LINQ

**1. Select() vs. SelectMany()**

**➤ Select(): Projects each element in a collection**

csharp

CopyEdit

var names = customers.Select(c => c.Name);

* Input: List<Customer>
* Output: List<string> (one per customer)

**➤ SelectMany(): Flattens nested collections (used for one-to-many)**

csharp

CopyEdit

var allOrders = customers.SelectMany(c => c.Orders);

* Input: List<Customer>
* Each Customer has List<Order>
* Output: List<Order> (flattened from all customers)

🔁 **It flattens**: turns List<List<T>> → List<T>

✅ Use when you want to access child collections across multiple parents.

**🔍 2. First() vs. FirstOrDefault()**

**➤ First()**

Returns the **first matching element**.  
Throws exception if no match found.

csharp

CopyEdit

var firstNoidaCustomer = customers.First(c => c.City == "Noida");

**➤ FirstOrDefault()**

Returns the **first matching element** or null (for reference types) if not found.

csharp

CopyEdit

var maybeCustomer = customers.FirstOrDefault(c => c.City == "Gurgaon");

✅ Use FirstOrDefault() when not sure if a match exists.

**🔍 3. Single() vs. SingleOrDefault()**

**➤ Single()**

Returns the **only matching element**.  
Throws:

* If none found
* If **more than one match** found

csharp

CopyEdit

var exactCustomer = customers.Single(c => c.CustomerId == 1);

**➤ SingleOrDefault()**

Same as Single(), but returns null (or default) if no match is found.

Throws:

* If **multiple matches** exist

csharp

CopyEdit

var oneOrNone = customers.SingleOrDefault(c => c.CustomerId == 999);

✅ Use SingleOrDefault() when expecting **0 or 1** result.

**🧩 Other Important LINQ Functions**

**🔹 Where()**

Filters based on a condition

csharp

CopyEdit

var delhiCustomers = customers.Where(c => c.City == "Delhi");

**🔹 OrderBy() / OrderByDescending()**

Sorts elements

csharp

CopyEdit

var sorted = customers.OrderBy(c => c.Name);

**🔹 GroupBy()**

Groups elements by a key

csharp

CopyEdit

var groupedByCity = customers.GroupBy(c => c.City);

**🔹 Any() / All()**

csharp

CopyEdit

bool hasNoidaCustomer = customers.Any(c => c.City == "Noida");

bool allFromDelhi = customers.All(c => c.City == "Delhi");

**🔹 Count() / Sum() / Average()**

csharp

CopyEdit

int orderCount = customers.SelectMany(c => c.Orders).Count();

int totalQty = customers.SelectMany(c => c.Orders)

.SelectMany(o => o.Items)

.Sum(i => i.Quantity);

**🔹 Distinct()**

csharp

CopyEdit

var uniqueCities = customers.Select(c => c.City).Distinct();

**🔹 Take(n) / Skip(n)**

csharp

CopyEdit

var top3 = customers.Take(3);

var skip2 = customers.Skip(2);

**✅ Summary Table**

| **Function** | **Returns** | **Throws Exception** | **Use When** |
| --- | --- | --- | --- |
| First() | First match | If none | At least one match guaranteed |
| FirstOrDefault() | First match or default | No | Match may or may not exist |
| Single() | Exactly one match | If 0 or >1 | Expecting exactly one match |
| SingleOrDefault() | One or none | If >1 | Expecting 0 or 1 match |
| Select() | One-to-one projection | No | Project specific field |
| SelectMany() | Flattens nested collections | No | Combine child collections |
| Where() | Filters collection | No | Multiple matching elements |