

Why Choose between SAS® Data Step or PROC SQL When You Can Have Both

Rebecca Callaway
SAS Institute Inc.

Michigan SAS Users Group meeting
September 2023



About your presenter

SAS Principal Technical Training Specialist, Rebecca Callaway, teaches by engaging with logic, visuals and analogies to spark critical thinking. She thrives on helping others learn the power of SAS to make their work easier and more efficient. She resides in San Diego, CA with her husband, Ken, and their cat Zigmo.

When she's not teaching technology, she is passionate about connecting with friends and family, enjoying the outdoors and the beauty of Southern California.

Rebecca has worked for SAS since May 2000 where she has instructed students on SAS programming including Base SAS topics, SQL, Macro programming, SAS Enterprise Guide, Office Analytics, Visual Analytics and Customer Intelligence.

Agenda

1. Reading Raw Data
2. Combining Data
3. Accumulating Data
4. Aggregating Data
5. Managing Data
6. Useful Links
7. Q&A

READING RAW DATA

READING RAW DATA

Can you read raw data with PROC SQL?

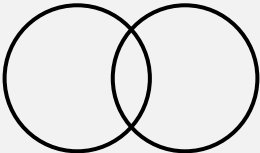
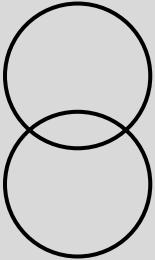
```
data dsrawdata;  
  infile datalines dlm=',';  
  input name $ gender $ age height;  
  datalines;  
Alfred,M,14,69  
Alice,F,13,56.5  
Barbara,F,13,65.3  
;  
run;
```

And the winner is : The Data step



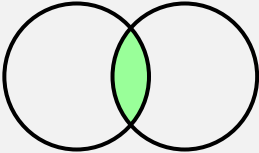
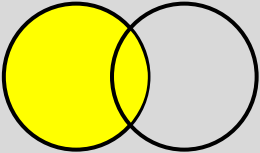
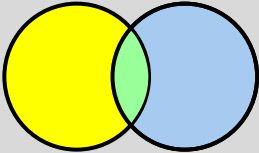
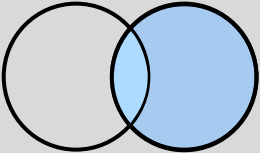
COMBINING DATA

COMBINING DATA TECHNIQUES

Visual	Stacking	PROC SQL	DATA Step	PROC APPEND
	Horizontal - Stack columns and align rows.	Joins	Merge	
	Vertical - Stack rows and align columns.	Set Operators	Concatenate	Append

COMBINING DATA: SQL JOINS

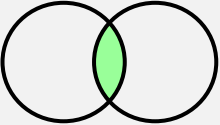
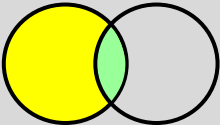
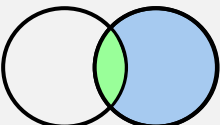
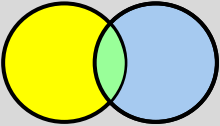
PROC SQL uses joins to combine tables horizontally. Requesting a join involves matching data from one row in one table with a corresponding row in a second table. Matching is typically performed on one or more columns.

Inner Joins	Return only matching rows.			
Outer Joins	Return all matching rows, plus nonmatching rows from one or both tables.	 Left	 Full	 Right

COMBINING DATA: DATA STEP

```
/*Data needs to be pre-sorted*/  
proc sort data=sashelp.prdsal2 out=prdsal2;  
    by state;  
run;  
  
proc sort data=sashelp.us_data out=us_data (  
    by statename;  
run;  
  
/*rename variables for common by variable*/  
data dsmerge;  
    merge prdsal2(in=inprd)  
          us_data(in=inus rename=(statename=state));  
    by state;  
    if inprd and inus;  
    keep country county product state population_2010;  
run;
```



