06 MySQL Operators and Clauses

MySQL Operators and Clauses

Introduction to MySQL Operators and Clauses

MySQL, like other relational database management systems, provides a rich set of operators and clauses that help us interact with our data. These are essential components of SQL queries that allow us to filter, sort, group, and manipulate data effectively.

1. Comparison Operators

Comparison operators allow us to compare values and return results based on whether the comparison is true or false.

Basic Comparison Operators

Operator	Description	Example
`=`	Equal	`Salary = 15000`
`!=` or `<>`	Not equal	`Gender <> 'M'`
`<`	Less than	`Salary < 15000`
`>`	Greater than	`Salary > 15000`
`<=`	Less than or equal	`Salary <= 15000`
`>=`	Greater than or equal	`Salary >= 15000`

Let's see these in action:

```sql
-- Find programmers with a salary equal to $15,000
SELECT * FROM programmers WHERE Salary = 15000;

-- Find programmers with a salary not equal to $15,000
SELECT * FROM programmers WHERE Salary != 15000;

-- Find programmers with a salary less than $15,000
SELECT * FROM programmers WHERE Salary < 15000;

-- Find programmers with a salary greater than $15,000
SELECT * FROM programmers WHERE Salary > 15000;
```

The BETWEEN Operator

The `BETWEEN` operator selects values within a given range (inclusive).

```sql
-- Find programmers with salaries between $14,000 and $16,000
SELECT Programmer_Name, Salary
FROM programmers
WHERE Salary BETWEEN 14000 AND 16000;
```

The IN Operator

The `IN` operator allows you to specify multiple values in a WHERE clause.

```sql
-- Find programmers who use Python or JavaScript as their primary language
SELECT Programmer_Name, Primary_Language
FROM programmers
WHERE Primary_Language IN ('Python', 'JavaScript');
```

The LIKE Operator

The `LIKE` operator is used for pattern matching with wildcards:

- `%` represents zero, one, or multiple characters
- `_` represents a single character

```sql
-- Find programmers whose names start with 'B'
SELECT Programmer_Name
FROM programmers
WHERE Programmer_Name LIKE 'B%';

-- Find programmers whose names have 'ar' anywhere in their name
SELECT Programmer_Name
FROM programmers
WHERE Programmer_Name LIKE '%ar%';
```

The IS NULL and IS NOT NULL Operators

These operators check if a value is NULL or not NULL.

```sql
-- If we had programmers with no secondary language (we don't in our data)
SELECT Programmer_Name
FROM programmers
WHERE Secondary_Language IS NULL;

-- Find programmers who have a secondary language specified
SELECT Programmer_Name, Secondary_Language
FROM programmers
WHERE Secondary_Language IS NOT NULL;
```

2. Logical Operators

Logical operators combine multiple conditions.

Operator	Description
`AND`	Returns true if both conditions are true
`OR`	Returns true if either condition is true
`NOT`	Returns true if the condition is false

```sql
-- Find female programmers with a salary > $15,000
SELECT Programmer_Name, Gender, Salary
FROM programmers
WHERE Gender = 'F' AND Salary > 15000;
```

```sql
-- Find programmers who use either Python or JavaScript as their primary language
SELECT Programmer_Name, Primary_Language
FROM programmers
WHERE Primary_Language = 'Python' OR Primary_Language = 'JavaScript';

-- Find programmers who do not use Python as their primary language
SELECT Programmer_Name, Primary_Language
FROM programmers
WHERE NOT Primary_Language = 'Python';
```

3. Arithmetic Operators

MySQL supports standard arithmetic operators that you can use in your queries.

Operator	Description
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`%`	Modulo (remainder)

```sql
-- Calculate 10% bonus for each programmer
SELECT Programmer_Name, Salary, Salary * 0.1 AS Bonus, Salary + (Salary * 0.1) AS Total_Pay
FROM programmers;

-- Calculate profit from each software (revenue - development cost)
SELECT Programmer_Name, Software_Name,
       Software_Cost * Sold AS Revenue,
       Development_Cost AS Cost,
       (Software_Cost * Sold) - Development_Cost AS Profit
FROM software;
```

4. Basic SQL Clauses

The SELECT Clause

The `SELECT` clause specifies which columns you want to retrieve.

```sql
-- Select specific columns
SELECT Programmer_Name, Salary FROM programmers;

-- Select all columns
SELECT * FROM programmers;

-- Select with calculated columns
SELECT Programmer_Name, Salary, Salary * 1.1 AS Increased_Salary
FROM programmers;
```

The FROM Clause

The `FROM` clause specifies which table(s) to retrieve data from.

```sql
-- Basic FROM usage
SELECT * FROM programmers;

-- Using FROM with table alias
SELECT p.Programmer_Name, p.Salary
FROM programmers p;
```

