

Are Strings Tying You In Knots?

Deb Cassidy, Computer Horizons Corporation, Indianapolis, IN

ABSTRACT

If the answer to the title question is "YE", then this presentation is for you. You'll learn how to make sure you get the entire string "YES" when creating a new variable. The presentation will also walk through examples of analyzing character data where you'll need to use SAS® functions such as SCAN, TRIM, LEFT, INDEX and COMPRESS to clean up your data before you can use procedures such as PROC FREQ.

The data and report requests are fictitious but it shows the types of problems that are commonly encountered when working with character data.

Please note that throughout this paper the layout of the results have been reformatted for presentation purposes. For example, the PROC TABULATE would normally have separator lines.

PROBLEM

There are some special potential users attending SUGI 20. The marketing manager wants to make sure the right marketing reps are present and there are enough brochures reserved for these special guests. A dataset was already created with the relevant information. You need to supply the following:

- # of people in each region
- # of people interested in each SAS product
- # of people that would fall into each pricing category
- list of people in matrix format showing their interests in each product (sort by last name)
- mailing list sorted by ZIPCODE.

You look at the request and think you will only need these PROCs: FREQ, SORT, PRINT, FORMS. The first three are "basic". PROC FORMS is a little more complicated and is site-specific because the options depend on the type of labels and printer you use. Fortunately, you have access to an example for your site so you just need to make sure the data is sorted by ZIPCODE.

When you do a PROC CONTENTS and PROC PRINT you discover a few complications.

EXAMPLE DATA

PROC CONTENTS shows the variables are:

VARIABLE	TYPE	LENGTH
ADD1	Char	30
ADD2	Char	30
INTEREST	Char	60
NAME	Char	30
NUM_PC	Num	8

PROC PRINT would show the first observation is:

NAME:	Cinde Rella
ADD1:	Castle Apartments, #B13
ADD2:	Mycastle, FL 12345
INTEREST:	Base, SAS/GRAPH, SAS/AF, Frame
NUM_PC:	15

PROBLEM, PART II

Now that you've seen the data, you'll need to do several things before you can run your PROCs.

- Create a region variable from state after state is extracted from ADD2
- Separate INTEREST into variables for each product
- Create a price category variable from NUM_PC
- Separate first and last name into variables from NAME
- Separate ZIP from ADD2.

CREATING NEW VARIABLES

The easiest task is to create region from state. (In real life, you have to extract state from ADD2 first but in a tutorial the presenter gets to rearrange things.) There are several ways of doing this. One is to use IF statements with the IN operator.

```
DATA STATEREG;  
  SET ORIGDATA;  
  IF STATE IN (CA, OR, WA, ....) THEN  
    REGION='WEST';  
  ELSE IF STATE IN (FL, GA, NC,...) THEN  
    REGION='SOUTH';  
    ... (rest of the regions)...  
  ELSE REGION='INVALID';
```

To get the number of people in each region:

```
PROC FREQ DATA=STATREG;
  TABLES REGION;
```

The results are:

REGION	FREQUENCY
EAST	16
INVA	1
MIDW	10
SOUT	24
WEST	9

However, some of the regions were truncated at 4 characters. You check the manual and it says PROC FREQ will truncate at 16 characters so what happened? PROC CONTENTS will show you that REGION was created as a character variable of length 4. SAS determines the length based on the first time it sees the variable. In this example, REGION was first created with the IF statement for "WEST" which is 4 characters. One solution would be to put the longest value first. In this case "INVALID" is the longest value but it has to be last because it is for observations that are not assigned in the preceding IFs. Another choice would be to pad your first value with blanks, "WEST " so it becomes the maximum length. This works but is very susceptible to future problems if the program is edited. A safer alternative is the use of a LENGTH statement.

```
LENGTH REGION $7;
...put IF statements after length statement
```

This statement says to create a REGION variable as character (\$) with a length of 7. Note that there is no "." after the \$7 like there is with a format or informat statement. You can put several variables into one LENGTH statement. If you have more than one variable with the same length then you can group them together and the length that follows the variables will be the length that applies. For example, in the following statement both REGION and PRICECAT will be character of length 7 and COUNTER will be numeric of length 4.

```
LENGTH REGION PRICECAT $7 COUNTER 4;
```

The results when you use a LENGTH statement first are:

REGION	FREQUENCY
EAST	16
INVALID	1
MIDWEST	10
SOUTH	24
WEST	9

The next task is to create a price category variable. This time you start with a LENGTH statement.

```
DATA TWO;
  SET ORIGDATA;
  LENGTH PRICECAT $5;
  IF NUM_PC =1 THEN PRICECAT='1';
  ELSE IF 2<=NUM_PC<=5 THEN PRICECAT='2-5';
  ELSE IF 6<=NUM_PC<=10 THEN PRICECAT='6-10';
  ELSE IF 11<=NUM_PC<=98 THEN PRICECAT='11+';
  ELSE IF NUM_PC=99 THEN PRICECAT='EDUC';
  ELSE IF NUM_PC=. THEN PRICECAT='NA';
```

The results are:

PRICECAT	FREQUENCY
EDUC	6
NA	2
1	18
11+	6
2-5	29
6-10	9

Although these results are "right", the order makes them a little confusing to read. "11+" appears after "1" and before "2-5" because strings are sorted one character at a time from left to right. EBCDIC and ASCII collating sequences also sort differently. PROC FREQ has options for changing the order but not one for "how I typed them".

One alternative for changing the order of the results is to use lead blanks since blanks are the smallest printable character. When you use an EBCDIC system, there must be a blank before "11+" to have it sort before "EDUC". ASCII users would only need a single blank in the first three categories. The length also needed to be changed to 6 for EBCDIC.

```
LENGTH PRICECAT $6;
  IF NUM_PC =1 THEN PRICECAT=' 1';
  ELSE IF 2<=NUM_PC<=5 THEN PRICECAT=' 2-5';
  ELSE IF 6<=NUM_PC<=10 THEN PRICECAT=' 6-10';
  ELSE IF 11<=NUM_PC<=98 THEN PRICECAT=' 11+';
  ELSE IF NUM_PC=99 THEN PRICECAT='EDUC';
  ELSE IF NUM_PC=. THEN PRICECAT='NA';
```

The results are:

PRICECAT	FREQUENCY
1	18
2-5	29
6-10	9
11+	6
EDUC	6
NA	2

While these are in order, you might not want the lead blanks. Can you imagine how many blanks you might need if you wanted "NA" to appear before "EDUC". PROC FREQ also limits the strings to 16 characters so you might have to cut off a string to

```
STATE = P ZIPCODE = 12345
```

Why did ZIPCODE work but not STATE? Actually neither one are what you might think. The blank between the comma and the state code caused the problem. The first and fourth characters of WORD2 are both blanks and SUBSTR counted them. ZIPCODE actually starts with a blank. To solve this problem you can either change your SUBSTR functions or use another function first to eliminate any lead blanks. The second method is safer because you never know when you'll have 2 or more lead blanks.

```
WORD2 = LEFT(WORD2);
```

This code will left-justify the string and eliminate any lead blanks. You would then follow this code with the SUBSTR function statements.

Someone might ask why we didn't use a comma and blank as the delimiters in the SCAN function. ZIPCODE would then be the third word and the blanks wouldn't be a problem because delimiters do not appear in the words from SCAN. This will work for the first case but not the second.

```
Mycastle, FL 12345
Lake Buena Vista, FL 12356
```

In this second case, WORD1 would be "Lake", WORD2 would be "Buena", and WORD3 would be "VISTA".

It would be nice if you could use the same processing to extract LAST NAME from NAME but there are no commas separating words or other patterns. Actually there is one thing. The last name is after the last blank. But how do you find the last blank? There isn't a SCAN from the right function but there is a REVERSE function.

```
REVNAME=REVERSE(NAME);
```

The observations are:

NAME	REVNAME
Cinde Rella	aller edniC
MICK MOUSE	ESUOM KCIM
Pluto I. Futures	serutuF .I otulP

It is obvious that there are lead blanks since the third name is longer than the others. In this example, NAME was created with a length of 30. Any name that is shorter will have blanks added to it to make it 30. The first name is 11 characters long so it will have an extra 19 trailing blanks. These trailing blanks will become lead blanks when the REVERSE

function is used. PROC PRINT would drop off any lead blanks that are common to all the values printed on a page.

