Custom Reports with Proc SQL: A Real-Life Example
Electra Small, MDRC, New York, NY

ABSTRACT
Customized tables or reports produced entirely by SAS®? When this topic comes up, most SAS users think immediately of Proc Tabulate or Proc Report. This paper highlights the powerful report- and table-generating capacities of Proc SQL. Using a real-world project as an example, Proc SQL is used to generate a custom Excel table for project managers to monitor the progress of a large survey being fielded in multiple states across the country. The key to Proc SQL’s report-generating power is its ability to use character and numeric operators and functions, logical statements, and arithmetic computations — all within the report-generating select statement. All these capabilities are employed to generate the desired survey-tracking report. To allow comparison, the paper also displays the coding and resulting tables of Proc Tabulate and Proc Report.

INTRODUCTION
A large social research project with several thousand sample members in several large cities conducted a telephone survey with a portion of the sample in each city. Survey-tracking files were created at regular intervals to relay the progress of the telephone survey effort. From these tracking files, the following progress report was to be produced and circulated to project team members.

Table 1
Project 12-Month Survey Sample and Response Rates

<table>
<thead>
<tr>
<th>Site</th>
<th>Intake Period</th>
<th>Sample Size</th>
<th>Fielded Sample Size</th>
<th>Respondent Sample Size</th>
<th>Response Rate (%)</th>
<th>Response Rate By Research Group</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E-Group</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Total</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Total</td>
</tr>
<tr>
<td>City A</td>
<td>09/02-03/03</td>
<td>74</td>
<td>74</td>
<td>59</td>
<td>79.8</td>
<td>30</td>
</tr>
<tr>
<td>City B</td>
<td>02/02-06/02</td>
<td>90</td>
<td>74</td>
<td>59</td>
<td>79.8</td>
<td>29</td>
</tr>
<tr>
<td>City C</td>
<td>01/02-06/02</td>
<td>35</td>
<td>35</td>
<td>29</td>
<td>82.9</td>
<td>14</td>
</tr>
<tr>
<td>City D</td>
<td>09/02-12/02</td>
<td>25</td>
<td>25</td>
<td>18</td>
<td>72.0</td>
<td>9</td>
</tr>
<tr>
<td>City E</td>
<td>01/02-06/02</td>
<td>50</td>
<td>37</td>
<td>29</td>
<td>80.3</td>
<td>15</td>
</tr>
<tr>
<td>City F</td>
<td>10/01-12/02</td>
<td>123</td>
<td>91</td>
<td>71</td>
<td>78.3</td>
<td>35</td>
</tr>
<tr>
<td>City G</td>
<td>10/02-03/03</td>
<td>131</td>
<td>65</td>
<td>26</td>
<td>40.0</td>
<td>12</td>
</tr>
<tr>
<td>City H</td>
<td>07/02-12/02</td>
<td>175</td>
<td>109</td>
<td>75</td>
<td>68.8</td>
<td>38</td>
</tr>
<tr>
<td>City I</td>
<td>09/02-10/02</td>
<td>40</td>
<td>36</td>
<td>24</td>
<td>66.7</td>
<td>13</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>743</td>
<td>546</td>
<td>390</td>
<td></td>
<td>195</td>
</tr>
</tbody>
</table>

The prototype table above was created in Excel via hand input, cutting and pasting from the output of numerous SAS procedures. This paper walks through the steps to create a version of this table using SAS Proc SQL and ODS. A brief comparison is also made to the coding and resulting tables of Proc Tabulate and Proc Report.

THE CHALLENGE
This table presents statistics for more than one sample (eligibles, fielded, respondents). Simply, the number of sample members who (1) were eligible for survey sample selection, (2) were chosen for the survey sample, and (3) responded to the survey. These samples are not mutually exclusive but, rather, are subsets. Regardless of the report-production tool used (SQL, Tabulate, Report), comparing samples that have members in common requires special coding. SAS procedures easily do “by-group processing” that compares mutually exclusive groups (boys versus girls, adults versus children, Mets versus Yankees). However, in our prototype table, the respondent sample is a subset of the fielded sample, and the fielded sample is a subset of the eligible sample. Since “by-group processing” cannot be used to produce this table, we have to tell SAS explicitly who belongs in the numerator and denominator of each group. In other words, we have to code the calculations manually.
RATIOS, PERCENTAGES, AND SUMS FOR DUMMIES

Therefore, before we start considering Proc SQL’s powerful report-producing abilities, we need to review a few tricks for dummies. You will be happy to find out that, in this case, the “dummy” is not you but dummy variables. Dummy variables generally have three values. The value of zero on a dummy variable always signifies “false.” The value of 1 on a dummy variable signifies “true.” And the value of missing (.) on a dummy variable signifies “not applicable” or “not discernable.”

