

Combining datasets

Statistics 426: SAS Programming

Module 10

2021

Methods for combining datasets

There are several ways to combine datasets in SAS. One thing SAS refuses to do (most often if not all the time) is to combine *unsorted* datasets. All datasets must be sorted before combining methods can be applied

PROC SORT will sort the observations based on values of a variable(s)

Sorting data

The SORT procedure

- rearranges the observations in a SAS data set
- creates a new SAS data set that contains the rearranged observations
- replaces the original SAS data set by default
- can sort on multiple variables
- can sort in ascending or descending order
- does not generate printed output
- treats missing values as the smallest possible values

General form PROC SORT

```
PROC SORT DATA=SAS-data-set <OUT=SAS-data-set>;  
BY <DESCENDING> BY-variable(s);  
RUN;
```

Where

OUT=: option specifies the output data set that contains the data in sorted order (without use of the OUT= option, PROC SORT permanently sorts the data set that is specified in the DATA= option)

BY-variable(s): (required) BY statement specifies one or more variables whose values are used to sort the data, in the order listed in the BY statement

DESCENDING: option in the BY statement sorts observations in descending order; if more than one variable in the BY statement, DESCENDING applies only to the variable that immediately follows it

mtcars dataset

```
filename car url 'https://webpages.uidaho.edu/~reanaes/Data/mtcars.csv';  
  
data mtcars;  
infile car dsd missover firstobs=2;  
input mpg cyl disp hp drat wt qsec am gear carb;  
run;  
proc print data=mtcars;  
run;
```

mtcars print

The SAS System

Obs	mpg	cyl	disp	hp	drat	wt	qsec	am	gear	carb
1	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4
2	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4
3	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4
4	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3
5	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3
6	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3
7	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3
8	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4
9	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4
10	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4

Sort by mpg

```
proc sort data=mtcars
    out=sortcars;
    by mpg;
run;
proc print data=sortcars;
run;
```

mpg sort print

The SAS System

Obs	mpg	cyl	disp	hp	drat	wt	qsec	am	gear	carb
1	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3
2	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3
3	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3
4	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3
5	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3
6	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5
7	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3
8	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3
9	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3
10	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5

sort mpg.png

Sort by mpg and cyl

```
proc sort data=mtcars
    out=sortcars;
    by mpg cyl;
run;
proc print data=sortcars;
run;
```

mpg,cyl sort print

The SAS System

Obs	mpg	cyl	disp	hp	drat	wt	qsec	am	gear	carb
1	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3
2	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3
3	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3
4	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3
5	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3
6	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5
7	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3
8	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3
9	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3
10	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5

sort mpgcyl.png

Sort by mpg (ascending) and cyl (descending)

```
proc sort data=mtcars
    out=sortcars;
    by mpg descending cyl;
run;
proc print data=sortcars;
run;
```

mpg,cyl (desc) sort print

The SAS System										
Obs	mpg	cyl	disp	hp	drat	wt	qsec	am	gear	carb
1	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3
2	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3
3	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3
4	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3
5	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3
6	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5
7	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3
8	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3
9	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3
10	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5

sort mpg-cyl.png

Combining data frames

Appending adds the observations in the second dataset directly to the end of the original dataset

Concatenating copies all observations from the first dataset and then copies all observations from one or more successive datasets into a new dataset

Merging involves combining observations from two data frames by one or more common variables. Observations can be merged based on their positions in the original datasets or merged by one or more common variables

Appending

Appending: the APPEND procedure add the observations from one dataset to the end of another dataset; observations of the second dataset added directly to the end of the original dataset

Requirements of PROC APPEND

- only two datasets can be used at one time in one PROC APPEND
- the observations in the BASE= dataset are not read
- the variable information in the descriptor portion (PROC CONTENTS) of the BASE= dataset cannot change

Like-structured datasets: both datasets have all the same variables

Unlike-structured datasets: both datasets can have some variables in common but both datasets do not have all the same variables

General form of APPEND

```
PROC APPEND BASE=SAS-dataset1
DATA=SAS-dataset2;
RUN;
```

BASE=: dataset names the dataset to which observations will added

DATA=: dataset names the dataset containing the observations that are added to the BASE= dataset

Appending unlike-structured datasets

If you try to append without a special option (with unlike-structured datasets), SAS will give you an error. “No appending done because of anomalies listed above. Use FORCE option to append these files.” Some other warnings are about variables not found on the BASE= dataset; specific variables will not be added to the BASE= dataset.

