**1. SQL Exercise - Advanced concepts**

**Exercise 1: Ranking and Window Functions**

Goal: Use ROW\_NUMBER(), RANK(), DENSE\_RANK(), OVER(), and PARTITION BY. Scenario: Find the top 3 most expensive products in each category using different ranking functions. Steps:

Step 1: Use ROW\_NUMBER() to assign a unique rank within each category.

Code:

WITH ProductRanks AS (

SELECT

ProductID,

ProductName,

Category,

Price,

ROW\_NUMBER() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS RowNum

FROM

Products

)

SELECT

ProductID,

ProductName,

Category,

Price,

RowNum

FROM

ProductRanks

WHERE

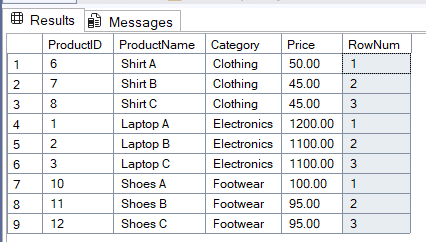
RowNum <= 3

ORDER BY

Category,

RowNum;

Output:



Step 2: Use RANK() and DENSE\_RANK() to compare how ties are handled.

Code:

WITH RankedProducts AS (

SELECT

ProductID,

ProductName,

Category,

Price,

RANK() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS RankNum,

DENSE\_RANK() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS DenseRankNum

FROM

Products

)

SELECT

ProductID,

ProductName,

Category,

Price,

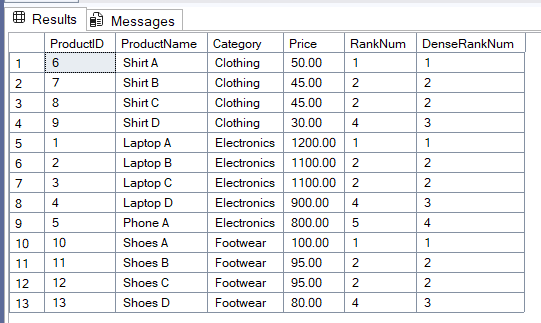
RankNum,

DenseRankNum

FROM

RankedProducts;

Output:



## **RANK()**

* Assigns a rank number to each row within a partition.
* If there are ties, the same rank is given to the tied rows, but the next rank(s) are skipped.
* So, it can leave gaps in the rank sequence.

## **DENSE\_RANK()**

* Also assigns a rank to each row.
* If there are ties, the same rank is given, but it does not skip any ranks.
* So, the ranking numbers are dense, with no gaps.

**4. SQL Exercise - Stored procedure**

**Exercise 1: Create a Stored Procedure**

Goal: Create a stored procedure to retrieve employee details by department.

Step 1:

Define the stored procedure with a parameter for DepartmentID.

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

EmployeeID,

FirstName,

LastName,

DepartmentID,

Salary,

JoinDate

FROM Employees

WHERE DepartmentID = @DepartmentID;

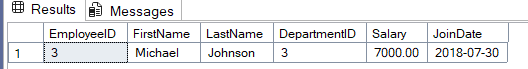
END;

Step 2: Write the SQL query to select employee details based on the DepartmentID.

Code:

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 3;

Output:



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

Goal: Create a stored procedure that returns the total number of employees in a department.

Step 1: Define the stored procedure with a parameter for DepartmentID.

Code:

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

COUNT(\*) AS EmployeeCount

FROM

Employees

WHERE

DepartmentID = @DepartmentID;

END;

Step 2: Write the SQL query to count the number of employees in the specified department.

Code:

