Mr. Excel claims that 80% of the macros he develops for clients will make use of an advanced filtering technique.

Learn more about filtering

Filtered data displays only the rows that meet criteria that you specify and hides rows that you do not want displayed. After you filter data, you can copy, find, edit, format, chart, and print the subset of filtered data without rearranging or moving it.

You can also filter by more than one column. Filters are additive, which means that each additional filter is based on the current filter and further reduces the subset of data.

Note When you use the Find dialog box to search filtered data, only the data that is displayed is searched; data that is not displayed is not searched. To search all the data, clear all filters.

The three types of filters
Using AutoFilter, you can create three types of filters: by a list's values, by a format, or by criteria. Each of these filter types is mutually exclusive for each range of cells or column table. For example, you can filter by cell color or by a list of numbers, but not by both; you can filter by icon or by a custom filter, but not by both.

The work that auto filters can do can also be done with loops however auto filters are much faster.

The following sub uses a loop to find some text and then highlight it.

Sub Using a For Loop to Check for Occurrences of a Given Text
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub OldLoop( )
FinalRow = Cells(Rows, Count, 1).End(xlUp).Row
For i = 2 To FinalRow
  If Cells(i, 4) = "Ford" Then
    Cells(i, 1).Resize(1, 8).Interior.ColorIndex = 4
  End If
Next I
End Sub
(If you want to run this one you need to supply the appropriate data found back in the loops chapter.)

If deleting rows it should be done carefully from the bottom up. Notice the following loop is iterating its values in reverse order.

**Sub Using a For Loop to Delete Rows Associated with Specific Item**

// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

```vba
Sub OldLoopToDelete()
    FinalRow = Cells(Rows, Count, 1).End(xlUp).Row
    For i = FinalRow To 2 Step -1
        If Cells(i, 4) = "Ford" Then
            Rows(i).Delete
        End If
    Next i
End Sub
```

(A case study appears in the chapter at this point making the point again that auto filters are faster and more efficient than loops.)

**The Magic of AutoFilters**

The AutoFilter Method selects matching records in a single line of code.

**Example**

```
Range("A1").AutoFilter Field:=4, Criteria1:="Ford"
```

**Turn off AutoFilter 'Drop Downs'**

The following line by itself turns off the auto filter drop-downs and clears the filter.

**Example**

```
Range("A1").AutoFilter
```

ActiveSheet.ShowAllData can be used to keep the drop-downs on and to show all the data except the filtered data.

**Example**

```
ActiveSheet.ShowAllData
```
The AutoFilter Method Signature

Range.AutoFilter(Field, Criteria1, Operator, Criteria2, VisibleDropDown)

where all fields are optional and may include the following.

- Field is the Column number
- Criteria1 is search criteria
- Operator is the logical operator
  - An XLAutoFilterOperator
  - xlAnd, xlOr, xlFilterCellColor, etc.
- Criteria2 is a second condition that is compared with the first.

Dealing With Unwanted Rows, Using Offset( ) & Resize( )

When a Range is selected the header is also selected. An offset may be used to skip the header but this includes the blank line at the end.

// note Offset(1) is equivalent to (Offset 1, 0) or one row down

Sub AutoFilter & an Offset
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub DeleteFord()
'skips header, but also deletes blank line below
  Range("A1:A1").AutoFilter Field:= 4, Criteria1:= "Ford"
  Range("A1").CurrentRegion.Offset(1).EntireRow.Delete
  Range("A1").AutoFilter
End Sub

This form is OK for some things but not suitable for all scenarios. For instance, if filtered rows are being color encoded, the blank row at the end is also painted.

The Resize method can correct the situation by removing the added row at the end of a call on an Offset method.

When CurrentRegion is used with Range("A1") the whole dataset is being referenced.

Example
Range("A1").CurrentRegion
Sub ColorFord()
    DataHt = Range("A1").CurrentRegion.Rows.Count
    Range("A1").Autofilter Field:=4, Criteria1:= "Ford"

    With Range("A1").CurrentRegion.Offset(1).Resize(DataHt – 1)
        ' No need to use 'VisibleCellsOnly for formatting
        .Interior.ColorIndex = 4
        .Font.Bold = True
    End With
    ' Clear the AutoFilter and Remove drop downs

    Range("A1").Autofilter
End Sub

Excel 2007 Introduced New Filter Features

Excel 2007 introduced

- selecting multiple items
- filtering by color
- by icon
- by top 10
- virtual data filters

Excel 2010 Introduced

- new search box in the filter drop-downs

All these features can be coded equivalently in VBA.

