SAS/MACROS

Macro is a stored text that can be set anywhere in SAS Program and expanded.

(Or)

Macro is a stored code that can be used repeatedly with the macro name.

Uses of SAS/Macros

-> Reduce the amount of programming needed for repeated tasks

Example:-

When we have to extract 10 Excel files from external location to SAS We need to write 10 times Proc import statements but using Macros we can write in single step.

```
%macro import(file,dname);
  Proc import datafile="&file" out=&dname
  Dbms=excel replace;
  Getnames=yes;
  Run;
%mend;
  %import(E:\sas\macro\data1.xls,ds1);
  %import(E:\sas\macro\data2.xls,ds2);
  %import(E:\sas\macro\data3.xls,ds3);
  %import(E:\sas\macro\data4.xls,ds4);
  %import(E:\sas\macro\data5.xls,ds5);
  %import(E:\sas\macro\data6.xls,ds6);
  %import(E:\sas\macro\data7.xls,ds7);
  %import(E:\sas\macro\data8.xls,ds8);
  %import(E:\sas\macro\data9.xls,ds9);
  %import(E:\sas\macro\data10.xls,ds10);
```

-> Minimize on typing and execution errors
-> Making the variables global
-> Simplifying the code
-> In Macros we have only one data type that is character.

Generally Macro code takes longer time to write and debug than standard SAS code, But if you find yourself writing similar code over and over again then macros may make your job easier.

THE MACRO PROCESSOR

The most important concept to keep in mind whenever you write macro code is that You are writing a program that writes a program.

Here is how it works.

When you submit a standard SAS program, SAS compiles and then immediately executes it. But when you write macro code, there is an extra step. Before SAS can compile and execute your program, SAS must pass your macro statements to the macro processor which then "resolves" your macros generating standard SAS code. Because you are writing a program that writes a program, this is sometimes called meta-programming.

TURNING ON MACRO PROCESSOR

Before you can use macros you must have check Macro system is turned on or not, this option is turned on by default, may be its turned off by some times, to find out whether it's on or off execute below program lines

```
Proc options option=macro;
Run;
```

See the log

If you see the option MACRO then macro processor is turned on
If you see the option NOMACRO then macro processor is turned off

```
3    Proc options option=macro;
4    Run;

    SAS (r) Proprietary Software Release 9.2 TS2M2

 MACRO             Allow use of SAS macro facility
 NOTE: PROCEDURE OPTIONS used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

MACROS Vs MACRO VARIABLES

SAS Macro program consists two basic building blocks (Macro triggers):

1) **Macros**
2) **Macro variables.**

Macros start with a percent sign (%) and
Macro variables start with an ampersand (&)

Macro variables are 2 types

i) System defined (Automatic macro variables)
ii) User defined Macro variables

User defined macro variables are again 2 types **i) Global ii) Local**

SCOPE OF THE MACRO VARIABLES

Macro variables can have two kind of scope

1) **Global**
2) **Local**

A macro variable is Local if it is defined inside a macro
A macro variable is Global if it is defined open code or outside a macro

We can use a Local macro variable only inside its own macro
But we can use Global macro variable anywhere in sas program

YOU MAY QUOTE ME ON THAT

Another caveat to keep in mind is that the macro processor doesn't check for macros inside single quotes. To get around this, simply use double quotes for titles or other quoted strings that contain macro code.

AUTOMATIC MACRO VARIABLES

Automatic macro variables are created by the macro processor and they supply a variety of information.

The three-letter prefix SYS is reserved for use by SAS for automatic macro variables. So we can't create macro variables on the name start of SYS.

```
%put _automatic_;
```

Run above program to see all the automatic macro variable information in sas log.

Examples (How can you use automatic macro variables):-

```
Data ds;
Set example;
Where date="&sysdate9.";
Run;
```

```
PROC SQL;
CREATE TABLE USCLMAMI as
SELECT CARRIER,
          PAID_DATE,
          PAYMENT_AMT,
          SVC_BEG_DATE,
          CH_NETWORK,
          CORP_NAME
FROM DSNW.DWS_US_AMISYS_CLM
WHERE CARRIER in( 'SM','CH')
AND PAID_DATE ="&sysdate.";
Quit;
```

```
Options nodate nonumber;
Proc print data=ds;
title1 "information on &sysday";
Footnote "information on &sysdate";
footnote2 "information on &systime";
footnote3 "information on &sysdate9";
```

```
footnote4 "information on &sysver";
Run;
```

Automatic variables are

Read/Write

SYSBUFFER unmatched text from %INPUT
SYSCC the current condition code that SAS returns to your operating environment (the operating environment condition code)
SYSCMD last unrecognized command from the command line of a macro window
SYSDEVIC name of current graphics device
SYSDMG return code that reflects an action taken on a damaged data set
SYSDSN name of most recent SAS data set in two fields
SYSFILRC return code set by the FILENAME statement
SYSLAST name of most recent SAS data set in one field
SYSLCKRC return code set by the LOCK statement
SYSLIBRC return code set by the LIBNAME statement
SYSMSG message for display in macro window
SYSPARM value specified with the SYSPARM= system option
SYSPBUFF text of macro parameter values
SYSRC various system-related return codes

Read-Only

SYSCHARWIDTH the character width value
SYSDATE the character value representing the date a SAS job or session began executing (two-digit year)
SYSDATE9 the character value representing the date a SAS job or session began executing (four-digit year)
SYSDAY day of week SAS job or session began executing
SYSENV foreground or background indicator
SYSERR return code set by SAS procedures and the DATA step
SYSINDEX number of macros that have begun execution during this session
SYSINFO return code information
SYSJOBID name of current batch job or user id (varies by host environment)
SYSMENV current macro execution environment
SYSPROCESSID the process id of the current SAS process
SYSPROCESSNAME the process name of the current SAS process
SYSSCP the abbreviation of an operating system
SYSSCPL the name of an operating system
SYSSITE the number assigned to your site
SYSSTARTID the id generated from the last STARTSAS statement
SYSSTARTNAME the process name generated from the last STARTSAS statement
SYSTIME the character value of the time a SAS job or session began executing
SYSUSERID the userid or login of the current SAS process
SYSVER release or version number of SAS software executing
SYSVLONG release number and maintenance level of SAS software
Automatic variables are global except SYSPBUFF, which is local.
USE %PUT _AUTOMATIC_ to see all the automatic variables in sas log.

