How Should I Combine My Data, Is The Question
Stephen Thompson, Schering-Plough Corporation, Kenilworth, NJ
Asha Sharma, Dun & Bradstreet Corporation, Murray Hill, NJ

ABSTRACT
As SAS® programmers we are often required to combine two or more files and data sets for processing. How many times and how many of us have combined files without really knowing if we have achieved the desired results. The task of combining data sets can be at times scary and laborious. This leads us to the question of what is the most efficient technique or method to use depending on our objective. There are many factors such as file size, resources, operating systems, time constraints, etc. that need to be taken into consideration while making this decision. Often the method chosen is the one we are most familiar with but which may not be the most proficient and the right one for processing.

INTRODUCTION
The techniques we will share will not only take the fear out of using various methods available to us but will also explore various approaches available to us. Some of the methods we will share with you are - Various types of MERGE in a DATA STEP, SET statements in DATA step, PROC APPEND, UPDATE statement, PROC SQL and Table Lookups using PROC FORMAT. Simple examples will be used to highlight the various methodologies mentioned above.

REASONS FOR COMBINING DATA SETS
There are many reasons for combining SAS data sets. Some of the reasons are:

- To add observations from two or more existing SAS data sets to a new data set.
- To combine variables from two or more data sets to a new data set with a new structure.
- To extracting a smaller subset of records or variables.
- To correct overlooked something when creating the initial data set.

1. THE MERGE STATEMENT
The MERGE statement is one of the most powerful, flexible, and most utilized tools for combining SAS data sets. The MERGE statement is used to combine observations from at least two SAS data sets into single observations in a new SAS data set. The way the SAS system combines records is highly dependent on whether the MERGE statement is accompanied with a BY statement.

SYNTAX
MERGE data-set1 <options>
   Data_set2 <options>
   .
   .
   .
   data_setn <options>;
run;

CONSIDERATIONS
- It is important to keep in mind that when merging SAS data sets the original SAS data sets remain unchanged. While working on a data step the SAS system maintains a temporary data structure in the computer memory called the program data vector (PDV). If some variable(s) appear in more than one data set the value from the right most data set named on the merge statement will be used. The values in the program data vector (PDV) are over written as the merge proceeds.
- If two or more variables have the same name, they must also have the same type. The length is determined by the first variable encountered.
- Unequal number of records in a data set will create a new data set with observations equal in number to that of the largest data set. As the smaller data sets are exhausted, values of their variables are set to missing. The only exception is when the IN= option is used. The IN= option is used to indicate whether the data set contributed data to the current observation.
- Only if it is positively known the data are in the correct order and that no observations are missing is it justified to merge data sets without a BY variable. If there are no unique BY variables and observation(s) are missing or out of place, the result may be incorrect. Also, when using a BY statement the data must be sorted by the BY variable(s).
- The BY variable(s) from each contributing data set should have the same type and length.
- If there are duplicate observations for the BY variable(s), and if the number of observations varies by the BY group, the output data set will not contain missing values for the input data sets that did contribute values. The PDV retain values from the last observation of the shorter BY group, until the longest BY group is processed.

EFFICIENCY
Merging data sets is memory, CPU, and I/O time sensitive. Also, data sets need to be sorted in order to be merged with a BY statement. In order to minimize these and other constraints when merging SAS data sets, below are a few efficiency techniques to consider.

- Use DROP or KEEP statement in combination with DROP= or KEEP= data set option on the merge statement to get rid of variable(s) you do not need in the output data set. Eliminate variable(s) that you need in the PDV in the current data step from the output data set (The less variables in the input data sets the more resources you save).
- Create indexes for large data sets using the DATASETS procedure.
- Know the data you are working with.
CONSTRAINTS
These are a few pitfalls using the MERGE statement:
- The MERGE statement reads each data set record by record and compares each observation’s variable value(s) and creates a single observation in a new data set.
- When using a BY statement in conjunction with a MERGE statement – data must be sorted by the BY variable.

