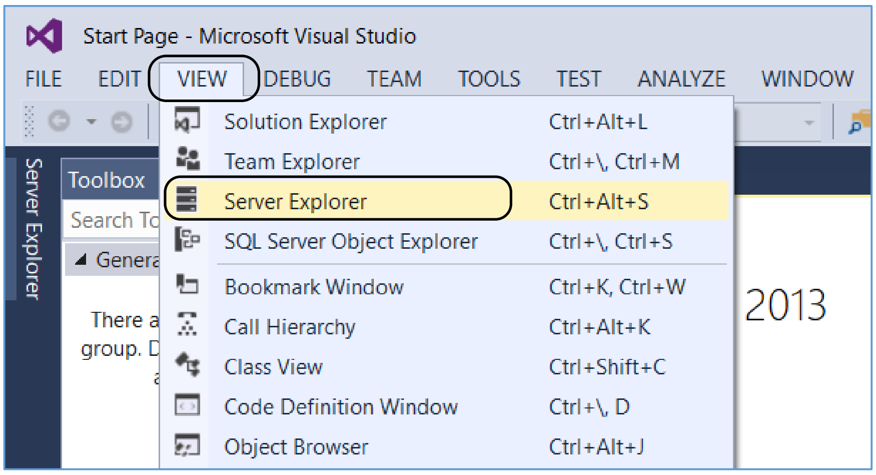
5 Airport

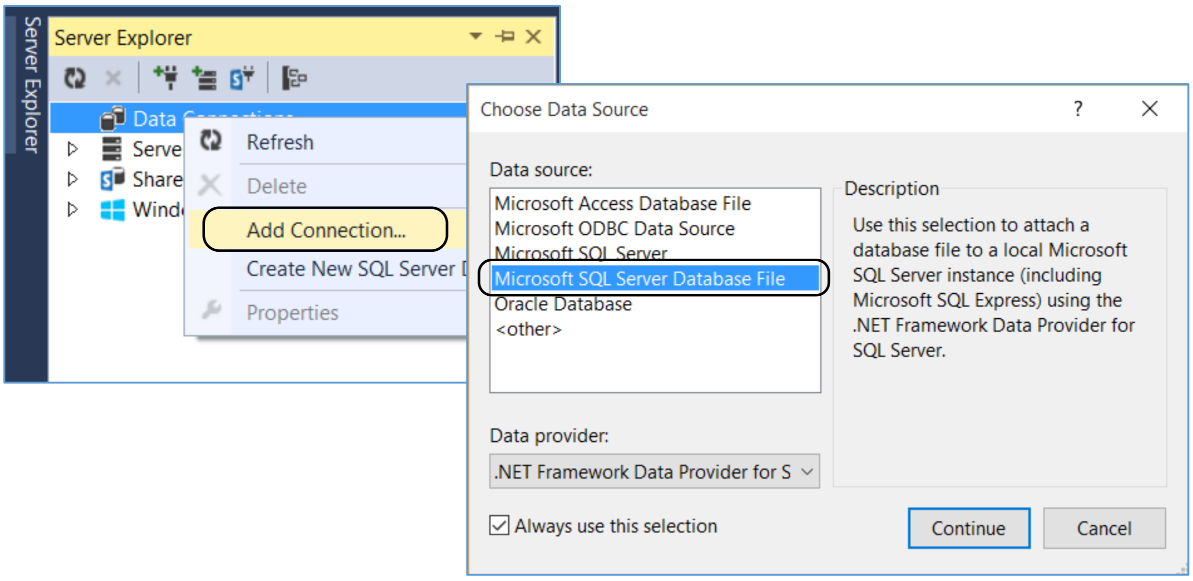
Most practical applications in C# require data to be stored in a database and accessed by the program. We will examine how this is done by setting up a small database of flights departing from an airport, then writing a C# program to display this data in a table.

Open *Visual Studio*, but do not start a new project yet.

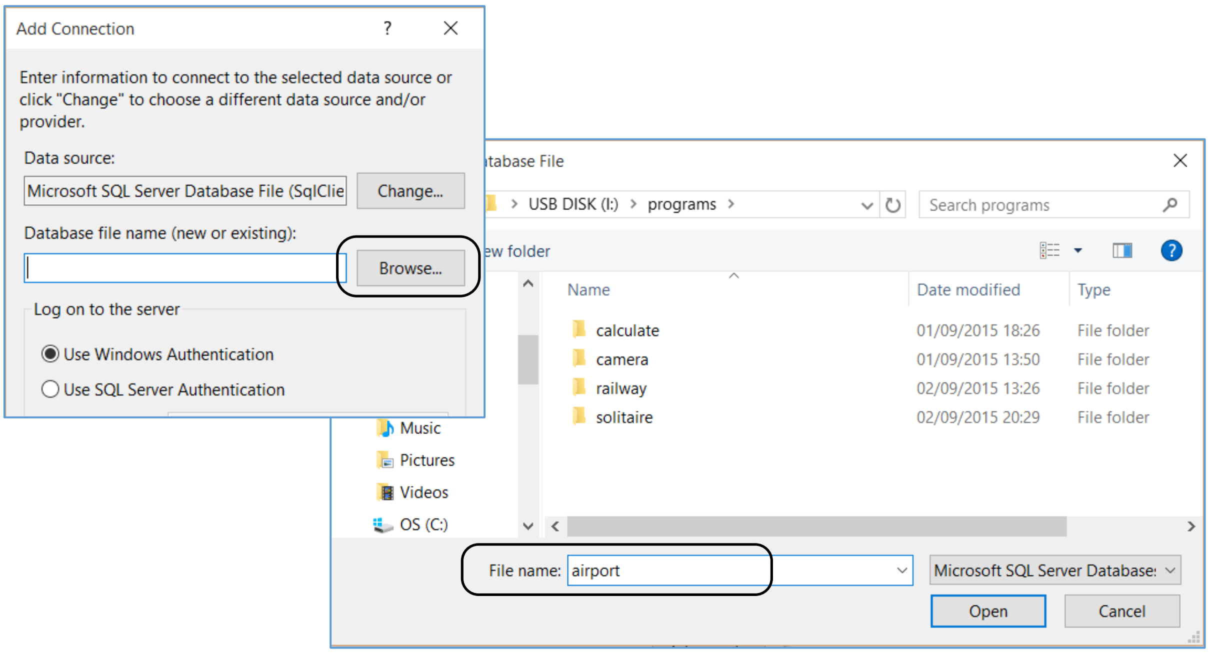
On the menu line, select '***View*** / ***Server Explorer***'