```
%put _automatic_;
```

Run above program to see all the automatic macro variable information in sas log
And look on the values for automatic macro variables.

AUTOMATIC AFDSID 0
AUTOMATIC AFDSNAME
AUTOMATIC AFLIB
AUTOMATIC AFSTR1
AUTOMATIC AFSTR2
AUTOMATIC FSPBDV
AUTOMATIC SYSBUFFR
AUTOMATIC SYSCC 0
AUTOMATIC SYSCHARWIDTH 1
AUTOMATIC SYSCMD
AUTOMATIC SYSDATE 27NOV10
AUTOMATIC SYSDATE9 27NOV2010
AUTOMATIC SYSDAY Saturday
AUTOMATIC SYSDEVIC
AUTOMATIC SYSDMG 0
AUTOMATIC SYSDSN _NULL_
AUTOMATIC SYSENCODING wlatin1
AUTOMATIC SYSENDIAN LITTLE
AUTOMATIC SYSENV FORE
AUTOMATIC SYSERR 0
AUTOMATIC SYSERRORTEXT
AUTOMATIC SYSFILRC 0
AUTOMATIC SYSHOSTNAME computer
AUTOMATIC SYSINDEX 0
AUTOMATIC SYSINFO 0
AUTOMATIC SYSJOBID 4144
AUTOMATIC SYSLAST _NULL_
AUTOMATIC SYSLCKRC 0
AUTOMATIC SYSLIBRC 0
AUTOMATIC SYSLOGAPPLNAME
AUTOMATIC SYSMACRONAME
AUTOMATIC SYSMAXLONG 2147483647
AUTOMATIC SYSMENV S
AUTOMATIC SYSMSG
AUTOMATIC SYSNCPU 2
AUTOMATIC SYSODSPATH SASUSER.TEMPLAT(UPDATE) SASHELP.TMPLMST(READ)
AUTOMATIC SYSPARM
AUTOMATIC SYSPBUFF
AUTOMATIC SYSPROCESSID 41D7F0343995F3B64018000000000000
AUTOMATIC SYSPROCESSNAME DMS Process
AUTOMATIC SYSPROCNAME
AUTOMATIC SYSRC 0
AUTOMATIC SYSSCP WIN
AUTOMATIC SYSSCPL XP_PRO
AUTOMATIC SYSSITE 0011600745
AUTOMATIC SYSSIZEOFLONG 4
AUTOMATIC SYSSIZEOFUNICODE 2

AUTOMATIC SYSSTARTID
AUTOMATIC SYSSTARTNAME
AUTOMATIC SYSTCPIPHOSTNAME computer
AUTOMATIC SYSTIME 10:11
AUTOMATIC SYSUSERID sasadm
AUTOMATIC SYSVER 9.2
AUTOMATIC SYSVLONG 9.02.02M2P090109
AUTOMATIC SYSVLONG4 9.02.02M2P09012009
AUTOMATIC SYSWARNINGTEXT

MACRO STATEMENTS

%Macro

Begins a macro definition or %macro creates a macro
Syntax: - %Macro Macro_Name(<Parameter1, Parameter2,..., ParameterN>)

%Mend

Ends a macro definition
Syntax: - %Mend;
%Macro_Name(Parameter1Value, Parameter2Value,..., ParameterNValue>)

Note:-If you specify a macro name with %Mend Statement that name should be match with %Macro Statement

It means both names should give same

Macro-name is a name you make up for your macro. The name must follow standard SAS naming conventions

(Start with a letter or underscore; contain only letters, numerals or underscores, and can be up to 32 characters in length).

The general form of Macro is (Creating Modular code with Macros)

Between %Macro and %Mend you can put any statements you want. The general form of a macro is

%Macro Macro_name;
Macro-Text
%Mend Macro_name;

The Modular approach to programming is good to avoiding Macro errors
First write standard SAS program and when it's bug free convert it into Macro program
Adding one future at a time.

Invoking a macro

After you have defined a macro you can invoke it by adding the percent sign in front of Macro. Like below

%Macro_name
A semicolon is not required when invoking a macro, Generally if you add it doesn't harm.

Example:-

```
%Macro Mac1;
Data ds;
Set sashelp.class;
Run;
Proc print data=ds;
Run;
%Mend;
%Mac1  -/-----  Invoking a Macro
```

%Put

Can used to write text (or) Macro variable information to SAS Log.

Syntax: - %put text

%put Macro_variable

Example:-

```
%put &Name.;
%put &Age.;
%put &Date.;
```

Prints all system defined macro variables in log.

```
%PUT _AUTOMATIC_;
```

Prints all global macro variables in log.

```
%PUT _GLOBAL_;
```

Prints all global and system defined macro variables in log.

```
%PUT _ALL_;
```

```
%PUT _USER_;
```

Prints macro variables that are created by the user in each environment.

%Include

Includes and executes SAS statements and data lines.

Syntax: - %Include source;

Example:-
```
%include "D:\prg.sas";
%include "D:\prg.txt";
```


STANSYS SOFTWARE SOLUTIONS
#7-1-621/113(67/3RT), Beside: Nagarjuna High School, Near: S.R.Nagar Community Hall,
S.R.Nagar, Hyd-38|Ph:9542195422|8143408688|www.stansys.co.in|stansys.sas@gmail.com

WAYS TO CREATE USER DEFINED MACRO VARIABLES

%Let

%Let Creates a Macro variable and assigns a value.

Syntax: - %LET Macro variable=value;

Examples:-

```
%Let Name= Akshara;
%Let Age= 20;
%Let Date=%sysfunc(today(),date9.);
```

See the values of above macro variables in SAS Log. Using

```
%put &Name.;
%put &Age.;
%put &Date.;
```

```
Data models;
Infile datalines;
Input Model $ 1-12 Class $ Price Frame $ 28-38;
Datalines;
Black Bora    Track      796 Aluminum
Delta Breeze Road       399 CroMoly
Jet Stream    Track     1130 CroMoly
Mistral       Road      1995 Carbon Comp
Nor'easter    Mountain  899 Aluminum
Santa Ana     Mountain  459 Aluminum
Scirocco      Mountain 2256 Titanium
Trade Wind    Road      759 Aluminum
;
Run;
```

```
%Let bikeclass = Mountain;
```

```
/* Use a macro variable to subset;*/
Proc print data = models noobs;
  Where Class = "&bikeclass";
  Format Price dollar6.;
  Title "Current Models of &bikeclass Bicycles";
Run;
```

Note that the macro variable **&bikeclass** will be global because it is created outside of a macro in open code. When you submit the program with a value of "Mountain", then the macro processor will resolve the macro variable and create below standard SAS code.

```
Proc print data = models noobs;
  Where Class = "Mountain";
  Format Price dollar6.;
  Title "Current Models of Mountain Bicycles";
Run;
```

Using let we can create both global & local macro variables. If it is within a macro it is Local or outside a macro or open code it is global.

%Global

Creates a Macro variable and assigns null value

If any macro variable is global that you call anywhere in the SAS system, Global macro variable is available during the execution of an entire SAS session

Syntax:- %Global Macrovariable;

```
%Global a;
%Put &a.;
```

```
%MACRO DATES;
%GLOBAL TODAY1 YESTERDAY THIS_MON LAST_MON DAY T_MON L_MON;
%LET TODAY1=%SYSFUNC(TODAY(),DATE9.);
%LET YESTERDAY=%SYSFUNC(INTNX(DAYS,"&TODAY1."D,-1),DATE9.);
%LET THIS_MON=%SYSFUNC(INTNX(MON,"&TODAY1."D,0,S),DATE9.);
%LET LAST_MON=%SYSFUNC(INTNX(MON,"&TODAY1."D,-1,S),DATE9.);
%LET DAY=%SYSFUNC(SUBSTR(&YESTERDAY.D,1,5));
%LET T_MON=%SYSFUNC(SUBSTR(&TODAY1.D,3,3));
%LET L_MON=%SYSFUNC(SUBSTR(&LAST_MON.D,3,3));
%LET BDATE=%SYSFUNC(INTNX(MONTH,"&TODAY1."D,0,B),DATE9.);
%LET EDATE=%SYSFUNC(INTNX(MONTH,"&TODAY1."D,0,E),DATE9.);
%MEND;
%DATES;
%PUT &TODAY1.;
%PUT &YESTERDAY.;
%PUT &THIS_MON.;
%PUT &LAST_MON.;
%PUT &DAY.;
%PUT &T_MON.;
%PUT &L_MON.;
%PUT &BDATE.;
%PUT &EDATE.;
```

