Operational & Logical Operators

Operators in SQL

SQL provides **operational operators** (arithmetic, comparison, and bitwise) and **logical operators** to manipulate, filter, and compare data. Below is an overview:

---

1. Operational Operators

a. Arithmetic Operators

Used for mathematical calculations.

	Operator	Description	Example
1	`+`	Addition	`SELECT salary + bonus AS total;`
2	`-`	Subtraction	`SELECT price - discount AS net;`
3	`*`	Multiplication	`SELECT quantity * price AS total;`
4	`/`	Division	`SELECT total / count AS average;`
5	`%`	Modulus (remainder)	`SELECT id % 2 AS is_odd;`

+ Neu

---

b. Comparison Operators

Used to compare values.

	Operator	Description	Example
1	`=`	Equal to	`SELECT * FROM users WHERE age = 30;`
2	`<>` or `!=`	Not equal to	`SELECT * FROM users WHERE age <> 30;`
3	`>`	Greater than	`SELECT * FROM orders WHERE amount > 50;`
4	`<`	Less than	`SELECT * FROM orders WHERE amount < 50;`
5	`>=`	Greater than or equal to	`SELECT * FROM orders WHERE amount >= 50;`
6	`<=`	Less than or equal to	`SELECT * FROM orders WHERE amount <= 50;`

7	`BETWEEN`	Within a range	`SELECT * FROM orders WHERE amount BETWEEN 10 AND 50;`
8	`IN`	Matches any value in a list	`SELECT * FROM users WHERE role IN ('Admin', 'User');`
9	`LIKE`	Pattern matching	`SELECT * FROM users WHERE name LIKE 'A%';`
10	`IS NULL`	Check for NULL values	`SELECT * FROM users WHERE manager_id IS NULL;`

+ Neu

---

c. Bitwise Operators

Used for bit-level operations (not common in most use cases).

	Operator	Description	Example
1	`&`	Bitwise AND	`SELECT 5 & 3; -- Output: 1`
2	`|`	`|`	Bitwise OR
3	`^`	Bitwise XOR	`SELECT 5 ^ 3; -- Output: 6`
4	`~`	Bitwise NOT	`SELECT ~5;`
5	`<<`	Left shift	`SELECT 5 << 2; -- Output: 20`
6	`>>`	Right shift	`SELECT 5 >> 2; -- Output: 1`

+ Neu

---

2. Logical Operators

Logical operators combine or negate multiple conditions.

	Operator	Description	Example
1	`AND`	Returns `TRUE` if all conditions are `TRUE`	`SELECT * FROM users WHERE age > 18 AND status = 'active';`
2	`OR`	Returns `TRUE` if any condition is `TRUE`	`SELECT * FROM users WHERE role = 'Admin' OR role = 'Manager';`
3	`NOT`	Negates a condition (`TRUE` becomes `FALSE`)	`SELECT * FROM users WHERE NOT age < 18;`
4	`BETWEEN`	Shortcut for range checks	`SELECT * FROM sales WHERE revenue BETWEEN 1000 AND 5000;`
5	`IN`	Checks if a value matches any in a list	`SELECT * FROM products WHERE category IN ('Electronics', 'Furniture', 'Books');`

6	`LIKE`	Pattern matching using `%` (any chars) or `_` (single char)	`SELECT * FROM users WHERE name LIKE 'A%'; -- Starts with 'A'`

+ Neu

Precedence of Logical Operators

When multiple logical operators are combined, SQL evaluates them in the following order:

1. **NOT**
2. **AND**
3. **OR**

Example:

```sql
SELECT *
FROM employees
WHERE NOT (age < 30) AND (role = 'Manager' OR department = 'HR');
```

Combining Operational and Logical Operators

You can combine these operators for more complex queries.

Example 1: Arithmetic + Logical

```sql
SELECT *
FROM products
WHERE price * quantity > 1000 AND category = 'Electronics';
```

Example 2: Logical with Comparison

```sql
SELECT *
FROM orders
WHERE order_date BETWEEN '2023-01-01' AND '2023-12-31'
  AND (status = 'Shipped' OR status = 'Delivered');
```

Best Practices

- Use **parentheses** to clarify logic when mixing `AND` and `OR`.
- Simplify complex conditions by breaking them into multiple lines for readability.
- Test your conditions with subsets of data to ensure correctness.

Would you like a practice exercise or more in-depth examples for any operator?