Chapter 3:
The IF Function and Table Lookup

Objectives
This chapter focuses on the use of IF and LOOKUP functions, while continuing to introduce other functions as well. Here is a partial list of what the chapter covers:

- Further practice of good design techniques
- More practice using logical functions
- Creating and using lookup tables and functions

Preparation
To be able to complete this chapter in about 3 hours it is essential that you are properly prepared. You should read the whole of this chapter very carefully.

Introduction
This chapter continues the examination of logical functions and expands on techniques that allow a choice to be made between which values to use in a calculation. One of the common ones is the LOOKUP function.

Exercise 1 - Sales Person Bonus Model

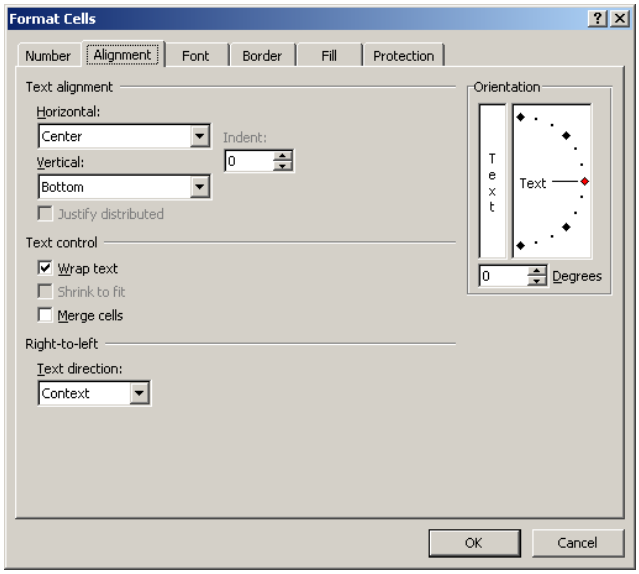
Open the file Exercise 1 in Support Files (Chapter 3) on the course website. This model has two worksheets – the Comments worksheet and the Sales_Record worksheet. Your aim is to add a column to the Sales_Record worksheet, which identifies whether the particular sales person has made sales less than the average of all sales or equal to or greater than the average. The cells in the new column might contain Equal/Above or Below, for example.

To do this you must first calculate the average of the sales. Enter a label such as Sales Average in a cell at the bottom of the column of names and in the cell next to it compose the formula that will calculate the average of the sales. You'll need to use the AVERAGE function, which you have seen in a previous chapter. If you name the Sales column your formula will simply look like this:

```
=AVERAGE(Sales)
```

Next enter a heading for the new column – perhaps Compared to Average. Make sure the heading is appropriately formatted for the cell width and also by font size etc. You'll probably want to choose the Home tab and Wrap Text inside the Alignment group, and to increase the cell height for the row that contains the column headings. Or you can click the Format command, found in the Cells group of the Home tab, and then select Format Cells… from the drop down menu (see Figure 3.1).

Figure 3.1 – the Alignment tab in the Format Cells window



The IF Function

Now you need to enter the formula that will decide whether the words Below or Equal/Above will appear in the first cell. Select that first cell (C2) and click on the Function button in the tool bar:

The Function button: 

The IF function can be found in the Logical group of functions (see Figure 3.2), and after pressing the OK button the argument specification panel shown in Figure 3.3 appears.

Figure 3.2 – inserting the IF function

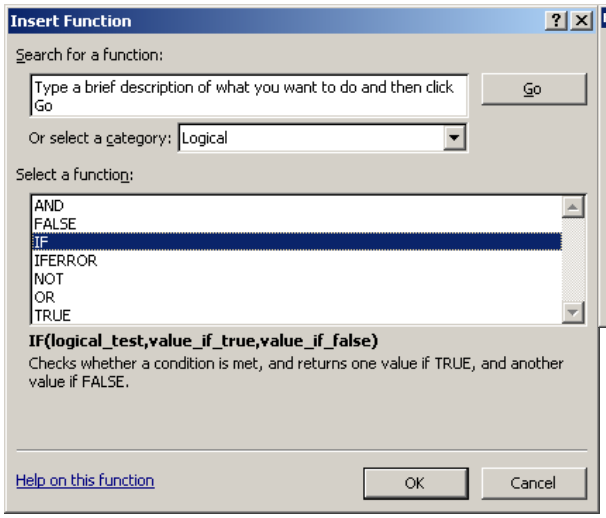
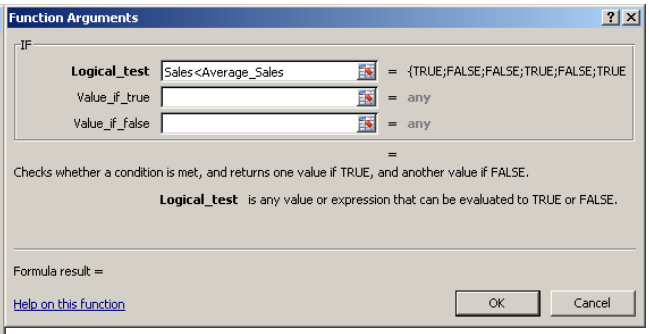


Figure 3.3 –argument panel



The Logical_test argument needs to compare the sales figure (for this sales person) to the average sales. If you didn't name the average sales value that you calculated recently then cancel this operation and go back and define a name for it. The logical test that you enter here (using the names you have defined) will be:

Sales < Average_Sales

You can use the Use in Formulas command in the Defined Names group within the Formulas tab.

