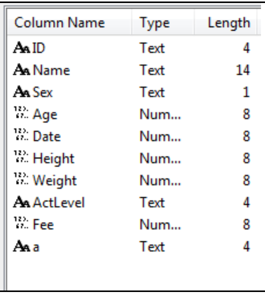
1. SUBSTR

The substr function extracts a text from the character variable, and the length of the new var is same as that of the string, it takes 3 arguments string name, start position and length.

* If the **substr** creates a new variable then the length of the new variable would be same as that of parent variable;

```
data a;
set sasuser.admit;
a= substr(actlevel,2,2);
run;
```



Column Name	Type	Length
ID	Text	4
Name	Text	14
Sex	Text	1
Age	Num...	8
Date	Num...	8
Height	Num...	8
Weight	Num...	8
ActLevel	Text	4
Fee	Num...	8
a	Text	4

2. Substr on left side

* The **substr** function extracts a text from the character variable, it could also be used to replace the values, when it is on the left side it assigns
the new values, suppose you need to change the first two values from 91 to 00, you can assign by putting it to left side;

```
data a;
x='(91) 9999265789';
substr(x,2,2)='00';
run;

proc print data=a;
run;
```

```
   The SAS System     14:52 Wednesday,
January 7, 2009    1

                                    Obs
x

                                      1
(00) 9999265789
```

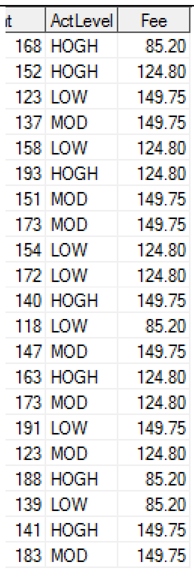
2a. Substr left side condition

* The **substr** function extracts a text from the character variable, it could also be used to replace the values, when it is on the left side it assigns
the new values, suppose you need to change the first two values from 91 to 00, you can assign by putting it to left side. it is similar to the **tranwrd** and **translate** funcntion;

*You can use the **substr** function conditionally changing the values;

*Let's say the aim is just to replace the HI to HO, so keeping LOW and MOD untouched;

```
data a;
set sasuser.admit;
if substr(actlevel,1,2)='HI' then do;
substr(actlevel,1,2)='HO';
end;
run;
```



t	ActLevel	Fee
168	HOGH	85.20
152	HOGH	124.80
123	LOW	149.75
137	MOD	149.75
158	LOW	124.80
193	HOGH	124.80
151	MOD	149.75
173	MOD	149.75
154	LOW	124.80
172	LOW	124.80
140	HOGH	149.75
118	LOW	85.20
147	MOD	149.75
163	HOGH	124.80
173	MOD	124.80
191	LOW	149.75
123	MOD	124.80
188	HOGH	85.20
139	LOW	85.20
141	HOGH	149.75
183	MOD	149.75

3. Substr counts blanks too

```
* The substr function extracts a text
from the character variable, it counts
blanks too, trying to get two chars would
```

```
The SAS System      14:52 Wednesday, January 7,
2009    3

    Obs            x             phone
```

```
return just one as blank is counted;


data a;
x='(91) 9999265789';
phone=substr(x,5,2);
run;

proc print data=a;
run;
```

```
     1     (91) 9999265789      9
```

4. Substr no third argument

```
* The substr function extracts a text
from the character variable, and if no
third argument then it would select the
whole string from that point;


data a;
x='(91) 9999265789';
phone=substr(x,6);
run;

proc print data=a;
run;
```

```
The SAS System     14:52 Wednesday, January 7,
2009   3

    Obs           x            phone

     1     (91) 9999265789   9999265789
```

5. Substr n value gt string length

```
* The substr function extracts a text
from the character variable, if n value
is gt string length then sas throws an
error;


data a;
x='(91) 9999265789';
phone=substr(x,6,11);
run;
```

```
55    data a;
56    x='(91) 9999265789';
57    phone=substr(x,6,11);
58    run;

NOTE: Invalid third argument to function SUBSTR
at line 57 column 7.
x=(91) 9999265789 phone=9999265789 _ERROR_=1
_N_=1
NOTE: The data set WORK.A has 1 observations
and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.06 seconds
      cpu time            0.03 seconds
```

6. Substr n do not take negative values

```
* The substr function extracts a text
from the character variable, if the
second argument is -ve then error;

data a;
x='(91) 9999265789';
phone=substr(x,-6,11);
run;
```

```
81    data a;
82    x='(91) 9999265789';
83    phone=substr(x,-6,11);
84    run;
NOTE: Invalid second argument to function
SUBSTR at line 83 column 7.
x=(91) 9999265789 phone=  _ERROR_=1 _N_=1
NOTE: The data set WORK.A has 1 observations
and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

```
**3rd Argument is negative then error***;

data a;
x='(91) 9999265789';
phone=substr(x,6,-2);
run;
```

```
87    data a;
88    x='(91) 9999265789';
89    phone=substr(x,6,-2);
90    run;
NOTE: Invalid third argument to function SUBSTR
at line 89 column 7.
x=(91) 9999265789 phone=9999265789 _ERROR_=1
_N_=1
NOTE: The data set WORK.A has 1 observations
and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

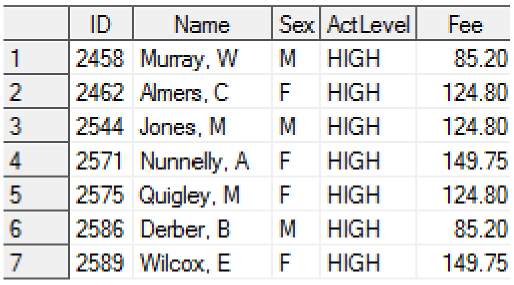
7. The =: as a substr

```
**The =: works as a substr for defined
number of bytes, it is just used in the
equality condition;


data a;
set sasuser.admit;
if actlevel =: 'HI';
run;


**You can use the where clause as well;


data a;
set sasuser.admit;
where actlevel =: 'HI';
run;
```



	ID	Name	Sex	ActLevel	Fee
1	2458	Murray, W	M	HIGH	85.20
2	2462	Almers, C	F	HIGH	124.80
3	2544	Jones, M	M	HIGH	124.80
4	2571	Nunnelly, A	F	HIGH	149.75
5	2575	Quigley, M	F	HIGH	124.80
6	2586	Derber, B	M	HIGH	85.20
7	2589	Wilcox, E	F	HIGH	149.75

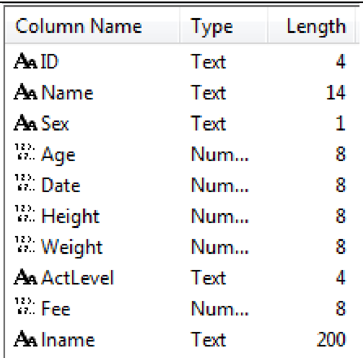
2. SCAN

The scan function searches for a particular string and puts the value in the target variable, the target variable length using the scan function is 200 chars, the delimiter is by default is blank

0. Length of new variable

```
**The length of the new variable created
becomes 200 bytes by default, here lname
would be of length 200;


data x;
set sasuser.admit;
lname=scan(name,2,',');
run;
```



