

ACCESS 2013

Creating Complex Queries



LESSON OUTLINE

Identifying Advanced Query Features

Creating Select Queries

Creating a Calculated Field in a Query

Creating and Running Parameter Queries

Creating and Running Action Queries

Concepts Review

Reinforce Your Skills

Apply Your Skills

Extend Your Skills

Transfer Your Skills

LEARNING OBJECTIVES

After studying this lesson, you will be able to:

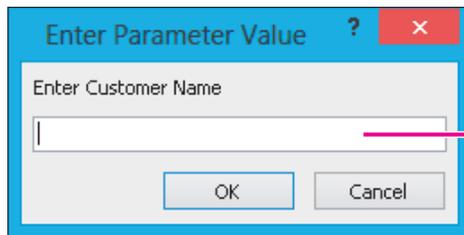
- Create a select query involving multiple tables
- Create a report based on multiple tables in a query
- Create and run parameter queries
- Create a calculated field in a query
- Create and run action queries

As the volume of data stored in database tables grows, so does the number of edits required to keep data for thousands of records up to date. Although relational databases reduce redundant data and increase efficiency, the accuracy and validity of a database is maintained through regular updates. In this lesson, you will explore queries designed to enhance the timeliness and accuracy of large relational databases. You will select and display desired fields in Datasheet View, calculate and summarize data, and use parameter queries that prompt you to enter values to generate or modify records. You will also use action queries to automate database tasks and specify criteria to display a subset of data to make updating, selecting, and deleting data more efficient.

CASE STUDY

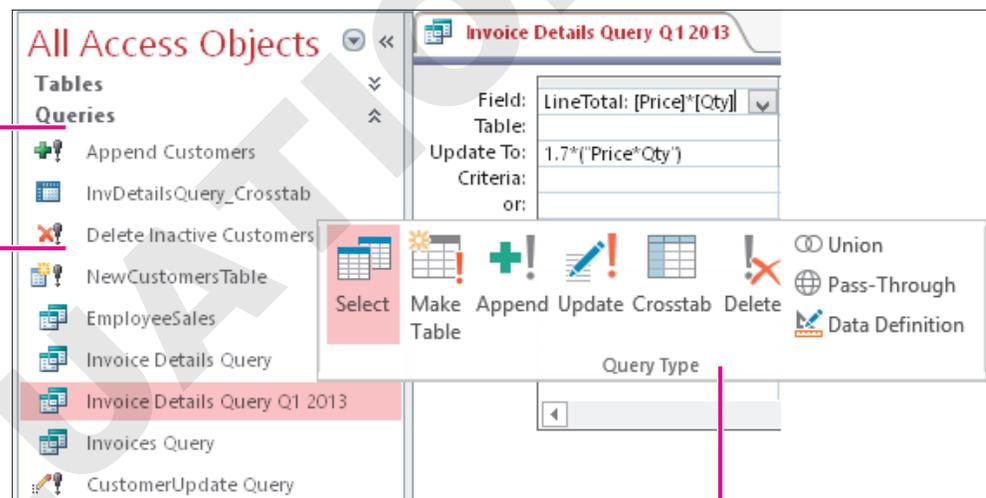
Handling Growing Databases

You are responsible for analyzing the data retrieval processes for the growing Winchester Web Design database. You decide to develop queries to increase the efficiency of data entry and updates. You will create a parameter query to display information for only one customer, an append query to add records to the Products list, and a delete query to remove older invoices. You will also create additional queries to perform cleanup and automate database tasks.



A parameter query prompts the user for a value in one or more table fields.

Query types are identified by icons in the Navigation Pane.



The Query Type section of the Design tab provides easy access to query tools.

Identifying Advanced Query Features

Video Library <http://labyrinthlab.com/videos> Video Number: AC13-V0801

You may be familiar with simple queries such as the select query, crosstab query, unmatched query, and duplicates query. These queries are designed to quickly locate and display records that meet specific conditions and criteria.

Access also offers a variety of query types that are designed to address more advanced needs. Each query type, from the basic to the more advanced, is identified in the following table.

ACCESS QUERY TYPES	
Query Type	Description
Select Query	Allows Access to retrieve desired fields from one or more tables that meet specific criteria conditions, and then sort, summarize, and calculate that data as needed.
Parameter Query	Prompts user to enter data that Access uses to filter records and return only a subset of records that match the value entered.
Crosstab Query	Displays row headings on the left side of the datasheet (i.e., <i>Customer ID</i>) and column headings across the top (i.e., <i>Products</i>), to sum, count, or average each column corresponding to the row field (i.e., total products purchased by customers).
Action Query	Performs one of four actions on a group of records: deletes records, updates records, appends records, or creates a new table.
SQL Query	Uses structured query language (SQL) to create a query. Very few developers program in the more difficult-to-use SQL, because any query designed with either the Wizard or in Design View is automatically converted to SQL when you choose SQL View.
Unmatched Query	Locates records in one table that have no match in another table. An unmatched query could ensure that each record in an <i>Invoices</i> table has a corresponding record in the <i>Customers</i> table.
Duplicates Query	Locates records containing duplicate field values in a single table or query. For example, a duplicates query could locate records in the <i>Customers</i> table that were entered more than once.

Querying Tables Containing No Relationships

When tables in a database have no established relationships, running queries on the tables can produce undesired results.

Cartesian Product Lists

A **Cartesian product list** displays all possible combinations from a database query which uses fields from two unrelated tables. Without relationships, Access has no idea how to relate the data contained in each table, so it presents every possible combination between the two tables in the query results datasheet, listing the same record many times. When the number of records

in each table is large, these queries take a long time to run and provide quite meaningless results.

Last Name	First Name	EmpID
Abrams	John	JFW
Abrams	John	JKK
Abrams	John	JMM
Abrams	John	MJW
Anders	Mark	JFW
Anders	Mark	JKK
Anders	Mark	JMM
Anders	Mark	MJW
Blaser	Helen	JFW
Blaser	Helen	JKK
Blaser	Helen	JMM
Blaser	Helen	MJW

Field:	CustLastName	CustFirstName	EmpID
Table:	OldCustomers	OldCustomers	Employees
Sort:			
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:			

Cartesian product query results where every customer is listed with all four of the employees for a small company, greatly compounding the number of records displayed.

Avoiding Cartesian Products

To prevent meaningless and sometimes huge Cartesian product lists, you must establish proper relationships before creating a query. When you create a query from multiple tables make sure that a join line connects fields in different tables and that referential integrity has been established in the Relationship window.

Creating Select Queries

Video Library <http://labyrinthlab.com/videos> Video Number: AC13-V0802

Select queries, which can display selected fields from one or more database tables in a single datasheet, are the most common type of query. You can design a select query to retrieve records based on criteria, and then sort those records in the query results datasheet. You can group and summarize data using aggregate functions such as sum, count, min, max, and avg. In addition, you can create calculated fields using values contained in other fields.

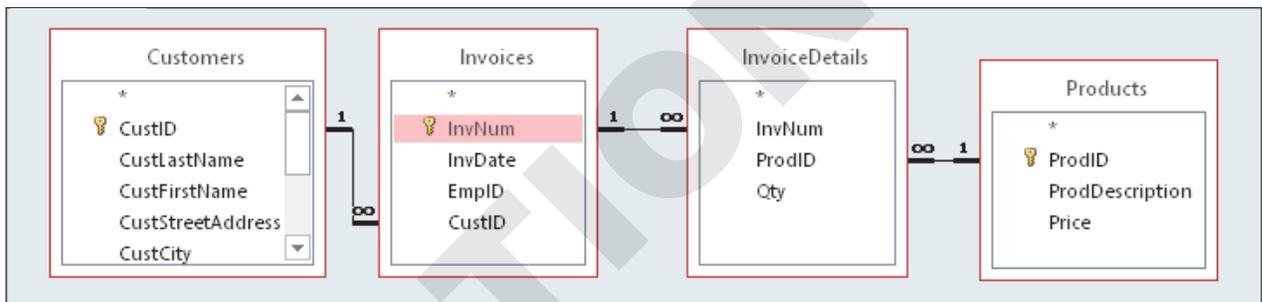
One major advantage to running queries is that they are dynamic, meaning they display up-to-date data each time they are run. Select queries do not store data; they simply display the data. You can then edit the data or use the data to create reports. However, the data remains stored in the database tables—not in the query results datasheet.

DEVELOP YOUR SKILLS AC08-D01

Create a Select Query

In this exercise, you will create a select query that displays data from different tables in the Winchester Web Design database.

1. Open **AC08-D01-WinWebDesign** from the **AC2013 Lesson 08** folder and save it as **AC08-D01-WinWebDesign- [FirstInitialLastName]**.
 Replace the bracketed text with your first initial and last name. For example, if your name is Bethany Smith, your filename would look like this: *AC08-D01-WinWebDesign-BSmith*.
2. **Expand** the Navigation Pane, if it is collapsed, and review the table objects contained in the database.
3. Choose **Create** → **Queries** → **Query Design** to create a new query.
4. Double-click the following table names in the **Show Table** dialog box: **Customers**, **Invoices**, **InvoiceDetails**, and **Products**.
5. Close the **Show Table** dialog box. Arrange the table title bars as shown.



6. Double-click the **InvNum** and **InvDate** fields in the **Invoices** table to add them to the query grid.
7. Double-click each of the following fields to add them to the query grid:

Table	Fields
Customers	CustLastName
Products	ProdDescription, Price
Invoice Details	Qty

8. Click in the **Sort** row of **InvNum** and choose **Ascending**.
9. **Save** the query as **Customer InvoiceDetails Query**.

10. Run  the query, returning 124 records.

InvNum	Invoice Date	Last Name	ID	Description	Price	Qty
1	3 /15/2012	Smith	SmithW	Image, Custom Designed	\$40.00	11
1	3 /15/2012	Smith	SmithW	Secondary Page	\$200.00	6
1	3 /15/2012	Smith	SmithW	Home Page, Nav, CSS, Design	\$400.00	1
2	4 /2 /2012	Santos	SantosE	Home Page, Nav, CSS, Design	\$400.00	1
2	4 /2 /2012	Santos	SantosE	Hourly Rate for Modifications	\$80.00	5
2	4 /2 /2012	Santos	SantosE	Image, Custom Designed	\$40.00	14
2	4 /2 /2012	Santos	SantosE	Secondary Page	\$200.00	7
3	5 /11/2012	Santos	SantosE	Secondary Page	\$200.00	2
3	5 /11/2012	Santos	SantosE	Image, Custom Designed	\$40.00	6
4	5 /30/2012	Smith	SmithW	Blog, Integrated into Site	\$300.00	1

Unless directed otherwise, keep Access and any database or database objects being used open at the end of each exercise.

Creating a Calculated Field in a Query

Video Library <http://labyrinthlab.com/videos> Video Number: AC13-V0803

Queries display fields contained in tables. You can also create a calculated field, which assigns a name to a new field and uses existing fields to perform mathematical or logical operations. Calculated fields can also be used to concatenate, or combine, fields, such as first and last names. For example:

Last Name	First Name	ID: [CustLastName]+Left([CustFirstName],1)
Smith	William	SmithW

Calculated fields generate up-to-date results each time you run the query. This can be quite valuable when the underlying data changes.

Identifying Features of a Calculated Field

A calculated field includes field name(s), operators, and punctuation marks that tell Access how to perform the calculation.

Calculated Field Terms

Calculated fields follow the same order of precedence used in all mathematical equations. Calculations contained in parentheses are performed first, and then multiplication and division, followed by addition and subtraction, in a left to right order. In Access, each calculated field is named and constructed as described in the following table.

BASIC CALCULATED FIELD TERMINOLOGY	
Term	Description
Calculated Field Name	Unique name assigned to a calculated field, which is followed by a colon to separate it from the rest of the expression.
Arithmetic Operators	Add (+), subtract (-), divide (/), multiply (*), and exponential (^).
Logical Operators	Equals (=), greater than (>), less than (<), greater than or equal to (>=), and so on; used to compare values.
Expression	The combination of field names and arithmetic and logical operators required to perform the calculation; basically an Access formula.
Field Names	Fields from database tables used in calculations, which Access encloses within square brackets ([]).

The following table provides a guide on how to create a calculated field.

