

TALLER LINQ

Crear las consultas LINQ y mostrar los datos para obtener lo siguiente:

```
List<Alumno> listaAlumnos = new List<Alumno>()
{
    new Alumno("Eva",20,6.0,1,1,100),
    new Alumno("Ana" ,22,7.0,1,1,100),
    new Alumno("Rosa" ,22,4.0,1,15,100),
    new Alumno("Ot",20,3.0,1,2,101),
    new Alumno("Iu" ,30,6.8,2,3,101),
    new Alumno("Pep" ,32,6.9,3,3,101),
    new Alumno("Laia" ,30,2.3,4,4,101),
    new Alumno("Quim" ,32,1.7,4,4,101),
    new Alumno("Raul",20,6.0,5,5,102),
    new Alumno("valeria" ,25,7.0,6,6,103),//10 estudiantes
    new Alumno("valentina" ,24,4.0,7,7,104),
    new Alumno("katherine",22,3.0,8,8,105),
    new Alumno("julian" ,30,6.8,9,9,106),
    new Alumno("John" ,22,5.9,10,10,107),
    new Alumno("layla" ,27,2.3,11,11,108),
    new Alumno("Lesley" ,28,1.7,12,12,109),
    new Alumno("Melisa" ,29,6.8,13,12,109),
    new Alumno("Juan" ,36,5.9,14,13,110),
    new Alumno("Fernando" ,39,2.3,15,1,100),
    new Alumno("rafael" ,48, 7.2,14,14,110),//20 estudiantes
    new Alumno("bertha" ,36,7.0,2,15,104),
    new Alumno("erika",29,5.5,2,16,108),
    new Alumno("alberto" ,46,3.6,4,16,109),
    new Alumno("Jorge eduardo" ,39,7.1,4,16,105),
    new Alumno("rafaela" ,27,2.3,1,1,107),
    new Alumno("maria jose" ,31,3.8,2,1,100),
    new Alumno("Fabio" ,29,6.8,2,2,101),
    new Alumno("nick" ,36,5.9,2,3,103),
    new Alumno("juan esteban" ,39,2.3,2,4,101),
    new Alumno("nicolas" ,40,8.3,2,4,104),//30 estudiantes
    new Alumno("Diego",40,8.3,2,1,105),
    new Alumno("Margari",18,6.0,2,3,105),
    new Alumno("rafaela Nieto" ,27,2.3,1,1,107)
```

Sentencias Linq que utilice en los primeros 7 puntos

```
List<string> lstOrdenadosSoloNombres = (from d in listaAlumnos
                                         orderby d.Nombre.Length descending
                                         select (d.Edad + " Nombre: "+d.Nombre)).ToList();

List<Alumno> datos = (from d in listaAlumnos
                      orderby d.Nombre
                      select (d)).ToList();

var queryLongitud = from a in listaAlumnos
                    group a by a.Nombre.Length into g
                    orderby g.Key descending
                    select new
                    {
                        Nombre = g.Key,
                        Edad = g.Key,
                        Alumnos = g.Take(4)
                    };

var query = from a in listaAlumnos
            group a by a.Nota into g
            orderby g.Key descending
            select new
            {
                Nota = g.Key,
                Alumnos = g.Take(4)
            };
};
```

Esta es para agruparlos por edad

```
var queryEdad = from a in listaAlumnos
                group a by a.Edad into g
                orderby g.Key descending
                select new
                {
                    Edad = g.Key,
                    Alumnos = g.Take(4)
                };
};
```

1. Alumnos que han aprobado mayores de 30 años.

```
foreach (var dato in datos)
{
    //Console.WriteLine(dato.Edad);

    if (dato.Nota > 6)
    {
        if (dato.Edad > 30)
        {
            Console.WriteLine(dato);
        }
    }
}/*
```

Output

(Nombre-Edad-Nota)

```
C:\> Select Microsoft Visual Studio
bertha - 36 - 7
Diego - 40 - 8.3
Jorge eduardo - 39 - 7.1
nicolas - 40 - 8.3
Pep - 32 - 6.9
rafael - 48 - 7.2
```

2. Agrupar por Aprobado/Suspendido y mostrar la lista

```
//Listar por aprobados y suspendidos
foreach (var dato in datos)
{
    //Listar por aprobados
    if (dato.Nota >= 6)
    {
        Console.WriteLine("Aprobado: " + dato);
    }
}
//Listar por aprobados suspendidos
foreach (var dato in datos)
{
    if (dato.Nota < 6)
    {
        Console.WriteLine("Suspendido: " + dato);
    }
}/*
```

Output

```
Microsoft Visual Studio Debug Console

Aprobado: Ana - 22 - 7
Aprobado: berthia - 36 - 7
Aprobado: Diego - 40 - 8.3
Aprobado: Eva - 20 - 6
Aprobado: Fabio - 29 - 6.8
Aprobado: Iu - 30 - 6.8
Aprobado: Jorge eduardo - 39 - 7.1
Aprobado: julian - 30 - 6.8
Aprobado: Melisa - 29 - 6.8
Aprobado: nicolas - 40 - 8.3
Aprobado: Pep - 32 - 6.9
Aprobado: rafael - 48 - 7.2
Aprobado: Raul - 20 - 6
Aprobado: valeria - 25 - 7
Suspendido: alberto - 46 - 3.6
Suspendido: erika - 29 - 5.5
Suspendido: Fernando - 39 - 2.3
Suspendido: John - 22 - 5.9
Suspendido: Juan - 36 - 5.9
Suspendido: juan esteban - 39 - 2.3
Suspendido: katherine - 22 - 3
Suspendido: Laia - 30 - 2.3
Suspendido: layla - 27 - 2.3
Suspendido: Lesley - 28 - 1.7
Suspendido: maria jose - 31 - 3.8
Suspendido: nick - 36 - 5.9
Suspendido: Ot - 20 - 3
Suspendido: Quim - 32 - 1.7
Suspendido: rafaela - 27 - 2.3
Suspendido: Rosa - 22 - 4
Suspendido: valentina - 24 - 4
```

3. Agrupar por la longitud del nombre ordenado de mayor a menor

```
foreach (var nombre in lstOrdenadosSoloNombres)
{
    Console.WriteLine("Edad "+nombre);
}
```

Output

```
Microsoft Visual Studio Debug Co
Edad 27 Nombre: rafaela
Edad 40 Nombre: nicolas
Edad 30 Nombre: julian
Edad 28 Nombre: Lesley
Edad 29 Nombre: Melisa
Edad 48 Nombre: rafael
Edad 36 Nombre: berthha
Edad 27 Nombre: layla
Edad 29 Nombre: erika
Edad 29 Nombre: Fabio
Edad 40 Nombre: Diego
Edad 22 Nombre: Rosa
Edad 30 Nombre: Laia
Edad 32 Nombre: Quim
Edad 20 Nombre: Raul
Edad 22 Nombre: John
Edad 36 Nombre: Juan
Edad 36 Nombre: nick
Edad 20 Nombre: Eva
Edad 22 Nombre: Ana
Edad 32 Nombre: Pep
Edad 20 Nombre: Ot
Edad 30 Nombre: Iu
```