EXAMPLE
The example below encompasses logic for efficiency as well as other data set options available with the MERGE statement:

```sas
proc datasets lib=a nolist;
    Modify setup;
    Index create acct;
run;

data mrg (index=(acct));
    merge temp(in=ina keep=acct t1 t2)
        a.setup(in=inb keep=acct age salary);
    drop = age;
    if age > 54;
    by acct;
    if inb;
run;
```

2. TABLE LOOKUP
Table Lookup is a forgotten tool by many SAS users. However, it has the potential for being the most useful and productive tool available in the SAS System. Although many SAS programmers have used PROC Format to assign descriptive labels to values in a SAS Data Set, there are a lot more ways in which to use PROC FORMAT. Here we will take a more detail look at some of the capabilities of Formats.

SYNTAX
PROC FORMAT Statement
PROC Format <option-list>;

CONSIDERATIONS
- Formats exist independently of variables and must be associated with them by the use of a Format statement in a procedure or Data Step. When you create formats using the PROC Format code, the formats are not automatically linked to any variables you may use in any subsequent procedures or steps. You have to create a format before you can associate it with a variable.
- Formatted values are always character variables. If numeric values are needed for calculations, convert character to numeric by using the input function.
- The techniques we will explore to combine SAS Data Sets involves creating and storing your own user define formats (using the Value statement) and creating a user-defined format from a SAS Data Set using the Control Data Set (CNTLIN).
- The value, invalue or picture statement is generally used when you have a short list of values. While the (Input Control Data Set) CNTLIN option is used when you have a long list of items.

EFFICIENCY
- If you plan on using the formats repeatedly, store the formats in a permanent SAS data library.
- Use less CPU time than merging, especially since you don’t have to sort the data sets; especially if you are working with very large SAS data sets.
- The data sets maintained their original order.
- The PUT statement can be used in conjunction with the formats to select records and/or create new variable(s).
- Values needing labels can be can be in a range or list form.

CONSTRAINTS
- The biggest constraint of all is that Lookup Tables are harder to comprehend and require more coding.
- PROC FORMAT generates no output of its own.
- The key that is being used for the lookup variable must be unique, duplicate values are not permitted in the data set used to create the formats.
- Format cannot be applied to more than one data set at a time.
- Storage space could be a problem because entire format is loaded into memory.

EXAMPLES
The examples below show the use the PROC FORMAT with the value statement followed by the logic for the Control Data Set (CNTLIN).

Value statement:

```sas
proc format;
  Value sicfmt;
    3124 = 'agriculture'
    4562 = 'mining'
    6725 = 'computer'
    9451 = 'construction'
    other = 'no industry'
  ;
run;

data look1;
    set indstry;
    sicdesc = put(sic,sicfmt.);
run;
```

Control Data Set:
- START – variable to translate. If you are using a range of values, START should be the lower bound and use END as the upper bound.
- LABEL – The label you want to assign to the values.
- FMTNAME – The format name you are creating.
data look2;
  set sic.desc;
  rename sic=start;
  siclabel=label;
  fmtname='$sicfmt';
run;

proc format cntlin=look2;
run;

data look3 (keep=sic msa name addr);
  set look.orig;
  p90_sic=input(put (sic,sicfmt.),1.0);
run;

proc print data=look3;
run;

3. UPDATE STATEMENT
The UPDATE statement is generally used to combine data from a master SAS data set and a transaction SAS data set. When combining SAS data sets you can either change the value(s) in the master SAS data set, add observations and/or variable(s) to the master SAS data set.

SYNTAX
UPDATE master-sas-data-set
  Transaction-sas-data-set;
  BY variable-name;
  END=variable-name;

CONSIDERATIONS
• There must be exactly two SAS data sets on an UPDATE statement and the master data set must be listed first.
• Both the master-sas-data-set and the transaction-sas-data-set must have at least one unique variable in both SAS data sets. The master data set cannot contain any duplicate observations; else a warning message is issued. However, the transaction data sets can have duplicate observations.
• If the transaction data set has fewer variables than the master data set, only the variables that are in both data sets will be updated.
• The UPDATE statement must be accompanied by a BY statement with one or more BY variable.
• The SAS data sets must be sorted by the BY variable(s) or appropriate indexes.
• If the transaction data set contains additional records by the By variable(s) that are not in the master data set, the new observations will be added to the new data set.
• Regular missing values in a transaction SAS data set does not replace values in the master SAS data sets. If you intent on replacing values in a master SAS data set with missing value, the MISSING statement must be used on the data step.

