Customized Reports - Have It Your Way!

Deb Cassidy

ABSTRACT

This presentation will cover some examples to create a report just the way you need it. Two main areas will be covered. The first area will be manipulating the raw data to provide what you really need. The second area will focus on formatting the text within the report. Different methods will be explored to try to accomplish the same report. This exploration will discuss factors to help you decide which method may be best for your next custom report.

The examples covered will be creating an evaluation form and a letter to report the evaluation results to the individuals being evaluated. While this might sound simple, the actual requirements might throw you a few curves. For example, the letter needs to contain the individual's score, a group average and an overall average.

Some of the code shown will include the use of DATA _NULL_, PROC REPORT, ODS, and PROC MEANS as well as combining and splitting data into the variables you really need. Some of the methods shown will involve export data to Excel or Word for the final formatting. This presentation is intended as a overview of some of the many things you can do to create customized reports.

PAPER ORGANIZATION

This paper will show a section of code followed by comments on certain parts of that code. Some comments will be an explanation of what the code does while others will be notes as to why I wrote the code in a particular way.

Please note that spacing in the code was adjusted to fit the column width of this paper.

FIRST STEPS

The first step in creating a custom report is to get your data. Sounds simple, doesn't it? Well, if you are lucky enough to have SAS dataset, it is. In this example, data were from several different sources and were in a variety of formats.

My first file was actually a PROC PRINT created with a BY statement that had been saved as a text file. A sample is shown at the end of the paper.

```
data one;
infile 'f:\schedule_short.txt'
      missover pad;
input line $200.;
if index(line,"SUGI 27 Data")>0
                 then flag=1;
else if substr(line,1,1)='0' or
      substr(line,1,5)="     "
                 then flag=1;
else if substr(line,1,5)='PAPER' or
      substr(line,50,5)='-----'
                 then flag=1;
sct1=index(line,'Section=');
day1=index(line,'DAY=');
when1=index(line,'WHEN=');
length sct2 when2  $2 day2 $3
        papernum start end $5;
if sct1>0 then do;
     sct2=substr(line,sct1+8,2);
     day2=substr(line,day1+4,3);
     when2=substr(line,when1+5,2);
end;
papernum=substr(line,1, 5);
start   =substr(line,13,5);
end     =substr(line,22,5);
run;
```

Since I wasn't sure where each field was on each line and I was doing this interactively, I actually wrote the code in sections. After I'd write a section, I used FSView to review the results and add additional code until I had everything identified. The above code is the final version.

The INFILE statement reads in each line as one very long line of 200 characters each. The MISSOVER option prevents the program from going to a new input line if it does not find values in the current line for all the INPUT statement variables. When an INPUT statement reaches the end of the current record, values that are expected but not found are set to missing. This will occur in my case when the only thing on a line is a carriage return.

The PAD option controls whether records read from an external file are padded with blanks to the length specified in the LRECL= option. Since not all the lines are $200 characters, my "line" variable would be blank for the shorter lines.

The INDEX function returns the starting place of the specified string. If the string isn't found, 0 is returned. This helps me to identify what kind of data is in each line.

The SUBSTR function extracts a portion of a variable. The parameters provide the name of the original variable, which column to start in and how many columns to extract. However, since the starting point of "SECTION=" can vary in each line, I've specified the column to start in as an expression rather than a fixed column.

Once I read in the data, I still had the problem than the section name and some other information were stored in a different record than each author's information. This DATA step resolves that problem.

```
data two;
 set one;
retain section day when;
if sct2 ne ' ' then do;
    section=sct2;
    day=day2;
    when=when2;
end;
if papernum='-----' then delete;
if flag=1 then delete;
run;
```

RETAIN will carry the values for section, day and when from record to record. The values are reset every time I encounter a record with a value for the sct2 field.

This step also deletes any records that are no longer needed for the author information.

MORE FIRST STEPS

Now to add to the confusion, I actually had 2 paper numbers. One was the original submission number and the other was assigned to accepted papers in ascending order based on their place in the program. The latter is the number the evaluators would use to look up an abstract or a paper in the proceedings. Of course, since this was assigned later, it was contained in a different file. I received this one as a file with commas as a delimiter.

```
data three;
 infile 'f:\sugi27_papernum.csv'
          dsd dlm=',';
length papernum $5. pprnum 8.
   section $2. author $15. title $125.;
input papernum $ pprnum section $
   author $ title $;
if pprnum=. then delete;
run;
```

The DLM option specifies the comma as the delimiter while the DSD option tells SAS to assume there are blanks between consecutive commas and to delete any quotes around text fields.

