

Introduction to MS ACCESS 2007



Tables
Relationships
Queries
Forms
Reports



Table of Contents

Database concepts	2
What is a relational database?	2
Some important Access data types & field sizes	3
Tables	5
Creating a database table	5
Extending your database table	6
Exercise.....	7
Relationships	10
Relationships between database tables.....	10
Exercise.....	11
Queries	14
Introduction to Queries	14
Creating a Select query	14
More Select queries.....	18
Action queries.....	21
Update queries	23
Forms	26
Creating a form to view your data	26
Creating a form with a subform	27
Advanced forms.....	29
Reports	32
Reports	32
Exercise.....	32
The Access Switchboard	34
Why use a switchboard?	34
Exercise.....	34

This training manual is licensed under a [Creative Commons Attribution 3.0 License](https://creativecommons.org/licenses/by/3.0/)

Compiled by Jane M. Nash

For information please email jnash@telkomsa.net



Database concepts

What is a relational database?

A relational database consists of a set of separate (but related) tables.

- Each table contains data about one business entity, such as customers, stock, or employees.
- Within a table, there will be one record for each individual customer, stock item, employee etc. So a table consists of a set of unique records.
- Each record consists of many fields (eg for employees you might want to store surname, first name, phone number, address, salary). Each of these fields is defined as being of a particular data type (text, number etc), and can have rules associated with it (eg a postal code must consist of 4 digits).

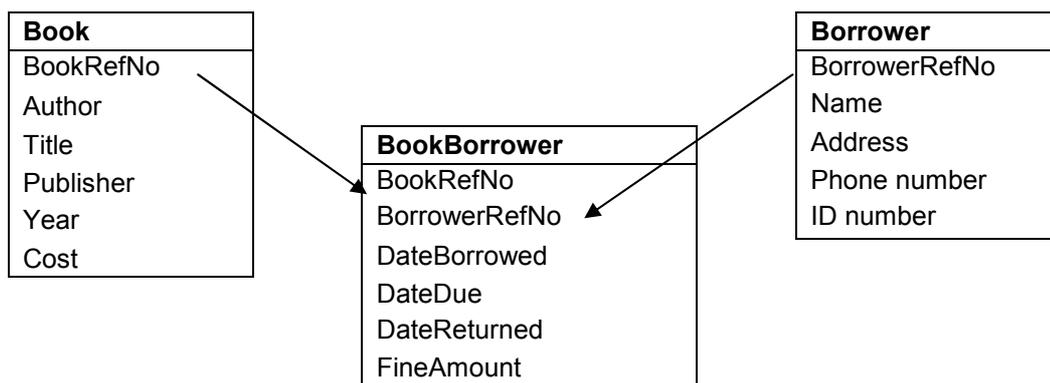
Since a table consists of unique records, in this example one for each employee, you need a way to tell the different employees apart, even if two of them have the same surname or live at the same address. To do this you would probably have an “employee number” field. The field that is used to uniquely define each record in a table, is called a primary key.

In order to link separate tables together, the primary key of one table is repeated in another table (where it is not a primary key – instead it is called a foreign key). Thus the employee number may be recorded on a sales invoice, to link that sale to a particular employee. This is known as creating a relationship.

Library example

Consider a library using a relational database to keep track of books in stock, borrowers, books that have been borrowed, and details of fines.

- For each **book** you need to keep track of author, title, publisher details...
- For each **borrower** you want to record the name, address, phone number...
- The BookBorrower table exists to allow for the situation that one borrower may take out more than one book, and one book may be taken out by more than one borrower; so there is not a one-to-one relationship between borrowers and books. Borrowers may take out several books at a time (or none), which may have different return dates, and borrowers are charged fines on any books that are overdue.



Some important Access data types & field sizes

Data Type	Field size
Text / Memo	Text holds up to 255 characters, which should be plenty for most applications. Memo uses a large block of storage (over 65 000 characters) which most of the time is likely to stay empty, so avoid this data type.
Number	Byte : positive whole numbers up to 255, does not allow much room for expansion, use with care. Integer : WHOLE NUMBERS up to about 32 000, does not allow decimal points. Long integer : WHOLE NUMBERS up to about 2 billion, does not allow decimal points but for e.g. invoice numbers this should be enough to see you through the lifetime of your business. Single : DECIMAL numbers with up to 7 decimal places. Double/decimal : up to 28 decimal places, requires more storage and not likely to be needed in a business application so avoid them. <i>NOTE: when you use percentages, 50% is stored as 0.5, so it cannot be an integer, it must be size single.</i>
Autonumber	Is a useful feature whereby the computer allocates sequential Long Integer numbers to each record, so you don't have to remember what numbers have already been used. Very good for invoice numbers, or for identifying sequential transactions.
Date/time	Allows you to calculate the number of days between two dates, eg from date of order to date of delivery.
Currency	Limited to 2 decimal places so it avoids rounding errors in calculations.

Test yourself: What data type and field size would you use for

- A customer's ID number?
- A customer's age?
- The date of purchase?
- The quantity of goods in stock?
- A customer's outstanding balance?
- The total VAT charged on a purchase?
- The percentage interest charged on overdue accounts?

Question: How does a database differ from a spreadsheet?

(tables & relations; data types, input masks, validation rules; queries, forms & reports)



Example of a database table and its structure

To give you an idea of how a database table is structured, open the **Motor step1** database (on the G drive or on your training CD). Find the **Car** table in the left pane, and double-click to open it.

RegNumber	Make	Model	Colour	Year of first	Price	Kilometers	Sold
CA 100-000	VW	Polo	White	2006	R 89,000.00	21000	Yes
CA 110-103	Mazda	626	Silver	2003	R 73,000.00	140000	No
CA 110-184	Mercedes	200	Black	2001	R 175,000.00	140000	No
CA 110-645	BMW	318	Maroon	2003	R 88,000.00	144000	No
CA 111-111	Toyota	Yaris	White	2006	R 86,000.00	21000	No
CA 120-691	BMW	318	Silver	2003	R 95,000.00	146000	No
CA 120-820	Nissan	Skyline	Yellow	2002	R 75,000.00	145000	Yes
CA 130-234	Renault	Clio	Silver	2003	R 67,000.00	150000	Yes
CA 130-340	Ford	Fiesta	Green	2001	R 60,000.00	156000	No
CA 130-756	Ferrari	380GTB	Silver	2002	R 170,000.00	150000	Yes
CA 140-194	Nissan	Skyline	Black	2001	R 71,000.00	160000	No
CA 140-234	Nissan	Tiida	Silver	2002	R 65,000.00	150000	No
CA 155-555	Nissan	Skyline	Black	2001	R 72,000.00	230000	No
CA 160-987	Toyota	Yaris	Brown	2002	R 66,000.00	160000	No
CA 166-666	BMW	318	Ivory	2001	R 66,000.00	290000	No
CA 170-432	Renault	Megane	Tan	2002	R 75,000.00	160000	Yes
CA 177-777	Toyota	Yaris	Ivory	2001	R 61,000.00	350000	No
CA 180-098	Mazda	323	Brown	2002	R 62,000.00	160000	No
CA 190-564	Freelander	4.2	Grey	2001	R 95,000.00	175000	No
CA 199-000	Nissan	Skyline	Green	2001	R 75,000.00	210000	No
CA 199-564	Toyota	Corolla	Grey	2001	R 100,000.00	175000	Yes

Use the **View** icon at the top left of your screen to change to *Design* view. Instead of seeing rows of *Car* data, you'll see the different data *fields* that exist in the table.

Field Name	Data Type
RegNumber	Text
Make	Text
Model	Text
Colour	Text
Year	Number
Price	Currency
Kilometers	Number
Sold	Yes/No

As you click on each field, you'll also see its *Properties*, which are used to specify rules, restrictions and formatting requirements for that field.

Field Properties	
General Lookup	
Field Size	Integer
Format	
Decimal Places	Auto
Input Mask	0000
Caption	Year of first licensing
Default Value	
Validation Rule	> = 2000
Validation Text	No models older than 2000 bought or sold
Required	Yes
Indexed	No
Smart Tags	
Text Align	General



Tables

Creating a database table

In your spare time, you've started a beadwork business from your home, and want to set up a database to keep track of your products, customers, sales, etc. You realize that this will take more planning and effort than simply using a spreadsheet – can you suggest two reasons why it would be a good idea to invest your time in developing a database system?

- 1.
- 2.

Your first database table will be used to store data about your products. It will have the name **Products**, and will contain three fields called *Product code*, *Description*, and *Selling price*. (When you view the data in this table, each product that you sell will occupy a separate row of the table, and each field name will appear as a column heading.)

Product code is used to identify each different type of product that you sell, and will consist of a 6 character code. The first 3 characters indicate the type of product, and then 3 digits are used as a sequence number. For example

<i>Product code</i>	<i>Description</i>	<i>Selling price</i>
NCK001	40 cm necklace	R 35.00
NCK002	50 cm necklace	R 45.00
BRC001	single strand bracelet	R 20.00
BRC002	multi-strand bracelet	R 40.00
MAT001	beaded placemat	R 60.00

Instructions

1.	Start Access and create a new database using a blank Access database. Save it on your F drive with the name Beads . (It would be a good idea to create folder called "Access" on your F drive, where you can save all your Access exercises.)
2.	Click on <i>Create – Table</i> to create a new table called Products . Use the <i>View</i> icon to select <i>Design</i> view and enter your first field name (<i>Product code</i>). This must be data type text, with a field size of 6. Change the <i>Required</i> setting to be Yes.
3.	The product code will be the primary key for this table. Click the <i>Primary Key</i> icon on your toolbar and check that the key shows on the left of the field name. Do you remember why your table needs to have a primary key?
4.	You want to make sure that all product codes entered by your staff are in the correct format. Enter an input mask of ">LLL000" to ensure that data entered is in upper case, with 3 letters followed by three numbers. (NB. Don't type the quotation marks.) Note: the ">" means that everything entered must become upper case; the "LLL" means that the first 3 characters must be letters; the "000" means that the next 3 characters must be numbers.
5.	Your first field has now been defined – click the Save icon on the Quick Access toolbar, then continue to the next field.
6.	Click on the line below <i>Product code</i> . Your next field has the name <i>Description</i> . It should also be type Text, and you must decide on a suitable field size.
7.	Change the Required setting to Yes, to ensure that a description will always be provided.
8.	Your third (and last) field must have the name <i>Selling price</i> , and should be type currency .
9.	None of your products may be sold below R10.00, otherwise you will definitely end up going out of business!. Enter a validation rule of ">=10", and validation text saying "Minimum selling price is R10". (NB. Don't type the quotation marks.)