Often, when producing tables, we want to report percentages. The percentage, however, is not one of the statistics or summary functions available in Proc SQL. Therefore, dummy variables can be powerful aids in reporting percentages via Proc SQL, because the mean of a dummy variable is a ratio, and when you multiply a ratio by 100, you get a percentage. A mean is one of the summary functions available in Proc SQL. Ratios range in value from zero to 1, and percentages range in value from zero to 100.

Say, for example, there are 6 males and 4 females in the sample. You create a MALE dummy variable; all 6 males have a value of 1 (or true), and the 4 females have a value of 0 (or not true). A mean of the MALE dummy variable yields the ratio 0.6 (6/10). Further, multiplying this ratio by 100 yields the answer you hoped for: The sample is 60 percent male.

Various sample sizes are also often presented in tables. Though Proc SQL does allow you to request a count, a count of the MALE dummy in the example above would yield 10, or the number of “nonmissing,” or valid, cases. But if you want to report the male sample size (the number of men) in your table, another property of the dummy variable will help you. When you sum dummy variables, you get the number (or sample size) that is true for that variable. So summing the MALE dummy variable yields the answer you hoped for: The sample has a subgroup of 6 males.

LOGIC FOR DUMMIES

There is no need to use a data step to produce dummy variables for every percentage and sample size that you want to report. There is no need because of the way SAS and Proc SQL resolve logical statements. A “logical statement” is a statement that can be evaluated as true or false. When you create a variable using a logical statement (for example MALE=GENDER EQ “M”;), SAS assigns a value of 1 to the variable on the left of the equals sign if the statement on the right of the equals sign is true for that case. However, if SAS determines that the statement is not true (including if the input variables are missing), SAS assigns a value of zero to the variable for that case. So logical variables are dummy variables and can be used to calculate percentages and sample sizes in the ways described in the section above. SAS allows you to go further, by allowing you to submit logical statements for evaluation without the need to create a variable. For example, if you had a variable GENDER (6 males and 4 females in the data set) and wanted to present the percentage and number of males in the sample, you could calculate these statistics using logical statements in Proc SQL without the need to create the MALE dummy variable. To calculate the percentage of males in the sample, you could simply submit the following statement:

```
SELECT MEAN(GENDER='M') AS PCTMALE FROM MYDATASET;
```

To calculate the number of males in the sample, you could simply submit the following statement:

```
SELECT SUM(GENDER='M') AS SSMALE FROM MYDATASET;
```

THE DATA

To make the programming instructions easy to understand, a small made-up data set will be used:

```
data SURVEYTRCK;
  SITE=1; INTAKE='01JAN2004'D; ELIGIBLE=0; FIELDED=0; RESPONDED=0; RESGRP='E'; OUTPUT;
  SITE=1; INTAKE='01JAN2004'D; ELIGIBLE=0; FIELDED=0; RESPONDED=0; RESGRP='C'; OUTPUT;
  SITE=1; INTAKE='01DEC2004'D; ELIGIBLE=1; FIELDED=0; RESPONDED=0; RESGRP='E'; OUTPUT;
  SITE=1; INTAKE='01DEC2004'D; ELIGIBLE=1; FIELDED=0; RESPONDED=0; RESGRP='C'; OUTPUT;
  SITE=1; INTAKE='02JAN2004'D; ELIGIBLE=0; FIELDED=1; RESPONDED=0; RESGRP='E'; OUTPUT;
  SITE=1; INTAKE='02JAN2004'D; ELIGIBLE=0; FIELDED=1; RESPONDED=0; RESGRP='C'; OUTPUT;
  SITE=1; INTAKE='02OCT2004'D; ELIGIBLE=1; FIELDED=1; RESPONDED=1; RESGRP='E'; OUTPUT;
  SITE=1; INTAKE='02OCT2004'D; ELIGIBLE=1; FIELDED=1; RESPONDED=1; RESGRP='C'; OUTPUT;
  SITE=2; INTAKE='01JAN2004'D; ELIGIBLE=0; FIELDED=0; RESPONDED=0; RESGRP='E'; OUTPUT;
  SITE=2; INTAKE='01JAN2004'D; ELIGIBLE=0; FIELDED=0; RESPONDED=0; RESGRP='C'; OUTPUT;
  SITE=2; INTAKE='01DEC2004'D; ELIGIBLE=1; FIELDED=0; RESPONDED=0; RESGRP='E'; OUTPUT;
  SITE=2; INTAKE='01DEC2004'D; ELIGIBLE=1; FIELDED=0; RESPONDED=0; RESGRP='C'; OUTPUT;
  SITE=2; INTAKE='02JUN2004'D; ELIGIBLE=1; FIELDED=1; RESPONDED=0; RESGRP='E'; OUTPUT;
  SITE=2; INTAKE='02JUN2004'D; ELIGIBLE=1; FIELDED=1; RESPONDED=0; RESGRP='C'; OUTPUT;
  SITE=2; INTAKE='03JUL2004'D; ELIGIBLE=1; FIELDED=1; RESPONDED=1; RESGRP='E'; OUTPUT;
  SITE=2; INTAKE='03JUL2004'D; ELIGIBLE=1; FIELDED=1; RESPONDED=1; RESGRP='C'; OUTPUT;
  SITE=2; INTAKE='04SEP2004'D; ELIGIBLE=1; FIELDED=1; RESPONDED=1; RESGRP='E'; OUTPUT;
  SITE=2; INTAKE='04SEP2004'D; ELIGIBLE=1; FIELDED=1; RESPONDED=1; RESGRP='C'; OUTPUT;
```

2
As you can see, we have only 2 sites in this example data set, and each of the sites has exactly 8 records. Sample intake ranged between January 2004 and September 2004 in both sites. Both sites have 6 cases eligible to be included in the survey sample, but only 4 cases were chosen for the survey sample (fielded). In both sites, 2 people responded to the survey; 1 respondent was in the experimental group, and 1 respondent was in the control group.

CODING THE REPORT
Let’s walk through the building of this custom table and the manual calculations, column by column. To view the results of each column’s calculations, turn to the table displayed under “Checking the Results” below.

COLUMN 1 (SITE): GROUP-BY PROCESSING
As you review the prototype table, the first thing that stands out is that each row presents the information about one site or city where the survey is being conducted. However, the survey-tracking data set that we are using has one row for each individual in the research project. The rows (observations) in the data set need to be summarized by the SITE variable, using the GROUP BY clause in Proc SQL. Further, not only do we want the data summarized by SITE, but we want the rows in the table to be ordered by the values of the SITE variable. So starting the SQL coding, we have the following:

PROC SQL;
SELECT SITE AS SITE LABEL='Site'
FROM SURVEYTRCK
GROUP BY SITE
ORDER BY SITE;
QUIT;

COLUMN 2 (INTAKE PERIOD): SUMMARY FUNCTIONS AND OPERATORS
For this column, we simply need to present the earliest and the latest dates that sample members joined the sample in each city. We use Proc SQL’s MIN and MAX summary functions to calculate these for each site. Both Proc Means and Proc SQL will allow you to calculate the mean, minimum, and maximum of date variables, but only Proc SQL will allow you to present the results formatted with a SAS date format. The following code will return the range of the dates that we need:

PROC SQL;
SELECT SITE AS SITE LABEL='Site',
MIN(INTAKE) AS MINTAKE FORMAT=MMYYS7.,
MAX(INTAKE) AS MAXTAKE FORMAT=MMYYS7.,
FROM SURVEYTRCK
GROUP BY SITE
ORDER BY SITE;
QUIT;

The code above will return these values in two separate columns (MINTAKE and MAXTAKE). The prototype table, however, presents the range in one column. To present our date range in this fashion, we must calculate the minimum and maximum dates and then present the results as a concatenated (“smushed-together”) character string. This next section of code demonstrates how most of the SAS functions and operators that are available in the data step are also usable in a Proc SQL statement. In this case, we use the PUT function to convert the date range into a character string, and we use the concatenation operator to add a dash in the middle of the range:

PROC SQL;
SELECT SITE AS SITE LABEL='Site',
PUT(MIN(INTAKE),MMYYS7.)||"-"||PUT(MAX(INTAKE),MMYYS7.) AS INTAKE LABEL='Intake Period'
FROM SURVEYTRCK
GROUP BY SITE
ORDER BY SITE;
QUIT;

COLUMNS 3 TO 5 (SAMPLE SIZES): SUMMING SAMPLE DUMMIES
The next three columns in the prototype table simply present the number of sample members who (1) were eligible for survey sample selection, (2) were chosen for the survey sample, and (3) responded to the survey. In the input data set, there are three dummy variables that indicate each sample member’s status relative to these subgroups. Summing these dummy variables presents the sample sizes we need:

PROC SQL;
SELECT SITE AS SITE LABEL='Site',
      PUT(MIN(INTAKE),MMYYS7.)||"-"||PUT(MAX(INTAKE),MMYYS7.) AS INTAKE LABEL='Intake Period',
      SUM(ELIGIBLE) AS ELIGIBLE LABEL='Eligible Sample Size',
      SUM(FIELDED) AS FIELDED LABEL='Fielded Sample Size',
      SUM(RESPONDED) AS RESPONDED LABEL='Respondent Sample Size'
FROM SURVEYTRCK
GROUP BY SITE
ORDER BY SITE;

COLUMN 6 (SURVEY RESPONSE RATES): CALCULATING MEANS AND PERCENTAGES
The survey response rate is simply the percentage of the fielded sample who have responded to the survey. As explained earlier in this paper, the mean of a dummy variable will yield the “rate,” or percentage, we need for this column. Deciding how to calculate this mean depends on how the data are coded in the input data set. We can use Proc SQL’s mean summary function if our RESPONDED dummy variable is equal to 1 when fielded members responded, equal to 0 when fielded members have not yet responded, and (most important) equal to missing (.) when a sample member is not part of the fielded sample. Simply put, we want the response rate to be calculated on the fielded base, not the eligible base. If our RESPONDED dummy variable is missing for everyone who is not part of the fielded sample (missing for people who were only eligible for the survey but were not fielded), then the Proc SQL mean summary function will calculate the response rate correctly:

PROC SQL;
    {snip},
    MEAN(RESPONDED)*100 LABEL='Response Rate(%)'
    {snip}
QUIT;

However, in the input data set, the dummy variables have a value of 1 to signify “true” and a value of 0 for all others, including “not applicable” sample members. So we need to control the way that the mean is calculated, by coding the membership of the numerator and denominator:

PROC SQL;
    SELECT SITE AS SITE LABEL='Site',
            PUT(MIN(INTAKE),MMYYS7.)||"-"||PUT(MAX(INTAKE),MMYYS7.) AS INTAKE LABEL='Intake Period',
            SUM(ELIGIBLE) AS ELIGIBLE LABEL='Eligible Sample Size',
            SUM(FIELDED) AS FIELDED LABEL='Fielded Sample Size',
            SUM(RESPONDED) AS RESPONDED LABEL='Respondent Sample Size',
            (SUM(RESPONDED)/SUM(FIELDED))*100 AS RESPRATE FORMAT=5.1 LABEL='Response Rate (%)'
FROM SURVEYTRCK
GROUP BY SITE
ORDER BY SITE;

COLUMNS 6 AND 8 (RESEARCH GROUP SAMPLE SIZES): LOGICAL STATEMENTS
These columns give us the opportunity to use our newfound knowledge about how Proc SQL resolves logical statements, saving us the need to create a dummy, or logical, variable. The survey-tracking data set has a research group variable (RESGRP) with the values of "E" and "C" for the experimental and control groups, respectively. To count the number of experimentals in the survey sample, we need to count the number of sample members for whom the following logical statement is true: RESGRP='E' AND FIELDED=1. The FIELDED=1 can also be just written as “FIELDED.” SAS interprets this as “where FIELDED is true” (or greater than zero). So the code for Column 6 is as follows:

PROC SQL;
    SELECT SITE AS SITE LABEL='Site',
            PUT(MIN(INTAKE),MMYYS7.)||"-"||PUT(MAX(INTAKE),MMYYS7.) AS INTAKE LABEL='Intake Period',
            SUM(ELIGIBLE) AS ELIGIBLE LABEL='Eligible Sample Size',
            SUM(FIELDED) AS FIELDED LABEL='Fielded Sample Size',
            SUM(RESPONDED) AS RESPONDED LABEL='Respondent Sample Size',
            (SUM(RESPONDED)/SUM(FIELDED))*100 AS RESPRATE FORMAT=5.1 LABEL='Response Rate (%)',
            SUM(RESGRP='E' AND FIELDED) AS TFIELDED LABEL='Fielded Treatment Sample Size',
            SUM(RESGRP='C' AND FIELDED) AS CFIELDED LABEL='Fielded Control Sample Size'
FROM SURVEYTRCK
GROUP BY SITE
ORDER BY SITE;
COLUMNS 7 AND 9 (RESEARCH GROUP RATES): LOGICAL STATEMENTS

The research group response rates are just easy calculations of the number in each group who responded, divided by
the number in each group who were fielded. The report code is:

```sql
PROC SQL;
SELECT SITE AS SITE LABEL='Site',
     PUT(MIN(INTAKE),MMYYS7.)||"-"||PUT(MAX(INTAKE),MMYYS7.) AS INTAKE LABEL='Intake Period',
     SUM(ELIGIBLE) AS ELIGIBLE LABEL='Eligible Sample Size',
     SUM(FIELDED) AS FIELDED LABEL='Fielded Sample Size',
     SUM(RESPONDED) AS RESPONDED LABEL='Respondent Sample Size',
     (SUM(RESPONDED)/SUM(FIELDED))*100 AS RESPRATE FORMAT=5.1 LABEL='Response Rate (%)',
     SUM(RESGRP='E' AND FIELDED) AS TFIELDED LABEL='Fielded Treatment Sample Size',
     (SUM(RESGRP='E' AND RESPONDED)/SUM(RESGRP='E' AND FIELDED))*100 AS TRESPRATE FORMAT=5.1 LABEL='Treatment Response Rate (%)',
     SUM(RESGRP='C' AND FIELDED) AS CFIELDED LABEL='Fielded Control Sample Size',
     (SUM(RESGRP='C' AND RESPONDED)/SUM(RESGRP='C' AND FIELDED))*100 AS CRESPRATE FORMAT=5.1 LABEL='Control Response Rate (%)'
FROM SURVEYTRCK
GROUP BY SITE
ORDER BY SITE;
```

CHECKING THE RESULTS

Let's take a quick look at the table that we have built so far (compressed for presentation):

```
<table>
<thead>
<tr>
<th>Site</th>
<th>Intake Period</th>
<th>Eligible Sample Size</th>
<th>Fielded Sample Size</th>
<th>Respondent Sample Size</th>
<th>Fielded Treatment Sample Size</th>
<th>Fielded Control Sample Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01/2004-12/2004</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>50.0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>01/2004-12/2004</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>50.0</td>
<td>2</td>
</tr>
</tbody>
</table>
```

SUMMARY ROW

The SQL report is close to the prototype table but is missing the last/summary row. Let’s insert a summary row into
the data set. Using the example data above, let’s insert a row with SITE=99 (for total), ELIGIBLE=12, FIELDED=8,
and RESPONDED=4. With Proc SQL, creating a summary row is easy. Notice that the survey-tracking data set that
we have been using is a temporary data set with only six variables (SITE, INTAKE, ELIGIBLE, FIELDED,
RESPONDED, RESGRP), so creating a row that summarizes the data items shouldn’t be that involved. We can
simply copy and paste the “by site” select statement that we created above and modify a few things. The main
modification that we will make is deleting the GROUP BY statement. Without a GROUP BY statement, SAS will — by
default — create the summary numbers using everyone in the data set. It is important that the summary select
statement include all the same columns (variables) as the “by site” select statement. Also remember to preserve the
same order of variables:

```sql
SELECT 99 AS SITE,
     ' ' AS INTAKE,
     SUM(ELIGIBLE) AS ELIGIBLE,
     SUM(FIELDED) AS FIELDED,
     SUM(RESPONDED) AS RESPONDED,
     . AS RESPRATE,
     SUM(RESGRP='E' AND FIELDED) AS TFIELDED LABEL='Fielded Treatment Sample Size',
     . AS TRESPRATE,
     SUM(RESGRP='C' AND FIELDED) AS CFIELDED LABEL='Fielded Control Sample Size',
     . AS CRESPRATE
FROM SURVEYTRCK;
```

In the lines above, notice that we have used Proc SQL to fill in missing literals for columns for which we don’t want to
calculate any summary statistics.
COMBINING THE "BY SITE" RESULTS SET AND THE TOTALS RESULTS SET: UNION OPERATOR

Now we have two Proc SQL select statements. One select produces one summary line for each city in the survey, and the other select produces one line of total sample sizes. To concatenate results for these two select statements (which have the same structure and variable names), you can use the Union Operator in Proc SQL. The following statement concatenates the results from the two selects and prints the report:

```sql
PROC SQL;
SELECT SITE AS SITE LABEL='Site',
    PUT(MIN(INTAKE),MMYY7.)||'-'||PUT(MAX(INTAKE),MMYY7.) AS INTAKE LABEL='Intake Period',
    SUM(ELIGIBLE) AS ELIGIBLE LABEL='Eligible Sample Size',
    SUM(FIELDED) AS FIELDED LABEL='Fielded Sample Size',
    SUM(RESPONDED) AS RESPONDED LABEL='Respondent Sample Size',
    (SUM(RESPONDED)/SUM(FIELDED))*100 AS RESPRATE FORMAT=5.1 LABEL='Response Rate (%)',
    (SUM(RESGRP='E' AND RESPONDED)/SUM(RESGRP='E' AND FIELDED))*100 AS TRESPRATE
        FORMAT=5.1 LABEL='Treatment Response Rate (%)',
    (SUM(RESGRP='C' AND RESPONDED)/SUM(RESGRP='C' AND FIELDED))*100 AS CRESPRATE
        FORMAT=5.1 LABEL='Control Response Rate (%)'
FROM SURVEYTRCK
GROUP BY SITE
UNION CORR
    SELECT 99 AS SITE,
        ' ' AS INTAKE,
        SUM(ELIGIBLE) AS ELIGIBLE,
        SUM(FIELDED) AS FIELDED,
        SUM(RESPONDED) AS RESPONDED,
        . AS RESPRATE,
        . AS TRESPRATE,
        SUM(RESGRP='E' AND FIELDED) AS CFIELDED LABEL='Fielded Control Sample Size',
        . AS CRESPRATE
        FROM SURVEYTRCK
ORDER BY SITE;
```

FIELDED            Fielded            Fielded
Eligible Fielded Respondent          Treatment Treatment  Control  Control
Sample  Sample     Sample Response    Sample Response   Sample Response
Site Intake Period      Size    Size       Size Rate (%)      Size  Rate (%)     Size Rate (%)
──────────────────────────────────────────────────────────────────────────────────────────────
1 01/2004-12/2004       6       4          2     50.0         2      50.0        2     50.0
2 01/2004-12/2004       6       4          2     50.0         2      50.0        2     50.0
99                      12       8          4       .          4        .         4       .

SENDING THE REPORT TO EXCEL: ODS HTML DESTINATION

The easiest way to send a report like this to Excel is to use the ODS HTML destination. Excel reads HTML files without the need for any type of file-import wizard. Notice that an ODS statement has been added at the beginning of the programming code below. This sends the report to an HTML file. The style= clause at the end of the ODS statement specifies the table style template to be used. The table style template controls things like the font types and sizes used in Excel, the column widths, the frame around the table, and whether the table contains any shading, font colors, background colors, and so on. ODS has several style templates available. Refer to SAS’s online documentation for information about the ODS style templates available in each version of SAS. If you don’t like any of the ODS table styles available, you can create your own table style template and in this way control via SAS all the table attributes listed above and more. To create your own table style template, you use Proc Templates.

While sending the table to Excel, this seems like a good time to add a few finishing touches. First, let’s add a user-defined format for the site variable so that we can display the names of the survey cities instead of a numeric representation. Let’s also add titles to match the titles in the prototype table. Putting it all together, the following is the full code needed to create the report, format the report, and send the report to the ODS HTML destination:

```sql
ODS HTML FILE="SURVEYTRCK.HTML" STYLE=SASWEB;
RUN;
PROC FORMAT;
```
VALUE SITE
1='NYC'
2='OHIO'
99='TOTAL';

TITLE1 'Table 1';
TITLE2 'Project 12-Month Survey Sample and Response Rates';

PROC SQL;
SELECT SITE AS SITE FORMAT=SITE. LABEL='Site',
    PUT(MIN(INTAKE),MMYYS7.)||"-"||PUT(MAX(INTAKE),MMYYS7.) AS INTAKE LABEL='Intake Period',
    SUM(ELIGIBLE) AS ELIGIBLE LABEL='Eligible Sample Size',
    SUM(FIELDED) AS FIELDED LABEL='Fielded Sample Size',
    SUM(RESPONDED) AS RESPONDED LABEL='Respondent Sample Size',
    (SUM(RESPONDED)/SUM(FIELDED))*100 AS RESPRATE FORMAT=5.1 LABEL='Response Rate (%)',
    SUM(RESGRP='E' AND FIELDED) AS TFIELDED LABEL='Fielded Treatment Sample Size',
    (SUM(RESGRP='E' AND RESPONDED)/SUM(RESGRP='E' AND FIELDED))*100 AS TRESPRATE FORMAT=5.1 LABEL='Treatment Response Rate (%)',
    SUM(RESGRP='C' AND FIELDED) AS CFIELDED LABEL='Fielded Control Sample Size',
    (SUM(RESGRP='C' AND RESPONDED)/SUM(RESGRP='C' AND FIELDED))*100 AS CRESPRATE FORMAT=5.1 LABEL='Control Response Rate (%)'
FROM SURVEYTRCK
GROUP BY SITE
UNION CORR
SELECT 99 AS SITE,
    ' ' AS INTAKE,
    SUM(ELIGIBLE) AS ELIGIBLE,
    SUM(FIELDED) AS FIELDED,
    SUM(RESPONDED) AS RESPONDED,
    . AS RESPRATE,
    SUM(RESGRP='E' AND FIELDED) AS TFIELDED LABEL='Fielded Treatment Sample Size',
    . AS TRESPRATE,
    SUM(RESGRP='C' AND FIELDED) AS CFIELDED LABEL='Fielded Control Sample Size',
    . AS CRESPRATE
FROM SURVEYTRCK
ORDER BY SITE;
QUIT;
RUN;

ODS HTML CLOSE;

Opening the file SURVEYTRCK.HTML in Excel will display the following table:

<table>
<thead>
<tr>
<th>Site</th>
<th>Intake Period</th>
<th>Eligible Sample Size</th>
<th>Fielded Sample Size</th>
<th>Respondent Sample Size</th>
<th>Response Rate (%)</th>
<th>Fielded Treatment Sample Size</th>
<th>Treatment Response Rate (%)</th>
<th>Fielded Control Sample Size</th>
<th>Control Response Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NYC</td>
<td>01/2004-12/2004</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>50.0</td>
<td>2</td>
<td>50.0</td>
<td>2</td>
<td>50.0</td>
</tr>
<tr>
<td>OHIO</td>
<td>01/2004-12/2004</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>50.0</td>
<td>2</td>
<td>50.0</td>
<td>2</td>
<td>50.0</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>12</td>
<td>8</td>
<td>4</td>
<td>.</td>
<td>4</td>
<td>.</td>
<td>4</td>
<td>.</td>
</tr>
</tbody>
</table>

COMPARING PROC SQL TO OTHER TABLING PROCEDURES
Space limitations prevent extensive discussion of the syntax required to produce the same table using Proc Tabulate or Proc Report. However, in the interest of completeness, the Proc Tabulate and Proc Report syntax and the resulting tables are displayed below. In comparison, the level of programming complexity for the three methods is relatively
similar. Further, the appearance of the tables produced by each of these methods is very similar. Therefore, the
decision about which method to use to produce the desired table comes down to personal preference.

**PROC TABULATE**

Let's start by first reviewing the Proc Tabulate code and table:

```
PROC FORMAT;
VALUE SITE
1='NYC'
2='OHIO' ;

DATA SURVEYTRCK;
SET SURVEYTRCK;
TFIELDED=RESGRP='E' AND FIELD=1;
CFIELDED=RESGRP='C' AND FIELD=1;
TRESPONDED=RESGRP='E' AND RESPONSE=1;
CRESPONDED=RESGRP='C' AND RESPONSE=1;

LABEL
SITE='Site'
INTAKE='Intake Period'
ELIGIBLE='Eligible Sample Size'
FIELD='Fielded Sample Size'
RESPONSE='Respondent Sample Size'
TFIELD='Fielded Treatment Sample Size'
CFIELD='Fielded Control Sample Size';
RUN;

ODS HTML FILE="SURVEYTRKTAB.HTML" STYLE=SASWEB;
RUN;

TITLE;
TITLE2 'Table 1';
TITLE3 'Project 12-Month Survey Sample and Response Rates';
PROC TABULATE DATA= SURVEYTRCK;
CLASS SITE;
VAR INTAKE ELIGIBLE FIELD RESPONSE TFIELD CFIELD TRESPONSE CRESPONSE;
TABLE SITE ALL='Total',
INTAKE*(MIN=F=MMYYS7. MAX=F=MMYYS7.)
ELIGIBLE*SUM
FIELD*SUM
RESPONSE*SUM
RESPONSE=' 'PCTSUM<FIELD>=’Response Rate (%)' 
TFIELD*SUM
TRESPONSE=' 'PCTSUM<TFIELD>=’Treatment Response Rate (%)' 
CFIELD*SUM
CRESPONSE=' 'PCTSUM<CFIELD>=’Control Response Rate (%)' ;
FORMAT SITE SITE.;
KEYLABEL
MIN= '
MAX= '
SUM= '
PCTSUM=' '
RUN;
ODS HTML CLOSE;
RUN;
```
Table 1
Project 12-Month Survey Sample and Response Rates

<table>
<thead>
<tr>
<th>Site</th>
<th>Intake Period</th>
<th>Eligible Sample Size</th>
<th>Fielded Sample Size</th>
<th>Respondent Sample Size</th>
<th>Response Rate (%)</th>
<th>Fielded Treatment Sample Size</th>
<th>Treatment Response Rate (%)</th>
<th>Fielded Control Sample Size</th>
<th>Control Response Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;NYC&quot;</td>
<td>01/2004</td>
<td>6.00</td>
<td>4.00</td>
<td>2.00</td>
<td>50.00</td>
<td>2.00</td>
<td>50.00</td>
<td>2.00</td>
<td>50.00</td>
</tr>
<tr>
<td>&quot;OHIO&quot;</td>
<td>01/2004</td>
<td>6.00</td>
<td>4.00</td>
<td>2.00</td>
<td>50.00</td>
<td>2.00</td>
<td>50.00</td>
<td>2.00</td>
<td>50.00</td>
</tr>
<tr>
<td>Total</td>
<td>01/2004</td>
<td>12.00</td>
<td>8.00</td>
<td>4.00</td>
<td>50.00</td>
<td>4.00</td>
<td>50.00</td>
<td>4.00</td>
<td>50.00</td>
</tr>
</tbody>
</table>

PROC REPORT
The table produced by Proc Report (below) looks the most like the prototype table we were given. Proc report is the only report-production tool that is able to produce the nested column headings of the prototype table. Note that the site variable needed to be converted to a character variable for the table summarization to work. The Proc Report code and resulting table follow:

```sas
PROC FORMAT;
VALUE $SITE '1'='NYC'
   '2'='OHIO'
   '9'='TOTAL' ;

data SURVEYTRCK;
set SURVEYTRCK;
XSITE=PUT(SITE,1.);
TFIELDED=RESGRP='E' AND FIELDED=1;
CFIELDED=RESGRP='C' AND FIELDED=1;
TRESPONDED=RESGRP='E' AND RESPONDED=1;
CRESPONDED=RESGRP='C' AND RESPONDED=1;
label
XSITE='Site'
ELIGIBLE='Eligible Sample Size'
FIELDED='Fielded Sample Size'
RESPONDED='Respondent Sample Size'
TFIELDED='Fielded Sample Size'
CFIELDED='Fielded Sample Size';
RUN;
ODS HTML FILE="SURVEYTRKREP.HTML" STYLE=SASWEB;
run;
title2 'Table 1';
title3 'Project 12-Month Survey Sample and Response Rates';
PROC REPORT DATA=SURVEYTRCK NOWD;
COLUMN XSITE INTAKE,(MIN MAX) ELIGIBLE FIELDED RESPONDED RESRATE
('Treatment Group' TFIELDED TRESPONDED TRESRATE)
('Control Group' CFIELDED CRESPONDED CRESRATE);
DEFINE XSITE /GROUP FORMAT=$SITE.;
DEFINE INTAKE/ANALYSIS 'Intake Period';
DEFINE MIN /FORMAT=MMYYS7. '';
DEFINE MAX /FORMAT=MMYYS7. '';
DEFINE ELIGIBLE /ANALYSIS SUM;
DEFINE FIELDED /ANALYSIS SUM;
DEFINE RESPONDED /ANALYSIS SUM WIDTH=10;
DEFINE RESRATE /COMPUTED FORMAT=5.1 'Response Rate (%)' WIDTH=10;
DEFINE TFIELDED /ANALYSIS SUM;
DEFINE TRESPONDED /ANALYSIS SUM NOPRINT;
DEFINE TRESRATE /COMPUTED FORMAT=5.1 'Response Rate (%)' WIDTH=10;
DEFINE CFIELDED /ANALYSIS SUM;
DEFINE CRESPONDED /ANALYSIS SUM NOPRINT;
DEFINE CRESRATE /COMPUTED FORMAT=5.1 'Response Rate (%)' WIDTH=10;

COMPUTE RESRATE;
RESRATE=100*(RESPONDED.SUM/FIELDED.SUM);
ENDCOMP;

COMPUTE TRESRATE;
TRESRATE=100*(TRESPONDED.SUM/TFIELDED.SUM);
ENDCOMP;

COMPUTE CRESRATE;
CRESRATE=100*(CRESPONDED.SUM/CFIELDED.SUM);
ENDCOMP;

RBREAK AFTER /SUMMARIZE;
COMPUTE AFTER;
XSITE='9';
ENDCOMP;
RUN;

Table 1

<table>
<thead>
<tr>
<th>Site</th>
<th>Intake Period</th>
<th>Eligible Sample Size</th>
<th>Fielded Sample Size</th>
<th>Respondent Sample Size</th>
<th>Response Rate (%)</th>
<th>Fielded Sample Size</th>
<th>Response Rate (%)</th>
<th>Fielded Sample Size</th>
<th>Response Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NYC</td>
<td>01/2004-12/2004</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>50.0</td>
<td>2</td>
<td>50.0</td>
<td>2</td>
<td>50.0</td>
</tr>
<tr>
<td>OHIO</td>
<td>01/2004-12/2004</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>50.0</td>
<td>2</td>
<td>50.0</td>
<td>2</td>
<td>50.0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>01/2004-12/2004</td>
<td>12</td>
<td>8</td>
<td>4</td>
<td>50.0</td>
<td>4</td>
<td>50.0</td>
<td>4</td>
<td>50.0</td>
</tr>
</tbody>
</table>

CONCLUSION

Proc SQL's computational abilities are similar to those found in the data step. Proc SQL summarization abilities are similar to those of Proc Means, Summary, Tabulate, or Report. In Proc SQL, a full range of standard SAS functions is available to manipulate your data. Although not mentioned in this paper, IF/ELSE processing is also possible with the Proc SQL Case clause. A good deal of control over the look and formatting of the output is available with the format and label clauses of Proc SQL, and Proc SQL allows you to utilize all the formatting features of ODS. In comparison with the other tabling tools available in SAS (Proc Tabulate and Proc Report), the level of programming complexity required to produce the prototype table was relatively similar. Further, the appearance of the table produced by Proc SQL is very similar to the tables produced by Proc Tabulate and Proc Report. All these features combine to make Proc SQL an important report-production tool.

CONTACT INFORMATION

Electra Small
Senior Technical Associate II
MDRC
212-340-8824 (Direct)
16 East 34th Street, 19th Floor
New York, NY 10016
Electra.Small@mdrc.org

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.