3

SQL Editing Reimagined

Database developers typically spend a great deal of time creating and editing SQL queries, so it only makes sense to make this experience as helpful, convenient and productive as possible. Azure Data Studio has truly reimagined how developers interact with SQL coding, and for that matter, all the languages supported on the platform.

This is accomplished by focusing directly on keyboard interactions, which include IntelliSense, keywords, code snippets, and existing object definitions. Much of the ADS User Interface is also configurable, providing customizable color themes, zoom levels, window options, fonts and icons.

# IntelliSense, Snippets and Object Definitions

To get started with entering SQL queries, you can either click on ‘New query’ from the Welcome page, or for more specificity, right click on your target Database in the ‘Side Bar’, and then choose ‘New Query’ as shown in figure 3-1:

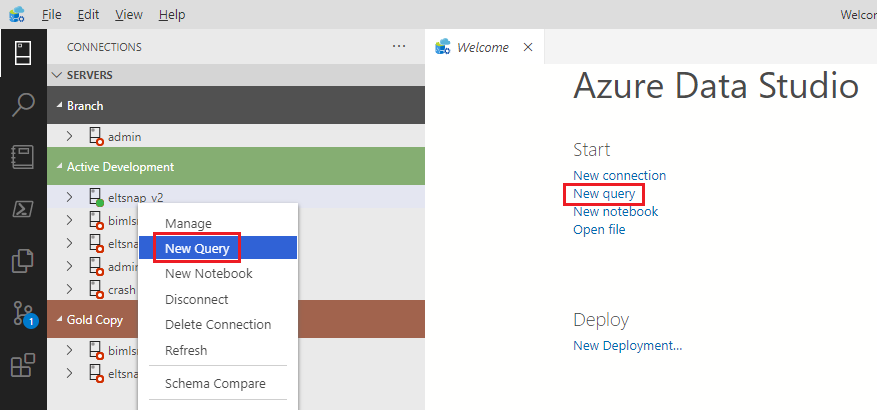


Figure 3-1. Create a New SQL Query

This will open a blank editing window which where you simply start typing. As you start typing, each keystroke may offer a suggestion, providing IntelliSense as shown in the pop-up window for Figure 3-2. Notice that the FROM keyword is highlighted in the sorted pop-up list which indicates that you simply need to hit the tab key to implement this substitution.

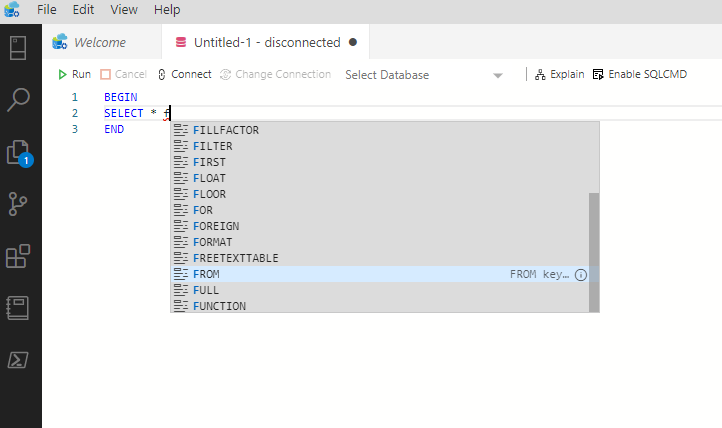


Figure 3-2. IntelliSense Keyword Pop-up

The up and down arrows provide navigation within this list, as does a mouse-click on the desired keyword. Also notice that figure 3-2 has the ‘Side Bar’ hidden. Use the key sequence Ctrl+B to toggle this screen section off and on. You can achieve the same effect by clicking on an icon in the ‘Activity Bar’, or optionally use the top ‘Menu Bar’ selections: View, Appearance, Show Side Bar.

## Code Snippets

One of my favorite editing features in ADS are ‘Code Snippets’. These can be a huge timesaver, are integrated into the IntelliSense user experience, and are customizable. As an example, let’s say you wanted to create a table. By typing createtable you will see the ‘camel cased’ snippet sqlCreatTable pop-up. You just hit ‘Tab’ when highlighted, or click on the snippet in the pop-up, and you will get a ‘Create Table’ template as displayed in figure 3-3:

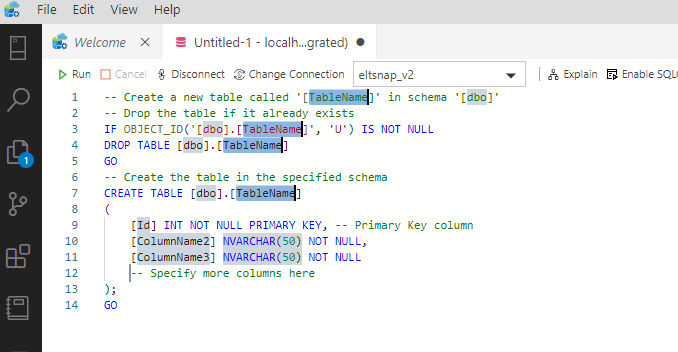


Figure 3-3. Create Table (Default) Snippet

You may be surprised to now see 4 blinking cursors! This is because you are automatically placed in the process of completing defined variable replacements, the first of which (TableName) has a total of 4 instances. If you were to type ‘product’ the variable replacement would occur 4 times and figure 3-4 would be the result.

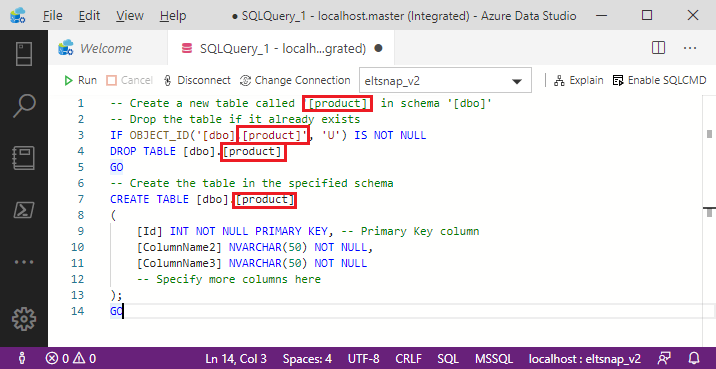


Figure 3-4. Snippet Variable Replacement

To move to the next defined variable, press the ‘Tab’ key again and you will see the next variable (used for the schema name) which also has 4 occurrences that can be changed simultaneously. The next ‘Tab’ will take you to the first table column (in this case: [Id]), and so on until all variables have been visited.

This ‘default’ (built-in) snippet is a nice start to creating a table, but you may be thinking “I’d like my snippet customized for our coding standards”. Not to worry, in chapter 9 we will cover how you can easily create your own snippets. Just like the built-in snippets, these will automatically surface in your SQL editor window using the same IntelliSense driven keystrokes.

## Object Definitions

While you are editing your SQL Queries, it is a common requirement to reference ‘Object Definitions’ within your database model. For example, let’s say you are querying a certain table column and need to know if it could contain NULL values. In this case, the standard IntelliSense capability of suggesting ‘column names’ falls a bit short. Instead, what is needed is the full definition of the table object.

Since your database could contain hundreds of tables, each of which could have many columns, it can be a pain to quickly retrieve table and column definitions by browsing these object definitions in the ‘Side Bar’. To remedy this situation, ADS provides direct access to object definitions, without leaving the editor window. Simply ‘right click’ on any table name in your query, and a couple options will pop-up. Figure 3-5 captures this pop-up when right clicking on the table name oledb\_connection:

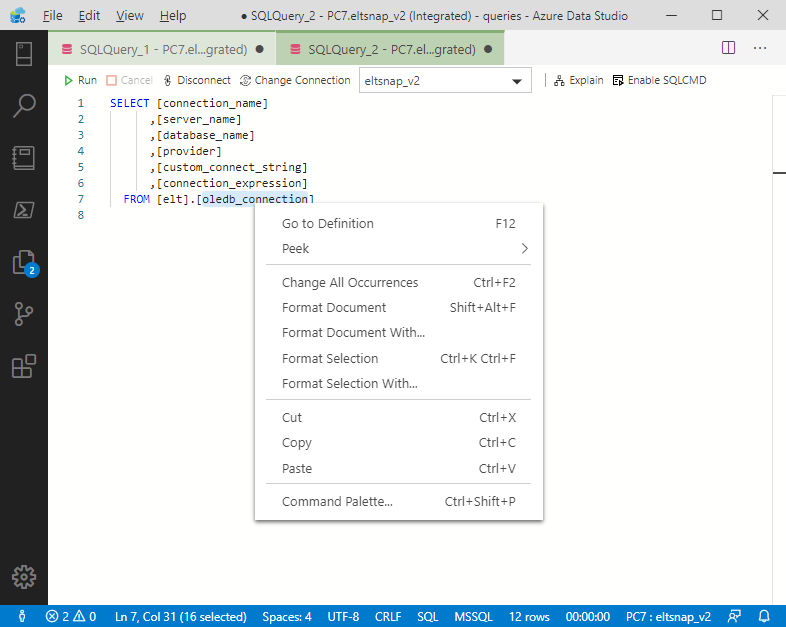


Figure 3-5. Accessing ‘Object Definitions’

The first two options on this pop-up will provide you with the table definitions. The first “Go to Definition” option will open a new editor window with the table definition in the form of a table create statement. Since this is a runnable script, this method provides a convenient way to change the definition of the table if needed (assuming you are not concerned with losing any data within in the table).

The second “Peek” option, will provide you with the same definition, but in this case displayed in the existing editor window as shown in the figure 3-6:

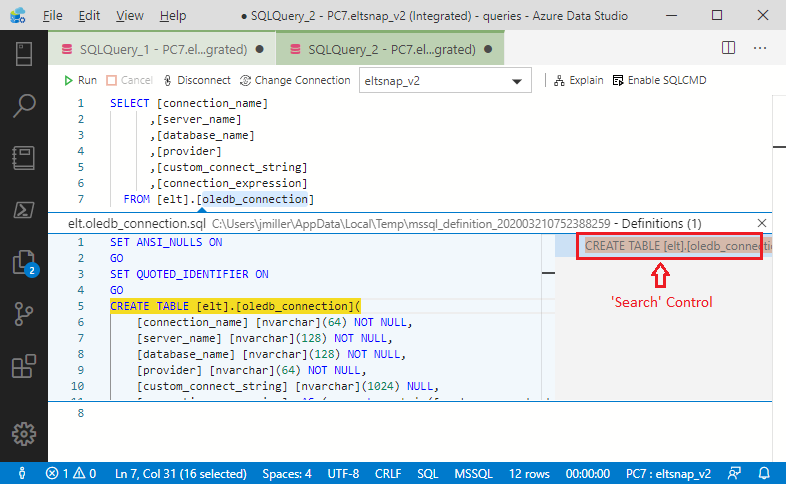


Figure 3-6. Peek ‘Object Definitions’ Option

In the event the table definition includes many columns, you can use the ‘Search Control’ on the right side of the screen to search for a specific column definition.

