

# Access 2007 Queries

## Table of contents

<b>INTRODUCTION TO QUERIES .....</b>	<b>2</b>
<b>QUERY JOINS .....</b>	<b>2</b>
INNER JOINS .....	2
OUTER JOINS .....	3
CHANGE A JOIN PROPERTY .....	4
REMOVING A JOIN.....	4
<b>CREATE QUERIES .....</b>	<b>5</b>
USING CRITERIA IN A QUERY .....	5
SPECIFY ALTERNATE CRITERIA SETS BY USING OR.....	5
ADD CRITERIA TO THE QUERY .....	5
CRITERIA FOR TEXT, MEMO, AND HYPERLINK FIELDS.....	6
CRITERIA FOR NUMBER, CURRENCY, AND AUTONUMBER FIELDS.....	8
CRITERIA FOR DATE/TIME FIELDS .....	9
CRITERIA FOR OTHER FIELDS .....	13
USE THE QUERY WIZARD TO BUILD A QUERY.....	13
RUN A QUERY .....	14
CREATE A SELECT QUERY IN DESIGN VIEW.....	14
BUILD A SELECT QUERY BY USING TABLES WITH A MANY-TO-MANY RELATIONSHIP .....	15
USE PARAMETERS IN QUERIES .....	16
CREATE A PARAMETER QUERY .....	16
RUN A PARAMETER QUERY .....	16
MATCHING PART OF A FIELD VALUE WITH A PARAMETER STRING.....	16
SPECIFY PARAMETER DATA TYPES IN THE QUERY PARAMETER DIALOG BOX .....	18
<b>USING EXPRESSIONS IN A QUERY.....</b>	<b>18</b>
USE EXPRESSIONS AS QUERY CRITERIA.....	18
ADD CALCULATIONS TO THE QUERY .....	19
ENTER EXPRESSION CRITERIA IN THE QUERY DESIGN GRID .....	20
UNDERSTANDING THE EXPRESSION BUILDER.....	20
START THE EXPRESSION BUILDER FROM A QUERY .....	22
CALCULATE GROUP TOTALS BY USING A TOTALS QUERY.....	22
<b>ABOUT ANSI-89 AND ANSI-92 WILDCARDS .....</b>	<b>23</b>
ANSI-89 WILDCARD CHARACTERS.....	23
ANSI-92 WILDCARD CHARACTERS.....	24
FIND WHICH ANSI STANDARD A DATABASE SUPPORTS .....	24

## Introduction to queries

Using a query, you can answer very specific questions about your data that would be difficult to answer by looking at table data directly. A query is a request for data results, for action on data, or for both. You can use a query to answer a simple question, to perform calculations, to combine data from different tables, or even to add, change, or delete table data. Queries that you use to retrieve data from a table or to make calculations are called *select queries*. Queries that add, change, or delete data are called *action queries*.

Sometimes you may want to review all of the data from a table, but at other times, you may want to review only the data from certain fields, or you may want to review data only if certain fields meet certain criteria. To review data using criteria, you use a select query.

A select query is a type of database object that shows information in Datasheet view. A query can get its data from one or more tables, from existing queries, or from a combination of the two. The tables or queries from which a query gets its data are referred to as its *recordsource*.

Whether you create simple select queries by using a wizard or by working in Design view, the steps are essentially the same. You choose the recordsource that you want to use and the fields that you want to include in the query — and, optionally, you specify criteria to refine the results.

After you have created a select query, you run it to see the results. Running a select query is simple — you just open it in Datasheet view. After saving, you can then reuse it whenever you need, for example, as a recordsource for a form, report, or another query.

## Query Joins

Relational databases consist, at the most basic level, of tables that bear logical relationships to each other. You use relationships to connect tables on fields that they have in common. A relationship is represented in a query by a join.

When you add tables to a query, Microsoft Office Access 2007 creates joins that are based on relationships that have been defined between the tables. You can manually create joins in queries, even if they do not represent relationships that have already been defined. If you use other queries (instead of, or in addition to, tables) as sources of data for a query, you can create joins between the source queries, and also between those queries and any tables that you use as sources of data.

Joins behave similarly to query criteria in that they establish rules that the data must match to be included in the query operations. Unlike criteria, joins also specify that each pair of rows that satisfy the join conditions will be combined in the recordset to form a single row.

When you include multiple tables in a query, you use joins to help you get the results you are looking for. A join helps a query return only the records from each table you want to see, based on how those tables are related to other tables in the query.

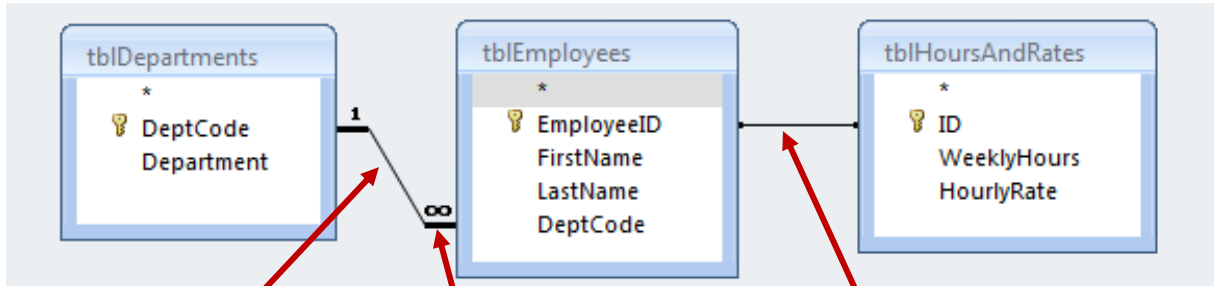
There are four basic types of joins: inner joins, outer joins, cross joins, and unequal joins. Inner and outer joins are most common.

### Inner joins

Inner joins are the most common type of join. They tell a query that rows from one of the joined tables correspond to rows in the other table, on the basis of the data in the joined fields. When a query with an inner join is run, only those rows where a common value exists in both of the joined tables will be included in the query operations.

Most of the time, you don't need to do anything to use an inner join. If you previously created relationships between tables in the Relationships window, Access automatically creates inner joins when you add related tables in query Design view. If referential integrity is enforced, Access also displays a "1" above the join line to show which table is on the "one" side of a one-to-many relationship and an infinity symbol ( $\infty$ ) to show which table is on the "many" side.

