ASSIGNMENT – 2

CSA0593 – DBMS

NAME : S.VARSHA

REG NO : 192311425

DATE : 15:11:2024

1. **Develop a database to manage students, courses, professors, and grades.**
    - **Mode cl tables for students, courses, professors, and grades.**
    - **Write stored procedures for enrolling in courses and assigning grades.**
    - **Implement triggers to update course availability and student records.**
    - **Write SQL queries to analyze student performance and course popularity.**

ANSWER:

- **CONCEPTUAL ER DIAGRAM**

This database manages students, courses, professors, and grades, with tables for each entity and relationships like enrollments and teaching. Stored procedures handle enrolling students in courses and assigning grades, while triggers update course availability and student records dynamically. SQL queries analyze student performance (e.g., average grades) and course popularity (e.g., enrollment counts).

CODE

```
-- Create the Students table
CREATE TABLE Students (
    StudentID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    DOB DATE NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Phone VARCHAR(15),
    Address TEXT
);

-- Create the Professors table
CREATE TABLE Professors (
    ProfessorID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Department VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Phone VARCHAR(15)
);

-- Create the Courses table
CREATE TABLE Courses (
    CourseID INT AUTO_INCREMENT PRIMARY KEY,
```

```
    CourseName VARCHAR(100) NOT NULL,
    Credits INT NOT NULL,
    MaxCapacity INT NOT NULL,
    AvailableSeats INT NOT NULL,
    ProfessorID INT,
    FOREIGN KEY (ProfessorID) REFERENCES
Professors(ProfessorID)
);

-- Create the Grades table
CREATE TABLE Grades (
    GradeID INT AUTO_INCREMENT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    Grade CHAR(2),
    GradeDate DATE,
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);

-- Create an Enrollment table to manage student-course relationships
CREATE TABLE Enrollments (
    EnrollmentID INT AUTO_INCREMENT PRIMARY KEY,
```

```
    StudentID INT,
    CourseID INT,
    EnrollmentDate DATE NOT NULL,
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
```

Column Name	Data Type	Description
student_id	INT	Primary Key, Unique Student Identifier
first_name	VARCHAR(100)	Student's First Name
last_name	VARCHAR(100)	Student's Last Name
email	VARCHAR(150)	Student's Email Address
date_of_birth	DATE	Student's Date of Birth
phone_number	VARCHAR(15)	Student's Phone Number

Column Name	Data Type	Description
course_id	INT	Primary Key, Unique Course Identifier
course_name	VARCHAR(200)	Name of the Course
course_description	TEXT	Description of the Course
professor_id	INT	Foreign Key to Professors table
max_students	INT	Maximum Number of Students for the Course
available_seats	INT	Number of Seats Available

Column Name	Data Type	Description
professor_id	INT	Primary Key, Unique Professor Identifier
first_name	VARCHAR(100)	Professor's First Name
last_name	VARCHAR(100)	Professor's Last Name
email	VARCHAR(150)	Professor's Email Address
department	VARCHAR(100)	Professor's Department

Column Name	Data Type	Description
grade_id	INT	Primary Key, Unique Grade Identifier
student_id	INT	Foreign Key to Students table
course_id	INT	Foreign Key to Courses table
grade	CHAR(2)	Grade Assigned (A, B, C, D, F, etc.)

CONCLUSION

This database efficiently manages students, courses, professors, and grades by using well-structured relational tables and logical relationships. Stored procedures simplify common tasks like enrolling students in courses and assigning grades. Triggers maintain data integrity by automatically updating course availability and student records. Comprehensive SQL queries provide insights into student performance and course popularity, enabling data-driven decisions. This robust system ensures scalability, data accuracy, and ease of use for academic institutions.