The default length for text fields from a delimited file is 8 so the length statement overrides the default. There were some "extra" records in the file which could be deleted since they didn't have a value for the PPRNUM variable.

```
proc sort data=three;
 by papernum;
run;

proc sort data=two;
 by papernum;
run;
```

The above steps simply sort my data. Although you could avoid this if you made some changes in subsequent code, this did let me use FSView to review the data in a meaningful order.

EVEN MORE FIRST STEPS

Well, just when I thought I had all the data, I discovered I was missing e-mail addresses. The person who sent this file included a header row.

```
data four;
 infile 'f:\primary_author_email.csv'
       dsd dlm=',';
 length num 8 primauth $30. email $40.;
 input num primauth email;
 if _n_=1 then delete;
run;
```

_N_ is an automatic variable which indicates the number of times SAS has looped through the DATA step. In this case, the first loop will be reading the header row so I can use _N_=1 to delete that record.

THE FINAL FIRST STEPS

Now I finally get to create the dataset with all my variables.

```
data all1;
merge two
 (Keep=papernum day start end when)
     three;
 by papernum;
run;

proc sql;
create table all2 as
 select a.*, b.*
 from all1 a full join four b
 on a.pprnum=b.num
```

```
 order by section, day, when, start;
quit;

data all error;
 set all2;
      if pprnum=. then output error;
 else if email=" " then output error;
 else output all;
run;
```

I needed to do this in 2 steps because one file contained the submission number, one contained the accepted paper number and the third contained both. Since the two with the submission number were already sorted, I just used a MERGE to combine them. I could have resorted the data but I opted to use PROC SQL to combine the new dataset with the third based on the accepted paper number. A FULL JOIN was used since I knew I did not have a 1-1 match in both datasets based on the record count and I wanted to be able to see the differences. I could have used PROC SQL for both steps but I couldn't have done it as one SQL step since the full join can only be done on two datasets at a time.

The KEEP= option on the dataset is used to bring in only the variables I need from that dataset. I could have eliminated them earlier by using either a KEEP= or DROP= option on the dataset when I created it. I could have used a DROP or KEEP statement within the data step itself. Remember the = sign is used with the option and the option is in parantheses.

The next DATA step simply separates my "good" data from my "bad" data. It isn't really necessary since I could have used a WHERE or IF later to separate the records. It was done merely as a matter of convenience to easily check my data.

WRITING THE REPORT

Finally, I get to the point where I can write out the report. The following is one method to do this. My experience has always been with writing out CSV files and then using Microsoft Excel to do the formatting. Using DATA _NULL_ lets me have total control over how fields are written out. Excel lets me format cell by cell. However, the downside of having all this control is that it can be time-consuming to write the code and do the formatting. A sample of the evaluation form is shown at the end of the paper.

```
proc format;
 value $section
  'AA'='ADVANCED TUTORIALS'
  'BT'='BEGINNING TUTORIALS'
  'AD'='APPLICATIONS DEVELOPMENT'
  'CC'='CODERS CORNER'
  'DM'='DATA MINING TECHNIQUES'
  'DP'='DATA PRESENTATION'
  'DW'='DATA WAREHOUSING AND ENTERPRISE
SOLUTIONS'
  'ET'='EMERGING TECHNOLOGIES'
  'HO'='HANDS-ON WORKSHOPS'
  'PO'='POSTERS'
  'PS'='PROFESSIONAL DEVELOPMENT AND
USER SUPPORT'
  'ST'='STATISTICS AND DATA ANALYSIS'
  'SY'='SYSTEMS ARCHITECTURE';
VALUE $sectchr
  'AA'='Ginger Carey'
  'BT'='Carnetta Francis-Minor'
  'AD'='Susan Kenny & Steve Wilson'
  'CC'='Rich Livornese & Duke Owen'
  'DM'='Olivia Parr'
  'DP'='Helen Carey & Lauren Haworth'
  'DW'='John Bentley & Andy Kuligowski'
  'ET'='Greg Barnes-Nelson & Joe
Kelley'
  'HO'='Debbie Buck & Warren Stinson'
  'PO'='Lori Griffin & Helen-Jean
Talbott'
  'PS'='Deb Cassidy'
  'ST'='Kim LeBouton'
  'SY'='Rob Fecht';
run;
```