Start Page - Microsoft Visual Studio
FILE
EDIT
VIEW
DEBUG
TEAM
TOOLS
TEST
ANALYZE
WINDOW
Solution Explorer Ctrl+Alt+L
Team Explorer Ctrl+\, Ctrl+M
Server Explorer Ctrl+Alt+S
SQL Server Object Explorer Ctrl+\, Ctrl+S
Bookmark Window Ctrl+K, Ctrl+W
Call Hierarchy Ctrl+Alt+K
Class View Ctrl+Shift+C
Code Definition Window Ctrl+\, D
Object Browser Ctrl+Alt+J
Toolbox
Server Explorer
2013

Right-click on '***Data Connections***', then select '***Add Connection***'.

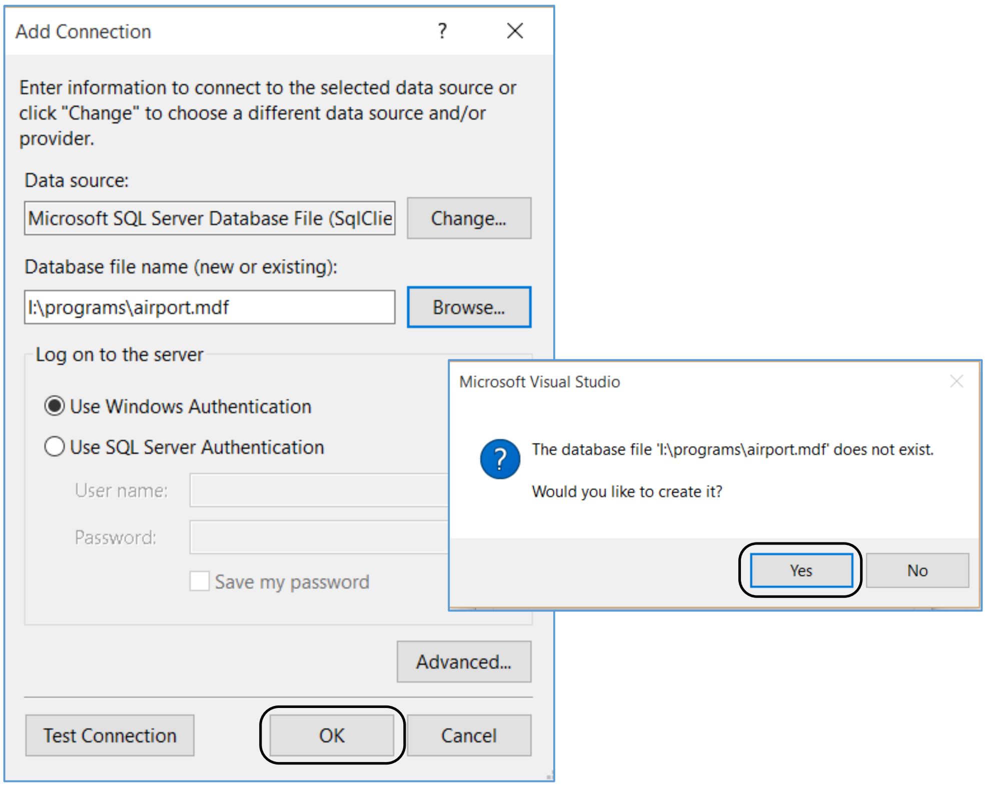

Server Explorer
Data Connections
Refresh
Delete
Add Connection...
Create New SQL Server Database
Properties
Choose Data Source
Data source:
Microsoft Access Database File
Microsoft ODBC Data Source
Microsoft SQL Server
Microsoft SQL Server Database File
Oracle Database
<other>
Description
Use this selection to attach a database file to a local Microsoft SQL Server instance (including Microsoft SQL Express) using the .NET Framework Data Provider for SQL Server.
Data provider:
.NET Framework Data Provider for S
Always use this selection
Continue
Cancel

Choose '***Microsoft SQL Server Database File***' as the Data source.


