EmployeeApp

Program.cs 9

Program.cs > ...

```
using System;
using System.Data;
using System.Data.SqlClient;

0 references
class Program
{

    2 references
    private static string connectionString = "Server=localhost;Database=EmployeeDB;Trusted_Connection=True;";

    0 references
    static void Main(string[] args)
    {

        InsertEmployee("Ayush", "Raj", 3, 7500.00m, DateTime.Now);


        GetEmployeesByDepartment(3);
    }

    1 reference
    public static void InsertEmployee(string firstName, string lastName, int departmentId, decimal salary, DateTime joinDate)
    {
        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand("sp_InsertEmployee", conn);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@FirstName", firstName);
            cmd.Parameters.AddWithValue("@LastName", lastName);
            cmd.Parameters.AddWithValue("@DepartmentID", departmentId);
            cmd.Parameters.AddWithValue("@Salary", salary);
            cmd.Parameters.AddWithValue("@JoinDate", joinDate);

            conn.Open();
            cmd.ExecuteNonQuery();
            conn.Close();

            Console.WriteLine("✅ Employee inserted successfully.");
        }
    }

    1 reference
    public static void GetEmployeesByDepartment(int departmentId)
    {
        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand("sp_GetEmployeesByDepartment", conn);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@DepartmentID", departmentId);

            conn.Open();
            SqlDataReader reader = cmd.ExecuteReader();

            Console.WriteLine("📋 Employees in Department " + departmentId + ":");
            while (reader.Read())
            {
                Console.WriteLine($"ID: {reader["EmployeeID"]}, Name: {reader["FirstName"]} {reader["LastName"]}, Salary: {reader["Salary"]}, Joined: {reader["JoinDate"]}");
            }

            reader.Close();
            conn.Close();
        }
    }
}
```

OUTPUT:

```
ta.SqlClient package instead.'
✅ Employee inserted successfully.
📋 Employees in Department 3:
nt package instead.'
✅ Employee inserted successfully.
📋 Employees in Department 3:
📋 Employees in Department 3:
ID: 1, Name: Ayush Raj, Salary: 7500.00, Joined: 24-06-2025 00:00:00
PS C:\Users\Lenovo\OneDrive\Desktop\Cognizant\week2\SQL\EmployeeApp>
```

Moq

MailSender.cs | CustomerComm.cs | CustomerCommTests.cs CustomerComm.Tests | CustomerComm.Tests.csproj.nuget.g.props | CustomerComm.Tests | CustomerCommTests.cs CustomerCommLib

CustomerComm.Tests > obj > CustomerComm.Tests.csproj.nuget.g.props

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<Project ToolsVersion="14.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup Condition=" '$(ExcludeRestorePackageImports)' != 'true' ">
    <RestoreSuccess Condition=" '$(RestoreSuccess)' == '' ">True</RestoreSuccess>
    <RestoreTool Condition=" '$(RestoreTool)' == '' ">NuGet</RestoreTool>
    <ProjectAssetsFile Condition=" '$(ProjectAssetsFile)' == '' ">$(MSBuildThisFileDirectory)project.assets.json</ProjectAssetsFile>
    <NuGetPackageRoot Condition=" '$(NuGetPackageRoot)' == '' ">$(UserProfile)\.nuget\packages\</NuGetPackageRoot>
    <NuGetPackageFolders Condition=" '$(NuGetPackageFolders)' == '' ">C:\Users\Lenovo\.nuget\packages\</NuGetPackageFolders>
    <NuGetProjectStyle Condition=" '$(NuGetProjectStyle)' == '' ">PackageReference</NuGetProjectStyle>
    <NuGetToolVersion Condition=" '$(NuGetToolVersion)' == '' ">6.14.0</NuGetToolVersion>
  </PropertyGroup>
  <ItemGroup Condition=" '$(ExcludeRestorePackageImports)' != 'true' ">
    <SourceRoot Include="C:\Users\Lenovo\.nuget\packages\" />
  </ItemGroup>
  <ImportGroup Condition=" '$(ExcludeRestorePackageImports)' != 'true' ">
    <Import Project="$(NuGetPackageRoot)microsoft.testing.platform\1.5.3\buildTransitive\net9.0\Microsoft.Testing.Platform.props" Condition="Exists('$(NuGetPackageRoot)m
    <Import Project="$(NuGetPackageRoot)microsoft.testing.platform.msbuild\1.5.3\buildTransitive\Microsoft.Testing.Platform.MSBuild.props" Condition="Exists('$(NuGetPack
    <Import Project="$(NuGetPackageRoot)microsoft.testing.extensions.telemetry\1.5.3\buildTransitive\net9.0\Microsoft.Testing.Extensions.Telemetry.props" Condition="Exis
    <Import Project="$(NuGetPackageRoot)nunit3testadapter\5.0.0\build\netcoreapp3.1\NUnit3TestAdapter.props" Condition="Exists('$(NuGetPackageRoot)nunit3testadapter\5.0.
    <Import Project="$(NuGetPackageRoot)nunit\4.3.2\build\NUnit.props" Condition="Exists('$(NuGetPackageRoot)nunit\4.3.2\build\NUnit.props')" />
  </ImportGroup>
</Project>
```

CustomerComm.Tests > CustomerComm.Tests.csproj

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net9.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Moq" Version="4.20.72" />
    <PackageReference Include="NUnit" Version="4.3.2" />
    <PackageReference Include="NUnit3TestAdapter" Version="5.0.0" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\CustomerCommLib\CustomerCommLib.csproj" />
  </ItemGroup>

</Project>

```

CustomerComm.Tests > CustomerCommTests.cs > ...

```
using NUnit.Framework;
using Moq;
using CustomerCommLib;

namespace CustomerCommLib.Tests
{
    [TestFixture]
    0 references
    public class CustomerCommTests
    {
        3 references
        private Mock<IMailSender> _mockMailSender = null!;
    2 references
    private CustomerCommLib.CustomerComm _customerComm = null!;

        [OneTimeSetUp]
        0 references
        public void Setup()
        {
            _mockMailSender = new Mock<IMailSender>();
            _mockMailSender.Setup(x => x.SendMail(It.IsAny<string>(), It.IsAny<string>())).Returns(true);
    _customerComm = new CustomerCommLib.CustomerComm(_mockMailSender.Object);
        }

        [Test]
        0 references
        public void SendMailToCustomer_ShouldReturnTrue()
        {
            bool result = _customerComm.SendMailToCustomer();
    Assert.That(result, Is.True);
        }
    }
}
```

CustomerCommApp > Program.cs > ...

```
using CustomerCommLib;  // <-- Add this if you have a namespace

0 references
class Program
{
    0 references
    static void Main()
    {
        // Call your library method or class here
        Console.WriteLine("Calling CustomerCommLib...");

        // Example: assuming you have a class like `Notifier` with a method `SendEmail`
        // var notifier = new Notifier();
        // notifier.SendEmail("test@example.com", "Subject", "Body");
    }
}
```

CustomerCommLib > CustomerComm.cs > ...

```
namespace CustomerCommLib
{
    1 reference
    public class CustomerComm
    {
        2 references
        IMailSender _mailSender;

        0 references
        public CustomerComm(IMailSender mailSender)
        {
            _mailSender = mailSender;
        }

        0 references
        public bool SendMailToCustomer()
        {
            _mailSender.SendMail("cust123@abc.com", "Some Message");
            return true;
        }
    }
}
```

Output:

```
Passed!  - Failed: 0, Passed: 1, Skipped: 0, Total: 1
```

Nunit

CalcLibrary.deps.json

bin > Debug > net9.0 > {} CalcLibrary.deps.json > ...

```
{
  "runtimeTarget": {
    "name": ".NETCoreApp,Version=v9.0",
    "signature": ""
  },
  "compilationOptions": {},
  "targets": {
    ".NETCoreApp,Version=v9.0": {
      "CalcLibrary/1.0.0": {
        "runtime": {
          "CalcLibrary.dll": {}
        }
      }
    }
  },
  "libraries": {
    "CalcLibrary/1.0.0": {
      "type": "project",
      "serviceable": false,
      "sha512": ""
    }
  }
}
```

CalcLibraryTests.csproj

CalcLibraryTests > CalcLibraryTests.csproj

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net9.0</TargetFramework>
    <LangVersion>latest</LangVersion>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <IsPackable>false</IsPackable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="coverlet.collector" Version="6.0.2" />
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.12.0" />
    <PackageReference Include="NUnit" Version="4.2.2" />
    <PackageReference Include="NUnit.Analyzers" Version="4.4.0" />
    <PackageReference Include="NUnit3TestAdapter" Version="4.6.0" />
  </ItemGroup>

  <ItemGroup>
    <Using Include="NUnit.Framework" />
  </ItemGroup>

</Project>

```

CalcLibrary.GlobalUsings.g.cs

obj > Debug > net9.0 > CalcLibrary.GlobalUsings.g.cs

```
1
2   global using global::System;
3   global using global::System.Collections.Generic;
4   global using global::System.IO;
5   global using global::System.Linq;
6   global using global::System.Net.Http;
7   global using global::System.Threading;
8   global using global::System.Threading.Tasks;
9
```

CalcLibrary.GlobalUsings.g.cs | Calculator.cs

Calculator.cs

```
1   namespace CalcLibrary
2   {
3       public class Calculator
4       {
5           public int Add(int a, int b)
6           {
7               return a + b;
8           }
9       }
10  }
11
```

Output:

```
PS C:\Users\Lenovo\OneDrive\Desktop\CalcLibraryTests> dotnet clean
>> dotnet build
>> dotnet test
>>

Build succeeded in 0.5s
Restore complete (0.4s)
  CalcLibrary succeeded (0.5s) → C:\Users\Lenovo\OneDrive\Desktop\CalcLibrary\CalcLibrary\bin\Debug\net9.0\CalcLibrary.dll
  CalcLibraryTests succeeded (3.2s) → bin\Debug\net9.0\CalcLibraryTests.dll

Build succeeded in 7.0s
Restore complete (0.4s)
  CalcLibrary succeeded (0.2s) → C:\Users\Lenovo\OneDrive\Desktop\CalcLibrary\CalcLibrary\bin\Debug\net9.0\CalcLibrary.dll  CalcLibraryTests succeeded (0.2s) →
bin\Debug\net9.0\CalcLibraryTests.dll
NUnit Adapter 5.0.0.0: Test execution started
Running all tests in C:\Users\Lenovo\OneDrive\Desktop\CalcLibraryTests\bin\Debug\net9.0\CalcLibraryTests.dll
   NUnit3TestExecutor discovered 1 of 1 NUnit test cases using Current Discovery mode, Non-Explicit run
NUnit Adapter 5.0.0.0: Test execution complete
  CalcLibraryTests test succeeded (11.1s)

Test summary: total: 1, failed: 0, succeeded: 1, skipped: 0, duration: 11.1s
Build succeeded in 12.5s
PS C:\Users\Lenovo\OneDrive\Desktop\CalcLibraryTests>
```

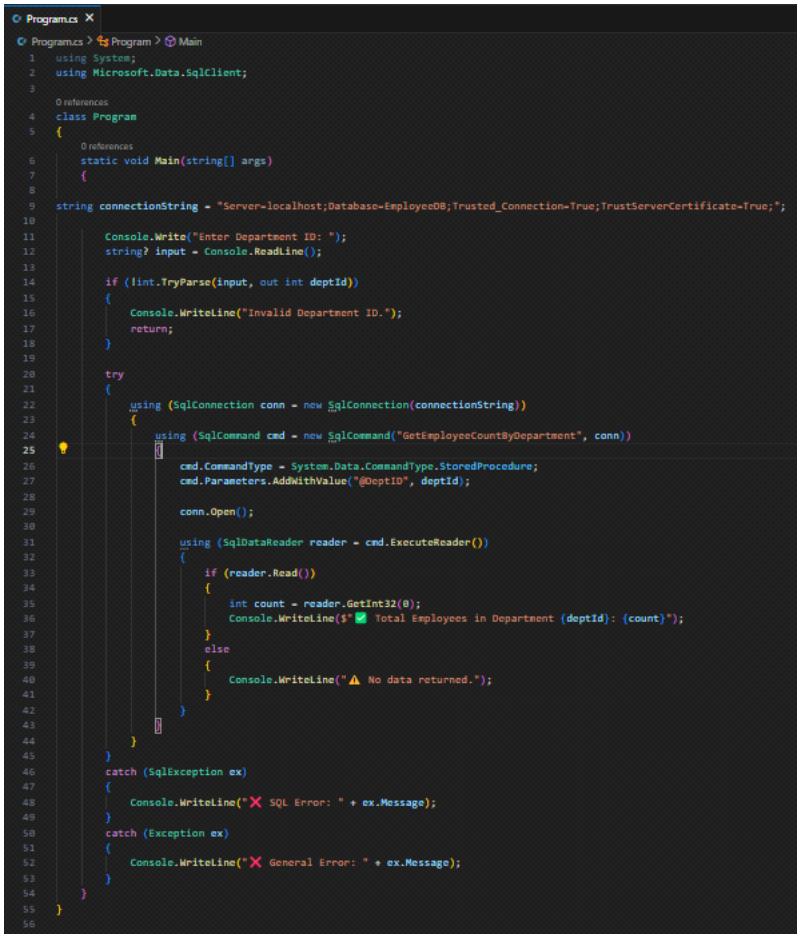
Ranking

```
using System;
using System.Collections.Generic;
using System.Linq;

public class Product
{
    public string Name { get; set; }
    public string Category { get; set; }
    public decimal Price { get; set; }
}

class Program
{
    static void Main()
    {
        var products = new List<Product>
        {
            new Product { Name = "iPhone", Category = "Electronics", Price = 999 },
            new Product { Name = "TV", Category = "Electronics", Price = 799 },
            new Product { Name = "Camera", Category = "Electronics", Price = 799 },
            new Product { Name = "Shirt", Category = "Clothing", Price = 49 },
            new Product { Name = "Jeans", Category = "Clothing", Price = 59 },
            new Product { Name = "Jacket", Category = "Clothing", Price = 99 },
            new Product { Name = "Sweater", Category = "Clothing", Price = 99 }
        };

        Console.WriteLine("=== DENSE_RANK Equivalent ===");
        var denseRankResult = products
            .GroupBy(p => p.Category)
            .SelectMany(g =>
            {
                var ordered = g.OrderByDescending(p => p.Price).ToList();
                int rank = 0;
                decimal? prevPrice = null;
                return ordered.Select(p =>
                {
                    if (prevPrice != p.Price)
                    {
                        rank++;
                        prevPrice = p.Price;
                    }
                    return new
                    {
                        p.Name,
                        p.Category,
                        p.Price,
                        DenseRank = rank
                    };
                });
            })
            .Where(p => p.DenseRank <= 3);

        foreach (var p in denseRankResult)
        {
            Console.WriteLine($"{p.Category,-12} | {p.Name,-10} | ${p.Price,-5} | DenseRank: {p.DenseRank}");
        }
    }
}
```

OUTPUT:

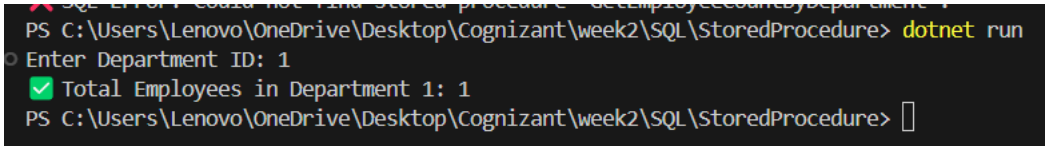
```
PS C:\Users\Lenovo\OneDrive\Desktop\Cognizant\week2\SQL\RankingExample> dotnet run
C:\Users\Lenovo\OneDrive\Desktop\Cognizant\week2\SQL\RankingExample\Program.cs(7,19): warning CS8618: Non-nullable property 'Name' must contain a non-null value
 when exiting constructor. Consider adding the 'required' modifier or declaring the property as nullable.
C:\Users\Lenovo\OneDrive\Desktop\Cognizant\week2\SQL\RankingExample\Program.cs(8,19): warning CS8618: Non-nullable property 'Category' must contain a non-nuly '
Category' must contain a non-null value why 'Categoy 'Category' must contain a non-null value when exiting constructor. Consider adding the y 'Category' must co
y 'Category' must contain a non-null value when exiting constructor. Consider adding the 'required' modifier or declaring the property as nullable.
=== DENSE_RANK Equivalent ===
Electronics  | iPhone     | $999   | DenseRank: 1
Electronics  | TV         | $799   | DenseRank: 2
Electronics  | Camera     | $799   | DenseRank: 2
Clothing     | Jacket     | $99    | DenseRank: 1
Clothing     | Sweater    | $99    | DenseRank: 1
Clothing     | Jeans      | $59    | DenseRank: 2
Clothing     | Shirt      | $49    | DenseRank: 3
PS C:\Users\Lenovo\OneDrive\Desktop\Cognizant\week2\SQL\RankingExample>
```

Stored Procedure



```
using System;
using Microsoft.Data.SqlClient;

class Program
{
    static void Main(string[] args)
    {

string connectionString = "Server=localhost;Database=EmployeeDB;Trusted_Connection=True;TrustServerCertificate=True;";

        Console.Write("Enter Department ID: ");
        string? input = Console.ReadLine();

        if (!int.TryParse(input, out int deptId))
        {
            Console.WriteLine("Invalid Department ID.");
            return;
        }

        try
        {
            using (SqlConnection conn = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand("GetEmployeeCountByDepartment", conn))
                {
                    cmd.CommandType = System.Data.CommandType.StoredProcedure;
                    cmd.Parameters.AddWithValue("@DeptID", deptId);

                    conn.Open();

                    using (SqlDataReader reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            int count = reader.GetInt32(0);
                            Console.WriteLine($"✅ Total Employees in Department {deptId}: {count}");
                        }
                        else
                        {
                            Console.WriteLine("⚠️ No data returned.");
                        }
                    }
                }
            }
        }
        catch (SqlException ex)
        {
            Console.WriteLine("❌ SQL Error: " + ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine("❌ General Error: " + ex.Message);
        }
    }
}
```

OUTPUT:



```
PS C:\Users\Lenovo\OneDrive\Desktop\Cognizant\week2\SQL\StoredProcedure> dotnet run
Enter Department ID: 1
✅ Total Employees in Department 1: 1
PS C:\Users\Lenovo\OneDrive\Desktop\Cognizant\week2\SQL\StoredProcedure>
```