MS Access - Queries

An Access query is a question that you ask about the information stored in Access tables. You build queries with the Access query tools. Your query can be a simple question about data in a single table, or it can be a more complex question about information stored in several tables. Queries are flexible. They allow you to look at your data in virtually any way you can imagine. Most database systems are continually evolving and changing over time. Quite often, the original purpose of a database is very different from its current use. Here's just a sampling of what you can do with Access queries:

- ***Choose tables.*** You can obtain information from a single table or from many tables that are related by some common data. Suppose you're interested in seeing the customer name along with the items purchased by each type of customer. When using several tables, Access combines the data as a single recordset (a set of records that meet given criterion).
- ***Choose fields.*** Specify which fields from each table you want to see in the recordset. For example, you can select the customer name, zip code, sales date, and invoice number from tblCustomers and tblSales.
- ***Provide criteria.*** Record selection is based on selection criteria. For example, you might want to see records for only a certain category of products.
- ***Sort records.*** You might want to sort records in a specific order. For example, you might need to see customer contacts sorted by last name and first name.
- ***Perform calculations.*** Use queries to perform calculations such as averages, totals, or counts of data in records.
- ***Create tables.*** Create a brand-new table based on data returned by a query.
- ***Display query data on forms and reports.*** The recordset you create from a query might have just the right fields and data needed for a report or form. Basing a report or form on a query means that, every time you print the report or open the form, you see the most current information contained in the tables.
- ***Use a query as a source of data for other queries (subquery).*** You can create queries that are based on records returned by another query. This is very useful for performing ad hoc queries, where you might repeatedly make small changes to them criteria. In this case, the second query filters the first query's results.
- ***Make changes to data in tables.*** Action queries can modify multiple rows in the underlying tables as a single operation. Action queries are frequently used to maintain data, such as updating values in specific fields, appending new data, or deleting obsolete information.

Access combines a query's records and, when executed, displays them in Datasheet view by default. The set of records returned by a query is commonly called (oddly enough) a recordset. A recordset is a dynamic set of records. The recordset returned by a query is not stored within the database unless you've directed Access to build a table from those records.

The query design window has three primary views:

- ***Design view:*** Where you create the query.
- ***Datasheet view:*** Displays the records returned by the query.

- ***SQL view:*** Displays the SQL statement behind a query.

The Query Designer consists of two sections:

- ***The table/query pane (top):*** This is where tables or queries and their respective field lists are added to the query's design. You'll see a separate field list for each object to add. Each field list contains the names of all the fields in the respective table or query. You can resize a field list by clicking the edges and dragging it to a different size. You may want to resize a field list so that all a table's fields are visible.
- ***The Query by Design (QBD) grid (bottom):*** The QBD grid holds the field names involved in the query and any criteria used to select records. Each column in the QBD grid contains information about a single field from a table or query contained within the upper pane.

The QBD grid has six labeled rows:

- ***Field:*** This is where field names are entered or added.
- ***Table:*** This row shows the table the field is from. This is useful in queries with multiple tables.
- ***Sort:*** This row enables sorting instructions for the query.
- ***Show:*** This row determines whether to display the field in the returned recordset.
- ***Criteria:*** This row consists of the criteria that filter the returned records.
- ***Or:*** This row is the first of a number of rows to which you can add multiple query criteria.

Understanding selection criteria

Selection criteria are filtering rules applied to data as they're extracted from the database. Selection criteria tell Access which records you want to look at in the recordset. A typical criterion might be "all sellers," or "only those vehicles that are not trucks," or "products with retail prices greater than $75."

You specify criteria in the Criteria row of the QBD grid. You designate criteria as an expression. The expression can be simple (like "trucks" or "not trucks"), or it can take the form of complex expressions using built-in Access functions.

Entering simple string criteria

Open ***qry_cars*** and add the text condition to field Category. We are looking for the category Cars.

You could enter the criteria expression in any of these other ways:

- CARS
- =CARS
- "CARS"
- ="CARS"

By default, Access is not case sensitive, so any form of the word cars works just as well as this query's criteria. You could just as well enter Not Cars in the criteria for the column to return all products that are not cars (trucks, vans, and so on).

To erase the criteria in the cell, select the contents and press Delete, or select the contents and right-click Cut from the shortcut menu that appears.

