THE POWER TO KNOW®

Beyond the Basics of SAS® Macro

Course Notes

Beyond the Basics of SAS® Macro Course Notes was developed by Michele Ensor. Additional contributions were made by John McCall. Instructional design, editing, and production support was provided by the Learning Design and Development team.

Beyond the Basics of SAS® Macro Course Notes

Book code E71609, course code SGF20BMA, prepared date 11Feb2020. SGF20BMA_001

Table of Contents

To learn more…

For information about other courses in the curriculum, contact the SAS Education Division at 1-800-333-7660, or send e-mail to training@sas.com. You can also find this information on the web at http://support.sas.com/training/ as well as in the Training Course Catalog.

For a list of SAS books (including e-books) that relate to the topics covered in this course notes, visit https://www.sas.com/sas/books.html or call 1-800-727-0025. US customers receive free shipping to US addresses.

Lesson 1 Macro Basics

1.1 The “Straightforward” Basics

1.01 Multiple Choice Question

What is the overall purpose of the SAS macro facility?

a. data creation
b. date manipulation
c. mathematics calculation
d. procedure execution
e. text substitution

3

SAS Macro Facility

The SAS macro facility is a text substitution facility for reducing the amount of text that you must enter to do common tasks.

The macro facility has two components:

Macro Processor	Macro Language
the portion of Base SAS that does the macro work	the syntax that communicates with the macro processor

5

1.02 Multiple Choice Question

What are the two main tools of the macro language?

a. macro data sets
b. macro procedures
c. macro programs
d. macro styles
e. macro variables

6

Macro Tools

Macro variables and programs are the two main tools of the macro language.

Macro Variables	Tends to be text substitution on a smaller and simpler scale.
Macro Programs	Tends to be text substitution on a larger and more complex scale through the use of macro variables and macro statements.

8

1.03 Multiple Choice Question

Which two symbols trigger the macro processor?

a. #
b. $
c. %
d. &
e. *

9

Macro Triggers

The ampersand (&) followed by a name references a macro variable.

```
&dsn
```

```
&location
```

```
&month
```

The percent sign (%) followed by a name references a macro program, a macro statement, or a macro function.

```
%dsn
```

```
%location
```

```
%month
```

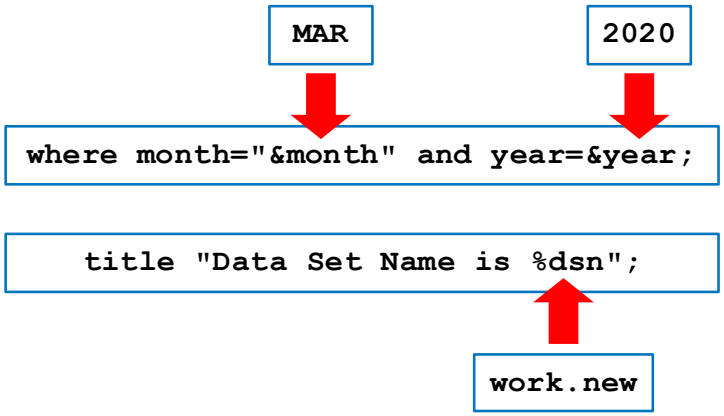
```
%do year = 2010 %to 2020;
```

```
%substr(&date,6,4)
```

11

Macro Triggers within Quotes

If a macro trigger is referenced in a quoted string, double quotation marks must be used in order for the macro trigger to be processed.



12

1.04 Multiple Choice Question

What are three methods for creating a macro variable?

a. %INCLUDE statement
b. %LET statement
c. CALL SYMPUTX routine in the DATA step
d. CRMACRO function in the DATA step
e. INTO clause in the SQL procedure
f. MODIFY statement in the DATASETS procedure

13

Macro Variables

Macro variables can be created using the following methods:

- %LET statement

%LET *macro-variable* **=** *value*;

- CALL SYMPUTX routine in the DATA step

CALL SYMPUTX(*macro-variable*, *value***);**

- INTO clause in the SQL procedure

SELECT *value* **INTO :***macro-variable*

15

%LET Statement

The %LET statement is a stand-alone statement that creates a macro variable and assigns it a value.

```
%let YearCurr=2020;
%let YearPrev=2019;
```

The macro processor processes any syntax containing macro triggers such as macro functions.

```
%let YearCurr=%sysfunc(year(%sysfunc(today())));
%let YearPrev=%eval(&YearCurr-1);
```

16

CALL SYMPUTX Routine in the DATA Step

The CALL SYMPUTX routine assigns a value produced in a DATA step to a macro variable.

```
data _null_;
   YC=year(today());
   YP=YC-1;
   call symputx('YearCurr',YC);
   call symputx('YearPrev',YP);
run;
```

Note: The CALL SYMPUTX routine removes both leading and trailing blanks.