Column Name	Type	Length
ID	Text	4
Name	Text	14
Sex	Text	1
Age	Num...	8
Date	Num...	8
Height	Num...	8
Weight	Num...	8
ActLevel	Text	4
Fee	Num...	8
lname	Text	200

1. SCAN function

```
* The scan function searches for a
particular string and puts the value in
the target variable, the target variable
length using the scan function is 200
chars, here school is 200 char variable
now, the delimiter by default is
blank.....IMP

it also takes 3 arguments first being
string second the nth word and third the
delimiter(by default blank), it divides
string into the chunks divided by
delimiters;


data a;
x='amit ka school tha kv';
school=scan(x,5);
run;

proc print data=a;
run;
```

```
                   The SAS System     14:52
Wednesday, January 7, 2009    5

Obs              x               school

1      amit ka school tha kv       kv
```

2. SCAN function - delimiter

```
* The scan function searches for a
particular string and puts the value in
the target variable, the target variable
```

```
                   The SAS System     14:52
Wednesday, January 7, 2009    8

Obs              x               school
```

```
length using the scan function is
200 chars, here school is 200 char
variable now, the delimiter by default is
blank, you can use the third argument as
the delimiter;


data a;
x='amit~ka~school~tha~kv';
school=scan(x,5,'~');
run;

proc print data=a;
run;
```

```
    1      amit~ka~school~tha~kv     kv
```

3.SCAN function - delimiter + blank

```
* if a string has blank and delimiter
then what scan picks;

data a;
x='amit~ ka~ school~ tha~* kv';
school=scan(x,5,'~');
name=scan(x,1,'~');
run;

proc print data=a;
run;
```

```
                                                      The SAS
System     14:52 Wednesday, January 7, 2009   9

Obs        x                   school     name

1   amit~ ka~ school~ tha~* kv      * kv      amit
```

4.imp SCAN function - delimiter+blank +dlm as first char

```
* let's see if a string has blank and
delimiter then what scan picks, even if
the string starts with delimiter, it
actually works good even if first char is
delimiter;

data a;
x='~amit~ ka~ school~ tha~* kv';
school=scan(x,1,'~');
run;


proc print data=a;
run;
```

```
                The SAS System     14:52
Wednesday, January 7, 2009   11

Obs            x                  school

1   ~amit~ ka~ school~ tha~* kv     amit
```

5.Leading blanks/ Multiple delimiters

```
*If there are leading blanks in a string
they are not part of the value, asked in
the Barclays interview(in other words
multiple occurrence of delimiters are
considered as one delimiter);


data a;
name='   Amit      Kumar    Singh   ';
lname=scan(name,3);
run;

proc print data=a;
run;

*******************************;
data a;
name='#####Amit######Kumar######Singh  ';
lname=scan(name,3,'#');
run;
```

```
The SAS System     14:52 Wednesday, January 7,
2009  12

Obs              name              lname

1      Amit      Kumar     Singh     Singh
```

```
The SAS System     14:52 Wednesday, January 7,
2009  13

Obs          name                    lname

1 ####Amit####Kumar######Singh     Singh
```

6.Scan from RIGHT

```
**This is very important thing, consider
the below example, the data is not
consistent and you wanna pick the last
name;
data x;
input @1 name $5. @7 marks;
datalines;
a b c 20
k   d 15
a     25
n a b 20
k a   30
;
run;

data master;
set x;
lname=scan(name,-1);
run;
proc print data=master;
run;

*******email id example*************;

data email;
length id $25;
input id$;
datalines;
a,b@n.com
b@tt.com
a@b@cc.com
;
run;


data a;
set email;
domain_name=scan(id,-1,'@');
run;

proc print data=a;
run;
```

```
The SAS System     14:52 Wednesday, January 7,
2009  14

Obs     name     marks    lname

1      a b c      20       c
2      k   d      15       d
3      a          25       a
4      n a b      20       b
5      k a        30       a
```

```
The SAS System     14:52 Wednesday, January 7,
2009  15

                               domain_
 Obs         id                 name

  1      a,b@n.com         n.com
  2      b@tt.com          tt.com
  3      a@b@cc.com        cc.com
```

7.Scan with multiple dlm

```
The SAS System     14:52 Wednesday, January 7,
*The scan can read diff dlms if no third
argument is provided;

data a;
str='a*b/k';
x=scan(str,1);
y=scan(str,2);
z=scan(str,-1);
run;

proc print data=a;
run;
```

```
2009   16

Obs      str      x     y     z

 1      a*b/k     a     b     k
```

8. when the nth word does not exist

```
* when the nth word does not exist in
input string, the output will be blank;

data a;
x='amit~ka~school~tha~kv';
school=scan(x,6,'~');
run;

proc print data=a;
run;
```

```
The SAS System     14:52 Wednesday, January 7,
2009   18

Obs              x               school

 1      amit~ka~school~tha~kv
```

3. Compress,compbl,strip

3.a. Compress

Compress squeezes the string and takes the delimiter blank as default and same length as parent string, removes the leading and trailing blanks +internal blanks if blank is the delimiter which is the default case;

0. compress output length

```
*The length of the compressed variable is
same as that of the parent variable;

data a;
name=' c v vfv';
name1=compress(name);
run;

proc sql;
describe table a;
quit;
```

```
325
326  proc sql;
327  describe table a;
NOTE: SQL table WORK.A was created like:

create table WORK.A( bufsize=4096 )
  (
   name char(8),
   name1 char(8)
  );

328  quit;
```

1. compress

```
*Here compress squeezes the string and
takes the delimiter blank as default and
same length as parent string, removes the
leading and trailing blanks +internal
blanks if blank is the delimiter which is
the default case;

data amit;
x='ab c   d';
y=compress(x);
run;

proc print data=amit;
run;
```

```
The SAS System     14:52 Wednesday, January 7,
2009   20

Obs       x         y

 1      ab c    d    abcd
```

```
           2. compress - delimiter
*Here compress squeezes the string and
takes the delimiter blank as default,
lets say you have delimiter as ',', here
the delimiter gets removed only, not the
internal blanks;

data amit;
x='ab, c,  d';
y=compress(x,',');
run;

proc print data=amit;
run;
```

```
The SAS System     14:52 Wednesday, January 7,
2009  21

Obs       x          y

 1     ab, c,  d    ab c  d
```

```
      3. Third argument, 1 dlm at a time
* The SAS would not recognize the third
argument in a compress function, so
compress take only 2 arguments;

data amit;
x='ab,c,~d';
y=compress(x,',', '~');
run;
```

```
The SAS System     14:52 Wednesday, January 7,
2009  27

Obs       x         y

 1     ab,c,~d    abc~d

WARNING: In a call to the COMPRESS function or
routine, the modifier "~" not valid.
NOTE: The data set WORK.AMIT has 1 observations
and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.12 seconds
      cpu time            0.04 seconds
```

```
* In case you want to delete multiple
delimiters in one go, you need to put all
delimiters in second argument in any
order;

data amit;
x='ab,c,*&%$ ~d';
y=compress(x,'~$%&* ,');
run;

proc print data=amit;
run;
```

```
The SAS System     14:52 Wednesday, January 7,
2009  28

Obs        x           y

 1     ab,c,*&%$ ~d    abcd
```

