Conrad Carlberg

BUSINESS ANALYSIS:

Microsoft® Excel® 2010

ANSWER KEY BUSINESS QUESTIONS
CONTROL COMPANY FINANCES
FORECAST SALES
PREPARE BUSINESS CASES
MAKE BETTER INVESTMENT DECISIONS
IMPROVE QUALITY

que®

Business Analysis: Microsoft® Excel 2010

International Standard Book Number-10: 0-7897-4317-5
International Standard Book Number-13: 978-0-7897-4317-6

Library of Congress Cataloging-in-Publication Data

Carlberg, Conrad George
Business analysis, Microsoft Excel 2010 / Conrad Carlberg.
p. cm.
ISBN 978-0-7897-4317-6
1. Corporations—Finance. 2. Corporations—Finance—Computer programs. 3. Business planning—Computer programs. 4. Microsoft Excel (Computer file) I. Title.
HG4026.C254 2010
005.54—dc22
2010017029

Printed in the United States of America

First Printing: June 2010

07 06 05 04 4 3 2 1

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Que Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark. Intuit and QuickBooks are registered trademarks of Intuit, Inc."

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Bulk Sales

Que Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside of the United States, please contact

International Sales 1-317-428-3341
international@pearsontechgroup.com

Associate Editor
Greg Wiegand

Acquisitions Editor
Loretta Yates

Development Editor
Sondra Scott

Technical Editor
P.K.Hari Hara Subramanian

Managing Editor
Sandra Schroeder

Project Editor
Mandie Frank

Copy Editor
Gill Editorial Services

Indexer
Lisa Stumpf

Production
Mark Shirar

Designer
Anne Jones

INTRODUCTION

IN THIS INTRODUCTION

The book you just opened is different from most others on Excel that you might have seen. That's because it focuses on a topic that is deeply important to us all: money.

The novelist Rex Stout once wrote, facetiously, "The science of accounting has two main branches, first addition, and second subtraction." I kept that in mind when I was casting about for the book's theme. I wanted to write a book that would show people how to maximize profit, the result of combining those two branches.

Profit, of course, is not revenue. I can't teach you how to create revenue—that's more a matter for the heart, not the head—nor would I want to offer you MBA or CPA material. I did set out to write a book that any person engaged in any level of business could use as a refresher, from basic financial documents such as general ledgers and income statements, to the operational methods such as statistical process control, to the procedures such as business case analysis that underlie investment decisions.

I also wanted to structure this book around the most popular and sophisticated numeric analysis program available: Microsoft Excel. Therefore, each chapter in *Business Analysis: Microsoft® Excel 2010* provides information about a different business task or procedure and discusses how best to apply Excel in that situation.

This book references many Excel functions and capabilities that you might already use in your daily business activities. You might also find discussions of tools that you have never used or that you might never have considered using in the context of business analysis.

After all, no one can be completely familiar with every option in an application as extensive as Excel. Several Internet newsgroups frequented by Excel

users answer technical questions. Years ago, a user asked how to enter a number in a worksheet cell so that Excel would treat the number as text. (This is quite a basic operation.) Surprisingly, the question was posted by one of the most experienced, best-known, and creative Excel consultants in the country. I thought that it was a put-on and responded appropriately, but it turned out that the question was genuine.

So we all have gaps in our knowledge. The purpose here is to help fill in some of the gaps that might have entered your knowledge base since your last course in business or since you first learned how to use a worksheet.

Business Analysis: Microsoft® Excel 2010 uses case studies—that is, situations that are typical of decisions or problems that you might face on any given workday. These case studies first discuss the problem itself: why it's a problem and how a solution can contribute to a company's profitability. Then the case studies demonstrate at least one possible solution that uses Excel as a tool. The intent is for you to mentally put yourself in the situation described, work through it, and then apply or adapt the solution to an actual situation that you face.

Taking It on Faith

Since the second edition of this book was published, the financial markets have sustained some severe shocks. Enron, which had been ranked seventh in the Fortune 500, entered bankruptcy, its CEO entered prison, and $60 billion in stock and $2 billion in pensions vanished. An old and highly respected accounting firm, Arthur Andersen, was found guilty of obstruction of justice in the Enron case. Although the Supreme Court later overturned the finding, "Uncle Artie's" staff fell from 28,000 to 200. And, by no means last, WorldCom revealed that it had improperly booked $3.8 billion in expenses—then, one month later, it filed for bankruptcy. Other familiar names that gained unwanted notoriety: Global Crossing, Tyco, and Adelphia.

The basis for all this corporate malfeasance was the cooking of the books. Transactions were kept off the financial reports, and earnings and losses were misstated, in efforts to paint a pretty picture and pump up the stock price.

After publication of this book's third edition, the United States suffered the worst and longest financial slowdown since the Depression of the 1930s. The recession of 2008–2009 had many causes, but the catalysts were pretty clear. Mortgage lenders extended home loans to people who couldn't afford them—at least, not when home prices stopped rising or variable rate loans adjusted up. Investment banks started to package and sell derivative instruments such as collateralized debt obligations and credit default swaps. The former hid bad mortgages among the good. The latter constituted a form of unregulated insurance that did not have sufficient reserve funds to survive a market bottom.

To some degree, the system works on trust. As investors, creditors, customers, and employees, all of us rely on financial reports such as income statements to make decisions about our lives. We rely on independent assessments of the risks assumed by investment firms

such as Madoff Investment Securities and funding agencies such as Fannie Mae to guide our investment decisions. When the dollar amounts that are used to calculate those figures are seriously misrepresented, we can make seriously bad decisions.

Nothing here or anywhere else can fully protect you from people who keep bad news off the books. You have to be as close to things as Sherron Watkins, the Enron vice president, to see what's really going on in time to phone the cops. Even the Securities and Exchange Commission somehow overlooked the improbability of the 72 consecutive months of profitable returns reported by Madoff's company. So it might seem pointless to pay attention to income statements, balance sheets, and other reports of a company's financial status. But it's not pointless.

In the third edition of this book, I wrote this:

> The vast majority of North American businesses are generally honest, and if they sometimes skate, it's not by a really indecent margin. If you want to adopt a cynical viewpoint, consider that the incentives to misrecord financials are all wrong for small and midsize businesses. It's in large businesses where the temptations are really huge, and—at least, since 2002—that's where the scrutiny is greatest.

I was wrong, obviously. Greater scrutiny, earlier on, might have at least mitigated the reckless behaviors that took down Bear-Stearns, Merrill-Lynch, Lehman Brothers, IndyMac Bank, and so on. None of these could be considered "small" or "midsize" businesses, but they escaped the light of day that might have reined in their excesses until it was too late.

Nevertheless, it's still not "...pointless to pay attention to income statements, balance sheets, and other reports of a company's financial status." Plenty of analysts paid attention to those reports, directed their own assets accordingly, and blew whistles.

Alas, no one seems to have been listening.

Renamed and Improved Functions in Excel 2010

Business Analysis with Microsoft Excel has never been intended as a version book—that is, it has never been issued as a new edition every time Microsoft issues a new version of Excel. You are reading the book's fourth edition, yet Excel has had seven separate releases (not counting service releases) since this book was first published.

The third edition took the Excel 2007 user interface, most recognizable by the Ribbon, into account in describing how to carry out various analyses in Excel. The current edition does the same, but it describes your actions first using the Ribbon, and then describes them assuming that you're using the menu structure from Excel 1997–2003.

The most recent version, Excel 2010, has a variety of functions with new names, which are mainly existing functions whose capabilities have been divided among new functions. Most are statistical functions pertaining to the unit normal distribution, the t distribution, the F distribution and so on.

For example, the existing `TDIST` function returns a one- or a two-tailed test, depending on how the user sets one of its arguments. Which tail of the distribution can be specified by subtraction, because the distribution is symmetric.

The new functions divide the capabilities among three different functions: `T.DIST` (one-tail, left), `T.DIST.RT` (one-tail, right), and `T.DIST.2T` (two-tail). The capabilities were always there, but Excel 2010 assigns them specifically instead of relying on arguments and arithmetic manipulation such as `1 - TDIST(2.07, 15, 2)`.

This book takes account of the new functions and uses them where appropriate. There are three issues to bear in mind, discussed next.

Compatibility

Excel 2010 continues to recognize the functions that are being replaced; in fact, they are referred to as "compatibility functions." This means that Excel 2010 will recognize, say, the `TDIST` function in a workbook that you create in an earlier version such as Excel 2000.

It also means that you can continue to use the existing compatibility functions in new workbooks that you create using Excel 2010. Assuming that Microsoft intends to sunset the compatibility functions eventually, you have a window during which you can gradually switch to the new function set.

Excel 2010 has an auto-complete feature for functions. You can see it in action in Figure I.1.

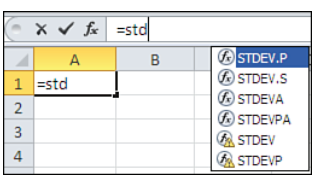


Figure I.1
The compatibility functions appear at the bottom of the list box.

Two compatibility functions are `STDEV` and `STDEVP`. When you type an equal sign and the first few letters in the function name, a list of functions that start with those letters appears. Click one of them to auto-complete the function, up to and including the opening parenthesis.

Notice in Figure I.1 that the compatibility functions appear at the bottom of the list. This is a reminder to the user that Microsoft is beginning to deprecate their use, and you should instead try to use `STDEV.P` or `STDEV.S`.

Consistency

Consistency functions is the term that, at least during the beta period for Excel 2010, Microsoft is using for the new functions. The rationale is as follows.

Prior to 2010, some functions such as `STDEV` had two versions. For example, `STDEV` is intended for use when the numbers you give it to work with are a sample from a population

of values. In contrast, STDEVP is intended for use when the numbers you provide it *are* the population.

Microsoft regards the final *P* in the function name STDEVP to be inconsistent; after all, there's no final *S* in STDEV, even though it's intended to be used on a sample of data.

So, to make the names consistent, Excel 2010 has two "consistency functions" that calculate standard deviations: STDEV.S and STDEV.P. The consistency arises because both functions use the same naming pattern: the *name* of the function (STDEV) followed by a period and a *descriptor* (S or P) that identifies how you're expected to use it: on a Sample or on a Population.

This new naming scheme applies principally to statistical functions, and to the reference distribution functions in particular. (The distinction between the cumulative distribution function (CDF) and the probability density function (PDF) continues to be handled by the function arguments.)

