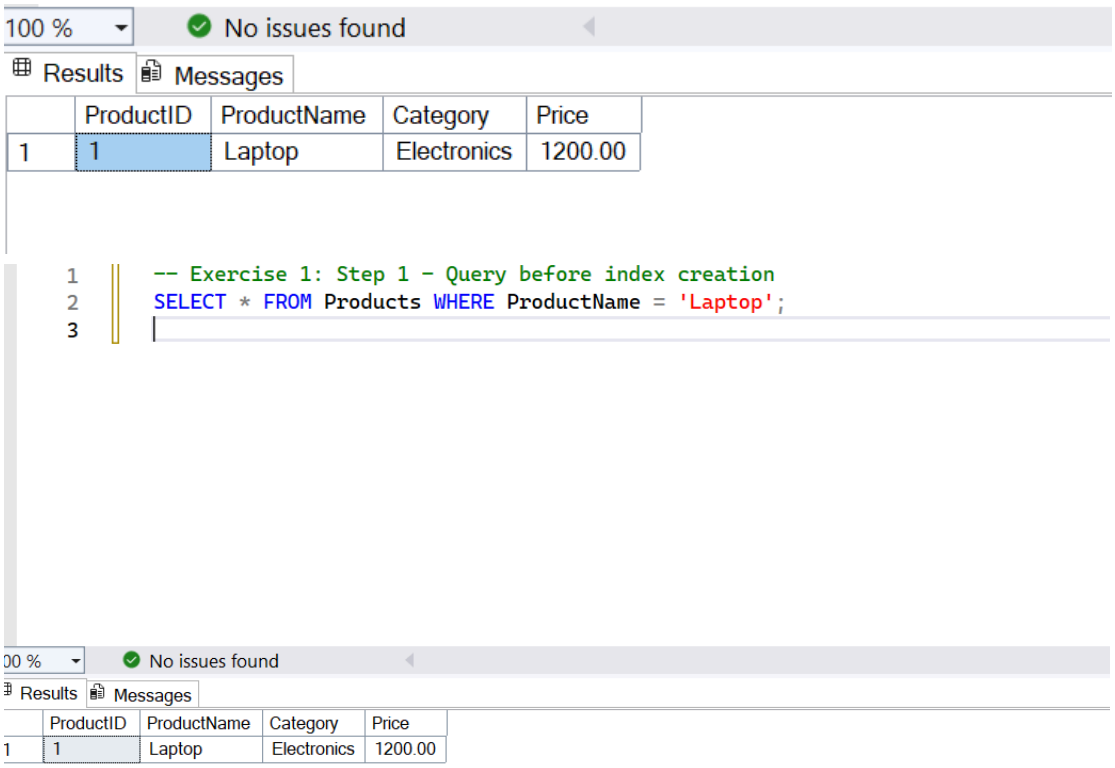
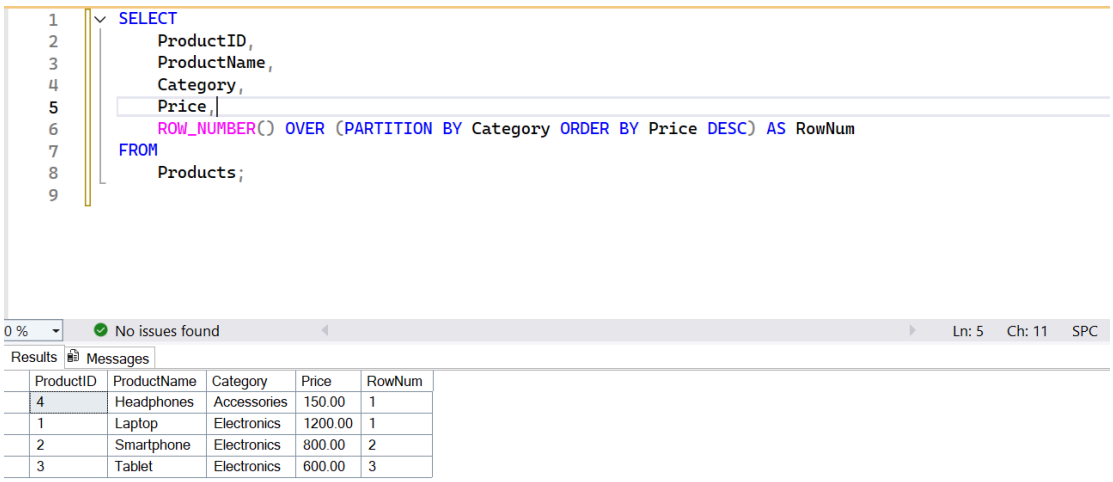
Exercise - 1

```sql
USE anandita db;

-- Enable statistics to see execution time and IO
SET STATISTICS TIME ON;
SET STATISTICS IO ON;

-- Step 1: Query BEFORE index creation
PRINT 'Query execution BEFORE creating index:';
SELECT * FROM Products WHERE ProductName = 'Laptop';

-- Step 2: Create the non-clustered index
PRINT 'Creating non-clustered index...';
CREATE NONCLUSTERED INDEX IX_Products_ProductName
ON Products (ProductName);

-- Step 3: Query AFTER index creation
PRINT 'Query execution AFTER creating index:';
SELECT * FROM Products WHERE ProductName = 'Laptop';

-- Turn off statistics
SET STATISTICS TIME OFF;
SET STATISTICS IO OFF;
```



```sql
SELECT
    ProductID,
    ProductName,
    Category,
    Price,
    ROW_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS
RowNum
FROM
    Products;
```



