Contents

About This Book ................................................................................................................. xiii

PART 1   SQL Processing with SAS  1

Chapter 1 • Performing Queries Using PROC SQL .................................................. 3
  Overview ......................................................................................................................... 4
  PROC SQL Basics ........................................................................................................ 4
  Writing a PROC SQL Step ............................................................................................ 6
  Selecting Columns ......................................................................................................... 8
  Specifying the Table ..................................................................................................... 10
  Specifying Subsetting Criteria ...................................................................................... 11
  Ordering Rows ............................................................................................................... 11
  Querying Multiple Tables ............................................................................................. 13
  Summarizing Groups of Data ...................................................................................... 17
  Creating Output Tables ............................................................................................... 19
  Additional Features ...................................................................................................... 20
  Summary ....................................................................................................................... 20
  Quiz ............................................................................................................................... 22

Chapter 2 • Performing Advanced Queries Using PROC SQL ............................. 25
  Overview ......................................................................................................................... 26
  Viewing SELECT Statement Syntax ........................................................................... 27
  Displaying All Columns ............................................................................................... 28
  Limiting the Number of Rows Displayed .................................................................. 29
  Eliminating Duplicate Rows from Output ............................................................... 31
  Subsetting Rows By Using Conditional Operators ................................................... 32
  Subsetting Rows By Using Calculated Values ................................................................ 40
  Enhancing Query Output ............................................................................................. 42
  Summarizing and Grouping Data ................................................................................ 48
  Subsetting Data By Using Subqueries ....................................................................... 61
  Subsetting Data By Using Noncorrelated Subqueries ............................................... 63
  Subsetting Data By Using Correlated Subqueries .................................................... 69
  Validating Query Syntax ............................................................................................. 71
  Additional Features ...................................................................................................... 72
  Summary ....................................................................................................................... 73
  Quiz ............................................................................................................................... 76

Chapter 3 • Combining Tables Horizontally Using PROC SQL ......................... 81
  Overview ......................................................................................................................... 82
  Understanding Joins ..................................................................................................... 82
  Generating a Cartesian Product ................................................................................... 83
  Using Inner Joins .......................................................................................................... 85
  Using Outer Joins ......................................................................................................... 93
  Creating an Inner Join with Outer Join-Style Syntax ................................................ 100
  Comparing SQL Joins and DATA Step Match-Merges ............................................. 100
  Using In-Line Views .................................................................................................... 105
  Joining Multiple Tables and Views ............................................................................. 109
Chapter 9 • Introducing Macro Variables ................................................................. 287
  Overview ........................................................................................................... 288
  Basic Concepts ................................................................................................. 289
  Using Automatic Macro Variables .................................................................... 291
  Using User-Defined Macro Variables ............................................................... 293
  Processing Macro Variables .............................................................................. 296
  Displaying Macro Variable Values in the SAS Log ............................................ 299
  Using Macro Functions to Mask Special Characters ......................................... 302
  Using Macro Functions to Manipulate Character Strings ............................... 306
  Using SAS Functions with Macro Variables ..................................................... 314
  Combining Macro Variable References with Text ............................................. 316
  Summary ........................................................................................................... 320
  Quiz .................................................................................................................. 323

Chapter 10 • Processing Macro Variables at Execution Time ............................... 327
  Overview ........................................................................................................... 328
  Creating a Macro Variable during DATA Step Execution ............................... 329
  Creating Multiple Macro Variables during DATA Step Execution .................. 343
  Referencing Macro Variables Indirectly ......................................................... 346
  Obtaining Macro Variable Values during DATA Step Execution .................... 352
  Creating Macro Variables during PROC SQL Step Execution ....................... 354
  Working with PROC SQL Views ..................................................................... 361
  Using Macro Variables in SCL Programs .......................................................... 362
  Summary ........................................................................................................... 364
  Quiz .................................................................................................................. 367

Chapter 11 • Creating and Using Macro Programs ............................................. 371
  Overview ........................................................................................................... 372
  Basic Concepts ................................................................................................. 373
  Developing and Debugging Macros .................................................................. 378
  Using Macro Parameters .................................................................................. 381
  Understanding Symbol Tables ......................................................................... 387
  Processing Statements Conditionally ............................................................... 396
  Processing Statements Iteratively .................................................................... 407
  Using Arithmetic and Logical Expressions ...................................................... 411
  Summary ........................................................................................................... 414
  Quiz .................................................................................................................. 417

Chapter 12 • Storing Macro Programs ................................................................. 421
  Overview ........................................................................................................... 422
  Understanding Session-Compiled Macros ....................................................... 422
  Storing Macro Definitions in External Files ..................................................... 423
  Storing Macro Definitions in Catalog SOURCE Entries ................................. 425
Using the Autocall Facility .................................................. 429
Using Stored Compiled Macros ............................................. 433
Summary ................................................................. 439
Quiz ................................................................. 441

PART 3  Advanced SAS Programming Techniques  445

Chapter 13 • Creating Indexes ............................................. 447
Overview ................................................................. 448
Using Indexes .............................................................. 448
Creating Indexes in the DATA Step ........................................ 449
Managing Indexes with PROC DATASETS ................................ 452
Managing Indexes with PROC SQL ........................................ 454
Documenting and Maintaining Indexes .................................... 455
Summary ................................................................. 461
Quiz ................................................................. 462

Chapter 14 • Combining Data Vertically .................................. 465
Overview ................................................................. 466
Using a FILENAME Statement .............................................. 466
Using the FILEVAR= Option .............................................. 469
Appending SAS Data Sets .................................................. 477
Additional Features ........................................................ 485
Summary ................................................................. 486
Quiz ................................................................. 488

Chapter 15 • Combining Data Horizontally ................................ 495
Overview ................................................................. 496
Reviewing Terminology ..................................................... 497
Working with Lookup Values Outside of SAS Data Sets .............. 500
Combining Data with the DATA Step Match-Merge ...................... 502
Using PROC SQL to Join Data .......................................... 506
Comparing DATA Step Match-Merges and PROC SQL Joins ........... 507
Combining Summary Data and Detail Data ................................. 516
Using an Index to Combine Data ......................................... 521
Using a Transaction Data Set ............................................. 525
Summary ................................................................. 528
Quiz ................................................................. 532

Chapter 16 • Using Lookup Tables to Match Data ....................... 537
Overview ................................................................. 538
Using Multidimensional Arrays ............................................ 538
Populating an Array from a SAS Data Set ................................ 542
Using PROC TRANSPOSE ............................................... 548
Merging the Transposed Data Set ......................................... 553
Using Hash Objects as Lookup Tables .................................... 558
Summary ................................................................. 570
Quiz ................................................................. 573

Chapter 17 • Formatting Data ............................................... 579
Overview ................................................................. 580
Creating Custom Formats Using the VALUE Statement .................. 580
Creating Custom Formats Using the PICTURE Statement .............. 583
Managing Custom Formats ............................................... 588
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 18: Modifying SAS Data Sets and Tracking Changes</td>
<td>818</td>
</tr>
<tr>
<td>Chapter 19: Introduction to Efficient SAS Programming</td>
<td>819</td>
</tr>
<tr>
<td>Chapter 20: Controlling Memory Usage</td>
<td>819</td>
</tr>
<tr>
<td>Chapter 21: Controlling Data Storage Space</td>
<td>820</td>
</tr>
<tr>
<td>Chapter 22: Using Best Practices</td>
<td>820</td>
</tr>
<tr>
<td>Chapter 23: Querying Data Efficiently</td>
<td>821</td>
</tr>
<tr>
<td>Chapter 24: Creating Functions with PROC FCMP</td>
<td>822</td>
</tr>
</tbody>
</table>

**Index** .................................................................................. 823
# Chapter 1
Performing Queries Using PROC SQL

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>4</td>
</tr>
<tr>
<td>Introduction</td>
<td>4</td>
</tr>
<tr>
<td>PROC SQL Basics</td>
<td>4</td>
</tr>
<tr>
<td>Overview</td>
<td>4</td>
</tr>
<tr>
<td>How PROC SQL Is Unique</td>
<td>5</td>
</tr>
<tr>
<td>Writing a PROC SQL Step</td>
<td>6</td>
</tr>
<tr>
<td>Overview</td>
<td>6</td>
</tr>
<tr>
<td>The SELECT Statement</td>
<td>7</td>
</tr>
<tr>
<td>Selecting Columns</td>
<td>8</td>
</tr>
<tr>
<td>Overview</td>
<td>8</td>
</tr>
<tr>
<td>Creating New Columns</td>
<td>9</td>
</tr>
<tr>
<td>Specifying the Table</td>
<td>10</td>
</tr>
<tr>
<td>Specifying Subsetting Criteria</td>
<td>11</td>
</tr>
<tr>
<td>Ordering Rows</td>
<td>11</td>
</tr>
<tr>
<td>Overview</td>
<td>11</td>
</tr>
<tr>
<td>Ordering by Multiple Columns</td>
<td>12</td>
</tr>
<tr>
<td>Querying Multiple Tables</td>
<td>13</td>
</tr>
<tr>
<td>Overview</td>
<td>13</td>
</tr>
<tr>
<td>Specifying Columns That Appear in Multiple Tables</td>
<td>14</td>
</tr>
<tr>
<td>Specifying Multiple Table Names</td>
<td>15</td>
</tr>
<tr>
<td>Specifying a Join Condition</td>
<td>15</td>
</tr>
<tr>
<td>Ordering Rows</td>
<td>16</td>
</tr>
<tr>
<td>Summarizing Groups of Data</td>
<td>17</td>
</tr>
<tr>
<td>Example</td>
<td>17</td>
</tr>
<tr>
<td>Summary Functions</td>
<td>18</td>
</tr>
<tr>
<td>Creating Output Tables</td>
<td>19</td>
</tr>
<tr>
<td>Overview</td>
<td>19</td>
</tr>
<tr>
<td>Example</td>
<td>19</td>
</tr>
<tr>
<td>Additional Features</td>
<td>20</td>
</tr>
<tr>
<td>Summary</td>
<td>20</td>
</tr>
<tr>
<td>Text Summary</td>
<td>20</td>
</tr>
<tr>
<td>Sample Programs</td>
<td>22</td>
</tr>
<tr>
<td>Points to Remember</td>
<td>22</td>
</tr>
<tr>
<td>Quiz</td>
<td>22</td>
</tr>
</tbody>
</table>
Overview

Introduction

Sometimes you need quick answers to questions about your data. You might want to query (retrieve data from) a single SAS data set or a combination of data sets to do the following:

- examine relationships between data values
- view a subset of your data
- compute values quickly.

The SQL procedure (PROC SQL) provides an easy, flexible way to query and combine your data. This chapter shows you how to create a basic query using one or more tables (data sets). You learn how to create a new table from your query.

PROC SQL Basics

Overview

PROC SQL is the SAS implementation of Structured Query Language (SQL), which is a standardized language that is widely used to retrieve and update data in tables and in views that are based on those tables.

The following chart shows terms used in data processing, SAS, and SQL that are synonymous. The SQL terms are used in this chapter. A SAS data set (or SAS data file) can be a table or a view.

