The George Washington University
Department of Statistics

STAT 6197 – Spring 2019
Week 10 – March 29, 2019

Major Topic: Macro Functions, and Working with Macros

Detailed Topics:

1) Macro Quoting Functions
2) Macro Variable Functions
3) Iterative Processing for Generating SAS Code (Sequential vs. Non-Sequential Processing)
4) Conditional Processing Using Macro Facility
5) Generating Data-Dependent SAS Code
6) Saving and Accessing Macros

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

References:

1. Relevant Chapters/Sections - Delwiche L, and Slaughter S. *The Little SAS Book: A Primer*, Fifth Edition Paperback – November 7, 2012

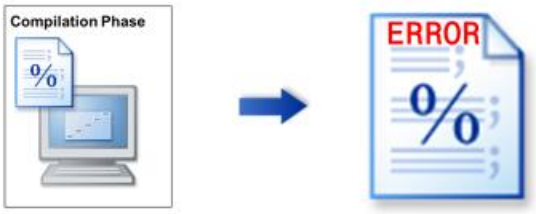
2. Exercises from Relevant Chapters/Sections - Ottesen RA, Delwiche LD, and Slaughter SJ. *Exercises and Projects for The Little SAS Book*, Fifth Edition Paperback – July 1, 2015

3. SAS® 9.4 Macro Language: Reference, Sixth Edition (Online Documentation)

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Business Scenario

Macro programs often contain special characters, which can be misinterpreted during macro compilation.



4

%STR and %NRSTR Functions

Special characters and mnemonics can cause unexpected results during compilation of a macro or a macro language statement open code.

Both %STR and %NRSTR mask the following characters and mnemonics					% NRSTR additionally masks macro triggers
+	=	#	NE	'	%
-	¬	blank	LE	"	&
*	^	AND	LT	)	
/	~	OR	GE	(	
<	;	NOT	GT		
>	,	EQ	IN		

The %STR and %NRSTR macro quoting functions are used to ignore the normal meaning of the special characters such as those listed above and ensure that they are treated as part of the value.

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

3.01 Quiz – Correct Answer

Create a macro variable that, when referenced, clears titles and footnotes.

```
%let clear=title; footnote;
```

What is the problem here?

The first semicolon ends the %LET statement. The macro variable CLEAR stores the text *title* only, without the semicolon.

7

%STR Function

The %STR function is a macro quoting function that ***masks*** the normal meaning of special characters and mnemonics in constant text.

%STR(*any text*)

```
54   %let clear=%str(title; footnote;);
55
56   title 'All Students';
57   footnote 'Fall Semester';
58
59   proc print data=sashelp.class;
60   run;

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.03 seconds
      cpu time            0.01 seconds

61
62   options symbolgen;
63   &clear
SYMBOLGEN:  Macro variable CLEAR resolves to title; footnote;
SYMBOLGEN:  Some characters in the above value which were subject to macro
            quoting have been unquoted for printing.
```

8 m203d01

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%STR Function

The %STR function

- does ***not*** mask macro triggers:
  & %
- requires additional syntax to mask special characters that normally come in pairs:
  " ' ( )

9

%NRSTR Function

The %NRSTR function is similar to the %STR function, but %NRSTR also masks macro triggers.

```
%let company=%nrstr(AT&T);
```

m203d03

29

Ex1_%str_%nrstr.sas

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Protecting Commas

The %STR function masks the comma.

```
%macro name(fullname);
   %let first=%scan(&fullname,2);
   %let last=%scan(&fullname,1);
   %let newname=&first &last;
   %put &newname;
%mend name;

%name(%str(Taylor, Jenna))
```

```
Jenna Taylor
```

m203d02a

12

Protecting Commas

To specify a comma delimiter, use the %STR function.

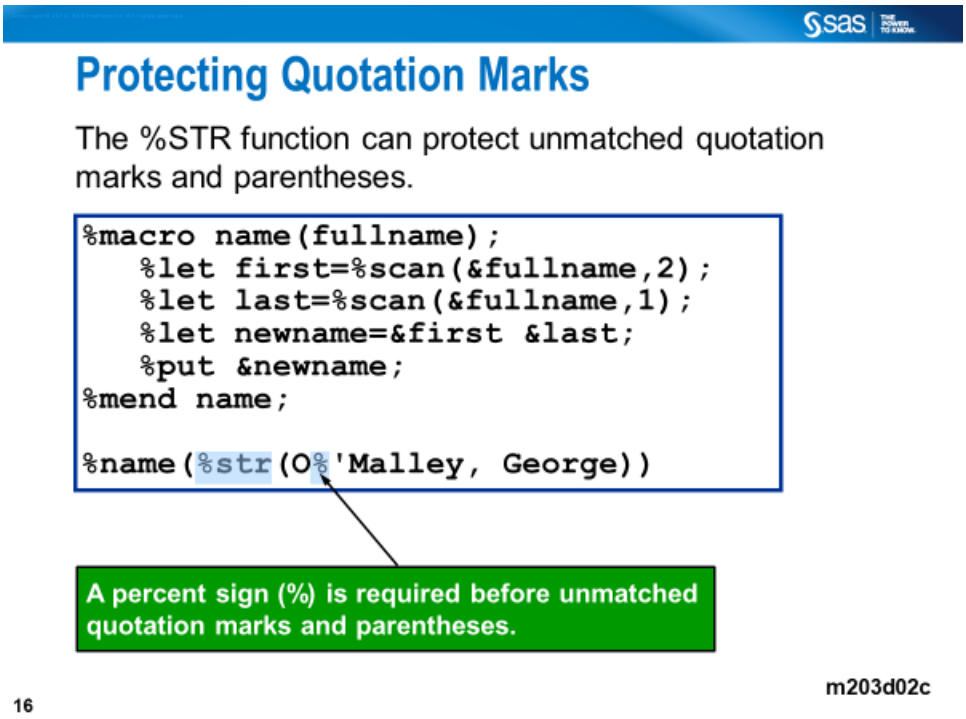
```
%macro name(fullname);
   %let first=%scan(&fullname,2,%str(,));
   %let last=%scan(&fullname,1,%str(,));
   %let newname=&first &last;
   %put &newname;
%mend name;

%name(%str(Taylor, Jenna))
```