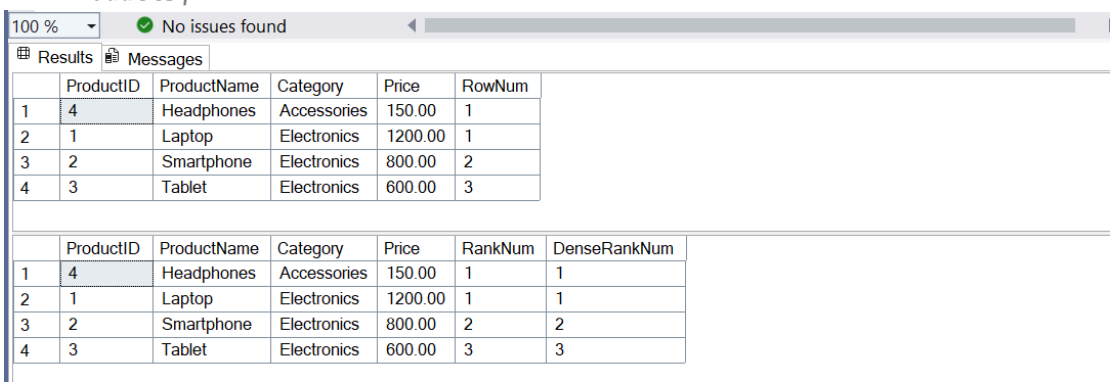
```
SELECT
    ProductID,
    ProductName,
    Category,
    Price,
    ROW_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum
FROM
    Products;
```

0 % — No issues found — Ln: 5 Ch: 11 SPC

Results | Messages

ProductID	ProductName	Category	Price	RowNum
4	Headphones	Accessories	150.00	1
1	Laptop	Electronics	1200.00	1
2	Smartphone	Electronics	800.00	2
3	Tablet	Electronics	600.00	3

```sql
    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;
```



100 % — No issues found

Results | Messages

	ProductID	ProductName	Category	Price	RowNum
1	4	Headphones	Accessories	150.00	1
2	1	Laptop	Electronics	1200.00	1
3	2	Smartphone	Electronics	800.00	2
4	3	Tablet	Electronics	600.00	3

	ProductID	ProductName	Category	Price	RankNum	DenseRankNum
1	4	Headphones	Accessories	150.00	1	1
2	1	Laptop	Electronics	1200.00	1	1
3	2	Smartphone	Electronics	800.00	2	2
4	3	Tablet	Electronics	600.00	3	3

```sql
    SELECT
    ProductID,
    ProductName,
    Category,
    Price
FROM (
```

