

Introduction to LibreOffice Base (LOB)-2

Relationships

Robin Beaumont

Date: Saturday, 07 April 2012 e-mail: robin@organplayers.co.uk

Contents

2.	LEARNING OUTCOMES CHECK LIST	2
3.	INTRODUCTION.....	2
4.	RELATIONSHIPS IN LIBREOFFICE BASE.....	2
5.	CREATING RELATIONSHIPS.....	3
6.	WHAT ADDING REFERENTIAL INTEGRITY DOES WHEN ENTERING DATA....	6
7.	AN INTRODUCTION TO DEPENDENCY	8
8.	DELETING A TABLE WITH RELATIONSHIPS.....	8
9.	SUMMARY	8
10.	CHECK WHAT YOU HAVE LEARNT.....	8

This handout is part of a series.

Please see section 7.1 at:

www.robin-beaumont.co.uk/virtualclassroom/contents.html

Please note that before you start this practical chapter you should have completed database concepts (1) and (2) along with practical chapter 1 available at the above link

2. Learning outcomes check list

This practical chapter aims to provide you with the necessary skills and knowledge to achieve the learning outcomes listed below. After you have completed this practical chapter you should come back to these points ticking off those with which you feel happy.

Learning outcome	Tick box
Be able to create foreign keys in LibreOffice Base	<input type="checkbox"/>
Be able to use the Relationships window	<input type="checkbox"/>
Be able to set the appropriate options in the relationships dialogue box to enforce 'referential integrity'	<input type="checkbox"/>
Be able to create, edit and save relationships in LibreOffice Base	<input type="checkbox"/>
Understand the concept of dependency (referential integrity) and the effects this has when adding, editing or deleting records	<input type="checkbox"/>

3. Introduction

In the last LibreOffice Base practical chapter we created the tables for the consultations database giving the database the name Cons1. During this chapter we will carry on developing it further by adding relationships between the tables. If you have lost your copy of cons1 you can download a copy from: http://www.robin-beaumont.co.uk/virtualclassroom/chap8/libreoffice/libreoffice_base_files.zip

You should have also worked through the 'database concepts 2 Relationships' chapter, first to ensure you have gained an understanding of what relationships are, before completing this practical chapter.

4. Relationships in LibreOffice Base

When we created the six tables that make up the consultations database in the last session we added the necessary foreign keys to the tables. What has this done exactly?

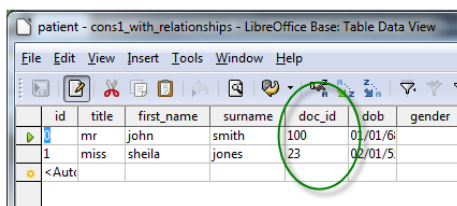
Exercise 1. investigating foreign keys

Open up the cons1 database you created.

Open up the patient table

You should now be in datasheet view (the view that shows the contents of each record) rather than design view.

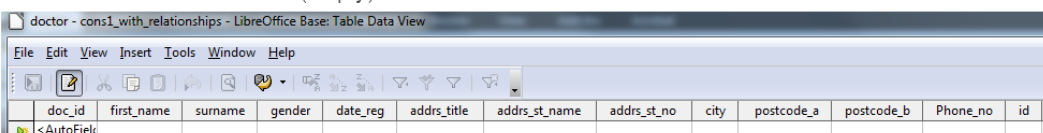
patient table



id	title	first_name	surname	doc_id	dob	gender
1	mr	john	smith	100	01/01/6	
2	miss	sheila	jones	23	02/01/5	

You should have two records in the patient table and the doctors table should be empty:

Doctor table (empty)



doc_id	first_name	surname	gender	date_reg	addr_s_title	addr_s_name	addr_s_no	city	postcode_a	postcode_b	Phone_no	id
--------	------------	---------	--------	----------	--------------	-------------	-----------	------	------------	------------	----------	----

But the doc_ids do not refer to any records in the doctors table!!

