7 Creating Complex Queries

As your database grows, so will the need to quickly retrieve and modify exact data. Complex queries help with this because they allow you to further refine search results and perform actions that modify records. In this chapter, you will explore queries designed to enhance the timeliness and accuracy of large relational databases. You will create crosstab queries and use parameter queries that prompt you to enter values to generate or modify records. You will also create action queries to update databases and automate database tasks.

LEARNING OBJECTIVES

- Create a crosstab query
- Create a find unmatched query
- Create a find duplicates query
- Create and run parameter queries
- Create and run action queries

Project: 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 both data entry and updates, as well as to better analyze data. The tools you will use include crosstab queries for data analysis, parameter queries that will prompt the user for input, and action queries to update and maintain the database.

Crosstab Queries

Crosstab queries allow you to easily analyze data. A crosstab query lists the fields to be grouped on the left side (rows) of the datasheet, and it arranges the fields to be summarized across the top (columns) so you can calculate sums, averages, counts, or totals by both group and subgroup. For example, if you have a database that contains sales records for your employees, the description of each product they sell, and their total sales for each product, you could create a crosstab query to display the total sales by product for each employee.

Original Data

Employee	Product Description	Line Total
JFW	Secondary Page	$1,200.00
JFW	Image, Custom Designed	$440.00
JFW	Home Page, Nav, CSS, Design	$400.00
MJW	Image, Custom Designed	$560.00
MJW	Home Page, Nav, CSS, Design	$400.00
MJW	Secondary Page	$1,400.00
MJW	Hourly Rate for Modifications	$400.00
JMM	Image, Custom Designed	$240.00
JMM	Secondary Page	$400.00

The original data format is arranged by record.

Reorganized by Crosstab Query

Emp Name	Tot Sales	Home Pg	2nd Page	Blogs	Carts	Images	Hourly
Kramer	$13,680.00	$800.00	$7,600.00	$600.00		$2,520.00	$2,160.00
Mansfield	$10,520.00	$400.00	$4,800.00	$600.00	$1,200.00	$1,680.00	$1,840.00
Waters	$20,080.00	$1,600.00	$10,000.00	$1,200.00	$1,200.00	$2,080.00	$4,000.00
Winchester	$17,100.00	$2,000.00	$8,800.00	$300.00	$800.00	$3,040.00	$2,160.00

Using a crosstab query, you can display the data grouped by employee with totals for the various products.

Both tables and queries can be used as the basis of crosstab queries. You can create a crosstab query while working with an existing query in Design View using the Crosstab option; alternatively, use the Query Wizard.

Design→Query Type→Crosstab

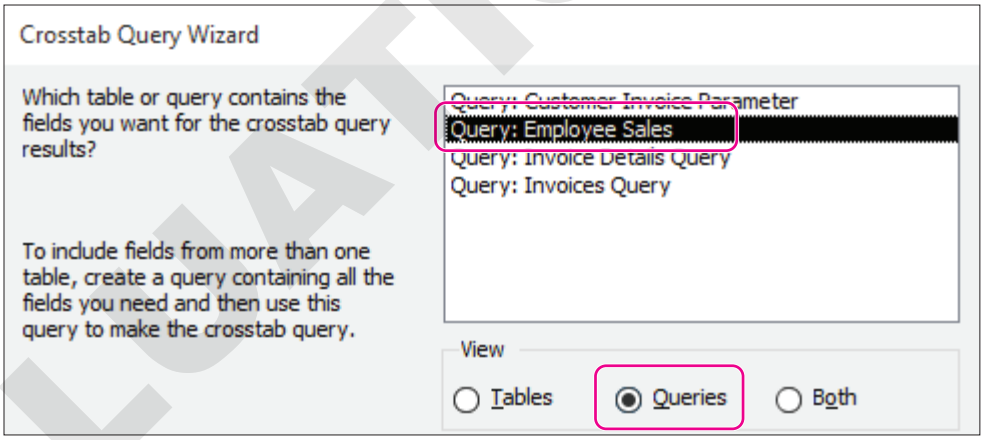
Watch the video "Creating Crosstab Queries."

DEVELOP YOUR SKILLS: A7-D1

In this exercise, you will create a crosstab query that lists every employee and their total invoice amount generated by product.

1. Open **A7-D1-WinDesign** from your **Access Chapter 7** folder and save it as: **A7-D1-WinDesignRev**
2. Double-click the **Employee Sales** query to run it and display the resulting datasheet.

   The query contains line item sales data. Each employee has multiple transactions, and each transaction contains the product description, price, and quantity. The LineTotal field is a calculated field that multiplies price by quantity. You will use this query as the basis for your crosstab query.
3. Close the Employee Sales query.
4. Choose **Create→Queries→Query Wizard**.
5. Choose **Crosstab Query Wizard** and click **OK**.
6. Choose the **Queries** view option and then choose **Query: Employee Sales** from the query list.



7. Click **Next** to accept the Employees Sales query as the basis of your crosstab query.
8. Choose **EmpLastName** from the Available fields list and add it to the Selected Fields list.