Now that you have REVERSEd the string, you can use SCAN to break REVNAME into first and last names. When there are several delimiters together in a string, they will be treated as one. If the first character is a delimiter, it will be ignored. Don't forget to actually put a real blank in for the delimiter. The length is also specified so you don't end up with the default for SCAN of 200 characters. In this case, it is the same as the original variable just in case there was anyone with a really long first name (or typographical errors).

```
LENGTH LAST FIRST $30;
LAST =SCAN (REVNAME,1,' ');
FIRST=SCAN (REVNAME,2,' ');
```

The results are:

LAST	FIRST
eller	edniC
ESUOM	KCIM
serutuF	.I otulP

Now you just REVERSE the FIRST and LAST names again and you are ready to put them together in the form LAST, FIRST. You'll need to concatenate the variables and include the comma and blank between the values of the variables. The challenge will be to find the concatenate symbol on your keyboard for your system! The concatenate symbol is "||" - that is 2 straight solid lines. Symbols may appear correct on your screen but may not be interpreted correctly. Once you find the symbols the code is:

```
LAST2 = LEFT(REVERSE(LAST));
FIRST2= REVERSE(FIRST);
NEWNAME = LAST2 || ', ' || FIRST2;
```

LAST name was LEFT-justified after REVERSE to eliminate lead blanks. You can combine several functions and they will be processed from the inside out. Just don't forget the () for each function. The results are:

Rella	, Cinde
MOUSE	, Mick
Futures	, Pluto I.

Oops! FIRST name should also have been LEFT-justified. But why the extra blanks before the comma? LAST name is still 30 characters. While the LEFT function removed lead blanks it also caused trailing blanks to be added to maintain a

add the lead blanks. Formats provide an alternative that can also be used across systems.

```
IF      NUM_PC =1 THEN PRICENUM=1;
ELSE IF 2<=NUM_PC<=5 THEN PRICENUM=2;
ELSE IF 6<=NUM_PC<=10 THEN PRICENUM=3;
ELSE IF 11<=NUM_PC<=98 THEN PRICENUM=4;
ELSE IF   NUM_PC =99 THEN PRICENUM=5;
ELSE IF   NUM_PC =. THEN PRICENUM=5;
```

```
PROC FORMAT;
  VALUE PRICE
    1='1'
    2='2-5'
    3='6-10'
    4='11+'
    5='NA'
    6='EDUC';
```

```
PROC FREQ;
  TABLES PRICENUM;
  FORMAT PRICENUM PRICE.;
```

The results are:

PRICENUM	FREQUENCY
1	18
2-5	29
6-10	9
11+	6
NA	2
EDUC	6

The order of the results is 1-6 but the value is printed with the text from the format. This puts the "EDUC" category after the "NA". If you put any lead blanks in the format, they will appear in the frequency table. You could also use this technique with the REGION variable above to make sure "INVALID" appears at the end.

EXTRACTING VALUES FROM EXISTING VARIABLES

The rest of the presentation shows many of the functions that SAS has included for working with character data. How do we get STATE and ZIPCODE from ADD2? Fortunately, when you looked at your observations you see that all ADD2 lines were entered in the form "CITY, STATE ZIPCODE". The comma and blank between CITY and STATE and the blank between STATE and ZIPCODE are important. STATE is a 2-character postal code. Some ZIPCODEs were entered as 5 characters while others are in the ZIP+4 format. This difference will not cause a problem but other differences might. If some observations had been entered in a different form, you might still be able to do the processing but it would be more complicated. There may be times when you have no choice except to edit each observation.

The first observation is:

Mycastle, FL 12345

There are actually several ways to extract STATE and ZIPCODE. The method that will be shown later for last name could also be used here. The SCAN function will break a string into "words" based on the delimiters.

```
WORD1 = SCAN(ADD2,1,',');
WORD2 = SCAN(ADD2,1',,');
```

The above code will break ADD2 into words using a comma as the delimiter. The first word will be all characters up to the first comma but not including the comma. The second word will be everything after the first comma up to the second comma. In this case, there is no second comma in ADD2 so it will be to the end of the string. If you asked for a third word, you would get a blank.

You can specify one or more delimiters or use the defaults. You must put the delimiter(s) in quotes or SAS will think you are using a variable name (, is not a valid variable name so you'll really have an error). Do not put any variable names in quotes or SAS will think you want the string "ADD2" scanned rather than the value of the variable.

For the first observation the results are:

```
WORD1 = Mycastle
WORD2 = FL 12345
```

PROC CONTENTS will reveal a characteristic of the SCAN function that often causes problems although it doesn't here. Variables created with a SCAN function will always have a length of 200 unless you specify a length before you use the function.

