How to use LINQ

Some of the C# clauses for working with LINQ

```
from
where
orderby
select
join
```

Features of LINQ

- Query language is integrated with C#
- Provides IntelliSense, compile-time syntax checking, and debugging support
- Same basic syntax for each type of query
- Provides designer tools that create *object-relational mappings*

The three stages of a query operation

1. Get the data source. If the data source is an array, for example, you must declare the array and then assign values to its elements.
2. Define the query expression.
3. Execute the query to return the results.

A LINQ query that retrieves data from an array

Code that defines the array

```
int[] numbers = new int[6];
for (int i = 0; i < numbers.Length; i++)
    numbers[i] = i;
```

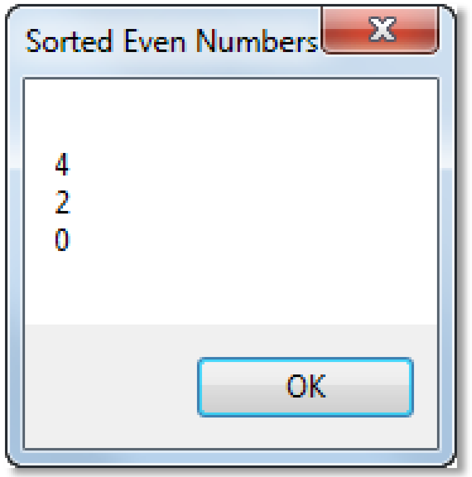
A statement that defines the query expression

```
var numberList = from number in numbers
                 where number % 2 == 0
                 orderby number descending
                 select number;
```

Code that executes the query

```
string numberDisplay = "";
foreach (var number in numberList)
    numberDisplay += number + "\t\t\n";
MessageBox.Show(numberDisplay, "Sorted Even Numbers");
```

The resulting dialog box



```
4
2
0
```

The syntax of the from clause

```
from [type] elementName in collectionName
```

An example that uses an array of decimals

A statement that gets the data source

```
decimal[] salesTotals =
    new decimal[4] {1286.45m, 2433.49m, 2893.85m,
        2094.53m};
```

A statement that defines the query expression

```
var salesList = from sales in salesTotals
                select sales;
```

Code that executes the query

```
decimal sum = 0;
foreach (var sales in salesList)
    sum += sales;
```

An example that uses a generic list of invoices as the data source

The Invoice class

```
public class Invoice
{
    public int InvoiceID { get; set; }
    public int CustomerID { get; set; }
    public DateTime InvoiceDate { get; set; }
    public decimal ProductTotal { get; set; }
    public decimal SalesTax { get; set; }
    public decimal Shipping { get; set; }
    public decimal InvoiceTotal { get; set; }
}
```

A statement that gets the data source

```
List<Invoice> invoiceList = InvoiceDB.GetInvoices();
```

An example that uses a generic list of invoices as the data source (cont.)

A statement that defines the query expression

```
var invoices = from invoice in invoiceList
               select invoice;
```

Code that executes the query

```
decimal sum = 0;
foreach (var invoice in invoices)
    sum += invoice.InvoiceTotal;
```

The syntax of the where clause

```
where condition
```

An example that filters the salesTotals array

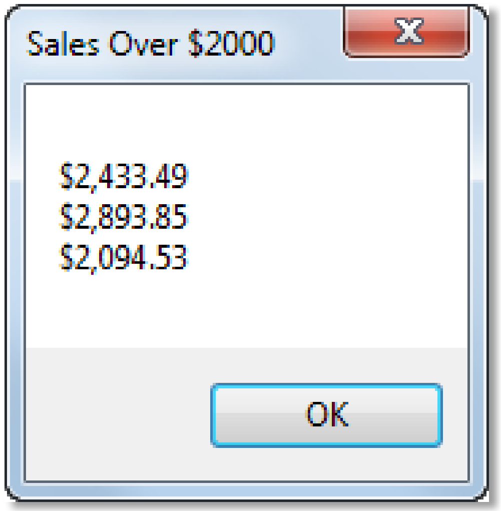
A query expression that returns only sales greater than $2000

```
var salesList = from sales in salesTotals
                where sales > 2000
                select sales;
```

Code that executes the query

```
string salesDisplay = "";
foreach (var sales in salesList)
    salesDisplay += sales.ToString("c") + "\t\t\n";
MessageBox.Show(salesDisplay, "Sales Over $2000");
```

