Operator	Definition	Example
1. Select	কোন একটা variable বা object কে select করতে ব্যবহার হয়।	List<int> numbers=new List<int>(){1,2,3,4,5}; Var methodSyntax=(from num in numbers select num).ToList(); এখানে num , numbers এর সব গুলো উপাদান কে represent করছে। foreach লুপ এর মত numbers এর সব value গুলো এখন num এর মধ্যে আছে। এখন সেই num কে select করে লিস্ট এ কনভার্ট করলাম। তাহলে 1,2,3,4,5 এর একটা লিস্ট পাবো।

2. SelectMany	এটার ব্যবহার একটা উদাহরন দিয়ে ভালো ভাবে বুঝা যাবে। ধরি অনেক কয়টা Biscuit এর Packet আছে। সব গুলো Biscuit এর packet ধরে Biscuit বের করে একটা বড় packet এ রাখবো।	Example 1:
		List<string> names = new List<string>()
		{"Zakir","Zahid" };
		
		var query = (from name in names
		from ch in name
		select ch).ToList();
		
		var method = names.SelectMany(ch => ch).ToList();
		
		এখানে প্রথমে Zakir, Zahid ওয়ার্ড ২ টা কে ধরবে সেখান থেকে একটা একটা করে character গুলো কে বের করে লিস্ট বানিয়ে query or method এ রাখবে। তখন result হবে এমন [Z,a,k,i,r,Z,a,h,i,d].
		
		Example 2:
		List<Employee> employee = new
		List<Employee>()
		{
		new Employee(){Id=0, Name="Zakir", sector=new
		List<string>(){"Front-End","Back-End","Database" }
		},
		
		new Employee(){Id=1, Name="Keya", sector=new
		List<string>(){ "Front-End","Mobile","Server" } },
		
		new Employee(){Id=2, Name="Jewel", sector=new
		List<string>(){ "Mobile","Marketing","UI" } } };

		var methodResult = employee.SelectMany(x => x.sector).ToList(); এখানে SelectMany প্রতিটা Sector কে ধরে একটা লিস্ট এ কনভার্ট করবে। Result হবে। ["Front-End","Back-End","Database","Front-End","Mobile","Server","Mobile","Marketing","UI"]
3. Where	Where operator মেইনলি ব্যবহার হয় condition apply করার জন্য।	List<string> names = new List<string>() {"Zakir","Keya","Zahid","Jahangir" }; var query = (from name in names where name.Length > 5 select name).ToList(); যে সকল value এর length , 5 অপেক্ষা বড় হবে শুধু সেগুলাকেই select করবে।

4. OfType	একটা নির্দিষ্ট Data Type কে select করতে ব্যবহার হয়।	List<Object> MyObject = new List<Object>() { "Zakir", "Zahid", "Limon", "Pranto", 1, 2, 3, 4, 5.5, 6.3, 7.4, 8, 9, 10 };
		
		//Query Syntax
		var findString = (from str in MyObject
		where str is string
		select str).ToList();
		
		var stringFind = MyObject.OfType<string>().ToList();
		
		এখানে MyObject লিস্ট থেকে শুধু string type ই select হয়ে লিস্ট হবে।

5. OrderBy	নির্দিষ্ট কোন sequence এ সাজানোর জন্য ব্যবহার হয়।	List<int> numbers = new List<int>() {1,4,10,5,6,7,8,3,9,2 };
		
		var query=(from number in numbers
		orderby number
		select number).ToList();
		
		var method = numbers.OrderBy(number => number).ToList();

		ছোট থেকে বড় মানে 1,2,3,4---------10 এভাবে সাজাতে OrderBy ব্যবহার হয়েছে।

6. OrderByDescending	OrderBy এর মতই। শুধু উল্টা।	List<int> numbers = new List<int>() {1,4,10,5,6,7,8,3,9,2 };
		
		var query=(from number in numbers
		orderby number descending
		select number).ToList();
		
		var method = numbers.OrderByDescending(number => number).ToList();
		