10.	Save your Products table again.
11.	Use the <i>View</i> icon to select Datasheet view. Type in the data shown on the previous page. Try entering records that have no numbers in the product code, or no description, or a selling price of R2, to make sure that all your rules are working.
12.	Use the <i>Office button</i> to save your database, before we continue working on it.

Product code	Description	Selling price	Add New Field
NCK001	40cm necklace	R 35.00	
NCK002	50 cm necklace	R 45.00	
BRC001	single strand bracelet	R 20.00	
BRC002	multi-strand bracelet	R 40.00	
MAT001	beaded placemat	R 60.00	
*			

Extending your database table

In the last exercise you created a database table called **Products**, containing three fields called *Product code*, *Description*, and *Selling price*. In Datasheet view, your table should contain five rows of data.

In order to track the sales and profitability of your different products, you want to define a number of different product categories. Initially, these will be *Jewellery* and *Household*; but as your product range expands you may want to add other categories.

Instructions

1.	The existing Beads database should be open. How many tables does it contain?
2.	Click on <i>Create – Table</i> and call the new table Categories . Change to <i>Design</i> view and enter the field name <i>Group</i> . This should be type Text, 20 characters long, and must be a Required field.
3.	<i>Group</i> will be the only field in this table, and so it will also be the primary key . Click the Primary key icon on your toolbar and check that the key shows on the left of the field name.
4.	Click the <i>Save</i> icon and then change to Datasheet view.
5.	In the first record, enter <i>Jewellery</i> , and in the second record enter <i>Household</i> . Save your table and close it.
6.	Every item in your Products table must belong to an existing category. Open the Products table in <i>Design</i> view, and insert a new row below the <i>Description</i> field.
7.	Give your new row the field name <i>Group</i> and for the Data type select <i>Lookup wizard</i> . Make sure the first option is selected (to look up your values in a table) and then click <i>Next</i> .
8.	The table name you will be using is Categories , so just click <i>Next</i> . You want the <i>Group</i> field to be linked to the items in your Products table, so click the arrow sign to select the <i>Group</i> field. The <i>Group</i> field name should be automatically moved to the right of the screen, and then click <i>Next</i> .



9.	Select the <i>Group</i> field to be sorted ascending, and click <i>Next</i> again.
10.	You should see the two possible Group values that were stored in your Categories table. Click <i>Finish</i> and save your table.
11.	Change back to <i>Datasheet</i> view and select the correct category for each product from the drop-down list. Save the table and close it.

Product code	Description	Group	Selling price
BRC001	single strand bracelet	Jewellery	R 20.00
BRC002	multi-strand bracelet	Jewellery	R 40.00
MAT001	beaded placemat	Household	R 60.00
NCK001	40cm necklace	Jewellery	R 35.00
NCK002	50 cm necklace		R 45.00

12.	You have now created two tables in your database. Click <i>Database Tools - Relationships</i> and check how the Categories table is linked to the items in your Products table.
13.	Save your database before we continue. (The result of this exercise can be viewed in the updated database Beads step1 .)

Test yourself:

1. Which of the following would make the best primary key for a database table?

- Your student number
- Your surname
- Your first name

2. What input mask would best suit your South African ID number?

- LLLLLLLLLLLLLL
- 0000000000000

3. What data type would be best for storing a postal code?

- Number
- Text

Exercise

Before we move on to new database concepts, this is an opportunity to check that you've mastered the basics of creating a database table in Access.

A friend of yours works at a video rental store. The owner recently employed an IT "expert" to set up a database for storing details of videos, customers and rentals. Unfortunately, the expert left town before the job had been completed. You have now been asked to take over...

Before you start this exercise, you need to copy the **Video step1** database from the G drive (or from your training CD) to your own F drive. Until you've done this, you won't be able to make any changes to the database.

Follow the instructions below to complete the required database tables. If you don't complete this exercise in class then you should finish it on your own at a later stage.

1.	Start Access, and open the Video step1 database that you copied on to your F drive. It has two existing tables, Video and Rental . Remember that each table in a database stores information about a different business entity.
----	--



	If you see a security warning, you can go ahead and enable content for the database.
--	--



2.	You are going to create a new database table to store customer information, which involves defining all the fields that are needed and rules to ensure data integrity. Start by clicking <i>Create – Table</i> , and change to <i>Design</i> view, which allows you to enter any relevant rules. Name the table as Customer , (Remember that Datasheet view simply gives you a screen similar to Excel, for entering data.)
3.	The first field to be entered is for the customer account number. <ul style="list-style-type: none"> • Give it the field name CustNumber. • Decide on a suitable data type (text, number, etc). • Don't worry about filling in the Description column – that is just to provide more info for your own use. • At the bottom of the screen, enter the correct field size for the number of characters to be used. • Enter a suitable input mask so that users have to enter three letters and then three numbers. • This field is required to always be entered for every record. • Set <i>Allow zero length</i> to be No.
4.	The second field to be entered is for the customer surname. <ul style="list-style-type: none"> • Give it the field name Surname. • Decide on a suitable data type (text, number, etc). • At the bottom of the screen, enter a suitable field size for the number of characters to be used. • This field is also required. Set <i>Allow zero length</i> to be No.
5.	Create the following fields: <ul style="list-style-type: none"> • Initials (name, data type and size) • Address1 (name, data type and size) • Address2 (name, data type and size) • Postcode (NB Is this text or number? Also use an input mask) • Phone (NB Is this text or number?) • Units (name, data type, and be careful of the field size option)
6.	Identify which one of your fields is the primary key (i.e. different for every customer). Click on the field that is the primary key, and then click the primary key icon on the toolbar. A little key symbol should appear on the left of the field name.
7.	Save your table and close it.
8.	Open the Customer table again (you now want to be in datasheet view, not design view) and add the data that is shown at the end of this exercise.
9.	Test your database table design: <ul style="list-style-type: none"> • Try deleting the surname for Davids: a record with a blank surname should not be accepted, since this is a required field. (Put the surname back in to complete the record.) • Try changing the customer number for Canter to CANT01: if your input mask is correct, it should refuse to allow the letter T instead of 0. Leave it as CAN001.
10.	Save the Customer table and close the database. Well done!

CustNumber	Surname	Initials	Address1	Address2	Postcode	Phone	Units
ADA001	Adams	A	21 Main Road	Rondebosch	7700	0216892435	17
BES001	Bester	B	3 New Haven	Rosebank	7700	0216981234	2
CAN001	Canter	C	102 Forest Hills	Mowbray	7700	0216689436	25
DAV001	Davids	D	29 Westerford Road	Newlands	7700	0216984343	51
EDW001	Edwards	E	18 Smuts Avenue	Claremont	7700	0216984321	0
FES001	Fester	F	18 Hart Street	Claremont	7700	0216983412	0



Relationships

Relationships between database tables

When using a relational database such as Access, the different sets of data that are stored in separate tables, are linked to each other through common fields. Thus information about the doctors in a medical practice could be stored in one table, information about the patients could be stored in another table, and a third table with consultation information would link each patient to a specific doctor for each consultation.

For example

Table	Fields
DOCTOR	DocNumber, DocSurname, DocPhone, DocAddress
PATIENT	PatientNumber, PatientSurname, PatientFirstName, PatientAddress, PatientDateOfBirth
CONSULTATION	ConsNumber, ConsDate, DocNumber, PatientNumber, ConsCharge, ConsReason

A field whose value uniquely identifies each record in a table, is referred to as a *primary key*. This is essential to facilitate data retrieval from the database, especially when multiple tables are involved.

When the primary key from one table is included in a second table in order to form a relationship between them, it is called a *foreign key* in the second table. Thus DocNumber would be a primary key in the DOCTOR table, and a foreign key in the CONSULTATION table.

Referential integrity ensures that correct relationships are preserved when records are added to or deleted from a table in the database. For example, when a patient visits a doctor and a new record is added to the CONSULTATION table, the patient must have a valid PatientNumber in the PATIENT table (otherwise who will the account be sent to?). The consultation record must also be completed with a valid DocNumber, otherwise if there is a subsequent problem, nobody will know which doctor was responsible.

Cascading of updates and deletes should not be applied unless you have a good reason to do so. For example, by cascading deletes, if a doctor were deleted from the DOCTOR table after leaving the practise, all records of the consultations he held would also be deleted. In some instances this feature can be useful (when an employee leaves a business, details of his/her car registration are no longer required), but in general, you should avoid cascading deletes unless you have considered the potential consequences.