The resulting dialog box


Sales Over $2000

```
$2,433.49
$2,893.85
$2,094.53
```

An example that filters the generic list of invoices

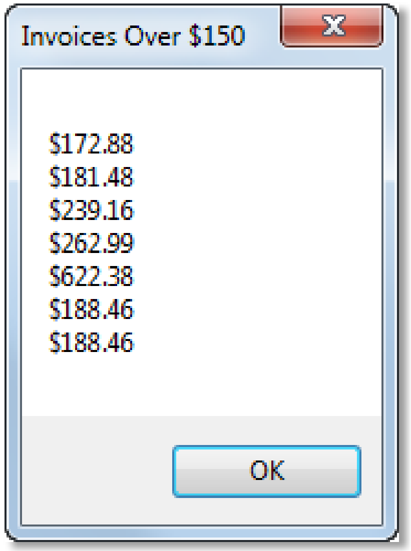
A query expression that returns invoices with totals over $150

```
var invoices = from invoice in invoiceList
               where invoice.InvoiceTotal > 150
               select invoice;
```

Code that executes the query

```
string invoiceDisplay = "";
foreach (var invoice in invoices)
    invoiceDisplay +=
        invoice.InvoiceTotal.ToString("c") + "\t\t\n";
MessageBox.Show(invoiceDisplay, "Invoices Over $150");
```

The resulting dialog box



```
$172.88
$181.48
$239.16
$262.99
$622.38
$188.46
$188.46
```

The syntax of the orderby clause

```
orderby expression1 [ascending|descending]
     [, expression2 [ascending|descending]]...
```

An example that sorts the salesTotals array

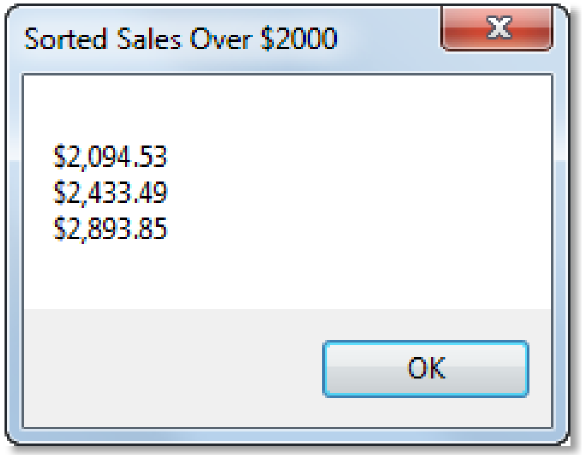
A query expression that sorts the sales in ascending sequence

```
var salesList = from sales in salesTotals
                where sales > 2000
                orderby sales
                select sales;
```

Code that executes the query

```
string salesDisplay = "";
foreach (var sales in salesList)
    salesDisplay += sales.ToString("c") + "\t\t\t\n";
MessageBox.Show(salesDisplay, "Sorted Sales Over $2000");
```

The resulting dialog box


Sorted Sales Over $2000

```
$2,094.53
$2,433.49
$2,893.85
```

An example that sorts the generic list of invoices

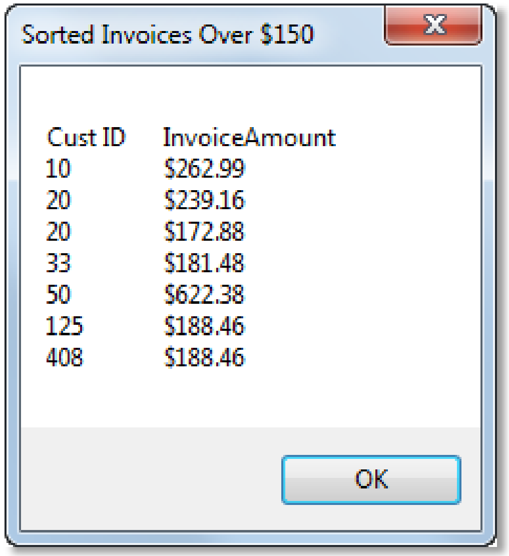
A query expression that sorts the invoices by customer ID and invoice total

```
var invoices = from invoice in invoiceList
               where invoice.InvoiceTotal > 150
               orderby invoice.CustomerID,
                       invoice.InvoiceTotal descending
               select invoice;
```

Code that executes the query

```
string invoiceDisplay = "Cust ID\tInvoice amount\n";
foreach (var invoice in invoices)
    invoiceDisplay += invoice.CustomerID + "\t"
                    + invoice.InvoiceTotal.ToString("c")
                    + "\n";
MessageBox.Show(invoiceDisplay,
    "Sorted Invoices Over $150");
```

The resulting dialog box



```
Cust ID    InvoiceAmount
10         $262.99
20         $239.16
20         $172.88
33         $181.48
50         $622.38
125        $188.46
408        $188.46
```

Two ways to code the select clause

```
select columnExpression

select new [type] { [PropertyName1 =] columnExpression1
                 [, [PropertyName2 =] columnExpression2]... }
```

An example that selects key values from a sorted list

The employee sales sorted list

```
SortedList<string, decimal> employeeSales =
    new SortedList<string, decimal>
    { ["Anderson"] = 1286.45m, ["Menendez"] = 2433.49m,
      ["Thompson"] = 2893.85m, ["Wilkinson"] = 2094.53m };
```