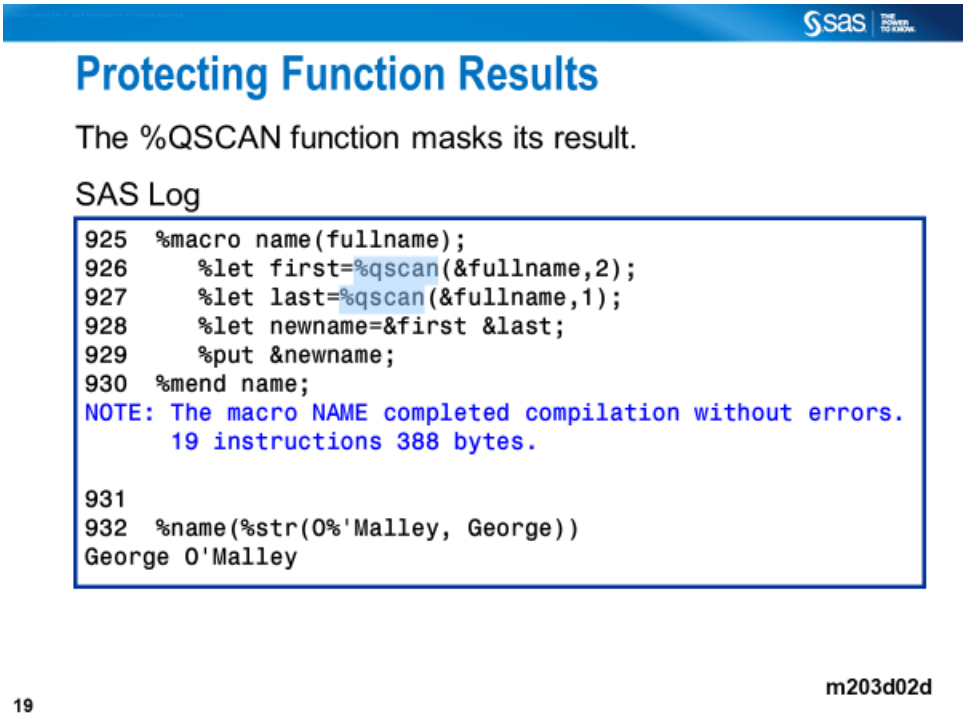
```
Jenna Taylor
```

m203d02b

13

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts





Ex16_scan_qscan_nrbquote.sas

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Protecting Function Results

Many macro functions and autocall macros have "Q" equivalents that return masked results.

%CMPRES	%QCMPRES
%LEFT	%QLEFT
%LOWCASE	%QLOWCASE
%SCAN	%QSCAN
%SUBSTR	%QSUBSTR
%SYSFUNC	%QSYSFUNC
%TRIM	%QTRIM
%UPCASE	%QUPCASE

20

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

```
 1  *Ex1_%str_%nrstr1.sas;
 2  %LET mvar1 = Beth%STR(%'s) Assignment Report;
 3  %LET mvar2 = %STR(Beth%'s Assignment Report);
 4  %let Course=%str(   Stat 4197);
 5  %let step=%str(proc print; run;);
 6  %let mvar3 =  %NRSTR(AT&T);
 7  %let mvar4 =  %NRSTR(%of defective bulbs);
 8  %PUT &=mvar1;
 9  %PUT &=mvar2;
10  %PUT &=mvar3;
11  %PUT &=mvar4;
12  %PUT &=step;
13  %put &=Course;
```

(Partial SAS Log)

```
MVAR1=Beth's Assignment Report
9    %PUT &=mvar2;
MVAR2=Beth's Assignment Report
10   %PUT &=mvar3;
MVAR3=AT&T
11   %PUT &=mvar4;
MVAR4=%of defective bulbs
12   %PUT &=step;
STEP=proc print; run;
13   %put &=Course;
COURSE=   Stat 4197
```

Lines 2-3: Note the use of the percent sign before a quotation mark in the %STR function.

Line 4: %STR preserves the leading blank.

Line 5: Without the %STR function, the macro variable STEP would store the text *proc print* only, without the semicolon; SAS would issue a warning.

Lines 6-7: %NRSTR (no rescan string) prevents rescanning and thereby ignores meaning of & and %.

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Business Scenario

Macro programs often contain special characters, which can be misinterpreted during macro execution.

35

3.07 Quiz – Correct Answer

SAS Log

The text **OR** was misinterpreted as a logical operator.

```
1042  %macro where(state);
1043     %if &state=NC %then %put Southeast;
1044     %else %if &state=OR %then %put Northwest;
1045     %else %put Unknown;
1046  %mend where;
NOTE: The macro WHERE completed compilation without errors.
      21 instructions 404 bytes.
1047
1048  %where(NY)
ERROR: A character operand was found in the %EVAL function or %IF
       condition where a numeric operand is required. The condition
       was: &state=OR
ERROR: The macro WHERE will stop executing.
```

m203d04a

37

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Protecting Constant Text

The %STR function masks mnemonics in constant text.

SAS Log

```
1049  %macro where(state);
1050     %if &state=NC %then %put Southeast;
1051     %else %if &state=%str(OR) %then %put Northwest;
1052     %else %put Unknown;
1053  %mend where;
NOTE: The macro WHERE completed compilation without errors.
      21 instructions 404 bytes.
1054
1055  %where(NY)
Unknown
```

m203d04b

38

Protecting Resolved Values

The %STR function masked the resolved value of **state**.

```
1063  %macro where(state);
1064     %if %str(&state)=NC %then %put Southeast;
1065     %else %if %str(&state)=%str(OR) %then %put Northwest;
1066     %else %put Unknown;
1067  %mend where;
NOTE: The macro WHERE completed compilation without errors.
      21 instructions 408 bytes.
1068
1069  %where(OR)
Northwest
```

