

Advanced Excel Techniques

OAIR Conference 2007
Columbus, OH

Ashutosh Nandeshwar



Contents

1	Shortcuts	1
1.1	Take Control of the “Ctrl” Key	1
1.1.1	Cut-Copy-Paste	1
1.1.2	Move Around	1
1.1.3	Misc	1
1.2	General	2
1.3	Names	4
2	Formulae	5
2.1	IF	5
2.2	VLOOKUP	7
2.3	COUNTIF and SUMIF	8
2.4	MATCH, INDEX, and OFFSET	8
2.5	SUMPRODUCT	11
3	Magic of Array Formulae	15
3.1	Count Unique Rows	15
3.2	Count Duplicates	16
3.3	Average	16
3.3.1	Average Excluding Zeros	16
3.3.2	Average of n Large or Small Numbers	17
4	Visual Basic for Applications (VBA)	19
4.1	Where to Put the Code?	19
4.2	Macro Recorder	19
4.3	Hello World!	20
4.4	Autofit Columns	20
4.5	Manual Calculation	21
4.6	Footer	22
4.7	Print Formula	22
4.8	User-Defined Function (UDF): Reverse String	23
4.9	Concatenate Range	24
4.10	Flip a Range	25
4.11	Fill Blank Cells	27
5	Resources	29
	Bibliography	31

A Keyboard Shortcuts from cpearson.com

33

List of Figures

1.1	Fixed Number of Zeros using TEXT and REPT Function	3
1.2	Quick tool	3
1.3	Creating Names	4
2.1	Simple IF formula	5
2.2	IF with AND and nested IF	6
2.3	IF formula with AND and OR	6
2.4	Nested IFs	6
2.5	VLOOKUP Formula with ISNA	7
2.6	Vlookup Formula with Range	8
2.7	Examples of COUNTIF Function	9
2.8	Examples of SUMIF Function	9
2.9	Example of MATCH Function	10
2.10	Example of INDEX Function	10
2.11	Using MAX, MATCH, and INDEX Functions	11
2.12	Example of OFFSET Function	12
2.13	Example of SUMPRODUCT and SUM as an Array Formula	12
2.14	Colors of Range Names	13
2.15	Example of SUMPRODUCT and SUM for Multiple Conditions	13
2.16	Array Coercion Example	14
3.1	Example of AVERAGE Function Excluding Zeros	16
3.2	AVERAGE and SUM of Top Three and Bottom Three	18
4.1	Steps to Create Custom Toolbar Buttons for Macros [1]	21
4.2	Example of Reverse String UDF	23
4.3	Example of Concatenate Range Function	24
4.4	Example of Flip Procedure	25
4.5	Example of Fill Blanks Procedure	27

Listings

4.1	Hello World Procedure	20
4.2	Autofit Column and Unwrap text	20
4.3	Manual Calculation Macro	22
4.4	Procedure to Set Footer	22
4.5	Print Formula in Adjacent Cells	22
4.6	Reverse String UDF	23
4.7	Palindrome Function	24
4.8	Concatenate Range Function	24
4.9	Procedure to Flip a Range	26
4.10	Procedure to Fill the Blanks	28

Chapter 1

Shortcuts

1.1 Take Control of the “Ctrl” Key

1.1.1 Cut-Copy-Paste

To paste special (values): for 2003 and lower, follow this sequence- Ctrl + C, Ctrl + V, Alt + E, V, Enter

1.1.2 Move Around

Note: To select data between current position and position you want to go to, hold the Shift key

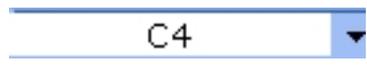
- To go to cell A1- Hold Ctrl and Press Home
- To go to the last cell in the data- Hold Ctrl and Press End
- To go to the last non-blank cell in a column from top- Hold Ctrl and Press Down Arrow
- To go to the last non-blank cell in a column from bottom- Hold Ctrl and Press Up Arrow
- To go to the last non-blank cell in a row from left- Hold Ctrl and Press Right Arrow
- To go to the last non-blank cell in a row from right- Hold Ctrl and Press Left Arrow
- To go to the next worksheet- Hold Ctrl and Press Page Down
- To go to the next worksheet- Hold Ctrl and Press Page Up

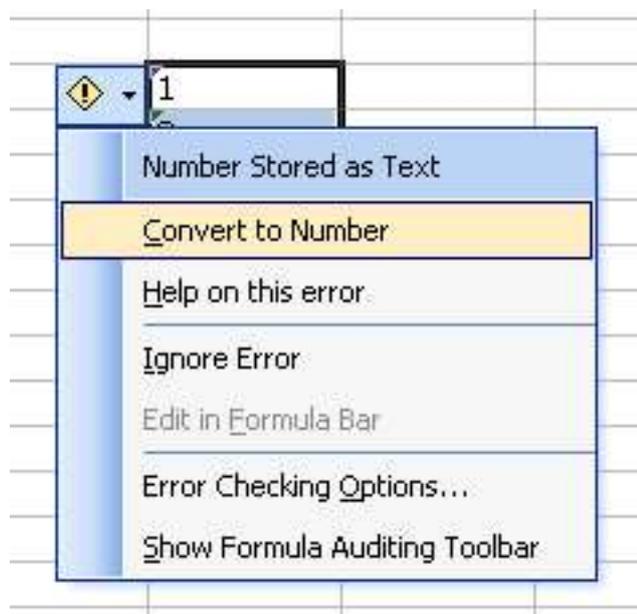
1.1.3 Misc

- To select the current row- Hold Shift and Press Spacebar
- To select the current row- Hold Ctrl and Press Spacebar
- To format the current selection- Hold Ctrl and Press 1
- To zoom in or out in the worksheet- Hold Ctrl and Scroll mouse wheel up or down

- To view the formulae instead of their values- Hold Ctrl and press tilde (~)
- To force a new line in the text- Hold Alt and press Enter
- To change the reference (relative to absolute) of a range in a formula- Select the range in the formula and Hit F4 to cycle through the references

1.2 General

- To fill the data or formula down from the top cell to all the cells below- Double-click the thick plus() symbol at the right-bottom corner of the cell
Note: filling-down will stop whenever a blank cell on the left-side is encountered
- To fill the data or formula, when double-clicking does not work, follow these steps:
 1. Go to the first cell that you would like to copy or fill in other cells
 2. Press F2 to edit the formula in that cell
 3. In the name box , type in the address of the last cell of the range, where you want to fill data or formula, then Hit Enter while Holding the Shift key
- To convert “textnumbers” to numbers use any of these methods:
 1. Type 0 in a blank cell, copy that cell, and Paste Special > Add over the range, which has “textnumbers”
 2. Select the column that has “textnumbers”, and use Text to Columns on this range (without providing any delimiters)
 3. Use SmartTags



- To get fixed number of zeros in front of a number use this formula

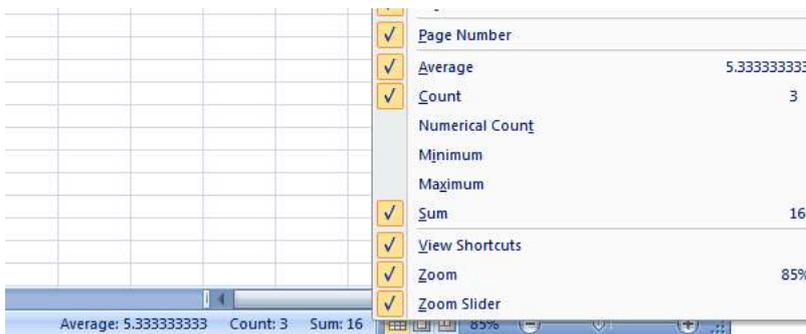
=TEXT(value,REPT("0",n))

where value is the number that you want to format and n is the number of zeros you want in front of the number. See Figure 1.1.

