Advanced Excel

TeachUcomp, Inc.

...it's all about you

TeachUcomp, Inc.
Phone: (877) 925-8080
Web: http://www.teachucomp.com

Introduction and Overview

Welcome to Teachucomp, Inc.'s Advanced Excel course. This class caps the student's knowledge of Microsoft Excel, one of the most popular spreadsheet programs available today. This class is designed to give the student a firm grasp of the advanced concepts in Excel. We will spend much time understanding the database-like qualities that Excel possesses and enhancing our skills in using formula functions.

Excel is an excellent program to learn, as the skills that we learn in Excel apply to many other programs as well, especially Access. It is the recommended starting point for learning database programs as it contains some basic database functionality and features.

Excel is a multi-featured spreadsheet program in which you can create powerful spreadsheets that can manipulate numbers and store data for you. It is a very powerful program, and has many advanced features that can automate and simplify your work. Whether you want it to create charts, spreadsheets, or data sources, Excel can assist you in accomplishing your tasks quickly and easily.

This class will focus on the advanced concepts of the Excel program. We will learn how to create and manage databases in Excel, write advanced formulas, sort and organize data, create forms, and set up macros.

Table of Contents

Chapters/Lessons: Page(s):

CHAPTER 23- TABLES

23.1- CREATING A TABLE

23.2- ADDING AND EDITING RECORDS

23.3- INSERTING RECORDS AND FIELDS

23.4- DELETING RECORDS AND FIELDS

23.1- Creating a Table:

You can use Excel to store information in tables in Excel. If you used Excel 2003, the tables were called "lists" for some reason. These two terms refer to the same objects, however. They are simply being referred to properly as tables now. An Excel table is simply information stored in a table format. This simply means that the different types of information that you want to collect are stored in columns, which are also often called "fields" in database terminology. Each "field" contains a separate type of information. Examples could be: "First Name," "Last Name," "Title," "Address," "City," "State," and so forth. Each row in the table is called a "record." A record is a single entry in which you record each type of field information about one "thing" in your database. For example, using the fields in the previous example, a record in that table might contain the information: "John," "Doe," "Mr.," "111 Nowhere Ln.," "Holt," "MI."

In a table there can be no **entirely** blank columns or rows! Anytime you leave an **entire** row or column blank, Excel assumes that it is the end of the table. Therefore, any records that you enter below the skipped line or after the skipped column will not be treated as if it were part of the same table.

Before you create a table in Excel, you need to consider what information you want to collect. Sometimes, it is easier to think of what fields to create after thinking of the subject of the table, first. For example, if you wanted to create a table to record customer data, you would need to think about what information you want to collect about your customers. The types of information that you decide to track will become the "fields" (columns) in your table.

For the purpose of the example, assume that you decided to record your customer's name, address, city, state, and zip code. When thinking of the field structure of the table, you need to consider just how detailed you want to be with the customer's information. Poor decisions in the planning phase can be problematic later. For example, do you want to record the customer's name in one field or in more than one field? If you ever want to sort the database by the last name of the customer, you will probably want to store the customer's name in at least two fields: "firstname" and "lastname." Noting little things like this during the creation process can save time in editing the table structure later on, after it becomes a problem. Once you have decided what information you would like to record in which field, you enter the titles of these fields as the top row of your table. The top row in your table is a special row and is often called the "field names row" or the "header row." It is always the top row in a table, and it displays the names of the fields for which you are collecting data.

Once you have the header row created, you can click and drag over the header row and then define it as a being a "table." This will make some of the table management features of Excel easier to use. To do this after selecting the header row, click the "Table" button in the "Tables" group on the "Insert" tab in the Ribbon. In the "Create Table" dialog box that appears, you will see the reference to the selected cells appear in the "Where is the data for your table:" text box. You simply check the "My table has headers" checkbox and then click the "OK" button.

This will then create the table area within the worksheet and add a new row- into which you can enter your first table record. If you have clicked into the table area, you will note that each field in the header row has a drop-down button applied to it. These are "Auto-Filters," which are used to filter data in the table. We will look at using those in a later section. You will also notice that the table has a different formatting than the rest of the worksheet area. This table style encloses any records that you want to identify as being part of your table. Note that you can place your mouse pointer over the lower right corner of the table, until you see a double-pointed, black arrow appear, and then click an drag with your mouse to resize the table border, if needed. However, don't enclose any entirely blank rows or columns, as you probably don't want a bunch of empty columns or rows in your table. This can interfere with sorting and filtering the data that you do want to store within the table.

23.2- Adding and Editing Records:

Once you have created the field structure of the table and labeled the "header row," you are then ready to enter your first record. A record is simply all of the information (determined by the fields) that pertains to a single entry. In the example of a "customer table," it would be the information that pertains to a single customer. You enter the first record in a table immediately under the header row. Unlike Excel 2003, the next row will not automatically appear to expand the border of your table when you enter a new record. Instead, simply use the "Tab" key to move to the next new row when you are finished entering one record to automatically add a new row to the table where you can then enter your next record.

Each piece of information recorded should match the corresponding field into which it is entered. For example, if you have a customer table with the fields: "title," "firstname," and "lastname," and a customer named "Mrs. Jane Smith;" then her record would look like this in the table: |Mrs.| Jane| Smith|. Additional records are always appended to the next row in the table, never entirely skipping a row.

After you have created your table and entered your records, you can edit the information in the cells just as you would in a normal worksheet. Just select the cell that contains the information that you want to edit and then change it. Use the "Tab" key to exit and move to the next cell in the table when finished with data entry. Just as when entering records, you can also format any records or information as desired.

23.3- Inserting Records and Fields:

You can also insert records or fields from a database into the middle of the table rather than appending them onto the bottom row of the table. This is the same process used to insert new columns or new rows. So the number that you select will be the number that you insert. Also, new columns will insert to the left of the selection and new rows will insert above the selection. You start by clicking and dragging over the headings of the columns or rows, selecting the number to insert. You can then then right-click on the selected headings. From the pop-up menu that appears, select the "Insert" command. Remember to fill the new columns or rows with data so that you do not have any entirely blank rows or columns in your table.

23.4- Deleting Records and Fields:

You can also delete records or fields from a database by right-clicking on the gray column or row heading and then selecting "Delete" from the pop-up menu that appears. Note that this will remove the **entire** column or row, allowing the database to remain intact with no skipped columns or rows. Make sure that you do not simply select some cells and then clear their contents as this tends to leave entirely blank columns and rows, which we should not have in a table.

ACTIONS- TABLES

CREATING A TABLE:

1. Select the cell into which you want to type your first field name.
2. Type the field name, and then press the “Tab” key on your keyboard to move to the cell to the right.
3. Repeat step 2 until you have made your entire header row.
4. Select the header row by clicking and dragging over the cells that contain the header row labels.
5. Click the “Table” button in the “Tables” group on the “Insert” tab in the Ribbon.
6. In the “Create Table” dialog box, check the “My table has headers” checkbox.
7. Click “OK.”

ADDING NEW RECORDS TO A TABLE:

1. Select the first field cell in the next available empty row within the table.
2. Type the information into the field.
3. Press “Tab” on your keyboard to move to the cell to the right.
4. Enter the appropriate data for that field.
5. Repeat steps 3 and 4 until the new record is fully entered. Then press the “Tab” key on your keyboard to create a new blank row for the next record.

EDITING TABLE RECORDS:

1. Select the cell that contains data that you want to edit.
2. Make any changes to the data as necessary.
3. Press “Tab” on your keyboard to exit the cell and save your changes.

INSERTING NEW ROWS INTO A TABLE:

1. Right-click on the gray row heading above which you want to insert a new row.
2. Select “Insert” from the pop-up menu that appears.
3. Move into the new row that you inserted and enter all of the information for that record.

INSERTING NEW FIELDS INTO A TABLE:

1. Right-click on the gray column heading to the right of where you would like to insert the new field.
2. Select the “Insert” command from the pop-up menu that appears.
3. Select the cell that is in the header row and type the label for the new field.
4. Press “Enter” on your keyboard to exit the cell and move down into the column.
5. Enter the new field’s information for all of your existing records, if needed.

DELETING COLUMNS AND ROWS FROM A TABLE:

1. Select the column headings or row headings for the fields or records you want to delete.
2. Right-click the selected headings.
3. Select “Delete” from the pop-up menu that appears.

EXERCISES- TABLES

Purpose:

1. To be able to create a table in Excel.

Exercises:

1. Open up the Excel application.
2. Create a new blank workbook.
3. Enter the data into the workbook region shown in the picture below in “Sheet1” of the new workbook.
4. Select the data that you have entered, including the header row.
5. Click the “Table” button in the “Tables” group on the “Insert” tab in the Ribbon.
6. In the “Create Table” dialog box, check the “My table has headers” checkbox, and then click “OK.” Your table should appear as shown in the picture below when you select a cell within the table area.

	A	B	C	D	E	F	G	H
1	Region	Salesperson	City	Quarter	Month	Date	Daily Sales	
2	North	Joy Jameson	Lansing	First	Jan	1/2/2007	$ 3,426.00	
3	North	Ken Sterling	Lansing	First	Jan	1/2/2007	$ 2,203.00	
4	North	Joshua Smith	Lansing	First	Jan	1/2/2007	$ 417.00	
5	North	Donna Smith	Detroit	First	Jan	1/2/2007	$ 7,012.00	
6	North	Jeffrey Richards	Detroit	First	Jan	1/2/2007	$ 6,407.00	
7	North	Mekhi Jones	Detroit	First	Jan	1/2/2007	$ 4,703.00	
8	North	Laverne Lawless	Chicago	First	Jan	1/2/2007	$ 8,179.00	
9	North	Sara Swanson	Chicago	First	Jan	1/2/2007	$ 5,391.00	
10	North	Hector Veracruz	Chicago	First	Jan	1/2/2007	$ 1,115.00	
11								

7. Click the “Save” button within the Quick Access toolbar to open the “Save As” dialog box.
8. Use the “Save As” dialog box to save the file to the “Documents” folder on your computer, and name the file “Adv- Sample.”
9. Click the “X” button in the upper right corner of the application window to exit.

CHAPTER 24- SORTING DATA

24.1- SORTING DATA

24.2- CUSTOM SORT ORDERS

Sorting Data

24.1- Sorting Data:

Any table can be sorted using any of the fields available. Sorting is one of the main reasons that you create tables. It allows you to easily organize information in the table records. By default, Excel can sort alphabetically or numerically in either "ascending" (A-Z, 1-9) or "descending" (Z-A, 9-1) order. It will sort from top to bottom by default, but it can also be set to sort from left to right. You can sort a table by a single column within the table or by multiple columns.

When you sort data, you need to define by which fields it will sort the data and in which order it will sort the data. For example, if you sorted only by the "last name" field in a table, there might be multiple entries with the same last name. That is when applying a secondary sort, by the "first name" field for example, can ensure that your employees are in alphabetical order by last name, then by first name.

To sort a table by a single column, just click into a cell within the column by which you'd like to sort the data. Then click either the "Sort A to Z" or "Sort Z to A" buttons in the "Sort & Filter" group on the "Data" tab in the Ribbon to sort the table in either ascending or descending order by the column's data values.

You can perform a multi-column sort on a table by first clicking into the table and then clicking the "Sort" button in the "Sort & Filter" group on the "Data" tab in the Ribbon. In the "Sort" dialog box that appears, just use the first "Sort by" drop-down to select the name of the first field by which you want to sort the data. This is called the "primary sort." To the right of that, ensure that the "Sort On" drop-down is set to "Values." Then select the desired option for the primary sort order: "A to Z" or "Z to A."

You can then add more sorting levels by clicking the "Add Level" button. You can then repeat the process using the next sorting row to create additional sorting levels, if desired. When you have finished adding all of the desired sorting levels into this dialog box, click the "OK" button to apply the sorting that you set. If you make a mistake after sorting data, you can undo it by clicking the "Undo" button in the Quick Access toolbar immediately after performing the sort.

You can remove sorting that has been applied to a table by simply clicking the "Sort" button in the "Sort & Filter" group on the "Data" tab in the Ribbon, and then selecting the sorting level to delete from the "Sort" dialog box. Then click the "Delete Level" button in the "Sort" dialog box to delete the selected sorting level. Then click the "OK" button when you are finished.

24.2- Custom Sort Orders:

Sometimes you may want to sort data using a sort order that isn't ascending or descending. These are called "custom sort orders." An example of a custom sort order would be the chronological sorting of the days of the week: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday. You can create your own custom sort orders in Excel. Once they are created, custom sort orders are available to all worksheets in Excel.

To create a custom sort order, type the items that you would like to have in the custom sort order down a column in a worksheet. Then click and drag over the items to select them. Next, if using Excel 2007 click the Microsoft Office button and select the "Excel Options" button to display the "Options" dialog box. Then click the "Popular" category at the left side of the dialog box. If using Excel 2010, click the "File" tab in the Ribbon and then click the "Options" command at the left side of the backstage view to open the "Options" dialog box. Then click the "Advanced" category at the left side of the "Options" dialog box.

In either version, you will next click the "Edit Custom Lists…" button to open the "Custom Lists" dialog box. In this dialog box, you will see a reference to the currently selected cell range in the text box in the lower right corner. Click the "Import" button that to import the contents of the selected cells into the "List entries:" box. The custom sort order will also appear in the "Custom lists:" box at the left side of this tab. Note that you can also delete custom lists that you have created in this tab, as well. To delete a custom list

24.2- Custom Sort Orders (cont.):

that you have created, select it from the “Custom lists:” list box at the left side of this tab. Then click the “Delete” button at the right side of this tab. You will have to click “OK” in the confirmation dialog box that appears to delete the selected list. When you are finished creating and deleting lists in this dialog box, click “OK” to save your changes.

To apply a custom sort order, click anywhere into the data table which you want to sort and then click the “Sort” button in the “Sort & Filter” group on the “Data” tab in the Ribbon. In the “Sort” dialog box that appears, use the “Sort by” drop-down to select the name of the field that contains the data to which you want to apply a custom sort order. Next, use the “Sort On” drop-down to select what you want to sort. This will typically be “Values.” Then click the “Order” drop-down and select “Custom List…” from the drop-down menu. In the “Custom Lists” dialog box, select the custom sort order to use, and then click “OK” to return to the “Sort” dialog box. Here you can then click “OK” to apply the custom sort to the specified field.

ACTIONS-
SORTING DATA

SORTING DATA:

1. To sort a table by a single column, just click into a cell within the column by which you'd like to sort the data. Then click either the "Sort A to Z" or "Sort Z to A" buttons in the "Sort & Filter" group on the "Data" tab in the Ribbon to sort the table in either ascending or descending order by the column's data values.
2. You can perform a multi-column sort on a table by first clicking into the table and then clicking the "Sort" button in the "Sort & Filter" group on the "Data" tab in the Ribbon.
3. In the "Sort" dialog box that appears, just use the first "Sort by" drop-down to select the name of the first field by which you want to sort the data. To the right of that, ensure that the "Sort On" drop-down is set to "Values." Then select the desired option for the primary sort order: "A to Z" or "Z to A."
4. You can then add more sorting levels by clicking the "Add Level" button. You can then repeat step 3 above, using the next sorting row to create a additional sorting levels, if desired.
5. When you have finished adding all of the desired sorting levels into this dialog box, click the "OK" button to apply the sorting that you set.
6. If you make a mistake after sorting data, you can undo it by clicking the "Undo" button in the Quick Access toolbar immediately after performing the sort.
7. You can remove sorting that has been applied to a table by simply clicking the "Sort" button in the "Sort & Filter" group on the "Data" tab in the Ribbon, and then selecting the sorting level to delete from the "Sort" dialog box. Then click the "Delete Level" button in the "Sort" dialog box to delete the selected sorting level. Then click the "OK" button when you are finished.

USING CUSTOM SORT CRITERIA:

1. To create a custom sort order, type the items that you would like to have in the custom sort order down a column in a worksheet. Then click and drag over the items to select them.
2. Next, if using Excel 2007 click the Microsoft Office button and select the "Excel Options" button to display the "Options" dialog box. Then click the "Popular" category at the left side of the dialog box.
3. If using Excel 2010, click the "File" tab in the Ribbon and then click the "Options" command at the left side of the backstage view to open the "Options" dialog box. Then click the "Advanced" category at the left side of the "Options" dialog box.
4. In either version, click the "Edit Custom Lists…" button to open the "Custom Lists" dialog box.
5. In this dialog box, you will see a reference to the currently selected cell range in the text box in the lower right corner. Click the "Import" button to import the contents of the cells into the "List entries:" box.
6. To delete a custom list that you have created, select it from the "Custom lists:" list box at the left side of this tab. Then click the "Delete" button at the right side of this tab. You will have to click "OK" in the confirmation dialog box that appears to delete the selected list.
7. When you are finished creating and deleting lists in this dialog box, click "OK" to save your changes.
8. To apply a custom sort order, click anywhere into the data table which you want to sort and then click the "Sort" button in the "Sort & Filter" group on the "Data" tab in the Ribbon.
9. In the "Sort" dialog box that appears, use the "Sort by" drop-down to select the name of the field that contains the data to which you want to apply a custom sort order.
10. Next, use the "Sort On" drop-down to select what you want to sort. This will typically be "Values."
11. Then click the "Order" drop-down and select "Custom List…" from the drop-down menu.
12. In the "Custom Lists" dialog box that appears, select the custom sort order to use, and then click the "OK" button to return to the "Sort" dialog box.
13. In the "Sort" dialog box, click the "OK" button to apply the custom sort to the specified field.

EXERCISES- SORTING DATA

Purpose:

1. To be able to sort data in a table.

Exercises:

1. Open up the “Adv- Sample” workbook.
2. Click cell B3 to select it.
3. Click the “Sort A to Z” button in the “Sort & Filter” group on the “Data” tab in the Ribbon to sort the table by the “Salesperson” field in ascending order.
4. Click the “Sort” button in the “Sort & Filter” group on the “Data” tab in the Ribbon.
5. Use the “Sort by” drop-down to select “City.”
6. Use the “Sort On” drop-down to select “Values.”
7. Use the Order” drop-down to select “A to Z.”
8. Click the “Add Level” button to create a new sorting level.
9. Use the “Then by” drop-down to select “Daily Sales.”
10. Use the “Sort On” drop-down to select “Values.”
11. Use the “Order” drop-down to select “Largest to Smallest.”
12. Click the “OK” button.
13. Click the “Clear” button in the “Sort & Filter” group on the “Data” tab in the Ribbon.
14. Click the “Save” button in the Quick Access Toolbar to save your changes.
15. Click the “X” button in the upper right corner of the application window to exit.

CHAPTER 25- FILTERING DATA

25.1- Using AutoFilters:

AutoFilter is a useful tool that allows you to quickly filter your data tables. However, you **must** have a header row in your table for AutoFilter to function. When you defined your table in Excel, you automatically have AutoFilter drop-downs placed into the header row for each column. You can toggle the AutoFilters on and off in your table by selecting any cell in your table and then clicking the "Filter" button in the "Sort & Filter" group on the "Data" tab in the Ribbon.

When AutoFilter is enabled, if you click on any of the drop-down arrows on the field headers, you will see a drop-down list of available filtering options followed by a listing of all of the different and unique values found within the selected field. You can manually filter a list to ensure that only the values that you want to see are checked within the list of values shown. If you click the "(Select All)" choice when all of the values are checked, it will uncheck all of them. This will allow you to quickly check only the values which you wish to see. When you click the "OK" button, Excel will filter the database table to show only records in the table that have a matching value for that field. All of the other records will be hidden, not deleted.

You can tell when a filter is being applied, and to which field it is being applied, because when a filter is being used on a field the drop-down arrow next to that field appears with a small "funnel" icon to let you know that a filter is being applied. You can apply multiple kinds of filters using this tool, and you may also apply more than one filter on a table at a time using the various columns.

Once you have applied an AutoFilter, you can remove it to show all of the records again. One way is to select the AutoFilter drop-down arrow that is currently being applied, and then check the "(Select All)" checkbox from the list of values shown in the drop-down menu to show all of the records that were being hidden by the filter. Alternately, you could also click the "Clear" button in the "Sort & Filter" group on the "Data" tab in the Ribbon.

25.2- Using the Top 10 AutoFilter:

You can apply a special type of filter to number fields called a "Top 10 AutoFilter" that can show you a specified number of the top or bottom percent or items in a field within the table. For example, it defaults to showing the top 10 percent of a column, but you could also change it to show you the bottom 5 items (by value) with a column, as well. Note that this filter cannot be applied to text fields, as they have no numeric ranking by which to base a value.

To apply a "Top 10 AutoFilter," click the AutoFilter drop-down arrow button next to the column heading for the field by which you want to filter the table. Next, roll down to the "Number Filters" choice. Then select the "Top 10…" option from the side menu that appears to open the "Top 10 AutoFilter" dialog box. In this dialog box, select the first drop-down and pick either "Top" or "Bottom." Next, enter a number into the spinner box in the center of the dialog box. Finally, use the drop-down on the right to pick either "Items" or "Percent." What is displayed across the dialog box is the filter setting that you will apply to the selected column. So you could view only the "Top 10 Items," or "Bottom 40 Percent", or any other variation using this dialog box. When you have created the desired filter, just click "OK" to apply it.

25.3- Applying a Custom AutoFilter:

You can also display a special type of filter called a "Custom AutoFilter" that you can customize by using any available comparison operator in conjunction with wildcard characters. Using a custom AutoFilter allows you to display records using a custom set of comparison criteria that you create. To apply a custom AutoFilter, just click the AutoFilter drop-down arrow button next to the column header

25.3- Applying a Custom AutoFilter (cont.):

by which you want to filter the data. Roll down to either the "Text Filters," "Number Filters" or "Date Filters" command in the drop-down menu. The command name changes depending upon the type of data stored within the column. In the side menu that appears, you will see many of the available comparison operators. You can select one of these, if desired, or you can simply choose the "Custom Filter…" command. When you do this, Excel will open the "Custom AutoFilter" dialog box.

In the "Custom AutoFilter" dialog box, you select the desired comparison criteria from the upper left drop-down box. Next to it, in the upper right drop-down box, you can either enter a value or select a value from the drop-down list. If you enter a value, note that you can choose to use wildcard characters to represent unknown data values within the field, if desired. The characters that you can use, and what they represent, are shown in a small listing at the bottom of this dialog box. So, for example, if you entered "J??" as a criteria value, you would filter for 3-character words that start with a "j," like "Joe," "Jim," "Jam," and others. If you specified "J*" as your filter criteria, you would filter for any word that starts with the character "J," like "Jennifer," "Joe," "Jacob," and so on, regardless of the length of the word. Wildcard characters can appear before or after the known values to find unknown characters that either proceed, appear between, or follow the known characters.

You can also specify a second filter criteria as you create the first, if you wish. To do this, create the first criteria and then select either the "And" or "Or" option buttons in the middle of the "Custom AutoFilter" dialog box to connect the two conditions that you create. If you select the "And" option, then the value in the field must meet both conditions to be displayed. If you select the "Or" condition, the field values will be displayed if they meet either criteria specified. Next, create the second criteria in the same way that you created the first criteria.

When you have created your own custom filter criteria, just click "OK" to apply the selected criteria to the selected field.

25.4- Creating Advanced Filters:

You can create advanced filters that will allow you to filter data using multiple "And" and "Or" criteria, if necessary. To create an advanced filter, however, you must first create a criteria range within the workbook into which you will enter the criteria to filter your table. A criteria range is a duplicate of the header row of your table that is physically separated from the rest of the table. It *must* contain a header row of fields that is identical to the header row of the table that you are going to filter. Below this header row, you then enter the criteria against which you want to filter the table.

When creating a criteria range, it may be useful to note that it does not have to be on the same worksheet. You could have one worksheet with the table data, and then a separate worksheet that contains your criteria range or criteria ranges. You can create multiple criteria ranges if you like, but you can only apply one range at a time to the table. No matter where you decide to place the criteria range within your workbook, however, you *must* leave room between the criteria range and your table if they are contained within the same worksheet! This is very important for the filter to work correctly. Also, the criteria range must be at least one column by two rows in length. This leaves at least one row available for entry of filtering criteria. Also, the field names listed in criteria rows must be spelled exactly as they appear in the table, but are not case-sensitive.

As long as you have created a criteria range that consists of one top row of criteria labels, and at least one row beneath it where you can enter criteria conditions, then you can use the criteria range to create an advanced filter. Before applying the filter, you must enter the necessary criteria conditions under the corresponding fields within the criteria rows in the criteria range. When you have created a criteria range

25.4- Creating Advanced Filters (cont.):

and entered at least one criteria that you want to use as a filter, you are then ready to apply the filter.

To apply the filter, select a cell in the table to which you want to apply the filter, and then click the "Advanced" button in the "Sort & Filter" group on the "Data" tab in the Ribbon. Doing this will open the "Advanced Filter" dialog box.

At the top of the "Advanced Filter" dialog box in the "Action" section, choose the "Filter the table, in-place" option button. The "List range:" text box should show a reference to the table that will be filtered. All you have to do is click into "Criteria range:" text box, and then click and drag over the entire criteria range, including the header row and any criteria rows that you have added, to select it. Make sure your selection encompasses both the labels and the criteria in the criteria range. Then click "OK" to filter your table based on the criteria you supplied.

To remove the filter after applying it, click into the filtered data within your table and then click the "Clear" button in the "Sort & Filter" group on the "Data" tab in the Ribbon. This will then display all of the records in your table again.

25.5- Applying Multiple Criteria:

You can apply multiple criteria to the criteria rows in your criteria range. However, the conditions must be logically joined together using "And" and "Or" statements. You use the "And" condition to specify that a record must meet multiple criteria at the same time in order to be included in the filtered data. When creating the criteria in the criteria range to apply an advanced filter, you can create multiple criteria and join them together with the "And" condition by simply inserting both the criteria in the same criteria row, under their respective field headings in the criteria range prior to applying the filter.

For example if you had two fields, "Daily Sales" and "City," you could place =">2000" in one row under the "Daily Sales" column and you could place "=Chicago" under "City" in the **same** criteria row. As long as you place both of these criteria in the same row, they will be interpreted by Excel as being one criteria with an "And" condition joining them. If you applied this criteria, you would see all the records in your table where the city was listed as "Chicago" ***and*** there was more than 2000 dollars in "Daily Sales."

You use the "Or" condition to specify that a record can meet one or more of the criteria in order to be included in the filtered record set. To specify an "Or" condition, you write your first criteria in one of the criteria rows available, using multiple criteria in the same row for "And" conditions, if needed. Then you place the next set of criteria that you want to look for using an "Or" condition into a separate criteria row directly beneath the first criteria row. You can then apply the advanced filter. However, be sure to select all of the criteria rows that are needed when selecting the criteria range in the "Advanced Filter" dialog box.

Using the aforementioned example, assume that you wanted to see any record in your table where the city was equal to "Chicago" **or** the record contained "Daily Sales" greater than 2000. To do this you would have to place =">2000" under the "Daily Sales" field header in one criteria row, and then enter "=Chicago" under the "City" field header in a **separate** criteria row. If you then filter the table with this criteria, you would see all of the records in your table where the "City" field contained "Chicago" ***or*** where "Daily Sales" were greater than 2000.

It can sometimes be confusing to think of the "And" and "Or" conditions, and be able to pick which one you need at first. Just remember that if you use the "And" condition, a record must meet all of the criteria joined together by the "And" condition in order to be included in the result set. If you use the "Or" condition, a record can meet any one of the criteria joined by the "Or" condition to be included in the result set.

25.6- Using Complex Criteria:

You have many advanced criteria that you can create in the criteria range in order to filter your selected table. In this lesson, we will review some of the advanced criteria techniques that you can apply to the criteria entered into the criteria range when creating an advanced filter.

First, it should be mentioned that when you use a comparison operator, the values must be proceeded by the equal sign. However, this causes Excel to evaluate the entry that follows the equal sign as if it were a formula. This can cause unexpected results when using comparison operators with text and number fields. Therefore, you should enter the comparison criteria in the following general format: ="*=entry*" where *=entry* is the comparison operator and associated text or value by which you wish to filter the field.

Many times when filtering data, you wish to see records where a field's value is equal to a value entered into the criteria range. In these cases, you would enter ="=entry" as the criteria. However, there are many other comparison criteria that you can use for text and number fields You can use the following comparison operators when creating complex criteria: > (greater than), < (less than), >= (greater than or equal to), <= (less than or equal to), and <> (does not equal). So for example, entering the criteria expression of ="<>Chicago" under a "City" field in the criteria range would filter the table to show all records where the "City" field did not equal "Chicago."

Also note that you can use wildcard characters in order to search for unknown values. You can use the asterisk (*) to represent multiple unknown characters and you can use the question mark (?) to represent a single unknown character. To use a wildcard character, just decide under which text field you want to place the criteria and in which criteria row. Then simply use the wildcard characters in conjunction with the known values to search for matching records. For example, if you had a "Last Name" column from which you wanted to filter for anyone whose last name started with a "J," you could enter ="=J*" under the "Last Name" column as a wildcard criteria.

It is also possible to create a criteria that matches the first few characters in the field's values and returns any matching values. This is akin to setting a criteria that looks for fields that "begin with." In this case, you do not enter any comparison operator into the criteria, but rather enter only the first characters for which you want to find matches. For example, entering "Car" as a criteria underneath a "Last Name" field would return last names that begin with "Car," like "Carson" and "Carlisle."

25.7- Copying Filtered Results to a New Location:

You can copy the results of an applied filter to a new location instead of having to always filter the table in its original place. To do this, create your criteria filters as you normally would, then click into the data that you wish to filter, and then click the "Advanced" button in the "Sort & Filter" group on the "Data" tab in the Ribbon.

In the "Advanced Filter" dialog box, choose the option for "Copy to another location" in the "Action" section at the top. Then select the cells that contain your "Table range:" and "Criteria range:" as normal. Next, click into the "Copy to:" text box and then click the cell that will become the upper left corner of the cell range to which you wish to paste the filtered data. Finally, click "OK" to filter the table and copy the results to the location you selected.