The descriptors that Microsoft will use in the new names for old functions are as follows:

- DIST—This descriptor is attached to a function that returns a particular distribution such as the normal curve. It calls for the area under the curve to the *left* of the user-supplied value (CDF); alternatively, it can return the height of the curve at that point (PDF).
- DIST.RT—This descriptor calls for the area under a curve to the *right* of a user-supplied value.
- DIST.2T—This descriptor returns the area under the curve to the right of one value and to the left of the complementary value (often its negative). That is, it returns the probability associated with a two-tailed test.
- INV—This descriptor returns what Excel terms the *inverse* of the distribution. Students of elementary-to-intermediate statistics will probably think of this as the criterion value for a statistical test: for example, the value that Student's t-statistic must exceed to be regarded as statistically significant at some level.
- S—As discussed earlier, this descriptor returns a statistic for a sample of observations.
- P—Also as noted earlier, this descriptor returns a statistic for a population of observations.

Of course, the changes to Excel's function set go well beyond the calculation of standard deviations. There are changes to other statistical functions, a few of which include these:

- VAR is now VAR.S
- VARP is now VAR.P
- BETADIST is now BETA.DIST
- BETAINV is now BETA.INV

How This Book Is Organized

You can look in the table of contents or the index of *Business Analysis with Microsoft Excel, Fourth Edition* whenever you encounter an unfamiliar or obscure situation and read about how to solve it with the analysis tools in Excel. To make it easier to find related situations, this book is divided into four parts:

- **Part I, "Analyzing Financial Statements"—**This part discusses fundamental financial concepts and tools such as income statements, balance sheets, cash flow, and ratio analysis.
- **Part II, "Financial Planning and Control"—**This part covers budgeting methods such as pro formas, forecasting trends, and quality-control procedures, including process measurement and defect analysis.
- **Part III, "Investment Decisions"—**You'll find business case analysis and profit planning in this part. It covers strategies for structuring and testing business cases, as well as ways to quantify and manage the degree of risk involved in entering a new line of business. Also included is a chapter on fixed assets, which normally account for the greatest portion of a company's capital investment.
- **Part IV, "Sales and Marketing"—**Sales and marketing analysis, costing and pricing, and margin analysis are covered here. Since the publication of the original edition of this book, many businesses have placed their financial and operational records in true relational databases and in applications intended specifically for accounting. Therefore, this edition includes a chapter that explains the most effective ways to import data into Excel directly from databases, from accounting report files and from websites. Another chapter focuses on moving the data the opposite direction, from Excel into a database.

There's also a glossary that briefly defines important terms.

As I mentioned earlier, it's important that you be able to dip into this book to find particular topics and to use the information without necessarily reviewing everything that came before. Therefore, certain tips and recommendations on using Excel are (briefly) repeated from time to time. And in each chapter you will find full, step-by-step descriptions of how to accomplish a given task using Excel.

Two Special Skills: Named Ranges and Array Formulas

Have you ever had to interpret someone else's worksheet? Or have you ever had to use a worksheet that you constructed months or perhaps years ago, and then been completely unable to figure out what you had in mind when you constructed it? You probably have and, if so, you know what a headache it can be.

The principal difficulty with many otherwise useful worksheets is that their authors don't document them. Consider this worksheet formula:

```
=IF(AND(B12<30000,A12<5),C14*D14*.05,C14*D14*.075)
```

It could take a couple of minutes to figure out what that formula is up to, even if you know the worksheet's basic purpose. However, it would take you only a few seconds if the author had used this formula instead:

```
=IF(AND(YearToDateSales<30000,Tenure<5),Units*Price*LowCommission,
Units*Price*HighCommission)
```

It's not too difficult to infer what this formula says:

> If this person's sales during this year are less than $30,000, and this person was hired fewer than five years ago, return the sales amount times the lower commission; otherwise, return the sales amount times the higher commission.

To help make your work self-documenting, in many instances, you should give names to Excel worksheet cells, ranges, and constants. You can then use those names in your formulas and functions so that you can see it's multiplying sales dollars by a commission, not simply one relatively anonymous cell by another. Because you'll find this approach taken throughout the book, it's reviewed here.

Assigning Names

To name a *cell* or *range*, begin by selecting it on the worksheet. Click the Ribbon's Formulas tab and click the Define Name drop-down in the Defined Names group. Choose Define Name in the drop-down list and type the name you want to use in the Name box. You can also specify the name's scope as sheet-level or book-level (see Chapter 2, "Balance Sheet: Current Assets," for more information on that distinction.) If you're using a version of Excel that precedes Excel 2007, choose Insert, Name, Define, and type the name you want to use in the Names in Workbook edit box; then click OK.

Or use this quicker method: After you have selected the cell or range, click in the Name box (immediately above the column header for column A and left of the drop-down arrow), type the name, and press Enter.

To name a *constant* such as `LowCommission`, click the Define Name dropdown (or choose Insert, Name, Define), and type the name of the constant in the Names in Workbook edit box. Then, in the Refers To edit box, type the value that you want to assign to the constant and click OK. (You can't use the Name box to define a constant directly.)

A side benefit of using names instead of cell or range addresses is that you can paste names into formulas as you are creating them. After you have started typing a formula, click the Ribbon's Formulas tab and select the Use in Formula drop-down from the Defined Names group. Then click the name you want in the drop-down list. (In earlier versions, you can choose Insert, Name, Paste and select the name you want to use from the Paste Name list box.) This approach saves you keystrokes and helps prevent misspellings. Also, you don't have to recall existing names: They're right there in the list box.

When you choose a name for a range or a constant, consider using both uppercase and lowercase letters: for example, `TotalLiabilities`. Mixing uppercase and lowercase makes the name easier to read. (Compare to `totalliabilities`.) You should probably avoid using all uppercase letters. Excel's worksheet function names (for example, `SUM` and `AVERAGE`) use all uppercase letters, and you don't want to define a name that could be confused with a function.

Blank spaces and certain special characters, such as the percent symbol, aren't allowed in names. Some people like to use an underscore in place of a space, preferring `Total_Liabilities` over `TotalLiabilities`.

Using Array Formulas

Many of the formulas described in this book are a special type of Excel formula called an *array formula*. An array formula contains an array of values or a reference to an array of worksheet cells, as shown here:

```
=SUM(IF(MOD(ROW(SheetRange),2)=0,SheetRange))
```

This sums the values in the worksheet range named `SheetRange` if they are in an even-numbered row. The formula requires a special keyboard sequence to enter it correctly. On a computer running Windows, the sequence is Ctrl+Shift+Enter—that is, simultaneously hold down the Ctrl and Shift keys as you press Enter.

You can tell that Excel has interpreted your formula as an array formula if you see curly braces (sometimes termed *French braces*) around it in the formula bar. For example, the formula shown previously appears in the formula bar like this:

```
{=SUM(IF(MOD(ROW(SheetRange),2)=0,SheetRange))}
```

Don't type the braces yourself. If you do, Excel interprets the formula as text.

These are termed array formulas because they have within them arrays that you don't usually see and that Excel doesn't normally expect. For example, if expanded, the previous formula would show an array of the row numbers in `SheetRange`. Excel doesn't normally expect that you'll present an *array* of values as conditions in an `IF` function (Excel normally expects a single value as an IF condition), so you signal that's what you've done by using the Ctrl+Shift+Enter keyboard sequence.

You can explore the inner workings of array formulas by using the Evaluate Formula tool in Excel 2002 or later.. Click the Ribbon's Formulas tab and choose Evaluate Formula in the Formula Auditing group. In earlier versions, begin by choosing Tools, Formula Auditing, Evaluate Formula.

Conventions Used in This Book

Business Analysis with Microsoft Excel, Fourth Edition uses a few typeface, terminology, and formatting conventions to emphasize special information:

- A sequence like this:

  Ctrl+Enter

  means that you should hold down the Ctrl key as you press Enter.
- When you should select a sequence of options from an Excel Ribbon group, you will see this:

  Click the Data tab, then choose What-If Analysis, Goal Seek.

  This means that you should first click the Ribbon's Data tab, then the What-If Analysis dropdown, and then click on Goal Seek in the dropdown menu. The sequences for Excel 2003 and earlier, with no Ribbon, are analogous and are usually specified in the text.
- Data or formulas that you enter in an Excel worksheet cell are shown like this:

  `=SUM(CumulativeNetIncome)/ProductLife`
- New terms, or information that needs special emphasis, are shown in *italic*.
- Information about performing a task more efficiently or alternative ways to go about a task appear in tips. Tips are set apart from the main text like this:

  TIP To copy the selected cells, press Ctrl+C.

- Information that is related to the current topic but that might not apply to it directly is shown like this:

  NOTE There is one distinct IRR for each change in sign in a series of cash flows.

Summarizing Transactions: From the Journals to the Balance Sheet

4

IN THIS CHAPTER

Chapter 3, "Valuing Inventories for the Balance Sheet," covered the topic of inventory valuation in detail. The various methods discussed there are needed to fill in the current assets section of the balance sheet. This chapter focuses on recording transactions in journals, cataloging transactions in ledgers, and summarizing the information in the balance sheet.

Following is the basic structure of a balance sheet:

- The Assets section consists of the company's current assets (typically including cash, accounts receivable, inventory, and prepaid expenses) and its fixed assets. This section also contains any other assets that do not fit neatly within the current and fixed classifications.
- The Liabilities and Owner's Equity section consists of the company's current and long-term liabilities; it also includes the owner's equity as described next. Typically, these include accounts payable, short- and tong-term notes payable, and a few other types of liabilities that vary according to a company's line of business.
- The difference between the company's assets and its liabilities is the *equity*—that portion of the company's worth that belongs to its owner or owners.

The first three chapters of this book introduced some fundamental concepts, such as accounts, revenues, assets, debits, and credits. They also discussed some of the functional relationships among these concepts.

This chapter introduces ways that you can use Excel to establish the structural relationships among accounts, journals, and ledgers. It describes how to manage the flow of information about revenue, expenses, and profit by using Excel workbooks, worksheets, and Visual Basic for Applications (VBA) code.

Why This Chapter Is Here

I dithered for some time before deciding to include this chapter in the current edition of this book. Since the time that I first wrote it, in the mid-1990s, accounting software for personal computers has become much more popular in the marketplace. The user interfaces have evolved from arcane and puzzling to smooth and friendly. The cost of the accounting software has plummeted, and in some cases the software is offered by publishers as a loss leader for more profitable products.

I had to ask myself, in that environment for business computing, if it made sense to retain this chapter. It talks about special journals and subsidiary ledgers and how transactions move through them to appear, in the aggregate, on income statements and balance sheets.

