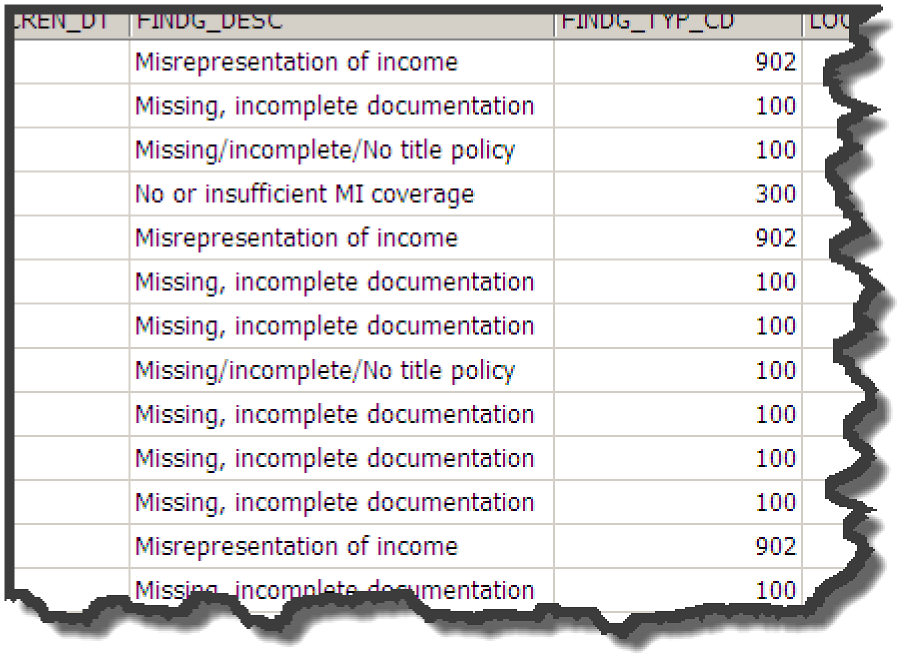
Practical Uses of Arrays and String Searches

ABSTRACT

As a Base SAS Developer, we find ourselves seeking ways of applying SAS to real world scenarios. Gathering transactional data and processing it in search of specific scenarios and trends is one such business case that is routinely required. This paper demonstrates a technique leveraging Proc Transpose, Data Step Arrays, and String concatenation to capture transactional data in search of specific criteria. Identifying transactions that meet business conditions or trends will most likely be an ongoing need.

INTRODUCTION

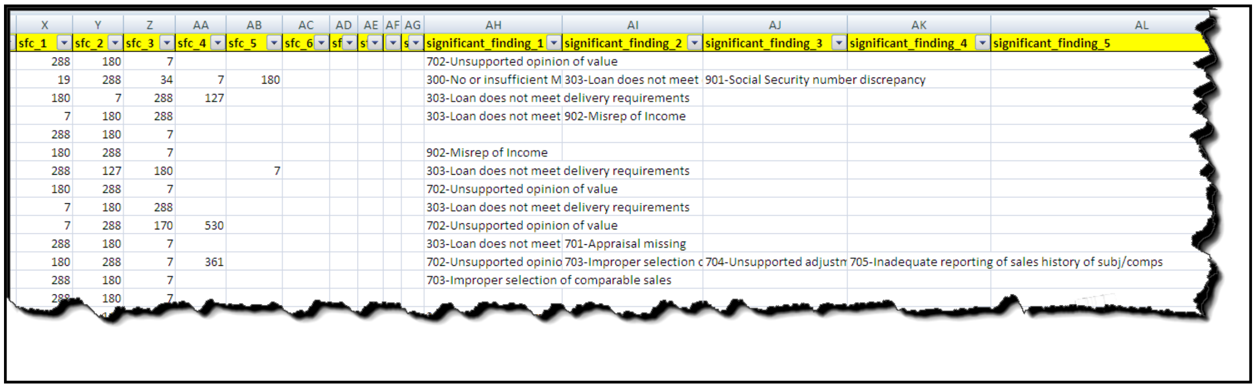
In this example, we will take transactional data and search for specific conditions. We will leverage the use of Proc Transpose and Data Step arrays to gain the desired results. Below is a sample of transactional data.