### Creating a Snippet for Column Definitions

You may be thinking, “This is helpful for retrieving a column definition located a single table”, but what if I want to see how the same column is defined in all tables?”. Good question, and one that could be answered by creating a snippet. A good place to start with snippets is writing the base query, which for our case will use the INFORMATION\_SCHEMA.COLUMNS system view on the msdb ‘’system’ database. In this query, we will be searching for all definitions of the plan\_name column:

select \* from msdb.INFORMATION\_SCHEMA.COLUMNS where COLUMN\_NAME = 'plan\_name'

A subset of the results from running the above query are shown in figure 3-7:

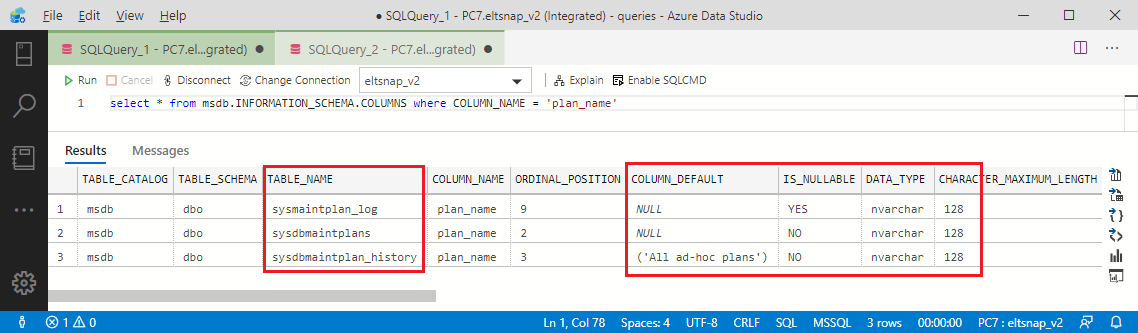


Figure 3-7. Sample Query using Information\_Schema.Columns

The result set above reveals that the column plan\_name is found in 3 tables within the msdb database, has a consistent data type, but varies in terms of nullability and default values, commensurate with how each table uses the column. You could now save this snippet (or ‘template’) as a stand-alone query simply residing in your file system, or convert it to a formal ‘ADS Snippet’. The former could be accessed by using the ‘Menu Bar’ File, Open commands, and the later would be retrieved by keystrokes directly in the SQL editor window. Another consideration with this decision is that a formal ‘ADS Snippet’ can optionally provide variable substitution, which can greatly simplify the re-use of your custom snippet.

The next section will cover how to save your ADS queries and customized snippets.

# Saving Queries and Snippets

When working with multiple ‘file based’ queries, it is helpful to organize related scripts into a common folder structure. To achieve this, you simply select (or optionally create) a folder using the ‘Menu Bar’ File, Open Folder command as shown in figure 3-8. This will establish your ‘current’ folder context:

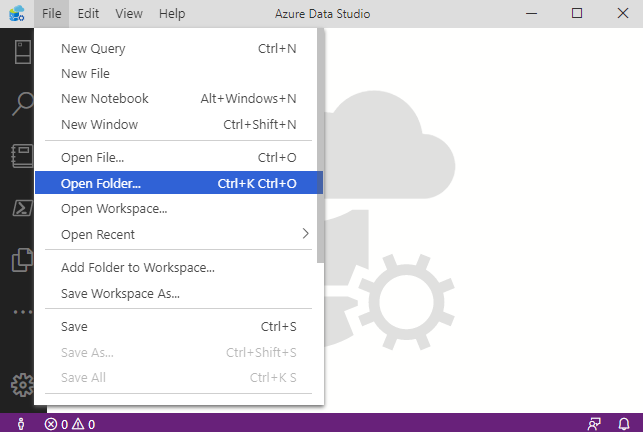


Figure 3-8. File, Open Folder Command

In the case you need to create a New folder, you can still use the ‘Open Folder’ dialog box. This is done by clicking in the ‘white space’ (next to the existing folders) where you will be able to enter a new folder name via a pop-up window. The navigation for this user action is presented in figure 3-9:

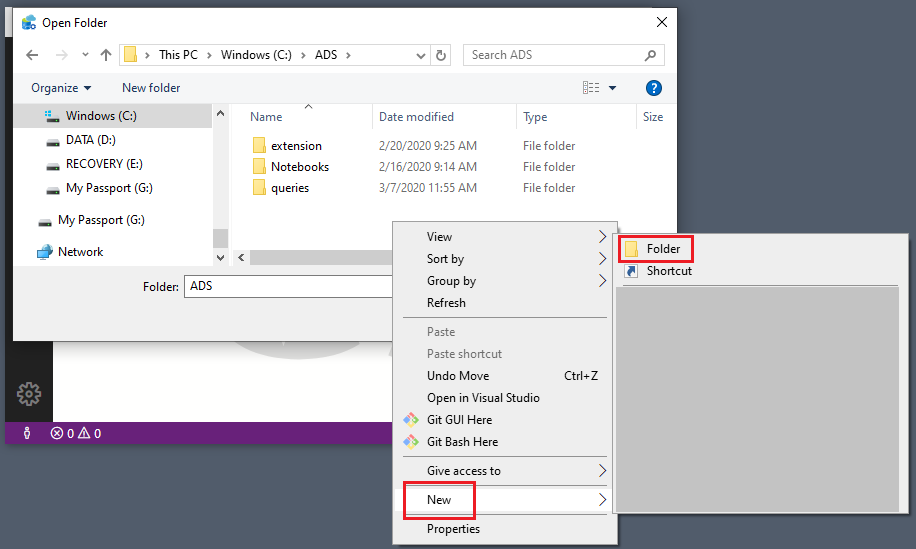


Figure 3-9. Specifying a Folder Name for Queries

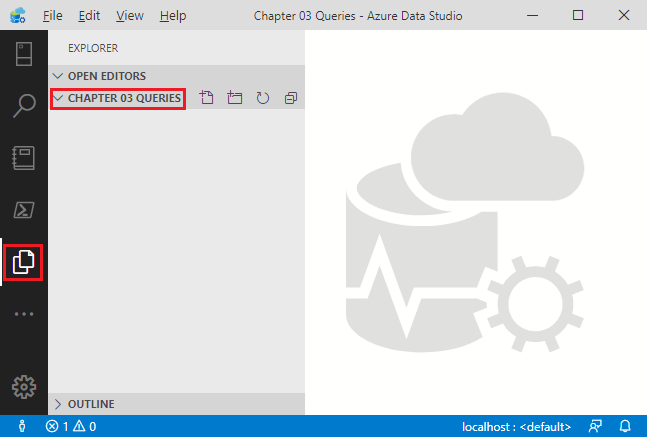
Once you have selected your ‘current’ folder context, queries and scripts that you subsequently save will be placed in this folder by default. The File icon in the ‘Activity Bar’ as shown in Figure 3-10 will provide the name of your current folder context.

Figure 3-10. Current Folder Context

Keep in mind that your working folders could later be tied to GitHub or other source control system. Consequently, your folder organization and naming conventions should be considered. Even if you are not sharing with others, you may find that GitHub is a convenient repository to store your queries and scripts. This is both in terms of safe keeping, as well as accessibility when away from your primary workstation. See Chapter 13 for a ‘Deep Dive’ into GitHub and ADS.

Now that we have a ‘current’ folder, let’s tweak and save our earlier INFORMATION\_SCHEMA.COLUMNS query into the file system. Here is the bit more generic query we’ll use:

select \* from INFORMATION\_SCHEMA.COLUMNS where COLUMN\_NAME = 'column\_name'

As you might have guessed, pressing CTRL + S will open the ‘Save’ dialog box, or you could use File, Save, from the ‘Menu Bar’. In either case you will receive the dialog box shown in figure 3-11 where you can name your file-based query:

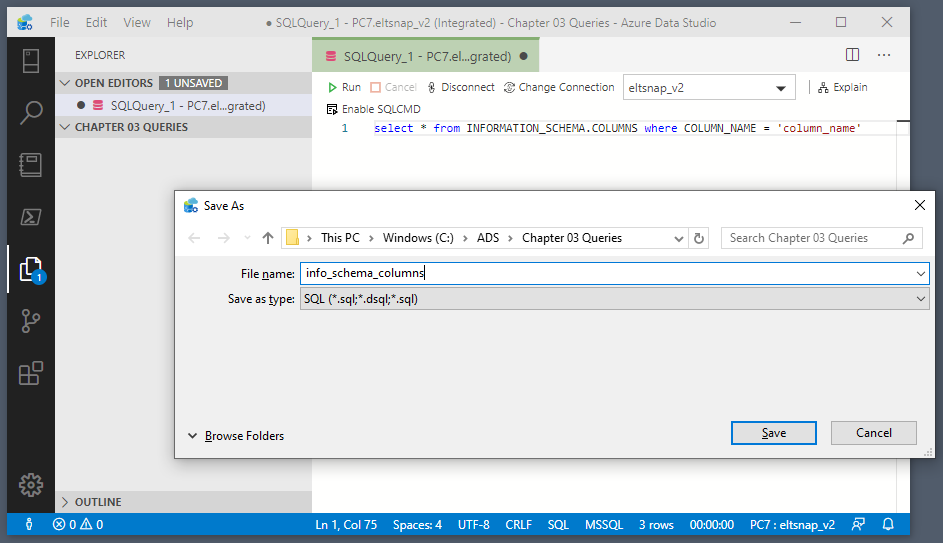


Figure 3-11. Save Query File

Ok, saving a file is admittedly a pretty basic user action. However, what if you would like to save this query as a reusable ADS Snippet? Well for starters we will want to make another tweak to this script which will invoke variable substitution logic on re-use. This is achieved by replacing 'column\_name' with the parameter syntax ${1:TableName}:

select \* from INFORMATION\_SCHEMA.COLUMNS

where COLUMN\_NAME = '${1:ColumnName}'

Note: for simpler snippet coding, we will place this query on a single line in the full json snippet syntax:

{ "Information Schema for Columns": {

"prefix": "InfoSchemaColumns",

"body": "select \* from INFORMATION\_SCHEMA.COLUMNS where COLUMN\_NAME = '${1:ColumnName}'" } }

The first line has the literal "Information Schema for Columns " which is the snippet name. The next line contains the prefix “InfoSchemaColumns” which will cause this snippet to surface based on character matches, which do not necessarily need to be sequential. For example, this snippet would be found by typing ‘infcol’. The third line is the snippet code itself, which will appear upon pop-up selection in your editor window.

