Tables & Relationships in MS Access 2010

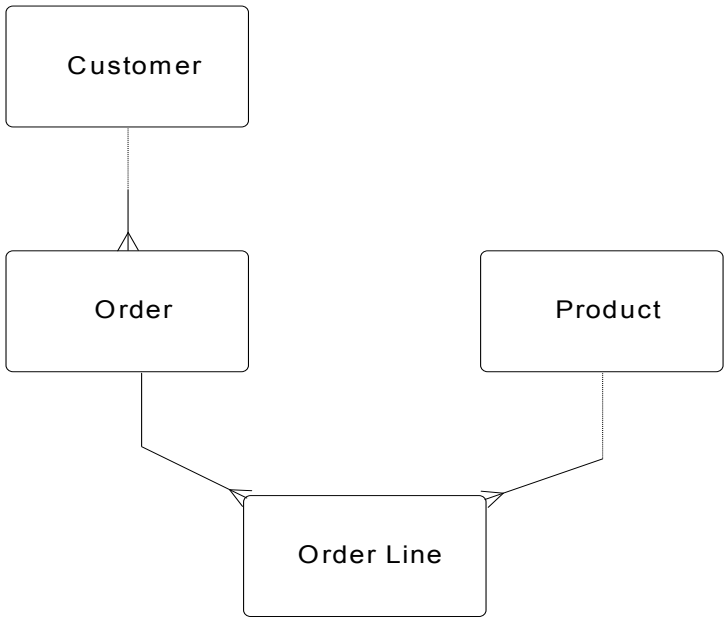
To illustrate the processes involved in designing and implementing a database, you will be developing a simple database for a retailing business - a supplier of hand and power tools to the building trade.

ACME TOOL SALES

Acme Tool Sales stocks everything from screwdrivers to cement-mixers. The warehouse keeps a large inventory of **products** in stock, and the product records have to be kept up to date. The business has many regular **customers**, mainly trades people, who have accounts with Acme. The sales team take **orders** either by telephone or in person, and fill out online order forms which may contain several **lines of products for one order**.

By using the process of data modelling we can arrive at the required structure of the database. This involves drawing an initial Logical Data Model then normalisation of the sample documents given below to validate the model. This has already been done and the results are provided below but you should check this data modelling for yourself.

Initial Logical Data Model



The staff of Acme want to be able to enter and view the details for customers and products in a user friendly manner. They want to be able to enter new sales transactions

and view previous transactions, and they want to be able to know immediately if an order item is in stock.

The sales team need to be able to print mailing labels and produce a stock catalogue to keep customers up to date. They need to be able to select customers by post code, or sort by surname, or address customers by first name in a mail shot.

The management want to be able to print ad hoc reports giving information on Acme's performance, e.g. total annual sales, fastest selling stock, best sales month, best customers etc.

From this brief description, it is possible to define three main groups of data that need to be managed:

1. **Products**

Example Stock Card from the Warehouse:

Product ID - HP-0001
Description - CEMENT-MIXER
Unit price - £150.00
Units in stock - 50

2. **Customers**

Example Customer Record:

Customer ID - 001C0
Business Name - D.B.S.
Address 45, Swanlake Avenue,
Ranton, RY2 6NR
Contact - Mr Dave Fowl
Tel - 0566 23498745
Note - Good payer

3. **Sales transactions**

Example Order Form:

Order ID - 1 **Date** - 02/01/99
Customer ID – 002JC
Customer Name Dovetail Joiners
Address Unit 4 Forest Business Park
Nopoint,
NP3 1GH

Prod ID	Description	Quantity	Unit Price	Cost
HT0001	Tenon saw	1	£10	£10
HT0005	Chisel (2)	1	£5	£5
PT0009	Plane	1	£34	£34

Customer Name and *Business Name* are synonyms.

The data fields (attributes) on the example order form include *Cost* and *Total* but these can be calculated from other data so we do not need to store them.

You should *not* include any fields that can be calculated from other data in the **same table**, such as cost, from quantity and unit price. This would result in data redundancy.

Therefore the complete data set will be as follows:

Order ID, Order Date, Customer ID, Business Name, Address Line 1, Address Line 2, Post Code, Contact Surname, Contact First name, Contact Title, Phone Number, Notes, Product ID, Product Description, Quantity, Unit Price, Units in Stock, Supplier,

This is a simplified list for the purposes of the ACME database exercises.

Normalise these attributes and validate the Logical Data Model.

Having normalised this data and successfully validated the Logical Data Model, we can also assign the attributes to their appropriate tables (relations/entities).

Customers

Customer ID, Business Name, Address Line1, Address Line2, Post Code, Contact Surname, Contact First name, Contact Title, Phone Number, Notes

Products

Product ID, Product Description, Unit Price, Units in Stock, Supplier

Orders

Order ID, Order Date, Customer ID *(foreign key relating the order to a customer)*

Order Lines

Order ID *(relates it to an Order)*, Product ID *(relates it to the Product details)*, Quantity

In addition to identifying the fields, you have to specify what kind of data the field will contain because Access treats different types of data in different ways. The data types are listed below.

Note: Hyperlink, Attachment and Lookup Wizard are also available as data types in Access, but we will not use them for setting up our database.

Data Types in Access

Text Text up to 255 characters, or numbers which are not used for calculations e.g. telephone numbers, serial numbers etc. Note that Access does not reserve unused parts of text fields, so set size as large as the predicted maximum.

Memo Similar to Text but used for variable amounts of text up to 1 gigabyte of characters (of which 64k characters can be displayed in a control), e.g. miscellaneous personal notes.

Number Numerical data that will be used in calculations (except currency). Size can be 1, 2, 4 or 8 bytes corresponding to Byte, Integer, Long Integer and Single or Double precision floating-point numbers. Replication ID (16 bytes) is only of interest if you intend to maintain duplicate databases. Decimal definitions can also be used.

Integers are smaller whole numbers from -32,768 to +32,767 and can be used for many purposes giving greater efficiency than long or floating-point numbers.

Date/Time Dates and times e.g. Order Date. Note that both date and time are stored in a single field.

Currency Money values e.g. Unit Price. Uses fixed point in calculations, which is quicker than floating point, so use this definition if appropriate.

AutoNumber Automatic numbering of records starting at 1 or can be set to Random Value. **CANNOT BE UPDATED**! Size is 4 bytes, type Long Integer.

Yes/No Any field which has only two values such as Yes/No, True/False, On/Off, Male/Female etc.

OLE object e.g. a graphic, a spreadsheet etc. - it will be LINKED or EMBEDDED in the record if it is BOUND to the database data.

NOTE: UNBOUND OLE objects can be used in a form or report, independent of the data, e.g. the company's logo. OLE objects and Memos cannot be indexed.

Hyperlink Link to an Internet resource. 64,000 characters or less

Attachment A special field that enables you to attach external files to an Access database.

Lookup

Wizard Displays data from another table.

Rules for naming fields 1: Access allows names of up to 64 characters with spaces. Choose meaningful names that make it easy to remember what the data is.

Rules for naming fields 2: Do not give the same name to more than one field. Do not use the characters (**.**) (**!**) or (**[ ]**) because these have specialised uses in Access.

Complete list of Fields & Data Types for ACME Database

Below are listed the four tables, their fields, and data types for the ACME database. Refer to this list when creating the tables.

Table	Field	Data Type
Customers:	CustomerID	Text
	BusinessName	Text
	AddressLine1	Text
	AddressLine2	Text
	PostCode	Text
	ContactTitle	Text
	ContactFirstname	Text
	ContactSurname	Text
	PhoneNumber	Text
Products:	ProductID	Text
	ProductDescription	Text
	UnitPrice	Currency
	UnitsInStock	Number (Integer)
	Supplier	Text
Orders:	OrderID	AutoNumber
	CustomerID	Text
	OrderDate	Date/Time
Order Lines:	OrderID	Number (L. Integer)
	ProductID	Text
	Quantity	Number (Integer)

Creating the ACME Database

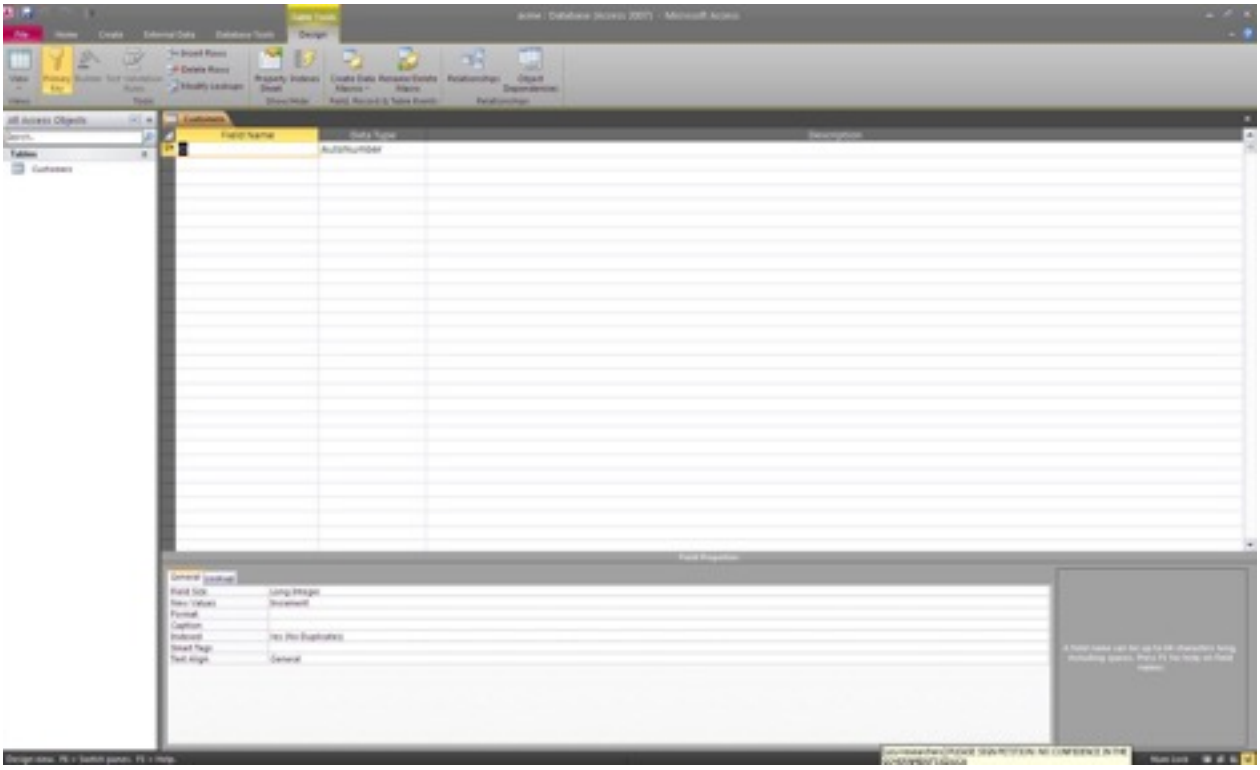
We will use our data model to build the database. Start Access. Click on the Blank Database icon and call the database **ACME** retaining the .accdb file extension.

PLEASE NOTE *There have been many different versions of Access. They are not all completely compatible. If you want to work off campus as well as in the labs make sure that you can still use your database with the Access version in the labs.*

The appearance of various parts of Access changes with each new version and the screen shots used in these documents may look a little different from the version you are using. You may need to interpret these differences but that is good for your learning process! You should try to understand what you are trying to achieve and not simply follow instructions blindly.

Creating the Customer Table

From the ACME database window create a new table using Design View. This can be obtained by clicking Design View on the View icon or right clicking the Table1 tab or clicking on the Design View icon at the bottom right-hand corner. You are asked immediately to name the table; name it Customers.



The Table Design window has all the usual window features - control box, maximise and minimise buttons, and a scroll bar. In addition, in the lower right of the display there is an information window which provides a description of the selected area.

- ➢ click the pointer in the first Field Name column (as above) and type **CustomerID**
- ➢ Press the **Tab** key to move to the Data Type column, click on the arrow to open the drop down list and select **Text**

If you had fields with obscure names you would now type in an explanation of the field in the Description column, but since these ACME fields are self-explanatory, you may leave out that part. Note that "Text" is the default value if you skip the Data Type column.

- ➢ Click in the next blank field name and type in the next field name and data type from the list shown above for the Customers table. Continue in this way until all the fields have been added (for the Customer table only at this stage).

Setting the Primary Key

The Primary Key is a field (or combination of fields) that uniquely identifies each record in a table. In the Customers table, the CustomerID is unique to each customer so it is this field that will be designated the Primary Key. When you created the Customers table the first field (in this case CustomerID) was set already as the Primary Key, so you should not have to set the Primary Key. If you need to set a Primary Key:

- ➢ click in the required field (press Ctrl and click any other fields for multiple field keys)
- ➢ click the **Primary Key** icon in the **Tools** group on **Design** ribbon

An image of a key appears on the selector box of the selected field(s).

If you place the key in the wrong field, simply repeat the instructions above in the correct field. The key will move to the new field.

Saving Your Customer Table

The first time you save a table you will be presented with a dialog box in order to give it a name. It will then automatically be saved as an object within the ACME database. When you are satisfied that you have entered the details correctly, save the Customers table by right clicking the table's tab and clicking **Close**. Click **Yes** to save the changes.

If you try to save a table without a Primary Key, Access will prompt you with a message "There is no primary key defined. Create primary key?" Choose the Cancel button, set the Primary Key and then save again.

To re-open the Customers table in design view, right click the table name in the Navigation Pane and click **Design View** or double click the table to open it and then right click the tab to change to **Design View**.

Field Properties

There are many properties that can be defined for a field, affecting the data in the field, its appearance or maybe the way it will be displayed in a datasheet. Some of these are used in Forms or Reports rather than Tables. You can get a comprehensive list of these in HELP. Here is a summary of commonly-used Field Properties:

Field Size: Controls the size of a Text field or the size of a Number field. For other data types that have fixed field sizes, this property is not present.

Format: Controls how data is ***displayed***, e.g. convert all letters to uppercase – and applies to datasheets, forms and reports. See examples of Formats below. Note: Format does not affect the way data is ***stored*** in the database.

Input Mask: Used to control how users enter data. Uses control characters similar to Format. See below for examples.

Caption: Provides an alternative field name if required.

Default Value: Specifies the value that automatically appears in a field when you create a new record.

Validation Rule: Specifies requirements for data entered into an entire record, an individual field, or a control on a form or report.

Validation Text: Specifies the error message that appears if a user violates a validation rule.

Required: If this is set to YES then a record cannot be saved unless there is a value entered in this field; e.g. you *must* have a Customer ID

Allow Zero Length: For text fields, if set to YES then the user can enter ““ which means there is no data for this field, e.g. a customer does not have a telephone number. Note that in Access you will have to test specifically for the data to be "null" or "zero length string", maybe using an "if" expression. This rule applies to tables and forms controls. See Access HELP, FORMAT PROPERTY for more details.

Indexed: If a field is indexed (placed in numerical or alphabetical order) it speeds up searches. Key fields are indexed automatically – Yes (No Duplicates) for simple keys and Yes (Duplicates OK) for compound keys and other fields such as surnames.

Formats

Custom formats display data in the way you have specified, regardless of how it is entered. Custom formats are designed using special symbols as place markers. Examples used for the different data types are shown in the following lists.

Text	**Indicates**
@	character required; e.g.(@@@)@@@ : (077)534
>	changes all text in field to upper case
<	changes all text in field to lower case

Number

#	place for digit (leading or trailing zeros not shown)
0	place for digit (leading or trailing digits shown) e.g. ##,###.00 : 56.83 5.70 5,243.04 100.00

Date

d-m-y	3-10-08
dd/mm/yy	03/10/08
dddd d mmmm yyyy	Friday 3 October 2008
Time	h=hours, m=minutes, s=seconds, am/pm = 12 hr format e.g. hh:mm:ss = 12:08:45, h:mm AM/PM = 4:45 PM
Yes/No	e.g. ;“Smoker”;“Non-Smoker” displays Smoker for true and Non-Smoker for false.

Customising Input Masks

The table below shows the symbols that can be used to create masks (or templates) for serial numbers, postcodes, telephone numbers etc.

Mask character	**Indicates**
0	single digit – obligatory
9	single digit – optional
#	single digit, space, - or + sign – optional
L	single letter – obligatory
?	single letter – optional
A	single letter or digit – obligatory
a	single letter or digit – optional
&	any single character or space – obligatory
C	any single character or space – optional
.,:;-/	decimal point, thousands, date and time separators
>	characters to right appear in upper case
<	characters to right appear in lower case
\	character following is not to be interpreted as a mask character

Example Mask	**Sample Value**
\IBM-000-0000->L	IBM-372-6839-G
00000-000000	01752-232345

>L<???????????? Smith

**More information on Input Masks can be found on Microsoft's website:
http://office.microsoft.com/en-us/access-help/control-data-entry-formats-with-input-masks-HA010096452.aspx

Setting the Customer Field Properties

Now you can go through each field in the Customer table and define some field properties.

- ➢ click in the field *CustomerID* then press the **F6** key to move to the Field Properties box
- ➢ Change the **Field Size** property to **5**
- ➢ Change the **Required** property to **Yes**
- ➢ Change the **Indexed** property to **Yes (No Duplicates)** because you do not want two customers with the same ID.

ACME uses a standard form of numbering for its customers consisting of three digits followed by two letters to indicate the type of trade e.g. 001PB. The letters PB indicate a plumber. CustomerID is a Primary Key field so it is a good idea to ensure accurate entry by means of a customised Input Mask.

- ➢ make the **Input Mask** property **000>LL**

Do not confuse zero (0) with a capital O when creating an input mask.

- ➢ Continue with the rest of the fields with properties as follows:

BusinessName .....**Field size:** 50
Caption: Company
Indexed: Yes (Duplicates OK)
AddressLine1 .....**Field size:** 30
AddressLine2 .....**Field size:** 30
PostCode .....**Field size:** 8
Format: >
Indexed: Yes (Duplicates OK)
ContactTitle .....**Field size:** 4
Caption: Title
ContactFirstname .**Field size:** 30
Caption: Forename
ContactSurname ..**Field size:** 30
Caption: Surname
Indexed: Yes (Duplicates OK)

PhoneNumber.....**Field size:** 15

➢ Save these changes to your table by either right clicking the table's tab and clicking **Save** or clicking on the  icon in the Quick Access toolbar.

➢

At this stage you have no data in the table so you can make changes to the structure without concern about the effect on existing records. Beware of making changes *after* entering records. For instance, if you reduce the field size property you may truncate data already in the table. One further change you will make now is to add an extra field. The ACME management keep personal records about their customers and also log their phone calls. This information is variable in quantity so is best stored in a Memo field.

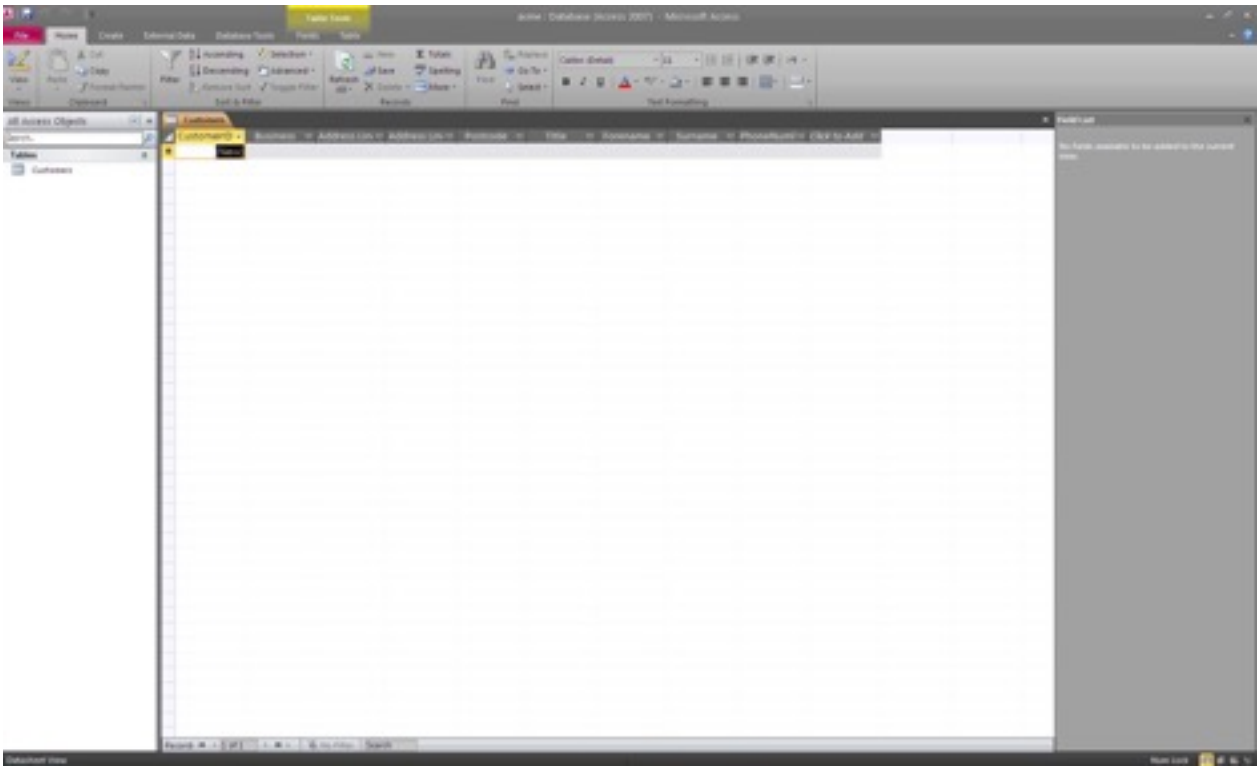
➢ enter an extra field below *PhoneNumber* called *Notes,* data type: *Memo*. (There are no field properties to set for this field.)

➢ Save the table again.

Now you have created your Customer table and set the field properties, you can start to enter a sample record. At the moment you are in the Table design view. To enter data you need to see the table in a rows and columns format called the **Datasheet View**. To change to the Datasheet View:

➢ either right click the table's tab and click Datasheet View or click the **View** group button

The datasheet view displays the field names as column headings, with the cursor in the empty CustomerID field, the row is indicated by an asterisk (*), ready to receive data input.



Entering a record in the datasheet

➢ type in the following record, pressing ↵ or → or the Tab key to move from field to field (there is no entry for the Notes field)

(Customer ID) 005PB, Martins Plumbing Ltd, 222 Coburg Road, Plymouth, PL5 6HD. Mr John Martin, Tel. 01752 354672

When you reach the end of the record and press ↵, the record is saved automatically and the cursor moves to the start of the next blank record.

If you make an error, press ESC to undo the current field.

➢ enter this second record

Customer ID 006PB, Jackson Plumbing Ltd, 54 Drakes Road, Plymouth, PL3 5AS. Mr Fred Jackson, Tel. 01752 645187

Deleting a record from the datasheet

To delete a record from the datasheet you must first highlight it. Do this to the Martins Plumbing record:

➢ place the pointer on the record selector box (at the start of the record) and click.

The whole record will now be outlined and highlighted.

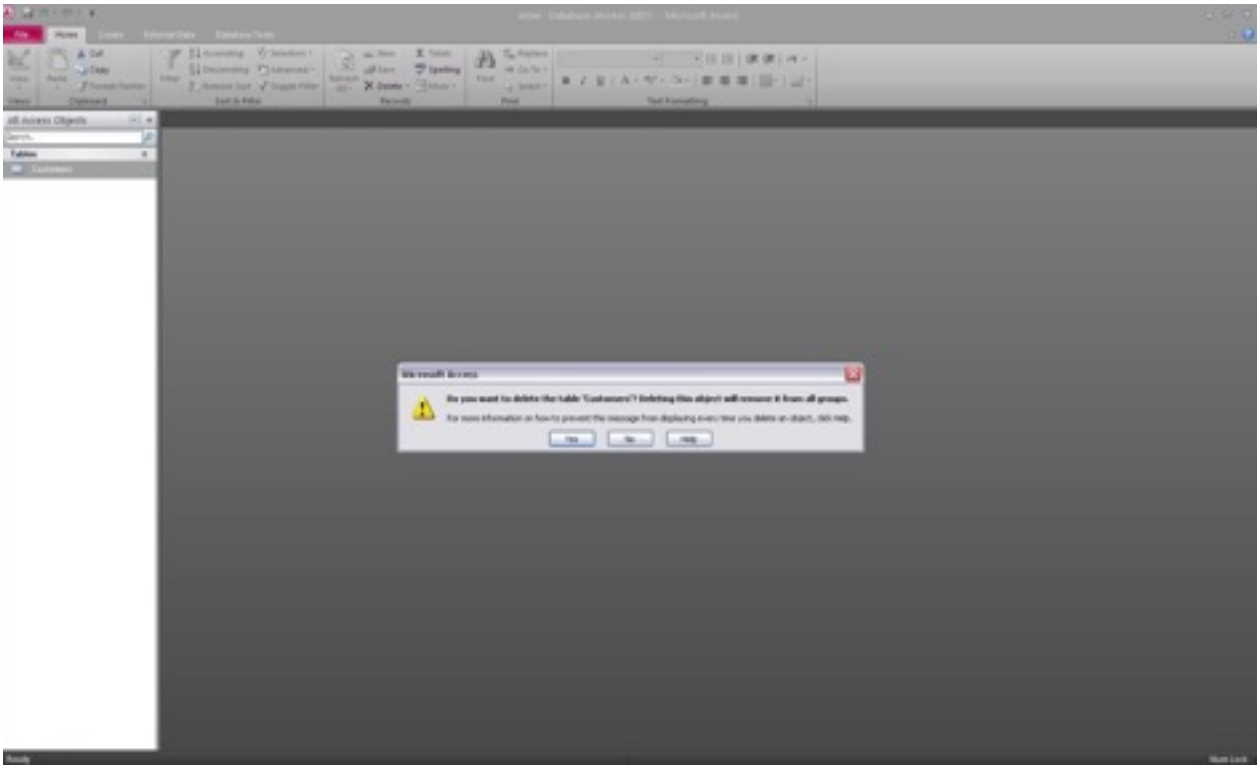
Press **Delete** and then click **OK** to confirm your action in the dialog box.

To demonstrate the capabilities of Access you need to have some data to work with. Because entering lots of records is tedious and time consuming, a customer table containing data has been prepared in advance. Appending records to your table from another source involves techniques that have not yet been introduced. Instead what you will do is *delete* the table you have just created, and *import* a new Customer table complete with records. The fields and properties of the new table are identical to the one you have just created.

Deleting a table or other object

➢ to close the customer table, right click the table's tab and click **Close** – this will automatically save any changes

➢ in the Navigation Pane, right click the **Customers** table and select **Delete,** then click **Yes** to confirm



Importing tables

Before you import, download the Acme database from the portal to a convenient location (such as your U-drive.)

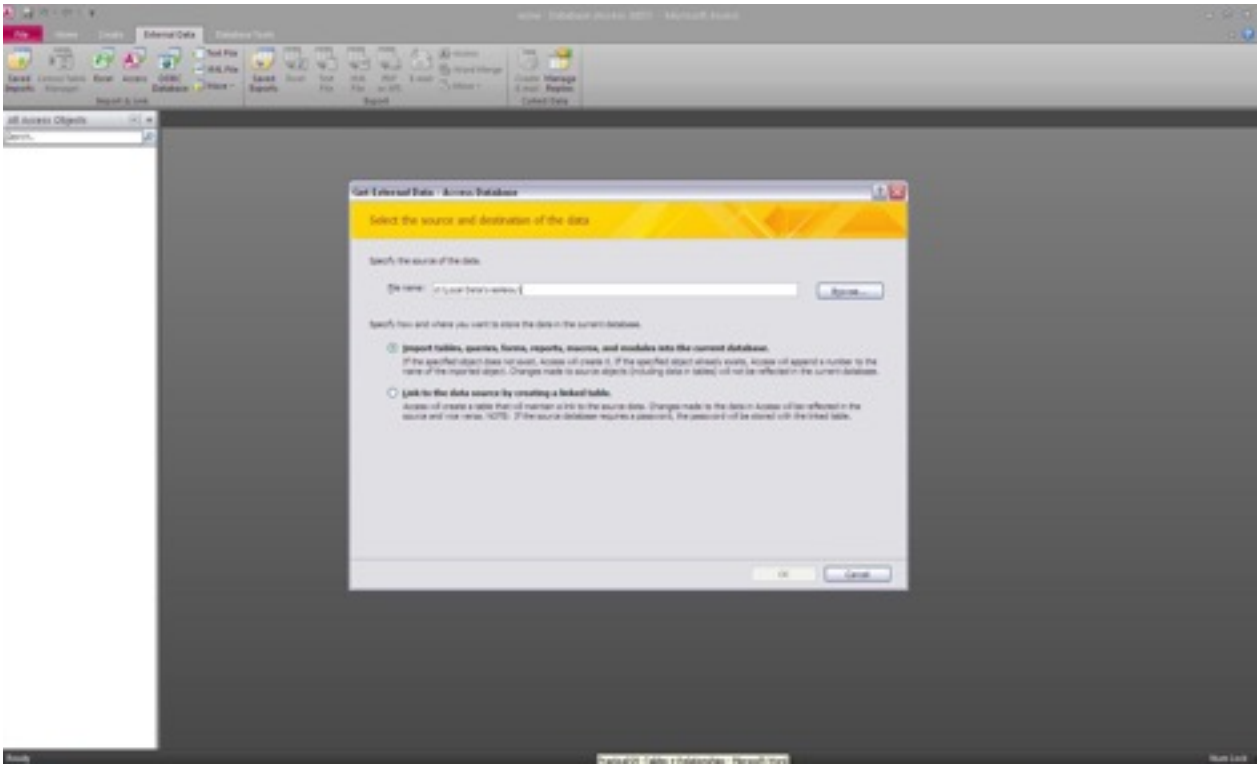
The following instructions apply to any kind of object you want to import, not just to tables. Ensure the open Database Window is the one from which you have just deleted the Customers table (see above).

➢ click on the **External Data** ribbon tab

➢ from the **Import** group click the **Access** icon

Note – It is normally best to IMPORT data from another Access Database, which is what we are now going to do, even if it is an earlier version. Use the LINK option if the data is from another source or if you want to maintain the original table for use by other software.

The Import dialog box will appear from which you can browse the data source to which you downloaded the Acme database.



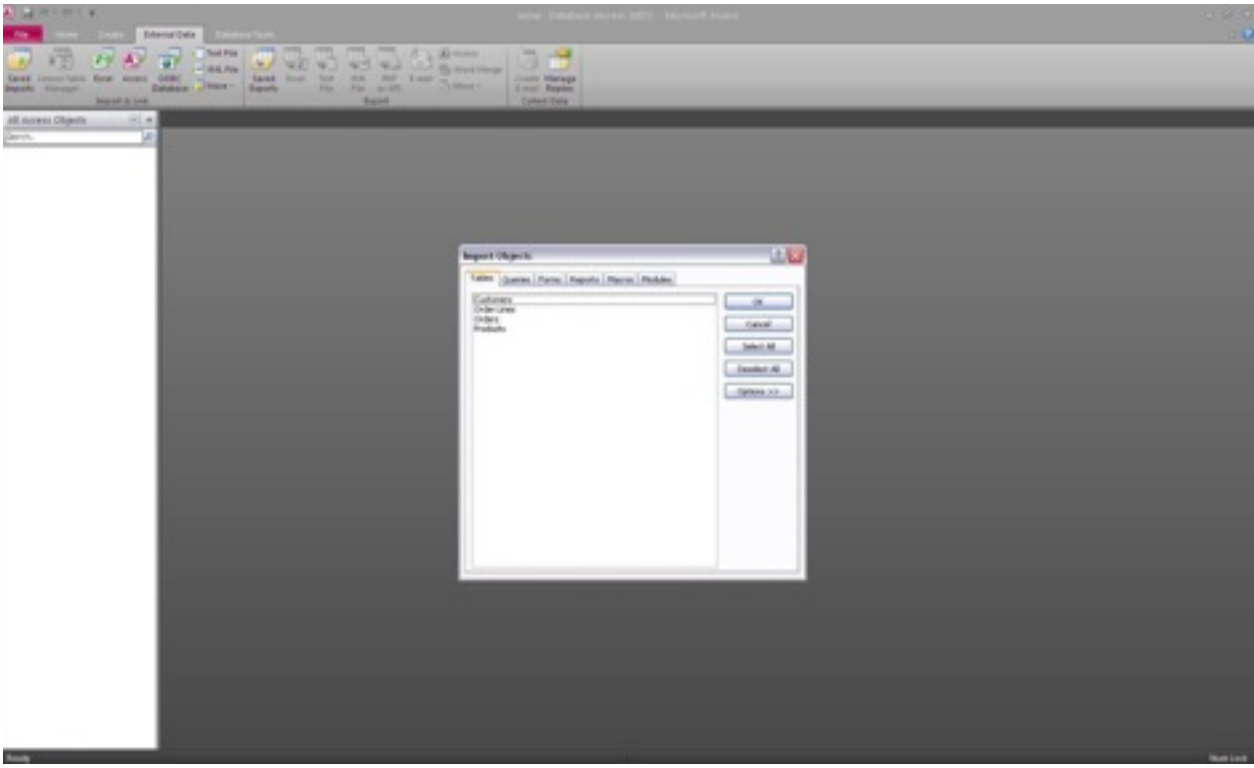
➢ ensure the **Import...** option button is selected

➢ click **OK**

A dialogue box with a list of tables is presented

In the Import Objects dialog box, make the following selections:

- ➢ select the **Tables** tab (if not already selected)
- ➢ then **Select All**
- ➢ click on **Options >>**
- ➢ deselect the **Relationships check box -** DO NOT Import Relationships
- ➢ Under **Import Tables** select **Definition and Data**



- ➢ click **OK**
- ➢ you should receive a message ‘All objects were imported successfully’
- ➢ click **Close**

You now have imported four tables: Customers, Orders, Order Lines and Products.

Data Sheet and Design Views of Tables

When you are in the datasheet view you can quickly change to the design view and vice-versa by using the toolbar buttons:

Datasheet View Design View

➢ open the Customers table and try swapping between the two views

➢ inspect the fields and field properties and see if you can relate them to the data in the records

Sorting data

By default the data in the Customers datasheet is sorted by the first column. You can re-organise it into alphabetical order by company name, by Surname, by town (AddressLine2), or by any other field, by using the sort buttons on the toolbar. To sort the data:

➢ click in the *Company* field and then click

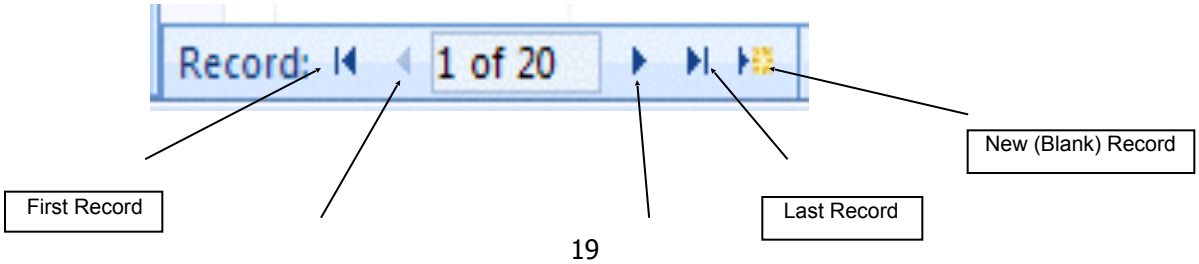
The Company names are now listed in alphabetical order.

➢ try sorting the data using other fields such as *Surname* and *Post Code* and try sorting in reverse order

➢ if you want to return to the original state, click  (**Clear All Sorts**)

Record Navigation

At the bottom of the datasheet are the record navigation buttons used for moving through the record list. Note that the starred button to the right of the last record button enables the "Enter a New Record" mode.



Previous record Next Record

The Orders, Order Lines and Products Tables

The complete the Acme database contains four tables. You do not need to create any of the tables yourself because you have imported them. We will now look briefly at the Orders, Order Lines and Products tables. The fields and data types are as follows:

Table	Field	Data Type
Orders	OrderID	Autonumber
	CustomerID	Text
	OrderDate	Date/Time
Order Lines	OrderID	Number
	ProductID	Text
	Quantity	Number
Products	ProductID	Text
	ProductDescription	Text
	UnitPrice	Currency
	UnitsInStock	Number
	Supplier	Text

The Primary Key field in the Orders table is *OrderID*. This is an Autonumber data type which automatically assigns a unique value to each record. (See HELP for more details.)

Fields that relate tables have to be the same data type (though they do not have to have the same name in each table).

The *OrderID* field (data type AutoNumber) in the Orders table is related to the *OrderID* field in the Order Lines table. But this related field cannot also be an Autonumber. Instead it is given a data type that is compatible with Autonumber, and that is Number, with the field size property set to Long Integer.

In the Order Lines table, neither *OrderID* nor *ProductID* on their own will uniquely identify a record, but together they will. Therefore there has to be a *Compound* Primary Key for the Order Lines table - *OrderID* and *ProductID*.

PRACTICE TASK: Learn how to set a compound key by deleting the key in the Order Lines table and resetting it. Highlight the two key fields using CTRL or SHIFT keys, and click the KEY button. Order Lines now has no key. Save the changes. To set the key,

highlight the two as before and click the KEY button. The first column in the table design view will be the first element of the compound key by default, so you have to reverse the order if you want to modify the key.

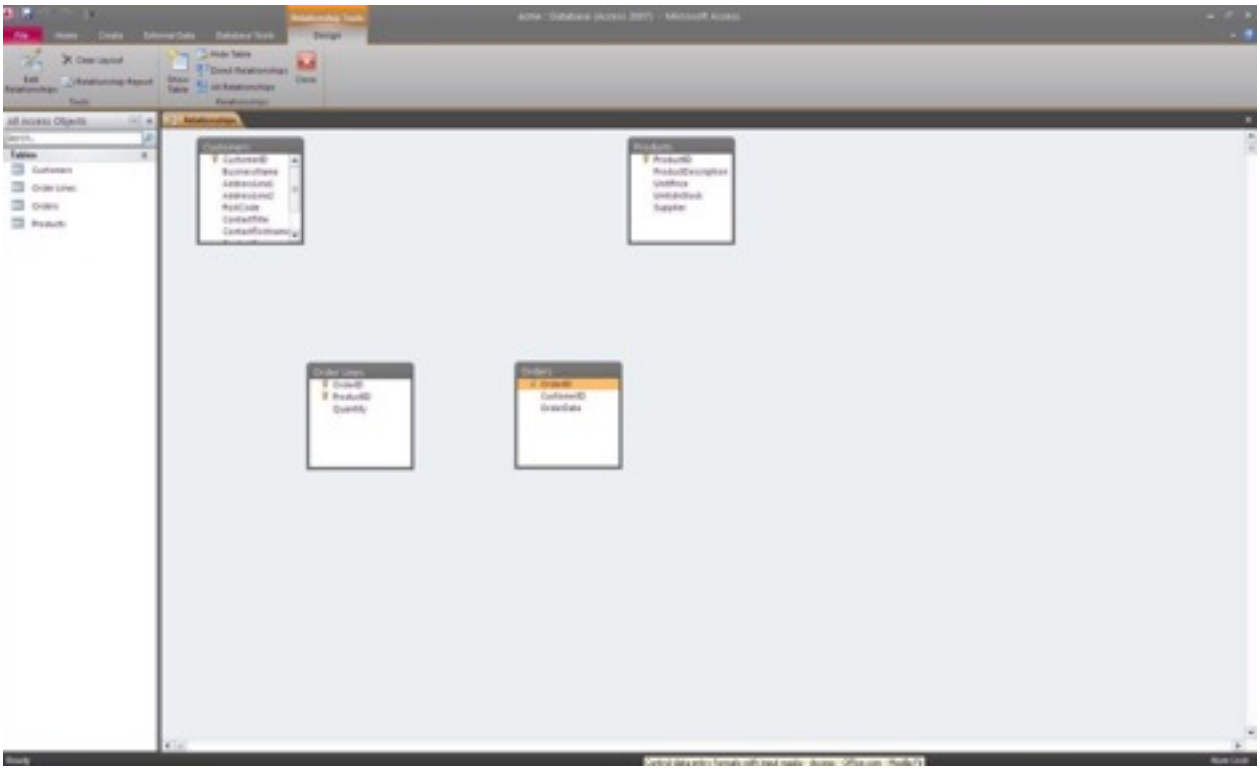
Relationships

The final task in creating the database is to link the tables together and Access provides a simple graphical method of doing this.

➢ on the **Database Tools** ribbon tab, click the **Relationships** icon on the **Show/ Hide** group

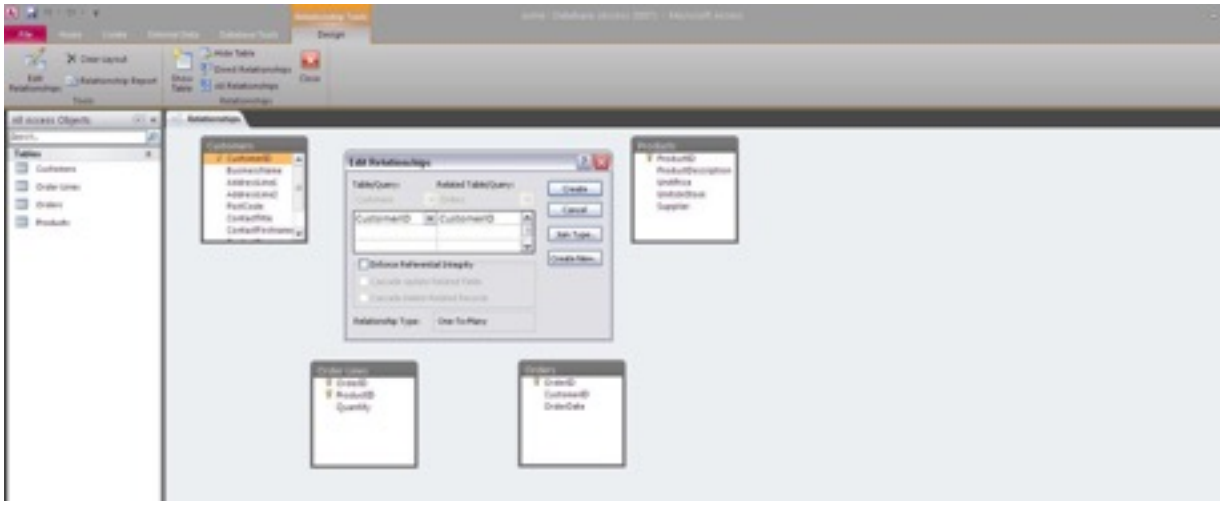
➢ in the **Show Table** dialogue box, highlight all four tables, then click **Add**, then **Close**

The Relationships window shows all four tables represented by small windows containing the field names. If they appear in different positions, move them around by dragging their title bars until they are arranged as shown below.



➢ To link the tables: In the Customers table, click *CustomerID* and drag it to the *CustomerID* field in the Orders table and release

The Relationships dialog box will appear



➢ click the box **Enforce Referential Integrity** - this will for example prevent an order being entered for a customer who does not already exist in the Customer table.