The FORCE option forces the observations to be appended when the DATA= dataset contains variables that are not in the BASE= dataset

FORCE option in APPEND

FORCE is needed when

- variables are not in the BASE=dataset
- variables do not have the same type as the variables in the BASE= dataset
- variables in the DATA= dataset are longer than the variables in the BASE= dataset

```
PROC APPEND BASE=SASdataset1
DATA=SASdataset2 FORCE;
RUN;
```

Appending like-structured datasets

The following slides have the code to produce new datasets for appending examples

Emps,Emps2008 data

```
data Emps;
  input First $ Gender $ HireYear;
  cards;
Stacey F 2006
Gloria F 2007
James M 2007
;
run;
proc print data=Emps;
run;
```

```
data Emps2008;
  input First $ Gender $ HireYear;
  cards;
Brett M 2008
Renee F 2008
;
run;
proc print data=Emps2008;
run;
```

Emps,Emps2008 print

The SAS System

Obs	First	Gender	HireYear
1	Stacey	F	2006
2	Gloria	F	2007
3	James	M	2007

The SAS System

Obs	First	Gender	HireYear
1	Brett	M	2008
2	Renee	F	2008

Emps2009,Emps2010 data

```
data Emps2009;
  input First $ HireYear;
  cards;
Sara 2009
Dennis 2009
;
run;
proc print data=Emps2009;
run;

data Emps2010;
  input First $ HireYear Country $;
  cards;
Rose 2010 Spain
Eric 2009 Spain
;
run;
proc print data=Emps2010;
run;
```

Emps2009,Emps2010 print

The SAS System

Obs	First	HireYear
1	Sara	2009
2	Dennis	2009

The SAS System

Obs	First	HireYear	Country
1	Rose	2010	Spain
2	Eric	2009	Spain

Appending like-structures

```
proc append base=Emps
            data=Emps2008;
run;
proc print data=Emps;
run;
```

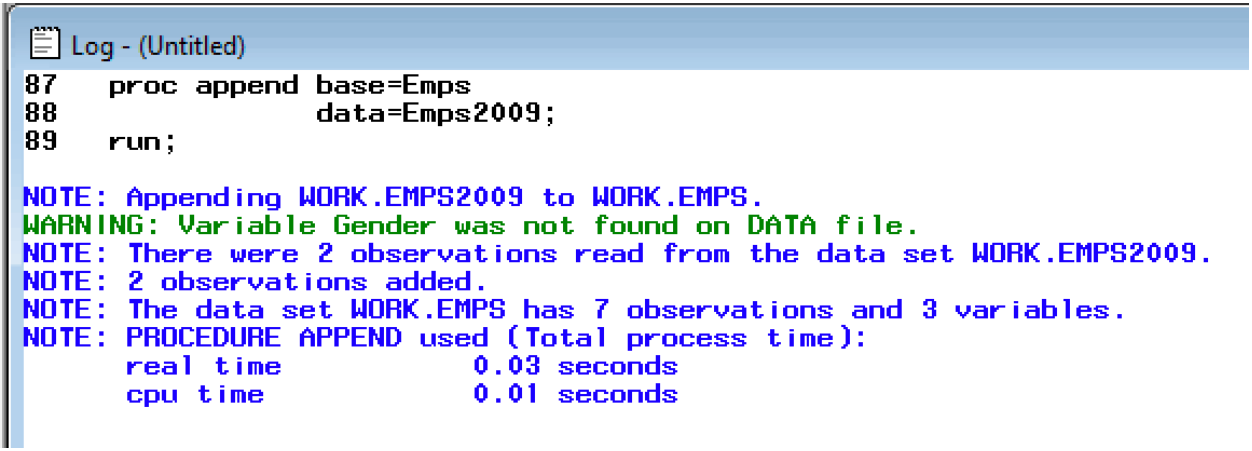

Updated Emps print I

Obs	First	Gender	HireYear
1	Stacey	F	2006
2	Gloria	F	2007
3	James	M	2007
4	Brett	M	2008
5	Renee	F	2008

