



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 1.1

Student Name: Shivam Kumar

UID: 23BCS14207

Branch: CSE

Section/Group: KRG – 1B

Semester: 5th

Date of Performance: 22/07/25

Subject Name: ADBMS

Subject Code: 23CSP-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.
- 2. Requirements(Hardware/Software):** MySQL, PostgreSQL, Oracle, or SQL Server
- 3. DBMS script and output:**

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.

```
CREATE TABLE Authors (  
  Author_ID INT PRIMARY KEY,  
  Author_Name VARCHAR(100),  
  Country VARCHAR(100)  
);
```

```
CREATE TABLE Books (  
  Book_ID INT PRIMARY KEY,  
  Title VARCHAR(100),  
  Author_ID INT,  
  FOREIGN KEY (Author_ID) REFERENCES Authors(Author_ID)  
);
```

```
INSERT INTO Authors (Author_ID, Author_Name, Country) VALUES (1, 'NEIL GAIMAN', 'UNITED KINGDOM');  
INSERT INTO Authors (Author_ID, Author_Name, Country) VALUES (2, 'KAZUO ISHIGURO', 'JAPAN');  
INSERT INTO Authors (Author_ID, Author_Name, Country) VALUES (3, 'CHIMAMANDA ADICHIE', 'NIGERIA');
```

```
INSERT INTO Books (Book_ID, Title, Author_ID) VALUES(101, 'AMERICAN GODS', 1);  
INSERT INTO Books (Book_ID, Title, Author_ID) VALUES(102, 'NEVER LET ME GO', 2);  
INSERT INTO Books (Book_ID, Title, Author_ID) VALUES(103, 'PURPLE HIBISCUS', 3);
```

```
SELECT  
  B.Title AS Book_Title,  
  A.Author_Name,  
  A.Country  
FROM  
  Books B  
INNER JOIN  
  Authors A ON B.Author_ID = A.Author_ID;
```

☒ Autocommit Rows 10 Save Run

```
SELECT  
  B.Title AS Book_Title,  
  A.Author_Name,  
  A.Country  
FROM  
  Books B  
INNER JOIN  
  Authors A ON B.Author_ID = A.Author_ID;
```

Results Explain Describe Saved SQL History

BOOK_TITLE	AUTHOR_NAME	COUNTRY
AMERICAN GODS	NEIL GAIMAN	UNITED KINGDOM
NEVER LET ME GO	KAZUO ISHIGURO	JAPAN
PURPLE HIBISCUS	CHIMAMANDA ADICHIE	NIGERIA

3 rows returned in 0.01 seconds [Download](#)

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.

```
CREATE TABLE Departments (  
    Dept_ID INT PRIMARY KEY,  
    Dept_Name VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE Courses (  
    Course_ID INT PRIMARY KEY,  
    Course_Name VARCHAR(100) NOT NULL,  
    Dept_ID INT,  
    FOREIGN KEY (Dept_ID) REFERENCES Departments(Dept_ID)  
);
```

```
INSERT INTO Departments (Dept_ID, Dept_Name) VALUES (1, 'INFORMATION TECHNOLOGY');  
INSERT INTO Departments (Dept_ID, Dept_Name) VALUES (2, 'AEROSPACE ENGINEERING');  
INSERT INTO Departments (Dept_ID, Dept_Name) VALUES (3, 'ENVIRONMENTAL SCIENCE');  
INSERT INTO Departments (Dept_ID, Dept_Name) VALUES (4, 'STATISTICS');  
INSERT INTO Departments (Dept_ID, Dept_Name) VALUES (5, 'ASTROPHYSICS');
```

```
INSERT INTO Courses (Course_ID, Course_Name, Dept_ID) VALUES (101, 'CLOUD COMPUTING', 1);  
INSERT INTO Courses (Course_ID, Course_Name, Dept_ID) VALUES (102, 'MACHINE LEARNING', 1);  
INSERT INTO Courses (Course_ID, Course_Name, Dept_ID) VALUES (103, 'NETWORK SECURITY', 1);  
INSERT INTO Courses (Course_ID, Course_Name, Dept_ID) VALUES (104, 'AERODYNAMICS', 2);  
INSERT INTO Courses (Course_ID, Course_Name, Dept_ID) VALUES (105, 'ROCKET PROPULSION', 2);  
INSERT INTO Courses (Course_ID, Course_Name, Dept_ID) VALUES (106, 'ECOLOGY', 3);  
INSERT INTO Courses (Course_ID, Course_Name, Dept_ID) VALUES (107, 'CLIMATE MODELING', 3);  
INSERT INTO Courses (Course_ID, Course_Name, Dept_ID) VALUES (108, 'PROBABILITY THEORY', 4);  
INSERT INTO Courses (Course_ID, Course_Name, Dept_ID) VALUES (109, 'BLACK HOLE PHYSICS', 5);  
INSERT INTO Courses (Course_ID, Course_Name, Dept_ID) VALUES (110, 'COSMOLOGY', 5);  
INSERT INTO Courses (Course_ID, Course_Name, Dept_ID) VALUES (111, 'ARTIFICIAL INTELLIGENCE', 1);
```

```
CREATE USER Shivam_207 IDENTIFIED BY dirdam51;  
GRANT SELECT ON course TO Shivam_207;
```

```
SELECT Dept_Name  
FROM Departments  
WHERE Dept_ID IN (  
    SELECT Dept_ID  
    FROM Courses  
    GROUP BY Dept_ID  
    HAVING COUNT(*) > 2  
);
```

