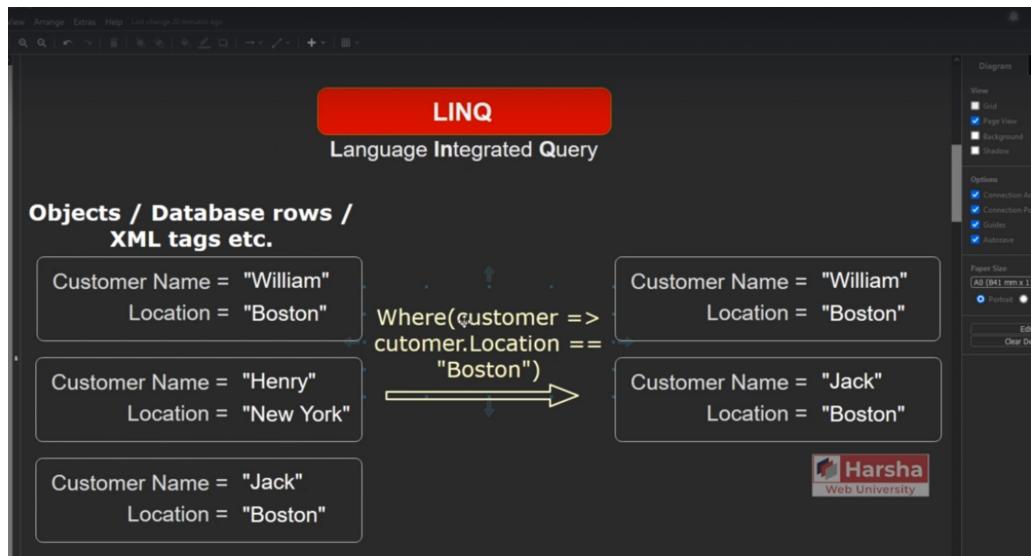
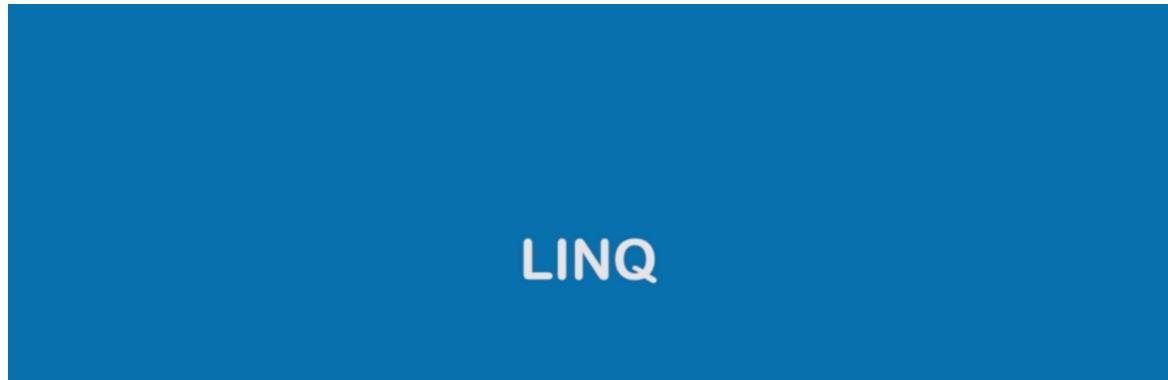


Section 25 : LINQ

28 July 2024 20:40

1. Linq Basics



It says that you will write only one query against any type of data source your data can be in the form of list of objects or database rows maybe SQL server or oracle or any other database rows or you might have data in the form of other data sources such as xml tags or csv files etc

you will write only one type of queries against all types of data sources for example we are writing a query saying that I would like to retrieve the customers where the location equal to boston no matter what is the actual data source from this list of customers you will get the corresponding matching rows so you can print that results or you can store into another variable

or you can do anything with these results.

Link is actually a framework which has been supported by c-sharp and vb.net languages.

