Enhancing Code with the SQL Procedure
Jeff Goodrich, STATPROBE, Inc., Lexington, KY
Jon Fowler, STATPROBE, Inc., Lexington, KY

ABSTRACT
This paper will illustrate ways to enhance your code with SQL. We will compare the SQL Procedure versus the base SAS statements necessary to produce summary information. A step by step approach will be used to compare code from both techniques, which produce equivalent results.

INTRODUCTION
To enhance code usually refers to accomplishing more (or equivalent results) with less (program code) or improving the time required to reach the desired result. When producing summary information, the SQL procedure can be used to perform several tasks at once. Whereas with base SAS code, several steps, such as Data steps, Proc freqs, and Proc sorts are required to produce the same result. Our goal, in the following example, is to illustrate a way the SQL procedure can be used to reduce the amount of code necessary to produce an Adverse Event summary table.

Example - Producing an Adverse Event Summary Table
We want to produce an AE summary table that will look like the following table below.

<table>
<thead>
<tr>
<th>BODY SYSTEM</th>
<th>Treatment Group</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Patients with an AE</td>
<td>222</td>
<td>221</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>BODY AS A WHOLE</td>
<td>187</td>
<td>186</td>
<td>147</td>
<td></td>
</tr>
<tr>
<td>Abdomen Enlarged</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Abdominal Pain</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CARDIOVASCULAR SYSTEM</td>
<td>34</td>
<td>35</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>Arrhythmia</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Hemorrhage</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DIGESTIVE SYSTEM</td>
<td>185</td>
<td>177</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>Abnormal Stools</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Anorexia</td>
<td>7</td>
<td>22</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Approach A - Using Base SAS Statements
Step1 - We have two data sets (see below), AE which contains the adverse event information and Randgrp which contains the treatment each patient received. We want to merge the two data sets by randomization number in order to establish the treatment group of the patients reporting adverse events.

```
Proc sort data = study.ae (keep = randno bterm pterm onsetdt)
  out = ae;
  by randno bterm pterm onsetdt;
  where pterm ne " ";
Run;
```
Proc sort data = study.randgrp (keep = randno rand)
on = randgrp;
    by randno;
Run;

Data aerand;
merge randgrp (in = a) ae (in = b);
    by randno;
if a and b;
Run;

Partial listing of AERAND data set

<table>
<thead>
<tr>
<th>Randomization Number</th>
<th>Preferred Term</th>
<th>Body System</th>
<th>Numeric AE Start Date</th>
<th>Treatment Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Abdominal Pain</td>
<td>Body as a Whole</td>
<td>13400</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>Headache</td>
<td>Body as a Whole</td>
<td>13418</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>Constipation</td>
<td>Digestive</td>
<td>13400</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>Dyspepsia</td>
<td>Digestive</td>
<td>13417</td>
<td>2</td>
</tr>
</tbody>
</table>

Step2 - We want to count each patient who had an adverse event. We use the ‘first.’ to ensure that we count each patient with an adverse event only once. The variable parm is created to be our output variable for column 1 of the summary table.

Data all_ae;
    set aerand;
    by randno bterm pterm onsetdt;
if first.randno;
Run;

Proc freq data = all_ae noprint;
    tables rand / out = all_ae (drop = percent);
Run;

Data all_ae;
    set all_ae;
    length parm $60;
    parm = "All Patients with an AE";
Run;

All_ae data set

<table>
<thead>
<tr>
<th>Treatment Group</th>
<th>Frequency Count</th>
<th>Parm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>222</td>
<td>All Patients with an AE</td>
</tr>
<tr>
<td>2</td>
<td>221</td>
<td>All Patients with an AE</td>
</tr>
<tr>
<td>3</td>
<td>180</td>
<td>All Patients with an AE</td>
</tr>
</tbody>
</table>

Step3 - We want to count the number of body systems associated with an adverse event per patient.

Data all_bterm;
    set aerand;
    by randno bterm pterm onsetdt;
if first.bterm;
Run;

Proc freq data = all_bterm noprint;
    tables rand * bterm / out = all_bterm (drop = percent);
Run;

Data all_bterm;
    set all_bterm;
    length parm $60;
    parm = " " || trim(bterm);
Run;
### Partial listing of All_bterm data set

