Working with Characters Data

MWSUG 2014
October 6th 2014
Chicago

CSUG Fall 2014
October 8th 2014
Calgary
Agenda

• What is “Character Data”?
• How long is my Character Variable?
• Manipulating Character Data
  • Truncation Functions
  • Concatenation Functions
  • Length Functions
  • Change Case Functions
  • Substring Functions
  • Misc. Functions
“Characters”?
What is “Character Data”? (rephrase)

“Character data consists of any combination of letters, symbols, and numeric characters.”

Quote from Microsoft documentation

<table>
<thead>
<tr>
<th>ASCII</th>
<th>One byte per letter (upper and lower case), or number, or select special characters (the ones on a standard typewriter), plus a few non-printable: Carriage Return, Line Feed, Tab, Bell, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBCDIC</td>
<td></td>
</tr>
</tbody>
</table>

What is “Character Data”?

“Character data consists of any combination of letters, symbols, and numeric characters.”

Quote from Microsoft documentation

<table>
<thead>
<tr>
<th>ASCII</th>
<th>What about alphabetic characters used in other languages? Æ Ç Ñ</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBCDIC</td>
<td>What about special punctuation and symbols? ™ ¿</td>
</tr>
<tr>
<td></td>
<td>What about curly quotes? “ ”</td>
</tr>
</tbody>
</table>
What is “Character Data”?

“Character data consists of any combination of letters, symbols, and numeric characters.”

Quote from Microsoft documentation

**Unicode**

Many more characters can be defined and stored.

BUT … it takes more than one byte of storage to do so. (Nothing is free ...)

#SASGF14
How Long is a Character Variable?

- The default length of a character is 200 characters (unless, of course, overridden).

NOT ANYMORE

The MAXIMUM length of a character variable also used to be 200 bytes – now it's 32,767 bytes.

(THAT would be an undesirable default.)
How Long is a Character Variable?

The default length of a character is now based on its first use. (Again, unless overridden.)

**Hardcoded Value**

The length of the FIRST value to which the variable is set (which may not be the longest value … well, for better or worse, it just became the longest value.)
How Long is a Character Variable?

- The default length of a character is now based on its first use. (Again, unless overridden.)

  Value built from other character variable(s)  The TOTAL length of the variable(s) used to build the new variable.

  If you want to be safe, use a LENGTH statement – at the top of your DATA step.
How Long is a Character Variable?

data temp;
length w $ 2  y $ 40.;
retain w ""
;
x = "abcdefghijklmnopqrstuvwxyz";
y = "";
z = "1234567890";
output;
Y = "----1----2----+" ||
    "----3----4----5";

a1 = substr(x,14);
a2 = substr(x,14,13);
a3 = x || y;
output;
run;
Agenda

• What is “Character Data”?  
• How long is my Character Variable?  
• Manipulating Character Data
  • Truncation Functions  
  • Concatenation Functions  
  • Length Functions  
  • Change Case Functions  
  • Substring Functions  
  • Misc. Functions
Manipulation: Truncation Functions

- LEFT()
- RIGHT()
- TRIM()
- TRIMN()
- STRIP()
- COMPRESS()
- COMPBL()
- REPEAT()
- REVERSE()
LEFT() vs. RIGHT()

- Functions designed to aligned character variables:
  - LEFT( ) Left aligns a SAS character expression
  - RIGHT( ) Right aligns a character expression
**LEFT() vs. RIGHT()**

```sas
data sample;
  set sample;
  left = '*' || left(string) || '*';
  right = '*' || right(string) || '*';
run;
```

Note alignment in each value

<table>
<thead>
<tr>
<th>Obs</th>
<th>string</th>
<th>Left</th>
<th>right</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joe Smith</td>
<td>*Joe Smith</td>
<td>* Joe Smith*</td>
</tr>
<tr>
<td>2</td>
<td>Roma Brown</td>
<td>*Roma Brown</td>
<td>* Roma Brown*</td>
</tr>
<tr>
<td>3</td>
<td>Alice Wonde</td>
<td>*Alice Wonde</td>
<td>* Alice Wonde*</td>
</tr>
<tr>
<td>4</td>
<td>Li Wang</td>
<td>*Li Wang</td>
<td>* Li Wang*</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>
Functions designed to truncate character variables:

- **TRIM( )**  
  Both remove trailing blanks.