<table>
<thead>
<tr>
<th>Data Processing</th>
<th>SAS</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>SAS data file</td>
<td>table</td>
</tr>
<tr>
<td>record</td>
<td>observation</td>
<td>row</td>
</tr>
<tr>
<td>field</td>
<td>variable</td>
<td>column</td>
</tr>
</tbody>
</table>
PROC SQL can often be used as an alternative to other SAS procedures or the DATA step. You can use PROC SQL to do the following:

- retrieve data from and manipulate SAS tables
- add or modify data values in a table
- add, modify, or drop columns in a table
- create tables and views
- join multiple tables (whether they contain columns with the same name)
- generate reports.

Like other SAS procedures, PROC SQL also enables you to combine data from two or more different types of data sources and present them as a single table. For example, you can combine data from two different types of external databases, or you can combine data from an external database and a SAS data set.

**How PROC SQL Is Unique**

PROC SQL differs from most other SAS procedures in several ways:

- Unlike other PROC statements, many statements in PROC SQL include clauses. For example, the following PROC SQL step contains two statements: the PROC SQL statement and the SELECT statement. The SELECT statement contains several clauses: SELECT, FROM, WHERE, and ORDER BY.

  ```
  proc sql;
  select empid, jobcode, salary,
       salary*.06 as bonus
  from sasuser.payrollmaster
  where salary<32000
  order by jobcode;
  ```

- The PROC SQL step does not require a RUN statement. PROC SQL executes each query automatically. If you use a RUN statement with a PROC SQL step, SAS ignores the RUN statement, executes the statements as usual, and generates the note shown below in the SAS log.
Table 1.1 SAS Log

1884 proc sql;
1885    select empid,jobcode,salary,
1886           salary*.06 as bonus
1887       from sasuser.payrollmaster
1888       where salary<32000
1889       order by jobcode;
1890 run;
NOTE: PROC SQL statements are executed immediately;
The RUN statement has no effect.

• Unlike many other SAS procedures, PROC SQL continues to run after you submit a
  step. To end the procedure, you must submit another PROC step, a DATA step, or a
  QUIT statement, as shown:

  proc sql;
     select empid,jobcode,salary,
           salary*.06 as bonus
     from sasuser.payrollmaster
     where salary<32000
     order by jobcode;
quit;

  When you submit a PROC SQL step without ending it, the status bar displays the
  message:

  PROC SQL running

  Note: As a precaution, SAS Enterprise Guide automatically adds a QUIT statement
to your code when you submit it to SAS. However, you should get in the habit of
adding the QUIT statement to your code.

Writing a PROC SQL Step

Overview

Before creating a query, you must first reference the library in which your table is stored.
Then you write a PROC SQL step to query your table.
General form, basic PROC SQL step to perform a query:

```sql
PROC SQL;
  SELECT column-1,... column-n>
  FROM table-1|view-1,...table-n|view-n>
  WHERE expression>
  GROUP BY column-1,... column-n>
  ORDER BY column-1,... column-n>
;
```

Here is an explanation of the syntax:

- **PROC SQL**: invokes the SQL procedure
- **SELECT**: specifies the column(s) to be selected
- **FROM**: specifies the table(s) to be queried
- **WHERE**: subsets the data based on one or more conditions
- **GROUP BY**: classifies the data into groups based on the specified column(s)
- **ORDER BY**: sorts the rows that the query returns by the value(s) of the specified column(s).

**CAUTION:** Unlike other SAS procedures the order of clauses with a SELECT statement in PROC SQL is important. Clauses must appear in the order shown above.

**Note:** A query can also include a HAVING clause, which is introduced at the end of this chapter. To learn more about the HAVING clause, see Chapter 2, “Performing Advanced Queries Using PROC SQL,” on page 26.

### The SELECT Statement

The SELECT statement, which follows the PROC SQL statement, retrieves and displays data. It consists of clauses that begin with a keyword, and is followed by one or more components. The SELECT statement in the following sample code contains four clauses: the required clauses SELECT and FROM, and the optional clauses WHERE and ORDER BY. The end of the statement is indicated by a semicolon.

```sql
proc sql;
  select empid, jobcode, salary,
     salary*.06 as bonus
  from sasuser.payrollmaster
  where salary<32000
  order by jobcode;
```

**Note:** A PROC SQL step that contains one or more SELECT statements is referred to as a PROC SQL query. The SELECT statement is only one of several statements that can be used with PROC SQL.

The following PROC SQL query creates the output report that is shown:
A PROC SQL query produces a result set that can be output as a report, a table, or a PROC SQL view.

<table>
<thead>
<tr>
<th>Type of Output</th>
<th>PROC SQL Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>report</td>
<td>SELECT</td>
</tr>
<tr>
<td>table</td>
<td>CREATE TABLE</td>
</tr>
<tr>
<td>PROC SQL view</td>
<td>CREATE VIEW</td>
</tr>
</tbody>
</table>

*Note:* The CREATE TABLE statement is introduced later in this chapter. You can learn about creating tables in Chapter 5, “Creating and Managing Tables Using PROC SQL,” on page 167. You can learn more about PROC SQL views in Chapter 7, “Creating and Managing Views Using PROC SQL,” on page 248.

You learn more about the SELECT statement in the following sections.

### Selecting Columns

**Overview**

To specify which column(s) to display in a query, you write a SELECT clause, the first clause in the SELECT statement. After the keyword SELECT, list one or more column names and separate the column names with commas. In the SELECT clause, you can both specify existing columns (columns that are already stored in a table) and create new columns.
The following SELECT clause specifies the columns EmpID, JobCode, Salary, and **bonus**. The columns EmpID, JobCode, and Salary are existing columns. The column named **bonus** is a new column.

```sql
proc sql;
select empid,jobcode,salary,
    salary*.06 as bonus
from sasuser.payrollmaster
where salary<32000
order by jobcode;
```

### Creating New Columns

You can create new columns that contain either text or a calculation. New columns appear in output, along with any existing columns that are selected. Keep in mind that new columns exist only for the duration of the query, unless a table or a view is created.

To create a new column, include any valid SAS expression in the SELECT clause list of columns. You can assign a column alias, a name, to a new column by using the keyword **AS** followed by the name that you would like to use.

**Note:** A column alias must follow the rules for SAS names.

In the sample PROC SQL query, shown below, an expression is used to calculate the new column: the values of Salary are multiplied by .06. The keyword **AS** is used to assign the column alias **bonus** to the new column.

```sql
proc sql;
select empid,jobcode,salary,
    salary*.06 as bonus
from sasuser.payrollmaster
where salary<32000
order by jobcode;
```

A column alias is useful because it enables you to reference the column elsewhere in the query.

**Note:** You can learn more about referencing a calculated column from other clauses in Chapter 2, "Performing Advanced Queries Using PROC SQL," on page 26.

Also, the column alias appears as a column heading in the output.

The following output shows how the calculated column **bonus** is displayed. Notice that the column alias **bonus** appears in lowercase, exactly as it is specified in the SELECT clause.
In the SELECT clause, you can specify a label for an existing column or a new column. If both a label alias and a column alias are specified for a new column, the label is displayed as the column heading in the output. If only a column alias is specified, it is important that you specify the column alias exactly as you want it to appear in the output.

Note: You can learn about creating new columns that contain text and about specifying labels for columns in Chapter 2, “Performing Advanced Queries Using PROC SQL,” on page 26.

### Specifying the Table

After writing the SELECT clause, you specify the table to be queried in the FROM clause. Enter the keyword FROM, followed by the name of the table, as shown:

```sql
proc sql;
  select empid, jobcode, salary,
       salary*0.06 as bonus
  from sasuser.payrollmaster
  where salary<32000
  order by jobcode;
```

The PROC SQL step above queries the permanent SAS table Payrollmaster, which is stored in a SAS library to which the libref Sasuser has been assigned.

---

1 Displaying labels for a column is further determined by the LABEL/NOLABEL system option. If this option is set to NOLABEL, then the label not displayed as the column heading in the output. This option can be set by your site administrator.
Specifying Subsetting Criteria

To subset data based on a condition, use a WHERE clause in the SELECT statement. As in the WHERE statement and the WHERE command used in other SAS procedures, the expression in the WHERE clause can be any valid SQL expression. In the WHERE clause, you can specify any column(s) from the underlying table(s). The columns specified in the WHERE clause do not have to be specified in the SELECT clause.

In the following PROC SQL query, the WHERE clause selects rows in which the value of the column Salary is less than 32,000. The output is also shown.

```
proc sql;
    select empid, jobcode, salary,
    salary*.06 as bonus
    from sasuser.payrollmaster
    where salary<32000
    order by jobcode;
```

<table>
<thead>
<tr>
<th>EmpID</th>
<th>JobCode</th>
<th>Salary</th>
<th>bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td>FA1</td>
<td>$31,661</td>
<td>1899.66</td>
</tr>
<tr>
<td>1422</td>
<td>FA1</td>
<td>$31,430</td>
<td>1880.10</td>
</tr>
<tr>
<td>1113</td>
<td>FA1</td>
<td>$31,314</td>
<td>1878.84</td>
</tr>
<tr>
<td>1132</td>
<td>FA1</td>
<td>$31,378</td>
<td>1882.63</td>
</tr>
<tr>
<td>1094</td>
<td>FA1</td>
<td>$31,175</td>
<td>1870.5</td>
</tr>
<tr>
<td>1789</td>
<td>SCP</td>
<td>$25,666</td>
<td>1539.36</td>
</tr>
<tr>
<td>1564</td>
<td>SCP</td>
<td>$25,366</td>
<td>1581.96</td>
</tr>
<tr>
<td>1354</td>
<td>SCP</td>
<td>$25,669</td>
<td>1540.14</td>
</tr>
<tr>
<td>1101</td>
<td>SCP</td>
<td>$25,212</td>
<td>1572.72</td>
</tr>
<tr>
<td>1658</td>
<td>SCP</td>
<td>$25,120</td>
<td>1507.2</td>
</tr>
<tr>
<td>1405</td>
<td>SCP</td>
<td>$25,278</td>
<td>1516.63</td>
</tr>
<tr>
<td>1104</td>
<td>SCP</td>
<td>$25,124</td>
<td>1507.44</td>
</tr>
</tbody>
</table>
```

Ordering Rows

**Overview**

The order of rows in the output of a PROC SQL query cannot be guaranteed, unless you specify a sort order. To sort rows by the values of specific columns, you can use the ORDER BY clause in the SELECT statement. Specify the keywords ORDER BY, followed by one or more column names separated by commas.

In the following PROC SQL query, the ORDER BY clause sorts rows by values of the column JobCode:

```
proc sql;
```
```
select empid, jobcode, salary, 
       salary*.06 as bonus 
from sasuser.payrollmaster 
where salary<32000 
order by jobcode;
```

**Note:** In this example, the ORDER BY clause is the last clause in the SELECT statement, so the ORDER BY clause ends with a semicolon.

In the output of the sample query, shown below, the rows are sorted by the values of JobCode. By default, the ORDER BY clause sorts rows in ascending order.

<table>
<thead>
<tr>
<th>EmpID</th>
<th>JobCode</th>
<th>Salary</th>
<th>Bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td>FA1</td>
<td>$31,061</td>
<td>1899.05</td>
</tr>
<tr>
<td>1422</td>
<td>FA1</td>
<td>$31,430</td>
<td>1880.10</td>
</tr>
<tr>
<td>1113</td>
<td>FA1</td>
<td>$31,314</td>
<td>1876.84</td>
</tr>
<tr>
<td>1132</td>
<td>FA1</td>
<td>$31,378</td>
<td>1882.69</td>
</tr>
<tr>
<td>1004</td>
<td>FA1</td>
<td>$31,175</td>
<td>1870.5</td>
</tr>
<tr>
<td>1789</td>
<td>SCP</td>
<td>$25,686</td>
<td>1539.38</td>
</tr>
<tr>
<td>1564</td>
<td>SCP</td>
<td>$26,366</td>
<td>1581.95</td>
</tr>
<tr>
<td>1354</td>
<td>SCP</td>
<td>$25,689</td>
<td>1540.14</td>
</tr>
<tr>
<td>1101</td>
<td>SCP</td>
<td>$26,212</td>
<td>1572.72</td>
</tr>
<tr>
<td>1658</td>
<td>SCP</td>
<td>$25,120</td>
<td>1507.2</td>
</tr>
<tr>
<td>1405</td>
<td>SCP</td>
<td>$25,278</td>
<td>1516.08</td>
</tr>
<tr>
<td>1104</td>
<td>SCP</td>
<td>$25,124</td>
<td>1507.44</td>
</tr>
</tbody>
</table>

To sort rows in descending order, specify the keyword DESC following the column name. For example, the preceding ORDER BY clause could be modified as follows:

```
order by jobcode desc;
```

In the ORDER BY clause, you can alternatively reference a column by the column's position in the SELECT clause list rather than by name. Use an integer to indicate the column's position. The ORDER BY clause in the preceding PROC SQL query has been modified, below, to specify the column JobCode by the column's position in the SELECT clause list (2) rather than by name:

```
proc sql; 
   select empid, jobcode, salary, 
       salary*.06 as bonus 
from sasuser.payrollmaster 
where salary<32000 
order by 2;
```

**Ordering by Multiple Columns**

To sort rows by the values of two or more columns, list multiple column names (or numbers) in the ORDER BY clause, and use commas to separate the column names (or numbers). In the following PROC SQL query, the ORDER BY clause sorts by the values of two columns, JobCode and EmpID:
proc sql;
    select empid, jobcode, salary,
        salary*.06 as bonus
    from sasuser.payrollmaster
    where salary<32000
    order by jobcode, empid;

The rows are sorted first by JobCode and then by EmpID, as shown in the following output.

<table>
<thead>
<tr>
<th>EmpID</th>
<th>JobCode</th>
<th>Salary</th>
<th>bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1094</td>
<td>FA1</td>
<td>$31,175</td>
<td>1870.5</td>
</tr>
<tr>
<td>1113</td>
<td>FA1</td>
<td>$31,314</td>
<td>1878.84</td>
</tr>
<tr>
<td>1132</td>
<td>FA1</td>
<td>$31,378</td>
<td>1882.68</td>
</tr>
<tr>
<td>1422</td>
<td>FA1</td>
<td>$31,436</td>
<td>1886.16</td>
</tr>
<tr>
<td>1570</td>
<td>FA1</td>
<td>$31,561</td>
<td>1899.86</td>
</tr>
<tr>
<td>1101</td>
<td>SCP</td>
<td>$26,212</td>
<td>1572.72</td>
</tr>
<tr>
<td>1104</td>
<td>SCP</td>
<td>$25,124</td>
<td>1507.44</td>
</tr>
<tr>
<td>1354</td>
<td>SCP</td>
<td>$25,669</td>
<td>1540.14</td>
</tr>
<tr>
<td>1405</td>
<td>SCP</td>
<td>$25,278</td>
<td>1516.68</td>
</tr>
<tr>
<td>1554</td>
<td>SCP</td>
<td>$25,366</td>
<td>1581.95</td>
</tr>
<tr>
<td>1558</td>
<td>SCP</td>
<td>$25,120</td>
<td>1507.2</td>
</tr>
<tr>
<td>1789</td>
<td>SCP</td>
<td>$25,556</td>
<td>1539.36</td>
</tr>
</tbody>
</table>

**Note:** You can mix the two types of column references, names and numbers, in the ORDER BY clause. For example, the preceding ORDER BY clause could be rewritten as follows:

order by 2, empid;

You can also reference column aliases in the ORDER BY clause. Here is an example:

order by 2, empid, bonus;

---

**Querying Multiple Tables**

**Overview**

This topic deals with the more complex task of extracting data from two or more tables. Previously, you learned how to write a PROC SQL step to query a single table. Suppose you now want to examine data that is stored in two tables. PROC SQL enables you to combine tables horizontally, in other words, to combine rows of data.
In SQL terminology, combining tables horizontally is called joining tables. Joins do not alter the original tables.

Suppose you want to create a report that displays the following information for employees of a company: employee identification number, last name, original salary, and new salary. There is no single table that contains all of these columns, so you must join the two tables Sasuser.Salcomps and Sasuser.Newsals. In your query, you want to select four columns, two from the first table and two from the second table. You also need to ensure that the rows that you join belong to the same employee. To check this, you want to match employee identification numbers for rows that you merge and to select only the rows that match.

This type of join is known as an inner join. An inner join returns a result set for all of the rows in a table that have one or more matching rows in another table.

Note: For more information about PROC SQL joins, see Chapter 3, “Combining Tables Horizontally Using PROC SQL,” on page 82.

You can write a PROC SQL step to combine tables. To join two tables for a query, you can use a PROC SQL step such as the one below. This step uses the SELECT statement to join data from the tables Salcomps and Newsals. Both of these tables are stored in a SAS library to which the libref Sasuser has been assigned.

```sql
proc sql;
select salcomps.empid, lastname,
    newsals.salary, newsalary
from sasuser.salcomps, sasuser.newsals
where salcomps.empid=newsals.empid
order by lastname;
```

We examine each clause of this PROC SQL step.

### Specifying Columns That Appear in Multiple Tables

When you join two or more tables, list the columns that you want to select from both tables in the SELECT clause. Separate all column names with commas.

If the tables that you are querying contain same-named columns and you want to list one of these columns in the SELECT clause, you must specify a table name as a prefix for that column. Specifying a table-name prefix with a column that only exists in one table is syntactically acceptable.

Note: Prefixing a table name to a column name is called qualifying the column name.

The following PROC SQL step joins the two tables Sasuser.Salcomps and Sasuser.Newsals, both of which contain columns named EmpID. To tell PROC SQL
where to read the column EmpID, the SELECT clause specifies the table name
Salcomps as a prefix for Empid. The Newsals prefix for Salary is not required, but it is
correct syntax and it identifies the source table for this column.

```sql
proc sql;
  select salcomps.empid, lastname,
        newsals.salary, newsalary
  from sasuser.salcomps, sasuser.newsals
  where salcomps.empid=newsals.empid
  order by lastname;
```

**Specifying Multiple Table Names**

When you join multiple tables in a PROC SQL query, you specify each table name in the
FROM clause, as shown below:

```sql
proc sql;
  select salcomps.empid, lastname,
        newsals.salary, newsalary
  from sasuser.salcomps, sasuser.newsals
  where salcomps.empid=newsals.empid
  order by lastname;
```

As in the SELECT clause, you separate names in the FROM clause (in this case, table
names) with commas.

**Specifying a Join Condition**

As in a query on a single table, the WHERE clause in the SELECT statement selects
rows from two or more tables, based on a condition. When you join multiple tables,
ensure that the WHERE clause specifies columns with data whose values match. If none
of the values match, then zero rows are returned. Also, the columns in the join condition
must be of the same type. The SQL procedure does not attempt to convert data types.

In the following example, the WHERE clause selects only rows in which the value for
EmpID in Sasuser.Salcomps matches the value for EmpID in Sasuser.Newsals. Qualified
column names must be used in the WHERE clause to specify each of the two EmpID
columns.

```sql
proc sql;
  select salcomps.empid, lastname,
        newsals.salary, newsalary
  from sasuser.salcomps, sasuser.newsals
  where salcomps.empid=newsals.empid
  order by lastname;
```

The output is shown, in part, below.
### Note:
In the table Sasuser.Newsales, the Salary column has the label Employee Salary, as shown in this output.

### CAUTION:
If you join tables that do not contain one or more columns with tables that do not have matching data values, several unexpected results might occur. Either you might produce a large amount of data or you might produce all possible row combinations.

---

### Ordering Rows

As in PROC SQL steps that query just one table, the ORDER BY clause specifies which column(s) should be used to sort rows in the output. In the following query, the rows are sorted by LastName:

<table>
<thead>
<tr>
<th>EmpID</th>
<th>LastName</th>
<th>Employee Salary</th>
<th>NewSalary</th>
</tr>
</thead>
<tbody>
<tr>
<td>E00042</td>
<td>ANDERSON</td>
<td>$32,000</td>
<td>$38,023.96</td>
</tr>
<tr>
<td>E00006</td>
<td>ANDERSON</td>
<td>$31,000</td>
<td>$33,753.70</td>
</tr>
<tr>
<td>E00008</td>
<td>BADINE</td>
<td>$85,000</td>
<td>$93,811.78</td>
</tr>
<tr>
<td>E00021</td>
<td>BAKER JR.</td>
<td>$43,000</td>
<td>$43,386.40</td>
</tr>
<tr>
<td>E00002</td>
<td>BOWER</td>
<td>$27,000</td>
<td>$31,153.98</td>
</tr>
<tr>
<td>E00027</td>
<td>BOWMAN</td>
<td>$31,000</td>
<td>$35,579.22</td>
</tr>
<tr>
<td>E00030</td>
<td>BREWER</td>
<td>$38,000</td>
<td>$41,055.05</td>
</tr>
<tr>
<td>E00025</td>
<td>BROCKLEBANK</td>
<td>$23,000</td>
<td>$25,673.57</td>
</tr>
<tr>
<td>E00015</td>
<td>BROWN</td>
<td>$41,000</td>
<td>$45,394.20</td>
</tr>
<tr>
<td>E00041</td>
<td>BRUTON</td>
<td>$45,000</td>
<td>$53,399.58</td>
</tr>
<tr>
<td>E00049</td>
<td>CHASE JR.</td>
<td>$29,000</td>
<td>$32,892.87</td>
</tr>
<tr>
<td>E00024</td>
<td>COCKERHAM</td>
<td>$21,000</td>
<td>$21,213.88</td>
</tr>
<tr>
<td>E00032</td>
<td>COUCH</td>
<td>$24,000</td>
<td>$28,775.81</td>
</tr>
<tr>
<td>E00018</td>
<td>CROSS</td>
<td>$33,000</td>
<td>$35,947.80</td>
</tr>
</tbody>
</table>
Summarizing Groups of Data

We can use PROC SQL steps to create detail reports. But you might also want to summarize data in groups. To group data for summarizing, you can use the GROUP BY clause. The GROUP BY clause is used in queries that include one or more summary functions. Summary functions produce a statistical summary for each group that is defined in the GROUP BY clause.

**Example**

The following example demonstrates the GROUP BY clause and summary functions.

Suppose you want to determine the total number of miles traveled by frequent-flyer program members in each of three membership classes (Gold, Silver, and Bronze). Frequent-flyer program information is stored in the table Sasuser.Frequentflyers. To summarize your data, you can submit the following PROC SQL step:

```sql
proc sql;
    select membertype,
        sum(milestraveled) as TotalMiles
    from sasuser.frequentflyers
    group by membertype;
```

In this case, the SUM function totals the values of the MilesTraveled column to create the TotalMiles column. The GROUP BY clause groups the data by the values of MemberType.

As in the ORDER BY clause, in the GROUP BY clause that you specify the keywords GROUP BY, followed by one or more column names separated by commas.

The results show total miles by membership class (MemberType).
Note: If you specify a GROUP BY clause in a query that does not contain a summary function, your clause is changed to an ORDER BY clause, and a message to that effect is written to the SAS log.

Summary Functions

To summarize data, you can use the following summary functions with PROC SQL. Notice that some functions have more than one name to accommodate both SAS and SQL conventions. Where multiple names are listed, the first name is the SQL name.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG, MEAN</td>
<td>mean or average of values</td>
</tr>
<tr>
<td>COUNT, FREQ, N</td>
<td>number of nonmissing values</td>
</tr>
<tr>
<td>CSS</td>
<td>corrected sum of squares</td>
</tr>
<tr>
<td>CV</td>
<td>coefficient of variation (percent)</td>
</tr>
<tr>
<td>MAX</td>
<td>largest value</td>
</tr>
<tr>
<td>MIN</td>
<td>smallest value</td>
</tr>
<tr>
<td>NMISS</td>
<td>number of missing values</td>
</tr>
<tr>
<td>PRT</td>
<td>probability of a greater absolute value of student's $t$</td>
</tr>
<tr>
<td>RANGE</td>
<td>range of values</td>
</tr>
<tr>
<td>STD</td>
<td>standard deviation</td>
</tr>
<tr>
<td>STDERR</td>
<td>standard error of the mean</td>
</tr>
<tr>
<td>SUM</td>
<td>sum of values</td>
</tr>
<tr>
<td>T</td>
<td>student's $t$ value for testing the hypothesis that the population mean is zero</td>
</tr>
<tr>
<td>USS</td>
<td>uncorrected sum of squares</td>
</tr>
<tr>
<td>VAR</td>
<td>variance</td>
</tr>
</tbody>
</table>
Creating Output Tables

Overview

To create a new table from the results of a query, use a CREATE TABLE statement that includes the keyword AS and the clauses that are used in a PROC SQL query: SELECT, FROM, and any optional clauses, such as ORDER BY. The CREATE TABLE statement stores your query results in a table instead of displaying the results as a report.

General form, basic PROC SQL step for creating a table from a query result:

PROC SQL;
CREATE TABLE table-name AS
SELECT column-1,...,column-n>
FROM table-1|view-1,...,table-n|view-n>
<WHERE expression>
<GROUP BY column-1,...,column-n>>
<ORDER BY column-1,...,column-n>>;

Here is an explanation of the syntax:

table-name
specifies the name of the table to be created.

Note: A query can also include a HAVING clause, which is introduced at the end of this chapter. To learn more about the HAVING clause, see Chapter 2, “Performing Advanced Queries Using PROC SQL,” on page 26.

Note: The CREATE TABLE statement does not generate output. To view the contents of the table, use a SELECT statement as described in “The SELECT Statement” on page 7.

Example

Suppose that after determining the total miles traveled for each frequent-flyer membership class in the Sasuser.Frequentflyers table, you want to store this information in the temporary table Work.Miles. To do so, you can submit the following PROC SQL step:

proc sql;
create table work.miles as
select membertype,
sum(milestraveled) as TotalMiles
from sasuser.frequentflyers
group by membertype;

Because the CREATE TABLE statement is used, this query does not create a report. The SAS log verifies that the table was created and indicates how many rows and columns the table contains.
**Table 1.2  SAS Log**

NOTE: Table WORK.MILES created, with three rows and two columns.

**T I P**  In this example, you are instructed to save the data to a temporary table that is deleted at the end of the SAS session. To save the table permanently in the Sasuser library, use the libref Sasuser instead of the libref Work in the CREATE TABLE clause.

---

**Additional Features**

To further refine a PROC SQL query that contains a GROUP BY clause, you can use a HAVING clause. A HAVING clause works with the GROUP BY clause to restrict the groups that are displayed in the output, based on one or more specified conditions.

For example, the following PROC SQL query groups the output rows by JobCode. The HAVING clause uses the summary function AVG to specify that only the groups that have an average salary that is greater than 40,000 is displayed in the output.