Entering other simple criteria

You can also specify criteria for Numeric, Date, and Yes/No fields. Simply enter the example data in the criteria field just as you did for text fields. In almost every case, Access understands the criteria you enter and adjusts to correctly apply the criteria to the query's fields.
It's also possible to add more than one criteria to a query. For example, suppose that you want to look only at contacts who live in Connecticut and have been customers since January 1, 2010 (where OrigCustDate is greater than or equal to January 1, 2010). This query requires criteria in both the State and OrigCustDate fields.
To do this, it's critical that you place both examples on the same criteria row. Follow these steps to create this query:

1. Create a new query starting with tblContacts.
2. Add OrigCustDate, LastName, FirstName, and State to the QBD grid.
3. Enter CT in the Criteria cell in the State column.
4. Enter >= 01/01/2010 in the Criteria cell in the OrigCustDate column. Access adds pound sign characters (#) around the date in the criteria box.
5. Run the query.
6. Name query as Criteria_Contacts.

Access uses comparison operators to compare Date fields to a value. These operators include less than (<), greater than (>), equal to (=), or a combination of these operators.
Notice that Access automatically adds pound sign (#) delimiters around the date value. Access uses these delimiters to distinguish between date and text data. The pound signs are just like the quote marks Access added to the "Cars" criteria. Because OrigCustDate is a DateTime field, Access understands what you want and inserts the proper delimiters for you.

Creating Multi-table Queries

After you create the tables for your database and decide how the tables are related to one another, you're ready to build multi-table queries to obtain information from several related tables. A multi-table query presents data as if it existed in one large table.
Create new query according to this:

1. Create a new query by clicking the Query Design button on the Create tab of the Ribbon.
2. Add tblCustomers, tblSales, tblSalesLineItems, and tblProducts by double-clicking each table's name in the Show Table dialog box.
3. Add there the fields Company, SaleDate, SellingPrice, Description.
4. Find data where SellingPrice is >30.
5. Run query.

Using Operators and Expressions in Access (Operators_Expressions.accdb)

Operators

Operators let you compare values, put text strings together, format data, and perform a wide variety of tasks. You use operators to instruct Access to perform a specific action against one or more operands. The combination of operators and operands is known as an expression.

Types of operators

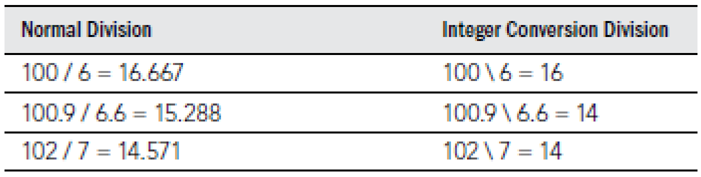
- Mathematical
- Comparison
- String
- Boolean (logical)
- Miscellaneous

Basic mathematical operators:

+	Addition
–	Subtraction
*	Multiplication
/	Division
\	Integer division
^	Exponentiation
Mod	Modulo

*The integer division operator: *

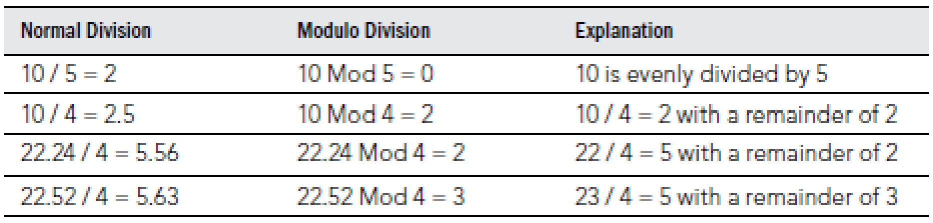
The integer division operator (\) takes any two numbers (number1 and number2), rounds them up or down to integers, divides the first by the second (number1 / number2), and then drops the decimal portion, leaving only the integer value:



Normal Division	Integer Conversion Division
100 / 6 = 16.667	100 \ 6 = 16
100.9 / 6.6 = 15.288	100.9 \ 6.6 = 14
102 / 7 = 14.571	102 \ 7 = 14

The modulo division operator: Mod

The modulo operator (Mod) takes any two numbers (number1 and number2), rounds them up or down to integers, divides the first by the second (number1 / number2), and then returns the remainder.



