Experiment 1.1

Student Name: Madhav | **UID:** 23BAI70107

Branch: BE-AIT-CSE | **Section/Group:** 23AML_KRG-1(A)

Semester: 5th | **Date of Performance:** 8 January, 2025

Subject Name: ADBMS | **Subject Code:** 23CSP-333

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 AIT_1A;
use AIT_1A;
```

    b. Create tables TBL_Author and TBL_Books:

```
create table TBL_Author(
        Author_id Int primary key,
        Author_name varchar(max), -- max keyword 255 is the size
        countory varchar(max)
)
create table TBL_Books(
       Book_id Int primary key,
       Book_title varchar(max), -- max keyword 255 is the size
       AuthorId Int
       Foreign key (AuthorId) references TBL_Author(Author_id),)
```

c. Insert the values in the tables:

```
I SERT I TO TBL_Author (Author_id, Author_name, countory) VALUES
(1, 'George Orwell', 'United Kingdom'),
(2, '▫aruki Murakami', 'Japan'),
(3, 'Chinua Achebe', '▫igeria'),
(4, 'J.K. Rowling', 'United Kingdom'),
(5, 'Gabriel García Márquez', 'Colombia'),
(6, 'Mark Twain', 'United States');

I SERT I TO TBL_Books (Book_id, Book_title, AuthorId) VALUES
(101, '1984', 1),
(102, 'Kafka on the Shore', 2),
(103, 'Things Fall Apart', 3),
(104, '▫arry flotter and the Sorcerer Stone', 4),
(105, 'One  undred Years of Solitude', 5),
(106, 'Adventures of  uckleberry Finn', 6);
```

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

```
SELECT b.Book_title as 'Book  ame', a.Author_name as [Author ame],
a.countory [Country]
from TBL_Books as b
inner join TBL_Author as a
on b.AuthorId = a.Author_id;
```

5. **Output:**

	Name	Owner	Type	Created_datetime
1	TBL_Author	dbo	user table	2025-07-23 10:05:00.990

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	Author_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	Author_name	varchar	no	-1			yes	no	yes	SQL_Latin1_General_CP1_CI_AS
3	countory	varchar	no	-1			yes	no	yes	SQL_Latin1_General_CP1_CI_AS

Figure 1 TBL_Author Description

	Name	Owner	Type	Created_datetime
1	TBL_Books	dbo	user table	2025-07-23 10:05:07.500

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	Book_id	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	Book_title	varchar	no	-1			yes	no	yes	SQL_Latin1_General_CP1_CI_AS
3	AuthorId	int	no	4	10	0	yes	(n/a)	(n/a)	NULL

Figure 2 TBL_Books Description

	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

Figure 3 Select Command

6. **Learning Outcome:**

a. I learnt how to create and manage relational databases using SQL.
b. I learnt how to define primary and foreign key constraints to link tables.

c. I learnt how to insert multiple records into SQL tables efficiently.
d. I learnt how to use INNER JOIN to retrieve combined data from related tables.

MEDIUM - LEVEL

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 Department Table
CREATE TABLE Department (
    DeptID I T flRIMARY KEY,
    Dept ame VARC AR(100)
);

-- Create Course Table
CREATE TABLE Course (
    CourseID I T flRIMARY KEY,
    Course ame VARC AR(100),
    DeptID I T,
    FOREIG  KEY (DeptID) REFERE CES Department(DeptID)
);
```

   b. Insert the values.

```
I SERT I TO Department VALUES
(1, 'Computer Science'),
(2, 'flhysics'),
(3, 'Mathematics'),
(4, 'Chemistry'),
(5, 'Biology');
I SERT I TO 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, 'flhysical Chemistry', 4),
(109, 'Genetics', 5),
```

```
(110, 'ComputerNetworks', 1),
(111, 'Linux/Unix systems', 1),
(112, 'Matrix', 3),
(113, 'Space flhysics', 2);
```

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

```
SELECT
    D.Dept ame,
    (SELECT COU T(*)
     FROM Course C
     W ERE C.DeptID = D.DeptID) AS CourseCount
FROM Department D;
```

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

```
SELECT
    D.Dept ame
FROM Department D
W ERE (
    SELECT COU T(*)
    FROM Course C
    W ERE C.DeptID = D.DeptID
) >= 2;
```

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

```
create login test_login with password = 'Test@123';
create user test_user for login test_login;
execute as user = 'test_user';
grant select on Course to test_user;
```

5. **Output:**

	Name	Owner	Type	Created_datetime
1	Course	dbo	user table	2025-07-23 10:30:38.313

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	CourseID	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	CourseName	varchar	no	100			yes	no	yes	SQL_Latin1_General_CP1_CI_AS
3	DeptID	int	no	4	10	0	yes	(n/a)	(n/a)	NULL

Figure 4 Course table description

	Name	Owner	Type	Created_datetime
1	Department	dbo	user table	2025-07-23 10:30:38.310

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	DeptID	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	DeptName	varchar	no	100			yes	no	yes	SQL_Latin1_General_CP1_CI_AS

Figure 5 Department table description

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

Figure 6 Number of courses under each department

	DeptID	DeptName	CourseCount
1	1	Computer Science	4
2	2	Physics	3
3	3	Mathematics	3

Figure 7 Departments that offer more than two courses.

6. Learning Outcomes:

a. Learned to design normalized database schemas using primary and foreign keys to maintain referential integrity between related entities.
b. Developed proficiency in inserting and managing structured data across relational tables.
c. Mastered the use of **correlated subqueries** to dynamically count related records for each row in a parent table.
d. Applied **scalar subqueries** within SELECT and WHERE clauses to filter and compute aggregated results per row context.
e. 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.