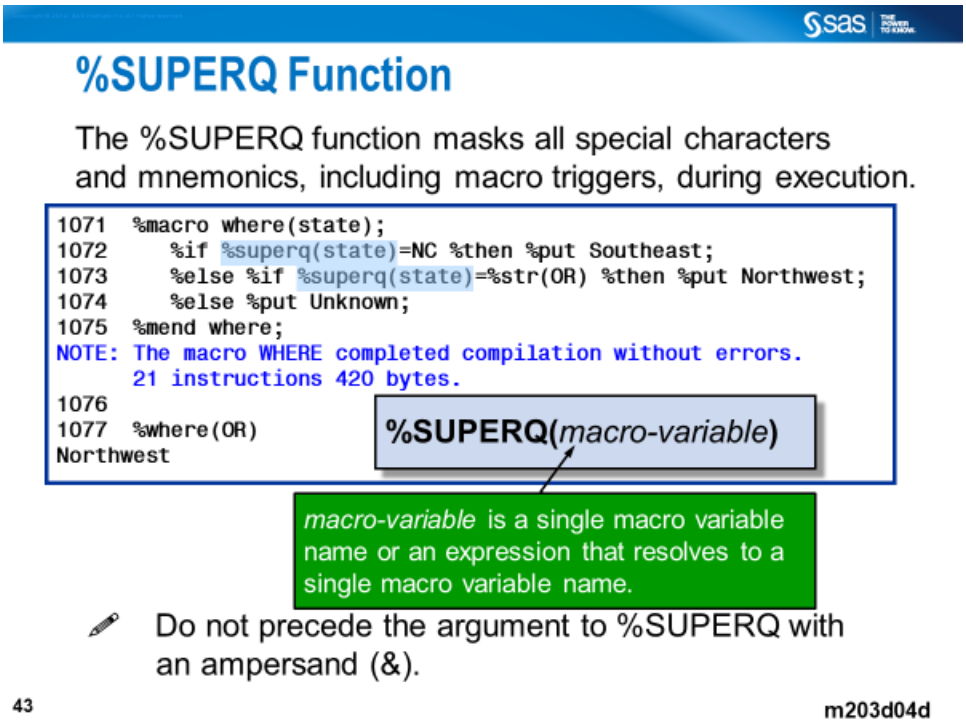
The %STR function is

- ***not*** recommended for masking resolved values
- recommended for masking ***constant text*** only.

m203d04c

42

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts



%SUPERQ Function

The %SUPERQ function masks all special characters and mnemonics, including macro triggers, during execution.

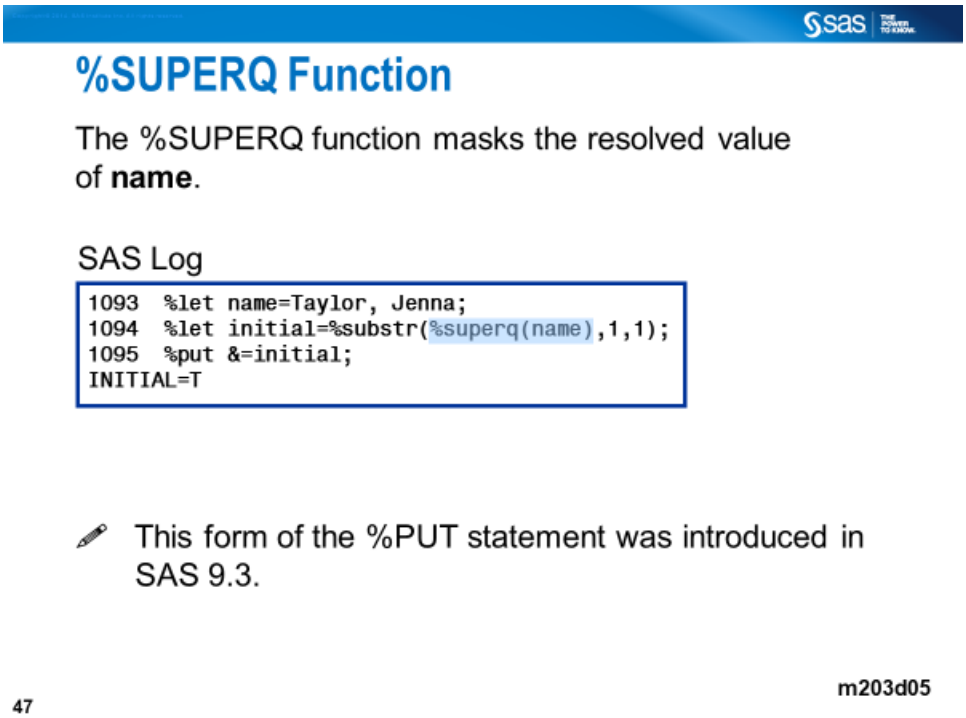
```
1071  %macro where(state);
1072     %if %superq(state)=NC %then %put Southeast;
1073     %else %if %superq(state)=%str(OR) %then %put Northwest;
1074     %else %put Unknown;
1075  %mend where;
NOTE: The macro WHERE completed compilation without errors.
      21 instructions 420 bytes.
1076
1077  %where(OR)
Northwest
```

%SUPERQ(*macro-variable*)

macro-variable is a single macro variable name or an expression that resolves to a single macro variable name.

- Do not precede the argument to %SUPERQ with an ampersand (&).

43 m203d04d



%SUPERQ Function

The %SUPERQ function masks the resolved value of **name**.

SAS Log

```
1093  %let name=Taylor, Jenna;
1094  %let initial=%substr(%superq(name),1,1);
1095  %put &=initial;
INITIAL=T
```

- This form of the %PUT statement was introduced in SAS 9.3.

47 m203d05

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%SUPERQ Function

The %SUPERQ function operates only on the values of the macro variable and mask all items that require quoting during macro execution. (Carpenter, 2016)

```
1  *Ex3_P_Superq.sas;
2  data _Null_;
3  Call symputx('xmvar1', 'AT&T % of Employees Aged 25-49');
4  run;
5  %let xmvar2=%superq(xmvar1);
6  %put &=xmvar2;
```

(Partial SAS Log)

```
XMVAR2=AT&T % of Employees Aged 25-49
```

Line 5: The %SUPERQ function accepts as its argument only the name of a single macro variable with no leading ampersand.

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Tip: Protecting Resolved Values

Execution-time quoting functions protect resolved values.

49

Summary: Macro Quoting Functions

	Macro Triggers & % Not Masked	Macro Triggers & % Masked
Constant text (compile time)	%STR	%NRSTR
Resolved values (execution time)		%SUPERQ

50

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%BQUOTE and %NRBQUOTE

%BQUOTE AND %NRBQUOTE functions mask values during execution of a macro or a macro language statement in open code.

The %BQUOTE (also called blind quote) is used to remove meaning from unanticipated characters in resolved text during macro execution.

```
 1  *Ex2_BQUOTE_NRBQUOTE.sas;
 2  options symbolgen;
 3  data test;
 4     store="Kids'Corner";
 5     call symput('s',store);
 6  run;
 7  %MACRO BQ;
 8     %IF %BQUOTE(&s) NE %THEN %PUT *** valid ***;
 9     %ELSE %PUT *** null value ***;
10  %MEND BQ;
11  %BQ
```

(Partial SAS Log)

```
*** valid ***
```

Line 4: The DATA step assigns the value **Kids' Corner** to the variable STORE, enclosing the character string in double quotation marks.