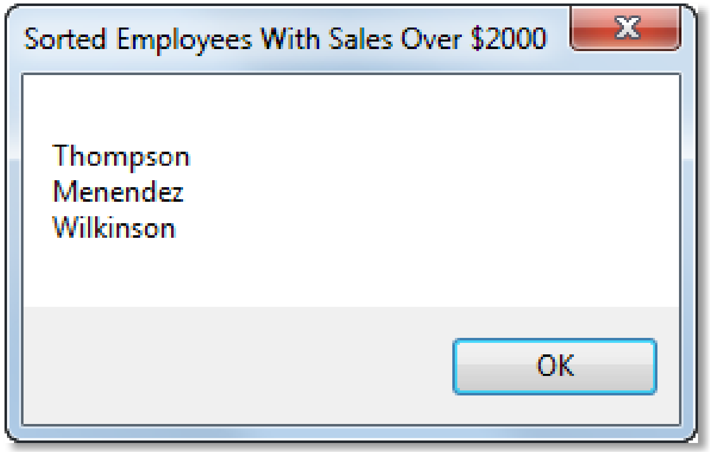
A query expression that selects the employee names

```
var employeeList = from sales in employeeSales
                   where sales.Value > 2000
                   orderby sales.Value descending
                   select sales.Key;
```

Code that executes the query

```
string employeeDisplay = "";
foreach (var employee in employeeList)
    employeeDisplay += employee + "\t\t\t\n";
MessageBox.Show(employeeDisplay,
    "Sorted Employees With Sales Over $2000");
```

The resulting dialog box



```
Thompson
Menendez
Wilkinson
```

A query expression that creates an anonymous type from the list of invoices

```
var invoices = from invoice in invoiceList
               where invoice.InvoiceTotal > 150
               orderby invoice.CustomerID,
                       invoice.InvoiceTotal descending
               select new { invoice.CustomerID,
                            invoice.InvoiceTotal };
```

The basic syntax of the join clause

```
join elementName in collectionName
    on keyName1 equals keyName2
```

An example that joins data from two generic lists

The Customer class

```
public class Customer
{
    public int CustomerID { get; set; }
    public string Name { get; set; }
}
```

Code that gets the two data sources

```
List<Invoice> invoiceList = InvoiceDB.GetInvoices();
List<Customer> customerList = CustomerDB.GetCustomers();
```

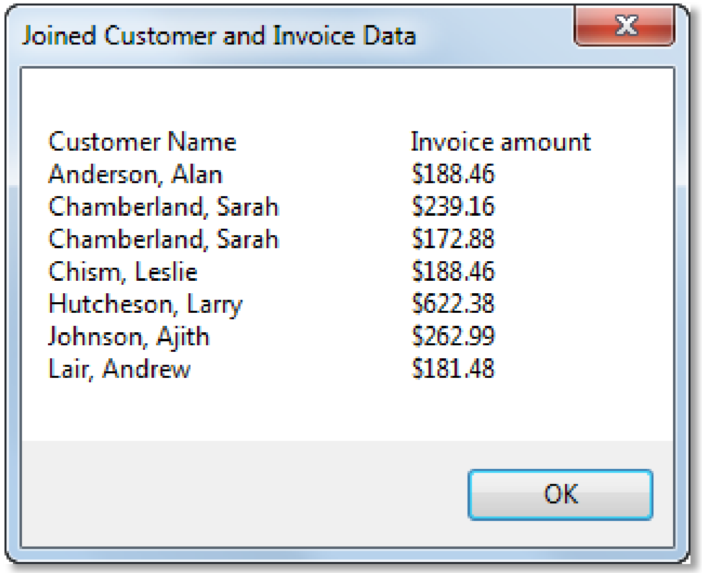
A query expression that joins data from the two data sources

```
var invoices = from invoice in invoiceList
               join customer in customerList
               on invoice.CustomerID equals customer.CustomerID
               where invoice.InvoiceTotal > 150
               orderby customer.Name,
                       invoice.InvoiceTotal descending
               select new { customer.Name,
                            invoice.InvoiceTotal };
```

Code that executes the query

```
string invoiceDisplay = "Customer Name\t\tInvoice amount\n";
foreach (var invoice in invoices)
{
    invoiceDisplay += invoice.Name + "\t\t";
    invoiceDisplay += invoice.InvoiceTotal.ToString("c") + "\n";
}
MessageBox.Show(invoiceDisplay,
     "Joined Customer and Invoice Data");
```

The resulting dialog box



Joined Customer and Invoice Data

```
Customer Name                Invoice amount
Anderson, Alan               $188.46
Chamberland, Sarah           $239.16
Chamberland, Sarah           $172.88
Chism, Leslie                $188.46
Hutcheson, Larry             $622.38
Johnson, Ajith               $262.99
Lair, Andrew                 $181.48
```