	DATA Step Merge	PROC SQL Join
	<pre> data empsauc; merge empsau(in=Emps) phonec(in=Cell); by EmpID; if Emps=1 and Cell=1; run; </pre>	<pre> proc sql; create table empsauc as select First, Gender, e.EmpID, Phone from empsau e, phonec p where e.EmpID=p.EmpID; quit; </pre>
	<pre> data empsauc; merge empsau(in=Emps) phonec; by EmpID; if Emps=1; run; </pre>	<pre> proc sql; create table empsauc as select First, Gender, e.EmpID, Phone from empsau e left join phonec p on e.EmpID=p.EmpID; quit; </pre>
	<pre> data empsauc; merge empsau phonec(in=Cell); by EmpID; if Cell=1; run; </pre>	<pre> proc sql; create table empsauc as select First, Gender, p.EmpID, Phone from empsau e right join phonec p on e.EmpID=p.EmpID; quit; </pre>
	<pre> data empsauc; merge empsau phonec; by EmpID; run; </pre>	<pre> proc sql; create table empsauc as select First, Gender, coalesce(e.EmpID, p.EmpID), Phone from empsau e full join phonec p on e.EmpID=p.EmpID; quit; </pre>

COMPARING MERGE AND SQL JOIN

	Match-Merge	SQL Inner Join
Number of datasets/ size	No limit to number or size other than disk space.	Max number of tables in join 256.

COMPARING MERGE AND SQL JOIN

	Match-Merge	SQL Inner Join
Number of datasets/ size	No limit to number or size other than disk space.	Max number of tables in join 256.
Data processing	sequential so that observations with duplicate BY values are joined one-to-one.	Cartesian product for duplicate BY values.

COMPARING MERGE AND SQL JOIN

	Match-Merge	SQL Inner Join
Number of datasets/ size	No limit to number or size other than disk space.	Max number of tables in join 256.
Data processing	sequential so that observations with duplicate BY values are joined one-to-one.	Cartesian product for duplicate BY values.
Output datasets	Multiple data sets can be created.	Only one data set can be created

COMPARING MERGE AND SQL JOIN

	Match-Merge	SQL Inner Join
Number of datasets/ size	No limit to number or size other than disk space.	Max number of tables in join 256.
Data processing	sequential so that observations with duplicate BY values are joined one-to-one.	Cartesian product for duplicate BY values.
Output datasets	Multiple data sets can be created.	Only one data set can be created
Complex business logic	using IF-THEN or SELECT/WHEN logic.	CASE logic; however, not as flexible as DATA step syntax.

COMPARING MERGE AND SQL JOIN

	Match-Merge	SQL Inner Join
Number of datasets/ size	No limit to number or size other than disk space.	Max number of tables in join 256.
Data processing	sequential so that observations with duplicate BY values are joined one-to-one.	Cartesian product for duplicate BY values.
Output datasets	Multiple data sets can be created.	Only one data set can be created
Complex business logic	using IF-THEN or SELECT/WHEN logic.	CASE logic; however, not as flexible as DATA step syntax.
Sorted/indexed datasets	Prerequisite for merging	Not necessary

COMPARING MERGE AND SQL JOIN

	Match-Merge	SQL Inner Join
Number of datasets/ size	No limit to number or size other than disk space.	Max number of tables in join 256.
Data processing	sequential so that observations with duplicate BY values are joined one-to-one.	Cartesian product for duplicate BY values.
Output datasets	Multiple data sets can be created.	Only one data set can be created
Complex business logic	using IF-THEN or SELECT/WHEN logic.	CASE logic; however, not as flexible as DATA step syntax.
Sorted/indexed datasets	Prerequisite for merging	Not necessary
Join condition	Equality only	Unequal joins can be performed.

COMPARING MERGE AND SQL JOIN

	Match-Merge	SQL Inner Join
Number of datasets/ size	No limit to number or size other than disk space.	Max number of tables in join 256.
Data processing	sequential so that observations with duplicate BY values are joined one-to-one.	Cartesian product for duplicate BY values.
Output datasets	Multiple data sets can be created.	Only one data set can be created
Complex business logic	using IF-THEN or SELECT/WHEN logic.	CASE logic; however, not as flexible as DATA step syntax.
Sorted/indexed datasets	Prerequisite for merging	Not necessary
Join condition	Equality only	Unequal joins can be performed.
Same named variables	Same named BY variables must be available in all data sets.	Same named variables do not have to be in all data sets.

