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**	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:

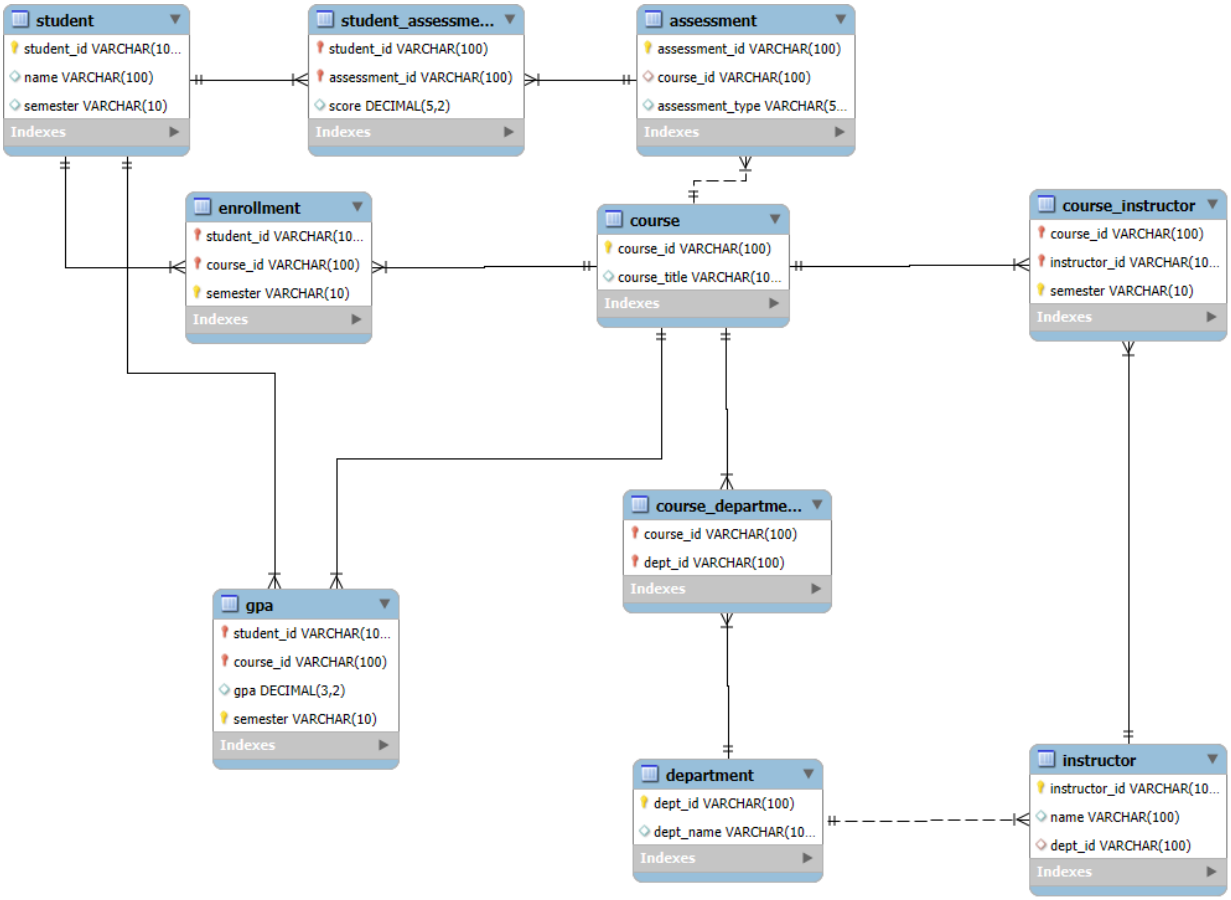
a) What are **transactions** in a database system? Write a simple example using SQL (e.g., placing an order).

