**Aim of Practical 5: Write and execute SQL queries- subqueries, joins.**

**Practical 5 part 1**

SQL Subqueries

## What is a Subquery?

A **subquery** is a query inside another query, used to retrieve intermediate results before executing the main query.

## Types of Subqueries:

1. **Single-row subqueries** → Return a single value.
2. **Multi-row subqueries** → Return multiple values.
3. **Correlated subqueries** → Reference columns from the outer query.

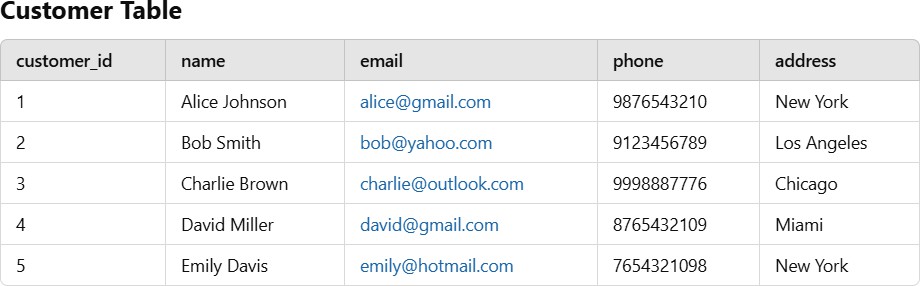
# Example Database: Supermarket

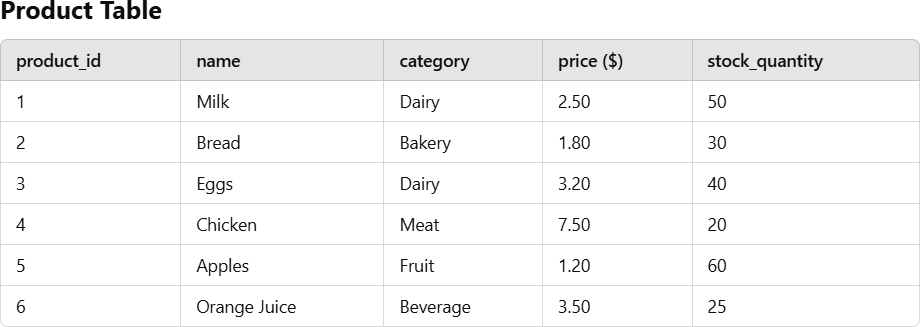
Use a **SupermarketDB** with the following tables:

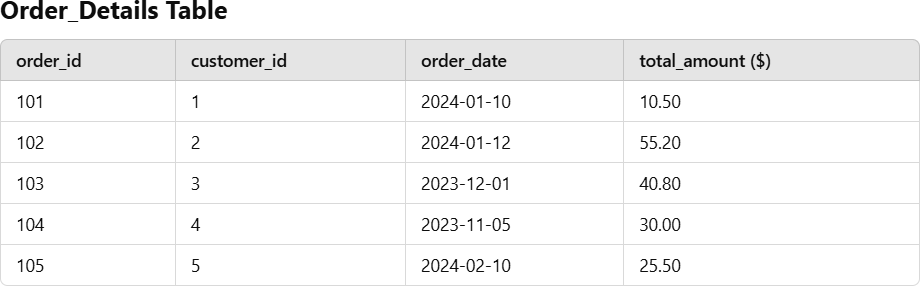
**Customer** (customer\_id, name, email, phone, address)

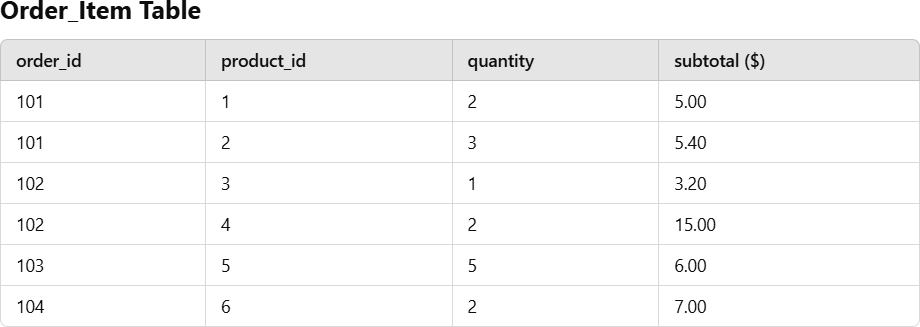
**Product** (product\_id, name, category, price, stock\_quantity) **Order\_Details** (order\_id, customer\_id, order\_date, total\_amount) **Order\_Item** (order\_id, product\_id, quantity, subtotal)