COMBINING DATA: PROC SQL

```
/*Proc SQL Join*/  
proc sql;  
    create table sqljoin as  
        select country, county, product, statename, population_2010  
            from sashelp.prdsal2 as p, sashelp.us_data as us  
            where p.state = us.statename;  
quit;
```

Variables don't need to have the same name for the match-merge to work correctly and data does not have to be presorted.

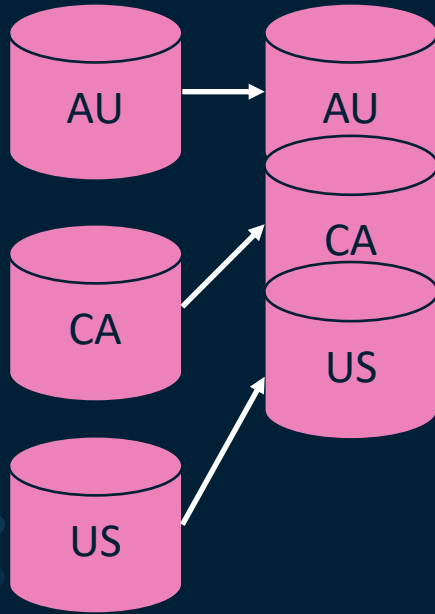
And the winner is : PROC SQL

DEMO – JOINING TABLES

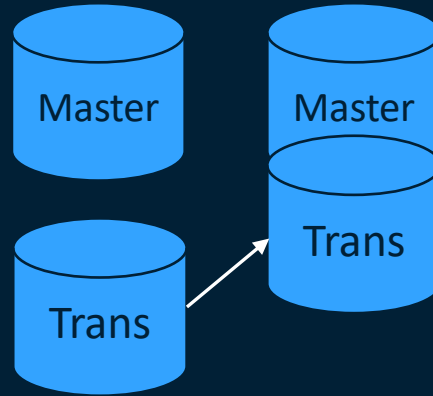
And the winner is : PROC SQL

VERTICAL CONCATINATION

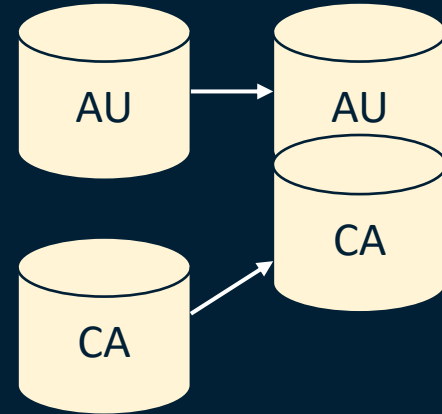
DATA STEP CONCATENATE



PROC APPEND



PROC SQL



VERTICAL CONCATINATION COMPARISON

	Data Step Concatenate	PROC APPEND	SQL Concatenate
Concepts	SET statement	PROC APPEND	SET operators EXCEPT, INTERSECT, UNION
Number of tables that can be stacked	Unlimited	2 at a time	2 at a time
Grow data wide simultaneously	Y	N	N
Advantages	Arrays, hash objects, do loops, ability to write to multiple output datasets in one read	Only observations of appending data set are read (efficiency)	Simple syntax but the data step is truly the winner

DEMO – VERTICAL CONCATINATION

And the winner is : DATA Step

ACCUMULATING DATA

ACCUMULATING DATA

```
/*3 Accumulating Data*/  
/*Proc sql accumulating data*/  
data shoes;  
    set sashelp.shoes;  
    obs=_n_;  
run;  
  
proc sql;  
    create table sqlrunning as  
        select region, product, sales,  
            (select sum(a.sales) from shoes as a  
             where a.obs <= b.obs) as Running_total  
            from shoes as b;  
quit;
```



ACCUMULATING DATA

```
/*Data step accumulating data */  
data dsrunning;  
    set shoes;  
    keep region product sales running_total;  
    running_total + sales;  
run;
```