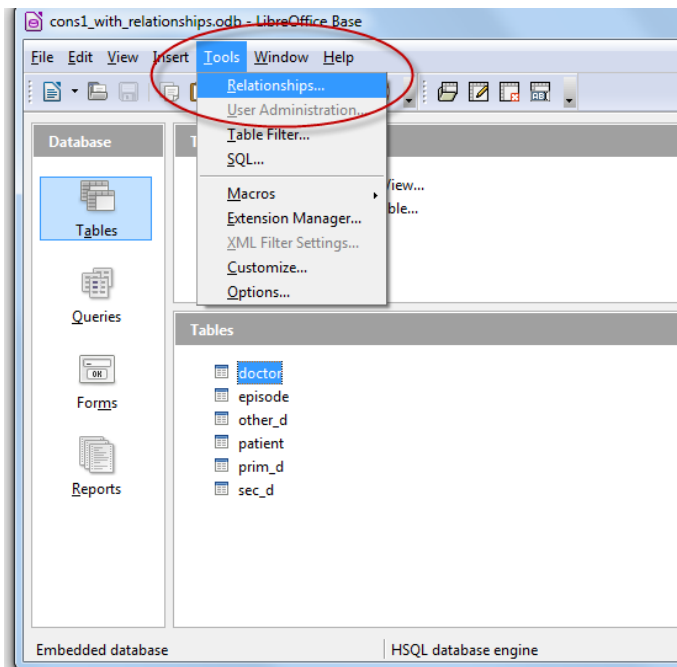
The above exercise demonstrates clearly that just adding foreign keys to a table does not ensure that the computer adheres to the foreign key principle (i.e. foreign key values can only be those that already exist as a primary key in the referencing field). Luckily the DBMS (Database Management System), Base, provides a 'relationships' facility to ensure we do not end up in the situation demonstrated above.

Key point:

Adding foreign key fields to tables does nothing to ensure that the values in them actually exist in the master table where they are the key. To do this you need to set up 'relationships' in the DBMS.

Exercise 2. investigating deleting records

Delete the records in the patient table, for details refer back to the last practical chapters material). This is essential for you to be able to carry out the next exercise successfully.



5. Creating relationships

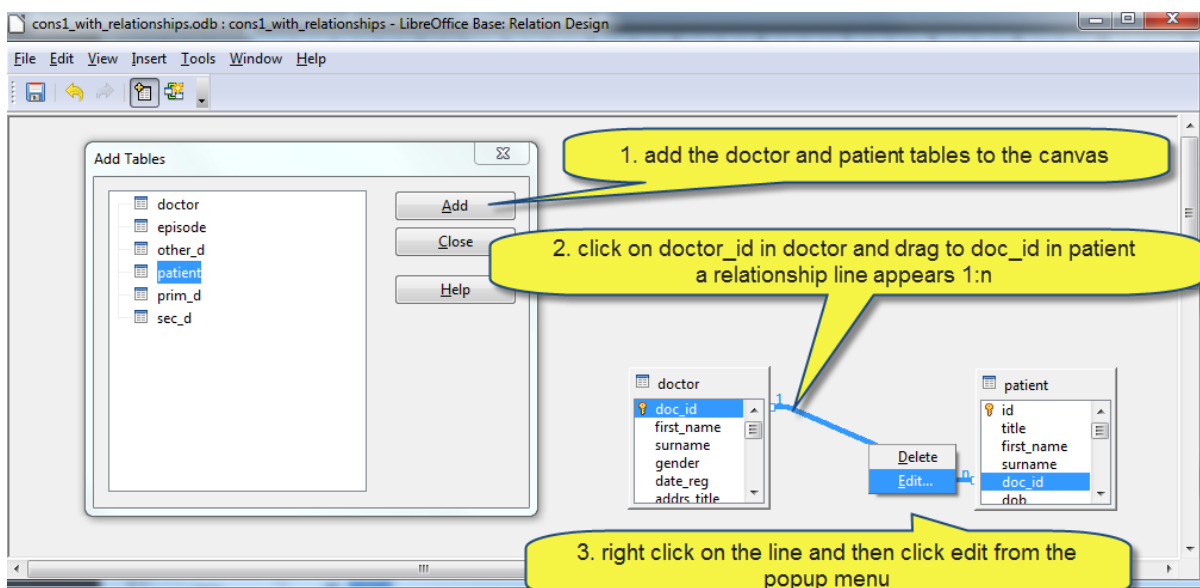
This is achieved in Base by using the **Relationships** window which you call up by clicking on the tools option of the main menu. In the relationships window you should now have a 'add tables' dialogue box in front of you requesting which table(s) you would like to add.

From within the Show table dialogue box:

Choose doctor by highlighting it then click the add button.