Legacy Excel allowed for choosing only two items using logical AND and OR operators as is shown in the following example.

And OR Operator Code Excerpt
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Range("A1").Autofilter Field:= 4,
Criteria1:= "Ford ", _
Operator:= xlOr, _
Criteria2:= "General Motors"
The xlTopTenOperator

A bit confusing, the xlTopTenItems operator ended up being able to select for any of a numbers of items, like the top 5 or top 12, not just the top 10, implied by it's name. as is shown in the following example.

TopTen Macro
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub TopTenFilter( )
    ' Top 12 Revenue Records
End Sub

While Excel 2010 offered more options, the additions were 'shoe-horned' into the old working model where an operator is specified along with two items.

Choosing More Than Two Items With the xlFilterValues Operator

The new xlFilterValues operator can be used to use more than two items. This operator works by allowing an Array to enclose additional target items.

Filtering By Array With xlFilterValues

Range("A1").AutoFilter Field:=4, _
    Criteria1:= Array("General Motors", "Ford", "Chrysler", "Mazda"), _
    Operator:= xlFilterValues

Following is a sub that merges some of the formatting discussed earlier with this latest example.

Example

Sub ColorCars2()
DataHt = Range("A1").CurrentRegion.Rows.Count
Range("A1").AutoFilter Field:=4, _
Criteria1:=Array("General Motors", "Ford", "Chrysler", "Mazda"), _
Operator:=xlFilterValues
With Range("A1").CurrentRegion.Offset(1).Resize(DataHt - 1)
    ' No need to use 'VisibleCellsOnly for formatting
    .Interior.ColorIndex = 3
    .Font.Bold = True
End With
' Clear the AutoFilter and Remove drop downs
Range("A1").AutoFilter
End Sub
The Search Box & Alternate VBA Code

The Macro Recorder used with the Search Box ends up recording a hard coded list of items matching the search criteria which is not very dynamic. The Search Box effectively is searching for a text subset within a given text.

VBA code uses asterisks as wild cards to achieve the same end.

Example

Range("A1").AutoFilter Field:= 4 Criteria1:= "F*"  // would find Ford and Fiat

The following code shows the wild card notation in action.

Choosing Multiple Items By Wild Card
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub FilterSearchBox()
    ' Top Ten Revenue Records
    Worksheets("NewFilter").Select
    Range("A1:A1").AutoFilter Field:=4, _
    Criteria1:="**at**"
End Sub

Following are a set of Mr. Excels macros dealing with color filtering.

// Downloads have the 'Worksheets("NewFilter").Select' line in them

Filtering By Color

Filtering by color is facilitated by using the xlFilterFontColor operator and criteria specified as RGB or Red Green Blue values. Each of the following macros needs little more than the given title to understand what they are doing.

Macro That Filters By Color
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub FilterByFontColor()
    Range("A1").AutoFilter Field:=6, _
    Criteria1:=RGB(255, 0, 0), _
    Operator:=xlFilterFontColor
End Sub

The following one is a little odd to understand at first glance. It filters all records that have not explicitly been assigned a specific color.
Macro Finding Records With No Particular Font Color
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub FilterNoFontColor()
    Range("A1").AutoFilter Field:=6, _
    Operator:=xlFilterAutomaticFontColor
End Sub

In each of these examples, the operator tips off what aspect is being filtered.

Macro Finding Records With a Particular Fill Color
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub FilterByFillColor()
    Range("A1").AutoFilter Field:=6, _
    Criteria1:=RGB(255, 0, 0), _
    Operator:=xlFilterCellColor
End Sub

Finding a record without a fill color is achieved with the xlFilterNoFill operator in conjunction with not specifying any criteria.

Macro Finding Records Without a Fill Color
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub FilterNoFillColor()
    Range("A1").AutoFilter Field:=6, _
    Operator:=xlFilterNoFill
End Sub

The next macro shows that Icons in Records can also be used to filter datasets.

Macro Filtering By Icon
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub FilterIcon()
    Range("A1:A1").AutoFilter Field:=8, _
    Criteria1:=ActiveWorkbook.IconSets(xl5ArrowsGray).Item(5), _
    Operator:=xlFilterIcon
End Sub
Dynamic Filters // the book touches on doing this in the Excel GUI as well

Perhaps the most powerful filters in Excel are dynamic filters which allow selecting ranges based on special criteria. They are somewhat like virtual tables found in SQL Database searches.

