FUNCTIONS

-> Character Functions
-> Numeric Functions
-> Date Functions

CHARACTER FUNCTIONS

---------- Functions that change the case of characters----------

```
Data ds;
Infile datalines;
Input Name&$15. Sex$ Age Height Weight;
Datalines;
Alfred Carter  M 14 69 112.5
Alice Davis  F 13 56.5 84
Barbara Smith  F 13 65.3 98
Carol  F 14 62.8 102.5
Henry  M 14 63.5 102.5
James Kennedy  M 12 57.3 83
Jane  F 12 59.8 84.5
Janet Smit  F 15 62.5 112.5
Jeffrey Truman  M 13 62.5 84
John Killer  M 12 59 99.5
Joyce Free  F 11 51.3 50.5
;
Run;
```


UPCASE

Converts all letters in an argument to upper case

Syntax:-Upcase(string)

Example:-

```
Data ds1;
Set ds;
Name1=Upcase(Name);
Run;
```

/*converting letters into upper case and keeping them same column for same order like base dataset*/

```
Data ds2;
Set ds;
Name=Upcase(Name);
Run;
```

/*creating different column with uppercase letters*/

```
Data ds3(drop=Name);
Set ds;
Name1=Upcase(Name);
Run;
```

LOWCASE

Converts all letters in an argument to lowercase

Syntax:-lowcase(string)

Example:-

```
Data ds4;
Set ds2;
Name1=lowcase(Name);
Gender=lowcase(sex);
Run;
```

PROPCASE

Converts all words in an argument to proper case (like I Am Krishna)

Propcase means In a word first letter is capital rest of all small

Syntax:-Propcase(string)

Example:-

```
Data ds5;
Set ds2;
Name1=propcase(Name);
Run;
```

```
Data allcase;
a=lowcase('THIS IS A DOG');
b=propcase(a);
c=propcase(lowcase('THIS IS A DOG'));
d=Upcase('this is a dog');
Put a=;
Put b=;
Put c=;
Put d=;
Run;
```


----------Functions that extract part of strings----------

```
Data ds;
Infile datalines;
Input idno name&$18.  Team $ strtwght endwght ;
Datalines;
1331 Jason Schock Long  blue 187 172
1067 Kanoko Nagasaka  green 135 122
1251 Richard Rose  blue 181 166
1192 Charlene Armstrong  yellow 152 139
1352 Bette Long Schock  green 156 137
1262 Yao Chen Garg  blue 196 180
1124 Adrienne Fink  green 156 142
;
Run;
```

SCAN

Selects a given word from a character expression

Selects particular word from character string

Syntax:- SCAN(string ,n<, delimiter(s)>)

Examples:-

```
Data scn1;
Set ds;
New_Name=scan(Name,2);
Run;

Data scn2;
Set ds;
Name1=scan(Name,1);
Name2=scan(Name,2);
Name3=scan(Name,3);
Run;

Data scn3;
Set ds;
New_Name=scan(Name,-3);
Run;

Data scn4;
Set ds;
New_Name=scan(Name,-1);
Run;

Data scn5;
A='Madan,Mohan,Moorthy';
B=Scan(A,2,',');
Run;
```

Scan function returns Length of new variable is 200

SUBSTR

Takes substrings of matrix elements
Selects particular part from character string

Syntax:- SUBSTR(matrix, position<, length>)

Examples:-

```
Data sbstr1;
Set ds;
New_Name=substr(Name,1,5);
Run;

Data sbstr2;
a='krishnamohan';
b=substr(a,8,5);
Run;
Data sbstr3;
a='krishnamohan';
Substr(a,1,7)='madhan';
Run;

Data sbstr4;
a=put(today(),date9.);
b=substr(a,3,3);
Run;
```

Substr function returns Length of new variable is equal to string

--------Functions that combines two or more strings together strings---------

CAT

Concatenates character strings without removing leading or trailing blanks

Syntax:- CAT(string-1 <, ... string-n>)

```
Data cat1;
a='      The   Olym';
b='pic Arts Festi';
c='      val includes works by D        ';
d='ale Chihuly.';
Result=cat(a,b,c,d);
Put result $char.;
Run;
```

```
Cat functions returns
Length of new variable is 200
```

CATT

Concatenates character strings and removes trailing blanks

Syntax:-CATT(string-1 <, ...string-n>)

```
Data cat2;
a='      The   Olym';
b='pic Arts Festi';
c='      val includes works by D       ';
d='ale Chihuly.';
Result=catt(a,b,c,d);
Put result $char.;
Run;
```

CATS

Concatenates character strings and removes leading and trailing blanks

Syntax:-CATS(string-1 <, ...string-n>)

```
Data cat3;
a='       The   Olym';
b='pic Arts Festi';
c='       val includes works by D      ';
d='ale Chihuly.';
Result=cats(a,b,c,d);
Put result $char.;
Run;
```

CATX

Concatenates character strings, removes leading and trailing blanks, and inserts separators

Syntax:-CATX(separator, string-1 <, ...string-n>)

```
Data cat4;
a='The Olympic';
b='Arts Festival';
c='includes works by';
d='Dale Chihuly.';
Result=catx('***',a,b,c,d);
Put result $char.;
Run;
```

```
Data cat5;
Separator='%%$%%';
a='  The   Olym';
b='pic Arts Festi';
c='  val includes works by D  ';
d='ale Chihuly.';
Result=catx(separator,a,b,c,d);
Put result $char.;
Run;
```

----------Functions that remove blanks from string----------

LEFT

Left aligns a SAS character expression

Syntax :- LEFT(string)

```
Data remblank1;
a='        My Name Is Ram';
b=left(a);
Run;
```

RIGHT

Right aligns a character expression

Syntax :- RIGHT(string)

```
Data remblank2;
a='My Name Is Ram        ';
b=right(a);
Run;
```