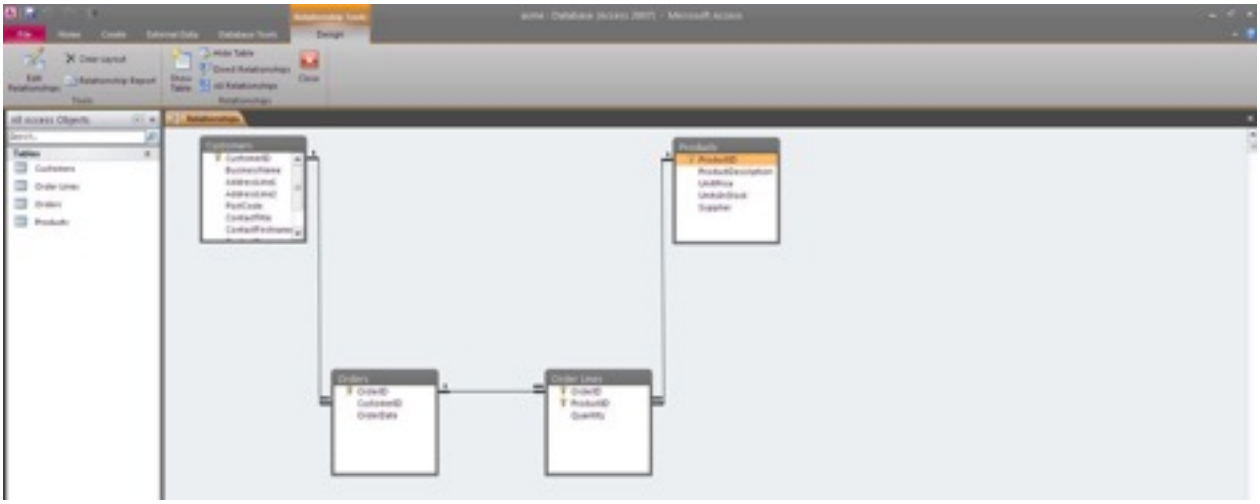
➢ click **Create**

*A line will appear linking the CustomerID fields in the Customers and Orders tables. A figure **1** appears at the **One** side of the relationship, and an infinity symbol appears for the **Many** side.*

➢ drag *OrderID* in Orders to *OrderID* in Order Lines and repeat the steps as for *CustomerID*

➢ drag *ProductID* in Products to *ProductID* in Order Lines and repeat the steps as for *CustomerID*

When all the links have been made, the Relationship window should look like the illustration below:



➢ when you are satisfied that the arrangement is correct, click **Close** and reply **Yes** to save the changes

Saving and Backing up your Database - *READ THIS CAREFULLY!*

This is most important. Even if you are using your own personal PC to develop your database, you cannot be sure the database file will not be corrupted or wiped by some accident. Using a PC not your own is even riskier. IT IS YOUR RESPONSIBILITY TO ENSURE YOU HAVE AN UP TO DATE COPY OF YOUR DATABASE!

When working on your database, the database file (the accdb file) will be saved in your chosen destination after you have closed your work at the end of a session. This could be on your U-drive. Copy the accdb file at least TWICE (i.e. TWO separate backup copies, for safety) onto two disks or flash drives. You can copy your work to your PC at home, if you have one with MS Access installed, and carry on working there but **there may be conflicts between different versions!**

The Access Tools

Tables, Forms, Queries, Reports and **Macros** are the tools Access uses to manage data. (There are also **Modules** but these will not be described here.) You use each tool to implement a particular part of your database.

Tables store information in rows and columns. Each row is a complete record of some thing (a customer, a client, a product etc.) Each column is a different item of data in the record (name, address, post code, product name, price etc.)

Forms are used to enter and display data.

Queries are used to select, re-arrange, update and organise data.

Reports are used to print information derived from tables or queries.

Each of these tools, and Database Create, has a **Wizard** associated with it. The Wizard can help you automate the process of creating an object by asking questions on a series of dialog boxes.

How does Access store a database?

Each Table, Form, Query and Report that you create is called an **Object.** All objects within a database are stored together in one database disk file with the file extension **.accdb**. This means that every time you use your database, you have only one file to save, which contains all the objects (unlike most other database systems which use separate files for each object).

Forms in MS Access 2010

Forms

A form gives you a more user-friendly way to enter and view data in your database. Although the datasheet is useful for viewing a lot of records at the same time, the form has several advantages:

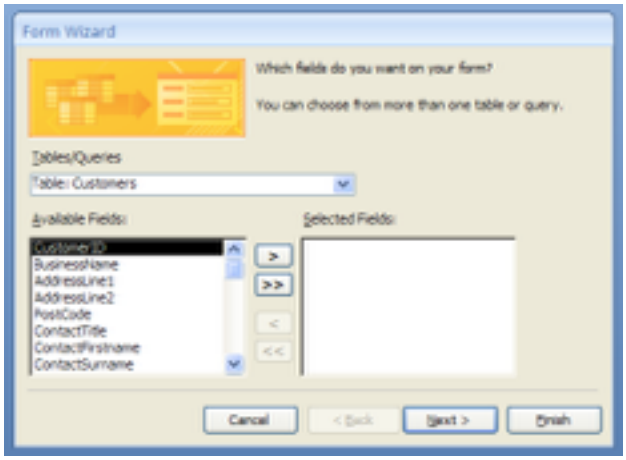
- ➢ in a form you can view one record at a time on screen and see all the fields (in a datasheet, some of the columns can disappear off screen)
- ➢ you can add explanatory text to the screen to assist the user with data entry (in the datasheet there are only the column captions)
- ➢ it is easier to enter records using a screen form which looks like a paper form than it is using a datasheet
- ➢ forms can be customised with all sorts of useful components including list boxes, radio buttons and pictures

Creating a Form using the Form Wizard

The first form you will create is a simple form for entering a new customer into the Customers table in the Acme database which you created in the previous section.

- ➢ open the Acme database
- ➢ highlight (click) the table for which you want to create a form – in this case *Customers*
- ➢ click **Create**, **More Forms** and then **Form Wizard**

The New Form dialog box will appear



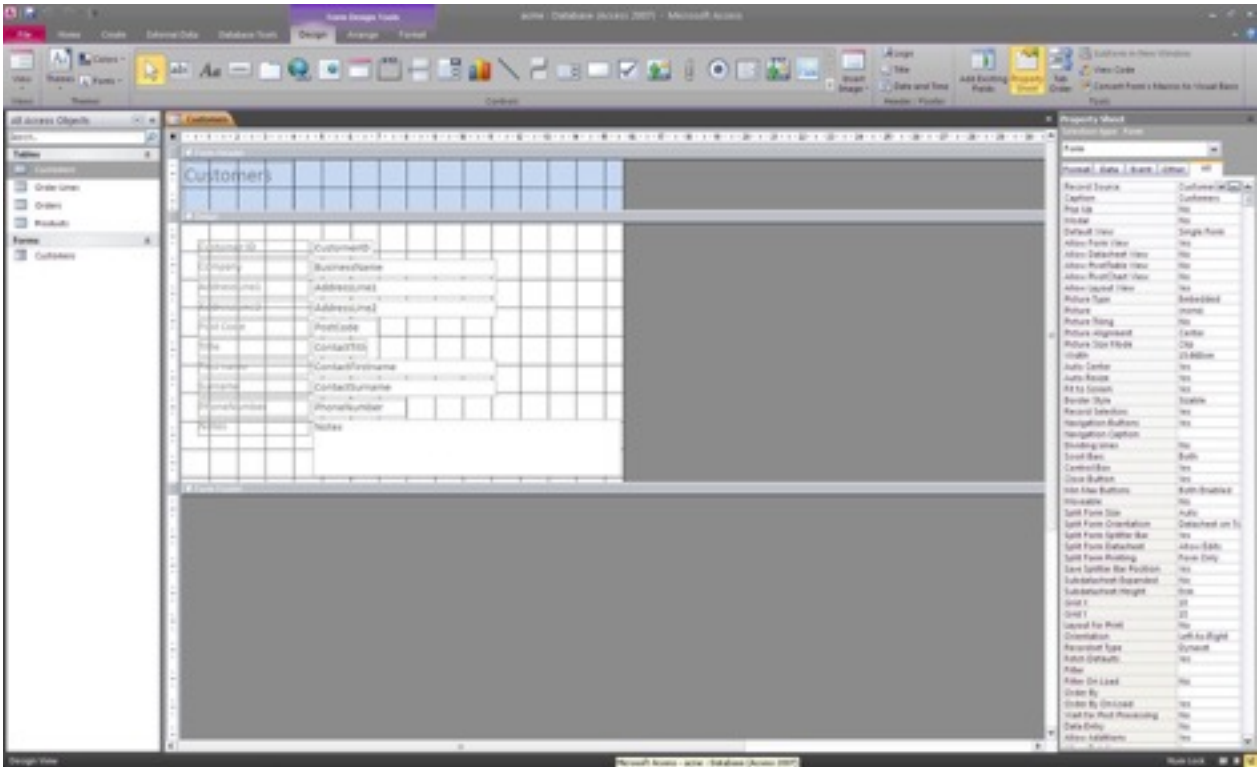
The Wizard displays two list boxes within the dialogue box. The one on the left shows all the fields in the Customers table. The blank box on the right will show all the fields you select to appear on the form.

Since the purpose of this form is to enter new records in the table you should include all the fields.

You select a field by highlighting it and clicking the button. You can select fields in whatever order you prefer. All the fields can be selected at once by clicking the button. A field can be removed from the form list by highlighting it and clicking

- ➢ select and then click **Next** to see the next dialog box
- ➢ select **Columnar** (if not already selected) and click **Next**
- ➢ now select a Style – note this is previewed for you – then click **Next**
- ➢ select 'ACME Customers' as the title for the form (the default given is the table name); ensure 'Modify the form's design' option button is selected then click **Finish**

After a few moments the Form appears in Design View:



Note that the number of columns or fields is small enough to display the form as single column.

The form as it stands is not very elegant – the fields are not arranged in the most efficient manner. Your next task is to make some changes to improve your form design. For experienced users, the main use for Wizards is to prototype a form quickly with the required columns. This form can then be edited to give the required result. You can choose a theme, or manually choose colours, font styles etc.

This is an example of the final Form:

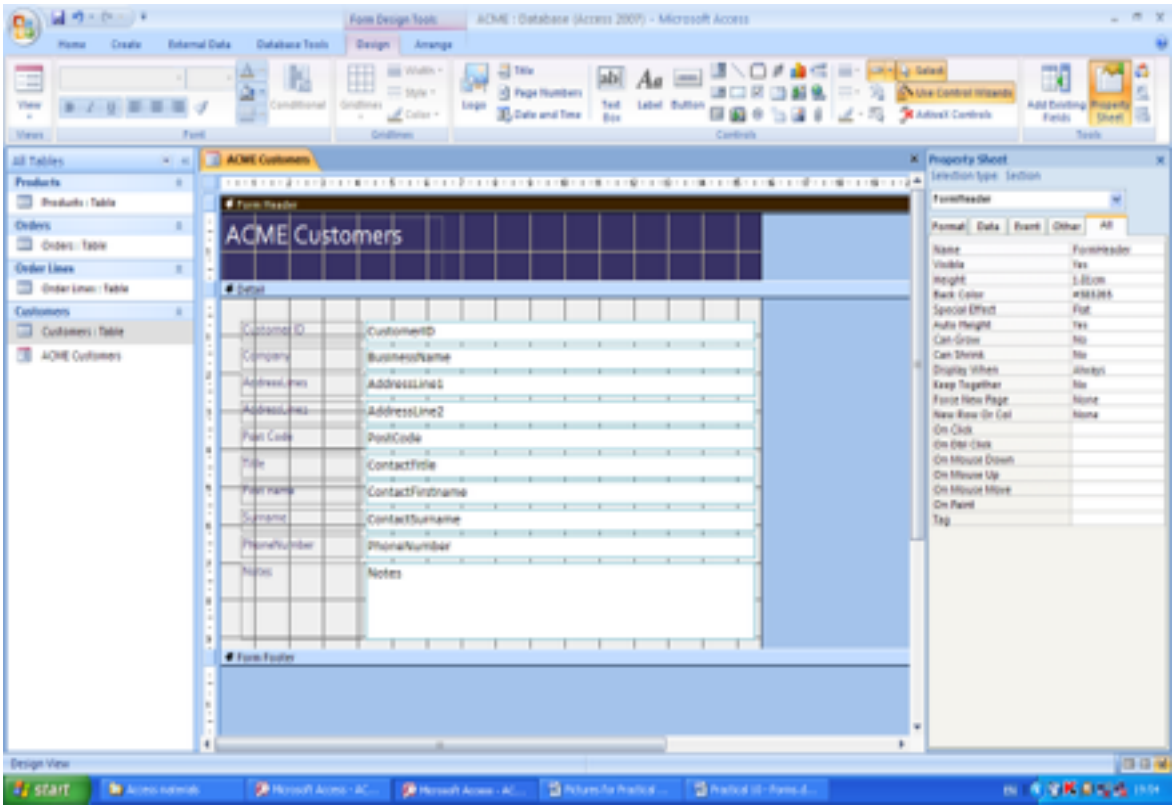
Use this illustration as a reference to help you make your changes.

Customising your Form

In the exercises you will make the following changes to your form:

- move the fields and captions
- change fonts and font sizes
- modify the Tab Order
- delete/change captions
- add a company logo to the form header
- use colour to differentiate between areas of the form
- change the Title field to a Combo (selection) box

(Remember: To make changes to the form design you must be in Design View – you should still be in Design View, but switch if necessary.)



Take a few minutes to familiarise yourself with the two ribbons that will be of most use in modifying the form. These are **Arrange** and (especially) **Design**. As you move the pointer over each of the tools in the various groups, think about what effect this would have on the form. (When you want to see how a change will look in the final version, switch to Form View.)

A first sight the Design screen looks very complicated, but you can make some changes to tidy it up. You will not need the Field List window at the moment so close it. If the grid is not visible, right click in the Detail area and click **Grid**.

The Form Design View

Your form, in design view, consists of three distinct areas:

This area contains information that remains the same from one record to the next. In this case it is the title “ACME Customers”.

This area contains all your Customer Fields and their captions. The values for these fields are different for each customer.

You may have to scroll down the Design window to see this one. The Footer does not contain anything for this Customer Form.

Controls

Everything in a Form Design is called a Control. The fields are controls, captions are controls, the label in the Form Header is a control. (There are many other types of control which can be created using the tools - you will be creating one later). All of these controls have Properties.

Properties

You have already been introduced to the concept of properties when looking at the data types of the fields in your tables. In the Form Design View *everything* has properties - the form itself, the Form Header, the Detail, the Form Footer and of course all the controls. If you want to see the Property Sheet for any of the controls, click the control and then click **Property Sheet**. The list of properties changes as you click different controls. Look at the properties as you click on different areas of the form and see the different properties which are available for different controls. The list of properties is extensive but you will be concerned with only a few of them.

Moving Fields and Captions

Because the form has been created via the Form Wizard in a linked structure (Columnar), you need to remove the linkage so that you can move the fields and captions independently of one another. To do this:

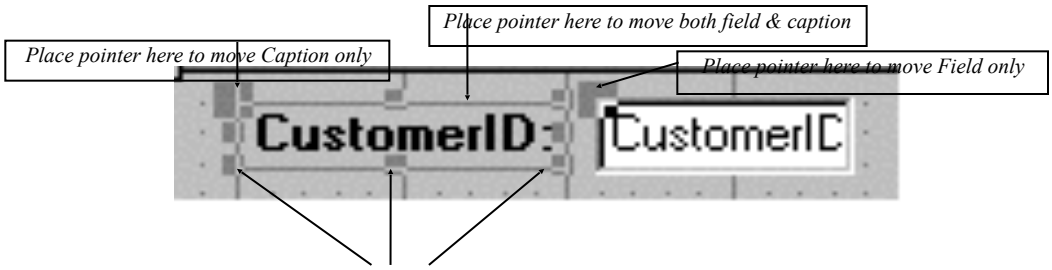
- ➢ click on the arrowed crosshair in the box at the top left corner of the Detail section (this highlights all the captions and fields)
- ➢ then click **Arrange**, then **Remove**

Controls (in this case, the fields and captions) can now be selected and re-arranged and re-sized individually.

The simplest bit of customising you can do to a form is to move the fields and captions around. In order to move an item you first have to select it:

➢ click on the caption of *CustomerID* to select it

When the field and the caption are selected, handles appear as in the diagram below. The smaller handles are used for re-sizing. The two large handles on the top left corners are used for moving the field and caption.



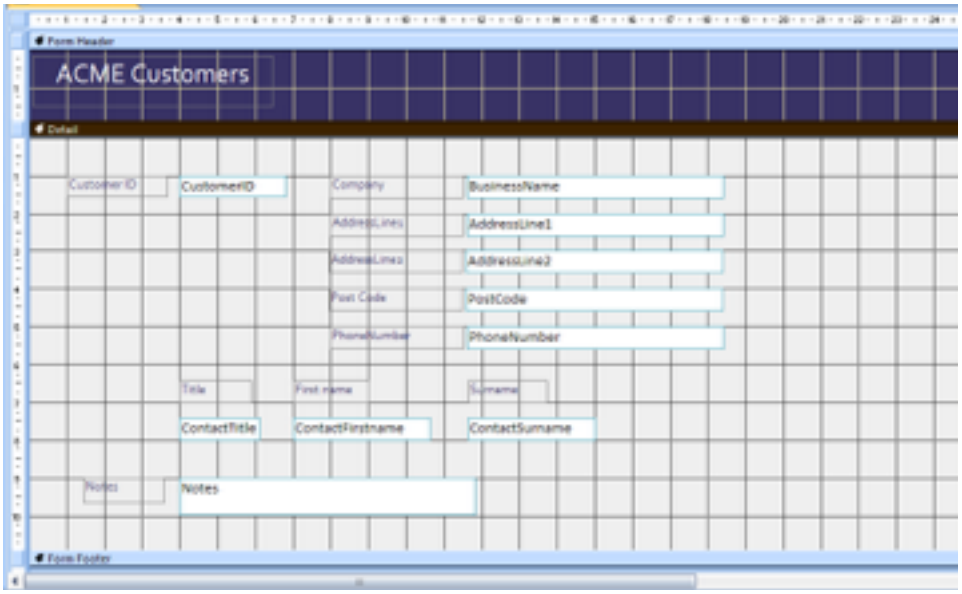
Place pointer on any of these handles to re-size

When the pointer is on a large handle it changes to an arrowed crosshair. This indicates that you can move the caption or field on its own.

When the pointer is on the border of a field or caption it again becomes an arrowed crosshair, but both caption and field will move together.

When the pointer is placed on a smaller handle it becomes a double-headed arrow, which you can use to re-size the object (make it larger or smaller).

- ➢ make the form window wider by placing the pointer on the right hand window border and dragging it to the right (close the Properties window if necessary)
- ➢ drag all the controls into the approximate positions shown below (you can move labels independently of the fields)



It will probably be helpful to begin by re-sizing the *CustomerID* field and then moving the company, address and post code fields to the right. These, too can be re-sized. Note that several fields can be re-sized together by dragging the pointer over them to select them.

Labels

To modify the Form Title to "ACME Customers Records", click in the (Label) box in the Form Header and add the required text. Then click on the edge of the box and move/re-size it as necessary. Open the Property Sheet and select a font (Font Name) and size (Font Size) of your choice.

When you are making design changes to your form it is a good idea to switch periodically to the Form View to check how these changes appear.

➢ select Form View to check how it looks

If you now sequence through the fields using the Tab Button, you will note that they do not Tab in the right order. You must now correct the Tab ordering.

➢ return to Design View

➢ click **Arrange**, then **Tab Order**

➢ click **Detail** in the Tab Order dialogue box to see the current Tab sequence

In the list of fields displayed, the PhoneNumber field should be between PostCode and ContactTitle. It must be selected and moved.

➢ select *PhoneNumber* by clicking in the grey selector box at the left of the text

➢ place the pointer on the selector box of the highlighted field and drag it up to the correct position

➢ click OK

➢ select Form View and check that you can now Tab through all the fields in the correct sequence

Remember to Save your changes regularly.

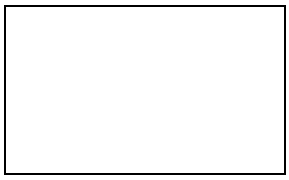
If you accidentally delete a field or a label, or want to back out a change then remember that you can Undo unwanted or accidental changes.

Making further changes to the form layout

- select Design View again and ensure that **Property Sheet** is open
- click in the label ‘Customer ID’ and add a colon; then (under Design) click to right justify the text – note that you can see the properties change on the Property Sheet (you can also effect changes through the Property Sheet)
- click in the label ‘Company’ and do the same
- click in the label ‘AddressLine1’andchange the caption to ‘Address:’ then right justify the control
- select the label ‘AddressLine2’ and press **Delete**
- click in the label ‘Post Code’, add a colon and right justify the control
- click in the label ‘Phone Number’ and change it to ‘Tel:’, then right justify
- add a colon to each of ‘Title’, ‘First name’ and ‘Surname’
- click in the label ‘Note’ and change it to ‘Comments:’, then right justify and look at the result in Form View – it should appear something like this

Adding a Logo

The next task is to place the company logo into the Form Header. The logo is an Object that has been created in a drawing application. It must be placed within an Object Frame or Picture Frame in the Form Header. This can be done in three ways, described below.



METHOD 1 - You can copy and paste the Logo image directly from this document.

This may require extra space to be created in the header. Place the pointer on the lower boundary of the form header and drag it down until there is sufficient space for the logo. Re-size the logo as required and change the Back and Border colours if you wish to – it is best to use the Property Sheet to do this.

METHOD 2 – To insert a bitmap (bmp) image on the form, select the **Unbound Object Frame** tool in **Controls** group under **Design**.

- ➢ place the pointer (which now has the Object Frame tool attached) in the form header, hold down the mouse button and drag a frame to contain the logo - when you release it the Insert Object dialog box will appear
- ➢ select **Bitmap Image** from the List box and the radio button **Create from File** as we will get the logo as a ready-made image
- ➢ click the **Browse** button that appears

The Browse dialog box will appear. This is where you will locate the company logo graphic file.

- ➢ select the required drive and file name then click **OK** to return to the Insert Object dialog box

- ➢ click **OK** to see the logo imported into the form

- ➢ with the logo frame selected, re-size it to display the logo

METHOD 3 – for most types of image file.

- ➢ select the **Image** tool (see above)

- ➢ place the pointer (which now has the Image tool attached) in the form header, hold down the mouse button and drag a frame to contain the logo - when you release it the Insert Picture box will appear

- ➢ locate the company logo graphic (gif) file, select it and click **OK**

- ➢ resize the frame as necessary to fit the logo

The complete logo is now visible within the frame, but the background (and border) of the logo may stand out from the form header. This can be corrected by modifying the ***Back Style, Back Colour and Border Colour*** *properties; but note that this may not work for all image file types.*

- ➢ lastly, change the **Name** property to *Logo*

- ➢ switch to Form View to check on the results - if you are not happy with any of the changes, go back to Design View and change them

Changing the background colours

The next design change you will make to the form is to give different colours to the backgrounds of the Form Header and Detail. Select a darker colour for the Header and a pastel colour for the Detail so that the text can be read easily.

To change the background colour of the Form Header:

- ➢ select Design View

- ➢ click in a blank area within the Form Header to select it

- ➢ in the properties window scroll the list and click the **Back Colour** property, then click the button to reveal the colour palette

- ➢ select the colour of your choice and click **OK**

- ➢ click in the header area to see the change

To change the background colour of the Form Detail:

➢ click in a blank area within the Form Detail

➢ select the **Back Colour** property, open the colour palette, select your colour, click **OK** and then click on the form Detail to see the changes

If the labels have not taken on the same colour as the Form Detail background then do the following:

The **Back Style** property of all the labels must be changed to *Transparent* so that the labels will take on the same colour as the Detail background. To make this task easier, you can select *all* the labels and only change the **Back Style** property once. To make a multiple selection:

➢ click on any label to select it

➢ hold down the Shift Key and click on all the other labels in turn

➢ in the properties window (whose title bar now states “Multiple Selection”) select the **Back Style** property and change it to *Transparent*

➢ click back on the form to see the change

If you want, you can change the appearance of fields and/or labels by altering the **Special Effect** property from *Normal* to one of the other properties (for example, *Sunken.)* Select one object and try it to see the effect. If you find an effect you like, change all labels or all fields (or all objects) to the same.

Using the Wizard to create a Combo Box

A Combo Box is a control that provides a list of values to select from (and you can also type in a value not in the list). This is useful for data entry because it reduces the amount of typing required, and reduces the possibility of typing errors.

In your Customer form you will replace the *ContactTitle* field by a Combo Box with a value list of “Mr”, “Mrs”, “Ms” and “Miss”.

➢ ensure the Customer form is in Design View

➢ select the field *Contact Title* and press **Delete** - the field and label will both disappear

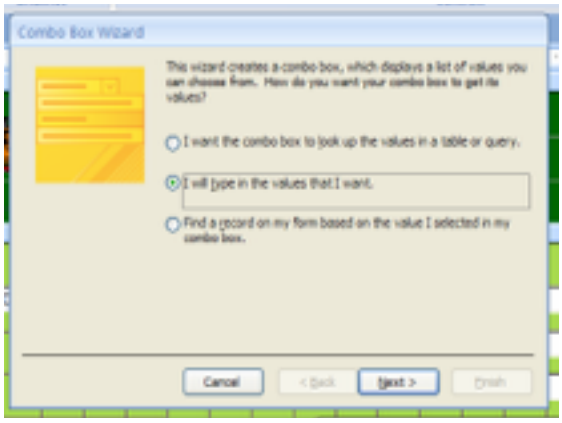
➢ click **Use Control Wizards** in the **Controls** group under **Design**

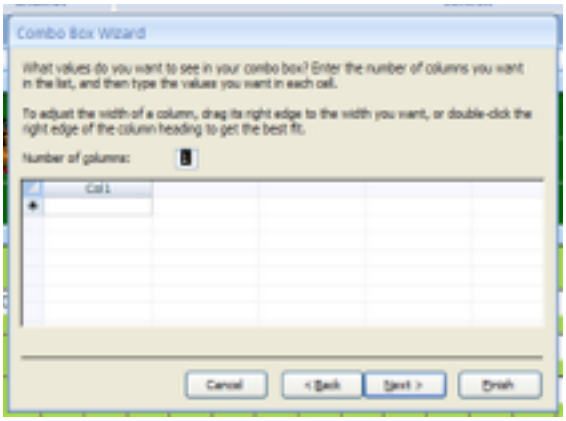
➢ click the Combo Box tool

➢ click the **Add Existing Fields** icon (in **Tools**)

➢ select the field *ContactTitle*, hold down the mouse button, drag the field on to the form and release to see the Control Wizard dialog box (don't worry about positioning it accurately - this can be done later)

➢ click the radio button "I will type in the values...." (as below) and click **Next**

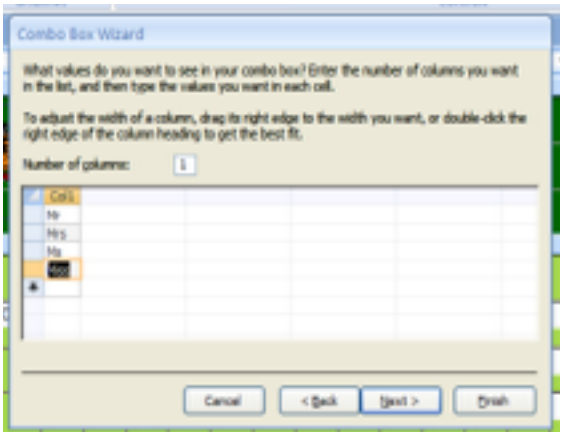




➢ accept the default of **1 i**n the **Number of Columns** box then click in the space under **Col1**

➢ type *Mr* and tab on to the next record; type *Mrs, Ms, Miss* into successive records

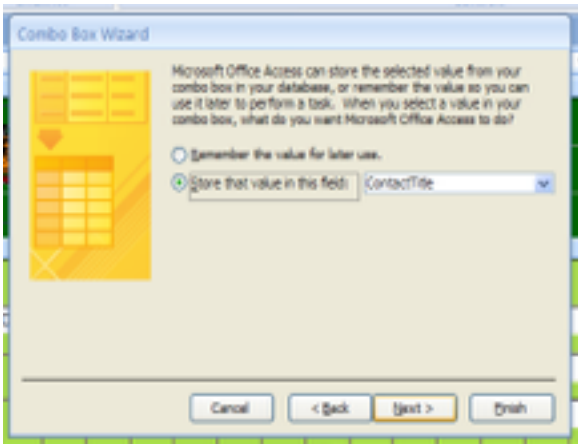
➢ make the column narrower by dragging the right border in



➢ click **Next** to go to the next dialog box

➢ the radio button “Store that value...” should be selected and the field *ContactTitle* should be shown for storing the value (if not, then select it)

➢ th*en click **Next*** to see the last dialog box



➢ enter the word *Title* for the label and click **Finish**

➢ re-position the Combo Box and label as required

➢ open the **Property Sheet**

➢ change the **Back Style** property of the label to *Transparent* (if necessary)

➢ change the **Name** property of the label to *Title*

➢ change the **Name** property of the Combo Box to *Contact Title*

Whenever you create any new control it automatically goes to the bottom of the Tab order. Therefore you must reset the Tab order.

➢ position cursor in main area of form and click right button to see the Tab Order option

➢ select *ContactTitle*, move it to the correct position then click **OK**

If you had not changed the Name property of the Combo Box then the Tab Order dialog box would have shown a default field number, which Access automatically allocates, instead of the name *ContactTitle*, in the list of fields.

It now only remains for a label "Contact:" to be created (refer to the target form below).

- ➢ select the Label tool and then place the pointer with the tool attached on the form
- ➢ place the cross-hairs just to the left of the Combo Box control and click on the form
- ➢ type Contact: - the label will expand as you type
- ➢ select the label *Contact* and change the **Back Style** property to *Transparent* (if necessary)
- ➢ adjust the position of the label
- ➢ change to Form View and compare your form with the example below

- ➢ if you are satisfied, save your changes (or return to Design View to make some further changes)
- ➢ do not forget to save changes regularly to protect the work you have done and also backup your database after each practical

Your form is almost complete. One further design change you can make is to get rid of unnecessary features from the form. You do not need scroll bars because all the fields are visible on screen, and you do not need the Maximum and Minimum buttons because you do not need to change the size of the form. To remove them:

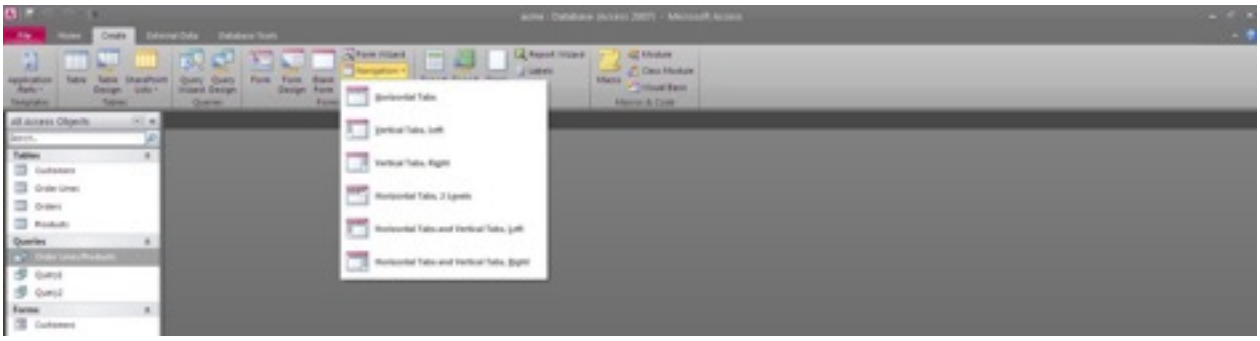
- ➢ in **Design View**, select the Form by clicking the box at the top left
- ➢ open the **Property Sheet** (ensuring it is the one for the Form)
- ➢ change the **Scroll Bars** property to *Neither*
- ➢ change the **Min Max Buttons** property to *None*
- ➢ save these changes
- ➢ re-open the Form to see the latest changes

You now have a well laid out and easy to use form for entering new data or viewing individual records.

Use your own imagination to design a similar form to enter Product Details into the ACME database with, for example, the Supplier name in a Combo Box.

Navigation Forms

Access 2010 introduces an entirely new form intended specifically as a navigation tool for users. Navigation forms include a number of tabs that provide instant access to any number of other forms in a form/subform arrangement. The Navigation ribbon button offers a number of button placement options.



Queries in MS Access 2010

What is a Query?

A query is a question you ask about the data in your database such as “Which of our customers bought POSCH tools in February?” The data that answers the question can be from one table or several and it is the query that brings the information together. A query allows you to see the data that you choose, in the order that you want it. Queries are very important if you are to gain the most benefit from using a database.

When you run a query, Access gathers the data you ask for in a **dynaset** (you can see this in the Property Sheet.) A dynaset is an updateable type of **record set** that is any set of records defined by a table or query. Therefore when you run a query the information is always up to date. Also certain types of query can be used to update the underlying tables. They also allow you to perform calculations on your data, to create sources of data for forms, reports, graphs and other queries, to make global changes to tables and create other tables.

This session will introduce two types of query. They are ***Select queries and Action queries***. With Access you can also create more complicated queries (**Cross-tab, Union, Pass-through and Data-definition queries)**. These will be covered later.

The type of query most commonly used is a select query. A select query allows you to select records, create new calculated fields and summarise your data.

An action query can be used to make changes to many records in one action. They are used to make new tables, to update records, to append records and to delete records.

Creating some simple examples will give you a better understanding of queries.

Creating a query

There are four main stages involved in creating a query:

Setting up the query - opening a new query window and telling the database which fields from which tables contain the data you require

Choosing the criteria - telling the database which of the data you want to see and how you want to see it

Running the query - checking that you have the required information and can retrieve the information

Saving the query - once you are happy you can then name and save the query for future use

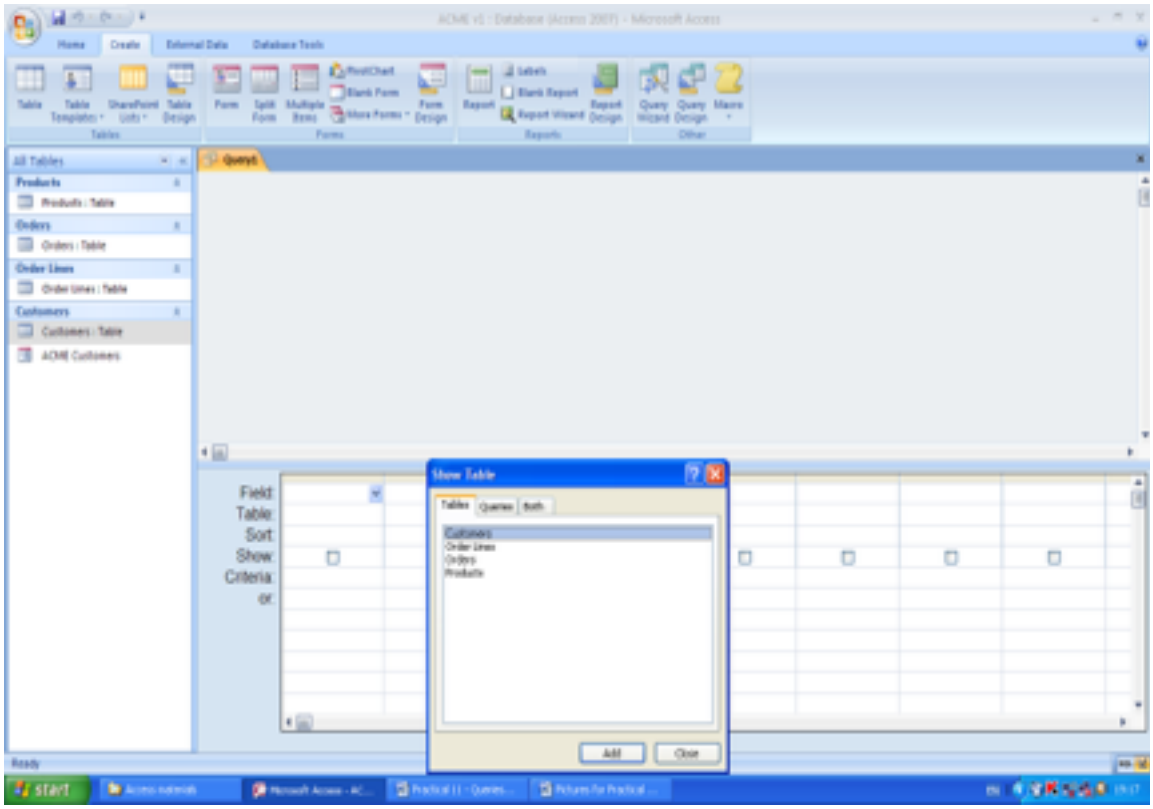
Creating a simple select query

For our first example you are going to create a select query to ask for a mailing list of all the customers who are Joiners & Carpenters.

Setting up the query

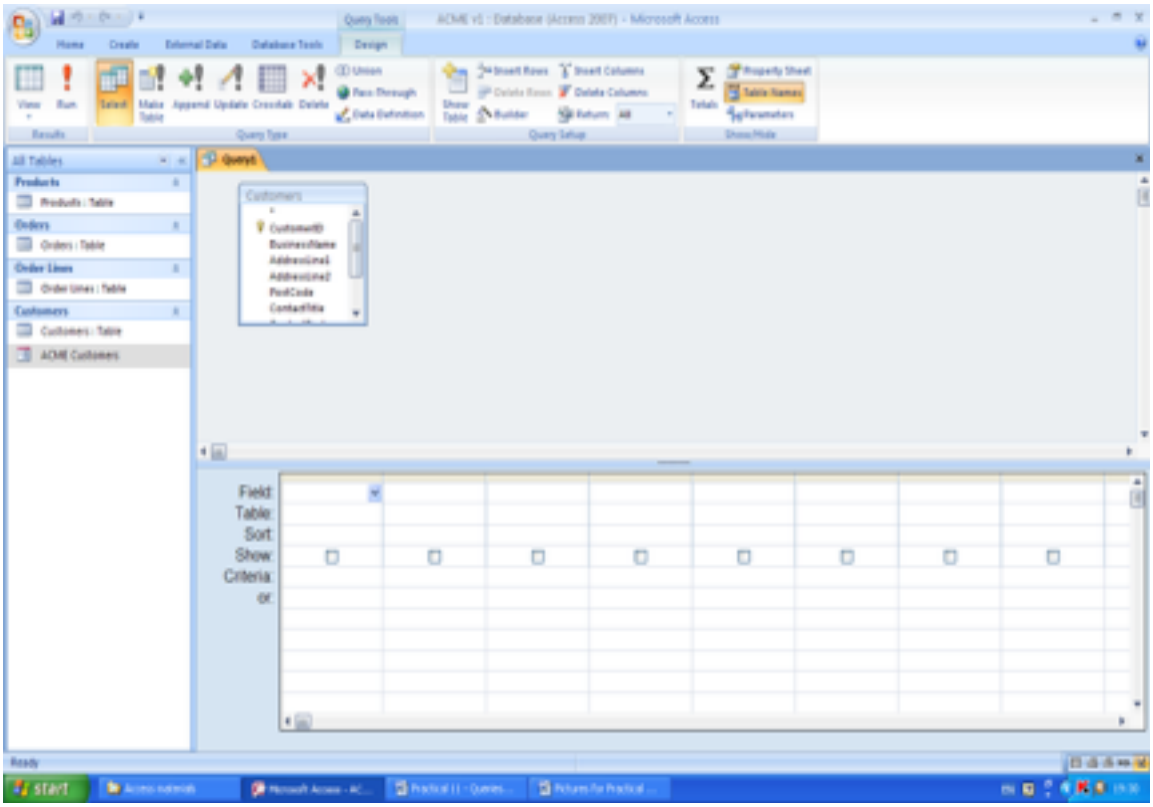
- open the ACME database
- click **Create**, then **Query Design**

*You are presented with an 'empty' query and a **Show Table** list box with the Customers table highlighted – this is the table you want to use as the basis of the query.*



- click **Add**, then **Close**

If you decide that you should have added another table you can reopen the **Show Table** dialog box by clicking the Show Table icon in the **Query Setup** group.



*You should have the **Design** ribbon for **Query Tools** showing.*

➢ ensure **Select** is highlighted – this is the type of query you want to create

The query window is a graphical query-by-example (QBE) tool. That means that you can select, drag and manipulate the objects in the window with a mouse. You now need to add the required fields to the query.

➢ re-size the *Customers* table to show all the fields

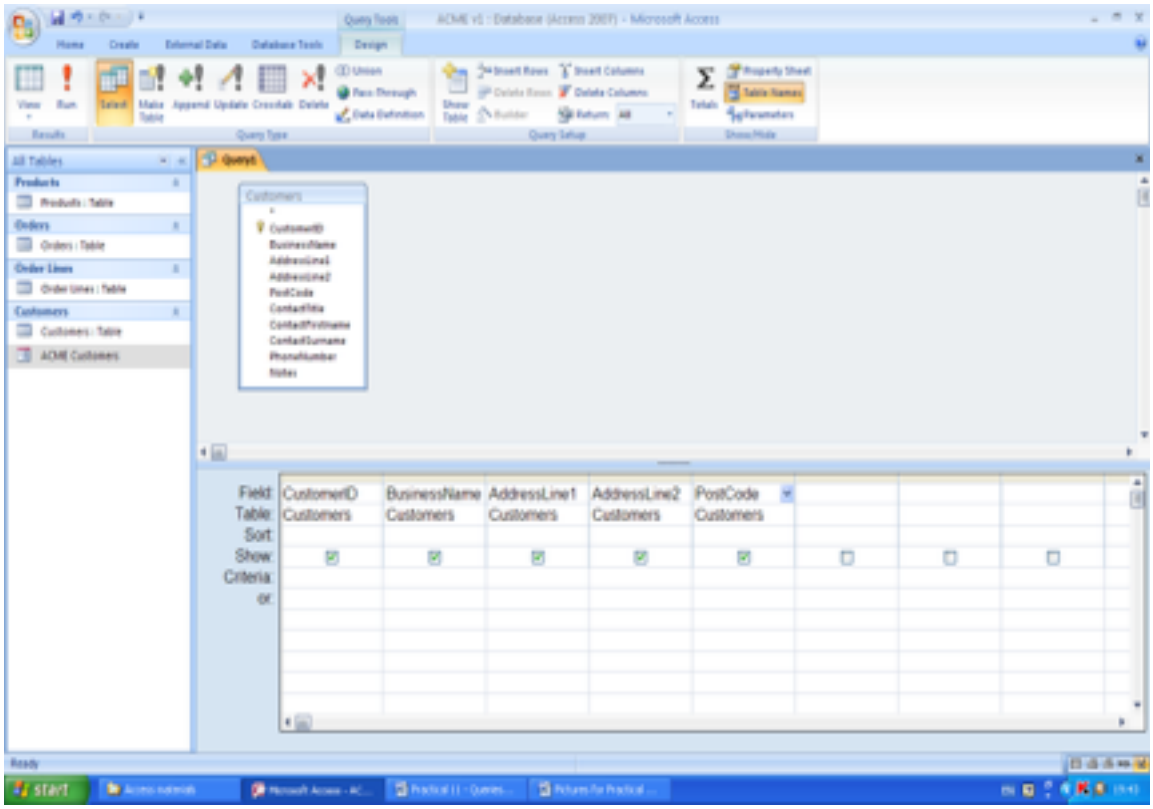
There are four ways of adding a field to a query: (1) you can double-click on a field in the field list. (2) you can select, drag and drop a field from the list. (3) you can click on the cell in the QBE grid and select a field from the drop down list. (4) you can type the name of the chosen field in the cell.

