**📘 Day 3: Core SQL Operators & Logic**

🎯 **Goal**: Master logical operators, conditional expressions, and **subqueries for deeper filtering** and decision-making.

**⏱️ Time to Invest: 2 hours**

**Split into:**

* **40 mins: Concept Understanding**
* **40 mins: Guided Practice (Syntax Overview)**
* 40 mins: mins Self-practice + Debugging mistakes

**🧱 1. Core SQL Concepts in Detail**

**These tools help you write flexible, readable, and dynamic SQL queries. They're essential before learning joins, and used in filtering, reporting, and real-world analysis.**

**🔹 AS – Alias**

**Purpose**: Rename a column or table in your query result for better readability.

Table name: PARESH RANJAN ROUT as paresh, (new name paresh)

**Why use it**:

* Makes your results clear, especially in reports or joins.
* Useful for formatting column names (e.g., total\_price instead of SUM(price)).

**Syntax**:

SELECT column\_name AS new\_name FROM table\_name;

**Example**:

SELECT name AS product\_name, price AS cost FROM products;

⏩ Output will show:

| **product\_name** | **cost** |
| --- | --- |
| Pro Mouse | 500 |

**🔹 LIMIT & OFFSET – Pagination**

**Purpose**: Control how many rows your query returns, and where to start from.

**Use cases**:

* Show only the top results
* Paginate search results

**Syntax**:

SELECT \* FROM table\_name

ORDER BY column\_name

LIMIT number OFFSET skip;

**Example**:

SELECT \* FROM products

ORDER BY price DESC

LIMIT 3 OFFSET 2;

⏩ This skips the top 2 rows and returns the next 3 based on highest price.

**🔹 IS NULL / IS NOT NULL**

**Purpose**: Find rows where a value is missing (NULL) or exists.

**Why**:

* NULL is a special value meaning “unknown” or “not entered”.
* You **can’t** use = NULL (this will not work).

**Syntax**:

SELECT \* FROM orders

WHERE delivery\_date IS NULL;

**🔹 NOT**

**Purpose**: Reverse a condition (opposite of a match).

**Syntax**:

SELECT \* FROM products

WHERE NOT category = 'Laptop';

SELECT \* FROM orders

WHERE delivery\_date IS NOT NULL;

✅ Returns all products **except** those in the Laptop category.

**🔹 <> or != – Not Equal**

**Purpose**: Filters values **not matching** the condition.

**Syntax**:

SELECT \* FROM products

WHERE price <> 1000;

✅ Returns all products **except** those priced at 1000.

<> is standard SQL, != works in MySQL/Postgres but not all systems.

**🔹 AND – All Conditions Must Be True**

**Purpose**: Combine multiple filters. The row must meet **every** condition.

**Syntax**:

SELECT \* FROM products

WHERE category = 'Laptop' AND price > 40000;

✅ Shows only Laptops that cost more than 40,000.

**🔹 OR – Any Condition Can Be True**

**Purpose**: Use when **at least one** condition is enough.

**Syntax**:

SELECT \* FROM products

WHERE category = 'Laptop' OR price < 600;

✅ Returns any **Laptops**, plus **any product under 600**, even if not a laptop.

**40 mins: Guided Practice (Syntax Overview)**

**✅ Summary Table (Quick Look)**

| **Operator** | **Purpose** | **Example** |
| --- | --- | --- |
| AS | Rename column/table | SELECT name AS product\_name FROM products |
| LIMIT | Limit result count | SELECT \* FROM products LIMIT 5 |
| OFFSET | Skip rows | SELECT \* FROM products LIMIT 5 OFFSET 10 |
|  |  |  |
| IS NULL | Check for NULL | WHERE delivery\_date IS NULL |
| NOT | Reverse a condition | WHERE NOT category = 'Laptop' |
| **<> / !=** | Not equal | WHERE price <> 1000 |
| AND | All conditions must be true | WHERE price > 500 AND category = 'Laptop' |
| OR | Any condition can be true | WHERE category = 'Laptop' OR price < 500 |
|  |  |  |

40 mins: mins Self-practice + Debugging mistakes