		বড় থেকে ছোট মানে 10,9,8,-------1 এভাবে সাজাতে OrderByDescending ব্যবহার হয়েছে।

7. ThenBy	ধরি একজনের FirstName=Zakir & LastName=Hossain এবং অন্যজনের FirstName=Zakir & LastName=Rahman তাহলে FirstName কে OrderBy করে সাজাবো এবং LastName কে ThenBy এর সাহায্যে সাজাবো। ThenBy অনেক গুলা হতে পারে।	List<Student> student = new List<Student>() { new Student(){Id=0, FirstName="Zakir", LastName="Hossain", Age=25, Email="zakir@gmail.com"}, new Student(){Id=1, FirstName="Keya", LastName="Rahman", Age=24, Email="keya@gmail.com"}, new Student(){Id=2, FirstName="Zakir", LastName="Rahman", Age=23, Email="zakir@gmail.com"}, new Student(){Id=3, FirstName="Keya", LastName="Hossain", Age=35, Email="keya@gmail.com"}, new Student(){Id=4, FirstName="Zakir", LastName="Hossain", Age=15, Email="zakir@gmail.com"}, new Student(){Id=5, FirstName="Keya", LastName="Rahman", Age=14, Email="keya@gmail.com"},

		new Student(){Id=6, FirstName="Zakir",
		LastName="Rahman", Age=13,
		Email="zakir@gmail.com"},
		
		new Student(){Id=7, FirstName="Keya",
		LastName="Hossain", Age=15,
		Email="keya@gmail.com"},
		};
		
		//Method Syntax
		var studentMethodSingle = student.OrderBy(std =>
		std.FirstName).ThenBy(std => std.Age).ToList();
		
		//Query Syntax
		var studentQuerySingle=(from std in student
		orderby std.FirstName, std.LastName ascending
		select std).ToList();

8. ThenByDescending	ThenBy এর মতই শুধু উল্টা।	আগের মতই শুধু ThenBy এর জায়গায় ThenByDescending হবে এবং ascending এর জায়গায় descending হবে।

9. Reverse	Reverse ব্যবহার হয় কোন Data Source এর Data কে উল্টা করে সাজাতে।	List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
		
		//var IntReversed = numbers.AsEnumerable().Reverse();
		//Or
		var IntReversed = numbers.AsQueryable().Reverse();
		foreach (var item in IntReversed)
		{
		Console.WriteLine(item);
		}
		
		//Mixed Syntax
		var querySyntax = (from num in numbers
		select num).Reverse();
		
		লিস্ট এ কনভার্ট করার প্রয়োজন নাই। এমনিতেই লিস্ট এ কনভার্ট হয়ে যাবে।

10. All	All চেক করে সবগুলা একি condition এ আছে কিনা।	List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }; var isAll = numbers.All(x => x % 2 == 0); Console.WriteLine($"All= {isAll}"); যদি numbers এর সবগুলা উপাদান ২ দ্বারা বিভাজ্য হয় তাহলে print হবে True আর না হলে False. এখানে False print হবে।
11. Any	Any চেক করে যেকোনো একটা উপাদান নির্দিষ্ট কোন condition এ আছে কিনা।	List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }; var isAny = numbers.Any(x => x % 2 == 0); Console.WriteLine($"Any={isAny}"); যদি numbers এর যেকোনো একটা উপাদান ২ দ্বারা বিভাজ্য হয় তাহলে print হবে True আর না হলে False. এখানে True print হবে।
12. Contains	Contain মানে থাকা	List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }; var isContains = numbers.Contains(2); Console.WriteLine($"Contains={isContains}"); যদি numbers এ 2 থাকে তাহলে print হবে True আর না থাকলে print হবে False. এখানে print হবে True কারন numbers এ 2 আছে।

13. Distinct	Distinct ডুপ্লিকেট উপাদান কে বাদ দেয়। অনেক্ টা math এর Set{ } এর মতো।	List<string> letter1 = new List<string>()
		{"A","B","C","D","Z","K","A","B","C","D" };
		
		var msd = letter1.Distinct().ToList();
		var qsd = (from lett in letter1
		select lett).Distinct().ToList();
		
		এটা letter1 থেকে duplicate A,B,C,D বাদ দিয়ে [“A”,”B”,”C”,”D”,”Z”,”K”] লিস্ট তৈরি করবে।

14. Except	কোন একটা উপাদান ছাড়া বাকি গুলা	List<string> letter1 = new List<string>() {"A","B","C","D","Z","K","A","B","C","D" };

	বাছাই করতে ব্যবহার হয়।	List<string> letter2 = new List<string>() {"A","B","C","D" };
		
		var mse = letter1.Except(letter2).ToList();
		
		var qse = (from lett in letter1
		select lett).Except(letter2).ToList();
		
		letter1 এর মধ্যে যে সকল উপাদান আছে টা যদি letter2 এর মদ্ধেও থাকে তাহলে সেই উপাদান গুলি বাদ দিয়ে বাকি গুলা নিয়ে লিস্ট তৈরি হবে।
		যেমন এখানে ["Z","K"] তৈরি হবে।