Unless you're someone who wants to use Excel to maintain the accounting records of a small business, you have little need for that sort of information. For example, special journals are tools that make *manual* accounting easier. They help categorize transactions into useful groupings that enable the accounting process to roll up the debits and credits into the right ledgers and then into the proper accounts on the balance sheet and income statement.

But specialized accounting software has no need of structures such as special journals. Computers are intrinsically stupid, but they're also fast and accurate. So accounting software for personal computers tends to dispense with manual tools such as special journals and even deemphasizes the three traditional ledger divisions (general, accounts receivable, and accounts payable). The results still exist, of course, but the process of moving from the individual transactions to the financial summaries has been streamlined to take advantage of the computer's speed and accuracy.

So why would you care about what this chapter discusses? I spent quite some time thinking about how I could use my allotted page count talking about topics that are more interesting than the role of subsidiary ledgers in the management of your business.

I finally decided to keep this chapter, albeit with some major changes from previous editions. I had three basic reasons for that decision:

- People still talk about journals and ledgers, and the general journal certainly hasn't gone away.
- Perhaps you are building your own accounting application using Excel. I did—and I'd hate to think that I'm alone in the Excel universe.
- The chapter's topic provides an ideal context for introducing concepts and tools concerning VBA—tools that you can learn to apply in thousands of other situations.

If none of those reasons grabs you, for goodness sake move on. I think you'll like Chapter 5, "Working Capital and Cash Flow Analysis."

Understanding Journals

The basic flow of information for business transactions follows this sequence of events:

1. A business transaction occurs—usually, a sale or a purchase.
2. Information about the transaction is recorded in a *journal*. The journal usually records the information about the transactions in chronological order. For example, one record might contain data on a sale that took place on March 1, the second record might describe a purchase that was made on March 2, the third record might have data on a payment that was made on March 3, and so on.
3. Information about the transactions is copied (or *posted*) from the journal to a *ledger*. This ledger has a different section for each account, such as accounts receivable or notes payable. Within each section, information is usually recorded chronologically. The main difference between the journal and the ledger is that the ledger categorizes the information from the journal into specific accounts, whereas the journal records the information chronologically—transactions involving several different accounts would be adjacent if they occurred on the same date.
4. Information in the ledger is summarized to obtain a total for each account at the end of an accounting period. These totals are used to prepare financial statements such as the income statement and the balance sheet.

See Figure 4.1 for an example of a general journal.

Why qualify the term *journal* with the word *general*? Because keeping only one journal and one ledger gets cumbersome. Notice in Figure 4.1 that only three transactions are shown in the general journal. These transactions pertain to three relatively infrequent events: the return of some merchandise from a customer, the return of some inventory to a supplier, and the purchase of office equipment. All the remaining transactions during June are kept together in special journals.

Figure 4.1
Use the general journal as a catchall for transactions that don't belong in special journals.

	A	B	C	D	E	F
1	Date	Explanation	Account	✓	Debit	Credit
2						
3	6/8/2011	Received books back from Judith Jones	Sales Returns		$ 42	
4			Accounts Receivable			$ 42
5						
6	6/17/2011	Returned books to Lenney Distributing for credit	Accounts Payable		$ 1,525	
7			Purchase Returns			$ 1,525
8						
9	6/21/2011	Purchased PC on account	Office Equipment		$ 1,863	
10			Accounts payable			$ 1,863
11						

General Journal

Ready 100%

If you had only one journal, the task of posting information from the journal to the ledger could become too time-consuming. Furthermore, using only one journal makes finding information about a specific sale or payment more tedious, even if you use Excel's lookup functions. However, computer applications whose sole purpose is accounting largely do away with this problem, and with special journals as well.

Understanding Special Journals

Companies sometimes use *special journals*, which are places to record information about particular types of transactions. The most frequently occurring transactions tend to be sales to customers and payments to creditors. Also, many companies do business with their customers and suppliers on both a cash basis and a credit basis. These frequent transactions create the need for four special journals—two for payments and two for receipts, two for cash and two for credit:

- A *cash receipts journal* contains information about payments that you receive from customers. These payments could take the form of currency, such as when a customer hands you $20 to purchase an item, or a check, such as when you receive payment for an earlier credit purchase. It might also contain information about other cash receipts, such as capital investments in the company.
- A *cash payments journal* contains information about payments that you make to creditors and suppliers. Normally, these payments are checks that you write, but they could also be payments made using currency. This journal also contains information about operating expenses that you pay in cash, such as salaries or office rent.
- A *sales journal* contains information about credit sales that you make. Together with sales information in the cash receipts journal, the sales journal accounts for all sales that your business makes.
- A *purchases journal* contains information about credit purchases that you make from your suppliers. Together with cash purchase information in the cash payments journal, the purchases journal accounts for all purchases that your business makes from suppliers.

Various other types of transactions exist that aren't recorded in these special journals—they are recorded in the general journal, which acts as a catchall for miscellaneous transactions.

Your own business might have a category of transactions that occur frequently but do not fit well into these four special journals. If you're going to use special journals at all, you should choose the ones that make sense for your line of business; any structure that has special journals for the most frequently occurring types of transactions will do.

For example, suppose that you run a car rental agency. You probably don't purchase cars from suppliers very often, but you might regularly send your cars to garages and body shops for maintenance. In that case, you might use a special repairs and maintenance journal instead of a special purchases journal.

Structuring the Special Sales Journal

The structure of your special journals depends on the journal's purpose and the information you intend to keep in it. Figure 4.2 shows an example of a special sales journal.

Figure 4.2
The special sales journal for Bell Books records credit purchases by its customers.

F10 =SUM(SJDebitsCash)

	A	B	C	D	E	F	G	H	I	J
1	Date	Account debited	Invoice	✓	Debit					
2	6/3/2011	Howell, Fred	620	✓	326.67					
3	6/8/2011	Jackson, Ellen	621	✓	954.57					
4	6/10/2011	Brown, Elaine	622	✓	223.86					
5	6/11/2011	Jones, Judith	623	✓	498.00					
6	6/20/2011	Howell, Fred	624	✓	165.00					
7	6/25/2011	Thomas, Eric	625	✓	98.00					
8	6/28/2011	Anderson, David	626	✓	145.00					
9										
10						2,411.10				
11										

Sales Journal

Ready 100%

Notice these key elements of the sales journal:

- Each account has a different customer name (for example, Fred Howell, Ellen Jackson, and so on). These accounts are summarized in the accounts receivable ledger so that Bell Books can keep track of whether a customer owes money on an account (and, if so, how much).See the "Creating Subsidiary Ledgers" section later in this chapter for more information on the accounts receivables ledger.
- Each customer account in the sales journal is an individual account receivable—an asset account—therefore, the transaction amount is recorded in the sales journal as a debit. (Recall that asset accounts and expense accounts record increases as debits.)
- A sales journal usually contains several transactions, and a particular customer often shows up more than once. For example, Figure 4.2 shows that Fred Howell has made two purchases during June. But the journal itself does not summarize his account. Customer account summaries are found in the accounts receivable ledger.
- Unlike the general journal shown in Figure 4.1, the special sales journal has no column titled Credit. The reason is that the offsetting credit amounts are accumulated in the general ledger's sales account.
- The check marks in Column D, shown in Figure 4.2, indicate that a particular transaction has been posted from the sales journal to the sales account in the general ledger.

Finding and Using Special Symbols

You can show a variety of special characters in Excel by choosing a particular font. These characters can represent the entire cell entry or only a portion of the entry. For example, to show the check marks in Figure 4.2, the cells were formatted using the Wingdings font. When a cell is formatted with this font, entering the formula `=CHAR(252)` causes Excel to display a check mark.

To find a particular symbol, you can enter a numeric series from 0 to 255 in, say, cells A1:A256 of a worksheet. To do so, click the Home tab, select Fill from the Editing group, and click Series.. In versions prior to Excel 2007, use Edit, Fill, Series.

With the numbers 0 through 255 in column A, enter this formula in cell B1:

```
=CHAR(A1)
```

Then copy and paste from B1 into the range B2:B256. Select B1:B256, click the Home tab, and select Symbol from the Font drop-down box in the Font group. In versions prior to Excel 2007, choose Format, Cells.

Then, using the Font tab, assign the range a font such as Symbol. Look through the B1:B256 range to see whether it contains the symbol you want. When you find it, use the combination of the value, the CHAR function, and the font to display the symbol. You can assign different fonts to different characters in a text entry by highlighting the character in the formula box and continuing exactly as you would to format a full cell.

Another approach is to click the Ribbon's Insert tab and select Symbol from the Text menu. Select a font, such as Wingdings, and scroll down until you find the symbol you're looking for. Click it and then click Insert. If you're using Excel 2002 or 2003, start by choosing Symbol from the Insert menu.

Your choice of method is just a matter of personal preference. Therefore, choose the one that makes it easier for you to find a given symbol.

Structuring the Special Purchases Journal

The purpose of the special purchases journal differs from that of the special sales journal, and it's structured a little differently. Figure 4.3 shows Bell Books' special purchases journal.

Figure 4.3
The special purchases journal for Bell Books records credit purchases from its suppliers.

F8 =SUM(PJCreditsCash)

	A	B	C	D	E	F	G	H	I
1	Date	Account credited	Invoice date	✓	Credit				
2	6/2/2011	Lenney Distributing	6/2/2002	✓	2,262.21				
3	6/9/2011	Neal Publishing	6/4/2002	✓	840.85				
4	6/15/2011	Neal Publishing	6/9/2002	✓	963.00				
5	6/21/2011	Lenney Distributing	6/16/2002	✓	885.00				
6	6/24/2011	Lenney Distributing	6/25/2002	✓	1,433.00				
7									
8						6,384.06			
9									

Purchases Journal

Ready 100%

For tracking purposes, the sales journal uses the invoice *number* in column C. In contrast, the purchases journal uses the *date* of the supplier's invoice; this helps Bell Books track the length of time a payable invoice has been outstanding. Of course, if you want, you can also show the supplier's invoice number in the purchases journal. Doing so will help you keep your records straight if you have a supplier who might send you more than one invoice with the same date.

Another difference between the sales and purchases journals is that the purchases journal shows the amount of the purchase as a credit, whereas the sales journal shows the amount of a sale as a debit. The purchases are posted to accounts payable, a liability account that records the company's noncash purchases, so an increase in its balance is recorded as a credit. The sales are posted to accounts receivable, an asset account that records the company's noncash sales, so balance increases are recorded as debits.

Again, there is no debit column in the purchases journal because all entries in this journal are noncash purchases. The offsetting debit entry is found in the accounts payable ledger.

Structuring the Cash Receipts Journal

The two special journals named `Sales` and `Purchases` account for all of Bell Books' noncash transactions. It's still necessary to account for the cash receipt and cash payment transactions. Figure 4.4 shows the special cash receipts journal.

