Lab 08
Student Registration

Using ADO.NET Disconnected Layer

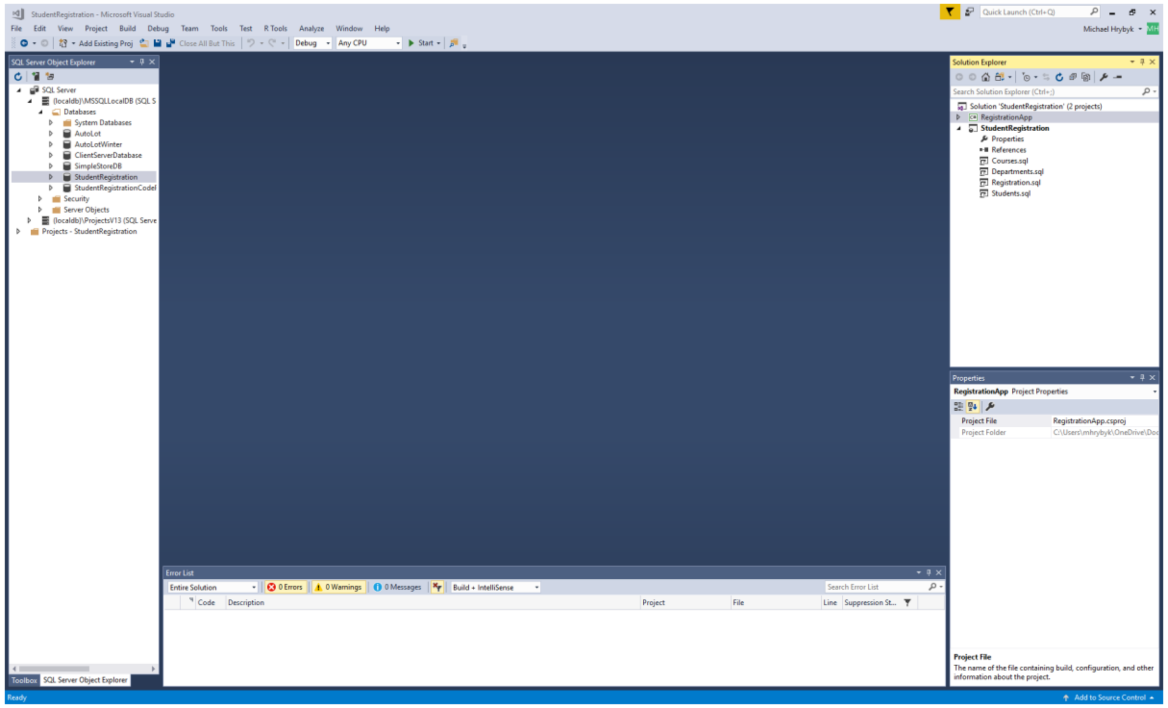
Registration Database and Application

- Download Visual Studio project from Blackboard – Lab08StudentRegistration.zip
  - Unpack the file, you will find Lab08StudentRegistration project folder
  - Click on the solution file (StudentRegistration.sln)
  - There will be two projects
    - StudentRegistration
      - Contains the SQL scripts to create all of the database tables
    - RegistrationApp
      - Partially completed application that uses the ADO disconnected layer to populate datagridview controls
      - SImilar to AutoLot and other examples covered in the text and in class.
- Description
  - From the StudentRegistration database
    - Populate four datagridview controls, one for each table
      - Students
      - Courses
      - Registration
      - Department
  - In the fifth datagridview control, show a list of all students who have registered for courses, but only display
      - Student Last Name
      - Course ID
      - Course Department ID
      - Course Title
    - This will require a compound join using LINQ

