***LINQ Notes 3***

**Group Join**

**Group Join**

var groupJoin = suppliers.GroupJoin(buyers,

s => s.District,

b => b.District,

(s, buyersGroup) => new

{

s.Name,

s.District,

Buyers = buyersGroup

});

foreach(var grp in groupJoin)

{

Console.WriteLine("{0} District:{1}", grp.Name, grp.District);

foreach( var item in grp.Buyers )

{

Console.WriteLine(" {0}", item.Name);

}

}

Equivalent

var matching = from s in suppliers

orderby s.District

join b in buyers on s.District equals b.District into buyersGroup

select new

{

s.Name,

s.District,

Buyers = buyersGroup

};

Change To

var groupJoin = suppliers.GroupJoin(buyers,

s => s.District,

b => b.District,

(s, buyersGroup) => new

{

s.Name,

s.District,

Buyers = buyersGroup.OrderBy(b => b.Name)

});

**Filtering by Type**

object[] mix = { 1, "string", 'd', new List<int>() { 1, 2, 3, 4 }, new List<int>() { 5, 2, 3, 4 }, "dd", 's', "Hello Kitty", 1, 2, 3, 4, };

Extract all integers from this array

var allIntegers = mix.OfType<int>();

Console.WriteLine(string.Join(", ",allIntegers));

var suppliers = from p in people

where p is Supplier

select p;

var suppliers2 = people.Select( p => p ).Where( p => p is Supplier );

foreach( var item in suppliers )

{

Console.WriteLine(item.GetType().ToString());

}

Change to

var suppliers = from p in people

where p is Supplier

where (p as Supplier).Age == 20

select p;

var suppliers2 = people.Select( p => p ).Where( p => p is Supplier ).Where( p => (p as Supplier).Age == 20 );

var suppliers = from p in people

where p is Supplier

let s = p as Supplier

where s.Age > 19 && s.Age < 25

select p;

var suppliers2 = people.Select(p => p).Where(p => p is Supplier).Select(p => p as Supplier ).Where(p => p.Age > 19 && p.Age < 25);

can also do the following:

var supp = people.OfType<Buyer>().Where(b => b.Age == 20 );

**Filtering with Where and things to consider**

Performance is better when conditions are in a single where clause.

var buyers = from p in people2

where p is Buyer2

let b = p as Buyer2

where b.Age == 20

where b.ID < 5

where b.Height > 100

where b.Weight > 50

where b.Height < 80

select b;

var buyers2 = from p in people2

where p is Buyer2

let b = p as Buyer2

where (b.Age == 20 && b.ID < 5) && (b.Height > 100 || (b.Weight > 50 && b.Height < 80))

select b;

var buyers3 = people2.OfType<Buyer2>().Where(b => (b.Age == 20 && b.ID < 5) && (b.Height > 100 || (b.Weight > 50 && b.Height < 80)));

**Converting**

var toCollection = from p in people2

// Some gnarly query

select p;

Person[] personArray = toCollection.ToArray<Person>();

List<Person> personList = toCollection.ToList<Person>();

var buyersToSuppliers = people2.OfType<Buyer2>().ToList().ConvertAll(b => new Supplier { Age = b.Age });

*Use debugger to examine*

var buyersToSuppliers2 = from p in people

where p is Buyer2

let b = p as Buyer2

select new Supplier

{

Age = b.Age

};

*Can convert to list or array.*

var stringNumbers = numbers.ConvertAll(n => n.ToString());

var stringNumbers2 = from n in numbers

select n.ToString();

**Ordering collections**

var orderedPeople = people3.OrderBy(p => p.Age);

foreach (var item in orderedPeople)

{

Console.WriteLine("{0}", item.Age);

}

var orderedPeople2 = people3.OrderBy(p => p.ID).OrderBy(p => p.Age);

foreach (var item in orderedPeople)

{

Console.WriteLine("{0} - {1}", item.Age, item.ID);

}

var orderedPeople3 = people3.OrderBy(p => p.ID).ThenBy(p => p.Age);

foreach (var item in orderedPeople3)

{

Console.WriteLine("{0} - {1}", item.Age, item.ID);

}

**Generating sequences – Enumerable.Range and Repeat**

var manyNumbers = Enumerable.Repeat(5,50);

Console.WriteLine(string.Join(",",manyNumbers));

var numbers5 = Enumerable.Range(1, 50);

Console.WriteLine(string.Join(",", numbers5));

var numbers6 = Enumerable.Range(1, 10).Where(n => n % 2 == 1);

Console.WriteLine(string.Join(",", numbers6));

var evenNumbers = from n in Enumerable.Range(1, 10)

where n % 2 == 0

select n;

var squared = Enumerable.Range(1, 10).Select(n => n \* n);

Console.WriteLine(string.Join(",", squared));

Random rnd = new Random();

var randoms = Enumerable.Range(1, 10).Select(\_ => rnd.Next() % 10);

Console.WriteLine(string.Join(",", randoms));

var alphabet = Enumerable.Range(0, 26).Select(c => (char)(c + 'a') );

Console.WriteLine(string.Join(",", alphabet));

**Checking collections for equality**

Console.WriteLine(catNames == catNames2);

Console.WriteLine(Equals(catNames, catNames2));