Note: we go into much more detail on snippets in Chapter 09.

To save the snippet, press CTRL+SHIFT+P (or from the ‘Menu Bar’ click on View, Command Palette), enter ‘snippet’ in the search box, and select “Preferences: Configure User Snippets” as shown in figure 3-12

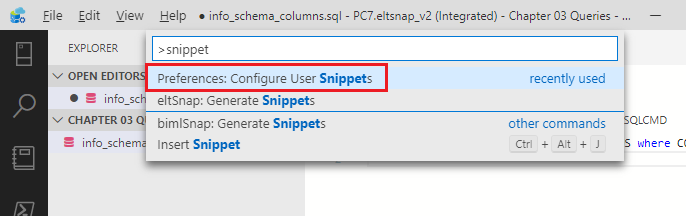


Figure 3-12. Configure User Snippets

Next enter ‘sql’ into the snippet search, and select the file: sql.json as displayed in figure 3-13:

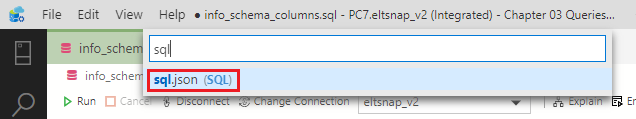


Figure 3-13. sql.json Snippets File

And finally paste in your json script as shown in figure 3-14:

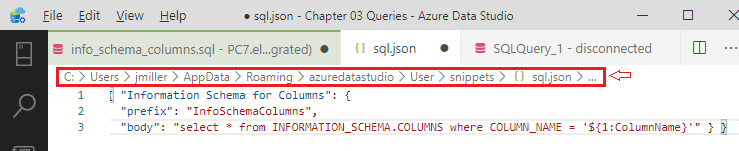


Figure 3-14. A Sample sql.json Snippet

Notice the above window also provides the physical location of the ‘sql.json’ file that you are modifying. Press CTRL+S to save your changes, and enable your new snippet to be used. Press CTRL+N to create a new query window and type the character sequence: ‘infcol’. You should see the snippet pop-up as rendered in figure 3-15:

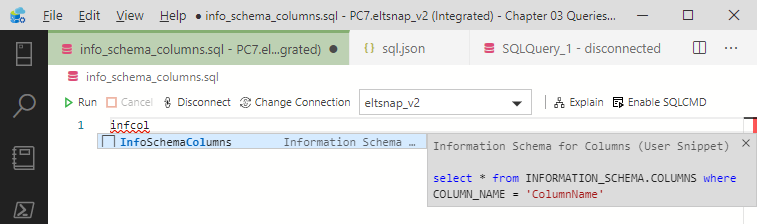


Figure 3-15. Using IntelliSense to Find a Snippet

Selecting this snippet will produce the SQL code in your editor window, with the cursor highlighting the field (in this case ‘ColumnName’) which you will want to replace with the actual column name you are researching. Figure 3-16 displays the snippet which is now readily available with just a few keystrokes:

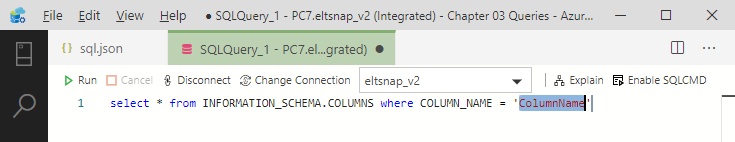


Figure 3-16. Variable Highlighting for a Snippet

As mentioned earlier, we have much more content to come later in the book on ‘ADS Snippets’, which by the way can apply to any of the Azure Data Studio supported languages including PowerShell and Python.

# Top Down View with Minimap

Life at times can be detailed and complex. The same can be true of ‘SQL code’, or for that matter ‘code’ written in any language. Most database developers have written, reviewed, or maintained SQL code that exceeds hundreds, or perhaps even a thousand lines. While ‘big code’ can be intimidating, ADS has a Minimap feature that can at least soften the pain of working with voluminous lines of SQL.

For the following example you can use any larger script. If more lines are needed, just use copy/paste to multiply the SQL code you have since we will not be ‘executing’ this code.

To get started with Minimap, you will first want to turn this feature on via the ‘Command Palette” by pressing CRTL+SHIFT+P, entering ‘minimap’, and selecting “View: Toggle Minimap” as shown in figure 3-17:

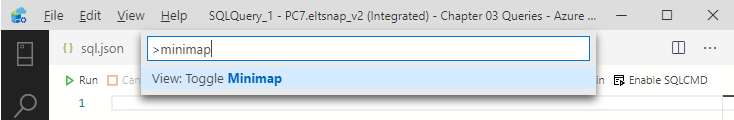


Figure 3-17. Toggle Minimap On

With any script in place, and with Minimap turned on, you will see a birds-eye outline of your code on the right side of the editor window, with your current ‘cursor position’ highlighted as a thin blue line as displayed in figure 3-18:

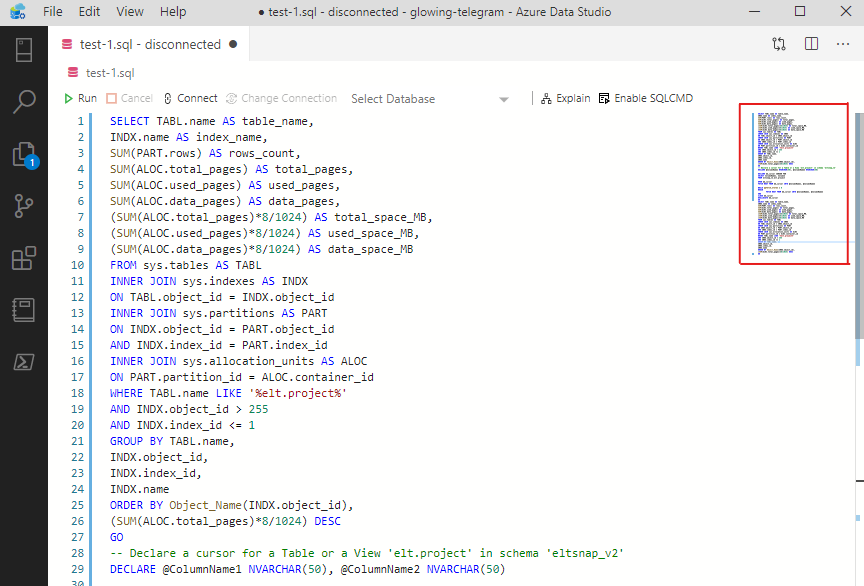


Figure 3-18. SQL Editing with Minimap Outline

Syntax error are reflected in the Minimap with red highlighting as shown in figure 3-19. This is helpful by providing immediate feedback, as well as proximity of the error, even if the actual (readable) SQL code is ‘off screen’.

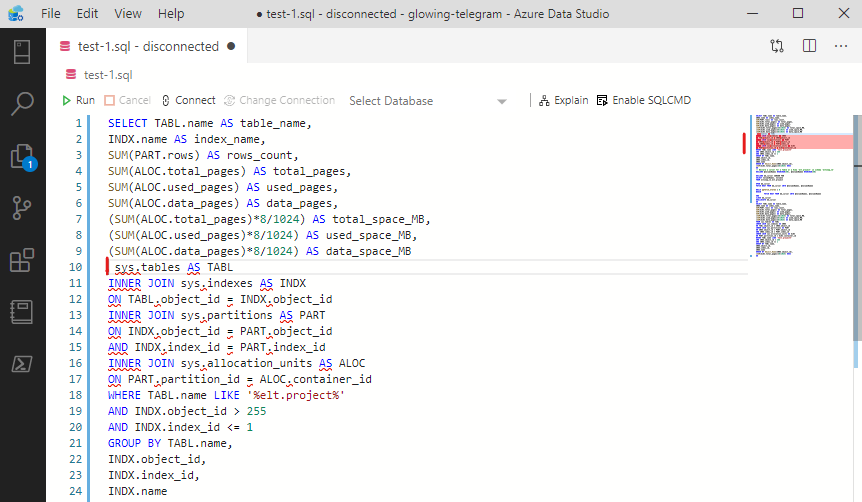


Figure 3-19. Minimap Syntax Errors

Another nice feature of Minimap is the ability to view a large selection of your code, even if it spans hundreds of lines, and requiring excess ‘scrolling’ to visualize. Figure 3-20 demonstrated this capability, albeit on a smaller scale.

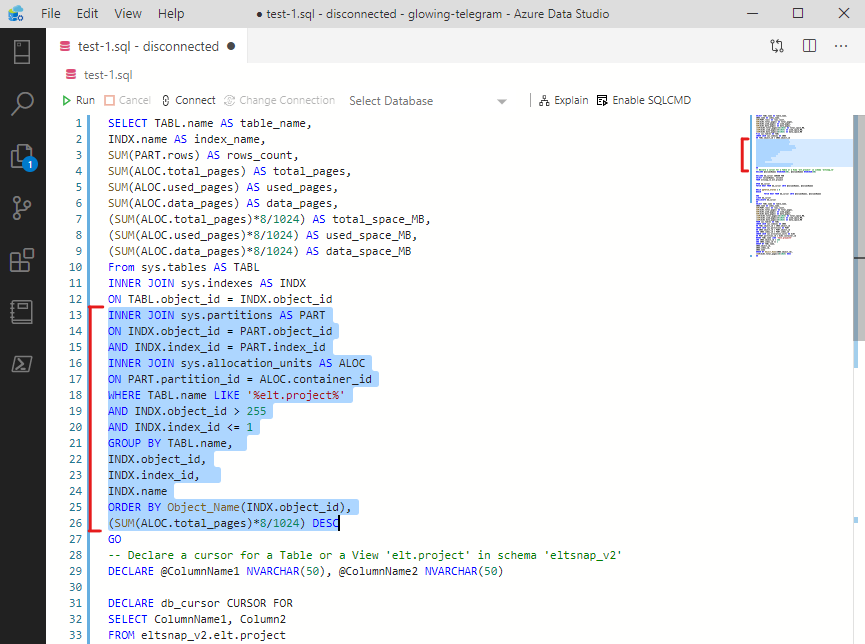


Figure 3-20. Minimap Code Selection

Finally, Minimap provides a convenient way to visualize all locations containing text you are searching for within the entire script. Figure 3-21 shows the result of searching for a table called elt.projects within a SQL script.

