### 🏗️ Project: SQL Analytics for Retail Business Insights

#### ****Objective****

As a data engineer at a fictional retail company, ShopSmart, you're tasked with analyzing customer orders, product sales, and employee performance. You will work with a relational dataset involving customers, orders, employees, and products.

#### ****Dataset Overview****

You are provided with 4 CSV files representing tables in a database:

* **customers** – Customer information
* **orders** – Sales orders made by customers
* **products** – Product catalog
* **employees** – Employees who handled the orders

### 🗄️ ****Database Setup (Required)****

Before starting your analysis, you must:

1. Create a new PostgreSQL database (e.g., retail\_project).
2. Create 4 corresponding tables: customers, orders, products, and employees.
3. Import the provided CSV files into their respective tables.
4. Ensure appropriate data types:
   * order\_date: DATE
   * price: DECIMAL(10,2)
   * quantity: INTEGER
   * IDs: INTEGER or SERIAL PRIMARY KEY as appropriate

You may use pgAdmin, psql, or any ETL script to load the data.

#### ****Key Requirements****

You must write **SQL queries** to extract valuable business insights using:

* Joins (INNER, LEFT)
* Subqueries (correlated and non-correlated)
* Window functions (ROW\_NUMBER, RANK, SUM OVER, etc.)
* Views (create at least 2 views for reusable queries)

### 🔍 ****Task List****

1. **Customer Analysis**
   * Find the top 5 customers by total purchase amount.
   * Identify customers who haven’t made any orders.
   * Write a view called customer\_order\_summary with customer name, total order count, and total amount spent.
2. **Sales Analysis**
   * List the top 3 products with the highest total sales value.
   * Write a query to show monthly sales totals (by product) using a window function.
   * Identify any product that has not been ordered at all.
3. **Employee Performance**
   * Calculate the total sales handled by each employee.
   * Use RANK() to rank employees by their total sales.
   * Find the employee with the highest average order value.
4. **Advanced Queries**
   * Use a correlated subquery to find customers whose total order value is above the average total across all customers.
   * Create a view called high\_value\_orders listing orders where the order total exceeds the 90th percentile of all order totals.

### 💡 Tips

* Assume reasonable data types (e.g., order\_date as DATE, price as DECIMAL).
* Document your queries and views clearly.
* Optimize queries for readability and performance.

### ✅ Deliverables

* SQL scripts with all queries and views.
* An optional notebook or markdown file with brief explanations per query.
* Bonus: A report with 2–3 key business insights you found interesting.