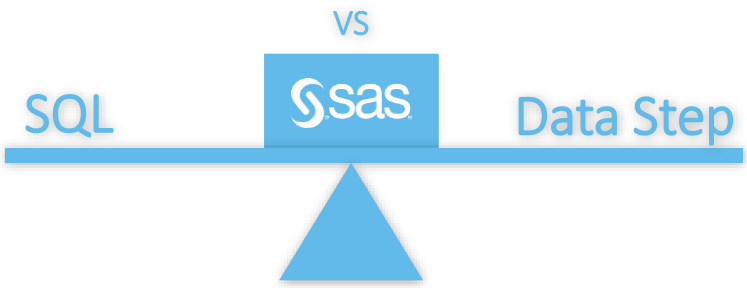
PROC SQL vs. DATA Step Processing

T Winand, Customer Success Technical Team

Agenda

PROC SQL VS. DATA STEP PROCESSING

- Comparison of DATA Step and PROC SQL capabilities
- Joining SAS data using the DATA Step and PROC SQL
- Data Step and SQL Output
- Additional Comparisons
- Additional Resources



Comparison of DATA Step and PROC SQL capabilities

PROC SQL vs. DATA Step

Type of processing

- DATA step is typically sequential processing
- PROC SQL uses an optimizer – may see dissimilar results

DATA Step

- creating SAS data sets (SAS data files or SAS views)
- creating SAS data sets from input files that contain raw data (external files)
- creating new SAS data sets from existing ones by subsetting, merging, modifying, and updating existing SAS data sets
- analyzing, manipulating, or presenting your data
- computing the values for new variables
- report writing, or writing files to disk or tape
- retrieving information
- file management

PROC SQL

- retrieve and manipulate data that is stored in tables or views.
- create tables, views, and indexes on columns in tables.
- create SAS macro variables that contain values from rows in a query's result.
- add or modify the data values in a table's columns or insert and delete rows. You can also modify the table itself by adding, modifying, or dropping columns.
- send DBMS-specific SQL statements to a database management system (DBMS) and retrieve DBMS data.

Capability	DATA Step	PROC SQL
Creating SAS data sets (SAS data files or SAS views)	X	X
Create Indexes on tables	X	X
Creating SAS data sets from input files that contain raw data (external files)	X	
Analyzing, manipulating, or presenting your data	X	X (listing reports)
Writing external files to disk or tape	X	
Computing the values for new variables	X	X
Retrieving system information	X	
File management	X	
Create SAS macro variables that contain values from rows in a query's result	X	X
Send DBMS-specific SQL statements to a database management system (DBMS) and retrieve DBMS data		X

Capability	DATA Step	PROC SQL
Use DO loops	X	
Use Arrays	X	
IF … THEN … ELSE processing	X	X
Use Object Oriented programming with JAVA or Hash objects	X	

Joining SAS data using the DATA Step and PROC SQL

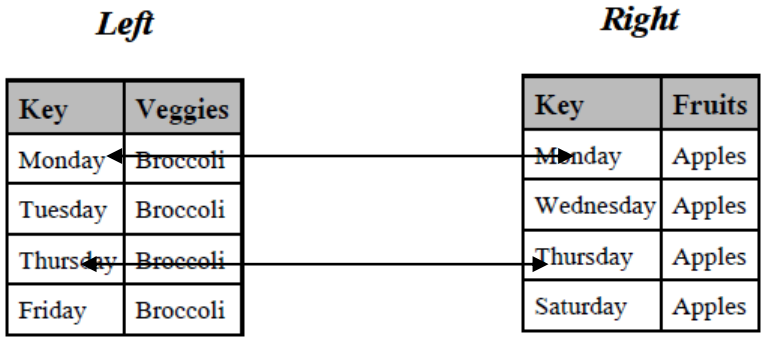
Types of Joins

- Natural
  - Uses no ‘keys’ – typically termed a Cartesian product
- Inner
- Outer Joins
  - Left
  - Right
  - Full

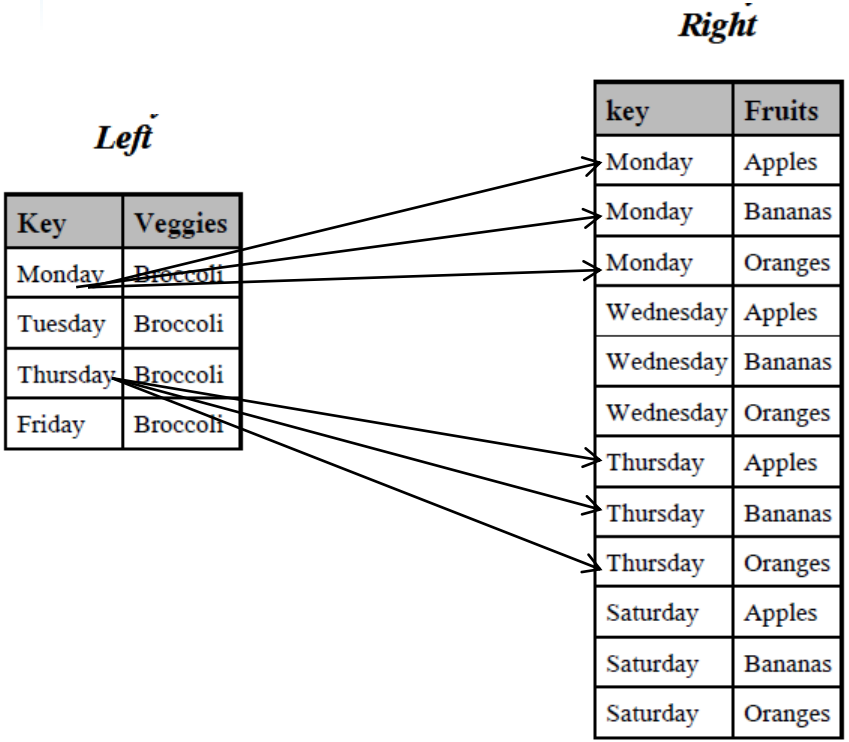
Joining Data

- One to One
- One to Many
- Many to One
- Many to Many

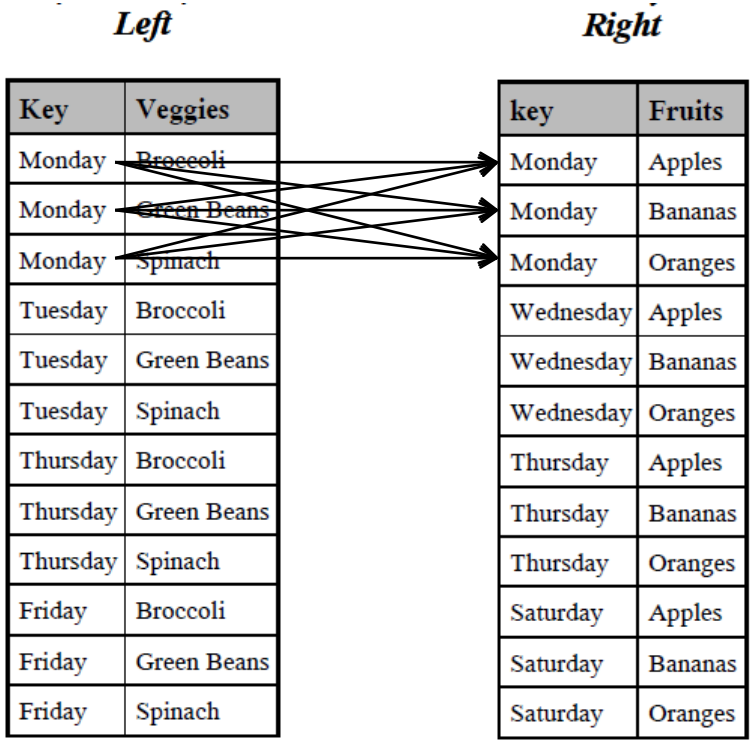
One to One



One to Many



Many to Many



Left

Key	Veggies
Monday	Broccoli
Monday	Green Beans
Monday	Spinach
Tuesday	Broccoli
Tuesday	Green Beans
Tuesday	Spinach
Thursday	Broccoli
Thursday	Green Beans
Thursday	Spinach
Friday	Broccoli
Friday	Green Beans
Friday	Spinach

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

Data Step & SQL Output

INNER JOIN – MANY TO MANY

Data_Inner

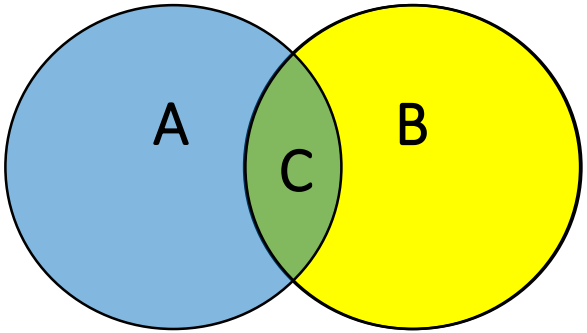
Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Green Beans	Bananas
Monday	Spinach	Oranges
Thursday	Broccoli	Apples
Thursday	Green Beans	Bananas
Thursday	Spinach	Oranges

SQL_Inner

Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Oranges
Monday	Broccoli	Bananas
Monday	Green Beans	Apples
Monday	Green Beans	Oranges
Monday	Green Beans	Bananas
Monday	Spinach	Apples
Monday	Spinach	Oranges
Monday	Spinach	Bananas
Thursday	Broccoli	Apples
Thursday	Broccoli	Oranges
Thursday	Broccoli	Bananas
Thursday	Green Beans	Apples
Thursday	Green Beans	Oranges
Thursday	Green Beans	Bananas
Thursday	Spinach	Apples
Thursday	Spinach	Oranges
Thursday	Spinach	Bananas

Types of Joins

- Inner Join
  - The intersection of two or more sets
  - Return only matching rows



One to One

SAMPLE DATA

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

SQL

```
proc sql;
/*   create table SQL_Join as*/
   select *
   from Left a , Right b
   where a.key = b.key
<groupby>
<orderby>

   ;
quit;
```

Proc SQL supports run group processing

SQL

DEFAULT INNER JOIN - ONE TO ONE

```
proc sql;
/*    create table SQL_Join as*/
    select a.*, b.fruits
    from Left a , Right b
    where a.key = b.key
    ;
quit;
```

One to One Joins
SQL_Join

Key	Veggies	Fruits
Monday	Broccoli	Apples
Thursday	Broccoli	Apples

Data Step

DEFAULT JOIN – ONE TO ONE

```
data Data_Merge;
   merge Left Right;
   by key;
run;
```

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

One to One Joins
Data_Merge

Key	Veggies	Fruits
Monday	Broccoli	Apples
Tuesday	Broccoli	
Wednesday		Apples
Thursday	Broccoli	Apples
Friday	Broccoli	
Saturday		Apples

Data Step

INNER JOIN – ONE TO ONE

```
data Data_Inner;
   merge Left(in=left)
         Right(in=right);
   by key;
   if left and right;
run;
```

One to One Joins
Data_Inner

Key	Veggies	Fruits
Monday	Broccoli	Apples
Thursday	Broccoli	Apples

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

SQL

INNER JOIN – ONE TO ONE

```
proc sql;
   select a.*, b.fruits
   from Left a inner join Right b
      on a.key = b.key
   ;
quit;
```

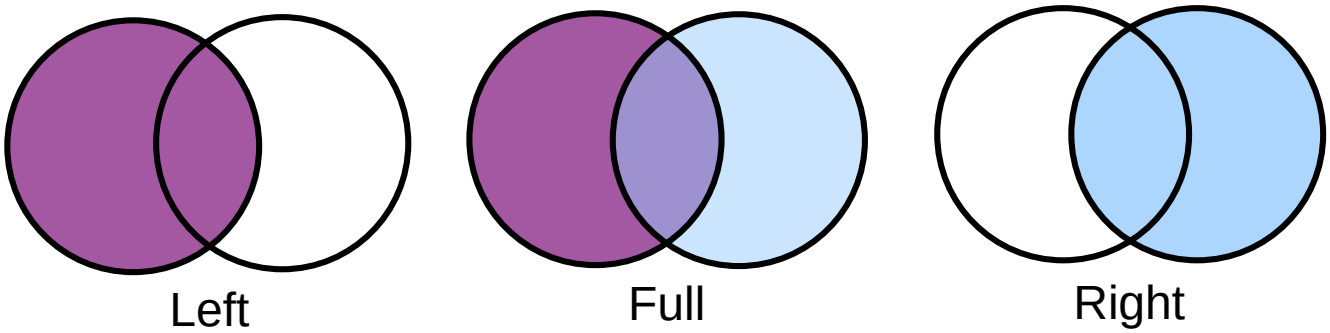
One to One Joins
SQL_Inner

Key	Veggies	Fruits
Monday	Broccoli	Apples
Thursday	Broccoli	Apples

Alternate Syntax

Outer Joins

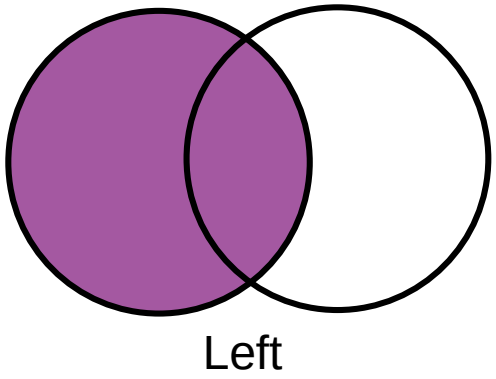
- return all matching rows, plus nonmatching rows from one or both tables
- can be performed on only two tables or views at a time.



Outer Joins

LEFT JOIN

Retrieve the matching rows as well as the non-matches from the left table



Reminder

SAMPLE DATA

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

SQL

LEFT JOIN – ONE TO ONE

```
proc sql;
    select a.*, b.fruits
    from Left a
       Left join
         Right b
       on a.key = b.key
    ;
quit;
```

One to One Joins
SQL_Left

Key	Veggies	Fruits
Monday	Broccoli	Apples
Tuesday	Broccoli	
Thursday	Broccoli	Apples
Friday	Broccoli	

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

Data Step

LEFT JOIN – ONE TO ONE

```
data Data_Left;
   merge Left(in=left)
         Right(in=right);
   by key;
   if left;
run;
```

One to One Joins
Data_Left

Key	Veggies	Fruits
Monday	Broccoli	Apples
Tuesday	Broccoli	
Thursday	Broccoli	Apples
Friday	Broccoli	

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

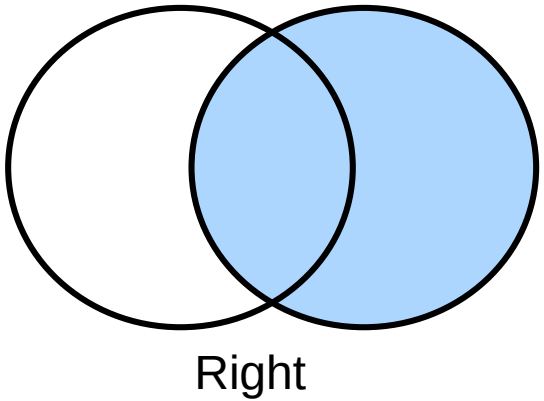
Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

Outer Joins

RIGHT JOIN

Retrieve the matching rows as well as the non-matches from the right table



Reminder

SAMPLE DATA

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

SQL

RIGHT JOIN – ONE TO ONE

```
proc sql;
   select b.key as Key
         , a.Veggies
         , b.Fruits
   from Left a
      right join
        Right b
      on a.key = b.key
   ;
quit;
```

SQL_Right

Key	Veggies	Fruits
Monday	Broccoli	Apples
Wednesday		Apples
Thursday	Broccoli	Apples
Saturday		Apples

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

Data Step

RIGHT JOIN – ONE TO ONE

```
data Data_Right;
   merge Left(in=left)
         Right(in=right);
   by key;
   if right;
run;
```

Data_Right

Key	Veggies	Fruits
Monday	Broccoli	Apples
Wednesday		Apples
Thursday	Broccoli	Apples
Saturday		Apples

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

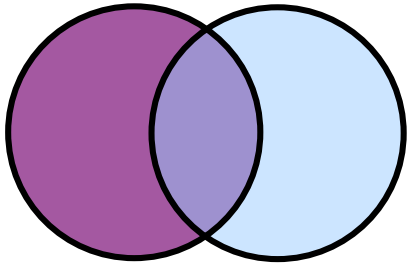
Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

Outer Joins

FULL JOIN

Retrieve the matching rows as well as the non-matches from the left table and the non-matches from the right table.



Reminder

SAMPLE DATA

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

SQL

FULL (OUTER) JOIN – ONE TO ONE

```
proc sql;
/*   create table SQL_Outer as*/
   select a.key as Key,
            a.Veggies,
          b.Fruits
   from Left a
      Full join
        Right b
      on a.key = b.key
   ;
quit;
```

The SAS System

Key	veggies	fruits
Friday	Broccoli	
Monday	Broccoli	Apples
		Apples
Thursday	Broccoli	Apples
Tuesday	Broccoli	
		Apples

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

The COALESCE Function

The COALESCE function returns the value of the first non-missing argument.

General form of the COALESCE function:

COALESCE(*argument-1*,*argument-2*<, *...argument-n*)

SQL

FULL (OUTER) JOIN – ONE TO ONE

```
proc sql;
/*   create table SQL_Outer as*/
   select coalesce(a.key,b.key) as
Key
         , a.Veggies
         , b.Fruits
   from Left a
      Full join
        Right b
      on a.key = b.key
   ;
quit;
```

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

SQL_Outer

Key	Veggies	Fruits
Monday	Broccoli	Apples
Tuesday	Broccoli	
Wednesday		Apples
Thursday	Broccoli	Apples
Friday	Broccoli	
Saturday		Apples

Data Step

FULL (OUTER) JOIN – ONE TO ONE

```
/* Just a simple merge */

data Data_Outer;
   merge Left(in=left)
         Right(in=right);
   by key;
run;
```

Data_Outer

Key	Veggies	Fruits
Monday	Broccoli	Apples
Tuesday	Broccoli	
Wednesday		Apples
Thursday	Broccoli	Apples
Friday	Broccoli	
Saturday		Apples

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

Data Step & SQL Output

One to Many

SAMPLE DATA

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

SQL

INNER JOIN - ONE TO MANY

```
/* inner join with one to many data */
title 'SQL_join';
proc sql;
  select a.key, a.veggies, b.fruits
    from left a, right b
         where a.key=b.key
         ;
quit;
```

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

SQL_Join

Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges

Data Step

INNER JOIN - ONE TO MANY

```
data oneToMany;
   merge left right;
   by key;
run;
title 'data_merge';
proc print; run;
```

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

Data_Merge

Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Tuesday	Broccoli	
Wednesday		Apples
Wednesday		Bananas
Wednesday		Oranges
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges
Friday	Broccoli	
Saturday		Apples
Saturday		Bananas
Saturday		Oranges

Data Step

INNER JOIN - ONE TO MANY

```
title 'Data_Inner';
data oneToMany;
   merge left(in=left) right(in=right);
   by key;
   if left and right;
run;
proc print;
run;
```

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

Data_Inner

Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges

Data Step & SQL Output

LEFT JOIN – ONE TO MANY

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

Data_Left

Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Tuesday	Broccoli	
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges
Friday	Broccoli	

SQL_Left

Key	Veggies	Fruits
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Monday	Broccoli	Apples
Tuesday	Broccoli	
Thursday	Broccoli	Apples
Thursday	Broccoli	Oranges
Thursday	Broccoli	Bananas
Friday	Broccoli	

Data Step & SQL Output

RIGHT JOIN – ONE TO MANY

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

Data_Right

Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Wednesday		Apples
Wednesday		Bananas
Wednesday		Oranges
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges
Saturday		Apples
Saturday		Bananas
Saturday		Oranges

SQL_Right

Key	Veggies	Fruits
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Monday	Broccoli	Apples
Wednesday		Oranges
Wednesday		Bananas
Wednesday		Apples
Thursday	Broccoli	Apples
Thursday	Broccoli	Oranges
Thursday	Broccoli	Bananas
Saturday		Oranges
Saturday		Bananas
Saturday		Apples

Data Step & SQL Output

FULL (OUTER) JOIN – ONE TO MANY

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

Data_Outer

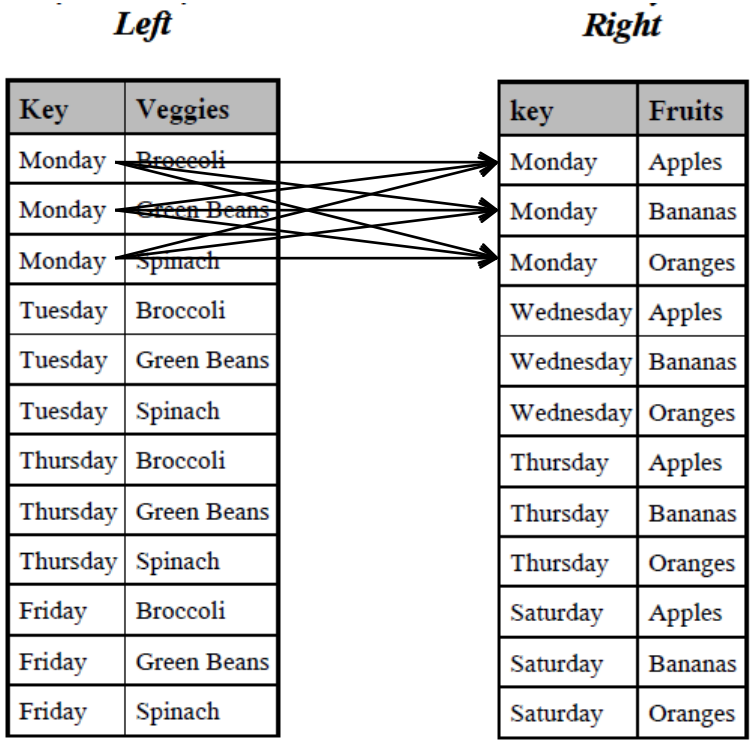
Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Tuesday	Broccoli	
Wednesday		Apples
Wednesday		Bananas
Wednesday		Oranges
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges
Friday	Broccoli	
Saturday		Apples
Saturday		Bananas
Saturday		Oranges

SQL_Outer

Key	Veggies	Fruits
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Monday	Broccoli	Apples
Tuesday	Broccoli	
Wednesday		Oranges
Wednesday		Bananas
Wednesday		Apples
Thursday	Broccoli	Apples
Thursday	Broccoli	Oranges
Thursday	Broccoli	Bananas
Friday	Broccoli	
Saturday		Oranges
Saturday		Bananas
Saturday		Apples

SQL

MANY TO MANY



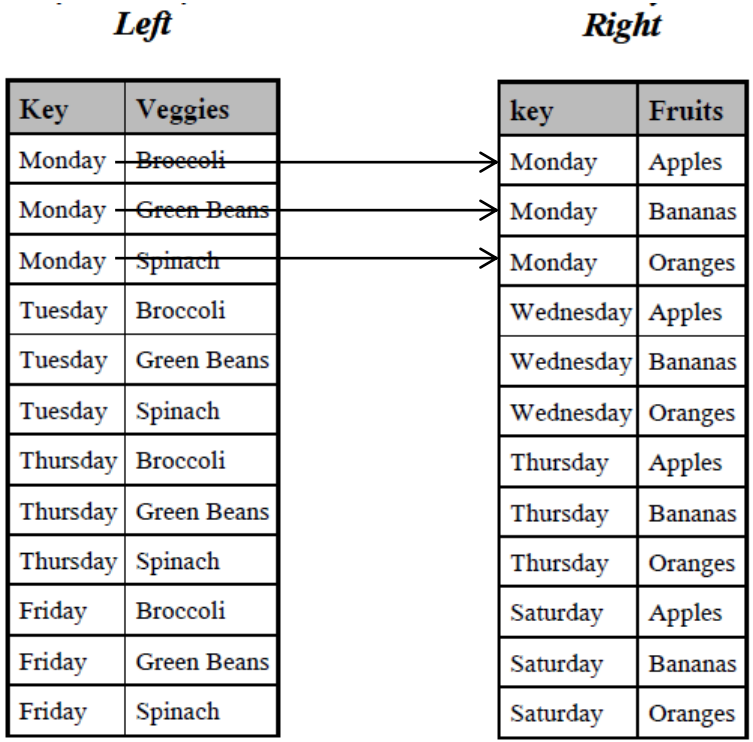
Key	Veggies
Monday	Broccoli
Monday	Green Beans
Monday	Spinach
Tuesday	Broccoli
Tuesday	Green Beans
Tuesday	Spinach
Thursday	Broccoli
Thursday	Green Beans
Thursday	Spinach
Friday	Broccoli
Friday	Green Beans
Friday	Spinach

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

Proc SQL does a Cartesian product with a Many to Many join.

Data Step Merge

MANY TO MANY



Left

Key	Veggies
Monday	~~Broccoli~~
Monday	~~Green Beans~~
Monday	~~Spinach~~
Tuesday	Broccoli
Tuesday	Green Beans
Tuesday	Spinach
Thursday	Broccoli
Thursday	Green Beans
Thursday	Spinach
Friday	Broccoli
Friday	Green Beans
Friday	Spinach

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

- Data step joins the first record of dataset 1 to dataset 2, the second record of the first dataset to the second row of the second dataset and so on. No Cartesian product.

SQL

INNER JOIN – MANY TO MANY

```
title 'SQL_inner';
proc sql;
    select a.key, veggies, fruits
           from left2 as a, right2 as b
            where a.key=b.key
;
quit;
```

Left

Key	Veggies
Monday	Broccoli
Monday	Green Beans
Monday	Spinach
Tuesday	Broccoli
Tuesday	Green Beans
Tuesday	Spinach
Thursday	Broccoli
Thursday	Green Beans
Thursday	Spinach
Friday	Broccoli
Friday	Green Beans
Friday	Spinach

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

SQL_Inner

Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Oranges
Monday	Broccoli	Bananas
Monday	Green Beans	Apples
Monday	Green Beans	Oranges
Monday	Green Beans	Bananas
Monday	Spinach	Apples
Monday	Spinach	Oranges
Monday	Spinach	Bananas
Thursday	Broccoli	Apples
Thursday	Broccoli	Oranges
Thursday	Broccoli	Bananas
Thursday	Green Beans	Apples
Thursday	Green Beans	Oranges
Thursday	Green Beans	Bananas
Thursday	Spinach	Apples
Thursday	Spinach	Oranges
Thursday	Spinach	Bananas

Data Step

INNER JOIN – MANY TO MANY

```
data Data_Inner;
   merge left(in=left)
         right(in=right);
   by key;
   if left and right;
run;
Proc Print;
run;
```

Left

Key	Veggies
Monday	Broccoli
Monday	Green Beans
Monday	Spinach
Tuesday	Broccoli
Tuesday	Green Beans
Tuesday	Spinach
Thursday	Broccoli
Thursday	Green Beans
Thursday	Spinach
Friday	Broccoli
Friday	Green Beans
Friday	Spinach

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

Data_Inner

Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Green Beans	Bananas
Monday	Spinach	Oranges
Thursday	Broccoli	Apples
Thursday	Green Beans	Bananas
Thursday	Spinach	Oranges

SQL

LEFT JOIN – MANY TO MANY

```
proc sql;
    select a.key, veggies, fruits
           from left2 as a left join right2 as b
            on a.key=b.key
;
quit;
```

Left

Key	Veggies
Monday	Broccoli
Monday	Green Beans
Monday	Spinach
Tuesday	Broccoli
Tuesday	Green Beans
Tuesday	Spinach
Thursday	Broccoli
Thursday	Green Beans
Thursday	Spinach
Friday	Broccoli
Friday	Green Beans
Friday	Spinach

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

SQL_Left

Key	Veggies	Fruits
Monday	Green Beans	Bananas
Monday	Spinach	Bananas
Monday	Broccoli	Bananas
Monday	Green Beans	Oranges
Monday	Spinach	Oranges
Monday	Broccoli	Oranges
Monday	Green Beans	Apples
Monday	Spinach	Apples
Monday	Broccoli	Apples
Tuesday	Spinach	
Tuesday	Green Beans	
Tuesday	Broccoli	
Thursday	Broccoli	Apples
Thursday	Spinach	Apples
Thursday	Green Beans	Apples
Thursday	Broccoli	Oranges
Thursday	Spinach	Oranges
Thursday	Green Beans	Oranges
Thursday	Broccoli	Bananas
Thursday	Spinach	Bananas
Thursday	Green Beans	Bananas

Data Step

LEFT JOIN – MANY TO MANY

```
data Data_Left;
   merge left(in=left)
         right(in=right);
   by key;
   if left;
run;
Proc Print;
run;
```

Left

Key	Veggies
Monday	Broccoli
Monday	Green Beans
Monday	Spinach
Tuesday	Broccoli
Tuesday	Green Beans
Tuesday	Spinach
Thursday	Broccoli
Thursday	Green Beans
Thursday	Spinach
Friday	Broccoli
Friday	Green Beans
Friday	Spinach

Right

key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

Data_Left

Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Green Beans	Bananas
Monday	Spinach	Oranges
Tuesday	Broccoli	
Tuesday	Green Beans	
Tuesday	Spinach	
Thursday	Broccoli	Apples
Thursday	Green Beans	Bananas
Thursday	Spinach	Oranges
Friday	Broccoli	
Friday	Green Beans	
Friday	Spinach	

Data Step & SQL Output

RIGHT JOIN – MANY TO MANY

Data_Right

Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Green Beans	Bananas
Monday	Spinach	Oranges
Wednesday		Apples
Wednesday		Bananas
Wednesday		Oranges
Thursday	Broccoli	Apples
Thursday	Green Beans	Bananas
Thursday	Spinach	Oranges
Saturday		Apples
Saturday		Bananas
Saturday		Oranges

SQL_Right

Key	Veggies	Fruits
A	Green Beans	Bananas
A	Green Beans	Oranges
A	Green Beans	Apples
A	Spinach	Bananas
A	Spinach	Oranges
A	Spinach	Apples
A	Broccoli	Bananas
A	Broccoli	Oranges
A	Broccoli	Apples
C		Oranges
C		Bananas
C		Apples
D	Broccoli	Apples
D	Broccoli	Oranges

Data Step & SQL

FULL (OUTER) JOIN – MANY TO MANY

```
proc sql;
    select coalesce(a.key,b.key) as Key,
           a.veggies, b.fruits
          from left as a full join right as b
           on a.key=b.key
;
quit;

data Data_Outer;
   merge left(in=left)
         right(in=right);
   by key;
run;
Proc Print;
run;
```

Data Step & SQL Output

FULL (OUTER) JOIN – MANY TO MANY

Data_Outer

Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Green Beans	Bananas
Monday	Spinach	Oranges
Tuesday	Broccoli	
Tuesday	Green Beans	
Tuesday	Spinach	
Wednesday		Apples
Wednesday		Bananas
Wednesday		Oranges
Thursday	Broccoli	Apples
Thursday	Green Beans	Bananas
Thursday	Spinach	Oranges
Friday	Broccoli	
Friday	Green Beans	
Friday	Spinach	
Saturday		Apples
Saturday		Bananas
Saturday		Oranges

SQL_Outer

Key	Veggies	Fruits
A	Green Beans	Bananas
A	Green Beans	Oranges
A	Green Beans	Apples
A	Spinach	Bananas
A	Spinach	Oranges
A	Spinach	Apples
A	Broccoli	Bananas
A	Broccoli	Oranges
A	Broccoli	Apples
B	Spinach	
B	Green Beans	
B	Broccoli	
C		Oranges
C		Bananas
C		Apples
D	Broccoli	Apples
D	Broccoli	Oranges
D	Broccoli	Bananas
D	Spinach	Apples
D	Spinach	Oranges

Summary

SQL joins and DATA step merges may produce the same output for the following data:

- One to one
- One to many

SQL joins and DATA step merges produce dissimilar results when data represents a many to many structure.

Additional Comparisons

Conditional Processing

- IF THEN statement in the DATA step
  - Very flexible
- CASE expression in SQL

SQL

CASE EXPRESSION

```
proc sql;
  select name, case
    when continent = 'North America' then 'US'
    when continent = 'Oceania' then 'Pacific Islands'
    else 'None'
    end as region
  from states;
```

Data Step

IF THEN STATEMENT

```
data new;
    set states;
    if continent = 'North America'
        then region ='US';
    else if continent = 'Oceania'
        then region = 'Pacific Islands';
    else region='None';
run;
```

Indexes

Indexes can be created by

- SQL
- DATA step (at data set creation)

Indexes may also be administered through SQL.

Data Step

INDEXES

```
data health.test(index=(memberID));
   set health.claims_sample;
run;
```

Indexes are created at the time of data set creation. PROC DATASETS can be used to change or maintain the indexes.

SQL

INDEXES

```
proc sql;
   drop index providerId from health.test;
   create unique index ProviderID on
  health.provider(providerID);
```

PROC SQL can be used to create and administer indexes.

Subsetting

- Use the WHERE clause in PROC SQL to select only the rows of data that meet a certain condition.

- Use the WHERE statement or WHERE= option in the DATA step to select only the rows of data that meet a certain condition

Subsetting

```
proc sql;
  create table sql_subset as
  select * from sashelp.cars
  where make='Acura' and type='Sedan';
quit;
```

```
data data_subset;
   set sashelp.cars;
   where make='Acura' and type='Sedan';
run;
```

Sorting, summarizing and creating new variables

- PROC SQL can sort, summarize, and create new variables in the same step
- The DATA step requires separate steps to accomplish sorting, summarizing, and creating new variables

Sorting, summarizing, creating new variables

```
proc sql;
   title 'SQL - Total Reimbursement';
   title2 'for Each Trial Phase in Test 1';
   select phase, sum(reimbursement) as tot_reimbursement
      label='Total Reimbursement' format=dollar15.
         from o.datafile
         where test_level='Test 1'
         group by Phase
         order by phase;
quit;
```

Sorting, summarizing, creating new variables

```
proc summary data=o.datafile;
   where test_level='Test 1';
   class phase;
   var reimbursement;
   output out=tot_reimburse sum=tot_reimbursement;
run;

proc sort data=tot_reimburse;
   by phase;
run;

proc print data=tot_reimburse noobs label;
   var phase tot_reimbursement;
   format tot_reimbursement dollar15.;
   label tot_reimbursement='Total Reimbursement';
   title 'PROCS - Total Reimbursement';
   title2 'for Each Trial Phase in Test 1';
   where _type_=1;
run;
```

Data Step

CREATING EXECUTION TIME MACRO VARIABLES-

```
data _null_;
   call symputx(' items ', ' text to assign’);
   call symputx(' x ', 123.456);
run;
```

Both the DATA step and SQL can create macro variables at execution time.

The DATA step might be considered more flexible.

SQL

CREATING EXECUTION TIME MACRO VARIABLES

```
proc sql noprint;
   select country, barrels
      into :country1, :barrels1
      from sql.oilrsrvs;
```

Data Step

LOOPING

The DATA step allows the following capabilities for iterative processing:

- DO WHILE
- DO UNTIL
- Iterative DO

Data Step

LOOPING

DO WHILE

The condition is evaluated at the top of the loop. Looping is terminated based on a condition.

```
data test;
   J = 1;
   do while(J lt 3);
      put J=;
      J+ +1;
   end;
   put 'do J: ' J=;
run;
```

Partial SAS Log:

```
J=1
J=2
do J: J=3
```

Data Step

LOOPING

DO UNTIL

The UNTIL test is evaluated at the bottom of the loop. Looping is terminated based on a condition.

```
data test;
   K = 1;
   do until(K ge 3);
      put K=;
      K+ +1;
   end;
   put 'do K: ' K=;
run;
```

Partial SAS Log:

```
K=1
K=2
do K: K=3
```

Data Step

LOOPING

Iterative DO

The iterative DO processes statements a specified number of times.

```
data test;
   L = 0;
   do L = 1 to 2;
      put L=;
   end;
   put 'do L: ' L=;
run;
```

Partial SAS Log:

```
L=1
L=2
do L: L=3
```

PROC SQL has the capability to produce basic line-oriented reports, while the DATA step provides maximum flexibility for creating highly customized reports.

Data Step

PRESENTING YOUR DATA

The DATA step code below produces the block-oriented report

```
data _null_;
  set heartsorted;
  file print notitles  header=head;
  put @10 'Patient ID: ' pat_id
      @30 'Gender:       ' sex /
      @30 'Height:       ' height/
      @30 'Weight:       ' weight/
      @30 'Cholesterol:  ' cholesterol //

  return;
  head:
     put @22 'Patient Information' //;
  return;
run;
```

```
                    Patient Information


Patient ID: 124     Gender:       Female
                    Height:       62.25
                    Weight:       132
                    Cholesterol:  250

Patient ID: 125     Gender:       Female
                    Height:       65.75
                    Weight:       158
                    Cholesterol:  242

Patient ID: 126     Gender:       Male
                    Height:       66
                    Weight:       156
                    Cholesterol:  281
```

DBMS Pass through

ORACLE EXAMPLE (FROM EG)

```
PROC SQL;
   CONNECT TO ORACLE as con1
    (path=exadat user="sascxg" pw="{sas002}F3E0253E4A824F6817A03031");

   CREATE TABLE WORK.QUERY_FOR_CLAIMS_SAMPLE(label="Query") AS
   SELECT *
      FROM CONNECTION TO con1 (
      SELECT "t1"."MEMBERID",
          "t1"."SVC_YEAR",
          "t1"."CHGAMT",
          "t1"."LNDISCAMT",
          "t1"."EERESP",
          "t1"."PAIDAMT"
         FROM "CLAIMS_SAMPLE" "t1");

   DISCONNECT FROM con1;
QUIT;
```

Thoughts and Conclusions

How do the techniques compare?

OBSERVATIONS, BASED ON OUR EXAMPLES

- When joining tables, carefully review your output results to ensure that the code, whether PROC SQL or DATA Step, produces the results you want and expect
- In cases of one-to-one and one-to-many joins, PROC SQL and the DATA Step can produce the same results
- In cases of many-to-many joins, PROC SQL and the DATA Step produce different results
- Both PROC SQL and the DATA Step perform subsetting of your data very efficiently
- PROC SQL incorporates many tasks into the same step, but it is not always more efficient resource-wise. Benchmark your results, if the metrics matter.

How do the techniques compare?

BENEFITS AND ADVANTAGES*

SQL:

- Provides the combined functionality of the DATA step and several base SAS procedures
- PROC SQL code may execute faster for smaller tables
- PROC SQL code is more portable for non-SAS programmers and non-SAS applications
- Processing does not require explicit code to presort tables
- Processing does not require common variable names to join on, although same type and length are required
- By default, a PROC SQL SELECT statement prints the resultant query; use the NOPRINT option to suppress this feature
- Efficiencies within specific RDBMS are available with Pass-thru code (connect to) for the performance of joins
- Use of aliases for shorthand code may make some coding tasks easier

*From **DATA Step vs. PROC SQL: What's a neophyte to do?** Proceedings of 29th SAS User Group International Conference, by Craig Dickstein, Tamarack Professional Services, Jackman, ME and Ray Pass, Ray Pass Consulting, Hartsdale, NY

How do the techniques compare?

BENEFITS AND ADVANTAGES*

NON-SQL BASE SAS:

- DATA step set operators can handle more data sets at a time than PROC SQL outer joins
- Non-SQL techniques can open files for read and write at the same time
- Customized DATA step report writing techniques (DATA _NULL_) are more versatile than using PROC SQL SELECT clauses
- The straightforward access to RDBMS tables as if they were SAS data sets negates the need to learn SQL constructs
- Input of non-RDBMS external sources is easier

*From **DATA Step vs. PROC SQL: What's a neophyte to do?** Proceedings of 29th SAS User Group International Conference, by Craig Dickstein, Tamarack Professional Services, Jackman, ME and Ray Pass, Ray Pass Consulting, Hartsdale, NY

How do the techniques compare?

WHICH TECHNIQUE TO USE?

- Which technique more efficiently handles the task at hand?
- Which technique allows you to be more collaborative with your peers and coworkers?
- Which technique is easier to maintain?
- Which technique are you more familiar with?

Additonal Resources

support.sas.com

Resources

- Base SAS documentation

http://support.sas.com/documentation/onlinedoc/base/index.html

- SAS Training

https://support.sas.com/edu

- RSS & Blogs

http://support.sas.com/community/rss/index.html

http://blogs.sas.com

- Discussion Forums

http://communities.sas.com

support.sas.com

Resources

- SAS® SQL Procedure User's Guide

http://documentation.sas.com/?docsetId=sqlproc&docsetTarget=p020urejdmvi7vn1t9avbvazqapu.htm&docsetVersion=9.4&locale=en

- Papers & SAS Notes

http://support.sas.com/resources/papers/sgf09/336-2009.pdf

http://support.sas.com/kb/20/783.html

- SAS Training

https://support.sas.com/edu/schedules.html?id=336&ctry=US

- Yes, Proc SQL is Great, But I Love Data Step Programming, What's a SAS User To Do? Mira J. Shapiro, MJS Consultants, Bethesda, MD
- DATA Step versus PROC SQL Programming Techniques Kirk Paul Lafler, Software Intelligence Corporation
- DATA Step vs. PROC SQL: What's a neophyte to do? Craig Dickstein, Tamarack Professional Services, Jackman, ME Ray Pass, Ray Pass Consulting, Hartsdale, NY

support.sas.com

Papers & Blogs

To search past papers click here

Questions?

- Thank you for your time and attention!