The following standard LINQ query operators belong to **Conversion Operators**category  
ToList  
ToArray  
ToDictionary  
ToLookup  
Cast  
OfType  
AsEnumerable   
AsQueryable   
  
   
  
**ToList operator** extracts all of the items from the source sequence and returns a new **List<T>**. This operator causes the query to be executed immediately. This operator does not use deferred execution.  
  
**Example 1:**Convert int array to List<int>

using System;

using System.Collections.Generic;

using System.Linq;

namespace Demo

{

    class Program

    {

        public static void Main()

        {

            int[] numbers = { 1, 2, 3, 4, 5 };

            List<int> result = numbers.ToList();

            foreach (int i in result)

            {

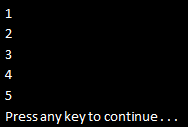
                Console.WriteLine(i);

            }

        }

    }

}

**Output:**   
   
  
**ToArray** operator extracts all of the items from the source sequence and returns a new Array. This operator causes the query to be executed immediately. This operator does not use deferred execution.  
  
**Example 2:**Convert List<string> to string array. The items in the array should be sorted in ascending order.

using System;

using System.Collections.Generic;

using System.Linq;

namespace Demo

{

    class Program

    {

        public static void Main()

        {

            List<string> countries = new List<string> { "US", "India", "UK", "Australia", "Canada" };

            string[] result = (from country in countries

                               orderby country ascending

                               select country).ToArray();

            foreach (string str in result)

            {

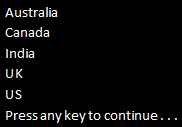
                Console.WriteLine(str);

            }

        }

    }

}

**Output:**   
   
  
**ToDictionary** operator extracts all of the items from the source sequence and returns a new Dictionary. This operator causes the query to be executed immediately. This operator does not use deferred execution.  
  
**Example 3 :**Convert List<Student> to a Dictionary. StudentID should be the key and Name should be the value. In this example, we are using the overloaded of ToDictionary() that takes 2 parameters   
**a) keySelector** - A function to extract a key from each element  
**b) elementSelector** - A function to produce a result element from each element in the sequence

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 }

            };

            Dictionary<int, string> result = listStudents

                                                                     .ToDictionary(x => x.StudentID, x => x.Name);

            foreach (KeyValuePair<int, string> kvp in result)

            {

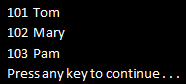
                Console.WriteLine(kvp.Key + " " + kvp.Value);

            }

        }

    }

}

**Output:**   
   
  
**Example 4 :** Convert List<Student> to a Dictionary. StudentID should be the key and Student object should be the value. In this example, we are using the overloaded of ToDictionary() that takes 1 parameter  
**a) keySelector** - A function to extract a key from each element

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 }

            };

            Dictionary<int, Student> result = listStudents.ToDictionary(x => x.StudentID);

            foreach (KeyValuePair<int, Student> kvp in result)

            {

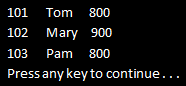
                Console.WriteLine(kvp.Key + "\t" + kvp.Value.Name + "\t" + kvp.Value.TotalMarks);

            }

        }

    }

}

**Output:**   
   
  
**Please Note:** Keys in the dictionary must be unique. If two identical keys are created by the keySelector function, the following System.ArgumentException will be thrown at runtime.  
Unhandled Exception: System.ArgumentException: An item with the same key has already been added.  
  
**ToLookup** creates a Lookup. Just like a dictionary, a Lookup is a collection of key/value pairs. A dictionary cannot contain keys with identical values, where as a Lookup can.  
  
**Example 5:** Create 2 Lookups. First lookup should group Employees by JobTitle, and second lookup should group Employees by City.

using System;

using System.Collections.Generic;

using System.Linq;

namespace Demo

{

    public class Employee

    {

        public string Name { get; set; }

        public string JobTitle { get; set; }

        public string City { get; set; }

    }

    class Program

    {

        public static void Main()

        {

            List<Employee> listEmployees = new List<Employee>

            {

                new Employee() { Name = "Ben", JobTitle = "Developer", City = "London" },

                new Employee() { Name = "John", JobTitle = "Sr. Developer", City ="Bangalore" },

                new Employee() { Name = "Steve", JobTitle = "Developer", City = "Bangalore"},

                new Employee() { Name = "Stuart", JobTitle = "Sr. Developer", City ="London" },

                new Employee() { Name = "Sara", JobTitle = "Developer", City = "London" },

                new Employee() { Name = "Pam", JobTitle = "Developer", City = "London" }

            };

            // Group employees by JobTitle

            var employeesByJobTitle = listEmployees.ToLookup(x => x.JobTitle);

            Console.WriteLine("Employees Grouped By JobTitle");

            foreach (var kvp in employeesByJobTitle)

            {

                Console.WriteLine(kvp.Key);

                // Lookup employees by JobTitle

                foreach (var item in employeesByJobTitle[kvp.Key])

                {

                    Console.WriteLine("\t" + item.Name + "\t" + item.JobTitle + "\t" + item.City);

                }

            }

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

            // Group employees by City

            var employeesByCity = listEmployees.ToLookup(x => x.City);

            Console.WriteLine("Employees Grouped By City");

            foreach (var kvp in employeesByCity)

            {

                Console.WriteLine(kvp.Key);

                // Lookup employees by City

                foreach (var item in employeesByCity[kvp.Key])

                {

                    Console.WriteLine("\t" + item.Name + "\t" + item.JobTitle + "\t" + item.City);

                }

            }

        }

    }

}

Output:   