QUICK REFERENCE		CREATING A CALCULATED FIELD
Task	Procedure	
Create a calculated field in a query	<ul style="list-style-type: none"> ■ Display query in Design View; add table field lists that contain the values you want to use in the expression. ■ Click Field row for first available column in the query grid. ■ Type calculated field expression in the cell using the following structure: New Calculated Field Name: [Field]Operand[Field]. 	

DEVELOP YOUR SKILLS AC08-D02

Add Calculations to a Select Query

In this exercise, you will add a calculated field and a concatenated field to the Customer InvoiceDetails Query.

1. Display the **Customer InvoiceDetails Query** in **Design View**.
2. Click in the **Field row** of the first available column after the **Qty** column.
3. Type **LineTotal: Price * Qty** in the Field row and tap **Enter**.
Double-click the right border of the new field's column heading, if necessary, to see the entire calculation.
4. Click the new **LineTotal** field and open the **Property Sheet**.
5. Choose **Currency** for the **Format** property.
6. Click in the first available **Field row** after **LineTotal**.
7. Type **ID: [CustLastName] +Left ([CustFirstName] , 1)** in the Field row and tap **Enter**.
This produces a resulting ID field that consists of the customer's last name followed by the first character of the customer's first name.

- Select the new concatenated **ID** field and drag it to the left of ProdDescription in the query grid.

ID: [CustLastName]+Left([CustFirstName], 1)	ProdDescription	Price	Qty	LineTotal: [Price]*[Qty]
	Products	Products	InvoiceDetails	
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

The new ID now follows the CustLastName field.

- Choose **Design**→**Results**→**Run** .

InvNum	Invoice Date	Last Name	ID	Description	Price	Qty	LineTotal
	3 /15/2012	Smith	SmithW	Image, Custom Designed	\$40.00	11	\$440.00
1	3 /15/2012	Smith	SmithW	Secondary Page	\$200.00	6	\$1,200.00
1	3 /15/2012	Smith	SmithW	Home Page, Nav, CSS, Design	\$400.00	1	\$400.00
2	4 /2 /2012	Santos	SantosE	Home Page, Nav, CSS, Design	\$400.00	1	\$400.00
2	4 /2 /2012	Santos	SantosE	Hourly Rate for Modifications	\$80.00	5	\$400.00
2	4 /2 /2012	Santos	SantosE	Image, Custom Designed	\$40.00	14	\$560.00
2	4 /2 /2012	Santos	SantosE	Secondary Page	\$200.00	7	\$1,400.00
3	5 /11/2012	Santos	SantosE	Secondary Page	\$200.00	2	\$400.00
3	5 /11/2012	Santos	SantosE	Image, Custom Designed	\$40.00	6	\$240.00
4	5 /30/2012	Smith	SmithW	Blog, Integrated into Site	\$300.00	1	\$300.00

The new concatenated ID field and the calculated LineTotal field.

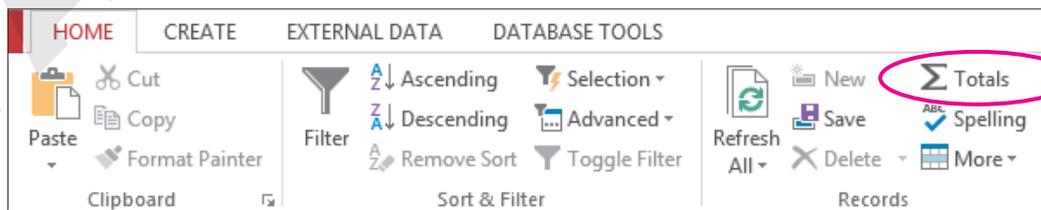
- Double-click the right border of each column heading to size the columns to display data.
- Save  the Customer InvoiceDetails Query.

Adding Totals to Datasheets

Video Library <http://labyrinthlab.com/videos> Video Number: AC13-V0804

Now that you have calculated the value of items in the Customer InvoiceDetails Query, it is possible to obtain a total count and value of all transactions. The Totals button appears on the Ribbon in the Records group of the Home tab.

FROM THE RIBBON
Home→Records→
Totals to add a Total row
in Datasheet View



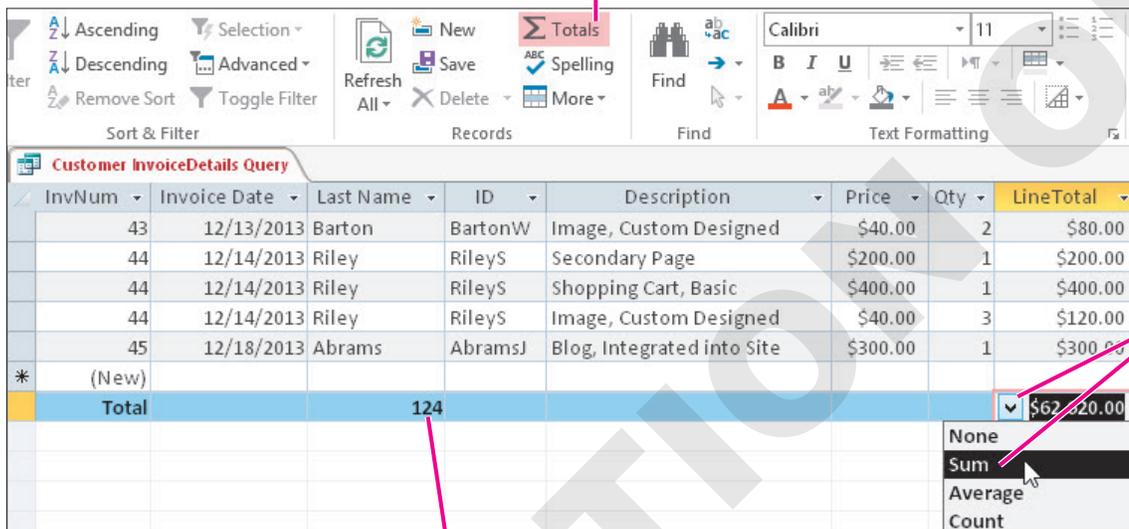
DEVELOP YOUR SKILLS AC08-D03

Add Total Row to the Query Datasheet

In this exercise, you will add a Total row to the Customer InvoiceDetails Query results datasheet.

1. Display the **Customer InvoiceDetails Query** in **Design View**.
2. **Run**  the Customer InvoiceDetails Query.
3. Follow these steps to add the Total row to the query results datasheet:

A Choose **Home**→**Records**→**Totals**.



B Click the menu button in the Total row for the Last Name field and select **Count**.

C Click the menu button in the Total row for the LineTotal field and select **Sum**.

The Total row is added below the new row at the bottom of the datasheet. Access calculates the total value of all the LineTotal items and displays the results.



To remove totals from displaying each time you run the query, choose Home→Records→Totals again.

4. Save and close the **Customer InvoiceDetails Query**.

Creating a Multi-Table Report Based on a Query

Video Library <http://labyrinthlab.com/videos> Video Number: AC13-V0805

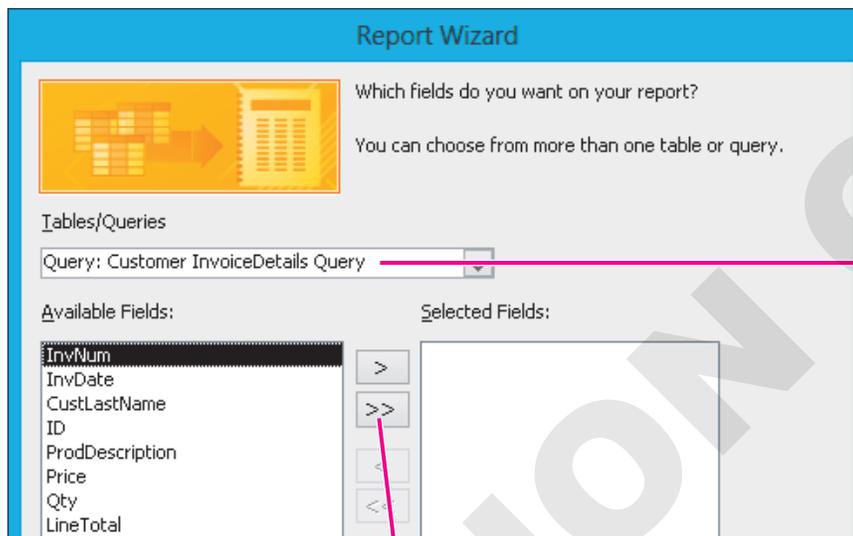
After you create a query that contains fields from multiple tables, you can use that query as the basis for creating a multi-table report. If the tables are joined in a relationship, creating a multi-table report is as easy as building a simple report.

DEVELOP YOUR SKILLS AC08-D04

Create a Report Using a Query

In this exercise, you will create a new multi-table report using the Customer InvoiceDetails Query as the source. Then you will format the report.

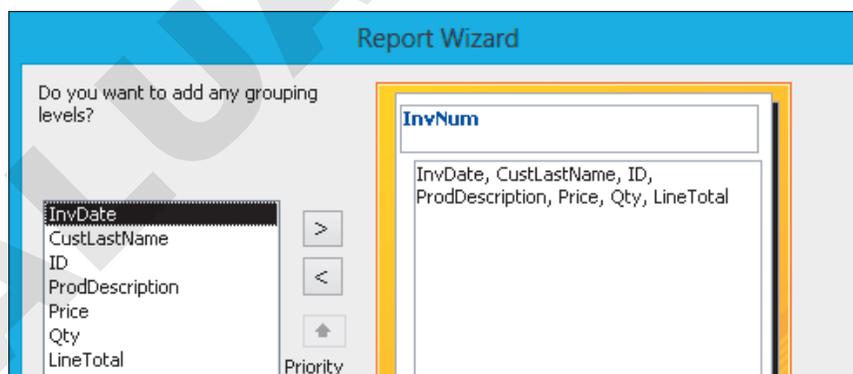
1. Choose **Create**→**Reports**→**Report Wizard**.
2. Follow these steps to create a report from a query using the Report Wizard:



A Select **Query: Customer InvoiceDetails Query** from the Tables/Queries list.

B Click the **Move All** button to move all listed fields to the Selected Fields list.

3. Click **Next**.

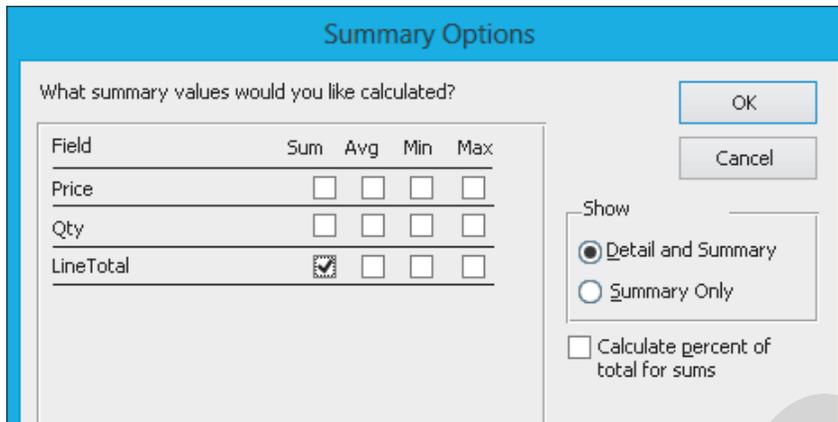


4. With the **InvNum** field selected, click **Move** .

If Access sets the ProdID, or any other field as the grouping level, click the Move button to move the unsolicited field back to the left pane. Then select the InvNum field and move it over as the grouping level.

5. Click **Next**. On the next Wizard screen, click **Summary Options**.

6. In the Summary Options screen, click the checkbox to **Sum** the LineTotal.



7. Click **OK** to close the Summary Options window. Click **Next**.

Access asks you to select a layout for the report.

