

---

## 1. Need for a Database

A database is essential for managing and organizing large amounts of data efficiently. Key reasons include:

- **Data Organization:** Provides a structured way to store data using tables.
- **Data Integrity:** Ensures accuracy through constraints (e.g., primary keys, foreign keys).
- **Data Security:** Allows for access control and data protection.
- **Data Retrieval:** Enables efficient querying to fetch or manipulate data.
- **Scalability:** Handles large volumes of data and concurrent users effectively.

**Use and create commands:**

**use aaryadb;**

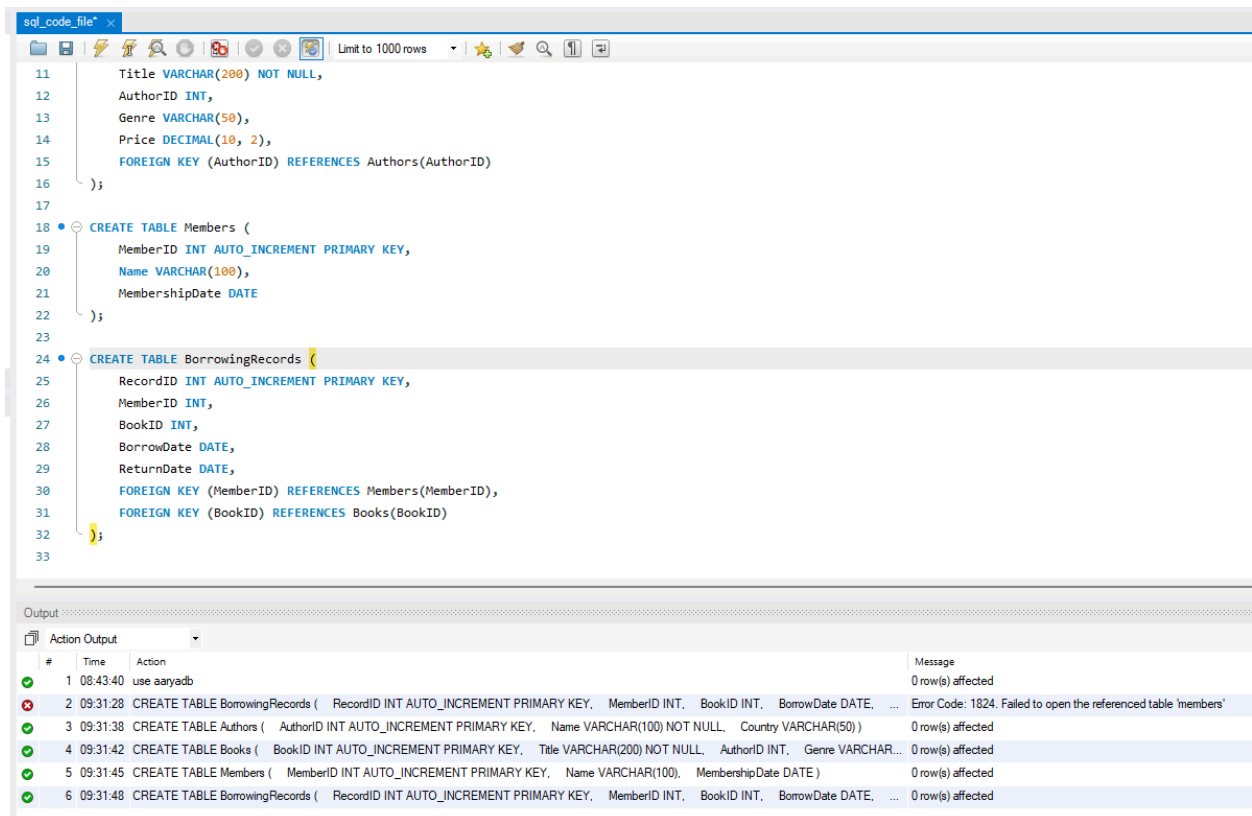
```
CREATE TABLE Authors (  
    AuthorID INT AUTO_INCREMENT PRIMARY KEY,  
    Name VARCHAR(100) NOT NULL,  
    Country VARCHAR(50)  
);
```

```
CREATE TABLE Books (  
    BookID INT AUTO_INCREMENT PRIMARY KEY,  
    Title VARCHAR(200) NOT NULL,  
    AuthorID INT,  
    Genre VARCHAR(50),  
    Price DECIMAL(10, 2),  
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)  
);
```

```
CREATE TABLE Members (  
    MemberID INT AUTO_INCREMENT PRIMARY KEY,  
    Name VARCHAR(100),  
    MembershipDate DATE  
);
```

```
CREATE TABLE BorrowingRecords (  
    RecordID INT AUTO_INCREMENT PRIMARY KEY,  
    MemberID INT,  
    BookID INT,  
    BorrowDate DATE,  
    ReturnDate DATE,  
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),  
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
```