Lab Requirements

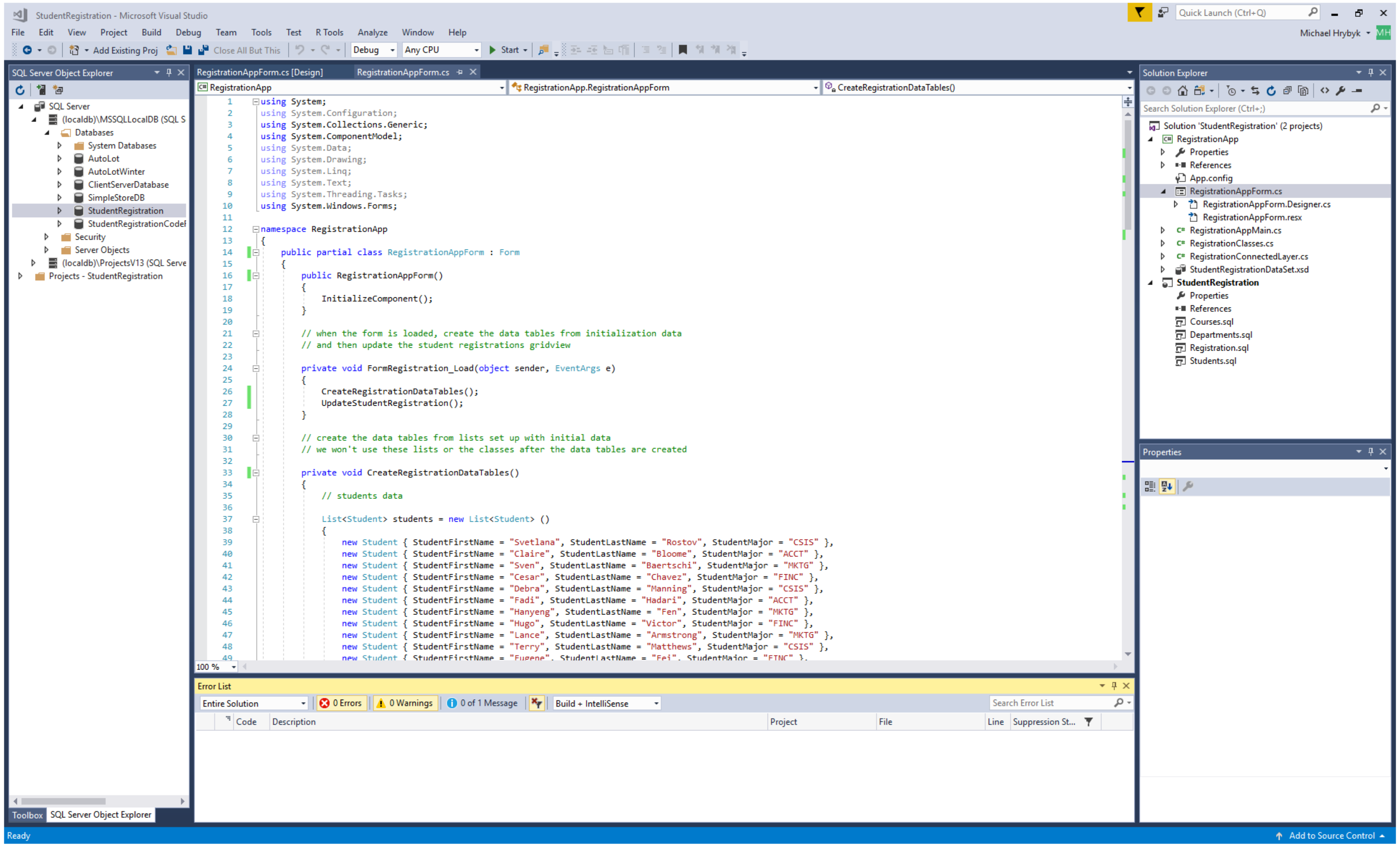
- Project Tasks
  - Create the database – name it StudentRegistration
  - Create all tables within the StudentRegistration SQL project (do NOT do this via the database directly)
  - Create the dataset – name it StudentRegistrationDataSet
  - Read and understand the code in the RegistrationApp C# project
  - Complete the unfinished code in FormRegistrationApp.cs
    - Task 1: add code to fill the dataGridViewRegistration control
    - Task 2: in the ButtonUpdate_Click() event method, update the datagridview controls for the database tables
    - Task 3: in the UpdateStudentRegistration() method
      - Join students, registration, and courses tables to get the courses students are registered for, then update the studentRegistrationDataGridView control (hint: by updating its datasource table). You must use LINQ for this.
      - Do NOT use SQL script or connected layer for this.
  - Test your code
    - Try editing the Registration dataGridView control by adding/changing registration, and then Update
    - Add and delete records from various tables, then check the database directly to see if it worked.
- Show instructor completed lab for credit.

