Experiment 1.1

Student Name: Chandan kumar
UID: 23BAI70147

Branch: BE CSE (Hons.) (AIML)
Section/Group: 23AML_KRG-1

Semester: 5th
Date of Performance: 23-07-2025

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 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),
    COUNTRY VARCHAR(MAX)
);

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

c. Insert the values in the tables:

```
INSERT INTO TBL_AUTHOR (AUTHOR_ID, AUTHOR_NAME, COUNTRY) VALUES
(1, 'Ernest Hemingway', 'United States'),
(2, 'Franz Kafka', 'Czech Republic'),
(3, 'Khaled Hosseini', 'Afghanistan'),
(4, 'Margaret Atwood', 'Canada'),
(5, 'Orhan Pamuk', 'Turkey'),
(6, 'Albert Camus', 'France');

INSERT INTO TBL_BOOKS (BOOK_ID, BOOK_TITLE, AUTHOR_ID) VALUES
(1, 'The Old Man and the Sea', 1),
(2, 'The Metamorphosis', 2),
(3, 'The Kite Runner', 3),
(4, 'The Handmaid''s Tale', 4),
(5, 'Snow', 5),
(6, 'The Stranger', 6);
```

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

```
SELECT B.BOOK_TITLE AS [Book Title], A.AUTHOR_NAME AS [Author Name], A.COUNTRY AS [Country]
FROM TBL_BOOKS AS B
INNER JOIN
TBL_AUTHOR AS A
ON
B.AUTHOR_ID = A.AUTHOR_ID;
```

5. Output:

	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	COUNTRY	varchar	no	-1			yes	no	yes	SQL_Latin1_General_CP1_CI_AS

Figure 1 TBL_Author Description

	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	AUTHOR_ID	int	no	4	10	0	yes	(n/a)	(n/a)	NULL

Figure 2 TBL_Books Description

	Book Title	Author Name	Country
1	The Old Man and the Sea	Ernest Hemingway	United States
2	The Metamorphosis	Franz Kafka	Czech Republic
3	The Kite Runner	Khaled Hosseini	Afghanistan
4	The Handmaid's Tale	Margaret Atwood	Canada
5	Snow	Orhan Pamuk	Turkey
6	The Stranger	Albert Camus	France

Figure 3 Select Command

6. **Learning Outcome:**

a. Gained knowledge in creating and managing relational databases using SQL.
b. Learned to define primary and foreign key constraints to establish relationships between tables.
c. Acquired skills in efficiently inserting multiple records into SQL tables.
d. Developed the ability 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.

```sql
CREATE TABLE TBL_DEPARTMENT
(
    DEPARTMENT_ID INT PRIMARY KEY,
    DEPARTMENT_NAME VARCHAR(100) NOT NULL
);

CREATE TABLE TBL_COURSE
(
    COURSE_ID INT PRIMARY KEY,
    COURSE_NAME VARCHAR(100) NOT NULL,
    DEPARTMENT_ID INT,
    FOREIGN KEY (DEPARTMENT_ID) REFERENCES TBL_DEPARTMENT(DEPARTMENT_ID)
);
```

   b. Insert the values.

```sql
INSERT INTO TBL_DEPARTMENT (DEPARTMENT_ID, DEPARTMENT_NAME) VALUES
(1, 'Electrical Engineering'),
(2, 'Mechanical Engineering'),
(3, 'Civil Engineering'),
(4, 'Biology'),
(5, 'Economics');
```

```
INSERT INTO TBL_COURSE (COURSE_ID, COURSE_NAME, DEPARTMENT_ID) VALUES
(1, 'Circuit Analysis', 1),
(2, 'Digital Signal Processing', 1),
(3, 'Power Systems', 1),
(4, 'Thermodynamics', 2),
(5, 'Fluid Mechanics', 2),
(6, 'Structural Analysis', 3),
(7, 'Geotechnical Engineering', 3),
(8, 'Genetics', 4),
(9, 'Microbiology', 4),
(10, 'Macroeconomics', 5),
(11, 'International Trade', 5);
```

c. Use a subquery to count the number of courses under each department and filter and retrieve only those departments that offer more than two courses.

```
SELECT DEPARTMENT_NAME
FROM TBL_DEPARTMENT
WHERE DEPARTMENT_ID IN
(
    SELECT DEPARTMENT_ID
    FROM TBL_COURSE
    GROUP BY DEPARTMENT_ID
    HAVING COUNT(*) > 2
)
```

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

```
CREATE LOGIN TEST_LOGIN_DEVANSH
WITH PASSWORD = 'LOGIN@321';

CREATE USER TEST_LOGIN_DEVANSH
FOR LOGIN TEST_LOGIN_DEVANSH;

EXECUTE AS USER = 'TEST_USER_DEVANSH';

GRANT SELECT ON TBL_COURSE TO TEST_LOGIN_DEVANSH;
```

5. Output:

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

Figure 4 Department table description

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

Figure 5 Department table description

	DEPARTMENT_NAME	COURSECOUNT
1	Electrical Engineering	3
2	Mechanical Engineering	2
3	Civil Engineering	2
4	Biology	2
5	Economics	2

Figure 6 Number of courses under each department offering more than two courses

6. Learning Outcomes:

a. Gained expertise in designing normalized database schemas by defining primary and foreign keys to ensure referential integrity between related entities.
b. Acquired skills in inserting, updating, and managing structured data across relational tables.
c. Learned to use correlated subqueries for dynamically counting related records for each row in a parent table.
d. Applied scalar subqueries within **SELECT** and **WHERE** clauses to filter data and compute aggregated results in a row-specific context.
e. Developed hands-on experience in implementing user-level access control by using **GRANT** for assigning **SELECT**-only permissions and **EXECUTE** AS with **REVERT** to securely switch and restore user contexts.