```sql
proc sql;
    select jobcode, avg(salary) as Avg
    from sasuser.payrollmaster
    group by jobcode
    having Avg > 40000
    order by jobcode;
```

*Note:* You can learn more about the use of the HAVING clause in Chapter 2, “Performing Advanced Queries Using PROC SQL,” on page 26.

---

**Summary**

**Text Summary**

**PROC SQL Basics**

PROC SQL uses statements that are written in Structured Query Language (SQL), which is a standardized language that is widely used to retrieve and update data in tables and in views that are based on those tables. When you want to examine relationships between data values, subset your data, or compute values, the SQL procedure provides an easy, flexible way to analyze your data.

PROC SQL differs from most other SAS procedures in several ways:

- Many statements in PROC SQL, such as the SELECT statement, include clauses.
- The PROC SQL step does not require a RUN statement.
- PROC SQL continues to run after you submit a step. To end the procedure, you must submit another PROC step, a DATA step, or a QUIT statement.
Writing a PROC SQL Step
Before creating a query, you must assign a libref to the SAS library in which the table to be used is stored. Then you submit a PROC SQL step. You use the PROC SQL statement to invoke the SQL procedure.

Selecting Columns
To specify which column(s) to display in a query, you write a SELECT clause as the first clause in the SELECT statement. In the SELECT clause, you can specify existing columns and create new columns that contain either text or a calculation.

Specifying Tables
You specify the tables to be queried in the FROM clause.

Specifying Subsetting Criteria
To subset data based on a condition, write a WHERE clause that contains an expression.

Ordering Rows
The order of rows in the output of a PROC SQL query cannot be guaranteed, unless you specify a sort order. To sort rows by the values of specific columns, use the ORDER BY clause.

Querying Multiple Tables
You can use a PROC SQL step to query data that is stored in two or more tables. In SQL terminology, this is called joining tables. Follow these steps to join multiple tables:
1. Specify column names from one or both tables in the SELECT clause and, if you are selecting a column that has the same name in multiple tables, prefix the table name to that column name.
2. Specify each table name in the FROM clause.
3. Use the WHERE clause to select rows from two or more tables, based on a condition.
4. Use the ORDER BY clause to sort rows that are retrieved from two or more tables by the values of the selected column(s).

Summarizing Groups of Data
You can use a GROUP BY clause in your PROC SQL step to summarize data in groups. The GROUP BY clause is used in queries that include one or more summary functions. Summary functions produce a statistical summary for each group that is defined in the GROUP BY clause.

Creating Output Tables
To create a new table from the results of your query, you can use the CREATE TABLE statement in your PROC SQL step. This statement enables you to store your results in a table instead of displaying the query results as a report.

Additional Features
To further refine a PROC SQL query that contains a GROUP BY clause, you can use a HAVING clause. A HAVING clause works with the GROUP BY clause to restrict the groups that are displayed in the output, based on one or more specified conditions.
Sample Programs

Querying a Table
proc sql;
  select empid,jobcode,salary,
       salary*.06 as bonus
  from sasuser.payrollmaster
  where salary<32000
  order by jobcode;
quit;

Summarizing Groups of Data
proc sql;
  select membertype,
       sum(milestraveled) as TotalMiles
  from sasuser.frequentflyers
  group by membertype;
quit;

Creating a Table from the Results of a Query on Two Tables
proc sql;
  create table work.miles as
    select salcomps.empid, lastname,
           newsals.salary, newsalary
    from sasuser.salcomps, sasuser.newsals
    where salcomps.empid=newsals.empid
    order by 2;
quit;

Points to Remember

• Do not use a RUN statement with the SQL procedure.
• Do not end a clause with a semicolon unless it is the last clause in the statement.
• When you join multiple tables, be sure to specify columns that have matching data values in the WHERE clause.
• To end the SQL procedure, you can submit another PROC step, a DATA step, or a QUIT statement.

Quiz

Select the best answer for each question. After completing the quiz, check your answers using the answer key in the appendix.

1. Which of the clauses in the PROC SQL program below is written incorrectly?

   proc sql;
   select style sqfeet bedrooms
   from choice.houses
where sqfeet ge 800;

a. SELECT
b. FROM
c. WHERE
d. both a and c

2. How many statements does the program below contain?

proc sql;
   select grapes,oranges,
      grapes + oranges as sumsales
   from sales.produce
   order by sumsales;

a. two
b. three
c. four
d. five

3. Complete the following PROC SQL query to select the columns Address and SqFeet from the table List.Size and to select Price from the table List.Price. (Only the Address column appears in both tables.)

proc sql;
   ____________
   where size.address = price.address;
   from list.size,list.price;

a. select address,sqfeet,price
b. select size.address,sqfeet,price
c. select price.address,sqfeet,price
d. either b or c

4. Which of the clauses below correctly sorts rows by the values of the columns Price and SqFeet?

a. order price, sqfeet
b. order by price,sqfeet
c. sort by price sqfeet
d. sort price sqfeet

5. Which clause below specifies that the two tables Produce and Hardware be queried? Both tables are located in a library to which the libref Sales has been assigned.

a. select sales.produce sales.hardware
b. from sales.produce sales.hardware
c. from sales.produce,sales.hardware
d. where sales.produce, sales.hardware

6. Complete the SELECT clause below to create a new column named Profit by subtracting the values of the column Cost from those of the column Price.

select fruit,cost,price,
   __________________
a. Profit=price-cost  
b. price-cost as Profit  
c. profit=price-cost  
d. Profit as price-cost  

7. What happens if you use a GROUP BY clause in a PROC SQL step without a summary function?  
a. The step does not execute.  
b. The first numeric column is summed by default.  
c. The GROUP BY clause is changed to an ORDER BY clause.  
d. The step executes but does not group or sort data.  

8. If you specify a CREATE TABLE statement in your PROC SQL step,  
a. the results of the query are displayed, and a new table is created.  
b. a new table is created, but it does not contain any summarization that was specified in the PROC SQL step.  
c. a new table is created, but no report is displayed.  
d. results are grouped by the value of the summarized column.  

9. Which statement is true regarding the use of the PROC SQL step to query data that is stored in two or more tables?  
a. When you join multiple tables, the tables must contain a common column.  
b. You must specify the table from which you want each column to be read.  
c. The tables that are being joined must be from the same type of data source.  
d. If two tables that are being joined contain a same-named column, then you must specify the table from which you want the column to be read.  

10. Which clause in the following program is incorrect?  
```sql
proc sql;
 select sex,mean(weight) as avgweight
 from company.employees company.health
 where employees.id=health.id
 group by sex;
```

a. SELECT  
b. FROM  
c. WHERE  
d. GROUP BY
Index

**Special Characters**
- `_ATDATETIME_` audit trail variable 631
- `_ATMESSAGE_` audit trail variable 631
- `_ATOBSNO_` audit trail variable 631
- `_ATOPCODE_` audit trail variable 631
- `_ATRETURNCODE_` audit trail variable 631
- `_ATUSERID_` audit trail variable 631
- `_DSEMTR_` mnemonic 619
- `_DSENMR_` mnemonic 619
- `_DSENOM_` mnemonic 619
- `_IORC_` automatic variable 524, 619
- `_SOK_` mnemonic 619
- `_TEMPORARY_` keyword 538
- `_TYPE_` variable 774, 776
- `;` (semicolon) 375
- `?` conditional operator 33, 34
- `. ` (period) 318
- `&` (ampersand) 290
- `%` (percent sign)
  - macro programs and specifying directives 586
  - tokens and 304
- `%a` directive 586
- `%A` directive 586
- `%b` directive 586
- `%B` directive 586
- `%QUOTE` function 305
- `%CMPRES` statement 430
- `%COPY` statement 437
- `%d` directive 586
- `%DATATYP` statement 430
- `%DO-%END` statement 397, 407, 410
- `%EVAL` function 411
- `%GLOBAL` statement 388
- `%H` directive 586
- `%i` directive 586
- `%IF-%THEN-%ELSE` macro statement 396, 397, 400, 403, 406
- `%INCLUDE` statement 423
- `%INDEX` function 311
- `%j` directive 586
- `%LEFT` statement 430
- `%LENGTH` function 309
- `%LET` statement
  - macro parameter support 381
  - processing 331
  - user-defined macro variables and 294
- `%LOCAL` statement 389
- `%LOWCASE` statement 430
- `%m` directive 586
- `%M` directive 586
- `%MACRO` statement
  - creating stored compiled macros 434
  - general form 373
  - macro parameter support 383, 384, 385
  - PARMBUFF option 386
  - SOURCE option 435, 437
  - STORE option 434
- `%MEND` statement 373
- `%NRSTR` function 305
- `%p` directive 586
- `%PUT` statement 300
- `%QLOWCASE` statement 430
- `%QSCAN` function 313
- `%QSUBSTR` function 310
- `%QSYSFUNC` function 315
- `%QUPCASE` function 308
- `%S` directive 586
- `%SCAN` function 312
- `%STR` function 303
- `%SUBSTR` function 309
- `%SYSEVALF` function 413
- `%SYSFUNC` function 314
- `%SYSRC` autocall macro 619
- `%TRIM` statement 430
- `%U` directive 586
- `%UPCASE` function 306
- `%w` directive 586
- `%y` directive 586
- `%Y` directive 586
- `=`*` conditional operator
  - description 33
  - general form 39
  - WHERE conditions and 755
A

access methods, selecting 750
accumulator variables 519
ADD clause, ALTER TABLE statement (SQL) 209
ADD method 562
aliases
column 9, 40, 88
in-line views 107
table 89
ALL conditional operator 33, 68
ALL keyword
DELETE statement (DATASETS) 641
EXCEPT set operator and 135, 136
INTERSECT set operator and 140, 142
set operations and 131
UNION set operator and 146, 148
ALTER TABLE statement, SQL
  procedure
    ADD clause 209
    DROP clause 211
    functionality 72
    general form 209
    MODIFY clause 212
ALTER= data set option 630
ampersand (&) 290
ANY conditional operator
comparison operator and 66
description 33
APPEND procedure
    BASE= data set option 477
    DATA= data set option 477
    FORCE option 479, 480, 482
general form 477
arguments, summary functions and 48, 50, 51
arithmetic expressions in macro programs 411, 413
ARRAY statement
    combining data with 501
    creating arrays 543
general form 538
    loading array elements 544
    lookup values and 501
    stored array values and 542
arrays
    combining data in 501
    creating 543
defined 501
    loading elements 544
    multidimensional 538
    reading values 546
    stored values 542, 543, 544, 546
AS keyword, CREATE TABLE statement (SQL) 177
ATTRIB statement 501
attributes 560, 724
AUDIT statement, DATASETS procedure 629, 634
audit trails
    controlling 634
    controlling data in 631, 632
    initiating 629
    overview 628
    reading files 630
    USER_VAR statement 633
    variables 631, 632
autocall libraries
    accessing macros 431
    creating 429
default 430
defined 429
automatic macro variables
defined 290
functionality 291
global symbol table and 290, 387
AVG function 55

B

base table 497
BASE= data set option 477
BEFORE_IMAGE option, LOG statement 632
benchmark guidelines 656
best practices
    conditional logic 710
    DO groups 716
    eliminating data passes 721, 723, 724
    executing only necessary statements 708
    reading/writing essential data 725, 727, 728, 730
    subsetting variables 730, 731
BETWEEN-AND conditional operator
description 33
general form 34
identifying conditions to optimize 754
binary search 501
buffers
    controlling number of 660, 661, 663
    controlling page size 660, 661
    BUFNO= data set option 663
    BUFNO= system option 663
    BUFSIZE= system option 661
    BY statement, DATA step
        handling duplicate values 613
        MODIFY statement and 611, 619
        TRANSPOSE procedure 553
    BY variable
        DATA step match-merge 502, 511
        joining tables 100, 101, 102
C
CALCULATED keyword 40
CALL MISSING routine 562
Cartesian product 83, 512
CASE expression
general form 202
INSERT statement (SQL) 206
