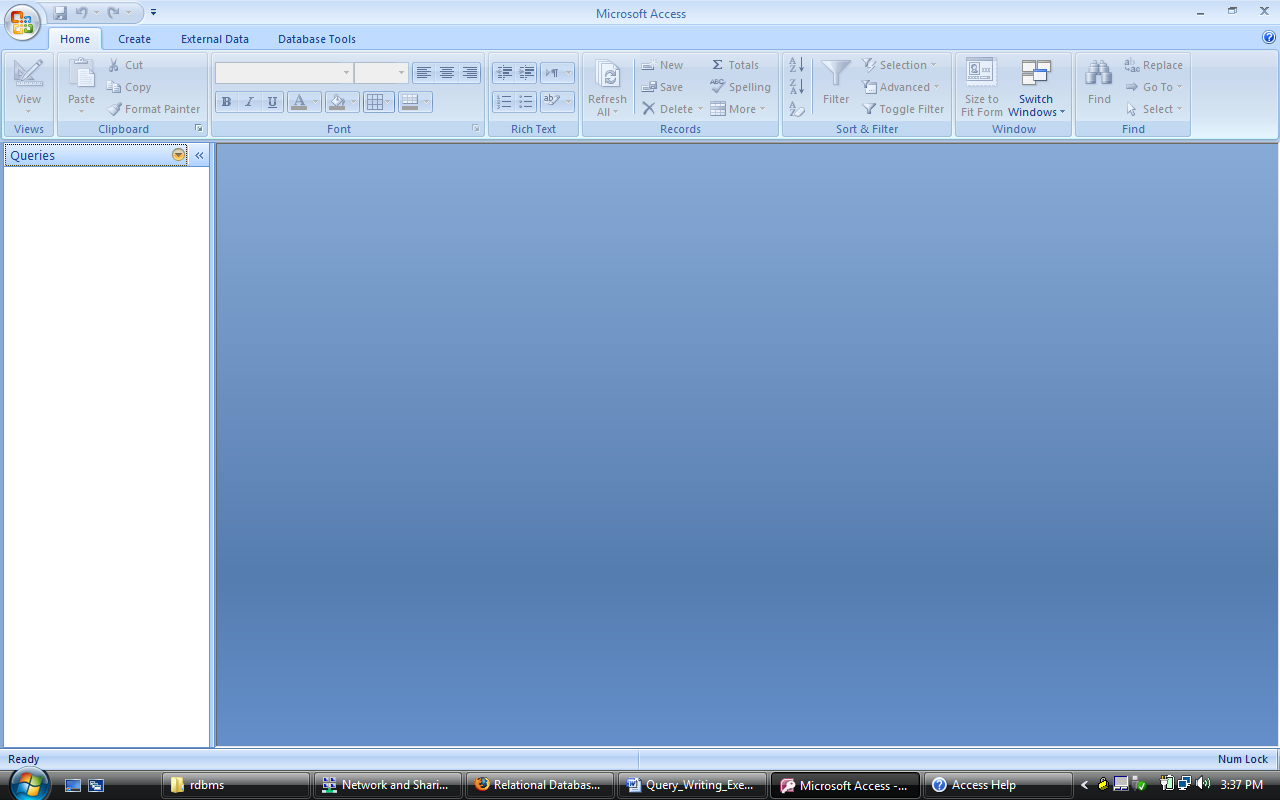
**Using the Query GUI in MS-Access**

This tutorial will guide you through the process of writing SQL queries using MS-Access (2007 version). This will be an important skill to have when it comes time to extract data from the MySQL database you’ll be using for your final project.

If you need to quit in the middle of this tutorial or want to save any of these queries for later use, select File > Save As and give the query an appropriate name. You’ll then see it in the list of Queries and can run it again later by double-clicking on it.

**Review the tables in the database:**

1. Download the baseball\_stats.mdb database from the lesson text (at the end of the Relational Database Concepts page).
2. Open the database in MS-Access.



One part of the Access interface that you’ll use frequently is the “Navigation Pane”, which is situated on the left side of the application window. The top of the Navigation Pane is just beneath the “Ribbon” (the strip of controls that runs horizontally along the top of the window).