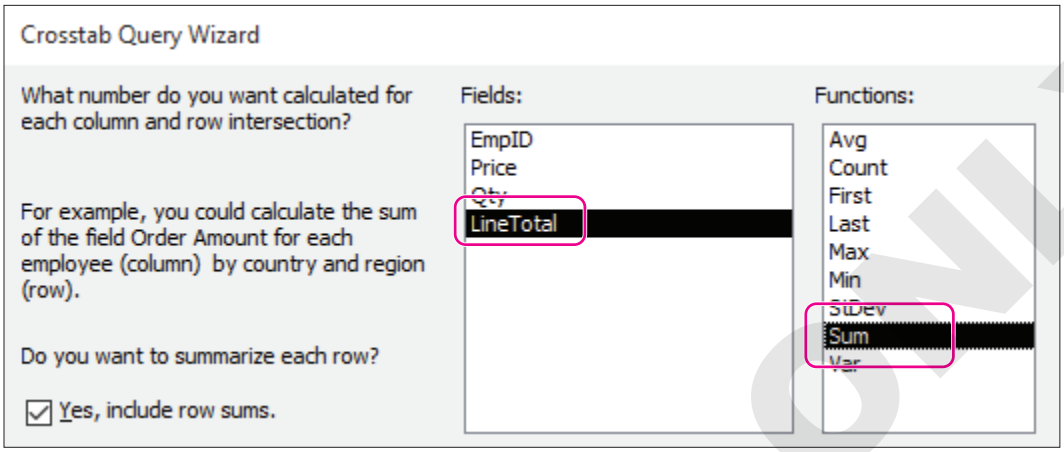
   Your crosstab query will display employee last names as row headings in the query results datasheet. Each employee will have a single row, with their last name displayed in the first cell of the row and their sales information displayed in the other row cells.
9. Click **Next** and choose **ProdDescription** for the column headings.

   The various product descriptions (Blog, Home Page, Web Page) will appear as column headings.

Tip! *The sample query at the bottom of the Wizard can be a useful guide for deciding which fields to use as the row and column headings.*

10. Click **Next**, choose **LineTotal** from the Fields list, and then choose **Sum** from the Functions list.

    The crosstab query will examine all transactions in the underlying Employee Sales query and sum the line totals for each product description. For example, a grand total of all line totals will be created where the product description is Blog and the employee is Kramer.



11. Leave the **Yes, Include Row Sums** option checked and click **Next**.

    This option creates one additional column in the datasheet to hold a total for each employee. The total will be the sum of all cells in the crosstab query datasheet for that employee.

12. Leave the default query name as *Employee Sales_Crosstab* and click **Finish**.

13. Take a moment to examine the query results.

Last Name	Total Of Lin	Blog	Home Page	Hourly Billir	Image	Shopping Ca	Web Page
Kramer	$13,680.00	$600.00	$800.00	$2,160.00	$2,520.00		$7,600.00
Mansfield	$10,520.00	$600.00	$400.00	$1,840.00	$1,680.00	$1,200.00	$4,800.00
Waters	$20,120.00	$1,200.00	$1,600.00	$4,000.00	$2,120.00	$1,200.00	$10,000.00
Winchester	$16,940.00	$300.00	$2,000.00	$2,160.00	$2,880.00	$800.00	$8,800.00

    Complete the Query Wizard steps again, if necessary, so you fully understand how the query options produce the resulting datasheet.

14. Close the query.

Find Queries

Database tables often contain common fields that link, or relate, the tables. For example, the Product ID field from a Products table also appears in an Invoices table, so invoices can show only existing products. Thus, it's important that records entered in one table have a matching record in the related table. That is, an invoice should never show a product that is not in the Products table.

Sometimes databases are poorly designed and incorrect data is allowed. For example, a user may enter a product that does not exist into a new invoice. Data is also sometimes imported from other data sources, which might result in incorrect or duplicate data. Fortunately, Access provides two additional Query Wizard options to help resolve these types of data conflicts.

Find Unmatched Query

The find unmatched query locates records in one table that have no matching records in another table. For example, you could create a find unmatched query to ensure each record in an Invoice table has a corresponding record in a Customers or Products table.

Find Duplicates Query

A find duplicates query locates records containing duplicate field values in a single table or query datasheet. For example, you could create a find duplicates query to locate all customers with the same last name in a Customers table or to find all customers located in a particular state or ZIP code.

DEVELOP YOUR SKILLS: A7-D2

In this exercise, you will create a query to locate records in the Customers table that do not have a matching customer ID in the Invoices table. Then you will create a query to identify records with duplicate customer last names.

1. Choose **Create**→**Queries**→**Query Wizard**, choose **Find Unmatched Query Wizard**, and click **OK**.
2. Click **Next** to choose the Customers table.

   When the query is created, records from the Customers table will appear in the query results.
3. Choose the **Invoices** table and click **Next**.

   You will set up the query to find records in the Invoices table that do not have a matching customer record in the Customers table. Notice that the CustID fields are chosen as the matching fields in the Wizard screen. They're automatically chosen because a one-to-many relationship is already set up between these fields in the Customers and Invoices tables.
4. Click **Next** to accept CustID as the matching field.
5. Add **CustLastName**, **CustFirstName**, **CustPhone**, and **CustEmail** to the Selected Fields list.

   These are the fields from the Customers table that will be displayed in the query results, creating a contact list of customers who don't have invoices in the system. In other words, these are customers who haven't purchased anything in a while, making them potential prospects for new sales.
6. Click **Next** and then click **Finish** to accept the default query name *Customers Without Matching Invoices*.

   Your query should produce a data set of three records only.

Customers Without Matching Invoices

Last Name	First Name	Telephone	Email
Abrams	John	(941) 555-9902	JPAbrams@email.com
Fleetwood	Candace	(941) 555-9256	CandyWin@email.com
Winkler	Samuel	(941) 555-2054	SamWinkler45@email.com

7. Close the datasheet after reviewing it.

Create a Find Duplicates Query

8. Choose **Create**→**Queries**→**Query Wizard**, choose **Find Duplicates Query Wizard**, and click **OK**.

9. Click **Next** to choose the Customers table as the query to search for duplicate field values.
10. Add **CustLastName** to the Duplicate Value Fields list and click **Next**.

