# Table of Contents

## Introduction

Who Is the Teach Yourself SQL Book For? ................................................................. 2  
DBMSs Covered in This Book ........................................................................ 2  
Conventions Used in This Book ....................................................................... 3  

## 1 Understanding SQL

Database Basics ...................................................................................................... 5  
What Is SQL? .......................................................................................................... 10  
Try It Yourself ........................................................................................................ 11  
Summary .................................................................................................................. 12  

## 2 Retrieving Data

The SELECT Statement ............................................................................................ 13  
Retrieving Individual Columns ............................................................................. 14  
Retrieving Multiple Columns ............................................................................... 16  
Retrieving All Columns ......................................................................................... 18  
Retrieving Distinct Rows ...................................................................................... 19  
Limiting Results ..................................................................................................... 20  
Using Comments .................................................................................................... 23  
Summary .................................................................................................................. 25  

## 3 Sorting Retrieved Data

Sorting Data ............................................................................................................. 27  
Sorting by Multiple Columns ................................................................................ 29  
Sorting by Column Position ................................................................................. 30  
Specifying Sort Direction ...................................................................................... 31  
Summary .................................................................................................................. 34
## Filtering Data

Using the WHERE Clause .................................................. 35
The WHERE Clause Operators ........................................... 37
Summary ........................................................................... 42

## Advanced Data Filtering

Combining WHERE Clauses .................................................. 43
Using the IN Operator ....................................................... 47
Using the NOT Operator ..................................................... 49
Summary ........................................................................... 51

## Using Wildcard Filtering

Using the LIKE Operator ..................................................... 53
Tips for Using Wildcards ................................................... 60
Summary ........................................................................... 60

## Creating Calculated Fields

Understanding Calculated Fields ....................................... 61
Concatenating Fields .......................................................... 62
Performing Mathematical Calculations ............................... 68
Summary ........................................................................... 70

## Using Data Manipulation Functions

Understanding Functions .................................................... 71
Using Functions .................................................................. 73
Summary ........................................................................... 80

## Summarizing Data

Using Aggregate Functions ............................................... 81
Aggregates on Distinct Values .......................................... 89
Combining Aggregate Functions ....................................... 90
Summary ........................................................................... 91
C \textbf{SQL Statement Syntax} \hspace{1cm} 247
\begin{itemize}
  \item ALTER TABLE \hspace{1cm} 248
  \item COMMIT \hspace{1cm} 248
  \item CREATE INDEX \hspace{1cm} 248
  \item CREATE PROCEDURE \hspace{1cm} 249
  \item CREATE TABLE \hspace{1cm} 249
  \item CREATE VIEW \hspace{1cm} 249
  \item DELETE \hspace{1cm} 250
  \item DROP \hspace{1cm} 250
  \item INSERT \hspace{1cm} 250
  \item INSERT SELECT \hspace{1cm} 251
  \item ROLLBACK \hspace{1cm} 251
  \item SELECT \hspace{1cm} 252
  \item UPDATE \hspace{1cm} 252
\end{itemize}

D \textbf{Using SQL Datatypes} \hspace{1cm} 253
\begin{itemize}
  \item String Datatypes \hspace{1cm} 254
  \item Numeric Datatypes \hspace{1cm} 256
  \item Date and Time Datatypes \hspace{1cm} 257
  \item Binary Datatypes \hspace{1cm} 258
\end{itemize}

E \textbf{SQL Reserved Words} \hspace{1cm} 259

\textbf{Index} \hspace{1cm} 265
Ben Forta is Adobe Systems’ Director of Developer Relations and has more than 20 years of experience in the computer industry in product development, support, training, and product marketing. He is the author of the best-selling *Sams Teach Yourself SQL in 10 Minutes*, spinoff titles on MySQL and SQL Server T-SQL, *ColdFusion Web Application Construction Kit* and *Advanced ColdFusion Application Development* (both published by Adobe Press), *Sams Teach Yourself Regular Expressions in 10 Minutes*, as well as books on Flash, Java, Windows, and other subjects. He has extensive experience in database design and development, has implemented databases for several highly successful commercial software programs and websites, and is a frequent lecturer and columnist on Internet and database technologies. Ben lives in Oak Park, Michigan, with his wife Marcy and their seven children. Ben welcomes your e-mail at ben@forta.com and invites you to visit his website at http://forta.com/.
Acknowledgments

Thanks to the team at Sams for all these years of support, dedication, and encouragement. A special thank-you to Mark Taber for encouraging this long awaited update and for suggesting and facilitating the code coloring, which significantly enhances the readability and value of this new edition.

Thanks to my colleague Greg Wilson for his thorough technical review.

Thanks to the many hundreds of you who provided feedback on the first three editions of this book. Fortunately, most of it was positive, and all of it was appreciated. The enhancements and changes in this edition are a direct response to your feedback, which I continue to welcome.

Thanks to the dozens of colleges and universities who have made this book part of their IT and computer science curriculums. Being included and trusted by professors and teachers this way is immensely rewarding and equally humbling.

And finally, thanks to the more than one-quarter million of you who bought the previous editions of this book, making it not just my best-selling title, but also one of the best-selling books on the subject. Your continued support is the highest compliment an author can ever be paid.

—Ben Forta
We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we’re doing right, what we could do better, what areas you’d like to see us publish in, and any other words of wisdom you’re willing to pass our way.

You can e-mail or write directly to let us know what you did or didn’t like about this book—as well as what we can do to make our books stronger.

*Please note that we cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail we receive, we might not reply to every message.*

When you write, please be sure to include this book’s title and author as well as your name and contact information.