Create database and tables



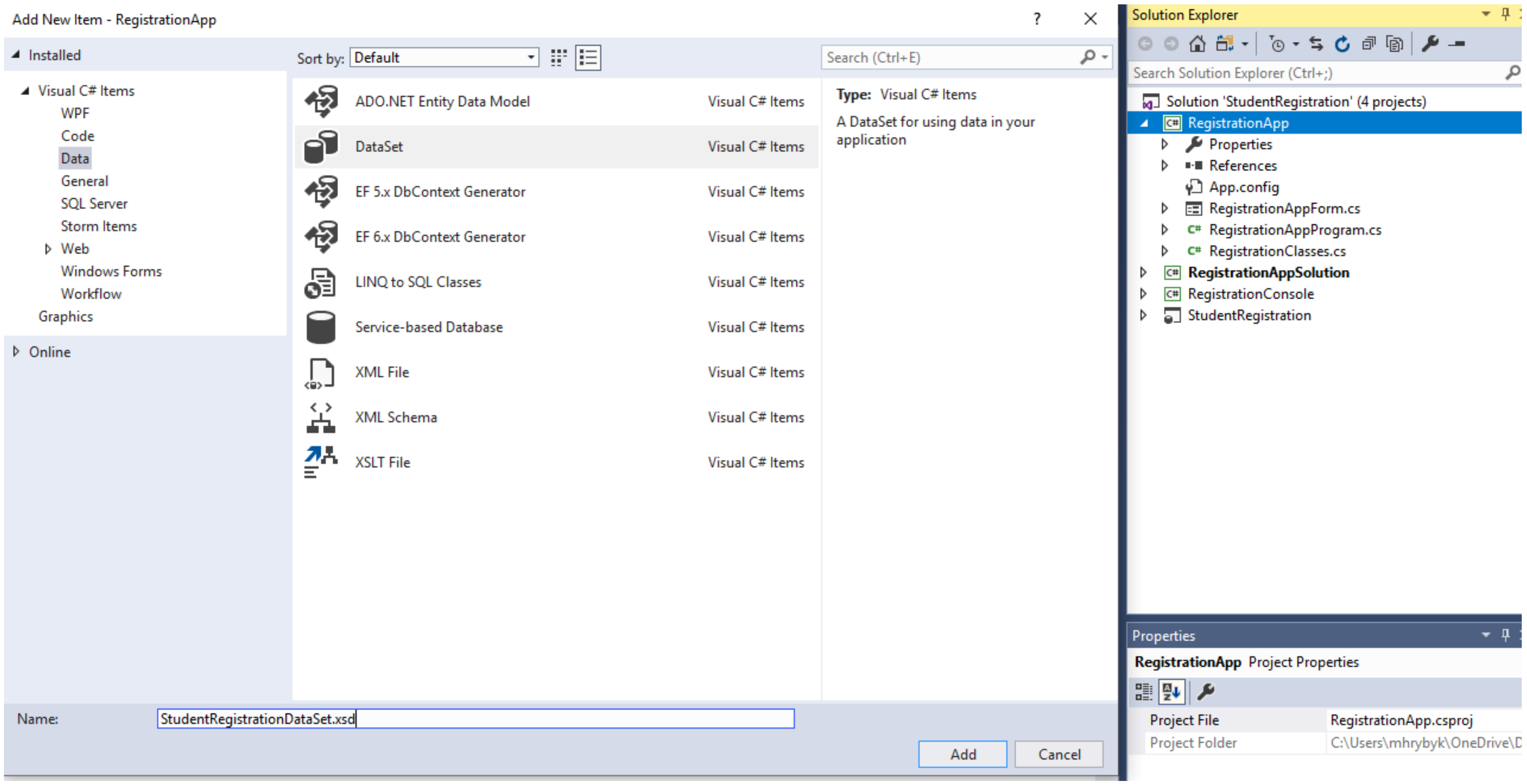
- Under Server Explorer, create StudentRegistration database
- Set StudentRegistration (SQL Project) as Start Up Project
- Deploy (hit Start button)
  - Or use Publish
- Check to see that tables were created