```
Proc sql;
Create table USCAR1A_temp as
Select CARRIER,
       MAX(ANNIV) as ANNIV
From USMBRAMI_TEMP
WHERE END_DATE GE "&EDATE"D
And KP EQ 'Y'
And CARRIER NOT IN('RZ','KS','OK','MO','TP')
Group by CARRIER;
Quit;
```

See how I am using Global Macro variable **EDATE** in SAS SQL program

See how I am using above Global Macro variables in Proc Report

```
OPTIONS MISSING=0;
Proc report data =My_SAS.SBB headline headskip missing nowindows SPLIT='*' ;
Column    TM_CODE  TM_NAME    STATE_HEAD_CODE  SH_NAME  AM_CODE  AM_NAME
aa,(INWARDDAY INWARDMON INWARDLMON inwarddiff)
ab,(NCADAY NCAMON NCALMON ncadiff);
define TM_CODE/group 'TM Code' noprint ;
define TM_NAME/group  'TM Name' ;
define STATE_HEAD_CODE/group  'SH Code' noprint ;
define SH_NAME/group  'SH Name' ;
define AM_CODE/group 'AM Code' ;
define AM_NAME/group 'AM Name' ;
define aa/across 'Inwards';
define ab/across 'NCA';
define INWARDDAY/analysis "&DAY";
define NCADAY/analysis "&DAY";
define INWARDMON/analysis "&T_MON";
define INWARDLMON/analysis "&L_MON";
define inwarddiff/analysis "Difference*in MTD" ;
define NCAMON/analysis "&T_MON";
define NCALMON/analysis "&L_MON";
define ncadiff/analysis "Difference*in MTD";
break after TM_NAME/summarize skip style={foreground=rose font_weight=bold};
break after SH_NAME/summarize style={foreground=green font_weight=bold};
RBREAK AFTER/SKIP SUMMARIZE style={foreground=VIBG font_weight=bold};
compute after TM_NAME;
TM_NAME=trim(TM_NAME)||' Total'; SH_NAME='';
endcomp;
compute after SH_NAME;
SH_NAME=trim(SH_NAME)||' Total';
endcomp;
compute after;
TM_NAME='Grand Total';
endcomp;
title "Sub-Broker Active AM Report As on &YESTERDAY";
footnote1;
footnote2;
footnote3;
footnote4 "Note: This is a system generated mail. Please do not respond to this mail.";
run;
ODS HTML CLOSE;
```

See how I am using Global Macro variables **DAY**,**T_MON**, **L_MON** in Proc report

HTML - SUB-BROKER ACTIVE AM.html

%Local

Creates a Macro variable and assigns null value

Syntax:- %Local Macrovariable;

It is useful to create a Macro variable and assigns null value that is available only during the execution of particular macro.

IT SHOULD BE WRITTEN WITH IN A MACRO, SHOULD NOT WRITE AS OPEN CODE.

```
%Local b;
%Put &b.;
```

```
%MACRO mac2;
%Local Name Age Date;
%Let Name=Akshara;
%Let Age=20;
%let Date=%sysfunc(today(),date9.);
%MEND;
%mac2
%put &Name. ;
%put &Age. ;
%put &Date. ;
```

Remember below rules while using Global & Local variables.

-> You can't use %Global statement to convert local macro variables into global macro variables.

-> While creating global macro variables inside the %macro and %mend definition you must specify the %global statement before you use the macro variable.

Example:-

```
%MACRO mac3;
%Global Name Amount Date;
%Let Name=Akshara;
%Let Age =20;
%let Date=%sysfunc(today(),date9.);
%MEND;
%mac3
%put &Name. ;
%put &Age. ;
%put &Date. ;
```

-> The %global statement creates macro variables with null values, you still have initialized them using the %let statement or call symput function

-> If you define both %global and %local with same name the macro processor uses the value of local variable during the execution of macro, means contains local variable.

Example:-

```
%Let x=10;
%Macro mac1;
```

```
%let x=20;
%Put &x;
%Mend;
%mac1
%Put &x;
```

Observe below programs and see the result of programs.

1)
```
%Let x=10;
   %Macro mac1;
   %let x=20;
   %Mend;
   %Put &x;
```

2)
```
%Let x=10;
   %Macro mac1;
   %let x=20;
   %Mend;
   %MAC1;
   %Put &x;
```

3)
```
%Let x=10;
   %Macro mac1;
   %local x;
   %let x=20;
   %Mend;
   %MAC1;
   %Put &x;
```


Call Symput

Creates Macro variable in data step.

Gets information from dataset to Macro variable.

Syntax:- Call symput('Macrovariable', Value);

Examples:-

```
Data ds;
Call symput('Amount',10000);
Run;
%Put &Amount;
```

```
Data ds;
Set sashelp.class;
Call symput('Stdname', Name);
Run;
%Put &Stdname.;
```

In above program Stdname macro variable gets last value of Name variable, but if you want pick some or all the values use below program

```
Data ds;
Set sashelp.class;
Call symput('Stdname' ||compress(_n_), Name);
Run;
%Put &Stdname1.;
%Put &Stdname2.;
%Put &Stdname3.;
%Put &Stdname4.;
%Put &Stdname5.;

Data ds;
Set sashelp.class;
If Age >= 18 then call symput("status", "Adult");
Else call symput("status", "Minor");
Run;
%Put &status.;

Proc sql;
Connect to oracle as krishna
(User=misrep  Password="{sas001}QzBycDByX3Qz" Path=srmprod);
Create table dates as
Select * from connection to krishna
(Select
to_char(to_date(SYSDATE-1),'dd-Month-yyyy') as last_1 ,
to_char(add_months(to_date(SYSDATE-1,'dd-Mon-yyyy'),-1),'Month') as last_2,
to_char(add_months(to_date(SYSDATE-1,'dd-Mon-yyyy'),-2),'Month') as last_3,
to_char(add_months(to_date(SYSDATE-1,'dd-Mon-yyyy'),-2),'Month') as last_4,
to_char(add_months(to_date(SYSDATE-1),-1),'dd-Month-yyyy') as last_5,
to_char(add_months(to_date(SYSDATE-1,'dd-Mon-yyyy'),-2),'Month') as last_6,
to_char(add_months(to_date(SYSDATE-1,'dd-Mon-yyyy'),-3),'Month') as last_7,
to_char(add_months(to_date(SYSDATE-1,'dd-Mon-yyyy'),-3),'Month') as last_8,
to_char(add_months (to_char (trunc(SYSDATE-1, 'MM')),-1),'Month') as last_mon,
to_char (trunc(SYSDATE-1),'DD-Mon-YYYY') as this_day ,
to_char (trunc(SYSDATE-1),'Month') as this_mon from dual
);
Disconnect from krishna;
Quit;

Data _null_;
Set dates;
ka=last_1;
kb=last_2;
kc=last_3;
kd=last_4;
ke=last_5;
kf=last_6;
kg=last_7;
kh=last_8;
ki=last_mon;
kj=this_day ;
```

```
kk=this_mon ;
call symput('last_1',ka);
call symput('last_2',kb);
call symput('last_3',kc);
call symput('last_4',kd);
call symput('last_5',ke);
call symput('last_6',kf);
call symput('last_7',kg);
call symput('last_8',kh);
call symput('last_mon',ki);
call symput('this_day',kj);
call symput('this_mon',kk);
Put ka kb kc kd ke kf kg kh ki kj kk;
Run;
```

Into in SQL

Creates a Macro variable in Proc SQL (SAS SQL Language)

```
Data ds1;
infile datalines;
input id name$ age sex$ sal;
datalines;
001 abc 23 F 23000
002 dfe 25 M 24500
002 mno 21 F 25000
004 rst 28 M 23000
005 xyz 30 M 34000
;
Run;
```



```
proc sql noprint;
select sum(sal) into: salary from ds1;
Quit;
%Put &salary.;
```

```
Proc sql noprint;
Select name into:empname from ds1 ;
Quit;
%Put &empname.;
```

Above macro variable &empname prints only first empname(abc) but if you required all the values of empname use below program.

