BAP – ASSIGNMENT ON INTERESTING SAS FUNCTIONS

ASHMITHA BONAVENTURE.M

LIST OF FUNCTIONS

- INTCK
- INTNX
- DATDIF, YRDIF
- call MISSING
- SPEDIS
- notalpha, notdigit, notalnum
- HOLIDAY
- CLOSE
- Mentionable Syntaxes of a few

INTCK

- The following function counts the number of time intervals in a given time span.
  - The possible time intervals can be: DAY, WEEKDAY, WEEK, TENDAY, SEMIMONTH, MONTH, QTR, SEMIYEAR, YEAR.
  - From: specifies a SAS date, time or datetime value that identifies the beginning of the time span.
  - TO: specifies a SAS date, time or datetime value that identifies the end of the time span.
- USAGE : INTCK (‘interval’, from, to);
- Partial intervals are not counted.

INTCK SAS statement	Value
Weeks1=INTCK(‘week’ , ’31DEC2009’d , ’01JAN2010’d)	0
Months=INTCK(‘Month’ , ’31DEC2009’d , ’01JAN2010’d)	1
Years=INTCK(‘Year’ , ’31DEC2009’d , ’01JAN2010’d)	1
Week2=INTCK(‘week’ , ’31DEC2009’d , ’03JAN2010’d)	1

INTNX

- The INTNX Function determines the time based on start-from time and increments of the intervals.
- The function returns a SAS date, time or datetime values
  - Interval can be : DAY, WEEKDAY, WEEK, TENDAY, SEMIMONTH, MONTH, QTR, SEMIYEAR, YEAR
  - Start-from: specifies the starting SAS date, time, datetime.
  - Increment: specifies a negative (back to the past) or positive integer (to the future).
  - Alignment: forces the alignment of the returned date to be the beginning (‘b’), middle (‘m’), or end (‘e’) of the time interval. The default is the beginning.

SAS INTNX function	Result
INTNX(‘month’, ’01NOV2010’d, 5);	18718 (April 1, 2011)
INTNX(‘month’, ’01NOV2010’d, 5, ‘b’);	18718 (April 1, 2011)
INTNX(‘month’, ’01NOV2010’d, 5, ‘m’);	18732 (April 15, 2011)
INTNX(‘month’, ’01NOV2010’d, 5, ‘e’);	18747 (April 30, 2011)

DATDIF, YRDIF

- DATDIF counts # of dates between two dates.
- YRDIF counts # of years between two dates.
- General Syntax:
  - DATDIF(Start_date, End_date, basis);
  - YRDIF(Start_date, End_date, basis);
  - Start_Date specifies the starting date as a SAS date value.
  - End_Date specifies the end date as a SAS date value.
- Basis is a string specifies the basis for calculating the date or year difference. The basis is ‘n/m’ , where n is the # of days per months, and m is number of days per year. For example, ‘30/360’ uses 30 days per months to calculate # of months, and use 360 days to calculate # of years.

```
DATEDF1=DATDIF('01SEP1984'D,'01NOV2010'D, '30/360');
DATEDF2=DATDIF('01SEP1984'D,'01NOV2010'D, 'ACT/ACT');
YEARDF1=YRDIF('01SEP1984'D,'01NOV2010'D, ’30/360');
YEARDF2=YRDIF('01SEP1984'D,'01NOV2010'D, ‘ACT/ACT’);
```

Results :

Obs	**DATEDF1**	**DATEDF2**	**YEARDF1**	**YEARDF2**
1	9420	9557	26.1667	26.1662

CALL MISSING

- The CALL MISSING routine assigns a character missing value (a blank) to each character variable in the argument list. If the current length of the character variable equals the maximum length, the current length is not changed. Otherwise, the current length is set to 1.
- Syntax : CALL MISSING(varname1<, varname2, …>);
  - varname : specifies the name of SAS character or numeric variables.
- We can also mix character and numeric variables in the argument list.

```
prod='shoes';
invty=7498; sales=23759;
call missing(of _all_);
put prod= invty= sales=;
```

Results : prod= invty=. sales=.

SPEDIS

- Determines the likelihood of two words matching, expressed as the asymmetric spelling distance between the two words.
- For each category of spelling mistake, the function assigns penalty points. For example, if you get the first letter wrong, you incur a large penalty. If you place two letters in the wrong order (ie versus ei for example), you get a fairly small number of penalty points. When the function has checked for each category of errors, it divides the total penalty points by the length of the first string.

```
data words;
  input Operation $ Query $ Keyword $;
  Distance = spedis(query,keyword);
  Cost = distance * length(query);
  datalines;
match      fuzzy      fuzzy
singlet    fuzy       fuzzy
```

RESULTS:

Obs	Operation	Query	Keyword	Distance	Cost
1	match	fuzzy	fuzzy	0	0
2	singlet	fuzy	fuzzy	6	24

NOTALPHA, NOTDIGIT, NOTALNUM

- NOTALPHA : Searches a character string for a nonalphabetic character, and returns the first position at which the character is found.
- NOTDIGIT : Searches a character string for any character that is not a digit, and returns the first position at which that character is found.
- NOTALNUM : Searches a character string for a non-alphanumeric character, and returns the first position at which the character is found.
- We also have NO PUNCT or SPACE.
- For, String = "Abc123, yog376"

  NOTDIGIT(string) : 1

  NOTALPHA(string) : 4

  NOTALNUM(string) : 7

  NOTPUNCT(string) : 1

  NOTSPACE(string) : 1

HOLIDAY

- Returns a SAS date value of a specified holiday for a specified year.
- Syntax : HOLIDAY(‘holiday', year)
- year : is a numeric constant, variable, or expression that specifies a four-digit year. If you use a two-digit year, then you must specify the YEARCUTOFF= system option.

```
boxing = holiday('boxing', 2007);

format boxing date9.;

put boxing;
```

Results : 26DEC2007

- Comparisons :
  - In some cases, the HOLIDAY function and the NWKDOM function return the same result. For example, the statement HOLIDAY('THANKSGIVING', 2007); returns the same value as NWKDOM(4, 5, 11, 2007); .
  - In other cases, the HOLIDAY function and the MDY function return the same result. For example, the statement HOLIDAY('CHRISTMAS', 2007); returns the same value as MDY(12, 25, 2007);

CLOSE

- Closes a SAS data set.
- Syntax : CLOSE(data-set-id)
- CLOSE returns zero if the operation was successful, or returns a non-zero value if it was not successful. Close all SAS data sets as soon as they are no longer needed by the application.
- Examples : This example uses OPEN to open the SAS data set PAYROLL. If the data set opens successfully, indicated by a positive value for the variable PAYID, the example uses CLOSE to close the data set.

```
%let payid=%sysfunc(open(payroll,is));

%if &payid > 0 %then

   %let rc=%sysfunc(close(&payid));
```

EXTRA SYNTAXES

- UPCASE(character value) returns the character values all in UPPER case.
    - Ex: UPCASE(‘Mission street’) returns ‘MISION STREET’
- LOWCASE(character value) returns the strings all in lower case.
    - Ex: LOWCASE(‘Mission street’) returns ‘mission street’
- PROPCASE(character value) returns the value with 1st character upper case and the rest in lower cases.
    - Ex: PROPCASE(‘MISSION street’) returns ‘Mission Street’

CNTD.

- TRANWRD function replaces or removes all occurrences of a pattern of characters from within a character string.
- A situation using TRANWRD is to update existing variables in place, such as change ‘MISS’ to ‘MS.’, change ‘Doctor’ to Dr.’ and so on.
- Syntax : TRANWRD(source, target, replacement);

THANK YOU !