

Using a Tab Control to show related data

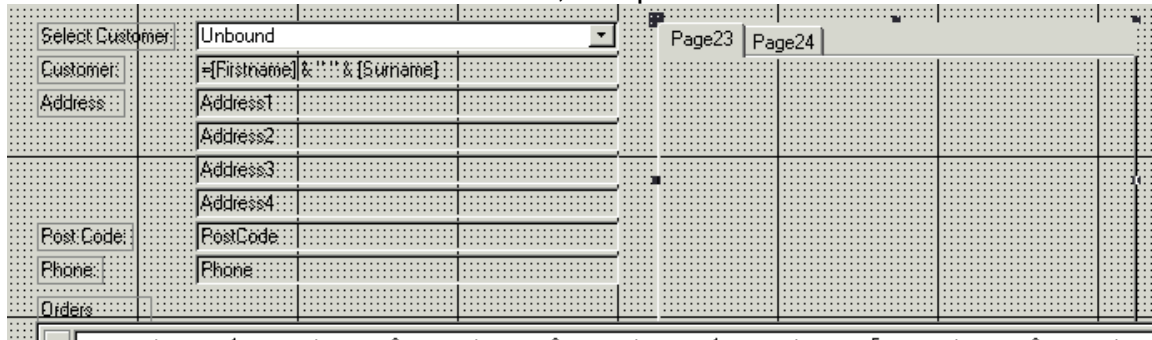
You can use a tab control to present several pages of information on a single form. This is especially useful when you're working with many controls (e.g. fields) that can be sorted into two or more categories.

For example, you might use a tab control on a CustomerOrderView form to separate personal information and order information.

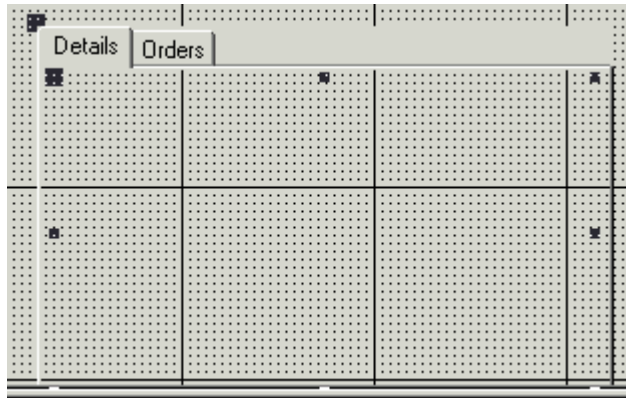
To achieve this:

Open the CustomerOrderView form in design mode.

Select the tab control from the Toolbox, and place it on the form.



Change the Name property of the first page to Details, and of the second page to Orders.



Cut the fields to appear on the Details page from the main form [highlight and ctrl-X]

Highlight the Details page

Paste the fields onto the Details page [ctrl-V]

Move the tab control into position:

Now cut the OrderView subform from the main form, highlight the Orders tab control page, and copy the OrderView subform onto it.

You will probably have to readjust the sizes of the main form (height if the detail: 4.5", width 8").

The two pages of the tab control on the form should now look like this:

Customer Orders

28 June 2001 Snowdon Double Glazing : Customer Orders

Select Customer: [Dropdown]

Details Orders

Customer: Richard Jones

Address: University of Greenwich
Wellington Street
Woolwich
London

Post Code: SE18 PF

Phone: 0181 331 8708

Customer Orders

28 June 2001 Snowdon Double Glazing : Customer Orders

Select Customer: [Dropdown]

Details Orders

Orders

Order Date: 26/06/2001 Payment Method: C

frmOrderLineView

Description	Quantity	Selling Price	Value
Front Door (Glazed)	1	£150.00	£150.00
Fron Door (Unglazed)	1	£100.00	£100.00

No Of Items: 2 No Of Order: 2 Total Value: £250.00

Record: 1 of 1

The first page, Details, show the fields from the tblCustomer table

The second page, Orders, has the OrderView form placed on it as subform, so it shows the related orders and orderlines.

As you can see, this gives a clearer, less cluttered display for the user.