SELECT statement (SQL) 206
UPDATE statement (SQL) 201, 203, 205
case operand
CASE expression, UPDATE statement
(SQL) 203, 205
defined 203
case sensitivity in macro comparisons 406
CATALOG access method, FILENAME statement 427
CATALOG procedure
general form 426, 589
managing formats 589
catalogs, storing macro definitions 425, 426, 427
CENTILES option, CONTENTS procedure 757
CHANGE statement, DATASETS procedure 459, 640
character constants 45
character data types
column widths and 171
defined 170
flowing characters in columns 269
character strings
%INDEX function 311
%LENGTH function 309
%QSCAN function 313
%QSUBSTR function 310
%UPCASE function 308
%SCAN function 312
%SUBSTR function 309
%UPCASE function 306
macro character functions 306
character variables, storing 677
CHART procedure 318
CHECK constraint type
functionality 187, 188, 620
in column specification 188
CLASS statement
TABULATE procedure 582
class variables
_TYPE_ values 774, 776
combining 778
summary statistics and 771, 772
CNTLIN= option, FORMAT procedure 594, 595
CNTLOUT= option, FORMAT procedure 598
COALESCE function 102, 103
column alias
CALCULATED keyword and 40
defined 9
renaming columns with 88
column constraints 188
column modifiers 172
column widths 171
columns
adding to tables 209
altering in tables 209
as multiple arguments for summary functions 51
combining 130
counting non-missing values 56
counting unique values 57
creating constraints 188
creating new 9
creating tables by defining 168
deleting from tables 211
eliminating duplicate 87
flowing characters within key 226, 229
modifying in tables 212
ordering by multiple 12
outside summary functions 52
overlaying 130
processing calculated 40
renaming with column aliases 88
rows numbers in output 267
selecting 8
specifying formats and labels 43
specifying in multiple tables 14
specifying subsets in tables 175
summary functions and 50, 51
viewing all 28
comments in macro programs 380
COMPARE procedure 680
comparison operators
ALL conditional operator and 68
ANY conditional operator and 66
identifying conditions to optimize 754
subqueries and 65
compiling macro programs 373, 374
composite indexes
creating 232, 233
defined 228, 448, 449
COMPRESS function 472
COMPRESS= data set option 687
COMPRESS= system option 687
compressing data files
costs of 761
storage considerations 685, 686, 687, 689, 690, 692
concatenating data
appending data sets 477, 479, 480, 482
creating delimiting list of values 359
defined 466
FILENAME statement 466
INFILE statement 469, 471, 472, 473, 474, 476
conditional operators 32
conditional processing
best practices 710
case sensitivity in macro programs 406
for macro programs 396, 397, 400, 403, 406
constants 45, 302
CONTAINS conditional operator
description 33
general form 34
identifying conditions to optimize 754
CONTENTS procedure
CENTILES option 757
displaying index specifications 235
maintaining indexes 455
reporting buffer information 661
CONTENTS statement
CATALOG procedure 426
DATASETS procedure 455, 626, 661, 757
conventional join 85
COPY procedure 458
COPY statement, DATASETS procedure 458
copying
data sets 458, 625
tables 178
CORR keyword
EXCEPT set operator and 135, 136
INTERSECT operator and 141, 142
OUTER UNION set operator and 153, 156
set operations and 131
UNION set operator and 147, 148
correlated subqueries
defined 62
EXISTS conditional operator 70
indexes and 229
NOT EXISTS conditional operator 70
subsetting data 69
COUNT function 55
CREATE INDEX statement, SQL procedure
creating composite indexes 232, 233
creating multiple indexes 232
creating simple indexes 232, 233
displaying index specifications 233
functionality 226
general form 231, 454
CREATE statement, SQL procedure 72
CREATE TABLE statement, SQL procedure
AS keyword 177
copying tables 178
creating constraints 188
creating constraints outside column specifications 191
creating empty tables 168
creating like other tables 174
creating output tables 19
creating tables from query results 177
displaying table structure 173
FORMAT= option 172
FROM clause 178
general form for creating output tables 19
general form with column specifications 169
general form with constrain specification 191
general form with constraint specification 189
general form with LIKE clause 174
general form with query clauses 177
INFORMAT= option 172
LABEL= option 172
LIKE clause 174
MESSAGE= option 189, 191
MSGTYPE= option 189, 191
SELECT clause 178
specifying column modifiers 172
specifying column widths 172
specifying data types 170
specifying empty tables like other tables 174
specifying subsets of columns 175
CREATE VIEW statement, SQL procedure
FROM clause 253
general form 249
USING clause 253

D
data files
accessing observations directly 689
COMPRESS= data set option 687
COMPRESS= system option 687
compressed structure 686
compression process 685, 692
number of pages 760
POINTOBS= data set option 689
reasons for compressing 686
REUSE= data set option 690
REUSE= system option 690
debugging
  FEEDBACK option, SQL procedure 29
  macro programs 378, 379, 380
DECLARE statement 565
DEFAULT= option, LENGTH statement
  (DATA) 678
DEFINEDATA method 561, 565
DEFINEDONE method 561
DEFINEKEY method 561, 565
DELETE statement, DATASETS
  procedure 640, 641
DELETE statement, SQL procedure
  audit trails 628
  functionality 72
  general form 207
  updating views 255
deleting
  columns from tables 211
  indexes 239
  rows in tables 207
  tables 216
  views 257
delimiters
  in macro programs 375
  in macro variable names 318
dense match 499
DESCRIBE statement
  DATA step 698
  SQL procedure 72
DESCRIBE TABLE CONSTRAINTS
  statement, SQL procedure 197
DESCRIBE TABLE statement, SQL
  procedure
  displaying Dictionary table definitions 276
  displaying index specifications 233
  displaying indexes 226
  displaying table structure 173
  general form 173, 233
DESCRIBE VIEW statement, SQL
  procedure 251
detail reports 765
Dictionary tables
  functionality 264, 275
  querying 276
digit selectors 584
direct access 749
directives, specifying pictures 584, 585
DISTINCT keyword 31
DO loops 544, 716
DO UNTIL loops 520
dot notation method 561
DOUBLE option, SQL procedure 268
DROP clause, ALTER TABLE statement
  (SQL) 211
DROP INDEX statement, SQL procedure 239, 454
DROP statement
  DATA step 730, 731
  SQL procedure 72
DROP statement, SQL procedure 731
DROP TABLE statement, SQL procedure 216
DROP VIEW statement, SQL procedure 257
DROP= data set option
  general form 175
  subsetting variables 730, 731
duplicate columns, eliminating 87
duplicate rows
  eliminating from output 31
  processing unique vs. 130
duplicate values in data sets 613, 616

E
  efficiency
    assessing needs 652, 653, 654
    benchmark guidelines 656
    comparing resource usage 759, 760, 761
    computer resources 652
detail reports 765
    estimating observations 756
    identifying available indexes 750, 752
    identifying conditions to optimize 754, 755
    indexes and WHERE processing 747, 749
tools for summarizing data 767, 768, 774
    tracking resources 655
    trade-offs in 654
empty tables 168, 174
END= option, INFILE statement 473
error handling
  for row insertions 194
  monitoring I/O error conditions 619
  stopping SQL procedure 279
ERROR_IMAGE option, LOG statement 632
ERRORSTOP option, SQL procedure 279
evaluating performance 271
EXCEPT set operator
  ALL keyword and 135, 136
  CORR keyword and 135, 136
  functionality 129, 132
EXCLUDE statement, FORMAT
  procedure 588, 599
EXEC option, SQL procedure 279
EXISTS conditional operator correlated queries and 70
description 33
external files selecting observations from 728
storing macro definitions in 423
storing raw data filenames in 485

F
FEEDBACK option, SQL procedure 29
FILENAME statement 427, 466
FILEVAR= option, INFILE statement 469
FIND method 562, 566, 567
FIRSTOBS= option, WHERE statement (PRINT) 727
FLOW option, SQL procedure 269
FMTERR system option 593
FMTLIB keyword, FORMAT procedure 588
FMTSEARCH= system option 592
FOOTNOTE statement, SQL procedure 44
FORCE option, APPEND procedure 479, 480, 482
FOREIGN KEY constraint type
constraint specification 191
functionality 187, 188, 620
in column specification 188
FORMAT procedure
binary search 501
CNTLIN= option 594, 595
CNTLOUT= option 598
EXCLUDE statement 588, 599
FMTLIB keyword 588
general form 588
LIB= option 592
PICTURE statement 583, 586
SELECT statement 588, 599
VALUE statement 501, 502, 580, 581, 582
FORMAT statement, TABULATE procedure 582
FORMAT= option
CREATE TABLE statement (SQL) 172
SELECT statement, SQL procedure 43
TABULATE procedure 582
formats
associating with variables 501
creating data sets from 598
creating from data sets 594, 595, 598
creating with PICTURE statement 583, 586
creating with VALUE statement 580, 581
FMTERR system option 593
FMTSEARCH= system option 592
managing 588, 589
multilabel 582
NOFMTERR system option 593
permanently assigning 591, 592, 593
specifying for columns 43
substituting to avoid errors 593
with overlapping ranges 581
Forward Re-Scan rule 347
FREQ procedure 711, 768
FROM clause
CREATE TABLES (SQL) 178
CREATE VIEW statement (SQL) 253
INSERT statement (SQL) 185
SELECT statement (SQL) 10, 15, 83, 105
FSEDIT procedure 689
FSLIST procedure 467
full outer join 93, 97
FULLSTIMER system option 655
functions
%QSYSFUNC function 315
%SYSFUNC function 314
macro variable support 314
G
GCHART procedure 318, 711
general integrity constraints 188, 621
generation data sets
creating 637
defined 636
initiating 637
processing 638, 639
GENMAX= option, MODIFY statement (DATASETS) 637
GENNUM= data set option 638, 639, 641
global symbol table
%GLOBAL statement 388
defined 290
macro variables and 290, 387
GPLOT procedure 318, 711
GROUP BY clause, SELECT statement (SQL)
selecting groups 58
summarizing groups of data 17, 48, 49
summary functions and 51, 54
groups, selecting 58
H
hash objects
CALL MISSING routine 562
creating from data sets 564
DATA step component objects 560
declaring 560
defined 558
defining keys and data 561
loading key/data values 562
multiple data variables 565
processing 563
retrieving matching data 562
retrieving multiple data values 566
return codes with FIND method 567
SET statement and 565
structure 559
HAVING clause, SELECT statement (SQL) 20, 58
HIST keyword 641

I
I/O processing
comparing resource usage 759, 760, 761
measuring 660
IC CREATE statement, DATASETS procedure 622
IC DELETE statement, DATASETS procedure 627
IDXNAME=data set option
controlling index usage 763
functionality 237
general form 238
IDXWHERE=data set option
controlling index usage 763
functionality 237
general form 237
IF-THEN/ELSE statement
%IF-%THEN comparison 397
best practices 710, 716
lookup values outside data sets 500, 501
IN conditional operator
description 33
general form 35
identifying conditions to optimize 754
in-line views
assigning aliases 107
functionality 105
multiple tables and 106
INDEX CREATE statement, DATASETS procedure 452
INDEX DELETE statement, DATASETS procedure 452
INDEX= option, DATA step 449
indexes
accessing rows in tables 227
benefits 229, 230, 749
combining data with 521, 522, 524
composite 228, 232, 233, 448, 449
controlling usage 763
copying data sets 458
costs of using 230, 749
creating 231
creating in DATA step 449
creating multiple 232
creation guidelines 231, 761
defined 226, 448
displaying specifications 233
documenting 455
dropping 239
identifying available 750, 752
maintaining 455
managing usage 235, 237
managing with DATASETS procedure 452
managing with SQL procedure 454
modifying observations located by 614
querying 229, 747, 748, 749, 750
reasons for not using 452
renaming data sets 459
renaming variables 460
simple 227, 232, 448, 449
subsetting data 230
types of 227, 449
unique 228, 233
INFILE statement
assigning names of files to be read 471
COMPRESS function 472
date functions 474
END= option 473
FILEVAR= option 469
general form 469
INTNX function 476
INFORMAT= option, CREATE TABLE statement (SQL) 172
INITIATE statement, DATASETS procedure 629
inner joins
combining data horizontally 507, 511, 512, 514
defined 14, 85
general form with outer join 100
INOB= option, SQL procedure 265
input data sources 498
INPUT function 315, 547
INPUTC function 315
INPUTN function 315
INSERT statement, SQL procedure
audit trails 628
CASE expression 206
controlling UNDO processing 194
FROM clause 185
functionality 72
general form 181, 182, 185
handling errors in row insertions 194
inserting rows from query results 185
inserting rows of data in tables 180, 181, 182
