Advanced Reporting with Microsoft Excel

Doug Hennig
Stonefield Software Inc.
Email: doug@doughennig.com
Corporate Web sites: stonefieldquery.com
stonefieldsoftware.com
Personal Web site: DougHennig.com
Blog: DougHennig.BlogSpot.com
Twitter: DougHennig

It's easy to add a feature to your application to export to Microsoft Excel: XFRX, Craig Boyd's GridExtras, several VFPX projects including Greg Green's excellent WorkbookXLSX, and even a simple COPY TYPE XL5 can create an Excel document. However, such documents are simple, boring lists of data. This session looks at techniques to create attractive, useful reports in Excel, driven with data from your VFP applications.

Introduction

Microsoft Excel is pretty much ubiquitous in business settings. It's not only good for calculations, but thanks to its interactive nature and variety of data visualizations (PivotTables, PivotCharts, maps, etc.), it's highly used as a reporting tool.

It's so easy to add an Excel export feature to your applications that not providing it seems like a missing feature. In this document, we'll look at a variety of techniques to do that but also how to make your Excel documents pop so users won't find them just to be a boring lists of data.

Creating Excel documents

Let's look at a variety of ways to create Excel documents from VFP data.

COPY TO

The COPY TO command supports several formats Excel can open:

- XLS: generates an Excel 2.0 worksheet file.
- XL5: generates an Excel 5.0 worksheet file.
- CSV: generates a comma-separated values (CSV) file with a header row of field names.
- Delimited: generates a CSV file (with a TXT extension unless you specify otherwise) without a header row.

The advantages of COPY TO are:

- It's fast.
- It only needs a single line of code and there are no dependencies on other tools.
- The workstation doesn't need to have Excel installed to create the file, only to open it.

The disadvantages are:

- The XLS and XL5 formats are old, so they may need to be converted to the newer XLSX format, and only support 65,535 rows, which may be a deal-breaker. Also, XLS files can't be edited by default due to policy settings.
- Memo fields aren't supported, which also may be a deal-breaker.
- You have no control over formatting.
- COPY TO creates a new file so templates aren't supported (I'll discuss templates later in this document).
- Some values of character fields may be treated as numeric in CSV files. For example, a product code like "10E06" is treated as scientific notation, resulting in "1.00E+06."

- Logical values are treated as text ("T" and "F") rather than Boolean in CSV files.
- Dates are output to CSV using your SET DATE format, which may cause a problem when Excel imports them. Also, blank dates are treated as text (" / / " or " - - ") rather than empty dates.

If the disadvantages aren't a concern (for example, there are less than 65,535 rows and no memo fields) and you don't care about formatting, COPY TO XL5 is a reasonable choice for quick-and-dirty output.

CSVProcessor

CSVProcessor is a VFPX project (https://github.com/atlopes/csv) from Antonio Lopes that provides additional control over CSV output. For example, it supports outputting memo fields.

CSVProcessor has a dependency on another VFPX project, Name Syntax Checker (https://github.com/atlopes/names), so you must install both and add several of the PRGs to your project.

Here's an example of creating a CSV file:

```
do CSV
loCSV = createobject('CSVProcessor')
use sample
* Set loCSV.HeaderRow to .F. to omit header row
loCSV.LogicalTrue  = 'TRUE'
loCSV.LogicalFalse = 'FALSE'
loCSV.Export('doc6.csv')
```

CSVProcessor has several advantages over COPY TO CSV:

- Dates are output as YYYY-MM-DD so Excel can understand the format properly, and blank dates are output as blank.
- You can configure CSVProcessor to output logical values as "TRUE" and "FALSE", which Excel expects, rather than text.
- Memo fields are supported, including carriage returns.

However, the result is still a CSV file so some of the disadvantages discussed in the previous section are still applicable.

GridExtras

As its name suggests, Craig Boyd's GridExtras provides additional features to grids: hiding and showing columns, rearranging columns, searching and filtering, and, pertinent to this discussion, output to Excel. Adding GridExtras to a grid is easy: drop a GridExtra object from GridExtras.vcx onto a form and tell it which grid its associated with using code like this:

```
This.oGridExtra.GridExpression = 'Thisform.grdCustomers'
This.oGridExtra.Setup()
```

Right-click any of the headers in the grid to display the GridExtra menu shown in **Figure 1**. I won't discuss the other features of GridExtras here, just the Excel Export function.

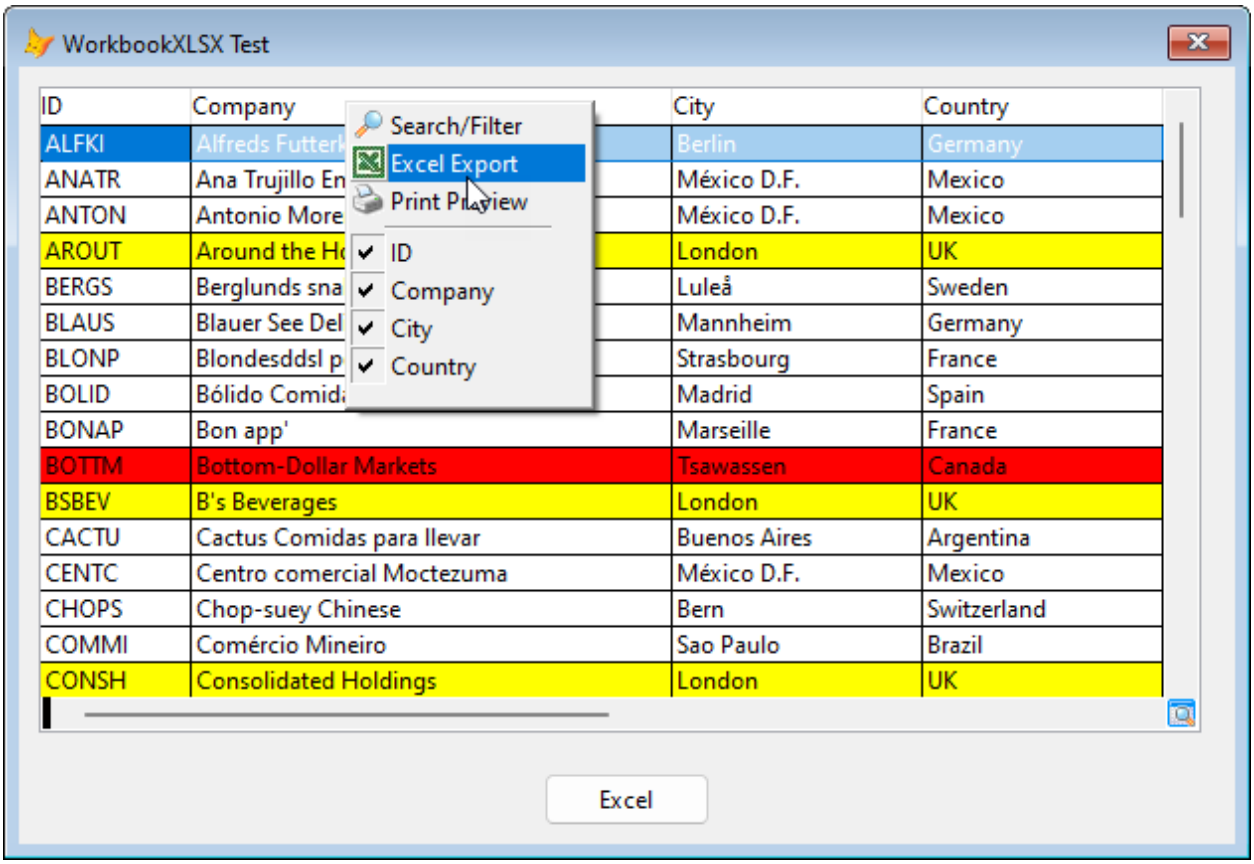


Figure 1. GridExtras adds a menu of features to a VFP grid.

The Excel Export function creates a simple Excel document containing the visible columns in the grid but, unlike WorkbookXLSX which we'll discuss later, not respecting any formatting of the grid. Note that Excel must be installed along with the Microsoft Access Database Engine.

What makes GridExtras interesting is how it performs the export. All the code is in GridExtraProcs.prg. The CopyToExcel procedure begins by calling CreateExcelTemplate to create an empty XLSX document. It does that by generating a binary file from a set of hard-coded bytes. CopyToExcel then creates an OLE DB connection to the document:

```
m.loConnection = CreateObject ( "ADODB.Connection")
m.loConnection.ConnectionString = [Provider=Microsoft.ACE.OLEDB.12.0;Data Source="] +
   m.tcXLSFile + [";Extended Properties="Excel 12.0 Xml;HDR=Yes;";]
m.loConnection.Open()
```

It finally uses a CursorAdapter to create and update a "table" (really, a worksheet) in the document from the records in the cursor displayed in the grid.

GridExtras pre-dates WorkbookXLSX, so while it was an interesting technique in its day, there are better mechanisms now as we'll see. However, if you want to use the functionality of GridExtras, you get rudimentary Excel output with no extra work so it may be worth considering.

XFRX

XFRX (https://eqeus.com) is a reporting add-on for VFP developers. It provides a customizable preview window that has a lot more features than the VFP preview window and supports output to many different file formats, including PDF, Microsoft Word, and Excel. FoxyPreviewer, another reporting add-on, also supports Excel output but I haven't used it so I won't discuss it. Rick Schummer presented a session comparing XFRX and FoxyPreviewer titled "Visual FoxPro Reporting: XFRX vs. FoxyPreviewer" at the May 2022 Virtual Fox Fest (https://virtualfoxfest.com). You can watch a video of his presentation at https://youtu.be/dOAM70CFnt8.

XFRX has two mechanisms to output to Excel: using a report with a custom ReportListener and using the XFRX_CopyToXLSX function.

ReportListener

XFRX works with any VFP FRX report because it uses a ReportListener subclass to do its magic. Here's an example:

```
loListener = XFRX('XFRX#LISTENER')
lnReturn = loListener.SetParams('xfrx1.xlsx', , .T., , , , 'NATIVE_FXLSX')
if lnReturn = 0
   report form Customers object loListener
endif lnReturn = 0
```

The SetParams method specifies options such the name of the file to create, the type of output, and whether to open the file after creation.

XFRX has four ways to output to Excel. "XLS" and "XLSPLAIN" generate XLS files and require Excel to be installed. "NATIVE_FXLSX" and "NATIVE_PFXLSX" generate XLSX files and don't require Excel. "NATIVE_PFXLSX" is a much cleaner output so that's the one I recommend. In both cases, the Excel document looks as much like the output of the report as possible; see **Figure 2**. The XFRX documentation (https://eqeuscom.atlassian.net/wiki/spaces/DOC/pages/3899449/Excel+specific+features) provides details for adjustments you can make to the output.

	A	B	C	D	E	F	G	H	I
1					Customer Listing				
2	06/18/202							Page	1
4	ID	Company Name	Contact Name		City	Region	Country		
6	ALFKI	Alfreds Futterkiste	Maria Anders		Berlin	.NULL.	Germany		
7	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo		México D.F.	.NULL.	Mexico		
8	ANTON	Antonio Moreno Taquería	Antonio Moreno		México D.F.	.NULL.	Mexico		
9	AROUT	Around the Horn	Thomas Hardy		London	.NULL.	UK		
10	BERGS	Berglunds snabbköp	Christina Berglund		Luleå	.NULL.	Sweden		
11	BLAUS	Blauer See Delikatessen	Hanna Moos		Mannheim	.NULL.	Germany		
12	BLONP	Blondesddsl père et fils	Frédérique Citeaux		Strasbourg	.NULL.	France		
13	BOLID	Bólido Comidas preparadas	Martín Sommer		Madrid	.NULL.	Spain		
14	BONAP	Bon app'	Laurence Lebihan		Marseille	.NULL.	France		
15	BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln		Tsawassen	BC	Canada		
16	BSBEV	B's Beverages	Victoria Ashworth		London	.NULL.	UK		
	CACTU	Cactus Comidas para	Patricio Simpson		Buenos Aires	.NULL.	Argentina		

Figure 2. Excel documents created by XFRX look as much like the report as possible.

While XFRX makes attractive Excel documents (well, as attractive as your FRX report is), I don't think they're as useful as other types of output because of the formatting. If I want a file to look as much like the VFP report as possible, I usually output it to PDF.

XFRX_CopyToXLSX

The XFRX_CopyToXLSX function in XFRX.prg creates an Excel document from a cursor using a single command. Here's an example:

```
lnError = XFRX_CopyToXLSX(alias(), 'Documents\xfrx3.xlsx', , 'Customers')
```

It also supports formatting options, including conditional formatting, column width, header names and styles, and text trimming, using a callback function specified as the third parameter. However, I couldn't get the example code at https://eqeuscom.atlassian.net/wiki/spaces/DOC/pages/10485784/Data+export+to+XLSX+and+ODS to work so I wasn't able to test it.

