

Create an expression

Often, you need information that isn't directly available in a field in your database. For example, you might need to calculate **sales tax** on an order, or calculate the **order total** itself. At other times, you'll need to supply a query or filter with criteria — information that determines which records you work with. Or you might want to set a default value for a field or control, or a validation rule for a field or table. In all of these cases, you use an expression.

This article introduces expressions. You will learn what an expression is and why and where you use one. You will also learn about the Expression Builder, a tool in Access that you can use to help you build expressions. Finally, you will learn how to create an expression, either by using the Expression Builder or by building one from scratch.

What is an expression?

In Access, an expression is equivalent to a formula in Excel.

An expression consists of a number of possible elements that are used, alone or in combination, to produce a result.

Elements can include identifiers (the names of fields, controls, or properties), operators such as + (plus) or - (minus), functions, constants, and values.

You use an expression to perform a calculation, retrieve the value of a control, supply criteria to a query, define rules, create calculated controls and calculated fields, and define a group level for a report.

Here are some sample expressions.

Expression	Purpose
=[RequiredDate]-[ShippedDate]	Calculates the difference between the values in two text box controls on a report
Date()	Sets the default value for a field in a table to the current date
ExtendedPrice: CCur([Order Details].UnitPrice*[Quantity]*(1-[Discount])/100)*100	Creates a calculated field in a query
Between #1/1/2005# And #12/31/2005#	Used to enter criteria for a Date/Time field in a query
=[Orders Subform].Form!OrderSubtotal	Returns the value of the OrderSubtotal control on the Orders subform that is on the Orders form
> 0	Sets a validation rule for a numeric field in a table

As you can see by reviewing the preceding sample expressions, an expression in Access is not just a calculation. Expressions are used for a variety of different purposes.

You'll also notice that the sample expressions differ from each other in some ways. For example, some expressions start with the = operator.

When calculating a value for a control on a form or report, you use the = operator to start the expression. In other instances, you don't use the = operator. For example, when you enter an expression in a query, or in the DefaultValue or ValidationRule property of a field or control, you don't use the = operator.

Identifiers

An identifier is the name of a field, property, or control. You use an identifier in an expression to refer to the value that is associated with a field, property, or control. For example, consider the expression =[RequiredDate]-[ShippedDate]. This expression subtracts the value of the ShippedDate field or control from the value of the RequiredDate field or control. In this expression, both RequiredDate and ShippedDate are identifiers.

Operators

Access supports a variety of operators, including the expected arithmetic operators such as +, -, * (multiply), / (divide), as well as comparison operators for comparing values, text operators for concatenating text, logical operators for determining true or false values, and other operators specific to Access. For details about operators, see the Table of operators section in this article.

Functions

Functions are built-in procedures that you can use in your expressions. You use them to perform calculations, manipulate text and dates, summarize data, and perform a wide variety of operations. For example, one of the more commonly used functions is Date. The Date function returns the current date. You might use it in an expression that sets the default value for a field in a table. That way, whenever a new record is added, the field defaults to the current date.

Some functions require arguments. An argument is a value that serves as input to the function. If a function requires more than one argument, you separate the arguments with a comma. For example, consider the Format function in the following example expression:

=Format(Date(),"mmm d, yyyy")

In this example, we supply two arguments. The first is the Date function. You can often supply the value that a function returns as an argument to yet another function. In this case, we supply the current date, as returned by the Date function. The second argument, separated from the first by a comma, is a text string that tells the Format function how to format the date. Note that the text string is enclosed in quotation marks. As a general rule, when you need to supply text, place it inside quotation marks.

Access supports a long list of built-in functions. For more information about the functions that are available, see the Help topics Functions (by category) and Functions (alphabetical).

Constants

A **constant is a named item whose value remains constant while Access is running. The constants you'll use most often in your expressions are True, False, and Null.**

You can also define your own constants in Visual Basic for Applications (VBA) that you can use in VBA procedures. VBA is the programming language that Access uses.

Note You cannot use Microsoft Visual Basic constants in custom functions that you use in your expressions. For example, Visual Basic has constants for the days of the week: vbSunday represents Sunday, vbMonday represents Monday, and so on. Each constant has a corresponding numeric value: The numeric value for vbSunday is 1, vbMonday is 2, and so on. You cannot use these constants in a custom function that is called from within an expression. You must instead use the numeric values.

Values

You can employ literal values in your expressions. **Numeric values can be a series of digits, including a sign and a decimal point, if needed.** In the absence of a sign, Access assumes a positive value. To make a value negative, include the minus sign (-). You can also use scientific notation. To do so, include E or e and the sign of the exponent (for example, 1.0E-6).

Text string values should be placed within quotation marks.

In some circumstances, Access will supply them for you. For example, when you type text in an expression for a validation rule or for query criteria, Access will supply the quotation marks automatically.

For example, if you type the text Paris, Access displays "Paris" in the expression. If you want an expression to produce a string that is actually enclosed in quotation marks, you can enclose the nested string either in single (') quotation marks or within three sets of double (") quotation marks. For example, the following expressions are equivalent:

```
Forms![Contacts]![City].DefaultValue = ' "Paris" '  
Forms![Contacts]![City].DefaultValue = " " "Paris" " "
```

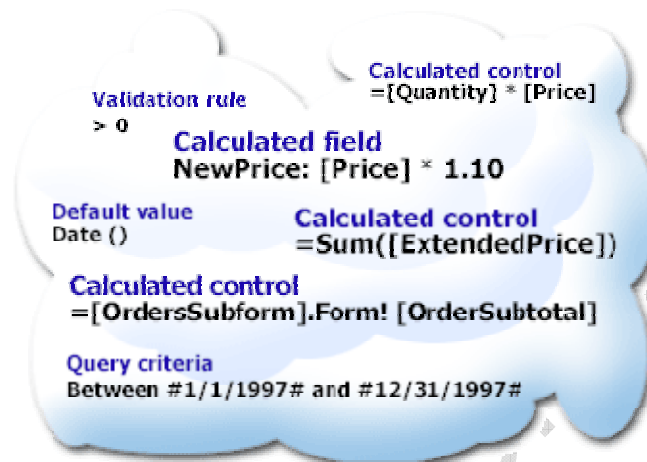
Date/Time values should be enclosed in number signs (#). For example, #3-7-05#, #7-Mar-05#, and #Mar-7-2005# are all valid Date/Time values. When Access sees a valid Date/Time value enclosed by # characters, it automatically treats the value as a Date/Time data type.

