In this video we will discuss the concept of **deferred execution**. LINQ queries have two different behaviors of execution  
**1.** Deferred execution  
**2.** Immediate execution   
  
   
  
**LINQ operators can be broadly classified into 2 categories based on the behaviour of query execution**  
**1. Deferred or Lazy Operators -**These query operators use deferred execution.   
Examples - select, where, Take, Skip etc  
**2. Immediate or Greedy Operators -**These query operators use immediate execution.   
Examples - count, average, min, max, ToList etc  
  
Let us understand these 2 behaviors with examples.   
  
**LINQ Deferred Execution Example**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Demo

{

    public class Student

    {

        public int StudentID { get; set; }

        public string Name { get; set; }

        public int TotalMarks { get; set; }

    }

    class Program

    {

        public static void Main()

        {

            List<Student> listStudents = new List<Student>

            {

                new Student { StudentID= 101, Name = "Tom", TotalMarks = 800 },

                new Student { StudentID= 102, Name = "Mary", TotalMarks = 900 },

                new Student { StudentID= 103, Name = "Pam", TotalMarks = 800 }

            };

            // LINQ Query is only defined here and is not executed at this point

            // If the query is executed at this point, the result should not display Tim

            IEnumerable<Student> result = from student in listStudents

                                          where student.TotalMarks == 800

                                          select student;

            // Add a new student object with TotalMarks = 800 to the source

            listStudents.Add(new Student { StudentID = 104, Name = "Tim", TotalMarks = 800 });

            // The above query is actually executed when we iterate thru the sequence

            // using the foreach loop. This is proved as Tim is also included in the result

            foreach (Student s in result)

            {

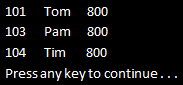
                Console.WriteLine(s.StudentID + "\t" + s.Name + "\t" + s.TotalMarks);

            }

        }

    }

}

**Output:**   
   
  
**LINQ Immediate Execution Example 1**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Demo

{

    public class Student

    {

        public int StudentID { get; set; }

        public string Name { get; set; }

        public int TotalMarks { get; set; }

    }

    class Program

    {

        public static void Main()

        {

            List<Student> listStudents = new List<Student>

            {

                new Student { StudentID= 101, Name = "Tom", TotalMarks = 800 },

                new Student { StudentID= 102, Name = "Mary", TotalMarks = 900 },

                new Student { StudentID= 103, Name = "Pam", TotalMarks = 800 }

            };

            // Since we are using ToList() which is a greedy operator

            // the LINQ Query is executed immediately at this point

            IEnumerable<Student> result = (from student in listStudents

                                           where student.TotalMarks == 800

                                           select student).ToList();

            // Adding a new student object with TotalMarks = 800 to the source

            // will have no effect on the result as the query is already executed

            listStudents.Add(new Student { StudentID = 104, Name = "Tim", TotalMarks = 800 });

            // The above query is executed at the point where it is defined.

            // This is proved as Tim is not included in the result

            foreach (Student s in result)

            {

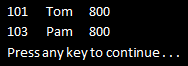
                Console.WriteLine(s.StudentID + "\t" + s.Name + "\t" + s.TotalMarks);

            }

        }

    }

}

**Output:**   
   
  
**LINQ Immediate Execution Example 2**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Demo

{

    public class Student

    {

        public int StudentID { get; set; }

        public string Name { get; set; }

        public int TotalMarks { get; set; }

    }

    class Program

    {

        public static void Main()

        {

            List<Student> listStudents = new List<Student>

            {

                new Student { StudentID= 101, Name = "Tom", TotalMarks = 800 },

                new Student { StudentID= 102, Name = "Mary", TotalMarks = 900 },

                new Student { StudentID= 103, Name = "Pam", TotalMarks = 800 }

            };

            // Since we are using Count() operator, the LINQ Query is executed at this point

            int result = (from student in listStudents

                          where student.TotalMarks == 800

                          select student).Count();

            // Adding a new student object with TotalMarks = 800 to the source

            // will have no effect on the result as the query is already executed

            listStudents.Add(new Student { StudentID = 104, Name = "Tim", TotalMarks = 800 });

            // The above query is executed at the point where it is defined.

            // This is proved as Tim is not included in the count

            Console.WriteLine("Students with Total Marks = 800 : " + result);

        }

    }

}

**Output:**   
force linq query execute immediately