# Advanced SQL Server Exercise Solutions

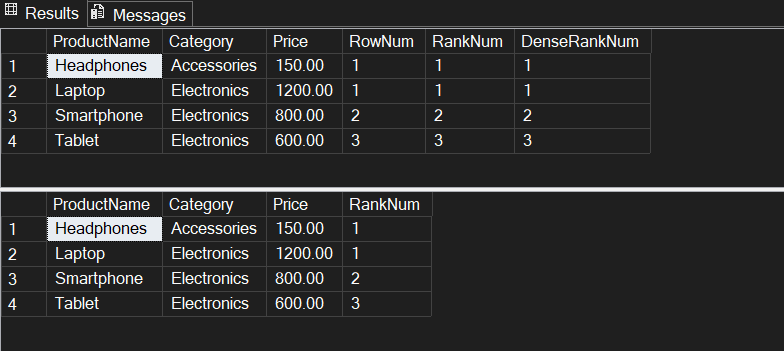
## Advanced SQL Exercises

### Exercise 1: Ranking and Window Functions

*Code:*

|  |
| --- |
| SELECT  ProductName,  Category,  Price,  ROW\_NUMBER() OVER(PARTITION BY Category ORDER BY Price DESC) AS RowNum,  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;      with RankedProducts as(select ProductName, Category, Price, DENSE\_RANK() over (partition by Category order by Price DESC) as RankNum from Products)  select ProductName, Category, Price, RankNum from RankedProducts where RankNum <= 3; |

*Output*



## Stored Procedure

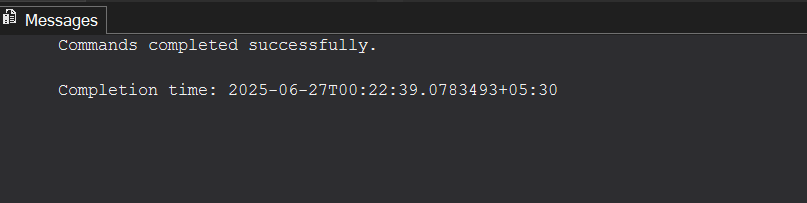
### Exercise 1: Create a Stored Procedure

*Q1) Define the stored procedure with a parameter for DepartmentID*

*Code:*

|  |
| --- |
| CREATE PROCEDURE sp\_GetEmployeesByDepartment  @DepartmentID INT  AS  BEGIN  SET NOCOUNT ON;  output    SELECT  EmployeeID,  FirstName,  LastName,  Salary,  JoinDate  FROM  Employees  WHERE  DepartmentID = @DepartmentID;  END  GO |

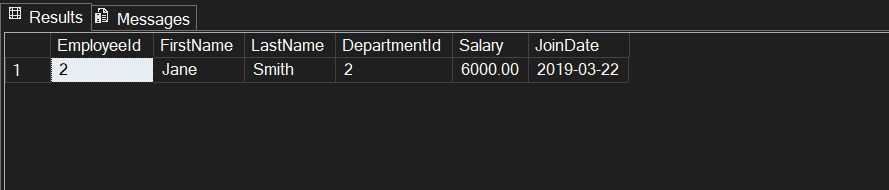
*Output*



*Q2) Write the SQL query to select employee details based on the DepartmentID.*

|  |
| --- |
| *select \* from Employees where DepartmentId=2;* |

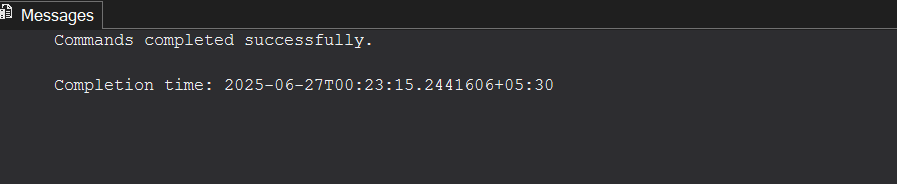
*Output*



*Q3) Creating 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  SET NOCOUNT ON;    INSERT INTO Employees (FirstName, LastName, DepartmentID, Salary, JoinDate)  VALUES (@FirstName, @LastName, @DepartmentID, @Salary, @JoinDate);  END  GO |