8. Choose the **Outline** layout and the **Landscape** orientation options; click **Next**.

9. Type **Customer Invoice Report** for the report title and click **Finish**.

Customer Invoice Report

InvNum	Invoice Date	Last Name	ID	Description	Price	Qty	LineTotal
1	3 /15/2012	Smith	SmithW	Home Page, Nav, CSS, Desi	\$400.00	1	\$400.00
	3 /15/2012	Smith	SmithW	Secondary Page	\$200.00	6	\$1,200.00
	3 /15/2012	Smith	SmithW	Image, Custom Designed	\$40.00	11	\$440.00
Summary for 'InvNum' = 1 (3 detail records)							
Sum							2,040.00
2	4 /2 /2012	Santos	SantosE	Secondary Page	\$200.00	7	\$1,400.00
	4 /2 /2012	Santos	SantosE	Image, Custom Designed	\$40.00	14	\$560.00
	4 /2 /2012	Santos	SantosE	Hourly Rate for Modificatio	\$80.00	5	\$400.00
	4 /2 /2012	Santos	SantosE	Home Page, Nav, CSS, Desi	\$400.00	1	\$400.00
Summary for 'InvNum' = 2 (4 detail records)							
Sum							2,760.00

The report displays in Print Preview, showing invoice totals and summary totals for each transaction and a grouping level on the Customer Number field. Each order is identified, and the items ordered for each order are shown.

Access 2013

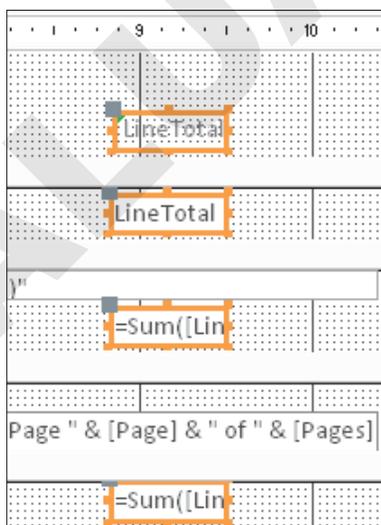
Format Report Controls

10. Click  and switch to **Design View**. Open the **Property Sheet**, if necessary.
11. Select the **InvNum** label. Type **Invoice Number** for the **Caption** property; type **1.1"** for the **Width** property, and type **.25"** for the **Left** property.
12. Set the following Property Sheet values to position and size the listed label and text box controls (if there is no value for a control listed in the table below, do not change that property):

Control	Width	Left
InvNum text box	.5	1.5
Invoice Date label	1	.25
InvDate text box	1	.25
CustLastName text box	1	
ID label		2.5
ID text box	1	2.5
Description label		3.6
ProdDescription text box	2.25	3.6
Price label		6.5
Price text box	1	6
Qty label		7.2
Qty text box		7
="Page " control (Page Footer)	1.5	4

Format Multiple Report Controls

13. Click the **Line Total label**, press **[Ctrl]**, and click each **LineTotal** text box to select all four controls.



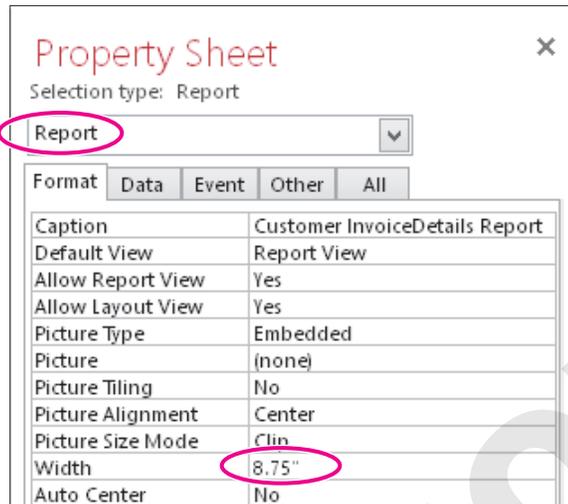
14. Type **1"** for the **Width** property and **7.75"** for the **Left** property.

Delete Report Controls

15. Select the =**Summary for** control across the **InvNum Footer** section and tap **Delete**.
16. Select the =**Now()** control on the left side of the **Page Footer** and tap **Delete**.

Change Report Properties and Preview the Report

17. If necessary, choose **Report** in the **Selection Type** list box at the top of the Property Sheet and type **8.75** for the **Width** property of the report.



If the report width will not change to 8.75, verify that no controls extend beyond that point. You may have to resize and reposition the controls to solve the problem.

18. Switch to **Print Preview** to see your changes.

Customer Invoice Report							
Invoice Number 1							
Invoice Date	Last Name	ID	Description	Price	Qty	LineTotal	
3 /15/2012	Smith	SmithW	Home Page, Nav, CSS, Design	\$400.00	1	\$400.00	
3 /15/2012	Smith	SmithW	Secondary Page	\$200.00	6	\$1,200.00	
3 /15/2012	Smith	SmithW	Image, Custom Designed	\$40.00	11	\$440.00	
Sum						\$2,040.00	
Invoice Number 2							
Invoice Date	Last Name	ID	Description	Price	Qty	LineTotal	
4 /2 /2012	Santos	SantosE	Secondary Page	\$200.00	7	\$1,400.00	
4 /2 /2012	Santos	SantosE	Image, Custom Designed	\$40.00	14	\$560.00	
4 /2 /2012	Santos	SantosE	Hourly Rate for Modifications	\$80.00	5	\$400.00	
4 /2 /2012	Santos	SantosE	Home Page, Nav, CSS, Design	\$400.00	1	\$400.00	
Sum						\$2,760.00	

19. Save the **Customer Invoice Report**.

Creating and Running Parameter Queries

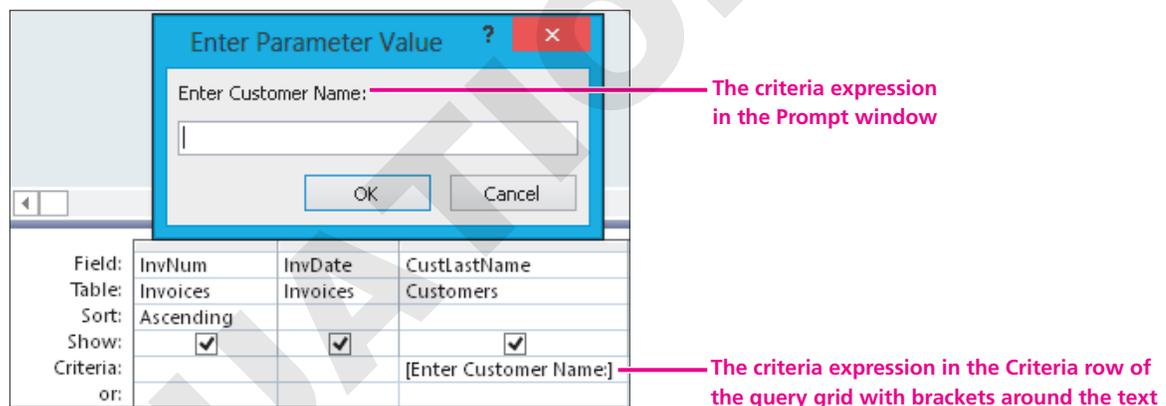
Video Library <http://labyrinthlab.com/videos> Video Number: AC13-V0806

In programming, a parameter is a value or type of variable that is used to pass input to a routine, or in Access, to a query. A **parameter query** is a select query that prompts the user to enter new criteria values each time they run the query; the query then generates results based on the value(s) entered. Parameter queries enable you to enter criteria to limit results without accessing Design View each time you want to run the query. This is especially useful when you create a query for users who are unfamiliar with query design.

Setting Up a Parameter Query

Parameter queries use the Criteria row of the query grid to create a criteria expression, or the text that you want to appear in the prompt. A prompt is the text that asks—or prompts—the user to enter a value. When you run a parameter query, you enter a value in the prompt box, and the query results datasheet displays only records containing the value entered.

In most cases, the parameter field appears in the query results datasheet. However, you can also use the field to limit query results without displaying the field in the results datasheet.



Sample query grid with criteria expression and prompt box

Formatting the Criteria Expression

Parameter criteria expressions must be enclosed in square brackets ([]). If you omit the brackets, Access places quotation marks around the text and searches for a value that matches the text rather than using the text as a user prompt.

Without the square brackets [] Access searches for a value of *Enter Customer Name*, which will not return any records.

Including the brackets directs Access to display the prompt *Enter Price:* when the query is run.

CustLastName	ID: [CustLastName]	ProdDescription	Price
Customers		Products	Products
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
"Enter Customer Name"			[Enter Price:]

Creating Complex Parameter Prompts

Suppose you want the user to enter a product ID each time the query is run. Because the products may be handled by multiple employees, you may also want the view a specific product’s sales by a particular employee. To create multiple prompts for this scenario, you would type [Enter ProdID] in the ProdID criteria row and [Enter EmpID] in the EmpID criteria row. You can also set prompts for multiple values in the same query column or include logical criteria such as greater than (>) and less than (<).

The following table shows common examples of parameter query criteria and the results they would create.

EXAMPLES OF PARAMETER QUERY CRITERIA FOR A SINGLE FIELD	
Parameter Criteria	Result
Between [What is the start date?] And [What is the end date?]	Prompts the user to enter the starting date. Then the user enters the end date, directing Access to display records that fall between the two dates entered.
<[What is the highest price you will pay?]	Displays the prompt shown within brackets. After the user enters a value, Access displays all records for values less than the one entered.

QUICK REFERENCE	CREATING A PARAMETER QUERY
<p>Task</p> <p>Create a parameter query</p>	<p>Procedure</p> <ul style="list-style-type: none"> Choose Create→Queries→Query Design Add required tables to the query, and then add the desired fields from the table field list(s) to the query grid. Click the criteria row for the column for which users will enter a value and type the prompt text within square brackets in a format similar to [Enter Date].

Access 2013

DEVELOP YOUR SKILLS AC08-D05

Create and Run a Parameter Query

In this exercise, you will create and run a parameter query that prompts the user for a customer's last name.

1. Click the **Customer InvoiceDetails Query**.
2. Press **[Ctrl]+[C]** to copy the query, then press **[Ctrl]+[V]** and name the copy **Customer Invoice Parameter Query**.
3. Display **Customer Invoice Parameter Query** in **Design View**.
4. Click the criteria row for the **CustLastName** column and type this text: **[Enter Customer Name:]**

Field:	InvNum	InvDate	CustLastName
Table:	Invoices	Invoices	Customers
Sort:	Ascending		
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:			[Enter Customer Name:]

Be sure to include the square brackets.

Run and Test the Query

5. Choose **Design**→**Results**→**Run** . Type **Abrams** in the parameter box and press **[Enter]**.

InvNum	Invoice Date	Last Name	ID	Description	Price	Qty	LineTotal
42	12/6/2013	Abrams	AbramsJ	Secondary Page	\$200.00	5	\$1,000.00
42	12/6/2013	Abrams	AbramsJ	Home Page, Nav, CSS, Design	\$400.00	1	\$400.00
45	12/18/2013	Abrams	AbramsJ	Blog, Integrated into Site	\$300.00	1	\$300.00
*(New)							
Total			3				\$1,700.00

Access returns records that contain the value **Abrams** in the customer last name field.

6. Close the query, saving changes to the datasheet if prompted.

Creating and Running Action Queries

Video Library <http://labyrinthlab.com/videos> Video Number: AC13-V0807

An **action query** performs an action that modifies a database table or a group of records in a table. Action queries can modify, move, update, or delete groups of records with a single action. You can even use an action query to create a new table by adding various fields from other tables.

Each time you open an action query, *Access runs and generates the results*. So, if you create an update query designed to increase prices by 10 percent on all items in a table, Access will increase those prices every time you run the query. Because action queries do not open the table that they are updating, a user may run the query a second time, doubling the effect of the query.



Every time you open an action query, Access runs and generates results. It is good practice to delete action queries after running them to help maintain the validity of data in the database because these changes cannot easily be undone.

Identifying Action Query Types

There are four basic types of action queries available in Access. They are described in the following table.

ACTION QUERY TYPES	
Action Query Type	Description
Make Table Query	A query that creates a new table from the selected data in one or more tables. For example, you may want to create a backup table to store 2012 records before you delete them from your main file.
Append Query	A query that adds a group of records from one or more sources to the end of one or more tables. For example, you could add a customer to a database the first time he or she places an order.
Update Query	A query that makes global changes to a group of records in one or more tables. For example, you can increase the prices of all products in a specific category or update phone numbers that change when the phone company adds a new area code.
Delete Query	A query that deletes a group of records from one or more tables. For example, you could create a delete query to remove records for a discontinued line of products.

Identifying Queries by Their Icons

Access identifies each type of query with a different icon. The following table displays the various icons and their query type.