Line 5: The SYMPUT routine assigns the value of the variable STORE to a macro variable.

Lines 7-17: The macro BQ uses the %BQUOTE function to enable the %IF … macro statement to accept the unmatched single mark.

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%BQUOTE – Another Example

```
13  *Ex2_BQUOTE_NRBQUOTE.sas;
14  %MACRO BQ_x;
15    %LOCAL state;
16    data _null_;
17      State_Name='NE';
18      CALL SYMPUT('State', State_Name);
19    run;
20    %IF %BQUOTE(&State)=%STR(NE) %THEN
21        %PUT Nebraska Dept. of Health;
22  %put &State;
23  %MEND BQ_x;
24  %BQ_x
```

(Partial SAS LOG)

```
SYMBOLGEN:  Macro variable STATE resolves to NE
Nebraska Dept. of Health
SYMBOLGEN:  Macro variable STATE resolves to NE
NE
```

Line 20: The %BQUOTE function is used to resolve the macro variable STATE to NE by removing the meaning of NE (i.e., not equal). The %STR function causes the macro processor to interpret the special character NE as text in this macro program statement while compiling (constructing) the macro.

The %NRBQUOTE function is useful when you want a value to be resolved when first encountered, if possible, but you do not want any ampersands or percent signs in the result to be interpreted as operators by an explicit or implicit %EVAL function." [SAS® Documentation]

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Summary: Macro Quoting Functions

	Macro Triggers & % Not Masked	Macro Triggers & % Masked
Constant text (compile time)	%STR	%NRSTR
Resolved values (execution time)	%BQUOTE	%SUPERQ

59

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Macro Character Functions that apply character string manipulations to the *values of macro variables*

%SUBSTR function extracts part of a string from a macro variable value based on position.

%QSUBSTR function additionally masks special characters and mnemonic operators (no example given)

%LENGTH finds the lengths of character strings and text expressions.

%SCAN function extracts nth word from the value of a macro variable.
%QSCAN function additionally masks special characters and mnemonic operators.
Ex16_scan_qscan_nrbquote.sas

%UPCASE function changes macro variable values from lowercase to uppercase.

%QUPCASE additionally masks special characters and mnemonic operators

%INDEX function determines the location of the first character of a character string of a macro variable value within a source.

Ex5_Some_macro_functions.sas

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%SYSFUNC

%SYSFUNC function executes most of the data step functions. "Because %SYSFUNC is a macro function, you do not need to enclose character values in quotation marks as you do in DATA step functions" (SAS® Documentation).

%QSYSFUNC function additionally masks special characters and mnemonic operators.

%SYSFUNC and PUTN only take macro variables, not data step variables.

Ex4_Percent_sysfunc.sas

Ex6_Sysfunc.sas"

Ex23_Sysfunc_DateStamp.sas

Ex5_Some_macro_functions.sas

Business Scenario

Macro applications might require iterative processing.

The iterative %DO statement can execute macro language statements and generate SAS code.

46

%DO Statement vs. Do Loop

```
1  *Ex13_Percent_Do_DoLoop.sas;
2  *%Do Statement in a Macro;
3  %macro pdo;
4        %local i;
5        %do i=1 %to 5;
6            %put i=&i;
7        %end;
8   %mend pdo;
9   %pdo
```

```
i=1
i=2
i=3
i=4
i=5
```

```
11    * Do Loop in a Data Step;
12   data _Null_;
13    do i = 1 to 5;
14      output;
15      put (_All_) (=);
16    end;
17   run;
```

```
i=1
i=2
i=3
i=4
i=5
```

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Simple Loops

Example: Display a series of macro variables in the SAS log by repeatedly executing %PUT within a macro loop.

```
%macro putloop;
   %do i=1 %to &numrows;
      %put Country&i is &&country&i;
   %end;
%mend;
```

%DO *index-variable=start* **%TO** *stop* <**%BY** *increment*>;
text
%END;

SAS Log

```
200  %putloop
Country1 is Australia
Country2 is Canada
Country3 is Germany
Country4 is Israel
Country5 is Turkey
Country6 is United States
Country7 is South Africa
```

48 m105d07

Simple Loops

%DO *index-variable=start* **%TO** *stop* <**%BY** *increment*>;
text
%END;

- %DO and %END are valid only inside a macro definition.
- *index-variable* is a macro variable created in the local symbol table if it does not already exist in another symbol table.
- *start*, *stop*, and *increment* values can be any valid macro expressions that resolve to integers.
- The %BY clause is optional. (The default *increment* is 1.)
- *text* can be any of the following:
  - constant text
  - macro variables or expressions
  - macro statements
  - macro calls

49

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Generating SAS Code Iteratively with a Macro

```
 1  *Ex22_Percent_DoLoop_Macro.sas;
 2  %macro runit (first=, last=);
 3        %local yr;
 4        %do yr=&first %to &last;
 5          data have_20%sysfunc(putn(&yr,z2.));
 6                exp=20%sysfunc(putn(&yr,z2.))/4;
 7              year=20%sysfunc(putn(&yr,z2.));
 8           run;
 9         %end ;
10  %mend runit;
11  %runit(first=08, last=15);
12
13  data all_yrs;
14   retain year;
15    set Have:;
16  run;
17  proc print data=all_yrs noobs split='*';
18  label year= 'Survey Year'
19        exp= 'Mean expenses*per person per year';
20  run;
```

Lines 2-10: Define the macro.
Line 3: Declare index variable of the macro loop as a local macro variable.
Line 2: Specify the keyword parameters in the %macro statement.
Line 11: Invoke the macro.

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Displaying the Generated SAS Code from a Macro

```
1  *Ex11_MFILE.sas;
2  options nocenter nodate nonumber;
3  %LET path=C:\SASCourse\Week10;
4  * Directing generated code to a SAS file;
5  %macro read_year(start=, stop=);
6    %do i = &start %to &stop;
7      data Data&i;
8        infile "&path\yob&i..txt" DLM=',';
9        input (name sex) ($) count;
10       title "Listing from Data&i Data Set";
11     proc print data=Data&i (obs=5) noobs;
12     run;
13   %end;
14 %mend read_year;
15 Filename mprint "&Path\Ex12_Generated_Code1.sas";
16 options mprint mfile;
17 %read year (start=2012, stop=2015)
```

Open and see the “Generated_Code1.sas” file.

System options for the Macro Language

MLOGIC => Macro logic is examined.

