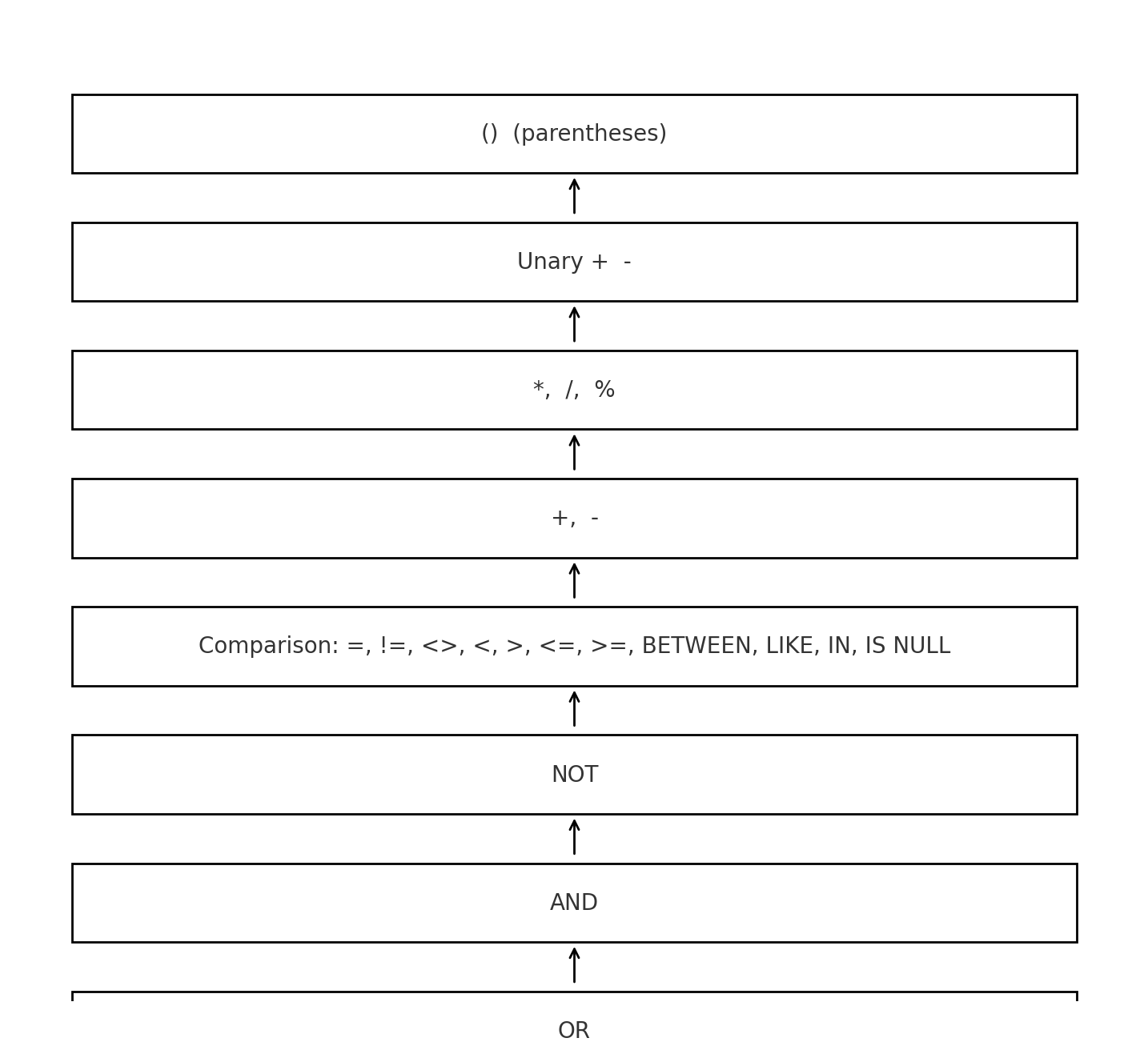
**SQL Operators & Query Solving**

**Agenda**

* Arithmetic Operators
* Comparison Operators
* Logical Operators
* Set Operators
* Other Operators
  + IN Operator
  + Between Operator
  + Like Operator
  + IS NULL/ IS NOT NULL
* String Operator
* Pattern Matching Operators
* Exists Operator
* Query Solving

