Experiment 1

Student Name: Deepansh Kainth **UID:** 23BAI70184

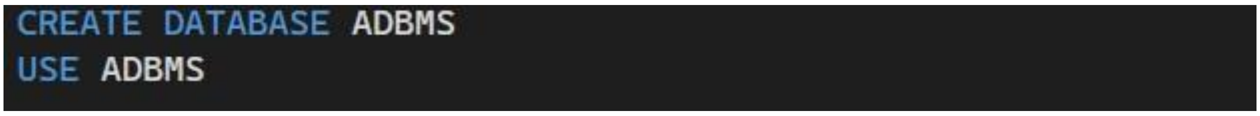
Branch: AIT-CSE **Section/Group:** 23AML_KRG-1

Semester: 5th **Date of Performance:** 23 July, 2025

Subject Name: ADBMS **Subject Code:** 23CSH-282

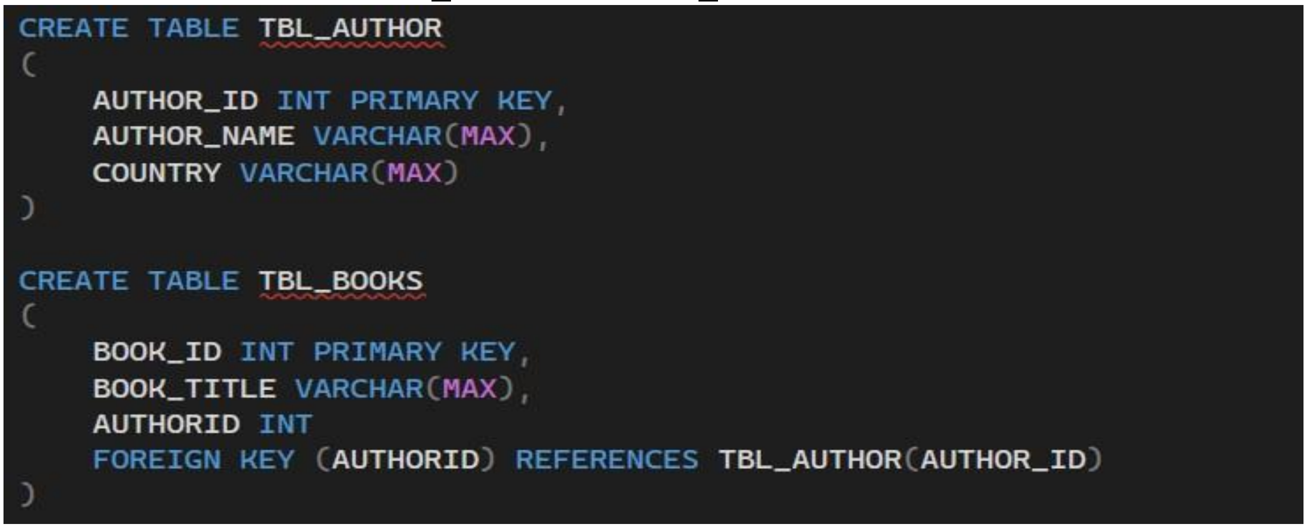
EASY - LEVEL

1. **Problem Title:** Author-Book Relationship Using Joins and Basic SQL Operations.
2. **Procedure (Step-by-Step):** Design two tables — one for storing author details and the other for book details.
   a. Ensure a foreign key relationship from the book to its respective author.
   b. Insert at least three records in each table.
   c. Perform an INNER JOIN to link each book with its author using the common author ID.
   d. Select the book title, author name, and author's country.
3. **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.
4. **SQL Commands:**
   a. Create the database and use it:



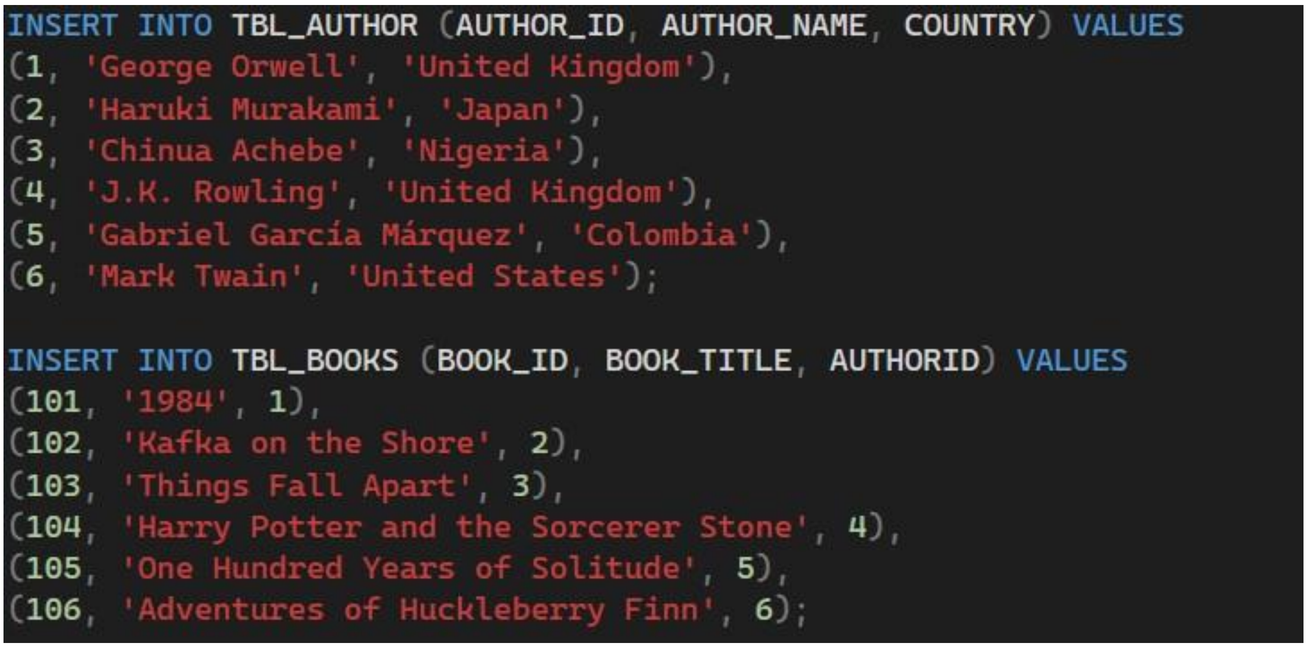
```
CREATE DATABASE ADBMS
USE ADBMS
```

   b. Create tables TBL_Author and TBL_Books:



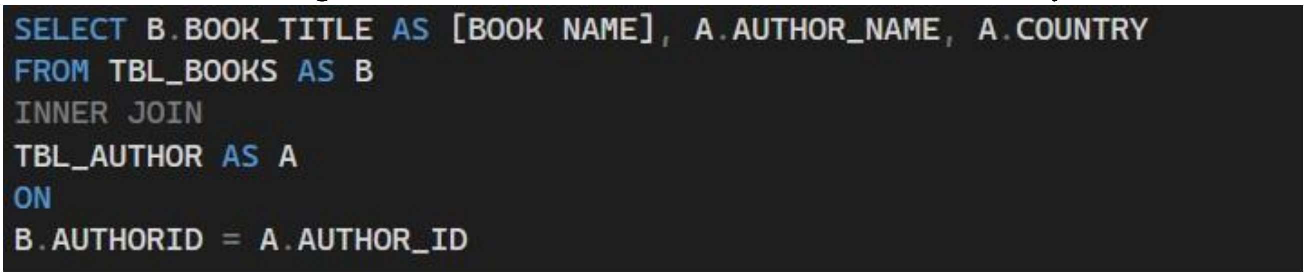
```
CREATE TABLE TBL_AUTHOR
(
    AUTHOR_ID INT PRIMARY KEY,
    AUTHOR_NAME VARCHAR(MAX),
    COUNTRY VARCHAR(MAX)
)

CREATE TABLE TBL_BOOKS
(
    BOOK_ID INT PRIMARY KEY,
    BOOK_TITLE VARCHAR(MAX),
    AUTHORID INT
    FOREIGN KEY (AUTHORID) REFERENCES TBL_AUTHOR(AUTHOR_ID)
)
```

c. Insert the values in the tables:



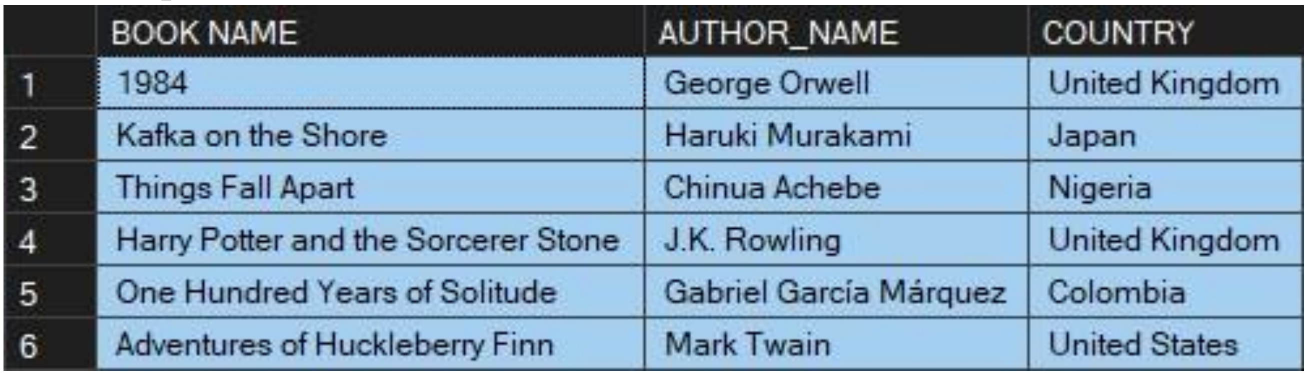
```sql
INSERT INTO TBL_AUTHOR (AUTHOR_ID, AUTHOR_NAME, COUNTRY) VALUES
(1, 'George Orwell', 'United Kingdom'),
(2, 'Haruki Murakami', 'Japan'),
(3, 'Chinua Achebe', 'Nigeria'),
(4, 'J.K. Rowling', 'United Kingdom'),
(5, 'Gabriel García Márquez', 'Colombia'),
(6, 'Mark Twain', 'United States');

INSERT INTO TBL_BOOKS (BOOK_ID, BOOK_TITLE, AUTHORID) VALUES
(101, '1984', 1),
(102, 'Kafka on the Shore', 2),
(103, 'Things Fall Apart', 3),
(104, 'Harry Potter and the Sorcerer Stone', 4),
(105, 'One Hundred Years of Solitude', 5),
(106, 'Adventures of Huckleberry Finn', 6);
```

d. Selecting the book title, author name, and author's country:



```sql
SELECT B.BOOK_TITLE AS [BOOK NAME], A.AUTHOR_NAME, A.COUNTRY
FROM TBL_BOOKS AS B
INNER JOIN
TBL_AUTHOR AS A
ON
B.AUTHORID = A.AUTHOR_ID
```

5. **Output:**



	BOOK NAME	AUTHOR_NAME	COUNTRY
1	1984	George Orwell	United Kingdom
2	Kafka on the Shore	Haruki Murakami	Japan
3	Things Fall Apart	Chinua Achebe	Nigeria
4	Harry Potter and the Sorcerer Stone	J.K. Rowling	United Kingdom
5	One Hundred Years of Solitude	Gabriel García Márquez	Colombia
6	Adventures of Huckleberry Finn	Mark Twain	United States

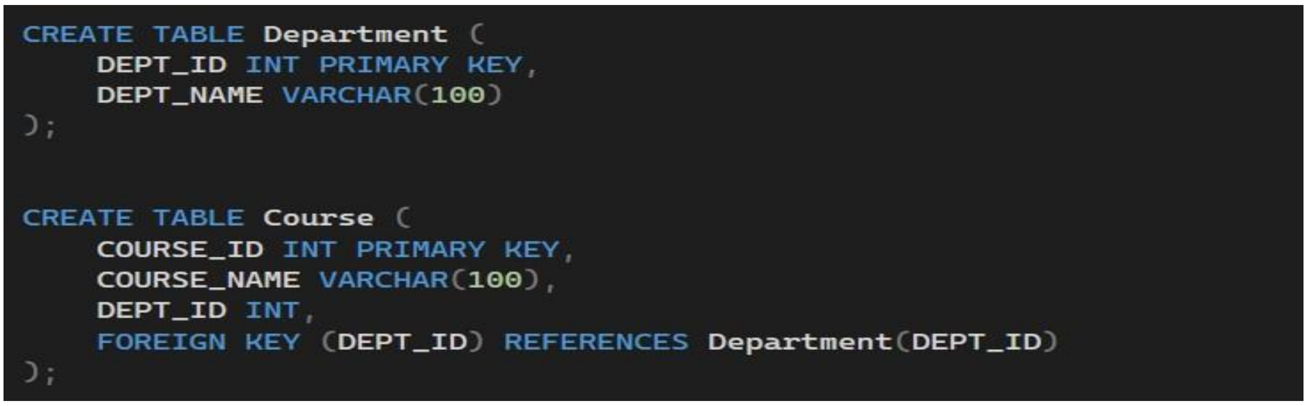
6. **Learning Outcome:**
   - I learnt how to create and manage relational databases using SQL.
   - I learnt how to define primary and foreign key constraints to link tables.
   - I learnt how to insert multiple records into SQL tables efficiently.
   - I learnt how to use INNER JOIN to retrieve combined data from related tables.

