Calc Guide

Chapter 13
Calc as a Simple Database

A guide for users and macro programmers

Copyright

Authors

Andrew Pitonyak

Feedback

Maintainer: Andrew Pitonyak [andrew@pitonyak.org]
Please direct any comments or suggestions about this document to: authors@documentation.openoffice.org

Publication date and software version

Published 8 June 2010. Based on OpenOffice.org 3.2.

Contents

Copyright....2
Introduction....4
Associating a range with a name....5
  Named range....5
  Database range....10
Sorting....11
Filters....13
  Auto filters....13
  Standard filters....14
  Advanced filters....18
  Manipulating filtered data....21
Calc functions similar to database functions....21
  Count and sum cells that match conditions: COUNTIF and SUMIF. 23
  Ignore filtered cells using SUBTOTAL....24
  Using formulas to find data....25
    Search a block of data using VLOOKUP....25
    Search a block of data using HLOOKUP....26
    Search a row or column using LOOKUP....27
    Use MATCH to find the index of a value in a range....27
    Examples....28
  ADDRESS returns a string with a cell’s address....29
  INDIRECT converts a string to a cell or range....30
  OFFSET returns a cell or range offset from another....30
  INDEX returns cells inside a specified range....31
Database-specific functions....33
Conclusion....34

Introduction

A Calc document is a very capable database, providing sufficient functionality to satisfy the needs of many users. This chapter presents the capabilities of a Calc document that make it suitable as a database tool. Where applicable, the functionality is explained using both the GUI (Graphical User Interface) and macros.

Note Although this document was initially created for macro programmers, the content should be accessible to all users. If you do not use macros, then skip those portions that deal with macros. On the other hand, if you want to learn more about macros, be certain to check out the book *OpenOffice.org Macros Explained*.

In a database, a record is a group of related data items treated as a single unit of information. Each item in the record is called a field. A table consists of records. Each record in a table has the same structure. A table can be visualized as a series of rows and columns. Each row in the table corresponds to a single record and each column corresponds to the fields. A spreadsheet in a Calc document is similar in structure to a database table. Each cell corresponds to a single field in a database record. For many people, Calc implements sufficient database functionality that no other database program or functionality is required.

While teaching, a spreadsheet might be used as a grading program. Each row represents a single student. The columns represent the grades received on homework, labs, and tests (see Table 1). The strong calculation capability provided in a spreadsheet makes this an excellent choice.

Table 1. Simple grading spreadsheet

	A	**B**	**C**	**D**	**E**	**F**	**G**
1	Name	Test 1	Test 2	Quiz 1	Quiz 2	Average	Grade
2	Andy	95	93	93	92	93.25	
3	Betty	87	92	65	73	79.25	
4	Bob	95	93	93	92	93.25	
5	Brandy	45	65	92	85	71.75	
6	Frank	95	93	85	92	91.25	
7	Fred	87	92	65	73	79.25	

	A	B	C	D	E	F	G
8	Ilsub	70	85	97	79	82.75	
9	James	45	65	97	85	73	
10	Lisa	100	97	100	93	97.5	
11	Michelle	100	97	100	65	90.5	
12	Ravi	87	92	86	93	89.5	
13	Sal	45	65	100	92	75.5	
14	Ted	100	97	100	85	95.5	
15	Tom	70	85	93	65	78.25	
16	Whil	70	85	93	97	86.25	

Tip Although the choice to associate a row to a record rather than a column is arbitrary, it is almost universal. In other words, you are not likely to hear someone refer to a column of data as a single database record.

Associating a range with a name

In a Calc document, a range refers to a contiguous group of cells containing at least one cell. You can associate a meaningful name to a range, which allows you to refer to the range using the meaningful name. You can create either a *database range*, which has some database-like functionality, or a *named range*, which does not. A name is usually associated with a range for one of three reasons:

1) Associating a range with a name enhances readability by using a meaningful name.
2) If a range is referenced by name in multiple locations, you can point the name to another location and all references point to the new location.
3) Ranges associated to a name are shown in the Navigator, which is available by pressing the *F5* key or clicking on the icon. The Navigator allows for quick navigation to the associated ranges.

Named range

The most common usage of a named range is, as its name implies, to associate a range of cells to a meaningful name. For example, I created a range named *Scores*, and then I used the following equation: =SUM(Scores). To create a named range, select the range to define. Use **Insert > Names > Define** to open the Define Names dialog. Use the Define Names dialog to add and modify one named range at a time.

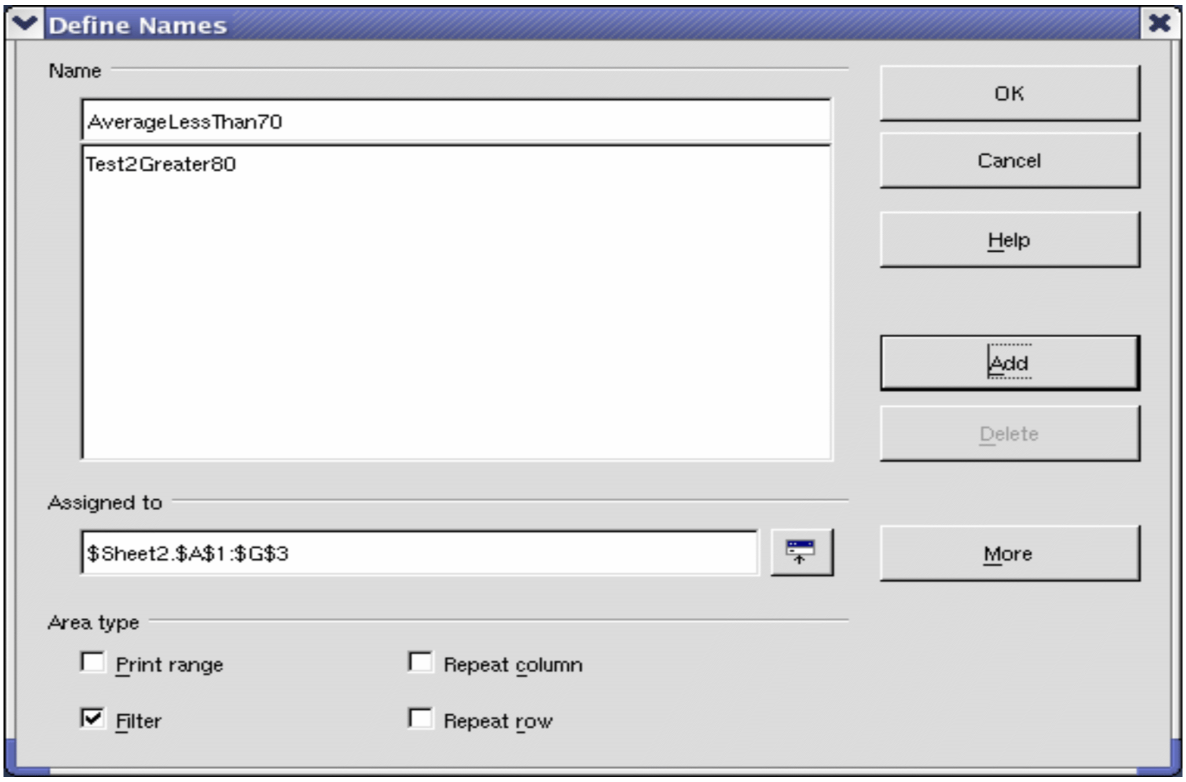


Figure 1. Define a named range.

In a macro, a named range is accessed, created, and deleted using the NamedRanges property of a Calc document. Use the methods hasByName(name) and getByName(name) to verify and retrieve a named range. The method getElementNames() returns an array containing the names of all named ranges. The NamedRanges object supports the method addNewByname, which accepts four arguments; the name, content, position, and type. The macro in Listing 1 creates a named range, if it does not exist, that references a range of cells.