The value of this Logical_test expression should not be thought of as a number (although it might be represented as a number by the computer), nor should it be thought of as a word. Instead you should think of its value as the abstract true or false – either it is true that sales is less than the average, or it is false (sales is not less than the average). Thinking in this way is key to constructing the argument correctly.

The Value_if_true argument should simply be the text "Below" – so type this (with double quotes, "….") into the text box now, and then the string "Equal/Above" into the Value_if_false text box.

Press the OK button and you should find that the formula has chosen the string Below as the value for this first cell, because the sales amount for salesperson Bushby is in fact less than the average sales. Fill the formula down the column to see the complete results. The first few rows of the Sales_Record worksheet should look like Figure 3.4.

Figure 3.4 – part of the Sales_Record worksheet

	A	B	C
1	**Salesperson**	**Sales**	**Compared to Average**
2	BUSHBY	$200,448	Below
3	CAMPBELL	$259,500	Equal/Above
4	CORREIA	$341,628	Equal/Above
5	DHILLON	$232,521	Below
6	FERNANDES	$302,356	Equal/Above
7	FRASER	$187,671	Below
8	GRAVAS	$209,988	Below
9	HALL	$163,171	Below

Adding a Bonus

Let's add one further step to this. Suppose that we want to calculate a bonus of 10% of sales for those who achieved sales of at least one standard deviation better than the average. A standard deviation describes how values in a sample are distributed about the average. Typically 68% of values fall within one standard deviation above or below the average. So if a salesperson sells more than "the average plus one standard deviation"

they have done very well compared to others in the group. The standard deviation can be calculated using a function just like the average was.

The steps required here include labeling a cell for the standard deviation, creating the formula to calculate the value, labeling a new column for the bonus, and creating the formula to calculate those values. Note that if the sales for a particular person is less than the average plus one standard deviation the bonus cell would best be left blank.

In the next row under the Sales Average enter a label such as Standard Deviation: and in the adjacent cell create the formula to calculate the value. The function is called STDEV.P and can be found in the Statistical group of functions.

Next enter a column heading – Bonus – and format it appropriately.

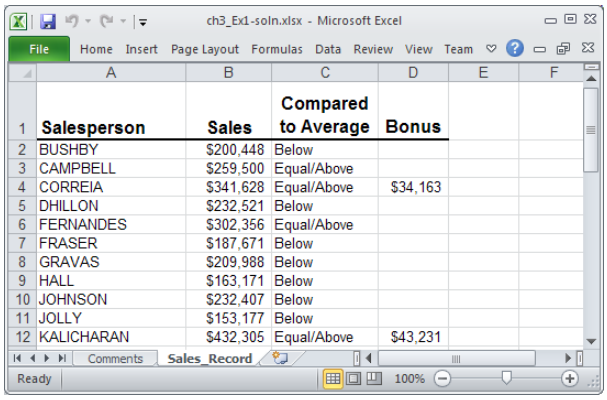
The formula for the Bonus column will again use an IF function. This time however you need to implement the arguments as follows:

Logical_test:	sales > sales average + one standard deviation
Value_if_true:	10% of sales
Value_if_false:	empty

These arguments are written in normal English – they do not use the cell names and symbols that you need to use in implementing the formula. You cannot write 10% of sales in the text box for Value_if_true for example – you must translate it first into the correct symbols and names that the spreadsheet recognises. The easiest way to implement the Value_if_false argument is to type two double quotes (""). This makes the cell appear to be empty, although it's not quite the same as actually being empty. You should be able to figure out how to implement the other arguments yourself.

The first few rows of the worksheet should look like Figure 3.5.

Figure 3.5 – part of the Sales_Record worksheet with Bonus included



	A	B	C	D	E	F
1	Salesperson	Sales	Compared to Average	Bonus		
2	BUSHBY	$200,448	Below			
3	CAMPBELL	$259,500	Equal/Above			
4	CORREIA	$341,628	Equal/Above	$34,163		
5	DHILLON	$232,521	Below			
6	FERNANDES	$302,356	Equal/Above			
7	FRASER	$187,671	Below			
8	GRAVAS	$209,988	Below			
9	HALL	$163,171	Below			
10	JOHNSON	$232,407	Below			
11	JOLLY	$153,177	Below			
12	KALICHARAN	$432,305	Equal/Above	$43,231		

Exercise 2 - Kasch Pulse Recovery Study

Open the file Exercise 2 in Support Files (Chapter 3) on the course website. You'll see a model with two worksheets – a Comments worksheet and a Fitness_Data worksheet. It contains a list of names, along with associated gender, age, and pulse rate measurement. The pulse rate is measured after 5 seconds of rest following 3 minutes of exercise and it is an indication of the fitness level of the individual. Your aim here is to add a new column identifying the fitness level of the subjects in the study. Part of the Fitness_Data worksheet with this column added is shown in Figure 3.6.