*Output*



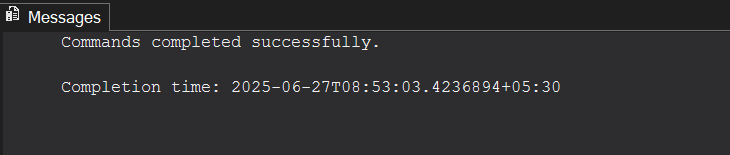
### Exercise 5: Return Data from a Stored Procedure

*1) Create Stored Procedure to Count Employees by Department with Output Parameter*

*Code*

|  |
| --- |
| create procedure sp\_GetEmployeeCountByDepartment  @DepartmentId int,  @EmployeeCount int output  as  begin  set nocount on;    select @EmployeeCount = count(\*)  from Employees  where DepartmentId=@DepartmentId;  end  go |

*Output*

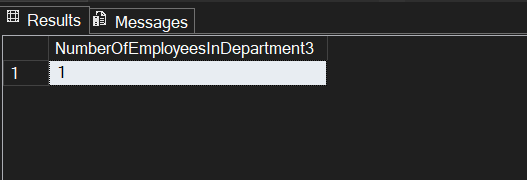


*2) Save the stored procedure by executing the Stored procedure content.*

*Code*

|  |
| --- |
| declare @TotalEmployees int;    exec sp\_GetEmployeeCountByDepartment  @DepartmentId = 3,  @EmployeeCount = @TotalEmployees output;    select @TotalEmployees as NumberOfEmployeesInDepartment3;  go |