4. Agrupar por la longitud del nombre y mostrar aquellos grupos cuya suma de edades es mayor de 60

```
var sumaEdadGrupo = 0;

foreach (var item in queryLongitud)
{
    sumaEdadGrupo = 0;
    //Console.WriteLine("Edad: " + item.Edad);
    //Console.WriteLine("Numero Estu: " + item.Alumnos.Count());

    foreach (var alumno in item.Alumnos)
    {
        sumaEdadGrupo = sumaEdadGrupo + alumno.Edad;

        if (sumaEdadGrupo > 60)
        {
            Console.WriteLine("Alumnos: {0}", alumno.Nombre);
        }
    }
    if (sumaEdadGrupo > 60)
    {
        Console.WriteLine("Suma edades del grupo: " + sumaEdadGrupo + " años");
        Console.WriteLine("");
    }
}
```

Output

```
Microsoft Visual Studio Debug Console

alumnos: alberto
alumnos: rafaela
alumnos: nicolas
suma edades del grupo: 138 años

alumnos: Melisa
alumnos: rafael
suma edades del grupo: 135 años

alumnos: Fabio
alumnos: Diego
suma edades del grupo: 125 años

alumnos: Quim
alumnos: Raul
suma edades del grupo: 104 años

alumnos: Pep
suma edades del grupo: 74 años
```

5. Cuantos alumnos hay.

```
foreach (var dato in datos)
{
    contador = datos.Count();
}
Console.WriteLine("Hay "+contador+" estudiantes");
```

Output

```
Select Microsoft Visual Studio

/ 31 estudiantes
```

6. Agrupas los alumnos por notas mostrar cuantos de cada grupo y en un sub menú mostrar la lista de ellos

```
foreach (var item in query)
{
    Console.WriteLine("Nota {0}", item.Nota);
    Console.WriteLine(item.Alumnos.Count());

    foreach (var alumno in item.Alumnos)
    {
        Console.WriteLine("Alumnos: {0}", alumno.Nombre);
    }
}
```

Output

```
2
Alumnos: Rosa
Alumnos: valentina
Nota 3.8
1
Alumnos: maria jose
Nota 3.6
1
Alumnos: alberto
Nota 3
2
Alumnos: Ot
Alumnos: katherine
Nota 2.3
4
Alumnos: Laia
Alumnos: layla
Alumnos: Fernando
Alumnos: rafaela
Nota 1.7
2
Alumnos: Quim
Alumnos: Lesley
```

7. Agrupas los alumnos por Edad cuantos de cada grupo y en un sub menú mostrar la lista de ellos

```
foreach (var item in queryEdad)
{
    Console.WriteLine("Edad {0}", item.Edad);
    Console.WriteLine("Hay "+ item.Alumnos.Count()+" Alumno en este grupo");

    foreach (var alumno in item.Alumnos)
    {
        Console.WriteLine("Alumnos: {0}", alumno.Nombre);
    }
}
```

Output

```
Microsoft Visual Studio Debug Console

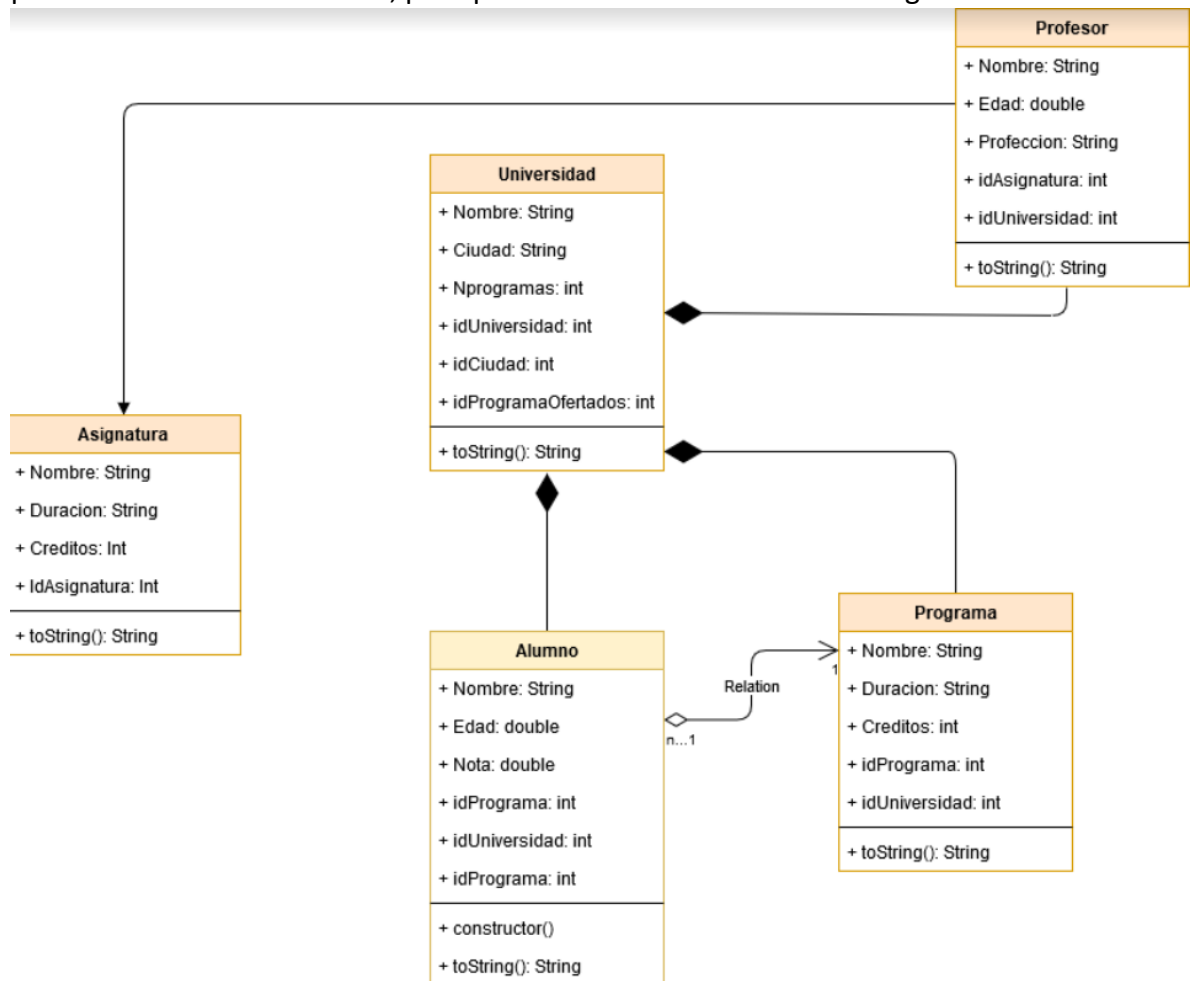
Hay 1 Alumno en este grupo
Alumnos: Lesley
Edad 27
Hay 2 Alumno en este grupo
Alumnos: layla
Alumnos: rafaela
Edad 25
Hay 1 Alumno en este grupo
Alumnos: valeria
Edad 24
Hay 1 Alumno en este grupo
Alumnos: valentina
Edad 22
Hay 4 Alumno en este grupo
Alumnos: Ana
Alumnos: Rosa
Alumnos: katherine
Alumnos: John
Edad 20
Hay 3 Alumno en este grupo
Alumnos: Eva
Alumnos: Ot
Alumnos: Raul

C:\Users\jhond\source\repos\tallerLinq\tallerLi
0.
To automatically close the console when debuggi
le when debugging stops.
Press any key to close this window . . .
```