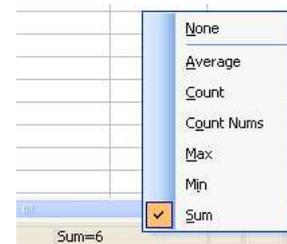
	A	B	C
54	Number	Formatted	Formula
55	589	000000589	=TEXT(A55,REPT("0",9))
56	55555	000055555	=TEXT(A56,REPT("0",9))
57	999999999	999999999	=TEXT(A57,REPT("0",9))

Figure 1.1: Fixed Number of Zeros using TEXT and REPT Function

- To get the unique records from the data follow these steps:
 1. Select the column or click a cell in the range or list you want to filter
 2. On the Data menu, point to Filter, and then click Advanced Filter
 3. Do one of the following:
 - (a) To filter the range or list in place, similar to using AutoFilter, click Filter the list, in-place
 - (b) To copy the results of the filter to another location, click Copy to another location. Then, in the Copy To box, enter a cell reference
 4. Select the Unique records only check box
- Quick Tool provide stats, such as Average, Count, Count Nums, Max, Min, and Sum of the data from a selected range without entering any formulae. To change the calculation, right click on the bottom toolbar and choose the desired statistic, see Figure 1.2



(a) Quick tool in 2007



(b) Quick tool in 2003

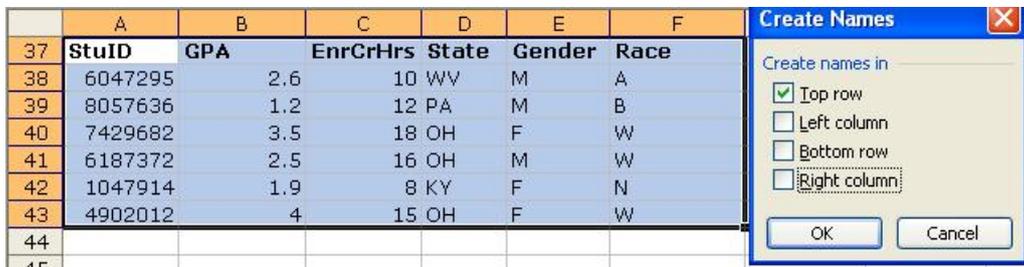
Figure 1.2: Quick tool

- To view statistics, such as Sum, Average, Max, and Min on the data with AutoFilter on, use the SUBTOTAL function. For example, =SUBTOTAL(9,A1:A20) will return the sum of the cells from the range A1 to A20 that are filtered and visible.

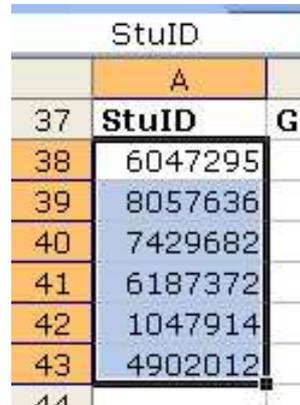
1.3 Names

Names are helpful, as they are easy to remember, and rather than providing a reference to a range, we can directly refer to them by using names. We can assign names to formulae and range. Names are specially useful when you have big data table, and you would like to do counts and sums, or use other formulae. One quick way to define names is using the “Create Name” feature (see Figure 1.3(a)). To use this follow these steps:

1. Select your data (shortcut: Ctrl + A, or go to the first cell (Ctrl + Home) and press Ctrl + End)
2. Click on the Insert Menu, then click on the Name option, and then click on Create, or Ctrl+Shift+F3
3. Check the Top row box to give names to the columns of data



(a) Create Names Box



(b) Name in the Name Box

Figure 1.3: Creating Names

After the names are created, if you select the complete data range, it would show the name of the range in the name box, see Figure 1.3(b).

Chapter 2

Formulae

2.1 IF

The syntax of an IF formula is:

```
IF(logical_test,value_if_true,value_if_false)
```

For example, if we create an indicator for non-residents, and the condition is that if the permanent state is not equal to OH, then non-resident indicator is equal to Y, else N. We can write this formula two ways (See Figure 2.1):

```
=IF(B2<>"OH", "Y", "N")  
=IF(B3="OH", "N", "Y")
```

	A	B	C	D
1	StuID	PermanentState	Non-ResidentInd	Formula
2	6047295	WV	Y	=IF(B2<>"OH", "Y", "N")
3	8057636	PA	Y	=IF(B3="OH", "N", "Y")
4	7429682	OH	N	=IF(B4<>"OH", "Y", "N")
5	6187372	OH	N	=IF(B5<>"OH", "Y", "N")
6	1047914	KY	Y	=IF(B6<>"OH", "Y", "N")
7	4902012	OH	N	=IF(B7<>"OH", "Y", "N")

Figure 2.1: Simple IF formula

If we want to add some more conditions to IF formula, we can use AND, OR, or nested IFs (up to seven for 2003¹ and lower, and 64 for 2007².) For example, we want scholarship indicator to be Y if the GPA is greater than equal to 2.5 and need greater than equal to 500. This can be achieved two ways (see Figure 2.2):

1. Using AND

```
=IF(AND(B12>=2.5,C12>=500), "Y", "N")
```

2. Two nested IFs

```
=IF(B15>=2.5, IF(C15>=500, "Y", "N"), "N")
```

	A	B	C	D	E
11	StuID	GPA	Need	ScholarshipInd	Formula
12	6047295	2.6	600	Y	=IF(AND(B12>=2.5,C12>=500),"Y","N")
13	8057636	1.2	5000	N	=IF(AND(B13>=2.5,C13>=500),"Y","N")
14	7429682	3.5	200	N	=IF(AND(B14>=2.5,C14>=500),"Y","N")
15	6187372	2.5	1000	Y	=IF(B15>=2.5,IF(C15>=500,"Y","N"),"N")
16	1047914	2	500	N	=IF(B16>=2.5,IF(C16>=500,"Y","N"),"N")
17	4902012	4	2000	Y	=IF(B17>=2.5,IF(C17>=500,"Y","N"),"N")

Figure 2.2: IF with AND and nested IF

	A	B	C	D	E	F
20	StuID	GPA	EnrCrHrs	CrisesWithF	AtRiskInd	Formula
21	6047295	2.6	10	0	N	=IF(OR(AND(B21<2,C21<10),D21>1),"Y","N")
22	8057636	1.2	12	2	Y	=IF(OR(AND(B22<2,C22<10),D22>1),"Y","N")
23	7429682	3.5	18	0	N	=IF(OR(AND(B23<2,C23<10),D23>1),"Y","N")
24	6187372	2.5	16	1	N	=IF(OR(AND(B24<2,C24<10),D24>1),"Y","N")
25	1047914	1.9	8	1	Y	=IF(OR(AND(B25<2,C25<10),D25>1),"Y","N")
26	4902012	4	15	0	N	=IF(OR(AND(B26<2,C26<10),D26>1),"Y","N")

Figure 2.3: IF formula with AND and OR

We can build any condition for the first argument of the IF formula that results in a TRUE or FALSE. For example, if we create an “at-risk” indicator, based on two conditions:

1. If the current GPA is less than 2 **AND** current enrolled hours are less than 10, **OR**
2. Number of courses with F is greater than 1

We can build a formula using IF, AND, and OR to achieve this (see Figure 2.3):

```
=IF(OR(AND(B21<2,C21<10),D21>1),"Y","N")
```

Of course, we can create a lot more complicated IF formulae by nesting multiple IFs. For example, if we want to convert college codes to college description, we can use nested IFs to achieve that (see Figure 2.4); however, a better way to do that is using VLOOKUP.

```
=IF(A29="A","College of Arts",IF(A29="B","College of Business",IF(A29="T",
"College of Technology",IF(A29="E","College of Engineering","NA"))))
```