TRIM

Removes trailing blanks from character expressions and returns one blank if the expression is missing

Syntax :- TRIM(string)

```
Data remblank4;
Input part1 $ 1-10 part2 $ 11-20;
hasblank=part1||part2;
noblank=part1||part2;
Put hasblank;
Put noblank;
Datalines;
apple sauce
;
Run;
```

```
Data remblank5;
Input part1$ part2$ ;
hasblank=part1||part2;
noblank=trim(part1)||part2;
Put hasblank;
Put noblank;
Datalines;
```

```
apple   sauce
;
Run;
Data remblank5;
Input part1$1-8 part2$9-13 part3$15-20 ;
hasblank=part1||part2||part3;
noblank=trim(part1)||trim(part2)||part3;
Put hasblank;
Put noblank;
Datalines;
apple   sauce mixer
orange       hand
;
Run;
Data remblank6;
x="    ";
y=">"||trim(x)||"<";
Put y;
Run;
```

TRIMN

Removes trailing blanks from character expressions and returns a null string (zero blanks) if the expression is missing

Syntax :- TRIMN(string)


```
Data remblank6a;
x="    ";
z=">"||trimn(x)||"<";
put z;
Run;
```

Difference between Trim and Trimn

When expression is there both Trim and Trimn gives same result,
If expression is missing Trim returns one blank space but Trimn gives zero blank space.

STRIP

Returns a character string with all leading and trailing blanks removed

Syntax :- STRIP(string)

```
Data remblank3;
Input string $char8.;
Original = '*' || string || '*';
Stripped = '*' || strip(string) || '*';
Datalines;
abcd
  abcd
    abcd
abcdefgh
 x y z
;
Run;
```

COMPRESS

Removes specific characters from a character string
Returns a character string with specified characters removed from the original string.
Syntax :- COMPRESS(<source><, chars><, modifiers>)

Examples:-

Compressing Blanks

```
Data remblank7;
a='          AB              C D             ';
b=compress(a);
Run;
```

```
Data remblank7a;
x='1   2       3   4   5';
y=compress(x);
Put y;
Run;
```

Compressing Lowercase Letters

```
Data remblank7b;
x='123-4567-8901 B b 234-5678-9012 c';
y=compress(x,'abcd');
put y;
Run;
```

```
Data remblank7c;
x='123-4567-8901 B b 234-5678-9012 c';
y=compress(x,'abcd', 'I');
put y;
Run;
```


Compressing Upper case Letters

```
Data remblank7B;
x='123-4567-8901 B b 234-5678-9012 c';
y=compress(x,'ABCD');
put y;
Run;
```

```
Data remblank7B;
x='123-4567-8901 B b 234-5678-9012 c';
y=compress(x,'ABCD', 'I');
Run;
```

Compressing Space Characters

```
Data remblank7c;
x='1   2   3   4   5';
y=compress(x,,'s');
Run;
```

Keeping Characters in the List

```
Data remblank7d;
x='Math A English B Physics A';
y=compress(x,'ABCD','k');
Run;
```

```
Data remblank7d;
x='Math a English b Physics a';
y=compress(x,'abcd');
Run;
Data remblank7d;
x='Math a English b Physics a';
y=compress(x,'abcd','k');
put y;
Run;
```

Separating Numbers Or Text from Alphanumeric data

```
Data ds;
Infile datalines;
Input NAME $20.;
Datalines;
sas123sap
java456oracle
;
Run;
Data ds1;
Set ds;
a=compress(name,' ','a');
Run;
Data ds2;
Set ds;
a=compress(name,' ','AK');
Run;
data _null_ ;
string='StudySAS Blog! 17752. ' ;
string1=compress(string,'') ; *Compress spaces. This is default
string2=compress(string,'','ak');*Compress alphabetic chars(1,2etc)
string3=compress(string,'','d') ; *Compress numerical values
string4=compress(string,'','l');*Compress  lowercase characters
string5=compress(string,'','u');*Compress uppercase characters
string6=compress(string,'A','k');*Keeps only specified characters
string7=compress(string,'!.','P');*Compress Punctuations only(ALL SPECIAL CHARACTER)
string8=compress(string,'s','i');*upper/lower case specified characters
string9=compress(string,'','a');*Compress all upper\lower case  characters
string10=compress(string,'','s') ; * Compress or delete spaces
string11=compress(string,'','kd') ; *Compress alphabets (Keeps only digits)
put string1= ;
put string2= ;
put string3= ;
put string4= ;
put string5= ;
put string6= ;
put string7= ;
put string8= ;
```

```
put string9= ;
put string10=;
put string11=;
run ;
```

COMPBL

Removes multiple blanks from a character string.

Syntax:-Compbl(source)

```
Data remblank9;
x='my    name    is    ram';
y=compbl(x);
Run;

Data remblank9a;
x='My    ';
y='    Name  ';
z='    is Ram';
a=x||y||z;
b=compbl(a);
Run;

Data ds1;
Infile datalines;
Input id$ fname$ lname$ sal;
Datalines;
001 mohan arisela 60000
002 padma narni 45000
003 varma maddina 50000
;
Run;

Data remblank10;
Set ds1;
Name1=fname||lname;
Name2=cat(fname,lname);
Name2a=cat(trim(fname),lname);
Name3=compbl(fname||lname);
Run;
```

----------Functions that substitute letters or words in string----------

TRANSLATE

Replaces specific characters in a character expression

```
Data trns1;
x=translate('XYZW','AB','VW');
Put x;
Run;

Data trns2;
x=translate('abc','sh', 'cg');
Put x;
Run;
```