Console.WriteLine(catNames.Equals(catNames2));

Console.WriteLine(catNames.SequenceEqual(catNames2));

**Set Operations – Distinct, Intersect, Union, Except**

var distinct = st1.Distinct();

foreach( var item in distinct )

{

Console.WriteLine("{0}", item);

}

Console.WriteLine("--------");

var distinct2 = st2.Distinct();

foreach (var item in distinct2)

{

Console.WriteLine("{0}", item);

}

Console.WriteLine("--------");

var intDistinct = ints.Distinct();

foreach (var item in intDistinct)

{

Console.WriteLine("{0}", item);

}

Console.WriteLine("--------");

var intersect = st1.Intersect(st2);

foreach (var item in intersect)

{

Console.WriteLine("{0}", item);

}

Console.WriteLine("--------");

var intIntersect = ints.Intersect(ints2);

foreach (var item in intIntersect)

{

Console.WriteLine("{0}", item);

}

Console.WriteLine("--------");

var union = st1.Union(st2);

foreach (var item in union)

{

Console.WriteLine("{0}", item);

}

Console.WriteLine("--------");

var intUnion = ints.Union(ints2);

foreach (var item in intUnion)

{

Console.WriteLine("{0}", item);

}

Console.WriteLine("--------");

var except = st1.Except(st2);

foreach (var item in except)

{

Console.WriteLine("{0}", item);

}

Console.WriteLine("--------");

var intExcept = ints.Except(ints2);

foreach (var item in intExcept)

{

Console.WriteLine("{0}", item);

}

Console.WriteLine("--------");

var intExcept2 = ints2.Except(ints);

foreach (var item in intExcept2)

{

Console.WriteLine("{0}", item);

}

Console.WriteLine("--------");

var except2 = st2.Except(st1);

foreach (var item in except2)

{

Console.WriteLine("{0}", item);

}

**Partitioning Operations**

int[] ints4 = ints3.Skip(10).ToArray();

Console.WriteLine(string.Join(",", ints4));

int[] ints5 = ints3.Take(5).ToArray();

Console.WriteLine("Take(5) " + string.Join(",", ints5));

int[] ints6 = ints3.Skip(8).Take(5).ToArray();

Console.WriteLine("Skip(8).Take(5) " + string.Join(",", ints6));

int[] ints7 = ints3.OrderByDescending(i => i).Take(7).ToArray();

Console.WriteLine("OrderByDescending(i => i).Take(7) " + string.Join(",", ints7));

int[] ints8 = ints3.SkipWhile(i => i < 5).ToArray();

Console.WriteLine("SkipWhile(i=>i<5) " + string.Join(",", ints8));

int[] ints9 = ints3.TakeWhile(i => i < 5).ToArray();

Console.WriteLine("TakeWhile(i => i < 5) " + string.Join(",", ints9));

int[] ints10 = ints3.SkipWhile(i => i < 3).TakeWhile(i => i < 5 ).ToArray();

Console.WriteLine("SkipWhile(i => i < 3).TakeWhile(i => i < 5 ) " + string.Join(",", ints10));

**Quantifying Operations**

Console.WriteLine(ints3.All(i => i > 0));

Console.WriteLine(st1.All(c => c < 'f'));

Console.WriteLine(ints3.Any(i => i > 5));

Console.WriteLine(st1.Any(c => c == 'a'));

Console.WriteLine(st1.Any());

string empt = String.Empty;

Console.WriteLine(empt.Any());

Console.WriteLine(ints3.Contains(3));

Console.WriteLine(st1.Contains('i'));

**Concatenation**

int[] concatenated = intsb.Concat(intsc).ToArray();

Console.WriteLine("Concat(intsc) " + string.Join(",", concatenated));

int[] concatenated2 = intsa.Concat(intsb.Concat(intsc)).ToArray();

Console.WriteLine("Concat(intsb.Concat(intsc)) " + string.Join(",", concatenated2));

int[] intsd = intsa.Take(intsa.Length / 2).Concat(intsa.Skip(intsa.Length / 2).Select(i => i \* i)).ToArray();

Console.WriteLine("Take(intsa.Length / 2).Concat(intsa.Skip(intsa.Length / 2).Select(i => i \* i)) " + string.Join(",", intsd));

**Aggregation Operations**

int sumInts = intse.Aggregate((tot, i) => tot + i);

Console.WriteLine("Aggregate((t, i) => t + i) " + string.Join(",", sumInts));

// Same as

int t = 0;

for (int i = 0; i < intse.Length; i++)

{

t += intse[i];

}

int prodInts = intse.Aggregate((tot, i) => tot \* i);

Console.WriteLine("Aggregate((t, i) => t \* i) " + string.Join(",", prodInts));

// Same as

int p = 0;

for (int i = 0; i < intse.Length; i++)

{

p \*= intse[i];

}

Console.WriteLine("{0}", intse.Sum());

Console.WriteLine("{0}", intse.Average());

Console.WriteLine("{0}", people3.Sum(pp => pp.Age ));

Console.WriteLine("{0}", people3.Average(pp => pp.Age));

Console.WriteLine("{0} {1}", people3.Min(pp => pp.Age ), people3.Max(pp => pp.Age));