```
Proc sql noprint;
Select name into:empname separated by ' ' from ds1 ;
Quit;
%Put &empname.;
```

Observe in below real time Report how I created Macros thru Sql and how I used those macros in required places


SQL_MACRO.docx

PARAMETERS

names one or more local macro variables whose values you specify when you invoke the macro. A parameter list can contain any number of macro parameters separated by commas. The macro variables in the parameter list are usually referenced in the macro.

Syntax:- %MACRO MACRONAME(parameter-1><. . . ,parameter-n>);

Types of Parameters

- Positional Parameters
- Keyword Parameters

Positional Parameters

specifies one or more positional parameters. You can specify positional parameters in any order, but in the macro invocation, the order in which you specify the values must match the order you list them in the %MACRO statement. If you define more than one positional parameter, use a comma to separate the parameters.

Keyword Parameters

Names one or more macro parameters followed by equal signs. You can specify default values after the equal signs. If you omit a default value after an equal sign, the keyword parameter has a null value. Using default values enables you to write more flexible macro definitions and reduces the number of parameters that must be specified to invoke the macro. To override the default value, specify the macro variable name followed by an equal sign and the new value in the macro invocation.

Examples:-

Positional Parameters

Importing

```
%MACRO IMPORT(FILE,DNA
ME);
PROC IMPORT DATAFILE="&FILE" OUT=&DNAME
DBMS=EXCEL REPLACE;
RUN;
%MEND;
%IMPORT(C:\DOCUMENTS AND
SETTINGS\TSIPL0403\DESKTOP\KRISHNA\MACROS_EX\DATA1.XLS,STD_DATA);
%IMPORT(C:\DOCUMENTS AND
SETTINGS\TSIPL0403\DESKTOP\KRISHNA\MACROS_EX\DATA2.XLS,EMP_DATA);
```

Exporting

```
%MACRO EXPORT(FILE,DNAME);
PROC EXPORT OUTFILE="&FILE" DATA=&DNAME
DBMS=EXCEL REPLACE;
RUN;
```

```
%MEND;
%EXPORT(C:\DOCUMENTS AND
SETTINGS\TSIPL0403\DESKTOP\KRISHNA\MACROS_EX\STD_DATA.XLS,WORK.STD_DATA);
%EXPORT(C:\DOCUMENTS AND
SETTINGS\TSIPL0403\DESKTOP\KRISHNA\MACROS_EX\EMP_DATA.XLS,WORK.EMP_DATA);
```

Subsetting

```
%MACRO SUBSET(DSN1,DSN2);
DATA &DSN1;
SET &DSN2;
RUN;
%MEND;
%SUBSET(DS1,SASHELP.CLASS);
%SUBSET(DS2,WORK.DEMO);
%SUBSET(DS3,KRISHNA.PROD);

%MACRO SUBSET(NEW,ENAME,COND);
DATA &NEW;
SET &ENAME;
WHERE &COND;
RUN;
%MEND;
%SUBSET(DEMO1,SASHELP.CLASS,AGE>=12);
%SUBSET(DEMO2,SASHELP.CLASS,%STR(SEX='M'));
%SUBSET(DEMO3,SASHELP.CLASS,%NRSTR(SEX='M' & AGE>=12));

%MACRO PRINT(DSN);
PROC PRINT DATA=&DSN;
RUN;
%MEND;
%PRINT(WORK.DEMO);
%PRINT(KRISHNA.DATASET);
%PRINT(SASHELP.CLASS);
```

Plots

```
%MACRO MAC11(DSN,VAXIS,HAXIS);
PROC GPLOT DATA=&DSN;
PLOT &VAXIS*&HAXIS;
RUN;
%MEND;
%MAC11(SASHELP.CLASS, NAME,HEIGHT);
%MAC11(SASHELP.CLASS, NAME,WEIGHT);
```

Data Access

```
%MACRO IMP(DB,CON,TSH,DNAME);
PROC SQL;
CONNECT TO &DB(&CON);
CREATE TABLE &DNAME AS
SELECT*FROM CONNECTION TO &DB
(SELECT*FROM &TSH);
```

```
DISCONNECT FROM &DB;
QUIT;
%MEND;
%IMP(EXCEL,%STR(PATH='D:\SAS SOURCE\XLS\ALLESH.XLS'),[SAMPLE$],DS1);
%IMP(EXCEL,%STR(PATH='D:\SAS SOURCE\XLS\DEMO.XLS'),[SHEET2$],DS2);
%IMP(ACCESS,%STR(PATH='D:\SAS SOURCE\MS-ACCESS\DATA.MDB'),EMPLOY,DS3);
%IMP(ORACLE,(USER=SCOTT PASSWORD=TIGER),DEPT,DS4);
```

Keyword Parameters

```
%MACRO MAC11(DSN=, VAXIS=, HAXIS=);
PROC GPLOT DATA=&DSN;
PLOT &VAXIS*&HAXIS;
RUN;
%MEND;
%MAC11(VAXIS=NAME, HAXIS=HEIGHT, DSN=SASHELP.CLASS);
%MAC11(DSN=SASHELP.CLASS, VAXIS=NAME, HAXIS=WEIGHT);

%MACRO PRINT(DSN=);
PROC PRINT DATA=&DSN;
RUN;
%MEND;
%PRINT(DSN=WORK.DEMO);
%PRINT(DSN=KRISHNA.DATASET);
%PRINT(DSN=SASHELP.CLASS);
```


%DO Statement

```
DATA DS;
DO I=1 TO 10;
OUTPUT;
END;
STOP;
RUN;
```

Appending

```
%MACRO MAC15(DS);
PROC APPEND BASE=DSS DATA=&DS FORCE;
RUN;
%MEND;
%MAC15(DS1);
%MAC15(DS14);
%MAC15(DS24);

%MACRO MAC15(DS);
```

```
%DO I=1 %TO 5;
%LET VAR1=%SYSFUNC(CAT(&DS,&I));
PROC APPEND BASE=DSS DATA=&VAR1 FORCE;
RUN;
%END;
%MEND;
%MAC15(DS);
```


MACRO AUTO CALL LIBRARIES

We can store our macros in a central location called an autocall library
Macros from that library can be shared by programs and programmers
Macros default stores in work library,
Using MSTORED SASMSTORE options we can store Macros in to required libraries.

Syntax:-MSTORED SASMSTORE =<AUTOCALL LIBRARY>;

Example:-

```
OPTIONS MSTORED SASMSTORE=KRISHNA;
%MACRO PRINT(DNAME)/STORE;
 PROC PRINT DATA=&DNAME;
 RUN;
%MEND;
%PRINT
```

Use the MAUTOSOURCE and SASAUTOS= system options to tell SAS where to look for macros. Then you can invoke a macro even though the original macro does not appear in your program.

MSTORED:- Use stored compiled macros

SASMSTORE:- Libref associated with SAS data library containing catalog of compiled stored macros (specify a library where you want to store your macros permanent)

MAUTOSOURCE:- Specifies whether the autocall feature is available.
Syntax:- SASAUTOS=Library specification;

SASAUTOS= system option:- Specifies the location of one or more autocall libraries.
Syntax:- SASAUTOS=Library specification;

Example:-

```
OPTIONS MSTORED SASMSTORE=KRISHNA;
OPTIONS SASAUTOS=KRISHNA;

%PRINT(SASHELP.CARS);
```

NESTED MACRO

If we write a macro block or macro call inside of the another macro block
Then these macro blocks are called Nested Macros.