QUERY TYPES AND THEIR ICONS			
Icon	Query Type	Icon	Query Type
	Select query		Update query
	Make Table query		Delete query
	Append query		Crosstab query

QUICK REFERENCE	CREATING AND RUNNING ACTION QUERIES
Task	Procedure
Create an append query	<ul style="list-style-type: none"> ■ Create a new query in the source database and add all table fields to the query grid. ■ Choose Design→Query Type→Append; enter the name of the destination table. ■ Save and run the query.
Create an update query	<ul style="list-style-type: none"> ■ Create a new query in the source database and add all table fields to the query grid. ■ Set criteria. ■ Choose Design→Query Type→Update. ■ Save and run the query.
Create a make table query	<ul style="list-style-type: none"> ■ Create a new query in the source database and add all table fields to the query grid. ■ Set criteria if required. ■ Choose Design→Query Type→Make Table. ■ Save and run the query.
Create a delete query	<ul style="list-style-type: none"> ■ Create a new query in the source database and add all table fields to the query grid. ■ Set criteria. ■ Choose Design→Query Type→Delete. ■ Save and run the query.

Enabling Content

Action queries require that content within a database be enabled. As a result, if you did not click the Enable Content button found at the top of the Access window when you first opened the database, Access will display an error message advising you to enable content before you can create or run action queries.

Creating a Make Table Query

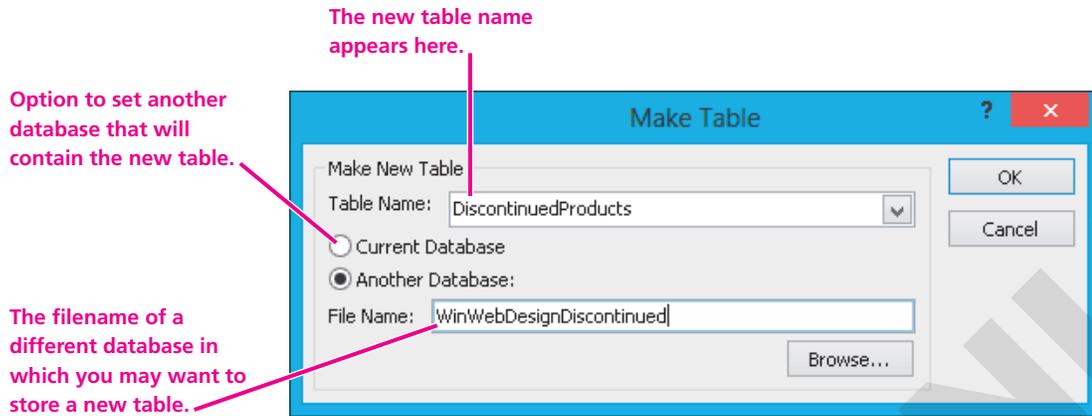
A **make table query** is an action query that can create a new table based on data from multiple tables in a database. It's also a great way to place data produced from a calculated query field into a table.



If you rerun a make table query, Access will replace the existing table with a new one with new records that meet the query criteria. To retain your existing query-created table, rename and save it.

Moving Data to Tables and Databases

When you create a new table using a make table query, Access prompts you for a table name and even allows you to save the data in another database. An example of moving records to another database would be for archival purposes when they become obsolete, such as when a product is no longer available.



DEVELOP YOUR SKILLS AC08-D06

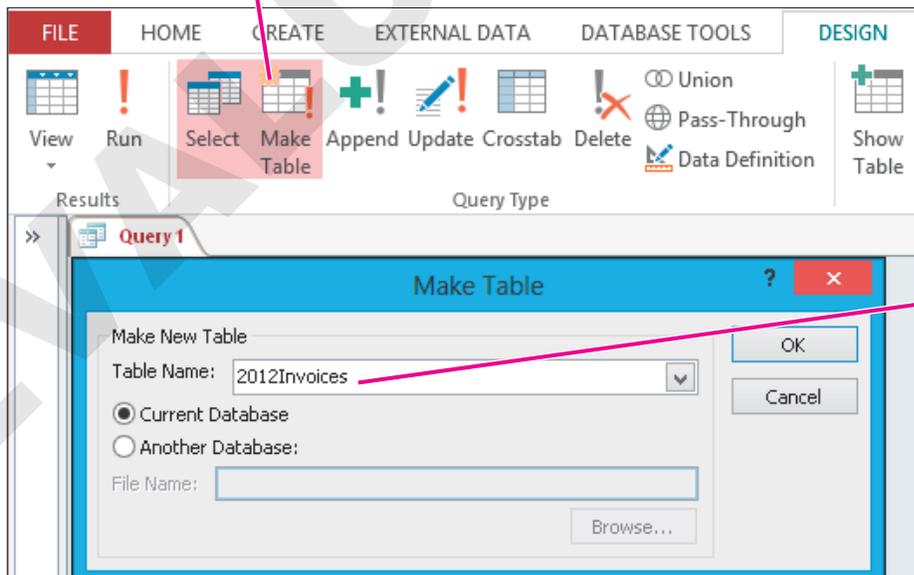
Create a Make Table Query

In this exercise, you will create a make table action query to save 2012 invoice records in a new table.

1. Run the **Invoices Query**.
2. Switch to **Design View**.
3. Type **Between 1/1/2012 and 12/31/2012** in the Criteria row for the **InvDate** field.

Field:	InvNum	InvDate
Table:	Invoices	Invoices
Sort:		
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:		Between #1/1/2012# And #12/31/2012#

4. Follow these steps to create a make table query:
 - A Choose **Design**→**Query Type**→**Make Table**.



5. Click **OK**.
6. Run the query.
Access displays an error message warning you that you are about to paste 61 records into a new table.
7. Choose **Yes** to continue.
Access creates the table and displays it in the list of tables in the Navigation Pane.

Verify the Table

8. Display the **2012Invoices** table in **Datasheet View**.

InvNum	InvDate	ProdDescription	Price	Qty	LineTotal
	3/15/2012	Home Page, Nav, CSS, Design	\$400.00	1	\$400.00
1	3/15/2012	Secondary Page	\$200.00	6	\$1,200.00
1	3/15/2012	Image, Custom Designed	\$40.00	11	\$440.00
2	4/2/2012	Home Page, Nav, CSS, Design	\$400.00	1	\$400.00
2	4/2/2012	Secondary Page	\$200.00	7	\$1,400.00
2	4/2/2012	Image, Custom Designed	\$40.00	14	\$560.00
2	4/2/2012	Hourly Rate for Modifications	\$80.00	5	\$400.00
2	5/11/2012	Secondary Page	\$200.00	2	\$400.00

9. Close the **2012Invoices** table and the **Invoices Query**.
Do not save the query. The 2012Invoices table has been successfully created, and you would not want to risk overwriting it.

Creating an Append Action Query

Video Library <http://labyrinthlab.com/videos> Video Number: AC13-V0808

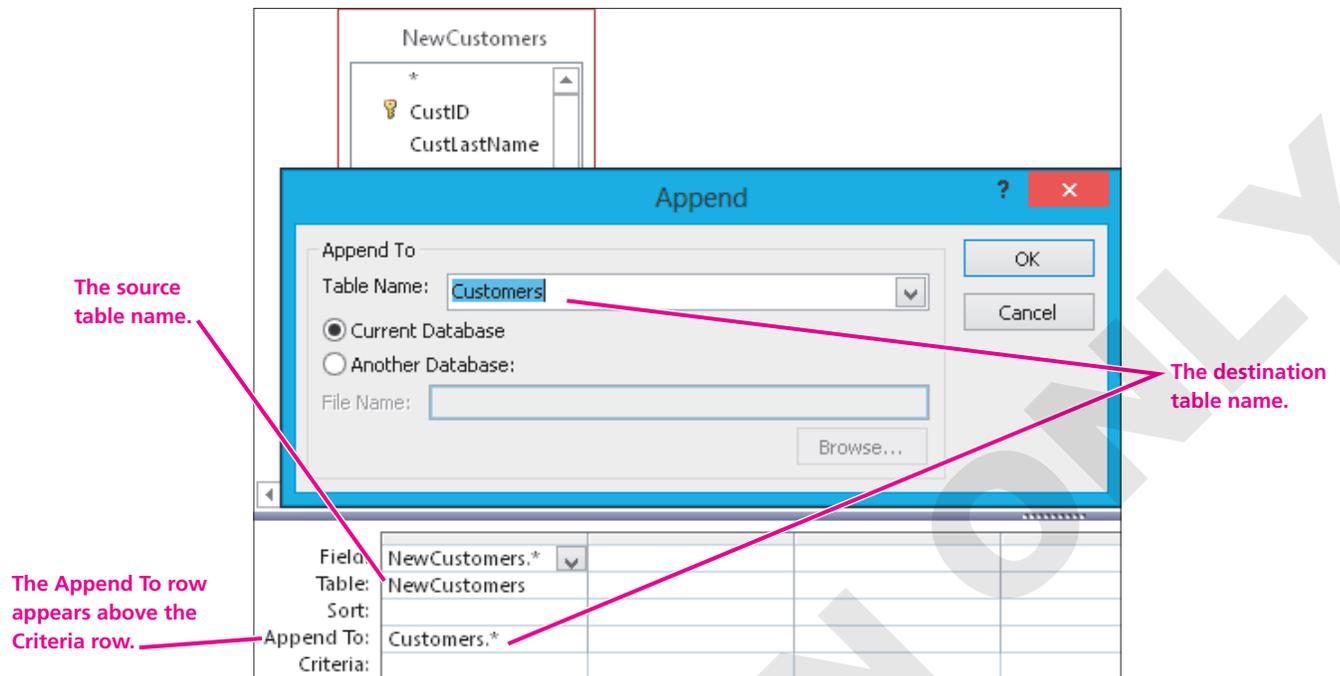
An **append query** is an action query that adds a group of records from one or more tables to the end of one or more tables in the same or in another database. For example, if you want to offer a new set of products, you could use an append action query to add the new items to the existing products table. Or you might use an append action query to automatically add new customers to the customers table the first time a customer places an order.

Formatting the Source and Destination Tables

When you create an append action query, the table containing the records you want to add to another table is called the source table. The table receiving the records is the destination table. To successfully run an append action query, the structures—field names, data types, and field order—for both tables should be the same.

Identifying the Source and Destination Tables

Append queries are created in the database containing the source table. When you run the query, the Append dialog box prompts you to identify the destination database and table. Access identifies the destination table in the Append To row of the query grid.



DEVELOP YOUR SKILLS AC08-D07

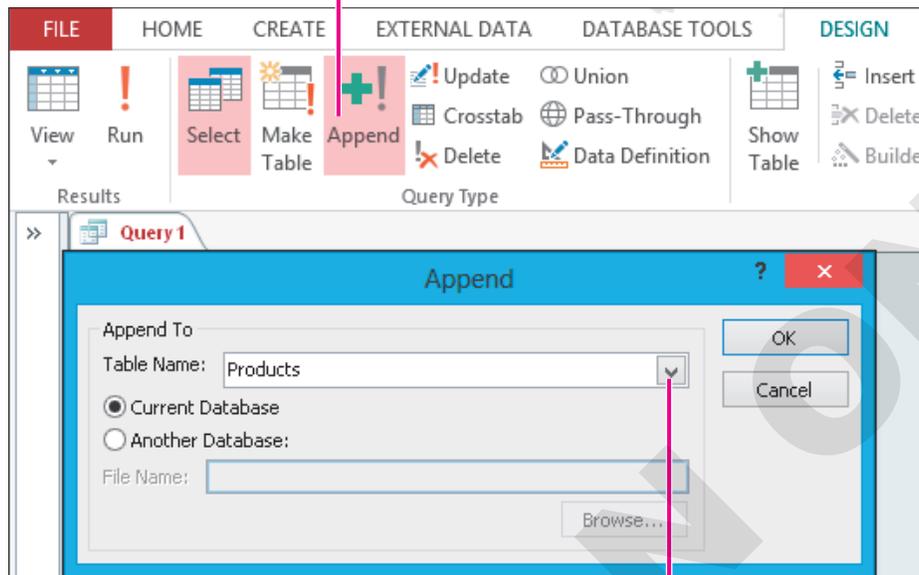
Create an Append Action Query

In this exercise, you will create an action query to append records from the *New Products* table to the existing *Products* table.