Now we're going to create some relationships:

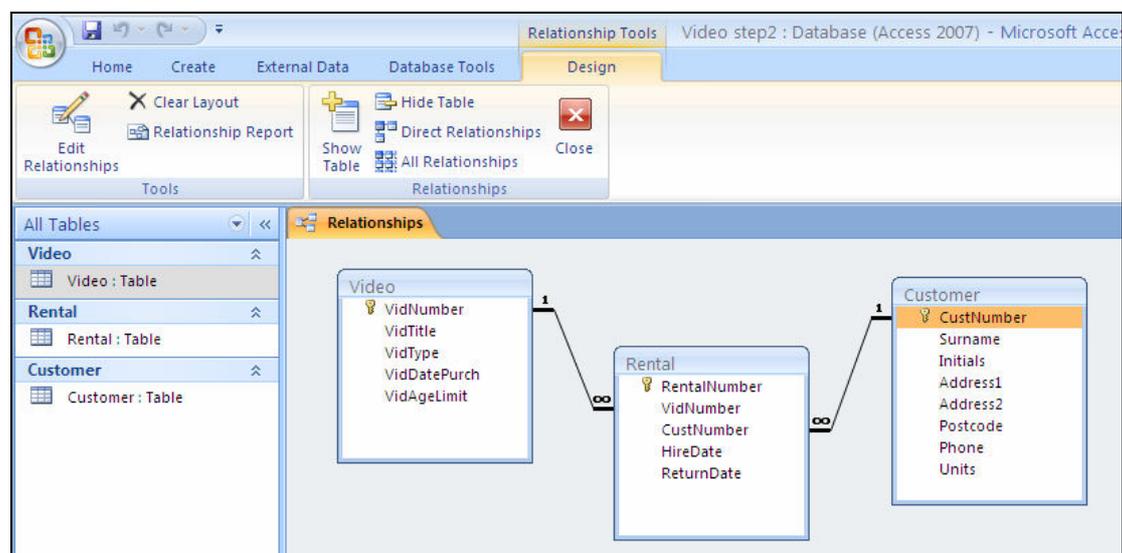
The **Video step1** database that you worked on for the last exercise has a **Video** table, a **Customer** table and a **Rental** table. Re-open the database and answer the questions below:

1.	Which field is the primary key in the Video table? What is its data type & field size?
2.	Which field is the primary key in the Customer table? What is its data type & field size?
3.	Which field is the primary key in the Rental table? What is its data type & field size?



Now you are going to create relationships between these tables, so that you can ensure data integrity, as well as being able to retrieve information across all three tables.

4.	Look at the screen shot that follows and make sure that you understand how these three tables can be related by their common fields.
5.	On the toolbar, click the <i>Database Tools – Relationships</i> . You should see a window with the Video and Rental tables already related to each other.
6.	Any time that a video is rented, it must only be booked out to an existing customer. This requires a <i>relationship</i> (link) between the Customer and the Rental tables.
7.	Click the <i>Show Table</i> icon on the toolbar to add the Customer table to your window.
8.	Create the relationship by holding down the mouse click on the <i>CustNumber</i> field in the Customer table, and dragging it to the <i>CustNumber</i> field on the Rental table.
9.	In the window that pops up, make sure that referential integrity is enforced (why?). If you are not able to enforce referential integrity, then one of your CustNumber fields is incorrect.
10.	Save the Video database and close it. (The solution to the exercise is provided as Video step2 .)



Exercise

The last time that you worked on the **Beads** database, it contained two tables: the Products table and the Categories table. It has now magically acquired a Sales table, containing records of product sales for the past month. For this exercise, you are going to practise setting field properties and creating relationships.

To ensure that you have this latest version of the Beads database, copy **Beads step2** from the G drive or from your training CD on to your F drive.



Open the **Sales** table in Design view and see the fields that it contains:

SaleNumber is a sequential number generated by Access.

SaleDate is the date of the sale.

Customer is used to store customer account numbers for credit sales. (We will use account number CASH01 for cash sales.)

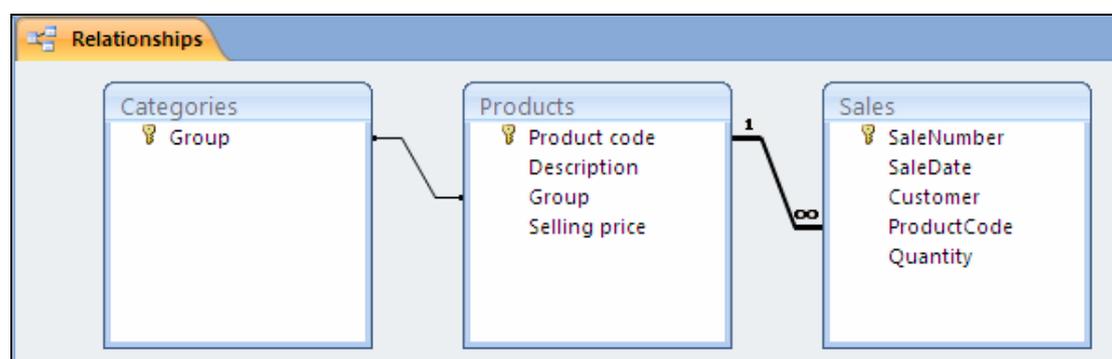
ProductCode refers to the product code that is the primary key in the Products table.

Quantity is the quantity of that particular product that has been sold.

Note that *ProductCode* occurs in both the **Products** table and the **Sales** table. This is your clue that you will use it to create a relationship between the two tables.

- When you create the relationship, should you enforce referential integrity?
- When you create the relationship, should you 'cascade deletes'?

1.	Which field is the primary key in the Sales table? What is its data type & field size?
2.	For the <i>SaleDate</i> field, enter a Default Value so that the Sale Date will automatically be the current date. Tip: use the function =date()
3.	For the <i>Customer</i> field, enter an Input Mask so that the customer number will always be four letters followed by two numbers (e.g. CASH01)
4.	Make sure that the <i>ProductCode</i> field in the Sales table is the same Data Type and Field Size as it is in the Products table. Correct it if necessary.
5.	For the <i>Quantity</i> field, enter a Validation Rule to ensure that the quantity is always greater than zero. There should also be Validation Text to say "Please enter a sale quantity". This message will be displayed if a quantity of 0 is entered.
6.	Exit the Sales table and save your changes.
7.	You are now going to create a relationship between the Products table and the Sales table. i.e. every sale will relate a specific product.
8.	Go to <i>Database Tools – Relationships</i> . Click <i>Show Tables</i> and add the Sales table to your window.
9.	Create the relationship by holding down the left click on the <i>ProductCode</i> field in the Product table, and dragging it across to the <i>ProductCode</i> field in the Sales table.
10.	In the window that pops up, make sure that referential integrity is enforced (why?). If you are not able to enforce referential integrity, then one of your ProductCode fields is incorrect – check its data type and field size in both tables.
11.	Close the Relationships window and save your changes.





Queries

Introduction to Queries

Access provides several ways to view, change and extract data from your database. Most of the time you will do this by using *queries*, which are short “programs” requesting that data be retrieved from, added to, or modified in a table. You will usually create a query because data stored in one or more tables is needed in a form or a report. Rather than storing the results of a query, Access stores the request that was processed in order to extract the data.

Two basic types of queries that can be used are:

- **Select queries** are used to view subsets of the tables from which the data is selected, providing a different “view” of a table.
- **Action queries** make changes to many records in a single operation. There are four types of action query: *delete* and *update* queries remove or modify records in an existing table; *append* queries add new records to existing tables, and *make table* queries create entirely new tables by generating new records.

Queries can be expressed in two different ways: Query By Example (QBE) or Structured Query Language (SQL). Most queries written in one format are convertible to the other, but for this course you will only use QBE.

Using Query by Example

QBE is the default format for creating queries in Access, based on the following details:

Field is used to specify the name of a field in an underlying data source.

Table specifies the data source that contains the field above.

Sort will return records sorted by the field above, or unsorted.

Show indicates that field data should be shown in the query result. If unchecked, the field can be used to sort, but will not itself be shown.

Criteria contains a string which includes any number of criteria to be matched by the field above, separated by AND.

Or is used to extend the criteria for the field, independently of the preceding string.

Field:					
Table:					
Sort:					
Show:	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Criteria:					
or:					

Don't worry - you'll quickly become familiar with these options after you have created your first few queries!

Creating a Select query

To create a query, you will open your Access database, and then click *Create – Other – Query Design*.

Select the **data tables** on which your query will be based by using the *Show Table* dialog box. Select each table that you want to use in the query before closing the dialog box. If you have selected multiple tables, make sure that the relationship between them is displayed.

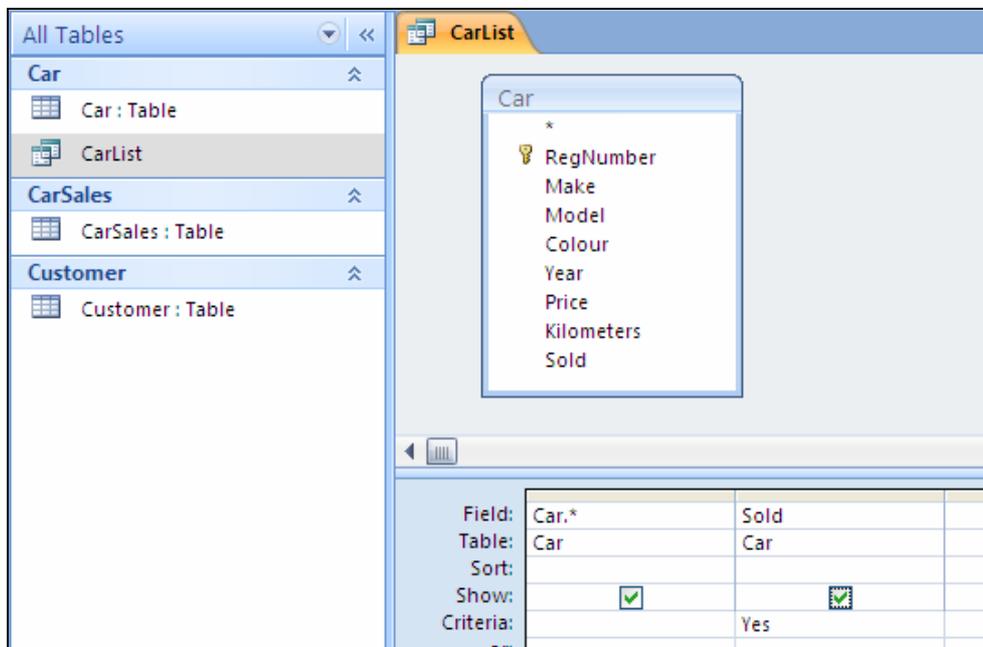
Use the **design grid** to specify the fields, criteria etc to be used, and save the query with an appropriate name. The records which will be included in the query result can be viewed by clicking the *View* icon on the toolbar, and the actual result of the query is obtained by clicking the *Run* icon.