The screenshot shows a Visual Studio code editor with the following C# code:

```
18 {
19     //collection of objects
20     List<Employee> employees = new List<Employee>()
21     {
22         new Employee() { EmpID = 101, EmpName = "Henry", Job = "Designer", City = "Boston" },
23         new Employee() { EmpID = 102, EmpName = "Jack", Job = "Developer", City = "New York" },
24         new Employee() { EmpID = 103, EmpName = "Gabriel", Job = "Analyst", City = "Tokyo" },
25         new Employee() { EmpID = 104, EmpName = "William", Job = "Manager", City = "Tokyo" },
26         new Employee() { EmpID = 105, EmpName = "Alexa", Job = "Manager", City = "New York" }
27     };
28
29     foreach (Employee emp in employees)
30     {
31     }
32     employees.Where(emp => emp.Job == "Manager");
33 }
34
35 }
36 }
```

Annotations in red highlight the list creation, the employee objects, and the Where clause. A purple arrow points from the text "Lamda Expression" down to the Where clause. Two red arrows point from the word "False" to the two Manager entries in the list.

Lamda Expression

The group of yield return values can form a IEnumerable object.

```

new Employee() { EmpID = 101, EmpName = "Sach", Job = "Developer", City = "New York" },
new Employee() { EmpID = 103, EmpName = "Gabriel", Job = "Analyst", City = "Tokyo" },
new Employee() { EmpID = 104, EmpName = "William", Job = "Manager", City = "Tokyo" },
new Employee() { EmpID = 105, EmpName = "Alexa", Job = "Manager", City = "New York" } ✓
};

foreach (Employee emp in employees) .           I Enumerable<Employee>
{
}

var result = employees.Where(emp => emp.Job == "Manager");

```



Understanding LINQ → **Next: Advantages of LINQ**

What

- LINQ is a 'uniform query syntax' that allows you to retrieve data from various data sources such as arrays, collections, databases, XML files.

```

graph LR
    Developer((Developer)) -- Uses --> LINQ[LINQ]
    LINQ -- "to Retrieve from" --> Collection[Collection of Objects  
LINQ to Collections]
    LINQ -- "to Retrieve from" --> SQL[SQL Server Database  
LINQ to SQL]
    LINQ -- "to Retrieve from" --> Entity[Entity Framework DbSet  
LINQ to Entities]
    LINQ -- "to Retrieve from" --> ADO[ADO.NET DataSet  
LINQ to DataSet]
    LINQ -- "to Retrieve from" --> XML[XML Files  
LINQ to XML]

```

How

LINQ Query - Example

```

var result = Customers.Where(temp => temp.Location == "New York").ToList();
//returns a list of customers from New York location.

```

1:20

› **Single Syntax - To Query Multiple Data Sources**

- › Developer uses the same LINQ syntax to retrieve information from various data sources such as collections, SQL Server database, Entity Framework DbSet's, ADO.NET DataSet etc.



› **Compile-Time Checking of Query Errors**

- › Errors in the LINQ query will be identified while compilation time / while writing the code in Visual Studio.

› **IntelliSense Support**

- › The list of properties of types are shown in VS IntelliSense while writing the LINQ queries.

Classification	LINQ Extension Methods / LINQ Operators
Filtering	Where, OfType
Sorting	OrderBy, OrderByDescending, ThenBy, ThenByDescending, Reverse
Grouping	GroupBy
Join	Join
Project	Select, SelectMany
Aggregation	Average, Count, Max, Min, Sum
Quantifiers	All, Any, Contains
Elements	ElementAt, ElementAtOrDefault, First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault
Set Operations	Distinct, Except, Intersect, Union
Partitioning	Skip, SkipWhile, Take, TakeWhile
Concatenation	Concat
Equality	SequenceEqual
Generation	DefaultEmpty, Empty, Range, Repeat
Conversion	AsEnumerable, AsQueryable, Cast, ToArray, ToDictionary, ToList

2. OrderBy

OrderBy

OrderBy Next: First

What

- OrderBy() method sorts collection based on given lambda expression (property) and returns a new collection with sorted elements.

