Skill- Advanced SQL

File -1 - SQL Exercise - Advanced concepts

Name - Exercise 1: Ranking and Window Functions

Code :-

1. **Create a New Database**

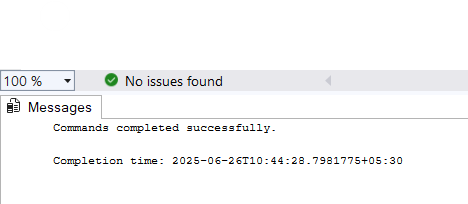
CREATE DATABASE OnlineRetailStore;

GO

USE OnlineRetailDB;

GO

**Output:-**



Step 2: Create the Products Table

Code:-

CREATE TABLE Products (

ProductID INT PRIMARY KEY IDENTITY(1,1),

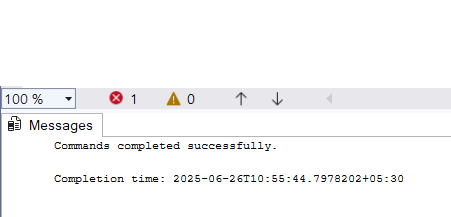
ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10, 2)

);

Output:-



Step 3: Insert Sample Data into Products Table

Code:-

INSERT INTO Products (ProductName, Category, Price)

VALUES

('Laptop Pro X', 'Electronics', 1500.00),

('Smartphone Z', 'Electronics', 800.00),

('Bluetooth Speaker', 'Electronics', 200.00),

('Gaming Console', 'Electronics', 500.00),

('4K Monitor', 'Electronics', 500.00),

('Leather Sofa', 'Furniture', 1200.00),

('Dining Table', 'Furniture', 1100.00),

('Office Chair', 'Furniture', 300.00),

('Bookshelf', 'Furniture', 300.00),

('TV Stand', 'Furniture', 450.00),

('Running Shoes', 'Footwear', 150.00),

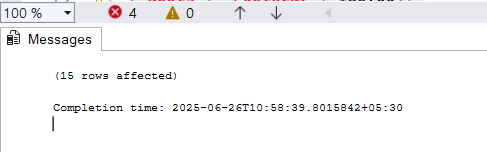
('Sandals', 'Footwear', 50.00),

('Sneakers', 'Footwear', 120.00),

('Formal Shoes', 'Footwear', 150.00),

('Boots', 'Footwear', 180.00);

Output :-



Find the top 3 most expensive products in each category using different ranking functions.

Steps:

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

Code :-

WITH RankedProducts AS (

SELECT

Category,

ProductName,

Price,

ROW\_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum

FROM Products

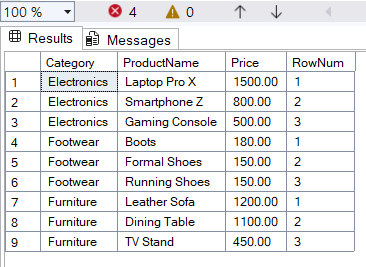
)

SELECT \*

FROM RankedProducts

WHERE RowNum <= 3;

Output :-



1. Use RANK() and DENSE\_RANK() to compare how ties are handled.

Code:-

SELECT

Category,

ProductName,

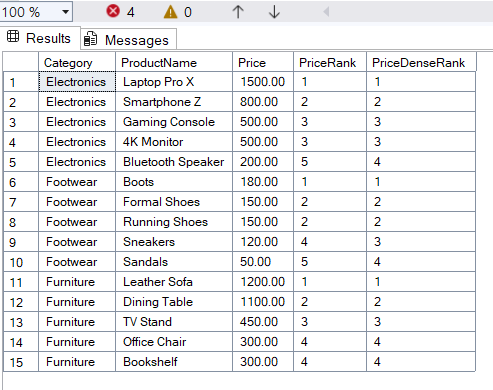
Price,

RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS PriceRank,

DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS PriceDenseRank

FROM Products;

Output:-



\* top 3 products in each category using RANK \*

Code:-

WITH Ranked AS (

SELECT

Category,

ProductName,

Price,

RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS PriceRank

FROM Products

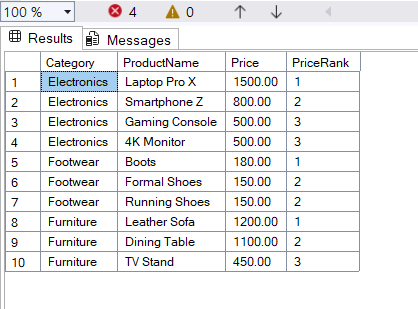
)