Add Connection
Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.
Data source:
Microsoft SQL Server Database File (SqlClie
Change...
Database file name (new or existing):
Browse...
Log on to the server
Use Windows Authentication
Use SQL Server Authentication
atabase File
USB DISK (I:) > programs >
Search programs
ew folder
Name
Date modified
Type
calculate 01/09/2015 18:26 File folder
camera 01/09/2015 13:50 File folder
railway 02/09/2015 13:26 File folder
solitaire 02/09/2015 20:29 File folder
Music
Pictures
Videos
OS (C:)
File name: airport
Microsoft SQL Server Database
Open
Cancel

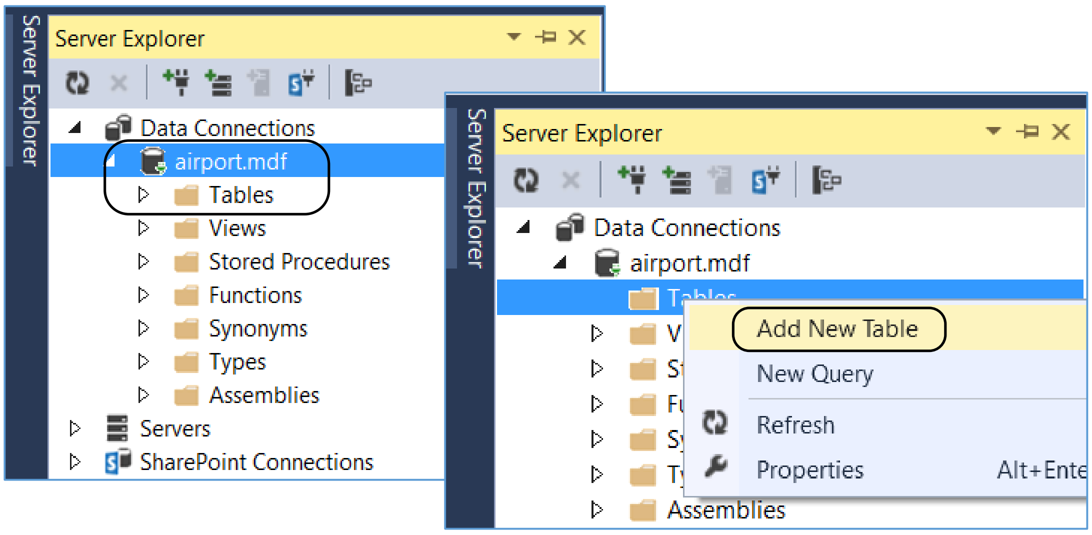
Use the '***Browse***' option to navigate to the location where your C# programs are stored. Give the file name '***airport***' for the database which will be created.

Click '***OK***', then answer '***Yes***' that you wish to create the database file:


Add Connection
Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.
Data source:
Microsoft SQL Server Database File (SqlClie
Change...
Database file name (new or existing):
I:\programs\airport.mdf
Browse...
Log on to the server
Use Windows Authentication
Use SQL Server Authentication
User name:
Password:
Save my password
Advanced...
Test Connection
OK
Cancel
Microsoft Visual Studio
The database file 'I:\programs\airport.mdf' does not exist.
Would you like to create it?
Yes
No

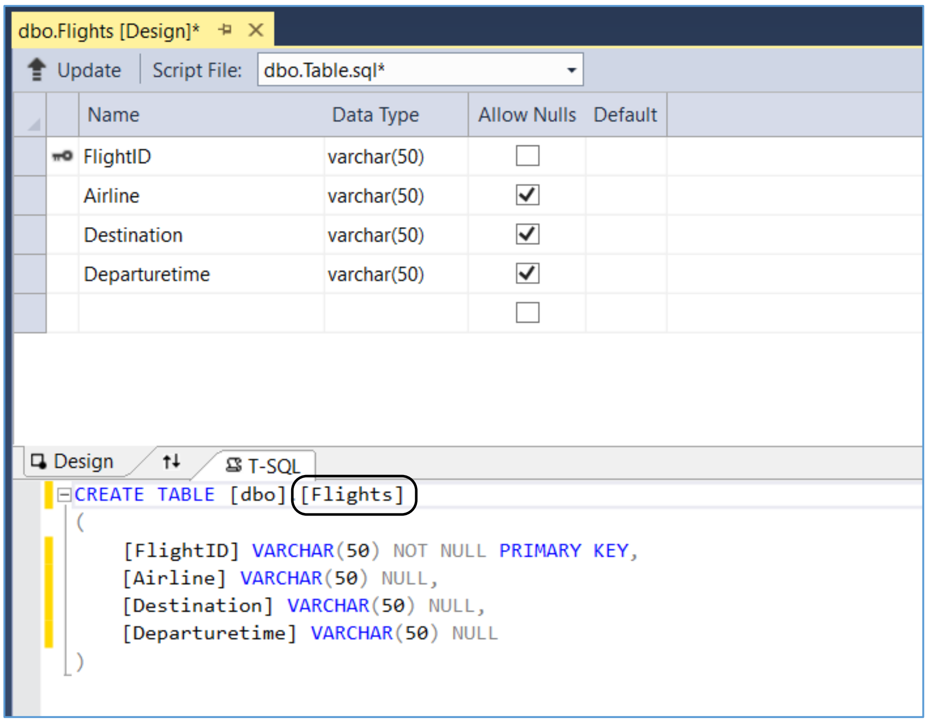
A database icon '***airport.mdf***' should now appear in the Server Explorer window. Click the small arrow symbol to the left of the icon to open a contents list.