17

CALL SYMPUTX Routine in the DATA Step

In the following example, a different macro variable with the prefix of **student** is created on each iteration of the DATA step. In addition, the macro variable **count** is created on the last iteration.

```
data _null_;
   set sashelp.class end=last;
   call symputx(cat('student',_N_),name);
   if last=1 then call symputx('count',_N_);
run;
```

18

INTO Clause in the SQL Procedure

The INTO clause assigns a value produced in the SELECT clause of the SQL procedure to a macro variable.

```
proc sql;
   select put(max(year),4.)
      into :YearMax
      from work.prdsale;
quit;
```

19

:macro-variable

specifies one or more macro variables. Leading and trailing blanks are not trimmed from values before they are stored in macro variables.

INTO Clause in the SQL Procedure

This example creates two macro variables, one for the maximum year value and one for the minimum year value.

```
proc sql;
   select put(max(year),4.),
          put(min(year),4.)
      into :YearMax, :YearMin
      from work.prdsale;
quit;
```

20

INTO Clause in the SQL Procedure

This example creates one macro variable with each distinct value of **make** separated by a blank.

```
select distinct make
   into :CarMakes separated by ' '
   from sashelp.cars;
```

The following example creates a macro variable per each of the 38 distinct values of **make**.

```
select distinct make
   into :Make1-:Make38
   from sashelp.cars;
```

21

:*macro-variable* SEPARATED BY '*characters*'

specifies one macro variable to contain all the values of a column with values separated by one or more characters. Leading and trailing blanks are trimmed from values before they are stored in the macro variable.

:*macro-variable-1* – :*macro-variable-n*

specifies a numbered list of macro variables. Leading and trailing blanks are trimmed from values before they are stored in macro variables.

INTO Clause in the SQL Procedure

The following example also creates a macro variable per each distinct value of **make**. However, SQL determines the number of macro variables to create. For this example, there are 38 distinct values of **make**, so there will be 38 macro variables created.

```
select distinct make
   into :Make1-
   from sashelp.cars;
```

- The automatic macro variable **SQLOBS** is created when the SQL procedure is used. **SQLOBS** contains the number of rows produced by a SELECT statement.
- For this example, **SQLOBS** will be equal to 38.

22

sas

1.05 Multiple Choice Question

Where are macro variables stored?

a. local and global symbol tables
b. macro data sets
c. raw data files
d. SAS catalogs and item stores
e. symbolgen tables

23

sas

Macro Variable Storage

When a macro variable is created, the macro processor adds the macro variable to a symbol table.

Symbol Table	
sysdate9	29MAR2020
location	District of Columbia
month	MAR
year	2020
dsn	sashelp.class

memory

There are two types of symbol tables: global and local.

25

Global and Local Symbol Tables

Global Symbol Table	• Created at the beginning of a SAS session. • Exists for the duration of the SAS session. • Stores automatic macro variables, macro variables defined outside a macro program, and macro variables defined as global inside a macro program.
Local Symbol Table	• Created when a macro program starts executing. • Exists only during execution of the macro program. • Stores macro variables defined as local inside a macro program.

26

1.06 Multiple Choice Question

Which statement displays the value of a macro variable in the SAS log?

a. %GLOBAL statement
b. %LOG statement
c. %MACRO statement
d. %PUT statement
e. %VALUE statement

27

%PUT Statement

The %PUT statement writes text and the values of macro variables to the SAS log.

```
%put _all_;
```

```
14   %put _all_;
GLOBAL DSN sashelp.class
GLOBAL MONTH MAR
GLOBAL LOCATION District of Columbia
GLOBAL YEAR 2020
 ...
AUTOMATIC SYSDATE 29MAR20
AUTOMATIC SYSDATE9 29MAR2020
 ...
```

Note: _ALL_ lists the values of all user-generated and automatic macro variables.