3.b. Compbl

The compbl function compress the in between blanks as the name suggest and makes the blank gap to uniform 1, The LEADING MULTIPLE BLANKS ARE REDUCED TO 1 BLANK ONLY; length of output variable is same as that of parent variable

```
* The compbl function compress the in
between blanks as the name suggest and
makes the blank gap to uniform 1, The
LEADING MULTIPLE BLANKS ARE REDUCED TO 1
BLANK ONLY;

data amit;
x='            a  b          c        d
';
y=compbl(x);
run;

proc print data=amit;
run;
```

```
The SAS System     14:52 Wednesday, January 7,
2009  38

Obs              x                 y

1     a  b        c        d    a b c d
```

```
* SAS fires error as compbl can have only
one argument;


data amit;
x='a**b*c*d*e';
y=compbl(x,'*');
run;

proc print data=amit;
run;
```

```
500  y=compbl(x,'*');
         ------
         72
ERROR 72-185: The COMPBL function call has too
many arguments.

501  run;

NOTE: The SAS System stopped processing this
step because of errors.
WARNING: The data set WORK.AMIT may be
incomplete.  When this step was stopped there
were 0
         observations and 2 variables.
WARNING: Data set WORK.AMIT was not replaced
because this step was stopped.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

3.c. Strip

The STRIP function just strips the leading and trailing blank(s) and no effect on internal blanks; length of output variable is same as that of parent variable.

```
* It just strips the leading and trailing
blank(s) and no effect on internal
blanks;

data amit;
x='  x  c d  ';
y=strip(x);
run;

proc print data=amit;
run;
```

```
The SAS System     14:52 Wednesday, January 7,
2009  40

Obs      x         y

 1     x  c d    x  c d
```

4.Index, Indexc, Indexw

4.a. Index

The INDEX function searches the position of the excerpt or required string in the source string and if excerpt is not found then it returns 0.

```
data _null_;
x='ab  cd fg hi';
y=index(x,'hi');
put y=;
run;

******************or*************;
data _null_;
x='ab  cd fg hi';
y='hi';
z=index(x,y);
put z=;
run;
```

```
1     data _null_;
2     x='ab  cd fg hi';
3     y=index(x,'hi');
4     put y=;
5     run;

y=11
NOTE: DATA statement used (Total process time):
      real time           0.33 seconds
      cpu time            0.00 second
```

```
2.Index,trim n compress
* The index function can be used
with the trim if the excerpt has
trailing blanks, as trim removes the
trailing blanks;

data _null_;
x='a cc dd';
y='dd  ';
```

```
19    x='a cc dd';
20    y='dd  ';
21    z=index(x,trim(y));
22    put z=;
23    run;

z=6
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
```

```
z=index(x,trim(y));
put z=;
run;
```

```
      cpu time            0.00 seconds
```

```
*you can use compress too;

data _null_;
x='a cc dd';
y='dd  ';
z=index(x,compress(y));
put z=;
run;
```

```
38    put z=;
39    run;

z=6
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 second
```

3.Conditional processing

```
* The index function can be used to
do conditional processing;

data x;
set sasuser.admit;
if index(actlevel,"HI") gt 0 then
do;
output;
end;
run;
```

```
42    data x;
43    set sasuser.admit;
44    if index(actlevel,"HI") gt 0 then do;
45    output;
46    end;
47    run;

NOTE: There were 21 observations read from the data
set SASUSER.ADMIT.
NOTE: The data set WORK.X has 7 observations and 9
variables.
NOTE: DATA statement used (Total process time):
      real time           0.13 seconds
      cpu time            0.01 seconds
```

4.b. Indexc

The INDEXC function searches for the occurrence of the first character of the first excerpt from left to right and if nothing is found then returns 0, good thing about indexc is that it can take multiple excerpt strings as input.

```
*it will return the first occurrence
of any string to be searched
irrespective of sequence of
arguments;


data _null_;
x='abc def';
y=indexc(x, 'de','c');

put y=;
run;
```

```
48    data _null_;
49    x='abc def';
50    y=indexc(x, 'de','c');
51
52    put y=;
53    run;

y=3
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

```
*********************;
data _null_;
x='abc def';
y=indexc(x,'bc', 'c');

put y=;
run;
```

```
y=2
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

4.c. Indexw

The indexw searches for words in the source string. The INDEXW function searches source, from left to right, for the first occurrence of excerpt and returns the position in source of the substring's first character. If the substring is not found in source, then INDEXW returns a value of 0. If there are multiple occurrences of the string, then INDEXW returns only the position of the first occurrence.
The substring pattern must begin and end on a word boundary. For INDEXW, word boundaries are delimiters, the beginning of source, and the end of source. If you use an alternate delimiter, then INDEXW does not recognize the end of the text as the end data.
INDEXW has the following behaviour when the second argument contains blank spaces or has a length of 0:

- If both source and excerpt contain only blank spaces or have a length of 0, then INDEXW returns a value of 1.
- If excerpt contains only blank spaces or has a length of 0, and source contains character or numeric data, then INDEXW returns a value of 0.

```
* The indexw searches for words in
the source string, here indexw
searches for a whole word bd and it
identifies word in source string by
the delimiter blank, so there are 4
words in source string ab cd abd and
bd, it does not pick abd while it
pics bd as bd is an individual word;

data _null_;
x='ab cd abd bd';
y=indexw(x,'bd');
put y=;
run;
```

```
data _null_;
x='ab cd abd bd';
y=indexw(x,'bd');
put y=;
run;

y=11
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

```
*it takes the delimiter as argument
also******************;

data _null_;
x='ab~cd~abd~bd';
y=indexw(x,'bd','~');
put y=;
run;
```

```
data _null_;
x='ab~cd~abd~bd';
y=indexw(x,'bd','~');
put y=;
run;

y=11
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

5. Left

This function removes the leading blanks and the blanks now moves to right, thus the string gets aligned to LEFT, the length of the new string is equal to the parent string;.

```
*here we can see by simple put it
does not print leading or trailing
blanks so we concatenated it with
'*' to show up the leading and
trailing blanks.

data x;

a='  cc bb';
b=left(a);
c='*'||a||'*';
d='*'||b||'*';

put a=;
put b=;
put c=;
put d=;

run;
```

```
data x;

a='  cc bb';
b=left(a);
c='*'||a||'*';
d='*'||b||'*';

put a=;
put b=;
put c=;
put d=;

run;

a=cc bb
b=cc bb
c=*  cc bb*
d=*cc bb  *
NOTE: The data set WORK.X has 1 observations and 4
variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

```
2.Left and trim
* This function removes the leading
blanks (L) and aligns the string to
left+ the trim function removes the
trailing blanks so a good combo;
```

```
c='*'||b||'*';
d='*'||trim(b)||'*'
put c=;
put d=;
run;

c=*cc bb   *
```

```
data x;
a='  cc bb  ';
b=left(a);
c='*'||b||'*';
d='*'||trim(b)||'*';
put c=;
put d=;
run;
```

```
d=*cc bb*
NOTE: The data set WORK.X has 1 observations and 4
variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds
```

6. Right

This function removes the trailing blanks and the blanks now moves to left, thus the string gets aligned to RIGHT, the length of the new string is equal to the parent string;.

```
*here we can see by simple put it
does not print leading or trailing
blanks so we concatenated it with
'*' to show up the leading and
trailing blanks.  ;

data x;
a='  cc bb  ';
b=Right(a);
c='*'||a||'*';
d='*'||b||'*';

put a=;
put b=;
put c=;
put d=;
run;
```

```
data x;
a='  cc bb  ';
b=Right(a);
c='*'||a||'*';
d='*'||b||'*';
put a=;
put b=;
put c=;
put d=;
run;


a=cc bb
b=cc bb
c=*  cc bb  *
d=*    cc bb*
NOTE: The data set WORK.X has 1 observations and 4
variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

7.a. Trim

This function removes the trailing blanks (you will be able to see results Only when you use it with concatenation).