```
%MACRO PRINT(DS);
PROC PRINT DATA=&DS;
RUN;
%MEND;
%MACRO SPRINT(DNAME1, DNAME2, VAR);
PROC SORT DATA=&DNAME1 OUT=&DNAME2;
BY &VAR;
RUN;
%PRINT(&DNAME2);
%MEND;
%SPRINT (SASHELP.CLASS,DS,SEX);
```


DEBUGGING MACRO ERRORS

Checking for errors and eliminating errors from program is called debugging

Macro Debugging:- Checking for errors in macro program to avoid errors.

How we can make a Bug free Macro Program

-> Write a standard program without bugs
-> Add your %MACRO and %MEND statements for standard sas program
-> add the macro logic one feature at a time
-> add your macro variables, one at a time, and so on, until your macro is complete and bug-free

Macro Debugging Options

NOMERROR|MERROR
NOSERROR|SERROR
NOMLOGIC|MLOGIC
NOMPRINT|MPRINT
NOSYMBOLGEN|SYMBOLGEN

MERROR | NOMERROR

when this option is on, SAS will issue a warning if you invoke a macro that SAS cannot find.

```
OPTIONS MERROR;
%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &d2;
Where &cond;
Run;
%Mend;
%Mac1(ds1,sashelp.class,%str(sex='F'));
%Mac1(ds2,sashelp.class,%nrstr(age>=14));
```

WARNING: Apparent invocation of macro MAC1 not resolved.
Why because we created a Macro called **MAC11** but while invoking we specified **Mac1**. So use **%Mac11** while invoking a macro.

Check below program and see the log there should be no Warning because MERROR option is turned off.

```
OPTIONS NOMERROR;
%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &d2;
Where &cond;
Run;
%Mend;
%Mac1(ds1,sashelp.class,%str(sex='F'));
%Mac1(ds2,sashelp.class,%nrstr(age>=14));
```

--------Correct Program-------

```
OPTIONS MERROR;
%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &d2;
Where &cond;
Run;
%Mend;
%Mac11(ds1,sashelp.class,%str(sex='F'));
%Mac11(ds2,sashelp.class,%nrstr(age>=14));
```

SERROR | NOSERROR

when this option is on, SAS will issue a warning if you use a macro variable that SAS cannot find.

```
OPTIONS SERROR;
%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &ds2;
```

```
Where &cond;
Run;
%Mend;
%Mac11(ds1,sashelp.class,%str(sex='F'));
%Mac11(ds2,sashelp.class,%nrstr(age>=14));
```

WARNING: Apparent symbolic reference DS2 not resolved.
Why because we created a Macro variable called **d2** but while resolving we specified **ds2**
So use **&d2** while resolving a macro variable.

Check below program and see the log there should be no Warning because SERROR option is turned off.

```
OPTIONS NOSERROR;
%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &ds2;
Where &cond;
Run;
%Mend;
%Mac1(ds1,sashelp.class,%str(sex='F'));
%Mac1(ds2,sashelp.class,%nrstr(age>=14));
```

--------Correct Program-------

```
OPTIONS SERROR;
%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &d2;
Where &cond;
Run;
%Mend;
%Mac11(ds1,sashelp.class,%str(sex='F'));
%Mac11(ds2,sashelp.class,%nrstr(age>=14));
```

MLOGIC | NOMLOGIC

when this option is on, SAS prints in your log details about the execution of macros and Macro variables.

```
OPTIONS MLOGIC;
%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &d2;
Where &cond;
Run;
%Mend;
%Mac11(ds1,sashelp.class,%str(sex='F'));
%Mac11(ds2,sashelp.class,%nrstr(age>=14));
```

```
MLOGIC(MAC11):  Beginning execution.
MLOGIC(MAC11):  Parameter D1 has value ds1
MLOGIC(MAC11):  Parameter D2 has value sashelp.class
```

```
MLOGIC(MAC11):  Parameter COND has value _sex_'F'_

NOTE: There were 9 observations read from the data set SASHELP.CLASS.
      WHERE sex='F';
NOTE: The data set WORK.DS1 has 9 observations and 5 variables.

MLOGIC(MAC11):  Ending execution.
149  %Mac11(ds2,sashelp.class,%nrstr(age>=14));
MLOGIC(MAC11):  Beginning execution.
MLOGIC(MAC11):  Parameter D1 has value ds2
MLOGIC(MAC11):  Parameter D2 has value sashelp.class
MLOGIC(MAC11):  Parameter COND has value _age__14_

NOTE: There were 9 observations read from the data set SASHELP.CLASS.
      WHERE age>=14;
NOTE: The data set WORK.DS2 has 9 observations and 5 variables.
```

If you don't want to prints in your log, details about the execution of macros and macro variables

```
OPTIONS NOMLOGIC;
%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &d2;
Where &cond;
Run;
%Mend;
%Mac11(ds1,sashelp.class,%str(sex='F'));
%Mac11(ds2,sashelp.class,%nrstr(age>=14));
```


MPRINT | NOMPRINT

when this option is on, SAS prints in your log the standard SAS code generated by macro processor.

```
OPTIONS MPRINT;
%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &d2;
Where &cond;
Run;
%Mend;
%Mac11(ds1,sashelp.class,%str(sex='F'));
%Mac11(ds2,sashelp.class,%nrstr(age>=14));
```

```
%Mac11(ds1,sashelp.class,%str(sex='F'));
MPRINT(MAC11):   Data ds1;
MPRINT(MAC11):   Set sashelp.class;
MPRINT(MAC11):   Where sex='F';
MPRINT(MAC11):   Run;

%Mac11(ds2,sashelp.class,%nrstr(age>=14));
MPRINT(MAC11):   Data ds2;
MPRINT(MAC11):   Set sashelp.class;
MPRINT(MAC11):   Where age>=14;
```

```
MPRINT(MAC11):   Run;
```

If you don't want to prints in your log, details about the standard SAS code generated by macros.

```
OPTIONS NOMPRINT;

%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &d2;
Where &cond;
Run;
%Mend;
%Mac11(ds1,sashelp.class,%str(sex='F'));
%Mac11(ds2,sashelp.class,%nrstr(age>=14));
```

SYMBOLGEN | NOSYMBOLGEN

when this option is on, SAS prints in your log the values of macro variables.

```
OPTIONS SYMBOLGEN;
%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &d2;
Where &cond;
Run;
%Mend;

%Mac11(ds1,sashelp.class,%str(sex='F'));
%Mac11(ds2,sashelp.class,%nrstr(age>=14));
```



```
%Macro Mac11(d1, d2, cond) ;
%Mac11(ds1,sashelp.class,%str(sex='F'));
SYMBOLGEN:  Macro variable D1 resolves to ds1
SYMBOLGEN:  Macro variable D2 resolves to sashelp.class
SYMBOLGEN:  Macro variable COND resolves to sex='F'
SYMBOLGEN:  Some characters in the above value which were subject to macro quoting have
been unquoted for printing.