To separate WORD2 into STATE and ZIPCODE, we'll use the SUBSTR function which is also found in other languages. It extracts a "subset" of a string based on the positions you specify.

```
STATE =SUBSTR(WORD2,1,2);
ZIPCODE=SUBSTR(WORD2,4);
```

STATE is created by extracting 2 characters from WORD2 starting at position 1 in the string. ZIPCODE is created by starting at position 4. Since the number of characters was not specified, all characters through the end of the string will be extracted. The starting position is required but the number of characters to extract is not. The results are:

length of 30. The TRIM function will eliminate trailing blanks. You really need:

```
LAST2 = TRIM(LEFT(REVERSE(LAST)));
```

Results are:

```
Rella, Cinde
MOUSE, MICK
Futures, Pluto I.
```

To avoid using TRIM, you can put the LEFT and REVERSE in as part of the statement with concatenate. If you do this the trailing blanks are never added because you didn't create a 30-character variable. The code would be:

```
NEWNAME=LEFT(REVERSE(LAST)) || ', ' ||
REVERSE(FIRST);
```

PROC CONTENTS will show NEWNAME has a length of 62: LAST name 30 + FIRST name 30 + comma 1 + blank 1. You would probably want to specify a length of 31 for the original 30 characters plus the comma. The blank was already counted in the original 30 - you just moved it around.

Although NAME is now in the right form, it is still a little difficult to read since only some names are in all caps. This should be fixed after you REVERSE and LEFT-justify FIRST and LAST but before they are concatenated. This is done by separating the first letter from the rest of the name. The first letter will be capitalized while the rest will all be lowercase. The UPCASE function will capitalize a string. Unfortunately, there is no LOWCASE function. This is done by using TRANSLATE which will switch a character to another character. Starting at the point in the code above where FIRST and LAST were first created, the code is:

```
FIRST=LEFT(REVERSE(FIRST));
LAST =LEFT(REVERSE(LAST));
```

```
LENGTH FIRSTLTR LASTLTR $1
NEWNAME $31;
FIRSTLTR=SUBSTR(FIRST,1);
LASTRST =SUBSTR(FIRST,2);
LASTLTR =SUBSTR(LAST,1);
LASTRST =SUBSTR(LAST,2);
```

```
FIRSTLTR=UPCASE(FIRSTLTR);
LASTLTR =UPCASE(LASTLTR);
```

```
FIRSTRST=TRANSLATE(FIRSTRST,
'abcdefghijklmnopqrstuvwxyz,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ);
LASTRST =TRANSLATE(LASTRST,
'abcdefghijklmnopqrstuvwxyz,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ);
```

```
NEWNAME=LASTLTR || TRIM(LASTRST) ||
', ' || FIRSTLTR || FIRSTRST;
```

SUBSTR is used to separate the first letters from the rest of the name. A length statement is used so we don't have to use TRIM later. The first letter is always one character. TRIM will be needed for the rest of the LAST name (LASTRST) because that string will vary for each observation. We don't need to TRIM the rest of the FIRST name because these will be trailing blanks and they will be dropped because of the length restriction anyway. The TRANSLATE function may look confusing because you put the new characters first followed by the old characters. By typing one string directly under the other one you can see that 'A' becomes 'a', 'B' become 'b', etc. Results are:

```
Rella, Cinde
Mouse, Mick
Futures, Pluto I.
```

We did not account for the possibility of middle names or initials. We also didn't account for any names such as "John Claude Van Damme" which have a 2-part last name as well as a middle name. You could add code to account for these situations, live with a few mistakes or edit your final dataset. In most cases, I choose one of the latter two options because the time to write the code isn't worth the effort for a few cases. I would probably use PROC FREQ on FIRST and LAST name to see if I noticed anything strange.