Listing 1. Create a named range that references $Sheet1.$B$3:$D$6.

```
Sub AddNamedRange()
  Dim oRange     ' The created range.
  Dim oRanges    ' All named ranges.
  Dim sName$     ' Name of the named range to create.
  Dim oCell      ' Cell object.
  Dim s$

  sName$ = "MyNRange"
  oRanges = ThisComponent.NamedRanges
  If NOT oRanges.hasByName(sName$) Then
```

```
      REM I can obtain the cell address by obtaining the cell
      REM and then extracting the address from the cell.
      Dim oCellAddress As new com.sun.star.table.CellAddress
      oCellAddress.Sheet = 0     'The first sheet.
      oCellAddress.Column = 1    'Column B.
      oCellAddress.Row = 2       'Row 3.

      REM The first argument is the range name.
      REM The second argument is formula or expression to
      REM use. The second argument is usually a string that
      REM defines a range.
      REM The third argument specifies the base address for
      REM relative cell references.
      REM The fourth argument is a set of flags that define
      REM how the range is used, but most ranges use 0.
      REM The fourth argument uses values from the
      REM NamedRangeFlag constants (see Table 2).
      s$ = "$Sheet1.$B$3:$D$6"
      oRanges.addNewByName(sName$, s$, oCellAddress, 0)
    End If
    REM Get a range using the created named range.
    oRange = ThisComponent.NamedRanges.getByName(sName$)

    REM Print the string contained in cell $Sheet1.$B$3
    oCell = oRange.getReferredCells().getCellByPosition(0,0)
    Print oCell.getString()
  End Sub
```

The method addNewByname() accepts four arguments; the name, content, position, and type. The fourth argument to the method addNewByName() is a combination of flags that specify how the named range will be used (see Table 2). The most common value is 0, which is not a defined constant value.

Table 2. com.sun.star.sheet.NamedRangeFlag constants.

Value	Name	Description
1	FILTER_CRITERIA	The range contains filter criteria.
2	PRINT_AREA	The range can be used as a print range.
4	COLUMN_HEADER	The range can be used as column headers for printing.
8	ROW_HEADER	The range can be used as row headers for printing.

The third argument, a cell address, acts as the base address for cells referenced in a relative way. If the cell range is not specified as an absolute address, the referenced range will be different based on where in the spreadsheet the range is used. The relative behavior is illustrated in Listing 2, which also illustrates another usage of a named range—defining an equation. The macro in Listing 2 creates the named range **AddLeft**, which refers to the equation A3+B3 with C3 as the reference cell. The cells A3 and B3 are the two cells directly to the left of C3, so, the equation `=AddLeft()` calculates the sum of the two cells directly to the left of the cell that contains the equation. Changing the reference cell to C4, which is below A3 and B3, causes the AddLeft equation to calculate the sum of the two cells that are to the left on the previous row.

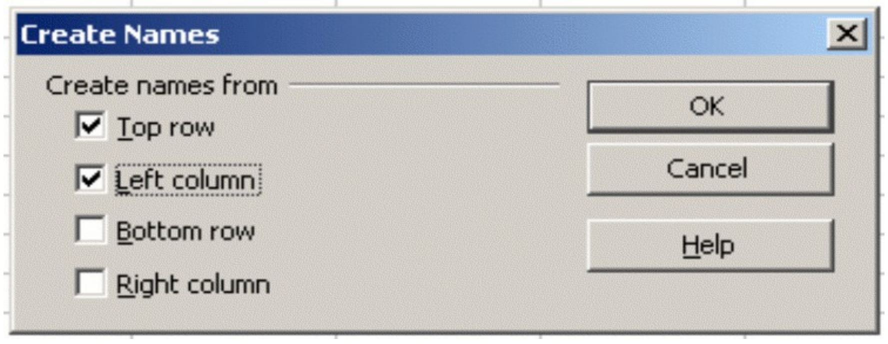
Listing 2. Create the AddLeft named range.

```
Sub AddNamedFunction()
  Dim oSheet          'Sheet that contains the named range.
  Dim oCellAddress    'Address for relative references.
  Dim oRanges         'The NamedRanges property.
  Dim oRange          'Single cell range.
  Dim sName As String 'Name of the equation to create.

  sName = "AddLeft"
  oRanges = ThisComponent.NamedRanges
  If NOT oRanges.hasByName(sName) Then
    oSheet = ThisComponent.getSheets().getByIndex(0)
    oRange = oSheet.getCellRangeByName("C3")
    oCellAddress = oRange.getCellAddress()
    oRanges.addNewByName(sName, "A3+B3", oCellAddress, 0)
  End If
End Sub
```

Tip Listing 2 illustrates two capabilities that are not widely known. A named range can define a function. Also, the third argument acts as the base address for cells referenced in a relative way.

Select the range containing the headers and the data and then use **Insert > Names > Create** to open the Create Names dialog (see Figure 2), which allows you to simultaneously create multiple named ranges based on the top row, bottom row, right column or left column. If you choose to create ranges based on the top row, one named range is created for each column header—the header is not included in the named range. Although the header is not included in the range, the text in the header is used to name the range.



Top row [✓]
Left column [✓]
Bottom row
Right column

Figure 2. Define a named range.

The macro in Listing 3 creates three named ranges based on the top row of a named range.

Listing 3. Create many named ranges.

```
Sub AddManyNamedRanges()
  Dim oSheet    'Sheet that contains the named range.
  Dim oAddress  'Range address.
  Dim oRanges   'The NamedRanges property.
  Dim oRange    'Single cell range.

  oRanges = ThisComponent.NamedRanges
  oSheet = ThisComponent.getSheets().getByIndex(0)
  oRange = oSheet.getCellRangeByName("A1:C20")
  oAddress = oRange.getRangeAddress()
  oRanges.addNewFromTitles(oAddress, _
                   com.sun.star.sheet.Border.TOP)
End Sub
```

The constants in Table 3 determine the location of the headers when multiple ranges are created using the method addNewFromTitles().

Table 3. com.sun.star.sheet.Border constants.

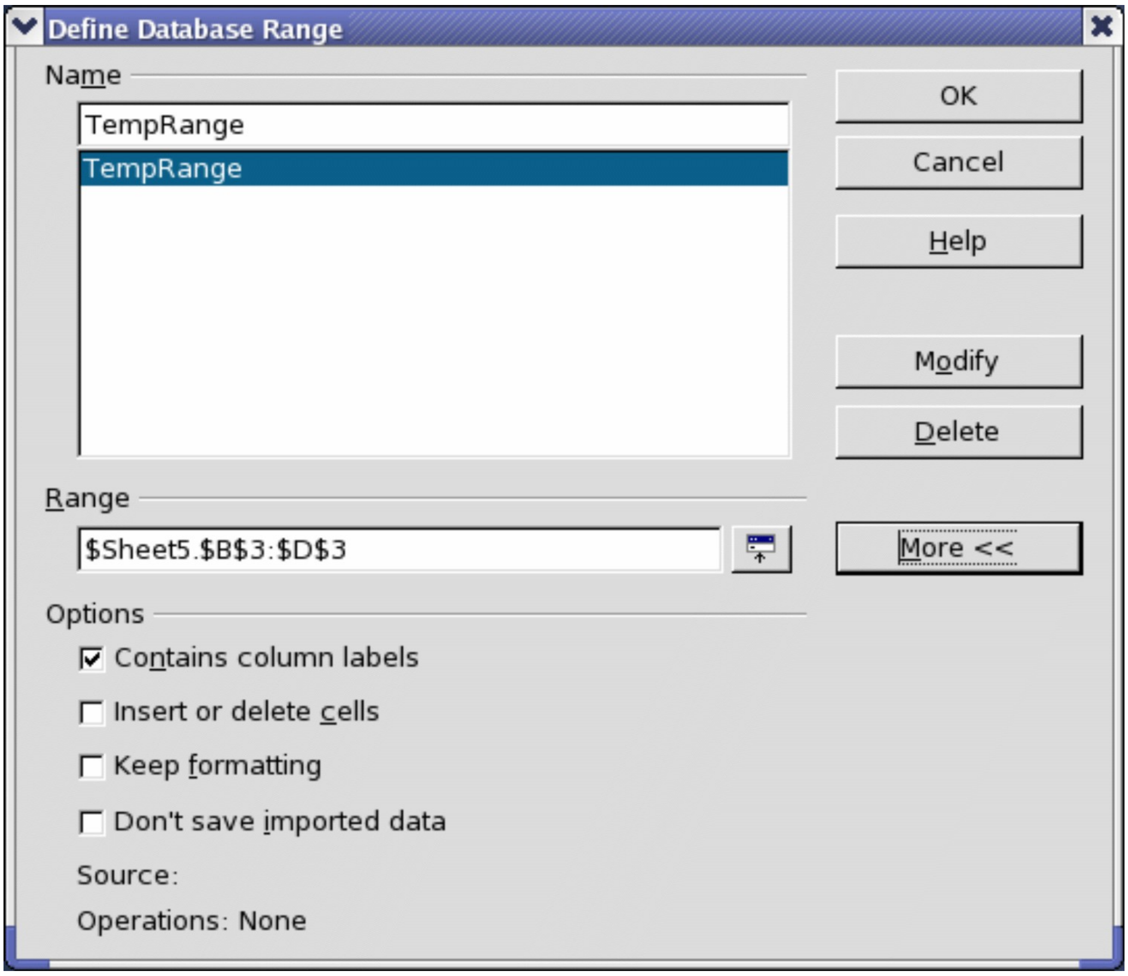
Value	Name	Description
0	TOP	Select the top border.
1	BOTTOM	Select the bottom border.
2	RIGHT	Select the right border.
3	LEFT	Select the left border.