    You are looking only for the records of customers who have the same last name.
11. Add **CustFirstName** and **CustPhone** to the Additional Query Fields list and click **Next**.

    These fields will appear in the resulting datasheet but aren't used in the duplicate values test. Only CustLastName is being checked for duplicates.
12. Name the query **`Customers with the Same Last Name`** and click **Finish**.

    The query results show just two customers with the same last name (Roberts).

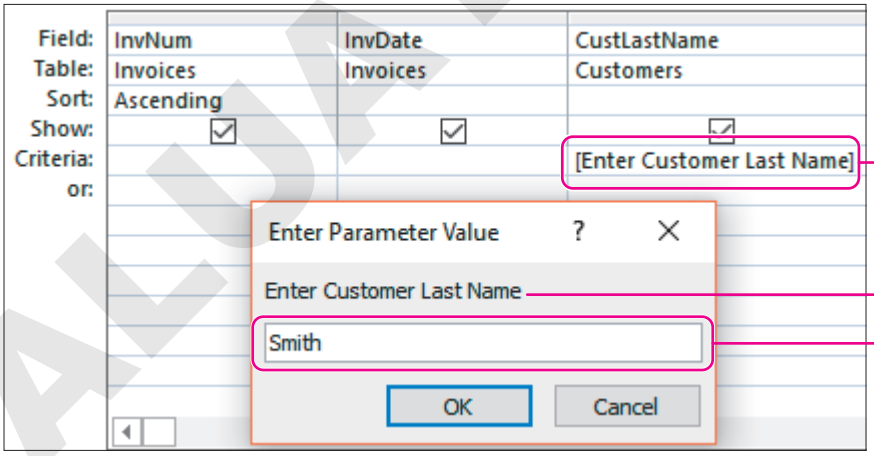
Customers with the Same Last Name

Last Name	First Name	Telephone
Roberts	John	(941) 555-7820
Roberts	Ilsa	(941) 555-7821

13. Close the query after reviewing the results.

Parameter Queries

A parameter query is a select query that prompts users to enter new criteria values each time the query is run. The query then generates results based on the value(s) entered. For example, a parameter query that searches for customers with a specific last name might prompt the user to enter the desired last name when the query is run. The query then returns only records containing the last name entered by the user. Parameter queries are created by enclosing the desired prompt text with square brackets, [ ], in the query Criteria row.



Surround the parameter query prompt text in square brackets in a Criteria cell.

The prompt text appears here when the query is run.

The user enters the desired parameter value here.

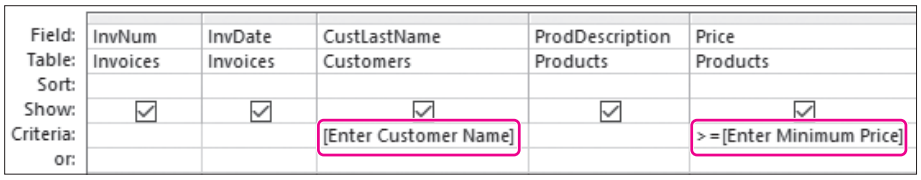
Customer Invoice Parameter

InvNum	Invoice Date	Last Nam	Description	Price	Qty	LineTotal
1	3 /14/2017	Smith	Home Page	$400	1	$400
1	3 /14/2017	Smith	Web Page	$200	6	$1,200
1	3 /14/2017	Smith	Image	$40	11	$440

In this example, only records in which the customer last name is *Smith* are returned.

Complex Parameter Queries

Suppose you want to see all items purchased by a particular customer and those equal to or greater than a particular price; for example, all items purchased by Smith with a price greater than or equal to $300. You can do this by creating an AND condition using parameters in the CustLastName and Price fields.



You can also create expressions with prompts for multiple values in the same query field or include logical criteria such as greater than (>), less than (<), and equal to (=).

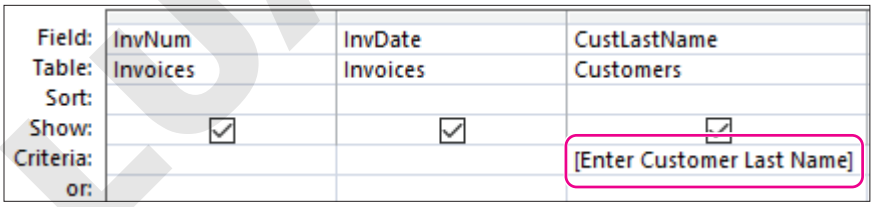
EXAMPLES OF PARAMETER QUERY CRITERIA FOR A SINGLE FIELD

Parameter Criteria	Result
Between [What is the start date?] And [What is the end date?]	These criteria prompt the user to enter start and end dates. Access recognizes the Between and And expressions and returns dates within the range entered.
>=[Enter minimum price]	This displays the prompt *Enter minimum price* and returns only records greater than or equal to the price entered.

DEVELOP YOUR SKILLS: A7-D3

In this exercise, you will use parameters to return customer records based on user input.

