In this video, we will discuss **Grouping by multiple keys**. In LINQ, an anonymous type is usually used when we want to group by multiple keys.   
  
   
  
Let us understand this with an example. We will be using the following **Employee**class in this demo. This is the same class used in [Part 18](http://csharp-video-tutorials.blogspot.com/2014/07/part-18-groupby-in-linq.html). Please watch [Part 18](http://csharp-video-tutorials.blogspot.com/2014/07/part-18-groupby-in-linq.html) before proceeding.

public class Employee

{

    public int ID { get; set; }

    public string Name { get; set; }

    public string Gender { get; set; }

    public string Department { get; set; }

    public static List<Employee> GetAllEmployees()

    {

        return new List<Employee>()

        {

            new Employee { ID = 1, Name = "Mark", Gender = "Male",   
                                         Department = "IT" },

            new Employee { ID = 2, Name = "Steve", Gender = "Male",   
                                         Department = "HR" },

            new Employee { ID = 3, Name = "Ben", Gender = "Male",   
                                         Department = "IT" },

            new Employee { ID = 4, Name = "Philip", Gender = "Male",   
                                         Department = "IT" },

            new Employee { ID = 5, Name = "Mary", Gender = "Female",   
                                         Department = "HR" },

            new Employee { ID = 6, Name = "Valarie", Gender = "Female",   
                                         Department = "HR" },

            new Employee { ID = 7, Name = "John", Gender = "Male",   
                                         Department = "IT" },

            new Employee { ID = 8, Name = "Pam", Gender = "Female",   
                                         Department = "IT" },

            new Employee { ID = 9, Name = "Stacey", Gender = "Female",   
                                         Department = "HR" },

            new Employee { ID = 10, Name = "Andy", Gender = "Male",   
                                         Department = "IT" },

        };

    }

}

**Example 1:** Group employees by **Department**and then by **Gender**. The employee groups should be sorted first by **Department**and then by **Gender**in ascending order. Also, employees within each group must be sorted in ascending order by Name.

var employeeGroups = Employee.GetAllEmployees()

                                        .GroupBy(x => new { x.Department, x.Gender })

                                        .OrderBy(g => g.Key.Department).ThenBy(g => g.Key.Gender)

                                        .Select(g => new

                                        {

                                            Dept = g.Key.Department,

                                            Gender = g.Key.Gender,

                                            Employees = g.OrderBy(x => x.Name)

                                        });

foreach(var group in employeeGroups)

{

    Console.WriteLine("{0} department {1} employees count = {2}",

        group.Dept, group.Gender, group.Employees.Count());

    Console.WriteLine("--------------------------------------------");

    foreach (var employee in group.Employees)

    {

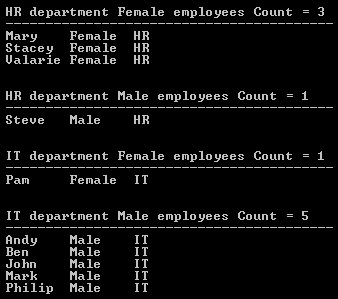
        Console.WriteLine(employee.Name + "\t" + employee.Gender

            + "\t" + employee.Department);

    }

    Console.WriteLine(); Console.WriteLine();

}

**Output:**   
   
  
**Example 2:** Rewrite Example 1 using **SQL like syntax**

varemployeeGroups=fromemployeeinEmployee.GetAllEmployees**()**

groupemployeebynew

**{**

employee.Department**,**

employee.Gender

**}**intoeGroup

orderbyeGroup.Key.Departmentascending**,**  
eGroup.Key.Genderascending

selectnew

**{**

Dept=eGroup.Key.Department**,**

Gender=eGroup.Key.Gender**,**

Employees=eGroup.OrderBy**(**x=>x.Name**)**

**};**