Example

Customer Name = "Scott" Location = "Hyderabad"	Customer Name = "Allen" Location = "New York"
Customer Name = "Jones" Location = "New Delhi"	Customer Name = "Jones" Location = "New Delhi"
Customer Name = "Allen" Location = "New York"	Customer Name = "Scott" Location = "Hyderabad"

OrderBy()

How

OrderBy Extension Method - Declaration

```
OrderBy(Func<TSource, TKey> keySelector)
```

OrderBy Extension Method - Usage

```
var result = Customers.OrderBy(temp => temp.CustomerName).ToList();
//returns a list of customers sorted based on customer name.
```

10:54

OrderBy Next: First

TheBy
Descending

ThenByDescending Extension Method - Declaration

```
ThenByDescending(Func<TSource, TKey> keySelector)
```

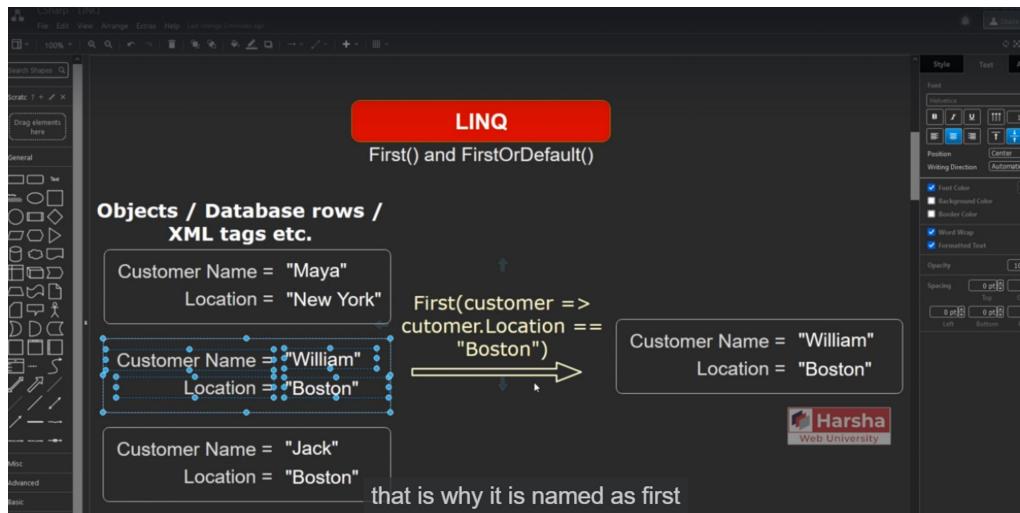
ThenByDescending Extension Method - Usage

```
var result = Customers.OrderBy(temp => temp.Location)
    .ThenByDescending(temp => temp.CustomerName).ToList();
//returns a list of customers sorted based on location (ascending) and customer name (descending).
```

Harsha
Web University

3. FirstOrDefault

First() and FirstOrDefault()



First **Next: FirstOrDefault**

What

- First() method returns first element in the collection that matches with the collection.
- It throws exception if no element matches with the condition.

Example

Customer Name = "Scott"
Location = "Dallas"

Customer Name = "Smith"
Location = "Dallas"

Customer Name = "Allen"
Location = "New York"

First()

Customer Name = "Scott"
Location = "Dallas"

How

First Extension Method - Declaration

First(Func<TSource, bool> predicate)

First Extension Method - Usage

var result = Customers.First(temp => temp.Location == "Dallas");

//returns the first customer from Hyderabad location.

What

- > `FirstOrDefault()` method returns first element that matches with the condition.
- > It returns null if no element matches with the condition.

Example

```
Customer Name = "Scott"  
Location      = "Hyderabad"  
  
Customer Name = "Smith"  
Location      = "New Delhi"  
  
Customer Name = "Allen"  
Location      = "New York"
```

`FirstOrDefault()`

null

How**FirstOrDefault Extension Method - Declaration**

```
FirstOrDefault(Func<TSource, bool> predicate)
```