E-mail: feedback@sams-publishing.com

Mail: Reader Feedback
Sams Publishing/Pearson Education
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
This page intentionally left blank
Introduction

SQL is the most widely used database language. Whether you are an application developer, database administrator, web application designer, mobile app developer, or Microsoft Office user, a good working knowledge of SQL is an important part of interacting with databases.

This book was born out of necessity. I had been teaching Web application development for several years, and students were constantly asking for SQL book recommendations. There are lots of SQL books out there. Some are actually very good. But they all have one thing in common: for most users they teach just too much information. Instead of teaching SQL itself, most books teach everything from database design and normalization to relational database theory and administrative concerns. And while those are all important topics, they are not of interest to most of us who just need to learn SQL.

And so, not finding a single book that I felt comfortable recommending, I turned that classroom experience into the book you are holding. *Sams Teach Yourself SQL in 10 Minutes* will teach you SQL you need to know, starting with simple data retrieval and working on to more complex topics including the use of joins, subqueries, stored procedures, cursors, triggers, and table constraints. You’ll learn methodically, systematically, and simply—in lessons that will each take 10 minutes or less to complete.

Now in its fourth edition, this book has taught SQL to over a quarter million English speaking users, and has been translated into over a dozen other languages too so as to help users the globe over. And now it is your turn. So turn to Lesson 1, and get to work. You’ll be writing world class SQL in no time at all.
Who Is the Teach Yourself SQL Book For?

This book is for you if

- You are new to SQL.
- You want to quickly learn how to get the most out of SQL.
- You want to learn how to use SQL in your own application development.
- You want to be productive quickly and easily in SQL without having to call someone for help.

DBMSs Covered in This Book

For the most part, the SQL taught in this book will apply to any Database Management System (DBMS). However, as all SQL implementations are not created equal, the following DBMSs are explicitly covered (and specific instructions or notes are included where needed):

- Apache Open Office Base
- IBM DB2
- Microsoft Access
- Microsoft SQL Server (including Microsoft SQL Server Express)
- MariaDB
- MySQL
- Oracle (including Oracle Express)
- PostgreSQL
- SQLite

Example databases (or SQL scripts to create the example databases) are available for all of these DBMSs on the book webpage at http://forta.com/books/0672336073/.
Conventions Used in This Book

This book uses different typefaces to differentiate between code and regular English, and also to help you identify important concepts.

Text that you type and text that should appear on your screen is presented in monospace type.

It will look like this to mimic the way text looks on your screen.

Placeholders for variables and expressions appear in monospace italic font. You should replace the placeholder with the specific value it represents.

This arrow (➥) at the beginning of a line of code means that a single line of code is too long to fit on the printed page. Continue typing all the characters after the ➥ as though they were part of the preceding line.

**NOTE:**
A Note presents interesting pieces of information related to the surrounding discussion.

**TIP:**
A Tip offers advice or teaches an easier way to do something.

**CAUTION:**
A Caution advises you about potential problems and helps you steer clear of disaster.

**PLAIN ENGLISH**
New Term icons provide clear definitions of new, essential terms.
Input ▼
The Input icon identifies code that you can type in. It usually appears next to a listing.

Output ▼
The Output icon highlights the output produced by running a program. It usually appears after a listing.

Analysis ▼
The Analysis icon alerts you to the author’s line-by-line analysis of a program.
LESSON 2
Retrieving Data

In this lesson, you’ll learn how to use the SELECT statement to retrieve one or more columns of data from a table.

The SELECT Statement

As explained in Lesson 1, “Understanding SQL,” SQL statements are made up of plain English terms. These terms are called keywords, and every SQL statement is made up of one or more keywords. The SQL statement that you’ll probably use most frequently is the SELECT statement. Its purpose is to retrieve information from one or more tables.

Keyword

A reserved word that is part of the SQL language. Never name a table or column using a keyword. Appendix E, “SQL Reserved Words,” lists some of the more common reserved words.

To use SELECT to retrieve table data you must, at a minimum, specify two pieces of information—what you want to select, and from where you want to select it.
NOTE: Following Along with the Examples

The sample SQL statements (and sample output) throughout the lessons in this book use a set of data files that are described in Appendix A, “Sample Table Scripts.” If you’d like to follow along and try the examples yourself (I strongly recommend that you do so), refer to Appendix A which contains instructions on how to download or create these data files.

It is important to understand that SQL is a language, not an application. The way that you specify SQL statements and display statement output varies from one application to the next. To assist you in adapting the examples to your own environment, Appendix B, “Working in Popular Applications,” explains how to issue the statements taught throughout this book using many popular applications and development environments. And if you need an application with which to follow along, Appendix B has recommendations for you too.

Retrieving Individual Columns

We’ll start with a simple SQL SELECT statement, as follows:

Input ▼

```sql
SELECT prod_name
FROM Products;
```

Analysis ▼

The previous statement uses the SELECT statement to retrieve a single column called `prod_name` from the `Products` table. The desired column name is specified right after the SELECT keyword, and the FROM keyword specifies the name of the table from which to retrieve the data. The output from this statement is shown in the following:
Output ▼

prod_name
--------------------
Fish bean bag toy
Bird bean bag toy
Rabbit bean bag toy
8 inch teddy bear
12 inch teddy bear
18 inch teddy bear
Raggedy Ann
King doll
Queen doll

NOTE: Unsorted Data

If you tried this query yourself you might have discovered that the data was displayed in a different order than shown here. If this is the case, don’t worry—it is working exactly as it is supposed to. If query results are not explicitly sorted (we’ll get to that in the next lesson) then data will be returned in no order of any significance. It may be the order in which the data was added to the table, but it may not. As long as your query returned the same number of rows then it is working.

