Experiment 1.1

Student Name: Shubham Singh
UID: 23BCS13877
Branch: B.E CSE
Section: 23BCS_KRG-2_A
Semester: 5th
Date of Performance:24/07/25
Subject Name: ADBMS
Subject Code: 23CSH-333

1. **Aim:** To design and manipulate a University Database using SQL that involves creating relational tables for Students, Courses, Enrollments, and Professors, inserting and retrieving data using JOINs, managing access control with GRANT/REVOKE, and handling transaction control using COMMIT and ROLLBACK.

Easy-Level Problem

Problem Title: Author-Book Relationship Using Joins and Basic SQL Operations

Procedure (Step-by-Step):

1. Design two tables — one for storing author details and the other for book details.
2. Ensure a foreign key relationship from the book to its respective author.
3. Insert at least three records in each table.
4. Perform an INNER JOIN to link each book with its author using the common author ID.
5. Select the book title, author name, and author's country.

Medium-Level Problem

Problem Title: Department-Course Subquery and Access Control

Procedure (Step-by-Step):

1. Design normalized tables for departments and the courses they offer, maintaining a foreign key relationship.
2. Insert five departments and at least ten courses across those departments.
3. Use a subquery to count the number of courses under each department.
4. Filter and retrieve only those departments that offer more than two courses.
5. Grant SELECT-only access on the courses table to a specific user.

Hard-Level Problem

Problem Title: Transaction Management and Savepoint Simulation in Student Enrollments

Procedure (Step-by-Step):

1. Create three normalized tables — one each for students, courses, and enrollments.
2. Insert sample data for students and courses, then begin a transaction.
3. Add one enrollment successfully, then create a SAVEPOINT.
4. Attempt to insert a faulty or invalid enrollment to simulate an error.

5. Roll back only to the SAVEPOINT (not the entire transaction), then commit the valid data.
6. Finally, join all three tables to display the student's name, the course title they enrolled in, and the grade they received.

2. Tools Used: Oracle Live SQL

3. Code & Output:

Easy-Level Problem Code:

```
  CREATE TABLE Author (
    author_id INT PRIMARY KEY,
    author_name VARCHAR(50),
    country VARCHAR(30)
);

CREATE TABLE Book (
    book_id INT PRIMARY KEY,
    title VARCHAR(100),
    author_id INT,
    FOREIGN KEY (author_id) REFERENCES Author(author_id)
);

INSERT INTO Author (author_id, author_name, country) VALUES
(1, 'Chetan Bhagat', 'India'),
(2, 'Ruskin Bond', 'India'),
(3, 'Amish Tripathi', 'India');

INSERT INTO Book (book_id, title, author_id) VALUES
(101, 'Five Point Someone', 1),
(102, 'The Blue Umbrella', 2),
(103, 'The Immortals of Meluha', 3);

SELECT
    Book.title,
    Author.author_name,
    Author.country
FROM
    Book
INNER JOIN Author
    ON Book.author_id = Author.author_id;
```

	TITLE	AUTHOR_NAME	COUNTRY
1	Five Point Someone	Chetan Bhagat	India
2	The Blue Umbrella	Ruskin Bond	India
3	The Immortals of Me	Amish Tripathi	India

Figure 1: Output of Easy-Level Problem

Medium-Level Problem:

```
CREATE TABLE Department (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);

CREATE TABLE Course (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(100),
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES Department(dept_id)
);

INSERT INTO Department (dept_id, dept_name) VALUES
(1, 'Computer Science'),
(2, 'Electronics'),
(3, 'Mechanical'),
(4, 'Civil'),
(5, 'Electrical');

INSERT INTO Course (course_id, course_name, dept_id) VALUES
(101, 'DBMS', 1),
(102, 'Data Structures', 1),
(103, 'Operating Systems', 1),
(104, 'Digital Electronics', 2),
(105, 'Microprocessors', 2),
(106, 'Thermodynamics', 3),
(107, 'Fluid Mechanics', 3),
(108, 'Structural Analysis', 4),
(109, 'Electrical Machines', 5),
(110, 'Power Systems', 5);

SELECT
    dept_name,
    (SELECT COUNT(*)
     FROM Course
     WHERE Course.dept_id = Department.dept_id) AS total_courses
```

```
FROM Department
WHERE (SELECT COUNT(*)
      FROM Course
      WHERE Course.dept_id = Department.dept_id) > 2;

GRANT SELECT ON Course TO 'student_user';
```

DEPT_NAME	TOTAL_COURSES
No items to display.	

Figure 2: Output of Middle-Level Problem

Hard-Level Problem:

```
CREATE TABLE Students1 (
   student_id INT PRIMARY KEY,
   student_name VARCHAR(50)
);

CREATE TABLE Courses1 (
   course_id INT PRIMARY KEY,
   course_title VARCHAR(100)
);

CREATE TABLE Enrollments (
   enrollment_id INT PRIMARY KEY,
   student_id INT,
   course_id INT,
   grade VARCHAR(2),
   FOREIGN KEY (student_id) REFERENCES Students1(student_id),
   FOREIGN KEY (course_id) REFERENCES Courses1(course_id)
);

INSERT INTO Students1 (student_id, student_name) VALUES
(1, 'Amit Sharma'),
(2, 'Neha Singh'),
(3, 'Ravi Kumar');

INSERT INTO Courses1 (course_id, course_title) VALUES
(101, 'Database Management'),
(102, 'Computer Networks'),
(103, 'Operating Systems');
```

```
START TRANSACTION;

INSERT INTO Enrollments (enrollment_id, student_id, course_id, grade) VALUES
(1, 1, 101, 'A');

SAVEPOINT sp1;

INSERT INTO Enrollments (enrollment_id, student_id, course_id, grade) VALUES
(2, 5, 102, 'B');

ROLLBACK TO sp1;

COMMIT;

SELECT
   Students1.student_name,
   Courses1.course_title,
   Enrollments.grade
FROM Enrollments
INNER JOIN Students1 ON Enrollments.student_id = Students1.student_id
INNER JOIN Courses1 ON Enrollments.course_id = Courses1.course_id;
```

	STUDENT_NAME	COURSE_TITLE	GRADE
1	Amit Sharma	Database Manageme	A

Figure 3: Output of Hard-Level Problem

5. Learning Outcomes (What I have learnt):

- Understand how to design a relational schema for a real-world university system.
- Practice creating and linking tables using SQL.
- Use JOINs to query multi-table data meaningfully.
- Implement data access control using GRANT/REVOKE.
- Handle transactions safely using COMMIT and ROLLBACK.