Figure 3.6 – part of the Fitness_Data worksheet

	A	B	C	D	E
1	**Name**	**Gender**	**Age**	**Pulse Rate**	**Fitness Rating**
2	BUSHBY	M	26	75	good
3	CAMPBELL	M	52	97	poor
4	CORREIA	M	33	86	good
5	DHILLON	F	29	104	poor
6	FERNANDES	M	56	75	good
7	FRASER	M	35	87	good
8	GRAVAS	F	22	111	poor

Ex 2.1 - Two Fitness Ratings

One simple approach is to ignore age and gender and to state that if the pulse rate is under 95 the fitness level is good, otherwise it is poor.

You should be able to implement this yourself. Enter a column title and an IF function formula that calculates either Good or Poor as the values for the cells in the column.

Ex 2.2 - Many Different Ratings

Actually it is better to identify levels of fitness – say Excellent, Good, Average, Fair, and Poor. Ignoring age and gender you could define these levels according to the following table:

Pulse Rate	**Fitness Rating**
Less than 80	Excellent
80 to <90	Good
90 to <105	Average
105 to <115	Fair
115 and greater	Poor

To implement this using an IF function requires careful analysis. The IF function template require three arguments – a logical test, the value to return if the test is true, and the value to return if the test is false. In this example there is not just one logical test – there are 5.

We could begin by saying IF(Pulse_Rate < 80, "Excellent", ????). This is certainly correct so far, since if the pulse rate is indeed less than 80 we do want the word Excellent to appear in the cell. The question remains as to what to use for the Value_if_false argument. In other words, what is the answer if the pulse rate is greater than or equal to 80?

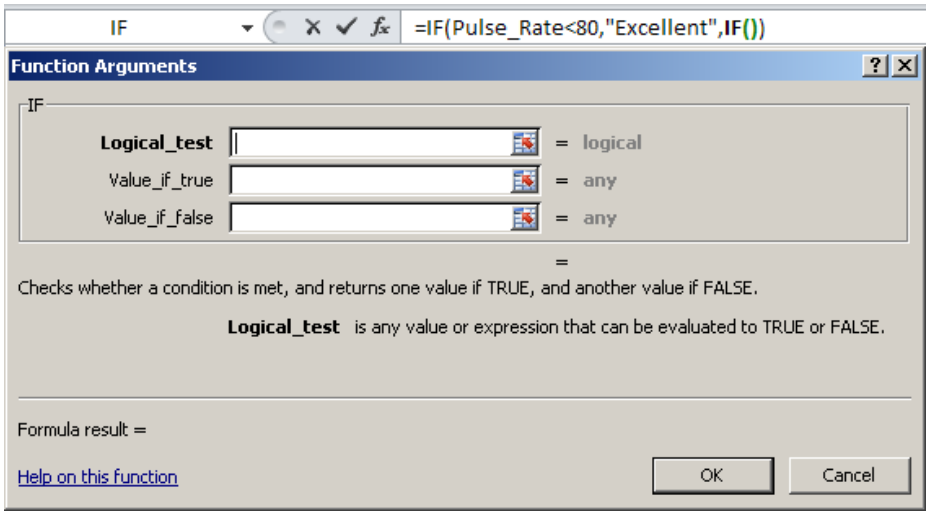
Well, we don't know the answer without further testing – so we must replace the Value_if_false argument with another IF function:

IF(Pulse_Rate < 80, "Excellent", IF(Pulse_Rate < 90, "Good", ????))

Now we have the answer if the pulse rate is less than 80, and if it's greater than or equal to 80 we also have the answer if it's less than 90. But if the pulse rate is greater than 90 we still don't have the answer. Clearly, we need to continue this process of building what is called a "nested" IF function.

To implement this formula, first name the Pulse Rate column and then use the function wizard to start creating the nested IF formula. You should be easily able to enter the correct specification for the first Logical_test and Value_if_true text boxes. For the Value_if_false text box you want to enter another IF function. The easiest way to do this is to click on the name box (which is in the top left just above the argument panel and should be displaying IF). You should get a new IF function argument panel, as shown in Figure 3.7.

Figure 3.7 – the first nested IF function



Observe that a new IF function has been entered in the formula under construction in the formula bar. This new IF function has no arguments at this time, but as you specify the arguments in this panel you'll see them inserted into the formula in the formula bar.

You need to enter Pulse_Rate < 90 as the Logical_test and "Good" as the Value_if_true. For the Value_if_false you need to enter another IF function. Your formula should now look like this

```
=IF(Pulse_Rate<80,"Excellent",IF(Pulse_Rate<90,"Good",IF()))
```

and another empty argument panel should be displayed.

