|  |  |  |  |
| --- | --- | --- | --- |
|  | **Department of Software Engineering**  **Mehran University of Engineering and Technology, Jamshoro** | | |
|  | **Course: DATABASE MANAGEMENT SYSTEMS (BSSE123)** | | |
| **Instructor** | Ms. Sana Faiz | **Assignment Type** | Complex Engineering Problem |
| **Semester** | 2nd | **Year** 2025 | 1st |
| **Assignment Date** | - - 2025 | **Submission Deadline** | 20-04 -2025 |
| **Assessment Score** | 15 Marks |  |  |

**Class Assignment (Complex Engineering Problem - CEP)**

**Subject: DATABASE MANAGEMENT SYSTEMS Batch : 24BSAI Year: 1st Semester: 2nd**

# Instructions

** This assignment is based on Database Management Systems and must reflect real-world database design and management challenges.**

** The assignment will be submitted by each student individually.**

** Each question carries separate marks. Question 1 carries 10 marks, and Question 2 carries 5 marks (Total: 15 marks).**

** Assignment Assessment Rubrics are provided in Table 1 below.**

** Students must submit:**

* **A clearly labeled ER diagram (designed using any tool).**
* **Relational schemas with proper PKs and FKs.**
* **SQL queries written in proper syntax along with output.**
* **Explanations for normalization and transaction handling.**

** At the time of submission, students must be able to explain their database design, SQL logic, and reasoning behind normalization.**

**Assignment Assessment Rubrics (5 Marks) Roll# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Table.1 Rubrics Definitions**

| **Criteria** | **Proficient (5 Marks)** | **Acceptable (3 Marks)** | **Unacceptable (0 Marks)** |
| --- | --- | --- | --- |
| **ER Diagram & Schema Design** | Diagram is complete, well-structured, with correct relationships, cardinality, PKs & FKs | Minor issues in relationships or constraints; mostly accurate structure | Diagram missing key entities or relationships; incorrect structure |
| **SQL Query & Result Logic** | SQL query is syntactically correct, efficient, and meets all requirements | Query is functional but may be partially correct or inefficient | Query is missing, incorrect, or fails to meet basic objectives |
| |  | | --- | | **Transaction Management & Concurrency** | | |  | | --- | | **Transaction Management & Concurrency** | | |  | | --- | | **Transaction Management & Concurrency** | | |  | | --- | | **Transaction Management & Concurrency** | |

**ATTEMPT ALL QUESTIONS.**

**Question 1: [10 Marks]**

You are developing an **Academic Management System** for a university with the following requirements:

* Students enroll in multiple courses each semester.
* Each course may be taught by **multiple instructors over time** (e.g., co-teaching across semesters).
* Students receive **multiple assessments** per course (e.g., quizzes, assignments, midterm, final), each with a weight and score.
* Final grades should be converted to **GPA scale** per course and stored.
* Each instructor belongs to a **department**, but **courses can be co-hosted by multiple departments**.

**Tasks:**

a) Design a **comprehensive ER diagram** capturing entities, relationships, cardinalities, and constraints.  
b) Convert your diagram into **relational schemas**, highlighting all **PKs, FKs**, and constraints.  
c) Write a complex **SQL query** to generate a **student transcript** showing:

Course title, semester, instructor name(s), all assessment scores with types, GPA per course.  
 (Order by semester and GPA descending).

#### **Question 2: [5 Marks]**

A small online bookstore database allows multiple users to place and update orders at the same time.