1. Open the **Products** table and notice that it contains six records.
2. Open the **NewProducts** table to see the records that will be appended to the **Products** table.
3. Close both tables.
4. Choose **Create**→**Queries**→**Query Design**  to create a new query.
5. Add the **NewProducts** table to the query window. Close the **Show Table** dialog box.
6. Double-click the **asterisk (*)** in the **NewProducts** table to add all the fields to the query grid.

7. Follow these steps to change the query to an append query:

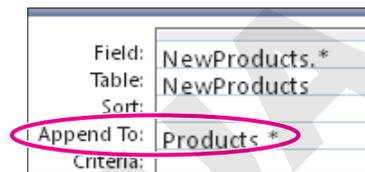
A Choose **Design**→**Query Types**→**Append**.



B Click the **Table Name** menu button and choose **Products**.

8. Click **OK** to close the Append window.

Access adds the Append To row to the query grid with the destination table name.



9. Choose **Design**→**Results**→**Run**.

Access displays a warning that you are about to append five rows to another table.

10. Choose **Yes** to proceed.



Nothing *appears* to happen when you run the query. You can only see the changes after you open the destination table to which the records were appended. *Don't run the query again!* If you do, Access will add the same records to the destination table again.

11. Display the **Products** table in **Datasheet View** to verify that the new records were appended.

ProdID	Description	Price
01HP	Home Page, Nav, CSS, Design	\$400.00
02SP	Secondary Page	\$200.00
03BL	Blog, Integrated into Site	\$300.00
04SC	Shopping Cart, Basic	\$400.00
05IM	Image, Custom Designed	\$40.00
06HR	Hourly Rate for Modifications	\$80.00
07LC	Logo Creation	\$100.00
08PS	Photo Shoot, 1 hour onsite	\$100.00
09IM	Image Map	\$40.00
10SS	Slide Show	\$100.00
11QR	QR Code	\$50.00

The Products table now contains the new products.

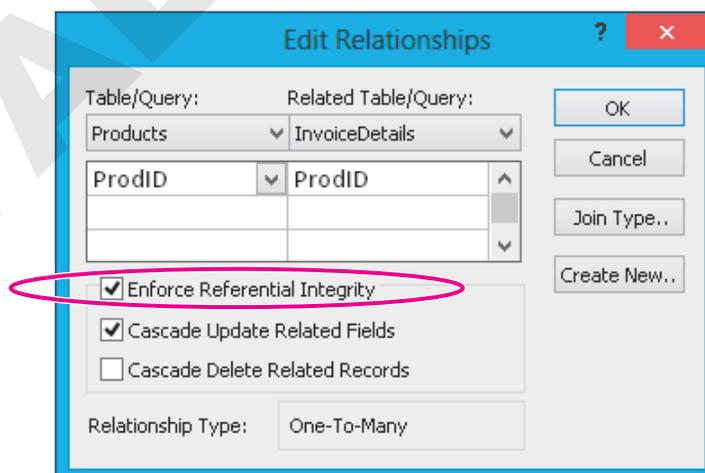
12. Save the query as **Append Products**, and close it.
13. Close the **Products** table, but leave the database open.
Remember: Do not run the query again.

Creating an Update Query

Video Library <http://labyrinthlab.com/videos> Video Number: AC13-V0809

An **update query** is an action query that makes global changes to a group of records in one or more tables. For example, you can use an update query to increase the prices for all products in a specific category or update the area code for phone numbers that change when the phone company adds a new one.

To ensure that corresponding fields in related tables are updated consistently, check the Cascade Update Related Fields checkbox in the Edit Relationships window.



Identifying the Query Grid Update Row

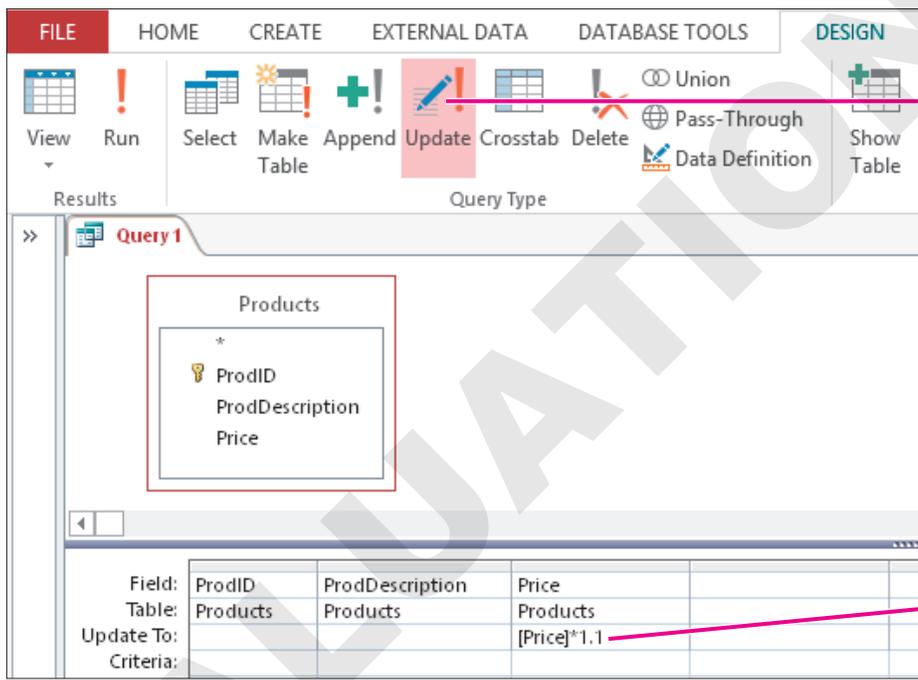
Append, update, crosstab, and delete queries all add a query-specific row to the query grid. The update query places an Update To row in the query grid so that you tell Access how to update the desired field(s). In most cases, this will be changing one value to another by substitution, mathematical operation, formula, or comparison.

DEVELOP YOUR SKILLS AC08-D08

Create an Update Query

In this exercise, you will create an update action query that increases the prices of all items in the Products table by 10 percent.

1. Choose **Create**→**Queries**→**Query Design** , add the **Products** table to the query, and close the **Show Table** dialog box.
2. Double-click each field in the field list to add them to the query grid.
3. Follow these steps to create an update action query to increase prices by 10 percent:



A Choose **Design**→**Query Type**→**Update**.

B Click the **Update To** row of the **Price** column and type **[Price]*1.1**.

Field:	ProdID	ProdDescription	Price
Table:	Products	Products	Products
Update To:			[Price]*1.1
Criteria:			

You must type the brackets or select Price from the drop-down menu that appears when you type a P as you start to type Price.



If you type "Price" without brackets, Access will not recognize it as a field, which means the query will base the calculation on something other than a value and wipe out all existing prices.

4. **Run**  the query to update the values in the Products table.

Access displays a warning message that you are about to update 11 row(s) and will not be able to reverse the action.

5. Choose **Yes** to run the query and then close the query, naming it **UpdatePricing Query**.



Don't run this query again, or you will increase prices by another 10 percent.

Verify the Data Update

6. Display the **Products** table in **Datasheet View**.

ProdID	Description	Price
01HP	Home Page, Nav, CSS, Design	\$440.00
02SP	Secondary Page	\$220.00
03BL	Blog, Integrated into Site	\$330.00
04SC	Shopping Cart, Basic	\$440.00
05IM	Image, Custom Designed	\$44.00
06HR	Hourly Rate for Modifications	\$88.00
07LC	Logo Creation	\$110.00
08PS	Photo Shoot, 1 hour onsite	\$110.00
09IM	Image Map	\$44.00
10SS	Slide Show	\$110.00
11QR	QR Code	\$55.00

The original price for a homepage with navigation and CSS design was \$400. It is now \$440—a 10 percent increase.

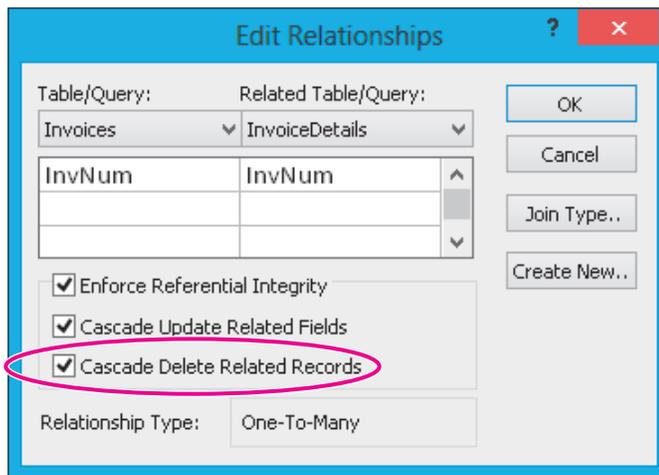
7. Close the **Products** table.

Creating a Delete Query

Video Library <http://labyrinthlab.com/videos> Video Number: AC13-V0810

A **delete query** is an action query that deletes a group of records from one or more tables. For example, you could create a delete query to remove records for a discontinued line of products or to delete records you have appended to another table to prevent inadvertently running an append query multiple times.

To ensure that corresponding records in related tables will all be deleted concurrently, check the Cascade Delete Related Records checkbox in the Edit Relationships window.



When you create a delete action query, Access replaces the Sort row of the query grid with the Delete row. You can set criteria for specific fields in a table to identify the conditions that must be met in order to delete records or set no criteria to remove all records from a table.

DEVELOP YOUR SKILLS AC08-D09

Create a Delete Query

Earlier you used a make table query to create a 2012Invoices table for older invoices. In this exercise, you will create a delete query to remove the 2012 invoices from the Invoices table.

1. Choose **Create**→**Queries**→**Query Design** .
2. Add the **Invoices** table to the query grid. Close the **Show Table** dialog box. Double-click the **InvDate** field to add it to the query grid.
3. Type **Between 1/1/2012 and 12/31/2012** in the **Criteria** row for the **InvDate** field.
4. Choose **Design**→**Query Type**→ **Delete** to create a delete action query.
5. Save the query as **Delete 2012 Invoices**; run the query.
Access warns that you are about to delete 19 records.
6. Choose **Yes** to remove all records meeting the 2012 criteria from the Invoices and InvoiceDetails tables.

Because of cascading deletes in the relationship between the Invoices and InvoiceDetails tables, the 2012 records are deleted from both tables.

- Open the **Invoices** table and the **InvoiceDetails** table to review the results.

Invoices		InvoiceDetails		
InvNum	Invoice Date	InvNum	ProdID	Qty
20	1 / 5 / 2013	20	04SC	1
21	1 / 12 / 2013	20	05IM	14
22	1 / 17 / 2013	20	06HR	5
23	2 / 1 / 2013	21	02SP	12
24	2 / 7 / 2013	21	05IM	18
25	2 / 9 / 2013	21	06HR	6
26	2 / 12 / 2013	22	06HR	4
27	2 / 23 / 2013	23	02SP	2
28	3 / 9 / 2013	23	06HR	3
29	3 / 12 / 2013	24	01HP	1
30	3 / 21 / 2013	24	02SP	13
31	3 / 29 / 2013	24	03BL	1
32	4 / 6 / 2013	24	05IM	19
33	8 / 3 / 2013	24	06HR	6
34	8 / 5 / 2013	25	05IM	8
35	9 / 2 / 2013	25	06HR	3
36	10 / 15 / 2013	26	02SP	6
		26	05IM	14
		26	06HR	5
		27	02SP	6
		27	05IM	14

All the invoices from 2012 have been deleted from the Invoices table and the corresponding records have also been removed from the InvoiceDetails table.

- Close all open objects. Close the database and exit **Access**.

Concepts Review

To check your knowledge of the key concepts introduced in this lesson, complete the Concepts Review quiz by choosing the appropriate access option below.

If you are...	Then access the quiz by...
Using the Labyrinth Video Library	Going to http://labyrinthlab.com/videos
Using eLab	Logging in, choosing Content, and navigating to the Concepts Review quiz for this lesson
Not using the Labyrinth Video Library or eLab	Going to the student resource center for this book

Reinforce Your Skills



REINFORCE YOUR SKILLS AC08-R01

Create Queries with Calculations, a Report from a Query, and Parameter Queries

Kids for Change needs to refine the reports it generates. In this exercise, you will create a select query to calculate the cost of each activity based on staff salary and activity duration. You will also add a Totals row to generate the sum of staffed activities. Then, you will use a parameter query to return donor records by state.

