PROC SQL vs. DATA Step Processing

Mary-Elizabeth ("M-E") Eddlestone
Analytical Technical Advisor
Global Customer Success Organization
https://www.linkedin.com/in/meeddlestone
M-E.Eddlestone@sas.com

AGENDA

01 Comparison of DATA Step and PROC SQL capabilities

02 Joining SAS data using the DATA Step and PROC SQL

03 Q&A

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

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
- Sequential

PROC SQL

- Lags in performance without help
- Always sort large tables before joining in SQL
- 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

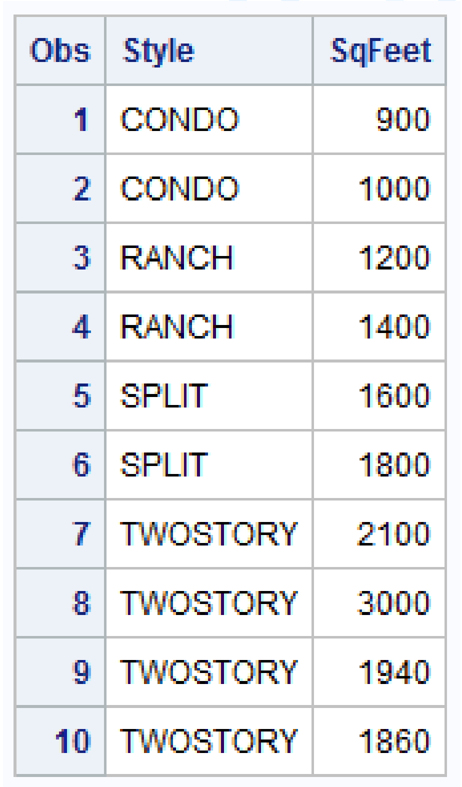
PROC SQL

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

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

PARTIAL LOG:

```
%put There were &sqlobs 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

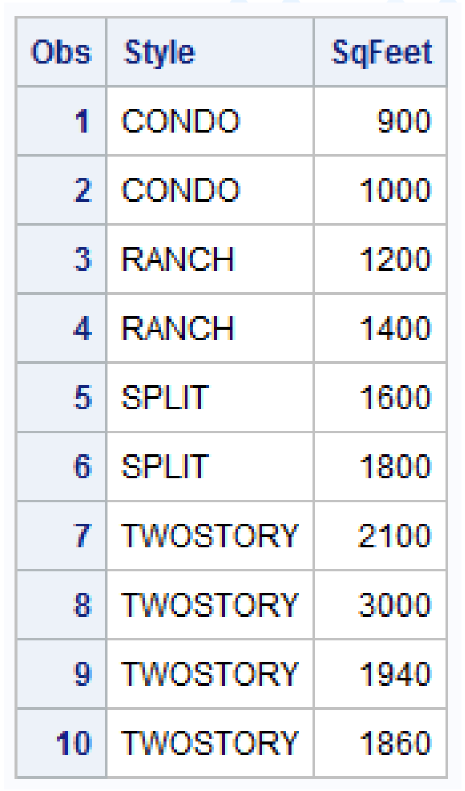
Creating Macro Variables

PROC SQL

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

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

```
PARTIAL LOG:
%put There were &sqlobs 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/lefunctionsref/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

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

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

Looping in the DATA Step

DO UNTIL

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

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

Looping in the DATA Step

Iterative DO

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

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

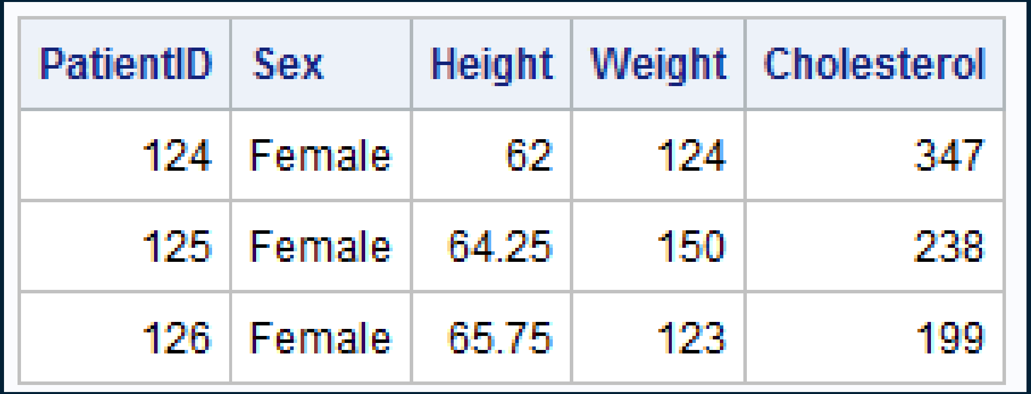
Looping in the Data Step

- 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.