The section above creates formats to be used to write out the longer section name or the section chair names based on the section abbreviation. A word of caution, however. I had hyphenated names but did not have any names like O'Leary. If I did, I would have to make adjustments in the formats and in the output file so Excel properly interpreted the single quote as part of the name.

```
data _null_;
 file 'F:\EVALS_FORMS.CSV';
 SET ALL;
 by section day when start;
 dlm=',';
 retain counter;

 length delvol $33.;
     if section='PO' then
delvol='Stage Presence';
else delvol=
'Delivery Volume & Stage Presence';
auth_num=trim(author) || "   (P" ||
put(pprnum,z3.) || "-27)";

IF _N_=1 THEN
 put
'c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,ROWTYPE
';
if first.when or counter>6 then do;
  counter=1;
```

```
if _n_ ne 1 then put "page_break" dlm
dlm dlm dlm dlm dlm dlm dlm dlm dlm
'0';
 put section $section. dlm DLM DLM dlm
dlm day " " when  DLM section $sectchr.
dlm dlm dlm "ID____" dlm '1' /
"RATING SCALE 0=VERY POOR   1=POOR
2=AVERAGE    3=GOOD   4=EXCELLENT"
 dlm dlm dlm dlm dlm dlm dlm dlm dlm
dlm '6'
;

put 'Title & Abstract,Presentation
Content,Presentation Organization,'
    delvol ',Presentation
Timing,Quality & Content of Visuals,'
'Response to Questions,Quality of Paper
in Proceedings,Content of Paper in
Proceedings,'
'Visual Effort,2';
end;
  put start "--" end dlm +(-1) '"'
auth_num +(-1) '"' DLM  dlm +(-1) '"'
title + (-1) '"'
dlm dlm dlm dlm dlm dlm dlm '3' /
   dlm dlm dlm dlm dlm dlm dlm dlm dlm
dlm  '4' /
   'COMMENTS' dlm dlm dlm dlm dlm dlm
dlm dlm dlm dlm '5';
counter+1;
run;
```

The BY statement creates system variables which will identify the first record and last record within each change of the variables in the BY statement. You will have an automatic variable in the form FIRST.variable_name and LAST.variable_name for each variable in your BY statement. These variables will help me to control when certain things are written.

I created a variable named DLM to hold the delimiter because it was easier for me to type dlm that it was to type ‘,’. There’s just something about all those quotes and commas together which I find difficult to read after I’ve been staring at a computer screen all day.

The RETAIN statement creates a variable to hold the number of records I’ve processed. RETAIN will carry the value to the next record for me. There is only room for 6 presentations on each page so I’ll need to repeat the header when I move to another page.

The Posters section needs to have a different description so I created a variable to hold the two versions. I could have done the same work by using IF statements or a format later in the code but this was the easiest for me.

The author and paper number need to be together in the same cell in Excel. I also need to format the paper number in the form Pxxx-27 where xxx is the assigned paper number. The double bars are the symbol for concatenate. The hard part for some of you will be to figure out where those bars are on your keyboard since it can vary by keyboard, operating system and any interface you may be using! TRIM will drop any trailing blanks from the author’s name so the paper number follows immediately after it. You may not think you had blanks but they were added to make each name be the specified length of the name variable. However, I do need one blank so I put it back it before the parenthesis for the paper number. The Z3. format will put leading zeroes in the paper number.

For convenience once I get to Excel, I create a header row as my very first line. _N_=1 was used to identify my first record in the same manner as earlier in the code. The ROWTYPE field is created to help in formatting within Excel. Each type of row will have its own number.

I wanted to start a new page any time there was a change in the “when” variable which identified morning/afternoon sessions. This is where automatic variables created with the BY statement are used. I also needed to start a new page if there were more than 6 speakers within that session. Any time I start I new page, I need to reset the counter variable. One thing that confuses people is whether to reset a counter variable to 0 or to 1. The answer: It depends. It depends on where you reset the counter and where you increment it. In this case, I’m resetting it after SAS has read in the observation and I increment it at the end of the processing for a current observation. If you need to increment the counter when SAS reads the observation, you’ll probably want to reset the value to 0.