The purpose of using *referential integrity* is to prevent orphan records and to keep references synchronized so that you don't have any records referencing other records that no longer exist. You enforce referential integrity by enabling it for a table relationship. Once enforced, Access rejects any operation that would violate referential integrity for that table relationship. Access rejects updates that change the target of a reference, and also deletions that remove the target of a reference. See the Help section of Access 2007 for more information regarding referential integrity.



Join line

Visual indication that a relationship has been established between the two tables, and referential integrity has been enforced (one-to-many symbols).

Visual indication that this is either a manual join, or that a relationship has been established without enforcing referential integrity (no symbols).

Even if you haven't created relationships, Access automatically creates inner joins if you add two tables to a query and those tables each have a field with the same or compatible data type and one of the join fields is a primary key

## Outer Joins

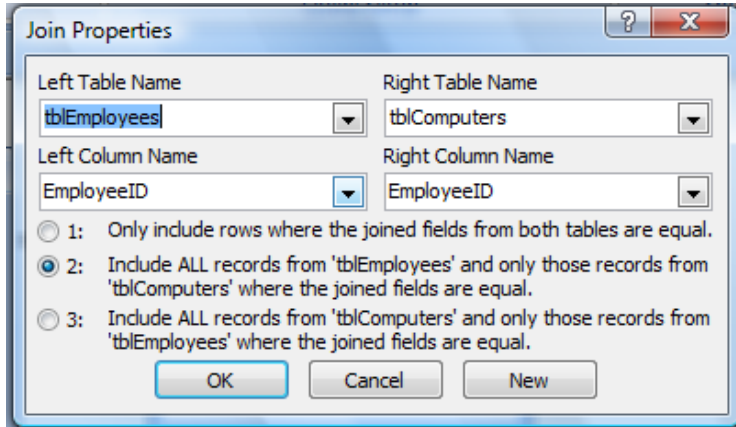
Outer joins tell a query that although some of the rows on both sides of the join correspond exactly, the query should include all of the rows from one table, and also those rows from the other table that share a common value on both sides of the join.

Because some of the rows on one side of an outer join will not have corresponding rows from the other table, some of the fields returned in the query results from that other table will be empty when the rows do not correspond.

When you choose option 2 or option 3, an arrow is shown on the relationship line. This arrow points to the side of the relationship that shows only matching rows

## Change a join property

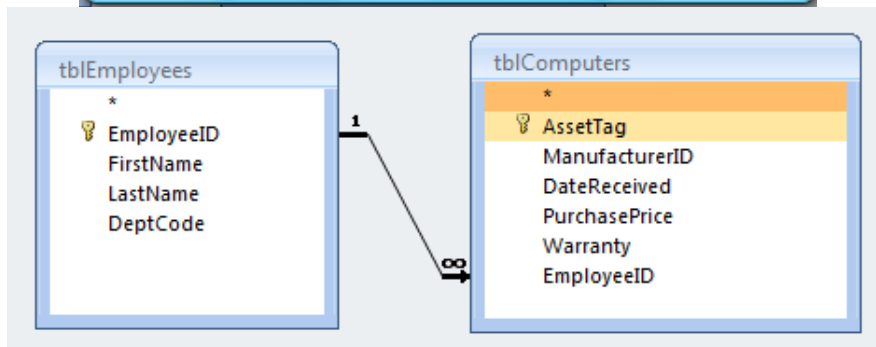
- 1) In query Design view, double-click the join you want to change.
  - a) The **Join Properties** dialog box appears.
- 2) In the **Join Properties** dialog box, note the choices listed beside option **2** and option **3**.
- 3) Click the option that you want to use, and then click **OK**.
  - a) Access displays the join and shows an arrow that points from the data source where all rows will be included to the data source where only those rows that satisfy the join condition will be included.



Note the join line after applying a left outer join (#2) in the Join Properties dialog box.

The arrow points to the table where only matches will display in the results of the query. In this case, you would see all of the employees in the query results, regardless of whether there was a computer assigned to them.

Otherwise, by default, only employees with a matching record in tblComputers would be displayed in the results (#1 in the join properties, an inner join, the default for a select query).



## Removing a join

If you create a join by mistake, for example, a join between two fields that have dissimilar data types, you can delete it. To delete the join:

- 1) In the query design grid, click the join you want to remove.
- 2) Press **DELETE**.
  - a) Alternatively, you can right-click the join you want to remove, and then click **Delete**

## Create queries

Sometimes, the process of building and using queries is a simple matter of selecting fields from a table, perhaps applying some criteria, and then viewing the results. But what if, as is more often the case, the data you need is spread out in more than one table? You may have cases in which a query that is based on one table gives you the information you need, but pulling data from another table would help to make the query results even clearer and more useful. For example, suppose you have a list of employee IDs that appear in your query results. You realize it would be more useful to view the employee name in the results, but the employee names are located in a different table. Fortunately, you can build a query that combines information from multiple sources.

### Using Criteria in a Query

A query criterion is a rule that identifies the records that you want to include in a query, and you use criteria when you do not want to see all the records in a given set of data. For example, the criterion  $>25$  AND  $<50$  returns values greater than 25 and less than 50. A criterion such as "Chicago" OR "Paris" OR "Moscow" returns only the records for those cities.

### Specify alternate criteria sets by using OR

Suppose you want to see all the records where City equals London and where at least one kind of contact information — either the address or the phone number — is available. You want to combine the criteria by using the OR operator, like this:

To specify alternate criteria, use both the **Criteria** and **Or** rows in the design grid. All records that meet the criteria defined either in the Criteria row or in the Or row are included in the result.

### Add criteria to the query

To restrict the records that are returned in the query results, you can specify one or more criteria.

You can think of a query criterion as a condition that you specify for a field. The criterion specifies a condition, based on field values, that expresses what you want to include in the query, such as "show only those records where the value of City is London".

- 1) Open the query in Design view.
- 2) Type the criterion for each field in the appropriate **Criteria** row.
  - a) Remember that the criteria you specify must match the data type of the field.
- 3) Switch to Datasheet view to see the results.