	A	B	C
28	CollegeCode	CollegeDesc	Formula
29	A	College of Arts	=IF(A29="A","College of Arts",IF(A29="B","College of Business",IF(A29="T","College of Technology",I
30	B	College of Business	=IF(A30="A","College of Arts",IF(A30="B","College of Business",IF(A30="T","College of Technology",I
31	T	College of Technology	=IF(A31="A","College of Arts",IF(A31="B","College of Business",IF(A31="T","College of Technology",I
32	E	College of Engineering	=IF(A32="A","College of Arts",IF(A32="B","College of Business",IF(A32="T","College of Technology",I

Figure 2.4: Nested IFs

¹<http://office.microsoft.com/en-us/excel/HP051992911033.aspx>

²<http://office.microsoft.com/en-us/excel/HP100738491033.aspx>

2.2 VLOOKUP

VLOOKUP formula searches for a value in the first column of a table array and returns a value in the same row from another column in the table array. The V in VLOOKUP stands for vertical. Use VLOOKUP instead of HLOOKUP when your comparison values are located in a column to the left of the data that you want to find. The syntax of VLOOKUP formula is:

VLOOKUP(lookup_value,table_array,col_index_num,range_lookup)

In the above example, our lookup value is the college code and table array is a table with the codes and its descriptions. The column index is the column that we want to return after matching a code, and the range lookup is an argument to search for exact matches (1 or TRUE) or approximate matches (0 or FALSE); however, if the search is for an approximate match, then the values in the first column need to be sorted in an ascending order. If the exact match argument is set to true, and if no match is found, this formula would return a #N/A error. To overcome this error we can nest an IF formula with ISNA function to check if the VLOOKUP formula causes an error. See Figure 2.5 and notice the shaded area for ISNA function.

```
=IF(ISNA(VLOOKUP(B50,$A$37:$B$40,2,0)),"Not found",
VLOOKUP(B50,$A$37:$B$40,2,0))
```

	A	B	C	D
35		Lookup Table		
36		CollegeCode	CollegeDesc	
37		A	College of Arts	
38		B	College of Business	
39		T	College of Technology	
40		E	College of Engineering	
41				
42	StuID	CollegeCode	CollegeDesc	Formula
43	6047295	A	College of Arts	=VLOOKUP(B43,\$B\$37:\$C\$40,2,0)
44	8057636	B	College of Business	=VLOOKUP(B44,\$B\$37:\$C\$40,2,0)
45	7429682	A	College of Arts	=VLOOKUP(B45,\$B\$37:\$C\$40,2,0)
46	6187372	T	College of Technology	=VLOOKUP(B46,\$B\$37:\$C\$40,2,0)
47	1047914	E	College of Engineering	=VLOOKUP(B47,\$B\$37:\$C\$40,2,0)
48	4902012	B	College of Business	=VLOOKUP(B48,\$B\$37:\$C\$40,2,0)
49	4902013	G	#N/A	=VLOOKUP(B49,\$B\$37:\$C\$40,2,0)
50	4902013	G	Not found	=IF(ISNA(VLOOKUP(B50,\$B\$37:\$C\$40,2,0)), "Not found",VLOOKUP(B50,\$B\$37:\$C\$40,2,0))

Figure 2.5: VLOOKUP Formula with ISNA

In Excel 2007, we can use function IFERROR to check if a formula is generating an error, and this function will evaluate the first argument, if it generates an error, the function will return the value supplied in the second argument.

```
=IFERROR(VLOOKUP(B49,$B$37:$C$40,2,0),"Not found")
```

We can use the approximate match argument when we want to return ranges. For example, if we want to return the range on GPA based on the following conditions:

1. if GPA is ≥ 0 and < 2 , the range is poor,
2. if GPA is ≥ 2 and < 2.5 , the range is OK,
3. if GPA is ≥ 2.5 and < 3 , the range is moderate,
4. if GPA is ≥ 3 and < 3.5 , the range is good,
5. if GPA is ≥ 3.5 , the range is very good

We can construct a lookup table as given in Figure 2.6(a), and use it in a VLOOKUP formula as given in Figure 2.6(b).

	A	B
60	GPA	GPARange
61		0 Poor
62		2 OK
63	2.5	Moderate
64	3	Good
65	3.5	Very Good

	A	B	C	D
67	StuID	GPA	GPARange	Formula
68	6047295	0	Poor	=VLOOKUP(B68,\$A\$61:\$B\$65,2,1)
69	8057636	1.2	Poor	=VLOOKUP(B69,\$A\$61:\$B\$65,2,1)
70	7429682	3.6	Very Good	=VLOOKUP(B70,\$A\$61:\$B\$65,2,1)
71	6187372	2.6	Moderate	=VLOOKUP(B71,\$A\$61:\$B\$65,2,1)
72	1047914	2	OK	=VLOOKUP(B72,\$A\$61:\$B\$65,2,1)
73	4902012	3.9	Very Good	=VLOOKUP(B73,\$A\$61:\$B\$65,2,1)

(a) Lookup Table

(b) Vlookup Formula

Figure 2.6: Vlookup Formula with Range

2.3 COUNTIF and SUMIF

COUNTIF function is useful when you quickly want to know a count based on a criterion (for multiple criteria use Pivot tables, use array formulae, or use COUNTIFS in 2007.) See Figure 2.7 for examples; the syntax of COUNTIF is:

COUNTIF(range,criteria)

SUMIF function is similar to the COUNTIF function; however, it offers two ranges—one to test the criterion, and the other to sum based on the criterion. If sum range is omitted this function will sum the criterion range. See Figure 2.8 for some examples; the syntax of SUMIF is:

SUMIF(range,criteria,sum_range)

2.4 MATCH, INDEX, and OFFSET

MATCH by itself is nothing special; it returns the position of an item in an array; however, using some creativity and other functions wonderful formulae can be written. The syntax of MATCH is:

MATCH(lookup_value,lookup_array,match_type)

	A	B	C	D
76	StuID	Gender	GPA	
77	6047295	M	1.79	
78	8057636	F	3.53	
79	7429682	M	2.50	
80	6187372	F	0.97	
81	1047914	F	1.74	
82	4902012	M	3.42	
83				
84	Question	Answer	Formula	
85	Number of females	3	=COUNTIF(B77:B82,"F")	
86	GPA > 2	3	=COUNTIF(C77:C82,">2")	
87	GPA > 2 and <= 3	1	=COUNTIF(C77:C82,">2")-COUNTIF(C77:C82,">3")	

Figure 2.7: Examples of COUNTIF Function

	A	B	C
89	StuID	State	FinAid
90	6047295	OH	\$2,000
91	8057636	WV	\$500
92	7429682	OH	\$600
93	6187372	PA	\$1,500
94			
95	Question	Answer	Formula
96	Financial aid for OH residents	\$2,600	=SUMIF(B90:B93,"OH",C90:C93)
97	Financial aid for non-OH residents	\$2,000	=SUMIF(B90:B93,"<>OH",C90:C93)

Figure 2.8: Examples of SUMIF Function

where `lookup_value` is the value we want to find in the lookup array, and `match_type` determines the match type between exact and approximate (greater or smaller) match. See Figure 2.9 for an example.