Appending unlike-structured datasets

```
proc append base=Emps
            data=Emps2009;
run;
proc print data=Emps;
run;
```

Updated Emps II log



```
Log - (Untitled)
87  proc append base=Emps
88      data=Emps2009;
89  run;

NOTE: Appending WORK.EMPS2009 to WORK.EMPS.
WARNING: Variable Gender was not found on DATA file.
NOTE: There were 2 observations read from the data set WORK.EMPS2009.
NOTE: 2 observations added.
NOTE: The data set WORK.EMPS has 7 observations and 3 variables.
NOTE: PROCEDURE APPEND used (Total process time):
      real time          0.03 seconds
      cpu time           0.01 seconds
```

log.png

Updated Emps print II

The SAS System

Obs	First	Gender	HireYear
1	Stacey	F	2006
2	Gloria	F	2007
3	James	M	2007
4	Brett	M	2008
5	Renee	F	2008
6	Sara		2009
7	Dennis		2009

Appending unlike-structured datasets

```
proc append base=Emps
            data=Emps2010;
run;
proc print data=Emps;
run;
```

Updated Emps III log

```
Log - (Untitled)
92 proc append base=Emps
93      data=Emps2010;
94 run;

NOTE: Appending WORK.EMPS2010 to WORK.EMPS.
WARNING: Variable Country was not found on BASE file. The variable will not be added to the BASE
        file.
WARNING: Variable Gender was not found on DATA file.
ERROR: No appending done because of anomalies listed above.
       Use FORCE option to append these files.
NOTE: 0 observations added.
NOTE: The data set WORK.EMPS has 7 observations and 3 variables.
NOTE: Statements not processed because of errors noted above.
NOTE: PROCEDURE APPEND used (Total process time):
      real time          0.03 seconds
      cpu time           0.01 seconds

NOTE: The SAS System stopped processing this step because of errors.

95 proc print data=Emps;
96 run;

NOTE: There were 7 observations read from the data set WORK.EMPS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.03 seconds
      cpu time           0.01 seconds
```

log.png

Updated Emps print III

The SAS System

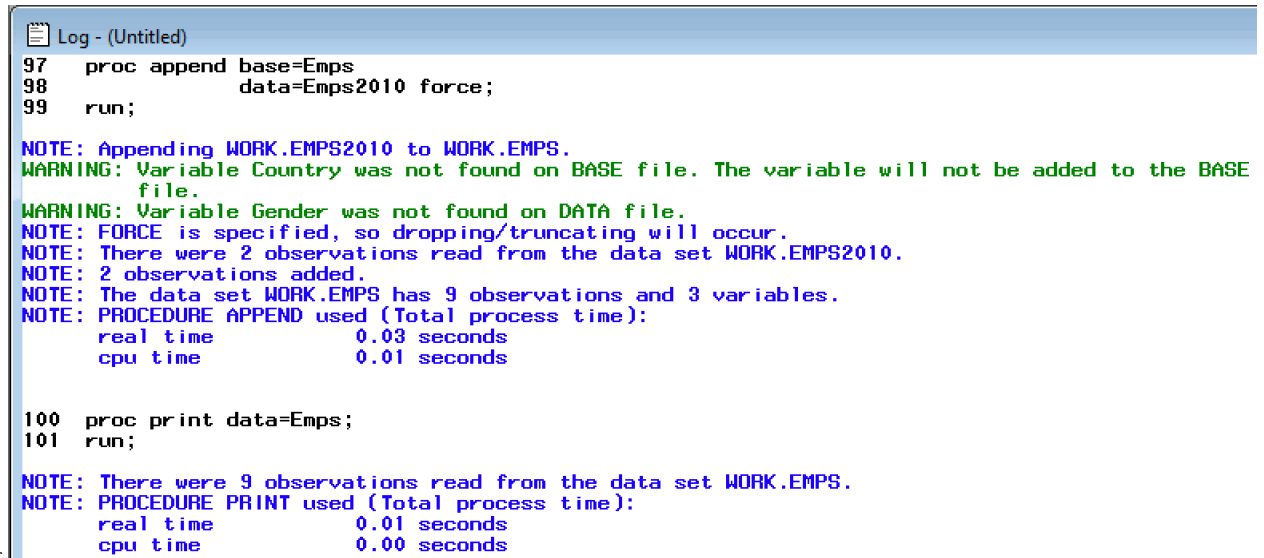
Obs	First	Gender	HireYear
1	Stacey	F	2006
2	Gloria	F	2007
3	James	M	2007
4	Brett	M	2008
5	Renee	F	2008
6	Sara		2009
7	Dennis		2009

