Mastering SELECT Statement

LEARN

1. Introduction to SELECT Statement

The SELECT statement is the foundation of querying in SQL. It is used to extract data from one or more tables based on specified criteria.

Key Features:

- Allows selecting specific columns or all columns using *.
- Can be combined with other clauses like WHERE, ORDER BY, GROUP BY, etc.
- Used in almost every SQL operation related to data retrieval.

Basic Syntax:

```
SELECT column1, column2, ...
FROM table_name;
```

Example:

```
SELECT StudentName, Age
FROM Students;
```

This query retrieves only the StudentName and Age columns from the Students table.

2. Using Aliases for Columns

Aliases in SQL provide temporary names for columns or tables, improving readability and clarity of result sets.

Benefits:

- Makes output more user-friendly.
- Helpful when using calculations or aggregate functions.

Syntax:

```
SELECT column_name AS alias_name
FROM table_name;
```

Example:

```
SELECT StudentName AS Name, Age AS Years
FROM Students;
```

This displays column headers as "Name" and "Years" in the output instead of their original names.

---

3. Filtering Data with WHERE Clause

The WHERE clause filters records to retrieve only those that meet certain conditions.

Use Cases:

- Retrieve records that match specific criteria.
- Narrow down large datasets.

Syntax:

```
SELECT *

FROM table_name

WHERE condition;
```

Example:

```
SELECT *

FROM Students

WHERE Age > 18;
```

This returns all student records where the age is greater than 18.

---

4. Applying Conditions to SELECT Queries

Conditions refine what data is retrieved. SQL supports many conditional operators.

Comparison Operators:

Operator	Description	Example
=	Equal to	Age = 20
>	Greater than	Age > 18
<	Less than	Age < 25

Operator	Description	Example
>=	Greater than or equal	Age >= 18
<=	Less than or equal	Age <= 30
<> or !=	Not equal	Age <> 20

These are used to build logic in WHERE clauses.

---

5. Combining Multiple Conditions with AND, OR

When one condition isn't enough, you can combine multiple conditions using logical operators.

Logical Operators:

- **AND**: All conditions must be true.
- **OR**: At least one condition must be true.
- **NOT**: Negates a condition.

Example with AND:

```
SELECT *
FROM Students
WHERE Age > 18 AND Grade = 'A';
```

Retrieves records where both conditions are satisfied.

Example with OR:

```
SELECT *

FROM Students

WHERE Age < 18 OR Grade = 'B';
```

Retrieves students who are either younger than 18 or have a Grade 'B'.

Combining AND and OR with Parentheses:

```
SELECT *

FROM Students

WHERE (Age > 18 AND Grade = 'A') OR City = 'Mumbai';
```

Parentheses help in grouping conditions to define precedence.

PRACTISE

Task 1: Basic SELECT

Write a query to retrieve employee names and salaries:

```
SELECT EmployeeName, Salary

FROM Employees;
```

Task 2: Using Aliases

Format column headers as "Product" and "Cost":

```
SELECT ProductName AS Product, Price AS Cost

FROM Products;
```

Task 3: WHERE Clause

Get a list of employees from a specific department:

```
SELECT *

FROM Employees

WHERE Department = 'Sales';
```

Task 4: Combined Conditions

Filter students who meet both age and grade requirements:

```
SELECT *

FROM Students

WHERE Age > 17 AND Grade = 'A';
```

FAQ

- **Q:** What does the SELECT statement do?
    - **A:** It fetches data from a database table and presents it in a structured form.
- **Q:** Can SELECT be used without a WHERE clause?
    - **A:** Yes, it then returns all rows from the table.
- **Q:** What's the purpose of aliases?
    - **A:** Aliases help rename column headers in the result set, improving clarity.
- **Q:** Can I use multiple conditions in WHERE?
    - **A:** Yes, using AND, OR, and NOT, combined with parentheses to manage logic.
- **Q:** Are there any limitations to WHERE clause?

- **A:** WHERE cannot be used with aggregate functions (like COUNT, AVG) directly. For that, use HAVING clause.