

SAS Advanced Programmer

SAP2 Macro

SAP205 Storing Macros, Generating Data-Dependent Code, Validating Parameters and Documenting Macros

```
%let path=~ / EMC1V2;
```

```
libname mc1 "&path/data";
```

2011 Storms

The MEANS Procedure

Variable	N	Minimum	Mean	Maximum
MaxWindMPH	80	6	75	155
MinPressure	80	920	975	1006

2014 Storms

The MEANS Procedure

Variable	N	Minimum	Mean	Maximum
MaxWindMPH	85	29	84	161
MinPressure	85	900	968	1012



```
%macro printtable(dsn=sashelp.cars,obs=5)
    /des='Print a table. Parms: DSN= OBS=';
proc print data=&dsn(obs=&obs);
run;
%mend printtable;
```

```

*****;

* Activity 5.01                                *;

* 1) This program creates a macro named PrintTable in *;
* the default work.sasmacr catalog. PROC CATALOG *;
* lists the contents of work.sasmacr. *;
* 2) Run the program. The report includes all macros in *;
* work.sasmacr, including PRINTTABLE and its *;
* description. *;
* 3) Uncomment the second CONTENTS statement. This *;
* statement will create a table including the macros *;
* in work.sasmacr. The WHERE= data set option is *;
* used to filter the results. *;
* 4) Run the program. How many macros include TABLE in *;
* the name? *;

```

```

*****;

```

/*Activity 5.01

Open m105a01.sas from the activities folder. This program creates a macro named PrintTable in the default macro catalog and uses PROC CATALOG to list the contents of the catalog.

Run the PRINTTABLE macro definition and verify that it compiles without errors.

Copy the following SQL step and paste it after the %PRINTTABLE macro definition.

```

proc sql;
select *
  from dictionary.catalogs
  where objname='PRINTTABLE';
quit;

```

Run the step and view the report. Note the value of Library Name and Member Name.

This is the default macro catalog in your environment.

Review the PROC CATALOG step. If necessary, change work.sasmacr in the CATALOG= option to reference the default macro catalog you identified in Step 1. Run the step.

This report includes all macros in the default macro catalog, including PRINTTABLE and its description.

Run the program and examine the output table. This report includes all macros in the default macro catalog,

including PRINTTABLE and its description.

Uncomment the second CONTENTS statement. This statement creates a table named work.macrolist.

The table includes the macros in the catalog specified in the CATALOG= option.

The WHERE= data set option filters the results, including only macros with the string TABLE in their name.

Note: The single quotation marks prevent macro resolution, so the % is treated as a wildcard for the LIKE operator

rather than a macro trigger.

How many macros are included in the output table?

Note: Type a numeric value for your answer.

*/

```
%macro printtable(dsn=sashelp.cars,obs=5) /des='Print a table. Parns: DSN= OBS=';
```

```
    proc print data=&dsn(obs=&obs);
```

```
    run;
```

```
%mend printtable;
```

```
proc sql;
```

```
select *
```

```
    from dictionary.catalogs
```

```
    where objname='PRINTTABLE';
```

```
quit;
```

Library Name	Member Name	Member Type	Object Name	Object Type	Object Description	Date Created	Date Modified	Object Alias	Library Concatenation Level
WORK	SASMAC1	CATALOG	PRINTTABLE	MACRO	Print a table. Parns: DSN= OBS=	06JUN21:23:32:39	06JUN21:23:32:39		0

```
proc catalog catalog=work.sasmac1;

  contents;

  contents out=macrolist(where=(name like '%TABLE%'));

run;
```

Table: WORK.MACROLIST View: Column names Filter: (none)

Columns

LIBNAME	MEMNAME	NAME	TYPE	DESC	DATE	CRDATE
1 WORK	SASMAC1	PRINTTABLE	MACRO	Print a table. Params: DSN= OBS=	06/06/21	06/06/2021 23:37
2 WORK	SASMAC1	SEARCH_TABLES	MACRO		06/07/21	06/07/2021 06:36
3 WORK	SASMAC1	WEB_DROP_TABLE	MACRO		06/07/21	06/07/2021 06:36
4 WORK	SASMAC1	WEB_OPEN_TABLE	MACRO		06/07/21	06/07/2021 06:36

*****;

- * Activity 5.02 *;
- * 1) Highlight the PROC CATALOG step and run the *;
- * selected code. Notice in the report that the *;
- * DATATYP macro program is not listed. *;
- * 2) Run the %LET and %PUT statements that call the *;
- * %Datatyp macro. View the log and confirm the macro *;
- * runs successfully. *;
- * 3) Highlight the PROC CATALOG step again and run the *;
- * selected code. Is DATATYP included in the report? *;

*****;

/*Activity 5.02

Open m105a02.sas from the activities folder and perform the following tasks:

Review the PROC CATALOG step. If necessary, update the CATALOG= option to reference the default macro catalog in your environment.

Highlight the PROC CATALOG step and run the selected code. Notice in the report that the %Datatyp macro is not listed.

Run the %LET and %PUT statements that call the %Datatyp macro. View the log and confirm the macro runs successfully.

Highlight the PROC CATALOG step again and run the selected code.

Is %Datatyp included in the report?

```
*/
```

```
proc catalog catalog=work.sasmac1;
```

```
    contents;
```

```
run;
```

```
%let val1=ABC;
```

```
%let val2=123;
```

```
%put The data type of Val1 is %datatyp(&val1);
```

```
%put The data type of Val2 is %datatyp(&val2);
```

```
73          %let val1=ABC;
```

```
74          %let val2=123;
```

```
75
```

```
76          %put The data type of Val1 is %datatyp(&val1);
```

```
The data type of Val1 is CHAR
```

```
77          %put The data type of Val2 is %datatyp(&val2);
```

```
The data type of Val2 is NUMERIC
```

Contents of Catalog WORK.SASMAC1					
#	Name	Type	Create Date	Modified Date	Description
1	CLEANUP_SASSTUDIO_SEARCH	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
2	DATATYP	MACRO	06/06/2021 23:47:13	06/06/2021 23:47:13	
3	DO_SEARCH	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
4	MERGE_DATA	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
5	MIGRATE_PREFERENCES	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
6	PRINTTABLE	MACRO	06/06/2021 23:37:11	06/06/2021 23:37:11	Print a table. Parms: DSN= OBS=
7	QLEFT	MACRO	06/06/2021 23:47:13	06/06/2021 23:47:13	
8	QTRIM	MACRO	06/06/2021 23:47:13	06/06/2021 23:47:13	
9	SASSTUDIO_GET_ZOS_DS_INFO	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
10	SEARCH_COLUMNS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
11	SEARCH_FILES_AND_FOLDERS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
12	SEARCH_FTP_SHORTCUT	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
13	SEARCH_LIBREFS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
14	SEARCH_MVS_DATASET	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
15	SEARCH_MVS_DATASETS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
16	SEARCH_TABLES	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
17	SGDESIGN	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
18	SHOW_ZOS_DATASET_ATTRIBUTES	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
19	STUDIOCOPY	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
20	STUDIODELETE	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
21	STUDIO_CAS_START	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
22	VERIFY	MACRO	06/06/2021 23:47:13	06/06/2021 23:47:13	
23	WEB_DROP_TABLE	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
24	WEB_LIST_CATALOGS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
25	WEB_LIST_ENTRIES	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
26	WEB_OPEN_FILE	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
27	WEB_OPEN_TABLE	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
28	WEB_OPEN_URL	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
29	WEB_REPLAY_GRSEG	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	

/******

* Storing Macros: Demo *

*****/

/* Section 1 */

data _null_;

set mc1.country_codes;

call symputx(CountryCode, CountryName);

```
run;
```

```
proc sql;
```

```
select *
```

```
from dictionary.macros
```

```
order by Scope, Name;
```

```
quit;
```

Macro Scope	Macro Variable Name	Offset into Macro Variable	Macro Variable Value
AUTOMATIC	AFDSID	0	0
AUTOMATIC	AFDSNAME	0	
AUTOMATIC	AFLIB	0	
AUTOMATIC	AFSTR1	0	
AUTOMATIC	AFSTR2	0	
AUTOMATIC	FSPBDV	0	
AUTOMATIC	SYSADDRBITS	0	64
AUTOMATIC	SYSBUFFR	0	
AUTOMATIC	SYSCC	0	0
AUTOMATIC	SYSCHARWIDTH	0	1
AUTOMATIC	SYSCMD	0	
AUTOMATIC	SYSDATASTEP	0	
AUTOMATIC	SYSDATE	0	07JUN21
AUTOMATIC	SYSDATE9	0	07JUN2021
AUTOMATIC	SYSDAY	0	Monday

GLOBAL	SYSUSERNAME	0	u58304328
GLOBAL	SYS_SQL_IP_ALL	0	-1
GLOBAL	SYS_SQL_IP_STMT	0	
GLOBAL	TR	0	Turkey
GLOBAL	US	0	United States
GLOBAL	USERDIR	0	homeu58304328
GLOBAL	VAL1	0	ABC
GLOBAL	VAL2	0	123
GLOBAL	ZA	0	South Africa
GLOBAL	_BASEURL	0	https://odamidusw2.oda.sas.comSASStudio
GLOBAL	_CLIENTAPP	0	'SAS Studio'
GLOBAL	_CLIENTAPPABREV	0	Studio
GLOBAL	_CLIENTAPPVERSION	0	3.8
GLOBAL	_CLIENTMACHINE	0	097090129229.BIZ.SPECTRUM.COM
GLOBAL	_CLIENTMODE	0	wip
GLOBAL	_CLIENTUSERID	0	u58304328
GLOBAL	_CLIENTUSERNAME	0	u58304328
GLOBAL	_CLIENTVERSION	0	3.8
GLOBAL	_EXECENV	0	SASStudio
GLOBAL	_MACRO_FOUND	0	0
GLOBAL	_METAUSER	0	u58304328
GLOBAL	_SASHOSTNAME	0	odaws03usw2
GLOBAL	_SASPROGRAMFILE	0	homeu58304328EMC1V2demosm105d02.sas
GLOBAL	_SASPROGRAMFILEHOST	0	odaws03usw2
GLOBAL	_SASSERVERNAME	0	SASApp
GLOBAL	_SASWORKINGDIR	0	/pbr/biconfig/940/Lev1/SASApp
GLOBAL	_SASWSTEMP_	0	homeu58304328.sasstudio.images36308de61898434a80a9c17a87dfb7a1
GLOBAL	_SASWS_	0	homeu58304328

```
/* Section 2 */
```

```

proc sql;
select *
  from dictionary.macros
 where Scope='GLOBAL' and
        Name not like 'SYS%' and
        Name not like 'SQL%' and
        Name not like 'SASWORK%' and
        Name not like 'GRAPH%' and
        Name not like 'STUDIO%' and
        Name not like 'OLD%' and
        Name not like '^_%' escape '^' and
        Name ne 'CLIENTMACHINE' and
        Name ne 'USERDIR';
quit;

