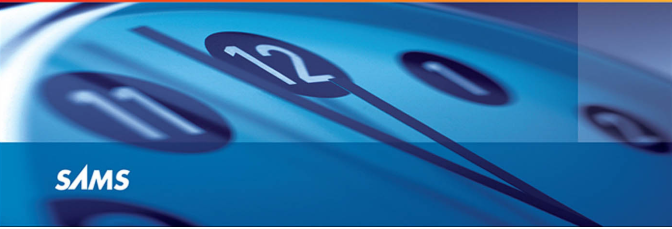
Ryan Stephens
Arie D. Jones
Ron Plew

SIXTH EDITION

Includes Coverage of **Oracle and Microsoft SQL Implementations**

Sams Teach Yourself SQL in 24 Hours



SAMS

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Ryan Stephens
Arie D. Jones
Ron Plew

Sams **Teach Yourself**

SQL

in 24 Hours

SIXTH EDITION

SAMS 800 East 96th Street, Indianapolis, Indiana, 46240 USA

Sams Teach Yourself SQL in 24 Hours, Sixth Edition

Copyright © 2016 by Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearsoned.com/permissions/. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

Microsoft and Microsoft SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

ISBN-13: 978-0-672-33759-8
ISBN-10: 0-672-33759-2

Library of Congress Control Number: 2015915011

Printed in the United States of America

First Printing December 2015

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

Editor-in-Chief
Mark Taub

Executive Editor
Laura Lewin

Development Editor
Michael Thurston

Managing Editor
Kristy Hart

Senior Project Editor
Lori Lyons

Copy Editor
Apostrophe Editing Services

Indexer
Lisa Stumpf

Proofreader
Debbie Williams

Technical Editors
Marshall Pyle

Jacinda Simmerman

Editorial Assistant
Olivia Basegio

Cover Designer
Mark Shirar

Compositor
Nonie Ratcliff

Contents at a Glance

Part I: An SQL Concepts Overview

HOUR 1 Welcome to the World of SQL 1

Part II: Building Your Database

HOUR 2 Defining Data Structures 19
3 Managing Database Objects 33
4 The Normalization Process 55
5 Manipulating Data 67
6 Managing Database Transactions 81

Part III: Getting Effective Results from Queries

HOUR 7 Introduction to Database Queries 93
8 Using Operators to Categorize Data 109
9 Summarizing Data Results from a Query 133
10 Sorting and Grouping Data 145
11 Restructuring the Appearance of Data 159
12 Understanding Dates and Times 181

Part IV: Building Sophisticated Database Queries

HOUR 13 Joining Tables in Queries 197
14 Using Subqueries to Define Unknown Data 213
15 Combining Multiple Queries into One 229

Part V: SQL Performance Tuning

HOUR 16 Using Indexes to Improve Performance 243
17 Improving Database Performance 253

Part VI: Using SQL to Manage Users and Security

HOUR 18 Managing Database Users 269
19 Managing Database Security 283

Part VII: Summarized Data Structures

HOUR 20 Creating and Using Views and Synonyms 297
21 Working with the System Catalog 313

Part VIII: Applying SQL Fundamentals in Today's World

HOUR 22 Advanced SQL Topics 323
23 Extending SQL to the Enterprise, the Internet, and the Intranet 337
24 Extensions to Standard SQL 347

Part IX: Appendixes

APPENDIX A Common SQL Commands 357
B Installing Oracle and Microsoft SQL 363
C Answers to Quizzes and Exercises 369
D Bonus Exercises 411
E Glossary 425
Index 429

Table of Contents

Part I: An SQL Concepts Overview

HOUR 1: Welcome to the World of SQL 1

SQL Definition and History 1

SQL Sessions 8

Types of SQL Commands 8

Canary Airlines: The Database Used in This Book 11

Summary 15

Q&A 16

Workshop 16

Part II: Building Your Database

HOUR 2: Defining Data Structures 19

What Is Data? 19

Basic Data Types 20

Summary 28

Q&A 29

Workshop 29

HOUR 3: Managing Database Objects 33

Database Objects and Schema 33

Tables: The Primary Storage for Data 35

Integrity Constraints 44

Summary 49

Q&A 50

Workshop 50

HOUR 4: The Normalization Process 55
Normalizing a Database 55
Denormalizing a Database 63
Summary 64
Q&A 64
Workshop 64

HOUR 5: Manipulating Data 67
Overview of Data Manipulation 67
Populating Tables with New Data 68
Updating Existing Data 73
Deleting Data from Tables 75
Summary 76
Q&A 77
Workshop 77

HOUR 6: Managing Database Transactions 81
What Is a Transaction? 81
Controlling Transactions 82
Poor Transactional Control 90
Summary 90
Q&A 90
Workshop 91

Part III: Getting Effective Results from Queries

HOUR 7: Introduction to Database Queries 93
The SELECT Statement 93
Case-Sensitivity 101
Fundamentals of Query Writing 102
Summary 106
Q&A 106
Workshop 107

HOUR 8: Using Operators to Categorize Data **109**
What Is an Operator in SQL? 109
Comparison Operators 110
Logical Operators 113
Conjunctive Operators 120
Negative Operators 123
Arithmetic Operators 128
Summary 130
Q&A 131
Workshop 131

HOUR 9: Summarizing Data Results from a Query **133**
Aggregate Functions 133
Summary 141
Q&A 142
Workshop 142

HOUR 10: Sorting and Grouping Data **145**
Why Group Data? 145
The GROUP BY Clause 146
GROUP BY Versus ORDER BY 150
CUBE and ROLLUP Expressions 153
The HAVING Clause 155
Summary 156
Q&A 157
Workshop 157

HOUR 11: Restructuring the Appearance of Data **159**
ANSI Character Functions 159
Common Character Functions 160
Miscellaneous Character Functions 169
Mathematical Functions 172
Conversion Functions 173
Combining Character Functions 176

Summary . . . 177
Q&A . . . 177
Workshop . . . 178

HOUR 12: Understanding Dates and Times . . . 181
How Is a Date Stored? . . . 181
Date Functions . . . 183
Date Conversions . . . 188
Summary . . . 193
Q&A . . . 194
Workshop . . . 194

Part IV: Building Sophisticated Database Queries

HOUR 13: Joining Tables in Queries . . . 197
Selecting Data from Multiple Tables . . . 197
Understanding Joins . . . 198
Join Considerations . . . 207
Summary . . . 210
Q&A . . . 210
Workshop . . . 211

HOUR 14: Using Subqueries to Define Unknown Data . . . 213
What Is a Subquery? . . . 213
Embedded Subqueries . . . 219
Correlated Subqueries . . . 223
Subquery Performance . . . 225
Summary . . . 225
Q&A . . . 226
Workshop . . . 226

HOUR 15: Combining Multiple Queries into One . . . 229
Single Queries Versus Compound Queries . . . 229
Compound Query Operators . . . 230
Using ORDER BY with a Compound Query . . . 235
Using GROUP BY with a Compound Query . . . 237

Retrieving Accurate Data . . . 238
Summary . . . 239
Q&A . . . 239
Workshop . . . 240

Part V: SQL Performance Tuning

HOUR 16: Using Indexes to Improve Performance . . . 243
What Is an Index? . . . 243
How Do Indexes Work? . . . 244
The CREATE INDEX Command . . . 245
Types of Indexes . . . 245
When Should Indexes Be Considered? . . . 248
When Should Indexes Be Avoided? . . . 248
Altering an Index . . . 250
Dropping an Index . . . 250
Summary . . . 251
Q&A . . . 251
Workshop . . . 252