1. Display the **Customer Invoice Parameter** query in **Design View**.
2. Click in the **CustLastName** criteria field and enter the criterion: **`[Enter Customer Last Name]`**



3. Run the query, type **`Roberts`** in the parameter box that appears, and click **OK**.

   All line items from invoices 34 and 7, where the customer last name is Roberts, *are returned.*

Create an AND Parameter Condition

4. Switch to **Design View** and enter this parameter in the Price criteria box: **`>=[Enter Minimum Price]`**

5. Run the query and enter **Roberts** in the first parameter box and **300** in the second.

   Now the record set has records only with Roberts *in the Last Name field in which the price is greater than or equal to 300.*

Customer Invoice Parameter

InvNum	Invoice Date	Last Nam	Description	Price	Qty	LineTotal
34	8 /4 /2018	Roberts	Home Page	$400	1	$400
34	8 /4 /2018	Roberts	Blog	$300	1	$300
7	7 /10/2017	Roberts	Home Page	$400	1	$400
7	7 /10/2017	Roberts	Shopping Cart	$400	1	$400

6. Close the query, saving the changes.

Action Queries

An action query performs an action that modifies a database table or a group of records in a table. They 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 and data from other tables.

An action query is run whenever it is opened, so if you create an update query designed to increase prices by 10% on all items in a table, Access will increase those prices every time you run the query. Action queries do this without opening the underlying tables being modified by the query. For this reason, an action query may accidentally be run more than once, inadvertently changing the underlying table data multiple times. It's good practice *to not* save action queries after running them or to delete action queries if they are saved. This will help maintain the validity of the database as changes to the underlying data cannot easily be undone.

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.

Make Table Queries

A make table query is an action query that can create a new table using data from multiple database tables. It's also a great way to move data produced from a calculated query field into a table. 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. A reason to move records to another database, for example, would be to archive them when they become obsolete, such as when a product is no longer available. When you rerun a make table query, Access will replace the table created with the previous running of the query. To retain the previously created table, you must first rename it; this way it won't be replaced.

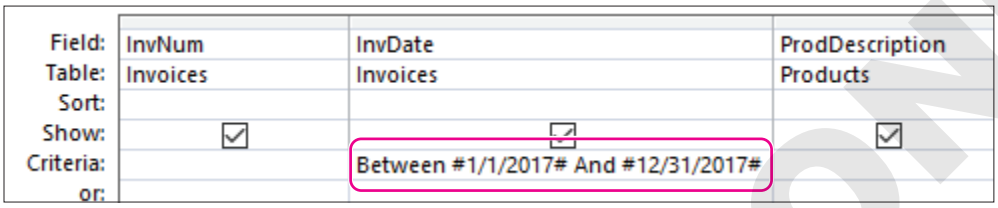
Design→Query Type→Make Table

DEVELOP YOUR SKILLS: A7-D4

In this exercise, you will create a make table action query to save all of the 2017 invoice records in a new table.

1. Open **Invoices Query** in **Design View**.
2. Enter **`Between 1/1/2017 And 12/31/2017`** as the criterion for the **InvDate** field.

   This criterion will produce a datasheet with old invoices no longer needed in the database. You'll then use the make table query feature to move the records to a new table.



3. Choose **Query Tools→Design→Query Type→Make Table**.
4. Enter **`2017 Invoices`** as the table name and click **OK**.
5. **Run** the query and choose **Yes** to paste 61 rows into a new table.

   The new table named 2017 Invoices appears at the top of the Tables section in the Navigation pane.
6. Open the new **2017 Invoices** table in **Datasheet View**.

   Notice that all of the line items listed have a 2017 invoice date.
7. Close the 2017 Invoices table and then close the invoices query without saving the changes.

   It's important not to save the query because it's used for other purposes and because you want to preserve the new table without risking an overwrite of it.
8. Open the **Invoices** table in **Datasheet View**.

   The Invoices table still contains the 2017 records.

Note! *Make table queries don't remove data from underlying tables; they simply copy the data to new tables.*

9. Close the Invoices table.

Append Queries

An append query 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 from a new products table to the existing products table. Or you might use an append query to automatically add new customers to the Customers table the first time a customer places an order.

Formatting the Source and Destination Tables

In an append query, the table that records are drawn from is called the source table. The table receiving the records is the destination table. To successfully run an append 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 that contains the source table. When the query is run, 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.

Design→Query Type→Append

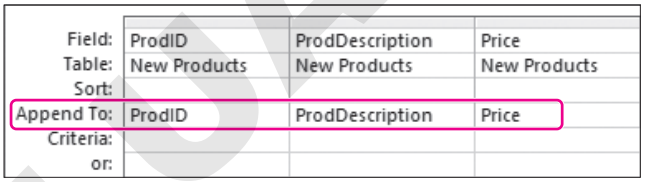
DEVELOP YOUR SKILLS: A7-D5

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 five records that will be appended to the Products table.

   The tables have the same field structure, which includes field order and matching field names and field data types.
3. Close both tables and then choose **Create→Queries→Query Design** to create a new query.
4. Add the **New Products** table to the query window and then close the Show Table dialog box.
5. Add all fields from the New Products table to the query grid in the same order they appear in the New Products list.
6. Choose **Query Tools→Design→Query Type→Append**.
7. Click the **Table Name menu** button, choose **Products**, and click **OK**.

   An Append To row is added to the query. When you run the query, it will copy all records from the underlying New Products table to the Products table. The tables have identical field structures, so the data will drop right into the existing table.



Field:	ProdID	ProdDescription	Price
Table:	New Products	New Products	New Products
Sort:			
Append To:	ProdID	ProdDescription	Price
Criteria:			
or:			

8. **Run** the query and choose **Yes** to append the five rows to the Products table.

   Nothing appears to happen when you run the query—but don't run it again! You will see the changes only after you open the destination table to which the records were appended. If you do run the query again, Access will add the same records to the destination table again, creating duplicate data.
9. Display the **Products** table in **Datasheet View** to verify that the new records were appended.

   The Products table should now contain 11 records.
10. Close the Products table and then close the new append query, saving it as: **`Append Products`**

Update Queries

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 every product in a specific category or to update the area code for phone numbers that change when the phone company adds or changes an area code. To ensure the 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 can 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.