Now the only thing left is to process the INTEREST variable. The request was for the number of people interested in each SAS product. PROC CONTENTS shows that INTEREST can be 60 characters but you already know that PROC FREQ will truncate values at 16 characters. PROC TABULATE solves the problem of how to get frequencies on longer character strings.

```
PROC TABULATE;
CLASS INTEREST;
TABLES INTEREST ALL, N;
```

The ALL in the TABLE statement will give an overall total. Results are:

INTEREST	N
af, fsp, base, frame	3
insight	4
not given	4
Base sas, sas/graph	11
Base SAS	9
Base SAS, sas/graph	1
Base SAS, STAT, QC, OR, IML	4
Base SAS, PH-Clin	3
Base, sas/graph	1
Base, Graph, AF, Frame,	
Spectravision	1
Base, SAS/GRAPH, SAS/AF, Frame	1

Base, SAS/AF, SAS/FSP, Frame	15
BASE	1
BASE, AF, FSP, FRAME	2
BASE, CONNECT	2
JMP	6
QC	2
ALL	70

Long values such as the one with Spectravision will wrap and the extra blank values before "Frame" also appear. However, this still isn't what you want because you want to count everyone who specified an interest in a product regardless of the other products they specified. To do this, you need a variable with only one product listed. If a person specified multiple products, you'll need an observation for each product. Just as with FIRST and LAST name, you can use SCAN to separate the string into words because each product was separated by a comma. The difference is that the new variable will have the same variable name for the different words and you will use an OUTPUT statement after you create each word.

```
PRODUCT=SCAN(INTEREST,1,','); OUTPUT;
PRODUCT=SCAN(INTEREST,2,','); OUTPUT;
```

...continue until you have accounted for the maximum number of words...

How do you know the maximum number of words? Won't people who have fewer products listed have some blank values? Sometimes you can tell the maximum number of words by looking at the data. A more accurate way is to count the number of commas. Once again there is no function to do this but you can do it indirectly.

The LENGTH function is used to determine the actual length of a value excluding any trailing blanks. Don't confuse the function with the LENGTH statement which sets the length for the variable for all observations. The COMPRESS function is used to eliminate certain characters from a string. While TRANSLATE changed the character, COMPRESS completely removes the character and shortens the string. To count the occurrences of a character, find the length of the original value and subtract the length after you've eliminated that character. In this case, the number of products will be 1 more than the number of commas because the last product is not followed by a comma.

```
PROD_CNT=LENGTH(INTEREST) -
          LENGTH(COMPRESS(INTEREST, ','))
          + 1;
```

You will now have an accurate product count for each observation. It is also easier to use a DO LOOP to write your statements.

```
DO I = 1 TO PROD_CNT;
  PRODUCT=SCAN(INTEREST,I,',');
  OUTPUT;
END;
```

This saves you from guessing how many words there are in the string. It also avoids outputting blank values because someone listed fewer products or missing products for someone with a lot of products listed. Partial results are:

PRODUCT	N
Frame	1
sas/graph	2
AF	1
Frame	1
Graph	1
Spectravision	1
SAS/AF	1
SAS/GRAPH	1
Base	3
Base SAS	1
BASE	1

You'll notice that you also need to LEFT-justify and UPCASE PRODUCT but there is still a consistency problem. People referred to the same product in different ways such as "AF" and "SAS/AF". You'll need to solve this problem by using IF statements or FORMATS as shown earlier to create the price category data. Don't forget the quotes and to specify a character format by using \$ as the first character of the format name.

```
IF PRODUCT IN ('SAS/AF', 'AF')
  THEN PROD='SAS/AF';
ELSE IF ...;

PROC FORMAT;
  VALUE $PROD
    'SAS/AF'='SAS/AF'
    'AF'    ='SAS/AF'
    ...;
```

If you opt to use FORMATS, values with the same format text will be grouped together. They will appear in the frequency table in the order where the first value would be and not where the formatted value would be unless you change the options.

The last task is to create separate variables for each product. An 'X' will indicate the person has an interest in that product. Although SCAN could be used another function is easier. INDEX will return the position where the specified character string was found in the original string. If the string is not found, a 0 is returned. Before you start the check,

UPCASE INTEREST so you don't have to worry about case differences. The code is:

```
INTEREST =UPCASE(INTEREST);  
  
LENGTH BASE SASAF GRAPH ... $1;  
IF INDEX(INTEREST,'BASE') >0 THEN BASE ='X';  
IF INDEX(INTEREST,'AF' ) >0 THEN SASAF='X';  
IF INDEX(INTEREST,'GRAPH')>0 THEN GRAPH='X';  
... rest of products ...
```

If you did the earlier count on PRODUCT you will know which products to list or you may need to review your data. The new variables were created with a length of 1 since you wanted the letter 'X'. Don't use ELSE statements here because each of the IF conditions are not mutually exclusive. If you are checking for strings that might be within other strings, you will have to write some additional code. For example, if your products were 'AF' and 'GRAF', you would incorrectly include 'GRAF' in the 'AF' group.