MPRINT => Macro code is written to the SAS Log.

SYMBOLGEN => Macro variables are resolved in the SAS Log.

MFILE => SAS Code is generated in the file specified.

Below is the SAS macro-generated code (Ex12_Generated_Code_SET1.sas).

```
data Data2012;
infile "C:\SASCourse\Week10\yob2012.txt" DLM=',';
input (name sex) ($) count;
title "Listing from Data2012 Data Set";
proc print data=Data2012 (obs=5) noobs;
run;
data Data2013;
infile "C:\SASCourse\Week10\yob2013.txt" DLM=',';
input (name sex) ($) count;
title "Listing from Data2013 Data Set";
proc print data=Data2013 (obs=5) noobs;
run;
data Data2014;
infile "C:\SASCourse\Week10\yob2014.txt" DLM=',';
input (name sex) ($) count;
title "Listing from Data2014 Data Set";
proc print data=Data2014 (obs=5) noobs;
run;
data Data2015;
infile "C:\SASCourse\Week10\yob2015.txt" DLM=',';
input (name sex) ($) count;
title "Listing from Data2015 Data Set";
proc print data=Data2015 (obs=5) noobs;
run;
```

Displaying the Generated SAS Code from a Macro (continued)

***Ex11_MFILE.sas;**

```
18  * Create a macro to generate text on the SET statement;
19  %macro names(prefix, initial, maxnum);
20    %do i=&initial %to &maxnum;
21      &prefix&i
22    %end;
23    ;
24  %mend names;
25  *Concatenate data sets using the macro already created;
26  Filename mprint "&Path\Generated_Code_SET1.sas" ;
27  options mprint mfile;
28  data all;
29    set %names(Data,2012,2015)
30  ;
31  run;
32  proc print data=all; run;
```

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%Do %End Processing with a Non-Sequential List of Values of a Macro Variable

```
 1  *Ex7_%DO_Nonsequential1.sas;
 2  options nonumber nocenter nodate;
 3  %LET list = %str(sashelp.class|
 4                   sashelp.cars|
 5                   sashelp.retail);
 6  /* Count # of values in the string */
 7  %LET count=%sysfunc(countw(&list, %STR(|)));
 8  %macro loop;
 9   /* Loop through the total # of data sets */
10   %do i = 1 %to &count;
11     title  "%left(%SCAN(&list, &i, %STR(|)))";
12     proc print data=%scan(&list,&i, %str(|))
13                 (obs=5) noobs;
14      run;
15   %end;
16  %mend loop;
17  %loop
```

The above macro does not have a parameter list.

```
 1  *Ex8_%DO_Nonsequential2.sas;
 2  options nocenter nodate nonumber symbolgen;
 3  %macro loop(dslist);
 4       /* Count the # of values in the string */
 5       %let xcount=%sysfunc(countw(&dslist, %STR(|)));
 6       /* Loop through the total # of data sets */
 7       %do i = 1 %to &xcount;
 8          title  "%left(%scan(&dslist,&i,%str(|)))";
 9          proc print data=%scan(&dslist,&i,%str(|))
10                     (obs=5) noobs;
11        run;
12        %end;
13  %mend loop;
14  %loop(%str(sashelp.class|sashelp.cars|sashelp.retail))
```

The above macro has a parameter list.

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Macro with a series of macro variables (common prefix with a numeric suffix)

```
 1  *Ex14_VList_HList.sas;
 2  options nodate nonumber symbolgen;
 3  %macro VList;
 4  %let ds1 = class;
 5  %let ds2 = revhub2;
 6  %let ds3 = iris;
 7  %let dscount = 3;
 8  %do j = 1 %to &dscount;
 9    title1 "%upcase(sashelp.&&ds&j)";
10    proc print data=sashelp.&&ds&j (obs=5) noobs;
11    run;
12  %end;
13  %mend VList;
14  %VList
```

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Macro with a list of values of a macro Variable (No keyword or positional parameters)

```
16  *Ex14_VList_HList.sas;
17  options nodate nonumber symbolgen;
18  %macro HList;
19    %local dslist j;
20    %let dslist = class revhub2 iris;
21    %do j =1 %to %sysfunc(countw(&dslist));
22       title1 "sashelp.%scan(&dslist,&j)";
23       proc print data= sashelp.%scan(&dslist, &j)(obs=5) noobs;
24       run;
25    %end;
26  %mend HList;
27  %HList
```

Line 20: The macro variable DSLIST contains the data set names.

Lines 21-26: The %DO loop is used to step through the list with %DO loop index (&J) serving as the word counter. The %SCAN is used to retrieve the &jth word (data set names).

Line 27: The call to macro prints observations from each of the three data sets.

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Guideline 5: Comment Macro Applications

Partial SAS Program

```
%macro archive(dsn);

   /*-----------------------------------------------------*/
   /* ARCHIVE - Archive data set, appending current date  */
   /* to build a new (unique by day) name for the copy.   */
   /*-----------------------------------------------------*/
   /* Parameter DSN: Master table to be archived          */
   /*-----------------------------------------------------*/
   /* Requires these other macros:                        */
   /*   %VALIDDSN - validates a SAS data set name         */
   /*-----------------------------------------------------*/

%* %put Activating the MPRINT and MLOGIC options.;

*options mprint mlogic;
```

21

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Conditional Processing

%IF ... **%THEN** *action*;
%ELSE *action*;

These *actions* can follow keywords %THEN and %ELSE:

- a macro language statement
- a macro variable reference
- a macro call
- any text

10

Idea Exchange

What is the difference between %IF-%THEN and IF-THEN?

	%IF-%THEN	IF-THEN
Valid in	Macro definition	DATA step
Performs	SAS code (text) processing	Data processing
Passed to	Macro processor	DATA step compiler
Purpose	Determine what SAS code to place on the input stack for tokenization, compilation, and eventual execution	Determine what DATA step statement (or statements) to execute during each execution-time iteration of the DATA step

24

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Monitoring Macro Execution

The MLOGIC system option displays macro execution messages in the SAS log.

Partial SAS Log

```
494  %macro reports;
495     %daily
496     %if &sysday=Friday %then %weekly;
497  %mend reports;
498
499  options mlogic;
500  %reports
MLOGIC(REPORTS):  Beginning execution.
MLOGIC(DAILY):  Beginning execution.
MLOGIC(DAILY):  Ending execution.
MLOGIC(REPORTS):  %IF condition &sysday=Friday is TRUE
MLOGIC(WEEKLY):  Beginning execution.
MLOGIC(WEEKLY):  Ending execution.
MLOGIC(REPORTS):  Ending execution.
```

OPTIONS MLOGIC;

The default setting is NOMLOGIC.

m105d01a

11

Processing Complete Steps

Method 2 Create a single macro with %DO and %END statements to generate text that contains semicolons.