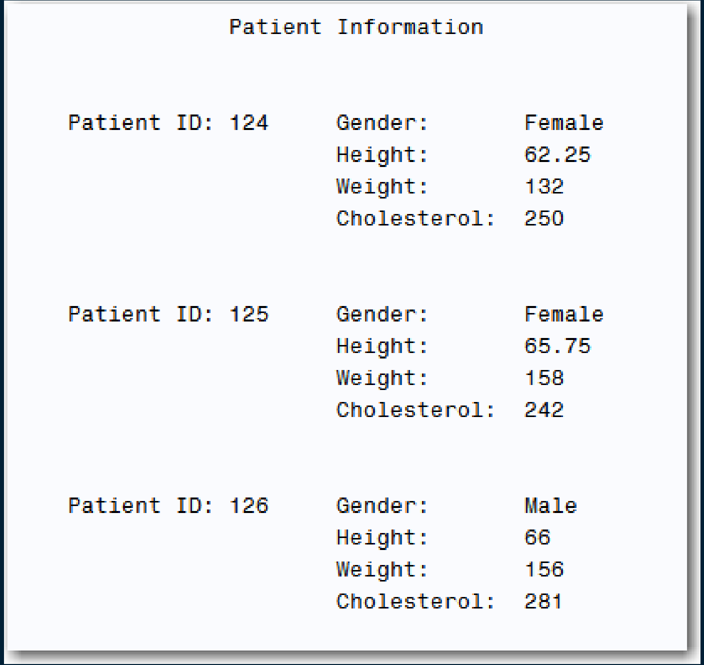


PatientID	Sex	Height	Weight	Cholesterol
124	Female	62	124	347
125	Female	64.25	150	238
126	Female	65.75	123	199

