Advanced SQL with SAS®

Christian FG Schendera

The correct bibliographic citation for this manual is as follows: Schendera, Christian FG. 2022. *Advanced SQL with SAS®.* Cary, NC: SAS Institute Inc.

Advanced SQL with SAS®

Copyright © 2022, SAS Institute Inc., Cary, NC, USA
ISBN 978-1-955977-90-6 (Hardcover)
ISBN 978-1-955977-87-6 (Paperback)
ISBN 978-1-955977-88-3 (Web PDF)
ISBN 978-1-955977-89-0 (EPUB)

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

May 2022

Contents

Preface*

SQL (Structured Query Language) is *the* programming language for the definition, query, and manipulation of very large data sets in relational databases. Worldwide. SQL is a quasi-industry standard. In June 2010, a web search for the term “SQL” with Google resulted in 135 million hits. In February 2021, the same Google search found 307 million hits. SQL has been a standard at ANSI (American National Standards Institute) since 1986 and at ISO since 1987. With PROC SQL, it is possible to query data from tables or views; create tables, views, new variables, and indexes; merge data from tables or views; create summary parameters or statistics; perform complex calculations (new variables); update values in SAS tables; or create and design reports with the Output Delivery System (ODS) (Schendera, 2011).

This volume shows what else PROC SQL can do. There are areas where PROC SQL goes far beyond the ANSI standard, for example:

- PROC SQL handles **missing values** differently than the ANSI standard. While entries represent the presence of information, missing values indicate the opposite, the absence of information. Chapter 2 presents the handling of missing values: defining (system- and user-defined), querying, as well as searching and replacing alphanumerical missing values.
- Automatic **data integrity checks** using integrity constraints and audit trails, as well as multiple value checks, outliers, filters, and unification of undesired strings (Chapter 3).
- PROC SQL also supports the use of the **SAS macro language**. This difference is of such importance that Chapter 4, the most extensive chapter, is dedicated to the interaction between SQL and the SAS Macro Facility. This chapter presents how to automate and accelerate processes using macro variables (automatic, user-defined) and macro programs including the listwise execution of commands and the use of loops.

There are some other “**SAS extras”** that go beyond the ANSI standard. However, no separate chapter covers them. These include countless SAS functions and function calls (see the overview in Subsection 10.2.), and other functionalities like CALCULATED, Boolean expressions, or remerging.

Chapters 6 through 9 extend PROC SQL focusing on **performance and efficiency**, which are covered from multiple angles: programming languages (hash objects, FedSQL, and DS2), programming environments (client/server, grid, or cloud), or special techniques (for example, in-memory or distributed processing). **SAS syntax** for PROC SQL, **SAS functions and function calls**, and **DBMS accesses** are covered in Chapter 10. PROC SQL supports many more evaluation functions than required by the ANSI standard. (See Chapter 10.)

In addition to functionality and programming, Base SAS users might need to get used to a slightly different **terminology.** (See Table 0.1.) In Base SAS, for example, a data file is called a SAS file, a data line is called an observation, and a data column is called a variable. In PROC SQL, however, a data file is called a table, a data row is called a row, and a data column is called a column. The names in RDBMS are again slightly different. Other models or modeling languages (ERM, UML) use different terminology.

Table 0.1 Terminology in PROC SQL, Base SAS, and RDBMS

Structure/Language	SQL	Base SAS	RDBMS
Data File	Table	SAS Data Set	Relation
Data Row	Row	Observation / Record	Tuple / Record
Data Column	Column	Variable	Attribute / Field
Data Cell	Value	Value	(Attribute) Value

Base SAS and PROC SQL elements are often used simultaneously in the same SAS program as this book will show in many places. Experience has shown that a strict terminological **differentiation** is therefore unnecessary. Although this book prefers SQL terminology, it will also use Base SAS terminology. This book therefore uses the terms "table", "SAS file", or "data set" interchangeably for a data set. In contrast to the RDBMS term, "data set" refers to the totality of all rows (even if a file should only contain one row).

PROC SQL creates empty tables, views, or "only" performs queries. What is the difference between tables, views, and queries?