One point to note is that when you do copy the results to a new location, the location specified must be in the active worksheet- meaning the worksheet that contains the table that is being filtered.

25.8- Using Database Functions:

You can use the criteria range that you have created to also perform database functions on the data stored in your table. Database functions are much like your typical functions, like “AVERAGE,” “SUM,” or “PRODUCT,” but will only be performed on values in rows that match a certain criteria that you specify.

For example, if you wanted to “SUM” a “Daily Sales” field, but **only** if the “City” field was equal to “Chicago,” you could use the database function of “DSUM” to do that. Most of the database functions simply perform your usual mathematical operations, but only for records that match the criteria that you specify. You can view the various database functions by looking at the “Database” category of functions shown in the function drop-down list inside the “Insert Function” dialog box.

You’ll notice that these functions are all strikingly similar in the structure of their syntax. Most are simply equivalent to the same function, but with the letter “D” added to the front. For example, you use the “AVERAGE” function to find the average of selected cells. To use the database average, the function simply becomes “DAVERAGE.” Second, each database function only requires three arguments: “Database,” which is the table cell range reference; “Field,” which is the name of the field (enclosed in double quotes), or the column number of the field, that contains the values upon which you want to perform the selected function; and “Criteria,” which is a cell reference to the criteria range which contains the criteria used to decide which rows will be included in the function.

ACTIONS-
FILTERING DATA

USING AUTOFILTERS:

1. You can toggle the AutoFilters on and off in your table by selecting any cell in your table and then clicking the "Filter" button in the "Sort & Filter" group on the "Data" tab in the Ribbon. When AutoFilter is enabled, if you click on any of the drop-down arrows on the field headers, you will see a drop-down list of available filtering options followed by a listing of all of the different and unique values found within the selected field.
2. You can manually filter a list to ensure that only the values that you want to see are checked within the list of values shown. If you click the "(Select All)" choice when all of the values are checked, it will uncheck all of them. This will allow you to quickly check only the values which you wish to see.
3. When you click the "OK" button, Excel will filter the database table to show only records in the table that have a matching value for that field. All of the other records will be hidden, not deleted.
4. Once you have applied an AutoFilter, you can remove it to show all of the records again. One way is to select the AutoFilter drop-down arrow that is currently being applied, and then check the "(Select All)" checkbox from the list of values shown in the drop-down menu to show all of the records that were being hidden by the filter.
5. Alternately, you could also click the "Clear" button in the "Sort & Filter" group on the "Data" tab in the Ribbon.

USING THE TOP 10 AUTOFILTER:

1. Click the AutoFilter drop-down arrow button next to the column heading for the field by which you want to filter the table.
2. Roll your mouse down to the "Number Filters" choice.
3. Select the "Top 10…" option from the side menu that appears.
4. In the "Top 10 AutoFilter" dialog box, select the first drop-down and pick either "Top" or "Bottom."
5. Next, enter a number into the spinner box in the center of the dialog box.
6. Finally, use the drop-down on the right to pick either "Items" or "Percent." What is displayed across the dialog box is the filter setting that you will apply to the selected column.
7. When you have created the desired filter, just click "OK" to apply it.

APPLYING A CUSTOM AUTOFILTER:

1. Click the AutoFilter drop-down arrow button next to the column header by which you want to filter the data.
2. Roll down to either the "Text Filters," "Number Filters" or "Date Filters" command in the drop-down menu. The command name changes depending upon the type of data stored within the column.
3. In the side menu that appears, you will see many of the available comparison operators. You can select one of these, if desired, or you can simply choose the "Custom Filter…" command. When you do this, Excel will open the "Custom AutoFilter" dialog box.
4. In the "Custom AutoFilter" dialog box, you select the desired comparison criteria from the upper left drop-down box.
5. Next to it, in the upper right drop-down box, you can either enter a value or select a value from the drop-down list

(cont.)

APPLYING A CUSTOM AUTOFILTER (CONT.):

6. You can also specify a second filter criteria as you create the first, if you wish. To do this, create the first criteria and then select either the "And" or "Or" option buttons in the middle of the "Custom AutoFilter" dialog box to connect the two conditions that you create. If you select the "And" option, then the value in the field must meet both conditions to be displayed. If you select the "Or" condition, the field values will be displayed if they meet either criteria specified. Next, create the second criteria in the same way that you created the first criteria.
7. When you have created your own custom filter criteria, just click "OK" to apply the selected criteria to the selected field.

CREATING ADVANCED FILTERS:

1. To create an advanced filter, you must first create a criteria range within the workbook into which you will enter the criteria to filter your table. A criteria range is a duplicate of the header row of your table that is physically separated from the rest of the table. It must contain a header row of fields that is identical to the header row of the table that you are going to filter. Below this header row, you then enter the criteria against which you want to filter the table. The criteria range must be at least one column by two rows in length. This leaves at least one row available for entry of filtering criteria.
2. When you have created a criteria range and entered at least one criteria that you want to use as a filter, you are then ready to apply the filter.
3. To apply the filter, select a cell in the table to which you want to apply the filter, and then click the "Advanced" button in the "Sort & Filter" group on the "Data" tab in the Ribbon. Doing this will open the "Advanced Filter" dialog box.
4. At the top of the "Advanced Filter" dialog box in the "Action" section, choose the "Filter the table, in-place" option button. The "List range:" text box should show a reference to the table that will be filtered.
5. Click into "Criteria range:" text box, and then click and drag over the entire criteria range, including the header row and any criteria rows that you have added, to select it. Make sure your selection encompasses both the labels and the criteria in the criteria range.
6. Click "OK" to filter your table based on the criteria you supplied.
7. To remove the filter after applying it, click into the filtered data within your table and then click the "Clear" button in the "Sort & Filter" group on the "Data" tab in the Ribbon.

APPLYING MULTIPLE CRITERIA:

1. You use the "And" condition to create multiple criteria that must all be met in order to display a records from the table. You join them together with the "And" condition by simply inserting the criteria in the **same** criteria row, under their respective field headings, in the criteria range prior to applying the filter.
2. You use the "Or" condition to specify that a record can meet one or more of the criteria in order to be included in the filtered record set. To specify an "Or" condition, you write your first criteria in one of the criteria rows available, using multiple criteria in the same row for "And" conditions, if needed. Then you place the next set of criteria that you want to look for into a **separate** criteria row directly beneath the first criteria row. You can then apply the advanced filter.
3. Be sure to select all of the criteria rows that are needed when selecting the criteria range in the "Advanced Filter" dialog box.

ACTIONS- FILTERING DATA

USING COMPLEX CRITERIA:

1. You should enter the comparison criteria in the following general format: =“=entry” where =entry is the comparison operator and associated text or value by which you wish to filter the field.
2. You can use the following comparison operators when creating complex criteria: > (greater than), < (less than), >= (greater than or equal to), <= (less than or equal to), and <> (does not equal).
3. Also note that you can use wildcard characters in order to search for unknown values. You can use the asterisk (*) to represent multiple unknown characters and you can use the question mark (?) to represent a single unknown character.
4. It is also possible to create a criteria that matches the first few characters in the field’s values and returns any matching values. In this case, you do not enter any comparison operator into the criteria, but rather enter only the first characters for which you want to find matches.

COPYING FILTERED RESULTS TO A NEW LOCATION:

1. Create your criteria filters as you normally would, and then click into the data that you wish to filter.
2. Click the “Advanced” button in the “Sort & Filter” group on the “Data” tab in the Ribbon.
3. In the “Advanced Filter” dialog box, choose the option for “Copy to another location” in the “Action” section at the top.
4. Select the cells that contain your “Table range:” and “Criteria range:” as normal.
5. Next, click into the “Copy to:” text box and then click the cell that will become the upper left corner of the cell range to which you wish to paste the filtered data.
6. Finally, click “OK” to filter the table and copy the results to the location you selected.
7. One point to note is that when you do copy the results to a new location, the location specified must be in the active worksheet- meaning the worksheet that contains the table that is being filtered.

USING DATABASE FUNCTIONS:

1. You can use the criteria range that you have created to also perform database functions on the data stored in your table. You can view the various database functions by looking at the “Database” category of functions shown in the function drop-down list inside the “Insert Function” dialog box.
2. Each database function only requires three arguments: “Database,” which is the table cell range reference; “Field,” which is the name of the field (enclosed in double quotes), or the column number of the field, that contains the values upon which you want to perform the selected function; and “Criteria,” which is a cell reference to the criteria range which contains the criteria used to decide which rows will be included in the function.

EXERCISES-
FILTERING DATA

Purpose:

1. To be able to filter table data.

Exercises:

1. Open up the “Adv- Sample” workbook.
2. Select a cell within the table.
3. Click the “City” AutoFilter drop-down and click the “(Select All)” choice to de-select all selections in the drop-down menu.
4. Click the “Lansing” filter choice.
5. Click the “OK” button in the AutoFilter drop-down menu.
6. Click the “City” AutoFilter drop-down and click the “(Select All)” choice to select all selections in the drop-down menu.
7. Click the “OK” button in the AutoFilter drop-down menu.
8. Select the “DailySales” AutoFilter drop-down, roll your mouse pointer down to the “Number Filters” menu choice, and select “Top 10…” from the side menu that appears.
9. Use the drop-downs and spinner boxes in the “Top 10 AutoFilter” dialog box to change the filter to read “Top 5 Percent.”
10. Click “OK.”
11. Use the “DailySales” AutoFilter drop-down to select “(Select All).”
12. Click “OK.”
13. Copy the cell range A1:G1 to the cell range A14:G14.
14. In cell C15, type =“=Detroit”.
15. In cell G15, type =“> 5000”.
16. In cell C16, type =“=Chicago”.
17. In cell G16, type =“> 5000”.
18. Select any cell in the table range of A1 through G10.
19. Click the “Advanced” button in the “Sort & Filter” group on the “Data” tab in the Ribbon.
20. Click into “Criteria Range:” text box.
21. Select cells A14:G16.
22. Click “OK” in the “Advanced Filter” dialog box.
23. Select any cell in the table range of A1 through G10.
24. Click the “Clear” button in the “Sort & Filter” group on the “Data” tab in the Ribbon.
25. Click into cell A19, and type “Average amount of sale from Detroit or Chicago that is over $5000 dollars:.”
26. Click into cell A20.
27. Type this formula: “=DAVERAGE(A1:G10,7,Criteria)” and exit the cell.
28. Select cell A20 and format it as “Currency.”
29. Select the cell range of A14:G20.
30. Click the “Clear” drop-down button in the “Editing” group on the “Home” tab in the Ribbon, and then choose the “Clear All” command.
31. Select cell A1.
32. Click the “Save” button in the Quick Access Toolbar to save your changes.
33. Click the “X” button in the upper right corner of the application window to exit.

CHAPTER 26-
USING WHAT-IF ANALYSIS

26.1- USING DATA TABLES

26.2- USING SCENARIO MANAGER

26.3- USING GOAL SEEK

26.1- Using Data Tables:

Data tables are a handy way of being able to change one or two variables in a formula to view and compare the different possible results in a comparison table. You can create either single-variable data tables or double-variable data tables. For example, you can create a single-variable data table that computes and compares the possible different loan payment amounts for a loan based on different interest rates. Using the same example, you could also create a double-variable data table that displays the different loan payments that could be made for various interest rates and different loan repayment lengths.

When you create a data table, you must organize it in a specific way based on whether it is a single-variable data table, or a double-variable data table. All data tables consist of three basic parts: an "input" column or "input" row (or both, if creating a double-variable data table); an "output" column or "output" row (or both, if creating a double-variable data table); and the formula that is being evaluated within the data table.

The variables that you wish to modify within the formula must be entered as separate cell references within the formula being evaluated. If needed, you can create a small table that contains the cells to which you make references in the data table's formula, unless you already have cells being referenced for that purpose already existing in the worksheet.

In a single-variable data table, you have the option of arranging the data table in either a columnar layout or in a row layout. In the "column" style, you place the formula being evaluated at the top of the "output" column, which is the right of the two columns. The left column is the "input" column that contains all of the possible values that you wish to evaluate for the variable reference used by the formula.

In the "row" style, you place the formula that is being evaluated in the cell to the left of the "output" row, which is the bottom of the two rows. The top row is the "input" row and contains all of the possible values that you wish to evaluate for the variable reference used by the formula.

In double-variable data tables, there is only one way to layout the data table. You must create both an "input" column and an "input" row. In a double-variable data table, the formula that is being evaluated is placed at the top of the "input" column and to the left of the "input" row. The square that extends to the right and down from there is the "output" grid, where we will see the various outcomes of the formula being evaluated for the different variable intersections that are placed into the "input" column and the "input" row.

To create the data table, select the cell range that includes the formula being evaluated, the "input" column(s) and/or row(s), and the "output" column(s) and/or row(s). Then click the "What-If Analysis" button in the "Data Tools" group on the "Data" tab in the Ribbon. Choose the "Data Table…" command from the drop-down menu to open the "Data Table" dialog box.

Next, click into "Row input cell:" text box and then select the cell in the worksheet that represents the changing variable shown in the "input" row, if you have one. Then click into the "Column input cell:" text box and select the cell in your worksheet that represents the variable that is changing in the "input" column, if you have one. In a single-variable table, you will only fill in one of these two boxes. The one which you choose depends upon whether you created the data table using a column layout or a row layout. In a double-variable data table you must fill-in both boxes, indicating which cell reference to use for your "input" column and which cell reference to use for your "input" row. When you are done, click the "OK" button in the "Data Table" dialog box to fill the data table with output information, based on the values you placed in your "input" column(s) and/or row(s).

Now you can adjust the values in the "input" column(s) and/or row(s) in your data table to view various possible outcomes of the formula that is being evaluated.

26.2- Using Scenario Manager:

Sometimes you want to create a worksheet that contains several sets of saved values that you can easily switch between in order to compare possible variations in projected data within the worksheet. In Excel, you can save different sets of values as a scenario in your worksheet. Then you can switch between the saved scenarios in order to compare the potential outcomes. For example, if you wanted to create a worksheet that could be used to display different financial projections, you could create different scenarios to project different factors that might occur to change the worksheet results. You could create a worksheet that contains the current year's revenues, and then create additional scenarios based on that data that show a 5% increase, a 10% increase, or a 4% downturn. You can then switch between these different sets of data, as needed.

To create a scenario based on your current data, click the "What-If Analysis" button in the "Data Tools" group on the "Data" tab in the Ribbon. Then select the "Scenario Manager…" command from the button's drop-down menu in order to open the "Scenario Manager" dialog box. Click the "Add…" button at the right side of this dialog box to open the "Add Scenario" dialog box. In the "Add Scenario" dialog box, type a name for the new scenario into the "Scenario name:" text box. Give it a name that describes which scenario you are modeling.

Next, click into the "Changing cells:" text box. You can then select the cells within the worksheet that you want to change for the given scenario. You can select up to 32 different cells in the worksheet to change, which will allow you to create complex scenarios. You can type a comment about the scenario into the "Comment:" text box, if desired. When you are ready to proceed, click "OK."

In the "Scenario Values" dialog box that appears, you enter the values for the selected variable cells in your worksheet. When you have the values that you want to shown in the scenario entered, click "OK." The new scenario will appear in the list of scenarios displayed within the "Scenario Manager" dialog box.

To show a scenario that you have created, you must select the name of the scenario to view in the "Scenario Manager," and then click the "Show" button. The selected cell values in the worksheet will change to the values saved by the scenario. To close the "Scenario Manager" dialog box and view the changes, you can click the "Close" button in the "Scenario Manager" dialog box.

It is important to note that if you want to show the data as it was in the worksheet before displaying the scenario, you may want to create a "Current Values" scenario, in which you display the values as shown before applying scenarios. That way, you can easily revert the data back it its original state. Otherwise, you will have to click "Undo" button to revert the data back to its original values.

You can edit any scenarios that you have created to modify the variable values associated with each scenario. To do this, open the "Scenario Manger" dialog box again and then select the name of the scenario to modify from the list shown. Then click the "Edit…" button to open the "Edit Scenario" dialog box. Here you can edit any of the "Changing cells:" listed. Click "OK" when you are done to save the changed scenario.

You can delete scenarios that you no longer need within the "Scenario Manager" dialog box, as well. To do this, open the "Scenario Manager" dialog box and then select the name of the scenario that you want to delete. Then click the "Delete" button in the "Scenario Manager" dialog box to remove it instantly.

You can also import other scenarios from different worksheets into your current worksheet. However, this is only effective if the changing cells are the same in both worksheets. To merge scenarios from one worksheet into another worksheet, open the worksheet from which you want to import the scenarios. Next, open the worksheet into which you wish to merge the scenarios from the first sheet. Open the "Scenario Manager" dialog box, and click the "Merge…" button. This will display the "Merge Scenarios" dialog box. In the "Merge Scenarios" dialog box, you use the "Book:" drop-down to select the workbook which contains the scenarios you want to import. You will then view all of the worksheets within the selected workbook. Click any worksheet name in the "Sheet:" list, and a message will display at the bottom of this

26.2- Using Scenario Manager (cont.):

dialog box that tells you how many scenarios are attached to that worksheet. Select the worksheet that contains the scenarios that you want to import, and click "OK" to import them into the current worksheet.

You can easily compare the different results of scenarios within a scenario report. This report will show the different scenarios in the current worksheet, the changing values in each, and the different results that they generate. This makes it easy to compare different projections.

To create a scenario report, click the "Summary…" button in the "Scenario Manager" dialog box. This will launch the "Scenario Summary" dialog box. Click the option button that represents the type of report that you want to generate: "Scenario summary" or "Scenario PivotTable report."

Next, click into the "Result Cells:" text box, and then select the cells within the workbook whose values are changed by the different scenarios. When you are ready to create the selected report, click "OK."

The report will appear as a new worksheet in the workbook, which you can then click to view. In the "Scenario Summary" report, you will show the data in an outlined format, which you can collapse and expand to compare data. In the "Scenario PivotTable" report, you will also view the same data, but it is laid out as a PivotTable report which you can modify to compare data as needed.

26.3- Using Goal Seek:

Excel provides another great tool for assisting you when know the result that you want a formula to return, but you do not know the value needed to create the desired result. This tool is called "Goal Seek." Anytime you have a situation where you know the result that you need to obtain, but are unsure of one of the values that would be needed in order to attain that goal, you can use "Goal Seek" to help you find the missing value required.

For example, if you knew that you could pay up to $600 dollars per month on a loan, Goal Seek could help you determine what the amount that you could borrow would be for known loan terms.

To use "Goal Seek," just click the "What-If Analysis" button in the "Data Tools" group on the "Data" tab in the Ribbon. Then select the "Goal Seek…" command from the drop-down menu in order to open the "Goal Seek" dialog box. In this dialog box, you will need to enter three pieces of information: the cell that contains the result that you want to set to a given "goal" value, the "goal" value itself, and which cell contains the value that you need to change in order to achieve the goal value.

Click into the "Set cell:" text box and then click the cell within the worksheet that contains the result that you want to set to a given goal value. Then click into the "To value:" text box and enter the goal value that you want to find. Then click into the "By changing cell:" text box, and then click the cell in the worksheet that contains the variable that you want to change in order to achieve the desired result. Once that is entered, just click the "OK" button to instantly see the result. To accept the result and place it into your worksheet, click the "OK" button again or just click the "Cancel" button to discard the information.

ACTIONS-
USING WHAT-IF ANALYSIS

USING DATA TABLES:

1. Select the cell range that includes the formula being evaluated, the "input" column(s) and/or row(s), and the "output" column(s) and/or row(s).
2. Then click the "What-If Analysis" button in the "Data Tools" group on the "Data" tab in the Ribbon. Choose the "Data Table…" command from the drop-down menu to open the "Data Table" dialog box.
3. Next, click into "Row input cell:" text box and then select the cell in the worksheet that represents the changing variable shown in the "input" row, if you have one.
4. Then click into the "Column input cell:" text box and select the cell in your worksheet that represents the variable that is changing in the "input" column, if you have one.
5. When you are done, click the "OK" button in the "Data Table" dialog box to fill the data table with output information, based on the values you placed in your "input" column(s) and/or row(s).
6. Now you can adjust the values in the "input" column(s) and/or row(s) in your data table to view various possible outcomes of the formula that is being evaluated.

USING SCENARIO MANAGER:

1. To create a scenario based on your current data, click the "What-If Analysis" button in the "Data Tools" group on the "Data" tab in the Ribbon. Then select the "Scenario Manager…" command from the button's drop-down menu in order to open the "Scenario Manager" dialog box.
2. Click the "Add…" button at the right side of this dialog box to open the "Add Scenario" dialog box.
3. In the "Add Scenario" dialog box, type a name for the new scenario into the "Scenario name:" text box.
4. Next, click into the "Changing cells:" text box. You can then select the cells within the worksheet that you want to change for the given scenario. You can select up to 32 different cells.
5. You can type a comment about the scenario into the "Comment:" text box, if desired.
6. When you are ready to proceed, click "OK."
7. In the "Scenario Values" dialog box that appears, enter the values for the selected variable cells in your worksheet. When you have the values that you want to shown in the scenario entered, click "OK."
8. The new scenario will appear in the list of scenarios shown within the "Scenario Manager" dialog box.
9. To show a scenario that you have created, you must select the name of the scenario to view in the "Scenario Manager," and then click the "Show" button.
10. To close the "Scenario Manager" dialog box, you can click the "Close" button.
11. After showing a scenario, you can click the "Undo" button to revert the data back to its original values.
12. To edit a scenario that you have created, open the "Scenario Manger" dialog box again and then select the name of the scenario to modify from the list shown. Then click the "Edit…" button.
13. In the "Edit Scenario" dialog box, you can edit any of the "Changing cells:" listed.
14. Click "OK" when you are done to save the changed scenario.
15. You can delete scenarios that you no longer need. To do this, open the "Scenario Manager" dialog box and then select the name of the scenario that you want to delete. Then click the "Delete" button in the "Scenario Manager" dialog box to remove it instantly.
16. You can import other scenarios from different worksheets into your current worksheet. However, this is only effective if the changing cells are the same in both worksheets. To merge scenarios from one worksheet into another worksheet, open the worksheet from which you want to import the scenarios.
17. Next, open the worksheet into which you wish to merge the scenarios from the first sheet.

(cont.)

USING SCENARIO MANAGER (CONT.):

18. Open the “Scenario Manager” dialog box, and click the “Merge…” button.
19. In the “Merge Scenarios” dialog box, you use the “Book:” drop-down to select the workbook which contains the scenarios you want to import.
20. Click any worksheet name in the “Sheet:” list, and a message will display at the bottom of this dialog box that tells you how many scenarios are attached to that worksheet. Select the worksheet that contains the scenarios that you want to import, and click “OK” to import them into the current worksheet.
21. To create a scenario report, click the “Summary…” button in the “Scenario Manager” dialog box.
22. In the “Scenario Summary” dialog box, click the option button that represents the type of report that you want to generate: “Scenario summary” or “Scenario PivotTable report.”
23. Next, click into the “Result Cells:” text box, and then select the cells within the workbook whose values are changed by the different scenarios.
24. When you are ready to create the selected report, click “OK.”

USING GOAL SEEK:

1. To use “Goal Seek,” just click the “What-If Analysis” button in the “Data Tools” group on the “Data” tab in the Ribbon. Then select the “Goal Seek…” command from the drop-down menu in order to open the “Goal Seek” dialog box.
2. Click into the “Set cell:” text box and then click the cell within the worksheet that contains the result that you want to set to a given goal value.
3. Then click into the “To value:” text box and enter the goal value that you want to find.
4. Then click into the “By changing cell:” text box, and then click the cell in the worksheet that contains the variable that you want to change in order to achieve the desired result.
5. Once that is entered, just click the “OK” button to instantly see the result.
6. To accept the result and place it into your worksheet, click the “OK” button again or just click the “Cancel” button to discard the information.

EXERCISES-
Using What-If Analysis

Purpose:

1. To be able to use data tables and scenarios.

Exercises:

1. Open up the “Adv- Sample” workbook.
2. Select “Sheet2.”
3. Select cell F2, and type “Commission.”
4. Select cell H1, and type “Minimum Sales Amount.”
5. Select cell H2, and type “1000.” Format this as “Accounting Number Format.”
6. Select cell H3, and type “Percent Commission.”
7. Select cell H4, and type “.05.” Format this as a “Percent.”
8. Select cell H6, and type “Sold.”
9. Select cell I6, and type “Earned.”
10. Select cell H7, and type “1000.” Format this as “Accounting Number Format.”
11. Select cell I7, and type “=H2*H4.”
12. Select cell H8, and type “2000.” Format this as “Accounting Number Format.”
13. Select cell H9, and type “3000.” Format this as “Accounting Number Format.”
14. Select cell H10, and type “4000.” Format this as “Accounting Number Format.”
15. Select cell H11, and type “5000.” Format this as “Accounting Number Format.”
16. Select cells H7:I11.
17. Click the “What-If Analysis” button in the “Data Tools” group on the “Data” tab in the Ribbon.
18. Select the “Data Table…” command from the drop-down menu.
19. Click into the “Column input cell:” text box.
20. Select cell H2.
21. Click “OK.”
22. Format the selection as “Accounting Number Format.”
23. Select cell H4.
24. Click the “What-If Analysis” button in the “Data Tools” group on the “Data” tab in the Ribbon.
25. Select the “Scenario Manager…” command from the drop-down menu.
26. In the “Scenario Manager” dialog box, click the “Add…” button.
27. Type “Current Commission Rates” in the “Scenario name” text box.
28. Ensure that cell “H4” appears in the “Changing cells:” text box.
29. Click “OK.”
30. Click “OK” in the “Scenario Values” dialog box.
31. Click the “Add…” button again in the “Scenario Manager” dialog box.
32. Type “7% Commission” as the “Scenario Name.”
33. Click “OK.”
34. Type “.07” in text box 1, and then click the “OK” button.
35. Select the “Flat 7% Commission Rate” scenario, and click the “Show” button.
36. View the change in the worksheet.
37. Select the “Current Commission Rates” scenario, and click the “Show” button.
38. Click the “Close” button to close the “Scenario Manager” dialog box.
39. Select “Sheet1.”
40. Click the “Save” button in the Quick Access Toolbar to save your changes.
41. Click the “X” button in the upper right corner of the application window to exit.

CHAPTER 27-
TABLE-RELATED FUNCTIONS

27.1- THE HLOOKUP AND VLOOKUP FUNCTIONS

27.2- USING THE IF, AND, AND OR FUNCTIONS

27.1- The Hlookup and Vlookup Functions:

Now you will examine some formulas that are commonly used when dealing with tables. The first functions that we will discuss are in the "Lookup & Reference" function category. There are two primary functions within this function set that are used to lookup table data. You can use these functions to lookup a column value in a table and then return a corresponding column value from the same row. The first, "HLOOKUP," will look up a data value in a table that is structured in rows (with a "header column" on the left). The second and more commonly used function, "VLOOKUP," will lookup data values in a traditional columnar table. Based on the layout of the table in which you are looking up data, you will need to use either one or the other to lookup a data value in a table.

There are three required arguments and one optional fourth argument that you must know before you can create a lookup function. "Arguments" are the additional pieces of information that a function requires in order to be performed. First, you must know the "lookup value." This is the value which you want to lookup within the table. This is most commonly a cell reference. Second, you need to know the cell range reference of the table in which you want to lookup the first value. This is called the "table array." Third, you need to know the "column or row index number." This is the number of columns to the right of the leftmost column, or the number of rows down from the top row of the table, within which Excel must look for the data that you want it to return. The optional fourth argument is "range lookup." This is a logical value ("TRUE" or "FALSE") which you can enter, if needed. If omitted, this argument is assumed to be "TRUE." What it does is specify whether or not you wish to perform a "ranged" lookup on the data within the database. If you need to find an exact match on your data, often when you are looking up text values, you can insert "FALSE" for the "range lookup" argument to find an exact match in the table.

For example, let's say that you have a columnar customer table in a worksheet. In that sheet, you want to create a cell that will automatically lookup the customer's address when you type the customer's company name in an adjacent cell. Assume that the first column of the table contains the customer's company names and the third column of the table contains the customer's addresses. You can then write a function that will take the value of the cell adjacent to it (the lookup value), compare it to the customer table (the table array), lookup the customer's company name in the first column until it finds a match, and then count over three (3) columns (the column index number) to find the customer's address! Since we want the function to find an exact match on the customer name, we will then add the "FALSE" value to the "range lookup" argument in this case.

After you created the formula, you could type a customer name into the cell that is being looked up and the cell into which you had placed the formula would display the customer's address after looking it up in the table. In this case, we would use the "VLOOKUP" function because the example stated that we had a traditional columnar table in which we were looking up the data. The syntax of a lookup function is either:

=VLOOKUP(lookup_value,table_array,column_index_number,*range_lookup*)
OR
=HLOOKUP(lookup_value,table_array,row_index_number,*range_lookup*)

When you are entering the "table array" value for the lookup function, it may be useful to assign it a range name or make it an absolute reference. Named ranges are always assigned as absolute references and this will ensure that the "table array" argument will not change if you copy and paste this formula. If you don't use named ranges, you can assign the "table array" as an absolute reference by typing the dollar symbol ($) before **both** the column reference letter **and** the row reference number for both cell references in the range. Remember, if you don't do this, the lookup function may not work if you copy it to a new location.

27.2- Using the IF, AND, and OR Functions:

We will now discuss "Logical" functions, which you can use to perform logical tests on the values in cells and then return a result based on whether or not the value in the cell passed or failed the test. Logical formulas take the form of an "If…then…else" statement. You must know at least three different arguments before you can write a logical function: the "logical test" that you want to apply to the cell, the cell value or formula to return if the test returns a "TRUE" value (or "passes" the logical test), and the cell value or formula to return if the test returns a "FALSE" value (or "fails" the logical test). When you write logical functions, they must have a certain syntax. That is:

=IF(logical_test,true_response,false_response)

Note that if you want the formula to display a text response for the true response or the false response, then you must place the response inside of double quotation marks (" "). If you want the cell to display dates, these must be enclosed within pound signs (# #). The only time you wouldn't mark the type of value to return is if you want the cell to display a numerical result or calculate a formula.

