

## PROC SQL VS. DATA STEP PROCESSING



Mary-Elizabeth ("M-E") Eddlestone  
Principal Analytics Engineer, SAS Customer Loyalty  
Customer Engagement & Support

### AGENDA

- Comparison of DATA Step and PROC SQL capabilities
- Joining SAS data using the DATA Step and PROC SQL

## COMPARISON OF DATA STEP AND PROC SQL CAPABILITIES



### DATA STEP VS. PROC SQL

#### CAPABILITIES COMPARISON

- **DATA Step** is typically sequential processing
  - Rows of data are processed in the same order they started in
    - Row 1, row 2, row 3, etc.
  - Output for most joins will have a consistent order
  - Can do random access using POINT= or KEY= options on SET statement
- **PROC SQL** uses an optimizer – dissimilar results
  - SQL has no concept of row order – does not process sequentially
  - Rows can be, and often are, returned in a random order unless an ORDER BY clause is used
  - GROUP BY does not include ORDER BY

SQL Optimizer explained: <http://support.sas.com/techsup/technote/ts553.html>

## DATA STEP VS. PROC SQL

### DATA Step

- No issues as most processing for single tables as well as joins is sequential

### PROC SQL

- Lags in performance without help
- Always sort large tables before joining in SQL
- NOT indexed!
- Many times a series of 2-table joins will out-perform a multi-table (3+) join in PROC SQL



## DATA STEP WHAT CAN I DO WITH IT?

- 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


- ANSI standard SQL is the base
- SAS-isms added (functions, formats, labels, etc.)

```
PROC SQL;  
  SELECT <* | col1<, col2<... , coln>>  
  FROM <input-table-name> << as > alias-name>  
  ;  
QUIT;
```


## PROC SQL WHAT CAN I DO WITH IT?

- 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
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	



## PROCESSING COMPARISONS



### CONDITIONAL PROCESSING

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

**PROC SQL** | **CONDITIONAL PROCESSING: 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** | **CONDITIONAL PROCESSING: 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 PROC SQL

```
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.



## CREATING MACRO VARIABLES PROC SQL

```
proc sql noprint;
  select distinct style
  into :style01 - :style04
  from work.houses;

%put There were &sqllobs distinct values.;
```

PARTIAL LOG:  
 %put There were &sqllobs distinct values.;  
 There were 4 distinct values.

Obs	Style	SqFeet
1	CONDO	900
2	CONDO	1000
3	RANCH	1200
4	RANCH	1400
5	SPLIT	1600
6	SPLIT	1800
7	TWOSTORY	2100
8	TWOSTORY	3000
9	TWOSTORY	1940
10	TWOSTORY	1860





## CREATING MACRO VARIABLES

### PROC SQL

```
proc sql noprint;
  select distinct style
  into :style01 -
  from work.houses;

%put There were &sqllobs distinct values.;
```

PARTIAL LOG:

```
%put There were &sqllobs distinct values.;
There were 4 distinct values.
```

Obs	Style	SqFeet
1	CONDO	900
2	CONDO	1000
3	RANCH	1200
4	RANCH	1400
5	SPLIT	1600
6	SPLIT	1800
7	TWOSTORY	2100
8	TWOSTORY	3000
9	TWOSTORY	1940
10	TWOSTORY	1860

## DATA STEP

### CREATING 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.

**DATA STEP RETRIEVING SYSTEM INFORMATION**

System information can be retrieved by using DOPEN, DINFO and other related functions within the DATA step.

**DATA STEP RETRIEVING SYSTEM INFORMATION**

**DOPEN** opens a directory, and returns a directory identifier value.

```
data diropts;  
  length foption $ 12 charval $ 40;  
  keep foption charval;  
  rc=filename("mydir", "physical-name");  
  did = dopen("mydir");  
  numopts=doptnum(did);  
  do i=1 to numopts;  
    foption=doptname(did,i);  
    charval=dinfo(did,foption);  
    output;  
  end;  
run;
```



## DATA STEP RETRIEVING SYSTEM INFORMATION

**DOPTNUM** returns the number of informational items that are available for a directory.