TRANWRD

Replaces or removes all occurrences of a word in a character string

Syntax:-TRANWRD(source,target,replacement)

```
Data trnw1;
name='Mrs.Krishna';
name1=tranwrd(name, "Mrs.", "Mr.");
put  name name1;
run;
```

```
Data trnw2;
Infile datalines;
Input salelist $;
target='FISH';
replacement='NIP';
salelist1=tranwrd(salelist,target,replacement);
Datalines;
CATFISH
;
Run;
```

```
Tranwrd function returns
Length of new variable is 200
```

```
Data trnw2a;
Infile datalines;
Input salelist $;
length target $10 replacement $3;
target='FISH';
replacement='NIP';
salelist1=tranwrd(salelist,target,replacement);
Datalines;
CATFISH
;
Run;
```


The LENGTH statement left-aligns TARGET and pads it with blanks to the length of **10.** This causes the TRANWRD function to search for the character string 'FISH ' in SALELIST Because the search fails, this line is written to the SAS log: CATFISH

You can use the TRIM function to exclude trailing blanks from a target or replacement variable. Use the TRIM function with TARGET

```
Data trnw2b;
Infile datalines;
Input salelist $;
length target $10 replacement $3;
target='FISH'; replacement='NIP';
salelist1=tranwrd(salelist,trim(target),replacement);
Datalines;
CATFISH
;
Run;
```

----------Functions that searches for characters----------

INDEX

Searches a character expression for a string of characters

Syntax:- INDEX(source, excerpt)

```
Data ind1;
a='ABC.DEF (X=Y)';
b='D';
x=index(a,b);
Put x;
Run;
```

```
Data ind2;
a='ABC.DEF (X=Y)';
b='X=Y';
x=index(a,b);
Put x;
Run;
```

```
Data ind3;
Infile datalines;
input name $ 1-12 age;
Datalines;
Harvey Smith 30
John West    35
Jim Cann     41
James Harvey 32
Harvy Adams  33
;
Run;
```

Now, let's use the index function to find the cases with "Harvey" in the name

```
Data ind3a;
Set ind3;
x = index(name, "Harvey");
Run;
```

INDEXC

Searches a character expression for special characters, and returns the position of the characters

Syntax:-INDEXC(source,excerpt-1<,... excerpt-n>)

```
Data indc1;
a='ABC.DEP (X2=Y1)';
x=indexc(a,'.');
Run;
```

```
Data indc2;
a='ABC.DEP (X2=Y1)';
b='=';
x=indexc(a,b);
Run;
```

INDEXW
Searches a character expression for a specified string as a word
Syntax:- INDEXW(source, excerpt<,delimiter>)

```
Data indw1;
s='asdf adog dog';
p='dog';
x=indexw(s,p);
Run;
```

```
Data indw2;
s='abcdef x=y';
p='def';
x=indexw(s,p);
Run;
```

---------- *Other Character Functions* ----------

LENGTH
Returns length of string
Syntax:- LENGTH(string)

```
Data len;
a='Mr.Krishna';
b=length(a);
Run;
```


REVERSE
Returns string in reverse order
Syntax:- REVERSE(string)

```
Data rev;
a='Mr.Krishna';
b=reverse(a);
Run;
```

QUOTE
Ads double quotes to character values
Syntax:- QUOTE(string)

```
Data quot1;
a='Mr.Krishna';
b=quote(a);
Run;
```

DEQUOTE
Removes double quotes to character values
Syntax:- DEQUOTE(string)

```
Data quot2;
Set quot1;
c=dequote(a);
Run;
```

```
Quote & Dequote functions returns
   Length of new variable is 200
```

```
Data quot3;
Infile datalines;
Input id name$ sal;
Datalines;
001 abc 5000
002 def 6000
003 xyz 7000
;
Run;
Data quot3a;
Set quot3;
name1=quote(name);
name2=quote(trim(name));
name3=dequote(name2);
Run;
```

RANK

Returns the position of a character in the ASCII or EBCDIC collating sequence.

Syntax:-RANK(x)

The RANK function returns an integer that represents the position of the first character in the character expression. The result depends on your operating environment.


```
Data rnk1;
Infile datalines;
Input id name$ sal;
Rank_var=RANK(name);
Datalines;
001 clarc 5000
002 def 4000
003 clark 7000
;
Run;
Data rnk2 ;
a=Rank('A');
b=Rank('krishna'); /* It gives position of first character only*/
Run;
```

REPEAT

Returns a character value that consists of the first argument repeated n+1 times.

Syntax:- Repeat(Argument,n)

```
Data rep;
Infile datalines;
Input id name$ sal;
x=repeat(name,10);
Datalines;
001 clarc 5000
002 def 4000
003 clark 7000
;
Run;
```

SOUNDEX

Encodes a string to facilitate searching.
Encodes a string and gives same result for same pronunciation strings in variable
Syntax:- SOUND(Argument)

```
Data snd;
Infile datalines;
Input id name$ sal;
y=soundex(name);
Datalines;
001 clarc 5000
002 def 4000
003 clark 7000
;
Run;
```

COLLATE

Returns a character string in ASCII or EBCDIC collating sequence.
Syntax:- (start-position<,end-position>) | (start-position<,,length>)

```
Data col1;
x=collate(45,99);
put @1 x ;
Run;

Data col2;
x=collate(1,,49);
put @1 x ;
Run;
```

ASCII Result

```
Data col3;
x=collate(48,,10);        /*start-position<,,length*/
y=collate(48,57);         /*start-position<,end-position */
put @1 x @14 y;
Run;
```

EBCIDIC Result

```
Data col4;
x=collate(240,,10);       /*start-position<,,length*/
y=collate(240,249);       /*start-position<,end-position */
put @1 x @14 y;
Run;
```

The maximum end-position for the EBCDIC collating sequence is 255.
ASCII collating sequences, end-position values between 0 and 127

STANSYS SOFTWARE SOLUTIONS
#7-1-621/113(67/3RT), Beside: Nagarjuna High School, Near: S.R.Nagar Community Hall, S.R.Nagar, Hyd-38|Ph:7671076710 | 9542195422 |www.stansys.co.in|sas@stansys.co.in