Why use expressions?

In Access, you use expressions often and for a variety of reasons. For example, when you want to calculate a value for a field on a form, you use an expression. Let's say you want to calculate the total dollar amount for a line item in an invoice. Typically, you do not store the line item total in the database. Instead, you calculate it as needed from two items that you should store in the database — quantity and price.

```
=CCur([Quantity]*[Price])
```

This expression multiplies quantity times price and then uses the CCur function (convert to currency) to convert the result to the Currency data type. You usually use a text box control to hold a calculated value, but you can use any control that has a ControlSource property. A control that has an expression as its control source is called a calculated control. If the control is a text box, you can enter the expression directly in the text box. You can also enter it in the ControlSource property in the property sheet.



Besides using an expression to calculate a needed value that isn't stored in the database, you often use an expression to supply information to a query. For example, suppose you want to see product sales that were shipped within a certain time frame. You can enter a criteria expression that uses the Between operator to define the date range. Access returns only the rows that match the criteria and have a shipped date within the specified date range:

Between #1/1/2004# And #12/31/2004#

You might also decide that a good place to calculate a line item total is in the query that provides the form or report with its data. A column in a query that results from such a calculation is called a calculated field. For example, the following expression in a query calculates the line item totals with an applied discount:

ExtendedPrice: CCur([Order Details].[UnitPrice]*[Quantity]*(1-[Discount])/100)*100

The expression gives the resulting column the name ExtendedPrice. Another good use of an expression is to provide a default value for a field in a table or for a control. For example, if you have a date field that you want to default to the current date, you can type Date() in the Default Value property box for that field. You can also use an expression to set a validation rule. For example, you might use a validation rule that requires the date that is entered to be greater than or equal to the current date. In that case, you set the value in the Validation Rule property box to >= Date().

Where and how to use expressions

You can use an expression in many places. For example, tables, queries, forms, reports, and macros all have places where you can use an expression. In addition, when you write VBA code for an event procedure or for a module, you often use expressions that are similar to those that you use in an Access object, such as a table or query.

In expressions, field names and control names must be enclosed in brackets; for example, [Unit Price]. If you enter a name that doesn't include spaces or special characters, Access automatically encloses it in brackets. If a name includes spaces or special characters, you must type the brackets yourself. You can give a name to a calculated control by setting the Name property. The name must be unique among all the control names on the form or report. It also must be different from any field or control name that is used in the expression for that control, and it should be different from any field name in the underlying table or query. You can use this name when you want to refer to the value in the control in other expressions on the form or report.

The most common places where you use expressions include the following:

In a text box control on a form or report

You use an expression in a text box control to create a calculated control. For example, suppose you want to create a subtotal that sums all of the line items on an order form. The subtotal might look something like this.




Subtotal:	\$814.50
Freight	\$29.46
Total:	\$843.96

To calculate the subtotal, you place a text box control on the form and set the ControlSource property of the text box to the following expression:

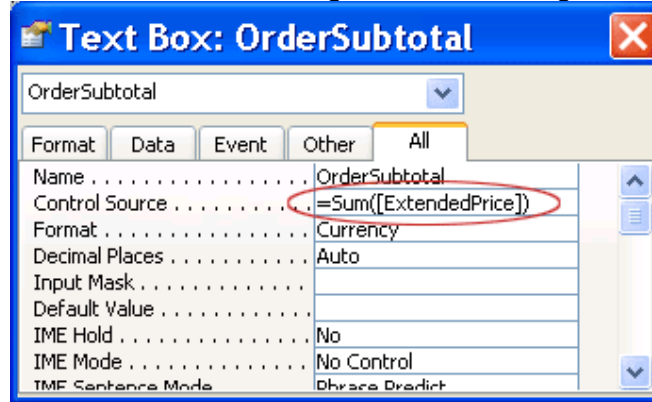
=Sum([ExtendedPrice])

The Sum function calculates the total for a set of values from your record source — in this case, the column named ExtendedPrice.

To enter an expression in a text box control :

1. In the Database window, under Objects, click Forms.
2. Click the form, and then click Design in the Database window.
3. Click the text box to select it.
4. On the View menu, click Properties to display the property sheet for the text box.
Access displays the property sheet for the text box.
5. Change the ControlSource property of the text box to = followed by the expression, or click the Build button  to the right of the property box to create an expression by using the Expression Builder. For example, to calculate the subtotal shown above, you type =Sum([ExtendedPrice]).

Your property sheet will look something like the following illustration.



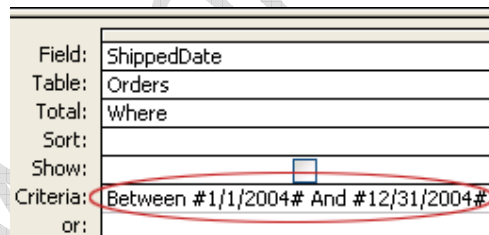
6. Close the property sheet.

In the criteria cell in the design grid of a query

You use an expression to define criteria in a query. Access then returns only those rows that match the criteria. For example, suppose you want to see all orders whose shipped date occurs in the year 2004. To enter the criteria, you enter the following expression in the Criteria cell for the ShippedDate column in your query:


Between #1/1/2004# And #12/31/2004#

The ShippedDate column will look something like the following illustration.



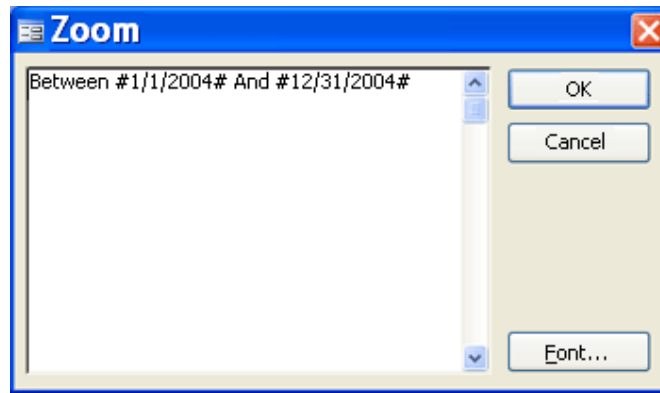
The expression is used to determine whether the shipped date falls in the date range that you specified. Note that the date values are surrounded by number signs (#). Access treats a value within number signs as a Date/Time data type.

