CONDITIONAL (OPERATOR)

What is Conditional Operator ?

Conditional operators, also known as ternary operators, are operators that take three operands and evaluate a condition. Depending on whether the condition is true or false, these operators return different values. The most common conditional operator is the ternary conditional operator (**?:**), which is a shorthand way of expressing an **if-else** statement.

Here's the basic syntax of the ternary conditional operator:

plaintextCopy code

condition ? expression_if_true : expression_if_false

- If the **condition** is true, the operator evaluates to **expression_if_true**.
- If the **condition** is false, the operator evaluates to **expression_if_false**.

Example in a Programming Language (JavaScript):

javascriptCopy code

let age = 20; let eligibility = (age >= 18) ? "Eligible to vote" : "Not eligible to vote"; console.log(eligibility);

What is a logical Operator ?

Logical operators (**AND**, **OR**, and **NOT**) are used to combine or manipulate conditions in logical expressions. These operators are commonly used in programming and database queries to create more complex conditions based on multiple criteria.

1. **AND Operator (&& in many programming languages):**
    - **Definition:** Returns true only if both conditions on either side of the operator are true.

Example:

javascriptCopy code

let age = 25; let hasLicense = true; if (age >= 18 && hasLicense) { console.log("Can drive"); } else { console.log("Cannot drive"); }

In this example, the condition **(age >= 18 && hasLicense)** is true only if the person is 18 or older and has a valid license.

2. **OR Operator (|| in many programming languages):**
    - **Definition:** Returns true if at least one of the conditions on either side of the operator is true.

Example:

javascriptCopy code

let isWeekend = false; let isHoliday = true; if (isWeekend || isHoliday) { console.log("Time to relax"); } else { console.log("Regular work day"); }

In this example, the condition **(isWeekend || isHoliday)** is true if it's either the weekend or a holiday.

3. **NOT Operator (! in many programming languages):**
    - **Definition:** Returns the opposite boolean value of the condition it precedes. If the condition is true, **NOT** makes it false, and if it's false, **NOT** makes it true.

Example:

javascriptCopy code

let hasPermission = false; if (!hasPermission) { console.log("Access denied"); } else { console.log("Access granted"); }

In this example, the condition **!hasPermission** is true if **hasPermission** is false, and vice versa.

These logical operators are not limited to programming; they are also widely used in relational databases for constructing complex queries with multiple conditions.

Database Example (using SQL):

sqlCopy code

SELECT * FROM employees WHERE age >= 30 AND department = 'IT';
-- This SQL query retrieves employees who are 30 years or older and work in the IT department.

Understanding and using these logical operators are fundamental to constructing conditional statements and queries in various programming languages and database systems.

What is Aggregate Function ?

Aggregate functions in databases are functions that perform a calculation on a set of values and return a single value. These functions operate on a group of rows and are commonly used in conjunction with the GROUP BY clause to perform calculations on subsets of data. Here are some common aggregate functions with definitions and examples:

1. **COUNT():**
    - **Definition:** Counts the number of rows in a result set or the number of non-null values in a column.

Example:

sqlCopy code

SELECT COUNT(*) FROM orders;

This query counts the total number of rows in the "orders" table.

2. **SUM():**
    - **Definition:** Calculates the sum of values in a numeric column.

Example:

sqlCopy code

SELECT SUM(total_amount) FROM sales;

This query calculates the total sum of the "total_amount" column in the "sales" table.

3. **AVG():**
    - **Definition:** Calculates the average (mean) value of a numeric column.

Example:

sqlCopy code

SELECT AVG(salary) FROM employees WHERE department = 'IT';

This query calculates the average salary of employees in the "IT" department.

4. **MIN():**
    - **Definition:** Finds the minimum value in a column.

Example:

sqlCopy code

SELECT MIN(price) FROM products;

This query retrieves the minimum price from the "products" table.

5. **MAX():**
    - **Definition:** Finds the maximum value in a column.

Example:

sqlCopy code

SELECT MAX(score) FROM exam_results;

This query retrieves the maximum score from the "exam_results" table.

These aggregate functions can be used with the GROUP BY clause to perform calculations on subsets of data based on certain criteria. For example:

sqlCopy code

SELECT department, AVG(salary) FROM employees GROUP BY department;

This query calculates the average salary for each department in the "employees" table.

Aggregate functions are essential for summarizing and analyzing data in relational databases, and they play a crucial role in generating meaningful insights from large datasets.