If you already have XFRX, this is a quick way to create an Excel document.

Excel automation

Assuming it's installed on the computer, Excel is available as a COM object you can control from VFP. Instantiate Excel.Application and then set properties and call methods as necessary. To open an existing Excel document, use code like:

```
loExcel    = createobject('Excel.Application')
loDocument = loExcel.Workbooks.Open(lcDocument)
```

Let's look at an example: creating a PivotTable from VFP data. This code, taken from CreatePivotTable.prg, assumes an Excel document contains the source data for a PivotTable in Sheet1. First, get an object for the source data:

```
with loExcel
   lnRecords  = reccount()
   lnColumns  = fcount()
   lcEndRange = alltrim(GetColumnletter(lnColumns)) + alltrim(str(lnRecords + 1))
   loSource   = .Sheets[1].Range('A1:' + lcEndRange)
```

Next, add a new sheet named PivotTable to the document and create a PivotTable in the upper left corner:

```
   loPivotSheet      = .WorkSheets.Add()
   loPivotSheet.Name = 'PivotTable'
   loDestination     = loPivotSheet.Range('A1')
   try
      loPivot = loPivotSheet.PivotTableWizard(1, loSource, loDestination, ;
         'PivotTable', .T., .T.)
   catch to loException
      llReturn = .F.
   endtry
```

This code sets up the PivotTable. taFields is an array containing the name of each field to be used in the PivotTable, where in the PivotTable to place it (row, column, or value), and the numeric format.

```
   lnFields = alen(taFields, 1)
   for lnI = 1 to lnFields
      if taFields[lnI, 2] <> xlHidden
         lcCurField = taFields[lnI, 1]
         loField    = loPivot.PivotFields[lcCurField]
         loField.Orientation = taFields[lnI, 2]
         if taFields[lnI, 2] = xlDataField
            loField.NumberFormat = taFields[lnI, 3]
         endif taFields[lnI, 2] = xlDataField
      endif taFields[lnI, 2] <> xlHidden
   next lnI
```

Finally, save the spreadsheet:

```
   loExcel.DisplayAlerts = .F.
   loDocument.Save()
   loExcel.Quit()
endwith
```

Although there's full documentation for the Excel object model online, you might find creating an Excel script (formerly called macros) useful to determine how to automate Excel. On the Automate tab, click Record Actions, do the steps you want to automate, click Stop, and then look at the generated code in the Code Editor. For example, I selected some cells, chose Format, Format Cells, Font, and selected Bold. The following code was generated:

```
function main(workbook: ExcelScript.Workbook) {
   let selectedSheet = workbook.getActiveWorksheet();
   // Set font bold to true for range B8:C15 on selectedSheet
   selectedSheet.getRange("B8:C15").getFormat().getFont().setBold(true);
}
```

Converting a script to VFP can be tricky. In this case, the VFP equivalent is:

```
loExcel    = createobject('Excel.Application')
loWorkBook = loExcel.Workbooks.Open(lcDocument)
loSheet    = loWorkBook.ActiveSheet()
loSheet.Range("B8:C15").Font.Bold = .T.
loWorkBook.Save()
loExcel.Quit()
```

The code in yellow was converted from the script. The rest of the code is needed to instantiate Excel, open the document, save the document, and quit Excel. But the converted code isn't a direct correlation to the script code. There is no getRange method of a Sheet; instead it's Range. Instead of using setBold, you set the Bold property. The best way to figure out what to do is to debug the code and let VFP IntelliSense help you with object members as you can see in **Figure 3**. You can even set loExcel.Visible to .T. so you can immediately see the effect of a command as you execute it.

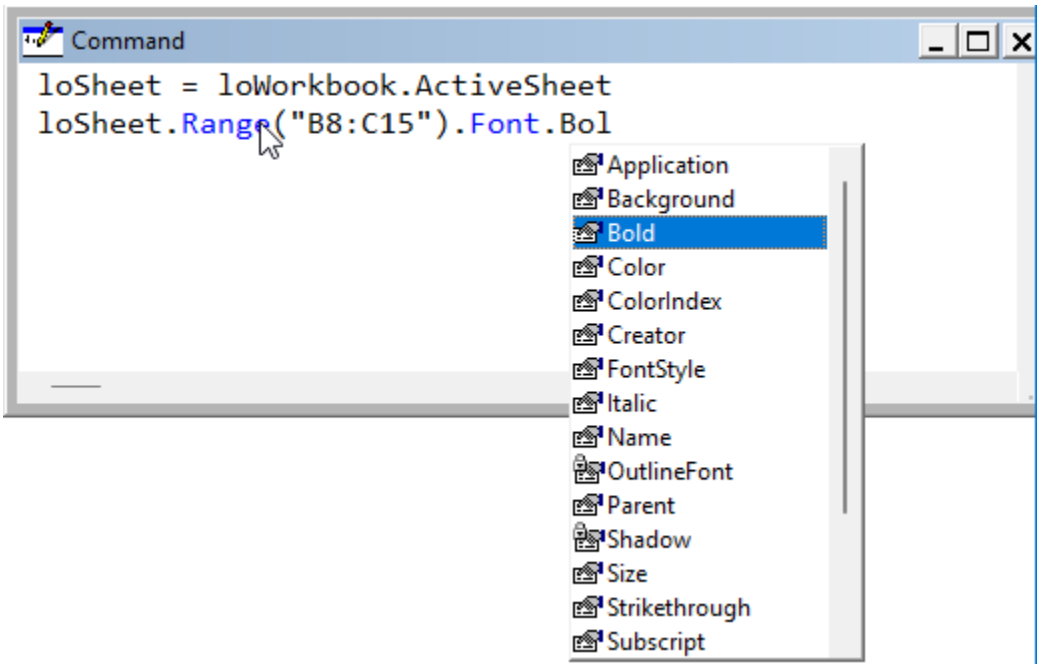


Figure 3. VFP IntelliSense can help with Excel automation.

The advantages of using Excel automation are:

- You have full access to the Excel object model so you can do anything the object model supports.
- VFP provides IntelliSense for the Excel COM object.

- Because you can read an existing document, make whatever changes are necessary, and save to a different document, you can use Excel automation for template-based reporting, discussed later in this document.

The disadvantages are:

- Excel must be installed on the workstation.
- Excel automation is slow compared to other techniques.
- It usually involves writing a lot of code.
- You have to learn the Excel API to do anything more than the simplest tasks.
- You must change the code when the document being output or the template changes.

WorkbookXLSX

Greg Green has created an incredible tool named WorkbookXLSX, available as a VFPX project at https://github.com/ggreen86/XLSX-Workbook-Class. WorkbookXLSX has a large set of features, including:

- Writes XLSX documents directly; Excel doesn't have to be installed on the computer.
- Supports creating a document from a VFP cursor or grid with a single method call.
- Reads existing XLSX documents so you can make necessary changes and save to the same or a new document.
- Allows you to programmatically assign values and formats (value formats, borders, colors, etc.) to cells, merge cells, and many other operations.

WorkbookXLSX consists of a single class: VFPXWorkbookXLSX in VFPXWorkbookXLSX.vcx. You can either drop it on a form or instantiate it into a variable and call the desired methods. Here's an example that creates Customers.xlsx from a cursor named Customers:

```
loExcel = newobject('VFPxWorkbookXLSX', 'VFPxWorkbookXLSX.vcx')
loExcel.SaveTableToWorkbookEx('Customers', 'Customers.xlsx')
```

The resulting document is shown in **Figure 4**.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Customer	Company	Contact N	Contact Ti	Address	City	Region	Postal Coc	Country	Phone	Fax	
2	ALFKI	Alfreds Fu	Maria Anc	Sales Repı	Obere Str.	Berlin		12209	Germany	030-00743	030-0076545	
3	ANATR	Ana Trujill	Ana Trujill	Owner	Avda. de l	México D.F.		05021	Mexico	(5) 555-47	(5) 555-3745	
4	ANTON	Antonio M	Antonio M	Owner	Matadero	México D.F.		05023	Mexico	(5) 555-3932		
5	AROUT	Around th	Thomas H	Sales Repı	120 Hanov	London		WA1 1DP	UK	(171) 555-	(171) 555-6750	
6	BERGS	Berglunds	Christina I	Order Adr	Berguvsvä	Luleå		S-958 22	Sweden	0921-12 34	0921-12 34 67	
7	BLAUS	Blauer Se	Hanna Mo	Sales Repı	Forsterstr	Mannheim		68306	Germany	0621-0846	0621-08924	
8	BLONP	Blondesd	Frédériqu	Marketing	24, place	Strasbourg		67000	France	88.60.15.3	88.60.15.32	
9	BOLID	Bólido Co	Martín Sor	Owner	C/ Araquil	Madrid		28023	Spain	(91) 555 2	(91) 555 91 99	
10	BONAP	Bon app'	Laurence	Owner	12, rue de	Marseille		13008	France	91.24.45.4	91.24.45.41	
11	BOTTM	Bottom-D	Elizabeth	Accountin	23 Tsawas	Tsawasser	BC	T2F 8M4	Canada	(604) 555-	(604) 555-3745	
12	BSBEV	B's Bevera	Victoria A	Sales Repı	Fauntlero	London		EC2 5NT	UK	(171) 555-1212		
13	CACTU	Cactus Co	Patricio Si	Sales Age	Cerrito 33	Buenos Aires		1010	Argentina	(1) 135-55	(1) 135-4892	
14	CENTC	Centro co	Francisco	Marketing	Sierras de	México D.F.		05022	Mexico	(5) 555-33	(5) 555-7293	
15	CHOPS	Chop-sue	Yang Wan	Owner	Hauptstr.	Bern		3012	Switzerlar	0452-076545		
16	COMMI	Comércio	Pedro Afo	Sales Assc	Av. dos Lu	Sao Paulo	SP	05432-043	Brazil	(11) 555-7647		
17	CONSH	Consolida	Elizabeth	Sales Repı	Berkeley	London		WX1 6LT	UK	(171) 555-	(171) 555-9199	
18	DRACD	Drachenbl	Sven Ottli	Order Adr	Walserwe	Aachen		52066	Germany	0241-0391	0241-059428	
19	DUMON	Du monde	Janine Lab	Owner	67, rue de	Nantes		44000	France	40.67.88.8	40.67.89.89	
20	EASTC	Eastern Co	Ann Devo	Sales Age	35 King Ge	London		WX3 6FW	UK	(171) 555-	(171) 555-3373	
21	ERNSH	Ernst Han	Roland M	Sales Man	Kirchgasse	Graz		8010	Austria	7675-3425	7675-3426	
22	FAMIA	Familia Ar	Aria Cruz	Marketing	Rua Orós,	Sao Paulo	SP	05442-030	Brazil	(11) 555-9857		
23	FISSA	FISSA Fab	Diego Roe	Accountin	C/ Moralz	Madrid		28034	Spain	(91) 555 9	(91) 555 55 93	
24	FOLIG	Folies gou	Martine R	Assistant	184, chaus	Lille		59000	France	20.16.10.1	20.16.10.17	
25	FOLKO	Folk och f	Maria Lars	Owner	Åkergatan	Bräcke		S-844 67	Sweden	0695-34 67	21	
26	FRANK	Frankenve	Peter Frar	Marketing	Berliner P	München		80805	Germany	089-08773	089-0877451	
27	FRANR	France res	Carine Sch	Marketing	54, rue Ro	Nantes		44000	France	40.32.21.2	40.32.21.20	
28	FRANS	Franchi S.	Paolo Acc	Sales Repı	Via Monte	Torino		10100	Italy	011-49882	011-4988261	
29	FURIB	Furia Baca	Lino Rodri	Sales Man	Jardim da	Lisboa		1675	Portugal	(1) 354-25	(1) 354-2535	
30	GALED	Galería de	Eduardo S	Marketing	Rambla de	Barcelona		08022	Spain	(93) 203 4	(93) 203 4561	

Customers

Figure 4. WorkbookXLSX can create an Excel document from a cursor with a single method call.

That's a boring-looking document, so let's pass some additional parameters to auto-fit columns and create a table format:

```
loExcel.SaveTableToWorkbookEx('Customers', 'Customers.xlsx', .NULL., .T., ;
   'Customers', 1, 1, .T., TABLE_STYLE_LIGHT13)
```

TABLE_STYLE_LIGHT13 is a constant defined in VFPXWorkbookXLSX.h that specifies a particular table style. The result, shown in **Figure 5**, is more attractive.