Figure 4.4
Normally, a cash account is debited for cash payments.

C15 =SUM(CRJDebitsCash)

	A	B	C	D	E	G	H	I	J	K
1			Debits			Credits				
2				Other Accounts				Accounts	Sales	Other
3			Cash			Account		Receivable		Accounts
4	Date	Explanation	Amount	Account	Amount	Credited	✓	Amount	Amount	Amount
5	6/1/2011	Add'l investment	$ 52,000.00			J. Bell, Capital				$ 52,000.00
6	6/6/2011	Retail sales	$ 76.68						$ 76.68	
7	6/7/2011	Retail sales	$ 124.89						$ 124.89	
8	6/8/2011	Invoice dated 6/3	$ 326.67			Howell, Fred	✓	$ 326.67		
9	6/12/2011	Sale of 3rd floor	$ 18,000.00	Notes Rcvble	$ 6,000.00	Building				$ 24,000.00
10	6/12/2011	Invoice dated 6/8	$ 954.57			Jackson, Ellen	✓	$ 954.57		
11	6/19/2011	Invoice dated 6/15	$ 223.86			Brown, Elaine	✓	$ 223.86		
12	6/25/2011	Retail sales	$ 87.43						$ 87.43	
13	6/29/2011	Bank loan	$ 13,000.00			Notes Payable				$ 13,000.00
14										
15			$84,794.10		$6,000.00			$1,505.10	$289.00	$89,000.00
16										

Cash Receipts Journal

Ready 100%

The structure of the cash receipts journal is quite different from the structure of the sales and purchases journals. As explained previously, all transactions entered in the sales journal are credit transactions that are posted to one ledger account: the sales account. Similarly, all transactions in the purchases journal, also credit transactions, are posted to the purchases ledger account. However, you might want to post cash transactions, both receipts and payments, to one of several accounts. The structure of the cash receipts and cash payments journals builds in the additional flexibility required.

Usually, a transaction in the cash receipts journal is posted to the cash ledger account. The total ($84,794.10) of all the individual transactions that appears in column C of Figure 4.4 is posted to the cash account in the general ledger. For example, the owner of Bell Books invests an additional $52,000 in the company on June 1 (see cell C5 of Figure 4.4). This investment comes in the form of cash and, consequently, is posted—as part of the total cash receipts in column C—as a debit here. It also appears as a debit in the general ledger's cash account. (See the "Creating the General Ledger" section later in this chapter for more information about the general ledger.)

Notice that on June 12, Bell Books sold the third floor of its building to another company for $24,000. (It's a small building.) Bell Books received $18,000 of the $24,000 in cash and accepted a note receivable from the buyer for the remaining $6,000. (See cells C9 and E9 of Figure 4.4.) The $6,000 *could* have been entered in the general journal instead of in the cash receipts journal; however, it's convenient to keep the two portions of the transaction together so that you can see the entire transaction in one place.

This transaction illustrates the purpose of the columns titled Other Accounts, shown in columns D and E of the worksheet in Figure 4.4. Column D contains the name of the ledger

account where the transaction will be posted, and column E contains the debit amount that will be posted there.

The amounts in the Debits section of the cash receipts journal are posted to the general ledger as follows:

- The *total* of the receipts in column C, $84,794.10, is shown in cell C15 of Figure 4.4. It is posted as one total value to the general ledger's cash account. (Again, see the "Creating the General Ledger" section later in this chapter for more information about the general ledger.)
- The *individual amounts* of any receipts in column E are posted in the general ledger to the accounts that are named in column D. In Figure 4.4, that's just one account: notes receivable (abbreviated in the figure as Notes Rcvble).

Figure 4.4 also shows the Credits section of the cash receipts journal, which has a structure similar to its Debits section. Two main ledger accounts are credited when transactions are posted from the cash receipts journal: accounts receivable (column I) and sales (column J).

For example, Bell Books received a check on June 8 from Fred Howell as payment for an invoice dated June 3. The transaction is shown in row 8 of the cash receipts journal, in Figure 4.4, as follows:

1. An entry for the amount of the check is made in cell C8, indicating that the general ledger account `Cash` is to be debited by $326.67.
2. An entry showing the account that is to be credited is made in cell G8: Fred Howell's account will be credited by $326.67. That's an account receivable, thus an asset, so a reduction in its balance due to a payment is recorded as a credit in the next step.
3. An entry showing the amount of the check is made in cell I8, indicating that the ledger account named `Accounts Receivable` is to be credited by $326.67.
4. When the amount of $326.67 is actually posted to accounts receivable, a check mark is entered in cell H8 to indicate that the posting has been made.

The posting of $326.67 as a debit to `Cash` and as a credit to accounts receivable shows that the amount is moved *into* `Cash` *from* accounts receivable at the point that the payment is received.

Entering Sales in Cash and Sales Ledgers

As another example, when a customer pays $76.68 by check on June 6 (see row 6 in Figure 4.4), that amount is entered into cell C6 to show that the account named `Cash` is to be debited by $76.68. The same figure is entered into cell J6 to show that the account named `Sales` is to be credited by $76.68.

The reason for entering the sale amount of $76.68 in both the cash and the sales ledger accounts is due to a concept that this book has assumed but not yet made explicit: *double-entry* accounting. Every business transaction must be entered as both a debit and a credit, and the debits and credits must be made in different accounts. The double-entry method

has various benefits, one of which is that the sum of all debit entries in the ledger should equal the sum of all credit entries. If the two sums are not equal, you know that the business's accounts are not in balance and an error has been made somewhere.

For example, consider the transaction shown in row 13 of Figure 4.4. On June 29, Bell Books took out a bank loan for $13,000. In exchange for the loan papers, the bank wrote a check to Bell Books for $13,000, and the company deposited it in a checking account. Therefore, Bell Books' cash assets have increased by $13,000. But the company has not suddenly become $13,000 richer by depositing a check; eventually, it will need to repay the loan. Therefore, the company's liabilities have also increased by $13,000, which is documented by increasing the account named `Notes Payable` by $13,000.

The net effect, of course, is that the company's worth remains the same because loans aren't profits. In contrast, when Bell Books sells a book to a customer for cash, four events occur:

1. Its cash account (an asset account) is debited.
2. Its sales account (a revenue account) is credited.
3. Its inventory account (an asset account) is eventually credited.
4. Its cost of goods sold account (a revenue account) is eventually debited.

If the amount involved in 1 and 2 is greater than the amount involved in 3 and 4, the company makes a profit. Buy low and sell high.

Finally, notice that column H, in the Credits section of the cash receipts journal, indicates with a check mark whether a receipt of funds has been posted. The only entries in this journal that are ever marked as posted are payments to accounts receivable. The reason is that accounts receivable maintains detailed information about specific accounts (for example, Fred Howell's account, Ellen Jackson's account, and so on). Therefore, when it receives funds for a specific customer account, Bell Books posts the amount to that account.

In contrast, the company can post a total amount for cash sales to the general ledger's sales account. In that account, there's no particular reason to maintain information about who bought an item from Bell Books for cash.

> **NOTE**
> Of course, there are plenty of marketing, regulatory, and legal reasons to collect and track that sort of cash transaction data. A nonprofit, for example, must record individual cash donations to support its donors' tax deductions. But there is no particular reason having solely to do with financial accounting that would cause you to record the name of the kid who just bought a skate key from your five-and-dime. (Especially because five-and-dimes aren't around any longer to sell the skate keys that aren't being made.)

Structuring the Cash Payments Journal

The sales journal and the purchases journal collect information about noncash transactions, and the cash receipts journal collects information about cash paid to the company.

Figure 4.5
Normally, the cash account is credited for cash payments.

D15 =SUM(CPJCreditsCash)

	A	B	C	D	E	F	H	I	J	K	L
1				Credits			Debits				
2					Other Accounts				Accounts	Purchases	Other
3				Cash			Account		Payable	Amount	Accounts
4	Date	Check	Explanation	Amount	Account	Amount	Debited	✓	Amount		Amount
5	6/1/2011	2416	Phone, May	$ 324.66			Phone Expense				$324.66
6	6/2/2011	2417	Bookmarks	$ 255.55						$255.55	
7	6/8/2011	2418	Invoice, 6/2	$ 2,262.21			Lenney Distributing	✓	$2,262.21		
8	6/17/2011	2420	Paid salaries	$ 5,252.20			Salaries				$5,252.20
9	6/26/2011	2421	Invoice, 6/16	$ 840.85			Neal Publishing	✓	$840.85		
10	6/27/2011	2422	Books	$ 525.13						$525.13	
11	6/28/2011	2423	Books	$ 872.66						$872.66	
12	6/29/2011	2424	Ad in News	$ 95.09			Advertising				$95.09
13	6/29/2011	2425	Insurance (3 yrs.)	$ 990.95			Unexpired Insurance				$990.95
14											
15				$ 11,419.30					$ 3,103.06	$ 1,653.34	$ 6,662.90
16											

Cash Payments Journal

Ready 100%

Unfortunately, the company must also pay out cash, and those payments are recorded in the cash payments journal, shown in Figure 4.5.

The overall structure of this journal is the same as that of the cash receipts journal, with one major difference: The Credits section is shown to the left of the Debits section, instead of to the right. Normally, debits are shown to the left of credits, but in a special journal, you can put the columns in any sequence.

It's more convenient to show the Credits section to the left of the Debits section in the cash payments journal because it places the Cash column on the left side of the worksheet, where it's more easily accessible. The accessibility is important because, in cash transaction journals, every transaction contains a cash entry.

Notice in Figure 4.5 that the specific ledger accounts in columns J and K are accounts payable and purchases, respectively. The reason is that they are the accounts most frequently debited when your company makes a cash payment. Typically, you use other accounts, such as salaries and telephone expenses, only once a month when checks are written to employees and to the phone company. Again, the way your company does business should determine which accounts you show as columns in the cash payments journal and which ones you show as line items in the Other Accounts column, such as column L in Figure 4.5.

Excel Tables and Dynamic Range Names

Before continuing with the topics of journals and ledgers, it's useful to take a detour into the topics of Excel tables and named ranges. The reason is that tables and ranges can help you manage your journals and ledgers (and other data groupings).

Prior to the 2007 version, Excel used what were called *lists*. A list is not a formal structure such as a pivot table or an embedded chart. It is a way of organizing data that conforms to certain Excel requirements. A list is a rectangular grouping of adjacent cells, with different records (for example, people, accounts, or transactions) in different rows, and with different fields (for example, person's first name, account balance, or transaction date) in different columns. The name of each field is in the list's first row.