	A	B	C
99	CollegeCode		
100	A	MatchCode	T
101	B	Position	3
102	T	Formula	=MATCH(C100,A100:A103)
103	E		

Figure 2.9: Example of MATCH Function

INDEX function in an array form returns an item from an array when given the row and the column number. See Figure 2.10 for an example. The syntax of INDEX is:

`INDEX(array,row_num,column_num)`

	A	B	C
105	CollegeCode		
106	A	Row Number	3
107	B	Column Number	1
108	T	Value	T
109	E	Formula	=INDEX(A106:A109,C106,C107)

Figure 2.10: Example of INDEX Function

As you can see, MATCH and INDEX by themselves are not very useful; however, when we combine we can produce great results. For example, let's look at the survey, where we asked people to rate themselves (categories: Don't know, Newbie, Savvy, Expert) on Excel (concepts: Excel program itself, Conditional formatting, Spreadsheet formatting and working, Simple formulae, Advanced formulae, Filters, Advanced Filters, VBA (macros) Pivot tables, Data analysis), and I wanted to find the category that maximum people rated themselves, for each concept (see Figure 2.11). To achieve this I used this formula:

`=INDEX(B1:E1,0,MATCH(MAX(B2:E2),B2:E2))`

where the range B1:E1 has the categories, and the range B2:E2 had the ratings. For the concept "Excel program itself", maximum number (14) of people rated themselves as "Savvy", but to find this using a formula is a three step process:

1. Find out the maximum number of ratings for each concept, and this is done using the MAX function, which returns 14 for the range B2:E2.

`=MAX(B2:E2)`

2. Find out the position of this maximum value in the range B2:E2, and this is done using the MATCH function, which returns 3.

```
=MATCH(14,B2:E2))
```

3. Find out the category name for this column position from the range B1:E1, and this is done using the INDEX function, which returns “Savvy.”

```
=INDEX($B$1:$E$1,0,3)
```

	A	B	C	D	E	F	G
1	answer options	Don't know	Newbie	Savvy	Expert	Max	Formula
2	Excel program itself	0	0	14	5	Savvy	=INDEX(\$B\$1:\$E\$1,0,MATCH(MAX(B2:E2),B2:E2))
3	Conditional formatting	1	11	6	1	Newbie	=INDEX(\$B\$1:\$E\$1,0,MATCH(MAX(B3:E3),B3:E3))
4	Spreadsheet formatting and working	0	2	11	6	Savvy	=INDEX(\$B\$1:\$E\$1,0,MATCH(MAX(B4:E4),B4:E4))
5	Simple formulae	0	0	11	8	Savvy	=INDEX(\$B\$1:\$E\$1,0,MATCH(MAX(B5:E5),B5:E5))
6	Advanced formulae	1	10	6	2	Newbie	=INDEX(\$B\$1:\$E\$1,0,MATCH(MAX(B6:E6),B6:E6))
7	Filters, Advanced Filters	0	9	9	1	Savvy	=INDEX(\$B\$1:\$E\$1,0,MATCH(MAX(B7:E7),B7:E7))
8	VBA (macros)	3	15	1	0	Newbie	=INDEX(\$B\$1:\$E\$1,0,MATCH(MAX(B8:E8),B8:E8))
9	Pivot tables	2	4	8	5	Savvy	=INDEX(\$B\$1:\$E\$1,0,MATCH(MAX(B9:E9),B9:E9))
10	Data analysis	1	4	10	4	Savvy	=INDEX(\$B\$1:\$E\$1,0,MATCH(MAX(B10:E10),B10:E10))

Figure 2.11: Using MAX, MATCH, and INDEX Functions

OFFSET returns a reference to a range that is a specified number of rows and columns from a cell or range of cells. In a sense, this function is similar to the INDEX function; however, rather than returning a value from an array, OFFSET returns a cell or range reference. The syntax of OFFSET function is:

```
OFFSET(reference,rows,cols,height,width)
```

When I am using the values from a PivotTable to move to a nicer format, I have found particular use of the OFFSET function. For example, consider the table obtained in Figure 2.12(a), and we would like to report the yield percentages in a nicer format as shown in Figure 2.12(b). This is achieved by using this OFFSET formula, and dragging it across and down:

```
=OFFSET($B15,0,B$21)
```

2.5 SUMPRODUCT

SUMPRODUCT function multiplies the input arrays, and then adds them. The syntax of SUMPRODUCT is:

```
SUMPRODUCT(arr1,arr2,...,arr30) for Excel 2003
```

```
SUMPRODUCT(arr1,arr2,...,arr255) for Excel 2007
```

	A	B	C	D	E	F	G	H	I	J
13	2004				2005			2006		
14	College	Applied	Enrolled	Yield	Applied	Enrolled	Yield	Applied	Enrolled	Yield
15	A	10	5	50.00%	12	8	66.67%	9	5	55.56%
16	B	20	4	20.00%	18	5	27.78%	21	6	28.57%
17	T	50	25	50.00%	55	30	54.55%	52	31	59.62%
18	E	80	40	50.00%	70	35	50.00%	66	30	45.45%
19	N	5	2	40.00%	10	2	20.00%	12	4	33.33%

(a) Pivot Table Data

	A	B	C	D
21	OffsetColumn	2	5	8
22	College	2004	2005	2006
23	A	50.00%	66.67%	55.56%
24	B	20.00%	27.78%	28.57%
25	T	50.00%	54.55%	59.62%
26	E	50.00%	50.00%	45.45%
27	N	40.00%	20.00%	33.33%

(b) Using OFFSET Function

Figure 2.12: Example of OFFSET Function

Figure 2.13 is an example of SUMPRODUCT function, and SUM formula used as an array formula. Array formulae are entered by pressing SHIFT, CONTROL, and ENTER at the same time. Results of both formulae are achieved in two-steps:

1. Multiply each item of the arrays:

$$\begin{aligned}
 5 \times 6 &= 30 \\
 7 \times 8 &= 56 \\
 6 \times 9 &= 54 \\
 9 \times 5 &= 45
 \end{aligned}$$

2. Sum these values:

$$30 + 56 + 54 + 45 = 185$$

	A	B	C	D	E	F	G
29	X			Y			
30	5	7		6	8		
31	6	9		9	5		
32							
33	Product of X and Y and Sum		SUMPRODUCT	185	{=SUMPRODUCT(A30:B31,D30:E31)}		
34			SUM	185	{=SUM(A30:B31*D30:E31)}		

Figure 2.13: Example of SUMPRODUCT and SUM as an Array Formula

The real magic of array formulae is in counting and summing based on multiple conditions. For example, if we want to count number of students based on some criteria, we can use either SUMPRODUCT, or SUM as an array formula. We will use the same data (1.3(a)) and the names created in Section 1.3. Let’s create counts for following conditions:

1. State is OH AND GPA is greater than 2.4 AND Gender is M. We can use these formulae to find this count:

```
=SUMPRODUCT(--(State="OH"),--(GPA>2.4),--(Gender="M"))
{=SUM((State="OH")*(GPA>2.4)*(Gender="M"))}
```

2. Enrolled hours are greater than 10 AND Gender is F AND Race is W. We can use these formulae to find this count:

```
=SUMPRODUCT(--(EnrCrHrs>10),--(Gender="F"),--(Race="W"))
{=SUM((EnrCrHrs>10)*(Gender="F")*(Race="W"))}
```

You can see in Figure 2.14 that as soon as you type the range name, Excel colors the range, which is stored in a name. See Figure 2.15 for above two formulae. When Excel

	A	B	C	D	E	F	G
37	StuID	GPA	EnrCrHrs	State	Gender	Race	
38	6047295	2.6	10	WV	M	A	
39	8057636	1.2	12	PA	M	B	
40	7429682	3.5	18	OH	F	W	
41	6187372	2.5	16	OH	M	W	
42	1047914	1.9	8	KY	F	N	
43	4902012	4	15	OH	F	W	
44							
45							
46	Count Required	Answer					
	State is OH AND GPA is greater than 2.4 AND Gender is M						
47		=SUMPRODUCT(--(State="OH"),--(GPA>2.4),--(Gender="M"))					
48		SUMPRODUCT(array1, [array2], [array3], [array4], [array5], ...)					
49							