☒ Autocommit Rows Save Run

```

SELECT Dept_Name
FROM Departments
WHERE Dept_ID IN (
    SELECT Dept_ID
    FROM Courses
    GROUP BY Dept_ID
    HAVING COUNT(*) > 2
);

```

Results Explain Describe Saved SQL History

DEPT_NAME
INFORMATION TECHNOLOGY

1 rows returned in 0.01 seconds [Download](#)

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

```

CREATE TABLE Students (
    Student_ID INT PRIMARY KEY,
    Student_Name VARCHAR(100),
    Major VARCHAR(100)
);

```

```
CREATE TABLE Course (  
    Course_ID INT PRIMARY KEY,  
    Course_Title VARCHAR(100),  
    Credits INT  
);
```

```
CREATE TABLE Enrollments (  
    Enrollment_ID INT PRIMARY KEY,  
    Student_ID INT,  
    Course_ID INT,  
    Grade VARCHAR(2),  
    FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID),  
    FOREIGN KEY (Course_ID) REFERENCES Course(Course_ID)  
);
```

```
INSERT INTO Students (Student_ID, Student_Name, Major) VALUES (1, 'AARAV MEHRA', 'COMPUTER SCIENCE');  
INSERT INTO Students (Student_ID, Student_Name, Major) VALUES (2, 'PRIYA SINGH', 'ELECTRONICS');  
INSERT INTO Students (Student_ID, Student_Name, Major) VALUES (3, 'VIVAN KHANNA', 'MECHANICAL');
```

```
INSERT INTO Course (Course_ID, Course_Title, Credits) VALUES (101, 'DATA ANALYTICS', 4);  
INSERT INTO Course (Course_ID, Course_Title, Credits) VALUES (102, 'MACHINE LEARNING', 3);  
INSERT INTO Course (Course_ID, Course_Title, Credits) VALUES (103, 'THERMODYNAMICS', 4);
```

```
INSERT INTO Enrollments (Enrollment_ID, Student_ID, Course_ID, Grade) VALUES (201, 1, 101, 'A');
```

```
SAVEPOINT valid_enrollment;
```

```
BEGIN  
    INSERT INTO Enrollments (Enrollment_ID, Student_ID, Course_ID, Grade) VALUES (202, 999, 102, 'B');  
EXCEPTION  
    WHEN OTHERS THEN  
        ROLLBACK TO valid_enrollment;  
END;  
/
```

```
COMMIT;
```

```
SELECT S.Student_Name, C.Course_Title, E.Grade  
FROM Enrollments E  
INNER JOIN Students S ON E.Student_ID = S.Student_ID  
INNER JOIN Course C ON E.Course_ID = C.Course_ID;
```

Autocommit Rows 10 Save Run

```
SELECT S.Student_Name, C.Course_Title, E.Grade
FROM Enrollments E
INNER JOIN Students S ON E.Student_ID = S.Student_ID
INNER JOIN Course C ON E.Course_ID = C.Course_ID;
```

Results Explain Describe Saved SQL History

STUDENT_NAME	COURSE_TITLE	GRADE
AARAV MEHRA	DATA ANALYTICS	A

1 rows returned in 0.00 seconds [Download](#)

4. Learning Outcomes :

- Learn how to model a university database system using relational schema design principles.
- Gain hands-on experience in defining and connecting multiple SQL tables.
- Master querying across multiple tables using JOIN operations to retrieve insightful data.
- Apply access control mechanisms by managing user privileges with GRANT and REVOKE statements.
- Ensure data consistency and integrity by managing transactions with COMMIT and ROLLBACK commands.