Harsha
Web University

FirstOrDefault Extension Method - Usage

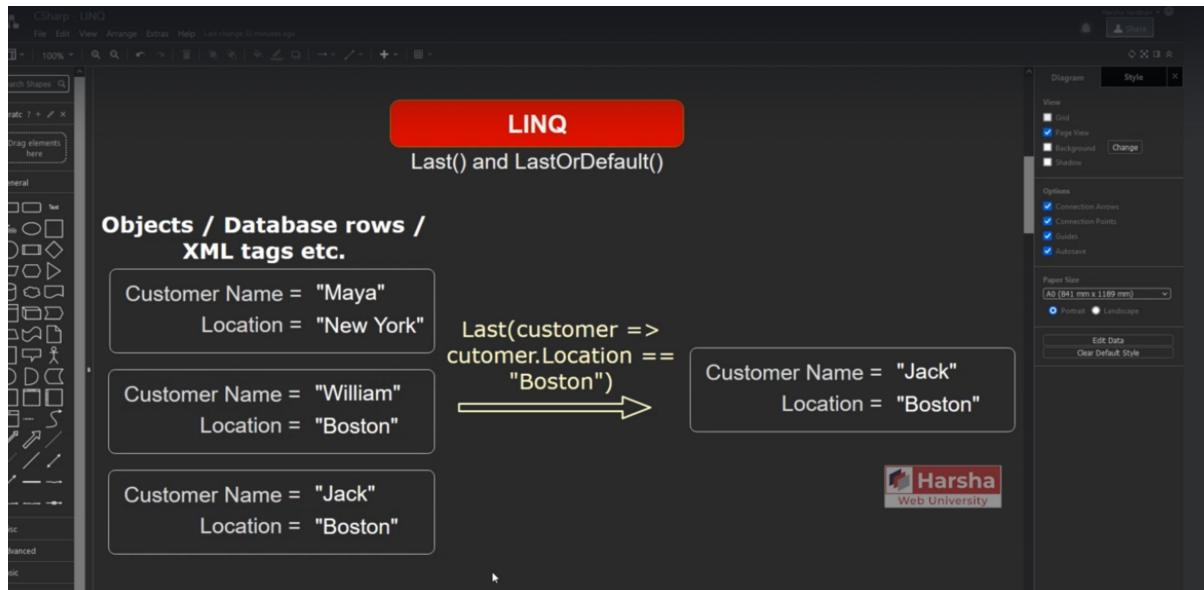
null

```
var result = Customers.FirstOrDefault(temp => temp.Location == "London");  
//returns the first customer from London location (or) returns null if not exists.
```

5:24

4. Last and LastOrDefault

Last() and LastOrDefault()



Last **Next: LastOrDefault**

What

- › `Last()` method returns last element in the collection that matches with the collection.
- › It throws exception if no element matches with the condition.

Example

Customer Name = "Scott"
Location = "Dallas"

Customer Name = "Smith"
Location = "Dallas"

Customer Name = "Allen"
Location = "New York"

Last()

Customer Name = "Smith"
Location = "Dallas"

How

Last Extension Method - Declaration

```
Last(Func<TSource, bool> predicate)
```

Last Extension Method - Usage

```
var result = Customers.Last(temp => temp.Location == "Dallas");
//returns the last customer from Dallas location.
```

47

LastOrDefault

Next: Single

What

- > LastOrDefault() method returns last element that matches with the condition.
- > It returns null if no element matches with the condition.

Example

Customer Name = "Scott"	Location = "Hyderabad"
Customer Name = "Smith"	Location = "New Delhi"
Customer Name = "Allen"	Location = "New York"

LastOrDefault() → null

How

LastOrDefault Extension Method - Declaration

```
LastOrDefault(Func<TSource, bool> predicate)
```

LastOrDefault Extension Method - Usage

```
var result = Customers.LastOrDefault(temp => temp.Location == "London");
//returns the last customer from London location (or) returns null if not exists.
```

5. ElementAt() and ElementAtOrDefault()

ElementAt() and ElementAtOrDefault()

LINQ

ElementAt() and ElementAtOrDefault()

Objects / Database rows / XML tags etc.

Customer Name = "Maya"	Location = "New York"
Customer Name = "William"	Location = "Boston"
Customer Name = "Jack"	Location = "Boston"

ElementAt(1) → Customer Name = "Willaim"
Location = "Boston"

```

8   {
9     //collection of objects
10    List<Employee> employees = new List<Employee>()
11    {
12      new Employee() { EmpID = 101, EmpName = "Henry", Job = "Designer", Salary = 7232 },
13      new Employee() { EmpID = 102, EmpName = "Jack", Job = "Developer", Salary = 4500 },
14      new Employee() { EmpID = 103, EmpName = "Gabriel", Job = "Analyst", Salary = 1293 },
15      new Employee() { EmpID = 104, EmpName = "William", Job = "Manager", Salary = 2873 },
16      new Employee() { EmpID = 105, EmpName = "Alexa", Job = "Manager", Salary = 6232 }, 1|
17      new Employee() { EmpID = 106, EmpName = "Jessica", Job = "Manager", Salary = 4545 }
18    };
19
20    //ElementAt
21    employees.Where(emp => emp.Job == "Manager").ElementAt(1)
22
23    Console.ReadKey();
24  }
25}

```

here this index will be against only the



Where() methods return a list of Ienumerable, based on that ElementAt method executes for each of the employee object.