29

%PUT Statement

```
%put &month &year;
```

```
18   %put &month &year;
MAR 2020
```

```
%put &=location;
```

```
19   %put &=location;
LOCATION=District of Columbia
```

```
%put Today is &sysdate9;
```

```
20   %put Today is &sysdate9;
Today is 29MAR2020
```

30

1.07 Multiple Choice Question

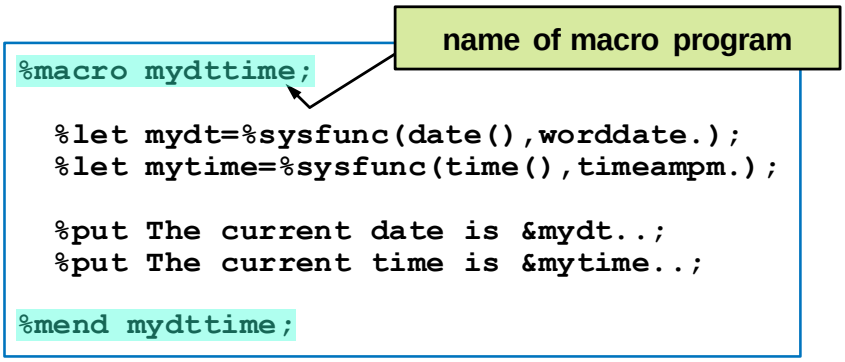
Which statements start and end a macro program?

a. %DO and %END statements
b. %INPUT and %PUT statements
c. %MACRO and %MEND statements
d. %MBEGIN and %MSTOP statements
e. %START and %END statements

31

Macro Programs

A macro program starts with the %MACRO statement and ends with the %MEND statement.



33

Macro Programs

To call (or invoke) a macro, precede the name of the macro program with a percent sign (%).

```
%mydttime
```

Note: A semicolon is not needed at the end of a macro call.

```
2926  %mydttime
The current date is March 29, 2020.
The current time is 4:01:09 PM.
```

34

1.08 Multiple Choice Question

Which two methods are valid for specifying parameters in a macro program?

a. data set parameters
b. keyword parameters
c. positional parameters
d. sorted parameters
e. tab delimited parameters

35

Macro Program Parameters

A macro program can include a parameter list that names one or more local macro variables whose values you specify when you invoke the macro.

Positional Parameters:

```
%macro cars(mke, cyl);
```

Keyword Parameters:

```
%macro cars(mke=Audi, cyl=6);
```

Note: A parameter list can contain any number of macro parameters separated by commas.

37

Positional Parameters

Positional parameters can be specified in any order in the %MACRO statement. The order of the parameters when invoking the macro program must match the order of the %MACRO statement.

```
%macro cars(mke, cyl);
proc print data=sashelp.cars;
   where make="&mke" and cylinders=&cyl;
   title "&mke &cyl Cylinders";
run;
%mend cars;

%cars(Ford,8)
```

38

Keyword Parameters

Keyword parameters can be specified with a default value in the %MACRO statement. The order of the parameters when invoking the macro program does not matter. However, the macro variable name must be specified if overriding the default value.

```
%macro cars(mke=Audi, cyl=6);
proc print data=sashelp.cars;
   where make="&mke" and cylinders=&cyl;
   title "&mke &cyl Cylinders";
run;
%mend cars;

%cars(cyl=4)
```

39

1.09 Multiple Choice Question

Which two sets of statements are valid in a macro program?

a. %ARRAY and %DO statements
b. %DO and %END statements
c. %DROP and %KEEP statements
d. %IF and %WHERE statements
e. %IF - %THEN and %ELSE statements

40

Iterative and Conditional Processing

The following statements must exist in a macro program.

- Iterative statements:

```
%DO macro-variable = start %TO stop;

%END;
```

- Conditional processing statements:

```
%IF expression %THEN action;
< %ELSE %IF expression %THEN action; >
< %ELSE action; >
```

42

The macro conditional processing statements can be in open code (outside of a macro program) if the *action* that is associated with both the %THEN and %ELSE statements include a %DO statement.

Iterative Processing

The %DO iterative statement executes a section of a macro program repetitively based on the value of an index variable.

```
%macro allcars;
%do i = 1 %to 38;
   proc print data=sashelp.cars noobs;
      where make="%scan(&makes,&i,~)";
      title "Make %scan(&makes,&i,~)";
   run;
%end;
%mend allcars;

%allcars
```

43

continued...