HOUR 17: Improving Database Performance . . . 253
What Is SQL Statement Tuning? . . . 253
Database Tuning Versus SQL Statement Tuning . . . 254
Formatting Your SQL Statement . . . 254
Full Table Scans . . . 260
Other Performance Considerations . . . 261
Cost-Based Optimization . . . 264
Summary . . . 266
Q&A . . . 266
Workshop . . . 266

Part VI: Using SQL to Manage Users and Security

HOUR 18: Managing Database Users . . . 269
User Management in the Database . . . 269
The Management Process . . . 272

Tools Utilized by Database Users 279
Summary 279
Q&A 280
Workshop 280

HOUR 19: Managing Database Security 283
What Is Database Security? 283
What Are Privileges? 284
Controlling User Access 287
Controlling Privileges Through Roles 291
Summary 293
Q&A 293
Workshop 294

Part VII: Summarized Data Structures

HOUR 20: Creating and Using Views and Synonyms 297
What Is a View? 297
Creating Views 300
Updating Data Through a View 307
Dropping a View 307
Performance Impact of Nested Views 308
What Is a Synonym? 308
Summary 310
Q&A 310
Workshop 311

HOUR 21: Working with the System Catalog 313
What Is the System Catalog? 313
How Is the System Catalog Created? 314
What Is Contained in the System Catalog? 315
System Catalog Tables by Implementation 316
Querying the System Catalog 317
Updating System Catalog Objects 320

Summary 320
Q&A 321
Workshop 321

Part VIII: Applying SQL Fundamentals in Today's World

HOUR 22: Advanced SQL Topics 323
Cursors 323
Stored Procedures and Functions 326
Triggers 329
Dynamic SQL 331
Call-Level Interface 331
Using SQL to Generate SQL 332
Direct Versus Embedded SQL 333
Windowed Table Functions 333
Working with XML 334
Summary 335
Q&A 335
Workshop 336

HOUR 23: Extending SQL to the Enterprise, the Internet, and the Intranet 337
SQL and the Enterprise 337
Accessing a Remote Database 339
SQL and the Internet 342
SQL and the Intranet 343
Summary 344
Q&A 344
Workshop 344

HOUR 24: Extensions to Standard SQL 347
Various Implementations 347
Example Extensions 350
Interactive SQL Statements 353
Summary 354
Q&A 354
Workshop 355

Part IX: Appendixes

APPENDIX A: Common SQL Commands 357

SQL Statements 357

SQL Clauses 360

APPENDIX B: Installing Oracle and Microsoft SQL 363

Windows Installation Instructions for Oracle 363

Windows Installation Instructions for Microsoft SQL Server 365

APPENDIX C: Answers to Quizzes and Exercises 369

Hour 1, “Welcome to the World of SQL” 369

Hour 2, “Defining Data Structures” 371

Hour 3, “Managing Database Objects” 373

Hour 4, “The Normalization Process” 374

Hour 5, “Manipulating Data” 376

Hour 6, “Managing Database Transactions” 379

Hour 7, “Introduction to the Database Queries” 380

Hour 8, “Using Operators to Categorize Data” 383

Hour 9, “Summarizing Data Results from a Query” 386

Hour 10, “Sorting and Grouping Data” 388

Hour 11, “Restructuring the Appearance of Data” 390

Hour 12, “Understanding Dates and Times” 391

Hour 13, “Joining Tables in Queries” 392

Hour 14, “Using Subqueries to Define Unknown Data” 394

Hour 15, “Combining Multiple Queries into One” 396

Hour 16, “Using Indexes to Improve Performance” 397

Hour 17, “Improving Database Performance” 399

Hour 18, “Managing Database Users” 403

Hour 19, “Managing Database Security” 404

Hour 20, “Creating and Using Views and Synonyms” 404

Hour 21, “Working with the System Catalog” 406

Hour 22, “Advanced SQL Topics” 407

Hour 23, “Extending SQL to the Enterprise, the Internet, and the Intranet” 408

Hour 24, "Extensions to Standard SQL" 409

APPENDIX D: Bonus Exercises **411**

APPENDIX E: Glossary **425**

Index **429**

About the Authors

For more than 20 years each, the authors have studied, applied, and documented the SQL standard and its application to critical database systems in this book. The authors are experts in data management, specializing in Oracle, Microsoft, and other leading technologies.

Ryan Stephens is the co-founder and CEO of Perpetual Technologies, Inc. and Indy Data Partners in Indianapolis. Ryan has studied and consulted in the IT field for more than 20 years, specializing in data management, SQL, and Oracle. Ryan authored and taught database and SQL classes for Indiana University-Purdue University in Indianapolis for 5 years, and was a programmer analyst for the Indiana Army National Guard for 12 years. Ryan has written a variety of database and SQL books for Sams Publishing.

Arie D. Jones is the Vice President for Emerging Technologies for Indy Data Partners, Inc. (IDP) in Indianapolis. Arie leads IDP's team of experts in planning, design, development, deployment, and management of database environments and applications to achieve the best combination of tools and services for each client. He is a regular speaker at technical events and has authored several books and articles pertaining to database-related topics.

Ronald Plew is retired as co-founder and vice president of Perpetual Technologies, Inc. Ron studied and consulted in the field of relational database technology for more than 20 years and has co-authored several books for Sams Publishing. Ron taught SQL and database classes for Indiana University-Purdue University in Indianapolis for 5 years. He is a retired programmer analyst from the Indiana Army National Guard.

Dedication

This book is dedicated to my strong and driven wife, Jill, and to my three children by whom I'm equally smitten and amazed—Daniel, Autumn, and Alivia.

—Ryan

I would like to dedicate this book to my wife, Jackie, for being understanding and supportive during the long hours that it took to complete this book.

—Arie

Acknowledgments

Thank you to all who have been supportive and patient during the writing of the six editions of this book. This includes, but is probably not limited to, our family, friends, employees, partners, and random people in public who might sense our preoccupied minds. Thank you especially to Ron Plew, my original co-author of this book and business partner. Neither the first edition of this book, nor our company, would exist without your contributions; and to Arie Jones for spearheading this edition and taking on the bulk of the work, while providing leadership during the day at Indy Data Partners. Thanks also to Marshall Pyle and Jacinda Simmerman for technical edits and your perspectives to improve the quality of this text for our readers. And as always, thanks to the staff at Sams Publishing for your attention to detail and patience. It is always a pleasure working with you.

—Ryan

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.

We welcome your comments. You can email or write 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 better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the authors and editors who worked on the book.

Email: consumer@samspublishing.com

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

Reader Services

Register your copy of *Sams Teach Yourself SQL in 24 Hours, Sixth Edition*, at informit.com for convenient access to downloads, updates, and corrections as they become available. To start the registration process, go to informit.com/register and log in or create an account*. Enter the product ISBN, 9780672337598, and click Submit. Once the process is complete, you will find any available bonus content under Registered Products.

*Be sure to check the box that you would like to hear from us in order to receive exclusive discounts on future editions of this product.

This page intentionally left blank

HOUR 9

Summarizing Data Results from a Query

What You'll Learn in This Hour:

- Definition of functions
- Using aggregate functions
- Summarizing data with aggregate functions
- Results from using functions

In this hour, you learn about SQL's aggregate functions. You can perform a variety of useful functions with aggregate functions, such as getting the highest total of a sale or counting the number of orders processed on a given day. The real power of aggregate functions will be discussed in the next hour when you tackle the `GROUP BY` clause.

