**`DATABASE MANAGEMENT SYSTEM**

**ASSIGNMENT-I**

1. **E-Learning Platform Course and Student Progress Management:**

**Tables for students, courses, lessons, progress, assessments, and enrollment, enforcing integrity through primary and foreign keys.**

Conceptual Diagram **:**

|  |
| --- |
| **Student** |
| **Student\_ID (PK)** |
| **Name** |
| **Email** |

|  |
| --- |
| **Enrollment** |
| **Enrollment\_ID (PK)** |
| **Student\_ID (FK)** |
| **Course\_ID (FK)** |
| **EnrollmentDate** |

|  |
| --- |
| **Course** |
| **Course\_ID (PK)** |
| **Course Name** |
| **Description** |

|  |
| --- |
| **Lesson** |
| **Lesson\_ID (PK)** |
| **Course\_ID (FK)** |
| **Lesson Title** |
| **Lesson Content** |

|  |
| --- |
| **Progress** |
| **Progress\_ID (PK)** |
| **Enrollment\_ID (FK)** |
| **Lesson\_ID (FK)** |
| **Completion Status** |
| **Completion Date** |

|  |
| --- |
| **Assessment** |
| **Assessment\_ID (PK)** |
| **Course\_ID (FK)** |
| **Assessment Title** |
| **Max Score** |

|  |
| --- |
| **Assessment Score** |
| **Score\_ID (PK)** |
| **Assessment\_ID (FK)** |
| **Enrollment\_ID (FK)** |
| **Score** |

Logical Model:

|  |
| --- |
| Enrollment |
| Enrollment\_ID (PK) |
| Student\_ID (FK) |
| Course\_ID (FK) |
| EnrollmentDate |

|  |
| --- |
| Assessment Score |
| Score\_ID (PK) |
| Assessment\_ID (FK) |
| Enrollment\_ID (FK) |
| Score |

|  |
| --- |
| Student |
| Student\_ID (PK) |
| Name |
| Email |

|  |
| --- |
| Course |
| Course\_ID (PK) |
| Course Name |
| Description |

|  |
| --- |
| Lesson |
| Lesson\_ID (PK) |
| Course\_ID (FK) |
| Lesson Title |
| Lesson Content |

|  |
| --- |
| Assessment |
| Assessment\_ID (PK) |
| Course\_ID (FK) |
| AssessmentTitle |
| Max Score |

**Stored Procedures**

**a. Enroll in Course**

This procedure enrolls a student in a specified course.

sql

CREATE PROCEDURE EnrollInCourse(IN studId INT, IN courseId INT)

BEGIN

INSERT INTO Enrollments (student\_id, course\_id, enrollment\_date)

VALUES (studId, courseId, CURDATE());

END;

**b. Update Lesson Completion**

This procedure updates a lesson's completion status for a student.

sql

CREATE PROCEDURE UpdateLessonCompletion(IN studId INT, IN lessonId INT)

BEGIN

UPDATE Progress

SET completion\_status = TRUE, completion\_date = CURDATE()

WHERE student\_id = studId AND lesson\_id = lessonId;

END;

**c. Calculate Course Score**

Calculates the total score for a student in a particular course.

sql

CREATE PROCEDURE CalculateCourseScore(IN studId INT, IN courseId INT)

BEGIN

SELECT SUM(marks\_obtained) AS total\_score

FROM Assessment\_Scores s

JOIN Assessments a ON s.assessment\_id = a.assessment\_id

WHERE s.student\_id = studId AND a.course\_id = courseId;

END;

Triggers:

**Trigger to Update Course Completion Status**

This trigger checks if all lessons in a course have been completed by a student. If true, it updates the course as completed.

CREATE TRIGGER UpdateCourseCompletion AFTER UPDATE ON Progress

FOR EACH ROW

BEGIN

DECLARE totalLessons INT;

DECLARE completedLessons INT;

-- Get the total number of lessons in the course

SELECT COUNT(\*)

INTO totalLessons

FROM Lesson

WHERE CourseID = (SELECT CourseID FROM Enrollment WHERE EnrollmentID = NEW.EnrollmentID);

-- Get the number of completed lessons in the enrollment

SELECT COUNT(\*)

INTO completedLessons

FROM Progress

WHERE EnrollmentID = NEW.EnrollmentID AND CompletionStatus = TRUE;

-- Check if all lessons are completed

IF totalLessons = completedLessons THEN

UPDATE Enrollment

SET CourseCompleted = TRUE

WHERE EnrollmentID = NEW.EnrollmentID;

END IF;

END;

**SQL Queries for Reports**

**a. Generate Report on Student Progress**

Shows the progress of students in each course.

sql

SELECT s.student\_id, s.first\_name, s.last\_name, c.course\_name,

COUNT(p.lesson\_id) AS completed\_lessons,

(SELECT COUNT(lesson\_id) FROM Lessons WHERE course\_id = c.course\_id) AS total\_lessons

FROM Progress p

JOIN Students s ON p.student\_id = s.student\_id

JOIN Lessons l ON p.lesson\_id = l.lesson\_id

JOIN Courses c ON l.course\_id = c.course\_id

WHERE p.completion\_status = TRUE

GROUP BY s.student\_id, c.course\_id;

**b. Generate Report on Popular Courses**

Lists courses with the highest number of enrollments.

sql

SELECT c.course\_name, COUNT(e.enrollment\_id) AS enrollment\_count

FROM Enrollments e

JOIN Courses c ON e.course\_id = c.course\_id

GROUP BY c.course\_id

ORDER BY enrollment\_count DESC;

**c. Generate Assessment Scores Report**

Shows average scores for each assessment.

sql

SELECT a.assessment\_name, AVG(s.marks\_obtained) AS average\_score

FROM Assessment\_Scores s

JOIN Assessments a ON s.assessment\_id = a.assessment\_id

GROUP BY a.assessment\_id;