Following is a brief example followed by a list of dynamic criteria options.

**Macro Filtering By Dynamic Filter**

// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

```vba
Sub FilterDynamicDate()
    Range("A1").AutoFilter Field:=3, _
    Criteria1:=xlFilterNextYear, _
    Operator:=xlFilterDynamic
End Sub
```

**The List of Dynamic Criteria Option**

- **Values**
  - xlFilterAboveAverage
  - xlFilterBelowAverage

- **Future Periods**
  - xlFilterTomorrow
  - xlFilterNextWeek ‘starts Sunday’
  - xlFilterNextMonth
  - xlFilterNextQuarter
  - xlFilterNextYear

- **Current Periods**
  - xlFilterToday
  - xlFilterThisWeek
  - xlFilterThisMonth
  - xlFilterThisQuarter
  - xlFilterThisYear

- **Past Periods**
  - xlFilterYesterday
  - xlFilterLastWeek
  - xlFilterLastMonth
  - xlFilterLastQuarter
  - xlFilterLastYear
• Specific Quarters
  • xlFilterDatesInPeriodQuarter1
  • xlFilterDatesInPeriodQuarter2
  • xlFilterDatesInPeriodQuarter3
  • xlFilterDatesInPeriodQuarter4

• Specific Months
  • xlFilterDatesInPeriodJanuary  // through to
  • xlFilterDatesInPeriodDecember

As of Microsoft Excel 2013 the 2013 book states that combining these criteria is not supported yet.

Selecting Visible Cells Exclusively  // See Figure 11.3 in 2013 book page 203

If rows have been hidden using the 'Hide Rows' command, any formatting applied to the CurrentRegion will also be applied to the hidden rows. To avoid this you need to use the 'Visible Cells Only' option of the 'Go To Special Dialog'.

To achieve the same result in VBA use the following line of code.

Example

Range("A1"). CurrentRegion.SpecialCells(xlCellTypeVisible)

The Go To Special Dialog Case Study

The Case Study on page 204 of the 2013 book shows how using a 'Go To Special Dialog' feature to hide rows that evaluated to a text value, rather than hiding that data in a loop made a sub work much more efficiently

Using Advanced Filters in VBA vs Excel

Using Advanced Filters in the Excel interface is much less convenient than using them in VBA code.

As a general strategy Mr. Excel suggests that two advanced filters are frequently used together, one to get a unique list of customers and a second a filter that addresses some aspect on an individual customer basis.
In VBA, advanced filtering is based on the AdvancedFilter method which has the following parameters.

**AdvancedFilter Method Arguments**

- **Action**
  - `xlFilterInPlace`
  - `xlFilterCopy`
    - if using `xlFilterCopy`
    - specify where to copy to
    - via `CopyToRange`
    - example, `CopyToRange("K1")`

- **Criteria**
  - optional and may be omitted
  - include `CriteriaRange` to filter a range
  - example, `CriteriaRange:= Range(L1:L2)`

- **Unique**
  - to return only unique values
  - specify `Unique:= True`

Mr. Excel does little to characterize how the advanced filter works. The following article at the Microsoft site provides a deeper explanation of what Advanced Filters are.

**Advanced Filter Article @ Microsoft**


"The following code sets up a single column output range two columns to the right of the last-used column in the data range." -Mr. Excel

**Get Unique Customer Macro**

// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub GetUniqueCustomers()

    Dim IRange As Range
    Dim ORange As Range

    ' Find the size of today's dataset
    FinalRow = Cells(Rows.Count, 1).End(xlUp).Row
    NextCol = Cells(1, Columns.Count).End(xlToLeft).Column + 2

End Sub
' Set up output range. Copy heading from D1 there
Range("D1").Copy Destination:=Cells(1, NextCol)
Set ORange = Cells(1, NextCol)

' Define the Input Range
Set IRange = Range("A1").Resize(FinalRow, NextCol - 2)

' Do the Advanced Filter to get unique list of customers
IRange.AdvancedFilter Action:=xlFilterCopy, CopyToRange:=ORange, Unique:=True
End Sub

Revisiting The End Method

The Range object returned by the End method represents the cell at the end of the region that contains the source range. Using the End method is equivalent to pressing End+Up Arrow, End+Down Arrow, End+Left Arrow, or End+Right Arrow.