);



The screenshot shows a SQL IDE window titled 'sql\_code\_file'. The editor contains SQL commands for creating three tables: `BorrowingRecords`, `Members`, and `Books`. The `BorrowingRecords` table has a primary key `RecordID` and foreign keys to `Members` and `Books`. The `Members` table has a primary key `MemberID`. The `Books` table has a primary key `BookID` and a foreign key to `Authors`. The output window at the bottom shows the execution of these commands. Command 2 (creating `BorrowingRecords`) failed with error 1824 because the referenced table 'members' did not exist at that time. Command 3 (creating `Authors`) succeeded. Command 4 (creating `Books`) succeeded. Command 5 (creating `Members`) succeeded. Command 6 (re-creating `BorrowingRecords`) succeeded.

```
11 Title VARCHAR(200) NOT NULL,
12 AuthorID INT,
13 Genre VARCHAR(50),
14 Price DECIMAL(10, 2),
15 FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
16 );
17
18 CREATE TABLE Members (
19 MemberID INT AUTO_INCREMENT PRIMARY KEY,
20 Name VARCHAR(100),
21 MembershipDate DATE
22 );
23
24 CREATE TABLE BorrowingRecords (
25 RecordID INT AUTO_INCREMENT PRIMARY KEY,
26 MemberID INT,
27 BookID INT,
28 BorrowDate DATE,
29 ReturnDate DATE,
30 FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
31 FOREIGN KEY (BookID) REFERENCES Books(BookID)
32 );
33
```

Output

#	Time	Action	Message
1	08:43:40	use aaryadb	0 row(s) affected
2	09:31:28	CREATE TABLE BorrowingRecords ( RecordID INT AUTO_INCREMENT PRIMARY KEY, MemberID INT, BookID INT, BorrowDate DATE, ...	Error Code: 1824. Failed to open the referenced table 'members'
3	09:31:38	CREATE TABLE Authors ( AuthorID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(100) NOT NULL, Country VARCHAR(50) )	0 row(s) affected
4	09:31:42	CREATE TABLE Books ( BookID INT AUTO_INCREMENT PRIMARY KEY, Title VARCHAR(200) NOT NULL, AuthorID INT, Genre VARCHAR...	0 row(s) affected
5	09:31:45	CREATE TABLE Members ( MemberID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(100), MembershipDate DATE )	0 row(s) affected
6	09:31:48	CREATE TABLE BorrowingRecords ( RecordID INT AUTO_INCREMENT PRIMARY KEY, MemberID INT, BookID INT, BorrowDate DATE, ...	0 row(s) affected