- A **table** is nothing more than a SAS data set. SAS tables are often a consequence of queries. A SAS file (a) describes data, (b) contains data, and (c) requires storage space. The last two points are important; they serve to delimit views.
- A **view** (a) describes data, (b) contains no data itself, and therefore (c) does not require any storage space. SAS views are only a view on data or results, without having to physically contain the data itself. Views and tables have in common that they are each a result of a query or a sequence of queries. Tables and views could also be rewritten as "stored queries," for example, as the result of a SELECT statement or more complex queries.
- **Queries** are a collection of conditions (for example, "take all data from table A", "query only rows with a value in column X greater than 5 from view B", and so on). Queries in their basic form only query data and write their result immediately to the SAS output. Depending on the programming (including the addition of CREATE, AS, and QUIT), queries can output the result of their query in (a) tables, (b) views, (c) directly to the SAS output, or (d) to SAS macro variables. Queries can also contain one or more subqueries. A subquery is a query (in parentheses) that is embedded within another query. If there are several subqueries, the innermost one is executed first followed by all the others in the direction of the outermost query.

All concepts assume that a data row, observation, or tuple must be uniquely identifiable by at least one so-called **key** (also called a primary key or ID). A key is generally unique and thus uniquely identifies a row in a table, not its relative position in the table, which can always change by sorting. A key is generally used less for describing a data row than for managing it. A *primary* key is usually used to identify the rows in your own table, a *foreign* key is used to identify the rows in another table, usually its primary key. Keys are essential for joining two or more tables and therefore must not be deleted or changed.

These remarks so far about joins, scenarios, and keys were mainly concerned with data in the sense of valid or *existing* values. A whole chapter of this book is dedicated to the absence of information, the handling of *missing* values (also called NULL values, NULL, or missings). In contrast to the ANSI standard, the handling of **missing values** is based on a different definition. In the ANSI standard, expressions such as "5 > NULL", "0 > NULL" or "-5 > NULL" become *NULL*. In Boolean and comparison **operators**, however, these and other expressions become **true**, and this is also true when working with SAS.

About This Book

This book provides an in-depth look at working with SQL from SAS. SQL is, especially in the SAS variant, a very powerful programming language. This book presents the **topics** of missing values, data quality including integrity constraints and audit trails, macro variables and programs, the interplay of PROC SQL with geodata and even spherical distances, performance and efficiency, alternative programming languages, numerous other aids, tips and tricks, or the use of the numerous SAS functions and function calls. From time to time, users are explicitly warned of potentially **undesirable** results. Undesirable results when sorting, grouping, or joining tables with missing values often occur when one does not **know** how PROC SQL or SAS work in detail.

This book is intended as an in-depth consolidation and extension of ANSI SQL and is intended for advanced users. The chapters are organized so that they each illustrate different facets of programming and using PROC SQL. By the end of this book, you should be an advanced user of PROC SQL, capable of thinking conceptually beyond SQL topics and implementing data quality, performance, and automation including integrity constraints, performance (tuning), and programming using the SAS Macro Facility, hash objects, or FedSQL. You know what missing values are and can deal with missing values in queries, analyses, and joins. In addition to missing values, you know other criteria for data quality and SQL techniques to ensure it, including integrity constraints, audit trails, SAS functions, and queries. You will be able to accelerate and automate PROC SQL functionalities using macro variables and macro programs. You would also be no stranger to handling coordinate data using PROC SQL. If you need to handle very large amounts of data, you can also "translate" PROC SQL functionalities into alternative programming approaches using hash objects, FedSQL, or DS2 programs. Also, you know the respective advantages of these programming languages, not limited to the CAS environment. You can distinguish between performance and efficiency and know numerous options to speed up processing from programming languages to programming environments, as well as fundamental and SQL-specific processing optimization approaches. You will also know the benefits of SAS dictionaries and PROC SQL over the DATA step when updating tables with the UPDATE statement. Last but not least, the SAS functions and function calls will give you another impression of the power of SAS. Nevertheless, SAS can do so much more.