An extension method that extends the String data type

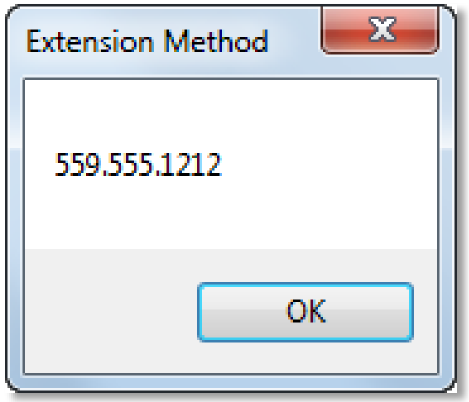
A class with an extension method that formats a phone number

```
public static class StringExtensions
{
    public static string FormattedPhoneNumber(this string phone,
        string separator)
    {
        return phone.Substring(0, 3) + separator
            + phone.Substring(3, 3) + separator
            + phone.Substring(6, 4);
    }
}
```

Code that uses the extension method

```
string phoneNumber = "5595551212";
string formattedPhoneNumber =
phoneNumber.FormattedPhoneNumber(".");
MessageBox.Show(formattedPhoneNumber + "\t", "Extension Method");
```

The resulting dialog box



```
559.555.1212
```

Extension methods used to implement common C# clauses for LINQ

Clause	Method
`where`	`Where`
`orderby`	`OrderBy, OrderByDescending, ThenBy, ThenByDescending`
`select`	`Select`
`join`	`Join`

The basic syntax of a lambda expression

```
[(]parameterList[)] => expression
```

A lambda expression that tests a condition

A statement that declares the delegate type at the class level

```
delegate bool compareDel(decimal total);
```

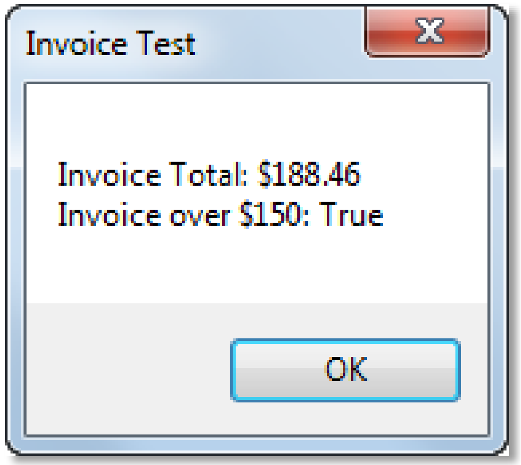
A statement that defines the lambda expression and assigns it to a variable created from the delegate type

```
compareDel invoiceOver150 = total => total > 150;
```

Code that executes the lambda expression

```
decimal invoiceTotal = 188.46m;
string invoiceMessage = "";
invoiceMessage += "Invoice Total: " +
                  invoiceTotal.ToString("c") +
                  "\n" + "Invoice over $150: " +
                  invoiceOver150(invoiceTotal);
MessageBox.Show(invoiceMessage, "Invoice Test");
```

The resulting dialog box

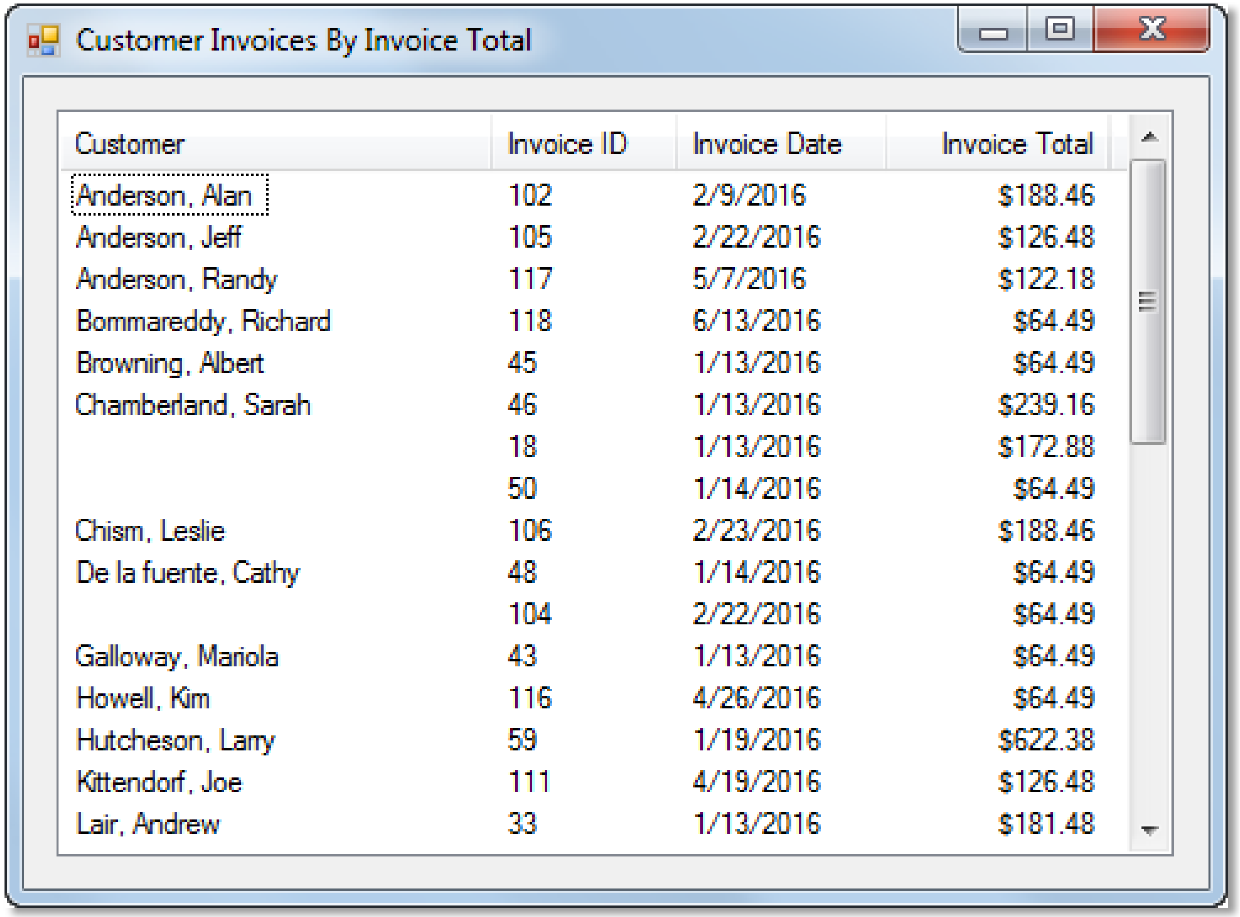


```
Invoice Total: $188.46
Invoice over $150: True
```

A query that uses extension methods and lambda expressions

```
var invoices = invoiceList
               .Where(i => i.InvoiceTotal > 150)
               .OrderBy(i => i.CustomerID)
               .ThenByDescending(i => i.InvoiceTotal)
               .Select(i => new { i.CustomerID,
                                  i.InvoiceTotal });
```

The Customer Invoice form



Customer Invoices By Invoice Total

Customer	Invoice ID	Invoice Date	Invoice Total
Anderson, Alan	102	2/9/2016	$188.46
Anderson, Jeff	105	2/22/2016	$126.48
Anderson, Randy	117	5/7/2016	$122.18
Bommareddy, Richard	118	6/13/2016	$64.49
Browning, Albert	45	1/13/2016	$64.49
Chamberland, Sarah	46	1/13/2016	$239.16
	18	1/13/2016	$172.88
	50	1/14/2016	$64.49
Chism, Leslie	106	2/23/2016	$188.46
De la fuente, Cathy	48	1/14/2016	$64.49
	104	2/22/2016	$64.49
Galloway, Mariola	43	1/13/2016	$64.49
Howell, Kim	116	4/26/2016	$64.49
Hutcheson, Larry	59	1/19/2016	$622.38
Kittendorf, Joe	111	4/19/2016	$126.48
Lair, Andrew	33	1/13/2016	$181.48