<table>
<thead>
<tr>
<th>Treatment Group</th>
<th>Body System Code</th>
<th>Frequency Count</th>
<th>Parm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Body as a Whole</td>
<td>187</td>
<td>Body as a Whole</td>
</tr>
<tr>
<td>1</td>
<td>Cardiovascular</td>
<td>34</td>
<td>Cardiovascular</td>
</tr>
<tr>
<td>2</td>
<td>Body as a Whole</td>
<td>186</td>
<td>Body as a Whole</td>
</tr>
<tr>
<td>2</td>
<td>Cardiovascular</td>
<td>35</td>
<td>Cardiovascular</td>
</tr>
<tr>
<td>3</td>
<td>Body as a Whole</td>
<td>147</td>
<td>Body as a Whole</td>
</tr>
<tr>
<td>3</td>
<td>Cardiovascular</td>
<td>29</td>
<td>Cardiovascular</td>
</tr>
</tbody>
</table>

Step4 - We want to count the number of preferred terms associated with an adverse event per patient.

```plaintext
Data all_pterm;
  set aerand;
  by randno bterm pterm onsetdt;
  if first.pterm;
Run;

Proc freq data = all_pterm noprint;
  tables rand * bterm * pterm / out = all_pterm (drop = percent);
Run;
```

### Partial listing of All_pterm data set

<table>
<thead>
<tr>
<th>Treatment Group</th>
<th>Body System Code</th>
<th>Preferred Term</th>
<th>Frequency Count</th>
<th>Parm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Body as a Whole</td>
<td>Abdomen Enlarged</td>
<td>1</td>
<td>Abdomen Enlarged</td>
</tr>
<tr>
<td>1</td>
<td>Body as a Whole</td>
<td>Abdominal Pain</td>
<td>38</td>
<td>Abdominal Pain</td>
</tr>
<tr>
<td>2</td>
<td>Body as a Whole</td>
<td>Abdominal Pain</td>
<td>41</td>
<td>Abdominal Pain</td>
</tr>
<tr>
<td>2</td>
<td>Body as a Whole</td>
<td>Accidental Injury</td>
<td>10</td>
<td>Accidental Injury</td>
</tr>
<tr>
<td>3</td>
<td>Body as a Whole</td>
<td>Abdomen Enlarged</td>
<td>1</td>
<td>Abdomen Enlarged</td>
</tr>
<tr>
<td>3</td>
<td>Body as a Whole</td>
<td>Abdominal Pain</td>
<td>25</td>
<td>Abdominal Pain</td>
</tr>
</tbody>
</table>

Step5 - Group all counts together by setting each data set to create Sum_AE. Then, sort by body system, preferred term, and parm to prepare for transpose.

```plaintext
Data sum_ae;
  set all_ae
  all_bterm
  all_pterm;
Run;

Proc sort data = sum_ae;
  by bterm pterm parm;
Run;
```

### Partial listing of Sum_ae data set

<table>
<thead>
<tr>
<th>Treatment Group</th>
<th>Body System Code</th>
<th>Preferred Term</th>
<th>Frequency Count</th>
<th>Parm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>222</td>
<td>All Patients with an AE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>221</td>
<td>All Patients with an AE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>180</td>
<td>All Patients with an AE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Body as a Whole</td>
<td>Abdominal Pain</td>
<td>187</td>
<td>Body as a Whole</td>
</tr>
<tr>
<td>2</td>
<td>Body as a Whole</td>
<td>Abdominal Pain</td>
<td>186</td>
<td>Body as a Whole</td>
</tr>
<tr>
<td>3</td>
<td>Body as a Whole</td>
<td>Abdominal Pain</td>
<td>147</td>
<td>Body as a Whole</td>
</tr>
<tr>
<td>1</td>
<td>Body as a Whole</td>
<td>Abdomen Enlarged</td>
<td>1</td>
<td>Abdomen Enlarged</td>
</tr>
</tbody>
</table>
Step 6 - Finally, transpose the data so that a column is created for each treatment group.

```sas
Proc transpose data = sum_ae
    out = final (keep = parm _1 _2 _3);
    var count;
    by bterm pterm parm;
    id rand;
Run;
```

Partial listing of Final data set

<table>
<thead>
<tr>
<th>Parm</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Patients with an AE</td>
<td>222</td>
<td>221</td>
<td>180</td>
</tr>
<tr>
<td>Body as a Whole</td>
<td>187</td>
<td>186</td>
<td>147</td>
</tr>
<tr>
<td>Abdominal Enlarged</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Abdominal Pain</td>
<td>38</td>
<td>41</td>
<td>25</td>
</tr>
</tbody>
</table>

**Approach B - Using the SQL Procedure**

Step 1 - Create data set AERAND as in previous step 1. Notice the Adverse Event start date does not have to be included since the ‘first.’ technique will not be used.

```sql
Proc sql;
create table aerand as
    select  a.randno, a.bterm, a.pterm, b.rand
    from study.ae (where = (pterm ne " ")) as a
    inner join
    study.randgrp as b
    on (a.randno = b.randno);
Quit;
```

