1 Everything You Ever Needed to Know About Spreadsheets but Were Too Afraid to Ask

This book assumes you have some experience working with spreadsheets. You won't need to be a spreadsheet expert, but if this is your first-time opening Excel, you might find this chapter a bit challenging. If that's you, I would recommend pairing this chapter with a *For Dummies* book or a beginner-level online class.

Even so, what follows in this chapter might still surprise the most seasoned, self-professed Excel pros. So, regardless of your Excel experience, this chapter should not be skipped! In the following pages, we'll describe a wide variety of Excel features that we'll use throughout the book.

Before moving forward, let's talk about the different versions of Excel out there and how they might affect you. First, everything in this book will work seamlessly in Excel 365 and Excel 2016 and beyond for Windows.

This book is going to use Excel 365 desktop for Windows. Excel 365 generally represents the latest versions of Excel, to which Microsoft pushes monthly updates. Some institutions still use enterprise versions of Excel such as Excel 2016 and Excel 2019. These versions will work, too. To ensure you are using the latest version of Excel, call the IT department at your school or your office and let them know you'd like to get the latest build. They'll know what you mean.

The story for Mac is a bit different. If you're on a Mac, some of keystrokes will be different. There are different icons and button locations. Power Query, Excel's data wrangling powerhouse, has fewer features in the Mac version as of this writing. Still, you should be able to get by just fine.

This book requires a desktop version of Excel. Though you can work in Excel through their online platform and through SharePoint, neither of these environments is suitable for this book as of this writing. That may change in time, but for now, assume everything from here on out is for Excel on the desktop.

Some Sample Data

> **NOTE**
>
> The Excel workbook used in this chapter, `Concessions.xlsx`, is available for download at the book's website at `www.wiley.com/go/datasmart2e`.

Let's start with some sample data.

You don't know this about me, but I love hot dogs. (Seriously, I have a Chicago-style hot dog tattoo.) A dream of mine is to one day run a hot dog stand. Let's say that dream happens, and I open up a concession stand to serve the sporting events of a local high school. If you've already opened `Concessions.xlsx`, let's start on the first tab, Basketball Game Sales.

At the end of each night, the point-of-sale system spits out the day's takings. It looks like in Figure 1.1.

	A	B	C	D	E
1	**Item**	**Category**	**Price**	**Profit**	**Actual Profit**
2	Beer	Beverages	$ 4.00	50%	$ 2.00
3	Hamburger	Hot Food	$ 3.00	67%	$ 2.00
4	Popcorn	Hot Food	$ 5.00	80%	$ 4.00
5	Pizza	Hot Food	$ 2.00	25%	$ 0.50
6	Bottled Water	Beverages	$ 3.00	83%	$ 2.50
7	Hot Dog	Hot Food	$ 1.50	67%	$ 1.00
8	Chocolate Dipped Cone	Frozen Treat	$ 3.00	50%	$ 1.50
9	Soda	Beverages	$ 2.50	80%	$ 2.00
10	Chocolate Bar	Candy	$ 2.00	75%	$ 1.50
11	Hamburger	Hot Food	$ 3.00	67%	$ 2.00
12	Beer	Beverages	$ 4.00	50%	$ 2.00
13	Hot Dog	Hot Food	$ 1.50	67%	$ 1.00
14	Licorice Rope	Candy	$ 2.00	50%	$ 1.00
15	Chocolate Dipped Cone	Frozen Treat	$ 3.00	50%	$ 1.50
16	Nachos	Hot Food	$ 3.00	50%	$ 1.50
17	Pizza	Hot Food	$ 2.00	25%	$ 0.50
18	Beer	Beverages	$ 4.00	50%	$ 2.00
19	Soda	Beverages	$ 2.50	80%	$ 2.00
20	Beer	Beverages	$ 4.00	50%	$ 2.00

Figure 1.1: Concession stand sales

This data is laid out in *tabular format*. This is likely something you're very familiar with. In Excel it's made up of *rows*, *columns*, and *cells*.

Some areas of data science may call these by different names. For instance, a row might be called a *record*, *observation*, or *tuple*. A column might be called a *field*, *feature*, or *dimension*. In truth, it doesn't matter what you call them so long as you use them well. However, you should take note that those around you might use different terms depending upon their field.

Accessing Quick Descriptive Statistics

Excel has the ability to instantly provide summary statistics—such as average, sum, min, and max—in the status bar. However, most of these measures aren't enabled by default. You'll likely want to refer to these continuously along your data journey.

To see what I mean, select cell E2 and then press Ctrl+Shift+Down (⌘+Shift+Down on a Mac). This will automatically highlight the data region of the entire column, spanning from E2:E200.

TIP

If you're the type who loves keyboard shortcuts, my friend, David Bruns, has put together a very handy list of shortcuts for both Mac and PC on his website, Excel Jet. See `https://exceljet.net/shortcuts`.

Look at the lower-right portion of your status bar. It should show an average of $1.79. Right-click the average label in your status bar, and you'll see multiple measures you can select (see Figure 1.2). Go ahead and select Average, Count, Numerical Count, Minimum, Maximum, and Sum. Once complete, you'll see they're now all available in the status bar. You'll appreciate having these measures at a moment's glance.

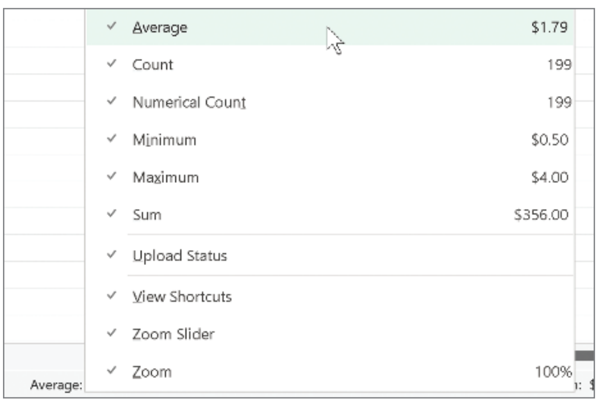


Figure 1.2: When you right-click the status bar, you have the option to have additional descriptive statistics reported to you. Select all of them.

Excel Tables

Perhaps you have experience with Excel formulas. You know, for instance, we could place a formula like `=AVERAGE(E2:E200)` in a blank cell to find the average.

Unfortunately, the cell reference (E2:E200) inside the `AVERAGE` formula is a bit of problem for the data scientist. What if we want to add 50 records? We would have to remember to update the `AVERAGE` formula to the new cell address of E2:E250. What if we moved the data from column E to column F? We would *again* have to ensure the `AVERAGE` formula pulls from F2:F250. And when you think about it, what does E2:E200 or F2:F250 really tell us about the data it represents?

You may have accepted that clunky formulas and misaligned references are just part of Excel. But I'm here to tell you there's a better way. Excel tables were created to meet the challenges described.

To apply an Excel table, place the cursor anywhere inside the data region. On the Insert tab, click Table (see Figure 1.3).

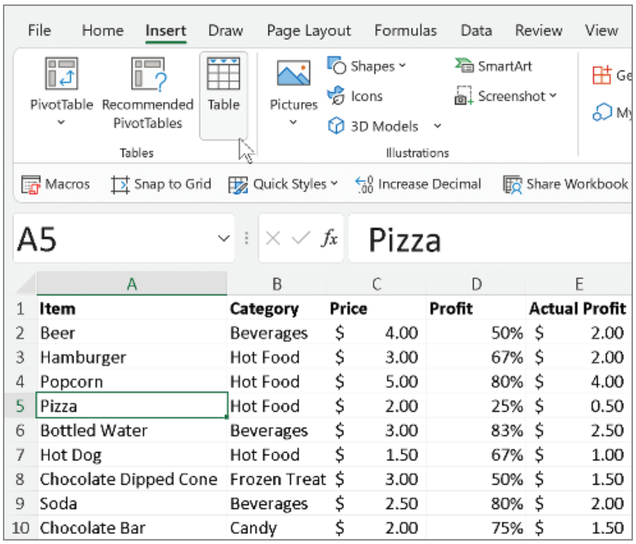
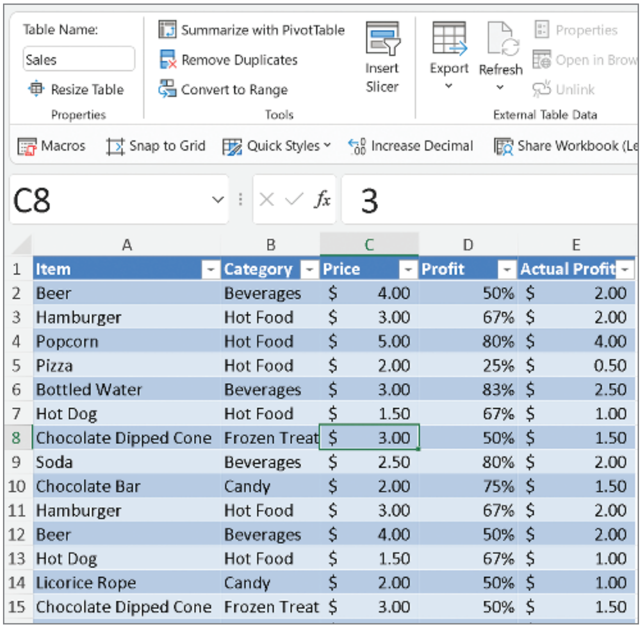


Figure 1.3: To insert an Excel table, place your cursor anywhere in the table region. Then, on the Insert tab, click the Table button.

