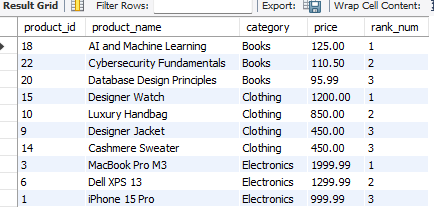
**Advanced SQL**

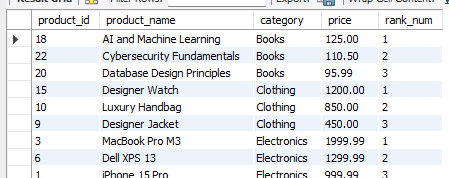
**Exercise 1: Ranking and Window Functions**

| **CREATE TABLE products (  product\_id INT PRIMARY KEY,  product\_name VARCHAR(100),  category VARCHAR(50),  price DECIMAL(10, 2) );**  **INSERT INTO products (product\_id, product\_name, category, price) VALUES**  **(1, 'iPhone 15 Pro', 'Electronics', 999.99),**  **(2, 'Samsung Galaxy S24', 'Electronics', 899.99),**  **(3, 'MacBook Pro M3', 'Electronics', 1999.99),**  **(4, 'iPad Air', 'Electronics', 599.99),**  **(5, 'AirPods Pro', 'Electronics', 249.99),**  **(6, 'Dell XPS 13', 'Electronics', 1299.99),**  **(7, 'Sony WH-1000XM5', 'Electronics', 399.99),**  **(8, 'Nintendo Switch', 'Electronics', 299.99),**  **(9, 'Designer Jacket', 'Clothing', 450.00),**  **(10, 'Luxury Handbag', 'Clothing', 850.00),**  **(11, 'Premium Jeans', 'Clothing', 180.00),**  **(12, 'Silk Dress', 'Clothing', 320.00),**  **(13, 'Leather Boots', 'Clothing', 280.00),**  **(14, 'Cashmere Sweater', 'Clothing', 450.00),**  **(15, 'Designer Watch', 'Clothing', 1200.00),**  **(16, 'Programming Masterclass', 'Books', 89.99),**  **(17, 'Data Science Handbook', 'Books', 75.50),**  **(18, 'AI and Machine Learning', 'Books', 125.00),**  **(19, 'Web Development Guide', 'Books', 65.99),**  **(20, 'Database Design Principles', 'Books', 95.99),**  **(21, 'Cloud Computing Essentials', 'Books', 89.99),**  **(22, 'Cybersecurity Fundamentals', 'Books', 110.50);**  **SELECT 'All Products:' as info;**  **SELECT \* FROM products ORDER BY category, price DESC;**  **SELECT**  **product\_id,**  **product\_name,**  **category,**  **price,**  **ROW\_NUMBER() OVER (PARTITION BY category ORDER BY price DESC) as row\_num**  **FROM products**  **ORDER BY category, price DESC;**  **SELECT**  **product\_id,**  **product\_name,**  **category,**  **price,**  **RANK() OVER (PARTITION BY category ORDER BY price DESC) as rank\_num**  **FROM products**  **ORDER BY category, price DESC;**  **SELECT**  **product\_id,**  **product\_name,**  **category,**  **price,**  **DENSE\_RANK() OVER (PARTITION BY category ORDER BY price DESC) as dense\_rank\_num**  **FROM products**  **ORDER BY category, price DESC;** |
| --- |

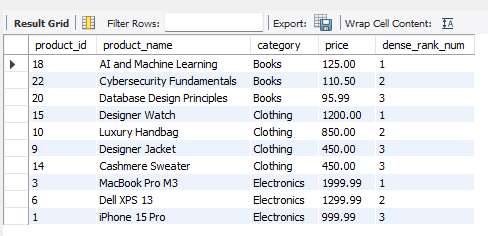
| **WITH ranked\_products AS (  SELECT   product\_id,  product\_name,  category,  price,  RANK() OVER (PARTITION BY category ORDER BY price DESC) as rank\_num  FROM products ) SELECT \* FROM ranked\_products WHERE rank\_num <= 3 ORDER BY category, price DESC;** |
| --- |

Output-

| WITH ranked\_products AS (  SELECT   product\_id,  product\_name,  category,  price,  ROW\_NUMBER() OVER (PARTITION BY category ORDER BY price DESC) as rank\_num  FROM products ) SELECT \* FROM ranked\_products WHERE rank\_num <= 3 |
| --- |

Output-

| WITH ranked\_products AS (  SELECT   product\_id,  product\_name,  category,  price,  DENSE\_RANK() OVER (PARTITION BY category ORDER BY price DESC) as dense\_rank\_num  FROM products ) SELECT \* FROM ranked\_products WHERE dense\_rank\_num <= 3 ORDER BY category, price DESC; |
| --- |

Output-

**Employee Management System SQL Exercises**

**Exercise 1: Create a Stored Procedure**

;

| DELIMITER //  CREATE PROCEDURE sp\_GetEmployeesByDepartment(  IN p\_DepartmentID INT ) BEGIN  SELECT   e.EmployeeID,  e.FirstName,  e.LastName,  d.DepartmentName,  e.Salary,  e.JoinDate  FROM Employees e  INNER JOIN Departments d ON e.DepartmentID = d.DepartmentID  WHERE e.DepartmentID = p\_DepartmentID; END // DELIMITER ; |
| --- |

| DELIMITER //  CREATE PROCEDURE sp\_InsertEmployee(  IN p\_FirstName VARCHAR(50),  IN p\_LastName VARCHAR(50),  IN p\_DepartmentID INT,  IN p\_Salary DECIMAL(10,2),  IN p\_JoinDate DATE ) BEGIN  DECLARE new\_employee\_id INT;    -- Get the next available EmployeeID  SELECT COALESCE(MAX(EmployeeID), 0) + 1 INTO new\_employee\_id FROM Employees;    -- Insert the new employee  INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID, Salary, JoinDate)  VALUES (new\_employee\_id, p\_FirstName, p\_LastName, p\_DepartmentID, p\_Salary, p\_JoinDate);    -- Return the new employee ID  SELECT new\_employee\_id AS NewEmployeeID; END //  DELIMITER ; |
| --- |

| CALL sp\_GetEmployeesByDepartment(3); |
| --- |



Output-

| CALL sp\_InsertEmployee('Sarah', 'Wilson', 2, 5800.00, '2025-06-29'); CALL sp\_GetEmployeesByDepartment(2); |
| --- |

Output-

**Exercise 5: Return Data from a Stored Procedure-**

**code-**

| **DELIMITER //**  **CREATE PROCEDURE sp\_GetEmployeeCountByDepartment(  IN p\_DepartmentID INT ) BEGIN  SELECT   d.DepartmentName,  COUNT(e.EmployeeID) AS TotalEmployees  FROM Departments d  LEFT JOIN Employees e ON d.DepartmentID = e.DepartmentID  WHERE d.DepartmentID = p\_DepartmentID  GROUP BY d.DepartmentID, d.DepartmentName; END //**  **DELIMITER ;** |
| --- |

**Output-**

| **CALL sp\_GetEmployeeCountByDepartment(2); CALL sp\_GetEmployeeCountByDepartment(1);** |
| --- |