A simple SELECT statement similar to the one used above returns all the rows in a table. Data is not filtered (so as to retrieve a subset of the results), nor is it sorted. We’ll discuss these topics in the next few lessons.

TIP: Terminating Statements

Multiple SQL statements must be separated by semicolons (the ; character). Most DBMSs do not require that a semicolon be specified after single statements. But if your particular DBMS complains, you might have to add it there. Of course, you can always add a semicolon if you wish. It’ll do no harm, even if it is, in fact, not needed.
**TIP: SQL Statement and Case**

It is important to note that SQL statements are case-insensitive, so `SELECT` is the same as `select`, which is the same as `Select`. Many SQL developers find that using uppercase for all SQL keywords and lowercase for column and table names makes code easier to read and debug. However, be aware that while the SQL language is case-insensitive, the names of tables, columns, and values may not be (that depends on your DBMS and how it is configured).

**TIP: Use of White Space**

All extra white space within a SQL statement is ignored when that statement is processed. SQL statements can be specified on one long line or broken up over many lines. So, the following three statements are functionality identical:

```
SELECT prod_name
FROM Products;
```

```
SELECT prod_name FROM Products;
```

```
SELECT
prod_name
FROM
Products;
```

Most SQL developers find that breaking up statements over multiple lines makes them easier to read and debug.

---

**Retrieving Multiple Columns**

To retrieve multiple columns from a table, the same `SELECT` statement is used. The only difference is that multiple column names must be specified after the `SELECT` keyword, and each column must be separated by a comma.

**TIP: Take Care with Commas**

When selecting multiple columns be sure to specify a comma between each column name, but not after the last column name. Doing so will generate an error.
The following SELECT statement retrieves three columns from the products table:

**Input ▼**

```
SELECT prod_id, prod_name, prod_price
FROM Products;
```

**Analysis ▼**

Just as in the prior example, this statement uses the SELECT statement to retrieve data from the Products table. In this example, three column names are specified, each separated by a comma. The output from this statement is shown below:

**Output ▼**

<table>
<thead>
<tr>
<th>prod_id</th>
<th>prod_name</th>
<th>prod_price</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNBG01</td>
<td>Fish bean bag toy</td>
<td>3.4900</td>
</tr>
<tr>
<td>BNBG02</td>
<td>Bird bean bag toy</td>
<td>3.4900</td>
</tr>
<tr>
<td>BNBG03</td>
<td>Rabbit bean bag toy</td>
<td>3.4900</td>
</tr>
<tr>
<td>BR01</td>
<td>8 inch teddy bear</td>
<td>5.9900</td>
</tr>
<tr>
<td>BR02</td>
<td>12 inch teddy bear</td>
<td>8.9900</td>
</tr>
<tr>
<td>BR03</td>
<td>18 inch teddy bear</td>
<td>11.9900</td>
</tr>
<tr>
<td>RGAN01</td>
<td>Raggedy Ann</td>
<td>4.9900</td>
</tr>
<tr>
<td>RYL01</td>
<td>King doll</td>
<td>9.4900</td>
</tr>
<tr>
<td>RYL02</td>
<td>Queen dool</td>
<td>9.4900</td>
</tr>
</tbody>
</table>

**NOTE: Presentation of Data**

As you will notice in the above output, SQL statements typically return raw, unformatted data. Data formatting is a presentation issue, not a retrieval issue. Therefore, presentation (for example, displaying the above price values as currency amounts with the correct number of decimal places) is typically specified in the application that displays the data. Actual retrieved data (without application-provided formatting) is rarely used.
Retrieving All Columns

In addition to being able to specify desired columns (one or more, as seen above), SELECT statements can also request all columns without having to list them individually. This is done using the asterisk (*) wildcard character in lieu of actual column names, as follows:

**Input ▼**

```
SELECT *
FROM Products;
```

**Analysis ▼**

When a wildcard (*) is specified, all the columns in the table are returned. The column order will typically, but not always, be the physical order in which the columns appear in the table definition. However, SQL data is seldom displayed as is. (Usually, it is returned to an application that formats or presents the data as needed). As such, this should not pose a problem.

**CAUTION: Using Wildcards**

As a rule, you are better off not using the * wildcard unless you really do need every column in the table. Even though use of wildcards may save you the time and effort needed to list the desired columns explicitly, retrieving unnecessary columns usually slows down the performance of your retrieval and your application.

**TIP: Retrieving Unknown Columns**

There is one big advantage to using wildcards. As you do not explicitly specify column names (because the asterisk retrieves every column), it is possible to retrieve columns whose names are unknown.
Retrieving Distinct Rows

As you have seen, SELECT returns all matched rows. But what if you do not want every occurrence of every value? For example, suppose you want the vendor ID of all vendors with products in your products table:

**Input ▼**

```
SELECT vend_id
FROM Products;
```

**Output ▼**

```
vend_id
---------
BRS01
BRS01
BRS01
DLL01
DLL01
DLL01
DLL01
FNG01
FNG01
```

The SELECT statement returned 14 rows (even though there are only four vendors in that list) because there are 14 products listed in the products table. So how could you retrieve a list of distinct values?

The solution is to use the DISTINCT keyword which, as its name implies, instructs the database to only return distinct values.

**Input ▼**

```
SELECT DISTINCT vend_id
FROM Products;
```
Analysis ▼

SELECT DISTINCT vend_id tells the DBMS to only return distinct (unique) vend_id rows, and so only four rows are returned, as seen in the following output. If used, the DISTINCT keyword must be placed directly in front of the column names.

Output ▼

vend_id
----------
BRS01
DLL01
FNG01

CAUTION: Can't Be Partially DISTINCT

The DISTINCT keyword applies to all columns, not just the one it precedes. If you were to specify SELECT DISTINCT vend_id, prod_price, all rows would be retrieved unless both of the specified columns were distinct.

