Paper 1786-2014

Tips to Use Character String Functions in Record Lookup

Anjan Matlapudi
Corporate Medical Economics
AmeriHealth Caritas Family of Companies
3rd Floor, 200 Stevens Drive,Philadelphia, PA 19113

ABSTRACT

This paper gives you a better idea of how and where to use the record lookup functions to locate observations where a variable has some characteristic. Various related functions are illustrated to search numeric and character values in this process. Code is shown with time comparisons. I will discuss possible three ways to retrieve records using in SAS® DATA step, PROC SQL and Perl regular expression. Real and CPU time processing issues will be highlighted when comparing to retrieve records using these methods.

Program was written for the PC using SAS 9.2 on Windows XP 62 bit environment. All the tools discussed are in BASE SAS®. The typical attendee or reader will have some experience in SAS, but not a lot of experience dealing with large number of data.

INTRODUCTION

A common task is to look for a particular field in databases ranging in size from less than a million to more than a billion. SAS Character functions are required to do this. I ask myself whether all of these functions would work in DATA step, PROC SQL and Perl Regular Expression. In this paper, I will introduce the code in several tables solving different problems for each function. In some cases PROC SQL is omitted because conditional use of the functions. This way many of us would be aware of how best we can use some of the commonly used functions in several ways.

To make the tests I used the following code to randomly generate 10 million observations with character and numeric variables using RANUIN function. DO LOOP =1 to 10e7 will generate 10 million records, but this code can generate any number of records by changing log exponential. I tested most of the functions to find one or many records out of the 10 million and noted processing time to read, modify and write character variables.

CREATE TEST RECORDS USING RANUNI FUNCTION:

```
*----Test data-----*;
data TenMillonRecs;
format FirstName LastName $10. MiddleName $1. Name $25.  SSN 9.;
string1="abcdefghijklmnopqrstuvwxyz";
  string2="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
     string3=reverse("abcdefghijklmnopqrstuvwxyz");
        stringlen=length(string1);
         do i=1 to 1e7;
          random=ranuni(95959);
            FirstName =''; LastName = '' ; MiddleName = '';
              length=int(ranuni(0)*3)+8; *int truncate decimal point ;
                  do j=1 to length;
                     pick=int(ranuni(0)* stringlen)+1;
                      FirstName=substr(string1,pick,1)||FirstName;
                     MiddleName = substr(string2,1,1);
                    LastName=substr(string3,pick,1)||LastName;
                  Name = propcase(FirstName||" "||MiddleName||" "||LastName);
            SSN = input(compress(put(1e9*random,z9.)),9.);
           end;
```

```
        output;
      end;
  keep ssn Name;
run;

*---print 10 records out of 10 million Records---*;
proc print data= TenMillonRecs (obs=10);
title  "First 10 Records out of 10 Million Test Records";
run;
```

DATA STEP AND PROC SQL:

SAS ® is an excellent tool to accommodate many functions in many ways; it has flexibility to use these functions in SAS for programmers with different sets of skills. SAS implemented SQL (PROC SQL) in version 6.0 and Perl Script in version 9.0 for better flexibility to retrieve information. In this paper, I will touch base some of the functions for records look up and also highlight the real and CPU (Central Processing Unit) time taken to run each function in the same environment using DATA step and PROC SQL. In most cases the function name itself describes its role; however, I highlighted purpose of each function in the bellow tables. If you want to know more detail, you can always approach SAS help or online documentation.

The code bellow shows some of the highlighted functions in DATA step and PROC SQL. The next column for each function represents return value, real/ CPU time in seconds. All these functions are tested on same environment, each statement run several times for processing the real time comparison.

FUNCTION	RETURNS	REALTIME/CPU
LIKE		
Purpose: Search option for specific string of characters.		

`data Like_out;`
`set TenMillonRecs;`
`where name` **`like`** `'Anjam%';`
`run;`
	Anjamvfk A
	Zmqzneup
	440019853
		1.56/1.42

`proc sql;`
`create table Like_tbl as`
`select * from TenMillonRecs`
`where name` **`like`** `'Anjam%';`
`quit;`
	Anjamvfk A
	Zmqzneup
	440019853
		1.93/2.67

SUBSTR		
Purpose: Extracts part of the string specified by the start and length parameters.		