REN_DT	FINDG_DESC	FINDG_TYP_CD	LO
	Misrepresentation of income	902	
	Missing, incomplete documentation	100	
	Missing/incomplete/No title policy	100	
	No or insufficient MI coverage	300	
	Misrepresentation of income	902	
	Missing, incomplete documentation	100	
	Missing, incomplete documentation	100	
	Missing/incomplete/No title policy	100	
	Missing, incomplete documentation	100	
	Missing, incomplete documentation	100	
	Missing, incomplete documentation	100	
	Misrepresentation of income	902	
	Missing, incomplete documentation	100	

PROC TRANSPOSE – ONE ROW PER ACCOUNT

Here we want to restructure the transactions so that we have one row per account, with all the transactions as columns. This will allow us to examine all the transactions in one observation. One benefit is that we can traverse backwards or forwards thru the columns and also check for multiple conditions while we are there. Also, we can retain information from the transactions while examining all the columns on the same observation. Here is a sample visual of what we will see after transposing the transactional data.



X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL
sfc_1	sfc_2	sfc_3	sfc_4	sfc_5	sfc_6	sf	s		s	significant_finding_1	significant_finding_2	significant_finding_3	significant_finding_4	significant_finding_5
288	180	7								702-Unsupported opinion of value				
19	288	34	7	180						300-No or insufficient M	303-Loan does not meet	901-Social Security number discrepancy		
180	7	288	127							303-Loan does not meet delivery requirements				
7	180	288								303-Loan does not meet	902-Misrep of Income			
288	180	7												
180	288	7								902-Misrep of Income				
288	127	180		7						303-Loan does not meet delivery requirements				
180	288	7								702-Unsupported opinion of value				
7	180	288								303-Loan does not meet delivery requirements				
7	288	170	530							702-Unsupported opinion of value				
288	180	7								303-Loan does not meet	701-Appraisal missing			
180	288	7	361							702-Unsupported opinio	703-Improper selection c	704-Unsupported adjustn	705-Inadequate reporting of sales history of subj/comps	
288	180	7								703-Improper selection of comparable sales				
288	180	7												

Now let’s look at the code:

```
proc transpose data=sig_findings_outb
    out=sig_findings_tran prefix=significant_finding_;
       var FINDG_DESC_OUT ;
       by ln_id ln_revw_oid findings_count findg_category_both;
run;
```

DATA STEP ARRAY – COMPLEMENTS PROC TRANSPOSE

Working with the transposed columns is easy with the use of a data step array. SAS will auto load the columns for you with the given syntax:

```
data _null_ ;
set all_findings_tran;
array findg_array {*} significant_finding_: ;
…
```