Let's try creating a select query:

	Check whether you already have a copy of the Motor step1 database on your F drive. If not, then copy it from the G drive (or from the CD) to your home folder.
1.	Open the Motor step1 database. We are going to begin by making a very simple query that just shows all the cars in the Car table. Click <i>Create – Other – Query Design</i> . Add the Car table and then close the <i>Show Table</i> window. (Note that you can drag the borders of the various screen elements to adjust their sizes.)
2.	In the Car table, double click on the asterisk (*) in the first row. This is used to represent all the fields in the table. <i>Car.*</i> will appear as a field in the query grid.
3.	On the toolbar, click the <i>Run</i> icon, and you will see a list of all the data records in the table. How many records are there?
4.	Now we want to see only the cars that have been sold. Go back to Design View, and double click on the <i>Sold</i> field in the Car table. It will appear in the grid next to <i>Car.*</i> . In the Criteria row of the <i>Sold</i> column, type the word <i>Yes</i> . Run the query and this time only the records for sold cars will be shown. How many records are there?



5.	Save your query, and give it the name CarList . Any time that you exit a query, Access will check whether you have saved it, and if not then it will remind you to do so. Close the query window and check that your CarList query is still there.
6.	Run the CarList query again and notice that the rightmost column, which was used to select the <i>Sold</i> cars, doesn't really serve any informative purpose and could be confusing to a user. Fix this by going to Design view and clicking to uncheck the Show box in the <i>Sold</i> column. Run the query again to see how the output has changed.
7.	Now we want to sort the sold cars in descending order of price. In Design View, add the <i>Price</i> field to the query grid. Use the <i>Sort</i> row to select Descending, and click to uncheck the Show box. Run the modified query.
8.	Save your query again (still with the same name) and then test your understanding by changing the query to show only <i>Unsold</i> cars in Ascending order of <i>Price</i> . Run the query, then close the query and save it.



Most of the time you won't want to show all the fields in your table. In fact, the first step in creating a query is to decide which tables and which fields you need to use. Then add the required tables to the new query, before selecting your fields.

1.	Create a new query based only on the Car table, which includes the car registration number, make, model, year, price and sold status. Show all of these fields. Enter the criteria needed to show only <i>Unsold</i> cars. Run the query and make sure that it works correctly. Save the query with the name Unsold .
2.	You have a customer who wants to buy a Toyota Yaris. In the same query, add criteria of Toyota for <i>Make</i> and Yaris for <i>Model</i> . Sort the records in Descending order of <i>Year</i> , and run the query.
3.	Close the query and save your changes (still with the same name). You should now have two queries in the database, called Carlist and Unsold .

All Tables		Unsold					
	RegNumber	Make	Model	Year of first	Price	Sold	
Car	CA 300-000	Toyota	Yaris	2006	R 85,000.00	No	
Car : Table	CA 111-111	Toyota	Yaris	2006	R 86,000.00	No	
CarList	CA 740-000	Toyota	Yaris	2004	R 66,000.00	No	
Unsold	CA 160-987	Toyota	Yaris	2002	R 66,000.00	No	
CarSales	CA 177-777	Toyota	Yaris	2001	R 61,000.00	No	
CarSales : Table	CS 560-000	Toyota	Yaris	2000	R 62,000.00	No	
Customer	*						
Customer : Table							

A bit more theory about queries:

- When you save a query, you don't save the actual data records, you just save the instructions about what data you want to retrieve. So if you run the same query on two different days, you could get two different sets of results, depending on what cars have been sold in the meantime. This makes queries very useful to managers who want to be able to see up-to-date information at the press of a button.
- You have been making select queries, which display selected records from the database. However, if a user makes changes to the data that is displayed by the query, the same changes will be stored in the underlying database table. Sometimes you might want users to be able to see database records but not be able to change them – in that case you would probably use a form to display the query results, and then change the properties of the form so that the data can't be edited.

Now let's try making a slightly more advanced query:

	<i>Your last customer enquired about buying a Toyota Yaris. Now you have a customer looking for a Nissan Skyline. Rather than changing the criteria for every customer, we are going to make something called a "parameter query" (a type of select query).</i>
1.	Make a copy of the Unsold query, by selecting it and then using a right-click to copy and paste. Call the copy SelectCar .
2.	Open the SelectCar query in Design view. We are going to change the criteria so that instead of specifying one make and model, Access will ask the user what make and model are wanted. You do this by enclosing a question in square brackets in place of the criteria, and the answer that is typed in will become the criteria when the query is run. Change the criteria for <i>Make</i> so that instead of showing "Toyota" it shows [What make?]. Change the criteria for <i>Model</i> so that instead of showing "Yaris" it shows [What model?].



3.	When you run the query, it should ask for a make and then for a model. Enter "Nissan" for the make and "Skyline" for the model. Can you see how this function makes a query more useful?
4.	Close your query and save it.

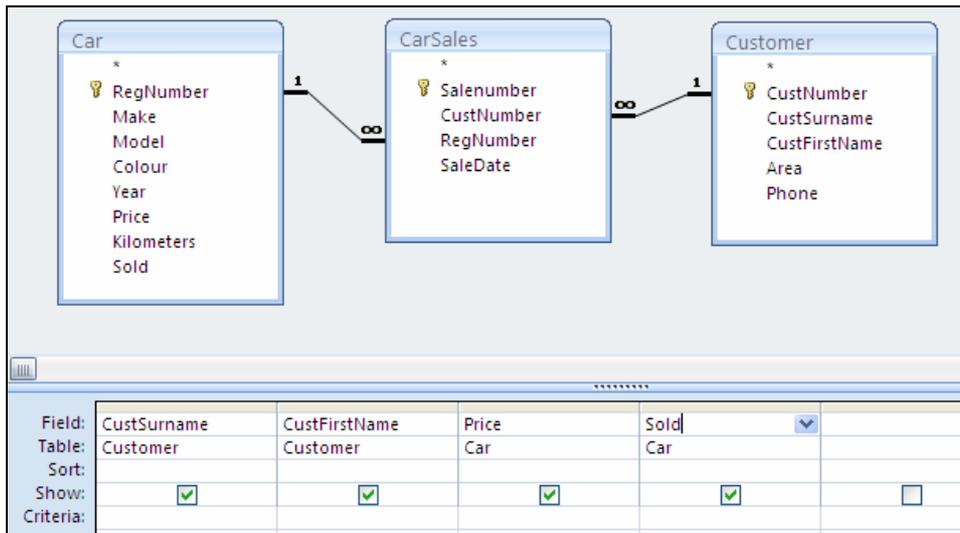
More Select queries

Using multiple tables

The real value of queries lies in their capability to relate data from different data sources. Inside your Access database, the Relationships window shows the relationship between tables by means of a line marked at each end with a "one" or a "many" symbol. If no relationship already exists between the two tables to be used in a query, you can create one by dragging the key from one data source to another. This will not affect relationships between the tables outside of the query.

Let's try creating a select query using multiple tables:

1.	The Motor database should be open. You are going to make a query that shows the customer name and the car price for all the cars that have been sold. First check in the <i>Relationships</i> window under the <i>Database Tools</i> tab, to see which tables you will need to provide this data. <ul style="list-style-type: none"> • Customer name comes from the Customer table. • Car price and Sold status come from the Car table. • BUT there is no way of relating these two tables without using the CarSales table. (Please check this out for yourself) • So you are going to need all three tables in your query.
2.	Under the Home tab, click <i>Create – Other – Query Design</i> . Add all three tables and then close the <i>Show Table</i> window.
3.	Double click or drag the fields for customer surname, customer first name, price and sold status to the query grid.
4.	Run the query and check that you get 20 records. Save the query with the name Sold .



Test yourself: Can you sort your query results in descending order of Price?



Calculated fields

Calculations can be performed on either numeric or text data before it is used in a query.

For example, instead of showing the firstname and surname as separate fields, you might want to combine them into a single field (surname followed by a comma followed by the first name).

Start by giving a name to this new combined field, e.g. Fullname. The name must be followed by a colon (:) to indicate that is a calculated field and won't be found in one of the tables.

Then specify how the new field must be calculated. If you want to refer to fields that are already in tables, then their names must be enclosed in square brackets.