- **TRIMN( )**  
  **DIFFERENCE:**
  If string is totally blank:
  - TRIM( ) returns single blank: Length 1
  - TRIMN( ) returns empty string: Length 0
TRIM() vs. TRIMN()

data sample;
  input string $char14. ;
datalines;
Joe Smith
  Roma Brown
  Alice Wonderland
  Li Wang
;
contains trailing blanks
contains leading blanks
contains leading and trailing blanks
contains leading, trailing blanks and multiple blanks in between
(last line contains a blank string)
TRIM() vs. TRIMN()

data sample;
   input string $char14.;
datalines;
   Joe Smith
   Roma Brown
   Alice Wonderland
   Li    Wang
run;
TRIM() vs. TRIMN()

data sample;
  input $char14.;
  datalines;
  Joe Smith
  Roma Brown
  Alice Wonderland
  Li Wang
  ;

Note difference in how blank value is handled.
Note lack of difference in how other values are handled.
Functions designed to truncate character variables:

- **TRIM( )**
- **TRIMN( )**
- **STRIP( )**
- **COMPRESS( )**

**STRIP( ) vs. TRIM( ) vs. TRIMN( )**

- Each will remove trailing blanks.
- **STRIP( )** will also remove leading blanks.
data sample;
set sample;
  strip      = '*' || strip(string)       || '*';
  trim_left  = '*' || trim(left(string))  || '*';
  trimn_left = '*' || trimn(left(string)) || '*';
run;

The SAS System
Output of STRIP, TRIM(LEFT) and TRIMN(LEFT)

<table>
<thead>
<tr>
<th>original</th>
<th>strip</th>
<th>trim_left</th>
<th>trimn_left</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Joe Smith</td>
<td><em>Joe Smith</em></td>
<td><em>Joe Smith</em></td>
<td><em>Joe Smith</em></td>
</tr>
<tr>
<td>* Roma Brown</td>
<td><em>Roma Brown</em></td>
<td><em>Roma Brown</em></td>
<td><em>Roma Brown</em></td>
</tr>
<tr>
<td>* Alice Wonde*</td>
<td><em>Alice Wonde</em></td>
<td><em>Alice Wonde</em></td>
<td><em>Alice Wonde</em></td>
</tr>
<tr>
<td>* Li Wang *</td>
<td><em>Li Wang</em></td>
<td><em>Li Wang</em></td>
<td><em>Li Wang</em></td>
</tr>
</tbody>
</table>

No difference between STRIP and TRIMN( LEFT() )
(except efficiency)
COMPRESS() vs. COMPBL() 

- Functions designed to truncate character variables:
  - TRIM( )
  - TRIMN( )
  - STRIP( )
  - COMPRESS( )

  COMPRESS removes a specified character(s) from a string. Default = “ ” if no characters specified.
  COMPBL allows compression of multiple blanks into a single blank.
COMPRESS() vs. COMPBL()
data zipcode;
input zipcode $14.;
zipcode1 = compress(zipcode);
zipcode2 = compress(zipcode,' ()?');
zipcode3 = compress(zipcode,' - ()?');
zipcode4 = compress(zipcode,'ABCD','A');
zipcode5 = compress(zipcode,'23456','k');
datalines;
  22168-  12 34
  22168- (1234?)
  MN55346 - mn44
;
1 – removes blanks
2 – remove blanks, parens, question mark
3 – remove blanks, parens, question marks, and dashes.
4 – REMOVE specified characters ABCD
5 – KEEP specified characters 23456
COMPRESS() vs. COMPBL()  