To enter criteria in the query design grid

1. In the Database window, under Objects, click Queries.
2. Click the query, and then click Design in the Database window.
3. Click in the criteria cell in the column for which you want to enter matching criteria.
4. Type the criteria expression, or click the Build button  on the toolbar to create an expression by using the Expression Builder.

Note Do not precede the criteria expression with the = operator.

If you want a larger area in which to enter an expression, press SHIFT+F2 to display the Zoom box.

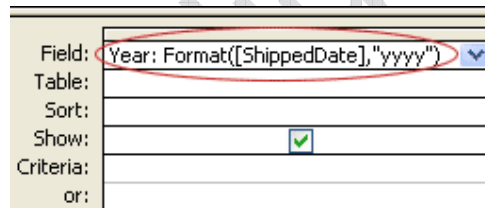


In the field cell in the design grid of a query


You use an expression to create a calculated field in a query. For example, suppose you want to display the year that an order was shipped as part of a query. To create the calculated field, you place the following expression in the field cell of an empty column in your query:

Year: Format([ShippedDate], "yyyy")

The expression uses the Format function to extract the year from the ShippedDate field and format it as four digits. Note that the resulting column is given the name Year by preceding the expression with Year:

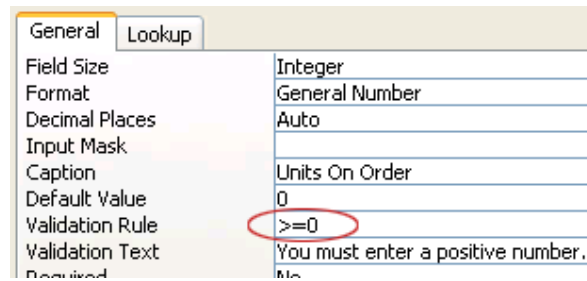


To enter a calculated field in query design view

1. In the Database window, under Objects, click Queries.
2. Click the query, and then click Design in the Database window.
3. Click the field cell in the column where you want to create the calculated field.
4. Type the expression, or click the Build button  on the toolbar to create an expression by using the Expression Builder. You should start the expression with a name followed by a colon. For example, you type ExtendedPrice: to start an expression that creates a calculated field called ExtendedPrice.

In the ValidationRule property of a field in a table

Another place where an expression comes in handy is the ValidationRule property of a field in a table. Suppose you want to enforce a rule that forces the [Units On Hand] field in the Inventory table to have a value greater than or equal to zero. In other words, inventory can never be a negative number. You can do this by using the expression shown in the following illustration.



To enter a validation rule for a field in a table

1. In the Database window, under Objects, click Tables.
2. Click the table, and then click Design in the Database window.
3. Click the field name for the field that you want.
4. Click the Validation Rule property box.
5. Type the expression, or click the Build button to the right of the property box to create an expression by using the Expression Builder.

Note Do not precede the expression with the = operator when you create a validation rule.

The key point to remember when you work with validation rule expressions is that they must resolve to True for the value to be accepted. So, in this example, the value for [Units On Hand] must be ≥ 0 . If it isn't, Access displays the text shown in the Validation Text property box. If you haven't entered any text in the Validation Text property box, Access displays its own message to indicate that the value you entered is prohibited by the validation rule for the field.

In the ValidationRule property of a control


You can also set the ValidationRule property for a control. For example, suppose you use a form to enter the date range for a report, and you want to ensure that the beginning date isn't earlier than #1/1/2004#. You can set the ValidationRule and ValidationText properties for the text box where you enter the beginning date to something like the following.

Property	Setting
ValidationRule	\geq #1/1/2004#
ValidationText	You cannot enter a date earlier than 1/1/2004.

If you try to enter a date earlier than #1/1/2004#, a message appears. After you click OK, you are returned to the text box.



To enter a validation rule for a control

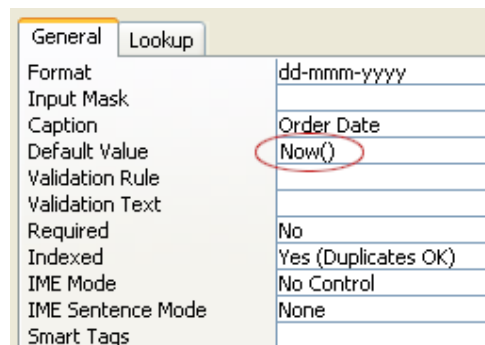
1. In the Database window, under Objects, click Forms or Reports.
2. Click the form or report, and then click Design in the Database window.
3. Click the control to select it.
4. Click the Properties button on the toolbar.
Access displays the property sheet for the control.
5. Click the All tab, and then click the Validation Rule property box.
6. Type the expression, or click the Build button  to the right of the property box to create an expression by using the Expression Builder.
Note Do not precede the expression with the = operator when you create a validation rule.
7. Optionally, change the ValidationText property.
8. Close the property sheet.

When you enter a value that is prohibited by the validation rule, you can press ESC while the insertion point is in the control to restore the original or default value. Then you can enter a value that satisfies the validation rule.

If the ControlSource property for your control is a field in a table, it's usually best to set the field's ValidationRule property in addition to that of the control. That way, the rule is enforced all the time, no matter which form or query is used to update the field.


In the DefaultValue property for a field in a table

You can use an expression to store a default value for a field in a table. For example, suppose you want to automatically insert the date and time into the OrderDate field when you add a new record. You can use an expression like the following.



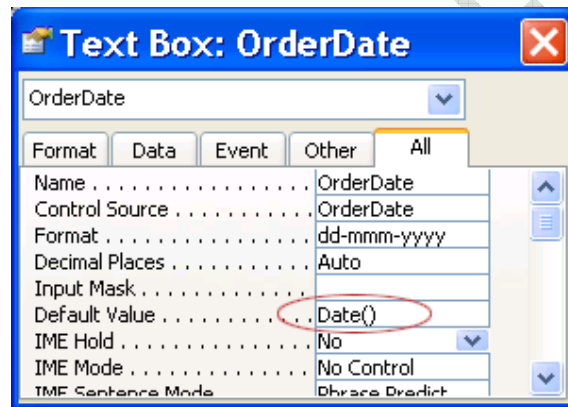
The expression uses the Now function to insert the date and time into the OrderDate field.