Figure 2.14: Colors of Range Names

	A	B	C
46	Count Required	Answer	Formula
47	State is OH AND GPA is greater than 2.4 AND Gender is M	1	=SUMPRODUCT(--(State="OH"),--(GPA>2.4),--(Gender="M"))
48		1	=SUM((State="OH")*(GPA>2.4)*(Gender="M"))
49	Enrolled hours are greater than 10 AND Gender is F AND Race is W	2	=SUMPRODUCT(--(EnrCrHrs>10),--(Gender="F"),--(Race="W"))
50		2	=SUM((EnrCrHrs>10)*(Gender="F")*(Race="W"))

Figure 2.15: Example of SUMPRODUCT and SUM for Multiple Conditions

sees an array argument, such as State="OH", it evaluates each cell in that range with that condition, and returns an array with TRUE or FALSE values. The double dashes or minus signs (--) are used for coercion to convert the logical values to numbers. For example, if put one minus sign in front of a logical value TRUE, then Excel returns -1, and if put one more minus sign in front of -1, then Excel will return positive 1 (see Figure 2.16.) In our criterion of State="OH", Excel returns this array with logical values:

{FALSE; FALSE; TRUE; TRUE; FALSE; TRUE}

When we coerce it with one minus sign, Excel returns this array:

{0; 0; -1; -1; 0; -1}

After coercing it with two minus signs, Excel returns this array:

{0; 0; 1; 1; 0; 1}

In our example of State="OH", GPA > 2.4, and Gender="M", Excel creates three arrays:

{FALSE; FALSE; TRUE; TRUE; FALSE; TRUE}

{TRUE; FALSE; TRUE; TRUE; FALSE; TRUE}

{TRUE; TRUE; FALSE; TRUE; FALSE; FALSE}

When using double minus sign in SUMPRODUCT function, Excel first multiplies the arrays with numbers, and then adds them, resulting in a count. In this case, the calculation was like this:

1. Multiply each item in the arrays $0 \times 1 \times 1 = 0$
 - $0 \times 0 \times 1 = 0$
 - $1 \times 1 \times 0 = 0$
 - $1 \times 1 \times 1 = 1$
 - $0 \times 0 \times 0 = 0$
 - $1 \times 1 \times 0 = 0$
2. Add the result of multiplication $0 + 0 + 0 + 1 + 0 + 0 = 1$

	A	B	C	D	E	F	G
53	State	Criterion	Formula	SingleMinusSign	Formula	DoubleMinusSign	Formula
54	WV	FALSE	{=State="OH"}	0	=-B54	0	=--B54
55	PA	FALSE	{=State="OH"}	0	=-B55	0	=--B55
56	OH	TRUE	{=State="OH"}	-1	=-B56	1	=--B56
57	OH	TRUE	{=State="OH"}	-1	=-B57	1	=--B57
58	KY	FALSE	{=State="OH"}	0	=-B58	0	=--B58
59	OH	TRUE	{=State="OH"}	-1	=-B59	1	=--B59

Figure 2.16: Array Coercion Example

The array formula using SUM will do exactly the same thing, but there is no need of coercing the arrays, as it automatically converts logical values to numbers. If we wish we can enter a formula like this:

=SUMPRODUCT((State="OH")*(GPA>2.4)*(Gender="M"))

This formula will also return the exact same results; however, it is not proper form providing arguments to this function, and results in slower performance.

Chapter 3

Magic of Array Formulae

Let's see some magical things an array formula can do. As mentioned earlier, while writing formulae for counting, range names are very useful. Following examples are with range names. In addition, array formulae should be entered using Ctrl + Shift + Enter. The curly braces surrounding the formula will automatically appear; do not enter formula with the braces. Some of the array formulae are taken from Andrew's Tips web-site with permission. I would encourage readers to go to his web-site to find more interesting and useful ways of array formulae. <http://www.andrewsexceltips.com>

3.1 Count Unique Rows

In our range of states, if we want to find out the number of unique values, we can write a formula like this:

```
{=SUM(1/COUNTIF(State,State))}
```

The way this formula works is very intriguing; it works in three parts:

1. COUNTIF function counts the number of states matching the criterion. In this case, we are supplying full range as criteria, as a result of that Excel returns an array with the count of the appearance of each row of the range, therefore, we can get this array:

```
{1;1;3;3;1;3}
```

For the states: WV, PA, and KY, COUNTIF returns 1, as they appear only once; however, for OH, COUNTIF returns 3.

2. Calculate the division of 1 by the counts. This results in this array:

```
{1;1;0.3333;0.3333;1;0.3333}
```

3. Sum this array, which results in 4.

If we had 2 rows of KY, the division would be $1/2 + 1/2$, again resulting in one. By dividing 1 by the counts, it creates equal fractions and by summing these fractions we get one for each duplicated row.

3.2 Count Duplicates

To count duplicates we can use a formula like this:

```
{=SUM(IF(COUNTIF(State,State)>1,1,0))}
```

In this case, the COUNTIF function will return how many times a state has appeared, but the IF function will return one for counts more than one, else will return zero, therefore, if there are unique rows, the IF function will return zero, else the duplicate rows are counted and summed.

3.3 Average

3.3.1 Average Excluding Zeros

If you would like to find out average of a range excluding zeros, such a formula can be written:

```
{=AVERAGE(IF(B63:B67>0,B63:B67,""))}
```

This formula works in three parts:

1. The IF function tests the condition, and creates an array of logical values TRUE or FALSE.
2. If it (the IF function) finds cells with values greater than zero, it returns that cell, else it returns a blank string "", which the AVERAGE function ignores while calculating.
3. Find an average of the values in that array

For example, consider that data given in Figure 3.1. Let's assume that these are the number of applicants for each college by year, and we would like to find out the average of applicants for each year, excluding zero applicants for a college.

	A	B	C	D
62	College	2004	2005	2006
63	A	5	8	0
64	B	0	5	6
65	T	25	30	31
66	E	40	35	30
67	N	2	0	4
68	Average	14.4	15.6	14.2
69	AvgExcludingZeros	18	19.5	17.75

=AVERAGE(B63:B67)

{=AVERAGE(IF(B63:B67>0,B63:B67,""))}

Figure 3.1: Example of AVERAGE Function Excluding Zeros

Note: This logic can be used to construct formulae for finding average (or sum, max) based on different criteria(for Excel 2003 and below, as Excel 2007 has SUMIFS, AVERAGEIF, and AVERAGEIFS.) For example, to find average of GPA of the students, who are **not** from OH, we can construct a formula like this:

```
{=AVERAGE(IF(State<>"OH",GPA,""))}
```

To find maximum GPA of students, who are not from OH, we can construct a formula like this:

```
{=MAX(IF(State<>"OH",GPA,0))}
```

We can add multiple criteria to this formula. For example, if we want to find maximum GPA of students, who are from OH, and are males, we can construct a formula like this:

```
{=MAX((State="OH")*(Gender="M")*GPA)}
```

If we want to find minimum GPA of students, who are from OH, and are males, we can construct a formula like this:

```
{=MIN(IF(State="OH",IF(Gender="F",GPA,""),"))}
```

If we would like to ignore the errors from a range and find average of that range, we can construct a formula like this:

```
{=AVERAGE(IF(ISERROR(GPA),"",GPA))}
```

3.3.2 Average of n Large or Small Numbers

Suppose we have a list of application sources and number of applicants from each source (see Figure 3.2) and we want to find out average number of applicants from top three sources. To achieve this we can use following formula, where the range of applicants is named as "Applicants":

```
{=AVERAGE(LARGE(Applicants,{1,2,3}))}
```

This formula works in two parts:

1. The LARGE function finds out the n^{th} largest number from a given range. In this case, we have provided an array $\{1, 2, 3\}$ to find out the largest, the second largest, and the third largest number from the range. The LARGE function will return an array with top three large numbers. In this example, these numbers are 49,47, and 33.
2. Average function finds out the average of this array, which is 43 in this example.