Many other exciting SQL topics had to be left out of the book due to lack of space and time, including the interaction of PROC SQL with SAS Studio, Enterprise Guide, or JMP, the strategic use of PROC SQL in Business Intelligence, the preparation of statistical data for complex statistical analyses using Enterprise Miner, or dynamic visualizations using JMP or other SAS applications, and so much more.

What Is SAS, for Me?

When I write about SAS, I like to start by saying that I have been working with SAS for more than 30 years. (See also Schendera, 2012/2011, 2004.) My focus is on Advanced Analytics and Business Intelligence with SAS. Many of the chapters and programs were written during long evenings, sometimes over weekends completely snowed in at a hotel, while preparing projects from all over Europe. Now, at the end of the last edits to this book, I see again and again and still with certain fascination, how many facets the use of PROC SQL from SAS can have. But **what is SAS** in the first place, personally speaking? SAS offers not just a wide range of products and services for almost every industry, role, or requirement. My personal answer to the question "What is SAS?" is fivefold. SAS is a product system, a company, a philosophy, a sophistication and an enthusiasm. Knowledge is power. Precise and timely knowledge is essential for information in performance and competition. SAS is *for* "The Power to Know." For the many solutions SAS offers, please refer to the SAS website. Readers can find information about topics, such as architecture, cloud computing, ETL, or data integration, on www.sas.com and through discussion groups (SAS-L, sasCommunity.org), a visit to one of the numerous SAS events (SAS Global Forum or local SAS Forums), or more specialized events (SAS@TDWI, SAS Government Executive Conference, or PhUSE).

*This is an updated version of the original 2011 preface.

Acknowledgments

Ten years ago, I published a **double volume** on SQL, a beginner's volume and an advanced volume. First of all, I would like to thank you for the overall positive reception and feedback on the German versions (Schendera, 2012/2011). Through writing these SAS books with heart and soul, I also made a great effort to make and keep them **relevant**. Consequently, this edition now contains a chapter introducing Cloud Analytic Services (CAS) and FedSQL (Chapter 7). Several chapters have been updated, including analysis of geodata (Chapter 5), programming with hash objects (Chapter 6, especially the section about fuzzy joins), and performance and efficiency (Chapter 8). I also updated the SAS functionalities, functions, and syntax to SAS 9.4. Some topics are fundamental as well as they are timeless. On the subject of missing values, for example, that miserable pitfall of programming and analysis, I managed to write 35 pages. I am fascinated again and again by the negative consequences empty data cells can have, and sometimes maybe even more about how clueless users sometimes are about this problem.

I have been working with SAS for over 30 years now (see also Schendera, 2004), but I still learn something new every day. If there's one thing I can say, it's that SAS is growing faster than you can ever learn it. When I was studying psychology at Heidelberg University, I quickly realized how important it is that you **understand exactly** where the data come from that you use for analysis and research. I had to know how SAS worked. I enrolled in SAS courses on main frames, still using punch cards. It was a plague! It took me two attempts to get my certificate. But then things took off. SAS got me hooked, and I developed an enthusiasm for SAS that sometimes kept me up all night. Although I wouldn't have called it work; it was, well, my passion. Which I would like to share with you and motivate you to get to know SAS—its power and versatility. Maybe to use it as a tool for good and make this world a better place.

I am grateful for advice, inspiration, or simply friendship to Prof.em. Gerd Antos (Martin-Luther-University Halle/S., Germany), Prof.em. Mark Galliker (University Bern, Switzerland), Prof. Kirk Lafler (San Diego State University), Roland Donalies (SAS Germany), Markus Grau (SAS Switzerland), Prof. Maria ReGester (Jacksonville, NC), Frank Ivis (CIHI, Canada), Suzanne Morgen and Siân Roberts (SAS Press), and Dr. Stefan Giesen (De Gruyter, Munich). I also want to thank the reviewers for their time and valuable comments: Lewis Church, Catherine Connolly, Vincent DelGobbo, Paul Grant, Mark Jordan, Suzanne Morgen, and Ross Richards. I want to especially highlight and appreciate Suzanne's efforts and contributions. Without you, this volume would not be on the level that it is now. I cannot thank you enough.