For calculations using text: join different text fields together by using the & symbol. To include additional characters, enclose them in quotation marks.

e.g. Fullname: [CustSurname] & ", " & CustFirstName

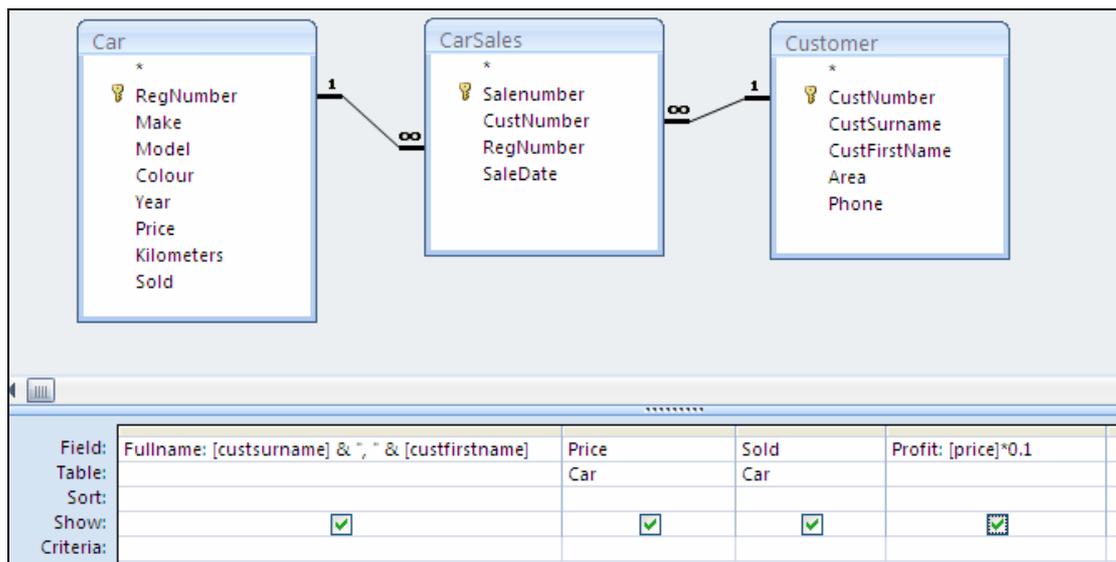
1.	Make a copy of the Sold query by using cut & paste. Give the copy the name Fullname .
2.	Replace the <i>CustSurname</i> field with a calculated field as given above. Delete the <i>CustFirstName</i> field. Check that your query works!

For calculations using numbers: use the usual arithmetic operators + - * / (but don't try to include any text or text fields!)

e.g. To calculate Profit at 10% of the selling price:

Profit: [Price] * 0.1

3.	Add another field to the Fullname query, to show the profit made on each sale.
4.	Run the query and then save it.



Aggregate functions

In order to make calculations based on more than one record, Access provides a number of additional functions which can compute values such as totals and statistical variances.

To make use of the aggregate functions, click the Totals [Σ] button on the toolbar while in a query's Design view, and a new row will appear in the QBE grid entitled "Total:". If you click in the Total field and then click the drop-down list, the functions available for the field in the top row of the grid will appear.

Group By uses a field to define breaks for another function on another field.

Sum calculates the sum of all values in a given field.

Avg computes the average of all values in a given field.

Min finds the minimum value of a given field across all records.

Max finds the maximum value of a given field across all records.

Count counts the number of records in the data source.

StDev determines the standard deviation of a given field across all records.

Var determines the statistical variance of a given field.

First returns the value in the first record.

Last returns the value in the last record.

Expression is used when the Totals row displays a calculation on some other field in the query.

Where is used when the Totals row is being used for another field which has criteria applied to it.

These functions include in their calculations only records which have met any selection criteria, and fields which contain non-null values.

1.	Create a new query, that you will use to show the average selling price for each make of car. This will be a new query using only the Car table, and using the fields <i>Make</i> and <i>Price</i> .
2.	Once you have added these fields to your query grid, click the <i>Totals</i> button and in the Totals row, select the Avg option for the <i>Price</i> field. Run the query.
3.	Add the <i>Price</i> field to the query a second time, and in the Totals row, select Max.
4.	Save your query with the name Average .

Field:	Make	Price	Price	
Table:	Car	Car	Car	
Total:	Group By	Avg	Max	▼
Sort:				
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Criteria:				

One more example:

This is a complicated query, using multiple tables, a calculated field and an aggregate function!

We want to find out the total profit made from each of our customers, where profit is 10% of the selling price. For this you will need the customer name from the Customer table, the profit amount calculated from the price in the Car table, plus the CarSales table to connect the records in the Customer and the Car tables.



Once you have worked out how to do this, save your query with the name **Profit**.

CustSurnam	CustFirstNai	profit
Benson	George	13700
Clooney	George	24500
Cole	Andy	6700
Dlamini	Justin	9600
Grant	Hugh	47000
Johnson	Michael	93900
Lee	Spike	18500
Small	James	37100
Smith	Derek	8900
Smith	Will	19500
Snipes	Weslie	12000
Winger	Debra	29500

Well done – you now know all that you need to know about Select queries! (You can find the solutions to these queries in **Motor step2**.)

Action queries

Up to now, all the queries you have done have been **Select queries**. Select queries show you the database records and fields that you have selected (and any calculated fields that you may have defined), but they do not change any of the values in your underlying database tables.

Action queries are used to

- copy fields or records to a new database table (make table query)
- delete records from database tables (delete query)
- change values in database tables (update query)
- add records from another database table (append query)

NB. The changes made by running an action query are permanent – you can't click the Undo button to reverse them.

To explore these different types of query, we will use the **Motor step2** database. Assume that you have decided to reduce the price of all Toyotas by 10%. You would use an Update query to change their prices. But if you make a mistake when you run the query, you can't reverse it! So before you do the Update query, you would need to run a MakeTable query, and make a backup table containing the current Toyota prices.

MakeTable queries

For a Make Table query, you will only include fields that you want to be copied to the new table, or that are going to be used to specify criteria. (Note: if you want to copy a complete table, then you can just copy & paste it. But if you only want to copy a few records or fields from a very large table, then that wouldn't be practical.)

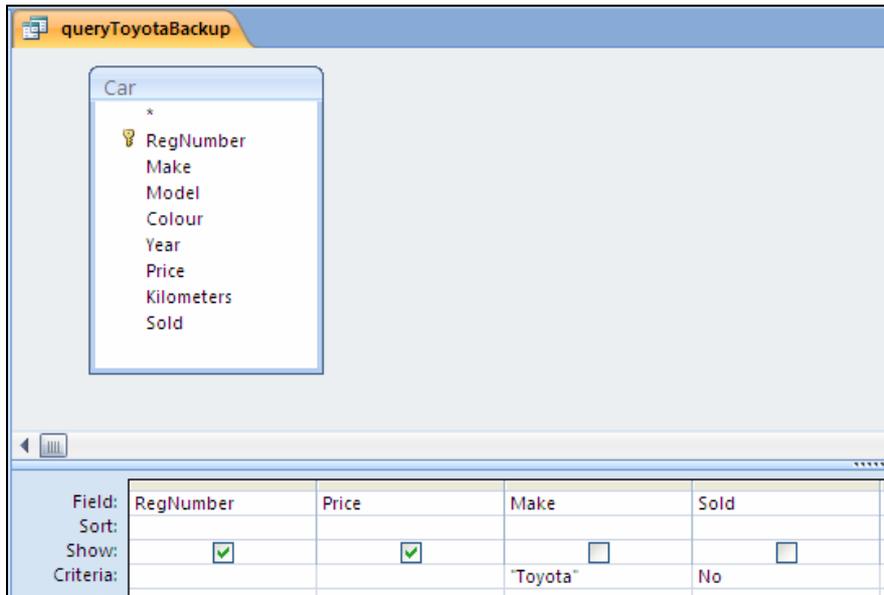
What fields are you going to copy to the new table? It won't be sufficient to only copy the car prices on their own, because you won't know which price belongs to which car! So you need to include the primary key (i.e. registration number) as well as the price.

What records are you going to copy to the new table? If you are going to reduce car prices, you can't reduce the price of cars that have already been sold. So you need to use the Sold field with criteria "No". Since you are only going to reduce the price of unsold Toyotas, then you also need the Make field with criteria "Toyota".



Check that you understand: to make a (useful) backup table with the current price of unsold Toyotas, you will need to use a MakeTable query with four fields: RegNumber, Price, Make and Sold.

1.	Start off by making the usual <i>Select</i> type query with the four fields you have identified, and put in the correct criteria for Make and Sold. Save it with the name queryToyotaBackup . Run the query to check that it only shows 10 unsold Toyotas. (Remember, as long as it is a Select query, it won't make any changes to the database.)
2.	In <i>Design</i> view, click the icon for the query type of <i>Make Table</i> . The new table will be called tableToyotaBackup , in the current database.
3.	Uncheck the <i>Show</i> boxes for Make and Sold – you don't need a backup of these fields, you are only using them to specify criteria.
4.	Run the query. Be warned: Access will not tell you that it has successfully made the new table. But if you don't get an error, then the table will have been made. Close the query and save the changes.
5.	The table you have just made will appear in the pane on the left side of the screen. Open it – it should only have registration numbers and prices (for 10 unsold Toyotas). Now it's safe to go ahead and change their prices, knowing that you have a backup if anything goes wrong.



Just to make sure that you've understood the whole process, try doing another MakeTable query that will make a backup of only the registration number and price for all the cars that have already been sold. Do not include the Sold field in the new table. Call the backup table tableSoldCars and call the query querySoldCars.

Delete queries

Delete queries are set up in a similar way to MakeTable queries. The difference is that you should only include fields that will be used to specify criteria for records that you want to delete from the database. Any records in the database table that meet these criteria will be totally and permanently deleted.

e.g. If you were going to scrap all unsold cars older than 1990, then you would first make a backup table containing their details. (Hopefully you know how to do that now.) After that you would set up a Delete query to remove unsold cars older than 1990 from the Car table.



The Delete query would have only two fields: Year, with the criterion "<1990", and Sold, with the criterion "No". When you run the Delete query, all records for cars meeting **both** criteria (i.e. older than 1999 and unsold) will be permanently deleted.

