCES
Course(C_code,C_type)
  ON UPDATE CASCADE,

        -- Foreign key 2:Relate to the Student
        CONSTRAINT fk_StudentAttend FOREIGN KEY(Regno) REFERENCES
User(User_id)
  ON UPDATE CASCADE ON DELETE CASCADE

);
```

```
mysql> desc Attendence_handle;
+-----------+---------------+------+-----+-------------------+-------------------+
| Field     | Type          | Null | Key | Default           | Extra             |
+-----------+---------------+------+-----+-------------------+-------------------+
| C_code    | char(8)       | NO   | PRI | NULL              |                   |
| C_type    | enum('P','T') | NO   | PRI | NULL              |                   |
| Week      | int           | NO   | PRI | NULL              |                   |
| Tec_id    | char(12)      | NO   | MUL | NULL              |                   |
| Handle_at | timestamp     | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----------+---------------+------+-----+-------------------+-------------------+
5 rows in set (0.00 sec)
```

ATTENDENCE HANDLE

```
CREATE TABLE Attendence_handle(
        C_code CHAR(8) NOT NULL,
        C_type ENUM('P','T') NOT NULL,
        Regno CHAR(12) NOT NULL,
        Medical_type ENUM('Mid','End','Attendence') NOT NULL,
        Week INT NOT NULL,
        Tec_id CHAR(12) NOT NULL,
   Handle_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        PRIMARY KEY(C_code,C_type,week),

        -- Foreign key 1:Relate to the Course
        CONSTRAINT fk_AttendenceHandle FOREIGN KEY(C_code,C_type) REFERENCES
Course(C_code,C_type)
   ON UPDATE CASCADE,

        -- Foreign key 2:Relate to Technical_officer
        CONSTRAINT fk_Officerhandle FOREIGN KEY(Tec_id) REFERENCES
User(User_id)
   ON UPDATE CASCADE ,
);
-- Foreign key 3:Relate to Student
        CONSTRAINT fk_Studenthandle FOREIGN KEY(Regno) REFERENCES
User(User_id)
   ON UPDATE CASCADE
);
```

QUIZ

```
mysql> desc Quiz;
+--------+---------------+------+-----+---------+-------+
| Field  | Type          | Null | Key | Default | Extra |
+--------+---------------+------+-----+---------+-------+
| C_code | char(8)       | NO   | PRI | NULL    |       |
| C_type | enum('P','T') | NO   | PRI | NULL    |       |
| Regno  | char(12)      | NO   | PRI | NULL    |       |
| Mark   | float(5,2)    | YES  |     | NULL    |       |
| Q_no   | int           | NO   | PRI | NULL    |       |
+--------+---------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
CREATE TABLE Quiz(
   C_code CHAR(8) NOT NULL,
   C_type ENUM('P','T') NOT NULL,
   Regno CHAR(12) NOT NULL,
   Mark FLOAT(5,2) CHECK (Mark >= 0
AND Mark <= 100),
   Q_no INT NOT NULL CHECK (Q_no >=
1 AND Q_no <= 3),

   PRIMARY KEY(C_code, C_type, Regno, Q_no),

   -- Foreign_Key 1: Relate to the course
   CONSTRAINT fk_QCourse FOREIGN KEY(C_code, C_type) REFERENCES
Course(C_code, C_type)
   ON UPDATE CASCADE,

   -- FOREIGN_KEY 2: Relate to the student
   CONSTRAINT fk_Qregno FOREIGN KEY(Regno) REFERENCES User(User_id)
ON UPDATE CASCADE ON DELETE CASCADE);
```

ASSIGNMNET

```
mysql> desc Assignment;
+--------+---------------+------+-----+---------+-------+
| Field  | Type          | Null | Key | Default | Extra |
+--------+---------------+------+-----+---------+-------+
| C_code | char(8)       | NO   | PRI | NULL    |       |
| C_type | enum('P','T') | NO   | PRI | NULL    |       |
| Regno  | char(12)      | NO   | PRI | NULL    |       |
| A_mark | float(5,2)    | NO   |     | NULL    |       |
| A_no   | int           | NO   | PRI | NULL    |       |
+--------+---------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
CREATE TABLE Assignment(
   C_code CHAR (8) NOT NULL,
   C_type ENUM('P','T') NOT NULL,
   Regno CHAR(12) NOT NULL,
   A_mark FLOAT(5,2) NOT NULL
CHECK (A_mark >= 0 AND A_mark <= 100),
   A_no INT NOT NULL,

   PRIMARY KEY(C_code, C_type, Regno, A_no),

   -- Foreign key 1: Relate to the Course
   CONSTRAINT fk_AssignmentCourse FOREIGN KEY(C_code, C_type)
      REFERENCES Course(C_code, C_type)
      ON UPDATE CASCADE,

   -- Foreign key 2: Relate to the Student
   CONSTRAINT fk_AssignmentStudent FOREIGN KEY(Regno)
      REFERENCES User(User_id)
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

PROJECT

```
mysql> desc Project;
+--------+---------------+------+-----+---------+-------+
| Field  | Type          | Null | Key | Default | Extra |
+--------+---------------+------+-----+---------+-------+
| C_code | char(8)       | NO   | PRI | NULL    |       |
| C_type | enum('P','T') | NO   | PRI | NULL    |       |
| Regno  | char(12)      | NO   | PRI | NULL    |       |
| P_mark | float(9,2)    | NO   |     | NULL    |       |
+--------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
CREATE TABLE Project(
        C_code CHAR (8) NOT NULL,
        C_type ENUM('P','T') NOT NULL,
        Regno CHAR(12) NOT NULL,
        P_mark FLOAT(9,2) NOT NULL CHECK (P_mark >= 0 AND P_mark <= 100),

        PRIMARY KEY(C_code,C_type,Regno),

        -- Foreign key 1:Relate to the Course
        CONSTRAINT fk_Courseproject FOREIGN KEY(C_code,C_type) REFERENCES
Course(C_code,C_type),

        -- Foreign key 2:Relate to the Student
        CONSTRAINT fk_ProjectStudent FOREIGN KEY(Regno) REFERENCES
User(User_id)
 ON UPDATE CASCADE ON DELETE CASCADE
);
```

MID EXAM

```
mysql> desc Mid_exam;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| C_code   | char(8)       | NO   | PRI | NULL    |       |
| C_type   | enum('P','T') | NO   | PRI | NULL    |       |
| Regno    | char(12)      | NO   | PRI | NULL    |       |
| Mid_mark | float(9,2)    | NO   |     | NULL    |       |
+----------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
CREATE TABLE Mid_exam(
   C_code CHAR(8) NOT NULL,
   C_type ENUM('P','T') NOT NULL,
   Regno CHAR(12) NOT NULL,
   Mid_mark FLOAT(9,2) NOT NULL CHECK (Mid_mark >= 0 AND Mid_mark <= 100),

   PRIMARY KEY(C_code, C_type, Regno),

   -- Foreign key 1: Relate to the Course
   CONSTRAINT fk_MidCourse FOREIGN KEY(C_code, C_type)
      REFERENCES Course(C_code, C_type)
      ON UPDATE CASCADE,

   -- Foreign key 2: Relate to the Student
   CONSTRAINT fk_MidStudent FOREIGN KEY(Regno)
      REFERENCES User(User_id)
      ON UPDATE CASCADE ON
   DELETE CASCADE
   );
```

END EXAM

```
mysql> desc End_exam;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| C_code   | char(8)       | NO   | PRI | NULL    |       |
| C_type   | enum('P','T') | NO   | PRI | NULL    |       |
| Regno    | char(12)      | NO   | PRI | NULL    |       |
| End_mark | float(9,2)    | NO   |     | NULL    |       |
+----------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

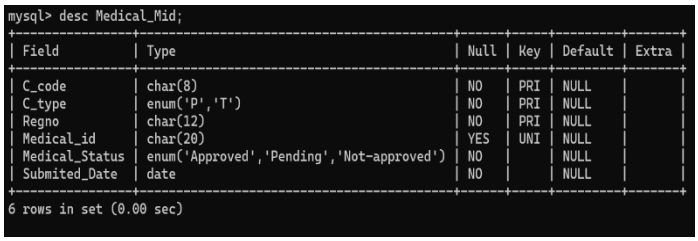
```
CREATE TABLE End_exam(
   C_code CHAR (8) NOT NULL,
   C_type ENUM('P','T') NOT NULL,
   Regno CHAR(12) NOT NULL,
   End_mark FLOAT(9,2) NOT NULL CHECK (End_mark >= 0 AND End_mark <= 100),

   PRIMARY KEY(C_code, C_type, Regno),

   -- Foreign key 1: Relate to the Course
   CONSTRAINT fk_EndCourse FOREIGN KEY(C_code, C_type)
      REFERENCES Course(C_code, C_type)
      ON UPDATE CASCADE,

   -- Foreign key 2: Relate to the Student
   CONSTRAINT fk_EndStudent FOREIGN KEY(Regno)
      REFERENCES User(User_id)
      ON UPDATE CASCADE ON DELETE CASCADE
   );
```

GRADE POINT