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

```
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

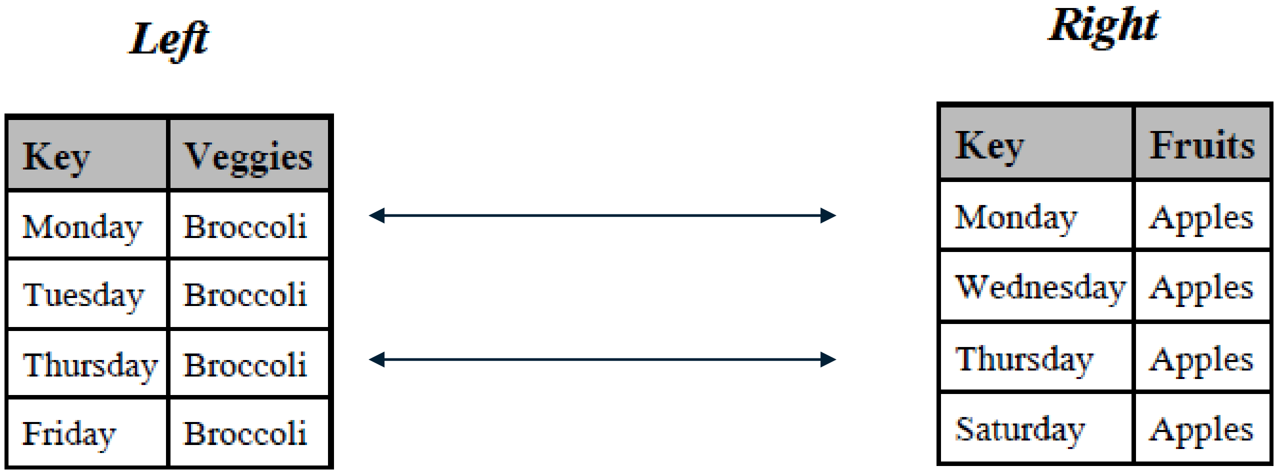
Types of Joins

- 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

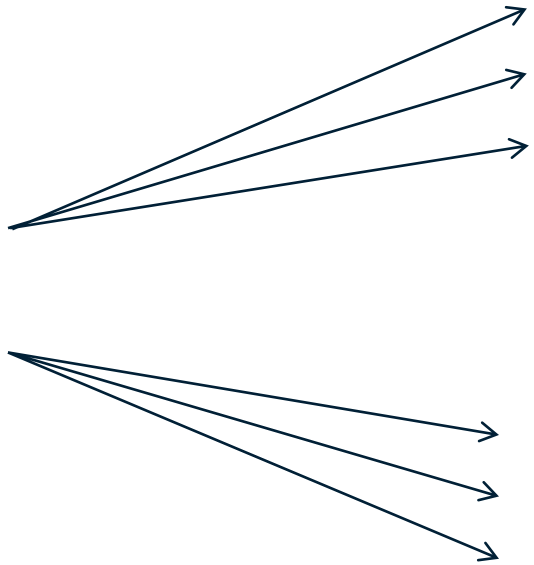
Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

One to Many

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli



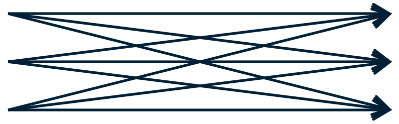
Right

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

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



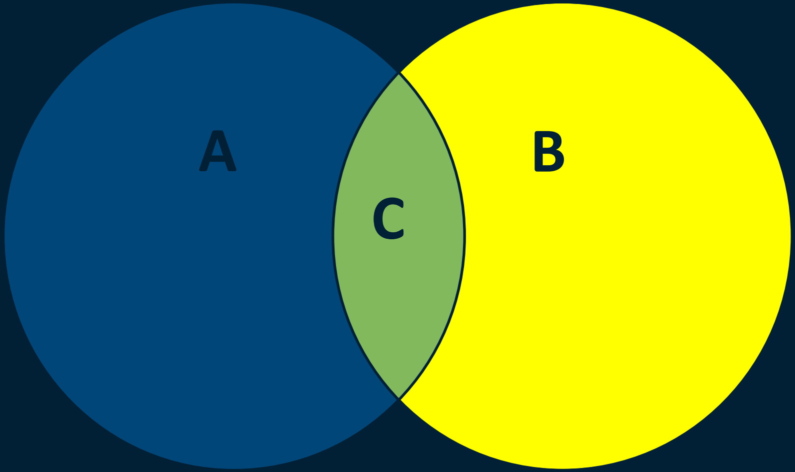
Right

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

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

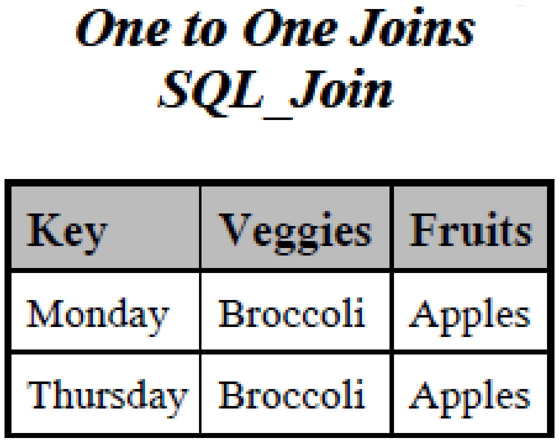
Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

Default Inner Join