`data SubStr_out;`
`set TenMillonRecs;`
`where` **`substr`**`(name,1,5)='Anjam';`
`run;`
	Anjamvfk A
	Zmqzneup
	440019853
		3.43/2.79

`proc sql;`
`create table SubStr_tbl as`
`select * from TenMillonRecs`
`where` **`substr`**`(name,1,5)= 'Anjam';`
`quit;`
	Anjamvfk A
	Zmqzneup
	440019853
		3.59/2.68

FIND

Purpose: Locate substring with in a string.

```
data Find_out;
set TenMillonRecs;
where find(name,'Anjamvfk   A
Zmqzneup')=1;
run;
```

```
Anjamvfk   A
Zmqzneup
440019853
```

3.93/2.29

```
proc sql;
create table find_tbl as
select * from TenMillonRecs
where find(name,'Anjamvfk   A
Zmqzneup')=1;
quit;
```

```
Anjamvfk   A
Zmqzneup
440019853
```

4.87/2.45

INDEX

Purpose: To locate starting portion of substring of a string.

```
data Index_out;
set TenMillonRecs;
where index(name,'Anjam')=1;
run;
```

```
Anjamvfk   A
Zmqzneup
440019853
```

4.75/2.53

```
proc sql;
create table Index_tbl as
select * from TenMillonRecs
where index(name,'Anjam')=1;
quit;
```

```
Anjamvfk   A
Zmqzneup
440019853
```

5.01/2.64

SCAN

Purpose: Extracts specified word from a character string.

```
data Scan_out;
length FirstName LastName $10. MiddleName
$1.;
set TenMillonRecs;
FirstName =scan(name,1,'');
MiddleName =scan(name,2,'');
LastName =scan(name,3,'');
/*where name ='Anjamvfk   A Zmqzneup';[1]*/
run;
```

```
10000000
observations and 5
variables
```

32.12/6.71

```
proc sql;
create table Scan_tbl as
select name,ssn,
scan(name,1,' ') as firstName,
scan(name,3,' ') as MiddleName,
```

```
10000000 rows and
5 columns.
```

4.06.03/30.92

[1] We can test all statements to read, modify and write one record using where clause option or we can retrieve 10 million records by removing where clause.

```
scan(name,2,' ') as LastName
from TenMillonRecs
/*[1]where name ='Anjamvfk   A Zmqzneup';*/;
quit;
```

TRANSLATE

Purpose: To exchange on character value to another.

```
data Translate_out (keep=result);
set TenMillonRecs;
result=
translate(name,'Anjan','Anjamvfk');
where name ='Anjamvfk   A Zmqzneup';
run;
```

```
Anjan      A
ZnqzneupA
```

0.87/0.85

```
proc sql;
create table Translate_tbl as
select
translate(name,'Anjan','Anjamvfk') as
result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

```
Anjan      A
Znqzneup
```

0.88/0.90

CAT

Purpose: Combine two strings.

```
data Cat_out (keep=result);
set TenMillonRecs;
result= cat(name,'is' ,' funny name');
where name ='Anjamvfk   A Zmqzneup';
run;
```

```
Anjamvfk   A
Zmqzneup    is
funny name
```

0.87/0.87

```
proc sql;
create table Cat_tbl as
select cat(name,'is' ,' funny name') as
result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

```
Anjamvfk   A
Zmqzneup    is
funny name
```

0.87/0.89

VERIFY

Purpose: Returns the position of the first character in a string that is not in any of several other strings.

```
data Verify_out (keep=result);
set TenMillonRecs;
result= verify(name,'Anjam');
where name ='Anjamvfk   A Zmqzneup';
run;
```

6

1.30/0.95

```
proc sql;
create table Verify_tbl as
select verify(name,'Anjam') as result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit
```

6

1.49/0.92

TRIM

Purpose: Removes trailing blanks from a character string.

```
data Trim_out;
set TenMillonRecs;
name= trim(name);
run;
```

```
10000000
observations and 2
variables
```

39.61/4.15

```
proc sql;
create table Trim_tbl as
select trim(name) as name,ssn
from TenMillonRecs;
quit;
```

```
10000000 rows and
2 columns
```

15.96/5.48

STRIP

Purpose: To strip leading or trailing blanks from character string.

```
data Strip_out(keep=result);
set TenMillonRecs;
if name= 'Anjamvfk   A Zmqzneup' then
name1 = '          Anjamvfk';
result = strip(name1) ;
where name ='Anjamvfk   A Zmqzneup';
run;

Proc SQL is omitted.
```

```
Anjamvfk
```