1 – removes blanks  
2 – remove blanks, parens, question mark  
3 – remove blanks, parens, question marks, and dashes.  
4 – REMOVE specified characters ABCD  
5 – KEEP specified characters 23456  

```
data zipcode;  
input zipcode $14.;  
zipcode1 = compress(zipcode);  
zipcode2 = compress(zipcode, '()?');  
zipcode3 = compress(zipcode, '-');  
zipcode4 = compress(zipcode, 'ABCD', 'A');  
zipcode5 = compress(zipcode, '23456', 'k');  
datalines;  
22168- 12 34  
22168- (1234?)  
MN55346 - mn44  
;  
The SAS System  
Listing of Zipcodes  
<table>
<thead>
<tr>
<th>zipcode</th>
<th>zipcode1</th>
<th>zipcode2</th>
<th>zipcode3</th>
<th>zipcode4</th>
<th>zipcode5</th>
</tr>
</thead>
<tbody>
<tr>
<td>22168- 12 34</td>
<td>22168-1234</td>
<td>22168-1234</td>
<td>221681234</td>
<td>22168- 12 34</td>
<td>226234</td>
</tr>
<tr>
<td>22168- (1234?)</td>
<td>22168-(1234?)</td>
<td>22168-1234</td>
<td>221681234</td>
<td>22168- (1234?)</td>
<td>226234</td>
</tr>
<tr>
<td>MN55346 - mn44</td>
<td>MN55346-mn44</td>
<td>MN55346-mn44</td>
<td>MN55346mn44</td>
<td>MN55346mn44</td>
<td>55346 - 44</td>
</tr>
</tbody>
</table>
```
Functions designed to aligned character variables:

**REVERSE( )** Reverses SAS character expression, by default the variable length is determined by the length of the first argument.

**REPEAT( )** Repeats the Character expression, by default the variable length is of 200 bytes.
REPEAT() vs. REVERSE()

data sample;
    set sample;
    backward = reverse(string);
    repeat   = repeat(string,1);
run;

The SAS System

<table>
<thead>
<tr>
<th>Obs</th>
<th>string</th>
<th>backward</th>
<th>repeat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joe</td>
<td>eoJ</td>
<td>Joe</td>
</tr>
<tr>
<td>2</td>
<td>Roma Brown</td>
<td>nworB amoR</td>
<td>Roma BrownRoma Brown</td>
</tr>
<tr>
<td>3</td>
<td>Alice</td>
<td>ecilA</td>
<td>Alice</td>
</tr>
</tbody>
</table>

Alphabetic List of Variables and Attributes

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>backward</td>
<td>Char</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>repeat</td>
<td>Char</td>
<td>200</td>
</tr>
<tr>
<td>1</td>
<td>string</td>
<td>Char</td>
<td>10</td>
</tr>
</tbody>
</table>
Agenda

• What is “Character Data”?  
• How long is my Character Variable?  
• Manipulating Character Data
  • Truncation Functions
  • Concatenation Functions
  • Length Functions
  • Change Case Functions
  • Substring Functions
  • Misc. Functions
Concatenation Functions

- CAT()
- CATT()
- CALL CATT Routine
- CATS()
- CALL CATS Routine
- CATX()
- CALL CATX Routine
Concatenation Functions

Functions designed to concatenate character variables:

- CAT( )
  - Concatenates character strings …
  - … without removing leading or trailing blanks.

- CATT( )
  - … and removes trailing blanks.

- CATS( )
  - … and removes leading and trailing blanks.

- CATX( )
  - … and removes leading and trailing blanks, and inserts separators between each string
Concatenation Functions

data sample;
  set sample;
  length cat catt cats $16 catx $20;
  text='Hello';
  cat =cat ('*',string,'*'); = ||
  catt=catt('*',string,'*'); = TRIM || or TRIMN ||
  cats=cats('*',string,'*'); = STRIP ||
  catx=catx('!',text,string); = STRIP || separator
run;
Concatenation Functions

```sas
data sample;
set sample;
length cat catt cats $16 catx $20;
text='Hello';
cat =cat ('*',string,'*');
catt=catt('*'.string.'*');
cats=cats('*',string,'*');
catx=catx('!',text,string);
run;
```