Conditional Processing

The %IF - %THEN / %ELSE statements conditionally process a portion of a macro program.

```
%macro carmake(mke);
%if &mke=Chevrolet or &mke=Toyota %then %do;
   proc freq data=sashelp.cars;
      where make="&mke";
      tables cylinders;
      title "Make &mke";
  run;
%end;
```

44

Conditional Processing

```
%else %do;
   proc print data=sashelp.cars;
      where make="&mke";
      title "Make &mke";
   run;
%end;
%mend carmake;

%carmake(Chevrolet)
%carmake(Toyota)
%carmake(Acura)
```

45

1.10 Multiple Choice Question

Which SAS system options are useful for debugging in the macro facility?

a. MLOGIC
b. MPRINT
c. MTEST
d. SYMBOLGEN
e. SYSPARM

46

Macro System Options

The following are SAS system options that are beneficial for debugging in the macro facility:

MLOGIC	Specifies whether the macro processor traces its execution for debugging in the SAS log.
MPRINT	Specifies whether SAS statements generated by macro execution are displayed in the SAS log.
SYMBOLGEN	Specifies whether the results of resolving macro variable references are written to the SAS log.

48

Macro System Options

```
options mlogic mprint symbolgen;
%carmake(Chevrolet)
```

```
MLOGIC(CARMAKE):  Beginning execution.
MLOGIC(CARMAKE):  Parameter MKE has value Chevrolet
SYMBOLGEN:  Macro variable MKE resolves to Chevrolet
SYMBOLGEN:  Macro variable MKE resolves to Chevrolet
MLOGIC(CARMAKE):  %IF condition &mke=Chevrolet or &mke=Toyota is TRUE
MPRINT(CARMAKE):   proc freq data=sashelp.cars;
SYMBOLGEN:  Macro variable MKE resolves to Chevrolet
MPRINT(CARMAKE):   where make="Chevrolet";
MPRINT(CARMAKE):   tables cylinders;
SYMBOLGEN:  Macro variable MKE resolves to Chevrolet
MPRINT(CARMAKE):   title "Make Chevrolet";
MPRINT(CARMAKE):   run;
```

49

1.2 The "Not So Straightforward" Basics

1.11 Multiple Choice Question

Which selection is a valid reference to the macro variable **name**?

a. %name.
b. %name%
c. &name.
d. &name&

51

Macro Variable Reference with Period Delimiter

A macro variable is typically referenced with an ampersand preceding the macro variable name.

```
&city
```

```
&county
```

```
&location
```

A macro variable can also be referenced with an ampersand, the macro variable name, and a period. The period forces the macro processor to recognize the end of the reference and does not appear in the resulting text.

```
&city.
```

```
&county.
```

```
&location.
```

53

Macro Variable Reference with Period Delimiter

The period in a macro variable reference is useful when a macro variable precedes constant text.

```
%let month=MAR;
%let year=2020;

%put &month&year;
```

→ MAR2020

```
%put &month&yeardata;
```

→ WARNING: Apparent symbolic reference YEARDATA not resolved.

```
%put &month&year.data;
```

→ MAR2020data

54

Macro Variable Reference with Period Delimiter

Use two periods if you need a period to follow the text resolved by the macro variable.

```
%let lib=sashelp;
%let dsn=class;

proc print data=&lib.&dsn;
run;
```

→ ERROR: File WORK.SASHELPCLASS does not exist.

```
proc print data=&lib..&dsn;
run;
```

→ sashelp.class

55

1.12 Multiple Choice Question

How many times does SAS scan the syntax **&&city&year**?

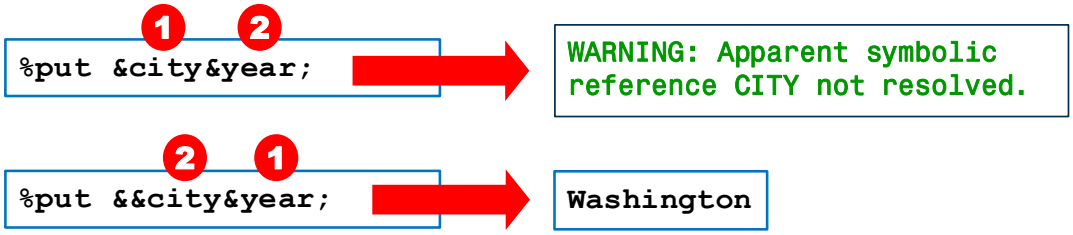
a. zero
b. one
c. two
d. three

56

Multiple Ampersands

Typically, macro variables are resolved reading left to right. Multiple ampersands are needed to resolve a macro variable out of order.

```
%let year=2020;
%let var=location;
%let city2020=Washington;
%let location2020=District of Columbia;
```



58

Multiple Ampersands

Multiple ampersands force the macro processor to scan the macro variable reference more than once.

- Two ampersands resolve to one ampersand.
- Scanning continues until all macro variables are resolved.

```
%let year=2020;
%let var=location;
%let city2020=Washington;
%let location2020=District of Columbia;
```

```
%put &&&var&year;
```

```
?
```

59

Multiple Ampersands

Multiple ampersands force the macro processor to scan the macro variable reference more than once.