Appending using FORCE

```
proc append base=Emps
            data=Emps2010 force;
run;
```

```
proc print data=Emps;
run;
```

FORCE Emps log



```
Log - (Untitled)
97 proc append base=Emps
98      data=Emps2010 force;
99 run;

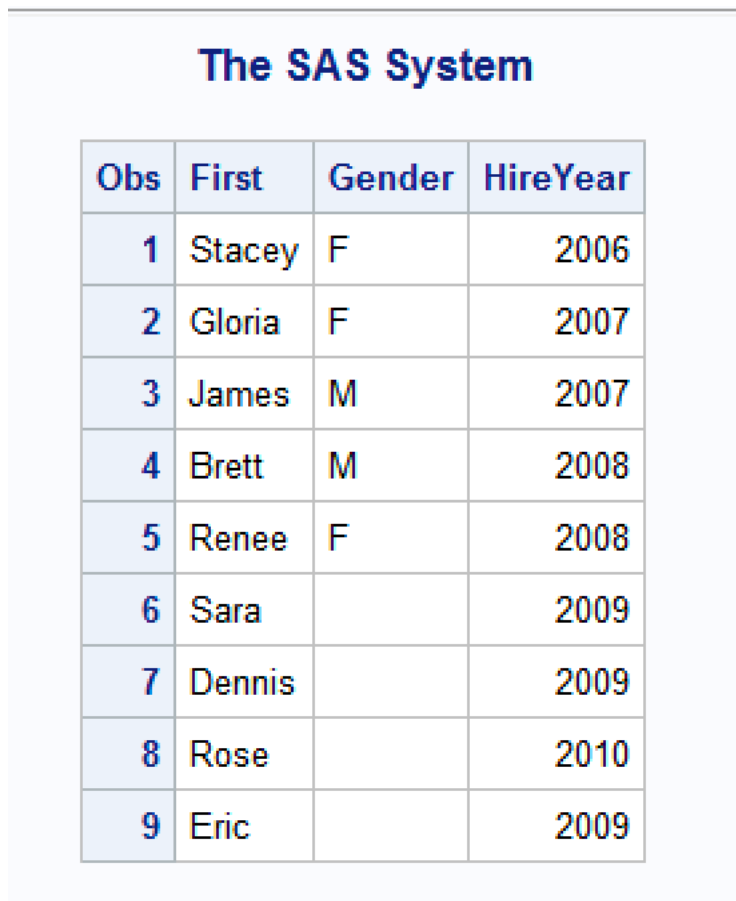
NOTE: Appending WORK.EMPS2010 to WORK.EMPS.
WARNING: Variable Country was not found on BASE file. The variable will not be added to the BASE
        file.
WARNING: Variable Gender was not found on DATA file.
NOTE: FORCE is specified, so dropping/truncating will occur.
NOTE: There were 2 observations read from the data set WORK.EMPS2010.
NOTE: 2 observations added.
NOTE: The data set WORK.EMPS has 9 observations and 3 variables.
NOTE: PROCEDURE APPEND used (Total process time):
      real time          0.03 seconds
      cpu time           0.01 seconds

100 proc print data=Emps;
101 run;

NOTE: There were 9 observations read from the data set WORK.EMPS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time           0.00 seconds
```