STANSYS
SOFTWARE SOLUTIONS

NUMERIC FUNCTIONS

MEAN

Returns the arithmetic mean (average)
Argument is numeric At least one non-missing argument is required otherwise, the function returns a missing value

Syntax: - MEAN(argument<,argument,...>)

```
Data ds1;
x1=mean(2,.,.,6);
x2=mean(2,4,5,6);
x3=mean(x1-x2); /*x3=mean(4-4.25)=-0.25/1=-0.25*/
x4=mean(of x1-x2); /*it means x1, x2 means 4,4.25 means 8.25/2=4.125*/
x5=mean(x1,x2);
Run;
```

```
Data ds1;
x1=mean(2,.,.,6);
x2=mean(2,4,5,6);
x3=mean(2,4,5,6);
y=mean(3,3,5);
x4=mean(x1-x2);
x5=mean(of x1-x3);
x6=mean(x1,x2);
xa=mean(2,5);
x7=mean(of x:);
x8=mean(of x1--x4);
Run;
```

MEDIAN

Computes median values.
Syntax: - MEDIAN(value1<, value2, ...>)

```
Data ds2;
x=median(2,4,1,3);
y=median(5,8,0,3,4);
z=median(5,.,0,.,4);
Run;
```

Difference between MEAN & MEDIAN
Mean will give average of numeric values
Ex:- x=mean(70,60,80,75,90)
x=70+60+80+75+90/5
x=375/5=75

In MEDIAN data will arrange from lowest to highest
in that data middle no is MEDIAN value
60,70,75,80,90
75 is mid value which is median value

Ex:- x=median(2,4,1,3);
in above example mid value is 4,1
it means 4+1=5
median value is 5/2=2.5;

MIN

Returns the smallest value

Syntax: - MIN(argument,argument,...)

```
Data ds3;
x1=min(7,4);
x2=min(2,.,6);
x3=min(2,-3,1,-1);
x4=min(0,4);
x6=min(of x1-x3);
x7=min(x1,x3);
Run;
```

MAX

Returns the largest value

Syntax:-MAX(argument,argument,...)

```
Data ds4;
x=max(8,3);
x1=max(2,6,.);
x2=max(2.-3,1,-1);
x3=max(3,.,-3);
x4=max(.,.,.);
x5=max(of x1-x3);
Run;
```

Argument is numeric. At least two arguments are required. The argument list may consist of a variable list, which is preceded by OF.

The MAX function returns a missing value (.) only if all arguments are missing.

RANGE

Returns the range of values

Syntax:- RANGE(argument,argument,...)

Argument is numeric At least one non missing argument is required. Otherwise, the function returns a missing value. The argument list can consist of a variable list, which is preceded by OF.

The RANGE function returns the difference between the largest and the smallest of the non missing arguments.

```
Data ds5;
x1=range(.,.);
x2=range(-2,6,3);
x3=range(2,6,3,.);
x4=range(1,6,3,1);
x5=range(of x1-x3);
run;
```

SUM

Returns the sum of the non missing arguments

Syntax:-SUM(argument,argument, ...)

Argument is numeric If all the arguments have missing values, the result is a missing value. The argument list can consist of a variable list, which is preceded by OF

```
Data ds6a;
x1=sum(4,9,3,8);
x2=sum(4,9,3,8,.);
x3=sum(of x1-x2);
Run;

Data ds6b;
x1=5;
x2=6;
x3=4;
x4=9;
y1=34;
y2=12;
y3=74;
y4=39;
result=sum(of x1-x4, of y1-y5);
Run;

Data ds6c;
x1=55;
x2=35;
x3=6;
x4=sum(of x1-x3, 5);
Run;

Data ds6d;
x1=7;
x2=7;
x5=sum(x1-x2);
Run;

Data ds6e;
y1=20;
y2=30;
x6=sum(of y:);
Run;
```

/*Sum Statement*/

Adds the result of an expression to an accumulator variable
Syntax:-variable+expression;

```
Data ds6;
x1=sum(4+9+3+8);
x2=sum(4+.+9+3+8+.);
Run;
```

SUM Function returns the sum of non missing values
Ex:- x2=sum(**4**,**.**,**9**,**3**,**8**,**.**); it gives value **24**
SUM Statement Adds the value into variable with non missing values
its won't consider missing values.
if missing value are there value is **.**
ex:- x2=sum(**4**+**.**+**9**+**3**+**8**+**.**); it gives value **.**

VAR

Returns the variance

Syntax:-VAR(argument,argument, ...)

argument is numeric. At least two non missing arguments are required. Otherwise, the function returns a missing value. The argument list can consist of a variable list, which is preceded by OF.

```
Data ds13;
x1=Var(4,2,3.5,6);
x2=Var(4,6,.);
x3=Var(of x1-x2);
Run;
```

SQRT

Returns the square root of a value.

Syntax :-SQRT(argument)

argument is numeric and must be nonnegative

```
Data ds14;
x1=sqrt(36);
x2=sqrt(25);
x3=sqrt(4.4);
x4=sqrt(-49);
Run;
```


NMISS

Returns the number of missing values

Syntax :-NMISS(argument<,...argument-n>)

argument is numeric. At least one argument is required. The argument list may consist of a variable list, which is preceded by OF.

```
Data ds15;
x1=nmiss(1,0,.,2,5,.);
x2=nmiss(1,0);
x3=nmiss(of x1-x2); /*x1=2 x2=0 so 2,0 it gives 0*/
Run;
```

N

Returns the number of non missing values

Syntax:-N(argument<,...argument-n>)

argument is numeric. At least one argument is required. The argument list may consist of a variable list, which is preceded by OF.

```
Data ds16;
X1=n(1,0,.,2,5,.);
X2=n(1,0);
X3=n(of x1-x2);
Run;
```

CMISS

Counts the number of missing arguments.