Click-right on '***Tables***' and select '***Add New Table***':



We will include four fields in our database, to record ***flight ID***, the ***airline***, the flight ***destination*** and ***departure time***. To keep things simple, select '***varchar(50)***' as the Data Type. This allows up to 50 characters to be entered in the field, including both letters and numbers as necessary.

Notice that the first row has a small '***key***' icon. This indicates that it is the ***primary key field*** of the table. Every record must have a different unique value for the primary key. Leave ticks in the '***Allow Nulls***' box for the other fields.

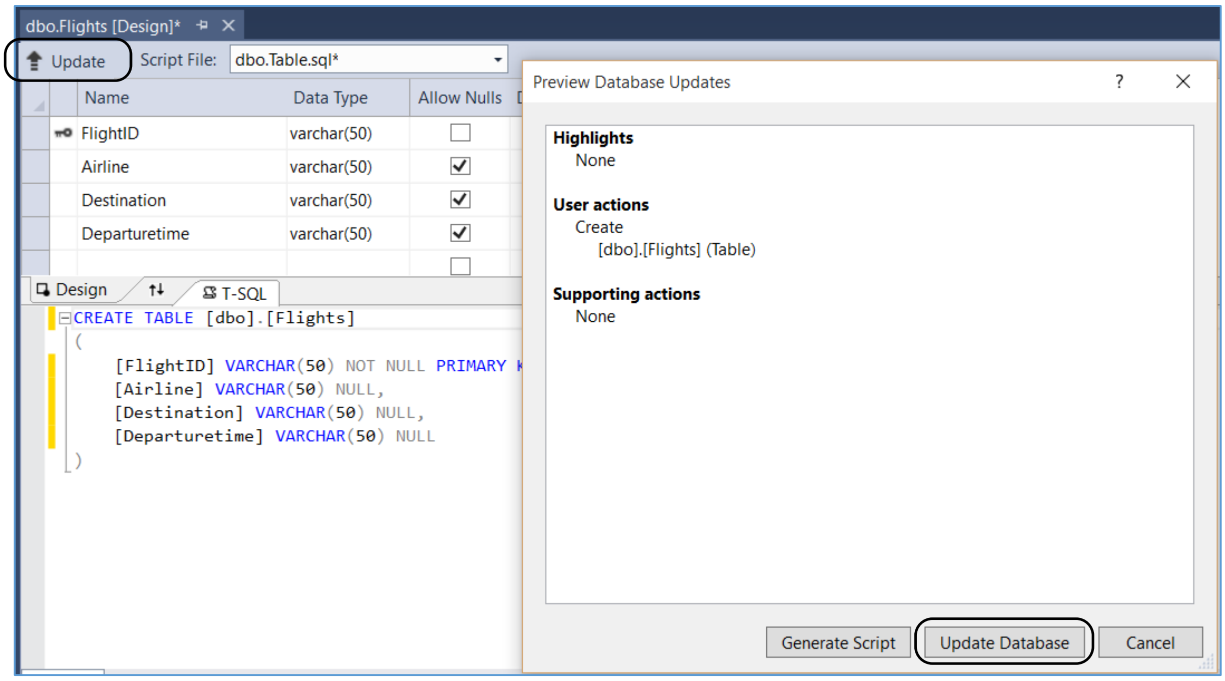


Name	Data Type	Allow Nulls	Default
FlightID	varchar(50)	☐	
Airline	varchar(50)	✓	
Destination	varchar(50)	✓	
Departuretime	varchar(50)	✓	
		☐	

```
CREATE TABLE [dbo].[Flights]
(
    [FlightID] VARCHAR(50) NOT NULL PRIMARY KEY,
    [Airline] VARCHAR(50) NULL,
    [Destination] VARCHAR(50) NULL,
    [Departuretime] VARCHAR(50) NULL
)
```

In the lower part of the page you will notice a section of ***SQL code*** which has been written by the program. This will be used by the computer to create the new table in the database. Go to the ***CREATE TABLE*** line and change the name of the table to '***Flights***'.

Click the ‘***Update***’ button above the list of fields. When the ***Database Update*** window appears, click the ‘***Update Database***’ button.



Name	Data Type	Allow Nulls
FlightID	varchar(50)	☐
Airline	varchar(50)	✓
Destination	varchar(50)	✓
Departuretime	varchar(50)	✓

```
CREATE TABLE [dbo].[Flights]
(
    [FlightID] VARCHAR(50) NOT NULL PRIMARY K
    [Airline] VARCHAR(50) NULL,
    [Destination] VARCHAR(50) NULL,
    [Departuretime] VARCHAR(50) NULL
)
```

