📘 Lesson 4: GROUP BY and ORDER BY in SQL

🎯 What You'll Learn

In this final lesson of the SQL basics series, you'll discover how to:

- Group and summarize your data using `GROUP BY`
- Sort and organize your data using `ORDER BY`
- Understand the difference between `GROUP BY` and `DISTINCT`
- Combine sorting with aggregation
- Customize sort order using multiple columns or column positions

---

⏪ Quick Recap

Up to now, you've learned how to:

- Create and populate tables
- Use `SELECT`, `FROM`, and `WHERE` to view and filter data

Now, you're ready to organize and summarize that data to uncover patterns and insights.

🧮 What is GROUP BY?

The `GROUP BY` clause helps you:

- Group data based on one or more columns (like Gender or Age)
- Perform **aggregate functions** (like COUNT, AVG, MAX) on each group
- Return **one row per group**, instead of one row per individual record

🆚 GROUP BY vs DISTINCT

Both identify unique values in a column, **but**:

- `DISTINCT` just shows which values exist
- `GROUP BY` not only shows the unique values, but **rolls up** data for each one — so you can count, average, and summarize

Example:

- `DISTINCT Gender` → Returns “Male” and “Female”
- `GROUP BY Gender` → Returns “Male” and “Female” **plus** how many of each there are

---

🔗 GROUPING WITH MULTIPLE COLUMNS

You’re not limited to grouping by one column.

For instance, grouping by **Gender and Age** will return a row for every unique combination (e.g., Male 30, Female 29, etc.).

This allows for more detailed summaries — like how many 30-year-old males vs. 29-year-old females you have.

Note: All grouped columns must be listed explicitly in your `GROUP BY` clause (unless they're derived fields).

✏️ Filtering with GROUP BY

You can still filter your data using `WHERE` before applying `GROUP BY`.

Example:

- Want to group employees **older than 31** by gender?
- You'd apply a filter first, then group and count the results.

This makes your grouping **more focused and relevant** to your query goals.

🧾 ORDER BY: Sorting Your Results

The `ORDER BY` clause lets you **sort your query results**.

By Default:

- SQL sorts in **ascending (A–Z / 0–9)** order

But you can also:

- Sort in **descending** order
- Sort by one or **multiple columns**
- Mix ascending and descending across columns

You can also:

- Sort by **column name**
- Sort by **column position** (like 1st, 2nd column)

💡 Pro tip: Sorting by position is handy in smaller queries when you don't want to type full column names.

🧠 Practical Example: Sorting a Summary

You might:

- Group by gender, count how many employees per group
- Then **sort** the results by the count
- Or sort by name for alphabetical output

You can control the exact presentation of your data — whether it's by value, frequency, or custom hierarchy.

📌 Recap

✅ `GROUP BY` helps summarize data by collapsing rows into categories
✅ You can combine it with aggregate functions like COUNT, AVG, MAX
✅ `ORDER BY` helps you control how results are displayed
✅ You can use multiple columns and even column positions for custom sorting
✅ These tools are essential for building real-world reports and dashboards

---

🎓 You've Completed SQL Basics!

Congratulations — you now understand the **core SQL commands**:

- `SELECT`, `FROM`, `WHERE`, `GROUP BY`, `ORDER BY`
- And how to use them together to filter, group, and sort data

You're ready to move on to **intermediate-level SQL**, which includes:

- `JOIN`s
- Subqueries
- Case logic
- Window functions
- And more!

🚀 What's Next?

Alex will be diving into **intermediate and advanced SQL topics** in future videos, as well as **portfolio projects** to build and showcase your skills.

Until then:

- Keep practicing
- Build your own tables
- Challenge yourself with data questions