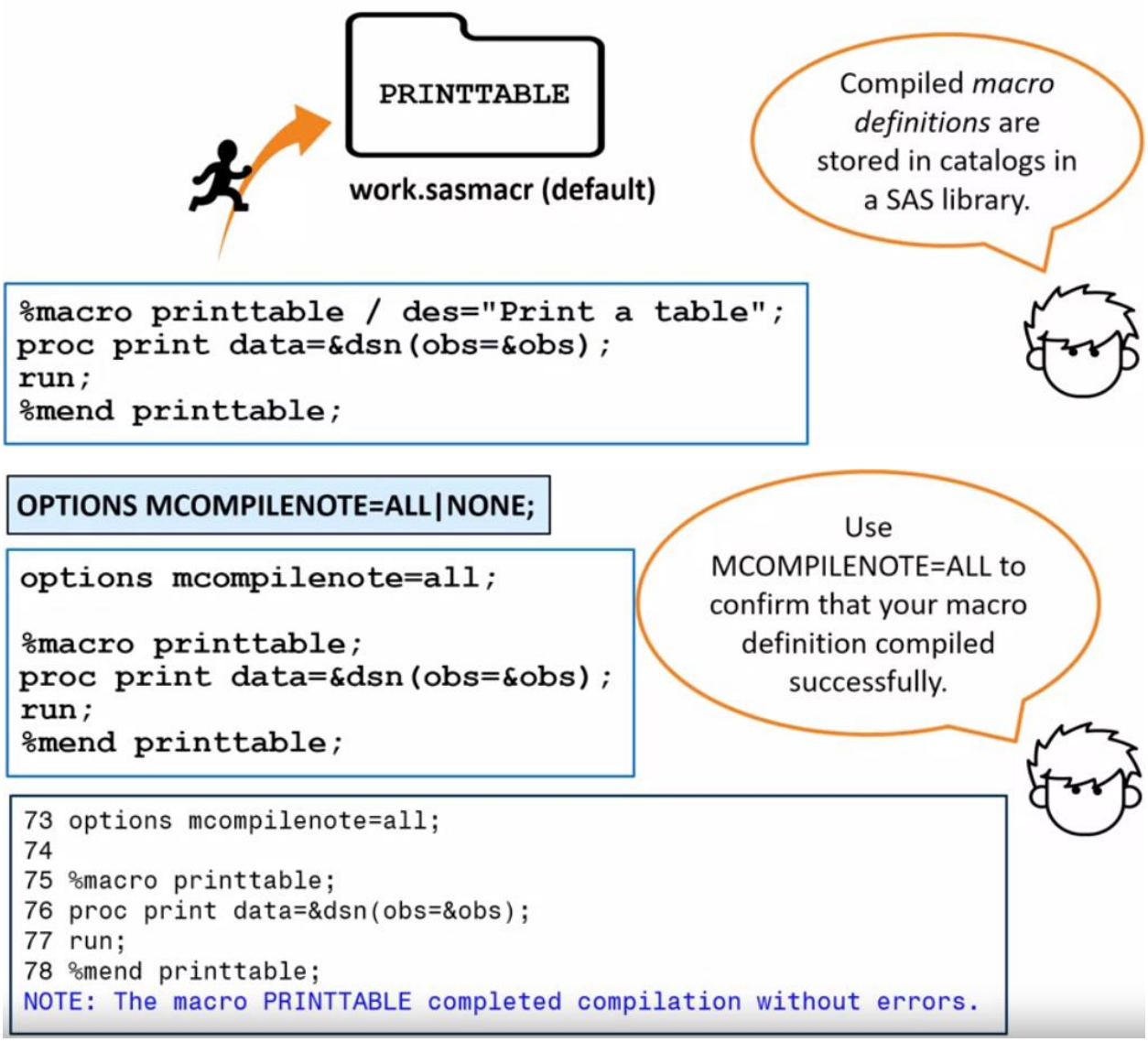
SAS Advanced Programmer

SAP2 Macro

SAP204 Defining and Calling a Macro, Macro Variable Scope, Conditional Processing, Iterative Processing

```
%let path=~/EMC1V2;

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



```
options mprint;
%printtable
options nomprint;
```

OPTIONS MPRINT;

OPTIONS NOMPRINT;

```
84 %printtable
MPRINT(PRINTTABLE): proc print data=sashelp.clas(obs=5);
ERROR: File SASHELP.CLAS.DATA does not exist.
MPRINT(PRINTTABLE): run;
```

name of the macro generating the code

SAS code generated by the macro

macro triggers are already resolved

Create a macro definition with parameters

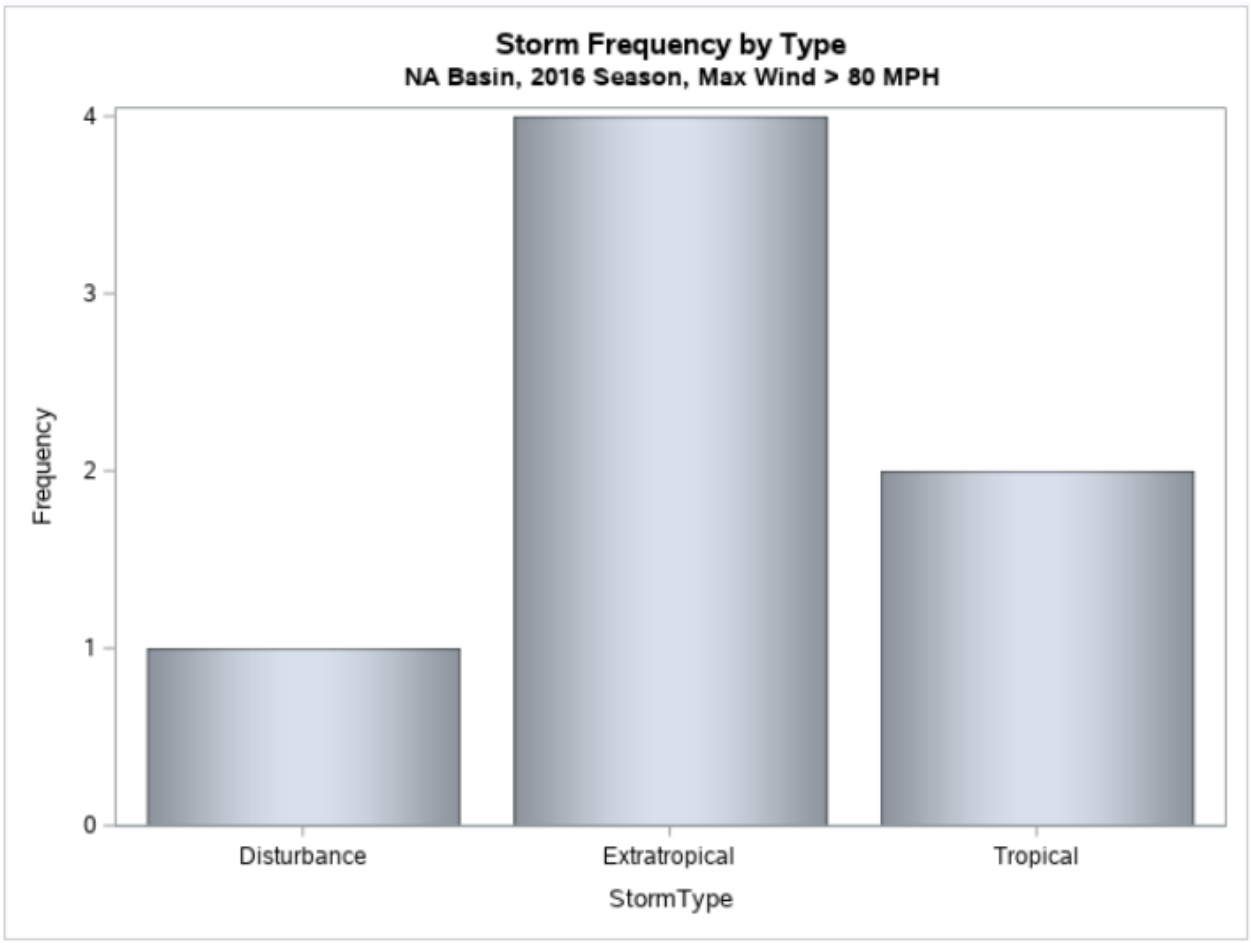
```
%macro printable(dsn,obs);
proc print data=&dsn(obs=&obs);
run;
%mend printtable;
```

```
%printtable(sashelp.cars,5)
```

Using macro parameters is a practical alternative to assigning macro variable values using %LET.

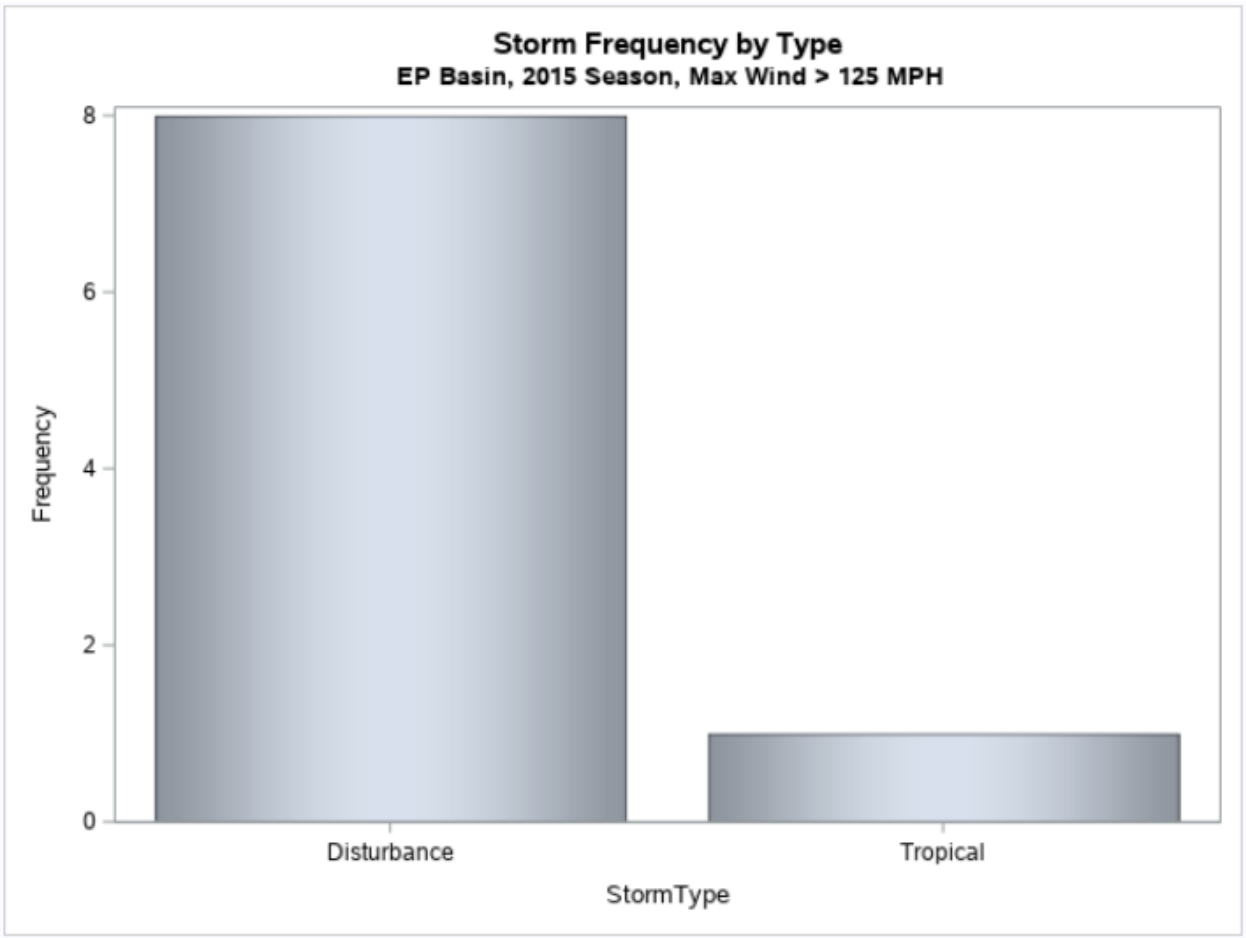
```
/***********************************
* Defining and Calling a Macro: Demo *
***********************************/
%let Basin=NA;
%let Season=2016;
%let MaxWind=80;

title1 "Storm Frequency by Type";
title2 "&Basin Basin, &Season Season, Max Wind > &MaxWind MPH";
```

```
proc sgplot data=mc1.storm_final;
   vbar StormType / dataskin=pressed;
   where Basin="&Basin" and Season=&Season and MaxWindMPH>&MaxWind;
run;
title;
```




```
%let Basin=EP;
%let Season=2015;
%let MaxWind=125;

title1 "Storm Frequency by Type";
title2 "&Basin Basin, &Season Season, Max Wind > &MaxWind MPH";
proc sgplot data=mc1.storm_final;
   vbar StormType / dataskin=pressed;
```

```
   where Basin="&Basin" and Season=&Season and MaxWindMPH>&MaxWind;
run;
title;
```



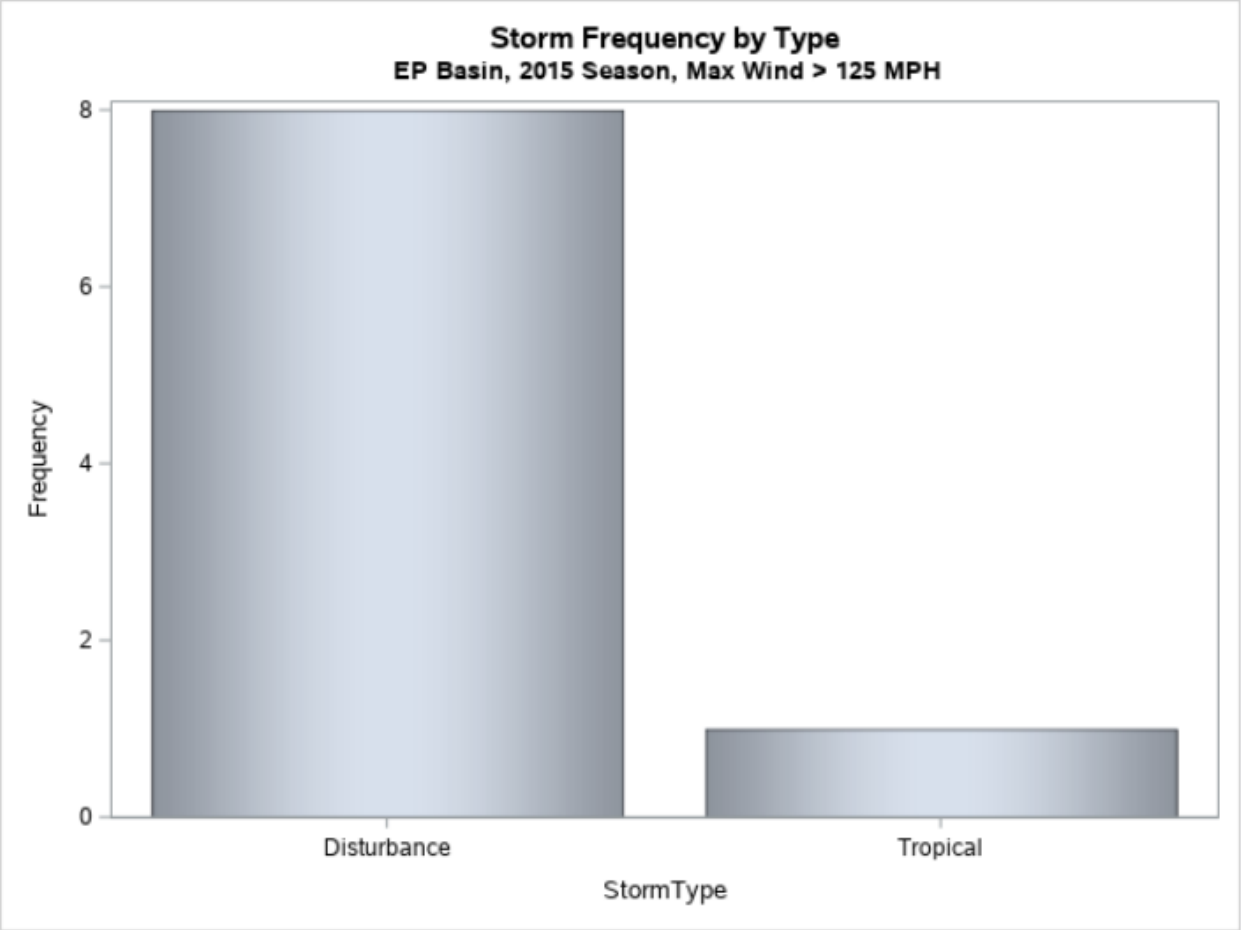

```
options mcompilenote=all;
%macro StormChart(Basin, Season, MaxWind);
title1 "Storm Frequency by Type";
title2 "&Basin Basin, &Season Season, Max Wind > &MaxWind MPH";
proc sgplot data=mc1.storm_final;
   vbar StormType / dataskin=pressed;
   where Basin="&Basin" and Season=&Season and MaxWindMPH>&MaxWind;
run;
title;
%mend StormChart;
```

```
NOTE: The macro STORMCHART completed compilation without errors.
      11 instructions 504 bytes.
```

options mprint;

%StormChart(EP,2015,125)

options nomprint;




```
MPRINT(STORMCHART):   title1 "Storm Frequency by Type";
MPRINT(STORMCHART):   title2 "EP Basin, 2015 Season, Max Wind > 125 MPH";
MPRINT(STORMCHART):   proc sgplot data=mc1.storm_final;
MPRINT(STORMCHART):   vbar StormType / dataskin=pressed;
MPRINT(STORMCHART):   where Basin="EP" and Season=2015 and MaxWindMPH>125;
MPRINT(STORMCHART):   run;
```

```
%macro printtable(dsn=sashelp.cars,obs=5);
proc print data=&dsn(obs=&obs);
run;
%mend printtable;
```

```
%printtable()
```

```
%printtable(dsn=sashelp.heart)
```

```
%printtable(obs=10,dsn=mc1.storm_final)
```

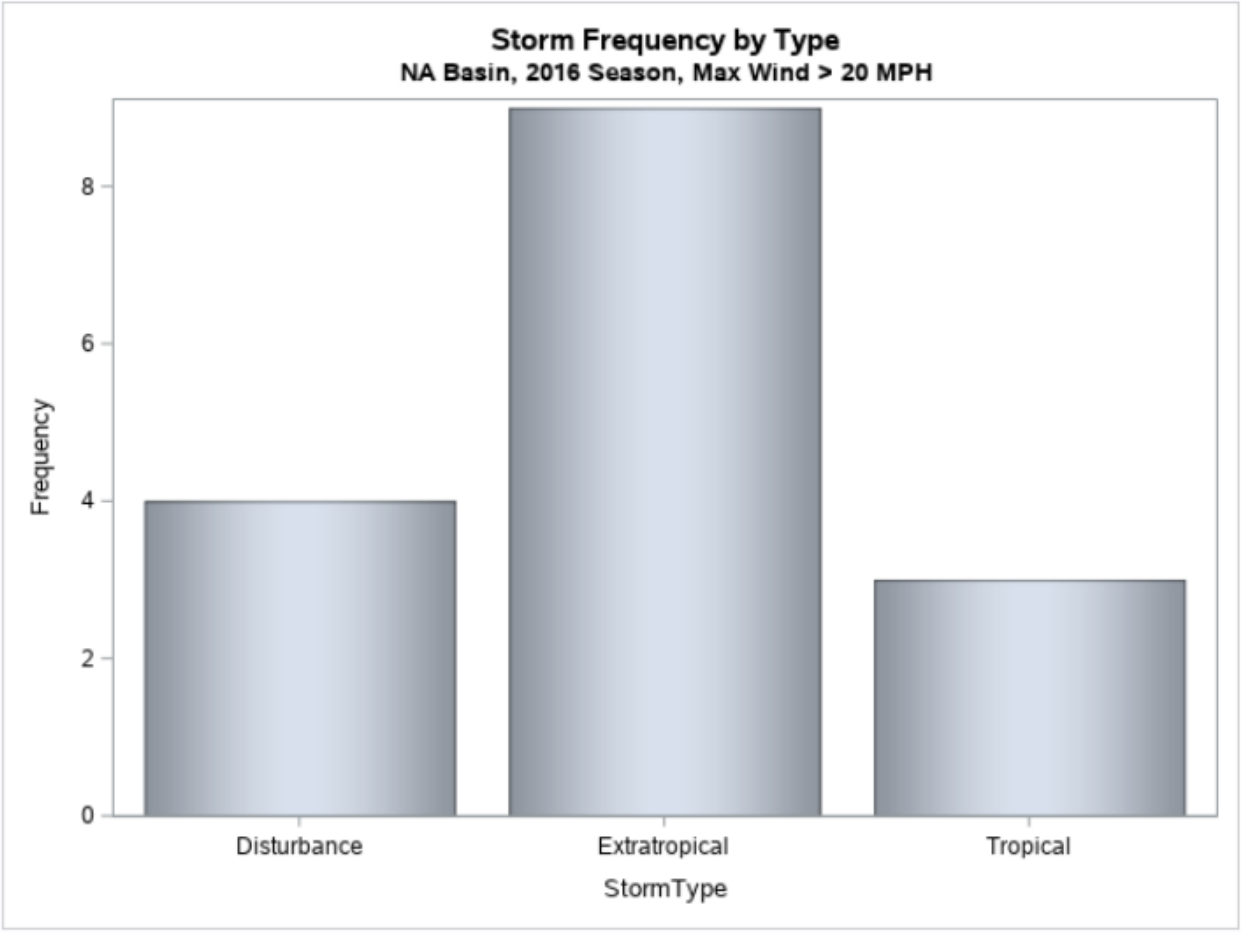
```
************************************************************;
*  Activity 4.03                                           *;
*  1) Modify the %MACRO statement to use keyword           *;
*     parameters. Assign default values of NA to Basin,    *;
*     2016 to Season, and 20 to MaxWind. Run the program   *;
*     and verify that the macro compiles successfully.     *;
*  %macro stormchart(basin=NA, season=2016, maxwind=20);   *;
*  2) Call the %StormChart macro with no parameter values  *;
*     provided. Confirm in the log that the default        *;
*     parameter values are used.                           *;
*       %stormchart()                                      *;
*  3) Call the %StormChart macro with Season as 2015 and   *;
*     Basin as EP. View the log. Is the table subset by    *;
*     MaxWind, even though it is not included as a         *;
*     parameter in the macro call?                         *;
************************************************************;


options mcompilenote=all;
```

```
%macro stormchart(basin=NA, season=2016, maxwind=20);
title1 "Storm Frequency by Type";
title2 "&basin Basin, &season Season, Max Wind > &maxwind MPH";
proc sgplot data=mc1.storm_final;
   vbar StormType / dataskin=pressed;
   where Basin="&basin" and Season=&season and MaxWindMPH>&maxwind;
run;
title;
%mend stormchart;
```

```
NOTE: The macro STORMCHART completed compilation without errors.
      11 instructions 528 bytes.
```

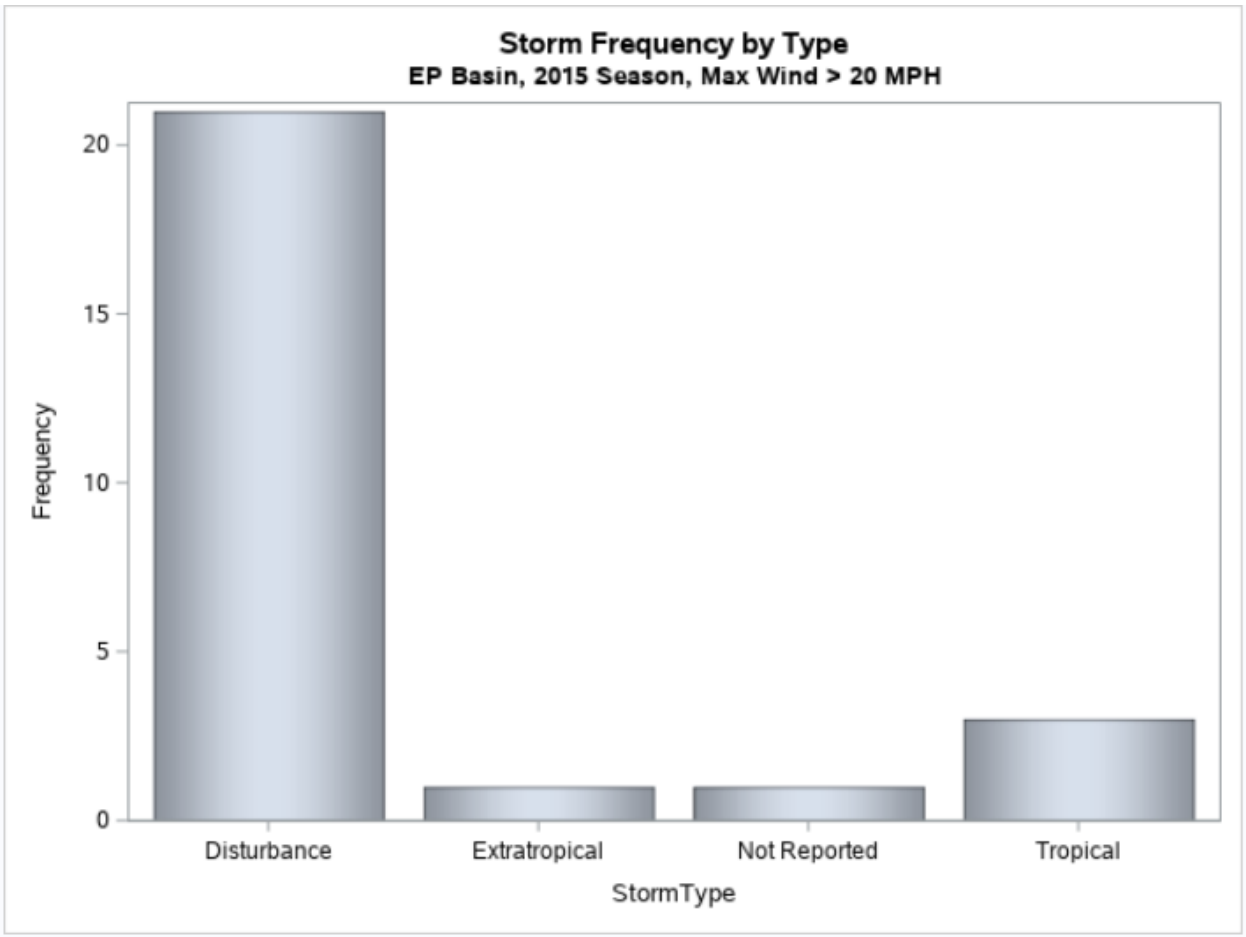
```
%StormChart()
```

```
NOTE: There were 16 observations read from the data set MC1.STORM_FINAL.
      WHERE (Basin='NA') and (Season=2016) and (MaxWindMPH>20);
```



%StormChart(Season=2015, Basin=EP)

```
NOTE: There were 26 observations read from the data set MC1.STORM_FINAL.
      WHERE (Basin='EP') and (Season=2015) and (MaxWindMPH>20);
```




```
%macro stormchart(Basin, Season=2016, MaxWind=111);
```

Basin doesn't have a default value, so a value must be provided in each macro call.

```
%stormchart(WP)
%stormchart(NA, Season=2015)
%stormchart(SI, MaxWind=100, Season=2014)
```

/***

* Defining and Calling a Macro: Practice #1 *

***/

/*Level 1 Practice: Defining and Using a Macro with Parameters

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

in the EMC1V2 folder to access your practice files.

Open m104p01.sas from the practices folder. Review and submit the program.

Verify that it generates a report for Gold customers that contains 680 rows.

Convert the code to a macro named Customers that accepts a positional parameter named Type.

Submit the macro definition and review the log to verify that the macro compiled successfully.

Call the Customers macro with the parameter value Gold. Verify that the title is Gold Customers and

that there are 680 rows in the report.

Call %Customers again with the parameter value High Activity. Verify that no rows were selected due to a case difference.

Modify the macro definition:

Convert the value of the parameter Type to uppercase.

Modify the WHERE clause to make the string comparison case insensitive.

Submit the macro definition and verify that it compiled correctly.

Call the macro with the parameter value High Activity.

Verify that the title is HIGH ACTIVITY Customers and that the report contains 461 rows.

Change the positional parameter to a keyword parameter with a default value of inactive.

Call %Customers with the value high activity for the keyword parameter.

Verify that the title is HIGH ACTIVITY Customers and that the report contains 461 rows.

Call %Customers without passing a parameter value. Verify that title is INACTIVE Customers.

How many rows are in the final report?

Note: Type a numeric value for your answer.

*/

```
%let type=Gold;
title "&type Customers";
proc sql number;
select Name, Age_Group, Type
```

```
   from mc1.customers
   where Type contains "&type";
quit;
title;
```

Gold Customers

Row	Customer Name	Customer Age Group	Type
1	Albert Collet	61-75 years	Gold high activity
2	Nogarji Mcadam	61-75 years	Gold low activity
3	Laurent Ollivon	15-30 years	Gold medium activity
677	Juan Marcos Iglesias López	46-60 years	Gold high activity
678	Luc Vandeloo	61-75 years	Gold low activity
679	Daniel Pallen	46-60 years	Gold high activity
680	Alessandra Macci	61-75 years	Gold low activity

```
%macro Customers(type=Gold);
title "&type Customers";
proc sql number;
select Name, Age_Group, Type
   from mc1.customers
   where Type contains "&type";
quit;
title;
%mend Customers;
```

```
NOTE: The macro CUSTOMERS completed compilation without errors.
      7 instructions 272 bytes.
```

```
%Customers
```

Gold Customers

Row	Customer Name	Customer Age Group	Type
1	Albert Collet	61-75 years	Gold high activity
2	Nogarji Mcadam	61-75 years	Gold low activity
3	Laurent Ollivon	15-30 years	Gold medium activity

```
%macro Customers(Type);
title "&type Customers";
proc sql number;
select Name, Age_Group, Type
   from mc1.customers
   where Type contains "&type";
quit;
title;
%mend Customers;

%Customers(Gold)

/* Part c. */
%macro customers(type);
%let type=%upcase(&type);
title "&type Customers";
proc sql number;
select Name, Age_Group, Type
   from mc1.customers
   where upcase(Type) contains "&type";
quit;
title;
%mend customers;

%customers(High Activity)
```

HIGH ACTIVITY Customers

Row	Customer Name	Customer Age Group	Type
1	Albert Collet	61-75 years	Gold high activity
2	Federico Gayo	46-60 years	Gold high activity
3	Jules Morton	15-30 years	Gold high activity
458	Lian Jarkowska	61-75 years	Club high activity
459	Stepán Astals Coma	46-60 years	Club high activity
460	Juan Marcos Iglesias López	46-60 years	Gold high activity
461	Daniel Pallen	46-60 years	Gold high activity

```
%macro customers(type=inactive);
%let type=%upcase(&type);
title "&type Customers";
proc sql number;
select Name, Age_Group, Type
   from mc1.customers
   where upcase(Type) contains "&type";
quit;
title;
%mend customers;
```

```
%customers(type=High Activity)
```

HIGH ACTIVITY Customers

Row	Customer Name	Customer Age Group	Type
1	Albert Collet	61-75 years	Gold high activity
2	Federico Gayo	46-60 years	Gold high activity
3	Jules Morton	15-30 years	Gold high activity

```
%customers()
```

INACTIVE Customers

Row	Customer Name	Customer Age Group	Type
1	Stephen Ralham	31-45 years	Club inactive
2	Micheline Giangrande	31-45 years	Club inactive
3	Giuseppe Covili	31-45 years	Club inactive
206	José María Barsanti	31-45 years	Club inactive
207	José María García González	15-30 years	Club inactive
208	Corinne Lilliu	31-45 years	Club inactive
209	Patrick Cugino	46-60 years	Club inactive

/***

* Defining and Calling a Macro: Practice #2 *

***/

/*Level 2 Practice: Using a Macro to Generate PROC MEANS Code

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

to access your practice files.

Open m104p02.sas from the practices folder. Submit the program and review the results.

Verify that the mean value of Total_Retail_Price for orders submitted by customer 51 is 314.74.

Convert the code to a macro named Orderstats that accepts the keyword parameters listed below:

var for the VAR statement variable list

class for the CLASS statement variable list

stats for the statistics to be calculated

decimals for the MAXDEC= value

Set default values for all parameters so that the code reproduces the original report when called without parameter values.

Modify the PROC MEANS code to reference the parameters.

Call the Orderstats macro without passing parameter values.

Verify that the mean value of Total_Retail_Price for Customer 51 is 314.74 (the same as the first time you submitted the code).

Call the macro again, with the parameter values listed below.

var=CostPrice_Per_Unit

decimals=0

stats=mean median

class=order_type

What is the median value for Order_type 3?

Note: Type a numeric value for your answer.

*/

```
options nolabel;
title 'Order Stats';
proc means data=mc1.orders maxdec=2 mean;
        var Total_Retail_Price;
        class Customer_ID;
run;
title;
options label;
```

Order Stats

The MEANS Procedure

Analysis Variable : Total_Retail_Price		
Customer_ID	N Obs	Mean
1	27	137.17
51	12	314.74
101	3	148.63

```
%macro Orderstats(var=Total_Retail_Price, class=Customer_ID, stats=mean, decimals=2);
options nolabel;
title 'Order Stats';
proc means data=mc1.orders maxdec=&decimals &stats;
        var &var;
        class &class;
```

```
run;

title;

options label;

%mend Orderstats;
```

%Orderstats

Order Stats

The MEANS Procedure

Analysis Variable : Total_Retail_Price		
Customer_ID	N Obs	Mean
1	27	137.17
51	12	314.74
101	3	148.63

%Orderstats(var=CostPrice_Per_Unit, decimals=0, stats=mean median, class=order_type)

Order Stats

The MEANS Procedure

Analysis Variable : CostPrice_Per_Unit			
Order_Type	N Obs	Mean	Median
1	11133	38	25
2	2375	43	29
3	2334	40	26

```
73
************************************************************;
 74         *  Activity 4.04                                  *;
 75         *  1) Notice that the %LET statement outside the %Test  *;
 76         *     macro program sets the value of X to OutsideMacro.*;
 77         *     Then the %LET statement inside the %Test macro    *;
 78         *     program sets the value of X to InsideMacro.       *;
 79        *  2) Submit the program and examine the log. What is the*;
 80         *     value of X before, during, and after macro        *;
 81         *     execution?                                        *;
 82
************************************************************;
 83
 84         /* Assign a value to X & write to log */
```

```
 85         %let X=OutsideMacro;
 86         %put NOTE: &=X before TEST macro execution.;
NOTE: X=OutsideMacro before TEST macro execution.
 87
 88         /* Define the macro test */
 89         %macro test;
 90             %let x=InsideMacro;
 91             %put NOTE: &=X during TEST macro execution.;
 92         %mend;
 93
 94         /* Execute the test macro */
 95         %test
NOTE: X=InsideMacro during TEST macro execution.
 96
 97         /* Check the value to X after execution */
 98         %put NOTE: &=X after TEST macro execution.;
NOTE: X=InsideMacro after TEST macro execution.


73
************************************************************;
 74         *  Activity 4.06                                           *;
 75         *  1) Review the program. Notice that the %Test macro      *;
 76         *     includes a parameter named X. X is assigned          *;
 77         *     different values before and during the macro call.*;
 78         *  2) Submit the program and examine the log to view the*;
 79         *     value of X before, during, and after macro           *;
 80         *     execution.                                           *;
 81         *  3) What is the scope of the X parameter in the %Test *;
 82         *     macro?                                               *;
 83         *  4) Add a %GLOBAL statement before the %PUT statement *;
 84        *     in the %Test macro definition to explicitly declare*;
 85         *     X as global scope. Does the program run              *;
 86         *     successfully?                                        *;
 87
************************************************************;
 88
 89          /* Assign a value to X & write to log */
 90         %let X=OutsideMacro;
 91         %put NOTE: &=X before TEST macro execution.;
NOTE: X=OutsideMacro before TEST macro execution.
 92
 93          /* Define the macro test */
 94         %macro test(X);
 95         %put NOTE: &=X during TEST macro execution.;
 96         %mend;
```

```
 97
 98          /* Execute the test macro */
 99         %test(InsideMacro)
NOTE: X=InsideMacro during TEST macro execution.
100
101         /* Check the value to X after execution */
102        %put NOTE: &=X after TEST macro execution.;
NOTE: X=OutsideMacro after TEST macro execution.
```

```
/**********************************
* Macro Variable Scope: Practice #5 *
**********************************/
```

/*Level 2 Practice: Controlling Macro Variable Scope

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

Open m104p05.sas from the practices folder.

Notice that macro variables are created inside the Scope macro using %LET statements.

Highlight the code in Section 1, run the selection, and review the log.

Note: The log labels global macro variables as GLOBAL and local macro variables with the name of the macro.

In which symbol table were the macro variables created?

Note: For each question in this practice, type either Local or Global for your answer.

*/

```
 /* Section 1: Macro variables created with %LET */
%macro scope;
%let stormtype1=Some damage;
%let stormtype2=Extensive damage;
%let stormtype3=Devastating damage;
%let stormtype4=Catastrophic damage;
%let stormtype5=Widespread catastrophic damage;
%put _user_;
%mend scope;
```

%scope

```
SCOPE STORMTYPE1 Some damage
SCOPE STORMTYPE2 Extensive damage
SCOPE STORMTYPE3 Devastating damage
SCOPE STORMTYPE4 Catastrophic damage
SCOPE STORMTYPE5 Widespread catastrophic damage
```

/*Find Section 2 in the program. Notice that macro variables are created inside the Scope macro using the SQL INTO clause.

Highlight the code in Section 2, run the selection, and review the log.

In which symbol table were the macro variables created? */

```
 /* Section 2: Macro variables created with SQL INTO */

%macro scope;

proc sql noprint;

select damage into :stormtype1-

        from mc1.storm_cat

        order by category;

quit;


%let num = &sqlobs;

%put _user_;

%mend scope;
```

%scope

```
SCOPE STORMTYPE1 Some damage
SCOPE STORMTYPE2 Extensive damage
SCOPE STORMTYPE3 Devastating damage
SCOPE STORMTYPE4 Catastrophic damage
SCOPE STORMTYPE5 Widespread catastrophic damage
```

/*Find Section 3 in the program.

Notice that macro variables are created inside the Scope macro using the CALL SYMPUTX routine in a DATA step.

Highlight the code in Section 3, being sure to include the %SYMDEL statement. Run the selection and review the log.

In which symbol table were the macro variables created?*/

```
 /* Section 3: Macro variables created with CALL SYMPUTX */

%macro scope;

data _null_;

        set mc1.storm_cat end=last;

        call symputx(cat('stormtype',_n_),Damage);

        if last=1 then

                call symputx('num',_n_);

run;

%put _user_;

%mend scope;


%scope


%symdel stormtype1 stormtype2 stormtype3 stormtype4 stormtype5 num;
```

```
 GLOBAL STORMTYPE1 Some damage
 GLOBAL STORMTYPE2 Extensive damage
 GLOBAL STORMTYPE3 Devastating damage
 GLOBAL STORMTYPE4 Catastrophic damage
 GLOBAL STORMTYPE5 Widespread catastrophic damage
```

/*Add a %LOCAL statement before the DATA step to define a macro variable named x.

Highlight the code in Section 3, run the selection, and review the log.

In which symbol table were the macro variables created?

```
 Section 4: Macro variables created with CALL SYMPUTX */

%macro scope;

%local x;
```

```
data _null_;
        set mc1.storm_cat end=last;
        call symputx(cat('stormtype',_n_),Damage);
        if last=1 then
                call symputx('num',_n_);
run;
%put _user_;
%mend scope;
```

%scope

```
SCOPE NUM 5
SCOPE STORMTYPE1 Some damage
SCOPE STORMTYPE2 Extensive damage
SCOPE STORMTYPE3 Devastating damage
SCOPE STORMTYPE4 Catastrophic damage
SCOPE STORMTYPE5 Widespread catastrophic damage
SCOPE X
```

/*Delete the %LOCAL statement and modify the CALL SYMPUTX statement to specify macro variables should be created in the local symbol table.

Highlight the code in Section 3, run the selection, and review the log.

Note: The %SYMDEL statement generates warnings because the macro variables listed are not currently in the global symbol table.

In which symbol table were the macro variables created?

Section 4: Macro variables created with CALL SYMPUTX */

```
%macro scope;
data _null_;
        set mc1.storm_cat end=last;
        call symputx(cat('stormtype',_n_),Damage,'l');
        if last=1 then
                call symputx('num',_n_,'l');
run;
```

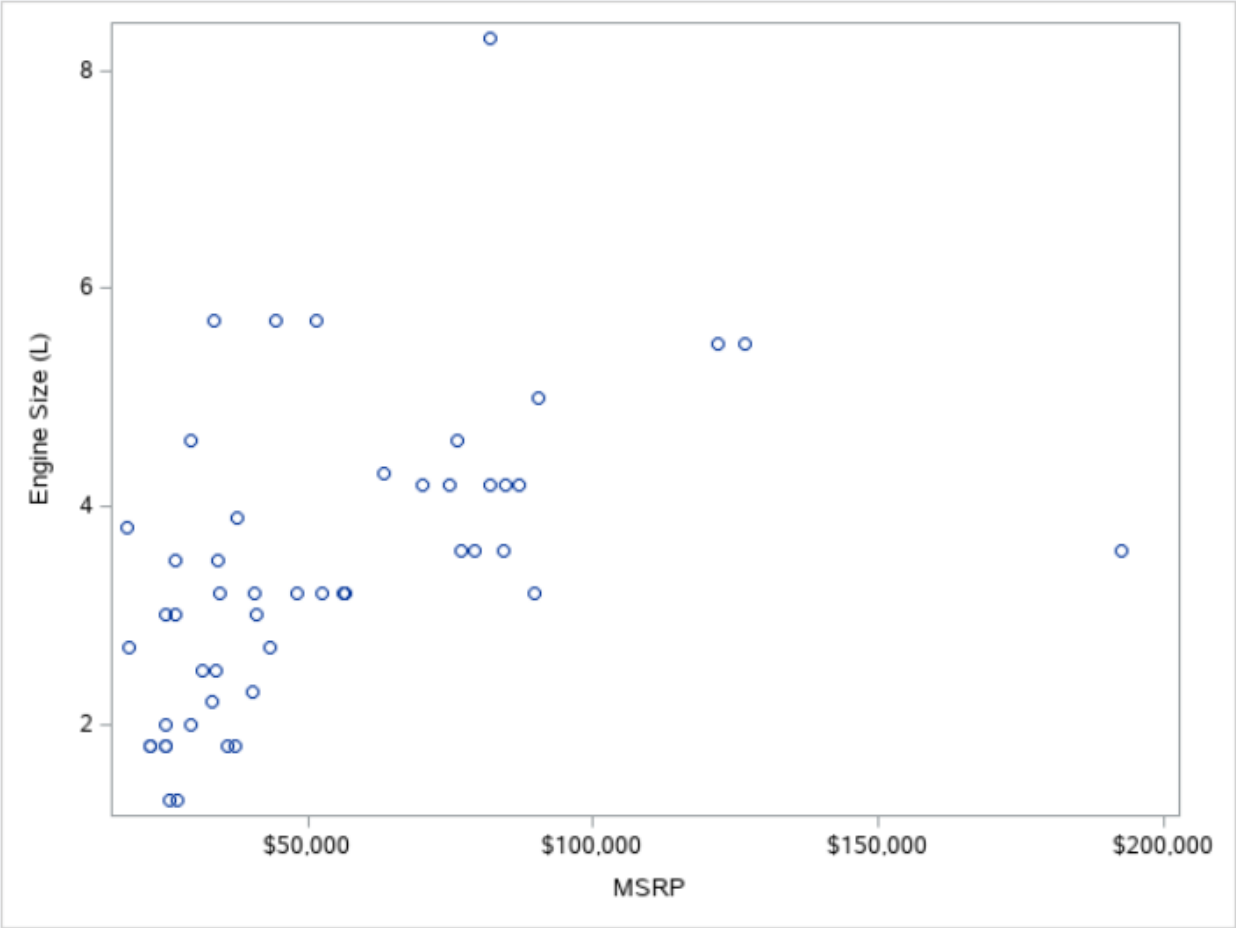
```
%put _user_;

%mend scope;


%scope


%symdel stormtype1 stormtype2 stormtype3 stormtype4 stormtype5 num;

 SCOPE NUM 5
 SCOPE STORMTYPE1 Some damage
 SCOPE STORMTYPE2 Extensive damage
 SCOPE STORMTYPE3 Devastating damage
 SCOPE STORMTYPE4 Catastrophic damage
 SCOPE STORMTYPE5 Widespread catastrophic damage


**********************************************************;

*  Activity 4.07                          *;

*  1) Notice that there is a missing semicolon after the  *;

*     DATA statement, which causes a syntax error. If     *;

*     there is a syntax error in the first DATA step,     *;

*     then the %PUT statement should execute and write a  *;

*     custom error message to the log, and the remaining  *;

*     PROC steps should not execute.                      *;

*  2) Run the program and look at the log. Notice that    *;

*     although there was an error in the DATA step, SAS   *;

*     still attempted to execute PROC PRINT and PROC      *;

*     SGPLOT.                                             *;

*  3) Identify the syntax error in the macro %IF         *;

*     statement. Fix the error and run the program again  *;

*     to confirm that the %PUT statement is executed.     *;

**********************************************************;


data sports;
```

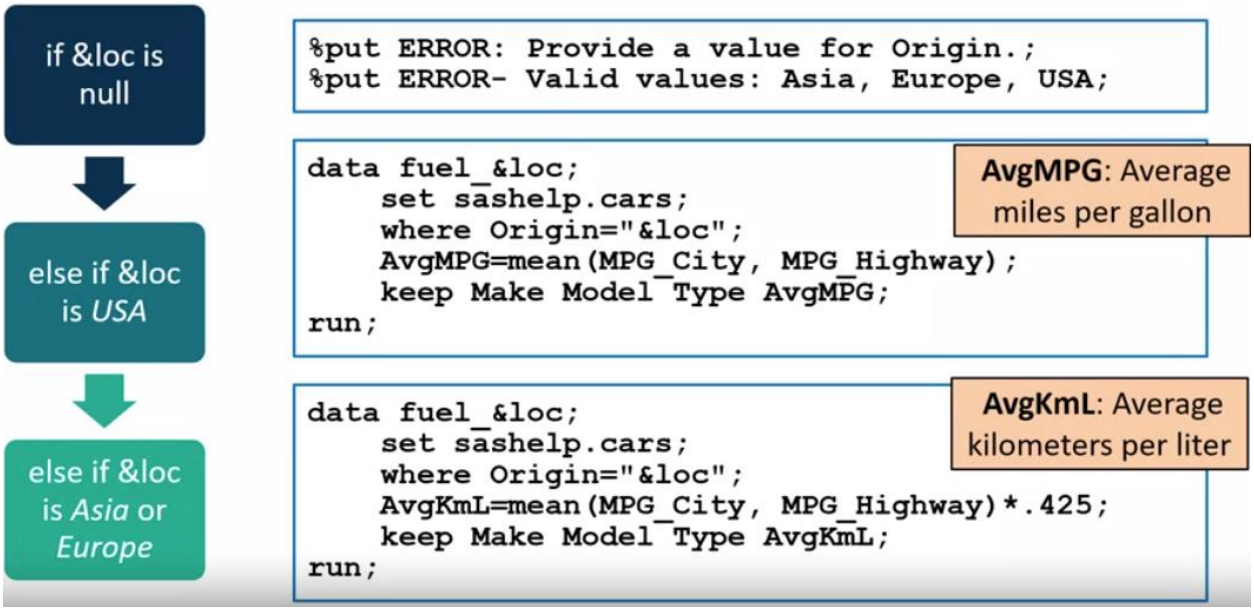
```
   set sashelp.cars;
   where Type="Sports";
   AvgMPG=mean(MPG_City, MPG_Highway);
run;

%if &syserr ne 0 %then %do;
   %put ERROR: The rest of the program will not execute;
%end;
%else %do;
          title "Sports Cars";
          proc print data=sports noobs;
             var Make Model AvgMPG MSRP EngineSize;
          run;
          title;
          proc sgplot data=sports;
             scatter x=MSRP y=EngineSize;
          run;
%end;
title;
```

Sports Cars

Make	Model	AvgMPG	MSRP	EngineSize
Acura	NSX coupe 2dr manual S	20.5	$89,765	3.2
Audi	RS 6 4dr	18.5	$84,600	4.2
Audi	TT 1.8 convertible 2dr (coupe)	24.0	$35,940	1.8
Audi	TT 1.8 Quattro 2dr (convertible)	24.0	$37,390	1.8
Audi	TT 3.2 coupe 2dr (convertible)	25.0	$40,590	3.2
BMW	M3 coupe 2dr	20.0	$48,195	3.2
BMW	M3 convertible 2dr	19.5	$56,595	3.2
BMW	Z4 convertible 2.5i 2dr	24.0	$33,895	2.5
BMW	Z4 convertible 3.0i 2dr	25.0	$41,045	3.0
Cadillac	XLR convertible 2dr	21.0	$76,200	4.6
Chevrolet	Corvette 2dr	21.5	$44,535	5.7
Chevrolet	Corvette convertible 2dr	21.5	$51,535	5.7
Chrysler	Crossfire 2dr	21.0	$34,495	3.2
Dodge	Viper SRT-10 convertible 2dr	16.0	$81,795	8.3
Ford	Mustang 2dr (convertible)	24.5	$18,345	3.8
Ford	Mustang GT Premium convertible 2dr	21.0	$29,380	4.6
Ford	Thunderbird Deluxe convert w/hardtop 2d	20.5	$37,530	3.9
Honda	S2000 convertible 2dr	22.5	$33,260	2.2
Hyundai	Tiburon GT V6 2dr	22.5	$18,739	2.7
Jaguar	XK8 coupe 2dr	22.0	$69,995	4.2

Porsche	Boxster S convertible 2dr	22.0	$52,365	3.2
Subaru	Impreza WRX 4dr	23.5	$25,045	2.0
Subaru	Impreza WRX STi 4dr	21.0	$31,545	2.5
Toyota	Celica GT-S 2dr	28.5	$22,570	1.8
Toyota	MR2 Spyder convertible 2dr	29.0	$25,130	1.8

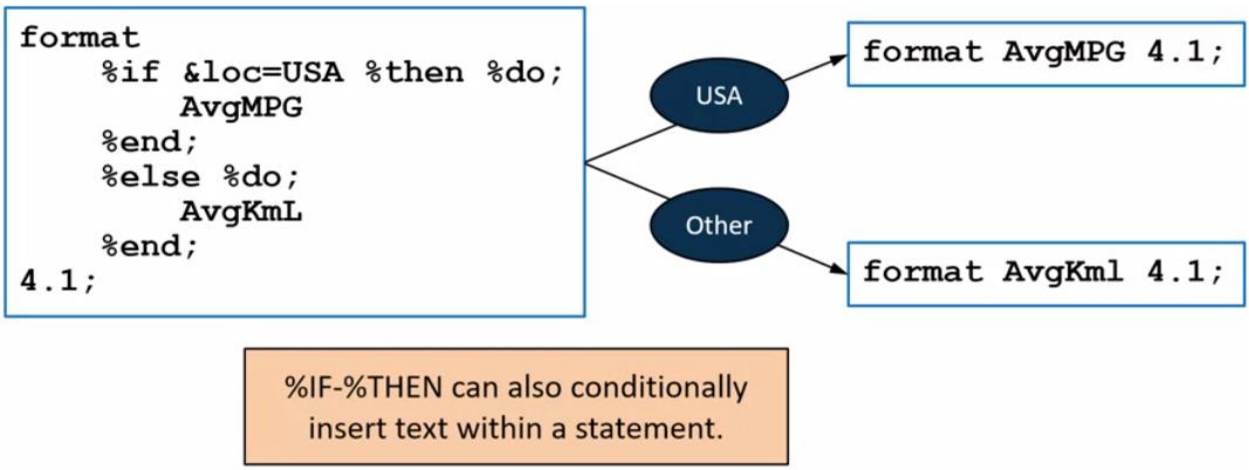


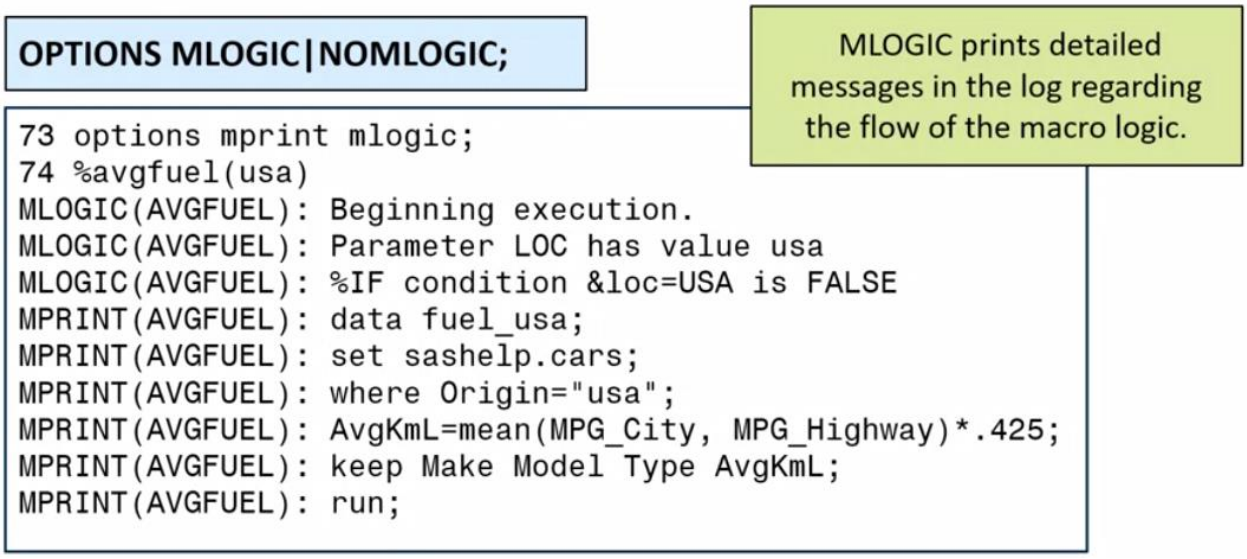



```
data fuel_&loc;
    set sashelp.cars;
    where upcase(Origin)="&loc";
    %if &loc=USA %then %do;
        AvgMPG=mean(MPG_City, MPG_Highway);
        keep Make Model Type AvgMPG;
    %end;
    %else %do;
        AvgKmL=mean(MPG_City, MPG_Highway)*.425;
        keep Make Model Type AvgKmL;
    %end;
run;
```

OR

%IF-%THEN logic can conditionally generate statements within a step.






```
*************************************************************;
*  Activity 4.08                                            *;
*  1) Review the program and notice that different DATA     *;
*     steps run, depending on the value of the loc          *;
*     parameter. The %AvgFuel macro call at the end of      *;
*     the program assigns Europe to the loc parameter.      *;
*  2) Run the program and view the log. Notice the MPRINT   *;
*     messages displaying the final program that was        *;
*     compiled and executed.                                *;
*  3) Run the %AvgFuel macro call with usa as the           *;
*     parameter value. View the log. Which DATA step was    *;
*     compiled and executed?                                *;
*************************************************************;

%macro avgfuel(loc);
%if &loc= %then %do;
    %put ERROR: Provide a value for Origin.;
```

```
    %put ERROR- Valid values: Asia, Europe, USA;
    %return;
%end;

%else %if &loc=USA %then %do;
      data fuel_&loc;
            set sashelp.cars;
            where Origin="&loc";
            AvgMPG=mean(MPG_City, MPG_Highway);
            keep Make Model Type AvgMPG;
      run;
%end;

%else %do;
      data fuel_&loc;
            set sashelp.cars;
            where Origin="&loc";
            AvgKmL=mean(MPG_City, MPG_Highway)*.425;
            keep Make Model Type AvgKmL;
      run;
%end;

title1 "Fuel Efficiency";
title2 "Origin: &loc";
proc print data=fuel_&loc;
run;
title;
%mend avgfuel;

options mprint mlogic;
%avgfuel(Europe)

options nomprint nomlogic;

MLOGIC(AVGFUEL):  Beginning execution.
 MLOGIC(AVGFUEL):  Parameter LOC has value Europe
 MLOGIC(AVGFUEL):  %IF condition &loc= is FALSE
 MLOGIC(AVGFUEL):  %IF condition &loc=USA is FALSE
 MPRINT(AVGFUEL):   data fuel_Europe;
 MPRINT(AVGFUEL):   set sashelp.cars;
 MPRINT(AVGFUEL):   where Origin="Europe";
 MPRINT(AVGFUEL):   AvgKmL=mean(MPG_City, MPG_Highway)*.425;
 MPRINT(AVGFUEL):   keep Make Model Type AvgKmL;
 MPRINT(AVGFUEL):   run;
```

```
 NOTE: There were 123 observations read from the data set
SASHELP.CARS.
       WHERE Origin='Europe';
 NOTE: The data set WORK.FUEL_EUROPE has 123 observations and 4
variables.
 MPRINT(AVGFUEL):   title1 "Fuel Efficiency";
 MPRINT(AVGFUEL):   title2 "Origin: Europe";
 MPRINT(AVGFUEL):   proc print data=fuel_Europe;
 MPRINT(AVGFUEL):   run;

 NOTE: There were 123 observations read from the data set
WORK.FUEL_EUROPE.

 MPRINT(AVGFUEL):   title;
 MLOGIC(AVGFUEL):  Ending execution.
```

Fuel Efficiency
Origin: Europe

Obs	Make	Model	Type	AvgKmL
1	Audi	A4 1.8T 4dr	Sedan	11.2625
2	Audi	A41.8T convertible 2dr	Sedan	11.2625
3	Audi	A4 3.0 4dr	Sedan	10.2000
4	Audi	A4 3.0 Quattro 4dr manual	Sedan	9.1375
5	Audi	A4 3.0 Quattro 4dr auto	Sedan	9.1375
6	Audi	A6 3.0 4dr	Sedan	9.9875
7	Audi	A6 3.0 Quattro 4dr	Sedan	9.1375
8	Audi	A4 3.0 convertible 2dr	Sedan	9.9875
9	Audi	A4 3.0 Quattro convertible 2dr	Sedan	9.1375
10	Audi	A6 2.7 Turbo Quattro 4dr	Sedan	9.1375
11	Audi	A6 4.2 Quattro 4dr	Sedan	8.7125

```
%macro avgfuel(loc);

%if &loc= %then %do;

   %put ERROR: Provide a value for Origin.;

   %put ERROR- Valid values: Asia, Europe, USA;

   %return;

%end;


%else %if &loc=USA %then %do;
```

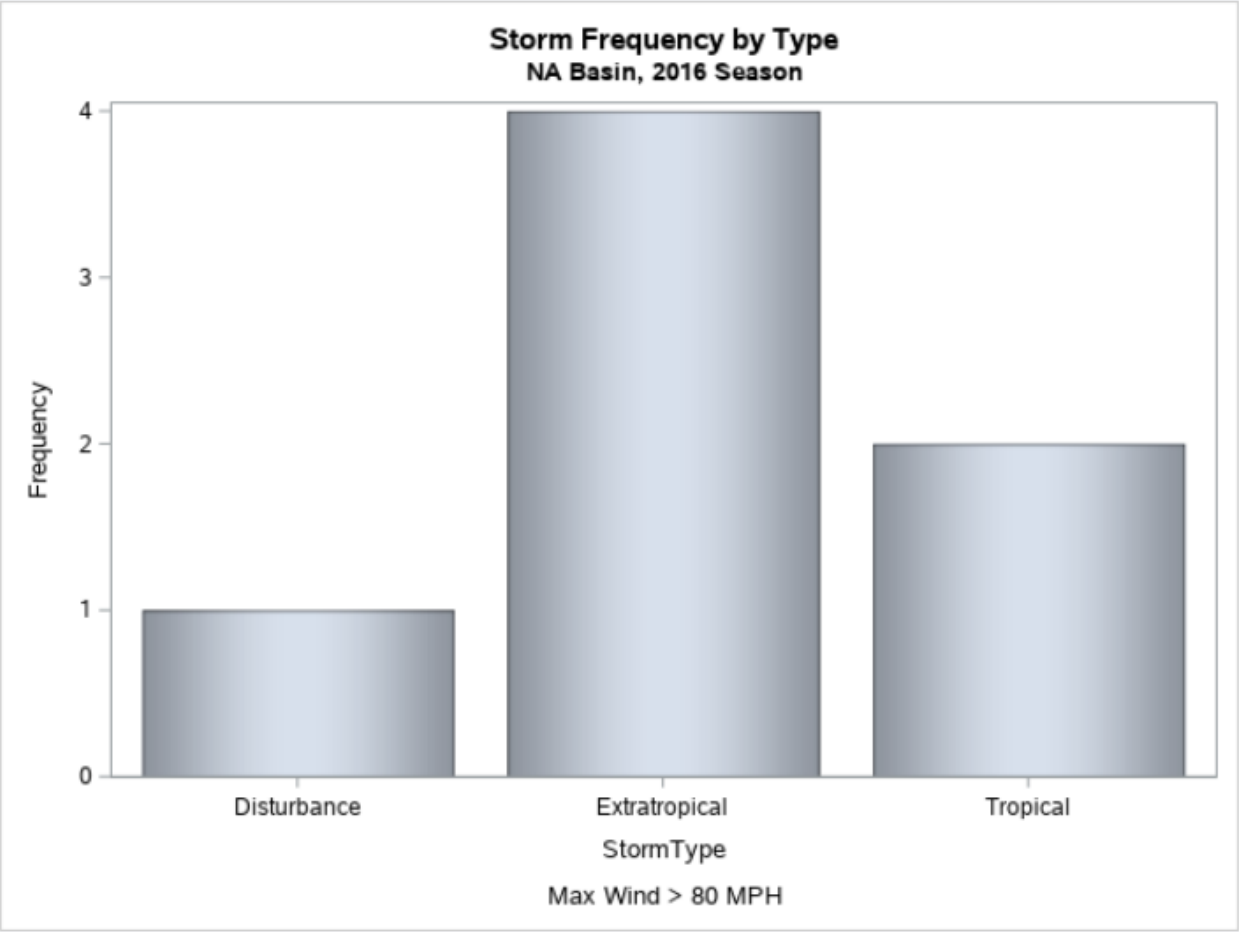
```
        data fuel_&loc;
                set sashelp.cars;
                where Origin="&loc";
                AvgMPG=mean(MPG_City, MPG_Highway);
                keep Make Model Type AvgMPG;
        run;
%end;

%else %do;
        data fuel_&loc;
                set sashelp.cars;
                where Origin="&loc";
                AvgKmL=mean(MPG_City, MPG_Highway)*.425;
                keep Make Model Type AvgKmL;
        run;
%end;

title1 "Fuel Efficiency";
title2 "Origin: &loc";
proc print data=fuel_&loc;
run;
title;
%mend avgfuel;

options mprint mlogic;

%avgfuel(USA)

options nomprint nomlogic;
```

```
MLOGIC(AVGFUEL):  Beginning execution.
 MLOGIC(AVGFUEL):  Parameter LOC has value USA
 MLOGIC(AVGFUEL):  %IF condition &loc= is FALSE
 MLOGIC(AVGFUEL):  %IF condition &loc=USA is TRUE
 MPRINT(AVGFUEL):   data fuel_USA;
 MPRINT(AVGFUEL):   set sashelp.cars;
 MPRINT(AVGFUEL):   where Origin="USA";
 MPRINT(AVGFUEL):   AvgMPG=mean(MPG_City, MPG_Highway);
 MPRINT(AVGFUEL):   keep Make Model Type AvgMPG;
 MPRINT(AVGFUEL):   run;

 NOTE: There were 147 observations read from the data set
SASHELP.CARS.
       WHERE Origin='USA';
 NOTE: The data set WORK.FUEL_USA has 147 observations and 4
variables.


MPRINT(AVGFUEL):   title1 "Fuel Efficiency";
 MPRINT(AVGFUEL):   title2 "Origin: USA";
 MPRINT(AVGFUEL):   proc print data=fuel_USA;
 MPRINT(AVGFUEL):   run;

 NOTE: There were 147 observations read from the data set
WORK.FUEL_USA.

 MPRINT(AVGFUEL):   title;
 MLOGIC(AVGFUEL):  Ending execution.
```

Fuel Efficiency
Origin: USA

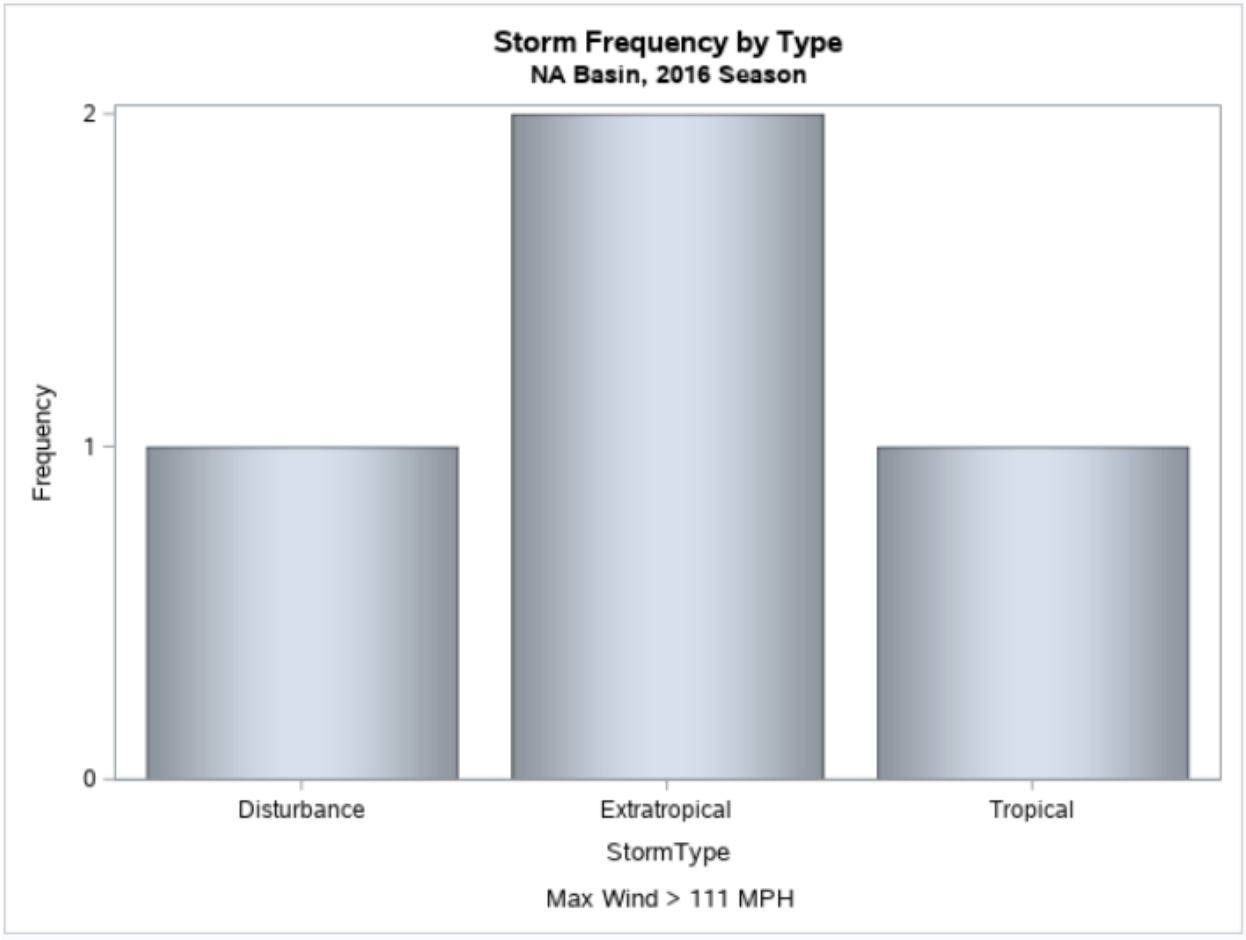
Obs	Make	Model	Type	AvgMPG
1	Buick	Rainier	SUV	18.0
2	Buick	Rendezvous CX	SUV	22.5
3	Buick	Century Custom 4dr	Sedan	25.0
4	Buick	LeSabre Custom 4dr	Sedan	24.5
5	Buick	Regal LS 4dr	Sedan	25.0
6	Buick	Regal GS 4dr	Sedan	23.0
7	Buick	LeSabre Limited 4dr	Sedan	24.5
8	Buick	Park Avenue 4dr	Sedan	24.5
9	Buick	Park Avenue Ultra 4dr	Sedan	23.0
10	Cadillac	Escalade	SUV	16.0

```
/*****************************
* Conditional Processing: Demo *
*****************************/

%macro stormchart(basin, season, maxwind);
title1 "Storm Frequency by Type";
title2 "&basin Basin, &season Season";
footnote "Max Wind > &maxwind MPH";
proc sgplot data=mc1.storm_final;
   vbar StormType / dataskin=pressed;
   where Basin="&basin" and Season=&season and MaxWindMPH>&maxwind;
run;
title;footnote;
%mend stormchart;

options mcompilenote=all;
%stormchart(NA,2016,80)
```




```
%macro stormchart(basin, season, cat);
%local maxwind;
%if &cat=5 %then %let maxwind=157;
%else %if &cat=4 %then %let maxwind=130;
%else %if &cat=3 %then %let maxwind=111;
%else %if &cat=2 %then %let maxwind=96;
%else %if &cat=1 %then %let maxwind=74;

title1 "Storm Frequency by Type";
title2 "&basin Basin, &season Season";
footnote "Max Wind > &maxwind MPH";
proc sgplot data=mc1.storm_final;
   vbar StormType / dataskin=pressed;
```

```
   where Basin="&basin" and Season=&season and MaxWindMPH>&maxwind;
run;
title;footnote;
%mend stormchart;


options mcompilenote=all;
%stormchart(NA,2016,3)
NOTE: The macro STORMCHART completed compilation without errors.
      60 instructions 1520 bytes.

NOTE: There were 4 observations read from the data set
MC1.STORM_FINAL.
      WHERE (Basin='NA') and (Season=2016) and (MaxWindMPH>111);
```

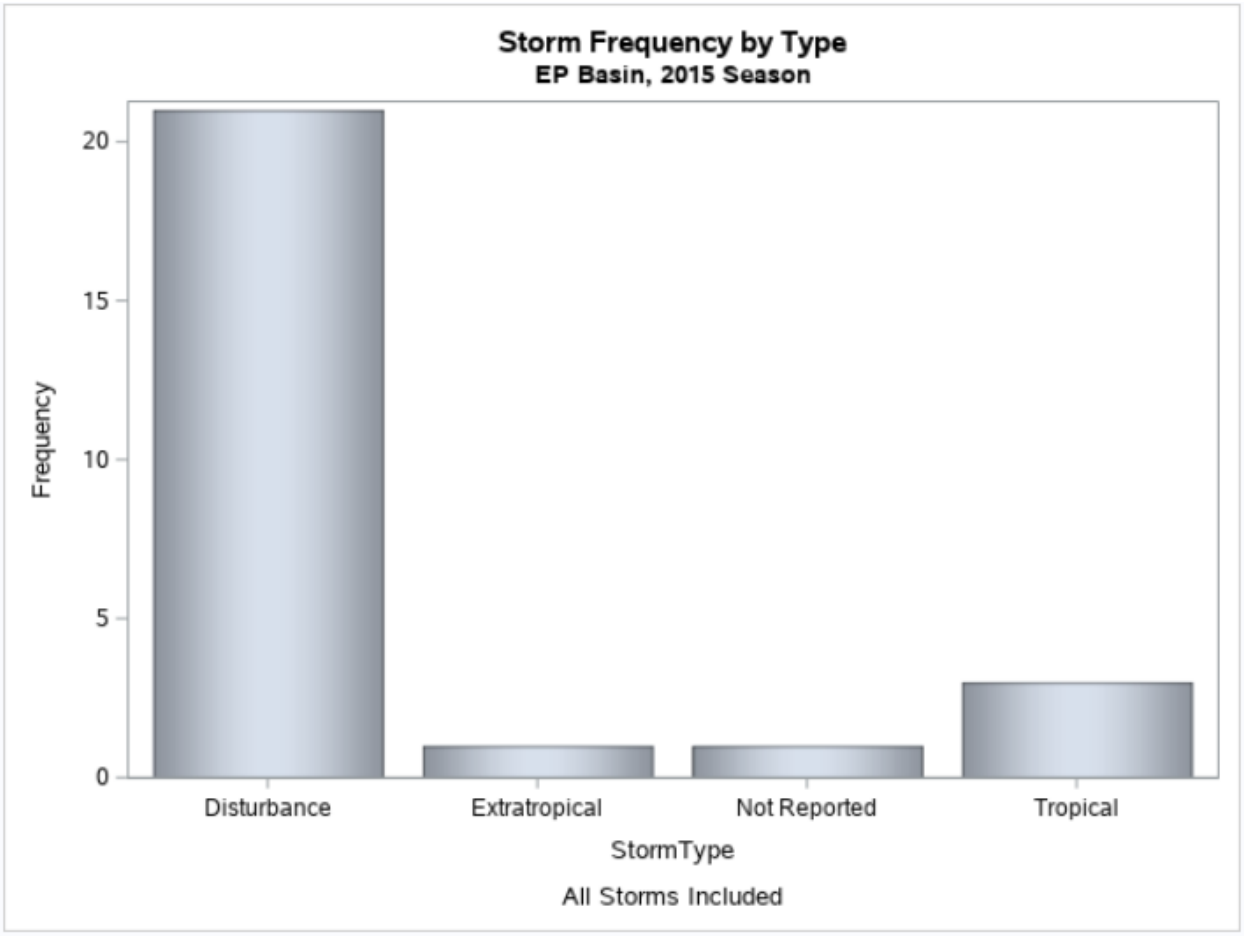



```
%macro stormchart(basin, season, cat);
%local maxwind;
```

```
%if &cat=5 %then %let maxwind=157;
%else %if &cat=4 %then %let maxwind=130;
%else %if &cat=3 %then %let maxwind=111;
%else %if &cat=2 %then %let maxwind=96;
%else %if &cat=1 %then %let maxwind=74;

title1 "Storm Frequency by Type";
title2 "&basin Basin, &season Season";
%if &cat ne  %then %do;
        footnote "Max Wind > &maxwind MPH";
%end;
%else %do;
        footnote "All Storms Included";
%end;

proc sgplot data=mc1.storm_final;
   vbar StormType / dataskin=pressed;
   where Basin="&basin" and Season=&season
   %if &cat ne  %then %do;
        and MaxWindMPH>&maxwind;
   %end;
;
run;
title;footnote;
%mend stormchart;

options mcompilenote=all mlogic mprint;
%stormchart(EP,2015)
NOTE: The macro STORMCHART completed compilation without errors.
```

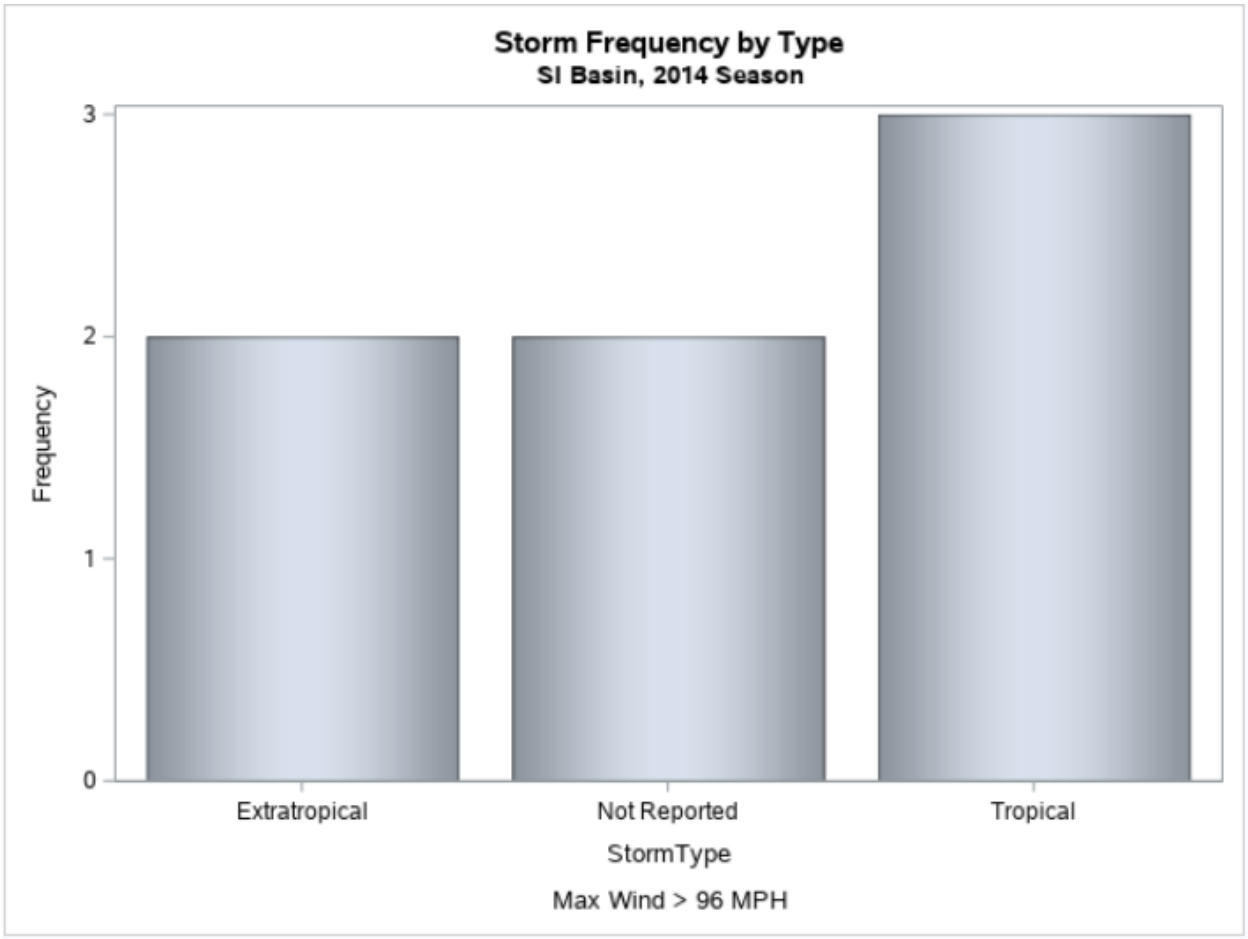
```
      84 instructions 2000 bytes.

MLOGIC(STORMCHART):  Beginning execution.
MLOGIC(STORMCHART):  Parameter BASIN has value EP
MLOGIC(STORMCHART):  Parameter SEASON has value 2015
MLOGIC(STORMCHART):  Parameter CAT has value
MLOGIC(STORMCHART):  %LOCAL  MAXWIND
MLOGIC(STORMCHART):  %IF condition &cat=5 is FALSE
MLOGIC(STORMCHART):  %IF condition &cat=4 is FALSE
MLOGIC(STORMCHART):  %IF condition &cat=3 is FALSE
MLOGIC(STORMCHART):  %IF condition &cat=2 is FALSE
MLOGIC(STORMCHART):  %IF condition &cat=1 is FALSE
MPRINT(STORMCHART):   title1 "Storm Frequency by Type";
MPRINT(STORMCHART):   title2 "EP Basin, 2015 Season";
MLOGIC(STORMCHART):  %IF condition &cat ne is FALSE
MPRINT(STORMCHART):   footnote "All Storms Included";
MPRINT(STORMCHART):   proc sgplot data=mc1.storm_final;
MPRINT(STORMCHART):   vbar StormType / dataskin=pressed;
MLOGIC(STORMCHART):  %IF condition &cat ne is FALSE
MPRINT(STORMCHART):   where Basin="EP" and Season=2015 ;
MPRINT(STORMCHART):   run;
```

```
NOTE: There were 26 observations read from the data set
MC1.STORM_FINAL.
      WHERE (Basin='EP') and (Season=2015);

MPRINT(STORMCHART):   title;
MPRINT(STORMCHART):  footnote;
MLOGIC(STORMCHART):  Ending execution.
```



%stormchart(SI,2014,2)

options nomlogic nomprint;



Columns

- Select all
- ID
- Type
- Group
- Country
- Name
- Age_Group
- Age
- BirthDate

Total rows: 1800 Total columns: 8 Rows 1-100

	ID	Type	Group	Country	Name	Age_Group
1	1	Gold high activity	Orion Club Gold members	FR	Albert Collet	61-75 years
2	51	No member status	No membership	FR	Jean-Claude Nogues	31-45 years
3	101	Club medium activity	Orion Club members	FR	Christine Herfroy	61-75 years
4	151	Club medium activity	Orion Club members	GB	Alex Hammersly	31-45 years
5	201	Gold low activity	Orion Club Gold members	US	Nogarji Mcadam	61-75 years
6	251	Gold medium activity	Orion Club Gold members	FR	Laurent Ollivon	15-30 years
7	301	Gold low activity	Orion Club Gold members	BE	Anneleen Vanlangenaeker	46-60 years
8	351	Gold high activity	Orion Club Gold members	ES	Federico Gayo	46-60 years
9	401	Gold medium activity	Orion Club Gold members	FR	Bruno Chauprade	46-60 years

/**********

* Conditional Processing: Practice #7 *

**********/

/*Level 1 Practice: Using Macro Conditional Processing to Choose Which SAS Statements are Generated

Reminder: If you restarted your SAS session,

you must submit the libname.sas program in the EMC1V2 folder to access your practice files.

Open m104p07.sas from the practices folder.

Modify the CustomerList macro to test the country parameter.

If the value is null, insert the following statements into the PROC PRINT step:

title "All Customers";

var ID Name Country Type Age_Group;

If the value is not null, insert the following statements into the PROC PRINT step:

title "Customers from Country: &country";

where Country="&country";

var ID Name Type Age_Group;

Submit the macro definition and call the macro using a null value.

Verify that the title is All Customers and that 1800 rows were included in the report.

Submit the macro call again with a value of AU for the country parameter.

How many customers are included in the report?

Note: Type a numeric value for your answer.

*/

```
%macro customerlist(country);
proc print data=mc1.customers noobs;
        %if &country ne %then %do;
                title "Customers from Country: &country";
                where Country="&country";
                var ID Name Type Age_Group;

        %end;
        %else %do;
                title "All Customers";
                var ID Name Country Type Age_Group;
```

```
        %end;
;
run;
%mend customerlist;


%customerlist
```

```
NOTE: The macro CUSTOMERLIST completed compilation without errors.
      23 instructions 624 bytes.
NOTE: There were 1800 observations read from the data set MC1.CUSTOMERS.
```

All Customers

ID	Name	Country	Type	Age_Group
1	Albert Collet	FR	Gold high activity	61-75 years
51	Jean-Claude Nogues	FR	No member status	31-45 years
101	Christine Herfroy	FR	Club medium activity	61-75 years
151	Alex Hammersly	GB	Club medium activity	31-45 years
201	Nogarji Mcadam	US	Gold low activity	61-75 years
251	Laurent Ollivon	FR	Gold medium activity	15-30 years
301	Anneleen Vanlangenaeker	BE	Gold low activity	46-60 years
351	Federico Gayo	ES	Gold high activity	46-60 years

```
%customerlist(AU)
```

```
NOTE: There were 90 observations read from the data set MC1.CUSTOMERS.
      WHERE Country='AU';
```

Customers from Country: AU

ID	Name	Type	Age_Group
2351	Kay Pachare	Gold high activity	15-30 years
2451	Meshlin Bhula	Gold high activity	15-30 years
3103	Brenda Leavey	Club medium activity	46-60 years
4704	Norlene Pinkus	No member status	46-60 years
5558	Rodger Schelleim	Gold low activity	31-45 years
6159	Pascal Filleti	Gold medium activity	61-75 years

```
/*************************************
* Conditional Processing: Practice #8 *
*************************************/

/*Level 2 Practice: Using Macro Conditional Processing in Open Code
```

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

Open m104p08.sas from the practices folder.

Run the program and verify that a table named Profit was created in the work library, and that a bar chart was produced.

Add a LIBNAME statement to create a library named Orion that points to the practices folder in the course files.

Use the path "&path/practices".

Add a %PUT statement to write the value of the automatic macro variable syslibrc to the log.

If the LIBNAME statement ran successfully, the value of syslibrc is zero. Otherwise, it is a value other than zero.

Run the LIBNAME and %PUT statements.

Modify the CREATE TABLE statement to write the Profit table to the Orion library.

Add macro conditional statements: If syslibrc is not equal to zero, then write a custom error message to the log

indicating the rest of the program will not execute because of the failure of the LIBNAME statement.

If syslibrc is equal to zero, run the PROC SQL and SGPLOT steps.

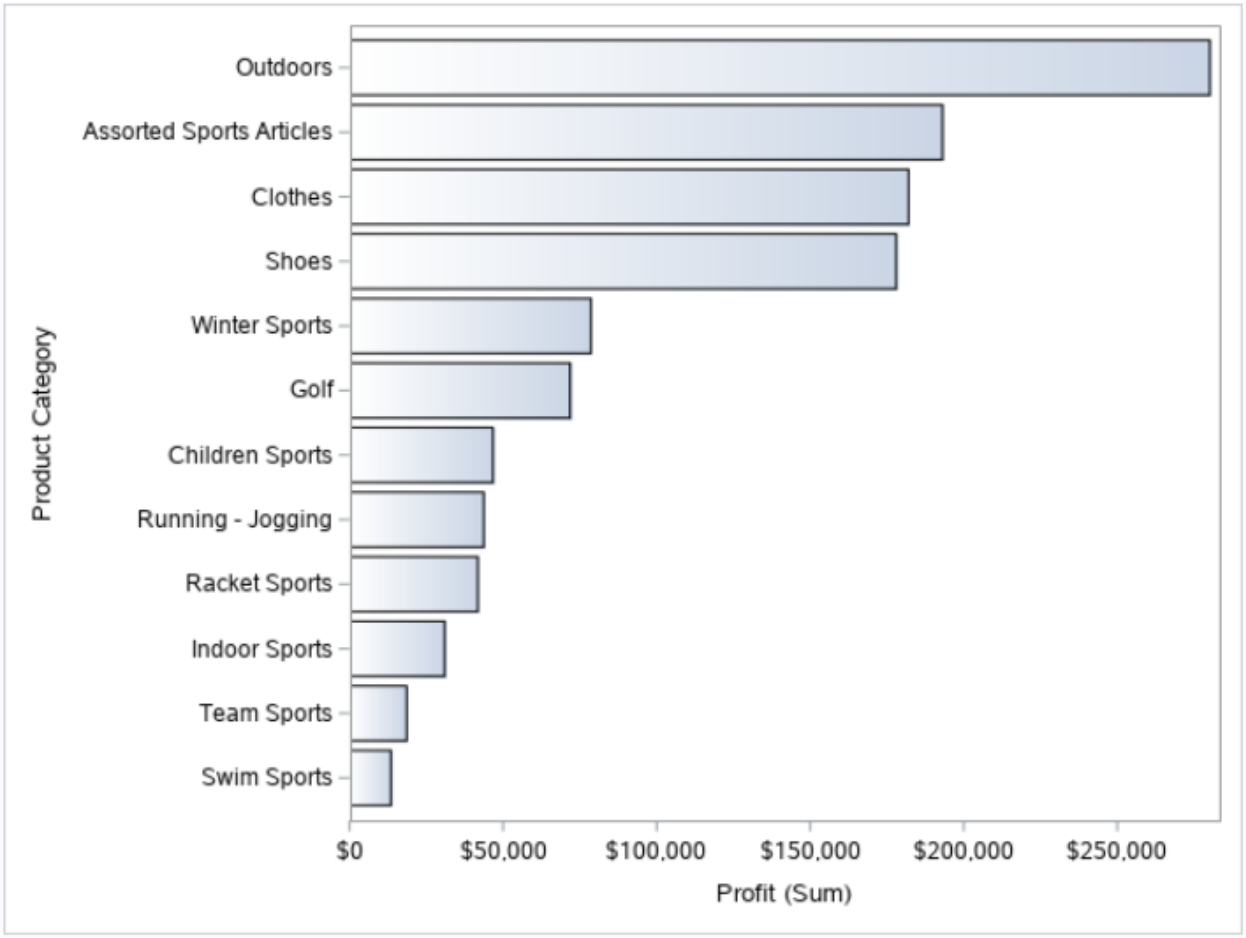
Test the program with and without an error in the LIBNAME statement.

If the graph is created successfully, which Product_Category value has the highest sum of Profit?

*/

```
proc sql;
create table profit as
select o.product_id, Product_Name, Product_Category,
   (Total_Retail_Price-(CostPrice_per_Unit*Quantity))
      as Profit format=dollar12.2
   from mc1.orders as o inner join mc1.products as p
         on o.product_id=p.product_id;
quit;
```

```
proc sgplot data=profit noautolegend;
   hbar Product_Category / response=Profit fillType=gradient categoryorder=respdesc;
run;
```




```
/*************************************
* Conditional Processing: Practice #8 *
*************************************/
```

/*Level 2 Practice: Using Macro Conditional Processing in Open Code

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

Open m104p08.sas from the practices folder.

Run the program and verify that a table named Profit was created in the work library, and that a bar chart was produced.

Add a LIBNAME statement to create a library named Orion that points to the practices folder in the course files.

Use the path "&path/practices".

Add a %PUT statement to write the value of the automatic macro variable syslibrc to the log.

If the LIBNAME statement ran successfully, the value of syslibrc is zero. Otherwise, it is a value other than zero.

Run the LIBNAME and %PUT statements.

Modify the CREATE TABLE statement to write the Profit table to the Orion library.

Add macro conditional statements: If syslibrc is not equal to zero, then write a custom error message to the log

indicating the rest of the program will not execute because of the failure of the LIBNAME statement.

If syslibrc is equal to zero, run the PROC SQL and SGPLOT steps.

Test the program with and without an error in the LIBNAME statement.

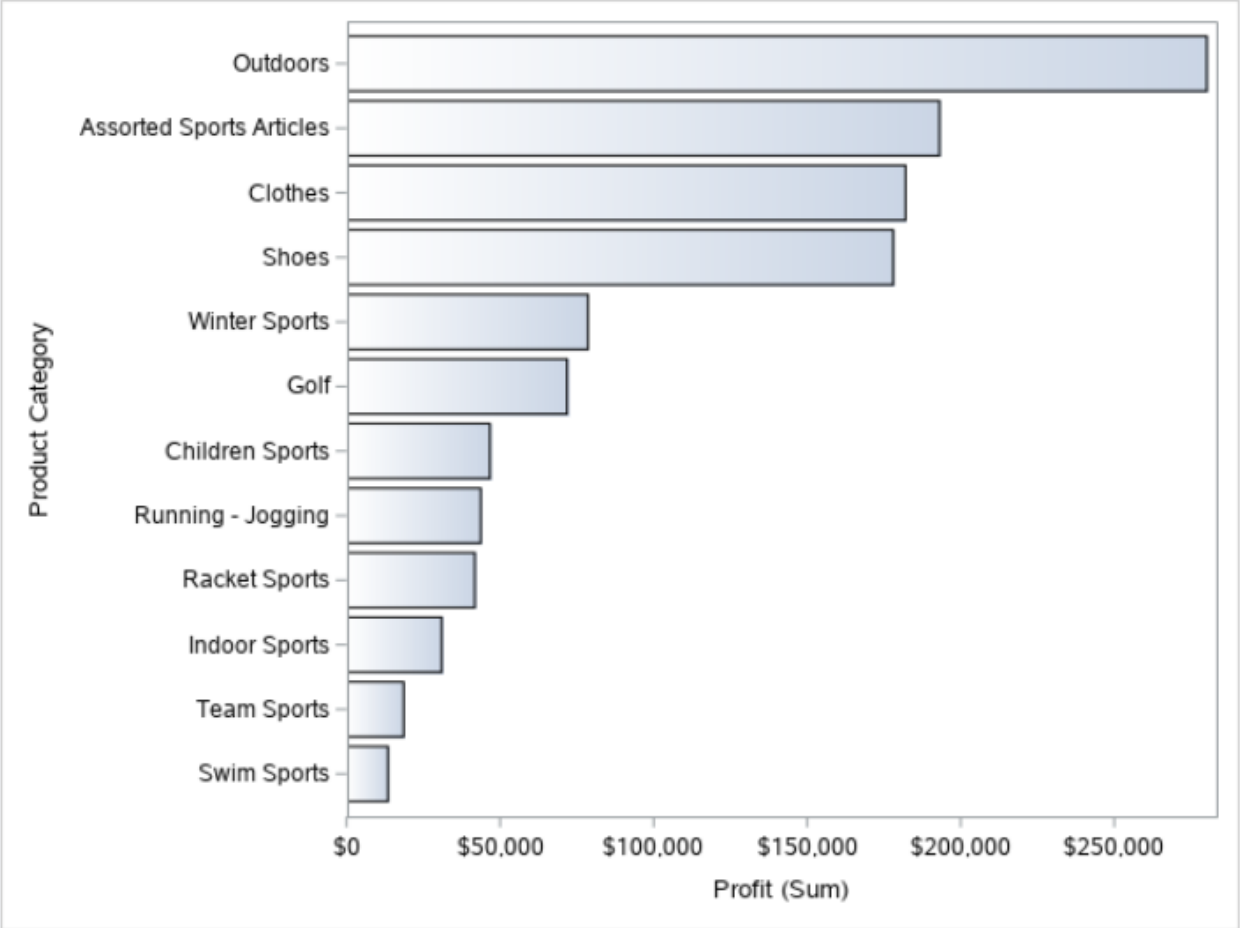
If the graph is created successfully, which Product_Category value has the highest sum of Profit?

*/

```
libname Orion "&path/practices";
%put &=syslibrc;


%if &syslibrc ne 0 %then %do;
        %put ERROR: Library Name Orion does not exist;
%end;
%else %do;
proc sql;
create table Orion.profit as
select o.product_id, Product_Name, Product_Category,
   (Total_Retail_Price-(CostPrice_per_Unit*Quantity))
      as Profit format=dollar12.2
   from mc1.orders as o inner join mc1.products as p
         on o.product_id=p.product_id;
quit;
```

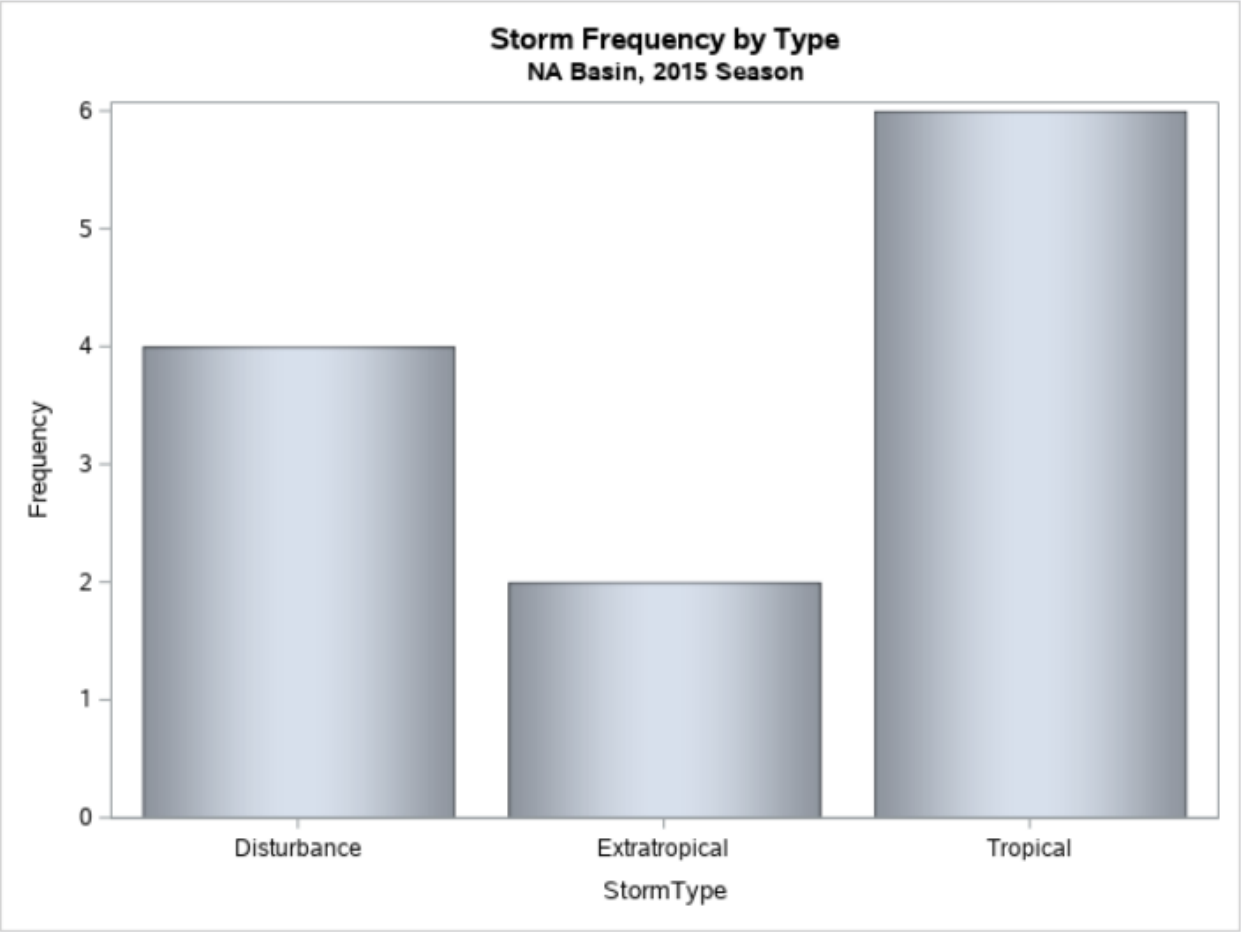
```
proc sgplot data=Orion.profit noautolegend;
   hbar Product_Category / response=Profit fillType=gradient categoryorder=respdesc;
run;
%end;
```

```
92          libname Orion "&path/practices";
NOTE: Libref ORION was successfully assigned as follows:
      Engine:        V9
      Physical Name: /home/u58304328/EMC1V2/practices
93          %put &=syslibrc;
SYSLIBRC=0
```

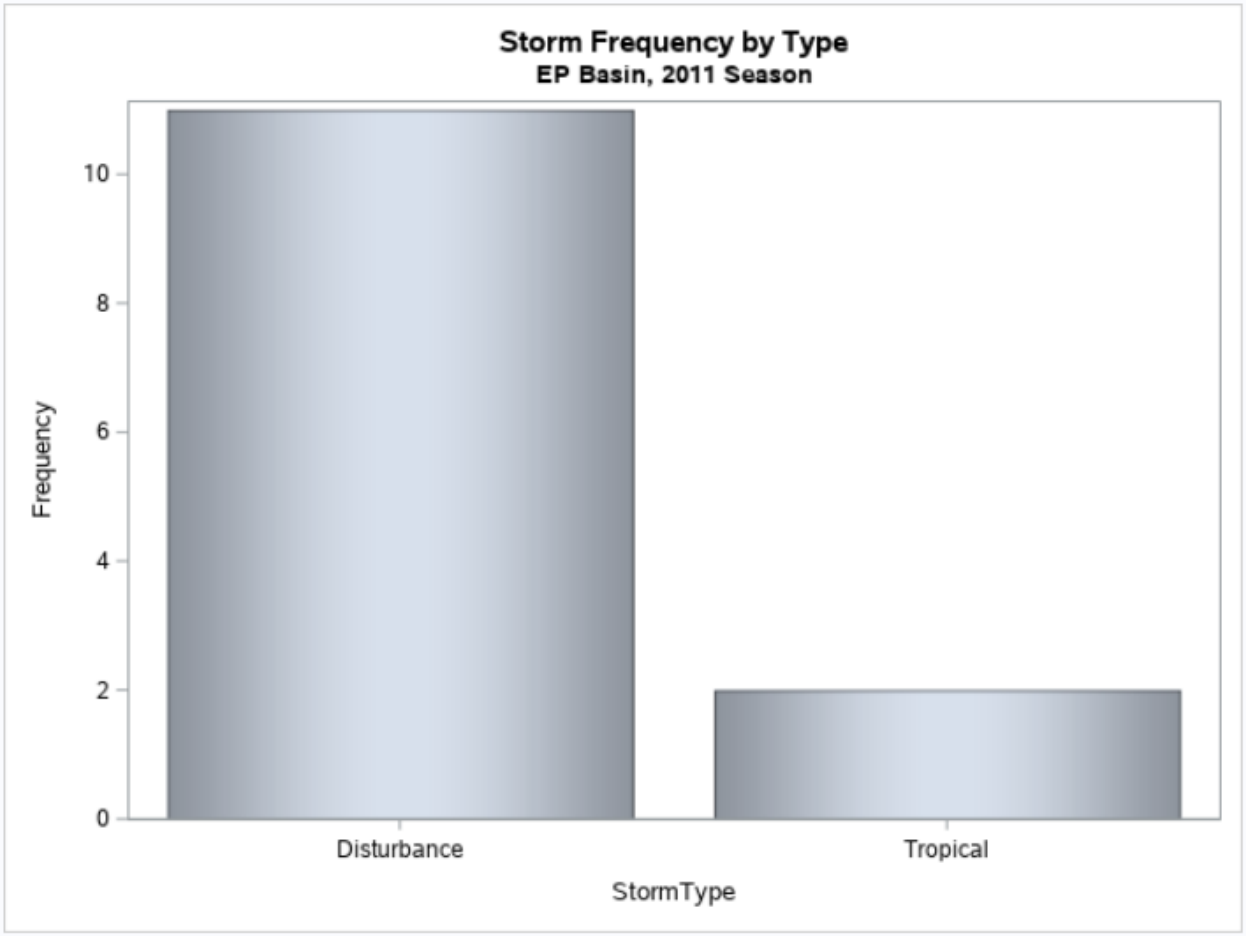


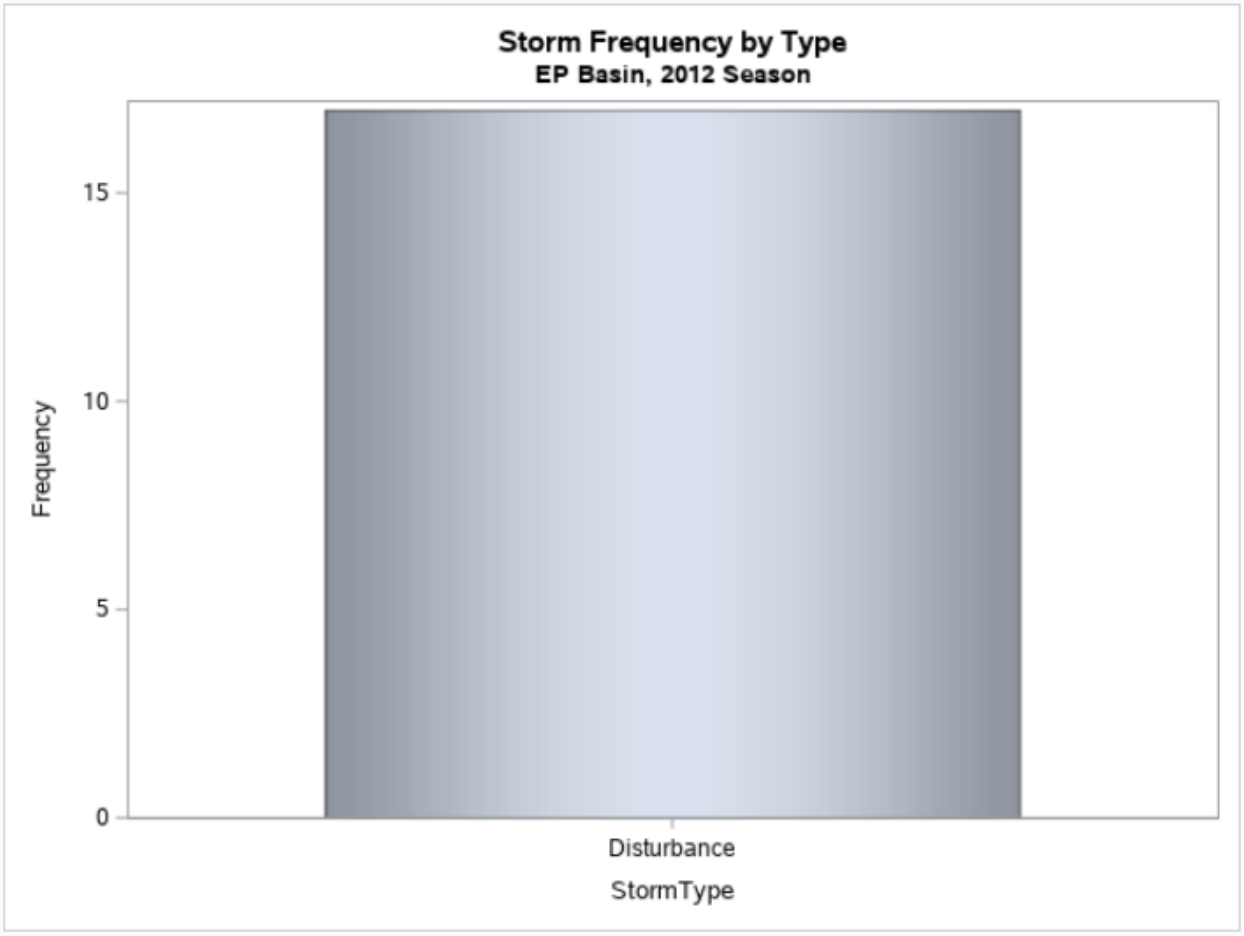

```
/***************************
* Iterative %DO Loops: Demo *
***************************/

options mcompilenote=all mlogic mprint;
%macro stormchart(basin, season);
title1 "Storm Frequency by Type";
title2 "&basin Basin, &season Season";
```

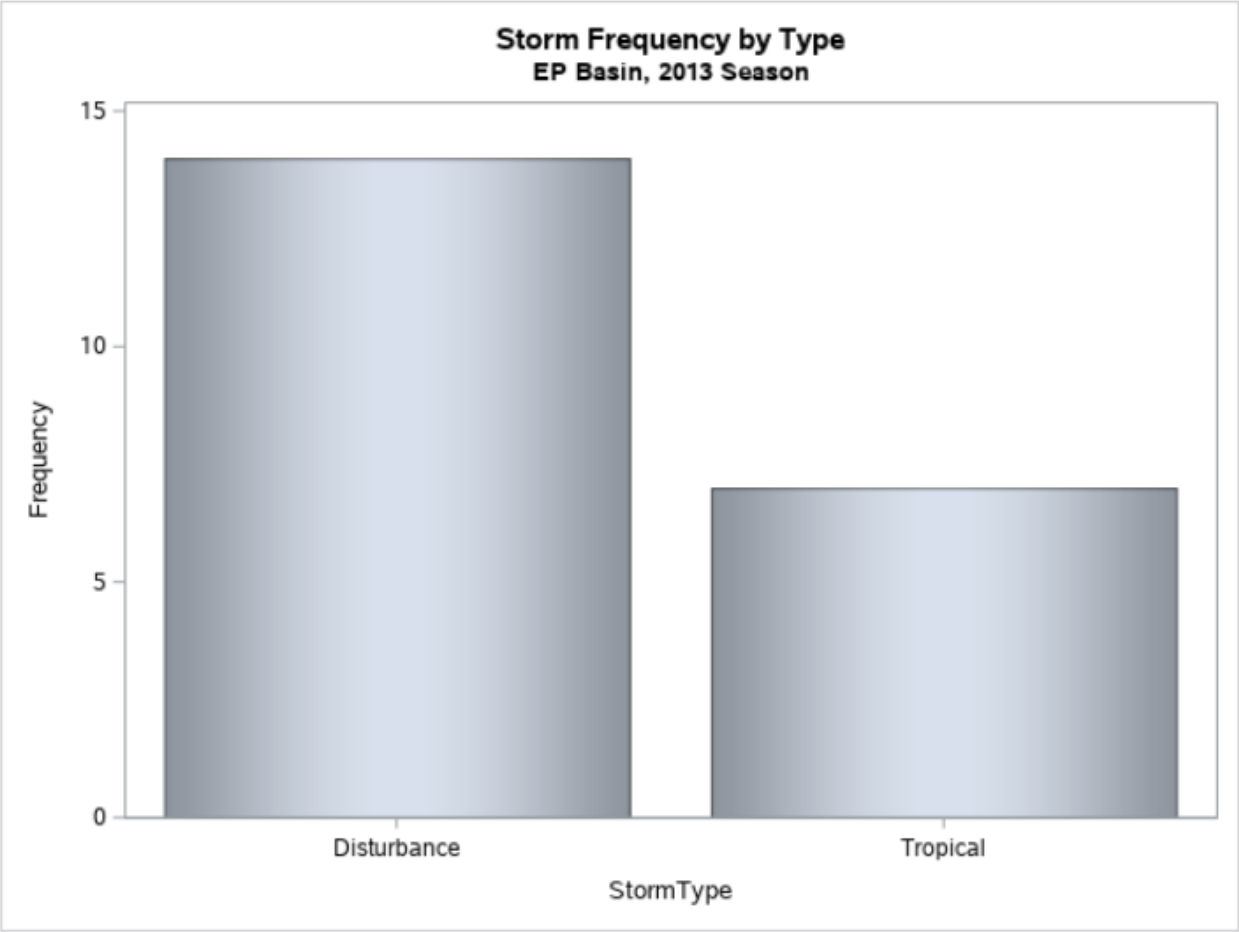
```
proc sgplot data=mc1.storm_final;
        vbar StormType / dataskin=pressed;
   where Basin="&basin" and Season=&season;
run;
title;
%mend stormchart;

%stormchart(NA,2015)

options mcompilenote=none nomlogic nomprint;
```




```
%macro StormChartRange(basin, start, stop);
%local Season;
        %do Season=&start %to &stop;
                %Stormchart(&basin, &season)
        %end;
%mend StormChartRange;
NOTE: The macro STORMCHARTRANGE completed compilation without errors.
       17 instructions 424 bytes.


%StormChartRange(EP,2011,2013)
```


Storm Frequency by Type
EP Basin, 2011 Season
Frequency
10
8
6
4
2
0
Disturbance
Tropical
StormType


Storm Frequency by Type
EP Basin, 2012 Season
15
10
5
0
Frequency
Disturbance
StormType

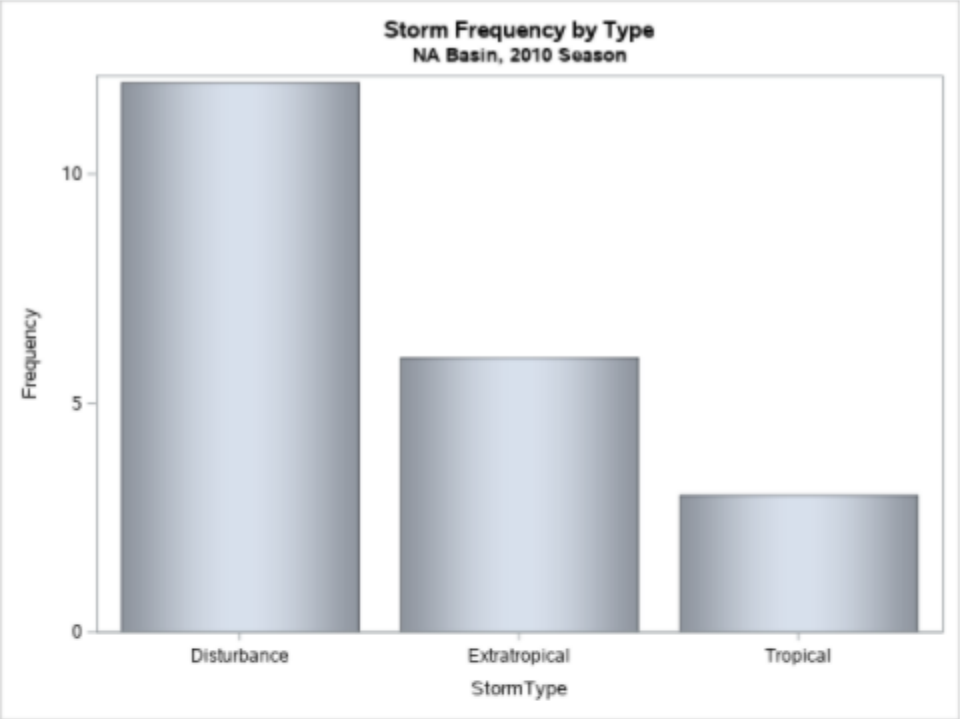


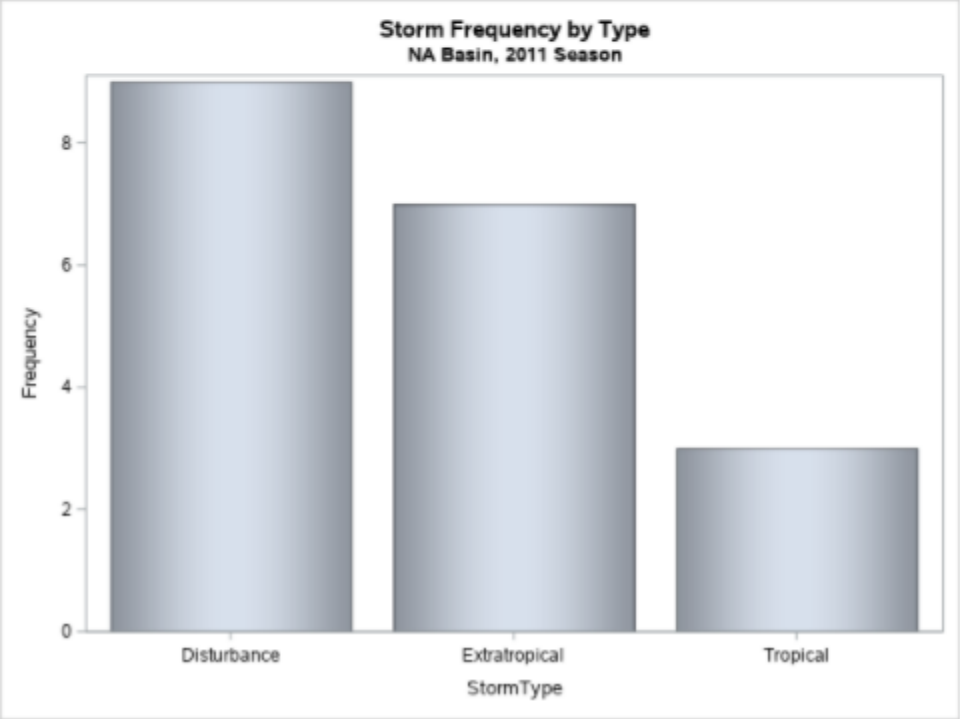

```
73         %StormChartRange(EP,2011,2013)
 MLOGIC(STORMCHARTRANGE):  Beginning execution.
 MLOGIC(STORMCHARTRANGE):  Parameter BASIN has value EP
 MLOGIC(STORMCHARTRANGE):  Parameter START has value 2011
 MLOGIC(STORMCHARTRANGE):  Parameter STOP has value 2013
 MLOGIC(STORMCHARTRANGE):  %LOCAL  SEASON
 MLOGIC(STORMCHARTRANGE):  %DO loop beginning; index variable SEASON;
start value is 2011; stop value is 2013; by value is 1.
 MLOGIC(STORMCHART):  Beginning execution.
 MLOGIC(STORMCHART):  Parameter BASIN has value EP
 MLOGIC(STORMCHART):  Parameter SEASON has value 2011
 MPRINT(STORMCHART):   title1 "Storm Frequency by Type";
 MPRINT(STORMCHART):   title2 "EP Basin, 2011 Season";
 MPRINT(STORMCHART):   proc sgplot data=mc1.storm_final;
 MPRINT(STORMCHART):   vbar StormType / dataskin=pressed;
 MPRINT(STORMCHART):   where Basin="EP" and Season=2011;
 MPRINT(STORMCHART):   run;
```

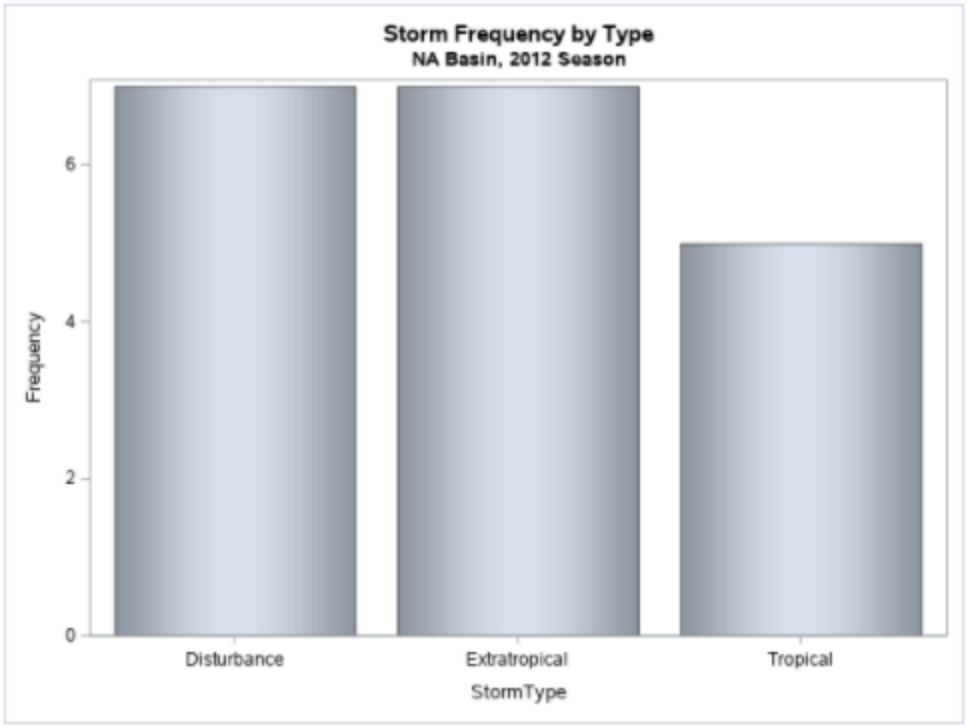
```
NOTE: There were 13 observations read from the data set
MC1.STORM_FINAL.
```

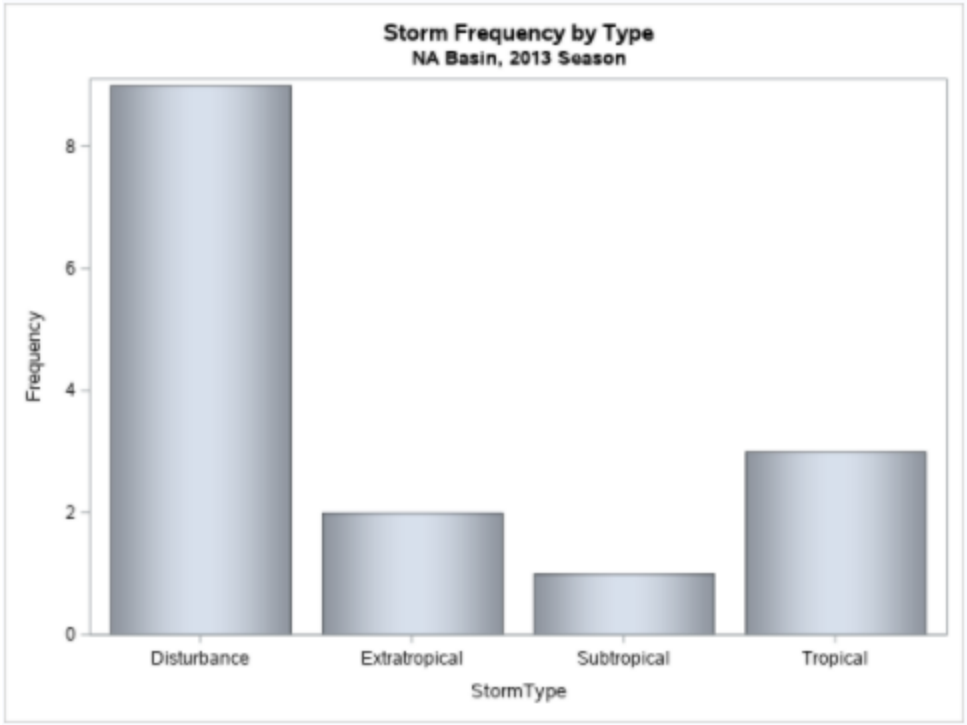
```
       WHERE (Basin='EP') and (Season=2011);

 MPRINT(STORMCHART):   title;
 MLOGIC(STORMCHART):  Ending execution.
 MLOGIC(STORMCHARTRANGE):  %DO loop index variable SEASON is now 2012;
loop will iterate again.
 MLOGIC(STORMCHART):  Beginning execution.
 MLOGIC(STORMCHART):  Parameter BASIN has value EP
 MLOGIC(STORMCHART):  Parameter SEASON has value 2012
 MPRINT(STORMCHART):   title1 "Storm Frequency by Type";
 MPRINT(STORMCHART):   title2 "EP Basin, 2012 Season";
 MPRINT(STORMCHART):   proc sgplot data=mc1.storm_final;
 MPRINT(STORMCHART):   vbar StormType / dataskin=pressed;
 MPRINT(STORMCHART):   where Basin="EP" and Season=2012;
 MPRINT(STORMCHART):   run;
NOTE: There were 17 observations read from the data set
MC1.STORM_FINAL.
       WHERE (Basin='EP') and (Season=2012);

 MPRINT(STORMCHART):   title;
 MLOGIC(STORMCHART):  Ending execution.
 MLOGIC(STORMCHARTRANGE):  %DO loop index variable SEASON is now 2013;
loop will iterate again.
 MLOGIC(STORMCHART):  Beginning execution.
 MLOGIC(STORMCHART):  Parameter BASIN has value EP
 MLOGIC(STORMCHART):  Parameter SEASON has value 2013
 MPRINT(STORMCHART):   title1 "Storm Frequency by Type";
 MPRINT(STORMCHART):   title2 "EP Basin, 2013 Season";
 MPRINT(STORMCHART):   proc sgplot data=mc1.storm_final;
 MPRINT(STORMCHART):   vbar StormType / dataskin=pressed;
 MPRINT(STORMCHART):   where Basin="EP" and Season=2013;
 MPRINT(STORMCHART):   run;
NOTE: There were 21 observations read from the data set
MC1.STORM_FINAL.
       WHERE (Basin='EP') and (Season=2013);

 MPRINT(STORMCHART):   title;
 MLOGIC(STORMCHART):  Ending execution.
 MLOGIC(STORMCHARTRANGE):  %DO loop index variable SEASON is now 2014;
loop will not iterate again.
 MLOGIC(STORMCHARTRANGE):  Ending execution.
```

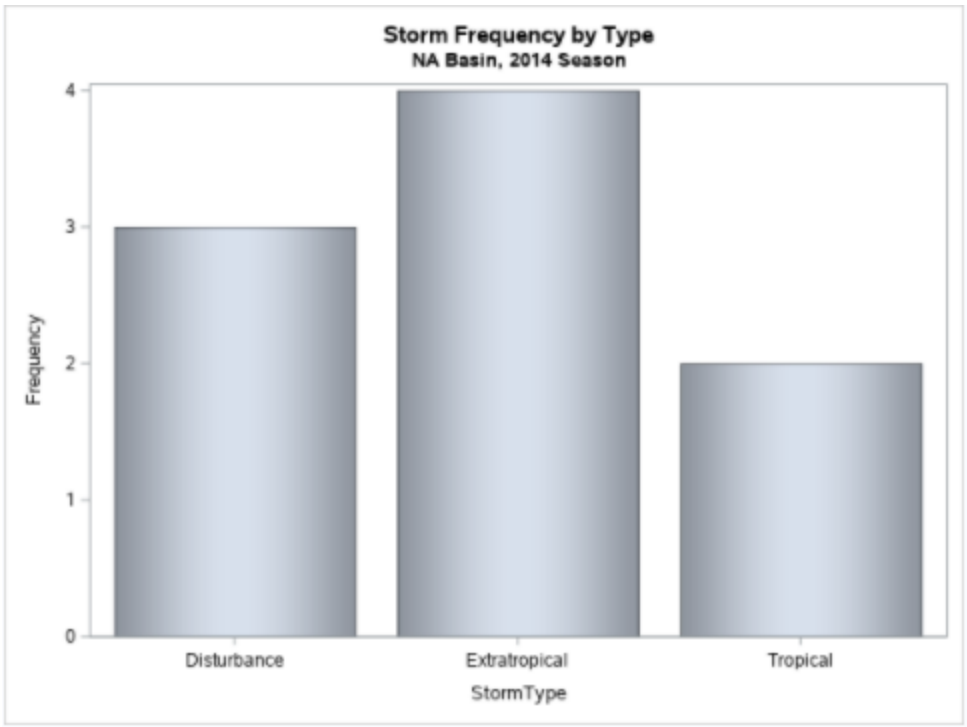
%StormChartRange(NA,2010,2016)

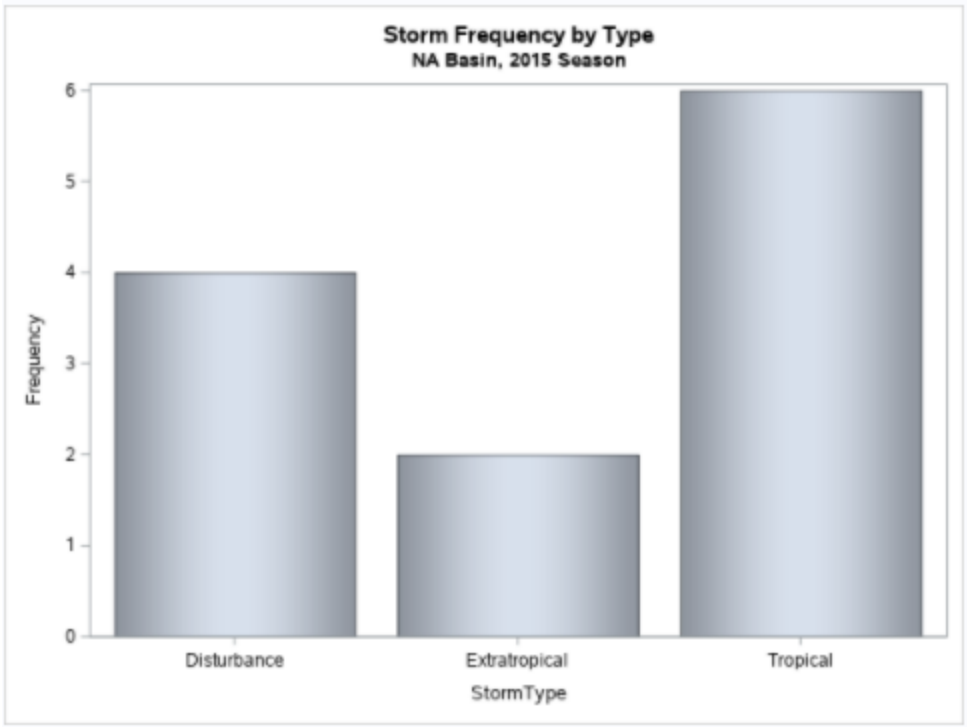





Storm Frequency by Type
NA Basin, 2012 Season
Frequency
0
2
4
6
Disturbance
Extratropical
Tropical
StormType


Storm Frequency by Type
NA Basin, 2013 Season
Frequency
0
2
4
6
8
Disturbance
Extratropical
Subtropical
Tropical
StormType





**********;

* Activity 4.09 *;

* 1) Run the program and examine the log. Identify and *;

```
*    correct the syntax error and rerun the program.    *;
*    Verify that the macro compiled successfully.       *;
*  2) Call the %AllCylinders macro with a start value of  *;
*    3 and stop value of 12.                            *;
*       %allcylinders(3,12)                             *;
*  3) Examine the reports and log. Why are there some    *;
*    values between 3 and 12 that did not produce       *;
*    reports?                                           *;
************************************************************;

options mlogic mprint;
%macro allcylinders(start,stop);
%do num=&start %to &stop;
   title "&num-Cylinder Cars";
   proc print data=sashelp.cars noobs;
      where Cylinders=&num;
      var Make Model Type Origin
         MSRP MPG_City MPG_Highway;
   run;
title;
%end;
%mend allcylinders;

%allcylinders(3,12)

options nomlogic nomprint;
```

3-Cylinder Cars

Make	Model	Type	Origin	MSRP	MPG_City	MPG_Highway
Honda	Insight 2dr (gas/electric)	Hybrid	Asia	$19,110	60	66

4-Cylinder Cars

Make	Model	Type	Origin	MSRP	MPG_City	MPG_Highway
Acura	RSX Type S 2dr	Sedan	Asia	$23,820	24	31
Acura	TSX 4dr	Sedan	Asia	$26,990	22	29
Audi	A4 1.8T 4dr	Sedan	Europe	$25,940	22	31

5-Cylinder Cars

Make	Model	Type	Origin	MSRP	MPG_City	MPG_Highway
Volvo	S60 2.5 4dr	Sedan	Europe	$31,745	20	27
Volvo	S60 T5 4dr	Sedan	Europe	$34,845	20	28
Volvo	S60 R 4dr	Sedan	Europe	$37,560	18	25
Volvo	S80 2.5T 4dr	Sedan	Europe	$37,885	20	27
Volvo	C70 LPT convertible 2dr	Sedan	Europe	$40,565	21	28
Volvo	C70 HPT convertible 2dr	Sedan	Europe	$42,565	20	26
Volvo	XC70	Wagon	Europe	$35,145	20	27

6-Cylinder Cars

Make	Model	Type	Origin	MSRP	MPG_City	MPG_Highway
Acura	MDX	SUV	Asia	$36,945	17	23
Acura	TL 4dr	Sedan	Asia	$33,195	20	28
Acura	3.5 RL 4dr	Sedan	Asia	$43,755	18	24

Toyota	Land Cruiser	SUV	Asia	$54,765	13	17
Volkswagen	Passat W8 4MOTION 4dr	Sedan	Europe	$39,235	18	25
Volkswagen	Phaeton 4dr	Sedan	Europe	$65,000	16	22
Volkswagen	Passat W8	Wagon	Europe	$40,235	18	25

10-Cylinder Cars

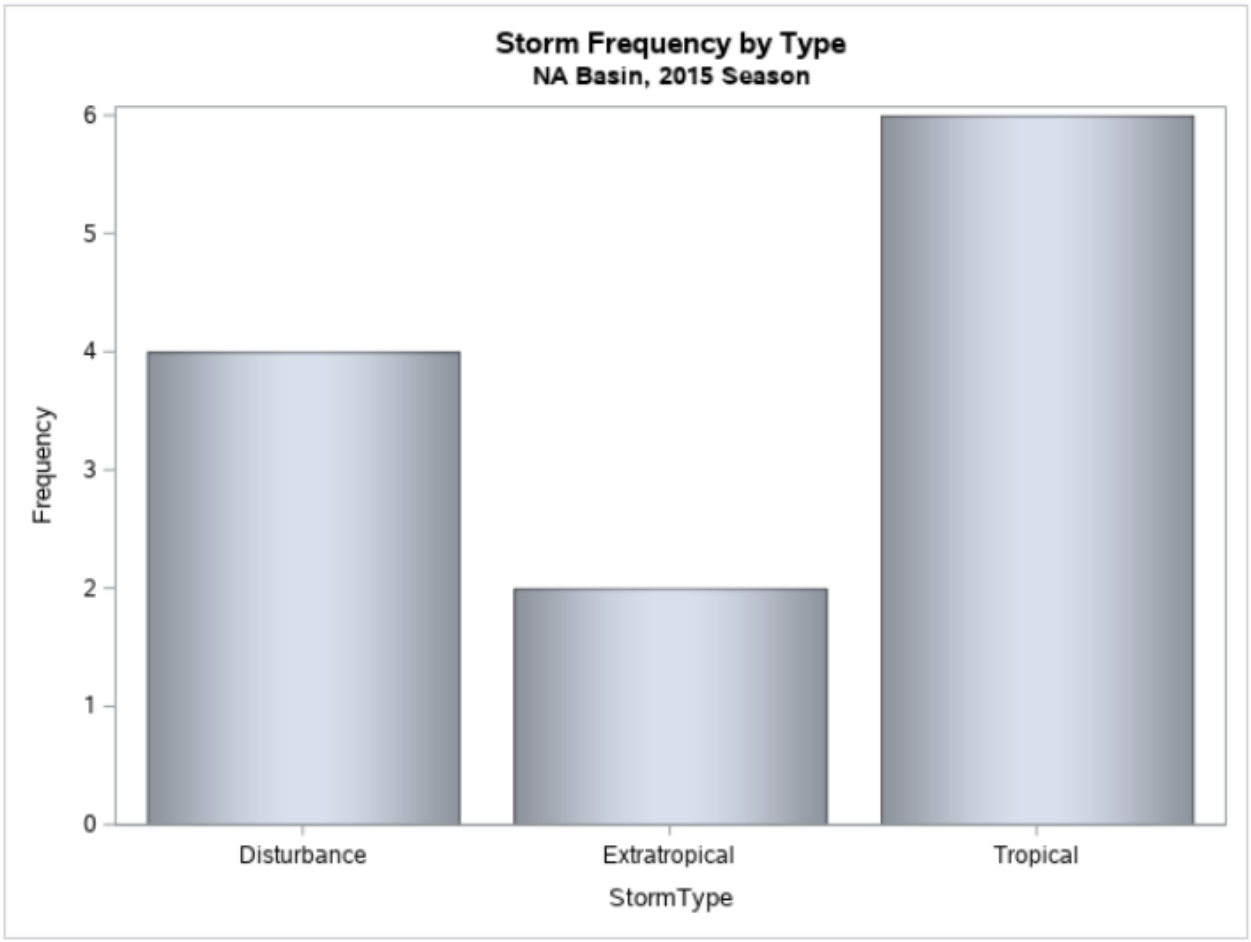
Make	Model	Type	Origin	MSRP	MPG_City	MPG_Highway
Dodge	Viper SRT-10 convertible 2dr	Sports	USA	$81,795	12	20
Ford	Excursion 6.8 XLT	SUV	USA	$41,475	10	13

12-Cylinder Cars

Make	Model	Type	Origin	MSRP	MPG_City	MPG_Highway
Mercedes-Benz	CL600 2dr	Sedan	Europe	$128,420	13	19
Mercedes-Benz	SL600 convertible 2dr	Sports	Europe	$126,670	13	19
Volkswagen	Phaeton W12 4dr	Sedan	Europe	$75,000	12	19

```
/*****************************
* Iterative %DO Loops: Demo *
*****************************/

options mcompilenote=all mlogic mprint;

%macro stormchart(basin, season);
title1 "Storm Frequency by Type";
title2 "&basin Basin, &season Season";
proc sgplot data=mc1.storm_final;
            vbar StormType / dataskin=pressed;
   where Basin="&basin" and Season=&season;
run;
title;
%mend stormchart;
```

%stormchart(NA,2015)

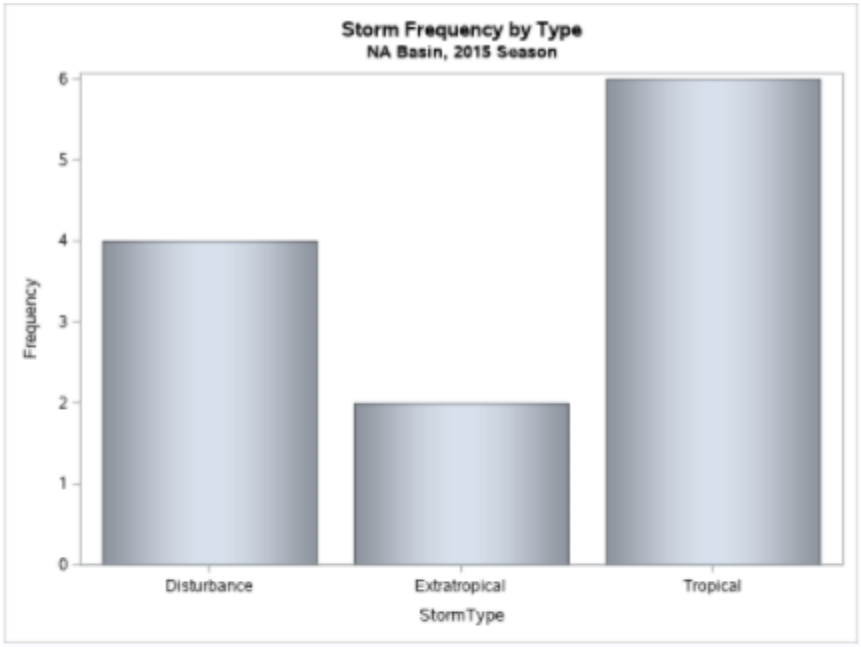


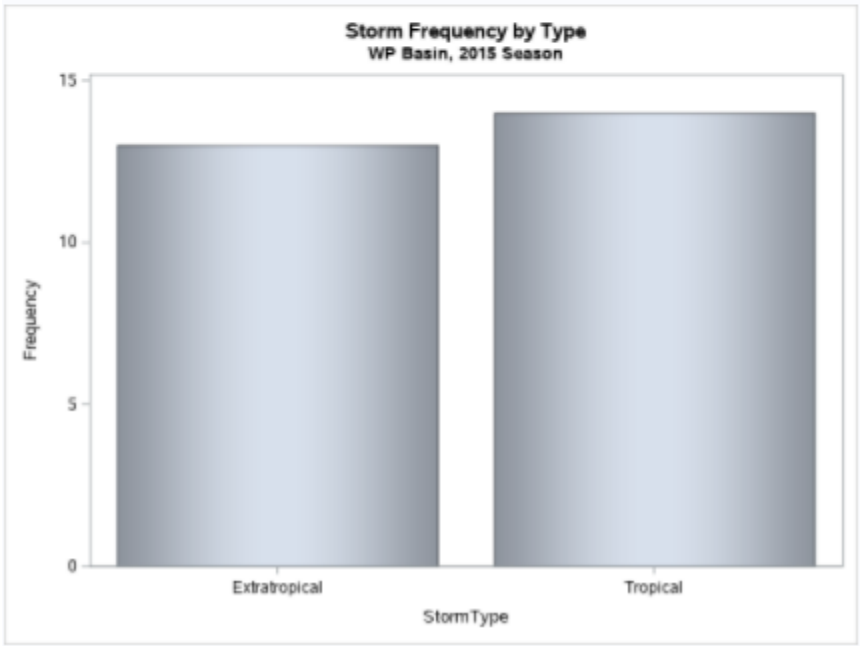

```
%macro allbasins(year);
%local i;
proc sql noprint;
select basin
        into :basin1-
        from mc1.storm_basin_codes;
quit;

%do i=1 %to &sqlobs;
        %stormchart(&&basin&i, &year)
%end;
```

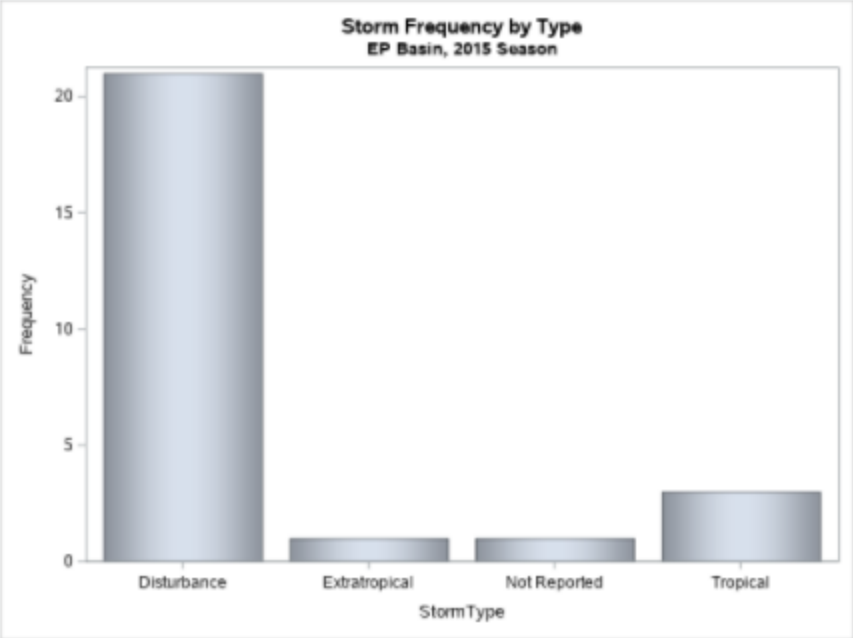
%mend allbasins;

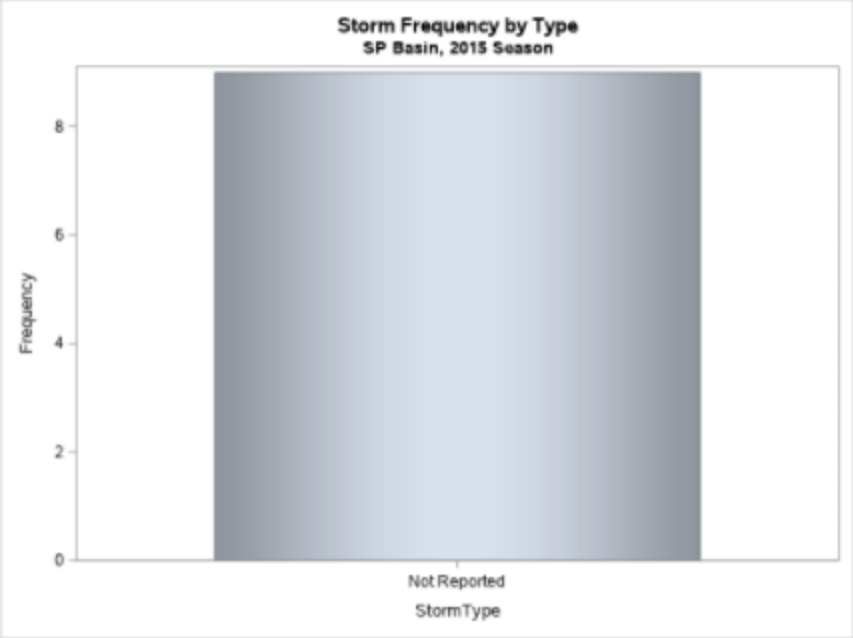
```
NOTE: The macro ALLBASINS completed compilation without errors.
      16 instructions 416 bytes.
```

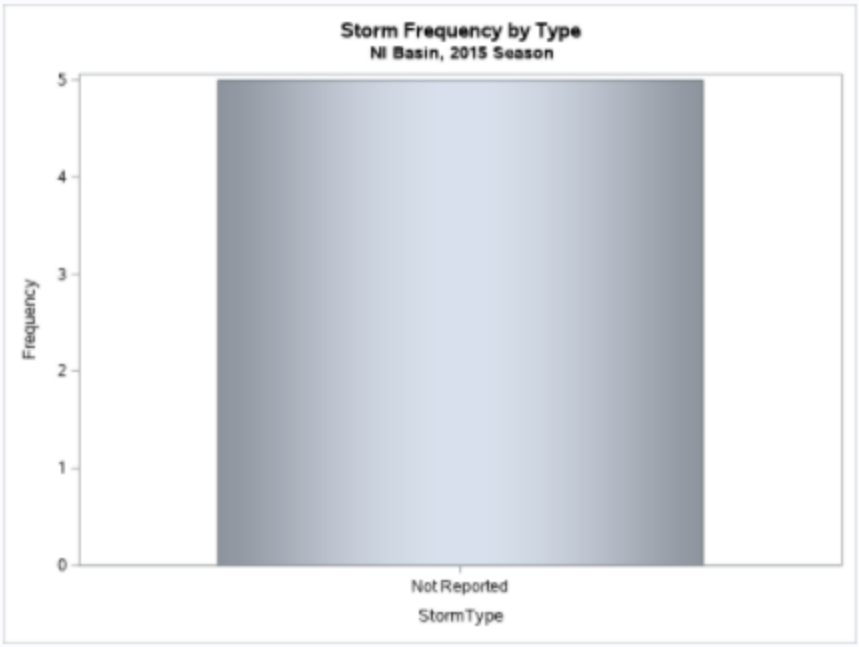
%allbasins(2015)

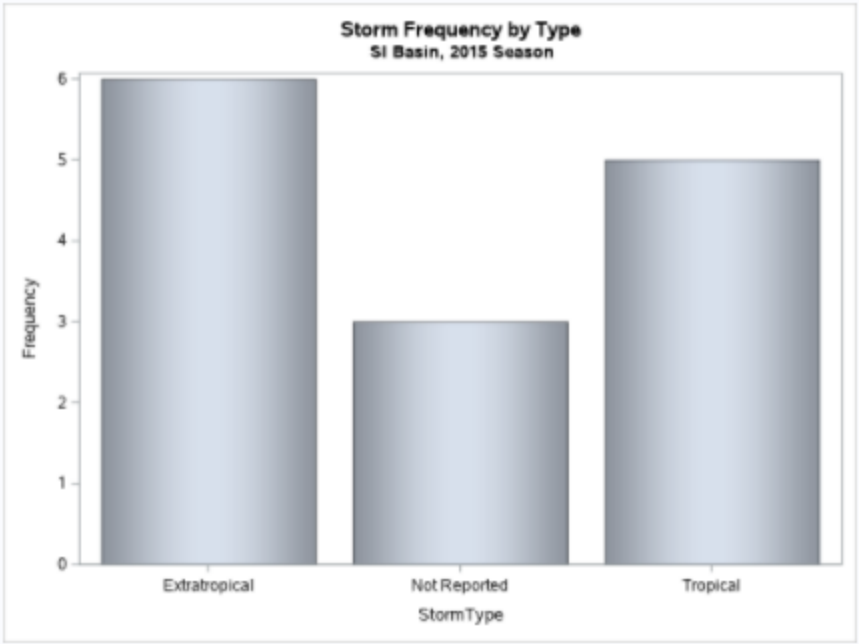


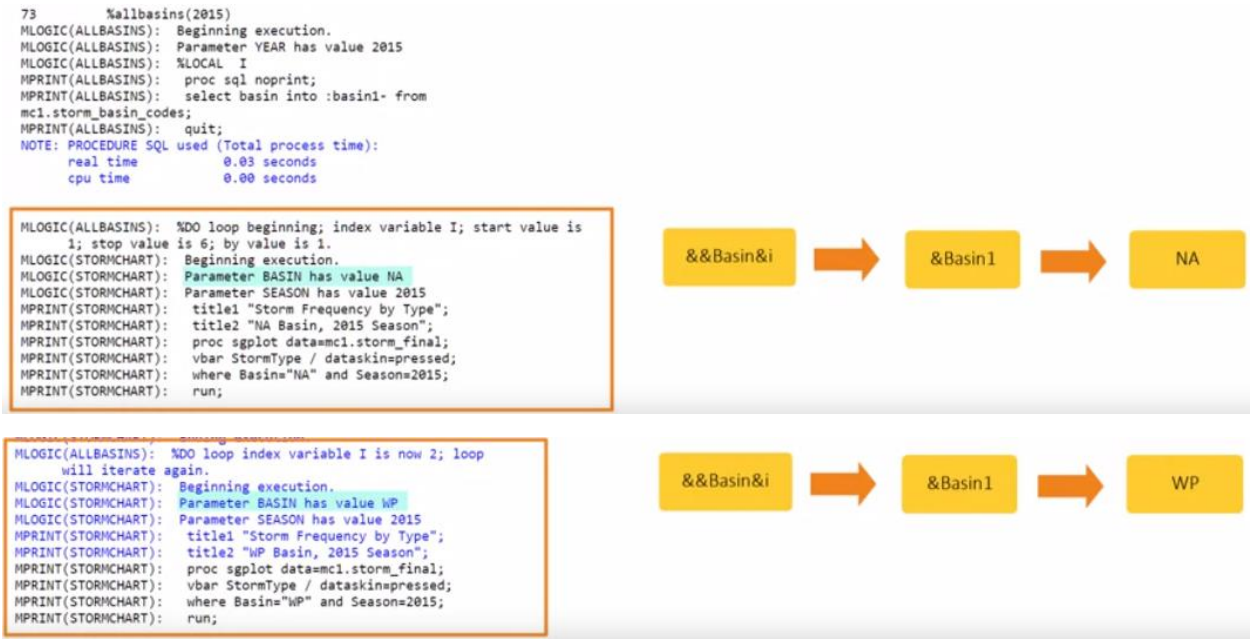



Storm Frequency by Type
EP Basin, 2015 Season
Frequency
20
15
10
5
0
Disturbance
Extratropical
Not Reported
Tropical
StormType


Storm Frequency by Type
SP Basin, 2015 Season
Frequency
8
6
4
2
0
Not Reported
StormType

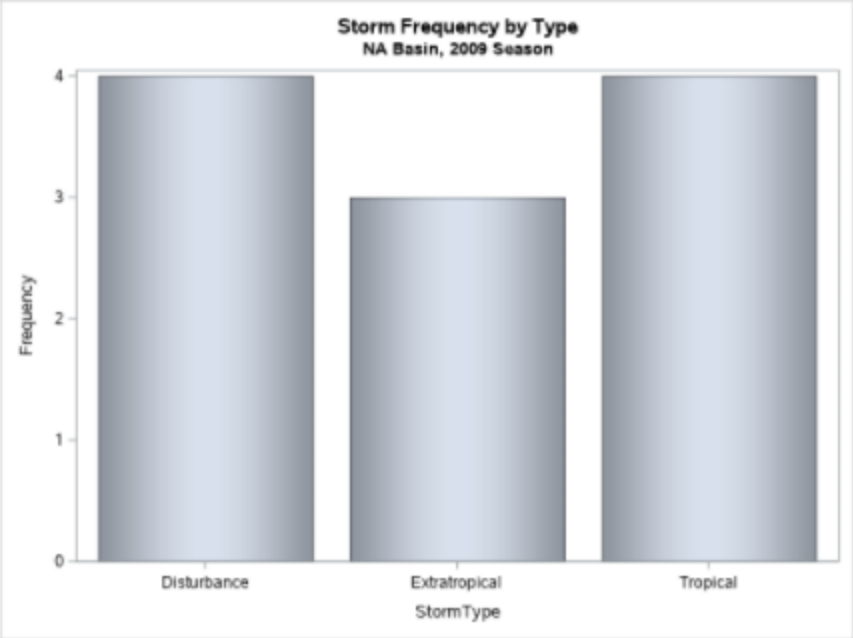

Storm Frequency by Type
NI Basin, 2015 Season
Frequency
0
1
2
3
4
5
Not Reported
StormType

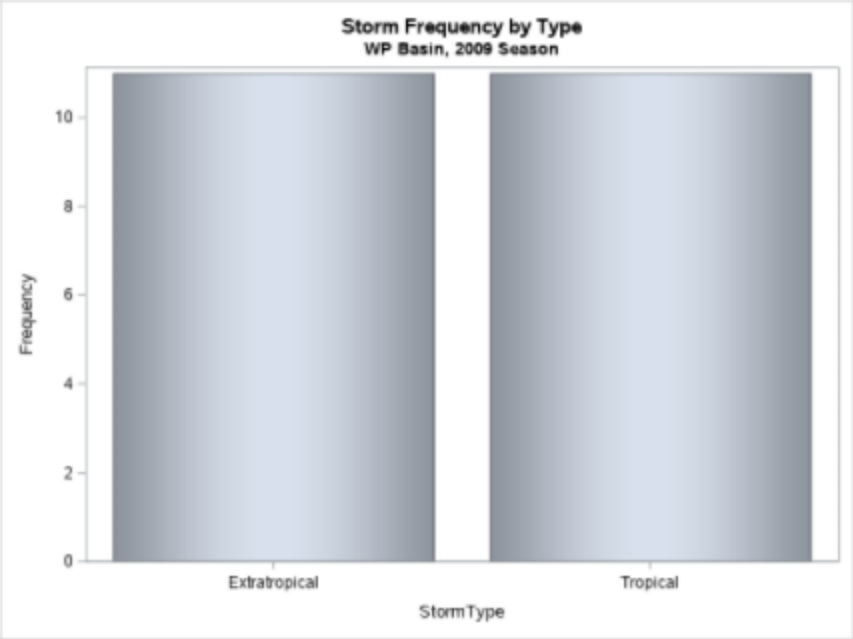

Storm Frequency by Type
SI Basin, 2015 Season
Frequency
0
1
2
3
4
5
6
Extratropical
Not Reported
Tropical
StormType

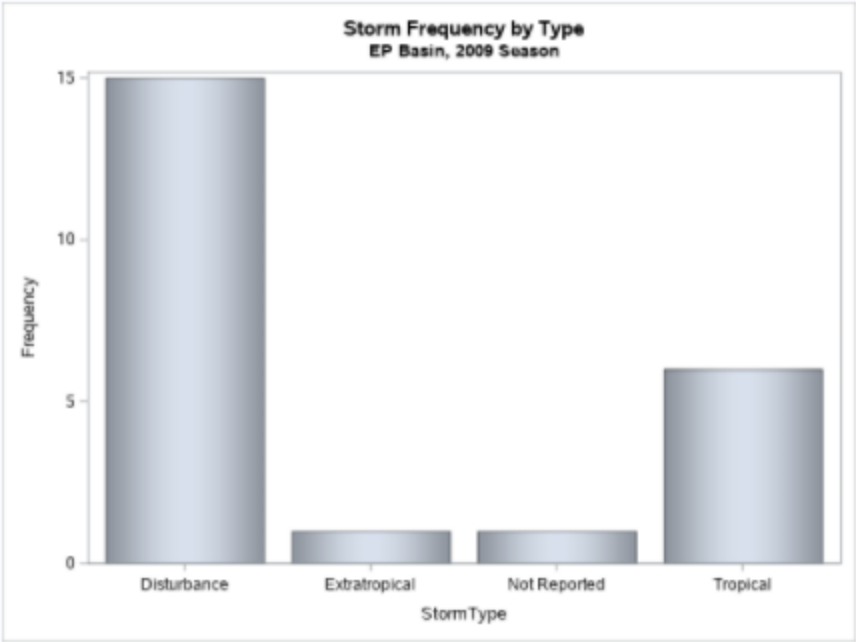


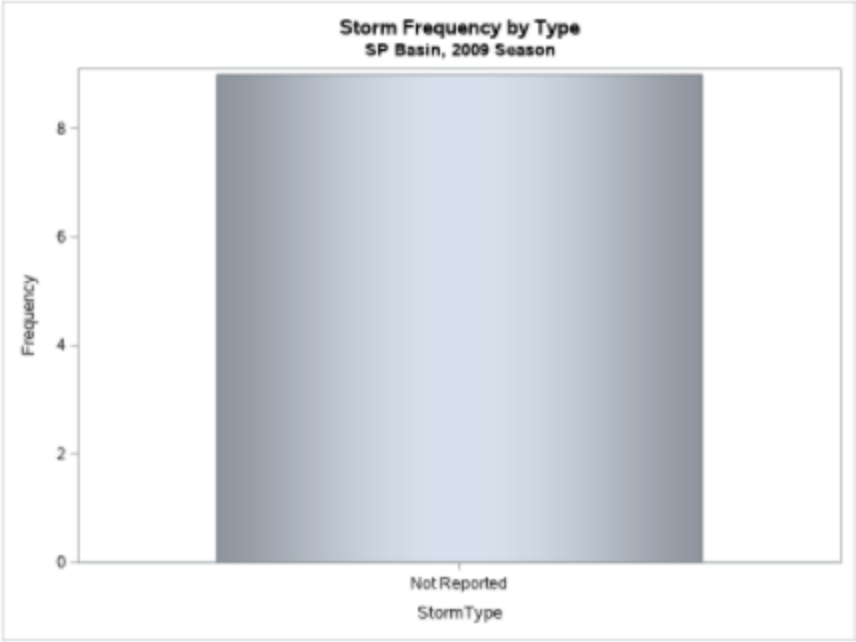
%allbasins(2009)

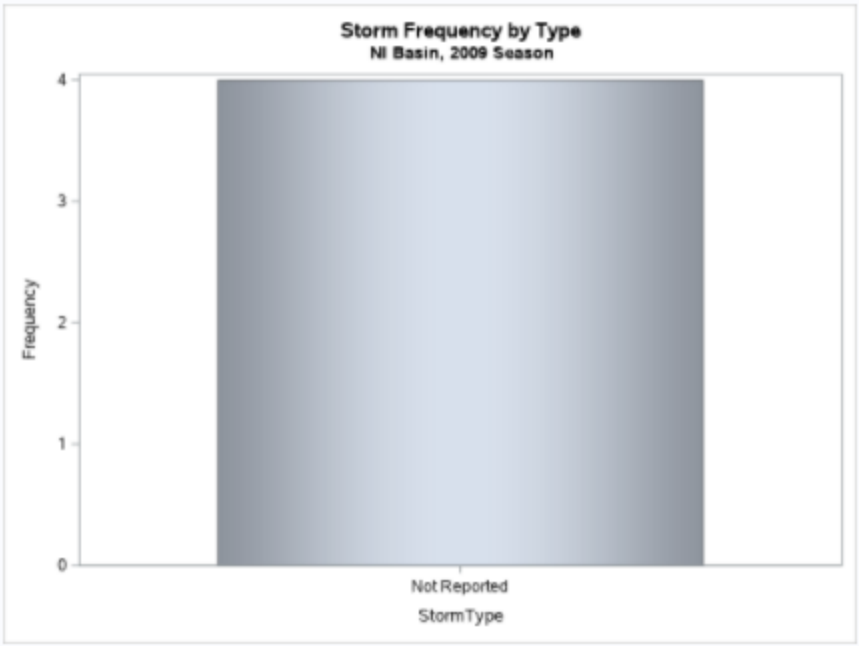
options nomlogic nomprint;

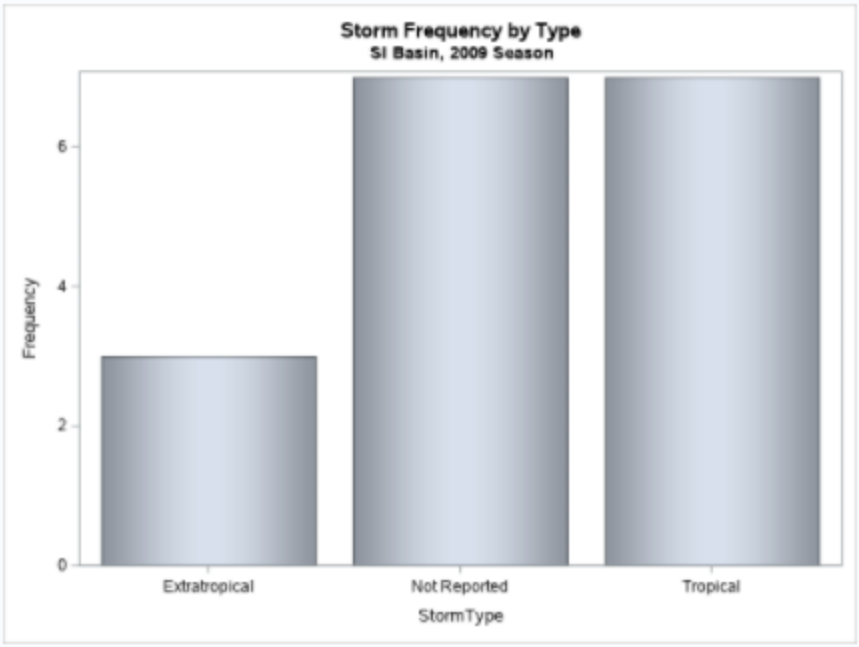

Storm Frequency by Type
NA Basin, 2009 Season
Frequency
4
3
2
1
0
Disturbance
Extratropical
Tropical
StormType


Storm Frequency by Type
WP Basin, 2009 Season
Frequency
10
8
6
4
2
0
Extratropical
Tropical
StormType


Storm Frequency by Type
EP Basin, 2009 Season
15
10
5
0
Frequency
Disturbance
Extratropical
Not Reported
Tropical
StormType


Storm Frequency by Type
SP Basin, 2009 Season
8
6
4
2
0
Frequency
Not Reported
StormType


Storm Frequency by Type
NI Basin, 2009 Season
Frequency
4
3
2
1
0
Not Reported
StormType


Storm Frequency by Type
SI Basin, 2009 Season
Frequency
6
4
2
0
Extratropical
Not Reported
Tropical
StormType

%DO %WHILE (*condition*);
text and macro program statements
%END;

Method #2

- *condition* is a macro expression that resolves to true or false.
- *condition* is evaluated at the top of the loop (at %DO).
  - If *condition* is true, statements between %DO and %END are processed.
  - If *condition* is false, the loop terminates.
  - If *condition* is false when entering the loop, statements between %DO and %END are not processed.

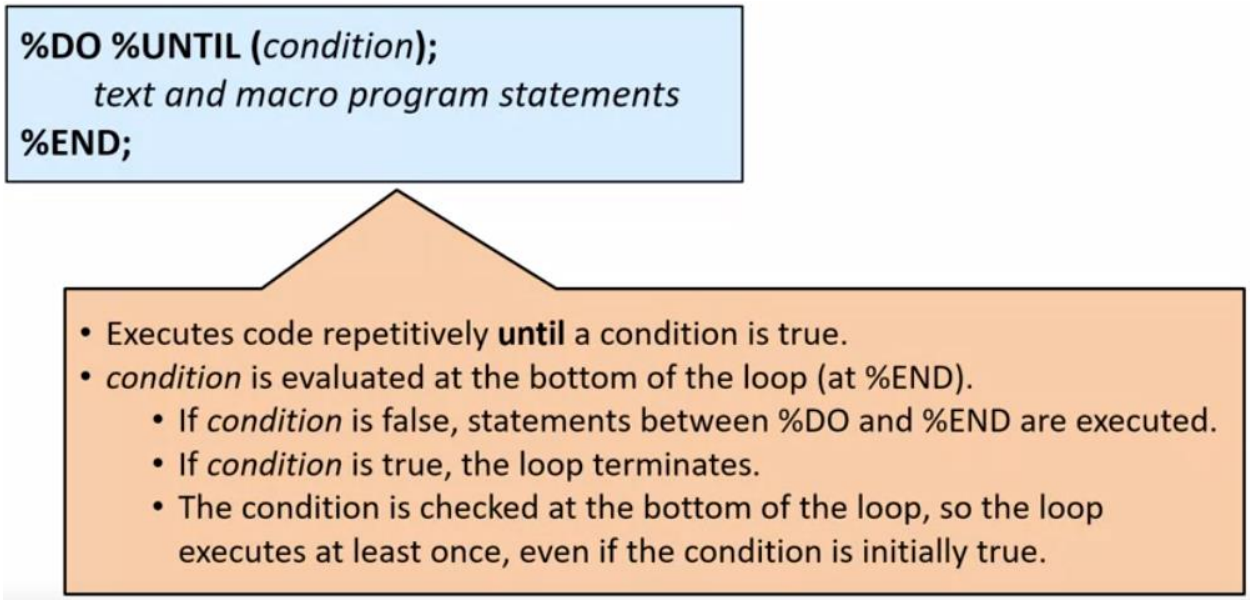
```
%macro cyllist(list);
%local i;
%let i=1;
%do %while (%scan(&list, &i) ne );
    %let num=%scan(&list, &i);
        title "&num-Cylinder Cars";
        proc print data=sashelp.cars noobs;
            where Cylinders=&num;
            var Make Model Type Origin
                MSRP MPG_City MPG_Highway;
        run;
    %let i=%sysevalf(&i+1);
    %end;
%mend cyllist;

%cyllist(3 4 5 6 8 10 12)
```

TRUE

%SCAN returns the first value in **List**.

Local Symbol Table

Name	Value
LIST	3 4 5 6 8 10 12
I	1



/**********

* Iterative Processing: Practice #10 *

**********/

/*Level 1 Practice: Using Iterative Processing and Indirect Referencing

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

Open m104p10.sas from the practices folder. Review and submit the program.

Verify that it generates a PROC MEANS report for Type 1 orders.

Convert the code to a macro named Orders that uses a %DO loop to generate a separate PROC MEANS step for order types 1, 2, and 3.

Add a call to the Orders macro.

Submit the program and verify that it generates three PROC MEANS reports, one for each order type.

Confirm that the mean of Total_Retail_Price for type 2 orders is 162.43.

Modify the program to insert a DATA _NULL_ step after the %MACRO statement:

Create a series of macro variables, type1 to typen, where n is the number of order types found in the mc1.order_type_codes table.

For each macro variable name, concatenate the prefix type with the value of order_type_code.

Assign the corresponding value of Order_Type to the macro variable.

Create a macro variable named numTypes and assign it the number of type macro variables created.

Hint: Use the END= option in the SET statement.

Modify the %DO loop:

Use numTypes as the stop value for the %DO loop.

Use an indirect reference to include the order type in the title as shown here: Order Type: Retail Store

Submit the program. What is the title of the last report?

Note: Copy and paste or type the title exactly as shown.

*/

```
title "Order Type: 1";
proc means data=mc1.orders sum mean maxdec=2;
   where Order_Type=1;
   var Total_Retail_Price CostPrice_Per_Unit;
run;
```

Order Type: 1

The MEANS Procedure

Variable	Label	Sum	Mean
Total_Retail_Price	Total Retail Price	1484964.54	133.38
CostPrice_Per_Unit	Cost Price Per Unit	422465.68	37.95

```
%macro Orders(start,stop);
%local i;
%do i=&start %to &stop;
        title "Order Type: &i";
        proc means data=mc1.orders sum mean maxdec=2;
        where Order_Type=&i;
        var Total_Retail_Price CostPrice_Per_Unit;
%end;
run;
%mend Orders;
```

%Orders(1,3)

Order Type: 2

The MEANS Procedure

Variable	Label	Sum	Mean
Total_Retail_Price	Total Retail Price	1484964.54	133.38
CostPrice_Per_Unit	Cost Price Per Unit	422465.68	37.95

Order Type: 3

The MEANS Procedure

Variable	Label	Sum	Mean
Total_Retail_Price	Total Retail Price	385779.96	162.43
CostPrice_Per_Unit	Cost Price Per Unit	102563.82	43.18

Order Type: 3

The MEANS Procedure

Variable	Label	Sum	Mean
Total_Retail_Price	Total Retail Price	350710.34	150.26
CostPrice_Per_Unit	Cost Price Per Unit	93676.58	40.14

```
%macro orders;
data _null_;
   set mc1.order_type_codes end=last;
   call symputx(cats("type",Order_type_code),Order_type);
   if last=1 then call symputx("numTypes", _n_);
run;
%do i=1 %to &numTypes;
   title "Order Type: &&type&i";
   proc means data=mc1.orders sum mean maxdec=2;
      where Order_Type=&i;
      var Total_Retail_Price CostPrice_Per_Unit;
   run;
%end;
```

%mend orders;

%orders

Order Type: Retail Store

The MEANS Procedure

Variable	Label	Sum	Mean
Total_Retail_Price	Total Retail Price	1484964.54	133.38
CostPrice_Per_Unit	Cost Price Per Unit	422465.68	37.95

Order Type: Catalog

The MEANS Procedure

Variable	Label	Sum	Mean
Total_Retail_Price	Total Retail Price	385779.96	162.43
CostPrice_Per_Unit	Cost Price Per Unit	102563.82	43.18

Order Type: Internet

The MEANS Procedure

Variable	Label	Sum	Mean
Total_Retail_Price	Total Retail Price	350710.34	150.26
CostPrice_Per_Unit	Cost Price Per Unit	93676.58	40.14

/**********

* Iterative Processing: Practice #11 *

**********/

/*Level 2 Practice: Using a %DO %UNTIL Loop

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

Open m104p11.sas from the practices folder. Review and submit the program.

Verify that the maximum wind speed for the 2015 season was 213 MPH.

Modify the Storms macro to accept a space-separated list of years.

Use a %DO %UNTIL loop and the %SCAN macro function to generate a report for each year in the list.

Submit the macro definition and verify that it completed compilation without errors.

Call %Storms to generate a report for the 2011 and 2014 seasons as shown here: %storms(2011 2014)

Which year had the highest maximum wind speed?

*/

```
%macro storms(yr);
title "&yr Storms";
proc means data=mc1.storm_final n min mean max maxdec=0;
   var MaxWindMPH MinPressure;
   where season=&yr;
run;
%mend storms;
```

%storms(2015)

2015 Storms

The MEANS Procedure

Variable	N	Minimum	Mean	Maximum
MaxWindMPH	93	35	87	213
MinPressure	93	872	965	1006

*Solution;

```
%macro storms(years);
%let i=1;
%let yr=%scan(&years,&i);
%do %until(&yr eq );
   title "&yr Storms";
   proc means data=mc1.storm_final n min mean max
            maxdec=0;
      var MaxWindMPH MinPressure;
      where season=&yr;
   run;
   %let i=%eval(&i+1);
   %let yr=%scan(&years,&i);
```

```
%end;

title;

%mend storms;


%storms(2011 2014)
```

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