Notice that the logical test in the second IF function does ***NOT*** say:

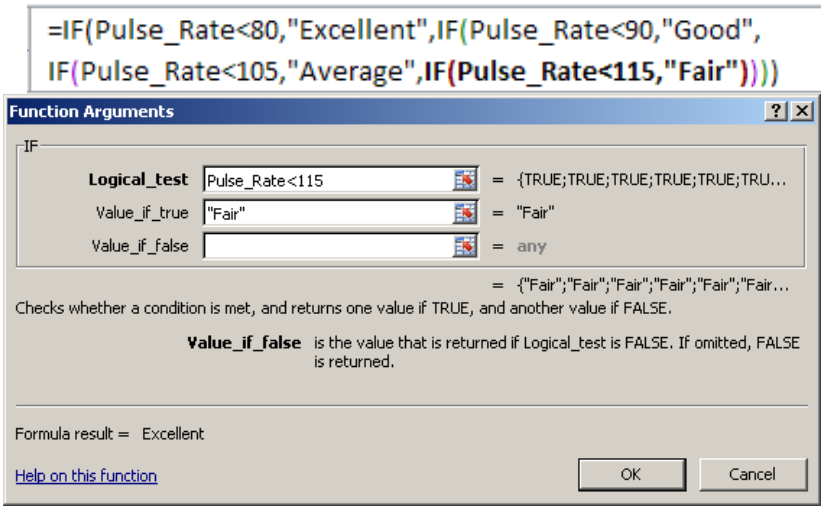
80 <= Pulse_Rate < 90

Beside the fact that this is illegal in Excel, it is also unnecessary because the preceding test accounts for the first part, i.e. 80 <= Pulse_Rate. If the logical test in the first IF function is true the answer is given as Excellent, and none of the other IF functions are carried out. Thus, if the second IF function is examined it must be because the first one was found to be false! This means that Pulse_Rate must definitely be greater than (or equal to) 80 and hence that first part of the logical test (80 <= Pulse_Rate) is redundant because it is already known to be true.

The method of writing nested IF functions that we are using here only works if you write the logical tests in order – increasing order of pulse rate in this case.

Continuing with the construction of nested IF functions, you'll end up with the formula in Figure 3.8, in which the last Value_if_false argument is yet to be specified.

Figure 3.8 – the final nested IF function argument panel



You could nest another IF function with the logical test Pulse_Rate >= 115 and the Value_if_true set to "Poor". In this case the Value_if_false could be left unspecified or set to "".

It is better to realise that having dealt with the cases of pulse rate <80, <90, <105 and <115 all values that are left **must be greater than 115** and hence rate "Poor". The Value_if_false argument for this last IF function can simply be specified as "Poor" rather than another function.

Complete the formula and fill it down the column to see that it works correctly.

Ex 2.3 - Dependence on Gender

Age dependence is left as an exercise at the end of this chapter, so let's first see how the gender dependence can be included in the Fitness Rating. The following table describes the fitness ratings for males and females:

For males:

Pulse Rate	**Fitness Rating**
Less than 80	Excellent
80 to <90	Good
90 to <105	Average
105 to <115	Fair
115 and greater	Poor

For females:

Pulse Rate	**Fitness Rating**
Less than 87	Excellent
87 to <100	Good
100 to <111	Average
111 to <123	Fair
123 and greater	Poor

To implement a calculation such as this implies we first need to say something like:

If (Gender is male, true value is computed as for a male,
false value is computed as for a female)

This means that the Value_if_true argument will be a series of nested IF functions which use the table for males and the Value_if_false argument will be a series of nested IF functions which use the table for females.

To do this you can modify the formula that you wrote in the previous part. The first thing to do however is to define a name for the gender column that you will use in the formula – so do that now.

To edit the previous formula select the top cell and click just after the = symbol in the formula bar. This is where you are going to start typing the new parts of the formula. A vertical line should be blinking between the = symbol and the I of IF, indicating where new characters that you type will appear.

Assuming Gender is the name you defined for the Gender column type the following:

IF(Gender="M",

The existing series of nested IF functions constitute the Value_if_true argument of this new IF function you are inserting.

To create the Value_if_false argument you can copy and paste the Value_if_true argument and then change the numbers in the Logical_test arguments to match the values in the table for females.

Carefully select from IF(Pulse_Rate<80 right to the end of the existing formula and choose Copy from the Clipboard group inside the Home tab.

Next click right at the end of the existing formula (i.e. after all of the parentheses) and type a comma. If you type the comma while the selection you just copied is still highlighted it will all be replaced by the comma, so take care with this.

Now choose Paste from the Clipboard group inside the Home tab, and the series of nested IF functions should appear again in the formula. Now change 80 to 87, 90 to 100, 105 to 111 and 115 to 123 as implied by the table above.