```

Macro Scope	Macro Variable Name	Offset into Macro Variable	Macro Variable Value
GLOBAL	AT	0	Austria
GLOBAL	AU	0	Australia
GLOBAL	BE	0	Belgium
GLOBAL	CA	0	Canada

GLOBAL	PATH	0	~/EMC1V2
GLOBAL	PL	0	Poland
GLOBAL	PT	0	Portugal
GLOBAL	SE	0	Sweden
GLOBAL	SI	0	Slovenia
GLOBAL	TR	0	Turkey
GLOBAL	US	0	United States
GLOBAL	VAL1	0	ABC
GLOBAL	VAL2	0	123
GLOBAL	ZA	0	South Africa

```
/* Section 2b */
```

```
proc sql;
```


select Name

into :Vars separated by ' '

from dictionary.macros

where Scope='GLOBAL' and

Name not like 'SYS%' and

Name not like 'SQL%' and

Name not like 'SASWORK%' and

Name not like 'GRAPH%' and

Name not like 'STUDIO%' and

Name not like 'OLD%' and

Name not like '^_%' escape '^' and

Name ne 'CLIENTMACHINE' and

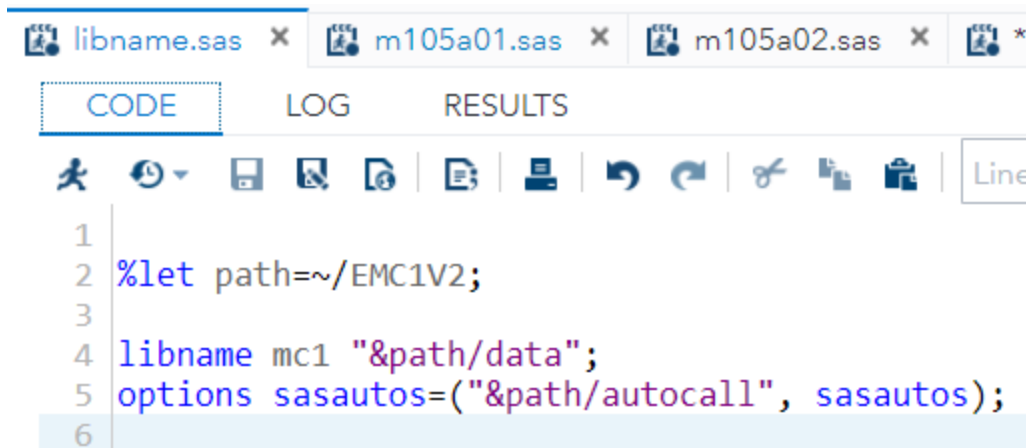
Name ne 'USERDIR';

quit;

%put &=vars;

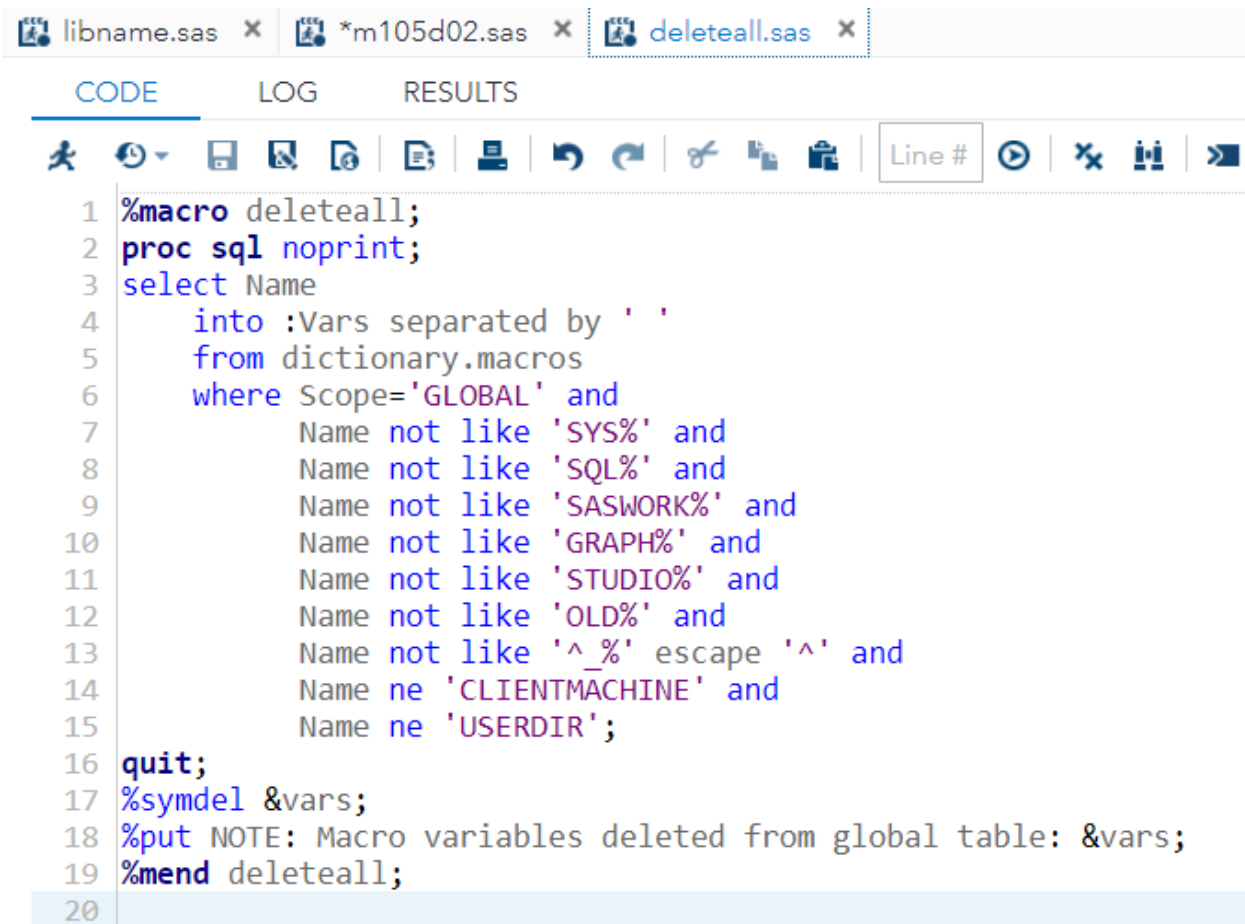
90 %put &=vars;

VAR=AT AU BE CA CH CI DE DK EG ES FI FR GB GR HR IE IL IN IT LT LU
MZ NL NO NZ PATH PL PT SE SI TR US VAL1 VAL2 ZA



The screenshot shows the SAS Studio interface. At the top, there are four tabs: 'libname.sas', 'm105a01.sas', 'm105a02.sas', and an unnamed tab. Below the tabs are three buttons: 'CODE', 'LOG', and 'RESULTS'. The 'CODE' button is selected. Below the buttons is a toolbar with various icons for file operations and editing. The main area is a code editor showing the following SAS code:

```
1  
2 %let path=~ /EMC1V2;  
3  
4 libname mc1 "&path/data";  
5 options sasautos("&path/autocall", sasautos);  
6
```



```
libname.sas x *m105d02.sas x deleteall.sas x
CODE LOG RESULTS
Line #
1 %macro deleteall;
2 proc sql noprint;
3 select Name
4     into :Vars separated by ' '
5     from dictionary.macros
6     where Scope='GLOBAL' and
7           Name not like 'SYS%' and
8           Name not like 'SQL%' and
9           Name not like 'SASWORK%' and
10          Name not like 'GRAPH%' and
11          Name not like 'STUDIO%' and
12          Name not like 'OLD%' and
13          Name not like '^_%' escape '^' and
14          Name ne 'CLIENTMACHINE' and
15          Name ne 'USERDIR';
16 quit;
17 %symdel &vars;
18 %put NOTE: Macro variables deleted from global table: &vars;
19 %mend deleteall;
20
```

/* Section 3 */

```
proc catalog cat=work.sasmac1;
```

```
    delete deleteall.macro;
```

```
run;
```

NOTE: Deleting entry DELETEALL.MACRO in catalog WORK.SASMAC1.

/* Section 4 */

```
data _null_;
```

```
    set mc1.country_codes;
```

```
    call symputx(CountryCode, CountryName);
```

```
run;
```

%deleteall

proc sql;

select *

from dictionary.macros

order by Scope, Name;

quit;

NOTE: Macro variables deleted from global table: &vars

Macro Scope	Macro Variable Name	Offset into Macro Variable	Macro Variable Value
GLOBAL	SQLRC	0	0
GLOBAL	SQLXOBS	0	0
GLOBAL	SQLXOPENERRS	0	0
GLOBAL	STUDIODIR	0	C:\home\user58304328\SASStudio
GLOBAL	STUDIODIRNAME	0	.sasstudio
GLOBAL	STUDIOPARENTDIR	0	C:\home\user58304328
GLOBAL	SYSCASINIT	0	0
GLOBAL	SYSSTREAMINGLOG	0	true
GLOBAL	SYSUSERNAME	0	user58304328
GLOBAL	SYS_SQL_IP_ALL	0	-1
GLOBAL	SYS_SQL_IP_STMT	0	
GLOBAL	USERDIR	0	C:\home\user58304328
GLOBAL	_BASEURL	0	https://odamid.usw2.oda.sas.com/SASStudio
GLOBAL	_CLIENTAPP	0	'SAS Studio'
GLOBAL	_CLIENTAPPABREV	0	Studio
GLOBAL	_CLIENTAPPVERSION	0	3.8
GLOBAL	_CLIENTMACHINE	0	097090129229.BIZ.SPECTRUM.COM
GLOBAL	_CLIENTMODE	0	wip
GLOBAL	_CLIENTUSERID	0	user58304328
GLOBAL	_CLIENTUSERNAME	0	user58304328
GLOBAL	_CLIENTVERSION	0	3.8
GLOBAL	_EXECENV	0	SASStudio
GLOBAL	_MACRO_FOUND	0	0
GLOBAL	_METAUSER	0	user58304328
GLOBAL	_SASHOSTNAME	0	odaws03.usw2
GLOBAL	_SASPROGRAMFILE	0	C:\home\user58304328\EMC1V2\myscripts\m105d02.sas
GLOBAL	_SASPROGRAMFILEHOST	0	odaws03.usw2
GLOBAL	_SASSERVERNAME	0	SASApp
GLOBAL	_SASWORKINGDIR	0	/pbr/bicconfig/940/Lev1/SASApp
GLOBAL	_SASWSTEMP_	0	C:\home\user58304328\SASStudio\images\5092948c62c41c689281501ed129f9e
GLOBAL	_SASWS_	0	C:\home\user58304328

*****;

* Activity 5.03 *;

* 1) Run the PROC CATALOG step and confirm that PROPCASE *;

* does not exist in work.sasmacr. *;

* 2) Create a new program and enter the following code. *;

```

* The %Propcase macro applies the PROPCASE function  *;
* to the text parameter. Do not run the program, but  *;
* save it as propcase.sas in the autocall folder.  *;
* %macro propcase(text);                               *;
*   %sysfunc(propcase(&text))                          *;
* %mend propcase;                                     *;
* 3) Open the libname.sas program. Add an OPTIONS      *;
* statement and the SASAUTOS= option to include the  *;
* autocall folder as an additional autocall library.  *;
* Run the program and save it.                        *;
* 4) Return to the m105a03.sas program and uncomment the *;
* %PUT statement. Run the program. Confirm that the  *;
* %Propcase macro executes successfully and verify  *;
* that it has been added to the work.sasmacr catalog. *;
*****;

```

/*Activity 5.03

Open m105a03.sas from the activities folder.

Review the PROC CATALOG step. If necessary, update the CATALOG= option to reference the default macro catalog

in your environment.

Run the PROC CATALOG step and confirm that PROPCASE does not exist in work.sasmacr.

Create a new program and enter the following code:

```

%macro propcase(text);
  %sysfunc(propcase(&text))
%mend propcase;

```

This PropCase macro applies the PROPCASE function to the text parameter.

Do not run the program, but save it as propcase.sas in the autocall folder.

Open the libname.sas program. Add an OPTIONS statement to set the SASAUTOS= option to include our &path/autocall folder

as an additional autocall library.

```
options sasautos=("&path/autocall", sasautos);
```

Run the modified libname.sas program and save it.

Return to the m105a03.sas program and uncomment the %PUT statement.

Run the program.

Did %PropCase execute successfully, and has it been added to the default catalog?

Note: Type Yes or No for your answer.

```
*/
```

```
*%put Testing the PROPCASE macro: %propcase(does PROPCASE Work?);
```

```
proc catalog cat=work.sasmac1;
```

```
  contents;
```

```
run;
```

Contents of Catalog WORK.SASMAC1					
#	Name	Type	Create Date	Modified Date	Description
1	CLEANUP_SASSTUDIO_SEARCH	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
2	DATATYP	MACRO	06/06/2021 23:47:13	06/06/2021 23:47:13	
3	DELETEALL	MACRO	06/07/2021 00:24:06	06/07/2021 00:24:06	
4	DO_SEARCH	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
5	MERGE_DATA	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
6	MIGRATE_PREFERENCES	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
7	PRINTTABLE	MACRO	06/06/2021 23:37:11	06/06/2021 23:37:11	Print a table. Parns: DSN= OBS=
8	QLEFT	MACRO	06/06/2021 23:47:13	06/06/2021 23:47:13	
9	QTRIM	MACRO	06/06/2021 23:47:13	06/06/2021 23:47:13	
10	SASSTUDIO_GET_ZOS_DS_INFO	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
11	SEARCH_COLUMNS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
12	SEARCH_FILES_AND_FOLDERS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
13	SEARCH_FTP_SHORTCUT	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
14	SEARCH_LIBREFS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
15	SEARCH_MVS_DATASET	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
16	SEARCH_MVS_DATASETS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
17	SEARCH_TABLES	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
18	SGDESIGN	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
19	SHOW_ZOS_DATASET_ATTRIBUTES	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
20	STUDIOCOPY	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
21	STUDIODELETE	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
22	STUDIO_CAS_START	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
23	VERIFY	MACRO	06/06/2021 23:47:13	06/06/2021 23:47:13	
24	WEB_DROP_TABLE	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
25	WEB_LIST_CATALOGS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
26	WEB_LIST_ENTRIES	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
27	WEB_OPEN_FILE	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
28	WEB_OPEN_TABLE	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
29	WEB_OPEN_URL	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
30	WEB_REPLAY_GRSEG	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
31	_COMPILED_MACRO_EXISTS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
32	EG_CONDITIONAL_DROPDS	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	

libname.sas ×
*m105a03.sas ×
propcase.sas ×

CODE
LOG
RESULTS

```

1 %macro propcase(text);
2     %sysfunc(propcase(&text))
3 %mend propcase;
4

```

```

1
2 %let path=~ /EMC1V2;
3
4 libname mc1 "&path/data";
5 options sasautos=("&path/autocall", sasautos);
6

```

%put Testing the PROPCASE macro: %propcase(does PROPCASE Work?);

```
proc catalog cat=work.sasmac1;
```

```
  contents;
```

```
run;
```

```
116          %put Testing the PROPCASE macro: %propcase(does PROPCASE
Work?);
```

Testing the PROPCASE macro: Does Propcase Work?

Contents of Catalog WORK.SASMAC1					
#	Name	Type	Create Date	Modified Date	Description
1	CLEANUP_SASSTUDIO_SEARCH	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
2	DATATYP	MACRO	06/06/2021 23:47:13	06/06/2021 23:47:13	
3	DELETEALL	MACRO	06/07/2021 00:24:06	06/07/2021 00:24:06	
4	DO_SEARCH	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
5	MERGE_DATA	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
6	MIGRATE_PREFERENCES	MACRO	06/07/2021 06:36:45	06/07/2021 06:36:45	
7	PRINTTABLE	MACRO	06/06/2021 23:37:11	06/06/2021 23:37:11	Print a table. Pams: DSN= OBS=
8	PROPCASE	MACRO	06/07/2021 00:36:46	06/07/2021 00:36:46	

```
*****;
```

```
* Activity 5.04
```

```
  *;
```

```
* 1) Notice that the program includes two SELECT    *;
```

```
* statements. Run the program. The first report    *;
```

```
* includes the distinct values of Origin. The second *;
```

```
* report includes a separate WHEN statement for each *;
```

```

* value of Origin. *;
* 2) Change the value of the Col macro variable to Type. *;
* Run the program again and observe the two reports. *;
* 3) What syntax can be used to create macro variables *;
* that store the distinct values as a delimited list? *;
*****

```

```

%let tab=sashelp.cars;

%let col=Origin;

proc sql;

select distinct &col

    from &tab;

select distinct cat('when ("', &col, '"') output ', &col)

    from &tab;

quit;

```

Origin
Asia
Europe
USA

when ("Asia ") output Asia
when ("Europe") output Europe
when ("USA ") output USA

```

%let tab=sashelp.cars;

%let col=Type;

proc sql;

select distinct &col

    into :tablist separated by " "

    from &tab;

```



```

select distinct cat('when ('', &col, '') output ', &col)
        into :whenlist separated by ";";
from &tab;
quit;

```

Type
Hybrid
SUV
Sedan
Sports
Truck
Wagon

when ("Hybrid ") output Hybrid
when ("SUV ") output SUV
when ("Sedan ") output Sedan
when ("Sports ") output Sports
when ("Truck ") output Truck
when ("Wagon ") output Wagon

```

/*****

```

```

* Generating Data-Dependent Code: Demo *

```

```

*****/

```

```

%let tab=mc1.storm_final;

```

```

%let col=Ocean;

```

```

proc sql noprint;

```

```

select distinct &col

```

```

        into :tabList separated by " "

```

```

        from &tab;

```

```

select distinct

```

```

        cat('when ('', &col, '') output ', &col)

```

```

        into :whenList separated by ";";

```

```

    from &tab;
quit;

proc sql;
select name, value
    from dictionary.macros
    where name in ("TABLIST", "WHENLIST");
quit;

```

```
%symdel tab col tablist whenlist;
```

Macro Variable Name	Macro Variable Value
TABLIST	Atlantic Indian Pacific
WHENLIST	when ("Atlantic") output Atlantic;when ("Indian ") output Indian;when ("Pacific ") output Pacific

```

%macro splittable_sql(tab, col);
proc sql noprint;
select distinct &col
    into :tablist separated by " "
    from &tab;
select distinct
    cat('when ("', &col, "') output ', &col)
    into :whenlist separated by ";"
    from &tab;
quit;

```

```

data &tablist;
    set &tab;
    select(&col);
        &whenlist;
    otherwise;

```

```

end;

run;

%mend splittable_sql;

/* Step 4 */

%splittable_sql(sashelp.cars, Type)

```

SAS Studio interface showing the OUTPUT DATA tab for the WORK.HYBRID table. The table has 3 rows and 15 columns. The columns are Make, Model, Type, Origin, DriveTrain, MSRP, Invoice, and EngineSize. The data shows three vehicles: a Honda Civic Hybrid, a Honda Insight, and a Toyota Prius.

Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize
1 Honda	Civic Hybrid 4dr manual (gas/electric)	Hybrid	Asia	Front	\$20,140	\$18,451	1
2 Honda	Insight 2dr (gas/electric)	Hybrid	Asia	Front	\$19,110	\$17,911	
3 Toyota	Prius 4dr (gas/electric)	Hybrid	Asia	Front	\$20,510	\$18,926	1

SAS Studio interface showing the OUTPUT DATA tab for the WORK.SEDAN table. The table has 262 rows and 15 columns. The columns are Make, Model, Type, Origin, DriveTrain, MSRP, Invoice, and EngineSize. The data shows various sedan models from Acura, Audi, and others.

Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize
1 Acura	RSX Type S 2dr	Sedan	Asia	Front	\$23,820	\$21,761	2
2 Acura	TSX 4dr	Sedan	Asia	Front	\$26,990	\$24,647	2.4
3 Acura	TL 4dr	Sedan	Asia	Front	\$33,195	\$30,299	3.2
4 Acura	3.5 RL 4dr	Sedan	Asia	Front	\$43,755	\$39,014	3.5
5 Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	\$46,100	\$41,100	3.5
6 Audi	A4 1.8T 4dr	Sedan	Europe	Front	\$25,940	\$23,508	1.8
7 Audi	A4 1.8T convertible 2dr	Sedan	Europe	Front	\$35,940	\$32,506	1.8
8 Audi	A4 3.0 4dr	Sedan	Europe	Front	\$31,840	\$28,846	3
9 Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	\$33,430	\$30,366	3
10 Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	\$34,480	\$31,388	3
11 Audi	A6 3.0 4dr	Sedan	Europe	Front	\$36,640	\$33,129	3
12 Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	\$39,640	\$35,992	3
13 Audi	A4 3.0 convertible 2dr	Sedan	Europe	Front	\$42,490	\$38,325	3

```

/* Step 5 */

%splittable_sql(mc1.storm_final, Basin)

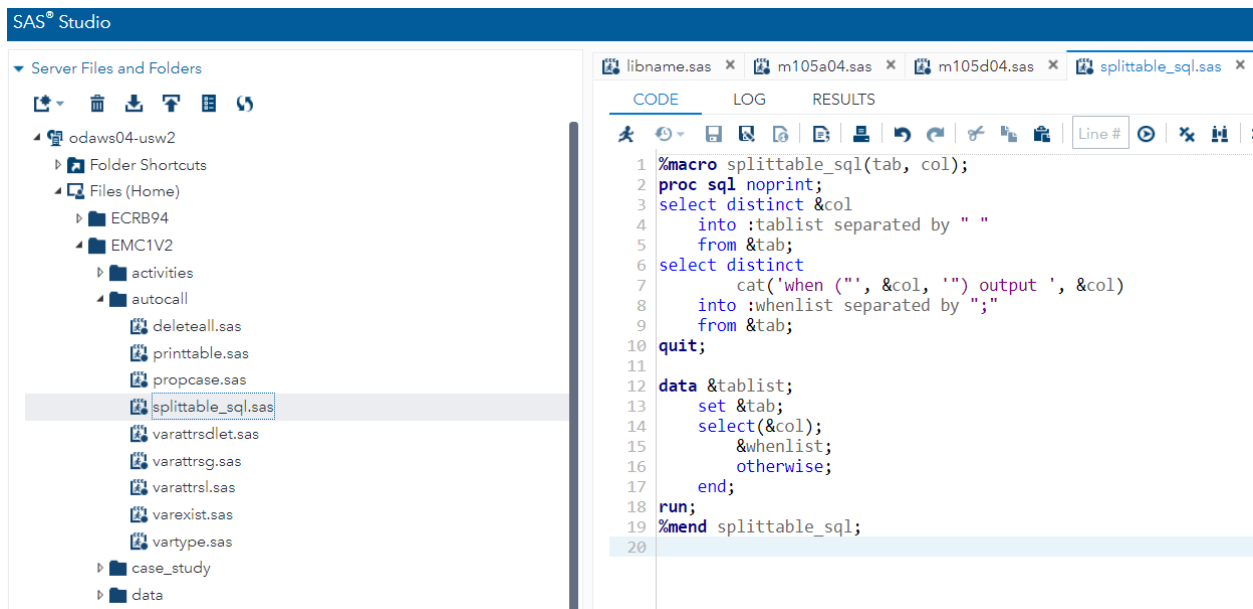
```

SAS Studio interface showing the 'OUTPUT DATA' tab for the 'WORK.KEP' table. The table contains 675 rows and 13 columns. The columns are: Season, Name, Basin, Ocean, StormType, MaxWindMPH, MaxWindKM, and MinPressure. The data is displayed in a grid format.

Season	Name	Basin	Ocean	StormType	MaxWindMPH	MaxWindKM	MinPressure
2017	ADRIAN	EP	Pacific		45	72	1000
2017	BEATRIZ	EP	Pacific		45	72	1000
2017	CALVIN	EP	Pacific		45	72	1000
2017	DORA	EP	Pacific		105	169	974
2017	EUGENE	EP	Pacific		115	185	960
2017	FERNANDA	EP	Pacific		145	233	947
2017	GREG	EP	Pacific		60	97	1001
2017	EIGHT-E	EP	Pacific		35	56	1001
2017	HILARY	EP	Pacific		105	169	972
2017	IRWIN	EP	Pacific		90	145	980
2017	ELEVEN-E	EP	Pacific		35	56	1000
2017	JOVA	EP	Pacific		40	64	1001
2017	KENNETH	EP	Pacific		130	209	952
2017	LIDIA	EP	Pacific		65	105	980
2017	OTIS	EP	Pacific		115	185	960
2017	MAX	EP	Pacific		85	137	980
2017	NORMA	EP	Pacific		75	121	980
2017	PILAR	EP	Pacific		45	72	1000
2017	RAMON	EP	Pacific		45	72	1000
2017	SELMA	EP	Pacific		40	64	1001
2016	PAU	EP	Pacific	Not Reported	98	158	972

SAS Studio interface showing the 'OUTPUT DATA' tab for the 'WORK.NA' table. The table contains 478 rows and 13 columns. The columns are: Season, Name, Basin, Ocean, StormType, MaxWindMPH, MaxWindKM, and MinPressure. The data is displayed in a grid format.

Season	Name	Basin	Ocean	StormType	MaxWindMPH	MaxWindKM	MinPressure
2017	ARLENE	NA	Atlantic		50	80	990
2017	KATIA	NA	Atlantic		105	169	972
2017	BRET	NA	Atlantic		45	72	1007
2017	CINDY	NA	Atlantic		60	97	992
2017	DON	NA	Atlantic		50	80	1007
2017	EMILY	NA	Atlantic		45	72	1005
2017	FRANKLIN	NA	Atlantic		85	137	981
2017	GERT	NA	Atlantic		105	169	967
2017	HARVEY	NA	Atlantic		130	209	938
2017	IRMA	NA	Atlantic		185	298	914
2017	JOSE	NA	Atlantic		155	249	938
2017	LEE	NA	Atlantic		115	185	962
2017	MARIA	NA	Atlantic		175	282	908
2017	NATE	NA	Atlantic		90	145	981
2017	OPHELIA	NA	Atlantic		115	185	960



```

/*****

```

* Generating Data-Dependent Code: Practice #1 *

```

*****/

```

/*Level 1 Practice: Generating Data-Dependent Listing Reports

Reminder: If you restarted your SAS session, you must submit the libname.sas program in the EMC1V2 folder.

Open m105p01.sas from the practices folder.

The program submits a PROC PRINT step to generate a report for the requested customer type in the mc1.customers table.

Submit the program and verify that a report listing Gold high activity customers is produced.

Include the code in a macro named GroupList with two parameters, Tab for the input table and Col for the selected column.

This macro generalizes the starter program so that a PROC PRINT report is generated for each unique value of the specified column.

Use a PROC SQL step to select the distinct uppercase values of the Col parameter.

Load the values into a numbered series of macro variables starting with Val1.

Read the column from the table specified by the Tab parameter.

Add a macro %DO loop to repeat the TITLE statement and PROC PRINT step once for each Valn macro variable created.

Note: Use the Sqllobs macro variable created automatically by the PROC SQL step to provide the stop value for the loop.

Use an indirect macro variable reference to replace GOLD HIGH ACTIVITY with Val1 for the first %DO loop iteration,

Val2 for the second iteration, and so on.

Call the GroupList macro with mc1.customers and Type as the parameter values.

Confirm that a separate PROC PRINT step is executed for each distinct value of Type.

Call the GroupList macro with sashelp.cars and DriveTrain as the parameter values.

What is the value of DriveTrain for the first report generated?

*/

title "Group: GOLD HIGH ACTIVITY";

proc print data=mc1.customers;

where upcase(Type)="GOLD HIGH ACTIVITY";

run;

NOTE: There were 234 observations read from the data set MC1.CUSTOMERS.

WHERE UPCASE(Type)='GOLD HIGH ACTIVITY';

Group: GOLD HIGH ACTIVITY

Obs	ID	Type	Group	Country	Name	Age_Group	Age	BirthDate
1	1	Gold high activity	Orion Club Gold members	FR	Albert Collet	61-75 years	63	-1133
8	351	Gold high activity	Orion Club Gold members	ES	Federico Gayo	46-60 years	53	2320
14	651	Gold high activity	Orion Club Gold members	US	Jules Morton	15-30 years	21	14163
15	701	Gold high activity	Orion Club Gold members	US	Daniel Rover	15-30 years	23	13328
16	751	Gold high activity	Orion Club Gold members	ES	Jordi Robles	15-30 years	21	14200

%macro GroupList(Tab, Col);

proc sql noprint;

select distinct upcase(&Col)

into :Val1-

from 	

quit;

%do i=1 %to &sqllobs;

title "Group: &&val&i";

```

proc print data=&tab;

    where upcase(&col)="%&&val&i";

run;

%end;

%mend GroupList;

%GroupList(mc1.customers, Type)

```

Group: CLUB HIGH ACTIVITY

Obs	ID	Type	Group	Country	Name	Age_Group	Age	BirthDate
18	851	Club high activity	Orion Club members	FR	Olivier Le Neve - Ricordel	15-30 years	23	13489
24	1151	Club high activity	Orion Club members	FR	Jacques Guilchard	31-45 years	38	7964
29	1401	Club high activity	Orion Club members	FR	Daphne Forot	15-30 years	23	13274
32	1551	Club high activity	Orion Club members	US	Brett Roerink	15-30 years	28	11500

Group: CLUB INACTIVE

Obs	ID	Type	Group	Country	Name	Age_Group	Age	BirthDate
10	451	Club inactive	Orion Club members	GB	Stephen Ralham	31-45 years	33	9663
22	1051	Club inactive	Orion Club members	FR	Micheline Giangrande	31-45 years	38	7916
23	1101	Club inactive	Orion Club members	IT	Giuseppe Covili	31-45 years	38	7981
27	1301	Club inactive	Orion Club members	US	Dickie Shellhorn	61-75 years	73	-4845

Group: CLUB LOW ACTIVITY

Obs	ID	Type	Group	Country	Name	Age_Group	Age	BirthDate
20	951	Club low activity	Orion Club members	IT	Rosellina Nizzetto	61-75 years	73	-5030
21	1001	Club low activity	Orion Club members	US	Wali Gatzmer	46-60 years	53	2206
34	1651	Club low activity	Orion Club members	IT	Samara Massabo'	61-75 years	73	-4803
46	2251	Club low activity	Orion Club members	DE	Susanne Laxner	61-75 years	68	-3062

Group: GOLD HIGH ACTIVITY

Obs	ID	Type	Group	Country	Name	Age_Group	Age	BirthDate
1	1	Gold high activity	Orion Club Gold members	FR	Albert Collet	61-75 years	63	-1133
8	351	Gold high activity	Orion Club Gold members	ES	Federico Gayo	46-60 years	53	2320
14	651	Gold high activity	Orion Club Gold members	US	Jules Morton	15-30 years	21	14163
15	701	Gold high activity	Orion Club Gold members	US	Daniel Rover	15-30 years	23	13328

Group: NO MEMBER STATUS

Obs	ID	Type	Group	Country	Name	Age_Group	Age	BirthDate
2	51	No member status	No membership	FR	Jean-Claude Nagues	31-45 years	33	9703
12	551	No member status	No membership	US	Blu Peachey	31-45 years	38	7746
13	601	No member status	No membership	FI	Ari Nippula	31-45 years	33	9828
52	2551	No member status	No membership	US	Christophor Scott	61-75 years	73	-4984

%GroupList(sashelp.cars, DriveTrain)

Group: ALL

Obs	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	MPG_City	MPG_Highway	Weight	Wheelbase	Length
1	Acura	MDX	SUV	Asia	All	\$36,945	\$33,337	3.5	6	265	17	23	4451	106	189
11	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	\$33,430	\$30,366	3.0	6	220	17	26	3583	104	179
12	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	\$34,480	\$31,388	3.0	6	220	18	25	3627	104	179
14	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	\$39,640	\$35,992	3.0	6	220	18	25	3880	109	192

/******

* Generating Data-Dependent Code: Practice #2 *

*****/

/*Level 2 Practice: Exporting Data to Separate Worksheets in Microsoft Excel

Reminder: If you restarted your SAS session, you must submit the libname.sas program in the EMC1V2 folder.

Open m105p02.sas from the practices folder. Review and submit the program.

In the course data folder, select the NASTorms.xlsx file, and click the Download icon.

Confirm storms from the NA basin are included and close the Excel file after viewing it.

Create a macro named BasinXLS that writes storms for each distinct value of Basin to a different worksheet

in an Excel workbook named BasinStorms.xlsx. Include the following logic in the macro program:

Use the automatic macro variable Syslibrc to verify whether the LIBNAME statement runs successfully (SYSLIBRC=0).

If it does not run successfully (SYSLIBRC≠0), write a custom error message to the log and terminate macro execution.

Create a numbered series of macro variables for each value of Basin and BasinName in the mc1.storm_basin_codes table.

Remove all spaces from the BasinName values before assigning them to macro variables.

Use a macro %DO loop to repeat the DATA step for each value of Basin.

Call the %BasinXLS macro and open the BasinStorms.xlsx file. Confirm that six worksheets are included.

What is the name of the first storm listed in the SouthIndian_Storms worksheet?

*/

```
libname storm xlsx "&path/NAstorms.xlsx";
```

```
data storm.NorthAtlantic_Storms;
```

```
    set mc1.storm_final;
```

```
    where Basin="NA";
```

```
run;
```

```
libname storm clear;
```

NOTE: There were 478 observations read from the data set
MC1.STORM_FINAL.

```
    WHERE Basin='NA';
```

NOTE: The data set STORM.NorthAtlantic_Storms has 478 observations
and 13 variables.

NOTE: The export data set has 478 observations and 13 variables.

*Solution;

```
%macro basinxls;
```

```
libname storm xlsx "&path/basinstorms.xlsx";
```

```
%if &syslibrc ne 0 %then %do;
```

```
    %put ERROR: BasinStorms.xlsx was not successfully assigned. Check the path and filename.;
```

```
%end;
```

```
%else %do;
```

```
    proc sql noprint;
```

```
    select Basin, compress(BasinName)
```

```
        into :basin1-, :basinname1-
```

```
            from mc1.storm_basin_codes;
```

```
quit;
```

```

%do i=1 %to &sqlobs;

data storm.&&basinname&i._Storms;

    set mc1.storm_final;

    where Basin="&&basin&i";

run;

%end;

%end;

libname storm clear;

%mend basinxls;

```

```
%basinxls
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	I
1	Season	Name	Basin	Ocean	StormType	MaxWindMPH	MaxWindKM	MinPressure	StartDate	EndDate	StormLength	Lat	Lon	
2	2017	ALFRED	SI	Indian		50	80	994	16-Feb-17	22-Feb-17	6			
3	2017	BLANCHE	SI	Indian		65	105	984	2-Mar-17	7-Mar-17	5			
4	2017	CALEB	SI	Indian		50	80	989	23-Mar-17	27-Mar-17	4			
5	2017	ERNIE	SI	Indian		140	225	922	5-Apr-17	10-Apr-17	5			
6	2017	FRANCES	SI	Indian		75	121	981	21-Apr-17	1-May-17	10			
7	2017	GREG	SI	Indian		40	64	997	29-Apr-17	3-May-17	4			
8	2017	CEMPAKA	SI	Indian		40	64	998	22-Nov-17	1-Dec-17	9			
9	2017	DAHLIA	SI	Indian		60	97	985	24-Nov-17	5-Dec-17	11			
10	2017	HILDA	SI	Indian		60	97	980	24-Dec-17	30-Dec-17	6			
11	2016	ANNABELLE	SI	Indian	Tropical	63	101	983	18-Nov-15	27-Nov-15	9	-15	75	
12	2016	BOHALE	SI	Indian	Tropical	40	64	995	7-Dec-15	16-Dec-15	9	-18.1	68.2	
13	2016	NONAME	SI	Indian	Not Reported	35	56	994	21-Dec-15	1-Jan-16	11	-12.5	131.4	
14	2016	CORENTIN	SI	Indian	Tropical	69	111	975	19-Jan-16	31-Jan-16	12	-21.6	71.9	
15	2016	STAN	SI	Indian	Not Reported	63	101	980	27-Jan-16	1-Feb-16	5	-18.7	118.9	
16	2016	DAYA	SI	Indian	Extratropical	44	71	983	7-Feb-16	13-Feb-16	6	-40.1	63.2	
17	2016	URIAH	SI	Indian	Extratropical	127	204	925	9-Feb-16	25-Feb-16	16	-19.3	79.6	
18	2016	EMERAUDE	SI	Indian	Tropical	127	204	940	14-Mar-16	23-Mar-16	9	-10.5	84.1	
19	2016	SEVENTEEN	SI	Indian	Tropical	52	84	993	27-Mar-16	30-Mar-16	3	-21.6	80	
20	2016	FANTALA	SI	Indian	Tropical	155	249	910	10-Apr-16	26-Apr-16	16	-9.8	50.8	
21	2015	ADJALI	SI	Indian	Tropical	69	111	987	15-Nov-14	24-Nov-14	9	-9.1	67.4	
22	2015	TWO	SI	Indian	Tropical	37	60	997	23-Nov-14	2-Dec-14	9	-15.6	61.9	
23	2015	KATE	SI	Indian	Tropical	104	167	950	21-Dec-14	4-Jan-15	14	-12.5	94	
24	2015	BANSI	SI	Indian	Extratropical	138	222	910	8-Jan-15	19-Jan-15	11	-17.5	57.3	
25	2015	CHEDZA	SI	Indian	Extratropical	66	106	975	13-Jan-15	22-Jan-15	9	-19.8	44	
26	2015	DIAMONDRA	SI	Indian	Extratropical	52	84	985	24-Jan-15	1-Feb-15	8	-18.8	78.9	
27	2015	EUNICE	SI	Indian	Extratropical	144	232	915	24-Jan-15	2-Feb-15	9	-18.5	68	
28	2015	FUNDI	SI	Indian	Tropical	63	101	978	5-Feb-15	11-Feb-15	6	-30.6	44.4	
29	2015	GLENDA	SI	Indian	Extratropical	58	93	970	20-Feb-15	3-Mar-15	11	-25.3	67	
30	2015	HALIBA	SI	Indian	Extratropical	52	84	991	4-Mar-15	13-Mar-15	9	-20.3	52.5	
31	2015	OLWYN	SI	Indian	Not Reported	86	138	955	8-Mar-15	14-Mar-15	6	-22.21	113.74	
32	2015	JOALANE	SI	Indian	Tropical	86	138	962	2-Apr-15	17-Apr-15	15	-15.1	64.7	
33	2015	IKOLA	SI	Indian	Not Reported	109	175	953	4-Apr-15	8-Apr-15	4	-16.85	92.07	
34	2015	QUANG	SI	Indian	Not Reported	115	185	950	27-Apr-15	1-May-15	4	-17.8	109.95	
35	2014	ONE	SI	Indian	Tropical	40	64	997	23-Oct-13	29-Oct-13	6	-11.8	66	
36	2014	AMARA	SI	Indian	Tropical	127	204	935	15-Dec-13	27-Dec-13	12	-19.6	64.7	
			NorthAtlantic_Storms		WestPacific_Storms		EastPacific_Storms		SouthPacific_Storms		NorthIndian_Storms		SouthIndian_Storms	

```
*****;
```


%mend splittable;

%splittable(sashelp.cars,Origin)

Table: WORK.ASIA View: Column names Filter: (none)

Columns: Select all, Make, Model, Type, Origin, DriveTrain, MSRP, Invoice, EngineSize, Cylinders, Horsepower, MPG_City, MPG_Highway

Property Value Label

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice
1	Acura	MDX	SUV	Asia	All	\$36,945	\$33,337
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	\$23,820	\$21,761
3	Acura	TSX 4dr	Sedan	Asia	Front	\$26,990	\$24,647
4	Acura	TL 4dr	Sedan	Asia	Front	\$33,195	\$30,299
5	Acura	3.5 RL 4dr	Sedan	Asia	Front	\$43,755	\$39,014
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	\$46,100	\$41,100
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	\$89,765	\$79,978
8	Honda	Civic Hybrid 4dr manual (gas/electric)	Hybrid	Asia	Front	\$20,140	\$18,451
9	Honda	Insight 2dr (gas/electric)	Hybrid	Asia	Front	\$19,110	\$17,911
10	Honda	Pilot LX	SUV	Asia	All	\$27,560	\$24,843
11	Honda	CR-V LX	SUV	Asia	All	\$19,860	\$18,419
12	Honda	Element LX	SUV	Asia	All	\$18,690	\$17,334
13	Honda	Civic DX 2dr	Sedan	Asia	Front	\$13,270	\$12,175
14	Honda	Civic HX 2dr	Sedan	Asia	Front	\$14,170	\$12,996
15	Honda	Civic LX 4dr	Sedan	Asia	Front	\$15,850	\$14,531

%splittable(sashelp.cars,DriveTrain)

Table: WORK.ALL View: Column names Filter: (none)

Columns: Select all, Make, Model, Type, Origin, DriveTrain, MSRP, Invoice, EngineSize, Cylinders, Horsepower, MPG_City, MPG_Highway

Property Value Label

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice
1	Acura	MDX	SUV	Asia	All	\$36,945	\$33,337
2	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	\$33,430	\$30,366
3	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	\$34,480	\$31,388
4	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	\$39,640	\$35,992
5	Audi	A4 3.0 Quattro convertible 2dr	Sedan	Europe	All	\$44,240	\$40,075
6	Audi	A6 2.7 Turbo Quattro 4dr	Sedan	Europe	All	\$42,840	\$38,840
7	Audi	A6 4.2 Quattro 4dr	Sedan	Europe	All	\$49,690	\$44,936
8	Audi	A8 L Quattro 4dr	Sedan	Europe	All	\$69,190	\$64,740
9	Audi	S4 Quattro 4dr	Sedan	Europe	All	\$48,040	\$43,556
10	Audi	TT 1.8 Quattro 2dr (convertible)	Sports	Europe	All	\$37,390	\$33,891
11	Audi	TT 3.2 coupe 2dr (convertible)	Sports	Europe	All	\$40,590	\$36,739
12	Audi	A6 3.0 Avant Quattro	Wagon	Europe	All	\$40,840	\$37,060
13	Audi	S4 Avant Quattro	Wagon	Europe	All	\$49,090	\$44,446
14	BMW	X3 3.0i	SUV	Europe	All	\$37,000	\$33,873
15	BMW	X5 4.4i	SUV	Europe	All	\$52,195	\$47,720

*****;

* Activity 5.06 *;

* 1) Review the program and notice that the query is *;

* reading all columns from the dictionary.columns *;

```

* table for sashelp.cars. Run the program and observe *;
* the information available in the generated report. *;
* 2) Modify the query to create a macro variable named *;
* Varlist that is a comma-delimited list of values in *;
* the Name column. *;
* 3) Add &Varlist after the colon in the %PUT statement. *;
* Run the program and verify that the log lists the *;
* column names. *;
* 4) Modify the program to examine the columns in *;
* sashelp.class. Run the program. How many columns *;
* are listed in the log? *;
*****

```

```
options nolabel;
```

```
proc sql;
```

```
select *
```

```
    from dictionary.columns
```

```
    where libname="SASHELP" and
```

```
           memname="CARS";
```

```
quit;
```

```
%put NOTE: Columns in SASHELP.CARS include: ;
```

```
options label;
```

libname	memname	memtype	name	type	length	npos	varnum	label	format	informat	idxusage	sortedby	xtype	notnull	precision	scale	transcode
SASHELP	CARS	DATA	Make	char	13	80	1					1	char	no	0	.	yes
SASHELP	CARS	DATA	Model	char	40	93	2					0	char	no	0	.	yes
SASHELP	CARS	DATA	Type	char	8	133	3					2	char	no	0	.	yes
SASHELP	CARS	DATA	Origin	char	6	141	4					0	char	no	0	.	yes
SASHELP	CARS	DATA	DriveTrain	char	5	147	5					0	char	no	0	.	yes
SASHELP	CARS	DATA	MSRP	num	8	0	6		DOLLAR8.			0	num	no	0	.	yes
SASHELP	CARS	DATA	Invoice	num	8	8	7		DOLLAR8.			0	num	no	0	.	yes
SASHELP	CARS	DATA	EngineSize	num	8	16	8					0	num	no	0	.	yes
SASHELP	CARS	DATA	Cylinders	num	8	24	9					0	num	no	0	.	yes
SASHELP	CARS	DATA	Horsepower	num	8	32	10					0	num	no	0	.	yes
SASHELP	CARS	DATA	MPG_City	num	8	40	11					0	num	no	0	.	yes
SASHELP	CARS	DATA	MPG_Highway	num	8	48	12					0	num	no	0	.	yes
SASHELP	CARS	DATA	Weight	num	8	56	13					0	num	no	0	.	yes
SASHELP	CARS	DATA	Wheelbase	num	8	64	14					0	num	no	0	.	yes
SASHELP	CARS	DATA	Length	num	8	72	15					0	num	no	0	.	yes

```
%macro varlist;
```

```
options nolabel;
```

```
proc sql;
```

```
select Name
```

```
    into :varlist separated by ","
```

```
    from dictionary.columns
```

```
    where libname="SASHELP" and
```

```
          memname="CARS";
```

```
quit;
```

```
%put NOTE: Columns in SASHELP.CARS include: &varlist;
```

```
options label;
```

```
%mend varlist;
```

```
%varlist
```

```
NOTE: Columns in SASHELP.CARS include:
```

```
Make,Model,Type,Origin,DriveTrain,MSRP,Invoice,EngineSize,Cylinders,Ho  
rsepower,MPG_City,MPG_Highway,Weight,Wheelbase,Length
```

```
%macro varlist;
```

```
options nolabel;
```

```

proc sql;
select Name
      into :varlist separated by ","
      from dictionary.columns
      where libname="SASHELP" and
            memname="CLASS";
quit;

%put NOTE: Columns in SASHELP.CLASS include: &varlist;

options label;
%mend varlist;

%varlist
NOTE: Columns in SASHELP.CLASS include: Name,Sex,Age,Height,Weight
options mprint;

%macro splittable(tab,col);

options sasautos=("&path/autocall",sasautos);

/* Ensure parameter values are uppercase */
%let tab=%upcase(&tab);
%let col=%upcase(&col);

/* If only a table name is provided, add prefix WORK. */
%if %scan(&tab,2)= %then %do;
    %let tab=WORK.&tab;
%end;

```

```

/* Check if the table exists */
%if %sysfunc(exist(&tab))=0 %then %do;
    %put ERROR: &tab does not exist.;
    %put ERROR- Macro will stop executing.;
    %return;
%end;

/* Check if the column exists in the selected table */
%else %if %varexist(&tab,&col)=0 %then %do;
    proc sql noprint;
    select Name
    into :varlist separated by ", "
    from dictionary.columns
    where libname="%scan(&tab,1)" and
    memname="%scan(&tab,2)";
    quit;
    %put ERROR: Column &col does not exist in &tab..;
    %put ERROR- Columns in &tab include &varlist..;
    %put ERROR- Macro will stop executing.;
    %return;
%end;

/* Create macro variables if COL is numeric */
%else %if %vartype(&tab,&col)=N %then %do;
    proc sql noprint;
    select distinct &col, cats("&col._", &col)
    into :val1-, :table1-
    from &tab

```



```

        where &col is not missing;
quit;
%end;

/* Create macro variables if COL is character */
%else %if %vartype(&tab,&col)=C %then %do;
proc sql noprint;
select distinct &col format=$quote34.,
        compress(&col,, 'nk')
into :val1-, :table1-
from &tab
where &col is not missing;
quit;
%end;

/* Build DATA step */
data
    %do i=1 %to &sqlobs;
        &&table&i
    %end;
;
set &tab;
select(&col);
%do i=1 %to &sqlobs;
    when (&&val&i) output &&table&i;
%end;
otherwise;
end;
run;

```

```
%mend splittable;
```

```
options mprint;
```

```
/* (1) Column with valid table names */
```

```
%splittable(mc1.storm_final,Ocean)
```

```
73      options mprint;
74      /* (1) Column with valid table names */
75      %splittable(mc1.storm_final,Ocean)
MPRINT(SPLITTABLE):  options sasautos=("s:/workshop/emc1v2/autocall",sasautos);
MPRINT(SPLITTABLE):  proc sql noprint;
MPRINT(SPLITTABLE):  select distinct OCEAN format=$quote34., compress(OCEAN,, 'nk') into
:val1-, :table1- from MC1.STORM_FINAL where OCEAN is not missing;
MPRINT(SPLITTABLE):  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

```
MPRINT(SPLITTABLE):  data Atlantic Indian Pacific ;
MPRINT(SPLITTABLE):  set MC1.STORM_FINAL;
MPRINT(SPLITTABLE):  select(OCEAN);
MPRINT(SPLITTABLE):  when ("Atlantic") output Atlantic;
MPRINT(SPLITTABLE):  when ("Indian") output Indian;
MPRINT(SPLITTABLE):  when ("Pacific") output Pacific;
MPRINT(SPLITTABLE):  otherwise;
MPRINT(SPLITTABLE):  end;
MPRINT(SPLITTABLE):  run;
```

```
NOTE: There were 3092 observations read from the data set MC1.STORM_FINAL.
NOTE: The data set WORK.ATLANTIC has 478 observations and 13 variables.
NOTE: The data set WORK.INDIAN has 654 observations and 13 variables.
NOTE: The data set WORK.PACIFIC has 1960 observations and 13 variables.
```

```
73      /* (2) Table does not exist */
```

```
74      %splittable(false, ID)
```

```
MPRINT(SPLITTABLE):  options
sasautos=("~/EMC1V2/autocall",sasautos);
```

```
ERROR: WORK.FALSE does not exist.
Macro will stop executing.
```

```
/* (3) Column does not exist */
```

```
%splittable(sashelp.cars, Test)
```

```
73      /* (3) Column does not exist */
```

```
74      %splittable(sashelp.cars, Test)
```

```
MPRINT(SPLITTABLE):  options
sasautos=("~/EMC1V2/autocall",sasautos);
MPRINT(SPLITTABLE):  proc sql noprint;
```

```
MPRINT(SPLITTABLE):  select Name into :varlist separated by ", "
from dictionary.columns where libname="SASHELP" and
memname="CARS";
MPRINT(SPLITTABLE):  quit;
```

ERROR: Column TEST does not exist in SASHELP.CARS.

Columns in SASHELP.CARS include Make, Model, Type, Origin, DriveTrain, MSRP, Invoice, EngineSize, Cylinders, Horsepower, MPG_City, MPG_Highway, Weight, Wheelbase, Length.
Macro will stop executing.

```
85
86 /* (4) Column includes values with invalid characters */
87 %splittable(sashelp.cars, Make)
88
```

Land Rover
Mercedes-Benz

```
/* (4) Column includes values with invalid characters */
```

```
%splittable(sashelp.cars, Make)
```

```
73          /* (4) Column includes values with invalid characters */
74          %splittable(sashelp.cars, Make)
MPRINT(SPLITTABLE):  options
sasautos=("~/EMC1V2/autocall",sasautos);
MPRINT(SPLITTABLE):  proc sql noprint;
MPRINT(SPLITTABLE):  select distinct MAKE format=$quote34.,
compress(MAKE,,'nk') into :val1-, :table1- from SASHELP.CARS where
MAKE is not missing;
MPRINT(SPLITTABLE):  quit;
```

```
MPRINT(SPLITTABLE):  data Acura Audi BMW Buick Cadillac Chevrolet
Chrysler Dodge Ford GMC Honda Hummer Hyundai Infiniti Isuzu
Jaguar Jeep Kia LandRover Lexus Lincoln MINI Mazda MercedesBenz
Mercury Mitsubishi Nissan Oldsmobile Pontiac Porsche Saab Saturn
Scion Subaru Suzuki Toyota Volkswagen Volvo ;
MPRINT(SPLITTABLE):  set SASHELP.CARS;
MPRINT(SPLITTABLE):  select(MAKE);
MPRINT(SPLITTABLE):  when ("Acura") output Acura;
MPRINT(SPLITTABLE):  when ("Audi") output Audi;
MPRINT(SPLITTABLE):  when ("BMW") output BMW;
```



```
MPRINT(SPLITTABLE):  when (2) output ORDER_TYPE_2;
MPRINT(SPLITTABLE):  when (3) output ORDER_TYPE_3;
MPRINT(SPLITTABLE):  otherwise;
MPRINT(SPLITTABLE):  end;
MPRINT(SPLITTABLE):  run;
```

NOTE: There were 15842 observations read from the data set MC1.ORDERS.

NOTE: The data set WORK.ORDER_TYPE_1 has 11133 observations and 12 variables.

NOTE: The data set WORK.ORDER_TYPE_2 has 2375 observations and 12 variables.

NOTE: The data set WORK.ORDER_TYPE_3 has 2334 observations and 12 variables.

AutoCall:

```
%macro splittable_sql(tab, col);
proc sql noprint;
select distinct &col
    into :tablist separated by " "
    from &tab;
select distinct
    cat('when ("', &col, '"') output ', &col)
    into :whenlist separated by ","
    from &tab;
quit;

data &tablist;
    set &tab;
    select(&col);
        &whenlist;
        otherwise;
    end;
run;
```

```
%mend splittable_sql;
```

```
%macro varexist(dsname,varname);
```

```
/* The OPEN function accesses the table and returns a  
data set ID that is assigned to the macro variable DSID */
```

```
%let dsid = %sysfunc(open(&dsname));
```

```
/* The VARNUM function searches the table for the specified  
variable and returns the variable number, which is assigned  
to the macro variable VAL. If the variable is not found, VARNUM  
returns 0. */
```

```
%let val = %sysfunc(varnum(&dsid,&varname));
```

```
/* Use the CLOSE function to close the table */
```

```
%let rc = %sysfunc(close(&dsid));
```

```
/* Create macro variable EXIST that is 1 if the  
column was found in the table and 0 otherwise. */
```

```
%if &val>0 %then %do;
```

```
    %let exist=1;
```

```
%end;
```

```
%else %do;
```

```
    %let exist=0;
```

```
%end;
```

```

/* Write the value of macro variable EXIST to the
input stack */

&exist

%mend;

%macro vartype(dsname,varname);

/* The OPEN function accesses the table (&DSNAME) and returns a
data set ID that is assigned to the macro variable DSID.
The data set ID is required for the other functions used. */

%let dsid = %sysfunc(open(&dsname));

/* The VARNUM function searches the data set ID (&DSID) for the
specified variable (&VARNAME) and returns the variable ID,
which is assigned to the macro variable VARID. If the variable
is not found, VARID will be 0. The variable ID value is used in
the next function. */

%let varid = %sysfunc(varnum(&dsid,&varname));

/* The VARTYPE function identifies a column based on the data
set ID and variable ID. The function returns C for character
or N for numeric and stores the value in the VAR macro variable. */

%let val = %sysfunc(vartype(&dsid,&varid));

```

```

/* The CLOSE function closes the selected data set. */

%let rc = %sysfunc(close(&dsid));

/* Write the value of macro variable VAL (C or N) to the
input stack */

&val

%mend;

%macro splittable(tab,col);

%if &tab= or &tab=? or %upcase(&tab)=HELP %then %do;

%put WARNING-*****;

%put WARNING-* SplitTable Macro      *;

%put WARNING-* Syntax: %splittable(tab,col) *;

%put WARNING-* Parameters:          *;

%put WARNING-* TAB: Table to split   *;

%put WARNING-* COL: Column to use to create *;

%put WARNING-*      separate tables *;

%put WARNING-*****;

%return;

%end;

/* Ensure parameter values are uppercase */

%let tab=%upcase(&tab);

%let col=%upcase(&col);

```



```
/* If only a table name is provided, add prefix WORK. */
```

```
%if %scan(&tab,2)= %then %do;
```

```
    %let tab=WORK.&tab;
```

```
%end;
```

```
/* Check if the table exists */
```

```
%if %sysfunc(exist(&tab))=0 %then %do;
```

```
    %put ERROR: &tab does not exist.;
```

```
    %put ERROR- Macro will stop executing.;
```

```
    %return;
```

```
%end;
```

```
/* Check if the column exists in the selected table */
```

```
%else %if %varexist(&tab,&col)=0 %then %do;
```

```
    proc sql noprint;
```

```
    select Name
```

```
        into :varlist separated by ", "
```

```
        from dictionary.columns
```

```
        where libname="%scan(&tab,1)" and
```

```
        memname="%scan(&tab,2)";
```

```
    quit;
```

```
    %put ERROR: Column &col does not exist in &tab.;
```

```
    %put ERROR- Columns in &tab include &varlist.;
```

```
    %put ERROR- Macro will stop executing.;
```

```
    %return;
```

```
%end;
```

```
/* Create macro variables if COL is numeric */
```

```

%else %if %vartype(&tab,&col)=N %then %do;
proc sql noprint;
select distinct &col, cats("&col._", &col)
    into :val1-, :table1-
    from &tab
    where &col is not missing;
quit;
%end;

```

```

/* Create macro variables if COL is character */
%else %if %vartype(&tab,&col)=C %then %do;
proc sql noprint;
select distinct &col format=$quote34.,
    compress(&col,,'nk')
    into :val1-, :table1-
    from &tab
    where &col is not missing;
quit;
%end;

```

```

/* Build DATA step */
data
    %do i=1 %to &sqlobs;
        &&table&i
    %end;
;
set &tab;
select(&col);
%do i=1 %to &sqlobs;

```

```

        when (&&val&i) output &&table&i;

    %end;

    otherwise;

end;

run;

%mend splittable;

```

```
%macro splittable(tab,col);
```

Changed with red text:

```

%if &tab= or &tab=? or %upcase(&tab)=HELP %then %do;

    %put WARNING-*****;

    %put WARNING-* SplitTable Macro          *;

    %put %nrstr(WARNING-* Syntax: %splittable(tab,col) *);

```

...

```

169      %splittable(help)
          *****
          * SplitTable Macro          *
          * Syntax: %splittable(tab,col) *
          * Parameters:                *
          *   TAB: Table to split      *
          *   COL: Column to use to create *
          *       separate tables      *
          *****

```

```
/******
```

```
* Validating Parameters: Practice #4 *
```

```
*****/
```

```
/*Level 1 Practice: Validating a Macro Parameter
```

Reminder: If you restarted your SAS session, you must submit the libname.sas program in the EMC1V2 folder.

Open m105p04.sas from the practices folder. Review and submit the program.

Modify the macro:

Convert the Type parameter value to uppercase.

Add %IF-%THEN and %ELSE statements to validate the Type parameter.

The macro should submit the TITLE statement and PROC FREQ step only if the Type parameter is LOW, MEDIUM, or HIGH.

If the Type parameter is not correct, the macro should write the following message to the log,

where xxxx is the Type parameter:

ERROR: Invalid TYPE: xxxx

ERROR: Valid TYPE values are HIGH, MEDIUM, LOW

Resubmit the macro definition and call the macro using a valid parameter value and verify that a report is produced.

Then call the macro with an invalid parameter value. Verify that the error message is displayed in the log.

Modify the macro to first test whether Type is null. If so, do not execute the TITLE statement or PROC FREQ step.

Instead, the macro should write this message to the log:

ERROR: Missing TYPE

ERROR: Valid TYPE values are HIGH, MEDIUM, LOW

Resubmit the macro definition and call the macro with a null parameter value, valid values in uppercase and lowercase,

and an invalid value.

How many low-activity customers are Italian (IT)?

Note: Type a numeric value for your answer.

*/

%macro custtype(type);

```
title "&Type Activity Customers by Country";  
proc freq data=mc1.customers order=freq;  
  where upcase(Type) contains "&type";  
  tables Country;  
run;  
title;  
%mend custtype;  
  
%custtype(MEDIUM)
```

MEDIUM Activity Customers by Country

The FREQ Procedure

Customer Country				
Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent
US	130	30.16	130	30.16
FR	62	14.39	192	44.55
DE	42	9.74	234	54.29
IT	41	9.51	275	63.81
ES	27	6.26	302	70.07
AU	24	5.57	326	75.64
GB	22	5.10	348	80.74
CA	17	3.94	365	84.69
NL	17	3.94	382	88.63
CH	8	1.86	390	90.49
BE	7	1.62	397	92.11
AT	5	1.16	402	93.27
SE	5	1.16	407	94.43
DK	4	0.93	411	95.36
FI	3	0.70	414	96.06
NO	3	0.70	417	96.75
PT	3	0.70	420	97.45
TR	3	0.70	423	98.14
GR	2	0.46	425	98.61
ZA	2	0.46	427	99.07
CI	1	0.23	428	99.30
LU	1	0.23	429	99.54
NZ	1	0.23	430	99.77
YU	1	0.23	431	100.00

```
MPRINT(CUSTTYPE):  title "MEDIUM Activity Customers by Country";
MPRINT(CUSTTYPE):  proc freq data=mc1.customers order=freq;
MPRINT(CUSTTYPE):  where upcase(Type) contains "MEDIUM";
MPRINT(CUSTTYPE):  tables Country;
MPRINT(CUSTTYPE):  run;
```

NOTE: There were 431 observations read from the data set MC1.CUSTOMERS.

WHERE UPCASE(Type) contains 'MEDIUM';

%macro custtype(type) / minoperator;

```

%if &type= %then %do;

    %put ERROR: Missing TYPE;

    %put ERROR: Valid TYPE values are HIGH, MEDIUM, LOW;

%end;

%else %do;

    %let type=%upcase(&type);

    %if &type in HIGH MEDIUM LOW %then %do;

        title "&Type Activity Customers by Country";

        proc freq data=mc1.customers order=freq;

            where upcase(Type) contains "&type";

            tables Country;

        run;

        title;

    %end;

%else %do;

    %put ERROR: Invalid TYPE: &type;

    %put ERROR: Valid TYPE values are HIGH, MEDIUM, LOW;

%end;

%end;

%mend custtype;

%custtype(HIGH)

```

HIGH Activity Customers by Country

The FREQ Procedure

Customer Country				
Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent
US	137	29.72	137	29.72
FR	51	11.06	188	40.78
IT	44	9.54	232	50.33
ES	38	8.24	270	58.57
DE	37	8.03	307	66.59
AU	30	6.51	337	73.10
GB	23	4.99	360	78.09
NL	20	4.34	380	82.43
CA	16	3.47	396	85.90
PT	11	2.39	407	88.29
AT	10	2.17	417	90.46
CH	10	2.17	427	92.62
BE	6	1.30	433	93.93
DK	4	0.87	437	94.79
FI	4	0.87	441	95.66
NO	4	0.87	445	96.53
SE	4	0.87	449	97.40
SI	3	0.65	452	98.05
TR	2	0.43	454	98.48
ZA	2	0.43	456	98.92
GR	1	0.22	457	99.13
IN	1	0.22	458	99.35
LT	1	0.22	459	99.57
LU	1	0.22	460	99.78
PL	1	0.22	461	100.00

%custtype(low)

LOW Activity Customers by Country

The FREQ Procedure

Customer Country				
Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent
US	141	29.75	141	29.75
IT	60	12.66	201	42.41
FR	50	10.55	251	52.95
DE	44	9.28	295	62.24
GB	31	6.54	326	68.78
ES	27	5.70	353	74.47
NL	27	5.70	380	80.17
AU	19	4.01	399	84.18
CA	16	3.38	415	87.55
BE	8	1.69	423	89.24
DK	8	1.69	431	90.93
CH	6	1.27	437	92.19
AT	5	1.05	442	93.25
FI	5	1.05	447	94.30
PT	5	1.05	452	95.36
SE	5	1.05	457	96.41
NO	4	0.84	461	97.26
TR	3	0.63	464	97.89
ZA	3	0.63	467	98.52
GR	2	0.42	469	98.95
HR	1	0.21	470	99.16
IL	1	0.21	471	99.37
LU	1	0.21	472	99.58
MZ	1	0.21	473	99.79
SI	1	0.21	474	100.00

```
%custtype(med)
```

```
73          %custtype(med)
ERROR: Invalid TYPE: MED
ERROR: Valid TYPE values are HIGH, MEDIUM, LOW
```

```
%custtype()
```

```
73          %custtype()
ERROR: Missing TYPE
ERROR: Valid TYPE values are HIGH, MEDIUM, LOW
```

```
/******
```

```
* Validating Parameters: Practice #5 *
```

```
*****/
```

```
/*Level 2 Practice: Validating Macro Parameters with Data Values
```

Reminder: If you restarted your SAS session, you must submit the libname.sas program in the EMC1V2 folder.

Open m105p05.sas from the practices folder. Review and submit the program.

Use PROC SQL to create the macro variables MinSeason and MaxSeason that store the minimum and maximum values of

the Season column in the mc1.storm_final table.

Add a condition to verify that the Season parameter is between MinSeason and MaxSeason.

If the Season parameter is outside of the valid range, write this message to the log:

ERROR: Valid Seasons are between <minSeason> and <maxSeason>

If the Season parameter is valid, use PROC SQL to create a space-delimited list of valid Basin codes from the mc1.storm_basin_codes table. Load the list into a macro variable.

If the Basin parameter is in the list of valid codes, execute the TITLE statements and the PROC SGPLOT step.

Note: Be sure to use the MINOPERATOR option in the %MACRO statement to enable the macro IN operator.

If the Basin parameter is not in the list of valid codes, write this message to the log:

ERROR: AA is an invalid basin code. Basin codes include NA WP EP SP NI SI.

Test the macro with valid and invalid values for both Basin and Season.

How many tropical storms were there in the EP basin in 2010?

Note: Type a numeric value for your answer.

```
*/
```

```
%macro stormchart(basin, season);
```

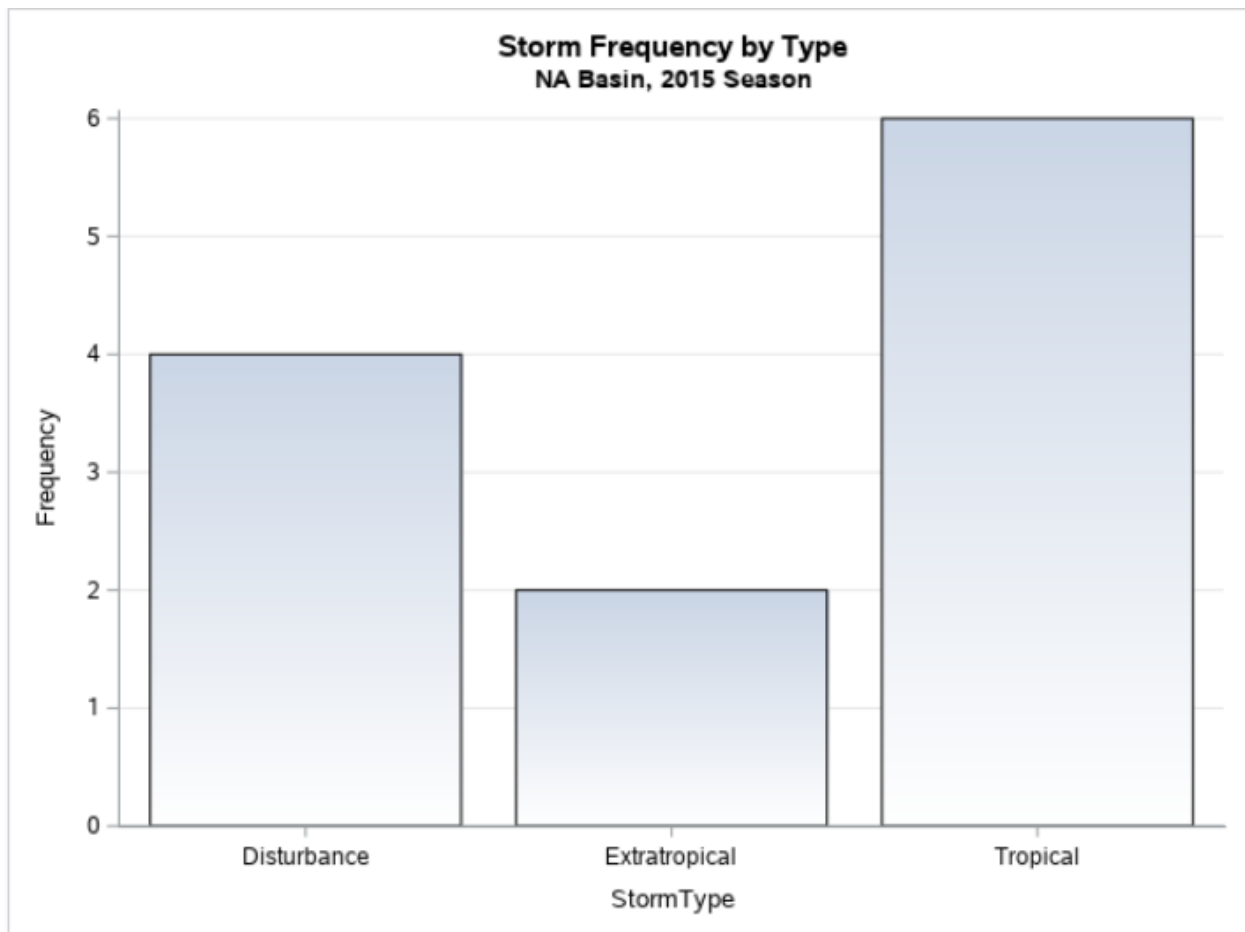
```
title1 "Storm Frequency by Type";
```

```

title2 "&basin Basin, &season Season";
proc sgplot data=mc1.storm_final noborder;
    vbar StormType / filltype=gradient;
    where Basin="&basin" and Season=&season;
    yaxis grid;
run;
title;
%mend stormchart;

%stormchart(NA,2015)

```



```

%macro stormchart(basin, season) / minoperator;
proc sql noprint;
select min(season), max(season)

```

```

        into :minseason trimmed, :maxseason trimmed
        from mc1.storm_final;
quit;

%if &season<&minseason or &season>&maxseason %then %do;
    %put ERROR: Valid Seasons are between 1980 and 2017;
%end;

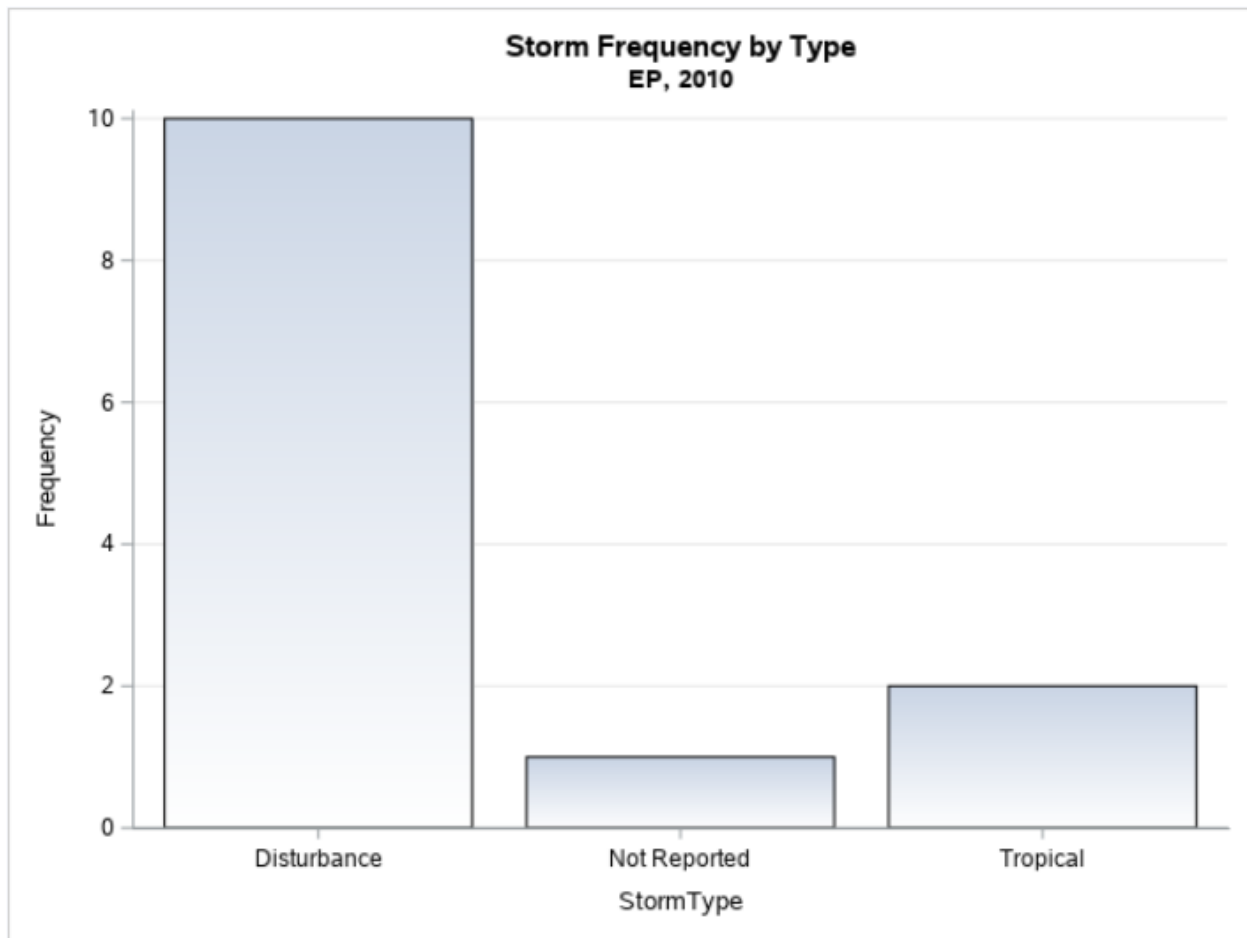
%else %do;
    proc sql noprint;
        select Basin
            into :basinlist separated by " "
            from mc1.storm_basin_codes;
quit;
%if &basin in &basinlist %then %do;
    title1 "Storm Frequency by Type";
    title2 "&basin, &season";
    proc sgplot data=mc1.storm_final noborder;
        vbar StormType / filltype=gradient;
        where Basin="&Basin" and Season=&Season;
        yaxis grid;
    run;
    title;
%end;
%else %do;
    %put ERROR: &basin is an invalid basin code. Basin codes include &basinlist..;
%end;
%end;
%mend stormchart;

```

```
%stormchart(NA,2020)
```

ERROR: Valid Seasons are between 1980 and 2017

```
%stormchart(EP,2010)
```



```
MPRINT(STORMCHART): title1 "Storm Frequency by Type";  
MPRINT(STORMCHART): title2 "EP, 2010";  
MPRINT(STORMCHART): proc sgplot data=mc1.storm_final noborder;  
MPRINT(STORMCHART): vbar StormType / filltype=gradient;  
MPRINT(STORMCHART): where Basin="EP" and Season=2010;  
MPRINT(STORMCHART): yaxis grid;  
MPRINT(STORMCHART): run;
```

NOTE: There were 13 observations read from the data set MC1.STORM_FINAL.

WHERE (Basin='EP') and (Season=2010);

```
%stormchart(AA,2010)
```

ERROR: AA is an invalid basin code. Basin codes include NA WP EP SP NI SI.