ABOUT INDEX CONTACT

LISTEN DATA
MAKE YOUR DATA TELL A STORY

HOME SAS R PYTHON DATA SCIENCE CREDIT RISK SQL EXCEL CALCULATORS OTHERS

SEARCH... GO

Home » SAS » SAS Macros Made Easy

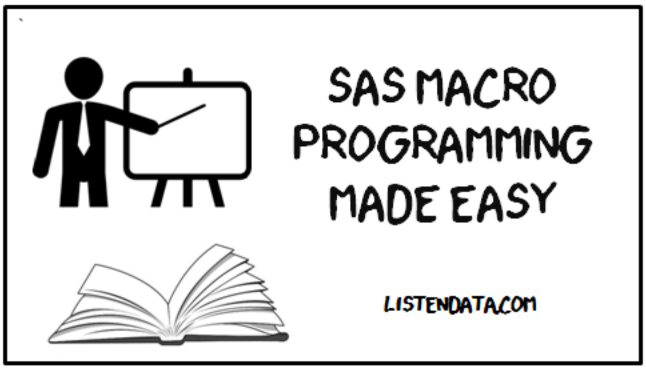
Follow us on Facebook

SAS MACROS MADE EASY

Deepanshu Bhalla 43 Comments SAS, SAS Macros

This tutorial explains SAS macros with practical examples. SAS Macro programming is considered advanced in SAS. Knowing SAS Macros gives you an advantage in the job market over other candidates. Upon completion of this tutorial, you will understand how to create SAS macros and where they can be used.

It is generally said that smart programming reduces workload and helps you win over your boss.



SAS Macro for Beginners

Table of Contents

Introduction to SAS Macros

SAS Macros are used to automate the repetitive task. It can make your work faster by automating the task that requires writing same lines of code every day. It can also be used when you design a complex algorithm and want to make usage of code user friendly so that people who are not comfortable with programming can use your algorithm.

Fundamentals of SAS Macros

The fundamentals of SAS macros involve understanding the basic concepts and components of SAS macro programming. Here are the basics you need to know:

Macro Variables

A macro variable is used to store a value in SAS. The value is always character. The character value can include variable-name, letters, numbers or any text you want substituted in your program.

Macro Variables are of two types -

1. **Local -** If the macro variable is defined inside a macro code, then scope is local. It would be available for use in that macro only and gets removed when the macro is finished.
2. **Global -** If the macro variable is defined outside a macro code, then scope is global. It can be use any where in the SAS program and gets removed at the end of the session.

5 ways to create SAS macro variables -

The following is a list of various ways to create a macro variable in SAS, along with examples.

1. %LET

The %LET can defined inside or outside a macro.

The syntax of the %LET statement is as follows -

> %LET macro-variable-name = value;

```
%LET x = 5;
```

Example - To store system date.

```
%let dt = &sysdate;
```

How to use Macro Variables

Macro variables are referenced by using ampersand (&) followed by macro variable name.

```
& <Macro variable Name>
```

To view in log window what macro variable would return, use %PUT statement :

```
%put &dt.;
%put NOTE : system data is &dt.;
```

Result : *NOTE : system data is* ***23DEC16***
Notice that unlike the PUT statement the text string is not enclosed in quotes. The quotes are not needed because, unlike in the DATA step, the macro facility does not need to distinguish between variable names and text strings.

2. Macro Parameters

Suppose you are asked to write a macro that returns mean value of a variable. The analysis variable, input and output data sets are dynamic.

```
%macro test (input =, ivar=, output=);
proc means data = &input noprint;
var &ivar;
output out = &output mean= ;
run;
%mend;
```

In the above code, test is a macro; input, ivar and output are local macro variables.

How to call a macro -

```
%test(input=sashelp.heart, ivar= height, output=test);
```

In this case, we are giving flexibility to users to provide information of input dataset name, the analysis variable and output dataset and the macro returns the mean value of the analysis variable in the output dataset.