Finally type a right parenthesis) at the very end to close the argument list for the new IF function you have inserted. The formula should look something like this, though this one has been formatted to make it easier to read:

```
IF(Gender = "M" ,
      IF(Pulse_Rate < 80, "Excellent", IF(Pulse_Rate < 90, "Good", IF(Pulse_Rate <
      105, "Average", IF(Pulse_Rate < 115, "Fair", "Poor" ) ) ) )  ,
      IF(Pulse_Rate < 87, "Excellent", IF(Pulse_Rate < 100, "Good", IF(Pulse_Rate <
      111, "Average", IF(Pulse_Rate < 123, "Fair", "Poor" ) ) ) )  )
```

Examine this formula very carefully to make sure that you understand all of its components. You should develop flexibility in how you build formulas – using the point and click method at times and at others just typing, or cutting and pasting.

Commentary

As you can see formulas can get quite complicated if you attempt to combine many of them. This is not a good practice and is done here mainly to demonstrate that it is confusing. There are better ways to tackle this kind of problem, as you'll soon see.

Exercise 3 **- Sales Discount Model**

This section extends our study of IF functions. In particular it introduces the idea of a compound logical test involving the use of the AND and OR functions.

Open the file Exercise 3 in Support Files (Chapter 3) on the course website. You'll see a Comments worksheet (read it carefully) and a Discounts worksheet, which contains the codes for various products sold in a store, the status of the product (C means current and D means discontinued), the quantity of the product in stock, and the average daily sales for the product.

Ex 3.1

As the store manager you want to hold a sale in order to attract customers and sell off old stock that is either discontinued or not selling very well. A product is not selling very well if there is so much in stock that at the average daily sales the stock would last for 15 days or more. If a product is discontinued or not selling well you decide to discount its price by 25%. You'll discount everything else by 10%.

So the task now is to implement the criteria just described so that you have a new column showing either 10% or 25% as the discount percentage for each stock item.

Label a column with the heading Discount Percent and then define names that you'll use in the formula for the other columns of data in the worksheet.

The key to the formula is indicated by the sentence:

> *If a product is discontinued or not selling well its price will be discounted by 25%, otherwise it will be discounted by 10%.*

This clearly implies an IF function, but what should the logical test be? The sentence indicates that the test is (*product is discontinued or not selling well*) which involves two logical parts. "*Is the product discontinued?"* is one part and "*Is the product not selling well?"* is the other. If the answer to one OR the other (or both) of these parts is true, then the discount should be 25%.

Joining two logical tests by an OR operator is achieved in Excel using the OR function.

Select the first cell in the new column in order to begin creating the formula, and then click on the Insert Function icon and select IF from the Insert Function window.

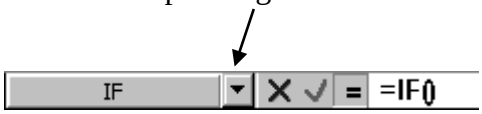
Three arguments must now be filled in: Logical_test, Value_if_true and Value_if_false. These can be done in any order, but most likely you will do them in sequence.

First fill in the logical test argument -- which will be the implementation of:

OR (*product is discontinued, product is not selling well*)

The statements *product is discontinued* and *product is not selling well* are the arguments of the OR function. You will have to convert these into the syntax of Excel.

First you need the OR function, so click on the downwards pointing arrow next to the name box on the far left of the formula bar:



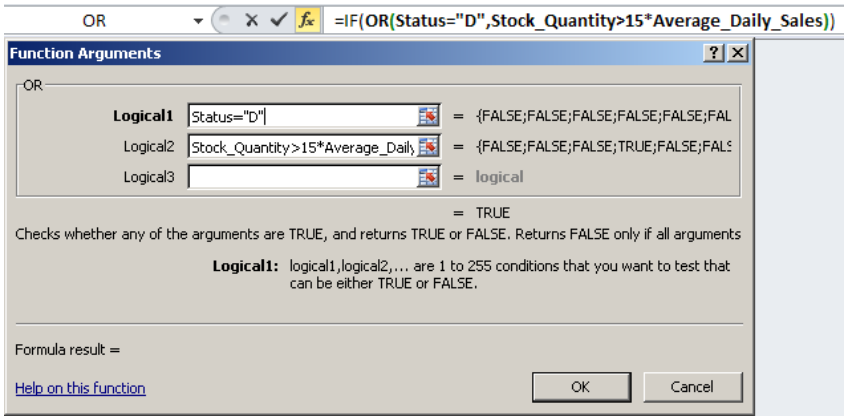
You should see a short list of functions with the last selection being More functions… . If OR is in the list, select it. Otherwise, select More functions… and find OR.

You should now be faced with the OR function argument panel which will have text boxes labeled Logical1 and Logical2. Logical1 should be the implementation of *product is discontinued* which is simply Status="D". Logical2 is the implementation of *product is not selling well*, which is calculated as

Stock_Quantity > 15*Average_Daily_Sales

The OR function argument panel should end up looking like Figure 3.9 (yours will be different if you defined different names for the various cell ranges). Now ***DO NOT*** Click the OK button. Had you clicked the OK button in the OR function argument panel Excel would have assumed the entire formula creation process was finished and produced an incorrect formula. You would not have been able to return to the Value_if_true and Value_if_false arguments later to complete the formula for IF. Instead of clicking the OK button in the OR function argument panel, move the cursor to the formula bar and position it between the I and F of IF and click the mouse. The function argument panel for the IF function will return and you can continue building the IF function.

