3. Compare the HCI of your improved form with the customer form as it was before you made any changes to it (see page 100).
   a In what ways is this form easier and more pleasant to use?
   b Has the re-arrangement of the fields made the connections between the data simpler to comprehend?
   c What further improvements could be made to the form to improve the HCI of this form?
4. Before leaving this activity it is important to realise that despite the changes we have made to this form, the basic data has not altered.

   Click on the datasheet view button and you will see the data is still as it was, and the order of the fields is still the same. The only changes we have made have been to the appearance of the form.

## Activity 6.3 – Inserting, deleting, updating and sorting

In this activity we will look at maintaining and sorting the data in an existing database.

1. Start *Access* and open your copy of the *Vet* database.

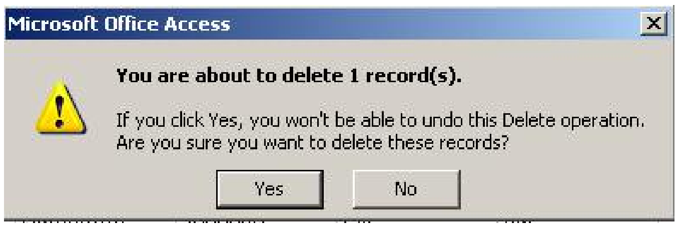
   Switch to either form view or datasheet view.

   Note: do *not* make the following changes in design view.
2. There are two new customers who visited Dr Harry today.

   Click on the new record button ▶✱ (one of the navigation buttons at the bottom of the form) and add the following data:
   - *Allen Collins* of *13 Hardy St, Newtown* (ph *4354678*) has a *3* year old pet *galah* called *Golly*
   - *Leanne Bradford* of *34 Ford Av, Harlaxton* (ph *4394567*) has a *12* year old collie called *Lassie*

   Both owe *$35.50* for today's visit.
3. *Peter Gillam* of Drayton has paid the money he owes and has moved away. His data can now be removed from the database.

   To do this click on his record and from the top menu choose *Edit > Delete Record*.



Microsoft Office Access

**You are about to delete 1 record(s).**

If you click Yes, you won't be able to undo this Delete operation.
Are you sure you want to delete these records?

   Click *Yes* to delete the record.

   Note: this delete action *cannot* be undone. Once the record is deleted it cannot be brought back.
4. Two customers have notified us that their details have altered. Make the following changes:
   - Nick Scott has moved around the corner to *4/15 Jenkins St*; he has kept the same phone number
   - Janet Quinn's old cockie has died; she has bought another which she is calling *George II*; it is *one* year old.

5. The data in the database is not very well organised at the moment. If we wish we can re-sort it into a different order.

   Sorting can be done in either form or datasheet view. In either, simply click on the field you want in order and then click on one of the two sort buttons (at right).

   a Click on the *surname* field. Sort the records into alphabetical order of *surnames*. Who is the last person alphabetically?

   b Sort the records into alphabetical order of *pet name*. Which is the first pet?

   c Sort the records in order of *amount owing*. Who owes the least money?

   d Sort the records of *amount owing* in descending order. Who owes the most money?

   e Sort the data into *suburbs*. How many records are there for the Harlaxton area?

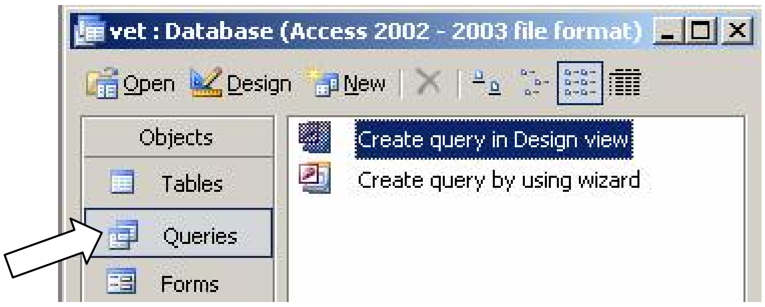
   f Sort the data by *pet type*. How many guinea pigs are there?

Note: It is not necessary to save an Access database before closing. Any changes you make are automatically recorded as you go. Data will however be saved in the order in which it was last sorted.