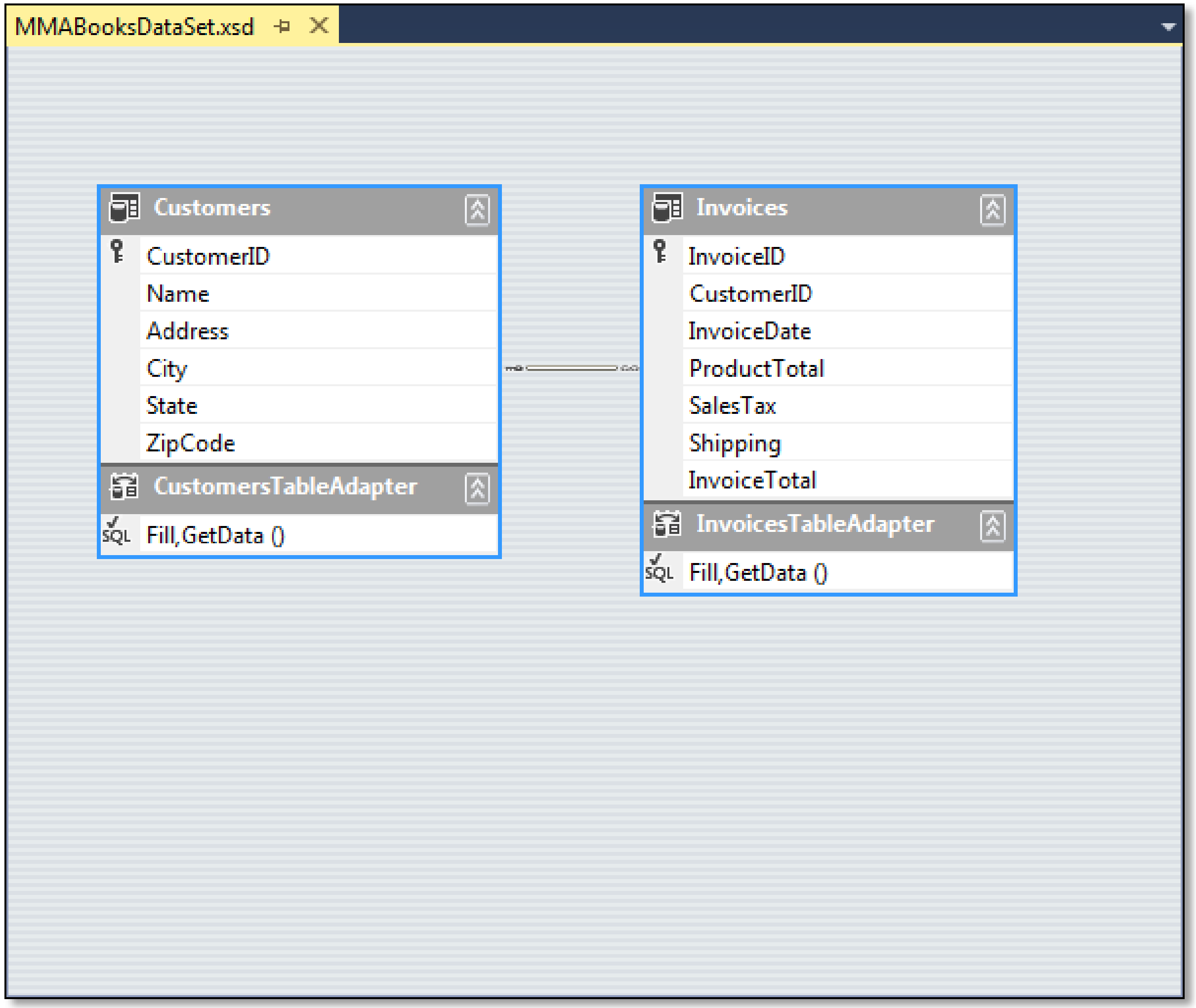
Customer Invoice form with generic lists

```
private void Form1_Load(object sender, EventArgs e)
{
    List<Customer> customerList =
        CustomerDB.GetCustomers();
    List<Invoice> invoiceList = InvoiceDB.GetInvoices();

    var invoices = from invoice in invoiceList
                   join customer in customerList
                   on invoice.CustomerID
                   equals customer.CustomerID
                   orderby customer.Name,
                           invoice.InvoiceTotal descending
                   select new { customer.Name,
                                invoice.InvoiceID,
                                invoice.InvoiceDate,
                                invoice.InvoiceTotal };

    string customerName = "";
    int i = 0;
```

Customer Invoice form with generic lists (cont.)

```
        foreach (var invoice in invoices)
        {
            if (invoice.Name != customerName)
            {
                lvInvoices.Items.Add(invoice.Name);
                customerName = invoice.Name;
            }
            else
            {
                lvInvoices.Items.Add("");
            }
            lvInvoices.Items[i].SubItems.Add(
                invoice.InvoiceID.ToString());
            lvInvoices.Items[i].SubItems.Add(
                Convert.ToDateTime(
                    invoice.InvoiceDate).ToShortDateString());
            lvInvoices.Items[i].SubItems.Add(
                invoice.InvoiceTotal.ToString("c"));
            i += 1;
        }
    }
```

The dataset schema



Customers
- CustomerID
- Name
- Address
- City
- State
- ZipCode

CustomersTableAdapter
- Fill,GetData ()

Invoices
- InvoiceID
- CustomerID
- InvoiceDate
- ProductTotal
- SalesTax
- Shipping
- InvoiceTotal

InvoicesTableAdapter
- Fill,GetData ()