ElementAt Next: Single

What

- Element() method returns an element in the collection at specified index.
- It throws exception if no element exists at the specified index; to get 'null' instead, use ElementOrDefault().

Example

Customer Name = "Scott"	Location = "Dallas"
Customer Name = "Smith"	Location = "Dallas"
Customer Name = "Allen"	Location = "New York"

ElementAt()

Customer Name = "Smith"	Location = "Dallas"
-------------------------	---------------------

How

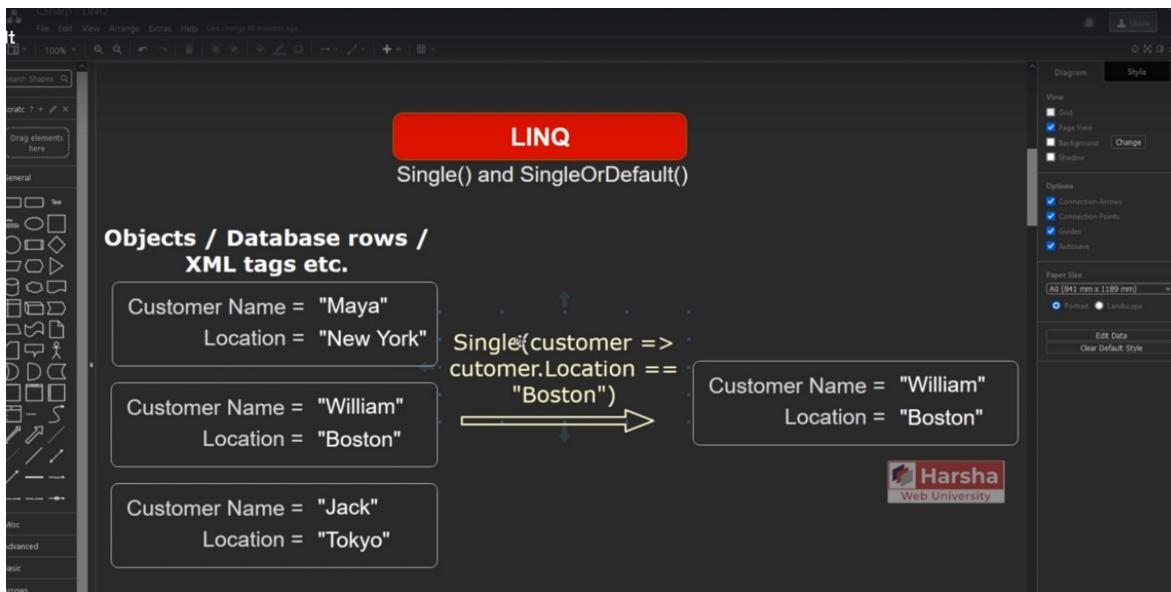
ElementAt Extension Method - Declaration

ElementAt(int index)

ElementAt Extension Method - Usage

```
var result = Customers.ElementAt(1); //returns the customer at index 1
```

Single() and SingleOrDefault()



It's similar to `First` / `FirstOrDefault`. The difference is, `Single` method expects there must be only one matching element in the whole list based on the given condition.

***Single* method throws exception in these following cases:**

- If more than one matching element is found, it throws exception.
- If no elements is found with the given condition, it throws exception.

***SingleOrDefault* method throws exception in these following cases:**

- If more than one matching element is found, it throws exception.

it returns Null if no element is found rather than throwing exception.

Next: SingleOrDefault

What

- It returns first element (only one element) that matches with the collection.
- It throws exception if no element or multiple elements match with the condition.