A "nested" logical function is one that places the cell through a second logical test if it "fails" the first. These functions are useful for determining the value of a cell by placing it through several different tests, displaying different results based on which test it passes. You can nest up to 64 additional IF statements behind your original. The syntax for these are:

=IF(logical_test _1,true_response,IF(logical_test_2,true_response,false_response))

You must remember to close all of the open parenthesis for every IF statement that you nest within the logical function at the end of the formula. In this case since there are two IF statements, there are two closing parentheses at the end of the formula.

Often it is the case that you will want to know if a cell meets multiple criteria. You can use the AND and OR functions to find this out. The AND function will return a "TRUE" value if the cell being evaluated passes **all** of the tests listed after the AND function. The OR function will return a true value if the cell being evaluated passes **any** of the logical tests that follow the OR function. Note that you can evaluate up to 255 different logical test after the AND and OR statements.

When you look at how you can combine these tests with the IF function, or many nested IF functions, you can begin to see how you can start to become a very powerful formula creator. Now you can run cells through a battery of tests, and then decide what function to perform or value to display based on the results shown from the tests. The general syntax when combining the IF function with the AND and OR functions is as follows:

=IF(AND(logical_test_1,logical_test_2,logical_test_3,etc.),true_response,false_response)

=IF(OR(logical_test_1,logical_test_2,logical_test_3,etc.),true_response,false_response)

ACTIONS-
TABLE-RELATED FUNCTIONS

GENERAL SYNTAX FOR THE HLOOKUP AND VLOOKUP FUNCTIONS:

1. =VLOOKUP(lookup_value,table_array,column_index_number,*range_lookup*)
2. =HLOOKUP(lookup_value,table_array,row_index_number,*range_lookup*)

GENERAL SYNTAX FOR THE IF FUNCTION:

1. =IF(logical_test,true_response,false_response)

GENERAL SYNTAX FOR NESTING IF FUNCTIONS:

1. =IF(logical_test _1,true_response,IF(logical_test_2,true_response,false_response))

GENERAL SYNTAX FOR USING THE IF FUNCTIONWITH THE AND/OR FUNCTIONS:

1. =IF(AND(logical_test_1,logical_test_2,logical_test_3,etc.),true_response,false_response)

2. =IF(OR(logical_test_1,logical_test_2,logical_test_3,etc.),true_response,false_response)

EXERCISES-
TABLE-RELATED FUNCTIONS

Purpose:

1. To be able to table-related functions.

Exercises:

1. Open up the “Adv- Sample” workbook and open it to “Sheet1.”
2. Select cell J1, and type “City.”
3. Select cell K1, and type “Percent Commission.”
4. Select cell J2, and type “Lansing.”
5. Select cell K2, and type “5%.”
6. Select cell J3, and type “Detroit.”
7. Select cell K3, and type “6%.”
8. Select cell J4, and type “Chicago.”
9. Select cell K4, and type “7%.”
10. Select cell H1 and type “Commission.” Allow the table to expand to include this new column.
11. Select cell H2.
12. Type
    =IF(C2="Lansing",PRODUCT(G2,VLOOKUP(C2,$J$1:$K$4,2,FALSE)),IF(C2="Detroit",PRODUCT(G2,VLOOKUP(C2,$J$1:$K$4,2,FALSE)),IF(C2="Chicago",PRODUCT(G2,VLOOKUP(C2,$J$1:$K$4,2,FALSE)),"No reference available")))
13. If Excel doe not fill in the column for you, then copy this formula to cells H3:H10.
14. Format cells H2:H10 as “Accounting Number Format.”
15. Click the “Save” button in the Quick Access Toolbar to save your changes.
16. Click the “X” button in the upper right corner of the application window to exit.

CHAPTER 28- SECURITY FEATURES

28.1- UNLOCKING CELLS

28.2- WORKSHEET PROTECTION

28.3- WORKBOOK PROTECTION

28.4- PASSWORD PROTECTING EXCEL FILES

28.1- Unlocking Cells:

You can prevent changes to your worksheets and workbooks in Excel. If worksheet protection is turned on, you cannot change any information in cells that are "locked." This is a nice feature, but it renders the worksheet useless because all cells in a worksheet are "locked" by default! Therefore, before you protect your worksheets, you must "unlock" the cells where you know that data entry will need to occur.

To unlock worksheet cells, just select the cells that you know people will need to change within the worksheet that you want to protect. Then click the "Format Cells" dialog box button in the lower right corner of the "Font" group on the "Home" tab in the Ribbon to open the "Format Cells" dialog box. In the "Format Cells" dialog box, click the "Protection" tab and then click the checkbox in front of the word "Locked" to de-select the locking of those cells. Then click "OK" to close the "Format Cells" dialog box. Now you are ready to apply worksheet protection.

Alternately, you also have a tool that allows you to create specified ranges of cells into which users can perform data entry within a protected worksheet, if they have the password that you specify. If there are ranges of cells which you wish some users could access, but not most general users, then you can specify an editable range of cells in the worksheet and then password protect it. Then people who wish to perform data entry within the range (once worksheet protection has been applied) will be prompted to enter the password in order to make the selected cell range editable. This varies from the "unlocking" of the worksheet cells in that these cells will be read-only for users who lack the password to edit them, while "unlocking" a cell makes it editable by all users.

To apply selective cell access, select the range of cells in the worksheet to which you wish to allow selective access. Then click the "Allow Users to Edit Ranges" button in the "Changes" group on the "Review" tab in the Ribbon. This will launch the "Allow Users to Edit Ranges" dialog box. Here, you can click the "New…" button at the right side of the dialog box to launch the "New Range" dialog box. In the "New Range" dialog box, enter a name for the cell range into the "Title:" text box. The cells that you selected will be shown in the "Refers to cells:" text box. In the "Range password:" text box, type the password that you want to use to allow users to edit this range. Then click "OK." You will have to re-type the password again in the next dialog box, and then click "OK" to confirm the password. Like all passwords, this needs to be written down in a secure location, because if you forget it you will not be able to selectively modify the range while worksheet protection is enforced.

You will now see the defined range in the "Allow Users to Edit Ranges" dialog box. Note that you can also select a range once you have created it, and click the "Modify…" button at the right to change the cell range defined by the title, the title itself, or the password for editing access in the "Modify Range" dialog box. You can also select a range to delete in the "Allow Users to Edit Ranges" dialog box and then click the "Delete" button at the right side of the "Allow Users to Edit Ranges" dialog box to delete the definition of the editable range if no longer needed.

28.2- Worksheet Protection:

When you protect a worksheet, you prevent accidental or malicious changes to the worksheet. Once worksheet protection is in place, no one can change the "locked" cells until the worksheet protection is removed. "Locked" cells can be changed freely as long as the worksheet isn't protected, which is why you must first "unlock" the cells that you want to change ***before*** you apply worksheet protection.

In Excel, you have many options, displayed as several checkboxes, that you can check to allow users to perform. There is quite a bit of control over exactly what users can and can't do. Optionally, you can also type a password that is required to be entered before the worksheet can be "unprotected." This option is a bit risky, because if you forget the password, you will not be able to remove the worksheet protection.

To apply worksheet protection, select the worksheet to protect in the workbook and then click the "Protect Sheet" button in the "Changes" group on the "Review" tab in the Ribbon. This will launch the "Protect Sheet" dialog box. Here you check the checkboxes for actions you want users to be able to accomplish. You can enter a password that allows you to unprotect the worksheet, if desired, by typing it into the Password to unprotect sheet:" text box. You will need to then re-enter the password in a confirmation dialog box that appears. Then click "OK" when you are finished.

When you unprotect a worksheet you are removing your specified worksheet security. To remove worksheet protection from a worksheet, click the "Unprotect Sheet" button in the "Changes" group on the "Review" tab in the Ribbon. If there is an associated password, it will prompt you to enter it into the "Unprotect Sheet" dialog box, and then click "OK." After doing that, the worksheet will be unprotected.

28.3- Workbook Protection:

When you protect a workbook, you prevent changes to the workbook's structure and its windows. You can prevent people from changing the size or shape of the workbook window. With the structural protection, people will not be allowed to add, delete, hide or move worksheets within the workbook.

You can protect the "Windows" and "Structure" in a workbook. You can also provide an optional password that is required to be entered before the workbook can be unprotected. This option is also a bit risky, because if you forget the password you will not be able to remove the workbook protection.

To apply workbook security, click the "Protect Workbook" button in the "Changes" group on the "Review" tab in the Ribbon. This will launch the "Protect Workbook" dialog box. Check the boxes for the objects that you want to protect, and enter a password, if desired. If you enter a password, you will need to re-enter it into the confirmation dialog box that appears, and then click the "OK" button. Then click "OK" when you are finished.

To remove workbook protection, click the "Unprotect Workbook" button in the "Changes" group on the "Review" tab in the Ribbon. If there is an associated password, it will prompt you to enter it into the "Unprotect Workbook" dialog box and then click "OK."

28.4- Password Protecting Excel Files:

When you password protect an Excel file, you prevent people from opening or modifying the file without knowing the password required to accomplish the specified operation. Once password protection is enabled on a file, no one can open or modify the file without supplying the appropriate passwords for both tasks.

Password protecting files poses a risk, because if you forget the password to either open or modify the files, you will not be able to open or modify the files in the future.

To apply password protection to an Excel file, open the file to which you want to apply password protection. Then if using Excel 2007, click the Microsoft Office button, and click the "Save As" command. If using Excel 2010, click the "File" tab in the Ribbon, and then click the "Save As" command.

In either version, this will open the "Save As" dialog box. Select the "Tools" button in the lower right corner of the "Save As" dialog box and then select the "General Options…" command to view the "General Options" dialog box. Here you can set the password protection on the file before saving it.

In the "File sharing" section of the "General Options" dialog box, enter any passwords that you want to apply to this file. You can set passwords to both "open" and/or "modify" the file. You can then click "OK" to set the passwords. You will need to reconfirm these passwords by retyping them in the confirmation dialog box that appears, and then click "OK" to return to the "Save As" dialog box where you will proceed to save the file, as normal.

Now in the future when a user tries to open the file they will need to enter the password to open the file that you specified. Once opened and viewable, if they try to make changes, they will then be prompted to enter the password to modify the file that you specified.

To remove the password protection, you will need to know the passwords for both opening and modifying the file, if any. Then, open the file (you'll need to supply the password to do this), and then open the "Save As" dialog box again. In the "Save As" dialog box, click the "Tools" button and then select the "General Options…" command in order to open the "General Options" dialog box again. This time, delete both passwords from the "File sharing" section of the "General Options" dialog box. Then just click "OK." You will then need to re-save the file as usual to remove the password protection.

ACTIONS-
Security Features

UNLOCKING WORKSHEET CELLS:

1. Select the cells in the worksheet which you want to unlock for future access.
2. Click the "Format Cells" dialog box button in the lower right corner of the "Font" group on the "Home" tab in the Ribbon to open the "Format Cells" dialog box.
3. Click the "Protection" tab.
4. Deselect the "Locked" check box.
5. Click "OK" to save your changes.

SPECIFYING EDITABLE RANGES:

1. Select the cells in the worksheet which you want to specify as an editable range.
2. Click the "Allow Users to Edit Ranges" button in the "Changes" group on the "Review" tab in the Ribbon.
3. In the "Allow Users to Edit Ranges" dialog box, click the "New…" button.
4. In the "New Range" dialog box, enter a name for the range into the "Title:" text box.
5. If needed, you can click the "Collapse Dialog" button at the right end of the "Refers to cells:" dialog box to collapse the dialog box down to select the cells for the range. However, if you selected the cells in Step 1, then you shouldn't need to perform this step.
6. In the "Range password:" text box, type in the password that you want to use to allow editing of the selected cells after worksheet protection is applied.
7. Click "OK."
8. In the "Confirm Password" dialog box, re-type the password to confirm it.
9. Click "OK."

MODIFYING EDITABLE RANGES:

1. Open the worksheet in which you want to modify an editable range.
2. Click the "Allow Users to Edit Ranges" button in the "Changes" group on the "Review" tab in the Ribbon.
3. In the "Allow Users to Edit Ranges" dialog box, select the name of the range to modify.
4. Click the "Modify…" button.
5. Make any changes to the selected range that you desire.
6. Click "OK."

DELETING EDITABLE RANGES:

1. Open the worksheet in which you want to delete an editable range.
2. Click the "Allow Users to Edit Ranges" button in the "Changes" group on the "Review" tab in the Ribbon.
3. In the "Allow Users to Edit Ranges" dialog box, select the name of the range to delete.
4. Click the "Delete" button.
5. Click "OK."

ACTIONS-
Security Features

PROTECTING A WORKSHEET:

1. Open the worksheet to protect and click the “Protect Sheet” button in the “Changes” group on the “Review” tab in the Ribbon.
2. Check any activities that you want to allow your users to perform.
3. Enter a password that you will need to unprotect the worksheet, if you like.
4. Click “OK.”
5. If you entered a password, you will need to re-enter the password in a separate confirmation dialog box and then click “OK” when finished.

UNPROTECTING A WORKSHEET:

1. Open the worksheet to which you applied protection and click the “Unprotect Sheet” button in the “Changes” group on the “Review” tab in the Ribbon.
2. Type the password and click “OK,” if required.

PROTECTING A WORKBOOK:

1. Open the workbook to protect and click the “Protect Workbook” button in the “Changes” group on the “Review” tab in the Ribbon.
2. Select which properties of the workbook to protect by checking the checkboxes for “Structure” and/or “Windows.”
3. Enter a password for the workbook that you will need to enter to unprotect the workbook, if you like.
4. Click “OK.”
5. Re-enter the password, if you opted to enter a password, and click “OK.”

UNPROTECTING A WORKBOOK:

1. Open the workbook from which you wish to remove protection and click the “Unprotect Workbook” button in the “Changes” group on the “Review” tab in the Ribbon.
2. Type the password, if required, and click “OK.”

PASSWORD PROTECTING EXCEL FILES:

1. Open the file to which you want to apply password protection.
2. Then if using Excel 2007, click the Microsoft Office button, and click the “Save As” command. If using Excel 2010, click the “File” tab in the Ribbon, and then click the “Save As” command.
3. In either version, select the “Tools” button in the lower right corner of the “Save As” dialog box and then select the “General Options…” command to view the “General Options” dialog box.
4. In the “File sharing” section of the “General Options” dialog box, enter any passwords that you want to apply to this file. You can set passwords to both “open” and/or “modify” the file.
5. You can then click “OK” to set the passwords.
6. You will need to reconfirm these passwords by retyping them in the confirmation dialog box that appears, and then click “OK” to return to the “Save As” dialog box where you will proceed to save the file, as normal.

REMOVING PASSWORD PROTECTION FROM EXCEL FILES:

1. Open the file (you'll need to supply the password to do this), and then open the "Save As" dialog box again.
2. In the "Save As" dialog box, click the "Tools" button again and then select the "General Options…" command in order to open the "General Options" dialog box.
3. This time, delete both passwords from the "File sharing" section of the "General Options" dialog box.
4. Then click the "OK" button.
5. You will then need to re-save the file as usual to remove the password protection.

EXERCISES-
SECURITY FEATURES

Purpose:

1. To be able to lock and unlock cells and worksheets.

Exercises:

1. Open up the “Adv- Sample” workbook and select “Sheet1”.
2. Select columns A through H.
3. Click the “Format Cells” dialog box button in the lower right corner of the “Font” group on the “Home” tab in the Ribbon.
4. Click the “Protection” tab.
5. Remove the check from the “Locked” check box.
6. Click “OK.”
7. Click the “Protect Sheet” button in the “Changes” group on the “Review” tab in the Ribbon.
8. Click “OK.”
9. Select cell K2, and try to type “.10” in it.
10. Click “OK” on the message that appears telling you that the cell is protected.
11. Select cell C2, and type “Detroit.”
12. Click the “Unprotect Sheet” button in the “Changes” group on the “Review” tab in the Ribbon.
13. Click the “Save” button in the Quick Access Toolbar to save your changes.
14. Click the “X” button in the upper right corner of the application window to exit.

CHAPTER 29- MAKING MACROS

29.1- RECORDING MACROS

29.2- RUNNING AND DELETING RECORDED MACROS

29.3- THE PERSONAL MACRO WORKBOOK

29.1- Recording Macros:

Macros are small programs that record your keystrokes as you perform a task, and then save the actions you performed as a Visual Basic Module- a type of program file. When you run the macro later, it will repeat your keystrokes, thus repeating your actions. This is why they are great for automating repetitive tasks. For example, pretend that you wanted to place your name and your company's information in the upper left cell of a worksheet. You could use a macro to record your keystrokes as you create it once, and then run the macro in the future. It would repeat the exact same keystrokes that you entered, effectively repeating the process instantaneously.

While you can see advanced options for creating macros on the "Developer" tab in the Ribbon, if it is enabled, you can also use the "Macros" group on the "View" tab in the Ribbon to record and playback basic macros that you record. In this lesson, we will examine how to record a basic macro using the commands found within this group in the Ribbon.

To begin to record a macro, you can click the "Macros" button in the "Macros" group on the "View" tab in the Ribbon. From the drop-down menu that appears, select the "Record Macro…" command. This will open the "Record Macro" dialog box. In the "Record New Macro" dialog box, enter a name for your new macro in the "Macro name" text box. Note that macro names cannot contain spaces!

Next, select the name of the workbook to which you want to attach the macro by selecting its name from the "Store macro in:" drop-down. If you do not change it, it will default to saving the macro into the current workbook. This is important only because a macro can only be run if it is attached to an open workbook, or stored in the "Personal Macro Workbook," which we will discuss later. You can also create a custom keyboard shortcut to use in conjunction with the "Ctrl" key by typing the desired shortcut key letter into the text box next to the "Ctrl +." If you decide to do this, make sure you don't overwrite an existing shortcut! For example the shortcut character of "p" would be a bad choice as "Ctrl + P," is already a keyboard shortcut for the "Print" command. If you aren't familiar with the keyboard shortcuts, it may be better if you don't assign one. When you are ready to start recording your actions, click "OK."

While recording your macro, you cannot use your mouse very much and you should minimize your mouse movements during the recording of the macro. Instead, try to use the keyboard as much as possible. Once you have finished recording your macro, click the "Macros" button in the "Macros" group on the "View" tab in the Ribbon. Select the "Stop Recording" command in order to stop recording the macro.

Unlike macros in the other Microsoft Office programs, in Excel the types of cell references that you make while recording a macro can be adjusted. For example, assume that when you began recording your macro, your "active cell" was cell A1. From there, you clicked into cell D1. When Excel records you doing that, it can either record that action as a relative reference or an absolute reference. That action, if recorded using relative references, would make the "active cell" move four cells to the right of whatever cell it was in when you started to playback the macro. If you recorded it in absolute terms, it would always move to cell D1 from wherever you started the macro. You can adjust the types of referencing used when recording a macro by clicking the "Macros" button in the "Macros" group and then selecting the "Use Relative References" command. By default, Excel macros will use absolute references. You can click this button to switch to relative cell referencing during your macro. You can click it again to switch back to absolute referencing when needed during or after recording.

Also remember that if you want to save a workbook that contains macros in Excel 2007 or later, you need to select the "Excel Macro-Enabled Workbook" choice from the "File type:" drop-down in the "Save As" dialog box.

29.2- Running and Deleting Macros:

To run a recorded macro, you can click the "Macros" button in the "Macros" group on the "View" tab in the Ribbon. If there are macros that are available to run, then you can choose the "View Macros" command from the button's drop-down. If you do not see this command, then you may not have any recorded macros available for use. Otherwise, once you select this command, you will see the "Macro" dialog box appear. You use this dialog box to manage your macros.

A listing of the macros that are available will appear in the large white list box shown within the "Macro" dialog box. To run a macro shown in this list, click on its name it select it. Then click the "Run" button to run the selected macro.

You can also delete macros that you no longer want or need using this dialog box. To delete a macro, select the name of the macro from the macro list and then click the "Delete" button. Click the "Yes" button in the confirmation message box that appears in order to delete the selected macro. Once you have finished using the "Macro" dialog box, click the "Close" button to close it.

29.3- The Personal Macro Workbook:

When you record macros, they are attached to the workbook in which you create them, by default. When a workbook that contains macros is open, the macros attached to it are available for use by all open workbooks. However, once you close the workbook to which the macros are attached, they are no longer available for use by other workbooks.

To remedy this, Excel provides a "Personal Macro Workbook" as a place to which you can store macros which you want to be universally accessible by all open workbooks on your computer. The "Personal Macro Workbook" is a workbook that opens every time you open Excel. However, it is a hidden workbook, so you do not normally see it and very many people aren't even aware that it exists. However, you can store your macros here to make them available to all open workbooks on your PC. Since the "Personal Macro Workbook" is always open and always hidden, it is always usable by all other open workbooks at that computer.

To save a macro into the "Personal Macro Workbook," you need to select "Personal Macro Workbook" from the "Store macro in:" drop-down that is available in the "Record Macro" dialog box. Then the macro that you record will be stored in that workbook.

Once you have stored macros to the "Personal Macro Workbook," you will need to unhide the workbook to edit or delete the macros in it. To hide and unhide the "Personal Macro Workbook," you must click the "Hide" or "Unhide" buttons in the "Window" group on the "View" tab in the Ribbon.

If you click the "Unhide" button after you have saved at least one macro to the "Personal Macro Workbook," you'll be presented with the "Unhide" dialog box. To unhide the "Personal Macro Workbook," select the "PERSONAL" file, and then click "OK." You will then see the "Personal Macro Workbook" appear. You can then edit or delete any macros saved to that workbook by using the "Macro" dialog box.

When you are finished, make sure that you still have the "Personal Macro Workbook" displayed, and then click the "Hide" button in the "Window" group on the "View" tab in the Ribbon to hide the "Personal Macro Workbook" again. Always remember to hide the "Personal Macro Workbook" when you are done editing or deleting its macros to ensure that the workbook doesn't get deleted accidentally.

ACTIONS-
MAKING MACROS

RECORDING MACROS:

1. Click the “Macros” button in the “Macros” group on the “View” tab in the Ribbon. From the drop-down menu that appears, select the “Record Macro…” command to open the “Record Macro” dialog box.
2. In the “Record New Macro” dialog box, enter a name for your new macro in the “Macro name” text box.
3. You can also create a custom keyboard shortcut to use in conjunction with the “Ctrl” key by typing the desired shortcut key letter into the text box next to the “Ctrl +.” If you decide to do this, make sure you don’t overwrite an existing shortcut!
4. Next, select the name of the workbook to which you would like to attach the macro from the “Store macro in:” drop-down.
5. Click the “OK” button to begin recording your macro.
6. Once you have finished recording your macro, click the “Macros” button in the “Macros” group on the “View” tab in the Ribbon. Select the “Stop Recording” command in order to stop recording the macro.
7. Also, if necessary, you can choose the “Pause Recording” command from the button’s drop-down menu to pause the macro while recording. You can click the “Macro” button, and then select the Resume Recorder” command in order to resume recording the macro when you are ready to continue. You can also adjust the types of referencing used when recording a macro by clicking the “Macros” button in the “Macros” group and then selecting the “Use Relative References” command. By default, Excel macros will use absolute references. You can click this button to switch to relative cell referencing during your macro. You can click it again to switch back to absolute referencing when needed during or after recording.

RUNNING AND DELETING MACROS:

1. To run a recorded macro, you can click the “Macros” button in the “Macros” group on the “View” tab in the Ribbon. Once you select this command, you will see the “Macro” dialog box appear.
2. A listing of the macros that are available will appear in the large white list box shown within the “Macro” dialog box. To run a macro shown in this list, click on its name it select it. Then click the “Run” button to run the selected macro.
3. You can also delete macros that you no longer want or need using this dialog box. To delete a macro, select the name of the macro from the macro list and then click the “Delete” button.
4. Click the “Yes” button in the confirmation message box that appears in order to delete the macro.
5. Once you have finished using the “Macro” dialog box, click the “Close” button to close it.

UNHIDING AND HIDING THE PERSONAL MACRO WORKBOOK:

1. If you click the “Unhide” button in the “Window” group on the “View” tab in the Ribbon after you have saved at least one macro to the “Personal Macro Workbook,” you will open the “Unhide” dialog box.
2. To unhide the “Personal Macro Workbook,” select the “PERSONAL” file, and then click “OK.”
3. You will then see the “Personal Macro Workbook” appear. You can then edit or delete any macros saved to that workbook by using the “Macro” dialog box.
4. When you are finished, make sure that you still have the “Personal Macro Workbook” displayed, and then click the “Hide” button in the “Window” group on the “View” tab in the Ribbon to hide the “Personal Macro Workbook” again. Always remember to hide the “Personal Macro Workbook” when you are done editing or deleting its macros to ensure that the workbook doesn’t get deleted accidentally.

EXERCISES-
MAKING MACROS

Purpose:

1. To be able to create and run macros.

Exercises:

1. Open up the “Adv- Sample” workbook and select “Sheet2.”
2. Click the “Macros” button in the “Macros” group on the “View” tab in the Ribbon and then select the “Record Macro…” command from the button’s drop-down menu.
3. For the “Macro name:” type “SwitchScenario.”
4. Select “This Workbook” from the “Store macro in:” drop-down.
5. Click “OK.”
6. Click the “What-If Analysis” button on the “Data Tools” group on the “Data” tab in the Ribbon and then select “Scenario Manager…” from the button’s drop-down menu.
7. In the “Scenario Manager” dialog box that appears, select the “7% Commission Rate” scenario, and then click the “Show” button.
8. Click the “Close” button.
9. Click the “Macros” button in the “Macros” group on the “View” tab in the Ribbon and then select the “Stop Recording” command from the button’s drop-down menu.
10. Click the “Macros” button in the “Macros” group on the “View” tab in the Ribbon and then select the “Record Macro…” command from the button’s drop-down menu.
11. For the “Macro name:” type “SwitchBack.”
12. Select “This Workbook” from the “Store macro in:” drop-down.
13. Click “OK.”
14. Click the “What-If Analysis” button on the “Data Tools” group on the “Data” tab in the Ribbon and then select “Scenario Manager…” from the button’s drop-down menu.
15. In the “Scenario Manager” dialog box that appears, select the “Current Commission Rates” scenario, and click the “Show” button.
16. Click the “Close” button.
17. Click the “Macros” button in the “Macros” group on the “View” tab in the Ribbon and then select the “Stop Recording” command from the button’s drop-down menu.
18. Click the “Macros” button in the “Macros” group on the “View” tab in the Ribbon and then select the “View Macros” command from the button’s drop-down menu.
19. In the “Macro” dialog box, select the “SwitchScenario” macro, and click the “Run” button.
20. Click the “Macros” button in the “Macros” group on the “View” tab in the Ribbon and then select the “View Macros” command from the button’s drop-down menu.
21. In the “Macro” dialog box, select the “SwitchBack” macro, and click the “Run” button.
22. You do not need to save your workbook, however if you wish to save a workbook that contains macros, you should select “Excel Macro-Enabled Workbook” in the “Save As” dialog box.
23. Click the “X” button in the upper right corner of the application window to exit.

Glossary of Functions
Index

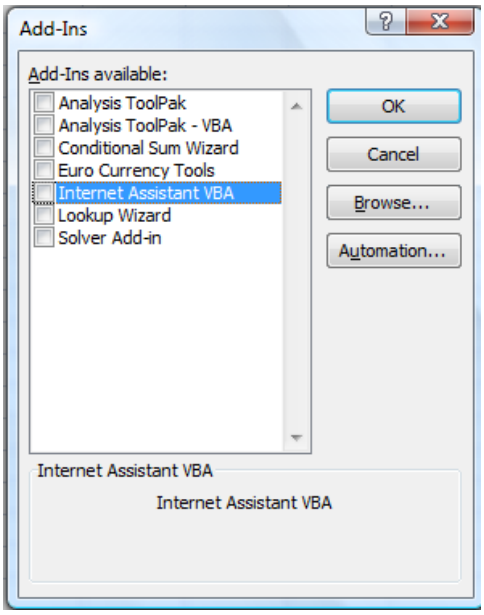
Glossary of Functions
Index

***Note on Functions:**

Please note that not all functions are available by default in Excel. Many of these functions must be loaded into Excel using the "Add-ins" available. To load an add-in click the Microsoft Office button and then click the "Excel Options" button. In the "Excel Options" window, select the "Add-Ins" category to the left. Use the "Manage:" drop-down at the bottom of this window to select "Excel Add-Ins." Then click the "Go…" button to open the "Add-Ins" dialog box.

Here you can check which add-ins you wish to load by placing a check next to the name of the add-in set you want. For example, if you wanted to use many of the financial functions, you will need to load the "Analysis ToolPak" add-in. Once you have selected the desired add-ins to load, click "OK." That will then load the selected add-ins and their related functions. You only need to load the functions once per computer.

Note that if you try to type the name of one of the functions that are loaded from an add-in set that you have not yet loaded, you will probably receive the "#NAME?" error message. If this is the case, make sure that you have typed the name of the function correctly (no spaces) and that you have the correct add-ins installed.



Glossary of Functions

DAVERAGE:

Description: Using the DAVERAGE function will return the average of selected database entries. It will find the average of a field in a table or database, based on specified criteria.

General Syntax: **=DAVERAGE(database,field,criteria)**

Arguments:
database is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1).
criteria is the cell reference of the criteria range used to filter the database or table.

DCOUNT:

Description: Using the DCOUNT function will return the count of cells that contain number values for selected database entries. It will find the count of cells that contain numbers in a field of a table or database, based on specified criteria.

General Syntax: **=DCOUNT(database,*field*,criteria)**

Arguments:
database is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1). This is an optional argument.
criteria is the cell reference of the criteria range used to filter the database or table.