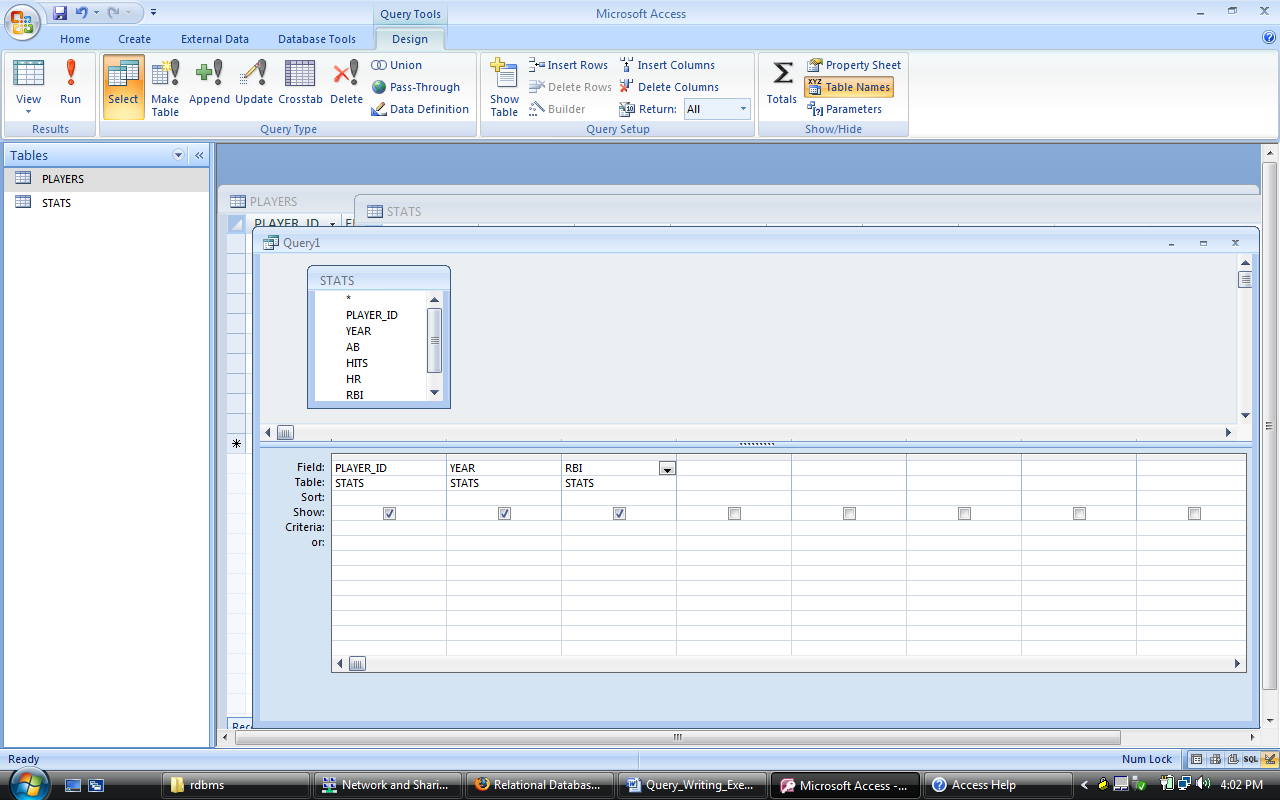
The Navigation Pane provides access to the objects stored in the database, such as tables, queries, forms and reports. When you first open the baseball\_stats.mdb database, the Navigation Pane should appear with the word Queries at the top, indicating that it is listing the queries stored in the database (at this point none).

1. Click on the strip at the top of the Navigation Pane (where it now says Queries) and select Tables to see a list of the database’s tables.
2. Double-click on a table’s name in the Navigation Pane to open it. Open both tables and note that the STATS table contains an ID for each player rather than his name. The names associated with the IDs are stored in the PLAYERS table.

**Write a simple Select query:**

With our first query, we’ll select all seasons in which the player had over 100 RBI.

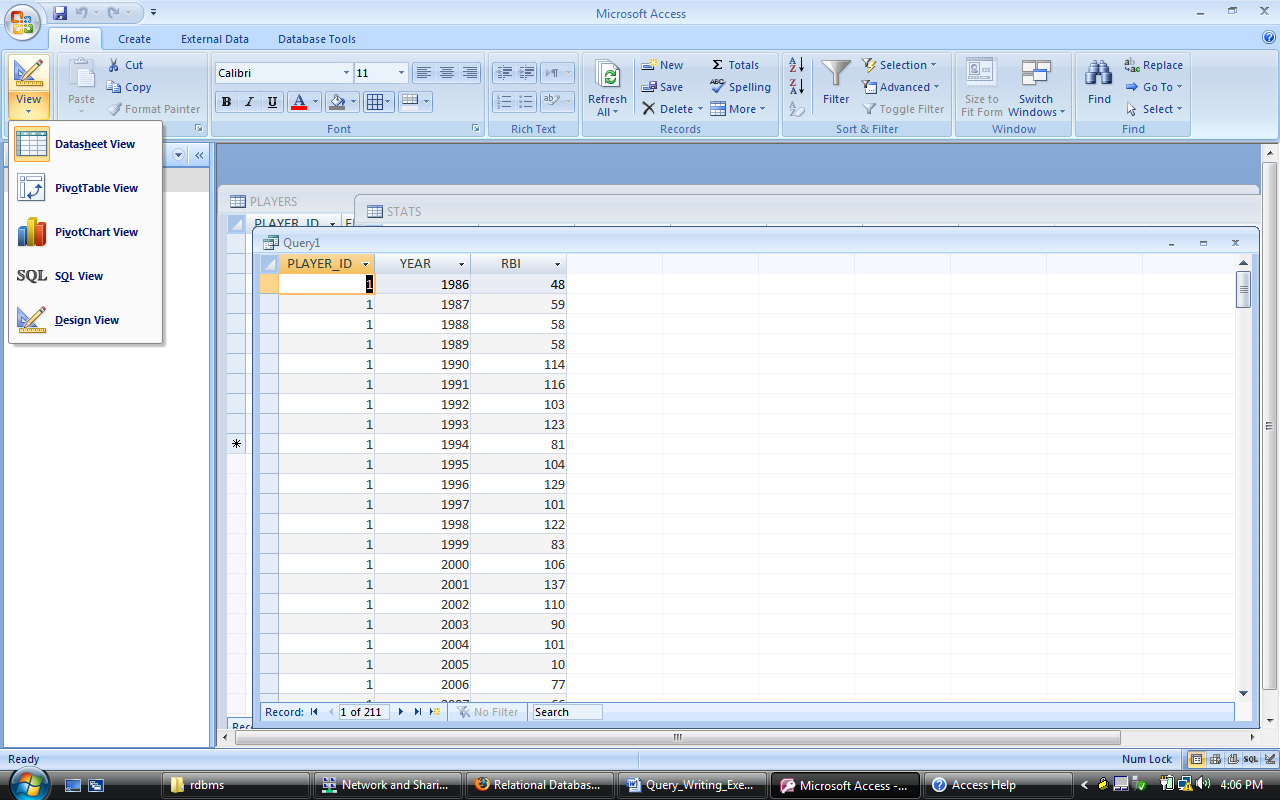
1. Click on the Create tab near the top of the application window.
2. Next click on the Query Design button (found on the right side of the Ribbon in the group of commands labeled as Other).
3. In the Show Table dialog, double-click on the STATS table to add it to the query and click Close.
4. Double-click on PLAYER\_ID in the list of fields in the STATS table to add that field to the design grid below.
5. Repeat this step to add the YEAR and RBI fields.



