* LINQ stands for Language Integrated Query.
* It Provides ability to .NET language like C# to create queries

and retrieve data from various data sources including SQL databases, XML , Objects , dataset.

* In the past, different types of data were stored in different database ,

requiring knowledge of various query languages (e.g., SQL, XML) to access and retrieve data.

* today LINQ addresses this issue by allowing developers to create queries using C# or

other .NET languages.

* It can be used with various data sources, including SQL databases, XML , Objects , dataset .
* If we have collection / data of any database then we don’t need to learn any particular language to query out that data. While we can use our native c# language to query out data .

2 types of syntax =

1. Query syntax

var data = from i in integerList

where i > 5

select i;

int min = data.Min();

1. Method syntax =

var data =

student.Where((i) => i.Subjects.Contains("Computers"))

.OrderBy(i => i.Name)

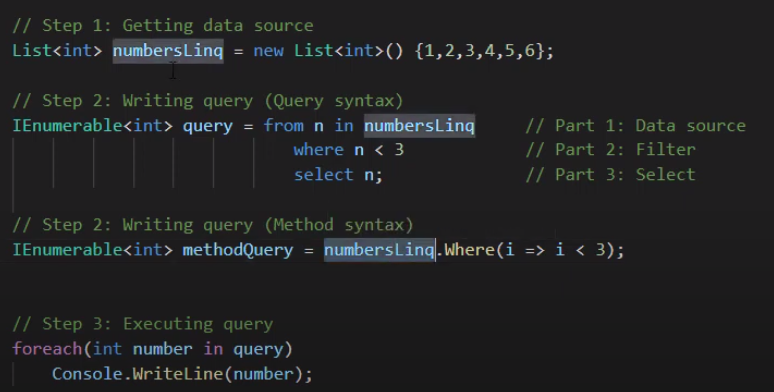
.Select(i=> new { i.Name, i.Gender , i.Subjects });

1. Mixed syntax =

var data = ( from i in integerList

where i < 5

select i).Sum();

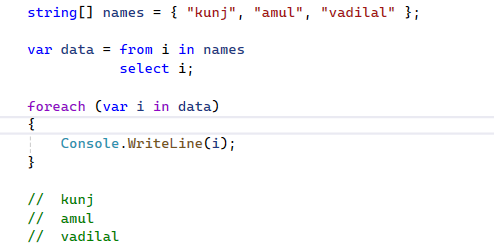


select =

(SQL) = select \* from employee

(LINQ) = from i in employee

select name;



or

var data = names.Select(i => i);

where =

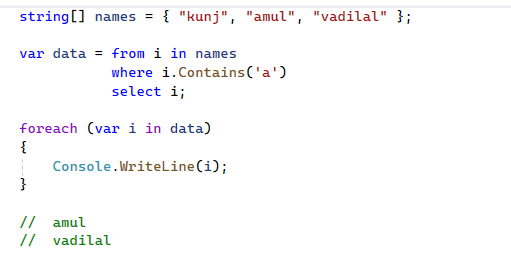
(SQL) = select \* from employee

where name like '%a%'

(LINQ) = from i in names

where name.Contains('a')

select i;



Or

var data1 = names.Where(i => i.Contains('a'));

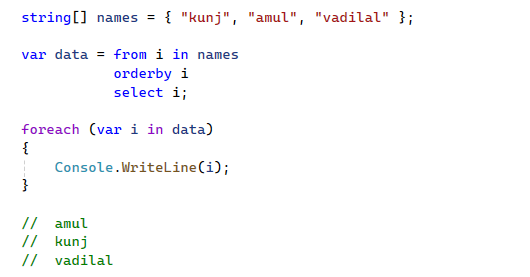
Order ( sorting ) =

Sorting Operators available in LINQ are:

1. ORDER BY = names.OrderBy( i => i.Name );
2. ORDER BY DESCENDING = names.OrderByDescending( i => i.Name );
3. THEN BY = names.OrderBy( i => i.FName ).ThenBy(i=>i.LName)
4. THEN BY DESCENDING = names.OrderBy( i => i.FName ).ThenByDescending (i=>i.LName)
5. REVERSE

* Then by is 2nd level of sorting.

First ord by first-name then order by last name.