DCOUNTA:

Description: Using the DCOUNTA function will return the count of cells that are not blank for selected database entries. It will find the count of cells that do not contain blanks in a field of a table or database, based on specified criteria.

General Syntax: **=DCOUNTA(database,*field*,criteria)**

Arguments:
database is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1). This is an optional argument.
criteria is the cell reference of the criteria range used to filter the database or table.

DGET:

Description: Using the DGET function will return the unique value found within a field of a table or database, based on specified criteria. If no record matches the criteria, this function will return the "#VALUE!" error. If more than one record matches the criteria, it will return the "#NUM!" error.

General Syntax: **=DGET(database,field,criteria)**

Arguments:
database is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1).
criteria is the cell reference of the criteria range used to filter the database or table.

Glossary of Functions

DMAX:

Description: Using the DMAX function will return the largest number found within a field of a table or database, based on specified criteria.

General Syntax: **=DMAX(database,field,criteria)**

Arguments: **database** is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1).
criteria is the cell reference of the criteria range used to filter the database or table.

DMIN:

Description: Using the DMIN function will return the smallest number found within a field of a table or database, based on specified criteria.

General Syntax: **=DMIN(database,field,criteria)**

Arguments: **database** is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1).
criteria is the cell reference of the criteria range used to filter the database or table.

DPRODUCT:

Description: Using the DPRODUCT function will multiply the values found within a field of a table or database, based on specified criteria.

General Syntax: **=DPRODUCT(database,field,criteria)**

Arguments: **database** is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1).
criteria is the cell reference of the criteria range used to filter the database or table.

DSTDEV:

Description: Using the DSTDEV function will estimate the standard deviation of values found within a field of a table or database, based on specified criteria.

General Syntax: **=DSTDEV(database,field,criteria)**

Arguments: **database** is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1).
criteria is the cell reference of the criteria range used to filter the database or table.

DSTDEVP:

Description: Using the DSTDEVP function will estimate the standard deviation of a population based on the entire population of values found within a field of a table or database using the records that match the specified criteria.

General Syntax: **=DSTDEVP(database,field,criteria)**

Arguments:
database is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1).
criteria is the cell reference of the criteria range used to filter the database or table.

DSUM:

Description: Using the DSUM function will add the values found within a field of a table or database, based on specified criteria.

General Syntax: **=DSUM(database,field,criteria)**

Arguments:
database is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1).
criteria is the cell reference of the criteria range used to filter the database or table.

DVAR:

Description: Using the DVAR function will estimate the variance of a population found within a field of a table or database, based on specified criteria.

General Syntax: **=DVAR(database,field,criteria)**

Arguments:
database is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1).
criteria is the cell reference of the criteria range used to filter the database or table.

DVARP:

Description: Using the DVARP function will estimate the variance of a population based on the entire population of values found within a field of a table or database using the records that match the specified criteria.

General Syntax: **=DVARP(database,field,criteria)**

Arguments:
database is the reference to the cell range or named range of the database or table which you want to use in the function.
field is the name of the field in the database upon which you perform the function. If this is entered as text, place it inside of double-quotes (" "). You may also reference the field by its column number within the database (with the left-most column in the table or database being column number 1).
criteria is the cell reference of the criteria range used to filter the database or table.

Glossary of Functions

DATE:

Description: Using the DATE function returns the serial number that represents a specific date. If the cell into which you enter this formula is formatted as "General" *before* creating the formula the output will still be displayed as a date.

General Syntax: **=DATE(year,month,day)**

Arguments: **year** is the reference to the cell or value (of 1 to 4 digits) that represents the year.
month is the reference to the cell or value that represents the month.
day is the cell reference or value that represents the day.

DATEVALUE:

Description: Use the DATEVALUE function to return the serial number of a date that is entered as a text string, enclosed in double-quotes (" ").

General Syntax: **=DATEVALUE(date_as_text)**

Arguments: **date_as_text** is the date text string, enclosed in double-quotes, for which you wish to find the serial number. E.g. "8/2/1975."

DAY:

Description: Use the DAY function to return the "day" number (from 1 to 31) of a date value that is entered by hand or referenced in a cell. Will not work on dates entered as "text" values.

General Syntax: **=DAY(date)**

Arguments: **date** is the date value from which you wish to extract the "day" number.

DAYS360:

Description: Use the DAYS360 function to return the number of days between two date values, based on a 360 day year (twelve 30-day months). Used by some accounting calculations to compute payments based on a 12-month 30-day accounting system.

General Syntax: **=DAYS360(start_date,end_date,***method***)**

Arguments: **start_date** is the first date in the date range.
end_date is the last date in the date range.
method is an optional argument that can be either "TRUE" or "FALSE." If omitted, "FALSE" is assumed. Use "TRUE" for the European method of calculation where the 31st is always equal to the 30th of the same month.

EDATE:

Description: Using the EDATE function returns the date value that is the indicated number of months before or after the date value specified. Often used to calculate maturity or due dates that fall on the same date as the month of issue.

General Syntax: **=EDATE(date,months)**

Arguments: **date** is the date value referenced by the formula.
months is the number of months before or after the date value specified. Can be either positive or negative.

EOMONTH:

Description: Using the EOMONTH function returns the date value of the end of the month that is the indicated number of months before or after the date value specified. Often used to calculate maturity or due dates that fall on the last day of the month.

General Syntax: **=EOMONTH(date,months)**

Arguments: **date** is the date value referenced by the formula.
months is the number of months before or after the date value specified. Can be either positive or negative.

HOUR:

Description: Using the HOUR function returns the "hour" number (from 0 to 23) of a time value. The "hour" ranges from 0 (12:00 AM) to 23 (11:00 PM).

General Syntax: **=HOUR(time)**

Arguments: **time** is the time value referenced by the formula, which contains the hour you want to find.

MINUTE:

Description: Using the MINUTE function returns the "minute" number (from 0 to 59) of a time value.

General Syntax: **=MINUTE(time)**

Arguments: **time** is the time value referenced by the formula, which contains the minute you want to find.

MONTH:

Description: Using the MONTH function returns the "month" number (from 1 to 12) of a date value. The "month" ranges from 1 (January) to 12 (December).

General Syntax: **=MONTH(date)**

Arguments: **date** is the date value referenced by the formula, which contains the month you want to find.

NETWORKDAYS:

Description: Using the NETWORKDAYS function returns the number of work days between two dates specified by the formula. This function excludes weekends and days identified as holidays.

General Syntax: **=NETWORKDAYS(start_date,end_date**,*holidays***)**

Arguments: **start_date** is the starting date value.
end_date is the ending date value.
holidays is an optional cell range reference to a range of cells that contain a listing of any date values between **start_date** and **end_date** to be excluded from the number of net work days.

NOW:

Description: Using the NOW function returns the date/time value of the current date and time.

General Syntax: **=NOW()**

Arguments: None.

SECOND:

Description: Using the SECOND function returns the "second" number (from 0 to 59) of a time value.

General Syntax: **=SECOND(time)**

Arguments: **time** is the time value referenced by the formula, which contains the second you want to find.

TIME:

Description: Using the TIME function returns the decimal number that represents the time specified. If the cell into which you enter this formula is formatted as "General" *before* entering this function, the result is displayed as a date/time value.

General Syntax: **=TIME(hour,minute,second)**

Arguments: **hour** is the number that represents the "hour" value.
minute is the number that represents the "minute" value.
second is the number that represents the "second" value.

TIMEVALUE:

Description: Using the TIMEVALUE function returns the decimal number of the time specified by a text string (enclosed in double-quotes).

General Syntax: **=TIMEVALUE(time_as_text)**

Arguments: **time_as_text** is the time text string, enclosed in double-quotes, for which you wish to find the serial number. E.g. "11:11 PM."

TODAY:

Description: Using the TODAY function returns the date value of the current date. If the cell into which you enter this formula is formatted as "General" *before* entering this function, the result is displayed as a date value.

General Syntax: **=TODAY()**

Arguments: None.

WEEKDAY:

Description: Using the WEEKDAY function returns the number of the day of the week (from 1 to 7) of a specified date value. The "weekday" number returned ranges from 1 (Sunday) to 7 (Saturday), by default.

General Syntax: **=WEEKDAY(date**,*day_number***)**

Arguments: **date** is the date value referenced by the formula for which you want to find the day of the week.
day_number is a number that determines which integer is used to represent which day. If 1, or omitted, returns 1 to 7 value for Sunday to Saturday. If 2, returns 1 to 7 value for Monday through Sunday. If 3, uses 0 to 6 for Monday through Sunday.

WEEKNUM:

Description: Using the WEEKNUM function returns a number that indicates in which week of the year the date specified in the formula falls.

General Syntax: **=WEEKNUM(date**,*week_number***)**

Arguments: **date** is the date value referenced by the formula.
week_number is a number that determines on which day the week begins. If 1, or omitted, the week begins on a Sunday. If 2, the week begins on a Monday.

WORKDAY:

Description: Using the WORKDAY function returns a date value that is the number of working days before or after the date specified in the formula.

General Syntax: **=WORKDAY(date,days**,*holidays***)**

Arguments: **date** is the date value specified by the formula.
days is the number of non-weekend and non-holiday days before or after the **date** specified. Can be either positive or negative.
holidays is an optional cell range reference to a range of cells that contain a listing of any date values to be excluded.

YEAR:

Description: Using the YEAR function returns the "year" number of a specified date value.

General Syntax: **=YEAR(date)**

Arguments: **date** is the date value referenced by the formula, which contains the year you want to find.

YEARFRAC:

Description: Using the YEARFRAC function returns the fraction of the year represented by the number of whole days between the first and last dates specified in the formula.

General Syntax: **=YEARFRAC(start_date,end_date**,*basis***)**

Arguments: **start_date** is the first date in the date range.
end_date is the last date in the date range.
basis is an optional numeric argument (from 0 to 4) that represents the day count to use. If 0, or omitted, it uses the "US 30/360" day count. If 1, uses "Actual/actual." If 2, uses "Actual/360." If 3, uses "Actual/365." If 4, uses "European 30/360."

BESSELI:

Description: Using the BESSELI function returns the modified Bessel function: the Bessel function evaluated for purely imaginary arguments.

General Syntax: **=BESSELI(x,n)**

Arguments: **x** is the value at which you want to evaluate the formula.
n is the order of the Bessel function as an integer.

Function: The **n**th order modified Bessel function of the variable **x** is: $I_n(x) = (i)^{-n} J_n(ix)$

Glossary of Functions

BESSELJ:

Description: Using the BESSELJ function returns the Bessel function.

General Syntax: **=BESSELJ(x,n)**

Arguments: **x** is the value at which you want to evaluate the formula.
n is the order of the Bessel function as an integer.

Function: The **n**th order Bessel function of the variable **x** is: $J_n(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!\Gamma(n+k+1)}\left(\frac{x}{2}\right)^{n+2k}$

where the Gamma function is: $\Gamma(n+k+1) = \int_0^{\infty} e^{-x} x^{n+k} dx$

BESSELK:

Description: Using the BESSELK function returns the modified Bessel function: the Bessel function evaluated for purely imaginary arguments.

General Syntax: **=BESSELK(x,n)**

Arguments: **x** is the value at which you want to evaluate the formula.
n is the order of the Bessel function as an integer.

Function: The **n**th order modified Bessel function of the variable **x** is:

$$K_n(x) = \frac{p}{2} i^{n+1}[J_n(ix) + iY_n(ix)]$$

where J_n is the J Bessel function and Y_n is the Y Bessel function.

BESSELY:

Description: Using the BESSELY function returns the Bessel function. The Bessel function is also known as the "Weber function" or the "Neumann function."

General Syntax: **=BESSELY(x,n)**

Arguments: **x** is the value at which you want to evaluate the formula.
n is the order of the Bessel function as an integer.

Function: The **n**th order Bessel function of the variable **x** is: $Y_n(x) = \lim_{v \to n} \frac{J_v(x)\cos(v\,\pi) - J_{-v}(x)}{\sin(v\,\pi)}$

where: $\mathrm{ERF}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$

BIN2DEC:

Description: Using the BIN2DEC function converts a binary number to a decimal number.

General Syntax: **=BIN2DEC(number)**

Arguments: **number** is the binary number that you want to convert to a decimal number.

Glossary of Functions

BIN2HEX:

Description: Using the BIN2HEX function converts a binary number to a hexadecimal number.

General Syntax: **=BIN2HEX(number**,*places***)**

Arguments: **number** is the binary number that you want to convert to a hexadecimal number.
places is an optional argument that determines the number of characters to use. If omitted, uses the minimum number of characters necessary. Useful for padding the result with leading zeros.

BIN2OCT:

Description: Using the BIN2OCT function converts a binary number to an octal number.

General Syntax: **=BIN2OCT(number**,*places***)**

Arguments: **number** is the binary number that you want to convert to an octal number.
places is an optional argument that determines the number of characters to use. If omitted, uses the minimum number of characters necessary. Useful for padding the result with leading zeros.

COMPLEX:

Description: Using the COMPLEX function returns a complex number of the form *x* + *yi* or *x* + *yj* for a specified real and imaginary coefficient.

General Syntax: **=COMPLEX(real_num,i_num,***suffix***)**

Arguments: **real_num** is the real coefficient of the complex number.
i_num is the imaginary coefficient of the complex number.
suffix is the suffix for the imaginary component of the complex number. If omitted, is assumed to be "i." You may use either "I" or "j" as the suffix, but not "I" and "J." All functions that accept two or more complex numbers require that all suffixes match.

CONVERT:

Description: Using the CONVERT function converts a number from one system of measurement to another system of measurement.

General Syntax: **=CONVERT(number,from,to)**

Arguments: **number** is the number of units **from** to convert.
from is the measurement unit of which you have **number** amount.
to is the measurement unit to which you wish to convert the **number**.

Specific Syntax: The following is a list of measurement system units that can be used as the **from** and **to** argument values. These must be entered in double-quotes in the function:

	Measurement	Entry	Measurement	Unit	Measurement:	Unit:
Weight and Mass:	Gram	"g"	Slug	"sg"	Pound Mass	"lbm"
	U (atomic mass)	"u"	Ounce Mass	"ozm"		
Distance:	Meter	"m"	Statute Mile	"mi"	Nautical Mile	"Nmi"
	Inch	"in"	Foot	"ft"	Yard	"yd"
	Angstrom	"ang"	Pica (1/72 in.)	"Pica"		
Time:	Year	"yr"	Day	"day"	Hour	"hr"
	Minute	"mn"	Second	"sec"		
Pressure:	Pascal	"Pa"	Atmosphere	"atm"	mm of Mercury	"mmHg"

(cont…)

CONVERT (cont.):

Specific Syntax (cont.): The following is a list of measurement system units that can be used as the **from** and **to** argument values. These must be entered in double-quotes in the function:

	Measurement	Entry	Measurement	Unit	Measurement:	Unit:
Force:	Newton	"N"	Dyne	"dyn"	Pound force	"lbf"
Energy:	Joule	"J"	Erg	"e"	Thermodynamic Calorie	"C"
	IT Calorie	"cal"	Electron volt	"eV"	Horsepower-hour	"HPh"
	Watt-hour	"Wh"	Foot-pound	"flb"	BTU	"BTU"
Power:	Horsepower	"HP"	Watt	"W"		
Magnetism:	Tesla	"T"	Gauss	"ga"		
Temperature:	Degree Celsius	"C"	Degree Fahrenheit	"F"	Degree Kelvin	"K"
Liquid Measure:	Teaspoon	"tsp"	Tablespoon	"tbs"	Fluid ounce	"oz"
	Cup	"cup"	U.S. Pint	"pt"	U.K. Pint	"uk_pt"
	Quart	"qt"	Gallon	"gal"	Liter	"l"

The following abbreviated unit prefixes can be added before any metric unit used:

Prefix	Abbreviation	Prefix	Abbreviation	Prefix:	Abbreviation:
exa	"E"	peta	"P"	tera	"T"
giga	"G"	mega	"M"	kilo	"k"
hecto	"h"	dekao	"e"	deci	"d"
centi	"c"	milli	"m"	micro	"u"
nano	"n"	pico	"p"	femto	"f"
atto	"a"				

DEC2BIN:

Description: Using the DEC2BIN function converts a decimal number to a binary number.

General Syntax: **=DEC2BIN(number**,*places***)**

Arguments: **number** is the decimal number that you want to convert to a binary number.
places is an optional argument that determines the number of characters to use. If omitted, uses the minimum number of characters necessary. Useful for padding the result with leading zeros.

DEC2HEX:

Description: Using the DEC2HEX function converts a decimal number to a hexadecimal number.

General Syntax: **=DEC2HEX(number**,*places***)**

Arguments: **number** is the decimal number that you want to convert to a hexadecimal number.
places is an optional argument that determines the number of characters to use. If omitted, uses the minimum number of characters necessary. Useful for padding the result with leading zeros.

DEC2OCT:

Description: Using the DEC2OCT function converts a decimal number to an octal number.

General Syntax: **=DEC2OCT(number**,*places***)**

Arguments: **number** is the decimal number that you want to convert to an octal number.
places is an optional argument that determines the number of characters to use. If omitted, uses the minimum number of characters necessary. Useful for padding the result with leading zeros.

Glossary of Functions

DELTA:

Description: Using the DELTA function tests whether or not two number values are equal. The function returns a "1" if they are equal or a "0" if they are not equal.

General Syntax: **=DELTA(number1,***number2***)**

Arguments: **number1** is the first number to compare.
number2 is the second number to compare. If omitted, is assumed to be zero.

ERF:

Description: Using the ERF function returns the error function integrated between the upper limit and lower limit specified.

General Syntax: **=ERF(lower,***upper***)**

Arguments: **lower** is the lower limit for integrating ERF.
upper is the upper limit for integrating ERF. If omitted, ERF integrates between zero and the lower limit.

Function:

$$\mathrm{ERF}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$$

$$\mathrm{ERF}(a,b) = \frac{2}{\sqrt{\pi}} \int_a^b e^{-t^2} dt = \mathrm{ERF}(b) - \mathrm{ERF}(a)$$

ERFC:

Description: Using the ERFC function returns the complimentary ERF function integrated between the lower bound specified and infinity.

General Syntax: **=ERFC(lower)**

Arguments: **lower** is the lower bound for integrating ERF.

Function:

$$\mathrm{ERFC}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt = 1 - \mathrm{ERF}(x)$$

GESTEP:

Description: Using the GESTEP function returns a "1" if the number used is greater than the step, otherwise returns a zero.

General Syntax: **=GESTEP(number**,*step***)**

Arguments: **number** is the number that you want to test against the step.
step is the threshold value. If omitted, uses "0."

HEX2BIN:

Description: Using the HEX2BIN function converts a hexadecimal number to a binary number.

General Syntax: **=HEX2BIN(number**,*places***)**

Arguments: **number** is the hexadecimal number that you want to convert to binary.
places is an optional argument that determines the number of characters to use. If omitted, uses the minimum number of characters necessary. Useful for padding the result with leading zeros.

HEX2DEC:

Description: Using the HEX2DEC function converts a hexadecimal number to a decimal number.

General Syntax: **=HEX2DEC(number)**

Arguments: **number** is the hexadecimal number that you want to convert to a decimal number.

HEX2OCT:

Description: Using the HEX2OCT function converts a hexadecimal number to an octal number.

General Syntax: **=HEX2OCT(number**,*places***)**

Arguments: **number** is the hexadecimal number that you want to convert to an octal number.
places is an optional argument that determines the number of characters to use. If omitted, uses the minimum number of characters necessary. Useful for padding the result with leading zeros.

IMABS:

Description: Using the IMABS function returns the absolute value, or "modulus," of a complex number in the *x + yi* or *x + yj* format.

General Syntax: **=IMABS(inumber)**

Arguments: **inumber** is the complex number for which you want the absolute value returned.

Formula: The absolute value of a complex number is represented by the function:

$$\mathrm{IMABS}(z) = |z| = \sqrt{x^2 + y^2}$$

where *z = x + yi*.

IMAGINARY:

Description: Using the IMAGINARY function returns the imaginary coefficient of a complex number in either the *x + yi* or *x + yj* format.

General Syntax: **=IMAGINARY(inumber)**

Arguments: **inumber** is the complex number for which you want the imaginary coefficient.

IMARGUMENT:

Description: Using the IMARGUMENT function returns the theta argument (θ), an angle expressed in radians, such that:

$$x + yi = |x + yi| \times e^{\theta} = |x + yi|(\cos\theta + i\sin\theta)$$

General Syntax: **=IMARGUMENT(inumber)**

Arguments: **inumber** is the complex number for which you want the argument theta (θ).

Function: $\mathrm{IMARGUMENT}(z) = \tan^{-1}\left(\frac{y}{x}\right) = \theta$ where $\theta \in]-\pi;\pi]$ and *z=x + yi*.

Glossary of Functions

IMCONJUGATE:

Description: Using the IMCONJUGATE function returns complex conjugate of a complex number in the *x + yi* or *x + yj* format, entered as a text value.

General Syntax: **=IMCONJUGATE(inumber)**

Arguments: **inumber** is the complex number for which you want the conjugate.

Formula: The conjugate of a complex number is: $\mathrm{IMCONJUGATE}(x+yi) = \bar{z} = (x-yi)$

IMCOS:

Description: Using the IMCOS function returns the cosine of a complex number in the *x + yi* or *x + yj* format, entered as a text value.

General Syntax: **=IMCOS(inumber)**

Arguments: **inumber** is the complex number for which you want the cosine.

Formula: The cosine of a complex number is: $\cos(x+yi) = \cos(x)\cosh(y) - \sin(x)\sinh(y)i$

IMDIV:

Description: Using the IMDIV function returns the quotient of two complex numbers in the *x + yi* or *x + yj* format, entered as a text value.

General Syntax: **=IMDIV(inumber1,inumber2)**

Arguments: **inumber1** is the complex number that is the numerator, or dividend.
inumber2 is the complex number that is the denominator, or divisor

Formula: The quotient of two complex numbers is: $\mathrm{IMDIV}(z_1, z_2) = \frac{(a+bi)}{(c+di)} = \frac{(ac+bd)+(bc-ad)i}{c^2+d^2}$

IMEXP:

Description: Using the IMEXP function returns the exponential of a complex number in the *x + yi* or *x + yj* format, entered as a text value.

General Syntax: **=IMEXP(inumber)**

Arguments: **inumber** is the complex number for which you want the exponential.

Function: The exponential of a complex number is: $\mathrm{IMEXP}(z) = e^{(x+yi)} = e^{x}e^{yi} = e^{x}(\cos y + i\sin y)$

IMLN:

Description: Using the IMLN function returns the natural logarithm of a complex number in the *x + yi* or *x + yj* format, entered as a text value.

General Syntax: **=IMLN(inumber)**

Arguments: **inumber** is the complex number for which you want the natural logarithm.

Function: The natural logarithm of a complex number is: $\ln(x+yi) = \ln\sqrt{x^2+y^2} + i\tan^{-1}\left(\frac{y}{x}\right)$

where $\theta \in]-\pi;\pi]$

IMLOG10:

Description: Using the IMLOG10 function returns the common logarithm of a complex number in the *x + yi* or *x + yj* format, entered as a text value.

General Syntax: **=IMLOG10(inumber)**

Arguments: **inumber** is the complex number for which you want the common logarithm.

Function: The common logarithm of a complex number can be calculated from the natural logarithm as shown:

$$\log_{10}(x+yi) = (\log_{10} e)\ln(x+yi)$$

IMLOG2:

Description: Using the IMLOG2 function returns the base-2 logarithm of a complex number in the *x + yi* or *x + yj* format, entered as a text value.

General Syntax: **=IMLOG2(inumber)**

Arguments: **inumber** is the complex number for which you want the base-2 logarithm.

Function: The base-2 logarithm of a complex number can be calculated from the natural logarithm as shown:

$$\log_{2}(x+yi) = (\log_{2} e)\ln(x+yi)$$

IMPOWER:

Description: Using the IMPOWER function returns a complex number in the *x + yi* or *x + yj* format raised to a specified power and displayed as a text value.

General Syntax: **=IMPOWER(inumber, number)**

Arguments: **inumber** is the complex number that you want to raise to a power.
number is the power to which you want to raise the complex number.

Function: A complex number raised to a power can be calculated as shown:

$$(x+yi)^n = r^n e^{i\theta n} = r^n \cos n\theta + ir^n \sin n\theta \text{ where: } r = \sqrt{x^2+y^2}$$

and: $\theta = \tan^{-1}\left(\frac{y}{x}\right)$

and: $\theta \in]-\pi;\pi]$

IMPRODUCT:

Description: Using the IMPRODUCT function returns the product of 2 to 29 complex numbers in the *x + yi* or *x + yj* format, entered as text values.

General Syntax: **=IMPRODUCT(inumbers)**

Arguments: **inumbers** is the set of 2 to 29 complex numbers, separated by commas, for which you want the product.

Formula: The product of two complex numbers is: $(a+bi)(c+di) = (ac - bd) + (ad + bc)i$

Glossary of Functions

IMREAL:

Description: Using the IMREAL function returns the real coefficient of a complex number in the *x + yi* or *x + yj* format, displayed as a text value.

General Syntax: **=IMREAL(inumber)**

Arguments: **inumber** is the complex number for which you want the real coefficient.

IMSIN:

Description: Using the IMSIN function returns the sine of a complex number in the *x + yi* or *x + yj* format, entered as a text value.

General Syntax: **=IMSIN(inumber)**

Arguments: **inumber** is the complex number for which you want the sine.

Formula: The sine of a complex number is: $\sin(x+yi) = \sin(x)\cosh(y) - \cos(x)\sinh(y)i$

IMSQRT:

Description: Using the IMSQRT function returns the square root of a complex number in the *x + yi* or *x + yj* format, entered as a text value.

General Syntax: **=IMSQRT(inumber)**

Arguments: **inumber** is the complex number for which you want the square root.

Formula: The square root of a complex number is: $\sqrt{x+yi} = \sqrt{r}\cos\left(\frac{\theta}{2}\right) + i\sqrt{r}\sin\left(\frac{\theta}{2}\right)$

where: $r = \sqrt{x^2+y^2}$ and: $\theta = \tan^{-1}\left(\frac{y}{x}\right)$ and: $\theta \in]-\pi;\pi]$

IMSUB:

Description: Using the IMSUB function returns the difference of two complex numbers in the *x + yi* or *x + yj* format, entered as a text value.

General Syntax: **=IMSUB(inumber1,inumber2)**

Arguments: **inumber1** is the complex number from which you subtract **inumber2**.
inumber2 is the complex number subtracted from **inumber1**.

Formula: The difference of two complex numbers is: $(a+bi)-(c+di) = (a-c)+(b-d)i$

IMSUM:

Description: Using the IMSUM function returns the sum of 2 to 29 complex numbers in the *x + yi* or *x + yj* format, entered as text values.

General Syntax: **=IMSUM(inumbers)**

Arguments: **inumbers** are the complex numbers for which you want the sum, separated by commas.

Formula: The sum of two complex numbers is: $(a+bi)+(c+di) = (a+c)+(b+d)i$

OCT2BIN:

Description: Using the OCT2BIN function converts an octal number to a binary number.

General Syntax: **=OCT2BIN(number**,*places***)**

Arguments: **number** is the octal number that you want to convert to a binary number.
places is an optional argument that determines the number of characters to use. If omitted, uses the minimum number of characters necessary. Useful for padding the result with leading zeros.

OCT2DEC:

Description: Using the OCT2DEC function converts an octal number to a decimal.

General Syntax: **=OCT2DEC(number)**

Arguments: **number** is the octal number that you want to convert to a decimal number.

OCT2HEX:

Description: Using the OCT2HEX function converts an octal number to a hexadecimal number.

General Syntax: **=OCT2HEX(number**,*places***)**

Arguments: **number** is the octal number that you want to convert to a hexadecimal number.
places is an optional argument that determines the number of characters to use. If omitted, uses the minimum number of characters necessary. Useful for padding the result with leading zeros.

EUROCONVERT:

Description: Using the EUROCONVERT function converts a number to euros, converts a number from euros to a euro member currency, or converts a number from one euro member currency to another by using the euro as an intermediary currency. The currencies available for conversion are those of European Union (EU) members that have adopted the euro. The function uses fixed conversion rates that are established by the EU.

General Syntax: **=EUROCONVERT(number,source,target**,*full_precision,triangulation_precision*)

Arguments: **number** is a value or cell reference containing the amount of currency that you want to convert.
source is a three-letter text string (enclosed in double-quotes), corresponding to the ISO code for the source currency. The following are accepted codes:

Country/Region:	Currency:	ISO Code:
Belgium	franc	BEF
Luxembourg	franc	LUF
Germany	deutsche mark	DEM
Spain	peseta	ESP
France	franc	FRF
Ireland	pound	IEP
Italy	lira	ITL
Netherlands	guilder	NLG
Austria	schilling	ATS
Portugal	escudo	PTE
Finland	markka	FIM
Greece	drachma	GRD
Euro member states	euro	EUR

target is the three-letter text string (enclosed in double-quotes), corresponding to the ISO code for the currency to which you want to convert the **number**.
full_precision is a logical value (TRUE or FALSE), that specifies how to display the result. If FALSE, or omitted, uses the calculation precision value to calculate the result and the display precision value to display the result. If TRUE, shows all significant digits resulting from the calculation.

ISO Code:	Calculation Precision:	Display Precision:
BEF	0	0
LUF	0	0
DEM	2	2
ESP	0	0
FRF	2	2
IEP	2	2
ITL	0	0
NLG	2	2
ATS	2	2
PTE	0	2
FIM	2	2
GRD	0	2
EUR	2	2

triangulation_precision is an integer that is greater than or equal to 3, which specifies the number of significant digits to use for the intermediate euro value created when converting between two euro member currencies. If omitted, Excel doesn't round the intermediate euro value

ACCRINT:

Description: Using the ACCRINT function returns the accrued interest on a security that pays periodic interest.