1. Click on the Run button (red exclamation point icon) on the left of the Ribbon to run the query. The result should contain the same number of records as the underlying STATS table (211), with the only difference being you’ve selected just the columns you’re interested in.

**View the SQL code generated by the GUI:**

1. At any time, you can view the SQL that’s created by your GUI settings by accessing the View drop-down list on the far-left side of Ribbon.



**Restrict the returned records to a desired subset:**

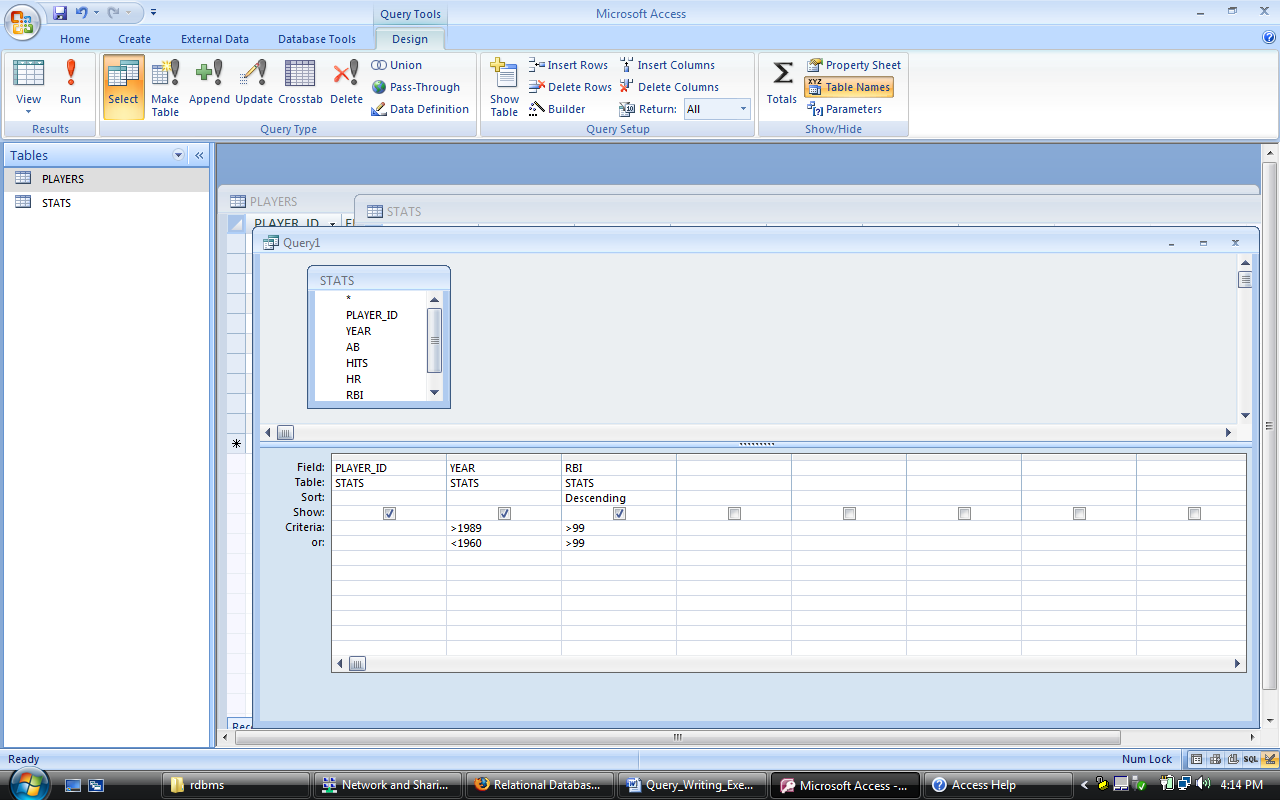
1. From the same View drop-down list, select Design View to return to the query design GUI.
2. In the design grid, set the Criteria value for the RBI field to >99.
3. Test the query by clicking on the red exclamation point again.