One to One

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



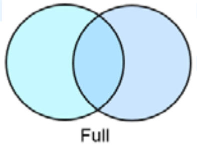
One to One Joins
SQL_Join

Key	Veggies	Fruits
Monday	Broccoli	Apples
Thursday	Broccoli	Apples

Default Join

One to One

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

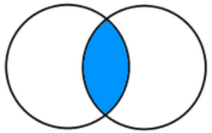


One to One Joins
Data_Merge

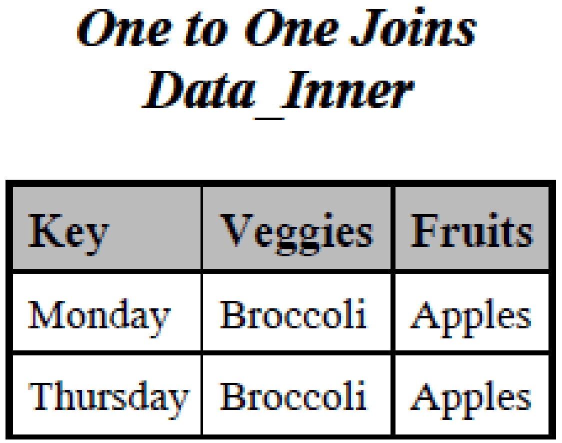
Key	Veggies	Fruits
Monday	Broccoli	Apples
Tuesday	Broccoli	
Wednesday		Apples
Thursday	Broccoli	Apples
Friday	Broccoli	
Saturday		Apples

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

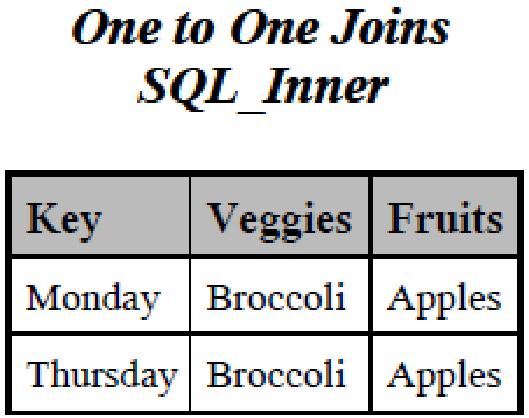
Key	Veggies	Fruits
Monday	Broccoli	Apples
Thursday	Broccoli	Apples

Inner Join

One to One

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

       on a.key = b.key
   ;
quit;
```