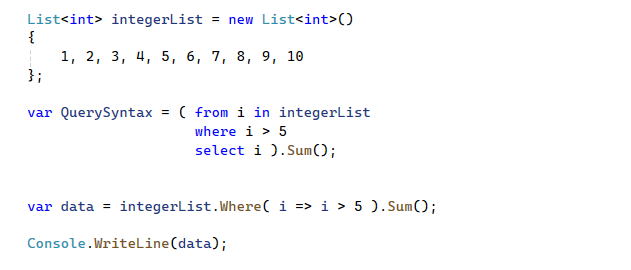
For asc =

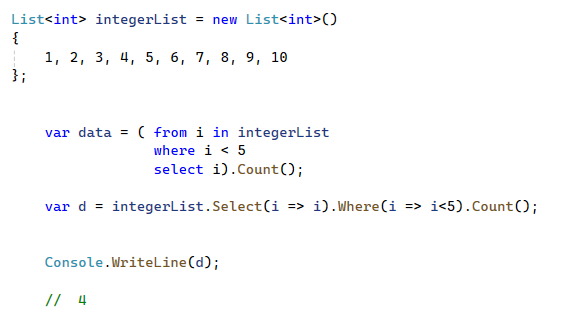
var a = names.OrderBy(i => i.Name);

for desc order =

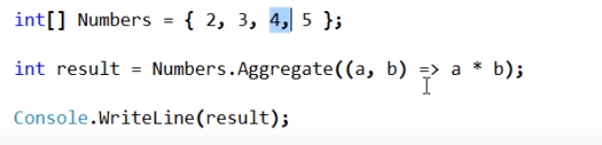
var a = names.OrderByDescending(i => i.Name);

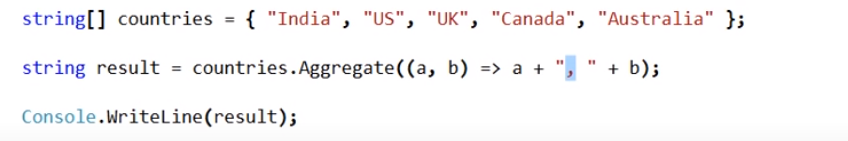
Sum / Max / Min / Count =





Aggregate =

 // 2\*3\*4\*5 = 120



= india,us,uk,..,Australia in string it will give comma separated

Take in linq / Top in sql =

var data = stu1.Take(1); // it will give top 1 value. method syntax

var b = (from i in stu1 // query syntax

select i).Take(2);

Takewhile =

It is same as Where = it use to check condition on all item.

In TakeWhile it will start from first item and check condition . and it stop after first false condition.

var a = stu1.TakeWhile(i => i.Gender is "Male");

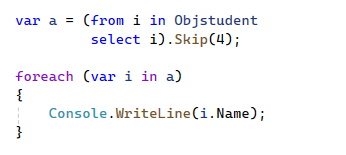
it stops after 1st false condition.

Use case of TakeWhile = we have transaction list in sorted manner of date.

And we want to retrieve till certain date at that time we can use Takewhile.

Skip in linq ( offset in sql )=

It will skip specified item from list. And display remaining item.



ToList() =

It is used to convert input elements into list.

We can convert array data into list by

int[] a = { 1, 2, 3, 4 }; array

List<int> num = a.ToList(); array convert into list

foreach (var i in b)

{

Console.WriteLine(i);

}

We can do vise versa

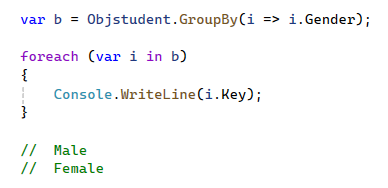
ToArray() =

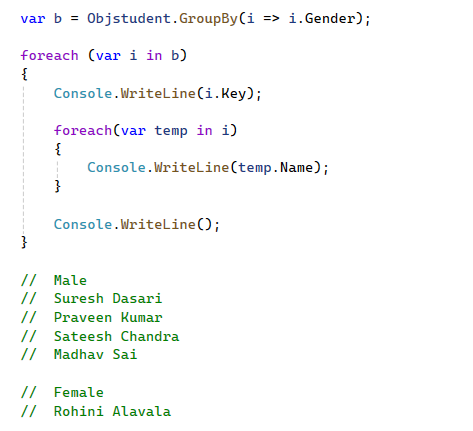
It is used to convert input elements into collection of array.

string[] countries = { "Uk", "Us", "Russia", "India", "Argentina", "Australia", "China" };

string[] arr = countries.ToArray();

Group by =





(Group by) is Lazy execution = The actual filtering and transformation happen when you iterate over 'query'.

var result = myList.Where (item => item.SomeCondition)

(ToLookup) is Immediate execution = The filtering happens immediately, and 'result' contains the filtered items.

var result = myList.Where (item => item.SomeCondition).ToList();