SQL

Complete SQL Guide for Beginners (A to Z)

I'll teach you SQL step-by-step with a small project: **a Library Management System**.

Step 1: Understanding Databases

A database stores data in tables (like Excel spreadsheets). Each table has:

- **Columns**: Categories of data (like Name, Age)
- **Rows**: Individual records (like one person's information)

Step 2: Creating a Database

```
CREATE DATABASE LibraryDB;
USE LibraryDB;
```

Step 3: Creating Tables (CREATE TABLE)

```
CREATE TABLE Books (
    BookID INT PRIMARY KEY AUTO_INCREMENT,
    Title VARCHAR(100),
    Author VARCHAR(50),
    YearPublished INT,
    Genre VARCHAR(30),
    Available BOOLEAN DEFAULT TRUE
);

CREATE TABLE Members (
    MemberID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(50),
    Email VARCHAR(50),
    JoinDate DATE
);

CREATE TABLE Borrowings (
    BorrowID INT PRIMARY KEY AUTO_INCREMENT,
    BookID INT,
    MemberID INT,
    BorrowDate DATE,
    ReturnDate DATE,
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID)
);
```

Step 4: Inserting Data (INSERT)

```
-- Adding books
INSERT INTO Books (Title, Author, YearPublished, Genre)
VALUES
    ('1984', 'George Orwell', 1949, 'Dystopian'),
    ('To Kill a Mockingbird', 'Harper Lee', 1960, 'Fiction'),
    ('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 'Fiction'),
    ('Pride and Prejudice', 'Jane Austen', 1813, 'Romance'),
    ('The Hobbit', 'J.R.R. Tolkien', 1937, 'Fantasy');

-- Adding members
INSERT INTO Members (Name, Email, JoinDate)
VALUES
    ('Alice Johnson', 'alice@email.com', '2024-01-15'),
    ('Bob Smith', 'bob@email.com', '2024-02-20'),
    ('Carol White', 'carol@email.com', '2024-03-10');

-- Adding borrowing records
INSERT INTO Borrowings (BookID, MemberID, BorrowDate, ReturnDate)
VALUES
    (1, 1, '2024-06-01', '2024-06-15'),
    (2, 2, '2024-06-05', NULL),
    (3, 1, '2024-06-10', '2024-06-20');
```

Step 5: Basic Queries (SELECT)

View all data

```
SELECT * FROM Books;
SELECT * FROM Members;
```

Select specific columns

```
SELECT Title, Author FROM Books;
```

With conditions (WHERE)

```
SELECT * FROM Books WHERE Genre = 'Fiction';
SELECT * FROM Books WHERE YearPublished > 1950;
```

Step 6: Filtering Data

AND, OR, NOT operators

```sql
SELECT * FROM Books WHERE Genre = 'Fiction' AND YearPublished > 1950;
SELECT * FROM Books WHERE Genre = 'Fiction' OR Genre = 'Fantasy';
SELECT * FROM Books WHERE NOT Genre = 'Fiction';
```

IN operator

```sql
SELECT * FROM Books WHERE Genre IN ('Fiction', 'Fantasy', 'Romance');
```

BETWEEN

```sql
SELECT * FROM Books WHERE YearPublished BETWEEN 1900 AND 1950;
```

LIKE (pattern matching)

```sql
SELECT * FROM Books WHERE Title LIKE '%The%';
SELECT * FROM Books WHERE Author LIKE 'J%';
```

IS NULL / IS NOT NULL

```sql
SELECT * FROM Borrowings WHERE ReturnDate IS NULL;  -- Books not returned yet
SELECT * FROM Borrowings WHERE ReturnDate IS NOT NULL;  -- Returned books
```

Step 7: Sorting Results (ORDER BY)

```sql
SELECT * FROM Books ORDER BY YearPublished;  -- Ascending (default)
SELECT * FROM Books ORDER BY YearPublished DESC;  -- Descending
SELECT * FROM Books ORDER BY Genre, Title;  -- Multiple columns
```

Step 8: Limiting Results (LIMIT)

```sql
SELECT * FROM Books LIMIT 3;  -- First 3 books
SELECT * FROM Books ORDER BY YearPublished DESC LIMIT 2;  -- 2 newest books
```

Step 9: Aggregate Functions

```sql
-- Count total books
SELECT COUNT(*) FROM Books;

-- Count books by genre
SELECT COUNT(*) FROM Books WHERE Genre = 'Fiction';

-- Average year published
```

```sql
SELECT AVG(YearPublished) FROM Books;

-- Oldest and newest book
SELECT MIN(YearPublished) AS Oldest, MAX(YearPublished) AS Newest FROM Books;

-- Total members
SELECT COUNT(*) AS TotalMembers FROM Members;
```

Step 10: Grouping Data (GROUP BY)

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

-- Borrowings per member
SELECT MemberID, COUNT(*) AS BorrowCount
FROM Borrowings
GROUP BY MemberID;

-- Filter groups with HAVING
SELECT Genre, COUNT(*) AS BookCount
FROM Books
GROUP BY Genre
HAVING BookCount > 1;
```

Step 11: Updating Data (UPDATE)

```sql
-- Mark book as returned
UPDATE Borrowings
SET ReturnDate = '2024-06-25'
WHERE BorrowID = 2;

-- Update book availability
UPDATE Books
SET Available = FALSE
WHERE BookID = 2;

-- Update member email
UPDATE Members
SET Email = 'alice.j@email.com'
WHERE MemberID = 1;
```

Step 12: Deleting Data (DELETE)