1. **Problem Title:** Course Subquery and Access Control 2.

Procedure (Step-by-Step):

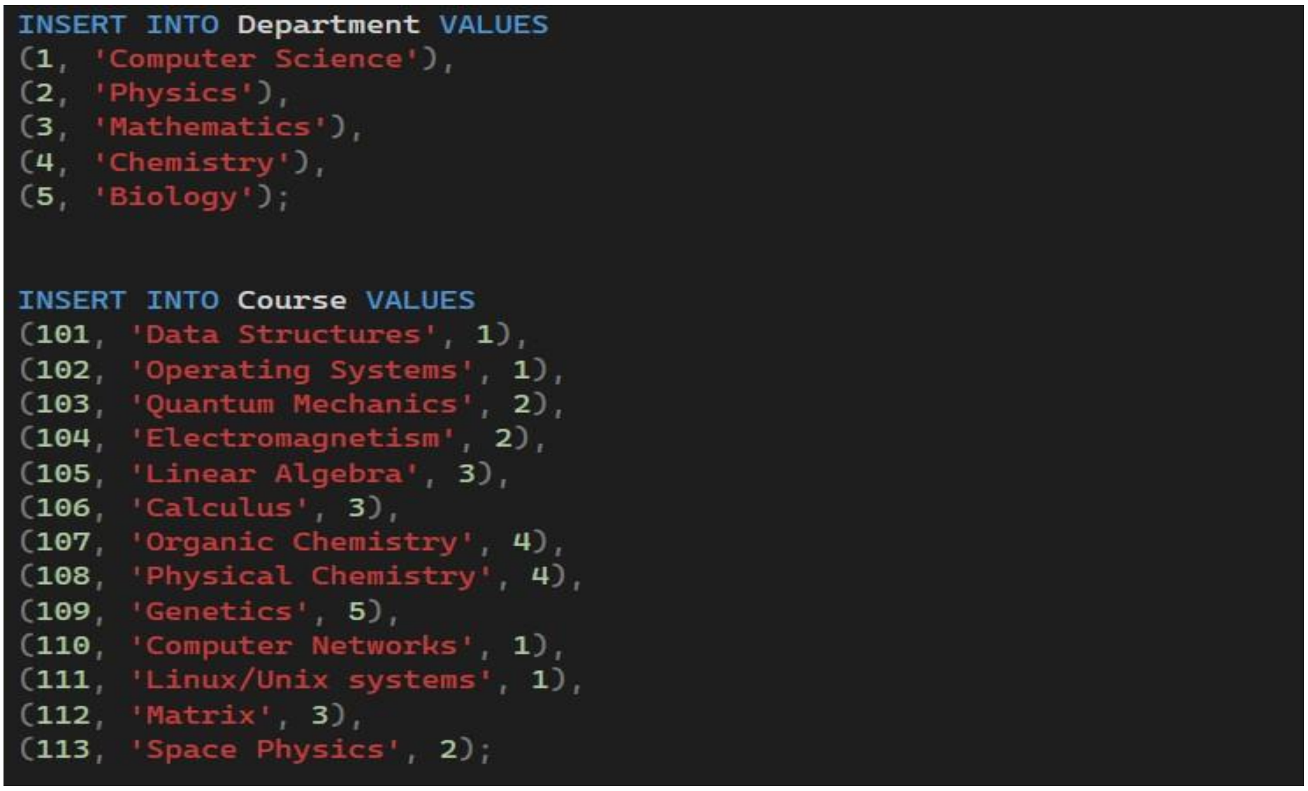
   a. Design normalized tables for departments and the courses they offer, maintaining a foreign key relationship.
   b. Insert five departments and at least ten courses across those departments.
   c. Use a subquery to count the number of courses under each department.
   d. Filter and retrieve only those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