More About Queries

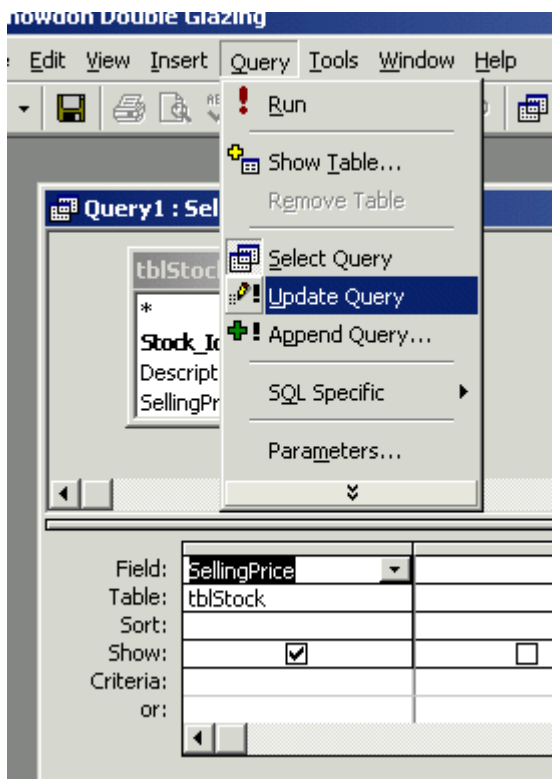
Update Query Example

This example, when run, will increase by 10% the selling price of all stock items which have a selling price of £50 or over.

Create a new query.

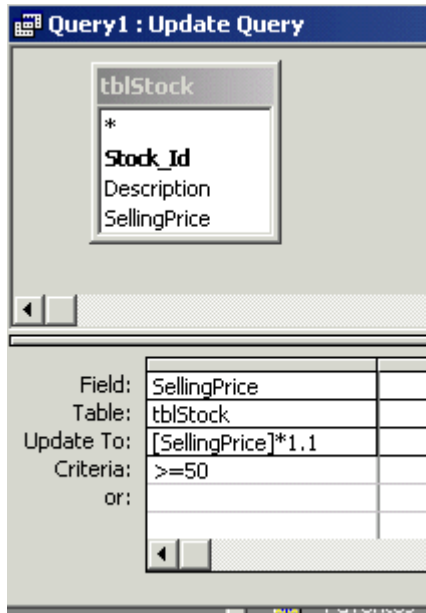
Add the stock table to the query.

Make it into an Update query by selecting this option from the Query dropdown on the menu bar.



Add the field SellingPrice to the QBE grid.

In the Update To row, type [SellingPrice] * 1.1.
In the Criteria row, type >=50



Note the square brackets surrounding SellingPrice. It is always a good idea to surround field names with square brackets.

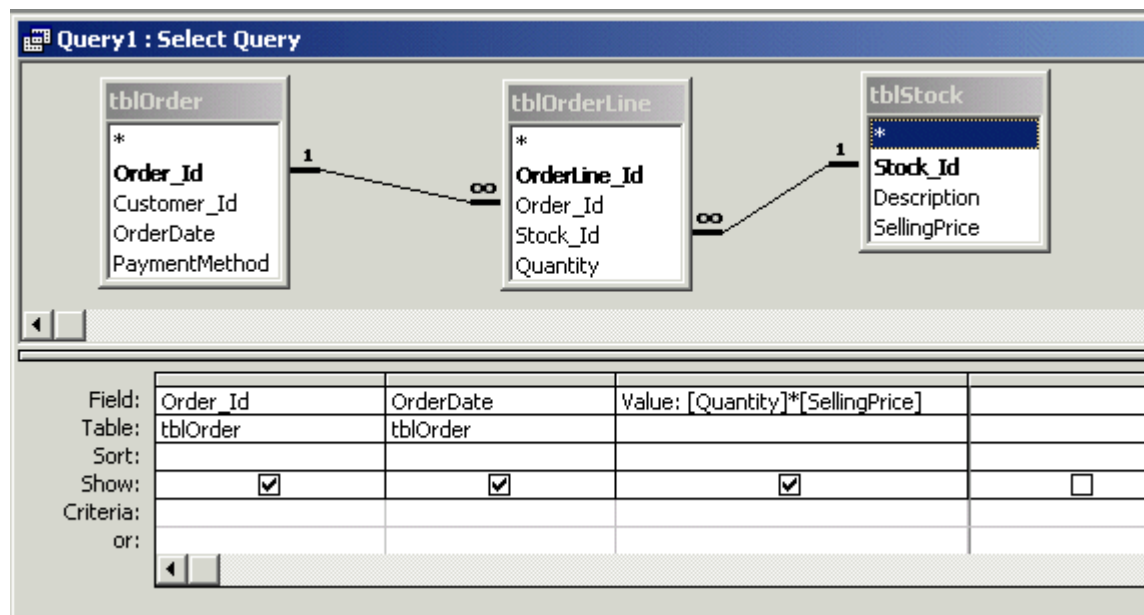
Before running the query please save the data either in another table called it tblstock2 or copy and paste the value in an excel spreadsheet so that you can compare whether the values have really been increased [updated].