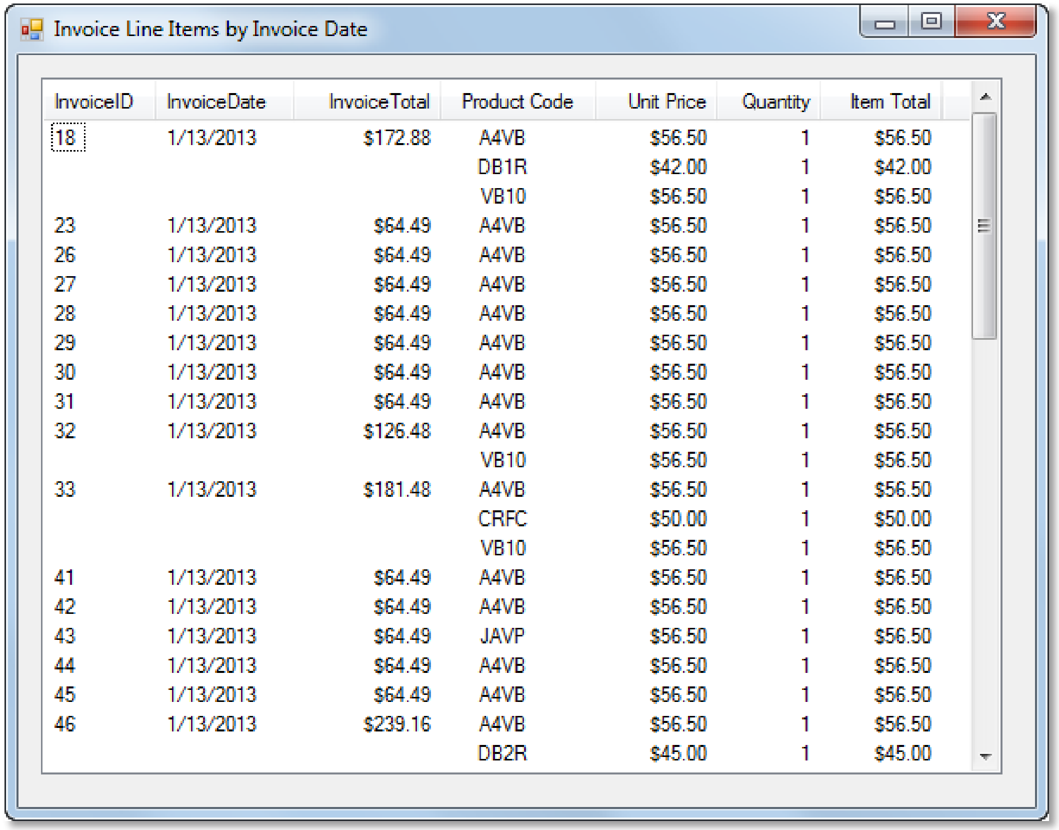
Customer Invoice form with typed dataset

```
MMABooksDataSet mmaBooksDataSet = new MMABooksDataSet();
InvoicesTableAdapter invoicesTableAdapter =
    new InvoicesTableAdapter();
CustomersTableAdapter customersTableAdapter =
    new CustomersTableAdapter();

private void Form1_Load(object sender, EventArgs e)
{
    invoicesTableAdapter.Fill(mmaBooksDataSet.Invoices);
    customersTableAdapter.Fill(mmaBooksDataSet.Customers);

    var invoices = from invoice in mmaBooksDataSet.Invoices
                   join customer in mmaBooksDataSet.Customers
                   on invoice.CustomerID equals customer.CustomerID
                   orderby customer.Name,
                           invoice.InvoiceTotal descending
                   select new { customer.Name,
                                invoice.InvoiceID,
                                invoice.InvoiceDate,
                                invoice.InvoiceTotal };

    string customerName = "";
    int i = 0;
```

Customer Invoice form with typed dataset (cont.)

```
        foreach (var invoice in invoices)
        {
            if (invoice.Name != customerName)
            {
                lvInvoices.Items.Add(invoice.Name);
                customerName = invoice.Name;
            }
            else
            {
                lvInvoices.Items.Add("");
            }
            lvInvoices.Items[i].SubItems.Add(
                invoice.InvoiceID.ToString());
            lvInvoices.Items[i].SubItems.Add(
                Convert.ToDateTime(
                    invoice.InvoiceDate).ToShortDateString());
            lvInvoices.Items[i].SubItems.Add(
                invoice.InvoiceTotal.ToString("c"));
            i += 1;
        }
    }
```

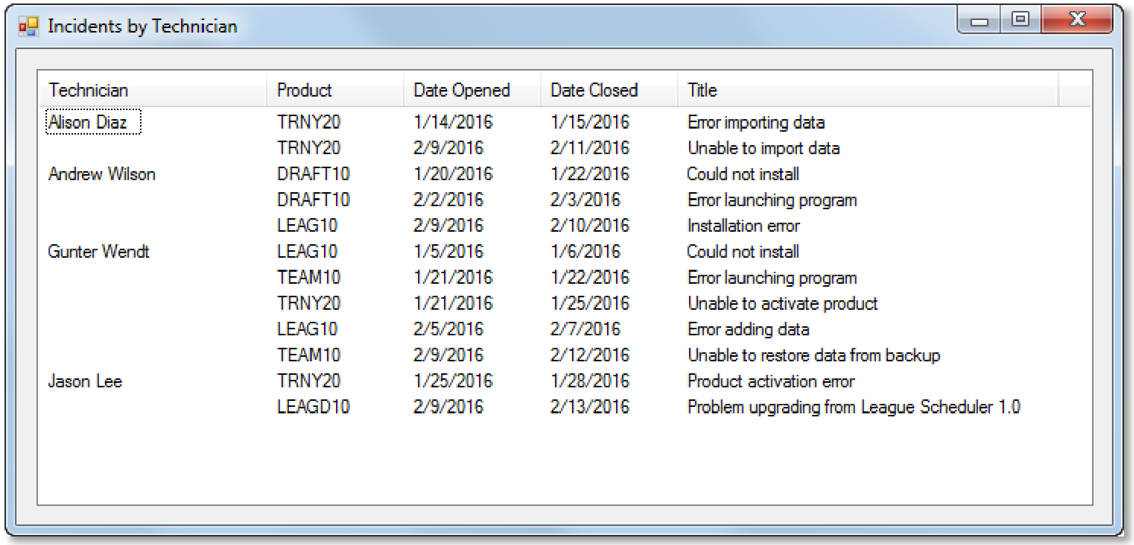
Extra 23-1 Use LINQ to create an Invoice Line Items application



