SQL 100 Days Challenge – Day 41 Reflection

Topic: Advanced Employee & Performance Analysis

Dataset Overview:

- **Employees Table:** Employee details including ID, Name, Department, HireDate, Salary, and ManagerID.
- **Performance Table:** Employee performance metrics by year — Rating, ProjectsCompleted, OvertimeHours.
- **Indexes:** Created on Name, Department, ManagerID (Employees) and EmpID (Performance) to improve query performance.

Key Concepts Practiced:

1. **Self-joins / Hierarchical Queries** – Fetching employees along with their managers.
2. **Filtering with Conditions** – Combining date and numeric conditions (HireDate < 2020 AND Salary > 60000).
3. **Aggregate Functions & Grouping** – Average performance ratings, yearly departmental ratings.
4. **Window Functions** – Ranking employees within departments, top N salaries.
5. **Joins** – INNER JOIN, LEFT JOIN, and joining the same table multiple times.
6. **Conditional Analysis** – Identifying performance improvement between years.
7. **Subqueries & CTEs** – Bonus challenge with total projects completed and ranking.
8. **Bonus:** Advanced ranking using RANK() over aggregate data.

Insights / Observations:

- Employees with **highest salary per department** were easily identified using RANK() and PARTITION BY.
- Performance ratings improved for some employees from 2021 → 2022 (e.g., Alice, Ethan).
- Employees with significant **overtime (>20 hours)** may indicate workload patterns or high dedication.
- Managers (George, Hannah, Ian) oversee multiple employees; hierarchical queries are key for reporting.
- Bonus ranking query highlighted the **most productive employees** in terms of projects completed.

Skills Reinforced:

- Writing **multi-join queries** with self-references.
- Using **window functions** like RANK() and ROW_NUMBER() for advanced analysis.
- Aggregation combined with **conditional logic** (HAVING, CASE).
- Structuring **CTEs** for clarity in complex queries.
- Practical experience with **realistic HR & performance data** scenarios.

Next Steps / Practice Recommendations:

- Try identifying **employees at risk of low performance** based on average ratings and overtime trends.
- Write queries to **analyze salary growth trends** over hire dates.
- Explore **department-wise project efficiency** using total projects vs average rating.
- Practice **dynamic ranking queries** with ties handled differently (DENSE_RANK() vs RANK()).