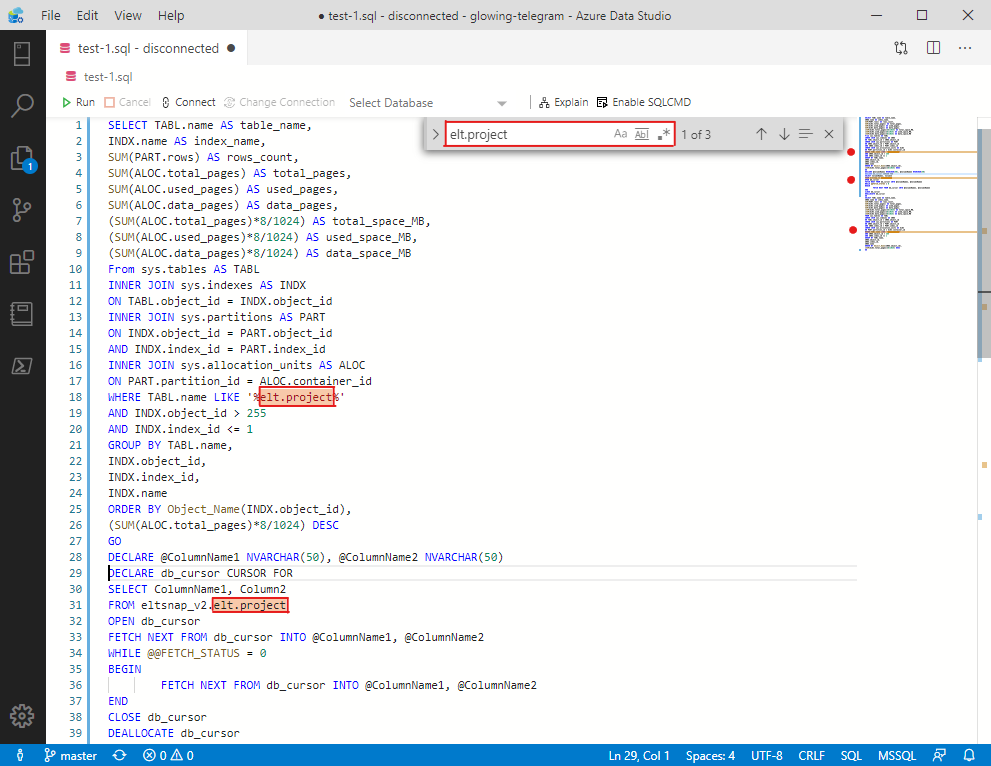


Figure 3-21. Minimap Search Results

So, the next time you find yourself editing a painfully large block of SQL code, remember to ‘toggle on’ Minimap to make your task at least a little more manageable.

# SQL Queries via the Command Terminal

An interesting capability of Azure Data Studio is the integration of a standard SQL editor with other language options. These language options are available by using Notebooks (introduced beginning in Chapter 5), and by using the integrated Terminal, which we’ll introduce next.

To open the Terminal window, press CTRL+` (backtick), or using the ‘Menu bar’, select View, Terminal. Once opened, you’ll see the Terminal pane on the bottom right section of the ADS application as shown in figure 3-22:

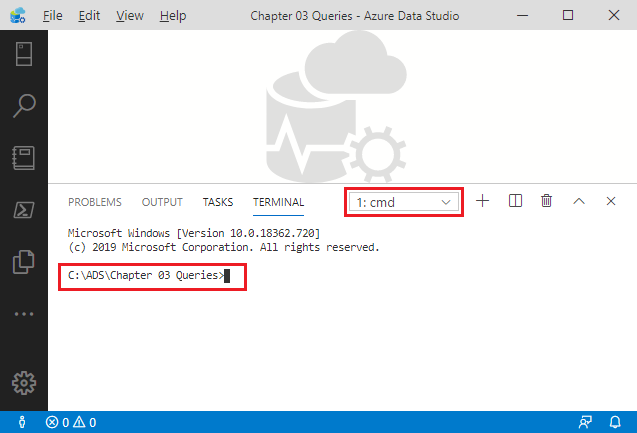


Figure 3-22. ADS Terminal Window

Notice the Terminal opens in the context of the ‘Windows Command Line’ and inherits the Folder context that was previously opened in ADS (in our case it was “Chapter 03 Queries”). Although we could query SQL via the Windows Command Line, we have more options when using PowerShell. To switch the context to PowerShell, you can simply type ‘powershell’ into the Terminal window as displayed in Figure 3-23:

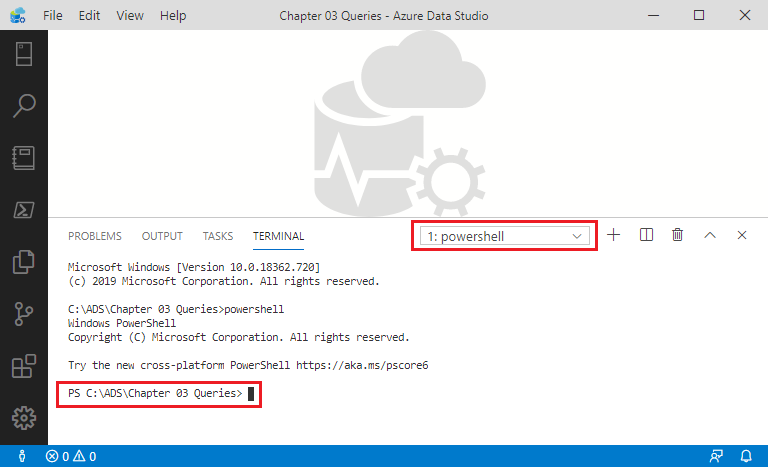


Figure 3-23. PowerShell Terminal Window

Note: In Chapter 15 we will use the new “PowerShell Integrated Console” which is activated by using the ‘Activity Bar’

To query your ‘local instance’ of SQL Server, enter the following command into the PowerShell command prompt

Invoke-Sqlcmd -Query "select \* from INFORMATION\_SCHEMA.TABLES" -ServerInstance "localhost"

And hit enter. You should see the SQL query results shown in figure 3-24:

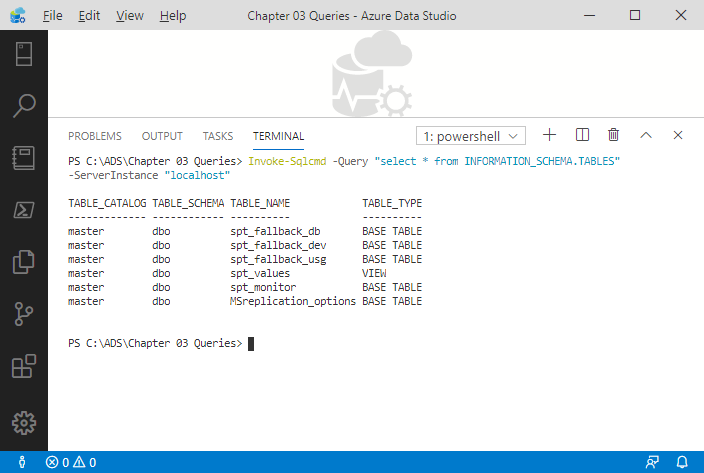


Figure 3-24. Running a SQL Query in PowerShell

Since we are in PowerShell, we have many other options for our result set, such as exporting as a CSV file. This can be achieved by running the following PowerShell script:

Invoke-Sqlcmd -Query "select \* from INFORMATION\_SCHEMA.TABLES" -ServerInstance "localhost" | export-csv -Delimiter ',' -Path "tables.csv" -NoTypeInformation

After running in the Terminal, the results of the above script are ‘piped’ to the file called ‘tables.csv’, instead of being returned into the terminal session. To see the file, click on the ‘Explorer’ Icon in the ‘Activity Bar’. The file should now be available in the ‘Side Bar’ as shown in figure 3-25:

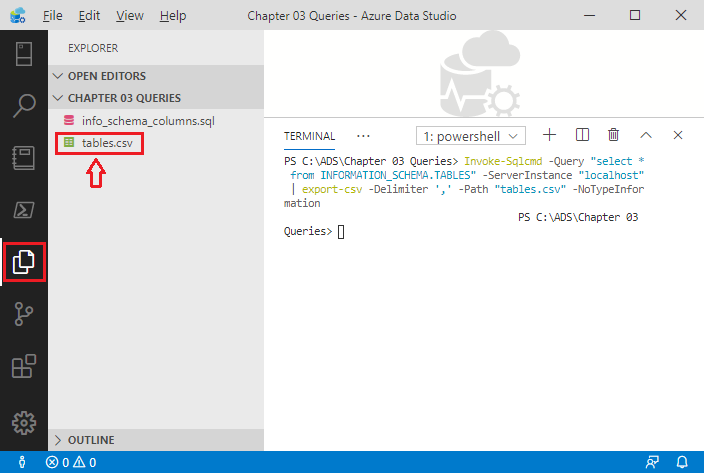


Figure 3-24. Exporting a Query using PowerShell

This is a simple but instructive example of the synthesis of traditional SQL queries, with the extended language support built into Azure Data Studio. This illustration leveraged the ADS Terminal window but read on as deeper and even surprising capabilities are ahead!

1

Your First Chapter

Welcome to the Apress template! We build our editorial processes around this Word template that you're looking at right now. Begin each new chapter by: 1) Opening this template, and 2) Selecting File -> Save As to create a new chapter file.

**Tip:** Don’t expect a WYSIWYG view of your book as it will appear to your readers. The template’s job is to facilitate your use of building blocks such as headings and lists and paragraphs that we support in our books.

# Chapter File Names

Any project involving many files benefits from a naming convention. We like to see the following format when you name your chapters:

ch01.doc

Optionally, you can choose to provide some additional text to jog your memory about each chapter's topic. For example:

ch01\_GettingStarted.doc

Make sure the chapter number is first. Include the leading zero for chapters 1 through 9. Keep the format meticulously consistent. Don’t mix things up by giving us ch01, then CH02, and then ch3. Keep to ch01 all the time. That way the chapters sort correctly when we look at a file listing.