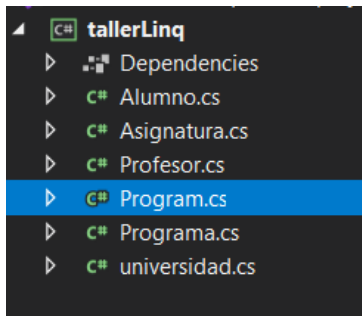

Haga el siguiente ejercicio en C#

Con la clase alumno cree una clase profesor, asignatura, programa y universidad cree las clases y los cambios de la primera clase para el ejercicio, cree el diagrama de clases y las relaciones y cree mínimo 40 consultas con Linq de las 4 clases sin incluir las primeras 7 consultas del ejercicio 1. Documente el código y cada consulta

Así hice el diagrama de clases pude hacer la ciudad de la universidad aparte, pero lo dejé hay porque por ahora solo vamos a trabajar en las consultas también se pueden hacer mas relaciones, pero para estas consultas hice este diagrama



Estas son las clases que se crearon



Listado de datos

Asignatura

```
List<Asignatura> listaAsignatura = new List<Asignatura>()
{
    new Asignatura("Matematicas", "14 semanas", 4, 1),
    new Asignatura("Fisica", "8 semanas", 3, 2),
    new Asignatura("Programacion", "15 semanas", 4, 3),
    new Asignatura("Sociales", "6 semanas", 2, 4),
    new Asignatura("Ingles", "14 semanas", 3, 5),
    new Asignatura("Redes", "14 semanas", 4, 6),
    new Asignatura("Seguridad", "16 semanas", 4, 7),
    new Asignatura("Estructuras", "14 semanas", 4, 8),
    new Asignatura("Bases de datos", "14 semanas", 4, 9),
    new Asignatura("Calculo I", "14 semanas", 3, 10), //10
    new Asignatura("Calculo II", "14 semanas", 3, 11),
    new Asignatura("Calculo III", "14 semanas", 3, 12),
    new Asignatura("Estadistica", "14 semanas", 3, 13),
    new Asignatura("web I", "14 semanas", 3, 14),
    new Asignatura("web II", "14 semanas", 3, 15), //15
    new Asignatura("Textox y discursos", "5 semanas", 1, 16),
    new Asignatura("laboratorio de fisica I", "5 semanas", 1, 2)
};
```

Profesor

```
List<Profesor> listaProfesores = new List<Profesor>()
{
    new Profesor("Julio jaramillo", "politico", 43, 4),
    new Profesor("Daniela parra", "Matematica pura", 32, 1),
    new Profesor("Stefania mendoza", "Matematica pura", 30, 1),
    new Profesor("Daniel ruiz", "Ingeniero Fisico", 28, 2),
    new Profesor("Hector calvis", "Ingeniero Fisico", 44, 2),
    new Profesor("Jaime rendon", "Ingeniero sistemas", 25, 8),
    new Profesor("oscar alfonso", "Ingeniero informatico", 36, 9),
    new Profesor("Jhonny ariztizabal", "Ingeniero de software", 38, 2),
    new Profesor("luisa fernanda", "Matematica pura", 28, 1),
    new Profesor("diego velez", "Ingeniero sistemas", 26, 6),
    new Profesor("brandon smith", "Licenciado en literatura", 39, 16),
    new Profesor("Jose smith", "licenciado en lenguas", 35, 5),
    new Profesor("Erika emilia", "Licenciada en lenguas", 39, 5),
    new Profesor("Fernando mendoza", "Ingeniero sistemas", 25, 15),
    new Profesor("Yuli orozco", "Medica", 25, 17), //15
    new Profesor("Daniela ceballos", "Medica", 30, 17)
};
```

Programas

```
List<Programa> listaProgramas = new List<Programa>()
{
    new Programa("Ingenieria en sistemas","10 semestres",180,1),
    new Programa("Ingenieria en informatica","10 semestres",179,2),
    new Programa("Ingenieria en fisica","10 semestres",160,3),
    new Programa("Matematica pura","10 semestres",175,4),
    new Programa("Ingenieria de alimentos","10 semestres",165,5),
    new Programa("Ingenieria electronica","10 semestres",178,6),
    new Programa("Ingenieria civil","10 semestres",180,7),
    new Programa("Tecnico en sistemas","4 semestres",80,8),
    new Programa("Salud ocupacional","4 semestres",75,9),
    new Programa("Medicina","10 semestres",180,10),
    new Programa("Sociales","9 semestres",152,11),
    new Programa("Lenguas modernas","10 semestres",180,12),
    new Programa("enfermeria","8 semestres",125,13),
    new Programa("Tecnologia en sistemas","6 semestres",100,14),
    new Programa("ingenieria agropecuaria","10 semestres",170,15),
    new Programa("ingenieria agropecuaria","10 semestres",165,16)
};
```

Universidad

```
List<universidad> listaUniversidades = new List<universidad>()
{
    new universidad("Universidad de caldas","Manizales",30,1,100),
    new universidad("Universidad de América","Bogota",25,2,101),
    new universidad("Politécnico Grancolombiano","Bogota",20,3,101),
    new universidad("Universidad Nacional","Bogota",50,4,101),
    new universidad("Universidad del Atlántico","Barranquilla",30,5,102),
    new universidad("Universidad del Quindio","Quindio",19,6,103),
    new universidad("Universidad de Antioquia","Medellin",40,7,104),
    new universidad("Universidad de Córdoba","Monteria",30,8,105),
    new universidad("Universidad de Manizales","Manizales",25,9,100),
    new universidad("Universidad Antonio Nariño","Quibdó",25,10,106),
    new universidad("Universidad Mariana","Pasto",24,11,107),
    new universidad("Universidad del valle","Cali",35,12,108),
    new universidad("Universidad javeriana","Bogota",50,13,101),
    new universidad("Universidad bolivariana","palmira",50,14,109),
    new universidad("Universidad autonoma de manizales","Manizales",32,15,100),
    new universidad("Universidad de boyaca","boyaca",12,16,110)
};
```

Primero llene los registros en todas las clases y le ingrese como 15 o más datos a cada uno están en el proyecto

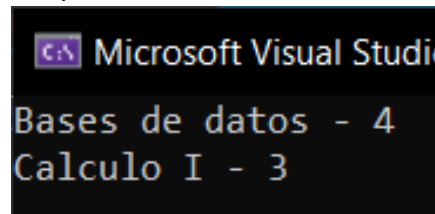
40 consultas de linq

1. Lo que hace esto es ordenar por nombre y traer los dos primeros valores take es como un limitador

```
List<string> lstOrdenadoMateria = (from d in listaAsignatura
                                   orderby d.Nombre
                                   select d.NombreYCreditos).Take(2).ToList();

foreach (var i in lstOrdenadoMateria)
{
    Console.WriteLine(i);
}
```