I translated and updated both volumes under rather challenging conditions. It is therefore a matter of the heart for me to dedicate this edition to **Captain Sir Tom Moore**. As a 99-year-old, he initially set out to raise £1,000 for NHS charities by walking 25m laps around his garden with the help of his walker. In the end, Capt. Sir Tom raised almost £33m ($45m) for NHS charities. His always positive role model and his "Tomorrow will be a good day" lifted people's spirits during the COVID-19 pandemic and beyond. My favorite quote of his is "Don't give in, just keep on going and things will certainly get better. That's the way to look at it." Capt. Sir Tom passed away in February 2021. The White House joined the United Kingdom and the world in honoring the memory of Captain Sir Tom Moore "who inspired millions through his life and his actions."

Again, I would like to thank Sigur Ros for their timeless artistic inspiration. I wish I could write as well as you can compose and create aural landscapes. Last, but not least, I thank my wife Xiao Yun Huang and my German and Chinese families and friends all over the world. If anything in this book should be still not clear or even incorrect, the responsibility lies solely with me.

Hergiswil NW, Switzerland
January 2022

Dr. CFG Schendera

Acknowledgments for the 2012 Edition

At this point I would like to take the opportunity to thank all those who have supported me in writing this book. First of all, I would like to thank you for the numerous feedback on "SQL with SAS: PROC SQL for Beginners" (Schendera, 2011), as well as "Data Management and Data Analysis with the SAS System" (Schendera, 2004). I was happy to implement the suggestion to include hash programming as an alternative to SQL.

I am grateful for i.a. technical advice and/or contribution in the form of syntax, data and/or documentation: Prof. Gerd Antos (Martin-Luther-University Halle/S., Germany), Prof. Wolfgang Auhagen (Martin-Luther-University Halle/S., Germany), Prof. Mark Galliker (University Bern, Switzerland), Prof. Jürgen Bock, Frank Ivis (Canadian Institute for Health Information, Toronto, Ontario, Canada), Kirk Lafler, Prof. Rainer Schlittgen, Mike Whitcher, as well as the exchange on SAS-L: e.g. Richard R. Allen, Michael Bramley, Dave Brewer, Nancy Brucken, David Carr, David Cassell, Laurel A. Copeland, Peter Crawford, Richard DeVenezia, Paul M. Dorfman, Harry "Bill" Droogendyk, Ron Fehd, Y. (Jim) Groeneveld, Gerhard Hellriegel, Sigurd Hermansen, Eric Hoogenboom, Ya Huang, Nick Longford, Gene Maguin, Dale McLerran, Lawrence H. Muhlbaier, Girish S. Patel, Prasad S. Ravi, Kattamuri Sarma, Howard Schreier, Karsten M. Self, Daniel Sharp, Erik Tilanus, Michel Vaillant, Kevin Viel, Ian Whitlock, John Whittington, Andre Wielki, Matthew M. Zack und last but not least the heroes of First Level Support at SAS in Heidelberg, Germany and Cary, NC.

My thanks go also to Wilhelm Petersmann and Markus Grau of SAS Switzerland (Wallisellen) for the generous provision of SAS software and technical documentation. A very special thanks goes to Roland Donalies (SAS Germany). I would also like to thank Mrs. Stephanie Walter, Mr. Thomas Ammon, and Mrs. Cornelia Horn of Oldenbourg Publishers (now: DeGruyter) for their confidence in also publishing Volume II and for their always generous support. My sincere thanks also go to Dr. Schechler and Kristin Beck. Volker Stehle (Mannheim) designed the artwork. I would like to thank Sigur Ros for their many years of artistic inspiration. If anything in this book should be unclear or incorrect, the responsibility lies solely with the author.

Hergiswil NW, Switzerland
August 2011

Dr. CFG Schendera

About This Book

What Does This Book Cover?

Structured Query Language (SQL) is the most widely used programming language and a quasi-industry standard. SQL is a standardized language that retrieves data from and updates data in tables and the views that are based on those tables. The SQL procedure implements SQL for SAS.