	A	B	C	D	E
1	Custo	Company Name	Contact Name	Contact Title	Address
2	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57
3	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222
4	ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312
5	AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.
6	BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8
7	BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57
8	BLONP	Blondesddsl père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber
9	BOLID	Bólido Comidas preparadas	Martín Sommer	Owner	C/ Araquil, 67
10	BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers
11	BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.
12	BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circus
13	CACTU	Cactus Comidas para llevar	Patricio Simpson	Sales Agent	Cerrito 333
14	CENTC	Centro comercial Moctezuma	Francisco Chang	Marketing Manager	Sierras de Granada 9993
15	CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29
16	COMMI	Comércio Mineiro	Pedro Afonso	Sales Associate	Av. dos Lusíadas, 23
17	CONSH	Consolidated Holdings	Elizabeth Brown	Sales Representative	Berkeley Gardens 12 Brewery
18	DRACD	Drachenblut Delikatessen	Sven Ottlieb	Order Administrator	Walserweg 21
19	DUMON	Du monde entier	Janine Labrune	Owner	67, rue des Cinquante Otages
20	EASTC	Eastern Connection	Ann Devon	Sales Agent	35 King George
21	ERNSH	Ernst Handel	Roland Mendel	Sales Manager	Kirchgasse 6
22	FAMIA	Familia Arquibaldo	Aria Cruz	Marketing Assistant	Rua Orós, 92
23	FISSA	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	Accounting Manager	C/ Moralzarzal, 86
24	FOLIG	Folies gourmandes	Martine Rancé	Assistant Sales Agent	184, chaussée de Tournai
25	FOLKO	Folk och fä HB	Maria Larsson	Owner	Åkergatan 24
26	FRANK	Frankenversand	Peter Franken	Marketing Manager	Berliner Platz 43
27	FRANR	France restauration	Carine Schmitt	Marketing Manager	54, rue Royale
28	FRANS	Franchi S.p.A.	Paolo Accorti	Sales Representative	Via Monte Bianco 34
29	FURIB	Furia Bacalhau e Frutos do Mar	Lino Rodriguez	Sales Manager	Jardim das rosas n. 32
30	GALED	Galería del gastrónomo	Eduardo Saavedra	Marketing Manager	Rambla de Cataluña, 23

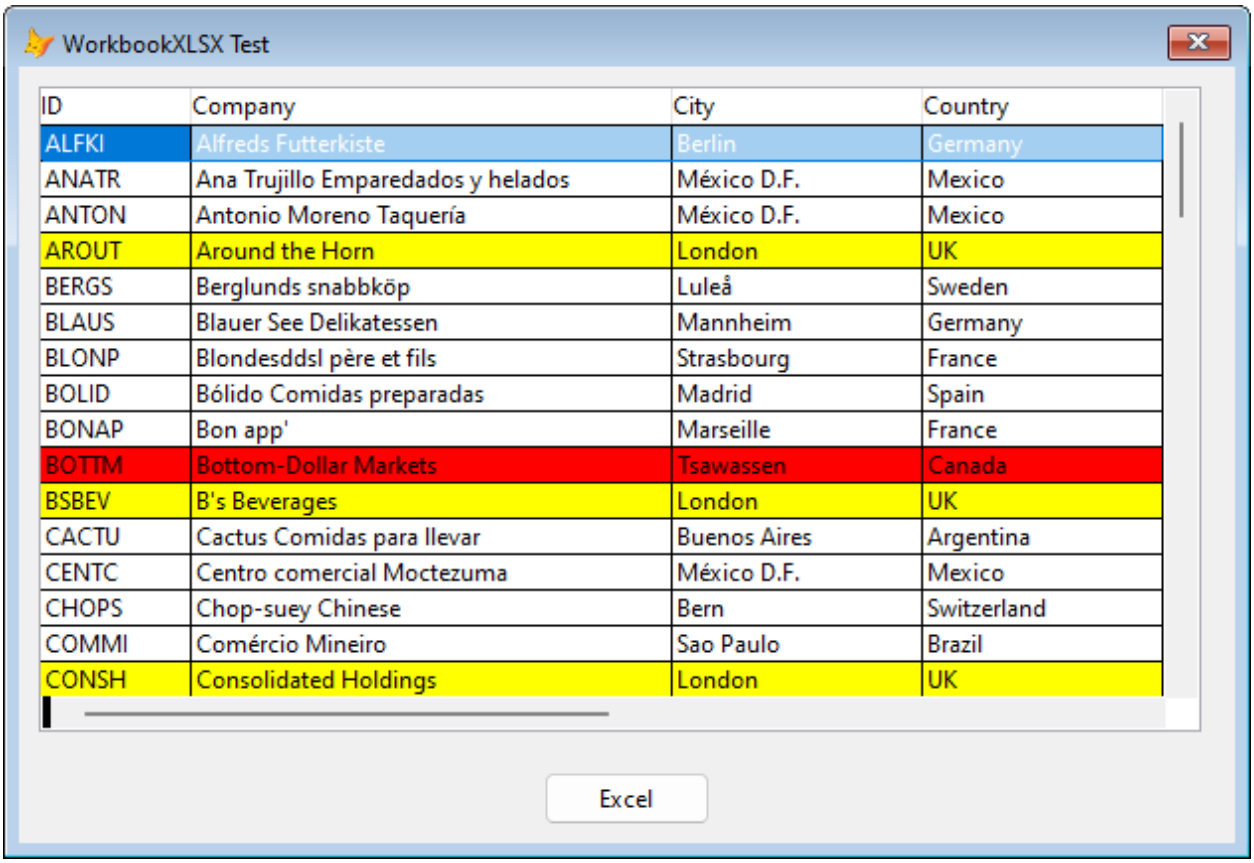
Customers

Figure 5. Specifying additional parameters can make a more attractive document.

You can create a document from a grid by calling SaveGridToWorkbook. It respects the formatting of the grid, including fonts, column widths, and colors (including dynamic colors).

```
lnWB = loExcel.CreateWorkbook('Customers.xlsx')
loExcel.SaveGridToWorkbook(Thisform.grdCustomers, lnWB, .T., .T., 'Customers', ;
   .F., .T.)
```

This code is taken from WorkbookXLSXTest.scx, included with the sample files accompanying this document and shown in **Figure 6**.



ID	Company	City	Country
ALFKI	Alfreds Futterkiste	Berlin	Germany
ANATR	Ana Trujillo Emparedados y helados	México D.F.	Mexico
ANTON	Antonio Moreno Taquería	México D.F.	Mexico
AROUT	Around the Horn	London	UK
BERGS	Berglunds snabbköp	Luleå	Sweden
BLAUS	Blauer See Delikatessen	Mannheim	Germany
BLONP	Blondesddsl père et fils	Strasbourg	France
BOLID	Bólido Comidas preparadas	Madrid	Spain
BONAP	Bon app'	Marseille	France
BOTTM	Bottom-Dollar Markets	Tsawassen	Canada
BSBEV	B's Beverages	London	UK
CACTU	Cactus Comidas para llevar	Buenos Aires	Argentina
CENTC	Centro comercial Moctezuma	México D.F.	Mexico
CHOPS	Chop-suey Chinese	Bern	Switzerland
COMMI	Comércio Mineiro	Sao Paulo	Brazil
CONSH	Consolidated Holdings	London	UK

Figure 6. The grid used as the source for an Excel document.

The DynamicBackColor property of the columns displays Canadian companies in red, US in green, and UK in yellow. The resulting Excel document, shown in **Figure 7**, looks just like the grid.

	A	B	C	D
1	ID	Company	City	Country
2	ALFKI	Alfreds Futterkiste	Berlin	Germany
3	ANATR	Ana Trujillo Emparedados y helados	México D.F.	Mexico
4	ANTON	Antonio Moreno Taquería	México D.F.	Mexico
5	AROUT	Around the Horn	London	UK
6	BERGS	Berglunds snabbköp	Luleå	Sweden
7	BLAUS	Blauer See Delikatessen	Mannheim	Germany
8	BLONP	Blondesddsl père et fils	Strasbourg	France
9	BOLID	Bólido Comidas preparadas	Madrid	Spain
10	BONAP	Bon app'	Marseille	France
11	BOTTM	Bottom-Dollar Markets	Tsawassen	Canada
12	BSBEV	B's Beverages	London	UK
13	CACTU	Cactus Comidas para llevar	Buenos Aires	Argentina
14	CENTC	Centro comercial Moctezuma	México D.F.	Mexico
15	CHOPS	Chop-suey Chinese	Bern	Switzerland
16	COMMI	Comércio Mineiro	Sao Paulo	Brazil
17	CONSH	Consolidated Holdings	London	UK
18	DRACD	Drachenblut Delikatessen	Aachen	Germany
19	DUMON	Du monde entier	Nantes	France
20	EASTC	Eastern Connection	London	UK
21	ERNSH	Ernst Handel	Graz	Austria
22	FAMIA	Familia Arquibaldo	Sao Paulo	Brazil
23	FISSA	FISSA Fabrica Inter. Salchichas S.A.	Madrid	Spain
24	FOLIG	Folies gourmandes	Lille	France
25	FOLKO	Folk och fä HB	Bräcke	Sweden
26	FRANK	Frankenversand	München	Germany
27	FRANR	France restauration	Nantes	France
28	FRANS	Franchi S.p.A.	Torino	Italy
29	FURIB	Furia Bacalhau e Frutos do Mar	Lisboa	Portugal
30	GALED	Galería del gastrónomo	Barcelona	Spain

Customers

Figure 7. WorkbookXLSX can create a document that looks just like the VFP grid it's sourced from.

You can create documents programmatically by calling methods to create a workbook, set column widths, set cell values and formats, and save the resulting document. TestWorkbookXLSX2.prg is a sample program that creates the document shown in **Figure 8**.

	A	B	C	D	E	F
1	Shipping List					
2	Date: 11/01/2022 to 11/30/2022					
3	Customer Name	Order #	Carrier	Sales	Freight	Items
4	B's Beverages	11081	Federal Shipping	$1,328.00	$45.59	2
5	Bon app'	11080	United Package	$1,820.80	$64.56	3
6	Drachenblut Delikatessen	11067	United Package	$86.85	$7.98	1
7	Folies gourmandes	11090	United Package	$756.00	$1.35	2
8	Folk och fä HB	11050	United Package	$900.00	$59.41	1
9	Franchi S.p.A.	11060	United Package	$266.00	$10.98	2
10	Gourmet Lanchonetes	11049	Speedy Express	$342.00	$8.34	2
11	HILARION-Abastos	11055	United Package	$1,727.50	$120.92	4
12	HILARION-Abastos	11086	Federal Shipping	$182.40	$4.41	2
13	Hanari Carnes	11022	United Package	$1,402.00	$6.27	2
14	Hungry Owl All-Night Grocers	11063	United Package	$1,445.50	$81.73	3
15	Island Trading	11083	Federal Shipping	$230.40	$16.37	2
16	Königlich Essen	11078	Federal Shipping	$717.60	$44.12	2
17	LINO-Delicateses	11095	United Package	$1,760.00	$64.45	4
18	Pericles Comidas clásicas	11084	United Package	$1,249.10	$83.49	4
19	Princesa Isabel Vinhos	11087	United Package	$672.00	$13.02	3
20	Queen Cozinha	11097	United Package	$925.10	$71.07	3
21	Rattlesnake Canyon Grocery	11089	Federal Shipping	$10,495.60	$708.95	4
22	Ricardo Adocicados	11091	United Package	$1,472.00	$64.33	2
23	Save-a-lot Markets	11064	Speedy Express	$4,722.30	$30.09	5
24	Seven Seas Imports	11082	Speedy Express	$1,051.20	$4.20	2
25	Tortuga Restaurante	11069	United Package	$360.00	$15.67	1
26	Victuailles en stock	11088	Federal Shipping	$496.00	$4.81	1
27	White Clover Markets	11066	United Package	$928.75	$44.72	3
28	White Clover Markets	11079	Speedy Express	$1,125.50	$60.18	3
29				$36,462.60	$1,637.01	63

Shipping

Figure 8. You can create Excel documents programmatically.

Here's the code that creates the document:

```
loExcel = newobject('VFPXWorkbookXLSX', 'WorkbookXLSX\VFPXWorkbookXLSX.vcx')
lnWB    = loExcel.CreateWorkbook(lcPath)
lnSheet = loExcel.AddSheet(lnWB, 'Shipping')
```

This code sets column widths:

```
loExcel.SetColumnWidth(lnWB, lnSheet, 1, 29)
loExcel.SetColumnWidth(lnWB, lnSheet, 2,  8)
```

```
loExcel.SetColumnWidth(lnWB, lnSheet, 3, 18)
```

This code creates a custom style that used for the header rows:

```
lnStyleHeading = loExcel.CreateFormatStyle(lnWB)
loExcel.AddStyleBorders(lnWB, lnStyleHeading, BORDER_LEFT + BORDER_RIGHT + ;
   BORDER_TOP + BORDER_BOTTOM, BORDER_STYLE_THIN)
loExcel.AddStyleFont(lnWB, lnStyleHeading, 'Calibri', 14, .T., .F., ;
   rgb(255, 255, 255))
loExcel.AddStyleFill(lnWB, lnStyleHeading, rgb(0, 128, 192))
loExcel.AddStyleHorizAlignment(lnWB, lnStyleHeading, CELL_HORIZ_ALIGN_CENTER)
```

This code sets the value and style for the header row and merges the cells in that row:

```
loExcel.SetCellValue(lnWB, lnSheet, 1, 1, 'Shipping List')
loExcel.SetCellStyleRange(lnWB, lnSheet, 1, 1, 1, 6, lnStyleHeading)
loExcel.MergeCells(lnWB, lnSheet, 1, 1, 1, 6)
```

This saves the document:

```
loExcel.SaveWorkbook(lnWB)
```

As you can see, similar to Excel automation, it can be a lot of code to write since you are manually editing every cell. However, the benefit is that you have complete control over the appearance of the document.