*Output*



## Additional important hands-on

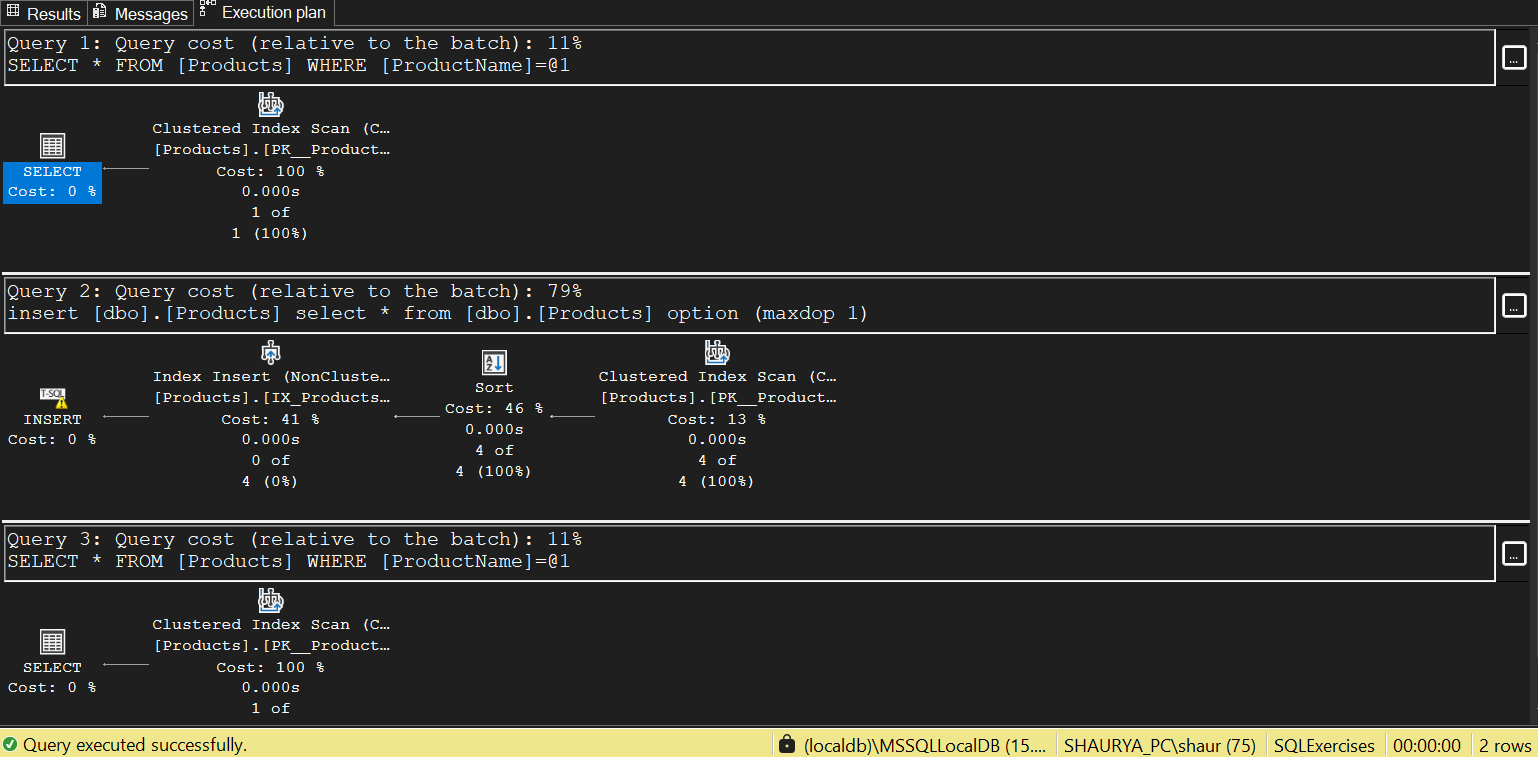
## SQL – Advanced Concepts

### Exercise – 1

*Code*

|  |
| --- |
| Select \* from Products where ProductName=’Laptop’; //before CREATE NONCLUSTERED INDEX IX\_Products\_ProductName ON Products (ProductName); Select \* from Products where ProductName=’Laptop’; //after |