```sql
-- Delete a specific borrowing record
DELETE FROM Borrowings WHERE BorrowID = 3;
```

```sql
-- Delete all fiction books (be careful!)
DELETE FROM Books WHERE Genre = 'Fiction';

-- Delete all records (use with extreme caution!)
DELETE FROM Borrowings;
```

Step 13: Joining Tables (JOIN)

INNER JOIN (matching records from both tables)

```sql
-- See who borrowed which books
SELECT Members.Name, Books.Title, Borrowings.BorrowDate
FROM Borrowings
INNER JOIN Members ON Borrowings.MemberID = Members.MemberID
INNER JOIN Books ON Borrowings.BookID = Books.BookID;
```

LEFT JOIN (all from left table, matching from right)

```sql
-- All members and their borrowings (even if they haven't borrowed)
SELECT Members.Name, Books.Title
FROM Members
LEFT JOIN Borrowings ON Members.MemberID = Borrowings.MemberID
LEFT JOIN Books ON Borrowings.BookID = Books.BookID;
```

RIGHT JOIN (all from right table, matching from left)

```sql
-- All books and who borrowed them
SELECT Books.Title, Members.Name
FROM Borrowings
RIGHT JOIN Books ON Borrowings.BookID = Books.BookID
LEFT JOIN Members ON Borrowings.MemberID = Members.MemberID;
```

Step 14: Aliases (AS)

```sql
SELECT
    m.Name AS MemberName,
    b.Title AS BookTitle,
    br.BorrowDate
FROM Borrowings br
JOIN Members m ON br.MemberID = m.MemberID
JOIN Books b ON br.BookID = b.BookID;
```

Step 15: Subqueries

```sql
-- Find members who borrowed books
SELECT Name
FROM Members
WHERE MemberID IN (SELECT DISTINCT MemberID FROM Borrowings);

-- Books never borrowed
SELECT Title
FROM Books
WHERE BookID NOT IN (SELECT BookID FROM Borrowings);

-- Most popular genre
SELECT Genre, COUNT(*) AS BorrowCount
FROM Books
WHERE BookID IN (SELECT BookID FROM Borrowings)
GROUP BY Genre
ORDER BY BorrowCount DESC
LIMIT 1;
```

Step 16: DISTINCT (Unique values)

```sql
SELECT DISTINCT Genre FROM Books;
SELECT DISTINCT MemberID FROM Borrowings;
```

Step 17: String Functions

```sql
-- Uppercase/Lowercase
SELECT UPPER(Title) FROM Books;
SELECT LOWER(Author) FROM Books;

-- Concatenate
SELECT CONCAT(Title, ' by ', Author) AS BookInfo FROM Books;

-- Substring
SELECT SUBSTRING(Title, 1, 10) FROM Books;

-- Length
SELECT Title, LENGTH(Title) AS TitleLength FROM Books;
```

Step 18: Date Functions

```sql
-- Current date
SELECT CURDATE();
SELECT NOW();

-- Date difference (days overdue)
SELECT
```

```
    Title,
    BorrowDate,
    DATEDIFF(CURDATE(), BorrowDate) AS DaysBorrowed
FROM Borrowings
JOIN Books ON Borrowings.BookID = Books.BookID
WHERE ReturnDate IS NULL;

-- Extract parts
SELECT YEAR(JoinDate), MONTH(JoinDate) FROM Members;
```

Step 19: Modifying Tables (ALTER TABLE)

```
-- Add new column
ALTER TABLE Books ADD Price DECIMAL(5,2);

-- Modify column
ALTER TABLE Books MODIFY Genre VARCHAR(50);

-- Delete column
ALTER TABLE Books DROP COLUMN Price;

-- Rename column
ALTER TABLE Members CHANGE Email EmailAddress VARCHAR(50);
```

Step 20: Creating Views (Virtual Tables)

```
CREATE VIEW ActiveBorrowings AS
SELECT
    m.Name AS Member,
    b.Title AS Book,
    br.BorrowDate
FROM Borrowings br
JOIN Members m ON br.MemberID = m.MemberID
JOIN Books b ON br.BookID = b.BookID
WHERE br.ReturnDate IS NULL;

-- Use the view
SELECT * FROM ActiveBorrowings;
```

Step 21: Indexes (Speed up queries)

```
CREATE INDEX idx_genre ON Books(Genre);
CREATE INDEX idx_author ON Books(Author);
```

Step 22: Transactions (All or Nothing)

```sql
START TRANSACTION;

INSERT INTO Borrowings (BookID, MemberID, BorrowDate)
VALUES (4, 3, CURDATE());

UPDATE Books SET Available = FALSE WHERE BookID = 4;

COMMIT;  -- Save changes
-- Or ROLLBACK; to undo
```

Practical Exercises

Try these queries on your library database:

1. Find all books published after 1950
2. List members who joined in 2024
3. Show books currently borrowed (ReturnDate IS NULL)
4. Count how many books each member has borrowed
5. Find the most popular author
6. List all Fantasy books sorted by title
7. Show members who have never borrowed a book
8. Calculate the average number of days books are borrowed

Quick Reference Card

- **CREATE**: Make new database/table
- **INSERT**: Add data
- **SELECT**: Retrieve data
- **WHERE**: Filter results
- **UPDATE**: Modify data
- **DELETE**: Remove data
- **JOIN**: Combine tables
- **GROUP BY**: Aggregate data
- **ORDER BY**: Sort results
- **LIMIT**: Restrict number of results

Would you like me to explain any specific part in more detail or help you practice with specific queries?