The advantages of WorkbookXLS are:

- It doesn't require Excel to be installed.
- Because you can read an existing document, make whatever changes are necessary, and save to a different document, you can use WorkbookXLSS for template-based reporting, discussed in the next section.
- It's much faster than Excel automation, as it uses cursors to hold the document contents, writes to the XML files that comprise the document, and zips the files to create the XLSX file.
- It has methods to output from a grid, include most formatting support, or a cursor.

The disadvantages are:

- There can be lots of code to write.
- WorkbookXLSX doesn't provide full access to the Excel object model so there may be some things you can't do that you can using Excel automation.
- There are a lot of methods and properties to learn (although the documentation is quite thorough).
- You must change the code when the document being output or the template changes.

There's a lot to like about WorkbookXLSX and Greg is constantly adding new features and fixing issues. I use WorkbookXLSX in just about every application I write.

Template-based reporting

I've been doing template-based reporting for the past few years and really like this technique. The basic idea is that you create an Excel document (the "template") that contains raw data in one sheet (you can use a separate document to contain the raw data but I like having a self-contained document) while the rest of the sheets manipulate that data (PivotTables, charts, etc.). The source of the raw data is a CSV or XML file generated from VFP. All the user does to update the document with the latest data is run the VFP process to generate the file and then open and refresh the Excel document. We'll look at how you can even automate that step.

The advantages of this technique are:

- It uses a template so you don't need extensive, complex code to generate the document.
- It's fast: all you're doing in VFP is generating a CSV or XML file.
- You don't need Excel installed on the machine generating the file (obviously the user needs Excel to open the document on their machine).
- You don't necessarily have to create or maintain the template: an Excel user with moderate knowledge can do that themselves. The document can be as complex as the user wants.
- There's no change to the code generating the file when the template changes unless data changes are needed.

There are few disadvantages:

- Getting started is sort of a chicken-and-egg approach: the CSV or XML file has to exist before the Excel document but you may not know what data the user needs until the document exists.

Let's go through the steps to use this technique.

Generate the data file

As I mentioned, the source of the raw data in the Excel document can be either a CSV or XML file. I've used both but prefer an XML file because it defines the structure of the data rather than Excel having to figure it out from the content. The rest of this document assumes an XML file.

We're going to create a sales analysis spreadsheet from a slightly altered version of the Northwind sample database that comes with VFP (I add UnitCost to the OrderDetails table so we can calculate profit). The following code generates the cursor shown in **Figure 9** and outputs it to an XML document.

```
select Region.RegionDescription as Region, ;
       Orders.OrderDate, ;
       month(Orders.OrderDate) as Month, ;
       year(Orders.OrderDate) as Year, ;
       Categories.CategoryName as Category, ;
       Products.ProductName as Product, ;
       OrderDetails.Quantity, ;
       OrderDetails.UnitPrice, ;
       OrderDetails.UnitCost, ;
       OrderDetails.UnitPrice * OrderDetails.Quantity as TotalPrice, ;
       OrderDetails.UnitCost * OrderDetails.Quantity as TotalCost, ;
       Customers.Country ;
   from Orders ;
       join OrderDetails on Orders.OrderID = OrderDetails.OrderID ;
       join Products on OrderDetails.ProductID = Products.ProductID ;
       join Categories on Products.CategoryID = Categories.CategoryID ;
       join Employees on Orders.EmployeeID = Employees.EmployeeID ;
       join EmployeeTerritories ;
          on Employees.EmployeeID = EmployeeTerritories.EmployeeID ;
       join Territories ;
          on EmployeeTerritories.TerritoryID = Territories.TerritoryID ;
       join Region on Territories.RegionID = Region.RegionID ;
       join Customers on Orders.CustomerID = Customers.CustomerID ;
   into cursor RawData

cursortoxml(alias(), 'sales.xml', 1, 512 + 48)
```

Rawdata

Region	Orderdate	Month	Year	Category	Product	Quantity	Unitprice	Unitcost	Totalprice	Totalcost	Country
Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella di Giovanni	5	34.8000	28.9954	174.0000	144.9770	France
Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean Hokkien Fried Mee	10	9.8000	8.9062	98.0000	89.0620	France
Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabrales	12	14.0000	7.7672	168.0000	93.2064	France
Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella di Giovanni	5	34.8000	28.9954	174.0000	144.9770	France
Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean Hokkien Fried Mee	10	9.8000	8.9062	98.0000	89.0620	France
Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabrales	12	14.0000	7.7672	168.0000	93.2064	France
Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabrales	12	14.0000	7.7672	168.0000	93.2064	France
Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella di Giovanni	5	34.8000	28.9954	174.0000	144.9770	France
Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean Hokkien Fried Mee	10	9.8000	8.9062	98.0000	89.0620	France
Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella di Giovanni	5	34.8000	28.9954	174.0000	144.9770	France
Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean Hokkien Fried Mee	10	9.8000	8.9062	98.0000	89.0620	France
Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabrales	12	14.0000	7.7672	168.0000	93.2064	France

Figure 9. The cursor used as the source of the raw data for our spreadsheet.

The cursor is denormalized—Month, Year, TotalPrice, and TotalCost are derived values—to make reporting easier. It's also named "RawData" so that's the name used in the XML.

This process can run from a scheduled task, such as nightly, or on demand, such as when the user chooses a menu item or clicks a button in an application.

Creating the template

- Create a new Excel document.
- Choose the Data tab, click Get Data, From File, From XML (**Figure 10**), and select the XML file in the dialog that appears.

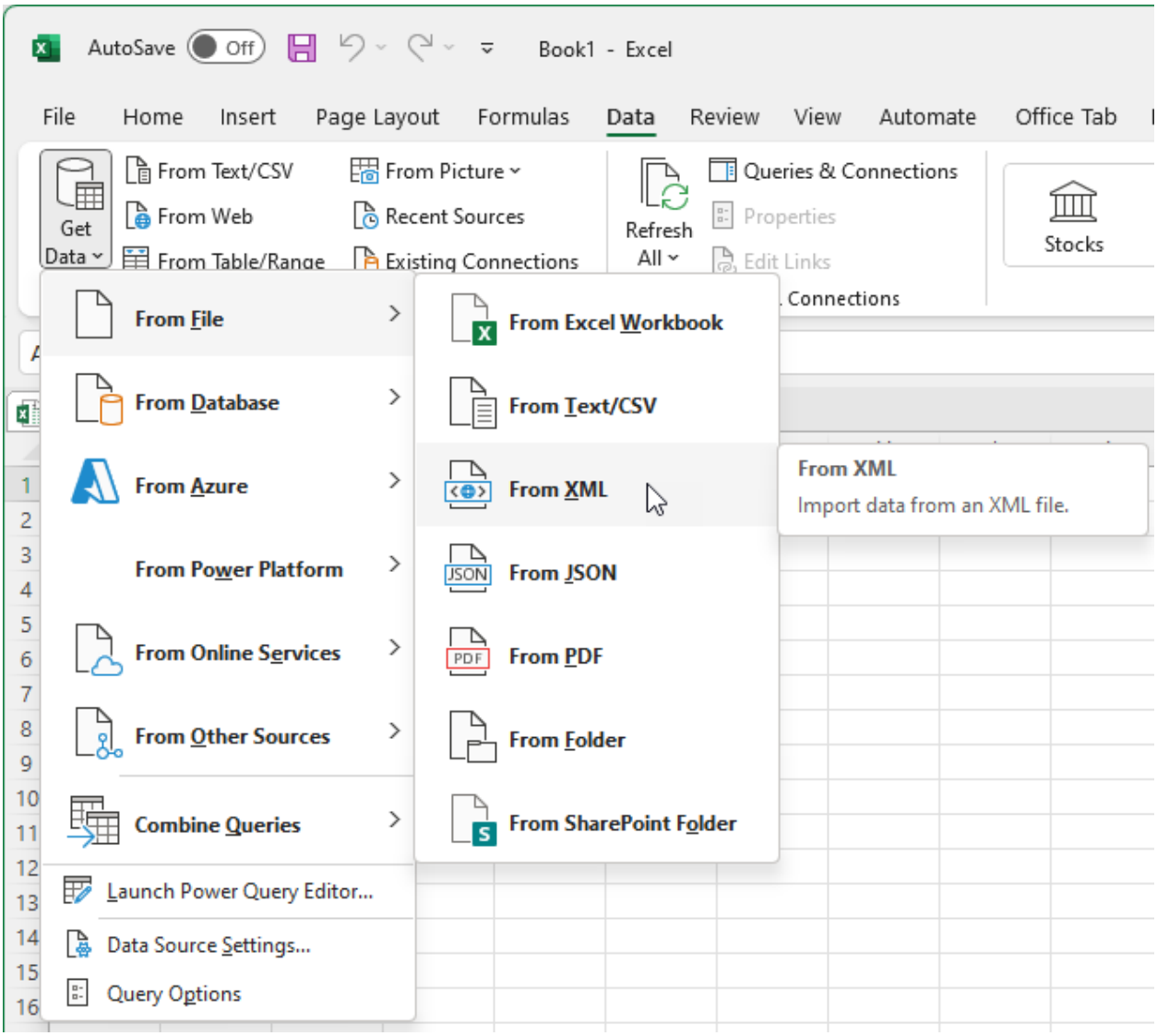
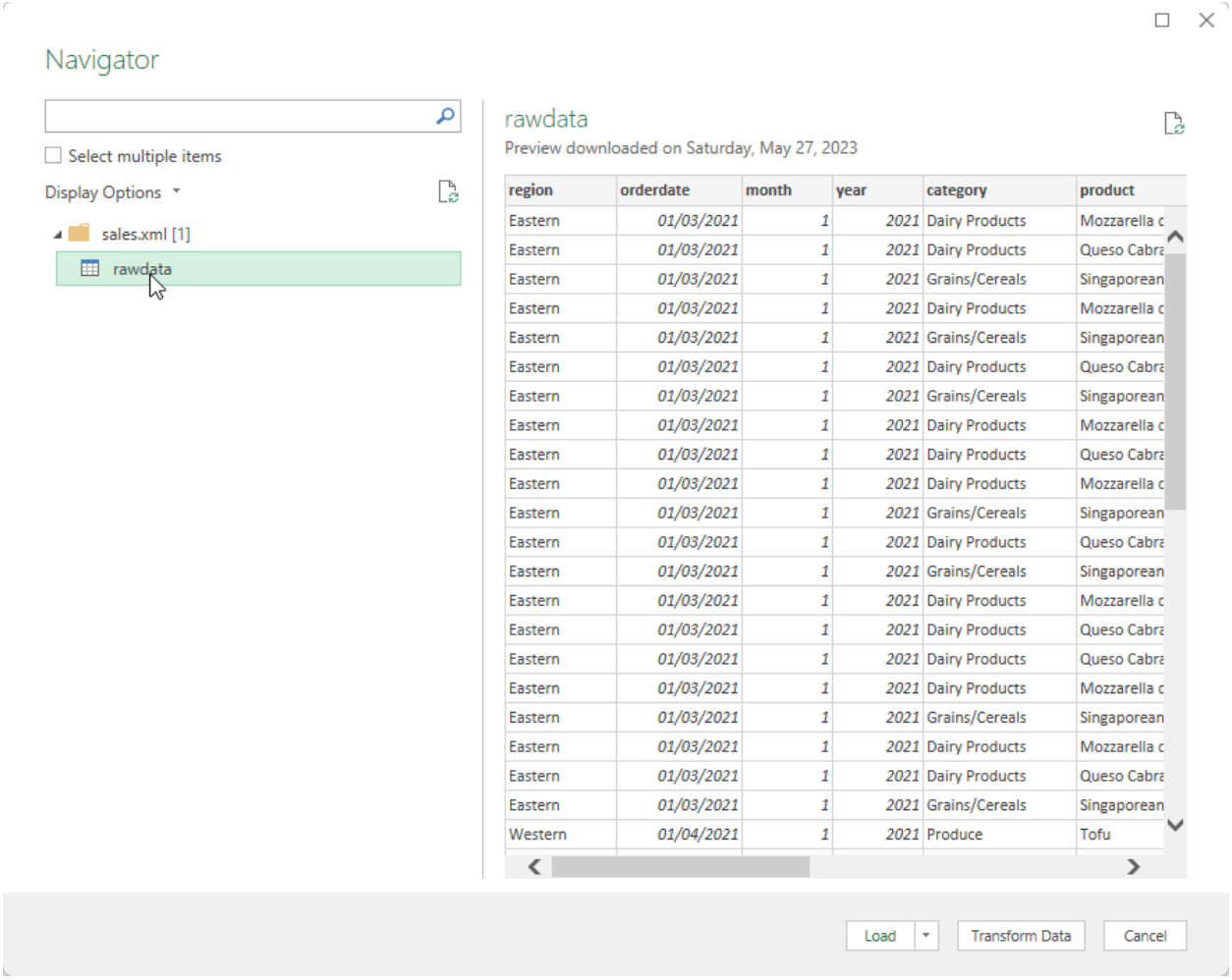


Figure 10. Getting data from the XML file into Excel.