There was never an Excel command to insert or build or convert or name a list. However, certain tasks such as building pivot tables and using the built-in Data Form required that basic data layout, and they still do.

Excel 2007 made a formal object of the informal list and called it a *table*. This creates some confusion with other, different structures such as pivot tables and data tables (the latter are used in so-called "What-If Analysis"), and Microsoft documentation repeatedly warns the reader about that sort of misunderstanding.

Tables in Excel 2007 and 2010 have some capabilities that lists do not:

- They have formal names: by default, `Table1`, `Table2`, and so on. You can change the default name if you want.
- Their number of rows and columns automatically grows as you add data adjacent to the rightmost column or bottommost row.
- They have optional total rows, shown at the bottom of the table, that show column statistics such as Sum or Count.
- Their columns can behave like range names.

For example, if the range A1:E6 in Figure 4.3 were an Excel table named `Table1`, you could use this formula

```
=SUM(Table1[Credit])
```

to get the total of the values in cells E2:E6. If the formula is in a cell that is part of the table, you don't need to use the table name as a qualifier. That is, again in Figure 4.3, this formula would work in a cell in or adjacent to a table in A1:E6:

```
=SUM([Credit])
```

If you have a range of data laid out like the data in Figure 4.3, you can convert it to a table in Excel 2007 or 2010. Just select a cell in the range, click the Ribbon's Insert tab, and click Table in the Tables group.

Having done so, you can show the total of the credits in the purchases journal using

```
=SUM(Table1[Credit])
```

or, if you've renamed the table, using something like this:

```
=SUM(PurchaseJournal[Credit])
```

Notice that the formula works much like using the `SUM` function with a named range. If the range E2:E6 in Figure 4.3 were named `CreditAmounts`, you could get the total of those cells with either this:

```
=SUM(E2:E6)
```

or this:

```
=SUM(CreditAmounts)
```

The basic structure of `=SUM(CreditAmounts)` is similar to the function as applied to a column in a table: `=SUM(Table1[Credit])`. But if a range name refers directly and specifically to, say, E2:E6, problems can arise. Those are discussed next.

Building Dynamic Range Names

Each of the first three chapters of this book discussed range names. You have seen how to specify a particular range of cells using a name that you provide: `Inventory_Product_Code`, for example. Those range names were static. The name always refers to the same set of cells unless you change the address that the name refers to.

Using static range names has several advantages but also some drawbacks. Suppose that you have five values in the range A1:A5. These values represent your company's revenues for the first five months of the year, and you have given the range A1:A5 the name `Revenues`. Elsewhere on the worksheet, you use the formula `=SUM(Revenues)` to display your total year-to-date revenues.

At the end of June, you enter June's revenue figure in cell A6. Now, to get the correct result for `=SUM(Revenues)`, you need to click the Formulas tab and select Name Manager in the Defined Names group, and include A6 in the `Revenues` range (prior to Excel 2007, use Insert, Name, Define).

That's a headache you don't need every month—actually, the real headache is remembering to do it.

Using the OFFSET Function in Dynamic Range Names

What you need is a way to make the name `Revenues` respond by expanding its own address when you add new values: You need a *dynamic* range name, not a static one. You can create a dynamic range name by using Excel's `OFFSET` function. When you define a name, you're not required to enter a specific worksheet address in the Refers To box. You can enter a value or a formula. For example, you could use this formula:

```
=OFFSET($A$1,0,0,COUNT($A:$A),1)
```

This formula illustrates the `OFFSET` function. It returns a reference to a range of cells and contains these arguments:

- **An anchor cell (or range of cells)—**In the example, it's one cell: $A$1. The anchor cell informs `OFFSET` which cell to use as its basis.
- **A `Rows` argument—**In the example, this is the first zero. The reference that `OFFSET` returns is shifted (or *offset*) that number of rows from the anchor cell. In this case, the reference is shifted from $A$1 by zero rows.
- **A `Columns` argument—**In the example, this is the second zero. `OFFSET` shifts the reference by that number of columns from $A$1. Again, this example shifts the reference by zero columns. So far, the `OFFSET` function is just returning a reference to its own anchor cell, $A$1.

- **A `Height` argument**—In this example, this is `COUNT($A:$A)`. Using the `COUNT` function informs `OFFSET` of how many numeric values exist in a range of cells, which is column A in this example. This is the heart of the dynamic range definition. When the number of numeric values in column A changes, it causes `COUNT` to recalculate. In turn, the `OFFSET` function recalculates and returns a reference with a different number of rows.
- **A `Width` argument**—In this example, this is the number `1`. It defines the number of columns in the range that `OFFSET` returns.

> **TIP** Use `COUNTA` instead of `COUNT` if your range could legitimately contain text values, not just numbers.

So this formula returns a reference that depends on the number of numeric values in column A:

```
=OFFSET($A$1,0,0,COUNT($A:$A),1)
```

Suppose that column A contains six numbers. The formula returns a reference that is offset from $A$1 by zero rows and zero columns. The reference is six rows high and one column wide. So it returns the reference A1:A6.

As soon as a seventh number is entered in column A, the `COUNT` and `OFFSET` functions combine to make the reference one row larger. The reference would then be A1:A7. The name `Revenues`, defined in this way, is not a static range name, but a dynamic one.

The dynamic range name isn't subject to the drawback to static range names mentioned at the beginning of this section. You no longer need to manually redefine the name `Revenues` when a new revenue figure is included with the existing values. Your formula `=SUM(Revenues)` automatically recalculates and returns the sum of all the values in the `Revenues` range.

> **CAUTION**
>
> Unfortunately, the dynamic range name comes with a drawback of its own. A value that's unintentionally added to the column causes the range to become larger. Continuing the `Revenues` example, suppose that you inadvertently entered a numeric value such as a date in column A. The `COUNT` function (and, therefore, the `OFFSET` function that includes the `COUNT`) would respond to the date's presence even if it were all the way down in cell A1048576. The `Revenues` range would contain one more row than the number of revenue values.
>
> In this particular example, that might not make much difference. But there are plenty of other situations in which you'd wind up with a serious error.

One further point: You must keep formulas that refer to the dynamic range outside of that range. Suppose that you put your `=SUM(Revenues)` formula in column A. The formula gets counted as one of the values that define the extent of the range, which means that the

Revenues range in the SUM function is helping to define itself. That's a circular reference error, and Excel won't let you get away with it unscathed—at the very least, you'll need to resolve an error message.

Using Dynamic Range Names in the Journals

The sales and the purchases journals use several dynamic range names. Each range name is sheet level. For example, there is a range named 'Purchases Journal'!Amount and one named 'Sales Journal'!Amount. Each journal has these names and definitions:

- TransactionDate—In the case of the sales journal, this range refers to the following:
  =OFFSET('Sales Journal'!$A$1,1,0,COUNT('Sales Journal'!$A:$A),1)
- Account—It's useful and convenient to make this name, as well as the names Posted and Amount, dependent on the size of the TransactionDate range. On the Sales worksheet, the name Account is defined as follows:
  =OFFSET('Sales Journal'!TransactionDate,0,1)

  The reference it returns is offset from TransactionDate by zero rows and one column. When you do not supply a Height or a Width argument for the OFFSET function, as in this example, it defaults to the Height and Width of the anchor argument. In this case, that corresponds to the TransactionDate range, so the Account reference has as many rows and columns as the TransactionDate.
- Posted—This name is defined as follows:
  =OFFSET('Sales Journal'!TransactionDate,0,3)

It is offset from TransactionDate by zero rows and three columns.

- SJDebitsCash—The definition is as follows:

  =OFFSET('Sales Journal'!TransactionDate,0,4)

The name SJDebitsCash uses a naming convention for the journals in the Financial Reports.xlsm workbook: the initials of the sheet name (here, SJ for sales journal), followed by either Debits or Credits, followed by the name of the account that has been debited or credited. The corresponding name in the purchases journal is PJCreditsCash.

The names TransactionDate, Amount, and Posted on the purchases journal use the sheet named 'Purchases Journal' instead of 'Sales Journal' before the exclamation point in the OFFSET function, but they are defined identically otherwise.

Choosing Between Tables and Dynamic Range Names

Both tables and dynamic range names have one particularly valuable advantage over static range names in Excel: They automatically redefine their size when you add new data. That means that if there's another part of the workbook—a chart, for example, or even just a formula—that takes its data from the named range or table, it will automatically update when the table updates or when the range is redefined.

> **TIP** If you use a dynamic range name as the source for a data series in an Excel chart, use its name qualified either by the name of the range's home sheet or by the name of the workbook: That is, if `Credits` is a dynamic range name, use `=SERIES(,,Sheet1!Credits,1)` or `=SERIES(,,Book1.xlsx!Credits,1)` rather than `=SERIES(,,Credits,1)`. You'll need the qualified name even if the chart is embedded in a worksheet that constitutes the name's scope.

So, how do you decide whether to use a dynamic range name or a table? It's largely a matter of personal preference, and my own is to use dynamic range names. But there's no reason that you should follow my personal preference. Instead, consider the following drawbacks.

If you have more than one column in a table, you need to specify both the table name and the column header that you want to use. For example:

```
=SUM(SalesJournal[Debits])
```

whereas if `Debits` were a dynamic range name, you could use this:

```
=SUM(Debits)
```

It's marginally easier for me to remember the name of a range than to remember the name of a table *and* the field header.

Furthermore, if I use the name of a table in the definition of a charted data series, Excel converts the table name to a worksheet reference. Thus, `=SERIES(,,Sheet1!Table3[Credits],1)` would get changed to something like `=SERIES(,,Sheet1!$D$2:$D$20,1)`.

However, if you subsequently add data to the table, the charted series definition is updated to show the new data. For example, `=SERIES(,,Sheet1!$D$2:$D$20,1)` might automatically become something like `=SERIES(,,Sheet1!$D$2:$D$25,1)`.

Dynamic range names have drawbacks, too. A dynamic range name determines the number of rows in the range by using `COUNT` or `COUNTA` to get the number of values already present, usually in a column. Therefore, if you have extraneous values in the column, say in A100 when the range you're interested in occupies A1:A20, the `COUNT` function can easily return one value too many and make the range one row too big.

The other side of that problem is missing values. If you have values in A1:A20 except for one empty cell in, say, A11, the count will be too small by one, and so will the range. This is one reason that I define names such as `Debits` as offsets from a transaction date range. There will be a transaction date for each transaction, but not necessarily a value in the Debits column for each transaction.

Using tables avoids both these problems at the cost of having to qualify the table's column name by the name of the table itself, and of losing the table name in a chart series definition.