```
mysql> desc Grade_point;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| Grade | char(3)      | NO   | PRI | NULL    |       |
| Point | decimal(3,2) | NO   |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
CREATE TABLE Grade_point (
   Grade CHAR(3) NOT NULL PRIMARY KEY,
   Point DECIMAL(3, 2) NOT NULL CHECK (Point >= 0.00 AND Point <= 4.00)
);
```

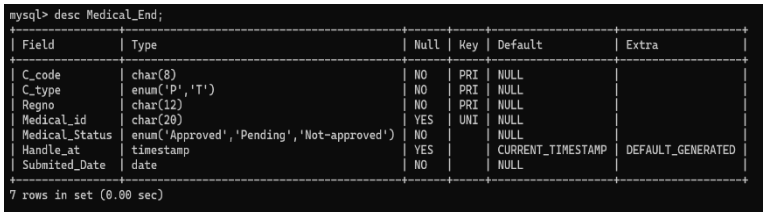
MEDICAL MID



```
mysql> desc Medical_Mid;
+----------------+-----------------------------------------+------+-----+---------+-------+
| Field          | Type                                    | Null | Key | Default | Extra |
+----------------+-----------------------------------------+------+-----+---------+-------+
| C_code         | char(8)                                 | NO   | PRI | NULL    |       |
| C_type         | enum('P','T')                           | NO   | PRI | NULL    |       |
| Regno          | char(12)                                | NO   | PRI | NULL    |       |
| Medical_id     | char(20)                                | YES  | UNI | NULL    |       |
| Medical_Status | enum('Approved','Pending','Not-approved') | NO   |     | NULL    |       |
| Submited_Date  | date                                    | NO   |     | NULL    |       |
+----------------+-----------------------------------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

```
CREATE TABLE Medical_Mid(
   C_code CHAR(8) NOT NULL,
   C_type ENUM('P','T') NOT NULL,
   Regno CHAR(12) NOT NULL,
   Medical_id CHAR(20) UNIQUE,
   Medical_Status ENUM('Approved','Pending','Not-approved') NOT NULL, -- Status should probably be NOT NULL
   Submited_Date DATE NOT NULL,                    -- Date should probably be NOT NULL

   PRIMARY KEY(Regno, C_code, C_type), -- Enforces one medical record per student per course's Mid exam

   -- Foreign key 1: Relate to the Course
   CONSTRAINT fk_Medical_MidCourse FOREIGN KEY(C_code, C_type)
      REFERENCES Course(C_code, C_type)
      ON UPDATE CASCADE,

   -- Foreign key 2: Relate to the Student
   CONSTRAINT fk_Medical_MidStudent FOREIGN KEY(Regno)
      REFERENCES User(User_id)
      ON UPDATE CASCADE
);
```

MEDICAL END



```
mysql> desc Medical_End;
+----------------+-----------------------------------------+------+-----+-------------------+-------------------+
| Field          | Type                                    | Null | Key | Default           | Extra             |
+----------------+-----------------------------------------+------+-----+-------------------+-------------------+
| C_code         | char(8)                                 | NO   | PRI | NULL              |                   |
| C_type         | enum('P','T')                           | NO   | PRI | NULL              |                   |
| Regno          | char(12)                                | NO   | PRI | NULL              |                   |
| Medical_id     | char(20)                                | YES  | UNI | NULL              |                   |
| Medical_Status | enum('Approved','Pending','Not-approved') | NO   |     | NULL              |                   |
| Handle_at      | timestamp                               | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| Submited_Date  | date                                    | NO   |     | NULL              |                   |
+----------------+-----------------------------------------+------+-----+-------------------+-------------------+
7 rows in set (0.00 sec)
```

```
CREATE TABLE Medical_End(
   C_code CHAR(8) NOT NULL,
   C_type ENUM('P','T') NOT NULL,
   Regno CHAR(12) NOT NULL,
   Medical_id CHAR(20) UNIQUE, -- Changed to UNIQUE, not PRIMARY KEY
   Medical_Status ENUM('Approved','Pending','Not-approved') NOT NULL,
   Handle_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   Submited_Date DATE NOT NULL,

   PRIMARY KEY(Regno, C_code, C_type), -- Enforces one medical record per student per course's End exam

   -- Foreign key 1: Relate to the Course
   CONSTRAINT fk_Medical_EndCourse FOREIGN KEY(C_code, C_type)
      REFERENCES Course(C_code, C_type)
      ON UPDATE CASCADE,

   -- Foreign key 2: Relate to the Student
   CONSTRAINT fk_Medical_EndStudent FOREIGN KEY(Regno)
      REFERENCES User(User_id)
      ON UPDATE CASCADE
);
```

MEDICAL ATTENDENCE

```
mysql> desc Medical_Attendence;
+----------------+-----------------------------------------------+------+-----+-------------------+-------------------+
| Field          | Type                                          | Null | Key | Default           | Extra             |
+----------------+-----------------------------------------------+------+-----+-------------------+-------------------+
| C_code         | char(8)                                       | NO   | PRI | NULL              |                   |
| C_type         | enum('P','T')                                 | NO   | PRI | NULL              |                   |
| Regno          | char(12)                                      | NO   | PRI | NULL              |                   |
| Medical_id     | char(20)                                      | YES  | UNI | NULL              |                   |
| Medical_Status | enum('Approved','Pending','Not-approved')     | NO   |     | NULL              |                   |
| Handle_at      | timestamp                                     | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| Submited_Date  | date                                          | NO   |     | NULL              |                   |
+----------------+-----------------------------------------------+------+-----+-------------------+-------------------+
7 rows in set (0.00 sec)
```

```
CREATE TABLE Medical_Attendence(
   C_code CHAR(8) NOT NULL,
   C_type ENUM('P','T') NOT NULL,
   Regno CHAR(12) NOT NULL,
   Medical_id CHAR(20) UNIQUE,
   Medical_Status
ENUM('Approved','Pending','Not-approved') NOT NULL,
   Handle_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   Submited_Date DATE NOT NULL,

   PRIMARY KEY(Regno, C_code, C_type),

   -- Foreign key 1: Relate to the Course
   CONSTRAINT fk_Medical_AttendenceCourse FOREIGN KEY(C_code, C_type)
      REFERENCES Course(C_code, C_type)
      ON UPDATE CASCADE,

   -- Foreign key 2: Relate to the Student
   CONSTRAINT fk_Medical_AttendenceStudent FOREIGN KEY(Regno)
      REFERENCES User(User_id)
      ON UPDATE CASCADE
);
```

MEDICAL HANDLE

Field	Type	Null	Key	Default	Extra
C_code	char(8)	NO	PRI	NULL	
C_type	enum('P','T')	NO	PRI	NULL	
Regno	char(12)	NO	PRI	NULL	
Medical_Type	enum('Mid','End','Attendance')	NO	PRI	NULL	
Lec_Handler	char(12)	NO	MUL	NULL	
Handle_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
CREATE TABLE Medical_handle(
   C_code CHAR(8) NOT NULL,
   C_type ENUM('P','T') NOT NULL,
   Regno CHAR(12) NOT NULL,
   Medical_Type ENUM('Mid', "End", "Attendence") NOT NULL,
   Lec_Handler VARCHAR(12) NOT NULL,
   Handle_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   PRIMARY KEY(Regno, C_code, C_type,Medical_Type),

   -- 2. Added FK to Course table
   CONSTRAINT fk_Medical_handle_course FOREIGN KEY(C_code, C_type)
      REFERENCES Course(C_code, C_type)
      ON UPDATE CASCADE,

   -- 3. Added FK to Student table
   CONSTRAINT fk_Medical_handle_student FOREIGN KEY(Regno)
      REFERENCES User(User_id)
      ON UPDATE CASCADE,

   -- Foreign key 04 : Related to Lecturer (Lec_Handler)
   CONSTRAINT fk_Medical_handle_lec FOREIGN KEY(Lec_Handler)
      REFERENCES User(User_id)
ON UPDATE CASCADE);
```

3.2 Architecture of Solution

This system uses classic three-layer architecture (or three-tier architecture) to manage and present faculty data efficiently. This design separates data storage, user presentation into independent components, enhancing maintainability and flexibility.

The layers are:

- Data Layer (MySQL): The foundation, responsible exclusively for storing the raw faculty data.
- Logic Layer (SQL Script/Procedures): The middle tier, responsible for defining the rules and operations (CREATE, READ, UPDATE, DELETE) that interact with Data Layer.
- Presentation Layer (Web/API): The top layer, responsible for the user interface or external communication, allowing clients to access the services provided by the Logic Layer.

3.3 Tools and Technologies that have use

- Draw.io: Used to draw ER diagram, relational schema.
- MySQL SERVER, VS code, Notepad: Used to create database and maintain.
- GitHub and GitHub Desktop: Version Control.
- Microsoft word : Use to create project report.

3.4 Security Measures

Multiple users are assigned different privileges to ensure database security.

- Admin: With All privileges with Grant Option for all the tables in the database
- Dean: With All privileges without Grant for all the tables in the database
- Lecturer: All privileges without Grant and user creation for all the tables in the database
- Technical Officer: Read, write, and update permissions for attendance related tables/views.
- Student: Read permission for final attendance and final marks/Grades tables/views

4.1 Introduce Users of Database

To ensure data security, integrity and controlled access several MySQL database user accounts were created in the system.

- ADMIN
  The admin account has all privileges with the GRANT OPTION. This allows the admin to manage the entire database, including creating new users and assigning privileges.

- DEAN
  The Dean account has all privileges without the GRANT OPTION. This allows the Dean to view and modify all tables but not create or modify user privileges.

- LECTURER
  The Lecturer account also has all privileges without the GRANT OPTION. Lecturers can manage student marks, attendance and results for their respective subjects but cannot alter user permissions.

- TECHNICAL OFFICER
  The Technical Officer has select, update, and insert permissions only for attendance-related tables and views. This allows the Technical Officer to manage attendance records efficiently while restricting access to other sensitive academic data.

- STUDENT
  The student account has select only access to final attendance records and results. Students can view their grades and attendance summaries but cannot modify any records.

4.2 Reasons for creating those users

- ADMIN
  To perform full system management, control user permissions and ensure the overall maintenance and security of the database.

- DEAN
  To allow the Dean to oversee academic and administrative operations without granting full system control.

- LECTURER
  To enable lecturers to handle their subject data while preventing unauthorized user management or privilege changes.

- TECHNICAL OFFICER
  To allow technical officer to manage attendance efficiently while restricting access to sensitive data such as marks or grades.

- STUDENT
  To provide transparency for students to view their academic progress while maintaining data confidentiality and protection.

5.1 Stored Procedures