Syntax: - CMISS (argument-1 <, argument-2,...>)

```
Data ds;
Set sashelp.class;
If name='Barbara' then height=.;
If name='Philip' then weight=.;
Run;
```

```
Data ds2;
Set ds;
If cmiss(of name--weight);
RUN;
```

```
Data ds3 ds4;
Set ds;
If cmiss (of name--weight) then output ds2;
Else output ds3;
Run;
```

COALESCE

Returns the first non-missing value from a list of numeric arguments.

Syntax: - COALESCE(argument-1<..., argument-n>)

```
Data Ds;
X = coalesce(42, .);
Y = coalesce(., 7, ., ., 42);
Z = coalesce(.5,10,.,12);
Run;
```


COALESCEC

Returns the first non-missing value from a list of character arguments.

Syntax: - COALESCE(argument-1<..., argument-n>)

```
Data ds;
X = coalescec(' ', 'hello');
Y = coalescec (' ', 'goodbye', 'hello');
Z = coalescec ('sas', 'stansys', 'krishna');
Run;
```

```
Coalescec function returns
Length of new variable is 200
```

LAG

Returns values from a queue.

Syntax:- LAG<n>(argument)

```
Data lg1;
  input x @@;
  a=lag1(x);
  b=lag2(x);
  c=lag3(x);
  d=lag(x);
  datalines;
1 2 3 4 5 6
;
Run;
Data lg2;
  input x @@;
  y=lag1(x+10);
  z=lag2(x);
  datalines;
1 2 3 4 5 6
;
Run;
```

********** LOCF (Last Observations Carry Forward) **********


```
DATA TEST;
INPUT PAT VISIT LABSTD;
CARDS;
101 1 0.1
101 2 0.3
101 3 .
101 5 0.1
101 6 0.9
102 1 0.7
102 2 0.3
102 3 .
102 5 0.4
102 6 0.9
;
Run;
Data test1;
Set test;
LABSTD1=lag(LABSTD);
If LABSTD=. then  LABSTD=LABSTD1;
Run;
```

```
DATA TEST2;
INPUT PAT VISIT LABSTD;
CARDS;
101 1 0.1
101 2 0.3
101 3 .
101 5 0.1
101 6 0.9
102 1 .
102 2 0.3
102 3 .
102 5 0.4
102 6 0.9
;
RUN;
Data test3;
Set test2;
LABSTD1=lag(LABSTD);
Run;
Proc sort data=test3;
By PAT;
Run;
```

Way#1

```
Data test4(drop=LABSTD1);
Set test3;
By pat;
If first.PAT and LABSTD=. then LABSTD=.;
Else if LABSTD=. then  LABSTD=LABSTD1;
Run;
```

Way#2

```
Data test3(drop=x y);
Set test2;
if PAT=101 then x=lag(labstd);
if PAT=102 then y=lag(labstd);
if PAT=101 and labstd=. then labstd=x;
if PAT=102 and labstd=. then labstd=y;
Run;
```

How can i find out difference between value1 values with value1 values like 40-20, 60-40, 80-60 etc?

Example:-

```
Data ds;
Infile datalines;
Input id  value1;
Datalines;
001 20
002 40
003 60
004 80
005 100
;
Run;
Data ds2(drop=value2 value4);
Set ds;
value2=lag(value1);
value3=sum(value1,-value2);
value4=value1-value2;
Run;
```

ANY DIGIT

Searches a character string for a digit and returns the first position at which it is found

Syntax:- ANYDIGIT(string <,start>)


```
DATA SEARCH_NUM;
INPUT STRING $60.;
dg = ANYDIGIT(STRING);
DATALINES;
This line has a 56 in it
two numbers 123 and 456 in this line
No digits here
;
Run;
```

ANY SPACE

Searches a character string for space returns the first position at which it is found

Syntax**:- ANYSPACE(string <,start>)**

```
DATA SEARCH_SPACE;
INPUT STRING $60.;
SP= ANYSPACE(STRING);
DATALINES;
This line has a 56 in it
two numbers 123 and 456 in this line
No digits here
;
Run;
```

How can you separate numeric values from alpha numeric value

```
DATA EN;
INPUT STRING $60.;
START = ANYDIGIT(STRING);
END = ANYSPACE(STRING,START);
IF START NE 0 THEN NUM = INPUT(SUBSTR(STRING,START,END-START),9.);
DATALINES;
This line has a 56 in it
two numbers 123 and 456 in this line
No digits here
;
Run;
```

Decimal Handling Functions

CEIL

Returns integer that is greater than or equal to the argument, fuzzed to avoid unexpected floating-point results

Syntax :-CEIL (argument)

```
Data ds7;
var1=2.1;
a=ceil(var1);
Run;
Data ds7;
b=ceil(-2.4);
c=ceil(1+1.e-11);
d=ceil(-1+1.e-11);
e=ceil(1+1.e-13);
f=ceil(223.456);
g=ceil(763);
h=ceil(-223.456);
Run;
```

FLOOR

Returns integer that is less than or equal to the argument, fuzzed to avoid unexpected floating-point results.

Syntax :-FLOOR (argument)

```
Data ds8;
var1=2.1;
a=floor(var1);
Run;
Data ds8;
b=floor(-2.4);
c=floor(1+1.e-11);
d=floor(-1+1.e-11);
e=floor(1+1.e-13);
f=floor(223.456);
g=floor(763);
h=floor(-223.456);
Run;
```

ABS

Returns the absolute value

Syntax :-ABS (argument)

If value is negative it converts into positive.

```
Data ds9;
x1=abs(2.4);
x2=abs(-3);
Run;
```

INT

Returns the integer value, fuzzed to avoid unexpected floating-point results.