In the Create Table dialog box, ensure that My Table Has Headers is selected, and then click OK.

You have now applied an Excel table. Your screen should look like in Figure 1.4.



	A	B	C	D	E
1	Item	Category	Price	Profit	Actual Profit
2	Beer	Beverages	$ 4.00	50%	$ 2.00
3	Hamburger	Hot Food	$ 3.00	67%	$ 2.00
4	Popcorn	Hot Food	$ 5.00	80%	$ 4.00
5	Pizza	Hot Food	$ 2.00	25%	$ 0.50
6	Bottled Water	Beverages	$ 3.00	83%	$ 2.50
7	Hot Dog	Hot Food	$ 1.50	67%	$ 1.00
8	Chocolate Dipped Cone	Frozen Treat	$ 3.00	50%	$ 1.50
9	Soda	Beverages	$ 2.50	80%	$ 2.00
10	Chocolate Bar	Candy	$ 2.00	75%	$ 1.50
11	Hamburger	Hot Food	$ 3.00	67%	$ 2.00
12	Beer	Beverages	$ 4.00	50%	$ 2.00
13	Hot Dog	Hot Food	$ 1.50	67%	$ 1.00
14	Licorice Rope	Candy	$ 2.00	50%	$ 1.00
15	Chocolate Dipped Cone	Frozen Treat	$ 3.00	50%	$ 1.50

Figure 1.4: Concession stand data with an Excel table applied

Whenever you create a new table, the Table Design ribbon tab appears, allowing you to interact with the table. As a first step for working with Excel tables, give the table a good name.

Excel will attempt to name the table for you with names like Table1, Table2, and mysteriously, Table1_2. You should never accept these default names (it's tacky!), but instead set a proper name reflecting the underlying dataset.

In the upper-left corner of the Table Design tab you can set the table's name. In Figure 1.4, I've set it to **Sales**. You'll quickly see why this is important.

Tables provide tons of features, akin to the data frames of Python and R that make doing data science in Excel that much easier. For one, as you scroll down an Excel table, the normal alphabetical column headers are replaced with the table's fields. This allows you to work with a table and know which column you're working with without freezing the top row. Take a look at Figure 1.5.

Filtering and Sorting

Tables have filtering and sorting already baked in (no need to apply the filtering feature on the Home or Data tab). For instance, if I want to simply look at the sales of hot dogs, I can filter the Item column by clicking the drop-down button in the header and selecting the item of interest (see Figure 1.6).

	Item	Category	Price		Profit	Actual Profit	
2	Beer	Beverages	$	4.00	50%	$	2.00
3	Hamburger	Hot Food	$	3.00	67%	$	2.00
4	Popcorn	Hot Food	$	5.00	80%	$	4.00
5	Pizza	Hot Food	$	2.00	25%	$	0.50
6	Bottled Water	Beverages	$	3.00	83%	$	2.50
7	Hot Dog	Hot Food	$	1.50	67%	$	1.00
8	Chocolate Dipped Cone	Frozen Treat	$	3.00	50%	$	1.50
9	Soda	Beverages	$	2.50	80%	$	2.00
10	Chocolate Bar	Candy	$	2.00	75%	$	1.50
11	Hamburger	Hot Food	$	3.00	67%	$	2.00
12	Beer	Beverages	$	4.00	50%	$	2.00
13	Hot Dog	Hot Food	$	1.50	67%	$	1.00
14	Licorice Rope	Candy	$	2.00	50%	$	1.00
15	Chocolate Dipped Cone	Frozen Treat	$	3.00	50%	$	1.50

Figure 1.5: Tables will replace the column headers with the column names. This means you can work with the table without having to freeze the header row.

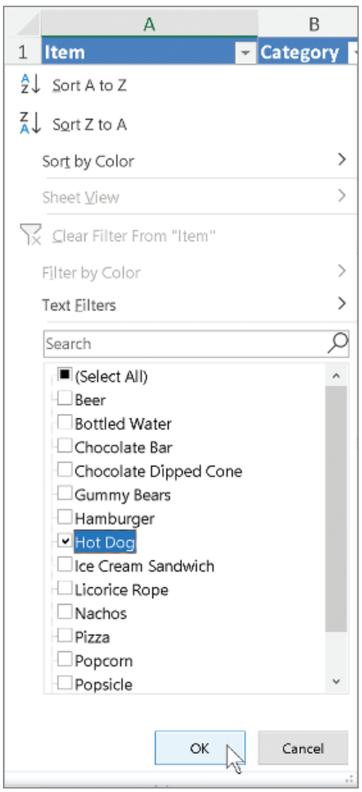


Figure 1.6: Tables already have filters bulit-in. To filter a specific column, press the gray drop-down button next to the column header.

To clear the filter, again, simply click the drop-down in the header of Items and select Clear Filter From Item.

Before moving on, take a look at the different options available in the drop-downs. Note that there are many ways to sort and filter your data.

Table Formatting

Excel's default formatting of tables is hideous. (Sorry, that's just how I feel.) I can't abide by the tacky overcolored defaults. For ease of reading the data in your tables, my recommendation is to use the Table style menu and select a table style from the Light category that does not include banded rows (see Figure 1.7).

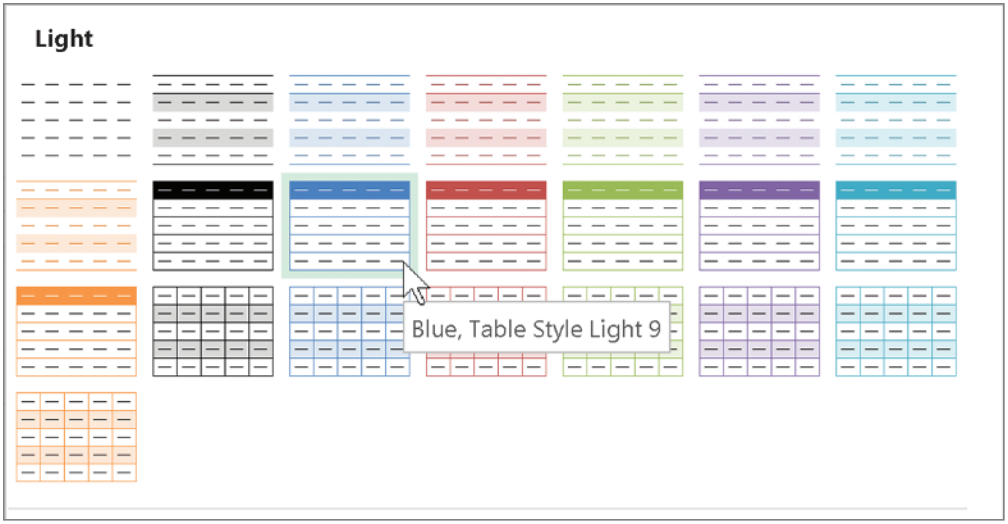


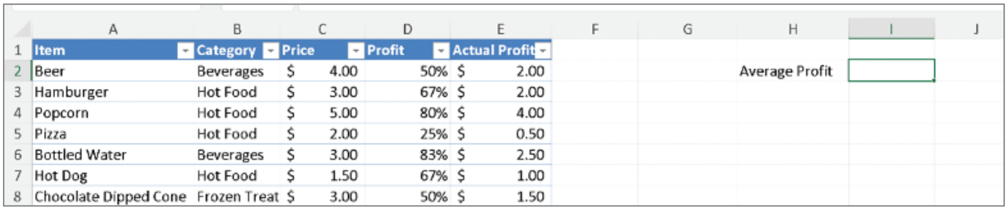
Figure 1.7: The default table formatting is overly colorful and distracting. However, the lighter styles will make your data easy to read and comprehend.