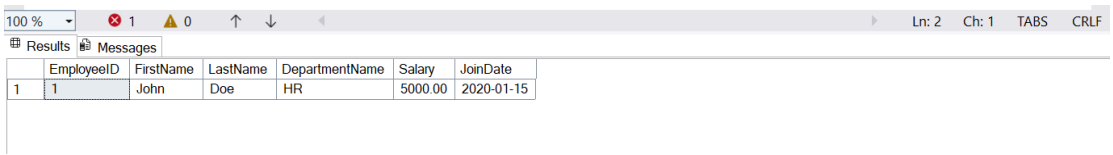
```
    SELECT
        ProductID,
        ProductName,
        Category,
        Price,
        ROW_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS
RowNum
    FROM
        Products
) AS Ranked
WHERE RowNum <= 3;
```

	ProductID	ProductName	Category	Price
1	4	Headphones	Accessories	150.00
2	1	Laptop	Electronics	1200.00
3	2	Smartphone	Electronics	800.00
4	3	Tablet	Electronics	600.00

Query executed successfully.

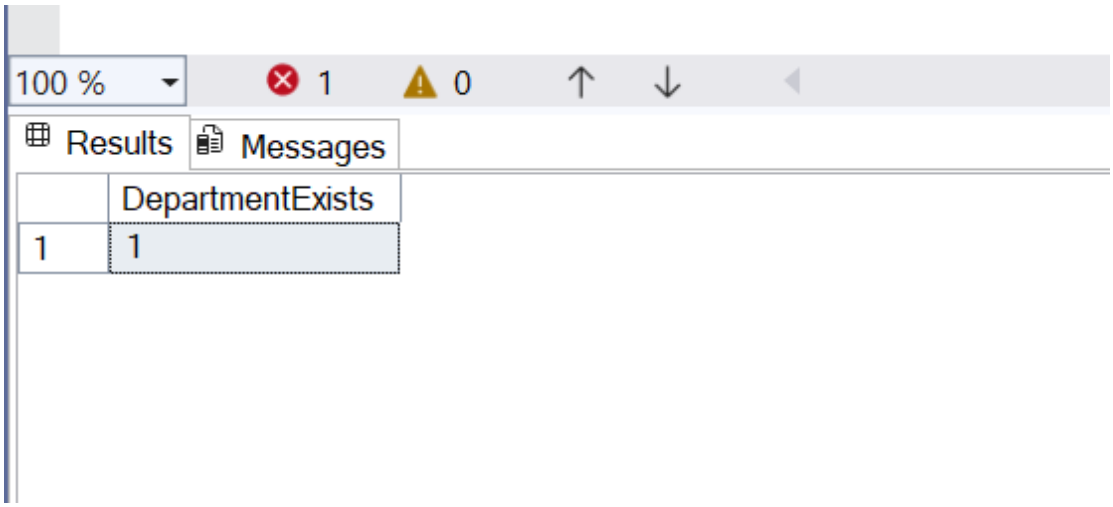
Exercidse 2:

```
-- Insert sample data into Departments
INSERT INTO Departments (DepartmentID, DepartmentName) VALUES
(1, 'HR'),
(2, 'Finance'),
(3, 'IT'),
(4, 'Marketing');

-- Insert sample data into Employees
INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID,
Salary, JoinDate) VALUES
(1, 'John', 'Doe', 1, 5000.00, '2020-01-15'),
(2, 'Jane', 'Smith', 2, 6000.00, '2019-03-22'),
(3, 'Michael', 'Johnson', 3, 7000.00, '2018-07-30'),
(4, 'Emily', 'Davis', 4, 5500.00, '2021-11-05');

DROP PROCEDURE IF EXISTS dbo.sp_GetEmployeesByDepartment;
GO

CREATE PROCEDURE dbo.sp_GetEmployeesByDepartment
    @DepartmentID INT
AS
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 = @DepartmentID;
END;
GO


EXEC dbo.sp_GetEmployeesByDepartment @DepartmentID = 1;
```



Exercise 3:

```
CREATE PROCEDURE dbo.sp_CheckDepartment
    @DepartmentID INT
AS
BEGIN
    IF EXISTS (SELECT 1 FROM Departments WHERE DepartmentID =
@DepartmentID)
        RETURN 1; -- Department exists
    ELSE
        RETURN 0; -- Department does not exist
END;
```




```
EXEC dbo.sp_GetEmployeesByDepartment @DepartmentID = 1;
```