## Criteria for Text, Memo, and Hyperlink fields

### Text, Memo, Hyperlink fields

To include records that...	Use this criterion	Query result
Exactly match a value, such as China	"China"	Returns records where the CountryRegion field is set to China.
Do not match a value, such as Mexico	Not "Mexico"	Returns records where the CountryRegion field is set to a country/region other than Mexico.
Begin with the specified string, such as U	Like U*	Returns records for all countries/regions whose names start with "U", such as UK, USA, and so on. <b>Note:</b> When used in an expression, the asterisk (*) represents any string of characters — it is also called a wildcard character.
Do not begin with the specified string, such as U	Not Like U*	Returns records for all countries/regions whose names start with a character other than "U".
Contain the specified string, such as Korea	Like "*Korea*"	Returns records for all countries/regions that contain the string "Korea".
Do not contain the specified string, such as Korea	Not Like "*Korea*"	Returns records for all countries/regions that do not contain the string "Korea".
End with the specified string, such as "ina"	Like "*ina"	Returns records for all countries/regions whose names end in "ina", such as China and Argentina.
Do not end with the specified string, such as "ina"	Not Like "*ina"	Returns records for all countries/regions that do not end in "ina", such as China and Argentina.
Contain null (or missing) values	Is Null	Returns records where there is no value in the field.
Do not contain null values	Is Not Null	Returns records where the value is not missing in the field.
Contain zero-length strings	"" (a pair of quotes)	Returns records where the field is set to a blank (but not null) value.

## Text, Memo, Hyperlink fields

To include records that...	Use this criterion	Query result
Do not contain zero-length strings	Not ""	Returns records where the CountryRegion field has a nonblank value.
Contains null values or zero-length strings	"" Or Is Null	Returns records where there is either no value in the field, or the field is set to a blank value.
Is not empty or blank	Is Not Null And Not ""	Returns records where the CountryRegion field has a nonblank, non-null value.
Follow a value, such as Mexico, when sorted in alphabetical order	>= "Mexico"	Returns records of all countries/regions, beginning with Mexico and continuing through the end of the alphabet.
Fall within a specific range, such as A through D	Like "[A-D]*"	Returns records for countries/regions whose names start with the letters "A" through "D".
Match one of two values, such as USA or UK	"USA" Or "UK"	Returns records for USA and UK.
Contain one of the values in a list of values	In("France", "China", "Germany", "Japan")	Returns records for all countries/regions specified in the list.
Contain certain characters at a specific position in the field value	Right([CountryRegion], 1) = "y"	Returns records for all countries/regions where the last letter is "y".
Satisfy length requirements	Len([CountryRegion]) > 10	Returns records for countries/regions whose name is more than 10 characters long.
Match a specific pattern	Like "Chi*?"	Returns records for countries/regions, such as China and Chile, whose names are five characters long and the first three characters are "Chi".

A criterion that you specify for a Hyperlink field is, by default, applied to the display text portion of the field value. To specify criteria for the destination Uniform Resource Locator (URL) portion of the value, use the **HyperlinkPart** expression. The syntax for this expression is as follows: **HyperlinkPart([Table1].[Field1],1) = http://www.microsoft.com/** (Where **Table1** = name of the table containing the hyperlink field, **Field1** = hyperlink field, and **http://www.microsoft.com** is the URL.)

## Criteria for Number, Currency, and AutoNumber fields

### Number, Currency, AutoNumber fields

To include records that...	Use this criterion	Query Result
Exactly match a value, such as 100	100	Returns records where the unit price of the product is \$100.
Do not match a value, such as 1000	Not 1000	Returns records where the unit price of the product is not \$1000.
Contain a value smaller than a value, such as 100	< 100 <= 100	Returns records where the unit price is less than \$100 (<100). The second expression (<=100) displays records where the unit price is less than or equal to \$100.
Contain a value larger than a value, such as 99.99	>99.99 >=99.99	Returns records where the unit price is greater than \$99.99 (>99.99). The second expression displays records where the unit price is greater than or equal to \$99.99.
Contain one of the two values, such as 20 or 25	20 or 25	Returns records where the unit price is either \$20 or \$25.
Contain a value that falls with a range of values	>49.99 and <99.99 -or- Between 50 and 100	Returns records where the unit price is between (but not including) \$49.99 and \$99.99.
Contain a value that falls outside a range	<50 or >100	Returns records where the unit price is not between \$50 and \$100.
Contain one of many specific values	In(20, 25, 30)	Returns records where the unit price is either \$20, \$25, or \$30.
Contain a value that ends with the specified digits	Like "*4.99"	Returns records where the unit price ends with "4.99", such as \$4.99, \$14.99, \$24.99, and so on.
Contain null (or missing) values	Is Null	Returns records where no value is entered in the UnitPrice field.
Contain non-null values	Is Not Null	Returns records where the value is not missing in the UnitPrice field.



## Criteria for Date/Time fields

### Date/Time fields

To include records that ...	Use this criterion	Query result
Exactly match a value, such as 2/2/2006	#2/2/2006#	Returns records of transactions that took place on Feb 2, 2006. Remember to surround date values with the # character so that Access can distinguish between date values and text strings.
Do not match a value, such as 2/2/2006	Not #2/2/2006#	Returns records of transactions that took place on a day other than Feb 2, 2006.
Contain values that fall before a certain date, such as 2/2/2006	< #2/2/2006#	Returns records of transactions that took place before Feb 2, 2006.  To view transactions that took place on or before this date, use the <= operator instead of the < operator.
Contain values that fall after a certain date, such as 2/2/2006	> #2/2/2006#	Returns records of transactions that took place after Feb 2, 2006.  To view transactions that took place on or after this date, use the >= operator instead of the > operator.
Contain values that fall within a date range	>#2/2/2006# and <#2/4/2006#	Returns records where the transactions took place between Feb 2, 2006 and Feb 4, 2006.  You can also use the <b>Between</b> operator to filter for a range of values. For example, Between #2/2/2006# and #2/4/2006# is the same as >#2/2/2006# and <#2/4/2006# .
Contain values that fall outside a range	<#2/2/2006# or >#2/4/2006#	Returns records where the transactions took place before Feb 2, 2006 or after Feb 4, 2006.

## Date/Time fields

To include records that ...	Use this criterion	Query result
Contain one of two values, such as 2/2/2006 or 2/3/2006	#2/2/2006# or #2/3/2006#	Returns records of transactions that took place on either Feb 2, 2006 or Feb 3, 2006.
Contain one of many values	In (#2/1/2006#, #3/1/2006#, #4/1/2006#)	Returns records where the transactions took place on Feb 1, 2006, March 1, 2006, or April 1, 2006.
Contain a date that falls in a specific month (irrespective of year), such as December	DatePart("m", [SalesDate]) = 12	Returns records where the transactions took place in December of any year.
Contain a date that falls in a specific quarter (irrespective of year), such as the first quarter	DatePart("q", [SalesDate]) = 1	Returns records where the transactions took place in the first quarter of any year.
Contain today's date	Date()	Returns records of transactions that took place on the current day. If today's date is 2/2/2006, you see records where the OrderDate field is set to Feb 2, 2006.
Contain yesterday's date	Date()-1	Returns records of transactions that took place the day before the current day. If today's date is 2/2/2006, you see records for Feb 1, 2006.
Contain tomorrow's date	Date() + 1	Returns records of transactions that took place the day after the current day. If today's date is 2/2/2006, you see records for Feb 3, 2006.
Contain dates that fall during the current week	DatePart("ww", [SalesDate]) = DatePart("ww", Date()) and Year([SalesDate]) = Year(Date())	Returns records of transactions that took place during the current week. A week starts on Sunday and ends on Saturday.

## Date/Time fields

To include records that ...	Use this criterion	Query result
Contain dates that fell during the previous week	$\text{Year}([\text{SalesDate}]) * 53 + \text{DatePart}(\text{"ww"}, [\text{SalesDate}]) = \text{Year}(\text{Date}()) * 53 + \text{DatePart}(\text{"ww"}, \text{Date}()) - 1$	Returns records of transactions that took place during the last week. A week starts on Sunday and ends on Saturday.
Contain dates that fall during the following week	$\text{Year}([\text{SalesDate}]) * 53 + \text{DatePart}(\text{"ww"}, [\text{SalesDate}]) = \text{Year}(\text{Date}()) * 53 + \text{DatePart}(\text{"ww"}, \text{Date}()) + 1$	Returns records of transactions that will take place next week. A week starts on Sunday and ends on Saturday.
Contain a date that fell during the last 7 days	Between Date() and Date()-6	Returns records of transactions that took place during the last 7 days. If today's date is 2/2/2006, you see records for the period Jan 24, 2006 through Feb 2, 2006.
Contain a date that belongs to the current month	$\text{Year}([\text{SalesDate}]) = \text{Year}(\text{Now}()) \text{ And } \text{Month}([\text{SalesDate}]) = \text{Month}(\text{Now}())$	Returns records for the current month. If today's date is 2/2/2006, you see records for Feb 2006.
Contain a date that belongs to the previous month	$\text{Year}([\text{SalesDate}]) * 12 + \text{DatePart}(\text{"m"}, [\text{SalesDate}]) = \text{Year}(\text{Date}()) * 12 + \text{DatePart}(\text{"m"}, \text{Date}()) - 1$	Returns records for the previous month. If today's date is 2/2/2006, you see records for Jan 2006.
Contain a date that belongs to the next month	$\text{Year}([\text{SalesDate}]) * 12 + \text{DatePart}(\text{"m"}, [\text{SalesDate}]) = \text{Year}(\text{Date}()) * 12 + \text{DatePart}(\text{"m"}, \text{Date}()) + 1$	Returns records for the next month. If today's date is 2/2/2006, you see records for Mar 2006.
Contain a date that fell during the last 30 or 31 days	Between Date( ) And DateAdd("M", -1, Date( ))	A month's worth of sales records. If today's date is 2/2/2006, you see records for the period Jan 2, 2006. to Feb 2, 2006
Contain a date that belongs to the current quarter	$\text{Year}([\text{SalesDate}]) = \text{Year}(\text{Now}()) \text{ And } \text{DatePart}(\text{"q"}, \text{Date}()) = \text{DatePart}(\text{"q"}, \text{Now}())$	Returns records for the current quarter. If today's date is 2/2/2006, you see records for the first quarter of 2006.
Contain a date that belongs to the previous quarter	$\text{Year}([\text{SalesDate}]) * 4 + \text{DatePart}(\text{"q"}, [\text{SalesDate}]) = \text{Year}(\text{Date}()) * 4 + \text{DatePart}(\text{"q"}, \text{Date}()) - 1$	Returns records for the previous quarter. If today's date is 2/2/2006, you see records for the last quarter of 2005.

## Date/Time fields

To include records that ...	Use this criterion	Query result
Contain a date that belongs to the next quarter	$\text{Year}([\text{SalesDate}]) * 4 + \text{DatePart}("q", [\text{SalesDate}]) = \text{Year}(\text{Date}()) * 4 + \text{DatePart}("q", \text{Date}()) + 1$	Returns records for the next quarter. If today's date is 2/2/2006, you see records for the second quarter of 2006.
Contain a date that falls during the current year	$\text{Year}([\text{SalesDate}]) = \text{Year}(\text{Date}())$	Returns records for the current year. If today's date is 2/2/2006, you see records for the year 2006.
Contain a date that belongs to the previous year	$\text{Year}([\text{SalesDate}]) = \text{Year}(\text{Date}()) - 1$	Returns records of transactions that took place during the previous year. If today's date is 2/2/2006, you see records for the year 2005.
Contain a date that belongs to next year	$\text{Year}([\text{SalesDate}]) = \text{Year}(\text{Date}()) + 1$	Returns records of transactions with next year's date. If today's date is 2/2/2006, you see records for the year 2007.
Contain a date that falls between Jan 1 and today (year to date records)	$\text{Year}([\text{SalesDate}]) = \text{Year}(\text{Date}())$ and $\text{Month}([\text{SalesDate}]) \leq \text{Month}(\text{Date}())$ and $\text{Day}([\text{SalesDate}]) \leq \text{Day}(\text{Date}())$	Returns records of transactions with dates that fall between Jan 1 of the current year and today. If today's date is 2/2/2006, you see records for the period Jan 1, 2006 to 2/2/2006.
Contain a date that occurred in the past	$< \text{Date}()$	Returns records of transactions that took place before today.
Contain a date that occurs in the future	$> \text{Date}()$	Returns records of transactions that will take place after today.
Filter for null (or missing) values	Is Null	Returns records where the date of transaction is missing.
Filter for non-null values	Is Not Null	Returns records where the date of transaction is known.

## Criteria for other fields

**Yes/No fields:** In the **Criteria** row, type **Yes** to include records where the check box is selected. Type **No** to include records where the check box is not selected.

**Attachments:** In the **Criteria** row, type **Is Null** to include records that do not contain any attachments. Type **Is Not Null** to include records that contain attachments.

**Lookup fields:** There are two types of Lookup fields: those that look up values in an existing data source (by using a foreign key), and those that are based on a list of values specified when the Lookup field is created.

Lookup fields that are based on a list of specified values are of the Text data type, and valid criteria are the same as for other text fields.

The criteria you can use in a Lookup field based on values from an existing data source depend on the data type of the foreign key, rather than the data type of the data being looked up. For example, you may have a Lookup field that displays Department Name, but uses a foreign key that is of the Number data type. Because the field stores a number instead of text, you use criteria that work for numbers; that is, =2 (#2 in the data source could equal the name “History.”)