- Select "rawdata" in the Navigator dialog and click Load (**Figure 11**).



region	orderdate	month	year	category	product
Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella c
Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabra
Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean
Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella c
Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean
Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabra
Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean
Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella c
Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabra
Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella c
Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean
Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabra
Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean
Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella c
Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabra
Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabra
Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella c
Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean
Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella c
Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabra
Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean
Western	01/04/2021	1	2021	Produce	Tofu

Figure 11. Previewing and loading the XML data.

The result is a new sheet named "rawdata" containing the VFP data in a table named "rawdata" (click the down arrow in the Name Box to see the name or see the Table Name entry in the Table Design tab). See **Figure 12**.

To avoid confusion between the rawdata sheet and the rawdata table, let's rename the table to "table_rawdata" by updating Table Name in the Table Design tab.

	A	B	C	D	E	F	G	H	I	J	K
1	region	orderdate	month	year	category	product	quantity	unitprice	unitcost	totalprice	totalcost
2	Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella di Giovanni	5	34.8	28.9954	174	144.977
3	Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabrales	12	14	7.7672	168	93.2064
4	Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean Hokkien Fried Mee	10	9.8	8.9062	98	89.062
5	Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella di Giovanni	5	34.8	28.9954	174	144.977
6	Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean Hokkien Fried Mee	10	9.8	8.9062	98	89.062
7	Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabrales	12	14	7.7672	168	93.2064
8	Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean Hokkien Fried Mee	10	9.8	8.9062	98	89.062
9	Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella di Giovanni	5	34.8	28.9954	174	144.977
10	Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabrales	12	14	7.7672	168	93.2064
11	Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella di Giovanni	5	34.8	28.9954	174	144.977
12	Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean Hokkien Fried Mee	10	9.8	8.9062	98	89.062
13	Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabrales	12	14	7.7672	168	93.2064
14	Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean Hokkien Fried Mee	10	9.8	8.9062	98	89.062
15	Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella di Giovanni	5	34.8	28.9954	174	144.977
16	Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabrales	12	14	7.7672	168	93.2064
17	Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabrales	12	14	7.7672	168	93.2064
18	Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella di Giovanni	5	34.8	28.9954	174	144.977
19	Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean Hokkien Fried Mee	10	9.8	8.9062	98	89.062
20	Eastern	01/03/2021	1	2021	Dairy Products	Mozzarella di Giovanni	5	34.8	28.9954	174	144.977
21	Eastern	01/03/2021	1	2021	Dairy Products	Queso Cabrales	12	14	7.7672	168	93.2064
22	Eastern	01/03/2021	1	2021	Grains/Cereals	Singaporean Hokkien Fried Mee	10	9.8	8.9062	98	89.062
23	Western	01/04/2021	1	2021	Produce	Tofu	9	18.6	4.5291	167.4	40.7619
24	Western	01/04/2021	1	2021	Produce	Manjimup Dried Apples	40	42.4	32.5886	1696	1303.544
25	Western	01/04/2021	1	2021	Produce	Tofu	9	18.6	4.5291	167.4	40.7619
26	Western	01/04/2021	1	2021	Produce	Manjimup Dried Apples	40	42.4	32.5886	1696	1303.544
27	Western	01/04/2021	1	2021	Produce	Manjimup Dried Apples	40	42.4	32.5886	1696	1303.544
28	Western	01/04/2021	1	2021	Produce	Tofu	9	18.6	4.5291	167.4	40.7619
29	Western	01/04/2021	1	2021	Produce	Manjimup Dried Apples	40	42.4	32.5886	1696	1303.544
30	Western	01/04/2021	1	2021	Produce	Tofu	9	18.6	4.5291	167.4	40.7619

rawdata Sheet1 +

Figure 12. The VFP data is loaded into an Excel table.

Note that when I say "table," I don't mean some rows and columns; I mean an Excel Table object. Tables are underused but have a lot of advantages:

- They can be referenced by name rather than range. For example, this table can be specified as "Table_rawdata" rather than rawdata!$A$1:$K$10130.
- Columns within a table can also be specified by name. For example, "=Table_rawdata[region]" refers to column A in the table without hard-coding the column name. This is especially useful if the structure of the table changes, such as new columns inserted between existing ones. Specifying a table or table[column] name is called a "structured reference."
- They auto-expand as data is added. For example, if more columns or rows are added, you'd have to edit everything referencing it as a range but not change anything if it's referenced as a table.
- Formulas can apply to a single cell in a table or an entire column. For example, entering a formula of "=Table_rawdata[@totalcost] *0.1" in cell L2 calculates 10% of cell K2. Entering "=Table_rawdata[totalcost] *0.1" instead fills column L with 10% of the values in column K.
- You can attach a Slicer to a table to provide easy filtering. I'll discuss Slicers in more detail later.

- You can format a table using one of the built-in styles. You can also have banded rows and/or columns (every second row and/or column is a different color), add a total row, and bold the first or last column.

You can convert a range into a Table by clicking Format as Table in the Home tab or pressing Ctrl+T.

You can rename the table if you wish: select any cell in the table, select the Table Design tab, and enter the desired name in the Table Name textbox.

See https://support.microsoft.com/en-us/office/using-structured-references-with-excel-tables-f5ed2452-2337-4f71-bed3-c8ae6d2b276e for more information on structured references, such as naming rules.

Presenting the data

Now let's present the data in a more attractive manner than just a table of raw numbers.

- Drag the tab for Sheet1 so it appears to the left of rawdata (not required but I like to have rawdata as the last sheet) and rename it to Sales (right-click the tab and choose Rename).
- Select any cell in the table on the rawdata sheet.
- From the Insert tab, choose Pivot Table, From Table/Range. In the dialog that appears (**Figure 13**), Table/Range is already filled in with "Table_rawdata". Select "Existing Worksheet," click the Sales sheet, and click in cell A1, then choose OK.

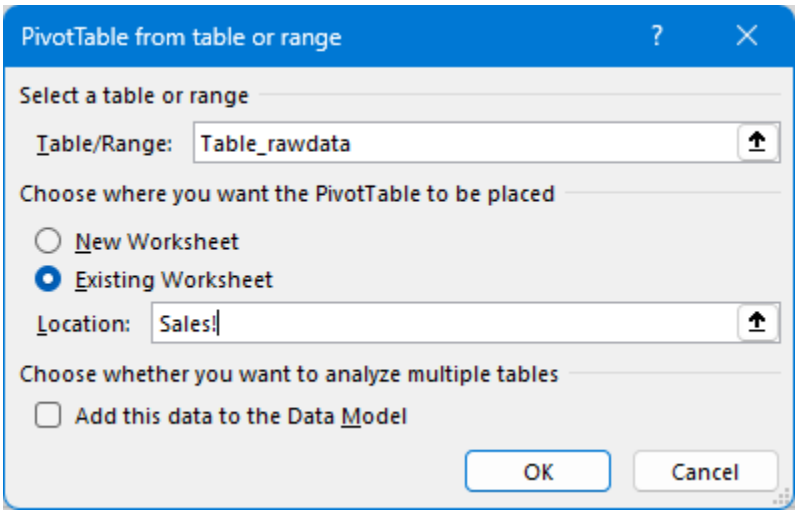


Figure 13. Insert a PivotTable from the table.

- In the PivotTable Fields panel, check "region," "year," and "totalprice." Drag "Sum of year" from the Values section in the panel to Columns (**Figure 14**).

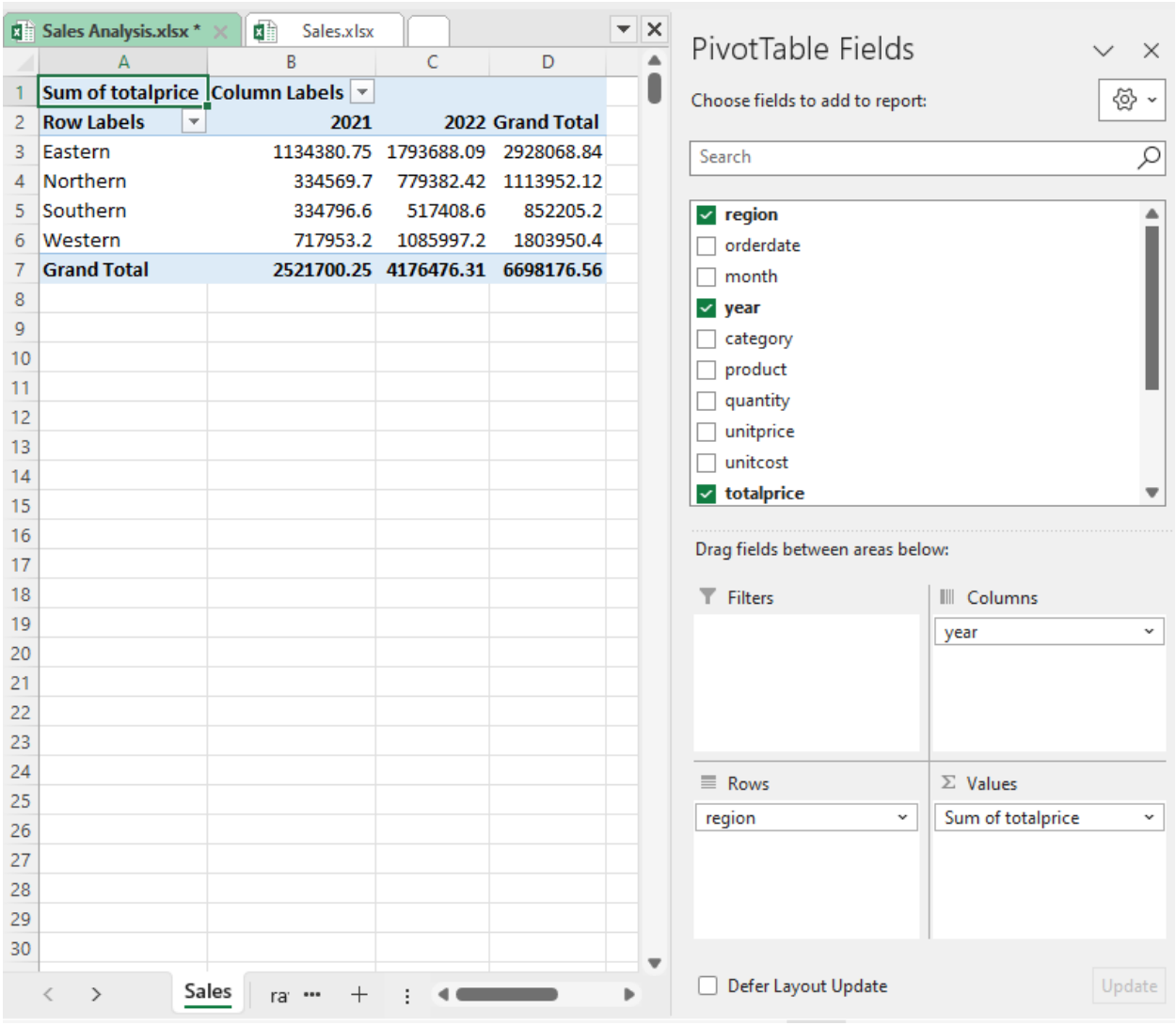


Figure 14. Set up the PivotTable with the desired row, column, and value fields.

- Click "Sum of totalprice" in the Values section and choose Value Field Settings. Enter "Sales" for Custom Name, click the Number Format button, and select Currency.
- Select cell B1 and enter "Year."
- Select cell A2 and enter "Region."
- The PivotTable should look like **Figure 15**.

	A	B	C	D
1	Sales	Year		
2	Region	2021	2022	Grand Total
3	Eastern	$1,134,380.75	$1,793,688.09	$2,928,068.84
4	Northern	$334,569.70	$779,382.42	$1,113,952.12
5	Southern	$334,796.60	$517,408.60	$852,205.20
6	Western	$717,953.20	$1,085,997.20	$1,803,950.40
7	Grand Total	$2,521,700.25	$4,176,476.31	$6,698,176.56
8				

Figure 15. We created a PivotTable quickly and easily.

Repeat these steps to create a PivotTable with category in the rows rather than region.

Now let's add a Slicer. A Slicer is a control that applies filtering to another object.