Output



Microsoft Visual Studio

Bases de datos - 4

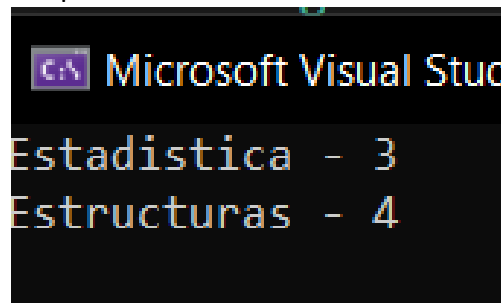
Calculo I - 3

2. Aquí con el Skip ignoran los 4 primeros y sigue con los dos siguientes

```
List<string> lstOrdenadoMateria = (from d in listaAsignatura
                                   orderby d.Nombre
                                   select d.NombreYCreditos).Skip(4).Take(2).ToList();

foreach (var i in lstOrdenadoMateria)
{
    Console.WriteLine(i);
}
```

Output



Microsoft Visual Studio

Estadística - 3

Estructuras - 4

3. Con esto agrego dos materias, pero de tipo string no es practico coger los créditos así pero sirve para probar la unión en linq

```
List<string> concatenarConUnion = new List<string>()
{
    "Geologia - 3", "Anatomia - 4"
};

List<string> lstaUnion = (from d in listaAsignatura
    orderby d.Nombre
    select d.NombreYCreditos).Union(concatenarConUnion).ToList();

foreach (var i in lstaUnion)
{
    Console.WriteLine(i);
}
```

Output

```
Bases de datos - 4
Calculo I - 3
Calculo II - 3
Calculo III - 3
Estadistica - 3
Estructuras - 4
Fisica - 3
Ingles - 3
Matematicas - 4
Programacion - 4
Redes - 4
Seguridad - 4
Sociales - 2
Textos y discursos - 1
web I - 3
web II - 3
Geologia - 3
Anatomia - 4
```

4. Ahora agregamos objetos en vez de un String esto es más práctico y lo adjuntamos y por sentencia landan obtenemos el resultado

```
List<Asignatura> concatenarConUnion = new List<Asignatura>()
{
    new Asignatura("Vias","12 semanas",3),
    new Asignatura("Medicina forence","14 semanas",3)
};

List<String> lstaUnion = (from d in listaAsignatura
                        select d)
                        .Union(concatenarConUnion)
                        .OrderBy(d => d.Nombre)
                        .Select(d => d.NombreYCreditos)
                        .ToList();
```

Output

```
Bases de datos - 4
Calculo I - 3
Calculo II - 3
Calculo III - 3
Estadistica - 3
Estructuras - 4
Fisica - 3
Ingles - 3
Matematicas - 4
Medicina forence - 3
Programacion - 4
Redes - 4
Seguridad - 4
Sociales - 2
Textox y discursos - 1
Vias - 3
web I - 3
web II - 3
```

5. Se puede hacer consultas linq dentro de una unión para hacer una mejor búsqueda y le regresa una colección

```
List<Asignatura> concatenarConUnion = new List<Asignatura>()
{
    new Asignatura("Vias","12 semanas",3),
    new Asignatura("Medicina forence","14 semanas",3)
};

List<String> lstaUnion = (from d in listaAsignatura
                          select d.NombreYCredito)
    .Union(
        from d in concatenarConUnion
        select d.NombreYCredito
    )
    .ToList();
```

Output

```
Matematicas - 4
Fisica - 3
Programacion - 4
Sociales - 2
Ingles - 3
Redes - 4
Seguridad - 4
Estructuras - 4
Bases de datos - 4
Calculo I - 3
Calculo II - 3
Calculo III - 3
Estadistica - 3
web I - 3
web II - 3
Textox y discursos - 1
Vias - 3
Medicina forence - 3
```

6. Aquí en podemos hacer 3 consultas en linq además con una union y ordenamiento

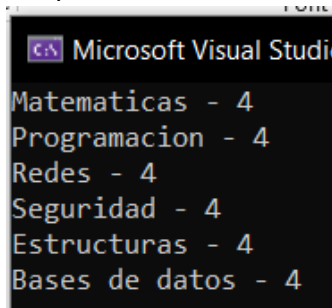
```
List<String> lstaUnion = (
    from a in
    (from d in listaAsignatura
     select d)
    .Union(
        from d in concatenarConUnion
        select d
    )
    orderby a.Nombre
    select a.NombreYCreditos
)
.ToList();
```

7. En esta consulta voy a buscar las materias de 4 créditos y me las va a listar

```
List<Asignatura> busqueda = (from d in listaAsignatura
                             where d.Creditos.Equals(4)
                             select d).ToList();

foreach (var i in busqueda)
{
    Console.WriteLine(i.NombreYCreditos);
}
```

Output



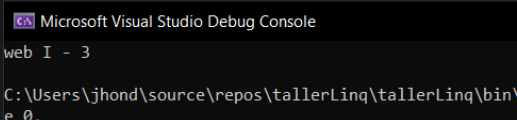
```
Microsoft Visual Studio
Matematicas - 4
Programacion - 4
Redes - 4
Seguridad - 4
Estructuras - 4
Bases de datos - 4
```

8. Con esta consulta podemos buscar las materias que están repetidas por ejemplo esta web II está dos veces repetida

```
List<Asignatura> concatenarConUnion = new List<Asignatura>()
{
    new Asignatura("Vias", "12 semanas", 3),
    new Asignatura("Medicina forense", "14 semanas", 3),
    new Asignatura("web I", "14 semanas", 3)
};

var queryRepetidos = from repetidos in listaAsignatura
                     join repetidos1 in concatenarConUnion
                     on repetidos.NombreYCreditos equals repetidos1.NombreYCreditos
                     select repetidos.NombreYCreditos;

foreach (var i in queryRepetidos)
{
    Console.WriteLine(i);
}
```



```
Microsoft Visual Studio Debug Console
web I - 3
C:\Users\jhond\source\repos\tallerLinq\tallerLinq\bin\
e 0.
```


9. Este fue uno de mis favoritos hace una búsqueda por créditos total de esas materias con esos créditos y por ultimo la cantidad de materias

```
var querysumaMaterias = from d in listaAsignatura
                        group d by d.Creditos into creditosGroup
                        select new
                        {
                            Creditos = creditosGroup.Key,
                            totalCreditos = creditosGroup.Sum(x => x.Creditos),
                            Materias = creditosGroup.Count()
                        };
```

Select Microsoft Visual Studio Debug Console

```
{ Creditos = 4, totalCreditos = 24, Materias = 6 }
{ Creditos = 3, totalCreditos = 24, Materias = 8 }
{ Creditos = 2, totalCreditos = 2, Materias = 1 }
{ Creditos = 1, totalCreditos = 1, Materias = 1 }
```

10. Esta sentencia coge la materia de créditos mas baja para el máximo solo es cambiar el Min() por el Max()

```
var queryMin = (from d in listaAsignatura
                orderby d.Nombre
                select d.Creditos).Max();
```

```
Console.WriteLine(queryMin);
```

```
var queryMin = (from d in listaAsignatura
                orderby d.Nombre
                select d.Creditos).Min();
```

11. Listar los ingenieros de sistemas mayores de 24 años

```
var queryProfe = (from d in listaProfesores
                  orderby d.Nombre
                  where d.Profession.Equals("Ingeniero sistemas")
                  select d).ToList();

foreach(var profesion in queryProfe)
{
    if(profession.Edad > 24)
    {
        Console.WriteLine(profession.Nombre + " - " + profesion.Profession);
    }
}
```

Output

```
diego velez - Ingeniero sistemas
Fernando mendoza - Ingeniero sistemas
Jaime rendon - Ingeniero sistemas
```

12. Agrupar los profesores por asignatura se unieron dos entidades la de profesor y la de asignatura con sus respectivos id y se agrupo

```
var asignaturaP = from d in listaProfesores
                  join c in listaAsignatura on d.idAsignaturaP equals c.idAsignatura
                  orderby d.idAsignaturaP
                  group d by new { c.idAsignatura, c.NombreA, d.Nombre } into grupo
                  select grupo;

foreach (var grupo in asignaturaP)
{
    Console.WriteLine(" ID: " + grupo.Key.idAsignatura+" Materia: "
        + grupo.Key.NombreA + "Profesor: "+grupo.Key.Nombre);
}
```

Output

```
ID: 1 Materia: MatematicasProfesor: Daniela parra
ID: 1 Materia: MatematicasProfesor: Stefania mendoza
ID: 1 Materia: MatematicasProfesor: luisa fernanda
ID: 2 Materia: FisicaProfesor: Daniel ruiz
ID: 2 Materia: FisicaProfesor: Hector calvis
ID: 2 Materia: FisicaProfesor: Jhonny ariztizabal
ID: 4 Materia: SocialesProfesor: Julio jaramillo
ID: 5 Materia: InglesProfesor: Jose smith
ID: 5 Materia: InglesProfesor: Erika emilia
ID: 6 Materia: RedesProfesor: diego velez
ID: 8 Materia: EstructurasProfesor: Jaime rendon
ID: 9 Materia: Bases de datosProfesor: oscar alfonso
ID: 15 Materia: web IIProfesor: Fernando mendoza
ID: 16 Materia: Textox y discursosProfesor: brandon smith
```

13. Mostrar profesores mayores que 25 y menores que 40 años

```
var rangoEdadP = (from d in listaProfesores
                  orderby d.Nombre
                  where (d.Edad > 25 && d.Edad < 40)
                  select d).ToList();

foreach (var i in rangoEdadP)
{
    Console.WriteLine(i.EdadYNombre);
}
```

Output

```
39 - brandon smith
28 - Daniel ruiz
32 - Daniela parra
26 - diego velez
39 - Erika emilia
38 - Jhonny ariztizabal
35 - Jose smith
28 - luisa fernanda
36 - oscar alfonso
30 - Stefania mendoza
```

14. Filtras materias que duren 14 semanas más su respectivo profesor
Aquí se unió y se llamo sus respectivos datos y se aplico condicionales

```
var MateriasPro = from d in listaProfesores
                  join c in listaAsignatura on d.idAsignaturaP equals c.idAsignatura
                  orderby d.idAsignaturaP
                  where c.Duracion.Equals("14 semanas")
                  group d by new { c.idAsignatura, c.NombreA, c.Duracion, d.Nombre } into grupo
                  select grupo;
```

Output

```
ID: 1 Materia: Matematicasduracion: 14 semanasProfesor: Daniela parra
ID: 1 Materia: Matematicasduracion: 14 semanasProfesor: Stefania mendoza
ID: 1 Materia: Matematicasduracion: 14 semanasProfesor: luisa fernanda
ID: 5 Materia: Inglesduracion: 14 semanasProfesor: Jose smith
ID: 5 Materia: Inglesduracion: 14 semanasProfesor: Erika emilia
ID: 6 Materia: Redesduracion: 14 semanasProfesor: diego velez
ID: 8 Materia: Estructurasduracion: 14 semanasProfesor: Jaime rendon
ID: 9 Materia: Bases de datosduracion: 14 semanasProfesor: oscar alfonso
ID: 15 Materia: web IIduracion: 14 semanasProfesor: Fernando mendoza
```

15. Este Contains es para obtener por una palabra especifica en un string yo aquí obtengo el nombre de todas las profesoras que se llaman Daniela

```
var obtenerNombresIguales = (from d in listaProfesores
                              orderby d.Nombre
                              where d.Nombre.Contains("Daniela")
                              select d).ToList();

foreach (var i in obtenerNombresIguales)
{
    Console.WriteLine("Profesora: "+i.Nombre);
}
```

Output

```
Profesora: Daniela ceballos
Profesora: Daniela parra
```

16. Aquí en este caso voy a filtrar todos los ingenieros y los médicos

```
var ingenierosYMedicos = (from d in listaProfesores
                          orderby d.Profesion
                          where d.Profesion.Contains("Ingeniero") || d.Profesion.Contains("Medica")
                          select d).ToList();

foreach (var i in ingenierosYMedicos)
{
    Console.WriteLine(": "+i.Profesion);
}
```

```
: Ingeniero de software
: Ingeniero Fisico
: Ingeniero Fisico
: Ingeniero informatico
: Ingeniero sistemas
: Ingeniero sistemas
: Ingeniero sistemas
: Medica
: Medica
```

17. Aquí lo bueno es que no hay que hacer condicionales en el Foreach si no dentro de linq

```
var ingenierosYMedicos = (from d in listaProfesores
                          orderby d.Profeccion
                          where d.Profeccion.Contains("Ingeniero") && d.Edad > 40
                          select d).ToList();

foreach (var i in ingenierosYMedicos)
{
    Console.WriteLine(": "+i.EdadYNombre);
}
```

Output

```
: 44 - Hector calvis
```

18. Aquí utilizo linq para colecciones de datos y añadir en una nueva lista y también creo una función en linq donde solo van a estar los mayores de edad estas funciones son para cuando la función landa no es soportada entonces se aplica en linq

```
var userFound = listaProfesores.Where(u => u.Edad > 18).ToList();

Func<Profesor, bool> where = new Func<Profesor, bool>((profesor) =>
{
    return profesor.Edad > 18;
});

List<Profesor> filtro = new List<Profesor>();
foreach(var profesor in listaProfesores)
{
    if (where(profesor))
    {
        filtro.Add(profesor);
    }
}
foreach (var i in filtro)
{
    Console.WriteLine(i.Nombre);
}
```

Output

```
Julio jaramillo
Daniela parra
Stefania mendoza
Daniel ruiz
Hector calvis
Jaime rendon
oscar alfonso
Jhonny ariztizabal
luisa fernanda
diego velez
brandon smith
Jose smith
Erika emilia
Fernando mendoza
Yuli orozco
Daniela ceballos
```

19. Esta es una agrupación por edad con una manera diferente utilizando `GroupBy` y `linq`

```
var groupEdades = listaProfesores.GroupBy(u => u.Edad).OrderByDescending(o => o.Key);

foreach (var edad in groupEdades)
{
    Console.WriteLine(edad.Key);
    var profesoresEdad = edad.ToList();

    foreach (var pro in profesoresEdad)
    {
        Console.WriteLine(pro.Nombre);
    }
}
```