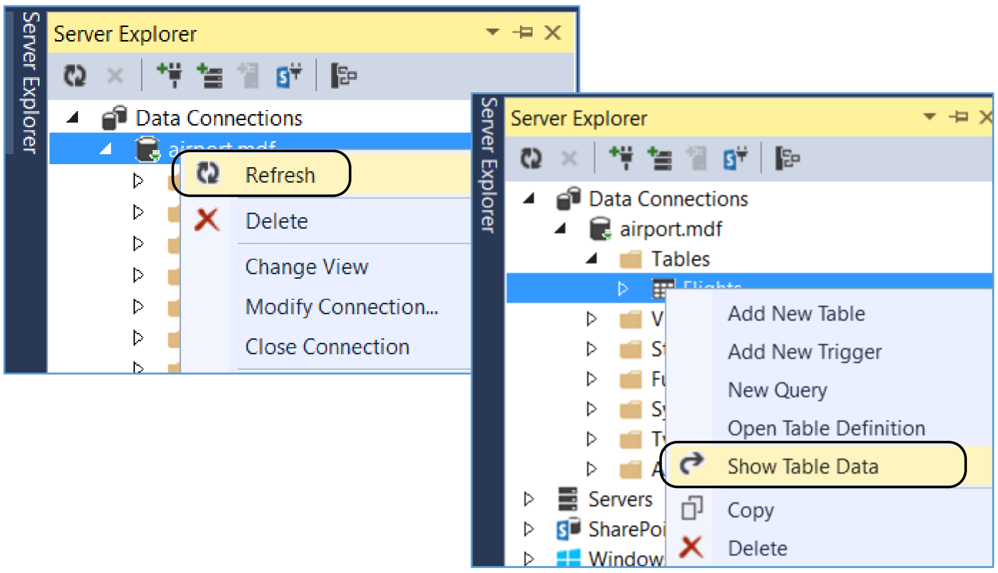
Highlights
None

User actions
Create
[dbo].[Flights] (Table)

Supporting actions
None

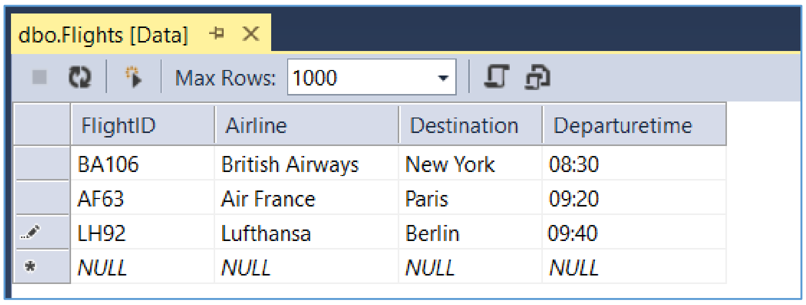
Close the ***Flights*** table design page and re-open the ***Server Explorer*** window. Right-click on airport.mdf and select '***Refresh***'.

Click the small arrow to the left of the ***Tables*** icon. The ***Flights*** table which you created should now be shown.



Right-click the ***Flights*** table, and select ‘***Show Table Data***’. An empty table will be displayed.

Add test data for a few aircraft departures, then click the cross on the tab to close the table.

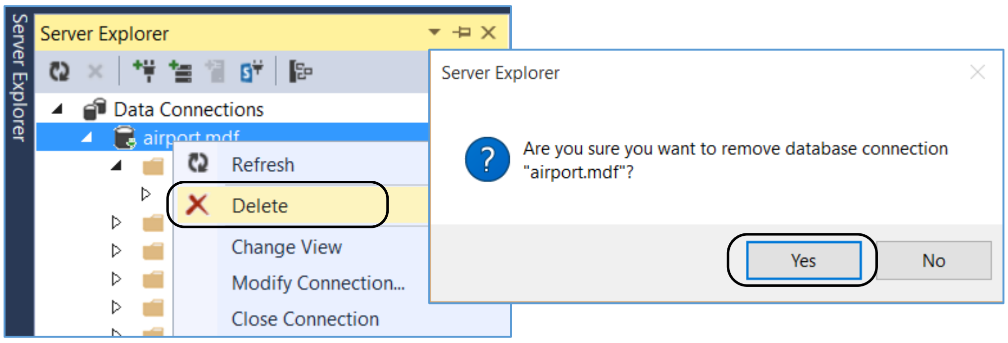


	FlightID	Airline	Destination	Departuretime
	BA106	British Airways	New York	08:30
	AF63	Air France	Paris	09:20
	LH92	Lufthansa	Berlin	09:40
*	NULL	NULL	NULL	NULL

Right click on '***Flights***' and use the '***Show Table Data***' option again, to check that your records have been saved into the database correctly. Click the cross to close the table again.

We are now going to write a C# program to access the ***Flights*** table from the database and display the information on screen. It is important to note that the program will not work correctly if the database is open in the ***Server Explorer*** window, so we will first close the database.