Design→Query Type→Update

DEVELOP YOUR SKILLS: A7-D6

In this exercise, you will create an update action query that increases the prices of every item in the Products table by 10%.

1. Open the **Products** table in **Datasheet View** and notice the Home Page price is $400.

   The update query will increase this and all other prices by 10%.

2. Close the Products table and then choose **Create→Queries→Query Design**.

3. Add the **Products** table to the query window and then close the Show Table dialog box.

4. Add all fields from the Products table to the query grid in the same order they appear in the Products list.

5. Choose **Query Tools→Design→Query Type→Update**.

   An Update To row is added to the query grid.

6. Click in the **Update To** cell for the Price field and enter: **`[Price]*1.1`**

   Be sure to include the square brackets, [ ], when entering this formula so Access will recognize Price as a field. Multiplying by 1.1 increases the price by 10%.

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

7. **Run** the query and choose **Yes** when the warning prompt appears to update 11 rows.

8. Close the query without saving it.

   Once again, it's good practice not to save action queries, such as update queries. Running a query by accident can corrupt data and recovering corrupted data is often difficult or impossible to do.

9. Open the **Products** table in **Datasheet View** and notice the Home Page price went from $400 to $440 (an increase of 10%).

10. Close the Products table.

Delete Queries

A delete query removes 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.

Preparing for Delete Queries

To ensure corresponding records in related tables will be deleted concurrently, check the Cascade Delete Related Records checkbox in the Edit Relationships window. When you set up a delete query, Access replaces the Sort row of the query grid with a 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 you can set no criteria to remove all records from a table.

Design→Query Type→Delete

DEVELOP YOUR SKILLS: A7-D7

In this exercise, you will create a delete query to remove the 2017 invoices from the Invoices table.

1. Open the **Invoices** table in **Datasheet View**.

   The table still has invoices dated from 2017. These invoices were copied to the 2017 Invoices table using a make table query in a previous exercise. Because make table queries do not delete records, you'll take care of this using a delete query.

2. Close the Invoices table and then choose **Create→Queries→Query Design**.
3. Add the **Invoices** table to the query window and then close the Show Table dialog box.
4. Add only the **InvDate** field to the query grid.
5. Enter **`Between 1/1/2017 And 12/31/2017`** as the criterion for the InvDate field.

   Access may view typing 2017 as an attempt to insert the 2017 Invoices table. To avoid adding this table to the criteria, tap Spacebar at the end of the criteria and then press Enter.

6. Choose **Query Tools→Design→Query Type→Delete**.

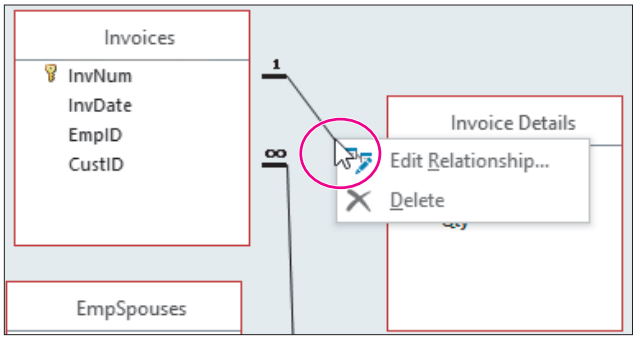
   A Delete row is added to the query grid.

7. **Run** the query and choose **Yes** when the prompt to delete 19 rows appears.

   A second warning appears, notifying you that the records cannot be deleted because there is a key violation. This is because the Cascade Deleted Records option is not activated for a relationship between the Invoices and Invoice Details table. This option must be activated for you to run delete queries.

8. Choose **No** in the warning message box and then choose **Database Tools→Relationships→Relationships**.

9. Right-click the join line between the Invoices and Invoice Details table and choose **Edit Relationship**.



10. Check the **Cascade Delete Related Records** box and click **OK**.
11. Close the Relationships window, saving the changes to the relationship layout if prompted.
12. **Run** ! the query again, choosing **Yes** when the warning prompt appears.

    This time, the delete query runs, removing the 2017 records from the Invoices table.
13. Close the query without saving it.

    Remember, it's good practice not to save action queries, especially when they are relatively easy to re-create, as in this example.
14. Open the **Invoices** table in **Datasheet View** and notice that the 2017 invoices have been removed.
15. Close the Invoices table and then close the database.

Self-Assessment

Check your knowledge of this chapter's key concepts and skills using the Self-Assessment in your ebook or online (eLab course or Student Resource Center).

Reinforce Your Skills

REINFORCE YOUR SKILLS: A7-R1

Create Crosstab and Find Queries

Kids for Change is fine-tuning its database. In this exercise, you will create a crosstab query to track donations and find queries to locate problem records.

1. Open the **A7-R1-K4C** database from your **Access Chapter 7** folder and save it as: **`A7-R1-K4CRev`**
2. Choose **Create→Queries→Query Wizard**  , choose **Crosstab Query Wizard** in the first screen, and click **OK**.
3. Choose the **Queries** view, choose **Donations Query** as the query that contains the fields you want in the results, and click **Next**.
4. In the next Wizard screen, move **DonorLName** to the Selected Fields list and click **Next**.

   Donor last names will become your row headings and the field where the results are grouped.
5. In the next Wizard screen, choose only **DonationDate** as the field to appear in the column headings and click **Next**.

   Because DonationDate is a date field, the Wizard asks you to choose an interval, such as day, month, or year.
6. Choose **Month** as the interval and click **Next**.
7. Choose **Amount** in the Fields list and **Sum** in the Functions list to identify the field that contains values and the function you want to use.
8. Click **Next**, leave the query name unchanged, and then finish the query.