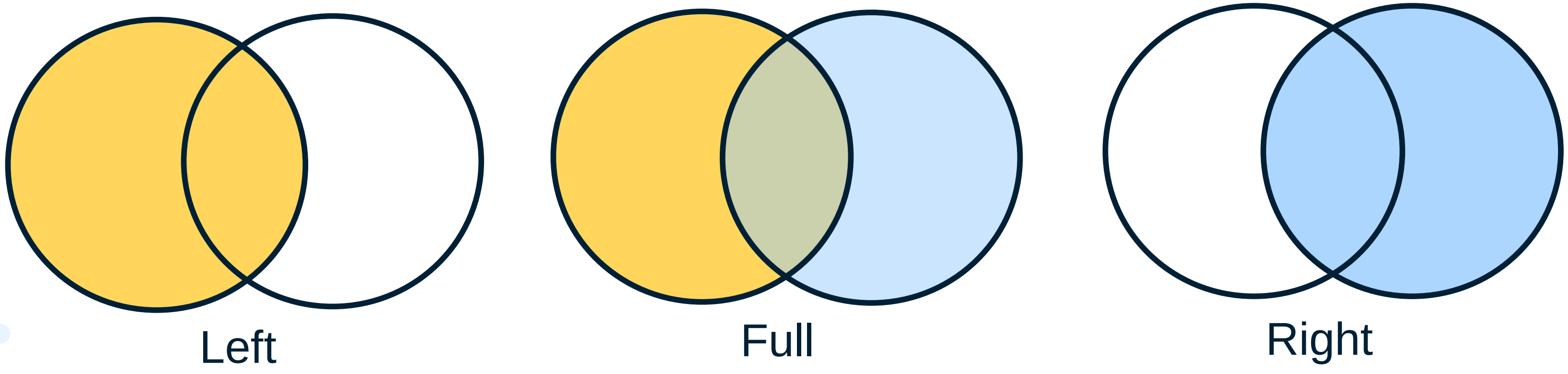


One to One Joins
SQL_Inner

Key	Veggies	Fruits
Monday	Broccoli	Apples
Thursday	Broccoli	Apples

Outer Joins

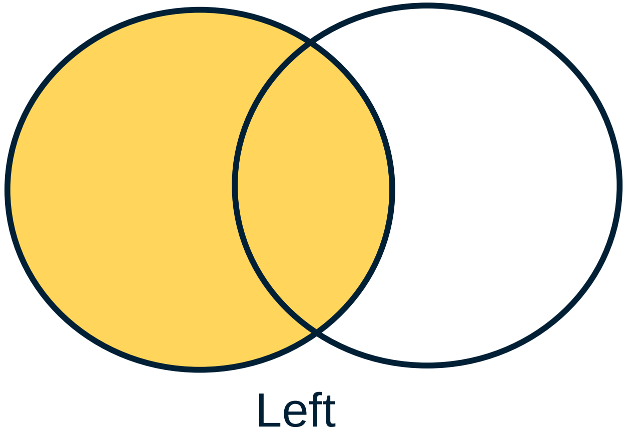
- Return all matching rows, plus nonmatching rows from one or both tables
- Can be performed on only two tables or views at a time.



Outer Joins

Left Join

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



Sample Data

Left

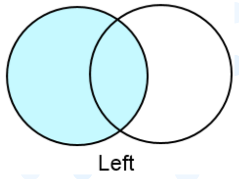
Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

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;
```

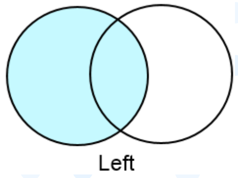
One to One Joins
SQL_Left

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;
```



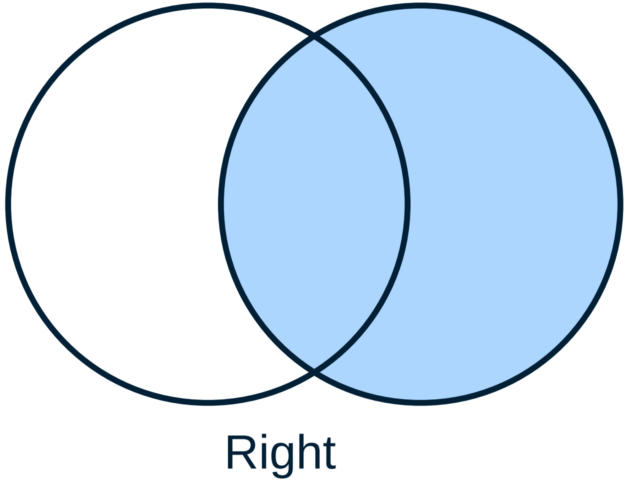
One to One Joins
Data_Left

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

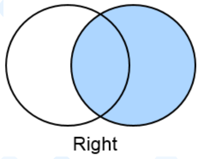
Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

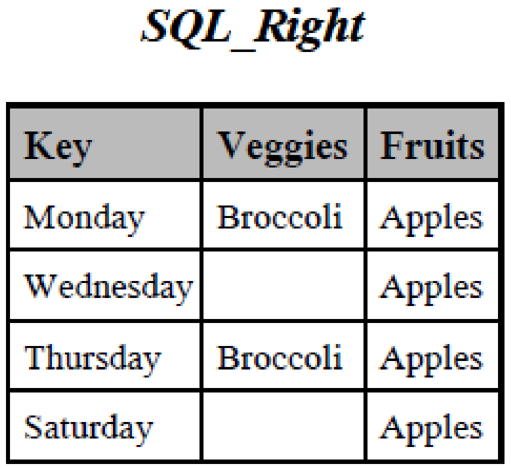
Right Join

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;
```

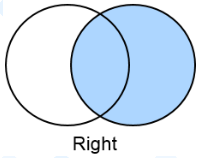


SQL_Right

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;
```

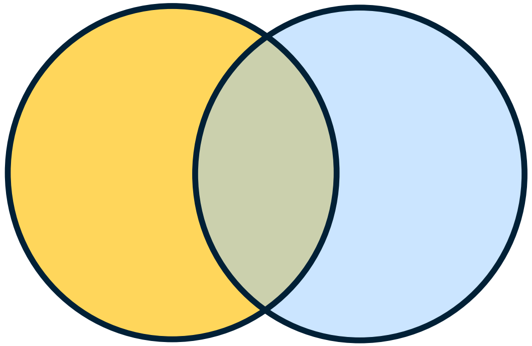
Data_Right

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.



