Application of the “INTO:” Host-Variable

Shara Auty

Problem

Insurance data contains multiple rows of transactions per policy holder.

It is necessary to summarize the data or to view the data in a different manner.

I needed to summarize by the various entries in a single column to understand non-renewals.

Data

The column I was interested in understanding was non-renew reason.

Why do policies not automatically renew?

- Underwriter set policy to non-renew
- Claims
- Ineligible for auto-renew

Data

- I had 274 non-renew reasons that could show up on a policy.
- A single policy could have multiple non-renewal reasons listed.
- I needed to summarize this information.

“INTO:” to the rescue!

INTO: creates one or more macro variables, based on the results of a SELECT statement.

INTO: is a valuable resource for creating a macro variable made up of values.

Overcomes several limitations in hard coding values, typos, resource constraints, etc…

“INTO:”

- Basic form of the “INTO:” Host-variable

```
SELECT <DISTINCT> object-item <,object-
item>…
  <INTO macro-variable-specification <, macro-variable-
    specification> …>
FROM from-list …;
```

“INTO:”

- INTO: can be used to generate a list of values.

- The list can be modified with modifiers:
  - ‘SEPARATED BY’
  - ‘QUOTE’
  - ‘NOTRIM’

Back to my problem

- I want to summarize the non-renew reasons by policy number by summing the total number of non-renew reasons listed on the policy.
- Issues to be aware of:
  - Not all policies will have the same list of non-renew reasons
  - Data source can be very large

Dynamic Solution

- My program must:
  - Account for a dynamic data source
  - Require minimal maintenance
- A program incorporating “INTO:” host-variable is just what I needed.
- Example I’ll use is from health care.

Step 1

- Read in the data
- Summarize using PROC MEANS
- Output to a new dataset
- Establish a variable which is the number of service visits, based on frequency.

```
DATA MASTER;
INPUT PAT_ID $
TG_GRP $
SVC_DT MMDDYY10.
PAID_AMT
;

DATALINES;
P1 TG01 01/21/1999 66
P1 TG12 02/10/1999 11
P1 TG03 03/16/1999 46
P1 TG15 03/16/1999 46
P1 TG04 05/09/1999 99
P1 TG04 05/10/1999 92
P1 TG04 05/15/1999 96
P1 TG03 01/25/1999 56
P1 TG12 11/29/1999 35
P1 TG04 01/12/1999 27
```

```
PROC SORT; BY TG_GRP;
```

```
PROC MEANS SUM NOPRINT NWAY DATA=MASTER;
CLASS PAT_ID TG_GRP;
VAR PAID_AMT;
OUTPUT OUT=TG_SUM(RENAME=(_FREQ_=VISIT) DROP=_TYPE_)
SUM=TOT_AMT;
RUN;
```

```
PROC PRINT DATA=TG_SUM;
TITLE 'TREATMENT GROUP SUMMARY BY PATIENT';
RUN;
```

Step 2

- Use INTO: host-variable in PROC SQL to generate a unique list of treatment groups separated by a space.

```
PROC SQL NOPRINT;
SELECT DISTINCT TG_GRP
INTO: TGLIST SEPARATED BY ' '
FROM MASTER;

%PUT &TGLIST;
```

```
64          %PUT &TGLIST;
TG01 TG03 TG04 TG11 TG12 TG15 TG18 TG46 TG74 TG88 TG99
```

- The list is made up of only those treatment groups present in the data and is stored in a macro variable.

Step 3

- Take the list of all available treatment groups and convert them into variables using the array feature.
- Assign the newly formed variables a ‘0’ using a DO LOOP.
- The loop relies on the DIM function which tracks the number of newly formed variables.

```
DATA NEWDATA (DROP=I);
SET TG_SUM;

ARRAY TEMP{*} &TGLIST;

DO I=1 TO DIM(TEMP);
TEMP{I}=0;
END;

PROC PRINT DATA=NEWDATA;
TITLE 'ESTABLISH TREATMENT GROUPS WITH VALUES SET TO ZERO';
RUN;
```

Step 4

- Tag the newly formed variables if the corresponding treatment group is present.
- Use the SAS CEIL function to scan the macro variable and populates the variables with a ‘1’.

```
DATA FIXDATA;
SET NEWDATA;

ARRAY TEMP{*} &TGLIST;

TEMP(CEIL(INDEX("&TGLIST",TRIM(TG_GRP))/(LENGTH(TG_GRP)+1))) = 1;

PROC PRINT DATA=FIXDATA;
TITLE 'TREATMENT GROUPS ASSIGNED 1 IF CONDITION IS PRESENT';
RUN;
```

Step 4 Let’s break it down…

- **TRIM(TG_GRP)** : trims any trailing spaces in TR_GRP.
- **INDEX("&TGLIST",TRIM(TG_GRP)**: searches &TGLIST macro and returns the column location of the trimmed value of TR_GRP.
- **LENGTH(TG_GRP)+1**: returns the number of characters in TR_GRP plus one for the space separating each TR_GRP (total length).
- **CEIL(…)**: returns the smallest integer value from the division of the index value and the total length.

