Sorting Query Result

LEARN

1. Introduction to ORDER BY Clause

The ORDER BY clause in SQL is used to sort the results of a SELECT query based on one or more columns. This clause helps to present query results in a meaningful order, which is essential for reporting and analysis.

Why Use ORDER BY?

- Makes large datasets more readable.
- Helps prioritize data (e.g., top-performing students, highest-priced items).
- Useful for ranking, pagination, or trend analysis.

Basic Syntax:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC|DESC];
```

- ASC stands for ascending order (default).
- DESC stands for descending order.

Example:

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

This will list all students from youngest to oldest, since it uses the default ascending order.

---

2. Sorting in Descending Order

To list results in reverse order, use the DESC keyword after the column name.

Example:

```
SELECT StudentName, Age
FROM Students
ORDER BY Age DESC;
```

This lists students from oldest to youngest.

Sorting in descending order is particularly useful when:

- Showing top scores.
- Displaying latest transactions.
- Sorting highest salary or prices first.

---

3. Sorting by Multiple Columns

You can sort by multiple columns by listing them in the ORDER BY clause, separated by commas. The query engine first sorts by the first column, then resolves any ties using the next column(s).

Example:

```
SELECT StudentName, Age, Grade
FROM Students
ORDER BY Age ASC, Grade DESC;
```

This query:

- Sorts all students by ascending age.
- For students with the same age, it sorts them by grade in descending order.

Practical Use Case:

- Displaying employees by department and then by name.
- Sorting sales by region and revenue.

---

4. Sorting by Column Index and Expressions

You can sort by a column's index number (position in SELECT clause) or by expressions.

Example Using Index:

```
SELECT StudentName, Age
FROM Students
ORDER BY 2 DESC;
```

This sorts by the second selected column (Age) in descending order.

Example Using Expression:

```
SELECT ProductName, Price * Quantity AS TotalValue
FROM Products
ORDER BY TotalValue DESC;
```

Here, the query sorts by a calculated column.

> Note: While using column indexes is allowed, it's not recommended in production because it reduces readability.

PRACTISE

Task 1: Sort Employees Alphabetically

List employee names and departments in alphabetical order.

```
SELECT EmployeeName, Department
FROM Employees
ORDER BY EmployeeName ASC;
```

Task 2: Sort Products by Price (High to Low)

Display product names and prices, sorted from most to least expensive.

```
SELECT ProductName, Price
FROM Products
ORDER BY Price DESC;
```

Task 3: Sort Orders by Date and Amount

List recent orders, and in case of the same date, sort them by amount.

```
SELECT OrderID, OrderDate, TotalAmount
FROM Orders
ORDER BY OrderDate DESC, TotalAmount ASC;
```

Task 4: Sort by Expression

Sort employees by total compensation (Salary + Bonus).

```
SELECT EmployeeName, Salary, Bonus
FROM Employees
```

```
ORDER BY (Salary + Bonus) DESC;
```

FAQ

- **Q:** What is the default sort order in SQL?
  - **A:** Ascending (ASC).
- **Q:** Can I sort by a column not listed in SELECT?
  - **A:** No, only if you are using SELECT * or explicitly include it in the SELECT clause.
- **Q:** How do I sort alphabetically?
  - **A:** Use ORDER BY column_name ASC for A-Z and DESC for Z-A.
- **Q:** Can I use expressions or functions in ORDER BY?
  - **A:** Yes. You can use mathematical expressions or SQL functions.
- **Q:** Is ORDER BY always the last clause in a SELECT query?
  - **A:** Yes, it should come after all other clauses like WHERE, GROUP BY, and HAVING.
- **Q:** Can I sort by a column's position?
  - **A:** Yes, though not recommended. Example: ORDER BY 2 refers to the second column in the SELECT list.