To enter a default value for a field in a table

1. In the Database window, under Objects, click Tables.
 2. Click the table, and then click Design in the Database window.
 3. Click the field name for the field that you want.
 4. Click the Default Value property box.
 5. Type the expression, or click the Build button  to the right of the property box to create an expression by using the Expression Builder.
- If a control is bound to a field in a table, and the field has a default value, the control's default value takes precedence.


In the DefaultValue property in a control

Another common place where you can use an expression is in the DefaultValue property of a control. The DefaultValue property of a control behaves similarly to the DefaultValue property of a field in a table. For example, if you want to enter the current date as the default value for an OrderDate text box, you can use an expression like the following.



This expression uses the Date function to return the current date, but not the time. If the text box control is bound to a field in a table, and the field has a default value, the control's default value takes precedence. It often makes better sense to set the DefaultValue property for the field in the table, because the default value is always applied unless a control overrides it.

To enter a default value for a control

1. In the Database window, under Objects, click Forms or Reports.
2. Click the form or report, and then click Design in the Database window.
3. Click the control to select it.
4. Click the Properties button on the toolbar.
Access displays the property sheet for the control.
5. Click the All tab, and then click the DefaultValue property box.
6. Type the expression, or click the Build button  to the right of the property box to create an expression by using the Expression Builder.
7. Close the property sheet.

In the Condition column in a macro

In some cases, you might want to carry out an action or series of actions in a macro only if a particular condition is true. For example, suppose you want an action to run only when the value of the Counter text box is 10. You use an expression to define the condition in the Condition column of the macro.

Macro Name	Condition	Action
Show Break	[Counter]=10	SetValue

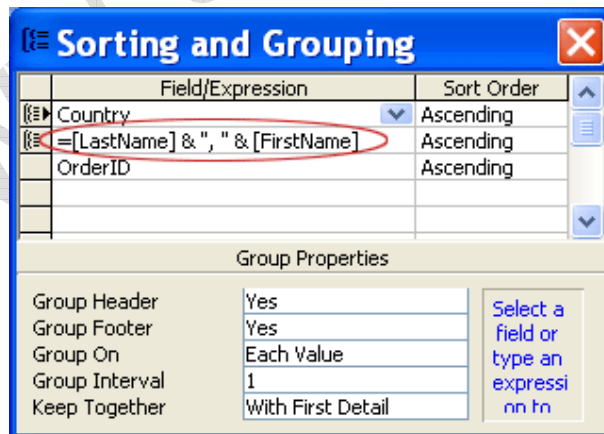
To enter a condition for a macro action

1. In the Database window, under Objects, click Macros.
2. Click the macro, and then click Design in the Database window.
3. Click the Condition cell for the macro action that you want to select.
If the Condition column is not visible, on the View menu, click Conditions.
4. Type a conditional expression.
5. Close the macro.

As with the ValidationRule property, the Condition column expression is a conditional expression. It must resolve to either True or False. The action takes place only when the condition is True.

In the Sorting and Grouping box

You use the Sorting and Grouping box to define group levels and sorting options for a report. You can group on a field or on an expression. For example, suppose you want to group your report by country, then by name, and then by order ID. If your name information is stored in separate fields — LastName and FirstName — you need to group on an expression similar to the one shown in the following illustration.



This expression uses the ampersand (&) operator to combine text values (this is often called string concatenation). For more information about combining text values, see the Combining text values section in this article.

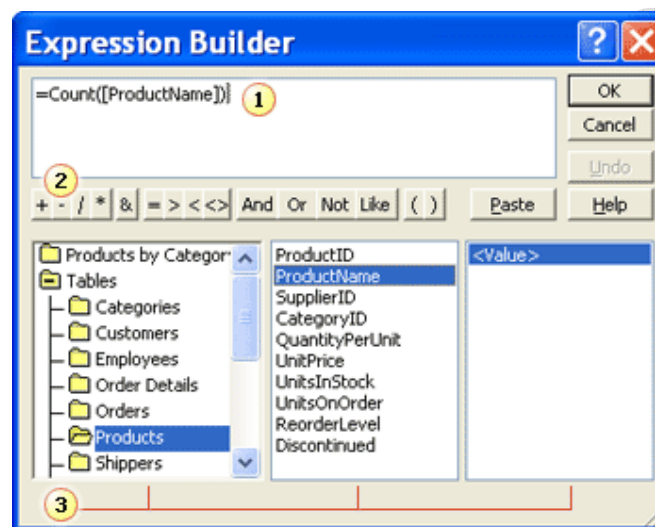
To enter an expression in the Sorting and Grouping box

1. In the Database window, under Objects, click Reports.
2. Click the report, and then click Design in the Database window.

3. Open the report in Design view.
4. On the View menu, click Sorting and Grouping.
5. Click a cell in the Field/Expression column.
6. Type an expression preceded by the = operator.

Using the Expression Builder

You can use the Expression Builder to help you build expressions. The Expression Builder is a tool that you can start from most places where you write expressions. It offers easy access to the names of the fields and controls that are used in your database, as well as to many of the built-in functions that are available to you when you write expressions. You can use the Expression Builder to create an expression from scratch, or you can select from some prebuilt expressions for displaying page numbers, the current date, and the current date and time.



1 Expression box

The upper section of the builder contains an expression box where you construct your expression. You use the three columns in the lower section of the builder to locate elements that you can paste into the expression box. You can also type parts of the expression directly into the expression box. Thus, you construct an expression by combining some amount of typing and pasting.

2 Operator buttons

The middle section of the Expression Builder displays buttons for commonly used operators. To insert an operator in the expression box, click the appropriate button. To display a list of operators that you can use in expressions, click the Operators folder in the lower-left column, and then click the category that you want in the middle column. The right column then lists all of the operators in the selected category. To insert an operator, double-click the operator, or select it and then click Paste.

3 Expression elements

The lower section contains three columns:

- The left column displays folders that list the tables, queries, forms, and reports in your database, as well as the available built-in functions and user-defined functions, constants, operators, and common expressions.
- The middle column lists specific elements or categories of elements for the folder that is selected in the left column. For example, if you click Built-In Functions in the left column, the middle column lists function categories.
- The right column lists the values, if any, for the elements that you selected in the left and middle columns. For example, if you click Built-In Functions in the left column and then click a function category in the middle column, the right column lists all of the built-in functions in the selected category.