- Click any cell in the first PivotTable and choose Slicer from the Insert tab. Check category and click OK. Drag the Slicer object so it's beside the PivotTable.
- Repeat for the second PivotTable but choose region for the field.

Clicking a category or a region in the Slicers filters the associated PivotTable to only show sales for the selected filter (**Figure 16**).

You can connect a Slicer to more than one PivotTable or PivotChart. Right-click the Slicer, choose Report Connections, and check the desired PivotTables and PivotCharts in the list.

You can format the Slicer using the options on the Slicer tab of the ribbon when the Slicer is selected.

	A	B	C	D
1	**Sales**	**Year**		
2	**Region**	**2021**	**2022**	**Grand Total**
3	Eastern	$88,182.35	$149,701.55	$237,883.90
4	Northern	$60,822.80	$78,717.30	$139,540.10
5	Southern	$24,626.80	$31,809.80	$56,436.60
6	Western	$46,676.00	$74,662.00	$121,338.00
7	**Grand Total**	**$220,307.95**	**$334,890.65**	**$555,198.60**
8				
9	**Sales**	**Year**		
10	**Category**	**2021**	**2022**	**Grand Total**
11	Beverages	$227,528.10	$417,491.25	$645,019.35
12	Condiments	$88,182.35	$149,701.55	$237,883.90
13	Confections	$164,440.50	$175,177.76	$339,618.26
14	Dairy Products	$251,419.70	$281,864.40	$533,284.10
15	Grains/Cereals	$66,799.40	$136,736.00	$203,535.40
16	Meat/Poultry	$147,719.40	$301,410.77	$449,130.17
17	Produce	$100,215.65	$116,026.80	$216,242.45
18	Seafood	$88,075.65	$215,279.56	$303,355.21
19	**Grand Total**	**$1,134,380.75**	**$1,793,688.09**	**$2,928,068.84**
20				
21				
22				

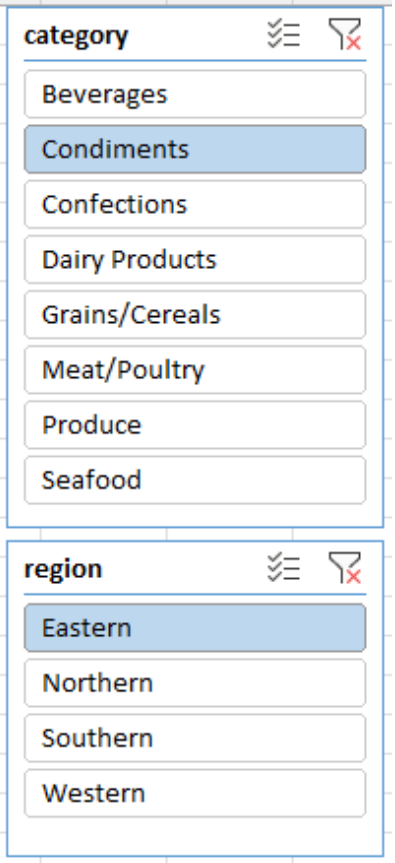


Figure 16. Slicers filter the data shown in PivotTables and other objects.

Let's add a Timeline. A Timeline is like a Slicer except it filters on dates. Click any cell in the first PivotTable and choose Timeline from the Insert tab. Check orderdate and click OK. Drag the Timeline object to a convenient location.

You can choose Days, Months, Quarters, or Years for the filter type and then drag the slider to show which values you want. See **Figure 17** for an example.

	A	B	C	D	E	F
1	Sales	Region	Year			
2		Eastern	Northern	Southern	Western	Grand Total
3	Category					
4	Beverages	$3,995.00	$425.20	$72.00	$2,678.00	$7,170.20
5	Condiments	$1,155.30	$396.10		$448.00	$1,999.40
6	Confections	$1,656.60	$229.30	$434.00	$65.00	$2,384.90
7	Dairy Products	$2,262.00	$485.60	$287.20	$1,655.00	$4,689.80
8	Grains/Cereals	$688.40	$265.60	$166.40	$267.00	$1,387.40
9	Meat/Poultry	$1,356.70	$938.10	$195.60	$657.00	$3,147.40
10	Produce	$1,083.00		$276.00	$828.00	$2,187.00
11	Seafood	$1,261.90	$200.00	$219.20	$96.00	$1,777.10
12	**Grand Total**	**$13,458.90**	**$2,939.90**	**$1,650.40**	**$6,694.00**	**$24,743.20**

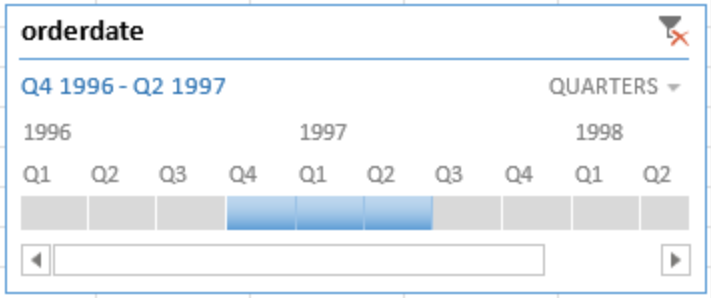
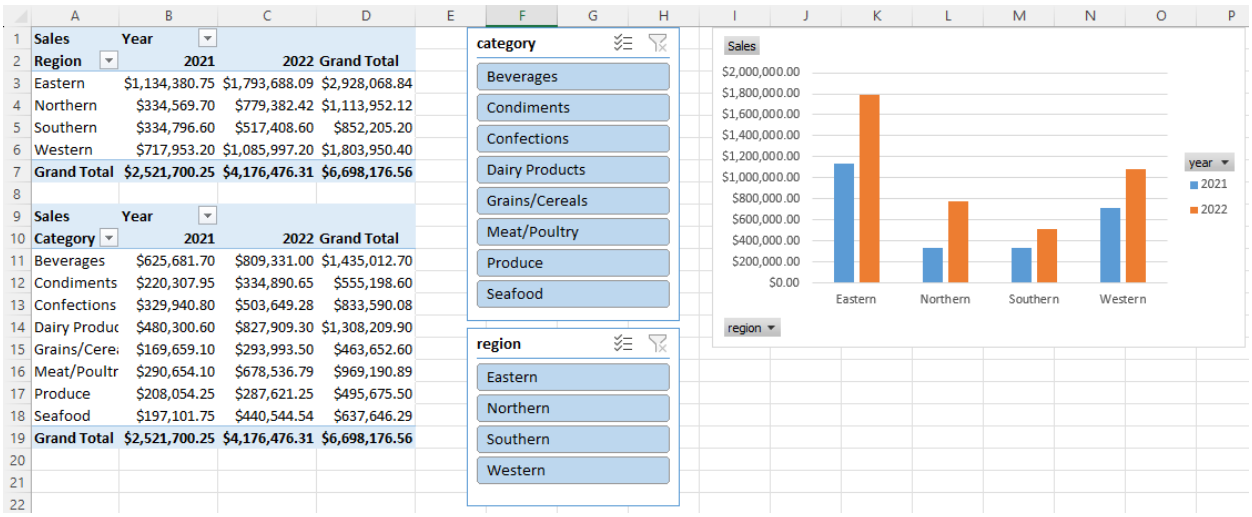


Figure 17. Timelines provide a visual tool to filter on dates.

As with Slicers, you can connect a Timeline to more than one PivotTable or PivotChart and format it as desired.

Rows of numbers often make people's eyes glaze over, so let's add a chart to visualize the data.

- Click any cell in the first PivotTable and from the PivotTable Analyze tab, choose PivotChart. Select Column and click OK.
- Move the chart beside the Slicer for the first PivotTable. The result is shown in **Figure 18**.



Sales	Year		
Region	2021	2022	Grand Total
Eastern	$1,134,380.75	$1,793,688.09	$2,928,068.84
Northern	$334,569.70	$779,382.42	$1,113,952.12
Southern	$334,796.60	$517,408.60	$852,205.20
Western	$717,953.20	$1,085,997.20	$1,803,950.40
Grand Total	$2,521,700.25	$4,176,476.31	$6,698,176.56

Sales	Year		
Category	2021	2022	Grand Total
Beverages	$625,681.70	$809,331.00	$1,435,012.70
Condiments	$220,307.95	$334,890.65	$555,198.60
Confections	$329,940.80	$503,649.28	$833,590.08
Dairy Produc	$480,300.60	$827,909.30	$1,308,209.90
Grains/Cere	$169,659.10	$293,993.50	$463,652.60
Meat/Poultr	$290,654.10	$678,536.79	$969,190.89
Produce	$208,054.25	$287,621.25	$495,675.50
Seafood	$197,101.75	$440,544.54	$637,646.29
Grand Total	$2,521,700.25	$4,176,476.31	$6,698,176.56

Figure 18. A PivotChart makes it easier to visualize the data.

Custom presentation

Some users find PivotTables intimidating, so let's create a sheet where the user can see daily sales broken down by region and day in a simpler manner.

- Create a new sheet: click the + in the tab strip at the bottom and rename it to Daily.
- Enter "Region" in A1, "Starting Date" in A2, "Date" in A4, and "Weekday" in A5.
- We want to create a selector for region similar to a Slicer. To do that, we need two things: the unique values of the region column and a data validation. In cell M1 of the rawdata sheet, enter this formula: =SORT(UNIQUE(Table_rawdata[region])). The UNIQUE function fills cells with unique values from a range and the SORT function sorts them. The result is that cells M1 through M4 contain "Eastern," "Northern," "Southern," and "Western." Then select cell B1 on the Daily sheet, click the Data tab, click Data Validation, and in the dialog that appears, set Allow to List and Source to "=rawdata!$M$1:$M$4. The resulting drop-down list is shown in **Figure 19**.

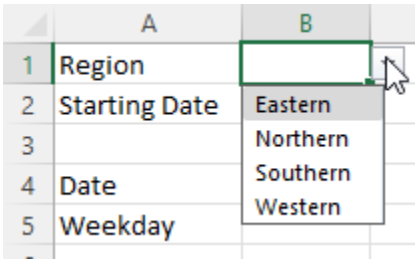


Figure 19. Drop-down lists are easy to create in Excel.

- Cell B2 is the starting date to use for daily sales, so enter a date such as "12/01/2021."

- In B4, put "=B2" so it contains the starting date. In C4, put "=B4+1" so it's the next day, then select that cell through H4 and press Ctrl+R to fill right so we have a week of dates.
- In B5, put "=WEEKDAY(B4)," then select the Home tab, Format, Format Cells, Custom, and enter "dddd" to format the cell as the day name. Select B5 through H5 and press Ctrl+R.
- In A7, type "Revenue." In A8, type "=UNIQUE(Table_rawdata[category])" to fill the next set of cells with the unique category names from the data table. In A16, type "Total Revenue."
- In B8, type "=SUMIFS(Table_rawdata[totalprice], Table_rawdata[region], $B$1, Table_rawdata[orderdate], B$4, Table_rawdata[category], $A8)." The syntax for the SUMIFS function is sum_range, criteria_range_1, criteria_range_2, etc. In this case, we're summing the totalprice column in Table_rawdata for those rows where the region column contains the value in B1, the orderdate column contains the value in row 4 of the current column (note the placement of "$," which indicates an absolute reference), and the category column contains the value in column A of the current row.
- Select cells B8 through H15 and press Ctrl+R and Ctrl+D to fill right and down.
- In B16, put "=SUM(B8:B15),", then select B16 through H16 and press Ctrl+R.
- To test that the formula works, select a region from the drop-down list in B1. Notice the cells fill with the sum of the sales for the specified category and date (**Figure 20**).

	A	B	C	D	E	F	G	H
1	Region	Eastern						
2	Starting Date	12/01/2021						
3								
4	Date	12/01/2021	12/02/2021	12/03/2021	12/04/2021	12/05/2021	12/06/2021	12/07/2021
5	Weekday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday
6								
7	Revenue							
8	Dairy Products	0	0	5846.4	0	0	0	0
9	Grains/Cereals	0	0	0	420	0	0	0
10	Produce	0	0	0	2120	0	18550	0
11	Condiments	0	0	0	0	0	1361.5	0
12	Seafood	0	0	0	90	0	0	0
13	Confections	0	0	0	475	0	0	0
14	Beverages	0	0	0	0	0	0	0
15	Meat/Poultry	0	0	0	1180.8	0	0	0
16	Total Revenue							

Figure 20. The SUMIFS formula sums a range using various selection criteria.

- Repeat the previous five steps for A18 to A27 but use “Cost” instead of “Revenue” and “totalcost” instead of “totalprice” in the formula.
- Using similar steps, create sections for profit using a formula of “=B8-B19” and profit margin using a formula of “=IFERROR(B30/B8,0)” (this displays a 0 for division by zero errors).
- Format the sheet, using borders, fonts, and numeric cell formats so it looks like **Figure 21**.