15. Intersect	দুইটা Data Source এর যে উপাদান গুলা একই বা Common হবে সেগুলা বাছাই হবে।	List<string> letter1 = new List<string>()
		{"A","B","C","D","Z","K","A","B","C","D" };
		
		List<string> letter2 = new List<string>() {"A","B","C","D" };
		
		var msi = letter1.Intersect(letter2).ToList();
		
		var qsi = (from lett in letter1
		select lett).Intersect(letter2).ToList();
		
		letter1 ও letter2 এর মধ্যে A,B,C,D common এগুলার লিস্ট তৈরি হবে।

16. Union	দুইটা Data Source এর উপাদান গুলির মিলের একটা এবং অমিলের সব কয়টা বাছাই হবে।	List<string> letter1 = new List<string>()
		{ "A","B","C","D","Z","K","A","B","C","D" };
		
		List<string> letter2 = new List<string>() {"A","B","C","D" };
		
		var msu = letter1.Union(letter2).ToList();
		
		var qsu = (from lett in letter1
		select lett).Union(letter2).ToList();
		
		["A","B","C","D","Z","K"] লিস্ট তৈরি হবে।

17. Take	কোন একটা Data Source এর নির্দিষ্ট index পর্যন্ত উপাদান বাছাই করতে ব্যবহার	List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }; var methodTake = numbers.Take(5).ToList();

	হয়। Take মানে নেয়া।	var queryTake = (from num in numbers select num).Take(5).ToList(); numbers লিস্ট থেকে প্রথম 5 টা উপাদান নিয়ে লিস্ট বানাবে। [1, 2, 3, 4, 5]

18. TakeWhile	একটা condition দিয়ে দিতে হবে। index এর যে উপাদান এ গিয়ে condition টা False হয়ে যাবে ততদুর পর্যন্ত উপাদান বাছাই হবে। অবশ্যই condition এ ">" ব্যবহার করা যাবে না কারন বাছাই শুরু হয় শুন্য (Zero) index থেকে। TakeWhile= Untill	List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
		
		var methodTakeWhile = numbers.TakeWhile(x=> x<5).Select(x=>x).ToList();
		
		var queryTakeWhile = (from num in numbers
		select num).TakeWhile(x => x < 5).ToList();
		
		প্রথম index থেকে X বা উপাদান নেয়া শুরু হবে। যখন উপাদানের মান 5 হবে তখন উপাদান নেয়া বন্ধ হয়ে জাবে।মানে 1 থেকে 4 পর্যন্ত নিয়ে লিস্ট বানাবে । [1, 2, 3, 4]

19. Skip	যত index পর্যন্ত skip করতে চাই সেটা লিখতে হবে। where ব্যবহার করতে চাইলে skip এর পূর্বে লিখতে হবে।	List<int> numbers = new List<int>() {1,2,3,4,5,6,7,8,9,10};
		
		var methodSkip = numbers.Skip(3).ToList();
		
		var querySkip = (from num in numbers
		select num).Skip(3).ToList();
		প্রথম 3 টা উপাদান skip করে 4 থেকে 10 পর্যন্ত নিয়ে লিস্ট বানাবে। [4,5,6,7,8,9,10]

20. SkipWhile	একটা condition দিতে হবে। যে উপাদানটির জন্য condition টা মিথ্যা হয়ে যাবে তার পূর্বের উপাদান গুলা skip হবে।	List<string> names = new List<string>() {"Zakir","Keya","Zahid","Mi","Setu" };
		
		var methodSkipWhile = names.SkipWhile((value, index) => value.Length > index).ToList();
		
		var querySkipWhile = (from nam in names
		select nam).SkipWhile((value,index)=> value.Length>index).ToList();
		Zakir এর Length 5 এবং index 0 . 5>0 (True)
		Keya এর Length 4 এবং index 1 . 4>0 (True)
		Zahid এর Length 5 এবং index 2 . 5>2 (True)
		Mi এর Length 2 এবং index 3 . 2>3 (False)