Set RegistrationApp as Startup Project



```csharp
using System;
using System.Configuration;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace RegistrationApp
{
    public partial class RegistrationAppForm : Form
    {
        public RegistrationAppForm()
        {
            InitializeComponent();
        }

        // when the form is loaded, create the data tables from initialization data
        // and then update the student registrations gridview

        private void FormRegistration_Load(object sender, EventArgs e)
        {
            CreateRegistrationDataTables();
            UpdateStudentRegistration();
        }

        // create the data tables from lists set up with initial data
        // we won't use these lists or the classes after the data tables are created

        private void CreateRegistrationDataTables()
        {
            // students data

            List<Student> students = new List<Student> ()
            {
                new Student { StudentFirstName = "Svetlana", StudentLastName = "Rostov", StudentMajor = "CSIS" },
                new Student { StudentFirstName = "Claire", StudentLastName = "Bloome", StudentMajor = "ACCT" },
                new Student { StudentFirstName = "Sven", StudentLastName = "Baertschi", StudentMajor = "MKTG" },
                new Student { StudentFirstName = "Cesar", StudentLastName = "Chavez", StudentMajor = "FINC" },
                new Student { StudentFirstName = "Debra", StudentLastName = "Manning", StudentMajor = "CSIS" },
                new Student { StudentFirstName = "Fadi", StudentLastName = "Hadari", StudentMajor = "ACCT" },
                new Student { StudentFirstName = "Hanyeng", StudentLastName = "Fen", StudentMajor = "MKTG" },
                new Student { StudentFirstName = "Hugo", StudentLastName = "Victor", StudentMajor = "FINC" },
                new Student { StudentFirstName = "Lance", StudentLastName = "Armstrong", StudentMajor = "MKTG" },
                new Student { StudentFirstName = "Terry", StudentLastName = "Matthews", StudentMajor = "CSIS" },
                new Student { StudentFirstName = "Eugene", StudentLastName = "Fei", StudentMajor = "FINC" },
```

Create StudentRegistrationDataSet



Add New Item - RegistrationApp

Installed

- Visual C# Items
  - WPF
  - Code
  - Data
  - General
  - SQL Server
  - Storm Items
  - Web
  - Windows Forms
  - Workflow
- Graphics

Online

Sort by: Default

Search (Ctrl+E)

ADO.NET Entity Data Model	Visual C# Items
DataSet	Visual C# Items
EF 5.x DbContext Generator	Visual C# Items
EF 6.x DbContext Generator	Visual C# Items
LINQ to SQL Classes	Visual C# Items
Service-based Database	Visual C# Items
XML File	Visual C# Items
XML Schema	Visual C# Items
XSLT File	Visual C# Items

Type: Visual C# Items

A DataSet for using data in your application

Name: StudentRegistrationDataSet.xsd

Add Cancel

Solution Explorer

Search Solution Explorer (Ctrl+;)

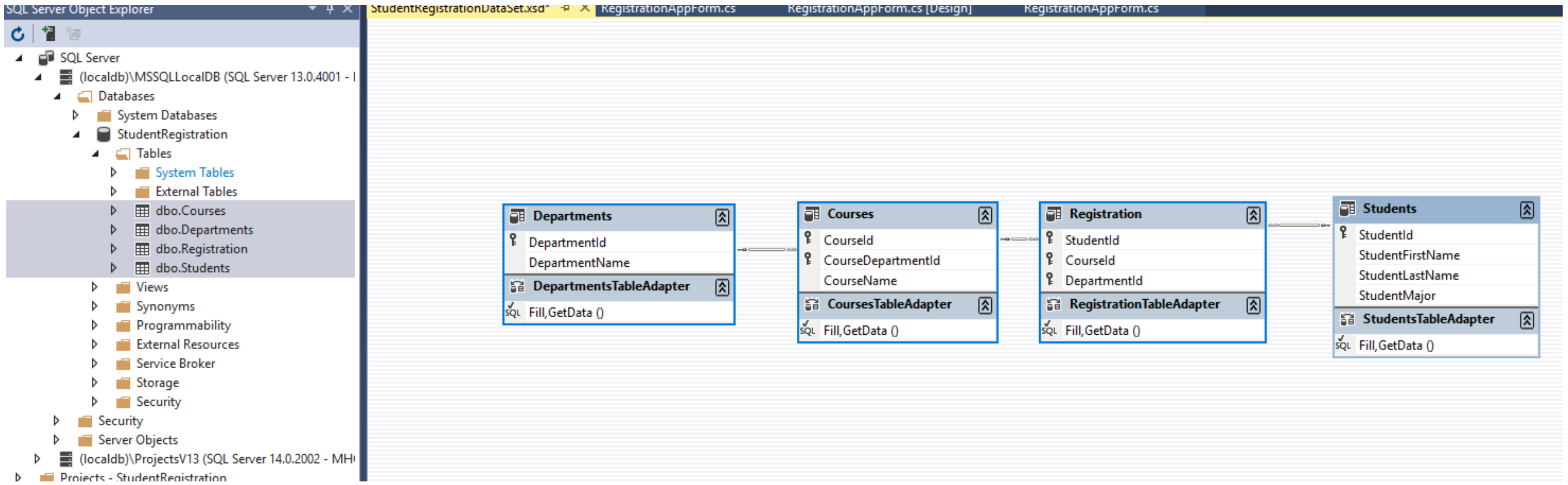
- Solution 'StudentRegistration' (4 projects)
  - RegistrationApp
    - Properties
    - References
    - App.config
    - RegistrationAppForm.cs
    - RegistrationAppProgram.cs
    - RegistrationClasses.cs
  - **RegistrationAppSolution**
  - RegistrationConsole
  - StudentRegistration