%Mac11(ds2,sashelp.class,%nrstr(age>=14));
SYMBOLGEN:  Macro variable D1 resolves to ds2
SYMBOLGEN:  Macro variable D2 resolves to sashelp.class
SYMBOLGEN:  Macro variable COND resolves to age>=14
SYMBOLGEN:  Some characters in the above value which were subject to macro quoting have
been unquoted for printing.
```

If you don't want to prints in your log, details about the values of macro variables.

```
OPTIONS NOSYMBOLGEN;

%Macro Mac11(d1, d2, cond) ;
Data &d1;
Set &d2;
Where &cond;
Run;
```

```
%Mend;
%Mac11(ds1,sashelp.class,%str(sex='F'));
%Mac11(ds2,sashelp.class,%nrstr(age>=14));
```

While you want the MERROR and SERROR options to be on at all times, you will probably want to turn on MLOGIC, MPRINT, and SYMBOLGEN one at a time.

MACRO FUNCTIONS

%Sysfunc

Execute SAS functions or user-written functions.

If we are using any Base SAS function in Macros that should call with %Sysfunc.

Syntax: - %SYSFUNC(Function(argument-1 <...argument-n>)<, format>)

Examples:-

```
%LET TODAY=%SYSFUNC(TODAY(),DATE9.);
%LET MON=%SYSFUNC(SUBSTR(&TODAY1.D,3,3));
TITLE "%SYSFUNC(DATE(),WORDDATE.) Absence Report";
```

The macro CHECKDS uses %SYSFUNC to execute the EXIST function, which checks the existence of a data set:

Execute below program and see the result

```
%macro checkds1(dsn);
  %if %sysfunc(exist(&dsn)) %then
    %do;
      proc print data=&dsn;
      Run;
    %end;
    %else
      %put The data set &dsn does not exist.;
```

```
%mend;
%checkds1(work.ds) ;
Result: The data set work.ds does not exist.
```

Execute below program and see the result

```
Data ds;
Set sashelp.cars;
Run;

%macro checkds(dsn);
  %if %sysfunc(exist(&dsn)) %then
    %do;
      Proc print data=&dsn;
      Run;
    %end;
    %else
      %put The data set &dsn does not exist.;
%mend;
%checkds(work.ds)
```

Above program check the dataset is exist or not, here dataset is there so it produces the statements like
PROC PRINT DATA=WORK.DS;
RUN;

%EVAL

Evaluates arithmetic and logical expressions using integer arithmetic.

Syntax:- %EVAL (arithmetic or logical expression)

Example:-

```
%Macro mac4;
%Let a=2;
%Let b=3;
%let c=%Eval(&a+&b);
%Mend;
%Mac4
%Put &a.;
%Put &b.;
%Put &c.;
```

%SYSEVALF

Evaluates arithmetic and logical expressions using floating-point arithmetic.

Note: When we need to do some arithmetic operations without decimal values we can use **%Eval**, but when we have decimal values we need to use **%Sysevalf** function.

Syntax:- %SYSEVALF (arithmetic or logical expression)

Example:-

```
%Macro mac5;
%Let a=2.3;
```

```
%Let b=3.2;
%let c=%Sysevalf(&a+&b);
%Mend;
%Mac5
%Put &a.;
%Put &b.;
%Put &c.;

%Macro mac6;
%Let a=2.3;
%Let b=3.2;
%let c=%Sysevalf(&a+&b);
%let d=%Sysevalf(&a+&b,boolean);
%let e=%Sysevalf(&a+&b,ceil);
%let f=%Sysevalf(&a+&b,floor);
%let g=%Sysevalf(&a+&b,int);
%Mend;
%Mac6
%Put &a.;
%Put &b.;
%Put &c.;
%Put The boolean value is: &d.;
%Put The ceil value is: &e.;
%Put The floor value is: &f.;
%Put The int value is: &g.;
```


SYMGET

Returns the value of a macro variable during DATA step execution.

Get the information from Macro variable to Datastep.

Syntax:- Symget(Argument)

Example:-

```
%Let a=yes;
Data ds;
b=Symget("a");
Run;
```

%LENGTH

Returns the length of a string.

Syntax:- **%LENGTH (character string)**

Example:-

```
%Let A=Krishna;
%Let B=Reddy;
%Put  A Variable value Krishna length is : %Length(&A);
%Put  B Variable value Reddy length is : %Length(&B);
%Put  A and B Variable value length is : %Length(&A&B);
```

%SCAN & %QSCAN

Search for a word that is specified by its position in a string.

Syntax:- %SCAN(argument, n<, delimiters>)

%QSCAN masks the same characters as the %NRBQUOTE function.

Syntax:- %QSCAN(argument, n<, delimiters>)

Example:-

```
%macro a;
  aaaaaa
%mend a;
%macro b;
  bbbbbb
%mend b;
```

```
%macro c;
  cccccc
%mend c;

%let x=%nrstr(%a*%b*%c);
%put X: &x;
%put The third word in X, with SCAN: %scan(&x,3,*);
%put The third word in X, with QSCAN: %qscan(&x,3,*);
```

%SUBSTR & %QSUBSTR

Search for a word that is specified by its position in a string.

Syntax:- %SUBSTR (argument, position<, length>)
%QSUBSTR (argument, position<, length>)

Example:-

```
%let a=one;
%let b=two;
%let c=%nrstr(&a &b);
%put C: &c;
%put With SUBSTR: %substr(&c,1,2);
%put With QSUBSTR: %qsubstr(&c,1,2);
```

Observe below real time code

```
%MACRO DATES;

%GLOBAL TODAY1 YEARA MONA MON1 YR1 DAY MON2 YR2 QBYR EDATE
BEGPRD CQMON CQMON1 CQYR FBDATE FEDATE LFBDATE LEDATE LFEDATE
Q1MON Q1MON1 Q2MON Q2MON1 Q3MON Q3MON1 EYR EMON BMON1A
BMON1 BYR1 BDATE BYR BMON M2A M2B M2 Y2 M3A M3B M3 Y3 M4A
M4B M4 Y4 M5A M5B M5 Y5 M6A M6B M6 Y6 M7A M7B M7 Y7 M8A
M8B M8 Y8 M9A M9B M9 Y9 M10A M10B M10 Y10 M11 M11B M11 Y11
M12A M12B M12 Y13 Y14 Y15 Y16 Y17 Y18 Y19 Y20 Y21 Y22 Y23 Y24
MON1 MON2 MON3 MON4 MON5 MON6 MON7 MON8 MON9 MON10
MON11 MON12 MON13 MON14 MON15 MON16 MON17 MON18 MON19
MON20 MON21 MON22 MON23 MON24 FMT_BDATE FMT_EDATE ;

%LET TODAY1=%SYSFUNC(TODAY(),YYMMDDN8.);

%LET YEARA = %EVAL(&TODAY1./10000);

%LET MONA = %EVAL(%EVAL(&TODAY1./100) - %EVAL(&YEARA. * 100));

%IF &MONA. EQ 1 %THEN %LET MON1 =12;
%ELSE %LET MON1 =%EVAL(&MONA. - 1);

%IF &MONA. EQ 1 %THEN %LET YR1 =%EVAL(&YEARA - 1);
%ELSE %LET YR1 =&YEARA.;

%IF &MON1. EQ 2 %THEN %LET DAY =28;
%ELSE %IF &MON1. EQ 4 OR &MON1. EQ 6 OR &MON1. EQ 9 OR &MON1. EQ 11
%THEN %LET DAY =30;
   %ELSE %LET DAY =31;
```