You won't be asked to run any Delete queries in this course – it's too risky if you make a mistake! But you should still understand the principles behind a Delete query.

Update queries

Update queries are a type of action query used to permanently change the values in a database table. Since you can't reverse these changes by clicking the Undo button, it's a good idea to first make a backup of the original table before running any update query.

Any field that you are going to update will be one of the data types that you already know, such as a number, or text, or date. The change you make to that field must be in keeping with its data type – you can't multiply a text field or add letters to a number field. This may seem obvious, but watch out for possible confusion when using fields such as postal codes that may look like a number but be defined as text.

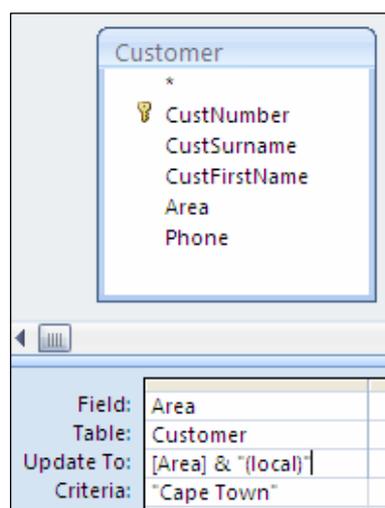
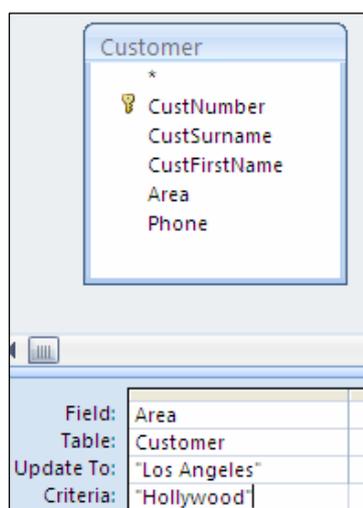
When you run an Update query, the previous value of the field that is shown in the query grid will be replaced by the new value given in the "Update To" row. Before you start, you need to be clear as to: *What field do you want to change? What must it be changed to? What criteria should be used if only some of the records need to be changed?*

This is a good time to revise the most frequently used operations – they are the same ones as you used in calculated fields when doing select queries. The difference is that in a calculated field, you had to start by giving the new field its own unique name. In an update, you are changing the value of a field that already exists, so it doesn't need a name, just the instructions about what calculation must be done.

Text fields

Here you are most likely to want to replace one text string with another text string, or else add some new text characters to an existing string.

1. If you wanted to replace the city name "Hollywood" with the name "Los Angeles", you would use the criteria "Hollywood" for the field holding the city name, so that only records for Hollywood are changed, and then Update the value to be "Los Angeles". Easily done.
2. If you wanted to add "(local)" to the Area field for all Cape Town customers, your criteria would be "Cape Town", and you would add the new text in quotes using an & sign.

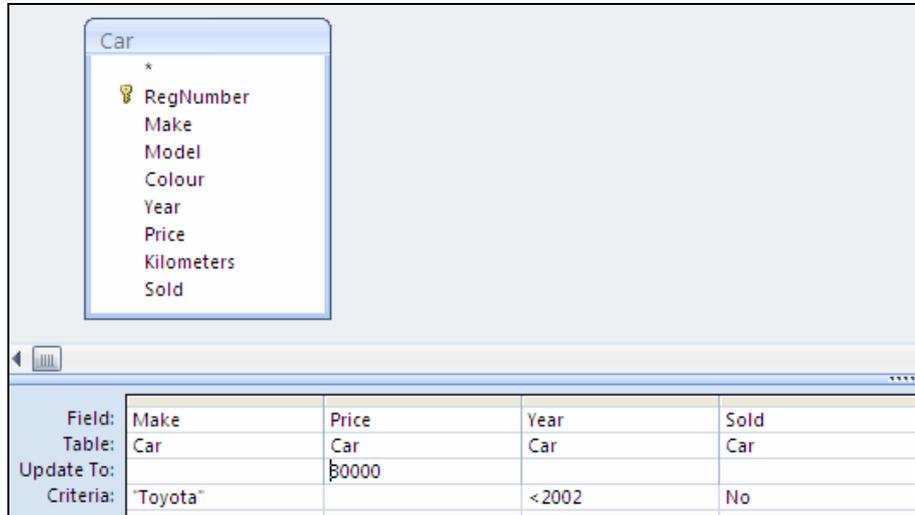


Number fields

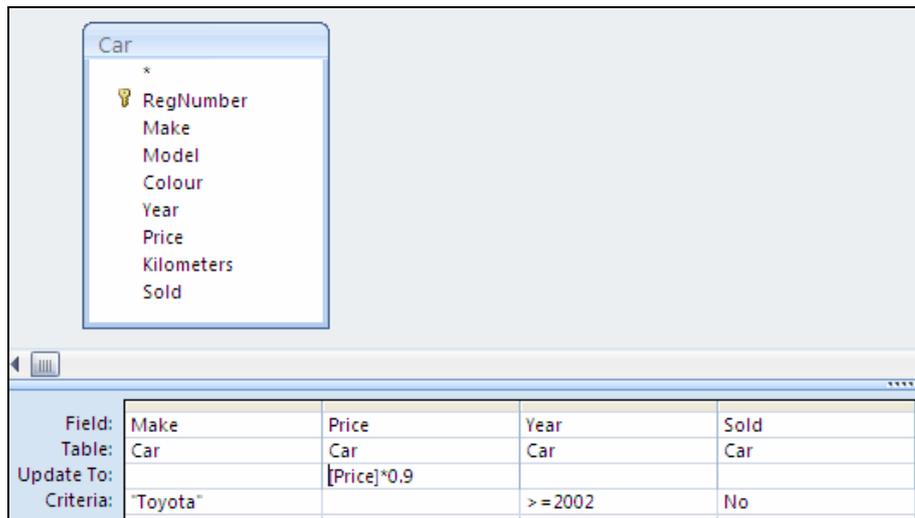
Here you would use arithmetic operations in the Update To row. If you want to replace an existing value with a new value, you need to first consider how the new value is calculated.



1. If all Toyotas prior to 2002 are to be priced at R30 000, you would enter 30 000 in the "Update To" row of the Price field. Remember that this should apply to Unsold Toyotas only, with a year less than 2002. All the selected Toyotas would then have exactly the same price.



2. If you wanted to reduce the price of all 2002 or later Toyota models by 10%, then you would need to use the current price as part of your calculation by putting it in square brackets, i.e. [Price]*0.9.



Try it yourself:

In the **Motor step2** database, create an Update query that will reduce the price of all unsold Nissans by 15%.

1.	Start by making a Select query, and include the fields for Make , Price and Sold .
2.	Then add the criteria needed to specify that the make is Nissan and that Sold is No .
3.	Now change the query type to be <i>Update</i> .
4.	Finally add the calculation for the new Price value in the <i>Update To</i> row.
5.	When you run the query, you will be warned that the values are about to be changed before



the changes are actually made.

Remember that it's always a good idea to make a backup before you start with any sort of action query - just in case you make an error in the query (for example, you forget to specify unsold cars only).

You have now learned how to make **select queries**, using multiple tables, including criteria, allowing the user to enter a parameter value, sorting the query output, and hiding fields that you don't want to be displayed. You can display new output columns based on existing data, and can include functions such as average, sum, maximum etc.

Using **action queries** you can make a backup table with selected fields and records, you can delete specified records from a table, and you can update field values. And you'll be relieved to know that's all we're going to cover with queries! (The database **Motor step3** will show you how all these queries have been constructed.)



Forms

Creating a form to view your data

A friend wants to get involved in your bead business, and is going to help by answering enquiries, quoting prices and taking orders. You want to provide an easy way for him to view product details without allowing him to change the data that is already on the system. You will do this by using a form that allows him to view existing data but not change it.

Instructions

1.	Start Access and open the Beads step3 database. Click on the <i>Create</i> tab.
2.	Click on <i>More Forms</i> and select <i>Form Wizard</i> . (This option gives you more control than the one-step <i>Forms</i> icon, but is a whole lot easier than using <i>Design</i> view.)
3.	In the first box, select the table called Products . Under <i>Available fields</i> click the double arrowhead, which will move all the field names to the right hand pane. Then click <i>Next</i> .
4.	Choose Columnar layout and click <i>Next</i> .
5.	Choose Office as your style and click <i>Next</i> .
6.	Click <i>Finish</i> to see your form. Save it with the name Products .

7.	You want your friend to only be able to view your existing products, and not change their details. To enforce this, you need to change to <i>Design</i> view.
8.	If you don't see the Property sheet at the right of your window, then click the <i>Property Sheet</i> icon. Select the <i>Data</i> tab.



9.	Click the little square at the left of the ruler bar, to view properties for the form as a whole. Under data properties, change <i>Allow Additions</i> to be No, <i>Allow Deletions</i> to be No, and <i>Allow Edits</i> to be No.
10.	Change back to Form view, and see how you can page through your product list without being able to change any of the values.
11.	Unfortunately your friend doesn't like this layout – he says that it takes too long to find a particular product. Luckily you can change the format so that several records show on the same screen.
12.	In Design view, click once again next to the ruler bar. Change the <i>Format</i> properties so that <i>Allow DataSheet View</i> is Yes, and <i>Default View</i> is Datasheet.
13.	Back in Form view, you can resize your form and your columns so that it is easy to read. Save the form again, and then close it.
14.	Note that although this <u>looks</u> the same as your original data table, it only provides a view of the data in the table and doesn't allow any changes to it. If you want to make any changes, you will have to do it in the original table and not by using this form.