Caution It is possible to create multiple named ranges with the same name. Creating multiple ranges with a single command increases the likelihood that multiple ranges will be created with the same name—avoid this if possible.

Database range

Although a database range can be used as a regular named range, a database range also defines a range of cells in a spreadsheet to be used as a database. Each row in a range corresponds to a record and each cell corresponds to a field. You can sort, group, search, and perform calculations on the range as if it were a database.

A database range provides behavior that is useful when performing database related activities. For example, you can mark the first row as headings. To create, modify, or delete a database range, use **Data > Define Range** to open the Define Data Range dialog (see Figure 3). When you first define a range, the Modify button shown in the example is labeled New.



TempRange

TempRange

$Sheet5.$B$3:$D$3

- [x] Contains column labels
- [ ] Insert or delete cells
- [ ] Keep formatting
- [ ] Don't save imported data

Source:

Operations: None

Figure 3. Define a database range.

In a macro, a database range is accessed, created, and deleted from the DatabaseRanges property. The macro in Listing 4 creates a database range named *MyName* and sets the range to be used as an auto filter.

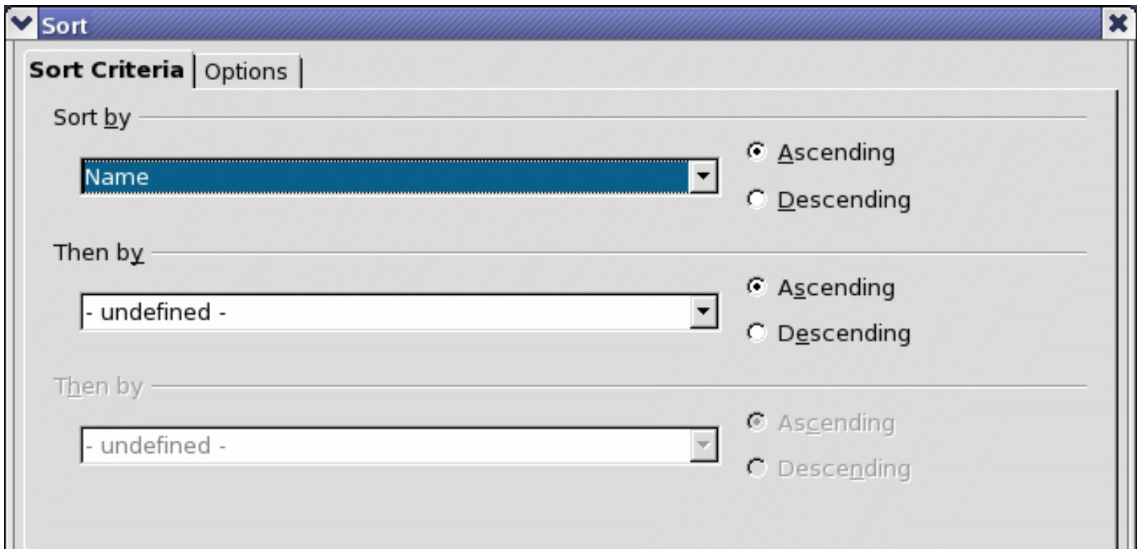
Listing 4. Create a database range and an auto filter.

```
Sub AddNewDatabaseRange()
  Dim oRange 'DatabaseRange object.
  Dim oAddr  'Cell address range for the database range.
  Dim oSheet 'First sheet, which will contain the range.
  Dim oDoc   'Reference ThisComponent with a shorter name.

  oDoc = ThisComponent
  If NOT oDoc.DatabaseRanges.hasByName("MyName") Then
    oSheet = ThisComponent.getSheets().getByIndex(0)
    oRange = oSheet.getCellRangeByName("A1:F10")
    oAddr = oRange.getRangeAddress()
    oDoc.DatabaseRanges.addNewByName("MyName", oAddr)
  End If
  oRange = oDoc.DatabaseRanges.getByName("MyName")
  oRange.AutoFilter = True
End Sub
```

Sorting

The sorting mechanism in a Calc document rearranges the data in the sheet. The first step in sorting data is to select the data that you want to sort. To sort the data in Table 1, select the cells from A1 to G16—if you include the column headers, indicate this in the sort dialog (see Figure 5). Use **Data > Sort** to open the Sort dialog (see Figure 4). You can sort by up to three columns or rows at a time.



Sort by: Name — [✓ Ascending] Descending

Then by: - undefined - — [✓ Ascending] Descending

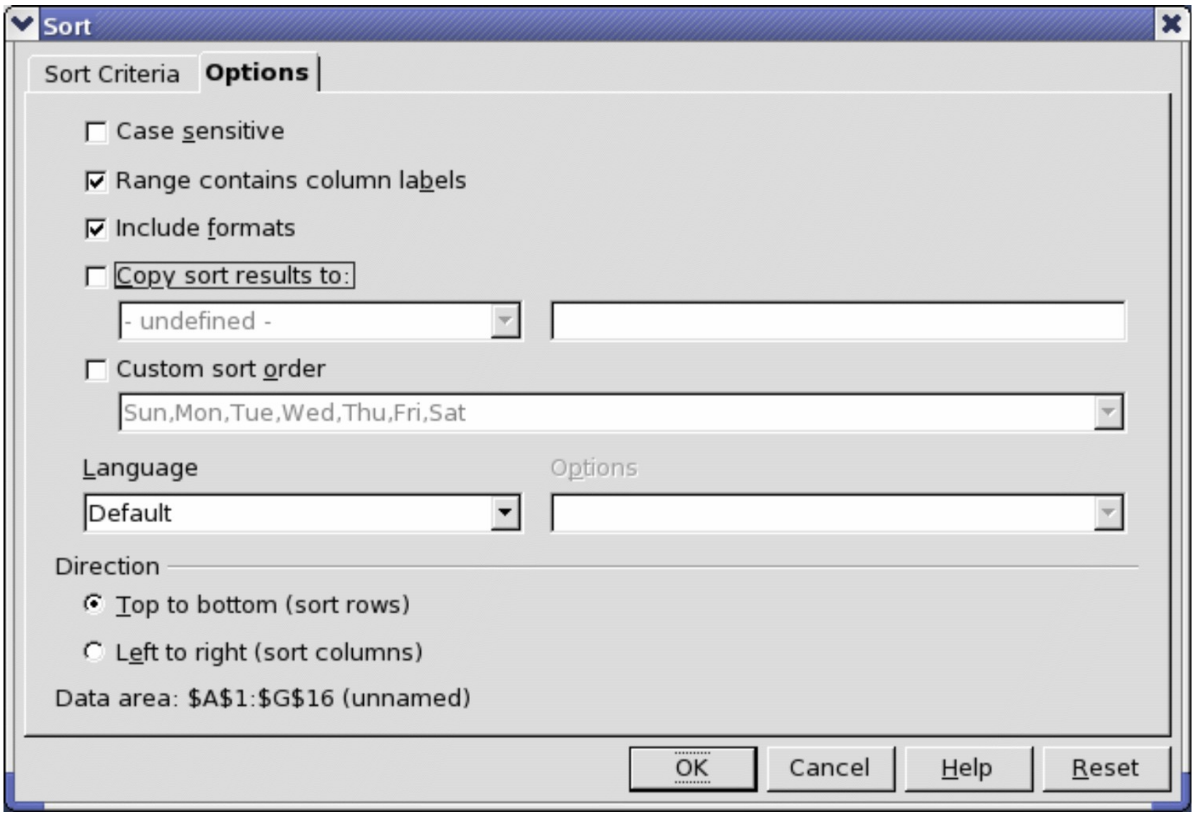
Then by: - undefined - — [✓ Ascending] Descending

Figure 4. Sort by the Name column.

Click on the Options tab (see Figure 5) to set the sort options. Check the **Range contains column labels** checkbox to prevent column headers from being sorted with the rest of the data. The Sort by list box in Figure 4 displays the columns using the column headers if the **Range contains column labels** checkbox in Figure 5 is checked. If the **Range contains column labels** checkbox is not checked, however, then the columns are identified by their column name; Column A, for example.

Normally, sorting the data causes the existing data to be replaced by the newly sorted data. The **Copy sort results to** checkbox, however, causes the selected data to be left unchanged and a copy of the sorted data is copied to the specified location. You can either directly enter a target address (Sheet3.A1, for example) or select a predefined range.