Limiting Results

SELECT statements return all matched rows, possibly every row in the specified table. What if you want to return just the first row or a set number of rows? This is doable, but unfortunately, this is one of those situations where all SQL implementations are not created equal.

In Microsoft SQL Server and Microsoft Access you can use the TOP keyword to limit the top number of entries, as seen here:

Input ▼

SELECT TOP 5 prod_name
FROM Products;
Output ▼

prod_name
-----------------
8 inch teddy bear
12 inch teddy bear
18 inch teddy bear
Fish bean bag toy
Bird bean bag toy

Analysis ▼

The previous statement uses the SELECT TOP 5 statement to retrieve just the first five rows.

If you are using DB2, well, then you get to use SQL unique to that DBMS, like this:

Input ▼

SELECT prod_name
FROM Products
FETCH FIRST 5 ROWS ONLY;

Analysis ▼

FETCH FIRST 5 ROWS ONLY does exactly what it suggests.

If you are using Oracle you need to count rows based on ROWNUM (a row number counter) like this:

Input ▼

SELECT prod_name
FROM Products
WHERE ROWNUM <= 5;

If you are using MySQL, MariaDB, PostgreSQL, or SQLite, you can use the LIMIT clause, as follows:


**Input ▼**

```
SELECT prod_name
FROM Products
LIMIT 5;
```

**Analysis ▼**

The previous statement uses the SELECT statement to retrieve a single column. LIMIT 5 instructs the supported DBMSs to return no more than five rows. The output from this statement is shown in the following code.

To get the next five rows, specify both where to start and the number of rows to retrieve, like this:

**Input ▼**

```
SELECT prod_name
FROM Products
LIMIT 5 OFFSET 5;
```

**Analysis ▼**

LIMIT 5 OFFSET 5 instructs supported DBMSs to return five rows starting from row 5. The first number is the number of rows to retrieve, and the second is where to start. The output from this statement is shown in the following code:

**Output ▼**

```
prod_name
------------
Rabbit bean bag toy
Raggedy Ann
King doll
Queen doll
```

So, LIMIT specifies the number of rows to return. LIMIT with an OFFSET specifies where to start from. In our example there are only nine products in the Products table, so LIMIT 5 OFFSET 5 returned just four rows (as there was no fifth).
**CAUTION: Row 0**
The first row retrieved is row 0, not row 1. As such, LIMIT 1 OFFSET 1 will retrieve the second row, not the first one.

**TIP: MySQL, MariaDB, and SQLite Shortcut**
MySQL, MariaDB, and SQLite support a shorthand version of LIMIT 4 OFFSET 3, enabling you to combine them as LIMIT 3,4. Using this syntax, the value before the , is the LIMIT and the value after the , is the OFFSET.

**NOTE: Not ALL SQL Is Created Equal**
I included this section on limiting results for one reason only, to demonstrate that while SQL is usually quite consistent across implementations, you can’t rely on it always being so. While very basic statements tend to be very portable, more complex ones tend to be less so. Keep that in mind as you search for SQL solutions to specific problems.

---

**Using Comments**

As you have seen, SQL statements are instructions that are processed by your DBMS. But what if you wanted to include text that you’d not want processed and executed? Why would you ever want to do this? Here are a few reasons:

- The SQL statements we’ve been using here are all very short and very simple. But, as your SQL statement grow (in length and complexity), you’ll want to include descriptive comments (for your own future reference or for whoever has to work on the project next). These comments need to be embedded in the SQL scripts, but they are obviously not intended for actual DBMS processing. (For an example of this, see the create.sql and populate.sql files used in Appendix B).
The same is true for headers at the top of SQL file, perhaps containing the programmer contact information and a description and notes. (This use case is also seen in the Appendix B .sql files.).

Another important use for comments is to temporarily stop SQL code from being executed. If you were working with a long SQL statement, and wanted to test just part of it, you could comment out some of the code so that DBMS saw it as comments and ignored it.

Most DBMSs supports several forms of comment syntax. We’ll Start with inline comments:

**Input ▼**

SELECT prod_name -- this is a comment
FROM Products;

**Analysis ▼**

Comments may be embedded inline using - - (two hyphens). Anything after the - - is considered comment text, making this a good option for describing columns in a CREATE TABLE statement, for example.

Here is another form of inline comment (although less commonly supported):

**Input ▼**

# This is a comment
SELECT prod_name
FROM Products;

**Analysis ▼**

A # at the start of a line makes the entire line a comment. You can see this format comment used in the accompanying create.sql and populate.sql scripts.
You can also create multi line comments, and comments that stop and start anywhere within the script:

**Input ▼**

```sql
/* SELECT prod_name, vend_id
FROM Products; */
SELECT prod_name
FROM Products;
```

**Analysis ▼**

/* starts a comments, and */ ends it. Anything between /* and */ is comment text. This type of comment is often used to comment out code, as seen in this example. Here, two SELECT statements are defined, but the first won’t execute because it has been commented out.