**Sort the returned records:**

1. Return to Design View.
2. In the design grid, click in the Sort cell under the RBI column and select Descending from the drop-down list. This will sort the records from highest RBI total to lowest.
3. Test the query.

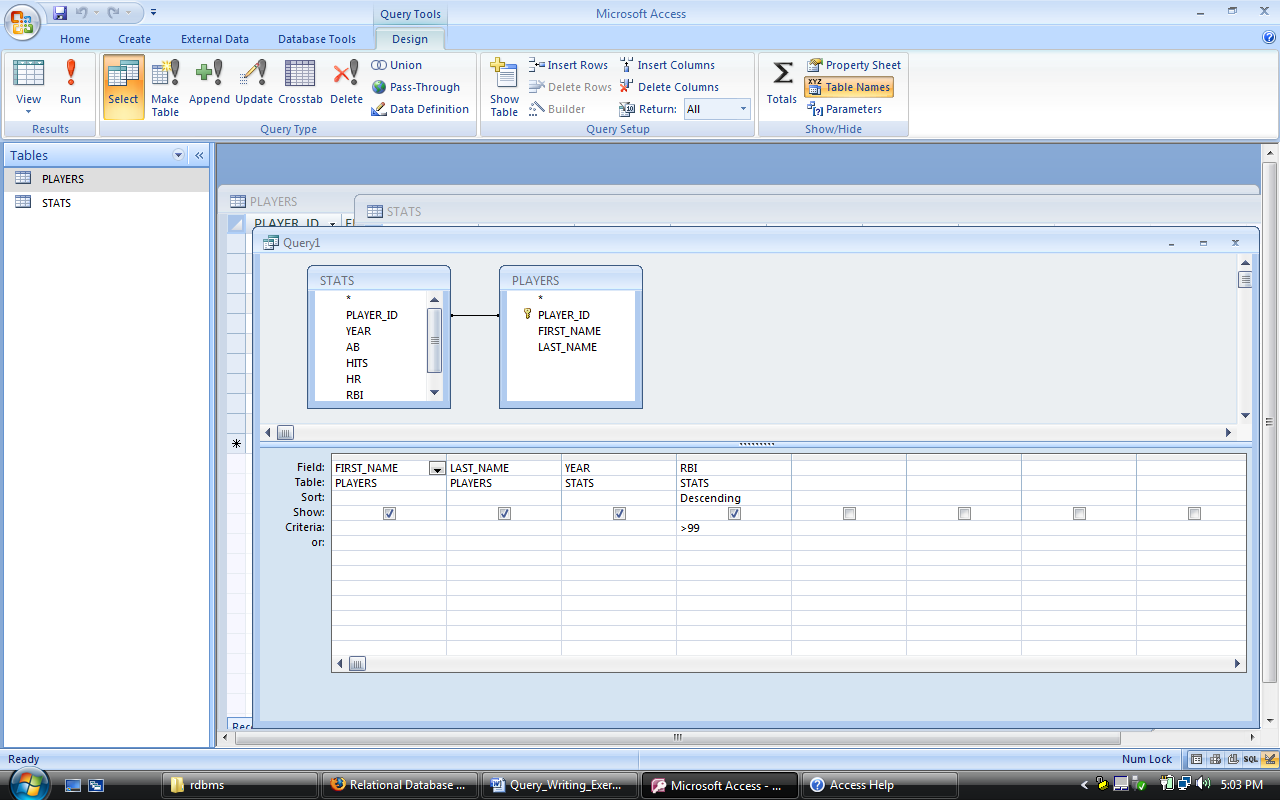
**Add additional criteria to the selection:**

1. Return to Design View and set the Criteria value for the YEAR field to >1989. This will limit the results to seasons of over 100 RBI since 1990.
2. Test the query.
3. Return to Design View and modify the Criteria value for the YEAR field to >1989 And <2000. This will further limit the results to just the 1990s.
4. Test the query.
5. Return to Design View and change the Criteria value for the YEAR field back to >1989. Beneath that cell (in the “or” cell), add <1960. As you should be able to guess, I’m asking you to write a query that identifies 100-RBI seasons since 1989 OR prior to 1960. However, the query as written at this point doesn’t quite yield that result. Instead, it would return 100-RBI seasons since 1989 and all seasons prior to 1960 (not just the 100-RBI ones). To produce the desired result, you need to repeat the >99 criterion in the RBI field’s or cell.