Partial listing of AERAND data set

<table>
<thead>
<tr>
<th>Randomization Number</th>
<th>Preferred Term</th>
<th>Body System Code</th>
<th>Treatment Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Abdominal Pain</td>
<td>Body as a Whole</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>Headache</td>
<td>Body as a Whole</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>Constipation</td>
<td>Digestive</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>Dyspepsia</td>
<td>Digestive</td>
<td>2</td>
</tr>
</tbody>
</table>

Step 2 - As before we want to count each patient who had an adverse event. This time however, to ensure we count each patient with an adverse event only once, we select the distinct randomization number grouped by type of treatment to count. And we again create parm as our output variable for column 1 of the summary.

```sql
Proc sql;
create table sum_ae as
    select rand, count (distinct randno) as count, "All Patients with an AE" as parm
    from aerand
    group by rand
    outer union corresponding
```

Step 3 - We use the same technique to count the number of body systems associated with an adverse event per patient. The ‘group by’ will change to treatment group and body system.

```sql
select rand, bterm, count (distinct randno) as count, " " || trim(bterm) as parm
    from aerand
    group by rand, bterm
    outer union corresponding
```

Step 4 - Finally, with our last count, we group by treatment group, body system, and preferred term.

```sql
select rand, bterm, pterm, count (distinct randno) as count, " " || trim(pterm) as parm
    from aerand
    group by rand, bterm, pterm
    order by bterm, pterm, parm;
Quit;
```
Step 5 - To set all the counts together, we use the set operator outer union with the keyword corresponding. To prepare for the transpose, we order the table sum_ae by body system, preferred term, and our output parameter parm.

```
Proc sql;
create table sum_ae as
    select rand, count (distinct randno) as count, "All Patients with an AE" as parm
    from aerand
    group by rand
    outer union corresponding

    select rand, bterm, count (distinct randno) as count, "   " || trim(bterm) as parm
    from aerand
    group by rand, bterm
    outer union corresponding

    select rand, bterm, pterm, count (distinct randno) as count, "        " || trim(pterm) as parm
    from aerand
    group by rand, bterm, pterm
    order by bterm, pterm, parm;
Quit;
```

Partial listing of Sum_ae data set

<table>
<thead>
<tr>
<th>Treatment Group</th>
<th>Body System Code</th>
<th>Preferred Term</th>
<th>Frequency Count</th>
<th>Parm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>222</td>
<td>All Patients with an AE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>221</td>
<td>All Patients with an AE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>180</td>
<td>All Patients with an AE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Body as a Whole</td>
<td>187</td>
<td>Body as a Whole</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Body as a Whole</td>
<td>186</td>
<td>Body as a Whole</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Body as a Whole</td>
<td>147</td>
<td>Body as a Whole</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Body as a Whole</td>
<td>Abdomen Enlarged</td>
<td>1</td>
<td>Abdomen Enlarged</td>
</tr>
<tr>
<td>3</td>
<td>Body as a Whole</td>
<td>Abdomen Enlarged</td>
<td>1</td>
<td>Abdomen Enlarged</td>
</tr>
</tbody>
</table>

Step 6 - Finish the process by transposing the data as before.

**CONCLUSION**

From our example above, we found that in both approaches the end result, our final data set, was the same. However, with the SQL procedure we were able to perform several tasks at once. First, we were able to create a data set, AERAND, in one step which, when using base SAS statements required two sorting procedures and a data step to merge the two data sets together. And finally, the second SQL we employed generated three different counts, set the results together, and sorted the final data set. In all, producing the equivalent result of 7 data steps, 3 frequency procedures, and one sorting procedure. As we mentioned before, one way of enhancing your code is doing more with less. The SQL procedure can be an effective tool to use in order to achieve that goal.

**TRADEMARKS**

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

**CONTACT INFORMATION**

Contact the authors at:

Jeff Goodrich  
STATPROBE, Inc.  
220 Lexington Green Circle, Suite 200  
Lexington, KY 40503  
(859) 971-8899 ext. 570  
Fax: (859) 273-7929  
Email: jeff.goodrich@lx.statprobe.com

Jon Fowler  
STATPROBE, Inc.  
220 Lexington Green Circle, Suite 200  
Lexington, KY 40503  
(859) 971-8899 ext. 554  
Fax: (859) 273-7929  
Email: jon.fowler@lx.statprobe.com