All the code between the DO and END creates the header portion of my finished report. There are consecutive delimiters being written because I need blank columns in Excel to be able to format properly. Several of my fields will wrap across multiple cells in Excel. The text “page_break” was inserted as an easy way to set page breaks after the file was in Excel. There isn’t a page break on the first record because I would end up with an extra page.

The PUT statement which starts with “put start” may look rather confusing. Since Excel needs to have double quotes to ensure that single quotes in a name are processed correctly, I have to actually write out each set of double quotes as separate fields. To get SAS to do

this, the double quotes need to be in single quotes. Clear, right? OK, I'll admit I did not get this part of the code right on the first try. I even needed to open the file in Notepad to verify what was written out from SAS. That's how I was able to spot quoting issues that Excel did not process correctly.

The other thing that Excel did not like was a blank after the delimiter. Since I was using the character field DLM, SAS added a blank when it was written out. To eliminated this blank, I just needed to back up a character. This is a simple task. You can use +(x) to move forward x number of columns. Specifying x as a negative number is how you move backwards.

CREATING THE EVALUATION LETTER

```
data ppr_info;
set (This code reads in SAS datasets
from each section - datasets have 1
record per paper.);
 run;

data eval_info;
set (Read in SAS datasets from each
section - datasets have 1 record per
judge per paper. The judge field was
only needed for data entry so it is
dropped here.);
  avg_eval_score=round
      (mean(of _numeric_),.1);
  if avg_eval_score=. then delete;
  comments=translate(comments,"'",'"');
  output;

 run;
```

Each section had two datasets - one with the details from each judge and one with information about the paper. In the detail dataset, the only numeric fields were the evaluators scores so I could use _NUMERIC_ to avoid having to list out each field. Since the number of judges varied for each section/paper, records were deleted if there were no scores. Some of the comments had double quotes within the text. This was going to cause a problem in Excel so the TRANSLATE function was used to change all double quotes to single quotes.

```
proc means data=eval_info noprint
  mean nway ;
  class papernum section group;
  var _numeric_;
  output out=ppr_scores
  (drop=_type_ _freq_) mean=;
run;
proc means data=ppr_scores noprint
  mean n nway ;
  class section group;
  var _numeric_;
  output out=section_scores
  (drop=_type_ _freq_)
  mean=sec1-sec11
  n(avg_eval_score)=count;
run;
proc means data=ppr_scores
 (where=(section ne 'PO')) noprint
  mean nway ;
  class group;
  var _numeric_;
  output out=conf_scores
  (drop=_type_ _freq_)
  mean=conf1-conf11;
run;
```

The MEANS procedure is used to summarize the results. The NWAY option specifies to output only the observations with the highest _TYPE_ value rather than every possible subgroup of the CLASS variables. The automatic variables _TYPE_ and _FREQ_ are dropped from the output dataset because they aren't needed for further processing when NWAY is used.

The N statistic is requested on the second MEANS procedure because groups within a section that have only 1 or 2 papers will not be shown a group average. The third MEANS exclude the Poster section from the overall conference score average. The group variable identifies whether the paper was invited, contributed or presented by SAS.

```
PROC SQL;
CREATE TABLE FINAL AS
Select PPR_SCORES.*,
SECTION_SCORES.*,
CONF_SCORES.*,
PPR_INFO.Primauth,
PPR_INFO.Email,
PPR_INFO.Title,
eval_info.comments
from WORK.PPR_SCORES left join
WORK.SECTION_SCORES on
ppr_scores.section=
section_scores.section and
ppr_scores.group=section_scores.group
left join conf_scores on
ppr_scores.group=conf_scores.group
left join ppr_info on
ppr_scores.papernum=ppr_info.papernum
left join eval_info on
ppr_scores.papernum=eval_info.papernum
order by papernum;
QUIT;
```