		তাই Mi এর পূর্বের Zakir, Keya, Zahid Skip হবে। [Mi,Setu] লিস্ট তৈরি হবে।

21. Join	২ অথবা তার অধিক টেবিল কে যুক্ত (Inner Join) করতে Join ব্যবহার হয়।	List<Student> student = new List<Student>()
		{
		new Student(){Id=0, Name="Zakir", StudentId=1},
		new Student(){Id=1, Name="Keya", StudentId=2},
		new Student(){Id=2, Name="Rozina", StudentId=3},
		new Student(){Id=3, Name="Rakib", StudentId=4},
		new Student(){Id=4, Name="Limon", StudentId=5}
		};
		
		List<Address> address =new List<Address>()
		{
		new Address(){Id=0, HomeAddress="Kumargara Kheyaghat", AddressId=1},
		new Address(){Id=1, HomeAddress="Bahadurpur Mothurapur", AddressId=2},
		new Address(){Id=2, HomeAddress="Choto Bishakol", AddressId=3},
		new Address(){Id=3, HomeAddress="Habiganj Sylhet", AddressId=4},
		new Address(){Id=4, HomeAddress="Narikel Para Chatmohar", AddressId=5} };
		
		List<Marks> mark = new List<Marks>()
		{
		new Marks(){Id=0, Mark=70, MarkId=1},
		new Marks(){Id=1, Mark=80, MarkId=2},
		new Marks(){Id=2, Mark=90, MarkId=3},
		new Marks(){Id=3, Mark=60, MarkId=4},
		new Marks(){Id=4, Mark=50, MarkId=5}
		};
		
		var methodTwoTableJoin = student.Join(address,
		std => std.StudentId,
		addr => addr.AddressId,
		(std, addr) => new { StudentName=std.Name,
		StudentAddress=addr.HomeAddress}).ToList();
		
		var queryTwoTableJoin = (from std in student

		join addr in address
		on std.StudentId equals addr.AddressId
		select new {
		StudentName=std.Name,
		StudentAddress=addr.HomeAddress}).ToList();
		
		//Join Three Table
		
		var joinThreeTable = (from std in student
		join addr in address
		on std.StudentId equals addr.AddressId
		join mrk in mark
		on std.StudentId equals mrk.MarkId
		select new {
		StudentName = std.Name,
		StudentAddress = addr.HomeAddress,
		StudentMark=mrk.Mark}).ToList();

22. GroupJoin	একটা টেবিল এ উপাদান থাকবে এবং অন্য টেবিল এ Group Name থাকবে। সেই Group এর আন্ডার এ উক্ত উপাদান গুলো কে বাছাই করে যুক্ত করতে GroupJoin ব্যবহার হয়।	List<Student> students = new List<Student>()
		{
		new Student(){Id=0, Name="Zakir", GroupId=1},
		new Student(){Id=1, Name="Zahid", GroupId=1},
		new Student(){Id=2, Name="Limon", GroupId=2},
		new Student(){Id=3, Name="Pranto", GroupId=2},
		new Student(){Id=4, Name="Rakib", GroupId=3},
		new Student(){Id=5, Name="Shuvo", GroupId=3},
		new Student(){Id=6, Name="Shovon"} };
		
		List<Category> categories = new List<Category>()
		{
		new Category(){Id=0, Group="Group A", StudentId=1},
		new Category(){Id=1, Group="Group B", StudentId=2},
		new Category(){Id=2, Group="Group C", StudentId=3},
		new Category(){Id=3, Group="Group D", StudentId=4}
		};
		var MethodGroup = categories.GroupJoin(students,
		cat => cat.StudentId,
		std => std.GroupId,
		(cat, std) => new { cat, std });
		
		var QueryGroup = from cat in categories
		join std in students

		on cat.StudentId equals std.GroupId into StudentVar select new { cat, StudentVar };

23. LeftJoin	এটা Operator নয়। GroupJoin দিয়েই করতে হয়। প্রথম টেবিল এর সব ডাটা নেবো আর পরের টেবিল থেকে শুধু মিলের গুলা নিয়ে Join করাই LeftJoin ।	List<Student> students = new List<Student>()
		{
		new Student(){Id=0, Name="Zakir", GroupId=1},
		new Student(){Id=1, Name="Zahid", GroupId=2},
		new Student(){Id=2, Name="Limon", GroupId=3},
		new Student(){Id=3, Name="Pranto", GroupId=4},
		new Student(){Id=4, Name="Rakib", GroupId=5},
		new Student(){Id=5, Name="Shuvo", GroupId=6},
		new Student(){Id=6, Name="Shovon", GroupId=7}};
		