Output

```
39
brandon smith
Erika emilia
38
Jhonny ariztizabal
36
oscar alfonso
35
Jose smith
32
Daniela parra
30
Stefania mendoza
Daniela ceballos
28
Daniel ruiz
luisa fernanda
26
diego velez
25
Jaime rendon
Fernando mendoza
Yuli orozco
```

20. Esta es parecida, pero utilizando `ToLookup` esto se utiliza para agrupar, más eficiente las listas, pero no sirve para manipular datos traídos de una base de datos solo de una lista aquí pueden ver que fue fácil de ordenar

```
var groupEdades = listaProfesores.ToLookup(u => "\n"+u.Profession).OrderByDescending(o => o.Key);

//GroupBy vs ToLookup

foreach (var edad in groupEdades)
{
    Console.WriteLine(edad.Key);
    var profesoresEdad = edad.ToList();

    foreach (var pro in profesoresEdad)
    {
        Console.WriteLine(pro.Nombre);
    }
}
```

Output

```
politico
Julio jaramillo

Medica
Yuli orozco
Daniela ceballos

Matematica pura
Daniela parra
Stefania mendoza
luisa fernanda

Licenciado en literatura
brandon smith
```

21. Unir los alumnos con su respectiva universidad y mostrarlos no imprimo la lista completa para no gastar tantas paginas

```
var AlumnoPrograma = from d in listaAlumnos
                      join p in listaProgramas
                      on d.Idprograma equals p.Idprograma
                      select (d.Nombre+" - "+p.Nombre);

foreach (var i in AlumnoPrograma)
{
    Console.WriteLine(i);
}
```

Output

```
Fernando - ingenieria agropecuaria
rafael - Tecnologia en sistemas
bertha - Ingenieria en informatica
erika - Ingenieria en informatica
alberto - Matematica pura
Jorge eduardo - Matematica pura
rafaela - Ingenieria en sistemas
maria jose - Ingenieria en informatica
Fabio - Ingenieria en informatica
nick - Ingenieria en informatica
juan esteban - Ingenieria en informatica
nicolas - Ingenieria en informatica
Diego - Ingenieria en informatica
```

22. Vamos ahora agruparlos por programa y mostrarlos lo podemos hacer de dos maneras con consultas linq o con linq y landa, es un poco complejo utilizar landa porque yo relaciono las otras consultas como si hiciera búsquedas en una base de datos SQL

```
var agruparUniversidadAlumno = from d in listaAlumnos
                                join c in listaProgramas on d.Idprograma equals c.Idprograma
                                orderby d.Idprograma
                                group d by new { c.Nombre, d.EdadYNombreAlumno } into grupo
                                select grupo;

foreach (var i in agruparUniversidadAlumno)
{
    Console.WriteLine(i.Key.Nombre+" - "+i.Key.EdadYNombreAlumno);
}
```

Output

```
Ingenieria en informatica - maria jose - 31
Ingenieria en informatica - Fabio - 29
Ingenieria en informatica - nick - 36
Ingenieria en informatica - juan esteban - 39
Ingenieria en informatica - nicolas - 40
Ingenieria en informatica - Diego - 40
Ingenieria en fisica - Pep - 32
Matematica pura - Laia - 30
Matematica pura - Quim - 32
Matematica pura - alberto - 46
Matematica pura - Jorge eduardo - 39
Ingenieria de alimentos - Raul - 20
Ingenieria electronica - valeria - 25
Ingenieria civil - valentina - 24
Tecnico en sistemas - katherine - 22
Salud ocupacional - julian - 30
```

23. Aquí lo que hice fue ordenarlo y que me los trajera en un array esto es muy útil cuando queremos hacer operaciones con números

```
var nombreOrdenadorProgramas = (from programa in listaProgramas
                                orderby programa.Idprograma
                                select programa.TotalCreditos).ToArray();

foreach (var i in nombreOrdenadorProgramas)
{
    Console.WriteLine(i.Equals(180));
}
```

Output

```
True
False
False
False
```

24. Esta de aquí es para coger todos los estudiantes y verificar si todos cumplen una condición lo vamos hacer con linq y landa el mensaje solo va aparecer una sola vez y eso es muy útil

```
bool all = listaAlumnos.All(u => u.Edad > 17 && u.Edad < 50);
if (all)
{
    Console.WriteLine("Todos cumplen");
}
else
{
    Console.WriteLine("No cumplen la condicion");
}
```

Output

```
Todos cumplen
```

25. Este es otra sentencia landa con linq que dice que exista por lo menos un usuario que tenga 18 años, Entonces yo le agregue un usuario y probé la sentencia y funciono

```
new Alumno("nicolas" ,40,8.3,2), //30 estudiantes
new Alumno("Diego" ,40,8.3,2),
new Alumno("Margari",18,6.0,3)

var any = listaAlumnos.Any(e => e.Edad == 18);
if (any)
{
    Console.WriteLine("Existe por lo menos uno que tenga 18 anios");
}
```

Output

```
Todos cumplen
Existe por lo menos uno que tenga 18 anios
```

26. Obtener el promedio de notas de los estudiantes por linq ya la Suma, Count, Min y el Max ya lo habíamos usado en consultas anteriores

```
var promedio = listaAlumnos.Average(e => e.Nota);  
Console.WriteLine(promedio);
```

Output

```
5.109375000
```

27. Esta sentencia es una búsqueda con doble select y doble where entonces de doble condición la probé en la lista programas dice los programas mayores de 150 en el numero de créditos y id del programa mayor que 0

```
var programasSelect = listaProgramas.Where(s => s.TotalCreditos > 150)  
    .Select(s => s)  
    .Where(st => st.Idprograma > 0)  
    .Select(s => s.Nombre);  
  
foreach (var programa in programasSelect)  
{  
    Console.WriteLine(programa);  
}
```

Output

```
Ingenieria en sistemas  
Ingenieria en informatica  
Ingenieria en fisica  
Matematica pura  
Ingenieria de alimentos  
Ingenieria electronica  
Ingenieria civil  
Medicina  
Sociales  
Lenguas modernas  
ingenieria agropecuaria  
ingenieria agropecuaria
```

28. Esta separa por los programas y los agrupa por el numero de semestres los mas interesante es cuando se hace la búsqueda dentro del ToList()

```
var programaGroup = from s in listaProgramas  
                    where s.Idprograma > 0  
                    group s by s.Duracion into sg  
                    orderby sg.Key  
                    select new { sg.Key, sg };  
  
foreach (var group in programaGroup)  
{  
    Console.WriteLine("\n"+group.Key);  
  
    group.sg.ToList().ForEach(st => Console.WriteLine(st.Nombre));  
}
```


Output

```
10 semestres
Ingenieria en sistemas
Ingenieria en informatica
Ingenieria en fisica
Matematica pura
Ingenieria de alimentos
Ingenieria electronica
Ingenieria civil
Medicina
Lenguas modernas
ingenieria agropecuaria
ingenieria agropecuaria

4 semestres
Tecnico en sistemas
Salud ocupacional

6 semestres
Tecnologia en sistemas

8 semestres
enfermeria

9 semestres
Sociales
```