➢ add the *CustomerID* field to the first cell in the Field row of the QBE grid

➢ in the same way add the fields *BusinessName, AddressLine1, AddressLine2* and *PostCode* to the Field row of the QBE grid

➢if you run out of columns you can use the horizontal scroll bar to bring more cells into view

The query window should now look like this.



If you ran your query at the moment it would give you the names and addresses of all your customers. You could have got this information by simply printing out all the records in the Customers table. If you only want to see certain records from your table you must specify *criteria* (or ask a question).

Choosing the criteria

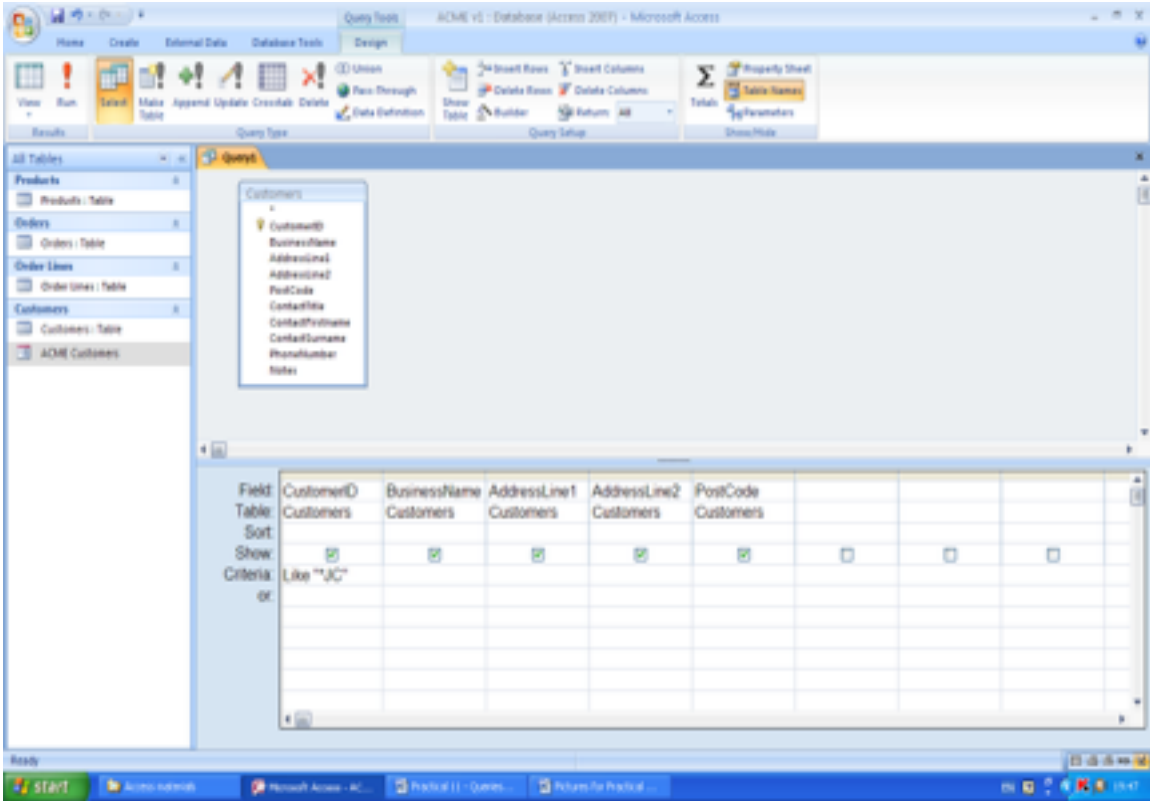
You only want the mailing list for those customers who are Joiners and Carpenters. The **Customers** table includes a field *CustomerID*. The letters at the end of each code denote the main trade or activity of that customer. So you only want the mailing list for those customers whose *CustomerID* ends in JC.

So that Access will find all the records where the *CustomerID* end in JC you have to use a *wildcard* as a substitute for the rest of the code. This is done by typing an asterisk *. For example if you wanted all names beginning with M then typing M* in the appropriate criteria cell will allow Access to find them all.

In this case you want all codes ending in JC so using *JC as the criteria will find them all.

- click in the Criteria cell for the *CustomerID* field
- type **Like "*JC"**

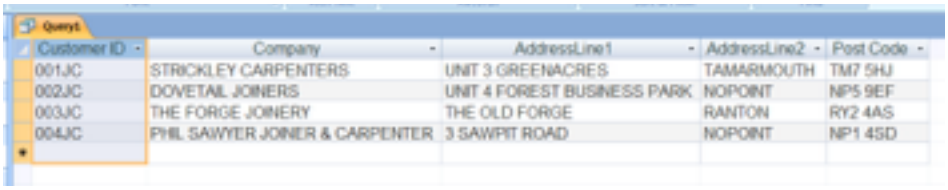
The query window should now look like this.



To see if you have the desired information you need to run the query. As you have seen with tables and forms all Access objects can be viewed either in Design View or Datasheet View. Looking at the toolbar you can see that you are currently in Design View for a Select Query.

- ➢ to run the query...
- ➢ click on the **Datasheet View** icon (or the **Run** icon)

This dynaset of records is the result of your query (note that the columns have been double-clicked to expand them to show all the data)



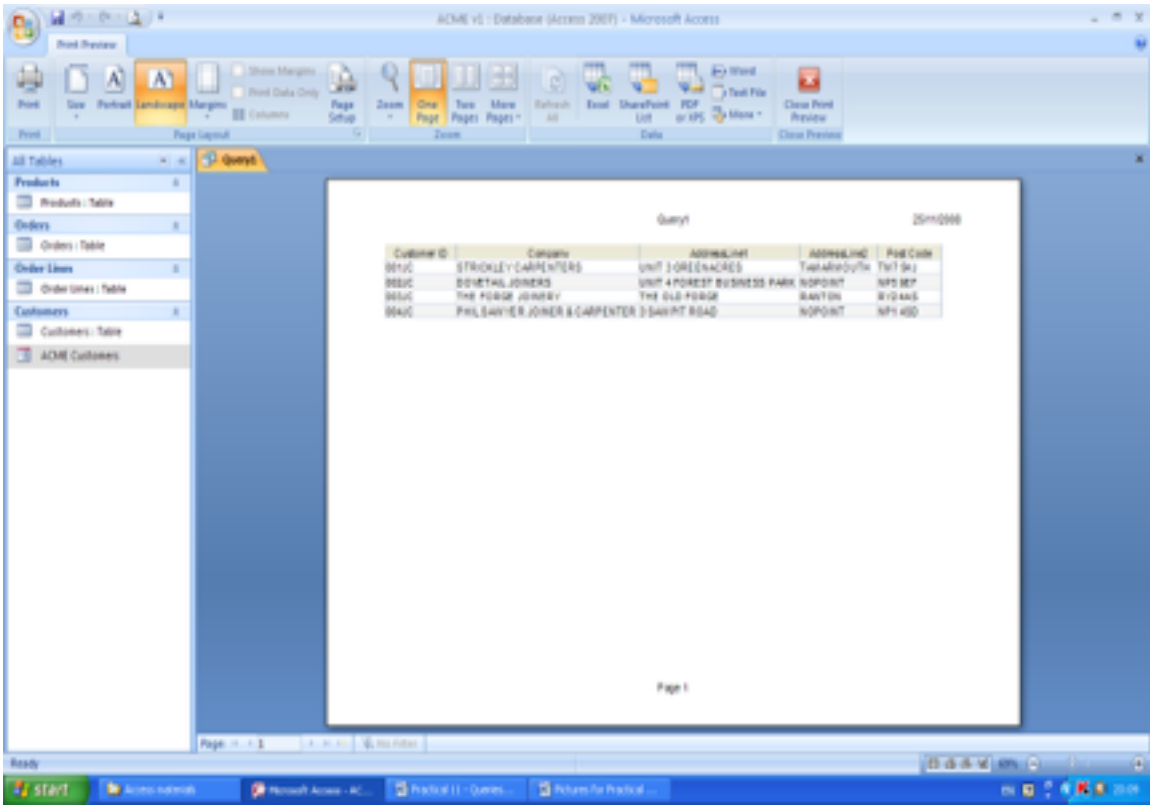
Customer ID	Company	AddressLine1	AddressLine2	Post Code
001JC	STRICKLEY CARPENTERS	UNIT 3 GREENACRES	TAMARMOUTH	TM7 5HJ
002JC	DOVETAIL JOINERS	UNIT 4 FOREST BUSINESS PARK	NOPOINT	NP5 9EF
003JC	THE FORGE JOINERY	THE OLD FORGE	RANTON	RY2 4AS
004JC	PHIL SAWYER JOINER & CARPENTER	3 SAWPIT ROAD	NOPOINT	NP1 4SD

It is always useful to be able to have a hard copy of your information. You probably do not have a printer available at the moment, but you can use the Print Preview facility to see how your finished item will look.

- ➢ you can preview print by using the **Print Preview** icon if it is available on the Quick Access toolbar...
- ➢ if this is not available, then click the **Office Button** (in the top left corner), then highlight **Print** and choose **Print Preview** from the opened menu list

You may find that not all of the table is printed on one page because the page is not wide enough. The easiest way to change that is to change the orientation to Landscape rather than Portrait.

- ➢ use the Print Preview ribbon to adjust Page Layout and Zoom as necessary to view the output – see the example below



Not all of Dovetail Joiners address may be shown, but the columns cannot be widened in the Print Preview view.

➢ click **Close Print Preview**

You can widen the AddressLine1 column to show all the address of Dovetail Joiners by placing the pointer on the table heading (field names) and dragging the column until it is wide enough (or double-clicking the right column edge). If you now check in the print preview window you will see that all the address is now shown. Now you are ready to name and save your query.

Saving your query

➢ right click the query tab and click **Save**

➢ type **Joiners and Carpenters Mail List** in the **Save As** dialogue box

➢ click **OK**

Your query is now listed on the Navigation Pane.

As well as being able to create new queries, you are now able to *Open* (run) or *Design* (make alterations in Design View) your existing query. You can do this by right clicking the query in the Navigation Pane and clicking Open or Design View.

This query can now be run whenever you want and it will always give you up-to-date information.

Experimenting with criteria

The query you have just created used a single criterion. Often a query will use a combination of criteria to define the information required. Now you are going to look at the QBE (query-by-example) grid in more detail.

➢ Open the query again in **Design View**

The Field row is where you choose which fields hold the data you require. If you click on the right-hand end of the *CustomerID* cell a drop down list of all the available fields appears.

The Sort row allows you to sort the dynaset alphabetically or numerically on any of the fields. If you click on the right hand end of a cell in the **Sort** row a list appears allowing you to choose Ascending, Descending or (not sorted)

➢ select **Ascending** in the *BusinessName* field

➢ click on the **Datasheet View** icon

The dynaset has changed order.

Joiners and Carpenters Mail List

Customer ID	Company	AddressLine1	AddressLine2	Post Code
002JC	DOVETAIL JOINERS	UNIT 4 FOREST BUSINESS PARK	NOPOINT	NP5 9EF
004JC	PHIL SAWYER JOINER & CARPENTER	3 SAWPIT ROAD	NOPOINT	NP1 4SD
001JC	STRICKLEY CARPENTERS	UNIT 3 GREENACRES	TAMARMOUTH	TM7 5HJ
003JC	THE FORGE JOINERY	THE OLD FORGE	RANTON	RY2 4AS

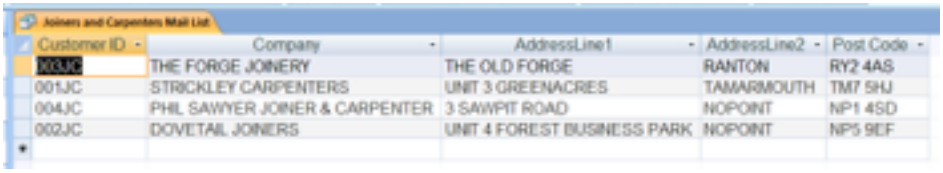
The records are now arranged with Business Names in ascending alphabetical order.

➢ return to **Design View**

➢ select **Descending** in the *BusinessName* field

➢ click on the **Datasheet View** icon

The order of the dynaset has been reversed.



Customer ID	Company	AddressLine1	AddressLine2	Post Code
003JC	THE FORGE JOINERY	THE OLD FORGE	RANTON	RY2 4AS
001JC	STRICKLEY CARPENTERS	UNIT 3 GREENACRES	TAMARMOUTH	TM7 5HJ
004JC	PHIL SAWYER JOINER & CARPENTER	3 SAWPIT ROAD	NOPOINT	NP1 4SD
002JC	DOVETAIL JOINERS	UNIT 4 FOREST BUSINESS PARK	NOPOINT	NP5 9EF

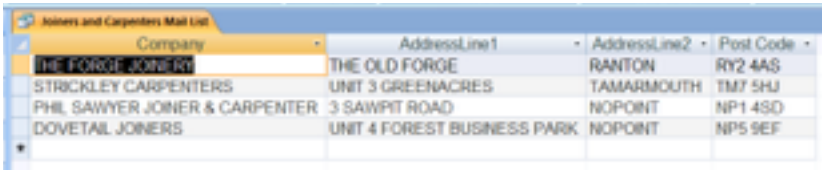
Experiment with sorting the dynaset alphabetically or numerically on the different fields.

The Show row allows you to control the display of fields included in the query.

For instance, in the Joiners and Carpenters Mail List you have identified the Joiners and Carpenters by their *CustomerID*, but you probably do not need to include this field in the Mailing list.

- ➢ In **Design View**, toggle off the show square of *CustomerID*
- ➢ click on the **Datasheet View** icon

The CustomerID field is no longer displayed.



Company	AddressLine1	AddressLine2	Post Code
THE FORGE JOINERY	THE OLD FORGE	RANTON	RY2 4AS
STRICKLEY CARPENTERS	UNIT 3 GREENACRES	TAMARMOUTH	TM7 5HJ
PHIL SAWYER JOINER & CARPENTER	3 SAWPIT ROAD	NOPOINT	NP1 4SD
DOVETAIL JOINERS	UNIT 4 FOREST BUSINESS PARK	NOPOINT	NP5 9EF

The Criteria row allows you to specify a range of criteria.

Common types of criteria are:

The value in a field is equal to the value you specify. In **Design View**, enter the value in the field's Criteria cell. For example, enter **Tamarmouth** in the criteria cell for the field *AddressLine2* to ask for all the customers from Tamarmouth and see the result in **Datasheet View**.

The value in a field contains a particular group of letters or numbers that you specify. A wildcard is used to take the place of that part of the field that can vary from record to record. You have already used a wildcard in the query you have set up (where *JC found all the codes ending in JC). Another example would be if you wanted all the customers whose name began with B. Then you would enter **B*** in the criteria cell for the field *BusinessName.*

The value in a field is one in a list of values that you specify. The **In( )** function is used. For example, use **In(Tamarmouth, Ranton)** in the criteria cell for the field *AddressLine2* to ask for all the customers from Tamarmouth and Ranton and see the result in **Datasheet View**.

Where the field holds a **numerical value** there is another commonly used type of criterion.

> **The value in a field lies within a range of values**. You might decide to reorder any products whose stock level falls below 5. The criterion in the criteria cell of the Units*InStock* field of the **Products** table would be **<5**. You can use **<, >** or **Between *?* and *?*** for values between two numbers.

Parameter queries

The query you have already created gives a mailing list for all customers who are Joiners and Carpenters. You could now set up a similar query for each of the other trades. The alternative is to create a parameter query that will ask which trade the mailing list is for.

Creating a parameter query

Setting up the query

This query will be identical to the last one except for the criteria. So why create a new one? You will be familiar with the concept of being able to open a word processor document, make edits and then save the new document with a different name. You then have two similar documents saved separately. This is the same with Access objects. Providing that you do not save or exit Access no changes to the original will be made.

You may have made a number of changes to your original query by experimenting with various aspects of the QBE grid. Before moving on to develop a parameter query, it is suggested that you save a version of the **Joiners and Carpenters Mail List** with the following characteristics:

- *CustomerID* field Show box toggled off
- *CustomerID* field Criteria cell contains **Like "*JC"**
- *BusinessName* field sorted in Ascending sequence

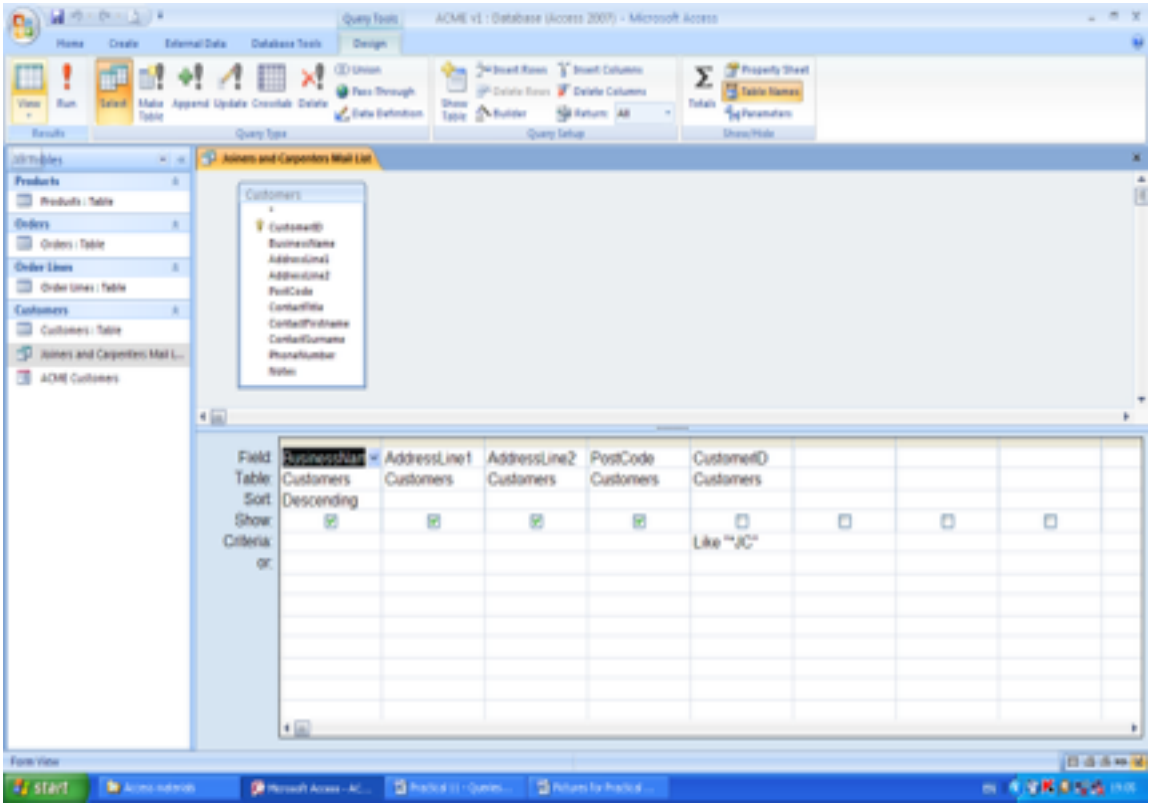
When you are satisfied with this, save the query, replying **Yes** to changes.

You are now ready to continue.

➢ if required, open the **Joiners and Carpenters Mail List** in **Design View**

Choosing the criteria

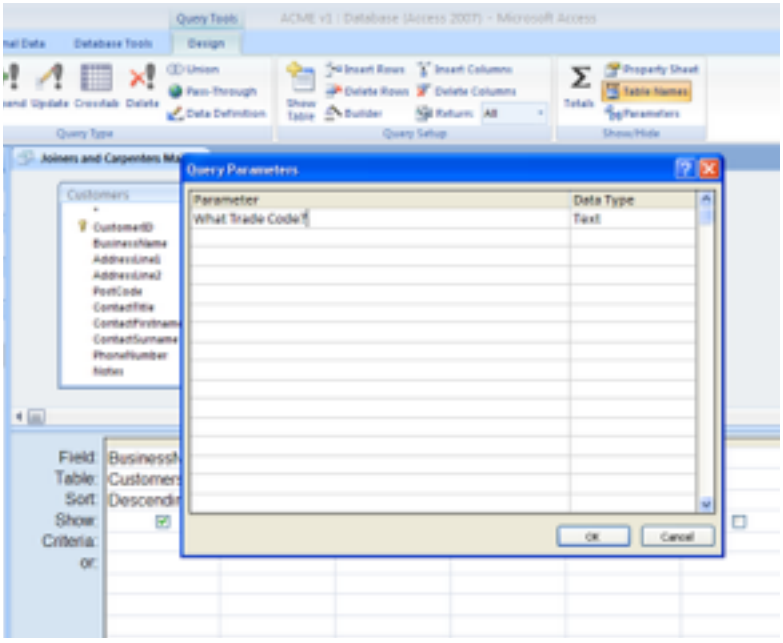
You may find that the *CustomerID* field has moved and is now the fifth field in the grid. Access has moved it because you chose that it should not be shown.



You want to be prompted for the customer's Trade Code when the query is run. If you enter text in square brackets, for example [What Trade Code?] then you will be prompted with that question on running the query. To try this out:

- type **[What Trade Code?]** in the criteria cell for *CustomerID*
- click the **Parameters** button in the **Show/Hide** group

A Query Parameters dialog box now appears.



➢ now type in the Parameter column *exactly* what you typed in the criteria cell without the squared brackets, i.e. **What Trade Code?**

Remember that you can move the dialog box (click on the blue title bar and drag) so you can see what you have typed in the criteria cell.

If the spellings are not identical Access prompts you twice and the query will not run correctly.

The data type for the parameter must match the data type of the field the parameter relates to.

➢ type **Text** in the Data Type column

(You probably found that Access filled this in automatically when you clicked in the Data Type cell.)

Running the query

➢ click **OK** to close the dialog box

➢ click on the **Datasheet** icon to run the query

An Enter Parameter Value dialog box appears.

➢ type **001PB**

➢ click on the **OK** button

The query runs and a single record is found – Goodenough Plumbing.

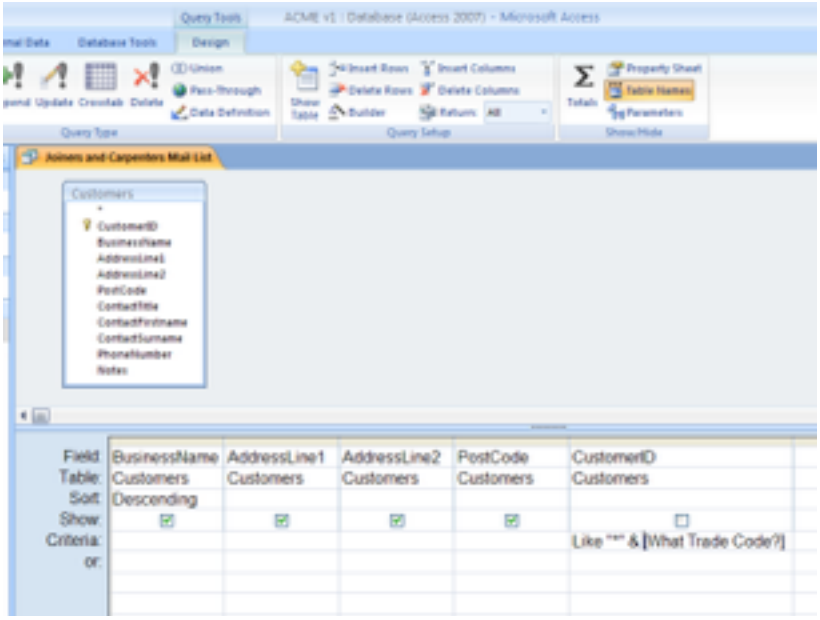
Modifying the query

Now you are able to ask for a different record each time the query is run. What you really want is to be able to ask for a different group of records each time.

To be able to ask for all the records with the same Trade Code you need to add a wildcard to the parameter.

➢ return to **Design View**

➢ type **Like "*" &** in front of **[What Trade Code?]**



The **&** joins (concatenates) the two criteria (i.e. the wildcard and the value entered in the Enter Parameter dialog box) and treats them as one criteria.

Running the query (try 2)

➢ click on the **Datasheet** icon to run the query

➢ type **LB** in the **Enter Parameter Value** dialogue box

➢ click **OK**

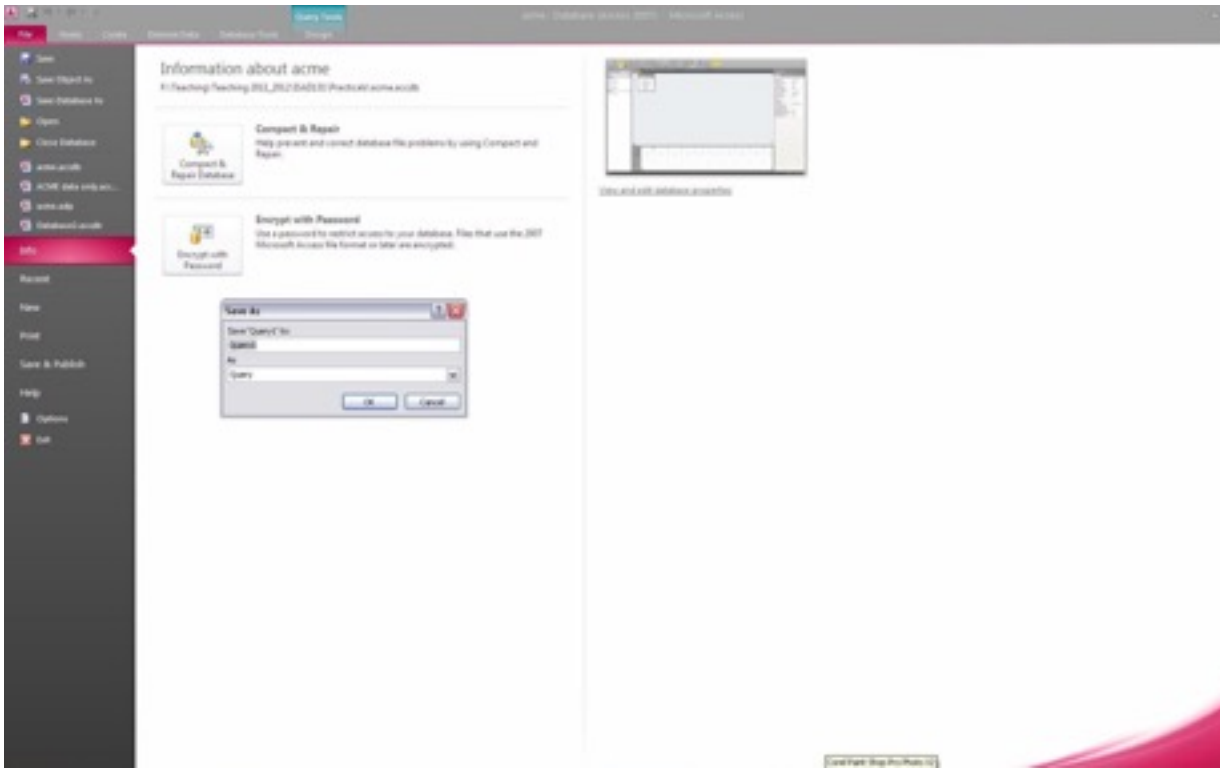
The query has found all the Large Builders

Company	AddressLine1	AddressLine2	Post Code
BUILDIT BUILDERS LTD	THE OLD MILL	RANTON	RY8 4TY
ELFLEIGH DEVELOPMENTS	2 GOBLIN CORNER	NOPOINT	NP7 2FY
ENSIGN IMPROVEMENTS	23 HIGH STREET	RANTON	RY1 3GB
HARRISON & SONS	714 CLOUDY VIEW	RANTON	RY5 7RW

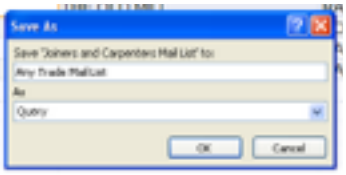
Saving the query

You are now ready to save the query. Remember that this is a copy of Joiners and Carpenters Mail List that has been altered, so you want to save it as a different query.

- click the **File** button in the top left corner
- click the option **Save Object As**



- type **Any Trade Mail List** as the new query name (check that the type is shown as Query)
- click **OK**



- close the new query

There are now two queries shown in the Navigation Pane. Note that the **Any Trade Mail List** will allow you to select individual customers as well as groups.

Subforms in Access 2010

What is a Subform?

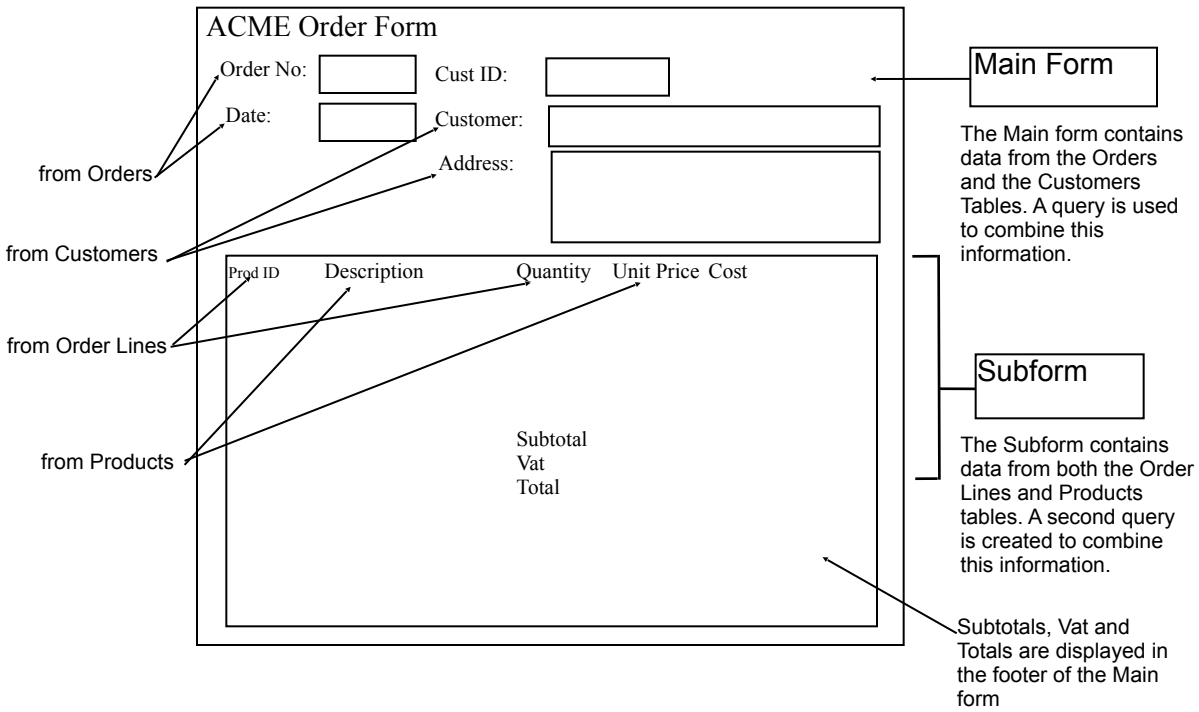
Previously you have created a simple data entry form “ACME Customer Records” but Access allows you to do a lot more with forms than entering or viewing a single record in a single table. A Subform is a form within a form. When a database has related tables then Access can use a Main form with Subform to display a record from one table and all the records from a related table at the same time. It is also possible to enter data into several tables via a single Main/Subform. This will be important for providing business functions for the assignment database.

The Main form displays a record from the ***Master*** (*One* end of the relationship) in a one-to-many relationship. At the same time the subform displays all the related records from the ***Detail*** table (*Many* end of the relationship, called the *Child* table in Access). It is also possible to have more than one subform on the same main form if there is more than one Child (Detail) table linked to the Master table.

An Order form is a good example of a Main/Subform arrangement. The Order is the Master record and the Detail records are the related Order Lines.

The Acme Order Form

The diagram of the Acme order form below illustrates the different parts that make up the Main/Subform.



Overview of the steps in creating a Main/Subform

Before embarking on the detailed instructions for creating the Acme Order Form, here is a brief overview of the steps involved:

- create a select query to combine data from the Orders table (OrderID, Order Date, CustomerID) and the Customer table (BusinessName, address information)
- create a select query to combine data from the Order Lines table (OrderID, ProductID, Quantity) and the Products table (ProductDescription, UnitPrice)
- use the Forms Wizard to create a columnar layout form based on the Customers/ Orders query, which will become the Main form
- use the Forms Wizard to create a datasheet layout form based on the Order Lines/ Products query, which will become the Subform
- make any required design changes to the Main form in Design View ensuring sufficient space to insert the Subform (it will be helpful to save a version of this at this point)
- make any required design changes to the Subform in Design View checking Datasheet View from time to time to ensure the correct column headings and layout
- open the Main form in Design View and 'drag and drop' the Subform into it
- adjust the size of the Subform until it fits neatly within the Main form boundary
- save the completed Main/Subform

In addition to steps described above you will set some extra properties applicable to subforms, and also create special calculated controls which will display the Cost of each order line, Subtotals, Vat and the Total cost of the order.

Constructing the queries

The first task is to construct the query that will allow you to combine data from the Customers and Orders tables. This query will be used to construct the Main form.

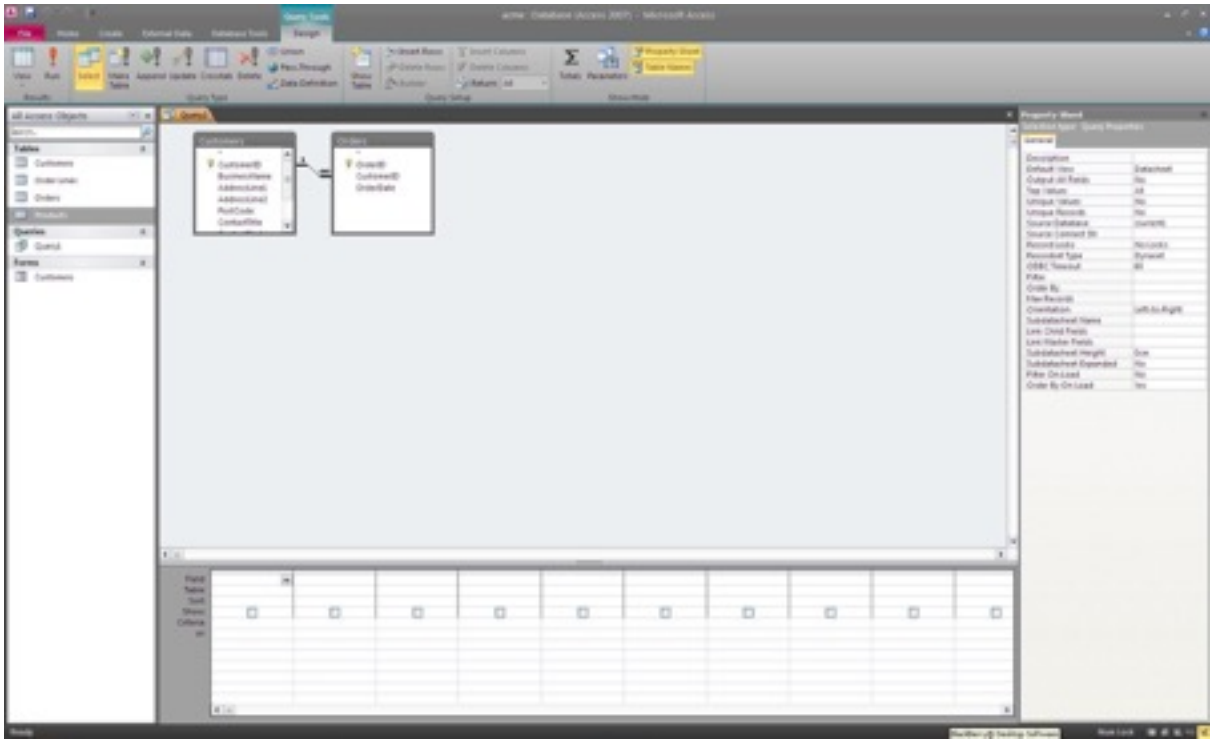
➢ open the Acme database

➢ click **Create** then **Query Design**

➢ highlight the *Customers* and *Orders* tables; click **Add** then **Close**

➢ ensure that the **Query Type** is **Select**

In the Query window you should now have the two linked tables as shown below. (Access has automatically opened a Select query, which is the type you want.) The next step is to select the fields you require. (If you are unsure how to set up fields in a query, review the relevant part of this document.)

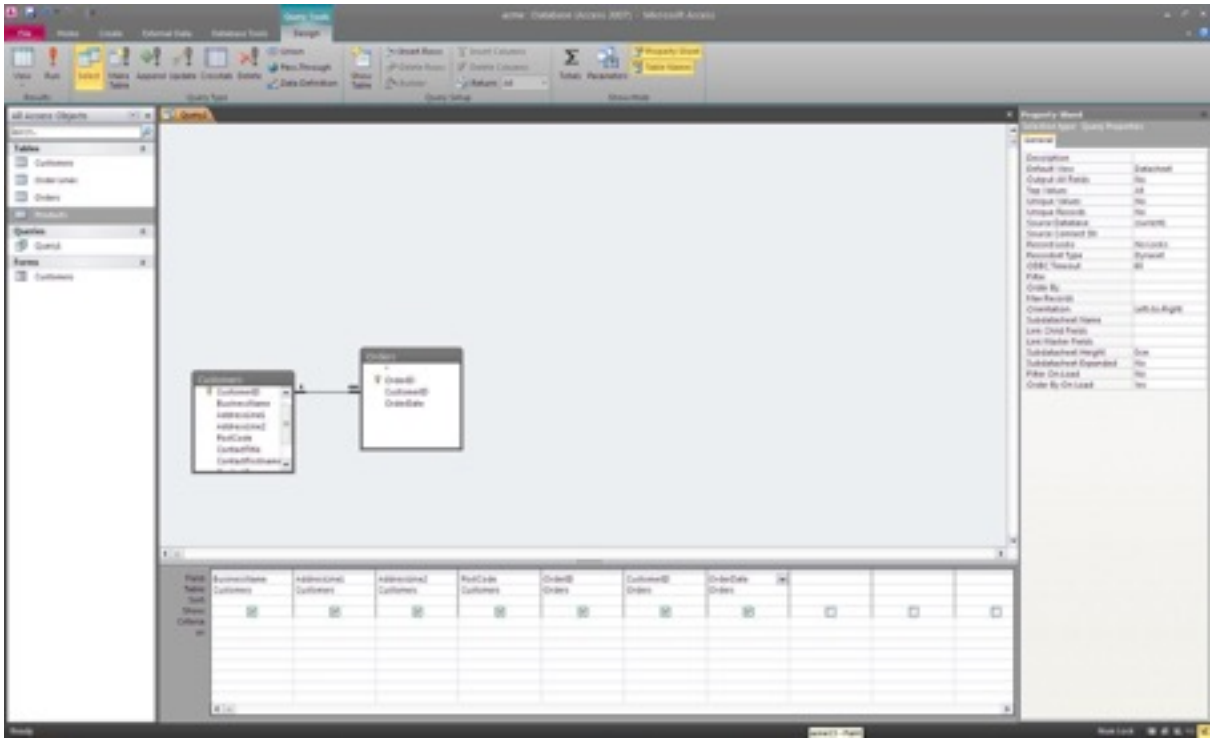


- from the Customers table add the following fields to the QBE grid: *BusinessName, AddressLine1, AddressLine2, PostCode*

- from the Orders table add the following fields to the QBE grid: *OrderID, CustomerID, OrderDate*

CustomerID **is present in both tables because it is the link field. In the QBE grid it is important to select** ***CustomerID*** **from the** ***Child*** **table (Orders) and <u>not</u> the** ***Master*** **table (Customers). This is because when a user enters a Customer ID on the order form, it must refer to a** ***CustomerID*** **that already exists in the Customer table.**

The QBE grid should now look like the picture below.



➢ **Close** and **Save** this query giving it the name **Customers/Orders**

The next step is to create a query to combine data from the Order Lines table and the Products table. This will be used to construct the subform.

➢ click **Create** then **Query Design**

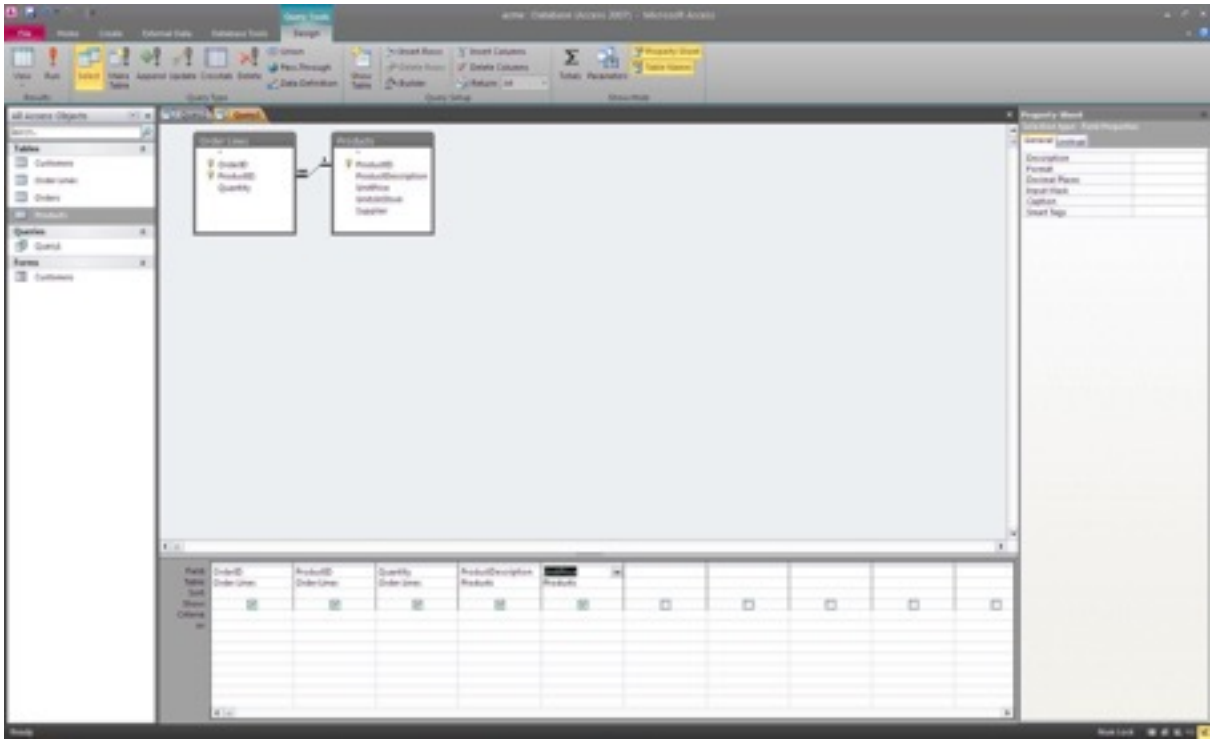
➢ highlight the *Order Lines* and *Products* tables; click **Add** then **Close**

➢ ensure that the **Query Type** is **Select**

➢ from the Order Lines table add the following fields to the QBE grid: *OrderID, ProductID, Quantity*

➢ from the Products table add the following fields to the QBE grid: *ProductDescription, UnitPrice*

The QBE grid should look like this.

