SQL Basics Tutorial Using a Blog Application

---

1. Introduction to SQL

What is SQL? Structured Query Language (SQL) is a standard language used to interact with relational databases. You can use SQL to create, modify, and query data.

---

2. Types of SQL Commands

- **DDL (Data Definition Language):** Used to define the structure of a database.
- `CREATE`, `ALTER`, `DROP`
- **DML (Data Manipulation Language):** Used to manipulate data within tables.
- `INSERT`, `UPDATE`, `DELETE`
- **DQL (Data Query Language):** Used to query data.
- `SELECT`
- **DCL (Data Control Language):** Used for permissions.
- `GRANT`, `REVOKE`

---

3. Create a Database and Tables

```
CREATE DATABASE blog_app;
USE blog_app;

CREATE TABLE users (
  id INT,
  username VARCHAR(50),
  email VARCHAR(100)
);

CREATE TABLE blogs (
  id INT,
  user_id INT,
  title VARCHAR(100),
  content TEXT,
  created_at DATE
);

CREATE TABLE comments (
  id INT,
  blog_id INT,
  user_id INT,
  comment TEXT,
  created_at DATE
);
```

---

4. Insert Data

```
INSERT INTO users VALUES (1, 'alice', 'alice@example.com');
INSERT INTO users VALUES (2, 'bob', 'bob@example.com');

INSERT INTO blogs VALUES (1, 1, 'First Blog', 'This is my first blog post.',
'2025-07-01');
INSERT INTO blogs VALUES (2, 2, 'Bob\'s Post', 'Hello world from Bob.',
'2025-07-02');

INSERT INTO comments VALUES (1, 1, 2, 'Great post, Alice!', '2025-07-03');
INSERT INTO comments VALUES (2, 2, 1, 'Thanks Bob!', '2025-07-04');
```

---

5. Basic Queries

```
-- Select all users
SELECT * FROM users;

-- Select only usernames
SELECT username FROM users;

-- Filtering with WHERE
SELECT * FROM blogs WHERE user_id = 1;

-- Using AND/OR
SELECT * FROM blogs WHERE user_id = 1 OR title LIKE '%Bob%';

-- Using BETWEEN
SELECT * FROM blogs WHERE created_at BETWEEN '2025-07-01' AND '2025-07-03';

-- Using LIKE
SELECT * FROM blogs WHERE title LIKE '%Post%';

-- LIMIT and OFFSET
SELECT * FROM blogs LIMIT 1 OFFSET 1;
```

---

6. Sorting and Grouping

```
-- Order by title
SELECT * FROM blogs ORDER BY title ASC;

-- Group comments by blog
SELECT blog_id, COUNT(*) as total_comments
```

```
FROM comments
GROUP BY blog_id;
```

-- Group by with HAVING clause

```
-- Blogs with more than 1 comment
SELECT blog_id, COUNT(*) AS total_comments
FROM comments
GROUP BY blog_id
HAVING COUNT(*) > 1;
```

---

7. JOINs with Practice Questions

Q1: Get blog titles along with usernames of authors

```
SELECT blogs.title, users.username
FROM blogs
JOIN users ON blogs.user_id = users.id;
```

Q2: List all comments with commenter and blog title

```
SELECT comments.comment, users.username, blogs.title
FROM comments
JOIN users ON comments.user_id = users.id
JOIN blogs ON comments.blog_id = blogs.id;
```

JOIN Types Diagram

```
           INNER JOIN              LEFT JOIN              RIGHT
JOIN              FULL OUTER JOIN

    A              B           A             B          A
B            A             B
     \            /             \           /
\           /              \           /
      \_________/               \________/
_________/               \__________/
      Only matching             All A + match         All B + match
All A + All B + match
```

---

8. Nested Queries and Subqueries

```
-- Users who have written at least one blog
SELECT * FROM users WHERE id IN (
  SELECT DISTINCT user_id FROM blogs
);

-- Count of blogs written by each user
SELECT username, (
  SELECT COUNT(*) FROM blogs WHERE blogs.user_id = users.id
) AS total_blogs
FROM users;
```

---

9. Correlated Subqueries

```
-- Get blogs with more than 1 comment
SELECT * FROM blogs b
WHERE (
  SELECT COUNT(*) FROM comments c WHERE c.blog_id = b.id
) > 1;
```

---

10. Alter Table Examples

```
-- Add a column
ALTER TABLE users ADD age INT;

-- Modify column type
ALTER TABLE blogs MODIFY title VARCHAR(150);

-- Drop a column
ALTER TABLE comments DROP COLUMN created_at;
```

---

11. Constraints (Adding After Learning Joins)

```
-- Add NOT NULL constraint
ALTER TABLE users MODIFY username VARCHAR(50) NOT NULL;

-- Add UNIQUE constraint
ALTER TABLE users ADD CONSTRAINT unique_email UNIQUE (email);

-- Add PRIMARY KEY
ALTER TABLE users ADD PRIMARY KEY (id);
```

```sql
-- Add FOREIGN KEY
ALTER TABLE blogs ADD CONSTRAINT fk_user FOREIGN KEY (user_id) REFERENCES
users(id);
```

---

12. Aggregate Functions

```sql
-- Total number of users
SELECT COUNT(*) FROM users;

-- Average age (after adding age)
SELECT AVG(age) FROM users;

-- Max and Min blog id
SELECT MAX(id), MIN(id) FROM blogs;

-- Sum of all blog ids
SELECT SUM(id) FROM blogs;
```

Summary

- You started with database creation
- Then added data step-by-step
- You learned how to filter, join, sort, group, and aggregate data
- Nested queries helped you ask advanced questions
- Constraints improve data quality

This is a foundational journey to SQL through a real-world blog application.