## Activity 6.4 – Simple queries

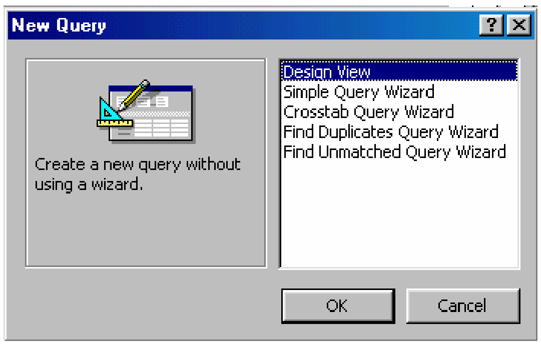
In the last activity we used the sort function to find information. In this activity we will use a more powerful data collection method called a *query* or *filter*.

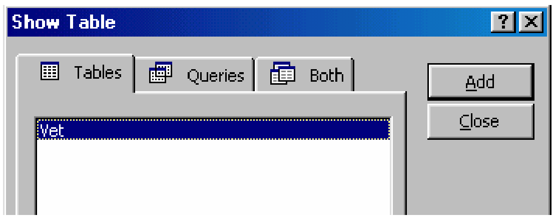
1. In the *Vet* database close any open form and switch to the *Queries* object:


vet : Database (Access 2002 - 2003 file format)
Open
Design
New
Objects
Tables
Queries
Forms
Create query in Design view
Create query by using wizard

2. Say we wish to just view the records of people who live in Newtown. To do this we will need to create a filter to block out all other records.

   a Click on the *New* button and choose *Design View*. (Or click *Create query in Design view*.)


New Query
Create a new query without using a wizard.
Design View
Simple Query Wizard
Crosstab Query Wizard
Find Duplicates Query Wizard
Find Unmatched Query Wizard
OK
Cancel

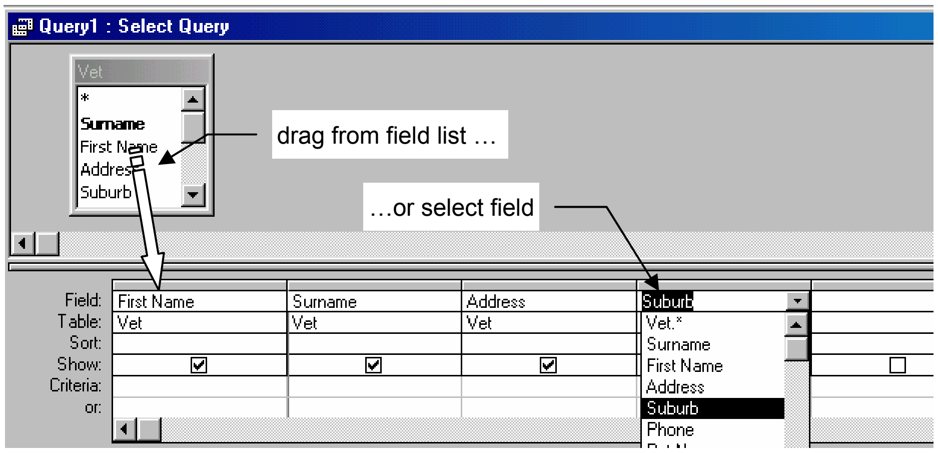


   In the dialogue that appears click on *Add* and then *Close*. This will give us the Vet table to work on.

In more complex databases we can build queries based on several different tables of information, each of which would have to be added to the query grid.

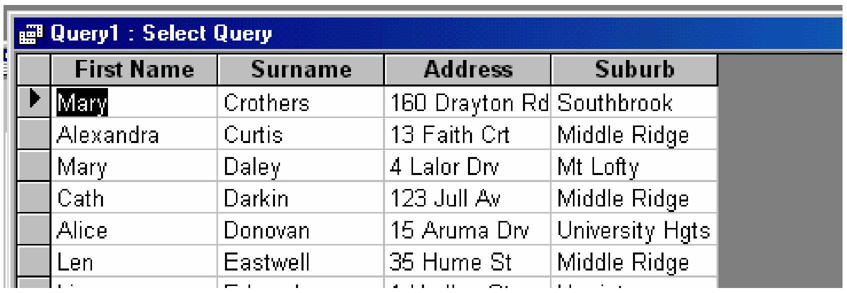
b We now need to decide which fields of information we want to display. Say we just want to show *first name, surname, address* and *suburb* only.

