**ADVANCED SQL**

**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:**

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

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

**3. Use PARTITION BY Category and ORDER BY Price DESC.**

Solution:  
1. SELECT

ProductID,

ProductName,

Category,

Price,

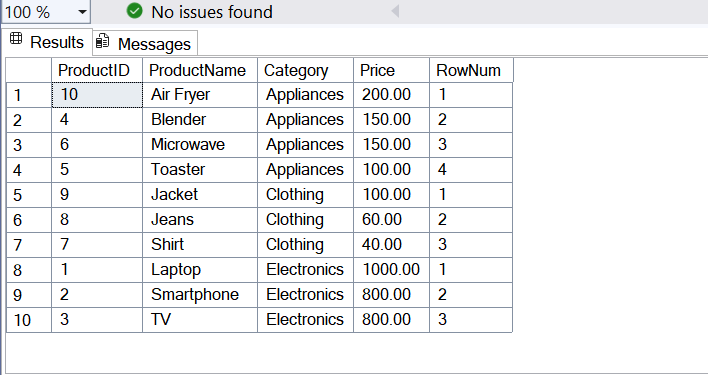
ROW\_NUMBER() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS RowNum

FROM Products;



1. SELECT

ProductID,

ProductName,

Category,

Price,

RANK() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS RankNum,

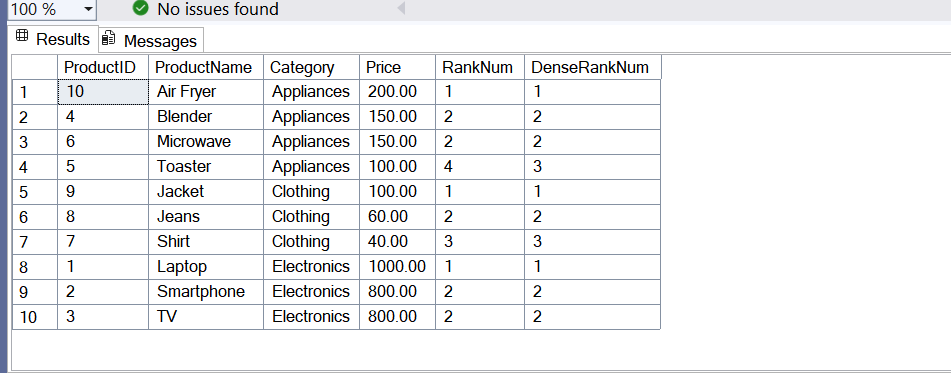
DENSE\_RANK() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS DenseRankNum

FROM Products;



1. WITH Ranked AS (

SELECT \*,

ROW\_NUMBER() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS RowNum

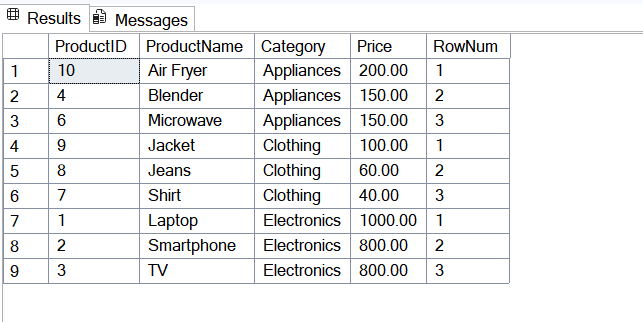
FROM Products

)

SELECT \*

FROM Ranked

WHERE RowNum <= 3;



WITH Ranked AS (

SELECT \*,

RANK() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS RankNum

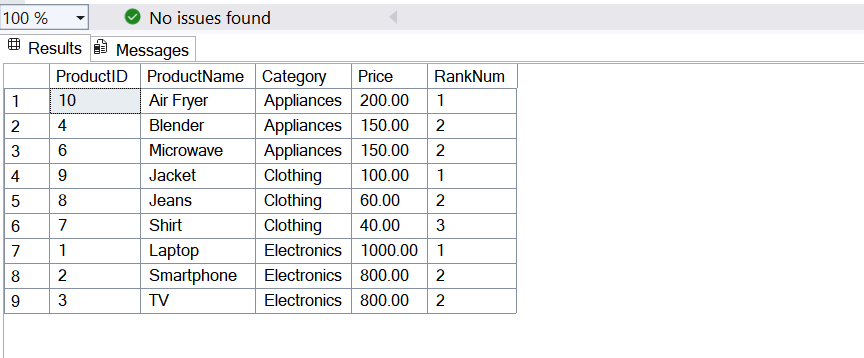
FROM Products

)

SELECT \*

FROM Ranked

WHERE RankNum <= 3;



WITH Ranked AS (

SELECT \*,

DENSE\_RANK() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS DenseRankNum

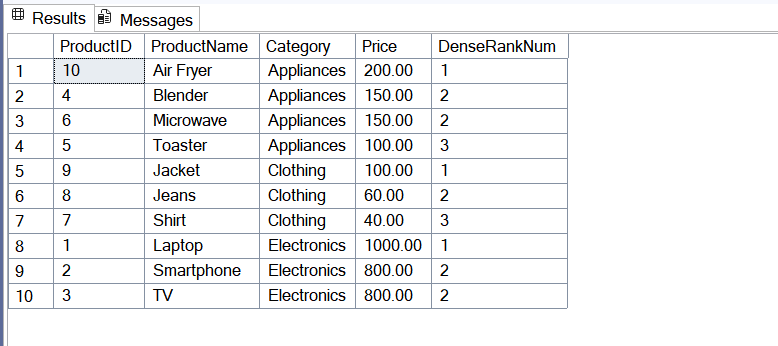
FROM Products

)

SELECT \*

FROM Ranked

WHERE DenseRankNum <= 3;