General Syntax: **=ACCRINT(issue,first_interest,settlement,rate**,*par*,**frequency**,*basis***)**

Arguments:

issue is the date of the security's issue.
first_interest is the security's first interest date.
settlement is the security's settlement date. This is the date after the issue date when the security is traded to the buyer.
rate is the security's annual coupon rate.
par is the security's par value. Excel assumes it is $1,000, if omitted.
frequency is the number of coupon payments per year. For annual payments, input 1. For semiannual payments, input 2. For quarterly payments, input 4.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

Formula: The function is calculated as: $ACCRINT = par \times \frac{rate}{frequency} \times \sum_{i=1}^{NC} \frac{A_i}{NL_i}$

where:

A_i is the number of accrued days for the i^{th} quasi-coupon period within odd period.
NC is the number of quasi-coupon periods that fit in odd period. If this number contains a fraction, it is raised to the next whole number.
NL_i is the normal length in days of the i^{th} quasi-coupon period within odd period.

ACCRINTM:

Description: Using the ACCRINTM function returns the accrued interest on a security that pays interest at maturity.

General Syntax: **=ACCRINTM(issue,maturity,rate**,*par*,*basis***)**

Arguments:

issue is the date of the security's issue.
maturity is the security's maturity date.
rate is the security's annual coupon rate.
par is the security's par value. Excel assumes it is $1,000, if omitted.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

Formula: The function is calculated as: $ACCRINTM = par \times rate \times \frac{A}{D}$

where:

A is the number of accrued days counted according to a monthly basis. For interest at maturity items, the number of days from the issue date to the maturity date is used.
D is the Annual Year Basis.

Glossary of Functions

AMORDEGRC:

Description: Using the AMORDEGRC function returns the depreciation for each accounting period. Used in the French accounting system. If an asset is purchased in the middle of the accounting period, the prorated depreciation is taken into account. The function is similar to AMORLINC, except that a depreciation coefficient is applied in the calculation depending on the life of the asset. If the life of the asset (1/rate) is between 3 and 4 years, the coefficient used is 1.5. If it is between 5 and 6 years, it uses 2. If it is more than 6 years, the coefficient used is 2.5.

General Syntax: **=AMORDEGRC(cost,purchase_date,first_period,salvage,period,rate**,*basis*)

Arguments:
cost is the cost of the asset.
purchase_date is the date of the purchase of the asset.
first_period is the date of the end of the first period.
salvage is the salvage value at the end of the life of the asset.
period is the period.
rate is the rate of depreciation.
basis is an integer that represents which year basis to use.

Integer:	Date System:
0 or omitted	360 (NASD)
1	Actual
3	365 days in a year
4	360 days in a year (European method)

AMORLINC:

Description: Using the AMORLINC function returns the depreciation for each accounting period. This function is provided for the French accounting system. If an asset is purchased in the middle of the accounting period, the prorated depreciation is taken into account.

General Syntax: **=AMORLINC(cost,purchase_date,first_period,salvage,period,rate**,*basis*)

Arguments:
cost is the cost of the asset.
purchase_date is the date of the purchase of the asset.
first_period is the date of the end of the first period.
salvage is the salvage value at the end of the life of the asset.
period is the period.
rate is the rate of depreciation.
basis is an integer that represents which year basis to use.

Integer:	Date System:
0 or omitted	360 (NASD)
1	Actual
3	365 days in a year
4	360 days in a year (European method)

COUPDAYBS:

Description: Using the COUPDAYBS function returns the number of days from the beginning of the coupon period to the settlement date.

General Syntax: **=COUPDAYBS(settlement,maturity,frequency**,*basis***)**

Arguments:
settlement is the security's settlement date.
maturity is the security's maturity date: the date the security expires.
frequency is the number of coupon payments per year. For annual payments, input 1. For semiannual payments, input 2. For quarterly payments, input 4.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:	Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360	1	Actual/Actual
2	Actual/360	3	Actual/365
4	European 30/360		

COUPDAYS:

Description: Using the COUPDAYS function returns the number of days in the coupon period that contains the settlement date.

General Syntax: **=COUPDAYS(settlement,maturity,frequency**,*basis***)**

Arguments:
settlement is the security's settlement date.
maturity is the security's maturity date: the date the security expires.
frequency is the number of coupon payments per year. For annual payments, input 1. For semiannual payments, input 2. For quarterly payments, input 4.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

COUPDAYSNC:

Description: Using the COUPDAYSNC function returns the number of days from the settlement date to the next coupon date.

General Syntax: **=COUPDAYSNC(settlement,maturity,frequency**,*basis***)**

Arguments:
settlement is the security's settlement date.
maturity is the security's maturity date: the date the security expires.
frequency is the number of coupon payments per year. For annual payments, input 1. For semiannual payments, input 2. For quarterly payments, input 4.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

COUPNCD:

Description: Using the COUPNCD function returns a number that represents the next coupon date after the settlement date. Make sure that the cell into which you enter this formula is formatted as a date cell to view the resulting number as a date value.

General Syntax: **=COUPNCD(settlement,maturity,frequency**,*basis***)**

Arguments:
settlement is the security's settlement date.
maturity is the security's maturity date: the date the security expires.
frequency is the number of coupon payments per year. For annual payments, input 1. For semiannual payments, input 2. For quarterly payments, input 4.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

COUPNUM:

Description: Using the COUPNUM function returns the number of coupons payable between the settlement date and maturity date, rounded up to the nearest whole coupon.

General Syntax: **=COUPNUM(settlement,maturity,frequency**,*basis***)**

Arguments:
settlement is the security's settlement date.
maturity is the security's maturity date: the date the security expires.
frequency is the number of coupon payments per year. For annual payments, input 1. For semiannual payments, input 2. For quarterly payments, input 4.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

COUPPCD:

Description: Using the COUPPCD function returns a number that represents the previous coupon date before the settlement date. Make sure that the cell into which you enter this formula is formatted as a date cell to view the resulting number as a date value.

General Syntax: **=COUPPCD(settlement,maturity,frequency**,*basis***)**

Arguments:
settlement is the security's settlement date.
maturity is the security's maturity date: the date the security expires.
frequency is the number of coupon payments per year. For annual payments, input 1. For semiannual payments, input 2. For quarterly payments, input 4.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

Glossary of Functions

CUMIPMT:

Description: Using the CUMIPMT function returns the cumulative interest paid on a loan between two periods specified.

General Syntax: **=CUMIPMT(rate,nper,pv,start_period,end_period,type)**

Arguments:
rate is the interest rate.
nper is the total number of payment periods.
pv is the present value.
start_period is the first period in the calculation. Payment periods are numbered beginning with 1.
end_period is the last period used in the calculation.
type is the timing of the payment. If 0, the payment is at the end of the period. If 1, payment is at the beginning of the period.

CUMPRINC:

Description: Using the CUMPRINC function returns the cumulative principal paid on a loan between **start_period** and **end_period**.

General Syntax: **=CUMPRINC(rate,nper,pv,start_period,end_period,type)**

Arguments:
rate is the interest rate.
nper is the total number of payment periods.
pv is the present value.
start_period is the first period in the calculation. Payment periods are numbered beginning with 1.
end_period is the last period used in the calculation.
type is the timing of the payment. If 0, the payment is at the end of the period. If 1, payment is at the beginning of the period.

DB:

Description: Using the DB function returns the depreciation of an asset for a specified period using the fixed-declining balance method.

General Syntax: **=DB(cost,salvage,life,period,***month***)**

Arguments:
cost is the initial cost of the asset.
salvage is the value of the asset after depreciation.
life is the number of periods over which the asset is depreciated.
period is the period for which you want to calculate the amount of depreciation.
month is the number of months in the first year. If omitted, is assumed to be 12.

Formula: The formula used to calculate depreciation using the fixed-declining method is:
*=(cost-total depreciation from prior periods)*rate*
where *rate = 1-((salvage/cost)^(1/life))*, rounded to 3 decimal places.
The calculation used is different for the first and last periods. When calculating the first period, DB uses the following calculation: *cost*rate*month/12*. In the last period, DB uses the following calculation:
*((cost-total depreciation from prior periods)*rate*(12-month))/12*.

DDB:

Description: Using the DDB function Returns the depreciation of an asset for a specified period using the double-declining balance method or some other method you specify. All arguments must be positive.

General Syntax: **=DB(cost,salvage,life,period,***factor***)**

Arguments:
cost is the initial cost of the asset.
salvage is the value of the asset after depreciation.
life is the number of periods over which the asset is depreciated.
period is the period for which you want to calculate the amount of depreciation.
factor is the rate at which the balance declines. If omitted, it is 2 (the double-declining balance method).

DISC:

Description: Using the DISC function returns the discount rate for a security.

General Syntax: **=DISC(settlement,maturity,pr,redemption**,*basis*)

Arguments:
settlement is the security's settlement date: the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
pr is the security's price per $100 face value.
redemption is the security's redemption value per $100 face value.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

DOLLARDE:

Description: Using the DOLLARDE function converts a dollar price expressed as a fraction into a dollar price expressed as a decimal number. Used to convert fractional dollar numbers, like securities prices, into decimal numbers.

General Syntax: **=DOLLARDE(fractional_dollar,fraction)**

Arguments:
fractional_dollar is the dollar value expressed as a fraction.
fraction is the integer used in the denominator of the fraction.

DOLLARFR:

Description: Using the DOLLARFR function converts a dollar price expressed as a decimal into a dollar price expressed as a fraction.

General Syntax: **=DOLLARFR(decimal_dollar,fraction)**

Arguments:
decimal_dollar is the dollar value expressed as a decimal.
fraction is the integer to use in the denominator of the fraction.

DURATION:

Description: Using the DURATION function returns the Macauley duration for an assumed par value of $100. Duration is defined as the weighted average of the present value of the cash flows and is used as a measure of a bond price's response to changes in yield.

General Syntax: **=DURATION(settlement,maturity,coupon,yld,frequency**,*basis*)

Arguments:
settlement is the security's settlement date: the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
coupon is the security's annual coupon rate.
yld is the security's annual yield.
frequency is the number of coupon payments per year, expressed as an integer.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

Glossary of Functions

EFFECT:

Description: Using the EFFECT function returns the effective annual interest rate, given the nominal annual interest rate and the number of compounding periods per year.

General Syntax: **=EFFECT(rate,nperiods)**

Arguments: **rate** is the nominal interest rate.
nperiods is the number of compounding periods per year.

Formula: The function is calculated according to the following formula:

$$EFFECT = \left(1 + \frac{Nominal\_rate}{Npery}\right)^{Npery} - 1$$

FV:

Description: Using the FV function returns the future value of an investment based on periodic, constant payments and a constant interest rate.

General Syntax: **=FV(rate,nper,pmt**,*pv*,*type***)**

Arguments: **rate** is the interest rate per period.
nper is the total number of payment periods in an annuity.
pmt is the payment made each period; it cannot change over the life of the annuity. Typically, **pmt** contains principal and interest but no other fees or taxes. If **pmt** is omitted, you must include the pv argument.
pv is the present value, or the lump-sum amount that a series of future payments is worth right now. If pv is omitted, it is assumed to be 0 (zero), and you must include the pmt argument.
type is the timing of the payment. If 0, the payment is at the end of the period. If 1, payment is at the beginning of the period.

FVSCHEDULE:

Description: Using the FVSCHEDULE function returns the future value of an initial principal after applying a series of compound interest rates. Use FVSCHEDULE to calculate the future value of an investment with a variable or adjustable rate.

General Syntax: **=FVSCHEDULE(principal,schedule)**

Arguments: **principal** is the present value.
schedule is an array of interest rates to apply. These are usually entered as a cell range reference to the rates listed in the worksheet. If entered numerically, they must all be enclosed in braces {}, with each rate separated by a comma from the other rate.

INTRATE:

Description: Using the INTRATE function returns the interest rate for a fully invested security.

General Syntax: **=INTRATE(settlement,maturity,investment,redemption**,*basis***)**

Arguments: **settlement** is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
investment is the amount invested in the security.
redemption is the amount to be received at maturity.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:	Integer:	Day Count Basis
0 or omitted	US (NASD) 30/360	1	Actual/Actual
2	Actual/360	3	Actual/365
4	European 30/360		

IPMT:

Description: Using the IPMT function returns the interest payment for a given period for an investment based on periodic, constant payments and a constant interest rate.

General Syntax: **=IPMT(rate,per,nper,pv**,*fv,type***)**

Arguments:
rate is the interest rate per period.
per is the period for which you want to find the interest.
nper is the total number of payment periods in an annuity.
pv is the present value that a series of future payments is worth right now.
fv is the future value that you want to attain after the last payment is made. If omitted, is assumed to be zero.
type is the timing of the payment. If 0, the payment is at the end of the period. If 1, payment is at the beginning of the period.

IRR:

Description: Using the IRR function returns the internal rate of return for a series of cash flows represented by the numbers in values. These cash flows do not have to be even, as they would be for an annuity. However, the cash flows must occur at regular intervals, such as monthly or annually. The internal rate of return is the interest rate received for an investment consisting of payments (negative values) and income (positive values) that occur at regular periods.

General Syntax: **=IRR(values,***guess***)**

Arguments:
values is an array (as either a cell range reference, or as numbers listed in braces {}) of the numbers for which you want to calculate the internal rate of return. Must contain at least one positive and one negative value. Uses the order of values to determine the order of cash flows. Be sure to place values in the desired order.
guess is a number that you guess is close to the result of IRR. Excel uses an iterative technique for calculating IRR. Starting with *guess*, IRR cycles through the calculation until the result is accurate within 0.00001 percent. If IRR can't find a result that works after 20 tries, the #NUM! error value is returned. In most cases you do not need to provide guess for the IRR calculation. If guess is omitted, it is assumed to be 0.1 (10 percent).

ISPMT:

Description: Using the ISPMT function returns the interest paid during a specific period of an investment. This function is provided for compatibility with Lotus 1-2-3 .

General Syntax: **=ISPMT(rate,per,nper,pv)**

Arguments:
rate is the interest rate for the investment.
per is the period for which you want to find the interest.
nper is the total number of payment periods for the investment.
pv is the present value of the investment right now.

MDURATION:

Description: Using the MDURATION function returns the modified Macauley duration for a security with an assumed par value of $100.

General Syntax: **=MDURATION(settlement,maturity,coupon,yld,frequency**,*basis*)

Arguments:
settlement is the security's settlement date: the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
coupon is the security's annual coupon rate.
yld is the security's annual yield.
frequency is the number of coupon payments per year, expressed as an integer.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

Formula: The modified duration is defined as follows:

$$MDURATION = \frac{DURATION}{1+\left(\frac{\text{Market yield}}{\text{Coupon payments per year}}\right)}$$

MIRR:

Description: Using the MIRR function returns the modified internal rate of return for a series of periodic cash flows. MIRR considers both the cost of the investment and the interest received on reinvestment of cash .

General Syntax: **=MIRR(values,finance_rate,reinvest_rate)**

Arguments:
values is an array (as either a cell range reference, or as numbers listed in braces {}) of the numbers for which you want to calculate the internal rate of return. Must contain at least one positive and one negative value. Uses the order of values to determine the order of cash flows. Be sure to place values in the desired order.
finance_rate is the interest rate you pay on the money used in the cash flows.
reinvest_rate is the interest rate you receive on the cash flows as you reinvest them.

Formula: If *n* is the number of cash flows in values, *frate* is the finance_rate, and *rrate* is the reinvest rate, then the formula for calculating MIRR is as follows:

$$\left(\frac{-NPV(rrate, values[positive]) * (1+rrate)^{n}}{NPV(frate, values[negative]) * (1+frate)}\right)^{\frac{1}{n-1}} - 1$$

NOMINAL:

Description: Using the NOMINAL function returns the nominal annual interest rate, given the effective rate and the number of compounding periods per year.

General Syntax: **=NOMINAL(effective_rate,npery)**

Arguments:
effective_rate is the effective interest rate.
npery is the number of compounding periods per year.

NPER:

Description: Using the NPER function returns the number of periods for an investment based on periodic, constant payments and a constant interest rate.

General Syntax: **=NPER(rate,pmt,pv**,*fv,type***)**

Arguments:
rate is the interest rate per period.
pmt is the payment made each period; it cannot change over the life of the annuity. Typically, **pmt** contains principal and interest but no other fees or taxes.
pv is the present value, or the lump-sum amount that a series of future payments is worth right now.
fv is the future value that you want to attain after the last payment is made. If omitted, is assumed to be zero.
type is the timing of the payment. If 0, the payment is at the end of the period. If 1, payment is at the beginning of the period.

NPV:

Description: Using the NPV function calculates the net present value of an investment by using a discount rate and a series of future payments (negative values) and income (positive values).

General Syntax: **=NPV(rate,values)**

Arguments:
rate is the rate of discount over the length of one period.
values are 1 to 29 arguments representing the payments and income. **Values** must be equally spaced in time and occur at the end of each period. NPV uses the **values** to interpret the order of cash flows. Be sure to enter your payment and income values in the correct sequence.

Formula: If *n* is the number of cash flows, the formula for NPV is: $\mathrm{NPV} = \sum_{i=1}^{n} \frac{values_i}{(1+rate)^i}$

ODDFPRICE:

Description: Using the ODDFPRICE function returns the price per $100 face value of a security having an odd (short or long) first period.

General Syntax: **=ODDFPRICE(settlement,maturity,issue,first_coupon,rate,yld,redemption, frequency**,*basis***)**

Arguments:
settlement is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
issue is the security's issue date.
first_coupon is the security's first coupon date.
rate is the security's interest rate.
yld is the security's annual yield.
redemption is the security's redemption value per $100 face value.
frequency is the number of coupon payments per year, expressed as an integer.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

ODDFYIELD:

Description: Using the ODDFYIELD function returns the yield of a security having an odd (short or long) first period.

General Syntax: **=ODDFYIELD(settlement,maturity,issue,first_coupon,rate,pr,redemption, frequency**,*basis*)

Arguments:

settlement is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
issue is the security's issue date.
first_coupon is the security's first coupon date.
rate is the security's interest rate.
pr is the security's price.
redemption is the security's redemption value per $100 face value.
frequency is the number of coupon payments per year, expressed as an integer.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

ODDLPRICE:

Description: Using the ODDLPRICE function returns the price per $100 face value of a security having an odd (short or long) last period.

General Syntax: **=ODDLPRICE(settlement,maturity,last_interest,rate,yld,redemption, frequency**,*basis*)

Arguments:

settlement is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
last_interest is the security's last coupon date.
rate is the security's interest rate.
yld is the security's annual yield.
redemption is the security's redemption value per $100 face value.
frequency is the number of coupon payments per year, expressed as an integer.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

ODDLYIELD:

Description: Using the ODDLYIELD function returns the yield of a security having an odd (short or long) last period.

General Syntax: **=ODDLYIELD(settlement,maturity,last_interest,rate,pr,redemption, frequency**,*basis*)

Arguments:
settlement is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
issue is the security's issue date.
last_interest is the security's last coupon date.
rate is the security's interest rate.
pr is the security's price.
redemption is the security's redemption value per $100 face value.
frequency is the number of coupon payments per year, expressed as an integer.
basis is the integer that represents which type of day count basis to use.

Integer:	Day Count Basis:
0 or omitted	US (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

PMT:

Description: Using the PMT function calculates the payment for a loan based on constant payments and a constant interest rate.

General Syntax: **=PMT(rate,nper,pv**,*fv,type*)

Arguments:
rate is the interest rate for the loan.
nper is the total number of payments for the loan.
pv is the present value, or principal.
fv is the future value that you want to attain after the last payment is made. If omitted, is assumed to be zero.
type is the timing of the payment. If 0, the payment is at the end of the period. If 1, payment is at the beginning of the period.

PPMT:

Description: Using the PPMT function returns the payment on the principal for a given period for an investment based on periodic, constant payments and a constant interest rate.

General Syntax: **=PPMT(rate,per,nper,pv**,*fv,type*)

Arguments:
rate is the interest rate per period.
per specifies the period.
nper is the total number of payment periods.
pv is the present value.
fv is the future value that you want to attain after the last payment is made. If omitted, is assumed to be zero.
type is the timing of the payment. If 0, the payment is at the end of the period. If 1, payment is at the beginning of the period.

PRICE:

Description: Using the PRICE function returns the price per $100 face value of a security that pays periodic interest.

General Syntax: **=PRICE(settlement,maturity,rate,yld,redemption,frequency**,*basis***)**

Arguments:
settlement is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
rate is the security's annual coupon rate.
yld is the security's annual yield.
redemption is the security's redemption value per $100 face value.
frequency is the number of coupon payments per year, expressed as an integer.
basis is the integer that represents which type of day count basis to use.

PRICEDISC:

Description: Using the PRICEDISC function returns the price per $100 face value of a discounted security.

General Syntax: **=PRICEDISC(settlement,maturity,discount,redemption**,*basis***)**

Arguments:
settlement is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
discount is the security's discount rate.
redemption is the security's redemption value per $100 face value.
frequency is the number of coupon payments per year, expressed as an integer.
basis is the integer that represents which type of day count basis to use. If omitted, uses the US (NASD) 30/360 day count.

PRICEMAT:

Description: Using the PRICEMAT function returns the price per $100 face value of a security that pays interest at maturity.

General Syntax: **=PRICEMAT(settlement,maturity,issue,rate,yld**,*basis***)**

Arguments:
settlement is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
issue is the security's issue date.
rate is the security's interest rate at date of issue.
yld is the security's annual yield.
basis is the integer that represents which type of day count basis to use. If omitted, uses the US (NASD) 30/360 day count.

PV:

Description: Using the PV function returns the present value of an investment.

General Syntax: **=PV(rate,nper,pmt**,*fv*,*type***)**

Arguments:
rate is the interest rate per period.
nper is the total number of payment periods.
pmt is the payment made each period.
fv is the future value that you want to attain after the last payment is made. If omitted, is assumed to be zero.
type is the timing of the payment. If 0, the payment is at the end of the period. If 1, payment is at the beginning of the period.

RATE:

Description: Using the RATE function returns the interest rate per period of an annuity. RATE is calculated by iteration and can have zero or more solutions. If the successive results of RATE do not converge to within 0.0000001 after 20 iterations, RATE returns the #NUM! error value.

General Syntax: **=RATE(nper,pmt,pv**,*fv*,*type*,*guess***)**

Arguments:
nper is the total number of payment periods in an annuity.
pmt is the payment made each period.
pv is the present value.
fv is the future value that you want to attain after the last payment is made. If omitted, is assumed to be zero.
type is the timing of the payment. If 0, the payment is at the end of the period. If 1, payment is at the beginning of the period.

RECEIVED:

Description: Using the RECEIVED function returns the amount received at maturity for a fully invested security.

General Syntax: **=RECEIVED(settlement,maturity,investment,discount**,*basis***)**

Arguments:
settlement is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
investment is the amount invested in the security.
discount is the security's discount rate.
basis is the integer that represents which type of day count basis to use. If omitted, uses the US (NASD) 30/360 day count.

SLN:

Description: Using the SLN function returns the straight-line depreciation of an asset for a period.

General Syntax: **=SLN(cost,salvage,life)**

Arguments:
cost is the initial cost of the asset.
salvage is the value of the asset after depreciation.
life is the number of periods over which the asset is depreciated.

SYD:

Description: Using the SYD function returns the sum-of-years' digits depreciation of an asset for a specified period.

General Syntax: **=SYD(cost,salvage,life,per)**

Arguments:
cost is the initial cost of the asset.
salvage is the value of the asset after depreciation.
life is the number of periods over which the asset is depreciated.
per is the period and must use the same units as **life**.

TBILLEQ:

Description: Using the TBILLEQ function returns the bond-equivalent yield for a Treasury bill.

General Syntax: **=TBILLEQ(settlement,maturity,discount)**

Arguments:
settlement is the date after issue when the bill is traded to the buyer.
maturity is the date when the bill expires.
discount is the bill's discount rate.

TBILLPRICE:

Description: Using the TBILLPRICE function returns the price per $100 face value for a Treasury bill.

General Syntax: **=TBILLPRICE(settlement,maturity,discount)**

Arguments:
settlement is the date after issue when the bill is traded to the buyer.
maturity is the date when the bill expires.
discount is the bill's discount rate.

TBILLYIELD:

Description: Using the TBILLYIELD function returns the yield for a Treasury bill.

General Syntax: **=TBILLYIELD(settlement,maturity,pr)**

Arguments:
settlement is the date after issue when the bill is traded to the buyer.
maturity is the date when the bill expires.
pr is the Treasury bill's price per $100 face value.

VDB:

Description: Using the VDB function returns the depreciation of an asset for any period you specify, including partial periods, using the double-declining balance method or some other method you specify.

General Syntax: **=VDB(cost,salvage,life,start_period,end_period**,*factor,no_switch*)

Arguments:
cost is the initial cost of the asset.
salvage is the value of the asset after depreciation.
life is the number of periods over which the asset is depreciated.
start_period is the starting period for which you want to calculate depreciation.
end_period is the ending period for which you want to calculate depreciation.
factor is the rate at which the balance declines. If omitted, 2 (double-declining) is assumed.
no_switch is a logical value that specifies whether or not to switch to straight-line depreciation when the depreciation is greater than the declining balance calculation. If omitted, or FALSE, Excel will switch. If TRUE, it will not switch.

XIRR:

Description: Using the XIRR function returns the internal rate of return for a schedule of cash flows that is not necessarily periodic. To calculate the internal rate of return for a series of periodic cash flows, use the IRR function.

General Syntax: **=XIRR(values,dates**,*guess*)

Arguments:
values is a series of cash flows that corresponds to a schedule of payments in dates. The first payment is optional and corresponds to a cost or payment that occurs at the beginning of the investment. If the first value is a cost or payment, it must be a negative value. All succeeding payments are discounted based on a 365-day year. The series of values must contain at least one positive and one negative value.
dates is a schedule of payment dates that corresponds to the cash flow payments. The first payment date indicates the beginning of the schedule of payments. All other dates must be later than this date, but they may occur in any order.
guess is a number that you guess is close to the result of XIRR. Excel uses an iterative technique for calculating XIRR. Starting with *guess*, XIRR cycles through the calculation until the result is accurate within 0.00001 percent. If XIRR can't find a result that works after 100 tries, the #NUM! error value is returned. In most cases you do not need to provide guess for the XIRR calculation. If guess is omitted, it is assumed to be 0.1 (10 percent).

XNPV:

Description: Using the XNPV function returns the net present value for a schedule of cash flows that is not necessarily periodic.

General Syntax: **=XNPV(rate,values,dates)**

Arguments:
rate is the discount rate to apply to the cash flows.
values is a series of cash flows that corresponds to a schedule of payments in dates. The first payment is optional and corresponds to a cost or payment that occurs at the beginning of the investment. If the first value is a cost or payment, it must be a negative value. All succeeding payments are discounted based on a 365-day year. The series of values must contain at least one positive value and one negative value.
dates is a schedule of payment dates that corresponds to the cash flow payments. The first payment date indicates the beginning of the schedule of payments. All other dates must be later than this date, but they may occur in any order.

YIELD:

Description: Using the YIELD function returns the yield on a security that pays periodic interest. Use YIELD to calculate bond yield.

General Syntax: **=YIELD(settlement,maturity,rate,pr,redemption,frequency**,*basis*)

Arguments:
settlement is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
rate is the security's annual coupon rate.
pr is the security's price per $100 face value.
redemption is the security's redemption value per $100 face value.
frequency is the number of coupon payments per year, expressed as an integer.
basis is the integer that represents which type of day count basis to use. If omitted, uses the US (NASD) 30/360 day count.

YIELDDISC:

Description: Using the YIELDDISC function returns the annual yield on a discounted security.

General Syntax: **=YIELDDISC(settlement,maturity,pr,redemption**,*basis*)

Arguments:
settlement is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
pr is the security's price per $100 face value.
redemption is the security's redemption value per $100 face value.
basis is the integer that represents which type of day count basis to use. If omitted, uses the US (NASD) 30/360 day count.

YIELDMAT:

Description: Using the YIELDMAT function returns the annual yield of a security that pays interest at maturity.

General Syntax: **=YIELDMAT(settlement,maturity,issue,rate,pr**,*basis*)

Arguments:
settlement is the date after issue when the security is traded to the buyer.
maturity is the date when the security expires.
issue is the security's issue date.
rate is the security's interest rate at the date of issue.
pr is the security's price per $100 face value.
redemption is the security's redemption value per $100 face value.
basis is the integer that represents which type of day count basis to use. If omitted, uses the US (NASD) 30/360 day count.

CELL:

Description: Using the CELL function returns information about the formatting, location, or contents of the upper-left cell in a reference. The CELL function is provided for compatibility with other spreadsheet programs.

General Syntax: **=CELL(info_type**,*reference***)**

Arguments: **info_type** is a text value that specifies what type of cell information you want. This table lists the possible values and their corresponding results:

info_type:	***returns:***
"address“	Reference of the first cell in reference, as text.
"col“	Column number of the cell in reference.
"color“	1 if the cell is formatted in color for negative values; otherwise returns 0 (zero).
"contents“	Value of the upper-left cell in reference; not a formula.
“filename“	Filename (including full path) of the file that contains reference, as text. Returns empty text ("") if the worksheet that contains reference has not yet been saved.
"format“	Text value corresponding to the number format of the cell. The text values for the various formats are shown in the following table. Returns "-" at the end of the text value if the cell is formatted in color for negative values. Returns "()" at the end of the text value if the cell is formatted with parentheses for positive or all values.
"parentheses“	1 if the cell is formatted with parentheses for positive or all values; otherwise returns 0.
"prefix“	Text value corresponding to the "label prefix" of the cell. Returns single quotation mark (') if the cell contains left-aligned text, double quotation mark (") if the cell contains right-aligned text, caret (^) if the cell contains centered text, backslash (\) if the cell contains fill-aligned text, and empty text ("") if the cell contains anything else.
"protect“	0 if the cell is not locked, and 1 if the cell is locked.
"row“	Row number of the cell in reference.
"type“	Text value corresponding to the type of data in the cell. Returns "b" for blank if the cell is empty, "l" for label if the cell contains a text constant, and "v" for value if the cell contains anything else.
"width“	Column width of the cell rounded off to an integer. Each unit of column width is equal to the width of one character in the default font size.

reference is the cell that you want information about. If omitted, information specified in **info_type** is returned for the last cell that was changed. The following list describes the text values CELL returns when **info_type** is "format," and reference is a cell formatted with a built-in number format.