Properties

RegistrationApp Project Properties

Project File	RegistrationApp.csproj
Project Folder	C:\Users\mhrybyk\OneDrive\D

Drag DB Tables over to DataSet



Task 1

- Initialize the database from the data in various Lists.
- Use Fill(), Insert(), and Update() methods from the table adapter.
- Then set the control’s DataSource to the table.
- Students table is already done for you

```
// add Students using adapter insert, then update and fill
// then bind the datasource to the table
// this is already done for you

foreach (Student s in students)
{
    // an alternative to using the Insert method

    //StudentRegistrationDataSet.StudentsRow row =
            studentRegistrationDataSet.Students.NewStudentsRow();
    //row.StudentFirstName = s.StudentFirstName;
    //row.StudentLastName = s.StudentLastName;
    //row.StudentMajor = s.StudentMajor;
    //studentRegistrationDataSet.Students.AddStudentsRow(row);

    studentsTableAdapter.Insert(s.StudentFirstName, s.StudentLastName, s.StudentMajor);
}
studentsTableAdapter.Update(studentsTable);
studentsTableAdapter.Fill(studentsTable);

dataGridViewStudents.DataSource = studentsTable;
```

Task 2

- In RegistrationAppForm.cs, complete the code for the ButtonUpdate_Click() method
- Make sure all data from all tables are shown in their respective controls

