Introduction to LibreOffice Base (LOB) (5)

Selection Queries-2

Robin Beaumont

Date: Friday, 02 September 2011 e-mail: robin@organplayers.co.uk

Contents

This handout is part of a series. Please see section 7 at:
www.robin-beaumont.co.uk/virtualclassroom/contents.html

1. Learning outcomes check list for the chapter

This chapter aims to provide you with a number of skills along with the necessary knowledge for you to achieve the learning outcomes listed below. After you have completed the chapter you should come back to these points ticking off those with which you feel happy.

.

Learning outcome	Tick box
Know how to select individual fields into the QBE grid	❑
Know how to specify which fields in the grid will be displayed in the result	❑
Know how to specify which fields to sort the results on	❑
Know how to prevent the display of duplicate values in a result	❑
Know what summary functions are	❑
Know how to specify a query which will count the number of records	❑
Know what the grouping function achieves	❑
Be able to specify grouped results	❑
Know what a calculated field is	❑
Be able to specify a calculated field	❑
Be able to count the number of records which contain NULL values	❑

2. Introduction

This chapter will continue to look at **select queries** increasing our repertoire of the various functions offered by LibreOffice Base (LOB) . This will be achieved by looking specifically at **summary** functions**,** also known as statistical functions in business circles, along with finding ways of manipulating the results for maximum use.

The examples in this chapter use the cons2 database, **patient** table that has been printed out in the exercises part of the previous practical session.

The first few sections provide you with additional information about how you specify queries which is used in the subsequent exercises.

Exercise 1. Opening the Database

Open the cons2 database now.

3. Query Specification

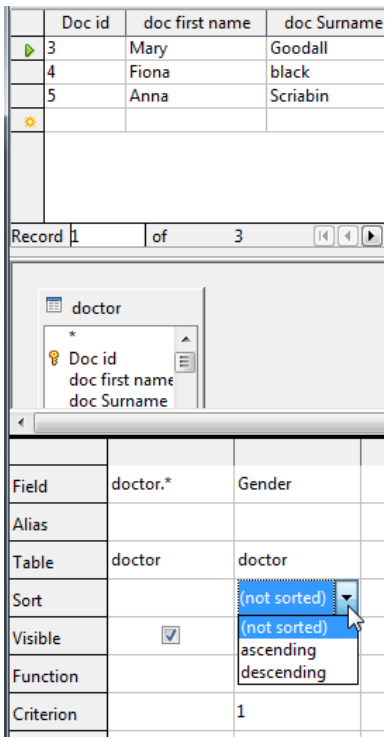
You can specify various field properties including position, sorting and removal of duplicate answers when you create a query. Details are provided below.

3.1 Field selection / position

During the last practical session we had all the fields in the QBE grid. However, you can specify:

- Which fields to have in the QBE by dragging individual fields from the field list. This is important when using aggregate functions described latter.
- Which fields to display in the results by clicking the show box in each of the fields in the QBE grid.
- Order of the fields in the results by moving ('clicking & dragging') a fields position in the QBE grid.

3.2 Sorting

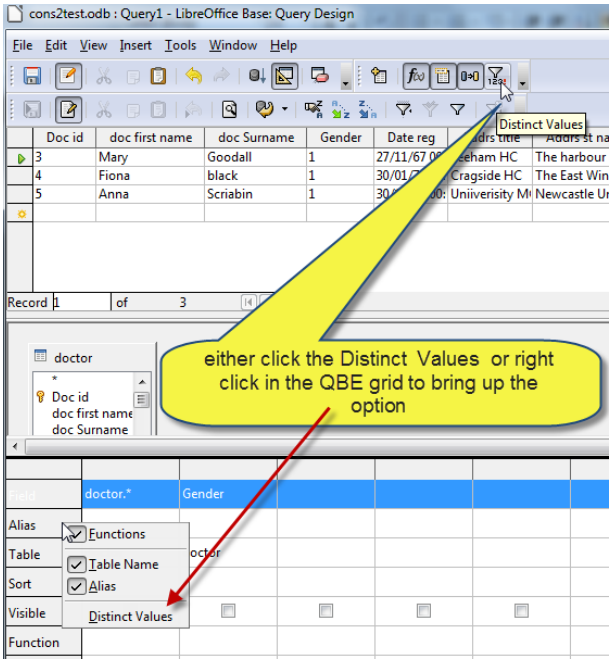


You can specify which field(s) to sort the result records on by selecting the appropriate option in the sort field of the QBE grid.

Exercise 2. sorting options

Create a query using the **Patient** table and inspect the sort options in the QBE grid (shown opposite).

3.3 Removing duplicate values in the results table



You can remove duplicate values in the results by setting the appropriate value in the query. You can set the distinct Values property of the query by either:

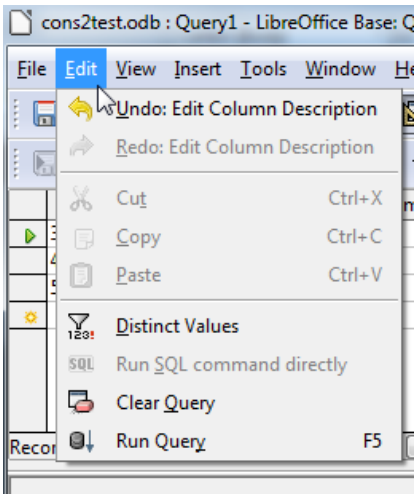
Right mouse click somewhere in the QBE grid, then click on the Distinct Values option. *or*

Click the Distinct Values icon on the speed bar.

3.4 Clearing the QBE grid

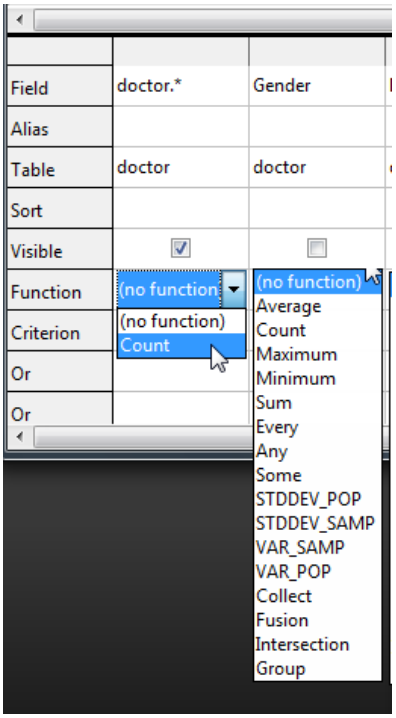
To clear all the fields in one swoop choose the menu option Edit -> Clear Query.

To clear a particular field in the QBE grid highlight it by clicking at the top of it then choose the menu option edit -> cut or delete or you can highlight it then press the delete key.