Syntax:-INT(argument)

```
Data ds10;
x1=INT(2.4);
x2=INT(2.5);
x3=INT(2.8);
X4=INT(-2.4);
Run;
```

MOD

Returns the remainder from the division of the first argument by the second argument, fuzzed to avoid most unexpected floating-point results.

Syntax:- MOD (argument-1, argument-2)


```
Data ds11;
X1=MOD(10,3);
Run;
Data ds;
A=123456;
X=INT(A/1000);
Y=MOD(A,1000);
Z=MOD(INT(A/100),100);
Run;
```

How to select odd numbers, even numbers and prime numbers from SAS dataset.

```
Data prime;
Do i=1 to 1000;
Output;
End;
Stop;
Run;
Data add even;
Set prime;
If mod(i,2)=0 then output even;
else output add;
/*If mod(i,2)=0 then output even;*/
/*If mod(i,2)=1 then output add;*/
Run;
```

```
Data prime_num;
Set prime;
If mod(i,1)= 0 and mod(i,i) = 0 and mod(i,2) ^= 0
and mod(i,3) ^= 0 and mod(i,5)^=0 and mod(i,7)^=0 then
pirime_numbers=i;
If i=3 or i=2 or i=5 or i=7 then pirime_numbers=i;
If not missing(pirime_numbers);
Put pirime_numbers;
Drop i;
Run;
```

ROUND

Rounds the first argument to the nearest multiple of the second argument, or to the nearest integer when the second argument is omitted.

Syntax:- ROUND (argument <,rounding-unit>)

```
Data ds12;
x1=ROUND(2.4);
x2=ROUND(2.5);
x3=ROUND(2.8);
X4=ROUND(-2.4);
X5=ROUND(-2.5);
Run;
Data rounding;
d1 = round(1234.56789,100) ;
d2 = round(1234.56789,10) ;
d3 = round(1234.56789,1)  ;
d4 = round(1234.56789,.1)  ;
d5 = round(1234.56789,.01)  ;
d6 = round(1234.56789,.001)  ;
d7 = round(1234.56789,.0001)  ;
d8 = round(1234.56789,.00001) ;
d9 = round(1234.56789,.1111)  ;
/* d10 has too many decimal places in the value for rounding-unit.*/
d10 = round(1234.56789,.11111) ;
Run;
```

Data type Converting Functions

INPUT

Converts data values from character to numeric data type with help of Informat

Syntax:- Input(variable, informat);

Example:-

```
Data ds1;
Infile datalines;
Input id$ name$ sal;
Datalines;
001 abc 60000
002 def 45000
003 xyz 50000
;
Run;
```

```
Data cn /*(drop=id rename=(id1=id))*/;
Set ds1;
id1=input(id, best.);
Run;
```

PUT

Converts data values from numeric to character data type with help of Format

Syntax:- put(variable, format);

Example:-

```
Data ds2;
Infile datalines;
Input id name$ sal;
Datalines;
001 abc 60000
002 def 45000
003 xyz 50000
;
Run;
```

```
Data nc/*(drop=id rename=(id1=id))*/;
Set ds2;
id1=put(id, $8.);
Run;
```

DATE FUNCTIONS

How Dates Works in SAS

The SAS system stores Date values as the number of elapsed days Since January 1, 1960

Ex:- January 03,1960 is stored as 2
January 02,1960 is stored as 1
January 01,1960 is stored as 0
December 31,1959 is stored as -1
December 30,1959 is stored as -2
December 31,1960 is stored as 365

The SAS system stores Time values as the number of elapsed seconds since midnight of that particular day.

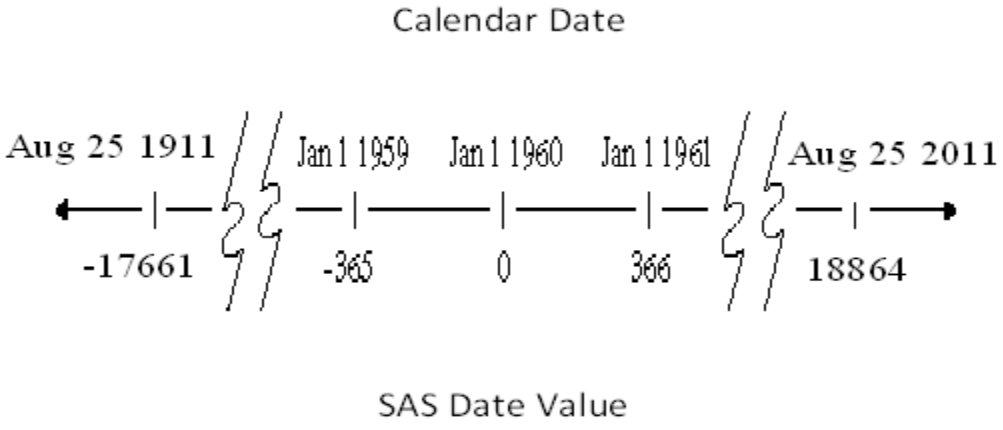
The SAS system stores Datetime values as the number of elapsed seconds since midnight January 1, 1960 12:00 am

And SAS system stores Date variables as the number of days since midnight January 1, 1960

Dates before January 01, 1960 are negative integers, after January 01, 1960 are positive integers

SAS Dates are valid from A.D. 1582 to A.D. 19,900.