force log.png

FORCE Emps



Obs	First	Gender	HireYear
1	Stacey	F	2006
2	Gloria	F	2007
3	James	M	2007
4	Brett	M	2008
5	Renee	F	2008
6	Sara		2009
7	Dennis		2009
8	Rose		2010
9	Eric		2009

force.png

Concatenation

Concatenating:

Use the SET statement in the DATA step. SET reads all observations and variables from one or more SAS datasets.

General form of DATA step with concatenation using SET statement:

```
DATA output-SAS-dataset;
SET SASdataset1 SASdataset2 ... ;
<additional SAS statements>;
RUN;
```

Concatenation logistics

Any number of SAS datasets can be in the SET statement and the observations from the first dataset in the SET statement appear first in the new dataset. The observations from the second dataset follow those from the first dataset, and so on.

You must know your data. By default, a compile-time error (refers to an error from either the operations performed by the compiler, programming language requirements or the properties of the program) occurs if the same variable is not the same type in all SAS datasets in the SET statement

Dataset structure

Like-structured datasets: both datasets have all the same variables

Unlike-structured datasets: both datasets can have some (at least one) variable(s) in common or the datasets have no variables in common

RENAME: Renaming variables as they are concatenated; RENAME= option changes the name of a variable

General form in DATA step

```
DATA output-SAS-dataset;
SET SASdataset1 SASdataset2(RENAME=(variablename2=variablename1));
<additional SAS statements>;
RUN;
```

More dataset creation

```
data EmpsDK;
  input First $ Gender $ Country $;
  cards;
Lars    M    Denmark
Kari    F    Denmark
Jonas   M    Denmark
;
run;
proc print data=EmpsDK;
run;

data EmpsFR;
  input First $ Gender $ Country $;
  cards;
Pierre M    France
Sophie F    France
```

```
;
run;
proc print data=EmpsFR;
run;
```

EmpsDK,EmpsFR print

The SAS System

Obs	First	Gender	Country
1	Lars	M	Denmark
2	Kari	F	Denmark
3	Jonas	M	Denmark

The SAS System

Obs	First	Gender	Country
1	Pierre	M	France
2	Sophie	F	France

Concatenating like-structures

```
data EmpsAll1;
  set EmpsDK EmpsFR;
run;
proc print data=EmpsAll1;
run;
```

EmpsAll1 print

The SAS System

Obs	First	Gender	Country
1	Lars	M	Denmark
2	Kari	F	Denmark
3	Jonas	M	Denmark
4	Pierre	M	France
5	Sophie	F	France

And more datasets...

```
data EmpsCN;
  input First $ Gender $ Country $;
  cards;
Chang  M  China
Li  M  China
Ming  F  China
;
run;
proc print data=EmpsCN;
run;
```

```
data EmpsJP;
  input First $ Gender $ Region $;
  cards;
Yoko  F  Japan
Nobuo  M  Japan
;
run;
proc print data=EmpsJP;
run;
```

EmpsCN,EmpsJP print

The SAS System

Obs	First	Gender	Country
1	Chang	M	China
2	Li	M	China
3	Ming	F	China

The SAS System

Obs	First	Gender	Region
1	Yoko	F	Japan
2	Nobuo	M	Japan

Concatenating unlike-structures

```
data EmpsAll2;  
  set EmpsCN EmpsJP;  
run;  
proc print data=EmpsAll2;  
run;
```


EmpsAll2 print

The SAS System

Obs	First	Gender	Country	Region
1	Chang	M	China	
2	Li	M	China	
3	Ming	F	China	
4	Yoko	F		Japan
5	Nobuo	M		Japan

