104-28

RETAINing Information to Identify Entity Characteristics

John D. Chapman, Ph.D., Markcelian Analytics, Inc., Cohasset, MA

ABSTRACT

A common requirement of data analysis is to identify an entity characteristic that is not explicit in the data but is derivable from it. The SAS RETAIN statement is a powerful and flexible tool for analysts and programmers with such a requirement. This paper introduces the RETAIN statement, along with its cousin the SUM statement, and illustrates its use for this class of analytic requirements. The concepts discussed apply to Base SAS® and will be of most interest to beginning and intermediate users with an interest in using the data step to derive information from data.

INTRODUCTION TO THE TASK OF IDENTIFYING ENTITY CHARACTERISTICS

Data commonly contains information about entities, such as customers, students, patients or stores. Some elements of data explicitly describe entity characteristics, such as the location of stores or the gender of students. Other characteristics may be derivable from the elements on a single record, e.g., the age of students derived from date of birth and school year. Frequently, however, information about entity characteristics is not explicit and is spread over many data records; the data must be manipulated in order to express the information in a form that is usable for analysis or reporting. For example, student records that list all courses taken and grades received do not explicitly identify the students who have never received an 'A', but taken together they do contain that information.

OVERVIEW OF RETAIN

Extracting entity characteristics from multiple data records requires consolidating and/or comparing data elements across records. The SAS RETAIN statement enables such actions.

By default data elements created in a data step are set to missing at the beginning of each DATA step iteration. The RETAIN statement is used to modify this behavior so that a value changes only at the behest of the programmer. It may also be used to assign an initial value to a data element. For example, the statement:

```
RETAIN sample 0;
```

will cause the creation of a data element named SAMPLE with an initial value of 0. The value of SAMPLE will continue to be 0 until it is explicitly changed, regardless of how many times the data step iterates.

RETAIN may also be invoked with the SUM statement, as in :

```
sample + <expression> ;
```

which is equivalent to:

```
RETAIN sample 0;
sample = sample + <expression> ;
```

RETAINed data elements can be used to carry information across iterations of the DATA step. In conjunction with other techniques such as sorting and using the FIRST and LAST automatic variables, the RETAIN statement enables the programmer to implement logic that requires information from multiple data records. Identifying entity characteristics is a valuable application of this capability.

SOME TYPICAL CASES

The following three case studies demonstrate the application of RETAIN to the task of identifying entity characteristics.

CASE #1

A simple case is when a single data element has all the information required about the desired characteristic. Consider, for example a dataset containing student grades for a term, which is illustrated by:

```
data Grade_Book ;
  infile datalines ;
  input Name $ Class $ Grade $;
datalines;
Alice     Math      A
Alice     English   B
Alice     Physics   A
Alice     History   B
Alice     French    C
Alice     Art       A
Joe       Math      B
Joe       English   C
Joe       Physics   C
Joe       History   B
Joe       French    B
Joe       Art       B
Sue       Math      A
Sue       English   A
Sue       Physics   B
Sue       History   A
Sue       French    B
Sue       Art       A
run;
```

The requirement is to identify students who qualify for the honor roll, which requires that they have received at least three A grades and no grade of C or lower. The data element GRADE in the GRADE_BOOK dataset has the required information. The following data step illustrates how information about grades is retained from record to record, in this case using the SUM statement, and then evaluated to determine which student names should be put in the HONOR_ROLL dataset.

```
     data Honor_Roll (keep = name); ;            5
       set Grade_Book ;
1      by name ;
3      if first.name then do;
          Num_A = 0;                           2
          Num_C = 0;
         end;
       if Grade = 'A' then Num_A + 1 ;
       if Grade >= 'C' then Num_C + 1 ;
4      if last.name and Num_A >=3 and Num_C = 0
     then output;
     run;
```

This data step has several elements that are typical of the technique:

1. The source dataset is SET by NAME, the variable that identifies students, which is the entity of interest. This of course requires that it be sorted or indexed by NAME.
2. There are one or more counters, NUM_A and NUM_C

in this case, which are RETAINed across iterations. Here they are coded using the SUM statement though RETAIN could be used explicitly as well.

3. The FIRST automatic variable is used to reset the counter(s) for each student.
4. The LAST automatic variable is used to control evaluation of the counter(s) and assignment of qualifying students to the honor roll. When the last record for Alice is evaluated NUM_A = 3 and NUM_C = 1. Alice's C in French disqualifies her so no record is output. When Joe's last record is encountered NUM_A = 2 so he does not qualify and GRADE_BOOK is still empty. Finally, at Sue's last record the criteria are met and Sue's name is written to GRADE_BOOK.
5. The result is a dataset with one name, Sue.

CASE #2

The objective in the second case is to identify 'loyal' customers in a set of customer purchase data. Loyal customers are defined as those who have made at least three purchases in any given 90 day period. The data provides a record for each purchase, the attributes of which are the date and amount of the purchase.

```
data PurchaseHistory ;
  format Date date9. Purchase dollar6.2;
  informat Date Date9. ;
  infile datalines ;
  input Customer_ID $  Date Purchase;
datalines;
1      27OCT2002      40
1      15DEC2002      25
1       9JAN2003      24
1      30JAN2003      50
2      15NOV2002      30
2      27FEB2003      40
2      27MAR2003      40
run;
```

The illustration is simplified by including only customers with at least three purchases and considering only the period beginning with a customer's first purchase and ending with his/her last purchase.