If you do not know the data type of the foreign key, you can inspect the source table in Design view to determine the data types of the field.

## Use the Query Wizard to build a query

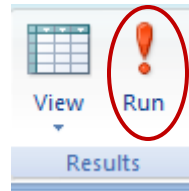
- 1) If you are not sure, check that the tables have a defined relationship in the Relationships window
- 2) **Create** tab on the Ribbon / **Other** group / **Query Wizard**.
- 3) In the **New Query** dialog box, click **Simple Query Wizard**, and then click **OK**.
- 4) In the **Tables/Queries** combo box, click the table that contains the basic information you want included in your query.
- 5) In the **Available Fields** list, click the first field you want to include in your query, and then click the single right arrow button to move that field to the **Selected Fields** list. Do the same with each additional field from that table that you want to include in your query. These can be fields that you want returned in the query output, or fields that you want to use to limit the rows in the output by applying criteria.
- 6) If you want to include additional fields that are not in the current table, in the **Tables/Queries** combo box, click the table that contains the related data you want to use to enhance your query results.
- 7) Add the fields that you want to use to enhance your query results to the **Selected Fields** list and then click **Next**.
- 8) Under **Would you like a detail or summary query?** Click either **Detail** or **Summary**.
  - a) If you do not want your query to perform any aggregate functions (**Sum**, **Avg**, **Min**, **Max**, **Count**, **StDev**, or **Var**), choose a detail query. If you do want your query to perform an aggregate function, choose a summary query. After you make your choice, click **Next**.
- 9) Click **Finish** to view the results.

## Run a query

A query is a set of instructions that you can use for working with data. You run a query to perform these instructions. In addition to returning results — which can be sorted, grouped, or filtered — a query can also create, copy, delete, or change data.

- 1) Locate the query in the Navigation Pane.
- 2) Do one of the following:
  - a) Double-click the query you want to run.
  - b) Click the query you want to run, and then press ENTER.

If the query you want to run is currently open in **Design view**, you can also run it by clicking **Run** in the **Results** group on the **Design** tab on the Ribbon.



## Create a select query in Design view

A select query is used to create subsets of data that you can use to answer specific questions. It can also be used to supply data to other database objects. Once you create a select query, you can use it whenever you need.

- 1) On the **Create** tab, in the **Other** group, click **Query Design**.
  - a) The Show Table dialog box appears.
- 2) In the dialog box, click the table that you want to use in the query, click **Add** to place the table in the upper section of the designer, and then click **Close**.
  - a) Alternatively, you can also double-click the table to add it to the upper section of the designer, and then click Close.
- 3) Add the fields that you want to use in your query to the design grid.
  - a) You can double-click each field, or drag and drop each field on a blank cell in the Field row.
- 4) In the query design grid, use the **Criteria** row to enter field criteria.
  - a) To use a field criterion without displaying the field in the query results, clear the check box in the **Show** row for that field.
  - b) To sort the results based on the values in a field, in the query design grid, click **Ascending** or **Descending** (depending on which way you want to sort the records) in the **Sort** row for that field.
- 5) **Design** tab / **Results** group / **Run**.
  - a) Access displays the query output in Datasheet view.

## Build a select query by using tables with a many-to-many relationship

Often, data in two tables are related to each other through a third table. This is usually the case because the data between the first two tables are related in a many-to-many relationship. Often, it is good database design practice to split a many-to-many relationship between two tables into two one-to-many relationships involving three tables. You do this by creating a third table called a junction table, or a relationship table, that has a primary key and a foreign key for each of the other tables. A one-to-many relationship is then created between each foreign key in the junction table and the corresponding primary key of one of the other tables. In such cases, you need to include all three tables in your query, even if you want to retrieve data from only two of them.

### 1) Create tab / Other group / Query Design.

a) The **Show Table** dialog box opens.

### 2) In the **Show Table** dialog box, double-click the two tables that contain the data you want to include in your query and also the junction table that links them, and then click **Close**.

a) All three tables appear in the query design workspace, joined on the appropriate fields.

### 3) Double-click each of the fields that you want to use in your query results.

a) Each field then appears in the query design grid.

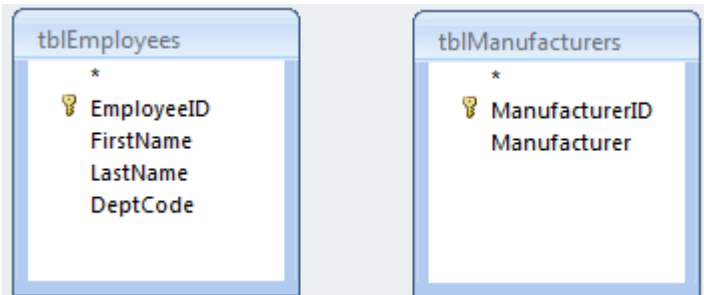
### 4) In the query design grid, use the **Criteria** row to enter field criteria.

a) To use a field criterion without displaying the field in the query results, clear the check box in the **Show** row for that field.

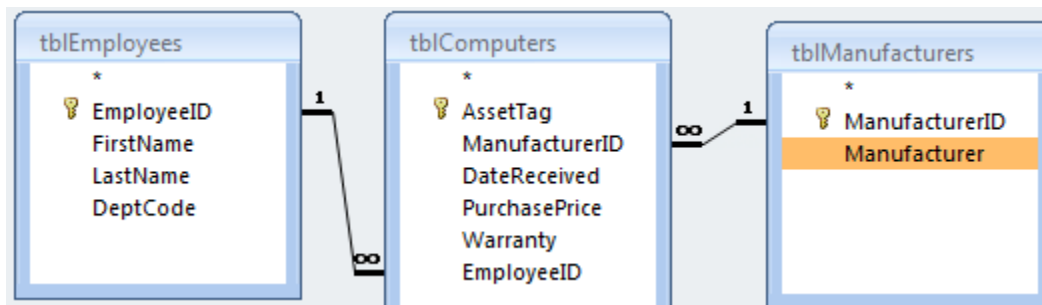
b) To sort the results based on the values in a field, in the query design grid, click **Ascending** or **Descending** (depending on which way you want to sort the records) in the **Sort** row for that field.

### 5) Design tab / Results group / Run.

a) Access displays the query output in Datasheet view.



Example: If you need you see the names of employees, and the name of the manufacturer for the computer they've been assigned, by just bringing in these two tables, Access would not know how to relate the tables and would return inaccurate results. There is no common field to link the table together.