Product	Description	Group	Selling price
BDG001	beaded badge	Jewellery	R 40.00
BRC001	single strand bracelet	Jewellery	R 20.00
BRC002	multi-strand bracelet	Jewellery	R 40.00
EAR001	earrings	Jewellery	R 30.00
MAT001	beaded placemat	Household	R 60.00
NCK001	40cm necklace	Jewellery	R 35.00
NCK002	50 cm necklace	Jewellery	R 45.00

It doesn't make much sense to create a form that prevents changes, when anybody can just go to the underlying table and make any changes they want – so later on we will add a menu that limits the users of your system to doing only what you allow them to do!

Creating a form with a subform

The Beads database contains three tables: Products, Categories and Sales.

- Which field relates the **Products** table to the **Categories** table?
- Which field relates the **Products** table to the **Sales** table?

Since all three tables are connected, we can combine data from all of them when constructing forms to enter or view data.

First we'll create a form to show the detailed sales for each product. The product descriptions are stored in the Products table, but the quantities sold are stored in the Sales table, so we'll need to use data from both of these tables.

1.	In the Beads database, click <i>Create and More Forms</i> , then select <i>Form Wizard</i> ..
2.	Start with the Products table, and select all its fields, so that they move to the right-hand pane.
3.	Then change to the Sales table, and select all of the fields <i>except for Product code</i> . (You already have that data in the Products table, so there's no point in repeating it on your form.)
4.	Click Next, and make sure that you are going to view your data by Products and not by Sales, as a <i>Form with Subforms</i> .



5.	On the next screen, select datasheet view.
6.	Choose any style that you like (the example shows Office).
7.	Finally, save the form with the name ProductSales , and click Finish.

The top half of the form shows details about one product, and the bottom part of the form lists all the sales of that product. Since there are seven different products, use the Navigation Buttons at the bottom of the window to move backwards and forwards through the different product records and make sure that the sales data changes as well. Note that new sales can also be entered inside each subform window.

Next we are going to make some changes to the appearance of this form. The navigation buttons inside the subform look cluttered, and are easily confused with the navigation buttons for the different products. So we are going to remove the navigation buttons from the subform.

1.	While the ProductSales form is still open, change to <i>Design</i> view.
2.	You should see a ruler bar at the top of the <i>main form</i> , and another ruler bar at the top of the <i>Sales subform</i> .
3.	Click on the small block immediately left of the subform ruler bar.
4.	If the Property Sheet is not displayed, then do a right click and select <i>Properties</i> .
5.	Under the <i>Format</i> tab, change <i>Navigation Buttons</i> to be No .
6.	Return to Form view. If necessary, return to design view and stretch the borders of the subform to show all the fields.
7.	Try adding a few more cash sales for different products to check that it all works correctly.
8.	Save your work, make sure that you are feeling impressed with what you have achieved, and close the database.

If your form doesn't function correctly, then don't panic – you can simply delete the ProductSales form and the Sales subform, and start again. (You can also find examples of all the Beads forms in the **Beads step4** database.)

Advanced forms

Forms can be based on queries as well as on tables. Remember that a query can contain calculated fields or show aggregate functions such as average and sum; so if you base your form on a query then it can show this sort of information. You can even base a form on a parameter query.

Generally you would base a form on a table if you want to enter new data into the table – in which case all the fields should be included. You are more likely to base a form on a query if you want to view only selected data (or records), which may come from multiple tables.

Note that every form has a **Header** and a **Footer** area, which can be used to display information such as a title or a date, that will remain on the form as you scroll through the different records in your database.

- To display text that doesn't change in the header or footer, use a **Label**.
- To display data (from an Access function or a database table) use a **Text box**.

In fact, when the wizard creates a form for you, it does just that: it creates a set of labels showing the field names and text boxes that display the changing data values.

Scenario

Now we're going to revisit the Video database, which as you may remember has three tables: Video, Customer and Rental. Make a copy of **Video step2** to get the latest version of the database, and open the Relationships window to see how the tables are related.

Instructions

1.	You have been asked to construct a form that will reduce the time spent entering and viewing Customer and Rental data. Use the wizard to construct a form and subform based on all the fields in these tables. Call the main form CustomerData .
2.	In Design view, drag the edges of the form and subform wider to allow all the data in the subform to be displayed. Adjust the column widths where necessary. Drag the



picture to the right edge of the main form. Experiment with selecting a group of text boxes by dragging an imaginary square around them and then moving them or widening them. (To move a group of fields, you will first need to right click after selecting them, and then select *Layout – Remove* to detach them from the underlying grid.) Try to end up with something that resembles the example below.

RentalNumber	VidNumber	HireDate	
2	8003	2009/02/21	200
7	7006	2009/03/07	200
8	8001	2009/03/13	
* (New)			

- In Design view, locate the area between the Form Header and the Detail sections at the top of your form. Click on the label that says CustomerData, change it to read "Customer and rental data", and resize the field to fit the text.
- Find the **Text Box** button on the toolbar and create a text box on the right hand side of the header. Type "= now()" inside the text box, so that it will display the current date and time. Click on the label just left of the text box (it probably says Text19) and delete it. View your form to see that it works. As you move through the different customer records, the header information will stay the same.

Your form is looking quite professional – let's add the finishing touches!

- Go back to Design view, and drag the bottom edge of the Form Footer down to expose some space below your form. You are going to add four Command Buttons: one to go back to the previous record, one to move to the next record, one to add a new record, and one to close the form.

Under *Form Design Tools*, click *Control - Button* on the toolbar, and draw your first button in the footer area. (The wizard should appear to guide you – if not, delete the empty command button, then click the magic wand at the right of the controls section and start this step again.) In the *Record Navigation* category, click on *Go to Previous Record*, then click *Next*. Select the *Text* option and finish. Go to form view and see



	what it looks like!
6.	<p>Back in Design view, use the Command Button again to draw another button. In the <i>Record Navigation</i> category, click on <i>Go to Next Record</i>, then click <i>Next</i>. Select the <i>Text</i> option and finish.</p> <p>For the third button, you need to select the Record Operations category, and click on Add New Record. And for the final button you will use Form Operations to Close Form.</p>
7.	If you need to tidying up the alignment of your command buttons, draw an imaginary line around them and then use <i>Arrange – Control Alignment - Top</i> to make them level.

Customer and rental data 2009/04/09 02:58:54 PM

CustNumber: ADA001 Address1: 21 Main Road
Surname: Adams Address2: Rondebosch
Initials: A Postcode: 7700
Phone: 0216892435
Units: 17

RentalNumber	VidNumber	HireDate	ReturnDate
2	8003	2009/02/21	2009/02/22
7	7006	2009/03/07	2009/03/09
8	8001	2009/03/13	
* (New)			

Record: 1 of 3 No Filter Search

Previous Record Next Record Add New Record Close Form

Record: 1 of 6 No Filter Search

If your form looks like the example, and the buttons work, then give yourself a pat on the back – that's quite an achievement!

If you didn't quite manage it or if you're not really sure how it all came together, then delete both the form and the subform and start all over again. The best way to learn is by practicing. (You can also view the result of these exercises in the database **Beads step4**.)

Reports

Reports

Reports and forms are very similar, so if you've mastered forms then you should find this section pretty straight forward!

Before starting your report design, it is important to identify the data that you want to display. If the fields that you need are not already contained in existing tables or queries, then you may need to first create a Select query to extract or calculate the fields that you want.

Note that when you create a report using the Report Wizard, the "Summary Options" button on the Sort Order dialogue box allows you to include totals and subtotals for the numeric fields in your report.

The Reports window in Design view can be used to edit the content and layout of an automatically generated report. It contains several regions including:

Page header. This appears on every physical page, and usually displays a descriptive title. Since it detracts from the space available for the data of the report, keep it brief.

Detail region. Holds the data controls that underlie the report.

Page footer. Appears at the bottom of each page, usually holding a page number and date.

The formatting of the final report appearance can be adjusted further in Design view by selecting and aligning controls, by changing the font attributes, and by adjusting Properties such as numeric formats.

Exercise

For this exercise, you'll continue to work with the **Video step2** database.

Requirements

You have been asked to create a report that will list the contact details of each customer on your books.

1.	Use the Report Wizard to select the all the fields from the Customer table (see example on the next page). Do not add any grouping levels. Sort the report lines in ascending order of surname. Use tabular layout in portrait orientation, and choose any style that you like. Leave the name of the report as Customer .
2.	Close print Preview, change to Design view, and notice <ul style="list-style-type: none"> • The Report Header section, which will be printed at the top of the first page of the report. It contains a label with static text content. • The Page Header section which will be printed at the top of each page. This also contains labels with text content. • The Detail section which will be repeated for each line of data. This contains "text boxes", which display values (either text or numbers) obtained from database fields. • The Page Footer region which will be repeated at the bottom of each page. This contains text boxes created by the Report Wizard which display Access functions and not database fields.
3.	Still in Design view, click on the label in the report header, drag it wider, and change the text to read "List of customers". (Also be aware that one Detail line in the report design may be converted to multiple lines in the actual report shown with Print Preview.)
4.	Save your report.



List of customers

Surname	Initials	CustNumbe	Address1	Address2	Postcode	Phone	Units
Adams	A	ADA001	21 Main Road	Rondebosch	7700	0216892435	17
Bester	B	BES001	3 New Haven	Rosebank	7700	0216981234	2
Center	C	CAN001	102 Forest Hills	Mowbray	7700	0216689436	25
Davids	D	DAV001	29 Westerford Road	Newlands	7700	0216984343	51
Edwards	E	EDW001	18 Smuts Avenue	Claremont	7700	0216984321	0
Fester	F	FES001	18 Hart Street	Claremont	7700	0216983412	0

09 April 2009 Page 1 of 1

Now you are going to create a report called **Rental History** that gives the details of each video and lists all the customers that have rented it together with dates. This report will be based on several tables.

