📘 Advanced SQL – Lesson 1: Common Table Expressions (CTEs)

🎯 What You'll Learn

In this lesson, you'll learn how to:

- Use Common Table Expressions (CTEs) for cleaner, modular queries
- Understand how CTEs differ from subqueries and temp tables
- Simplify complex queries using temporary, named result sets

---

🧠 What is a CTE?

A **Common Table Expression (CTE)** is a **temporary, named result set** that lives only during the execution of a single query.

CTEs help break down complex logic by:

- Letting you organize subqueries clearly
- Improving query readability and maintenance
- Making it easy to reuse logic in the same query

💡 You define a CTE using the `WITH` keyword, which is why it's sometimes called a "WITH query."

📌 Key Features of a CTE

- **Temporary:** Only exists during the execution of a single query

- **In-Memory:** Not written to tempdb like temp tables
- **Reusable:** Acts like a subquery but can be referenced more cleanly
- **Scoped:** Must be followed immediately by a `SELECT` statement — otherwise, it won't work

---

✅ Why Use a CTE?

CTEs are especially useful when:

- You want to simplify complex queries (e.g., using `PARTITION BY`)
- You need to calculate something once and reuse it
- You're working on data transformations or reporting queries

---

🔄 Example Use Case

Imagine you want to calculate the **average salary per gender** using a `PARTITION BY` clause, and then query from that data to select specific columns (like first name and average salary).

Instead of repeating the whole query logic, you can:

1. Create a CTE with the full logic once
2. Run simple `SELECT` statements against the CTE

This keeps your queries **modular, clean, and reusable**.

🚫 Things to Watch Out For

- CTEs **must** be followed by a query — you can't reference them later or separately

- CTEs **do not persist** — running the `SELECT` on its own won't work unless the CTE is included

- CTEs are not ideal for very large datasets where performance and persistence are critical — use temp tables or views instead in those cases

---

📌 Recap

✅ CTEs simplify your SQL logic by creating a temporary, reusable result set
✅ Use them to avoid repeating logic in complex queries
✅ They only exist during one query execution — they aren’t stored or reusable across scripts
✅ Ideal for breaking down logic like `PARTITION BY`, aggregations, or multi-step transformations