```
*here we can see by simple put it
does not print leading or trailing
blanks so we concatenated it with
'*' to show up the leading and
trailing blanks.

data x;
a='cc bb  ';
b=a||'*';
c=trim(a)||'*';
put b=;
put c=;
run;
```

```
data x;
a='cc bb  ';
b=a||'*';
c=trim(a)||'*';
put b=;
put c=;
run;


b=cc bb  *
c=cc bb*
NOTE: The data set WORK.X has 1 observations and 3
variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

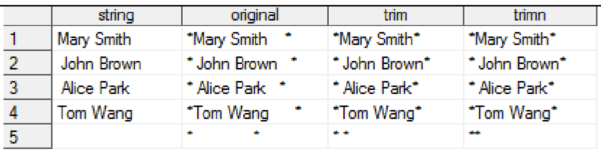
7.b. TRIMN

TRIM vs. TRIMN - Both TRIM and TRIMN remove trailing blanks from a character string. The only difference is how they deal with blank strings. If there is a blank string variable, the TRIM function returns one blank whereas the TRIMN function returns no blank characters.

```
data sample;
input string $char14.;
datalines;
Mary Smith    /* contains trailing blanks */
```

```
 John Brown    /* contains leading blanks */
 Alice Park    /* contains leading and trailing blanks */
Tom Wang       /* contains leading, trailing and multiple blanks
in between */
               /* contains a blank string */
;


data sample;
set sample;
 original = '*' || string || '*';
 trim = '*' || trim(string) || '*';
 trimn = '*' || trimn(string) || '*';
run;
```



	string	original	trim	trimn
1	Mary Smith	*Mary Smith *	*Mary Smith*	*Mary Smith*
2	John Brown	* John Brown *	* John Brown*	* John Brown*
3	Alice Park	* Alice Park *	* Alice Park*	* Alice Park*
4	Tom Wang	*Tom Wang *	*Tom Wang*	*Tom Wang*
5		* *	* *	**

8. TRANWARD

The tranwrd function helps in the replacement of a string in a char variable.

```
* suppose you want to change in
sasuser.admit the values of the
actlevel ;

data amit;
length actlevel1 $10;
set sasuser.admit;
actlevel1=tranwrd(lowcase(actlevel),'high',
'higher');
run;

proc print data=amit(keep = actlevel1);
run;

*You can replace the existing variable as
well************;

data amit;
set sasuser.admit;
actlevel=tranwrd(actlevel,'HI', 'ho');
run;
```

```
Obs    actlevel1

 1     higher
 2     higher
 3     low
 4     mod
 5     low
 6     higher
 7     mod
 8     mod
 9     low
10     low
11     higher
12     low
13     mod
14     higher
```

```
2.Multiple occurrence
* Converting the multiple occurence of a
string;

data x;
input name$;
datalines;
highhigh
high
cc
;
run;

data a;
set x;
name=tranwrd(name,'high','hi');
run;

proc print data=a;
run;
```

```
The SAS System         00:29 Wednesday, July
1, 2009   3

Obs    name
 1     hihi
 2     hi
 3     cc
```

```
*The tranwrd function replaces a particular
text of the variable in the dataset by a
new text,
lets say you want to change the sex to nsex
with M to Male;
```

```
data a;
set sasuser.admit;
nsex=tranwrd(sex,'M','Male');
run;
```

9. TRANSLATE

The translate function changes the string character wise the character in to and from should match, any a in string would be changed to 1 and so on.

```
data x;
name='amit kumar';
newname=translate(name,'123','ami');
*translate(string,to,from);
run;

proc print data=x;
run;
```

```
The SAS System         00:29 Wednesday, July 1, 2009
6

Obs       name         newname

 1      amit kumar    123t ku21r
```

```
* The translate function changes the
string character wise the character in
to and from should match, any a in
string would be changed here a would
be changed to 1 , m to 2 and i to
blank;


data x;
name='amit kumar';
newname=translate(name,'12','ami');
/*translate(string,to,from)*/
run;

proc print data=x;
run;
```

```
The SAS System         00:29 Wednesday, July 1, 2009
7

Obs       name         newname

 1      amit kumar    12 t ku21r
```

```
**ALSO if from is unbalanced then no
issues***********;

data x;
name='amit kumar';
newname=translate(name,'123','am');
/*translate(string,to,from)*/
run;
proc print data=x;
run;
```

```
The SAS System         00:29 Wednesday, July 1, 2009
8

Obs       name         newname

 1      amit kumar    12it ku21r
```

```
***********Multiple Inputs**********
* The inputs are fed in groups of to and
from,here a is changed to 1 and m to 2
and k to 3 and u to 4;

data x;
name='amit kumar';
name=translate(name,'12','am','34','ku');
run;
proc print data=x;
run;
```

```
The SAS System         00:29 Wednesday, July 1,
2009   9

Obs       name

 1      12it 3421r
```

```
***********Last one prevails*********
* If the from characters overlap then the
last one prevails here first a points to
1 and second a to 9, so the second a
prevails;
```

```
The SAS System         00:29 Wednesday, July 1,
2009   10

Obs       name
```

```
data x;
name='amit kumar';
name=translate(name,'12','am','94','au');
run;
proc print data=x;
run;
```

```
1      92it k429r
```

10. LOWCASE

It converts the string in to small letters or in lower case.

```
data amit;
x='AMIT';
y=lowcase(x);
x=lowcase('NANA');
run;

proc print data=amit;
run;
```

```
The SAS System         00:29 Wednesday, July 1, 2009

Obs     x       y

 1      nana     amit
```

11. UPCASE

It converts the string in to capital letters or in upper case.

```
data amit;
x='amit';
y=upcase(x);
x=upcase('nana');
run;

proc print data=amit;
run;
```

```
The SAS System         00:29 Wednesday, July 1, 2009

Obs     x       y

 1      NANA     AMIT
```

12. PROPCASE

It converts the string in to proper case. First letter of each word in upper case and all other latters in lower case.

```
data amit;
x='AMIT';
y=propcase(x);
z=propcase('nana is a good boy');
run;

proc print data=amit;
run;
```

```
The SAS System         00:29 Wednesday, July 1, 2009

Obs     x       y                z

 1      AMIT     Amit     Nana Is A Good Boy
```