# Named Styles

We like to think about writing a book chapter as being similar to building with Mega® Bloks. Our template provides a set of named paragraph styles. These are what you must use to write your chapter. You can't invent your own styles; you must use the building-blocks in the set.

Figure 1-1 shows a typical view of this template as first opened in Word. The view varies between Mac OS X and Microsoft Windows, and between the various versions of Word, but you should see something similar to Figure 1-1. Notice the horizontal listing of named paragraph styles in the Ribbon, the first of which is Body Text, and then comes Bullet, and so on.

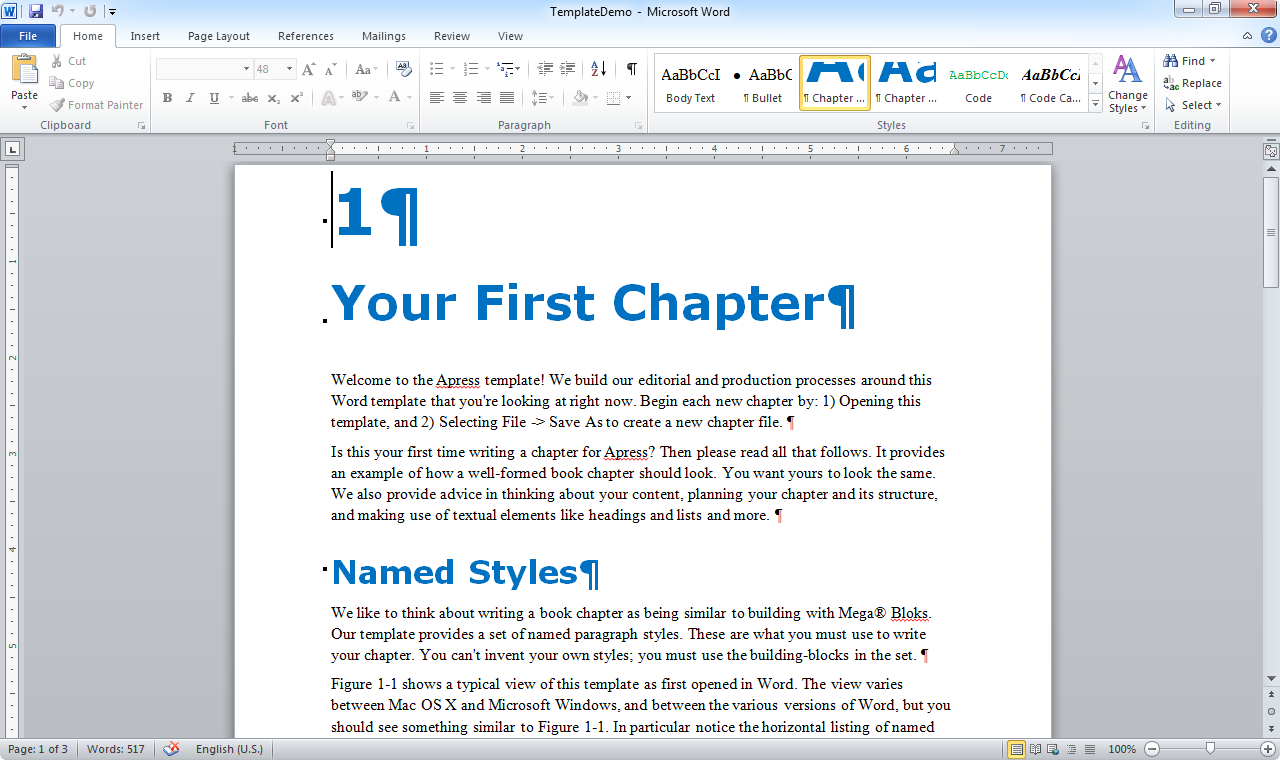


Figure 1-1. Opening view of this template in Word 2010

Pay attention to the Style group in the Ribbon. Click the chapter number at the top of this template to see the highlight move to indicate a Chapter Number paragraph. Click anywhere in the chapter's name and see that it's in a Chapter Title paragraph. Click into this paragraph that you’re reading now to see that it's a Body Text paragraph.

Figure 1-1's initial view isn't very useful. It's too tedious to select styles from the in-Ribbon gallery. Each version of Word provides a different way to open a vertical listing of styles that you can place to your left as you write. Figure 1-2 shows the tiny icon to click in Word 2010 for Microsoft Windows, and Figure 1-3 shows what to click in Word 2011 for Mac OS X.



Figure 1-2. Icon to open a Styles panel in Word 2010 for Windows

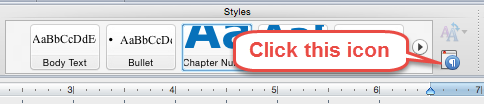


Figure 1-3. Icon to open a Styles panel in Word 2011 for OS X

A Styles panel like the one at left in Figure 1-4 is what you're after. Now you can highlight or put your cursor into any paragraph and click a style name in the panel to apply the named style. In this way you should apply a named style to each and every paragraph in a chapter.

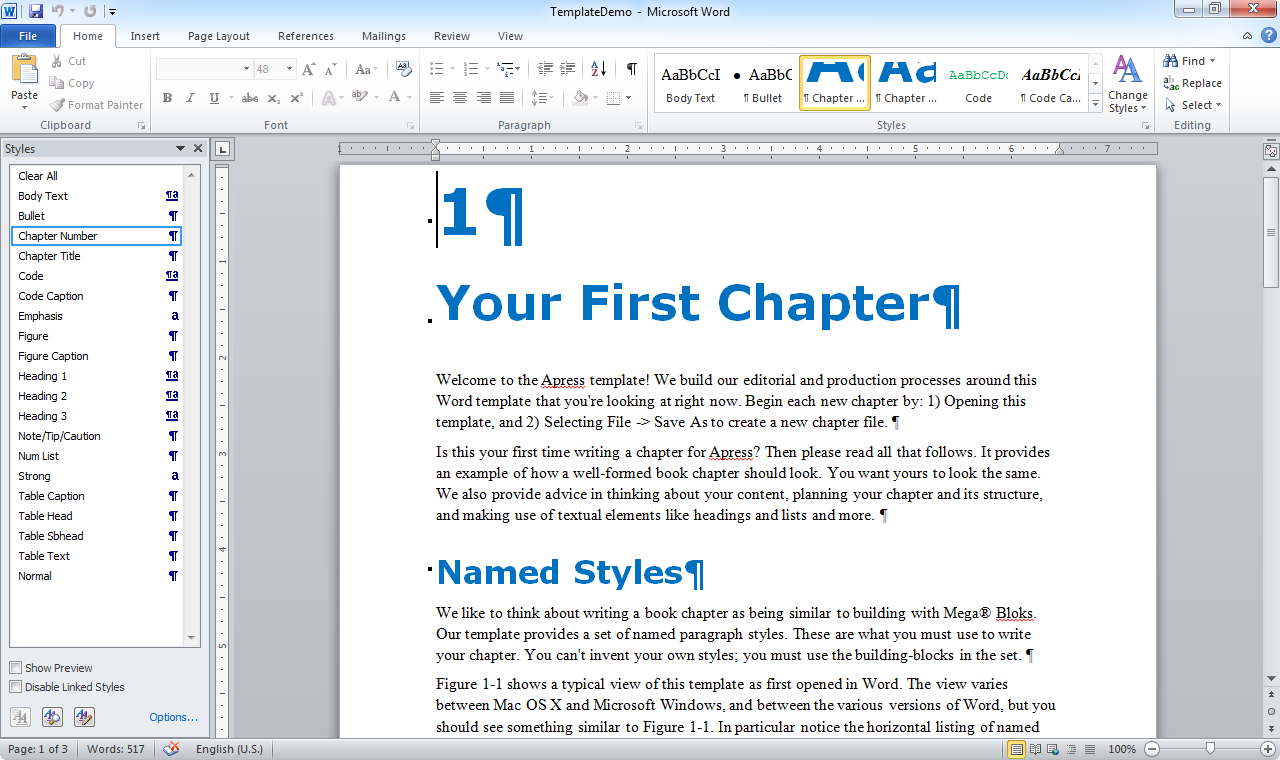


Figure 1-4. Word 2010 with the Styles panel open to the left

You can also select a style and begin typing. For example, you can select Body Text in a blank paragraph and begin typing a series of Body Text paragraphs one after the other. Some authors prefer to type an entire section as body text so as to stay focused on their content without distraction, and then go back afterward and apply other styles as needed.

Don't leave blank paragraphs for spacing purposes, and don't leave any paragraphs without applying one of our styles to them. Use only our styles, and be sure that every paragraph has one of our styles attached.

# Flowable Text

Writing is not about the printed page anymore. The world is changed, and two-thirds or more of your readers will be on electronic devices. Good publishing in today's market is about creating flowable text. Everything you've ever read about writing for the web now applies to books.

The left-side image in Figure 1-5 shows an example of complex formatting we now like to avoid. You're looking at a triple-level list as viewed from Amazon.com's Kindle app on the iPhone 5. The look begins to get awkward as nested items are increasingly pushed to the right. A large area of the display surface goes unused, and it's difficult for readers to come to grips with the three kingdoms when the top-level bullets are separated by many "pages" of nested content.

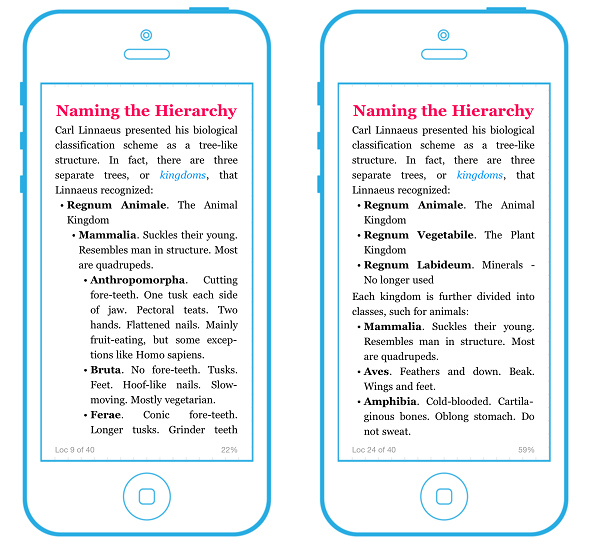


Figure 1-5. Complex versus simple formatting

The right-side image shows a better way to present the same content. The look is more appealing to the eye because the column-width is consistent from top to bottom. List levels are presented independently, making it easy for readers on everything from tiny iPhones to large desktop browser windows to take in each level at a glance. None of the display's surface area goes to waste.