SELECT \*

FROM Ranked

WHERE PriceRank <= 3;

Output:-



\* And using DENSE\_RANK: \*

Code :-

WITH DenseRanked AS (

SELECT

Category,

ProductName,

Price,

DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS PriceDenseRank

FROM Products

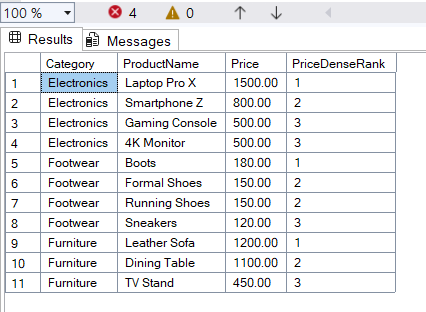
)

SELECT \*

FROM DenseRanked

WHERE PriceDenseRank <= 3;

Output:-



3. Use PARTITION BY Category and ORDER BY Price DESC

\* For ROW\_NUMBER() \*

Code :-

WITH RowNumberRanked AS (

SELECT

ProductID,

ProductName,

Category,

Price,

ROW\_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum

FROM Products

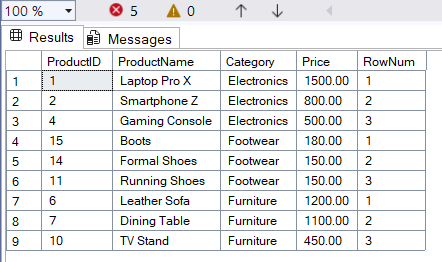
)

SELECT \*

FROM RowNumberRanked

WHERE RowNum <= 3;

Output:-



\* For RANK()\*

Code:-

WITH Ranked AS (

SELECT

ProductID,

ProductName,

Category,

Price,

RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS PriceRank

FROM Products

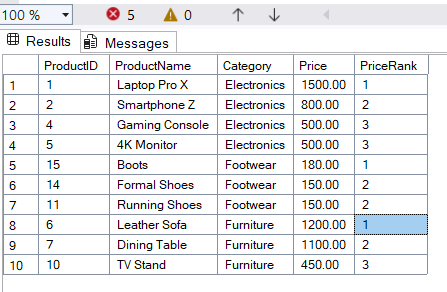
)

SELECT \*

FROM Ranked

WHERE PriceRank <= 3;

Output:-



\* For DENSE\_RANK() \*

Code:-

WITH DenseRanked AS (

SELECT

ProductID,

ProductName,

Category,

Price,

DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DensePriceRank

FROM Products

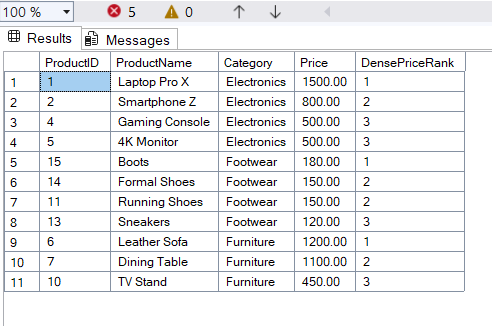
)

SELECT \*

FROM DenseRanked

WHERE DensePriceRank <= 3;

Output:-



File - 4. SQL Exercise - Stored procedure

Step 1: Create Tables

Code :-

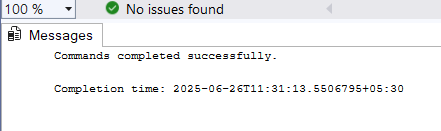
CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

Output:-



\* Create Employees table \*

Code:-

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY IDENTITY(1,1),

FirstName VARCHAR(50),

LastName VARCHAR(50),

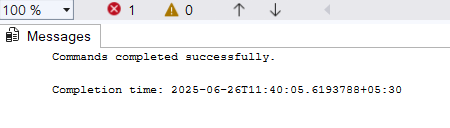
DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

Output:-



Step 2: Insert Sample Data

Code :-

\* Insert into Departments \*

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

\* Insert into Employees \*

INSERT INTO Employees (FirstName, LastName, DepartmentID, Salary, JoinDate) VALUES