**Employeee** (employee\_id, name, role, salary, hire\_date)











# Examples of Subqueries

## Find customers who placed orders over $50.00

SELECT \* FROM Customer

WHERE customer\_id IN (SELECT customer\_id FROM Order\_Details WHERE total\_amount > 50);

## Retrieve products that cost more than the average product price

SELECT \* FROM Product

WHERE price > (SELECT AVG(price) FROM Product);

## Find employees earning more than the lowest manager’s salary

SELECT \* FROM Employeee

WHERE salary > (SELECT MIN(salary) FROM Employeee WHERE role = 'Manager');

## Find employees hired after the most recent hire date of a cashier

SELECT \* FROM Employeee

WHERE hire\_date > (SELECT MAX(hire\_date) FROM Employeee WHERE role = 'Cashier');

## Find customers who haven’t placed any orders

SELECT \* FROM Customer

WHERE customer\_id NOT IN (SELECT customer\_id FROM Order\_Details);

## Find the name of the highest-paid employee

SELECT name FROM Employeee

WHERE salary = (SELECT MAX(salary) FROM Employeee);

## Retrieve the total revenue generated from orders placed in January 2024

SELECT SUM(total\_amount) FROM Order\_Details

WHERE order\_date BETWEEN TO\_DATE('2024-01-01', 'YYYY-MM-DD')

AND TO\_DATE('2024-01-31', 'YYYY-MM-DD');

## Find the most ordered product

SELECT name FROM Product

WHERE product\_id = (SELECT product\_id FROM (

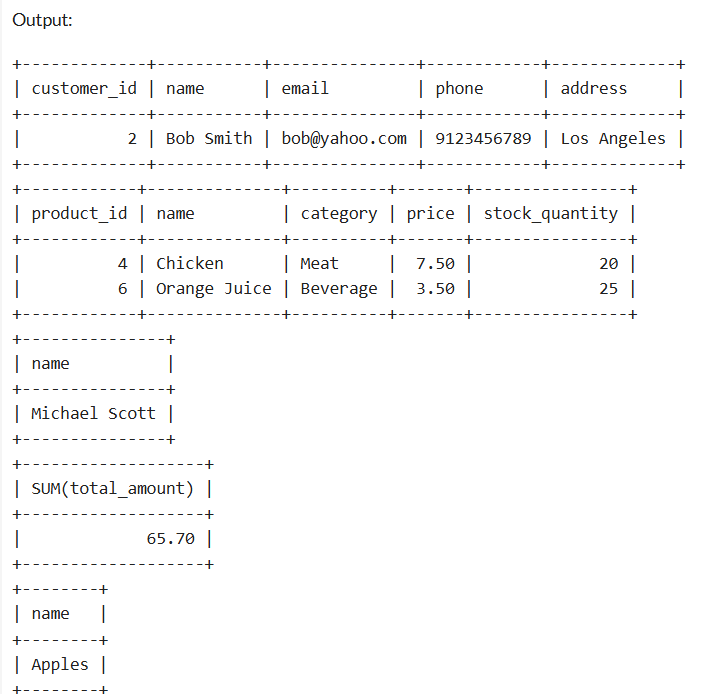
SELECT product\_id, SUM(quantity) AS total\_sold FROM Order\_Item

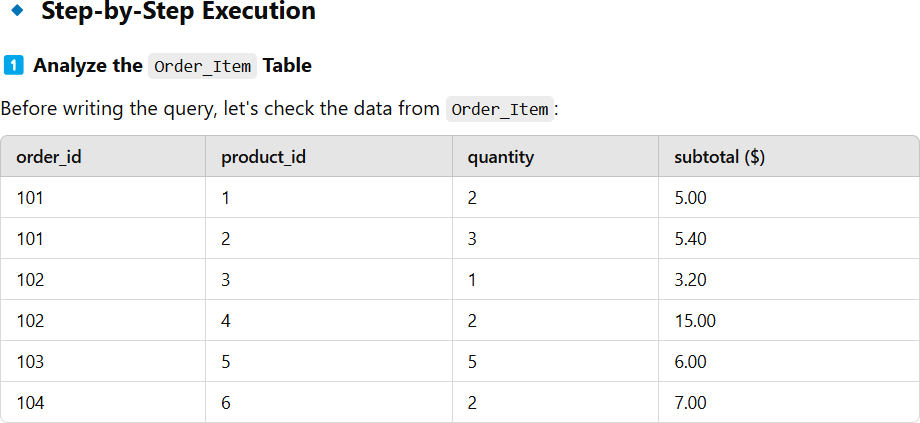
GROUP BY product\_id ORDER BY total\_sold DESC

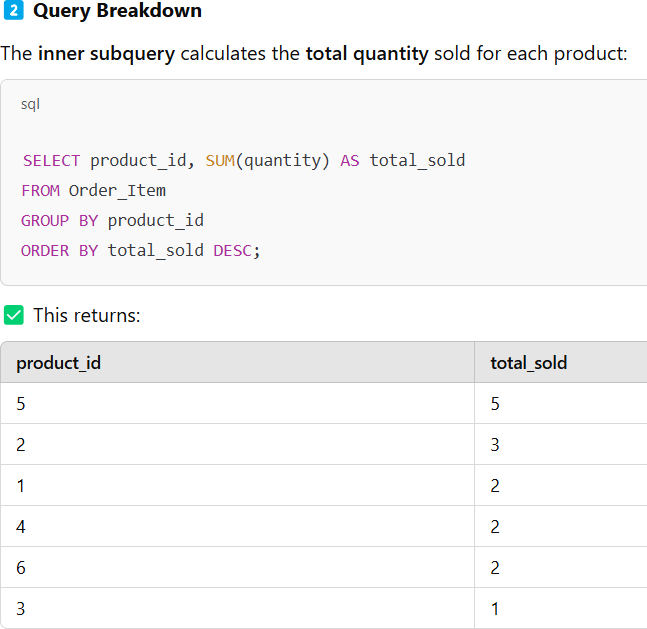
)

WHERE ROWNUM = 1

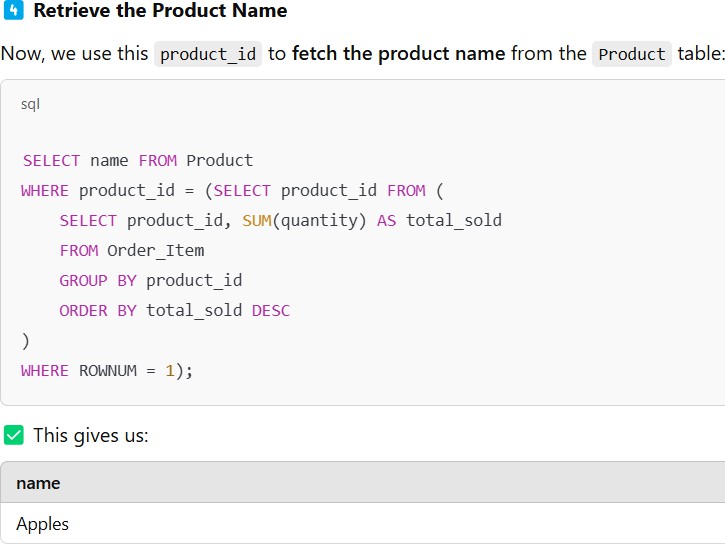
);







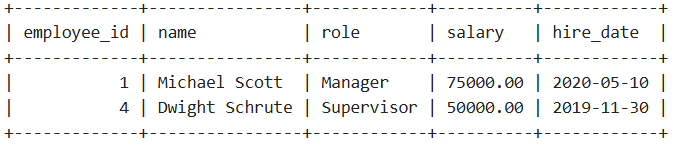




## Retrieve employees who earn above the average salary of all employees

SELECT \* FROM Employeee

WHERE salary > (SELECT AVG(salary) FROM Employeee);



## Find customers who placed orders only in 2023 but not in 2024

SELECT \* FROM Customer WHERE customer\_id IN (

SELECT customer\_id FROM Order\_Details

WHERE order\_date BETWEEN TO\_DATE('2023-01-01', 'YYYY-MM-DD')

AND TO\_DATE('2023-12-31', 'YYYY-MM-DD')

)

AND customer\_id NOT IN (

SELECT customer\_id FROM Order\_Details

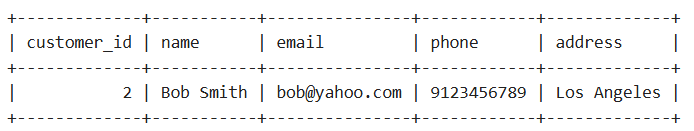
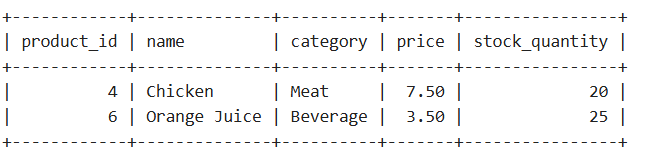
WHERE order\_date BETWEEN TO\_DATE('2024-01-01', 'YYYY-MM-DD')

AND TO\_DATE('2024-12-31', 'YYYY-MM-DD')

);



## Subquery Tasks

1. Find customers who placed orders over **$50**.
2. 
3. Retrieve products that cost more than the **average product price**.
4. 
5. Find employees hired after the **most recent hire date of a cashier**.
6. List customers who **haven’t placed any orders**.
7. Retrieve employees who earn **above the average salary**.
8. 