This example selects the cell at the top of column B in the region that contains cell B4.

VBA
Range("B4").End(xlUp).Select

The following reduced code does the same thing as the above dispensing with the Range variables. This is done so it is more obvious that the key action is still happening in a single filter line.

Reduced Version of the Above Macro
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub UniqueCustomerRedux()
' Copy a heading to create an output range
        Range("J1").Value = Range("D1").Value
' Do the Advanced Filter
        Range("A1").CurrentRegion.AdvancedFilter xlFilterCopy, _
        CopyToRange:=Range("J1"), Unique:=True
End Sub

The next code sample sorts the unique list of customers found and adds a SUMIF formula to get total revenue by customer.
Sub RevenueByCustomers()

    Dim IRange As Range
    Dim ORange As Range

    ' Find the size of today's dataset
    FinalRow = Cells(Rows.Count, 1).End(xlUp).Row
    NextCol = Cells(1, Column.Count).End(xlToLeft).Column + 2

    ' Set up output range. Copy heading from D1 there
    Range("D1").Copy Destination:=Cells(1, NextCol)
    Set ORange = Cells(1, NextCol)

    ' Define the Input Range
    Set IRange = Range("A1").Resize(FinalRow, NextCol - 2)

    ' Do the Advanced Filter to get unique list of customers
    IRange.AdvancedFilter Action:=xlFilterCopy, CopyToRange:=ORange, _
        Unique:=True

    ' Determine how many unique customers we have
    LastRow = Cells(65536, NextCol).End(xlUp).Row

    ' Sort the data
    Cells(1, NextCol).Resize(LastRow, 1).Sort Key1:=Cells(1, NextCol), _
        Order1:=xlAscending, _
        Header:=xlYes

    ' Add an array formula to get totals
    Cells(1, NextCol + 1).Value = "Revenue"
    Cells(2, NextCol + 1).FormulaArray = _
        "=SUM((R2C4:R" & FinalRow & "C4=RC[-1])*R2C6:R" & FinalRow & "C6)"
    If LastRow > 2 Then
        Cells(2, NextCol + 1).Copy Cells(3, NextCol + 1).Resize(LastRow - 2, 1)
    End If
End Sub

Code Populating a List Box or Combo Box Then Accessing The List

A Unique List of Customers can be transferred to a List Box or Combo Box. From there further action can be taken on individual customers on the list. This code is shown on page 262 and 263 in the 2010 book and pages 209 and 210 of the new book. This Chapter is too long to cover in a single class so this is code is left to the reader to look at.
Two Fields As Filter Criteria

The following code supplies a formula to find all the unique combinations of two fields, Customer and Product. See Figure 12-10 on page 265 of the 2010 book or Figure 11-8 on page 211 of the 2013 book. This might be useful to providing reports on customer by product or product by customer.

Unique Combinations of Two or More Fields
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub UniqueCustomerProduct()
    ' Find the size of today's dataset
    FinalRow = Cells(65536, 1).End(xlUp).Row
    NextCol = Cells(1, 255).End(xlToLeft).Column + 2

    ' Set up output range. Copy heading from D1 there
    Range("D1").Copy Destination:=Cells(1, NextCol)
    Range("B1").Copy Destination:=Cells(1, NextCol + 1)

    Set ORange = Cells(1, NextCol).Resize(1, 2)

    ' Define the Input Range
    Set IRange = Range("A1").Resize(FinalRow, NextCol - 2)

    ' Do the Advanced Filter to get unique list of customers & product
    IRange.AdvancedFilter Action:=xlFilterCopy, CopyToRange:=ORange, Unique:=True

    ' Determine how many unique rows we have
    LastRow = Cells(65536, NextCol).End(xlUp).Row

    ' Sort the data
    Cells(1, NextCol).Resize(LastRow, 2).Sort Key1:=Cells(1, NextCol), _
        Order1:=xlAscending, Key2:=Cells(1, NextCol + 1), _
        Order2:=xlAscending, Header:=xlYes
End Sub

The following reduction is provided just so we don't lose site that what we are doing is a filter on a set of values and copying the result to a new range.