The SAS System

Output of Concatenation Functions

<table>
<thead>
<tr>
<th>cat</th>
<th>catt</th>
<th>cats</th>
<th>catx</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Joe Smith</em></td>
<td><em>Joe Smith</em></td>
<td><em>Joe Smith</em></td>
<td>Hello!Joe Smith</td>
</tr>
<tr>
<td>* Roma Brown*</td>
<td>* Roma Brown*</td>
<td><em>Roma Brown</em></td>
<td>Hello!Roma Brown</td>
</tr>
<tr>
<td>* Alice Wonde*</td>
<td>* Alice Wonde*</td>
<td><em>Alice Wonde</em></td>
<td>Hello!Alice Wonde</td>
</tr>
<tr>
<td>* Li Wang*</td>
<td>* Li Wang*</td>
<td><em>Li Wang</em></td>
<td>Hello!Li Wang</td>
</tr>
</tbody>
</table>

TRIM "or TRIMN || separator
Concatenation Functions

CAT functions are more efficient than the combination of TRIM( ) and || for concatenation.

NOTE: Default Length is 200.
CALL CATS CATT and CATX Routines

CALL CATT ROUTINE: The CALL CATT routine returns the result in the first argument, result.

CALL CATT(result <, string-1, ...string-n>);

CALL CATS ROUTINE: The CALL CATS routine returns the result in the first argument, result.

CALL CATS(result <, string-1, ...string-n>);

CALL CATX ROUTINE: The CALL CATX routine returns the result in the second argument, result.

CALL CATX(separator, result<, string-1 , ...>string-n);
CALL CATS CATT and CATX Routines

data _null_;  
    length catt $ 40 cats $ 40 catx $ 50;  
    x='Rio is t  ';  
    y=' he Olym   ';  
    z=' pic site for 2016. ';  
    separator='&';  
    call catt(catt,x,y,z);  put catt;  
    call cats(cats,x,y,z);  put cats;  
    catx=catx(separator,catx,x,y,z);  put catx;  
run;
Agenda

• What is “Character Data”?
• How long is my Character Variable?
• Manipulating Character Data
  • Truncation Functions
  • Concatenation Functions
  • Length Functions
  • Change Case Functions
  • Substring Functions
  • Misc. Functions
Manipulation : Length Functions

- LENGTH()
- LENGTHN()
- LENGTHC()
- LENGTHM()
LENGTH( ), LENGTHN( ), LENGTHC( ), LENGTHM( )

- Functions designed to report variable value length:
  - `LENGTH( )` ... excluding trailing blanks, and returns 1 for a blank string.
  - `LENGTHN( )` ... excluding trailing blanks, and returns 0 for a blank string.
  - `LENGTHC( )` ... including trailing blanks.
  - `LENGTHM( )` Returns the amount of memory (in bytes) that is allocated for a character string.
LENGTH(), LENGTHN(), LENGTHC(), LENGTHM()

data how_long;
    one = "SASGF2014 ";
    Two = " ";
    length_one = length(one);
    lengthn_one = lengthn(one);
    lengthc_one = lengthc(one);
    lengthm_one = lengthm(one);
    length_two = length(two);
    lengthn_two = lengthn(two);
    lengthc_two = lengthc(two);
    lengthm_two = lengthm(two);
run;

Nonblank value with trailing blank. Missing value (Blank character).
data how_long;
  one = 'SASGF2014 ';
  Two = ' ';
  length_one  = length(one);
  lengthn_one = lengthn(one);
  lengthc_one = lengthc(one);
run;

Nonblank value with trailing blank. Missing value (Blank character).