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:-

```
SELECT
    c.course_title AS "Course Title",
    e.semester AS "Semester",
    GROUP_CONCAT(DISTINCT i.name SEPARATOR ", ") AS "Instructor Names",
    GROUP_CONCAT(CONCAT(a.assessment_type, ": ", sa.score) ORDER BY a.assessment_type SEPARATOR " | ") AS "Assessment Summary",
    g.gpa AS "GPA Value"
FROM
    enrollment e
JOIN student s
    ON e.student_id = s.student_id
JOIN course c
    ON e.course_id = c.course_id
LEFT JOIN course_instructor ci
    ON e.course_id = ci.course_id AND e.semester = ci.semester
LEFT JOIN instructor i
    ON ci.instructor_id = i.instructor_id
LEFT JOIN assessment a
    ON a.course_id = c.course_id
LEFT JOIN student_assessment sa
    ON sa.assessment_id = a.assessment_id AND sa.student_id = s.student_id
LEFT JOIN gpa g
    ON g.student_id = s.student_id AND g.course_id = c.course_id AND g.semester = e.semester
WHERE
    s.student_id = '24BSAI029'
GROUP BY
    c.course_title, e.semester, g.gpa
ORDER BY
   e.semester ASC, g.gpa DESC;
```

OUTPUT:

Course Title	Semester	Instructor Names	Assessment Summary	GPA Value
Applied Physics	1ST	Abdul Hakeem Memon	CEP: 15.00 | FINAL: 38.00 | MID: 28.00	4.00
Functional English	1ST	Ali Raza Khoso	CEP: 15.00 | FINAL: 35.00 | MID: 25.00	4.00
Introduction to Info. & Comm. Technologies	1ST	Zulfiqar Ali Dayo	CEP: 15.00 | FINAL: 19.00 | MID: 14.00	4.00
Programming Fundamentals	1ST	Ms Fahama Barakzai	CEP: 15.00 | FINAL: 37.00 | MID: 30.00	4.00
Professional Practices	1ST	Ms FATIMA	CEP: 15.00 | FINAL: 18.00 | MID: 14.00	3.50
Applied Calculus	2nd	Hameer Abro	CEP: 15.00 | MID: 30.00	4.00
Database Systems	2nd	Ms Sana Faiz	CEP: 15.00 | MID: 21.00	4.00
Economics & Management	2nd	Ms Dua	CEP: 15.00 | MID: 15.00	4.00
Islamic Studies	2nd	Hafiz Shoaib Kalhoro	CEP: 15.00 | MID: 14.00	4.00
Object Oriented Programming	2nd	Sajjad Ali	CEP: 15.00 | MID: 27.00	4.00
Pakistan Studies	2nd	Ms Maryam	CEP: 15.00 | MID: 12.00	4.00

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:

```sql
BEGIN;

-- Insert a new order
INSERT INTO Orders (OrderID, UserID, OrderDate)
VALUES (105, 31, '2025-04-20');

-- Insert order details
INSERT INTO OrderDetails (OrderDetailID, OrderID, BookID, Quantity)
VALUES (5, 105, 12, 1);

-- Update stock of the book
UPDATE Books SET Stock = Stock - 1 WHERE BookID = 12;

COMMIT;
```

(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.

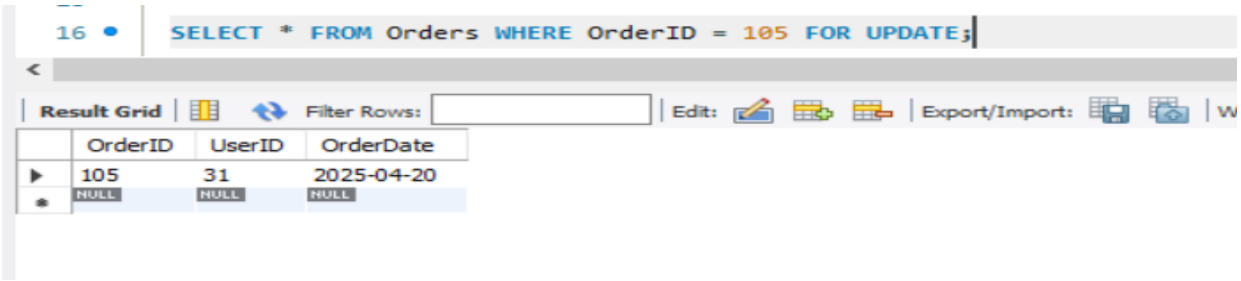
(c).

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:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```