How SAS Converts Calendar Dates to SAS Date Values



DATE

Returns the current date as a SAS date value
Returns today's date as a SAS date value

Syntax: - DATE()

```
Data ds1;
date1=date();
Run;
```

```
Data ds1a;
date1=date();
Format date1 date9. ;
Run;
```

TODAY

Returns the current date as a SAS date value

Syntax:-TODAY()

```
Data ds2;
Day=today();
Format day date9.;
Run;
```

DATETIME

Returns the current date and time of day as a SAS datetime value

Syntax:-DATETIME()

```
Data ds3;
a=datetime();
Format a datetime20.;
Run;
```

TIME

Returns the current time of day

Syntax:-TIME()

SAS assigns current system time as a SAS time value corresponding to 15:32:00 if the following statements are executed exactly at 3:32 PM:
Its gives 24 hour format

```
Data ds4;
Time=time();
Format time time8. ;
Run;
```

DAY

Returns the day of the month from a SAS date value

Syntax:-DAY()

```
Data ds5;
a='29Jan2010'd;
Day=day(a);
Run;
```

```
Data ds5a;
a=date();
b= day(a);
Format a date9.;
Run;
```

WEEK

Returns the week-number value

Syntax:-WEEK (<SAS_Date>, <descriptor>)

```
Data ds6;
X=week('29Jan2010'd);
Y= week('10Feb2010'd);
Z= week('31Dec2010'd);
Run;
Data ds6a;
X=date();
Y=week(x);
Format x date9. ;
Run;
```

WEEKDAY

Returns the day of the week from a SAS date value

For example 17Oct1991 Returns 5 because 17Oct1991 was Thursday so it's 5

Syntax:-WEEKDAY(date)

```
Data ds7;
week1=weekday('16Mar1997'd);
Run;
Data ds7a;
a=date();
week1=weekday(a);
Run;
```

MONTH

Returns the month from a SAS date value

Syntax:-MONTH (date)

```
Data ds8;
a='29Jan2010'd;
Mon=month(a);
Run;
Data ds8a;
a=today();
Mon=month(a);
Run;
```

QTR

Returns the quarter of the year from a SAS date value

Syntax:-QTR(date)

```
Data ds9;
a='29Jan2010'd;
Quarter=qtr(a);
Run;
Data ds9a;
a='15Nov2010'd;
b=today();
Quarter1=qtr(a);
Quarter2=qtr(b);
Run;
```

YEAR

Returns the year from a SAS date value
Gives four-digit numeric value that represents the year

Syntax:-YEAR(date)

```
Data ds10;
Date='25dec97'd;
y=year(date);
Run;
```

DHMS

Returns a SAS datetime value from date, hour, minute, and second

Syntax: - DHMS (date, hour, minute, second)


```
Data ds11;
a=dhms('15Nov2010'd,10,02,15);
Format a datetime. ;
Run;
Data ds11a;
a=dhms('15Nov2010'd,10,02,61);
b=dhms('15Nov2010'd,10,02,61);
Format a datetime. ;
Format b datetime20. ;
Run;
Data ds11b;
a=dhms('15Nov2010'd,10,.2,11);
Format a datetime.;
Run;
```

HMS

Returns a SAS time value from hour, minute, and second values

Syntax: - HMS (hour, minute, second)

```
Data ds12;
a=HMS(10,02,15);
Format a time.;
Run;
```

```
Data ds12;
a=HMS(10,02,15);
b=HMS(10,02,15);
c=HMS(10,02,15);
Format a time.;
Format b time5.;
Format c time8.;
Run;
```

HOUR

Returns the hour from a SAS time or datetime value

Syntax: - HOUR (<time | datetime>)

```
Data ds13;
a=hour('10:30't);
Run;
Data ds13a;
a='10:30:05't;
b=hour(a);
Format a time8. ;
Run;
```

MINUTE

Returns the minutes from a SAS time or datetime value

Syntax: - Minute (<time | datetime>)

```
Data ds14;
a='10:30:05't;
b=MINUTE(a);
Format a time5.;
Run;
```

SECOND

Returns the seconds from a SAS time or datetime value

Syntax: - Second (<time | datetime>)

```
Data ds14a;
a='10:30:05't;
b=second(a);
Format a time. ;
Run;
```

DATEJUL

Converts a Julian date to a SAS date value

Syntax: - DATEJUL(Julian-date)

Julian-date

Specifies a SAS numeric expression that represents a Julian date A Julian date in SAS is a date in the form yyddd or yyyyddd, Where yy or yyyy is a two-digit or four-digit integer that represents the year and ddd is the number of the day of the year
The value of ddd must be between 1 and 365 (or 366 for a leap year).

```
Data ds15;
a=Datejul(10001);
Format a date9.;
Run;
Data ds15a;
a=Datejul(10365);
Format a date9.;
Run;
```

JULDATE

Returns the Julian date from a SAS date value

Syntax: - JULDATE (date)

The JULDATE function converts a SAS date value to a five- or seven-digit Julian date If date falls within the 100-year span defined by the system option YEARCUTOFF=, the result has five digits:
The first two digits represent the year, and the next three digits represent the day of the year (1 to 365, or 1 to 366 for leap years)
Otherwise, the result has seven digits: the first four digits represent the year, and the next three digits represent the day of the year. For example, if YEARCUTOFF=1920, JULDATE would return 97001 for January 1, 1997,
and return 1878365 for December 31, 1878.


```
Data ds16;
a=juldate('01Jan2010'd);
Run;
Data ds16a;
a=date();
b=juldate(a);
Format a date9.;
Run;
```

MDY

Returns a SAS date value from month, day, and year values

Syntax: - MDY (month,day,year)

Month

Specifies a numeric expression that represents an integer from 1 through 12.

Day

Specifies a numeric expression that represents an integer from 1 through 31.