```
DELIMITER //
        CREATE PROCEDURE CA_a_course_a_student (IN Reg_no CHAR(12),IN Ccode CHAR(8))
        BEGIN
           SELECT
              C_code AS 'Course Code',
             CASE
                 WHEN C_Type = 'T' THEN
        'Theory'
                 ELSE 'Practical'
              END  AS 'Practical/Theory',
               CA AS 'CA Marks'
           FROM Result_status
           WHERE Regno = Reg_no
           AND C_code = Ccode;
        END//

CALL CA_a_course_a_student('TG/2023/003','ICT1253')//
```

Course Code	Practical/Theory	CA Marks
ICT1253	Practical	38.00
ICT1253	Theory	71.43

```
DELIMITER //
        CREATE PROCEDURE Final_mark_all_course_a_student (IN Reg_no CHAR(12))
        BEGIN
           SELECT
              C_code AS 'Course Code',
              CASE
                 WHEN C_Type = 'T' THEN 'Theory'
                 ELSE 'Practical'
              END AS 'Practical/Theory',
               Final_Mark AS 'Final Marks'
           FROM Final_result
           WHERE Regno = Reg_no;
        END//

CALL Final_mark_all_course_a_student('TG/2023/004')//
```

Course Code	Practical/Theory	Final Marks
ENG1222	Theory	86.75
ICT1212	Theory	64.20
ICT1222	Practical	86.40
ICT1233	Practical	73.30
ICT1233	Theory	76.45
ICT1242	Theory	78.88
ICT1253	Practical	WH
ICT1253	Theory	81.90
TCS1212	Theory	86.20
TMS1233	Theory	84.80

```
DELIMITER //
	CREATE PROCEDURE CA_all_course_a_student (IN Reg_no CHAR(12))
	BEGIN
	  SELECT
	    C_code AS 'Course Code',
	    CASE
	      WHEN C_Type = 'T' THEN 'Theory'
	      ELSE 'Practical'
	    END AS 'Practical/Theory',
	     CA AS 'CA Marks'
	  FROM Result_status
	  WHERE Regno = Reg_no;
	END//

CALL CA_all_course_a_student('TG/2023/005')//
```

Course Code	Practical/Theory	CA Marks
ICT1212	Theory	58.33
ICT1222	Practical	73.83
ICT1233	Practical	55.63
ICT1233	Theory	51.17
ICT1242	Theory	50.00
ICT1253	Practical	23.75
ICT1253	Theory	41.50
TCS1212	Theory	60.67
TMS1233	Theory	54.07

```
DELIMITER //
	CREATE PROCEDURE CA_whole_batch_a_course (IN Ccode CHAR(8))
	BEGIN
	  SELECT
	    C_code AS 'Course Code',
	    Regno AS 'Registration Number',
	    CASE
	      WHEN C_Type = 'T' THEN 'Theory'
	      ELSE 'Practical'
	    END  AS 'Practical/Theory',
	     CA AS 'CA Marks'
	  FROM Result_status
	  WHERE C_code = Ccode;
	END//

CALL CA_whole_batch_a_course('ICT1253')//
```

Course Code	Registration Number	Practical/Theory	CA Marks
ICT1253	TG/2023/001	Practical	35.33
ICT1253	TG/2023/002	Practical	40.33
ICT1253	TG/2023/003	Practical	38.00
ICT1253	TG/2023/004	Practical	MC
ICT1253	TG/2023/005	Practical	23.75
ICT1253	TG/2023/006	Practical	21.25
ICT1253	TG/2023/007	Practical	17.25
ICT1253	TG/2023/008	Practical	17.75
ICT1253	TG/2023/009	Practical	21.25
ICT1253	TG/2023/010	Practical	22.63
ICT1253	TG/2023/001	Theory	63.77
ICT1253	TG/2023/002	Theory	79.00
ICT1253	TG/2023/003	Theory	71.43
ICT1253	TG/2023/004	Theory	64.17
ICT1253	TG/2023/005	Theory	41.50
ICT1253	TG/2023/006	Theory	43.00
ICT1253	TG/2023/007	Theory	MC
ICT1253	TG/2023/008	Theory	48.83
ICT1253	TG/2023/009	Theory	45.33
ICT1253	TG/2023/010	Theory	51.00

```
DELIMITER //
	CREATE PROCEDURE attendence_whole_batch_a_course (IN Ccode CHAR(8))
	BEGIN
	   SELECT
	      Regno AS 'Registration Number',
	      C_code AS 'Course Code',
	      CASE
	         WHEN C_Type = 'T' THEN 'Theory'
	         ELSE 'Practical'
	      END  AS 'Practical/Theory',

	      Attendance_Percentage AS 'Attendence
	Percentage',
	      Eligibility
	   FROM Attendance_eligibility
	   WHERE C_code = Ccode;
	END//
```

Registration Number	Course Code	Practical/Theory	Attendence Percentage	Eligibility
TG/2023/001	ICT1253	Practical	100.00	EL
TG/2023/002	ICT1253	Practical	92.86	EL
TG/2023/003	ICT1253	Practical	100.00	EL
TG/2023/004	ICT1253	Practical	100.00	EL
TG/2023/005	ICT1253	Practical	92.86	EL
TG/2023/006	ICT1253	Practical	100.00	EL
TG/2023/007	ICT1253	Practical	100.00	EL
TG/2023/008	ICT1253	Practical	100.00	EL
TG/2023/009	ICT1253	Practical	100.00	EL
TG/2023/010	ICT1253	Practical	100.00	EL
TG/2023/001	ICT1253	Theory	100.00	EL
TG/2023/002	ICT1253	Theory	100.00	EL
TG/2023/003	ICT1253	Theory	100.00	EL
TG/2023/004	ICT1253	Theory	100.00	EL
TG/2023/005	ICT1253	Theory	100.00	EL
TG/2023/006	ICT1253	Theory	100.00	EL
TG/2023/007	ICT1253	Theory	100.00	EL
TG/2023/008	ICT1253	Theory	100.00	EL
TG/2023/009	ICT1253	Theory	100.00	EL
TG/2023/010	ICT1253	Theory	100.00	EL

```
CALL attendence_whole_batch_a_course('ICT1253')//

DELIMITER //
	CREATE PROCEDURE attendence_whole_batch_all_course ()
	BEGIN
	   SELECT
	      Regno AS 'Registration Number',
	      C_code AS 'Course Code',
	      CASE
	         WHEN C_Type = 'T' THEN 'Theory'
	         ELSE 'Practical'
	      END  AS 'Practical/Theory',
	      Attendance_Percentage AS 'Attendence Percentage',
	      Eligibility
	   FROM Attendance_eligibility;
	END//

CALL attendence_whole_batch_all_course ()//
```

Registration Number	Course Code	Practical/Theory	Attendence Percentage	Eligibility
TG/2023/001	ENG1222	Theory	100.00	EL
TG/2023/001	ICT1212	Theory	100.00	EL
TG/2023/001	ICT1222	Practical	100.00	EL
TG/2023/001	ICT1233	Practical	100.00	EL
TG/2023/001	ICT1233	Theory	100.00	EL
TG/2023/001	ICT1242	Theory	100.00	EL
TG/2023/001	ICT1253	Practical	100.00	EL
TG/2023/001	ICT1253	Theory	100.00	EL
TG/2023/001	TCS1212	Theory	92.86	EL
TG/2023/001	TMS1233	Theory	100.00	EL
TG/2023/002	ENG1222	Theory	100.00	EL
TG/2023/002	ICT1212	Theory	100.00	EL
TG/2023/002	ICT1222	Practical	100.00	EL
TG/2023/002	ICT1233	Practical	100.00	EL
TG/2023/002	ICT1233	Theory	100.00	EL
TG/2023/002	ICT1242	Theory	100.00	EL
TG/2023/002	ICT1253	Practical	92.86	EL
TG/2023/002	ICT1253	Theory	100.00	EL
TG/2023/002	TCS1212	Theory	92.86	EL
TG/2023/002	TMS1233	Theory	100.00	EL
TG/2023/003	ENG1222	Theory	100.00	EL
TG/2023/003	ICT1212	Theory	100.00	EL
TG/2023/003	ICT1222	Practical	100.00	EL
TG/2023/003	ICT1233	Practical	100.00	EL
TG/2023/003	ICT1233	Theory	100.00	EL

```
DELIMITER //
	CREATE PROCEDURE medical_attendence (IN regnum CHAR(12))
	BEGIN
	   SELECT Regno,Submited_date,Medical_Status,C_code
	   FROM medical_Attendence
	   WHERE regno = regnum;
	END//