**Exercise 2: Create a Stored Procedure**

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

**Steps:**

**1. Define the stored procedure with a parameter for DepartmentID.**

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

**3. Create a stored procedure named `sp\_InsertEmployee` with the following 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;**

Solution:  
  
USE EmployeeDB;

GO

-- Create Departments table

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

-- Create Employees table

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

);

-- Insert Departments

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

-- Insert Employees

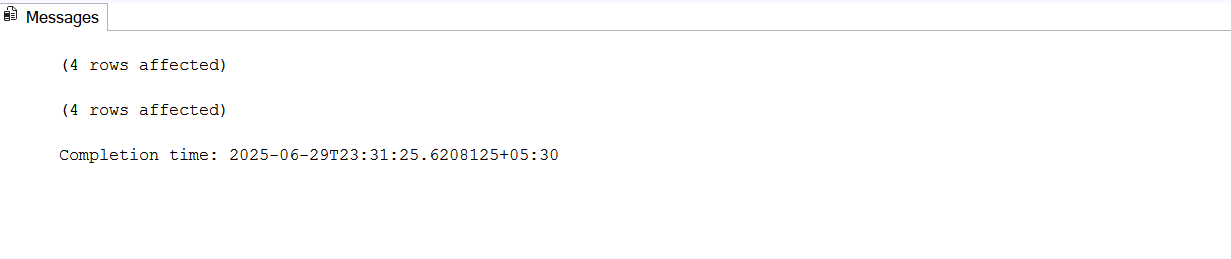
INSERT INTO Employees (FirstName, LastName, DepartmentID, Salary, JoinDate) VALUES

('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');



-- Stored Procedure to fetch employees by department

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

EmployeeID,

FirstName,

LastName,

Salary,

JoinDate

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

-- Stored Procedure to insert a new employee

CREATE PROCEDURE sp\_InsertEmployee

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@DepartmentID INT,

@Salary DECIMAL(10,2),

@JoinDate DATE

AS

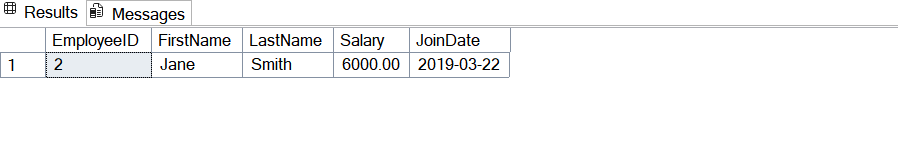
BEGIN

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

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

END;

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 2;



EXEC sp\_InsertEmployee

@FirstName = 'Avilasha',

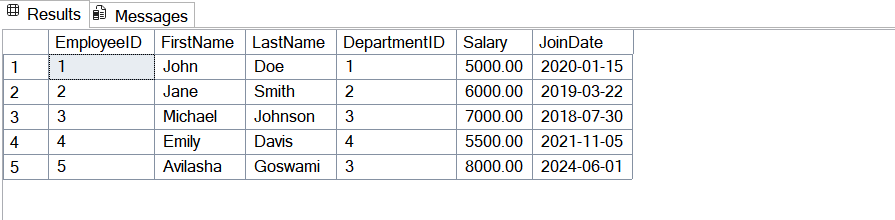
@LastName = 'Goswami',

@DepartmentID = 3,

@Salary = 8000.00,

@JoinDate = '2024-06-01';

SELECT \* FROM Employees;



**Exercise 3: Return Data from a Stored Procedure**

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

**department.**

**Steps:**

**1. Define the stored procedure with a parameter for DepartmentID.**

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

**3. Save the stored procedure by executing the Stored procedure content.**

Solution:

-- Stored Procedure to count employees in a department

CREATE PROCEDURE sp\_CountEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

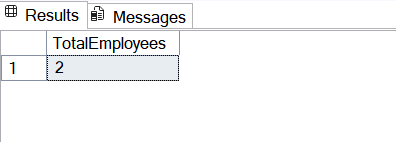
COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

EXEC sp\_CountEmployeesByDepartment @DepartmentID = 3;



**MOQ HANDSON**

**Testable Code With Moq**

using Castle.Core.Smtp;

using CustomerCommLib;

using Moq;

using NUnit.Framework;

namespace CustomerCommLib.Tests

{

public class CustomerCommTests

{

[Test]

public void SendMailToCustomer\_ShouldReturnTrue\_WhenMailIsSent()

{

// Arrange

var mockMailSender = new Mock<IMailSender>();

mockMailSender.Setup(m => m.SendMail(It.IsAny<string>(), It.IsAny<string>())).Returns(true);

var customerComm = new CustomerComm(mockMailSender.Object);

// Act

bool result = customerComm.SendMailToCustomer();

// Assert

Assert.IsTrue(result);

}

}

}

namespace CustomerCommLib

{

public class CustomerComm

{

IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

string to = "cust123@abc.com";

string message = "Some Message";

return \_mailSender.SendMail(to, message);

}

}

}

using System.Net;

using System.Net.Mail;

namespace CustomerCommLib

{

public class MailSender : IMailSender

{

public bool SendMail(string toAddress, string message)

{

try

{

MailMessage mail = new MailMessage();

SmtpClient smtpServer = new SmtpClient("smtp.gmail.com");

mail.From = new MailAddress("your\_email@gmail.com");

mail.To.Add(toAddress);

mail.Subject = "Test Mail";

mail.Body = message;

smtpServer.Port = 587;

smtpServer.Credentials = new NetworkCredential("username", "password");

smtpServer.EnableSsl = true;

smtpServer.Send(mail);

return true;

}

catch

{

return false;

}

}

}

public interface IMailSender

{

}

}