Check the **Custom sort order** checkbox to sort based on a predefined list of values. To set your own predefined lists, use **Tools > Options > OpenOffice.org Calc > Sort Lists** and then enter your own sort lists. Predefined sort lists are useful for sorting lists of data that should not be sorted alphabetically or numerically. For example, sorting days based on their name.



[ ] Case sensitive
[✓] Range contains column labels
[✓] Include formats
[ ] Copy sort results to:
- undefined -
[ ] Custom sort order
Sun,Mon,Tue,Wed,Thu,Fri,Sat
Language: Default
Options
Direction
(●) Top to bottom (sort rows)
( ) Left to right (sort columns)
Data area: $A$1:$G$16 (unnamed)

Figure 5. Set sort options.

Caution ⚠

When a cell is moved during a sort operation, external references to that cell are not updated. If a cell that contains a relative reference to another cell is moved, the reference is relative to the new position when sorting is finished. Know the behavior of references during sorting and do not be alarmed; this is almost always what you want—because the reference is to the right or left in the same row. Also, we have not found a spreadsheet program that exhibits a different behavior for references while sorting.

Filters

Use filters to limit the visible rows in a spreadsheet. Generic filters, common to all sorts of data manipulations, are automatically provided by the auto filter capability. You can also define your own filters.

Caution ⚠

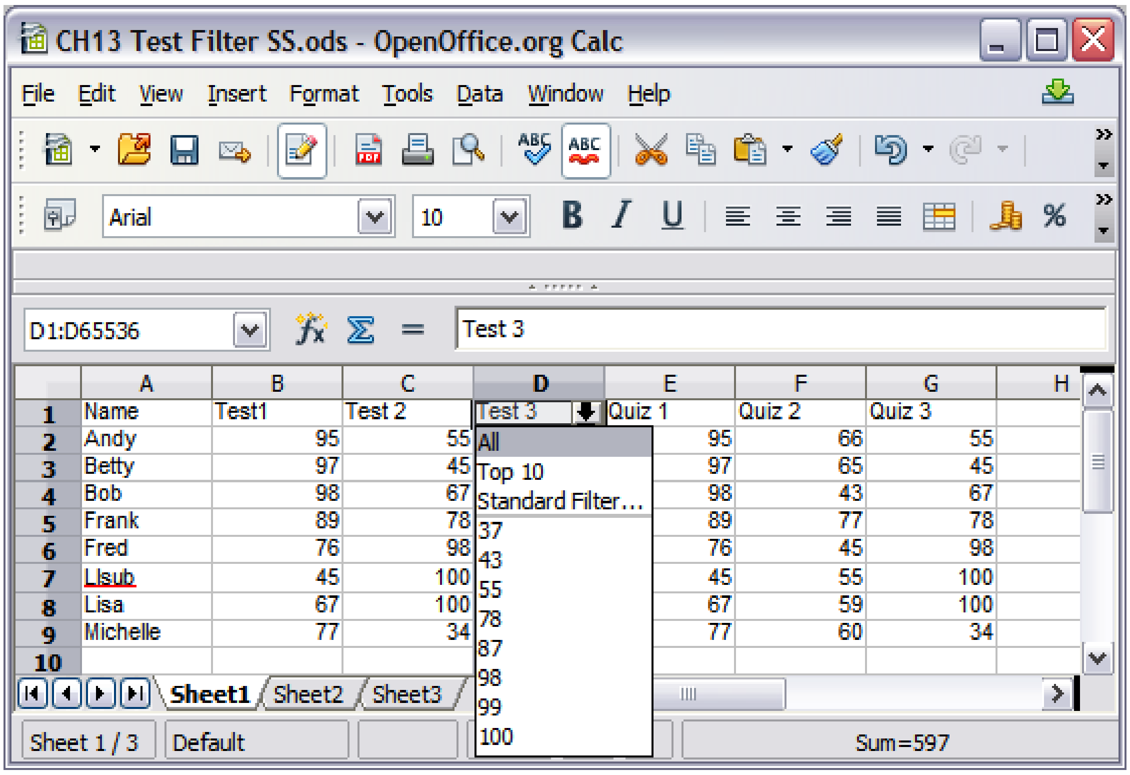
After applying a filter, some rows are visible and some rows are not. If you select multiple rows in one operation, you will also select the invisible rows contained between the selected visible rows. Operations, such as delete, act on all of the selected rows. To avoid this problem, you must individually select each of the filtered rows using the control key.

Auto filters

Use auto filters to quickly create easily accessible filters found to be commonly used in many different types of applications. After creating an auto filter for a specific column, a combo box is added to the column. The combo box provides quick access to each of the auto filter types.

- The All auto filter causes all rows to be visible.
- The Standard auto filter opens the Standard Filter dialog and is the same as the standard filter.
- The Top 10 auto filter displays the ten rows with the largest value. If the value 70 is in the top ten values, then all rows containing the value 70 in the filtered column are displayed. In other words, more than ten rows may be displayed.
- An auto filter entry is created for each unique entry in the column.

To create an auto filter, first select the columns to filter. For example, using the data in Table 1, select data in columns B and C. If you do not select the title rows, Calc asks if the title row or the current row should be used. Although you can place the auto filter in any row, only the rows below the auto filter are filtered. Use **Data > Filter > AutoFilter** to insert the auto filter combo box in the appropriate cell. Finally, use the drop-down arrow to choose an appropriate auto filter (see Figure 6).



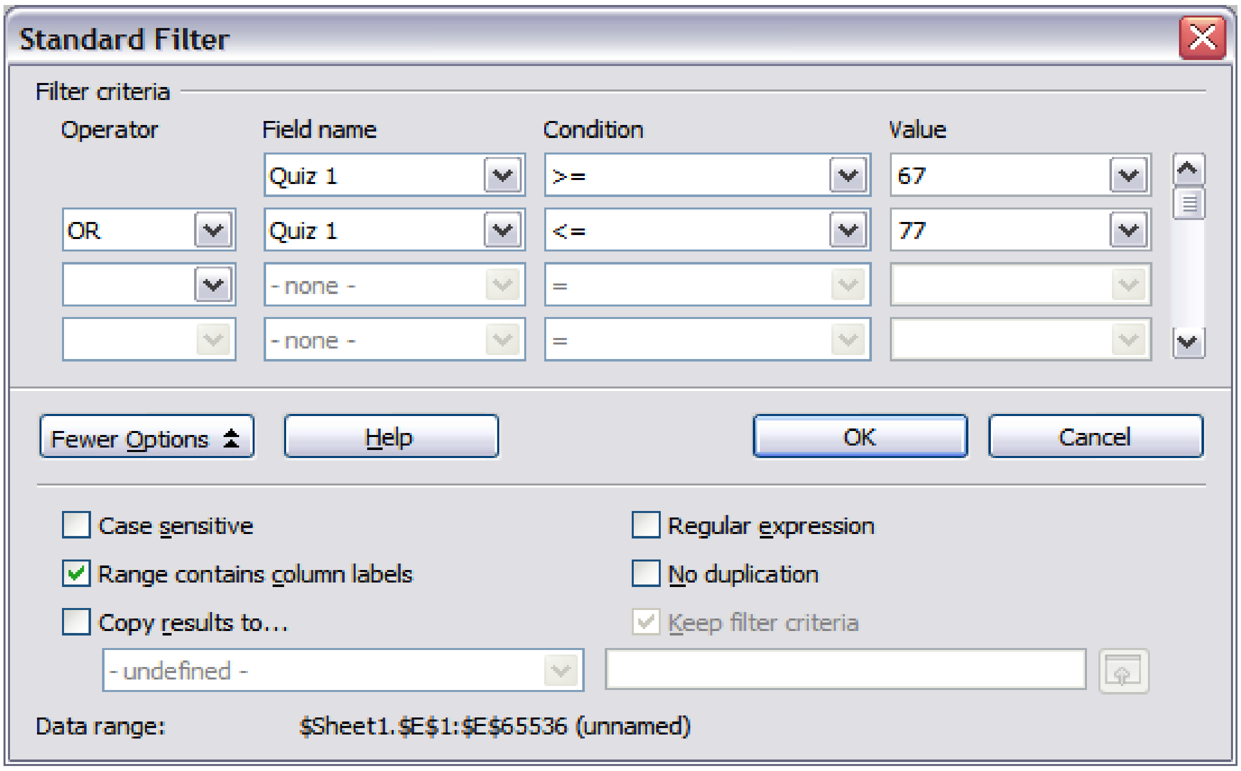
	A	B	C	D	E	F	G	H
1	Name	Test1	Test 2	Test 3	Quiz 1	Quiz 2	Quiz 3	
2	Andy	95	55		95	66	55	
3	Betty	97	45		97	65	45	
4	Bob	98	67		98	43	67	
5	Frank	89	78		89	77	78	
6	Fred	76	98		76	45	98	
7	Llsub	45	100		45	55	100	
8	Lisa	67	100		67	59	100	
9	Michelle	77	34		77	60	34	
10								

Figure 6: Use an auto filter with column C

Remove an auto filter by repeating the steps to create the auto filter—in other words, the menu option acts as a toggle to turn the auto filter on and off. When an auto filter is removed, the combo box is removed from the cell. The macro in Listing 4 demonstrates creating an auto filter for a range.