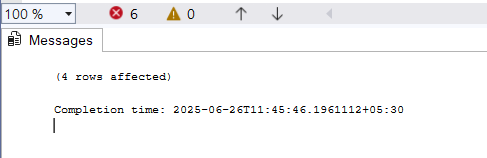
('John', 'Doe', 1, 5000.00, '2020-01-15'),

('Jane', 'Smith', 2, 6000.00, '2019-03-22'),

('Michael', 'Johnson', 3, 7000.00, '2018-07-30'),

('Emily', 'Davis', 4, 5500.00, '2021-11-05');

Output:-



Exercise 1: Create a Stored Procedure

-> Retrieve Employees by Department (Parameterized)

Code:-

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DeptID INT

AS

BEGIN

SELECT

EmployeeID,

FirstName,

LastName,

Salary,

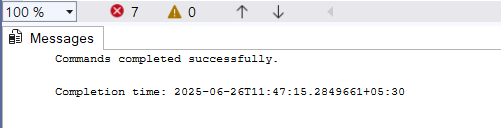
JoinDate

FROM Employees

WHERE DepartmentID = @DeptID;

END;

Output:-

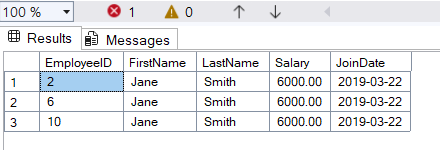


\* Example:- \*

Code:-

EXEC sp\_GetEmployeesByDepartment @DeptID = 2;

Output:-



-> Insert New Employee Procedure

Code:-

CREATE PROCEDURE sp\_InsertEmployee

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@DepartmentID INT,

@Salary DECIMAL(10,2),

@JoinDate DATE

AS

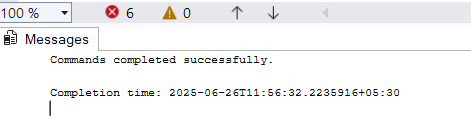
BEGIN

INSERT INTO Employees (FirstName, LastName, DepartmentID, Salary, JoinDate)

VALUES (@FirstName, @LastName, @DepartmentID, @Salary, @JoinDate);

END;

Output:-



\* Example:- \*

Code:-

EXEC sp\_InsertEmployee

@FirstName = 'Alice',

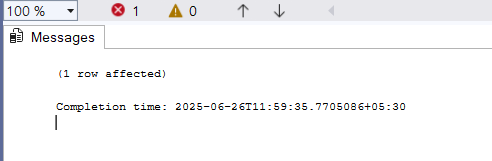
@LastName = 'Brown',

@DepartmentID = 1,

@Salary = 5200.00,

@JoinDate = '2022-09-01';

Output:-



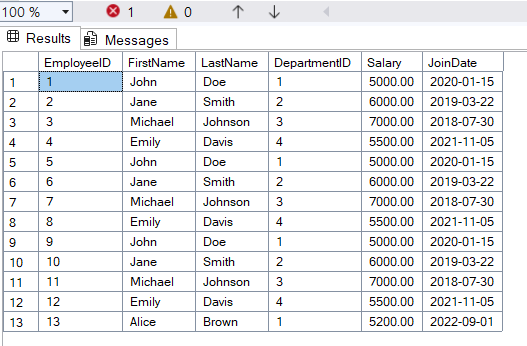
\* Verification Queries \*

// View All Employees

Code:-

SELECT \* FROM Employees;

Output:-

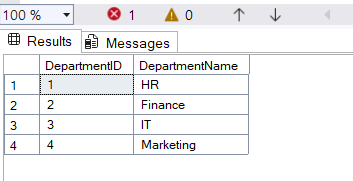


// View All Departments

Code:-

SELECT \* FROM Departments;

Output:-



Exercise 5: Return Data from a Stored Procedure

-> Create Stored Procedure:

Code:-

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DeptID INT

AS

BEGIN

SELECT

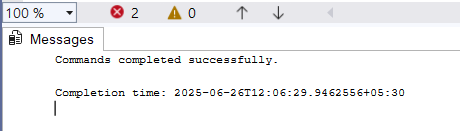
COUNT(\*) AS EmployeeCount

FROM Employees

WHERE DepartmentID = @DeptID;

END;

Output:-



-> Execute the Stored Procedure:

Code:-

EXEC sp\_GetEmployeeCountByDepartment @DeptID = 1;

Output:-