This SQL statement combines the overall conference scores with the section scores with the average for each

paper with other information for each paper such as the title. The result is one record per paper per judge. However, the file will be written out with one record per paper.

```
PROC FORMAT;
(The formats for section name and
section chairs are also required here.)
value $group
   'Con'='Contributed'
   'Inv'='Invited'
   'SAS'='SAS';
run;

data _null_;
 set final;
 by papernum;
 file "F:\LETTERS.CSV" lrecl=5000;
 dlm=',';
 group_long=trim(put(group,$group.));
 first_name=scan(primauth,1);
 if _n_=1 then put
'Papernum,Section,Group,Primary
Author,First_Name,Email,Title,
Title_Abstract,Present_Content,
Present_Organization,Stage_Presence,
Timing,Visuals,Questions,
Quality_Proceedings,
Content_Proceedings,Effort,
Avg_eval_score,
Sec_Title_Abstract,Sec_Present_Content,
Sec_Present_Organization,
Sec_Stage_Presence,Sec_Timing,
Sec_Visuals,Sec_Questions,
Sec_Quality_Proceedings,
Sec_Content_Proceedings,Sec_Effort,
Sec_Avg_eval_score,Group_Count,
Conf_Title_Abstract,
Conf_Present_Content,
Conf_Present_Organization,
Conf_Stage_Presence,Conf_Timing,
Conf_Visuals,Conf_Questions,
Conf_Quality_Proceedings,
Conf_Content_Proceedings,Conf_Effort,
Conf_Avg_eval_score,
Comments1,Comments2,Comments3,
Comments4,Comments5';

if first.papernum then do;
put  papernum +(-1) ','
section $section. ','  group_long ','
primauth ',' first_name ','
email +(-1) ','
'"' title '"' dlm
TITLE_ABSTRACT       3.1 dlm
PRESENT_CONTENT      3.1 dlm
PRESENT_ORGANIZATION 3.1 dlm
STAGE_PRESENCE       3.1 dlm
TIMING               3.1 dlm
VISUALS              3.1 dlm
QUESTIONS            3.1 dlm
QUALITY_PROCEEDINGS  3.1 dlm
CONTENT_PROCEEDINGS  3.1 dlm
EFFORT               3.1 dlm
avg_eval_score       3.1 dlm @;

if count<=2 then put
'N/A,N/A,N/A,N/A,N/A,N/A,N/A,N/A,N/A,
N/A,N/A,'
count dlm @;
ELSE PUT
sec1   3.1  dlm
sec2   3.1  dlm
sec3   3.1  dlm
sec4   3.1  dlm
sec5   3.1  dlm
sec6   3.1  dlm
sec7   3.1  dlm
sec8   3.1  dlm
sec9   3.1  dlm
sec10  3.1  dlm
sec11  3.1  dlm
COUNT  dlm @;
   if section='PO' then put
'N/A,N/A,N/A,N/A,N/A,N/A,N/A,N/A,N/A,
N/A,N/A'  @;
else put
conf1   3.1  dlm
conf2   3.1  dlm
conf3   3.1  dlm
conf4   3.1  dlm
conf5   3.1  dlm
conf6   3.1  dlm
conf7   3.1  dlm
conf8   3.1  dlm
conf9   3.1  dlm
conf10  3.1  dlm
conf11  3.1  @;
end;  ** End first.papernum;

if comments ne ' ' then  put ',"'
comments +(-1) '"' @;
if last.papernum then put;
run;
```

The LRECL option is used in this example to ensure the longest records are not cut off since the default record length under Windows is 256. The group name needed to be spelled out for the Excel file so the PUT function is used with the format. TRIM is used to eliminate any trailing blanks. The greeting in the letter just needs the first name of the author so SCAN is used to extract that from the full name. The disadvantage of this method is that anyone with a 2-part first name such as Billie Jo will be identified incorrectly. If you were in a situation where this would be unacceptable, you would be wiser to have a first name variable in your dataset.

_N_ is used to write out a header row that will be used as part of the mail merge. You'll notice that there are 5 comments fields since there could be comments from up to 5 judges.

Since the input file has multiple records per paper yet the output file needs to have a single record per paper, first.papernum is used to write just one record per paper. First, several fields to identify the paper are written as well as the average scores for the paper. The trailing @ is used to hold onto the current line to write out more fields.

There were two special cases which required reporting results as "N/A" rather than the actual scores. Section with less than three papers for a group did not get a section score. Posters did not get a conference score because they were a different type of presentation. In both cases, IF/THEN/ELSE is used to select the appropriate information to be written. The trailing @ is used again to continue holding the line for more fields.