```

data diropts;
  length foption $ 12 charval $ 40;
  keep foption charval;
  rc = filename("mydir", "physical-name");
  did = dopen("mydir");
  numopts = doptnum(did);
  do i = 1 to numopts;
    foption = doptname(did,i);
    charval = dinfo(did,foption);
    output;
  end;
run;

```

## DATA STEP RETRIEVING SYSTEM INFORMATION

**DOPTNAME** returns directory attribute information.

```

data diropts;
  length foption $ 12 charval $ 40;
  keep foption charval;
  rc = filename("mydir", "physical-name");
  did = dopen("mydir");
  numopts=doptnum(did);
  do i = 1 to numopts;
    foption = doptname(did,i);
    charval = dinfo(did,foption);
    output;
  end;
run;

```

## DATA STEP RETRIEVING SYSTEM INFORMATION

**DINFO** returns information about a directory.

```

data diropts;
  length foption $ 12 charval $ 40;
  keep foption charval;
  rc = filename("mydir", "physical-name");
  did = dopen("mydir");
  numopts=doptnum(did);
  do i = 1 to numopts;
    foption = doptname(did,i);
    charval = dinfo(did,foption);

    output;
  end;
run;

```

## DATA STEP RETRIEVING SYSTEM INFORMATION

**DCLOSE** closes the open resource.

```

data diropts;
  length foption $ 12 charval $ 40;
  keep foption charval;
  rc = filename("mydir", "physical-name");
  did = dopen("mydir");
  numopts=doptnum(did);
  do i = 1 to numopts;
    foption = doptname(did,i);
    charval = dinfo(did,foption);
    output;
  end;
  rc = dclose(did);
run;

```

## DATA STEP RETRIEVING SYSTEM INFORMATION

There are similar functions for dealing with files and members. For more information, see **SAS(R) 9.4 Functions and CALL Routines: Reference, Third Edition**

- FEXIST
- FOPEN
- FCLOSE
- MOPEN
- Etc.

<http://support.sas.com/documentation/cdl/en/lefuctionsref/67398/HTML/default/viewer.htm#titlepage.htm>



## LOOPING IN THE DATA STEP

The DATA step allows the following capabilities for iterative processing:

- DO WHILE
- DO UNTIL
- Iterative DO



## LOOPING IN THE DATA STEP

### DO WHILE

#### DO WHILE

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

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

Partial SAS Log:

```
J=0
J=1
J=2
do J: J=3
```



## LOOPING IN THE DATA STEP

### DO UNTIL

#### DO UNTIL

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

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

Partial SAS Log:

```
K=0
K=1
K=2
do K: K=3
```



## LOOPING IN THE DATA STEP

### ITERATIVE DO

#### 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
```

## LOOPING IN THE DATA STEP

### Do Which? Loop, Until or While? A Review Of Data Step And Macro Algorithms

Ronald J. Fehd, Centers for Disease Control, and Prevention, Atlanta, GA, USA

<http://www2.sas.com/proceedings/forum2007/067-2007.pdf>

## PROC SQL PRESENTING YOUR DATA

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

### Example SQL Output

PatientID	Sex	Height	Weight	Cholesterol
124	Female	62	124	347
125	Female	64.25	150	238
126	Female	65.75	123	199

## DATA STEP PRESENTING YOUR DATA

The DATA step provides maximum flexibility for creating highly customized reports. Consider the following sample report:

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



## DATA STEP PRESENTING YOUR DATA

The DATA step code below produced the block-oriented report seen previously.

```
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;
```