13. MEAN

The mean sas function can be used to calculate the mean of the values.

```
data amit;
x=10;
y=20;
mean_age=mean(x,y);
run;

proc print data=amit;
run;
```

```
The SAS System         00:29 Wednesday, July 1,
2009

Obs     x     y     mean_age

 1     10    20        15
```

```
*VARIABLE LIST***********;
*The same output could be used if we pass the
arguments as variable list;

data amit1;
x1=10;
x2=20;
mean_age=mean(of x1-x2); /* If you omit the OF
then it would calculate  x1-x2 and the mean of
the diff */
run;

proc print data=amit1;
run;
```

```
The SAS System         00:29 Wednesday, July 1,
2009

Obs    x1    x2     mean_age

 1     10    20        15
```

14. QUARTER

The qtr function calculates the Qauarter of the date and returns the value ranging from 1 to 4.

```
data amit;
set sasuser.empdata;
qtrs=qtr(hiredate);
run;

proc print data=amit;
var hiredate qtrs;
run;
```

```
The SAS System         00:29 Wednesday, July 1, 2009
Obs     HireDate    qtrs

 1     11MAR1992      1
 2     19DEC1983      4
 3     12MAR1985      1
 4     16OCT1989      4
 5     19DEC1981      4
 6     27APR1991      2
```

```
*Doing some subsetting*********;

data amit2;
set sasuser.empdata;
if qtr(hiredate) gt 2;
run;

proc print data=amit2;
run;
```

```
*passing date value;
data a;
date='16feb2012'd;
qtr=qtr(date);
run;

proc print data=a;
run;
```

```
The SAS System         00:29 Wednesday, July 1, 2009

Obs     date    qtr

 1      19039     1
```

15. SUM, SUM OF

The SUM functions gives the sum of 2 or more variables. It also provides the sum of range if variables.

```
x1=sum(4,9,3,8);                    24


x2=sum(4,9,3,8,.);                  24     Missing value still sum gets calculated


x1=9;
x2=39;
x3=sum(of x1-x2);                   48                  Sum for Range


x1=5; x2=6; x3=4; x4=9;
y1=34; y2=12; y3=74; y4=39;
result=sum(of x1-x4, of y1-y5);     183                 Range of two variables


x1=55;
x2=35;
x3=6;
x4=sum(of x1-x3, 5);                101                Range and a constant value


x1=7;
x2=7;
x5=sum(x1-x2);
                                      0               As diff gets calculated


y1=20;
y2=30;
x6=sum(of y:);                        50             Sum of all values of variable y
```

```
*The sum of function  can be used to calculate table with the sum of range of
variables;

data a;
sale1=5;
sale2=10;
sale3=15;
sale_sum=sum(of sale1-sale3);
run;
```

```
*The second way could be if you want to include all sale variables, use the colon wild
card;

data a;
sale1=5;
sale2=10;
sale3=15;
sale_sum=sum(of sale:);
run;
```

```
*The sum of function  can be used to calculate table with the sum of range of
variables, here just the sale1-sale4 sum is calculated and variable sale5 is created
with null value;

data a;
sale1=5;
sale2=10;
sale3=15;
sale4=5;
sale_sum=sum(of sale1-sale5);
```

```
run;
```

16. DAY

The day function calculates the day of the date and returns the value ranging from 1 to 30/31.

```
data amit;
set sasuser.empdata;
days=day(hiredate);
run;

proc print data=amit;
var hiredate days;
run;
```

```
The SAS System        00:29 Wednesday, July 1, 2009

Obs      HireDate     days

 50     17JAN1994      17
```

```
**Doing some subsetting*********;

data amit2;
set sasuser.empdata;
if day(hiredate) gt 10;
run;

proc print data=amit2;
run;
```

17. YEAR

The year function calculates the year from a date value.

```
data amit;
set sasuser.empdata;
yr=year(hiredate);
run;

proc print data=amit;
var hiredate yr;
run;
```

```
The SAS System        00:29 Wednesday, July 1, 2009

Obs      HireDate     yr

 50     17JAN1994     1994
```

```
**Doing some subsetting***;

data amit2;
set sasuser.empdata;
if year(hiredate)=1992;
run;


proc print data=amit2;
run;
```

18. WEEKDAY

The weekday function calculates the day of the date and returns the value ranging from 1 to 7, 1 being the Sunday and 2 Monday and so on.

```
data amit;
set sasuser.empdata;
wkday=weekday(hiredate);
run;

proc print data=amit;
```

```
The SAS System        00:29 Wednesday, July 1, 2009

Obs      HireDate     wkday

 50     17JAN1994     2
```

```
var hiredate wkday;
run;
```

```
**Doing some subsetting***;

data amit2;
set sasuer.empdata;
if weekday(hiredate) gt 5;
run;

proc print data=amit2;
run;
```

19. Month

The month function calculates the month of the date and returns the value ranging from 1 to 12.

```
data amit;
set sasuser.empdata;
mnths=month(hiredate);
run;

proc print data=amit;
var hiredate mnths;
run;
```

```
The SAS System        00:29 Wednesday, July 1, 2009

Obs     HireDate     mnths

 50     17JAN1994     1
```

```
**Doing some subsetting***;

data amit2;
set sasuer.empdata;
if month(hiredate) gt 10;
run;

proc print data=amit2;
run;
```

20. MDY

The mdy function creates a numeric date from the values of the month day and year.

```
data amit;
input name $ month year day;
datalines;
amit 10 1981 13
pre  04 1982 20
;
run;

data amit1;
set amit;
attrib bdy format=date9.;
bdy=mdy(month,day,year);
run;

proc print data=amit1;
run;
```

```
The SAS System        00:29 Wednesday, July 1, 2009

Obs    name    month    year    day         bdy

 1     amit     10      1981     13    13OCT1981
 2     pre       4      1982     20    20APR1982
```

```
**Practical use********;
data a;
set sasuser.empdata;
format hiredate1 date9.;
m=month(hiredate);
D=day(hiredate);
y=year(hiredate);
hiredate1=mdy(m,d,y);
if hiredate ne hiredate1 then flag=1;
else flag=0;
drop m d y;
run;

proc sql;
select count(*) as cnt_flag_1 from a where
flag=1;
quit;
```

```
The SAS System        00:29 Wednesday, July 1, 2009

        cnt_flag_1
        ----------
                 0
```

21. Date and Today

The date() or today() function can be used interchangeably and they do not need any arguments.

```
data amit2;
set sasuser.empdata;
format current date9.;
format currdate date7.;
current=today();
currdate=date();

run;

proc print data=amit2 (keep=current
currdate);
run;
```

```
The SAS System        00:29 Wednesday, July 1, 2009

Obs      current     currdate

 50     29JUN2017     29JUN17
```