Full

Sample Data

Left

Key	Veggies
Monday	Broccoli
Tuesday	Broccoli
Thursday	Broccoli
Friday	Broccoli

Right

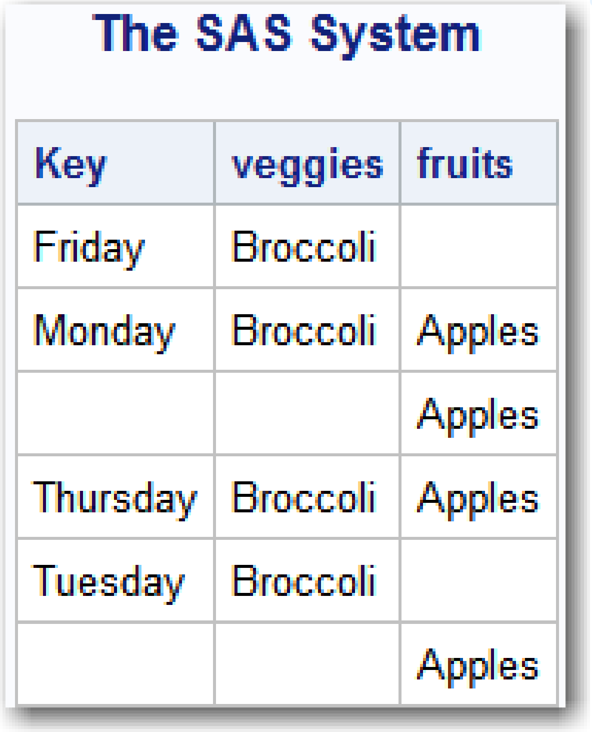
Key	Fruits
Monday	Apples
Wednesday	Apples
Thursday	Apples
Saturday	Apples

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

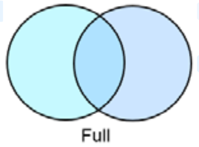
The COALESCE Function

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

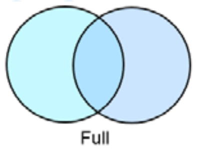
```
proc sql;
/*   create table SQL_Outer as*/
   select coalesce(a.key,b.key) as
Key
         , a.Veggies
         , b.Fruits
   from Left a
      Full join
        Right b
      on a.key = b.key
   ;
quit;
```