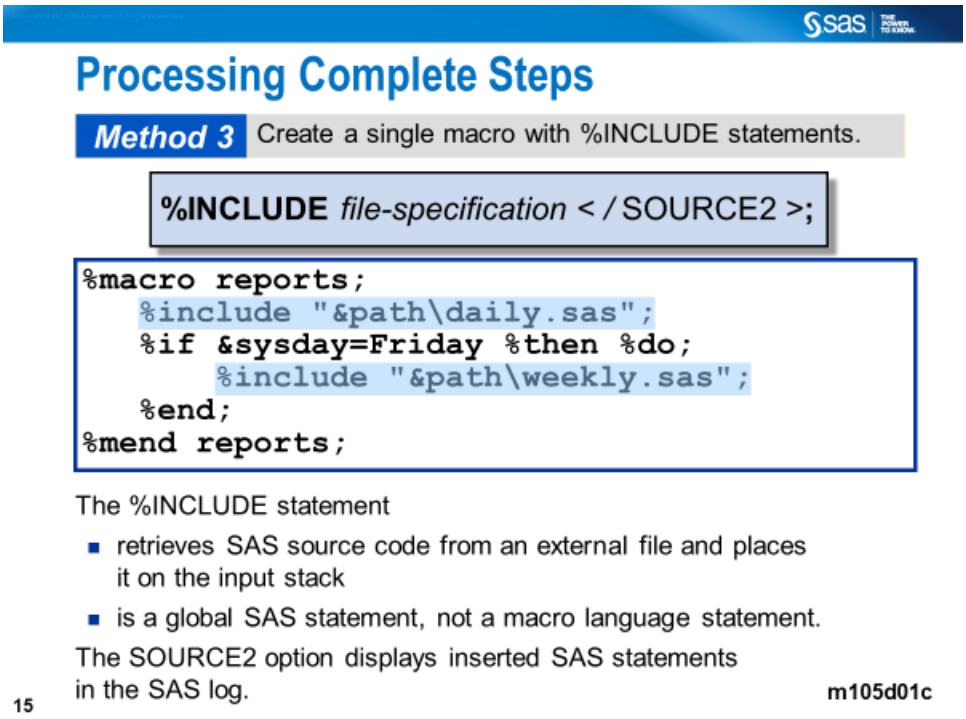
```
%macro reports;
   proc print data=orion.order_fact;
      where order_date="&sysdate9"d;
      var product_id total_retail_price;
      title "Daily sales: &sysdate9";
   run;
 %if &sysday=Friday %then %do;
   proc means data=orion.order_fact n sum mean;
      where orc
            "&sy                          9"d;
      var quant
      title "We
   run;
 %end;
%mend reports;
```

%IF *expression* **%THEN %DO;**
statement; *statement*;...
%END;
%ELSE %DO;
statement; *statement*;...
%END;

m105d01b

14

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts



Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%INCLUDE

- retrieves SAS source code from an external file and places it on the input stack
- is a global SAS statement, not a macro language statement.

%INCLUDE statement is used below (line 6) to identify a physical file name**.**

```
1  *Ex17_percent_include.sas;
2  filename jobs "c:\sascourse\Week10";
3  %include jobs(Part1)/source2;
4  %include jobs(Part2)/source2;
5  %include jobs(Part3)/source2;
```

Line 2:

Lines 3-5: %INCLUDE brings in SAS statements stored in external files (part1, part2, and part3). The SOURCE2 option displays inserted SAS statements in the SAS log. Note that the % include is not a macro statement.

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

```
1  *Ex18_Macro_Include.sas;
2  options symbolgen;
3  %include 'c:\SASCourse\Week10\Macro_List_Files.txt';
4  ods Excel file="C:\SASCourse\Week10\List_SAS_Files.xlsx" ;
5  %macro mexcel (start, stop);
6  %DO num = &START %TO &STOP;
7        ods excel options(sheet_name="Week&num");
8          data _null_;
9            file print;
10          %drive(C:\SASCourse\Week&num,sas)
11          run;
12 %end;
13 ods Excel close;
14 %mend mexcel;
15 /* Create an Excel Workbook with 2 sheets, listing
16    .SAS files from Week9 and Week10 folders, respectively*/
17 %mexcel (9,10)
```

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Saving Macros Using the Stored Compiled Macro Facility

Macros can be compiled and then stored temporarily or permanently for later use. When you compile a macro, it is usually stored in the temporary WORK.SASMACR catalog. However, you can store a macro in compiled form in a SAS macro catalog also named SASMACR, but in a permanent library.

```
 1  *Ex19_Compiled_Macro.sas;
 2  *** This is a revised version of the macro obtained from
 3      http://support.sas.com/kb/45/805.html
 4      [Sample 45805: List all files within a directory includir
 5       sub-directories]
 6      Revisions: Macro calls are wrapped in data steps;
 7
 8  /* The following two lines and the options to the macro
 9     are added by Pradip Muhuri
10   */
11  Libname MyLib 'C:\SASCourse\Compiled_Macros';
12  options mstored sasmstore=MyLib;
13  %macro drive(dir,ext) /store source
14           des='Listing SAS Files';
15    /* Defime macro variables */
16    %local filrf rc did memcnt name i;
```

…… 30 lines have been skipped here.

```
46  %mend drive;
47  %sysmstoreclear;    /* Added by Pradip Muhuri*/
```

Line 12: In the OPTIONS statement, the system option MSTORED turns on the use of stored compiled macros. The system option SASMSTORE= specifies the libref that contains a SAS catalog named SASMACR.

Lines 13-46: This is the macro definition. Note the STORE option (required), the SOURCE option, and the DES option on the %MACRO statement. The NEW.SASMACR catalog is opened when the entire macro written to it.

Line 47: The %SYSMSTORECLEAR statement closes the SASMACR catalog, preventing the libref from being reassigned.