Run the query and test the results.

Summing queries

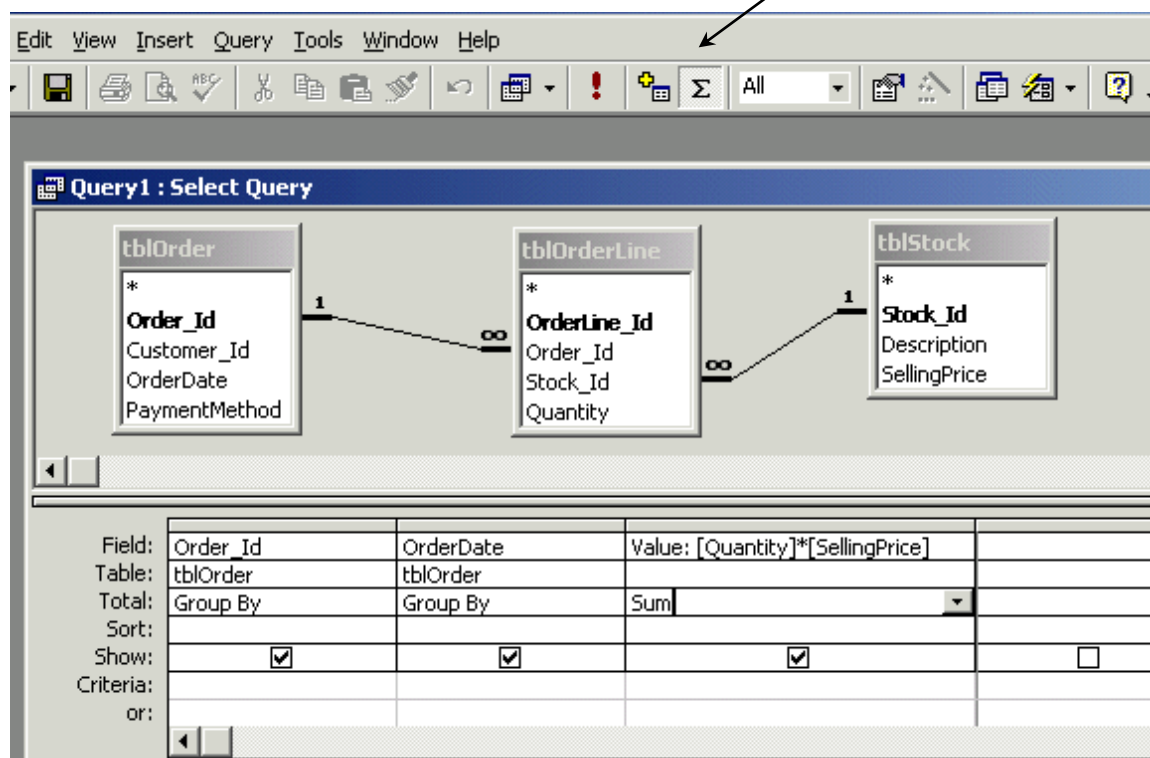
This is how to create a query, which will give the total value of all orderlines for each order.