Year

Specifies a two-digit or four-digit integer that represents the year
The YEARCUTOFF= system option defines the year value for two-digit dates

```
Data ds17;
x_birthday=mdy(8,27,90);
y_birthday=mdy(05,30,2009);
Format x_birthday worddate20. ;
Format y_birthday weekdate30. ;
Run;
```

YYQ

Returns a SAS date value from the year and quarter year

Year

Specifies a two-digit or four-digit integer that represents the year
The YEARCUTOFF= system option defines the year value for two-digit dates

Quarter

Specifies the quarter of the year (1, 2, 3, or 4)

Syntax: - YYQ(year, quarter)

```
Data ds18;
DateValue1=yyq(2001,3);
DateValue2=yyq(09,2);
Format DateValue1 date7.;
Format DateValue2 date7.;
Run;
```

TIMEPART

Extracts a time value from a SAS datetime value

Syntax: - TIMEPART (datetime)

```
Data ds19;
x=datetime();
y=timepart(x);
Format X datetime. Y time. ;
Run;
```


DATEPART

Extracts the date from a SAS datetime value

Syntax: - DATEPART(datetime)

```
Data ds20;
X=datetime();
Y=datepart(x);
Format x datetime. y ddmmyy10.;
Run;
Data ds20a;
x=datepart ('01Jan2010:05:30:26'dt);
Format x ddmmyy8.;
Run;
Data ds1;
Input id$ fname$ lname$ sal dob datetime.;
Format dob datetime. date date9. time time8.;
Date=datepart(dob);
Time=timepart(dob);
Datalines;
001 mohan arisela 60000 10jan1983:10:30:15
002 padma narni 45000 22feb1983:20:23:52
003 varma maddina 50000 30mar1983:06:55:25
;
Run;
```

INTCK

Returns the integer count of the number of interval boundaries between two dates, two times, or two datetime values

Syntax: - INTCK(interval, from, to)

Interval

Specifies a character constant, a variable, or an expression that contains a time interval such as SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QTR, SEMIYEAR and YEAR

```
DATA ds21;
BDATE='10SEP2008'D;
EDATE='14SEP2010'D;
ACTDATE=INTCK('DAYS', BDATE, EDATE);
RUN;
DATA ds21a;
BDATE='10SEP2008'D;
EDATE='14SEP2010'D;
ACTDATE=INTCK('months', BDATE, EDATE);
RUN;
DATA ds21b;
BDATE='10SEP2008'D;
EDATE='14SEP2010'D;
ACTDATE=INTCK('Semiyear', BDATE, EDATE);
RUN;
DATA ds21c;
y=trim('year     ');
date1='1sep1991'd + 300;
date2='1sep2001'd - 300;
Years= INTCK(y,date1,date2);
RUN;
```


YRDIF

Returns the difference in years between two dates

Syntax: - YRDIF (sdate,edate,basis)

sdate Specifies a SAS date value that identifies the starting date

edate Specifies a SAS date value that identifies the ending date

basis

Identifies a character constant or variable that describes how SAS calculates the date difference. The following character string is valid: '30/360' - Specifies a 30-day month and a 360-day year in calculating the number of years. Each month is considered to have 30 days, and each year 360 days, regardless of the actual number of days in each month or year

```
DATA ds22;
BDATE='10SEP2000'D;
EDATE='14SEP2010'D;
ACTYEARS=YRDIF(BDATE, EDATE, 'ACTUAL');
Format BDATE date9. EDATE date9. ;
RUN;
```

```
DATA ds22a;
Sdate='16Oct1998'd;
Edate='16Feb2003'd;
y30360=yrdif(sdate, edate, '30/360');
Yactact=yrdif(sdate, edate, 'ACT/ACT');
yact360=yrdif(sdate, edate, 'ACT/360');
yact365=yrdif(sdate, edate, 'ACT/365');
Run;
DATA ds22b;
Sdate='16Oct1998'd;
Edate='16Feb2003'd;
YRDIFF=yrdif(sdate, edate, '30/360');
DAYDIFF=yrdif(sdate, edate, 'ACT/365');
Run;
```

INTNX

Increments a date, time, or datetime value by a given interval or intervals, and returns a date, time, or datetime value

Syntax: -

INTNX (interval<multiple><.shift-index>, start-from, increment<, alignment>)

Interval

Specifies a character constant, a variable, or an expression that contains a time interval such as SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QTR, SEMIYEAR and YEAR


```
Data ds23;
Yr=intnx('year','05feb94'd,3);
Format yr date7. ;
Run;
Data ds23a;
Next=intnx('semiyear','01jan97'd,1);
Format next date9.;
Run;
Data ds23b;
X1='month     ';
X2=trim(x1);
Date='1jun1990'd - 100;
Next_month=intnx(x2,date,1);
Format Next_month date9.;
Run;
DATA DS23c;
TODAY1=TODAY(); FORMAT TODAY1 DATE9.;
CDATE=PUT (INTNX ('MONTH',TODAY1,0,'S'),DATE9.);
LMCDATE=PUT(INTNX('MONTH',TODAY1,-1,'S'),DATE9.);
BCDATE=PUT(INTNX('DAY',TODAY1,-1,'S'),DATE9.);
LMBCDATE=PUT(INTNX('MONTH',(TODAY1-1),-1,'S'),DATE9.);
BDATE=PUT(INTNX('MONTH',TODAY1,0,'B'),DATE9.);
EDATE=PUT(INTNX('MONTH',TODAY1,0,'E'),DATE9.);
RUN;
```

HOLIDAY

Returns a SAS date value for the holiday and year specified

Valid values for holiday are 'BOXING', 'CANADA', 'CANADAOBSERVED', 'CHRISTMAS', 'COLUMBUS', 'EASTER', 'FATHERS', 'HALLOWEEN', 'LABOR', 'MLK', 'MEMORIAL', 'MOTHERS', 'NEWYEAR','THANKSGIVING', 'THANKSGIVINGCANADA','USINDEPENDENCE', 'USPRESIDENTS', 'VALENTINES', 'VETERANS', 'VETERANSUSG', 'VETERANSUSPS', and 'VICTORIA'

For example: MOTHERS2011= HOLIDAY (' MOTHERS', 2011);

Syntax: - HOLIDAY ('holiday', year)

```
DATA DS24;
THANKSGIVING_2012=HOLIDAY (' THANKSGIVING ', 2012);
Format THANKSGIVING_2012 date9. ;
RUN;
```