```
%LET A=0;

%LET B=1;

%IF &MON1. GT 9 %THEN %LET MON2 =&MON1.;
%ELSE %LET MON2 =&A.&MON1.;

%LET YR2 = &YR1.;

%LET QBYR = %EVAL(&YR1. - 2);

%LET EDATE =&YR2.&MON2.&DAY.;

%LET BEGPRD = &QBYR.&MON2.&A.&B.;

%IF &MON1. EQ 1 %THEN %LET CQMON =11;
%ELSE %IF &MON1. EQ 2 %THEN %LET CQMON = 12;
%ELSE %LET CQMON = %EVAL(&MON1. - 2);

%IF &CQMON. GT 9 %THEN %LET CQMON1 =&CQMON.;
%ELSE %LET CQMON1 =%SYSFUNC(CAT(0,&CQMON.));

%IF &MON1. EQ 1 %THEN %LET CQYR =%EVAL(&YR2. - 1);
%ELSE %IF &MON1. EQ 2 %THEN %LET CQYR =%EVAL(&YR2. - 1);
%ELSE %LET CQYR =&YR2.;

%LET FBDATE = &CQYR.&CQMON1.&A.&B.;

%LET FEDATE = &EDATE.;

%LET LFBDATE = %EVAL(&FBDATE. - 10000);

%LET LEDATE = %EVAL(&EDATE. - 10000);

%LET LFEDATE = &LEDATE.;

%IF &CQMON. GT 3 %THEN %LET Q1MON = %EVAL(&CQMON. - 3);
%ELSE %LET Q1MON = %EVAL(&CQMON.+ 9);

%IF &Q1MON. GT 9 %THEN %LET Q1MON1 =&Q1MON.;
%ELSE %LET Q1MON1 =&A.&Q1MON.;

%IF &Q1MON. GT 3 %THEN %LET Q2MON =%EVAL(&Q1MON. - 3);
%ELSE %LET Q2MON =%EVAL(&Q1MON. + 9);

%IF &Q2MON. GT 9 %THEN %LET Q2MON1 =&Q2MON.;
%ELSE %LET Q2MON1 =&A.&Q2MON.;

%IF &Q2MON. GT 3 %THEN %LET Q3MON =%EVAL(&Q2MON - 3);
%ELSE %LET Q3MON =%EVAL(&Q2MON. + 9);

%IF &Q3MON. GT 9 %THEN %LET Q3MON1 = &Q3MON.;
%ELSE %LET Q3MON1 = &A.&Q3MON.;

%LET EYR = %EVAL(&EDATE./10000);

%LET  EMON = %SUBSTR(&EDATE.,5,2);

%IF &EMON. NE 12 %THEN %LET BMON1A =%EVAL(&EMON. + 1);
%ELSE %LET BMON1A =1;

%IF &BMON1A GE 10 %THEN %LET  BMON1 =&BMON1A.;
```

```
%ELSE %LET BMON1 =&A.&BMON1A.;

%IF &EMON EQ 12 %THEN %LET BYR1 =%EVAL(&EYR. - 1);
%ELSE %LET BYR1 =%EVAL(&EYR. - 2);

%LET A=0;

%LET B=1;

%LET BDATE = &BYR1.&BMON1.&A.&B.;

%LET BYR = %EVAL(&BDATE./10000);

%LET BMON = %SUBSTR(&BDATE.,5,2);

%LET M2A =%EVAL(&BMON. + 1);

%IF &M2A. GE 13 %THEN %LET M2B =%EVAL(&M2A. - 12);
%ELSE %LET M2B = &M2A.;

%IF &M2B. GE 10 %THEN %LET M2 = &M2B.;
%ELSE %LET M2 = &A.&M2B.;

%IF &M2A. GE 13 %THEN %LET Y2 =%EVAL(&BYR. + 1);
%ELSE %LET Y2 =&BYR.;

%LET M3A = %EVAL(&BMON + 2);


%IF &M3A. GE 13 %THEN %LET M3B =%EVAL(&M3A. - 12);
%ELSE %LET M3B =&M3A.;

%IF &M3B. GE 10 %THEN %LET M3 =&M3B.;
%ELSE %LET M3 =&A.&M3B.;

%IF &M3A. GE 13 %THEN %LET Y3 =%EVAL(&BYR. + 1);
%ELSE %LET Y3 =&BYR.;

%LET M4A = %EVAL(&BMON. + 3);

%IF &M4A. GE 13 %THEN %LET M4B = %EVAL(&M4A. - 12);
%ELSE %LET M4B = &M4A.;

%IF &M4B. GE 10 %THEN %LET M4 =&M4B.;
%ELSE %LET M4 =&A.&M4B.;

%IF &M4A. GE 13 %THEN %LET Y4 =%EVAL(&BYR. + 1);
%ELSE %LET Y4 =&BYR.;

%LET M5A = %EVAL(&BMON. + 4);

%IF &M5A. GE 13 %THEN %LET M5B =%EVAL(&M5A. - 12);
%ELSE %LET M5B =&M5A.;

%IF &M5B. GE 10 %THEN %LET M5 =&M5B.;
%ELSE %LET M5 =&A.&M5B.;

%IF &M5A. GE 13 %THEN %LET Y5=%EVAL(&BYR. + 1);
%ELSE %LET Y5 =&BYR.;

%LET M6A = %EVAL(&BMON. + 5);

%IF &M6A. GE 13 %THEN %LET M6B =%EVAL(&M6A. - 12);
```

```
%ELSE %LET M6B =&M6A.;

%IF &M6B. GE 10 %THEN %LET M6 =&M6B.;
%ELSE %LET M6 =&A.&M6B.;

%IF &M6A. GE 13 %THEN %LET Y6 =%EVAL(&BYR. + 1);
%ELSE %LET Y6 =&BYR.;

%LET M7A = %EVAL(&BMON. + 6);

%IF &M7A. GE 13 %THEN %LET M7B =%EVAL(&M7A. - 12);
%ELSE %LET M7B =&M7A.;

%IF &M7B. GE 10 %THEN %LET M7 =&M7B.;
%ELSE %LET M7 =&A.&M7B.;

%IF &M7A. GE 13 %THEN %LET Y7 =%EVAL(&BYR. + 1);
%ELSE %LET Y7 =&BYR.;

%LET M8A = %EVAL(&BMON. + 7);

%IF &M8A. GE 13 %THEN %LET M8B =%EVAL(&M8A. - 12);
%ELSE %LET M8B =&M8A.;

%IF &M8B. GE 10 %THEN %LET M8 =&M8B.;
%ELSE %LET M8 =&A.&M8B.;


%IF &M8A. GE 13 %THEN %LET Y8 =%EVAL(&BYR. + 1);
%ELSE %LET Y8 =&BYR.;

%LET M9A = %EVAL(&BMON. + 8);
%IF &M9A. GE 13 %THEN %LET M9B =%EVAL(&M9A. - 12);
%ELSE %LET M9B =&M9A.;

%IF &M9B. GE 10 %THEN %LET M9 =&M9B.;
%ELSE %LET M9 =&A.&M9B.;

%IF &M9A. GE 13 %THEN %LET Y9 =%EVAL(&BYR. + 1);
%ELSE %LET Y9 =&BYR.;

%LET M10A = %EVAL(&BMON. + 9);

%IF &M10A. GE 13 %THEN %LET M10B =%EVAL(&M10A. - 12);
%ELSE %LET M10B =&M10A.;