Understanding Ledgers

Just as you can choose which special journals to maintain for your business, you can choose which ledgers to maintain. If you kept only one ledger with detailed information about all accounts, it would lose some of its value as a summary document.

Therefore, it's normal to establish some *subsidiary ledgers* that contain detailed information from the journals. You can then keep information about specific sales and specific purchases in the subsidiary ledgers, and you can transfer their summary totals to a general ledger. Because of the frequency of transactions involving sales and purchases, many businesses keep information about these transactions in an accounts receivable subsidiary ledger and an accounts payable subsidiary ledger.

Creating the General Ledger

By keeping detailed information from the journals in these subsidiary ledgers, it's easier to check the status of your individual accounts with both creditors and customers. At the same time, you can keep the general ledger from becoming cluttered with detailed information about individual customers who owe you money, and about individual creditors who expect to be paid. See Figure 4.6, which displays a general ledger's asset and liability accounts.

Figure 4.6
The general ledger should show the account, date, and journal reference for each debit and credit.

D4 =SUM('Cash Receipts Journal'!CRJDebitsCash)

	A	B	C	D	E
1	Account	Date	Reference	Debit	Credit
2	*Asset & Liability Accounts*				
3	Cash	5/31/2011	Balance	$ 24,525.00	
4		6/30/2011	Cash Receipts	$ 84,794.10	
5		6/30/2011	Cash Payments		$ 11,419.30
6		6/30/2011	Balance	$ 97,899.80	
7	Notes Receivable	5/31/2011	Balance	$ -	
8		6/12/2011	Cash Receipts	$ 6,000.00	
9		6/30/2011	Balance	$ 6,000.00	
10	Accounts Receivable	5/31/2011	Balance	$ 3,256.00	
11		6/8/2011	General: Return of books		$ 42.00
12		6/30/2011	Sales	$ 2,411.10	
13		6/30/2011	Cash Receipts		$ 1,505.10
14		6/30/2011	Balance	$ 4,120.00	
15	Unexpired Insurance	5/31/2011	Balance	$ -	
16		6/29/2011	Cash Payments	$ 990.95	
17		6/30/2011	Balance	$ 990.95	
18	Building	5/31/2011	Balance	$ 96,000.00	
19		6/12/2011	Cash Receipts		$ 24,000.00
20		6/30/2011	Balance	$ 72,000.00	

General Ledger

Ready 100%

Every dollar entry in this ledger refers directly to either an entry in a journal or a total of the transactions in a journal. For example, the formula in cell D8 is as follows:

```
='Cash Receipts Journal'!$E$9
```

Figure 4.4 shows the cash receipts journal. Notice that the value of $6,000 in cell D8 of the general ledger in Figure 4.6 is the single entry in the cash receipts journal that represents the note accepted by Bell Books in partial payment for the third floor of its building. In contrast, the formula in cell D4 of Figure 4.6 is as follows:

```
=SUM('Cash Receipts Journal'!CRJDebitsCash)
```

Here, the name `CRJDebitsCash` is a dynamic range name that refers to this:

```
=OFFSET('Cash Receipts Journal'!TransactionDate,0,2)
```

This dynamic range does not include the sum in cell C15 in Figure 4.4. That's because the range is based on TransactionDate, which extends only through A13. OFFSET's optional Rows argument is not supplied, so the row count is taken from TransactionDate.

This example illustrates how a ledger entry summarizes all the transactions in a given category that appear individually in a journal.

So debits to the general ledger's cash account are based on the cash receipts journal. In contrast, credits to the general ledger's cash account are based on the cash payments journal. Figure 4.5 shows the cash payments journal, and column D contains the cash payments made during the month, totaling $11,419.30 (cell D15). The outflow of cash is represented in the general ledger by credits to the cash account. The formula in cell E5 of Figure 4.6 is as follows:

```
=SUM('Cash Payments Journal'!CPJCreditsCash)
```

Here, the name `CPJCreditsCash` is a dynamic range name scoped to the `Cash Payments Journal` worksheet that refers to this:

```
=OFFSET('Cash Payments Journal'!TransactionDate,0,3)
```

Consider the accounts receivable classification in Figure 4.6. It contains three amounts: `$42.00`, `$2,411.10`, and `$1,505.10`. The `$42.00` value represents the return of merchandise from a customer and is taken from the general journal.

This formula returns the `$2,411.10` value:

```
=SUM('Sales Journal'!SJDebitsCash)
```

The value is the sum of all sales in the sales journal, which by definition are noncash transactions. A range in the sales journal is named `SJDebitsCash` and refers to cells E2:E8 of that worksheet, shown in Figure 4.2. The range contains noncash sales that are recorded as debits to accounts receivable. (The dynamic range name `SJDebitsCash` was discussed in this chapter's section on dynamic range names in journals.)

This formula returns the `$1,505.10` value in cell E13 of Figure 4.6:

```
=SUM('Cash Receipts Journal'!CRJCreditsAR)
```

This refers to the total of the credits in accounts receivable, from the Credits section of the cash receipts journal (see Figure 4.4).

In this way, the activity in the accounts receivable account is summarized for the month. New credit sales are totaled in cell D12, and receipts for credit sales are totaled in cell E13, both in the general ledger. You might find it useful to open the `Financial Reports.xlsm` file and view each of its entries to determine its source in the journals. If you do, you will find that every value for June in the general ledger refers (either directly, or indirectly via the `SUM` function) to an entry in a journal.

Figure 4.7 shows the Revenue and Expense section of the general ledger.

Figure 4.7
All general ledger entries should be linked to the general journal or special journal transactions.

D60 {=SUM((B4:B59=DATE(2011,6,30))*(C4:C59="Balance")*D4:D59)}

	A	B	C	D	E
36	*Revenue & Expense Accounts*				
37	Sales	5/31/2011	Balance		$ -
38		6/30/2011	Sales		$ 2,411.10
39		6/30/2011	Cash Receipts		$ 289.00
40		6/30/2011	Balance		$ 2,700.10
41	Sales Returns	5/31/2011	Balance	$ -	
42		6/8/2011	General	$ 42.00	
43		6/30/2011	Balance	$ 42.00	
44	Purchases	5/31/2011	Balance	$ 8,827.00	
45		6/30/2011	Purchases	$ 6,384.06	
46		6/30/2011	Cash Payments	$ 1,653.34	
47		6/30/2011	Balance	$ 16,864.40	
48	Purchase Returns	5/31/2011	Balance		$ -
49		6/17/2011	General		$ 1,525.00
50		6/30/2011	Balance		$ 1,525.00
51	Salaries	5/31/2011	Balance	$ -	
52		6/17/2011	Cash Payments	$ 5,252.20	
53		6/30/2011	Balance	$ 5,252.20	
54	Telephone	5/31/2011	Balance	$ -	
55		6/1/2011	Cash Payments	$ 324.66	
56		6/30/2011	Balance	$ 324.66	
57	Advertising	5/31/2011	Balance	$ -	
58		6/29/2011	Cash Payments	$ 95.09	
59		6/30/2011	Balance	$ 95.09	
60				$210,698.10	$210,698.10

General Ledger

Ready 100%

The entries in the general ledger's Revenue and Expense section follow the pattern established in its Assets and Liabilities section. That is, each entry comes from a journal and is either a specific journal entry or the total of several journal entries that belong to the same account.

The values in cells D60 and E60 of Figure 4.7, `$210,698.10`, show that the accounts are in balance. As noted previously, the double-entry method is meant to ensure that the total of the debits for a given period equals the total of the credits for the same period. Each transaction is a debit to one account and a credit to another.

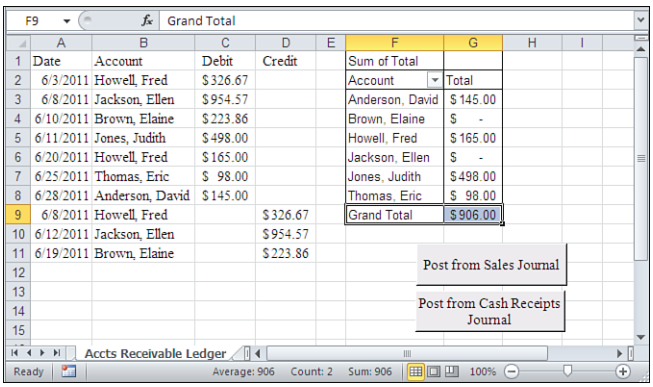
Cells D60 and E60 total the debit and credit account balances in the general ledger. They are equal, which is evidence that the accounts are in balance. However, just because the two totals are equal doesn't necessarily mean that all entries are accurate. For example, compensating errors might exist. But if the two amounts were unequal, it would demonstrate that at least one error existed somewhere.

Creating Subsidiary Ledgers

Bell Books uses two subsidiary ledgers: accounts receivable and accounts payable. The purpose of these ledgers is to keep tabs on accounts that customers have with Bell Books and that Bell Books has with its suppliers. Figure 4.8 shows the accounts receivable ledger.

Neither the accounts receivable ledger nor the accounts payable ledger links to the general ledger. Each subsidiary ledger repeats some of the information in the general ledger; specifically, the accounts receivable ledger provides details about the individual accounts for Bell Books' credit customers.

Figure 4.8
The accounts receivable ledger helps track the status of individual accounts.



Two worksheet buttons appear on the accounts receivable ledger. One is labeled Post from Sales Journal. That button contains VBA code that posts information about noncash sales from the sales journal to the accounts receivable ledger. Clicking the button causes the VBA code to run. The button labeled Post from Cash Receipts Journal also contains associated VBA code that posts information from the cash receipts journal to the accounts receivable ledger. The cash receipts represent payments for credit sales that were recorded in the sales journal.

The accounts receivable ledger also contains a pivot table that provides the current balance for each of the accounts receivable.

In Figure 4.8, cell D9 contains the value `$326.67`. That value represents the payment received from Fred Howell on June 8 that was entered into the cash receipts journal (refer to cell I8 of Figure 4.4). The value `$326.67` is transferred to the general ledger (refer to cell E13 of Figure 4.6) as part of the total of the credits to accounts receivable from the cash receipts journal.

This example illustrates that amounts posted to subsidiary ledgers must be included in two ledgers: both the subsidiary ledger and the general ledger. Doing so ensures that the account's amount, as shown in the general ledger, equals the amount shown in the subsidiary ledger.