Figure 3.9 – the OR function argument panel (note the formula in the formula bar)



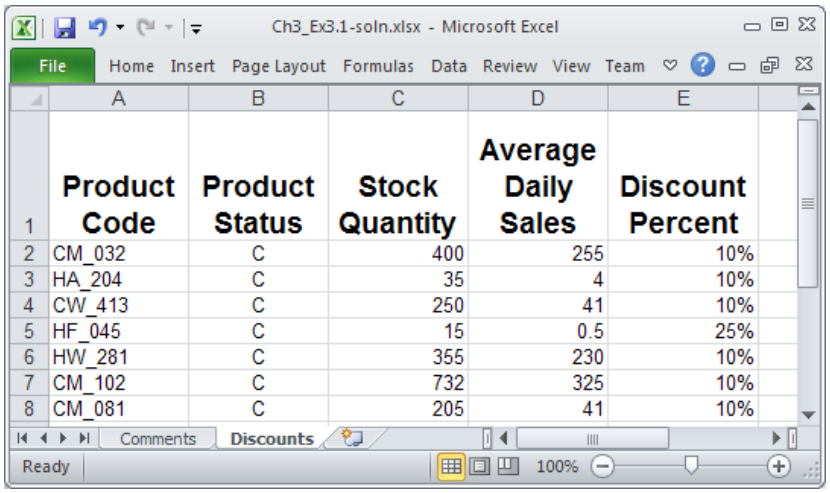
Commentary

This is what should have happened when OK is clicked in the OR function. The OR function is nested inside the IF function and the function argument panels should be

managed in a stack-like fashion. Clicking OK in OR should finish that panel and pop the stack. As a result the function argument panel for IF is again at the top of the stack ready for work to resume. That this doesn't happen is, in the author's opinion, an implementation error in Excel but the above described technique is an effective "work around" as long as you remember ***NOT TO CLICK*** on OK in a Function Argument panel, except for the function that appears right after the = in the formula bar.

Now fill in the Value_if_true argument – it will simply be 25%, being the discount percent if the item is discontinued or not selling well. The Value_if_false argument can also be filled in as 10%. (It is possible to type these values with the % symbol in the text box or formula. Excel turns the percentages into the equivalent decimal fractions 0.25 and 0.1.)

Figure 3.10 – part of the Discounts worksheet



	A	B	C	D	E
1	Product Code	Product Status	Stock Quantity	Average Daily Sales	Discount Percent
2	CM_032	C	400	255	10%
3	HA_204	C	35	4	10%
4	CW_413	C	250	41	10%
5	HF_045	C	15	0.5	25%
6	HW_281	C	355	230	10%
7	CM_102	C	732	325	10%
8	CM_081	C	205	41	10%

Ex 3.2 - An Alternative Sales Discount Policy

It is possible that instead of holding a store-wide sale (discounting every item) you might want to hold a sale only on women's clothing that is not selling well. You will discount such items by 25% in order to get rid of them. To rephrase this you can say:

IF (*it's a women's clothing item* and *it's not selling well*,
discount by 25% , *otherwise no discount*)

Notice that it is often very helpful to carefully consider the English (natural language) statement of a problem in order to clarify what has to be done.

This task is now not so hard. You can see that, by analogy with the previous example, you'll need to use the AND function with two arguments that implement the statements

it's a women's clothing item and *it's not selling well*. You already know how to do the *it's not selling well* part so the let's consider the *it's a women's clothing item* part.

It is often the case that product or item codes contain information about the objects they represent. In fact most identification codes do. For example, your driver's license number contains your gender and the month, day, and year of your birth in the last six digits!

In this worksheet the product codes begin with CW, CM, HA, HF, etc. CW means *clothing – women's*, CM means *clothing – men's*, HA means *hard goods – appliances* etc. So to determine if a product code represents an item of women's clothing you simply need to test:

are the first two characters of product code = CW

Fortunately most spreadsheet programs include functions that allow you to manipulate text. Common operations are:

- Extract the first few characters from a string
- Extract the last few characters from a string
- Find the length of a string
- Extract some number of characters starting from some position in the string

Although we only need to use one of the text functions in this example you should examine the other text functions as well.

In this case you want to extract the first two characters from the product code and test to see if they are equal to CW. Try to construct the IF function formula yourself. The Logical_test argument will consist of the AND function and as the Logical1 argument of the AND function you will need to use the LEFT function (from the Text group of functions) in order to specify the first 2 characters of the product code.

Be careful in how you build your formula. There are three nested functions: LEFT is inside AND which, in turn, is inside IF. After you complete LEFT, you will want to return to AND, and when you complete AND you will want to return to IF. In each case, remember ***NOT*** to click OK in the function argument panel but instead place the cursor on the function name in the formula bar and click the mouse. The function argument panel for that function returns and you can continue to work on it.

The completed formula should read

```
=IF(AND(LEFT(Product_Code,2)="CW",Stock_Quantity>15*Average_Daily_Sales),25%,"")
```

and the new column should have only two non-blank entries after you have filled the formula down the column.