- Two ampersands resolve to one ampersand.
- Scanning continues until all macro variables are resolved.

```
%let year=2020;
%let var=location;
%let city2020=Washington;
%let location2020=District of Columbia;
```

```
%put &&&var&year;
```

```
District of Columbia
```

60

1.13 Multiple Choice Question

Which of the following are macro quoting functions that mask the special meaning of text in a macro program statement in open code?

a. %MRSTR
b. %MQUOTE
c. %NRSTR
d. %QUOTE
e. %STR

61

Compilation Macro Quoting Functions

The following are macro quoting functions that mask special characters and mnemonic operators in constant text at macro compilation.

%STR	Masks the following special characters and mnemonic operators: + − * / < > = ¬ ^ ~ ; , # blank AND OR NOT EQ NE LE LT GE GT IN Also, masks the following characters when they are marked by a preceding percent sign: ' " ( )
%NRSTR	Masks the & and % in addition to the characters and mnemonic operators masked by %STR.

63

Compilation Macro Quoting Functions

The semicolon is being masked with the %STR function.

```
%let step=data new%str(;) x=1%str(;);
```

The apostrophe (unmatched quotation mark) is being masked with the %STR function and an extra percent sign (%).

```
%let title=Employee%str(%')s Report;
```

The ampersand is being masked with the %NRSTR function (No Resolve).

```
%let company=R%nrstr(&)D as of &sysdate9;
```

64

1.14 Multiple Choice Question

Which example of %SUPERQ has correct syntax to mask special characters in the macro variable **company**?

a. %superq(company)
b. %superq(&company)
c. %superq(%company)
d. %superq(nr,&company)

65

Execution Macro Quoting Functions

The %SUPERQ macro quoting function masks all special characters and mnemonic operators at macro execution.

```
%macro check(company);
%if ABC ne %superq(company) %then %do;
   %put Not a match;
%end;
%else %put Match;
%mend check;

%check(OR Insurance);
```

Note: The argument of %SUPERQ is the name of a macro variable with no leading ampersand.

67

Execution Macro Quoting Functions

%SUPERQ prevents the resolution of macro variables and macro references in the value of the specified macro variable.

```
data _null_;
   call symput('company','Smith&Jones');
run;

%let newcmpy=%superq(company);
%put The new company is &newcmpy;
```

```
1611  %put The new company is &newcmpy;
The new company is Smith&Jones
```

68

1.15 Multiple Choice Question

Which option enables the macro processor to recognize the IN operator?

a. INOP=YES
b. INOPERATOR
c. MACINOP
d. MACROIN=TRUE
e. MINOPERATOR

69

MINOPERATOR Option

The MINOPERATOR option specifies that the macro processor recognizes and evaluates the mnemonic IN as a logical operator.

```
%macro carsubset(type) / minoperator;
%if &type in SUV Truck Wagon %then %do;
...
%if &type in (Sedan Sports) %then %do;
...
```

- Parentheses are optional.
- The # symbol can be used in place of the word IN.
- The default delimiter for list elements is a blank.

71

MINDELIMITER Option

The MINDELIMITER option specifies the character to be used as the delimiter for the macro IN operator.

```
%macro carsubset(type) / minoperator
                         mindelimiter=',';
%if &type in SUV,Truck,Wagon %then %do;
...
%if &type in (Sedan,Sports) %then %do;
...
```

1.3 Solutions

Solutions to Activities and Questions

1.01 Multiple Choice Question – Correct Answer

What is the overall purpose of the SAS macro facility?

a. data creation
b. date manipulation
c. mathematics calculation
d. procedure execution
e. text substitution

4

1.02 Multiple Choice Question – Correct Answer

What are the two main tools of the macro language?

a. macro data sets
b. macro procedures
c. macro programs
d. macro styles
e. macro variables

7

1.03 Multiple Choice Question – Correct Answer

Which two symbols trigger the macro processor?

a. #
b. $
c. %
d. &
e. *

10

1.04 Multiple Choice Question – Correct Answer

What are three methods for creating a macro variable?

a. %INCLUDE statement
b. %LET statement
c. CALL SYMPUTX routine in the DATA step
d. CRMACRO function in the DATA step
e. INTO clause in the SQL procedure
f. MODIFY statement in the DATASETS procedure

14

1.05 Multiple Choice Question – Correct Answer

Where are macro variables stored?