Choose patient by highlighting it then click the add button.

Click the close button.



You should now have the two tables in the relationship window. You can resize the various tables to see all the fields by dragging the sides of the tables and also move the tables in the window until they're arranged in the layout you want. The next stage is to create the relationship between the two tables:

To create a relationship, you use the mouse to drag one or more fields **from the master table to the child table**. When creating a relationship, you drag from the primary key of one table (shown in bold in the field list) to a matching field (the foreign key) in the related table.

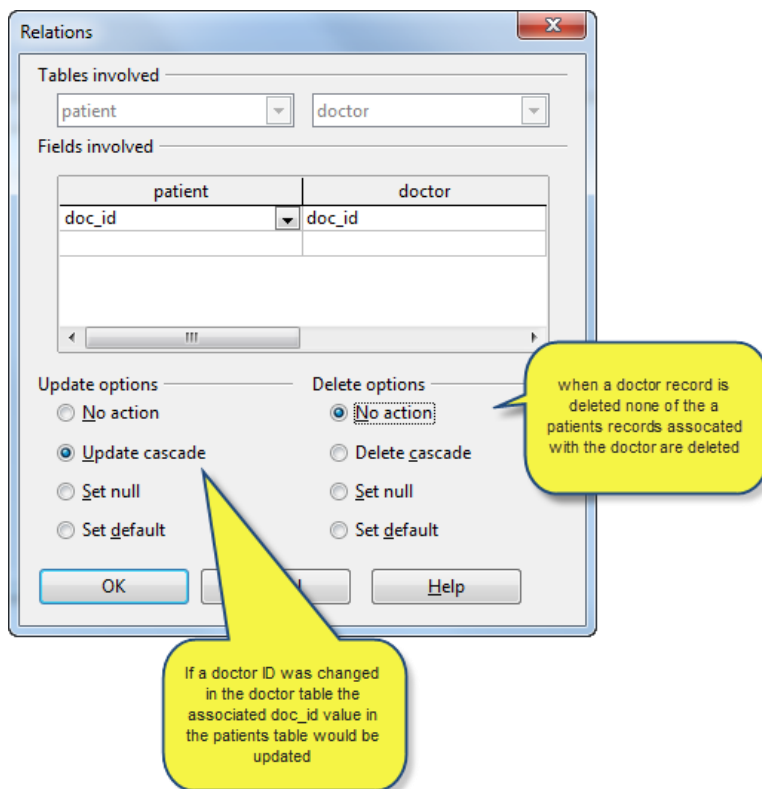
Exercise 3. Creating a link between the doctor and patients tables

Carry out the steps described above.

The relationships dialogue box now pops up because while we have created the link between the two tables we now need to specify certain referential integrity rules. If the **Relations** dialog box has not appeared you can call it up by clicking on the relationship line and selecting **Edit**. What we want is the make sure that:

- If a doctors record ID is changed the associated patient records stay with the same doctor (technically called enforcing cascade updating)
- When a doctors record is deleted than all the associated patients records are not also deleted (technically called preventing cascade deleting).

This is achieved by setting the following options:



Exercise 4. Enforcing appropriate referential integrity between the doctor and patients tables

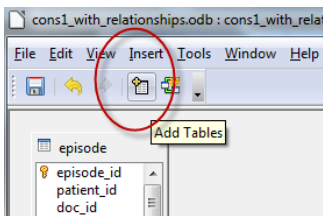
Carry out the steps described above.

If you are confronted with an error message such as "unable to create relationship as records unsuitable ..." You must have forgotten to delete the records in the tables as previously requested to do so at the beginning of this session. If this is the case please Delete them now and try again!

Key point:

Always drag from the primary key ('1' side of the relation) to the foreign key (the n=many side). If you don't you'll get the relationship the wrong way round!

We will now create the other relationships needed to make the database function, considering first the episode table.



To add more tables to the relationships canvas you may need to bring up the add table dialog box if you had closed it. You can achieve this by clicking on the **add table icon** shown opposite.

We now need to add the **episode** table to the canvas, do this by selecting the episode table and click **Add**. You can then close the Add tables dialog box.

We now need to add two more relationships and I can express specifically which fields we wish to link on by using the table_name dot field_name format:

Drag from Doctor.doc_id to episode.doc_id

Drag from patient.id to episode.patient.id

This indicates that a doctor has zero or more episodes and a patient also has zero or more episodes. The result is shown opposite.