0.87/0.87

RIGHT

Purpose: Align right side of character string.

```
data Right_out(keep= name1 result);
set TenMillonRecs;
if name= 'Anjamvfk   A Zmqzneup' then
name1 = 'Anjamvf          ';
result = right(name1) ;
where name ='Anjamvfk   A Zmqzneup';
run;

Proc SQL is omitted.
```

```
‘Anjamvfk
’
          Anjamvfk
```

0.89/0.90

LEFT

Purpose: Align left side of character string.

```
data Left_out(keep= name1 result);
set TenMillonRecs;
if name= 'Anjamvfk   A Zmqzneup' then
name1 = '         Anjamvf';
result = left(name1) ;
where name ='Anjamvfk   A Zmqzneup';
run;
proc print data =left_out;
run;

Proc SQL is omitted.
```

```
‘
Anjamvfk‘
Anjamvfk
```

0.87/0.87

COMPRESS

Purpose: Remove specified character value (including blanks) from a string.

```
data Compress_out;
set TenMillonRecs;
name= compress(name);
run;
```

```
10000000 observations and 2 variables
```

9.59/4.82

```
proc sql;
create table Compress_tbl as
select compress(name) as name,ssn
from TenMillonRecs;
quit;
```

```
10000000 rows and 2 columns
```

47.37/6.12

COMPBL

Purpose: Replace two or more blanks with single blank.

```
data Compbl_out;
length result $100.;
set TenMillonRecs;
result= compbl(name);
where name ='Anjamvfk   A Zmqzneup';
run;
```

```
Anjamvfk A Zmqzneup
```

0.90/0.89

```
proc sql;
create table Compbl_tbl as
select compbl(name) as result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

```
Anjamvfk A Zmqzneup
```

1.29/0.89

UPCASE

Purpose: Convert all letters to upper case.

```
data Upcause_out(keep=result);
set TenMillonRecs;
result= upcase(name);
where name ='Anjamvfk   A Zmqzneup';
run;
```

```
ANJAMVFK   A ZMQZNEUP
```

0.87/0.87

```
proc sql;
create table Upcause_tbl as
select upcase(name) as result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

```
ANJAMVFK   A ZMQZNEUP
```

0.87/0.87

LOWCASE

Purpose: Converts all letters to lower case.

```
data Lowcause_out(keep=result);
set TenMillonRecs;
result= lowcase(name);
where name ='Anjamvfk   A Zmqzneup';
run;
```

```
anjamvfk   a zmqzneup
```

0.86/0.86

```
proc sql;
create table Lowcause_tbl as
select lowcase(name) as result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

anjamvfk zmqzneup | a | 0.89/0.89

PROPCASE

Purpose: Capitalize the first letter of each word in a string.

```
data Propcase_out(keep=result);
set TenMillonRecs;
result= propcase(name);
where name ='Anjamvfk   A Zmqzneup';
run;
```

Anjamvfk Zmqzneup | A | 0.92/0.89

```
proc sql;
create table Propcase_tbl as
select propcase(name) as result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

Anjamvfk Zmqzneup | A | 0.89/0.87

ANYSPACE

Purpose: To locate first occurrence of white space.

```
data Anyspace_out (keep=result);
set TenMillonRecs;
result = anyspace(name) ;
where name ='Anjamvfk   A Zmqzneup';
run;
```

9 | 0.87/0.87

```
proc sql;
create table Anyspace_tbl as
select anyspace(name) as result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

9 | 1.34/0.92

FIRST

Purpose: Extracts the first character from a string.

```
data First_out(keep=result);
set TenMillonRecs;
result= first(name);
where name ='Anjamvfk   A Zmqzneup';
run;
```

A | 0.90/0.89

```
proc sql;
create table First_tbl as
select first(name) as name,ssn
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

A | 0.89/0.89

TRANWRD

Purpose: Substitute one or more words in a string.

```
data Tranwrd_out (keep=result);
set TenMillonRecs;
result = tranwrd(name,'Anjamvfk   A
Zmqzneup','Anjan Matlapudi') ;
where name ='Anjamvfk   A Zmqzneup';
run;
```

```
Anjan Matlapudi      2.32/0.96
```

```
proc sql;
create table Tranwrd_tbl as
select tranwrd(name,'Anjamvfk   A
Zmqzneup','Anjan Matlapudi') as result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

```
Anjan Matlapud       2.42/0.96
```

LENGTH

Purpose: Determine length of a character value (not counting trailing blanks).