3. **Sample Output Description:** The result shows the names of departments which are associated with more than two courses in the system.
4. **SQL Commands:**
   a. Create the tables.



```
CREATE TABLE Department (
    DEPT_ID INT PRIMARY KEY,
    DEPT_NAME VARCHAR(100)
);


CREATE TABLE Course (
    COURSE_ID INT PRIMARY KEY,
    COURSE_NAME VARCHAR(100),
    DEPT_ID INT,
    FOREIGN KEY (DEPT_ID) REFERENCES Department(DEPT_ID)
);
```

   b. Insert the values.



```
INSERT INTO Department VALUES
(1, 'Computer Science'),
(2, 'Physics'),
(3, 'Mathematics'),
(4, 'Chemistry'),
(5, 'Biology');


INSERT INTO Course VALUES
(101, 'Data Structures', 1),
(102, 'Operating Systems', 1),
(103, 'Quantum Mechanics', 2),
(104, 'Electromagnetism', 2),
(105, 'Linear Algebra', 3),
(106, 'Calculus', 3),
(107, 'Organic Chemistry', 4),
(108, 'Physical Chemistry', 4),
(109, 'Genetics', 5),
(110, 'Computer Networks', 1),
(111, 'Linux/Unix systems', 1),
(112, 'Matrix', 3),
(113, 'Space Physics', 2);
```

   c. Use a subquery to count the number of courses under each department.

```sql
SELECT
D.DEPT_NAME,
(SELECT COUNT(*)
 FROM Course AS C
 WHERE C.DEPT_ID = D.DEPT_ID) AS CourseCount
FROM Department AS D;
```

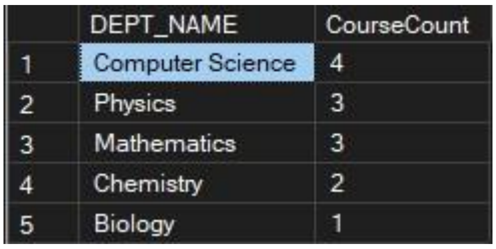
d. Filter and retrieve only those departments that offer more than two courses.

```sql
SELECT
DEPT_ID,
DEPT_NAME,
(SELECT COUNT(*)
 FROM Course
WHERE Course.DEPT_ID = Department.DEPT_ID) AS CourseCount
FROM Department
where (SELECT COUNT(*)
FROM Course
WHERE Course.DEPT_ID = Department.DEPT_ID) > 2;
```

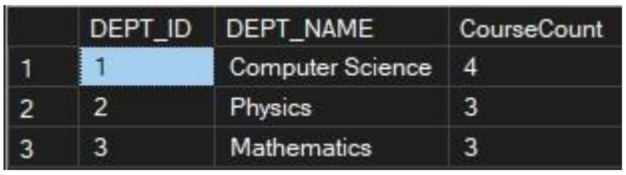
e. Grant SELECT-only access on the courses table to a specific user.

```sql
CREATE LOGIN TEST_LOGIN
WITH PASSWORD = 'TESTLOGIN@123';

CREATE USER TEST_USER
FOR LOGIN TEST_LOGIN

EXECUTE AS USER = 'TEST_USER'
GRANT SELECT, UPDATE ON COURSE TO TEST_USER
```

5. **Output:**



	DEPT_NAME	CourseCount
1	Computer Science	4
2	Physics	3
3	Mathematics	3
4	Chemistry	2
5	Biology	1



	DEPT_ID	DEPT_NAME	CourseCount
1	1	Computer Science	4
2	2	Physics	3
3	3	Mathematics	3

6. **Learning Outcomes:**

- Learned to design normalized database schemas using primary and foreign keys to maintain referential integrity between related entities.
- Developed proficiency in inserting and managing structured data across relational tables.
- Mastered the use of correlated subqueries to dynamically count related records for each row in a parent table.
- Applied scalar subqueries within SELECT and WHERE clauses to filter and compute aggregated results per row context.
- Gained practical experience in implementing user-level access control, using GRANT to assign SELECT-only privileges and EXECUTE AS with REVERT to switch and restore user contexts securely.