This data has two sources of information that must be considered to determine customer loyalty. The existence of a purchase, indicated by the presence of a record, is one source of relevant information, and the difference between dates on successive records is another. The following data step uses RETAINed variables to identify the loyal customers.

```
      data Loyal ;
        set PurchaseHistory ;
        by Customer_ID Date ;          1
        retain QualDate1 QualDate2 ;
 2      if first.Customer_ID then do ;
             QualDate1 = Date ;
             QualDate2 = . ;
             IsLoyal = 0 ;
          end;
        else if QualDate2 = . then do ;
            QualDate2 = Date ;
          end;
        else do ;
            if Date - QualDate1 > 90
 3             then IsLoyal + 1;
             QualDate1 = QualDate2 ;
             QualDate2 = Date ;
          end;
        if last.Customer_ID and IsLoyal = 0
           then output;
      run;
```

Features which distinguish this case from the previous one are:

1. The source data is processed by the entity to be described, as in case #1, and then for each customer by DATE, since the relationship between dates of purchase is an aspect of loyalty..
2. The RETAIN statement is used to initialize two variables that will carry the DATE from one purchase record to the next so time between purchases can be computed. At the beginning of each iteration QUALDATE1 is the date of the second previous purchase and QUALDATE2 is the date of the most recent purchase.
3. RETAINed information is used in conjunction with information in the current record to evaluate each customer's loyalty. When a record is encountered for which the second previous purchase (QUALDATE1) was more than 90 days before the current one (DATE) the customer is identified as not loyal. The values of DATE, QUALDATE1 and QUALDATE2 when this evaluation is done (i.e., beginning with the third purchase for each customer) are:

ID	QUALDATE1	QUALDATE2	DATE
1	27OCT2002	15DEC2002	09JAN2003
1	15DEC2002	09JAN2003	30JAN2003
2	15NOV2002	27FEB2003	27MAR2003

In both cases for customer 1 the difference between DATE and QUALDATE2 is less than 90 days, so ISLOYAL remains 0 until the last record and customer 1 is included in the LOYAL dataset. For customer 2 the difference exceeds 90 days, so ISLOYAL is assigned a value of 1 and no record is output for customer 2.

Note that in a situation like this the LAG function may also be used to RETAIN dates from one period to the next.

CASE #3

In our final case the requirement is to identify polypharmacy cases from prescription drug claim records. The polypharmacy cases to be identified are those patients who are taking two or more drugs, each prescribed by a different doctor, in each of three drug classes.

```
data RxClaims ;
  infile datalines ;
  input Patient_ID $ Doctor $ Drug_Class $
Drug_Num $;
datalines;
1   Miller  A   101
1   Sanchez B   111
1   Hayes   B   112
1   Corbin  C   113
1   Singh   C   133
1   Leary   A   104
2   Miller  A   102
2   Sanchez A   103
2   Hayes   B   114
2   Hayes   B   116
2   Corbin  C   123
2   Singh   C   125
3   Anders  A   101
3   Leary   A   103
3   Jones   B   115
3   Sanchez C   123
3   Singh   C   126
run;
```

To simplify matters, the illustration does not have multiple claims for a single drug.

Identifying the polypharmacy cases requires considering multiple attributes of each transaction record, in the context of the values of those attributes on other records. Such cases will often involve repeating the techniques described in the previous cases in a 'nested' fashion.

```
data PolyPharm (keep = Patient_ID);
  set RxClaims ;
  by Patient_ID Drug_Class Drug_Num;      1
  Retain Doctor1 ;
  if first.Patient_ID
     then Drug_Classes = 0 ;              3
  if first.Drug_Class then do ;
       Drug_Nums = 1 ;
       Doctor1 = Doctor ;
      end;
  if Drug_Nums = 1 and Doctor ne Doctor1
     then Drug_Nums + 1 ;
  if last.Drug_Class and Drug_Nums>1      2
     then Drug_Classes + 1 ;
  if last.Patient_ID and Drug_Classes>=3
     then output PolyPharm;
run;
```

This code illustrates such nesting of the techniques used in the previous cases.

1. Data is ordered and set by the entity variable (PATIENT_ID) and then by two criterion variables (DRUG_CLASS and DRUG_NUM).
2. A counter variable tracks each criterion variable, with the incrementing of the second dependent on the value of the first. The counters, DRUG_NUMS and DRUG_CLASSES, are RETAINed by use of the SUM statement.
3. The FIRST automatic variable is invoked iteratively, first for PATIENT_ID and then for DRUG_CLASS, to initialize the counters first for each patient and then for each drug class received by the patient.

It is left as an exercise for the reader to determine the value of DOCTOR1, DRUG_NUMS and DRUG_CLASSES at each step of each iteration, and the reason POLYPHARM includes patient 1 and not patients 2 or 3

ALTERNATIVES

It should be noted that in some cases PROC SQL can be used to accomplish this task more efficiently, most notably in cases (like the first) for which the requirement can be implemented solely by counting by sets. Even in these cases, though, there are instances when the methods discussed here may reasonably be used. These include when the programmer is unfamiliar with SQL, and when a data step has to be used anyways for other purposes.

CONCLUSION

The RETAIN statement is a useful tool for the analyst needing to identify and label a characteristic of entities when the information that identifies the characteristic is spread over multiple records.

REFERENCES

SAS Institute, Inc., 1999. *SAS Online Doc, Version 8*. Cary, NC; SAS Institute, Inc.

Aster, R. , 2000. *Professional SAS Programming Logic*. Paoli, PA; Breakfast Communications, Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

John D. Chapman, Ph.D.
Markcelian Analytics, Inc.
25 Virginia Lane
Cohasset, MA 02025
(781) 383-9793
jchapman@markcelian.com