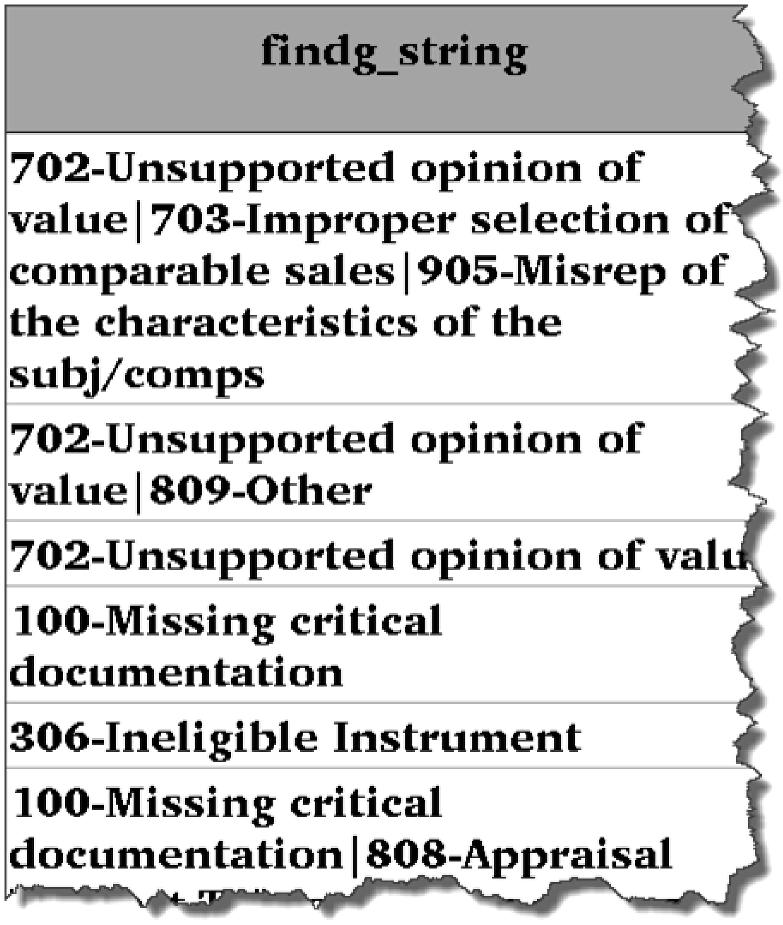
DATA STEP STRING – CONCATENATING STRINGS FROM MULTIPLE COLUMNS

Another way of utilizing the proc transpose results is to concatenate the columns into one string and then use the scan or find function to identify specific conditions. Notice the use of the *"of"* option in the catx function. Very nice ☺.

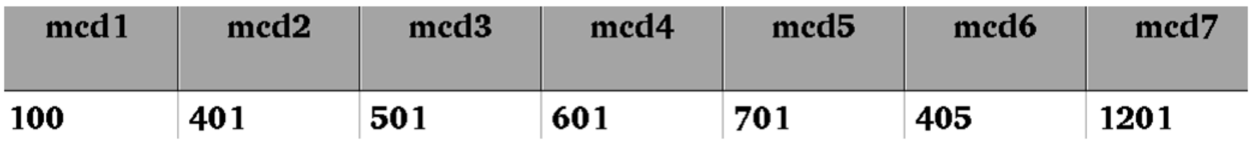
```
format findg_string $2000.
findg_string = catx("|",of significant_finding_:);
findg_string = tranwrd(findg_string,'.','');
```

LEVERAGING RESULTS

Let’s take advantage of the SAS array and string functions to find the results. The use case here is to find the accounts where multiple transaction types have occurred. Some sample data as a visual aid:



findg_string
702-Unsupported opinion of value|703-Improper selection of comparable sales|905-Misrep of the characteristics of the subj/comps
702-Unsupported opinion of value|809-Other
702-Unsupported opinion of valu
100-Missing critical documentation
306-Ineligible Instrument
100-Missing critical documentation|808-Appraisal



mcd1	mcd2	mcd3	mcd4	mcd5	mcd6	mcd7
100	401	501	601	701	405	1201

Now let's take a look at the code that produces the results. We are looking for any transactions in the array that are found in the string that we concatenated from the columns:

```
array mcd {7} $ ('100' '401' '501' '601' '701' '405' '1201') ;

…

do i = 1 to dim(mcd);
   if find(findg_string,compress(mcd(i)),'t') > 0 then do;
     mcd_flag = 'Y';  <<<<< when the desired business case is met, in this case we flag it
     leave;
   end;
 end;
```

ANOTHER USE CASE

What if you need to spool out the results of the proc transpose and don't know the maximum number of transposed variables (all of your column names)? This will be a dynamic value each execution of your proc transpose. We can accomplish this easily by using the DIM() function for arrays.

```
 data _null_ ;
 set all_findings_tran;
 array findg_array {*} significant_finding_: ;
 format max_array z2.;
 max_array = dim(findg_array);
 call symputx('max_findg',max_array);
 if _n_ = 1 then stop;
run;
```

CONCLUSION

The combination of proc transpose, data step arrays, and string functions can help identify many conditions and business scenarios for transactional data. This helps us see trends and outliers in data and answers the question: When and how often do combinations of events occur?

CONTACT INFO

Charles Basham
Business Analyst
Fannie Mae
chazbash@suddenlink.net