*Output*

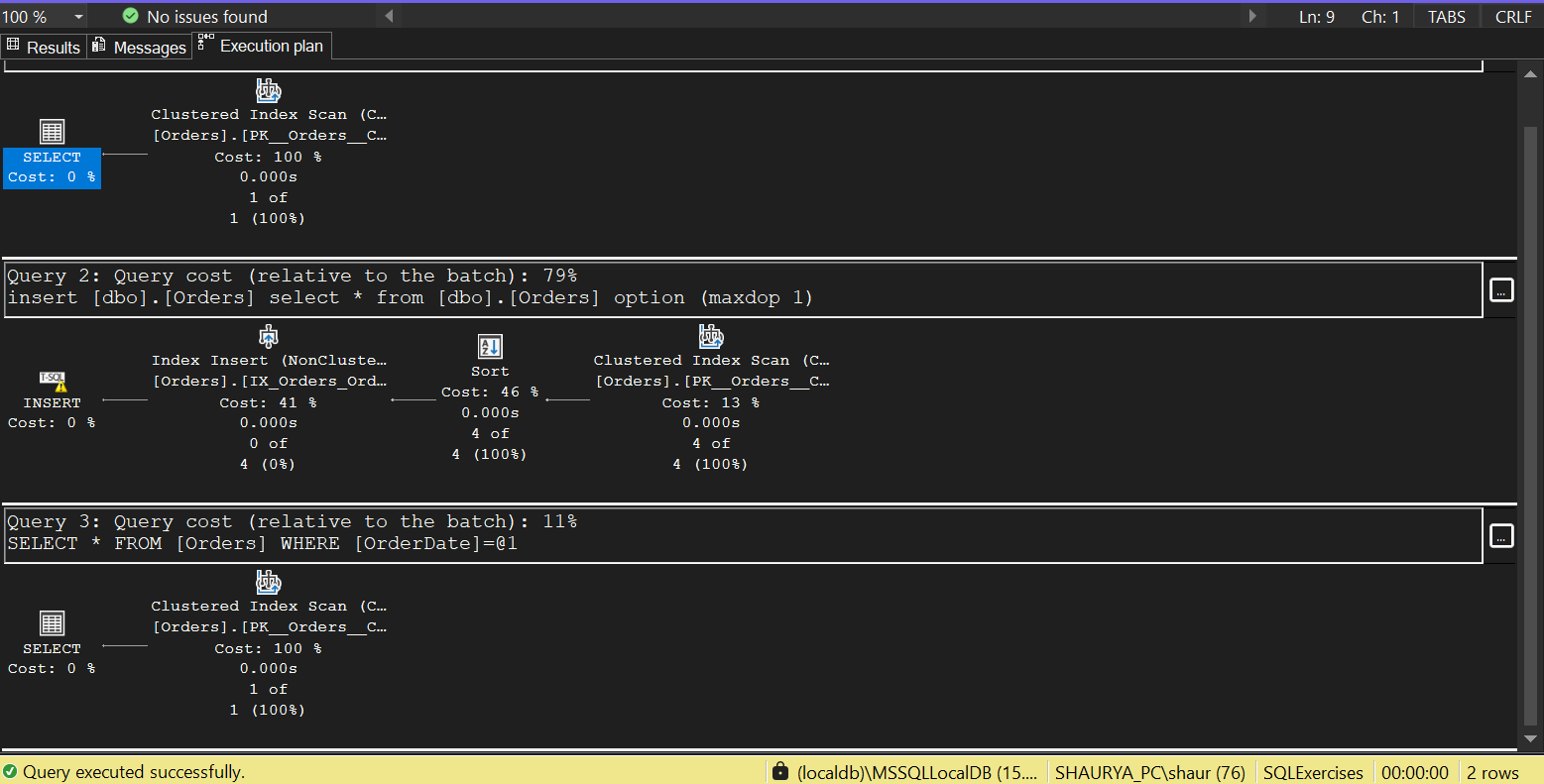


### Exercise – 2

*Code*

|  |
| --- |
| SELECT \* FROM Orders WHERE OrderDate = '2023-01-15';  CREATE NONCLUSTERED INDEX IX\_Orders\_OrderDate ON Orders(OrderDate);  SELECT \* FROM Orders WHERE OrderDate = '2023-01-15'; |