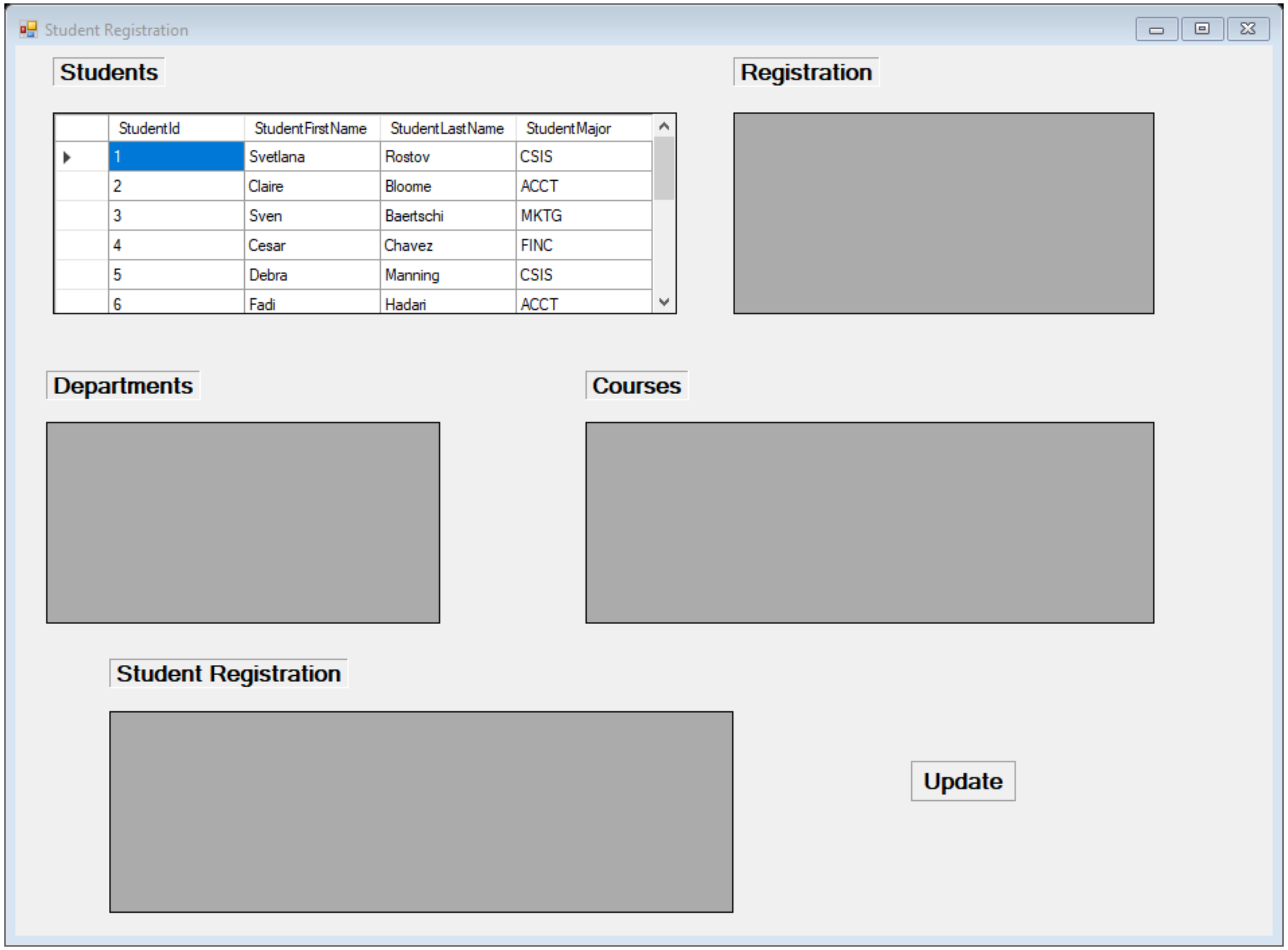
```
/// <summary>
        /// Update all of the tables then fill.
        /// This syncs the dataset with the database and the datagridview controls.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void ButtonUpdate_Click(object sender, EventArgs e)
        {
            // your code here


            // we now have the latest data so
            // list the courses and students registered requires multi join
            // so we have a separate gridview for this

            UpdateStudentRegistration();
        }
```

Task 3

- In RegistrationAppForm.cs, complete the UpdateStudentRegistration() method
- Join students, registration, and courses tables to get the courses students are registered for, then update the studentRegistrationDataGridView control (hint: use query.toList())
- You must use LINQ for this.
- Do NOT use SQL script or connected layer for this.