Create a Select Query

1. Start **Access**. Open **AC08-R01-K4C** from the **AC2013 Lesson 08** folder and save it as **AC08-R01-K4C- [FirstInitialLastName]**.
2. Choose **Create**→**Queries**→**Query Design** .
3. Double-click the **PaidStaff** and **Activities** table names in the Show Table dialog box; then close the dialog box.
4. Double-click the **StaffLastName**, **StaffFirstName**, **StaffPhone**, and **HrlySal** fields in the **PaidStaff** table to add them to the query grid.
5. Add the **Activity** and **Hours** fields from the **Activities** table to the query grid.
6. Click in the Sort row of **StaffLastName** and choose **Ascending**.
7. **Save**  the query as **ActivityCostQuery**.
8. Choose **Design**→**Results**→**Run** , returning 18 records.

Create Calculated and Concatenated Query Fields

9. Switch to **Design View**. Click in the **Field** row of the first available column after the **Hours** column.
10. Type **ActivityCost: HrlySal * Hours** in the Field row and tap **Enter**.
11. Double-click the right border of the column heading, if necessary, to see the entire calculation.
12. Right-click the new **ActivityCost** field and choose **Properties**.
13. Choose **Currency** for the **Format** property.
14. Click in the first available **Field** row after **ActivityCost**.
15. Type the following in the Field row then tap **Enter**: **ActStaffID: [StaffLastName] + Left ([StaffFirstName], 1)**
The expression produces a concatenated ActStaffID field that consists of the staffer's last name and first initial.
16. Select the new concatenated **ID** field and drag it to the left of **StaffLastName** in the query grid.
The new ActStaffID should now be the first field in the query grid.

17. Choose **Design**→**Results**→**Run** 

ActStaffID	Last Name	First Name	Telephone	Hrly Sal	Activity	Hrs	ActivityCost
BryantM	Bryant	Matthew	(941) 555-7523	\$20.25	Animal Shelter	2	\$40.50
DalharJ	Dalhar	Jamal	(941) 555-1692	\$23.00	Dog Walking	1	\$23.00
EarleK	Earle	Kevin	(941) 555-1368	\$18.00	Animal Shelter	2	\$36.00

The new concatenated ID field and the calculated ActivityCost field are displayed.

18. Save  the ActivityCostQuery.

Add the Totals Row to a Query

19. Choose **Home**→**Records**→**Totals**.

20. Click the menu button in the Total row for the **Last Name** field and choose **Count**.

21. Click the menu button in the Total row for the **ActivityCost** field and choose **Sum**.

Total	18						\$728.75
							None
							Sum
							Average
							Count

22. Save and close the ActivityCostQuery.

Create a Report Using a Query As Record Source

23. Choose **Create**→**Reports**→**Report Wizard** .

24. Choose **Query: ActivityCostQuery** from the **Tables/Queries** drop-down menu.

25. Add **StaffLastName**, **StaffFirstName**, **HrlySal**, **Activity**, **Hours**, and **ActivityCost** to the Selected Fields list; click **Next**.

26. Choose to group by **Activity**; click **Next**.

27. Click **Summary Options**.

28. Choose the **Sum** option for **ActivityCost**.

Field	Sum	Avg	Min	Max
HrlySal	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Hours	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ActivityCost	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

29. Click **OK** to close the window. Click **Next**.

30. Choose the **Outline** layout and click **Next** to keep the Portrait orientation.

31. Name the report **Activity Cost Report** and click **Finish**.

32. Click . Then close the **Activity Cost Report**.

Modify a Query to Request a Parameter Value to Select Records

33. Right-click the **Donations Query** in the Navigation Pane and choose **Open**.
All the donation records are returned.
34. Switch to **Design View**.
35. Type [**Enter State Abbr:**] in the **Criteria** row of the State field in the query grid, and tap **Enter**.
Be sure to include the square brackets.
36. Choose **Design**→**Results**→**Run** 
A window opens displaying the criteria you entered.
37. Type **MA** (Massachusetts) for the State Abbr. and click **OK**.
38. Save and close the **Donations Query**. Close the database. Exit **Access**.
39. Submit your final file based on the guidelines provided by your instructor.
To see examples of how your final file or files should look at the end of this exercise, go to the student resource center.

REINFORCE YOUR SKILLS AC08-R02

Create Action Queries

Kids for Change needs to fine-tune its database with several action queries. In this exercise, you will create a make table query that produces a table to store older donations, a query that appends new records to the Children table, an update query that reduces the duration of each activity by half, and a query that deletes old donations from the Donations table.

Create a Make Table Query

1. Start **Access**. Open **AC08-R02-K4C** from the **AC2013 Lesson 08** folder and save it as **AC08-R02-K4C-[FirstInitialLastName]**.
2. Right-click **Donations Query** in the Navigation Pane and choose **Copy**.
3. Right-click **Donations Query** again and choose **Paste**.
4. Name the new query **MakeTable2012** and click **OK**.
5. Right-click the **MakeTable2012** query in the Navigation Pane and choose **Open**.
Every 2012 donation record copied from the existing query is displayed in the new query.
6. Switch to **Design View**.
7. Scroll to the right to the **DonationDate** field and type **Between 1/1/2012 and 12/31/2012** in the **Criteria** row.
8. Choose **Design**→**Query Type**→**Make Table**.
9. Type **2012Donations** as the **Table Name** and leave **Current Database** selected.
10. Click **OK**.
11. Save and run the query.
Access warns that you are about to paste four records into a new table.

12. Choose **Yes** to continue.
Access adds the table to the list of tables in the Navigation Pane.
13. Display the **2012Donations** table in **Datasheet View**.
14. Close the **MakeTable2012** query and the **2012Donations** table.

Create an Append Query

15. Open the **Children** table to view its records; close the table.
16. Choose **Create**→**Queries**→**Query Design** .
17. Add the **NewChildren** table to the query window. Close the Show Table dialog box.
18. Double-click the **asterisk (*)** in the NewChildren field list to add all the fields to the query grid.
19. Save the query as **Append Children**.
20. Choose **Design**→**Query Types**→ .
21. Click the **Append To Table Name** menu button and choose **Children**.
22. Leave the Current Database option selected and click **OK**.
Access adds the Append To row to the query grid and places the destination table name in the row.
23. Choose **Design**→**Results**→**Run** 
Access warns that you are about to append 10 rows to another table.
24. Choose **Yes**.



You can only see the changes after you open the destination table to which the records were appended. *Do not run the query again!*

25. Close the **Append Children** query, saving changes if prompted.
26. Open the **Children** table in **Datasheet View** to verify that the records were appended.
The Children table should contain 27 children.
27. Close the **Children** table.

Create an Update Query

28. Display the **Activities** table in **Datasheet View** to see the current Hrs values.
Current activities are from two to four hours.
29. Close the **Activities** table.
30. Choose **Create**→**Queries**→**Query Design** .
31. Add the **Activities** table and close the **Show Table** dialog box.
32. Double-click the **Hours** field name in the **Activities** field list.
33. Choose **Design**→**Query Type**→  **Update**.
Access adds the Update row to the query grid.

34. Type **[Hours] /2** in the Update To row under **Hours** in the query grid.

Field:	Hours
Table:	Activities
Update To:	[Hours]/2
Criteria:	

This calculation will divide the current activity Hours value in half. Be sure to include square brackets around the Hours field name.

35. Save the query as **UpdateHours**.

36. Choose **Design**→**Results**→**Run** .

Access warns that you are about to update 25 rows.

37. Choose **Yes** to continue.

Do not run the query again.



38. Save and close the **Update Hours** query.

39. Display the **Activities** table in **Datasheet View** to see the new Hours values.

Because of the change to the Hours values, the End Time values are no longer correct. Change the End Time values, as desired, to reflect the difference in Hours.

40. Close the **Activities** table.

Create a Delete Query

41. Choose **Create**→**Queries**→**Query Design** .

42. Add the **Donations** table to the query. Close the **Show Table** dialog box.

43. Double-click the **DonationDate** field to add it to the query grid.

44. Type **Between 1/1/2012 and 12/31/2012** in the **Criteria** row for the **DonationDate** field and tap **[Enter]**.

Field:	DonationDate
Table:	Donations
Sort:	
Show:	<input checked="" type="checkbox"/>
Criteria:	Between #1/1/2012# And #12/31/2012#
or:	

45. Choose **Design**→**Query Type**→ **Delete**.

46. Save the query as **Delete2012Donations** and then run it.

Access warns that you are about to delete four rows from the specified table.

47. Choose **Yes** to remove all donations records meeting the 2012 criteria from the **Donations** table.

48. Close the **Delete2012Donations** query.

49. Open the **Donations** table to review the results.

The 2012 records are no longer in the Donations table.

50. Close all open objects and close the database. Exit **Access**.

51. Submit your final file based on the guidelines provided by your instructor.

To see examples of how your final file or files should look at the end of this exercise, go to the student resource center.

REINFORCE YOUR SKILLS AC08-R03

Create a Select Query, Report from a Query, Query Calculations, and Action Queries

Kids for Change needs several new queries. In this exercise, you will create a select query that counts donations and calculate the sum of all donations. You will also create a report from a query and change an existing select query to a parameter query. You will then create a make table query to append new records to a table, and an update query to reflect pay increases.

Create a Select Query

1. Start **Access**. Open **AC08-R03-K4C** from the **AC2013 Lesson 08** folder and save it as **AC08-R03-K4C-[FirstInitialLastName]**.
2. Choose **Create**→**Queries**→**Query Design** .
3. Double-click the **Donors** and **Donations** tables in the **Show Table** dialog box; then close the dialog box.
4. Double-click the **DonationType** field in the **Donations** field list to add it to the query grid.
5. Choose **Ascending** from the drop-down menu in the **Sort** row.
6. Add the **DonorLName** and **DonorFName** fields from **Donors** to the query grid.
7. Add the **DonationDate** and **Amount** fields from **Donations** to the query grid.
8. Save the query as **DonorsType**.
9. Choose **Design**→**Results**→**Run** .
The records are in alphabetical order by type, then name.

Create a Concatenated Query Field

10. Switch to **Design View**.
11. Click in the first available **Field** row after **Amount** and type the following, then tap **[Enter]**:
DonorType: [DonationType] + [DonorLName] + Left ([DonorFName] , 1)
This expression will produce a new DonorType field that consists of donation type, donor's last name, and donor's first initial.
12. Choose **Design**→**Results**→**Run** .
The new concatenated ID field is displayed.
13. Save the **DonorsType** query.

Add the Totals Row to a Query

14. Scroll down to the last few records in the datasheet.
15. Choose **Home**→**Records**→**Totals**.
16. Click the menu button in the Total row for the **Last Name** field and choose **Count**.

17. Click the menu button in the Total row for **Amount** and choose **Sum**.

	Total	30			\$22,050.00
				None	
				Sum	
				Average	
				Count	

18. Save the **DonorsType** query.

Create a Report Using a Query As Record Source

19. Choose **Create**→**Reports**→**Report Wizard** .
20. Choose **Query: DonorsType** from the **Tables/Queries** drop-down menu.
21. Add all the fields to the Selected Fields list; click **Next**.
22. Group by **DonationDate**; click **Next**.
23. Click **Summary Options**.
24. Choose the **Sum** option for **Amount**.

Field	Sum	Avg	Min	Max
Amount	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

25. Click **OK** to close the Summary Options window; click **Next**.
26. Choose the **Block** layout and **Portrait** orientation; click **Next**.
27. Name the report **DonorsType Report** and click **Finish**.
The report opens in Print Preview.
28. Click .
29. Switch to **Layout View**; resize and move the controls so all values are visible.
30. Switch to **Print Preview**.
31. Click  then save and close the **DonorsType Report**.

Modify a Query to Request a Parameter Value to Select Records

32. Right-click the **DonorsType** query in the Navigation Pane and choose **Design View**.
33. Type **[Enter Donor Type (Bus or Pvt) :]** in the **Criteria** row of the **DonationType** field in the query grid and tap **[Enter]**.
Be sure to include the brackets.
34. Save and close the **DonorsType** query.
35. Right-click the **DonorsType Report** in the Navigation Pane and choose **Print Preview**.
An Enter Parameter Value dialog box opens displaying the criteria you entered.

36. Type **Bus** for the Donor Type, and click **OK**.
The Donors Type Report includes only records for donors with a Type of Bus (or business).