Your content will be read from smart phones, tablets, Kindle devices, Nooks, web browsers on a PC, and last of all from the printed page. It's to your advantage to write simple content that can be reflowed onto devices no matter what their display dimensions. We've designed this template with the minimum-needed styles to help keep the content simple and flowable to any device.

**Tip:** Do not obsess over pagination or whether figures fall on one page or another as you write in the template. Readers will see varying views depending upon whether they read in print, from the Kindle or the Nook, from an online service such as Springer Link, or from PDF. It is not possible, nor should you try, to optimize the layout of your content, because it is impossible to know in what medium and format readers will see it.

# Headings and Structure

Chapters are composed of sections, and a section is a heading followed by content. It's a good idea when you begin to write a chapter to plan out the major topics. Write the main headings and be sure you're happy with how they flow before you begin writing their content. For example, Figure 1-6 shows our first cut at the main sections in this template as viewed from Word's Outline View.

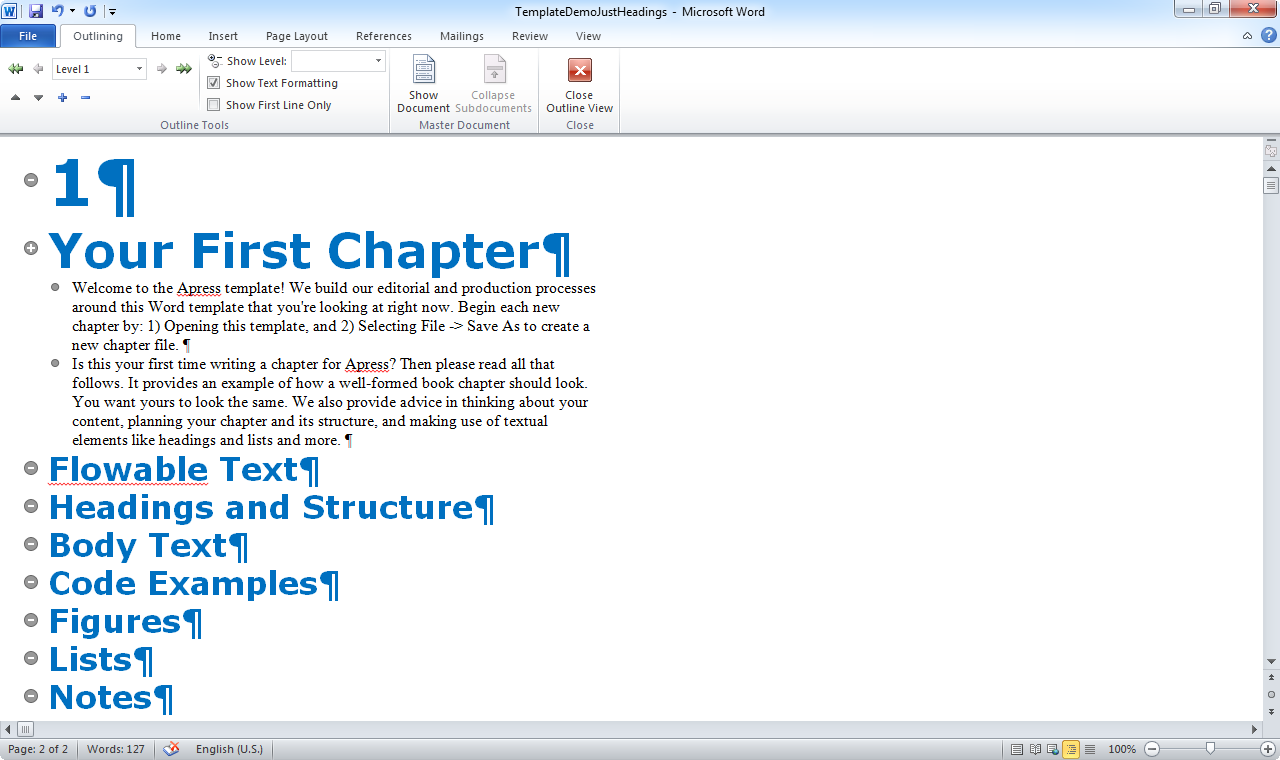


Figure 1-6. Our first cut at this template's major topics

You can also see in Figure 1-6 what we think of as the canonical form of a book chapter:

1. Chapter number and title
2. One to three paragraphs introducing the chapter
3. Then the main sections in their proper order

Apply the Heading-1 style to all your main headings. Select the heading paragraphs. Then press ALT-1, or click Heading 1 in the Styles panel shown earlier in Figure 1-4.

Begin each main section by recursively applying the same approach and dividing each section into subsections. Figure 1-7 shows an example that is our first cut at subdividing the upcoming section on "Lists". Press ALT-2 or click Heading-2 in the Styles panel to mark the subheadings.

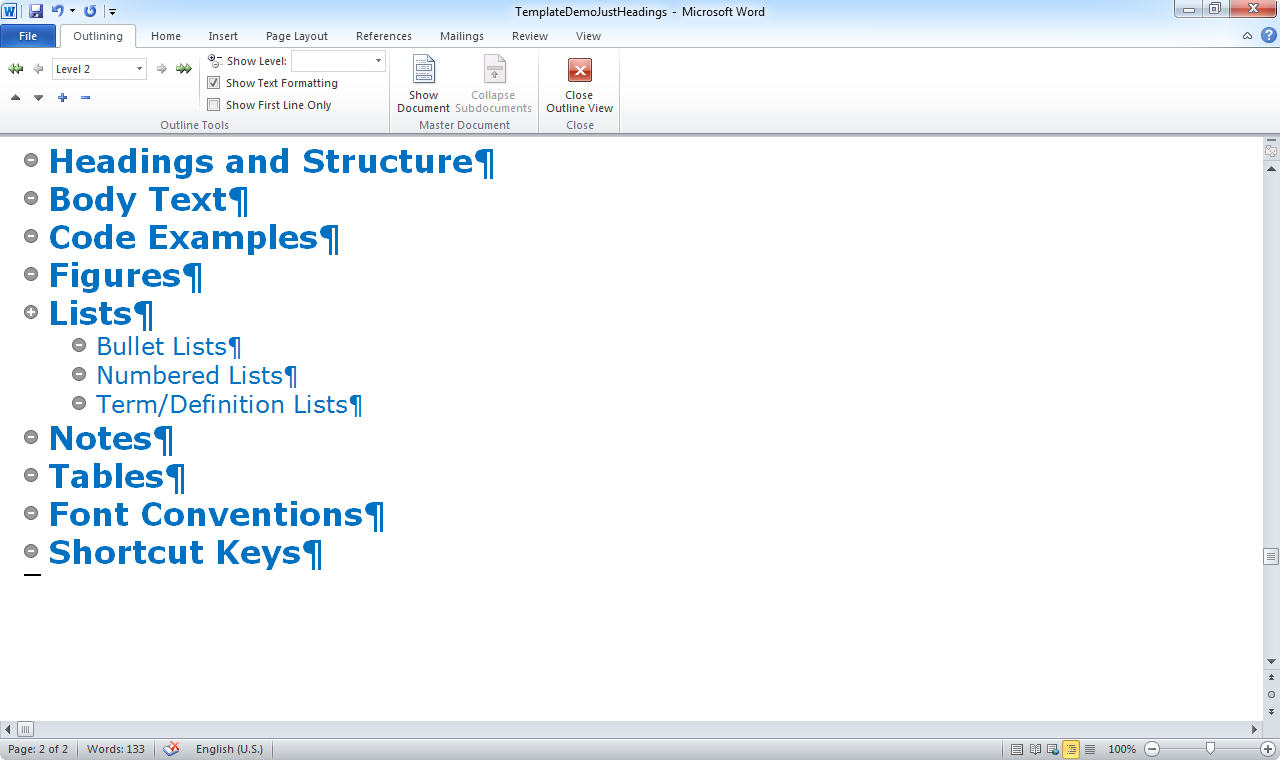


Figure 1-7. A shot at the subsections within a main topic

Think of the approach we’ve described as "Just in Time" chapter and section planning. Plan the main headings just as you begin writing a chapter. Plan each section's subtopics just as you begin that section. Make a good faith effort to build a solid plan, because doing so will clarify your thinking, and your writing! Then make adjustments as you go along, because you'll rarely nail the structure of a chapter or a section perfectly the first time.

**Note:** Our template provides for a third level of heading. We recommend using it only when you're able to divide a Heading 2 section into small subsections of just a paragraph or two each. If you're writing Heading-3 sections more than half a page in this template, we recommend restructuring your content so as to avoid such large sections so deeply nested.

Headings split a topic. Never allow just one main heading in a chapter, or just one subheading in a section. You can’t split something into just one piece, right? Anytime you divide a chapter or a section into sections or subsections, you must divide into at least two pieces. If you’ve written only one subheading in a section, then you either don’t need that subheading, or you need another that you’ve overlooked.

# Body Text

Let body text carry the day. Write the preponderance of you content into plain old body text paragraphs such as this one. Imagine someone reading your chapter without being able to see the headings, figures, lists, or any other elements. Would that imaginary reader still get your message – just from the body text? Aim for that to be the case.

Introduce everything. Begin each chapter with one to three paragraphs of introduction. Likewise, begin each section and subsection with a paragraph or two of introduction. Introduce special elements such as lists and figures and code examples from body text, before readers encounter them. You’ll see examples of how to write such introductions throughout this template.

Avoid “stacking” of non-body-text elements whenever possible. Placing several figures, tables, or numbered code listings in succession can be ok if done in moderation, but in the main pages appear less cluttered and are easier to parse when robust amounts of body text intervene between other elements in the text.

# Bold and Italic

We support **bold** and italic text through two character styles named Strong and Emphasis. Select text within a body paragraph that you want to be bold or italic, and then click Strong or Emphasis in the style panel.

Be sparing in your use of bold. We generally prefer that you avoid bold, with term/definition lists being one of the few exceptions that we allow. You’ll see an example of those in the upcoming section on “Lists”.

Use italic to introduce new terms. See how we used italic to introduce the named styles Strong and Emphasis in this section’s first paragraph. Italicize technical terms on first use, and provide a definition as well. From then on, you’re free to write those terms in normal text.