By adding a junction table to the design, Access can accurately display results by using tblComputers as a “drive-thru” table. It is linked, individually, to tblEmployees and tblManufacturers, and therefore can act as a middle-man for your query design. You do not have to use any fields from the junction table. If you now add the employee name fields and the manufacturer name field, Access will display accurate results by matching those fields through tblComputers.



## Use parameters in queries

Queries are useful for working with just those fields from a table that pertain to the task at hand. When you want to further restrict the data that you work with, based on the value contained in a field, you can use criteria in your query. Criteria are rules that you include in the design of a query — these rules specify values or patterns that you want the fields to match or contain to be returned by the query.

When you want a query to prompt you for a value or pattern every time you run it, you can create a parameter query. A parameter query is not a separate kind of query, rather, it extends the flexibility of a query.

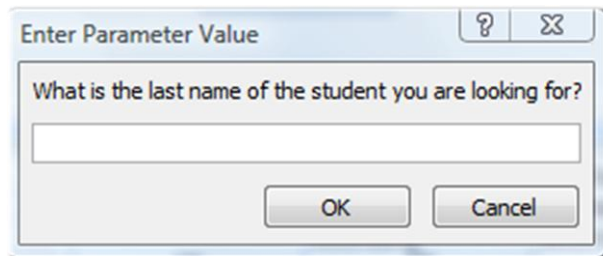
Creating a parameter query is as easy as creating a query that uses criteria. You can design a query to prompt you for one piece of information, such as a part number, or for more than one piece of information, such as two dates. For each parameter, a parameter query displays a separate dialog box that prompts you for a value for that parameter.

## Create a parameter query

- 1) Create a select query, and then open the query in Design view.
- 2) In the **Criteria** row of a field for which you want a parameter applied, type the text that you want the parameter dialog box to display, surrounded by square brackets, for example:

**[Type the Student's last name:]**

- 3) When you run the parameter query, the prompt appears in a dialog box without the square brackets.



- 4) Repeat step 2 for each parameter you want the query to collect and apply.

But what if you don't know what values you can specify? To make your parameter more flexible, you can use wildcard characters as part of the prompt. See the Wildcard section of this handout, starting on page 23.

## Run a parameter query

A parameter query prompts you for a value when you run it. When you supply the value, the parameter query applies it as a field criterion. Which field it applies the criterion to is specified in the query design. If you do not supply a value when prompted, the parameter query interprets your input as an empty string.

- 1) Locate the query in the Navigation Pane.
- 2) Do one of the following:
  - a) Double-click the query you want to run.
  - b) Click the query you want to run, then press ENTER.
- 3) When the parameter prompt appears, enter a value to apply as a criterion.

## Matching part of a field value with a parameter string

You may want a little variability in the way your query applies a parameter. For example, you might want a query to accept a text string and match that to any part of a field. You can do this by using the **Like** keyword in combination with wildcard characters. For example, you might want your query to prompt for a country/region of origin, but to match whenever the relevant field value contains the parameter string. To do this:



- 1) Create a select query, and then open the query in Design view.
- 2) In the **Criteria** row of the field for which you want the parameter applied, type **Like "\*" & [Type the text that you want to use as a prompt, and then type ] & "\*" & "**  
 Ex.: **Like "\*" & [Enter the country/region of origin] & "\*" & "**
  - a) When you run the parameter query, the prompt appears in the dialog box without the square brackets, and without the **Like** keyword or wildcard characters.
  - b) When the query accepts the parameter, it matches on values that contain the parameter string. For example, the parameter string **us** matches rows where the parameter field has a value of Australia and rows where the value is USA.

You can also use the **Like** keyword and wildcard characters to specify that a parameter should match the beginning or ending of a field value. To match the beginning of a field value, omit the quotation marks, the wildcard character, and the ampersand (&) that precede the opening square bracket. To match the ending of a field value, omit the ampersand, the quotation marks, and the wildcard character that follow the closing square bracket.

Field:	LastName
Table:	tblEmployees
Sort:	Ascending
Show:	<input checked="" type="checkbox"/>
Criteria:	Like [Last name, or beginning initial] & "*" & "
or:	

Using the **Like** keyword gives more flexibility to the prompt.

Using this example, a user could enter the first initial of the last name, a series of characters, or a full last name.

Enter Parameter Value	
Last name, or beginning initial	
<input type="text" value="b"/>	
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

LastName	FirstName
Ballantyne	Carl
Barefoot	Karen
Barnett	Amy
Barrows	Helene
Bernstein	Edward
Bonafede	Mike
Brown	Andrea
Byam	Vishna

Enter Parameter Value	
Last name, or beginning initial	
<input type="text" value="bar"/>	
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

LastName	FirstName
Barefoot	Karen
Barnett	Amy
Barrows	Helene
*	

Enter Parameter Value	
Last name, or beginning initial	
<input type="text" value="barnett"/>	
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

LastName	FirstName
Barnett	Amy
*	

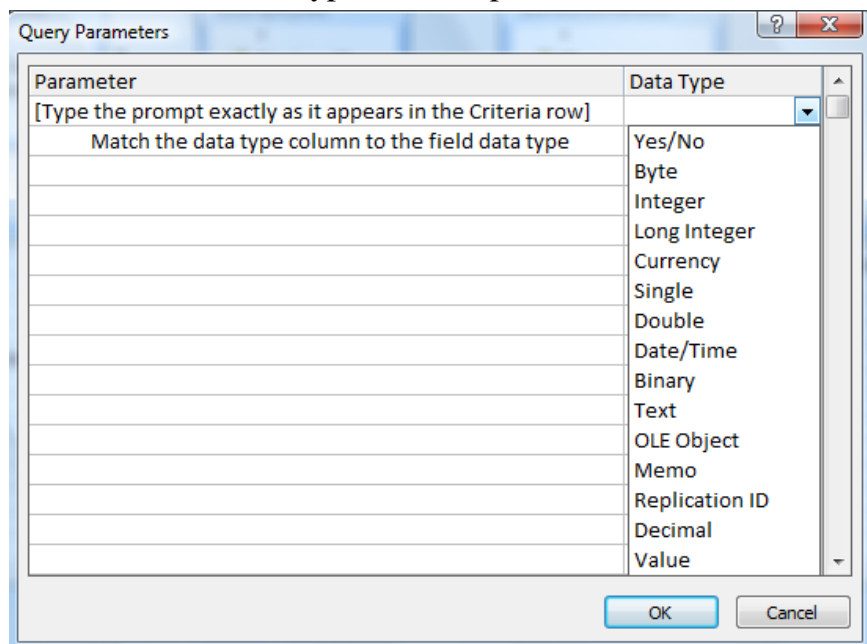
## Specify parameter data types in the Query Parameter dialog box

You can also specify what type of data a parameter should accept. You can set the data type for any parameter, but it is especially important to set the data type for numeric, currency, or date/time data. When you specify the data type that a parameter should accept, users see a more helpful error message if they enter the wrong type of data, such as entering text when currency is expected.