This book introduces the specifics of PROC SQL. Based on the premise that PROC SQL is developed for and executed in SAS, this book's mission is to present advanced SQL use. You will learn how to take advantage of the extra possibilities the power of SAS offers including:

- Essential topics like missing values and data quality with audit trails
- "Blind spots" like how missing values can affect even the simplest calculations and table joins
- SAS macro language and SAS macro programs
- SAS functions
- Integrity constraints
- SAS Dictionaries
- SAS Compute Server
- FedSQL on CAS

In addition to numerous tuning techniques, this book also touches on implicit and explicit pass-throughs, presents alternative SAS grid- and cloud-based processing environments, and compares SAS programming languages and approaches including FedSQL, CAS, DS2, and hash programming. Comparisons between SQL and FedSQL and overviews of SQL syntax and SAS functions help you see the possibilities at your disposal.

This book does not cover SQL basics like SQL logic, terminology, or how SQL works. It does not cover queries or joins, comparisons in programming and processing between Base SAS and PROC SQL, or calculating descriptive statistics, percentages, or working with weights. These fundamentals are specifically taught in a stand-alone volume that I wrote which is designed without SAS specifics and can be applied to systems other than SAS. Currently, the fundamental volume is only available in German.

This edition contains a chapter on FedSQL, which was not included in the "advanced" volume of the original German edition, the most comprehensive double-volume about PROC SQL worldwide at that time. The FedSQL part was written especially for this SAS Press edition. The chapter on geodata analysis has been updated and completely rewritten.

Is This Book for You?

This book is for SAS programmers, analysts, statisticians, and students who want to expand their SQL know-how. This book is also for users of other SQL variants who want to learn about the advantages of PROC SQL, the magic of macro programming, handle missing values, move from SQL to FedSQL, strive for more performance, or find alternatives to SQL and FedSQL.

Although the book addresses advanced topics, it is designed to progress from the simple and manageable to the complex and sophisticated. Some basic SQL and SAS programming skills might be helpful. You need to have access to licensed SAS software like SAS 9.4 or SAS Viya. You can also use SAS OnDemand for Academics or SAS Viya for Learners for some of the examples in this book.

What Should You Know about the Examples?

This book uses hundreds of SAS programs as examples, be it SQL, FedSQL, DS2, or others. The examples are chosen to cover hands-on needs like showing step-by-step how small adjustments to simple SQL programs turn them into powerfully accelerated SAS macro programs. All examples have been tested and are explained, sometimes code line by code line. Feedback from the SAS log or output is also explained, providing hundreds of explanations throughout the book. In more advanced programs, explanations detail even subtle differences in the structures of the generated tables. Some instances use “learning by mistake”, in which mistakes are made on purpose to demonstrate which feedback a user might find in the SAS output and log, if any, and train the eye where to look at the result generated.

The chapters follow principles of instructional/text psychology. Typically, an Advance Organizer precedes the chapter, then the contents=examples are usually arranged from the fundamental and simple to the sophisticated and advanced topics. This enables readers=users to have an early understanding and make programming easy, wins=successes.

Software Used to Develop the Book’s Content

Platforms: SAS 9.4; SAS Viya 3.5 plus Compute Server (SPRE), CAS.

Programming Environments: SAS Enterprise Guide 8.2, SAS 9.4 Editor Window; CAS WebApps: SAS Studio 5.2

Example Code and Data

With very few exceptions, this book uses SAS data sets from SASHELP to support self-study. A few exceptions are tiny data sets of a few rows especially tailored to explain specific SAS topics.

The list below compiles the most commonly used SAS data tables. Data is mainly from the SAS folder SASHELP and should be available in SAS when fully installed. The following short list does not contain dictionaries, views, and so on. The SAS tables are sorted alphabetically.

```
SASHELP.AIR
SASHELP.BUY
SASHELP.CLASS
SASHELP.ORSALES                               FREQUENTFLYERS
SASHELP.PRDSALE
SASHELP.PRDSAL2
SASHELP.PRDSAL3
SASHELP.SHOES
SASHELP.SYR1001
SASHELP.ZIPCODE
```

SAS OnDemand for Academics

This book is compatible with SAS OnDemand for Academics. If you are using SAS OnDemand for Academics, then begin here: https://www.sas.com/en_us/software/on-demand-for-academics.html.