Create the following query:



Now press the Totals icon, and adjust the Total field for the Value column:

Totals
Icon



Sa

Save the query as qryOrderValue and run it.

qryOrderValue : Select Query			
	Order_Id	OrderDate	Value
▶	18	26/06/2001	250
	19	26/06/2001	300
	20	27/06/2001	1750
	21	28/06/2001	150

Queries based on data on a form

This example show all orders, with their total value, which have an order date falling between the Start Date and End Date which can be entered on the form by the user.

28 June 2001 Snowdon Double Glazing : Order Query

Start Date: 01/06/2001
End Date: 30/06/2001

Orders	
Order Date	Value
26/06/2001	£250.00
26/06/2001	£300.00
27/06/2001	£1,750.00
28/06/2001	£150.00

Record: 1 of 4

When the user enters either the start date or the end date, the subform will show all the orders whose order date falls within the period.

Create a tabular form based upon the qryOrderValue query. Call it frmOrderValue.

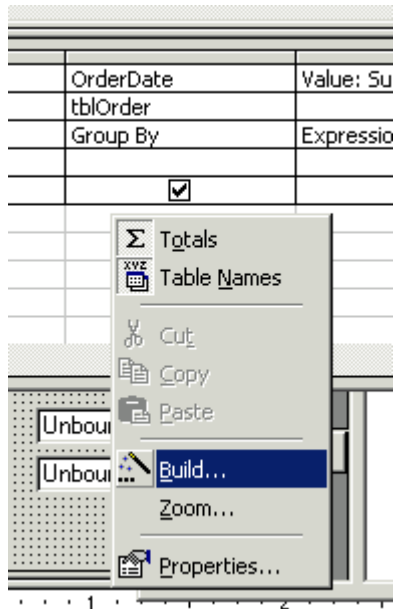
Create the main form with two unbound text boxes, formatted as Short Date. Name the text boxes StartDate and EndDate Call it frmOrderQuery.

Place the form frmOrderValue on the main form. There should be no primary / foreign key link between the two forms.

We now need to change the query which frmOrderValue is based on, so that it refers to the StartDate and EndDate text controls on the form.

Open qryOrderValue in design mode.

Click with the right button in the criteria field of the OrderDate column, and choose Build.

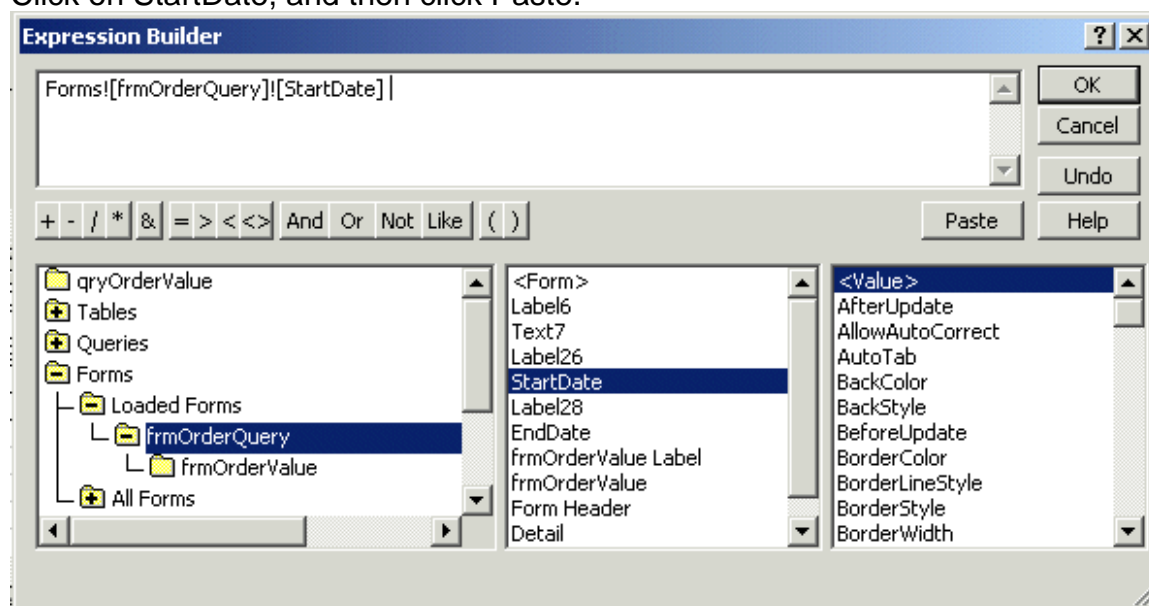


You are now in the Expression Builder. This makes it easier to get the syntax right when referring to different objects such as forms and text boxes.