*Output*

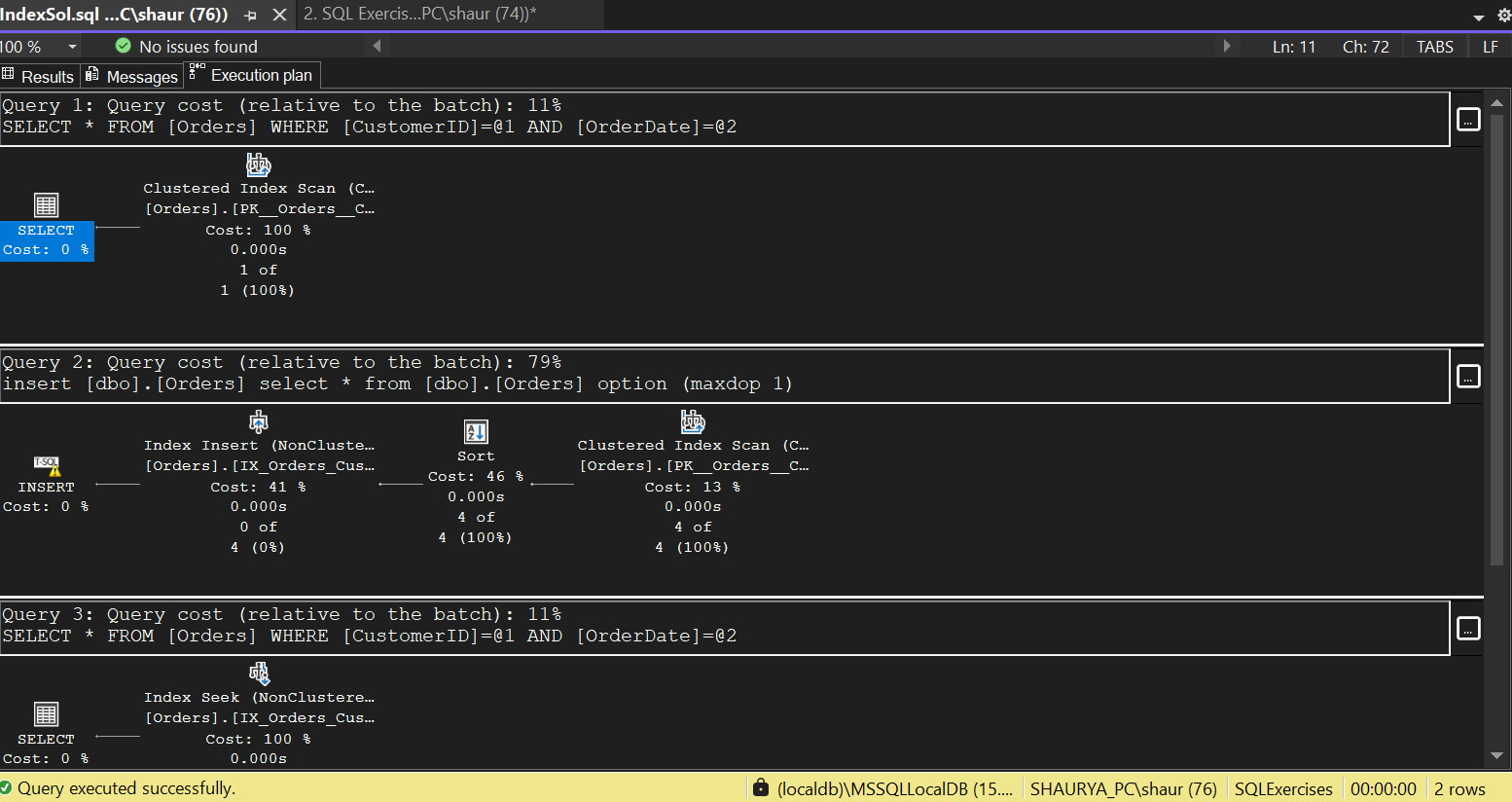


### Exercise – 3

*Code*

|  |
| --- |
| SELECT \* FROM Orders WHERE CustomerID = 1 AND OrderDate = '2023-01-15';  CREATE NONCLUSTERED INDEX IX\_Orders\_CustomerID\_OrderDate ON Orders(CustomerID, OrderDate);  SELECT \* FROM Orders WHERE CustomerID = 1 AND OrderDate = '2023-01-15'; |

*Output*



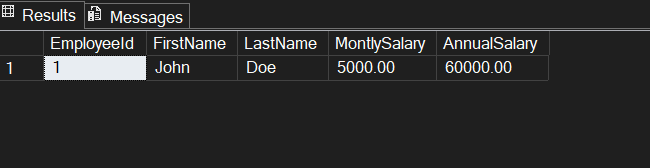
## SQL – Functions

### Exercise 7: Return Data from Scalar Functions

*Code*

|  |
| --- |
| select  E.EmployeeId,  E.FirstName,  E.LastName,  E.Salary as MontlySalary,  dbo.fn\_CalculateAnnualSalary(E.Salary) as AnnualSalary  from  Employees as E  where  E.EmployeeId=1;  go |

Output



## SQL – Stored Procedure

## Exercise 4: Execute a Stored Procedure

*Q1) Write the SQL command to execute the stored procedure with a DepartmentID parameter.*

|  |
| --- |
| *EXEC sp\_GetEmployeesByDepartment @DepartmentID = 3;* |

*Output*

