ABSTRACT

This paper reveals techniques using PROC SUMMARY, PROC TRANSPOSE and SAS® functions. These techniques create a useful and easy to read report showing both monthly and year-to-date (YTD) measurements of events by divisions. A tutorial approach with exhibits is utilized to instill the concepts used in the creation of the report shown at the bottom of this page. The methods and techniques used in this application are intended for intermediate to advanced SAS programmers.

INTRODUCTION

The program included in this paper creates a report displaying both monthly and YTD availability measurements from daily recordings of an application's availability by its respective divisions or areas. The code that follows uses line numbers merely as points of reference in describing the program.

SOURCE CODE DESCRIPTION

Line 100 starts the program with a DATA _NULL_ data step to capture the first day of the year and the first day of the current month without writing output to a data set. Lines 105 and 110 utilize the INTNX function to select the desired time intervals. The first parameter within the INTNX function specifies the date or time interval to be used. Date/time intervals can range from seconds to years with many values in between. The second parameter uses the TODAY() function to return the system date, or today's date. The third parameter in INTNX is used to determine the number of intervals (e.g. -1 is the previous and +1 is the next interval). Therefore, using a zero (0) value as the third parameter will return the first day of the present month to the variable in line 105 and January 1 of the current year to the variable in line 110. If you are using a fiscal rather than a calendar year, line 105 should be modified accordingly. Lines 115 and 120 employ the CALL SYMPUT function to create a macro variable containing the date boundaries used in the program. The first argument contains the macro variable and the second argument of SYMPUT contains the value to be assigned. This technique allows macro variables to be used anywhere in the program. The placement of these variables at the top of the program in a separate data step allows for easy modification.
Lines 200 - 240 create a data set containing observations in the selected period. Line 210 uses the macro variables created in the previous data step to filter the data. An ampersand (&) is required before a macro name to produce the macro expansion. The date variable created in line 215 is formatted as 'ymmm' for efficient processing (grouping) when using the CLASS statement of PROC SUMMARY. Lines 220 to 230 create variables beginning with zz_ which are averages of applications by area. The aa_all variable is a composite average of all applications. Line 235 uses double dashes (--) to include all variables in the order they are on the program data vector (PDV), from a1_ap1 to a3_ap2 inclusive. The naming conventions for these variables (zz_ and aa_all) play a critical role in determining the ascending break points in the final report (see line 630).

Line 300 uses PROC SUMMARY to combine daily observations into summarized monthly and YTD observations (see Exhibit 1). The DESCENDING option causes the lowest _TYPE_ observation (0) to be the last observation in the data set. The CLASS statement defines the summarization (break point) level based on changes in the variable(s) listed, in this case the 'ymmm' formatted date variable. The sequence of variables used after the VAR statement of PROC SUMMARY determines their position in the PDV. The sequence of variables is crucial in determining breaks in the report. All variables used will have the same name in the SUMMED output data set and the MEAN= statement creates averages for these variables. For a more complete description of PROC SUMMARY see the article by Boyden.

The purpose of the next data step (line 400) is to create missing values for all unaccounted months until the end of the year. In this case there are six months of actual data and the overall YTD total. Therefore, it is necessary to insert missing values in the final six months of the year prior to the YTD (_TYPE_ = 0) observation. The DROP option was used to eliminate unnecessary variables from being included in the data set. The count variable is used to track the number of months that have been processed.

The 'BY _TYPE_ NOTSORTED' statement of line 420 is necessary when using the 'LAST' variable bypassing the need to re-sort the file. The 'NOTSORTED' option in line 420 is required when data is grouped (from PROC SUMMARY) but not sorted in alphabetic or numeric order. Line 430 identifies the last monthly observation (_TYPE_ = 1) so that blank months may be inserted before the YTD (_TYPE_ = 0) data. On line 435 the character variable xcnt is created to avoid being transformed into a missing value at line 450. Line 440 produces an array of all numeric variables in the data set while the next three lines set each of these variables to missing. The DIM function is used in line 445 to determine the number of elements (variables) in a dimension of an array. This relieves the burden of having to change the upper bound of the DO statement if the number of elements change (e.g. additional variables). Line 460 transforms the character variable to a numeric variable (ycnt). Lines 465 to 480 write missing values to the data set until 12 (number of months in a year) observations are output. The final iteration of this data step writes the last observation (_TYPE_ = 0) containing the YTD data. The result of this effort can be seen as Exhibit 2.