4. Summary Functions

Summary functions produce a single value from a set of values. The screenshot below lists a few of the most common offered by LibreOffice Base, notice that the options available vary between the field you select:



LibreOffice Base Summary functions	
Function	**What it does**
Sum	Total of the values in a field.
Average	Average of the values in a field.
Minimum	Lowest value in a field.
Maximum	Highest value in a field.
Count(*) and Count(field name)	Number of values in a field (*= including null values).
STDDEV_POP, STDDEV_SAMP	Standard deviation (population and sample estimates) of the values in a field.
VAR_POP, VAR_SAMP	Variance of the values in a field (population and sample estimates).
any, some	(Boolean fields only) Returns True if one [or more?] values True otherwise false
every	(Boolean fields only) Returns True if ALL values True otherwise false
collect	
fusion	
interaction	
Group	Define the groups you want totals for. For example, to show total number of patients by doctor, select Group By for the doctor ID field. And use the count summery function in another copy of the field.

Microsoft Access Summary functions	
Function	What it does
Sum	Total of the values in a field.
Avg	Average of the values in a field.
Min	Lowest value in a field.
Max	Highest value in a field.
Count	Number of values in a field (not counting null values).
StDev	Standard deviation of the values in a field.
Var	Variance of the values in a field.
First	First value in a field.
Last	Last value in a field.
Group By	Define the groups you want totals for. For example, to show total number of patients by doctor, select Group By for the doctor ID field. And use the count summery function in another copy of the field.
To use summary functions in Access97 you must have the 'Totals' row visible in the QBE grid. To do this (when in the Query design window) choose the menu option ⌫ view then click on the totals menu item.	

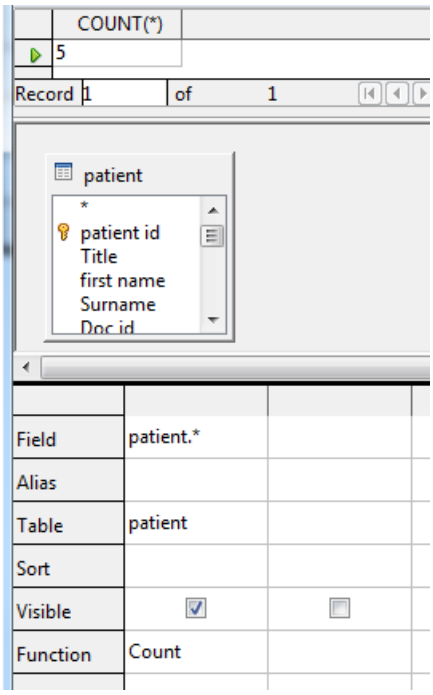
4.1 Counting the number of records

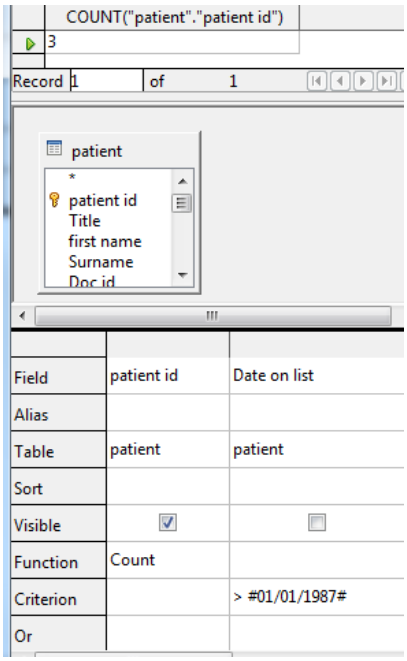
Exercise 3. Counting all the patient records

Clear the QBE grid if necessary.

Ensure you have the patient table from the cons2 database in the query window.

Drag the patient id field into the QBE grid from the patient field list (add the table to the query first if necessary). Select count from the selection box in the total cell.





Important: This query, like most in this chapter, only works if you have the specified fields in the QBE grid. Do not add all the fields to the QBE grid.

Exercise 4. Counting a subset of patient records

We will count the number of patients with a patient ID (i.e. those on the list) excluding those who joined the practice prior to 1987 or have no patient id value. To run this query we setup the QBE grid as shown opposite.

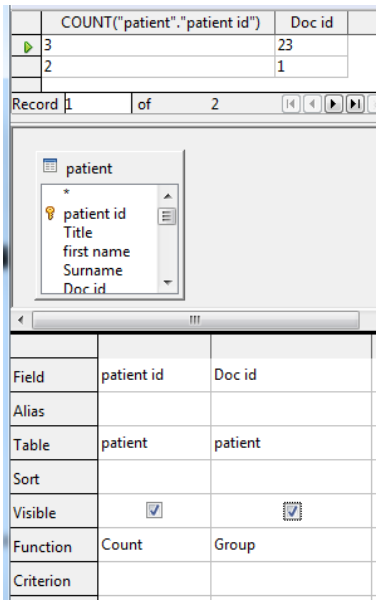
Important:

- The visible property of the Date on list field must be unchecked.
- The function cell of the Date on list field must be empty

While the above examples demonstrate how easy it is to produce counts for a sub set of records it should be mentioned that the above count function excludes null values from the analysis. If you have reason to believe that some of the records involved in the count were blank = null or the result is questionably or less than you expected you will need to use the count(*) function. You will learn how to do this when we discuss calculated fields latter. For now you can simply open the table and look at the 'record count' at the bottom of the screen in datasheet view.

4.2 Grouping

The usefulness of the above function is greatly enhanced when combined with a 'grouping function'. This allows you to produce totals for different groups of records. For example, say we wanted to count the number of patients each doctor had.



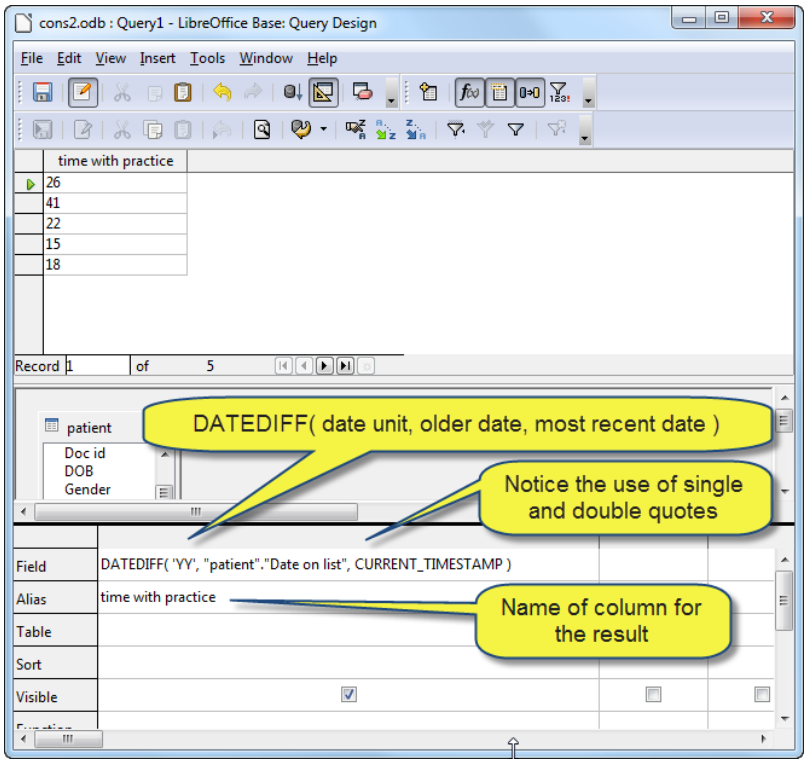
Exercise 5. Counting all the patients registered with each doctor

Setup the QBE grid as shown opposite. Remember to press the F4 key (possible twice) to run the query.