Aggregate Functions

Functions are keywords in SQL used to manipulate values within columns for output purposes. A *function* is a command normally used with a column name or expression that processes the incoming data to produce a result. SQL contains several types of functions. This hour covers aggregate functions. An *aggregate function* provides summarization information for a SQL statement, such as counts, totals, and averages.

The basic set of aggregate functions discussed in this hour are

- `COUNT`
- `SUM`
- `MAX`
- `MIN`
- `AVG`

The following query lists the employee information from the EMPLOYEES table. Note that some of the employees do not have data assigned in some of the columns. We use this data for most of this hour's examples.

```
SELECT TOP 10 EMPLOYEEID,LASTNAME,
 CITY,STATE,PAYRATE,SALARY
 FROM EMPLOYEES;

EMPLOYEEID  LASTNAME          CITY               STATE      PAYRATE        SALARY
----------- ----------------- ------------------ ---------- -------------- --------
1           Iner              Red Dog            NULL                      54000.00
2           Denty             Errol              NH         22.24          NULL
3           Sabbah            Errol              NH         15.29          NULL
4           Loock             Errol              NH         12.88          NULL
5           Sacks             Errol              NH         23.61          NULL
6           Arcoraci          Alexandria         LA         24.79          NULL
7           Astin             Espanola           NM         18.03          NULL
8           Contreraz         Espanola           NM         NULL           60000.00
9           Capito            Espanola           NM         NULL           52000.00
10          Ellamar           Espanola           NM         15.64          NULL

(10 row(s) affected)
```

COUNT

You use the COUNT function to count rows or values of a column that do not contain a NULL value. When used within a query, the COUNT function returns a numeric value. You can also use the COUNT function with the DISTINCT command to only count the distinct rows of a dataset. ALL (opposite of DISTINCT) is the default; it is not necessary to include ALL in the syntax. Duplicate rows are counted if DISTINCT is not specified. One other option with the COUNT function is to use it with an asterisk. COUNT(*) counts all the rows of a table including duplicates, regardless of whether a NULL value is contained in a column.

NOTE

DISTINCT Can Be Used Only in Certain Circumstances

You cannot use the DISTINCT command with COUNT(*), only with COUNT (*column_name*).

The syntax for the COUNT function follows:

```
COUNT [ (*) | (DISTINCT | ALL) ] (COLUMN NAME)
```

This example counts all employee IDs:

```
SELECT COUNT(EMPLOYEEID) FROM EMPLOYEES
```

This example counts only the distinct rows:

```
SELECT COUNT(DISTINCT SALARY)FROM EMPLOYEES
```

This example counts all rows for SALARY:

```
SELECT COUNT(ALL SALARY)FROM EMPLOYEES
```

This final example counts all rows of the EMPLOYEES table:

```
SELECT COUNT(*) FROM EMPLOYEES
```

COUNT(*) is used in the following example to get a count of all records in the EMPLOYEES table. There are 5,611 employees.

```
SELECT COUNT(*)
FROM EMPLOYEES;
-----------
5611

(1 row(s) affected)
```

CAUTION

COUNT(*) Is Different from Other Count Variations

COUNT(*) produces slightly different calculations than other count variations. This is because when the COUNT function is used with the asterisk, it counts the rows in the returned result set without regard to duplicates and NULL values. This is an important distinction. If you need your query to return a count of a particular field and include NULLs, you need to use a function such as ISNULL to replace the NULL values.

COUNT(EMPLOYEEID) is used in the next example to get a count of all the employee identification IDs that exist in the table. The returned count is the same as the last query because all employees have an identification number.

```
SELECT COUNT(EMPLOYEEID)
FROM EMPLOYEES;
-----------
5611

(1 row(s) affected)
```

COUNT([STATE]) is used in the following example to get a count of all the employee records that have a state assigned. Look at the difference between the two counts. The difference is the number of employees who have NULL in the STATE column.

```
SELECT COUNT([STATE])
FROM EMPLOYEES;
-----------
5147
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

The following examples obtain a count of all salary amounts and then all the distinct salary amounts in the EMPLOYEES table.

```
SELECT COUNT(SALARY )
FROM EMPLOYEES;
-----------
1359
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)

SELECT COUNT(DISTINCT SALARY )
FROM EMPLOYEES;
-----------
45
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

The SALARY column had a lot of matching amounts, so the DISTINCT values make the counts drop dramatically.

NOTE

Data Types Do Not Use COUNT

Because the COUNT function counts the rows, data types do not play a part. The rows can contain columns with any data type. The only thing that actually counts is whether the value is NULL.

SUM

The SUM function returns a total on the values of a column for a group of rows. You can also use the SUM function with DISTINCT. When you use SUM with DISTINCT, only the distinct rows are totaled, which might not have much purpose. Your total is not accurate in that case because rows of data are omitted.

The syntax for the SUM function follows:

```
SUM ([ DISTINCT ] COLUMN NAME)
```

CAUTION

SUM Must Be Numeric

The value of an argument must be numeric to use the SUM function. You cannot use the SUM function on columns that have a data type other than numeric, such as character or date.

This example totals the salaries:

```
SELECT SUM(SALARY) FROM EMPLOYEES
```

This example totals the distinct salaries:

```
SELECT SUM(DISTINCT SALARY) FROM EMPLOYEES
```

In the following query, the sum, or total amount, of all salary values is retrieved from the EMPLOYEES table:

```
SELECT SUM(SALARY)
FROM EMPLOYEES;
------------------------------
70791000.00
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

Observe the way the DISTINCT command in the following example skews the previous results by 68 million dollars. This is why it is rarely useful.

```
SELECT SUM(DISTINCT COST)
FROM EMPLOYEES;
------------------------------
2340000.00
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

The following query demonstrates that although some aggregate functions require numeric data, this is only limited to the type of data. Here the ZIP column of the EMPLOYEES table shows that the implicit conversion of the VARCHAR data to a numeric type is supported in Oracle:

```
SELECT SUM(ZIP)
FROM EMPLOYEES;
SUM(ZIP)
-----------
280891448
```

Some aggregate functions require numeric data; this is only limited to the type of data. If the data can be converted implicitly, for example, the string '12345' to an integer, then you can use the aggregate function. When you use a type of data that cannot be implicitly converted to a numeric type, such as the POSITION column, it results in an error, as in the following example:

```
SELECT SUM(POSITION)
FROM EMPLOYEES;
Msg 8117, Level 16, State 1, Line 1
Operand data type varchar is invalid for sum operator.
```

AVG

The AVG function finds the average value for a given group of rows. When used with the DISTINCT command, the AVG function returns the average of the distinct rows. The syntax for the AVG function follows:

```
AVG ([ DISTINCT ] COLUMN NAME)
```

NOTE

AVG Must Be Numeric

The value of the argument must be numeric for the AVG function to work.

The average value for all values in the EMPLOYEES table's SALARY column is retrieved in the following example:

```
SELECT AVG(SALARY)
FROM EMPLOYEES;
------------------------------
52090.507726
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

This example returns the distinct average salary:

```
SELECT AVG(DISTINCT SALARY)
FROM EMPLOYEES;
------------------------------
52000.000000
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

CAUTION

Sometimes Your Data Is Truncated

In some implementations, the results of your query might be truncated to the precision of the data type. You need to review your database system's documentation to ensure you understand what the normal precision for the various data types is. This will prevent you from unnecessarily truncating data and possibly getting an unexpected result due to the data not being of the proper precision.