### Insert commands:

```
INSERT INTO Authors (Name, Country)
VALUES
('J.K. Rowling', 'UK'),
('George R.R. Martin', 'USA'),
('J.R.R. Tolkien', 'UK');
```

```
INSERT INTO Books (Title, AuthorID, Genre, Price)
VALUES
('Harry Potter', 1, 'Fantasy', 20.99),
('Game of Thrones', 2, 'Fantasy', 25.99),
('The Hobbit', 3, 'Adventure', 15.99);
```

```
INSERT INTO Members (Name, MembershipDate)
VALUES
('Alice', '2023-01-01'),
('Bob', '2023-02-15');
```

```
INSERT INTO BorrowingRecords (MemberID, BookID, BorrowDate, ReturnDate)
VALUES
(1, 1, '2023-03-01', '2023-03-10'),
(2, 2, '2023-03-05', NULL);
```

```

34 • INSERT INTO Authors (Name, Country)
35 VALUES
36 ('J.K. Rowling', 'UK'),
37 ('George R.R. Martin', 'USA'),
38 ('J.R.R. Tolkien', 'UK');
39
40 • INSERT INTO Books (Title, AuthorID, Genre, Price)
41 VALUES
42 ('Harry Potter', 1, 'Fantasy', 20.99),
43 ('Game of Thrones', 2, 'Fantasy', 25.99),
44 ('The Hobbit', 3, 'Adventure', 15.99);
45
46 • INSERT INTO Members (Name, MembershipDate)
47 VALUES
48 ('Alice', '2023-01-01'),
49 ('Bob', '2023-02-15');
50
51 • INSERT INTO BorrowingRecords (MemberID, BookID, BorrowDate, ReturnDate)
52 VALUES
53 (1, 1, '2023-03-01', '2023-03-10'),
54 (2, 2, '2023-03-05', NULL);
55
56

```

Output			
Action Output			
#	Time	Action	Message
✓ 1	08:43:40	use aayadb	0 row(s) affected
✗ 2	09:31:28	CREATE TABLE BorrowingRecords ( RecordID INT AUTO_INCREMENT PRIMARY KEY, MemberID INT, BookID INT, BorrowDate DATE, ...	Error Code: 1824. Failed to open the referenced table 'members'
✓ 3	09:31:38	CREATE TABLE Authors ( AuthorID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(100) NOT NULL, Country VARCHAR(50))	0 row(s) affected
✓ 4	09:31:42	CREATE TABLE Books ( BookID INT AUTO_INCREMENT PRIMARY KEY, Title VARCHAR(200) NOT NULL, AuthorID INT, Genre VARCHAR...	0 row(s) affected
✓ 5	09:31:45	CREATE TABLE Members ( MemberID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(100), MembershipDate DATE)	0 row(s) affected
✓ 6	09:31:48	CREATE TABLE BorrowingRecords ( RecordID INT AUTO_INCREMENT PRIMARY KEY, MemberID INT, BookID INT, BorrowDate DATE, ...	0 row(s) affected
✓ 7	09:32:49	INSERT INTO Authors (Name, Country) VALUES ('J.K. Rowling', 'UK'), ('George R.R. Martin', 'USA'), ('J.R.R. Tolkien', 'UK')	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
✓ 8	09:32:52	INSERT INTO Books (Title, AuthorID, Genre, Price) VALUES ('Harry Potter', 1, 'Fantasy', 20.99), ('Game of Thrones', 2, 'Fantasy', 25.99), ('The Hobbit', 3, ...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
✓ 9	09:32:56	INSERT INTO Members (Name, MembershipDate) VALUES ('Alice', '2023-01-01'), ('Bob', '2023-02-15')	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0
✓ 10	09:33:00	INSERT INTO BorrowingRecords (MemberID, BookID, BorrowDate, ReturnDate) VALUES (1, 1, '2023-03-01', '2023-03-10'), (2, 2, '2023-03-05', NULL)	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0

## Select command:

SELECT \* FROM Books;

```

56 -- select command
57 • SELECT * FROM Books;
58

```

Result Grid				
BookID	Title	AuthorID	Genre	Price
1	Harry Potter	1	Fantasy	20.99
2	Game of Thrones	2	Fantasy	25.99
3	The Hobbit	3	Adventure	15.99
*	NULL	NULL	NULL	NULL

Books 1 ×			
Output			
Action Output			
#	Time	Action	Message
✓ 1	09:34:32	SELECT * FROM Books LIMIT 0, 1000	3 row(s) returned

