PROC SQL vs. DATA STEP
When we want to combine data horizontally, PROC SQL joins and DATA step match-merge can produce identical results when performing a one-to-one merge or a one-to-many merge. But when we do a many-to-many merge, PROC SQL and DATA step merge will produce different results. Let’s see a simple example:

```sas
data A;
  x=1; y=10; output;
  x=1; y=15; output;
  x=1; y=18; output;
  x=3; y=20; output;
run;

data B;
  x=1; z=15; output;
  x=1; z=12; output;
  x=3; z=30; output;
run;

proc sort data=a; by x; run;
proc sort data=b; by x; run;

data c1;
  merge a(in=aa) b(in=bb);
  by x;
  if aa;
note: merge statement has more than one data set with repeats of BY values.
run;
```

NOTE: MERGE statement has more than one data set with repeats of BY values.
NOTE: There were 4 observations read from the data set WORK.A.
NOTE: There were 3 observations read from the data set WORK.B.
NOTE: The data set WORK.C1 has 4 observations and 3 variables.

proc sql;
create table c2 as
select a.x, a.y, b.z
from a, b
where a.x=b.x;
quit;

NOTE: Table WORK.C2 created, with 7 rows and 3 columns.

The following are datasets C1 and C2; they are quite different:

<table>
<thead>
<tr>
<th>Dataset C2:</th>
<th>Dataset C1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
</tr>
</tbody>
</table>

From the above example, we can clearly see that when there is a many-to-many match on the values of the BY variable, a DATA step match-merge probably does not produce the desired output because the output data set will not contain all of the possible combinations of matching observations. On the other hand, when the datasets are joined using PROC SQL, each match appears as a separate observation in the output dataset, this probably is a more desired output.

Clinical Trial Example
Let's illustrate this with a real life example. Suppose we need to create creatinine clearance records for each visit. In the data, there is a measurement of serum creatinine by using the Cockroft-Gault formula:

For Males: \[
\text{CCr (ml/min)} = \frac{(140-\text{age}) \times \text{body weight (kg)}}{\text{creatinine (umol/L)} \times 0.814}
\]

For Females: \[
\text{CCr (ml/min)} = \frac{(140-\text{age}) \times \text{body weight (kg)}}{\text{creatinine (umol/L)} \times 0.814}
\]

In order to calculate the creatinine clearance, a patient's weight is needed. Weight measurements are stored in the VITAL dataset, and there are multiple weight measurements for different dates. We need to merge the LAB dataset with the VITAL dataset to get the last available weight on or before serum creatinine measurement date. This is a many-to-many merge, PROC SQL can finish this job quite easily while simply using DATA step match merge will not produce the desired result.

The following code will produce the LAB and VITAL datasets.
First, a fake lab serum creatinine dataset is created.

data lab;
  length lbtestcd $ 10 lbstresu $ 20 ;
  retain lbtestcd "CREAT" lbstresu "UMOL/L";
  do usubjid=1001 to 6000;
    if ranuni(0)<0.5 then sexcd=1;
    else sexcd=2;
    age=floor(47+sqrt(81)*rannor(1688));
    lbdt=floor(14535+sqrt(219950)*rannor(1688));
  do visitnum= 1 to 6 ;
    lbdt=lbdt+30;
  end;
run;
lbstresn = 70+sqrt(200)*rannor(1688);
format lbdt yymmdd10.
label
  usubjid = 'Unique Subject Identifier'
  sexcd = 'Sex Code 1=male 2=female'
  age = 'Age in Years'
  lbdt = 'Parameter Measurement Date'
  lbstresn = 'Parameter Value'
  lbtestcd = 'Parameter Name'
  lbstresu = 'Parameter Unit'
  visitnum = 'Visit identifier (numeric)'
;
output;
end;
end;
run;

NOTE: The data set WORK.LAB has 30000 observations and 8 variables.

proc sort data=lab;
  by usubjid lbdt;
run;

In order to create another vital sign dataset with reasonable date values, we need to get the first lab date for each patient.

proc sort data=lab;
  by usubjid lbdt;
run;

data firstlab;
  set lab;
  by usubjid lbdt;
  if first.usubjid;
  keep usubjid lbdt;
run;

Now we are ready to create a fake vital sign dataset.

data vital(drop=i);
  length vstestcd $ 10 vsstresu $ 20;
  retain vstestcd "WEIGHT" vsstresu "KG";
  do usubjid=1001 to 6000;
    do i= 1 to 10;
      vsstresn = floor(74.5+sqrt(276)*rannor(1688));
      label
        vstestcd = 'Vital Parameter Name'
        vsstresn = 'Vital Parameter Value'
        vsstresu = 'Vital Parameter Unit'
      ;
      output;
    end;
  end;
run;

proc sort data=vital;
  by usubjid;
run;

data vital(drop=lbdt);
  retain vsdt;
  merge vital(in=a) firstlab(in=b);
by usubjid;
if a;
if first.usubjid then vsdt=lbdt-1;
vsdt=vsdt-10;
format vsdt yymmdd10.;
label vsdt = 'Vital Parameter Measurement Date';
run;

NOTE: The data set WORK.VITAL has 50000 observations and 5 variables.