<table>
<thead>
<tr>
<th>one</th>
<th>two</th>
<th>length_one</th>
<th>lengthn_one</th>
<th>lengthc_one</th>
<th>lengthm_one</th>
<th>length_two</th>
<th>lengthn_two</th>
<th>lengthc_two</th>
<th>lengthm_two</th>
</tr>
</thead>
<tbody>
<tr>
<td>SASGF2014</td>
<td></td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Agenda

• What is “Character Data”?
• How long is my Character Variable?
• Manipulating Character Data
  • Truncation Functions
  • Concatenation Functions
  • Length Functions
  • Change Case Functions
  • Substring Functions
  • Misc. Functions
Manipulation : Change case Functions

- UPCASE()
- LOWCASE()
- PROPCASE()
Functions designed to change character values:

- **UPCASE( )**: Converts all uppercase letters to lowercase letters.
- **LOWCASE( )**: Converts all lowercase letters to uppercase letters.
- **PROPCASE( )**: Capitalizes the first letter of word in a string, leaving others to lower case.
Lowcase(), Upcase(), Propcase()
data ds_1;
length Name $ 80. ;
input Name $ & ;
datalines;
rosy, mike and others
ROSY, MIKE AND OTHERS
ROSY, MIKE aNd OTHERS
;
run;

data ds_2;
set ds_1;
/*convert it to other cases*/
upcase_var = upcase(Name);
propercase_var = propcase(Name);
run;

<table>
<thead>
<tr>
<th>Name</th>
<th>upcase_var</th>
<th>propercase_var</th>
<th>lowcase_var</th>
</tr>
</thead>
<tbody>
<tr>
<td>rosy, mike and others</td>
<td>ROSY, MIKE AND OTHERS</td>
<td>Rosy, Mike And Others</td>
<td>rosy, mike and others</td>
</tr>
<tr>
<td>ROSY, MIKE AND OTHERS</td>
<td>ROSY, MIKE AND OTHERS</td>
<td>Rosy, Mike And Others</td>
<td>rosy, mike and others</td>
</tr>
<tr>
<td>ROSY, Mike aNd OTHERS</td>
<td>ROSY, MIKE AND OTHERS</td>
<td>Rosy, Mike And Others</td>
<td>rosy, mike and others</td>
</tr>
</tbody>
</table>
Agenda

- What is “Character Data”? 
- How long is my Character Variable?
- Manipulating Character Data
  - Truncation Functions
  - Concatenation Functions
  - Length Functions
  - Change Case Functions
  - Substring Functions
  - Misc. Functions
Manipulation : Substring Functions

- SUBPAD Function
- SUBSTR (left of =) Function
- SUBSTR (right of =) Function
- SUBSTRN()
SUBPAD( ) – will return a variable with the length specified, padding the results with spaces.

SUBSTR(left of = ) – Replaces Character value contents

SUBSTR(right of = ) – Extracts the substring

SUBSTRN( ) – will return a substring and allows with length of zero
SUBPAD(), SUBSTR (left of =), SUBSTR (right of =), SUBSTRN()

data test;
length phone $50;
input phone $;
datalines;
  (312)555-2222
  (312)555-2121
  (312)555-3131
  (312)555-4141
run;

data test2;
set test;
substr1 = substr(phone, 7);
substr2 = substr(phone, 10, 4);
substr(phone, 2, 3) = '952';
run;
MANY functions help find a character(s) within a character string:

INDEX( )

Search a character string for the presence of a specified string, return the 1\textsuperscript{st} position of the latter within the former. (Search for “needle” in “haystack”.)

INDEXC( )

INDEX searches for entire “needle”.

INDEXC searches for any character in “needle” that exists in “haystack”.

INDEXW( )

INDEXW searches for entire “needle” BUT it must be a “word” separated by a delimiter(s).
Working with Character Data

MANY functions help find a character(s) within a character string:

INDEX( )

Search a character string for the presence of a specified string, return the 1st position of the latter within the former. (Search for “needle” in “haystack”.)

FIND( )

FIND searches for entire “needle”.

FINDC searches for any character in “needle” that exists in “haystack”.

FINDW searches for entire “needle” BUT it must be a “word” separated by a delimiter(s).
Working with Character Data

MANY functions help find a character(s) within a character string:

INDEX( )
FIND( )
VERIFY( )

**Searching**

Search a character string for the presence of a specified string, return the 1st position of the latter within the former.

Example: Search for “needle” in “haystack”. (Search for entire “needle”.)