**Summary**

In this lesson, you learned how to use the SQL SELECT statement to retrieve a single table column, multiple table columns, and all table columns. You also learned how to return distinct values and how to comment your code. And unfortunately, you were also introduced to the fact that more complex SQL tends to be less portable SQL. Next you’ll learn how to sort the retrieved data.
This page intentionally left blank
Symbols and Numerics

[] (brackets) wildcard, 58-59
, (commas) characters, 16
|| (concatenation) operator, 63
@ character, 193
^ character, 59
@@ERROR variable, 204
@@IDENTITY global variable, 195
_ (underscore) wildcard, 57-58
% (percent sign) wildcard, 54
| (pipe) symbol, 247
ABS() function, 79
Access, using, 235
accessing cursor data, 208-210
adding rows to tables, 250
advanced data filtering, 43
advanced joins, creating, 125-128, 132-134
aggregate functions, 81
ALL argument, aggregate functions, 89
ALL clause, 95
Allaire ColdFusion, 233-234
ALTER TABLE statement, 169-170
CHECK constraints, 219
CONSTRAINT syntax, 215-217
syntax, 248
AND operator, 43-44
ANSI SQL, 11
applications
data filtering, 36
ODBC client, 245
portable code, 72
working in SQL, 233
Access, 235
Allaire ColdFusion, 233-234
DB2, 235
Microsoft ASP, 236
arguments
aggregate, 90
DISTINCT, 89
AS keyword, 66
126-129, 134
ASC keyword, 33
ASP (Microsoft), 236-237
authentication, 224
authorization, 224
AVG() function, 82
DISTINCT argument, 89
NULL values, 83
BETWEEN operator, 40
BINARY datatype, 258
BIT datatype, 256
brackets ([]) wildcard, 58-59

C

calculated fields, 61-63, 66
mathematical calculations, 68-70
subqueries, 108-109
views, 184
Cartesian Product, 117-119
cascading delete feature, 217
case-sensitivity sort orders, 33
SQL statements, 16
CFQUERY CFQUERY tag pair, 234
CHAR string datatype, 255
check constraints, 218
checking for no value, 40
range of values, 40
clauses, 28
ALL, 95
HAVING, 96
ordering, SELECT statements, 101
positioning, 42
WHERE combining, 43, 46
joins, 117
operators, 37
positioning, 37-39
clients, formatting, 62
CLOSE statements, closing cursors, 211
closing cursors, 211
code, commenting, 72, 195
codes, portable, 72
ColdFusion (Allaire), 233-234
columns, 7. See also fields
aliases, 66-68
assigning new values, 156
columns

breaking data correctly, 8
derived, 68
foreign keys, 216
GROUP BY clause, 95
Identity fields, 195
INSERT SELECT statements, 151
inserting omitting columns, 149
list (using INSERT statements), 148
naming, fully qualified, 117
NULL value, 40, 166
padded spaces, 64
primary keys, 9-10, 167
retrieving, 14
   all, 18
   multiple, 16
   unknown, 18
separating names in queries, 16
sorting data, 30
   by multiple columns, 29-30
   by non-selected columns, 29-31
descending on multiple columns, 33
specifying by relative position, 96
updating multiple, 156
values, deleting, 157
combined queries, 137
creating, 138-139
sorting results, 143
UNION statement rules, 140
combining aggregate functions, 90
WHERE clauses, 43, 46
commas (,), 16
commenting code, 23-24, 72, 195
COMMIT statements, 201, 248
commits, 199
comparing datatypes, 258
compatibility
datatype, 8
operator, 37
complex joins, views, 179
concatenation fields, 62
concatenation operators, 63
conditions (joins), 134
configuring ODBC data sources, 245
CONSTRAINT syntax,
ALTER TABLE statements, 217
constraints, 213-214
cautions, 214
check, 218
foreign keys, 216
speed, 223
syntax, 215
unique, 217
copying
data, table to table, 152
tables, 152-153, 164
COS() function, 79
COUNT() function, 82-85, 90, 133
COUNT* subquery, 108 CREATE INDEX statement, 221, 248-249
CREATE TABLE statement, 163
CONSTRAINT syntax, 215
DEFAULT keyword, 168
required information, 164
syntax, 249
CREATE VIEW statement, 179, 249
creating calculated fields,
61-63, 66
mathematical calculations, 68-70
subqueries, 108-109
ColdFusion pages, 234, 242
combined queries, 138-139
cursors, 207
groups, 94-95
indexes, 221, 248
joins, 116
   advanced, 125-126,
   132, 134
   self, 126-127
savepoints, 203
search patterns, 53
stored procedures, 188,
191-194, 249
tables, 163-165, 249
triggers, 223
views, 179
   reusable, 180
rules and restrictions, 177
uses, 177
CROSS JOIN, 119
CT statement results, 21
currency datatypes, 256
cursors, 205
accessing data, 208-210
closing, 211
deallocating resources, 211
limitations, 206
opening, 208
options, 206
using, 207
Web-based applications, 206
Customers table, 227

data
breaking correctly, 8
calculated fields, 61
Cartesian Product, 119
consistency with stored procedures, 189
copying, table to table, 152
deleting, 157, 160
filtering, 35
   advanced, 43
   application level, 36
checking against single values, 38
checking for a range of values, 40
checking for non-matches, 38-39
indexes, 221
views, 183
wildcards, 53
formatting, 17
DETECTE statement

cursors
   accessing data, 208-210
   closing, 211
   creating, 207
   opening, 208
   using, 207

