WORKSHEET 1

Student Name: Anshum **UID: 23BCS11330**
Branch: BE-CSE **Section/Group: Krg_3A**
Semester: 5th **Subject Name: ADBMS**
Subject Code: 23CSP-333

1.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.

Sample Output Description:
When the join is performed, we get a list where each book title is shown along with its author's name and their country.

Code:

```
CREATE TABLE Authors (
   AuthorID INT PRIMARY KEY,
   AuthorName VARCHAR(max),
   Country VARCHAR(max)
);

CREATE TABLE Books (
   BookID INT PRIMARY KEY,
   Title VARCHAR(max),
   AuthorID INT FOREIGN KEY REFERENCES Authors(AuthorID)
);

INSERT INTO Authors (AuthorID, AuthorName, Country) VALUES
(101, 'A', 'IND'),
(102, 'C', 'SWZ'),
(103, 'N', 'UK');

INSERT INTO Books (BookID, Title, AuthorID) VALUES
(201, 'ABC', 101),
```

```
(202, 'XYZ', 102),
(203, 'AZ', 103);

SELECT
   B.Title AS BookTitle,
   A.AuthorName,
   A.Country
FROM
   Books AS B
INNER JOIN
   Authors AS A ON B.AuthorID = A.AuthorID;
```

Output:

BookTitle	AuthorName	Country
ABC	A	IND
XYZ	C	SWZ
AZ	N	UK

2. 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.

Sample Output Description:
The result shows the names of departments which are associated with more than two courses in the system.

Code:

```
CREATE TABLE Departments (
   DeptID INT PRIMARY KEY,
   DeptName VARCHAR(max)
);

CREATE TABLE Courses (
   CourseID INT PRIMARY KEY,
   CourseName VARCHAR(max),
   DeptID INT FOREIGN KEY REFERENCES Departments(DeptID)
);
INSERT INTO Departments (DeptID, DeptName) VALUES
(1, 'CS'),
(2, 'ME'),
(3, 'Physics'),
(4, 'Literature'),
(5, 'Maths');

INSERT INTO Courses (CourseID, CourseName, DeptID) VALUES
(1001, 'DS', 1),
(1002, 'OS', 1),
(1003, 'DBMS', 1),
(1004, 'Thermodynamics', 2),
(1005, 'Mechanics', 2),
(1006, 'Semiconductor Physics', 3),
(1007, 'Communication Skills', 4),
(1009, 'Calculus', 5),
(1010, 'EM', 5);

SELECT DeptName
 FROM Departments
WHERE DeptID IN (
   SELECT DeptID
   FROM Courses
   GROUP BY DeptID
   HAVING COUNT(*) > 2
);
GRANT SELECT ON Courses TO ABC;
```

Output:

```
Output:

DeptName
--------------------------------------------------------------------------------
CS
Msg 15151, Level 16, State 1, Server 1967960f9656, Line 38
Cannot find the user 'ABC', because it does not exist or you do not have permission.
```

3. 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.

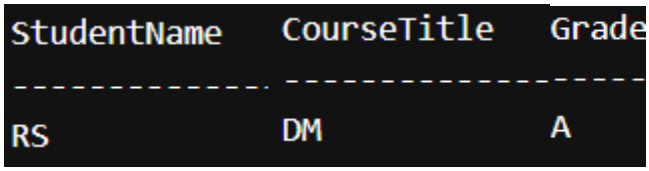
Sample Output Description:
After performing the join, we get a list of students with the courses they are enrolled in, along with their grades.

Code:

```
CREATE TABLE Students (
   StudentID INT PRIMARY KEY,
   StudentName VARCHAR(max)
);

CREATE TABLE CourseCatalog (
   CourseID INT PRIMARY KEY,
   CourseTitle VARCHAR(max)
);

CREATE TABLE Enrollments (
   EnrollmentID INT PRIMARY KEY,
   StudentID INT FOREIGN KEY REFERENCES Students(StudentID),
   CourseID INT FOREIGN KEY REFERENCES CourseCatalog(CourseID),
   Grade CHAR(2)
);
INSERT INTO Students VALUES
(1, 'RS'),
(2, 'MA'),
(3, 'LD');

INSERT INTO CourseCatalog VALUES
(10, 'DM'),
(11, 'CD'),
(12, 'WL');

BEGIN TRANSACTION;
INSERT INTO Enrollments VALUES (501, 1, 10, 'A');

SAVE TRANSACTION Enroll_Step1;
```

```
BEGIN TRY
   INSERT INTO Enrollments VALUES (502, 2, 999, 'B');
END TRY
BEGIN CATCH
   PRINT 'Error occurred. Rolling back to savepoint.';
   ROLLBACK TRANSACTION Enroll_Step1;
END CATCH;

COMMIT TRANSACTION;

SELECT
   S.StudentName,
   C.CourseTitle,
   E.Grade
FROM
   Enrollments E
JOIN Students S ON E.StudentID = S.StudentID
JOIN CourseCatalog C ON E.CourseID = C.CourseID;
```

Output:



StudentName	CourseTitle	Grade
RS	DM	A