🧠 SQL 100 Days Challenge – Day 69 Reflection

🗓️ Dataset Theme:

Products, Suppliers, Inventory, and Sales

This dataset simulated a retail supply chain where I practiced joins, aggregations, conditional logic, subqueries, window functions, ranking, and analytical business metrics.

🔍 Key Learnings:

1. **Join Logic:**
   Practiced 3-way joins between Products, Suppliers, and Inventory — built comfort identifying matching keys and grouping results correctly.
2. **Aggregation Mastery:**
   Used SUM(), AVG(), and ROUND() for revenue analysis by category — clear understanding of grouping and sorting by metrics.
3. **Conditional Logic:**
   Enhanced comfort with CASE statements by categorizing stock levels (Overstocked / Normal / Low Stock).
4. **Subqueries and CTEs:**
   Strengthened confidence with nested CTEs and correlated subqueries to analyze supplier performance and top-selling categories per region.
5. **Ranking and Analytical Functions:**
   Used RANK(), DENSE_RANK(), NTILE(), and LAG() — window functions now feel more intuitive and natural to use.
6. **Business Insights Queries:**
   Derived high-level KPIs such as total company revenue, top-performing products, and supplier tiers — reflecting real-world retail analytics.

⚙️ Challenges Faced:

- The **Bonus question (Stockout risk analysis)** was extremely tough — required logical breakdown using **average daily sales**, comparison with **reorder levels**, and dynamic stock forecasting.
- Initially struggled to calculate **company-wide total revenue** using CTE joins but later implemented it successfully with CROSS JOIN.
- Learned how to use nested logic efficiently for **risk detection** and **profitability evaluation**.

💡 **Concepts Practiced:**

- Multi-table joins (INNER JOIN, LEFT JOIN)
- Aggregation and grouping
- Conditional logic using CASE
- Subqueries (simple + correlated)
- CTE and nested CTE structures
- Ranking and analytical window functions (RANK, NTILE, LAG)
- Business-driven KPIs and reporting metrics

🌟 **Key Takeaway:**

Day 69 pushed the boundaries of **analytical query design** — from identifying top sellers to simulating **inventory health monitoring** and **revenue contribution** analytics.
These queries mirror real-world SQL use in **supply chain management**, **inventory forecasting**, and **profitability tracking**.