You construct your expression by typing text in the expression box and pasting elements from the other areas within the builder. For example, you can click in the lower-left column to see any of the objects in your database, as well as the functions, constants, operators, and common expressions. When you click an item in the left column, the other columns change accordingly. For example, when you click the name of a table in the left column, the middle column lists the fields in that table. When you double-click Functions and then click Built-In Functions, the middle column lists all of the function categories, and the right column lists the functions in those categories. When you double-click to insert a function into your expression, the function and the text that indicates the needed arguments for that function appear as placeholder text in the expression box. You can then replace that text with the correct argument values.


When you paste an identifier into your expression, the Expression Builder inserts only the parts of the identifier that are required in the current context. For example, if you start the Expression Builder from the property sheet of the Customers form and then paste an identifier for the Visible property of the form in your expression, the Expression Builder pastes only the property name Visible. If you use this expression outside of the context of the form, you must include the full identifier: Forms![Customers].Visible.

To start the Expression Builder in a table, form, or report

Click the property or action argument box that will contain the expression.

Click the Build button  next to the property.

To start the Expression Builder in a query

1. Click the cell in the design grid that will contain the expression. For example, click the Criteria cell for the column where you want to supply criteria, or click the Field cell for the column where you want to create a calculated field.
2. Click the Build button on the toolbar .

You can think of the Expression Builder as a way to look up and insert things that you might have trouble remembering, such as identifier names (for example, fields, tables, forms, and queries) and function names and arguments.

Using expressions for practical purposes

This section introduces some of the ways that you can use expressions to solve problems and calculate needed information for your forms, reports, and tables.

"Stamping" a new record with the current date and time

In some tables, it's important to keep track of ("stamp") the date or the date and time when you added a record. To have Access automatically fill in that value for you, you can create a field with the Date/Time data type and set the DefaultValue property for the field to Date() or Now(). The Date function returns the current date, as stored in your computer's system clock. The Now function returns the current date and time.

To add a date and time stamp field

1. In the Database window, under Objects, click Tables.
2. Click the table, and then click Design in the Database window.
3. Click in the Field Name column, in the first available empty row.
4. Type a name for the field, such as DateAdded.
5. Click in the Data Type column, and select Date/Time.
6. Click the General tab, and then click in the Default Value property box.
7. Type Now() or Date(), and then press the TAB key.
8. Close the table. If Access asks whether you want to save the changes, click Yes.

Now whenever you add a new record to the table, Access automatically inserts the date or the date and time into the DateAdded field.

Combining text values

When you want to combine the values in two or more text fields, you can use the ampersand (&) operator. For example, you might have an Employees form where you want to display the employee's full name in the form header. You can enter the employee's first and last names in text boxes in the detail section.

You can use this expression to display an employee's full name:

=[FirstName] & " " & [LastName]

To combine the text values in two or more controls, you use the & operator. Anything that you want to insert between the values — blank space, punctuation, or unchanging text — should be enclosed in quotation marks. In this example, the string " " inserts one blank space between the first name and the last name.

To add a text box with an expression for full name

1. In the Database window, under Objects, click Forms or Reports.
2. Click the form or report, and then click Design in the Database window.
3. In the toolbox, click the Text Box tool.
Note If the toolbox is not visible, on the View menu, click Toolbox.
4. Drag the pointer on the form or report to create the text box.
5. Click the text box to select it.
6. On the View menu, click Properties, and then click the Data tab.
7. Change the ControlSource property to =[FirstName] & " " & [LastName], and then press the TAB key.
8. Close the property sheet.

Some records might not have a value entered in a field that you are combining. The absent value is called a null value. When you use the &

operator and a field has no value, Access returns a zero-length string for that field. For example, if an employee's record has only a last name, the expression in the previous example returns a zero-length string for the First Name field, a space character, and the value in the Last Name field.

When you combine values, you might want to include a value, such as a comma, only when a value exists in a particular field. For example, suppose you have a Customer table and you want to combine the values in the City, Region, and Postal Code fields for a report. Some records might not have a value in the Region field. In that case, you end up with an unwanted comma before the PostalCode if you use the & operator.

To eliminate the unwanted comma, you can use the plus (+) operator, as shown in the following sample expression:

=[City] & (" " + [Region]) & " " & [PostalCode]

The + operator combines text in the same way as the & operator. However, the + operator also supports what is called Null propagation. This means that if any component of an expression is null, the entire expression is also null. In the previous example, consider the section that reads (" " + [Region]).

Because the + operator is used, the expression within the parentheses includes a comma only if a value exists in the Region field. If a value does not exist, Null propagation goes into effect, and the entire expression within the parentheses evaluates to a null value.

Creating calculated controls to perform arithmetic calculations

You can use expressions to add, subtract, multiply, and divide the values in two or more fields or controls. For example, suppose you record the date that a customer needs to receive an order and the date that the order is shipped. You can find out how many days early (or late) an order was shipped by subtracting the value in the ShippedDate field from the value in the RequiredDate field. You can do this because Access can perform arithmetic calculations on dates.

Control Source = [RequiredDate] - [ShippedDate]

The result of a date calculation is called an interval. This value contains a days component on the left side of the decimal point, and a time component on the right side. If the value returned is a positive number, you know how many days early the order was shipped. If it's negative, you know how many days late it was shipped. If the value is 0, you know the order was shipped on time.

To add a text box with an expression to calculate the number of days early or late

1. In the Database window, under Objects, click Forms or Reports.
2. Click the form or report, and then click Design in the Database window.
3. In the toolbox, click the Text Box tool.

Note If the toolbox is not visible, on the View menu, click Toolbox.

4. Drag the pointer on the form or report to create the text box.
5. Click the text box to select it.
6. On the View menu, click Properties, and then click the Data tab.

7. Change the ControlSource property to =[RequiredDate]-[ShippedDate], and then press the TAB key.
8. Close the property sheet.

When you use the +, -, *, /, or \ operator to perform a calculation on two values, and one of the values is null (that is, no value has been entered), the value of the expression is also null. For example, if one of the dates in the previous expression is null, the value of the entire expression is also null. On a report, that results in blank space. If you want to replace the null value with 0, you can use the Nz function to convert the null value to zero. For example:

=Nz([RequiredDate]-[ShippedDate],0)

Note You can design the fields in your table so that users can't enter null values. When you design the table, set the Required property for that field to Yes. You should also set the DefaultValue property for that field to a non-null value.

Adding the values in two controls

Often, you may want to add the values in two controls. For example, to calculate the total cost of an order, you add the values in the Subtotal and Freight controls, as shown in the following illustration.