37. Save and close the **DonorsType Report**.

Create a Make Table Query

38. Run the **Children List** query to see the records of all 27 children.

39. Switch to **Design View**.

40. Scroll to the right to the **BirthDate** field and type **<1/1/2002** in the **Criteria** row.

41. Choose **Design**→**Query Type**→.

42. Type **OlderChildren** in the **Table Name** text box, leave the Current Database option chosen, and click **OK**.

43. Run the query.

Access warns that you are about to paste three records into a new table.

44. Choose **Yes** to continue.

Access creates the table and adds it to the Navigation Pane.

45. Display the **OlderChildren** table in **Datasheet View**.

ChildID	ChildLastName	ChildFirstName	ChildStreet	ChildCity	ChildST	ChildZIP	ChildPhone	BirthDate	Mom
CasadoM	Casado	Marty	302 Waterside Ave	Bradenton	FL	34202	9415551652	11/24/1996	Sandy
JeffriesB	Jeffries	Bobby	47 Halyard Lane	Sarasota	FL	34232	9415556437	10/22/2000	Greta
CasadoA	Casado	Anna	302 Waterside Ave	Bradenton	FL	34202	9415551652	2/15/2000	Sandy

The datasheet lists the three records for the children born before January 1, 2002.

46. Save the query as **OlderChildren Query**; then close the query and the **OlderChildren** table.

Create an Append Query

47. Display the **Volunteers** table in **Datasheet View** to see the records of all nine volunteers; then close the table.

48. Choose **Create**→**Queries**→**Query Design** .

49. Add the **NewVolunteers** table to the query window. Close the Show Table dialog box.

50. Double-click the **asterisk (*)** in the **NewVolunteers** field list to add all the fields to the query grid.

51. Save the query as **Append Volunteers**.

52. Choose **Design**→**Query Types**→.

53. Click the **Table Name** drop-down menu arrow and choose **Volunteers**.

54. Leave the Current Database option and click **OK**.
Access adds the Append To row to the query grid and places the destination table name in the row.

55. Choose **Design**→**Results**→**Run** .
Access warns that you are about to append six rows to another table.

56. Choose **Yes** to proceed.



You will only see the changes after you open the destination table to which the records were appended. *Do not run the query again!*

57. Close the **Append Volunteers** query, saving changes if prompted.

58. Display the **Volunteers** table in **Datasheet View** to see the appended records.
The datasheet includes new volunteers 10 through 15.

59. Close the **Volunteers** table.

Create an Update Query

60. Display the **PaidStaff** table in **Datasheet View** to see the current HrlySal amounts.

61. Close the **PaidStaff** table.

62. Choose **Create**→**Queries**→**Query Design** .

63. Double-click the **PaidStaff** table in the Show Table dialog box. Close the dialog box.

64. Double-click the **HrlySal** field name in the field list to add it to the query grid.

65. Choose **Design**→**Query Type**→ **Update** to change the query to an update action query.

66. Type **[HrlySal] *1.05** in the **Update To** row of the **HrlySal** field.

Field:	HrlySal
Table:	PaidStaff
Update To:	[HrlySal]*1.05
Criteria:	

Include the brackets around the HrlySal field name.

67. Choose **Design**→**Results**→**Run** .
Access warns that you are about to update 18 row(s).

68. Choose **Yes** to continue.

69. Close the query, naming it **UpdatePay**.

70. Display the **PaidStaff** table in Datasheet View.
The HrlySal amounts for paid K4C staffers are increased by 5%.

71. Save and close the **PaidStaff** table.

Create a Delete Query

72. Choose **Create**→**Queries**→**Query Design** .

73. Add the **Children** table to the query and close the Show Table dialog box.

Hrly Sal	▼
\$21.26	
\$18.90	
\$18.64	
\$17.33	
\$22.05	
\$19.43	
\$18.38	

74. Double-click the **BirthDate** field to add it to the query grid.
75. Choose **Design**→**Query Type**→ **Delete** to create a delete action query.
76. Type <1/1/2002 in the **Criteria** row for the **BirthDate** field and tap **Enter**.

Field:	BirthDate
Table:	Children
Delete:	Where
Criteria:	<#1/1/2002#
or:	

77. Save the query as **DeleteOlderKids** and then run it.
78. Choose **Yes** to remove the three Children records that meet the date criteria.
79. Close the **DeleteOlderKids** query.
80. Open the **Children** table to verify that the older children Marty Casado, Anna Casado, and Bobby Jeffries are no longer in the table.
The records of these three older children are now stored in the OlderChildren table that you created when you ran the make table query earlier in this exercise.
81. Close all open objects then close the database. Exit **Access**.
82. Submit your final file based on the guidelines provided by your instructor.

Apply Your Skills



APPLY YOUR SKILLS AC08-A01

Create and Work with Queries

Universal Corporate Events, Ltd. is embarking on its second year under new leadership and wants to optimize data retrieval. In this exercise, you will use a select query to create a new concatenated ID field, add a Total row to count events and generate sums for revenue, taxes, and net revenue. Finally, you will create a parameter query that looks up personnel by city.

Create a Query with a Concatenated Field

1. Start **Access**. Open **AC08-A01-UCE** from the **AC2013 Lesson 08** folder and save it as **AC08-A01-UCE- [FirstInitialLastName]**.
2. Choose **Create**→**Queries**→**Query Design**.
3. Add the **Venues** and **VenueLiaisons** tables to the query. Close the Show Table dialog box.
4. Add the **LiaisonLName**, **LiaisonFName**, **LiaisonPhone**, and **LiaisonEmail** fields in the **VenueLiaisons** table to the query grid.
5. Add the **VenueName** field from the **Venues** table to the query grid.
6. Save the query as **VenueLiaisonID Query** and run it.
7. Switch to **Design View**.
8. Type **VenLiaID: [VenueID] + [LiaisonLName]** in the first available Field row after **VenueName** and tap **Enter**.
9. Select the new concatenated **VenLiaID** field and drag it to the left of **LiaisonLName** so that it is now the first field in the query grid.
10. Save and run the query.
The query results datasheet shows a new VenLiaID field that concatenates the VenueID and Liaison Last Name so that event organizers know who to contact regarding each venue.
11. Close the **VenueLiaisonID Query**.

Add the Totals Row to a Query

12. Open the **Event Revenue** query and scroll down to the last few records.
13. Choose **Home**→**Records**→**Totals**.
14. Choose **Count** from the list of functions in the Total row for the **Liaison** field.
15. Choose **Sum** for the **TotRevenue**, **Taxes**, and **NetRevenue** fields.
The Total row should display a Count of 42 for the number of staffed activities, Tot Revenue of \$184,235.00, Taxes of \$18,423.50, and Net Revenue of \$165,811.50.
16. Save and close the **Event Revenue** query.

Modify a Query to Request a Parameter Value to Select Records

17. Choose **Create**→**Queries**→**Query Design**.
18. Add the **Personnel** table to the query and close the Show Table dialog box.
19. Add these fields to the query grid: **PerLastName**, **PerFirstName**, **PerAddr**, **PerCity**, **PerPhone**, and **PerEmail**.
20. Type [**Enter City:**] in the **Criteria** row of the **PerCity** field in the query grid.
Be sure to include the brackets.
21. Run the query.
22. Type **Sarasota** in the **Enter Parameter Value** dialog box, and click **OK**.
The query results displaying five Personnel records with a City value of Sarasota.
23. Save the query as **CitySelect Query**.
24. Close all open objects and close the database. Exit **Access**.
25. Submit your final file based on the guidelines provided by your instructor.
To see examples of how your final file or files should look at the end of this exercise, go to the student resource center.

APPLY YOUR SKILLS AC08-A02

Create Action Queries

Universal Corporate Events, Ltd. needs several action queries in its database to handle specific situations. In this exercise, you will create a make table query to store older events, a query to append new records to the Schedules table, and an update query that changes personnel salary. Finally, you will create query to delete older events from the main Schedules table.

Create Make Table and Append Queries

1. Start **Access**. Open **AC08-A02-UCE** from the **AC2013 Lesson 08** folder and save it as **AC08-A02-UCE- [FirstInitialLastName]**.
2. Click the **Schedules Query** in the Navigation Pane and press **[Ctrl] + [C]**; then press **[Ctrl] + [V]** to create a copy of the Schedules Query.
3. Name the new query **MakeTable Old Events**.
4. Display the query in **Design View**.
5. In the Criteria row of the **EventDate** field, type **<01/01/2014**.
6. Choose **Design**→**Query Type**→**Make Table**.
7. Name the table **OlderEvents** and click **OK**.
8. Save and run the **MakeTable Old Events** query.
9. Choose **Yes** to paste the six rows and close the query.
10. Display the **OlderEvents** table in **Datasheet View**.
The datasheet includes only records of events scheduled before 2014.

11. Close the **OlderEvents** table.
12. Open **AC08-A02-UCE-Append** from the **AC2013 Lesson 08** folder and save it as **AC08-A02-UCE-Append-[FirstInitialLastName]**.
13. Display the **NewSchedules** table in **Datasheet View** to see the new records; close the table.
14. Choose **Create**→**Queries**→**Query Design**.
15. Add the **NewSchedules** table to the query window. Close the Show Table dialog box.
16. Double-click the asterisk (*) in the **NewSchedules** field list to add all the fields to the query grid.
17. Save the query as **Append Schedules**.
18. Choose **Design**→**Query Types**→**Append**.
19. Choose the **Another Database** option and click **Browse**.
20. Navigate to your **AC2013 Lesson 08** folder and double-click **AC08-A02-UCE-[FirstInitialLastName]**.
21. Choose **Schedules** for the **Table Name**.
22. Click **OK**.
Access adds the Append To row to the query grid and places the destination table name in the row.
23. Run the query, choosing **Yes** when warned that you are about to append 30 rows.
24. Close the **Append Schedules** query, saving changes if prompted.
25. Close **AC08-A02-UCE-Append-[FirstInitialLastName]**. Return to **AC08-A02-UCE-[FirstInitialLastName]**.
26. Display the **Schedules** table in **Datasheet View** to these the appended records.
The main Schedules table should now contain 72 records.
27. Close the **Schedules** table.

Create Update and Delete Queries

28. Display the **SalaryGrades** table in **Datasheet View** to see the current salaries; then close the table.
29. Choose **Create**→**Queries**→**Query Design**.
30. Add the **SalaryGrades** table and close the **Show Table** dialog box.
31. Double-click the **SalaryAmt** field name to add it to the query grid.
32. Choose **Design**→**Query Type**→**Update**.
Access adds the Update row to the query grid.
33. Type **[SalaryAmt] * .9** in the **Update To** row.
This calculation will result in a 10% reduction—leaving all of the salaried personnel who have agreed to a 10% salary reduction for the next year with 90% of their former salary.
34. Save the query as **Update Salaries**.

35. Run the query clicking **Yes** when warned that you are about to update 21 rows.
36. Save and close the **Update Salaries** query.
37. Display the **SalaryGrades** table in **Datasheet View** to see the new values.

Grade	Grade Name	Salaried	Salary
A1	President	<input checked="" type="checkbox"/>	\$247,500.00
B1	Executive-Base	<input checked="" type="checkbox"/>	\$90,000.00
B2	Executive-1st Level	<input checked="" type="checkbox"/>	\$99,000.00
B3	Executive-2nd Level	<input checked="" type="checkbox"/>	\$108,000.00
B4	Executive-3rd Level	<input checked="" type="checkbox"/>	\$117,000.00
C1	Management-Base	<input checked="" type="checkbox"/>	\$45,000.00

38. Close the **SalaryGrades** table.
39. Choose **Create**→**Queries**→**Query Design**.
40. Add the **Schedules** table to the query and close the Show Table dialog box.
41. Double-click the **EventDate** field to add it to the query grid.
42. Type <1/1/2014 in the **Criteria** row for **EventDate** and tap **[Enter]**.
43. Choose **Design**→**Query Type**→**Delete**.
44. Save the query as **Delete Old Events** and then run it, choosing **Yes** when Access warns that you are about to delete six rows from the specified table.
45. Close the **Delete Old Events** query.
46. Open the **Schedules** table.
47. Right-click the **Event Date** column heading and choose **Sort Oldest to Newest** to verify that only 2014 records are left in the table.
48. Save and close the **Schedules** table, and close the database. Exit **Access**.
49. Submit your final file based on the guidelines provided by your instructor.

