Learning SQL: A Comprehensive Guide to Operators, Expressions, and Database Operations

SQL Tutorial

July 2025

Contents

1 SQL Operators

An SQL operator is a reserved word or character used primarily in an SQL statement's `WHERE` clause to perform operations such as comparisons and arithmetic calculations. Operators specify conditions and serve as conjunctions for multiple conditions in a statement.

1.1 Arithmetic Operators

Arithmetic operators perform mathematical operations. Assuming variables `a = 10` and `b = 20`, the operators are:

- `+`: Addition (`a + b = 30`)
- `-`: Subtraction (`a - b = -10`)
- `*`: Multiplication (`a * b = 200`)
- `/`: Division (`b / a = 2`)
- `%`: Modulus (`b % a = 0`)

Examples:

```sql
SELECT 10 + 20;   -- Returns 30
SELECT 10 * 20;   -- Returns 200
SELECT 10 / 5;    -- Returns 2.0000
SELECT 12 % 5;    -- Returns 2
```

1.2 Comparison Operators

Comparison operators compare two values. Using `a = 10` and `b = 20`:

- `=`: Equal (`a = b` is false)
- `!=` or `<>`: Not equal (`a != b` is true)
- `>`: Greater than (`a > b` is false)
- `<`: Less than (`a < b` is true)
- `>=`: Greater than or equal (`a >= b` is false)
- `<=`: Less than or equal (`a <= b` is true)

Consider the `CUSTOMERS` table:

```sql
SELECT * FROM CUSTOMERS;
```

Output:

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Examples:

```sql
SELECT * FROM CUSTOMERS WHERE SALARY > 5000;
SELECT * FROM CUSTOMERS WHERE SALARY = 2000;
SELECT * FROM CUSTOMERS WHERE SALARY != 2000;
SELECT * FROM CUSTOMERS WHERE SALARY >= 6500;
```

1.3 Logical Operators

Logical operators combine conditions in a `WHERE` clause:

- `AND`: All conditions must be true.
- `OR`: At least one condition must be true.
- `NOT`: Reverses the condition.
- `LIKE`: Matches patterns using wildcards.
- `IN`: Matches any value in a list.
- `BETWEEN`: Matches values in a range.
- `EXISTS`: Checks for row existence.
- `ALL`: Compares to all values in a set.
- `ANY`: Compares to any value in a set.
- `IS NULL`: Checks for `NULL` values.

Examples:

```sql
SELECT * FROM CUSTOMERS WHERE AGE >= 25 AND SALARY >= 6500;
SELECT * FROM CUSTOMERS WHERE AGE >= 25 OR SALARY >= 6500;
SELECT * FROM CUSTOMERS WHERE NAME LIKE 'Ko%';
SELECT * FROM CUSTOMERS WHERE AGE IN (25, 27);
SELECT * FROM CUSTOMERS WHERE AGE BETWEEN 25 AND 27;
SELECT AGE FROM CUSTOMERS WHERE EXISTS
    (SELECT AGE FROM CUSTOMERS WHERE SALARY > 6500);
```

2 SQL Expressions

An expression combines values, operators, and SQL functions to evaluate to a value. They are used to query specific data sets.

2.1 Boolean Expressions

Boolean expressions fetch data based on matching a single value:

```sql
SELECT * FROM CUSTOMERS WHERE SALARY = 10000;
```

2.2 Numeric Expressions

Numeric expressions perform mathematical operations or use aggregate functions:

```sql
SELECT (15 + 6) AS ADDITION;
SELECT COUNT(*) AS RECORDS FROM CUSTOMERS;
```

2.3 Date Expressions

Date expressions retrieve system date and time:

```sql
SELECT CURRENT_TIMESTAMP AS Current_Timestamp;
```

3 Database and Table Operations

3.1 Creating a Database

The `CREATE DATABASE` statement creates a new database:

```sql
CREATE DATABASE TUTORIALSPOINT;
```

3.2 Selecting a Database

The `USE` statement selects a database:

```sql
USE TUTORIALSPOINT;
```

3.3 Creating a Table

The `CREATE TABLE` statement defines a table's structure:

```sql
CREATE TABLE CUSTOMERS (
    ID INT NOT NULL,
    NAME VARCHAR(20) NOT NULL,
    AGE INT NOT NULL,
    ADDRESS CHAR(25),
    SALARY DECIMAL(18, 2),
    PRIMARY KEY (ID)
);
```

3.4 Creating a Table from Another Table

Create a new table using data from an existing table:

```sql
CREATE TABLE SALARY AS
SELECT ID, SALARY
FROM CUSTOMERS;
```

3.5 Dropping a Table

The `DROP TABLE` statement removes a table and its data:

```sql
DROP TABLE SALARY;
```

3.6 Dropping a Database

The `DROP DATABASE` statement deletes a database:

```sql
DROP DATABASE TUTORIALSPOINT;
```

4 Data Manipulation

4.1 Inserting Data

The `INSERT INTO` statement adds new rows:

```sql
INSERT INTO CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);
```

4.2 Selecting Data

The `SELECT` statement retrieves data:

```sql
SELECT ID, NAME, SALARY FROM CUSTOMERS;
SELECT * FROM CUSTOMERS;
```

4.3 Updating Data

The `UPDATE` statement modifies existing records:

```sql
UPDATE CUSTOMERS
SET ADDRESS = 'Pune'
WHERE ID = 6;
UPDATE CUSTOMERS
SET ADDRESS = 'Pune', SALARY = 1000.00;
```

4.4 Deleting Data

The `DELETE` statement removes records:

```sql
DELETE FROM CUSTOMERS WHERE ID = 6;
DELETE FROM CUSTOMERS;
```

5 Conclusion

This document covers SQL operators, expressions, and basic database operations. Practice these queries in an SQL environment to solidify your understanding. For further learning, explore advanced topics like joins, subqueries, and indexing.