Abstract

Building a two dimensional report using PROC TABULATE or PROC MEANS may require several data steps. In some cases, PROC SQL can produce similar results using a simpler process. This paper presents a method for creating summaries by class and interval using PROC SQL.

Introduction

A two dimensional report presents a summarized statistic using two types of segmentation – one on the vertical axis and one on the horizontal axis. Two dimensional reports provide a concise way to align like segments and compare statistics. Reports with two or more dimensions of summary variables are commonly used in many industries.

To create effective two dimensional reports, start by designing a final report template that best meets your needs and work back through the reporting process. The final report should be designed to display the most useful statistics and segmentations in a meaningful way. Examples of two dimensional reports are shown below. The examples are taken from the finance industry.

Example Report - Credit Exposure

Credit exposure reports are a useful tool for monitoring risk exposure by product type. This example shows how a table of application data might be summarized to report on exposure by product type and credit score band.

<table>
<thead>
<tr>
<th>App #</th>
<th>Product</th>
<th>FICO</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>$0 Fee, 9.99%</td>
<td>785</td>
<td>$15,000</td>
</tr>
<tr>
<td>1001</td>
<td>$39 Fee, 19.99%</td>
<td>642</td>
<td>$1,500</td>
</tr>
<tr>
<td>1002</td>
<td>$0 Fee, 14.99%</td>
<td>738</td>
<td>$8,000</td>
</tr>
<tr>
<td>1003</td>
<td>$0 Fee, 19.99%</td>
<td>764</td>
<td>$12,000</td>
</tr>
</tbody>
</table>

Example Report – Application Volume (Code Example)

Comparisons between a recent period and an older period are useful for monitoring changing trends. This example shows how a table of application data might be summarized to report by school and time interval.

<table>
<thead>
<tr>
<th>App Mth</th>
<th>App ID</th>
<th>School</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>200806</td>
<td>1000</td>
<td>Duke</td>
<td>Appr</td>
</tr>
<tr>
<td>200806</td>
<td>1001</td>
<td>Michigan St</td>
<td>Deny</td>
</tr>
<tr>
<td>200806</td>
<td>1002</td>
<td>Michigan St</td>
<td>Appr</td>
</tr>
<tr>
<td>201005</td>
<td>1200</td>
<td>West Virginia</td>
<td>Appr</td>
</tr>
<tr>
<td>201005</td>
<td>1201</td>
<td>Butler</td>
<td>Deny</td>
</tr>
</tbody>
</table>

Reporting Steps

Two dimensional reports are frequently coded in SAS using three or more data steps. A standard process might include a step for each of the following:
- A data step to define reporting classes and intervals
- PROC TABULATE to summarize variables
- A second data step to label and format fields

The resulting code can be challenging to understand and difficult to modify.

In many cases, using PROC SQL can reduce code and simplify reporting processes. Presented below is the SAS code used to create an application volume report similar to the one shown above.
Sample Data Code
Reads in sample application data to summarize

```sas
DATA Data_Source;
  INPUT @01 App_Mth  6.
    @08 App_ID   4.
    @13 School   $13.
    @27 Status   $4.;
CARDS;
  200806 1000 Duke Appr
  201005 1011 Butler Deny
 Full Sample Not Shown;
RUN;
```

Basic Report Code
Creates a report on application and approval volume by school using proc sql

```sas
PROC SQL; SELECT
  School,
  /* Sum records that fall into the required reporting interval: */
  SUM(CASE WHEN 200906 <= App_Mth <= 201005 THEN 1 ELSE 0 END) AS "Applied 0906-1005"
  /* A second method that requires less code but is more cryptic: */
  SUM(200806 <= App_Mth <= 200905) AS "Applied 0806-0905"
  /* "200906 <= App_Mth <= 201005" returns 1 if true and 0 if false */
  SUM((200906 <= App_Mth <= 201005) AND (Status = "Appr")) AS "Approved 0906-1005"
  SUM((200806 <= App_Mth <= 200905) AND (Status = "Appr")) AS "Approved 0806-0905"
FROM Data_Source GROUP BY School
  /* Sort to prioritize key segments: */
ORDER BY "Applied 0906-1005" DESC;
QUIT;
```

Code Output:

<table>
<thead>
<tr>
<th>School</th>
<th>Applied 0906-1005</th>
<th>Applied 0806-0905</th>
<th>Approved 0906-1005</th>
<th>Approved 0806-0905</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michigan</td>
<td>67</td>
<td>50</td>
<td>33</td>
<td>30</td>
</tr>
<tr>
<td>West Virginia</td>
<td>20</td>
<td>27</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>Duke</td>
<td>15</td>
<td>13</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Butler</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Rolling Report Code
Creates a formatted, rolling report using proc sql and SAS Macros

```sas
%LET Cur_Mth = 201005;
/* The Add_Mths macro returns the result of adding to a month value in YYYYMM format: */
%INCLUDE "\SASDrive\SAS Macros\Add_Mths.sas" / NOSOURCE;
/* The Fmt_Mth macro requires a YYYYMM value and returns the corresponding MMMYY value: */
%INCLUDE "\SASDrive\SAS Macros\Fmt_Mth.sas" / NOSOURCE;
/* The macro variables below will update automatically based on Cur_Mth: */
%LET Y1_Beg = %Add_Mths(&Cur_Mth, -11); /* Y1_Beg equals "200906" */
%LET Y1_End = %Cur_Mth; /* Y1_End equals "201005" */
%LET Y1_Sfx = %Fmt_Mth(&Y1_Beg)-%Fmt_Mth(&Y1_End); /* Y1_Sfx equals "Jun09-May10" */
%LET Y2_Beg = %Add_Mths(&Cur_Mth, -23); /* Y2_Beg equals "200806" */
%LET Y2_End = %Add_Mths(&Cur_Mth, -12); /* Y2_End equals "200905" */
%LET Y2_Sfx = %Fmt_Mth(&Y2_Beg)-%Fmt_Mth(&Y2_End); /* Y2_Sfx equals "Jun08-May09" */
```
PROC SQL;
CREATE TABLE Source_Summary AS SELECT
   School,
   /* The macro variables are used to set the intervals and the field names */
   SUM((&Y1_Beg <= App_Mth <= &Y1_End) AS "Applied &Y1_Sfx"n,
   SUM((&Y2_Beg <= App_Mth <= &Y2_End) AS "Applied &Y2_Sfx"n,
   SUM((&Y1_Beg <= App_Mth <= &Y1_End) AND (Status = "Appr")) AS "Approved &Y1_Sfx"n,
   SUM((&Y2_Beg <= App_Mth <= &Y2_End) AND (Status = "Appr")) AS "Approved &Y2_Sfx"n
FROM Data_Source GROUP BY School
/* Order by the num of apps received last year: */
   ORDER BY "Applied &Y1_Sfx"n DESC

/* Add approval rate calculations to the report: */
SELECT *
   CASE WHEN "Applied &Y1_Sfx"n > 0 THEN "Approved &Y1_Sfx"n / "Applied &Y1_Sfx"n
   ELSE 0 END AS "Appr_Rate &Y1_Sfx"n FORMAT=PERCENT5.,
   CASE WHEN "Applied &Y2_Sfx"n > 0 THEN "Approved &Y2_Sfx"n / "Applied &Y2_Sfx"n
   ELSE 0 END AS "Appr_Rate &Y2_Sfx"n FORMAT=PERCENT5.
/* Keep only the top two schools by application volume: */
FROM Source_Summary (OBS=2)
; QUIT;

Code Output:

<table>
<thead>
<tr>
<th>School</th>
<th>Applied Jun09-May10</th>
<th>Applied Jun08-May09</th>
<th>Approved Jun09-May10</th>
<th>Approved Jun08-May09</th>
<th>Appr_Rate Jun09-May10</th>
<th>Appr_Rate Jun08-May09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michigan</td>
<td>67</td>
<td>50</td>
<td>33</td>
<td>30</td>
<td>49%</td>
<td>60%</td>
</tr>
<tr>
<td>West Virginia</td>
<td>20</td>
<td>27</td>
<td>12</td>
<td>11</td>
<td>60%</td>
<td>41%</td>
</tr>
</tbody>
</table>

Macro Code

Code for the Add_Mths and Fmt_Mth macros included above

%MACRO Add_Mths(Mth, Num_Mths);
   /* The Add_Mths macro returns the result of adding or subtracting a given */
   /* number of months to a SAS text value in YYYYMM format. */
   /* Mth: A month value in YYYYMM format. */
   /* Num_Mths: The number of months to add (a negative value for subtraction). */
   %LOCAL Num_Yrs;
   %LET Num_Yrs = %SYSFUNC(floor((%SUBSTR(&Mth, 5, 2) + &Num_Mths - 1) / 12));
   %EVAL(&Mth + 88 * &Num_Yrs + &Num_Mths);
%MEND Add_Mths;

%MACRO Fmt_Mth(Mth);
   /* The Fmt_Mth macro returns the result of reformatting a SAS text value from */
   /* YYYYMM format to MMMYY format. */
   /* Mth: A month value in YYYYMM format. */
   %LOCAL MM YY;
   %LET MM = %SUBSTR(&Mth, 5, 2);
   %LET YY = %SUBSTR(&Mth, 3, 2);
   %IF &MM = 01 %THEN Jan&YY; %ELSE_IF &MM = 02 %THEN Feb&YY;
   %ELSE &MM = 03 %THEN Mar&YY; %ELSE &MM = 04 %THEN Apr&YY;
   %ELSE &MM = 05 %THEN May&YY; %ELSE &MM = 06 %THEN Jun&YY;
   %ELSE &MM = 07 %THEN Jul&YY; %ELSE &MM = 08 %THEN Aug&YY;
   %ELSE &MM = 09 %THEN Sep&YY; %ELSE &MM = 10 %THEN Oct&YY;
   %ELSE &MM = 11 %THEN Nov&YY; %ELSE &MM = 12 %THEN Dec&YY;
%MEND;
**Conclusion**

When summarizing data, the features offered by PROC SQL can help reduce code and simplify reporting processes. SQL can do all of the following in one step:
- Conditional Assignments
- Data Aggregation
- Field Labeling
- Field Ordering
- Sorting / Prioritization

PROC SQL also has several SAS-inherited capabilities not available to standard SQL. The combination of SQL and SAS features provides unique advantages.

**Acknowledgements**

The reporting processes presented in this paper were developed with support from fellow SAS users. Special thanks to Andrew Larsen, Donalee Wanna, Dale Robinson, Grant Walker, James Jenson, Neil Davies, and Scott Pierson for their review and considerable feedback.

**Contact Information**

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

Nathan Lindquist  
4820 Park Commons Drive #219, St. Louis Park, MN 55416  
(952) 922-2481  
nathan.lindquist.mn@gmail.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.