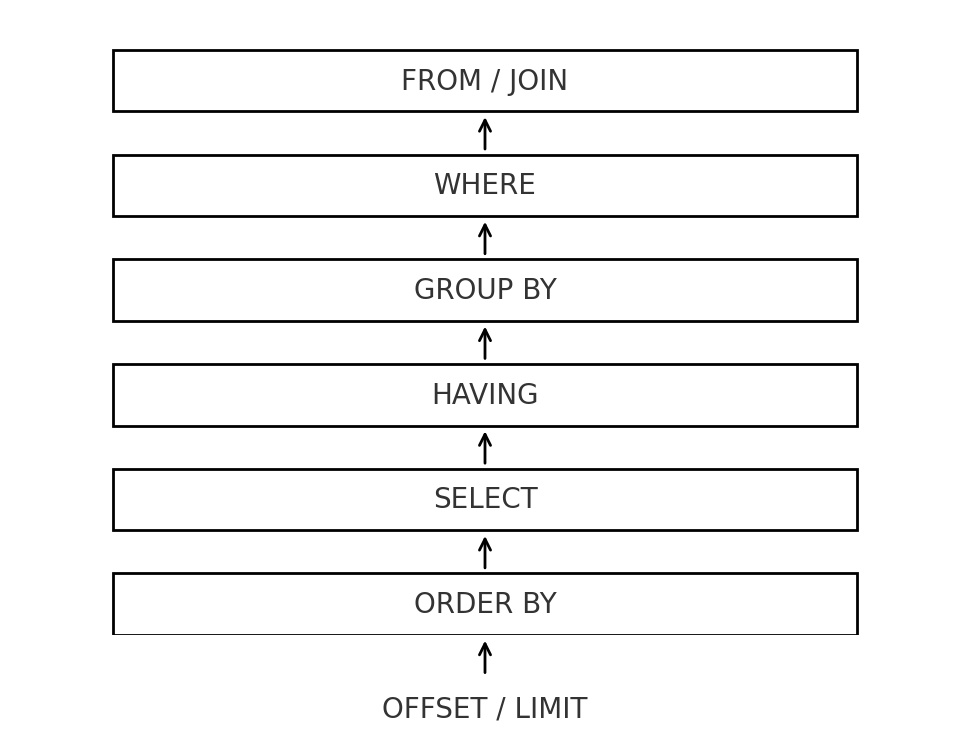
# Diagrams for Quick Understanding

## Operator Precedence (Highest → Lowest)



Note: Parentheses override everything. Arithmetic is evaluated before comparisons, which are evaluated before logical operators (NOT, AND, OR).

## SQL Logical Processing Order



Although you write SELECT first, SQL logically processes: -

FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY → LIMIT.

# Arithmetic Operators:

Use arithmetic to compute or transform numeric columns. These can appear in SELECT, WHERE, GROUP BY, HAVING, ORDER BY, and UPDATE clauses.

## Addition (+): Adds two values together.

## Subtraction (-): Subtracts one value from another.

## Multiplication (\*): Multiplies two values.

## Division (/): Divides one value by another. Beware of integer division in some DBs.

## Modulus (%): Remainder of division operation (e.g., bucketizing).

# 2. Comparison Operators

Comparison operators filter rows or drive conditional logic. Results are Boolean (TRUE / FALSE / UNKNOWN for NULLs). These operators typically appear in WHERE clause of an SQL Query.

## Equal to (=): Exact match or if one value is equal to another.

## Not Equal (!= or < >): Negation of equality.

## Greater Than (>): Strictly greater than a Specified number.

## Less Than (<): Strictly less than a Specified number.

## Greater Than or Equal (>=): Values greater than or equal to a specified number.

## Less Than or Equal (<=): Values less than or equal to a specified number.

**Note:** When NULL participates in a comparison, the result is UNKNOWN and the Row is filtered out by WHERE. Use IS NULL/IS NOT NULL to test for NULL explicitly.

# 3. Logical Operators

## It allows you to combine multiple conditions in a WHERE clause to filter data based on more complex criteria. These operators help in forming conditions that involve multiple criteria, determining whether a record meets one, several, or all those criteria

## AND: TRUE only if all conditions are TRUE.

## OR: TRUE if any condition is TRUE.

## NOT: Returns TRUE if the condition is False.

**Note:** Precedence: NOT runs before AND, which runs before OR. Use parentheses to avoid ambiguity.

## Logical Truth Table (simplified)

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **A AND B** | **A OR B** |
| TRUE | TRUE | TRUE | TRUE |
| TRUE | FALSE | FALSE | TRUE |
| FALSE | TRUE | FALSE | TRUE |
| FALSE | FALSE | FALSE | FALSE |

# 4. Set Operators:

Set operators combine result sets of two or more SELECT queries.

**Requirements:** Same Number of Columns + Compatible data types + Column order.

## UNION: Combines the result sets of two or more queries and removes duplicates.

## UNION ALL: Combines the result sets of two or more queries w/o removing duplicates.

## INTERSECT: Returns only the rows that are common between the two result sets.

## EXCEPT: Returns the rows from the first query that are not present in second query.

# 5. Other Operators

## IN: Allows you to specify multiple values in a WHERE clause. Shorthand for multiple OR comparisons.

## BETWEEN (inclusive): Selects values within a specified range.

## LIKE: Pattern matching in strings. Used with Wildcard characters (% and \_).

* **Wildcards:**

1. %: Represents zero or more characters.
2. \_: Represents a single character.

