Conquering the Fear of Shift Tables
Monal Kohli, Smith Hanley Consulting

ABSTRACT
Often people are intimidated by the word “Shift Tables” in any clinical trial Programming environment. Shift tables are actually pretty simple, they are just cross frequencies of one variable to another but presented in a different way. First it involves a thorough understanding of what is it that we will accomplish after having a shift table, why can’t a simple table do what a shift table does? Shift tables provide a better vision to data and encapsulates the changes in data from one point to another. Shift tables can get really interesting and challenging at the same time. This paper is an attempt to fight this fear which will help all of us get a better understanding of the anatomy of shift tables. It will help us understand the basics of why shift tables are created and a stepwise explanation of how to create them in a wink. Let’s start shifting!!

INTRODUCTION
The whole purpose of any shift table in any clinical study is to deduce how the results are varying from the baseline to various other visits in the study. Before even starting to code a shift table it is always helpful to analyze the numbers and come up with a template which surely will make life easier. Another common mistake which we all commit is not taking proper consideration of missing values. I will take an example and explain the anatomy, the basic code and at the end we will see a compact macro which will finally create a shift table at the end.

DATA USED FOR ANALYSIS
Below is the data that we will be using to understand the entire process. Here we are using two very simple datasets which contain data for two visits “Visit 1” and “Visit 2”. Visit 1 is the baseline. Visit 2 is post baseline.

```
Data visit1;
  infile cards missover;
  input @1 usubjid 5. @6 cholesterol $1. @8 trtgrp $1. @10 lab 3.;
  visit='Visit 1 /Baseline';
  cards;
  1001 A 1 1
  1002 N 1 1
  1003 A 2 1
  1004 N 1 1
  1005 N 1 1
;

Data visit2;
  infile cards missover;
  input @1 usubjid 5. @6 cholesterol $1. @8 trtgrp $1. @10 lab 3.;
  visit='Visit 2';
  cards;
  1001 N 1 1
  1002 N 1 1
  1003 N 2 1
  1004 A 2 1
  1005 N 1 1
;
RUN;
```
WHY DO WE NEED SHIFT TABLES?
In any clinical study it is crucial to look at the effect of the drug by comparing values at baseline and post –baseline . Shift tables can be created for any data but it is very common to compare results from labs, vitals and ECG . It can be extremely useful in understanding the changes from one time point to another . They make it easier to figure out any abnormality in data because the layout helps to get the count of subjects at two time points together . They can be very helpful in validating the summary tables and vice-versa.

UNDERSTANDING THE PROCESS
Before coding the shift tables it is very important to understand the data and the variable for which the shift table is created. We look at a particular variable values for any two time point using a shift table . To get a better understanding let’s look at the two datasets above. We have a variable Cholesterol whose values we want to compare at baseline and Visit 2 . The values that this variable can have are “Abnormal”, “Normal”, “Not Available” . It can be extremely helpful to see the subjects who had a “Abnormal” value at baseline and “Normal” value at Visit 2 . If this subject is not in placebo group, it can be a very good indicator about the efficacy of the drug. In a nutshell, this is really why we need a shift table for lot of different variables in a clinical study. It can add a lot of dimension to the data in one shot.

DIFFERENT LAYOUTS OF A SHIFT TABLE
Once we are acquainted with the different values a variable can have, in this case “Cholesterol” can have “Abnormal”, “Normal” and “Not Available” we can design the layout for the table. There can be basically layout. First layouts in this scenario.