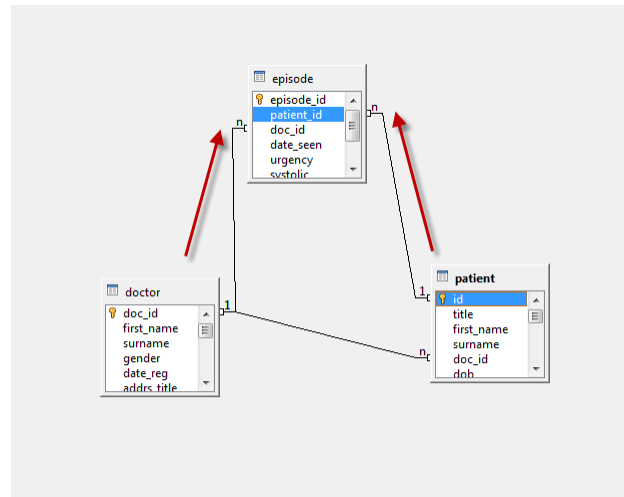
Now we need to think about the referential integrity constraints for each of these:

Do we want an episode to stick with a particular doctor if the doctor id is updated or deleted?

Do we want an episode to stick with a particular patient if the patient id is updated or deleted?

I think the answer to both of these is yes for updating but no for deleting. So we set the relations options dialog box the same as we did for the doctor patient relationship.

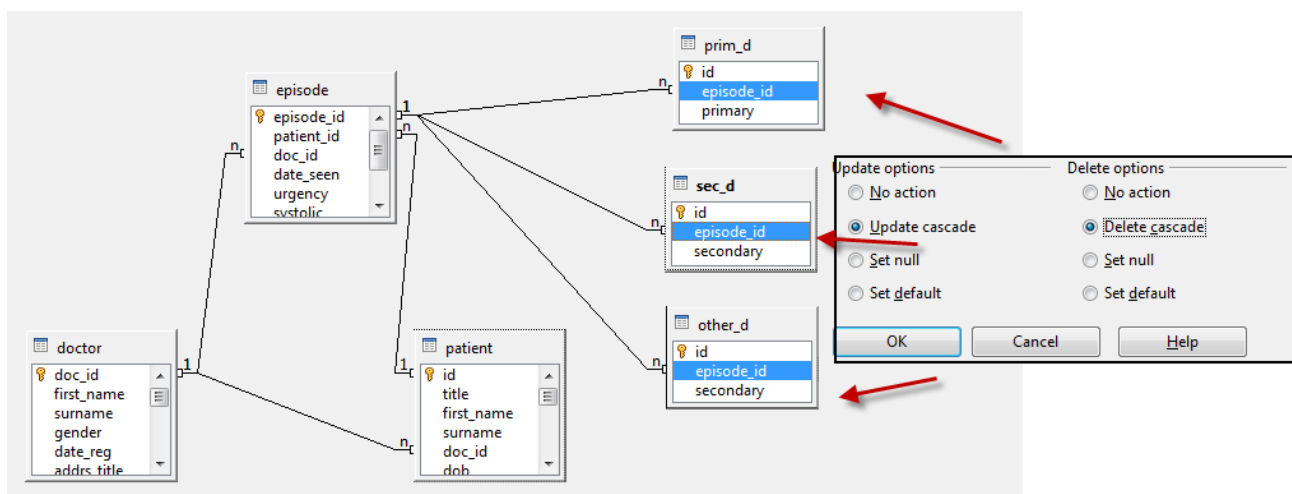
As an aside I would also say that patients and doctors records would never be deleted just archived (by setting a particular field value to dead/retired/left or archived etc.)



Exercise 5. adding the episode table and relationship to the window

Please follow the steps above to obtain the relationship window looking like the above screenshot, also remember to set the appropriate referential integrity rules.

We now need to setup the last set of relationships between the various diagnoses and episode. The final result is shown below.

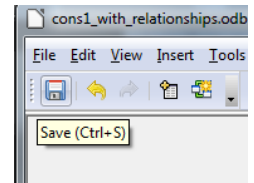


I feel that it is sensible to use the delete cascade option here because if we delete a episode it is irrelevant to have a diagnosis record hanging around without any knowledge of who it belongs to.