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;
```

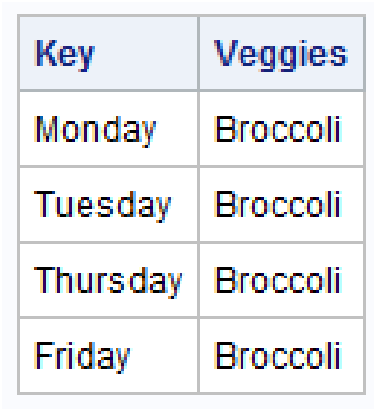
Data_Outer

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

ONE TO MANY

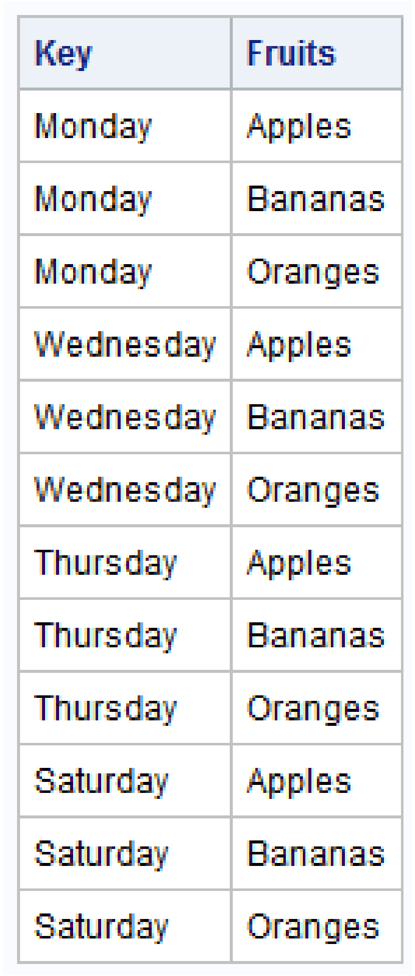
Sample Data

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

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

ONE TO MANY

Default Join

Data_Merge

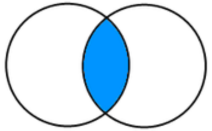
Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Tuesday	Broccoli	
Wednesday		Apples
Wednesday		Bananas
Wednesday		Oranges
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges
Friday	Broccoli	
Saturday		Apples
Saturday		Bananas
Saturday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

SQL_Join

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

ONE TO MANY

INNER JOIN



Data_Inner

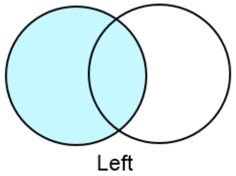
Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges

SQL_Inner

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

ONE TO MANY

LEFT JOIN



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	

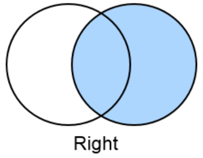
SQL_Left

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!

ONE TO MANY

RIGHT JOIN



Data_Right

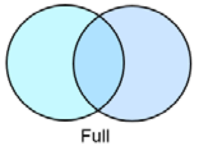
Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Wednesday		Apples
Wednesday		Bananas
Wednesday		Oranges
Thursday	Broccoli	Apples
Thursday	Broccoli	Bananas
Thursday	Broccoli	Oranges
Saturday		Apples
Saturday		Bananas
Saturday		Oranges

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

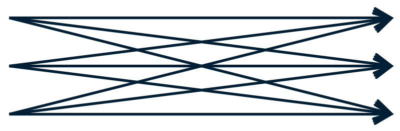
Key	Veggies	Fruits
Monday	Broccoli	Bananas
Monday	Broccoli	Oranges
Monday	Broccoli	Apples
Tuesday	Broccoli	
Wednesday		Oranges
Wednesday		Bananas
Wednesday		Apples
Thursday	Broccoli	Apples
Thursday	Broccoli	Oranges
Thursday	Broccoli	Bananas
Friday	Broccoli	
Saturday		Oranges
Saturday		Bananas
Saturday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

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

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

$$\longrightarrow$$
$$\longrightarrow$$
$$\longrightarrow$$

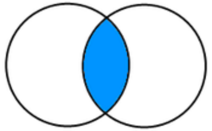
Right

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.
```

MANY TO MANY

Inner Join



Data_Inner

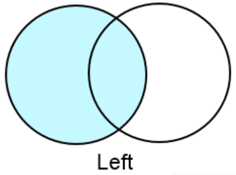
Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Green Beans	Bananas
Monday	Spinach	Oranges
Thursday	Broccoli	Apples
Thursday	Green Beans	Bananas
Thursday	Spinach	Oranges

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

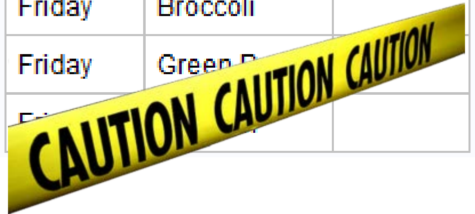
MANY TO MANY

Left Join



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 B	
Fri	[illegible]	

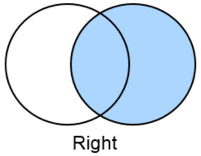


SQL_Left

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

MANY TO MANY

Right Join



Data_Right

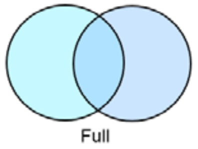
Key	Veggies	Fruits
Monday	Broccoli	Apples
Monday	Green Beans	Bananas
Monday	Spinach	Oranges
Wednesday		Apples
Wednesday		Bananas
Wednesday		Oranges
Thursday	Broccoli	Apples
Thursday	Green Beans	Bananas
Thursday	Spinach	Oranges
Saturday		Apples
Saturday		Bananas
Saturday		

SQL_Right

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

MANY TO MANY

Full (Outer) Join



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		
Saturd		Bananas
[illegible]		Oranges

SQL_Outer

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/hostunx/67132/PDF/default/hostunx.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 ® 9.3 SQL Procedure User's Guide

http://support.sas.com/documentation/cdl/en/sqlproc/63043/PDF/default/sqlproc.pdf

- SAS ® 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#
- SQL Joins -- The Long and The Short of It
  - http://support.sas.com/techsup/technote/ts553.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.jspa

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