Displaying the Information from the Catalog

The CATALOG procedure can be used to display the relevant information, including the description specified when the macro was compiled and stored. Below is an example.

```
1  *Ex20_Macro_Catalog.sas;
2  Libname Mylib 'C:\SASCourse\Compiled_Macros';
3  proc catalog catalog=Mylib.sasmacr;
4  contents;
5  run;
```

Contents of Catalog MYLIB.SASMACR

#	Name	Type	Create Date	Modified Date	Description
1	DRIVE	MACRO	10/31/2018 03:48:44	10/31/2018 03:48:44	Listing SAS Files

```
1  *Ex21_Access_Macro.sas;
2  libname Mylib 'C:\SASCourse\Compiled_Macros';
3  options mstored sasmstore=Mylib;
4  ods Excel file="C:\SASCourse\Week10\List_SAS_Files_x.xlsx";
5  %macro mexcel (start, stop);
6  %DO num = &START %TO &STOP;
7          ods excel options(sheet_name="Week&num");
8          data _null_;
9            file print;
10          %drive(C:\SASCourse\Week&num,sas)
11          run;
12 %end;
13 ods Excel close;
14 %mend mexcel;
15 /* Create an Excel Workbook with two sheets, listing
16   .SAS files from Week7 and Week8 folders, respectively*/
17 %mexcel (6,7)
```

Line 10: This is a call to macro which was earlier compiled and stored permanently.

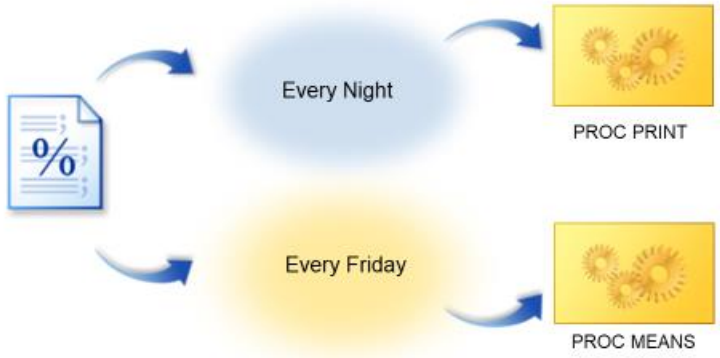
Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Running R code embedded in PROC IML to generate a list of file names from a folder

```
1 *Ex24_Week10_List_of_Files.sas;
2 title; title1; title2;
3 PROC IML;
4 SUBMIT / R;
5 setwd ("C:/SASCourse/Week10")
6 list.files(pattern="SAS*", full.names = TRUE, ignore.case = TRUE)
7 ENDSUBMIT;
8 QUIT;
```

Business Scenario

A daily sales report is generated every night. Every Friday, a weekly report is generated. Determine the best method to automate these reports.



4

Solutions

Three methods:

Method 1	Create multiple macros, including a driver macro.
Method 2	Create a single macro with %DO and %END statements.
Method 3	Create a single macro with %INCLUDE statements.

6

Processing Complete Steps

Method 1 Create separate macros for the **Daily** and **Weekly** programs.

```
%macro daily;
   proc print data=orion.order_fact;
      where order_date="&sysdate9"d;
      var product_id total_retail_price;
      title "Daily sales: &sysdate9";
   run;
%mend daily;

%macro weekly;
   proc means data=orion.order_fact n sum mean;
      where order_date between
          "&sysdate9"d-6 and "&sysdate9"d;
      var quantity total_retail_price;
      title "Weekly sales: &sysdate9";
   run;
%mend weekly;
```

m105d01a

7

continued...

Processing Complete Steps

Method 1 Create a driver macro that always calls the **Daily** macro and conditionally calls the **Weekly** macro.

```
%macro reports;
   %daily
   %if &sysday=Friday %then %weekly;
%mend reports;
```

%IF *expression* **%THEN** *action*;
%ELSE *action*;

- Character constants are
  - not quoted
  - case sensitive.
- The %ELSE statement is optional.
- %IF-%THEN and %ELSE statements can be used inside a macro definition only.

m105d01a

8

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Adding variable names dynamically to the PROC FREQ step using the %IF-%then in a macro

```
1  *Ex17_percent_If.sas;
2  %macro run_freq(row_var);
3     title 'SASHELP Data Set - Selected Frequency Tables';
4     proc freq data=sashelp.heart;
5           tables
6     %if &row_var ne  %then &row_var *;
7        smoking_status;
8     run;
9  %mend run_freq;
10
11 %run_freq()
12 %run_freq(sex)
13 %run_freq(Weight_Status)
```

Line 6: Here, we conditionally insert text into the middle of a statement to generate one-way or two-way frequency tables.

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Creating Data-Dependent Values of a Macro Variable for Table Look-Up

```
1  *Ex9_Macro_In_Operator.sas;
2  proc sql ;
3         select quote(strip(Name))
4         INTO  :Starts_withC separated by ','
5       from sashelp.demographics
6         where Name LIKE 'C%';
7  quit;
8  %PUT &Starts_withC;
```

(Partial SAS Log)

```
"CANADA","COSTA RICA","CUBA","CHILE","COLOMBIA","CZECH
REPUBLIC","CROATIA","CAMEROON","CAPE
VERDE","CENTRAL AFRICAN
REP.","CHAD","COMOROS","CONGO","CAMBODIA","CHINA","CYPRUS","COOK
ISLANDS","CORAL SEA ISLANDS"
```

Using a Macro Variable Reference to Select a List of Character Values in the PROC PRINT Step *without a Macro*

```
10  *Ex9_Macro_In_Operator.sas;
11   proc print data=sashelp.demographics;
12    var Name pop;
13    where Name in  (&Starts_withC);
14    run;
```

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Using a Macro Variable Reference to Select a List of Character Values in the PROC PRINT Step *with a Macro*

```
16  *Ex9_Macro_In_Operator.sas;
17  %macro cn / minoperator mindelimiter=',';
18    proc print data=sashelp.demographics;
19    var Name pop;
20    where Name in  (&Starts_withC);
21    run;
22  %mend cn;
23  %cn
```

The compiled macro is stored in a SAS catalog (WORK.SASMACR).

LINE 15: The MINOPERATOR specifies that the macro processor recognizes and evaluates the mnemonic **IN** and the special character # as logical operators when evaluating arithmetic or logical expressions during the execution of the macro. The MINDELIMETER=',' specifies the character to be used as the delimiter for the macro In operator.