The comments field is the only field that must be used from every record so the first.papernum section is ended. If there is a comment, you must first write a comma to separate the comment from the previous field. You also need to write the double quotes which Excel requires. SAS writes a blank after the end of a character field so +(-1) is used to back up a space as shown earlier in the code. Another double quote is used to end the comment. Once again, a trailing @ is used to continue holding the line for any more comments. When the last record is reached for a paper number, a PUT with nothing following it is used to finally release the line which has been held throughout all the records for that paper.

ALTERNATIVE METHODS

I won't discuss the alternative methods in the paper but they will be discussed in the presentation. This is for two reasons. One is because I don't have enough space left in the paper. The second is because some of the alternatives are experimental in V8.2 and I currently have V8.0 installed!

The basic idea is to use ODS and existing procedures. The following code will create a formatted spreadsheet using the default formatting for the frequency procedure using ODS.

```
ODS HTML
   BODY='F:\TEST.XLS';
proc freq;
 tables section;
run;
ODS HTML CLOSE;
```

Now the challenge is to find the best procedure to get close to my desired output. Perhaps PROC REPORT will work. The formatting might be accomplished by creating a custom styles and templates in ODS. One of the experimental features in V8.2 is writing out to RTF files which might be a better choice than Excel.

To get more information about my experiences with the alternative methods, contact me after the SESUG conference.

SUMMARY

The good news is that you can create any report you need using SAS. The bad news is that it may take a lot of coding. The other bad news is that you might have to use something else to do some formatting. However, V8 has increased the options available to get what you need - and hopefully without too much coding.

CONTACT INFORMATION

Deb Cassidy
debchome@mail.com

SAMPLE RAW DATA

SUGI 27 Data 4

---------- Section=AD DAY=Monday SESSION=' ' WHEN=AM ROOM=Southern Hemisphere IV ----------

PAPERNUM	TIME	END	COUNT	LNAME1	TITLE
I3003	8:00	8:50	21	Carpenter	Library and File Management: Building a
I3015	9:00	9:50	22	Ray	Large-scale System Development in Base S
W2009	10:00	10:20	23	Smith	Programming Tricks for Reducing Storage
W2286	10:30	10:50	24	Muller	Optimizing SAS® Version 8 in
W2079	11:00	11:20	25	Lee	Accessing Microsoft Excel and Microsoft
W2098	11:30	11:50	26	Eberhardt	Rev Up Your Spreadsheets with Some V8 Po
0	11:50	11:50	.	Session Ends	

---------- Section=AD DAY=Monday SESSION=' ' WHEN=PM ROOM=Southern Hemisphere IV ----------

SAMPLE EVALUATION FORM

APPLICATIONS DEVELOPMENT **Mon AM** **Susan Kenny & Steve Wilson** **ID____**

RATING SCALE 0=VERY POOR 1=POOR 2=AVERAGE 3=GOOD 4=EXCELLENT

Title & Abstract	Presentation Content	Presentation Organization	Delivery Volume & Stage Presence	Presentation Timing	Quality & Content of Visuals	Response to Questions	Quality of Paper in Proceedings	Content of Paper in Proceedings	Visual Effort
8:00 --8:50	**Carpenter (P021-27)**		**Library and File Management: Building a Dynamic Application**						

COMMENTS

9:00 --9:50	**Ray (P022-27)**		**Large-scale System Development in Base SAS®**						

COMMENTS

10:00 --10:20	**Lafler**		**SAS Performance Tuning for Programmers**						

COMMENTS

HEADER AND SAMPLE RECORDS FROM THE CSV FILE FOR THE EVALUATION FORM

```
c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,ROWTYPE
APPLICATIONS DEVELOPMENT                , , , , , Mon  AM , Susan Kenny & Steve Wilson       , , , ID____,
1
RATING SCALE 0=VERY POOR   1=POOR    2=AVERAGE    3=GOOD   4=EXCELLENT, , , , , , , , , , , 6
Title & Abstract,Presentation Content,Presentation Organization,Delivery Volume & Stage Presence
,Presentation Timing,Quality & Content of Visuals,Response to Questions,Quality of Paper in
Proceedings,Content of Paper in Proceedings,Visual Effort,2
8:00 --8:50 ,"Carpenter   (P021-27)", ,"Library and File Management: Building a Dynamic Application", , , ,
, , , 3
, , , , , , , , , , 4
COMMENTS, , , , , , , , , , 5
9:00 --9:50 ,"Ray   (P022-27)", ,"Large-scale System Development in Base SAS®", , , , , , , 3
, , , , , , , , , , 4
COMMENTS, , , , , , , , , , 5
10:00 --10:20 ,"Smith   (P023-27)", ,"Programming Tricks for Reducing Storage and Work Space", , , , , , , 3
, , , , , , , , , , 4
COMMENTS, , , , , , , , , , 5
```