<table>
<thead>
<tr>
<th>Layout I:</th>
<th>Final Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Treatment Group</td>
</tr>
<tr>
<td>1</td>
<td>Normal</td>
</tr>
<tr>
<td></td>
<td>Abnormal</td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
</tr>
<tr>
<td></td>
<td>Total</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Layout II:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Baseline</td>
</tr>
<tr>
<td>Abnormal</td>
<td>Normal</td>
</tr>
<tr>
<td></td>
<td>Abnormal</td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
</tr>
</tbody>
</table>

STEPS TO CREATE A SHIFT TABLE
Creation of a shift table involves realigning data, performing means to get the totals and output the results using “PROC REPORT”. Since the variable name is same in both the datasets (Baseline, Visit 2), first step is aligning the data in such a way that each subject has base value and post base value. Once we have that information we need to create columns in the dataset for each value in the postbase_val. Basically if post base values are “Missing”, “Abnormal” and “Normal” then we will have three columns “Missing”, “Abnormal” and “Normal”. Now we have data aligned in such a way that we can get a column wise sum for each treatment group, baseline value and the different post baseline values.

We are basically looking for a method to facilitate the counting of subjects in the following pattern:

<table>
<thead>
<tr>
<th>Base</th>
<th>Post Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Abnormal</td>
</tr>
<tr>
<td>Normal</td>
<td>Normal</td>
</tr>
<tr>
<td>Normal</td>
<td>Not Available</td>
</tr>
<tr>
<td>Abnormal</td>
<td>Normal</td>
</tr>
<tr>
<td>Abnormal</td>
<td>Abnormal</td>
</tr>
<tr>
<td>Abnormal</td>
<td>Not Available</td>
</tr>
</tbody>
</table>
And so on...
So basically we are creating a frequency of baseline value to different post baseline values. Now that we have looked at the process, let’s try to analyze how to create a shift table in SAS. This is just one way of creating the shift table but there can be numerous ways to program.

Let’s look at the SAS Steps now. For simplicity we will discuss the steps for only one treatment group.

**STEP I: REARRANGING DATA**

```sas
data base (keep=usubjid base_flg trtgrp lab)
  postbase (keep=usubjid lab postbase_flg trtgrp);
set visit2 (in=a) case (visit1);
  if trtgrp='1';
    if b then do;
      base_flg=cholesterol;
      output base;
    end;
  if a then do;
      postbase_flg=cholesterol;
      output postbase;
    end;
run;
```

In the above code snippet we are basically reading visit 1 and visit2 data and renaming the cholesterol variable so that we can identify the baseline value and post baseline value distinctly. Now that we have two separate datasets we can easily merge them so that every subject has a baseline value and post baseline value. We renamed the cholesterol variable in visit 1 dataset to base_flg and the cholesterol variable in the post baseline dataset to postbase_flg. We have a variable lab which will help us identify that the shift variable we are working on identifies “Cholesterol”. Next we want to assign an order to each of the variables for display purpose. We create two formats and based on those formats we assign the values in the dataset.

```sas
value base_val 1 = 'Abnormal'
  2 = 'Normal'
  3 = 'Not Available';
value pb_val 1 = 'Abnormal'
  2 = 'Normal'
  3 = 'Not Available';
value lab 1='Cholesterol';
```

In the above two formats we are basically assigning an order, meaning in the final table we want to see the baseline values in the following order “Abnormal”, “Normal”, “Not Available” and “Total”. Let’s see how we apply that to the dataset.

```sas
data final1;
merge base postbase;
by usubjid lab;
  if base_flg in ('') then baseorder=3;
  else do;
    if base_flg = 'A' then baseorder=1;
    else if base_flg = 'N' then baseorder=2;
  end;
  if postbase_flg in ('', '.') then missing=1;
  else do;
    if postbase_flg = 'A' then abnormal=1;
    else if postbase_flg = 'N' then normal=1;
  end;
run;
```
The first "IF" block assigns the baseline order. The second "IF" block creates the columns "missing", "abnormal" and "normal", these are post baseline values for cholesterol. This is how the dataset "final1" looks.

<table>
<thead>
<tr>
<th>usubjid</th>
<th>trtgrp</th>
<th>lab</th>
<th>base_flg</th>
<th>postbase_flg</th>
<th>baseorder</th>
<th>missing</th>
<th>abnormal</th>
<th>normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>1</td>
<td>A</td>
<td>N</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>1</td>
<td>N</td>
<td>N</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1005</td>
<td>1</td>
<td>N</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Let's look at the second step now.

