Advanced SQL

Exercise 1: Ranking and Window Functions

```sql
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;
```

```sql
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-

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	product_name	category	price	rank_num
18	AI and Machine Learning	Books	125.00	1
22	Cybersecurity Fundamentals	Books	110.50	2
20	Database Design Principles	Books	95.99	3
15	Designer Watch	Clothing	1200.00	1
10	Luxury Handbag	Clothing	850.00	2
9	Designer Jacket	Clothing	450.00	3
14	Cashmere Sweater	Clothing	450.00	3
3	MacBook Pro M3	Electronics	1999.99	1
6	Dell XPS 13	Electronics	1299.99	2
1	iPhone 15 Pro	Electronics	999.99	3

```sql
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-

product_id	product_name	category	price	rank_num
18	AI and Machine Learning	Books	125.00	1
22	Cybersecurity Fundamentals	Books	110.50	2
20	Database Design Principles	Books	95.99	3
15	Designer Watch	Clothing	1200.00	1
10	Luxury Handbag	Clothing	850.00	2
9	Designer Jacket	Clothing	450.00	3
3	MacBook Pro M3	Electronics	1999.99	1
6	Dell XPS 13	Electronics	1299.99	2
1	iPhone 15 Pro	Electronics	999.99	3

```sql
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-

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	product_name	category	price	dense_rank_num
18	AI and Machine Learning	Books	125.00	1
22	Cybersecurity Fundamentals	Books	110.50	2
20	Database Design Principles	Books	95.99	3
15	Designer Watch	Clothing	1200.00	1
10	Luxury Handbag	Clothing	850.00	2
9	Designer Jacket	Clothing	450.00	3
14	Cashmere Sweater	Clothing	450.00	3
3	MacBook Pro M3	Electronics	1999.99	1
6	Dell XPS 13	Electronics	1299.99	2
1	iPhone 15 Pro	Electronics	999.99	3

Employee Management System SQL Exercises

Exercise 1: Create a Stored Procedure

;

```sql
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-

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EmployeeID	FirstName	LastName	DepartmentName	Salary	JoinDate
3	Michael	Johnson	IT	7000.00	2018-07-30

```
CALL sp_InsertEmployee('Sarah', 'Wilson', 2, 5800.00, '2025-06-29');
CALL sp_GetEmployeesByDepartment(2);
```

Output-

EmployeeID	FirstName	LastName	DepartmentName	Salary	JoinDate
2	Jane	Smith	Finance	6000.00	2019-03-22
5	Sarah	Wilson	Finance	5800.00	2025-06-29

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);
```

	DepartmentName	TotalEmployees
▶	Finance	5

	DepartmentName	TotalEmployees
▶	HR	1