a. local and global symbol tables
b. macro data sets
c. raw data files
d. SAS catalogs and item stores
e. symbolgen tables

24

1.06 Multiple Choice Question – Correct Answer

Which statement displays the value of a macro variable in the SAS log?

a. %GLOBAL statement
b. %LOG statement
c. %MACRO statement
d. %PUT statement
e. %VALUE statement

28

1.07 Multiple Choice Question – Correct Answer

Which statements start and end a macro program?

a. %DO and %END statements
b. %INPUT and %PUT statements
c. %MACRO and %MEND statements
d. %MBEGIN and %MSTOP statements
e. %START and %END statements

32

1.08 Multiple Choice Question – Correct Answer

Which two methods are valid for specifying parameters in a macro program?

a. data set parameters
b. keyword parameters
c. positional parameters
d. sorted parameters
e. tab delimited parameters

36

1.09 Multiple Choice Question – Correct Answer

Which two sets of statements are valid in a macro program?

a. %ARRAY and %DO statements
b. %DO and %END statements
c. %DROP and %KEEP statements
d. %IF and %WHERE statements
e. %IF - %THEN and %ELSE statements

41

1.10 Multiple Choice Question – Correct Answers

Which SAS system options are useful for debugging in the macro facility?

a. MLOGIC
b. MPRINT
c. MTEST
d. SYMBOLGEN
e. SYSPARM

47

1.11 Multiple Choice Question – Correct Answer

Which selection is a valid reference to the macro variable **name**?

a. %name.
b. %name%
c. &name.
d. &name&

52

1.12 Multiple Choice Question – Correct Answer

How many times does SAS scan the syntax **&&city&year**?

a. zero
b. one
c. two
d. three

57

1.13 Multiple Choice Question – Correct Answer

Which of the following are macro quoting functions that mask the special meaning of text in a macro program statement in open code?

a. %MRSTR
b. %MQUOTE
c. %NRSTR
d. %QUOTE
e. %STR

62

1.14 Multiple Choice Question – Correct Answer

Which example of %SUPERQ has correct syntax to mask special characters in the macro variable **company**?

a. %superq(company)
b. %superq(&company)
c. %superq(%company)
d. %superq(nr,&company)

66

1.15 Multiple Choice Question – Correct Answer

Which option enables the macro processor to recognize the IN operator?

a. INOP=YES
b. INOPERATOR
c. MACINOP
d. MACROIN=TRUE
e. MINOPERATOR

Lesson 2 Doing More with Macro

2.1 SAS I/O Functions

SAS I/O Functions

SAS provides I/O functions that can be used to retrieve metadata information about a SAS data set.

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	11/07/2018 22:40:59	Observation Length	40
Last Modified	11/07/2018 22:40:59	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Student Data		
Data Representation	WINDOWS_64		
Encoding	us-ascii ASCII (ANSI)		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

Note: The SAS I/O functions retrieve information similar to the output of the CONTENTS procedure.

3

SAS I/O Functions

EXIST	Verifies the existence of a SAS library member such as a SAS data set.
OPEN CLOSE	Opens and closes a SAS data set.
ATTRC ATTRN	Returns the value of a character attribute (compressed, engine, or sorted-by) or a numeric attribute (create date or number of observations/variables) of a SAS data set.
VARLABEL VARLEN VARTYPE	Returns an attribute (label, length, or type) of a SAS data set variable.

4

EXIST Function

The macro processor can process the I/O function if used in combination with the %SYSFUNC macro function.

```
%macro data1(dsn);

%if %sysfunc(exist(&dsn))=1 %then %do;
   proc print data=&dsn(obs=10);
   run;
%end;
%else %put Not a valid data set.;

%mend data1;
```

Note: EXIST returns 1 if the data set exists, or 0 if data set does not exist.

OPEN/ATTRN/CLOSE Functions

```
%macro data2(dsn);
%let dsid=%sysfunc(open(&dsn));
%if &dsid ne 0 %then %do;
   %let numobs=%sysfunc(attrn(&dsid,nlobs));
   %let numvar=%sysfunc(attrn(&dsid,nvars));
   %let rc=%sysfunc(close(&dsid));
...
```

- OPEN returns a unique numeric data set identifier if the data set can be opened and returns a 0 if the data set cannot be opened.
- ATTRN and CLOSE use the data set identifier that OPEN returns.
- NLOBS and NVARS specify the number of logical observations and variables.
- CLOSE returns 0 if successful and nonzero if unsuccessful.