1. Using the wizard, select the data fields shown in the example below and view them by Video with no grouping. Sort by HireDate and use stepped layout.
2. In Design view, change the report header to match the example.
3. Left align the content of all the labels and data fields.
4. Save your report.

Rental history

2009/04/12 10:20:34 AM

Rental history

VidNumber	VidTitle	VidType	VidAgeLimit	Surname	Initials	CustNumber	HireDate
7003	The Vampire	Horror	18	Bester	B	BES001	2009/03/01
7005	Titanic	Drama	13	Bester	B	BES001	2009/03/14
				Fester	F	FES001	2009/02/26
7006	American Pie 2	Comedy	16	Adams	A	ADA001	2009/03/07
8001	Panic Room	Drama	16	Adams	A	ADA001	2009/03/13
				Bester	B	BES001	2009/02/16

Once again, if you're not really confident that you could do this on your own, then delete both the reports you have just created and repeat the exercise.



The Access Switchboard

Why use a switchboard?

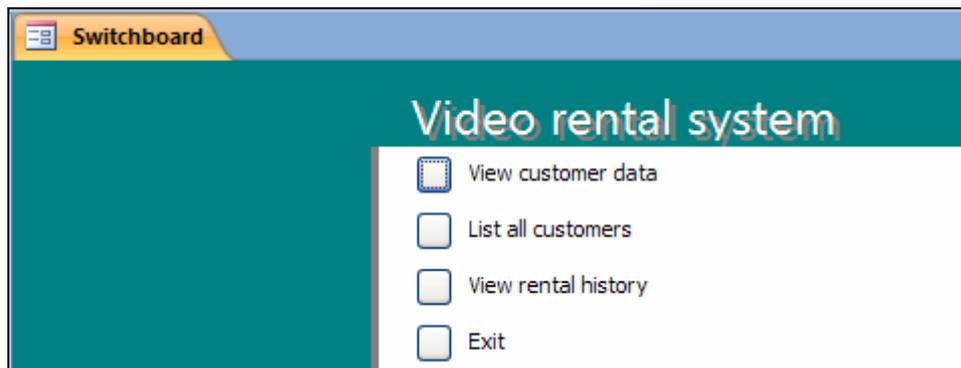
The *Switchboard* in Access provides a professional looking interface for viewing the forms and reports that you have created in your database. It can also help to prevent inexperienced Access users from accidentally editing or deleting important data.

A switchboard is basically a form linked to a database table. The form displays a set of labeled buttons, and the associated table stores instructions about what should happen when each of the buttons is clicked. For example, you may have one button that loads a specified form, another that displays a particular report, and another that closes Access.

Before creating your switchboard, you need to have developed all the forms and reports that it will activate - you can't link a switchboard button to a non-existent object. If the switchboard is going to display a form, then you also need to decide whether the form will be used to display existing records stored in the database, or only to add new records. Bear in mind that if a form is going to be used for adding new records, then it should include ALL the fields in the underlying table. Why do you think this is necessary?

A switchboard can have several levels. For a business accounting system, you might perhaps want a top-level page that has buttons labeled Debtors, Creditors, General Ledger, Cashbook and Stock. Each of these buttons would in turn link to another set of buttons, e.g. under Debtors you might have Create Invoice, Issue Credit Note, Print Statements, etc.

For now, we are going to create a single switchboard page (the Main Switchboard).

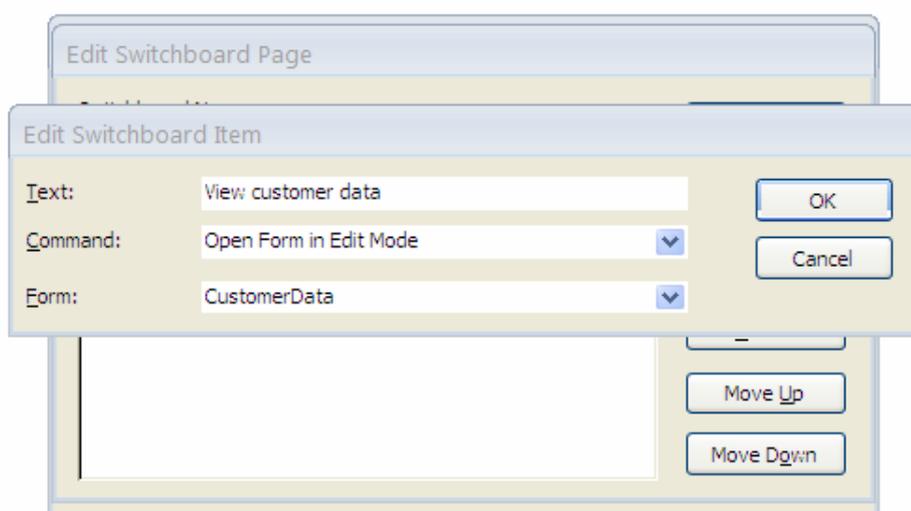


Exercise

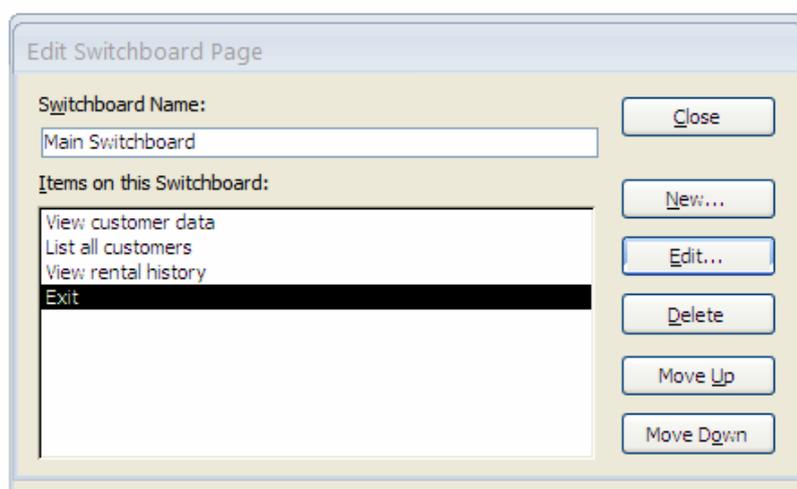
For this exercise, you'll continue to use your **Video step2** database. The switchboard we create will have links to the form called *CustomerData*, and the reports called *Customer* and *RentalHistory*.

Initially the main switchboard contains no items at all. You are going to add four new buttons, which will each result in a different action when clicked:

1.	To create your switchboard, click <i>Database Tools</i> and then <i>Switchboard Manager</i> . Since no switchboard exists yet for this database, you will be asked whether you want to create one – click Yes . An empty switchboard will be displayed, containing only a default Main Switchboard page. Click the Edit button to update the layout of this page.
2.	Click New to create your first switchboard item. In the <u>text</u> field type "View customer data" - this will become the label for item one. In the <u>command</u> field, select "Open form in edit mode" - this will allow you to view existing data. In the <u>form</u> field, select "CustomerData" – this is the form that you want displayed when the button is clicked. Now click OK to save your setup for item one.



3.	You should now be back at the main switchboard page, with one item completed. Click New to create your second switchboard item. In the text field type "List all customers". In the command field, select <i>Open report</i> . In the form field, select <i>Customer</i> . Click OK to save your setup for item two.
4.	Click New again to create your third switchboard item. In the text field type "View rental history". In the command field, select <i>Open report</i> . In the form field, select <i>Video</i> . Click OK to save your setup for item three.
5.	Click New one more time to create your final switchboard item. In the text field type "Exit". In the command field, select <i>Exit application</i> . Since you are not going to open a form or report when this button is clicked, you will not be presented with a form field as you were in the previous steps.



6.	Close the switchboard manager.
----	---------------------------------------

When a switchboard is created, Access automatically generates a **Switchboard table**, which records the actions associated with each button, and a **Switchboard form** to display the buttons and their text. If you delete either of these, then your switchboard will not function!

Check the left pane to see that you have a Switchboard form and a Switchboard items table in your database.

To edit the switchboard, you need to go back to *Database Tools* and the *Switchboard Manager*. We'll do that now, to change the title of the switchboard to something a bit more meaningful.

1.	To re-open your switchboard, click <i>Database Tools</i> and then <i>Switchboard Manager</i> . Since your database has an existing switchboard, you'll be taken straight to the edit window.
2.	Change the current switchboard name to be "Video rental system", and then close the switchboard manager.

The **Video step3 database** includes all the forms and reports that have been created so far, as well as the switchboard. If you want to re-do the entire switchboard from the beginning, then you'll first need to delete both the switchboard form and the switchboard items table.

Note that if a switchboard button has been created that opens a form in Add mode, then existing data will not be displayed when the button is clicked - the form can only be used for the entry of new data. Opening a form in Edit mode will allow existing data to be viewed and new records to be added. If you need to open a form in which existing data can be viewed but not changed, then you must create a switchboard button which open the form in Edit mode, and in the properties of the form itself you need to set Allow Edits to No.

You can also specify that when your Access database is loaded by a user, then the switchboard should be displayed instead of the usual database window. To do this:

Click *Access Options* under the *Office Button*, and select *Current Database*. From the *Display Form* list, select the Switchboard form. The next time that you load the database, your switchboard will automatically be displayed.

This introductory course has covered the following topics in Access:

- **Fields (data type, length, input mask, validation rule)**
- **Relations between tables (primary key, foreign key, referential integrity)**
- **Select queries (calculated fields, aggregate queries, parameter queries)**
- **Action queries (make table, update, delete)**
- **Forms**
- **Reports**
- **Switchboards**

Have fun using and growing your new skills!