HEADER AND SAMPLE RECORD FROM THE CSV FILE FOR THE EVALUATION LETTER

Papernum,Section,Group,Primary Author,First_Name,Email,Title,Title_Abstract,Present_Content,Present_Organization,Stage_Presence,Timing,Visuals,Questions,Quality_Proceedings,Content_Proceedings,Effort,Avg_eval_score,Sec_Title_Abstract,Sec_Present_Content,Sec_Present_Organization,Sec_Stage_Presence,Sec_Timing,Sec_Visuals,Sec_Questions,Sec_Quality_Proceedings,Sec_Content_Proceedings,Sec_Effort,Sec_Avg_eval_score,Group_Count,Conf_Title_Abstract,Conf_Present_Content,Conf_Present_Organization,Conf_Stage_Presence,Conf_Timing,Conf_Visuals,Conf_Questions,Conf_Quality_Proceedings,Conf_Content_Proceedings,Conf_Effort,Conf_Avg_eval_score,Comments1,Comments2,Comments3,Comments4,Comments5

P999-27,APPLICATIONS DEVELOPMENT,Invited ,Charlie Brown ,Charlie ,charliebrown@snoopy.com,"How Do You Provide Examples Without Giving Away Confidential Information? ", 4.0, 4.0, 3.5, 3.0, 4.0, 4.0, 3.0, 4.0, 4.0, 4.0, 3.7, 3.5, 3.3, 3.4, 3.3, 3.7, 3.5, 3.1, 3.5, 3.2, 3.8, 3.4, 5 , 3.5, 3.3, 3.4, 3.3, 3.7, 3.5, 3.1, 3.5, 3.2, 3.8, 3.4,"Nice introduction - you told audience what you were covering & you did it. Well organized. A little hard to hear you in the back of room at times. Nice humor in the slides and presentation. Please repeat questions before answering to ensure audience heard the question. Wonderful knowledge of material and well rehearsed before your presentation."

SAMPLE LETTER FROM WORD - MAIL MERGE COMBINES LETTER WITH EXCEL DATA
Dear «First_Name»,

This section is a message that is appropriate for all speakers. The customized part is the title and section name in the following sentence. Thank you for your presentation in entitled " «Title»" given in the «Section» Section at SUGI 27.

As promised here are the scores. The descriptions for each of the categories are also included.

Thank you
Ms. Conference Chair

	Your Average Score	Section Average	Conference Average
Title & Abstract	«Title_Abstract»	«Sec_Title_Abstract»	«Conf_Title_Abstract»
Presentation Content	«Present_Content»	«Sec_Present_Content»	«Conf_Present_Content»
Presentation Organization	«Present_Organization»	«Sec_Present_Organization»	«Conf_Present_Organization»
Delivery Volume & Stage Presence	«Stage_Presence»	«Sec_Stage_Presence»	«Conf_Stage_Presence»
Presentation Timing	«Timing»	«Sec_Timing»	«Conf_Timing»
Quality & Content of Visuals	«Visuals»	«Sec_Visuals»	«Conf_Visuals»
Response to Questions	«Questions»	«Sec_Questions»	«Conf_Questions»
Quality of Paper in Proceedings	«Quality_Proceedings»	«Sec_Quality_Proceedings»	«Conf_Quality_Proceedings»
Content of Paper in Proceedings	«Content_Proceedings»	«Sec_Content_Proceedings»	«Conf_Content_Proceedings»
Visual Effort	«Effort»	«Sec_Effort»	«Sec_Effort»
Overall Average	«Avg_eval_score»	«Sec_Avg_eval_score»	«Conf_Avg_eval_score»

COMMENTS FROM EVALUATORS

«Comments1»

«Comments2»

«Comments3»

«Comments4»

«Comments5»