SELECT clause 185
SET clause 180, 181
updating views 255
VALUES clause 180, 182, 194

integrity constraints
creating outside of column specifications 191
creating tables with 187
displaying for tables 197
documenting 626
enforcing 624
functionality 620, 621
general 188, 621
general form with specifications 191
in column specifications 188
placing in data sets 622, 624, 625
referential 188, 621
removing 627
INTERSECT set operator
ALL keyword and 140, 142
CORR keyword and 141, 142
functionality 129, 139
INTNX function 476
INTO clause, SELECT statement (SQL) 354, 356, 359
IS MISSING conditional operator 33, 36, 754
IS NULL conditional operator 33, 36, 754

J
joining data sets
combining summary/detail data 516, 517, 519
DATA step match-merge 502, 503
defined 496
indexes and 521, 522, 524
lookup values and 500, 501
multiple SET statements 514, 515
relationships between input data sources 498
SQL procedure support 506, 507, 511, 512, 514
terminology 497
transactional data sets 525, 526
transposed data sets 553, 554, 555, 556, 557
joining tables
advantages 104
COALESCE function 102, 103
comparing with DATA step match-merges 100, 101, 102
defined 14, 82
EXCEPT set operator 132
for rows with matching values 88
in-line views and 105, 106
indexes and 229
inner joins 14, 85
inner joins with outer-join general form 100
INTERSECT set operator 139
outer joins 93
OUTER UNION set operator 151, 156
processes defined 86
set operations 127
UNION set operator 144
with views 109

K
KEEP statement, DATA step 597, 730, 731
KEEP= data set option
general form 175
subsetting variables 730, 731
key columns 226, 229
key values
defined 497, 538
return codes with FIND method 567
key variables 497
KEY= option
MODIFY statement (DATA) 536, 614, 617, 619
SET statement (DATA) 522
keyword parameters 384, 385
keywords, modifying set operations 131

L
LABEL= option
CREATE TABLE statement (SQL) 172
SELECT statement, SQL procedure 43
labels, column 43
LEFT function 316
left outer join 93, 95
LENGTH statement, DATA step 677, 678
LIB= option, FORMAT procedure 592
LIBRARY= option, DATASETS procedure 456
librefs, views and 253
LIKE clause, CREATE TABLE statement (SQL) 174
LIKE conditional operator
description 33
general form 37
identifying conditions to optimize 754
specifying patterns 38
literal tokens 297
local symbol table 388, 389, 392
LOG statement
  BEFORE_IMAGE option 632
  controlling data in audit trails 632
  DATA_IMAGE option 632
  ERROR_IMAGE option 632
logical expressions in macro programs 411, 413
lookup tables
  defined 497, 538
  hash objects as 558
  multidimensional arrays and 538
  multiple 503
  stored array values 542, 543, 544, 546
  TRANSPOSE function 548, 551
  transposed data sets 553, 554, 555, 556, 557
LOOPS= option, SQL procedure 279

M
macro character functions
  %INDEX function 311
  %LENGTH function 309
  %QSCAN function 313
  %QSUBSTR function 310
  %QUCASE function 308
  %SCAN function 312
  %SUBSTR function 309
  %UPCASE function 306
  manipulating character strings 306
macro definitions 423
macro facility 289, 298
macro language 298
macro processor
  functionality 330
  macro facility and 298
  macro variables rules 390
  referencing macro variables 290
macro programs
  %DO-%END statement 397
  %EVAL function 411
  %GLOBAL statement 388
  %IF-%THEN/%ELSE macro statement 396, 397, 400, 403, 406
  %LOCAL statement 389
  %SYSEVALF function 413
arithmetic/logical expressions 411, 413
  calling 375
  case sensitivity and 406
  comments in 380
  compiling 373, 374
  conditionally processing statements 396, 397, 400, 403, 406
debugging 378, 379, 380
defined 373
developing 378, 379, 380
executing 376
functionality 372
iterative processing for 407, 410
monitoring execution 378, 379, 380
nested 392, 393
parameters and 381, 383, 384, 385, 386
Stored Compiled Macro Facility 433, 434, 435, 436, 437
storing definitions in catalog SOURCE entries 425, 426, 427
storing in autocall libraries 426, 429, 430, 431
storing macro in external files 423
storing session-compiled macros 422
symbol tables 387, 388, 389, 390, 392, 393, 395
macro quoting functions
  %BQUOTE function 305
  %NRSTR function 305
  %STR function 303
  masking special characters 302
macro triggers 298
macro variables
  %PUT statement 300
  automatic 290, 291, 387
  combining references with text 316
  creating delimitied list of values 359
  creating during DATA step 329
  creating during PROC SQL execution 354
  creating multiple during DATA step 343
  creating with INTO clause 356
  DATALINES statement and 290
delimiters in names 318
displaying values in SAS log 299, 300
  Forward Re-Scan rule 347
  functionality 288, 289
  in symbol tables 387, 388, 389, 390, 392, 393, 395
  macro facility and 289
  macro parameters and 381, 383, 384, 385, 386
  macro processor rules 390
  obtaining values during DATA step 352
  processing 296
  PUT function 315, 339
  referencing 290, 316
  referencing indirectly 346
  SAS function support 314
  SCL program support 362, 363
  SYMBOLGEN system option 299
  SYMGET function 352
SYMGETN function 363
SYMPUT routine 332, 333, 334, 336, 343
SYMPUTN routine 363
SYMPUTX routine 341
tokenization 297
triggers for 298
user-defined 290, 293, 387
many-to-many match 499, 509
masking special characters
%BQUOTE function 305
%NRSTR function 305
%STR function 303
macro function support 302
MAUTOLOCDISPLAY system option 432
MAUTOSOURCE system option 431
MCOMPILENOTE= system option 374
MEANS procedure
 generalize form 517
multilabel formats 582
NOPRINT option 517
NWAY option 778
OUTPUT statement 517, 780
tools for summarizing data 768, 771, 772
TYPES statement 776
WAYS statement 784
WHERE= data set option 780
memory
controlling number of buffers 660, 661, 663
controlling page size 660, 661
MEMRPT system option 655
MERGE statement, DATA step 502, 503
message characters 584
MESSAGE= option, CREATE TABLE statement (SQL) 189, 191
metadata 264
methods 560
missing values in data sets 614
MLF option, TABULATE procedure 582
MLOGIC system option 379
MLOGICNEST system option 395
MODIFY clause, ALTER TABLE statement (SQL) 212
MODIFY statement, DATA step
audit trails and 628
BY statement 611, 613, 619
functionality 608, 609
general form 610
KEY= option 536, 614, 617, 619
POINT= option 689
UNIQUE option 617
UPDatemODE= option 614
MODIFY statement, DATASETS procedure 452, 637
MONTH function 474
MPRINT system option 378
MPRINTNEST system option 393
MSGLEVEL= system option 236, 450, 764
MSGTYPE= option, CREATE TABLE statement (SQL) 189, 191
MSTORED system option 433, 436
multidimensional arrays 538
multilabel formats 582
MULTILABEL option, VALUE statement (FORMAT) 582
multiple columns 12, 51
multiple indexes 12, 51
multiple tables
querying 13
specifying columns in 14
specifying names 15
views and 106, 109

N
name tokens 298
NAME= option, TRANSPOSE procedure 551
nested macros 392, 393
NMISS function 55
NODouble option, SQL procedure 268
NOERRORSTOP option, SQL procedure 279
NOEXEC option, SQL procedure 71, 279
NOFLOW option, SQL procedure 269
NOFULLTER system option 593
NOFulSTIMER system option 656
NOMEMRPT system option 656
noncorrelated subqueries
defined 62
multiple-value 64
single-value 63
subsetting data 63
nonmatching data 499, 510
NONNUMBER option, SQL procedure 267
NOPRINT option
MEANS procedure 517
SELECT statement (SQL) 354
NOPROMPT option, SQL procedure 267, 279
NOSTATS system option 656
NOSTIMER option, SQL procedure 271
NOSTIMER system option 656
NOT EXISTS conditional operator 70
NOT NULL constraint type
constraint specification 191
functionality 187, 188, 620
in column specification 188
NOTSORTED option
TRANSPOSE procedure 554
NUMBER option, SQL procedure 267
number tokens 298
numeric data types 170, 171
numeric variables, storing 677, 678, 679, 680
NWAY option, MEANS procedure 778

O
OBS= option, WHERE statement (PRINT) 727
OBSBUF= data set option 699
one-to-many match 498, 509
one-to-one match 498, 508
OPTIONS statement, SQL procedure
BUFNO= option 663
BUFSIZE= option 661
COMPRESS= option 687
FMTERR option 593
FMTSEARCH= option 592
FULLSTIMER option 655
MAUTOLOCDISPLAY option 432
MAUTOSOURCE option 431
MCOMPILENOTE= option 374
MEMRPT option 655
MLOGIC option 379
MLOGICN option 395
MLOGICNEST option 395
MLOGICNEST system option 395
MPRINT option 378
MPRINT system option 378
MPRINTN option 393
MPRINTN option system 393
MSGLEVEL= option 236, 450
MSGLEVEL= system option 450
MSTORED option 433, 436
NOFMTERR option 593
NOFULLSTIMER option 656
NOMEMRPT option 656
NOSTATS option 656
NOSTIMER option 656
REUSE= option 690
SASAUTOS option 432
SASAUTOS= option 431
SASMSTORE option 436
SASMSTORE= option 433
STATS option 655
STIMER option 271, 655
SYMBOLGEN option 299
ORDER BY clause, SELECT statement (SQL)
ordering by multiple columns 12
ordering rows 11, 16
outer joins
functionality 93
general form 94
inner joins and 100
OUTER UNION set operator
CORR keyword and 153, 156
functionality 129, 151, 152
OUTOBS= option, SQL procedure
general form 29
restricting row processing 265
summary functions and 54
output
adding character constants 45
controlling 267
detail reports 765
double-spacing 268
eliminating duplicate rows from 31
enhancing for queries 42
flowing characters in columns 269
including row numbers 267
SYMPUT routine support 336
OUTPUT statement
DATA step 618
MEANS procedure 517, 780
output tables, creating 19

P
parameters
in macro programs 381, 383, 384, 385, 386
keyword 384, 385
PARMBUFF option 386
positional 383, 385
PARMBUFF option, %MACRO statement 386
patterns 37, 38
percent sign (%)
macro programs and 375
specifying directives 586
tokens and 304
performance
testing and evaluating 271
period (.) 318
PICTURE statement, FORMAT procedure
DATATYPE= option 585
general form 583
specifying directives 586
specifying pictures 584
PLOT procedure 318
POINT= option
MODIFY statement (DATA) 689
SET statement (DATA) 689
POINTOBS= data set option 689
positional parameters 383, 385
PREFIX= option, TRANSPOSE procedure 551

PRIMARY KEY constraint type
constraint specification 191
functionality 187, 188, 620
in column specification 188

PRINT procedure
detail reports 765
UNIFORM option 700
WHERE statement 727

PROC FCMP 789
process management
best practices 737, 738
conditional processing for macro programs 396, 397, 400, 403, 406
controlling execution 265
controlling output 267
controlling updates 618, 619
creating macro variables in DATA step 329
creating macro variables in PROC SQL 354, 356, 359
creating multiple macro variables in DATA step 343
Dictionary tables 264, 275
error handling 279
for macro programs 378, 379, 380
Forward Re-Scan rule 347
hash objects 563
iterative processing for macro programs 407, 410
macro variables 296, 390
macro variables in SCL programs 362, 363
obtaining macro variable values in DATA step 352
PUT function 315, 339
referencing macro variables indirectly 346
resetting options 273
restricting number of loops 279
specifying SQL options 264
SYMGET function 352
SYMGETN function 363
SYMPUT routine 332, 333, 334, 336, 343
SYMPUTN routine 363
SYMPUTX routine 341
testing and evaluating performance 271
working with views 361
PROMPT option, SQL procedure 267, 279
PUT function 315, 339, 501
PUTC function 315
PUTN function 315

Q
queries
comparing resource usage 759, 760, 761
creating output tables 19
creating tables from 177
detail reports 765
Dictionary tables and displaying all columns 28
eliminating duplicate rows from output 31
enhancing output 42
estimating observations 756
identifying available indexes 750, 752
identifying conditions to optimize 754, 755
in-line views 105
indexes and 229, 747, 748, 749, 750
inserting rows from 185
joining multiple tables and views 109
limiting number of rows displayed 29
ordering rows 11, 16
querying multiple tables 13
selecting columns 8
specifying subsetting criteria 11
specifying tables 10
subqueries for subsetting data 61
subsetting data using correlated subqueries 69
subsetting data using noncorrelated subqueries 63
subsetting rows using calculated values 40
subsetting rows with conditional operators 32
summarizing groups of data 17, 48
tools for summarizing data 767, 768, 774
validating general form 71
viewing all columns 28
viewing SELECT statement general form 27
views in 248
writing SQL procedure steps 6

R
raw data filenames, storing 485
referential integrity constraints 188, 621
REMOVE statement, DATA step 618
RENAME statement
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA step</td>
<td>597</td>
</tr>
<tr>
<td>DATASETS procedure</td>
<td>460</td>