3. INTO clause in PROC SQL

Example : Suppose you are asked to calculate average height and store in a macro variable.

```
proc means data = sashelp.heart noprint;
var height;
output out = test mean= avg_height;
run;
```

```
proc sql noprint;
select avg_height into :var1
from test;
quit;
%put &var1;
```

Result : 64.81318

4. CALL SYMPUT routine

The syntax of CALL SYMPUT is as follows:

```
CALL SYMPUT(macro_varname,value);
```

```
data _null_;
set test;
call symput ('var2',avg_height);
run;
%put &var2;
```

5. ITERATIVE %DO

Below is the syntax of iterative %DO -

```
%DO macro-variable = start %TO stop <%BY increment>;
. . . text . . .
```

```
%END;
```

Example -

```
%macro calcl(start,stop);
%do year = &start %to &stop;
data test;
set yr&year;
year = 2000 + &year;
run;
%end;
%mend calcl;
```

How to use conditional processing %IF %THEN ?

In SAS Macros, we can apply conditional statements using %IF %THEN like below -

```
options mindelimiter=,;
options minoperator;
%MACRO test();
%DO i = 1 %to 9 ;
%if &i in (1,3,5,7,9) %then %do;%PUT i = &i - odd;
%END;
%ELSE %DO;%PUT i = &i - even;
%end;
%end;%MEND;
%test();
```

If you are confused between IF-THEN and %IF-THEN and don't know when to use, check out the link below -

Tutorial : **Difference between IF THEN and %IF %THEN**

SAS Macro Functions

There are some SAS macro functions which help you to execute various operations within a macro. Below is a list of SAS Macro Functions.

1. %EVAL Function

It is used to perform mathematical and logical operation with macro variables.

Example -

```
%let x = 10;
%let y = 20;
%let z = &x * &y;
%put &z;
```

It returns "10*20".

```
%let z2 = %eval(&x*&y);
%put &z2;
```

It returns 200.

Note :

%let last = %eval (4.5+3.2); returns error as it cannot perform arithmetic calculations with operands that have the floating point values. It is when the **%SYSEVALF** function comes into picture.

```
%let last2 = %sysevalf(4.5+3.2);
%put &last2;
```

It returns 7.7

2. %SYSFUNC Function

There are several useful Base SAS function that are not directly available in Macro, %Sysfunc enables those function to make them work in a macro.

```
%let dt3 = %sysfunc(date(),yymmdd10.);
```

It returns 2016-12-23.

3. %STR Function

Usage I : This function removes the normal meaning of following token + – * /, > < = ; " LT EQ GT LE GE LE NE AND OR NOT blank.

Suppose we need to store PROC PRINT; RUN; command in a macro variable.

```
%let exmp0 = proc print; run;;
%put &exmp0;
```

Result : proc print
Since the semicolon following PRINT terminates the %LET statement. It does not consider RUN statement. To workaround this issue, let's use **%STR function**.

```
%let exmpl = %str(proc print; run;) ;
%put &exmpl;
```

Result : proc print; run;

Usage II : Precede with % sign when you use single or double quotation in macro

```
%let var=%str(a%");
%put &var;
```

Result : a"
If you would not use %STR function in the above example, you would not be able to store quotes in a macro variable.

Usage III : It also preserves leading and trailing blanks of the string.

```
%let dt= %str( a );
%put &dt;
```

Run the above code and compare it with the code below, you would understand the difference -

```
%let dt=  a ;
%put &dt;
```

4. %NRSTR Function

%NRSTR works similar to %STR works except it does not resolve the % and & but stop the macro triggers.

```
%let exmpl = %nrstr(proc print; run;) ;
%put &exmpl;
```

Result : proc print; run;

```
%put "Difference between %NRSTR(&SYSDATE9) and &SYSDATE9";
```

Result : "Difference between &SYSDATE9 and 23DEC2016"

In the above case, %NRSTR() stops the &SYSDATE9 macro function.