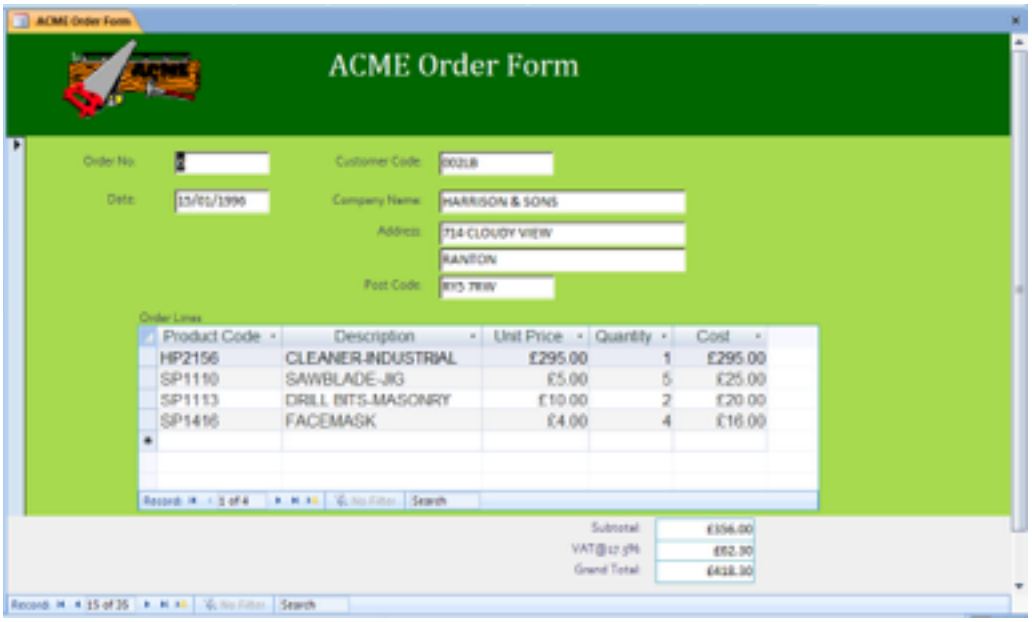


***ProductID* is in both tables. Remember to add it from the *Child* table (Order Lines) and not the *Master* table (Products)**

➢ **Close** and **Save** this query giving it the name **Order Lines/Products**

Constructing the Main/Subform

The form you are going to produce will look something like this.



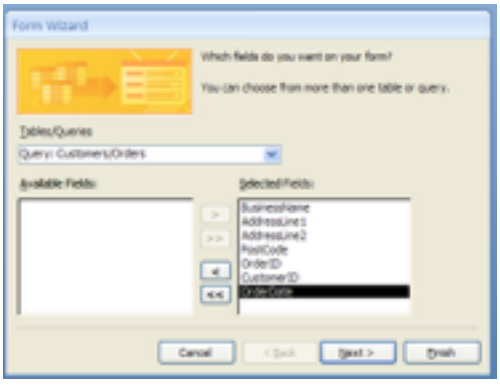
You can see that this is close to the diagrammatic representation of an order form at the beginning of this section.

Now that you have created both the required queries, you can start to construct the form itself.

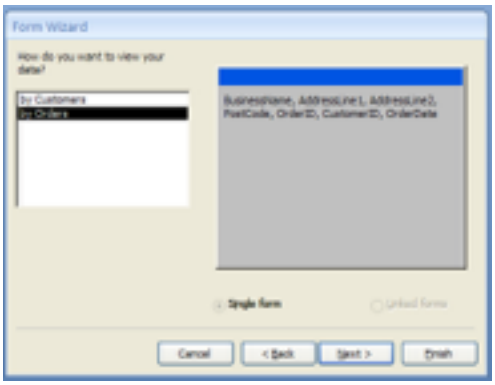
The Main Form

You are going to use the Form Wizard to create a columnar layout form that will constitute the Main form. Assuming that the database is open...

- ➢ click the **Create** tab, then **More Forms** in the **Forms** group
- ➢ click **Form Wizard**
- ➢ select the Query *Customers/Orders* in the drop-down box for the source Table or Query to be used for this form, as this contains the fields we want for our Main Form



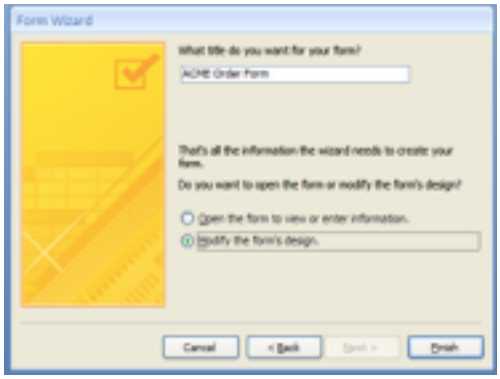
➢ select *all* the fields in the query, and click **Next**



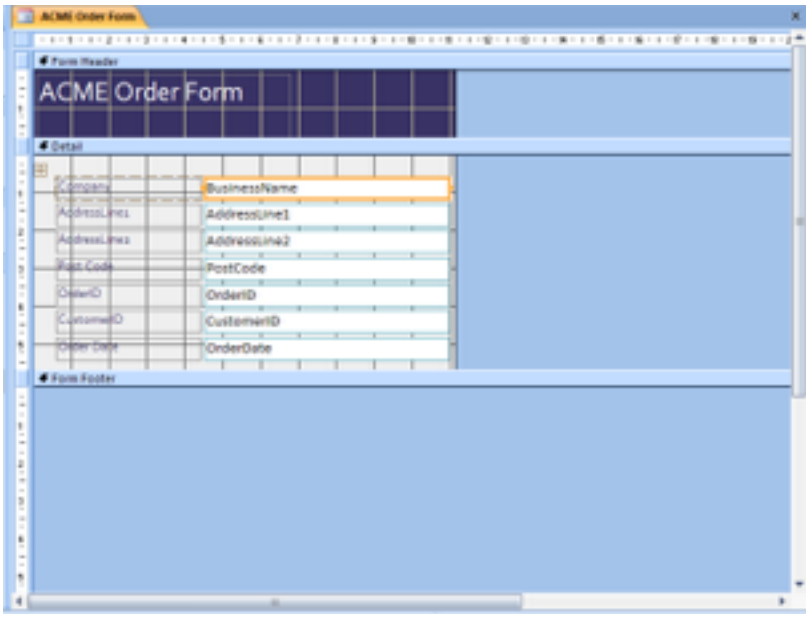
➢ we want to view our form **by Orders** (rather than Customers), so this is what we now choose.

(Note that if we view this form **by Customers**, we could set up a subform for the orders of each customer and then set up our Order Lines as a subform of that form. In this case, we are going for a simpler model of viewing the lists of order lines by orders.)

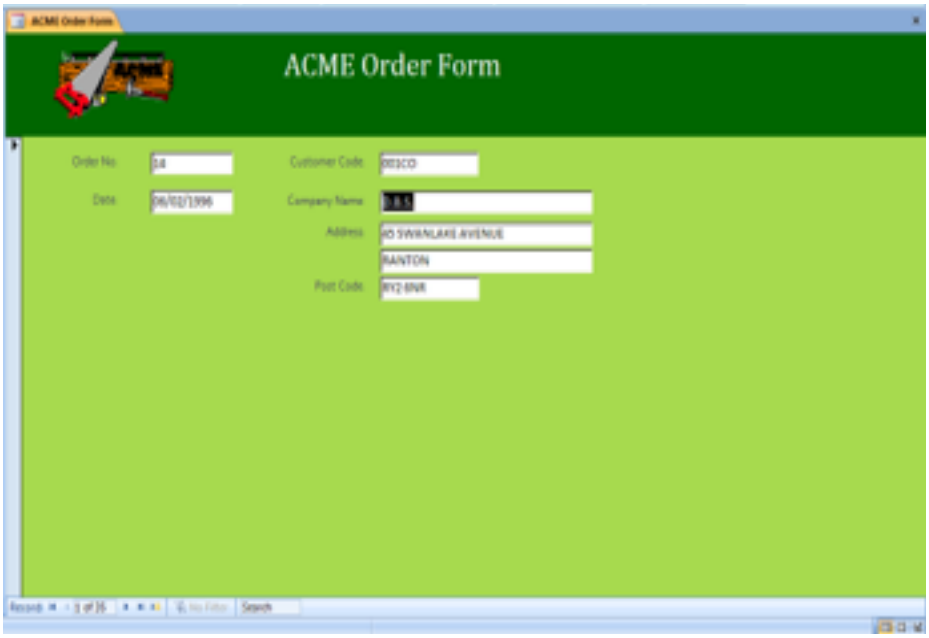
- ➢ select columnar layout (if not already selected) and click **Next**
- ➢ accept the default style and click **Next**
- ➢ type ACME Order Form in the title box (this is the label that will appear in your form header), select the radio button **Modify the form's design** and click **Finish**



The form design screen appears looking something like this.



The next step is to customise the Main form by making design changes so that it looks similar to this – you can customise it with your own choice of fonts and colours, but use the same "house style" you applied to the ACME Customers form.



This will involve such things as:

- dragging the borders to make the form wider and deeper
- re-positioning the fields and labels according to the picture above
- setting the Tabs in the correct order
- deleting the label *Address Line 2*
- changing the caption properties as follows: *OrderID* to Order No, *OrderDate* to Date, *CustomerID* to Customer Code, *Company* to Company Name, *AddressLine1* to Address, *PostCode* to Post Code (adjusting the sizes of the label boxes if necessary)

Remember that you can view and change properties by clicking Property Sheet

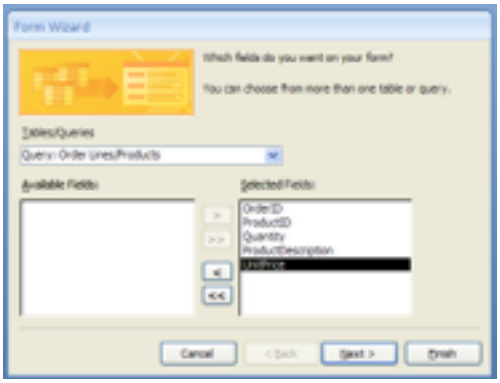
- possibly changing the **Back Style** and **Back Color** properties of labels and fields
- changing the **Back Color** property of the Header and Detail areas to suit your "house style"
- adding the company logo and adjusting as necessary
- checking the appearance of your form in Form View from time to time

When you are satisfied with the design, save the form with the name ACME Order Form Main.

The Subform

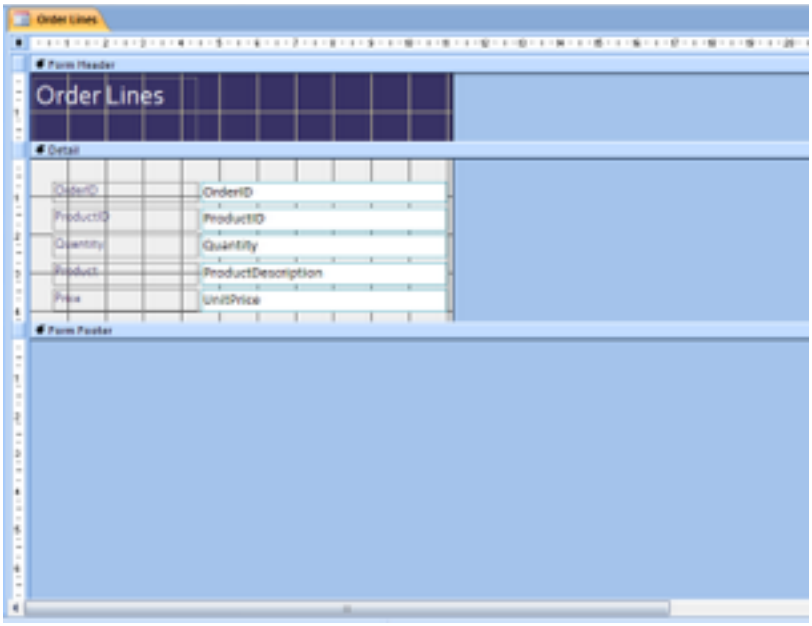
You are now going to use the Form Wizard to create a datasheet layout form which will constitute the Subform part of the Order form.

- ➢ click **Create**, **More Forms**, **Form Wizard**
- ➢ from the drop down box select the query **Order Lines/Products** and select all fields



- ➢ click **Next**
- ➢ select to view data *by Order Lines* and *Single form*; then click **Next**
- ➢ select *Datasheet* layout and click **Next**
- ➢ accept the default style and click **Next**
- ➢ if not already named thus, name the form *Order Lines*, click the radio button **Modify the form's design** and then click **Finish**

The form design screen will appear.



Because you have created the form in Datasheet layout, you now have four possible views of the form: Form, Datasheet, Layout and Design. When the form is incorporated as a subform, it will be seen in Datasheet View. To get an idea of what that will look like, select **Datasheet View**. (You can do this by right-clicking the form tab and then clicking the particular view, or clicking on the relevant view button in the bottom right-hand corner of the database window.)

The datasheet for all *the records in the Order Lines table appears.*

- ➢ return to **Design View**
- ➢ drag the form boundaries to give yourself plenty of room
- ➢ so that you can work on the fields independently as necessary, click on the arrowed cross-hair in the top left-hand corner of the Form Detail section (this highlights all the captions and fields), then click **Arrange**, then **Remove**

Because the subform will display in Datasheet view it does not require many design changes. Note that the Datasheet column headings are given by the captions. Make the following changes to the captions

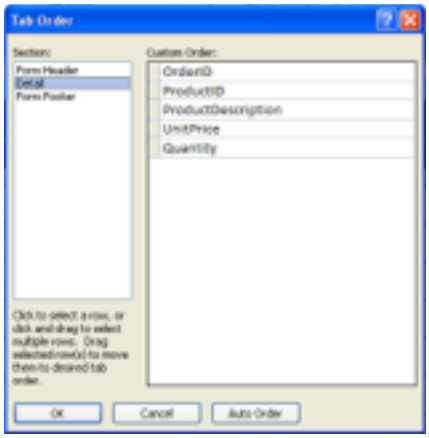
- ➢ change the *ProductID* caption to *Product Code*
- ➢ change the *Product* caption to *Description*
- ➢ change the *Price* caption to *Unit Price*

Remember to check your working by switching to **Datasheet View** from time to time.

The sequence of the columns in **Datasheet View** is given by the **Tab Order**. The required sequence is:

OrderID, ProductID, ProductDescription, UnitPrice, Quantity

➢ click **Arrange**, **Tab Order** and resequence the *Quantity* field as shown below



➢ click **OK**

➢ ensure fields have the correct format as necessary (e.g. currency) and justification (e.g. right for numeric) – you can check/modify justification in **Home**/ **Font** group

➢ check the result in **Datasheet View** and re-size columns as necessary to display fields and headings fully

Calculated Control for Cost of an Order LIne

Next you will create a new control – a text box that will contain the calculated cost of each order line.

➢ ensure there is sufficient space in the Detail area to add a field under *UnitPrice*

➢ click on **Text Box** (in the **Controls** group) and 'drop' the new control underneath the *UnitPrice* control

A new control appears consisting of a Text Box and a label. The Text Box is "Unbound" which means it is not bound to any field in a table.

➢ select the label and change the caption to *Cost*

➢ select the Text Box control and change the **Name** property to *Cost*

The control, now named *Cost,* is calculated by multiplying the *UnitPrice* by the *Quantity*. It is written as an expression thus: =[UnitPrice]*[Quantity]

➢ To tell Access where to find the data, enter this expression in the **Control Source** property of *Cost* (this is most easily achieved by clicking on the ellipsis [...] and

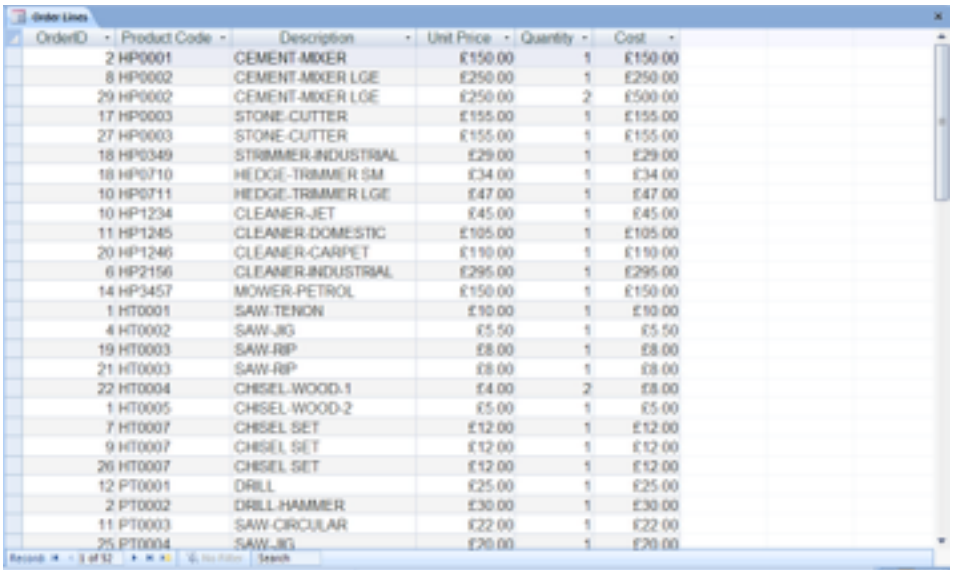
opening Expression Builder – you now have an expandable space into which any expression can be entered)

➢ in the **Format** property, select *Currency* (and right justify the field)

➢ switch to **Datasheet View** to adjust the column heading to get the best fit and to check that the new control is working correctly

If you see the error messages #Name? or #Error? in the *Cost* column it means you have made a mistake entering the expression. Return to Design View to correct it. You must take care to type the field names *exactly* as they are in the table. Access is not case sensitive but it will generate error messages if you do not get the letters and spaces correct.

The Datasheet View should look like this.



OrderID	Product Code	Description	Unit Price	Quantity	Cost
2	HP0001	CEMENT-MIXER	£150.00	1	£150.00
8	HP0002	CEMENT-MIXER LGE	£250.00	1	£250.00
29	HP0002	CEMENT-MIXER LGE	£250.00	2	£500.00
17	HP0003	STONE-CUTTER	£155.00	1	£155.00
27	HP0003	STONE-CUTTER	£155.00	1	£155.00
18	HP0349	STRIMMER-INDUSTRIAL	£29.00	1	£29.00
18	HP0710	HEDGE-TRIMMER SM	£34.00	1	£34.00
10	HP0711	HEDGE-TRIMMER LGE	£47.00	1	£47.00
10	HP1234	CLEANER-JET	£45.00	1	£45.00
11	HP1245	CLEANER-DOMESTIC	£105.00	1	£105.00
20	HP1246	CLEANER-CARPET	£110.00	1	£110.00
6	HP2156	CLEANER-INDUSTRIAL	£295.00	1	£295.00
14	HP3457	MOWER-PETROL	£150.00	1	£150.00
1	HT0001	SAW-TENON	£10.00	1	£10.00
4	HT0002	SAW-JIG	£5.50	1	£5.50
19	HT0003	SAW-RIP	£8.00	1	£8.00
21	HT0003	SAW-RIP	£8.00	1	£8.00
22	HT0004	CHISEL-WOOD-1	£4.00	2	£8.00
1	HT0005	CHISEL-WOOD-2	£5.00	1	£5.00
7	HT0007	CHISEL SET	£12.00	1	£12.00
9	HT0007	CHISEL SET	£12.00	1	£12.00
26	HT0007	CHISEL SET	£12.00	1	£12.00
12	PT0001	DRILL	£25.00	1	£25.00
2	PT0002	DRILL-HAMMER	£30.00	1	£30.00
11	PT0003	SAW-CIRCULAR	£22.00	1	£22.00
25	PT0004	SAW-JIG	£20.00	1	£20.00

Hiding a Column on a Subform

Although the Order ID field must be present because it is the link between the Main form and the Subform, it is not necessary to see it beside every order line. Therefore it can be hidden.

➢ in **Datasheet View**, place the pointer on the column name *OrderID* and click to highlight the column

➢ right click, and from the drop down menu click **Hide Columns**

The Order ID column disappears.

➢ return to **Design View**

Calculated Control for Order Subtotal

You are now going to create a second calculated control, which will calculate the total cost of all the order lines on an order. There is only one total on each order so this is placed in the Form Footer.

Since Subform Headers and Footers are not displayed in Datasheet View you may wonder why you should place a control in the Footer. The reason is that in order for the total to be displayed in the Footer of the Main form it must already exist in the Subform. It cannot be calculated directly on the Main form.

- ➢ place the pointer on the lower edge of the Form Footer band and drag down to create enough space for a control
- ➢ click the Text Box and 'drop' the new control in the Form Footer to see the "Unbound" control and label appear
- ➢ select the Text Box control and change the **Name** property to *Subtotal*; change the caption to *Subtotal*

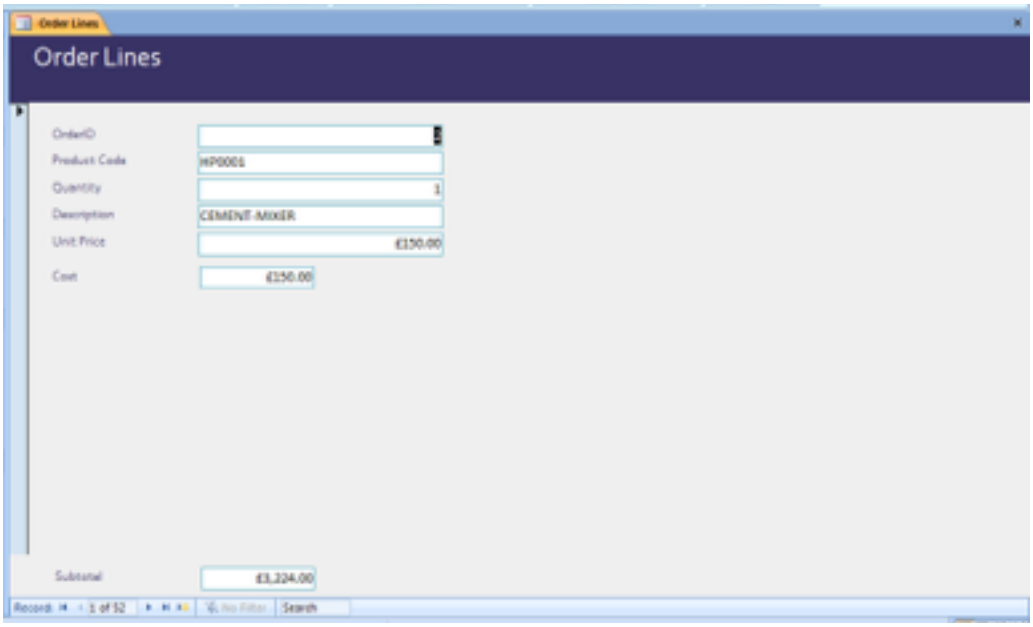
The unbound control needs a meaningful name so that it can be referred to elsewhere.

To calculate the total cost on an order you must add up all the individual costs of each order line. The expression that does this is:

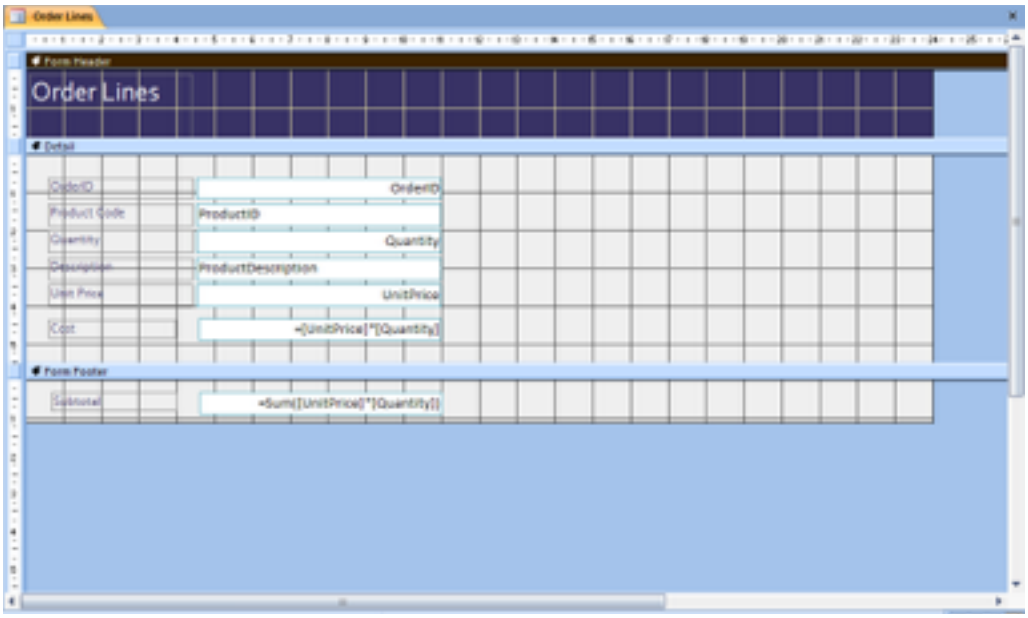
=Sum([UnitPrice]*[Quantity])

- ➢ enter this expression in the **Control Source** property using the Expression Builder dialogue box – be sure to type the brackets accurately
- ➢ in the **Format** property select *Currency* (so that the £ sign will be displayed) and ensure that the amount is right justified

Check that you have created the Subtotal correctly by switching to Form View which should look similar to the picture below.



Your Design View should similar to this.



You have almost completed the Subform. It only remains to set the Form properties so that the subform always displays as a datasheet and the scroll bars and navigation buttons are removed:

- in **Design View**, click on the top left control box of the whole form
- in the Property Sheet, change the **Scroll Bars** property to *Neither*
- change the **Navigation Buttons** property to *NO*

➢ **Close** the form and save the changes

Inserting the Subform

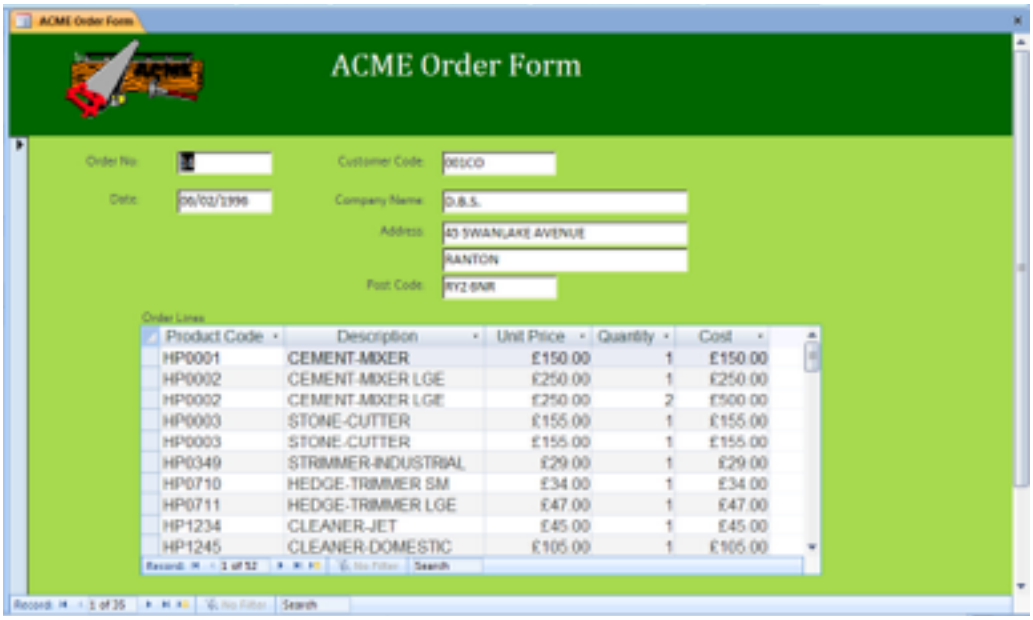
To place a subform into a main form is very straightforward: first the main form is opened in Design View, and then the subform is highlighted and dragged from the Navigation Pane and dropped into a vacant area.

➢ open the main form, (named ACME Order Form Main, or possibly ACME Order Form) in **Design View**

➢ highlight the subform (named Order Lines) and drag and drop it into the space below the main form fields – note that the subform is dragged on as a small rectangle – drop it at the point where you want the top left-hand corner of the subform to appear

The Subform appears as a very large control with a label. The details of the Subform cannot be seen in Design View in the same window as the Main form.

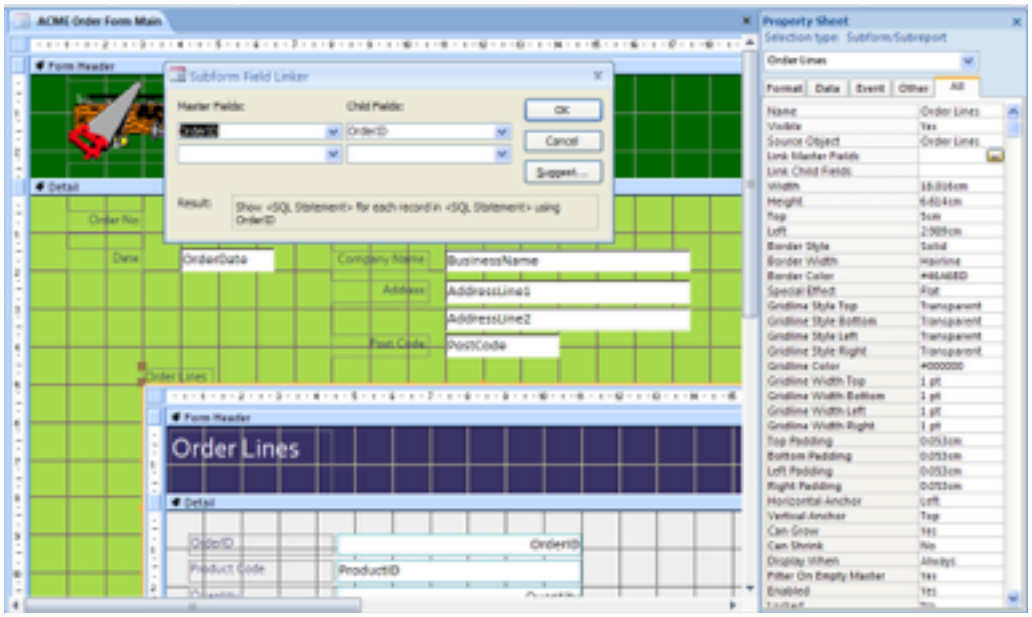
The Subform control is like any other control in that it has properties and can be re-sized. After re-sizing, the Main/Subform should look like the picture below. Note that the datasheet in the Subform still shows all records – the Subform must be linked to the Main form so that only the specific order lines relating to each order are displayed.



The most important properties in a subform are the ones that link it to the Main form. In this case it is the field *OrderID* which links the *Child* table (Order Lines) in the Subform to the *Master* table (Orders) in the Main form. To set the link properties:

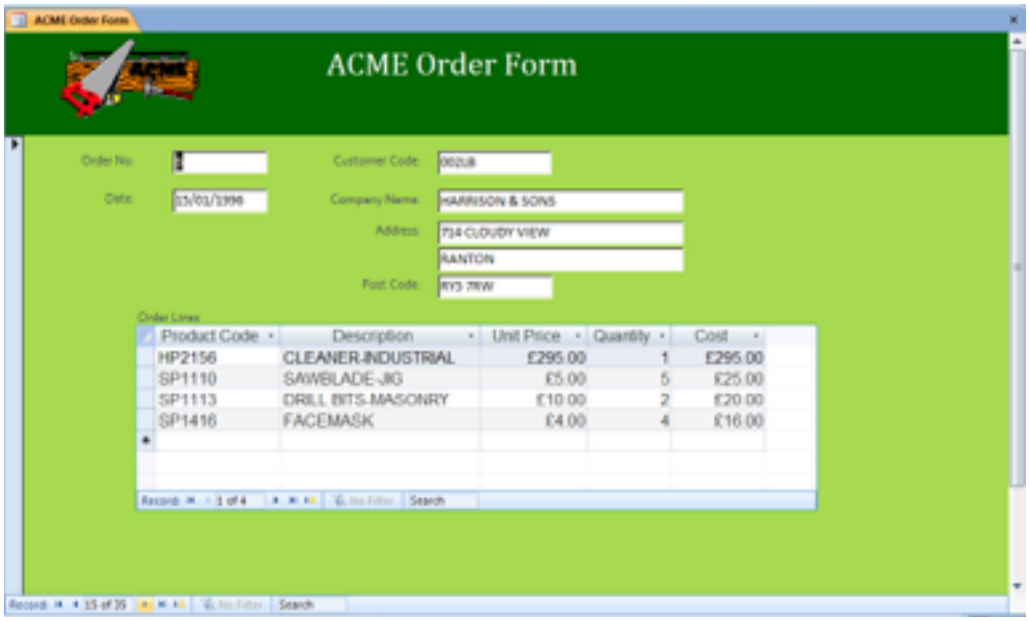
➢ (in **Design View**) highlight the subform and open the Property Sheet if it is not already on screen

➢ click in the ellipsis [...] in the **Link Master Fields** property - the **Subform Field Linker** dialogue box should open as shown below with *OrderId* already in the Master and Child fields



➢ click **OK**

➢ switch to Form View to see the result – it should appear like the picture below with only specific order lines displayed; (note that at this stage there are no main form subtotals showing – these have to be added)



Remember to check the size of the Subform section (can you see all the columns?) This may need a bit of trial and error to get right. Return to Design View to make any adjustments and then switch to Form View to check and so on.

When you are satisfied with the appearance, you may wish to save the changes at this point to protect the work you have completed. You can save the Main/Subform under a different name if you want, but this is not necessary.

When the form is in Design View, you have access to the subform by clicking on the subform control. You can also select the subform from the Navigation Pane and open in Design View.

Calculated Controls for Main form totals

At this stage the Order form does not display any totals. You know that a total for each order exists because you put a calculated control (*Subtotal)* in the Footer of the Subform. But how are you to display that value on the Main part of the form?

First you will create a new Text Box control in the Main form Footer. Then in it you will tell Access to display the value of *Subtotal* from the 'invisible' Subform Footer.

The Subtotal on the Main form

- ➢ return to **Design View** (or open the ACME Order Form in **Design View**)
- ➢ scroll down it until you can see the Form Footer band
- ➢ place the pointer on the lower edge of the Footer Band and drag it down until there is room for three 'total' controls
- ➢ click Text Box and drop the new control in the Footer just beneath the right hand end of the Subform control – an Unbound Text Box control and label appear
- ➢ change the caption to *Subtotal:*
- ➢ select the Unbound control and change the **Name** property to *Subtotal*

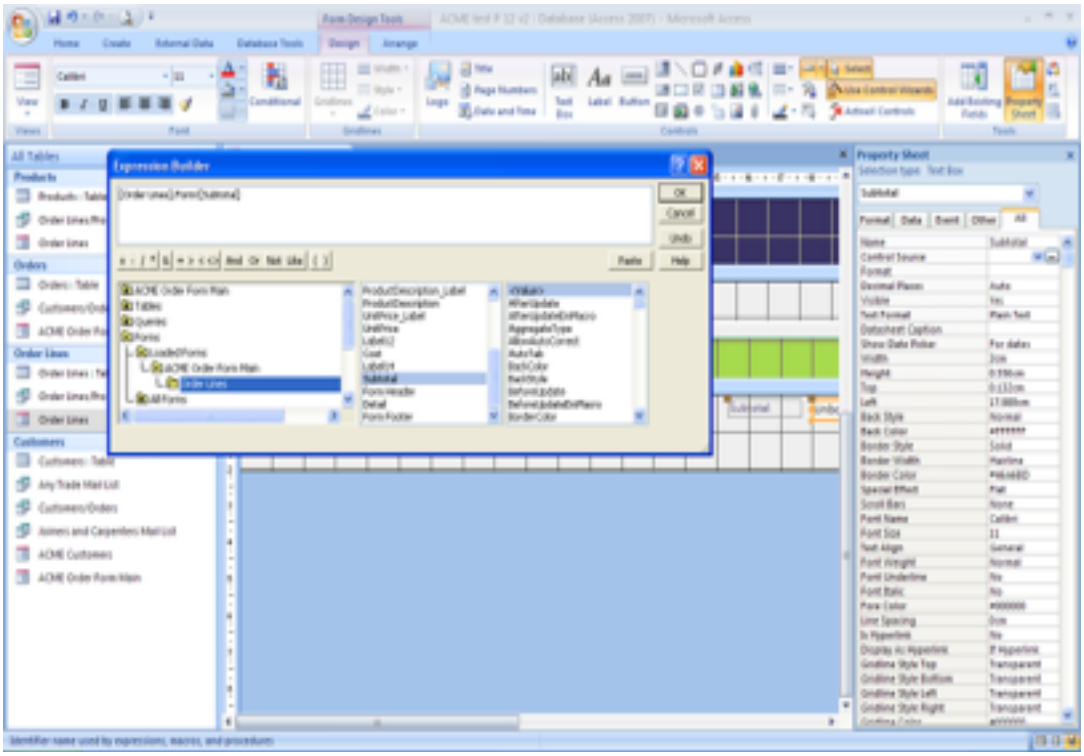
***Names* and *Captions* may be the same but you cannot give the same *Name* property to two controls in the same object.**

- ➢ for the Unbound control to display the *Subtotal* that is in the Subform Footer, you need to enter an expression in the **Control Source** property – click on the ellipsis [...]

Let Access do all the hard work! Use the Expression Builder dialogue box to identify the correct source control. Expression Builder is a really useful tool that can make life easier for an Access developer if used properly.

We want the Order Lines *Subtotal* we set up in the subform to be displayed in the Main form's footer so that it displays the sum owing for each particular order. What you need to do is find the Control Name required via the Expression Builder.

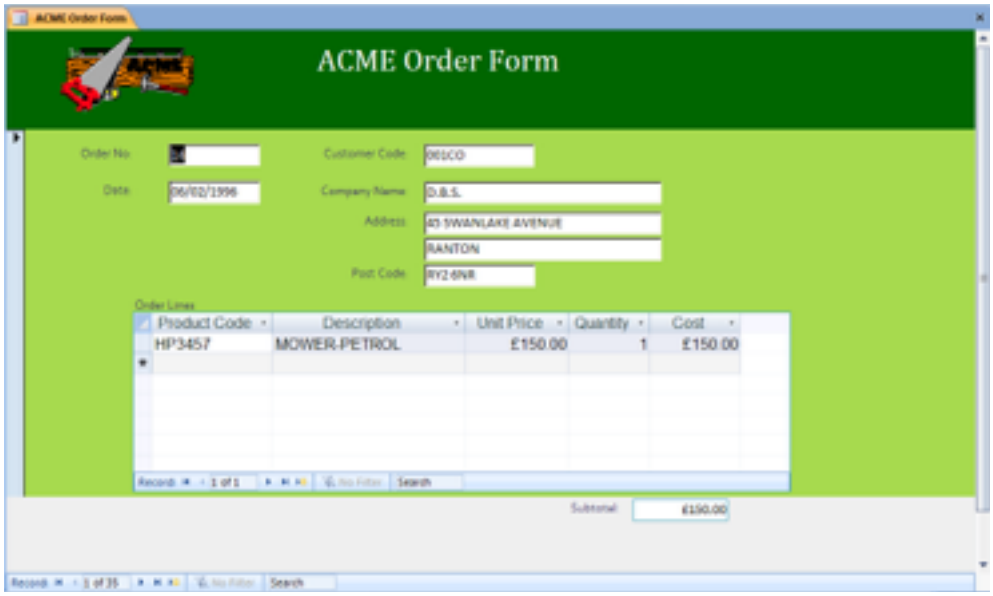
Note that the Order Lines subform is accessed via "Forms... Loaded Forms...". If you went down the "All Forms" route the expression for the control would look OK, but would not make sense to Access and it would display an error.



- ➢ trace down (by double-clicking each box) from Forms to Loaded Forms to ACME Order Form (be clear what you called it!) to Order Lines
- ➢ scroll down the list of available fields in the middle box and highlight *Subtotal*
- ➢ **Paste** the expression into the Expression Builder window and click **OK**

You could of course have typed in the correct expression directly into the Control Box if you preferred. As there is not much room, it is strongly advised that you always open the Expression Builder to enter Control Source information.

- ➢ set the **Format** property to *Currency* and right justify the field and the caption
- ➢ go to **Form View** to see the result



If you see #Name? or #Error? in your subtotal box then you have mistyped or selected the wrong expression. Go back to Design View, select the control, click in the *Control Source* property, and correct it.

The other two controls you will add are both very straightforward – a box to display the VAT payable and another to display the Grand Total.

VAT and the Grand Total

The VAT @ 17.5% is calculated using the expression:

=[Subtotal]*0.175

The Grand Total expression is even simpler:

=[Subtotal]+[VAT]

- ensure there is space for two more controls in the form Footer
- click Text Box and drop the new control in the Footer just beneath the Subtotal control – an Unbound Text Box control and label appear
- change the caption to *VAT@17.5%:*
- select the Unbound control and change the **Name** property to *VAT*
- set the **Control Source** property to *=[Subtotal]*0.175*
- set the **Format** property to *Currency* and right justify the field and caption
- check the **Form View** (do not bother about getting the alignment of the controls right just yet)

- ➢ click Text Box and drop the new control in the Footer just beneath the VAT control – an Unbound Text Box control and label appear
- ➢ change the caption to *Grand Total:*
- ➢ select the Unbound control and change the **Name** to *GrandTotal*
- ➢ set the **Control Source** property to *=[Subtotal]+[VAT]*
- ➢ set the **Format** property to *Currency* and right justify the field and caption
- ➢ check out **Form View** and if satisfied, close the form and save your changes, renaming the completed form if required

Aligning the Controls

Your form will be easier to read and look more professional if the controls, especially the totals, are neatly aligned.

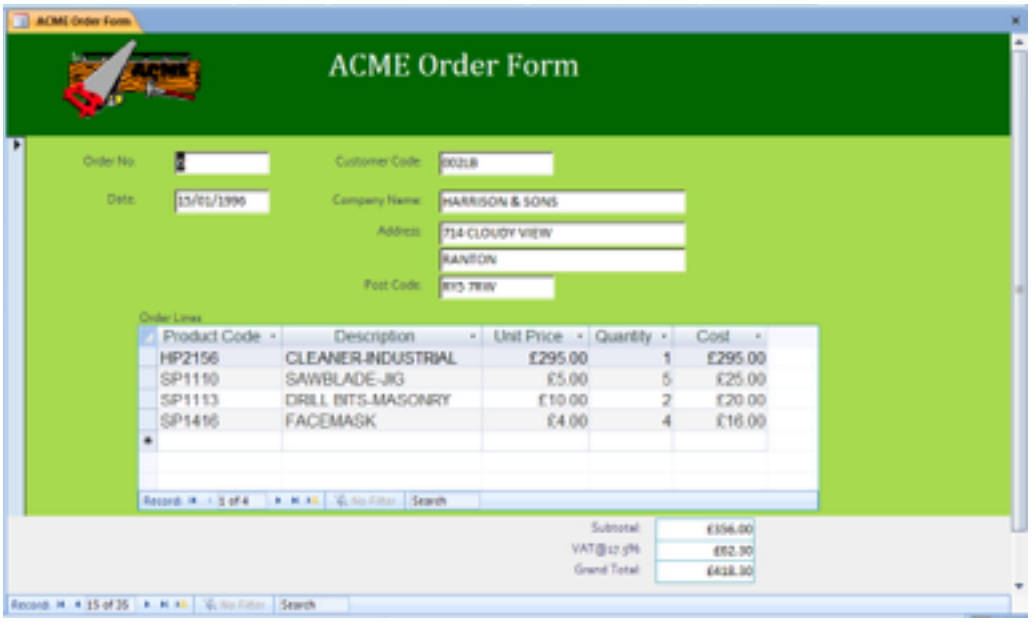
Access has a *Snap to Grid* feature – the default setting is *OFF*. Having *Snap to Grid* on can make it difficult to make fine adjustments to the positioning of a control. This means that when you move a control and then release it, the top left corner of the control will align to the closest grid line. To toggle the *Snap to Grid* feature on and off, click **Arrange, Snap to Grid**.