**STEP II: CALCULATING SUM**

Since we want to column wise sum the data for each treatment group and baseline we'll use a simple "PROC MEANS" to accomplish this.

```
proc means data=final1 noprint;
  by lab baseorder;
  var abnormal normal missing;
  output out=stat1 sum=cnt1 cnt2 cnt3;
run;
```

<table>
<thead>
<tr>
<th>lab</th>
<th>baseorder</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>cnt1</th>
<th>cnt2</th>
<th>cnt3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

The dataset "stat1" consists of counts for baseorder, cnt1, cnt2 and cnt3. In essence we have exactly what we need for the first treatment group. If we look carefully in the above data, we can see that the data for baseorder "3" is missing. In order to have a complete display of data in the final table we have to make up for this missing data. So the next step is filling up the missing data.

**STEP III: FILLING UP THE MISSING DATA**

There can be numerous ways to do it but the simplest is just merging with a dataset which contains a row for each baseorder value (1, 2, and 3) and columns (cnt1, cnt2, cnt3) each with missing values. When we merge the two datasets, the result will have all the baseorder values and in this case baseorder value "3" will have the values for cnt1, cnt2 and cnt3 as "0".

```
proc sort data=stat1 out=temp2(keep=lab) nodupkey; by lab; run;
data temp3;
  set temp2;
  by lab;
  retain cnt1-cnt3 0;
  do base_val=1 to 3;
    output;
  end;
run;
```

<table>
<thead>
<tr>
<th>lab</th>
<th>cnt1</th>
<th>cnt2</th>
<th>cnt3</th>
<th>base_val</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

The above are the contents of dataset "temp3". Next step is just merging the two datasets.

```
data stat2;
  merge temp3 stat1 end=eof;
```
by lab base_val;
keep base_val cnt1-cnt3 lab;
In the above merge we are merging temp3 (dummy data) with the actual data. This is done so that the baseorder values which have actual counts are not replaced by “0”.
Below is how the dataset stat2 looks.

<table>
<thead>
<tr>
<th>lab</th>
<th>cnt1</th>
<th>cnt2</th>
<th>cnt3</th>
<th>base_val</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

Final step is to realign the data so we have a row for each baseorder and every post baseline values.

**STEP IV: FINAL REALIGNMENT**

Finally we just need to output one observation per base_val and each of three counts. For a quick recap cnt1 signifies “Abnormal” post baseline value, cnt2 signifies “Normal” post baseline values and cnt3 signifies “Missing” post baseline values.

data stat3;set stat2;
array val[3] cnt1-cnt3;
do grp = 1 to 3;
   result1 = val[grp];
   output;
end;
drop cnt1-cnt3;
run;

<table>
<thead>
<tr>
<th>lab</th>
<th>base_val</th>
<th>grp</th>
<th>result1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Abnormal</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Normal</td>
<td>1</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Not Available</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Normal</td>
<td>Abnormal</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Normal</td>
<td>Normal</td>
<td>1</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Normal</td>
<td>Not Available</td>
<td>1</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Abnormal</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Normal</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Not Available</td>
<td>0</td>
</tr>
</tbody>
</table>

So we just finished getting the counts for treatment group “1” which is represented by variable “result1”. Using the same process lets’ look at the final counts from treatment group “2”. Here “result2” variable stands for treatment group “2”.

<table>
<thead>
<tr>
<th>lab</th>
<th>base_val</th>
<th>grp</th>
<th>result2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Abnormal</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Normal</td>
<td>1</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Not Available</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Normal</td>
<td>Abnormal</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Normal</td>
<td>Normal</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Normal</td>
<td>Not Available</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Abnormal</td>
<td>1</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Normal</td>
<td>0</td>
</tr>
</tbody>
</table>
To get the final totals we repeat the entire process without filtering the treatment group. The output from the totals looks like below:

<table>
<thead>
<tr>
<th>lab</th>
<th>base_val</th>
<th>grp</th>
<th>result3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Abnormal</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Normal</td>
<td>2</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Not Available</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Normal</td>
<td>Abnormal</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Normal</td>
<td>Normal</td>
<td>1</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Not Available</td>
<td>1</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Abnormal</td>
<td>1</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Normal</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Not Available</td>
<td>0</td>
</tr>
</tbody>
</table>

Finally we merge the three datasets by base_val and grp to get the shift table as below:

<table>
<thead>
<tr>
<th>lab</th>
<th>base_val</th>
<th>grp</th>
<th>result1</th>
<th>result2</th>
<th>result3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Abnormal</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Normal</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Not Available</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Normal</td>
<td>Abnormal</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Normal</td>
<td>Normal</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Normal</td>
<td>Not Available</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Abnormal</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Normal</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Not Available</td>
<td>Not Available</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**STEP V: PROC REPORT**

Now that we have our final dataset ready we can use a "PROC REPORT" to output the final shift table. The columns statement will use the following variables lab, base_val, grp, result1, result2 and result3. All the result variables connote the two treatment groups and the final totals. Let’s look the code and then the output.

```
proc report data=case2f nowd missing headline headskip split='?';
columns ( (lab base_val grp (trtgrp1 result1) (trtgrp2 result2) ("All?Subjects" result3)));
define lab/order order=data "variable" left style=[cellwidth=1.6in] style(header)=[just=left];
define base_val /display order order=data "Baseline" left style=[cellwidth=1.0in] style(header)=[just=left];
define grp /display order order=data "Post-Baseline" left style=[cellwidth=1.0in] style(header)=[just=left];
define result1 /display " " center style=[cellwidth=0.9in];
define result2 /display " " center style=[cellwidth=0.9in];
define result3 /display " " center style=[cellwidth=0.9in];
compute after base_val /style=[cellheight=3pt];
line " ";
endcomp;
run;
```
The output looks like this:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Baseline</th>
<th>Post-Baseline</th>
<th>trtgrp1</th>
<th>trtgrp2</th>
<th>All Subjects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Abnormal</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Normal</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
<td>Normal</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Abnormal</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Normal</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
<td>Normal</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
<td>Abnormal</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Not Available</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
<td>Not Available</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

If we want to look at some specific data in the shift table, like we want to see the count for subjects who had abnormal cholesterol at baseline and at post baseline we can add a compute block to PROC REPORT, something like this:

```plaintext
compute base_val;
    if base_val eq 1 then call define(_col_,"style","style={background=red}");
endcomp;
compute grp;
    if grp eq 1 then call define(_col_,"style","style={background=red}");
endcomp;
```

The CALL DEFINE statement is often used to write report definitions that other people will use in a windowing environment. It is only effective when Output delivery system is used. Let’s look at the new output now.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Baseline</th>
<th>Post-Baseline</th>
<th>trtgrp1</th>
<th>trtgrp2</th>
<th>All Subjects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cholesterol</td>
<td>Abnormal</td>
<td>Abnormal</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Normal</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
<td>Normal</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Abnormal</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Normal</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
<td>Normal</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
<td>Abnormal</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>Not Available</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Not Available</td>
<td>Not Available</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
CONCLUSION
Shift tables can be very effective in looking at changes from one point to another. But one has to be very careful about calculating the shifts properly. We looked into logic of developing the shift tables and then the actual SAS code to accomplish the same.

REFERENCES
SAS V9 Online Documentation

CONTACT INFORMATION
Author Name          Monal Kohli
Company               Smith Hanley Consulting
Address               114 Emily Pl
City state ZIP        Parsippany NJ 07054
Work Phone:           5168155727
Email:                monal.kohli@gmail.com

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.