dropping objects, 250

indexes
   cautions, 221
   creating, 221
   searching, 220

ODBC, 245

order entry systems, 226

security, 224

transaction
   processing, 197

datatypes, 8, 254

binary, 258

currency, 256

data and time, 257

defining, 216, 219

numeric, 256

reasons for use, 253

string, 254

date and time functions, 73, 76-78

DATE datatype, 257

DATEPART() function, 77

DATETIME datatype, 257

DB2, 235

DBMS (Database Management System), 6

accidental table deletion, 172

affecting sort order, 28

cascading delete feature, 217

constraints, 214

cursor options, 206

data and time

datatypes, 254

DB2, 235

functions, 71

indexes, 221

interactive tools, 115

joins, 121

LIKE operator, 54

security mechanisms, 224

separating statements, 15

specific operators, 41

SQL extensions, 11

stored procedures, 190

transaction processing, 200

triggers, 223

TRIM functions, 66

UNION keyword, 139

UNION statements, 143

UPDATE statements, 155

user-defined datatypes, 219

views
   creating, 176
   rules and restrictions, 178

DECIMAL datatype, 256

DECLARE statements creating, cursors, 207

stored procedures, 193

DEFAULT keyword, 168

defining datatypes, 216, 219

foreign keys, 216

ODBC Data Sources, 246

primary keys, 10, 215

DELETE FROM statement, 158

DELETE statement, 157

FROM keyword, 159

guidelines, 160

rollbacks, 201

security privileges, 158

syntax, 250

transaction processing, 199

triggers, 222

TRUNCATE TABLE statement, 160

WHERE clause, 158

How can we make this index more useful? Email us at indexes@samspublishing.com
deleting
cascading delete feature, 217
column values, 157
data, 157, 160
rows, 250
tables, 171
  preventing accidental deletion, 172
derived columns. See aliases
DESC keyword, 31
dictionary sort order, 33
displaying statement output, 14
DISTINCT argument
  AVG() function, 89
  COUNT() function, 90
downloading ready-to-use data files, 230
DROP statement, syntax, 250
DROP TABLE statement, 171
dropping database objects, 250
duplicate rows, eliminating, 141-142

E
eliminating duplicate rows, 141-142
EQUIJOIN, 120
establishing primary keys, 10
EXCEPT statements, 143
EXECUTE statements, 190-191
executing stored procedures, 190-191
EXP() function, 79
explicit commits, 201
extensions, 11

F
FETCH statement, 208, 210
fields, 62
  aliases, 66
    alternate uses, 67
    names, 68
calculated, 61-63, 66
  mathematical calculations, 68-70
  subqueries, 108-109
  views, 184
concatenating, 62
filtering
  by subquery, 104-105
data, 35
    advanced, 43
    application level, 36
    checking against single values, 38
    checking for a range of values, 40
    checking for non-matches, 38-39
  indexes, 221
  views, 183
  groups, 96
  wildcards, 53
fixed length strings, 254
FLOAT datatype, 256
foreign keys, 159, 216
benefits, 217
defining, 216
formatting
  clients, 62
  data, 17
  retrieved data with views, 180-183
  servers, 62
  statements, 165
  subqueries, 106
FROM clause, 116
FROM keyword
  DELETE statement, 159
  UPDATE statement, 157
full outer joins, 132
fully qualified column names, 117
functions, 71-72, 75
  aggregate, 81
    AVG(), 82
    combining, 90
    COUNT(), 84-85
    DISTINCT, 89
    MAX(), 85
    MIN(), 86-87
    SUM(), 87-88
data manipulation, 71
date and time, 73, 76-78
numeric, 73, 79
problems, 71
system, 73
text, 73
text manipulation, 74
  SOUNDEX() function, 75
  UPPER() function, 73

G
global variables,
  @@IDENTITY, 195
GRANT statements, 224
graphical interfaces, 115
GROUP BY clause. See also ORDER BY clause
  creating groups, 94-95
  relative position of columns, 96
  grouping
    data, 93, 99
    GROUP BY clause, 99
  ORDER BY clause, 100
  operators, 46
  groups
    creating, 94-95
    filtering, 96
    nested, 95

H-I
HAVING clauses, 96
Identity fields, 195
implementing transactions, 200
IN operator, 47
  advantages, 49
  combining with NOT operator, 51
  including duplicate rows, 141-142
  indexes, 220
  cautions, 221
  creating, 221, 248
  revisiting, 222
  searching, 220
  inner joins, 120, 129
  INSERT SELECT statement, 151
SELECT INTO statement comparison, 152
syntax, 251
INSERT statement
columns lists, 148
INTO keyword, 146
omitting columns, 149
partial rows, 149
rollbacks, 204
safety, 146
security privileges, 145
syntax, 250
table layout, 147
transaction
processing, 199
triggers, 222
VALUES, 148
inserting
columns, 149
data, 145-146
INTO keyword, 146
retrieved, 150
rows, 146
multiple, 152
partial, 149-150
INT datatype, 256
interactive DBMS
tools, 115
INTERSECT
statements, 143
INTO keyword, 146
IS NULL clause, 41
JOIN
joining tables, 113, 115
aliases, 125-128, 132
multiple, 121-123
natural joins, 129
performance concerns, 121
joins
aggregate functions, 132-133
Cartesian Product, 117
conditions, 134
creating, 116
advanced, 125-128, 132-134
self, 126-128
CROSS, 119
DBMS interactive
tools, 115
EQUIJOIN, 120
Inner, 120
natural, 129
outer, 129, 132
full, 132
types, 132
performance considerations, 121
pros, 115
self, instead of sub-queries, 128
types, 126
views, 179
WHERE clause, 117-119
K
keys
foreign, 216
primary, 9
keywords, 13
AND, 44
AS, 66, 126-129, 134
DEFAULT, 168
FROM, 157
IN, 49
INTO, 146
NOT, 50
OR, 45
REFERENCES, 217
UNION, 138
UNIQUE, 218
L
languages, SQL, 10,
259-263
LEFT keyword, 131
LIKE operator, 53-54
limiting SELECT statement results, 20-23
local variables, @
character, 193
LONG RAW datatype, 258
LTRIM() function, 66
M
managing transactions, 199
COMMIT
statements, 201
ROLLBACK
statements, 201
SAVEPOINT
statements, 202
manipulating tables, 163
complex structure changes, 170
deleting, 171
manipulation functions, 71
date and time, 73, 76-78
numeric, 79
text, 74
mathematical calculations, performing, 68-70
mathematical operators, 69
MAX() function, 82, 85-86, 90
Microsoft Access. See Access
Microsoft ASP, 236-237
Microsoft Query, 238
MIN() function, 82, 86
DISTINCT argument, 90
non-numeric data, 87
multiple tables, joining, 123
multiple rows, inserting, 152
MySQL, 240-241
N
names, aliases, 126
naming
aliases, 68, 91
columns, fully qualified, 117
indexes, 221
tables, renaming, 172
natural joins, 129
navigating tables, 205
NCHAR string
datatype, 255
nested groups, 95
NOT NULL values, 166
NOT operator, 49-51, 60