(Alternatively you can hold down the **Ctrl** key while moving a control. This temporarily suspends *Snap to Grid* allowing you to place the control more precisely.)

If necessary re-position *Subtotal, VAT* and *Grand Total.* This can be done by selecting and moving controls carefully to align (overlap) the borders. Controls can also be re-sized to help alignment. You should position these controls so there is no horizontal space between them.

When you are satisfied, close the form saving your changes.

Your finished product should now look something like this.



Using the Form

Filling in this order form requires the user to enter the Customer ID, Date, Product ID and the Quantity. All the other details will fill in automatically. You can cause the current date to be displayed automatically by setting the **Default Value** property of *Order Date* to **=Date()**. This is an Access function for displaying the current date. To display the Current Date and Time use the **Now()** function.

You do not want the user to be able to alter the automatically displayed details so you need to make them *read-only*. This is achieved by setting the locking property to YES and the ENABLING property to NO for these controls.

Locking a column or field prevents editing of the data in it, and dis-enabling prevents the field from getting the focus of the cursor, thus avoiding unnecessary confusion should the user attempt to edit the contents.

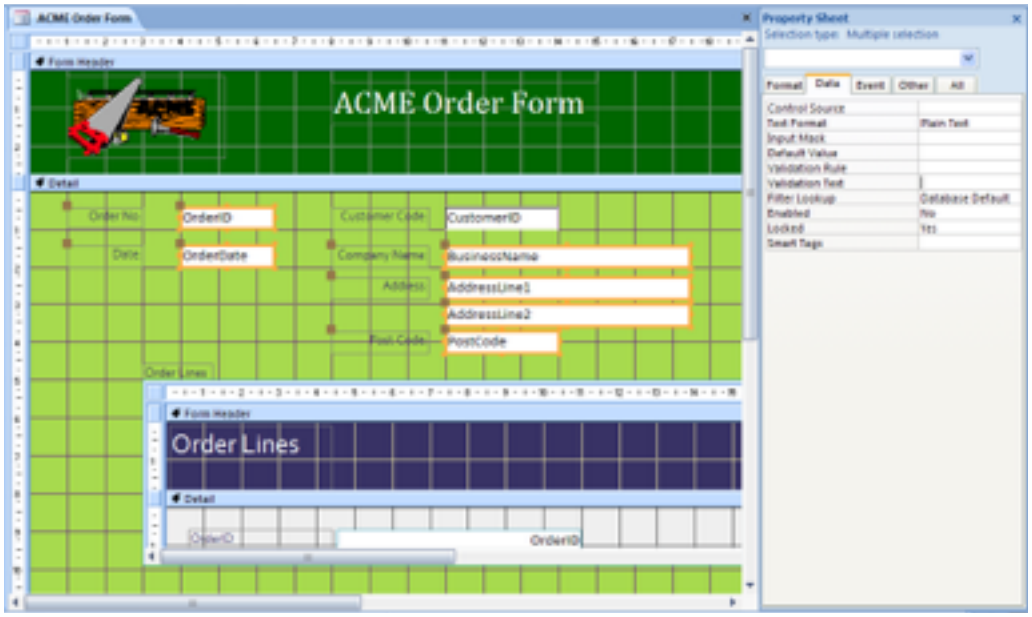
Locking and Dis-Enabling Controls

Controls are locked by setting their **Locked** property to *Yes* and dis-enabled from receiving cursor focus by setting the **Enabled** property to *No*. If there are several controls which need to be read-only, then they can be selected together and the **Locked** and **Enabled** properties set for the multiple selection. You need to make these settings on your Main/Subform for the controls: OrderID, OrderDate, BusinessName, AddressLine1, AddressLine2, PostCode, Product, UnitPrice, Cost, Subtotal, VAT and GrandTotal. But first make the default order date the current date.

- ➢ open the Order form in **Design View**
- ➢ select *OrderDate* and set the **Default Value** property to *=Now()*

- ➢ hold down the **Shift** key and make a multiple selection of these controls: OrderID, OrderDate, BusinessName, AddressLine1, AddressLine2, PostCode, Subtotal, VAT, GrandTotal (take care *not* to select CustomerID)

- ➢ set the **Locked** property to *Yes* and the **Enabled** property to *No* (see the illustration below)



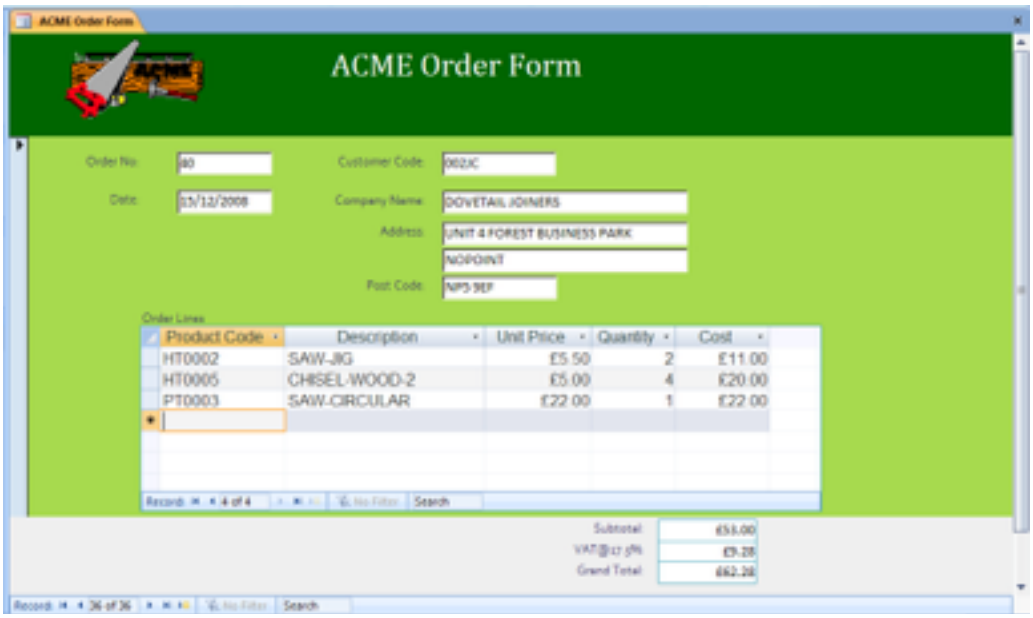
The Subform controls are also available via its design window. Note that if you select the Subform's overall control this will affect all the columns displayed! You should see the words "Multiple selection" in the property header if you are correctly selecting several controls for amendment.

- ➢ make a multiple selection of the following fields: ProductDescription, UnitPrice, Cost, Subtotal (take care *not* to include ProductID or Quantity) and set the **Locked** property to *Yes* and **Enabled** to *No*

- ➢ close the form saving your changes

Entering Sample Data

Your form is now ready to try out. Go to a blank form either by clicking the **New (blank) record** button shown in the Record Navigation bar at the bottom of the form, or by clicking **Home** (if not already selected), then **New** in the **Records** group. Refer to the sample data listings for Customers and Product IDs and try entering some new orders. An example is shown below.



At the moment, the largest order in the sample records is four order lines. The size of the subform is therefore only four lines deep. If you fill out an order with more than four lines, the subform will expand *up to a maximum defined by the size of the subform control that you designed – in the above example, it is 7 order lines.* You may need to go back to Design View and increase the depth of the subform control if you want to cater for a larger number of order lines.

Checklist for Subforms

- Have you understood what a Main/Subform can do?
- You constructed queries to combine data for a form.
- You used the Forms Wizard to create a Main form and a Subform.
- You inserted a Subform onto a Main form.
- Calculated controls were constructed.
- The form design was refined.
- Have you understood why controls should be locked?
- You entered some new order records.

Reports in MS Access 2010

In this session you will create several reports including a price list for ACME Sales and a set of mailing labels. Initially these will be created using the Access Report Wizard. You will then customise your reports in Design View. To do this will require a closer look at the Report Design View and some of the tools available in Access. There will then be a chance to look at the other options made available by the wizards.

What does a report do?

Reports allow you to organise and present information from your database. They are used when you want to print information in a professional manner, e.g. mailing labels, invoices, or sales reports. They can be customised to include graphics, graphs, calculated fields, different fonts, colour etc. They look like word-processed documents rather than typical database printouts providing you take a little care. They basically present customised view of your data.

Why not just print a form?

You have just created an order using a Main form and Subform. This shows that Access forms are very sophisticated and can also include graphics, graphs, calculated fields, different fonts, colour etc.

Reports are much more useful if you want to print information. However they cannot be used to enter or edit data. Reports allow you to group sets of records and create group totals and grand totals that would be either difficult or impossible with a form.

Obviously there is some overlap in the use of forms and reports. As a general rule it is best to use a report when you need a printed output, particularly where you want grouping of records and complex totals.

Types of Reports

1. **Tabular Reports**: These reports print data in rows and columns with groupings and totals. Variations include summary and group/total reports.
2. **Columnar Reports**: These reports print data and can include totals and graphs.
3. **Mailing – label Reports**: These reports create multicolumn labels or snaked-column reports.

Creating a report

There are three main stages needed to create a report with Access Wizards.

Deciding what you want to achieve - Planning is important. You should decide which tables and fields are required and you should have some idea of the final layout.

Creating the report (with the Report Wizard*)* - This involves opening a new report and making a series of decisions prompted by the wizard.

Customising the report - Once the wizard has created a basic report you can then easily make changes in the report design view.

The first report you will create is a price list for ACME Sales.

Deciding what you want to achieve

The price list will include the following fields, which are all in the *Products* table:

ProductDescription *ProductID* *Supplier* *UnitPrice*

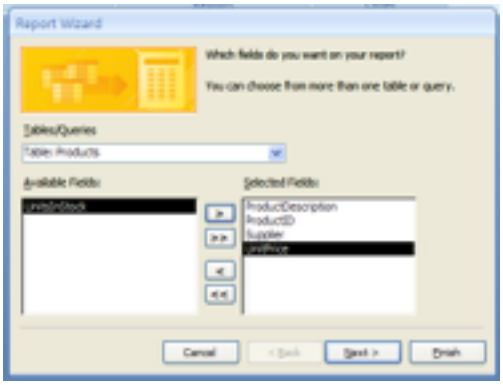
The layout will be as shown below and the report will be sorted alphabetically on the ProductDescription (name) field so that a potential purchaser can find any item easily.

ACME SALES - SPRING PRICE LIST			
ProductDescription	**ProductID**	**Supplier**	**UnitPrice**
~ ~ ~ ~ ~	~ ~ ~ ~ ~	~ ~ ~ ~ ~	~ ~ ~ ~ ~
~ ~ ~ ~ ~	~ ~ ~ ~ ~	~ ~ ~ ~ ~	~ ~ ~ ~ ~
~ ~ ~ ~ ~	~ ~ ~ ~ ~	~ ~ ~ ~ ~	~ ~ ~ ~ ~

Creating the report

- ➢ open the ACME database and highlight the *Products* table
- ➢ to create a new report based on this table, click **Create**, then **Report Wizard**

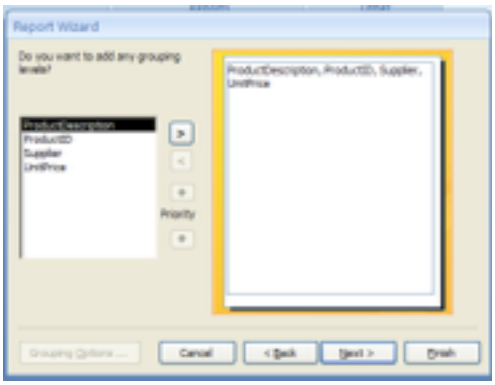
Creating a report is similar to creating a query - a dialogue box appears from which you can select the fields you want in the sequence you want them displayed – see the example below.



- ➢ ensure that the *Products* table is selected in the **Tables/Queries** box

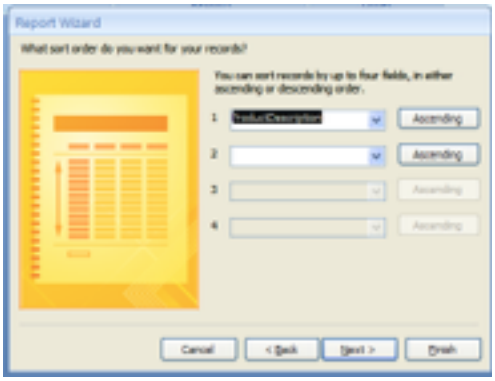
- ➢ from the **Available Fields** select the following in the correct sequence:
  ProductDescription, *ProductID*, *Supplier*, *UnitPrice*
- ➢ (this matches our report layout) – click **Next**

This dialogue box allows for grouping – we do not want any for this report.



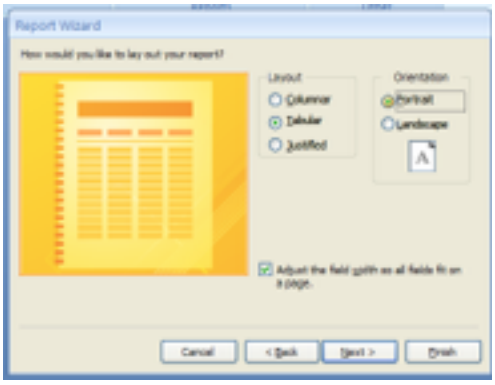
➢ click **Next**

This dialogue box allows for sorting. As per the requirement, we want to sort by Product Description.



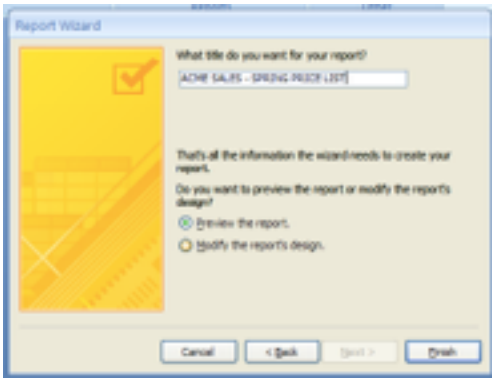
➢ enter this field name from the drop down list and accept the default sort sequence of **Ascending**; click **Next**

Use this dialogue box to set the layout for the report. In this case, we want a Tabular Layout and a Portrait Orientation. (Note that for wider reports, we would consider Landscape.) You could also click the other option buttons to view typical layouts.



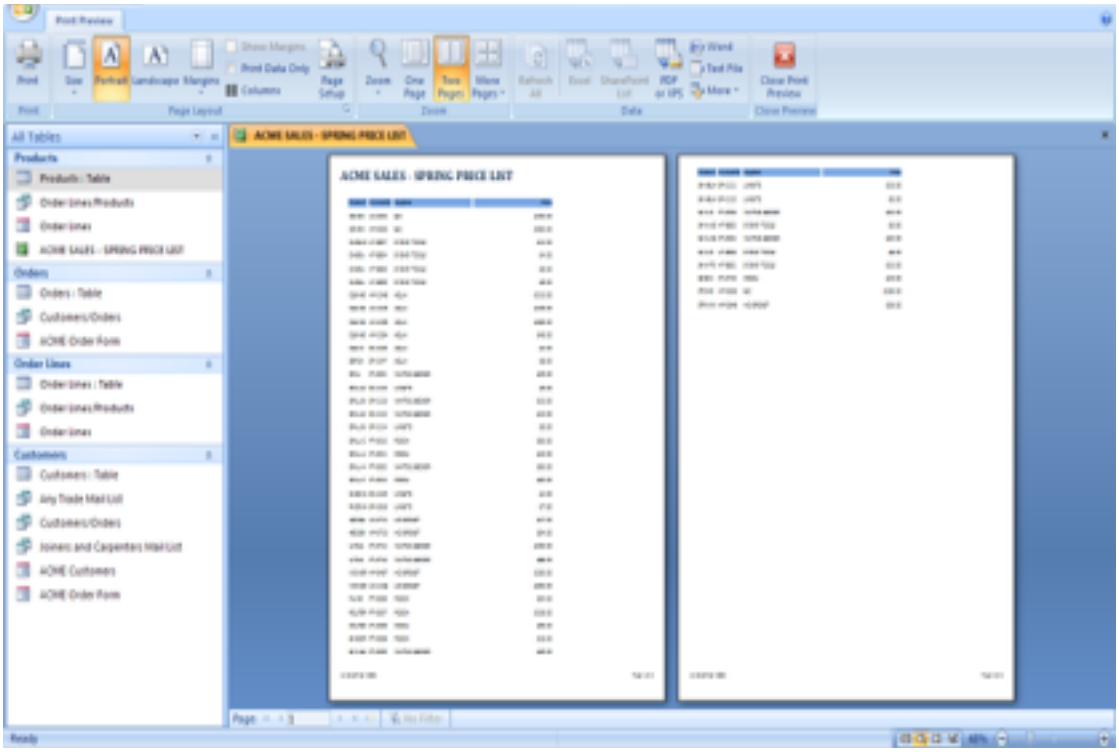
- ➢ click the option button for **Tabular** (if not already selected), then click **Next**
- ➢ choose a style, then click **Next**

The final dialogue box allows you to name the report and preview or modify the report. We want to see what the basic report looks like straight away.

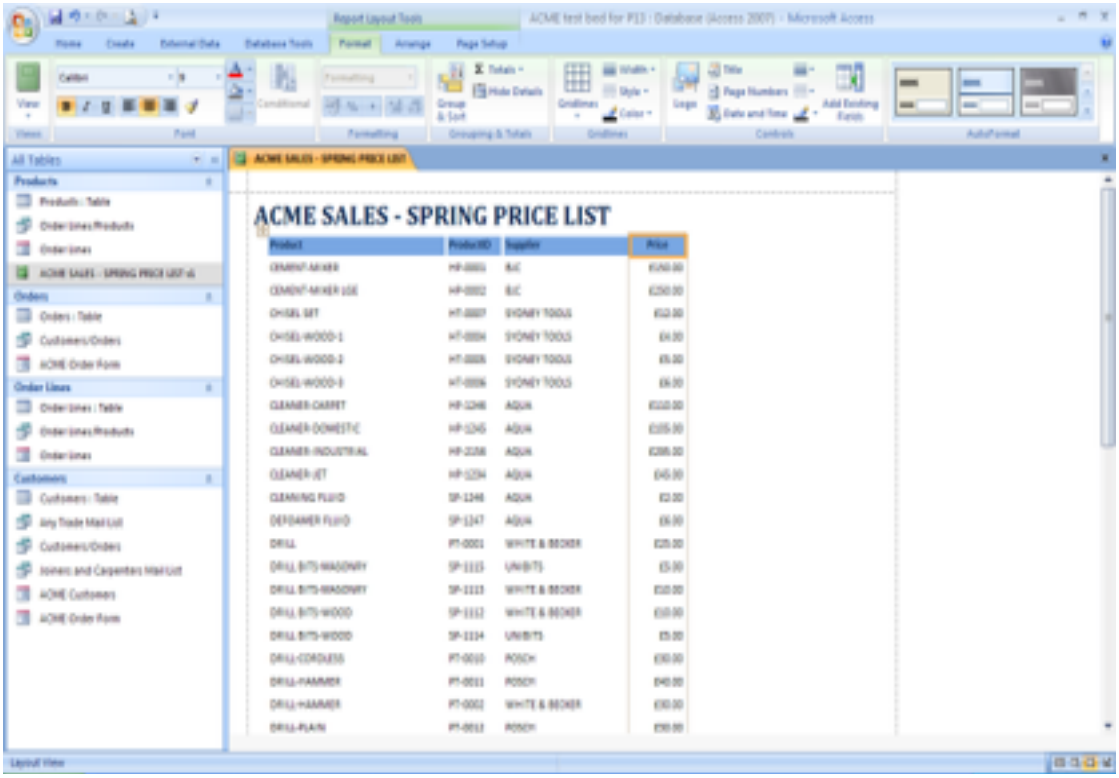


- ➢ enter the name of the report as: ACME SALES – SPRING PRICE LIST
- ➢ accept/select **Preview the report** and click **Finish**

The report is shown in **Print Preview**, 2-page layout. Investigate the other available views. Note that the column headings are based on the label captions – you can check this in **Design View**.



Whilst this is interesting, there is still some formatting to be done so that headings and field contents are fully visible. The required formatting is best accomplished in **Layout View** (although it can also be carried out in **Design View**.)



- select **Layout View**
- click the *Price* heading - this highlights the caption and outlines the column
- using the double-headed arrow on the right-hand edge of the caption, drag the box to the right to make the column wider; now drag the left-hand edge until it gives an appropriate width to the *Price* column – what you are doing is giving yourself some width to expand other columns
- drag the left-hand edge of the *Supplier* column to the right to give it a width which allows all field contents to be displayed fully
- do the same with the *ProductID* and *Product* columns

This is a 'quick and dirty' method of adjusting column widths. Use Design View to achieve similar effects. You can also adjust where headings appear in relation to columns of data (centring, left/right justifying...)

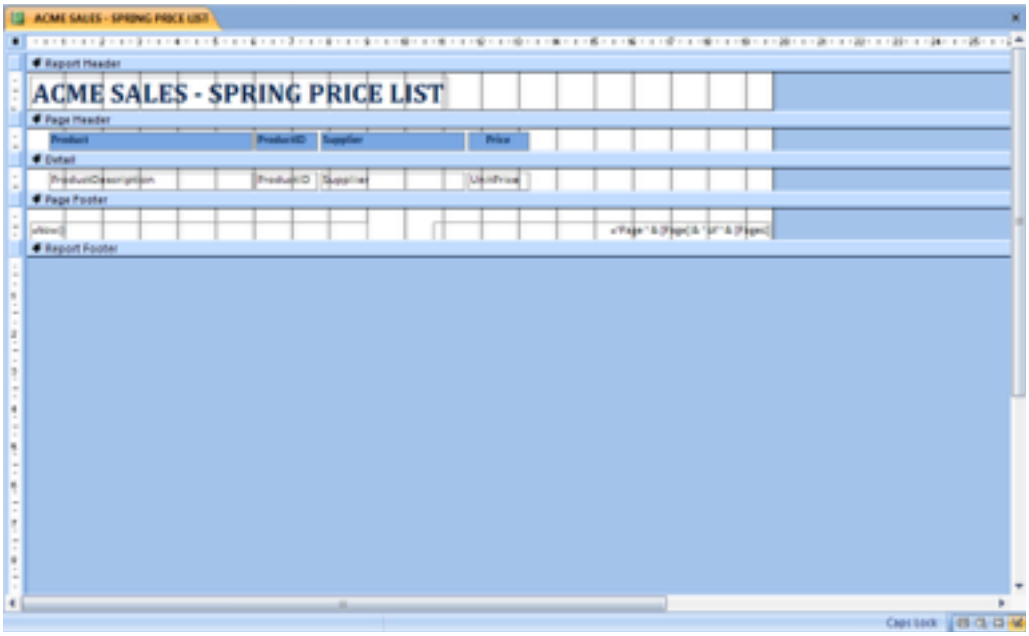
- when you have finished experimenting with layout and the different views, close the report, saving any required changes

Customising the report

As you can see from the printout, the report meets the basic requirements. We could, however, do some editing. To edit the report you need to be in Design View, so this is a good time to look at this in more detail.

Looking at Design View

To look at the report in the Design View click on the **Design** tab and then **View** and **Design View** or click the **Design View** icon in the group of 4 at the bottom right-hand corner of the window. As you can see, there are five distinct sections to a report.



The Report Header appears only at the beginning of the report.

The Page Header appears at the top of each report page for column headings.

The Detail section contains the main body of the report. It is repeated for every set of data until the record set(s) is (are) exhausted.

The Page Footer appears at the bottom of every page. It is used for footer information such as dates and page numbers.

The Report Footer appears once at the end of the report. It is used for grand totals.

It is also possible to have Group Headers and Group Footers. These would be used for a report with many groups of data.

Within these sections are controls. The controls used on your report are labels, text boxes and graphical lines. It is also be possible to use frames to hold graphs or pictures.

Labels contain titles. You can type whatever you like in them. In the report above, the column headings are taken from the label captions. These can be changed if required for the purposes of the report. Text boxes or controls can be used in several ways. They

can be used to bring the data from the source table or query to the report. Modify them using their Property Sheet. In a report, controls can be used to display such things as today's date and page numbers (using Access Functions such as **Now()**, which appears in the current report). They can also hold calculated fields such as totals, which we chose to omit for this report. Investigate the various views and check out the tools available – you should have met some of these already.

One new group is the **Grouping and Totals** one. Clicking **Group & Sort** brings up a dialogue box where you can see how your data is grouped and sorted – groupings and sorting can be modified here. At this stage, you could experiment with grouping (and/or sorting) by one or more fields; however, it would probably not lead to a very useful result. We will look at grouping later on.

Check out the properties of various controls. Remember, you can do this by highlighting the control and then clicking **Property Sheet**.

You can also click on **Add Existing Fields** if you want to include additional fields on any report. (This opens the **Field List** window.) The **View Code** button (in the **Tools** group) opens a window where you can write code in Visual Basic programming language, but this is well beyond the scope of these practicals.

Now you have had a closer look at Design View it is time to modify your report.

The first edit will be to delete the textbox in the Page Footer giving today's date.

- ➢ select the left-hand control in the Page Footer (containing '=Now()') and press Delete

For practice, change the title font and its size.

- ➢ select the label in the Report Header and drag it until it is in the middle
- ➢ open the Property Sheet for this control, then change the Font Size to 22 and the Font Name to one of your choice – note that you can also make this kind of change in the **Home** group

You will now amend the *Price* column. The column is at present wider than needed and the title is not placed directly above the data.

- ➢ click on the Price label to select it; replace **Price** in the label with **Price per Unit** (adjusting the width as necessary) and position the column heading over the prices
- ➢ when you are satisfied with the report, Close it saving the changes

Creating a report with the Mailing Label wizard

This report will be a set of mailing labels for the customers of ACME Sales. Labels come on continuous sheets or rolls, or are sheet fed (like A4 pages.) The labels are normally the peel-off type and are printed when fed through the printer.

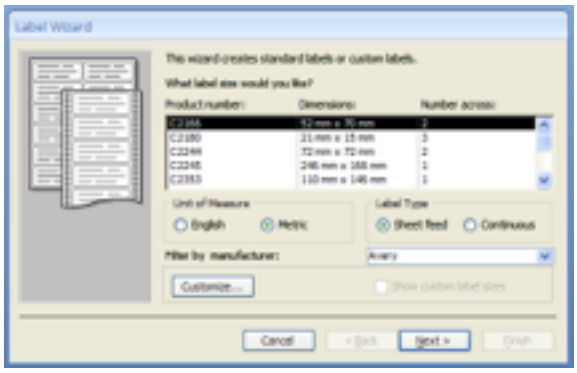
Deciding what you want to achieve

You have to know the size of the mailing label and what type it is. Once this is decided upon, creating the labels is a straightforward task.

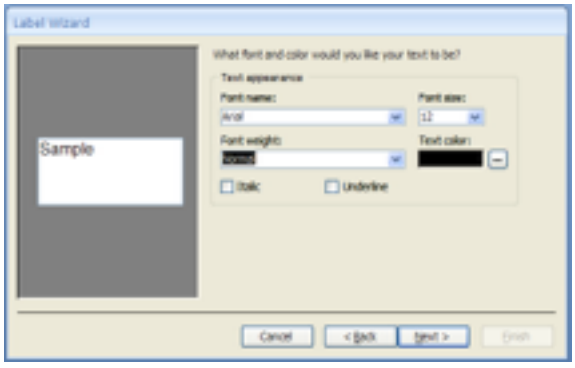
Creating the report

- ➢ open the ACME database and highlight the *Customers* table
- ➢ to create mailing labels based on this table, click **Create**, then **Labels**

You have to choose the layout depending on what labels you have. Since we are not using actual labels, it does not matter for now what we choose. Note that if we want to use non-standard labels, we can customise and name our new custom label for future reference.

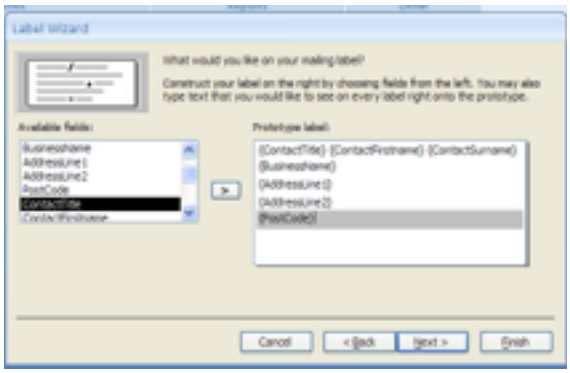


- ➢ in this case, we can accept the first label offered (note that the first dimension is depth and the second is width) – click **Next**

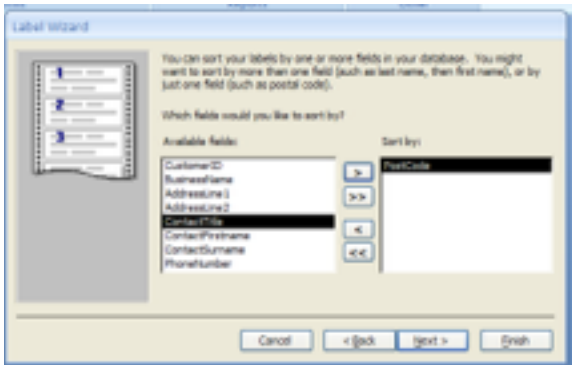


- ➢ choose a font of your own choice (or accept the default) – click **Next**

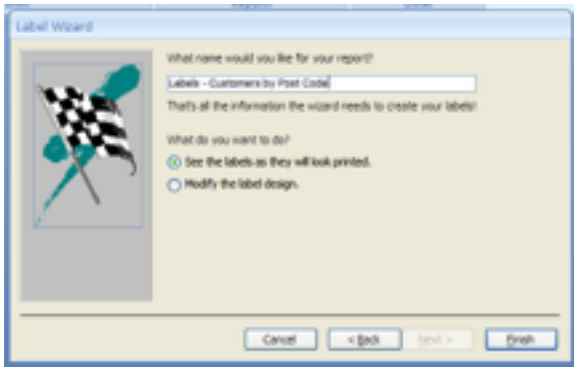
You are going to create a label layout like the one below. In order to do this, you select field names from the **Available** list to the **Prototype label** and format the label using spaces, new lines and any other useful punctuation or formatting.



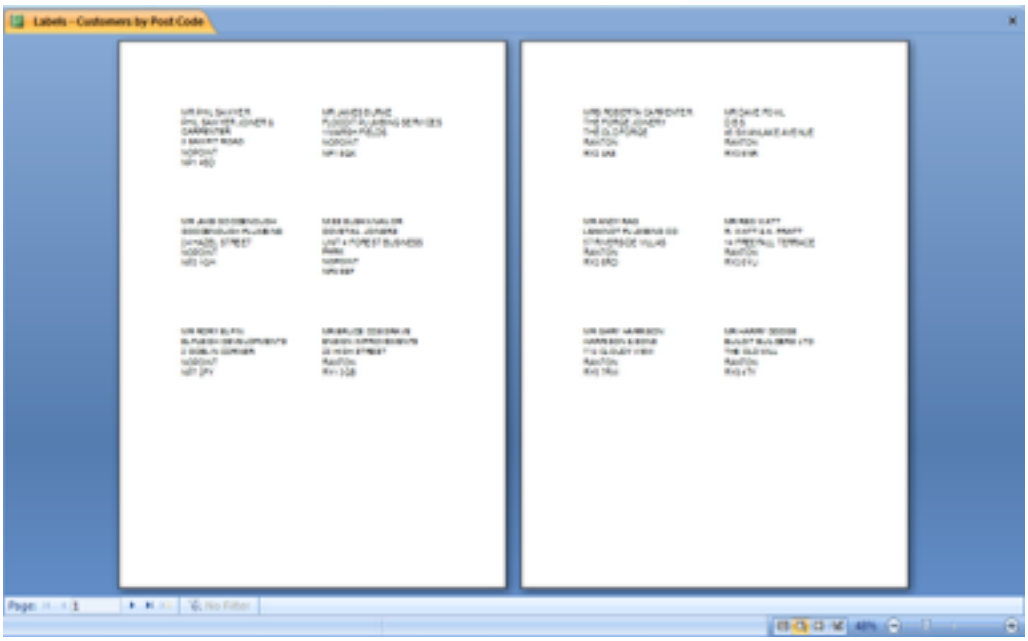
- highlight *ContactTitle* and click across to the label; the field name will be placed in braces (curly brackets { }) and the cursor will be placed immediately after the right brace; enter a space
- highlight *ContactFirstname* and click across to the label; enter a space
- highlight *ContactSurname* and click across to the label; press new line (carriage return/Enter)
- highlight *BusinessName* and click across to the label; press new line (carriage return/Enter)
- similarly, add *AddressLine1*, *AddressLine2* and *PostCode* to the label
- click **Next**



- you want to sort your labels; highlight *PostCode* and click across to the **Sort by** area
- click **Next**



- name the labels: Labels – Customers by Post Code
- accept the default setting to **See the labels as they will look printed**; click **Finish**



Note that you are in Print Preview – the view has been modified to show 2 pages. It should be plain that the labels are in Post Code sequence – this is a sensible sort sequence for mailing labels as the Post Office can give discounts for bulk mailings provided in this way. Have a look at other views.

The names (the first line in each label) are shown properly spaced although they are of variable length. This is because the Label Wizard automatically creates label controls preceded by **Trim** and enclosed in braces. **Trim()** is a function which strips leading and lagging spaces from **strings** (that is, groups of characters).

If you want to add a field not in the current table selected (Customers) you would have to use the Expression Builder via the Control Source facility in the Property window.

You may have to adjust the label width and/or font to get the labels to fit the page. You will be warned if the labels will not fit. You can also adjust the number of columns or modify the label spacing or size.

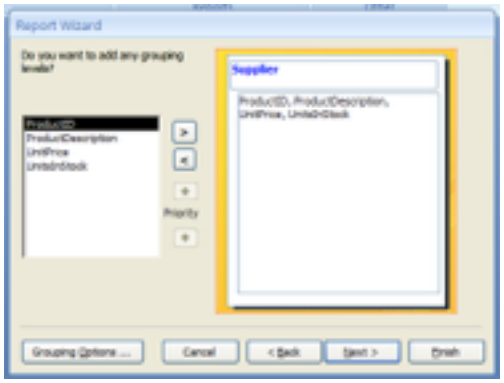
When you are satisfied with the result, close the Label report saving any required changes.

The other report wizards

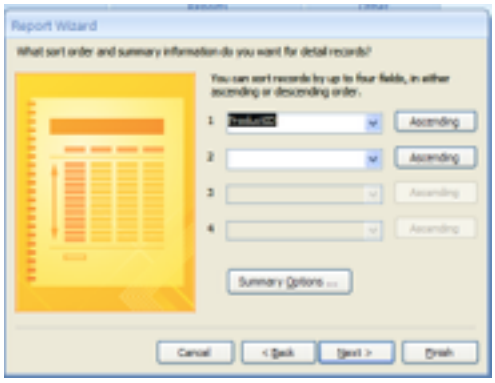
To have a look at the report wizards you will create a new report. In the ACME database window:

- ➢ open the ACME database (if not already open) and highlight the *Products* table
- ➢ to create a new report based on this table, click **Create**, then **Report Wizard**

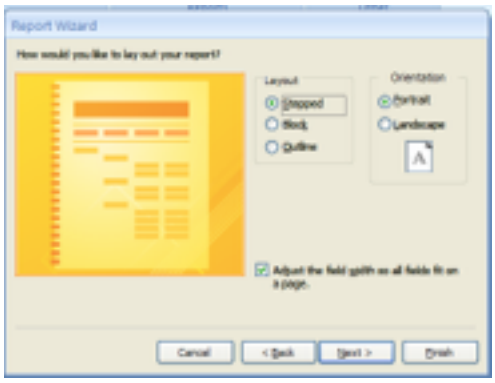
- ➢ to create the required report layout, highlight the field *Supplier* and click it across to **Selected Fields**; then click the double arrow to move the rest of the fields across to **Selected Fields** – this will give the required left-to-right column headings for the report
- ➢ click **Next**



- you want to group this report by *Supplier*, so in the 'grouping' dialogue box, select *Supplier* and click it across; then click **Next**



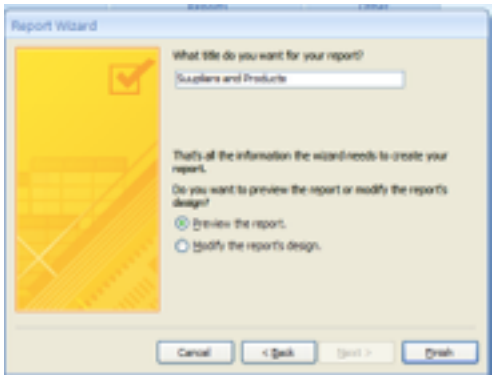
- you want to sort the report by *ProductID* – note that this will be within *Supplier* as the grouping (sort) field – select *ProductID* from the drop-down list (and accept the default sort sequence of Ascending)



- you are now offered a choice of layouts; experiment with the radio buttons to see how the various layouts will look

- select **Stepped** as the **Layout** with **Portrait** as the **Orientation** – click **Next**

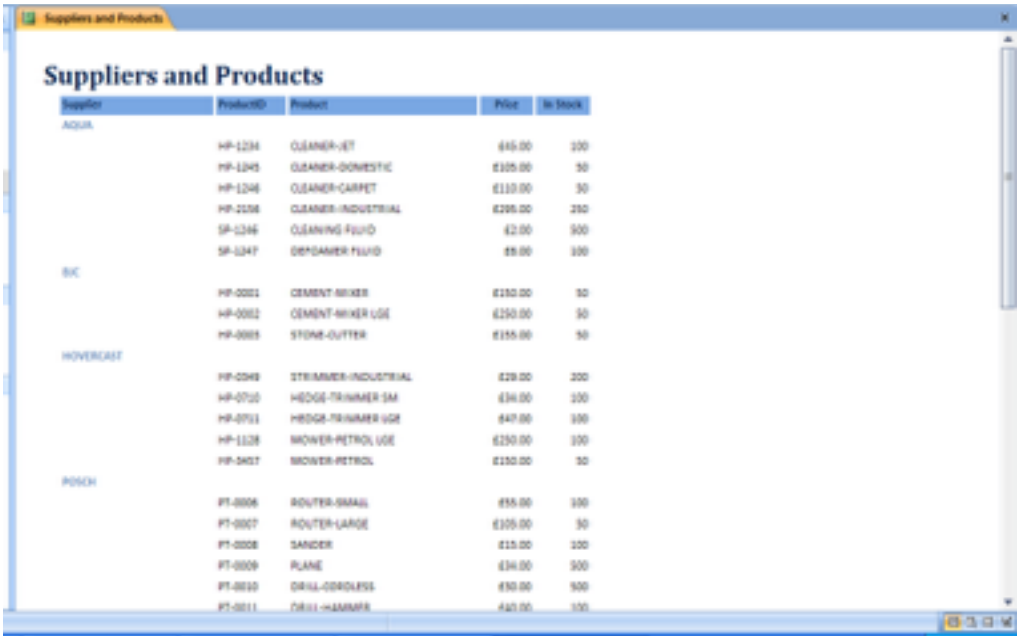
- accept the default style and click **Next**



- name the report: Suppliers and Products, and accept the default to **Preview the report**
- click **Next**

The report should look similar to the example below.

You can modify the report in **Layout View** or **Design View**. A quick adjustment to column widths in Layout View can produce a finished report as below.

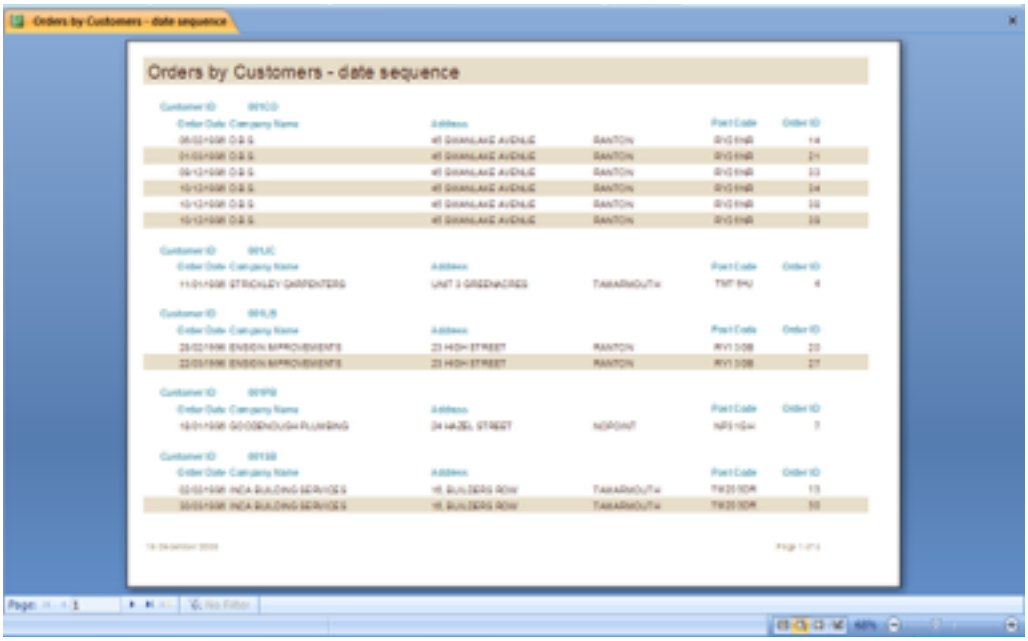


It makes sense to use a wizard or pre-formatted report unless the layout we need is so different it makes it easier to start from scratch. To do this, click **Create** then **Form**

Design View. You will get a ‘blank’ canvass to which you add required fields via the **Tools** and/or **Control** groups.

Exercise

Using the Report Wizard, and the Customers/Orders Query as the data source, produce a report to group the orders by *CustomerID* and sort the report by *OrderDate*. See the example below.



You can now quickly experiment with other styles. Notice that your choice of grouping affects the number of layout styles offered as some would not be sensible choices in these cases. To see what this means, try a report that does not have any grouping. Styles offered will be limited to Columnar, Tabular and Justified.

Checklist for Reports

Have you completed all of the following tasks?

- Understood what a report is for.
- Used the Tabular Report wizard to produce a draft price list.
- Carried out the edits to the price list.
- Produced a set of mailing labels.
- Experimented with other facilities of the Report wizard.
- Looked at the three different styles the wizard provides.

Expressions and Advanced Queries in MS Access 2010

You will now look at the constructs used in the more complicated queries, which can also be used for form or report controls.

In Access 2007, an expression is the equivalent of a formula in Office Excel 2007. An expression consists of a number of elements that you use alone or in combination to produce a result. Those elements can include:

- Identifiers – the names of table fields or controls on forms or reports, or the properties of those fields or controls
- Operators, such as **+** (plus) or **-** (minus)
- Functions, such as **SUM** or **AVG**, and the arguments that are used with them
- Constants – values that do not change – such as strings of text, or numbers that are not calculated by an expression

Expressions are combinations of OPERANDS and OPERATORS that evaluate or set required conditions. OPERANDS can be Functions, Column Variables or Constants. OPERATORS are the arithmetic or logical symbols. In Access, expressions can be coded in the conventional way, or the Expression Builder can be used.

For example, in the expression **[Unit Cost] < Target**, **[Unit Cost]** and **Target** are OPERANDS and **<** is an OPERATOR. This expression could be placed in a function as follows

if ([Unit Cost] < Target, “OK”, “Run Re-Cost Routine”)

Further, if this is preceded by the “=” operator, and placed in a CONTROL, it will evaluate to one of the conditional literal strings.