Region	Eastern						
Starting Date	12/01/2021						
Date	**12/01/2021**	**12/02/2021**	**12/03/2021**	**12/04/2021**	**12/05/2021**	**12/06/2021**	**12/07/2021**
Weekday	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**	**Monday**	**Tuesday**
Revenue							
Dairy Products	$0.00	$0.00	$5,846.40	$0.00	$0.00	$0.00	$0.00
Grains/Cereals	$0.00	$0.00	$0.00	$420.00	$0.00	$0.00	$0.00
Produce	$0.00	$0.00	$0.00	$2,120.00	$0.00	$18,550.00	$0.00
Condiments	$0.00	$0.00	$0.00	$0.00	$0.00	$1,361.50	$0.00
Seafood	$0.00	$0.00	$0.00	$90.00	$0.00	$0.00	$0.00
Confections	$0.00	$0.00	$0.00	$475.00	$0.00	$0.00	$0.00
Beverages	$0.00	$0.00	$0.00	$0.00	$0.00	$0.00	$0.00
Meat/Poultry	$0.00	$0.00	$0.00	$1,180.80	$0.00	$0.00	$0.00
Total Revenue	$0.00	$0.00	$5,846.40	$4,285.80	$0.00	$19,911.50	$0.00
Cost							
Dairy Products	$0.00	$0.00	$3,289.78	$0.00	$0.00	$0.00	$0.00
Grains/Cereals	$0.00	$0.00	$0.00	$240.95	$0.00	$0.00	$0.00
Produce	$0.00	$0.00	$0.00	$1,054.06	$0.00	$6,002.78	$0.00
Condiments	$0.00	$0.00	$0.00	$0.00	$0.00	$463.59	$0.00
Seafood	$0.00	$0.00	$0.00	$83.85	$0.00	$0.00	$0.00
Confections	$0.00	$0.00	$0.00	$150.72	$0.00	$0.00	$0.00
Beverages	$0.00	$0.00	$0.00	$0.00	$0.00	$0.00	$0.00
Meat/Poultry	$0.00	$0.00	$0.00	$183.85	$0.00	$0.00	$0.00
Total Cost	$0.00	$0.00	$3,289.78	$1,713.44	$0.00	$6,466.37	$0.00
Profit							
Dairy Products	$0.00	$0.00	$2,556.62	$0.00	$0.00	$0.00	$0.00
Grains/Cereals	$0.00	$0.00	$0.00	$179.05	$0.00	$0.00	$0.00
Produce	$0.00	$0.00	$0.00	$1,065.94	$0.00	$12,547.22	$0.00
Condiments	$0.00	$0.00	$0.00	$0.00	$0.00	$897.91	$0.00
Seafood	$0.00	$0.00	$0.00	$6.15	$0.00	$0.00	$0.00
Confections	$0.00	$0.00	$0.00	$324.28	$0.00	$0.00	$0.00
Beverages	$0.00	$0.00	$0.00	$0.00	$0.00	$0.00	$0.00
Meat/Poultry	$0.00	$0.00	$0.00	$996.95	$0.00	$0.00	$0.00
Total Profit	$0.00	$0.00	$2,556.62	$2,572.36	$0.00	$13,445.13	$0.00
Profit %							
Dairy Products	0.0%	0.0%	43.7%	0.0%	0.0%	0.0%	0.0%
Grains/Cereals	0.0%	0.0%	0.0%	42.6%	0.0%	0.0%	0.0%
Produce	0.0%	0.0%	0.0%	50.3%	0.0%	67.6%	0.0%
Condiments	0.0%	0.0%	0.0%	0.0%	0.0%	66.0%	0.0%
Seafood	0.0%	0.0%	0.0%	6.8%	0.0%	0.0%	0.0%
Confections	0.0%	0.0%	0.0%	68.3%	0.0%	0.0%	0.0%
Beverages	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Meat/Poultry	0.0%	0.0%	0.0%	84.4%	0.0%	0.0%	0.0%
Profit %	0.0%	0.0%	43.7%	60.0%	0.0%	67.5%	0.0%

Figure 21. The final sheet.

Conditional formatting

Create a sheet for month and year using similar steps for the Daily sheet. We'll also create drop-down lists for month and year using "=UNIQUE(TEXT(Table_rawdata[orderdate], "mmmm"))" and "=UNIQUE(Table_rawdata[year])" to create Month and Year ranges in the rawdata sheet. The formula for the data is slightly more complicated because we need to take the selected month (cell B2) and year (B2) into account:

=SUMIFS(Table_rawdata[totalprice], Table_rawdata[region], $B$1, Table_rawdata[month], B$5, Table_rawdata[year], B$7, Table_rawdata[category], $A10)

Let's add conditional formatting to the profit margin. Click in B43, select the Home tab, click Conditional Formatting, New Rule. In the New Formatting Rule dialog, choose Icon Sets for Format Style, choose the third icon set (flags) for Icon Style, then set the rest of the settings as shown in **Figure 22** (Icon Style will automatically change to "Custom").

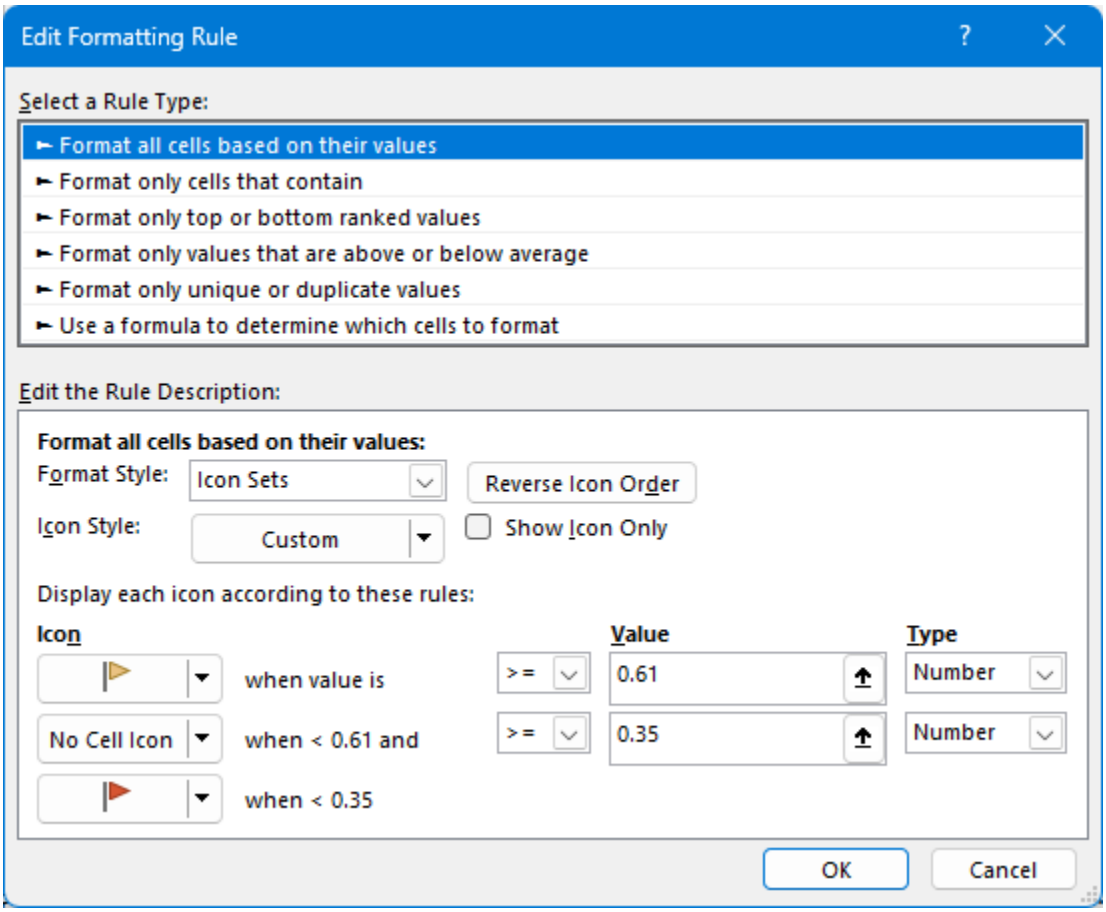


Figure 22. Conditional formatting rules allow you to flag cells containing values outside desired ranges.

When the profit margin is greater than 60%, we'll display a yellow flag. When it's between 35 and 60%, we'll display no symbol. When it's less than 35%, we'll display a red flag. See **Figure 23**.

42	**Profit %**												
43	Dairy Products	45.1%	50.5%	67.0%	59.7%	40.9%	52.2%	89.0%	47.2%	37.1%	40.0%	51.1%	47.0%
44	Grains/Cereals	63.9%	41.9%	64.8%	31.5%	53.3%	25.4%	22.2%	78.1%	58.4%	31.3%	57.4%	51.4%
45	Produce	42.7%	30.8%	23.3%	62.9%	96.7%	45.6%	31.7%	25.0%	67.2%	78.1%	40.8%	35.6%
46	Condiments	51.3%	39.6%	34.1%	36.4%	54.2%	30.9%	42.0%	5.8%	39.1%	36.3%	57.2%	35.1%
47	Seafood	27.7%	35.9%	50.6%	46.2%	69.1%	37.8%	68.9%	48.4%	8.7%	44.5%	28.9%	52.5%
48	Confections	50.7%	51.9%	26.7%	69.2%	62.3%	42.6%	81.9%	58.7%	32.3%	38.2%	46.7%	49.6%
49	Beverages	81.5%	47.9%	71.7%	38.1%	27.4%	50.7%	52.1%	51.6%	37.1%	39.4%	34.9%	52.6%
50	Meat/Poultry	41.7%	35.8%	42.9%	18.1%	69.4%	26.1%	38.0%	17.1%	63.9%	44.0%	48.5%	31.4%
51	**Profit %**	51.1%	43.8%	58.3%	42.6%	56.3%	38.7%	64.8%	43.5%	47.7%	42.0%	40.9%	46.4%

Figure 23. The conditional formatting flags profit margins that are too high or too low.

Mapping

Excel supports mapping geographical data. Let's create a map showing sales by country.

- Create a new worksheet and rename it to Map.
- Type "Country" in U1 and "Sales" in V1.
- Put this formula into U2: "=SORT(UNIQUE(Table_rawdata[country]))"
- Put this formula into V2: "=SUMIFS(Table_rawdata[totalprice], Table_rawdata[country], $U2)". Fill this formula down to the bottom of the countries list in column U.
- Select U1 through V22 and select Maps, Filled Maps from the Insert tab.
- Drag the resulting map to the upper left corner of the sheet and enlarge it.
- Enter "Sales by Country" for the map title.

The result is shown in **Figure 24**.

Country	Sales
Argentina	44230.9
Austria	811852.1
Belgium	207502.6
Brazil	542185.9
Canada	309158
Denmark	128950.7
Finland	105954.2
France	391896.3
Germany	1324156
Ireland	344694.8
Italy	90970.4
Mexico	150961.3
Norway	27550.85
Poland	12679
Portugal	54053.7
Spain	90272.12
Sweden	329208.1
Switzerland	205419
UK	343076.7
USA	1350175
Venezuela	333606.9

Figure 24. You can create a map in Excel with just a few mouse clicks.

If you want to get even fancier, you can create a 3-D map and make a video from it (**Figure 25**).