Standard filters

Use **Data > Filter > Standard Filter** to open Standard Filter dialog (see Figure 7) and limit the view based on 1 to 3 filter conditions. Use **Data > Filter > Remove Filter** to turn off the filter.


Standard Filter

Filter criteria

Operator	Field name	Condition	Value
	Quiz 1	>=	67
OR	Quiz 1	<=	77
	- none -	=	
	- none -	=	

Fewer Options · Help · OK · Cancel

- [ ] Case sensitive
- [✓] Range contains column labels
- [ ] Copy results to...
- [ ] Regular expression
- [ ] No duplication
- [✓] Keep filter criteria

- undefined -

Data range: $Sheet1.$E$1:$E$65536 (unnamed)

Figure 7: Use the standard filter

The macro in Listing 5 creates a simple filter for the first sheet.

Listing 5. Create a simple sheet filter.

```
Sub SimpleSheetFilter()
  Dim oSheet       ' Sheet that will contain the filter.
  Dim oFilterDesc  ' Filter descriptor.
  Dim oFields(0) As New com.sun.star.sheet.TableFilterField

  oSheet = ThisComponent.getSheets().getByIndex(0)

  REM If argument is True, creates an empty filter
  REM descriptor. If argument is False, create a
  REM descriptor with the previous settings.
  oFilterDesc = oSheet.createFilterDescriptor(True)

  With oFields(0)
    REM I could use the Connection property to indicate
    REM how to connect to the previous field. This is
    REM the first field so this is not required.
    '.Connection = com.sun.star.sheet.FilterConnection.AND
    '.Connection = com.sun.star.sheet.FilterConnection.OR

    REM The Field property is the zero based column
    REM number to filter. If you have the cell, you
    REM can use .Field = oCell.CellAddress.Column.
    .Field = 5
```

```
      REM Compare using a numeric or a string?
      .IsNumeric = True

      REM The NumericValue property is used
      REM because .IsNumeric = True from above.
      .NumericValue = 80

      REM If IsNumeric was False, then the
      REM StringValue property would be used.
      REM .StringValue = "what ever"

      REM Valid operators include EMPTY, NOT_EMPTY, EQUAL,
      REM NOT_EQUAL, GREATER, GREATER_EQUAL, LESS,
      REM LESS_EQUAL, TOP_VALUES, TOP_PERCENT,
      REM BOTTOM_VALUES, and BOTTOM_PERCENT
      .Operator = com.sun.star.sheet.FilterOperator.GREATER_EQUAL
    End With

    REM The filter descriptor supports the following
    REM properties: IsCaseSensitive, SkipDuplicates,
    REM UseRegularExpressions,
    REM SaveOutputPosition, Orientation, ContainsHeader,
    REM CopyOutputData, OutputPosition, and MaxFieldCount.
    oFilterDesc.setFilterFields(oFields())
    oFilterDesc.ContainsHeader = True
    oSheet.filter(oFilterDesc)
  End Sub
```

When a filter is applied to a sheet, it replaces any existing filter for the sheet. Setting an empty filter in a sheet will therefore remove all filters for that sheet (see Listing 6).

Listing 6. Remove the current sheet filter.

```
  Sub RemoveSheetFilter()
    Dim oSheet          ' Sheet to filter.
    Dim oFilterDesc     ' Filter descriptor.

    oSheet = ThisComponent.getSheets().getByIndex(0)
    oFilterDesc = oSheet.createFilterDescriptor(True)
    oSheet.filter(oFilterDesc)
  End Sub
```

Listing 7 demonstrates a more advanced filter that filters two columns and uses regular expressions. I noticed some unexpected behavior while working with Listing 7. Although you can create a filter descriptor using any sheet cell range, the filter applies to the entire sheet.

Listing 7. A simple sheet filter using two columns.

```
Sub SimpleSheetFilter_2()
  Dim oSheet          ' Sheet to filter.
  Dim oRange          ' Range to be filtered.
  Dim oFilterDesc     ' Filter descriptor.
  Dim oFields(1) As New com.sun.star.sheet.TableFilterField

  oSheet = ThisComponent.getSheets().getByIndex(0)
  oRange = oSheet.getCellRangeByName("E12:G19")

  REM If argument is True, creates an
  REM empty filter descriptor.
  oFilterDesc = oRange.createFilterDescriptor(True)

  REM Setup a field to view cells with content that
  REM start with the letter b.
  With oFields(0)
    .Field = 0                ' Filter column A.
    .IsNumeric = False        ' Use a string, not a number.
    .StringValue = "b.*"      ' Everything starting with b.
    .Operator = com.sun.star.sheet.FilterOperator.EQUAL
  End With
  REM Setup a field that requires both conditions and
  REM this new condition requires a value greater or
  REM equal to 70.
  With oFields(1)
    .Connection = com.sun.star.sheet.FilterConnection.AND
    .Field = 5                ' Filter column F.
    .IsNumeric = True         ' Use a number
    .NumericValue = 70        ' Values greater than 70
    .Operator = com.sun.star.sheet.FilterOperator.GREATER_EQUAL
  End With

  oFilterDesc.setFilterFields(oFields())
  oFilterDesc.ContainsHeader = False
  oFilterDesc.UseRegularExpressions = True
  oSheet.filter(oFilterDesc)
End Sub
```

Advanced filters

An advanced filter supports up to eight filter conditions, as opposed to the three supported by the simple filter. The criteria for an advanced filter is stored in a sheet. The first step in creating an advanced filter is entering the filter criteria into the spreadsheet.

1) Select an empty space in the Calc document. The empty space may reside in any sheet in any location in the Calc document.
2) Duplicate the column headings from the area to be filtered into the area that will contain the filter criteria.
3) Enter the filter criteria underneath the column headings (see Table 4). The criterion in each column of a row is connected with AND. The criteria from each row are connected with OR.

Table 4. Example advanced filter criteria

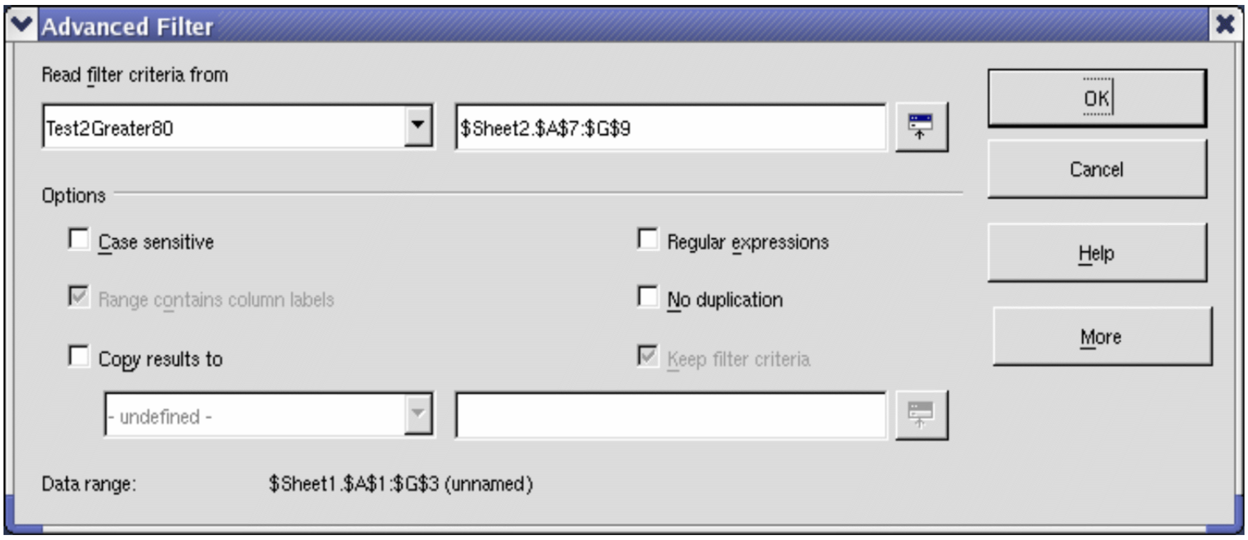
Name	Test 1	Test 2	Quiz 1	Quiz 2	Average	Grade
="Andy"		>80				
					<80	

Tip	Define named ranges to reference your advanced filter criteria and any destination ranges for filtered data (see Figure 1). Each appropriately configured named range is available in drop down list boxes in the Advanced Filter dialog (see Figure 8).