## Distinct command:

```
--  
59      -- distinct  
60 •    SELECT DISTINCT Genre FROM Books;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Genre
Fantasy
Adventure

## Where command:

```
62      -- where  
63 •    SELECT * FROM Books WHERE Price > 20;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

BookID	Title	AuthorID	Genre	Price
1	Harry Potter	1	Fantasy	20.99
2	Game of Thrones	2	Fantasy	25.99
*	NULL	NULL	NULL	NULL

## and , or commands:

SELECT \* FROM Books WHERE Genre = 'Fantasy' AND Price < 25;

SELECT \* FROM Books WHERE Genre = 'Fantasy' OR Price < 25;

```
65      -- AND, OR  
66 •    SELECT * FROM Books WHERE Genre = 'Fantasy' AND Price < 25;  
67 •    SELECT * FROM Books WHERE Genre = 'Fantasy' OR Price < 25;  
68  
69  
70
```

Result Grid | Filter Rows: | Edit: | Export/Import: |

BookID	Title	AuthorID	Genre	Price
1	Harry Potter	1	Fantasy	20.99
*	NULL	NULL	NULL	NULL

```
66 •    SELECT * FROM Books WHERE Genre = 'Fantasy' AND Price < 25;  
67 •    SELECT * FROM Books WHERE Genre = 'Fantasy' OR Price < 25;  
68
```

Result Grid | Filter Rows: | Edit: | Export/Import: |

BookID	Title	AuthorID	Genre	Price
1	Harry Potter	1	Fantasy	20.99
2	Game of Thrones	2	Fantasy	25.99
3	The Hobbit	3	Adventure	15.99
*	NULL	NULL	NULL	NULL

### Order by:

```
69      -- order by
70 •    SELECT * FROM Books ORDER BY Price DESC;
```

Result Grid

Filter Rows:

Edit:

	BookID	Title	AuthorID	Genre	Price
▶	2	Game of Thrones	2	Fantasy	25.99
	1	Harry Potter	1	Fantasy	20.99
	3	The Hobbit	3	Adventure	15.99
✱	NULL	NULL	NULL	NULL	NULL

### Like:

```
72      -- like
73 •    SELECT * FROM Books WHERE Title LIKE '%Harry%';
```

Result Grid

Filter Rows:

Edit:

	BookID	Title	AuthorID	Genre	Price
▶	1	Harry Potter	1	Fantasy	20.99
✱	NULL	NULL	NULL	NULL	NULL

### Inner join:

```
75      -- inner join
76 •    SELECT Members.Name AS MemberName, Books.Title AS BookTitle, BorrowingRecords.BorrowDate
77      FROM BorrowingRecords
78      INNER JOIN Members ON BorrowingRecords.MemberID = Members.MemberID
79      INNER JOIN Books ON BorrowingRecords.BookID = Books.BookID;
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	MemberName	BookTitle	BorrowDate
▶	Alice	Harry Potter	2023-03-01
	Bob	Game of Thrones	2023-03-05

## Left join:

```
81 -- left join
82 • SELECT Members.Name, BorrowingRecords.RecordID
83 FROM Members
84 LEFT JOIN BorrowingRecords ON Members.MemberID = BorrowingRecords.MemberID;
85
```

Result Grid

Filter Rows:

Export:



Wrap Cell Content:

	Name	RecordID
▶	Alice	1
	Bob	2


## Right join:


```
86 -- right join
87 • SELECT Books.Title, BorrowingRecords.RecordID
88 FROM Books
89 RIGHT JOIN BorrowingRecords ON Books.BookID = BorrowingRecords.BookID;
90
91
```

Result Grid

Filter Rows:

Export: 

Wrap Cell Content: 

	Title	RecordID
▶	Harry Potter	1
	Game of Thrones	2

## Group by:

```
91 -- group by
92 • SELECT Genre, COUNT(*) AS BookCount FROM Books GROUP BY Genre;
93
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	Genre	BookCount
▶	Fantasy	2
	Adventure	1