EFFICIENCY
Similar to MERGE you should avoid having unnecessary variables in the data sets. This will improve processing time as well as save space. Therefore, consider the following efficiency techniques when updating SAS data sets:
• Use DROP or KEEP statement in combination with DROP= or KEEP= data set option on the merge statement to get rid of variable(s) you do not need in the output data set. Eliminate variable(s) that you need in the PDV in the current data step from the output data set (The less variables in the input data sets the more resources you save).
• Create indexes for large data sets using the DATASETS procedure.
• Create transaction data set with only those variables that are being updated.

CONSTRAINTS
• The data must be sorted by the BY variable.
• The UPDATE statement processes the data sequentially.
• It compares each observation’s variable value(s) and creates a single observation in a new data set.

EXAMPLE
The examples below will illustrate the use of the UPDATE statement to update a master data set with a transaction data set that contains regular and special missing value characters.

/* Master Data Set */
data Final;
  Input id grade;
  Cards;
  1104   10
  1146  90
  1511  55
  2905  67
  3068  85
; /* Transaction data Set */
data Makeup;
  Input id grade;
  Missing X _ ;
  Cards;
  1104   73
  1146  90
  1511 _
  2905    X
  3068 _
; /* Code to Update Master with Transaction */
data newgrade;
  update final makeup;
  by id;
run;
proc print data=newgrade;
run;
Note:
For numeric variables special MISSING values can be any one of the 26 letters of the alphabet (upper or lower case) or the underscore (_).

For character variables special MISSING values is represented by the underscore (_).

Value(s) in the master data set that we did not want to change were set to missing (.) in the transaction data set.

The underscore (_) changes numeric variables in the master data set to a regular missing value(s), the period (.).

The letter X is used to change other value(s) in the master data set to special missing character.

4. THE SET STATEMENT
The SET statements in a data step let you join observations, much as you do with MERGE. If there are multiple data sets on a single SET statement, the first data set brings in observations from the first data set, the second SET statement brings in observations from another data set and so on. There are three common methods used to combine SAS data sets - Concatenation, Interleaving and Combining SAS data sets using multiple SET statements.

Concatenate SAS Data Sets
When concatenation data sets you can combine up to fifty data sets, one after another in the order they are specified.

Data all;
set a b c d…n;

Interleaving SAS Data Sets
The interleaving option brings in the data set in the sorted order. SAS concatenates the data sets within each by group. The data needs to be sorted by the variables specified in the BY statement prior to Interleaving.

proc sort data= a;
  by var1;
run;

proc sort data=n;
  by var1;
run;

EFFICIENCY
- Interleaving is used more often with data sets that are sorted. If your data sets must be sorted before you interleave them it might be more efficient to concatenate the data sets and then sort the results in one sort step.
- If no processing is required on the SAS data sets, it is more efficient to use PROC APPEND than concatenating the data sets.

CONSIDERATIONS
- When Interleaving SAS data sets, all the data sets must be sorted or index based on the BY variable.
- When there are multiple set statements in a DATA step, processing stops when EOF is reached on any of the data sets.
- With multiple SET statements, if the data sets contain variables with the same name, the values in the subsequent data set(s) will overlay the values in the prior data set(s).

5. THE APPEND PROCEDURE
PROC APPEND adds observations from one SAS data set to the end of another SAS data set. The procedure also writes a note to the SAS log that tells how many observations have been added to the data set.

Syntax
PROC APPEND BASE=|OUT= SAS-data-set
DATA=|NEW= SAS-data-set
FORCE;

The required arguments are:
BASE = SAS data set
OUT  = SAS data set
Name the data set to which you want to add observations. If the APPEND statement cannot find an existing data set with this name, it creates a new data set. The OUT = argument is equivalent to the BASE = argument. Either the BASE or the OUT = argument must be specified.

Options:
DATA = SAS data set
NEW = SAS data set
Name the data set containing observations you want to add to the end of SAS data set specified by the BASE =

argument. The NEW = argument is equivalent to the DATA option.

FORCE

The FORCE option is necessary
- If the DATA= data set contains variables that are not in the BASE= data set.
- Do not have the same type as the variables in the BASE= data set.
- Are longer than variables in the BASE= data set.

Example:

Concatenate SAS data set SMALL to Large.

Method1 - less efficient
data one:
  set large small;
run;

Method2 - more efficient
data one;
  proc append base = large data = small;
run;

In method 1, the SAS system reads all observations in both data sets. In method 2, the SAS system reads only the observations in data set TWO.