The next example uses two aggregate functions in the same query. Because some employees are paid hourly and others are on salary, you want to retrieve the average value for both PAYRATE and SALARY.

```
SELECT AVG(PAYRATE) AS AVG_PAYRATE, AVG(SALARY) AS AVG_SALARY
FROM EMPLOYEES;
AVG_PAYRATE                    AVG_SALARY
------------------------------ ------------------------------
18.473012                      52090.507726
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

Notice how the use of aliases makes the output more readable with multiple aggregate values. Also remember that the aggregate function can work on any numeric data. So you can perform calculations within the parentheses of the function as well. So if you need to get the average hourly rate of salaried employees to compare to the average rate of hourly employees, you could write the following:

```
SELECT AVG(PAYRATE) AS AVG_PAYRATE, AVG(SALARY/2040) AS AVG_SALARY_RATE
FROM EMPLOYEES;
AVG_PAYRATE                    AVG_SALARY_RATE
------------------------------ ------------------------------
18.473012                      25.5345625
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

MAX

The MAX function returns the maximum value from the values of a column in a group of rows. NULL values are ignored when using the MAX function. Using MAX with the DISTINCT command is an option. However, because the maximum value for all the rows is the same as the distinct maximum value, DISTINCT is useless.

The syntax for the MAX function is

```
MAX([ DISTINCT ] COLUMN NAME)
```

The following example returns the highest SALARY in the EMPLOYEES table:

```
SELECT MAX(SALARY)
FROM EMPLOYEES;
------------------------------
74000.00
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

This example returns the highest distinct salary:

```
SELECT MAX(DISTINCT SALARY)
FROM EMPLOYEES;
------------------------------
74000.00
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

You can also use aggregate functions such as MAX and MIN (covered in the next section) on character data. In the case of these values, collation of your database comes into play again. Most commonly your database collation is set to a dictionary order, so the results are ranked according to that. For example, say you perform a MAX on the CITY column of the employees table:

```
SELECT MAX(CITY) AS MAX_CITY
FROM EMPLOYEES;
MAX_CITY
------------------------------
Zwara

(1 row(s) affected)
```

In this instance, the function returned the largest value according to a dictionary ordering of the data in the column.

MIN

The MIN function returns the minimum value of a column for a group of rows. NULL values are ignored when using the MIN function. Using MIN with the DISTINCT command is an option. However, because the minimum value for all rows is the same as the minimum value for distinct rows, DISTINCT is useless.

The syntax for the MIN function is

```
MIN([ DISTINCT ] COLUMN NAME)
```

The following example returns the lowest SALARY in the EMPLOYEES table:

```
SELECT MIN(SALARY)
FROM EMPLOYEES;
------------------------------
30000.00
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

This example returns the lowest distinct salary:

```
SELECT MIN(DISTINCT SALARY)
FROM EMPLOYEES;
------------------------------
30000.00
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)
```

NOTE

DISTINCT and Aggregate Functions Don’t Always Mix

One important thing to keep in mind when using aggregate functions with the DISTINCT command is that your query might not return the wanted results. The purpose of aggregate functions is to return summarized data based on all rows of data in a table. When DISTINCT is used it is applied first to the results and then those results are passed on to the aggregate function, which can dramatically alter the results. You need to ensure that when you work with DISTINCT with aggregate functions that you understand this.

As with the MAX function, the MIN function can work against character data and returns the minimum value according to the dictionary ordering of the data.

```
SELECT MIN(CITY) AS MIN_CITY
FROM EMPLOYEES;
MIN_CITY
------------------------------
 AFB MunicipalCharleston SC