Equivalent Simplified Code Without the Last Bit of Sorting

Sub Advanced()
    Range("K1").Value = Range("B1").Value
    Range("J1").Value = Range("D1").Value
    Range("A1").CurrentRegion.AdvancedFilter Action:=xlFilterCopy, _
        CopyToRange:=Range("J1:K1"), Unique:=True
End Sub
The Two Aspects of Criteria Range in Advanced Filters

In Advanced Filters when Criteria Ranges are used, the first row holds one or more field header values which are matched to the data range being filtered.

The second row holds the operators used by the filter

}// refer back to the Microsoft article for examples

Using Criteria Ranges

Mr.Excel (on page 265, 2010 book, 212 in 2013 book) describes setting up a criteria range of two or more rows in a blank area of the worksheet. The figure 12.11 indicates Range J1:J2 is used for the criteria.

While we don't have time to look at this code, you can see it in action in spreadsheets that are part of the download resource.

Using Logical OR in Criteria

To create a logical OR scenario for two criteria put the criteria items on subsequent rows. If either of the products are found in the searched range they will be selected.

OR Example In a Criteria Range

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Products</td>
</tr>
<tr>
<td></td>
<td>X - 15</td>
</tr>
<tr>
<td></td>
<td>X - 31</td>
</tr>
</tbody>
</table>

Using Logical AND in Criteria

A Logical AND is created by placing two items in adjacent columns. The following example searches for the product X – 15 purchased in the East region.

AND Example In a Criteria Range

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Products</td>
</tr>
<tr>
<td></td>
<td>X - 15</td>
</tr>
</tbody>
</table>
More Complex Criteria

Arranging different elements from different columns on subsequent rows returns a result where records are selected where a product is found OR a record associated with the East region as is shown in the following example.

More Complex OR Example In a Criteria Range J1:K3

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Products</td>
<td>Region</td>
</tr>
<tr>
<td>East</td>
<td></td>
</tr>
<tr>
<td>X-31</td>
<td></td>
</tr>
</tbody>
</table>

// similar to Figure 12.15

Using a Formula in the Criteria of an Advanced Filter

To use a formula to do filtering the first row of the criteria range is left blank. Then in the second row, a formula is used that resolves to a boolean value, True or False.

Mr. Excel offers the example, to select all the records where Gross Profit Percentage is below 53%, the formula references the Profit in H2 and the Revenue in F2.

J1 is left blank and J2 holds the formula = (H2 / F2) < 0.53 . The Criteria range is then specified as “J1:J2”.

Excel will apply the formula to each row in the database. Where the formula evaluates to true it is included in the output range.

This technique is very fast and efficient.

Catching Errors For No Records When Using Filter in Place

As when using 'Copy' one needs to watch for the possibility that no records are found that match the criteria. The .SpecialCells method in this case will return a Runtime Error 1004.

Code like the following is useful to catch such an error.

Example Code Excerpt
On Error GoTo NoRecs
  For Each cell In Range("A2:A" & FinalRow).SpecialCells(xlCellTypeVisible)
    Ctr = Ctr + 1
    Next cell
  On Error GoTo 0
  MsgBox Ctr & " cells match the criteria"
Exit Sub

NoRecs:
  MsgBox "No records match the criteria"
End Sub

Using xlFilterCopy with All Records ( Unique Set to Default Value, False )

Sometimes you will want to select values that are not necessarily unique. In this case the Unique property does not need to be set as it is set to False by default.

A couple of tips Mr Excel offers;

• for a subset of fields, only copy those fields to the output range
• to reorder fields, reorder them in the output range.

To copy all columns, specify a single blank cell as the output range. This is examplified in the next macro.

Turning Off Drop Downs in the AutoFilter

Mr. Excel reports that one feature apparently available only in Excel VBA, and not in Excel is code where drop downs can be made selectively visible. This allows eliminating from view data columns that may be distracting.

Turning off Drop Downs For Selected Columns
  // from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub AutoFilterCustom( )
  Range("A1").AutoFilter Filed:=3, VisibleDrowDown:=False
  Range("A1").AutoFilter Filed:=5, VisibleDrowDown:=False
  Range("A1").AutoFilter Filed:=6, VisibleDrowDown:=False
  Range("A1").AutoFilter Filed:=7, VisibleDrowDown:=False
  Range("A1").AutoFilter Filed:=8, VisibleDrowDown:=False
End Sub
The following code is used to clear the selection.

Clearing the Selection
// from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Sub SimpleFilter()
    Worksheets("SalesReport").Select
    Range("A1").AutoFilter
    Range("A1").AutoFilter Field:= 4
End Sub

Assignment

No assignment to hand in on this one!