And the winner is :The Data Step



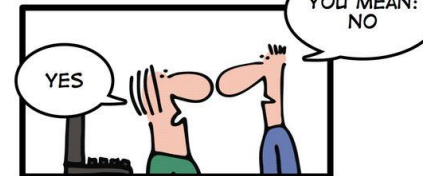
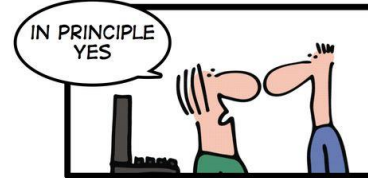
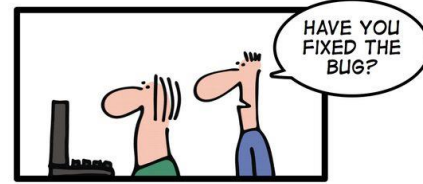
DEMO – ACCUMULATING DATA

And the winner is : Data Step

AGGREGATING DATA

AGGREGATING DATA

SIMPLY EXPLAINED



BOOLEAN LOGIC

AGGREGATING DATA

The Sashelp.BWeight data set provides 1997 birth weight data from National Center for Health Statistics. The data record live, singleton births to mothers between the ages of 18 and 45 in the United States who were classified as black or white. The data set contains 50,000 observations.

Figure 1.14 Sashelp.bweight — Infant Birth Weight

Sashelp.bweight --- Infant Birth Weight

The CONTENTS Procedure

Variables in Creation Order			
#	Variable	Type	Len Label
1	Weight	Num	8 Infant Birth Weight
2	Black	Num	8 Black Mother
3	Married	Num	8 Married Mother
4	Boy	Num	8 Baby Boy
5	MomAge	Num	8 Mother's Age
6	MomSmoke	Num	8 Smoking Mother
7	CigsPerDay	Num	8 Cigarettes Per Day
8	MomWtGain	Num	8 Mother's Pregnancy Weight Gain
9	Visit	Num	8 Prenatal Visit
10	MomEdLevel	Num	8 Mother's Education Level

The First Five Observations Out of 50,000

Weight	Black	Married	Boy	MomAge	MomSmoke	CigsPerDay	MomWtGain	Visit	MomEdLevel
4111	0	1	1	-3	0	0	-16	1	0
3997	0	1	0	1	0	0	2	3	2
3572	0	1	1	0	0	0	-3	3	0
1956	0	1	1	-1	0	0	-5	3	2
3515	0	1	1	-6	0	0	-20	3	0

AGGREGATING DATA

```
/*Data step aggregating data */  
/*prep data for summarizing*/  
  
proc sort data=sashelp.bweight out=bweight;  
by visit;  
run;  
/*4.1 finding the last row in a group*/  
  
data dslast;  
    set bweight;  
    by visit;  
    if last.visit;  
run;
```

AGGREGATING DATA

```
/*Data Step Aggregating Data*/
data dsboolean;
  set bweight;
  by visit;
  if first.visit then do;
    wgt4000=0;
    wle2500=0;
  end;
  if weight > 4000 and married=1 and momsmoke=1 then
    wgt4000 + 1;
  else if weight <=2500 and married=1 and momsmoke=1 then
    wle2500 + 1;
  if last.visit;
  label wgt4000 ='over average weight'
        wle2500 ='under average weight';
  keep visit wgt4000 wle2500;
run;
```

AGGREGATING DATA

```
proc sql;  
    create table slast as  
        select *, monotonic() as row_id from bweight  
        group by visit  
        having row_id = max(row_id);  
quit;
```


AGGREGATING DATA

```
/*Proc Sql aggregating data*/  
proc sql;  
    create table sqlboolean as  
        select visit,  
            sum(weight > 4000 and married=1 and momsmoke=1)  
            as wgt4000 'over average weight',  
            sum(weight <=2500 and married=1 and momsmoke=1)  
            as wle2500 'under average weight'  
        from sashelp.bweight  
        group by visit;  
quit;
```

And the winner is : PROC SQL

DEMO – AGGREGATING DATA

And the winner is : PROC SQL

MANAGING DATA

MANAGING DATA

```
data dsdict;  
    set sashelp.vcolumn;  
    keep libname memname name type length;  
    where upcase(name) contains 'ID' and libname='SASHELP'  
and type='num';  
run;
```

NOTE: There were 34 observations read from the data set SASHELP.VCOLUMN.