## 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

*Left*

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

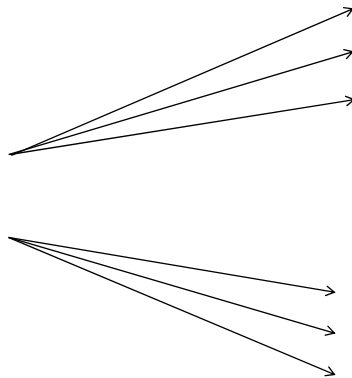
*Right*

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

## ONE TO MANY

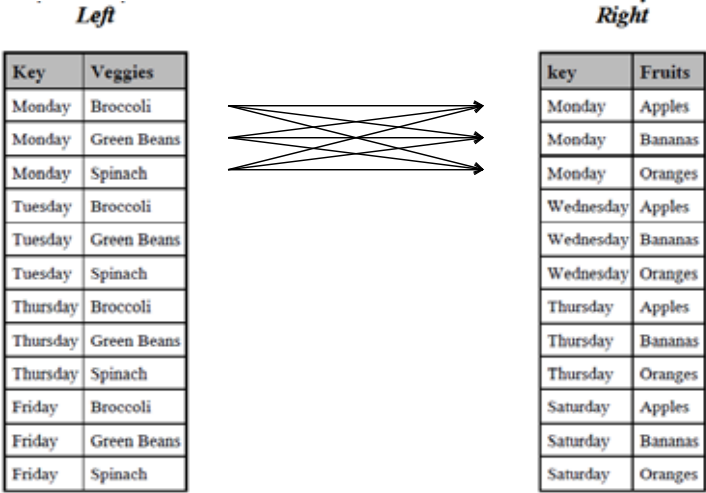
*Left*

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

*Right*

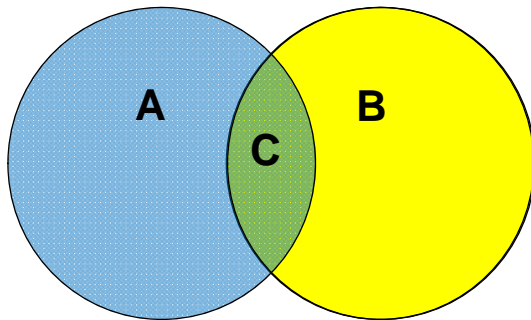
key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges

# MANY TO MANY



# TYPES OF JOINS

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



## SAMPLE DATA

*Left*

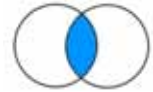
Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

*Right*

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

DEFAULT  
INNER JOIN

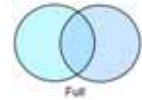
## ONE TO ONE

*One to One Joins*  
*SQL\_Join*

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

Key	Veggies	Fruits
Monday	Broccoli	Apples
Thursday	Broccoli	Apples

## DEFAULT JOIN ONE TO ONE

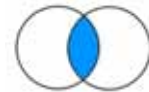


*One to One Joins*  
*Data\_Merge*

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

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

## INNER JOIN ONE TO ONE

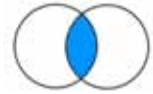


*One to One Joins*  
*Data\_Inner*

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

Key	Veggies	Fruits
Monday	Broccoli	Apples
Thursday	Broccoli	Apples

## INNER JOIN ONE TO ONE



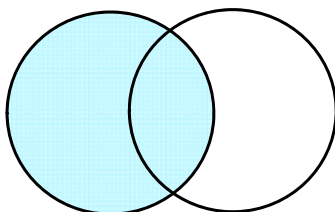
```
proc sql;
  select a.*, b.fruits
  from Left a inner join Right b
    on a.key = b.key
  ;
quit;
```

### One to One Joins SQL\_Inner

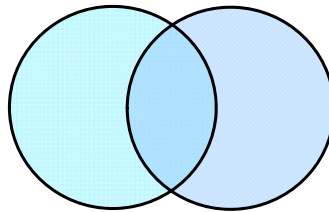
Key	Veggies	Fruits
Monday	Broccoli	Apples
Thursday	Broccoli	Apples

## OUTER JOINS

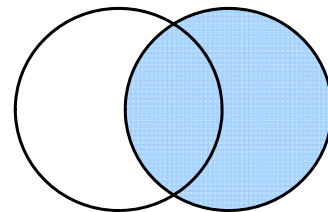
- Return all matching rows, plus nonmatching rows from one or both tables
- Can be performed on only two tables or views at a time.