Subtotal:	\$814.50
Freight	\$29.46
Total:	\$843.96

To calculate the order total, you create a text box in the detail section of the Orders form.

To add a text box with an expression that calculates a total

1. In the Database window, under Objects, click Forms or Reports.
2. Click the form or report, and then click Design in the Database window.
3. In the toolbox, click the Text Box tool.

Note If the toolbox is not visible, on the View menu, click Toolbox.
4. Drag the pointer on the form or report to create the text box.
5. Click the text box to select it.
6. On the View menu, click Properties, and then click the All tab.
7. Set the property values as shown in the following table.

Property	Setting
Name	Total
ControlSource	=[Subtotal]+[Freight]
Format	Currency

8. Close the property sheet.

Multiplying two values to calculate the sales tax

Suppose you need to calculate the sales tax for an order, and the sales tax rate is stored in the SalesTax control. You take the value in the Subtotal control and multiply it by the value in the SalesTax control to compute the tax. To calculate the sales tax, you create a text box in the detail section of the Orders form.

To add a text box with an expression to calculate the sales tax

1. In the Database window, under Objects, click Forms or Reports.
2. Click the form or report, and then click Design in the Database window.
3. In the toolbox, click the Text Box tool.
Note If the toolbox is not visible, on the View menu, click Toolbox.
4. Drag the pointer on the form or report to create the text box.
5. Click the text box to select it.
6. On the View menu, click Properties, and then click the All tab.
7. Set the property values as shown in the following table.
- 8.

Property	Setting
Name	SalesTax
ControlSource	=[Subtotal]*[SalesTaxRate]
Format	Currency

9. Close the property sheet.

Summing and counting

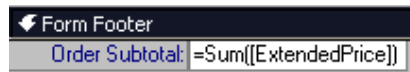
Often, you need to calculate a sum of the values stored in a group of records. For example, you might need to calculate a group total for the group footer in a report, or an order subtotal for the line items on a form. At other times, you might need to count the number of items, rather than to sum them. To calculate a sum for a group of records, you use the Sum function, and to count a group of records, you use the Count function. For example, to count the number of orders in a report that shows orders grouped by customer, you can use the following expression:

=Count([OrderID])

You can use field names in the argument expression for the Sum and Count functions, but not control names. The field names can come from a table or a query. You can even use the name of a calculated field from a query. However, when you want to total the values in a calculated control, you must repeat the expression that is used in the calculated control in the function. When you want to refer to the same expression more than once on a form, or if you are going to use a function, such as Sum, you should consider whether you can include the expression in the form's underlying query. That way, the calculation can be performed in the query instead of the form. It's often faster to perform a calculation in a query.

Calculating an order subtotal on a subform

When you create an order form, you often use a main form and a subform. The main form and subform are linked by a common field, such as OrderID. The main form contains order details, such as "bill to" and "ship to" information, and the subform contains details about the line items that were ordered, such as product, quantity, and unit price. The information in the main form comes from a query that includes the Orders table. The information in the subform comes from a query that includes the Order Details table. If you base the subform on a query that includes a calculated field to compute the extended price, you can sum the values in the ExtendedPrice field to calculate the subtotal. To calculate the order subtotal, you create a text box in the form footer of the Orders subform.



To add a text box with an expression that calculates the order subtotal on a subform

1. In the Database window, under Objects, click Forms.
2. Click the subform, and then click Design in the Database window.
3. In the toolbox, click the Text Box tool.
Note If the toolbox is not visible, on the View menu, click Toolbox.
4. Drag the pointer on the form to create the text box.
5. Click the text box to select it.
6. On the View menu, click Properties, and then click the All tab.
7. Set the property values as shown in the following table.

Property	Setting
Name	OrderSubtotal
ControlSource	=Sum([Extended Price])
Format	Currency

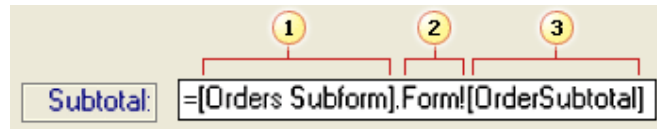
8. Close the property sheet.

Although the text box on the subform calculates the order subtotal, you display the value only on the main form. To hide the text box on the subform, you place it in the form footer of the subform and set the DefaultValue property for the subform to Datasheet View. Access doesn't display a form header or footer when you use a form in Datasheet view. Finally, you insert the subform into the main form.

A subform doesn't usually work correctly as an independent form. Subforms are usually designed to be dependent on a value in the main form. Access limits the subform to the appropriate records only after you insert the subform into the main form and establish the field that links the two forms (such as OrderID). For example, when you use an Orders form with a subform, the OrderID on the main form limits the records in the subform to those that have the same OrderID number.

Referring to the order subtotal on a main form

The controls on a main form and subform can refer to each other. To refer to a value on a subform, you use an expression. This is shown in the following illustration.



- 1 The name of the subform control on the main form
- 2 The Form property, which provides access to the subform's controls and properties
- 3 The name of the text box control on the subform

To display the order subtotal, create a text box in the detail section of the Orders form.

To display the subtotal from a subform in a text box in the detail section of the Orders form

1. In the Database window, under Objects, click Forms.
2. Click the Orders form, and then click Design in the Database window.
3. In the toolbox, click the Text Box tool.
 - Note** If the toolbox is not visible, on the View menu, click Toolbox.
4. Drag the pointer in the detail section of the form to create the text box.
5. Click the text box to select it.
6. On the View menu, click Properties, and then click the All tab.
7. Set the property values as shown in the following table.

Property	Setting
Name	Subtotal
ControlSource	= [Orders Subform].Form![OrderSubtotal]
Format	Currency

8. Close the property sheet.

Looking up a value in a table

When you design a form, you might want to display a value from a table or query other than the one that your form is bound to. For example, you might have a Products form that is bound to the Products table. After you design the form, however, you decide that you want it to display the name of the supplier contact. The name of the supplier contact comes from the Suppliers table. You can look up and display a value from another table or query by using the DLookup function. You supply three arguments to the DLookup function:

- The name of the field whose value you want to look up
- The table or query where the field is located
- The criteria to use to locate the record

To add the supplier contact, open the Products form in Design view and then add a text box with the label Contact Name. The expression to use for this text box is:

=DLookup("[ContactName]","[Suppliers]","[SupplierID]=" & Forms!Products!SupplierID)

To add the supplier contact text box

1. In the Database window, under Objects, click Forms.
2. Click the Orders form, and then click Design in the Database window.
3. In the toolbox, click the Text Box tool.
Note If the toolbox is not visible, on the View menu, click Toolbox.
4. Drag the pointer on the form to create the text box.
5. Click the text box to select it.
6. On the View menu, click Properties, and then click the All tab.
7. Set the property values, as shown in the following table.