   The query returns the total donations for each donor, organized by month.
9. Close the query.

Create a Find Unmatched Records Query

10. Launch the **Query Wizard**, choose **Find Unmatched Query Wizard**, and click **OK**.
11. Choose the **Activities** table and click **Next**.

    The Activities table will display in the query results.
12. Choose **Volunteers** as the table with related records and click **Next**.
13. Choose **Day** in the Activities table field list and **VolDay** in the Volunteers table field list and then click **Next**.
14. Add **Activity**, **Day**, and **MeetTime** to the Selected fields list and then click **Next**.
15. Click **Finish** to accept the default query name.

    Your query should return four records of activities that do not have a matching volunteer assigned to them.
16. Close the query when you have finished viewing the results.

Create a Find Duplicates Query

17. Launch the **Query Wizard**, choose **Find Duplicates Query Wizard**, and click **OK**.
18. Choose **Donors** as the table to check for duplicates and click **Next**.
19. Add **DonorLName** to the Duplicate-Value Fields list and click **Next**.

    You are looking only for records of donors with the same last name.
20. Add **DonorFName** and **DonorPhone** to the Additional Query Fields list and click **Next**.
21. Accept *Find Duplicates for Donors* as the default query name and click **Finish**.

    The query should return records for Clay Boltwood and Nancy Boltwood.
22. Close the database, saving the changes to any unsaved queries.

REINFORCE YOUR SKILLS: A7-R2

Create a Parameter Query

In this exercise, you will make a copy of an existing query. You will modify the copied query, turning it into a parameter query to return donor records by state.

1. Open the **A7-R2-K4C** database from your **Access Chapter 7** folder and save it as: **`A7-R2-K4CRev`**
2. Click **Donations Query** in the Navigation pane to select it.
3. Press Ctrl+C to copy and Ctrl+V to paste the copy.
4. Enter **`Donations by State`** as the new query name and click **OK**.
5. Display the **Donations by State** query in **Design View**.
6. Type **`[Enter State Abbreviation]`** in the Criteria cell of the DonorST field and tap Enter to complete the entry.

Field:	DonorID	DonorLName	DonorFName	DonorStreet	DonorCity	DonorST
Table:	Donations	Donors	Donors	Donors	Donors	Donors
Sort:	Ascending					
Show:	☑	☑	☑	☑	☑	☑
Criteria:						[Enter State Abbreviation]
or:						

7. **Run** ! the query.
8. Type **`MA`** in the Parameter Value prompt box and click **OK**.

   Only donations from Massachusetts donors are returned.
9. Close the query, saving the changes.
10. Choose **File→Close** to close the database; if you see a message regarding emptying the Clipboard, click **Yes**.

REINFORCE YOUR SKILLS: A7-R3

Create Action Queries

Kids for Change is improving and updating its records. In this exercise, you will create a make table query that produces a table to archive the 2017 donation records, 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.

1. Open the **A7-R3-K4C** database from your **Access Chapter 7** folder and save it as: **`A7-R3-K4CRev`**

   Remember, you must enable a database if you will be running action queries, so click the Enable Content button that appears after saving the file.

   To begin, you will create the make table query.
2. Open **Donations Query** and switch to **Design View**.
3. Scroll the query grid to the right and type **`Between 1/1/2017 And 12/31/2017`** in the Criteria cell of the DonationDate field.
4. Choose **Query Tools→Design→Query Type→Make Table**.
5. Type **`2017 Donations`** as the name of the new table and click **OK**.
6. **Run** the query and choose **Yes** in the warning box.
7. Close the Donations Query without saving the changes.

Create an Append Query

8. Open the **Children** table and notice that it contains 17 records; close the table.
9. Choose **Create→Queries→Query Design** to create a new query.
10. Add the **NewChildren** table to the query window and then close the Show Table dialog box.
11. Add all fields from the NewChildren table to the query grid in the same order they appear in the NewChildren list.
12. Choose **Query Tools→Design→Query Type→Append**.
13. Click the **Table Name menu** button ▾, choose **Children**, and click **OK**.

    The Append To row is added to the query grid.
14. **Run** the query and choose **Yes** to add the 10 rows to the Children table.
15. Close the query, saving it as: **`Append New Children`**

Create an Update Query

16. Display the **Activities** table in **Datasheet View** to see the current Hrs values (*2 or 4 hours*) and then close the table.
17. Choose **Create→Queries→Query Design**.
18. Add the **Activities** table and close the Show Table dialog box.
19. Add the **Hours** field to the query grid.
20. Choose **Query Tools→Design→Query Type→Update**.

    An Update To row is added to the query.

21. Type **`[Hours]/2`** in the Update To cell of the Hours field.

    It's important to include the square brackets so that Access can perform the correct calculation. This calculation will divide the current activity hours value in half.

22. **Run** ! the query and choose **Yes** to update 25 rows; close the query without saving it.
23. Open the **Activities** table and note the activities that were listed as 2 and 4 hours each are now 1 and 2 hours; close the table.

Create a Delete Query

24. Open the **Donations** table and notice it contains donations from the year 2017; close the Donations table.
25. Choose **Create→Queries→Query Design**  .
26. Add the **Donations** table to the query and then close the Show Table dialog box.
27. Double-click the **DonationDate** field to add it to the query grid.
28. Type **`Between 1/1/2017 And 12/31/2017`** in the Criteria row for the DonationDate field.
29. Choose **Query Tools→Design→Query Type→Delete**  .

    A Delete row is added to the query grid.