Left



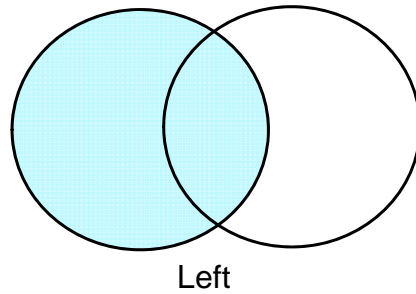
Full



Right

## OUTER JOINS

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



## SAMPLE DATA

### *Left*

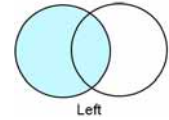
Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

### *Right*

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples



## LEFT JOIN ONE TO ONE

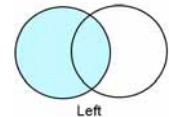


```
proc sql;
  select a.*, b.fruits
  from Left a
       Left join
       Right b
  on a.key = b.key
;
quit;
```

*One to One Joins*  
*SQL\_Left*

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

## LEFT JOIN ONE TO ONE



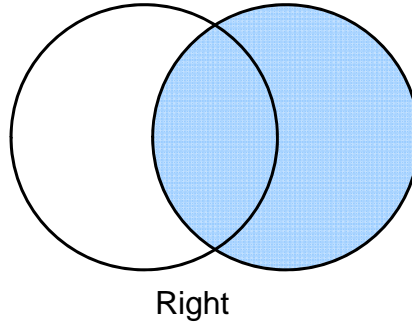
```
data Data_Left;
  merge Left(in=left)
        Right(in=right);
  by key;
  if left;
run;
```

*One to One Joins*  
*Data\_Left*

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

## OUTER JOINS

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



## SAMPLE DATA

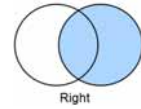
### *Left*

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

### *Right*

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

## RIGHT JOIN ONE TO ONE

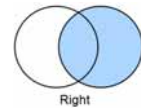


```
proc sql;
  select b.key as Key
         , a.Veggies
         , b.Fruits
  from Left a
       right join
       Right b
  on a.key = b.key
;
quit;
```

*SQL\_Right*

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

## RIGHT JOIN ONE TO ONE



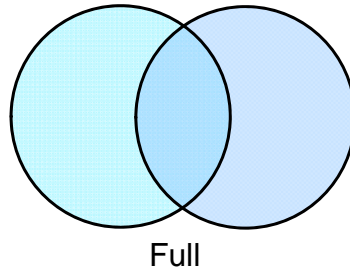
```
data Data_Right;
  merge Left(in=left)
        Right(in=right);
  by key;
  if right;
run;
```

*Data\_Right*

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

## 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.



## SAMPLE DATA

### *Left*

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

### *Right*

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

## 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

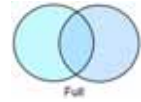
Key	veggies	fruits
Friday	Broccoli	
Monday	Broccoli	Apples
		Apples
Thursday	Broccoli	Apples
Tuesday	Broccoli	
		Apples

## 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*)

## FULL (OUTER) JOIN ONE TO ONE

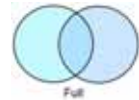


```
proc sql;
/* create table SQL_Outer as*/
select coalesce(a.key,b.key) as
Key
      , a.Veggies
      , b.Fruits
from Left a
Full join
Right b
on a.key = b.key
;
quit;
```