APPENDIX 1 – Code to Generate Shift Table

```sas
options mprint mlogic symbolgen;
%macro shift/parmbuff;
  %let num=1;
  %let lengthm=%scan(&syspbuff,&num);
  %do %while(&lengthm ne);
    %let num=%eval(&num+1);
    %let lengthm=%scan(&syspbuff,&num);
  %end;
  %if &num eq 5 %then %do;
    %let indata=%scan(&syspbuff,2,' ');
    %let trtgrp=%scan(&syspbuff,3,' ');
    %let outdata=%scan(&syspbuff,4,' ');
    %let visit=%scan(&syspbuff,5,' ');
  %end;
  %else %do;
    %if &num eq 4 %then %do;
      %let indata=%scan(&syspbuff,2,' ');
      %let outdata=%scan(&syspbuff,3,' ');
      %let visit=%scan(&syspbuff,4,' ');
    %end;
  %end;
data base;
set visit1(drop=visit);
  %if &num eq 5 %then if trtgrp=&trtgrp; %else ;
run;
proc sort data=&indata;by usubjid trtgrp lab;run;
proc sort data=base;by usubjid trtgrp lab ;run;
data baseline (keep=usubjid base_value lab trtgrp)
```
postbaseline (keep=usubjid postbase_val lab trtgrp) ;
set &indata(in=a) base(in=b);
   %if &num eq 5 %then   if trtgrp=&trtgrp; %else ;
      if b then do; base_value = cholesterol; output baseline;end;
      if a then do; postbase_val= cholesterol; output postbaseline;end;
run;

proc sort data=baseline;by usubjid lab;run;
proc sort data=postbaseline;by usubjid lab;run;

data merge&outdata;
merge baseline postbaseline;
by usubjid lab;
if base_value = ' ' then base_val=3;
else do;
    if base_value = 'A' then base_val=1;
    else if base_value = 'N' then base_val=2;
end;
if postbase_val = ' ' then missing=1;
else do;
    if postbase_val = 'A' then ABNORMAL=1;
    else if postbase_val = 'N' then NORMAL=1;
end;
run;
proc sort data=merge&outdata;by lab base_val;run;

proc means data=merge&outdata noprint;
by lab base_val;
var abnormal  normal missing ;
output out=stat1 sum=cnt1 cnt2 cnt3 ;
run;

proc sort data=stat1 out=temp2(keep=lab) nodupkey;by lab;run;

data temp3;
set temp2;
by lab;
    retain cnt1-cnt3 0;
    do base_val=1 to 3;
        output;
    end;
run;

data stat2;
merge temp3 stat1 end=eof;
by lab base_val;
keep base_val cnt1-cnt3 lab ;
    if cnt1 eq . then cnt1=0;
    if cnt2 eq . then cnt2=0;
    if cnt3 eq . then cnt3=0;
run;

data stat3;set stat2;format base_val base_val. grp pb_val. lab lab.;
array val[3] cnt1-cnt3;
do grp = 1 to 3;
   %if &num eq 5 %then %do;
      %let val=&trtgrp;
    %end;
%else %do;
data &outdata;
merge &indata.trtgrp1 &indata.trtgrp2 allsubs&pos;
by base_val grp;
format base_val base_val. grp pb_val. lab lab.;
run;
%mend;
%merge_all(indata=visit2,outdata=case2f);
ods rtf file = "C:\Output\shift_layout.RTF";
proc report data=case2f nowd missing headline headskip split='?';
columns ( (lab base_val grp ("trtgrp1" result1) ("trtgrp2" result2) ("All?Subjects" result3)));

define lab/order order=data "Laboratory Test" left style=[cellwidth=1.6in] style(header)=[just=left];
define base_val /display order order=data "Baseline" left style=[cellwidth=1.0in] style(header)=[just=left];
define grp / display order order=data "Post-Baseline" left style=[cellwidth=1.0in] style(header)=[just=left];
define result1 /display " " center style=[cellwidth=0.9in];
define result2 /display " " center style=[cellwidth=0.9in];
define result3 /display " " center style=[cellwidth=0.9in];
compute after base_val /style={cellheight=3pt} line " ";
endcomp;

compute base_val;
   if base_val eq 1 then call define(_col_,"style","style={background=red}");
endcomp;

compute grp;
   if grp eq 1 then call define(_col_,"style","style={background=red}");
endcomp;
run;
ods _ALL_ close;
run;
ods rtf close;
run;quit;