```
data Length_out;
set TenMillonRecs;
result = length(name) ;
where name ='Anjamvfk   A Zmqzneup';
run;
```

```
Anjamvfk   A
Zmqzneup             2.33/0.89
440019853      21
```

```
proc sql;
create table Length_tbl as
select name,ssn,length(name) as result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

```
Anjamvfk   A
Zmqzneup             2.29/0.89
440019853      21
```

REVERSE

Purpose: Reverse the order of character string.

```
data Reverse_out(keep=result);
set TenMillonRecs ;
result = reverse(name) ;
where name ='Anjamvfk   A Zmqzneup';
run;
```

```
puenzqmZ A
kfvmajnA             0.83/0.84
```

```
proc sql;
create table Reverse_tbl as
select reverse(name) as result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

```
puenzqmZ A           0.89/0.87
kfvmajnA
```

REPEATE

Purpose: Make several copies of a string.

```
data Repeat_out(keep=result);
set TenMillonRecs ;
result = repeat(name,3) ;
where name ='Anjamvfk   A Zmqzneup';
run

proc sql;
create table Repeat_tbl as
select repeat(name,3) as result
from TenMillonRecs
where name ='Anjamvfk   A Zmqzneup';
quit;
```

```
Anjamvfk   A
Zmqzneup
Anjamvfk   A
Zmqzneup
Anjamvfk   A
Zmqzneup
Anjamvfk   A
Zmqzneup
Anjamvfk   A
Zmqzneup
Anjamvfk   A
Zmqzneup
Anjamvfk   A
Zmqzneup
```

0.95/0.85

0.87/0.87

SPEDIS

Purpose: Computes spelling distance between words.

```
data Spedis_out;
set TenMillonRecs;
if name = 'Anjamvfk   A Zmqzneup' then
Name = 'Knowledge';
if  spedis(name,'nowledge' )le 27 then
output;
run;

Proc SQL is omitted
```

```
Knowledge
440019853
```

35.27/30.79

LAG

Purpose: To obtain previous value from the current character variable.

```
data Lag_out(keep=result);
set TenMillonRecs;
result = lag(name) ;
run;

Note: LAG does not work in Proc SQL.
```

```
10000000
observations and 1
variables
```

13.73/2.87

COUNT

Purpose: Counts number of times in a given substring in a string.

```
data Count_out(keep=result);
length name $100.;
set TenMillonRecs;
if name = 'Anjamvfk   A Zmqzneup' then
name = 'Random generated Random Name in
Random data';
result = count(Name,'Random') ;
where name ='Anjamvfk   A Zmqzneup';
run;

Proc SQL is omitted.
```

```
3
```

087/0.86

CHOOSEC

Purpose: Returns a character value that represents the results of choosing from a list of arguments.

```
data Choosec_out(keep=result);
set TenMillonRecs;
if name= 'Anjamvfk   A Zmqzneup' then
name1 = 'Anjan';
else if name = 'Kxlgoqhma  A Pcotljsnz'
then name2 = 'Matlapudi';
else if name = 'Taosezswy  A Gzlhvahdb'
then name3 = 'Anjan Matlapudi';
result = Choosec(3,name1,name2,name3) ;
where name in('Anjamvfk   A Zmqzneup',
'Kxlgoqhma  A Pcotljsnz','Taosezswy  A
Gzlhvahdb');
run;
Proc SQL is omitted.
```

```
Anjan Matlapudi      7.01/1.23
```

PERL REGULAR EXPERSION:

Perl Regular Expression in SAS has wide variety of functionality while working with matching patterns, text manipulation including validation and text replacement. Each variable holds 1-32676 bytes long as a charter string. I have tested some of the character string functions using 32676 bytes long character variable and I can able to successfully returned values of each function (Data not included). PRXPARSE and other functions have great deal while we are working with text manipulation. Bellow mentioned prx functions show how to retrieve records in DATA step and PROC SQL.

FUNCTION	RETURNS	REALTIME/CPU

PRXMATCH

Purpose: Searches for a pattern match and returns the position at which the pattern is found.

