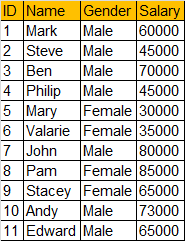
In this video we will discuss the use of **AsEnumerable**and **AsQueryable**operators in LINQ. Both of these operators belong to **Conversion Operators**category.   
  
   
  
**AsQueryable operator:** There are 2 overloaded versions of this method.   
  
One overloaded version converts System.Collections.IEnumerable to System.Linq.IQueryable  
  
The other overloaded version converts a generic System.Collections.Generic.IEnumerable<T> to a generic System.Linq.IQueryable<T>  
  
The main use of **AsQueryable**operator is unit testing to mock a **queryable**data source using an in-memory data source. We will discuss this operator in detail with examples in unit testing video series.  
  
**AsEnumerable operator:** Let us understand the use of this operator with an example. We will be using the following **Employees**table in this demo.   
   
  
**Step 1:** Execute the following SQL Script to create and populate **Employees**Table

Create Table Employees

(

     ID int primary key identity,

     Name nvarchar(50),

     Gender nvarchar(50),

     Salary int

)

GO

Insert into Employees Values('Mark','Male','60000')

Insert into Employees Values('Steve','Male','45000')

Insert into Employees Values('Ben','Male','70000')

Insert into Employees Values('Philip','Male','45000')

Insert into Employees Values('Mary','Female','30000')

Insert into Employees Values('Valarie','Female','35000')

Insert into Employees Values('John','Male','80000')

Insert into Employees Values('Pam','Female','85000')

Insert into Employees Values('Stacey','Female','65000')

Insert into Employees Values('Andy','Male','73000')

Insert into Employees Values('Edward','Male','65000')

GO

**Step 2:** Create a new Console Application. Name it **Demo**.  
  
**Step 3:**Right click on the Demo project in Solution Explorer and Add a new LINQ to SQL Classes. Name it **EmployeeDB.dbml**.  
  
**Step 4:**Click on **View**menu, and select **"Server Explorer".**  Expand **Data Connections**and then Drag and Drop **Employees**table onto **EmployeeDB.dbml**designer surface.  
  
**Step 5:**Copy and paste the following code in Program.cs file. The linq query in this sample, retrieves the **TOP 5 Male Employees By Salary**.

using System;

using System.Linq;

namespace Demo

{

    class Program

    {

        public static void Main()

        {

            EmployeeDBDataContext dbContext = new EmployeeDBDataContext();

            // TOP 5 Male Employees By Salary

            var result = dbContext.Employees.Where(x => x.Gender == "Male")

                                    .OrderByDescending(x => x.Salary).Take(5);

            Console.WriteLine("Top 5 Salaried Male Employees");

            foreach (Employee e in result)

            {

                Console.WriteLine(e.Name + "\t" + e.Gender + "\t" + e.Salary);

            }

        }

    }

}

**Step 6:** Now open **SQL Profiler**and run a new trace and then run the console application.  
  
**Step 7:**Notice that the following SQL Query is executed against the database.

exec sp\_executesql N'SELECT TOP (5) [t0].[ID], [t0].[Name], [t0].[Gender], [t0].[Salary]

FROM [dbo].[Employees] AS [t0]

WHERE [t0].[Gender] = @p0

ORDER BY [t0].[Salary] DESC',N'@p0 nvarchar(4000)',@p0=N'Male'

**Step 8:** Change the LINQ query in the console application   
  
**FROM**

varresult=dbContext.Employees.Where**(**x=>x.Gender=="Male"**)**

.OrderByDescending**(**x=>x.Salary**)**.Take**(**5**);**  
  
**TO**

varresult=dbContext.Employees.AsEnumerable**()**

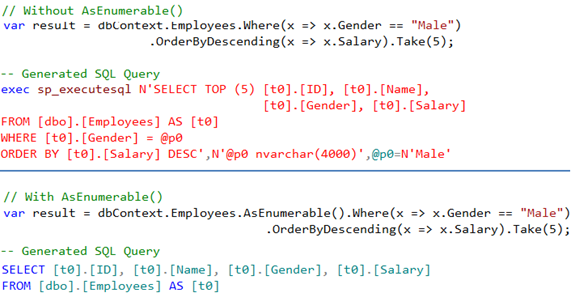
.Where**(**x=>x.Gender=="Male"**)**

.OrderByDescending**(**x=>x.Salary**)**.Take**(**5**);**

**Step 9:**Run the console application and notice the query generated in SQL Profiler.

SELECT [t0].[ID], [t0].[Name], [t0].[Gender], [t0].[Salary]

FROM [dbo].[Employees] AS [t0]

**Summary:**   
   
  
**AsEnumerable operator breaks the query into 2 parts**  
**1.** The "inside part" that is the query before AsEnumerable operator is executed as Linq-to-SQL  
**2.** The "ouside part" that is the query after AsEnumerable operator is executed as Linq-to-Objects  
  
So in this example the following SQL Query is executed against SQL Server, all the data is brought into the console application and then the WHERE, ORDERBY & TOP operators are applied on the client-side

SELECT [t0].[ID], [t0].[Name], [t0].[Gender], [t0].[Salary]

FROM [dbo].[Employees] AS [t0]

So in short, use **AsEnumerable** operator to move query processing to the client side.