After creating one or more sets of filter criteria, apply an advanced filter as follows:

1) Select the sheet ranges that contain the data to filter.
2) Use **Data > Filter > Advanced Filter** to open the Advanced Filter dialog (see Figure 8).
3) Select the range containing the filter criteria and any other relevant options.
4) Click **OK**.

Applying an advanced filter using a macro is simple (see Listing 8). The cell range containing the filter criteria is used to create a filter descriptor, which is then used to filter the range containing the data.


Advanced Filter

Read filter criteria from: Test2Greater80 — $Sheet2.$A$7:$G$9

Options

- [ ] Case sensitive
- [✓] Range contains column labels
- [ ] Copy results to: - undefined -
- [ ] Regular expressions
- [ ] No duplication
- [✓] Keep filter criteria

Data range: $Sheet1.$A$1:$G$3 (unnamed)

Figure 8. Apply an advanced filter using a previously defined named range.

Listing 8. Use an advanced filter.

```
Sub UseAnAdvancedFilter()
  Dim oSheet     'A sheet from the Calc document.
  Dim oRanges    'The NamedRanges property.
  Dim oCritRange 'Range that contains the filter criteria.
  Dim oDataRange 'Range that contains the data to filter.
  Dim oFiltDesc  'Filter descriptor.

  REM Range that contains the filter criteria
  oSheet = ThisComponent.getSheets().getByIndex(1)
  oCritRange = oSheet.getCellRangeByName("A1:G3")

  REM You can also obtain the range containing the
  REM filter criteria from a named range.
  REM oRanges = ThisComponent.NamedRanges
  REM oRange = oRanges.getByName("AverageLess80")
  REM oCritRange = oRange.getReferredCells()

  REM The data that I want to filter
  oSheet = ThisComponent.getSheets().getByIndex(0)
  oDataRange = oSheet.getCellRangeByName("A1:G16")

  oFiltDesc =
oCritRange.createFilterDescriptorByObject(oDataRange)
  oDataRange.filter(oFiltDesc)
End Sub
```

Change properties on the filter descriptor to change the behavior of the filter (see Table 5).

The filter created in Listing 8 filters the data in place. Modify the OutputPosition property to specify a different output position (see Listing 9). The filter descriptor must be modified before the filter is applied.

Table 5. Advanced filter properties.

Property	***Comment***
ContainsHeader	Boolean (true or false) that specifies if the first row (or column) contains headers which should not be filtered.
CopyOutputData	Boolean that specifies if the filtered data should be copied to another position in the document.
IsCaseSensitive	Boolean that specifies if the case of letters is important when comparing entries.
Orientation	Specifies if columns (com.sun.star.table.TableOrientation.COLUMNS) or rows (com.sun.star.table.TableOrientation.ROWS) are filtered.
OutputPosition	If if CopyOutputData is True , specifies the position where filtered data are to be copied.
SaveOutputPosition	Boolean that specifies if the OutputPosition position is saved for future calls.
SkipDuplicates	Boolean that specifies if duplicate entries are left out of the result.
UseRegularExpressions	Boolean that specifies if the filter strings are interpreted as regular expressions.

Listing 9. Copy filtered results to a different location.

```
REM Copy the output data rather than filter in place.
oFiltDesc.CopyOutputData = True

REM Create a CellAddress and set it for Sheet3,
REM Column B, Row 4 (remember, start counting with 0)
Dim x As New com.sun.star.table.CellAddress
x.Sheet = 2
x.Column = 1
x.Row = 3
oFiltDesc.OutputPosition = x
```

(Advanced material.) The OutputPosition property returns a copy of a struct. Because a copy is returned, it is not possible to set the individual values directly. For example, oFiltDesc.OutputPosition.Row

= 2 does not work (because you set the Row on the copy to 2, but do not change the original).

Manipulating filtered data

Filtered data copied to a new location may be selected, modified, and deleted at will. Data that is not copied, however, requires special attention because rows that do not match the filter criteria are simply hidden. OpenOffice.org behaves differently depending on how the cells became hidden and what operation is done.

Cells may be hidden using an outline, data filter, or the hide command. When data is moved by dragging or using cut and paste, all of the cells are moved—including the hidden cells. When copying data, however, filtered data includes only the visible cells and data hidden using an outline or the hide command copies all of the data.

Calc functions similar to database functions

Although every Calc function can be used for database manipulation, the functions in Table 6 are more commonly used as such. Some functions' names differ only by the letter appended at the end; AVERAGE and AVERAGEA, for example. Functions that do not end with the letter A operate only on numeric values and cells that contain text or are empty are ignored. The corresponding function whose name ends with the letter A, treats text values as a number with the value of zero; blank cells are still ignored.

Table 6. Functions frequently used as database functions.

Function	***Description***
AVERAGE	Return the average. Ignore empty cells and cells that contain text.
AVERAGEA	Return the average. The value of text is 0 and empty cells are ignored.
COUNT	Count the number of numeric entries; text entries are ignored.
COUNTA	Count the number of non-empty entries.
COUNTBLANK	Return the number of empty cells.
COUNTIF	Return the number of cells that meet the search criteria.

Function	Description
HLOOKUP	Search for a specific value across the columns in the first row of an array. Return the value from a different row in the same column.
INDEX	Return the content of a cell, specified by row and column number or an optional range name.
INDIRECT	Return the reference specified by a text string.
LOOKUP	Return the contents of a cell either from a one-row or one-column range or from an array.
MATCH	Search an array and return the relative position of the found item.
MAX	Return the maximum numeric value in a list of arguments.
MAXA	Return the maximum numeric value in a list of arguments. The value of text is 0.
MIN	Return the minimum numeric value in a list of arguments.
MINA	Return the minimum numeric value in a list of arguments. The value of text is 0.
MEDIAN	Return the median of a set of numbers.
MODE	Return the most common value in a data set. If there are several values with the same frequency, it returns the smallest value. An error occurs when a value doesn’t appear twice.
OFFSET	Return the value of a cell offset by a certain number of rows and columns from a given reference point.
PRODUCT	Return the product of the cells.
STDEV	Estimate the standard deviation based on a sample.
STDEVA	Estimate the standard deviation based on a sample. The value of text is 0.
STDEVP	Calculate the standard deviation based on the entire population.
STDEVPA	Calculate the standard deviation based on the entire population.
SUBTOTAL	Calculate a specified function based on a subset created using AutoFilters.
SUM	Return the sum of the cells.

Function	Description
SUMIF	Calculate the sum for the cells that meet the search criteria.
VAR	Estimate the variance based on a sample.
VARA	Estimate the variance based on a sample. The value of text is 0.
VARP	Estimate the variance based on the entire population.
VARPA	Estimate the variance based on the entire population. The value of a text is 0.
VLOOKUP	Search for a specific value across the rows in the first column of an array. Returns the value from a different column in the same row.

Most of the functions in Table 6 require no explanation, either because they are well understood (SUM, for example) or because if you need to use them then you know what they are (STDEV, for example). Unfortunately, some of the more useful functions are infrequently used because they are not well understood.

Count and sum cells that match conditions: COUNTIF and SUMIF

The COUNTIF and SUMIF functions calculate their values based on search criteria. The search criteria can be a number, expression, text string, or even a regular expression. The search criteria can be contained in a referenced cell or it can be included directly in the function call.

The COUNTIF function counts the number of cells in a range that match specified criteria. The first argument to COUNTIF specifies the range to search and second argument is the search criteria. Table 7 illustrates different search criteria using the COUNTIF function referencing the data shown in Table 1.

The first two arguments for SUMIF serve the same purpose as the arguments for COUNTIF; the range that contains the cells to search and the search criteria. The third and final argument for SUMIF specifies the range to sum. For each cell in the search range that matches the search criteria, the corresponding cell in the sum range is added into the sum.

Table 7. Examples of search criteria for the COUNTIF and SUMIF functions.