Exercise 4 - Multiple Selection Logic

Introduction

You will typically use an IF function when there are two things to choose from based on some condition being true or false. If there are many things to choose between – such as choosing if a fitness rating is excellent, good, average, fair or poor – an IF function can become very cumbersome. You saw this in an earlier exercise.

Since these cases are common, the LOOKUP function facilitates multiple selection. The basic idea is that you construct a table containing the values that can be returned (this is called the Result_vector), along with a list of the quantities that serve as the criterion determining which one you choose (this is called the Lookup_vector). For example, the fitness ratings are the value from which you want to choose, and the pulse rate value determines which one you choose. You must construct the lookup table in order of increasing values of the variable used to determine the choice (i.e. the pulse rate in the earlier example).

Ex 4.1 - Introductory Lookup Example

Open the file Exercise 4 in Support Files (Chapter 3) on the course website. The model contains a Comments worksheet, a Data worksheet, and a Lookup worksheet.

The Data worksheet simply contains a list of integer values in a column labeled Values and an empty column labeled Rating. The aim of this exercise is to calculate what the value in the Rating column should be for each number in the Values column. We want the following ratings:

Value	**Rating**
0 to < 5	very high
5 to <8	high
8 to <11	middle
11 to <15	low
15 and greater	very low

You could use an IF function but it would be quite long and cumbersome:

```
= IF(Values<5,"very high", IF(Values<8, "high", IF(Values<11, "middle",
      IF(Values<15, "low", "very low"))))
```

Instead, observe that the Lookup worksheet (shown in Figure 3.11) contains the table of choices for Rating along with the **lower bounds** of Value for which the rating applies. The idea is that each value in the Values column of the Data worksheet will be looked up

in the Boundary Value column of the Lookup worksheet and the corresponding Rating returned to the Rating column of the Data worksheet.

Figure 3.11 – the Lookup worksheet

	A	B
1	**Boundary Value**	**Rating**
2	0	very high
3	5	high
4	8	middle
5	11	low
6	15	very low

It is important for you to understand why this table is constructed in the way it is – why does it start with 0 in the Boundary Value column? Why doesn't it specify something like "0 to 4" as the first cell rather than just 0?

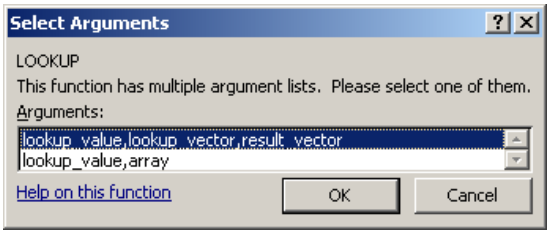
You need to understand that the syntax of Excel ***requires that you specify only the lower bounds*** of the rating ranges. Something like "0 to 4" attempts to specify both the lower and upper bound.

Creating the Lookup formula

Select the first cell in the Rating column of the Data worksheet (not the Lookup worksheet) and click on the function button in order to select the LOOKUP function from the Lookup & Reference group of functions.

The panel shown in Figure 3.12 will appear asking you to select which type of lookup function to use – one with three arguments or one with two arguments – make sure the three argument version is selected and click the OK button..

Figure 3.12 – selecting arguments for the Lookup function



The by now familiar function argument panel will appear – this time for the LOOKUP function which requires three arguments: Lookup_value, Lookup_vector, and Result_vector.

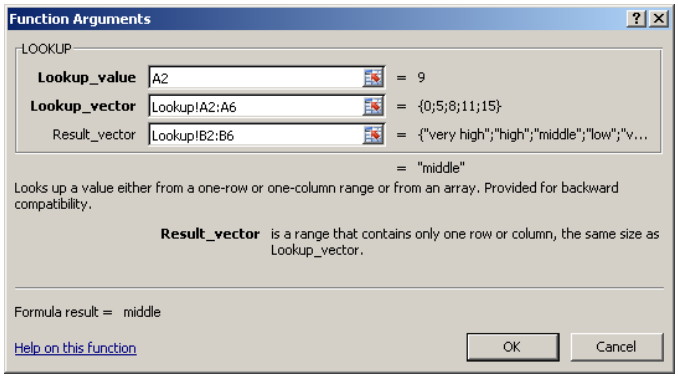
The Lookup_value is the single value from the Values column in the Data worksheet for which you want to find a rating. When you fill the formula down the column it will give results for all of the numbers in the Values column. Names have not been defined for the data in these worksheets so you should just click on the first cell in the Values column in order to specify this argument. You may need to move windows around in order to uncover this first cell.

The Lookup_vector is the place where you are going to try to find the lookup value in order to associate a rating with it. In this case the Lookup_vector is the Boundary Value column in the Lookup worksheet – so drag over that column in order to enter the argument.

The Result_vector is the place where the values you want to select from are found – in this case the Rating column in the Lookup worksheet – so drag over that column in order to enter this last argument.

You should end up with the arguments shown in Figure 3.13.