You can use the book’s contents without major adjustments to the example code and data to run in SAS OnDemand for Academics. Further functionality depends on the scope of SAS ODA. At the moment, SAS ODA does not support CAS processing of PROC FEDSQL as described in Chapter 7.

We Want to Hear from You

SAS Press books are written *by* SAS Users *for* SAS Users. We welcome your participation in their development and your feedback on SAS Press books that you are using. Please visit sas.com/books to do the following:

- Sign up to review a book
- Recommend a topic
- Request information on how to become a SAS Press author
- Provide feedback on a book

Learn more about this author by visiting his author page at http://support.sas.com/schendera. There you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more.

Author Feedback

Your feedback is encouraged, valued, and appreciated. If you have any suggestions for additions or improvements to this book, I would like to ask you to send your suggestions by email to the following address:

SAS_v2@method-consult.ch

Please use the keyword “Feedback SAS book” as the “Subject” and include at least the following information:

1. edition
2. page
3. keyword (for example, “typo”)
4. description (for example, “for statistical analysis”). Please comment the program code.

About The Author

Dr. Christian FG Schendera is a Senior SAS Data Scientist and managing director at Method Consult in Switzerland. An avid SAS user for 30+ years, his experience ranges from scientific consulting, project management, and feature-engineering to statistical modeling using SAS. He studied at Heidelberg University and Martin Luther University Halle-Wittenberg. While still a student, he started consulting in statistics and research methods, lecturing on SAS, applied statistics, and the knowledge-shaping role of non-scientific methods. Christian views constructing knowledge as a two-way street: every step from data collection to applying methods will affect results. He has published several books about statistics, data quality, programming, and SAS. He is author of the most comprehensive double-volume about PROC SQL worldwide. Further information and downloads can be found at www.method-consult.ch.

Learn more about this author by visiting his author page at http://support.sas.com/schendera. There you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more.

Chapter 1: Overview

This book is written for **advanced** users in PROC SQL. The handling of missing values (null values) in PROC SQL is an important topic since SAS SQL handles missing values differently from the ANSI standard. Failure to take these features into account can lead to potentially undesirable results when dealing with missing values. Other topics in this book include data quality, especially with integrity constraints, as well as special features in dealing with missing values and visual analysis of geodata and distances.

Three chapters in this book are particularly important as they help you to harness the power of SAS. The chapter on macro programming, for example, describes how the listwise execution of commands can speed up work with SAS many times over, both when programming and also executing programs. Two further chapters introduce two programming alternatives: programming with hash objects, as well as FedSQL and its possibilities to put it into practice using it in the procedures **FEDSQL**, CAS and DS2. The chapter on performance and efficiency compiles various possibilities of how to obtain even more performance, especially when handling large amounts of data. Further chapters and sections are reserved for overviews of SQL syntax, SAS functions and routines, as well as various special features of the SAS Pass-Through Facility for selected DBMS accesses. Because this book focuses on using many specific features in SAS, it is written primarily for SAS users, as well as anybody interested in gaining a deeper insight in the power of SAS.

This book also introduces a new processing platform, **Cloud Analytics Services** (CAS). Section 1.2 compares several programming languages and their advantages especially in the CAS environment. For details about when to use FedSQL and when to use SQL, please see Section 7.5.

1.1 Detailed Description of This Book

Table 1.1 Quick Finder

Chapter	Description
1	Overview of this book
2	Missing values: Definition, mode of operation, and conversion
3	Data quality: Integrity constraints (test rules) and audit trails, as well as finding duplicates and outliers.
4	Macro programming: From simple macro variables (SAS, SQL, user-defined), to helpful SAS macro programs including macros for list execution of commands or macros for retrieving system information
5	SQL for geodata, distances, and maps
6	Hashing as an alternative to SQL
7	FedSQL
8	Performance and efficiency: strategy and measures including narrowing, shortening, and compressing; sorting; and other tips and tricks including SAS dictionaries
9	Help, tips, and tricks
10	SAS syntax - PROC SQL, SAS functions, and SAS routines
11	References
12	Syntax index
13	Subject index

Chapter 1 is an *overview* and presents in a short summary the contents of this book.