5. %SCAN Function

The %SCAN function returns the nth word in a string.

```
%let var = var1 var2 var3;
%let varName =%scan(&var,1,%str( ));
%put &varName;
```

Result : var1

Concatenation of Macro Variables

Suppose you have 3 macro variables and the third variable is actually a concatenation of the first 2 variables' value.

```
%let x = var;
%let y = 1 ;
%let var1 = 25;
%let z = &&&x&y;
%put &x &y &z;
```

Result : var 1 25

> &z returns 25 because first && resolves to &, &x resolves to var, &y. resolves to 1. So var1 returns 25

Detailed Tutorial : Multiple Ampersand Macro Variables

How to store list of values in a macro variable

Suppose you have a list of names and you want to store them in a macro variable. It can be done by using **SEPARATED BY ' '** keyword in PROC SQL.

```
data temp;
input name $16.;
cards;
Deepanshu Bhalla
Dave Jhonson
Ram Prasad
;
run;
```

```
proc sql;
select name into: myvar separated by ','
from temp;
quit;
```

%put &myvar; returns Deepanshu Bhalla,Dave Jhonson,Ram Prasad

In this case, we have used comma(,) as a delimiter. We can use any other delimiter.

How to debug SAS Macros

There are some system options that can be used to debug SAS Macros:

1. MPRINT

MPRINT translates the macro language to regular SAS language. It displays all the SAS statements of the resolved macro code.

```
options mprint;
%macro test (input =,output=);
proc means data = &input noprint;
var height;
output out = &output mean= ;
run;
%mend;
%test(input=sashelp.heart,output=test);
```

It returns the following message in LOG window :

```
MPRINT(TEST):   proc means data = sashelp.heart noprint;
MPRINT(TEST):   var height;
MPRINT(TEST):   output out = test mean= ;
MPRINT(TEST):   run;
```

2. MLOGIC

It is very helpful when we deal with nested macros (Macro inside another macro). Often we use %DO loops and or %IF-%THEN-%ELSE statements inside the macro code and LOGIC option will display how the macro variable resolved each time in the LOG file as TRUE or FALSE.

```
options mlogic;
options mindelimiter=,;
options minoperator;
%MACRO test();
%DO i = 1 %to 9 ;
%if &i in (1,3,5,7,9) %then %do;
%PUT i = &i - odd;
%END;
%ELSE %DO;
%PUT i = &i - even;
%end;
%end;
```

```
%MEND;
%test();
```

In the log window, see what MLOGIC option produces -

```
MLOGIC(TEST):  %DO loop beginning; index variable I; start value is 1; stop value is 9; by value is 1.
MLOGIC(TEST):  %IF condition &i in (1,3,5,7,9) is TRUE
MLOGIC(TEST):  %PUT i = &i - odd
i = 1 - odd
MLOGIC(TEST):  %DO loop index variable I is now 2; loop will iterate again.
MLOGIC(TEST):  %IF condition &i in (1,3,5,7,9) is FALSE
MLOGIC(TEST):  %PUT i = &i - even
i = 2 - even
```

3. SYMBOLGEN

It writes the results of resolving macro variable references to the SAS log for debugging.

```
options symbolgen;
%macro test (input =,output=);
proc means data = &input noprint;
var height;
output out = &output mean= ;
run;
%mend;
%test(input=sashelp.heart,output=test);
```

SYMBOLGEN: Macro variable INPUT resolves to sashelp.heart
SYMBOLGEN: Macro variable OUTPUT resolves to test

Difference between MPRINT and SYMBOLGEN

See the log generated by these two options. **MPRINT option** prints all the statements within the macro (not just macro variables). Whereas, **SYMBOLGEN option** prints only the results of macro variables.

How to turn off macro debugging options

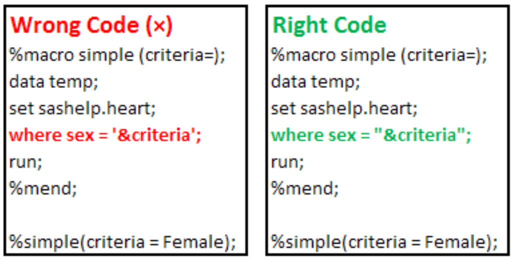
```
options nomprint nomlogic nosymbolgen;
```

All of these options can be turned off together as specified above. Or one of them can be turned off and remaining ones keep turned on.

