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** - We will discuss in this video  
**Left Outer Join** - Later Video  
**Cross Join** - Later Video   
  
   
  
In this video we will discuss implementing **INNER JOIN**in **LINQ**. If you have 2 collections, and when you perform an inner join, then only the matching elements between the 2 collections are included in the result set. Non - Matching elements are excluded from the result set.  
  
Let us understand Inner Join with an example. Consider the following **Department**and **Employee**classes. Notice that, **Employee**Andy does not have a department assigned. An inner join will not include his record in the result set.

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"},

            new Department { ID = 3, Name = "Payroll"},

        };

    }

}

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", DepartmentID = 2 },

            new Employee { ID = 6, Name = "Valarie", DepartmentID = 2 },

            new Employee { ID = 7, Name = "John", DepartmentID = 1 },

            new Employee { ID = 8, Name = "Pam", DepartmentID = 1 },

            new Employee { ID = 9, Name = "Stacey", DepartmentID = 2 },

            new Employee { ID = 10, Name = "Andy"}

        };

    }

}

**Example 1 :** Join the **Employees**and **Department**collections and print all the Employees and their respective department names.

var result = Employee.GetAllEmployees().Join(Department.GetAllDepartments(),

                                        e => e.DepartmentID,

                                        d => d.ID, (employee, department) => new

                                        {

                                            EmployeeName = employee.Name,

                                            DepartmentName = department.Name

                                        });

foreach (var employee in result)

{

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

}

**Output:**Notice that, in the output we don't have **Andy**record. This is because, Andy does not have a matching department in Department collection. So this is effectively an **inner join**.   
   
  
**Example 2 :**Rewrite Example 1 using SQL like syntax.

var result = from e in Employee.GetAllEmployees()

                    join d in Department.GetAllDepartments()

                    on e.DepartmentID equals d.ID

                    select new

                    {

                        EmployeeName = e.Name,

                        DepartmentName = d.Name

                    };

foreach (var employee in result)

{

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

}