In the left pane, click on Forms / Loaded Forms / frmOrderQuery

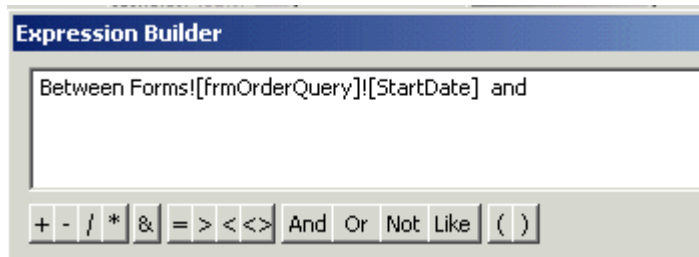
The middle pane shows all the controls available on that form.

Click on StartDate, and then click Paste.

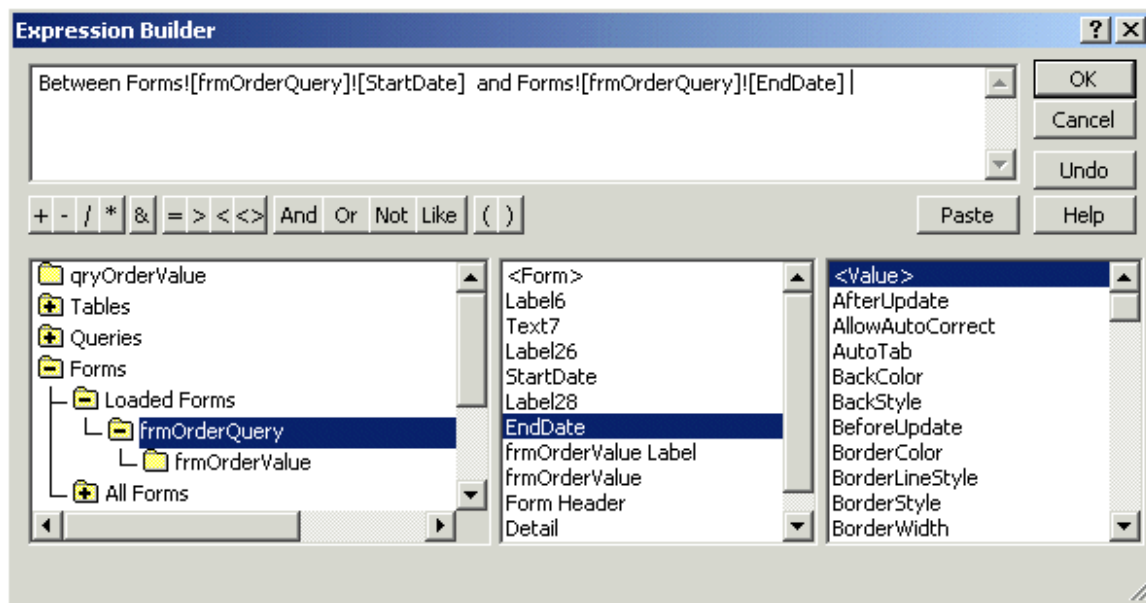


You can see that the correct syntax for referring to the StartDate text box on the frmOrderQuery form has now been produced.