Going forward, I will always adjust the table style to one of the easier-to-read options even when I haven't directed you to do so.

Structured References

Structured references are the single most important feature of Excel tables. Remember, at the start of the chapter, we focused on the issues of using cell references in formulas. Let's see how Excel tables switch things up.

In cell H2 on the Basketball Game Sales tab, I've created a label called Average Profit. To the right of it, in cell I2, I'll write my Average formula. You can set this up like in Figure 1.8.



	A	B	C	D	E	F	G	H	I	J
1	Item	Category	Price	Profit	Actual Profit					
2	Beer	Beverages	$ 4.00	50%	$ 2.00			Average Profit		
3	Hamburger	Hot Food	$ 3.00	67%	$ 2.00					
4	Popcorn	Hot Food	$ 5.00	80%	$ 4.00					
5	Pizza	Hot Food	$ 2.00	25%	$ 0.50					
6	Bottled Water	Beverages	$ 3.00	83%	$ 2.50					
7	Hot Dog	Hot Food	$ 1.50	67%	$ 1.00					
8	Chocolate Dipped Cone	Frozen Treat	$ 3.00	50%	$ 1.50					

Figure 1.8: The label "Average Profit" has been added to H2. The cell I2 is where we'll add our formula.

In cell I2, start typing `=AVERAGE` to begin your Average formula. Instead of using a cell reference, we'll use a reference to our table, which I have named Sales.

Note that as you begin to type **Sales**, the name Sales appears in the IntelliSense prompt with a table icon next to it. Once you have completed typing Sales (or selected it from the IntelliSense drop-down), the entire table is highlighted, reflecting that you are now referring to the table. You can see this in Figure 1.9.

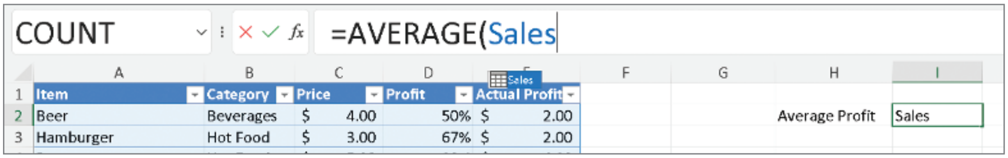


Figure 1.9: The Excel table was named Sales. As you type **Sales** into the formula bar, Excel recognizes that it's an Excel table and provides you with a direct, named reference.

Now that you have access to the table, you'll want to access a specific field. After the table name is entered, press the left square bracket ([) on your keyboard. This opens the table to allow you to select the field of choice (see Figure 1.10). Let's select Actual Profit. You can either type this field (make sure to add a right square bracket] at the end) or select it from the IntelliSense drop-down. Once complete, Excel will highlight the column accordingly, like in Figure 1.11.

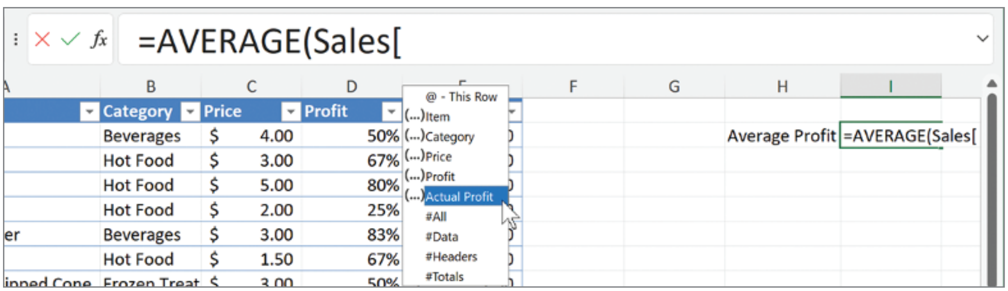
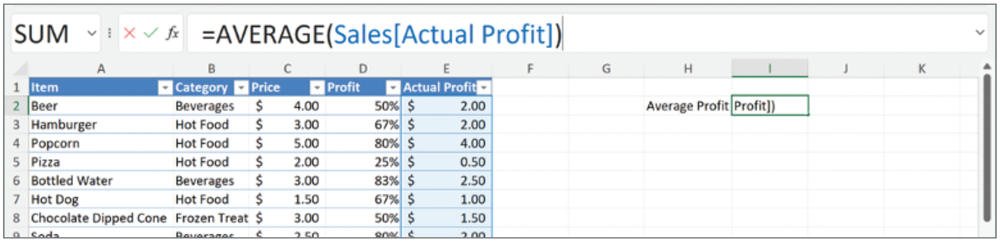


Figure 1.10: Once you've typed in the table name and added a left square bracket, you will have access to every field and additional table properties.



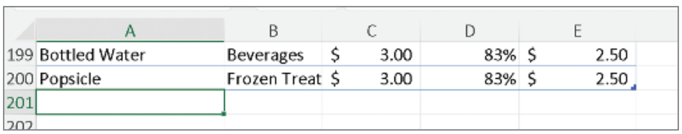
	Item	Category	Price	Profit	Actual Profit
2	Beer	Beverages	$ 4.00	50%	$ 2.00
3	Hamburger	Hot Food	$ 3.00	67%	$ 2.00
4	Popcorn	Hot Food	$ 5.00	80%	$ 4.00
5	Pizza	Hot Food	$ 2.00	25%	$ 0.50
6	Bottled Water	Beverages	$ 3.00	83%	$ 2.50
7	Hot Dog	Hot Food	$ 1.50	67%	$ 1.00
8	Chocolate Dipped Cone	Frozen Treat	$ 3.00	50%	$ 1.50

Figure 1.11: When referring to a table's column field in Excel, you will see the selected column highlighted like a cell reference.

Once you're happy with the formula, press Enter. The average profit spend is 1.79ish. To format this cell to a dollar amount, you can click the dollar sign icon ($) on the Home ribbon tab in the Number group to turn it into a two-decimal dollar amount.

Now, I want to draw your attention to the magic of tables and structured references. Structured references can grow and shrink based on how much data is contained in the table without having to adjust the formulas that use them. Let's see this in action.

Scroll all the way down to the bottom of the table and place your cell in the leftmost cell at the bottom of the table. In this case, that's cell A201 (see Figure 1.12).



	A	B	C	D	E
199	Bottled Water	Beverages	$ 3.00	83%	$ 2.50
200	Popsicle	Frozen Treat	$ 3.00	83%	$ 2.50
201					

Figure 1.12: To automatically add information to an Excel table, place your cursor in the cells directly under the last record and add your new data.

Let's add a new record in cell A201 by typing **Popsicle**. Then, press Enter. Note that the table has now grown to consume this new record. In the Profit field, add a large dollar amount like $2000. Your table should now look like in Figure 1.13.

	Item	Category	Price	Profit	Actual Profit
190	Popsicle	Frozen Treat	$ 3.00	83%	$ 2.50
191	Gummy Bears	Candy	$ 2.00	50%	$ 1.00
192	Pizza	Hot Food	$ 2.00	25%	$ 0.50
193	Bottled Water	Beverages	$ 3.00	83%	$ 2.50
194	Popcorn	Hot Food	$ 5.00	80%	$ 4.00
195	Beer	Beverages	$ 4.00	50%	$ 2.00
196	Pizza	Hot Food	$ 2.00	25%	$ 0.50
197	Popsicle	Frozen Treat	$ 3.00	83%	$ 2.50
198	Chocolate Bar	Candy	$ 2.00	75%	$ 1.50
199	Bottled Water	Beverages	$ 3.00	83%	$ 2.50
200	Popsicle	Frozen Treat	$ 3.00	83%	$ 2.50
201	Popsicle				$2,000

Figure 1.13: When you add new data to the bottom of a table, it will automatically grow to consume the new information. However, you won't need to change any of the formulas that refer to it.

Don't worry if the dollar amount is so large as to be an outlier; we'll end up deleting this record in a second. I simply want to demonstrate that the average actual profit in cell I2 has changed dramatically to 11.78ish! The new data has become part of the table. And the formulas used on it do not need to be changed as data is added or deleted. The best part is that you don't need to type these new records—you can simply copy and paste in new data to the bottom of the table as it becomes available.