Chapter 2 deals with the subject of **missing values**. While entries represent the presence of information, missing values indicate the opposite, the absence of information. In principle, missing values are unpleasant because they restrict the basis for deriving information. System-defined missing values are tricky because a user does not know why this data is missing. In the case of user-defined missing values, the user knows, but the data is still missing. What makes matters more difficult is that PROC SQL handles missing values differently than the ANSI standard. This chapter focuses on basic aspects of how PROC SQL handles missing data. These include: the *definition* of missing values, the *retrieval* of data from tables that contain missing values (Section 2.2), possibly undesirable effects of missing values on operations of *data analysis* and *data management* (Sections 2.3, 2.4, and 2.5), and fundamental *measures* for dealing with missing values (Section 2.6).

Chapter 3 introduces the topic of **data quality**. Data quality comes before analysis quality. (See Schendera, 2020/2007.) This chapter introduces SQL techniques for ensuring data quality in the problem areas of outliers, plausibility (Section 3.3), missing values (Section 3.1), uniformity (Section 3.4), and duplicates (Section 3.2). By integrating filters when accessing data via SQL, users can ensure that data is not entered incorrectly into the system or analysis in the first place. This chapter also introduces working with integrity constraints and audit trails (Section 3.1).

Chapter 4 introduces **macro variables** and **macro programs**. Using SAS macros, the scope of PROC SQL can be easily extended. The focus of this chapter is less on the introduction of programming SAS macros, but rather on their uncomplicated *application*. This chapter presents numerous examples of SAS macros in application-oriented sections, which extend the possibilities of working with PROC SQL, simplifying it through automation, and accelerating it. Because macros are based on SAS syntax (including PROC SQL), macros also embody all the advantages of syntax programming (see Schendera, 2005, 147ff), for example, validation, automation and reusability, speed, openness, clarity, and systematization. SAS syntax programming can be automated and used repeatedly with macros without the need to write or adapt new syntax commands in principle. This chapter is divided into four sections. Section 4.1 introduces *SAS macro variables*. Section 4.2 introduces programming *SAS macro programs*. Section 4.3 introduces the most important *elements of the SAS macro language*. Section 4.4 and all subsequent sections introduce interesting *SAS macro programs for special applications*.

Chapter 5 introduces the **analysis of geodata** with PROC SQL. The calculation of distances and related parameters such as time or costs is a common task. This section presents the calculation of **geographical distances** in two-dimensional and spherical space using different formats of coordinates (Sections 5.1 and 5.2). In addition, you will learn to calculate the shortest, fastest, or even cheapest distance. Further sections discuss projections, visualizations (Section 5.3), and so on.

Chapter 6 introduces the programming of PROC SQL or DATA step functionalities by means of **hash programming**. Two objects, the hash object and hash iterator, are introduced into the DATA step. These objects enable you to store, search, and query data from lookup tables. Since SAS 9.1 it has been possible to program hash objects specifically. Hash programs can be executed within a DATA step, the DS2 step (see also Section 7.3 and Table 1.2) and the FCMP procedure. One of the most important features of a hash object is that it resides completely in the physical memory of the DATA step. There are two reasons why a separate chapter is dedicated to hash programming: performance, especially with very large data volumes, and introducing terms and language elements for the DS2 section in the FedSQL chapter.

Chapter 7 introduces **FedSQL**. FedSQL is the next-level SQL. It is a vendor-neutral SQL dialect that offers a scalable, threaded, high-performance way to access, manage, analyze, and share nonrelational data in multiple data sources. FedSQL (Section 7.1) can perform federated queries (connect to multiple sources in one query); work with ANSI-1999 compliant data sources such as Google, Amazon, and Salesforce; use new ANSI data types; and work in a cloud environment like the SAS Cloud Analytics Services (CAS) framework. You can gradually expand your programming skills by moving from PROC SQL to PROC FEDSQL (Section 7.2), and from there to PROC CAS and also PROC DS2 (Section 7.3). Several overviews describe similar and different FedSQL functionalities compared to SQL and when executed on SAS versus CAS. Numerous examples illustrate where you can use PROC SQL and PROC FEDSQL and on which platforms (SAS 9.4, SAS Viya, and CAS).