CALL medical_attendence('TG/2023/007')//
```

Regno	Submited_date	Medical_Status	C_code
TG/2023/007	2025-08-05	Approved	ENG1222

```
DELIMITER //
	CREATE PROCEDURE medical_end (IN regnum CHAR(12))
	BEGIN
	   SELECT Regno,Submited_date,Medical_Status,C_code
	   FROM medical_end
	   WHERE regno = regnum;
	END//

CALL medical_end('TG/2023/004')//
```

Regno	Submited_date	Medical_Status	C_code
TG/2023/004	2025-11-10	Approved	ICT1253

```
DELIMITER //
	CREATE PROCEDURE medical_mid (IN regnum CHAR(12))
	BEGIN

	   SELECT Regno,Submited_date,Medical_Status,C_code
	   FROM medical_mid
	   WHERE regno = regnum;
	END//

CALL medical_mid('TG/2023/002')//
```

Regno	Submited_date	Medical_Status	C_code
TG/2023/002	2025-10-01	Approved	ICT1233

```
DELIMITER //
	CREATE PROCEDURE Final_mark_whole_batch_all_course ()
	BEGIN
	   SELECT
	      Regno AS 'Registration Number',
	      C_code AS "Course Code",
	      CASE
	         WHEN C_Type = 'T' THEN 'Theory'
	         ELSE 'Practical'
	      END AS 'Practical/Theory',
	       Final_Mark AS 'Final Marks'
	   FROM Final_result;
	END//

CALL Final_mark_whole_batch_all_course()//
```

Registration Number	Course Code	Practical/Theory	Final Marks
TG/2021/023	ICT1242	Theory	78.43
TG/2021/088	TMS1233	Theory	82.75
TG/2023/001	ENG1222	Theory	91.00
TG/2023/001	ICT1212	Theory	88.25
TG/2023/001	ICT1222	Practical	68.29
TG/2023/001	ICT1233	Practical	77.90
TG/2023/001	ICT1233	Theory	78.25
TG/2023/001	ICT1242	Theory	79.75
TG/2023/001	ICT1253	Practical	46.50
TG/2023/001	ICT1253	Theory	82.13
TG/2023/001	TCS1212	Theory	84.95
TG/2023/001	TMS1233	Theory	71.55
TG/2023/002	ENG1222	Theory	78.35
TG/2023/002	ICT1212	Theory	76.38
TG/2023/002	ICT1222	Practical	63.46
TG/2023/002	ICT1233	Practical	73.45
TG/2023/002	ICT1233	Theory	WH
TG/2023/002	ICT1242	Theory	78.50
TG/2023/002	ICT1253	Practical	58.43
TG/2023/002	ICT1253	Theory	71.30
TG/2023/002	TCS1212	Theory	84.15
TG/2023/002	TMS1233	Theory	68.83
TG/2023/003	ENG1222	Theory	62.20
TG/2023/003	ICT1212	Theory	83.05

```
DELIMITER //
	CREATE PROCEDURE attendence_a_student_all_course (IN Reg_no CHAR(12))
	BEGIN
	   SELECT
	      C_code AS 'Course Code',
	      CASE
	         WHEN C_Type = 'T' THEN
	'Theory'
	            ELSE 'Practical'
	      END  AS 'Practical/Theory',
	      Attendance_Percentage AS
	'Attendence Percentage',
	      Eligibility
	   FROM Attendance_eligibility
	   WHERE  Regno = Reg_no;
	END//

CALL attendence_a_student_all_course('TG/2023/004')//
```

Course Code	Practical/Theory	Attendence Percentage	Eligibility
ENG1222	Theory	100.00	EL
ICT1212	Theory	100.00	EL
ICT1222	Practical	92.86	EL
ICT1233	Practical	100.00	EL
ICT1233	Theory	100.00	EL
ICT1242	Theory	100.00	EL
ICT1253	Practical	100.00	EL
ICT1253	Theory	100.00	EL
TCS1212	Theory	100.00	EL
TMS1233	Theory	100.00	EL

```
DELIMITER //
	CREATE PROCEDURE attendence_a_student_a_course (IN Ccode CHAR(8), IN Reg_no
	CHAR(12))
	BEGIN
	   SELECT
	      C_code AS 'Course Code',
	      CASE
	         WHEN C_Type = 'T'
	THEN 'Theory'
	            ELSE 'Practical'
	      END  AS 'Practical/Theory' ,
	      Attendance_Percentage AS 'Attendence Percentage',
	      Eligibility
	   FROM Attendance_eligibility
	   WHERE C_code = Ccode AND Regno = Reg_no;
	END//

CALL attendence_a_student_a_course ('ICT1253','TG/2023/002')//
```

Course Code	Practical/Theory	Attendence Percentage	Eligibility
ICT1253	Practical	92.86	EL
ICT1253	Theory	100.00	EL

5.2 Views

```sql
CREATE OR REPLACE VIEW Medical_Eligible_Attendance AS -- Renamed for clarity based on logic
-- 1. Calculate the total attended hours and total scheduled hours
WITH Attendance_Calculations AS (
    SELECT
        A.Regno,
        A.C_code,
        A.C_type,
        -- Calculate the total attended hours (Present or Medically Excused)
        SUM(CASE WHEN A.Status IN ('Present', 'Medical') THEN A.L_Hours ELSE 0 END) AS Total_Attended_Hours,
        -- Calculate the total scheduled hours
        SUM(A.L_Hours) AS Total_Scheduled_Hours
    FROM
        Attendence A
    GROUP BY
        A.Regno,
        A.C_code,
        A.C_type
),
-- 2. Calculate Percentage (Corrected syntax: comma added after first CTE, comma removed in SELECT list)
Percentage AS (
    SELECT
        AC.Regno,
        AC.C_code,
        AC.C_type,
        -- Calculate Attendance Percentage based on total hours
        ROUND(
            (1.0 * AC.Total_Attended_Hours / NULLIF(AC.Total_Scheduled_Hours, 0)) * 100,
            2
        ) AS Attendance_Percentage
    FROM
        Attendance_Calculations AC
)
```

```sql
-- 3. Final Select: Filter for students meeting the 80% threshold
SELECT
    p.Regno,
    p.C_code,
    p.C_type,
    p.Attendance_Percentage
FROM
    Percentage p
WHERE
    p.Attendance_Percentage >= 80.00;
```

Regno	C_code	C_type	Attendance_Percentage
TG/2023/001	ENG1222	T	100.00
TG/2023/001	ICT1212	T	100.00
TG/2023/001	ICT1222	P	100.00
TG/2023/001	ICT1233	P	100.00
TG/2023/001	ICT1233	T	100.00
TG/2023/001	ICT1242	T	100.00
TG/2023/001	ICT1253	P	100.00
TG/2023/001	ICT1253	T	100.00
TG/2023/001	TCS1212	T	92.86
TG/2023/001	TMS1233	T	100.00
TG/2023/002	ENG1222	T	100.00
TG/2023/002	ICT1212	T	100.00
TG/2023/002	ICT1222	P	100.00
TG/2023/002	ICT1233	P	100.00
TG/2023/002	ICT1233	T	100.00
TG/2023/002	ICT1242	T	100.00
TG/2023/002	ICT1253	P	92.86
TG/2023/002	ICT1253	T	100.00
TG/2023/002	TCS1212	T	92.86
TG/2023/002	TMS1233	T	100.00
TG/2023/003	ENG1222	T	100.00
TG/2023/003	ICT1212	T	100.00
TG/2023/003	ICT1222	P	100.00
TG/2023/003	ICT1233	P	100.00
TG/2023/003	ICT1233	T	100.00
TG/2023/003	ICT1242	T	100.00
TG/2023/003	ICT1253	P	100.00
TG/2023/003	ICT1253	T	100.00
TG/2023/003	TCS1212	T	100.00
TG/2023/003	TMS1233	T	92.86

```
+--------------+---------+--------+-------------+-------+
| Regno        | C_code  | C_type | Final_Mark | Grade |
+--------------+---------+--------+-------------+-------+
| TG/2021/023 | ICT1242 | T      | 78.43       | A     |
| TG/2021/088 | TMS1233 | T      | 82.75       | A     |
| TG/2023/001 | ENG1222 | T      | 91.00       | A+    |
| TG/2023/001 | ICT1212 | T      | 88.25       | A+    |
| TG/2023/001 | ICT1222 | P      | 68.29       | B+    |
| TG/2023/001 | ICT1233 | P      | 77.90       | A     |
| TG/2023/001 | ICT1233 | T      | 78.25       | A     |
| TG/2023/001 | ICT1242 | T      | 79.75       | A     |
| TG/2023/001 | ICT1253 | P      | 46.50       | E     |
| TG/2023/001 | ICT1253 | T      | 82.13       | A     |
| TG/2023/001 | TCS1212 | T      | 84.95       | A     |
| TG/2023/001 | TMS1233 | T      | 71.55       | A-    |
| TG/2023/002 | ENG1222 | T      | 78.35       | A     |
| TG/2023/002 | ICT1212 | T      | 76.38       | A     |
| TG/2023/002 | ICT1222 | P      | 63.46       | B     |
| TG/2023/002 | ICT1233 | P      | 73.45       | A-    |
| TG/2023/002 | ICT1233 | T      | WH          | WH    |
| TG/2023/002 | ICT1242 | T      | 78.50       | A     |
| TG/2023/002 | ICT1253 | P      | 58.43       | B-    |
| TG/2023/002 | ICT1253 | T      | 71.30       | A-    |
| TG/2023/002 | TCS1212 | T      | 84.15       | A     |
| TG/2023/002 | TMS1233 | T      | 68.83       | B+    |
| TG/2023/003 | ENG1222 | T      | 62.20       | B     |
| TG/2023/003 | ICT1212 | T      | 83.05       | A     |
| TG/2023/003 | ICT1222 | P      | 84.19       | A     |
```