To delete the record, simply right-click the row label, 201, and select Delete. Do this and then double-check that I2 has changed back to its previous amount.

Adding Table Columns

In addition to being able to add records easily to the bottom of the table, you can also add custom column fields directly to the right of the table. Say, for instance, you would like to see which items generated more than $2 in actual profit.

In cell F1 (the topmost cell adjacent to the right of the table), type **>$2.00**. This will be your new column header. See how the table automatically grows to incorporate the new header! The best part: this new header will also appear in the IntelliSense drop-down.

Now, in cell F2, type = and then select the cell directly to the left of it. When you do this, Excel automatically fills in the formula with the structured reference `[@[Actual Profit]]`. Complete the formula `=[@[Actual Profit]>2`, like in Figure 1.14. Press Enter.

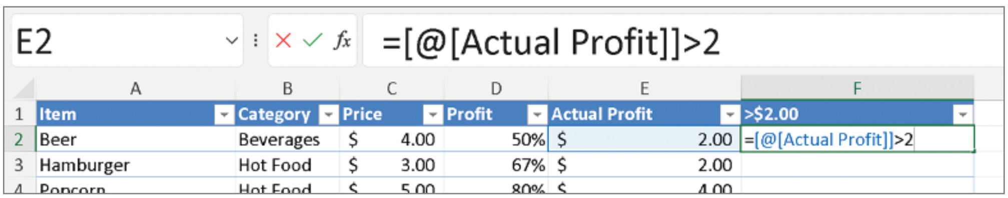


Figure 1.14: Structured references allow you to create column calculations that work on data as records and not cell references.

Presto! Note what happens—the formula automaticlaly populates the entire length of the table. No dragging necessary!

I know what you're thinking: *what's with the @?* This symbol means "at this row." In other words, rather than working on the entire column range (which we did when calculating the average), the @ symbol directs the table formula to work only on the cells in that row. This makes sense when you think about it: each record contains a particular piece of information, and you might want to know information only with respect to the fields of that record.

To delete this column, simply right-click the column header of Column F and select Delete.

Excel tables are a wonderful tool in Excel, and as you can see, they fix many of the issues described at the start of this section. For the rest of this book, we'll work with a mix of cell references and the structured references of tables. We'll choose the one that makes the most sense given the sitaution. As we go through the book, you'll see how best to use each type.

TIP

I would love to further extol Excel tables, but unfortunately there's a page limit from the publisher. If you'd like to master Excel tables, see these books:

- *Excel Tables: A Complete Guide for Creating, Using and Automating Lists and Tables* by Zack Barresse and Kevin Jones (Holy Macro Books, 2014).
- *Excel 2019 Bible* by Michael Alexander, Richard Kusleika, and John Walkenbach (Wiley, 2018)

Lookup Formulas

Lookup formulas are very important in data analysis. They allow us to find out information about one or more records as well as join information from different tables.

There are multiple lookup functions in Excel. However, the ones we're most concerned with are `VLOOKUP`, `INDEX/MATCH`, and the venerable `XLOOKUP`. To understand each function, let's understand why they exist and the workflows they are designed to solve.

VLOOKUP

Let's start with `VLOOKUP`. `VLOOKUP`s exist because of a specific data workflow. For instance, sometimes you will receive data that looks like in Figure 1.15.

Key	Value1	Value2	Value3
4524	...	...	...
2352	...	...	...
1834	...	...	...
3271	...	...	...

Figure 1.15: A table layout where the unique identifier the "key" is the leftmost column

The leftmost column is the *key*. A key column contains a unique identification for each record. Keys can be anything from a simple record number to a Social Security number, student ID, or SKU. As you'll see, `VLOOKUP` requires a table of this layout.

The breakdown of `VLOOKUP` looks like this:

```
VLOOKUP(what you're looking for, the entire table of where you're looking, the
column(s) of interest, whether to perform an exact or approximate match)
```

Let's take a look at each of these:

- **`what you're looking for`**: *This is your search term. In the case of* `VLOOKUP`, *it'll be the specific key you're looking for.*
- **`The entire table of where you're looking`**: *This is a reference to the entire table where you're looking, including the key column.*
- **`the column(s) of interest`**: *This is where you identify the columns you'd like to return.*
- **`whether to perform an exact or approximate match`**: `VLOOKUP` *allows you to either match the key exactly or approximately match it. In this book, we won't be dealing with approximate match. So, whenever you use* `VLOOKUP` *from examples in this book, make sure the last parameter is always a zero or a* `FALSE` *(which mean the same thing in Excel). This tells Excel we always want exact matching.*

`VLOOKUP` works by matching the search term to the key along the table's left side. Once the match is found, the key effectively "unlocks" the record. You can tell `VLOOKUP` what to return from the record by specifying the desired return column numbers in the column of interest parameter.

Enough talk—let's see it in Excel. In the Concessions file, select the Calories Worksheet tab. Note there is already a table there named NutritionInfo. In Figure 1.16, I've implemented the following formula in cell B19:

```
=VLOOKUP("Hot Dog", NutritionInfo, 2, FALSE)
```

As you can see, `Hot Dog` is the unique key. Once the `VLOOKUP` finds the key in the table, it returns the value associated with the column number provided. In this case, the `2` passed into the third parameter of the formula indicates we want the information from the second column.

	A	B	C	D	E	F
1	Item	Calories				
2	Beer	200				
3	Bottled Water	0				
4	Chocolate Bar	255				
5	Chocolate Dipped Cone	300				
6	Gummy Bears	300				
7	Hamburger	320				
8	Hot Dog	265				
9	Ice Cream Sandwich	240				
10	Licorice Rope	280				
11	Nachos	560				
12	Pizza	480				
13	Popcorn	500				
14	Popsicle	150				
15	Soda	120				
16						
17						
18	How many calories are in a Hot Dog?					
19	**VLOOKUP**	265	=VLOOKUP("Hot Dog", NutritionInfo, 2,FALSE)			

Figure 1.16: A `VLOOKUP` has been implemented in cell B19. In this case, the formula looks for the unique record associated with `Hot Dog` and returns information in the cell (the second column) to its right.

INDEX/MATCH

`VLOOKUP` solves one type of workflow for a specific data layout. But sometimes you're looking for something different—for instance, you want to find the key associated with a specific data point. Figure 1.17 shows a representation of this potential workflow.

Key	Value1	Value2	Value3
4524	...	...	...
2352	... ? ←	known info	→ ? ...
1834	...	...	...
3271	...	...	...

Figure 1.17: In some workflows, you know a value stored in another field, and you would like to know the associated key and other data from other columns.

In this case, you could use the function combination of `INDEX/MATCH`—so named because it combines the two functions together to achieve its aim. To understand how it works, let's take a look at these two functions:

```
= MATCH(what you're searching for, where you want to look, the match type)
```

- **`what you're searching for`**: *This is the search parameter.*
- **`where you want to look`**: *This is the row or column in which you want to look.*
- **`the match type`**: *This is for whether you want it to be an exact match or an approximate match. In this book, we'll only be dealing with exact matches, so simply set that to* `0` *or* `FALSE`.

Importantly, `MATCH` returns the record location where it finds its match.

This works out really well for the associated function, `INDEX`. `INDEX` takes in a row, column, or table of data, and let's you supply the coordinates of where you want to look. In that way, it functions like a reverse lookup. Let's take a look at `INDEX`:

```
=INDEX(table or column of interest, row of interest, column of interest)
```

- **`table or column of interest`**: *This is a cell or table reference to a table or column.*
- **`row of interest`**: *If you supply a 4 here, the fourth record within the column or table is returned. In this case, we'll place the results of our `MATCH` function here since that returns the record location.*
- **`column of interest`**: *This is an optional parameter. In this case, we're using data of only one dimension (rows or columns), so we don't need this. However, if you would like to look up values across an entire table, this parameter is very useful.*

Once we know where the record is located using `MATCH`, we can figure out the key to the associated person using `INDEX`. Let's see this in action in Excel.