Note: If a parameter is set to accept text data, any input is interpreted as text, and no error message is displayed.

To specify the data type for parameters in a query, use the following procedure:

- 1) With the query open in **Design view**, on the **Design** tab, in the **Show/Hide** group, click **Parameters**.
- 2) In the **Query Parameters** dialog box, in the **Parameter** column, type the prompt for each parameter for which you want to specify the data type. Make sure that each parameter matches the prompt that you use in the Criteria row of the query design grid.
- 3) In the **Data Type** column, select the data type for each parameter.



## Using Expressions in a Query

An expression is a combination of mathematical or logical operators, constants, functions, and names of fields, controls, and properties that evaluates to a single value. You use an expression when you need data that does not reside directly in a table. For example, the expression `[UnitPrice]*[Quantity]` multiplies the value in the UnitPrice field by the value in the Quantity field. You can use expressions in a wide variety of ways, and the process of creating and using them can become quite complex.

### Use expressions as query criteria

You can use an expression to define criteria in a query. Access then returns only those rows that match the criteria. For example, suppose that you want to see all the orders whose shipped date occurred in the year 2004. To enter the criteria, you type the following expression in the Criteria cell for the Date/Time column in your query. This example uses a Date/Time column called ShippedDate. To define a date range, enter your criteria in this manner: **Between #1/1/2004# And #12/31/2004#**

The **ShippedDate** column will look similar to the following.

Field:	ShippedDate
Table:	Orders
Total:	Where
Sort:	
Show:	<input type="checkbox"/>
Criteria:	Between #1/1/2004# And #12/31/2004#
or:	

For each record in the Orders table, if the value in the ShippedDate column falls in the date range that you specify, the record is included in the query output. Note that in the expression, you enclose the dates with pound signs (#). Access treats a value enclosed in pound signs as a Date/Time data type. Treating those values as date/time data enables you to perform calculations on those values, such as subtracting one date from another. However, if you do not type the pound signs, and if Access understands that what you typed is a date, it will automatically add the pound signs for you. Therefore, if you have entered dates as criteria, and do not see the pound signs, review what you have typed, as Access does not recognize it as a date. (Examples of correct date entry are: 1/1/07, or 1/1/2007.)

### Add calculations to the query

A well designed database does not store simple calculated values in tables. For example, a table might store a person's date of birth but not their current age. If you know both today's date and the person's date of birth, you can always calculate their current age, so there is no need to store that in the table. Instead, you create a query that calculates and displays the pertinent value. The calculations are made every time you run the query, so if the underlying data changes, so do your calculated results. When creating basic calculations, use the following syntax:

$$\text{Label: [FieldName] } + \text{ [FieldName] or number} \\ /$$

Label:	[Fieldname]	Operator	[Fieldname]	Number
Name of new field not in the query yet.	Name of the field you are doing math on. (Must be enclosed in square brackets.)	What type of math are you doing?	Name of the field you are doing math on. (Must be enclosed in square brackets.)	If doing math using a number, do not enclose in square brackets.

Examples:

If you wanted to figure out the gross pay for your employees, using your fields called **WeeklyHours** and **PayRate**, you would type:

$$\text{GrossPay:[WeeklyHours]*[PayRate]}$$

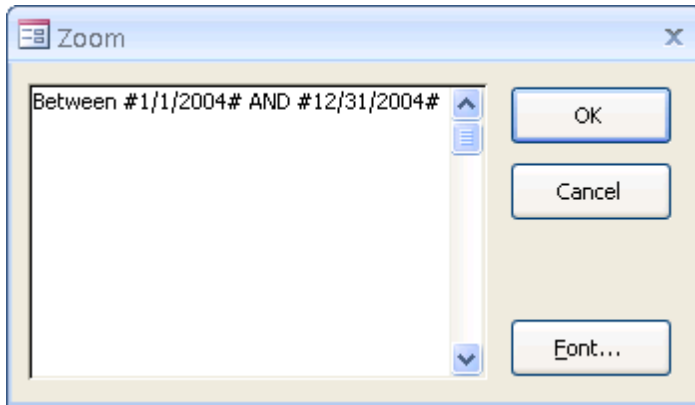
If you wanted to see how much the new price of your products would be with a 4% increase, using your field called **ProductPrice**, you would type:

$$\text{ProductIncrease:[ProductPrice]*1.04}$$

Note: Do not put a number value in square brackets. Square brackets are used to indicate the name of an object in your database.

## Enter expression criteria in the query design grid

- 1) Open your query in **Design View**.
- 2) Click in the **Criteria** cell in the column for which you want to enter your criteria.
  - a) To manually create your expression, type your criteria expression.



- If you want a larger area in which to enter an expression, press SHIFT+F2 to display the **Zoom** box.
- Alternatively, you can right click in the criteria row under the column you are adding the expression to and choose “**Zoom...**”


- b) To create your expression by using the Expression Builder, on the **Design** tab, in the **Query Setup** group, click **Builder**.
      - Alternatively, you can right click in the criteria row under the column you are adding the expression to and choose “**Build...**”

**Note:** Do not precede the criteria expression with the = operator; instead, begin the expression with a descriptive label followed by a colon. For example, type **Extended Price:** to provide the label for an expression that creates a calculated field called **Extended Price**. Then, enter the criteria for your expression after the colon.

## Understanding the Expression Builder

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

You can use the Expression Builder to create a new expression, or you can select from prebuilt expressions, including expressions for displaying page numbers, the current date, and the current date and time.

You can start the Expression Builder from most of the places in Microsoft Office Access 2007 where you would write expressions manually, such as the **Control Source** property of a control or the **Validation Rule** property of a table field. As a rule, if you see the **Build button** , you can click it to start the Expression Builder.