22. PUT

The PUT function is used to convert the numeric values to the character values for the SAS.

```
**If we do not use put function THE
SAS LOG INDICATES THE NUMERIC TO
CHARACTER CONVERSION OF THE AGE******;

data amit;
input name $ age monthsalary $ ;
datalines;

amit 34 123,45.00
na 23 213,45.00
;
run;

data amit1;
set amit;
nameage= name ||'/'|| age;
run;

proc print data=amit1;
run;
```

```
The SAS System        00:29 Wednesday, July 1, 2009   43

Obs    name    age    monthsalary          nameage

1      amit    34     123,45.0      amit    /          34
2      na      23     213,45.0      na      /          23

______________Log___________________
435
436  data amit1;
437  set amit;
438  nameage= name ||'/'|| age;
439  run;
NOTE: Numeric values have been converted to character
values at the places given by:
      (Line):(Column).
      438:23
NOTE: There were 2 observations read from the data set
WORK.AMIT.
NOTE: The data set WORK.AMIT1 has 2 observations and 4
```

```
variables.
NOTE: DATA statement used (Total process time):
      real time           0.02 seconds
      cpu time            0.00 seconds
```

```
**USING THE PUT FUNCTION***;

data amit;
input name $ age monthsalary $ ;
datalines;

amit 34 123,45.00
na 23 213,45.00
;
run;

data amit1;
set amit;
nameage= name ||'/'|| put(age,2.);
run;


proc print data=amit1;
run;
```

```
The SAS System        00:29 Wednesday, July 1, 2009   44

Obs     name     age     monthsalary       nameage


 1      amit      34      123,45.0       amit    /34
 2      na        23      213,45.0       na      /23

______________Log___________________
453
454  data amit1;
455  set amit;
456  nameage= name ||'/'|| put(age,2.);
457  run;

NOTE: There were 2 observations read from the data set
WORK.AMIT.
NOTE: The data set WORK.AMIT1 has 2 observations and 4
variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds
```

23. INPUT

The Input function is used to convert the character values to the numeric values for the SAS to perform the calculations.

```
* Here salary is been converted from the
character values to the numeric values;

data amit;
input name $ age monthsalary $ ;
datalines;

amit 34 123,45.00
na 23 213,45.00
;
run;

data amit1;
set amit;
salary=input(monthsalary, comma9.2);
yearsalry=salary*12;
run;
```

```
The SAS System          00:44 Wednesday, July 1, 2009

Obs    name    age    monthsalary    salary    yearsalry

1      amit    34     123,45.0        12345       148140
2      na      23     213,45.0        21345       256140
```

```
proc print data=amit1;
run;
```

24. CATX

The catx function helps in the concatenating of the character strings and no need of left trim.

The CATX function first copies item-1 to the result, omitting leading and trailing blanks. Then for each subsequent argument item-i, i=2, ..., n, if item-i contains at least one non-blank character, then 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. Blank items do not produce delimiters at the beginning or end of the result, nor do blank items produce multiple consecutive delimiters.

Length of Returned Variable
In a DATA step, if the CATX function returns a value to a variable that has not previously been assigned a length, then that variable is given a length of 200 bytes. If the concatenation operator (||) returns a value to a variable that has not previously been assigned a length, then that variable is given a length that is the sum of the lengths of the values which are being concatenated..

The catx function creates string of 200 bytes in length;

Function	Equivalent Code
CATX(SP, OF X1-X4)	TRIM(LEFT(X1))||SP||TRIM(LEFT(X2))||SP||TRIM(LEFT(X3))||SP||TRIM(LEFT(X4))

```
data a;
set sasuser.admit;
x=catx(',',actlevel,sex);
run;


proc sql;
describe table a;
quit;
```

```
NOTE: There were 21 observations read from the data
set SASUSER.ADMIT.
NOTE: The data set WORK.A has 21 observations and 10
variables.
NOTE: DATA statement used (Total process time):
      real time           0.40 seconds
      cpu time            0.03 seconds



25
26
27   proc sql;
28   describe table a;
NOTE: SQL table WORK.A was created like:

create table WORK.A( bufsize=16384 )
  (
   ID char(4),
   Name char(14),
   Sex char(1),
   Age num,
   Date num,
   Height num,
   Weight num,
   ActLevel char(4),
   Fee num format=7.2,
   x char(200)
  );

29   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.14 seconds
      cpu time            0.03 seconds
```

```
data amit;
input name $ month year day;
datalines;
amit 10 1981 13
pre  04 1982 20
;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs     name     month     year     day
 1      amit       10      1981      13
 2      pre         4      1982      20
```

```
run;

proc print data=amit;
run;

data amit1;
set amit;
namejoin=catx('*',name,day,year,month);
run;

proc print data=amit1;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs    name    month    year    day      namejoin

1      amit     10      1981     13    amit*13*1981*10
2      pre       4      1982     20    pre*20*1982*4
```

```
*CATX function with a series
Lets say you want to concatenate the value
of a variable;
data a;
sp='|';
x1='a';
x2='b';
x3='c';
string=catx(sp,of x1-x3);
run;
proc print data=a;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs    sp    x1    x2    x3    string

 1     |     a     b     c     a|b|c
```

```
*CATX function with a series of vars with
colon
*The catx just concatenates the value of
variables with the colon operator on
variable x;
data a;
x1=' a';
x2=' b';
x3='c';
string=catX(',',of x:);
run;
proc print data=a;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs    x1    x2    x3    string

 1     a     b     c     a,b,c
```

25. CAT

The cat just concatenates the value of variables.

The CAT function 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.

Length of Returned Variable
In a DATA step, if the CAT function returns a value to a variable that has not previously been assigned a length, then that variable is given a length of 200 bytes. If the concatenation operator (||) returns a value to a variable that has not previously been assigned a length, then that variable is given a length that is the sum of the lengths of the values which are being concatenated.

The cat function creates string of 200 bytes in length;

Function	Equivalent Code
CAT(OF X1-X4)	X1||X2||X3||X4

```
data a;
set sasuser.admit;
x=cat(name,sex);
run;

proc print data=a(keep = x);
```

```
The SAS System        00:44 Wednesday, July 1, 2009
Obs          x
1       Murray, W     M
2       Almers, C     F
3       Bonaventure, TF
4       Johnson, R    F
```

```
run;
```

```
5     LaMance, K     M
6     Jones, M       M
7     Reberson, P    F
8     King, E        M
9     Pitts, D       M
```

```
*CAT function with a series
*The cat just concatenates the value of
variables, it just concatenates and does
not remove space, leading or trailing;

data a;
x1=' a';
x2=' b';
x3='c';
string=cat(of x1-x3);
run;
proc print data=a;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs    x1    x2    x3    string

 1     a     b     c      a bc
```

```
*CAT function with a series of vars with
colon
*The cat just concatenates the value of
variables, it just concatenates and does
not remove space, leading or trailing;

data a;
x1=' a';
x2=' b';
x3='c';
string=cat(of x:);
run;
proc print data=a;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs    x1    x2    x3    string

 1     a     b     c      a bc
```

26. CATS

The cats just concatenates the value of variables. The cats is just equal to Strip or trim(left(var));.

The CATS function 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, SAS does not write a note to the log.

Length of Returned Variable
In a DATA step, if the CATS function returns a value to a variable that has not previously been assigned a length, then that variable is given a length of 200 bytes. If the concatenation operator (||) returns a value to a variable that has not previously been assigned a length, then that variable is given a length that is the sum of the lengths of the values which are being concatenated.

