SQL Arithmetic Operators

1. Addition (+)

Usage: Adds two values

Example:

```
SELECT 5 + 3 AS result;

Output: 8

SELECT price + 10 AS increased_price FROM products;
```

Explanation: In the first example, we simply add two constants. In the second, we add 10 to the price column of each row in the products table.

2. Subtraction ()

Usage: Subtracts the second value from the first

Example:

```
SELECT 10  4 AS result;
Output: 6

SELECT salary  taxes AS net_salary FROM employees;
```

Explanation: The first example subtracts 4 from 10. The second calculates the net salary by subtracting taxes from the gross salary for each employee.

3. Multiplication (*)

Usage: Multiplies two values

Example:

```
SELECT 6 * 3 AS result;
Output: 18

SELECT quantity * price AS total_value FROM order_items;
```

ENTRI elevate

Explanation: The first example multiplies 6 by 3. The second calculates the total value of each order item by multiplying quantity and price.

4. Division (/)

Usage: Divides the first value by the second

Example:

```
SELECT 15 / 3 AS result;
```

Output: 5

```
SELECT total_sales / number_of_days AS average_daily_sales FROM sales_report;
```

Explanation: The first example divides 15 by 3. The second calculates the average daily sales by dividing total sales by the number of days.

5. Modulus (%)

Usage: Returns the remainder of a division operation

Example:

```
SELECT 17 % 5 AS result;
```

Output: 2

```
SELECT order_id % 2 AS is_odd FROM orders;
```

Explanation: The first example returns the remainder when 17 is divided by 5. The second example determines if an order_id is odd (remainder 1) or even (remainder 0).

Important Notes:

Order of operations: SQL follows the standard PEMDAS (Parentheses, Exponents, Multiplication and Division, Addition and Subtraction) order.

NULL values: Any arithmetic operation involving a NULL value will result in NULL.

Data types: Be cautious when performing operations on different data types, as implicit type conversion may occur.

ENTRI elevate

Example combining multiple operators:

```
SELECT
   product_name,
   price,
   quantity,
   (price * quantity) AS total_cost,
   ((price * quantity) * 0.1) AS tax,
   (price * quantity) + ((price * quantity) * 0.1) AS total_with_tax
FROM
   order_items;
```

This query calculates the total cost, tax, and total with tax for each order item.

SQL Comparison Operators

ENTRI elevate

1. Equal to (=)

Usage: Checks if two values are equal

Example:

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

Explanation: This query retrieves all employees in the Sales department.

2. Not equal to (<> or !=)

Usage: Checks if two values are not equal

Example:

```
SELECT * FROM products WHERE category <> 'Electronics';
```

Explanation: This query selects all products that are not in the Electronics category.

3. Greater than (>)

Usage: Checks if the first value is greater than the second

Example:

```
SELECT * FROM orders WHERE total_amount > 1000;
```

Explanation: This query retrieves all orders with a total amount exceeding 1000.

4. Less than (<)

Usage: Checks if the first value is less than the second

Example:

```
SELECT * FROM inventory WHERE quantity < 10;
```

Explanation: This query finds all inventory items with a quantity less than 10.

5. Greater than or equal to (>=)

Usage: Checks if the first value is greater than or equal to the second

Example:

```
SELECT * FROM employees WHERE hire_date >= '20220101';
```

Explanation: This query selects all employees hired on or after January 1, 2022.

6. Less than or equal to (<=)

Usage: Checks if the first value is less than or equal to the second

Example:

```
SELECT * FROM products WHERE price <= 50;
```

Explanation: This query retrieves all products with a price of 50 or less.

7. BETWEEN

Usage: Checks if a value is within a range (inclusive)

Example:

```
SELECT * FROM orders WHERE order_date BETWEEN '20230101' AND '20231231';
```

Explanation: This query selects all orders placed in the year 2023.

ENTRI elevate

8. IN

Usage: Checks if a value matches any value in a list

Example:

```
SELECT * FROM products WHERE category IN ('Electronics', 'Appliances',
'Computers');
```

Explanation: This query retrieves products in the Electronics, Appliances, or Computers categories.

9. LIKE

Usage: Checks if a string matches a pattern

Example:

```
SELECT * FROM customers WHERE last_name LIKE 'S%';
```

Explanation: This query finds all customers whose last name starts with 'S'.

