Notes for COP 4020 – Linq 2 – Summer 2018

**Grouping**

var genderGroup = from p in people

group p by p.Gender; // Returns IEnumerable<IGrouping

foreach( var grp in genderGroup )

{

Console.WriteLine("{0}", grp.Key);

foreach( var p in grp )

{

Console.WriteLine(" {0} {1}", p.FirstName, p.LastName );

}

}

var groupWithConditions = from p in people

where p.Age > 20

group p by p.Age;

foreach (var grp in groupWithConditions)

{

Console.WriteLine("{0}", grp.Key);

foreach (var p in grp)

{

Console.WriteLine(" {0} {1} - {2}", p.FirstName, p.LastName, p.Age);

}

}

var alphabeticalGroup = from p in people

group p by p.FirstName[0];

foreach (var grp in alphabeticalGroup)

{

Console.WriteLine("{0}", grp.Key);

foreach (var p in grp)

{

Console.WriteLine(" {0} {1}", p.FirstName, p.LastName);

}

}

var alphabeticalGroup2 = from p in people

orderby p.FirstName

group p by p.FirstName[0];

foreach (var grp in alphabeticalGroup2)

{

Console.WriteLine("{0}", grp.Key);

foreach (var p in grp)

{

Console.WriteLine(" {0} {1}", p.FirstName, p.LastName);

}

}

var = from p in people

group p by new { p.Gender, p.Age };

foreach (var grp in )

{

Console.WriteLine("{0}", grp.Key);

foreach (var p in grp)

{

Console.WriteLine(" {0} {1}", p.FirstName, p.LastName);

}

}

var orderedKeys = from p in

orderby p.Count()

select p;

foreach (var grp in orderedKeys)

{

Console.WriteLine("{0}", grp.Key);

foreach (var p in grp)

{

Console.WriteLine(" {0} {1}", p.FirstName, p.LastName);

}

}

Console.WriteLine("Gender:{0}, Age:{1}", grp.Key.Gender, grp.Key.Age );

var peopleByAge = from p in people

group p by p.Age into ageGroup

orderby ageGroup.Key

select ageGroup;

foreach( var grp in peopleByAge )

{

Console.WriteLine("{0}", grp.Key);

foreach( var p in grp )

{

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

}

}

var = from p in people

group p by new { p.Gender, p.Age } into multiKeysGroup

orderby multiKeysGroup.Count()

select multiKeysGroup;

foreach( var grp in )

{

Console.WriteLine("Gender:{0},Age:{1}", grp.Key.Gender, grp.Key.Age );

foreach( var p in grp )

{

Console.WriteLine("{0} {1}", p.FirstName, p.LastName );

}

}

var peopleWithMultipleGrouping = from p in people

let ageSelection = p.Age < 20 ?

"Young" :

p.Age >= 20 && p.Age <= 22 ?

"Adult" :

"Senior"

group p by ageSelection;

foreach (var grp in peopleWithMultipleGrouping)

{

Console.WriteLine("{0}", grp.Key);

foreach (var p in grp)

{

Console.WriteLine(" {0} {1}", p.FirstName, p.LastName);

}

}

Then add

orderby p.FirstName,p.LastName

var howManyOfEachGroup = from p in people

group p by p.Gender into g

select new {Gender=g.Key,NumPeople=g.Count()};

foreach( var item in howManyOfEachGroup )

{

Console.WriteLine("Gender:{0} ({1})", item.Gender, item.NumPeople);

}

Change to

var howManyOfEachGroup = from p in people

group p by p.FirstName[0] into g

select new {Gender=g.Key,NumPeople=g.Count()};

var howManyOfEachGroup = from p in people

group p by p.Age into g

select new {Gender=g.Key,NumPeople=g.Count()};

var shortWarriors = warriors.Where(wh => (wh.Height == 100));

var shortWarriors2 = warriors.Where(wh => (wh.Height == 100))

.Select(wh => wh.Height);

Example:

static int[] stringToIntArray(string data)

{

int[] arrayFromString = data.Split(' ')

.Select(element => int.Parse(element))

.ToArray();

return (arrayFromString);