(1 row(s) affected)
```

Summary

Aggregate functions can be useful and are quite simple to use. In this hour you learned how to count values in columns, count rows of data in a table, get the maximum and minimum values for a column, figure the sum of the values in a column, and figure the average value for values in a column. Remember that NULL values are not considered when using aggregate functions, except when using the COUNT function in the format COUNT(*).

Aggregate functions are the first functions in SQL that you have learned in this book, but more follow in the coming hours. You can also use aggregate functions for group values, which are discussed during the next hour. As you learn about other functions, you see that the syntaxes of most functions are similar to one another and that their concepts of use are relatively easy to understand.

Q&A

Q. Why are NULL values ignored when using the MAX or MIN function?

A. A NULL value means that nothing is there, so there would be no maximum or minimum value.

Q. Why don't data types matter when using the COUNT function?

A. The COUNT function counts only rows.

Q. Does the data type matter when using the SUM or AVG function?

A. Not exactly. If the data can be implicitly converted to numeric data, then it will still work. It's less a function of what the data type is and more about what data is stored in it.

Q. Are you limited to using only column names inside of aggregate functions?

A. No, you can use any type of calculation or formula as long as the output corresponds to the proper type of data that the function is expecting to use.

Workshop

The following workshop is composed of a series of quiz questions and practical exercises. The quiz questions are designed to test your overall understanding of the current material. The practical exercises are intended to afford you the opportunity to apply the concepts discussed during the current hour, as well as build upon the knowledge acquired in previous hours of study. Please take time to complete the quiz questions and exercises before continuing. Refer to Appendix C, "Answers to Quizzes and Exercises," for answers.

Quiz

1. True or false: The AVG function returns an average of all rows from a SELECT column, including any NULL values.
2. True or false: The SUM function adds column totals.
3. True or false: The COUNT(*) function counts all rows in a table.
4. True or false: The COUNT([column name]) function counts NULL values.
5. Will the following SELECT statements work? If not, what fixes the statements?

   a.
   ```
   SELECT COUNT *
   FROM EMPLOYEES;
   ```
   b.
   ```
   SELECT COUNT(EMPLOYEEID), SALARY
   FROM EMPLOYEES;
   ```
   c.
   ```
   SELECT MIN(PAYRATE), MAX(SALARY)
   FROM EMPLOYEES
   WHERE SALARY > 50000;
   ```
   d.
   ```
   SELECT COUNT(DISTINCT EMPLOYEEID) FROM EMPLOYEES;
   ```
   e.
   ```
   SELECT AVG(LASTNAME) FROM EMPLOYEES;
   ```
   f.
   ```
   SELECT AVG(CAST(ZIP AS INT)) FROM EMPLOYEES;
   ```

Exercises

1. Use the EMPLOYEES table to construct SQL statements to solve the following exercises:
   a. What is the average salary?
   b. What is the maximum pay rate for hourly employees?
   c. What are the total salaries?
   d. What is the minimum pay rate?
   e. How many rows are in the table?
2. Write a query to determine how many employees are in the company whose last names begin with a ***G***.
3. Write a query to determine the minimum and maximum salary and pay rates per city for employees.
4. Write two sets of queries to find the first employee name and last employee name when they are listed in alphabetical order.
5. Write a query to perform an AVG function on the employee names. Does the statement work? Determine why it is that you got that result.
6. Write a query to display the average value of employees' salaries that takes NULL values into account. Hint: You won't be using the AVG function.

This page intentionally left blank

Index

A

accessing remote databases, 339-340
- JDBC (Java Database Connectivity), 340
- ODBC (Open Database Connectivity), 339-340
- OLE DB, 340
- vendor connectivity product, 340-341
- Web interfaces, 341-342

adding
- columns
  - auto-incrementing columns, 40-41
  - to tables, 40
- time to dates, 186-187

addition, 128

ADMIN OPTION, 288

aggregate functions, 133-134, 334
- AVG, 138-139
- COUNT, 134-136
- creating groups and, 147-150
- DISTINCT, 141
- MAX, 139-140
- MIN, 140-141
- SUM, 136-138

aliases, table aliases, 200

ALL, 118-120

ALTER ANY TABLE, 285

ALTER DATABASE, 285

ALTER TABLE, 39, 47, 357

ALTER USER, 285

ALTER VIEW, 300

altering
- indexes, 250
- users, 277-278

American National Standards Institute (ANSI), 2, 159

AND, 120-121

ANSI (American National Standards Institute), 2, 159

ANSI SQL, 2, 349

ANSI standard, SELECT, 348

ANY, 118-120

arguments, 94

arithmetic operators, 128
- addition, 128
- combinations, 130
- division, 129
- multiplication, 129
- subtraction, 128-129

arranging tables, FROM clause, 257

ASCII function, 172

asterisks, 94-95

auto-incrementing columns, adding to tables, 40-41

AVG, 138-139

avoiding
- full table scans, 260
- HAVING clause, 263
- large sort operations, 263
- OR, performance, 262-263

B

back-end applications, 337-338

BACKUP ANY TABLE, 285

BACKUP DATABASE, 285

base tables, joins, 207-208

basic data types, 20-21
- BOOLEAN values, 26-27
- date and time data types, 24-25
- decimal types, 23-24
- domains, 28
- fixed-length strings, 21
- floating-point decimals, 24
- integers, 24
- large object types, 22
- literal strings, 25-26
- NULL data types, 26
- numeric types, 22-23
- user-defined data types, 27
- varying-length strings, 21-22

batch loads, disabling indexes during, 264

benefits of normalization, 62-63

BETWEEN, 115

BLOB, 22

bonus exercises, 411-423

BOOLEAN values, 26-27

C

call-level interface (CLI), 331-332

Canary Airlines, 11-12

cardinality, columns, 250

Cartesian product, joins, 208-210

CASCADE, 289, 305

case-sensitivity, 101-102

- data, 68

CAST operator, 188

character functions, 159-160

- ASCII function, 172
- COALESCE, 170-171
- combining, 176-177
- CONCAT, 160-162
- ISNULL, 169-170
- LENGTH, 169
- LOWER, 163-164
- LPAD, 171
- LTRIM, 167-168
- REPLACE, 166
- RPAD, 171-172
- RTRIM, 168
- SUBSTR, 164-165
- TRANSLATE, 165-166
- UPPER, 162-163

character strings

- converting dates to, 191-192
- converting numbers to, 175-176
- converting to dates, 192-193
- converting to numbers, 173-174

check constraints, 48-49

CLI (call-level interface), 331-332

client/server model, 5-6

closing cursors, 326

COALESCE, 170-171

Codd, Dr. E.F., 2

coding standards, 257

collation, 101

column aliases, queries, 105-106

column lists, ordering, 70

columns, 14

- adding auto-incrementing columns, 40-41
- adding mandatory columns to tables, 40
- cardinality, 250
- controlling user access on individual columns, 289-290
- inserting data into limited columns, 69-70
- modifying, 41-42
- qualifying in queries, 199
- restricting access with views, 299
- tables, part of, 35-36
- updating
  - multiple columns, 74-75
  - value of single column, 74

combining

- arithmetic operators, 130
- character functions, 176-177

commands, 8-11

- data administration commands, 10-11
- DCL (Data Control Language), 10
- DDL (Data Definition Language), 9
- DML (Data Manipulation Language), 9-10
- DQL (Data Query Language), 10
- transactional control commands, 11

COMMIT, 11, 357

- controlling transactions, 83-84

comparing

- GROUP BY clause versus ORDER BY clause, 150-153
- single queries versus compound queries, 229-230

comparison operators, 110