**Retrieve data from multiple tables using a join:**

1. Return to Design View and click on the Show Table button (near the middle of the Ribbon).
2. Double-click on the PLAYERS table to add it to the field list, then close the Show Table dialog. Note that the PLAYER\_ID fields in each table are joined by a connecting line. This happens because PLAYER\_ID is defined as the primary key in the PLAYERS table and Access assumes you’ll want to create joins based on that primary key. Note: If a join is not automatically created when you need one, you can create it by clicking on the key field in one of the tables and dragging to the key field in the other table.
3. Modify the query so that it again just selects 100-RBI seasons without any restrictions on the YEAR.
4. Double-click on the FIRST\_NAME and LAST\_NAME fields to add them to the query.
5. Test the query. Looking at the output, you may be thinking that the PLAYER\_ID value is not really of much use anymore and that the names would be more intuitive on the left side rather than the right.
6. To remove the PLAYER\_ID field from the output, return to Design View and click on the thin gray bar above the PLAYER\_ID field in the design grid to highlight that field. Press the Delete key on your keyboard to remove that field from the design grid (and the query results).
7. To move the name fields, click on the FIRST\_NAME field’s selection bar in the design grid and drag to the right so that you also select the LAST\_NAME field. With both fields selected, click and drag them to the left so that they now appear before the YEAR and RBI fields.



1. Test the query.

**Use the Expression Builder to concatenate values from multiple fields:**

1. Return to Design View and right-click on the Field cell of the FIRST\_NAME column. On the context menu that appears, click on Build to open the Expression Builder. You should see the FIRST\_NAME field listed in the expression box already. Note the various buttons that appear in a row just beneath the text box. These buttons can be used in conjunction with field names to produce expressions in either the Field or Criteria rows of the design grid. Here, we’re going to create an expression that will concatenate the FIRST\_NAME value with the LAST\_NAME value in one column.
2. First, delete the FIRST\_NAME text in the expression box.
3. In the list of database objects found in the bottom left of the dialog, double-click on Tables. The Tables entry should expand to display the PLAYERS and STATS tables.
4. Click on PLAYERS to display its fields in the middle box. Double-click on the FIRST\_NAME field to add it to the expression. (Note that the field is added to the expression in a special syntax that includes the name of its parent table. This is done in case there are other tables involved in the query that also have a FIRST\_NAME field. If there were, this syntax would ensure that the correct FIRST\_NAME field would be utilized.)
5. The concatenation character in SQL is the ampersand (&). Click on the & button to add this character to the expression.
6. Click in the expression text box after the & to obtain a cursor and type:

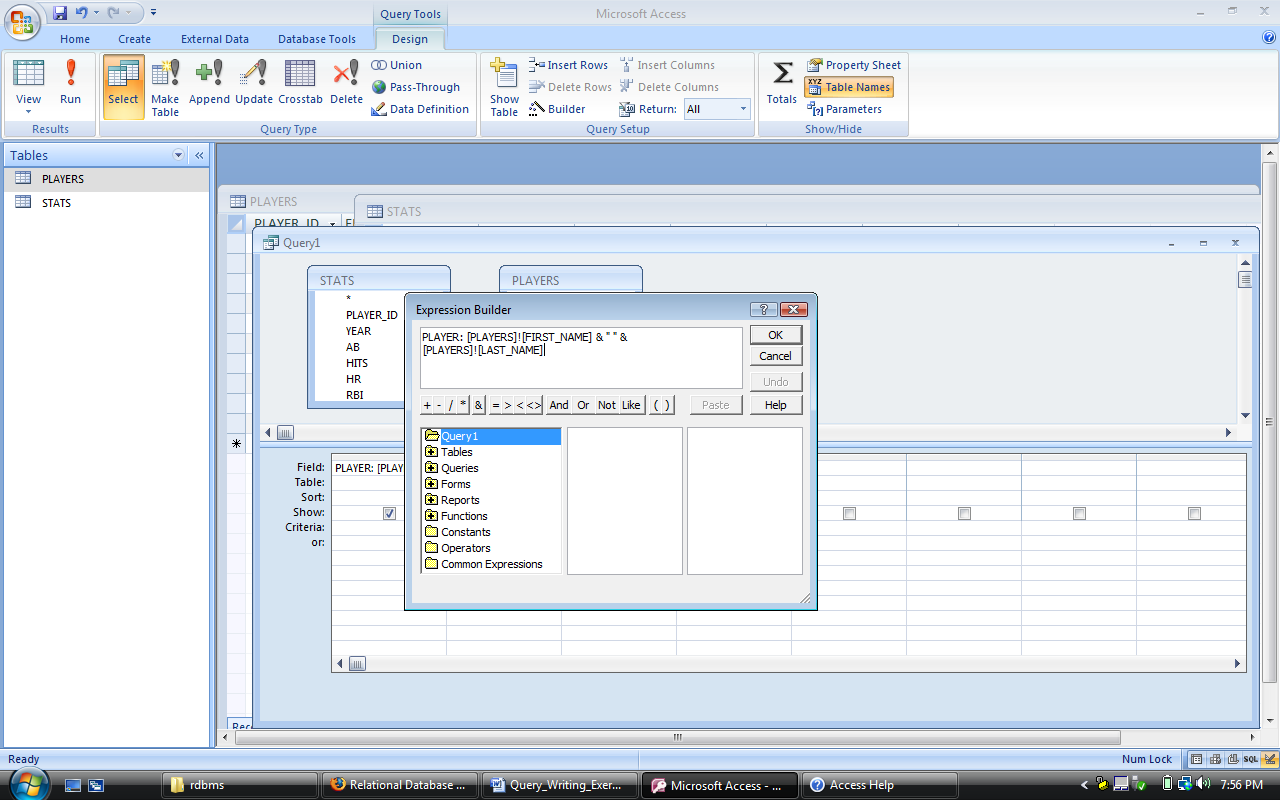
‘ ‘

The purpose of this step is to add a space between the FIRST\_NAME and LAST\_NAME values in the query results.

1. Click on the & button again.
2. Double-click on the LAST\_NAME field in the field list to add it to the expression.
3. Finally, place the cursor at the beginning of the expression and type the following:

PLAYER:

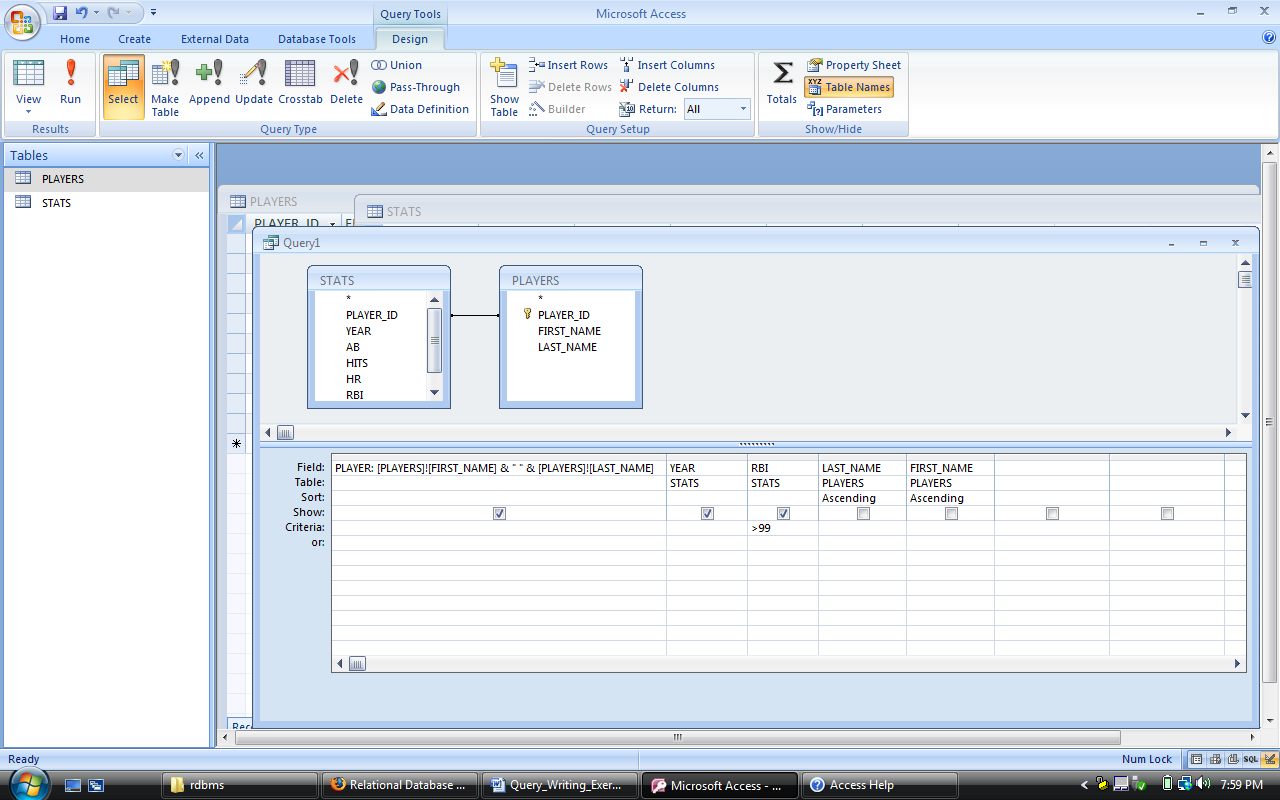
This specifies the column’s name in the output. Your expression should now look like this:



1. The LAST\_NAME column is now redundant, so remove it from the design grid as described above.
2. Run the query and confirm that the name values concatenated properly.