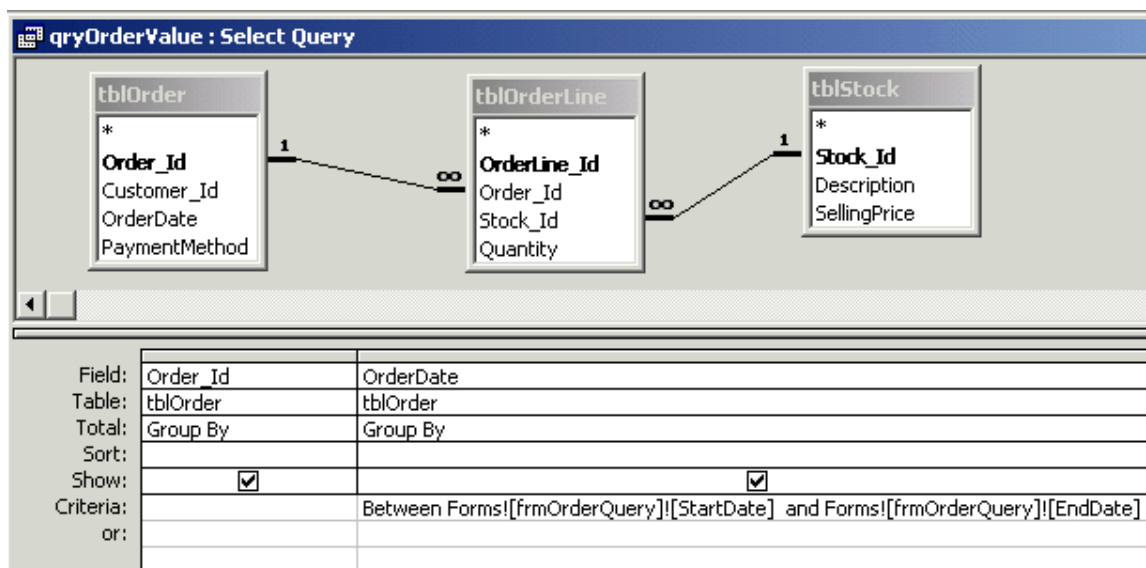
Edit this line:



And then add the reference to the EndDate text box:



Press OK and you will return to the query builder.



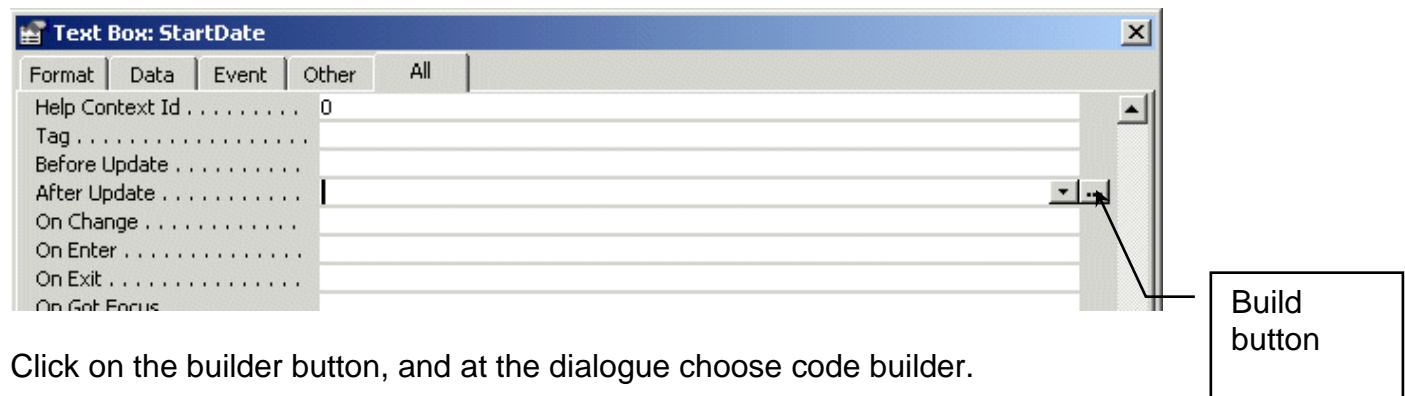
The criteria for OrderDate has now been built. The query will now show only those orders where the order date is between the dates entered by the user in the StartDate and EndDate text boxes.

Save the query.

The query will run when the form is opened. If the user changes the values of StartDate or EndDate, the query will need to be forced to run again to reflect the changes.

To do this, both the StartDate and the EndDate controls need code attached to their After Update events:

Open the property sheet of StartDate, and find the After Update event:



Click on the builder button, and at the dialogue choose code builder.

You are now in the Visual Basic editor

Edit the code to look like this:

```
Private Sub StartDate_AfterUpdate()  
Me.frmOrderValue.Requery  
End Sub
```

Exit the editor, and do the same thing with the EndDate text box.