Let's say we want to know which item has the most calories. In cells `B22:24`, I have broken down the steps to complete this formula (see Figure 1.18). In `B22`, we find the maximum calories from the `NutritionInfo[Calories]` column. In `B23`, we use `MATCH` to find where this value is located. Excel returns a 10. Finally, in `B24`, we use `INDEX` to look up the 10th record from NutritionInfo's Item column field. In `B25`, I've combined the formula into one.

```
=INDEX(NutritionInfo[Item], MATCH(MAX(B22), NutritionInfo[Calories], 0))
```

	A	B	C
1	Item	Calories	
2	Beer	200	
3	Bottled Water	0	
4	Chocolate Bar	255	
5	Chocolate Dipped Cone	300	
6	Gummy Bears	300	
7	Hamburger	320	
8	Hot Dog	265	
9	Ice Cream Sandwich	240	
10	Licorice Rope	280	
11	Nachos	560	
12	Pizza	480	
13	Popcorn	500	
14	Popsicle	150	
15	Soda	120	
16			
17			
18	How many calories are in a Hot Dog?		
19	**VLOOKUP**	265	=VLOOKUP("Hot Dog", NutritionInfo, 2,FALSE)
20			
21	Which item has the most calories?		
22	MAX	560	=MAX(NutritionInfo[Calories])
23	MATCH	10	=MATCH(B22,NutritionInfo[Calories],0)
24	INDEX	Nachos	=INDEX(NutritionInfo[Item],B23)
25	**INDEX/MATCH**	Nachos	=INDEX(NutritionInfo[Item], MATCH(MAX(NutritionInfo[Calories]), NutritionInfo[Calories], 0))
26			

Figure 1.18: `INDEX/MATCH` has been across helper cells `B22:B24`. It's also implemented as one formula in B25.

TIP

Combining lots of functions together is hard to read, so press Alt+Enter (Option+Return on a Mac) while in the formula bar to go to a new line. It will look like the following code. We'll be doing that for long formulas in this book.

```
=INDEX(
    NutritionInfo[Item],
    MATCH(MAX(B21),NutritionInfo[Calories],0)
)
```

USE `LET` TO AVOID REPEATING MULTIPLE COLUMNS

Take a look at the `INDEX`/`MATCH` function we wrote:

```
=INDEX(NutritionInfo[Item], MATCH(MAX(NutritionInfo[Calories]),
NutritionInfo[Calories], 0))
```

Notice that we have to repeat the column `NutritionInfo[Calories]` for this problem. It's not uncommon to have to repeat multiple references and functions in Excel, especially when working with complicated datasets.

If you have the most updated version of Excel 365, you can use the `LET` function to help you stay more organized. `LET` allows you to declare a variable that exists only within the formula. For instance, we could use this instead to make the formula more readable:

```
= LET(
    CaloriesColumn, NutritionInfo[Calories],
    INDEX(NutritionInfo[Item],
        MATCH(MAX(CaloriesColumn),CaloriesColumn, 0)
    )
  )
```

Notice how we replace `NutritionInfo[Calories]` info with the `CaloriesColumn` variable. We'll be using `LET` quite a bit in this book.

XLOOKUP

For many years, `VLOOKUP` and `INDEX`/`MATCH` were the mainstays of Excel developers. But as you can see, both have issues: with a `VLOOKUP` you need to know the column index of interest. And `INDEX`/`MATCH` just feels needlessly complicated. As a result, Microsoft released a new function that can do what `VLOOKUP` and `INDEX`/`MATCH` do but better.

You might be wondering why I'm even bringing up `VLOOKUP` and `INDEX`/`MATCH` if a better alternative is out there. The answer is that while `XLOOKUP` is considered generally available, not everyone has it yet. XLOOKUP is technically slower than INDEX/MATCH, so sometimes you might prefer the latter to save calculation time.

For those with XLOOKUP, this section is for you. For those who don't have it yet (you'll know you don't have it when it doesn't work), make sure you're using the latest version of Excel or Office 365.

XLOOKUP is effectively the best of both worlds. While there are many options in the XLOOKUP function, you really need to know only three things to get started: what you're looking for, the column of where you want to look, and the return column where the cell you return resides. Take a look at the representation in Figure 1.19.

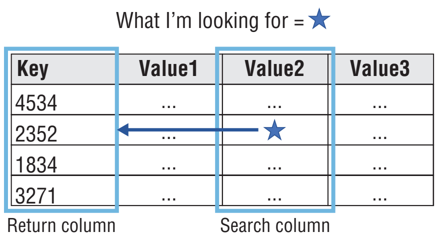


Figure 1.19: The elements required to implement an XLOOKUP

Let's implement XLOOKUP in Excel. Take a look at cell B28 in Figure 1.20. In this case, the goal is again to find the item with the most calories, this time using XLOOKUP. The following is the formula we use:

```
=XLOOKUP(MAX(NutritionInfo[Calories]), NutritionInfo[Calories],
NutritionInfo[Item])
```

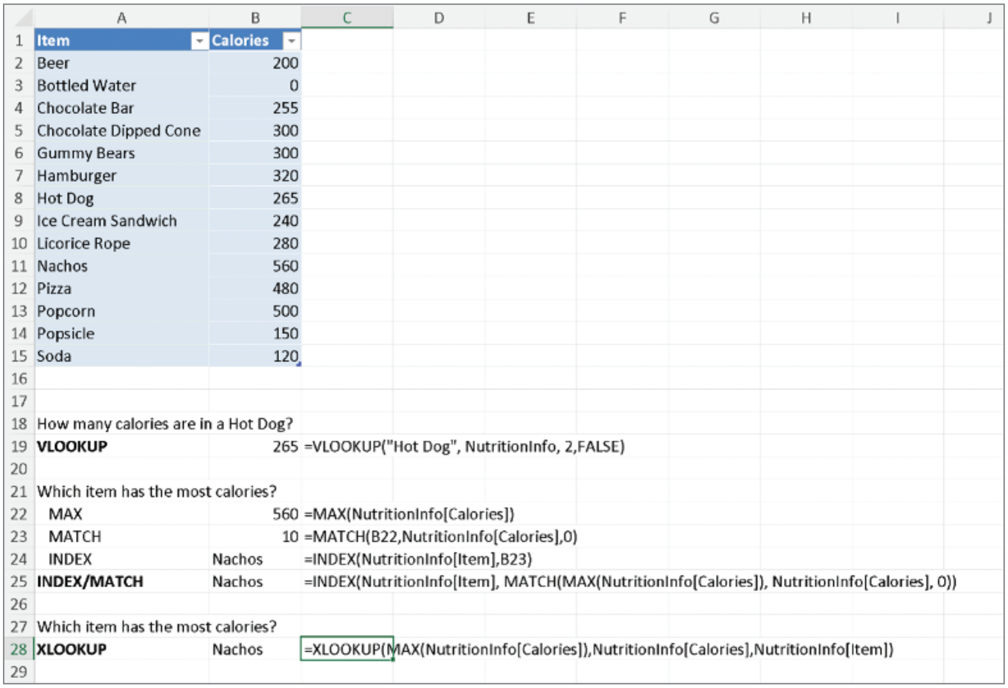
Notice that XLOOKUP contains many of the same elements of the INDEX/MATCH solution—it's just more condensed.

Once you understand lookup formulas, you're ready to move on to PivotTables.

PivotTables

What if you wanted to know the total counts of each item type you sold? Or you wanted to know revenue totals by item?

These questions are akin to "aggregate" or "group by" queries, which are a key element of descriptive analytics. The goal of descriptive analytics is to take large columns of data and push them through aggregator functions to get just a few descriptive numbers. PivotTables help us do this by grouping together different elements of a dataset and applying some type of aggregation function like SUM, MIN, COUNT, and others.



	A	B	C
1	Item	Calories	
2	Beer	200	
3	Bottled Water	0	
4	Chocolate Bar	255	
5	Chocolate Dipped Cone	300	
6	Gummy Bears	300	
7	Hamburger	320	
8	Hot Dog	265	
9	Ice Cream Sandwich	240	
10	Licorice Rope	280	
11	Nachos	560	
12	Pizza	480	
13	Popcorn	500	
14	Popsicle	150	
15	Soda	120	
16			
17			
18	How many calories are in a Hot Dog?		
19	**VLOOKUP**	265	=VLOOKUP("Hot Dog", NutritionInfo, 2,FALSE)
20			
21	Which item has the most calories?		
22	MAX	560	=MAX(NutritionInfo[Calories])
23	MATCH	10	=MATCH(B22,NutritionInfo[Calories],0)
24	INDEX	Nachos	=INDEX(NutritionInfo[Item],B23)
25	**INDEX/MATCH**	Nachos	=INDEX(NutritionInfo[Item], MATCH(MAX(NutritionInfo[Calories]), NutritionInfo[Calories], 0))
26			
27	Which item has the most calories?		
28	**XLOOKUP**	Nachos	=XLOOKUP(MAX(NutritionInfo[Calories]),NutritionInfo[Calories],NutritionInfo[Item])
29			

Figure 1.20: `XLOOKUP` is implemented in cell `B28`. Notice the `XLOOKUP` formula uses many of the same cell references as `INDEX/MATCH` but gets to the answer in fewer steps.

To apply a PivotTable, go back to your Basketball Game Sales sheet and click into the Sales table. On the Insert tab, click PivotTable. Click OK to create a new sheet for the PivotTable.[1]

In this new sheet, the PivotTable Builder will appear on the right side of the screen. The builder allows you to take the columns from the original selected data and use them as report filters, column and row labels for grouping, or values for aggregating.

Now, say you wanted to know total revenue by item. To get at that, you drag the Item tile in the PivotTable Builder into the Rows section and the Price tile into the Values section. This means that you'll be operating on revenue grouped by item name. This is shown in Figure 1.21.

[1] A PivotTable joke: A PivotTable walks into a bar. The bar tender says, "should I start a new tab?"

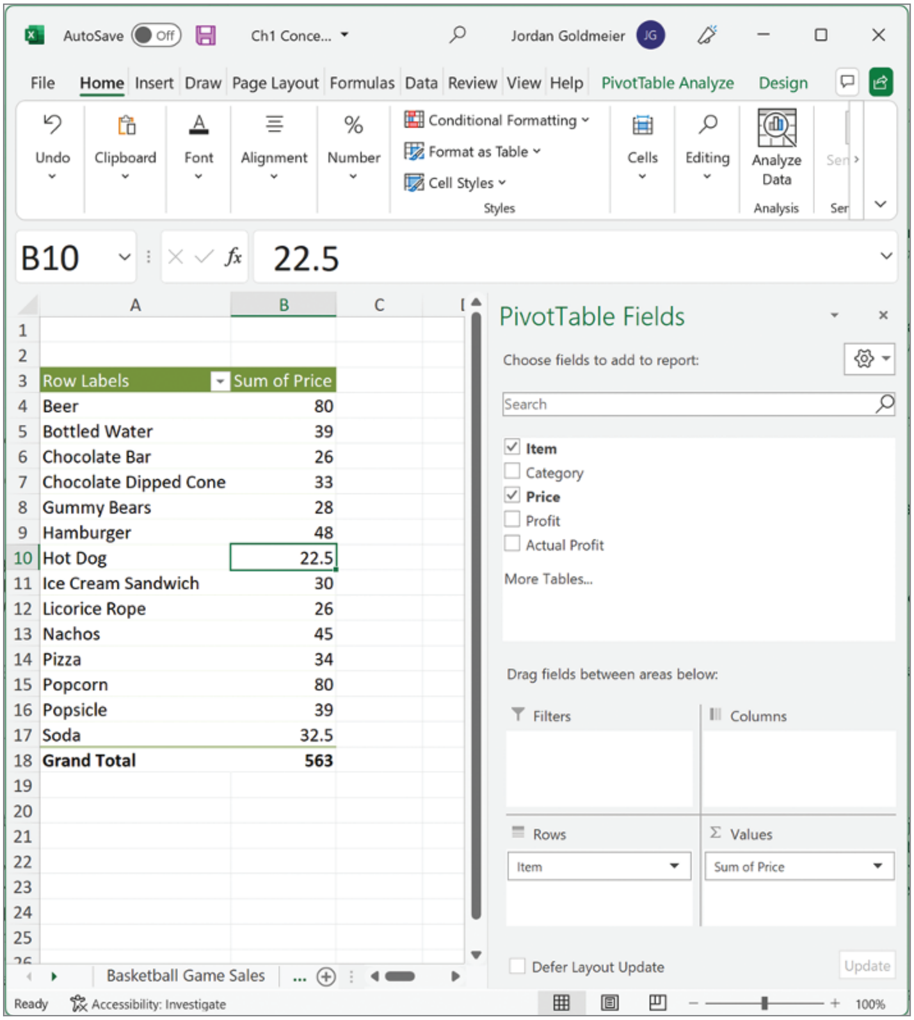
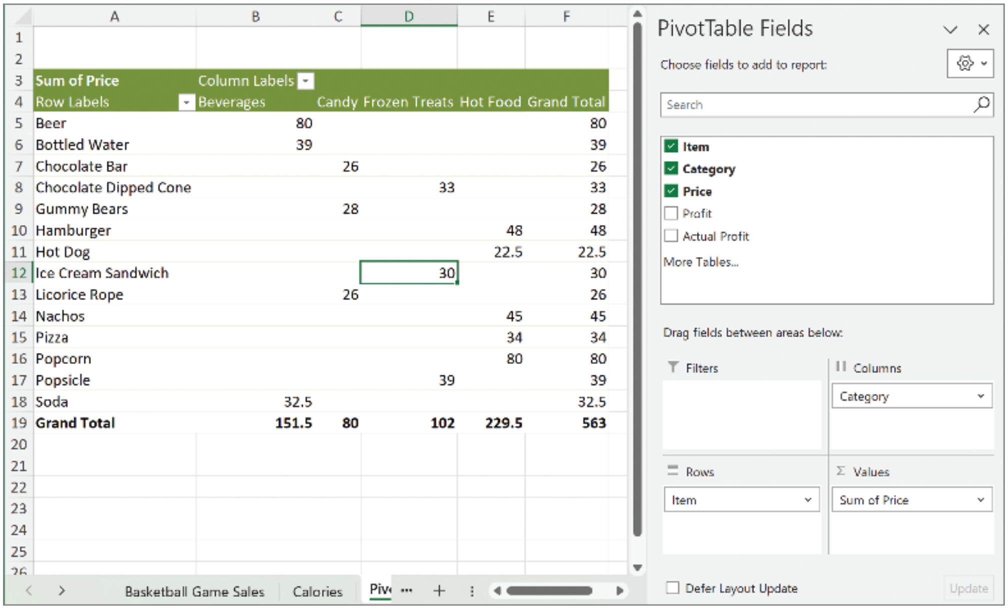


Figure 1.21: The PivotTable Builder and a count of sales by item

What if you wanted to break out these sums by category? Simply drag the Category tile into the Columns section of the builder. This gives the table shown in Figure 1.22. Note that the PivotTable in the figure automatically totals up rows and columns for you.

If you want to ever get rid of something from the table, just uncheck it or grab the tile from the section it's in and drag it out of the builder as if you were tossing it away. Go ahead and drop the Category tile. Feel free to swap in various row and column labels until you get the hang of what's going on. For instance, try to get a total calorie count sold by category using a PivotTable. When you're done with the PivotTable, change the worksheet tab name to "Pivot" to stay organized.



Sum of Price	Column Labels				
Row Labels	Beverages	Candy	Frozen Treats	Hot Food	Grand Total
Beer	80				80
Bottled Water	39				39
Chocolate Bar		26			26
Chocolate Dipped Cone			33		33
Gummy Bears		28			28
Hamburger				48	48
Hot Dog				22.5	22.5
Ice Cream Sandwich			30		30
Licorice Rope		26			26
Nachos				45	45
Pizza				34	34
Popcorn				80	80
Popsicle			39		39
Soda	32.5				32.5
Grand Total	**151.5**	**80**	**102**	**229.5**	**563**

Figure 1.22: Revenue by item and category

Using Array Formulas

Array formulas in Excel might be new to you, but they've been around for a while and have gone through some major evolutions since the original *Data Smart* was published.

In the early years of Excel, array formulas required a multistep process to implement. In the last several years, however, Microsoft introduced dynamic array formulas that no longer require the complications of old. The original *Data Smart* featured the older method of implementing array formulas. However, your version of Excel likely has the newer implementation.

Still, because some Excel versions don't yet have dynamic array formulas, I will explain how to do both in the section that follows.

Let's say you owe a percentage of your sales to the athletic program. In the Concessions workbook, there is a tab called Fee Schedule. The Fee Schedule is how much from each item will go to support the local school's athletic program. Note that this data on this tab is listed horizontally. We'll need to implement the `TRANSPOSE` formula to make it vertical.

So, how much money do you owe for last night's game? To answer that question, you need to multiply the total revenue of each item by the fee schedule and then sum them all up. In other words, you need to find the dot product between the two.

There's a great formula for this operation that will do all the multiplication and summation in a single step called `SUMPRODUCT`. Go to the Revenue By Item sheet, and add a label called **Total Cut for Coach** in cell `D1`. In `D2`, determine the `SUMPRODUCT` of the revenue and the fees by adding this formula:[2]

```
=SUMPRODUCT(B2:B15,TRANSPOSE('Fee Schedule'!B2:O2))
```

If `D2` immediately results in an answer of $57.60, then you have dynamic array formulas already available to you—and there's nothing additional you need to do. If not (if you instead receive a `#VALUE!` Error), you'll need to take a few extra steps.

In the past, every formula in Excel returned one cell by default. Even `TRANSPOSE` returns the first value in the transposed array. Array formulas, on the other hand, return an *array of values* and not just a single value. Older versions of Excel still need to be told you want to work with an array of values instead of one single value.

All you need to do is when you're done typing the formula in D2, instead of pressing Enter, press Ctrl+Shift+Enter (⌘+Return on a Mac). This will instruct Excel you want to use an array formula.

Solving Stuff with Solver

Doing data science in Excel often requires that you form your problem into an *optimization problem*. In optimization modeling, you find the best choice by either maximizing or minimizing one or more *objectives*. In fact, many data science practices, whether that's artificial intelligence, data mining, or forecasting, are actually just some data prep plus a model-fitting step that's actually an optimization model.

In Excel, optimization problems are solved using an Add-In that ships with Excel called Solver. Throughout this book, we'll be using Solver and its freeware counterpart, OpenSolver, when the problem is too big for Excel.

In Windows, you can add Solver by selecting File ➢ Options ➢ Add-ins, choosing Excel Add-Ins in the Manage drop-down, and clicking the Go button. Select the Solver Add-In box and click OK.

On a Mac, you can add by selecting Tools ➢ Add-Ins and then selecting `Solver.xlam` from the menu.

A Solver button will appear in the Analysis section of the Data tab in every version (see Figure 1.23).

You did it! Now that Solver is installed, let's start with our first optimization problem based on the data we already have.

[2] Another Excel joke: What do Excel users put in their hair? SUMPRODUCT.

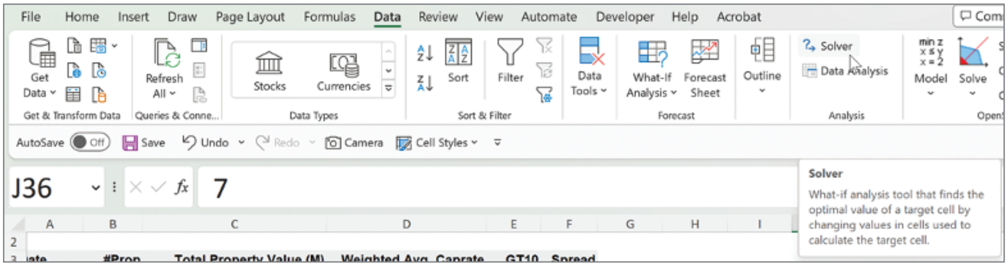


Figure 1.23: Solver appears on the far right of the Data tab.

Optimization problems always start with an objective. You are told you need 2,400 calories a day. So. . . *What's the fewest number of items you can buy from the snack stand to achieve that?* You could attempt to work this out in your head, but Solver can tell you!

To start, make a copy of the Calories sheet, name the sheet **Calories-Solver**. The easiest way to make a copy of a sheet is to simply right-click the tab you'd like to copy and select the Move or Copy menu. Next, click into the Excel table. Then from on the Table Design menu, click the Convert to Range button to convert the table back to a regular range of cells. Finally, highlight the range of cells and clear out the color formats. In this case, it's easier to work with Solver without the table. Next, delete everything that's not the calories table (rows 18-28). When you're finished, your sheet will look like in Figure 1.24.

We'll add the additional items shown in rows 16 and 17 Figure 1.24 in a moment.

	A	B	C	D	E
1	Item	Calories	How Many		
2	Beer	200			
3	Bottled Water	0			
4	Chocolate Bar	255			
5	Chocolate Dipped Cone	300			
6	Gummy Bears	300			
7	Hamburger	320			
8	Hot Dog	265			
9	Ice Cream Sandwich	240			
10	Licorice Rope	280			
11	Nachos	560			
12	Pizza	480			
13	Popcorn	500			
14	Popsicle	150			
15	Soda	120			
16		Total Items:	**0**	=SUM(C2:C15)	
17		Total Calories:	**0**	=SUMPRODUCT(B2:B15,C2:C15)	

Figure 1.24: Getting calorie and item counts set up

To get Solver to work, you need to provide it with a range of cells it can set with decisions. In this case, Solver needs to decide how many of each item to buy. So in Column C, next to the calorie counts, label the column **How many**. Solver will store its decisions to this column.

TIP

Excel considers blank cells to be 0s, so you needn't fill in these cells with anything to start. Solver will do that for you.

In cell `C16`, sum up the number of items to be bought above as follows:

```
=SUM(C2:C15)
```

And below that you can sum up the total calorie count of these items (which you'll want eventually to equal 2,400) using the `SUMPRODUCT` formula:

```
=SUMPRODUCT(B2:B15,C2:C15)
```

To the left of both of these cells add the labels **Total Items:** and **Total Calories:** respectively. This gives the initial sheet shown in Figure 1.24. Note I am including the `FORMULATEXT` function in column D to see the formulas I've implemented. I won't direct you to do this, but it helps keep track of everything as we go through the material.

Notice that we've just set up the problem to functionally work in Excel even as we haven't yet applied Solver. Sometimes the hardest part is creating the correct problem setup for Solver. Be prepared to make mistakes before you finally get the hang of it.

When you're ready, let's bring up the Solver window (see Figure 1.25). You can do this by clicking the Solver button on the Data tab (see Figure 1.23).

Problems like this are looking for four things: an objective cell, an optimization direction (minimization or maximization), some decision variables that can be changed by Solver, and some constraints. As you can see in Figure 1.25, the Solver window allows you to define these.

In your case, the objective is to minimize the total items in cell C16. The cells that Solver will alter are C2:C15. Figure 1.25 shows these settings implemented in the Solver Parameters dialog box. Go ahead and set those by click into each respective box and highlighting the appropriate cells into they match Figure 1.25. You'll also see two additional constraints that we haven't yet added. But we'll do that in a second.

One constraint will be that the total calories (C17) need to be equal to 2,400. We'll also have to constrain the cells in C2:C15 to be integer values—for instance, you can't buy 1.7 sodas.

To add the total calorie constraint, click the Add button and set C17 equal to 2,400, as shown in Figure 1.26. Use the middle drop-down to set the equals (=) parameter. When complete, hit the Add button (see Figure 1.26).

Similarly, add a constraint setting C2:C15 to be integers, as shown in Figure 1.27.

Click OK. If you've done everything right, your Solver window should look like in Figure 1.25. Go ahead and double-check it.

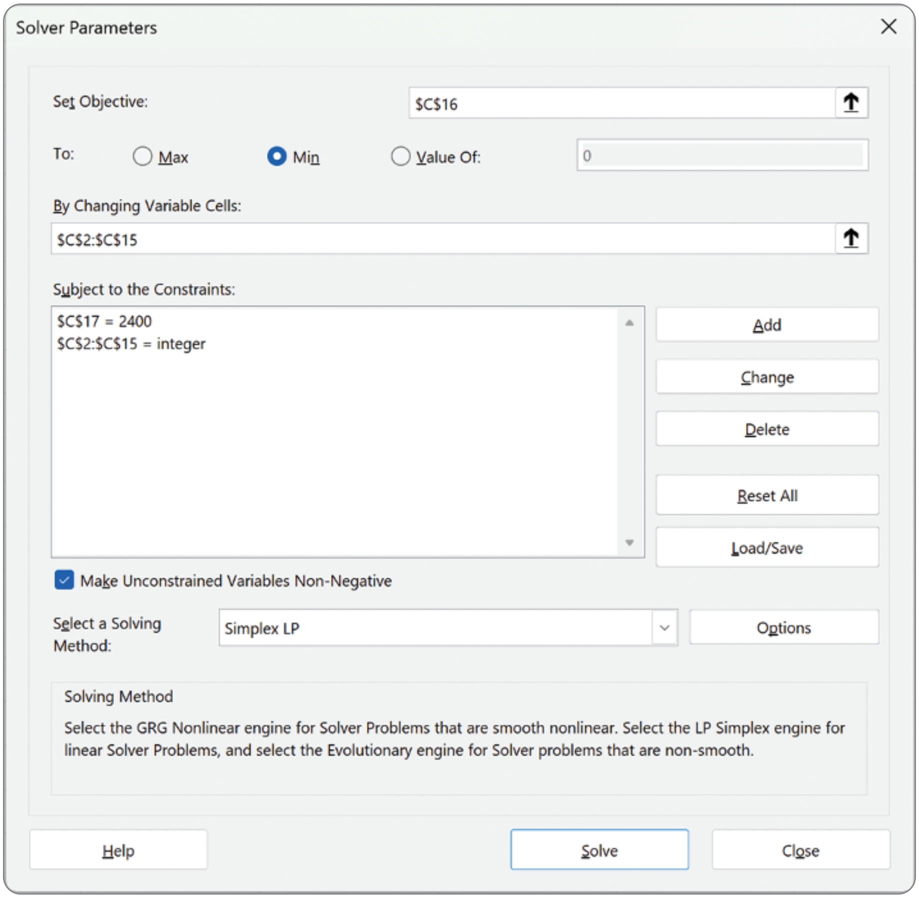


Figure 1.25: The Solver Parameters dialog box

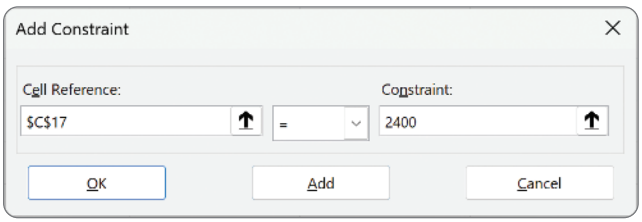


Figure 1.26: Adding a new constraint for Solver to use

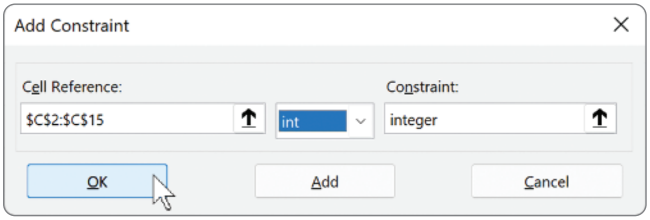


Figure 1.27: Adding an integer constraint

TIP

Solver's constraint window buttons are not exactly intuitively named. If you hit the Add button, the constraint is added (off screen), and a new blank dialog box appears. The OK button will attempt to add whatever you've already filled in and then exit back to the main window. Cancel will ignore any input and exit back to the main window.

Make sure the solving method is set to Simplex LP (see Figure 1.25). Simplex LP is appropriate for this problem, because it's *linear* (the *L* in LP stands for linear). Linear problems are those we can solve using linear combinations. For instance, using SUMPRODUCT to multiply the calories counts by quantity is a type of linear combination. Simplex LP works best for these.[3]

When you're ready, confidently click the Solve button. Excel should find a solution almost immediately. You should see the Solver Results dialog box in Figure 1.28. Click OK to keep the results. Click Cancel to let the cells go back to their previous values.

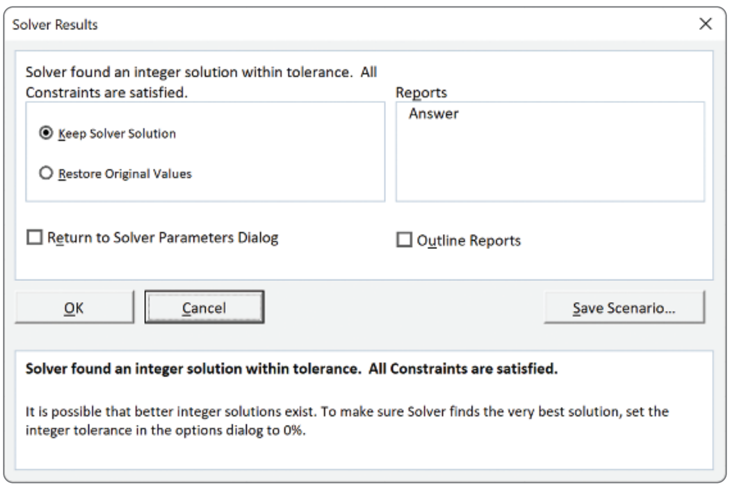


Figure 1.28: The Solver Results dialog box

[3] Examples of non-linear combinations would be taking a square root, a logarithm or exponential function. In these instances, we would opt for another algorithm provided by Solver.

My solution, as shown in Figure 1.29, is 5.

	A	B	C	D	E
1	Item	Calories	How Many?		
2	Beer	200	0		
3	Bottled Water	0	0		
4	Chocolate Bar	255	0		
5	Chocolate Dipped Cone	300	0		
6	Gummy Bears	300	0		
7	Hamburger	320	0		
8	Hot Dog	265	0		
9	Ice Cream Sandwich	240	0		
10	Licorice Rope	280	1		
11	Nachos	560	2		
12	Pizza	480	0		
13	Popcorn	500	2		
14	Popsicle	150	0		
15	Soda	120	0		
16		Total Items:	**5**	=SUM(C2:C15)	
17		Total Calories:	**2400**	=SUMPRODUCT(B2:B15,C2:C15)	

Figure 1.29: An optimized item selection

In this chapter, you learned a lot. First, we reviewed how to use Excel's status bar to quickly retrieve descriptive statistics. Next, you learned how to create Excel tables, which will allow you to stay organized. After this, we reviewed important LOOKUP functions and using dynamic array formulas. We reviewed PivotTables and how and when to bust out Solver.

Before ending this chapter, you should know that huge models don't fit very well in Excel. The version of Solver that comes packaged with Excel allows only 100 – 200 decision variables and constraints, depending on the version you're running. That's going to limit the size of the problems you can attack in this book.

If you want to go larger in Excel, you can buy a bigger version of Solver from Frontline Systems. Even better, use OpenSolver just as you'll do in the later sections of this chapter. Go to (www.coin-or.org) to download it. For some parts of this book, you're going to stick with Excel and Solver, for others you'll need to use OpenSolver. Don't worry, both tools vibe well together: you can set up your problem in Solver and then run it with OpenSolver. You'll see how in the chapters that follow.

NONLINEARITY CHECKS IN OPENSOLVER

OpenSolver will protest if it detects non linearity. We'll talk about non linearity as we go through this book. But I want to mention this warning now at the start of the book so you'll know what to do.

Non linearity won't be an issues for us, as we will be working with non linear problems. But that doesn't mean we want to get a warning each time it's detected. It's annoying, so let's fix it.

continues

(continued)

Here's what you'll do:

1. Set up your model, and add your constraints either to Solver or OpenSolver (they work interchangeably, so you can use either).
2. Next, from on the Data tab, click the OpenSolver Model dropdown and selection Options. . . (see Figure 1.30).

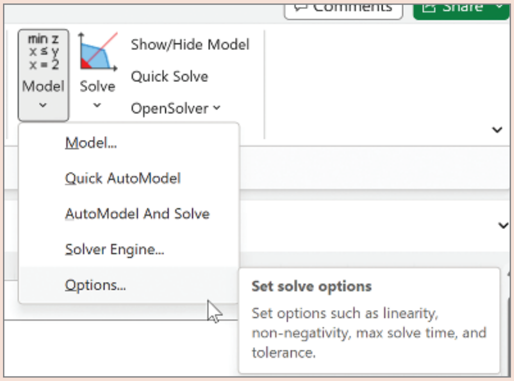


Figure 1.30: Setting the options for OpenSolver

3. From the OpenSolver – Solve Options dialog box, deselect the checkbox **Peform a quick linearity check on the model**. Then Press OK (see Figure 1.31).

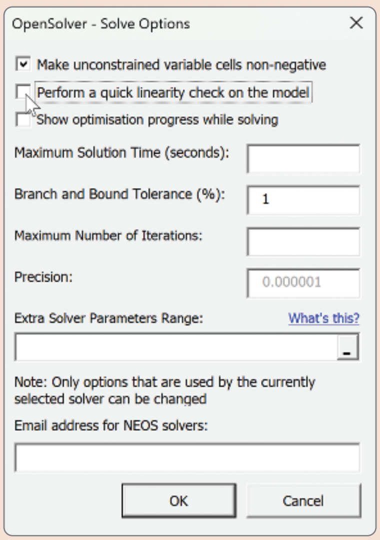


Figure 1.31: Deselect the Perform a quick linearity check on the model in the OpenSolver dialog box to ensure you can run the models described in this chapter.

You'll have to do this for each new model you create where you plan to run OpenSolver.