**Sorting by a field not included in the output:**

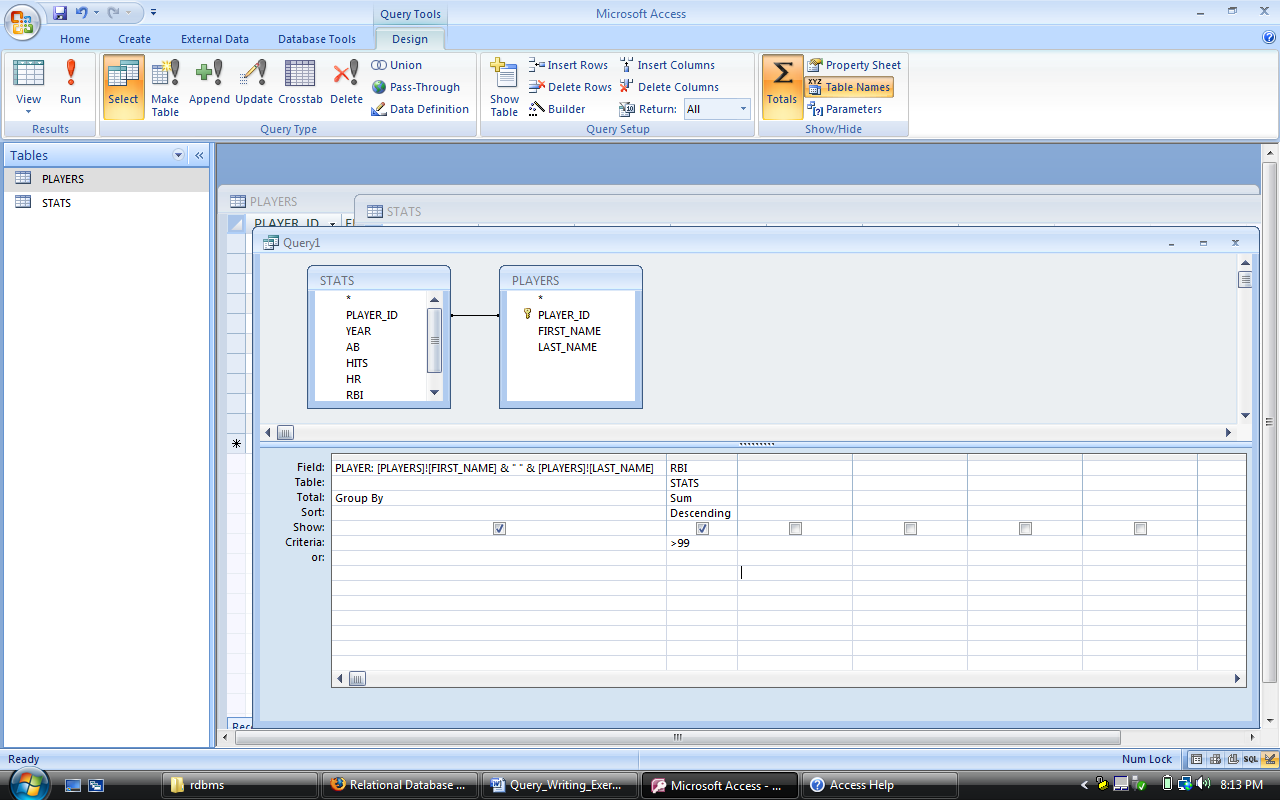
1. Suppose you wanted to sort this list by the last name, then first name, instead of by the number of RBI. Double-click on LAST\_NAME, then FIRST\_NAME to add them back to the design grid.
2. Delete the Descending setting from the RBI field’s Sort cell.
3. Choose the Ascending option in the Sort cell for both the LAST\_NAME and FIRST\_NAME fields.
4. Again, these fields are redundant with the addition of the PLAYER expression. Uncheck the Show checkbox for both fields so that they are not included in the query results. Note that the ordering of the Sort fields in the design grid is important. The results are sorted first based on the left-most Sort field, then by the next Sort field, then the next, etc.



1. Run the query and confirm that the records sorted properly.

**Adding a GROUP BY clause to calculate the sum of values in a field:**

1. We’re now going to calculate each player’s career RBI total. Return to Design View and remove the LAST\_NAME and FIRST\_NAME fields from the design grid.
2. Click on the Totals button on the right side of the Ribbon in the Show/Hide group of buttons. This will add a Total row to the design grid with each field assigned the value Group By.
3. Click in the Total cell under the RBI column and select Sum from the drop-down list. (Note the other options available on this list, especially Avg, Max and Min.)
4. Grouping by both the player’s name and the year won’t yield the desired results since the data in the STATS table are essentially already grouped in that way. We want the sum of the RBI values for each player across all years, so remove the YEAR column from the design grid.
5. Modify the query so that the results are sorted from highest career RBI total to lowest.