Normal Division	Modulo Division	Explanation
10 / 5 = 2	10 Mod 5 = 0	10 is evenly divided by 5
10 / 4 = 2.5	10 Mod 4 = 2	10 / 4 = 2 with a remainder of 2
22.24 / 4 = 5.56	22.24 Mod 4 = 2	22 / 4 = 5 with a remainder of 2
22.52 / 4 = 5.63	22.52 Mod 4 = 3	23 / 4 = 5 with a remainder of 3

Comparison operators

Comparison operators compare two values or expressions in an equation. There are six basic comparison operators:

=	Equal
<>	Not equal
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

String operators

Access has three string operators for working with strings. Unlike the mathematical and logical operators, the string operators are specifically designed to work with the string data type:

&	Concatenates operands.
Like	Operands are similar.
Not Like	Operands are dissimilar.

Understanding complex criteria

For many queries, complex criteria consist of a series of And*s* and Or*s*, as in these examples:

- State must be Connecticut or Texas.
- City must be Sunnyville and state must be Georgia.
- State must be MA or MO and city must be Springfield.

Task:

1. Open query Figure-03.
2. Set the conditions and find values where the State is CA or AZ and Category is not equal to Cars.
3. Run Query.

Options for conditions: “CA” Or “AZ” and <>”Cars”

“CA” and <>”Cars”
“AZ” and <>”Cars”

The Like operator and wildcards

Use the Like operator in the Criteria cell of a field to perform wildcard searches against the field’s contents. Access searches for a pattern in the field; you use the question mark (?) to represent a single character or the asterisk (*) for several characters. In addition to ? and *, Access uses three other characters for wildcard searches.

The question mark (?) stands for any single character located in the same position as the question mark in the example expression. An asterisk (*) stands for any number of characters in the same position in which the asterisk is placed. The pound sign (#) stands for a single digit (0–9) found in the position occupied by the pound sign. The brackets ([]) and the list they enclose stand for any single character that matches any one character in the list located within

the brackets. Finally, the exclamation point (!) inside the brackets represents the Not operator for the list—that is, any single character that does not match any character in the list.
These wildcards can be used alone or in conjunction with each other. They can even be used multiple times within the same expression.

Task:

1. Create query from table tblCustomers.
2. Find all records of contacts whose last name begins with Ca.
3. Run query and save it as LastName.

Understanding an Or operation

You use an Or operator in queries when you want a field to meet either of two conditions. For example, you might want to see all the records where the customer has an address in either New York or California. In other words, you want to see all records where a customer has addresses in NY, in CA, or both. The general expression for this operation is:

[State] = "NY" Or [State] = "CA"

The Or operator is used to specify multiple values for a field. For example, you use the Or operator if you want to see all records of buyers who live in AZ or CA or NY. To do this, follow these steps:

1. Open a new query in Design view and add tblCustomers and tblSales.
2. Add Company and State from tblCustomers and SalesDate from tblSales.
3. Click in the Criteria cell of State.
4. Type AZ Or CA Or NY in the cell. Access automatically places quotation marks around your example data—AZ, CA, and NY.
5. Run query and save it as Three_states.

Using And to specify a range

The And operator is frequently used in fields that have numeric or date/time data types. It's seldom used with text data types, although it can be this way in some situations. For example, you might be interested in viewing all buyers whose names start with the letters d, e, or f. The And operator can be used here (>="D" And <="G").
You use the And operator in queries when you want a field to meet two or more conditions that you specify. For example, you might want to see records of buyers who have purchased products between October 1, 2012, and March 31, 2013. In other words, the sale had to have occurred during the last quarter of the year 2012 and the first quarter of 2013. The general expression for this example is:

(SaleDate >= #10/1/2012#) And (SaleDate <= #3/31/2013#)

Task:

1. Create a new query using tblCustomers and tblSales.
2. Add Company from tblCustomers and SaleDate from tblSales.
3. Click in the Criteria cell of SaleDate.
4. Type >= #10/1/2012# And <= #3/31/2013# in the cell.

5. Run and save query as Date.

Formulas

Task:

1. Create query from table tblProducts.
2. Increase the RetailPrice with number 3.
3. Run and save query as Increased_Price.

Functions

Task:

1. Create query from table tblContacts.
2. Extract first two letters from the field LastName.
3. Run and save query as LastName_Extract.