}

**Grouping with Lambdas**

var simpleGrouping = people.GroupBy(p => p.Gender);

equivalent

var simpleGrouping = from p in people

group p by p.Gender;

var simpleGrouping = people.GroupBy(p => p.Gender);

foreach (var grp in simpleGrouping)

{

Console.WriteLine("{0}", grp.Key);

foreach (var p in grp)

{

Console.WriteLine(" {0} {1}", p.FirstName, p.LastName);

}

}

Change to

var simpleGrouping = people.Where(p => p.Age > 20)

.GroupBy(p => p.Gender);

var alphabeticalGroup = people.GroupBy(p => p.FirstName[0] );

foreach( var grp in alphabeticalGroup )

{

Console.WriteLine("{0}", grp.Key);

foreach( var p in grp )

{

Console.WriteLine(" {0} {1}", p.FirstName, p.LastName);

}

}

Change to

var alphabeticalGroup = people.OrderBy(p => p.FirstName).GroupBy(p => p.FirstName[0] );

Equavalent

var alphabeticalGroup = from p in people

orderby p.FirstName

group p by p.FirstName[0];

var multiKey = people.GroupBy(p => new { p.Gender, p.Age });

foreach( var grp in multiKey)

{

Console.WriteLine("{0}", grp.Key);

foreach( var p in grp )

{

Console.WriteLine(" {0} {1}", p.FirstName, p.LastName);

}

}

Change to

var multiKey = people.GroupBy(p => new { p.Gender, p.Age }).OrderBy(p => p.Count());

**Custom Keys**

var evenOrOddNumbers = arrayOfNumbers.GroupBy(n => (n % 2 == 0));

foreach(var grp in evenOrOddNumbers)

{

Console.WriteLine("{0}", grp.Key);

foreach( var n in grp )

{

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

}

}

Change to

var evenOrOddNumbers = arrayOfNumbers.GroupBy(n => (n % 2 == 0) ? "Even" : "Odd");

Then change to

var evenOrOddNumbers = arrayOfNumbers.OrderBy(n => n).GroupBy(n => (n % 2 == 0) ? "Even" : "Odd").OrderBy(n => n.Count());

Equivalent

var numbs = from n in arrayOfNumbers

orderby n

let evenOrOdd = (n%2==0) ? "Even" : "Odd"

group n by evenOrOdd into nums

orderby nums.Count()

select nums;

Go back to the expression

var ageGroups = people.GroupBy(p => p.Age < 20

? "Young" :

p.Age >= 20 && p.Age < 22

? "Adult" : "Senior");

var ageGroups = people.GroupBy(p =>

{

var young = p.Age < 20;

var adult = p.Age >= 20 && p.Age <= 22;

var age = young ? "Young" : adult ? "Adult" : "Senior";

return age;

});

**Using Select**

var howManyOfEachGroup = people.GroupBy(p => p.Gender)

.Select(g => new

{

Gender = g.Key,

numPeople = g.Count()

});

foreach( var grp in howManyOfEachGroup )

{

Console.WriteLine("{0}", grp.Gender);

Console.WriteLine(" {0}", grp.numPeople);

}

**Joining**

var innerJoin = suppliers.Join( buyers,

s => s.District,

b => b.District,

(s, b) => new

{

SupplierName = s.Name,

BuyerName = b.Name,

District = s.District

});

foreach( var grp in innerJoin )

{

Console.WriteLine("{0}, {1}, {2}", grp.District, grp.SupplierName, grp.BuyerName );

}

Change to

var compositeJoin = suppliers.Join(buyers,

s => new { s.District, s.Age }, // must be

b => new { b.District, b.Age }, // identical

(s, b) => new

{

SupplierName = s.Name,

BuyerName = b.Name,

District = s.District,

Age = s.Age

});

foreach(var grp in compositeJoin)

{

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

Console.WriteLine( " Supplier:{0}", grp.SupplierName );

Console.WriteLine( " Buyer:{0}", grp.BuyerName );

}

Equivalent

Var compositeJoin = from b in buyers

from s in suppliers on

new { b.District, b.Age }

equals

new { s.District, s.Age }

select new

{

Supplier = s,

BuyerName = b.Name;

};

**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)

});