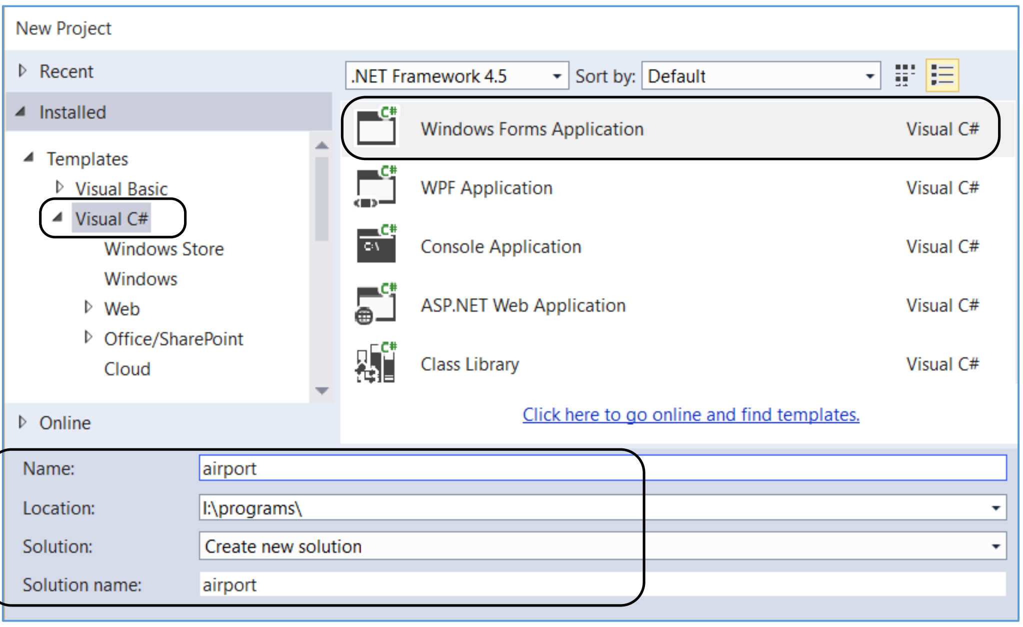
Go to the Server Explorer and right click the ***airport.mdf*** database icon. Select the '***Delete***' option, and confirm to remove the database connection by clicking '***Yes***'.



Are you sure you want to remove database connection "airport.mdf"?

We can now open the ***FILE*** option on the main menu and select '***New Project***'.

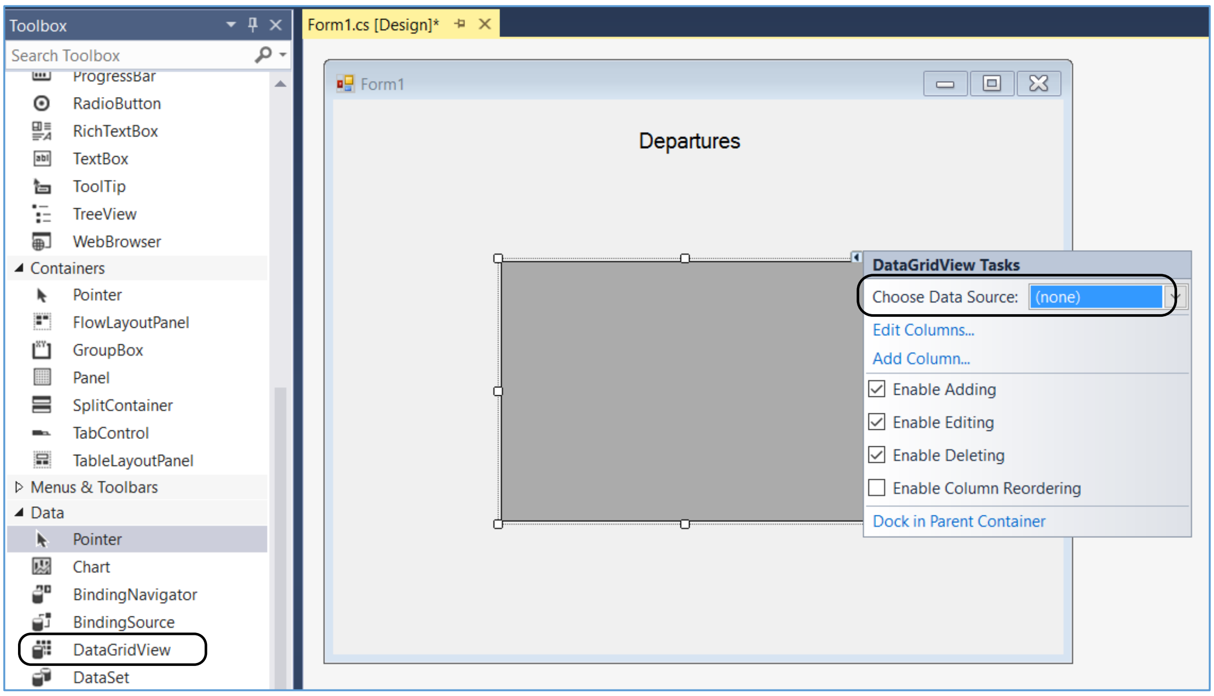
Choose '***Visual C#***' and '***Windows Forms Application***'. Give the name '***airport***'


New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows Store
Windows
Web
Office/SharePoint
Cloud
Online
.NET Framework 4.5
Sort by: Default
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Application Visual C#
Class Library Visual C#
Click here to go online and find templates.

Name: airport
Location: I:\programs\
Solution: Create new solution
Solution name: airport

Drag ***Form1*** to fill the design window. Open the Toolbox and add a '***Departures***' label as a heading.