29. Left outer join Use left outer join para mostrar a los estudiantes bajo cada estándar. Muestre el nombre del estándar incluso si no hay un alumno asignado a ese estándar. Agrupa por programa y los estudiantes que están en el

```
var studentsGroup = from stad in listaProgramas
                    join s in listaAlumnos
                    on stad.Idprograma equals s.Idprograma
                    into sg
                    select new
                    {
                        StandardName = stad.Nombre,
                        Students = sg
                    };

foreach (var group in studentsGroup)
{
    Console.WriteLine(group.StandardName);
    group.Students.ToList().ForEach(st => Console.WriteLine(st.Nombre));
}
```

Output

```
Ingenieria en sistemas
Eva
Ana
Rosa
Ot
rafaela
Ingenieria en informatica
Iu
bertha
erika
maria jose
Fabio
nick
juan esteban
nicolas
Diego
```

30. Este hace una asociación de dos agrupaciones para saber en qué carrera está el alumno

```
var studentsWithStandard = from stad in listaProgramas
                           join s in listaAlumnos
                           on stad.Idprograma equals s.Idprograma
                           into sg
                           from std_grp in sg
                           orderby stad.Nombre, std_grp.Nombre
                           select new
                           {
                               StudentName = std_grp.Nombre,
                               StandardName = stad.Nombre
                           };

foreach (var group in studentsWithStandard)
{
    Console.WriteLine("{0} Esta en {1}", group.StudentName, group.StandardName);
}
```

Output

```
erika Esta en Ingenieria en informatica
Fabio Esta en Ingenieria en informatica
Iu Esta en Ingenieria en informatica
juan esteban Esta en Ingenieria en informatica
maria jose Esta en Ingenieria en informatica
nick Esta en Ingenieria en informatica
nicolas Esta en Ingenieria en informatica
Ana Esta en Ingenieria en sistemas
Eva Esta en Ingenieria en sistemas
Ot Esta en Ingenieria en sistemas
rafaela Esta en Ingenieria en sistemas
Rosa Esta en Ingenieria en sistemas
Lesley Esta en Lenguas modernas
alberto Esta en Matematica pura
Jorge eduardo Esta en Matematica pura
Laia Esta en Matematica pura
Quim Esta en Matematica pura
John Esta en Medicina
julian Esta en Salud ocupacional
layla Esta en Sociales
Katherine Esta en Tecnico en sistemas
Juan Esta en Tecnologia en sistemas
rafael Esta en Tecnologia en sistemas
```

31. En consultas anidadas, una consulta se escribe dentro de una consulta. El resultado de la consulta interna se utiliza en la ejecución de la consulta externa. Así mismo pasa en esta consulta

```
var nestedQueries = from s in listaAlumnos
                    where s.Edad > 18 && s.Idprograma ==
                        (from std in listaProgramas
                         where std.Nombre == "Ingenieria en sistemas"
                         select std.Idprograma).FirstOrDefault()
                    select s;

nestedQueries.ToList().ForEach(s => Console.WriteLine(s.Nombre));
```

Output

```
CS Select Microsoft Visu
Eva
Ana
Rosa
Ot
rafaela
```

32. Poner valores por default linq y landa

```
var firProgramas = listaProgramas.First();
var firstProgramas = listaProgramas.FirstOrDefault(e => e.Nombre == "ingenieria agropecuaria");
var AlumnoDefault = listaAlumnos.FirstOrDefault(e => e.Edad == 10 && e.Nombre == "No tiene");
```

33. Este agrupa por ciudades de acuerdo cada estudiante

```
var groupCity = from u in listaUniversidades
                join s in listaAlumnos
                on u.IdCiudad equals s.IdCiudad
                into sg
                from std_grp in sg
                orderby u.Nombre, std_grp.Nombre
                select new
                {
                    StudentName = std_grp.Nombre,
                    StandardName = u.Ciudad
                };

foreach (var group in groupCity)
{
    Console.WriteLine("{0} Esta en {1}", group.StudentName, group.StandardName);
}
```

Output

```
maria jose Esta en Manizales
Rosa Esta en Manizales
Raul Esta en Barranquilla
nick Esta en Quindio
valeria Esta en Quindio
erika Esta en Cali
layla Esta en Cali
Fabio Esta en bogota
Iu Esta en bogota
juan esteban Esta en bogota
Laia Esta en bogota
Ot Esta en bogota
Pep Esta en bogota
Quim Esta en bogota
John Esta en Pasto
rafaela Esta en Pasto
Fabio Esta en Bogota
Iu Esta en Bogota
juan esteban Esta en Bogota
Laia Esta en Bogota
Ot Esta en Bogota
Pep Esta en Bogota
Quim Esta en Bogota
```

34. Agrupamos por ciudad y en cada ciudad esta sus respectivas universidades

```
var universidadCity = from s in listaUniversidades
                      group s by s.Ciudad into sg
                      orderby sg.Key
                      select new { sg.Key, sg };

foreach (var group in universidadCity)
{
    Console.WriteLine(group.Key);

    group.sg.ToList().ForEach(st => Console.WriteLine(st.Nombre));
}
```

Output

```
Barranquilla
Universidad del Atlántico
Bogota
Universidad de América
Politécnico Grancolombiano
Universidad Nacional
Universidad javeriana
boyaca
Universidad de boyaca
Cali
Universidad del valle
Manizales
Universidad de caldas
Universidad de Manizales
Universidad autonoma de manizales
Medellin
Universidad de Antioquia
Monteria
```

35. Agrupar los alumnos de la universidad de caldas con Nota mayores que 6.0 este código se lee de la siguiente manera primero se hace una consulta en listaAlumnos y con el where se hace una subconsulta para y hay comprueba con el primero el id para ahorrar tiempo porque todos tienen el mismo id y luego ya imprimimos

```
var alumnosCity = from s in listaAlumnos
                    where s.Nota > 6.0 && s.IdUniversidad ==
                        (from std in listaUniversidades
                         where std.IdUniversidad == s.IdUniversidad
                         select std.IdUniversidad).FirstOrDefault()
                    select s;

alumnosCity.ToList().ForEach(s => Console.WriteLine(s.Nombre));
```

Output

```
Universidad de Caldas
Ana
Iu
Pep
valeria
julian
Melisa
rafael
bertha
Jorge eduardo
Fabio
nicolas
Diego
```

36. Agrupar por nombre con una consulta linq entonces si tienen el mismo nombre se agrupa por ejemplo yo solo tengo 2 nombre repetidos

```
alumnosCity.ToList().ForEach(s => Console.WriteLine(s.Nombre));

var group2 = from s in listaAlumnos
              group s by s.Nombre.Contains("rafaela") into sg
              orderby sg.Key
              select new { sg.Key, sg };

foreach (var group in group2)
{
    Console.WriteLine("\n"+group.Key);
    group.sg.ToList().ForEach(st => Console.WriteLine(st.Nombre));
}
```

Output

```
True
rafaela
rafaela Nieto
```

37. En esta se unieron tres listas también es como unir tres tablas universidad, alumno y el programa que está escrito

```
var JoinDeTresListas = from d in listaAlumnos
                        join c in listaProgramas on d.Idprograma equals c.Idprograma
                        join s in listaUniversidades on d.IdUniversidad equals s.IdUniversidad
                        select new
                        {
                            Id = s.Nombre,
                            Programa = c.Nombre,
                            Nombre = d.Nombre
                            // other assignments
                        };

foreach (var i in JoinDeTresListas)
{
    Console.WriteLine(i);
}
```