If the Microsoft Excel format is:	***CELL returns:***
General	"G“
0	"F0“
#,##0	",0"
0.00	"F2"
#,##0.00	",2"
$#,##0_);($#,##0)	"C0"
$#,##0_);[Red]($#,##0)	"C0-"
$#,##0.00_);($#,##0.00)	"C2"
$#,##0.00_);[Red]($#,##0.00)	"C2-"
0%	"P0"
0.00%	"P2"
0.00E+00	"S2"
# ?/? or # ??/??	"G"
m/d/yy or m/d/yy h:mm or mm/dd/yy	"D4"
d-mmm-yy or dd-mmm-yy	"D1"
d-mmm or dd-mmm	"D2"
mmm-yy	"D3"
mm/dd	"D5"
h:mm AM/PM	"D7"
h:mm:ss AM/PM	"D6"
h:mm	"D9"
h:mm:ss	"D8"

ERROR.TYPE:

Description: Using the ERROR.TYPE function returns a number corresponding to one of the error values in Microsoft Excel or returns the #N/A error if no error exists. You can use ERROR.TYPE in an IF function to test for an error value and return a text string, such as a message, instead of the error value.

General Syntax: **=ERROR.TYPE(error_val)**

Arguments: **error_val** is the error value whose identifying number you want to find. Although **error_val** can be the actual error value, it will usually be a reference to a cell containing a formula that you want to test

If error_val is:	***ERROR.TYPE returns:***
#NULL!	1
#DIV/0!	2
#VALUE!	3
#REF!	4
#NAME?	5
#NUM!	6
#N/A	7
Anything else	#N/A

INFO:

Description: Using the INFO function returns information about the current operating environment. You should use this feature with caution as sensitive or confidential information could be revealed to other users!

General Syntax: **=INFO(type_text)**

Arguments: **type_text** is the text value that specifies what type of information you want returned.

***Type_text*:**	***Returns:***
"directory“	Path of the current directory or folder.
"memavail“	Amount of memory available, in bytes.
"memused“	Amount of memory being used for data.
"numfile“	Number of active worksheets in the open workbooks.
"origin“	Absolute A1-style reference, as text, prepended with "$A:" for Lotus 1-2-3 release 3.x compatibility. Returns the cell reference of the top and leftmost cell visible in the window, based on the current scrolling position.
"osversion“	Current operating system version, as text.
"recalc“	Current recalculation mode; returns "Automatic" or "Manual".
"release“	Version of Microsoft Excel, as text.
"system“	Name of the operating environment: Macintosh = "mac;“ Windows = "pcdos;“
"totmem“	Total memory available, including memory already in use, in bytes.

ISBLANK:

Description: You use the ISBLANK function in conjunction with the IF function to test whether or not the cell to which you refer is blank.

General Syntax: **=IF(ISBLANK(cell),true_value,false_value)**

Arguments:
cell is the cell that you want to test to see if it contains information.
true_value is the value to return if the **cell** is empty.
false_value is the value to return if the **cell** is not empty.

ISERR:

Description: You use the ISERR function in conjunction with the IF function to test whether or not the cell to which you refer contains any error value (*other than* the "#N/A" error).

General Syntax: **=IF(ISERR(cell),true_value,false_value)**

Arguments: **cell** is the cell that you want to test to see if it contains an error message.
true_value is the value to return if the **cell** does contain an error.
false_value is the value to return if the **cell** does not contain an error.

ISERROR:

Description: You use the ISERROR function in conjunction with the IF function to test whether or not the cell to which you refer contains any error value (*including* the "#N/A" error).

General Syntax: **=IF(ISERROR(cell),true_value,false_value)**

Arguments: **cell** is the cell that you want to test to see if it contains an error message.
true_value is the value to return if the **cell** does contain an error.
false_value is the value to return if the **cell** does not contain an error.

ISEVEN:

Description: You use the ISEVEN function in conjunction with the IF function to test whether or not the cell to which you refer contains an even number.

General Syntax: **=IF(ISEVEN(cell),true_value,false_value)**

Arguments: **cell** is the cell that you want to test to see if it contains an even number.
true_value is the value to return if the **cell** does contain an even number.
false_value is the value to return if the **cell** does not contain an even number.

ISLOGICAL:

Description: You use the ISLOGICAL function in conjunction with the IF function to test whether or not the cell to which you refer contains a logical value (TRUE or FALSE).

General Syntax: **=IF(ISLOGICAL(cell),true_value,false_value)**

Arguments: **cell** is the cell that you want to test to see if it contains a logical value.
true_value is the value to return if the **cell** does contain a logical value.
false_value is the value to return if the **cell** does not contain a logical value.

ISNA:

Description: You use the ISNA function in conjunction with the IF function to test whether or not the cell to which you refer contains an "#N/A" error message (value unavailable).

General Syntax: **=IF(ISNA(cell),true_value,false_value)**

Arguments: **cell** is the cell that you want to test to see if it contains the "#N/A" value.
true_value is the value to return if the **cell** does contain the "#N/A" value.
false_value is the value to return if the **cell** does not contain the "#N/A" value.

ISNONTEXT:

Description: You use the ISNONTEXT function in conjunction with the IF function to test whether or not the cell to which you refer contains a non-text value (even a blank cell).

General Syntax: **=IF(ISNONTEXT(cell),true_value,false_value)**

Arguments: **cell** is the cell that you want to test to see if it contains a non-text value.
true_value is the value to return if the **cell** does contain the a non-text value.
false_value is the value to return if the **cell** does not contain a non-text value.

ISNUMBER:

Description: You use the ISNUMBER function in conjunction with the IF function to test whether or not the cell to which you refer contains a number.

General Syntax: **=IF(ISNUMBER(cell),true_value,false_value)**

Arguments: **cell** is the cell that you want to test to see if it contains a number.
true_value is the value to return if the **cell** does contain the a number.
false_value is the value to return if the **cell** does not contain a number.

ISODD:

Description: You use the ISODD function in conjunction with the IF function to test whether or not the cell to which you refer contains an odd number.

General Syntax: **=IF(ISODD(cell),true_value,false_value)**

Arguments: **cell** is the cell that you want to test to see if it contains an odd number.
true_value is the value to return if the **cell** does contain an odd number.
false_value is the value to return if the **cell** does not contain an odd number.

ISREF:

Description: You use the ISREF function in conjunction with the IF function to test whether or not the value to which you refer is a cell reference.

General Syntax: **=IF(ISREF(value),true_value,false_value)**

Arguments: **value** is the value that you want to test to see if it contains a reference.
true_value is the value to return if the **value** does contain a reference.
false_value is the value to return if the **value** does not contain a reference.

ISTEXT:

Description: You use the ISTEXT function in conjunction with the IF function to test whether or not the cell to which you refer contains a text value.

General Syntax: **=IF(ISTEXT(cell),true_value,false_value)**

Arguments: **cell** is the cell that you want to test to see if it contains a text value.
true_value is the value to return if the **cell** does contain a text value.
false_value is the value to return if the **cell** does not contain a text value.

N:

Description: Using the N function returns a value converted to a number.

General Syntax: **=N(value)**

Arguments: **value** is the value you want to convert to a number. If **value** is a number, the function returns that number. If **value** is a date, the function returns the serial number of that date. If **value** is TRUE, the function returns 1. If **value** is FALSE, the function returns 0. If **value** is an error, the function returns that error. If **value** is anything else, the function returns 0.

NA:

Description: You use the NA function to return the "#N/A" error message. Used to mark empty cells.

General Syntax: **=NA()**

Arguments: none.

TYPE:

Description: You use the TYPE function to return a number which indicates the type of data in the value that you test.

General Syntax: **=TYPE(value)**

Arguments: **value** is any Excel value that you want to test for its data type. If TYPE returns a 1, the **value** is a number. If TYPE returns a 2, the **value** is text. If TYPE returns a 4, the **value** is a logical value. If TYPE returns a 16, the **value** is an error message. If TYPE returns a 64, the **value** is an array.

AND:

Description: You use the AND function to verify that multiple conditions specified by logical tests given in the arguments are all TRUE. If so, the AND function returns TRUE, otherwise it will return FALSE if any test returns FALSE.

General Syntax: **=AND(test1,test2,…)**

Arguments: **test1,test2,…** are a series of logical conditions, each separated by a comma, that must all evaluate to a TRUE value in order for the AND function to return a TRUE value.

FALSE:

Description: You use the FALSE function to return a logical FALSE value. You may also type the word FALSE into a cell and Excel will interpret that as the logical value FALSE.

General Syntax: **=FALSE()**

Arguments: none.

IF:

Description: Using the IF function will evaluate a logical test and will return one result if the logical test returns a TRUE value. It will return another result if the logical test returns a FALSE value.

General Syntax: **=IF(logical_test,true_result,false_result)**

Arguments: **logical_test** is a logical test or statement that can be evaluated to either TRUE or FALSE.
true_result is the value to be returned if the **logical_test** evaluates to a TRUE value.
false_value is the value to be returned if the **logical_test** evaluates to a FALSE value.

NOT:

Description: Using the NOT function will reverse the logical value returned by its argument.

General Syntax: **=NOT(logical_test)**

Arguments: **logical_test** is a logical test or statement that can be evaluated to either TRUE or FALSE.

OR:

Description: You use the OR function to verify that any condition given in set of logical tests returns a TRUE value. If so, the OR function returns TRUE, otherwise it will return FALSE if all tests return FALSE values.

General Syntax: **=OR(test1,test2,…)**

Arguments: **test1,test2,…** are a series of logical conditions, each separated by a comma.

TRUE:

Description: You use the TRUE function to return a logical TRUE value. You may also type the word TRUE into a cell and Excel will interpret that as the logical value TRUE .

General Syntax: **=TRUE()**

Arguments: none.

ADDRESS:

Description: Using the ADDRESS function will create a cell address as a text value, given the row and column references.

General Syntax: **=ADDRESS(row,column**,*abs_num,a1,sheet_text***)**

Arguments:

row is the row number to use in the cell reference.
column is the column letter to use for the cell reference.
abs_num is an integer that specifies what type of cell reference to return. If 1, or omitted, returns an absolute cell reference. If 2, returns an absolute row and relative column reference. If 3, returns a relative row and absolute column reference. If 4, returns a relative cell reference.
a1 is a logical value (TRUE or FALSE) that specifies either an A1 style cell reference (e.g. B5) or an R1C1 style reference (e.g. R5C2). If TRUE, or omitted, returns an A1 style cell reference.
sheet_text is a text reference to the name of the worksheet. Used for external cell references. If omitted, no sheet name is used in the cell reference.

AREAS:

Description: Using the AREAS function will return the number of areas in a reference. An area is a range of contiguous cells or a single cell.

General Syntax: **=AREAS(reference)**

Arguments: **reference** is a reference to a cell or range of cells and can refer to multiple areas. If you want to specify several references as a single argument, then you must include extra sets of parentheses so that Microsoft Excel will not interpret the comma as a field separator.

Glossary of Functions

CHOOSE:

Description: Using the CHOOSE function will return a value from a list of available values (up to 29) based on an index number specified.

General Syntax: **=CHOOSE(index_number,value1,value2…)**

Arguments: **index_number** is an integer (from 1 to 29) that represents which value of the listed values will be returned.
value1,value2… are the values (up to 29) that the CHOOSE function returns, based on the order that they are entered and the selected **index_number**.

COLUMN:

Description: Using the COLUMN function will return the column number of the given reference.

General Syntax: **=COLUMN(***reference***)**

Arguments: *reference* is the cell for which you want the column number. If omitted, returns the column of the cell into which the function is placed.

COLUMNS:

Description: Using the COLUMNS function will return the number of columns in a reference.

General Syntax: **=COLUMNS(array)**

Arguments: **array** is the cell range or reference for which you want the number of columns.

GETPIVOTDATA:

Description: Using the GETPIVOTDATA function will return visible data stored in a PivotTable report. Note that this formula may be needlessly complex, as you can enter a formula by typing = and the reference to the cell that contains the data you want.

General Syntax: **=GETPIVOTDATA(data_field,pivot_table**,*field1,item1,field2,item2,...***)**

Arguments: **data_field** is the name of the data field that contains the data you want to retrieve.
pivot_table is a reference to any cell in a PivotTable report. Used to determine which PivotTable report contains the data you want to retrieve.
field1,item1,field2,item2,... are up to 14 pairs of field names and item names that describe the data you want to retrieve. Field names and names for items other than dates and numbers are enclosed in quotation marks.

HLOOKUP:

Description: Using the HLOOKUP function searches for a value in the top row of a table, and returns a value in the same column from a row you specify.

General Syntax: **=HLOOKUP(lookup_value,table,row_number**,*range_lookup***)**

Arguments: **lookup_value** is the value to be looked up in the first row of the table.
table is a range reference or named range of the table in which you want to look.
row_number is the number of the row from which a matching value will be returned.
range_lookup is a logical value. If TRUE, or omitted, will return the next lowest value under the **lookup_value** if **lookup_value** doesn't match a value exactly. If FALSE, will return the exact match of the **lookup_value**.

HYPERLINK:

Description: Using the HYPERLINK function will create a link to the file given in the arguments.

General Syntax: **=HYPERLINK(link**,*friendly_name***)**

Arguments: **link** is the location of the internet address, network address, or file location of the item that you want to open when the user clicks the cell. Must be enclosed in double-quotes.
friendly_name is displayed text for the cell to be shown instead of the **link**. If omitted, will display the **link** instead.

INDEX:

Description: Using the INDEX function will return a value or the reference to a value from within a table or range. There are two forms of the INDEX() function: array and reference. The array form always returns a value or an array of values; the reference form always returns a reference.

Array Syntax: **=INDEX(array**,*row_num,column_num***)**

Array Arguments: **array** is a range of cells or an array constant, enclosed in braces {}.
row_num is the row number in the array from which to return a value.
column_num is the column number in the array from which to return a value.

Reference Syntax: **=INDEX(reference**,row_num,column_num,area_num**)**

Reference Arguments: **reference** is a reference to one or more cell ranges.
row_num is the row number in **reference** from which to return a reference.
column_num is the column number in **reference** from which to return a reference.
area_num is a range in **reference** from which to return the intersection of *row_num* and *column_num*. If omitted, uses area 1.

INDIRECT:

Description: Using the INDIRECT function returns the reference specified by a text string, immediately evaluated to display its contents. Use INDIRECT when you want to change the reference to a cell within a formula without changing the formula itself.

General Syntax: **=INDIRECT(ref_text**,*a1***)**

Arguments: **ref_text** is a cell reference.
a1 is a logical value that specifies the style of reference. If TRUE, or omitted, assumes the reference is in the A1 style. If FALSE, assumes a R1C1 style.

LOOKUP:

Description: Using the LOOKUP function returns a value either from a one-row or one-column range or from an array. The LOOKUP function has two syntax forms: vector and array. The vector form of LOOKUP looks in a one-row or one-column range (known as a vector) for a value and returns a value from the same position in a second one-row or one-column range. The array form of LOOKUP looks in the first row or column of an array for the specified value and returns a value from the same position in the last row or column of the array.

Vector Syntax: **=LOOKUP(lookup_value,lookup_vector,result_vector)**

Vector Arguments:
lookup_value is the value to be looked up in the first vector.
lookup_vector is the range that contains only one row or one column. Must be placed in ascending order or the function may not work correctly.
result_vector is an adjacent range of the same size and shape as **lookup_vector**.

Array Syntax: **=LOOKUP(lookup_value,array)**

Array Arguments:
lookup_value is the value to be looked up in the array.
array is the range of cells that contains text, numbers, or logical values that you want to compare with **lookup_value**. The values in array must be placed in ascending order for this function to work correctly. If **array** covers an area that is wider than it is tall (more columns than rows), LOOKUP searches for **lookup_value** in the first row. If array is square or is taller than it is wide (more rows than columns), LOOKUP searches in the first column. With HLOOKUP and VLOOKUP, you can index down or across, but LOOKUP always selects the last value in the row or column.

MATCH:

Description: Using the MATCH function returns the relative position of an item in an array that matches a specified value in a specified order. Use MATCH instead of one of the LOOKUP functions when you need the position of an item in a range returned instead of the item itself returned.

General Syntax: **=MATCH(lookup_value,lookup_array**,*match_type***)**

Arguments:
lookup_value is the value to be looked up.
lookup_array is a contiguous range of cells containing possible lookup values. **Lookup_array** must be an array or an array reference.
match_type specifies how Excel matches **lookup_value** with the **lookup_array**. If 1, or omitted, returns the largest value that is less than or equal to **lookup_value**. If 0, returns the first value that is exactly equal to **lookup_value**. If -1, returns the smallest value that is greater than or equal to **lookup_value**.

OFFSET:

Description: Using the OFFSET function returns a reference to a range that is a specified number of rows and columns from a cell or range of cells. The reference that is returned can be a single cell or a range of cells. You can specify the number of rows and the number of columns to be returned.

General Syntax: **=OFFSET(reference,rows,cols**,*height,width***)**

Arguments:
reference the cell or cell range from which you want to base the offset.
rows is the number of rows, up or down from the **reference** to which you want the upper left cell to refer. Can be positive or negative.
cols is the number of columns, to the left or right from the **reference** to which you want the upper left cell to refer. Can be positive or negative.
height is the number of rows to return. Must be positive. If omitted, returns the same number of rows as the **reference**.
width is the number of columns to return. Must be positive. If omitted, returns the same number of columns as the **reference**.

ROW:

Description: Using the ROW function will return the row number of the given reference.

General Syntax: **=ROW(***reference***)**

Arguments: *reference* is the cell for which you want the row number. If omitted, returns the row of the cell into which the function is placed.

ROWS:

Description: Using the ROWS function will return the number of rows in a reference.

General Syntax: **=ROWS(array)**

Arguments: **array** is the cell range or reference for which you want the number of rows.

RTD:

Description: Retrieves real-time data from a program that supports COM automation.

General Syntax: **=RTD(prog_id***,server,topic1,topic2,etc…***)**

Arguments: **prog_id** is the name of the ProgID of a registered COM automation add-in that has been installed on the local computer. Enclose the name in quotation marks.
server is the name of the server where the add-in should be run. If there is no server, and the program is run locally, leave the argument blank.
*topic1,topic2,etc…*is 1 to 28 parameters that together represent a unique piece of real-time data.

TRANSPOSE:

Description: The TRANSPOSE function returns a vertical range of cells as a horizontal range, or vice versa. TRANSPOSE must be entered as an array formula in a range that has the same number of rows and columns, respectively, as an array has columns and rows. Use TRANSPOSE to shift the vertical and horizontal orientation of an array on a worksheet.

General Syntax: **=TRANSPOSE(array)**

Arguments: **array** is an array of cells on the worksheet that you would like to transpose.

VLOOKUP:

Description: Using the VLOOKUP function searches for a value in the leftmost column of a table, and returns a value in the same row from a column that you specify.

General Syntax: **=VLOOKUP(lookup_value,table,col_number***,range_lookup***)**

Arguments: **lookup_value** is the value to be looked up in the first column of the table.
table is a range reference or named range of the table in which you want to look.
col_number is the number of the column from which a matching value will be returned.
range_lookup is a logical value. If TRUE, or omitted, will return the next lowest value under the **lookup_value** if **lookup_value** doesn't match a value exactly. If FALSE, will return the exact match of the **lookup_value**.

Glossary of Functions

ABS:

Description: Using the ABS function returns the absolute value of a number.

General Syntax: **=ABS(number)**

Arguments: **number** is the number for which you want the absolute value.

ACOS:

Description: Using the ACOS function returns the arccosine, or inverse cosine, of a number.

General Syntax: **=ACOS(number)**

Arguments: **number** is the cosine of the angle that you want and must be from -1 to 1.

ASIN:

Description: Using the ASIN function returns the arcsine, or inverse sine, of a number.

General Syntax: **=ASIN(number)**

Arguments: **number** is the sine of the angle you want and must be from -1 to 1.

ASINH:

Description: Using the ASINH function returns the inverse hyperbolic sine of a number.

General Syntax: **=ASINH(number)**

Arguments: **number** is any real number.

ATAN:

Description: Using the ATAN function returns the arctangent, or inverse tangent, of a number.

General Syntax: **=ATAN(number)**

Arguments: **number** is the tangent of the angle that you want.

ATAN2:

Description: Using the ATAN2 function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates. The arctangent is the angle from the *x* axis to a line containing the origin (0,0) and a point with coordinates (**x_num,y_num**).

General Syntax: **=ATAN2(x_num,y_num)**

Arguments: **x_num** is the x-coordinate of the point.
y_num is the y-coordinate of the point.

Glossary of Functions

ATANH:

Description: Using the ATANH function returns the inverse hyperbolic tangent of a number.

General Syntax: **=ATANH(number)**

Arguments: **number** is any real number between 1 and -1.

CEILING:

Description: Using the CEILING function returns the number rounded up, away from zero, to the nearest multiple of significance.

General Syntax: **=CEILING(number,significance)**

Arguments: **number** is the number that you wish to round.
significance is the multiple to which you wish to round.

COMBIN:

Description: Using the COMBIN function returns the number of combinations for a given number of items. Use COMBIN to determine the total possible number of groups for a given number of items.

General Syntax: **=COMBIN(number,number_chosen)**

Arguments: **number** is the number of items.
number_chosen is the number of items per combination.

COS:

Description: Using the COS function returns the cosine of a given angle.

General Syntax: **=COS(number)**

Arguments: **number** is the angle (in radians) for which you want the cosine. If the angle is in degrees, you can multiply the angle by PI()/180 or use the RADIANS function to return the radians of the angle before using the COS function.

COSH:

Description: Using the COSH function returns the hyperbolic cosine of a number.

General Syntax: **=COS(number)**

Arguments: **number** is any real number for which you want to find the hyperbolic cosine.

DEGREES:

Description: Using the DEGREES function converts an angle given in radians to degrees.

General Syntax: **=DEGREES(angle)**

Arguments: **angle** is the angle (in radians) that you want to convert.

Glossary of Functions

EVEN:

Description: Using the EVEN function returns a number rounded up to the nearest even integer.

General Syntax: **=EVEN(number)**

Arguments: **number** is the number that you want to round.

EXP:

Description: Using the EXP function returns *e* (the base of the natural logarithm) raised to the specified power.

General Syntax: **=EXP(number)**

Arguments: **number** is the power to which you want to raise the value *e*.

FACT:

Description: Using the FACT function returns the factorial of a number. The factorial is equal to 1*2*3*…***number**. Negative numbers will cause an error.

General Syntax: **=FACT(number)**

Arguments: **number** is the number for which you want the factorial.

FACTDOUBLE:

Description: Using the FACTDOUBLE function returns the double factorial of a number. Negative numbers will cause an error.

General Syntax: **=FACTDOUBLE(number)**

Arguments: **number** is the number for which you want the double factorial.

FLOOR:

Description: Using the FLOOR function returns the number rounded down, towards zero, to the nearest multiple of significance.

General Syntax: **=FLOOR(number,significance)**

Arguments: **number** is the number that you wish to round.
significance is the multiple to which you wish to round.

GCD:

Description: Using the GCD function returns the greatest common divisor of two or more integers. The greatest common divisor is the largest integer that divides the given numbers without a remainder.

General Syntax: **=GCD(numbers)**

Arguments: **numbers** are 1 to 29 comma-separated numbers for which you want the GCD.

Glossary of Functions

INT:

Description: Using the INT function rounds the given number down to the nearest integer.

General Syntax: **=INT(number)**

Arguments: **number** is the real number you want to round down to an integer.

LCM:

Description: Using the LCM function returns the least common multiple of integers. The least common multiple is the smallest positive integer that is a multiple of all integer arguments given.

General Syntax: **=LCM(numbers)**

Arguments: **numbers** are 1 to 29 comma-separated numbers for which you want the LCM.

LN:

Description: Using the LN function returns the natural logarithm of a number. Natural logarithms are based on the constant e (2.71828182845904).

General Syntax: **=LN(number)**

Arguments: **number** is the positive real number for which you want the natural logarithm.

LOG:

Description: Using the LOG function returns the logarithm of a number to the base you specify.

General Syntax: **=LOG(number**,*base***)**

Arguments: **number** is the positive real number for which you want the logarithm.
base is the base of the logarithm. If base is omitted, it is assumed to be 10.

MDETERM:

Description: Using the MDETERM function returns the matrix determinant of an array.

General Syntax: **=MDETERM(array)**

Arguments: **array** is the numeric array that contains an equal number of columns and rows.

MINVERSE:

Description: Using the MINVERSE function returns the inverse matrix for the matrix stored in an array.

General Syntax: **=MINVERSE(array)**

Arguments: **array** is the numeric array that contains an equal number of columns and rows.

MMULT:

Description: Using the MMULT function returns the matrix product of two arrays. The result is an array with the same number of rows as **array1** and the same number of columns as **array2**.

General Syntax: **=MMULT(array1,array2)**

Arguments: **array1,array2** are the arrays that you wish to multiply.

MOD:

Description: Using the MOD function returns the remainder after **number** is divided by **divisor**. The result has the same sign as **divisor**.

General Syntax: **=MOD(number,divisor)**

Arguments: **number** is the number to divide by the **divisor**.
divisor is the number by which you want to divide the **number**.

MROUND:

Description: Using the MROUND function returns a number rounded to the desired multiple.

General Syntax: **=MROUND(number,multiple)**

Arguments: **number** is the value to round.
multiple is the multiple to which you want to round the **number**.

MULTINOMIAL:

Description: Using the MULTINOMIAL function returns the ratio of the factorial of a sum of values to the product of factorials.

General Syntax: **=MULTINOMIAL(numbers)**

Arguments: **numbers** are 1 to 29 comma-separated values for which you want the multinomial.

ODD:

Description: Using the ODD function returns the **number** rounded up to the nearest odd integer.

General Syntax: **=ODD(number)**

Arguments: **number** is the number to round.

PI:

Description: Using the PI function returns the number 3.14159265358979, the mathematical constant pi, accurate to 15 digits.

General Syntax: **=PI()**

Arguments: none.

Glossary of Functions

POWER:

Description: Using the POWER function returns the result of a number raised to a power.

General Syntax: **=POWER(number,power)**

Arguments: **number** is the base number that you want to raise to the specified **power**.
power is the exponent to which the **number** is raised.

PRODUCT:

Description: Using the PRODUCT function multiplies all **numbers** and returns the product.

General Syntax: **=PRODUCT(numbers)**

Arguments: **numbers** are a comma-separated list of 1 to 30 values that you want to multiply.

QUOTIENT:

Description: Using the QUOTIENT function returns the integer portion of a division. Use this function when you want to discard the remainder of a division.

General Syntax: **=QUOTIENT(numerator,demonimator)**

Arguments: **numerator** is the number that you want to divide.
denominator is the number by which you want to divide the **numerator**.

RADIANS:

Description: Using the RADIANS function converts an angle (in degrees) to radians.

General Syntax: **=RADIANS(angle)**

Arguments: **angle** is the number (in degrees) of the angle that you want to convert.

RAND:

Description: Using the RAND function returns an evenly distributed random number greater than or equal to 0 and less than 1. A new random number is returned every time the worksheet is calculated.

General Syntax: **=RAND()**

Arguments: none.

RANDBETWEEN:

Description: Using the RANDBETWEEN function returns a random number between the numbers you specify. A new random number is returned every time the worksheet is calculated.

General Syntax: **=RANDBETWEEN(bottom,top)**

Arguments: **bottom** is the lowest number possibly generated.
top is the highest number possibly generated.

ROMAN:

Description: Using the ROMAN function converts an Arabic numeral to Roman, as a text value.

General Syntax: **=ROMAN(number**,*form***)**

Arguments: **Number** is the Arabic number that you want to convert.
form is an optional number argument that, if omitted, assumes 0, and provides the "Classic" Roman number format. You may use the numbers 1 to 4 for a more concise expression of the value, if needed.

ROUND:

Description: Using the ROUND function rounds a number to a specified number of digits.

General Syntax: **=ROUND(number,decimals)**

Arguments: **number** is the number to round.
decimals is the number of digits to display after the decimal point. If negative, rounds to the left of the decimal point.

ROUNDDOWN:

Description: Using the ROUNDDOWN function rounds a number "down," or towards zero.

General Syntax: **=ROUNDDOWN(number,decimals)**

Arguments: **number** is the number to round.
decimals is the number of digits to display after the decimal point. If negative, rounds to the left of the decimal point.

ROUNDUP:

Description: Using the ROUNDUP function rounds a number "up," or away from zero.

General Syntax: **=ROUNDUP(number,decimals)**

Arguments: **number** is the number to round.
decimals is the number of digits to display after the decimal point. If negative, rounds to the left of the decimal point.

SERIESSUM:

Description: The SERIESSUM function returns the sum of a power series based on the following:

$$\mathrm{SERIES}(x,n,m,a) = a_1 x^{n} + a_2 x^{(n+m)} + a_3 x^{(n+2m)} + \ldots + a_i x^{(n+(i-1)m)}$$

General Syntax: **=SERIESSUM(x,n,m,coefficients)**

Arguments: **x** is the input value to the power series.
n is the initial power to which you want to raise **x**.
m is the step by which to increase **n** for each term in the series.
coefficients is the set of coefficients by which each successive power of **x** is multiplied. The number of values in **coefficients** determines the number of terms in the power series.

SIGN:

Description: Using the SIGN function returns a 1 if the **number** is positive, a 0 if the **number** is zero, and a -1 if the **number** is negative.