### SQL\_Outer

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

## FULL (OUTER) JOIN ONE TO ONE



```
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

## SAMPLE DATA

## 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

\* Code is the same, only the data have changed.



## DEFAULT JOIN

## ONE TO MANY

*Data\_Merge*

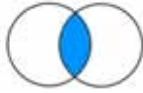
Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Tuesday	Broccoli	
Wednesday		Apples
Wednesday		Bananas
Wednesday		Oranges
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges
Friday	Broccoli	
Saturday		Apples
Saturday		Bananas
Saturday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

*SQL\_Join*

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



## INNER JOIN



## ONE TO MANY

*Data\_Inner*

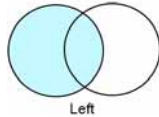
Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges

*SQL\_Inner*

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



## LEFT JOIN



Left

## ONE TO MANY

*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*

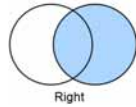
Key	Veggies	Fruits
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Monday	Broccoli	Apples
Tuesday	Broccoli	
Thursday	Broccoli	Apples
Thursday	Broccoli	Oranges
Thursday	Broccoli	Bananas
Friday	Broccoli	

Note the row order!





## RIGHT JOIN

*Data\_Right*

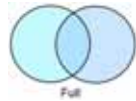
Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Wednesday		Apples
Wednesday		Bananas
Wednesday		Oranges
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges
Saturday		Apples
Saturday		Bananas
Saturday		Oranges

*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



ONE TO MANY

FULL (OUTER)  
JOIN*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



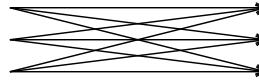
ONE TO MANY

MANY TO  
MANY

PROC SQL

*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



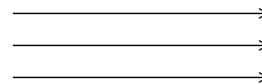
MANY TO  
MANY

DATA STEP MERGE

CAUTION CAUTION CAUTION

*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*

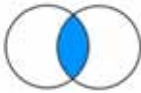
key	Fruits
Monday	Apples
Monday	Bananas
Monday	Oranges
Wednesday	Apples
Wednesday	Bananas
Wednesday	Oranges
Thursday	Apples
Thursday	Bananas
Thursday	Oranges
Saturday	Apples
Saturday	Bananas
Saturday	Oranges



NOTE: MERGE statement has more than one data set with repeats of BY values.


# 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

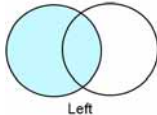
SQL_Inner		
Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Green Beans	Apples
Monday	Spinach	Apples
Monday	Broccoli	Bananas
Monday	Green Beans	Bananas
Monday	Spinach	Bananas
Monday	Broccoli	Oranges
Monday	Green Beans	Oranges
Monday	Spinach	Oranges
Thursday	Broccoli	Apples
Thursday	Green Beans	Apples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





# LEFT JOIN

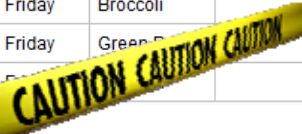
## MANY TO MANY




Left

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	

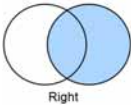
SQL_Left		
Key	Veggies	Fruits
Monday	Green Beans	Bananas
Monday	Spinach	Bananas
Monday	Broccoli	Bananas
Monday	Green Beans	Oranges
Monday	Spinach	Oranges
Monday	Broccoli	Oranges
Monday	Green Beans	Apples
Monday	Spinach	Apples
Monday	Broccoli	Apples
Tuesday	Spinach	
Tuesday	Green Beans	
Tuesday	Broccoli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






## RIGHT JOIN

## MANY TO MANY



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		

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



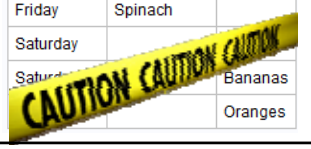
## FULL (OUTER) JOIN

## MANY TO MANY



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		
Saturday		Bananas
Saturday		Oranges

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



## JOINING DATA SUMMARY

- SQL joins and DATA step merges may produce the similar output, *other than row order*, for the following joins:
  - One to one
  - One to many
- SQL joins process many to many data correctly.
- The DATA step MERGE statement will not correctly perform a many to many join. This requires additional coding within the DATA step. A paper that shows an example: <http://www.lexjansen.com/nesug/nesug08/ff/ff03.pdf>