The cats function creates string of 200 bytes in length;

Function	Equivalent Code
CATS(OF X1-X4)	TRIM(LEFT(X1))||TRIM(LEFT(X2))||TRIM(LEFT(X3))||TRIM(LEFT(X4))

```
data a;
set sasuser.admit;
x=cats(name,sex);
run;

proc print data=a (keep = x);
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009
Obs          x
1     Murray, WM
2     Almers, CF
3     Bonaventure, TF
4     Johnson, RF
5     LaMance, KM
6     Jones, MM
```

```
7     Reberson, PF
8     King, EM
9     Pitts, DM
```

```
*CATS function with a series
*The cats just concatenates the value of
variables, it concatenates and removes
space, leading or trailing;

data a;
x1=' a ';
x2=' b';
x3=' c ';
string=cats(of x1-x3);
run;
proc print data=a;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs     x1     x2     x3     string

 1      a      b      c       abc
```

```
*CATS function with a series of vars with
colon

data a;
x1=' a';
x2=' b ';
x3=' c ';
string=cats(of x:);
run;
proc print data=a;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs     x1     x2     x3     string

 1      a      b      c       abc
```

27. CATT

CatT is equal to TRIM. The catt just concatenates the value of variables, after applying the TRIM on them

The CATT function 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.

Length of Returned Variable
In a DATA step, if the CATT function returns a value to a variable that has not previously been assigned a length, then that variable is given a length of 200 bytes. If the concatenation operator (||) returns a value to a variable that has not previously been assigned a length, then that variable is given a length that is the sum of the lengths of the values which are being concatenated.

The cats function creates string of 200 bytes in length;

Function	Equivalent Code
CATT(OF X1-X4)	TRIM(X1)||TRIM(X2)||TRIM(X3)||TRIM(X4)

```
data a;
set sasuser.admit;
x=catt(name,sex);
run;

proc print data=a(keep = x);
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009
Obs          x
1     Murray, WM
2     Almers, CF
3     Bonaventure, TF
4     Johnson, RF
5     LaMance, KM
6     Jones, MM
7     Reberson, PF
```

```
8    King, EM
9    Pitts, DM
```

```
*CATT function with a series
data a;
x1=' a';
x2=' b';
x3='c';
string=catt(of x1-x3);
run;

proc print data=a;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs    x1    x2    x3    string

 1     a     b     c      a bc
```

```
*CATT function with a series of vars with
colon

data a;
x1=' a';
x2=' b';
x3='c';
string=catt(of x:);
run;

proc print data=a;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs    x1    x2    x3    string

 1     a     b     c      a bc
```

28. FIND

* The find and index have the following differences;

1. The FIND function searches for substrings of characters in a character string, whereas the FINDC function searches for individual characters in a character string.

2. The FIND function and the INDEX function both search for substrings of characters in a character string. However, the INDEX function does not have the modifiers nor the start pos arguments.

```
*lets test the find function and the value
returned, here score has a value 4 name is
the 4 char in string;


data a;
x='my name is amit';
score=find(x,'name');
put score=;
run;
```

```
235  data a;
236  x='my name is amit';
237  score=find(x,'name');
238  put score=;
239  run;

score=4
NOTE: The data set WORK.A has 1 observations and 2
variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

```
*FIND function with a modifier
***The i modifier ignores the case of
substring;

data a;
x='my NAME is amit';
```

```
242  data a;
243  x='my NAME is amit';
244  score=find(x,'name','i');
245  put score=;
246  run;
```

```
score=find(x,'name','i');
put score=;
run;


***The t modifier strips the leading and
the trailing spaces in string and
substring;

data a;
x='my name is amit';
score=find(x,'name','t');
put score=;
run;
```

```
score=4
NOTE: The data set WORK.A has 1 observations and 2
variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


251  data a;
252  x='my name is amit';
253  score=find(x,'name','t');
254  put score=;
255  run;

score=4
NOTE: The data set WORK.A has 1 observation and 2
variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 second
```

```
* The Find function helps in the searching
of a string in the character variable, it
returns the position of the string if
found else 0;

data amit;
set sasuser.admit;
if find(name,'be','t') gt 0; /* thet
modifier here trims the trailing blanks of
the name var*/
run;

proc print data=amit;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs        Name         Sex    Age    Date     Height

1     Reberson, P       F      32      9        67
2     Eberhardt, S      F      49     27        64
3     Oberon, M         F      28     17        62
4     Derber, B         M      25     23        75
```

```
*FIND RETURNS THE STRING
POSITION************:
* we can find the string position 'be',
lets assign value of be to a var x;

data amit;
set sasuser.admit;
x= find(name,'be','t') ; /* thet modifier
here trims the trailing blanks of the name
var*/
run;

proc print data=amit(KEEP = NAME X);
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs    Name               x
 1     Murray, W         0
 2     Almers, C         0
 3     Bonaventure, T    0
 4     Johnson, R        0
 5     LaMance, K        0
 6     Jones, M          0
 7     Reberson, P       3
 8     King, E           0
 9     Pitts, D          0
10     Eberhardt, S      2
11     Nunnelly, A       0
12     Oberon, M         2
```

29. COUNT

The count function is used to count the ocuurence of a substring in a string

```
data a;
x='My nam is amit kumar, my expertise is
sas';

count_is=count(x,'is');
```

```
357  data a;
358  x='My nam is amit kumar, my expertise is sas';
360  count_is=count(x,'is');
362  put count_is=;
363  run;
```

```
put count_is=;
run;
```

```
count_is=3
NOTE: The data set WORK.A has 1 observations and 2
variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

```
*Modifiers in count
***The count again take two modifiers
similar to find, the i and t, i for
ignoring the case and t for trimming the
balnk space;
```

30. COUNTW

This function counts the number of words in a string

```
data a;
x='my name is amit';
y=countw(x);
put y=;
run;
```

```
data a;
x='my name is amit';
y=countw(x);
put y=;
run;

y=4
NOTE: The data set WORK.A has 1 observations and 2
variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

31. INT

The INT function returns the integer value of a numeric variable, thus discarding the decimal portion

```
data amit;
x=123.89;
y=int(x);
run;

proc print data=amit;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs        x        y

 1      123.89     123
```

32. ROUND

The ROUND function returns rounded value of a decimal number and by default it is 1, any value ge .50 would be in the next integer

```
data amit;
x1=123.89;
x2=123.28;
x3=123.46;
x4=123.00;
x5=123.99;


y1=round(x1,2);
y2=round(x2);
y3=round(x3);
y4=round(x4);
y5=round(x5);
run;

proc print data=amit;
run;
```

```
The SAS System         00:44 Wednesday, July 1, 2009

Obs      x1        x2        x3       x4      x5
y1     y2     y3     y4     y5

 1      123.89    123.28    123.46    123    123.99
124     123    123    123    124
```

33. N

The N function counts the number of non-missing values in a row

```
data a;
x1=2;
x2=3;
x3=4;

nvars_nonmiss=n(of x1-x3);
put nvars_nonmiss=;
run;

proc print data = a;
run;
```

```
The SAS System         00:44 Wednesday, July 1, 2009

                          nvars_
Obs     x1    x2    x3    nonmiss

 1       2     3     4       3
```

```
**or using the colon as the wild
character;

data a;
x1=2;
x2=3;
x3=4;

nvars_nonmiss=n(of x:);
put nvars_nonmiss=;
run;

proc print data = a;
run;
```

```
The SAS System         00:44 Wednesday, July 1, 2009

                          nvars_
Obs     x1    x2    x3    nonmiss

 1       2     3     4       3
```

34. NMISS

The NMISS function counts the number of missing values in a row.