```
data PrxMatch_out;
   set TenMillonRecs ;
   if prxmatch("m/Anjamvfk/oi",name)> 0
then x =1;
   else x =0;
run;
```

```
10000000 observations
with x =1 record and
remaining observation x
= 0
```

```
36.37/0.79
```

```
Proc sql;
create table PrxMatch_tbl as
select name,ssn,
prxmatch("m/Anjamvfk/oi",name) as x
from TenMillonRecs;
quit;
```

```
10000000 rows with
x =1 row and
remaining x = 0
```

```
37.02/6.26
```

PRXPARSE AND PRXSUBSTR

Purpose: Perl regular expression (PRX) can be used for substring of character string matching using PRXPARSE.

```
data PrxSubstr_out;
set TenMillonRecs;
 if _n_=1 then do;
    retain re;
       re = prxparse('/\w \w.+/');
        if missing(re) then do;
               stop;
           end;
        end;
      call
  prxsubstr(re,name,start,length);
   FirstName = substrn(name,start -11,
length -4);
  LastName = substrn(name,start+2,
length +1);
if start > 0 then output;
run;
```

```
Anjamvfk   A
Zmqzneup 440019853
1
12
14
Anjamvfk
Zmqzneup
With 10000000
observations
```

58.61/11.90

RECORD LOOKUP BY CHARACTER STRING vs. NUNBER FIELD:

I further demonstrated records retrieval using character and number fields as mentioned bellow. Real time and CPU time is noted to read one record out of 10 million.

VARIABLE	RETURNS	REALTIME/CPU
CHARACTER		

`proc print data = TenMillonRecs;`
`where name ='Anjamvfk A`
`Zmqzneup';`
`run;`
	Anjamvfk A Zmqzneup
	440019853
		0.95/0.90

`proc sql;`
`select * from TenMillonRecs`
`where name ='Anjamvfk A Zmqzneup';`
`quit;`
	Anjamvfk A Zmqzneup
	440019853
		0.96/0.89

NUMERIC		

`proc print data = TenMillonRecs;`
`where ssn =440019853;`
`run;`
	Anjamvfk A Zmqzneup
	440019853
		0.92/0.84

`proc sql;`
`select * from TenMillonRecs`
`where ssn =440019853;`
`quit;`
	Anjamvfk A Zmqzneup
	440019853
		0.93/0.84

CONCLUSION

So far I have demonstrated possible ways for record look up using DATA step, PROC SQL and Perl regular expression. I further generalize some tips using these functions for record look up.

- If you already know SQL, you will be pleased to know that you can use most of the functions in PORC SQL to create, read and modify variables in SAS data sets.

- Flexibility of character string functions is available while you are working with small to a large scale data. You can use all these functions in DATA step and PROC SQL. All these functions would also work in Oracle, SQL server and Access database with slight syntax modification in code.

- It is interesting to note that Perl Regular Expression is not limited to Data Step, we can use these functions in PROC SQL where ever is possible.

- Real time processing will give best suitable option to choose some of the functions. Based on the above queries, data step processing has been taken less time when compare PROC SQL.

By now you may have some clue that all most all the functions work in DATA step and PROC SQL and if you are dealing with large scale data, you can easily pick up the best suitable function in terms of time taken to run each function. Also, I have included most commonly used functions are at one place; some of you may take advantage instead of spending more time finding them.

REFERENCES

Ronald Cody, An Introduction to SAS Character Functions. Coder's Corner Paper 217-2007.

Paul A. Choate Quick Record Lookup without an index. *Tutorials*, Paper 031-2007.

David L. Cassel. The Basics of the PRX Functions – SAS, Coder's Corner Paper. Paper 223-2007.

ACKNOWLEDGMENTS

I would like to acknowledge Tom Donia, VP Corporate Medical Economics, AmeriHealth Caritas, and Roy Oaks, Director, Corporate Medical Economics, AmeriHealth Caritas for their support and encouragement.

Amerihealth Caritas is the nation's leader in the health care solutions with more than 30 years of experience managing care for individuals and families in publicly funded program.

CONTACT INFORMATION:

Your comments and questions are valued and encouraged. Contact the author at

Name Anjan Matlapudi
Corporate Medical Economics
Address 3rd Floor, 200 Stevens Drive
Philadelphia, PA 19113
Work Phone: (215)937-7252
Fax: (215)863-5100
E-mail: amatlapudi@amrihealthCaritas.com anjanmat@gmail.com