Important Tips: SAS Macros

Here are some important tips related to SAS Macros :

1. Use **double quotes** to reference macro variables.



Wrong Code (×)

```
%macro simple (criteria=);
data temp;
set sashelp.heart;
where sex = '&criteria';
run;
%mend;

%simple(criteria = Female);
```

Right Code

```
%macro simple (criteria=);
data temp;
set sashelp.heart;
where sex = "&criteria";
run;
%mend;

%simple(criteria = Female);
```

SAS Macros Tutorial

2. The quotes are not needed in %PUT.

```
%put NOTE : system data is &dt.;
```

3. Use **Proc PRINTTO** for saving log in an external text file.

```
proc printto log="C:\Users\Deepanshu\Downloads\LOG2.txt"
new;
run;
```

4. Clear **LOG** and **OUTPUT** Window

```
DM "Log" clear continue;
DM "Output" clear continue;
```

How to call SAS Macro from External Location

There are two main ways to call a SAS macro from an external location.

1. % INCLUDE

```
%include "C:\Users\Deepanshu\Downloads\test.sas";
%test;
```

Suppose your macro is stored in a location and you need to call it from the location. You can accomplish this task using %Include.

2. Autocall Macro Facility and Stored Compiled Macro

See the detailed explanation of these two methods

Practice Question - Try it Yourself

Calculate distinct number of categories (levels) in variable **make** in dataset **sashelp.cars** and store it in a macro variable and later multiply the macro variable by 2.5. *Answer it in the comment box below.*

Related Posts

- SAS PROC PRINT: Learn with Examples
- How to Raise a Number to a Power in SAS
- How to Filter Data in SAS
- SAS Standard Deviation: Learn with Examples
- Top Applications of SAS in Different Domains

SAS Tutorials : Top 100 SAS Tutorials

Spread the Word!

Share in Share Tweet

Deepanshu founded ListenData with a simple objective - Make analytics easy to understand and follow. He has over 10 years of experience in data science. During his tenure, he worked with global clients in various domains like Banking, Insurance, Private Equity, Telecom and HR.

While I love having friends who agree, I only learn from those who don't

Let's Get Connected

Email in LinkedIn

43 Responses to "SAS Macros Made Easy"

Rajesh December 28, 2015 at 9:08 AM

Nice Explanation.Thanks!

Reply Delete

Unknown December 28, 2015 at 8:13 PM

Excellent

Reply Delete

Unknown January 13, 2016 at 7:57 AM

Nice one..can you please give us some practical examples on macros...
Thanks in advance

Reply Delete

Unknown January 13, 2016 at 7:58 AM

Nice one..can you please give us some practical examples on macros...
Thanks in advance

Reply Delete

Vency February 16, 2016 at 3:29 PM

Good Explanation

Reply Delete

Unknown February 29, 2016 at 5:35 AM

Hi guys,

I have a dataset :

```
data imptbase.data_manipulation;
infile datalines;
input filename $ macroname $ sequence;
datalines;
area inserted 1
area deleted 2
area dropped 3
country inserted 1
country deleted 2
country sorted 3
;
run;
```

My intension is to create a macro variable called MAC1 to MAC&mx1 that will hold the macro names. Eg.for filename AREA mac1 will hold value as "Inserted", mac2 will hold "deleted" and so on.

NOTE : &mx1 will be clear from below given code.

```
%macro abc;
proc sql;
select count(distinct filename)into :ank from imptbase.data_manipulation;
%let ank1 = %eval(&ank);
quit;
proc sql;
```

```
%do i = 1 %to &ank1;

select distinct(filename) into :file1 - :file&ank1 from imptbase.data_manipulation;
select max(sequence) into :mx from imptbase.data_manipulation where filename = "&&file&i";

%let mx1 = %eval(&mx);

select distinct(macroname) into :mac1 - :mac&mx1 from imptbase.data_manipulation where filename = "&&file&i";

%put &file1 &file2 &mx &mac1 &mac2 &mac3;

%do j = 1 %to &mx1;

%&&mac&j;

%end;

%end;
quit;
%mend;
```

Desired value of mac1 = inserted, mac2 = deleted, mac3 = dropped.
But it is getting displayed in log as mac1 = deleted, mac2 = dropped, mac3 = inserted.

problem with this code is value of macroname are getting sorted alphbetically(dont know why?)