30. **Run** ! the query and choose **Yes** in the warning box.
31. Close the query without saving it and then open the **Donations** table and confirm that the 2017 records have been removed.
32. Close the database.

Apply Your Skills

APPLY YOUR SKILLS: A7-A1

Create Crosstab and Find Queries

Universal Corporate Events has asked you to create queries to analyze the company's data and identify unmatched and duplicate database records. In this exercise, you will respond to this request by creating crosstab, find unmatched records, and find duplicate records queries.

1. Open the **A7-A1-UCE** database from your **Access Chapter 7** folder and save it as: **`A7-A1-UCERev`**
2. Use the **Query Wizard** to create a crosstab query using these parameters:

View	**Query: Event Revenue**
Row Heading(s)	**VenueID**
Column Heading(s)	**ContactID**
Field(s)	**TotalRev**
Function(s)	**Sum**
Name	**`Contact Revenue by Venue`**

3. Finish the query.

   Seven data rows should be returned.
4. Close the query and then use the **Query Wizard** to create a find unmatched query using these parameters:

View	**Table: Venues**
Related Records	**Table: Schedules**
Fields in Venues	**VenueID**
Fields in Schedules	**VenueID**
Fields to see in query results	**VenueName, VenueStreet, VenueCity, VenueST, VenueZIP, VenuePhone, VenueWebSite**
Name	**`Venues Without Event Scheduled`**

5. Finish the query.

   Three data rows should be returned.
6. Close the query and then use the **Query Wizard** to create a find duplicates query using these parameters:

View	**Query: Event List**
Duplicate-Value Field	**EventDate**
Additional Fields	**VenueID, ContactID, MenuPlan, Guests**
Name	**`Find Double-Booked Dates`**

7. Finish the query.

   Two data rows should be returned.
8. Close the database.

APPLY YOUR SKILLS: A7-A2

Create a Parameter Query

In this exercise, you will create a parameter query to return personnel records by city.

1. Open the **A7-A2-UCE** database from your **Access Chapter 7** folder and save it as: **A7-A2-UCERev**
2. Create a new query using **Query Design** and add the **Personnel** table to the query.
3. Add the **PerLastName**, **PerFirstName**, **PerAddr**, **PerCity**, **PerPhone**, and **PerEmail** fields to the query.
4. Make **[Enter City]** a criterion for the PerCity field.
5. Run the query using **Sarasota** as the parameter value.

   The query should return five records in which the city is Sarasota.
6. Close the database, saving the query as: **Personnel City**

APPLY YOUR SKILLS: A7-A3

Create Action Queries

Universal Corporate Events is updating and consolidating its events data. In this exercise, you will create a make table query to archive the records for older events, an append query to add new records to the Schedules table, and an update query to change personnel salaries. Finally, you will create a query to delete older events from the main Schedules table.

1. Open the **A7-A3-UCE** database from your **Access Chapter 7** folder and save it as: **A7-A3-UCERev**

   You will start by creating a make table query.
2. Display **Schedules Query** in **Design View** and add the criterion **<01/01/2019** to the EventDate field.
3. Use the **Make Table** query type to create a new table with the name: **Older Events**
4. Run the query.

   The new table should contain six records.
5. Close the Schedules Query without saving the changes.

Create an Append Query

6. Create a new query, adding all fields from the **New Schedules** table.
7. Use the **Append** query type to convert the query to an append query using **Schedules** as the table name to append to.
8. Run the query, choosing **Yes** when asked if you wish to add the 30 rows to the Schedules table.
9. Close the query, saving it as: **Append Schedules**
10. Open the **Schedules** table to verify it contains 72 records; close the table.

Create an Update Query

11. Create a new query using the **SalaryGrades** table and adding only **SalaryAmt** to the query grid.
12. Use the **Update** query type to add an Update row to the query.
13. Use an update criterion that multiplies the SalaryAmt field by: **`1.07`**

    This will produce a 7% increase to the numbers in the SalaryAmt field.
14. Run the query, choosing **Yes** to update 21 records.
15. Close the query, saving it as: **`Salary Updates`**
16. Close the SalaryGrades table.

Create a Delete Query

17. Create a new query using only the **EventDate** field from the Schedules table.
18. Use the criterion **`<1/1/2019`** in the EventDate field.
19. Use the **Delete** query type and then run the query, choosing **Yes** to delete six records.
20. Close the new query without saving it.
21. Open the **Schedules** table to verify all records with an event date prior to 1/1/2019 have been deleted and then close the database.

Project Grader

PROJECT GRADER: A7-P1

Taylor Games: Updating Inventory

Taylor Games has new inventory it would like to add to its database. It has noticed that the selling prices for some of the new items are above or below the desired profit margin and need to be updated. In this exercise, you will start by using an append query to merge data with new inventory into the inventory table. Then, you will create a calculated field and a complex parameter query with criteria to identify which items are below and above the desired margin and make a new table using the results. Last, you will make an update query and modify the sales price so all items are within the desired margin.