General Syntax: **=SIGN(number)**

Arguments: **number** is the number for which you want the sign.

SIN:

Description: Using the SIN function returns the sine of a given angle.

General Syntax: **=SIN(angle)**

Arguments: **angle** is the number (in radians) of the angle for which you want the sine.

SINH:

Description: Using the SINH function returns the hyperbolic sine of a number.

General Syntax: **=SINH(number)**

Arguments: **number** is any real number.

SQRT:

Description: Using the SQRT function returns a positive square root.

General Syntax: **=SQRT(number)**

Arguments: **number** is the number for which you want the square root.

SQRTPI:

Description: Using the SQRTPI function returns the square root of (**number** * pi).

General Syntax: **=SQRTPI(number)**

Arguments: **number** is the number by which pi is multiplied.

SUBTOTAL:

Description: Using the SUBTOTAL function returns a subtotal in a table or database.

General Syntax: **=SUBTOTAL(function_num,refs)**

Arguments: **function_num** is a number that determines which function to use and whether or not the function should include or ignore hidden values in a table.

function	includes hidden	excludes hidden
AVERAGE	1	101
COUNT	2	102
COUNTA	3	103
MAX	4	104
MIN	5	105
PRODUCT	6	106
STDEV	7	107
STDEVP	8	108
SUM	9	109
VAR	10	110
VARP	11	111

refs is a comma-separated listing of 1 to 29 ranges or references for which you want the subtotal. Note that if you have nested subtotals in these ranges, they are ignored by this function to avoid duplication of values.

SUM:

Description: Using the SUM function adds the numbers in a range of cells.

General Syntax: **=SUM(numbers)**

Arguments: **numbers** is a comma-separated list of numbers or references which you want to add.

SUMIF:

Description: Using the SUMIF function adds the numbers in cells that match a specified criteria.

General Syntax: **=SUMIF(range,criteria**,*sum_range***)**

Arguments: **range** is the range of cells which you want to match against the **criteria**.
criteria is the criteria against which you wish to match the values in the **range**.
sum_range is the range of cells to add.

SUMPRODUCT:

Description: Using the SUMPRODUCT function multiplies corresponding components in the given arrays, and returns the sum of those products.

General Syntax: **=SUMPRODUCT(array1,array2,…)**

Arguments: **array1,array2…** is the list of arrays whose components you wish to multiply and then add together.

SUMSQ:

Description: Using the SUMSQ function returns the sum of the squares of the arguments.

General Syntax: **=SUMSQ(numbers)**

Arguments: **numbers** is a comma-separated list of 1 to 30 arguments or an array for which you want the sum of the squares.

SUMX2MY2:

Description: Using the SUMX2MY2 function returns the sum of the difference of squares of corresponding values in two arrays.

General Syntax: **=SUMX2MY2(array1,array2)**

Arguments: **array1** is the first array or range of values.
array2 is the second array or range of values.

SUMX2PY2:

Description: Using the SUMX2PY2 function returns the sum of the sum of squares of corresponding values in two arrays. The sum of the sum of squares is a common term in many statistical calculations.

General Syntax: **=SUMX2PY2(array1,array2)**

Arguments: **array1** is the first array or range of values.
array2 is the second array or range of values.

SUMXMY2:

Description: Using the SUMXMY2 function returns the sum of squares of differences of corresponding values in two arrays.

General Syntax: **=SUMXMY2(array1,array2)**

Arguments: **array1** is the first array or range of values.
array2 is the second array or range of values.

TAN:

Description: Using the TAN function returns the tangent of a given angle.

General Syntax: **=TAN(angle)**

Arguments: **angle** is the number (in radians) of the angle for which you want the tangent.

TANH:

Description: Using the TANH function returns the hyperbolic tangent of a number.

General Syntax: **=TANH(number)**

Arguments: **number** is any real number.

TRUNC:

Description: Using the TRUNC function truncates a number to an integer by removing the fractional part of the number.

General Syntax: **=TRUNC(number**,*decimals***)**

Arguments: **number** is the number that you want to truncate.
decimals is the number that specifies the precision of the truncation. If omitted, is 0.

Glossary of Functions

AVEDEV:

Description: Using the AVERAGE function returns the average of the absolute deviations of data points from their mean.

General Syntax: **=AVEDEV(numbers)**

Arguments: **numbers** is a comma-separated list of 1 to 30 arguments for which you want the average of the absolute deviations.

AVERAGE:

Description: Using the AVERAGE function returns the returns the average (arithmetic mean) of the arguments.

General Syntax: **=AVERAGE(numbers)**
Arguments: **numbers** is a comma-separated list of 1 to 30 arguments for which you want the average.

AVERAGEA:

Description: Using the AVERAGEA function returns the average (arithmetic mean) of the values in the list of arguments. In addition to numbers, text and logical values such as TRUE and FALSE are included in the calculation.

General Syntax: **=AVERAGEA(numbers)**
Arguments: **numbers** is a comma-separated list of 1 to 30 arguments for which you want the average.

BETADIST:

Description: Using the BETADIST function returns the beta cumulative distribution function.

General Syntax: **=BETADIST(x,alpha,beta***,a,b***)**

Arguments: **x** is the value between **a** and **b** at which to evaluate the function. **alpha** is a parameter of the distribution.
beta is a parameter of the distribution. *a* is an optional lower bound of the interval **x**.
b is an optional upper bound of the interval **x**.

BETAINV:

Description: Using the BETAINV function returns the inverse of the cumulative distribution function for a specified beta distribution. That is, if probability = BETADIST(x,...), then BETAINV(probability,...) = x. The beta distribution can be used in project planning to model probable completion times given an expected completion time and variability.

General Syntax: **=BETAINV(probablity,alpha,beta***,a,b***)**

Arguments: **probability** is a is a probability associated with the beta distribution. **alpha** is a parameter of the distribution.
beta is a parameter of the distribution. *a* is an optional lower bound of the interval **x**.
b is an optional upper bound of the interval **x**.

BINOMDIST:

Description: Using the BINOMDIST function returns the individual term binomial distribution probability. Use BINOMDIST in problems with a fixed number of tests or trials, when the outcomes of any trial are only success or failure, when trials are independent, and when the probability of success is constant throughout the experiment.

General Syntax: **=BINOMDIST(success,trials,probability,cumulative)**

Arguments: **success** is the number of successes in **trials**. **trials** is the number of independent trials.
probability is the probability of success on each trial.
cumulative is a logical value that determines the form of the function. If cumulative is TRUE, then BINOMDIST returns the cumulative distribution function, which is the probability that there are at most **number** successes; if FALSE, it returns the probability mass function, which is the probability that there are **number** successes.

Glossary of Functions

CHIDIST:

Description: Using the CHIDIST function returns the one-tailed probability of the chi-squared distribution. The c2 distribution is associated with a c2 test. Use the c2 test to compare observed and expected values.

General Syntax: **=CHIDIST(x,degrees)**

Arguments: **x** is the values at which you want to evaluate the distribution.
degrees is the number of degrees of freedom to use.

CHIINV:

Description: Using the CHIINV function returns the inverse of the one-tailed probability of the chi-squared distribution. If probability = CHIDIST(x,...), then CHIINV(probability,...) = x. Use this function to compare observed results with expected ones in order to decide whether your original hypothesis is valid.

General Syntax: **=CHIINV(x,degrees)**

Arguments: **x** is a probability associated with the chi-squared distribution.
degrees is the number of degrees of freedom to use.

CHITEST:

Description: Using the CHITEST function returns the test for independence. CHITEST returns the value from the chi-squared (c2) distribution for the statistic and the appropriate degrees of freedom. You can use c2 tests to determine whether hypothesized results are verified by an experiment.

General Syntax: **=CHITEST(actual_range,expected_range)**

Arguments: **actual_range** is the range of data that contains observations to test against expected values.
expected_range is the range of data that contains the ratio of the product of row totals and column totals to the grand total.

CONFIDENCE:

Description: Using the CONFIDENCE function returns a value that you can use to construct a confidence interval for a population mean. The confidence interval is a range of values. Your sample mean, x, is at the center of this range and the range is x ± CONFIDENCE.

General Syntax: **=CONFIDENCE(alpha,st_dev,size)**

Arguments: **alpha** is the significance level used to compute the confidence level.
st_dev is the population standard deviation for the data range and is assumed to be known.
size is the sample size.

CORREL:

Description: Using the CORREL function returns the correlation coefficient of the **array1** and **array2** cell ranges. Use the correlation coefficient to determine the relationship between two properties.

General Syntax: **=CORREL(array1,array2)**

Arguments: **array1** is the first array or range of values.
array2 is the second array or range of values.

Glossary of Functions

COUNT:

Description: Using the COUNT function counts the number of cells that contain numbers and also numbers within the list of arguments.

General Syntax: **=COUNT(numbers)**

Arguments: **numbers** is a comma-separated list of 1 to 30 arguments.

COUNTA:

Description: Using the COUNTA function counts the number of cells that are not empty and the values within the list of arguments. Use COUNTA to count the number of cells that contain data in a range or array.

General Syntax: **=COUNTA(numbers)**

Arguments: **numbers** are 1 to 30 arguments representing the values you want to count. In this case, a value is any type of information, including empty text ("") but not including empty cells.

COUNTBLANK:

Description: Using the COUNTBLANK function counts empty cells in a specified range of cells.

General Syntax: **=COUNTBLANK(range)**

Arguments: **range** is the range in which you want to count the blank cells.

COUNTIF:

Description: Using the COUNTIF function counts the number of cells within a range that meet the given criteria.

General Syntax: **=COUNTIF(range,criteria)**

Arguments: **range** is the range in which you want to count the blank cells.
criteria is the criteria that defines the cells to be counted.

COVAR:

Description: Using the COVAR function returns covariance, the average of the products of deviations for each data point pair. Use covariance to determine the relationship between two data sets.

General Syntax: **=COVAR(array1,array2)**

Arguments: **array1** is the first cell range of integers.
array2 is the second cell range of integers.

CRITBINOM:

Description: Using the CRITBINOM function returns the smallest value for which the cumulative binomial distribution is greater than or equal to a criterion value. Use this function for quality assurance applications.

General Syntax: **=CRITBINOM(trials,s_prob,alpha)**

Arguments: **trials** is the number of Bernoulli trials.
s_prob is the probability of a success on each trial.
alpha is the criterion value.

DEVSQ:

Description: Using the DEVSQ function returns the sum of squares of deviations of data points from their sample mean.

General Syntax: **=DEVSQ(numbers)**

Arguments: **numbers** are 1 to 30 comma-separated arguments for which you want to calculate the sum of squared deviations. You can also use a single array instead.

EXPONDIST:

Description: Using the EXPONDIST function returns the exponential distribution.

General Syntax: **=EXPONDIST(x,lambda,cumulative)**

Arguments: **x** is the value of the function.
lambda is the parameter value.
cumulative is a logical value that indicates which form of the exponential function to provide. If cumulative is TRUE, it returns the cumulative distribution function; if FALSE, it returns the probability density function.

FDIST:

Description: Using the FDIST function returns the F probability distribution. You can use this function to determine whether two data sets have different degrees of diversity.

General Syntax: **=FDIST(x,f_degrees1,f_degrees2)**

Arguments: **x** is the value at which the function is evaluated.
f_degrees1 is the numerator's degrees of freedom.
f_degrees2 is the denominator's degrees of freedom.

FINV:

Description: Using the FINV function returns the inverse of the F probability distribution. If p = FDIST(x,...), then FINV(p,...) = x. The F distribution can be used in an F-test that compares the degree of variability in two data sets

.

General Syntax: **=FINV(probability,f_degrees1,f_degrees2)**

Arguments: **probability** is a probability associated with the F cumulative distribution.
f_degrees1 is the numerator's degrees of freedom.
f_degrees2 is the denominator's degrees of freedom.

FISHER:

Description: Using the FISHER function returns the Fisher transformation at **x**. This transformation produces a function that is normally distributed rather than skewed. Use this function to perform hypothesis testing on the correlation coefficient.

General Syntax: **=FISHER(x)**

Arguments: **x** is a numeric value for which you want the transformation.

FISHERINV:

Description: Using the FISHERINV function returns the inverse of the Fisher transformation. Use this transformation when analyzing correlations between ranges or arrays of data.

General Syntax: **=FISHERINV(y)**

Arguments: **y** is the value for which you want to perform the inverse of the transformation.

FORECAST:

Description: Using the FORECAST function calculates, or predicts, a future value by using existing values. The predicted value is a y-value for a given x-value. The known values are existing x-values and y-values, and the new value is predicted by using linear regression. You can use this function to predict future sales, inventory requirements, or consumer trends.

General Syntax: **=FORECAST(x,known_y,known_x)**

Arguments: **x** is the data point for which you want to predict a value.
known_y is the dependent array or range of data.
known_x is the independent array or range of data.

FREQUENCY:

Description: Using the FREQUENCY function calculates how often values occur within a range of values, and then returns a vertical array of numbers.

General Syntax: **=FREQUENCY(data_array,bins_array)**

Arguments: **data_array** is an array of or reference to a set of values for which you want to count frequencies.
bins_array is an array of or reference to intervals into which you want to group the values in **data_array**.

FTEST:

Description: Using the FTEST function returns the result of an F-test. An F-test returns the one-tailed probability that the variances in array1 and array2 are not significantly different. Use this function to determine whether two samples have different variances.

General Syntax: **=FTEST(array1,array2)**

Arguments: **array1** is the first array or range of data.
array2 is the second array or range of data.

GAMMADIST:

Description: Using the GAMMADIST function returns the gamma distribution. You can use this function to study variables that may have a skewed distribution. The gamma distribution is commonly used in queuing analysis.

General Syntax: **=GAMMADIST(x,alpha,beta,cumulative)**

Arguments: **x** is the value at which to evaluate the function.
alpha is a parameter to the distribution.
beta is a parameter to the distribution. If beta = 1, it returns the standard gamma distribution.
cumulative is a logical value that determines the form of the function. If cumulative is TRUE, it returns the cumulative distribution function; if FALSE, it returns the probability density function.

GAMMAINV:

Description: Using the GAMMAINV function returns the inverse of the gamma cumulative distribution.

General Syntax: **=GAMMAINV(x,alpha,beta,cumulative)**

Arguments: **probability** is the probability associated with the gamma distribution.
alpha is a parameter to the distribution.
beta is a parameter to the distribution. If beta = 1, it returns the standard gamma distribution.

GAMMALN:

Description: Using the GAMMALN function returns the natural logarithm of the gamma function, Γ(x).

General Syntax: **=GAMMALN(x)**

Arguments: **x** is the value for which you want to calculate GAMMALN

GEOMEAN:

Description: Using the GEOMEAN function returns the geometric mean of an array or range of positive data.

General Syntax: **=GEOMEAN(numbers)**

Arguments: **numbers** is a comma-separated list of numbers for which you want to calculate the mean. You can also use an array reference, if desired.

GROWTH:

Description: Using the GROWTH function calculates predicted exponential growth by using existing data. GROWTH returns the y-values for a series of new x-values that you specify by using existing x-values and y-values. You can also use the GROWTH worksheet function to fit an exponential curve to existing x-values and y-values.

General Syntax: **=GROWTH(known_y's**,*known_x's,new_x's,const***)**

Arguments: **known_y's** is the set of y-values you already know in the relationship y = b*m^x. If the array **known_y's** is in a single column, then each column of *known_x's* is interpreted as a separate variable. If the array **known_y's** is in a single row, then each row of *known_x's* is interpreted as a separate variable. If any of the numbers in **known_y's** is 0 or negative, GROWTH returns the #NUM! error value.

known_x's is an optional set of x-values that you may already know in the relationship y = b*m^x. The array *known_x's* can include one or more sets of variables. If only one variable is used, **known_y's** and *known_x's* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, **known_y's** must be a vector (that is, a range with a height of one row or a width of one column). If *known_x's* is omitted, it is assumed to be the array {1,2,3,...} that is the same size as **known_y's**.

new_x's are new x-values for which you want GROWTH to return corresponding y-values. *New_x's* must include a column (or row) for each independent variable, just as *known_x's* does. So, if **known_y's** is in a single column, *known_x's* and *new_x's* must have the same number of columns. If **known_y's** is in a single row, *known_x's* and *new_x's* must have the same number of rows. If *new_x's* is omitted, it is assumed to be the same as *known_x's*. If both *known_x's* and *new_x's* are omitted, they are assumed to be the array {1,2,3,...} that is the same size as **known_y's**.

const is a logical value specifying whether to force the constant b to equal 1. If *const* is TRUE or omitted, b is calculated normally. If *const* is FALSE, b is set equal to 1 and the m-values are adjusted so that y = m^x.

Glossary of Functions

HARMEAN:

Description: Using the HARMEAN function returns the harmonic mean of a data set. The harmonic mean is the reciprocal of the arithmetic mean of reciprocals.

General Syntax: **=HARMEAN(numbers)**

Arguments: **numbers** is a comma-separated list of numbers for which you want to calculate the mean. You can also use an array reference, if desired.

HYPGEOMDIST:

Description: Using the HYPGEOMDIST function returns the hypergeometric distribution. HYPGEOMDIST returns the probability of a given number of sample successes, given the sample size, population successes, and population size. Use HYPGEOMDIST for problems with a finite population, where each observation is either a success or a failure, and where each subset of a given size is chosen with equal likelihood.

General Syntax: **=HYPGEOMDIST(sample_s,number_sample,population_s,number_population)**

Arguments: **sample_s** is the number of successes in the sample.
number_sample is the size of the sample.
population_s is the number of successes in the population.
number_population is the population size.

INTERCEPT:

Description: Using the INTERCEPT function calculates the point at which a line will intersect the y-axis by using existing x-values and y-values. The intercept point is based on a best-fit regression line plotted through the known x-values and known y-values. Use the INTERCEPT function when you want to determine the value of the dependent variable when the independent variable is 0 (zero).

General Syntax: **=INTERCEPT(known_y's,known_x's)**

Arguments: **known_y's** is the dependent set of observations or data.
known_x's is the independent set of observations or data.

KURT:

Description: Using the KURT function returns the kurtosis of a data set. Kurtosis characterizes the relative peakedness or flatness of a distribution compared with the normal distribution. Positive kurtosis indicates a relatively peaked distribution. Negative kurtosis indicates a relatively flat distribution.

General Syntax: **=KURT(numbers)**

Arguments: **numbers** is a comma-separated list of numbers for which you want to calculate the mean. You can also use an array reference, if desired.

LARGE:

Description: Using the LARGE function Returns the **k**-th largest value in a data set. You can use this function to select a value based on its relative standing. For example, you can use LARGE to return the highest, runner-up, or third-place score.

General Syntax: **=LARGE(array,k)**

Arguments: **array** is the array or range of data for which you want to determine the **k**-th largest value.
k is the position (from the largest) in the array or cell range of data to return.

LINEST:

Description:

Using the LINEST function calculates the statistics for a line by using the "least squares" method to calculate a straight line that best fits your data, and returns an array that describes the line. Because this function returns an array of values, it must be entered as an array formula. The equation for the line is: y = mx + b or y = m1x1 + m2x2 + ... + b (if there are multiple ranges of x-values) where the dependent y-value is a function of the independent x-values. The m-values are coefficients corresponding to each x-value, and b is a constant value. Note that y, x, and m can be vectors. The array that LINEST returns is {mn,mn-1,...,m1,b}. LINEST can also return additional regression statistics.

General Syntax:

=LINEST(known_y's,*known_x's,const,stats***)**

Arguments:

known_y's is the set of y-values you already know in the relationship y = mx + b. If the array is in a single column, then each column of *known_x's* is interpreted as a separate variable. If the array is in a single row, then each row of *known_x's* is interpreted as a separate variable.

known_x's is an optional set of x-values that you may already know in the relationship y = mx + b. The array can include one or more sets of variables. If only one variable is used, **known_y's** and *known_x's* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, **known_y's** must be a vector (that is, a range with a height of one row or a width of one column). If *known_x's* is omitted, it is assumed to be the array {1,2,3,...} that is the same size as **known_y's**.

const is a logical value specifying whether to force the constant b to equal 0. If *const* is TRUE or omitted, b is calculated normally. If *const* is FALSE, b is set equal to 0 and the m-values are adjusted to fit y = mx.

stats is a logical value specifying whether to return additional regression statistics. If *stats* is TRUE, LINEST returns the additional regression statistics, so the returned array is {mn,mn-1,...,m1,b;sen,sen-1,...,se1,seb;r2,sey;F,df;ssreg,ssresid}. If *stats* is FALSE or omitted, LINEST returns only the m-coefficients and the constant b. The additional regression statistics are as follows:

Statistic:	**Description:**
se1,se2,...,sen	The standard error values for the coefficients m1,m2,...,mn.
seb	The standard error value for the constant b (seb = #N/A when const is FALSE).
r2	The coefficient of determination. Compares estimated and actual y-values, and ranges in value from 0 to 1. If it is 1, there is a perfect correlation in the sample— there is no difference between the estimated y-value and the actual y-value. At the other extreme, if the coefficient of determination is 0, the regression equation is not helpful in predicting a y-value.
sey	The standard error for the y estimate.
F	The F statistic, or the F-observed value. Use the F statistic to determine whether the observed relationship between the dependent and independent variables occurs by chance.
df	The degrees of freedom. Use the degrees of freedom to help you find F-critical values in a statistical table. Compare the values you find in the table to the F statistic returned by LINEST to determine a confidence level for the model.
ssreg	The regression sum of squares.
ssresid	The residual sum of squares

LOGEST:

Description: Using the LOGEST function calculates an exponential curve that fits your data and returns an array of values that describes the curve. Because this function returns an array of values, it must be entered as an array formula. The equation for the curve is: y = b*m^x or y = (b*(m1^x1)*(m2^x2)*_) (if there are multiple x-values) where the dependent y-value is a function of the independent x-values. The m-values are bases corresponding to each exponent x-value, and b is a constant value. Note that y, x, and m can be vectors. The array that LOGEST returns is {mn,mn-1,...,m1,b}.

General Syntax: **=LOGEST(known_y's,***known_x's,const,stats***)**

Arguments:

known_y's is the set of y-values you already know in the relationship y = mx + b. If the array is in a single column, then each column of *known_x's* is interpreted as a separate variable. If the array is in a single row, then each row of *known_x's* is interpreted as a separate variable.

known_x's is an optional set of x-values that you may already know in the relationship y = mx + b. The array can include one or more sets of variables. If only one variable is used, **known_y's** and *known_x's* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, **known_y's** must be a vector (that is, a range with a height of one row or a width of one column). If *known_x's* is omitted, it is assumed to be the array {1,2,3,...} that is the same size as **known_y's**.

const is a logical value specifying whether to force the constant b to equal 1. If *const* is TRUE or omitted, b is calculated normally. If *const* is FALSE, b is set equal to 1 and the m-values are adjusted to fit y = m^x.

stats is a logical value specifying whether to return additional regression statistics. If *stats* is TRUE, LINEST returns the additional regression statistics, so the returned array is {mn,mn-1,...,m1,b;sen,sen-1,...,se1,seb;r2,sey;F,df;ssreg,ssresid}. If *stats* is FALSE or omitted, LINEST returns only the m-coefficients and the constant b. The additional regression statistics are as follows:

Statistic:	**Description:**
se1,se2,...,sen	The standard error values for the coefficients m1,m2,...,mn.
seb	The standard error value for the constant b (seb = #N/A when const is FALSE).
r2	The coefficient of determination. Compares estimated and actual y-values, and ranges in value from 0 to 1. If it is 1, there is a perfect correlation in the sample— there is no difference between the estimated y-value and the actual y-value. At the other extreme, if the coefficient of determination is 0, the regression equation is not helpful in predicting a y-value.
sey	The standard error for the y estimate.
F	The F statistic, or the F-observed value. Use the F statistic to determine whether the observed relationship between the dependent and independent variables occurs by chance.
df	The degrees of freedom. Use the degrees of freedom to help you find F-critical values in a statistical table. Compare the values you find in the table to the F statistic returned by LINEST to determine a confidence level for the model.
ssreg	The regression sum of squares.
ssresid	The residual sum of squares

LOGINV:

Description: Using the LOGINV function returns the inverse of the lognormal cumulative distribution function of x, where ln(x) is normally distributed with parameters mean and standard_dev. If p = LOGNORMDIST(x,...) then LOGINV(p,...) = x. Use the lognormal distribution to analyze logarithmically transformed data.

General Syntax: **=LOGINV(probability,mean,standard_dev)**

Arguments:

probability is a probability associated with the lognormal distribution.
mean is the mean of ln(x).
standard_dev is the standard deviation of ln(x).

LOGNORMDIST:

Description: Using the LOGNORMDIST function returns the cumulative lognormal distribution of x, where ln(x) is normally distributed with parameters mean and standard_dev. Use this function to analyze data that has been logarithmically transformed.

General Syntax: **=LOGNORMDIST(***x*,**mean,standard_dev)**

Arguments: *x* is the value at which you want to evaluate the function.
mean is the mean of ln(x).
standard_dev is the standard deviation of ln(x).

MAX:

Description: Using the MAX function returns the largest value in a set of values.

General Syntax: **=MAX(numbers)**

Arguments: **numbers** is a comma-separated list of 1 to 30 numbers for which you want to find the highest value.

MAXA:

Description: Using the MAXA function returns the largest value in a list of arguments. Text and logical values such as TRUE and FALSE are compared as well as numbers.

General Syntax: **=MAXA(values)**

Arguments: **values** is a comma-separated list of 1 to 30 values for which you want to find the largest value.

MEDIAN:

Description: Using the MEDIAN function returns the median of the given numbers. The median is the number in the middle of a set of numbers; that is, half the numbers have values that are greater than the median, and half have values that are less.

General Syntax: **=MEDIAN(numbers)**

Arguments: **numbers** is a comma-separated list of 1 to 30 numbers for which you want to find the median.

MIN:

Description: Using the MIN function returns the smallest value in a set of values.

General Syntax: **=MIN(numbers)**

Arguments: **numbers** is a comma-separated list of 1 to 30 numbers for which you want to find the lowest value.

MINA:

Description: Using the MINA function returns the smallest value in a list of arguments. Text and logical values such as TRUE and FALSE are compared as well as numbers.

General Syntax: **=MINA(values)**

Arguments: **values** is a comma-separated list of 1 to 30 values for which you want to find the smallest value.

MODE:

Description: Using the MODE function returns the most frequently occurring, or repetitive, value in an array or range of data.

General Syntax: **=MODE(numbers)**

Arguments: **numbers** is a comma-separated list of 1 to 30 numbers for which you want the mode.

NEGBINOMDIST:

Description: Using the NEGBINOMDIST function returns the negative binomial distribution. NEGBINOMDIST returns the probability that there will be **number_f** failures before the **number_s**-th success, when the constant probability of a success is **probability_s**. This function is similar to the binomial distribution, except that the number of successes is fixed, and the number of trials is variable. Like the binomial, trials are assumed to be independent.

General Syntax: **=NEGBINOMDIST(number_f,number_s,probability_s)**

Arguments: **number_f** is the number of failures.
number_s is the threshold number of successes.
probability_s is the probability of a success.

NORMDIST:

Description: Using the NORMDIST function returns the normal distribution for the specified mean and standard deviation. This function has a very wide range of applications in statistics, including hypothesis testing.

General Syntax: **=NORMDIST(x,mean,standard_dev,cumulative)**

Arguments: **x** is the value for which you want the distribution.
mean is the arithmetic mean of the distribution.
standard_dev is the standard deviation of the distribution.
cumulative is a logical value that determines the form of the function. If cumulative is TRUE, NORMDIST returns the cumulative distribution function; if FALSE, it returns the probability mass function.

NORMINV:

Description: Using the NORMINV function returns the inverse of the normal cumulative distribution for the specified mean and standard deviation.

General Syntax: **=NORMINV(probability,mean,standard_dev)**

Arguments: **probability** is a probability corresponding to the normal distribution.
mean is the arithmetic mean of the distribution.
standard_dev is the standard deviation of the distribution.

NORMSDIST:

Description: Using the NORMSDIST function returns the standard normal cumulative distribution function. The distribution has a mean of 0 (zero) and a standard deviation of one. Use this function in place of a table of standard normal curve areas.

General Syntax: **=NORMSDIST(z)**

Arguments: **z** is the value for which you want the distribution.

NORMSINV:

Description: Using the NORMSINV function returns the inverse of the standard normal cumulative distribution. The distribution has a mean of zero and a standard deviation of one.

General Syntax: **=NORMSINV(probability)**

Arguments: **probability** is a probability corresponding to the normal distribution.

PEARSON:

Description: Using the PEARSON function returns the Pearson product moment correlation coefficient, r, a dimensionless index that ranges from -1.0 to 1.0 inclusive and reflects the extent of a linear relationship between two data sets.

General Syntax: **=PEARSON(array1,array2)**

Arguments: **array1** is a set of independent values.
array2 is a set of dependent values

PERCENTILE:

Description: Using the PERCENTILE function returns the **k**-th percentile of values in a range. You can use this function to establish a threshold of acceptance.

General Syntax: **=PERCENTILE(array,k)**

Arguments: **array** is the array or range of data that defines relative standing.
k is the percentile value in the range 0..1, inclusive.

PERCENTRANK:

Description: Using the PERCENTRANK function returns the rank of a value in a data set as a percentage of the data set. This function can be used to evaluate the relative standing of a value within a data set.

General Syntax: **=PERCENTRANK(array,x,significance)**

Arguments: **array** is the array or range of data with numeric values that defines relative standing.
x is the value for which you want to know the rank.
significance is an optional value that identifies the number of significant digits for the returned percentage value. If omitted, it uses three digits (0.xxx).

PERMUT:

Description: Using the PERMUT function returns the number of permutations for a given number of objects that can be selected from number objects. A permutation is any set or subset of objects or events where internal order is significant. Permutations are different from combinations, for which the internal order is not significant. Use this function for lottery-style probability calculations.