EFFICIENCY
To add new observations to a SAS data set, PROC APPEND is the better choice. In SET statement, the SAS system must process all the observations in both data sets to create a new one. The APPEND procedure bypasses the processing of the data in the original data set and adds new observations to the end of the original data set. This procedure is especially useful if observations are frequently added to a SAS data set.

6. THE SQL PROCEDURE
The Structured Query language is the standardized language used to retrieve and update data stored in relational tables (or databases). A table in SQL is simply another term for SAS data set or view. One of the most powerful features of SQL is the ability to join multiple tables. When two tables are specified, each row of table A is matched with all rows of table B to produce an internal or intermediate table. Conceptually join operates by taking Cartesian product (product of number of rows in each of the source tables involved). Joins can be put in categories based on common characteristics. The two most significant are:

Inner Join
Returns only matching rows from the tables that meet the criteria in the where clause. Inner Join can be performed on up to 16 tables in a query expression.

For example we want the names of all the employees in Dept Sales. The employee names are in data.employee, the department names and employee name are in data.dept. The code below will extract the requested data:

proc sql;
  select *
  from data.dept a,
       data.employee b,
  where dept_num = 'Sales' and
    a.emp_name = b.emp_name;
run;

To extract the same data using traditional SAS code we need to concern ourselves with sorting the data sets and setting the code for match merge. The required code would be:

proc sort data=data.employee ;
  by emp_name;
run;

proc sort data=data.dept;
  by emp_name;
run;

data final;
  merge data.employee
        data.dept;
  by emp_name;
  if ( dept_num='Sales');
run;

proc print data=final ;
  var emp_name;
run;

Outer Join
An outer join is one that includes all rows from one or more tables in the query. All rows are subject to where clause. Outer joins are of three kinds: left, right and full. For example if we want the list of employees who are in data set employee but may not be in data set dept. Using the above code will exclude the employees that are not in the data dept. The code for outer join would be:

proc sql ;
  select *
  from data.employee left join  a
        data.dept b
  on a.emp_name = b.emp_name ;
run;

In the from clause we have alias the table names as a and b for ease of reference in the rest of the query. This code will return all the employee names from the table employee and will return department name for employee names that are in both the tables.

In summary in left join all rows from the first tables in from clause are returned in addition to those that meet the criteria. In the right join, all rows from the second table in the form clause are returned in addition to the ones that meet the criteria. In full join all rows that
match the criteria are returned, as well as, all the rows from both
tables that do not, subject to the where clause.

The following SAS program will also give us the same results.
As before we have two sorts, data step and PROC print.

```sas
proc sort data=dat.employees ;
   by emp_name;
run;

proc sort data=dat.dept;
   by emp_name;
run;

data merge;
   merge dat.employee(in=a)
       dat.dept;
      by emp_name;
      if a;
run;

proc print data=merge;
run;
```

**BENEFITS**
- Less complex code. A lot of work can be done by with
  minimum typing, coding and debugging can be done quickly.
- Quicker ad hoc reports
- Multiple DATA/PROC steps can be combined in SQL step.
- For small data sets PROC SQL is faster.
- Non SAS programmers can understand the code.
- Task specific SQL code may exist for another RDBMS that
  can be exported in SQL.

**CONSTRAINTS**
- Data step can read up to 100 data sets, Outer Join in SQL
  can only be performed on two tables at a time.
- With PROC SQL, a table can not be opened for input and
  output simultaneously.

**THINGS TO CONSIDER**
What are the tables?
What are the subsetting conditions?
What is the linking conditions?
What columns should be kept?

**CONCLUSION**
There are various methods available for combining SAS data
sets. Becoming familiar with several of the techniques for
merging data will give you more opportunities for selecting the
most efficient and direct method to get the job done. The authors
of this paper hope that by summarizing the techniques this way
will make it easier for users to determine when to apply the right

**REFERENCES**
SAS Institute Inc., SAS Language: Reference, Version 6, First

SAS Institute Inc., SAS Procedures Guide, Version 6,

**TRADEMARK**
SAS® is a register trademark or trademark of SAS
Institute Inc. in the USA and other countries.

**AUTHORS CONTACT INFORMATION**
Stephen Thompson
Email: stephen.thompson@spcorp.com

Asha Sharma
Email: sharmaa@dnb.com