The next section will now look at adding calculated fields to a set of results.

5. Adding calculated fields - dates

It is frequently necessary to create extra fields based upon calculations in other fields, such as present age from date of birth. To achieve this it is necessary to develop formulae ('calculations'), called "**expressions**" in computing parlance. The exercise below describes how to create a calculated field:



Exercise 6. Finding the length of time each patient has been registered with a doctor

Suppose you want to find out how long each of your patients have been with you. Considering the patient table you know it contains the date each joined (i.e. the "date on list" field) you therefore want to create a new field in your query results which provides the necessary value.

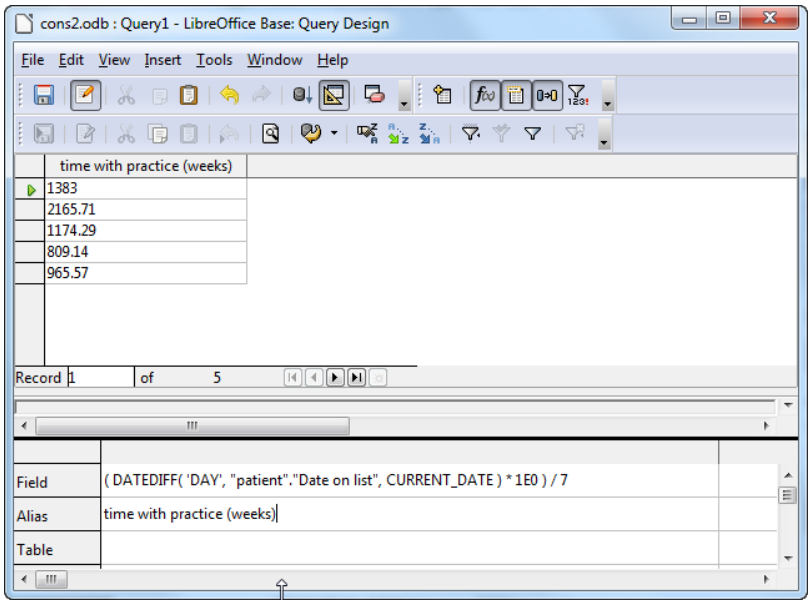
Setup the query grid as shown opposite to run the query.

Important things to note:

- the DATEDIFF function requires three values.
- time/date unit such as YEAR (YY), MONTH, DAY, HOUR, MINUTE (MI) and SECOND, I find it is better to use these terms rather than the cryptic mm, yy etc. The value needs to be in single quotes.
- the first date is the oldest one (the smaller value), and in this instance is the value from a field in the table, notice the way it is written; table_name.field_name with double quotes for both the table and field names.
- CURRENT_TIMESTAMP or CURRENT_TIME or CURRENT_DATE all provide the current date/time. Notice here I have not used any quotes.

The Alias cell allows you to specify a name for the result column.

You will notice that the above query has returned whole values. to get non integer values such as a certain number of decimal places you can adapt the expression in the FIELD cell a number of ways.



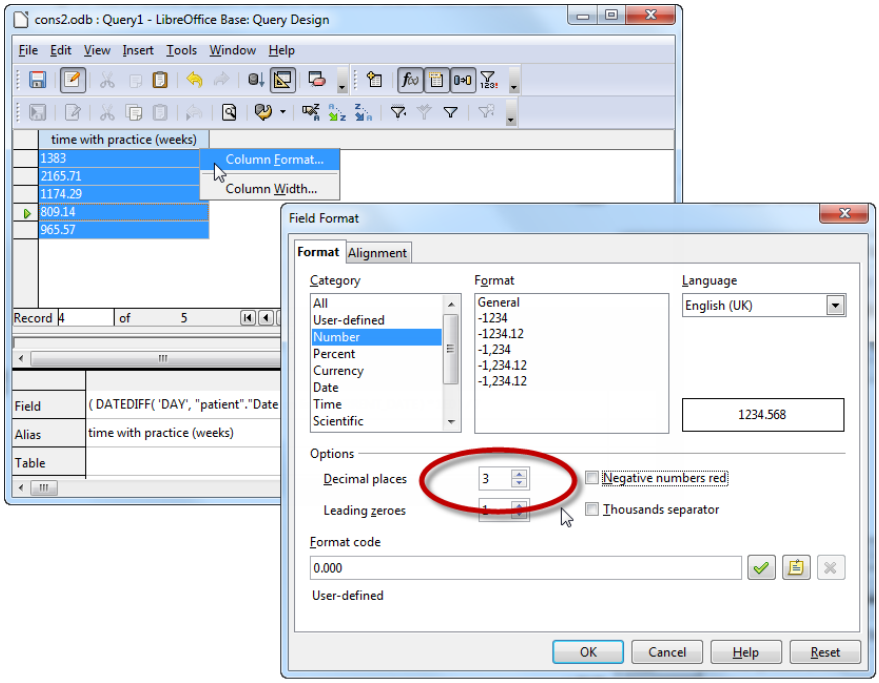
Firstly we can modify the expression by adding the **1E0** (0= zero) expression basically we multiple the result by a real number and then all subsequent calculations will result in a real rather than an integer value.

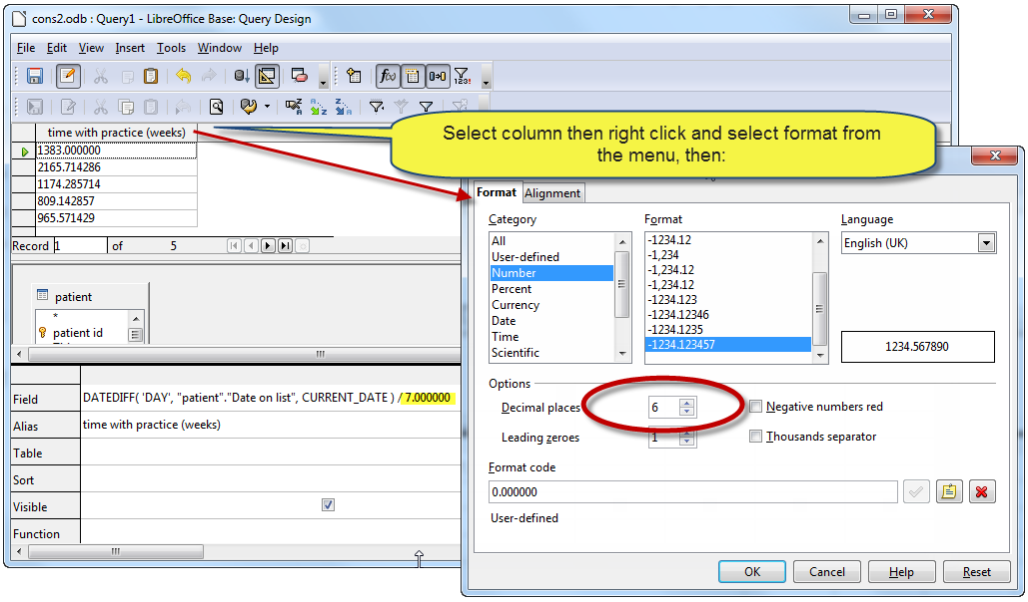
For example say we wanted the number of weeks to the accuracy of a day we could use the following expression:

(DATEDIFF('DAY', "patient"."Date on list", CURRENT_DATE) * 1E0) / 7

While this works, you might not see the result, to ensure you can you need to specify the FORMAT of the results column, you do this by selecting the column and then right clicking on the top cell which brings up the field format dialog box where you can specify the number of decimal places to display.

The second way is to type a value in the expression with a certain number of decimal places in it, for example by typing 7.00 in the above expression (and removing the 1E0) will produce results to two decimal places, and typing 7.000000 would produce calculations to 6 decimal places, this technique is shown below, again we need to set the format display option in the result column.





Conversely by just typing '7' result in producing results to zero decimal places of accuracy.

Exercise 7. Investigating formatting options for results

Repeat the various steps described above.

Re-run the query sorting the results by length of time with the practice (descending).

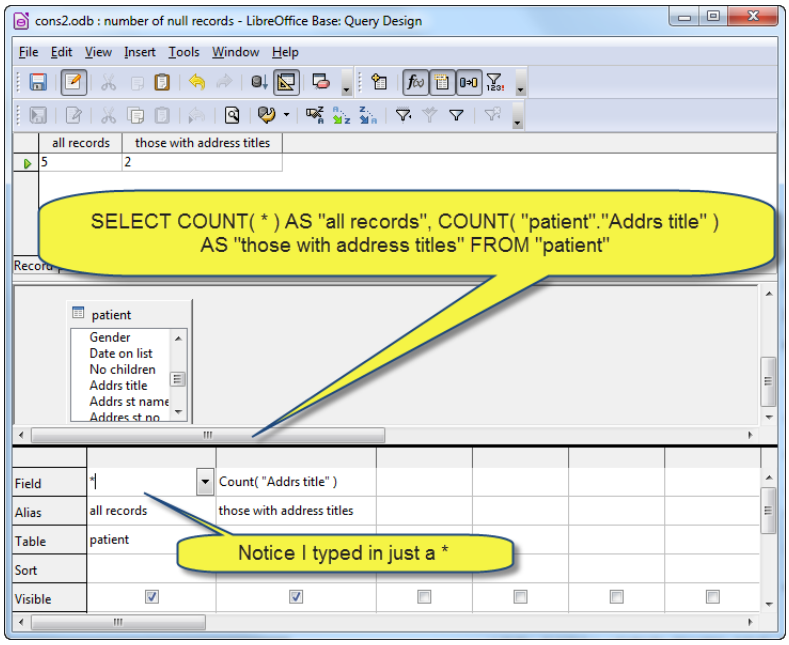
This is for those of you who like brainteasers find the average, minimum, maximum and standard deviation for patients time with practice in number of days.

The answer can be found at the end of this chapter.

The next section introduces you to some other ways of manipulating fields to provide valuable information, including ways of finding the number of records that are empty ('null'), and extracting parts of fields.

6. Finding the number of records that are empty ('null')

In the previous section we calculated the total number of records in the table including those with 'null' for the field by selecting the count option in the function cell of the QBE grid. Looking at the patient table we see that two out of the five records have a value in the 'addrs title' field. We will use the 'Count(fieldname)' function to count the total number of non empty records for this field. This is done by setting up the QBE grid setup as shown below.



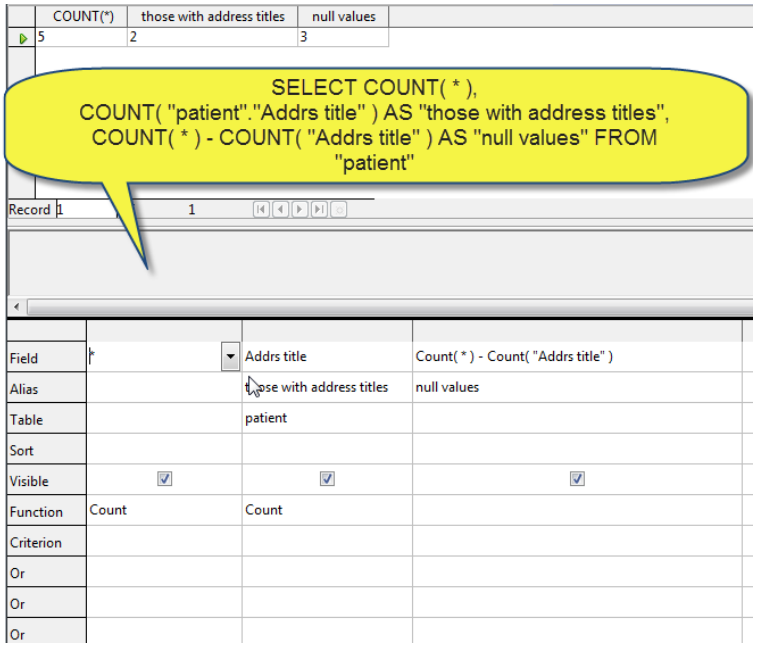
Strangely while I only entered the fieldname in the FIELD cell the resulting SQL has added the table name and in fact if you do enter the expression:

Count("patient"."Addrs title")

This also works fine.

The 'all records' field gives the number of all records, including ones with a empty Addrs title field while the count function includes only those with values in the Addrs title field.

You can also create another result field which provides the difference between the two values, that is to say it produces a count of null values for the Addrs title field as shown opposite.



Exercise 8. Investigating empty fields

Repeat the various steps described above.

