

# **EXPERIMENT-01**

## **AIM:**

- (i) Create Author and Book Tables using DDL Commands
- (ii) Insert Sample Records into Author and Book Tables
- (iii) Retrieve Book Titles Along with Author Information Using INNER JOIN

## **OBJECTIVE:**

The objective of this experiment is to understand the core components of database schema design, particularly the creation and linking of tables using primary and foreign keys.

It also aims to strengthen the practical knowledge of DDL (Data Definition Language) and DML (Data Manipulation Language) operations, including table creation, data insertion, and joining tables to retrieve meaningful insights.

By performing this experiment on the ByteSQL platform, students will gain hands-on experience in relational database management and writing efficient SQL queries for real-world data modeling scenarios.

## **PROCEDURE:**

- Launch the ByteSQL platform to perform SQL operations in an interactive environment.
- Use CREATE TABLE statements to define the Authors table with the following fields:
  - i. author\_id (Primary Key)
  - ii. name (VARCHAR)

iii. country (VARCHAR)

- Define the Books table using CREATE TABLE with the fields:

i. book\_id (Primary Key)

ii. title (VARCHAR)

iii. author\_id (Foreign Key referencing Authors.author\_id)

- Insert sample data into the Authors table using INSERT INTO commands with at least three distinct authors.
- Insert sample data into the Books table using INSERT INTO commands while ensuring each book is linked to a valid author via the author\_id foreign key.
- Use an INNER JOIN SQL query to combine both tables and retrieve the book titles, author names, and author countries, matching records based on the common author\_id.
- Validate the results by ensuring that each book is correctly displayed with its corresponding author's information as per the join condition.

## **PROBLEM STATEMENT:**

**Problem Statement 1:** Design a basic Book Management System by creating two relational tables: Authors and Books. The system must represent a one-to-many relationship, where one author can write multiple books, but each book is associated with only one author. Use appropriate primary key and foreign key constraints to maintain referential integrity between the tables.

## Query 1:

```
CREATE TABLE Authors (author_id INT PRIMARY KEY, name VARCHAR(50), country VARCHAR(50));
```

```
CREATE TABLE Books (book_id INT PRIMARY KEY, title VARCHAR(100), author_id INT, FOREIGN KEY (author_id) REFERENCES Authors(author_id));
```

```
DESCRIBE Authors;
```

```
DESCRIBE Books;
```

## OUTPUT 1:

**Create Author and Book Tables using DDL**  
Score: 5 | Difficulty: easy

**Problem Statement**

You are tasked with designing a basic book management system. Create two tables — Authors and Books — to represent a one-to-many relationship (one author can write multiple books). Use proper primary and foreign key constraints while designing the schema.

**Input Format:**

Table Authors with columns:

- author\_id (INT, Primary Key)
- name (VARCHAR(50))
- country (VARCHAR(50))

Table Books with columns:

- book\_id (INT, Primary Key)
- title (VARCHAR(100))
- author\_id (INT, Foreign Key referencing Authors)

**Output Format:**

- Authors and Books tables created. Print description of the table.

**SQL**

```
1 CREATE TABLE Authors (author_id INT PRIMARY KEY, name VARCHAR(50), country VARCHAR(50));
2 CREATE TABLE Books (book_id INT PRIMARY KEY, title VARCHAR(100), author_id INT, FOREIGN KEY (author_id) REFERENCES Authors(author_id));
```

**Test & Results**

Custom Input

Test Cases

Run Code

**Output:**

Field	Type	Null	Key	Default	Extra
author_id	int	NO	PRI	NULL	
name	varchar(50)	YES		NULL	
country	varchar(50)	YES		NULL	

  

Field	Type	Null	Key	Default	Extra
book_id	int	NO	PRI	NULL	
title	varchar(100)	YES		NULL	
author_id	int	YES	MUL	NULL	

123 ms

## TEST CASE 1:

**Create Author and Book Tables using DDL**  
Score: 5 | Difficulty: easy

**Problem Statement**

You are tasked with designing a basic book management system. Create two tables — Authors and Books — to represent a one-to-many relationship (one author can write multiple books). Use proper primary and foreign key constraints while designing the schema.

**Input Format:**

Table Authors with columns:

- author\_id (INT, Primary Key)
- name (VARCHAR(50))
- country (VARCHAR(50))

Table Books with columns:

- book\_id (INT, Primary Key)
- title (VARCHAR(100))
- author\_id (INT, Foreign Key referencing Authors)

**Output Format:**

- Authors and Books tables created. Print description of the table.

**SQL**

```
1 CREATE TABLE Authors (author_id INT PRIMARY KEY, name VARCHAR(50), country VARCHAR(50));
2 CREATE TABLE Books (book_id INT PRIMARY KEY, title VARCHAR(100), author_id INT, FOREIGN KEY (author_id) REFERENCES Authors(author_id));
3 DESCRIBE Authors;
4 DESCRIBE Books;
5
```

**Test & Results**

Custom Input

Test Cases

Submit

Test Case	Status	Test Case Info
Test Case 1	Passed	

**Problem Statement 2:** After creating the Authors and Books tables, your next task is to insert sample records into both tables. You must add at least three authors and three books, ensuring that each book correctly references an existing author through the author\_id field.

## Query 2:

```
INSERT INTO Authors VALUES (1, 'Ashish', 'India'), (2, 'Smaran', 'USA'), (3, 'Vaibhav', 'UK');
```

```
INSERT INTO Books VALUES (101, 'Data Science Basics', 1), (102, 'AI in Education', 2), (103, 'SQL Simplified', 1);
```

```
SELECT * FROM Authors;
```

```
SELECT * FROM Books;
```

## OUTPUT 2:

The screenshot shows a SQL editor interface with a sidebar on the left containing a table of contents and a main area on the right. The sidebar lists 'Insert Sample Records into Author and Book Tables' with a score of 5 and difficulty of 'easy'. The main area displays the problem statement, input/output format, and the execution results of the SQL queries.