Example

Customer Name = "Scott" Location = "Dallas"
Customer Name = "Smith" Location = "Dallas"
Customer Name = "Allen" Location = "New York"

Single() → InvalidOperationException

How

Single Extension Method - Declaration

```
Single(Func<TSource, bool> predicate)
```

Single Extension Method - Usage

```
var result = Customers.Single(temp => temp.Location == "Dallas");  
//returns the first (only one customer) from Dallas location
```

but it throws exception if none / multiple elements matches with the condition.

What

- It returns first element (only one element) that matches with the collection.
- It returns null if no element matches with the condition; but it throws exception if multiple elements match with the condition.

Example

Customer Name = "Smith"
Location = "Hyderabad"
Customer Name = "Allen"
Location = "New York"

SingleOrDefault() → null

How

SingleOrDefault Extension Method - Declaration

```
SingleOrDefault(Func<TSource, bool> predicate)
```

SingleOrDefault Extension Method - Usage

```
var result = Customers.SingleOrDefault(temp => temp.Location == "London");
//returns the first (only one customer) from London location
```

it throws exception if multiple elements matches with the condition; but null in case of no match.

for this case single method is useful, where EmpId is duplicate.

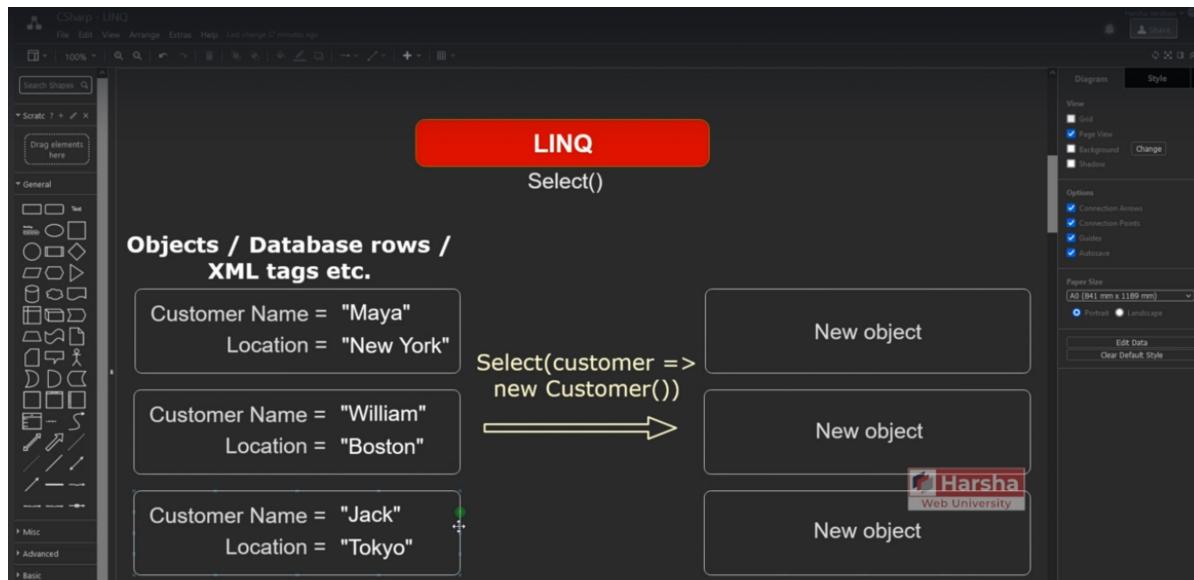
```
//collection of objects
List<Employee> employees = new List<Employee>()
{
    new Employee() { EmpID = 101, EmpName = "Henry", Job = "Designer", Salary = 7232 },
    new Employee() { EmpID = 102, EmpName = "Jack", Job = "Developer", Salary = 4500 },
    new Employee() { EmpID = 103, EmpName = "Gabriel", Job = "Analyst", Salary = 1293 },
    new Employee() { EmpID = 104, EmpName = "William", Job = "Manager", Salary = 2873 },
    new Employee() { EmpID = 105, EmpName = "Alexa", Job = "Manager", Salary = 6232 },
    new Employee() { EmpID = 102, EmpName = "Jessica", Job = "Manager", Salary = 4545 }
};

//Single
Employee resultEmployee1 = employees.Single(emp => emp.EmpID == 102);
Console.WriteLine(resultEmployee1.EmpID + ", " + resultEmployee1.Er
resultEmployee1.Job);

//SingleOrDefault
Employee resultEmplo the same with an exception like this.
if (resultEmployee1 != null)
{
    //Employee resultEmployee1 = employees.SingleOrDefault(emp => emp.EmpID == 102);
    Console.WriteLine(resultEmployee1.EmpID + ", " + resultEmployee1.Er
resultEmployee1.Job);
}
```