Output

```
{ I = Universidad de Antioquia, Programa = Ingenieria civil, Nombre = valentina }
{ I = Universidad de Córdoba, Programa = Tecnico en sistemas, Nombre = katherine }
{ I = Universidad de Manizales, Programa = Salud ocupacional, Nombre = julian }
{ I = Universidad Antonio Nariño, Programa = Medicina, Nombre = John }
{ I = Universidad Mariana, Programa = Sociales, Nombre = layla }
{ I = Universidad del valle, Programa = Lenguas modernas, Nombre = Lesley }
{ I = Universidad del valle, Programa = enfermeria, Nombre = Melisa }
{ I = Universidad javeriana, Programa = Tecnologia en sistemas, Nombre = Juan }
{ I = Universidad de caldas, Programa = ingenieria agropecuaria, Nombre = Fernando }
{ I = Universidad bolivariana, Programa = Tecnologia en sistemas, Nombre = rafael }
{ I = Universidad autonoma de manizales, Programa = Ingenieria en informatica, Nombre = berthia }
{ I = Universidad de boyaca, Programa = Ingenieria en informatica, Nombre = erika }
{ I = Universidad de boyaca, Programa = Matematica pura, Nombre = alberto }
{ I = Universidad de boyaca, Programa = Matematica pura, Nombre = Jorge eduardo }
{ I = Universidad de caldas, Programa = Ingenieria en sistemas, Nombre = rafaela }
{ I = Universidad de caldas, Programa = Ingenieria en informatica, Nombre = maria jose }
{ I = Universidad de América, Programa = Ingenieria en informatica, Nombre = Fabio }
{ I = Politécnico Grancolombiano, Programa = Ingenieria en informatica, Nombre = nick }
{ I = Universidad Nacional, Programa = Ingenieria en informatica, Nombre = juan esteban }
{ I = Universidad Nacional, Programa = Ingenieria en informatica, Nombre = nicolas }
{ I = Universidad de caldas, Programa = Ingenieria en informatica, Nombre = Diego }
{ I = Politécnico Grancolombiano, Programa = Ingenieria en informatica, Nombre = Margari }
{ I = Universidad de caldas, Programa = Ingenieria en sistemas, Nombre = rafaela Nieto }
```

38. Este es parecido aplicando una condicional where con 3 relaciones aquí filtre todos los de la universidad de caldas

```
var whereJoinDeTresListas = from d in listaAlumnos
                             join c in listaProgramas on d.Idprograma equals c.Idprograma
                             join s in listaUniversidades on d.IdUniversidad equals s.IdUniversidad
                             where d.IdUniversidad.Equals(1)
                             select new
                             {
                                 I = s.Nombre,
                                 Programa = c.Nombre,
                                 Nombre = d.Nombre
                                 // other assignments
                             };

foreach (var i in whereJoinDeTresListas)
{
    Console.WriteLine(i);
}
```

Output

```
{ I = Universidad de caldas, Programa = Ingenieria en sistemas, Nombre = Eva }
{ I = Universidad de caldas, Programa = Ingenieria en sistemas, Nombre = Ana }
{ I = Universidad de caldas, Programa = ingenieria agropecuaria, Nombre = Fernando }
{ I = Universidad de caldas, Programa = Ingenieria en sistemas, Nombre = rafaela }
{ I = Universidad de caldas, Programa = Ingenieria en informatica, Nombre = maria jose }
{ I = Universidad de caldas, Programa = Ingenieria en informatica, Nombre = Diego }
{ I = Universidad de caldas, Programa = Ingenieria en sistemas, Nombre = rafaela Nieto }
```

39. Esta es una búsqueda de profesores que solo dan laboratorio de física I se le relaciona el where con dicha materia

```
var CJoinDeTresListas = from s in listaAsignatura
                         join c in listaProfesores on s.idAsignatura equals c.idAsignaturaP
                         where s.NombreA == ("laboratorio de fisica I")
                         select new {
                             A = s.NombreA,
                             P = c.Nombre
                         };

foreach (var i in CJoinDeTresListas)
{
    Console.WriteLine(i);
}
```

Output

```
{ A = laboratorio de fisica I, P = Daniel ruiz }
{ A = laboratorio de fisica I, P = Hector calvis }
{ A = laboratorio de fisica I, P = Jhonny ariztizabal }
```

40. Aquí hay dos tipos de ordenamiento por la edad y por el nombre de la universidad

```
var UniversidadYEdadAlumnoGroup = from a in listaAlumnos
                                   join u in listaUniversidades on a.IdUniversidad equals u.IdUniversidad
                                   orderby a.Edad, u.Nombre
                                   select new
                                   {
                                       U = u.Nombre,
                                       E = a.Nombre,
                                       edad = a.Edad,
                                   };

UniversidadYEdadAlumnoGroup.ToList().ForEach(s => Console.WriteLine
("Student Name: {0}, Age: {1}, StandardID: {2}", s.U, s.E, s.edad));
```

Output

```
Student Name: Universidad de caldas, Age: rafaela, StandardID: 27
Student Name: Universidad de caldas, Age: rafaela Nieto, StandardID: 27
Student Name: Universidad Mariana, Age: layla, StandardID: 27
Student Name: Universidad del valle, Age: Lesley, StandardID: 28
Student Name: Universidad de América, Age: Fabio, StandardID: 29
Student Name: Universidad de boyaca, Age: erika, StandardID: 29
Student Name: Universidad del valle, Age: Melisa, StandardID: 29
Student Name: Politécnico Grancolombiano, Age: Iu, StandardID: 30
Student Name: Universidad de Manizales, Age: julian, StandardID: 30
Student Name: Universidad Nacional, Age: Laia, StandardID: 30
Student Name: Universidad de caldas, Age: maria jose, StandardID: 31
Student Name: Politécnico Grancolombiano, Age: Pep, StandardID: 32
Student Name: Universidad Nacional, Age: Quim, StandardID: 32
Student Name: Politécnico Grancolombiano, Age: nick, StandardID: 36
Student Name: Universidad autonoma de manizales, Age: bertha, StandardID: 36
Student Name: Universidad javeriana, Age: Juan, StandardID: 36
Student Name: Universidad de boyaca, Age: Jorge eduardo, StandardID: 39
Student Name: Universidad de caldas, Age: Fernando, StandardID: 39
Student Name: Universidad Nacional, Age: juan esteban, StandardID: 39
Student Name: Universidad de caldas, Age: Diego, StandardID: 40
Student Name: Universidad Nacional, Age: nicolas, StandardID: 40
Student Name: Universidad de boyaca, Age: alberto, StandardID: 46
Student Name: Universidad bolivariana, Age: rafael, StandardID: 48
```

Bibliografía

<https://www.tutorialsteacher.com/linq/sample-linq-queries>

<https://www.youtube.com/watch?v=QAK2fXIYaWQ>

<https://www.youtube.com/watch?v=NVWeYz2CFvc>

<https://www.youtube.com/user/NEKSZER>

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/basic-linq-query-operations>

<https://docs.microsoft.com/en-us/dotnet/csharp/linq/write-linq-queries>

<https://www.codingame.com/playgrounds/213/using-c-linq---a-practical-overview/linq-query-syntax>