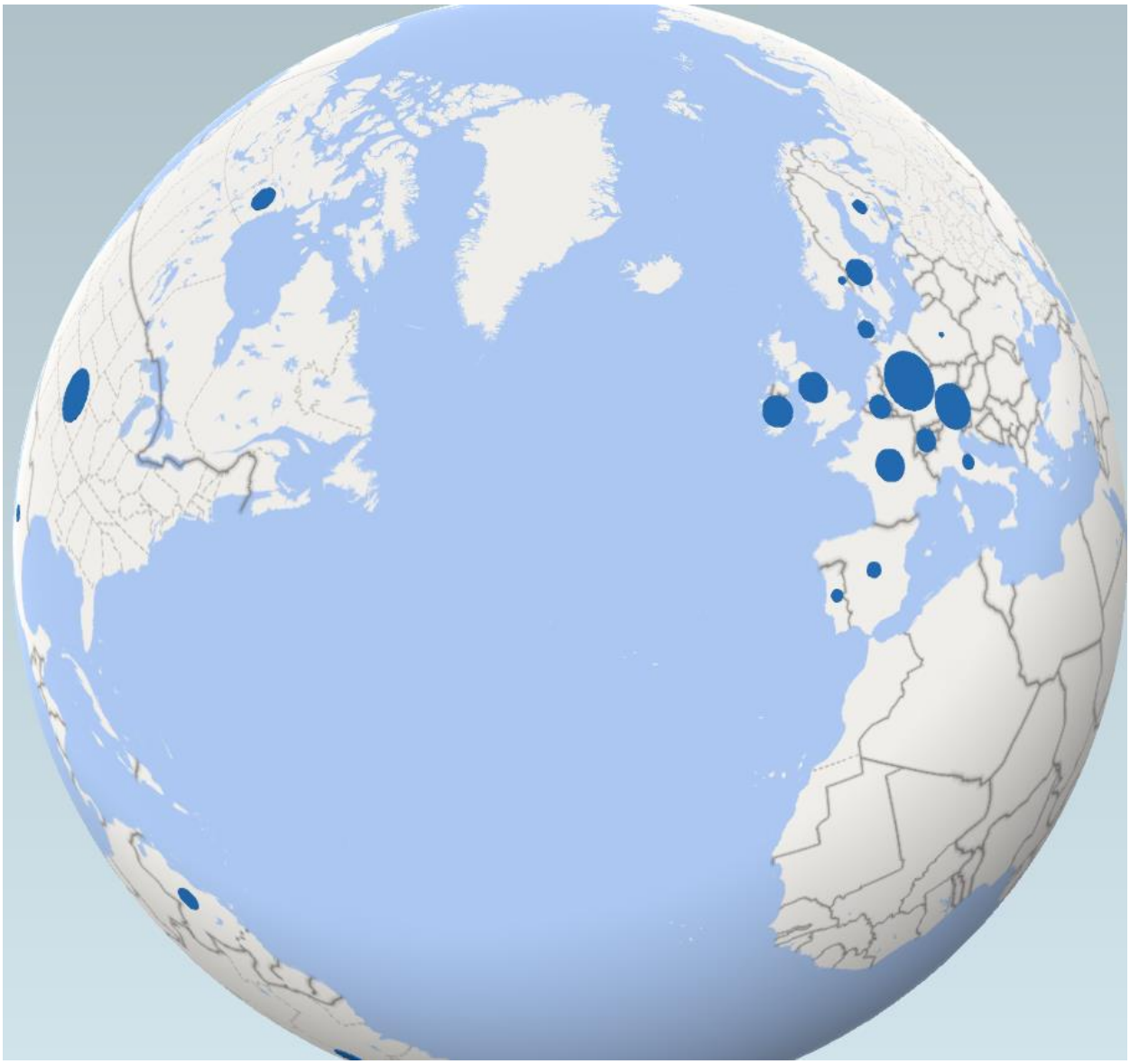


Figure 25. You can even create a 3-D map.

Making the path for the data relative

If you send the spreadsheet and XML file to someone else, they'll likely get an error when they open the spreadsheet or try to refresh it. The reason is that the path for the XML file is absolute and it's unlikely they put the files into the same drive and folder as you did. Let's make the spreadsheet use a relative path for the XML file:

- On the rawdata sheet, put this into an empty cell: "=LEFT(CELL("filename", $A$1), FIND("[",CELL("filename", $A$1), 1) - 1)." This formula returns the folder the spreadsheet is in and we'll use that same folder for the XML file.
- Name that cell "FilePath" by clicking the Name Box and entering "FilePath."

- On the Data tab, choose Queries and Connections, then double-click "rawdata" in the Queries and Connection panel to open Power Query Editor.
- Click Advanced Editor in the Home tab (**Figure 26**).
- Add this after the first line ("let") to create a variable named Path containing the content of the FilePath cell:

  ```
  Path = Excel.CurrentWorkbook(){[Name="FilePath"]}[Content]{0}[Column1],
  ```

- Change the Source statement to use the Path variable rather than a hard-coded path:

  ```
  Source = Xml.Tables(File.Contents(Path & "sales.xml")),
  ```

- Click Close & Load.

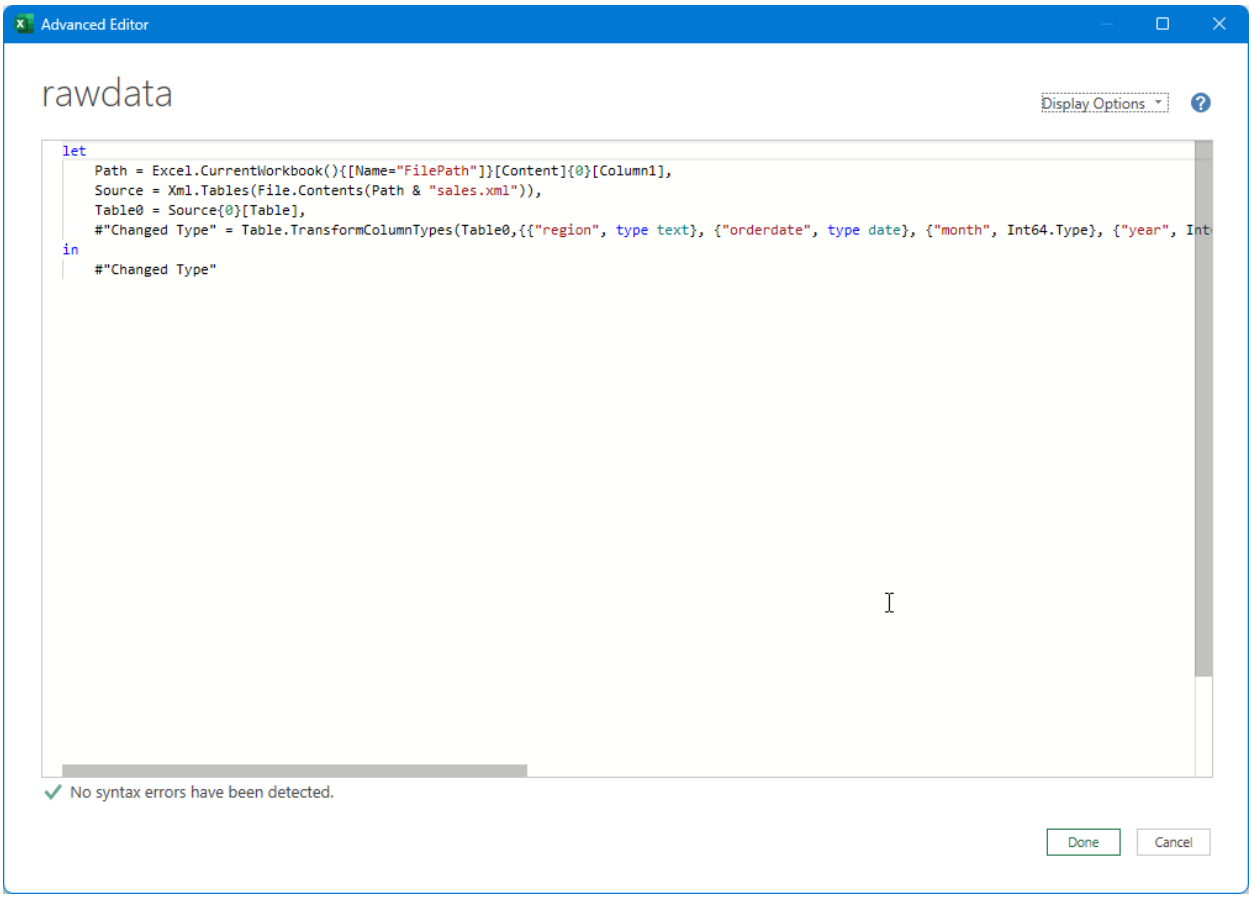


Figure 26. The Power Query Advanced Editor.

(Thanks to https://excel.tv/how-to-create-a-relative-file-path-in-power-query for this.)

Auto-refreshing

After generating the XML file, the user has to manually open the Excel document and refresh the data. You can automate that using the following code (substitute "Sales.xlsx" with the appropriate name):

```
try
   loExcel    = createobject('Excel.Application')
   loDocument = loExcel.Workbooks.Open(fullpath('Sales.xlsx'))
   loExcel.Sheets('rawdata').Select()
   loExcel.Range('A1').Select()
   loExcel.Selection.ListObject.QueryTable.Refresh(.F.)
   loExcel.DisplayAlerts = .F.
   loDocument.Save()
   loExcel.Visible = .T.
catch
   if vartype(loExcel) = 'O'
      loExcel.Quit()
   endif vartype(loExcel) = 'O'
endtry
```

Other useful Excel features

Here are some other Excel features useful for reporting purposes.

XLOOKUP

XLOOKUP is a new function that can be a replacement for the older VLOOKUP and HLOOKUP functions. XLOOKUP finds a value in an array (table or range) and returns an array for the match. In VFP terms, it's like doing a SEEK or LOCATE in a table to find a record and then returning one or more values from that record.

DATEDIF

DATEDIF determines the number of days, months, or years between two dates.

WORKDAY

WORKDAY returns a date the specified number of workdays (excluding weekends) before or after the specified date. You can optionally specify an array of holidays to exclude.

Sparklines

A sparkline (**Figure 27**) is like a miniature graph in a single cell. Select a range, choose the type of sparkline you want from the Insert tab, and specify the cell the sparkline should go in.

	A	B	C	D	E	F	G
1	Sales	Region	Year				
2		Eastern	Northern	Southern	Western	Grand Total	
3	Category						
4	Beverages	$5,375.60	$716.80	$334.40	$3,022.00	$9,448.80	
5	Condiments	$1,482.80	$776.10	$308.80	$773.50	$3,341.20	
6	Confections	$3,312.40	$914.30	$527.20	$755.50	$5,509.40	
7	Dairy Products	$3,760.00	$1,080.80	$785.60	$2,815.00	$8,441.40	
8	Grains/Cereals	$1,085.80	$508.80	$340.80	$533.00	$2,468.40	
9	Meat/Poultry	$2,348.80	$2,250.90	$300.40	$657.00	$5,557.10	
10	Produce	$1,885.00	$152.00	$276.00	$1,345.00	$3,658.00	
11	Seafood	$1,820.90	$914.90	$377.20	$381.50	$3,494.50	
12	**Grand Total**	**$21,071.30**	**$7,314.60**	**$3,250.40**	**$10,282.50**	**$41,918.80**	

Figure 27. Sparklines are a miniature graph in a single cell.

Custom lists

If you have to enter a series of values, such as day or month names, regions, divisions, product names, and so on, use the custom list feature in Excel. There are already custom lists for day and month names (full and abbreviated), but you can also create your own.

To use a custom list, enter the first value in a cell. Click the small square in the bottom right corner of the selection rectangle surrounding the cell and drag down or right. As you do, a tooltip will show you the value to be inserted into the current cell (**Figure 28**). Release the mouse button to fill the cells.

Figure 28. Custom lists make entry of a series of pre-defined values easy.

To create your own custom list, choose File, Options, Advanced, and in the General section, click the Edit Custom Lists button.

Summary

Most modern applications should provide a way to export to Excel. As we've seen in this document, it's easy to do. Plus, rather than just creating a boring list of data, with a little effort, you can create attractive and easy-to-use Excel documents the users are happy to work with. Using a template-based approach, you can even get them to do most of the work!

Biography

Doug Hennig is a partner with Stonefield Software Inc. He is the author of the award-winning Stonefield Query; the award-winning Stonefield Database Toolkit (SDT) (now open source); the MemberData Editor, Anchor Editor, and CursorAdapter and DataEnvironment builders that come with Microsoft Visual FoxPro; and the My namespace and updated Upsizing Wizard in Sedna. He also created several VFPX projects, including Project Explorer, OOP Menu, OOP Reports, and SFMail.

Doug is co-author of *VFPX: Open Source Treasure for the VFP Developer*, *Making Sense of Sedna and SP2*, *Visual FoxPro Best Practices For The Next Ten Years*, the *What's New in Visual FoxPro* series, and *Hacker's Guide to Visual FoxPro 7.0* (now open source). He was the technical editor of *Hacker's Guide to Visual FoxPro 6.0* and *The Fundamentals*. Doug wrote hundreds of articles in 20 years for *FoxRockX*, FoxTalk, FoxPro Advisor, Advisor Guide to Visual FoxPro, and CoDe magazines.

Doug spoke at every Microsoft FoxPro Developers Conference (DevCon) starting in 1997 and at user groups and developer conferences all over the world. He is one of the organizers of the Southwest Fox and Virtual Fox Fest conferences. He is one of the administrators for the VFPX VFP community extensions Web site. He was a Microsoft Most Valuable Professional (MVP) from 1996 through 2011. Doug was awarded the 2006 FoxPro Community Lifetime Achievement Award.


Hacker's Guide™ to Visual FoxPro® 7.0
Hentzenwerke Publishing
What's New in Visual FoxPro 7.0
Hentzenwerke Publishing
What's New in Visual FoxPro® 8.0
Tamar E. Granor & Doug Hennig
Hentzenwerke Publishing
What's New in Nine: Visual FoxPro's Latest Hits
Visual FoxPro Best Practices For the Next Ten Years
Making Sense of Sedna and SP2
dFPUG c/o ISYS GmbH
VFPX
Open Source Treasure for the VFP Developer
dFPUG c/o ISYS GmbH
stonefield Database TOOLKIT
stonefield Query
MVP
Microsoft® Most Valuable Professional
VFPx
FoxRockX