1. Download and open your Project Grader starting file.
   - *Using eLab:* Download **A7_P1_eStart** from the Assignments page. You *must* start with this file or your work cannot be automatically graded.
   - *Not using eLab:* Open **A7_P1_Start** from your **Access Chapter 7** folder.
2. Use these guidelines to create an Append query:
   - Create a new query adding all fields from the **New Inventory** table.
   - Append to the **Inventory** table in the current database.
   - Run the query to complete the append action, then save the query with the name: **`Inventory Append`**
3. Create a new query named **`Margin Parameter`** that uses all fields from the Inventory table.
4. In the **Margin Parameter** query, add a calculated field named **`Margin`** that calculates: **`Cost/Price`**
5. Add this parameter to the Margin Parameter query: **`Between`** [**`Enter Minimum Margin`**] **`And`** [**`Enter Maximum Margin`**]
6. In the Margin Parameter query, use the **Make Table** action to make a table in the current database with the Table Name: **`Items Within Margin`**
7. Run the query to create the Items Within Margin table using these minimum and maximum margin numbers:
   - Enter Minimum Margin: **`0.5`**
   - Enter Maximum Margin: **`0.75`**
8. Use these guidelines to create an Update query:
   - Add only the **Price** field from the Inventory table to the query grid.
   - Set the Update To criteria as: **`[cost]*1.5`**
   - Run the query and save it as: **`Margin Update`**
9. Save your database.
   - *Using eLab:* Save it to your **Access Chapter 7** folder as **`A7 _ P1 _ eSubmission`** and attach the file to your eLab assignment for grading.
   - *Not using eLab:* Save it to your **Access Chapter 7** folder as: **`A7 _ P1 _ Submission`**

PROJECT GRADER: A7-P2

WebVision: Create a Crosstab Query and Update Orders

WebVision has noticed some discrepancies in recent orders and customer records. It would also like to see the total sales for each sales rep for the customers they service. In this exercise, you will start by creating a Crosstab query. Then, you will create a Find Unmatched query to track down orders that do not contain any details and a Delete query to remove the incomplete orders. Last, you will create a Find Duplicates query to find duplicate customer records.

1. Download and open your Project Grader starting file.
   - *Using eLab:* Download **A7_P2_eStart** from the Assignments page. You *must* start with this file or your work cannot be automatically graded.
   - *Not using eLab:* Open **A7_P2_Start** from your **Access Chapter 7** folder.
2. Use these guidelines to create a Crosstab query using the **Crosstab Query Wizard**:
   - Contains fields from the **OrderDetails** query.
   - Use the **RepID** field for row headings.
   - Use the **Company Name** field for column headings.
   - Calculate each column and row intersection using the Sum function on the **Line Total** field.
   - Name the query: **`Rep Sales by Customer`**
3. Make these changes to the **Rep Sales by Customer** query:
   - Change the name of the Total of Line Total field to: **`Total`**
   - Set the Format field property to **Currency** for both the **Line Total** and **Total** fields.
4. Use these guidelines to create a Find Unmatched query:
   - The **Orders** table contains the records in the query results.
   - The **Order Details** table contains the related records.
   - Use the **OrderID** field in the Orders table and the **Order ID** field in the Order Details table as the matching fields.
   - Show all available fields in the query results.
   - Name the query: **`Orders Without Details`**
5. Use these guidelines to create a Delete query:
   - Base the query on the **Order Details** table.
   - Add **Order ID** to the query grid.
   - Set Criteria to delete records where the **Order ID** is greater than **5**.
   - Run the query and click **Yes** when prompted.

   If performed correctly, you will see a prompt notifying you that you are about to delete two rows.
   - Save the query with the name: **`Incomplete Order Delete`**
6. Use these guidelines to create a Find Duplicates query:
   - Search the **Customers** table for duplicate values.
   - Use **Billing Address**, **City**, **State/Province**, and **Postal Code** for fields that may contain duplicate information.
   - Add all remaining available fields to the Additional Query Fields list.
   - Name the query: **`Duplicate Customers`**

7. Save your database.
   - *Using eLab:* Save it to your **Access Chapter 7** folder as **`A7 _ P2 _ eSubmission`** and attach the file to your eLab assignment for grading.
   - *Not using eLab:* Save it to your **Access Chapter 7** folder as: **`A7 _ P2 _ Submission`**

Extend Your Skills

These exercises challenge you to think critically and apply your new skills in a real-world setting. You will be evaluated on your ability to follow directions, completeness, creativity, and the use of proper grammar and mechanics. Save files to your chapter folder. Submit assignments as directed.

A7-E1 That's the Way I See It

You would like to make several enhancements to the Blue Jean Landscaping database to ensure more accurate data in query results. Open **A7-E1-BJL** and save it as: **A7-E1-BJLRev**

Make a copy of the Service Invoices Query, naming it *Acre Rate Range*, and then modify it to add a parameter that prompts the user to enter the rate per acre. The query should return only records that match the acre rate entered by the user. The second update is to the StoreMerchandise table. Create a *New Merchandise* append query that appends all records from the NewMerchandise table to the StoreMerchandise table. Verify that your queries function properly.

A7-E2 Be Your Own Boss

Business has picked up at Blue Jean Landscaping! You're modifying the database to ensure efficiency and to cope with unexpected situations. Open **A7-E2-BJL** and save it as: **A7-E2-BJLRev**

Use the Store Inventory query as the basis for a new query named *Manufacturer Item Inventory* that prompts the user to enter a manufacturer and returns all records for that manufacturer. Create a *2019 Sales* query that creates a new table containing all 2019 sales records from the Merch Sales query. Finally, create a *New Customers* append query that appends all records from the NewCustomers table to the Customers table.

A7-E3 Demonstrate Proficiency

Stormy BBQ has asked you to refine the merchandising section of its database to reflect recent price changes and move older records. Open **A7-E3-SBQ** and save it as: **A7-E3-SBQRev**

Create a query to copy the 2018 merchandise sales records from the MerchSales table into a new table and then create a delete query to remove those records from the MerchSales query. Decide what names to assign to the new table and to either of the two queries you choose to save. Create a query that increases the list prices of all items in the Merchandise table by 5%. You decide whether to save the query after running it (and which name to use, if you save it).