Examples:

\[
\text{INDEX(“haystack”, “needle”)} = 0
\]

\[
\text{FIND(“haystack”, “needle”)} = 0
\]

\[
\text{VERIFY(“haystack”, “needle”)} = 0
\]

**Manipulating**

FIND searches for entire “needle”.
FINDC searches for any character in “needle” that exists in “haystack”.
FINDW searches for entire “needle” BUT it must be a “word” separated by a delimiter(s).

“STARTING POSITION” and various modifiers.

(In my opinion), FIND is easier to remember!

SO WHAT IS THE DIFFERENCE?

FIND has allows options such as

**QUICK OVERVIEW DUE TO TIME CONSTRAINTS.**

#SASGF14
Working with Character Data

MANY functions help find a character(s) within a character string:

- **INDEX( )**: Search a character string for the presence of a character NOT in a specified string, return the 1st position of the latter within the former. (Search for “hay” in “haystack”, report back when you find something that is not “hay”.)
- **FIND( )**: 
- **VERIFY( )**: 

**ONLY QUICK OVERVIEW DUE TO TIME CONSTRAINTS.**
Working with Character Data

MANY functions help find a character(s) within a character string:

- **ANYALNUM( )**
- **ANYGRAPH( )**
- **ANYPUNCT( )**
- **ANYALPHA( )**
- **ANYLOWER( )**
- **ANYSPACE( )**
- **ANYCNTRL( )**
- **ANYNAME( )**
- **ANYUPPER( )**
- **ANYDIGIT( )**
- **ANYPRINT( )**
- **ANYXDIGIT( )**
- **ANYFIRST( )**

Search a character string for `<something>`, return the 1st position of the latter within the former.
Working with Character Data

MANY functions help find a character(s) within a character string:

NOTALNUM() NOTGRAPH() NOTPUNCT()
NOTALPHA() NOTLOWER() NOTSPACE()
NOTCNTRL() NOTNAME() NOTUPPER()
NOTDIGIT() NOTPRINT() NOTXDIGIT()
NOTFIRST()

Search a character string for <something>, return the 1st position of something that IS NOT the latter within the former.
Agenda

• What is “Character Data”?  
• How long is my Character Variable?  
• **Manipulating Character Data**  
  • Truncation Functions  
  • Concatenation Functions  
  • Length Functions  
  • Change Case Functions  
  • Substring Functions  
  • Misc. Functions
Manipulation : Misc. Functions

- COUNT Function
- COUNTC Function
- IFC Function
- IFN Function
- MISSING Function
- TRANSLATE Function
- TRANWRD Function
Manipulation : Misc. Functions

**COUNT()** - counts substrings of characters in a character string.

**COUNTC()** - counts individual characters in a character string.

**IFC()** - Returns a character value of an expression based on whether the expression is true, false, or missing.

**IFN()** - Returns a numeric value of an expression based on whether the expression is true, false, or missing.
data temp;
xyz='This is a thistle? Yes, this is a thistle.';
howmanythis=count(xyz,'this');
howmanythis1=count(xyz,'this','i');
howmanyi = countc(xyz,'i');
run;

  The SAS System

  13:35 Monday, September

<table>
<thead>
<tr>
<th></th>
<th>howmanythis</th>
<th>howmanythis1</th>
<th>howmanyi</th>
</tr>
</thead>
<tbody>
<tr>
<td>xyz</td>
<td>This is a thistle? Yes, this is a thistle.</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
IFC() Vs IFN()

data grade;
input name $ grade;
performance = ifc(grade>80, 'Pass', 'Needs Improvement');
if grade>80 then perf='Pass '
else perf = 'Needs Improvement';
datalines;
John 74
Kareem 89
Kati 100
Maria 92
;
run;