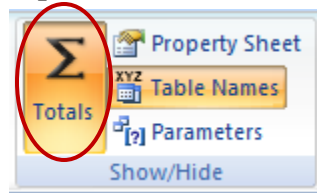
## Start the Expression Builder from a query

- 1) Open your query in **Design View**.
- 2) 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.
- 3) On the **Design** tab, in the **Query Setup** group, click **Builder**.
  - Alternatively, you can right click in the criteria row under the column you are adding the expression to and choose "Build..."

## Calculate group totals by using a totals query

The steps in this section explain how to create a totals query that calculates subtotals across groups of data. As you proceed, remember that by default, a totals query can include only the field or fields that contain your group data, such as a "categories" field, and the field that contains the data that you want to sum, such as a "sales" field. Totals queries cannot include other fields that describe the items in a category. If you want to see that descriptive data, you can create a second select query that combines the fields in your totals query with the additional data fields.


- 1) On the **Create** tab, in the **Other** group, click **Query Design**.
- 2) In the **Show Table** dialog box, double-click the table that you want to use in your query, and then click **Close**.
- 3) Double-click the field that you want to sum. Make sure that the field is set to either the Number or Currency data type.
  - a) If you try to sum values in non-numeric fields, such as a Text field, Access displays the **Data type mismatch in criteria expression** error message when you try to run the query.
  - b) **Note:** You can add additional numeric fields to the grid if you want to calculate grand totals for those fields. A totals query can calculate grand totals for more than one column.
- 4) On the **Design** tab, in the **Show/Hide** group, click **Totals**:



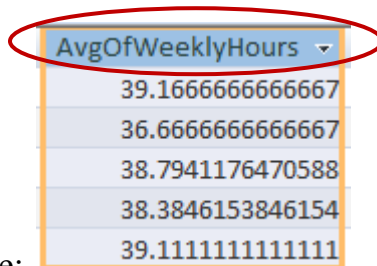
- 5) The **Total** row appears in the design grid and **Group By** appears in the cell by default.
- 6) Change the value in the cell in the **Total** row to whatever function you require from the drop down list.

Field:	Department	WeeklyHours	WeeklyGross: [Weel
Table:	tblDepartments	tblHoursAndRates	
Total:	Group By	Avg	Group By
Sort:			Group By
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Sum
Criteria:			Avg
or:			Min
			Max
			Count
			StDev
			Var
			First
			Last
			Expression
			Where



7) Click **Run**  to run the query and display the results in Datasheet view.

a) Note that Access appends "FunctionNameOf" to the beginning of the name of the field that you performed a calculation on:



AvgOfWeeklyHours
39.1666666666667
36.6666666666667
38.7941176470588
38.3846153846154
39.1111111111111

Example:

- To change the column heading to something more meaningful, switch back to **Design view**, and right-click in the **Field** row of the column in the design grid / choose **Properties** / change the “**Caption**” property to whatever you want displayed as a column label / close the Properties box.

8) Save the query.

## About ANSI-89 and ANSI-92 Wildcards

Access supports two sets of wildcard characters because it supports two standards for Structured Query Language — ANSI-89 and ANSI-92. As a rule, you use the ANSI-89 wildcards when you run queries and find-and-replace operations against Access databases — .mdb and .accdb files. You use the ANSI-92 wildcards when you run queries against Access projects — Access files connected to Microsoft SQL Server databases. Access projects use the ANSI-92 standard because SQL Server uses that standard.

### ANSI-89 wildcard characters

Use this set of wildcard characters when you run select and update queries against an Access database, but you do not use them in queries run against an Access project.

Character	Description	Example
*	Matches any number of characters. You can use the asterisk (*) anywhere in a character string.	<b>wh*</b> finds what, white, and why, but not awhile or watch.
?	Matches any single alphabetic character.	<b>B?ll</b> finds ball, bell, and bill.
[ ]	Matches any single character within the brackets.	<b>B[ae]ll</b> finds ball and bell, but not bill.
!	Matches any character not in the brackets.	<b>b[!ae]ll</b> finds bill and bull, but not ball or bell.
-	Matches any one of a range of characters. You must specify the range in ascending order (A to Z, not Z to A).	<b>b[a-c]d</b> finds bad, bbd, and bcd.
#	Matches any single numeric character.	<b>1#3</b> finds 103, 113, and 123.

## ANSI-92 wildcard characters

Use this set of wildcard characters when you run select and update queries against Access projects (.adp files), and when using either type of query to search databases set to use the ANSI-92 standard.


Character	Description	Example
%	Matches any number of characters. It can be used as the first or last character in the character string.	<b>wh%</b> finds what, white, and why, but not awhile or watch.
_	Matches any single alphabetic character.	<b>B_</b> ll finds ball, bell, and bill.
[ ]	Matches any single character within the brackets.	<b>B[ae]</b> ll finds ball and bell, but not bill.
^	Matches any character not in the brackets.	<b>b[^ae]</b> ll finds bill and bull, but not ball or bell.
-	Matches any one of a range of characters. You must specify the range in ascending order (A to Z, not Z to A).	<b>b[a-c]</b> d finds bad, bbd, and bcd.

### Notes:

- To find wildcard characters that reside in your data, enclose the character that you want to find in brackets, like so: [#]. Follow this rule when you search for asterisks (\*), question marks (?), pound signs (#), opening brackets ([), and hyphens (-). Do not use brackets when searching for exclamation points (!) or closing brackets (]). To find those characters by using the **Find and Replace** dialog box, type the character in the **Find What** box with no surrounding brackets. You follow the same approach when finding the characters by using a query. For example, the following syntax returns all records that contain an exclamation point, regardless of where the character resides in your data: **Like "!\*"**.
- If you are searching for a hyphen and other characters simultaneously, place the hyphen before or after all the other characters inside the brackets, like so: [-#\*] or [#\*-]. However, if you have an exclamation point (!) after the opening bracket, place the hyphen after the exclamation point: [!-].
- To search for a pair of opening and closing brackets ([]), you must enclose both characters in brackets, like so: [[]].

## Find which ANSI standard a database supports

Follow these steps to find and optionally change the ANSI setting for a given database.

- 1) Click the Microsoft Office Button , and then click **Access Options..**
  - a) The Access Options dialog box appears.
- 2) Click **Object Designers**, and in the **Query design section**, under **SQL Server Compatible Syntax (ANSI 92)**, do one of the following:
  - a) Select “This database” to change the open database to the ANSI-92 standard.
  - b) Clear the check box to set the open database to the ANSI-89 standard.
  - c) Select “Default for new databases” to set all new databases created with the open instance of Access to the ANSI-92 standard.
  - d) Clear the check box to set all new databases to the ANSI-89 standard / click OK