

Getting Familiar with SAS® Version 8.2 and 9.0 Enhancements

Sunil K. Gupta, Gupta Programming, Simi Valley, CA

ABSTRACT

Are you just getting around to realizing what SAS version 8.2 has to offer while recently being introduced to SAS version 9.0 at SAS User Group Conferences? Are you one of those programmers searching for a reason to migrate to 8.2? Would it not be nice to have a quick overview of some of the key benefits of versions 8.2 and 9.0 without going into all the details of each enhancement?

This paper will review some of the new features of SAS versions 8.2 and 9.0. SAS users can now take advantage of more relational database entry and update functions such as preventing data entry errors, tracking updates with audit trails and backing up data with generation data sets. In SAS version 9.0, numerous enhancements have been made to enable more control of hardware processing and to provide new functionality with new procedures, statements and options. Many features have been designed to make our programming lives easier.

INTRODUCTION

SAS Version 8.2 provides new features for concatenating libraries and catalogs, defining variable attributes, along with the data entry features including preventing data errors with Integrity Constraints, tracking data updates with audit trails, and backing up data with generation data sets.

SAS Version 9.0 provides new features for taking advantage of multiprocessing capabilities, new character and numeric functions and formats, new ODS Formatting, along with importing and exporting of Microsoft Excel 2002 Files.

Another useful item worth noticing is the Style Editor in the Enterprise Guide. The Style Editor provides a nice user interface to customize styles.

WHAT'S NEW IN SAS VERSION 8.2

In SAS Version 8.2, more flexibility is provided, for example in concatenating multiple libraries and catalogs and in defining variable and filenames. In addition, numerous enhancements have been made to enable more power and flexibility in programming. More power is provided with new procedures and options for preventing data entry errors, tracking updates, and backing up data.

Concatenating Libraries and Catalogs

Multiple libraries and catalogs can be accessed using a single libref. This provides greater flexibility to prioritize and organize SAS files. It is now possible to concatenate libraries to assign one logical name to several libraries.

When SAS reads a file from the library, it locates the first file with that file name in the concatenation. When SAS writes a file to a library, it writes to the first library in the concatenation.

```
Libname mylib ('c:\myproject1' 'c:\myproject2');  
  
Libname alllib (mylib, 'c:\otherprojects');
```

The mylib libref references the two directories myproject1 and myproject2. The alllib libref references the mylib libref and the otherprojects directory.

This feature is useful to search through multiple libraries using a single libref name without specifying different librefs within the program.

If both Version 6.12 and 8.2 files are in the same directory, specify the engine type on the LIBNAME statement to access Version 6.12 files otherwise they will not be read. It is best to create separate LIBNAME statements for each version.

```
Libname mylib v6 'pathname';
```

Directory Contains	Default Engine
All SAS Version 6 files	V6
All SAS Version 8.2 files	V7
Both SAS Version 6 and 8.2 files	V7
No SAS files	V7

The search order for a catalog is similar to that for the library concatenation.

```
CATNAME work.allprojects  
      (mylib.project1 mylib.project2);
```

Catalog entries are written to the first catalog listed in the concatenation (mylib.project1). When reading catalog entries, SAS searches the catalog in the first catalog listed before going to the other catalogs.

Defining Variable Attributes

A key benefit of version 8.2 is to take advantage of using up to 32 characters long variable names in the following:

- Library references
- File references
- Format name
- Informat names
- Variable names
- Data set names
- Macro variable names
- Macro names

Note that while the variable names are case sensitive and stored as mixed case, they are actually normalized for look-ups. This means that it is not possible to create two variables with the same name but different cases.

```
Options VALIDVARNAME = V7;  
  Allows for case sensitive and up to 32  
  characters long variable names  
  ex. My_Favorite_Variable
```

```
Options VALIDVARNAME = V6;  
  Version 6 rules apply.
```

```
Options VALIDVARNAME = any;  
  Allows variable names to contain special  
  characters.
```

WARNING: Only the Base product and the SAS/STAT product have been tested for use with VALIDVARNAME=ANY. Other use of this option is considered experimental and may cause undetected errors.

Another benefit of version 8.2 is the ability to have longer character values and labels. Character data can be up to 32,000 in length. Labels for both data sets and variables can be up to 256 in length.

```
My_Favorite_Variable = 'Glad I can have a very very  
  long char value here';
```

```
label My_Favorite_Variable = 'Glad I can have a very very  
  very long label here';
```

Preventing Data Errors with Integrity Constraints

General and Referential Integrity Constraints are helpful to enforce consistency and correctness of the data. SAS provides various tools to control the quality of data entered such as non-missing value, unique values and valid values. Data validation rules are applied whenever the data set is updated as in the DATA STEP with the SET or MODIFY statement or PROC SQL with the INSERT INTO

statement for example. In addition, data entry error messages can be displayed to better inform the user.

Once the integrity constraints have been established, they become permanent properties of the data set irregardless of how the data set is updated, ex. Data Step. The integrity constraints apply to any new data added, not to data already in the data set.

Integrity constraints can be applied to both numeric and character variables.

```
Proc datasets lib=mylib;  
  Modify test;  
    Test is the data set to contain integrity  
    constraint conditions.
```

```
Integrity constraint create  
  Req_Iname = not null(Iname)  
  message = 'Last Name is REQUIRED';
```

Req_Iname is the name of the NOT NULL constraint for the Iname variable. This prevents missing values to be entered in the Iname variable.

```
Integrity constraint create  
  Ok_gender = check(where=(gender in  
  ('Male', 'Female')))  
  message = 'Gender MUST be Male or Female';
```

Ok_gender is the name of the CHECK constraint for the gender variable. The WHERE clause defines the acceptable range of values, Male and Female. The CHECK constraint could be any acceptable condition using the IN or the RANGE operators for example.

```
Integrity constraint create  
  unique_id = unique(id)  
  message = 'ID value MUST be unique';
```

Unique_id is the name of the UNIQUE constraint for the id variable. This prevents duplicate values to be entered in id variable.

```
Run;
```

The PROC CONTENTS of the TEST data set below shows how the integrity constraints become permanent properties of the data set.

-----Alphabetic List of Integrity Constraints-----

Integrity #	Constraint Type	Where Variables Clause	User Message
1	Req_Iname Not Null	Iname	Last Name is REQUIRED
2	Ok_gender Check or	gender in ('Male', 'Female')	Gender MUST be Male Female ID value MUST be
3	Unique_id Primary Key id unique		

-----Alphabetic List of Indexes and Attributes-----

# of					
#	Index	Unique Option	Built by IC	Owned by IC	Unique Values
1	id	YES	YES	YES	1

As an alternative to PROC DATASETS, the SQL procedure can also be used to create integrity constraints. Note that by using this approach, SAS automatically names the integrity constraint variables.

```
Proc sql;
  Create table test
    (lname char(10) not null,
     gender char(10) check(gender in('Male','Female')),
     id num(4) unique);
quit;
```

The NOT NULL constraint is applied to prevent entering missing values in the lname variable.

-----Alphabetic List of Integrity Constraints-----

Integrity #	Constraint Type	Where Variables Clause	User Message
1	_CK0001_ Not Null	lname	Last Name is REQUIRED
2	_NM0001_ Check	gender in ('Male', or 'Female')	Gender MUST be Male or Female
3	_UN0001_ Primary Key	id	ID value MUST be unique

-----Alphabetic List of Indexes and Attributes-----

# of					
#	Index	Unique Option	Built by IC	Owned by IC	Unique Values
1	id	YES	YES	YES	1

Referential Integrity Constraints are helpful to establish relational models between data sets. This is essential to preserve the validity of a data warehouse system. Referential Integrity Constraints are helpful to define primary and secondary keys variables.

Tracking Data Updates with Audit Trails

Tracking data changes is easy with Audit Trails. Audit Trails are useful to provide a log of invalid data if used with integrity constraints. An audit trail file is created with the same name as the data set in the same SAS data library but as a member type of AUDIT.

The DATASETS procedure manages the audit trail data sets. TEST is the data set with an audit trail data set established. The audit trail data set name is also test but has audit for the memtype.

Although SAS provides several audit trail variables, you

can define user specific audit trail variables with the USER_VAR keyword. The code below creates two user specific audit trail variables— username, chngreason.

```
Proc datasets lib=mylib;
```

```
  Audit test;
  Initiate;
```

```
  User_var username $25
    label = 'Person making the update';
  User_var chngreason $50
    label = 'Reason for change';
```

```
Run;
```

The SQL procedure is used to insert one record in the TEST data set. Note that the user is required to enter values in the user specified audit trail variables.

```
Proc sql;
  Insert into mylib.test
  Set id = 4,
     lname = 'Smith',
     gender = 'F',
     username = 'Dan', /* need to define */
     chngreason = 'New Hire'; /* need to define */
Quit;
```

Since the audit trail feature was established for the TEST data set, the audit trail data set storing the data changes can be accessed to confirm changes made. The audit trail data set is specified with the data set option TYPE = AUDIT and is treated like any other data set.

```
Proc print data=mylib.test (type=audit);
Run;
```

All of the original data set variables, along with 6 audit trail variables (_AT*_) and the two user defined variables username and chngreason are displayed from PROC PRINT.

					_ A T R E _ A T			
					_ T _ A _ A T			
M			c		A	A	U	T
		u	s	h	T	T	R	T
S		g	e	r	E	O	N	S
S		e	r	e	T	B	C	E
I		n	n	a	I	S	O	R
A		d	a	s	M	N	D	I
n								
E								
O								
a								
G								
b								
m								
e								
s								
r								
d								
e								
n								
					_ _ _ _			
1	Smith	F	4	Dan	New Hire	09MAY2003:15:05:06	1	sunil DA
2	Smith	F	5	Dan	New Hire	09MAY2003:15:11:00	2	sunil DA

The following table is a description of all of the audit trail variables _AT*_ that are automatically updated and available in the audit trail data set.

AT* Variables

_AT*	Description
ATDATETIME	Stores the date and time of modification
ATUSERID	Stores the logon userid associated with modification
ATOBSNO	Stores the observation # affected by the modification
ATRETURNCODE	Stores the event return code
ATMESSAGE	Store the SAS log message at the time of the modification
ATOPCODE	Stores a code describing the type of operation

An important audit trail variable to store the type of operation performed is the _ATOPCODE_ variable.

ATOPCODE values are listed below

Code	Event
DA	Data Added record image
DD	Data Deleted record image
DR	Before-update record image
DW	After-update record image
EA	Observation add failed
ED	Observation delete failed
EU	Observation update failed

Another important audit trail variable is the _ATMESSAGE_ variable. This variable stores the SAS message written to the log at the time of the data set update.

Backing up Data with Generation data sets

With the new generation data set option, GENMAX, users can easily and automatically create backup data sets for each data set update. A version of the SAS data set can be created by using any of the replacement process such as a data step with a set or merge statement or a proc sort step.

It is important to realize that the GENMAX data set option should only be defined once.

The GENMAX = 3 data set option tells SAS to keep the current version of test2 (also referred to as the base version) along with a maximum of 2 historical copies.

```
Data test2 (genmax = 3);          /* first version */
Set test2;                        /* one obs- fname = 'Charles'; */
Run;
```

```
Data test2;                      /* second version */
Set test2;
If fname = 'Charles' then fname = 'Charlie';
Run;
```

```
Data test2;                      /* third version */
Set test2;
If fname = 'Charlie' then fname = 'Chuck';
Run;
```

This will print the second version of the data set. Users can reference the historical version of the data set test2 in any procedure by specifying the GENNUM = data set option according to the following table:

```
Proc print data=test2 (gennum = 2);
Run;
/* fname = 'Charlie'; */
```

TABLE NAME	ABSOLUTE GEN #	RELATIVE GEN #	FNAME
test2	3	0 (base – Most Recent)	Chuck
test2#002	2	-1 (2 nd most recent)	Charlie
test2#001	1	-2 (oldest version)	Charles

Either the absolute or the relative generation # can be used in the gennum data set option.

When saving to more than three versions of the data set in this example, the first most recent version is moved to the second most recent version and is replaced by the current version. The oldest version is deleted and replaced by the original second most recent version.

WHAT'S NEW IN SAS VERSION 9.0

Numerous enhancements have been made in version 9.0, providing more option for controlling hardware processing and new functionality with new procedures, functions, formats, statements and options.

Taking Advantage of Multiprocessing Capabilities

Specifying the THREADS options in the system options statement tells SAS to use threaded processing if it is available in the operating system. This option is available at the system level as well as at the procedure level for selected procedures. This option is useful to reduce the processing time.

```
OPTIONS THREADS;
```

What's new in SAS Procedures

Numerous enhancements have been made in selected SAS procedures to better control processing method and to allow more flexibility in programming.

For example, users can now directly specify the text to be placed in the HTML contents file to be linked to the cross tabulation table in the following PROC FREQ. This is easily accomplished without using PROC TEMPLATE or a different style.

```
PROC FREQ data=sashelp.class;  
  TABLE sex*age / CONTENTS = 'My Contents';  
Run;
```

The HTML file below shows the custom text that is hyperlinked with the PROC FREQ results.

The screenshot shows the SAS Output window with the following content:

Table of Contents

1. The Freq Procedure
Table Sex * Age
My Contents

The SAS System

The FREQ Procedure

Sex	Age						Total
	11	12	13	14	15	16	
F	1	2	2	2	2	0	9
	5.26	10.53	10.53	10.53	10.53	0.00	47.37
	11.11	22.22	22.22	22.22	22.22	0.00	
	50.00	40.00	66.67	50.00	50.00	0.00	
M	1	3	1	2	2	1	10
	5.26	15.79	5.26	10.53	10.53	5.26	52.63
	10.00	30.00	10.00	20.00	20.00	10.00	
	50.00	60.00	33.33	50.00	50.00	100.00	
Total	2	5	3	4	4	1	19
	10.53	26.32	15.79	21.05	21.05	5.26	100.00

If there is a need to preserve the SAS internal date and time when the input data set was created and last modified before the sorting the data set to another data set, then the DATECOPY option can be specified. Specifying the THREADS option in the PROC SORT statement overrides the system option and activates the multithreaded sorting facility.

```
PROC SORT data = demog  
  out =sands03.demog DATECOPY THREADS;  
  by gender;  
Run;
```

```
PROC CONTENTS data = demog;  
run;
```

The CONTENTS Procedure

Data Set Name	DEMOG	Observations	25
Member Type	DATA	Variables	7
Engine	V9	Indexes	0
Created	6:14 Tuesday, June 3, 2003	Observation Length	56
Last Modified	6:14 Tuesday, June 3, 2003	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label			
Data Representation	WINDOWS		
Encoding	wlatin1	Western (Windows)	

The SQL procedure can now reference up to 32 views and tables. In addition users have access to more dictionary tables and new columns in existing tables. The THREADS option is available to improve processing time.

SAS developers will be happy to learn that the INTO clause now supports leading zeros as macro names when creating macro variables.

```
Select * into :n01 - :n10 from test;
```

Creates 10 macro variables names
n01, n02, ... n10;

What's new in Character & Numeric Functions and Formats

New character and numeric functions and formats allow greater flexibility in meeting programming requirements.

There are over 200 new functions in version 9.0. The following new functions and informat will be described in this paper:

ANYALNUM, ANYALPHA
 ANYDTEW.
 CAT, CATT, CATS, CATX
 COUNT
 COMPARE
 MEDIAN, PCTL
 SYMPTX

ANYALNUM

Returns the first position of alphanumeric character. This is helpful for variables storing alphanumeric values.

```
data _null_;
  string = 'abc 123 +=';
  alpha = anyalnum(string);
  numeric = anyalnum(string, 5);
  other = anyalnum(string, 9);
  put alpha= numeric= other=;
run;
```

/* alpha=1 numeric=5 other=0 */

ANYALPHA

Returns the first position of alphabetic character.

```
data _null_;
  string = 'abc 123 +=';
  alpha = anyalpha(string);
  numeric = anyalpha(string, 5);
  other = anyalpha(string, 9);
  put alpha= numeric= other=;
run;
```

/* alpha=1 numeric=0 other=0 */

ANYDTEW.

This is an undocumented informat that is very helpful for reading dates in any valid date format. The user is not required to know the date format in advance. In addition, the date format can vary within the same date variable.

```
data dates;
input mydate anydtew.; format mydate mmddyy10.;
cards; /* results */
14JAN1921 /* 01/14/1921 - ddmmmyyyy */
14011921 /* 01/14/1921 - ddmmmyyyy */
01141921 /* 01/14/1921 - mmddyyyy */
;
run;
```

CAT, CATT, CATS, CATX

The family of CAT functions help reduce the complexity of concatenating strings.

CAT - concatenate multiple strings
 CATT - CAT plus TRIM()
 CATS - CAT plus TRIM() plus LEFT()

CATX - CAT plus TRIM() plus LEFT()
 plus a Separator Delimiter

CATX

```
Data test;
  length a $ 10;
  a = ' Good '; b = 'Afternoon';
  v9func = CATX(" ", a, b); /* Good Afternoon */
  oldway = trim(left(a)) || ' ' || trim(b);
run;
```

COUNT

The COUNT function helps reduce the complexity of using the INDEX and SUBSTR functions to count the number of occurrences of a text within a string.

```
Data test;
  a = 'Good Afternoon. Nice to be here.';
  x_oo = count(a, 'oo'); /* x_oo = 2 */
run;
```

COMPARE

Returns the left-most character position when two strings differ or the value 0 if no difference exists.

Modifiers -
 i or I ignores the case when comparing
 l or L removes leading blanks before comparing

```
data test;
infile datalines missover;
input string1 $char8. string2 $char8.
      modifiers $char8.;
result=compare(string1, string2, modifiers);
datalines; /* results */
1234567812345678 /* 0 */
123 abc /* -1 */
abc abx /* -3 */
aBc AbC i /* 0 */
abc abc l /* 0 */
;
run;
```

MEDIAN

Computes median value from list of non-missing values. Med = median(n1, n2, n3, n4);

Obs	n1	n2	n3	n4	med
1	1	2	3	4	2.5

PCTL

Computes percentiles. pctlv = pctl(25, n1,n2,n3,n4);

Obs	n1	n2	n3	n4	pctlv
1	1	2	3	4	1.5

SYMPUTX

Automatically converts numeric to character values and strips leading and trailing blanks.

```
Data _null_;
  call symputx('v9way', 99.9);
  call symput('oldway',
              trim(left(put(99.9, 4.1))));
run;

%put &v9way &oldway;
99.9 99.9
```

What's New in ODS Formatting

The new ODS Document procedure enables users to render, with a single run, multiple ODS output formats to one or more ODS destinations without rerunning a SAS procedure or Data Step. This produces a hierarchy of output objects that can be 'replayed'.

Two Step Process:

1. Create and Save Results as Document File
2. Replay Results from Document File

Step 1: Create and Save Results as Document File

```
ODS Document name = myresults (update);
ODS RTF file = 'c:\rtf\rtf.rtf';
```

```
Proc Freq data=demog;
  tables gender;
run;
```

```
ODS RTF CLOSE;
```

The results of PROC FREQ is saved to the template store myresults as the RTF file below is created.

The FREQ Procedure

Sex				
gender	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Man	4	16.00	4	16.00
Woman	21	84.00	25	100.00

Step 2: Replay Results from Document File

```
ODS PDF file = 'c:\pdf\pdf.pdf'
style = barrettsblue;
```

```
Proc Document name = myresults (update);
  replay;
run;
```

```
ODS PDF CLOSE;
```

By access the template store myresults, users can replay the results of PROC FREQ without rerunning PROC FREQ as a different file type and different style.

The SAS System

07:25

The FREQ Procedure

Sex				
gender	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Man	4	16.00	4	16.00
Woman	21	84.00	25	100.00

In version 9.0, the import and export wizards enable easy access to Microsoft Excel 2002 spreadsheets and Microsoft Access 2002 tables. Users have the power to identify a specific spreadsheet in a workbook with the SHEET statement.

Note that this is available for Excel 97, 2000 and 2002 and requires SAS/ACCESS for PC File Formats.

```
Proc export data=product1          /* product2 */
  outfile = 'c:\excel_file\product.xls'
  dbms = excel;
  SHEET = shoes;                   /* boots */
Run;
```

OTHER USEFUL ITEMS

Along with the new features in version 8.2 and 9.0, there is one additional item worth learning about.

The Style Editor from the Enterprise Guide is an essential tool to facilitate the development of custom templates. With the nice user interface, it becomes easier to learn and modify style attributes without knowing the syntax. Although there are some limitations with this approach, it offers a very nice tool to get familiar with style attributes.

Creating Custom Styles using the Style Editor in Enterprise Guide

- Tools
- Style Editor

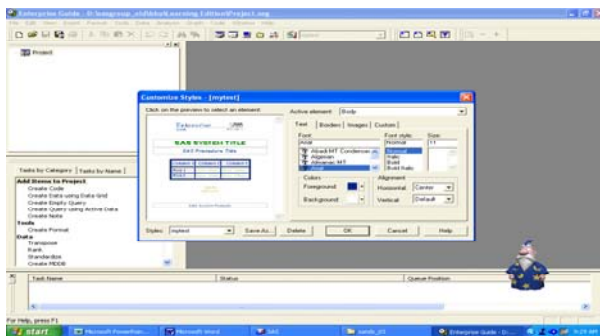
Click on preview item to select a style element

(System Title, Proc Title, Table, Header, Row Header, Data, System Footer)

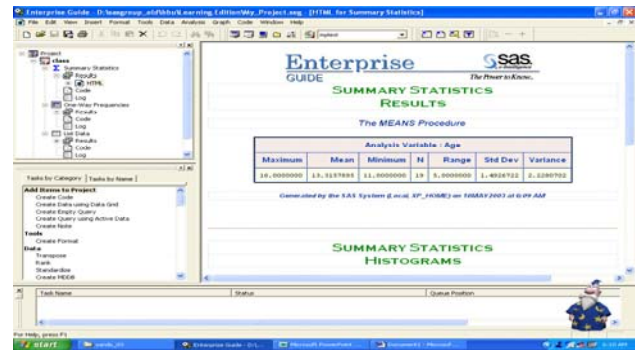
Text tab: font, font style, size, color, alignment
Border tab: attributes, color, style, width, margin, padding
Images, Custom tabs
Save as (new style)

- Tools
- Options
- Results
- Style (to set as default style)

The nice user interface makes it easy to change style attributes and see the results.



Once the custom style is created, it can be accessed to create custom output.



SUMMARY

This paper provides examples to introduce users to the new features of SAS version 8.2 and 9.0.

SAS users can now control the prevention of data entry errors, the tracking of data updates and the backing up data sets. In addition, users have more functionality with new procedures, statements and options to facilitate daily programming tasks.

REFERENCES

SAS version 8.2 and 9.0 On-Line Doc

Stroupe, Jane, "What's New in Version 7 and 8 for SAS Files?", June and July 2000, Bay Area SAS Users Group, SAS Institute Inc

Beatrous, Steve and James Holman, "Version 6 and Version 7: A Peaceful Co-Existence", SAS Institute Inc., Cary, NC

Marje, Fecht, "Making the Most of Version 9 Features", SUGI28, Prowerk Consulting

ABOUT THE AUTHOR

The author welcomes your comments & suggestions.

Sunil K. Gupta
Gupta Programming
SAS Certified Professional V6
213 Goldenwood Circle, Simi Valley, CA 93065
Phone: (805)-584-6182
E-mail: Sunil@GuptaProgramming.com
<http://www.GuptaProgramming.com>

Sunil is a principal consultant, trainer, and manager at Gupta Programming. He has been using SAS® software for over 10 years and is a SAS Certified Professional V6™. He has participated in over 6 successful FDA submissions. His consulting projects with pharmaceutical companies include the development of a Macro-Based Application for Report Generation and Customized Plots and Charts. He is also the author of Quick Results with the Output Delivery System and was a SAS Institute Quality Partner™ for over 5 years.

