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 RANUIN 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;
     end;
     Name = propcase(FirstName||" "||MiddleName||" "||LastName);
     SSN = input(compress(put(1e9*random,z9.)),9.);
   end;
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 below tables. If you want to know more detail, you can always approach SAS help or online documentation.

The code below 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.

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURNS</th>
<th>REALTIME/CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIKE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purpose: Search option for specific string of characters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data Like_out;</td>
<td>Anjamvfk A</td>
<td></td>
</tr>
<tr>
<td>set TenMillonRecs;</td>
<td>Zmqzneup</td>
<td></td>
</tr>
<tr>
<td>where name like 'Anjam%';</td>
<td>440019853</td>
<td>1.56/1.42</td>
</tr>
<tr>
<td>run;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>proc sql;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>create table Like_tbl as</td>
<td></td>
<td></td>
</tr>
<tr>
<td>select * from TenMillonRecs</td>
<td>Anjamvfk A</td>
<td></td>
</tr>
<tr>
<td>where name like 'Anjam%';</td>
<td>Zmqzneup</td>
<td>1.93/2.67</td>
</tr>
<tr>
<td>quit;</td>
<td>440019853</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBSTR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purpose: Extracts part of the string specified by the start and length parameters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data SubStr_out;</td>
<td>Anjamvfk A</td>
<td></td>
</tr>
<tr>
<td>set TenMillonRecs;</td>
<td>Zmqzneup</td>
<td></td>
</tr>
<tr>
<td>where substr(name,1,5)='Anjam';</td>
<td>440019853</td>
<td>3.43/2.79</td>
</tr>
<tr>
<td>run;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>proc sql;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>create table SubStr_tbl as</td>
<td></td>
<td></td>
</tr>
<tr>
<td>select * from TenMillonRecs</td>
<td>Anjamvfk A</td>
<td></td>
</tr>
<tr>
<td>where substr(name,1,5)='Anjam';</td>
<td>Zmqzneup</td>
<td>3.59/2.68</td>
</tr>
<tr>
<td>quit;</td>
<td>440019853</td>
<td></td>
</tr>
</tbody>
</table>
**FIND**

**Purpose:** Locate substring with in a string.

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

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

**INDEX**

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

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

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

**SCAN**

**Purpose:** Extracts specified word from a character string.

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

```plaintext
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 TenMillionRecs
/

QUIT

TRANSLATE

Purpose: To exchange on character value to another.

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

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

CAT

Purpose: Combine two strings.

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

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

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 TenMillionRecs;
result=
verify(name,'Anjam');
where name = 'Anjamvfk   A Zmqzneup';
run;

proc sql;
create table Verify_tbl as
select verify(name,'Anjam') as result
from TenMillionRecs
where name = 'Anjamvfk   A Zmqzneup';
quilt;
### TRIM

**Purpose:** Removes trailing blanks from a character string.

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

10000000 observations and 2 variables

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

10000000 rows and 2 columns

### STRIP

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

```sas
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.

### RIGHT

**Purpose:** Align right side of character string.

```sas
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.

### LEFT

**Purpose:** Align left side of character string.

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

Proc SQL is omitted.
**COMPRESS**

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

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

10000000 observations and 2 variables

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

10000000 rows and 2 columns

**COMPBL**

**Purpose:** Replace two or more blanks with single blank.

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

Anjamvfk A 0.90/0.89

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

Anjamvfk A 1.29/0.89

**UPCASE**

**Purpose:** Convert all letters to upper case.

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

ANJAMVFK   A 0.87/0.87

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

ANJAMVFK   A 0.87/0.87

**LOWCASE**

**Purpose:** Converts all letters to lower case.

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

anjamvfk a 0.86/0.86

```plaintext
proc sql;
create table Lowcase_tbl as
select lowcase(name) as result
from TenMillonRecs
where name = 'Anjamvfk   A Zmqzneup';
quit;
```
proc sql;
cREATE TABLE Lowcause_tbl AS
SELECT lowcase(name) AS result
FROM TenMillonRecs
WHERE name = 'Anjamvfk   A Zmqzneup';
QUIT;

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

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;

proc sql;
cREATE TABLE Propcase_tbl AS
SELECT propcase(name) AS result
FROM TenMillonRecs
WHERE name = 'Anjamvfk   A Zmqzneup';
QUIT;

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;

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

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;

proc sql;
cREATE TABLE Anyspace_tbl AS
SELECT anyspace(name) AS result
FROM TenMillonRecs
WHERE name = 'Anjamvfk   A Zmqzneup';
QUIT;

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;

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

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;

proc sql;
cREATE TABLE First_tbl AS
SELECT first(name) AS name, ssn
FROM TenMillonRecs
WHERE name = 'Anjamvfk   A Zmqzneup';
QUIT;

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;

PROC SQL
create table First_tbl as
select first(name) as name, ssn
from TenMillonRecs
where name = 'Anjamvfk   A Zmqzneup';
quit;
TRANWRD

**Purpose:** Substitute one or more words in a string.

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

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;
```

LENGTH

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

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

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

REVERSE

**Purpose:** Reverse the order of character string.

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

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

**Purpose:** Make several copies of a string.

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

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

**SPEDIS**

**Purpose:** Computes spelling distance between words.

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

**LAG**

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

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

Note: LAG does not work in Proc SQL.

**COUNT**

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

```plaintext
data Count_out(keep=result);
length name $100.;
set TenMillionRecs;
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.
CHOOSEC

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

data Choosec_out(keep=result);
set TenMillionRecs;
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.

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.

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURNS</th>
<th>REALTIME/CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRXMATCH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

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

37.02/6.26

36.37/0.79
PRXPARSE AND PRXSUBSTR

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

```plaintext
data PrxSubstr_out;
set TenMillonRecs;
if _n_=1 then do;
  retain re;
  re = prxparse('/\w \w.+/'); Anjamvfk   A Zmqzneup 440019853 1 12 14
  if missing(re) then do;
    stop;
  end;
end;
call prxsubstr(re,name,start,length); Anjamvfk   A Zmqzneup 440019853 58.61/11.90
  FirstName = substrn(name,start -11, With 10000000 observations
      length -4);
  LastName = substrn(name,start+2, observations
      length +1);
  if start > 0 then output;
run; Anjamvfk   A Zmqzneup
```

RECORD LOOKUP BY CHARACTER STRING vs. NUMBER FIELD:

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

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>RETURNS</th>
<th>REALTIME/CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>proc print data = TenMillonRecs; where name = 'Anjamvfk   A Zmqzneup'; run;</td>
<td>Anjamvfk   A Zmqzneup 440019853</td>
<td>0.95/0.90</td>
</tr>
<tr>
<td>proc sql; select * from TenMillonRecs where name = 'Anjamvfk   A Zmqzneup'; quit;</td>
<td>Anjamvfk   A Zmqzneup 440019853</td>
<td>0.96/0.89</td>
</tr>
<tr>
<td>NUMERIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>proc print data = TenMillonRecs; where ssn = 440019853; run;</td>
<td>Anjamvfk   A Zmqzneup 440019853</td>
<td>0.92/0.84</td>
</tr>
<tr>
<td>proc sql; select * from TenMillonRecs where ssn = 440019853; quit;</td>
<td>Anjamvfk   A Zmqzneup 440019853</td>
<td>0.93/0.84</td>
</tr>
</tbody>
</table>
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 PROC 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


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

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are trademarks of their respective companies.