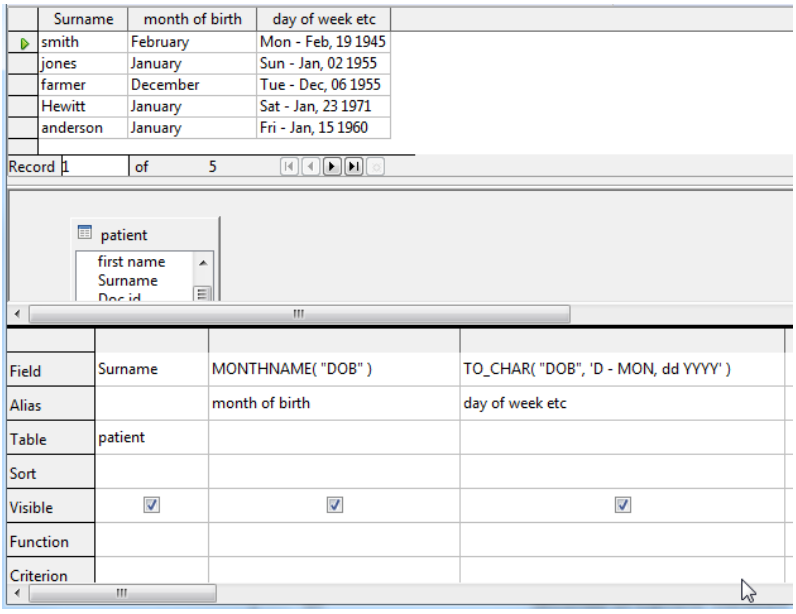
7. Date and Time functions

We have seen used in the above sections various functions, and Base contains over a hundred of which a subset relate to Date and Time manipulation, because these are very useful I have provided the table below for reference purposes as the LibrOffice Base documentation is difficult to find.

Function Name	Comments
DATEDIFF(string, datetime1, datetime2)	Returns the count of units of time elapsed from datetime1 to datetime2. The string indicates the unit of time and can have the following values 'ms'='millisecond', 'ss'='second','mi'='minute','hh'='hour', 'dd'='day', 'mm'='month', 'yy' = 'year'. Both the long and short form of the strings can be used. DATEDIFF('dd', '2007-08-01', '2007-09-01') = 31
DAYNAME(date)	Returns the name of the day of the week. DAYNAME('2007-09-01') = Saturday
DAYOFMONTH(date)	Returns the day of the month (1-31) DAYOFMONTH('2007-09-01') = 1
DAYOFWEEK(date)	Returns the day of the week (1 means Sunday) DAYOFWEEK('2007-09-01') = 7
DAYOFYEAR(date)	Returns the day of the year (1-366) DAYOFYEAR('2007-09-01') = 244
HOUR(time)	Return the hour (0-23) HOUR('21:16:04')
MINUTE(time)	Returns the minute (0-59) MINUTE('21:16:04') = 16
MONTH(date)	Returns the month (1-12) MONTH('2007-09-01') = 9
MONTHNAME(date)	Returns the name of the month, MONTHNAME('2007-09-01') = September
QUARTER(date)	Returns the quarter (1-4), with the new year starting in January QUARTER('2007-09-01') = 3
SECOND(time)	Returns the second (0-59) SECOND(CURRENT_TIME) MAY EQUAL 6
WEEK(date)	Returns the week of this year (1-53) WEEK('2007-09-01') = 35
YEAR(date)	Returns the year YEAR('2007-09-01') = 2007
CURRENT_DATE	Returns the current date CURRENT_DATE = 09/01/07
CURRENT_TIME	Returns the current time CURRENT_TIME = 10:44:28 PM
CURRENT_TIMESTAMP	Returns the current timestamp CURRENT_TIMESTAMP = 09/01/07 10:34 PM
TO_CHAR(datetime, format String)	Returns a string from a date or datetime, based on the format mask Following is a list of valid format mask character sequences,

Chr seq	returned value	example
YYYY	4 digit year	TO_CHAR(CURRENT_TIMESTAMP, 'YYYY') = 2007
YYY YY Y	Last 3, 2 or 1 digits of year	TO_CHAR(CURRENT_TIMESTAMP, 'YY') = 07
IYYY IYY IY	Last 4, 3, 2 digits of ISO year	TO_CHAR(CURRENT_TIMESTAMP, 'IY') = 07
MM	Month (01-12; JAN = 01)	TO_CHAR(CURRENT_TIMESTAMP, 'MM') = 09
MON	Abbreviated name of month	TO_CHAR(CURRENT_TIMESTAMP, 'MON') = Sep
MONTH	Name of month, padded with blanks to length of 9 characters.	TO_CHAR(CURRENT_TIMESTAMP, 'MONTH') = September
w	Week of year	TO_CHAR(CURRENT_DATE, 'w') = 35
W	Week of month	TO_CHAR(CURRENT_DATE, 'W') = 1
IW	Week of year (1-52 or 1-53) based on the ISO standard.	TO_CHAR(CURRENT_DATE, 'IW') = 35
d	Day of week (1-7)	TO_CHAR(CURRENT_DATE, 'd') = 1
D	Abbreviation for day of week	TO_CHAR(CURRENT_DATE, 'D') = Sat
DD dd	Day of month as 2 digits	TO_CHAR(CURRENT_DATE, 'DD') = 01
DDD	Day of year	TO_CHAR(CURRENT_DATE, 'DDD') = 244
H	Hour of day 0 - 23	TO_CHAR(CURRENT_TIME, 'H') = 23
HH	Hour of day 0-11	TO_CHAR(CURRENT_TIME, 'HH') = 11
m	Minute of current hour	TO_CHAR(CURRENT_TIME, 'm') = 48
s	Seconds of current minute	TO_CHAR(CURRENT_TIME, 's') = 33
a	AM or PM	TO_CHAR(CURRENT_TIME, 'a') = PM
All other characters	LITERALS	TO_CHAR(CURRENT_DATE, 'D - MON, dd YYYY') = Sat - Sep, 01 2007

The screenshot below shows some of the above date time functions being used:



Surname	month of birth	day of week etc
smith	February	Mon - Feb, 19 1945
jones	January	Sun - Jan, 02 1955
farmer	December	Tue - Dec, 06 1955
Hewitt	January	Sat - Jan, 23 1971
anderson	January	Fri - Jan, 15 1960

Record 1 of 5

patient
first name
Surname

Field	Surname	MONTHNAME("DOB")	TO_CHAR("DOB", 'D - MON, dd YYYY')
Alias		month of birth	day of week etc
Table	patient		
Sort			
Visible	☑	☑	☑
Function			
Criterion			

Many of the date/time functions are to do with either extracting part of a fiedl value or reformatting it in some way, so that you can have nicely presented reports showing the actual day of the week and month etc. rather than purely numbers, which are obviously the most efficient way of storing the information.

Notice in the examples opposite the use of single and double quotes.