Now that we have two fake datasets ready, we can proceed to merge them in order to calculate the creatinine clearance for each visit, which has a non-missing serum creatinine record and has an available weight on or before such visit. The following PROC SQL code performs the many-to-many merge:

proc sql;
create table crcl  (label='Creatinine Clearance Dataset' )as
select
   "CRCL" format $10.  as lbtestcd label='Parameter Name',
   "ML/MIN" format $20.as lbstresu label='Parameter Unit',
   a.usubjid, a.sexcd, a.age, a.lbdt, a.visitnum,
   a.lbstresn as creat label='Parameter Measurement',
   b.vsstresn as weight label='Weight (KG)',
   b.vsdt  label='Weight Measurement Date'
from lab as a left join vital as b
on a.usubjid=b.usubjid and a.lbdt>=b.vsdt
where a.lbstresn ne .
order by lbtestcd, lbstresu, usubjid, visitnum, lbdt, vsdt
;
quit;

NOTE: Table WORK.CRCL created, with 300000 rows and 10 columns.
NOTE: PROCEDURE SQL used:
   real time           20.44 seconds
   cpu time            1.37 seconds

As illustrated in the above simple example, the PROC SQL join contains all possible combinations of matching observations.

Here is one way to accomplish the same task by using a DATA step:

data crcl_;
set lab(keep=usubjid visitnum lbdt lbstresn sexcd age
   rename=(lbstresn=creat)) end=eof;
flag=0;
do pi=1 to numobs;
   set vital(rename=(usubjid=subid vsstresn=weight))
      point=pi nobs=numobs;
   if usubjid=subid and lbdt >= vsdt then do;
      flag=1;
      output;
   end;
   else if pi=numobs and flag=0 then do;
      weight=.;
      output;
   end;
end;
if eof then stop;
drop subid flag;
Comparing the performance of PROC SQL with a DATA step method, there is a huge CPU time difference for 5000 patients, which is typical population size for an ISS study, PROC SQL is much more efficient in such case.

We then need to sort them to get the last available weight, and then calculate the creatinine clearance value.

```sas
proc sort data=crcl;
  by usubjid visitnum lbdt vsdt;
run;
```

```sas
data crcl_(drop=creat weight);
  set crcl;
  by usubjid visitnum lbdt vsdt;
  if last.visitnum;
    lbstresn=0.85**(sexcd-1)*(140 - age)*weight/(creat*0.814);
    if lbstresn ne .;
run;
```

```sas
NOTE: There were 300000 observations read from the data set WORK.CRCL.
NOTE: The data set WORK.CRCL has 30000 observations and 8 variables.
```

The newly created creatinine clearance dataset is now ready to be concatenated together with the original lab data to create an updated lab dataset ready for statistical analysis.

```sas
data lab;
  set lab crcl ;
run;
```

```sas
NOTE: There were 30000 observations read from the data set WORK.LAB.
NOTE: There were 30000 observations read from the data set WORK.CRCL.
NOTE: The data set WORK.LAB has 60000 observations and 8 variables..
```

```sas
proc sort data=lab;
  by lbtestcd lbstresu usubjid visitnum;
run;
```

The above example shows that PROC SQL merge works well under certain circumstances, which requires a many-to-many merge. In such cases, simply using DATA step match-merge will not get the job done, but would require additional DATA step manipulations, which may involves either lengthy coding or short but resource intensive coding.

**CONCLUSION**

Combining data horizontally is very common in the SAS programming world. For a one-to-one or one-to-many merging, choosing PROC SQL or a DATA step match-merge is just a personal preference, since these two methods will give you identical results. However, when performing a many-to-many merge, PROC SQL is the preferred choice.

**REFERENCE**


CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact the authors at:

<table>
<thead>
<tr>
<th>Wenjie Wang</th>
<th>Jade Huang</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tech Data Services</td>
<td>Merck &amp; Co., Inc.</td>
</tr>
<tr>
<td>Bridgewater, NJ 08807</td>
<td>RY34-A320</td>
</tr>
<tr>
<td>(908) 304-6217</td>
<td>P.O. Box 2000</td>
</tr>
<tr>
<td><a href="mailto:wenjie.wang@novartis.com">wenjie.wang@novartis.com</a></td>
<td>Rahway, NJ 07065</td>
</tr>
</tbody>
</table>

|                         | (732) 594-3854                      |
| xiqun_Huang@merck.com    |                                    |

TRADEMARKS
SAS and all other SAS Institute Inc. product or service names are registered trademarks of SAS Institute Inc. in the US and other countries.