To see examples of how your final file or files should look at the end of this exercise, go to the student resource center.

APPLY YOUR SKILLS AC08-A03

Optimize Database Performance

Universal Corporate Events, Ltd must optimize database performance to fine-tune its operation. In this exercise, you will calculate weekly wages for hourly workers and assign new IDs based on job category, and then add a Totals row to count personnel and calculate wages. You will also create a report for a specific date range, and then create queries to: store hourly personnel records, append new records to another table, increase menu prices, and delete a closed venue.

Create Calculated and Concatenated Fields in a Query

1. Start **Access**. Open **AC08-A03-UCE** from the **AC2013 Lesson 08** folder and save it as **AC08-A03-UCE- [FirstInitialLastName]**.
2. Choose **Create**→**Queries**→**Query Design**.
3. Add the **Personnel** and **SalaryGrades** tables to the query window.
4. Add **PerLastName** and **PerFirstName** from the **Personnel** table.
5. Add **SalType**, **SalaryGradeName**, and **SalaryAmt** from the **SalaryGrades** table.
6. In the first available column after **SalaryAmt**, type **WrkWk: [HrlyWage] *40** in the Field row.
7. In the next column, type **SalaryID: [SalaryGrade] + [PerLastName]** in the Field row.
8. Save the query as **Personnel Wages** and run it.
9. Choose **Home**→**Records**→**Totals**.
10. In the **First Name** column, choose **Count** from the Total row list.
11. In the **WrkWk** columns, choose **Sum** from the Total row list.

Blake	Serena	<input type="checkbox"/>	Waitstaff-2nd Level		\$240.00	E3Blake
Tristan	Karen	<input type="checkbox"/>	Waitstaff-2nd Level		\$240.00	E3Tristan
Adams	Chuck	<input type="checkbox"/>	Bartender-Basic		\$220.00	F1Adams
Goldstein	Marv	<input checked="" type="checkbox"/>	Event Organizer	\$16,200.00		H1Goldstein
Kenworthy	Sally Anne	<input checked="" type="checkbox"/>	Event Organizer	\$16,200.00		H1Kenworth
Benson	Harold	<input checked="" type="checkbox"/>	Event Organizer	\$16,200.00		H1Benson
Wright	Fran	<input checked="" type="checkbox"/>	Event Organizer	\$16,200.00		H1Wright
*		<input type="checkbox"/>				
Total		20			\$1,720.00	

The datasheet displays the calculated 40-hour weekly wage for hourly workers, the concatenated ID comprised of each employee's salary grade and last name, the count of personnel, and the total of weekly wages for the hourly workers.

12. Save and close the **Personnel Wages** query.

Create a Report from a Parameter Query

13. Make a copy of the **Event Revenue** query and name it **Select Event Revenue**.
14. Open **Select Event Revenue** in **Design View**.
15. In the **EventDate** column in the query grid, type **Between [Enter Start Date] And [Enter End Date]** in the **Criteria** row.
16. Run the query.
17. Type **1/1/2014** for the **Enter Start Date** in the Enter Parameter Value box and click **OK**.
18. Type **6/30/2014** for the **Enter End Date** in the Enter Parameter Value box and click **OK**.
The datasheet should return records from January through June 2014.
19. Save and close **Select Event Revenue**.
20. Start the Report Wizard, choose **Query: Select Event Revenue** and select all the fields except **LiaisonID**.
21. Group the report by **VenueID**; open the Summary Options box, and check the boxes to **Sum** TotalRev, TaxDue, and NetRev.
22. Choose the **Outline** layout and the **Landscape** orientation.
23. Name the report **Selected Events** and click **Finish**.
24. Type **6/1/2014** for **Enter Start Date** in the Enter Parameter Value box. Click **OK**.
25. Type **6/30/2014** for **Enter End Date** in the Enter Parameter Value box. Click **OK**.
You may have to resize and move controls to better display the data.
26. Save and close the **Selected Events** report.

Create Make Table and Append Queries

27. Choose **Create**→**Queries**→**Query Design**, then add the **Personnel** and **SalaryGrades** tables to the query grid, and close the Show Table dialog box.
28. Add all the fields (*) from the **Personnel** table to the query grid.
29. Add the **SalType** field from the **SalaryGrade** table.
30. Clear the checkbox in the **Show** row of the **SalType** field.
31. Type **No** in the **Criteria** row of the **SalType** field.
No means that only records with an unchecked box (hourly employees) are selected.
32. Choose **Design**→**Query Type**→**Make Table**.
33. Name the table **HourlyPersonnel** and click **OK**.
34. Save the query as **MakeTable Hourly** and run it.
35. Choose **Yes** to paste the eight rows.
36. Close the **MakeTable Hourly** query.
37. Display the **HourlyPersonnel** table in **Datasheet View**.
The eight new records appear.

38. Close the **HourlyPersonnel** table.
39. Display the **NewMenus** table in Datasheet View to see the new records; close the table.
40. Choose **Create**→**Queries**→**Query Design**.
41. Add the **NewMenus** table to the query window and close the Show Table dialog box.
42. Double-click the **asterisk (*)** in the **NewMenus** field list to add all the fields to the query grid.
43. Save the query as **Append Menus**.
44. Choose **Design**→**Query Types**→**Append**.
45. Choose **Menus** for the **Table Name**.
46. Keep the **Current Database** option and click **OK**.
47. Run the query, choosing **Yes** to append seven rows.
48. Close the **Append Menus** query, saving changes if prompted.
49. Display the **Menus** table in **Datasheet View** to view the appended records.
The Menus table now contains 26 records, including the new items added to their menu selection.
50. Close the **Menus** table.

Create Update and Delete Queries

51. Choose **Create**→**Queries**→**Query Design**.
52. Add the **Menus** table to the query window and close the Show Table dialog box.
53. Add the **ChgPP** field to the query grid.
54. Choose **Design**→**Query Type**→**Update**.
Access places the Update To row in the query grid.
55. Type **[ChgPP] *1.05** in the Update To row of the ChgPP field.
56. Save the query as **Update Prices** and **run** it.
57. Choose **Yes** when Access warns to update the 26 rows.
58. Close the **Update Prices** query.
59. Display the **Menus** table in **Datasheet View** to see the 5 percent increase in menu prices.
60. Close the **Menus** table.
61. Choose **Create**→**Queries**→**Query Design**.
62. Add the **Schedules** table to the query grid and close the Show Table dialog box.
63. Add **VenueID** to the query grid.
64. Choose **Design**→**Query Type**→**Delete**.
Access adds a Delete row to the query grid.
65. Type **SaraYC** in the **Criteria** row for VenueID.

66. Save the query as **Delete Venue** and then run it, choosing **Yes** to delete two rows from the specified table.
67. Close the **Delete Venue** query.
68. Open the **Schedules** table; right-click the **VenueID column heading** and choose **Sort Z to A** to verify that the **SarasotaYC** records are no longer in the table.
The Sarasota Yachters' two scheduled events are no longer in the datasheet.
69. Save the **Schedules** table. Close all open objects and close the database. Exit **Access**.
70. Submit your final file based on the guidelines provided by your instructor.

Extend Your Skills



In the course of working through the Extend Your Skills exercises, you will think critically as you use the skills taught in the lesson to complete the assigned projects. To evaluate your mastery and completion of the exercises, your instructor may use a rubric, with which more points are allotted according to performance characteristics. (The more you do, the more you earn!) Ask your instructor how your work will be evaluated.

AC08-E01 That's the Way I See It

Blue Jean Landscaping needs to update the content of its database tables. Use the database you used in **AC07-E01**, or open **AC08-E01-BJL** and save your file to the **AC2013 Lesson 08** folder as **AC08-E01-BJL-[FirstInitialLastName]**.

First, concatenate CustIDfield and CustLastNamefield in the Customers table; name the field **NewCustID**. Add a calculated field to the Merch Sales Query that multiplies price by quantity sold. Add a Totals row and sum all sales. Next, create a report to request date range parameters using the Service Invoices Query as the record source with an InvDate date range of 10/15/2013–2/15/2014. Copy the Service Invoices Query and name it **Service Invoices Query 2013**. Select records with a 2013 InvoiceDate; name the new table **ServiceInvoices2013**. Finally, create an append query using **NewMerchandise** as the source and **StoreMerchandise** as the destination table for the new source table. Create an update query to increase prices in the StoreMerchandise table by 2% and a delete query to delete Service Invoices created in 2013.

You will be evaluated based on the inclusion of all elements, your ability to follow directions, your ability to apply newly learned skills to a real-world situation, your creativity, and your accuracy in creating objects and/or entering data. Submit your final file based on the guidelines provided by your instructor.

AC08-E02 Be Your Own Boss

Business has picked up at Blue Jean Landscaping and you must modify the database to ensure it is more efficient and can cope with unexpected situations. Open **AC08-E02-BJL** and save it to your **AC2013 Lesson 08** folder as **AC08-E02-BJL-[FirstInitialLastName]**.

First, in the MerchID field (StoreMerchandise table), concatenate the current ID and the Category field value. Count the number of items in the Store Inventory query to estimate whether you need a larger warehouse. Sum the Inventory Amt fields to find the total inventory value. Create a report that uses the Store Inventory query as the record source, and a parameter query to request a customer's last name to see their purchases. Copy the Merch Sales Query and name it **Sales by Customer**. Create a query to store 2013 sales records in a new table. Copy the Merch Sales Query and name it **Merch Sales Query 2013**. Modify the query to return records with a SalesDate from 2013. Name the new table **MerchSales2013**. Finally, create an append query to add all records in the NewCustomers to the Customers table, an update query to raise the AcreRate (Services table) by 5 percent, and a delete query to remove 2013 records from the MerchSales table.

You will be evaluated based on the inclusion of all elements, your ability to follow directions, your ability to apply newly learned skills to a real-world situation, your creativity, and your accuracy in creating objects and/or entering data. Submit your final file based on the guidelines provided by your instructor.

Transfer Your Skills



In the course of working through the Transfer Your Skills exercises, you will use critical-thinking and creativity skills to complete the assigned projects using skills taught in the lesson. To evaluate your mastery and completion of the exercises, your instructor may use a rubric, with which more points are allotted according to performance characteristics. (The more you do, the more you earn!) Ask your instructor how your work will be evaluated.

AC08-T01 Use the Web As a Learning Tool

Throughout this book, you will be provided with an opportunity to use the Internet as a learning tool by completing WebQuests. According to the original creators of WebQuests, as described on their website (WebQuest.org), a WebQuest is “an inquiry-oriented activity in which most or all of the information used by learners is drawn from the web.” To complete the WebQuest projects in this book, navigate to the student resource center and choose the WebQuest for the lesson on which you are currently working. The subject of each WebQuest will be relevant to the material found in the lesson.

WebQuest Subject: Advanced Boolean expressions and their use in advanced queries

Submit your files based on the guidelines provided by your instructor.

AC08-T02 Demonstrate Proficiency

As owner of Stormy BBQ, you are in charge of refining the database at your Key West location, which first opened on November 15, 2013. Open **AC08-T02-SBQ** from the **AC2013 Lesson 08** folder and save it as **AC08-T02-SBQ-[FirstInitialLastName]**.

First, create a select query using the Staff table to return name and contact information for just those staffers who work at the Key West location (Store=5). Next, create a concatenated ID field in the MerchSales Query using the last five digits of the SKU and the Staffer Last Name fields to analyze staffer productivity, and sort the MerchSales Query by SKU to see which items were sold. Count the line items and total the opening day MerchSalesAmt fields. Use the Daily Receipts Query as the source for a report to print daily receipts. Sort the query by ItemID to analyze item sales. Copy the MerchSales Query to create two Parameter queries. Name one **Sales by Sales ID**, requesting SalesID; name the other **Sales by SKU**, requesting the SKU.

Tip: Use Merchandise Popup and MenuItem's Popup to access SKU and ItemID.

Create a query to store 2013 merchandise sales records in a new table; then create a query to delete those records from the MerchSales table. Finally, create a query to add Cabinet City's products (see the NewMerchandise table in **AC08-T02-SBQ-Append** in your **AC2013 Lesson 08** folder) to the existing Merchandise table and create an update query to increase the menu item prices by 5%.

Submit your final file based on the guidelines provided by your instructor.

EVALUATION ONLY