VARTYPE Function

```
%macro data3(dsn);
%if %sysfunc(exist(&dsn))=1 %then %do;
   %let dsid=%sysfunc(open(&dsn));
   %let numvar=%sysfunc(attrn(&dsid,nvars));
   %let char=0;
   %do i=1 %to &numvar;
      %if %sysfunc(vartype(&dsid, &i))=C %then %do;
         %let char=%eval(&char+1);
      %end;
   %end;
   %let rc=%sysfunc(close(&dsid));
...
```

- The second argument of VARTYPE specifies the variable position number.
- VARTYPE returns C for a character variable or N for a numeric variable.

7

External Files Functions

External file functions are similar to SAS I/O functions except the external file functions return information that is associated with external files.

DOPEN DCLOSE	Opens and closes a directory.
DNUM	Returns the number of members in a directory.
DREAD	Returns the name of a directory member.

8

External Files Functions

External file functions are similar to SAS I/O functions except the external file functions return information that is associated with external files.

```
%macro runpgms(path);
filename mydir "&path";
%let dirid=%sysfunc(dopen(mydir));
%let dirnum=%sysfunc(dnum(&dirid));

%do i=1 %to &dirnum;
   %include "&path\%sysfunc(dread(&dirid,&i))" / source2;
%end;

%let rc=%sysfunc(dclose(&dirid));
filename mydir clear;
%mend runpgms;
```

2.2 Dictionary Tables

Dictionary Tables

Dictionary tables are special read-only PROC SQL tables. They retrieve information about all the SAS libraries, SAS data sets, SAS system options, and external files that are associated with the current SAS session.

Here are two examples of dictionary tables:

dictionary.tables	Contains information about known tables.
dictionary.columns	Contains information about columns in all known tables.

Note: Dictionary tables can be used only with PROC SQL.

11

Dictionary Tables

To see how each dictionary table is defined, submit a DESCRIBE TABLE statement.

```
proc sql;
   describe table dictionary.tables;
quit;
```

The results are written to the SAS log.

```
create table DICTIONARY.TABLES
  (
   libname char(8) label='Library Name',
   memname char(32) label='Member Name',
   memtype char(8) label='Member Type',
...
```

12

Number of Observations and Variables

In **dictionary.tables**, the **nlobs** and **nvar** variables specify the number of logical observations and variables in a table (**memname**) within a library (**libname**).

```
%let dsn=SASHELP.SHOES;

proc sql noprint;
   select nlobs, nvar
      into :numobs, :numvar
      from dictionary.tables
      where libname="%scan(&dsn,1,.)" and
            memname="%scan(&dsn,2,.)";
quit;
```

13

Tables with a Given Variable

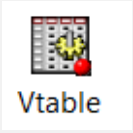
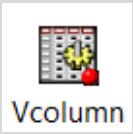
In **dictionary.columns**, the **name** variable specifies the name of each variable in a table (**memname**) within a library (**libname**).

```
%macro freq(lib,var);

proc sql noprint;
   select catx('.',libname,memname)
      into :tables separated by '~'
      from dictionary.columns
      where libname="%upcase(&lib)" and
            upcase(name)="%upcase(&var)";
quit;

...
```

14

Sashelp Views

SAS provides PROC SQL views, based on the dictionary tables, that can be used in other SAS procedures and in the DATA step. These views are stored in the Sashelp library.

dictionary.tables		Sashelp.Vtable
dictionary.columns		Sashelp.Vcolumn

15

Number of Character and Numeric Variables

The **sashelp.vcolumn** data set can be used in a DATA step to count the number of character and numeric variables.

```
%let dsn=SASHELP.CARS;

data _null_;
   set sashelp.vcolumn end=last;
   where catx('.',libname,memname)="&dsn";
   if type='char' then charcount+1;
   else if type='num' then numcount+1;
   if last=1 then do;
      call symputx('ccount',charcount);
      call symputx('ncount',numcount);
  end;
run;
```

16

2.3 DOSUBL Function

DOSUBL Function

The DOSUBL function enables the immediate execution of SAS code after a text string is passed.

DOSUBL(*text-string***);**

- The DOSUBL function is used in a DATA step. It can also be used with %SYSFUNC outside of a step boundary.
- The DOSUBL function returns a value of zero if the SAS code was able to execute, and returns a nonzero value if the SAS code was not able to execute.
- The DOSUBL function is an alternative to the CALL EXECUTE routine.