- combinations of, 113
- equality, 110
- less than and greater than, 111-112
- non-equality, 111

compliance, ANSI SQL, 349

composite indexes, 247

compound queries

- GROUP BY clause, 237-238
- ORDER BY clause, 235-236
- retrieving accurate data, 238-239
- versus single queries, 229-230

compound query operators, 230

- EXCEPT, 234-235
- INTERSECT, 233-234
- UNION, 230-232
- UNION ALL, 232-233

CONCAT, 160-162

concatenation, 160

conjunctive operators, 120

- AND, 120-121
- OR, 121-123

CONNECT, 8

constraints, dropping, 49. ***See also*** **integrity constraints**

controlling

- privileges through roles, 291
  - CREATE ROLE, 292
  - DROP ROLE, 292
  - SET ROLE, 292-293

transactions, 82-83
COMMIT, 83-84
RELEASE SAVEPOINT, 89
ROLLBACK, 85-86
ROLLBACK TO SAVEPOINT, 87-89
SAVEPOINT, 86
SET TRANSACTION, 89
user access, 287
GRANT, 287-288
on individual columns, 289
privileges, 290-291
PUBLIC database account, 289-290
REVOKE, 289

conversion functions, 173
converting numbers to character strings, 175-176
converting character strings to numbers, 173-174

conversions, date conversions, 188
date pictures, 189-191

converting
character strings to dates, 192-193
character strings to numbers, 173-174
dates to character strings, 191-192
numbers to character strings, 175-176

correlated subqueries, 223-224

cost-based optimization, performance, 264-265

COUNT, 104, 134-136

counting records in tables, 103-104

CREATE ANY TABLE, 285

CREATE DATABASE, 285

CREATE INDEX, 245, 357

CREATE PROCEDURE, 285

CREATE ROLE, 292, 357

CREATE SCHEMA, 275-276

CREATE SYNONYM, 309

CREATE TABLE AS, 358

CREATE TABLE, 37-39, 245, 285, 358

CREATE TRIGGER, 285, 329-330

CREATE TYPE, 358

CREATE USER, 273, 285, 358

CREATE VIEW, 285, 300, 358

cross joins, 208-210

CUBE expression, grouping data, 153-155

current date, 183-184

cursors, 323-324
closing, 326
fetching data, 325-326
grouping, 145
CUBE expression, 153-155
GROUP BY clause. *See* GROUP BY clause
HAVING clause, 155-156
ROLLUP expression, 153-155
inserting
in limited columns of tables, 69-70
from other tables, 70-72
in tables, 68-69
opening, 324
populating tables, 68
redundancy
denormalization, 63-64
logical database design, 57-58
retrieving accurate data, compound query operators, 238-239
selecting from multiple tables, 197
summarized data, views, 299
tables. *See* tables
updating, 73
multiple columns, 74-75
through views, 307
value of single column, 74

D

Data access, simplifying with views, 298-299

Data Control Language (DCL), 10

Data Definition Language (DDL), 9

data dictionary. *See* system catalog

data manipulation, overview, 67

Data Manipulation Language. *See* DML (Data Manipulation Language)

Data Query Language. *See* DQL (Data Query Language)

data types, 19
basic data types, 20-21
BOOLEAN values, 26-27
date and time data types, 24-25, 182
decimal types, 23-24
fixed-length strings, 21
floating-point decimals, 24
integers, 24
large object types, 22
NULL data types, 26
numeric types, 22-23

-
    - user-defined data types, 27
    - varying-length strings, 21-22
- **database administrator (DBA), 19**
- **database design information, system catalog, 316**
- **database management system (DBMS), 1**
- **database objects, schemas, 33-35**
- **database queries, 93**
  - case-sensitivity, 101-102
  - constraints, 103
  - SELECT, 93-96
    - FROM clause, 97
    - ORDER BY clause, 99-101
    - SELECT clause, 94-96
    - WHERE clause, 97-98
  - writing, 102-103
    - column aliases, 105-106
    - counting records in tables, 103-104
    - selecting data from another table, 105
- **database security, 283-284**
  - controlling privileges through roles, 291-292
    - CREATE ROLE, 292
    - DROP ROLE, 292
    - SET ROLE, 292-293
  - controlling user access, 287
    - GRANT, 287-288
    - on individual columns, 289
    - PUBLIC database account, 289-290
    - REVOKE, 289
- **database tuning, 254**
  - versus SQL statement tuning, 254
- **database vendors, 7-8**
- **databases**
  - defined, 4-5
  - denormalization, 63-64
  - relational, 5
  - web-based database systems, 6-7
- **DATE, 182**
- **date and time, 182**
  - adding time to dates, 186-187
  - current date, 183-184
  - leap years, 182
  - time zones, 184-185
- **date and time data types, 24-25, 182**
  - implementation-specific data types, 183
- **date and time storage, 181**
- **date conversions, 188**
  - date pictures, 189
- **date functions, 183, 187-188**
  - converting dates to character strings, 191-192
  - current date, 183-184
  - time zones, 184-185
- **date pictures, 189**
- **dates**
  - adding time to, 186-187
  - converting character strings to, 192-193
  - converting to character strings, 191-192
- **DATETIME, 24-25**
  - basic data types, literal strings, 25-26
  - converting character strings to dates, 192-193
  - converting dates to character strings, 191-192
  - data types, 182
  - date parts by platform, 189-191
- **DATETIME elements, 182**
- **DB_DATAREADER, 291**
- **DB_DATAWRITER, 291**
- **DB_DDLADMIN, 290**
- **DBA (database administrator), 19**
- **DBMS (database management system), 1**
- **DCL (Data Control Language), 10**
- **DDL (Data Definition Language), 9**
- **decimal types, 23-24**
- **DELETE statement, 75-76, 358**
  - subqueries, 218-219
- **deleting data from tables, 75-76**
- **denormalization, 63-64**
- **differences between implementation, 347-349**
- **Direct SQL, 333**
- **disabling indexes during batch loads, 264**
- **DISCONNECT, 8**
- **DISTINCT, 96**
  - aggregate functions, 141
  - MIN, 140-141
- **division, 129**
- **DML (Data Manipulation Language), 9-10**
  - INSERT statement, subqueries, 217
  - overview, 67
- **domains, data types, 28**
- **DQL (Data Query Language), 10**
  - SELECT, 93-94

DROP, 279

DROP INDEX, 359

DROP ROLE, 292

DROP SCHEMA, 277

DROP SYNONYM, 310

DROP TABLE, 285, 359

DROP USER, 285, 359

DROP VIEW, 307-308, 359

dropping
- constraints, 49
- indexes, 250-251
- schemas, 277
- synonyms, 310
- tables, 44
  - used by views, 298
- triggers, 331
- views, 307-308

dynamic SQL, 331

E

elements of
- DATETIME, 182
- tables, modifying, 40

embedded functions, 177

embedded SQL, 333

embedded subqueries, 219-223

end user needs, logical database design, 57

enterprises, 337
- back-end applications, 337-338
- front-end applications, 338-339

equality operators, 110

equijoins, 198-200

examples, Canary Airlines, 11-12

EXISTS, 117-118

EXIT, 8

EXPLAIN PLAN, 265

extensions to SQL, 349

EXTRACTVALUE, 334

F

FETCH, 325-326

fetching data from cursors, 325-326

fields, 14

first normal form, 58-59

fixed-length strings, 21

FLOAT, 24

floating-point decimals, 24

foreign key constraints, 46-47

formatting SQL statements, 254
- arranging tables in FROM clause, 257
- most restrictive condition, 258-259
- ordering join conditions, 257-258
- for readability, 255-257

FROM clause, 360
- arranging tables, 257
- with SELECT, 97

front-end applications, 338-339

full table scans, 243, 260-261
- avoiding, 260

functions
- aggregate functions, 133-134, 334
  - AVG, 138-139
  - COUNT, 134-136
  - creating groups, 147-150
  - MAX, 139-140
  - MIN, 140-141
  - SUM, 136-138
- character functions, 159-160
  - ASCII function, 172
  - COALESCE, 170-171
  - combining, 176-177
  - CONCAT, 160-162
  - ISNULL, 169-170
  - LENGTH, 169
  - LOWER, 163-164
  - LPAD, 171
  - LTRIM, 167-168
  - REPLACE, 166
  - RPAD, 171-172
  - RTRIM, 168
  - SUBSTR, 164-165
  - TRANSLATE, 165-166
  - UPPER, 162-163
- conversion functions, 173
  - converting character strings to numbers, 173-174
  - converting numbers to character strings, 175-176