```sql
        -- Priority 3: Grading logic based on final score
        WHEN FT.S_type IN ('Proper', 'Repeat') AND FT.Final_Total_Mark IS NOT NULL THEN
            CASE
                WHEN FT.Final_Total_Mark >= 85 THEN 'A+'
                WHEN FT.Final_Total_Mark >= 75 THEN 'A'
                WHEN FT.Final_Total_Mark >= 70 THEN 'A-'
                WHEN FT.Final_Total_Mark >= 65 THEN 'B+'
                WHEN FT.Final_Total_Mark >= 60 THEN 'B'
                WHEN FT.Final_Total_Mark >= 55 THEN 'B-'
                WHEN FT.Final_Total_Mark >= 50 THEN 'C+'
                WHEN FT.Final_Total_Mark >= 45 THEN 'C'
                WHEN FT.Final_Total_Mark >= 40 THEN 'C-'
                WHEN FT.Final_Total_Mark >= 35 THEN 'D'
                ELSE 'E'
            END

        -- Remaining scenarios
        ELSE 'F'
    END AS Grade

FROM
    Final_Total FT;
```

```sql
    -- Final Display Mark Logic
    CASE
        -- Priority 1: Overriding Status Checks (Suspending/MC/NE results in 'WH')
        WHEN FT.S_type = 'Suspend' THEN 'WH'
        WHEN FT.End_Status IN ( 'MC' , 'NE') OR FT.CA_Mark_Display = 'MC' THEN 'WH'

        -- Priority 2: Use the Numeric Total Mark
        WHEN FT.Final_Total_Mark IS NOT NULL THEN CAST(ROUND(FT.Final_Total_Mark, 2) AS CHAR(10))
        ELSE 'N/A'
    END AS Final_Mark,

    -- Final Grade Logic
    CASE
        -- Priority 1: Status checks that result in non-grade codes
        WHEN FT.S_type = 'Suspend' THEN 'WH'
        WHEN FT.End_Status IN ( 'MC' , 'NE') OR FT.CA_Mark_Display = 'MC' THEN 'WH'

        -- Priority 2: Automatic Failure (E) for component failure
        WHEN FT.CA_Status = 'Fail' OR FT.End_Status = 'Fail' THEN 'E'
```

```
    FROM
        Result_status RS
    INNER JOIN Student S
        ON RS.Regno = S.Reg_no -- Using S.Reg_no to match the Student table PRIMARY KEY
),

-- CTE to Calculate Final Total Mark
Final_Total AS (
    SELECT
        C.*,

        -- Calculate the Numeric Total Mark
        CASE
            -- Total mark is NULL if status is WH, MC, or NE
            WHEN C.End_Status IN ('NE', 'MC') OR C.CA_Mark_Display = 'MC' THEN NULL

            -- Otherwise, sum the component contributions (even if they are 0 due to failure)
            ELSE C.CA_Contribution + C.End_Contribution
        END AS Final_Total_Mark

    FROM Calculation_CTE C
)

-- Final SELECT to determine final mark and grade.
SELECT
    FT.Regno,
    FT.C_code,
    FT.C_type,

    -- Final Display Mark Logic
```

```
-- Final Result

CREATE OR REPLACE VIEW Final_result AS
WITH Calculation_CTE AS (
    SELECT
        RS.Regno,
        RS.C_code,
        RS.C_type,
        S.S_type,

        -- Status Fields from Result_status
        RS.CA AS CA_Mark_Display,
        RS.End_mark AS End_Mark_Display,
        RS.CA_Status,
        RS.End_Status,

        -- CA Contribution: (Scaled CA mark * Weight)
        CASE
            WHEN RS.CA_Status = 'Pass' AND RS.C_type = 'T' THEN CAST(RS.CA AS DECIMAL(10,2)) * 0.30
            WHEN RS.CA_Status = 'Pass' AND RS.C_type = 'P' THEN CAST(RS.CA AS DECIMAL(10,2)) * 0.40
            ELSE 0.00
        END AS CA_Contribution,

        -- End Contribution: (End Mark * Weight)
        CASE
            WHEN RS.End_Status = 'Pass' AND RS.C_type = 'T' THEN CAST(RS.End_mark AS DECIMAL(10,2)) * 0.70
            WHEN RS.End_Status = 'Pass' AND RS.C_type = 'P' THEN CAST(RS.End_mark AS DECIMAL(10,2)) * 0.60
            ELSE 0.00
        END AS End_Contribution
```

Regno	C_code	C_type	Attendance_Percentage	Eligibility
TG/2023/001	ENG1222	T	100.00	EL
TG/2023/001	ICT1212	T	100.00	EL
TG/2023/001	ICT1222	P	100.00	EL
TG/2023/001	ICT1233	P	100.00	EL
TG/2023/001	ICT1233	T	100.00	EL
TG/2023/001	ICT1242	T	100.00	EL
TG/2023/001	ICT1253	P	100.00	EL
TG/2023/001	ICT1253	T	100.00	EL
TG/2023/001	TCS1212	T	92.86	EL
TG/2023/001	TMS1233	T	100.00	EL
TG/2023/002	ENG1222	T	100.00	EL
TG/2023/002	ICT1212	T	100.00	EL
TG/2023/002	ICT1222	P	100.00	EL
TG/2023/002	ICT1233	P	100.00	EL
TG/2023/002	ICT1233	T	100.00	EL
TG/2023/002	ICT1242	T	100.00	EL
TG/2023/002	ICT1253	P	92.86	EL
TG/2023/002	ICT1253	T	100.00	EL
TG/2023/002	TCS1212	T	92.86	EL
TG/2023/002	TMS1233	T	100.00	EL
TG/2023/003	ENG1222	T	100.00	EL
TG/2023/003	ICT1212	T	100.00	EL
TG/2023/003	ICT1222	P	100.00	EL
TG/2023/003	ICT1233	P	100.00	EL
TG/2023/003	ICT1233	T	100.00	EL
TG/2023/003	ICT1242	T	100.00	EL
TG/2023/003	ICT1253	P	100.00	EL

```
    CASE
        -- Check standard percentage threshold (80%)
        WHEN (1.0 * AC.Total_Attended_Hours / AC.Total_Scheduled_Hours) * 100 >= 80
            THEN 'EL' -- Eligible

        -- Otherwise, Not Eligible
        ELSE 'NE'
    END AS Eligibility
FROM
    Attendance_Calculations AC
LEFT JOIN
    Medical_Attendence MA
    ON AC.Regno = MA.Regno AND AC.C_code = MA.C_code AND AC.C_type = MA.C_type;
```

```
-- Attendence Eligibility

CREATE OR REPLACE VIEW Attendance_eligibility AS
WITH Attendance_Calculations AS (
    SELECT
        A.Regno,
        A.C_code,
        A.C_type,
        -- Calculate the total attended hours (Present or Medically Excused)
        SUM(CASE WHEN A.Status IN ('Present', 'Medical') THEN A.L_Hours ELSE 0 END) AS Total_Attended_Hours,
        -- Calculate the total scheduled hours
        SUM(A.L_Hours) AS Total_Scheduled_Hours
    FROM
        Attendence A
    GROUP BY
        A.Regno,
        A.C_code,
        A.C_type
)

SELECT
    AC.Regno,
    AC.C_code,
    AC.C_type,
    -- Calculate Attendance Percentage based on total hours
    ROUND(
        (1.0 * AC.Total_Attended_Hours / AC.Total_Scheduled_Hours) * 100,
        2
    ) AS Attendance_Percentage,
```

```
-- Attendence > 80 %

CREATE OR REPLACE VIEW Eligible_Attendance AS -- Renamed for clarity based on logic
-- 1. Calculate the total attended hours and total scheduled hours
WITH Attendance_Calculations AS (
    SELECT
        A.Regno,
        A.C_code,
        A.C_type,
        -- Calculate the total attended hours (Present or Medically Excused)
        SUM(CASE WHEN A.Status IN ('Present') THEN A.L_Hours ELSE 0 END) AS Total_Attended_Hours,
        -- Calculate the total scheduled hours
        SUM(A.L_Hours) AS Total_Scheduled_Hours
    FROM
        Attendence A
    GROUP BY
        A.Regno,
        A.C_code,
        A.C_type
),

-- 2. Calculate Percentage (Corrected syntax: comma added after first CTE, comma removed in SELECT list)
Percentage AS (
    SELECT
        AC.Regno,
        AC.C_code,
        AC.C_type,
        -- Calculate Attendance Percentage based on total hours
        ROUND(
            (1.0 * AC.Total_Attended_Hours / NULLIF(AC.Total_Scheduled_Hours, 0)) * 100,
            2
        ) AS Attendance_Percentage
```

Regno	C_code	C_type	Attendance_Percentage
TG/2023/006	ICT1233	P	78.57

```
)

-- 3. Final Select: Filter for students meeting the 80% threshold
SELECT
    p.Regno,
    p.C_code,
    p.C_type,
    p.Attendance_Percentage
FROM
    Percentage p
WHERE
    p.Attendance_Percentage < 80.00;
```