REFERENCE: See Access Help for complete details of operators. Note that various classes of operators are available.

Operator Classes

Arithmetic operators

You use the arithmetic operators to calculate a value from two or more numbers or to change the sign of a number from positive to negative or vice versa.

Operator	Purpose	Example
+	Sum two numbers.	[Subtotal]+[SalesTax]
-	Find the difference between two numbers or indicate the negative value of a number.	[Price]-[Discount]
*****	Multiply two numbers.	[Quantity]*[Price]
/	Divide the first number by the second number.	[Total]/[ItemCount]
****	Round both numbers to integers, divide the first number by the second number, and then truncate the result to an integer.	[Registered]\[Rooms]
Mod	Divide the first number by the second number, and then return only the remainder.	[Registered] Mod [Rooms]
^	Raise a number to the power of an exponent.	Number ^ Exponent

Comparison operators

You use the comparison operators to compare values and return a result that is True, False, or Null.

Operator	Purpose	Example
<	Returns True if the first value is less than the second value.	Value1 < Value2
<=	Returns True if the first value is less than or equal to the second value.	Value1 <= Value2
>	Returns True if the first value is greater than the second value.	Value1 > Value2
>=	Returns True if the first value is greater than or equal to the second value.	Value1 >= Value2
=	Returns True if the first value is equal to the second value.	Value1 = Value2
<>	Returns True if the first value is not equal to the second value.	Value1 <> Value2

Note In all cases, if either the first value or the second value is null, the result is then also null. Because null represents an unknown value, the result of any comparison with a null value is also unknown.

Logical operators

You use the logical operators to combine two Boolean values and return a true, false, or null result. Logical operators are also referred to as Boolean operators.

Operator	Purpose	Example
And	Returns True when Expr1 and Expr2 are true.	Expr1 And Expr2
Or	Returns True when either Expr1 or Expr2 is true.	Expr1 Or Expr2
Eqv	Returns True when both Expr1 and Expr2 are true, or when both Expr1 and Expr2 are false.	Expr1 Eqv Expr2
Not	Returns True when Expr is not true.	Not Expr
Xor	Returns True when either Expr1 is true or Expr2 is true, but not both.	Expr1 Xor Expr2

Concatenation operators

You use the concatenation operators to combine two text values into one.

Operator	Purpose	Example
&	Combines two strings to form one string.	string1 & string2
+	Combines two strings to form one string and propagates null values (if one value is Null, the entire expression evaluates to Null).	string1 + string2

Special operators

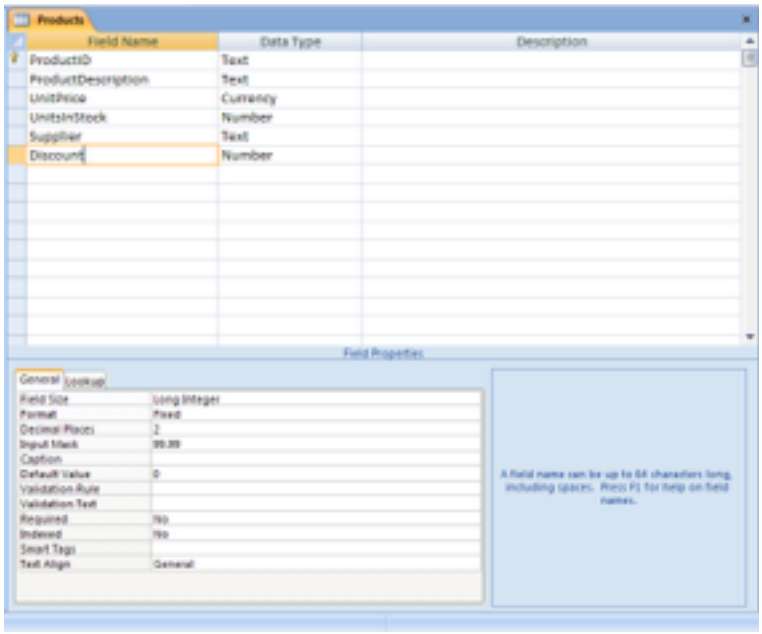
You use the special operators to return a True or False result as described in the following table.

Operator	Purpose	Example
Is Null or **Is Not Null**	Determines whether a value is Null or Not Null.	Field1 Is Not Null
Like "pattern"	Matches string values by using the wildcard operators **?** and *****.	Field1 Like "instruct*"
Between val1 And val2	Determines whether a numeric or date value is found within a range.	Field1 Between 1 And 10 - OR - Field1 Between #07-01-07# And #12-31-07#
In(val1,val2...)	Determines whether a value is found within a set of values.	Field1 In ("red","green","blue") - OR - Field1 In (1,5,7,9)

Entering Expressions in a multi-table query

For the new query we will need to amend the table definitions in the ACME database.

- ➢ open the *Products* table in **Design View**
- ➢ add a new field naming it *Discount* and define it as number of format 99.99 as per the picture below (do we need to allow for +ve and –ve?)



- ➢ switch to **Datasheet View** saving the changes to the *Products* table

We need to add some discount amounts for our new query.

- ➢ add the discount values as shown on the following page, and then close the table saving the changes

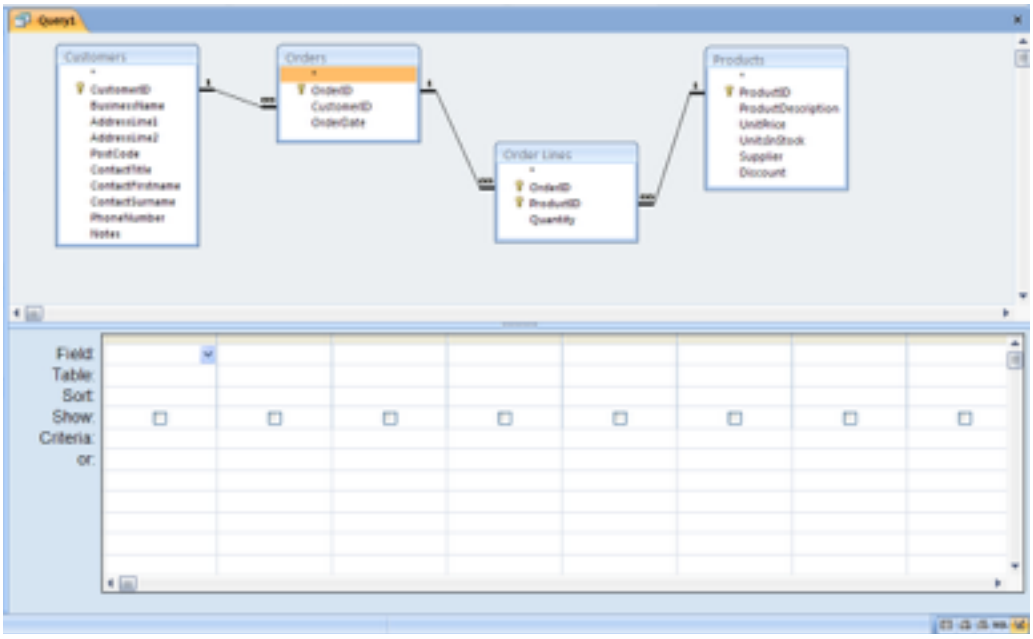
Products					
ProductID	Product	Price	In Stock	Supplier	Discount
HP-0001	CEMENT-MIXER	£150.00	50	BJC	5.00
HP-0002	CEMENT-MIXER LGE	£250.00	50	BJC	2.00
HP-0003	STONE-CUTTER	£155.00	50	BJC	3.00
HP-0349	STRIMMER-INDUSTRIAL	£29.00	200	HOVERCAST	4.00
HP-0710	HEDGE-TRIMMER SM	£34.00	100	HOVERCAST	4.00
HP-0711	HEDGE-TRIMMER LGE	£47.00	100	HOVERCAST	5.00
HP-1128	MOWER-PETROL LGE	£250.00	100	HOVERCAST	1.00
HP-1234	CLEANER-JET	£45.00	100	AQUA	2.00
HP-1245	CLEANER-DOMESTIC	£105.00	50	AQUA	1.00
HP-1246	CLEANER-CARPET	£110.00	50	AQUA	3.00
HP-2156	CLEANER-INDUSTRIAL	£295.00	250	AQUA	4.00
HP-3457	MOWER-PETROL	£150.00	50	HOVERCAST	3.00
HT-0001	SAW-TENON	£10.00	100	SYDNEY TOOLS	3.00
HT-0002	SAW-JIG	£5.50	100	SYDNEY TOOLS	2.00
HT-0003	SAW-RIP	£8.00	90	SYDNEY TOOLS	1.00
HT-0004	CHISEL-WOOD-1	£4.00	200	SYDNEY TOOLS	0.00
HT-0005	CHISEL-WOOD-2	£5.00	200	SYDNEY TOOLS	4.00
HT-0006	CHISEL-WOOD-3	£6.00	180	SYDNEY TOOLS	4.00
HT-0007	CHISEL SET	£12.00	100	SYDNEY TOOLS	5.00
PT-0001	DRILL	£25.00	100	WHITE & BECKER	3.00
PT-0002	DRILL-HAMMER	£30.00	100	WHITE & BECKER	2.00
PT-0003	SAW-CIRCULAR	£22.00	100	WHITE & BECKER	1.00
PT-0004	SAW-JIG	£20.00	90	WHITE & BECKER	1.00
PT-0005	SAW-BAND	£35.00	80	WHITE & BECKER	1.00
PT-0006	ROUTER-SMALL	£55.00	100	POSCH	0.00

Products					
ProductID	**Product**	**Price**	**In Stock**	**Supplier**	**Discount**
PT-0007	ROUTER-LARGE	£105.00	50	POSCH	0.00
PT-0008	SANDER	£15.00	100	POSCH	0.00
PT-0009	PLANE	£34.00	500	POSCH	0.00
PT-0010	DRILL-CORDLESS	£30.00	500	POSCH	2.00
PT-0011	DRILL-HAMMER	£40.00	100	POSCH	2.00
PT-0012	DRILL-PLAIN	£30.00	150	POSCH	3.00
PT-0713	LATHE-WOOD SM	£89.00	50	WHITE & BECKER	4.00
PT-0714	LATHE-WOOD LGE	£150.00	20	WHITE & BECKER	5.00
PT-0715	SCREWDRIVER-ELECTRIC	£15.00	200	POSCH	4.00
SP-1110	SAWBLADE-JIG	£5.00	300	UNIBITS	3.00
SP-1111	SAWBLADE-CIRCULAR	£15.00	150	UNIBITS	6.00
SP-1112	DRILL BITS-WOOD	£10.00	100	WHITE & BECKER	5.00
SP-1113	DRILL BITS-MASONRY	£10.00	150	WHITE & BECKER	4.00
SP-1114	DRILL BITS-WOOD	£5.00	250	UNIBITS	3.00
SP-1115	DRILL BITS-MASONRY	£5.00	250	UNIBITS	2.00
SP-1246	CLEANING FLUID	£2.00	500	AQUA	2.00
SP-1247	DEFOAMER FLUID	£6.00	100	AQUA	2.00
SP-1416	FACEMASK	£4.00	1000	UNIBITS	2.00
SP-1516	FACEMASK-RIGID	£7.50	500	UNIBITS	2.00

Adding a Query to the Related Tables

- click **Create** then **Query Design** to open a new Query
- highlight all the tables on the Show Table list box (you can do this by keeping Ctrl pressed and highlighting each table in turn), then click the Add button

- close the **Field List** and rearrange and re-size the tables so that you can see the layout clearly



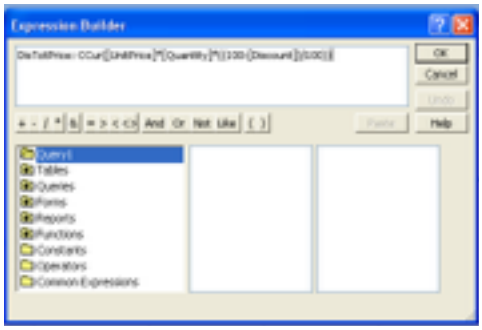
- add Order Lines(OrderID) to the query
- add Customers(BusinessName, PostCode and AddressLine2) to the Query –as before you can double click the field name in the table, drag and drop or select from a drop down list in the query column header in order to add a field name to the QBE grid

Note - The above form of defining required columns is used for accuracy and to save space. As you probably guessed, the name outside the left parenthesis is the table name, and the names separated by commas are the field names.

- click the **Totals** icon in the **Show/Hide** group to add the Total row to the Query Design grid - note that the default value **Group By** is added to the Total cell for each field on the query

- you are going to enter an expression in the next free column in the QBE grid - open **Expression Builder** by right clicking the **Field** cell and then click **Build...** from the drop down menu

- type: DisTotPrice: CCur([UnitPrice]*[Quantity]*((100-[Discount])/100)) into Expression Builder and click **OK** to enter it into the **Field** cell (what does this expression calculate?)

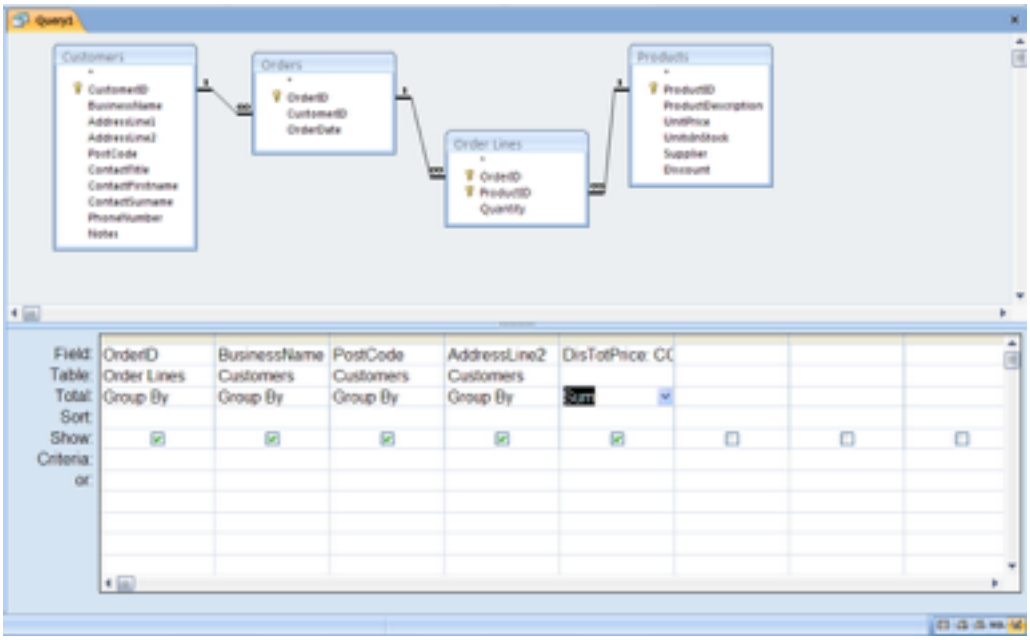


(When you have completed this part of the practical, look up the **CCur** function in Help and find out what it does.)

- click in the **Total** cell of the new column and select **Sum** from the drop-down list of aggregate functions - this will give you the total net amount (i.e. factoring in the discount) for the Order Lines of each Order

- make sure you leave the **Table** cell of the new column blank, as this column is a virtual object, i.e. it exists only when the query runs, and that all the columns you wish to display are checked off in the **Show** row

Your query should like the picture below.



➢ run the query either by clicking the **Datasheet View** icon or the **Run** icon – the result should be like the one below

Query1				
OrderID	Company	Post Code	AddressLine2	DisTotPrice
1	DOVETAIL JOINERS	NP5 9EF	NOPOINT	£48.50
2	ROB & BOB HANDYMEN	TM1 5CV	TAMARMOUTH	£171.90
3	J. LUCK (PLUMBER)	TM7 3FG	TAMARMOUTH	£14.40
4	STRICKLEY CARPENTERS	TM7 5HJ	TAMARMOUTH	£5.39
5	THE FORGE JOINERY	RY2 4AS	RANTON	£142.50
6	HARRISON & SONS	RY5 7RW	RANTON	£342.33
7	GOODENOUGH PLUMBING	NP3 1GH	NOPOINT	£11.40
8	R. WATT & A. PRATT	RY2 6YU	RANTON	£245.00
9	PHIL SAWYER JOINER & CARPENTER	NP1 4SD	NOPOINT	£96.84
10	BETA CONTRACTORS	TM5 4BY	TAMARMOUTH	£88.75
11	BUILDIT BUILDERS LTD	RY8 4TY	RANTON	£154.83
12	LEAKNOT PLUMBING CO	RY2 6RD	RANTON	£24.25
13	INCA BUILDING SERVICES	TM20 3DR	TAMARMOUTH	£55.00
14	D.B.S.	RY2 6NR	RANTON	£145.50

Query1				
OrderID	Company	Post Code	AddressLine2	DisTotPrice
15	FLOODIT PLUMBING SERVICES	NP1 8QK	NOPOINT	£67.70
16	B.J.SMALLBROTHER	TM10 6YZ	TAMARMOUTH	£19.60
17	ELFLEIGH DEVELOPMENTS	NP7 2FY	NOPOINT	£150.35
18	G SMITH BUILDING CONTRACTORS	TM3 7RT	TAMARMOUTH	£60.48
19	JONES BROTHERS	TM13 4FY	TAMARMOUTH	£22.92
20	ENSIGN IMPROVEMENTS	RY1 3GB	RANTON	£128.75
21	D.B.S.	RY2 6NR	RANTON	£7.92
22	LEAKNOT PLUMBING CO	RY2 6RD	RANTON	£37.40
23	THE FORGE JOINERY	RY2 4AS	RANTON	£15.00
24	HARRISON & SONS	RY5 7RW	RANTON	£69.30
25	ROB & BOB HANDYMEN	TM1 5CV	TAMARMOUTH	£19.80
26	PHIL SAWYER JOINER & CARPENTER	NP1 4SD	NOPOINT	£26.40
27	ENSIGN IMPROVEMENTS	RY1 3GB	RANTON	£228.75
28	LEAKNOT PLUMBING CO	RY2 6RD	RANTON	£68.60
29	HARRISON & SONS	RY5 7RW	RANTON	£607.60
30	INCA BUILDING SERVICES	TM20 3DR	TAMARMOUTH	£29.40
37	J. LUCK (PLUMBER)	TM7 3FG	TAMARMOUTH	£14.10
39	D.B.S.	RY2 6NR	RANTON	£14.10

? Do you think ALL orders for the product line will receive a discount automatically? How might ACME apply discounts? Can you work out what method you could use in Access to effect this Business Rule?

➢ when you are satisfied with the query, **Close** it, saving the changes and naming the query: Value of Orders after Discounts

Exercise

Create a form based on this query, in Datasheet format, but with better formatting and headings than the above.

Exercise

Set up **a multi-table Select Query** combining the *Customers*, *Products*, *Orders* and *Order Lines* tables. The query should list those whose Customers.AddressLine2 is Ranton or Tamarmouth, whose PostCode begins with TM1 or RY2, and whose gross order amount is between two values to be entered at run time as parameters. Total the gross amount column in the query.
(Some helpful pictures are in the appendix at the end of this practical.)

Exercise

Set up a basic **multi-table Select Query** combining the *Customers*, *Products*, *Orders* and *Order Lines* tables. The query should contain:

Customers(BusinessName, AddressLine2, PostCode)
Order Lines(OrderID, ProductID)

Save the query as: Products by Order and Customer. You will use this later on.

Using the Expression Builder in creating Queries

- ➢ open the “Products by Order and Customer” query in Design View
- ➢ click in the criteria cell of the *BusinessName* column
- ➢ click the **Builder** icon (in the **Query Setup** group) to display the Expression Builder window

You will notice that the lower left panel lists the objects available. Included is the current query. The lower middle panel shows the contents of the selected object, in this case a list of the columns used for the query. The lower right hand panel shows the value you wish to apply to the selected item.

- ➢ select the current query (if not already highlighted), and select *BusinessName* (if not already highlighted)
- ➢ click the **Like** button (in the little row of buttons in the middle of Expression Builder)
- ➢ type “*TS” (this will select customers whose names end in “TS”) and click **OK**
- ➢ run the query – you should get the following result

Products by Order and Customer - TS				
Company	**AddressLine 2**	**Post Code**	**OrderI D**	**ProductI D**
ENSIGN IMPROVEMENTS	RANTON	RY1 3GB	20	HP1246
ENSIGN IMPROVEMENTS	RANTON	RY1 3GB	20	SP1516
ENSIGN IMPROVEMENTS	RANTON	RY1 3GB	27	HP0003

Products by Order and Customer - TS				
Company	AddressLine 2	Post Code	OrderI D	ProductI D
ENSIGN IMPROVEMENTS	RANTON	RY1 3GB	27	PT0011
ELFLEIGH DEVELOPMENTS	NOPOINT	NP7 2FY	17	HP0003

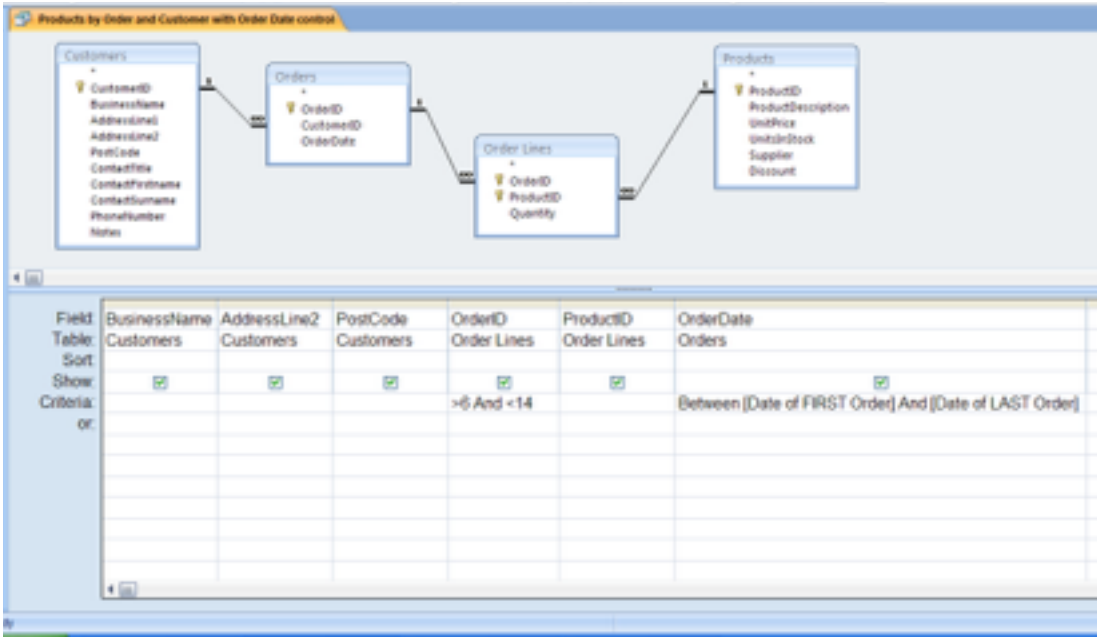
- save the query with the name: Products by Order and Customer - TS

Now try some examples for yourself.

- return to Design View and delete the criteria expression we just created
- select any column in the query grid and using the Expression Builder as above, create expressions of your own, test running each 'new' query
- when you have finished, discard any changes

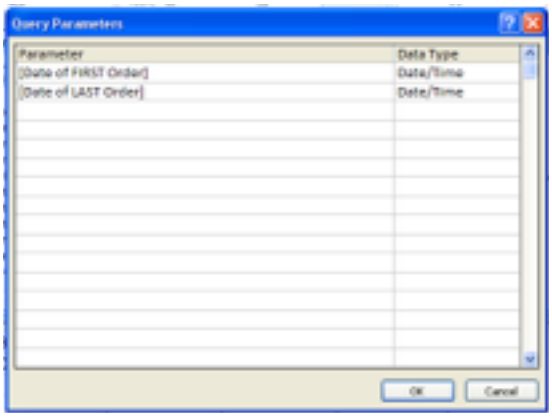
Exercise

Add *OrderDate* to the "Products by Order and Customer" query. Set two date parameters "Date of FIRST Order" and "Date of LAST Order". The criteria should search for records which contain dates between the First Order Date and the Last Order Date. In the same query, set the *OrderID* returned to be greater than 6 and less than 14.



Run the query and enter parameters of your choosing, trying several combinations so you can see for yourself the validity of the returned dynaset of rows (query records). Set the Parameters so their data type is correct. You can do this by clicking the **Query**

Parameters button (in the **Show/Hide** group). Setting the data type will ensure Access will check the data entered is of the correct type. (If you are unsure how to do this, check the "Any Trade Mail List" query for setting query parameters and data types.)



Note that YOU have to enter the correct parameter name – if you make a mistake, Access will run that parameter AS WELL as the ones you have defined in your Query – i.e. Access does not check this or enter the parameters you have entered as default.

Cross Table (Crosstab) Queries

A crosstab query is a spreadsheet-like summary of the things specified by the row and column headers created from your table. In this specialised type of total query, the Total row in the QBE pane is always active and cannot be toggled off in a crosstab query. In addition, the Total row of the QBE pane specifies a Group By option for both the row and the column headings. Like other total queries the Group By option specifies the row headings for the query datasheet and is based on the contents of the field. However, unlike other total queries the crosstab query also obtains its column headings from values in a field rather than from table field names.

For example, a database containing accounting information may need to look at sales in a periodic fashion – say by product line (as the row heading) within a month (which would be a column heading). The value shown in the cells could be total sales in pounds sterling for the month – see the example below.

Product	JAN	FEB	MAR	APR
SONY TV model 3	£3,435	£2,389	£3,345	£7,886
CASIO PALM PC XYZ	£1,256	£1,289	£2,335	£1,266

In the first example, we will create a Cross Table (Crosstab) Query which returns an output similar to the above. *ProductDescription* is the Row Heading, *OrderDate* grouped into Month of Order is the Column Heading, and the Sum of *Quantity* * *UnitPrice* provides the cell Values. You will need to identify the tables required for these fields to add them to the Crosstab query. (When you have added the tables to the query, you can then select the type of query.)

- enter the row heading (which is straightforward to do)
- the column heading can be returned by using the **Month()** function on *OrderDate*, but in Access this returns only a numeric output
- the value is an expression which is two table columns multiplied together – put this together using the Expression Builder, either by identifying the tables and columns or by typing in the required expression (do not forget that you want the **Sum** of the product of *Quantity* and *UnitPrice*)
- run the query – the output should be like the following (queries run very quickly on the small amount of data we have, and if there is a mistake, you will get valuable feedback which may enable you to correct any errors immediately)

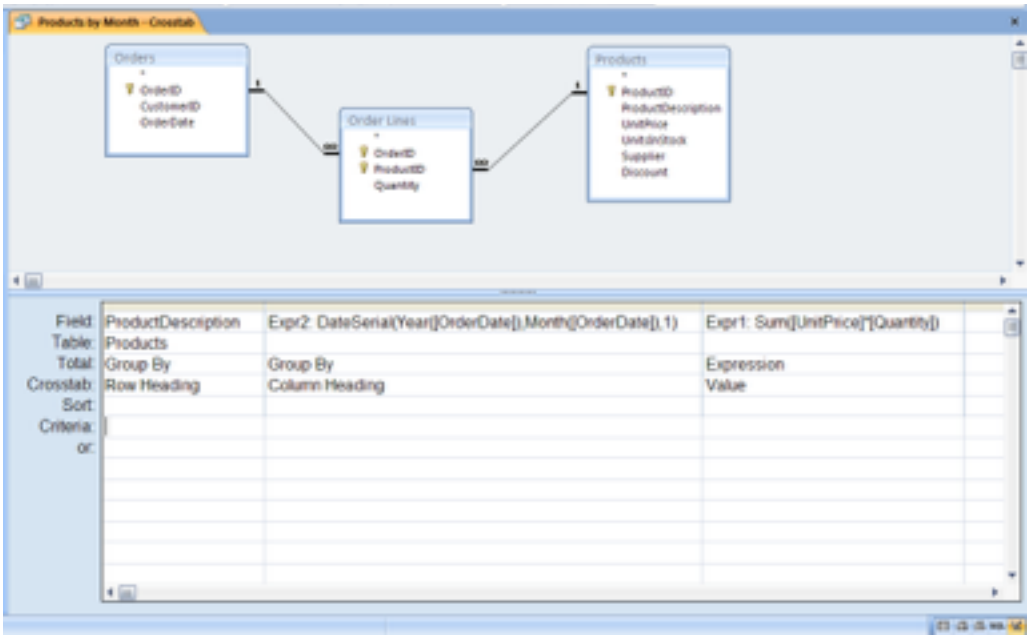
Product	1	2	3	12
CEMENT-MIXER	£150.00			
CEMENT-MIXER LGE	£250.00		£500.00	
CHISEL SET	£24.00		£12.00	
CHISEL-WOOD-1			£8.00	
CHISEL-WOOD-2	£5.00			
CLEANER-CARPET		£110.00		
CLEANER-DOMESTIC	£105.00			
CLEANER-INDUSTRIA	£295.00			
CLEANER-JET	£45.00			
CLEANING FLUID		£8.00		
DEFOAMER FLUID		£12.00		
DRILL	£25.00			
DRILL BITS-MASONR	£20.00			
DRILL BITS-WOOD		£30.00		
DRILL-CORDLESS			£90.00	
DRILL-HAMMER	£30.00	£40.00	£240.00	
DRILL-PLAIN	£30.00			
FACEMASK	£16.00			
FACEMASK-RIGID		£22.50		
HEDGE-TRIMMER LG	£47.00			
HEDGE-TRIMMER SM		£34.00		
LATHE-WOOD LGE	£150.00			
LATHE-WOOD SM	£89.00			
MOWER-PETROL		£150.00		
PLANE	£34.00			
ROUTER-SMALL		£55.00		

Record: 1 of 37 | No Filter | Search

➢ add another order for 4 CHISEL SETS for 3rd January 1997 for Customer 004JC – which tables do you need to modify with this new data?

Your user needs a string output for the column heading (month).

? which DATE functions will you use to do this (look at the next screen shot to see)



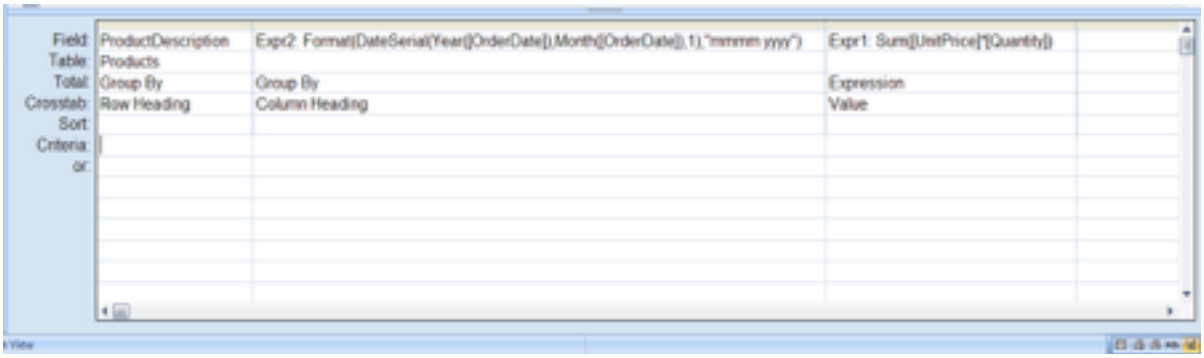
➢ modify your Crosstab query accordingly – look up the Functions used in Help

- run your modified query – note how Access deals with two values for the **Year** argument in the **DateSerial** function

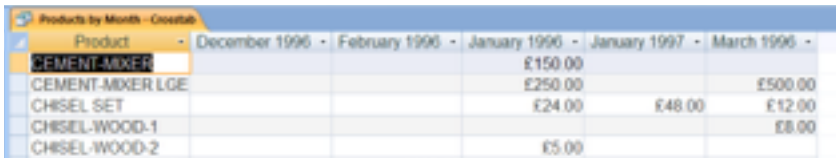
Finally, you need to modify the **DateSerial** function to give you a string value for the name of the month.

- to do this, you have to use the **Format** function on the **DateSerial** function

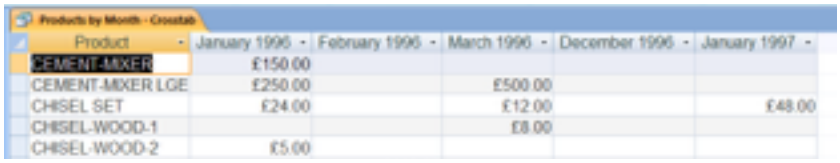
- look up the use of this function and see if you can work out how to use it here to produce column headings of the form "January 1996" (see the next screen shot if you need assistance)



- when you run the query the columns are now in alphabetical/numeric order rather than chronological; this is because the output is a string and is ordered ascending this way by default



- to deal with this and restore the column headings to the required sequence, you can edit the properties of the query and enter the column headings in the relevant property space PRECISELY AS FORMATTED as literal strings i.e. "January 1996", "February 1996" and so on – if you make any mistakes Access will not show that column (i.e. the Property entry will now take preference if there is an entry there)



Products by Month - Crosstab

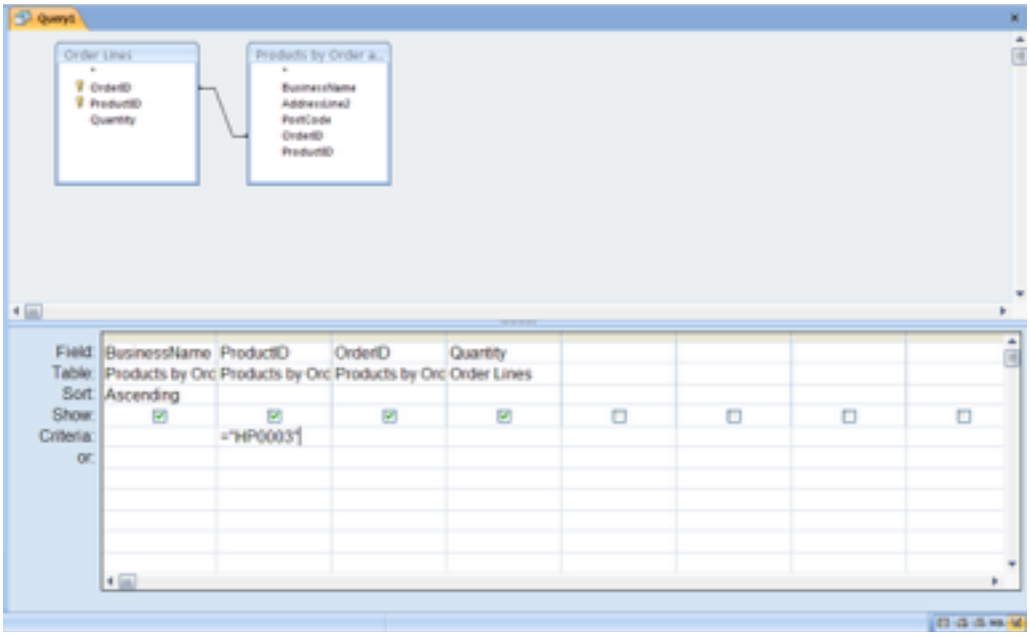
Product	January 1996	February 1996	March 1996	December 1996	January 1997
CEMENT-MIXER	£150.00				
CEMENT-MIXER LGE	£250.00		£500.00		
CHISEL SET	£24.00		£12.00		£48.00
CHISEL-WOOD-1			£8.00		
CHISEL-WOOD-2	£5.00				

Nested Queries

A query can be based on an existing query. Queries as well as tables can be added to a new query window, and then required columns can then be added to the query panel (the QBE grid).

Assume we want to find all customers with Company name (i.e. *BusinessName*) ending in "TS" which have orders for the product "HP003", and further, that we need to know the quantity of those orders. Say, for example, that the orders we are after have a certain name or name content, and that we need to know the product and quantity information for marketing or other business reasons. You already have the first part of this query as it was constructed earlier.

- ➢ click **Create**, then **Query Design**
- ➢ note that if you open the **Both** window in the **Show Table** dialogue box, you can see queries and tables – this is useful for what you need to do here – add the query "Products by Order and Customer - TS" and the Order Lines table to the query window and close the Show Table box
- ➢ in order to get the query to work correctly, the query and the table must be 'joined'; you achieve this by dragging and dropping OrderID from the table to the query (or vice versa)
- ➢ add *BusinessName*, *ProductID* and *OrderID* from the query and *Quantity* from the table
- ➢ enter the HP0003 criteria in the relevant cell
- ➢ enter Ascending as the sort criterion for *BusinessName*



➢ run the query – see below for the result you should have – when you are satisfied, close and save the query as: Quantity of HP0003 for TS companies

Query1			
Company	ProductID	OrderID	Quantity
ELFLEIGH DEVELOPMENTS	HP0003	17	1
ENSIGN IMPROVEMENTS	HP0003	27	2
ENSIGN IMPROVEMENTS	HP0003	27	1

Query1			
Company	ProductID	OrderID	Quantity
ELFLEIGH DEVELOPMENTS	HP0003	17	1
ENSIGN IMPROVEMENTS	HP0003	27	2
ENSIGN IMPROVEMENTS	HP0003	27	1

➢ click **Create**, then **Query Design**

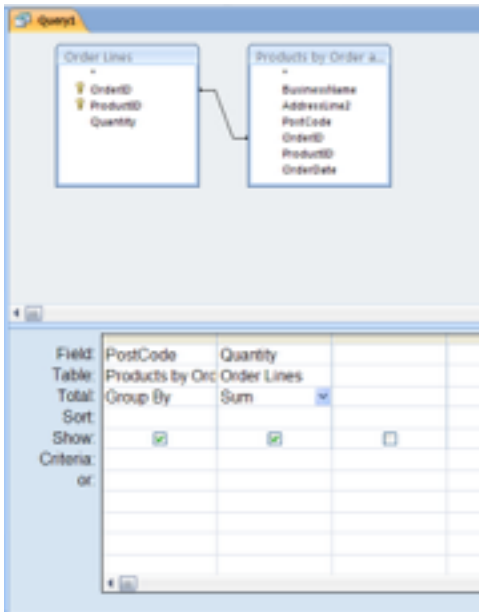
➢ add the query "Products by Order and Customer with Order Date control" and the Order Lines table to the query window

➢ 'join' the two objects on *OrderID*

➢ add *PostCode* and *Quantity* to the QBE grid

➢ click the **Totals** button in the **Show/Hide** group – this puts **Group By** in the Total row for both fields

➢ change the criterion on Quantity to give the sum of products sold to a particular postcode (this could be useful for delivery scheduling, for example)



➢ a result is shown below for all orders between 1-1-96 and 31-12-96

Query1	
Post Code	**SumOfQuantity**
NP1 4SD	4
NP3 1GH	1
RY2 6RD	1
RY2 6YU	1
RY8 4TY	9
TM20 3DR	1
TM5 4BY	4

Note that Access can FILTER Record Sets. For example, from a form or table based on a query you can effectively run a subquery using the Filter option (in the **Sort & Filter** group in the **Home** ribbon).

Summary of All Types of Queries in Access

(you may need to do some further reading about the different types of queries)

- Select: Explained in the section about queries.
- Update: Use an update query to delete individual field values from a table. An update query let you delete values by updating the existing values to either a null value (that is, no data) or a zero-length string (a pair of double quotation marks with no space between them).
- Delete: Use a delete query to remove entire records (rows) from a table or from two related tables, in one operation. Delete queries remove all the data in each field, including the key value that makes a record unique.

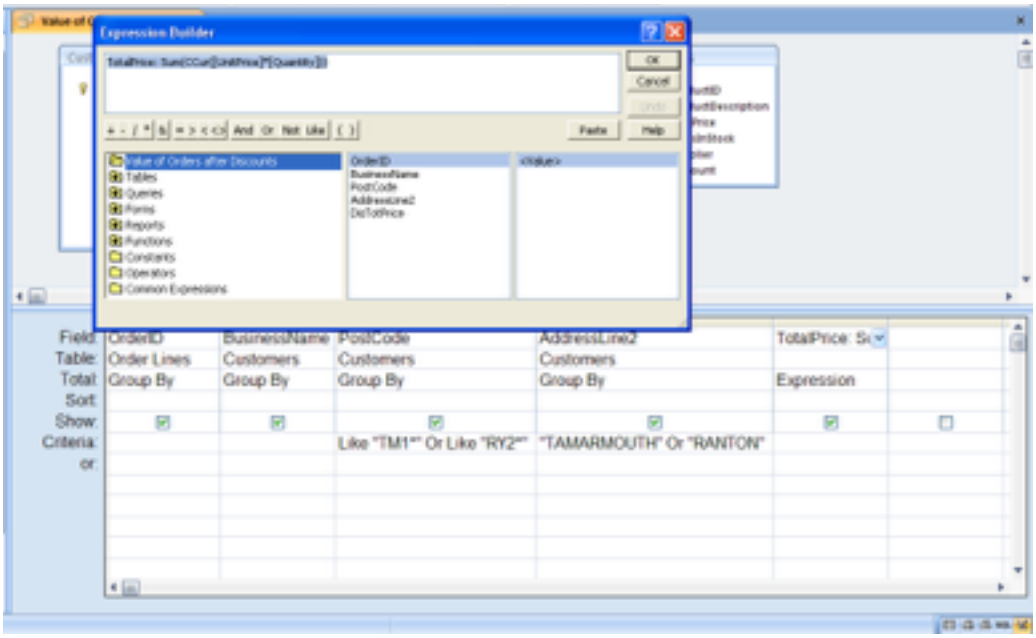
More information can be found on http://office.microsoft.com/en-us/access-help/delete-data-from-an-access-database-by-using-a-query-HA010342091.aspx

Checklist for Expressions and Advanced Queries

- The Expression Builder helps you to create the statements you need to produce the desired result. You can type in expressions without using Builder; however, it is easier to check syntax using it. Using Builder does not guarantee success though!
- You can create queries that use more than one table.
- Calculated data from a query may be useful in a form or report.
- Decide what you want to achieve before embarking on a complex query. Careful planning will save time.
- Management will often want summaries of data by product and period of time, for example. This can be achieved using a Crosstab query. Again, if you think carefully about what you want, it is not too difficult to do.
- We can produce a query of a query and so on nested as far as we need to in order to produce the information we want. It is vitally important that you understand precisely what you want to produce using a complex query. The production of the query in Access is then much easier!

Appendix – pictures for multi-table Select Query

This is a combination of the *Customers*, *Products*, *Orders* and *Order Lines* tables. The query should list those whose Customers.AddressLine2 is Ranton or Tamarmouth, whose PostCode begins with TM1 or RY2, and whose gross order amount is between two values to be entered at run time as parameters. Total the gross amount column in the query.



This shows the fields and criteria plus the expression to generate the total sum of orders.

Value of Orders for TM1 and RY2				
OrderID	Company	Post Code	AddressLine2	TotalPrice
2	ROB & BOB HANDYMEN	TM1 5CV	TAMARMOUTH	£180.00
5	THE FORGE JOINERY	RY2 4AS	RANTON	£150.00
8	R. WATT & A. PRATT	RY2 6YU	RANTON	£250.00
12	LEAKNOT PLUMBING CO	RY2 6RD	RANTON	£25.00
14	D.B.S.	RY2 6NR	RANTON	£150.00
16	B.J.SMALLBROTHER	TM10 6YZ	TAMARMOUTH	£20.00
19	JONES BROTHERS	TM13 4FY	TAMARMOUTH	£23.00
21	D.B.S.	RY2 6NR	RANTON	£8.00
22	LEAKNOT PLUMBING CO	RY2 6RD	RANTON	£38.00
23	THE FORGE JOINERY	RY2 4AS	RANTON	£15.00

Value of Orders for TM1 and RY2				
OrderID	Company	Post Code	AddressLine2	TotalPrice
25	ROB & BOB HANDYMEN	TM1 5CV	TAMARMOUTH	£20.00
28	LEAKNOT PLUMBING CO	RY2 6RD	RANTON	£70.00
39	D.B.S.	RY2 6NR	RANTON	£15.00

This is the intermediate result. Now we need to add the criteria to allow the entry of the order amount range.