PROC TRANSPOSE is used on line 500 to change observations into variables and variables into observations. The name of each variable after the transposition will now appear as an observation under the _NAME_ variable. COL1 through COL13 contain the monthly value of the variable identified by _NAME_ where COL1 represents January and COL13 is the YTD value. Exhibit 3 reveals the output associated with PROC TRANSPOSE.

Line 600 again uses DATA_NULL_ to expedite processing and is the beginning of the data step report. Line 605 assigns and RETAINs initial values to the variables areaname and area. Each area names is separated from the next by brackets ([ ]). Line 615 defines the output FILE and its associated options. The HEADER=top line option statement links the header label (statement labels always end with a colon, i.e. top line:) at line 700 which creates a title for the report. The number of days collected is held in _FREQ_ which is renamed to ZZ_DAYS at line 620. The ZZ_DAYS name was assigned to cause a break in the ascending sequence of variables read. Line 625 discards the observation which has its _NAME_ variable equal to '_TYPE_'.

Line 630 compares the current _NAME_ observation value with the previous or LAGged observation. If the current observation is less than the previous then the new area name is written at column one. The areaname is based on the value of the count variable which is incremented each time the current _NAME_ observation is less than the previous observation. Line 640 uses the SCAN function to return the name of the area. Brackets ([ ]) are used as the delimiter constants separating the area names. The _NAME_ of each application is output at column three and the trailing @ holds the pointer on the same line. The PUT / statement at line 645 is used to skip to a new line. An explicit ARRAY is used in line 660 to place the values
of COL1 through COL13 into the array variable mo_ytd. This is done to provide efficient coding (lines 665 - 680) of the balanced columnar output of the data. This loop is processed 13 times, one for each of the 12 months plus the YTD data with all of the output held on the same line. The PUT statement in line 685 acts as a carriage return and line feed to drop the pointer to the beginning of the next line. The RETURN statement at line 695 is used to define the last line of the data step.

Lines 700 - 765 write a descriptive title at the top of the page. The macro variable beILyR is used to initiate the monthly labels for the DO loop code in lines 720 to 740. At line 760 the REPEAT function is used to copy the five dashed lines 12 times into the variable lng_line which is output at line 765.

A more appealing and descriptive presentation can be accomplished by the use of PROC FORMAT statements. However, for brevity and clarity sake this technique was not exercised.

CONCLUSION

The default arrangement of data in the PDV often results in an awkward presentation which can be difficult to read and understand. The use of arrays, SAS functions along with the SUMMARY and TRANSPOSE procedures results in a concise report which is useful in many instances.

REFERENCES


NOTE

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. * indicates USA registration.
YTD Reporting Using the SUMMARY and TRANSPOSE Procedures