Let's say that you want to find average for top eight sources, it would not be preferable to write an array like this $\{1, 2, 3, 4, 5, 6, 7, 8\}$, therefore, we can use the ROW function to generate an array of continuous numbers. For example, to generate an array of numbers from one to eight this formula can be used:

```
=ROW(1:8)
```

However, this formula would not work if you add or delete rows between row number one and eight, therefore, to make it nonvolatile use the INDIRECT function, such as:

```
=ROW(INDIRECT("1:8"))
```

	A	B	C
72	Source	Applicants	
73		1	26
74		2	33
75		3	47
76		4	17
77		5	28
78		6	22
79		7	49
80		8	20
81		9	14
82		10	20
83	Question	Answer	Formula
84	AverageTop3	43	{=AVERAGE(LARGE(Applicants,ROW(INDIRECT("1:3"))))}
85	SumTop3	129	{=SUM(LARGE(Applicants,ROW(INDIRECT("1:3"))))}
86	AverageBottom3	17	{=AVERAGE(SMALL(Applicants,ROW(INDIRECT("1:3"))))}
87	SumBottom3	51	{=SUM(SMALL(Applicants,ROW(INDIRECT("1:3"))))}

Figure 3.2: AVERAGE and SUM of Top Three and Bottom Three

The INDIRECT function translates the text argument to Excel ranges. For example, if cell A1 has the text “Sheet2”, then the formula =INDIRECT(A1 & “!A1”) will return the value of the cell A1 from Sheet2. The modified formula for finding average of eight largest sources:

```
{=AVERAGE(LARGE(Applicants,ROW(INDIRECT("1:8"))))}
```

Using the same logic, formulae for finding average of n small numbers, or sum of n small or large numbers can be constructed.

- Average applicants from bottom three sources:

```
{=AVERAGE(SMALL(Applicants,ROW(INDIRECT("1:3"))))}
```

- Sum of applicants from top three sources:

```
{=SUM(LARGE(Applicants,ROW(INDIRECT("1:3"))))}
```

- Sum of applicants from bottom three sources:

```
{=SUM(SMALL(Applicants,ROW(INDIRECT("1:3"))))}
```

Chapter 4

Visual Basic for Applications (VBA)

Once a person asked me, “why to use VBA?” I stumbled to give a reply, as VBA is such a neat solution for me to customize Excel, it was ingrained in me, and I never developed a justification for it; however, in retrospect, these are some of the reasons why one should use VBA:

- When you have repetitive tasks, such as formatting, file creation, data manipulation
- Formulae are too complex to remain explainable. If you write some formulae that you cannot explain to yourself after some time, then it is better that you code them
- Customizing Excel for your work environment to increase productivity, such as creating add-ins, creating user-defined functions (UDFs), creating utilities
- When it is simply not possible to achieve what you want using Excel’s built-in functions and utilities

A few pages on VBA from the book *Excel 2007 Power Programming with VBA* by John Walkenbach, pp 135-138 [2].

4.1 Where to Put the Code?

There are numerous places to put the code, such as modules, worksheets, Thisworkbook, and personal workbook. When you record a macro, Excel asks the location of the macro, whether it is a new workbook, current workbook, or personal workbook. By default, Excel inserts a module in the active workbook and puts the code in that module.

Personal workbook is a common and a hidden workbook for all other workbooks. It is created when you record a macro for the first time. If there are numerous macros that you use regularly, then it is a good idea to store them in the personal workbook. If you store different macros in different workbooks, then accessing them can be an issue; however, if you store them in personal workbook, then the macros can be accessed from any workbook.

4.2 Macro Recorder

Macro recorder is a very good way to learn the objects and its properties, methods, and actions. Although Excel’s macro recorder creates a lot of junk in the code, it is a very

good start to learn about VBA. When you start recording, it records each of your action to VBA equivalent; however, it can only record the actions that are doable using Excel's interface. For example, if you would like to create a table of contents of your sheets in a workbook, you cannot record such a macro, and you would need to write code for it.

4.3 Hello World!

A simple “hello world!” procedure to produce a message box is given in Listing 4.1. Write or copy this program in a worksheet's VBE. To open the Visual Basic Editor (VBE) press Alt + F11. You can run this procedure three ways:

1. From macro list (press Alt + F8 to bring up the list)
2. From VBE by selecting the procedure and clicking on the play button  (or Press F5)
3. Hitting the shortcut key, if one was assigned

Listing 4.1: Hello World Procedure

```
Sub HelloWorld ()
MsgBox "Hello world", vbOKOnly, "My First Program"
End Sub
```

The keyword **Sub** tells VBA that a procedure is starting, and the keyword **End Sub** tells VBA that procedure has ended. A function is a procedure, but unlike regular procedures it returns a value, and to start a function, **Function** keyword is used. The word that follows after **Sub** or **Function** keywords is that name of that procedure or function. You can insert a new procedure or a function by using the insert menu from VBE.

MsgBox is VBA's in-built function, which returns a message box with the input values of prompt, button type, and title. There are few more input arguments of the **MsgBox** function; however, often these arguments are sufficient. The syntax of **MsgBox** function is:

```
MsgBox(prompt[, buttons] [, title] [, helpfile, context])
```

4.4 Autofit Columns

Often when I receive an Excel file, the columns width is too small to see everything from a cell or the text is wrapped, that's why I created a small macro or procedure to autofit columns and remove wrapping of text. Listing 4.2 is the code for it.

Listing 4.2: Autofit Column and Unwrap text

```
Sub AutofitColumn ()
'Run autofit method for all columns
'Columns A to IV for Excel 2003 and below
'Columns A to XFD For 2007
Columns("A:IV").EntireColumn.AutoFit
'Set the property of wrapping text to false for all columns
```

```
Columns("A:IV").WrapText = False
End Sub
```

In this example, the procedure name is **AutofitColumn**. Comments are included in the code to improve readability and explanation; in VBA, comments start with a single quote. **EntireColumn** is a property of the range, in this case columns, and **Autofit** is a method of a range, which is specified by row(s) or column(s). **Wraptext** is also a property of the range object. A range can contain cell(s), row(s), and column(s).

I even created a macro button for this by following the steps listed in Figure 4.1, given in Excel 2002 Power Programming with VBA by John Walkenback, pp. 231 [1].

1. Choose the View ⇄ Toolbars ⇄ Customize command. Excel displays the Customize dialog box.
When the Customize dialog box is displayed, Excel is in a special "customization" mode. The menus and toolbars are not active, but they can be customized.
 2. Click the Commands tab in the Customize dialog box.
 3. Scroll down and click Macros in the Categories list.
 4. In the Commands list, drag the second item (labeled Custom Button) to the desired toolbar.
 5. Right-click the new button to display a shortcut menu.
 6. Enter a new name for the button in the text box labeled Name. This is the "tooltip" text that appears when the mouse pointer moves over the button. This step is optional; if you omit it, the tooltip displays Custom.
 7. Right-click the new button and select Assign Macro from the shortcut menu. Excel displays its Assign Macro dialog box.
 8. Select the procedure from the list of macros.
 9. Click OK to close the Assign Macro dialog box.
 10. Click Close to close the Customize dialog box.
-  After you follow the process above, the new toolbar button always appears on the assigned toolbar – even when the workbook that contains the macro is not open. In other words, changes you make using the View ⇄ Toolbars ⇄ Customize command are "permanent." Clicking the new toolbar button item opens the workbook if it's not already open.

