Pre-Joining Topics

Week 1: MYSQL Basics

DQL (Data Query Language)

DQL is used to fetch data from the database. The main command is SELECT, which retrieves records based on the query. The output is returned as a result set (a temporary table) that can be viewed or used in applications.

SELECT:

SELECT is used to retrieve data from one or more tables, either all records or specific results based on conditions. It returns output in a tabular format of rows and columns.

- Extracts data from tables.
- Targets specific or all columns (*).
- Supports filtering, sorting, grouping, and joins.
- Results are stored in a result set.

Syntax

SELECT column1,column2.... FROM table_name;

Parameters:

- column1, column2: columns you want to retrieve.
- table_name: name of the table you're querying.

Example:

SELECT CustomerName, LastName
FROM Customer;

WHERE:

WHERE clause is used to filter rows based on specific conditions. Whether you are retrieving, updating, or deleting data, WHERE ensures that only relevant records are affected. Without it, your query applies to every row in the table! The WHERE clause helps you:

- Filter rows that meet certain conditions
- Target specific data using logical, comparison and pattern-based operators
- Control SELECT, UPDATE, DELETE or even INSERT statements

Syntax:

- *SELECT column1, column2*
  FROM table_name
  WHERE column_name operator value;

Parameters:

- column1, column2: Columns you want to retrieve
- table_name: Table you are querying from
- operator: Comparison logic (e.g., =, <, >, LIKE)
- value: The value or pattern to filter against

Importance of WHERE Clause

The WHERE clause is critical for several reasons:

- Data Accuracy: Filters data to return only relevant rows
- Performance: Reduces the amount of scanned data
- Flexibility: Works with many operators and conditions

Example:

*SELECT * FROM Emp1 WHERE Age=24;*

ORDER BY:

The ORDER BY clause in SQL is used to sort query results based on one or more columns in either ascending (ASC) or descending (DESC) order. Whether you are presenting data to users or analyzing large datasets, sorting the results in a structured way is essential.

- By default, it sorts in ascending order (lowest to highest).
- To sort in descending order, use the DESC keyword.

Syntax:

*SELECT * FROM table_name ORDER BY column_name ASC | DESC;*

Key Terms:

- table_name: name of the table.
- column_name: name of the column according to which the data is needed to be arranged.
- ASC: to sort the data in ascending order.

- DESC: to sort the data in descending order.

Example:

*SELECT * FROM students ORDER BY ROLL_NO DESC;*

GROUP BY:

GROUP BY statement groups rows that have the same values in one or more columns. It is commonly used to create summaries, such as total sales by region or number of users by age group.

Its main features include:

- Used with the SELECT statement.
- Groups rows after filtering with WHERE.
- Can be combined with aggregate functions like SUM(), COUNT(), AVG(), etc.
- Filter grouped results using the HAVING clause.
- Comes after WHERE but before HAVING and ORDER BY.

Query execution order: FROM -> WHERE -> GROUP BY -> HAVING -> SELECT -> ORDER BY.

Syntax:

```
SELECT column1, aggregate_function(column2)
FROM table_name
WHERE condition
GROUP BY column1, column2;
```

Parameters:

- aggregate_function: function used for aggregation, e.g., SUM(), AVG(), COUNT().
- table_name: name of the table from which data is selected.
- condition: Optional condition to filter rows before grouping (used with WHERE).
- column1, column2: Columns on which the grouping is applied.

Example:

```
SELECT subject, COUNT(*) AS Student_Count
FROM Student
GROUP BY subject;
```

```
SELECT subject, year, COUNT(*)
FROM Student
GROUP BY subject, year;
```

HAVING:

HAVING clause filters results after applying aggregate functions. Unlike WHERE, which filters individual rows, HAVING works with aggregated results.

Its main features include:

- Filters results based on aggregate functions.
- Supports Boolean conditions (AND, OR).
- Applied after aggregate functions in the query.
- Useful for summary-level or aggregated filtering.

Syntax:

```
SELECT AGGREGATE_FUNCTION(column_name)
FROM table_name
HAVING condition;
```

Example:

```
SELECT SUM(Salary) AS Total_Salary
FROM Employee
HAVING SUM(Salary) >= 250000;
```

References

GeeksforGeeks