General Syntax: **=PERMUT(number,number_chosen)**

Arguments: **number** is an integer that describes the number of objects.
number_chosen is an integer that describes the number of objects in each permutation.

POISSON:

Description: Using the POISSON function returns the Poisson distribution. A common application of the Poisson distribution is predicting the number of events over a specific time, such as the number of cars driving through an intersection in an hour.

General Syntax: **=POISSON(x,mean,cumulative)**

Arguments:
x is the number of events.
mean is the expected numeric value.
cumulative is a logical value that determines the form of the probability distribution returned. If cumulative is TRUE, it returns the cumulative Poisson probability that the number of random events occurring will be between zero and **x** inclusive; if FALSE, it returns the Poisson probability mass function that the number of events occurring will be exactly **x**.

PROB:

Description: Using the PROB function returns the probability that values in a range are between two limits. If *upper_limit* is not supplied, returns the probability that values in **x_range** are equal to **lower_limit**.

General Syntax: **=PROB(x_range,prob_range,lower_limit**,*upper_limit***)**

Arguments:
x_range is the range of numeric values of x with which there are associated probabilities.
prob_range is a set of probabilities associated with values in x_range.
lower_limit is the lower bound on the value for which you want a probability.
upper_limit is the optional upper bound on the value for which you want a probability.

QUARTILE:

Description: Using the QUARTILE function returns the quartile of a data set. Quartiles often are used in sales and survey data to divide populations into groups.

General Syntax: **=QUARTILE(array,quart)**

Arguments:
array is the array or cell range of numeric values for which you want the quartile value.
quart indicates which value to return.

If **quart** equals:	QUARTILE returns:
0	Minimum value
1	First quartile (25th percentile)
2	Median value (50th percentile)
3	Third quartile (75th percentile)
4	Maximum value

RANK:

Description: Using the RANK function returns the rank of a number in a list of numbers. The rank of a number is its size relative to other values in a list.

General Syntax: **=RANK(number,ref**,*order***)**

Arguments:
number is the number whose rank you want to find.
ref is an array of, or a reference to, a list of numbers. Nonnumeric values in ref are ignored.
order is a number specifying how to rank number. If order is 0 (zero) or omitted, Excel ranks number as if **ref** were a list sorted in descending order. If *order* is any nonzero value, Excel ranks number as if **ref** were a list sorted in ascending order.

RSQ:

Description: Using the RSQ function returns the square of the Pearson product moment correlation coefficient through data points in **known_y's** and **known_x's**. The r-squared value can be interpreted as the proportion of the variance in y attributable to the variance in x.

General Syntax: **=RSQ(known_y's,known_x's)**

Arguments: **known_y's** is an array or range of data points.
known_x's is an array or range of data points.

SKEW:

Description: Using the SKEW function returns the skewness of a distribution. Skewness characterizes the degree of asymmetry of a distribution around its mean. Positive skewness indicates a distribution with an asymmetric tail extending toward more positive values. Negative skewness indicates a distribution with an asymmetric tail extending toward more negative values.

General Syntax: **=SKEW(numbers)**

Arguments: **numbers** is a comma-separated list of 1 to 30 arguments for which you want to calculate skewness.

SLOPE:

Description: Using the SLOPE function returns the slope of the linear regression line through data points in **known_y's** and **known_x's**. The slope is the vertical distance divided by the horizontal distance between any two points on the line, which is the rate of change along the regression line.

General Syntax: **=SLOPE(known_y's,known_x's)**

Arguments: **known_y's** is an array or cell range of numeric dependent data points.
known_x's is the set of independent data points.

SMALL:

Description: Using the SMALL function returns the **k**-th smallest value in a data set. Use this function to return values with a particular relative standing in a data set.

General Syntax: **=SMALL(array,k)**

Arguments: **array** is the array or range of data for which you want to determine the **k**-th smallest value.
k is the position (from the smallest) in the array or cell range of data to return.

STANDARDIZE:

Description: Using the STANDARDIZE function returns a normalized value from a distribution characterized by **mean** and **standard_dev**.

General Syntax: **=STANDARDIZE(x,mean,standard_dev)**

Arguments: **x** is the value you want to normalize.
mean is the arithmetic mean of the distribution.
standard_dev is the standard deviation of the distribution.

STDEV:

Description: Using the STDEV function estimates standard deviation based on a sample. The standard deviation is a measure of how widely values are dispersed from the average value (the mean).

General Syntax: **=STDEV(numbers)**

Arguments: **numbers** is a comma-separated list of 1 to 30 number arguments corresponding to a sample of a population. You can also use a single array or a reference, instead.

STDEVA:

Description: Using the STDEVA function estimates standard deviation based on a sample. The standard deviation is a measure of how widely values are dispersed from the average value (the mean). Text and logical values such as TRUE and FALSE are included in the calculation.

General Syntax: **=STDEVA(values)**

Arguments: **values** is a comma-separated list of 1 to 30 values corresponding to a sample of a population. You can also use a single array or a reference, instead.

STDEVP:

Description: Using the STDEVP function calculates standard deviation based on the entire population given as arguments. The standard deviation is a measure of how widely values are dispersed from the average value (the mean).

General Syntax: **=STDEVP(numbers)**

Arguments: **numbers** is a comma-separated list of 1 to 30 number arguments corresponding to an entire population. You can also use a single array or a reference, instead.

STDEVPA:

Description: Using the STDEVP function calculates standard deviation based on the entire population given as arguments. The standard deviation is a measure of how widely values are dispersed from the average value (the mean). Text and logical values such as TRUE and FALSE are included in the calculation.

General Syntax: **=STDEVPA(values)**

Arguments: **values** is a comma-separated list of 1 to 30 values corresponding to an entire population. You can also use a single array or a reference, instead.

STEYX:

Description: Using the STEYX function returns the standard error of the predicted y-value for each x in the regression. The standard error is a measure of the amount of error in the prediction of y for an individual x.

General Syntax: **=STEYX(known_y's,known_x's)**

Arguments: **known_y's** is an array or range of dependent data points.
known_x's is an array or range of independent data points

TDIST:

Description: Using the TDIST function returns the Percentage Points (probability) for the Student t-distribution where a numeric value (x) is a calculated value of t for which the Percentage Points are to be computed. The t-distribution is used in the hypothesis testing of small sample data sets. Use this function in place of a table of critical values for the t-distribution.

General Syntax: **=TDIST(x,degrees_freedom,tails)**

Arguments: **x** is the numeric value at which to evaluate the distribution.
degrees_freedom is an integer indicating the number of degrees of freedom.
tails specifies the number of distribution tails to return. If tails = 1, TDIST returns the one-tailed distribution. If tails = 2, TDIST returns the two-tailed distribution.

TINV:

Description: Using the TINV function returns the t-value of the Student's t-distribution as a function of the probability and the degrees of freedom.

General Syntax: **=TINV(probability,degrees_freedom)**

Arguments: **probability** is the probability associated with the two-tailed Student's t-distribution.
degrees_freedom is the number of degrees of freedom with which to characterize the distribution.

TREND:

Description: Using the TREND function returns values along a linear trend. Fits a straight line (using the method of least squares) to the arrays **known_y's** and *known_x's*. Returns the y-values along that line for the array of new_x's that you specify.

General Syntax: **=TREND(known_y's**,*known_x's,new_x's,const***)**

Arguments: **known_y's** is the set of y-values you already know in the relationship y = mx + b. If the array is in a single column, then each column of *known_x's* is interpreted as a separate variable. If the array is in a single row, then each row of *known_x's* is interpreted as a separate variable.
known_x's is an optional set of x-values that you may already know in the relationship y = mx + b. The array can include one or more sets of variables. If only one variable is used, **known_y's** and *known_x's* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, **known_y's** must be a vector (that is, a range with a height of one row or a width of one column). If *known_x's* is omitted, it is assumed to be the array {1,2,3,...} that is the same size as **known_y's**.
new_x's are new x-values for which you want to return corresponding y-values. It must include a column (or row) for each independent variable, just as *known_x's* does. So, if **known_y's** is in a single column, *known_x's* and *new_x's* must have the same number of columns. If **known_y's** is in a single row, *known_x's* and *new_x's* must have the same number of rows. If you omit *new_x's*, it is assumed to be the same as *known_x's*. If you omit both, they are assumed to be the array {1,2,3,...} that is the same size as **known_y's**.
const is a logical value specifying whether to force the constant b to equal 0. If *const* is TRUE or omitted, b is calculated normally. If *const* is FALSE, b is set equal to 0 (zero), and the m-values are adjusted so that y = mx.

TRIMMEAN:

Description: Using the TRIMMEAN function returns the mean of the interior of a data set. TRIMMEAN calculates the mean taken by excluding a percentage of data points from the top and bottom tails of a data set. You can use this function when you wish to exclude outlying data from your analysis.

General Syntax: **=TRIMMEAN(array,percent)**

Arguments: **array** is the array or range of values to trim and average.
percent is the fractional number of data points to exclude from the calculation. For example, if percent = 0.2, 4 points are trimmed from a data set of 20 points (20 x 0.2): 2 from the top and 2 from the bottom of the set

TTEST:

Description: Using the TTEST function returns the probability associated with a Student's t-Test. Use TTEST to determine whether two samples are likely to have come from the same two underlying populations that have the same mean.

General Syntax: **=TTEST(array1,array2,tails,type)**

Arguments: **array1** is the first data set.
array2 is the second data set.
tails specifies the number of distribution tails. If tails = 1, TTEST uses the one-tailed distribution. If tails = 2, TTEST uses the two-tailed distribution.
type is the kind of t-Test to perform.

Type equals:	Test performed:	Type equals:	Test performed:	Type equals:	Test performed:
1	Paired	2	Two-sample equal variance	3	Two-sample unequal variance

VAR:

Description: Using the VAR function estimates variance based on a sample.

General Syntax: **=VAR(numbers)**

Arguments: **numbers** is a comma-separated list of 1 to 30 numbers corresponding to a sample of a population.

VARA:

Description: Using the VARA function estimates variance based on a sample. In addition to numbers, text and logical values such as TRUE and FALSE are included in the calculation.

General Syntax: **=VARA(values)**

Arguments: **values** is a comma-separated list of 1 to 30 values corresponding to a sample of a population.

VARP:

Description: Using the VARP function calculates variance based on the entire population.

General Syntax: **=VARP(numbers)**

Arguments: **numbers** is a comma-delimited list of 1 to 30 numbers corresponding to a population.

VARPA:

Description: Using the VARPA function calculates variance based on the entire population. In addition to numbers, text and logical values such as TRUE and FALSE are included in the calculation.

General Syntax: **=VARPA(values)**

Arguments: **values** is a comma-separated list of 1 to 30 values corresponding to a population.

WEIBULL:

Description: Using the WEIBULL function returns the Weibull distribution. Use this distribution in reliability analysis, such as calculating a device's mean time to failure.

General Syntax: **=WEIBULL(x,alpha,beta,cumulative)**

Arguments: **x** is the value at which to evaluate the function.
alpha is a parameter to the distribution.
beta is a parameter to the distribution.
cumulative determines the form of the function.

ZTEST:

Description: Using the ZTEST function Returns the one-tailed probability-value of a z-test. For a given hypothesized population mean, μ_0, ZTEST returns the probability that the sample mean would be greater than the average of observations in the data set (array)— that is, the observed sample mean.

General Syntax: **=ZTEST(array**,μ_0,*sigma*)

Arguments: **array** is the array or range of data against which to test μ_0.
μ_0 is the value to test.
sigma is the population (known) standard deviation. If omitted, the sample standard deviation is used.

Formula: ZTEST is calculated as follows when sigma is not omitted:

$$ZTEST(array, \mu_0) = 1 - NORMSDIST((\bar{x} - \mu_0)/(sigma / \sqrt{n}))$$

or when sigma is omitted:

$$ZTEST(array, \mu_0) = 1 - NORMSDIST((\bar{x} - \mu_0)/(s / \sqrt{n}))$$

where *x* is the sample mean AVERAGE(array); *s* is the sample standard deviation STDEV(array); and *n* is the number of observations in the sample COUNT(array).
ZTEST represents the probability that the sample mean would be greater than the observed value AVERAGE(array), when the underlying population mean is μ_0. From the symmetry of the Normal distribution, if AVERAGE(array) < μ_0, ZTEST will return a value greater than 0.5.
The following Excel formula can be used to calculate the two-tailed probability that the sample mean would be further from μ_0 (in either direction) than AVERAGE(array), when the underlying population mean is μ_0:
=2 * MIN(ZTEST(array, μ_0,sigma), 1 - ZTEST(array, μ_0,sigma)).

ASC:

Description: Using the ASC function changes full-width (double-byte) characters to half-width (single-byte) characters.

General Syntax: **=ASC(text)**

Arguments: **text** is the text or a reference to a cell that contains the text you want to change. If text does not contain any full-width letters, text is not changed.

BAHTTEXT:

Description: Using the BAHTTEXT function converts a number to Thai text and adds a suffix of "Baht." In Excel for Windows, you can change the Baht format to a different style by using "Regional Settings" or "Regional Options" in the Control Panel.

General Syntax: **=BAHTTEXT(number)**

Arguments: **number** is a number you want to convert to text, or a reference to a cell containing a number, or a formula that evaluates to a number.

CHAR:

Description: Using the CHAR function returns the character specified by a number. Use it to translate code page numbers you might get from files on other types of computers into characters.

General Syntax: **=CHAR(number)**

Arguments: **number** is a number between 1 and 255 specifying which character you want. The character is from the character set used by your computer.

Glossary of Functions

CLEAN:

Description: Using the CLEAN function removes all nonprintable characters from text. Use CLEAN on text imported from other applications that contains characters that may not print with your operating system.

General Syntax: **=CLEAN(text)**

Arguments: **text** is any worksheet information from which you want to remove nonprintable characters.

CODE:

Description: Using the CODE function returns a numeric code for the first character in a text string. The returned code corresponds to the character set used by your computer.

General Syntax: **=CODE(text)**

Arguments: **text** is the text for which you want the code of the first character.

CONCATENATE:

Description: Using the CONCATENATE function joins several text strings into one text string. The "&" operator can also be used instead of CONCATENATE to join text items.

General Syntax: **=CONCATENATE(text_values)**

Arguments: **text_values** are 1 to 30 text strings, numbers, or single-cell references to be joined into a single text item.

DOLLAR:

Description: Using the DOLLAR function converts a number to text using currency format, with the decimals rounded to the specified place. The format used is $#,##0.00_);($#,##0.00).

General Syntax: **=DOLLAR(number**,*decimals*)

Arguments: **number** is a number, a reference to a cell containing a number, or a formula that evaluates to a number.
decimals is the decimal precision. If it is negative, the number is rounded to the left of the decimal. If omitted, it is 2.

EXACT:

Description: Using the EXACT function compares two text strings and returns TRUE if they are exactly the same, FALSE otherwise. EXACT is case-sensitive but ignores formatting differences.

General Syntax: **=EXACT(text1,text2)**

Arguments: **text1** is the first text string.
text2 is the second text string.

FIND:

Description: Using the FIND function finds one text string within another text string, and returns the number of the starting position of the found text, from the first character of the text it is found within. FIND is case sensitive and doesn't allow wildcard characters.

General Syntax: **=FIND(find_text,within_text**,*start_num*)

Arguments: **find_text** is the text you want to find.
within_text is the text containing the text you want to find.
start_num specifies the character at which to start the search. If omitted, it is 1.

Glossary of Functions

FIXED:

Description: Using the FIXED function rounds a number to the specified number of decimals, formats the number in decimal format using a period and commas, and returns the result as text.

General Syntax: **=FIXED(number**,*decimals*,*no_commas***)**

Arguments: **number** is the number you want to round and convert to text.
decimals is the number of digits to the right of the decimal point.
no_commas is a logical value that, if TRUE, prevents FIXED from including commas in the returned text.

JIS:

Description: Using the JIS function converts half-width (single-byte) letters within a character string to full-width (double-byte) characters. For Japanese, this function changes half-width (single-byte) English letters or katakana within a character string to full-width (double-byte) characters.

General Syntax: **=JIS(text)**

Arguments: **text** is the text or a reference to a cell that contains the text you want to change. If text does not contain any half-width English letters or katakana, text is not changed

LEFT:

Description: Using the LEFT function returns the first character or characters in a text string, based on the number of characters you specify.
General Syntax: **=LEFT(text**,*num_chars***)**
Arguments: **text** is the text string that contains the characters you want to extract.
num_chars specifies the number of characters you want LEFT to extract. It must be greater than or equal to zero. If it is greater than the length of text, LEFT returns all of text. If it is omitted, it is assumed to be 1.

LEN:

Description: Using the LEN function returns the number of characters in a text string.
General Syntax: **=LEN(text)**
Arguments: **text** is the text whose length you want to find. Spaces count as characters.

LOWER:

Description: Using the LOWER function converts all uppercase letters in a text string to lowercase.
General Syntax: **=LOWER(text)**
Arguments: **text** is the text you want to convert to lowercase. LOWER does not change characters in text that are not letters.

MID:

Description: Using the MID function returns a specific number of characters from a text string, starting at the position you specify, based on the number of characters you specify.
General Syntax: **=MID(text,start_num,num_chars)**
Arguments: **text** is the text string containing the characters you want to extract.
start_num is the position of the first character you want to extract in text.
num_chars specifies the number of characters you want MID to return from text.

PHONETIC:

Description: Using the PHONETIC function extracts the phonetic (furigana) characters from a text string.
General Syntax: **=PHONETIC(text)**
Arguments: **text** is a text string or a reference to a single cell or a range of cells that contain a furigana text string.

PROPER:

Description: Using the PROPER function capitalizes the first letter in a text string and any other letters in text that follow any character other than a letter. Converts all other letters to lowercase letters.

General Syntax: **=PROPER(text)**

Arguments: **text** is text enclosed in quotation marks, a formula that returns text, or a reference to a cell containing the text you want to partially capitalize.

REPLACE:

Description: Using the REPLACE function replaces part of a text string, based on the number of characters you specify, with a different text string.

General Syntax: **=REPLACE(old_text,start_num,num_chars,new_text)**

Arguments: **old_text** is text in which you want to replace some characters.
start_num is the position of the character in **old_text** that you want to replace with **new_text**.
num_chars is the number of characters in **old_text** that you want to replace with **new_text**.
new_text is the text that will replace characters in **old_text**.

REPT:

Description: Using the REPT function repeats text a given number of times. Use REPT to fill a cell with a number of instances of a text string.

General Syntax: **=REPT(text,number_times)**

Arguments: **text** is the text you want to repeat.
number_times is a positive number specifying the number of times to repeat text.

RIGHT:

Description: Using the RIGHT function returns the last character or characters in a text string, based on the number of characters you specify.

General Syntax: **=RIGHT(text**,*num_chars*)

Arguments: **text** is the text string containing the characters you want to extract.
num_chars specifies the number of characters you want RIGHT to extract.

SEARCH:

Description: Using the SEARCH function returns the number of the character at which a specific character or text string is first found, beginning with *start_num*. Use SEARCH to determine the location of a character or text string within another text string so that you can use the MID or REPLACE functions to change the text.

General Syntax: **=SEARCH(find_text,within_text**,*start_num*)

Arguments: **find_text** is the text you want to find. You can use the wildcard characters of question mark (?) and asterisk (*), in **find_text**. A question mark matches any single character; an asterisk matches any sequence of characters. If you want to find an actual question mark or asterisk, type a tilde (~) before the character.
within_text is the text in which you want to search for find_text.
start_num is the character number in within_text at which you want to start searching.

SUBSTITUTE:

Description: Using the SUBSTITUTE function substitutes new_text for old_text in a text string. Use SUBSTITUTE when you want to replace specific text in a text string; use REPLACE when you want to replace any text that occurs in a specific location in a text string.

General Syntax: **=SUBSTITUTE(text,old_text,new_text**,*instance_num*)

Arguments: **text** is the text or the reference to a cell containing text for which you want to substitute characters.
old_text is the text you want to replace.
new_text is the text you want to replace **old_text** with.
instance_num specifies which occurrence of **old_text** you want to replace with **new_text**. If you specify *instance_num*, only that instance of old_text is replaced. Otherwise, every occurrence of **old_text** is changed to **new_text**.

T:

Description: Using the T function returns the text referred to by value. You do not generally need to use the T function in a formula because Microsoft Excel automatically converts values as necessary. This function is provided for compatibility with other spreadsheet programs.

General Syntax: **=T(value)**

Arguments: **value** is the value you want to test.

TEXT:

Description: Using the TEXT function converts a value to text in a specific number format.

General Syntax: **=TEXT(value,format_text)**

Arguments: **value** is a numeric value, a formula that evaluates to a numeric value, or a cell reference containing a numeric value.
format_text is a number format from the "Category" box on the "Number" tab in the "Format Cells" dialog box.

TRIM:

Description: Using the TRIM function removes all spaces from text except for single spaces between words. Use TRIM on text that you have received from another application that may have irregular spacing.

General Syntax: **=TRIM(text)**

Arguments: **text** is the text from which you want spaces removed.

UPPER:

Description: Using the UPPER function converts text to uppercase.

General Syntax: **=UPPER(text)**

Arguments: **text** is the text reference or text string you want converted to uppercase.

VALUE:

Description: Using the VALUE function converts a text string that represents a number to a number.

General Syntax: **=VALUE(text)**

Arguments: **text** is the text enclosed in quotation marks or a reference to a cell containing the text you want to convert.

Excel Keyboard Shortcuts

Category: Using Windows	
Command	**Key**
Switch to the next window	Alt + Tab
Switch to previous window	Alt + Shift + Tab
Close window	Ctrl + W or Ctrl + F4
Restore window after maximizing it	Alt + F5
Move clockwise to task pane	F6
Move counterclockwise to next task pane	Shift + F6
Switching windows when multiple windows are open	Ctrl + F6
Switch to previous window	Ctrl + Shift + F6
Maximize/Restore window	Ctrl + F10
Copy screen to clipboard	Print Screen
Copy window to clipboard	Alt + Print Screen

Category: Using Dialog Boxes	
Command	**Key**
Switch between screen and dialog box (if possible)	Alt + F6
Move to next option	Tab
Move to previous option	Alt + Tab
Move to next tab	Ctrl + Tab
Move to previous tab	Ctrl + Shift + Tab
Move between options in a drop-down menu or option group	Arrow keys
Perform button action or select/clear a checkbox	Spacebar
Select an option or select/clear a checkbox	Alt + underlined letter in option
Open a drop-down menu	Alt + Down Arrow
Select drop-down option	First letter of option
Close list/ Cancel	Esc
Run command	Enter

Category: Text	
Command	**Key**
Move to start of text	Home
Move to end of text	End
Move left one character	Left Arrow
Move right one character	Right Arrow
Move one word to left	Ctrl + Left Arrow
Move right one word	Ctrl + Right Arrow
Select/Deselect to left	Shift + Left Arrow
Select/Deselect to right	Shift + Right Arrow
Select/Deselect word left	Ctrl + Shift + Left Arrow
Select/Deselect word right	Ctrl + Shift + Right Arrow
Select to beginning	Shift + Home
Select to end	Shift + End

Category: Using the "Open" and "Save As" Dialog Boxes	
Command	**Key**
Show "Open" dialog box	Ctrl + O or Ctrl + F12
Show "Save As" dialog box	F12
Move to previous folder	Alt + 1
Move up one level	Alt + 2
Delete selected folder/file	Del or Delete
Create new folder	Alt + 4
Switch folder view	Alt + 5
Show shortcut menu	Shift + F10
Move between options	Tab
Open the "Look in" list	F4 or Alt + I

Category: Undoing and Redoing Actions	
Command	**Key**
Cancel action	Esc
Undo Action	Ctrl + Z
Redo/Repeat Action	Ctrl + Y

Excel Keyboard Shortcuts

Category: Moving and Scrolling in Worksheets/Workbooks	
Command	**Key**
Move one cell up, down, left or right	Arrow Keys
Move to the edge of the current data region	CTRL+ arrow key
Move to the beginning of the row	HOME
Move to the beginning of the worksheet	CTRL + HOME
Move to the last cell on the worksheet	CTRL + END
Move down one screen	PAGE DOWN
Move up one screen	PAGE UP
Move one screen to the right	ALT + PAGE DOWN
Move one screen to the left	ALT + PAGE UP
Move to the next sheet in the workbook	CTRL + PAGE DOWN
Move to the previous sheet in the workbook	CTRL + PAGE UP

Category: Entering Data	
Command	**Key**
Complete a cell entry and move down in the selection	ENTER
Start a new line in the same cell	ALT + ENTER
Fill the selected cell range with the current entry	CTRL + ENTER
Complete a cell entry and move up in the selection	SHIFT + ENTER
Complete a cell entry and move to the right in the selection	TAB
Complete a cell entry and move to the left in the selection	SHIFT + TAB
Cancel a cell entry	ESC
Delete the character to the left of the insertion point, or delete the selection	DELETE
Delete text to the end of the line	CTRL + DELETE
Move one character up, down, left or right	Arrow Keys
Move to the beginning of the line	HOME
Repeat the last action	F4 **or** CTRL + Y

Category: Selecting Cells, Columns, or Rows	
Command	**Key**
Extend the selection by one cell	SHIFT + arrow key
Extend the selection to the last nonblank cell in the same column or row as the active cell	CTRL + SHIFT + arrow key
Extend the selection to the beginning of the row	SHIFT + HOME
Extend the selection to the beginning of the worksheet	CTRL+ SHIFT + HOME
Extend the selection to the last used cell on the worksheet (lower-right corner)	CTRL + SHIFT + END
Select the entire column	CTRL + SPACEBAR
Select the entire row	SHIFT + SPACEBAR
Select the entire worksheet	CTRL + A
Select only the active cell when multiple cells are selected	SHIFT + BACKSPACE
Extend the selection down one screen	SHIFT + PAGE DOWN
Extend the selection up one screen	SHIFT + PAGE UP
Select whole data area around active cell	CTRL + SHIFT + *

Category: Other Functions	
Command	**Key**
Display the “Print” dialog box	CTRL + P
Insert a new worksheet	SHIFT + F11

Category: Inserting, deleting and copying a selection	
Command	**Key**
Copy the selection	CTRL + C
Cut the selection	CTRL + X
Paste the selection	CTRL + V
Clear the contents of the selection	DELETE
Delete the selection	CTRL + HYPHEN
Copy the selection	CTRL + C

Excel Keyboard Shortcuts

Category: Outlining Data	
Command	**Key**
Group rows or columns	ALT + SHIFT + RIGHT ARROW
Ungroup rows or columns	ALT + SHIFT + LEFT ARROW
Display or hide outline symbols	CTRL + 8
Hide selected rows	CTRL + 9
Unhide selected rows	CTRL + SHIFT + ((opening parenthesis)
Hide selected columns	CTRL + 0 (zero)
Unhide selected columns	CTRL + SHIFT +) (closing parenthesis)

Category: Working in Cells or the Formula Bar	
Command	**Key**
Edit the active cell and then clear it, or delete the preceding character in the active cell as you edit cell contents	BACKSPACE
Complete a cell entry	ENTER
Enter a formula as an array formula	CTRL + SHIFT + ENTER
Cancel an entry in the cell or formula bar	ESC
Display the Formula Palette after you type a function name in a formula	CTRL + A
Insert the argument names and parentheses for a function after you type a function name in a formula	CTRL + SHIFT + A
Insert a hyperlink	CTRL + K
Activate a hyperlink	ENTER (in a cell with a hyperlink)
Edit the active cell and position the insertion point at the end of the line	F2
Paste a defined name into a formula	F3
Paste a function into a formula	SHIFT + F3
Calculate all sheets in all open workbooks	F9
Calculate all sheets in the active workbook	CTRL + ALT + F9

Category: Working in Cells or the Formula Bar (cont.)	
Command	**Key**
Calculate the active worksheet	SHIFT + F9
Start a formula	= (equal sign)
Insert the AutoSum formula	ALT + = (equal sign)
Enter the date	CTRL + ; (semicolon)
Enter the time	CTRL + SHIFT + : (colon)
Copy the value from the cell above the active cell into the cell or the formula bar	CTRL + SHIFT + " (quotation mark)
Alternate between displaying cell values and displaying cell formulas	CTRL + ‘ (single left quotation mark)
Copy a formula from the cell above the active cell into the cell or the formula bar	CTRL + ‘ (apostrophe)
Display the AutoComplete list	ALT + DOWN ARROW

Category: Formatting Data	
Command	**Key**
Display the “Insert” cells dialog box	CTRL + SHIFT + + (plus sign)
Display the “Delete” cells dialog box	CTRL + SHIFT + - (minus sign)
Display the Style dialog box	Alt + ‘ (apostrophe)
Display the Format Cells dialog box	CTRL + 1
Apply the General number format	CTRL + SHIFT + ~
Apply the Currency format with two decimal places (negative numbers appear in parentheses)	CTRL + SHIFT + $
Apply the Percentage format with no decimal places	CTRL + SHIFT + %
Apply the Exponential number format with two decimal places	CTRL + SHIFT + ^
Apply the Date format with the day, month and year	CTRL + SHIFT + #

Category: Formatting Data (cont.)	
Command	**Key**
Apply the Time format with the hour and minute, and indicate A.M. or P.M.	CTRL + SHIFT + @
Apply the Number format with two decimal places, thousands separator, and minus sign (-) for negative values	CTRL + SHIFT + !
Apply the outline border	CTRL + SHIFT + &
Remove outline border	CTRL + SHIFT + _
Apply or remove bold formatting	CTRL + B or CTRL + 2
Apply or remove italic formatting	CTRL + I or CTRL + 3
Apply or remove an underline	CTRL + U or CTRL + 4
Apply or remove strikethrough formatting	CTRL + 5 or CTRL + 5
Hide rows	CTRL + 9
Unhide rows	CTRL + SHIFT + ((opening parenthesis)
Toggle between hiding, showing, and showing placeholder for objects	CTRL + 6
Hide columns	CTRL + 0 (zero)
Unhide columns	CTRL + SHIFT +) (closing parenthesis)