```
data a;
x1=2;
x2=3;
x3=.;
nvars_miss=nmiss(of x1-x3);
put nvars_miss=;
run;

proc print data = a;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

                                nvars_
Obs     x1     x2     x3        miss

 1       2      3      .          1
```

```
**or using the colon as the wild
character;

data a;
x1=2;
x2=3;
x3=.;
nvars_miss=nmiss(of x:);
put nvars_miss=;
run;

proc print data = a;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

                                nvars_
Obs     x1     x2     x3        miss

 1       2      3      .          1
```

35. INTCK

The INTCK function returns the value of the complete interval passed between two dates, it can take diff arguments like week, month, year.

```
data amit;
x='31dec2010';
y='01jan2011';
x1=input(x,date9.);
y1=input(y,date9.);
z1=intck('week',x1,y1);
z2=intck('month',x1,y1);
z3=intck('year',x1,y1);
run;

proc print data=amit;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs      x          y          x1       y1     z1     z2    z3

1   31dec2010   01jan2011   18627    18628    0      1     1
```

```
***or make the date as a value using the
d****;

data amit;
x='31dec2010'd;
y='01jan2011'd;
z1=intck('week',x,y);
z2=intck('month',x,y);
z3=intck('year',x,y);
run;

proc print data=amit;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs       x          y        z1     z2     z3

 1      18627      18628       0      1      1
```

```
**AGE FINDING*****;

data amit;
x='13oct1981';
y='01jul2010';
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs         x            y           x1        y1      z3

 1      13oct1981    01jul2010     7956     18444     29
```

```
x1=input(x,date9.);
y1=input(y,date9.);

z3=intck('year',x1,y1);
run;

proc print data=amit;
run;
```

```
*Lets try to find the age by using the
system date;


data a;
dob='13oct1981'd;
age_years=intck('year',dob,today());
put age_years=;
run;
```

```
data a;
dob='13oct1981'd;
age_years=intck('year',dob,today());
put age_years=;
run;

age_years=36
NOTE: The data set WORK.A has 1 observations and 2
variables.
NOTE: DATA statement used (Total process time):
      real time           0.05 seconds
      cpu time            0.00 seconds
```

36. INTNX

The INTNX function increments the day, year or month on the specified date, here it increments the date by 6 weeks, means date after 6 weeks including current week.

```
data _null_;
x=intnx('week', '23mar2012'd, 2);
put x date9.;
run;
```

```
data _null_;
x=intnx('week', '23mar2012'd, 2);
put x date9.;
run;

01APR2012
NOTE: DATA statement used (Total process time):
      real time           0.05 seconds
      cpu time            0.00 seconds
```

```
* The intnx function increments the day,
year or month on the specified date, you
can use put as well;

data amit;
x='13oct1981'd;
z=put(intnx('month',x,3,'e'),date9.);
run;

proc print data=amit;
run;
```

```
The SAS System       00:44 Wednesday, July 1, 2009

Obs      x         z

  1     7956    31JAN1982
```

```
* The intnx function increments the day,
year or month on the specified date, you
can use the same argument for incrementing
1 month from the date;

data amit;
x='13oct1981'd;
z=put(intnx('month',x,1,'same'),date9.);
put z=;
run;
```

```
data amit;
x='13oct1981'd;
z=put(intnx('month',x,1,'same'),date9.);
put z=;
run;

z=13NOV1981
NOTE: The data set WORK.AMIT has 1 observations and 2
variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

```
* The intnx function increments the day,
year or month on the specified date, you
can use the b or beginning argument then
the output date would be 1 month and first
day.;


data amit;
x='13oct1981'd;
z=put(intnx('month',x,1,'b'),date9.);
put z=;
run;
```

```
data amit;
x='13oct1981'd;
z=put(intnx('month',x,1,'b'),date9.);
put z=;
run;

z=01NOV1981
NOTE: The data set WORK.AMIT has 1 observations and 2
variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

```
* The intnx function increments the day,
year or month on the specified date, you
can use the e or end argument then the
output date would be 1 month and first
day.;

data amit;
x='13oct1981'd;
z=put(intnx('month',x,1,'e'),date9.);
put z=;
run;
```

```
data amit;
x='13oct1981'd;
z=put(intnx('month',x,1,'e'),date9.);
put z=;
run;

z=30NOV1981
NOTE: The data set WORK.AMIT has 1 observations and 2
variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

37. Datdif n yrdif

The datdif and yrdif function helps in calculating the days and year diff between the two dates.

'ACT/ACT'
uses the actual number of days between dates in calculating the number of years. SAS calculates this value as the number of days that fall in 365-day years divided by 365 plus the number of days that fall in 366-day years divided by 366.

```
data amit;
x='13oct1981';
y='03apr2012';
x1=input(x,date9.);
y1=input(y,date9.);
z3=yrdif(x1,y1,'actual');
z4=datdif(x1,y1,'actual');
run;

proc print data=amit;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs         x              y          x1       y1
z3         z4

1      13oct1981     03apr2012     7956     19086
30.4733     11130
```

```
data amit;
x='13oct1981'd;
y='03apr2012'd;
z3=yrdif(x,y,'actual');
z4=datdif(x,y,'actual');
run;

proc print data=amit;
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs      x         y          z3         z4

 1      7956     19086     30.4733     11130
```

```
*The today() function can be used as well
for finding the age;

data amit;
dob='13oct1981'd;
age=yrdif(dob,today(),'actual');
run;
```

```
The SAS System        00:44 Wednesday, July 1, 2009

Obs     dob      age

 1      7956     35.7123
```

```
proc print data=amit;
run;
```

```
data amit;
dob='13oct1981'd;
age=datdif(dob,today(),'actual');
run;

proc print data=amit;
run;
```

```
The SAS System       00:44 Wednesday, July 1, 2009

Obs     dob      age

 1     7956     13044
```

38. REVERSE

The reverse function just reverse the string, if there are leading blanks they become trailing.

```
data a;
x='abc';
change=reverse(x);
put change;
run;
```

```
data a;
x='abc';
change=reverse(x);
put change;
run;

cba
NOTE: The data set WORK.A has 1 observations
and 2 variables.
NOTE: DATA statement used (Total process
time):
      real time           0.35 seconds
      cpu time            0.01 seconds
```

```
* Let's say you have drug data, how would you
take out the dose strength in a new variable;

data a;
input trt$;
datalines;
a150
b120
130
cd300
;
run;


data b;
set a;
dose=reverse(substr(reverse(compress(trt)),1,3));
run;

proc print data = b;
run; 37 Functions
```

```
The SAS System       00:44 Wednesday, July
1, 2009

Obs     trt     dose

 1      a150     150
 2      b120     120
 3      130      130
 4      cd300    300
```