Remove bold and italic formatting by selecting the text in question and pressing CTRL-Spacebar to clear the formatting. You may also be able to click the Clear All option at the top of the style panel, though that option may not function in some Mac OS X versions of Word.

# Code Examples

Lines of code can take the form of numbered and unnumbered examples. You may also include short snippets of code within body text paragraphs. Think carefully about whether to number examples, and be consistent in your approach throughout your book.

## Numbered Examples

Listing 1-1 is a numbered example. It shows a simple query in the Oracle Database dialect of SQL. Create a numbered example by writing a Caption paragraph as shown, followed by one or more paragraphs in the Code style. Always introduce numbered examples by citing their number and saying what they do, as this paragraph does in the first two sentences.

Listing 1-1. A simple query in Oracle SQL

SELECT \*

FROM dual;

Consistency is everything in writing. If you choose to number your examples, then you must do so in a way that is consistent as perceived by your readers. For example, some authors choose to show intermediate snippets in a section as unnumbered examples, building up to a final example at the end of the section that is numbered.

## Unnumbered Examples

You can choose to show a snippet of code without numbering it. Following is the first line of the query from the previous section. This time there is no caption and number.

SELECT \*

Terminate your introductory sentence with a colon when that sentence leads directly into an unnumbered listing. For example, the following version of the query specifies the column name:

SELECT dummy

FROM dual;

The introduction to the first listing is the sentence “Following is the first line…”. That sentence does not terminate directly into the code example, so there is a period at the end of the sentence, and at the end of the paragraph. The introduction to the second listing is the sentence “For example, the following version….”. That sentence does terminate directly into the code example, hence the colon.

**Tip:** Never use a colon to lead into a numbered listing. A colon is only ever appropriate when it is at the end of an introductory sentence that is followed immediately by lines of code.

Listing 1-2 shows a final version of the query that returns the current date and time. It’s common in Oracle Database to invoke queries against dual to evaluate built-in functions such as sysdate.

Listing 1-2. Invoking dual to evaluate a function

SELECT sysdate

FROM dual;

The progression in this section shows how you might show a series of unnumbered examples leading up to a final example at the end of a section that is numbered.

## Tabs and Soft Returns

Avoid tab characters and soft returns in your code examples. Some programming editors align columns of code via tab characters, and those get carried over into your chapter file when you copy and paste. Tab characters especially are trouble, as they can make your code appear wildly different in PDF files and the various eBook readers on the market.

We recommend configuring Word to show tabs, paragraph marks, and soft returns. Doing so allows you to be sure of where your paragraphs really end, and you’re better able to spot unwanted tab and soft return characters.

## Code in Body Text

You’ll often refer to keywords, variable names, properties of a class, and sometimes even expressions from within body text. For example, you might want to explain the sysdate function from Listing 1-2. When referring to keywords and identifiers such as SELECT and sysdate, or to fragments of code like the from dual clause, highlight the code in question and apply the Code Inline style. Doing so causes the code to appear in a fixed-width typeface, helping readers distinguish your code from the body text surrounding it.

# Figures

Every figure in your book should be numbered, captioned, and introduced. You’ve seen several correct examples already in this template. Most of our books have some combination of screen shots and diagrams. Some of our books have photographs as well.

Figure 1-8 shows how to insert a figure into your text. Create a Figure paragraph followed by a Caption paragraph. Place your image into the Figure paragraph. Format the figure's text-wrapping style as "In line with text" to have the figure treated as a single character within its paragraph.

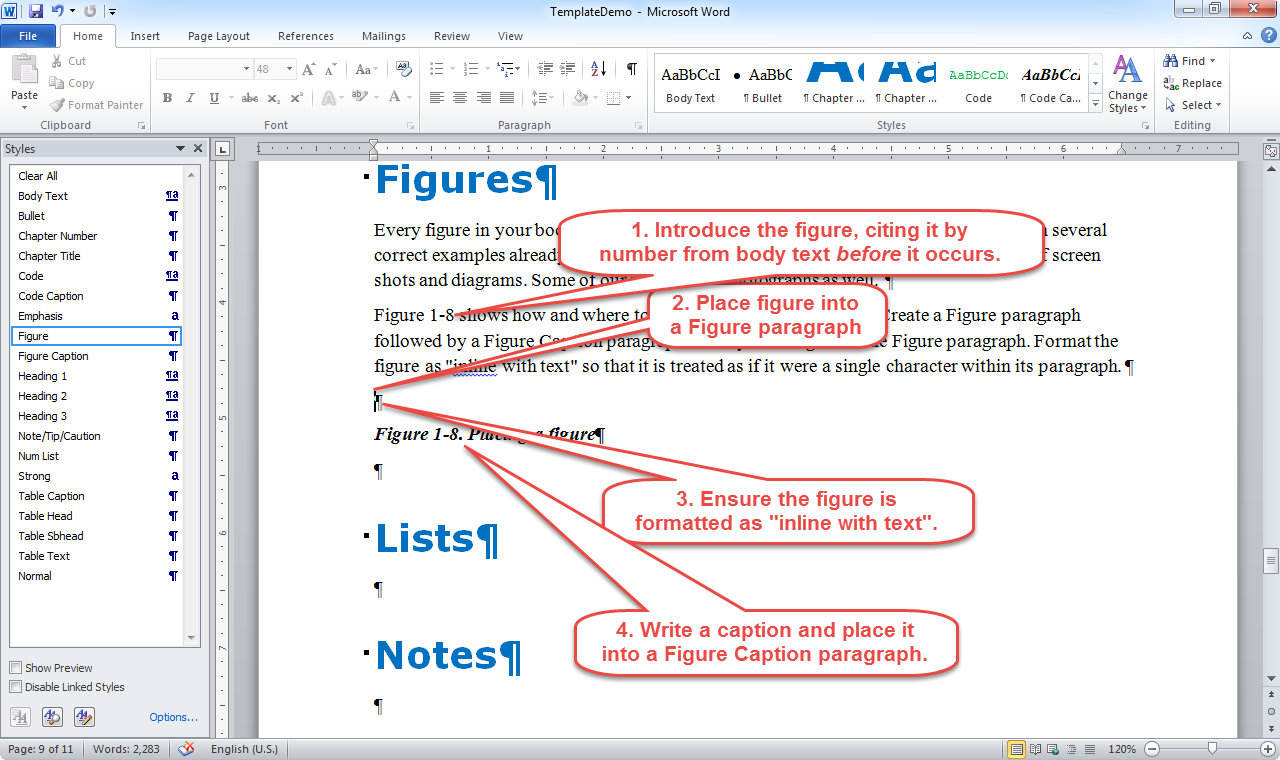


Figure 1-8. Placing a figure

Consistency is everything in writing. Strive for a consistent look and feel to your figures, including their size and scale. For example, screen shots in this template are shot with Word maximized on a 1280x800 pixel display. Other images are designed so that their text is not garishly out of line with the screenshots. The ribbon images in Figures 1-2 and 1-3 are exceptions that are at least kept consistent with each other.

**Note:** While we wrote much of this template's content on a Mac, we returned to Windows each time for screenshots. We used the same version of Word each time, and created our callouts in SnagIt using the same callout shape and color and font size each time. Consistency begets consistency.

Introduce your figures! The #1 mistake authors make with respect to figures is a failure to introduce. Each figure in this chapter is introduced from a Body Text paragraph prior to the figure's occurrence in the text. Do the same in your own writing. Never allow readers to encounter a figure without you first having told them what the figure is and what they should see in it and take away from it.

Format your figures so that they are "characters" in line with your text. Figure 1-9 shows the dialog in Word 2010 for Windows from which to specify "In line with text" as the Text Wrapping option. Each version of Word implements this functionality in slightly different ways from slightly different menus and dialogs. Find how to specify "In line with text" from your version of Word, and choose that option for each figure that you insert into a chapter.

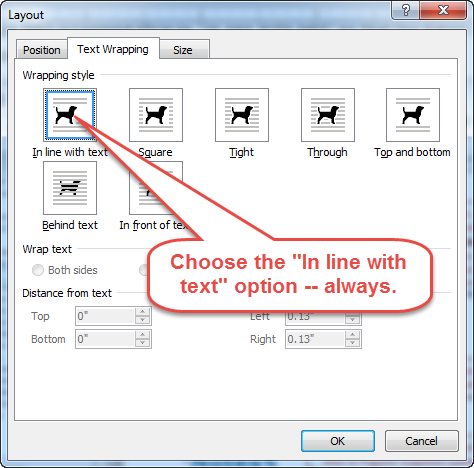


Figure 1-9. Choosing the correct text-wrapping option

Some of your figures may be diagrams. Use any drawing tool but Word to create those. Do not draw diagrams in Word itself, because Word treats each line and box and bit of text in such figures as independent elements that inevitably get mangled and thrown randomly across your pages as chapters get passed around, reviewed, and edited. Create diagrams using an external tool such as Lucidchart or Visio, or Gimp. Never use Word as a drawing tool. Never.

Be careful with color. Most eBook readers will see your figures in color, but print readers and some eBook readers see only shades of grey. Choose colors for your figures that convert well to greyscale. Print some images onto a standard, office laser printer as a test. Colors that don’t contrast well when printed in greyscale on your office printer won’t contrast any better in the printed book. Check your color choices and choose wisely. Limit the number of colors, and keep to a consistent set of colors across all your figures.

# Figure Files

Word is a one-way street. Place an image into Word, and it's impossible to get the original image back out again. We thus need all figures for a chapter in separate image files, and we recommend a workflow such as the following:

1. Capture a screenshot, or draw a diagram
2. Save the image to a PNG file
3. Insert the contents of that file into your chapter

Name your figure files as follows:

* Figure0101.png
* Figure0102.png
* Figure0103.png
* …

This naming convention gives us enough to determine chapter number and figure number for each image file that you give us. We like PNG files, but JPEG files are ok too.

Then give us two files whenever you submit a chapter:

* The chapter in Word, forming the name from the ISBN followed by the chapter number. For example: 978-1-4842-0062-9\_ch01.doc.
* A .ZIP file containing all the images for the chapter. For example: 978-1-4842-0062-9\_ch01\_figures.zip.

Give us new files anytime that you add, subtract, or reshuffle the figures within a chapter. Make sure that the names of your separate image files are always in synch with how the respective figures from those files are numbered and used within a chapter.