```
CREATE OR REPLACE VIEW Non_Eligible_Attendance AS -- Renamed for clarity based on logic
-- 1. Calculate the total attended hours and total scheduled hours
WITH Attendance_Calculations AS (
    SELECT
        A.Regno,
        A.C_code,
        A.C_type,
        -- Calculate the total attended hours (Present or Medically Excused)
        SUM(CASE WHEN A.Status IN ('Present') THEN A.L_Hours ELSE 0 END) AS Total_Attended_Hours,
        -- Calculate the total scheduled hours
        SUM(A.L_Hours) AS Total_Scheduled_Hours
    FROM
        Attendence A
    GROUP BY
        A.Regno,
        A.C_code,
        A.C_type
),

-- 2. Calculate Percentage (Corrected syntax: comma added after first CTE, comma removed in SELECT list)
Percentage AS (
    SELECT
        AC.Regno,
        AC.C_code,
        AC.C_type,
        -- Calculate Attendance Percentage based on total hours
        ROUND(
            (1.0 * AC.Total_Attended_Hours / NULLIF(AC.Total_Scheduled_Hours, 0)) * 100,
            2
        ) AS Attendance_Percentage
    FROM
        Attendance_Calculations AC
```

Regno	C_code	C_type	Attendance_Percentage
TG/2023/001	ENG1222	T	100.00
TG/2023/001	ICT1212	T	100.00
TG/2023/001	ICT1222	P	100.00
TG/2023/001	ICT1233	P	100.00
TG/2023/001	ICT1233	T	100.00
TG/2023/001	ICT1242	T	100.00
TG/2023/001	ICT1253	P	100.00
TG/2023/001	ICT1253	T	100.00
TG/2023/001	TCS1212	T	92.86
TG/2023/001	TMS1233	T	100.00
TG/2023/002	ENG1222	T	100.00
TG/2023/002	ICT1212	T	100.00
TG/2023/002	ICT1222	P	100.00
TG/2023/002	ICT1233	P	100.00
TG/2023/002	ICT1233	T	100.00
TG/2023/002	ICT1242	T	100.00
TG/2023/002	ICT1253	P	92.86
TG/2023/002	ICT1253	T	100.00
TG/2023/002	TCS1212	T	92.86
TG/2023/002	TMS1233	T	100.00
TG/2023/003	ENG1222	T	100.00
TG/2023/003	ICT1212	T	100.00
TG/2023/003	ICT1222	P	92.86
TG/2023/003	ICT1233	P	100.00
TG/2023/003	ICT1233	T	100.00
TG/2023/003	ICT1242	T	100.00
TG/2023/003	ICT1253	P	100.00

```sql
        -- Calculate Attendance Percentage based on total hours
        ROUND(
            (1.0 * AC.Total_Attended_Hours / NULLIF(AC.Total_Scheduled_Hours, 0)) * 100,
            2
        ) AS Attendance_Percentage
    FROM
        Attendance_Calculations AC
)

-- 3. Final Select: Filter for students meeting the 80% threshold
SELECT
    p.Regno,
    p.C_code,
    p.C_type,
    p.Attendance_Percentage
FROM
    Percentage p
WHERE
    p.Attendance_Percentage >= 80.00;
```

```sql
CREATE OR REPLACE VIEW GPA AS
-- CTE to map Grade to Point (Assumes Grade_point table exists)
WITH GradeCalc AS (
    SELECT
        R.Regno,
        R.C_code,
        R.C_type,
        R.Grade,

        -- Corrected: Join the Course table on both C_code and C_type
        CAST(C.Credit AS DECIMAL(5,2)) AS Credit,

        -- Determine the Point for calculation
        CASE
            -- Grades that should result in 0 points but ARE counted in the credit total (NE, E, D, F)
            WHEN R.Grade IN ('NE', 'E', 'D', 'F') THEN 0.00

            -- Grades that must be EXCLUDED from both points and credit totals (WH, MC)
            WHEN R.Grade IN ('WH', 'MC') THEN NULL

            -- Use the standard point for passing grades
            ELSE G.Point
        END AS Calculated_Point

    FROM
        Final_result R
    INNER JOIN
        Course C ON R.C_code = C.C_code AND R.C_type = C.C_type -- CORRECTED JOIN
    LEFT JOIN
        Grade_point G ON R.Grade = G.Grade
),

CreditTotals AS (
    SELECT
        Regno,

        -- Flag: Check only for 'WH' or 'MC' (These stop the numeric GPA output)
        MAX(CASE WHEN R.Grade IN ('WH', 'MC') THEN 1 ELSE 0 END) AS HasWithheldGrade,
```

```sql
CreditTotals AS (
    SELECT
        Regno,

        -- Flag: Check only for 'WH' or 'MC' (These stop the numeric GPA output)
        MAX(CASE WHEN R.Grade IN ('WH', 'MC') THEN 1 ELSE 0 END) AS HasWithheldGrade,

        -- 1. Total Credit for SGPA (CORRECTED: Excludes credits from WH/MC grades)
        SUM(
            CASE WHEN R.Grade IN ('WH', 'MC') THEN 0.00
            ELSE Credit
            END
        ) AS TotalCredit_SGPA,

        -- 2. Total Points for SGPA
        -- COALESCE handles NULL Calculated_Point from WH/MC, setting their point contribution to 0.00.
        SUM(COALESCE(Calculated_Point * Credit, 0.00)) AS TotalPoints_SGPA,

        -- 3. Total Credit for CGPA (CORRECTED: Excludes 'ENG1222' AND credits from WH/MC grades)
        SUM(
            CASE
                WHEN R.C_code = 'ENG1222' OR R.Grade IN ('WH', 'MC') THEN 0.00
                ELSE Credit
            END
        ) AS TotalCredit_CGPA,

        -- 4. Total Points for CGPA (Excludes 'ENG1222')
        SUM(
            CASE
                WHEN R.C_code = 'ENG1222' THEN 0.00
                ELSE COALESCE(Calculated_Point * Credit, 0.00)
            END
        ) AS TotalPoints_CGPA

    FROM
        GradeCalc R
    GROUP BY
        Regno
)
```

```sql
    SELECT
        Regno,
        --   SGPA Calculation
        CASE
            WHEN HasWithheldGrade = 1 THEN 'WH'
            WHEN TotalCredit_SGPA = 0 THEN 'N/A'
            ELSE
                CAST(
                    ROUND(
                        TotalPoints_SGPA / TotalCredit_SGPA,
                        2
                    )
                AS CHAR(10))
        END AS SGPA,

        --   CGPA Calculation
        CASE
            WHEN HasWithheldGrade = 1 THEN 'WH'
            WHEN TotalCredit_CGPA = 0 THEN 'N/A'
            ELSE
                CAST(
                    ROUND(
                        TotalPoints_CGPA / TotalCredit_CGPA,
                        2
                    )
                AS CHAR(10))
        END AS CGPA
    FROM
        CreditTotals;
```

Regno	C_code	C_type	Final_Mark	Grade
TG/2021/023	ICT1242	T	78.43	A
TG/2021/088	TMS1233	T	82.75	A
TG/2023/001	ENG1222	T	91.00	A+
TG/2023/001	ICT1212	T	88.25	A+
TG/2023/001	ICT1222	P	68.29	B+
TG/2023/001	ICT1233	P	77.90	A
TG/2023/001	ICT1233	T	78.25	A
TG/2023/001	ICT1242	T	79.75	A
TG/2023/001	ICT1253	P	46.50	E
TG/2023/001	ICT1253	T	82.13	A
TG/2023/001	TCS1212	T	84.95	A
TG/2023/001	TMS1233	T	71.55	A-
TG/2023/002	ENG1222	T	78.35	A
TG/2023/002	ICT1212	T	76.38	A
TG/2023/002	ICT1222	P	63.46	B
TG/2023/002	ICT1233	P	73.45	A-
TG/2023/002	ICT1233	T	WH	WH
TG/2023/002	ICT1242	T	78.50	A
TG/2023/002	ICT1253	P	58.43	B-
TG/2023/002	ICT1253	T	71.30	A-
TG/2023/002	TCS1212	T	84.15	A
TG/2023/002	TMS1233	T	68.83	B+
TG/2023/003	ENG1222	T	62.20	B
TG/2023/003	ICT1212	T	83.05	A
TG/2023/003	ICT1222	P	84.19	A
TG/2023/003	ICT1233	P	64.55	B
TG/2023/003	ICT1233	T	70.68	A-
TG/2023/003	ICT1242	T	66.03	B+
TG/2023/003	ICT1253	P	46.80	E
TG/2023/003	ICT1253	T	65.53	B+
TG/2023/003	TCS1212	T	85.03	A+
TG/2023/003	TMS1233	T	84.75	A
TG/2023/004	ENG1222	T	86.75	A+
TG/2023/004	ICT1212	T	64.20	B

```sql
-- result status
CREATE OR REPLACE VIEW Result_status AS
-- Rank and get top 2 quiz marks
WITH ranked_quiz AS (
    SELECT
        Regno, C_code, C_type, mark,
        ROW_NUMBER() OVER (
            PARTITION BY Regno, C_code, C_type
            ORDER BY COALESCE(mark, 0) DESC
        ) AS rn
    FROM Quiz
),

quiz_mark AS (
    SELECT
        Regno, C_code, C_type,
        AVG(mark) AS quiz_mark
    FROM ranked_quiz
    WHERE rn <= 2
    GROUP BY Regno, C_code, C_type
),