Generating Data-Dependent Code

Example: Create a separate data set for each value of a selected variable in a selected data set.

```
%split(data=orion.customer, var=country)
```

Partial SAS Log

```
MPRINT(SPLIT):   data AU CA DE IL TR US ZA ;
MPRINT(SPLIT):   set orion.customer;
MPRINT(SPLIT):   select(country);
MPRINT(SPLIT):   when("AU") output AU;
MPRINT(SPLIT):   when("CA") output CA;
MPRINT(SPLIT):   when("DE") output DE;
MPRINT(SPLIT):   when("IL") output IL;
MPRINT(SPLIT):   when("TR") output TR;
MPRINT(SPLIT):   when("US") output US;
MPRINT(SPLIT):   when("ZA") output ZA;
MPRINT(SPLIT):   otherwise;
MPRINT(SPLIT):   end;
MPRINT(SPLIT):   run;
```

55

Generating Data-Dependent Code

Step 1 Store unique data values into macro variables.

```
%macro split (data=, var=);
  proc sort data=&data(keep=&var) out=values nodupkey;
     by &var;
  run;
  data _null_;
     set values end=last;
     call symputx('val'||left(_n_),&var);
     if last then call symputx('count',_n_);
  run;
  %put _local_;
%mend split;

%split(data=orion.customer, var=country)
```

56

m105d09a

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Generating Data-Dependent Code

Step 2 Use loops to generate the DATA step.

```
%macro split (data=, var=);
  proc sort data=&data(keep=&var) out=values nodupkey;
     by &var;
  run;
  data _null_;
     set values end=last;
     call symputx('val'||left(_n_),&var);
     if last then call symputx('count',_n_);
  run;
  data
     %do i=1 %to &count;
        &&val&i
     %end;
  ;
     set &data;
        select(&var);
        %do i=1 %to &count;
           when("&&val&i") output &&val&i;
        %end;
        otherwise;
        end;
  run;
%mend split;
```

m105d09b

58

Generating Data-Dependent Code

Partial SAS Log

```
MPRINT(SPLIT):   data AU CA DE IL TR US ZA ;
MPRINT(SPLIT):   set orion.customer;
MPRINT(SPLIT):   select(country);
MPRINT(SPLIT):   when("AU") output AU;
MPRINT(SPLIT):   when("CA") output CA;
MPRINT(SPLIT):   when("DE") output DE;
MPRINT(SPLIT):   when("IL") output IL;
MPRINT(SPLIT):   when("TR") output TR;
MPRINT(SPLIT):   when("US") output US;
MPRINT(SPLIT):   when("ZA") output ZA;
MPRINT(SPLIT):   otherwise;
MPRINT(SPLIT):   end;
MPRINT(SPLIT):   run;

NOTE: There were 77 observations read from the data set ORION.CUSTOMER.
NOTE: The data set WORK.AU has 8 observations and 12 variables.
NOTE: The data set WORK.CA has 15 observations and 12 variables.
NOTE: The data set WORK.DE has 10 observations and 12 variables.
NOTE: The data set WORK.IL has 5 observations and 12 variables.
NOTE: The data set WORK.TR has 7 observations and 12 variables.
NOTE: The data set WORK.US has 28 observations and 12 variables.
NOTE: The data set WORK.ZA has 4 observations and 12 variables.
NOTE: DATA statement used (Total process time):
      real time           0.10 seconds
      cpu time            0.10 seconds
```

m105d09b

59

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Generating Data-Dependent Code

SAS metadata is available in the **sashelp.vstabvw** dynamic view.

```
proc print data=sashelp.vstabvw;
   where libname="ORION";
   title "SASHELP.VSTABVW";
run;
```

Partial PROC PRINT Output

SASHELP.VSTABVW

Obs	libname	memname	memtype
336	ORION	CITY	DATA
337	ORION	CONTINENT	DATA
338	ORION	COUNTRY	DATA
339	ORION	COUNTY	DATA
340	ORION	CUSTOMER	DATA
341	ORION	CUSTOMER_DIM	DATA

63 m105d10

Generating Data-Dependent Code

Partial SAS Log

```
PRINTLIB DSN1 AGESMOD
PRINTLIB DSN10 COUPONS
PRINTLIB DSN11 CUSTOMER
PRINTLIB DSN12 CUSTOMERDIM
PRINTLIB DSN13 CUSTOMERDIMMORE
PRINTLIB DSN14 CUSTOMERTYPE
PRINTLIB DSN15 CUSTOMER_DIM
PRINTLIB DSN16 CUSTOMER_TYPE
PRINTLIB DSN17 DAILY_SALES
PRINTLIB DSN18 DISCOUNT
PRINTLIB DSN19 EMPLOYEEADDRESSES
PRINTLIB DSN2 BIZLIST
PRINTLIB DSN20 EMPLOYEEDONATIONS
PRINTLIB DSN21 EMPLOYEEORGANIZATION
PRINTLIB DSN22 EMPLOYEEPAYROLL
PRINTLIB DSN23 EMPLOYEEPHONES
PRINTLIB DSN24 EMPLOYEE_PAYROLL
PRINTLIB DSN25 EUROPECUSTOMERS
```

65 m105d11a

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Generating Data-Dependent Code

Step 2 Use a macro loop to print every data set in the library.

```
%macro printlib(lib=WORK,obs=5);
  %let lib=%upcase(&lib);
  data _null_;
    set sashelp.vstabvw end=final;
    where libname="&lib";
    call symputx('dsn'||left(_n_),memname);
    if final then call symputx('totaldsn',_n_);
  run;
  %do i=1 %to &totaldsn;
    proc print data=&lib..&&dsn&i(obs=&obs);
       title "&lib..&&dsn&i Data Set";
    run;
  %end;
%mend printlib;
%printlib(lib=orion)
```

m105d11b

66

Generating Data-Dependent Code

Partial SAS Log

```
MPRINT(PRINTLIB):   proc print data=ORION.CUSTOMER(obs=5);
MPRINT(PRINTLIB):   title "ORION.CUSTOMER Data Set";
MPRINT(PRINTLIB):   run;
NOTE: There were 5 observations read from the data set ORION.CUSTOMER.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

MPRINT(PRINTLIB):   proc print data=ORION.CUSTOMER_DIM(obs=5);
MPRINT(PRINTLIB):   title "ORION.CUSTOMER_DIM Data Set";
MPRINT(PRINTLIB):   run;
NOTE: There were 5 observations read from the data set ORION.CUSTOMER_DIM.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

MPRINT(PRINTLIB):   proc print data=ORION.CUSTOMER_TYPE(obs=5);
MPRINT(PRINTLIB):   title "ORION.CUSTOMER_TYPE Data Set";
MPRINT(PRINTLIB):   run;
NOTE: There were 5 observations read from the data set ORION.CUSTOMER_TYPE.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

67

m105d11b

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts