SAS® GLOBAL FORUM 2020 #SASGF

Paper 5123-2020

The Curious CAT (Q, S, T, X) Functions

Jinson J. Erinjeri, Independent SAS Programmer; Saritha Bathi, Bristol-Myers Squibb;

ABSTRACT

The CAT family of functions CAT, CATQ, CATS, CATT, CATX in SAS are very useful in concatenating strings. These functions not only join strings, but also serve to write compact codes with an additional advantage of easier discernment of the same. The CAT family attain these positives by eliminating the need of complex and extended logics associated with multiple functions. In this paper, we present the various CAT functions with relevant examples of their application. The main objective of this paper is to remove the curiosity of the CAT family and encourage the programmers to add these functions as a must in their toolbox.

INTRODUCTION

The CAT functions were introduced in SAS 9.0 and the main purpose was to simplify the usage of multiple functions of TRIM, TRIMN, LEFT, RIGHT, STRIP along with the concatenation operator (||). The application of these multiple functions is used by programmers is prevalent to date. This paper re-emphasizes that the multiplicity approach can be easily substituted with the various breed of the CAT functions. In this paper, we will present the usage of these functions with simple applications and will also point to some legacy approaches which can be retired with our CAT's. In addition, we will shed some light on the important aspects of various CAT functions when presenting the applications. All applications presented this paper will be based on the dummy data set presented in Display 1.

	he_def	cpoth	cpcom	entdat	moddat	
18	H & E			01/22/13	01/22/13	CH6
19	stained H and E			05/10/12	05/10/12	CH7
20	stained H and E			04/13/11	05/10/12	CH8
21	H & E		ACR in addition to chronic biliary (inactive) disease	01/22/13	01/22/13	CH9
22	H and E			10/04/12	10/04/12	CH10
23	stained H and E			05/10/12	05/10/12	CH11
24	H and E			08/29/13	08/29/13	CH12
25	H and E			08/29/13	08/29/13	CH13
26	H & E		ACR Likely, portal and central	04/24/14	04/24/14	CH14
27	H and E		ACR is mild There is also vasculopathy	10/04/12	10/04/12	CH15
28	stained H and E			05/10/12	05/10/12	CH16
29	H & E	Possible ACR, biliary dx or drug-induced epithelial damage cannot be exclud	Possible ACR, but biliary disease or drug-induced epithelial damage cannot be excluded. There is mild ductular reactivity and CK7 ectopic staining.	01/22/13	01/22/13	CH17
30	H and E		paucity of bile ducts and fibrosis	10/04/12	10/04/12	CH18

Display 1. Snapshot of Dummy Dataset

APPLICATION 1: CAT FUNCTION

CAT function is the equivalent of concatenation operator || and is the most basic one among the CAT family. According to authors, the CAT function is the foundation for rest of the CAT family of functions. It is important to note that this operation results in string concatenation without the removal of leading or trailing blanks. The syntax for the CAT function is

CAT(item-**1<,...,item**-n>)

where item specifies a constant, variable, or expression, either character or numeric. If item is numeric, then its value is converted to a character string by using the BESTw. format. In this case, leading blanks are removed, and SAS does not write a note to the log.

Code 1 is the basic application of CAT function on strings with a comparison to concatenation operator (||). The output of code 1 and PROC CONTENTS of the resulting data set (app1) is shown in Displays 2 and 3 respectively. The compare variable (var1) in the output shows that the two results are identical. Display 3 shows that the length of variable using CAT function is 200 whereas it is 80 when using concatenation operator which is the sum of lengths of variables being concatenated.

Code 1

```
data app1;
       set malts;
       var1C=cat(ID,he_def);
       var1O=id||he_def;
       var1=compare(var1C,var1O);
run;
```

Code 2

```
data app2;
       set malts;
       var2C=cat(cpcom,he_def);
       var2O=cpcom||he_def;
       var2=compare(var2C,var2O);
run;
```

	he_def	id	var1C	var1O	var1
18	H & E	CH6	CH6 H & E	CH6 H & E	0
19	stained H and E	CH7	CH7 stained H and E	CH7 stained H and E	0
20	stained H and E	CH8	CH8 stained H and E	CH8 stained H and E	0
21	H & E	CH9	CH9 H & E	CH9 H & E	0
22	H and E	CH10	CH10 H and E	CH10 H and E	0
23	stained H and E	CH11	CH11 stained H and E	CH11 stained H and E	0
24	H and E	CH12	CH12 H and E	CH12 H and E	0
25	H and E	CH13	CH13 H and E	CH13 H and E	0
26	H & E	CH14	CH14 H & E	CH14 H & E	0
27	H and E	CH15	CH15 H and E	CH15 H and E	0
28	stained H and E	CH16	CH16 stained H and E	CH16 stained H and E	0

Display 2. Output of Code 1

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
3	cpcom	Char	181	$181.	$2000.	cpcom
2	cpoth	Char	75	$75.	$75.	cpoth
4	entdat	Num	8	MMDDYY8.	DATETIME22.3	entdat
1	he_def	Char	75	$75.	$75.	he_def
6	id	Char	5	$5.	$5.	id
5	moddat	Num	8	MMDDYY8.	DATETIME22.3	moddat
9	var1	Num	8			
7	var1C	Char	200			
8	var1O	Char	80			

Display 3. Output of PROC CONTENTS of app1 Dataset

Code 2 is a scenario with concatenation of two variables (he_def and cpcom) which has a resultant sum length of 256. The log for this operation is shown in Display 4 with a clear message on buffer allocation and invalid results. Also, note that the output data set in Display 5 shows a missing field for var2C compared to var2O. This issue can be resolved by setting appropriate lengths using the LENGTH statement when declaring the variable. Moreover, it is always advisable to verify and set the length of variable when concatenating strings.

```
57   data app2;
58       set malts;
59       var2C=cat(cpcom,he_def);
60       var2O=cpcom||he_def;
61       var2=compare(var2C,var2O);
62   run;

WARNING: In a call to the CAT function, the buffer allocated for the result was not long enough to contain the
         concatenation of all the arguments. The correct result would contain 256 characters, but the actual result might
         either be truncated to 200 character(s) or be completely blank, depending on the calling environment. The
         following note indicates the left-most argument that caused truncation.
NOTE: Argument 2 to function CAT('           '[12 of 181 characters shown],'h & e       '[12 of 75 characters shown]) at
      line 59 column 11 is invalid.
he_def=h & e cpoth=hapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR cpcom=  entdat=01/24/13
moddat=01/24/13 id=M1 var2C=  var2O=h & e var2=-182 _ERROR_=1 _N_=1
WARNING: In a call to the CAT function, the buffer allocated for the result was not long enough to contain the
         concatenation of all the arguments. The correct result would contain 256 characters, but the actual result might
         either be truncated to 200 character(s) or be completely blank, depending on the calling environment. The
         following note indicates the left-most argument that caused truncation.
NOTE: Argument 2 to function CAT('suboptimal t'[12 of 181 characters shown],'H and E     '[12 of 75 characters shown]) at
      line 59 column 11 is invalid.
he_def=H and E cpoth=  cpcom=suboptimal tissue quality entdat=02/15/12 moddat=02/15/12 id=M2 var2C=
var2O=suboptimal tissue quality
```

Display 4. Log of Code 2

	he_def	cpcom	var2C	var2O	var2
28	stained H and E			H and E	-182
29	H & E	Possible ACR, but biliary disease or drug-induced epithelial damage cannot be excluded. There is mild ductular reactivity and CK7 ectopic staining.		Possible ACR, but biliary disease or drug-induced epithelial damage cannot be excluded. There is mild ductular reactivity and CK7 ectopic staining. H & E	-1
30	H and E	paucity of bile ducts and fibrosis		paucity of bile ducts and fibrosis H and E	-1
31	H and E			and E	-182

Display 5. Output of Code 2

Code 3 shows the application with regards to concatenation of numeric data type with character. In this case, as per the conversion process, the numeric field is converted to character and the leading blanks are removed. Display 6 shows the various flavors of this concatenation with compare differences using the || operator.

Code 3

```
data app3;
  set malts;
  var3C=cat(entdat,moddat); var3O=entdat||moddat; var3=compare(var3C,var3O);
  var4C=cat(he_def,moddat); var4O=he_def||moddat; var4=compare(var4C,var4O);
  var5C=cat(moddat,he_def); var5O=moddat||he_def; var5=compare(var5C,var5O);

  var6C=cat(he_def,vvalue(moddat));
  var6O=he_def||vvalue(moddat);
  var6=compare(var6C,var6O);
run;
```

The variables var3, var4, var5 and var6 shows the comparison of the results using the CAT function and concatenating operator. It is important to note that var6 shows no differences because of the usage of VVALUE function which converts the values of its arguments into character values and the concatenation occurs after this in both var6C and var6O.

he_def	entdat	moddat	var3C	var3O	var3	var4C	var4O	var4	var5C	var5O	var5	var6C	var6O	var6
h & e	01/24/13	01/24/13	1938219382	19382 19382	1	h & e 19382	h & e 19382	76	. 19382h & e	19382h & e	1	h & e 01/24/13	h & e 01/24/13	0
H and E	02/15/12	02/15/12	1903819038	19038 19038	1	H and E 19038	H and E 19038	76	19038H and E	19038H and E	1	H and E 02/15/12	H and E 02/15/12	0

Display 6. Output of Code 3

One of the main advantages of using CAT functions is its support for the OF syntax which simplifies the coding process. For example, instead of writing CAT(var1,var2,var3,var4,var5,var6), the code can be simplified to CAT(OF var1-var6).

APPLICATION 2: CATS FUNCTION

CATS function concatenates two or more strings by removing both leading and trailing blanks. The syntax is same as the CAT function with the additional letter 'S' indicating that the blanks are stripped. Code 4 shows the usage of trimming blanks using the legacy approach of multiple functions (TRIM and LEFT) and CATS function. Note that CATS function is much more compact and usually faster.

Code 4

```
data app4;
      set malts;
      var7O=TRIM(LEFT(moddat))||TRIM(LEFT(entdat));
      var7C=cats(moddat,entdat);
      var7=compare(var7C,VAR7O);
      keep moddat entdat var7:;
run;
```

Display 7 shows the output of code 4 and there is a perfect match between var7O and var7C as shown by var7 equal with values equal to 0. However, note that the length of var7C is 200 bytes as compared to var7O. For more information on the length of returned variables using CATC function, the authors recommend referring SAS documentation [1].

modidate	entedate	var7O	var7C	var7
19382	19382	1938219382	1938219382	0
19038	19038	1903819038	1903819038	0
19326	19326	1932619326	1932619326	0
19326	19326	1932619326	1932619326	0
19389	19389	1938919389	1938919389	0
19082	19082	1908219082	1908219082	0
19169	19169	1916919169	1916919169	0
19521	19514	1952119514	1952119514	0

Display 7. Output of Code 4

APPLICATION 3: CATT FUNCTION

CATT function concatenates two or more strings by removing only trailing blanks. The syntax is same as the CAT function with the additional letter 'T' indicating that only the trailing blanks are trimmed. There are various articles which says that this is equivalent to the TRIM function and this need not be true. Code 5 and output in Display 8 shows the application of CATT as well as TRIM function.

Code 5

```
data app5;
       set malts;
       var8O=TRIM(moddat)||TRIM(entdat);
       var8C=catt(moddat,entdat);
       var8=compare(var8C,VAR8O);

        var9O=TRIM(he_def)||TRIM(cpoth);
       var9C=catt(he_def,cpoth);
       var9=compare(var9C,VAR9O);
       keep he_def cpoth mod: ent: var:;
run;
```

In Display 8, the compare variables var8 and var9 have values of 1 which shows that the TRIM function is not equivalent to CATT function and must be careful since the TRIM function returns at least one blank character. The equivalency can be obtained by using the TRIMN function which returns no blank character. Also, note that if there is a numeric or a date field, the TRIM as well as TRIMN function will not be equivalent. Neither of these functions strip leading blanks whereas the CAT family of functions does this automatically when dealing with numeric or date fields.

he_def	cpoth	entdat	moddat	var8O		var8C	var8	var9O	var9C	var9
h & e	hapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR	01/24/13	01/24/13	19382	19382	1938219382	1	h & ehapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR	h & ehapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR	0
H and E		02/15/12	02/15/12	19038	19038	1903819038	1	H and E	H and E	0
h and e	chronic biliary disease (aberrant ck7)	11/29/12	11/29/12	19326	19326	1932619326	1	h and echronic biliary disease (aberrant ck7)	h and echronic biliary disease (aberrant ck7)	0
H and E	chronic biliary disease	11/29/12	11/29/12	19326	19326	1932619326	1	H and Echronic biliary disease	H and Echronic biliary disease	0
H and E	Chronic biliary disease and extensive fibrosis	01/31/13	01/31/13	19389	19389	1938919389	1	H and EChronic biliary disease and extensive fibrosis	H and EChronic biliary disease and extensive fibrosis	0
H and N		03/30/12	03/30/12	19082	19082	1908219082	1	H and N	H and N	0
h and e		06/25/12	06/25/12	19169	19169	1916919169	1	h and e	h and e	0
H&E	chronic biliary disease	06/05/13	06/12/13	19521	19514	1952119514	1	H&Echronic biliary disease	H&Echronic biliary disease	0
H&E one slide stained		07/22/13	07/22/13	19561	19561	1956119561	1	H&E one slide stained	H&E one slide stained	0
	Can not rule out rejection; possibly modified ACR? Suspect biliary issue	05/06/14	05/06/14	19849	19849	1984919849	1	Can not rule out rejection; possibly modified ACR? Suspect biliary issue	Can not rule out rejection; possibly modified ACR? Suspect biliary issue	1
H&E		06/21/14	09/02/14	19968	19895	1996819895	1	H&E	H&E	0

Display 8. Output of Code 5

APPLICATION 4: CATX FUNCTION

The CATS function removes both leading and trailing blanks but if you need to introduce a single space, or one or more characters of choice between the strings, one can employ the CATX functions. The syntax for the CATX function is:

CATX(delimiter, item-1 <, ... item-n>)

The delimiter specifies a character string that is used as a delimiter between the concatenated items. The CATX function first copies item-1 to the result, omitting leading and trailing blanks. Next, for each subsequent argument item (ii=2, …, n), if item-i contains at least one non-blank character, CATX appends delimiter and item-i to the result, omitting leading and trailing blanks from item-i. CATX does not insert the delimiter at the beginning or end of the result. Note that blank items do not produce delimiters at the beginning or end of the result, nor do blank items produce multiple consecutive delimiters.
Code 6 shows the scenario of using a delimiter in the CATX function. The delimiter can also be a variable from a dataset as shown in var15C where the variable identifier (=999999) is used as the delimiter.

Code 6

```
data app6;
       set malts;
       identifier='999999';
       var13C=catx('-',moddat,entdat);
       var14C=catx('****',he_def,cpoth);
       var15C=catx(identifier,he_def,cpoth);
       keep he_def cpoth identifier mod: ent: var:;
run;
```

Display 9 shows various scenarios of CATX usage. Var13C has '-' between the concatenated variables moddat and entdat. For var14C, the first three observations do not have delimiters included since cpoth variable have missing values. In the fourth observation, both he_def and cpoth have missing values and hence the output is blank. Var15C has variable (identifier) from the dataset as delimiter for the first argument.

he_def	cpoth	entdat	moddat	identifier	var13C	var14C	var15C
H&E		02/06/13	02/13/13	999999	19402-19395	H&E	H&E
H&E		02/06/13	02/13/13	999999	19402-19395	H&E	H&E
H&E		01/31/14	01/31/14	999999	19754-19754	H&E	H&E
		12/17/14	12/17/14	999999	20074-20074		
H&E		02/06/13	02/13/13	999999	19402-19395	H&E	H&E
H&E		01/31/14	01/31/14	999999	19754-19754	H&E	H&E
H and E		07/20/11	05/07/12	999999	19120-18828	H and E	H and E
Tand R		05/20/13	05/20/13	999999	19498-19498	Tand R	Tand R
T and R		11/06/13	11/13/14	999999	20040-19668	T and R	T and R
h and e	No active cellular rejection, biliary changes or hepatitis	04/11/13	04/11/13	999999	19459-19459	h and e*****No active cellular rejection, biliary changes or hepatitis	h and e999999No active cellular rejection, biliary changes or hepatitis
H and E	Mild changes of active cellular rejection. ACR treated with steriods/tac	12/14/12	02/19/15	999999	20138-19341	H and E*****Mild changes of active cellular rejection. ACR treated with steriods/tac	H and E999999Mild changes of active cellular rejection. ACR treated with steriods/tac
H and E		11/25/11	09/18/14	999999	19984-18956	H and E	H and E
H/E, PAS-D, Trach, Cyto-7		08/22/13	08/22/13	999999	19592-19592	H/E, PAS-D, Trach, Cyto-7	H/E, PAS-D, Trach, Cyto-7
h an e		09/15/14	09/15/14	999999	19981-19981	h an e	h an e
H/E, TRICH, PAS-D,		08/21/13	08/21/13	999999	19591-19591	H/E, TRICH, PAS-D, RETIC,	H/E, TRICH, PAS-D, RETIC,

Display 9. Output of Code 6

APPLICATION 5: CATQ FUNCTION

The CATQ function is like CATX with the addition of quotation marks to any string that contains the specified delimiter. The Q in CATQ can refer to indicate quotation marks. The syntax for CATQ is

CATQ(modifiers <, delimiter>, item-1 **<, ...,** item-n>)

The modifier is a required argument and specifies a character constant, variable, or expression in which each non-blank character modifies the action of the CATQ function. The blanks are ignored and some of the commonly used modifiers are as follows:

1 or ' : uses single quotation marks when CATQ adds quotation marks to a string
2 or " : uses double quotation marks when CATQ adds quotation marks to a string
a or A: adds quotation marks to all the item arguments
c or C: uses comma as a delimiter
d or D: indicates that you have specified the delimiter argument
h or H: uses a horizontal tab as the delimiter
q or Q: adds quotation marks to item arguments that already contain quotation marks
s or S: strips leading and trailing blanks

The modifier is enclosed in quotation marks if it is a constant. The modifier can be expressed as a variable name or an expression. For additional options, the authors recommend referring the SAS documentation for CATQ functions [3].
The delimiter is the optional argument and specifies a character constant, variable, or expression that is used as a delimiter between concatenated strings. If this argument is specified, then the D modifier must also be specified.
Codes 7 and 8 shows various flavors of applying the CATQ functions with outputs presented in Displays 10 and 11 respectively.

Code 7

```
data app7;
  set malts;
  var17C=catq('1',he_def,cpoth);
  var18C=catq('2',he_def,cpoth);
```

Code 8

```
data app8;
  set malts;
  var20C=catq('D','****',he_def,cpoth);
  var21C=catq('S1',he_def,cpoth);
```

```
  var19C=catq('C',he_def,cpoth);
  keep he_def cpoth var:;
run;
```

```
  var22C=catq('SC',he_def,cpoth);
  keep he_def cpoth var:;
run;
```

In Code 7, variables var17C and var18C has modifiers '1' (single quotes) and '2' (double quotes) respectively. The output in Display 10 shows that these quotes are concatenated to each string and then concatenated. Var19C uses 'C' which is comma as the modifier.

he_def	cpoth	var17C	var18C	var19C
h & e	hapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR	'h & e ' 'hapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR'	"h & e " "hapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR"	h & e ,hapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR
H and E		'H and E ' ' '	"H and E " " " "	H and E ,
h and e	chronic biliary disease (aberrant ck7)	'h and e ' 'chronic biliary disease (aberrant ck7) '	"h and e " "chronic biliary disease (aberrant ck7) "	h and e ,chronic biliary disease (aberrant ck7)
H and E	chronic biliary disease	'H and E ' 'chronic biliary disease '	"H and E " "chronic biliary disease "	H and E ,chronic biliary disease
H and E	Chronic biliary disease and extensive fibrosis	'H and E ' 'Chronic biliary disease and extensive fibrosis '	"H and E " "Chronic biliary disease and extensive fibrosis "	H and E ,Chronic biliary disease and extensive fibrosis
H and N		'H and N ' ' '	"H and N " " " "	H and N ,

Display 10. Output of Code 7

Variable var20C has 'D' as the modifier with '****' as the delimiter and note that the delimiters are concatenated even if there are missing values in the variables as shown in Display 11. Variable var21C has 'S1' as the modifier where 'S' and '1' stands for strip and single quotes respectively. In this case leading and trailing blank are removed and then the quotes are added. Variable 22C is an example to show how CATQ function can be improvised to cater to ones needs. In this example, the modifier 'SC' is removing leading and trailing blanks and then comma is used as the delimiter.

he_def	cpoth	var20C	var21C	var22C
h & e	hapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR	h & e ****hapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR	'h & e' 'hapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR'	h & e,hapatetic pattern oncocytic/polyglucogen-like inclusions not supportive ACR
H and E		H and E ****	'H and E'	H and E
h and e	chronic biliary disease (aberrant ck7)	h and e ****chronic biliary disease (aberrant ck7)	'h and e' 'chronic biliary disease (aberrant ck7)'	h and e,chronic biliary disease (aberrant ck7)
H and E	chronic biliary disease	H and E ****chronic biliary disease	'H and E' 'chronic biliary disease'	H and E,chronic biliary disease
H and E	Chronic biliary disease and extensive fibrosis	H and E ****Chronic biliary disease and extensive fibrosis	'H and E' 'Chronic biliary disease and extensive fibrosis'	H and E,Chronic biliary disease and extensive fibrosis
H and N		H and N ****	'H and N'	H and N
h and e		h and e ****	'h and e'	h and e
H&E	chronic biliary disease	H&E ****chronic biliary disease	H&E 'chronic biliary disease'	H&E,chronic biliary disease
H&E one slide stained		H&E one slide stained ****	'H&E one slide stained'	H&E one slide stained

Display 11. Output of Code 8

CONCLUSION

The CAT family of functions are very useful in writing compact and readable codes. The various options available among and within these functions helps a programmer to maximize the usage to different programmatic challenges. The authors recommend the

programmers to add CAT functions as a must in their toolbox.

REFERENCES

https://documentation.sas.com/?docsetId=lefunctionsref&docsetTarget=n1e21rr6al5m2nn19r1fat5qxwrt.htm&docsetVersion=9.4&locale=en

https://documentation.sas.com/?docsetId=lefunctionsref&docsetTarget=n0p7wxtk0hvn83n1pveisbcp2ae9.htm&docsetVersion=9.4&locale=en

https://documentation.sas.com/?docsetId=lefunctionsref&docsetTarget=n11fqp8lmm22sen1cxlzksrtfefo.htm&docsetVersion=9.4&locale=en

SAS Online Documents. Available at http://support.sas.com/kb/46/672.html

CONTACT INFORMATION

Your comments/questions/criticisms are valued and encouraged. Please contact the authors at:

Jinson Erinjeri
Independent SAS Programmer
Email: jinson_je@yahoo.com

Saritha Bathi
Bristol-Myers Squibb
86 Morris Avenue,
Summit, NJ 07901
E-mail: sabathi@celgene.com