%IF &M10B. GE 10 %THEN %LET M10 =&M10B.;
%ELSE %LET M10 =&A.&M10B;

%IF &M10A. GE 13 %THEN %LET Y10 =%EVAL(&BYR. + 1);
%ELSE %LET Y10 =&BYR.;

%LET M11A = %EVAL(&BMON. + 10);

%IF &M11A. GE 13 %THEN %LET M11B =%EVAL(&M11A. - 12);
%ELSE %LET M11B =&M11A.;

%IF &M11B. GE 10 %THEN %LET M11 =&M11B.;
%ELSE %LET M11 =&A.&M11B.;

%IF &M11A. GE 13 %THEN %LET Y11 = %EVAL(&BYR. + 1);
```

```
%ELSE %LET Y11 = &BYR.;
%LET M12A = %EVAL(&BMON. + 11);
%IF &M12A. GE 13 %THEN %LET M12B =%EVAL(&M12A. - 12);
%ELSE %LET M12B =&M12A.;
%IF &M12B. GE 10 %THEN %LET M12 =&M12B.;
%ELSE %LET M12 =&A.&M12B.;
%IF &M12A. GE 13 %THEN %LET Y12 =%EVAL(&BYR. + 1);
%ELSE %LET Y12 =&BYR.;
%LET Y13 = %EVAL(&BYR. + 1);
%LET Y14 = %EVAL(&Y2. + 1);
%LET Y15 = %EVAL(&Y3. + 1);
%LET Y16 = %EVAL(&Y4. + 1);
%LET Y17 = %EVAL(&Y5. + 1);
%LET Y18 = %EVAL(&Y6. + 1);
%LET Y19 = %EVAL(&Y7. + 1);
%LET Y20 = %EVAL(&Y8. + 1);
%LET Y21 = %EVAL(&Y9. + 1);
%LET Y22 = %EVAL(&Y10. + 1);
%LET Y23 = %EVAL(&Y11. + 1);
%LET Y24 = %EVAL(&Y12. + 1);
%LET MON1 = &BYR.&BMON.;
%LET MON2 = &Y2.&M2.;
%LET MON3 =  &Y3.&M3.;
%LET MON4 =  &Y4.&M4.;
%LET MON5 =  &Y5.&M5.;
%LET MON6 =  &Y6.&M6.;
%LET MON7 =  &Y7.&M7.;
%LET MON8 =  &Y8.&M8.;
%LET MON9 =  &Y9.&M9.;
%LET MON10 =  &Y10.&M10.;
%LET MON11 =  &Y11.&M11.;
%LET MON12 =  &Y12.&M12.;
%LET MON13 =  &Y13.&BMON.;
%LET MON14 =  &Y14.&M2.;
%LET MON15 =  &Y15.&M3.;
```

```
%LET MON16 =  &Y16.&M4.;
%LET MON17 =  &Y17.&M5.;
%LET MON18 =  &Y18.&M6.;
%LET MON19 =  &Y19.&M7.;
%LET MON20 =  &Y20.&M8.;
%LET MON21 =  &Y21.&M9.;
%LET MON22 =  &Y22.&M10.;
%LET MON23 =  &Y23.&M11.;
%LET MON24 =  &Y24.&M12.;
%LET FMT_BDATE = &BDATE.;
%LET FMT_EDATE = &EDATE.;
%MEND;
%DATES
%PUT &TODAY1.;
%PUT &YEARA.;
%PUT &MONA.;
%PUT &MON1.;
%PUT &YR1.;
%PUT &DAY.;
%PUT &MON2.;
%PUT &YR2.;
%PUT &QBYR.;
%PUT &BEGPRD.;
%PUT &CQMON.;
%PUT &CQMON1.;
%PUT &CQYR.;
%PUT &FBDATE.;
%PUT &FEDATE.;
%PUT &LFBDATE.;
%PUT &LEDATE.;
%PUT &LFEDATE.;
%PUT &Q1MON.;
%PUT &Q1MON1.;
%PUT &Q2MON.;
%PUT &Q2MON1.;
%PUT &Q3MON.;
%PUT &Q3MON1.;
%PUT &EYR.;
%PUT &EMON.;
```

```
%PUT &BMON1A.;
%PUT &BMON1.;
%PUT &BYR1.;
%PUT &EDATE.;
%PUT &BDATE.;
%PUT &BYR.;
%PUT &BMON.;
%PUT &M2A.;
%PUT &M2B.;
%PUT &M2.;
%PUT &Y2.;
%PUT &M3A.;
%PUT &M3B.;
%PUT &M3.;
%PUT &Y3.;
%PUT &M4A.;
%PUT &M4B.;
%PUT &M4.;
%PUT &Y4.;
%PUT &M5A.;
%PUT &M5B.;
%PUT &M5.;
%PUT &Y5.;
%PUT &M6A.;
%PUT &M6B.;
%PUT &M6.;
%PUT &Y6.;
%PUT &M7A.;
%PUT &M7B.;
%PUT &M7.;
%PUT &Y7.;
%PUT &M8A.;
%PUT &M8B.;
%PUT &M8.;
%PUT &Y8.;
%PUT &M9A.;
%PUT &M9B.;
%PUT &M9.;
%PUT &Y9.;
%PUT &M10A.;
%PUT &M10B.;
```

```
%PUT &M10.;
%PUT &Y10.;
%PUT &M11A.;
%PUT &M11B.;
%PUT &M11.;
%PUT &Y11.;
%PUT &M12A.;
%PUT &M12B.;
%PUT &M12.;
%PUT &Y13.;
%PUT &Y14.;
%PUT &Y15.;
%PUT &Y16.;
%PUT &Y17.;
%PUT &Y18.;
%PUT &Y19.;
%PUT &Y20.;
%PUT &Y21.;
%PUT &Y22.;
%PUT &Y23.;
%PUT &Y24.;
%PUT &MON1.;
%PUT &MON2.;
%PUT &MON3.;
%PUT &MON4.;
%PUT &MON5.;
%PUT &MON6.;
%PUT &MON7.;
%PUT &MON8.;
%PUT &MON9.;
%PUT &MON10.;
%PUT &MON11.;
%PUT &MON12.;
%PUT &MON13.;
%PUT &MON14.;
%PUT &MON15.;
%PUT &MON16.;
%PUT &MON17.;
%PUT &MON18.;
%PUT &MON19.;
%PUT &MON20.;
```

```
%PUT &MON21.;
%PUT &MON22.;
%PUT &MON23.;
%PUT &MON24.;
%PUT &FMT_BDATE.;
%PUT &FMT_EDATE.;
```


MULTIPLE AMBERSANDS

```
%let a=b;
%let b=c;
%let c=a;
%put &a;
%put &&a;
%put &&&a;
%put &&&&a;
%put &&&&&a;
%put &&&&&&a;
%put &&&&&&&a;
%put &&&&&&&&a;
%put &&&&&&&&&a;
```

```
%let a=b;
%let b=c;
%let c=d;
%put &a;
%put &&a;
%put &&&a;
%put &&&&a;
%put &&&&&a;
%put &&&&&&a;
%put &&&&&&&a;
```

```
%put &&&&&&&&a;
%put &&&&&&&&&a;


%let a=1;
%let 1=a;
%let c=3;
%put &a;
%put &&a;
%put &&&a;
%put &&&&a;
%put &&&&&a;
%put &&&&&&a;
%put &&&&&&&a;
%put &&&&&&&&a;
%put &&&&&&&&&a;
```