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th>grade</th>
<th>performance</th>
<th>perf</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>74</td>
<td>Needs Improvement</td>
<td>Needs Improvement</td>
</tr>
<tr>
<td>2</td>
<td>Kareem</td>
<td>89</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>3</td>
<td>Kati</td>
<td>100</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>4</td>
<td>Maria</td>
<td>92</td>
<td>Pass</td>
<td>Pass</td>
</tr>
</tbody>
</table>
data sales;
input TotSales;
comm=ifn(TotSales > 10000, TotSales*.05, TotSales*.02);
if TotSales > 10000 then do comm_td = TotSales*.05;end;
else do;comm_td = TotSales*.02;end;
datalines;
25000
10000
500
10300
;
run;
MISSING() – If numeric and character expression contains missing value SAS returns ‘1’ else SAS returns ‘0’

TRANSLATE() – Replaces specific character

TRANWRD() - Replaces or removes all occurrences of a word in a character string
MISSING() Vs TRANSLATE() Vs TRANWRD()

data name;
input @1 sal $5. @6 fname $6.  
  @12 lname $20.;
datalines;
Miss  Joan  Smith  
Ms    Ann          
Miss Alice Cooper    
;  
run;

data show;
set name;
Sal_tranwrd = tranwrd(sal, "Miss", "Ms");
Sal_translate = translate(sal_tranwrd, "MS", "Ms");
if missing(lname) then comments = 'Last name is missing';
run;
MISSING() Vs TRANSLATE() Vs TRANWRD()

data name;
input @1 sal $5. @6 fname $6.
  @12 lname $20.;
datalines;
Miss  Joan  Smith
Ms   Ann
Miss Alice Cooper
;
data show;
  set name;
run;
Sal_tranwrd = tranwrd(sal, "Miss", "Ms");
Sal_translate = translate(sal_tranwrd, "MS", "Ms");
if missing(lname) then comments = 'Last name is missing';
run;
MISSING() Vs TRANSLATE() Vs TRANWRD()

data name;
input @12 sal $5.(fname $6. lname $20.);
datalines;
Miss  Joan  Smith
Ms   Ann
Miss Alice Cooper
;
run;
data show;
set name;
Sal_tranwrd = tranwrd(sal, "Miss", "Ms");
Sal_translate = translate(sal_tranwrd, "MS", "Ms");
if missing(lname) then comments = 'Last name is missing';
run;
Working with Character Data
Perl Regular Expressions

Introduced in Version 9

More complex than “traditional” SAS functions.

More robust than “traditional” SAS functions.

Example: Search for a 3-character string where the first two characters are numbers and the 3rd is a 0.

```sas
IF _N_ = 1 THEN
  PATTERN_NEEDLE = PRXPARSE( "(/d\d0/" );

PRXMATCH(PATTERN_NEEDLE, STRING_HAYSTACK )
```
Working with Character Data

Informats and Formats

$\langle nn \rangle$. vs. $\text{CHAR}\langle nn \rangle$.

PROC FORMAT

PUT( ) function and INPUT( ) function

ONLY QUICK OVERVIEW DUE TO TIME CONSTRAINTS.
Working with Character Data

Updates in recent releases:

Version 9.3: &SYSSIZEOFUNICODE.

Version 9.4: PUTC( ) and PUTN( ) now allow justification.

Version 9.4: SCAN( ) defaults resulting variable to length of 1st string processed.

Version 9.4: [NEW] TYPEOF( ) indicates whether an argument is character or numeric. (Only for WHERE clauses and Graphic Template Language [GTL].)
Working with Character Data

Updates in recent releases:

Version 9.4: [NEW] **FCOPY( )** copies a record from one FILEREF to another.

Version 9.4: [NEW] **DOSUBL( )** trades values between SAS and the CALLING ENVIRONMENT's macro variables (*not SAS macros*)

Version 9.4: [NEW] **SHA256( )** returns the result of a message digest of a specified string. (Cryptologists, take note!!)

**ONLY QUICK OVERVIEW DUE TO TIME CONSTRAINTS.**
Conclusion

This was an OVERVIEW of just 35 functions – designed to stimulate the curiosity, not to teach everything known about character data and how to work with it.
We hope it has done that – have fun as you continue to learn and to grow.

Contact:
swati_agarwal@optum.com
KuligowskiAndrew@gmail.com
Questions?