**Note:** You can begin to see that it's a fair bit of work to renumber images within a chapter. It's often better to avoid the need for renumbering. If you do have to renumber though, do the full job. Don't change the numbers in Word and leave us with a bunch of mismatched image files to sort out.

# Lists

We support three list types: bulleted, numbered, and term/definition. Each has its range of purpose, and it's important to know how to choose between them.

A good heuristic is to default to a bulleted list unless you have reason to do otherwise. For example, you can apply the Bullet style to a list of ingredients to a hamburger:

* Top bun
* Bottom bun
* Burger patty
* Cheese slice
* Onion slice
* Pickle
* Ketchup, mustard, and other condiments

If the ordering of list items is important, and only if it is important, you can choose to use a numbered list. For example, apply the Num List style to a list of steps in building a burger:

1. Place the bottom bun onto a small plate
2. Place the burger patty onto the bun
3. Top the patty with a slice of cheese
4. Sink an onion slice into the melted cheese
5. Spread on any condiments such as ketchup and mustard
6. Top it all off with a pickle
7. Close the burger by adding the top bun
8. Enjoy!

Word tends to continue numbering from one list to the next. When you begin a new list, you may need to right-click the first item and select Restart at 1. It is sometimes helpful to have a list continue the numbering from a prior list, but in the main you should begin each new list at 1.

Last is the term/definition list. Use this form when you have a term or a short phrase that is followed by a longer explanation. Apply the Bullet style, and additionally apply Strong (for bold) to the term or short phrase that begins each item. For example:

* **Top and bottom buns.** These are the slices of bread between which all other items are placed. The top bun will be the one that is gently crowned, often soft and puffy, and pleasing to look at. The bottom bun will be the less-elegant-looking piece that you'll want to hide at the bottom where it's out of sight.
* **The burger patty.** This is made of ground beef or similar. It forms the core of the sandwich. Vegan patties are an option for vegans and vegetarians.
* **Cheese slice.** Good old, American cheese is our favorite. But change things up now and then with a slice of Provolone or Baby Swiss.
* **Onion slice.** The thicker, the better, again in our opinion. Some like to Sautee the slice in butter. Fast-food restaurants often use chopped or diced onions instead.
* **Condiments.** These can include ketchup and mustard. Mayonnaise is sometimes used, and some restaurant chains will add on a "special sauce" that is reminiscent of Thousand Island salad dressing.
* **The pickle.** You are either a pickle person, or you are not. We like Claussen-brand pickles, which are always uncooked and kept refrigerated. They are easily available, and have a fresh taste not distant from the original cucumber.

Consistency is everything in writing. Make sure each item in a given list follows the same grammatical form. For example, each of our prior bulleted list items is a noun, while each step in our numbered list begins with a verb. Aim for a reasonably consistent length too. You can have some variation in length as in our term/definition example, but keep it within bounds.

**Tip:** Programmers can think of a list as representing a class. Each item in the list is then an instance of the class, and must follow the same physical and grammatical form.

# Notes

Notes are side comments that you make while writing your main content. They stand outside the main flow of the text and add color and interest. Sometimes they provide advice, or warn readers against making mistakes.

**Caution:** Write notes sparingly. Too many notes indicate poor organization of your content.

We support three main styles of note:

* **The plain note by which you provide an interesting tidbit of information.** For example, you might want to provide a URL to an interesting article or blog post bearing on your topic.
* **The tip by which you provide actionable advice to readers.** For example, you might suggest that they limit themselves to no more than one note per section, and further that such notes look best when they come at the very end of a section.
* **The caution by which you warn readers against making a mistake.** A good example would be a note warning readers against overly-long notes. Notes that are longer than a sentence or two strongly indicate a need to pay more attention to organizing and sequencing your content so as to present more of it in body text form.

Create a note paragraph by applying the Note/Tip/Caution style. Then write "Note: ", "Tip:", or "Caution:", as appropriate. Follow the colon with the text of your note. Keep the note short.

**Note:** One-sentence paragraphs also indicate poor chapter organization. Editors will sometimes take an easy out by converting one-sentence paragraphs into notes, provided there aren't too many of them.

Be sparing! An occasional note adds interest. Too many notes add clutter. This section is a good example of clutter. The term/definition list is enough to add interest. The three notes in addition to the list are too much. They overwhelm the body text and lend a cluttered and disorganized feel to the content. The number of context switches between different paragraph styles will cause readers to stumble and not grasp the content as easily as if more of it were in body text.

**Tip:** Let body text carry the day in your writing. Avoid clutter from too many paragraph types in too short of a space. Let this section be a lesson.

# Tables

The need to present information in tabular form is common in technical writing. There's a contradictory need for simplicity due to the rise of eBooks versus print, and the number of readers who will be reading your content on relatively "narrow" devices such as the Amazon Kindle reader. Favor simplicity, and the fewer the columns the better.

Table 1-1 shows a table that's too wide for comfort. The number of columns won't fit nicely on the printed page, and the result on an eBook reader such as the Kindle will be just awful. Your audience will not thank you for the mess that you'd be forcing them to parse and disentangle.

Table 1-1. An overly-wide list of burger recipes

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bun | Patty | Cheese | Onion | Mustard | Ketchup | Mayonnaise | Pickle |
| Soft White | Ground Beef | American | Sweet | Spicy Brown | Tomato | Hellmann's | Claussen |
| Whole Wheat | Morning Star Asian Veggie | None | Red | Honey or Dijon | None | Vegenaise Eggless | None |
| Pretzel Roll | Ground Lamb | Baby Swiss | Sweet | Whole Grain | None | None | None |

Your job as an author is to care for your readers and how they see your content no matter how they choose to read it. Keep in mind the narrow screen on a Kindle held vertically. Spend some time in getting creative about presenting information form that’ll be pleasing on a narrower display than print. For example, you can combine columns and take the approach in Table 1-2.

Table 1-2. A more readable list of burger recipes

|  |  |  |
| --- | --- | --- |
| Bun & Patty | Cheese & Onion | Condiments |
| Soft White  Ground Beef | American Cheese  Sweet Onion | Spicy Brown Mustard  Tomato Ketchup  Hellmann's Mayonaisse  Claussen Pickle |
| Whole Wheat  Morning Star Asian Veggie | No Cheese  Red Onion | Honey or Dijon Mustard  Vegenaise Eggless |
| Pretzel Roll  Ground Lamb | Baby Swiss  Sweet Onion | Whole Grain Mustard |

Tables 1-1 and 1-2 have column headings. It's sometimes helpful to have a secondary set of headings running down the left side. Table 1-3 shows another way to list those burger recipes, and this time each is given a name by which you can refer to it later.

Table 1-3. Secondary headings in the left-side column

|  |  |  |
| --- | --- | --- |
| Name | Main Ingredients | Condiments |
| Standard Burger | Soft White Buns  Ground Beef  American Cheese  Sweet Onion | Spicy Brown Mustard  Tomato Ketchup  Hellmann's Mayonaisse  Claussen Pickle |
| Vegan Burger | Whole Wheat Buns  Morning Star Asian Veggie Burger  No Cheese  Red Onion | Honey or Dijon Mustard  Vegenaise Eggless |
| Lamb's Delight | Pretzel Roll Bun  Ground Lamb  Baby Swiss  Sweet Onion | Whole Grain Mustard |

We support the following styles for use in creating tables, which we've used to create the three tables in this section. Help us by applying these same styles to your own table cells so that we know which cells are headings, which cells are side-headings, and which cells are data being presented.

* **Caption.** Apply this style to the line just above the table that gives the table's number, and the caption describing its contents.
* **Table Head.** Apply this style to your column heading cells. These should always be just one row at the very top.
* **Table Text.** Apply this style to all other cells.

Keep tables simple. Avoid anything more complex than Table 1-3. Talk to your editor if you do need something more complex, but know that you're doing a disservice to your eBook readers by creating anything other than a dead-simple table with just a few columns.

Don't try to apply final formatting. Trust us to do that. Just get your tables into the Word file in a default format that includes the border grid lines between cells, apply the preceding styles as appropriate, and let us do the rest.

**Note:** Microsoft Word fails in providing a mechanism by which to designate an amount of whitespace following a table and prior to the next paragraph of text. Your first body text paragraph following a table will butt right up against the bottom of that table, and that looks awful in the template, but rest assured that our production department will add spacing when the book is converted from Word into the publishing software that we use to produce our printed and electronic books. P.S., Give the Word team a nudge on this issue if you happen to work for Microsoft.

# Font Conventions

Consistency is everything in writing, and that extends to the conventions by which you identify file names, URLs, class names, procedure and function names, variable names, and so forth when such things appear in body text paragraphs.

Following are some conventions that we recommend:

* **New terms.** Italicize new terms on first use, and provide a quick definition. For example, a bulleted list is a list in which each item is introduced by a round dot.
* **File names.** Italicize file names such 978-1-4842-0062-9\_ch01.doc when they appear in a sentence.
* **URLs.** Treat URLs such as [www.apress.com](http://www.apress.com) and <http://www.springer.com> the same as file names. Italicize them so as to distinguish them from the surrounding text.
* **Class and variable names.** Italicize class names, variable names, keywords, and other elements of code that appear in body text. For example, an author writing on SQL is would have a lot to say about writing where clauses.
* **Snippets of code within a sentence.** A common query in Oracle Database is to select sysdate from dual, and we would render that query in a sentence using a fixed-width typeface.

Think about case as well. Some authors on SQL topics choose to uppercase SQL keywords in code examples and text by writing, for example, SELECT sysdate FROM dual. Be consistent! If you choose to go down this path, then make a point to uppercase consistently.

# Restrictions and Troubleshooting

We keep our template restricted so that you can’t inadvertently redefine our styles, or add new ones that clutter the list in the Styles panel. Our doing so protects you from Word’s propensity to automatically create all manner of unwanted and confusing styles that end up making your job as author that much harder.

However, you might need to make changes. In that event, the password to unlock the template is “Apress”. Use that password if you need it, but restrict the template again as soon as possible.

Remember the Clear All option to remove all but the default formatting for the paragraph being edited. This is a handy option for when you inadvertently press CTRL-B or CTRL-I to remove bold or italic formatting, only to realize that the Strong on Emphasis style is still applied to your text. Highlighting text and pressing Clear All resets that text back to the paragraph formatting.

Finally, we’ve overridden the Normal style so as to generate double-lines through any text that is styled using a style we do not support. If you see double-lines, then check to make sure you’ve applied one of our supported styles to that text. Double-lines through your text are a warning that you’ve not applied the correct style.