**The WHERE clause in SQL**

The WHERE clause is used to filter rows returned by a query. It specifies one or more conditions that each row must satisfy to be included in the result set.

**Basic syntax**

sql

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

* **condition**: a boolean expression that evaluates to true or false (or unknown). Only rows where the condition is true are returned.

**Common operators and examples**

**Comparison operators**

* =: equal
* != or <>: not equal
* <, >, <=, >=

sql

SELECT \* FROM employees WHERE department = 'Sales';

SELECT \* FROM products WHERE price > 100;

**Logical operators**

* AND: both conditions must be true
* OR: at least one condition must be true
* NOT: negates a condition

sql

SELECT \* FROM orders

WHERE status = 'Shipped' AND ship\_date >= '2024-01-01';

SELECT \* FROM customers

WHERE country = 'USA' OR country = 'Canada';

**BETWEEN, IN, LIKE**

* BETWEEN ... AND ...: within a range (inclusive)
* IN (...): value is in a list
* LIKE: pattern matching (wildcards)

sql

SELECT \* FROM sales

WHERE sale\_date BETWEEN '2024-01-01' AND '2024-12-31';

SELECT \* FROM users

WHERE role IN ('admin', 'moderator', 'editor');

SELECT \* FROM products

WHERE name LIKE 'Apple%'; -- starts with 'Apple'

SELECT \* FROM products

WHERE email LIKE '%@example.com'; -- ends with '@example.com'

**NULL handling**

* IS NULL and IS NOT NULL handle missing values

sql

SELECT \* FROM customers WHERE referral\_code IS NULL;

SELECT \* FROM orders WHERE delivered\_at IS NOT NULL;

**Functions and expressions**

You can use functions and expressions in WHERE clauses.

sql

SELECT \* FROM employees

WHERE LOWER(last\_name) = 'doe';

SELECT \* FROM events

WHERE event\_date >= NOW() - INTERVAL '7 days';

**Performance considerations**

* Use indexes on columns frequently used in WHERE clauses.
* Prefer SARGable conditions (simple comparisons high up, avoid wrapping indexed columns in functions).
* Combine conditions thoughtfully with AND/OR to avoid excessive row scans.
* Be mindful of data types (e.g., comparing strings to numbers can cause errors or poor performance).

**Examples: common scenarios**

1. **Filter by exact match**

sql

SELECT id, name

FROM students

WHERE student\_id = 12345;

1. **Filter by range and non-null**

sql

SELECT \*

FROM events

WHERE event\_date BETWEEN '2025-01-01' AND '2025-12-31'

AND location IS NOT NULL;

1. **Search with pattern and limit results**

sql

SELECT id, title

FROM articles

WHERE title LIKE '%SQL%'

ORDER BY published\_at DESC

LIMIT 10;

**Putting it together**

A WHERE clause can combine many conditions with AND, OR, and NOT to express complex filters:

sql

SELECT order\_id, customer\_id, total

FROM orders

WHERE status = 'Completed'

AND order\_date >= '2024-01-01'

AND (shipping\_city = 'New York' OR shipping\_city = 'Boston');

**SQL Operations: Arithmetic, Relational, Logical, and Membership**

Below is a concise guide to the main types of operations you’ll typically use in SQL expressions. Each category includes common operators, examples, and notes.

**1) Arithmetic Operations**

Arithmetic operators perform numeric calculations on columns or values.

* **Operators**: +, -, \*, /, % (modulo)
* **Notes**:
  + Division by zero yields an error (or NULL in some contexts, depending on dialect).
  + Use parentheses to control precedence.
* **Examples**:

-- Add two columns  
SELECT price \* quantity AS total\_cost  
FROM sales;

-- Compute a discounted price  
SELECT price \* (1 - discount\_rate) AS discounted\_price  
FROM products;

-- Remainder after division  
SELECT order\_id, total\_amount % 100 AS remainder  
FROM orders;

sql

## 2) Relational (Comparison) Operations

Relational operators compare values and yield a boolean result.

- \*\*Operators\*\*: `=`, `!=` or `<>`, `<`, `>`, `<=`, `>=`

- \*\*Notes\*\*:

- Some engines support `!=` as not equal; `<>` is the standard SQL form.

- Comparisons with `NULL` yield unknown; use `IS NULL`/`IS NOT NULL` for null checks.

- \*\*Examples\*\*:

```sql

-- Exact match

SELECT \* FROM employees WHERE department\_id = 5;

-- Not equal

SELECT \* FROM products WHERE category <> 'Obsolete';

-- Range check

SELECT \* FROM orders WHERE total\_amount >= 100;

-- NULL-safe comparisons (avoid direct NULL comparisons)

SELECT \* FROM users WHERE last\_login IS NULL;

**3) Logical Operations**

Logical operators combine boolean expressions to form more complex conditions.

* **Operators**: AND, OR, NOT
* **Notes**:
  + AND has higher precedence than OR.
  + Use parentheses to group conditions and clarify intent.
* **Examples**:

sql

-- Both conditions must be true

SELECT \* FROM orders

WHERE status = 'Shipped' AND ship\_date <= CURRENT\_DATE;

-- Either condition

SELECT \* FROM customers

WHERE country = 'USA' OR country = 'Canada';

-- Negation

SELECT \* FROM events WHERE NOT canceled;

**4) Membership Operations**

Membership operators test whether a value is in a set of values.

* **Operators**: IN (...), NOT IN (...)
* **Notes**:
  + Useful for filtering against a known list.
  + Be cautious with NULL inside NOT IN—it can yield unexpected results; prefer NOT EXISTS or carefully handle NULLs.
* **Examples**:

sql

-- Value is in a list

SELECT \* FROM users WHERE role IN ('admin', 'moderator', 'editor');

-- Value not in a list

SELECT \* FROM products WHERE category NOT IN ('Discontinued', 'Archived');