How can we make this index more useful? Email us at indexes@samspublishing.com
NULL values, 40
AVG() functions, 83
empty strings, 167
primary keys, 167
specifying, 167
tables, 166
numeric datatypes, 256
numeric functions, 73
numeric manipulation functions, 79
NVARCHAR string datatype, 255

obtaining sample tables and scripts, 229
ODBC
ASP, 236
data sources, 245-246
dates, 257
versions, 245
omitting columns, 149
OPEN CURSOR statements, 208
OPEN statements, opening cursors, 208, 211
opening cursors, 208, 211
operators, 38
AND, 43-44
BETWEEN, 40
compatibility, 37
concatenation, 63
DBMS specific, 41
grouping related, 46
HAVING clause, 96
IN, 47-49
LIKE, 53
mathematical, 69
NOT, 49-51
OR, 45
predicates, 54
WHERE clause, 37
OR operator, 45
Oracle
commits, 202
copying data between tables, 153
cursors
closing, 211
creating, 208
retrieving data, 208

date and time manipulation functions, 77
savepoints, 203
stored procedures, 191-194
triggers, 223
Oracle 8, 241
Oracle Express, 241
ORDER BY clause, 99
positioning, 29
SELECT statement, 28
UNION statements, 142
order entry systems, 226
order of evaluation parenthesis, 47
WHERE clauses, 46
ordering clauses, SELECT statements, 101
OrderItems table, 228-229
Orders table, 228
outer joins, 129
full, 132
syntax, 130
types, 132
overwriting tables, 165

padded spaces, 64
parenthesis, 46-47
partial rows, inserting, 149-150
percent sign (%), wildcard, 54
performance
indexes, 221
joins, 121
SQL experimentation, 123
subqueries, 108
UNION statements, 140
views, 177
PHP, 242
PI() function, 79
pipe (|) symbol, 247
placeholders. See savepoints
portable code, 72
positioning
sorting data by column position, 30
WHERE clause, 37-39
PostgreSQL, 243
predicates, 54
primary keys, 9-10
Customer table, 227
defining, 10, 215
NULL values, 167
OrderItems table, 229
Orders table, 228
Products table, 227
unique constraints, 217
Vendors table, 226
processing
stored procedures, 189
subqueries, 106
transactions, 198
transactions. See transaction processing
Products table, 226-227

QMF (Qery Management Facility) utility, 235
queries, 103
aggregate functions, 81
combined, 137
creating, 138-139
sorting results, 143
UNION statement rules, 140
WHERE clauses, 137
combining, 105
data formatting, 17
filtering results, 35
internal query optimizer, 140
multiple WHERE clauses, 140
result sets, 205
subqueries, 108
table aliases, 126
unsorted data results, 15
views, 176
wild cards (*), 18
quotes
numeric values, 256
string values, 255
quotes (**), 39

RAW datatype, 258
ready-to-use data files, downloading, 230
REAL datatype, 256
REFERENCES
keyword, 217
referential integrity, 213, 217
reformatting retrieved data with views, 180-183
relational databases
referential integrity, 213
sort order, 28
relational tables, 113-114
relative position, columns, 96
RENAME statement, 172
renaming tables, 172
replacing tables, 165
reserved words, 259-263
restrictions, views, 177
result sets, 205
retrieving columns, unknown, 18
data, 13, 81, 84-85
all columns, 18
FETCH statements, 208-210
individual columns, 14
inserting, 150
multiple columns, 16
reusable views, creating, 180
revisiting indexes, 222
REVOKE statements, 224
RIGHT keyword, 131
ROLLBACK statement, 201, 251
rollbacks, 201
savepoints, 203
statements, 204
using, 199
rows, 9
adding to tables, 250
cursors, 205
default values, 168
deleting, 250
duplicate, 141-142
filtering, 96
inserting, 146
check constraints, 219
multiple, 152
partial, 149-150
joins, 129
returning with UNION statements, 141
updating, 252
RTRIM() function, 64-65
rules
constraints, 214
views, 177
samples
scripts, obtaining, 229
tables, 226, 229
SAVEPOINT statements, 202
savepoints, 199, 203-204
scalability, 115
scale, 115
schematas, 7
scripts
ASP.NET, 237
comments, 23-24
downloading, 230
PHP, 242
samples, obtaining, 229
search patterns, 53-55
searching indexes, 220
wildcards, 53
% character, 54
[ ] characters, 58-59
^ character, 59
_ character, 57-58
security
data, 224
DELETE statement, 158
INSERT statements, 145
UPDATE statement, 155
SELECT * FROM statements, 179
SELECT INTO statements, 152
SELECT statement, 13
AS keyword, 66
AVG() function, 82
clauses, ordering, 101
concatenating columns, 63-64
creating groups, 94-95
DISTINCT keyword, 19-20
FROM clause, 116
GROUP BY clause, 94
inner joins, 130, 133
IS NULL clause, 41
joins, 115-116
ORDER BY clause, 28
DESC keyword, 31
positioning, 29
results, limiting, 20-23
retrieving individual columns, 14
subqueries, 104-106, 110
syntax, 252
UNION keyword, 138-140
WHERE clause, 35
combined queries, 137
combining, 44
IN operator, 47
NOT operator, 49-51
OR operators, 45
quotes, 39
self-joins, creating, 126-128
semicolons (;), 15
separating statements, 15
sequence (clauses), 101
servers, formatting, 62
SET command, 156
simplifying joins with views, 179
SIN() function, 79
Single Column Only subqueries, 107
SMALLDATETIME datatype, 257
SMALLINT datatype, 256
sorting
by non-selected columns, 29
case-sensitivity issues, 33
combined query results, 143
data, 27, 99
by column position, 30
descending on multiple columns, 33