- date functions, 183, 187-188
  - current date, 183-184
  - time zones, 184-185
- embedded functions, 177
- EXTRACTVALUE, 334
- mathematical functions, 172-173
- TRANSLATE, 160
- windowed table functions, 333-334

G

generating SQL, 332-333

GETDATE(), 184

GRANT, 273, 359

- controlling user access, 287-288

GRANT ENABLE TO, 333

GRANT ENABLE TO USERNAME, 333

GRANT OPTION, 288-289

granting privileges, 287

greater than operators, 111-112

GROUP BY clause, 145-146, 361

- with compound queries, 237-238
- creating groups with aggregate functions, 147-150
- grouping selected data, 147
- versus ORDER BY, 150-153

group functions, GROUP BY clause, 146

grouping data, 145

- CUBE expression, 153-155
- GROUP BY clause. *See* GROUP BY clause
- HAVING clause, 155-156
- ROLLUP expression, 153-155

GUI tools, 279

H

HAVING clause, 361

- avoiding, 263
- grouping data, 155-156

home pages, 342

I

implementation

- differences between, 347-349
- system catalog tables, 316-317

implicit indexes, 247-248

IN operator, 115-116

indexes

- altering, 250
- composite indexes, 247
- creating, 245
  - CREATE INDEX, 245
- disabling during batch loads, 264
- dropping, 250-251
- how they work, 244-245
- implicit, 247-248
- overview, 243-244
- reasons for using, 248
- reasons to avoid, 248-250
- single-column, 246
- unique, 246-247

ineffective indexes, 249

information stored in system catalog, 315

inner joins, 198-200

INSERT statement, 68-69, 286, 359

- subqueries, 217

INSERT(*column_name*), 286

inserting

- data
  - in limited columns of tables, 69-70
  - in tables, 68-69
  - from other tables, 70-72
- NULL values, 72-73

INSERT…SELECT, 359

installing

- Microsoft SQL Server on Windows, 365-367
- Oracle on Windows, 363-365

integers, data types, 24

integrity, referential integrity, 44-47, 62-63

integrity constraints, 44

- check constraints, 48-49
- foreign key constraints, 46-47
- NOT NULL constraints, 48
- primary key constraints, 44-45
- unique constraints, 46

interactive SQL statements, 353-354

International Standards Organization (ISO), 2

Internet

- making data available to employees and privileged customers, 342
- making data available worldwide, 342
- security, 343
- web interfaces, 341

INTERSECT, 233-234

INTERVAL command, 187

intranets

- making data available to employees and privileged customers, 342
- SQL and, 343-344

IS NOT NULL, 127

IS NULL, 127

ISNULL, 135, 169-170

ISO (International Standards Organization), 2

J

JDBC (Java Database Connectivity), 340

JOIN, 199

joins, 198

base tables, 207-208

Cartesian product, 208-210

cross joins, 208-210

equijoins, 198-200

joining multiple keys, 206-207

non-equijoin joins, 200-201

ordering, 257-258

outer joins, 201-204

self joins, 204-206

table aliases, 200

K

keys

joining multiple keys, 206-207

primary keys, 14, 44

keywords

COMMIT, 83

SELECT, 95

SET, 75

L

LAN (local area network), 6

large object types, 22

large sort operations, avoiding, 263

leap years, 182

LENGTH, 169

less than operators, 111-112

LIKE, 116-117

performance, 261-262

limitations of normalization, 63

local area network (LAN), 6

logical database design, normalization, 57

data redundancy, 57-58

end user needs, 57

logical model, 57

logical operators, 113-114

ALL, 118-120

ANY, 118-120

BETWEEN, 115

EXISTS, 117-118

IN, 115-116

IS NULL, 114

LIKE, 116-117

SOME, 118-120

LOWER function, 163-164

LPAD, 171

LTRIM, 167-168

M

mathematical functions, 172-173

MAX, 139-140

Microsoft SQL Server

installing, 365-367

privileges, 290-291

SELECT, 348-

users, creating, 274

MIN, 140-141

modifying

columns, 41-42

indexes, 250

tables

adding auto-incrementing columns, 40-41

adding mandatory columns, 40

ALTER TABLE, 39

columns, 41-42

elements of, 40

users, 277-278

most restrictive condition, SQL statements, formatting, 258-259

multiple columns, updating, 74-75

multiple keys, joining, 206-207

multiple tables, creating views from, 302-303

multiplication, 129

MySQL, 352-353

date parts, 191

users, creating, 275

N

naming conventions

normalization, 61-62

savepoints, 87

tables, 13, 39

negative operators, 123

IS NOT NULL, 127

NOT BETWEEN, 124-125

NOT EQUAL, 124

NOT EXISTS, 127-128

NOT IN, 125-126

NOT LIKE, 126-127

nested views, performance, 308

non-equality operators, 111

non-equijoin joins, 200-201

normal forms, 58

- first, 58-59
- second, 59-61
- third, 61

normalization, 55

- benefits of, 62-63
- limitations of, 63
- logical database design, 57
    - data redundancy, 57-58
    - end user needs, 57
- naming conventions, 61-62
- normal forms, 58
    - first, 58-59
    - second, 59-61
    - third, 61
- raw databases, 56

NOT BETWEEN, 124-125

NOT EQUAL, 124

NOT EXISTS, 127-128

NOT IN, 125-126

NOT LIKE, 126-127

NOT NULL constraints, 48

NOW, 184

NULL data types, 26

NULL value checker, 169-170

NULL values, 14-15

- inserting, 72-73

numbers

- converting character strings to, 173-174
- converting to character strings, 175-176

numeric types, 22-23

O

ODBC (Open Database Connectivity), 339-340

OLE DB, 340

opening cursors, 324

operators, 109-113

- arithmetic operators, 128
    - addition, 128
    - combinations, 130
    - division, 129
    - multiplication, 129
    - subtraction, 128-129
- comparison operators, 110-113
    - equality, 110
    - less than and greater than, 111-112
    - non-equality, 111
- conjunctive operators, 120
    - AND, 120-121
    - OR, 121-123
- defined, 109
- EXCEPT, 234-235
- INTERSECT, 233-234
- logical operators, 113-114
    - ALL, 118-120
    - ANY, 118-120
    - BETWEEN, 115
    - EXISTS, 117-118
    - IN, 115-116
    - IS NULL, 114
    - LIKE, 116-117
    - SOME, 118-120
- negative operators, 123
    - IS NOT NULL, 127
    - NOT BETWEEN, 124-125
    - NOT EQUAL, 124
    - NOT EXISTS, 127-128
    - NOT IN, 125-126
    - NOT LIKE, 126-127
- UNION, 229-230
- UNION ALL, compound query operators, 232-233

optimization, cost-based, 264-265

OR, 121-123

- avoiding, 262-263

Oracle

- date parts, 189-190
- EXTRACTVALUE, 334
- privileges, 290
- SELECT, 348-349
- Users, creating, 273-274
- for Windows, installing, 363-365

Oracle Fusion Middleware, 340

ORDER BY clause, 146, 361

- compound queries, 235-236
- versus GROUP BY clause, 150-153
- SELECT, 99-101
- views, 306-307

ordering

- column lists, 70
- join conditions, 257-258

outer joins, 201-204

P

parentheses, 122

- arithmetic operators, 130
- DISTINCT, 96

parsing, 264

performance
- cost-based optimization, 264-265
- database tuning, 254
  - versus SQL statement tuning, 254
- HAVING clause, avoiding, 263
- indexes, 250
  - disabling during batch loads, 264
- large sort operations, avoiding, 263
- LIKE, 261-262
- nested views, 308
- OR, avoiding, 262-263
- SQL statement tuning, 253
- stored procedures, 263-264
- subqueries, 225
- wildcards, 261-262

performance statistics, system catalog, 316

performance tools, 265

PL/SQL, 351-352

poor transactional control, 90

populating tables with data, 68

primary key constraints, 44-45

primary keys, 14

privileges, 284
- controlling with roles
  - CREATE ROLE, 292
  - DROP ROLE, 292
  - SET ROLE, 292-293
- controlling user access, 290-291
- granting/revoking, 287
- object privileges, 286
- Oracle, 290
- SQL Server, 290-291
- system privileges, 285

pseudocolumns, 184

PUBLIC database account, 289-290

Q

qualifying columns in queries, 199

queries
- case-sensitivity, 101-102
- compound queries. *See* compound queries
- constraints, 103
- SELECT, 93-94
  - FROM clause, 97
  - ORDER BY clause, 99-101
  - SELECT clause, 94-96
  - WHERE clause, 97-98