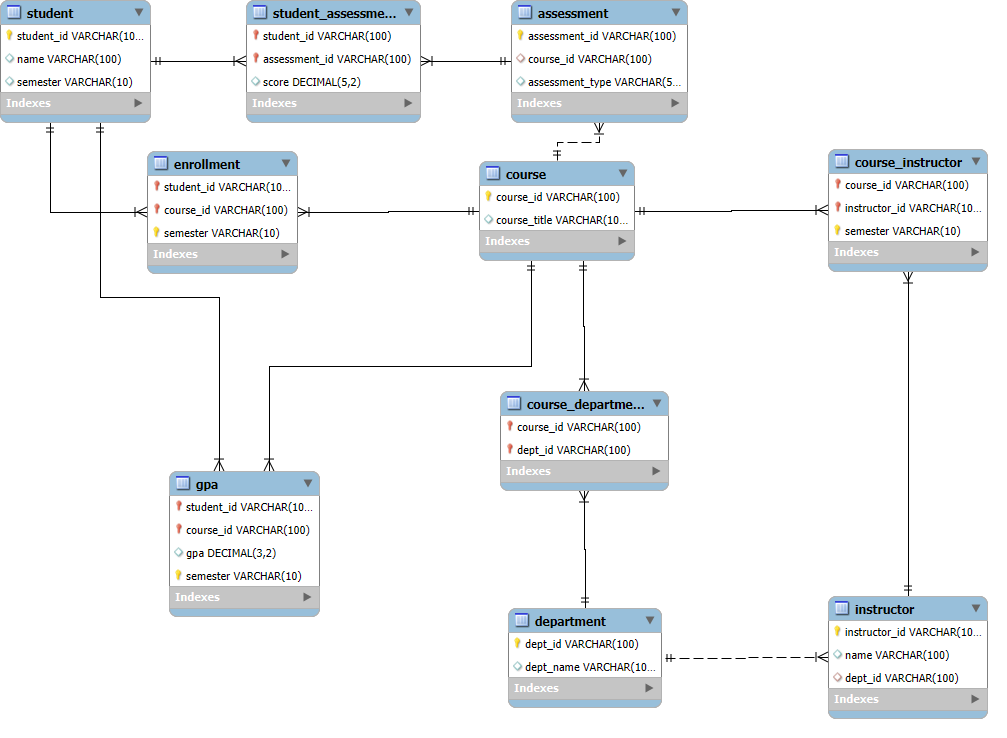
**Tasks:**

a) What are **transactions** in a database system? Write a simple example using SQL (e.g., placing an order).

b) Define any **two ACID properties** and explain why they are important in multi-user environments.

c) What can go wrong if two users try to update the same order at the same time? Suggest a **basic solution** to handle this issue.

