

Increasing Student Progress Monitoring via PROC REPORT and ExcelXP Tagset

Elayne Reiss, Seminole County Public Schools, Sanford, FL

ABSTRACT

This paper is geared toward the SAS® user who has some experience with general reporting techniques in Base SAS but has not explored the deeper options. It tackles the issue of how to mass-generate sleek, detailed, multi-tabbed Excel reports using the right combination of the many options provided by PROC REPORT, the SAS ExcelXP Tagset, and macro programming. Some recent changes to the ExcelXP Tagset (as of June 2007) will be highlighted in the discussion. Although the paper's example involves the creation of student progress monitoring reports for every teacher in a public K-12 school district, the coding techniques contained within are highly applicable to report -building processes found in almost any industry.

INTRODUCTION

Considering the popularity of accountability for student performance in the K-12 education system, end users (teachers, principals, and district administrators) frequently request ad-hoc reports from district analysts to obtain a better grasp on how well students have performed on yearly state-mandated standardized assessments. Although the level of data manipulation expertise among end users may vary from person to person, almost all agree on the helpfulness of the Excel workbook. PDF reports are generally a wise choice for most applications, but when users want to view their own data in different orders and with different combinations of variables, they need a format in which they can manipulate the information themselves. In a school system or any other setting where resources for detailed online report distribution systems are often limited, it is essential to have the ability to still create high-quality, user-friendly reports that can serve a wide cross-section of end users.

Suppose that you (the analyst) have your data neatly prepared in SAS, but how can you transform that large database into individualized and informative reports for the entire school district (or company), considering your restraints? All of the tools you need are available within the Base SAS product: PROC REPORT, the ExcelXP Tagset, and macro programming. This paper will explore the nuances of how to join the many features of PROC REPORT (such as compute blocks and individualized column preferences) with an array of options in ExcelXP (such as sheet naming and labeling) to create distribution-ready reports requiring practically no post-processing prior to distribution.

ABOUT THE DATASET

While this paper will focus on the tips and tricks necessary to generate and format physical reports rather than data setup, it is still helpful to briefly learn about the dataset that will be used in this example.

	schoolnum	TCHNAME	pernum	LASTNAME	FIRSTNAME	grade	pr1_grd	gainr	rprof	mastr	pr1_mastr	dssr	pr1_dssr	ethnic	frl	priexp	lep	stdcu
36	652	CURTIN	9273016	MUNOZ	MARIA	4	3	1	1	4	4	1795	1634	W	0		ZZ	1
37	652	CURTIN	8616213	ROSENFELD	PETER	4	3	1	1	4	4	1759	1755	W	0		ZZ	1
38	652	CURTIN	6049192	SMITH	ROGER	4	3	0	1	3	4	1630	1676	W	0		ZZ	1
39	652	CURTIN	5465378	VIDAL	ANDRE	4	3	1	1	4	3	1695	1476	W	0		ZZ	1
40	652	CURTIN	4829166	WEGMAN	BENJIMAN	4	3	1	1	4	4	1923	1792	M	0	L	ZZ	1
41	652	HOFFMAN	5326839	ANDERSON	SAMANTHA	4	3	0	1	3	4	1660	1640	W	0		ZZ	1
42	652	HOFFMAN	8953869	BROOKS	RICKY	4	3	1	1	5	5	2275	2514	W	0	L	ZZ	1
43	652	HOFFMAN	5115451	BROWN	AARON	4	3	0	1	3	4	1601	1561	W	0		ZZ	1
44	652	HOFFMAN	6374303	CAMPBELL	JOHN	4	3	0	1	3	4	1595	1543	W	0		ZZ	1
45	652	HOFFMAN	5687439	CLARK	EVAN	4	3	1	1	4	4	1935	1695	W	0		ZZ	1
46	652	HOFFMAN	5399651	DAVIS	CHRISTINA	4	3	1	1	4	4	1877	1798	B	0	L	ZZ	1
47	652	HOFFMAN	5544206	DOUGLAS	DANIEL	4	3	0	0	2	2	1361	1185	W	0		ZZ	1
48	652	HOFFMAN	4165440	FREEMAN	CHRISTOPHER	4	3	1	1	3	3	1648	1446	W	0		ZZ	1
49	652	HOFFMAN	4687692	HALL	TODD	4	3	1	1	3	2	1578	1106	B	1	L	ZZ	1
50	652	HOFFMAN	5157037	KLEIN	CLAYTON	4	3	0	0	1	2	1179	1051	W	0		ZZ	1
51	652	HOFFMAN	7851593	LEWIS	DOMINICK	4	3	1	1	5	4	2070	1822	W	0		ZZ	1
52	652	HOFFMAN	5493979	MEYERS	JAMES	4	3	0	1	3	4	1648	1628	W	0		ZZ	1
53	652	HOFFMAN	3250317	MITCHELL	ANDREW	4	3	1	1	3	3	1478	1476	W	0		ZZ	1

Figure 1: The Dataset

The dataset we will use for the example (Figure 1) contains performance and demographic statistics for every elementary student in the school district who took at least two consecutive administrations of the statewide reading assessment test. These demographic items include school number, teacher name, student name, grade, test scores for both years, and whether or not the student made at least a year's worth of academic progress in reading, known

as a “learning gain.” The dataset also includes the percentile rank in which each student scored, since special attention must be given to those students scoring at or below the 25th percentile among their classmates at their respective schools.

The goal of reporting this data is to provide a single Excel file for each school containing separate lists of students and their performance scores for each teacher. The files also need a level of visual appeal in order for teachers and principals to feel “invited” to utilize the information. Although the lack of grouping or computed variables in these roster lists may suggest that this task is properly suited for PROC PRINT, PROC REPORT will be used to allow for simplified code modification in future expansions of this report.

FIRST STEPS: A STYLE-FREE PROC REPORT

First, we will examine the basics of the PROC REPORT code that will be used to generate these tables. Later, we will explore how specifying the ExcelXP Tagset as a destination can completely transform an otherwise mundane roster into a completely distributable information source.

Making a Simple Report

As stated earlier, the information itself is almost simple enough to be displayed through PROC PRINT. The PROC REPORT procedure looks like the following:

```
proc report data = read nowd split='*';  
  by schoolnum tchname;  
  column permnum lastname firstname grade pr1_grd low25r dssrank_r gainr mastr  
         pr1_mastr dssr pr1_dssr ethnic frl priexp lep;  
  define permnum / 'PermNum';  
  define lastname / 'Last ';  
  define firstname / 'First' ;  
  define low25r / 'Low*Quartile' format=ynfmt.;  
  ...  
  <additional definitions follow>  
run;
```

Figure 2: Simple PROC REPORT Code

This handful of defined column headings and occasional variable formats does not produce readily usable output. The reports are generated by school and teacher, but lack visual appeal. Otherwise, the structure of the table is acceptable—we simply need to improve the aesthetic and distribution issues.

Simple Report Meets ExcelXP

Our task is to turn the simple report into an easily accessible (and still editable) finished product. Without adding other options to the “simple” PROC REPORT code from Figure 2, here’s how to turn the simple report into a slightly more improved version.

First, the PROC REPORT code from Figure 2 needs some Output Delivery System (ODS) statements that will tell SAS to send the report to an Excel workbook. To begin writing output to ODS, simply place the following statements before PROC REPORT:

```
ods listing close;      /*closes basic SAS "listing" output*/  
ods tagsets.ExcelXP     /*specifies use of ExcelXP Tagset*/  
  path = 'C:\My SAS Files\SESUG' /*common filepath*/  
  file = 'filename.xls'  /*specific filename*/
```

To close the ODS destination, place **ods tagsets.ExcelXP close** after the PROC REPORT code, followed by **ods listing** to re-open the regular SAS listing output destination. When we add these few lines to the code, our report begins to take some shape.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	TCHN													
2	permn	LAST	FIRST	grade	pr1_g	low25	dssra	gainr	mastr	pr1_	dssr	pr1_d	ethnic	fri
3	5326839	ANDER	SAMAN	4	3	0	66.40%	0	3	4	1660	1640	W	0
4	8953869	BROOK	RICKY	4	3	0	98.60%	1	5	5	2275	2514	W	0
5	5115451	BROW	AARON	4	3	0	51.70%	0	3	4	1601	1561	W	0
6	6374303	CAMP	JOHN	4	3	0	48.30%	0	3	4	1595	1543	W	0
7	5687439	CLARK	EVAN	4	3	0	72.00%	1	4	4	1935	1695	W	0
8	5399651	DAVIS	CHRIST	4	3	0	85.30%	1	4	4	1877	1798	B	0
9	5544206	DOUGL	DANIEL	4	3	1	9.80%	0	2	2	1361	1185	W	0
10	4165440	FREEM	CHRIST	4	3	0	32.90%	1	3	3	1648	1446	W	0
11	4687692	HALL	TODD	4	3	1	7.00%	1	3	2	1578	1106	B	1
12	5157037	KLEIN	CLAYT	4	3	1	6.30%	0	1	2	1179	1051	W	0
13	7851593	LEWIS	DOMINI	4	3	0	90.20%	1	5	4	2070	1822	W	0
14	5493979	MEYER	JAMES	4	3	0	64.30%	0	3	4	1648	1628	W	0
15	3250317	MITCH	ANDRE	4	3	0	37.80%	1	3	3	1478	1476	W	0
16	4968153	ORTIZ	KELSE	4	3	0	49.00%	1	4	4	1847	1549	H	0
17	1544647	PARKE	MATTH	4	3	0	92.30%	1	4	4	1918	1853	W	0
18	5591743	PERRY	BRAND	4	3	0	95.80%	0	4	5	1795	1938	W	0
19	5444228	PHILLIP	MELIN	4	3	0	28.00%	0	2	3	1338	1415	W	3
20	9548449	PREST	MORG	4	3	1	17.50%	0	1	3	1302	1306	W	0
21	5345474	RUSSE	KATIE	4	3	0	67.10%	0	3	4	1677	1652	W	0
22	5644796	SCHMI	CORY	4	3	1	24.50%	0	2	3	1443	1391	W	0
23	5544749	SONG	TARA	4	3	1	16.10%	1	5	3	1970	1300	A	0
24	5417482	STEW	EMILIA	4	3	0	37.10%	1	3	3	1566	1470	W	0

Figure 3: Unformatted ODS Output

Since no particular style was specified, SAS automatically uses a default ODS style. The Figure 3 results are clearly superior to those you would obtain from listing output, but still have some stylistic issues to overcome. There are too many columns to fit into the report to keep such a large font size, Excel placed some superfluous trailing zeroes on each percentile value, some columns can use some centering and resizing...and the gray background is a bit drab for the taste of most educators. However, before we discover how some well-planned style options can transform this report, we must still resolve the issue of how to mass-produce these reports for each teacher and school.

Macro Programming: So Many Files, So Many Tabs!

By default, the ExcelXP Tagset will automatically create a separate table for each by-group if no options are specified. This functionality works perfectly for this case, but we would like to create separate files for each school, instead of having one giant multi-tabbed file containing all schools. A simple macro fix can resolve this issue:

```
%macro bysch_report (sch, schname); /*sch = sch. number, schname = school name*/
    proc sql; /*create a subset of the main table with a single school's data*/
        create table read as
            select *
            from sesug.elem_read where schoolnum = "&sch";
    quit;
    <previously stated PROC REPORT code>
%mend;
```

Figure 4: The Macro

This simple combination of statements (Figure 4) will allow us to create reports for as many or as few schools as we wish, using macro calls such as **%bsch_report (652, Goodnight Elementary);** The existing PROC REPORT code from the "simple" version does not need to change, aside from the specification of file name in the ExcelXP options. An easy way to uniquely identify each file by school is to adopt the school number as part of the file name. For example, the file definition of **file = "low25read_&sch.xls"** would resolve to "low25read_652.xls" for Goodnight Elementary. Notice how the statement featured two consecutive periods. The second one takes care of the period naturally found between the filename and extension, but the first one serves as a

separator between the macro variable **&sch** and any of the subsequent text. Without that period, SAS would try to resolve the nonexistent macro variable **&sch.xls**.

HOW ABOUT A LITTLE STYLE?

When utilizing stylistic elements that are likely to be reused in other reports, such as font types, font sizes, colors, and margins, it is a good idea to work with PROC TEMPLATE to create definitions for these features in a single procedure. For this application, we will begin with the “Printer” style as a base and build from there. PROC TEMPLATE can seem a bit intimidating at first for the novice to navigate, but a helpful place to begin when planning to edit style definitions is with the code of the parent definition. This definition will appear in the log by entering the following code:

```
proc template;
    source styles.Printer;

run;
```

A view of the long list of elements reveals several items of interest: fonts are mostly Times New Roman 10-point; the margins (found in Body) are set to use procedurally specified margins; and colors include shades of gray, black, and white. For this report, we would like to use mostly 8-point Helvetica, pre-coded margins, and a base of strictly black and white. Figure 5 demonstrates how to make these changes:

```
proc template;

    define style Styles.SESUG; /*create a style definition from Printer*/
    parent = Styles.Printer;

    replace Body from Document / /*change margins (leave top margin alone)*/
        bottommargin = 0.5in topmargin = _undef_
        rightmargin = 0.25in leftmargin = 0.25in;

    replace fonts / /*Smaller Helvetica is preferred.*/
        'TitleFont2' = ("Helvetica", 10pt)
        'docFont' = ("Helvetica", 8pt) /* Data in table cells */
        <more font definitions>

    replace color_list / /*Makes color references easier later.*/
        "white" = cxxxxxxx
        "black" = cx000000;

    replace colors / /*Turn everything black and white.*/
        "docbg" = color_list("white")
        "docfg" = color_list("black")
        <more color definitions>

    end;

run;
```

Figure 5: Style Definitions Using a Template

Application of this new style definition, “SESUG,” simply works as follows:

```
ods tagsets.ExcelXP
```

```
path = 'C:\My SAS Files\SESUG' file = "low25read_&sch..xls" style=SESUG;
```

The table will now appear in black and white 8-point Helvetica with pre-defined margins.

FINE-TUNING THE OPTIONS

Aside from the inherent multi-tabbed functionality provided by the ExcelXP Tagset, most of the options that have been displayed thus far are compatible with any ODS output destination. This next section will showcase how some of the ExcelXP-specific options work and how to use the common PROC REPORT options in conjunction to add more depth to these reports.

Titles

Titles are simple, right? Whatever text contained in quotes following a TITLE statement appears at the top of your printout. Style definitions, as described in the last section, even allow you to easily change the appearance of those titles in ODS output by specifying different font faces, weights, and sizes. That flexibility still exists within the ExcelXP Tagset, but with some notable caveats.

In this example, we would like to place the school name (one of the two parameters specified in the macro call) and the teacher name (our only by-variable in the finished version of the table) in the title of each sheet, so that when the users print out all of the sheets in the workbook, they can easily identify each class. The following code in the PROC REPORT would produce those results:

```
title "2007 FCAT Reading Gains Student List";
```

```
title2 "&schname.,Teacher: #byval(tchname)";
```

Adding the NOBYLINE option would produce cleaner output by eliminating the default byline from the body of each page. In Excel, using this code without any other specified options would produce the desired output in the header of each page. However, remember those ODS style options? Because the ExcelXP Tagset is written in XML, any font size and weight changes written in the PROC TEMPLATE style definitions do not apply to any TITLE or FOOTNOTE statements that appear in the Excel header or footer. As a result of that coding issue, the ExcelXP Tagset includes the **print_header** and **print_footer** options. Instead of specifying your titles and footnotes the usual way, you can place text within those options and use XML to change the font size or weight. We would like our title to appear in 11-point Bold Helvetica and our footnote to appear in 8-point Helvetica, so we would use the following code in our ODS options:

```
ods tagsets.ExcelXP options
```

```
(print_header='&"Helvetica,Bold"&11 2007 FCAT Reading  
Gains Student List' print_footer='&8 SCPS-ER-Data Analysis &D');
```

The **"** statements are used in place of quotation marks within the XML language and the **&mp** statements specify each additional style element. The use of **&D** in the **print_footer** definition will place today's date at the end of the footnote. More of these elements are found within the ExcelXP Tagset help, which can be accessed by specifying **doc='help'** within the options definition.

We have solved most of the header and footer issues, but what about the school and teacher name? In order to have that unique information displayed on each page, we need to be able to reference both the macro variable for school name and the by-value of teacher name. Placing that information within the **print_header** definition will not work, since Excel is expecting XML code within the option definition, not SAS code associated with a **#byval** statement. Therefore, we need to compromise a bit. Using the ExcelXP option of **embedded_titles='yes'** forces any text in the TITLE statements to be placed within the body of the table, not in the header. However, it does not interfere with any text defined in the **print_header** option, so you can have headers with style *in addition to* within-body titles through SAS macro variable and by-value calls on the same page. The results are displayed in Figure 8.

Worksheet Tab Labels

Part of the ease of generating tables into multiple worksheets within a single workbook is the ability to navigate quickly through the formatted data. The default worksheet labels on each tab of an ExcelXP-generated workbook, however, can leave a bit to be desired. The ExcelXP Tagset provides the finished workbook with different worksheet labels depending on the SAS procedure used. Default PROC REPORT labels follow the pattern of "Table X – Detailed and or Summa...", where X represents the number of the table. For any end user, this system does not provide friendly navigation of worksheets. Fortunately, the ExcelXP Tagset includes the **sheet_label** option, where a definition of **sheet_label='Teacher'** would yield tabs labeled "Teacher", "Teacher 2", etc. However, asking a school principal to locate Teacher Smith in a list of tabs where teachers are only identified by a number will not result in much praise for user-friendliness. Using a statement such as **sheet_label='#byval(tchname)'** will not work either, due to Excel once

again expecting XML language, not SAS code. Thanks to an update in the March 2007 version of the tagset, we can resolve that issue!

```
ods tagsets.ExcelXP options (sheet_interval='bygroup' sheet_label=" "
suppress_bylines='yes');
```

Although the ExcelXP Tagset places each by-group table in separate worksheets by default, the combination of **sheet_interval='bygroup'** and **sheet_label=""** will now place the value of each by-group into the worksheet names. The use of **suppress_bylines='yes'** needs to be used instead of **options nobyline** to remove the byline text from the body of the report, due to a behind-the-scenes technical issue beyond the scope of this paper. Schools can now receive their files complete with tabs containing each teacher's name instead of a number. The results of this code are displayed in Figure 6.



Figure 6: By-Teacher Tab Labels

Other Formatting Issues

The table displayed in Figure 3 faced some additional formatting issues, such as inadequate column widths and improperly formatted percentages. Unlike the previously mentioned formatting problems solved by the Tagset options, these items will be resolved through DEFINE statements on a by-variable basis within the PROC REPORT code.

For example, let's examine the definition of the variable `dssrank_r`, which displays each student's percentile rank. If you find the variable's column in Figure 1, you will notice that it was formatted in the dataset as a percentage with one decimal place. However, Figure 3 showed that once the variable reached Excel, zeroes appeared at the end of each figure to align with Excel's default percentage setting of two decimal places. The ExcelXP Tagset allows the user to place Excel formatting commands within the SAS DEFINE statements to fine-tune how variables are displayed in the finished product. To fix the `dssrank_r` issue, we will use the following line of code within PROC REPORT:

```
define dssrank_r / 'Percentile' style(column)={tagattr='format:0.0%'
cellwidth=66px};
```

Those of you who are otherwise familiar with PROC REPORT should recognize the **style(column)** option. The **tagattr** option is the function of the ExcelXP Tagset that allows you to explicitly define formatting through XML. We would like a percentage with one decimal place, so the code snippet above will achieve that effect. As for the **cellwidth** option, it is probably a good idea to run a sample report in Excel first, find the column widths you desire, and edit them as needed thereafter in the SAS code. Setting the columns a few pixels wider than the exact value you desire tends to work best.

Options defined in the ODS definition to make the report formatting more complete include:

- **orientation='landscape'** : Setting the page orientation with SAS OPTIONS will not work.
- **center_horizontal='yes'** : If the report does not take up the whole width of the page, it will now be centered horizontally for printing.
- **row_repeat='header'** : When printing, the first row will show up on each page for multi-paged reports, thus enhancing readability.
- **autofilter='all'** : Using the electronic format, users may want to view only selected subgroups of students based on a demographic value. This option will automatically apply the Excel AutoFilter to all columns, making that feature more enticing for the end user.
- **missing_align='center'** : Even if the rest of the column is centered, missing values will still show up in Excel as right-aligned. This option allows you to override that occurrence to provide a more uniform look.

A SPLASH OF COLOR

As described in the Introduction section, the goal of this particular report is to make principals and teachers more aware of which students in each class are progressing in reading performance, especially among those scoring below the 25th percentile. Using color to highlight students in different achievement categories can effectively draw attention when the end users view the reports. This section will briefly discuss the coding intricacies involved in “traffic lighting” using PROC REPORT.

The COMPUTE block in PROC REPORT allows us to use if-then programming in conjunction with CALL DEFINE statements to specify certain conditions under which the procedure should fill a row with a particular color. In this example, no variables are defined as GROUP or ORDER. As a result, we will have to rule out using COMPUTE with a location (i.e. COMPUTE BEFORE or COMPUTE AFTER), since the “location variable” must be defined as GROUP or ORDER. Without this usage option, we can only perform the traffic lighting function through the computation of an actual variable. Although we do not have any variables to compute in a numerical sense, such as a mean or sum, we can still compute a non-displayed dummy variable, which we will call “c.”

The computations in this example involve three variables: low25r (yes/no to membership in the lowest quartile), gainr (yes/no to making a learning gain in reading), and dssrank_r (percentile rank). In order for the if-then programming within this COMPUTE block to work, we *must* specify these three referenced variables as DISPLAY variables (since they are not GROUP, ORDER, or COMPUTED). Our COMPUTE block will resemble the following:

```
define c / noprint;
compute c;
    if low25r = 1 & gainr = 0 then call define (_row_, "style", "style=[background = CXFF99FF]");
    else if low25r = 0 & gainr = 0 then call define (_row_, "style", "style=[background = CXFFFF66]");
    else if low25r = 1 & gainr = 1 then call define (_row_, "style", "style=[background = CX99CC00]");
    else if low25r = . then call define (_row_, "style", "style=[background = CX777777]");
    if .255 <= dssrank_r < .305 then call define ("dssrank_r", "style", "style=[background= CX99CCFF]");
endcomp;
```

Figure 7: COMPUTE Block for Color

For the code in Figure 7 to work properly, the dummy variable must be defined to the right of any variable within the COMPUTE block. Otherwise, SAS will treat the other three variables as undefined. Note how the first four if-then statements result in a CALL DEFINE for _ROW_ and the last statement involves a single variable. When using CALL DEFINE for stylistic elements, you can set definitions for an entire row/column or just a single cell. Figure 8 showcases the final results of this code.

CONCLUSION

Through PROC REPORT, the ExcelXP Tagset, style templates, and macro programming, we have turned a plain dataset into a user-friendly and aesthetically pleasing Excel workbook for easy distribution. Figure 8 displays an example of a final printed report. The titles take advantage of both Excel-based style specifications and SAS-based macro and by-group coding. Column widths are appropriate for each variable. The percentages are displayed with the desired number of decimal places. Font faces and sizes accommodate the amount of data on the page. Colors direct the reader's attention to crucial levels of student achievement.

One word of caution involves the format of the finished file. As stated in various places throughout the paper, the ExcelXP Tagset actually generates an XML file, not an XLS file. If you generate a file with many (i.e. over 50) worksheets and open it in Excel, you may find yourself waiting for over 10 minutes for the file to fully open. XML workbooks tend to be much larger in file size (and slower to open) than XLS files, but a subsequent “Save As XLS” will save you from spending the same amount of opening time in the future. This step represents the only real post-processing required in the project but represents a relatively small sacrifice for many huge benefits.

The concept of using an ODS Tagset or a macro may seem a bit intimidating at first to a beginner, but if you take the time to explore the option lists and keep an open mind to the possibilities, you'll be amazed by the volume of quality reports you can create and customize in a quick, efficient manner. The end users will certainly appreciate your work!

2007 FCAT Reading Gains Student List															
Goodnight Elementary, Teacher: HOFFMAN															
Pemidum	Last	First	Grade 2007	Grade 2006	Low Quartile	Percentile	Gain 06-07	Mastery 2007	Mastery 2006	DSS 2007	DSS 2006	Ethnicity	FRL	ESE	ESOL
5326839	ANDERSON	SAMANTHA	4	3	No	66.4%	No	3	4	1660	1640	W			
8953869	BROOKS	RICKY	4	3	No	98.6%	Yes	5	5	2275	2514	W		L	
5115451	BROWN	AARON	4	3	No	51.7%	No	3	4	1601	1561	W			
6374303	CAMP BELL	JOHN	4	3	No	48.3%	No	3	4	1595	1543	W			
5687439	CLARK	EVAN	4	3	No	72.0%	Yes	4	4	1935	1695	W			
5399651	DAVIS	CHRISTINA	4	3	No	85.3%	Yes	4	4	1877	1798	B		L	
5544206	DOUGLAS	DANIEL	4	3	Yes	9.8%	No	2	2	1361	1185	W			
4165440	FREEMAN	CHRISTOPHER	4	3	No	32.9%	Yes	3	3	1648	1446	W			
4687692	HALL	TODD	4	3	Yes	7.0%	Yes	3	2	1578	1106	B	FRL	L	
5157037	KLEIN	CLAYTON	4	3	Yes	6.3%	No	1	2	1179	1051	W			
7851593	LEVIS	DOMINICK	4	3	No	90.2%	Yes	5	4	2070	1822	W			
5493979	MEYERS	JAMES	4	3	No	64.3%	No	3	4	1648	1628	W			
3250317	MITCHELL	ANDREW	4	3	No	37.8%	Yes	3	3	1478	1476	W			
4968153	ORTIZ	KELSEY	4	3	No	49.0%	Yes	4	4	1847	1549	H			
1544647	PARKER	MATTHEW	4	3	No	92.3%	Yes	4	4	1918	1853	W			
5591743	PERRY	BRANDIE	4	3	No	95.8%	No	4	5	1795	1938	W		L	
5444228	PHILLIPS	MELINDA	4	3	No	28.0%	No	2	3	1338	1415	W	FRL		
9548449	PRESTON	MORGAN	4	3	Yes	17.5%	No	1	3	1302	1306	W			
5345474	RUSSELL	KATIE	4	3	No	67.1%	No	3	4	1677	1652	W			
5644796	SCHMIDT	CORY	4	3	Yes	24.5%	No	2	3	1443	1391	W		K	
5544749	SONG	TARA	4	3	Yes	16.1%	Yes	5	3	1970	1300	A			
5417482	STEWART	EMILIA	4	3	No	37.1%	Yes	3	3	1566	1470	W			

SCPS-ER-Data Analysis 7/18/2007

Figure 8: The Final Report (Printed)

ACKNOWLEDGMENTS

The author would like to acknowledge the many wonderful SAS Research and Development team members who not only wrote the ExcelXP Tagset code, but also graciously continue to answer related questions on the SAS Support ODS and Base Reporting Forum.

CONTACT INFORMATION

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

Elayne Reiss
 Seminole County Public Schools
 400 E. Lake Mary Blvd.
 Sanford, FL 32773-7127
 E-mail: Elayne_Reiss@scps.us
 Web: <http://www.scps.k12.fl.us>

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.