Figure 4.1: Steps to Create Custom Toolbar Buttons for Macros [1]

4.5 Manual Calculation

If you have many formulae and the calculation is set to automatic, you are bound to face performance issues. With every change in the workbook, Excel will calculate all the formulae. One simple solution would be to set the calculation to manual (*Tools* → *Options* → *Calculation* → *Check the manual radio button*), and then perform calculations only on the selected range using the code given in Listing 4.3. In Listing 4.3, Calculate is a method for the selected range. This is particularly useful when you are developing big spreadsheet models, and want to test your models. To be really effective you should assign a shortcut key to this macro, to do assign a shortcut key follow these steps:

1. Press Alt + F8 to bring the list of macros
2. Select the macro and Hit Options

3. Assign the shortcut key

Listing 4.3: Manual Calculation Macro

```

Sub ManCalc()
    Selection.Calculate
End Sub

```

4.6 Footer

I like to have footers on every worksheet that I send out with my initials, date, and page number. I have the procedure given in Listing 4.4 to achieve that.

Listing 4.4: Procedure to Set Footer

```

Sub Footer()
With ActiveSheet
    .PageSetup.LeftFooter = "From the Office of RPIE," &
Chr(13) & "&D"
    .PageSetup.RightFooter = "ARN" & Chr(13) & "Page &P of &N"
End With
End Sub

```

With and **End With** keywords are used to work with an object, so that we do not have to explicitly write that object's full reference inside the **With** block.

Chr function is VBA's in-built function, which returns the character for a given ASCII code. In this example, I have used ASCII character 13 to insert a line.

Activsheet is a property of the active workbook, which represents the active sheet in the active workbook.

PageSetup object holds all the information on page setup attributes of a worksheet.

LeftFooter and **RightFooter** are properties of **PageSetup** object, which can either return or set text on the left side or the right side of the worksheet.

4.7 Print Formula

For this handout I have used the procedure given in Listing 4.5 to print formulae of the selection to adjacent cells as text, so that we can see values on the left hand side and the formula on the right hand side.

Listing 4.5: Print Formula in Adjacent Cells

```

1 Sub Conv2Text()
2 Dim c As Range
3 For Each c In Selection
4     c.Offset(0, 1) = Chr(39) & c.Formula
5 Next c
6 End Sub

```

In Listing 4.5, line number 2 defines a variable *c* as a range. To define a variable we use the keyword **Dim**. Line number 3 starts a **For Each** loop to iterate through all the cells from the selected range. **For Each** loop is a special type of loop, which loops through the **Collections**. The common for loop looks like this:

```

For i = 1 to 10
lines of code
....
....
Next i

```

The left hand side part on line number 4 is the location adjacent to a cell, which is found using the OFFSET function (remember Section 2.4 on page number 11). Here we are telling Excel to set a value of a cell that is one column to the right off the current cell.

The right hand side part of line number 4 is setting the value of the offset cell with the formula of the current cell and adding an apostrophe (**Chr(39)='**) to make it a text value.

4.8 User-Defined Function (UDF): Reverse String

This example will show you how to create a user-defined function (UDF). Let's say that we want to reverse a given string, so if the input is "abcd", the output would be "dcba." This function is given in Listing 4.6.

Listing 4.6: Reverse String UDF

```

Function ReverseString(sInputString As String) As String
    ReverseString = StrReverse(sInputString)
End Function

```

As mentioned in Section 4.3 on page number 20, a function is a procedure that returns a value. In this example, it is taking an input and returning the output as a string. A function's input argument are specified in the parenthesis after the function name. In this example, we have used VBA's in-built function StrReverse, which reverses the input string. Functions that are completely based, such as this one, on other functions are known as "wrappers."

User-defined functions are used exactly the same as Excel's in-built functions; however, you need to specify the path, where these functions are created. For example, if your UDFs are stored in the personal workbook, to use them in other workbooks, you would refer them with "personal.xls!" or "personal.xlsb!", depending on Excel's version, as a prefix. An example is shown in Figure 4.2.

	A	B	C
90	Reverse String Example		
91	InputString	OutputString	Formula
92	john	nhoj	=PERSONAL.XLSb!reversestring(A92)
93	word	drow	=PERSONAL.XLSb!reversestring(A93)
94	saippuakivikauppias	saippuakivikauppias	=PERSONAL.XLSb!reversestring(A94)

Figure 4.2: Example of Reverse String UDF

By the way, Saippuakivikauppias is a Finish word for "soap-stone vendor", and it is claimed to be the world's longest palindrome. That reminds me that I have a function given in Listing 4.7, which checks if a word is a palindrome or not.

Listing 4.7: Palindrome Function

```

Function IsPalindrome(sInput As String) As Boolean
If LCase(sInput) = StrReverse(LCase(sInput)) Then
    IsPalindrome = True
End If
End Function

```

4.9 Concatenate Range

I use this function almost everyday, as it lets me concatenate a range without inserting & signs or putting a comma in the CONCATENATE function. One repetitive use that I have found is to create OR/AND conditions for Access queries. I copy-paste the field values of a column from Access, do some filtering and my conditions are ready. Then I use this CONCATFUNC to create a string to use in my Access query. Let's say I want to limit the data from semesters summer 2005, fall 2005, and spring 2006 in Access. To do that I will have to write this '2005M' OR '2005F' OR '2006S' in the criteria part under the semester field in Access. If I have a long list of restrictions, then typing this manually will be very time consuming, and that's where this function, given in Listing 4.8, is useful. An example is given in Figure 4.3.

	A	B	C
97	Semester		
98	2006S	Concatenated String	2006S" or "2005M" or "2005F
99	2005M	Formula	=PERSONAL.XLSb!concatfunc(A98:A100,CHAR(34) & " or " & CHAR(34))
100	2005F		

Figure 4.3: Example of Concatenate Range Function

Listing 4.8: Concatenate Range Function

```

1 Function ConCatFunc(rngRng2bCated As Range ,
2 Optional sChar2bAdded As String = ",") As String
3 Dim sOutput As String , c As Range
4 On Error GoTo ConCatFunc_Error
5
6 For Each c In rngRng2bCated
7     sOutput = sOutput & c.Value & sChar2bAdded
8 Next c
9
10 sOutput = Left(sOutput , Len(sOutput) - Len(sChar2bAdded))
11 ConCatFunc = sOutput
12
13 On Error GoTo 0
14 Exit Function
15
16 ConCatFunc_Error :
17     ConCatFunc = "#Error#"
18

```

19 **End Function**

This function takes two inputs: the range to be concatenated and the character to be inserted in between each string. Notice on line number 1 of the Listing 4.8 the keyword **Optional**, which means the user doesn't need to supply this argument, and by default it will insert a comma between each string.

This function features some minimal error handling. Whenever VBA comes across an error, it raises a flag and generates an error message. A good programming practice is to provide "error handlers", so that the user is not left clueless. The statement **On Error GoTo ConCatFunc_Error** tells VBA to go to the label `ConCatFunc_Error` if an error is raised. A label in VBA can be specified by typing the label name and putting a colon immediately after it.

Line number 7 removes the extra character at the end of the output string by using the VBA functions **Left** and **Len**. The function **Left** takes out user-specified number of characters from the input string starting from the left side. **Len** function counts the number of characters in the input string.

4.10 Flip a Range

Often, I need to flip a range, as shown in the Figure 4.4, and for that I have a procedure, which is given in Listing 4.9.