Go to the '***Data***' section of the Toolbox and select the '***DataGridView***' component. Drag the mouse to position the DataGrid component on Form1 – it will appear only as an empty grey box at this stage. Do not change any of the entries in the list which appears. The '***Data Source***' should remain as '***(none)***'

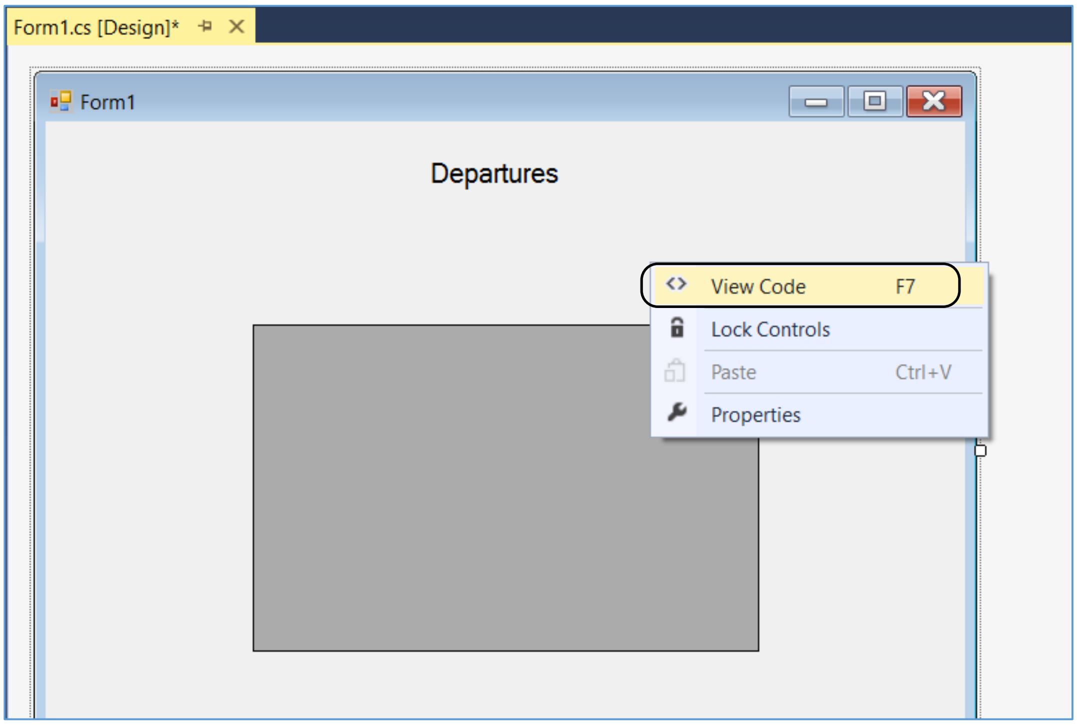


Choose Data Source: (none)

☑ Enable Adding
☑ Enable Editing
☑ Enable Deleting
☐ Enable Column Reordering

We can now write the program code to access the database.

Right-click on the form and select '***View Code***' to open the programming window



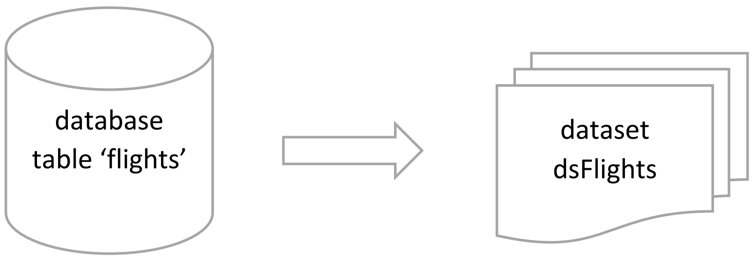
We will begin by creating a method for displaying the flights data. Add a line of code in the ***Form1( )*** method which will call this method when the program runs:

```
namespace airport
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            DisplayFlights();
        }

        private void DisplayFlights()
        {

        }
    }
}
```

Add a program line at the start of the ***Form1*** class which gives the name and pathway that you chose for the database on your particular computer. Notice that double back-slash characters must be used. This line has deliberately been placed in a position where it can easily be found, in case the program is transferred to a different computer and the pathway to the database needs to be changed.

```
namespace airport
{
    public partial class Form1 : Form
    {
        string databaseLocation = "C:\\C#\\airport.mdf;";

        public Form1()
        {
            InitializeComponent();
            DisplayFlights();
        }
```

Displaying the database records in a C# program is a two-stage process. Firstly the data has to be read-in from the database on the hard drive or other external memory device. The data is then stored as a ***dataset*** in the electronic RAM memory, where it is available for display on screen:



Add a ‘***using System.Data.SqlClient***’ directive to the start of the program, which will make available the code necessary to access the database.

```
using System.Text;
using System.Windows.Forms;

using System.Data.SqlClient;

namespace airport
{
```

We than add lines to the DisplayFlights method. Please note that the line beginning:

SqlConnection con = ...

should be entered as a single line of code with no line breaks.

```
    public partial class Form1 : Form
    {
        string databaseLocation = "C:\\C#\\airport.mdf;";

        public Form1()
        {
            InitializeComponent();
            DisplayFlights();
        }

        private void DisplayFlights()
        {
            DataSet dsFlights = new DataSet();

            SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;
                AttachDbFilename="+databaseLocation + "Integrated Security=True;
                Connect Timeout=30; User Instance=True");
        }
    }
}
```

The ‘***DataSet***’ line creates the storage area in the RAM memory to hold the records which we will input from the database.