Figure 3.13 – the Lookup function arguments



There is something very wrong with this formula! See if you can discover what it is before you proceed any further.

Correcting the Lookup Formula

Click the OK button (the answer displayed for this first cell should be correct – but please check it) and then fill the formula down the Rating column in the Data worksheet. Practically all the results from the formula will show #N/A! What has gone wrong?

Look carefully at the formulas that have been filled down the column and you'll see that the Lookup_vector and the Result_vector arguments have changed as the formula was filled down the column. This is that relative versus absolute cell reference problem that

we encountered first in Chapter 2 (pages 4 and 5) – it has been deliberately resurrected as a reminder to you at this time.

Select the first cell of the Rating column and type the $ symbols in the appropriate places to make the Lookup_vector and Result_vector absolute cell references and then fill the formula down the column again. This time the correct results should appear. This problem would not have occurred had we defined names for the data in these worksheets.

Exercise 5 - Kasch Pulse Recovery Study Again

You worked with this model earlier in the chapter, but it would really benefit from using a lookup table and function rather than an IF function.

Open the Exercise 2 model again found in the Support Files. Do not use the file you worked with previously – start afresh. There is a Comments worksheet and the Fitness_Data worksheet with the columns Name, Gender, Age, and Pulse Rate.

Ex 5.1 - Fitness Rating without Gender or Age

To begin with we'll once again ignore age and gender and implement the fitness rating choices as determined by the pulse rates using the table:

Pulse Rate	**Fitness Rating**
Less than 80	Excellent
80 to <90	Good
90 to <105	Average
105 to <115	Fair
115 and greater	Poor

Add a new worksheet to the model (using the Insert worksheet command) and name it RatingLookup. You need to construct a lookup table derived from the table shown above. You should be able to do this by analogy to the previous exercise.

Next, title a new column in the Fitness_Data worksheet and create the lookup formula for the first cell of the column. The Lookup_value should be the Pulse Rate, the Lookup_vector should be a column from your new RatingLookup worksheet containing the pulse rate boundary values, and the Result_vector should be a column from the RatingLookup worksheet containing the fitness ratings.

Be sure to define names for the various data in these worksheets!

Ex 5.2 - Gender Dependence

As you saw previously, the fitness rating depends on gender as described in the following table. You implemented this previously using a fairly complex series of IF functions but here you'll see how it can be done using LOOKUP functions.

The basic idea is to say:

If (subject is male, lookup the table for males,
lookup the table for females)

For males:

Pulse Rate	**Fitness Rating**
Less than 80	Excellent
80 to <90	Good
90 to <105	Average
105 to <115	Fair
115 and greater	Poor

For females:

Pulse Rate	**Fitness Rating**
Less than 87	Excellent
87 to <100	Good
100 to <111	Average
111 to <123	Fair
123 and greater	Poor

You can consider the Value_if_true argument of the IF function to be a nested LOOKUP function using the fitness rating table for males, and the Value_if_false argument to also be a nested LOOKUP function but this time using the table for females.

The two lookup tables can be part of the same worksheet – you only need to modify the RatingLookup worksheet to include a column of pulse rate boundary values for women. The existing pulse rate boundary values are for men. Your RatingLookup worksheet should end up like Figure 3.14.

Figure 3.14 – the lookup table for fitness ratings of men and women

	A	B	C
1	**Men**	**Women**	**Fitness Rating**
2	0	0	Excellent
3	80	87	Good
4	90	100	Average
5	105	111	Fair
6	115	123	Poor

The IF function with nested LOOKUP functions should be similar to the following (similar because you might have chosen to define different names for the data):

```
=IF(Gender="M",LOOKUP(Pulse_Rate,Men,Rating),LOOKUP(Pulse_Rate,Women,Rating))
```

You will have to type parts of this formula since the point and click method of nesting functions does not allow you to return to a previous function in order to complete the specification of its arguments if you have already used a nested function as an argument.

Ex 5.3 - Age and Gender Dependence

In fact the fitness rating of an individual depends not only on their gender but also on their age, according to the following table:

For males:

Ages:	**6-15**	**16-29**	**30-60**	**Fitness**
	Pulse Rate	**Pulse Rate**	**Pulse Rate**	**Rating**
	less than 82	less than 75	less than 76	Excellent
	82 to <92	75 to <83	76 to <90	Good
	92 to <103	83 to <92	90 to <106	Average
	103 to <113	92 to <99	106 to <120	Fair
	113 and greater	99 and greater	120 and greater	Poor

For females:

Ages:	**6-15**	**16-29**	**30-60**	**Fitness**
	Pulse Rate	**Pulse Rate**	**Pulse Rate**	**Rating**
	less than 92	less than 84	less than 86	Excellent
	92 to <104	84 to <94	86 to <100	Good
	104 to <118	94 to <105	100 to <117	Average
	118 to <130	105 to <116	117 to <130	Fair
	130 and greater	116 and greater	130 and greater	Poor

This is considerably more difficult to implement in Excel. Give it a try to see if you are up to the task.