Assignment_mark AS (
    SELECT
        Regno, C_code, C_type,
        SUM(COALESCE(A_mark, 0)) AS Ass_mark
    FROM Assignment
    GROUP BY Regno, C_code, C_type
),

Project_mark AS (
    SELECT
        Regno, C_code, C_type,
        MAX(COALESCE(P_mark, 0)) AS Project_mark
    FROM Project
    GROUP BY Regno, C_code, C_type
),
```

```sql
-- Base CTE joins all components (marks, medicals, attendance)
Base_CA AS (
    SELECT
        en.Regno,
        en.C_code,
        en.C_type,
        q.quiz_mark,
        a.Ass_mark,
        m.Mid_mark,
        p.Project_mark,
        e.End_mark,
        mm.Medical_status AS Mid_Medical_status,
        me.Medical_Status AS End_Medical_status,
        ae.Eligibility AS Attendance_Eligibility

    FROM Course_Enrollment en
    LEFT JOIN quiz_mark q
        ON en.Regno = q.Regno AND en.C_code = q.C_code AND en.C_type = q.C_type
    LEFT JOIN Assignment_mark a
        ON en.Regno = a.Regno AND en.C_code = a.C_code AND en.C_type = a.C_type
    LEFT JOIN Mid_exam m
        ON en.Regno = m.Regno AND en.C_code = m.C_code AND en.C_type = m.C_type
    LEFT JOIN Project_mark p
        ON en.Regno = p.Regno AND en.C_code = p.C_code AND en.C_type = p.C_type
    LEFT JOIN End_exam e
        ON en.Regno = e.Regno AND en.C_code = e.C_code AND en.C_type = e.C_type
    LEFT JOIN Medical_mid mm
        ON en.Regno = mm.Regno AND en.C_code = mm.C_code AND en.C_type = mm.C_type
    LEFT JOIN Medical_End me
        ON en.Regno = me.Regno AND en.C_code = me.C_code AND en.C_type = me.C_type
    LEFT JOIN Attendance_eligibility ae
        ON en.Regno = ae.Regno AND en.C_code = ae.C_code AND en.C_type = ae.C_type
),
```

```sql
-- CTE calculates CA mark (as a decimal) or assigns 'MC'
Calculated_CA AS (
    SELECT
        c.Regno,
        c.C_code,
        c.C_type,
        c.End_mark,
        c.End_Medical_status,
        c.Attendance_Eligibility,

        -- CA Calculation and Mid MC Check
        CASE
            WHEN c.Mid_Medical_status IN ('Approved', 'Pending') THEN 'MC'
            ELSE
                CAST(
                    CASE
                        WHEN c.C_code IN ('ICT1242' , 'TCS1212' , 'TMS1233') THEN COALESCE(c.quiz_mark, 0) * 0.1 + COALESCE(c.Ass_mark, 0) * 0.05 +
                        COALESCE(c.Mid_mark, 0) * 0.15
                        WHEN c.C_code = 'ICT1212' THEN COALESCE(c.quiz_mark, 0) * 0.1 + COALESCE(c.Mid_mark, 0) * 0.2
                        WHEN c.C_code = 'ICT1222' THEN COALESCE(c.quiz_mark, 0) * 0.05 + COALESCE(c.Mid_mark, 0) * 0.15 + COALESCE(c.Project_mark, 0) * 0.2
                        WHEN c.C_code IN ('ICT1233','ICT1253') THEN COALESCE(c.quiz_mark, 0) * 0.1 + COALESCE(c.Mid_mark, 0) * 0.1 + COALESCE(c.
                        Project_mark, 0) * 0.15 + COALESCE(c.Ass_mark, 0) * 0.1
                        ELSE COALESCE(c.Mid_mark, 0) * 0.2 + COALESCE(c.Ass_mark, 0) * 0.1
                    END
                AS DECIMAL(10, 2))
        END AS CA
    FROM Base_CA c
),
```

```sql
-- CTE to scale the CA mark for the 40-point pass check
Final_Marks AS (
    SELECT
        c.*,
        CASE
            WHEN c.CA = 'MC' THEN NULL
            WHEN c.C_type = 'T' THEN CAST(c.CA AS DECIMAL(10,2)) * (100.0 / 30.0)
            WHEN c.C_type = 'P' THEN CAST(c.CA AS DECIMAL(10,2)) * (100.0 / 40.0)
            ELSE CAST(c.CA AS DECIMAL(10,2))
        END AS Scaled_CA_Mark
    FROM Calculated_CA c
)

-- Final SELECT to determine status based on scaled marks
SELECT
    f.Regno,
    f.C_code,
    f.C_type,

    -- Final CA Mark (Display) - FIX: Use ROUND(..., 2)
    CASE
        WHEN f.CA = 'MC' THEN 'MC'
        -- Rounds the mark to 2 decimal places before casting to a display string
        ELSE CAST(ROUND(f.Scaled_CA_Mark, 2) AS CHAR(10))
    END AS CA,

    -- CA Status Check (Uses the Scaled_CA_Mark for >= 40 check)
    CASE
        WHEN f.CA = 'MC' THEN 'MC'
        WHEN f.Scaled_CA_Mark >= 40.0 THEN 'Pass'
        ELSE 'Fail'
    END AS CA_Status,
```

```sql
    -- End_mark Output (Mark or Status)
    CASE
        WHEN f.Attendance_Eligibility = 'NE' THEN 'NE'
        WHEN f.End_Medical_status IN ('Approved', 'Pending') THEN 'MC'
        WHEN f.End_mark IS NULL THEN '0.00'
        ELSE CAST(f.End_mark AS CHAR(10))
    END AS End_mark,

    -- End Status Check
    CASE
        WHEN f.Attendance_Eligibility = 'NE' THEN 'NE'
        WHEN f.End_Medical_status IN ('Approved', 'Pending') THEN 'MC'
        WHEN f.End_mark IS NULL THEN 'Fail'
        WHEN CAST(f.End_mark AS DECIMAL(10,2)) >= 35 THEN 'Pass'
        ELSE 'Fail'
    END AS End_Status