OTHER FUNCTIONS

There are four other character functions that were not shown here. RIGHT is similar to LEFT in that it justifies your values. INDEXC is similar to INDEX except it looks for any character you list instead of the entire string. For example, it will look for the characters 'S' or 'A' instead of the entire string 'SAS'.

VERIFY returns the position of a character that is in the original string but not in your specified string. If you thought ZIPCODE had an invalid value you could use:

```
CHECK=VERIFY(ZIPCODE,'0123456789-');
```

If ZIPCODE had any letters or other symbols in it then CHECK would be assigned the position number of that character.

REPEAT simply repeats a string the specified number of times. Beware of lead and trailing blanks if you use it.

Although SUBSTR was shown here several times, it can also be used to change a part of a string just like TRANSLATE will change individual characters. For example:

```
X='XXSUG';  
SUBSTR(X,1,2)='MW';
```

When you do a PROC PRINT you'll find XXSUG has become MWSUG!

CONCLUSION

There is a lot of information here but you should now be able to handle almost any character data you get. I said almost because I didn't mention non-printable characters and fields longer than 200 characters and a few other problems. When you work with character data, you should remember to:

- Use PROC CONTENTS to verify lengths
- Specify lengths
- Look for patterns to use to break up strings
- Watch out for leadding and trailing blanks
- Consider formats if order is important.

SAS and the other products are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.

Contact:

(Home)

Deb Cassidy

3809 Secretariat Court

Columbus, OH 43221

(614) 771-7861 (answering machine)

(Work)

Computer Horizons Corp.

