📘 Intermediate SQL – Lesson 6: Aliasing Columns and Tables

🎯 What You'll Learn

In this lesson, you'll learn how to:

- Use **aliases** to temporarily rename columns and tables
- Improve query readability with cleaner output and more meaningful labels
- Apply aliases in real-world scenarios including joins and aggregations

---

🧠 What is Aliasing in SQL?

Aliasing is a way to give a temporary name to a column or table in your SQL query.
It doesn't affect the data in your database — it's purely for readability and cleaner outputs.

✏️ Why Use Aliases?

Aliasing:

- Makes your query easier to read
- Helps label custom or calculated columns
- Simplifies long or complex table names
- Clarifies which table a column is from — especially when joining multiple tables

---

🧾 Column Aliases

You can rename a column in your output using an alias. This is helpful when:

- Combining columns (e.g., full names)
- Using aggregate functions (e.g., `AVG(Salary)`)
- Creating calculated or derived columns

💡 Example:
Instead of showing a column called `no column name`, alias it as `AverageSalary` so your output is clear and meaningful.

📁 Table Aliases

You can also shorten a **table name** using an alias. This is useful when:

- A table name is long
- You're using the same table multiple times (like in a join)
- You want to keep your query clean, especially with multiple joins

💡 Best practice:
Use **meaningful aliases** (e.g., `demo` for `EmployeeDemographics`) instead of single-letter aliases like `a` or `b`. This helps others (and future-you) understand your query faster.

📉 Bad Practice vs. Good Practice

❌ Poor Aliasing	✅ Good Aliasing
`SELECT a.EmployeeID` `FROM EmployeeDemographics AS a`	`SELECT demo.EmployeeID` `FROM EmployeeDemographics AS demo`

Avoid generic aliases like `a`, `b`, `c`. Instead, give your aliases context, like `sal` for `EmployeeSalary`.

📌 Recap

✅ Aliasing helps improve **query clarity and output presentation**
✅ Use it for columns (especially custom or aggregated ones)
✅ Use it for tables in complex joins or long queries
✅ Avoid one-letter aliases; aim for meaningful, self-explanatory names
✅ Aliases are **temporary** and **do not change** your actual data