**Problem Statement**

After creating the Authors and Books tables, your next task is to insert sample records. Insert at least 3 authors and 3 books, ensuring books reference valid authors using the foreign key.

**Input Format:**

- Pre-existing Authors and Books table structures from Problem 1.

**Output Format:**

**Authors Table:**

author_id	name	country
1	Ashish	India
2	Smaran	USA
3	Vaibhav	UK

**Books Table:**

book_id	title	author_id
101	Data Science Basics	1

**SQL Editor**

```
1 INSERT INTO Authors VALUES (1, 'Ashish', 'India'), (2, 'Smaran', 'USA'), (3, 'Vaibhav', 'UK');
2 INSERT INTO Books VALUES (101, 'Data Science Basics', 1), (102, 'AI in Education', 2), (103, 'SQL Simplified', 1);
3 SELECT * FROM Authors;
4 SELECT * FROM Books;
```

**Test & Results**

Custom Input

Test Cases

Run Code

**Output:**

```
author_id | name | country |
+-----+-----+-----+
1 | Ashish | India |
2 | Smaran | USA |
3 | Vaibhav | UK |
+-----+-----+-----+
book_id | title | author_id |
+-----+-----+-----+
101 | Data Science Basics | 1 |
102 | AI in Education | 2 |
103 | SQL Simplified | 1 |
+-----+-----+-----+
```

143 ms

## TEST CASE 2:

The screenshot shows a SQL editor interface with a sidebar on the left containing a table of contents and a main area on the right. The sidebar lists 'Insert Sample Records into Author and Book Tables' with a score of 5 and difficulty of 'easy'. The main area displays the problem statement, input/output format, and the execution results of the SQL queries.

**Problem Statement**

After creating the Authors and Books tables, your next task is to insert sample records. Insert at least 3 authors and 3 books, ensuring books reference valid authors using the foreign key.

**Input Format:**

- Pre-existing Authors and Books table structures from Problem 1.

**Output Format:**

**Authors Table:**

author_id	name	country
1	Ashish	India
2	Smaran	USA
3	Vaibhav	UK

**Books Table:**

book_id	title	author_id
101	Data Science Basics	1

**SQL Editor**

```
1 INSERT INTO Authors VALUES (1, 'Ashish', 'India'), (2, 'Smaran', 'USA'), (3, 'Vaibhav', 'UK');
2 INSERT INTO Books VALUES (101, 'Data Science Basics', 1), (102, 'AI in Education', 2), (103, 'SQL Simplified', 1);
3 SELECT * FROM Authors;
4 SELECT * FROM Books;
```

**Test & Results**

Custom Input

Test Cases

Run Code

**Test Case Table:**

Test Case	Status	Test Case Info
Test Case 1	Passed	

**Problem Statement 3:** Given two tables, Authors and Books, retrieve the titles of all books along with their author's name and country. This involves creating tables, inserting data, and using an INNER JOIN to combine records based on author\_id.

### Query 3:

```
SELECT Books.title, Authors.name, Authors.country
```

```
FROM Books INNER JOIN Authors ON Books.author_id = Authors.author_id;
```

### OUTPUT 3:

### TEST CASE 3:

**33x Retrieve Book Titles Along with Author Information Using INNER JOIN**  
Score: 5 | Difficulty: easy

**1 Problem Statement**

Given two tables, Authors and Books, retrieve the titles of all books along with their author's name and country. This involves creating tables, inserting data, and using an INNER JOIN to combine records based on author\_id.

**2 Input Format:**

- Pre-existing Authors and Books table structures from Problem 1.

Table Authors with columns:

- author\_id (INT, Primary Key)
- name (VARCHAR(50))
- country (VARCHAR(50))

Table Books with columns:

- book\_id (INT, Primary Key)
- title (VARCHAR(100))
- author\_id (INT, Foreign Key referencing Authors)

**3 Output Format:**

- A list of books with their title, name of the author, and

```
1 SELECT Books.title, Authors.name, Authors.country
2 FROM Books INNER JOIN Authors ON Books.author_id = Authors.author_id;
3
```

**Test & Results** **Submit**

**Custom Input** **Run Code**

Custom Input

Test Cases

Output:

title	name	country
Data Science Basics	Ashish	India
AI in Education	Smaran	USA
SQL Simplified	Ashish	India

201 ms

**23x Retrieve Book Titles Along with Author Information Using INNER JOIN**  
Score: 5 | Difficulty: easy

**1 Problem Statement**

Given two tables, Authors and Books, retrieve the titles of all books along with their author's name and country. This involves creating tables, inserting data, and using an INNER JOIN to combine records based on author\_id.

**2 Input Format:**

- Pre-existing Authors and Books table structures from Problem 1.

Table Authors with columns:

- author\_id (INT, Primary Key)
- name (VARCHAR(50))
- country (VARCHAR(50))

Table Books with columns:

- book\_id (INT, Primary Key)
- title (VARCHAR(100))
- author\_id (INT, Foreign Key referencing Authors)

**3 Output Format:**

- A list of books with their title, name of the author, and

```
1 SELECT Books.title, Authors.name, Authors.country
2 FROM Books INNER JOIN Authors ON Books.author_id = Authors.author_id;
3
```

**Test & Results** **Submit**

**Custom Input** **Run Code**

Custom Input

Test Cases

Test Case	Status	Test Case Info
Test Case 1	Passed	