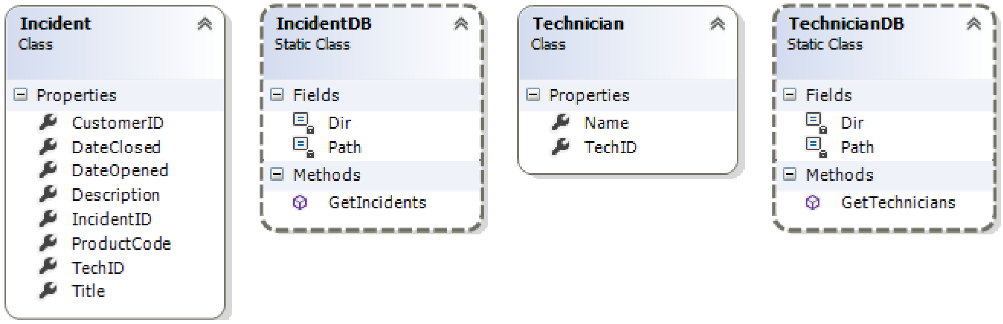
InvoiceID	InvoiceDate	InvoiceTotal	Product Code	Unit Price	Quantity	Item Total
18	1/13/2013	$172.88	A4VB	$56.50	1	$56.50
			DB1R	$42.00	1	$42.00
			VB10	$56.50	1	$56.50
23	1/13/2013	$64.49	A4VB	$56.50	1	$56.50
26	1/13/2013	$64.49	A4VB	$56.50	1	$56.50
27	1/13/2013	$64.49	A4VB	$56.50	1	$56.50
28	1/13/2013	$64.49	A4VB	$56.50	1	$56.50
29	1/13/2013	$64.49	A4VB	$56.50	1	$56.50
30	1/13/2013	$64.49	A4VB	$56.50	1	$56.50
31	1/13/2013	$64.49	A4VB	$56.50	1	$56.50
32	1/13/2013	$126.48	A4VB	$56.50	1	$56.50
			VB10	$56.50	1	$56.50
33	1/13/2013	$181.48	A4VB	$56.50	1	$56.50
			CRFC	$50.00	1	$50.00
			VB10	$56.50	1	$56.50
41	1/13/2013	$64.49	A4VB	$56.50	1	$56.50
42	1/13/2013	$64.49	A4VB	$56.50	1	$56.50
43	1/13/2013	$64.49	JAVP	$56.50	1	$56.50
44	1/13/2013	$64.49	A4VB	$56.50	1	$56.50
45	1/13/2013	$64.49	A4VB	$56.50	1	$56.50
46	1/13/2013	$239.16	A4VB	$56.50	1	$56.50
			DB2R	$45.00	1	$45.00

Use LINQ to join the data in two List<> objects and then display that data in a ListView control.

Project 5-2 Display incidents by technician



Technician	Product	Date Opened	Date Closed	Title
Alison Diaz	TRNY20	1/14/2016	1/15/2016	Error importing data
	TRNY20	2/9/2016	2/11/2016	Unable to import data
Andrew Wilson	DRAFT10	1/20/2016	1/22/2016	Could not install
	DRAFT10	2/2/2016	2/3/2016	Error launching program
	LEAG10	2/9/2016	2/10/2016	Installation error
Gunter Wendt	LEAG10	1/5/2016	1/6/2016	Could not install
	TEAM10	1/21/2016	1/22/2016	Error launching program
	TRNY20	1/21/2016	1/25/2016	Unable to activate product
	LEAG10	2/5/2016	2/7/2016	Error adding data
	TEAM10	2/9/2016	2/12/2016	Unable to restore data from backup
Jason Lee	TRNY20	1/25/2016	1/28/2016	Product activation error
	LEAGD10	2/9/2016	2/13/2016	Problem upgrading from League Scheduler 1.0



Develop an application that uses LINQ to display closed incidents by technician.