		List<Category> categories = new List<Category>()
		{
		new Category(){Id=0, Group="Group A", StudentId=1},
		new Category(){Id=1, Group="Group B", StudentId=2},
		new Category(){Id=2, Group="Group C", StudentId=3},
		new Category(){Id=3, Group="Group D", StudentId=4}
		};
		
		var QueryLeftJoin = (from std in students
		join cat in categories
		on std.GroupId equals cat.StudentId into StudentGroups
		from stdgroup in StudentGroups.DefaultIfEmpty()
		select new { StudentName = std.Name, StudentGroup = stdgroup != null ? stdgroup.Group : "NA" }).ToList();
		
		foreach (var item in QueryLeftJoin) {
		Console.WriteLine($"StudentName={item.StudentName} Group={item.StudentGroup}");}

24. ElementAt	কোন একটা Data Source এ index number দিয়ে Element বা উপাদান বের করতে এটা ব্যবহার হয়। কিন্তু যদি index number এ কোন উপাদান না থাকে তাহলে Error দেখাবে।	List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
		
		//Method Syntax
		var FourthElement = numbers.ElementAt(3);
		Console.WriteLine(FourthElement);
		
		//Query Syntax
		var queryElementAt = (from num in numbers
		select num).ElementAt(3);

		Console.WriteLine(queryElementAt); লিস্ট থেকে 3 নাম্বার index এর Element বা উপাদান নিয়ে Print করবে। মানে 4 print করবে।

25. ElementAtDefault	ElementAt , Error দেখায়। কিন্তু ElementAtDefault কোন Error দেখায় না। Index Number এ কোন উপাদান না থাকলে শূন্য রিটার্ন করে।	List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
		
		//Method Syntax
		var EleventhElement= numbers.ElementAtOrDefault(10);
		Console.WriteLine(EleventhElement);
		
		//Query Syntax
		var queryElementAtOrDefault = (from num in numbers
		select num).ElementAtOrDefault(10);
		Console.WriteLine(queryElementAtOrDefault);
		
		10 নাম্বার index এ কোন উপাদান নেই। তাই Error Show করার কথা কিন্তু ElementAtDefault ব্যবহার করার কারনে শূন্য (Zero) Print করবে।

26. First	কোন একটা Data Source এর প্রথম উপাদান বের করতে First ব্যবহার হয়। কিন্তু এমন কোন condition যদি দেয়া হয় যার কারনে কোন উপাদান না থাকে যেমনঃ কোন একটা লিস্ট এ 10 পর্যন্ত ডাটা আছে কিন্তু বলে দিলাম ডাটা 10 এর চাইতে বড় হবে তাহলে First Operator কোন ডাটা খুজে পাবে না তাই Error দেখাবে।	List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
		
		var num = numbers.First();
		Console.WriteLine(num);
		
		প্রথম উপাদান মানে 1 print হবে।
		
		var num2 = numbers.Where(x => x > 5).First();
		Console.WriteLine(num2);
		
		5 এর চাইতে বড় উপাদান শুরু 6 থেকে । তাই এখানে 6 print হবে।

27. FirstOrDefault	First এর সমস্যা সমাধান করে FirstOrDefault . উপাদান না থাকলে শূন্য রিটার্ন করে।	List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }; var num4 = numbers.FirstOrDefault(x => x == -1); Console.WriteLine(num4);

		এখানে -1 কোন উপাদানই নেই। তাই Error Show করার কথা। কিন্তু FirstOrDefault বাবহারের কারনে শূন্য(Zero) print হবে।
28. Last , LastOrDefault	First, FirstOrDefault এর মতই। শুধু Last উপাদান নিয়ে কাজ করবে।	
29. Single	কোন একটা Data Source এ একটা মাত্র উপাদান থাকলে সেটা বের করে আনবে। এক এর অধিক উপাদান থাকলে Single ব্যবহার হয় না। যদি একটা উপাদান ও না থাকে তাহলে Error দেখাবে।	List<int> number = new List<int>() { 5 }; var num = number.Single(); Console.WriteLine(num); Data Source এ একটা মাত্র উপাদান আছে তাই সেটা print হবে ।
30. SingleOrDefault	Data Source এ কোন ডাটা বা উপাদান না থাকলে Error দেখানোর পরিবর্তে শূন্য(Zero) দেখাবে।	List<int> numbers = new List<int>() { }; var nums = numbers.SingleOrDefault(); Console.WriteLine(nums); Data Source এ কোন উপাদান নেই তাই শূন্য (Zero) print হবে।