Chapter 8 introduces the topics of performance and efficiency, especially in the context of programming with PROC SQL. **Performance** refers to the performance of users or systems. **Efficiency** describes the performance of users or systems taking into account investment, environmental factors, or sustainability. Therefore, *performant* is when the required amount of data is optimally processed in a minimal amount of time. On the other hand, *efficient* is when the required amount of data is optimally processed using a reasonable effort (time and money). Efficiency is therefore a measure of performance, taking into account the necessary costs. Not everything that is programmed quickly is also efficient in the sense that the system is able to process the program with high performance. Conversely, developing performant programs can be so costly that the cost of manpower is disproportionate to the performance gained on the system side. Next to techniques like reducing, shortening, or tuning (Sections 8.2–8.5), this chapter also touches on the specifics of the Implicit and Explicit Pass-Through (Section 8.6, and see also Section 10.3), and presents alternative grid- and cloud-based processing environments like SAS Grid Computing and SAS Cloud Analytic Services (Section 8.8).

Chapter 9 introduces a chapter full of further **help, tips,** and **hints** for various aspects of working with PROC SQL. Section 9.2, for example, introduces working with SAS dictionaries.

Chapter 10 primarily provides **SAS syntax** for PROC SQL, SAS functions and routines, and DBMS accesses. Section 10.3 features an in-depth discussion of the Implicit and Explicit Pass-Throughs as complementary approaches.

Chapters 11 to 13 contain **references** and the indexes for SAS **syntax** and **keywords**.

By the end of this book, you should know the advanced features and possibilities using PROC SQL, PROC FEDSQL, or FEDSQL in PROC CAS or DS2. You will know when to use which SAS language when it comes to macro or complex programming, multi-threading, or in-memory processing. From a more high-level point of view, you will have learned the basics of data quality and fundamentals in performance and efficiency, illustrated by SQL, but easily transferable to the other programming languages. By the end of this book, you should also understand that this is only an introduction. There is still so much more. Welcome to SAS.

1.2 Which SAS Language for the CAS Environment?

This book introduces several programming languages related to SQL and also a new processing platform, CAS. Table 1.2 aims to make clear from the outset which languages and language elements are suitable for CAS, where they can be found in this book, and what their strengths and weaknesses are.

Table 1.2 Programming Languages for CAS

CAS Support For …	DATA Step	PROC SQL	FEDSQL (7.1-7.3)	DS2 (7.3)
Formats	●		●	●
In-Memory Processing (CAS)	●		●	●
Multi-Threading (SMP, MMP)	●		●	●
Precision (new integer types)			●	●
Pass-through (implicit, explicit)			●	●
Hash Objects (see Chapter 6)	●			
Unlimited use of SAS Macro Facility (see Chapter 4)	●			
Advanced package and methods development				●
Compatibility with special SAS hosting platforms				●
SAS Data Quality functions (see Chapter 10)	●			
BY-group processing using FIRST. and LAST.	●			●

Note: The **PROC SQL** column is **empty.** Table 1.2 highlights the fact that PROC SQL is not supported on CAS. On CAS, you can use SQL language elements, but not the procedure. Please see also Table 7.5-1 for details about when to use PROC FEDSQL and when to use PROC SQL.

PROC SQL programming expertise is a good starting point into possibilities of CAS by putting FedSQL to practice. For example, you can also use FedSQL statements in a DS2 program, which is another easy way to take advantage of the new integer data types.

I extended an overview originally provided by SAS Institute on how to choose the appropriate programming language for CAS by adding some more features and also a column for PROC SQL. Chapters 6 and 7 will discuss these programming languages (DS2, Hash Objects, FedSQL) from several angles.

Chapter 4 highlights some SAS enhancements of PROC SQL, which to date cannot be put into practice in the CAS environment.