Exercise 6. Completing the relationships

Now add the last three, diagnosis, tables, to the relationships window and also add the necessary relationships to obtain the diagram on the previous page.

Make sure you save your work by either pressing the short cut key combination Ctrl+s or clicking on the save icon in the window. It should be noted that saving while in the relationships window also saves the actual referential integrity rules as well as the layout picture in the relationships window.



6. What adding Referential Integrity does when entering data

The following exercise demonstrates some of the advantages (and disadvantages) of creating relationships and referential integrity constraints.

Exercise 7. Trying to add an invalid child record

Add the following record to the patient table.

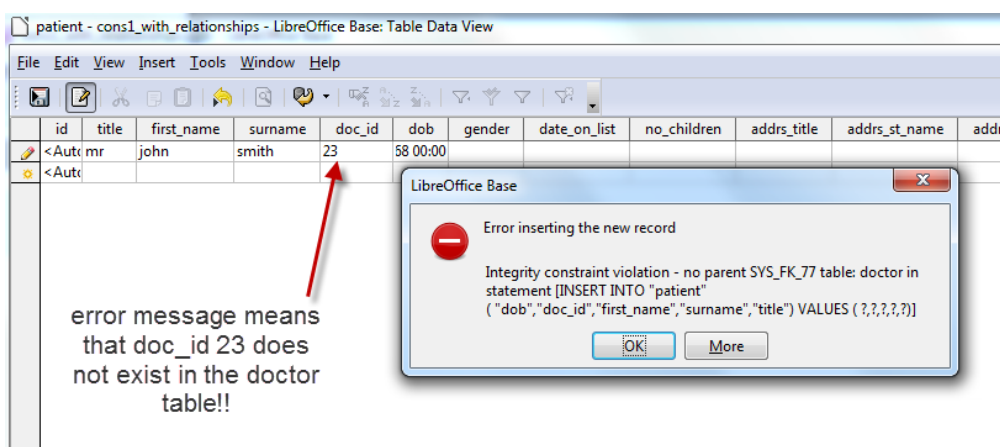
Remember to add the record to the database and not just the screen you need to move to another record (row) in the table or press Ctrl+s to give the computer the necessary kick.

title	First name	surname	Doc_id	dob
mr	john	smith	23	01/01/1968

What happens. Do you consider this useful?

Clicking OK on the error message box does not make the record disappear from the screen to do that you need to press the **Esc** key.

The error is basically saying hold on you don't have a doctor with an id value of 23 to link to! You can't link a child (i.e. patient record) to a non-existent parent (doctor). To solve the problem we either need to select an appropriate doctor id or create a doctor record with the appropriate id.

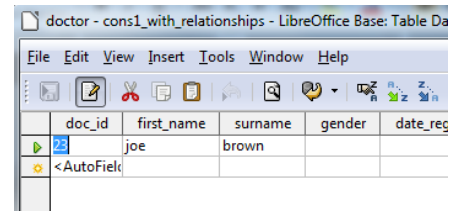


Exercise 8. adding a parent (doctor) record

Add a record to the doctor table with an ID of 23. (although this is an autovalue field you can add the value for the first record - it is only subsequent ones you can't)

You can put any value you want into the other fields but remember to add the record to the database and not just the screen by either moving to another record (row) in the table or clicking the save icon.

Then go back to the patient table and try again putting the patient record in.



doc_id	first_name	surname	gender	date_reg
23	joe	brown		
<AutoField				

Exercise 9. Editing a primary key that is referenced as a foreign key somewhere else

You should now have a record in the doctor table with a ID of 23 and a record in the patients table with a Doctor ID value of 23. The value of the other fields do not matter for now.

Try changing the doctor ID value in the doctor table record to 100.

What happens to the patient record?

You will have noticed that the value in the child record in the Patient table is automatically updated.

Another type of integrity constraint was concerned with deletions, we will investigate now.