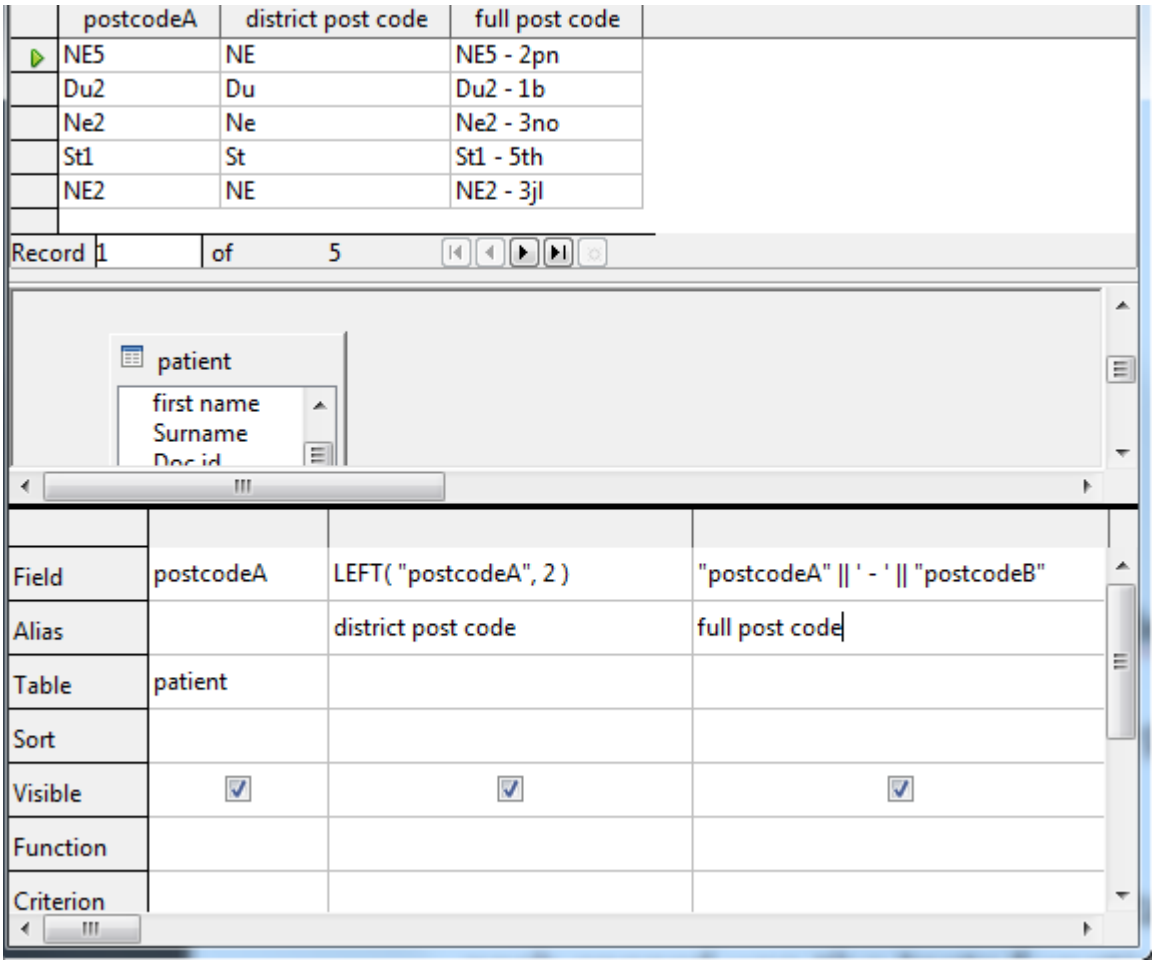
Double quotes around the field name and single quotes for the date/time formatting options, also notice how the use is made of the dash character and spaces.

8. String functions

The following table, also taken from the Base help documentation, provides similar descriptions for several string (i.e. text) functions.

Function Name	Comments
CHAR(c)	Returns the character string corresponding to the given ASCII (or Unicode) value C. **Note:** In some SQL CLI implementations, a null is returned if the range is outside 0..255. In HSQLDB, the corresponding Unicode character is returned unchecked. CHAR(79) = O
CHAR_LENGTH(str)	Returns the length of the string in characters CHAR_LENGTH('ONE') = 3
CONCAT(str1,str2)	Returns str1 + str2 CONCAT('ONE', 'HUNDRED') = ONEHUNDRED
LCASE(s)	Converts s to lower case LCASE('ONE') = one
LEFT(s,count)	Returns the leftmost count of characters of s **Note:** boundry conditions are handled in the following order of precedence: if is null, then null is returned if count is less than 1, then a zero-length String is returned if count is greater than the length of s, then a copy of s is returned - requires double quoting - use SUBSTRING() instead LEFT('ONE', 2) = ON
LENGTH(s)	Returns the number of characters in s LENGTH('ONE') = 3
LTRIM(s)	Removes all leading blanks in s LTRIM(' ONE ') = "ONE "
LOWER(s)	Converts s to lower case LOWER('ONE') = one
REPEAT(s,count)	Returns s repeated count times REPEAT('X', 4) = XXXX
REPLACE(s,replace,s2)	Replaces all occurrences of replace in s with s2 REPLACE('WHAT XXXX BROWN COW.', 'XXXX', 'NOW') = WHAT NOW BROWN COW.
RIGHT(s,count)	Returns the rightmost count of characters of s **Note:** Boundry conditions are handled in the following order of precedence: if is null, null is returned if count is less than one, a zero-length String is returned if count is greater than the length of s, a copy of s is returned RIGHT('THIS AND THAT', 4) = THAT
RTRIM(s)	Removes all trailing spaces RTRIM(' ONE ') = " ONE"
SPACE(count)	Returns a string consisting of count spaces SPACE(4) = " "
SUBSTR(s,start[,len]) **SUBSTRING(s,start[,len])**	Returns the substring starting at start (1=left) with length len **Note:** The rules for boundary conditions on s, start and length are, in order of precedence: 1.) if s is null, return null 2.) If length is less than 1, return null. 3.) If start is 0, it is treated as 1. 4.) If start is positive, count from the beginning of s to find the first character position. 5.) If start is negative, count backwards from the end of s to find the first character. 6.) If, after applying 2.) or 3.), the start position lies outside s, then return null 7.) if length is omitted or is greater than the number of characters from the start position to the end of s, return the remainder of s, starting with the start position. SUBSTR('HERE I AM', 6, 1) = I ; SUBSTR('HERE I AM', 6) = I AM
SUBSTRING(s FROM start [FOR len])	Alternate syntax where s may be a string expression SUBSTRING ('HERE I AM' FROM 6 FOR 1) = I ; SUBSTRING ('HERE I AM' FROM 6) = I AM
TRIM([[LEADING | TRAILING | BOTH] [TRIMSTR]] FROM s)	Returns the character sequence s, with the leading, trailing or both the leading and trailing occurences of the first character of the character sequence trimstr removed. If trimstr is not supplied SPACE is used. TRIM(BOTH FROM ' ONE ') = "ONE" ; TRIM (BOTH 'O' FROM 'OONEOO') = NE *** Note** with escape processing DISABLED the command may be abbreviated to TRIM (FROM ' ONE ') AS "OUTPUT" = "ONE" - An alias must be used in this case also
UCASE(s)	Converts s to upper case UCASE('one') = ONE
UPPER(s)	Converts s to upper case UPPER('one') = ONE

The screenshot below shows some examples of the above functions:



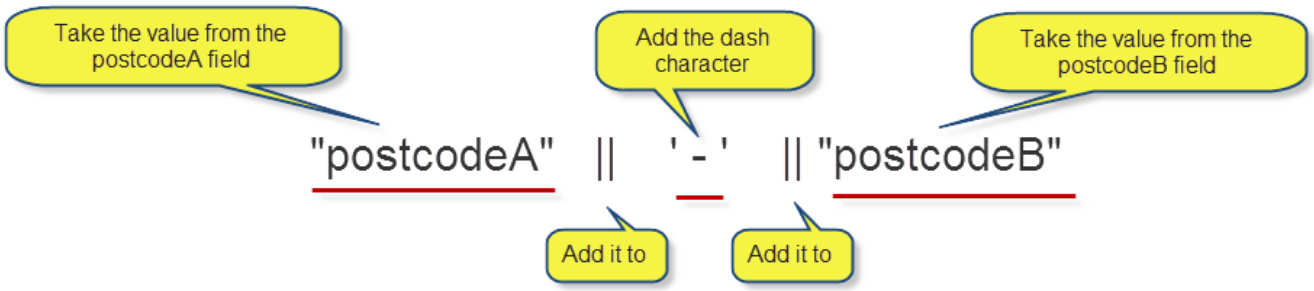
postcodeA	district post code	full post code
NE5	NE	NE5 - 2pn
Du2	Du	Du2 - 1b
Ne2	Ne	Ne2 - 3no
St1	St	St1 - 5th
NE2	NE	NE2 - 3jl

Record 1 of 5

patient
first name
Surname
Doc id

Field	postcodeA	LEFT("postcodeA", 2)	"postcodeA" || ' - ' || "postcodeB"
Alias		district post code	full post code
Table	patient		
Sort			
Visible	☑	☑	☑
Function			
Criterion			

Notice the use of the string concatenation operator "||" basically it allows you to add things together, also you often need to add spaces or dashes, as I have done above to stop you getting just a series of 5 or 6 characters for each record.



Exercise 9. time/date and string functions

Replicate the various examples I have given on the last two pages. Try experimenting with your own enhancements.

8.1 Revision Exercises

The cons3 database contains, besides the doctor records and patient records, a set of records in the episode table.

Episode Table

episode id	Patient ID	Doc id	Date seen	urgency	systolic	diastolic
1	5	1	01/01/95	yes	240	130
2	5	1	05/01/95	No	235	135
3	5	1	11/01/95	No	180	100
4	5	1	17/01/95	No	170	95
5	5	1	28/01/95	No	175	95
6	5	1	10/02/95	No	170	100
7	5	1	27/02/95	No	170	95
8	5	1	20/03/95	No	180	90
9	4	1	03/02/96	No	145	80
10	4	1	23/02/96	No	150	85
11	1	23	23/04/94	yes	130	70
12	1	1	06/05/94	yes		
13	2	2	13/04/73	yes		
14	3	3	02/06/90	yes	190	90
15	4	2	03/03/96	No	165	85

The episode table provides details of each visit. The visit has a unique ID, doctor ID and a patient id. Notice that this means any patient can see any doctor (it does not need to be the doctor whose list they are on). Answers are provided at the end of this chapter.

The above records in the episode table, in the cons3 database, should be used for the following exercises.

1. Calculate the number of visits to each doctor. Arrange the fields in the QBE grid to show Doctor ID following by the count
2. Re-run the above query this time having the results ranked from highest to lowest
3. Find the date of the first and last episode for each doctor, ranked by doctor id from lowest to highest
4. Find the minimum, average, sample standard deviation (rather than population) and maximum diastolic BP recorded, display the average and standard deviation values to 4 decimal places.
5. Find for each doctor the number of urgent and routine episodes
6. How many episodes where there in 1995
7. Find the average diastolic and systolic BP for each doctor. Also indicate the number of episodes that make up each average figure, what is the problem with using the number of episodes field for this purpose which field or field might be a more useful choice?

8.2 Summary

This practical chapter has investigated several very useful QBE functions in Base. This included common summary statistics including the min and max functions. We also used these last functions with date fields. The section has also looked at a few of the ways that data can be displayed using various options such as field ordering and record ordering. Calculated fields were demonstrated along with a list of useful date / text manipulation functions.

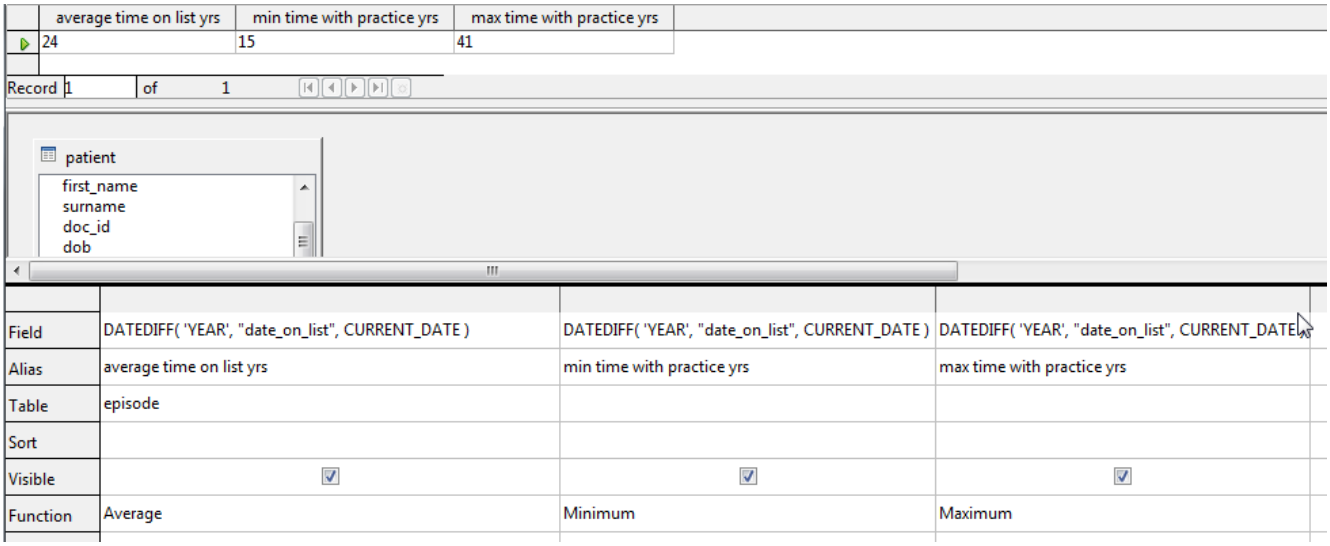
8.3 Check what you have learnt

Now go back to the beginning of the material for the chapter and read through the 'Learning outcomes check list'. How many can you tick? If you are not sure about any in particular read through the relevant sections again.