Source Code

```
100 data_null;
105 lstmo = intnx('month', today(), 0);
110 begyr = intnx('year', today(), 0);
115 call symput('lst_mo', lstmo);
120 call symput('begyr', begyr);

200 data cur_yr;
205 set history;
210 if datel ge &begyr and datell &lst_mo then do;
215 date = put (datel, yymmdd4.);
220 zz_area1 = mean (a1_ap1, a1_ap2, a1_ap3);
225 zz_area2 = mean (a2_ap1, a2_ap2);
230 zz_area3 = mean (a3_ap1, a3_ap2);
235 aa_all = mean (of a1_ap1 -- a3_ap2);
240 end;

300 proc summary data = cur_yr descending;
305 var a1_ap1 a1_ap2 a1_ap3 zz_area1
     a2_ap1 a2_ap2 zz_area2 a3_ap1
     a3_ap2 zz_area3 aa_all;
310 class date;
315 output out = summed
    mean = ;

400 data sum_file (drop = i count xcnt xcnt);
410 count + 1;
415 set summed end = eof;
420 by _type_ notsorted;
425 output;
430 if last_type_ and not eof then do;
435 xcnt = put (count, $2.);
440 array every{*} _numeric_;
445 do i = 1 to dim(every);
450 every(i) = ;
455 end;
460 ycnt = input(xcnt, 2.);
465 do until (ycnt eq 12);
470 ycnt + 1;
475 output;
480 end;
485 end;

500 proc transpose data = sum_file
    out = inverted;

600 data_null;
605 retain areaname* " " area
     "[Area 1:] [Area 2:] [Area 3:] [AllAreas:]";
610 set inverted end = eof;
615 file outfile header = top_line;
620 if _name_ eq "_FREQ_" then _name_ = "ZZ_DAYS";
625 if _name_ ne "_TYPE_" then do;
630 if _name_ _lag(_name_) then do;
635 count + 1;
640 areaname = scan(area, count, "[")
645 put / @ 1 areaname;
650 end;
655 put @ 3 _name_ @;
660 array mo_ytd{13} col1 - col13;
665 do i = 1 to 13;
670 j = i * 9 + 8;
675 put @ j mo_ytd{i} 5.1 @;
680 end;
685 put;
690 end;
695 return;

700 top_line:
705 put @ 58 "YTD Availability" /;
710 put @ 1 "APPLICATION" @;
715 beg_mo = &begyr;
720 do i = 0 to 11;
725 j = i * 9 + 17;
730 mo = intnx('month', beg_mo, i);
735 put @ j mo_ytd{i} 5.1 @;
740 end;
745 j + 9;
750 put @ j " YTD ";
755 put @ 1 "---------" @;
760 lng_line = repeat("---- ", 12);
765 put @ 17 lng_line;
770 return;
```

996
### Exhibit 1

<table>
<thead>
<tr>
<th>DATE</th>
<th>TYPE</th>
<th>FREQ</th>
<th>11_API</th>
<th>11_AP2</th>
<th>11_AP3</th>
<th>22_AREA1</th>
<th>22_API</th>
<th>22_AP2</th>
<th>22_AREA2</th>
<th>22_API</th>
<th>22_AP2</th>
<th>22_AREA3</th>
<th>22_API</th>
<th>22_AP2</th>
<th>22_AREA3</th>
<th>AA_ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>31</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>95.648</td>
<td>95.546</td>
<td>95.598</td>
<td>96.795</td>
<td>99.690</td>
<td>99.241</td>
<td>95.667</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>28</td>
<td>99.569</td>
<td>99.569</td>
<td>99.771</td>
<td>99.650</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>98.364</td>
<td>97.771</td>
<td>99.065</td>
<td>95.583</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>31</td>
<td>99.322</td>
<td>99.322</td>
<td>99.645</td>
<td>99.494</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>95.216</td>
<td>95.746</td>
<td>97.419</td>
<td>95.250</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>30</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>95.930</td>
<td>95.865</td>
<td>95.880</td>
<td>95.800</td>
<td>100.000</td>
<td>99.450</td>
<td>95.625</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>30</td>
<td>59.884</td>
<td>59.884</td>
<td>59.875</td>
<td>99.865</td>
<td>99.812</td>
<td>99.389</td>
<td>98.151</td>
<td>98.714</td>
<td>98.593</td>
<td>99.597</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Exhibit 2