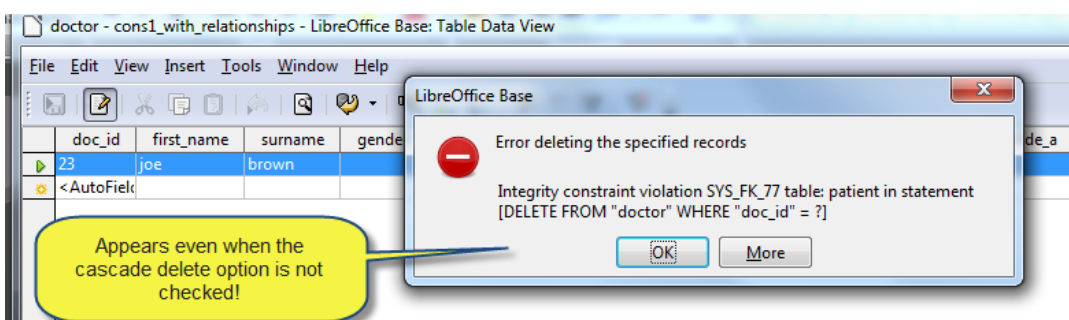
Exercise 10. Deleting a primary key that is referenced as a foreign key somewhere else

You should now have a record in the doctor table with a ID of 100 and a record in the patients table with a Doctor ID value of 100. The value of the other fields do not matter for now.

Try deleting the record in the doctor table. What happens? Describe below:

If necessary, press the Esc key to reset the record.

Some DBMS's always ensure that you can't delete the parent record if there are child records associated with it regardless of how you may have specified the integrity rules. For example in the last exercise above we set the integrity rule between Doctor and patient to only cascade updates not deletes so you would have thought it would be alright to delete a patient record that is associated with some patient records, but unfortunately the Base DBMS overrules the option and does not allow it here!



7. An introduction to dependency

What is happening in the above exercises is all to do with dependency. One table is said to be dependent upon another. The terms parent (i.e. master) and child are often used and I find the best way to think about dependency is to consider the various actions over time using the CRUD matrix discussed in Database Concepts (1).

Process:	Time -> -> ->	
	First action	Second action
Creation	Master	Child
Updating master:	Master	Child updated
Updating child:	Master	Child updated only if value is in master field
Deleting a master:	Master deleted only if no children exist	
Deleting a child:		No problem

The diagram is explained below:

Creation of a record: with a foreign key value requires that the value already exists in the table in which the foreign key is the primary key (doctor ID in doctor table must exist before creating a patient record with the doctor ID).

Updating of a master table key field: (the one which does not have the foreign key) This updated value is propagated down to the child foreign key values.

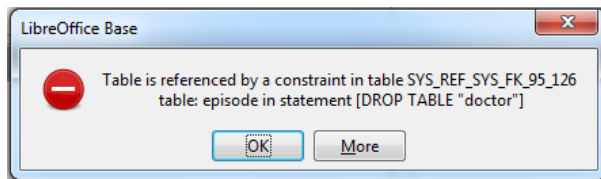
Updating a child table record: (i.e. a record in the patient table if it were not related to the episode table). This is not a

problem unless you try to change the foreign key value to one that does not exist in the master table.

Deleting of a master table key field: (i.e. where the Key is used as a foreign key somewhere else), as we discovered in the exercise above, this situation can create problems.

Deleting a child table record: (i.e. a record in the patient table if it were not related to the episode table). This is not a problem. Infanticide is a allowable activity for a database manager!

8. Deleting a Table with relationships



If you try to delete a table which forms part of a relationship you will find that an error dialog box appears. An example is given opposite of such a dialog box if I try to delete the doctor table. To be able to delete the table you first need to delete any relationships to/from it by going into the Relationships window and deleting them,

you do this by simply selecting each relationship and pressing the delete key in the window.

9. Summary

In this practical chapter we have seen how LibreOffice Base creates and maintains relationships using the Relationships window by creating several such relationships in the consultations database (cons1).

We have also seen the possible problems that can occur if the table structure (i.e. the model) is not thought through, particularly with regard to dependency and updating / deleting records. The added complexity of defining relationships in the model and specifying them in LibreOffice Base is unavoidable as maintenance of **referential integrity** (i.e. synchronisation of all the references = relations) is the core aim of a database. This was clearly demonstrated in the first exercise where anything could be entered into a foreign key field resulting in parent records being related to non-existent doctors (technically called 'dangling pointers')!

10. Check what you have learnt

Now go back to the beginning of the material for this practical chapter and read through the 'Learning outcomes check list'. How many can you tick? If you're not sure about any particular one go back through the relevant exercise, and then if you still have problems try discussing it with other students on the forum.