🧠 Week 1: Advanced MySQL Sprint (5 Questions/Day)

Day 1: The Power of Window Functions

1. **Dense Rank vs Rank**: Find the 3rd highest salary in each department. If two employees have the same salary, they should share the same rank.
2. **Running Total**: Calculate the cumulative sum of sales for each month in 2025.
3. **Lead/Lag**: Find the difference in sales between the current month and the previous month for each product.
4. **Row Number**: Identify and select only the most recent order for every customer without using GROUP BY.
5. **NTILE**: Divide customers into 4 quartiles based on their total spending.

Day 2: Advanced Joins & Subqueries

1. **Self Join**: Find all employees who earn more than their managers.
2. **Anti-Join**: Find products that have never been ordered (use NOT EXISTS instead of NOT IN for better performance).
3. **Cross Join**: Generate a report showing all possible combinations of "Size" and "Color" for a product table, even if they don't exist in inventory.
4. **Correlated Subquery**: Find employees whose salary is higher than the average salary of their specific department.
5. **Multiple Joins**: Retrieve customer names, their total orders, and the name of the last product they purchased.

Day 3: Common Table Expressions (CTEs)

1. **Recursive CTE**: Generate a series of numbers from 1 to 10 (or a list of dates for the current month).
2. **Cleaning Duplicates**: Write a query using a CTE and ROW_NUMBER() to delete duplicate records from a table while keeping the one with the lowest ID.
3. **Multi-level Aggregation**: Use a CTE to first find the total sales per region, and then find the average sales across all regions.
4. **Top N per Group**: Using a CTE, find the top 2 highest-selling products in each category.
5. **Sequential Data**: Identify users who logged in for 3 or more consecutive days.

Day 4: Conditional Logic & Case Statements

1. **Pivoting Data**: Convert a "Long" table (ID, Month, Revenue) into a "Wide" table (ID, Jan_Rev, Feb_Rev, Mar_Rev).
2. **Dynamic Binning**: Categorize orders into 'Small' (< $100), 'Medium' ($100-$500), and 'Large' (> $500) and count them.
3. **Handling Nulls**: Use COALESCE to display 'No Manager' for employees who don't have a

manager_id.
4. **Complex Filters**: Select customers who have purchased 'Product A' AND 'Product B' but NEVER 'Product C'.
5. **Flagging**: Create a flag column that marks orders as 'Delayed' if the shipping date is > 5 days after the order date.

Day 5: Strings, Dates & Performance Tuning

1. **Regex**: Find all email addresses in a table that do not follow the standard username@domain.com format.
2. **Date Diff**: Calculate the average "Churn Time" (days between the first and last purchase) for customers.
3. **String Splitting**: Extract the first name from a full name column where the format is "Lastname, Firstname".
4. **Indexing Simulation**: Explain why a query using WHERE YEAR(date) = 2025 is slower than WHERE date >= '2025-01-01' AND date <= '2025-12-31'.
5. **Optimization**: Rewrite a query that uses OR across three different columns into a more efficient UNION query.

Day 6 & 7: Case Study Review

- **Day 6**: Pick a real-world dataset (like your Swiggy project data) and write 5 complex business questions for it.
- **Day 7**: Review and refine your code. Ensure all your scripts have clear comments explaining the **"Why"** behind the logic.