

Preface



The idea for this book started when I taught a two-day introductory SQL course. The students were data analysts at a college that had recently converted its administrative information system to an Oracle database. While the students all had technical backgrounds in research techniques, few had any experience with SQL. Yet, to do their jobs, they needed to get information out of the database. Since SQL is ultimately the conduit into any Oracle database, I was there to get them started working productively.

I had a limited amount of time and a novice but motivated group of students. These factors forced me to ask, “What do these people *really* need to know to start doing useful work with SQL? What topics should I include in the few available hours that would be most helpful?” Questions like these require that you distinguish the truly important items from those of lesser importance. And once this is done, you’re left with a set of topics that require an unconventional organization to tie them together with some coherence.

I remember when I first started learning SQL. I’d set myself a very simple task. I wanted a report that showed the gender distribution of a group of people. It took me a week of reading and experimenting to get the report, a week spent writing SQL that didn’t work and writing more SQL to find out what was wrong with the earlier SQL. By the end of the week, I’d worked my way through an Oracle documentation manual and an SQL book I’d purchased. I didn’t find the key piece of missing information until I discovered correlated

subqueries while deep into the book. It turned out that in order to define the population for my report, I needed two correlated subqueries.

The realization that an advanced topic like correlated subqueries might be required to do what I considered to be a trivial report left a strong impression on me. Wasn't there some better way to structure SQL training materials so that they more closely reflected the actual practice of constructing queries? It was this organizing principle I used when developing materials for the SQL course I taught to the data analysts.

And it seemed to work well. So I began writing this book. I'd written about eight chapters when I had the opportunity to teach an SQL programming course for the Division of Continuing Education at the University of Colorado at Boulder. The challenge was again the same: how to provide highly motivated students with little or no SQL background with the most helpful ideas and techniques in the limited time available (18 hours).

While teaching the continuing education course over the year that followed, I experimented with different ways to present ideas, different figures to make the ideas clearer, and different exercises to strengthen the ideas. Make no mistake, SQL is not an easy topic. The book you see now, however, incorporates the experiences of many students.

Oracle SQL Primer is a teaching tool for use in training programs, in college classrooms, and by individuals going-it-alone. The intended audience is people who will actually use a database to help influence day-to-day operations or policy planning in an organization—people like end-users, data analysts, policy researchers, and business analysts. What connects these people is their need to query an Oracle database so they can retrieve useful information. People interested primarily in database administration will find much of value in this book but should be aware that it does not address the range of SQL issues such as database security that may confront them.

This book is also strictly an introduction. Hence the word *Primer*. You will not find discussions of topics like the development of complex data views that might interest information technology staff. Nor will you find discussions of topics on how to optimize query performance. These and other topics, while very important, would dilute the central focus of the book—teaching you how to query Oracle databases easily and accurately.

The book may be an introduction. But even experienced SQL programmers will find topics that instruct. For example, the emphasis on query populations

(the Who in a query) is unique among SQL books.

Another aspect that I think readers will find unique is the idea of SQL as a craft. In *The Sociological Imagination* by C. Wright Mills, you'll find a marvelous appendix entitled "On Intellectual Craftsmanship" that describes how the conduct of sociological research is really a craft that requires a certain mind-set and determination. It is more than just a collection of skills but is rather a way of thinking. It's the same for SQL too.

Programming of any kind is basically a creative act. For example, features like elegance are important in all programming. This concept, borrowed from mathematics, asserts that there is greater beauty in a simple, direct solution than in a lengthier, more convoluted one. In this sense, programming is an art, a craft that requires time and experience to mature.

But SQL seems to have a special place among programming languages. I don't think this has anything to do with its marketplace importance. Certainly SQL is absolutely essential for anyone working with relational databases. But there is something else about SQL that makes it a craft.

SQL requires a way of thinking that is unique and sometimes nonintuitive. I first realized this when I watched experienced programmers try to learn SQL for the first time. Like anyone learning a new task, even experienced programmers flounder in the beginning. That is to be expected. But there was still something strange about the SQL they wrote. Their queries didn't feel right. Sure, their table joins were often a frightful jumble, just like every other SQL beginner. But it was more than that.

Their queries started at the wrong place, implying that their entire orientation to SQL needed shifting. The SQL `SELECT` command is deceptively simple, consisting of just seven main clauses that are structured in a topdown orientation. You select the data columns from the tables you want, with the conditions that are necessary, and then display the results by sub-groups or in a specific sort order. Seems simple enough. Except that SQL queries don't work in a top-down way. Instead, you start in the middle with the `FROM` and `WHERE` clauses to define a query population. Then you go back to the beginning to identify data columns for the report. This may require additional tables in the `FROM` clause and additional `WHERE` logic to access those tables. So you write SQL in this strange way of starting in the middle, going back to the beginning, adding more to the middle, and then adding an ending.

SQL might be a "structured query language," but the structure doesn't come

from the clauses in the language; it comes from a way of thinking—a way of defining populations, accessing data about those populations, and displaying the data in a way that provides information about the population. SQL at its best is a craft, and this book tries to convey that sense of craft.

Several conventions are used in this book to provide consistency. SQL, SQL*Plus, and PL/SQL reserved words (e.g., the `SELECT` command) appear in **SMALL CAPITALS**. Whenever new concepts are first introduced, the defined words appear in *italics*. SQL programs and report results, which in earlier times all got printed on a line printer with monospaced fonts, appear in the `OCR B` font like `this`. Table names in the sample database (e.g., the `ADDRESS` table) are displayed in uppercase. The default format for dates is `DD-MON-YYYY`.

The database used in this book for exercises and examples concerns a fictitious college in upstate New York called Komenda College. All data were generated based on probability distributions, so they may feel real to you at times, but in fact the data are entirely simulated.

Finally, a word about my use of language. I frequently use the words *we*, *us*, *you*, and *I*. This was done deliberately. The easiest way to learn SQL is elbow-to-elbow with someone as you both work on a problem. You've got something that needs a solution. The motivation is high and the scope of the issue narrow. Learning takes place very effectively in this type of situation. While I can't create this one-to-one work setting in a book, I've written the book as if you and I are working together on a problem. Hopefully, this technique conveys some of the bonding between colleagues that happens when working together.

I hope you enjoy reading and working with this book as much as I enjoyed writing it.

Gary M. Lewis
Longmont, Colorado