9. Answers to selected exercises

Answer to **Exercise 7. Investigating formatting options for results**

Some summary statistics about the patients at the practice:



average time on list yrs	min time with practice yrs	max time with practice yrs
24	15	41

Record 1 of 1

patient
first_name
surname
doc_id
dob

Field	DATEDIFF('YEAR', "date_on_list", CURRENT_DATE)	DATEDIFF('YEAR', "date_on_list", CURRENT_DATE)	DATEDIFF('YEAR', "date_on_list", CURRENT_DATE
Alias	average time on list yrs	min time with practice yrs	max time with practice yrs
Table	episode		
Sort			
Visible	☑	☑	☑
Function	Average	Minimum	Maximum

Answers to revision exercises on page 12

Exercise 1

doc _d	number of episodes for doctor
1	11
23	1
2	2
3	1

Record 1 of 4

episode
*

Field	doc _d	episode_id
Alias		number of episodes for doctor
Table	episode	episode
Sort		
Visible	☑	☑
Function	Group	Count
Criterion		

Exercise 2

doc _d	number of episodes for doctor
1	11
2	2
3	1
23	1

Record 1 of 4

episode
*
episode_id
patient_id
doc _d
date_seen
urgency

Field	doc _d	episode_id
Alias		number of episodes for doctor
Table	episode	episode
Sort		descending
Visible	☑	☑
Function	Group	Count
Criterion		

Exercise 3

doc _d	first date seen	last date seen
1	06/05/94	23/02/96
2	13/04/73	03/03/96
3	02/06/90	02/06/90
23	23/04/94	23/04/94

Record 1 of 4

episode
*
episode_id
patient_id
doc _d
date_seen

Field	doc _d	date_seen	date_seen
Alias		first date seen	last date seen
Table	episode	episode	episode
Sort	ascending		
Visible	☑	☑	☑
Function	Group	Minimum	Maximum

Exercise 4

min	mean	SD sample	maximun
70	96.0000	18.1606	135

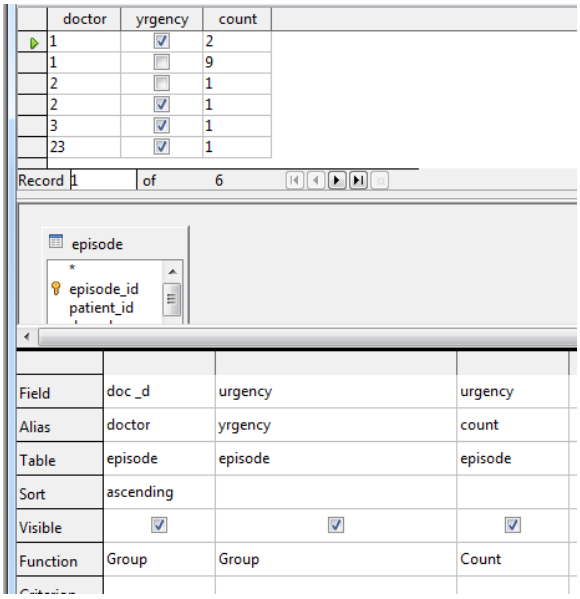
Record 1 of 1

episode
*
episode_id
patient_id
doc _d
date_seen
urgency

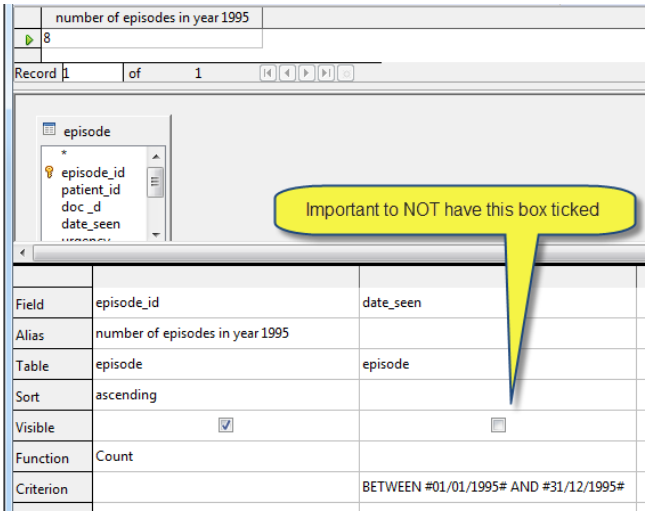
To see 4 decimal places, select the column then right mouse click and edit the formatting options

Field	diastolic	diastolic	diastolic	diastolic
Alias	min	mean	SD sample	maximun
Table	episode	episode	episode	episode
Sort	ascending			
Visible	☑	☑	☑	☑
Function	Minimum	Average	STDDEV_SAMP	Maximum
Criterion				

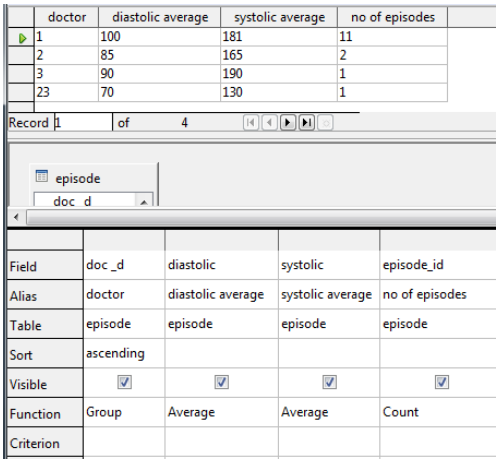
Exercise 5



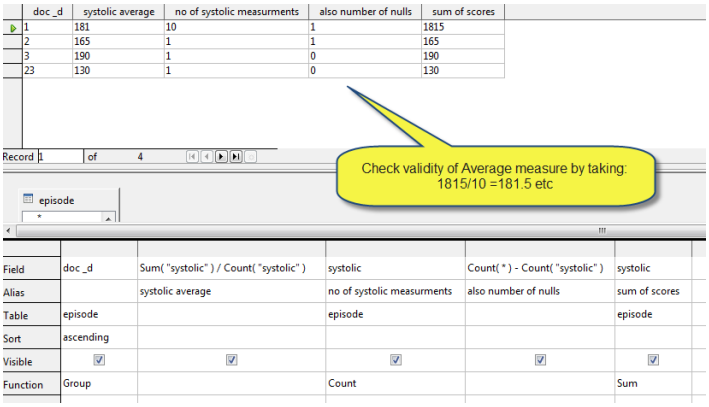
Exercise 6



Exercise 7



Exercise 7b



Note: In the last exercise we used the AVERAGE measure sometimes there is a danger that the average is incorrectly calculated or example we have two episodes for doctor id =2 but only one of them has blood pressure readings. If we just divided those single values by the number of episodes for the particular doctor rather than by the number of episodes for the doctor that had BP recordings we would end up with invalid averages. Luckily the AVERAGE function in Base does this but it is always both checking (see exercise 7b above).

Similarly there is a danger that we have not checked that the result has any values past the decimal placed if we have not reformatted the results column.

10. Web links

SQL links: http://www.wiscorp.com/SQLStandards.html