Value of Orders for TM1 and RY2				
OrderID	Company	Post Code	AddressLine2	TotalPrice
12	LEAKNOT PLUMBING CO	RY2 6RD	RANTON	£25.00
22	LEAKNOT PLUMBING CO	RY2 6RD	RANTON	£38.00
28	LEAKNOT PLUMBING CO	RY2 6RD	RANTON	£70.00

This is the result of requesting orders between £25 and £100 (note that Between includes values on upper and lower limits in results). We can check the accuracy against the 'whole' result above.

Option Groups, Command Buttons and Splash Screens in Access 2010

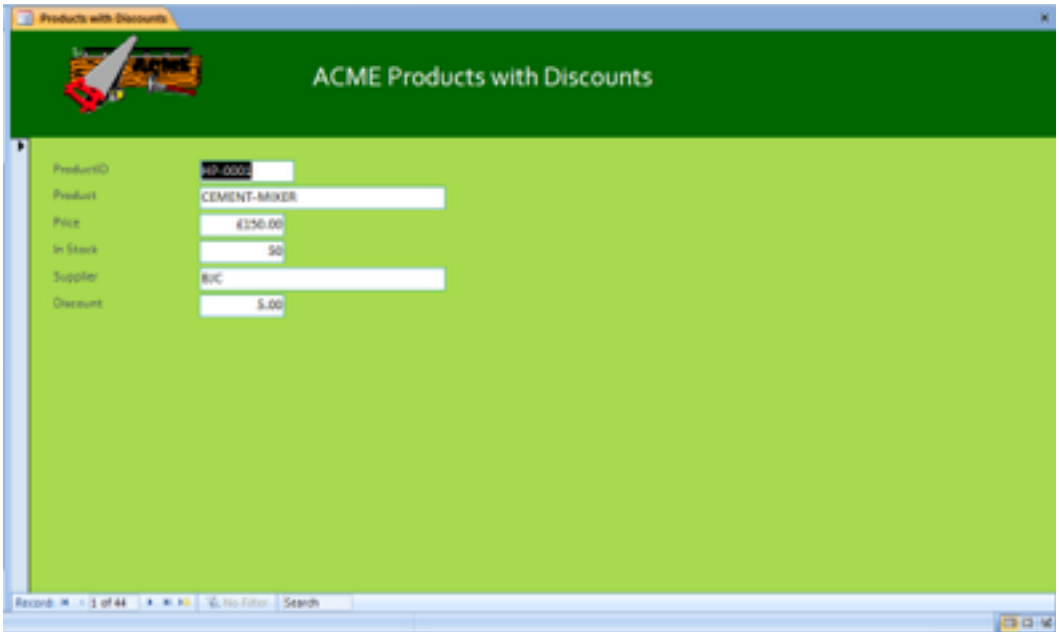
Option Groups

- If one of the column-data items has a limited number of values (and is always likely to be so) it makes sense to set up an Option Group in the data entry form.
- Providing a set of allowable choices greatly cuts down on mistakes in data entry, and can also speed up data entry. Thus there are good business reasons to implement an option group if possible.
- Assume that the discounts we give on our products will be in whole number percentages from 0 (no discount) to 6.
- We will create a set of option buttons to enter/edit these values.

To start, you need to create a straightforward form based on the *Products* table.

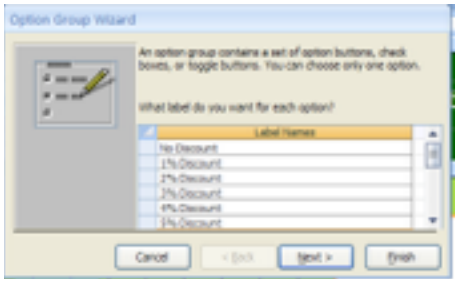
➢ open the ACME database

➢ create a basic form for the *Products* table using the Form Wizard (review Practical 10 if necessary) – see the picture below – use all fields in the given sequence with a columnar layout; name the form: Products with Discounts



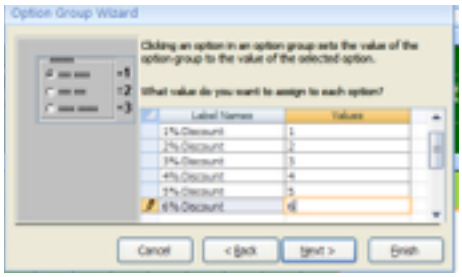
Make sure the **Use Control Wizards** button (in the **Controls** group) is activated. You can toggle it off and on to check. With the Wizard on, the next operation will automatically open.

- click the **Option Group** icon; as you drag the pointer on to the form it changes to a small cross-hair with an option group icon
- click in a space to the right of the existing fields to drop the option group box – note that the area of the group will increase to accommodate the controls
- enter the option item labels to be “No Discount”, “1% Discount” and so on to “6% Discount” (as in the picture below) and click **Next**

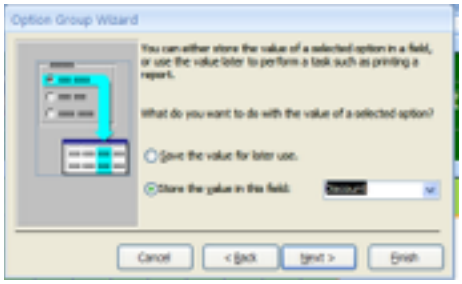


- choose a default that will apply if no option is specifically chosen – you can choose the most common discount, but it is recommended that you choose the “safe” option of No Discount (this may be preferable, as if you offer no, or low discount you can increase it if necessary – to reduce the discount offered will not be very popular, and may lose customers); then click **Next**

- you can now assign the values to each option button – set these values at 0 for the No Discount button and so on to 6 for the 6% Discount button (as below); then click **Next**

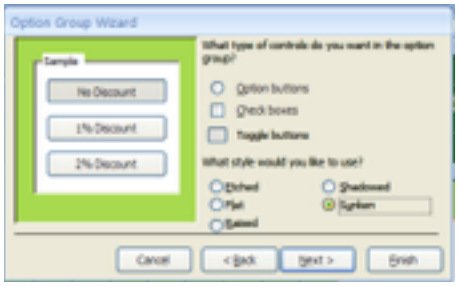


- next, you are invited to store the value for later use (e.g. in a control, macro or Basic module) or to write it back to an existing column-data item



- we will take the latter option, and update the *Discount* column with this value

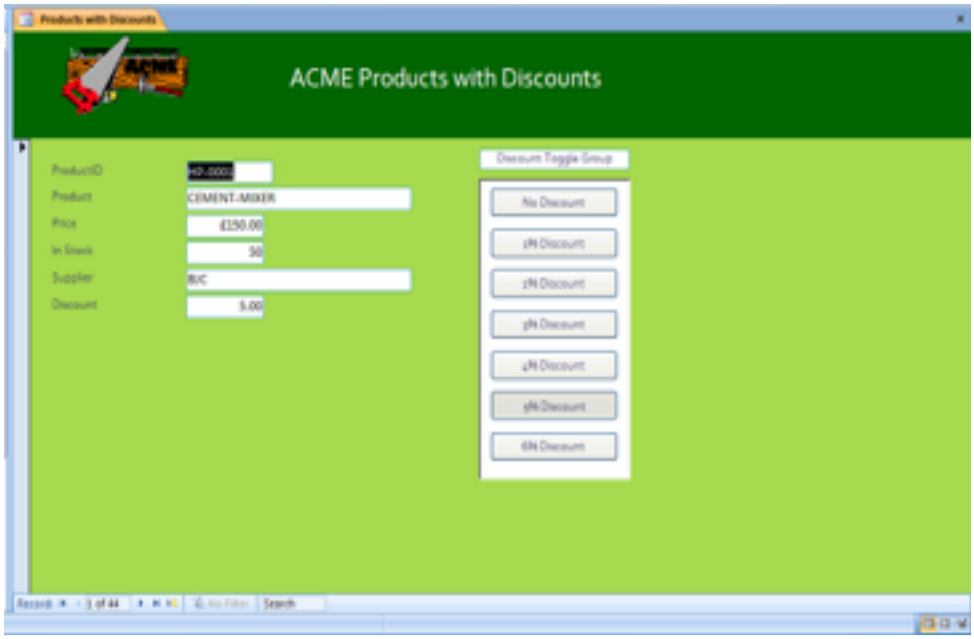
➢ at this point, you need to select the kind of control to apply to the option group



➢ choose Toggle buttons and a style of Sunken – you should see the default “No Discount” highlighted as per the example above; click **Next**

➢ as a caption (label) for the option group enter: Discount Toggle Group

➢ click **Finish** – the completed form should look something like the example below (in Form View)



Note that as you move through the records, the toggle button that corresponds to the given discount is highlighted.

➢ make any further adjustments you want, and then close the form saving your changes

➢ Command Buttons

Access can run commonly-used routines from forms with COMMAND BUTTONS.

When clicked, these buttons run MACROS which are sets of instructions to do particular operations (with or without run-time parameters). Access has many of the most useful Macros already prepared to run with Command Buttons and we are going to see these now.

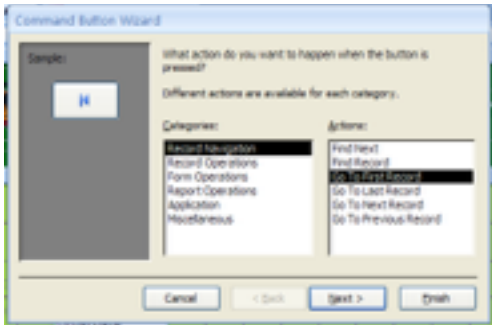
- ➢ in the Database window, highlight the *Orders* table, click **Create**, **More Forms**, then **Form Wizard**

- ➢ you can create a basic form using all the fields by selecting them all across in the first dialogue box and clicking **Finish** – by default, the form will be named Orders

Make sure the **Use Control Wizards** icon is highlighted.

- ➢ click the **Button** icon (in **Controls**) so it is highlighted and move the pointer into the Form Detail; draw a button (from the cross hair) about 1 cm square to the right of the fields on the form – a dialogue box will open up

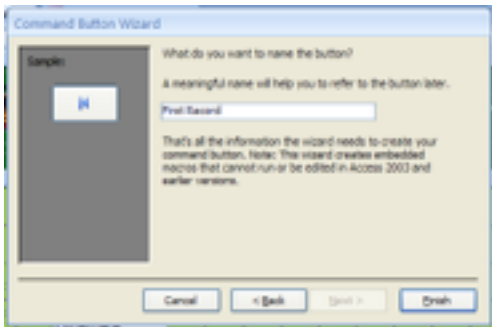
- ➢ choose **Record Navigation** as the Category, then **Go To First Record** as the Action (this button will have code associated with it, i.e. a Macro, which will move the form data display to the first record in the dynaset when activated)



Now you have to decide whether you want a picture or a text label for the button, to indicate what will happen when it is clicked.

- ➢ in this case, select Picture and choose one from the samples offered (you can look at the options in turn and choose one – this is a non-functional matter in that it does not affect in WHAT will happen, but such choices are often important for the user's benefit)

- name the button – since this is the name that will be given to the Macro procedure underlying it, it is good to make it relevant – later modules and macros may be added so the naming of buttons should be systematic

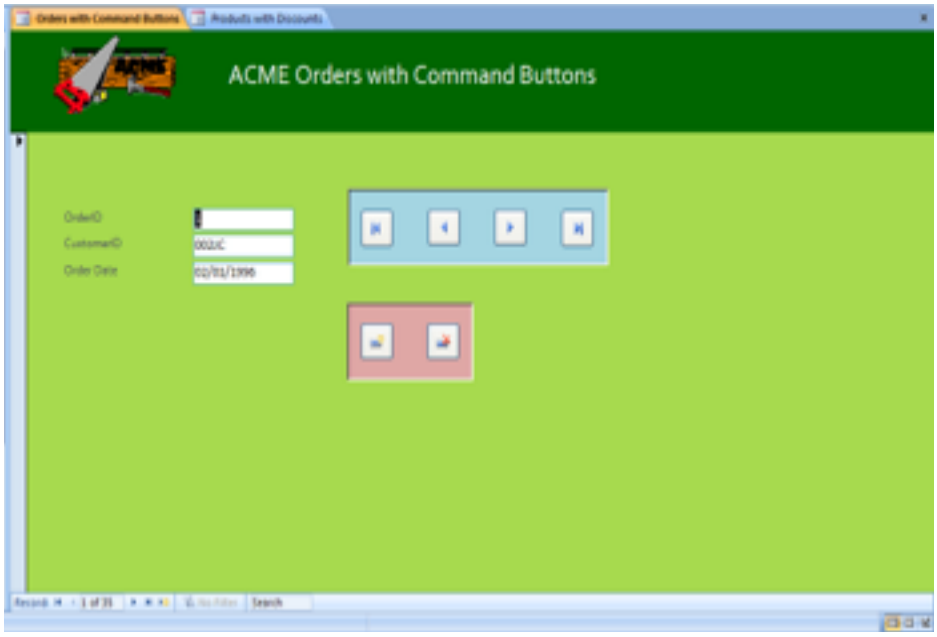


- click **Finish**; the button is now on the form – you can check out its operation in Form View

- similarly, add buttons for Next Record, Previous Record, Last Record, Delete Record and Add Record – you will need to select **Record Operations** in the dialogue box to create buttons for these last two

- group them by bordering the Navigation buttons (First Record...) with a rectangle, and the Procedural ones (Add/Delete Record) with another – note that these groupings are part of the non-functional design, they make no difference to what the buttons do

Depending on how you apply the rectangles and fill them, you may have to use **Arrange** and **Bring to Front** to show the buttons. You can also use the Property Sheet to help define the buttons and rectangles with their 'fill' properties. An example of an Order form with Command Buttons is shown below.



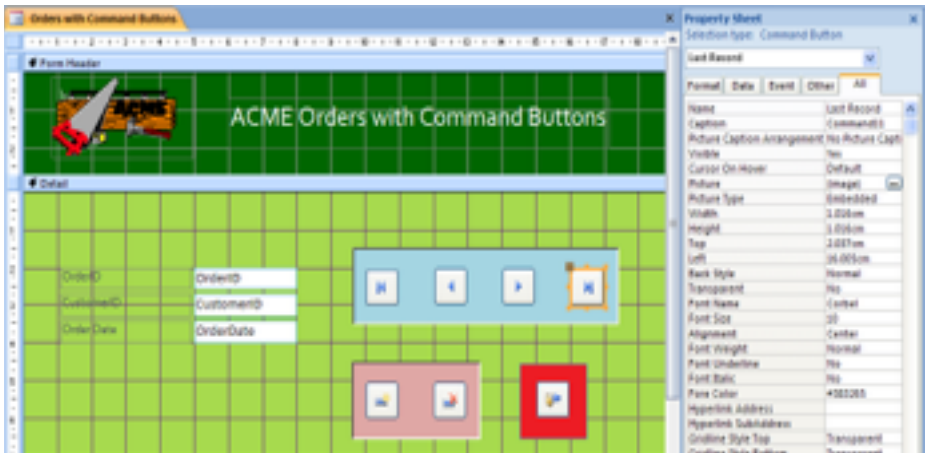
➢ when you are satisfied with the result, close the form saving your changes and giving your form a suitable name

Editing the buttons

Buttons are objects with properties which can be modified after initial set up.

➢ open the form you have just created in Design View and select one of the buttons

➢ open the Property Sheet for the button you have selected; it has the name that you gave it when you created it



➢ note that Picture Caption Arrangement says ‘No Picture Caption’; it is possible to add text to the button by altering the Picture Caption Arrangement to one of the positions (e.g. Top) and putting the required text against Caption

(Note that if you do this, you may have to resize the button to allow for the text.)

Good design means you may decide to standardise the sizes of some or all buttons.

- ➢ if you want to provide additional information or a help message, enter it in **Status Bar Text** (via Property Sheet) for the button; the text or message appears in the Status Bar (assuming it is activated) when the button is right clicked in Form View

Other buttons

Command Buttons can open other programs; for example, it might be useful when browsing your customer form to be able to activate Word to write a personal promotion letter to a particularly important customer.

Splash Screens

When your application is opened by the users, they will want to go straight to their particular part of the application, not a general opening Access screen. In order to facilitate this, we will give them an opening or “Splash” screen which will have the company logo and special design or message.

Also on this splash screen, we will give the users some options to go to depending on the application. This will be the **MAIN MENU**.

It is possible there may be some sub-menus and so on, and in a finished application (rather than a prototype) the user will navigate via the menu system to do what they need to do. The user may have to enter run-time parameters, but ideally they will not be bothered with database windows or other development-based screens.

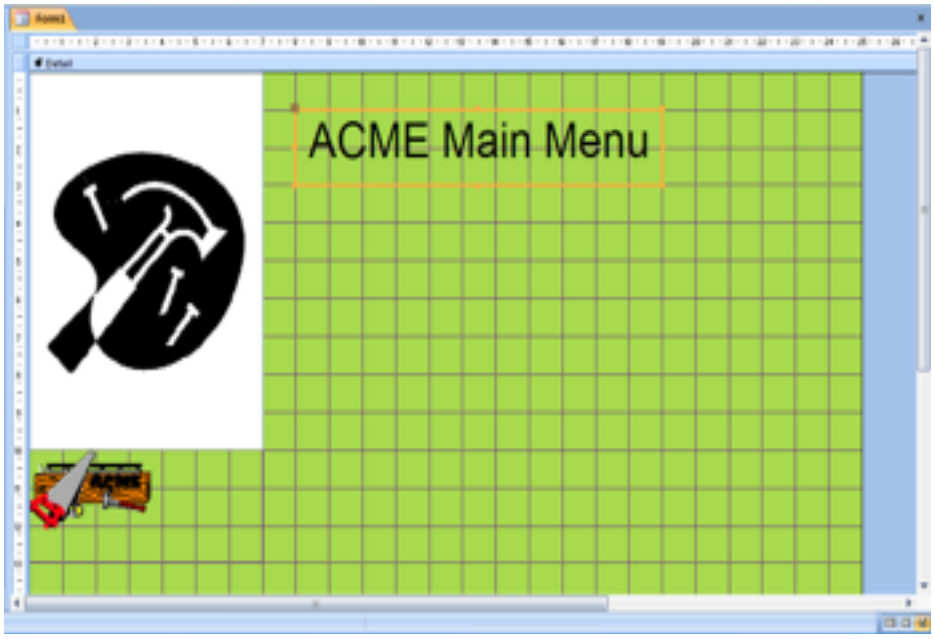
- ➢ in the database window, click **Create** and then **Form Design** – this will generate a blank form which we can use to produce a ‘switchboard’, that is, a starting point, for the application

- ➢ expand the design area to fill the available space

- ➢ leaving room for the buttons which will be used to access the "sub-menu" screens, paste in a picture suitable for the ACME Company

HINT: The easiest way to provide a picture for the splash screen is to open Word and import something from Clip Art. Access cannot import all picture formats; however, Word can handle WMF files from Clip Art files. You can then copy and paste the image from Word into the splash screen and position and re-size it accordingly. (Note that Paint Shop Pro, if you have it, will load/convert most commonly used graphical formats.)

The example used is from the Clip Art library supplied with MS Word (and is shown in Design View) along with the logo which has also been copied and pasted.



We will now add some Command Buttons that will take us to the next level menus or forms. (At this stage, when constructing a real application, you need to plot the structure of your application on a diagram so that the user navigation is clear.)

- assume that one of the important and frequent tasks is to add/edit product details
- add a button to open the 'Products with Discounts' form directly from the main menu screen – re-size the button as necessary to make it distinctive
- make sure the button describes accurately what function the user will get by clicking it
- switch to Form View to check that the navigation is working
- in order to return to the main menu, you would have to close the form manually; modify the 'Products with Discounts' form by adding a Close Form button
- similarly, add two more buttons to the main menu to navigate to the 'Customers' form and the 'Orders with Command Buttons' form; in both cases, modify the forms with Command Buttons so that they can be closed automatically
- finally, add a Close Form button to the Main Menu so that the application can be shut down – the result should be similar to the example below

Checklist for Groups, Buttons and Splash Screens

- When a column (field) has a small and limited number of choices, it is helpful to represent the selection as an Option Group.
- Option Groups can be set up using a Wizard in the **Design** group.
- You can use the method to store the value entered for use in another object in the application – for example a Basic module or macro.
- Option grouping in Access makes for a visually appealing solution which is also easy to work with.
- Restricting inputs on a data entry form is an excellent method of avoiding errors. It has the further advantage of making data entry faster.
- Navigating your application is much easier using Command Buttons in Access. Command Buttons can navigate from the opening screen, and back again.
- Clicking the button activates a macro that performs some designated action.
- Set up Command Buttons from the available tools with the form in Design View. This guides you through all the processes including the choice of event you want to trigger when the button is clicked. Make sure the Wizard is on!
- An opening graphic for the application can be an unbound form. An unbound object (such as is used for the splash screen) is not dependent on any other object.

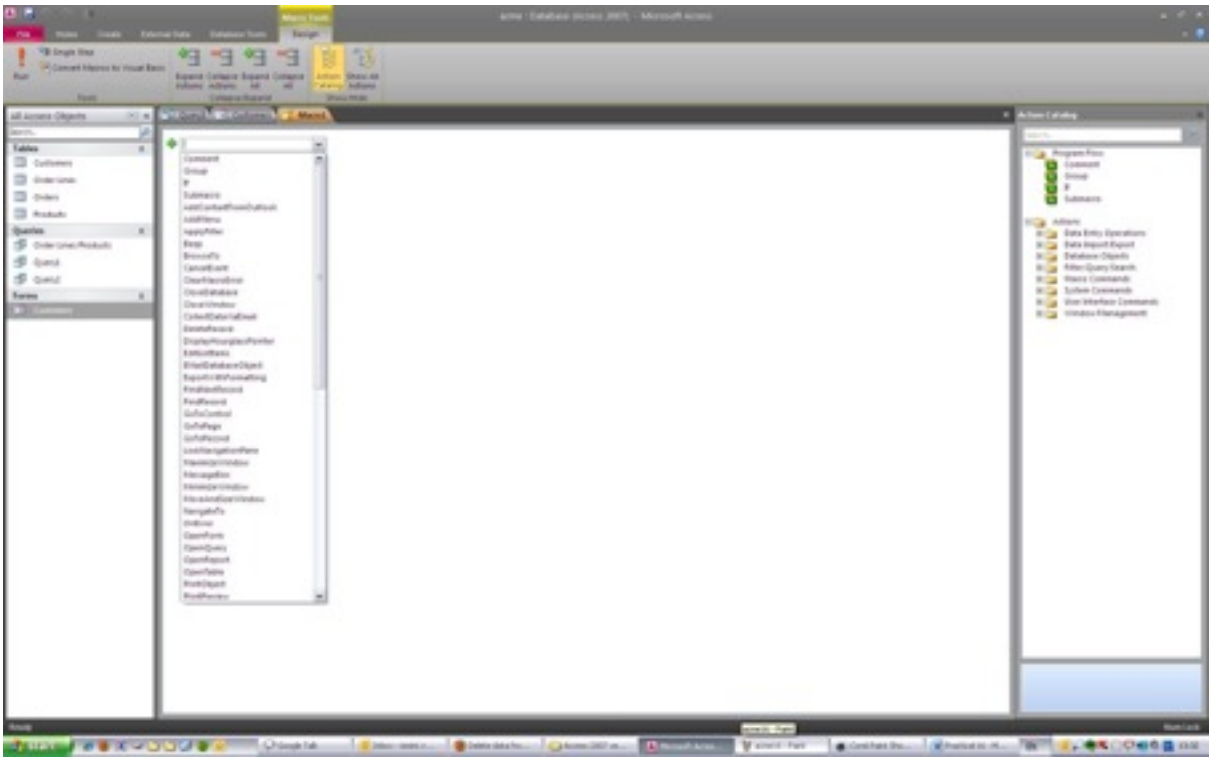
Macros in Access 2010

Macros

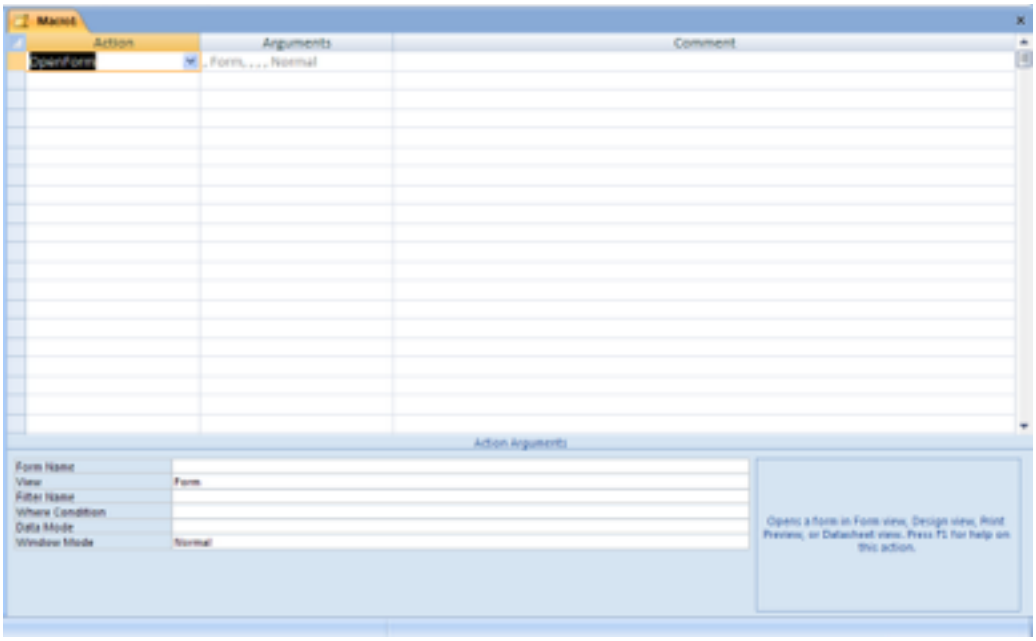
- A macro is a tool that allows you to automate tasks in Access.
- Macros are lists of instructions performed sequentially, rather like a short program.
- It is useful to save regularly-needed routines as a macro to save time and to standardise an application.
- Macros can call other macros.
- Macros can be assigned to short cut keys.
- There are a number of routines already available to use in Macros. We have seen some of them in a previous practical when we were setting up Command Buttons: OpenForm and GoToRecord are two examples.

➢ open the ACME database and click **Create** and then the **Macro** icon (in the **Macros & Code** group) .

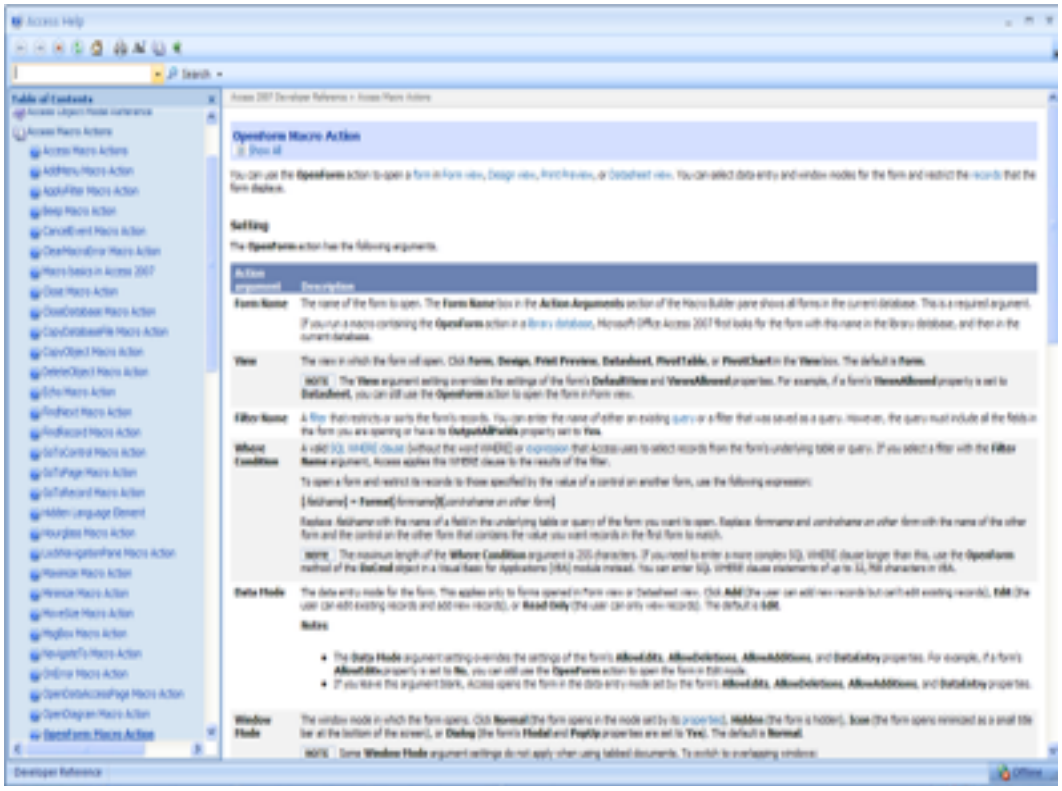
➢ on the left is the **Action** column; click on the drop down menu button to view a list of the available macros

➢ 

- click on some of the macros to see any **Arguments** (in the middle column), any **Action Arguments** (i.e. the parameters of the macro) in a panel at the bottom of the Macro window, and a description of the macro's function in a panel to the right of the Action Arguments – the example below shows the result of clicking the *OpenForm* macro



- Press F1 (as advised) to see Help on this particular macro and read it through carefully



We will set up a macro that will open the Products form that contains the option buttons for discounts.

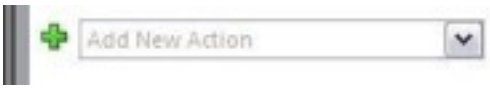
- close or minimise Help
- you should now be seeing the Macro window – if you are in the database window, click Create, then Macro
- in the Action column, select the **OpenForm** macro
- in the Action Arguments panel, click in Form Name and then on the drop down arrow which appears at the right-hand side
- select the 'Products with Discounts' form (or the name you gave this form) from the list
- ensure that the View parameter is set to Form (which is the default)
- leave Filter Name and Where Condition blank (which are the defaults)
- set the Data Mode parameter to Edit as we want the form to appear in Edit mode when it is opened (click in the space and then select the parameter from the drop down list)

- ensure that Window Mode is set to Normal (which is the default) – the form will then appear in accordance with its property definitions

- close the macro saving the changes and naming it as: Open Products wD Form (or something similar)
- the macro will be listed on the Navigation Pane under “Unrelated Objects”; double click the macro name and you should see the required form open up ready for editing

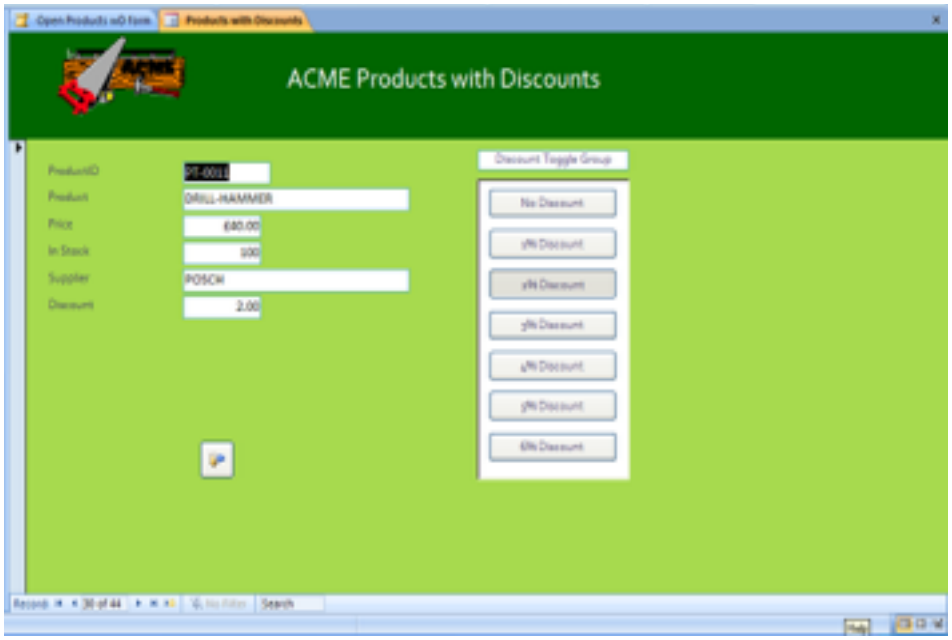
We will now edit this macro to enhance its capabilities. When the form opens, we may decide that we need to open a particular record, say Record number 30.

- Right click on the macro created above and select Design View
- click the drop down and select **GoToRecord**



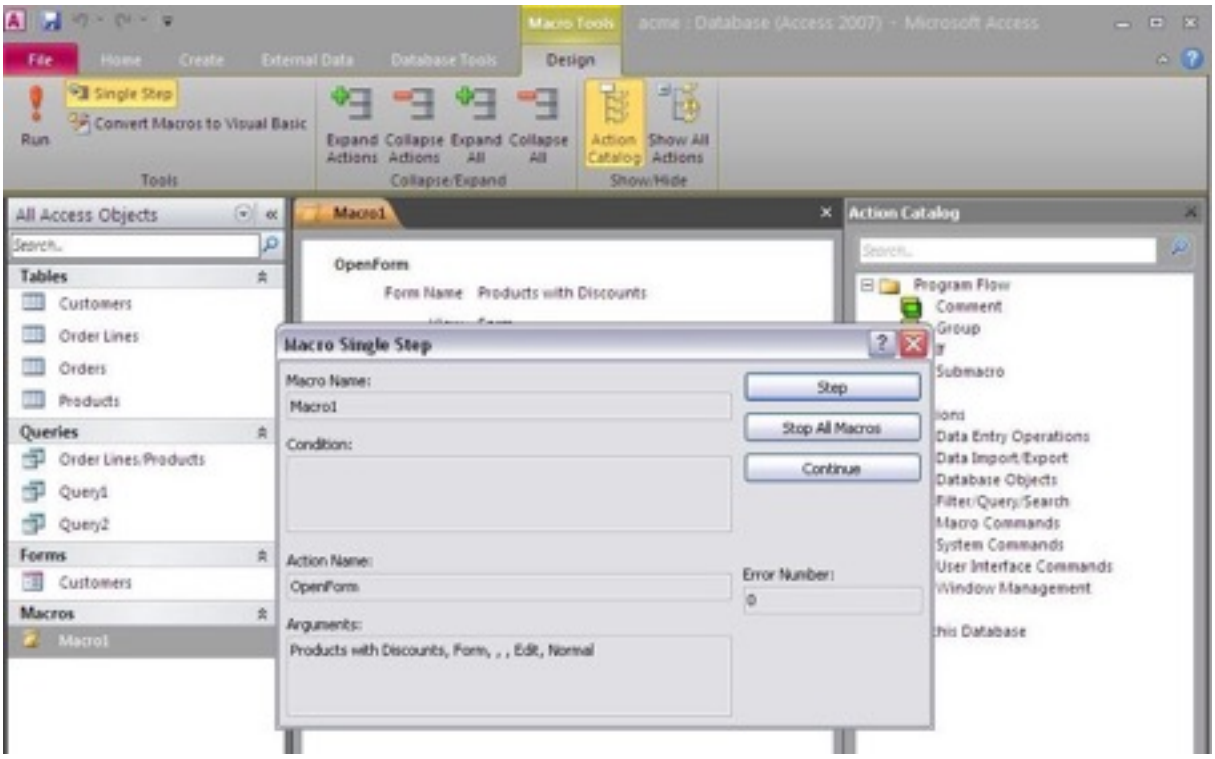
- in the Action Arguments panel, set the Object Type to Form
- next, set Object Name to Products with Discount (from the drop down list)
- set the Record parameter to Go To (selected from the drop down list)
- finally, type in 30 in the Offset cell

➢ click the **Run** icon to execute the macro (saving your changes) to check that it works correctly



Now we have two elements in the macro, working one after the other, automatically when it is run. What if for some reason, say for testing, we want to run some or maybe just one of the elements?

- ➢ open the revised macro in Design View
- ➢ click the **Single Step** icon (in the **Tools** group)
- ➢ click the **Run** icon and the Single Step screen will appear (as below)



- ➢ choose **Step** to run each element of the macro discretely
- ➢ note that **Continue** will run the rest of the macro to the end; it also removes single step mode
- ➢ try both these options on the macro you have just saved

Note that Single Step will remain operational until you turn it off

Autoexec Macro

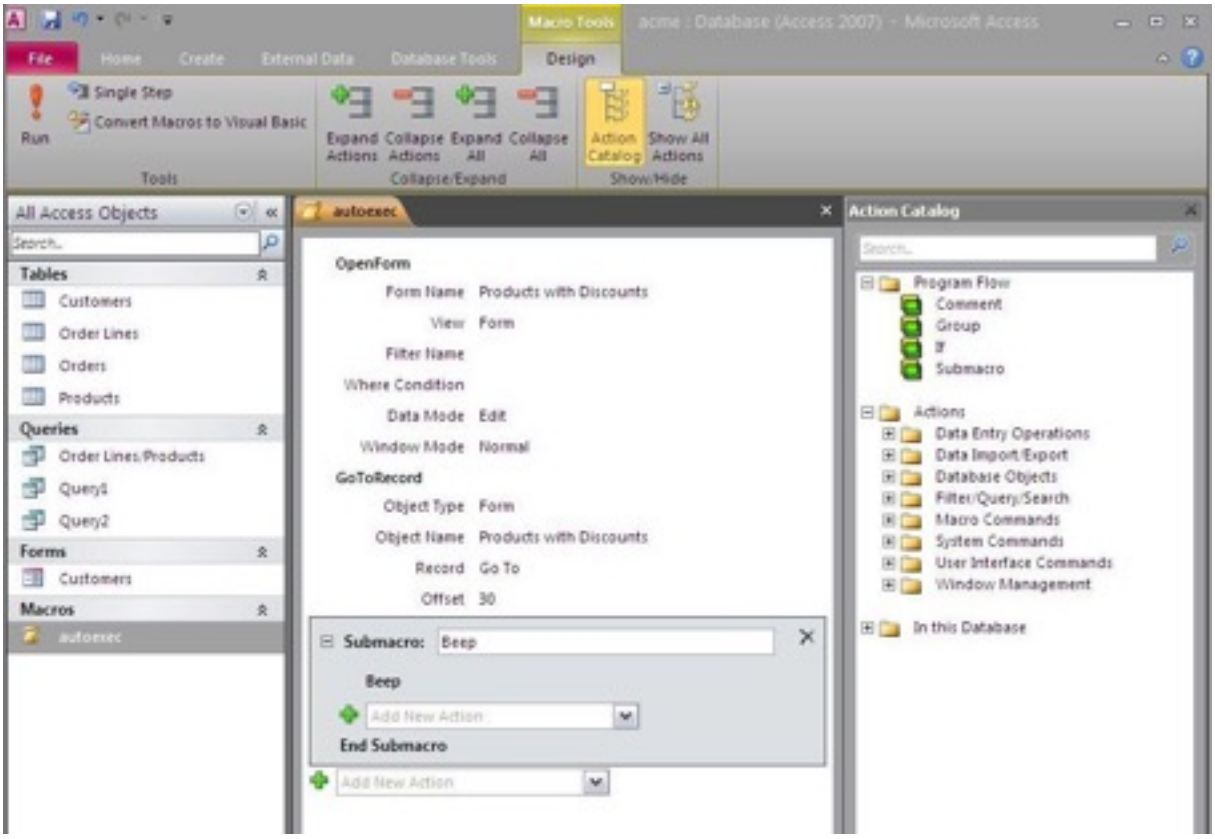
If you save a macro under the name “Autoexec” when your database is opened, this macro will always run. **So we can save our SPLASH opening screen under an OpenForm macro to go straight into our application on opening**. If you want to disable this function, just edit the name and re-save before closing your database.

More about Macros!

We will now go further in developing Macros. Access enables us to create a collection of Macros under one general group name. They can be picked out from a list when run. We will do this by adding to the macro we have already created beginning by editing it to add a ‘beep’ sound.

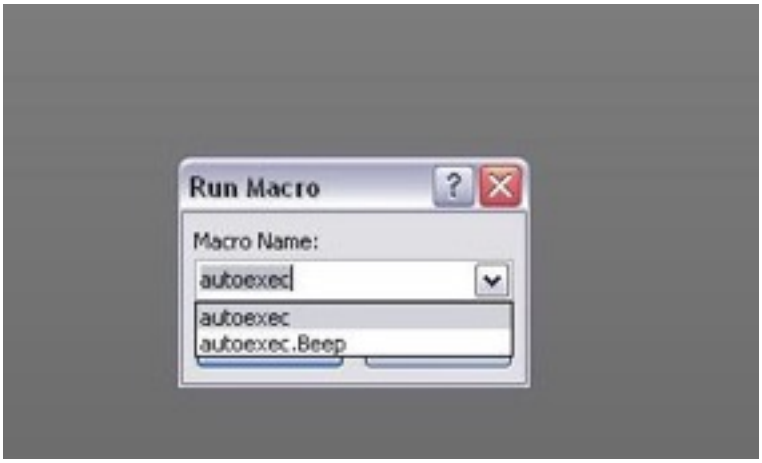
- open the macro in Design View
- click on the macro GoToRecord
- right click and choose “Make Submacro Block”
- name the submacro “Beep”
- choose **Beep** from the drop down list
- run the macro to see the effect of this last addition (try it in single step mode as well as normal running)

The next macro will be entered in the same window so it can be run from the same group heading. Enter the new macro as shown below. The OpenForm is to be the ‘Orders with Command Buttons’ form. The argument of the GoToRecord command is to be New.



➢ close the macro saving the changes

➢ ensure the macro is highlighted, then click **Database Tools** and **Run Macro** (in the **Macro** group) to see the effect of this last addition – note that you are presented with a drop down dialogue box from which you can select one of the macros to run (see below)



- ➢ try running each macro in single step mode as well – note that if you do not make a selection, then only the first macro in the group will run

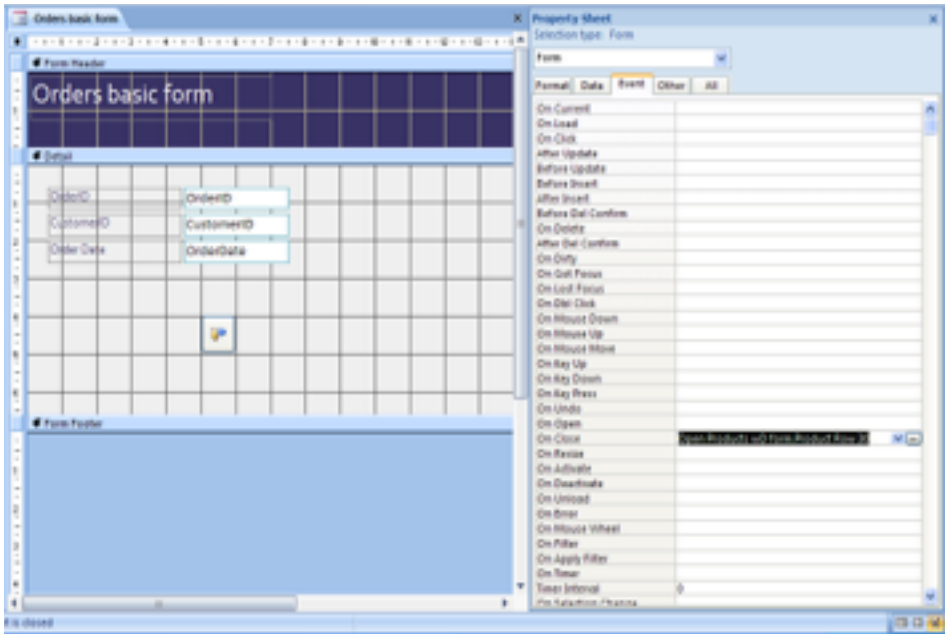
Grouping macros together in this way can make large developments much easier to manage. By giving the absolute name of the macro (group_name.macro_name) you can call any of the group macros from (for example) any Access Basic module attached to the application.