Criteria Type	Function	Result	Description
Number	=COUNTIF(B1:C16; 95)	3	Finds numeric values of 95.
Text	=COUNTIF(B1:C16; "95")	3	Finds numeric or text values of 95.
Expression	=COUNTIF(B1:C16; ">95")	6	Finds numeric values greater than 95.
Expression	=COUNTIF(B1:C16; 2*45+5)	3	Finds only numeric values of 95.
Regular expression	=COUNTIF(B1:C16; "9.*")	12	Finds numbers or text that start with 9.
Reference a cell	=COUNTIF(B1:C16; B3)	3	Finds a number or number and text depending on the data type in cell B3.
Regular expression	=SUMIF(A1:A16; "B.*"; B1:B16)	227	Sum Column B for names in Col. A starting with the letter B.

Ignore filtered cells using SUBTOTAL

The SUBTOTAL function applies a function (see Table 8) to a range of data, but it ignores cells hidden by a filter and cells that already contain a SUBTOTAL. For example, `=SUBTOTAL(2, "B2:B16")` counts the number of cells in B2:B16 that are not hidden by a filter.

Table 8. Function index for the SUBTOTAL function.

Function index	Function
1	AVERAGE
2	COUNT
3	COUNTA
4	MAX
5	MIN
6	PRODUCT
7	STDEV

Function index	Function
8	STDEVP
9	SUM
10	VAR
11	VARP

Tip Do not forget that the SUBTOTAL function ignores cells that use the SUBTOTAL function. Say you have a spreadsheet that tracks investments. The retirement investments are grouped together with a subtotal. The same is true of regular investments. You can use a single subtotal that includes the entire range without worrying about the subtotal cells.

Using formulas to find data

Calc offers numerous methods to find data in a sheet. For example, **Edit > Find & Replace** moves the display cursor based on simple and advanced searching. Use **Data > Filter** to limit what is displayed rather than simply moving the cursor. Calc also offers lookup functions used in formulas, for example a formula to look up a student's grade based on their test score.

Search a block of data using VLOOKUP

Use VLOOKUP to search the first column (columns are vertical) of a block of data and return the value from another column in the same row. For example, search the first column for the name "Fred" and then return the value in the cell two columns to the right. VLOOKUP supports two forms:

```
VLOOKUP(search_value; search_range; return_column_index)
VLOOKUP(search_value; search_range; return_column_index;
sort_order)
```

The first argument, `search_value`, identifies the value to find. The search value can be text, a number, or a regular expression. For example, **Fred** searches for the text Fred, **4** searches for the number 4, and **F.*** is the regular expression for finding something that starts with the letter F.

The second argument, `search_range`, identifies the cells to search; only the first column is searched. For example, **B3:G10** searches the same sheet containing the VLOOKUP formula and **Sheet2.B3:G10** searches the range B3:G10 on the sheet named Sheet2.

The `return_column_index` identifies the column to return; a value of 1 returns the first column in the range. The statement `=VLOOKUP("Bob"; A1:G9; 1)` finds the first row in A1:G9 containing the text **Bob,** and returns the value in the first column. The first column is the searched column, so the text **Bob** is returned. If the column index is 2, then the value in the cell to the right of Bob is returned: column B.

The final column, `sort_order`, is optional. The default value for `sort_order` is 1, which specifies that the first column is sorted in ascending order; a value of 0 specifies that the data is not sorted. A non-sorted list is searched by sequentially checking every cell in the first column for an exact match. If an exact match is not found, the text **#N/A** is returned.

A more efficient search routine is used if the data is sorted in ascending order. If one exact match exists, the returned value is the same as for a non-sorted list; but it is faster. If a match does not exist, the largest value in the column that is less than or equal to the search value is returned. For example, searching for 7 in (3, 5, 10) returns 5 because 7 is between 5 and 10. Searching for 27 returns 10, and searching for 2 returns #N/A because there is no match and no value less than 2.

Use VLOOKUP when:

- The data is arranged in rows and you want to return data from the same row. For example, student names with test and quiz scores to the right of the student's name.
- Searching the first column of a range of data.

Search a block of data using HLOOKUP

Use HLOOKUP to search the first row (rows are horizontal) of a block of data and return the value from a row in the same column. HLOOKUP supports the same form and arguments as VLOOKUP:

```
HLOOKUP(search_value; search_range; return_row_index)
HLOOKUP(search_value; search_range; return_row_index; sort_order)
```

Use HLOOKUP when:

- The data is arranged in columns and you want to return data from the same column. For example, student names with test and quiz scores underneath the student's name.
- Searching the first row of a range of data.

Search a row or column using LOOKUP

LOOKUP is similar to HLOOKUP and VLOOKUP. The search range for the LOOKUP function is a single sorted row or column. LOOKUP has two forms:

```
LOOKUP(search_value; search_range)
LOOKUP(search_value; search_range; return_range)
```

The search value is the same as HLOOKUP and VLOOKUP. The search range, however, must be a single row or a single column; for example, A7:A12 (values in column A) or C5:Q5 (values in row 5). If the return_range is omitted, the matched value is returned. Using LOOKUP without a return range is the same as using HLOOKUP or VLOOKUP with a column index of 1.

The return range must be a single row or column containing the same number of elements as the search range. If the search value is found in the fourth cell in the search range, then the value in the fourth cell in the return range is returned. The return range can have a different orientation than the search range. In other words, the search range can be a row and the return range may be a column.

Use LOOKUP when:

- The search data is sorted in ascending order.
- The search data is not stored in the same row, column, or orientation as the return data.

Use MATCH to find the index of a value in a range

Use MATCH to search a single row or column and return the position that matches the search value. Use MATCH to find the index of a value in a range. The supported forms for MATCH are as follows:

```
=MATCH(search_value; search_range)
=MATCH(search_value; search_range; search_type)
```

The search value and search range are the same as for LOOKUP. The final argument, search type, controls how the search is performed. A search type of 1, sorted in ascending order, is the default. A search type of -1 indicates that the list is sorted in descending order. A search type of 0 indicates that the list is not sorted. Regular expressions can only be used on an unsorted list.

Use MATCH when:

- You need an index into the range rather than the value.

- The search data is in descending order and the data is large enough that the data must be searched assuming that it is sorted; because it is faster to sort a sorted list.

Examples

Consider the data in Table 1. Each student's information is stored in a single row. Write a formula to return the average grade for Fred. The problem can be restated as Search column A in the range A1:G16 for Fred and return the value in column F (column F is the sixth column). The obvious solution is `=VLOOKUP("Fred"; A2:G16; 6)`. Equally obvious is `=LOOKUP("Fred"; A2:A16; F2:F16)`.

It is common for the first row in a range to contain column headers. All of the search functions check the first row to see if there is a match and then ignore it if it does not contain a match, in case the first row is a header.

What if the column heading **Average** is known, but not the column containing the average? Find the column containing Average rather than hard coding the value 6. A slight modification using MATCH to find the column yields `=VLOOKUP("Fred"; A2:G16; MATCH("Average"; A1:G1; 0))`; notice that the heading is not sorted. As an exercise, use HLOOKUP to find Average and then MATCH to find the row containing Fred.

As a final example, write a formula to assign grades based on a student's average score. Assume that a score less than 51 is an F, less than 61 is an E, less than 71 is a D, less than 81 is a C, less than 91 is a B, and 91 to 100 is an A. Assume that the values in Table 9 are in Sheet2.

Table 9. Associate scores to a grade.

	A	B
1	Score	Grade
2	0	F
3	51	E
4	61	D
5	71	C
6	81	B
7	91	A

The formula =`VLOOKUP(83; $Sheet2.$A$2:$B$7; 2)` is an obvious solution. Dollar signs are used so that the formula can be copied and pasted to a different location and it will still reference the same values in Table 9.

ADDRESS returns a string with a cell's address

Use ADDRESS to return a text representation of a cell address based on the row, column, and sheet; ADDRESS is frequently used with MATCH. The supported forms for ADDRESS are as follows:

```
ADDRESS(row; column)
ADDRESS(row; column; abs)
ADDRESS(row; column; abs; sheet)
```

The row and column are integer values where ADDRESS(1; 1) returns **$A$1**. The abs argument specifies which portion is considered absolute and which portion is considered relative (see Table 10); an absolute address is specified using the $ character. The sheet is included as part of the address only if the sheet argument is used. The sheet argument is treated as a string. Using `ADDRESS(MATCH("Bob";A1:A5 ; 0); 2)` with the data in Table 9 returns **$B$2.**

Tip Calc supports numerous powerful functions that are not discussed here. For example, the ROW, COLUMN, ROWS, and COLUMNS statements are not discussed; a curious person would investigate these functions.