Automating the Posting Process

You can put buttons in various places in an Excel workbook; for example, you can put buttons on worksheets and user forms such as dialog boxes. By associating the button with VBA code, you can cause Excel to take the actions you want it to when the button is clicked. Here's how to create a button on a worksheet using Excel 2007 or Excel 2010:

1. Click the Developer tab and then click Insert in the Controls group.

> **TIP** If you don't see the Developer tab, establish it by clicking the File tab, choosing Excel Options, and clicking Customize Ribbon.

2. Click the Command Button icon in the ActiveX Controls section. Move the mouse pointer over the worksheet.
3. Holding down the left mouse button, drag across and down on the worksheet to indicate where you want the button to appear. Then release the mouse button.
4. The Command button is established on the worksheet. Right-click the Command button to invoke a shortcut menu. One of its items is View Code. When you are ready to provide the VBA code that is to run when the button is clicked, select the View Code item.
5. The button has a default label, `CommandButton1` if it's the first Command button on the worksheet. You can change the label in various ways, including this one: Right-click the button and select CommandButton Object, Edit from the shortcut menu. When you move your mouse pointer over the button's label, the pointer changes to an I-bar. Hold down the left mouse button and drag across the label to highlight it. Then type whatever text you want to appear on the button.
6. Deselect the button by clicking any worksheet cell.

The Post from Sales Journal button shown in Figure 4.8 is linked to the VBA procedure named `PostFromSalesToAR` in Listing 4.1, which actually performs the posting.

Listing 4.1 VBA Procedure PostFromSalesToAR

```
Option Explicit
Option Base 1

Sub PostFromSalesToAR()
Dim SalesDate As Range, Acct As Range, Posted As Range, SalesAmount As Range
Dim ThisTransaction As Integer, NextEntryRow As Long


With Sheets("Sales Journal")
   .Activate
   Set SalesDate = .Range("TransactionDate")
   Set Acct = .Range("Account")
   Set Posted = .Range("Posted")
   Set SalesAmount = .Range("SJDebitsCash")
End With
NextEntryRow = ThisWorkbook.Sheets("Accts Receivable Ledger") _
    .Range("TransactionDate").Rows.Count

For ThisTransaction = 1 To Acct.Rows.Count
    If Posted(ThisTransaction) <> Chr(252) Then
        With ThisWorkbook.Sheets("Accts Receivable Ledger")
            .Range("TransactionDate").Offset(NextEntryRow, 0).Resize(1, 1) = _
                SalesDate(ThisTransaction)
            .Range("AccountNames").Offset(NextEntryRow, 0).Resize(1, 1) = _
                Acct(ThisTransaction)
            .Range("Purchases").Offset(NextEntryRow, 0).Resize(1, 1) = _
                SalesAmount(ThisTransaction)
        End With
        Posted(ThisTransaction).FormulaR1C1 = "=CHAR(252)"
        NextEntryRow = NextEntryRow + 1
```

```
    End If
Next ThisTransaction


With ThisWorkbook.Sheets("Accts Receivable Ledger")
   .Activate
   .PivotTables("ARSummary").PivotCache.Refresh
End With
End Sub
```

The first two lines of code in Listing 4.1 set two general options. `Option Explicit` requires you to explicitly declare all variables that you use in the code. If this option is not set, new variables could be declared implicitly on the fly—simply typing a variable name would create it. Because this approach makes the code much more difficult to trace and debug, it's wise to use `Option Explicit` and to explicitly declare each variable. (See the discussion of the `Dim` statements later in this section.)

The second option, `Option Base 1`, requires that the first element of all VBA arrays be element number one. Omitting this option causes Excel to treat the first element of an array as element number zero. Suppose that the first element of `MyArray` was `Fred Howell`. Using `Option Base 1`, you would refer to this value as `MyArray(1)`. Without Option Base 1, you would refer to it as `MyArray(0)`. If you prefer to start counting at one instead of at zero, use `Option Base 1` in your VBA code.

The next statement in Listing 4.1, `Sub PostFromSalesToAR( )`, classifies and names the procedure. VBA has two types of procedures: functions and subroutines. The keyword `Sub` identifies the procedure as a subroutine. VBA procedures always have a set of parentheses following the name. If the parentheses enclose a variable name or names, those variables are being passed as arguments to the subroutine. (The topic of *arguments* is lengthy and complicated. For now, just be aware that arguments help provide values of variables to the procedures that use them.) In this case, no variables are being passed to the subroutine.

The next two statements are `Dim` (short for *dimension*) statements. Using `Option Explicit` requires the code to explicitly declare all variables. The `Dim` statements declare the existence and types of several variables. The variable named `ThisTransaction`, for example, is declared as type `Integer`. This means that the variable cannot take on a numeric value that has a fractional component, such as `3.1416`, and cannot take on a text value such as "Ellen".

The four variables that are declared as type `Range` will later refer specifically to worksheet ranges. After the `Dim` statement declares the variables, the `Set` statements assign them to ranges. In this case, the variables are simply conveniences: The code can refer to the variable instead of to the name of the range in the worksheet where it exists and to the workbook that contains the worksheet.

Consider the next statement in Listing 4.1:

```
With Sheets("Sales Journal")
```

The statement starts what's called a `With` block. That's a set of statements that pertain to the object named in the `With` statement. In this case, the next five statements all have to do

with the sheet named `Sales Journal`. It's a sort of shorthand; instead of repeating the name of the worksheet in each of the five statements, the code encloses them inside a `With . . . End With` structure.

Here's the first statement in the `With` block:

```
.Activate
```

This statement causes Excel to make the sheet named `Sales Journal` the active worksheet. The `Sales Journal` worksheet, of course, contains the sales transactions that will be posted. Without the `With` block, the statement would have had to be this:

```
Sheets("Sales Journal").Activate
```

This probably doesn't seem like much of a savings, but the code can do the same thing with the following four statements, which assign the four `Range` variables to named ranges:

```
        Set SalesDate = .Range("TransactionDate")
        Set Acct = .Range("Account")
        Set Posted = .Range("Posted")
        Set SalesAmount = .Range("SJDebitsCash")
       End With
```

These statements cause the variable `SalesDate` to stand in for the range named `TransactionDate` in the sales journal, `Acct` to stand in for the range named `Account`, and so on. Then the end of the `With` block is indicated.

When the range variables have been set, the code determines the next available row in the `AcctsReceivable` sheet to post a transaction. It does so by counting the number of rows that are presently in the `AcctsReceivable` range:

```
NextEntryRow = ThisWorkbook.Sheets("Accts Receivable Ledger") _
    .Range("TransactionDate").Rows.Count
```

`NextEntryRow` will be used as an offset, so we can be sure not to overwrite what is at present the final row of data on the `Accts Receivable Ledgers` worksheet.

The statement names three objects that are separated by dots. In VBA, this is termed *dot notation* and denotes a hierarchy: An object that follows the dot belongs to an object that precedes the dot. This was implicit in the `With` structure just discussed.

The statement refers to a range named `TransactionDate`. That range belongs to a sheet named `Accts Receivable Ledger`. In turn, that sheet belongs to the `ThisWorkbook` object—that is, the workbook containing the VBA code that's executing.

The `TransactionDate` range has a `Rows` property, which returns a collection of the rows that belong to the `TransactionDate` range. That collection of rows itself has properties, one of which is the `Count` property. The `Count` property returns the number of objects in a collection. As used here, it returns the number of rows in the `TransactionDate` range found on the `AcctsReceivable` sheet that is in the workbook containing the VBA code.

The number of rows in the range is assigned to the variable `NextEntryRow`. That variable is used later in the code as an argument to the `Offset` function, and it determines which row the next transaction will be posted to.

Now the meat of the procedure begins. Consider this statement in Listing 4.1:

```
For ThisTransaction = 1 To Acct.Rows.Count
```

It starts a loop that executes once for each row in the range named `Account`, which is represented by the `Acct` variable. The approach is similar to that used to find the next entry row: `Acct.Rows.Count` returns the number of rows in the range represented by the `Acct` variable. The loop steps through the sales journal row by row, looking for transactions that should be posted to the accounts receivable ledger.

The `For` loop starts at `1` and ends at `Acct.Rows.Count`; therefore, it executes as many times as there are rows in that range. The `Acct` variable represents the range named `Account` on the `Sales Journal` worksheet. So the loop executes once for each row in that range.

The first statement inside the loop is as follows:

```
If Posted(ThisTransaction) <> Chr(252) Then
```

This statement causes the statements that follow it to execute if its condition is satisfied, but to be skipped otherwise. The condition is that a particular value does not equal the ANSI character associated with the number `252`. (Using the Wingdings font, that's a check mark.) The particular value that's tested is the element in the `Posted` range that corresponds to the current value of `ThisTransaction`. The `Posted` variable refers to the range named `Posted` in the sales journal. Notice how it's declared in the second line of the procedure.

In brief, the first time that the loop executes, `ThisTransaction` has a value of `1`. Excel examines the first value in `Posted` to see if it equals `Chr(252)`: that is, to see if it appears on the worksheet as a check mark. If that value is not a check mark, subsequent statements are executed because the transaction has not yet been posted; otherwise, the transaction has already been posted and the subsequent statements (up to `End If`) are skipped.

> **NOTE** The VBA code discussed here uses the `Chr` function, whereas the `CHAR` function is used on an Excel worksheet. VBA has its own set of functions, as does Excel. VBA's `Chr` function is equivalent to Excel's `CHAR` function.

On the other hand, if the current transaction has not yet been posted—if the result of `Chr(252)` is not found in the current element of the `Posted` range—the statements that follow the `If` should be executed, up to the `End If` statement. Executing those statements results in the posting of the transaction to accounts receivable.

More on Dot Notation

Next, the code begins another `With` block:

```
With ThisWorkbook.Sheets("Accts Receivable Ledger")
```

Inside the `With` blocks are references to objects (such as a worksheet), methods (such as a worksheet's `Activate` method), and properties (such as a cell's `Font` property). Some of the references are preceded by a dot (for example, `.Range`), with nothing immediately preceding

that dot. Anything with nothing but a dot preceding it is taken to belong to whatever object is named in the `With` statement. In this case, all objects, methods, and properties referred to inside the `With` block belong to the sheet named `Accts Receivable Ledger`, which, in turn, belongs to `ThisWorkbook`.

Inside the block, several statements use ranges belonging to the `Accts Receivable Ledger` sheet. If the `With` block weren't used, the code would need to repeatedly qualify those ranges by referring to the sheet where it's found, as well as the workbook where the sheet is found. But because the `With` block is used, the code needs to refer to that sheet only once: in the `With` statement.

