Different Uses of Aggregation (Research & Reflect)

What is the difference between GROUP BY and ORDER BY?

GROUP BY	ORDER BY
• Used to group rows that have the same values in specified columns into summary rows.	• Used to sort the result set by one or more columns, either ascending (ASC, default) or descending (DESC).

Why do we use HAVING instead of WHERE when filtering aggregate results?

WHERE	HAVING
• filters rows *before* aggregation	• filters groups *after* aggregation.

We use **HAVING** instead of **WHERE** when filtering aggregate results because:

- **WHERE**: filters **individual rows before** any grouping or aggregation.
- **HAVING**: filters **aggregated groups after** the GROUP BY has been applied.

What are common beginner mistakes when writing aggregation queries?

- Using WHERE instead of HAVING to filter aggregates
- Selecting non-aggregated columns without grouping
- Forgetting to include all non-aggregated columns in GROUP BY
- Misunderstanding COUNT(*) vs COUNT(column)
- Not aliasing aggregate columns

When would you use COUNT(DISTINCT ...), AVG(...), and SUM(...) together?

- You use them together when you want a **multi-angle summary** of your data quantity, uniqueness, and magnitude all in one result.

How does GROUP BY affect query performance, and how can indexes help?

- GROUP BY can slow down queries on large datasets because it needs to **scan, sort, and aggregate** data.
- Creating **indexes on the GROUP BY column(s)** helps by reducing scan time and speeding up grouping.
- **Covering indexes** (that include all needed columns) can avoid table access entirely.
- **B-tree indexes** help avoid extra sorting during grouping.
- Use tools like EXPLAIN to check if indexes are used.