Can anybody plz help to get the desired output.

Thanks in advance.

Reply Delete

Replies

Anvesh Venishetty October 1, 2019 at 2:35 AM

When you are creating macros mac1 to mac&mx1 you need to sort based on sequence so that you will get the required output.
please replace the below code after creating the &mx1 macro variable

```
select distinct(macroname) into :mac1 - :mac&mx1 from imptbase.data_manipulation where filename = "&&file&i" order by sequence;
```

Delete

Reply

Anonymous March 11, 2016 at 1:21 AM

Hi Guys
Could someone please tell me that a SAS Macro use PDV or not.
If yes then in what circumstances because how much i know Macro facility does not have PDV.

Deepanshu could you please add a page about this in SAS Macro section. If you could then i will be very much thankfull to you.

Reply Delete

Replies

nikhil September 21, 2016 at 12:15 AM

Hi,
I guess you are confusing yourself by comparing SAS PDV with SAS MACRO Processing, both are different.

SAS PDV get created while data step processing while SAS macro gets processed by SAS macro processor.

Delete

Reply

Jeshwika November 15, 2016 at 2:04 AM

Nice tutorial. Heowever, you mentioned %LET. Does it work?
Thanks,
Jeshwika

Reply Delete

Replies

Deepanshu Bhalla December 23, 2016 at 8:37 AM

It works. Please let me know if you face any issue(s) related to it.

Delete

Anonymous January 13, 2017 at 2:18 AM

Plz, can u give explanation about autocall macros.?

Delete

Reply

Arun February 21, 2017 at 5:38 AM

Hi Deepanshu,

There is one mmore I know iss %window statement

Reply Delete

Anonymous March 21, 2017 at 5:04 PM

I wish I had found this blog earlier! It's great. :)

Reply Delete

Anonymous March 25, 2017 at 1:45 AM

Answer to exercise:

```
proc sql;
select count(distinct make) into :mk from sashelp.cars;
quit;
```

```
%put &mk.;
%put %sysevalf(&mk.*2.5);
```

Reply Delete

Swati April 29, 2017 at 5:05 AM

awesome blog man! keep it up!

Reply Delete

Unknown June 13, 2017 at 9:54 PM

HI ALl,
i have below mentioned data.

```
data tab1;
input custid$ @6 saledate mmddyy10. @17 variety $9. @26 quantity ;
format saledate mmddyy10.;
datalines;
240w 02-07-2003 ginger 120
240w 02-07-2003 protea 180
356w 02-08-2003 heliconia 60
356w 02-08-2003 anthurium 300
188r 02-11-2003 ginger 24
188r 02-11-2003 anthurium 24
240w 02-12-2003 heliconia 48
240w 02-12-2003 protea 48
356w 02-12-2003 'ginger' 240
;
run;
```

i made below macro for same.

```
%macro new (flowertype=);
proc print data=tab1;
where variety ="&flowertype" & custid="&id";
run;
%mend;
%new (flowertype=ginger);
```

Can anyone tell me how to make macro of dates. Like i make flowertype and its ginger when i will mention any flowertype in macro then it will give me print. Same way i want to make macro of Date. if i will mention particular date then macro will show me the result of that date only.

Reply Delete

Replies

Arjun Singh Bisht August 13, 2019 at 6:01 AM

By simply putting d after quotation mark in the where clause.
E.g. where saledate = “&date”d ;

Delete

Reply

Pavani K July 19, 2017 at 5:26 PM

This comment has been removed by the author.

Reply Delete

Pavani K July 19, 2017 at 9:35 PM

This comment has been removed by the author.