FROM Final_Marks f;
```

Regno	C_code	C_type	CA	CA_Status	End_mark	End_Status
TG/2020/010	TCS1212	T	0.00	Fail	0.00	Fail
TG/2021/006	TMS1233	T	0.00	Fail	0.00	Fail
TG/2021/023	ICT1242	T	56.10	Pass	88.00	Pass
TG/2021/088	TMS1233	T	54.17	Pass	95.00	Pass
TG/2023/001	ENG1222	T	81.67	Pass	95.00	Pass
TG/2023/001	ICT1212	T	84.17	Pass	90.00	Pass
TG/2023/001	ICT1222	P	77.73	Pass	62.00	Pass
TG/2023/001	ICT1233	P	62.00	Pass	88.50	Pass
TG/2023/001	ICT1233	T	74.17	Pass	80.00	Pass
TG/2023/001	ICT1242	T	74.50	Pass	82.00	Pass
TG/2023/001	ICT1253	P	35.33	Fail	77.50	Pass
TG/2023/001	ICT1253	T	63.77	Pass	90.00	Pass
TG/2023/001	TCS1212	T	79.00	Pass	87.50	Pass
TG/2023/001	TMS1233	T	86.83	Pass	65.00	Pass
TG/2023/002	ENG1222	T	85.00	Pass	75.50	Pass
TG/2023/002	ICT1212	T	84.27	Pass	73.00	Pass
TG/2023/002	ICT1222	P	80.65	Pass	52.00	Pass
TG/2023/002	ICT1233	P	51.63	Pass	88.00	Pass
TG/2023/002	ICT1233	T	MC	MC	78.00	Pass
TG/2023/002	ICT1242	T	82.00	Pass	77.00	Pass
TG/2023/002	ICT1253	P	40.33	Pass	70.50	Pass
TG/2023/002	ICT1253	T	79.00	Pass	68.00	Pass
TG/2023/002	TCS1212	T	77.50	Pass	87.00	Pass
TG/2023/002	TMS1233	T	85.93	Pass	61.50	Pass
TG/2023/003	ENG1222	T	44.00	Pass	70.00	Pass
TG/2023/003	ICT1212	T	90.17	Pass	80.00	Pass
TG/2023/003	ICT1222	P	82.98	Pass	85.00	Pass
TG/2023/003	ICT1233	P	72.13	Pass	59.50	Pass
TG/2023/003	ICT1233	T	75.77	Pass	68.50	Pass
TG/2023/003	ICT1242	T	83.60	Pass	58.50	Pass
TG/2023/003	ICT1253	P	38.00	Fail	78.00	Pass
TG/2023/003	ICT1253	T	71.43	Pass	63.00	Pass
TG/2023/003	TCS1212	T	80.43	Pass	87.00	Pass
TG/2023/003	TMS1233	T	90.00	Pass	82.50	Pass
TG/2023/004	ENG1222	T	82.67	Pass	88.50	Pass

6.1 Problems

- Complexity of Table Structures
  Creating tables like attendance was challenging, especially in in determining appropriate foreign keys.

- Data Type Errors
  Errors related to incompatible data types arose when inserting data into tables.

- Diagram Adjustments
  Frequent modifications were needed for the ER diagram and relational map to align with system requirements.

- Data Maintenance
  Managing and updating data in tables was cumbersome and prone to errors.

6.2 Solutions

- Modular Data Sets :
  Divided attendance data into separate files by weekdays to streamline entry and reduce complexity.

- Automated Data Typing:
  Implemented repetitive typing checks for attendance data to ensure consistency and minimize input errors.

- Dynamic Diagram Updates:
  Updated the ER diagram and relational map as needed to reflect new requirements, ensuring alignment with database structure.

- Data Validation Checks:
  Introduced pre-insertion data validation to catch and resolve data type mismatches early.

7. Backend Hosting Choices & Justification

If we were to host the database backend, we would choose a cloud provider like

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform (GCP)

Reasons for Selecting a Managed Cloud Database Service as follows,

- High Availability of Data and Resources

  Cloud services are ultimately known for the high availability of services, data, and resources. The services/ solutions, tools, and data are available anytime, anywhere you want, which is the biggest advantage one can have for the best outcome.

- Easy Implementation

  Cloud implementation generally deals with delivering software applications or deliverables/hardware to the end-user. The end-user need not know about the cloud supporting implementation and can reach the cloud service provider's support team when required. Cloud implementation comes in various forms or models, and you can select one essential for your organization.

- Resource Scalability

  The cloud's flexible infrastructure allows you to scale up and down as needed. Data centers force you to make informed guesses about your IT needs and acquire servers in bulk.

- Cost-Effective

  Allows you to pay for only what you use because most cloud providers operate on a "utility" pricing system. The monthly cost is determined by the amount of data used, the amount of storage required, the amount of bandwidth necessary, and the amount of computing power required.

- Data Security

  Cloud infrastructure and software services protect data against accidental loss, malicious activity, and natural events like fires, floods, and earthquakes. Cloud storage allows users the protection for controlling cybercrime and other security threats.

8.1 Things/changes that do in backend

- We want to buy a database server and creating data base into the online server.
- Change some data types.
- We have a lot of data in our database.
- Update Database Connection

9. Individual contribution

TG/2023/1699 - N.N. KUMUDUMALI

I managed the foundational setup and security of the entire database system, ensuring access control and system integrity. Key contributions included:

- User account creation and privileges - Creating all five mandatory MySQL user accounts to our database, I assigned the specific privileges for them.
- Security measures - Using usernames and passwords for all users protects the database from unauthorized access to special tables.
- Data population - Populating the user tables with initial and sufficient data for all roles in this system.

Followings are the tables, views and procedures that I created for this project.

Tables

Here are tables that I created.

i. USER

This table has details about the users in our database. I include a dean, 5 technical officers, 8 lecturers, an admin, 11 proper students and 8 repeat students as users.

ii. LECTURER

This table has lecturers' details who can access our database. All lecturers have their own ID, name and email address. All of them assign their department in the faculty.

iii. STUDENT

This table has students' details. Each student has a unique registration number to identify in the faculty.

iv. TECHNICAL OFFICER

I used this table to store the technical officer's details who was responsible for the students' attendance.

v. ADMIN

This table includes an admin's details.

vi. DEAN

I stored details of Dean who is responsible for the overall system in the university.

vii. MEDICAL ATTENDENCE

Views

These are the views that I created for our TEC_MIS ;

i. Medical_Eligible_Attendance

I used this view to show the details about the students' attendance eligibility with accepted medicals for the End-semester examination in the university.

ii. Medical_Non_Eligible_Attendance

Using this view I hope to show the details about students' attendance eligibility for the End-semester examination, with the medical submissions and but they are not eligible for sit the examination.

Procedures

Followings are the procedures I created.

i. CA_whole_batch_a_course

Using this one, you can find the CA marks of the whole batch for a specific course.

ii. CA_all_course_a_student

Using this one, can find CA marks of the whole batch for all courses in this semester.

Git profile : https://github.com/Nimeka1206

TG/2023/1734 – J.S.M.R. JAYASOORIYA

I focused on establishing the course details, enrollment details and some attendance related tasks:

- Database and some table creation - Initiating the database creation and structuring tables for some entities in the EER-Diagram.
- Enrollment - Managing the students' enrollment data.
- Eligibility and attendance constraints - Implementing the logic to determine full eligibility of students.

Followings are the tables, views and procedures that I created for this project.

Tables

Here are some tables that I created.

i. DEPARTMENT

This table has details about the departments in our university. I included three departments with their unique IDs and names.

ii. COURSE

This table has courses' details that can follow in our university. All courses have their own ID, name and number of credits. All of them are assigned to the department of ICT.

iii. COURSE_ENROLLMENT

This table has students' registration numbers, some of the basic course details.

iv. ATTENDANCE

I used this table to store the students' attendance for the courses with the week, attendance of the students this week and hours of the lectures.

v. ATTENDANCE_HANDLE

This table includes the details about who manages the attendance details of the courses in various weeks.

Views

These are the views that I created for our TEC_MIS;

i. Eligible_Attendance

I used this view to show the details about the students' attendance eligibility who do not submit the medicals for their absence.

ii. Non_Eligible_Attendance

Using this view I hope to show the details about students' attendance eligibility who are not eligible for the End-semester examination, and they do not submit the medicals for their absence.

Procedures

Followings are the procedures I created.

i. CA_a_course_a_student

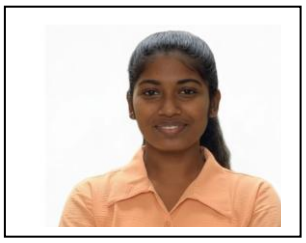
Using this one, you can find the CA marks for a specific student in a specific course.

ii. Final_mark_all_course_a_student

Using this one, you can find final marks of the whole batch for the all courses in this semester.

Git Profile : https://github.com/Rumesha2003

TG/2023/1747 – E.P.H.P. GUNASEKARA



I took ownership related to academic performance tracking, from raw marks to final results and grading. This involved.

- Marks data structure - Creating and populating tables to store all types of marks data, including quizzes, assignments, mid & end exams and project scores.
- Grading logic - Implementing the stored procedures necessary to calculate grade points and assign final grades according to the UGC Commission Circular No. 12-2024 guidelines.
- Attendance tables and logic - Designing the tables and stored logic for session-wise attendance tracking and subsequent percentage calculation.

Followings are the tables, views and procedures that I created for this project.

Tables

Here are some tables that I created.

i. QUIZ

This table has details about the quizzes that students faced in this semester. All courses have three quizzes for a semester and for calculating the CA marks only get the highest two quizzes.

ii. ASSIGNMENT

This table has assignments' details that are done by the students in the semester. All assignments have their own number.

iii. PROJECT

This table has students' projects marks.

iv. MID EXAM

I used this table to store the students' mid-semester examination results.

v. END EXAM

This table includes details about marks of the students for their end-semester examination.

Views

These are the views that I created for our TEC_MIS.

i. Final_result

I used this view to show the details about the students' attendance eligibility who do not submit the medicals for their absence.

ii. Attendance_eligibility

Using this view I hope to calculate summary of attendance records, clearly stating the attendance percentage and the final eligibility status for every student in every course component.

Procedures

Followings are the procedures I created.

i. Attendance_A_student_A_Course

Using this one, can retrieve the detailed attendance summary percentage and eligibility status for one specific student in one specific course component.

ii. Attendance_A_Student_All_Course

Using this one, you can find the attendance summary, percentage and eligibility status for one specific student across all courses they are registered for.

iii. Final_Marks_Whole_Batch_All_Course

I used this one to retrieve the final marks for every student in the entire batch for all courses.

Git Profile : https://github.com/Prabhasha2003

TG/2023/1772 – S.S.T.S. ROOPASINGHE

I was primarily responsible for implementing the entire attendance and medical record keeping system. This includes.

- Medical handling - Implementing the specific feature to record medical submissions and integrate this into the attendance calculation, covering categories for students with medicals.
- GPA calculation - Developing the required procedures for calculating and displaying SGPA and CGPA for individual students and the batch summary.

Followings are the tables, views and procedures that I created for this project.

Tables

Here are some tables that I created.

i. MEDICAL_MID

This table has details about the medical submissions for mid-semester examination. Each medical has unique ID.

ii. MEDICAL_END

This table has medical details that are submitted by the students for the end-semester examination.

iii. MEDICAL_ATTENDANCE

This table has medical submission details for the covering of daily or periodic absence.

iv. MEDICAL_HANDLE

I used this table for tracks who handled and approved/ rejected a medical submission.

v. GRADE_POINT

This table serves as a lookup table to map letter grades to their corresponding numeric grade point.

Views

These are the views that I created for our TEC_MIS.

i. Result_Status

I used this view to calculate the CA marks and determine the pass/ fail status for both the CA and the end exam, incorporating attendance and medical status overrides.

ii. GPA

Using this view, I hope to calculate the final SGPA and CGPA for every student based on the results generated by the result view.

Procedures

Followings are the procedures I created.

i. Attendance_Whole_Batch_A_Course

Using this one, can retrieve the final attendance percentage and eligibility status for all students enrolled in one specific course unit.

ii. Attendance_Whole_Batch_All_Course

Using this one, can find the final attendance percentage and eligibility status for every student in the entire batch for all courses.

iii. Medical_mid

To fetch a specific student's medical submission details from the medical_mid table using their registration number as input.

iv. Medical_end

This code used to fetch a student's medical submission history for their end-semester exams.

v. Medical_attendence

This code used to fetch a student's medical submission history for course attendance.

Git Profile : https://github.com/SewwandiSSTR

10 References

- ESDS Blogs and Accolades
https://www.esds.co.in/blog/top-12-reasons-to-pick-cloud-services/

- w3cschool.com
https://www.w3schools.com/mysql/default.asp

- Lecture Materials