1980 E. 116th Street, Suite 325

Carmel, IN 46032

(317) 574-5600

FINAL CODE AND RESULTS

This is one possibility for a real program. You could do several of the steps in more than one way. Unfortunately, I did it one way and then changed the code while I was writing the paper so there is an error in the text. I tried to write the paper with the easiest coding first and I wasn't looking at all my examples. First name can not be obtained from the name using SCAN. It works except for Pluto I. Futures will have his middle initial appear. The code I originally used is shown below. My apologies for any confusion this may have caused.

```
DATA FINAL;
  SET ORIG;
  LENGTH LAST FIRST $30
    FIRSTLTR LASTLTR $1
    NEWNAME $31 REGION $7
    BASE GRAPH SASAF FRAME SPECTRA $1;
  REVNAME=LEFT(REVERSE(NAME));
  LAST =SCAN(REVNAME,1,' ');
  FIRSTSP=INDEX(REVNAME,' ');
  FIRST=SUBSTR(REVNAME,FIRSTSP);

  LAST2 =LEFT(REVERSE(LAST));
  FIRST2=LEFT(REVERSE(FIRST));

  FIRSTLTR=UPCASE(SUBSTR(FIRST2,1,1));
  LASTLTR =UPCASE(SUBSTR(LAST2,1,1));
  FIRSTRST=SUBSTR(FIRST2,2);
  LASTRST =SUBSTR(LAST2,2);

  FIRSTRST=TRANSLATE(FIRSTRST,
    'abcdefghijklmnopqrstuvwxyz',
    'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
  LASTRST =TRANSLATE(LASTRST,
    'abcdefghijklmnopqrstuvwxyz',
    'ABCDEFGHIJKLMNOPQRSTUVWXYZ');

  NEWNAME=LASTLTR || TRIM(LASTRST) || ', ' ||
    FIRSTLTR || FIRSTRST;

  STZIP=SCAN(ADD2,2,',');
  STATE =SUBSTR(LEFT(STZIP),1,2);
  ZIPCODE=SUBSTR(LEFT(STZIP),4);
  IF STATE IN ('FL') THEN REGION='SOUTH';
  ELSE IF STATE IN ('CA') THEN REGION='WEST';
  * REST OF THE STATES WOULD ALSO BE HERE;

  INTEREST=UPCASE(INTEREST);
  IF INDEX(INTEREST,'BASE') > 0 THEN BASE  ='X';
  IF INDEX(INTEREST,'GRAPH') > 0 THEN GRAPH ='X';
  IF INDEX(INTEREST,'AF') > 0 THEN SASAF  ='X';
  IF INDEX(INTEREST,'FRAME') > 0 THEN FRAME ='X';
  IF INDEX(INTEREST,'SPECT') > 0 THEN SPECTRA='X';

  IF NUM_PC =1 THEN PRICENUM=1;
  ELSE IF 2<=NUM_PC<=5 THEN PRICENUM=2;
  ELSE IF 6<=NUM_PC<=10 THEN PRICENUM=3;
  ELSE IF 11<=NUM_PC<=98 THEN PRICENUM=4;
  ELSE IF NUM_PC =99 THEN PRICENUM=6;
  ELSE IF NUM_PC =. THEN PRICENUM=5;

PROC SORT DATA=FINAL;
  BY NEWNAME;

PROC PRINT DATA=FINAL SPLIT='*';
  VAR NEWNAME BASE GRAPH SASAF FRAME SPECTRA;
  LABEL NEWNAME='NAME';
  TITLE 'INTERESTS OF POTENTIAL USERS';

PROC FREQ DATA=FINAL;
  TABLES REGION / NOCUM NOPCT;
  TITLE 'REGIONS FOR POTENTIAL USERS'
```

```

PROC SORT DATA=FINAL;
  BY ZIPCODE;
  **** PUT IT PROC FORMS CODE FOR MAILING LABELS;

PROC FORMAT;
  VALUE PRICE
    1='1'
    2='2-5'
    3='6-10'
    4='11+'
    5='NA'
    6='EDUC';

PROC FREQ DATA=FINAL;
  TABLES PRICENUM /NOCUM NOPCT;
  FORMAT PRICENUM PRICE.;
  TITLE '# POTENTIAL USERS IN EACH PRICE CATEGORY';

DATA PRODUCTS;
  SET ORIG (KEEP=INTEREST);
  INTEREST=UPCASE(INTEREST);
  NUMCOMMA=LENGTH(INTEREST) - LENGTH(COMPRESS(INTEREST,',')) +1;
  DO I=1 TO NUMCOMMA;
    PRODUCT=SCAN(INTEREST,I,',' );
    OUTPUT;
  END;

DATA PRODUCTS; SET PRODUCTS;
  IF PRODUCT IN ('SAS/GRAPH','GRAPH') THEN PRODUCT='SAS/GRAPH';
  ELSE IF PRODUCT IN ('SAS/AF','AF') THEN PRODUCT='SAS/AF';
  ELSE IF PRODUCT IN ('BASE','BASE SAS') THEN PRODUCT='BASE SAS';
PROC TABULATE DATA=PRODUCTS;
  CLASS PRODUCT;
  TABLES PRODUCT ALL, N;
  TITLE 'NUMBER POTENTIAL USERS INTERESTED IN EACH PRODUCT';

```

RESULTS

FIRST 5 OBSERVATIONS

OBS	NAME	ADD1	ADD2
1	Cinde Rella	Castle Apartments, #B13	Mycastle, FL 12354
2	MICK MOUSE	123 Steamboat Way	DisneyWorld, FL 14235
3	Pluto I. Futures	1999 Planet Trail	Epcot, FL 99999-9999
4	DON DUCK	14 Quack Road	Disneyland, CA 98765
5	Abe Lincoln	16 Presidential Hall	History, FL 00001

OBS	NUM_PC	INTEREST
1	15	Base, SAS/GRAPH, SAS/AF, Frame
2	6	Base, sas/graph
3	.	Base, Graph, AF, Frame, Spectravision
4	7	Base SAS, SAS/GRAPH
5	1	Base

INTERESTS OF POTENTIAL USERS

OBS	NEWNAME	BASE	GRAPH	SASAF	FRAME	SPECTRA
1	Duck, Don	X	X			
2	Futures, Pluto i.	X	X	X	X	X
3	Lincoln, Abe	X				
4	Mouse, Mick	X	X			
5	Rella, Cinde	X	X	X	X	

REGIONS FOR POTENTIAL USERS

REGION	Frequency
SOUTH	4
WEST	1

POTENTIAL USERS IN EACH PRICE CATEGORY

PRICENUM	Frequency
1	1
6-10	2
11+	1
NA	1

NUMBER POTENTIAL USERS INTERESTED IN EACH PRODUCT

	N
PRODUCT	
FRAME	1.00
AF	1.00
FRAME	1.00
GRAPH	1.00
SAS/AF	1.00
SAS/GRAPH	3.00
SPECTRAVISION	1.00
BASE SAS	5.00
ALL	14.00