Reply Delete

Pavani K July 19, 2017 at 9:37 PM

Hello,

I Have similar data like given in the following dataset. What i am trying to do is subset the data for each value of location.

```
Data Test;
Input Name$ Location$;
cards;
Jhon
Mary
Ann
Alfred
Bob
Joe
;
run;
```

I have written the following macro.

```
%macro Subset (Dset= ,var= );
data &Dset;
set &Dset;
where Location=&var;
run;
%mend;
```

Calling macro:

```
%Subset(Dset=Test,var="");
```

I understand that it will not work because of the spl char and to mask or remove the meaning of them we have to use %STR function.

But I am not understanding where to use that %STR function.

Can anyone help me please.

Thank you so much.

Reply Delete

Pavani K July 19, 2017 at 9:38 PM

This comment has been removed by the author.

Reply Delete

Pavani K July 19, 2017 at 9:40 PM

Hello,

I Have similar data like given in the following dataset. What i am trying to do is subset the data for each value of location.

```
Data Test;
```

```
Input Name$ Location$;
cards;
Data Test;
Input Name$ Location$;
cards;
Jhon office
Mary parlour
Ann Bank
Alfred office
Bob Bank
Joe Parlour
;
run;
```

The Location values are enclosed in< > like ex,

I have written the following macro.

```
%macro Subset (Dset= ,var= );
data &Dset;
set &Dset;
where Location=&var;
run;
%mend;
```

Calling macro:

```
%Subset(Dset=Test,var="");
```

I understand that it will not work because of the spl char and to mask or remove the meaning of them we have to use %STR function.

But I am not understanding where to use that %STR function.

Can anyone help me please.

Thank you so much.

Reply Delete

Replies

Karan August 20, 2017 at 9:08 PM

YOU CAN ALSO USE THE FOLLOWING CODE :

```
PROC SQL NOPRINT;
SELECT DISTINCT LOCATION INTO: LOC SEPARATED BY ' '
FROM TEST;
QUIT;

DATA &LOC;
SET TEST;
IF Location="%SCAN(&LOC,1,' ')" THEN OUTPUT %SCAN(&LOC,1,' ');
ELSE IF Location ="%SCAN(&LOC,2,' ')" THEN OUTPUT %SCAN(&LOC,2,' ');
ELSE IF Location="%SCAN(&LOC,3,' ')" THEN OUTPUT %SCAN(&LOC,3,' ');
RUN;
```

Delete

Karan August 20, 2017 at 9:11 PM

the code which you have written must be like this :

```
%macro Subset (Dset1=,Dset= ,var= );
data &Dset1;
set &Dset;
where Location="&var";
run;
%mend;


%Subset(Dset1=BANK1,Dset=Test,var=Bank);
```

Delete

Reply

Anonymous September 14, 2017 at 5:39 AM

The count is 95.

Reply Delete

Anonymous October 15, 2017 at 4:39 AM

```
data abcd;
input Acct CU_IND TRNS_CNT TRNS_AMT SUM;
cards;
1 0 0 0 0
1 1 5 50 55
1 1 5 50 55
1 0 0 0 0
1 1 2 100 102
1 1 2 100 102
2 0 0 0 0
2 0 0 0 0
2 0 0 0 0
2 1 1 250 251
2 1 1 250 251
2 1 1 250 251
3 1 2 250 252
3 1 2 200 202
3 0 0 0 0
3 0 0 0 0
3 1 1 850 851
3 1 1 850 851
3 1 1 850 851
4 1 1 90 91
4 1 1 90 91
;
run;
required output is
1 1 5 50 55
1 1 2 100 102
2 1 1 250 251
3 1 1 850 851
4 1 1 90 91
```

Reply Delete

Replies

Unknown May 21, 2019 at 2:58 PM

```
proc sort data=abcd nodup out=pp;
by _all_;
run;
```