QUESTIONS?



SUPPORT.SAS.COM  
RESOURCES

## DOCUMENTATION

- Base SAS Documentation
  - <http://support.sas.com/documentation/onlinedoc/base/index.html>
- SAS 9.4 Companion for:
  - Unix Environments  
<http://support.sas.com/documentation/cdl/en/hostunix/67132/PDF/default/hostunix.pdf>
  - Windows  
<http://support.sas.com/documentation/cdl/en/hostwin/67279/PDF/default/hostwin.pdf>
  - z/OS  
<http://support.sas.com/documentation/cdl/en/hosto390/67131/PDF/default/hosto390.pdf>



SUPPORT.SAS.COM  
RESOURCES

## DOCUMENTATION

- SAS<sup>®</sup> 9.3 SQL Procedure User's Guide  
<http://support.sas.com/documentation/cdl/en/sqlproc/63043/PDF/default/sqlproc.pdf>
- SAS<sup>®</sup> 9.4 SQL Procedure User's Guide  
<http://support.sas.com/documentation/cdl/en/sqlproc/65065/PDF/default/sqlproc.pdf>
- Base SAS Procedure Documentation (including PROC SQL)  
<http://support.sas.com/documentation/cdl/en/proc/66663/PDF/default/proc.pdf>



SUPPORT.SAS.COM  
RESOURCES

## PAPERS AND SAS NOTES

- Usage Note 20783: Helpful resources about the SQL procedure
  - <http://support.sas.com/kb/20/783.html>
- Inside PROC SQL's Query Optimizer
  - <https://support.sas.com/techsup/technote/ts320.html#>
- Dear Miss SASAnswers: A Guide to Efficient PROC SQL Coding
  - <http://support.sas.com/resources/papers/sgf09/336-2009.pdf>



SUPPORT.SAS.COM  
RESOURCES

## • RSS & Blogs

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

<http://blogs.sas.com>

## • Discussion Forums

<http://communities.sas.com/index.jspx>





The one place for all your SAS Training needs.  
[support.sas.com/training](https://support.sas.com/training)

It's where you'll find the latest information on:

- New training courses and services
- Special offers and discounts
- The latest course schedules
- New training locations
- Events and conferences
- SAS certification news
- And, much more.

Everything you need – in one place.  
Visit and bookmark it today.

## SAS EDUCATION | TRAINING OPTIONS

- SAS Programming 1: Essentials
  - <https://support.sas.com/edu/schedules.html?id=277&ctry=US>
- SAS SQL 1: Essentials
  - <https://support.sas.com/edu/schedules.html?id=336&ctry=US>



## SAS EDUCATION | TRAINING OPTIONS

SAS Programming 1 is now available for **FREE** as an e-learning module!

- SAS Programming 1: Essentials
  - <https://support.sas.com/edu/schedules.html?id=277&ctry=US>
- SAS SQL 1: Essentials
  - <https://support.sas.com/edu/schedules.html?id=336&ctry=US>



Log In | Worldwide Sites | Contact Us | Search

Products & Solutions | Industries | Support & Training | Customer Stories | Partners | Community | About SAS

Home > Products & Solutions > SAS University Edition

# SAS® University Edition

Pay nothing. Gain everything.

[Download free software](#)

Free SAS® software. An interactive, online community. Superior training and documentation. And the analytical skills you need to secure your future.


[http://www.sas.com/en\\_us/software/university-edition.html](http://www.sas.com/en_us/software/university-edition.html)

SAS | THE POWER TO KNOW

**sas** Products & Solutions Industries Support & Training Customer Stories Partners Community About SAS

## Features


- **An intuitive interface** that lets you interact with the software from your PC, Mac or Linux workstation. Learn more about [SAS Studio](#).
- **A powerful programming language** that's easy to learn, easy to use. Learn more about [Base SAS](#).
- **Comprehensive, reliable tools** that include state-of-the-art statistical methods. Learn more about [SAS/STAT](#).
- **A robust, yet flexible matrix programming language** for more in-depth, specialized analysis and exploration. Learn more about [SAS/IML](#).
- **Out-of-the-box access to PC file formats** for a simplified approach to accessing data. Learn more about [SAS/ACCESS](#).



Download the SAS® University Edition fact sheet [▶](#)

[http://www.sas.com/en\\_us/software/university-edition.html](http://www.sas.com/en_us/software/university-edition.html)

**sas** THE POWER TO KNOW



Presentation prepared by:

- Ron Coleman
- Cynthia Johnson
- Jeff Simpson
- M-E Eddlestone

[sas.com](http://sas.com)