We will be looking at using Visual Basic in more detail later in the course.

Invoking the Requery method on a form results in the underlying query to be re-run.
So this code forces the query underlying the subform to run when the dates are changed.

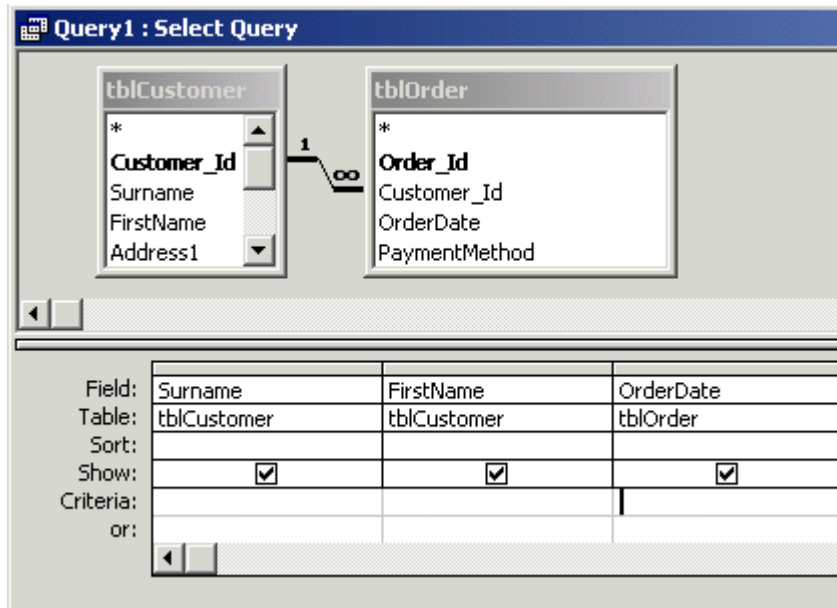
Test the form with your own data.

=====

Outer Join Queries

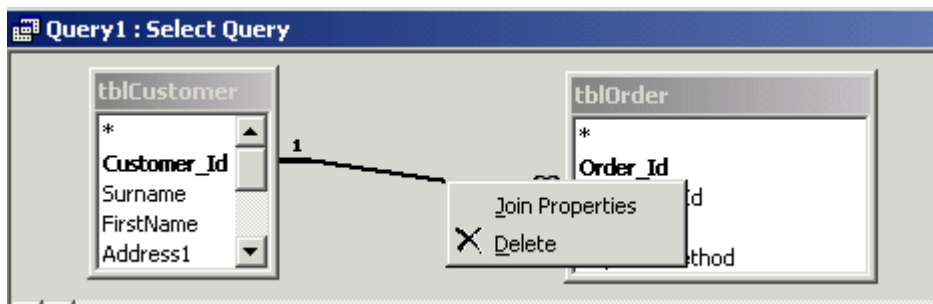
Enter a customer into your database, but do not enter an order for this customer.

Build the following query:



Run the query. Your customer with no order does not appear. This is because the query only shows records from tblCustomer having at least one corresponding record in tblOrder.

We can change the query to show all customers, regardless of whether they have orders. Right click on the relationship line between the tables.



And choose Join Properties.
Choose the second option:

Join Properties

Left Table Name: **tblCustomer** Right Table Name: **tblOrder**

Left Column Name: **Customer_Id** Right Column Name: **Customer_Id**

☐ 1: Only include rows where the joined fields from both tables are equal.

☒ 2: Include ALL records from 'tblCustomer' and only those records from 'tblOrder' where the joined fields are equal.

☐ 3: Include ALL records from 'tblOrder' and only those records from 'tblCustomer' where the joined fields are equal.

OK Cancel New

And press OK.
Run the query.

	Surname	FirstName	OrderDate
▶	Jones	Richard	26/06/2001
	Jones	Richard	26/06/2001
	Jones	Richard	27/06/2001
	Jones	Richard	28/06/2001
	Knight	Joan	
	Adams	Tom	
	Order	No	
*			

Record: 1 of 7

You can see in my example, that where a customer does not have an order, the OrderDate field is null.

Use this to only show customers who do not have orders.

CHECKPOINT: SDGCheck6