Delete

Reply

Unknown January 3, 2018 at 9:13 PM

It is very helpful. Thank you so much!

```
/* 1. create macro test */

%macro test (input=, ivar=, out=);
proc sort data=&input out=sort;
by &ivar;
run;

data c;
set sort;
by &ivar;
if first.&ivar;
run;

proc means data=c;
output out=&out N=num;
run;
%mend;

/* 2. call macro test */

%test(input=sashelp.cars, ivar=make, out=m);

/* 3. proc sql */

proc sql;
select num into: var1 separated by ','
from m;
quit;

%put &var1;

/* It shows:38 */

%let x=&var1*2.5;

%let y=%sysevalf(&x);

%put &y;

/* It shows: 95 */
/* End */
```

Reply Delete

Unknown April 8, 2018 at 2:23 PM

i have a question,

suppose we have a data set consisting of the monthly income of a group of people over the years and we have to extract the income data of every third month of the people. how it could be done using macros?

Reply Delete

Unknown August 4, 2018 at 11:07 AM

Good Job. Very useful blog :)

Reply Delete

Unknown August 15, 2018 at 6:14 AM

```
select distinct make into :k1 from sashelp.cars;
%put &k1*2.5;
```

Reply Delete

Replies

Unknown August 15, 2018 at 6:17 AM

```
proc sql;
select distinct make into :k1 from sashelp.cars;
quit;
%put &k1*2.5;
```

Delete

Reply

Unknown January 14, 2019 at 7:21 AM

hi..i'am unable to run macros in SAS studio..can any one help??

Reply Delete

Anonymous May 2, 2019 at 11:10 PM

"Calculate distinct number of categories (levels) in variable make in dataset sashelp.cars and store it in a macro variable and later multiply the macro variable by 2.5" please check code below

```
proc sql noprint;
select count(distinct make) into:makecnt from sashelp.cars;
quit;

%put %sysevalf(&makecnt*2.5);
```

Reply Delete

karan July 30, 2019 at 9:26 AM

what rubbish !! program for iterative %do not working

Reply Delete

Bharat Matakijai November 7, 2019 at 11:41 PM

you're doing such a great job Deepanshu. your whole material is better than any other source that is available online.This blog helped me get a job. Forever greatfull!! Thank you

Reply Delete

Unknown November 22, 2020 at 7:10 AM

```
proc sql;
Select count(distinct make) into: make
From sashelp.cars;
quit;
run;
option symbolgen;
%let make2 = %sysevalf(&make * 2.5);
%put &make &make2;
38 95
```

Thank you for detailed article with macro details not found in other resources. Kudos on your site filled with information on SAS, R, and Python. Will be a frequent visitor here.

Vladimir

Reply Delete

Unknown June 28, 2021 at 11:03 PM

Hi Everyone can anyone help me where from I can learn SAS Base/Advanced programming?
I need to work on macros as well aslo...it is required in current work.

If anyone have pdf of sas advance book please share with me.

Reply Delete

Anonymous July 20, 2021 at 11:37 PM

```
PROC sql;
create table made as
select distinct(count(*)) as make
from sashelp.cars;
quit;

data _null_;
set made;
call symput("make", make);
run;

%put &make;

data new;
newmake=%sysevalf(&make*2.5);
run;
```

Reply Delete

pankaj ray January 20, 2023 at 11:53 AM

@Deepanshu Do you know Amit Kumar/Sir?

Reply Delete

Replies

Deepanshu Bhalla January 20, 2023 at 12:05 PM

No

Delete

Reply

Pavankalyangoud March 14, 2023 at 3:21 AM

```
%MACRO ABC;
PROC SQL;
CREATE TABLE PAV AS
SELECT DISTINCT (MAKE)FROM SASHELP.CARS
QUIT;
%MEND;

%ABC;

%LET A=%SYSEVALF(&ABC*&2.5);
```

```
ERROR: A character operand was found in the %EVAL function or %IF condition where a numeric operand
is required. The condition was: &ABC*&2.5
```

THERE ARE CHARACTER VALUES IN CARS DATASET SO CALCULATIONS R NOT POSSIBLE

Reply Delete

To leave a comment, click the button below to sign in with Google.

SIGN IN WITH GOOGLE

← PREV NEXT →