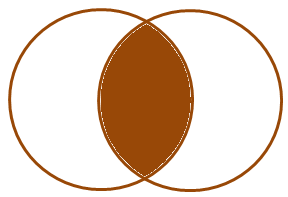
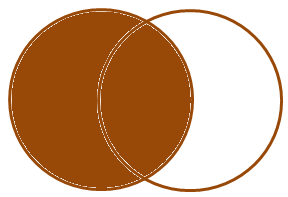
**The following are the different types of joins in LINQ**  
Group Join - Discussed in [Part 21](http://csharp-video-tutorials.blogspot.com/2014/07/part-21-group-join-in-linq.html)  
Inner Join - Discussed in [Part 22](http://csharp-video-tutorials.blogspot.com/2014/08/part-22-inner-join-in-linq.html)  
Left Outer Join - We will discuss in this video  
Cross Join - Later Video   
  
   
  
In this video we will discuss implementing **LEFT OUTER JOIN in LINQ**.   
  
With **INNER JOIN**only the matching elements are included in the result set. Non-matching elements are excluded from the result set.   
   
  
With **LEFT OUTER JOIN**all the matching elements + all the non matching elements from the left collection are included in the result set.   
   
  
Let us understand implementing **Left Outer Join**with an example. Consider the following **Department**and **Employee**classes. Notice that, Employee Mary does not have a department assigned. An inner join will not include her record in the result set, where as a Left Outer Join will.

public class Department

{

    public int ID { get; set; }

    public string Name { get; set; }

    public static List<Department> GetAllDepartments()

    {

        return new List<Department>()

        {

            new Department { ID = 1, Name = "IT"},

            new Department { ID = 2, Name = "HR"},

        };

    }

}

public class Employee

{

    public int ID { get; set; }

    public string Name { get; set; }

    public int DepartmentID { get; set; }

    public static List<Employee> GetAllEmployees()

    {

        return new List<Employee>()

        {

            new Employee { ID = 1, Name = "Mark", DepartmentID = 1 },

            new Employee { ID = 2, Name = "Steve", DepartmentID = 2 },

            new Employee { ID = 3, Name = "Ben", DepartmentID = 1 },

            new Employee { ID = 4, Name = "Philip", DepartmentID = 1 },

            new Employee { ID = 5, Name = "Mary" }

        };

    }

}

Use **DefaultIfEmpty**() method on the results of a group join to implement **Left Outer Join**  
  
**Example 1** : Implement a **Left Outer Join**between **Employees**and **Department**collections and print all the Employees and their respective department names. Employees without a department, should display **"No Department"**against their name.

var result = from e in Employee.GetAllEmployees()

                    join d in Department.GetAllDepartments()

                    on e.DepartmentID equals d.ID into eGroup

                    from d in eGroup.DefaultIfEmpty()

                    select new

                    {

                         EmployeeName = e.Name,

                         DepartmentName = d == null ? "No Department" : d.Name

                    };

foreach (var v in result)

{

    Console.WriteLine(v.EmployeeName + "\t" + v.DepartmentName);

}

**Output:** Notice that, we also have **Mary**record in spite of she not having a department. So this is effectively a left outer join.   
   
  
**Example 2 :** Rewrite **Example 1**using extension method syntax.

var result = Employee.GetAllEmployees()

                        .GroupJoin(Department.GetAllDepartments(),

                                e => e.DepartmentID,

                                d => d.ID,

                                (emp, depts) => new { emp, depts })

                        .SelectMany(z => z.depts.DefaultIfEmpty(),

                                (a, b) => new

                                {

                                        EmployeeName = a.emp.Name,

                                        DepartmentName = b == null ? "No Department" : b.Name

                                });

foreach (var v in result)

{

    Console.WriteLine(" " + v.EmployeeName + "\t" + v.DepartmentName);

}

To implement **Left Outer Join**, with extension method syntax we use the **GroupJoin()**method along with **SelectMany()**and **DefaultIfEmpty()**methods.