## IS NULL / IS NOT NULL: Tests for NULL explicitly.

# 6. String Operators & Functions:

Manipulate text for formatting, searching, or constructing labels.

## Concatenation ('||'): Concatenates two or more strings together.

## UPPER / LOWER: Case conversion.

## SUBSTRING: Extract a portion of a string.

## TRIM / LTRIM / RTRIM: Remove spaces or characters.

# 7. Pattern Matching Operators – Advanced

Power techniques for flexible matching within strings. Advanced form of LIKE operators.

## Examples:

## Q1: Find users whose Name starts with any vowel?

|  |
| --- |
| SELECT \* FROM dataset WHERE Name LIKE '[AEIOUaeiou]%'; |

## Q2: Find users whose Name does not start with a vowel?

|  |
| --- |
| SELECT \* FROM dataset WHERE Name LIKE '[^AEIOUaeiou]%'; |

## Note: The ^ inside the brackets negates the pattern, meaning it will match any string that does not start with a vowel.

# 8. EXISTS Operator:

EXISTS returns TRUE if the subquery returns at least one row. Efficient for membership/existence checks.

**Q1: Check if a User exist from Delhi?**

|  |
| --- |
| SELECT \* FROM dataset  WHERE EXISTS (SELECT 1 FROM dataset WHERE s.City = 'Delhi'); |

**Note:** EXISTS stops scanning as soon as it finds one matching row, which can be faster than IN for correlated checks.

# 9. Query Solving (Practice)

## Q1) Find the total salary of users who are from Delhi, Mumbai, or Chennai?

|  |
| --- |
| SELECT SUM(Salary) AS TotalSalary FROM dataset WHERE City IN ('Delhi', 'Mumbai', 'Chennai'); |

## Q2) Count users with salary between 40,000 and 80,000 in Delhi?

|  |
| --- |
| SELECT COUNT(\*) AS UserCount FROM dataset  WHERE Salary BETWEEN 40000 AND 80000  AND City = 'Delhi'; |

## Q3) Average salary of users not from Delhi?

|  |
| --- |
| SELECT AVG(Salary) AS AverageSalary FROM dataset WHERE City < > 'Delhi'; |

## Q4) Highest salary where Age < 30 OR Salary > 70,000?

|  |
| --- |
| SELECT MAX(Salary) AS MaxSalary FROM dataset WHERE Age < 30 OR Salary > 70000; |

## Q5) Minimum salary for IT or Finance?

|  |
| --- |
| SELECT MIN(Salary) AS MinSalary FROM dataset WHERE Department IN ('IT', 'Finance'); |

## Q6) Total users in each city?

|  |
| --- |
| SELECT City, COUNT(\*) AS TotalUsers FROM dataset GROUP BY City; |

## Q7) Total salary by department for users older than 30?

|  |
| --- |
| SELECT Department, SUM(Salary) AS TotalSalary FROM dataset WHERE Age > 30 GROUP BY Department; |

## Q8) Users from Delhi/Mumbai with Age > global average age?

|  |
| --- |
| SELECT \* FROM dataset WHERE City IN ('Delhi', 'Mumbai')  AND Age > (SELECT AVG(Age) FROM dataset); |

## Q9) Total salary in each department where salary > department average?

|  |
| --- |
| SELECT Department, SUM(Salary) AS TotalSalary FROM dataset d WHERE Salary > (SELECT AVG(Salary) FROM dataset s WHERE s.Department = d.Department) GROUP BY Department; |

## Q10) Count users with Name starting 'A' and Salary > 60,000?

|  |
| --- |
| SELECT COUNT(\*) AS UserCount FROM dataset WHERE Name LIKE 'A%'  AND Salary > 60000; |

## Q11) Average salary where Department is not 'HR'?

|  |
| --- |
| SELECT AVG(Salary) AS AverageSalary FROM dataset WHERE Department < > 'HR'; |

## Q12) Highest salary per department?

|  |
| --- |
| SELECT Department, MAX(Salary) AS MaxSalary FROM dataset GROUP BY Department; |

## Q13) Count of users per city with Salary > 50,000?

|  |
| --- |
| SELECT City, COUNT(\*) AS UserCount FROM dataset WHERE Salary > 50000 GROUP BY City; |

## Q14) Count users whose Salary is NULL?

|  |
| --- |
| SELECT COUNT(\*) AS NullSalaryUsers FROM dataset WHERE Salary IS NULL; |