Step 5

- Perform a patient level summary of the data using PROC MEANS.

```
PROC MEANS SUM NOPRINT NWAY DATA=FIXDATA;
CLASS PAT_ID ;
VAR VISIT TOT_AMT &TGLIST;
OUTPUT OUT=PAT_SUM(DROP= _FREQ_ _TYPE_) SUM=;
RUN;

PROC PRINT DATA=PAT_SUM;
TITLE 'PATIENT SUMMARY REPORT';
RUN;
```

Output Data

TREATMENT GROUP SUMMARY BY PATIENT

Obs	PAT_ID	TG_GRP	VISIT	TOT_AMT
1	P1	TG01	1	66
2	P1	TG03	2	102
3	P1	TG04	4	314
4	P1	TG12	3	78
5	P1	TG15	1	46
6	P1	TG18	2	95
7	P1	TG99	1	12
8	P2	TG04	1	61
9	P2	TG11	2	79
10	P2	TG12	1	61
11	P2	TG46	1	61
12	P2	TG74	1	61
13	P2	TG88		

ESTABLISH TREATMENT GROUPS WITH VALUES SET TO ZERO

Obs	PAT_ID	TG_GRP	VISIT	TOT_AMT	TG01	TG03	TG04	TG11	TG12	TG15	TG18	TG46	TG74	TG88	TG99
1	P1	TG01	1	66	0	0	0	0	0	0	0	0	0	0	0
2	P1	TG03	2	102	0	0	0	0	0	0	0	0	0	0	0
3	P1	TG04	4	314	0	0	0	0	0	0	0	0	0	0	0
4	P1	TG12	3	78	0	0	0	0	0	0	0	0	0	0	0
5	P1	TG15	1	46	0	0	0	0	0	0	0	0	0	0	0
6	P1	TG18	2	95	0	0	0	0	0	0	0	0	0	0	0
7	P1	TG99	1	12	0	0	0	0	0	0	0	0	0	0	0
8	P2	TG04	1	61	0	0	0	0	0	0	0	0	0	0	0
9	P2	TG11	2	79	0	0	0	0	0	0	0	0	0	0	0
10	P2	TG12	1	61	0	0	0	0	0	0	0	0	0	0	0
11	P2	TG46	1	61	0	0	0	0	0	0	0	0	0	0	0
12	P2	TG74	1	61	0	0	0	0	0	0	0	0	0	0	0
13	P2	TG88	2	107	0	0	0	0	0	0	0	0	0	0	0

Output Data

TREATMENT GROUPS ASSIGNED 1 IF CONDITION IS PRESENT

Obs	PAT_ID	TG_GRP	VISIT	TOT_AMT	TG01	TG03	TG04	TG11	TG12	TG15	TG18	TG46	TG74	TG88	TG99
1	P1	TG01	1	66	1	0	0	0	0	0	0	0	0	0	0
2	P1	TG03	2	102	0	1	0	0	0	0	0	0	0	0	0
3	P1	TG04	4	314	0	0	1	0	0	0	0	0	0	0	0
4	P1	TG12	3	78	0	0	0	0	1	0	0	0	0	0	0
5	P1	TG15	1	46	0	0	0	0	0	1	0	0	0	0	0
6	P1	TG18	2	95	0	0	0	0	0	0	1	0	0	0	0
7	P1	TG99	1	12	0	0	0	0	0	0	0	0	0	0	1
8	P2	TG04	1	61	0	0	1	0	0	0	0	0	0	0	0
9	P2	TG11	2	79	0	0	0	1	0	0	0	0	0	0	0
10	P2	TG12	1	61	0	0	0	0	1	0	0	0	0	0	0
11	P2	TG46	1	61	0	0	0	0	0	0	0	1	0	0	0
12	P2	TG74	1	61	0	0	0	0	0	0	0	0	1	0	0
13	P2	TG88	2	107	0	0	0	0	0	0	0	0	0	1	0

PATIENT SUMMARY REPORT

Obs	PAT_ID	VISIT	TOT_AMT	TG01	TG03	TG04	TG11	TG12	TG15	TG18	TG46	TG74	TG88	TG99
1	P1	14	713	1	1	1	0	1	1	1	0	0	0	1
2	P2	8	430	0	0	1	1	1	0	0	1	1	1	0

References

This discussion was based on two papers

Using the Magical Keyword "INTO:" in PROC SQL
by Thiru Satchi

Getting More Out of "INTO" in PROC SQL: An Example for Creating Macro Variables by Mary-Elizabeth Eddlestone