- single queries versus compound queries, 229-230
- subqueries. *See* subqueries
- writing, 102-103
  - column aliases, 105-106
  - counting records in tables, 103-104
  - selecting data from another table, 105

Query Analyzer, 265

querying system catalog, 317-320

quotation marks, 69, 161

R

RDBMS (relational database management system), 1
- collation, 101

READ ONLY, 89

READ WRITE, 89

readability, formatting SQL statements, 255-257

REAL, 22

records, 14
- counting in tables, 103-104

redundancy, data
- denormalization, 63-64
- logical database design, 57-58

REFERENCES, 286

REFERENCES(*column_name*), 286

referential integrity, 62-63

relational database management system (RDBMS), 1, 101

relational databases, 5

RELEASE SAVEPOINT, 89

remote databases, accessing, 340
- JDBC (Java Database Connectivity), 340
- ODBC (Open Database Connectivity), 340
- OLE DB, 340
- vendor connectivity products, 340-341
- web interfaces, 341

removing user access, 279

REPLACE, 166

RESTRICT option, 44
- REVOKE, 289

REVOKE, 279, 289, 359

revoking privileges, 287

roles, controlling privileges, 291

ROLLBACK, 360
- controlling transactions, 85-86

rollback area, transactions, 82-83

ROLLBACK TO SAVE, controlling transactions, 87-89

ROLLUP expression, grouping data, 153-155

rows, 14

tables, 36-37

RPAD, 171-172

RTRIM, 168

S

SAVEPOINT, 86, 360

savepoints, naming conventions, 87

schemas

creating, 275-276

database objects and, 33-35

dropping, 277

versus users, 271-272

second normal form, 59-61

security, 283-284

controlling privileges with roles, 291-292

CREATE ROLE, 292

DROP ROLE, 292

SET ROLE, 292-293

controlling user access, 287

GRANT, 287-288

on individual columns, 289

privileges, 290-291

PUBLIC database account, 289-290

REVOKE, 289

Internet, 343

intranets, 343

privileges, 284

views, 299

SELECT, 10, 286, 360

ANSI standard, 348

FROM clause, 97

DML (Data Manipulation Language), 67

keywords, 94

Oracle, 348-349

ORDER BY clause, 99-101

SELECT clause, 94-96

SQL Server, 348-349

subqueries, 215-217

WHERE clause, 97-98

SELECT ANY TABLE, 285

SELECT clause, 94-96

selected data, grouping (GROUP BY clause), 147

selecting data

from another user's table, 105

from multiple tables, 197

self joins, 204-206

sessions, SQL, 8

CONNECT, 8

DISCONNECT, 8

EXIT, 8

SET keyword, 75

SET ROLE, 292-293

SET TRANSACTION, 11, 89

simplifying data access with views, 298-299

single queries versus compound queries, 229-230

single quotation marks, 161

single-column indexes, 246

SOME, 118-120

SP_ADDUSER, 274

special characters, 161

SQL (Structured Query Language)

ANSI SQL, 2

direct SQL, 333

generating, 332-333

intranets and, 343-344

overview, 2

SQL clauses, 360

FROM, 360

GROUP BY, 361

HAVING, 361

ORDER BY, 361

SELECT, 360

WHERE, 361

SQL extensions, 349

SQL optimizer, 255

SQL Server

date parts, 189

installing, 365-367

privileges, 290-291

SELECT, 348

users, creating, 274

SQL sessions, 8

CONNECT, 8

DISCONNECT, 8

EXIT, 8

SQL statement tuning, 253

versus database tuning, 254

SQL statements

formatting, 254

arranging tables in FROM clause, 257

most restrictive condition, 258-259

ordering join conditions, 257-258

for readability, 255-257

interactive SQL statements, 353-354

SQL-99, 2

SQL-2011, 2-4

standards

- coding standards, 257
- SQL-2011, 3-4
- table-naming standards, 13

storage, date and time, 181

stored procedures, 263-264, 326-329

strings

- fixed-length strings, 21
- literal strings, 25-26
- varying-length strings, 21-22

subqueries

- correlated, 223-224
- DELETE statement, 218-219
- embedded, 219-223
- INSERT statement, 217
- overview, 213-215
- performance, 225
- SELECT, 215-217
- unknown values, 215
- UPDATE statement, 218

SUBSTR function, 164-165

substrings, 160

subtraction, 128-129

SUM, 136-138

summarized data, views, 299

synonyms, 308-309

- creating, 309
- dropping, 310

SYSDATE, 184

system catalog

- creation of, 314-315
- database design information, 316
- information, 315
- overview, 313-314
- performance statistics, 316
- querying, 317-320
- security information, 316
- tables by implementation, 316-317
- updating objects, 320
- user data, 316

T

table aliases, 200

table-naming standards, 13

tables, 35

- ALTER TABLE, 39
- arranging in FROM clause, 257
- base tables, joins, 207-208
- columns, 14, 35-36
  - modifying, 41-42
- counting records in, 103-104
- CREATE TABLE statement, 37-39
- creating
  - from existing tables, 42-44
  - from views, 305-306
- data
  - deleting, 75-76
  - inserting, 68-69
- dropping, 44
  - from views, 298
- fields, 14
- full table scans, 260-261
- inserting data from other tables, 70-72
- modifying
  - adding auto-incrementing columns, 40-41
  - adding mandatory columns, 40
- ALTER TABLE, 39
- elements of, 40
- naming conventions, -39
- NULL values, 14-15
- populating with data, 68
- primary keys, 14
- records, 14
- rows, 14, 36-37
- selecting data from multiple, 197
- system catalog, 316-317

temporal database support, 3

terminals, 6

testing WHERE clause, 258

TEXT, 22

third normal form, 61

TIME, 182

time, adding to dates, 186-187

time zones, 184-185

TIMESTAMP, 182

TKPROF, 265

tools, GUI tools, 272

transactions, 81

- controlling, 82-83
  - COMMIT, 83-84
  - RELEASE SAVEPOINT, 89
  - ROLLBACK, 85-86
  - ROLLBACK TO SAVEPOINT, 87-89
  - SAVEPOINT, 86
  - SET TRANSACTION, 89
- defined, 81-82
- poor transactional control, 90
- rollback area, 82-83

TRANSLATE, 160, 165-166
triggers, 329
  creating, 329-330
  dropping, 331-

U

UNION, 229-232
UNION ALL, 232-233
unique constraints, 46
unique indexes, 246-247
unknown values, subqueries, 215
UPDATE statement, 74, 286, 360
  subqueries, 218
UPDATE(*column_name*), 286
updating
  data, 73
    multiple columns, 74-75
    through views, 307
    value of single column, 74
  system catalog objects, 320
UPPER function, 162-163
USAGE, 286
user access
  controlling, 287
    GRANT, 287
    on individual columns, 289
    privileges, 290-291
    PUBLIC database account, 289-290
    REVOKE, 289
  removing, 279
user data, system catalog, 316
user management, 269-272
  GUI tools, 279
  schemas
    creating, 275-276
    dropping, 277
    versus users, 271-272
  types of users, 270
  user access, removing, 279
  user sessions, 278
  users
    altering, 277-278
    creating, 272
    creating in MySQL, 275
    creating in Oracle, 273-274
    creating in SQL Server, 274
user sessions, 278
user-defined data types, 27
users
  altering, 277-278
  creating, 272
    in MySQL, 275
    in Oracle, 273-274
    in SQL Server, 274
  versus schemas, 271-272
  types of, 270

V

varying-length strings, 21-22
vendor connectivity products, 340-341
vendors, database vendors, 7-8
view dependencies, 303
views, 297-298
  creating, 300
    WITH CHECK OPTION, 304-305
    from multiple tables, 302-303
    from a single table, 300-302
    tables from, 305-306
    from views, 303-304
  dropping, 307-308
  dropping tables, 298
  nested views, performance, 308
  ORDER BY clause, 306-307
  as security, 299
  simplifying data access, 298-299
  summarized data, 299
  updating data, 307

W

WAN (wide area network), 6
web interfaces, 341
web-based database systems, 6-7
WHERE clause, 74, 76, 97-98, 103
  testing, 258
wide area network (WAN), 6
wildcards, performance, 261-262
windowed table functions, 333-334
Windows
  Microsoft SQL Server, installing, 365-367
  Oracle, installing, 363-365

WITH CHECK OPTION, 304-305

writing

database queries, 102-103

column aliases, 105-106

selecting data from another table, 105

queries, counting records in tables, 103-104

X-Y-Z

XML, 334-335