Select() LINQ Method

Select()



Select → **Next: Reverse**

What

- It returns collection by converting each element into another type, based on the conversion expression.

Example

From `List<Customer>` (containing customers for Scott, Smith, and Allen) to `List<RegisteredCustomer>` (containing registered customers for Scott, Smith, and Allen).

Select()

How

Select Extension Method - Declaration

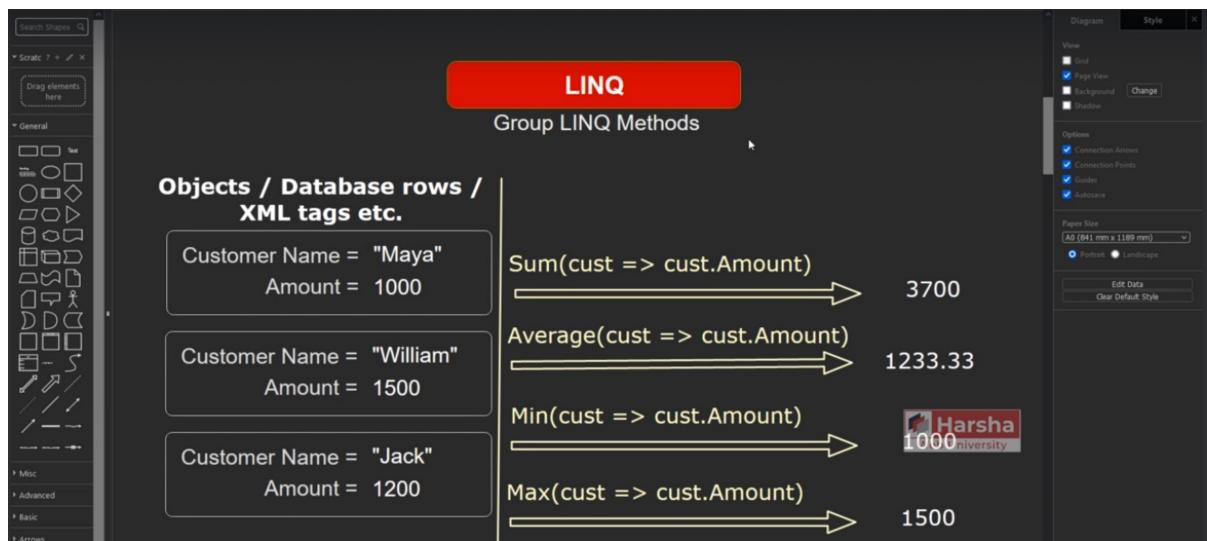
```
Select(Func<TSource, TResult> selector)
```

Select Extension Method - Usage

```
var result = Customers.Select(temp => new RegisteredCustomer()
    { CustomerName = temp.CustomerName, Location = temp.Location } );
```

//converts all customers into a collection of RegisteredCustomer class.

Min(), Max(), Count(), Sum(), Average()



Classification	LINQ Extension Methods / LINQ Operators
Filtering	Where, OfType
Sorting	OrderBy, OrderByDescending, ThenBy, ThenByDescending, Reverse
Grouping	GroupBy
Join	Join
Project	Select, SelectMany
Aggregation	Average, Count, Max, Min, Sum
Quantifiers	All, Any, Contains
Elements	ElementAt, ElementOrDefault, First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault
Set Operations	Distinct, Except, Intersect, Union
Partitioning	Skip, SkipWhile, Take, TakeWhile
Concatenation	Concat
Equality	SequenceEqual
Generation	DefaultEmpty, Empty, Range, Repeat
Conversion	AsEnumerable, AsQueryable, Cast, ToArray, ToDictionary, ToList