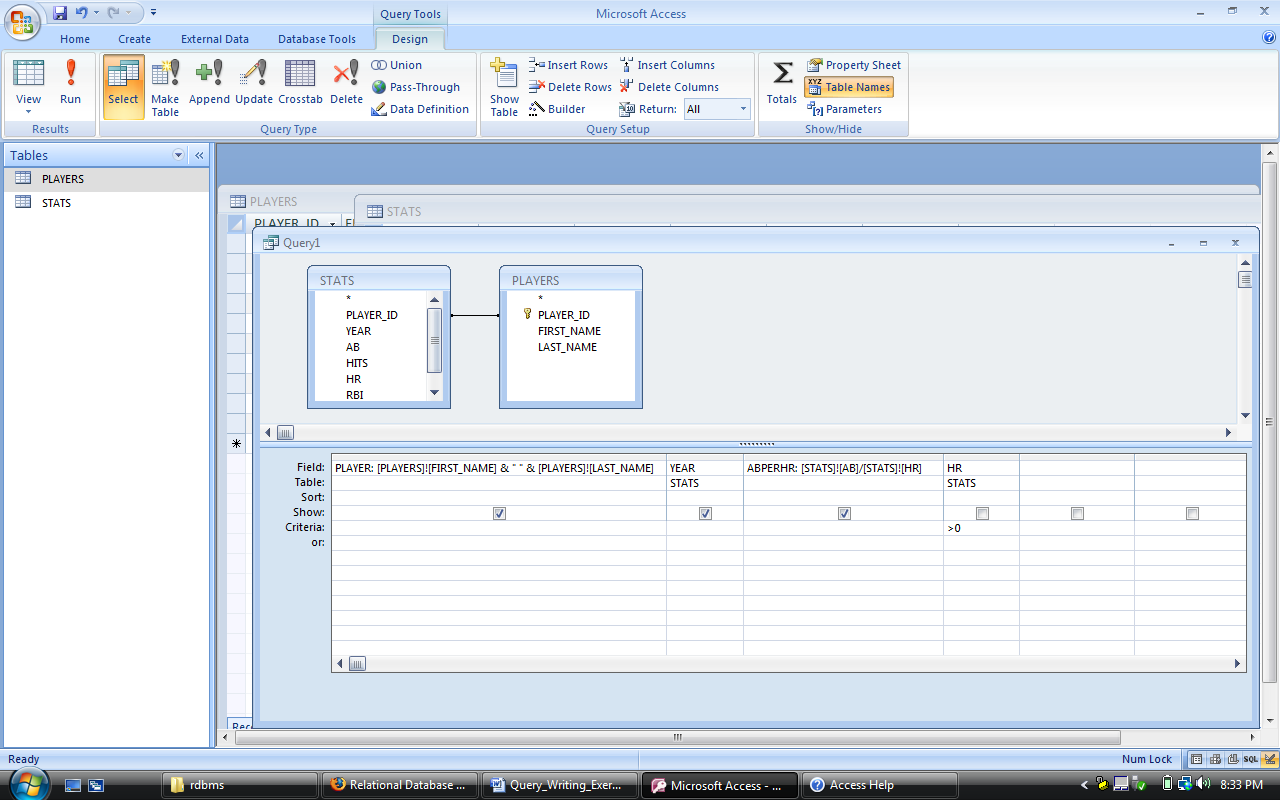
1. Run the query and note that the RBI column name defaults to ‘SumOfRBI’. You can override this by returning to Design View and entering a custom name (such as CareerRBI) and a colon before the RBI field name (following the PLAYER field example).

**Use the Expression Builder to perform a calculation based on multiple field values:**

1. Home run hitters are often compared based on the frequency of their home run hitting (i.e., ‘Player X averages a home every Y times at bat’). Let’s calculate this value for each player and season.
2. Click on the Totals button on the Query Design toolbar to toggle off the Total row in the design grid.
3. Remove the RBI field from the design grid.
4. Add the YEAR field to the design grid.
5. In the third column (which is empty), right-click in the Field cell and click on Build to pull up the Expression Builder as done earlier.
6. Double-click on Tables, then click on STATS to see a list of its fields.
7. Use the GUI to build the following expression:

ABPERHR: [STATS]![AB]/[STATS]![HR]

1. If a player had 0 HR in a season, this calculation will generate an error (can’t divide by 0), so add the HR field to the design grid, add >0 to its Criteria cell and uncheck its Show checkbox.
2. Sort the query results by the ABPERHR value from lowest to highest (since low values are better than high values).



1. Run the query and note that the calculated values include several digits after the decimal point. These digits aren’t really necessary, so return to Design View, right-click on the ABPERHR field and select Properties.
2. A Property Sheet pane will appear on the right side of the window. Select Fixed from the Format drop-down list and set the Decimal Places value to 1.
3. Re-run the query and confirm that you get the desired results.

**A few final notes:**

* A query is not just a snapshot of the data at the time you created it. If you save your query and the underlying data tables change, your query will reflect those changes the next time you run it.
* In this tutorial, you’ve dealt only with SELECT queries. However, SQL can also be used to add new records to tables and update or delete existing records. These other types of queries can be created by clicking on the buttons to the right of the Select button on the Ribbon.
* It isn’t necessary to save changes to the database itself. For example, if you had made changes to the tabular records themselves, those changes are automatically saved. The only changes you do need to save are to the design of the objects in the database (i.e., if you create a new table, query, form, or other database object).

**Practice Exercises:**

1. Write a query that displays all fields from the STATS table for players on the San Francisco Giants (SFG).
2. Write a query that outputs the batting average (HITS divided by AB) for each player and season. Format this value so that only 3 digits appear after the decimal point (e.g., .275).
3. Write a query that displays all fields from the STATS table along with the names of the players in the format “Ruth, Babe”.
4. Write a query that lists the highest 1-season RBI total each player had.
5. Write a query that calculates the number of years each player played for each of his teams.