Each statement inside this `With` block accomplishes some task in the posting of a transaction from the sales journal to the accounts receivable ledger. For example, consider the first statement inside the `With` block:

```
.Range("TransactionDate").Offset(NextEntryRow, 0).Resize(1, 1) = _
    SalesDate(ThisTransaction)
```

Notice the following aspects of the statement:

- Because of the `With` statement, the fragment `.Range("TransactionDate")` is taken to belong to the sheet named `AcctsReceivable` in `ThisWorkbook`.
- The `Offset` fragment describes a range that is offset from the `TransactionDate` range by `NextEntryRow` rows and by zero columns. The first time through the loop, `NextEntryRow` equals the number of rows in the `TransactionDate` range. (Refer to the assignment statement that precedes the `For` loop.) So the offset to the `TransactionDate` range refers to one row below it and in the same column.
- The `Resize` fragment temporarily defines the size of the range as containing one row and one column: thus, one cell.
- This cell is set equal to the value in the `Sales Journal` sheet, referred to by the variable `SalesDate`, for the current value of `ThisTransaction`, which is the transaction being posted. That is, the date of the current transaction is placed in the `TransactionDate` range in the `Accts Receivable Ledger` worksheet.

This process is repeated in the remainder of the `With` block so that the account name and purchase amount are also posted.

The `End With` statement terminates the `With` block. Then a check mark is placed in the `Posted` range of the `Sales` worksheet by means of this statement:

```
Posted(ThisTransaction).FormulaR1C1 = "=CHAR(252)"
```

The variable that controls the offset to the ranges on the `Accts Receivable Ledger` worksheet is incremented. Therefore, the next time the loop executes, the sales date, account, and amount are placed one row farther down:

```
NextEntryRow = NextEntryRow + 1
```

Then the `If` block is terminated with an `End If`. (This marks the end of the statements that are executed if a transaction in the sales journal has not yet been posted.) The final statement in the loop is this:

```
Next ThisTransaction
```

Control returns to the beginning of the loop if the value of `ThisTransaction` does not yet exceed `Accts.Rows.Count`. When the loop has executed once for each row in the range named `Account` in the `Sales` worksheet, the loop ends.

After the final instance of the loop has completed, another `With` block is started for the ledger sheet. The `Accts Receivable Ledger` sheet is activated, and its pivot table is refreshed. The purpose of refreshing the data in the pivot table is to cause the table to incorporate the transactions that have just been posted to the ledger:

```
With ThisWorkbook.Sheets("AcctsReceivable")
   .Activate
   .PivotTables("ARSummary").PivotCache.Refresh
End With
```

Using Pivot Tables to Summarize Individual Accounts Receivable

The purpose of the pivot table is to summarize the current status of all the individual accounts receivable. Figure 4.8 shows the data posted from the sales journal and the cash receipts journal. Notice, for example, that Fred Howell has made two purchases, one for $326.67 and one for $165.00, and accounts receivable has been debited for those two transactions. Howell has made payment for the first transaction but not for the second. And the pivot table shows that Howell still owes $165.00 for the second transaction.

This is managed by creating a calculated field named `Total` inside the pivot table, using that field as the pivot table's data field, and using `Account` as the row field. More specifically, the following steps are taken. (You need to take them only once, when you first add the pivot table to the worksheet.)

1. Click the Formulas tab and then select Define Name from the Defined Names group.
2. Type `DataRange` in the Name box. In the Refers To box, type `=OFFSET(TransactionDate,0,0,ROWS(TransactionDate),5)` and click OK.
3. Select cell F1. Click the Insert tab and select PivotTable from the Tables group.
4. The Create PivotTable dialog box appears. In its Table/Range box, type `DataRange`. Make sure that the Location box contains a reference to cell F1. Click OK.
5. A pivot table schematic appears on the worksheet, the pivot table field list appears, and the Ribbon displays the pivot table options groups. Drag Account from the field list to the Drop Row Fields Here area on the worksheet.
6. On the Ribbon, click Formulas in the Tools group and then click Calculated Field in the drop-down menu.
7. In the Name box, type `Total`.
8. In the formula box, type `= Debit - Credit` and click OK.

9. If the new, calculated `Total` field does not automatically appear in the pivot table's data area, drag it there from the Field list.
10. Close the Field list by clicking its Close box.

The result of this maneuvering is a pivot table with one row for each account. Associated with each account in the table is a calculated field, named `Total` in this example. That field expresses the difference between an account's debits and its credits, thus summarizing the information in the `Purchases` and `Payments` ranges on the `AcctsReceivable` worksheet.

> **TIP** Each sheet in Financial Reports.xlsm contains a sheet-level range named `TransactionDate`. In some circumstances, it's necessary to qualify a reference to `TransactionDate` with the name of the sheet. But if you have, for example, the `Accts Receivable Ledger` sheet active when you name the range in step 2, you don't need to qualify the range name with the name of the sheet. Excel assumes that you mean the instance of the range that belongs to the active sheet.

Pivot tables do not respond immediately to changes in their underlying data sources, regardless of whether the source is a worksheet range or an external data source. In this respect, pivot tables are different from worksheet formulas, defined names, and charted data series, which *do* recalculate immediately when their source changes.

If a change occurs to a pivot table's data source, it's necessary to refresh the pivot table, and that's the purpose of this VBA statement:

```
.PivotTables("ARSummary").PivotCache.Refresh
```

After the pivot table has been refreshed, the subroutine ends with the `End Sub` statement.

You can write this VBA code in slightly more efficient ways. For example, you could create more variables that refer to worksheet ranges in the `Accts Receivable Ledger` worksheet as well as in the `Sales` worksheet. However, the structure was chosen to illustrate a variety of VBA capabilities, including the `With and If` statements, the `Offset` and `Resize` methods, and the automated redefinition of range addresses.

These VBA subroutines and their associated buttons are replicated for the accounts payable ledger (see Figure 4.9).

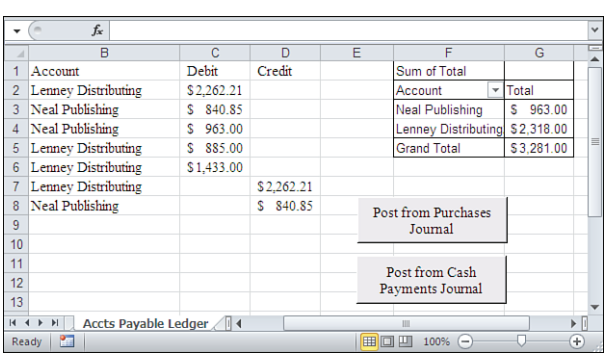


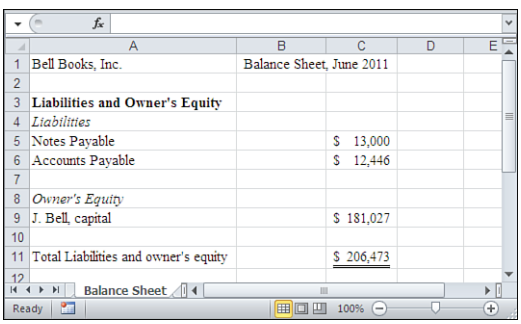
Figure 4.9
The accounts payable ledger details information about open accounts with the company's suppliers.

The main difference in the VBA code that posts to the accounts payable ledger is that the code accesses the purchases journal instead of the sales journal. The code that manages postings from the cash payments and cash receipts journals is also similar to the code for the purchases and sales journals; the principal difference consists of which journals are accessed.

Getting a Current Liabilities Balance

Now that a structure for journals and ledgers has been defined, it's straightforward to move amounts for liabilities into the balance sheet (see Figure 4.10).

Figure 4.10
The balance sheet for Bell Books, June 2011 (liabilities and owner's equity) links directly to the general ledger's asset and liability accounts.



	A	B	C	D	E
1	Bell Books, Inc.	Balance Sheet, June 2011			
2					
3	**Liabilities and Owner's Equity**				
4	*Liabilities*				
5	Notes Payable		$ 13,000		
6	Accounts Payable		$ 12,446		
7					
8	*Owner's Equity*				
9	J. Bell, capital		$ 181,027		
10					
11	Total Liabilities and owner's equity		$ 206,473		
12					

Balance Sheet
Ready 100%

Notes payable, accounts payable, and owner's equity on the balance sheet are linked to their general ledger balances.

Although this example illustrates only two types of liabilities, accounts payable and notes payable, other types of liabilities exist that you might need to include in your ledger accounts:

- **Taxes payable—**You often need to estimate the taxes due on both income and salaries. You should consult an accountant or tax lawyer to determine the percentage rates to apply against your estimated income and estimated salaries. With these amounts, you can establish journal and ledger accounts that contain the proper estimates.
- **Salaries payable—**Often you pay employees' salaries before closing your books for a given accounting period. When that occurs, the days that elapse after payment is made and before the books are closed usually result in the accrual of salary amounts. You need to pay those accruals after closing the books. Journal and ledger accounts that accumulate these salaries help you keep track of these liabilities.
- **Interest payable—**Depending on whether a note is discounted, you might want to account for interest on the note on a periodic basis instead of on the date that the note is actually paid. An interest payable account allows you to accrue this liability over time.
- **Unearned revenue—**Sometimes a customer pays you for a product or service that won't be delivered until after the books are closed. In that case, the revenue represents

both an asset in the cash account and a liability (until delivery has occurred), which you can account for with an unearned revenue account.

- **Long-term debt—**If you have taken out a loan whose payable date is longer than a year from the date that the books are closed, you should keep this amount separate from the Current Liabilities section of the balance sheet. Its amount would be listed in a separate account, perhaps named `Long-Term Debt`, and listed in the Long-Term Liabilities section of the balance sheet.

Summary

This chapter has shown how you can use Excel to create an account structure in a workbook that contains both a general journal and special journals, and in a workbook that contains both a general ledger and subsidiary ledgers. As they occur, transactions are entered into the journals in chronological order. The transactions are then posted to the appropriate accounts in the ledger workbook. This accounts for the bulk of the work involved in creating both the Assets and the Liabilities sections of a balance sheet.

You also learned in this chapter some VBA techniques for automating the process of posting from journals to their associated ledger accounts and how to use Excel's pivot table facility to determine the current outstanding balance of an account. Those techniques, such as structuring a procedure and using `With` blocks, apply to many coding problems, not just posting from journals to ledgers.

Chapter 5, "Working Capital and Cash Flow Analysis," discusses an important technique for determining your company's financial position. Although the accrual method of accounting is, for most companies, the most accurate way to match revenue and expenses to determine profit, it tends to obscure the way the company handles its most liquid asset, cash. You will learn techniques for tracking the flow of cash and working capital in Chapter 5.

INDEX

Symbols

A

B

C

D

E

J-K

L

P

S

T

U

V

W

X-Y-Z