To get this put these four fields in the first four columns. This can be done either by clicking and dragging from the field list, or by using the drop down arrows in each column:



| | | | | | |
|---|---|---|---|---|---|
| Field: | First Name | Surname | Address | Suburb | |
| Table: | Vet | Vet | Vet | Vet.* | |
| Sort: | | | | | |
| Show: | ✓ | ✓ | ✓ | | ☐ |
| Criteria: | | | | | |
| or: | | | | | |

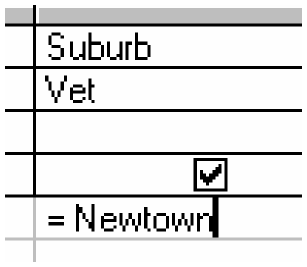
c To see what we have achieved so far click on the *Run* button on the toolbar at top.



| First Name | Surname | Address | Suburb |
|---|---|---|---|
| Mary | Crothers | 160 Drayton Rd | Southbrook |
| Alexandra | Curtis | 13 Faith Crt | Middle Ridge |
| Mary | Daley | 4 Lalor Drv | Mt Lofty |
| Cath | Darkin | 123 Jull Av | Middle Ridge |
| Alice | Donovan | 15 Aruma Drv | University Hgts |
| Len | Eastwell | 35 Hume St | Middle Ridge |

Just the four selected columns are displayed. (While these are displayed they can also be sorted if desired.)

d Now we need to filter out the suburbs that are not Newtown.



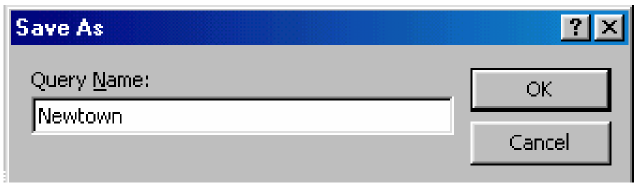
| |
|---|
| Suburb |
| Vet |
| |
| ✓ |
| = Newtown |

Return to design view and under *Suburb* in the *Criteria:* row write = Newtown:

When you run the query only data for Suburb = Newtown is displayed.

(Note that when you return to design view of this query Newtown now has " " quotes around it. *Access* adds these because *suburb* is a text field.)

e Close this query and when prompted save it as *Newtown*:

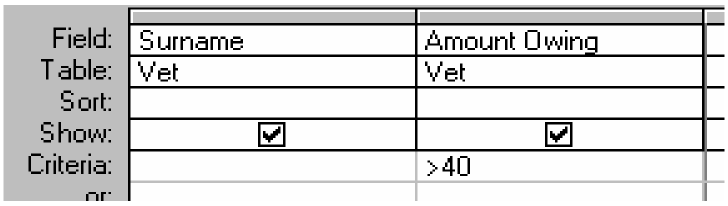


Query Name: Newtown

3. Our next query will find people who owe more than $40.00 .

   a In the *Queries* tab again click on *New,* then *Design View* and *Add* the *Vet* table.

   b This time we will only work on surname and amount owing:

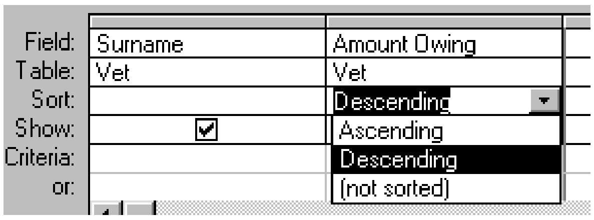


|  |  |  |
|---|---|---|
| Field: | Surname | Amount Owing |
| Table: | Vet | Vet |
| Sort: |  |  |
| Show: | ✓ | ✓ |
| Criteria: |  | >40 |
| or: |  |  |

   In the criteria row put >40 (i.e. show people who owe more than $40.00) and run the query.

   c Once the query has found these people we might want to find who owes the most. We can either do a sort in the datasheet view or we can make the sorting part of the query.