	A	B	C	D	E
1	6	5	4	2	1

(a) Before Flipping

	A	B	C	D	E
1	1	2	4	5	6

(b) After Flipping

Figure 4.4: Example of Flip Procedure

Listing 4.9: Procedure to Flip a Range

```

1 Sub flip ()
2
3 Dim Arr As Variant
4 Dim myrange As Range
5 Dim arRetArr() As Variant, lArrBnd As Long, i As Long
6
7 On Error GoTo flip_Error
8
9 Set myrange = Range(Selection.Address)
10 Arr = myrange 'store the selected values in an array
11
12 ' check if there is only one column or not
13 If myrange.Columns.Count = 1 Then
14     lArrBnd = UBound(Arr, 1)
15     ReDim arRetArr(lArrBnd, 0)
16     For i = 0 To lArrBnd - 1
17         'flip the array
18         arRetArr(i, 0) = Arr(lArrBnd - i, 1)
19     Next i
20     Range(Selection.Address) = arRetArr
21 'check if there is only one row or not
22 ElseIf myrange.Rows.Count = 1 Then
23     lArrBnd = UBound(Arr, 2)
24     ReDim arRetArr(0, lArrBnd)
25     For i = 0 To lArrBnd - 1
26         'flip the array
27         arRetArr(0, i) = Arr(1, lArrBnd - i)
28     Next i
29     Range(Selection.Address) = arRetArr
30 Else
31     MsgBox "Your selection contains multiple rows or columns."
32 & vbCrLf & -
33     "This macro will only work on either one column or one row",
34     vbCritical, "Flip Error"
35 End If
36
37 On Error GoTo 0
38 SmoothExit_flip:
39     Exit Sub
40
41 flip_Error:
42     MsgBox "Error " & Err.Number & " (" & Err.Description & ")
43 in procedure flip"
44     Resume SmoothExit_flip
45 End Sub

```

4.11 Fill Blank Cells

This procedure is very helpful when we have blank cells in a range and we would like to fill the blank cells with the values from the non-blank cells above it. I use it when I copy and paste data from Pivot Tables. The procedure listed in Listing 4.10 is from <http://www.ozgrid.com>, another good place for Excel and VBA tips and tricks. An example of this procedure is shown in Figure 4.5.

	A		A
102	FillBlanks	102	FillBlanks
103	A	103	A
104		104	A
105		105	A
106	B	106	B
107		107	B
108		108	B
109	C	109	C

(a) Before Filling the Blanks (b) After Filling the Blanks

Figure 4.5: Example of Fill Blanks Procedure

Listing 4.10: Procedure to Fill the Blanks

```
1 'taken from http://www.ozgrid.com/Excel/excel-fill-blank-cells.htm
2 Sub FillBlanks ()
3 Dim rRange1 As Range, rRange2 As Range
4 Dim iReply As Integer
5     If Selection.Cells.Count = 1 Then
6         MsgBox "You must select your list and include the blank
7 cells", - vbInformation, "OzGrid.com"
8         Exit Sub
9     ElseIf Selection.Columns.Count > 1 Then
10        MsgBox "You must select only one column", -
11        vbInformation, "OzGrid.com"
12        Exit Sub
13    End If
14
15    Set rRange1 = Range(Selection.Cells(1, 1), -
16        Cells(65536, Selection.Column).End(xlUp))
17
18    On Error Resume Next
19    Set rRange2 = rRange1.SpecialCells(xlCellTypeBlanks)
20    On Error GoTo 0
21
22    If rRange2 Is Nothing Then
23        MsgBox "No blank cells Found", -
24        vbInformation, "OzGrid.com"
25        Exit Sub
26    End If
27
28    rRange2.FormulaR1C1 = "=R[-1]C"
29
30    iReply = MsgBox("Convert to Values", vbYesNo + vbQuestion,
31 "OzGrid.com")
32    If iReply = vbYes Then rRange1 = rRange1.Value
33 End Sub
```

Chapter 5

Resources

1. My blog: <http://www.nandeshwar.info/projects/xlblog/>
2. ASAP Utilities (A must have!): <http://www.asap-utilities.com>
3. Discussion board at Mr. Excel.com: <http://www.mrexcel.com/board2/index.php>
4. Ozgrid.com: <http://www.ozgrid.com/Excel>
5. Chip Pearson's Excel page: <http://www.cpearson.com/excel/mainpage.aspx>
6. Daily dose of Excel: <http://www.dailydoseofexcel.com/>
7. John Walkenbach's spreadsheet page: <http://j-walk.com/ss/>
8. Tushar Mehta's Excel page: <http://www.tushar-mehta.com/excel/>
9. Books list: <http://www.nandeshwar.info/projects/xlblog/labels/Books.html>

Bibliography

- [1] J. Walkenbach. *Excel 2002 power programming with VBA*. M&T Books, 2001.
- [2] J. Walkenbach. *Excel 2007 Power Programming with VBA*. Wiley; John Wiley distributor, 2007.

Appendix A

Keyboard Shortcuts from cpearson.com

Excel Keyboard Shortcuts from: <http://www.cpearson.com/excel/KeyboardShortcuts.htm>

Key	Alone	Shift	Ctrl	Alt	Shift Ctrl
F1	Help	What's This Help		Insert Chart Sheet	
F2	Edit Mode	Edit Comment		Save As	
F3	Paste Name Formula	Paste Function	Define Name		Names From Labels
F4	Repeat Action	Find Again	Close Window	Quit Excel	
F5	Goto	Find	Restore Window Size		
F6	Next Pane	Prev Pane	Next Workbook	Switch To VBA	Prev Workbook
F7	Spell Check		Move Window		
F8	Extend Selection	Add To Selection	Resize Window	Macro List	
F9	Calculate All	Calculate Worksheet	Minimize Workbook		
F10	Activate Menu	Context Menu	Restore Workbook		
F11	New Chart	New Worksheet	New Macro Sheet	VB Editor	
F12	Save As	Save	Open		Print
A			Select All		Formula Arguments
B			Bold		
C			Copy		
D			Fill Down	Data Menu	
E				Edit Menu	
F			Find	File Menu	Font Name
G			Goto		
H			Replace	Help Menu	
I			Italics	Insert Menu	
J					
K			Insert Hyperlink		
L					
M					
N			New Workbook		
O			Open Workbook	Format Menu	Select Comments
P			Print		Font Size
Q					
R			Fill Right		
S			Save		
T				Tools Menu	
U			Underline		
V			Paste		
W			Close Workbook	Window Menu	

Excel Keyboard Shortcuts from: <http://www.cpearson.com/excel/KeyboardShortcuts.htm>

Key	Alone	Shift	Ctrl	Alt	Shift Ctrl
X			Cut		
Y			Repeat Active		
Z			Undo		
` (~)			Toggle Formula View		General Format
1 (!)			Cell Format		Number Format
2 (@)			Toggle Bold		Time Format
3 (#)			Toggle Italics		Date Format
4 (\$)			Toggle Underline		Currency Format
5 (%)			Toggle Strikethru		Percent Format
6 (^)			a		Exponent Format
7 (&)			a		Apply Border
8 (*)			Outline		Select Region
9 (())			Hide Rows		Unhide Rows
0 ())			Hide Columns		Unhide Columns
-			Delete Selection	Control Menu	No Border
= (+)	Formula			Auto Sum	Insert dialog
[Direct Dependents		Direct Precedents
]			All Dependents		All Precedents
; (semicolon)			Insert Date	Select Visible Cells	Insert Time
' (apostrophe)				Style	Copy Cell Value Above
: (colon)			Insert Time		
/			Select Array		Select Array
\			Select Differences		Select Unequal Cells
Insert	Insert Mode		Copy		
Delete	Clear		Delete To End Of Line		
Home	Begin Row		Start Of Worksheet		
End	End Row		End Of Worksheet		
Page Up	Page Up		Previous Worksheet	Left 1 screen	
Page Down	Page Down		Next Worksheet	Right 1 screen	
Left Arrow	Move Left	Select Left	Move Left Area		
Right Arrow	Move Right	Select Right	Move Right Area		
Up Arrow	Move Up	Select Up	Move Up Area		
Down Arrow	Move Down	Select Down	Move Down Area	Drop down list	
Space Bar	Space	Select Row	Select Column	Control Box	Select All
Tab	Move Right	Move Left	Next Window	Next Application	Previous Window
BackSpace			Goto Active Cell		