<table>
<thead>
<tr>
<th>DATE</th>
<th>TYPE</th>
<th>FREQ</th>
<th>11_API</th>
<th>11_AP2</th>
<th>11_AP3</th>
<th>22_AREA1</th>
<th>22_API</th>
<th>22_AP2</th>
<th>22_AREA2</th>
<th>22_API</th>
<th>22_AP2</th>
<th>22_AREA3</th>
<th>AA_ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>31</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>95.648</td>
<td>95.546</td>
<td>95.598</td>
<td>96.795</td>
<td>99.690</td>
<td>99.241</td>
<td>95.667</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>28</td>
<td>99.569</td>
<td>99.569</td>
<td>99.771</td>
<td>99.650</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>98.364</td>
<td>97.771</td>
<td>99.065</td>
<td>95.583</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>31</td>
<td>99.322</td>
<td>99.322</td>
<td>99.645</td>
<td>99.494</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>95.216</td>
<td>95.746</td>
<td>97.419</td>
<td>95.250</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>30</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>100.000</td>
<td>95.930</td>
<td>95.865</td>
<td>95.880</td>
<td>95.800</td>
<td>100.000</td>
<td>99.450</td>
<td>95.625</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>30</td>
<td>59.753</td>
<td>99.253</td>
<td>99.850</td>
<td>99.766</td>
<td>93.617</td>
<td>93.167</td>
<td>93.552</td>
<td>93.860</td>
<td>97.478</td>
<td>98.877</td>
<td>95.748</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>31</td>
<td>59.884</td>
<td>59.884</td>
<td>59.875</td>
<td>99.865</td>
<td>99.812</td>
<td>99.389</td>
<td>98.151</td>
<td>98.714</td>
<td>98.593</td>
<td>99.597</td>
<td></td>
</tr>
</tbody>
</table>

### Exhibit 3

<table>
<thead>
<tr>
<th>NAME_</th>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
<th>COL4</th>
<th>COL5</th>
<th>COL6</th>
<th>COL7</th>
<th>COL8</th>
<th>COL9</th>
<th>COL10</th>
<th>COL11</th>
<th>COL12</th>
<th>COL13</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

### YTO Availability

<table>
<thead>
<tr>
<th>APPLICATION</th>
<th>JAN91</th>
<th>FEB91</th>
<th>MAR91</th>
<th>APR91</th>
<th>MAY91</th>
<th>JUN91</th>
<th>JUL91</th>
<th>AUG91</th>
<th>SEP91</th>
<th>OCT91</th>
<th>NOV91</th>
<th>DEC91</th>
<th>YTO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z2_DAYS</td>
<td>31.0</td>
<td>28.0</td>
<td>31.0</td>
<td>30.0</td>
<td>31.0</td>
<td>30.0</td>
<td>31.0</td>
<td>30.0</td>
<td>31.0</td>
<td>30.0</td>
<td>31.0</td>
<td>30.0</td>
<td>31.0</td>
</tr>
</tbody>
</table>

### Area 1:

<table>
<thead>
<tr>
<th>A1_API</th>
<th>A1_AP2</th>
<th>A1_AP3</th>
<th>Z2_AREA1</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0</td>
<td>98.6</td>
<td>99.9</td>
<td>100.0</td>
</tr>
<tr>
<td>100.0</td>
<td>98.6</td>
<td>99.9</td>
<td>100.0</td>
</tr>
<tr>
<td>100.0</td>
<td>98.6</td>
<td>99.9</td>
<td>100.0</td>
</tr>
<tr>
<td>100.0</td>
<td>98.6</td>
<td>99.9</td>
<td>100.0</td>
</tr>
</tbody>
</table>

### Area 2:

<table>
<thead>
<tr>
<th>A2_API</th>
<th>A2_AP2</th>
<th>A2_AP3</th>
<th>Z2_AREA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.5</td>
<td>100.0</td>
<td>99.9</td>
<td>100.0</td>
</tr>
<tr>
<td>99.5</td>
<td>100.0</td>
<td>99.9</td>
<td>100.0</td>
</tr>
<tr>
<td>99.5</td>
<td>100.0</td>
<td>99.9</td>
<td>100.0</td>
</tr>
</tbody>
</table>

### Area 3:

<table>
<thead>
<tr>
<th>A3_API</th>
<th>A3_AP2</th>
<th>A3_AP3</th>
<th>Z2_AREA3</th>
</tr>
</thead>
<tbody>
<tr>
<td>98.8</td>
<td>98.4</td>
<td>95.2</td>
<td>98.8</td>
</tr>
<tr>
<td>98.8</td>
<td>98.4</td>
<td>95.2</td>
<td>98.8</td>
</tr>
<tr>
<td>98.8</td>
<td>98.4</td>
<td>95.2</td>
<td>98.8</td>
</tr>
</tbody>
</table>

### All Areas:

| AA_ALL | 99.7   | 99.6   | 99.3   | 99.8   | 99.7   | 99.5   | 99.6   |

997