</tr>
<tr>
<td>RENAME= data set option</td>
<td>552</td>
</tr>
<tr>
<td>REPLACE statement, DATA step</td>
<td>618</td>
</tr>
<tr>
<td>REPORT procedure</td>
<td>768</td>
</tr>
<tr>
<td>RESET statement, SQL procedure</td>
<td>72, 273</td>
</tr>
<tr>
<td>resource management</td>
<td></td>
</tr>
<tr>
<td>assessing needs</td>
<td>652, 653, 654</td>
</tr>
<tr>
<td>benchmark guidelines</td>
<td>656</td>
</tr>
<tr>
<td>comparing probable usage</td>
<td>759, 760, 761</td>
</tr>
<tr>
<td>comparing summarization tools</td>
<td>768, 772</td>
</tr>
<tr>
<td>computer resources</td>
<td>652</td>
</tr>
<tr>
<td>efficiency trade-offs</td>
<td>654</td>
</tr>
<tr>
<td>system options for</td>
<td></td>
</tr>
<tr>
<td>RETAIN statement, DATA step</td>
<td>597</td>
</tr>
<tr>
<td>return codes (FIND method)</td>
<td>567</td>
</tr>
<tr>
<td>REUSE= data set option</td>
<td>690</td>
</tr>
<tr>
<td>REUSE= system option</td>
<td>690</td>
</tr>
<tr>
<td>right outer join</td>
<td>93, 96</td>
</tr>
<tr>
<td>rows</td>
<td></td>
</tr>
<tr>
<td>accessing in tables</td>
<td>227</td>
</tr>
<tr>
<td>counting all</td>
<td>55</td>
</tr>
<tr>
<td>counting number of</td>
<td>55</td>
</tr>
<tr>
<td>deleting in tables</td>
<td>207</td>
</tr>
<tr>
<td>duplicate</td>
<td>31, 130</td>
</tr>
<tr>
<td>eliminating duplicates from output</td>
<td>31</td>
</tr>
<tr>
<td>estimating number of</td>
<td>756</td>
</tr>
<tr>
<td>handling insertion errors</td>
<td>193</td>
</tr>
<tr>
<td>inserting in tables</td>
<td>180, 181, 182</td>
</tr>
<tr>
<td>joining tables with matching values</td>
<td>88</td>
</tr>
<tr>
<td>limiting number displayed</td>
<td>29</td>
</tr>
<tr>
<td>ordering</td>
<td>11, 16</td>
</tr>
<tr>
<td>ordering by multiple columns</td>
<td>12</td>
</tr>
<tr>
<td>processing unique vs. duplicate</td>
<td>130</td>
</tr>
<tr>
<td>restricting processing</td>
<td>265</td>
</tr>
<tr>
<td>row numbers in output</td>
<td>267</td>
</tr>
<tr>
<td>updating based on CASE expression</td>
<td>203, 205, 206</td>
</tr>
<tr>
<td>updating values</td>
<td>198</td>
</tr>
<tr>
<td>updating with different expressions</td>
<td>201</td>
</tr>
<tr>
<td>updating with same expression</td>
<td>199</td>
</tr>
<tr>
<td>RUN statement</td>
<td>737, 738</td>
</tr>
<tr>
<td>RUN-group processing</td>
<td></td>
</tr>
</tbody>
</table>

**S**

**SAS log**

- %PUT statement: 300
- displaying macro variable values: 299, 300
- MCOMPILENOTE= option: 374
- SYMBOLGEN system option: 299
- SASAUTOS= system option: 431

**SASMSTORE system option**: 436

**SASMSTORE= system option**: 433

**SCL programs**

- macro variables in: 362, 363
- SYMGETN function: 363
- SYMPUTN routine: 363

**security, table views**: 254

**SELECT clause**

- CREATE TABLE statement (SQL): 178
- INSERT statement (SQL): 185
- SELECT statement (SQL): 8, 14, 28, 29, 31, 48, 49

**SELECT statement, FORMAT procedure**

- best practices: 710
- managing formats: 588, 599

**SELECT statement, SQL procedure**

- best practices: 710
- CASE expression: 206
- creating new columns: 9
- DISTINCT keyword: 31
- FEEDBACK option: 29
- FROM clause: 10, 15, 83, 105
- functionality: 7, 26
- general form: 27
- general form for inner join: 85, 100
- general form for outer join: 94
- general form for set operations: 127
- GROUP BY clause: 17, 48, 49, 51, 54, 58
- HAVING clause: 20, 58
- INTO clause: 354, 356, 359
- LABEL= option: 43
- NOPRINT option: 354
- ORDER BY clause: 11, 12, 16
- ordering by multiple columns: 12
- ordering by multiple columns: 12
- ordering rows: 11
- querying multiple tables: 13
- SELECT clause: 8, 14, 28, 29, 31, 48, 49
- selecting columns: 8
- specifying columns in multiple tables: 14
- specifying multiple table names: 15
- specifying subset criteria: 11
- specifying tables: 10
- summarizing groups of data: 48, 49
- VALIDATE keyword: 72
- viewing all columns: 28
- WHERE clause: 11, 15, 32, 40
- SELECT/WHEN statement: 716

**semicolon (;) in macro programs**: 375

**sequential access**: 748

**semicolon (;) in macro programs**: 375

**session-compiled macros**: 422

**SET clause**

- INSERT statement (SQL): 180, 181
- UPDATE statement (SQL): 199
set operations
combining and overlaying columns 130
defined 127
EXCEPT set operator 129, 132, 135, 136
INTERSECT set operator 129, 139, 140, 141, 142
modifying results via keywords 131
OUTER UNION set operator 129, 151, 156
processing multiple operations 128
processing single operations 128
processing unique vs. duplicate rows 130
UNION set operator 129, 144, 146, 147, 148
SET statement, DATA step
hash objects and 565
KEY= option 522
multiple 514, 515
POINT= option 689
simple indexes
creating 232
defined 227, 448, 449
SORT procedure
WHERE statement 723
sorting
ordering by multiple columns 12
ordering rows 11
sounds-like conditional operator
description 33
general form 39
WHERE conditions and 755
SOURCE entries, storing 425, 426, 427
SOURCE option, %MACRO statement 435, 437
sparse match 499
special characters, masking
%BQUOTE function 305
%NRSTR function 305
%STR function 303
macro function support 302
special tokens 298
spelling variations 39
SQL procedure
accessing metadata 264
combining data horizontally 507, 511, 512, 514
creating macro variables 354, 356, 359
detail reports 765
DOUBLE option 268
ERRORSTOP option 279
EXEC option 279
FEEDBACK option 29
FLOW option 269
functionality 4
general form 6, 102
INOS= option 265
INSERT statement 180
joinsing data 506
LOOPS= option 279
managing indexes 454
NODouble option 268
NOERRORSTOP option 279
NOEXEC option 71, 279
NOFLOW option 269
NONUMBER option 267
NOSPROMPT option 267, 279
NOSTIMER option 271
NUMBER option 267
OUTOBS= option 29, 265
PROMPT option 267, 279
specifying options 265
STIMER option 271
tools for summarizing data 768
UNDO_POLICY option 194
unique features 5
writing steps 6
SQLLOOPS macro variable 279
STATS system option 655
STIMER option, SQL procedure 271
STIMER system option 271, 655
storage
array values 542, 543, 544, 546
best practices 735
compressing data files 685, 686, 687, 689, 690, 692
conserving with DATA step views 696, 697, 698, 699, 700, 701
in autocall libraries 426, 429, 430, 431
macro definitions in catalog SOURCE entries 425, 426, 427
macro definitions in external files 423
permanent locations for formats 592
raw data filenames in data sets 485
raw data filenames in external files 485
reducing for character variables 677
reducing for numeric variables 677, 678, 679, 680
session-compiled macros 422
Stored Compiled Macro Facility 433, 434, 435, 436, 437
STORE option, %MACRO statement 434
STORED Compiled Macro Facility 433, 434, 435, 436, 437
SUBMIT block 362
subqueries
correlation operator in 65
correlated 62, 229
noncorrelated 62
subsetting data 61
subsetting columns in tables 175
subsetting data
  correlated subqueries and 69
  indexes and 230
  noncorrelated subqueries and 63
  subqueries and 61
subsetting IF statement 708, 725, 727
subsetting rows
  functionality 15
  specifying criteria 11
  using calculated values 40
  using conditional operators 32
subsetting variables 730, 731
SUBSTR function 547, 754, 755
sum statement 519
summarizing data
  comparing tools 767, 768, 772, 774
  GROUP BY clause, SELECT statement (SQL) 17, 48, 49
  SELECT clause and 49
summary functions
  functionality 18, 48
  GROUP BY clause and 51, 54
  number of arguments 48
  on groups of data 49
  SELECT clause and 49
  with columns outside function 52
  with multiple arguments 51
  with single arguments 50
SUMMARY procedure
  multilabel formats 582
  tools for summarizing data 768
summary statistics
  class variables and 771, 772
  combining data and 516, 517, 519
  MEANS procedure 517, 771
  sum statement 519
symbol tables
  %GLOBAL statement 388
  %LOCAL statement 389
  defined 290
  global 290, 387
  local 388, 389, 392
SYMBOLGEN system option 299
SYMGET function 352, 362
SYMGETN function 363
SYMPUT routine
  creating multiple macro variables 343
  DATA step expressions and 336
  DATA step variables and 334
  general form 332, 344
  SCL program support 362
SYMPUTN routine 363
SYMPUTX routine 341
SYSDATE automatic macro variable 292
SYSDATE9 automatic macro variable 292, 310, 314
SYSDAY automatic macro variable 292
SYSERR automatic macro variable 292
SYJOBID automatic macro variable 292
SYSLAST automatic macro variable 292, 312
SYSPARM automatic macro variable 292
SYSSCP automatic macro variable 292
SYSTENV automatic macro variable 292
SYSTIME automatic macro variable 292, 314
SYSVER automatic macro variable 292
T
table aliases, specifying 89
TABLE statement, TABULATE procedure 582
tables
  base 497
  copying 178
  creating from query results 177
  creating like others 174
  creating output tables 19
  creating with integrity constraints 187, 188
  displaying integrity constraints 197
  displaying structure 173
  dropping 216
  empty 168, 174
  generating Cartesian products 83
  joining with SQL procedure 506
  methods of creating 168
  querying multiple 13
  specifying 10
  specifying data types 170
  specifying multiple table names 15
  symbol 290
  updating values 198
  views and 106, 109, 248, 254
  virtual 249
TABULATE procedure
  CLASS statement 582
  FORMAT statement 582
  FORMAT= option 582
  MLF option 582
  multilabel formats 582
  TABLE statement 582
  tools for summarizing data 768
  testing performance 271
text, macro variable references and 316
TITLE statement, SQL procedure 44
titles, specifying 44
tokens
  literal 297
  macro triggers and 298
macro variables and 297
name 298
number 298
percent sign and 304
special 298
transaction data sets 525, 526, 611
TRANSPOSE procedure
adding variable names 551
BY statement 553
general form 548
NAME= option 551
NOTSORTED option 554
PREFIX= option 551
RENAME= data set option 552
VAR statement 549
transposed data sets 553, 554, 555, 556, 557
TRIM function 754
TYPE= data set option 630
TYPES statement, MEANS procedure 776

U
UNDO_POLICY option, SQL procedure 194
UNIFORM option, PRINT procedure 700
UNION set operator
   ALL keyword and 146, 148
   CORR keyword and 147, 148
   functionality 129, 144
UNIQUE constraint type
   constraint specification 191
   functionality 187, 188, 620
   in column specification 188
   unique indexes 228, 233
UNIQUE option
   DATA step 450
   MODIFY statement (DATA) 617
UNIVARIATE procedure 711, 768
update process
   controlling 618, 619
   for views 255
UPDATE statement, DATA step 526
UPDATE statement, SQL procedure
   audit trails and 628
   CASE expression 201, 203
   controlling UNDO processing 194
   functionality 72
   general form 199
   SET clause 199
   updating table row values 198
   updating views 255
   WHERE clause 199, 201, 255
UPDATE MODE= option, MODIFY statement (DATA) 614
user variables 633
user-defined macro variables
   %LET statement and 294
   defined 290
   global symbol table and 387
   USER_VAR statement 633
   USING clause, CREATE VIEW statement (SQL) 253

V
VALIDATE keyword 72
VALUE statement, FORMAT procedure
   combining data 502
   creating formats 501, 580
   creating formats with overlapping ranges 581
   MULTILABEL option 581, 582
   value/identifier pairs 448
VALUES clause, INSERT statement (SQL)
   functionality 180, 182
   handling errors for row insertions 194
VAR statement, TRANSPOSE function 549
variables
   accumulator 519
   adding descriptive names 551
   appending with different lengths 480
   appending with different types 482
   associating formats with 501
   audit trail 631, 632
   best practices 724
   character 677
   class 771, 772, 774, 776, 778
   hash objects and 565
   key 497
   listing in data sets 596
   numeric 677, 678, 679, 680
   renaming 460
   subsetting 730, 731
   summary statistics and 771, 772
   user 633
   WHERE conditions and 755
VIEW= option, DATA step 698
views
   benefits 249
   conserving storage space 696, 697, 698, 699, 700, 701
   creating 249
   deleting 257
   displaying definitions 251
   dropping 257
   enhancing table security 254
   functionality 250, 361
   in queries 248
in-line 105, 106, 107
librefs and 253
managing 252
tables and 106, 109, 248, 254
updating 255
VIEWTABLE window 628
virtual tables 249

WHERE condition
compound optimization and 755
controlling index usage 763
not optimized 755
optimizing 754
printing centile information 757
WHERE statement
DATA step 725, 727, 747, 749, 750, 752
PRINT procedure 727
SORT procedure 723
WHERE= data set option 780
word scanner
macro triggers and 298
tokenization and 297
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore for additional books and resources.