**Question 1:-**

**(a). ER DIAGRAM:** 

**(b). Relational Schemas:-**

**//Creating DB**

**CREATE DATABASE AcademicManagementSystem;**

**USE AcademicManagementSystem;**

**// Department table**

**CREATE TABLE Department (**

**dept\_id VARCHAR(100)PRIMARY KEY,**

**dept\_name VARCHAR(100)**

**);**

**//Instructor table**

**CREATE TABLE Instructor (**

**instructor\_id VARCHAR(100) PRIMARY KEY,**

**name VARCHAR(100),**

**dept\_id VARCHAR(100),**

**FOREIGN KEY (dept\_id) REFERENCES Department(dept\_id)**

**);**

**//Course table**

**CREATE TABLE Course (**

**course\_id VARCHAR(100) PRIMARY KEY,**

**course\_title VARCHAR(100)**

**);**

**// Course\_Department table**

**CREATE TABLE Course\_Department (**

**course\_id VARCHAR(100),**

**dept\_id VARCHAR(100),**

**PRIMARY KEY (course\_id, dept\_id),**

**FOREIGN KEY (course\_id) REFERENCES Course(course\_id),**

**FOREIGN KEY (dept\_id) REFERENCES Department(dept\_id)**

**);**

**// Student table**

**CREATE TABLE Student (**

**student\_id VARCHAR(100) PRIMARY KEY,**

**name VARCHAR(100),**

**semester VARCHAR(10)**

**);**

**// Enrollment Table**

**CREATE TABLE Enrollment (**

**student\_id VARCHAR(100),**

**course\_id VARCHAR(100),**

**semester VARCHAR(10),**

**PRIMARY KEY (student\_id, course\_id, semester),**

**FOREIGN KEY (student\_id) REFERENCES Student(student\_id),**

**FOREIGN KEY (course\_id) REFERENCES Course(course\_id)**

**);**

**// Course\_Instructor Table**

**CREATE TABLE Course\_Instructor (**

**course\_id VARCHAR(100),**

**instructor\_id VARCHAR(100),**

**semester VARCHAR(10),**

**PRIMARY KEY (course\_id, instructor\_id, semester),**

**FOREIGN KEY (course\_id) REFERENCES Course(course\_id),**

**FOREIGN KEY (instructor\_id) REFERENCES Instructor(instructor\_id)**

**);**

**// Assessment Table**

**CREATE TABLE Assessment (**

**assessment\_id VARCHAR(100) PRIMARY KEY,**

**course\_id VARCHAR(100),**

**assessment\_type VARCHAR(50),**

**FOREIGN KEY (course\_id) REFERENCES Course(course\_id)**

**);**

**// Student\_Assessment Table**

**CREATE TABLE Student\_Assessment (**

**student\_id VARCHAR(100),**

**assessment\_id VARCHAR(100),**

**score DECIMAL(5,2),**

**PRIMARY KEY (student\_id, assessment\_id),**

**FOREIGN KEY (student\_id) REFERENCES Student(student\_id),**

**FOREIGN KEY (assessment\_id) REFERENCES Assessment(assessment\_id)**

**);**

**//GPA Table**

**CREATE TABLE GPA (**

**student\_id VARCHAR(100),**

**course\_id VARCHAR(100),**

**gpa DECIMAL(3,2),**

**semester VARCHAR(10),**

**PRIMARY KEY (student\_id, course\_id, semester),**

**FOREIGN KEY (student\_id) REFERENCES Student(student\_id),**

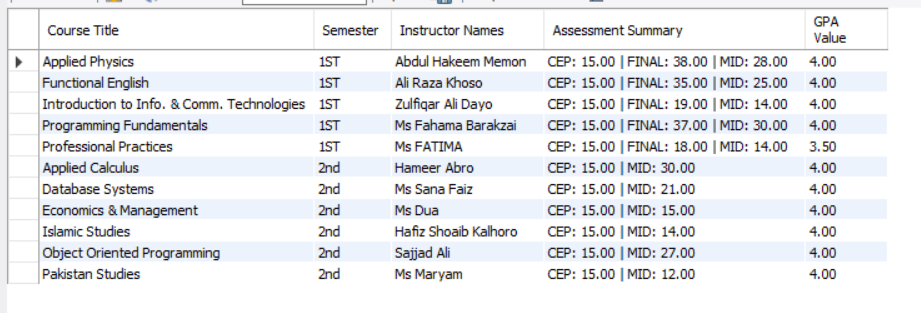
**FOREIGN KEY (course\_id) REFERENCES Course(course\_id)**

**);**

**(C).SQL Complex Query:-**



**OUTPUT:**



**Question 2:-**

**(a).**

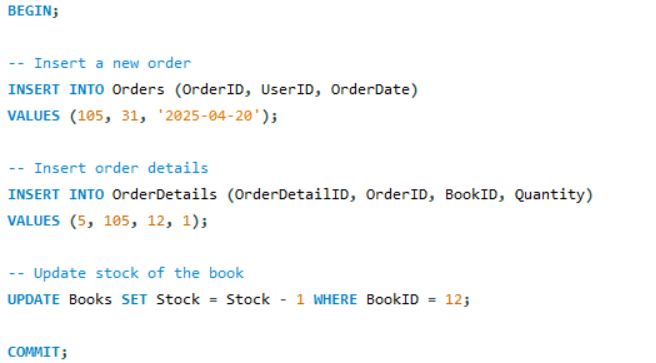
**Answer:**

A **transaction** is a group of one or more SQL operations executed as a single unit of work. The goal is to make sure **either all operations succeed or none do**, ensuring database consistency.

For example, placing an order involves inserting a new order, order details, and updating the stock of the book. All of these should succeed together.

If any step fails, the entire transaction can be **rolled back** to keep data safe.

**Example Query:**



**(b).**

**Answer:**

ACID properties ensure reliable transactions in databases. Two important properties are:

1. **Atomicity**
   * Ensures that all operations within a transaction are **completed fully or not at all**.
   * In multi-user environments, this prevents partial updates when errors occur.
2. **Isolation**
   * Ensures that **concurrent transactions** do not interfere with each other.
   * For example, two users placing or updating orders at the same time won’t affect each other’s transactions.

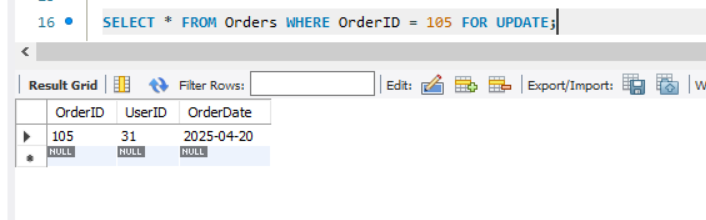
**(c).**

**Answer:**

If two users update the **same order at the same time**, problems like:

* **Lost updates**
* **Inconsistent data**
* **Deadlocks** may occur.

**Basic Solution:**



* This ensures only **one user can modify the row at a time**.

Or implement **transactions with isolation levels** like: 