How can we make this index more useful? Email us at indexes@samspublishing.com
multiple columns, 29-30
non-selected columns, 31
ORDER BY clause, 100
retrieved, 27
specifying direction, 31
datatype functionality, 253, 262
SOUNDEX() function, 75
spaces, padded, 64
specifying dates, 257
default values, 168
NULL values, 167
sort direction, 31
statements, 14
speed
constraints versus triggers, 223
deleting data, 160
SQL, 10
advanced features, 213
advantages, 11
ALL clause, 95
column aliases, 66
commits, 201
cursors
closing, 211
creating, 207
retrieving data, 210
DATEPART() function, 77
deleting/updating data, 161
experimentation with operations, 123
extensions, 11
Identity fields, 195
INNER JOIN syntax, 120
INSERT statements, 146
keywords, 13
local variables, @ character, 193
reserved words, 259-263
savepoints, 203
statements, clauses, 28
stored procedures, 193
transaction processing, 200
triggers, 223
working in popular applications, 233
SQL Server Management
Studio, 239
SQRT() function, 79
statements
ALTER TABLE, 169-170, 248
case, 16
CFQUERY/CFQUERY
tag pairs, 234
clauses, 28
COMMIT, 248
CREATE INDEX, 221,
248-249
CREATE TABLE
required information, 164
syntax, 249
CREATE VIEW, 179, 249
DELETE, 157, 160
FROM keyword, 159
syntax, 250
displaying output, 14
DROP, 250
DROP TABLE, 171
formatting, 165
GRANT, 224
grouping related operators, 46
INSERT
omitting columns, 149
safety, 146
security privileges, 145
syntax, 250
VALUES, 148
INSERT SELECT,
syntax, 251
OPEN CURSOR, 208
RENAME, 172
REVOKE, 224
ROLLBACK,
syntax, 251
rollbacks, 199, 204
SELECT, 13
DISTINCT keyword, 19-20
results, limiting, 20-23
syntax, 252
specifying, 14
stored procedures. See stored procedures syntax, 247
terminating, 15
UNION, 138-140
UPDATE, 155-156,
160, 252
white space, 16
writing, 225
stored procedures, 187
benefits, 189
commenting code, 195
creating, 188,
191-194, 249
executing, 190-191
functionality, 194
Identity fields, 195
justification, 188
Oracle, 191-194
syntax, 189
triggers, 222
storing
datatype functionality, 253
date and time values, 257
numeric values, 255
strings, 254
string datatypes, 254
strings
fixed length, 254
quotes, 255
search, wildcards, 54
variable-length, 254
subqueries, 103
calculated fields, 108-109
COUNT*, 108
filtering by, 104-105
formatting, 106
joins, 127
performance, 108
processing, 106
SELECT statement, 104
self joins instead, 128
Single Column Only, 107
UPDATE statement, 157
WHERE clauses, 107
SUM() function, 82, 87-88
trimming padded spaces

How can we make this index more useful? Email us at indexes@samspublishing.com
troubleshooting accidental table deletion, 172
TRUNCATE TABLE statement, 160

underscore (_)
wildcard, 57-58
UNION ALL statements, 142
UNION statements combined queries, creating, 138-139
duplicate row handling, 141
limits, 140
ORDER BY clause, 142
types, 143
UNION, 140
unions. See combined queries
unique constraints, 217
UNIQUE keyword, 218
unsorted data, 15
UPDATE statement
cautions, 155
FROM keyword, 157
guidelines, 160
security privileges, 155
SET command, 156
subqueries, 157
syntax, 252
table names, 156
transaction processing, 199
triggers, 222
WHERE clause, 156
updating data, 155-156, 160
multiple columns, 156
tables, 169-170
UPPER() function, 73
user-defined datatypes, 219

V
values
columns deleting, 157
concatenation, 63
default, 168
NULL, 167
searching for (indexes), 220
trimming padded space, 65
VARBINARY
datatype, 258
variable-length strings, 254
Vendors table, 226
views
calculated fields, 184
creating, 177-180, 249
DBMS consistency, 176
filtering unwanted data, 183
joins, 179
performance concerns, 177
reformatting retrieved data, 180-183
rules and restrictions, 177
SELECT statement, 175
uses, 176
virtual tables, 175-176

W-X-Y-Z
web-based applications, cursors, 206
WHERE clause, 35
BETWEEN operator, 40
combining, 43
order of evaluation, 46
with queries, 137
DELETE statements, 158
filtering, 97
IN operator, 47
joins, 117-119
NOT operators, 49, 51
operators, 37
OR operators, 45
parenthesis, 47
positioning, 37, 39
quotes, 39
SOUNDEX() function, 76
subqueries, 107
UPDATE statements, 155-156
wildcards, 53
white space, SQL statements, 16
whitespace, 16
wildcards, 18, 53
[] (brackets) characters, 58-59
_ (underscore) character, 57-58
^ character, 59
natural joins, 129
positioning in search patterns, 55
search patterns, 55
wrappers, ODBC, 245
writing SQL statements, 225
stored procedures, 189