---

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:

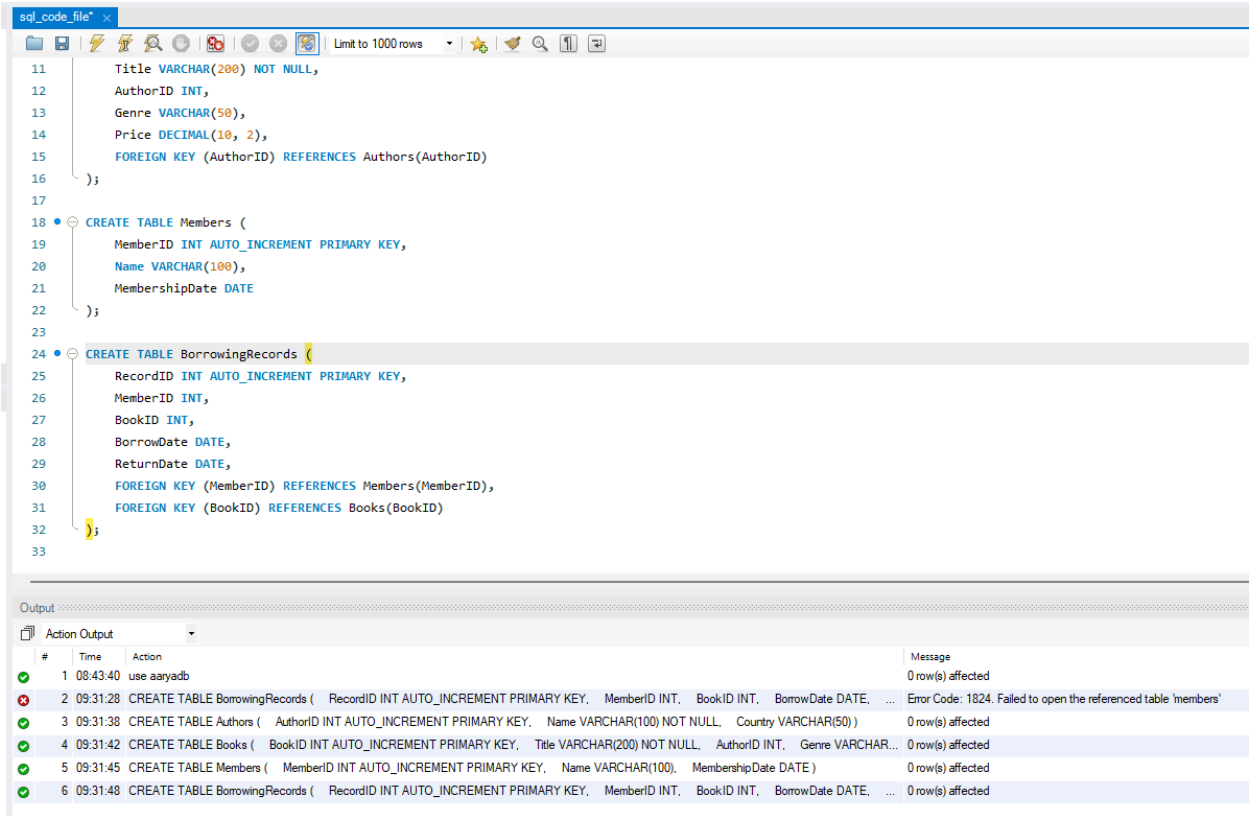
```
use aaryadb;
CREATE TABLE Authors (
   AuthorID INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(100) NOT NULL,
   Country VARCHAR(50)
);

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)
```

);



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);

```sql
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);
```

Output

Action 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
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;

```sql
-- select command
SELECT * FROM Books;
```

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
3	The Hobbit	3	Adventure	15.99
NULL	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:

```
-- distinct
SELECT DISTINCT Genre FROM Books;
```

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

Genre
Fantasy
Adventure

Where command:

```
-- where
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	NULL

and , or commands:

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

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

```
-- AND, OR
SELECT * FROM Books WHERE Genre = 'Fantasy' AND Price < 25;
SELECT * FROM Books WHERE Genre = 'Fantasy' OR Price < 25;
```

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

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

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

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	NULL

Order by:

```
-- order by
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:

```
-- like
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:

```
-- inner join
SELECT Members.Name AS MemberName, Books.Title AS BookTitle, BorrowingRecords.BorrowDate
FROM BorrowingRecords
INNER JOIN Members ON BorrowingRecords.MemberID = Members.MemberID
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:

```sql
-- left join
SELECT Members.Name, BorrowingRecords.RecordID
FROM Members
LEFT JOIN BorrowingRecords ON Members.MemberID = BorrowingRecords.MemberID;
```

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

Name	RecordID
Alice	1
Bob	2

Right join:

```sql
-- right join
SELECT Books.Title, BorrowingRecords.RecordID
FROM Books
RIGHT JOIN BorrowingRecords ON Books.BookID = BorrowingRecords.BookID;
```

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

Title	RecordID
Harry Potter	1
Game of Thrones	2

Group by:

```sql
-- group by
SELECT Genre, COUNT(*) AS BookCount FROM Books GROUP BY Genre;
```

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;

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

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

Genre	AvgPrice
Fantasy	23.490000

Union:

SELECT Name FROM Members
UNION
SELECT Name FROM Authors;

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

Result Grid | Filter Rows:

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

Alter:

```
ALTER TABLE Members ADD Email VARCHAR(100);
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');
```

```
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');
```

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

	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;

```
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');
```

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

	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;
```