Table 10. Values supported by the abs argument to ADDRESS.

Value	Description
1	Use absolute addressing. This is the default value if the argument is missing or an invalid value is used. `ADDRESS(2; 5; 1)` returns $E$2.
2	Use an absolute row reference and a relative column reference. `ADDRESS(2; 5; 2; "Blah")` returns Blah.E$2.
3	Use a relative row reference and an absolute column reference. `ADDRESS(2; 5; 3)` returns $E2.
4	Use relative addressing. `ADDRESS(2; 5; 4)` returns E2.

INDIRECT converts a string to a cell or range

Use INDIRECT to convert a string representation of a cell or range address to a reference to the cell or range. Table 11 contains examples accessing data as shown in Table 9.

Table 11. Examples using INDIRECT.

Example	Comment
INDIRECT("A2")	Returns cell A2, which contains **Bob**.
INDIRECT(G1)	If Cell G1 contains the text A2, then this returns **Bob**.
SUM(INDIRECT("B1:B5"))	Returns the sum of the range B1:B5, which is **194**.
INDIRECT(ADDRESS(2; 1))	Returns the contents of cell $A$2, which is **Bob**.

OFFSET returns a cell or range offset from another

Use OFFSET to return a cell or range offset by a specified number of rows and columns from a given reference point. The first argument, specifies the reference point. The second and third arguments specify the number of rows and columns to move from the reference point; in other words, where the new range starts. The OFFSET function has the following syntax:

```
OFFSET(reference; rows; columns)
OFFSET(reference; rows; columns; height)
OFFSET(reference; rows; columns; height; width)
```

Tip If the width or height is included, the OFFSET function returns a range. If both the width and height are missing, a cell reference is returned.

If the height or width are missing, they default to 1. If the height is present, then a range reference is returned rather than a cell reference. Using values from Table 1, Listing 10 uses OFFSET to obtain the quiz scores for the student named Bob.

Listing 10. Complex example of OFFSET.

```
=SUM(OFFSET(INDIRECT(ADDRESS(MATCH("Bob";A1:A16; 0); 4)); 0; 0; 1; 2))
```

In its entirety, Listing 10 is complex and difficult to understand. Table 12 isolates each function in Listing 10, providing an easy to understand explanation of how the example works.

Table 12. Breakdown of Listing 10.

Function	Description
MATCH("Bob";A1:A16; 0)	Return 4 because Bob is the fourth entry in column A.
ADDRESS(4; 4)	Return $D$4.
INDIRECT("$D$4")	Convert $D$4 into a reference to the cell D4.
OFFSET($D$4; 0; 0; 1; 2)	Return the range D4:E4.
SUM(D4:E4)	Return the sum of Bob’s quiz scores.

Although Listing 10 works as intended, it breaks easily and unexpectedly. Consider, for example, what happens if the range is changed to A2:A16. MATCH returns an offset into the provided range, so `MATCH("Bob";A2:A16 ; 0)` returns 3 rather than 4. `ADDRESS(3; 4)` returns $D$3 rather than $D$4 and Betty’s quiz scores are returned instead of Bob’s. Listing 11 uses a slightly different method to obtain Bob’s quiz scores.

Listing 11. Better use of OFFSET.

```
=SUM(OFFSET(A1; MATCH("Bob"; A1:A16; 0)-1; 3; 1; 2))
```

Table 13 contains a description of each function used in Listing 11. To help convince yourself that Listing 11 is better than Listing 10, replace A1 with A2 in both Listing 11 and Table 13 and notice that you still obtain Bob’s quiz scores.

Table 13. Breakdown of Listing 11.

Function	Description
MATCH("Bob";A1:A16; 0)-1	Return 3 because Bob is the fourth entry in column A.
OFFSET(A1; 3; 3; 1; 2)	Return the range D4:E4.
SUM(D4:E4)	Return the sum of Bob’s quiz scores.

Tip The first argument to OFFSET can be a range so you can use a defined range name.

INDEX returns cells inside a specified range

INDEX returns the cells specified by a row and column number. The row and column number are relative to the upper left corner of the specified reference range. For example, using `=INDEX(B2:D3; 1; 1)` returns the cell B2. Table 14 lists shows the syntax for using the INDEX function.

Table 14. Syntax for INDEX.

Syntax	**Description**
INDEX(reference)	Return the entire range.
INDEX(reference; row)	Return the specified row in the range.
INDEX(reference; row; column)	Return the cell specified by row and column. A row and column of 1 returns the cell in the upper left corner of the range.
INDEX(reference; row; column; range)	A reference range can contain multiple ranges. The range argument specifies which range to use.

The INDEX function can return an entire range, a row, or a single column (see Table 14). The ability to index based on the start of the reference range provides some interesting uses. Using the values shown in Table 1, Listing 12 finds and returns Bob's quiz scores. Table 15 contains a listing of each function used in Listing 12.

Listing 12. Return Bob's quiz scores.

```
=SUM(OFFSET(INDEX(A2:G16; MATCH("Bob"; A2:A16; 0)); 0; 3; 1; 2))
```

Table 15. Breakdown of Listing 12.

Function	**Description**
MATCH("Bob";A2:A16; 0)	Return 3 because Bob is the third entry in column A2:A16.
INDEX(A2:A16; 3)	Return A4:G4—the row containing Bob's quiz scores.
OFFSET(A4:G4; 0; 3; 1; 2)	Return the range D4:E4.
SUM(D4:E4)	Return the sum of Bob's quiz scores.

Tip A simple range contains one contiguous rectangular region of cells. It is possible to define a multi-range that contains multiple simple ranges. If the reference consists of multiple ranges, you must enclose the reference or range name in parentheses.

If reference argument to the INDEX function is a multi-range, then the range argument specifies which simple range to use (see Table 16).

Table 16. Using INDEX with a multi-range.

Function	Returns
=INDEX(B2:G2; 1; 2)	93
=INDEX(B5:G5; 1; 2)	65
=INDEX((B2:G2;B5:G5); 1; 2)	93
=INDEX((B2:G2;B5:G5); 1; 2; 1)	93
=INDEX((B2:G2;B5:G5); 1; 2; 2)	65

Database-specific functions

Although every Calc function can be used for database manipulation, the functions in Table 17 are specifically designed for use as a database. The descriptions in Table 17 use the following terms interchangeably: row and record, cell and field, and database and all rows.

Table 17. Database functions in a Calc document.

Function	Description
DAVERAGE	Return the average of all fields that matches the search criteria.
DCOUNT	Count the number of records containing numeric data that match the search criteria.
DCOUNTA	Count the number of records containing text data that match the search criteria.
DGET	Return the contents of a field that matches the search criteria.
DMAX	Return the maximum content of a field that matches the search criteria.
DMIN	Return the minimum content of a field that matches the search criteria.
DPRODUCT	Return the product of the fields that matches the search criteria.
DSTDEV	Calculate the standard deviation using the fields that match the search criteria. The fields are treated as a sample.
DSTDEVP	Calculate the standard deviation using the fields that match the search criteria. The fields are treated as the entire population.
DSUM	Return the sum of all fields that matches the search criteria.

Function	Description
DVAR	Calculate the variance using the fields that match the search criteria. The fields are treated as a sample.
DVARP	Calculatesthe variance using the fields that match the search criteria. The fields are treated as the entire population.

The syntax for the database functions are identical.

```
DCOUNT(database; database field; search criteria)
```

The database argument is the cell range that defines the database. The cell range should contain the column labels (see Listing 13). The following examples, assume that the data from Table 1 is placed in Sheet 1 and the filter criteria in Table 4 is placed in Sheet 2.

Listing 13. The database argument includes the headers.

```
=DCOUNT(A1:G16; "Test 2"; Sheet2.A1:G3)
```

The database field specifies the column on which the function operates after the search criteria is applied and the data rows are selected. The database field can be specified using the column header name or as an integer. If the column is specified as an integer, 0 specifies the entire data range, 1 specifies the first column, 2 specifies the second column, and so on. Listing 14 calculates the average test score for the rows that match the search criteria.

Listing 14. “Test 2” is column 3.

```
=DAVERAGE(A1:G16; "Test 2"; Sheet2.A1:G3)
=DAVERAGE(A1:G16; 3; Sheet2.A1:G3)
```

The search criteria is the cell range containing search criteria. The search criteria is identical to the advanced filters; criteria in the same row is connected by AND and criteria in different rows is connected by OR.

Conclusion

A Calc document provides sufficient database functionality to satisfy the needs of most people. The infrequently used database functions, such as OFFSET and INDEX, are worth the time to learn and they can save yourself time in the long run.