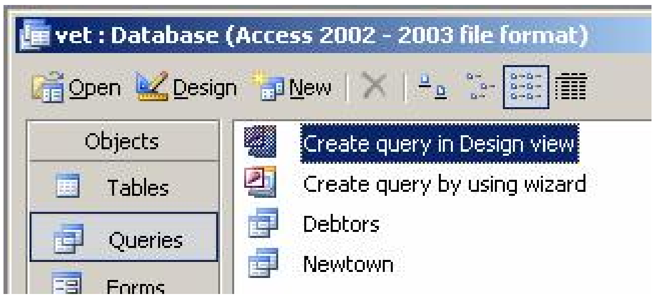
   To do this simply select *Descending* from the drop down list in the sort row under *Amount Owing*:



|  |  |  |
|---|---|---|
| Field: | Surname | Amount Owing |
| Table: | Vet | Vet |
| Sort: |  | Descending |
| Show: | ✓ | Ascending |
| Criteria: |  | Descending |
| or: |  | (not sorted) |

   We now have the people who owe more than $40, with those owing the most first.

   d Save this query as *Debtors*.

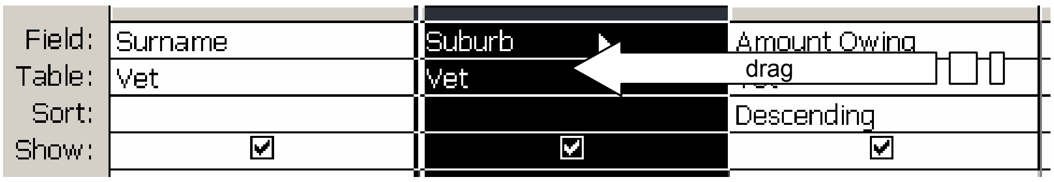

vet : Database (Access 2002 - 2003 file format)
Open
Design
New
Objects
Tables
Queries
Forms
Create query in Design view
Create query by using wizard
Debtors
Newtown

4. The advantage of making the sort part of a query is that we can sort on more than one field.

   Say Dr Harry wants to send a debt collector around to collect from those who owe him more than $40. He needs a list of customers by suburb, with the customer owing the most in each suburb listed first.

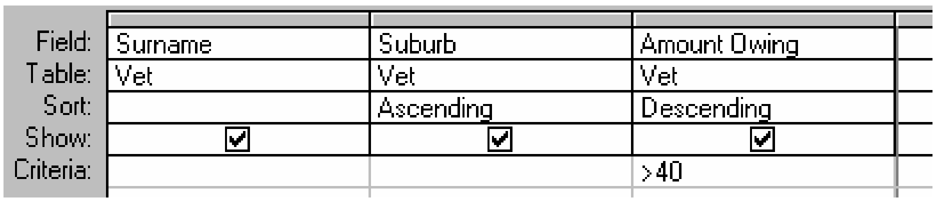
   a Open the *Debtors* query in design view. (In the *Query* object click once on *Debtors* then on *Design*, or double click on *Debtors* and then switch to design view.)

   b Add the *Suburb* field and click and drag it so that it is ahead of the *Amount Owing* field.



|  |  |  |  |
|---|---|---|---|
| Field: | Surname | Suburb | Amount Owing |
| Table: | Vet | Vet | Vet |
| Sort: |  |  | Descending |
| Show: | ✓ | ✓ | ✓ |

c Sort the suburbs into alphabetical order:



| | | | |
|---|---|---|---|
| Field: | Surname | Suburb | Amount Owing |
| Table: | Vet | Vet | Vet |
| Sort: | | Ascending | Descending |
| Show: | ✓ | ✓ | ✓ |
| Criteria: | | | >40 |

d Run the query.

Why was it necessary to put suburb before amount owing? (Hint: *Access* works out queries from left to right.)

5 Make queries to determine the following:

a What is the maximum amount owing in Harristown?

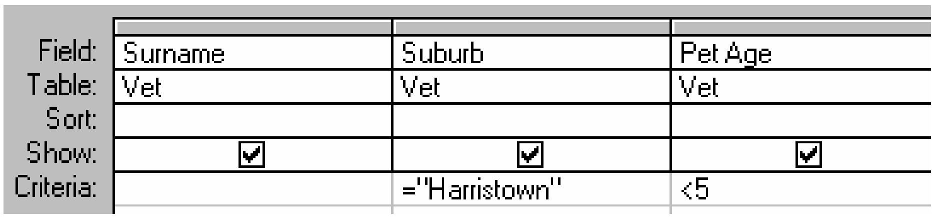
b Find the name of the cat Dr Harry has treated most recently.

c What is the name of the six year old dog that has the most amount owing on it?

6. Queries can work on more than one condition.

a Say we wish to find out how many owners live in the Harristown area *AND* have pets under 5 years old.