Macros triggered by Events

Events are the actions of OBJECTS. For example, a table can be opened, or closed; a form can be loaded or can have 'focus'. In general, events are actions recognised by an object, such as a mouse click or key press, for which you can define a response. An event can be caused by a user action or an Access Basic statement, or it can be triggered by the system.

Using properties associated with events, you can tell Access to run a macro, call an Access Basic function, or execute an event procedure in response to an event. We will construct an example.

- ➢ open the 'Orders basic' form in Design View
- ➢ open the Property Sheet for the whole form – if the form is not already selected, you can do this by clicking in the square button in the top left hand corner of the form, and then clicking on the Property Sheet icon
- ➢ click in the event **On Close** (you can view just the events by clicking the Event tab within Property Sheet)
- ➢ using the drop down list (via the arrow), enter the name of the macro to open the Product form at record 30



- close the 'Orders basic' form saving the changes
- open the 'Orders basic' form
- close it and see that the chosen macro is triggered by this event (note that the Products form is opened as a tabbed window – if you want to open forms as resizable windows, or use the Minimise, Maximise and MoveSize macro commands, you need to set the Access Options to **Overlapping Windows**; this setting is obtained via the Office Button)

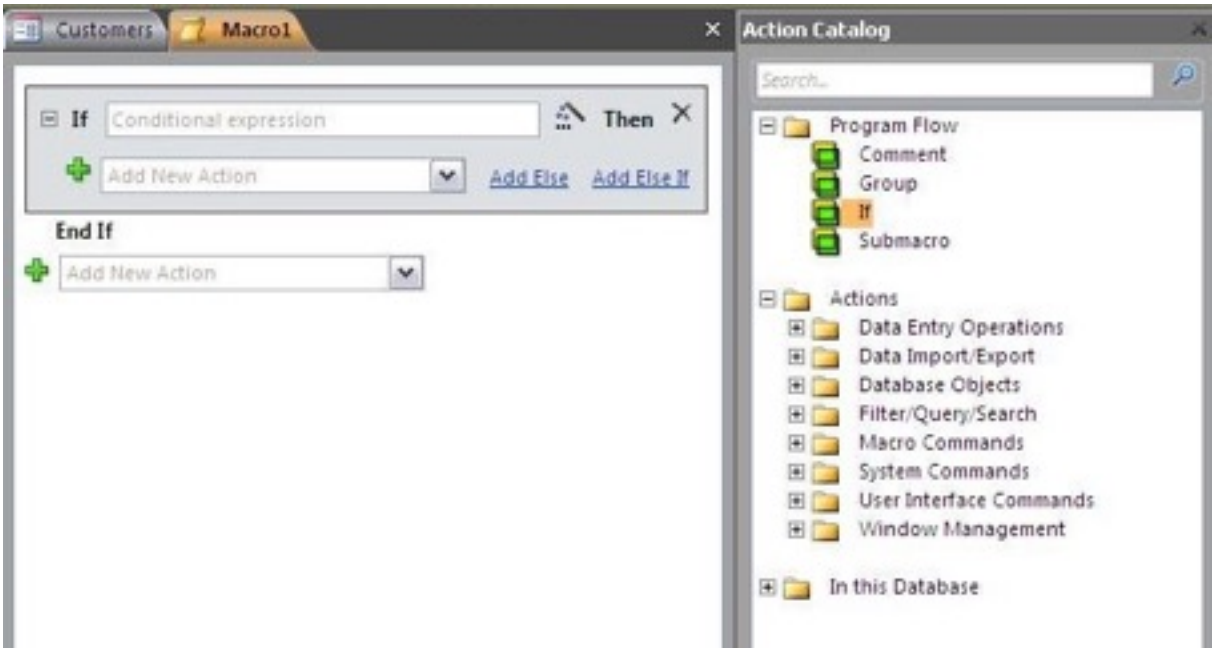
Triggering via Events is a very powerful tool for the Access Developer. Once objects are created, they can be easily made to act and react to situations. Conditions can be built into Macros giving even more control. That is the next topic.

Building conditions into macros

There is another hidden column in the Macro window – CONDITIONS. ACME would like to review all prices below £50 as they would like to increase profitability and in reviewing these prices, consider if they could be increased. The macro to write will test *UnitPrice* when this control is 'focused' in the form 'Products with Discounts' (or the name you assigned to the products form with the options group if it is different).

The first step is to write the macro.

- click on **Create**, then **Macro**
- Double Click If at the Action Catalog



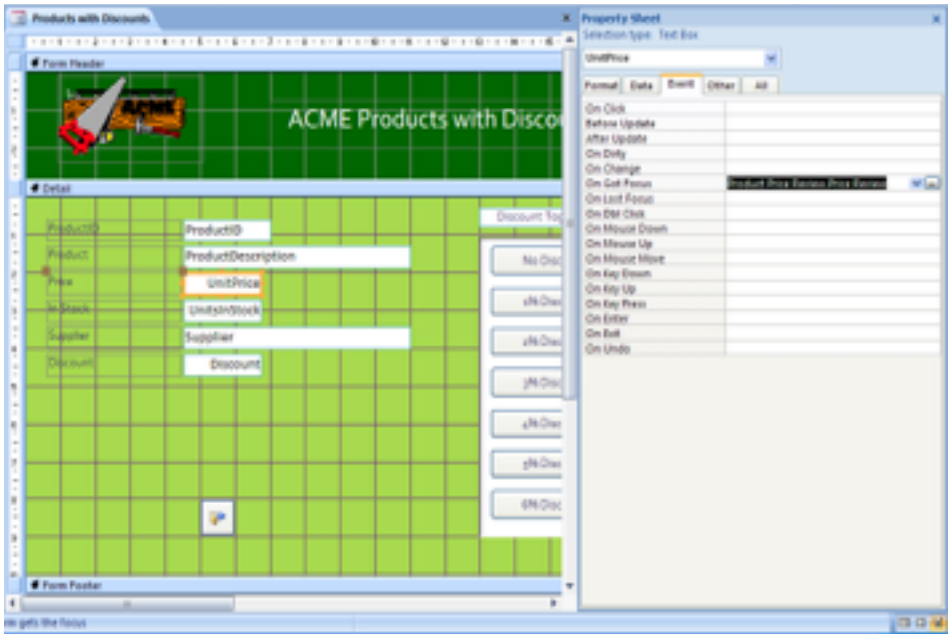
➢ set out the macro as shown below – this will produce the message box when a price is at or below £50, providing the Unit Price field is 'focused' – this can be done by tabbing to the field when the form is opened

Next, we associate the macro we have just created with the field when it is 'focused'.

➢ open the 'Products with Discounts' form in Design View

➢ highlight *UnitPrice* and open its Property Sheet

➢ under the Event tab, click in the **On Got Focus** cell and select the name of the macro just created ('Product Price Review.Price Review')



➢ close the form saving the changes

Note the definition of the required control [Forms]! etc, is not required to be in this "absolute", i.e. unambiguous format since the macro runs from the form itself. **[UnitPrice]** would do; however, the example illustrates the full syntax.

➢ to check the working of this, open the 'Products with Discounts' form, tab to the *UnitPrice* field and, using the field navigation buttons at the bottom, cycle through all the records; you should get a warning message every time a price at or below £50 appears

Data validation is an obvious and simple use of macros. True and False conditions can be added to the Conditions column for example and actions based upon values returned. Triggering a macro on a **Before Update** event on a control in a form could be useful to warn of and prevent the entering of an illegal value, e.g. one which is outside the range of stock locations for a particular item in a warehouse.

Checklist for Macros

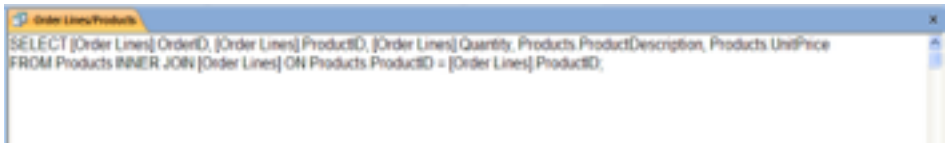
- Macros are lists of instructions. They are like small programs.
- They are useful when a particular action is repeatedly performed.
- A series of actions may be easily recorded and “played back” when required.
- Macros can be run directly as required by the user, or they may be triggered by something else in the database application.
- Macros can be collected together under a group name but can be run separately. This has the advantage of minimizing the overall number of macro names.
- Assigning a macro to a command button is a good way of streamlining an application and giving it a professional look.
- Adding such buttons to an unbound form is a good way of opening an application (often called a “Splash Screen”.)
- Using an AUTOEXEC macro enables the application to start with a customized opening screen rather than the database window – again adding to the professional appearance.
- Object EVENTS such as the Closing of a Form, or the Updating of a Column via a Form can TRIGGER macros. The macro can be entered via the Property Sheet.
- Macros can be used for data validation when entering validation rules at the table level is not desirable or where it does not give the degree of control required.

Creating SQL scripts in Access 2010

So far, your querying of the ACME database has used the Query By Example (QBE) graphical interface. Access stores every query as an SQL language command script. Some types of querying can only be done using SQL.

SQL is a database language. It is non-procedural – you tell the database what you want, and in what circumstances and with what criteria, and then run the script. We will begin by looking at the SQL that lies behind some of the queries we have written for ACME.

- ➢ open the ACME database
- ➢ run the query **Order Lines/Products**
- ➢ examine the rows returned from this query
- ➢ click on **View** then **SQL View** to see the SQL (which should look similar to the screen below)



Note that the query names may be different from yours.

We will have a closer look at the SQL script shown.

SELECT is the key command word in the SQL query statement. This key word is used in SQL for querying or extracting data. The rest of the statement indicates the tables to be used and the particular records (rows) to be returned based on any supplied conditions.

(NOTE: the conventions used below are for clarity of meaning, but are not used in the actual coding: **{ }** (braces) enclose required syntax, [ ] (square brackets) enclose optional elements.)

SELECT {list of columns to be returned}
FROM {list of tables used with relational characteristics (e.g. JOINs)}
[WHERE selection criteria for rows to be returned]
[GROUP BY specification used for grouping]
[HAVING selection criteria for groups]
[ORDER BY sort order];

The example from the ACME database queries related tables, hence the use of INNER JOIN. Below you will find more detailed reference to these keywords.

Some Important Examples of SQL code

Creating Tables

The basic structure for this is as follows:

CREATE TABLE table_name (column_name data_type [column_size]
[, column_name data_type [column_size] …])

This creates a table of fields and properties as set out in the statement. Note that only one column is necessary for a valid table, and that column_size is often a required property.

Now for a simple example: We need to create a Repairs table for hardware returned to ACME with problems. We will store the Customer and Product ID, together with a Memo column to describe the fault.

- ➢ open the ACME database (if not already open)
- ➢ click **Create** then **Query Design**
- ➢ close the Show Table dialogue box without adding any tables
- ➢ click the **SQL** icon (in the **Results** group)
- ➢ you will get a blank screen except for the default SQL “SELECT;”
- ➢ enter the following script (overwriting the default SQL) – note that you can enter it continuously or line by line

CREATE TABLE Repairs (ProductID Text(6), CustomerID Text(5),
[Details of Problem] Memo)

- ➢ click the **Run** icon
- ➢ assuming no errors, open the Repairs table from the Navigation Pane to check that it has been created correctly

Note that the column name ‘Details of Problem’ needed square brackets because it contains spaces. (Alternatively, you could have used underscores instead of spaces to make a continuous field name.)

Indexes

Indexes have been introduced earlier as an important feature of good database design. The command following creates an index on a column definition when it follows that definition directly.

CONSTRAINT index_name {PRIMARYKEY | UNIQUE | REFERENCES
foreign_table [(foreign_column)]}

As you can see, the index can be defined as a primary key or a unique index. A relationship can be established with another (foreign) table using REFERENCES. The foreign_column item is required if this column is not a primary key.

ALTER TABLE enables you to add new columns or delete a column with:

(ADD COLUMN column_name …)
(DROP COLUMN column_name ..)

Similar commands can be used to enable DROP INDEX and DROP TABLE. Details can be found in Access On-Line Help.

We will now generate our own SQL example from scratch. Follow the same steps as we did above to Create the Repairs table until you get the SQL edit window.

- ➢ you get a blank screen except for the default opening “SELECT;”
- ➢ we are now ready to enter our SQL command

TASK: To produce from the Customers table a list of the Customer names, Post Codes and Phone numbers for those in phone area “0566” (for example, for market research or delivery information.)

Note that in the notes below, the arrow brackets <> will be used to denote the beginning and end of the expression to be typed in.

- ➢ we will accept the default expression
- ➢ BEFORE the semi-colon, we need to enter the names of the columns we want to be returned in our query
- ➢ SO: enter <BusinessName, PostCode, PhoneNumber> as the list of columns required.

You MUST enter the object names precisely. If you do not, you may not get a flagged error, but SQL will interpret the mistake as a PARAMETER.

- ➢ on a new line, enter <FROM Customers>, to indicate from which table the columns will be taken
- ➢ on a new line, enter <WHERE PhoneNumber Like “0566*”> – this uses an expression similar to one in the example we looked at previously

- ➢ on a new line, enter <ORDER BY BusinessName>
- ➢ the complete program is listed below

```
SELECT BusinessName , PostCode, PhoneNumber
FROM Customers
WHERE PhoneNumber Like "0566*"
ORDER BY BusinessName;
```

- ➢ run the program – the result is shown below

Query1

Company	Post Code	PhoneNumber
BUILDIT BUILDERS LTD	RY8 4TY	0566 67429845
D.B.S.	RY2 6NR	0566 23498745
ENSIGN IMPROVEMENTS	RY1 3GB	0566 77766459
HARRISON & SONS	RY5 7RW	0566 78920201
LEAKNOT PLUMBING CO	RY2 6RD	0566 39256491
R. WATT & A. PRATT	RY2 6YU	0566 34572992
THE FORGE JOINERY	RY2 4AS	0566 39758453

Note that the column(s) determining the criteria need NOT be included in those columns returned.

Now you have a go!

TASK: ACME want to check that they are not overstocking. So as a quick check, they want to query the Products table and list the ProductDescription, UnitPrice and UnitsInStock where UnitsInStock >= 200 for purposes of efficient stock management.

Follow the steps as above, ensuring that the object names are exactly as they are defined in the underlying table. When you run the query, it should return, of course, those products (and only those products) which have an in-stock quantity of greater than or equal to 200.

Query1

Product	Price	In Stock
DRILL BITS-WOOD	£5.00	250
DRILL BITS-MASONRY	£5.00	250
CLEANING FLUID	£2.00	500
FACEMASK	£4.00	1000
FACEMASK-RIGID	£7.50	500
STRIMMER-INDUSTRIAL	£29.00	200
CLEANER-INDUSTRIAL	£295.00	250
CHISEL-WOOD-1	£4.00	200
CHISEL-WOOD-2	£5.00	200
PLANE	£34.00	500
DRILL-CORDLESS	£30.00	500
SCREWDRIVER-ELECTRIC	£15.00	200
SAWBLADE-JIG	£5.00	300

More advanced examples of SQL

SQL can run any type of query, including the cross-tab queries we did earlier. We will take a closer look at its use with aggregate functions in general.

SELECT {list of columns to be returned}, aggregate_function(column_name)
AS alias_name
FROM {list of tables}
[WHERE selection criteria for rows to be returned]
[GROUP BY specification used for grouping]
[HAVING aggregate criteria]
[ORDER BY sort order];

The alias_name is used to head the column referred to when the command runs. We will use it in the next example.

The aggregate functions are listed below:

FUNCTION	RESULT
Avg	Average
Count	Count
Min, Max	Minimum or Maximum values of a column
StDev, StDevP	Standard Deviation (see HELP for more detail)
Sum	Sum of column values returned
Var, VarP	Variance (see HELP for more detail)

For example the SELECT command segment for returning the average of the returned list of PRICE would be written Avg(PRICE).

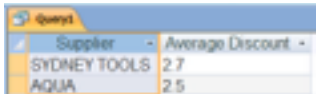
TASK: We are to list the Average Discounts of the Products table for the Suppliers SYDNEY TOOLS and AQUA for the goods whose UnitPrice is greater than £9. We want to list in descending order of discounts and also to head the discounts column with “Average Discount”.

Following the steps previously covered for entering SQL commands, enter the following script and run it.

```
SELECT Supplier, Avg(Discount) as [Average Discount]
FROM Products
WHERE Supplier = "SYDNEY TOOLS" or Supplier = "AQUA" and UnitPrice > 9
GROUP BY Supplier
ORDER BY Avg(Discount) DESC;
```

Note that the alias is enclosed in square brackets because it contains spaces.

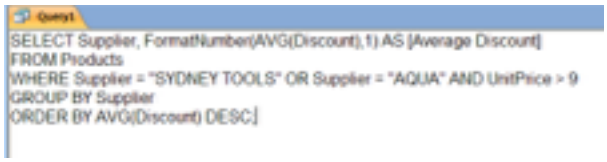
Your result should be as follows:



Supplier	Average Discount
SYDNEY TOOLS	2.7
AQUA	2.5

When you run the query as given above, you will get an excessive number of decimal places in the Average for Sydney Tools. In order to get sensible precision, you need to use a formatting command (see Access HELP for details.)

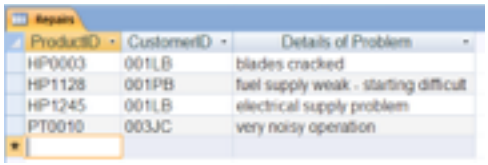
The output shown was achieved by amending the query as follows.



```
SELECT Supplier, FormatNumber(AVG(Discount),1) AS [Average Discount]
FROM Products
WHERE Supplier = "SYDNEY TOOLS" OR Supplier = "AQUA" AND UnitPrice > 9
GROUP BY Supplier
ORDER BY AVG(Discount) DESC;
```

Creating JOINS using SQL

➢ enter the following data into the recently created REPAIRS table



ProductID	CustomerID	Details of Problem
HP0003	001LB	blades cracked
HP1128	001PB	fuel supply weak - starting difficult
HP1245	001LB	electrical supply problem
PT0010	003JC	very noisy operation

We are now going to use the SQL route to give the results of a join of this table to both the Products and Customers tables. This is an INNER JOIN or EQUI JOIN. It returns rows where a precise equivalence exists. In this case, we are getting the name of the Product and Customer and the details of the problem. We are only interested in those rows where the ProductID and CustomerID match the values on the Repairs table.

Access can only establish relationships using the QBE method on precise equivalences. Some other databases, for example FOXPRO, can establish graphical interface relationships on inequalities. To do this in Access you must use SQL or a combination of queries, maybe on already related tables.

➢ close the Repairs table, saving the changes

➢ click **Create**, then **Query Design**

➢ close the **Show Table** window

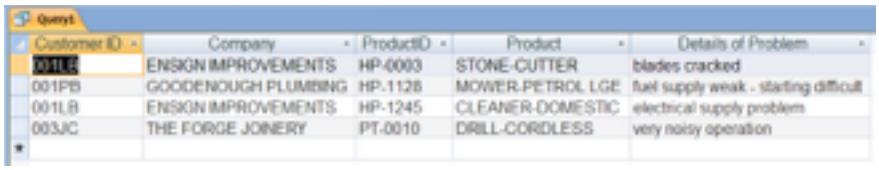
➢ click the **SQL** icon

➢ enter the following query; (note that where we have the same column names in different tables, we define the columns fully as **table_name.column_name**)

SELECT Customers.CustomerID, BusinessName, Products.Product ID,

```
        ProductDescription, [Details of Problem]
FROM Products INNER JOIN (Customers INNER JOIN Repairs ON
     Customers.CustomerID = Repairs.CustomerID) ON
     Products.ProductID = Repairs.ProductID;
```

➢ run the query – you should get the following output



Customer ID	Company	ProductID	Product	Details of Problem
001LE	ENSIGN IMPROVEMENTS	HP-0003	STONE-CUTTER	blades cracked
001PB	GOODENOUGH PLUMBING	HP-1128	MOWER-PETROL LGE	fuel supply weak - starting difficult
001LB	ENSIGN IMPROVEMENTS	HP-1245	CLEANER-DOMESTIC	electrical supply problem
003JC	THE FORGE JOINERY	PT-0010	DRILL-CORDLESS	very noisy operation

You have NOT established relationships by this command! Verify this by clicking on the Relationships icon (on the Database Tools ribbon.) To establish relationships via SQL we must use the CONSTRAINT key word, shown at the beginning of the SQL section.

LEFT OUTER JOINS includes all of the records from the first – often called the "left-hand" of two tables, even if there are no matching values for records in the second (right-hand) table. We could see from this sort of query, for example, which salesmen had or had not made a sale for a particular period, rather than just those who had sold something.

RIGHT OUTER JOINS are as the same as left, but the second (right hand) table supplies all records in this case (whether there is a match or not).

Creating UNIONs using SQL

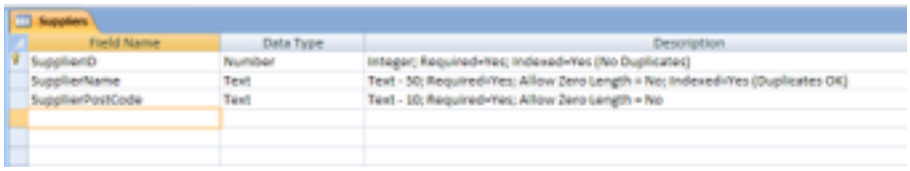
Combining two or more select queries can be done using the UNION command. In fact you can do this ONLY using SQL in Access.

We will now build an example using the ACME database. Firstly we will create a new table called **Suppliers**.

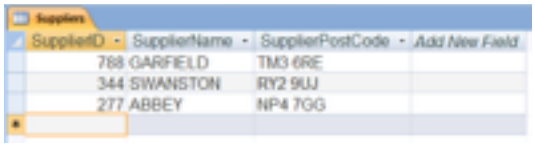
➢ open the ACME database (if not already open)

➢ click **Create**, then **Table**, then switch to **Design View** – you will be asked to save and name the table – name it: Suppliers

➢ set up the table as shown below; (note that DESCRIPTION information has been added to advise you how to define the field)



➢ having defined the new table, enter the data as shown below



We can now write a UNION Query based on Select statements on the tables Suppliers and Customers.

Assume we want to find those Customers and Suppliers who are located in similar areas for the purposes of arranging direct, ex-factory delivery.

➢ close the Suppliers table saving the changes

➢ click **Create**, then **Query Design**

➢ close the **Show Table** dialogue box and click the **SQL** icon

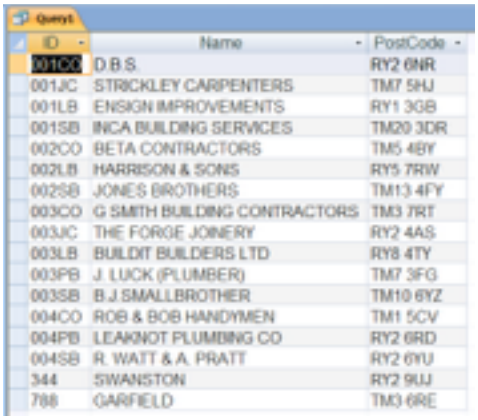
➢ enter the following query

```
SELECT CustomerID AS ID, BusinessName AS Name, PostCode
FROM Customers
WHERE PostCode Like "RY*" OR PostCode Like "TM*"
UNION
SELECT SupplierID AS ID, SupplierName AS Name, PostCode
FROM Suppliers
WHERE SupplierPostCode Like "RY*"  OR SupplierPostCode Like "TM*";
```

The number of columns in each SELECT statement must be the same. Also, you are allowed only ONE sort statement (if you use one), which must come at the end of the script.

Notice that the UNION SQL query can deal with different data types in the corresponding columns. In the above, the IDs are text and integer. The returned column defaults to text. See the result below.

➢ run the query – you should get the output shown below



ID	Name	PostCode
001CC	D.B.S.	RY2 6NR
001JC	STRICKLEY CARPENTERS	TM7 5HJ
001LB	ENSIGN IMPROVEMENTS	RY1 3GB
001SB	INCA BUILDING SERVICES	TM20 3DR
002CO	BETA CONTRACTORS	TM5 4BY
002LB	HARRISON & SONS	RY5 7RW
002SB	JONES BROTHERS	TM13 4FY
003CO	G SMITH BUILDING CONTRACTORS	TM3 7RT
003JC	THE FORGE JOINERY	RY2 4AS
003LB	BUILDIT BUILDERS LTD	RY8 4TY
003PB	J. LUCK (PLUMBER)	TM7 3FG
003SB	B.J SMALLBROTHER	TM10 6YZ
004CO	ROB & BOB HANDYMEN	TM1 5CV
004PB	LEAKNOT PLUMBING CO	RY2 6RD
004SB	R. WATT & A. PRATT	RY2 6YU
344	SWANSTON	RY2 9UJ
788	GARFIELD	TM3 6RE

Subqueries

Subqueries (i.e. Querying a Query) can be implemented using the nested query method already covered. The general form of the subquery which can be implemented using SQL is shown below.

SELECT {column_list}
FROM {table_list}
{WHERE table1_name.column_name IN}
(SELECT {select_statement}
[GROUP BY group_criteria]
[HAVING aggregate_criteria]
[ORDER BY sort_criteria]);

➢ enter the following SQL script (you should know how to do this by now!)

```
SELECT Customers.CustomerID, BusinessName, PhoneNumber
FROM Customers
WHERE Customers.CustomerID IN
    (SELECT Orders.CustomerID
    FROM Orders
    WHERE OrderDate BETWEEN #01/01/96# AND #29/02/96#);
```

➢ run the query and satisfy yourself that the result is what you would have expected

Action Queries

(1) Make table Queries

If you want to base a new table on an existing object, rather than create from scratch (compare with CREATE TABLES queries previously), you can do so using the following general form:

SELECT [DISTINCT | ALL] {select_list}
INTO {new_table_name}
FROM {source_table_name}
[WHERE select_criteria]

(2) Appending rows from one table to another

The general form is:

INSERT INTO {destination_table}
SELECT [DISTINCT | ALL] {select_list}
FROM {source_table}
[WHERE append_criteria]

(3) Deleting rows from a table

Do this by the following general form:

DELETE FROM {table_name}
[WHERE delete_criteria]

If you omit the WHERE clause, ALL ROWS will be deleted!

(4) Updating rows

There is a new key word here, **SET**:

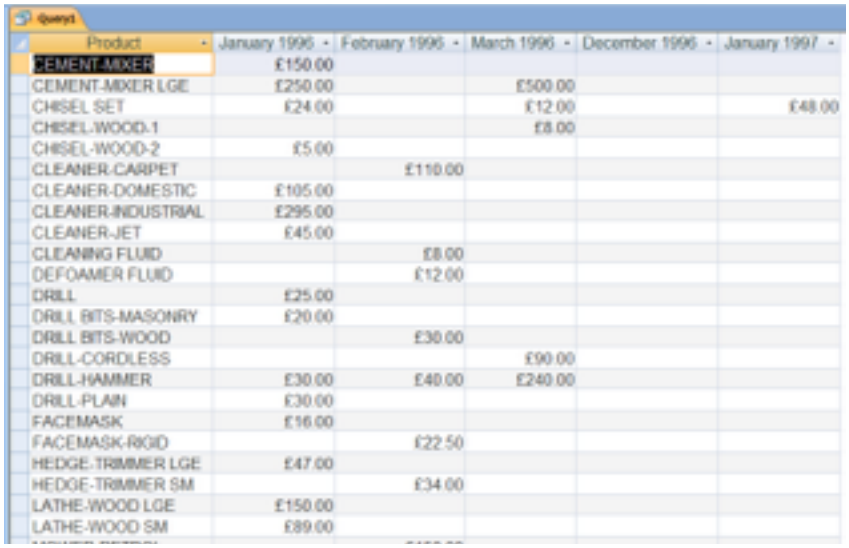
UPDATE {table_name}
SET column1_name = new_value1 [, column2_name = new_value2...]
[WHERE update_criteria]

The update criteria will be similar in form to the examples we have seen elsewhere in other types of queries.

(1) Cross Table output

In this case, the general form can be gleaned from an earlier example of the Cross_Tab query we produced using the QBE method. First the code, followed by some of the resulting output.

```
TRANSFORM Sum([Quantity]*[UnitPrice]) AS Expr2
SELECT Products.ProductDescription
FROM Products INNER JOIN (Orders INNER JOIN [Order Lines] ON
        Orders.OrderID = [Order Lines].OrderID)
        ON Products.ProductID = [Order Lines].ProductID
GROUP BY Products.ProductDescription
PIVOT Format(DateSerial(Year([OrderDate]),Month([OrderDate]),1),
        "mmmm yyyy")
        In ("January 1996","February 1996","March 1996",
        "December 1996","January 1997");
```



Product	January 1996	February 1996	March 1996	December 1996	January 1997
CEMENT-MIXER	£150.00				
CEMENT-MIXER LGE	£250.00		£500.00		
CHISEL SET	£24.00		£12.00		£48.00
CHISEL-WOOD-1			£8.00		
CHISEL-WOOD-2	£5.00				
CLEANER-CARPET		£110.00			
CLEANER-DOMESTIC	£105.00				
CLEANER-INDUSTRIAL	£295.00				
CLEANER-JET	£45.00				
CLEANING FLUID		£8.00			
DEFOAMER FLUID		£12.00			
DRILL	£25.00				
DRILL BITS-MASONRY	£20.00				
DRILL BITS-WOOD		£30.00			
DRILL-CORDLESS			£90.00		
DRILL-HAMMER	£30.00	£40.00	£240.00		
DRILL-PLAIN	£30.00				
FACEMASK	£16.00				
FACEMASK-RIGID		£22.50			
HEDGE-TRIMMER LGE	£47.00				
HEDGE-TRIMMER SM		£34.00			
LATHE-WOOD LGE	£150.00				
LATHE-WOOD SM	£89.00				

Running a Query from a Command Button

As we have already remarked, a Union query has to be effected in Access using SQL. And if we want to, we can run this or any other SQL script from a COMMAND BUTTON.

- ➢ open the ‘Customers basic form’ in Design View
- ➢ using the Command Button Wizard, add a Button to run the UNION query you produced earlier; (hint: you will need to use a Miscellaneous category and a Run Query action)
- ➢ label the Button with: Ranton and Tamarmouth Deliveries
- ➢ adjust font and/or button size as you please to show the label
- ➢ try out the form

FOOTNOTE

Some notes on SQL syntax are given at the end of this document. For more examples and information, read one of the numerous texts on SQL. You should be aware that Access SQL differs from the standard dialect of SQL in some respects. Sometimes the same code will be valid in both cases but will produce different results in different databases! You will use a more standard dialect next year when you study SQL in greater depth.

Access on-line HELP is useful. Look for SQL, reserved words, and then seek what you want. Unfortunately the error messages in Access are not very helpful when using the SQL interface. Often you will have simply named an object wrongly, or missed a bracket or the final semi-colon.

Checklist on Access SQL

- SQL or Structured Query Language is a command language specifically designed to query databases for information.
- It is implemented in all major relational databases although the dialect used may differ.
- All queries written in Access produce code that can be viewed from the SQL viewer whether or not the query was initiated by it or via the QBE grid.
- In Access, most queries can be effected via QBE although it is sometimes easier or only possible in SQL.
- SQL can be written off line from Access in any program editor or using a word processor in text mode. It can be copied and pasted into the Access SQL window and run, saved and edited as required.
- In Access the Help given if an SQL program fails is often not very helpful! However, reference to the SQL Reserved Words listing in the Access Help file will sometimes be of assistance. Experience of SQL helps – always check that you have used the correct field names if you get an error!

Appendix – SQL Notes Reference

Note that all components of SQL are not sensitive to case except for character (text) literals which must be enclosed in single quotes. Thus 'smith', 'SMITH', and 'Smith' are each different literals. This is important when searching for attribute values (fields) which are character (text) strings; unless modified by "wildcard" pattern-matching characters, only exact matches will be retrieved. In this document, the following convention is used to define SQL statements.

Reserved words are shown in upper case; for example, SELECT.

User-defined words are shown in lower case; for example, customer_number.

A vertical bar "|" indicates a selection may be made; for example, a | b | c (means a or b or c).

Braces "{ }" indicate a required element; for example, {a}.

Square brackets "[ ]" indicate an optional element; for example, [b].

An ellipsis "…" indicates optional repetition of an element; for example, {a | b} [,c…] (means a or b followed by zero or more repetitions of c separated by commas)

Legibility can be maintained by laying out commands neatly using new lines and indentation as per the examples below.

Data Manipulation Language command SELECT

All database queries in SQL are effected through the SELECT command; it has the following format.

```
SELECT [DISTINCT | ALL] {* | [column_expression [AS new_name] ] [,…] }
        FROM table_name [alias] [,…]
        [WHERE condition]
        [GROUP BY column_list] [HAVING condition]
        [ORDER BY column_list];
```

column_expression	is a column name or expression based on a column name
table_name	is the name of an existing database table or view
alias	is an abbreviation for the given table or view
column_list	is a list of one or more column names
;	is used as the statement terminator

A SELECT statement is processed according to the following sequence.

FROM	allocates the table(s) to be used
WHERE	if present, filters the rows according to the supplied condition
GROUP BY	if present, groups the rows by the same column value
HAVING	if present, filters the groups according to the supplied condition
SELECT	specifies the columns in the new (resulting) relation
ORDER BY	if present, specifies the sequence of columns in the new (resulting) relation

STAFF

e_no	fname	surname	address	telno	position	sex	dob	salary	branch
456	John	White	5 Old Rd, Plymouth	789345	Manager	M	1-Oct-60	30000	B5
127	Ann	Beech	31 Oak Rise, Plympton	218765	Snr Asst	F	3-Apr-67	12000	B3
366	David	Ford	15 The Close, Oreston	145790	Deputy	M	23-Jan-70	18000	B3
429	Mary	Howe	28 Glebe Lane, Saltash		Assistant	F	3-Sep-75	9000	B7
721	Susan	Brand	51 Broad St, Plymouth	367890	Manager	F	14-Nov-65	24000	B3
735	Julie	Lee	17 The Acres, Plymstock	347245	Assistant	F	7-Aug-79	9000	B5

Query: List all staff details (=> retrieve all rows and all columns).

```
SELECT  *
        FROM  staff;
```

The result will be the complete Staff table. "*" allows the retrieval of all columns without needing to specify them individually.

Query: List staff salaries showing employee number and name (=> retrieve specific columns)

```
SELECT  e_no, fname, surname, salary
        FROM  staff;
```

e_no	fname	surname	salary
456	John	White	30000

127	Ann	Beech	12000
366	David	Ford	18000
429	Mary	Howe	9000
721	Susan	Brand	24000
735	Julie	Lee	9000

Note that if duplication of rows was possible, but not required in the result, then the reserved word DISTINCT can be applied to eliminate the duplicates. Thus

```
SELECT  DISTINCT  e_no, fname, surname, salary
        FROM  staff;
```

Query: List staff salaries as a monthly figure showing employee number and name (=> retrieve specific columns, also using an expression and an AS phrase).

```
SELECT  e_no, fname, surname, salary/12 AS monthly_salary
        FROM  staff;
```

e_no	fname	surname	monthly_salary
456	John	White	2500
127	Ann	Beech	1000
366	David	Ford	1500
429	Mary	Howe	750
721	Susan	Brand	2000
735	Julie	Lee	750

Calculated (or 'computed', or 'derived') fields can be produced using addition, subtraction, multiplication, and division. More than one column can be included in the calculation, and parentheses can be used to clarify the precedence of operations. If the AS phrase is not included, the column heading for the output will be determined by the SQL dialect implementation.

Query: List all staff with a salary greater than £10,000 showing employee number, name, and position (=> retrieve specific rows according to a condition).

```
SELECT  e_no, fname, surname, position, salary
        FROM  staff
        WHERE  salary  >  10000;
```

e_no	fname	surname	position	salary

456	John	White	Manager	30000
127	Ann	Beech	Snr Asst	12000
366	David	Ford	Deputy	18000
721	Susan	Brand	Manager	24000

The conditions (or predicates) for row selection which can be applied are of five types.

Comparison compare the value of a field (or expression) against the value of an expression

The following arithmetic operators can be used

=	is equal to	<>	is not equal to
<	is less than	<=	is less than or equal to
>	is greater than	>=	is greater than or equal to

The following Boolean (logical) operators can be used

NOT, AND, OR

Range test whether the value of a field (or expression) falls within specified boundaries

Format: WHERE column_expression BETWEEN lower_bound AND upper_bound

Set membership test whether the value of a field (or expression) equals one in a set of values

Format: WHERE column_expression IN (set_value [,…])

Pattern match test whether the value of a field (or expression) matches a pattern (string)

Format: WHERE column_expression LIKE 'pattern_expression'

The following "wildcard" characters can be used

%	(percent)	any sequence of zero or more characters
_	(underscore)	any single character

Null test whether the value of a field (or expression) is null (=empty)

Format: WHERE column_expression IS [NOT] NULL

Using the given STAFF table, work out the resulting new table for the following SELECT statements. See if the statement could have been written differently to provide the same result. In addition, work out what (natural language) query is behind the SELECT statement.

```
SELECT  e_no, fname, surname, position, salary
        FROM  staff
```

```
        WHERE  salary  BETWEEN  20000  AND  30000;

SELECT  e_no, fname, surname, position
        FROM  staff
        WHERE  position  IN  ('Manager', 'Deputy');

SELECT  e_no, fname, surname, address
        FROM  staff
        WHERE  address  LIKE  '%Plym%';

SELECT  e_no, fname, surname, address
        FROM  staff
        WHERE  telno  IS NULL;
```

Sorting

Sorting is effected by using the ORDER BY clause; the column_list is a list of column names to be sorted. Columns can be sorted into ASCending (smallest first) or DESCending (largest first) sequence. Some dialects allow sorting by columns not specified in the SELECT clause. Null values are sorted to one end or the other according to the dialect.

Query: List staff according to salary with the largest value at the top.

```
SELECT  e_no, fname, surname, salary
        FROM  staff
        ORDER BY salary DESC;
```

e_no	fname	surname	salary
456	John	White	30000
721	Susan	Brand	24000
366	David	Ford	18000
127	Ann	Beech	12000
429	Mary	Howe	9000
735	Julie	Lee	9000

Aggregate functions

There are 5 available which allow basic statistical analysis of data.

COUNT	returns the number of values in a specified column
SUM	returns the sum of the values in a specified column
AVG	returns the average of the values in a specified column

MIN	returns the smallest value in a specified column
MAX	returns the largest value in a specified column

All functions can be used on numeric fields. COUNT, MIN, and MAX can also be used on non-numeric fields. COUNT(*) counts all rows in a table; in other words, it includes nulls and duplicates. These functions can only be used in the SELECT or HAVING clauses; they operate on a single column and return a single value. Only non-null values are operated on. If duplicate values are to be disregarded, then the DISTINCT keyword must be used in the SELECT clause.

Query: Find the number of managers and the sum of their salaries.

```
SELECT COUNT(e_no) AS no_managers, SUM(salary) AS tot_salary
    FROM staff
    WHERE position = 'Manager';
```

no_managers	tot_salary
2	54000

Query: Find the minimum, maximum, and average salaries.

```
SELECT MIN(salary) AS min_sal, MAX(salary) AS max_sal, AVG(salary) AS
avg_sal
    FROM staff;
```

min_sal	max_sal	avg_sal
9000	30000	17000

Grouped queries

The summaries looked at so far are akin to the totals which might be placed at the end of a report. It is often useful to have sub-totals and other groupings provided. This can be achieved by using the GROUP BY clause; this is then called a grouped query. The effect is to group the data from the SELECTed table(s) according to the grouping columns specified in the GROUP BY clause. There are restrictions on using this clause. Each grouping column must be single-valued per group, and the SELECT clause may only contain column names, aggregate functions, constants, or expressions limited to combinations of these elements.

All column names in the SELECT clause must appear in the GROUP BY clause unless they are the argument of an aggregate function. In contrast, column names

may appear in the GROUP BY clause that have not been specified in the SELECT clause. Null values are considered to be equivalent for the purposes of grouping data.

Query: Find the number of staff working in each branch and the sum of their salaries.

```
SELECT  branch, COUNT(e_no)  AS  no_staff, SUM(salary)  AS  sum_sals
        FROM  staff
        GROUP BY branch
        ORDER BY branch;
```

branch	no_staff	sum_sals
B3	3	54000
B5	2	39000
B7	1	9000

Conceptually, SQL has performed the following sequence of operations.

1. The staff were grouped by branch so that staff in the same branch were brought together.
2. For each group, the number of staff were counted and their salaries summed.
3. The 3 rows of output were sorted by branch identifier in (default) ASCending sequence.

The grouping of data can be restricted by using the HAVING clause. Its operation is similar to the WHERE clause; but whereas WHERE filters rows, HAVING filters groups. Column names must be specified in the GROUP BY clause or be the argument of an aggregate function in order to be included in the HAVING clause.

Query: Find the number of staff working in each branch and the sum of their salaries for all branches with more than one member of staff.

```
SELECT  branch, COUNT(e_no)  AS  no_staff, SUM(salary)  AS  sum_sals
        FROM  staff
        GROUP BY  branch
        HAVING  COUNT(e_no)  >  1
        ORDER BY  branch;
```

branch	no_staff	sum_sals
B3	3	54000
B5	2	39000

Subqueries and Multi-table queries

A subquery is made when a SELECT statement is embedded within another SELECT statement. This "inner" SELECT or subselect can only be applied in a WHERE or HAVING clause. There are three types of subquery.

Scalar returns a single row and column (= a single value)
Row returns a single row which can contain several columns
Table returns a relation (= several rows by several columns)

The query examples so far looked at apply the SELECT statement to a single table; this is obviously limiting. SQL allows a number of tables to be used in a SELECT statement by an implied join. Note that multiple table access can also be effected using a subquery.

Assuming that a table of properties for rent exists (property) and a table of potential clients (client), the following query could be satisfied.

Query: List all clients who have viewed a property for rent along with any comments.

```
SELECT  client.c_no, fname, surname, prop_no, comment
        FROM  client, property
        WHERE  client.c_no  =  property.c_no
```

c_no	fname	surname	prop_no	comment
C56	Paul	Elliott	P108	
C56	Paul	Elliott	P034	too small
C56	Paul	Elliott	P098	
C62	Alice	Mills	P034	no dining room
C76	James	Bolton	P098	too remote

Joins therefore represent 1 to many sets of either foreign key or compound key master to detail relationships.

Note that, as with single table SELECTs, results of joins can be sorted (ORDER BY) and grouped (GROUP BY). Other types of join can be specified as per the taught relational operators. Thus NATURAL JOIN and OUTER JOIN are typical of what is possible. These can be specified in the FROM clause in the SELECT command.