18

DOSUBL Function with Constant Text

```
data _null_;
   text1='proc contents data=sashelp.class; run;';
   rc=dosubl(text1);
   text2='proc print data=sashelp.class(obs=10); run;';
   rc=dosubl(text2);
run;
```

```
data _null_;
   rc=dosubl('proc contents data=sashelp.class; run;');
   rc=dosubl('proc print data=sashelp.class(obs=10); run;');
run;
```

19

DOSUBL Function with Macro

The following example creates two PROC steps for each data set whose name starts with a given string for a given library:

```
%macro contents(lib,dsnstart);
data _null_;
   set sashelp.vtable(keep=libname memname);
   where libname="%upcase(&lib)"
         and memname:"%upcase(&dsnstart)";
   rc=dosubl('proc contents data='||"&lib"||'.'||
             trim(memname)||'; run;');
   rc=dosubl('proc print data='||"&lib"||'.'||
             trim(memname)||'(obs=10); run;');
run;
%mend contents;

%contents(sashelp,pr)
```

20

DOSUBL Function with Macro

DATA step starts execution.

_N_ = 1	PROC CONTENTS and PROC PRINT execute for SASHELP.PRDSAL2.
_N_ = 2	PROC CONTENTS and PROC PRINT execute for SASHELP.PRDSAL3.
_N_ = 3	PROC CONTENTS and PROC PRINT execute for SASHELP.PRDSALE.
_N_ = 4	PROC CONTENTS and PROC PRINT execute for SASHELP.PRICEDATA.
_N_ = 5	PROC CONTENTS and PROC PRINT execute for SASHELP.PROJ4DEF.

DATA step ends execution.

21

DOSUBL Function Referencing Macro Program

```
%macro twoprocs(table);
proc contents data=&table;
run;
proc print data=&table(obs=10);
run;
%mend twoprocs;
```

```
%macro contents(lib,dsnstart);
data _null_;
   set sashelp.vtable(keep=libname memname);
   where libname="%upcase(&lib)"
         and memname:"%upcase(&dsnstart)";
   table=cats(libname,'.',memname);
   rc=dosubl('%twoprocs('||table||')');
run;
%mend contents;
```

2.4 Macro Termination and Branching

Macro Termination and Branching

The following macro statements are beneficial for terminating or branching within a macro program:

%RETURN statement **%RETURN;**	Causes normal termination of the currently executing macro.
%GOTO statement **%GOTO** *label*;	Branches macro processing to the specified label (the spot where you want execution to branch to).

24

Macro Termination

The %RETURN statement causes normal termination of the currently executing macro.

```
%macro means(dsn,class,var);

%if %sysfunc(exist(&dsn))=0 %then %do;
   %put ERROR: Not a valid data set.;
   %return;
%end;

...   SKIPS

%mend means;

%means(sashelp.xyz,region,sales)
```

```
602  %means(sashelp.xyz,region,sales)
ERROR: Not a valid data set.
```

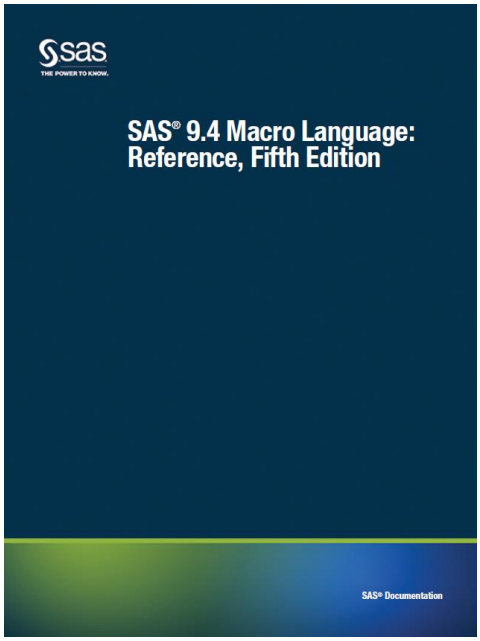
25

Macro Branching

The %GOTO statement branches macro processing to the specified label (the spot where you want execution to branch to).

```
%let posclass=%sysfunc(varnum(&dsid,&class));
   %if &posclass=0 %then %goto invalid;

%let posvar=%sysfunc(varnum(&dsid,&var));
   %if &posvar=0 %then %goto invalid;

%let vart=%sysfunc(vartype(&dsid,&posvar));
   %if &vart=C %then %do;
      %invalid: %put ERROR: Variable is not valid.;
      %let dsid=%sysfunc(close(&dsid));
      %return;
 %end;
```

26



Macro Documentation

http://support.sas.com/documentation/cdl/en/mcrolref/69726/PDF/default/mcrolref.pdf

27