In this case *both* Suburb = Harristown *and* Pet Age < 5 must be true for the records displayed.

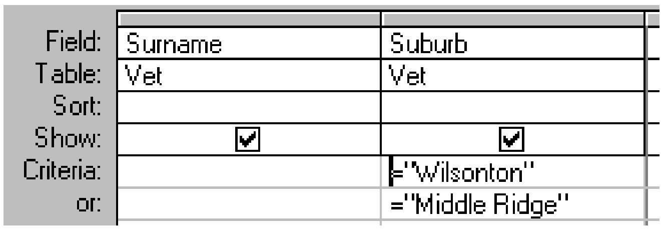


| | | | |
|---|---|---|---|
| Field: | Surname | Suburb | Pet Age |
| Table: | Vet | Vet | Vet |
| Sort: | | | |
| Show: | ✓ | ✓ | ✓ |
| Criteria: | | ="Harristown" | <5 |

Develop and test the above query.

b On the other hand we might want to find which owners live in the Wilsonton *OR* the Middle Ridge area.

In this case *either* Suburb = Wilsonton *or* Suburb = Middle Ridge. To show this we place both conditions in the *Suburb* column:



| | | |
|---|---|---|
| Field: | Surname | Suburb |
| Table: | Vet | Vet |
| Sort: | | |
| Show: | ✓ | ✓ |
| Criteria: | | ="Wilsonton" |
| or: | | ="Middle Ridge" |

Develop and test this query.

Remember:

*AND* means *both* conditions have to be true. To *AND* conditions place them in *separate* columns.

*OR* means *either* condition can be true. To *OR* conditions place them in the *same* column.

7. Create queries to find out answers to the following:
   a How many owners have dogs?
   b What is the number of owners who live in the Mt Lofty area *and* who owe more than $40?
   c How many owners have pets aged between 3 *and* 10?
   d How many owners made their last visit between 1st April *and* 31st Oct last year?
   e How many pets are rodents (rat, mouse, hamster *or* guinea pig)?
   f How many owners live in the Middle Ridge area *and* have a dog *or* a cat?
   g How many cats has Dr Harry seen this year?

8. This form of developing queries where fields are dragged onto a grid and sample data entered into criteria rows is called query by example (QBE).
   a Explain why you think this form might be called query *by example*.
   b QBE is used to set up and use a query. How difficult is QBE to understand, simple, moderate, difficult or complex?
   c What features of QBE do you think support this assessment.
   d Identify any metaphors or affordances used in the QBE system.
   e Suggest another possible way of querying a database other than QBE.

## Activity 6.5 – Video database

Having seen some of the different things we can do with a database it is now time to create one of our own. The database we will develop will be for a video store and will have tables for videos to hire, members who borrow them, and a list of who has which video out on hire.

1. Click on the *New* button to create a new blank database called *Video Hire*. Save it to a suitable folder.

2. a Choose the *Tables* object and double click on *Create table in Design view*:
   b Enter the field names at right.

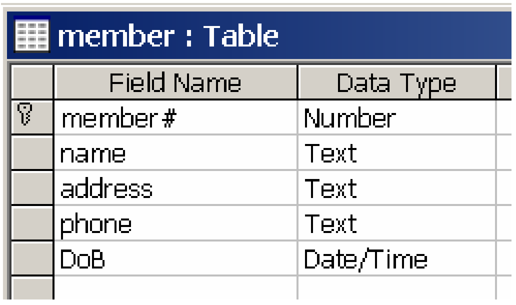
   *Access* will automatically select *Text* as the data type for each field but this is not always suitable.

   Click on the data type for member#, and then on the drop down arrow that appears.

   Choose *Number* from the list offered.

   c Set the indicated data type for the other fields.

   (Why is *phone* set as text?)


member : Table

| Field Name | Data Type |
|---|---|
| member# | Number |
| name | Text |
| address | Text |
| phone | Text |
| DoB | Date/Time |
| | |

   d In turn click on the data type for each field and set suitable properties at the bottom.

   e.g. for the text data types set an appropriate field length i.e. how many letters long *name* or *address* might be; make the number property a fixed integer with no decimal places; choose your preferred date format.

3. In organising and querying data it is necessary to have a way of identifying each record. In this database the best way of telling records apart will be the member number.