The ‘***SqlConnection***’ line gives the computer the information which it needs to connect to the database. Apart from the location of the database, we are also telling it what software package to use (‘SQL EXPRESS’), and how long to keep trying to access data before giving up with an error message (30 seconds).

Things can often go wrong when accessing a database. To avoid the computer crashing, add a TRY...CATCH block to the DisplayFlights method:

```
private void DisplayFlights()
{
    DataSet dsFlights = new DataSet();
    SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;
        AttachDbFilename="+databaseLocation + "Integrated Security=True;
        Connect Timeout=30; User Instance=True");

    try
    {

    }
    catch
    {
        MessageBox.Show("File error");
    }
}
```

We now add lines of code to open the connection to the database and tell the computer what data we wish to retrieve. The command ‘***SELECT * FROM flights***’ means: ‘***bring back all the data from the Flights table***’

```
private void DisplayFlights()
{
    DataSet dsFlights = new DataSet();

    SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;
        AttachDbFilename="+databaseLocation + "Integrated Security=True;
        Connect Timeout=30; User Instance=True");

    try
    {
        con.Open();
        SqlCommand cmFlight = new SqlCommand();
        cmFlight.Connection = con;
        cmFlight.CommandType = CommandType.Text;
        cmFlight.CommandText = "SELECT * FROM flights";
    }
    catch
    {
        MessageBox.Show("File error");
    }
}
```

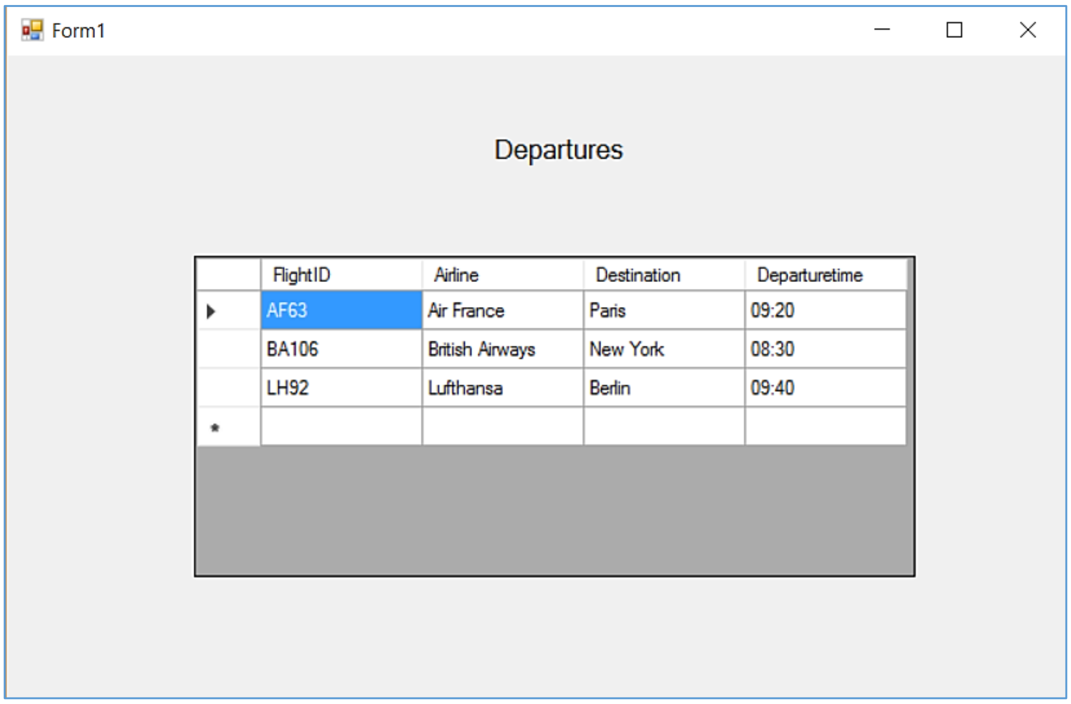
The final steps are to

- retrieve the data from the database,
- transfer it into the dataset,
- close the database connection,
- then link the dataset to the DataGrid on Form1 so that the records are visible

Add the lines of code shown below:

```
try
{
    con.Open();
    SqlCommand cmFlight = new SqlCommand();
    cmFlight.Connection = con;
    cmFlight.CommandType = CommandType.Text;
    cmFlight.CommandText = "SELECT * FROM flights";

    SqlDataAdapter daFlights = new SqlDataAdapter(cmFlight);
    daFlights.Fill(dsFlights);
    con.Close();

    dataGridView1.DataSource = dsFlights.Tables[0];
}
catch
{
    MessageBox.Show("File error");
}
```

Build and run the program. If all goes well, your database table should appear on screen. It may be necessary to return to the design view and adjust the size of the ***DataGrid*** component, so that all fields are visible.



FlightID	Airline	Destination	Departuretime
AF63	Air France	Paris	09:20
BA106	British Airways	New York	08:30
LH92	Lufthansa	Berlin	09:40

If you receive a '***File error***' message, check very carefully through the lines of code in the ***DisplayFlights( )*** method, and check the pathway which you entered for your database location. File handling is quite complicated, and it is easy to make an error.