Property	Setting
Label	Contact Name
ControlSource	=DLookup("[ContactName]","[Suppliers]","[SupplierID]=" & Forms!Products!SupplierID)

8. Close the property sheet.

This expression searches in the Suppliers table and returns the contact name of the supplier whose supplier ID matches the value in the SupplierID control on the Products form. Note how the & operator is used to construct the third argument. A common error that you should avoid is placing quotation marks around the entire argument instead of around only the text before the & operator.

Note As an alternative to using the DLookup function, you can alter the underlying query to include the information that you need. Using a query is often more efficient.

Printing the date that is printed on a report

On many reports, you may want to print the date that the report was generated. To have Access fill in a date for you, use either the Now function or the Date function. The Now function returns the current date and time, as stored in your computer's clock. The Date function returns only the current date. You can format the result of either of these functions in any one of the available date and time formats.

Suppose you want to print an invoice report, and you want the print date of the invoice to appear in Medium Date format (for example, 31-Dec-04). Type the expression =Date() in the ControlSource property of the text box, or in the text box itself.

To add the print date to a report

1. In the Database window, under Objects, click Reports.
2. Click the report, and then click Design in the Database window.
3. In the toolbox, click the Text Box tool.

- Note** If the toolbox is not visible, on the View menu, click Toolbox.
4. Drag the pointer in the report to create the text box.
- Note** If the page footer is not visible, on the View menu, click Page Header/Footer.
5. Click the text box to select it.
 6. On the View menu, click Properties, and then click the All tab.
 7. Set the property values as shown in the following table.

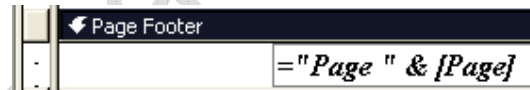
Property	Setting
Name	DatePrinted
ControlSource	=Date()
Format	Medium Date

8. Close the property sheet.

Printing the page number on a report

When you design a report that is longer than one page, you will probably want to add page numbers. You can add page numbers by using the Page property, which automatically numbers the pages when you preview or print the report. The Page property is available only when you preview or print a report, so it doesn't appear in the property sheet.

You use the Page property in the ControlSource property of a text box just as you use a function, such as Now or Date. Place the text box in the page header or page footer of the report. (Note that you don't include parentheses following the Page property.)



You can construct an expression that uses both the Page property and the Pages property. The Pages property returns the total number of pages in the report. For example, the following expression produces page numbering in the format Page 1 of 10.

=\"Page \" & [Page] & \" of \" & [Pages]

To add the preceding style for page numbering to a report

1. In the Database window, under Objects, click Reports.
 2. Click the report, and then click Design in the Database window.
 3. In the toolbox, click the Text Box tool.
- Note** If the toolbox is not visible, on the View menu, click Toolbox.
4. Drag the pointer in the page footer section of the report to create the text box.
- Note** If the page footer is not visible, on the View menu, click Page Header/Footer.
5. Click the text box to select it.
 6. On the View menu, click Properties, and then click the Data tab.

7. Change the value in the Control Source property box to ="Page " & [Page] &" of " & [Pages].
8. Close the property sheet.

Note In the Expression Builder, Access includes some common expressions that you can use for page numbering.

Printing part of a value on a report

If the first or last few characters in a field have a special meaning, you can organize a report around them. For example, if the first two characters in a product identification code indicate the type of product, you can group the products by the first two characters in the code and then identify each group by printing the characters in a group header.

You use the Left function to extract the first n characters of a value in a Text field and the Right function to extract the last n characters. In both cases, the first argument is the field name or text expression, and the second argument is the number of characters that you want to extract.

The following table shows expressions that employ these functions.

If the value in Part ID is	This expression	Returns
AA105	=Left([PartID],2)	AA
AA105	=Right([PartID],3)	105

For more information about the Left and Right functions, see the Left Function and Right Function articles.

Using letter separators in an alphabetical list

For fast identification in a list of products, you can group the products by the first letter of their names and print the letter in a group header, as shown in the following illustration.

Alphabetical List of Products

14-Mar-2005

A

Product Name:

Aniseed Syrup

B

Product Name:

Boston Crab Meat

C

Product Name:


Camembert Pierrot

Camarvon Tigers

To start a new group each time the first letter of the product name changes and then sort the products within each group alphabetically, set the properties in the Sorting and Grouping box as shown in the following table.

Field/Expression	SortOrder	Group Header	Group Footer	Group On	Group Interval	Keep Together
ProductName	Ascending	Yes	Yes	Prefix Characters	1	Whole Group
ProductName	Ascending	No	No	Each Value	1	No

To display the Sorting and Grouping box

1. Open the report in Design view.
2. On the View menu, click Sorting and Grouping (or click Sorting and Grouping  on the toolbar).

To print only the first letter of the name at the beginning of a new group, use this expression in the text box in the ProductName header:

=Left([ProductName],1)

Printing the numeric equivalent of a date

You can organize the records in a report by the numeric values for a period of time — a period such as a year, quarter, month, or week. For example, a year is divided into 53 calendar weeks. (The first and last weeks of the year are often partial weeks.) For example, the numeric value for the week from 18-Dec-94 to 24-Dec-94 is 52. Using these numeric values, you can group the orders that were shipped by the week of the year.

To find the numeric value of a date, you use the DatePart function. The format for this function is:

DatePart(interval, date)

The interval argument is the abbreviation for the part of the date that you want returned. Examples of valid abbreviations are "yyyy" for a four-digit year, "q" for a calendar quarter, and "m" for a month. The date argument is a field name or a literal date, such as "1-Jul-94."

Note Optionally, you can add two arguments to the DatePart function — one for the first day of the week, and the other for the first week of the year. In these arguments, you can accept the values that are set by the FirstWeekday and FirstWeek view options, or you can specify your own values.

The following table lists examples of the results that are returned for a field named "Holiday." The Holiday field can store the holidays that are celebrated in the countries where your company does business.