10. IS NULL / IS NOT NULL

Usage: Checks if a value is NULL or not NULL

Example:

```
SELECT * FROM employees WHERE manager_id IS NULL;
```

Explanation: This query retrieves all employees who don't have a manager (i.e., toplevel employees).

Important Notes:

Case sensitivity: Depending on the database system and configuration, string comparisons may or may not be casesensitive.

NULL comparisons: Use IS NULL or IS NOT NULL for comparing with NULL values. Other comparison operators with NULL always result in NULL.

Combining conditions: You can use AND, OR, and NOT to combine multiple conditions in a WHERE clause.

ENTRI elevate

Example combining multiple comparison operators:

```
SELECT *
FROM products
WHERE (category = 'Electronics' OR category = 'Computers')
  AND price BETWEEN 100 AND 1000
  AND stock_quantity > 0
  AND product_name LIKE '%Laptop%';
```

This query finds all instock laptop products in the Electronics or Computers categories, priced between 100 and 1000.

SQL Logical Operators

1. AND

Usage: Returns true if both conditions are true

Example:

```
SELECT * FROM employees
WHERE department = 'Sales' AND salary > 50000;
```

Explanation: This query retrieves all employees in the Sales department with a salary greater than 50000.

2. OR

Usage: Returns true if at least one condition is true

Example:

```
SELECT * FROM products
WHERE category = 'Electronics' OR price < 100;
```

Explanation: This query selects all products that are either in the Electronics category or have a price less than 100.

3. NOT

Usage: Negates a condition

Example:

```
SELECT * FROM customers WHERE NOT country = 'USA';
```

Explanation: This query retrieves all customers who are not from the USA.

4. IN

Usage: Checks if a value matches any value in a list

Example:

```
SELECT * FROM orders WHERE status IN ('Shipped', 'Delivered', 'Completed');
```

Explanation: This query selects all orders with a status of Shipped, Delivered, or Completed.

5. NOT IN

Usage: Checks if a value does not match any value in a list

Example:

```
SELECT * FROM products WHERE category NOT IN ('Clothing', 'Shoes', 'Accessories');
```

Explanation: This query retrieves all products that are not in the Clothing, Shoes, or Accessories categories.

6. EXISTS

Usage: Checks if a subquery returns any rows

Example:

```
SELECT * FROM departments WHERE EXISTS (SELECT 1 FROM employees WHERE employees.department_id = departments.id);
```

Explanation: This query selects all departments that have at least one employee.

7. NOT EXISTS

Usage: Checks if a subquery returns no rows

Example:

```
SELECT * FROM products WHERE NOT EXISTS (SELECT 1 FROM order_items WHERE order_items.product_id = products.id);
```

Explanation: This query retrieves all products that have never been ordered.

ENTRI elevate

8. BETWEEN ... AND ...

Usage: Checks if a value is within a range (inclusive)

Example:

```
SELECT * FROM employees WHERE hire_date BETWEEN '20200101' AND '20221231';
```

Explanation: This query selects all employees hired between January 1, 2020, and December 31, 2022.

9. IS NULL / IS NOT NULL

Usage: Checks if a value is NULL or not NULL

Example:

```
SELECT * FROM customers WHERE phone IS NULL;
```

Explanation: This query retrieves all customers who don't have a phone number recorded.

Important Notes:

Order of operations: SQL follows the standard order of operations for logical expressions (NOT, AND, OR).

Shortcircuit evaluation: Some database systems may use shortcircuit evaluation for AND and OR operations.

Parentheses: Use parentheses to explicitly define the order of operations in complex logical expressions.

ENTRI elevate

Example combining multiple logical operators:

```
SELECT *
FROM products
WHERE (category = 'Electronics' OR category = 'Computers')
  AND price BETWEEN 100 AND 1000
  AND manufacturer IN ('Apple', 'Samsung', 'Dell')
  AND (description LIKE '%laptop%' OR description LIKE '%smartphone%')
  AND NOT EXISTS (
    SELECT 1 FROM discontinued_products
    WHERE discontinued_products.product_id = products.id
  );
```

This query finds all active products in the Electronics or Computers categories, priced between 100 and 1000, manufactured by Apple, Samsung, or Dell, with "laptop" or "smartphone" in the description, and not in the discontinued products list.

ENTRI
elevate