## Having:

SELECT Genre, AVG(Price) AS AvgPrice FROM Books GROUP BY Genre HAVING AVG(Price) > 20;

```
94 -- having
95 • SELECT Genre, AVG(Price) AS AvgPrice FROM Books GROUP BY Genre HAVING AVG(Price) > 20;
96
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	Genre	AvgPrice
▶	Fantasy	23.490000

### Union:

```
SELECT Name FROM Members
UNION
SELECT Name FROM Authors;
```

```

97      -- union
98      SELECT Name FROM Members
99      UNION
100     SELECT Name FROM Authors;
101
```

---

Result Grid | Filter Rows:

	Name
▶	Alice
	Bob
	J.K. Rowling
	George R.R. Martin
	J.R.R. Tolkien

### Alter:

```

103      ALTER TABLE Members ADD Email VARCHAR(100);
104      select * from members;
```

---

Result Grid | Filter Rows:  Edit:

	MemberID	Name	MembershipDate	Email
▶	1	Alice	2023-01-01	NULL
	2	Bob	2023-02-15	NULL
✱	NULL	NULL	NULL	NULL

### Stored Procedure:

```
DELIMITER //
CREATE PROCEDURE GetBooksByAuthor(IN author_name VARCHAR(100))
BEGIN
    SELECT Books.Title, Books.Genre
    FROM Books
    INNER JOIN Authors ON Books.AuthorID = Authors.AuthorID
    WHERE Authors.Name = author_name;
END //
DELIMITER ;
CALL GetBooksByAuthor('J.K. Rowling');
```

```

107 DELIMITER //
108 • CREATE PROCEDURE GetBooksByAuthor(IN author_name VARCHAR(100))
109 BEGIN
110     SELECT Books.Title, Books.Genre
111     FROM Books
112     INNER JOIN Authors ON Books.AuthorID = Authors.AuthorID
113     WHERE Authors.Name = author_name;
114 END //
115 DELIMITER ;
116 • CALL GetBooksByAuthor('J.K. Rowling');
117

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	Title	Genre
▶	Harry Potter	Fantasy

### Prepared Statement:

```

PREPARE stmt FROM 'SELECT * FROM Members WHERE Name = ?';
SET @name = 'Alice';
EXECUTE stmt USING @name;
DEALLOCATE PREPARE stmt;

```

```

107 DELIMITER //
108 • CREATE PROCEDURE GetBooksByAuthor(IN author_name VARCHAR(100))
109 BEGIN
110     SELECT Books.Title, Books.Genre
111     FROM Books
112     INNER JOIN Authors ON Books.AuthorID = Authors.AuthorID
113     WHERE Authors.Name = author_name;
114 END //
115 DELIMITER ;
116 • CALL GetBooksByAuthor('J.K. Rowling');
117

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	Title	Genre
▶	Harry Potter	Fantasy



### Callable Statement:

-- Callable statement to add a new member

DELIMITER //

CREATE PROCEDURE AddMember(IN member\_name VARCHAR(100), IN join\_date DATE)

BEGIN

INSERT INTO Members (Name, MembershipDate) VALUES (member\_name, join\_date);

END //

DELIMITER ;

CALL AddMember('Charlie', '2023-05-01');

select \* from Members;

```
125 -- Callable statement to add a new member
126 DELIMITER //
127 • CREATE PROCEDURE AddMember(IN member_name VARCHAR(100), IN join_date DATE)
128 • BEGIN
129 •     INSERT INTO Members (Name, MembershipDate) VALUES (member_name, join_date);
130 • END //
131 DELIMITER ;
132
133 • CALL AddMember('Charlie', '2023-05-01');
134 • select * from Members;
```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
MemberID	Name	MembershipDate	Email	
1	Alice	2023-01-01	NULL	
2	Bob	2023-02-15	NULL	
3	Charlie	2023-05-01	NULL	
*	NULL	NULL	NULL	