**1. SQL Exercise - Advanced concepts**

**Exercise 1: Ranking and Window Functions**

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10, 2)

);

GO

INSERT INTO Products (ProductID, ProductName, Category, Price)

VALUES

(1, 'Wireless Mouse', 'Electronics', 25.99),

(2, 'Bluetooth Speaker', 'Electronics', 55.49),

(3, 'Laptop Stand', 'Electronics', 55.49),

(4, 'Office Chair', 'Furniture', 199.99),

(5, 'Desk Lamp', 'Furniture', 39.99),

(6, 'Bookshelf', 'Furniture', 199.99),

(7, 'Notebook', 'Stationery', 4.99),

(8, 'Pen Set', 'Stationery', 3.49),

(9, 'Planner', 'Stationery', 4.99);

GO

SELECT

Category,

ProductName,

Price,

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

FROM Products;

GO

SELECT

Category,

ProductName,

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;

GO

WITH RankedProducts AS (

SELECT

Category,

ProductName,

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

)

SELECT \*

FROM RankedProducts

ORDER BY Category, Price DESC;

GO

WITH RankedProducts AS (

SELECT

Category,

ProductName,

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

)

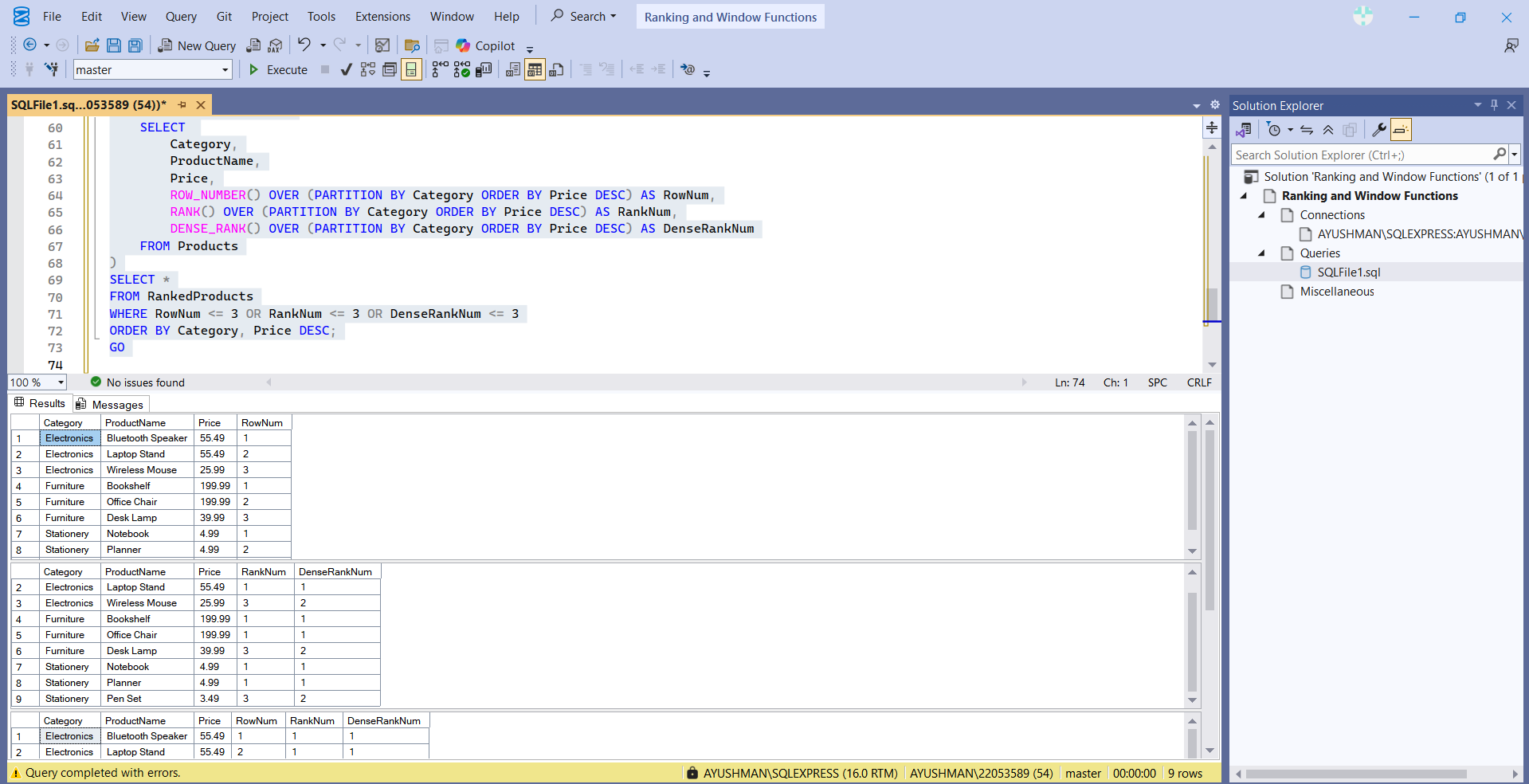
SELECT \*

FROM RankedProducts

WHERE RowNum <= 3 OR RankNum <= 3 OR DenseRankNum <= 3

ORDER BY Category, Price DESC;

GO



**4. SQL Exercise - Stored procedure**

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

GO

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10, 2),

JoinDate DATE

);

GO

INSERT INTO Departments (DepartmentID, DepartmentName)

VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

GO

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');

GO

SELECT \* FROM Departments;

GO

SELECT \* FROM Employees;

GO

**Exercise 1: Create a Stored Procedure**

CREATE PROCEDURE 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

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;

GO

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 2;

EXEC sp\_InsertEmployee

@FirstName = 'Ava',

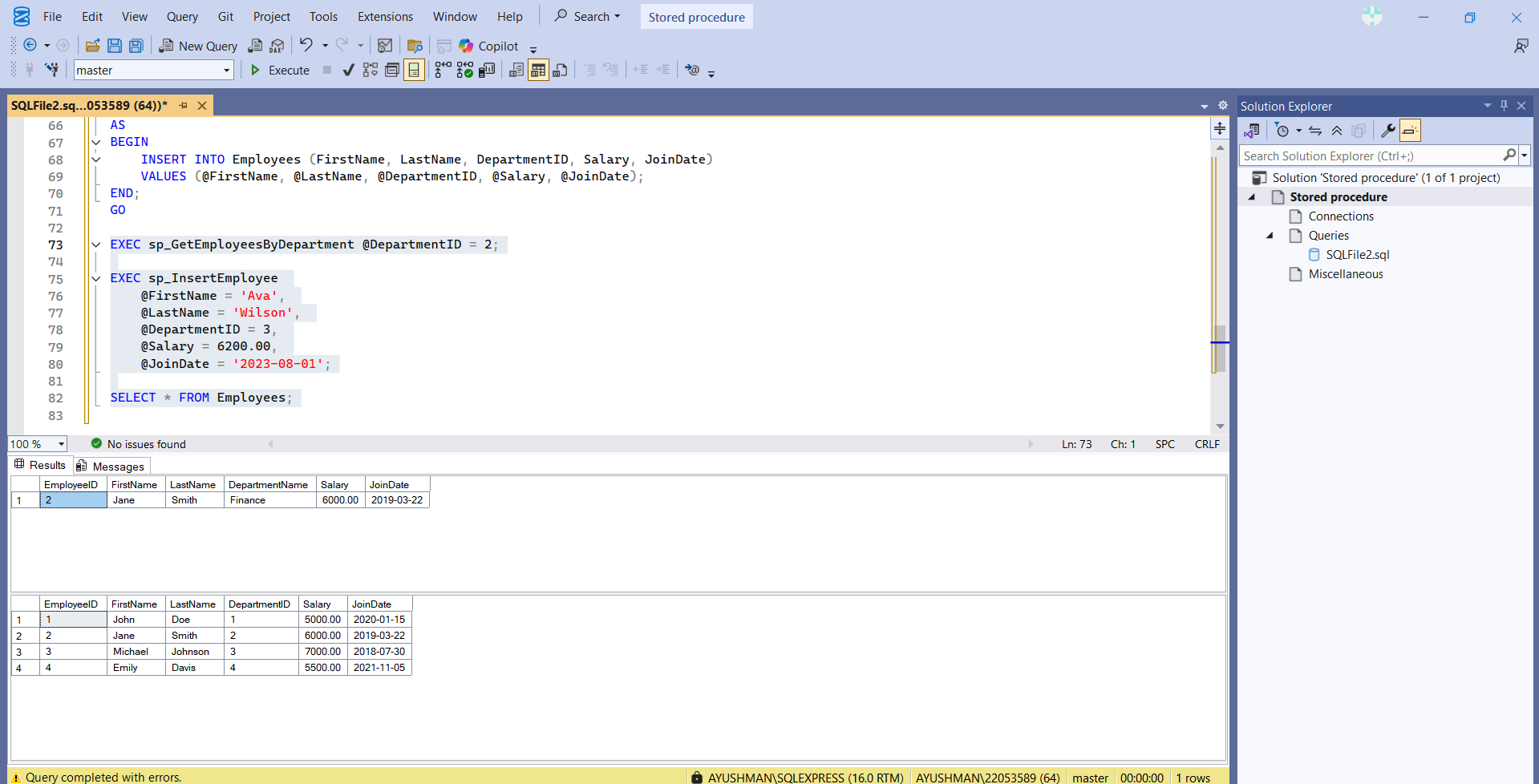
@LastName = 'Wilson',

@DepartmentID = 3,

@Salary = 6200.00,

@JoinDate = '2023-08-01';

SELECT \* FROM Employees;



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

IF OBJECT\_ID('sp\_GetEmployeeCountByDepartment', 'P') IS NOT NULL

DROP PROCEDURE sp\_GetEmployeeCountByDepartment;

GO

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

D.DepartmentName,

COUNT(E.EmployeeID) AS TotalEmployees

FROM Departments D

LEFT JOIN Employees E ON D.DepartmentID = E.DepartmentID

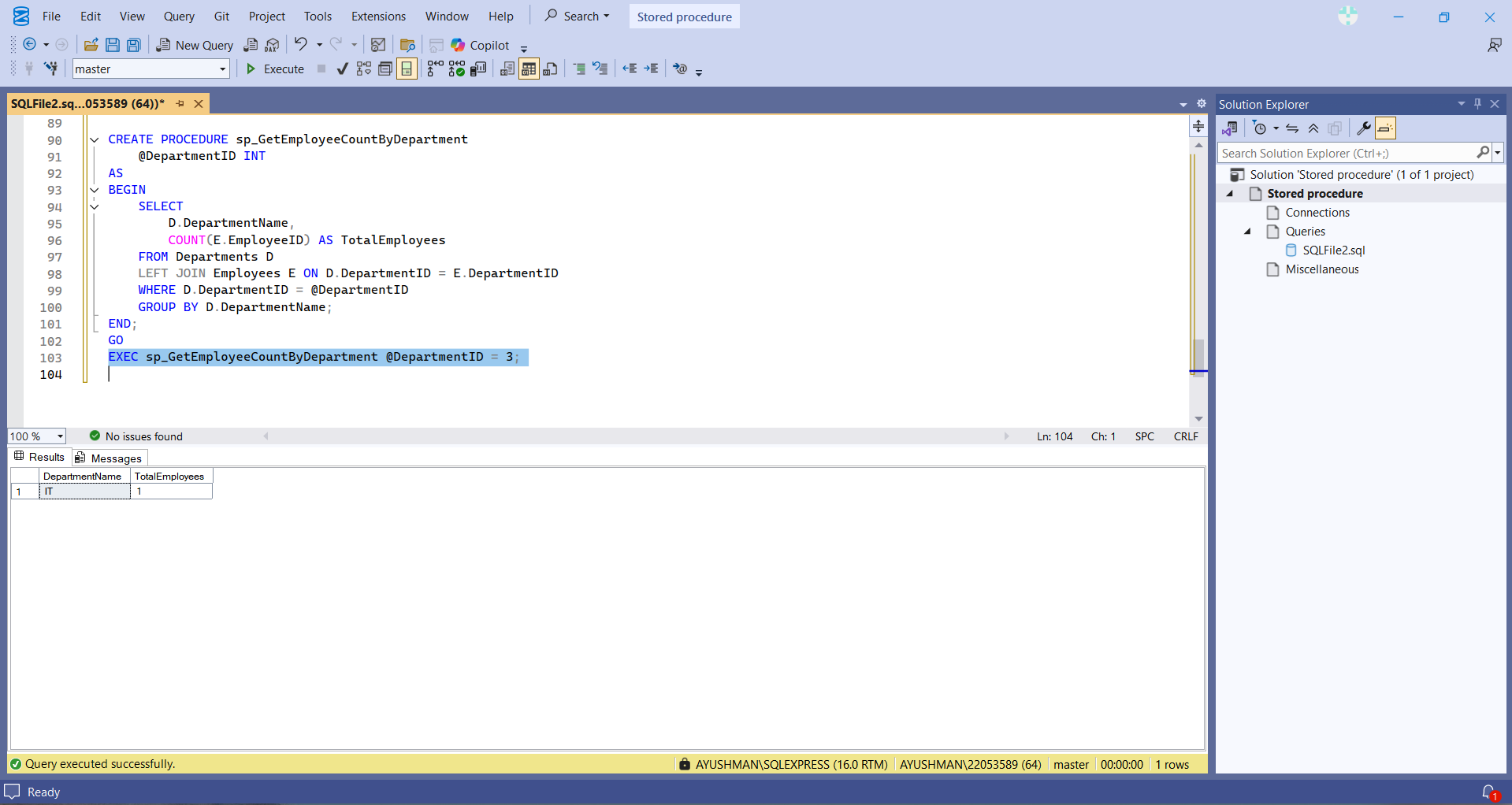
WHERE D.DepartmentID = @DepartmentID

GROUP BY D.DepartmentName;

END;

GO

EXEC sp\_GetEmployeeCountByDepartment @DepartmentID = 3;



**Exercise 4: Execute a Stored Procedure**

CREATE PROCEDURE 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;

**5. SQL Exercise – Functions**

**Exercise 7: Return Data from a Scalar Function**

IF OBJECT\_ID('dbo.GetYearlySalary', 'FN') IS NOT NULL

DROP FUNCTION dbo.GetYearlySalary;

GO

CREATE FUNCTION dbo.GetYearlySalary (@MonthlySalary INT)

RETURNS INT

AS

BEGIN

RETURN @MonthlySalary \* 12;

END;

GO

-- Test Function

SELECT dbo.GetYearlySalary(50000) AS YearlySalary;

-- Use in Table

SELECT

Name,

Salary AS MonthlySalary,

dbo.GetYearlySalary(Salary) AS AnnualSalary

FROM Employees;

**2. SQL Exercise – Index**

SELECT \* FROM Products WHERE ProductName = 'Laptop';

CREATE NONCLUSTERED INDEX IX\_Products\_ProductName

ON Products (ProductName);

SELECT \* FROM Products WHERE ProductName = 'Laptop';

SELECT \* FROM Orders WHERE OrderDate = '2023-01-15';

ALTER TABLE Orders DROP CONSTRAINT PK\_\_Orders; -- Replace with actual constraint name if different

ALTER TABLE Orders ADD CONSTRAINT PK\_Orders\_OrderID PRIMARY KEY NONCLUSTERED (OrderID);

CREATE CLUSTERED INDEX IX\_Orders\_OrderDate

ON Orders (OrderDate);

SELECT \* FROM Orders WHERE OrderDate = '2023-01-15';

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';