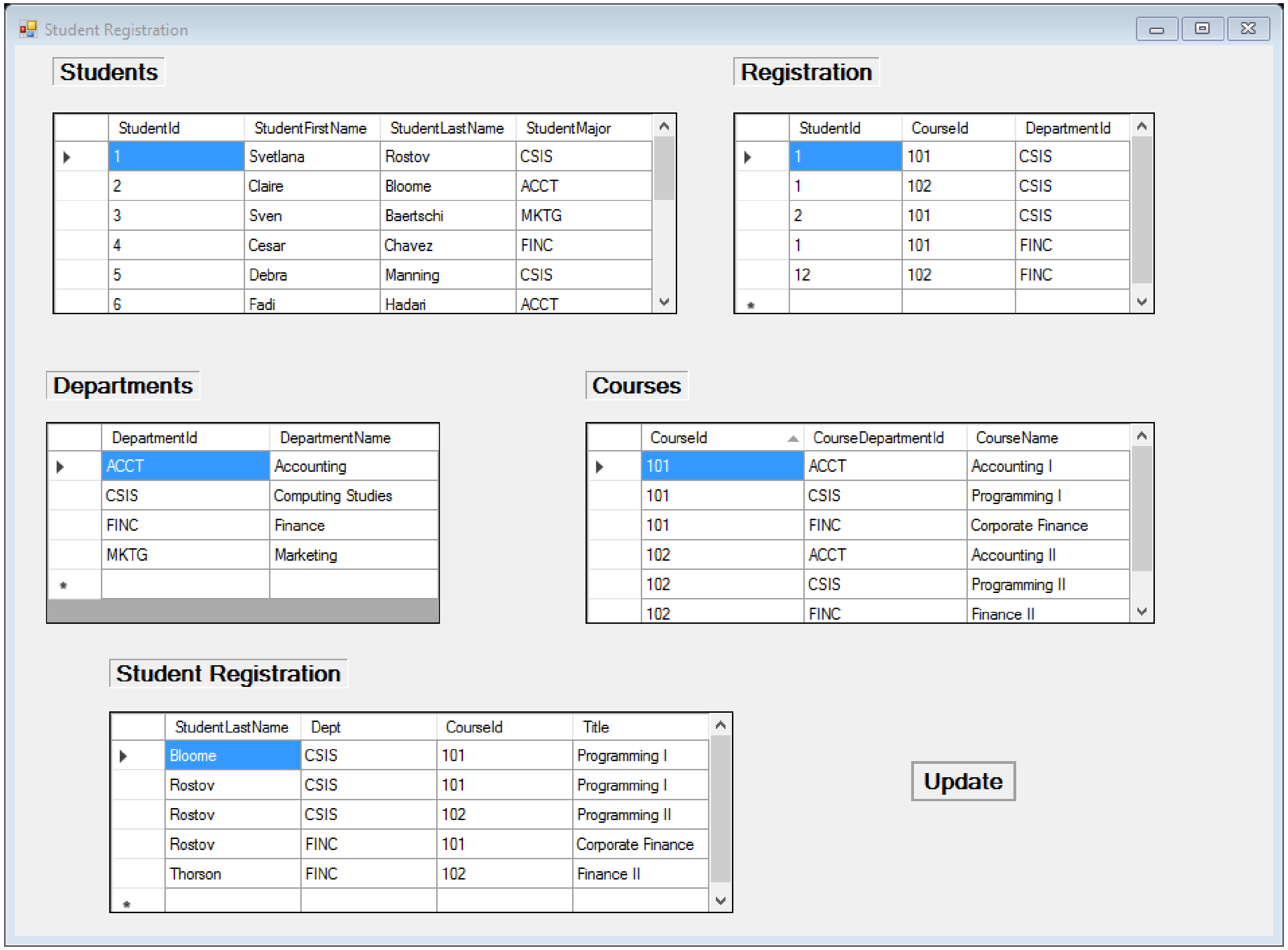
```
/// <summary>
        /// Update the student registration datagridview
        /// </summary>
        private void UpdateStudentRegistration()
        {

            // using linq, join students, registration, and courses to get the courses students are
                     registered for

            // your code here




            // don't forget to bind the query result to the control's DataSource
            // your code here

        }
```

Initial program output (DB creation)



	StudentId	StudentFirstName	StudentLastName	StudentMajor
▶	1	Svetlana	Rostov	CSIS
	2	Claire	Bloome	ACCT
	3	Sven	Baertschi	MKTG
	4	Cesar	Chavez	FINC
	5	Debra	Manning	CSIS
	6	Fadi	Hadari	ACCT

Final Program Output



Students

	StudentId	StudentFirstName	StudentLastName	StudentMajor
▶	1	Svetlana	Rostov	CSIS
	2	Claire	Bloome	ACCT
	3	Sven	Baertschi	MKTG
	4	Cesar	Chavez	FINC
	5	Debra	Manning	CSIS
	6	Fadi	Hadari	ACCT

Registration

	StudentId	CourseId	DepartmentId
▶	1	101	CSIS
	1	102	CSIS
	2	101	CSIS
	1	101	FINC
	12	102	FINC
*			

Departments

	DepartmentId	DepartmentName
▶	ACCT	Accounting
	CSIS	Computing Studies
	FINC	Finance
	MKTG	Marketing
*		

Courses

	CourseId	CourseDepartmentId	CourseName
▶	101	ACCT	Accounting I
	101	CSIS	Programming I
	101	FINC	Corporate Finance
	102	ACCT	Accounting II
	102	CSIS	Programming II
	102	FINC	Finance II

Student Registration

	StudentLastName	Dept	CourseId	Title
▶	Bloome	CSIS	101	Programming I
	Rostov	CSIS	101	Programming I
	Rostov	CSIS	102	Programming II
	Rostov	FINC	101	Corporate Finance
	Thorson	FINC	102	Finance II
*				