If the value in Holiday is	This expression	Returns
1-Jan-94	=DatePart("w",[Holiday])	7 (day of week)

31-Dec-94	=DatePart("ww",[Holiday])	53 (week of year)
31-Dec-94	=DatePart("yyyy",[Holiday])	1994 (four-digit number of year)

Comparing the results of multiple years

When you want to analyze the sales results from more than one year, it's convenient to group the results by a period of time, such as a quarter or month. That way, you can quickly see how the performance for a time period in one year compares with the same time period in another year. For example, suppose you want to see a Summary of Sales by Quarter report that shows the number of orders that were shipped and the sales totals.

Summary of Sales by Quarter
15-Mar-2005

Quarter: 1

<i>Year:</i>	<i>Orders Shipped:</i>	<i>Sales:</i>
1997	92	\$143,703
1998	178	\$276,330


Quarter: 2

<i>Year:</i>	<i>Orders Shipped:</i>	<i>Sales:</i>
1997	92	\$145,655
1998	90	\$161,362

To create the group headers and footers and specify the sort order for this report, you set the properties in the Sorting and Grouping box as shown in the following table. Note that you use an expression to group by the quarter when the orders were shipped.

Field/Expression	SortOrder	Group Header	Group Footer	Group On	Group Interval	Keep Together
=DatePart("q", [ShippedDate])	Ascending	Yes	Yes	Each Value	1	Yes
ShippedDate	Ascending	No	Yes	Year	1	No
ShippedDate	Ascending	No	No	Each Value	1	No
OrderID	Ascending	No	No	Each Value	1	No

To display the Sorting and Grouping box

1. Open the report in Design view.
2. On the View menu, click Sorting and Grouping (or click Sorting and Grouping  on the toolbar).

To print the quarter number at the beginning of a new group, you place a text box in the group header by using the same expression that you used in the Sorting and Grouping box:

=DatePart("q", [ShippedDate])

Calculating line item totals

Suppose you want an invoice report that displays information about an order. You need to calculate the extended price (the total sales for each product) for line items. You first create a query that provides the data for the report. You include in that query all of the fields that you need from all of the tables that you need, such as the Orders table, the Order Details table, and the Customers table. You can then create a calculated field in the query design grid that calculates the extended price for each product on the invoice.

To create a calculated field

1. In the Database window, under Objects, click Queries.
2. Click the query, and then click Design in the Database window.
3. Click in the Field row of an empty column in the query design grid.
4. Type a name, a colon (:), and an expression in the Field cell. To calculate the extended price, you can use an expression like this:

ExtendedPrice: CCur([Order Details].UnitPrice*[Quantity]*(1-[Discount])/100)*100

Note that when you create a calculated field in the query design grid, do not precede your expression with the = operator.

Identifying orders that were shipped late

Under the best of circumstances, orders are shipped on time. Occasionally, some orders may go out after the required date, and you want to identify these on a report. To mark all orders that were shipped late on a report that tracks shipments, you can print a check mark in a check box that is labeled Shipped Late. Because most orders are shipped on time, the check mark is easier to spot than text that spells out On time or Late.

The expression for this report compares the value in the Shipped Date field to the value in the Required Date field. If the value in the Shipped Date field is greater (that is, a later date) than the value in the Required Date field, the expression returns the value True, and the report shows a check mark in the check box. If the value is False, the report leaves the check box empty.

To add the ShippedLate check box to a report

1. In the Database window, under Objects, click Reports.
2. Click the report, and then click Design in the Database window.
3. In the toolbox, click the Check Box tool.

Note If the toolbox is not visible, on the View menu, click Toolbox.

4. Drag the pointer in the detail section of the report to create the check box.
5. Click the check box to select it.
6. On the View menu, choose Properties, and then click the All tab.
7. Set the properties for the check box as shown in the following table.

Property	Setting
Name	ShippedLate
ControlSource	=[Shipped Date]>[Required Date]
Visible	Yes

8. Close the property sheet.

Table of operators

Access supports a variety of operators, including arithmetic operators such as +, -, * (multiply), and / (divide), as well as comparison operators for comparing values, text operators for concatenating text, logical operators for determining true or false values, and other operators specific to Access. For details about using these operators, see the following sections:

Arithmetic operators

You use the arithmetic operators to calculate a value from two or more numbers or to change the sign of a number from positive to negative.

Operator	Purpose	Example
+	Sum two numbers.	[Subtotal]+[SalesTax]
-	Find the difference between two numbers or indicate the negative value of a number.	[Price]-[Discount]
*	Multiply two numbers.	[Quantity]*[Price]
/	Divide the first number by the second number.	[Total]/[ItemCount]
\	Round both numbers to integers, then divide the first number by the second number. Truncate the result to an integer.	[Registered]\[Rooms]
Mod	Divide the first number by the second number and return only the remainder.	[Registered] Mod [Rooms]
^	Raise a number to the power of an exponent.	Number ^ Exponent

Comparison operators

You use the comparison operators to compare values and return a result that is true, false, or null.

Operator	Purpose
<	Determine if the first value is less than the second value.
<=	Determine if the first value is less than or equal to the second value.
>	Determine if the first value is greater than the second value.
>=	Determine if the first value is greater than or equal to the second value.
=	Determine if the first value is equal to the second value.
<>	Determine if the first value is not equal to the second value.

Logical operators

You use the logical operators to combine two values and return a true, false, or null result. You might also see the logical operators referred to as Boolean operators.

Operator	Usage	Description
And	Expr1 And Expr2	True when Expr1 and Expr2 are true.
Or	Expr1 Or Expr2	True when either Expr1 or Expr2 is true.
Eqv	Expr1 Eqv Expr2	True when both Expr1 and Expr2 are true or both Expr1 and Expr2 are false.
Not	Not Expr	True when Expr is not true.
Xor	Expr1 Xor Expr2	True when either Expr1 is true, or Expr2 is true, but not both.

Concatenation operators

You use the concatenation operators to combine two text values into one.

Operator	Usage	Description
----------	-------	-------------

&	string1 & string2	Combines two strings to form one string.
+	string1 + string2	Combines two strings to form one string and propagates null values.

Special operators

You use the special operators as described in the following table.

Operator	Description	For More Information
Is (Not) Null	Determines whether a value is Null or Not Null.	
Like "pattern"	Matches string values by using wildcard operators ? and *.	Like Operator
Between val1 And val2	Determines whether a numeric or date value falls within a range.	Between...And Operator
In(string1,string2...)	Determines whether a string value is within a set of string values.	In Operator