WHERE UPCASE(name) contains 'ID' and (libname='SASHELP') and
(type='num');

NOTE: The data set WORK.DSDICT has 34 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time	2.86 seconds
user cpu time	1.26 seconds
system cpu time	1.42 seconds
memory	6505.20k
OS Memory	29432.00k



MANAGING DATA

```
proc sql;  
select libname, memname, name, type, length  
      from dictionary.columns  
      where upcase(name) contains 'ID'  
            and libname='SASHELP' and type='num';  
quit;
```

NOTE: Table WORK.SQLDICT created, with 34 rows and 5 columns.

quit;

NOTE: PROCEDURE SQL used (Total process time):

real time	0.77 seconds
user cpu time	0.37 seconds
system cpu time	0.34 seconds
memory	5623.92k
OS Memory	29176.00k

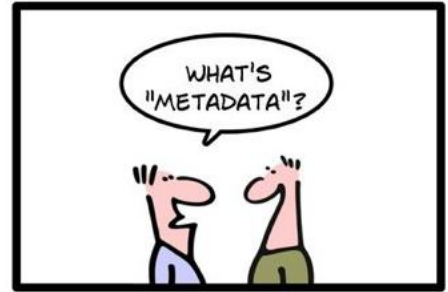
And the winner is :PROC SQL

Investigate common columns

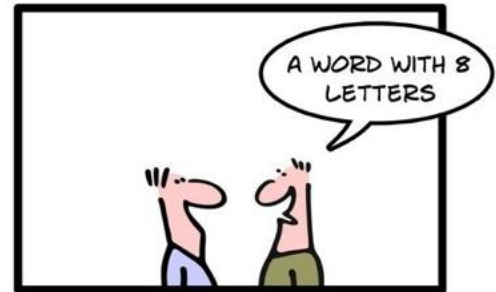
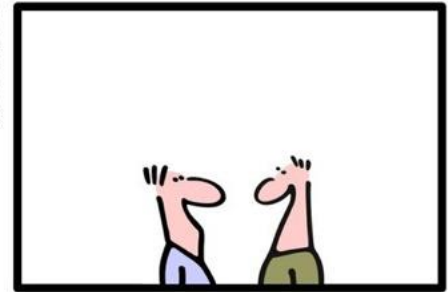
What happens if you do not know your data, and you want SAS to retrieve all same-named columns in a library.

```
proc sql;  
  select name, memname, type, length  
    from dictionary.columns  
   where libname = 'SASHELP'  
   group by name  
   having count(name) > 1  
   order by name;  
  
quit;
```

SIMPLY EXPLAINED:
METADATA



geek & poke



DEMO – MANAGING DATA

And the winner is : PROC SQL

SUMMARY – DATASTEP VS PROC SQL

	Data Step	Proc SQL
Transparent	Yes	No
Optimizer	No	Yes
Provides Control	More	Less
Approach	Micro	Macro
Best	Raw data Repetitive processing across-arrays Locating first & last in group	Joins Metadata Databases

USEFUL LINKS/REFERENCES

- Monotonic function in SAS
- What's in a name, SQL Joins vs Set Operators
- Know thy data, Techniques for Data Exploration
- Huang, Chao "Top 10 SQL trips in SAS", SAS Global Forum 2014
- Shankar, Charu, "#1 SAS programming tip for 2012", SAS Training Post, May 10, 2012

***Special thanks to Charu Shankar for creating this presentation.**

Charu.Shankar@sas.com

***Additional thanks to Michele Ensor for her contributions. Michele.Ensor@sas.com**



Thank you

Questions & Comments

EMAIL Rebecca.Callaway@sas.com

LINKEDIN <https://ca.linkedin.com/in/rebeccazcallaway>

sas.com