The WHERE Clause

The `WHERE` clause filters records based on specified conditions.

```sql
-- Basic WHERE condition
SELECT * FROM programmers WHERE Salary > 15000;

-- Multiple conditions with logical operators
SELECT * FROM programmers
WHERE Primary_Language = 'Python' AND Salary > 14000;
```

The ORDER BY Clause

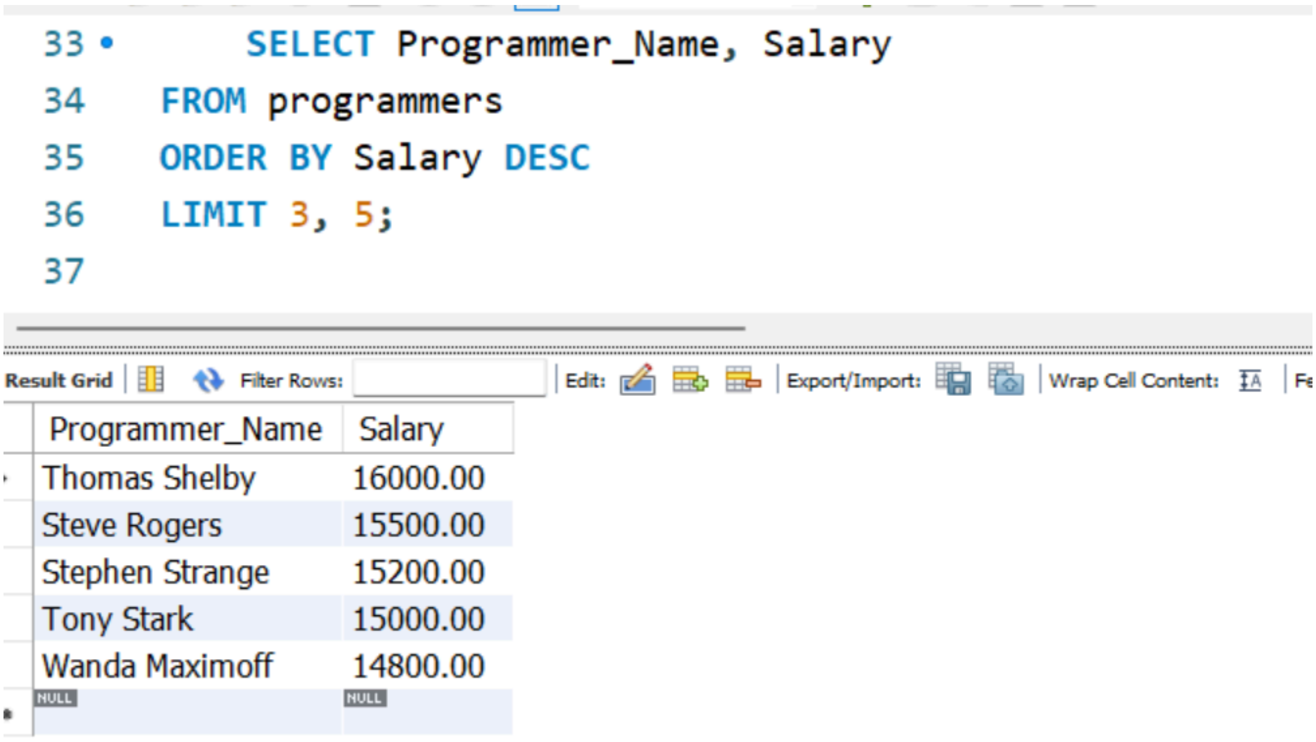
The `ORDER BY` clause sorts the result set.

```sql
-- Sort by salary in ascending order (default)
SELECT Programmer_Name, Salary
FROM programmers
ORDER BY Salary;

-- Sort by salary in descending order
SELECT Programmer_Name, Salary
FROM programmers
ORDER BY Salary DESC;

-- Sort by multiple columns
SELECT Programmer_Name, Primary_Language, Salary
FROM programmers
ORDER BY Primary_Language, Salary DESC;
```

The LIMIT Clause

The `LIMIT` clause restricts the number of rows returned.

```sql
-- Return only the top 5 highest-paid programmers
SELECT Programmer_Name, Salary
FROM programmers
ORDER BY Salary DESC
LIMIT 5;

-- Skip the first 3 results and return the next 5 (pagination)
SELECT Programmer_Name, Salary
FROM programmers
ORDER BY Salary DESC
LIMIT 3, 5;  -- Skip 3, take 5
```



```sql
SELECT Programmer_Name, Salary
FROM programmers
ORDER BY Salary DESC
LIMIT 3, 5;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Programmer_Name	Salary
Thomas Shelby	16000.00
Steve Rogers	15500.00
Stephen Strange	15200.00
Tony Stark	15000.00
Wanda Maximoff	14800.00
NULL	NULL