Concatenating unlike-structures with RENAME

```
data EmpsAll3;  
  set EmpsCN EmpsJP(rename=(Region=Country));  
run;  
proc print data=EmpsAll3;  
run;
```

EmpsAll3 print

The SAS System

Obs	First	Gender	Country
1	Chang	M	China
2	Li	M	China
3	Ming	F	China
4	Yoko	F	Japan
5	Nobuo	M	Japan

Interleave

Interleaving: intersperses observations from two or more datasets, based on one or more common variables

The SET statement with a BY statement in the DATA step interleaves the SAS datasets. It combines the datasets by the one or more common variables in the BY statement.

General form:

```
DATA output-dataset;  
SET SASdataset1 SASdataset2 ... ;  
BY <DESCENDING> by-variable(s);  
<additional SAS statements>  
RUN;
```

Typically, it is more efficient to first sort datasets and *then* interleave them as opposed to concatenating then sorting

Interleaving with RENAME

Must sort first

```
proc sort data=empscn;  
by first;  
run;  
proc sort data=empsjp;  
by first;  
run;  
  
data EmpsAll4;  
  set EmpsCN EmpsJP(rename=(Region=Country));  
  by first;  
run;  
proc print data=EmpsAll4;  
run;
```

EmpsAll4 print

The SAS System

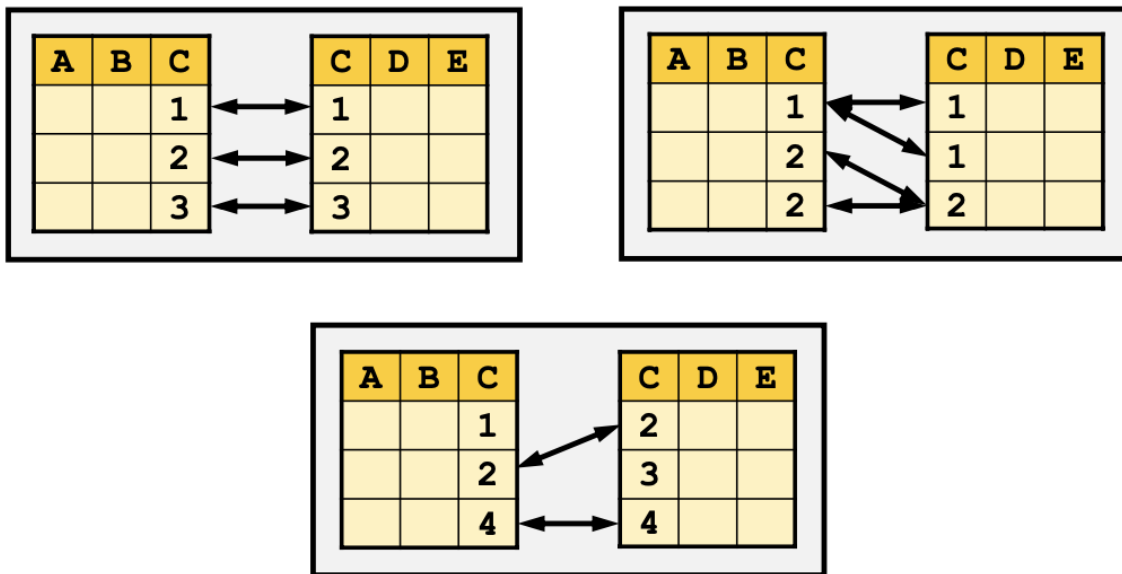
Obs	First	Gender	Country
1	Chang	M	China
2	Li	M	China
3	Ming	F	China
4	Nobuo	M	Japan
5	Yoko	F	Japan

Types of merging

Match-merging: combines observations from 2 or more datasets into a single observation in a new dataset based on the values of one or more common variables

Match-Merging

Match-merging combines observations from two or more SAS data sets into a single observation in a new data set based on the values of one or more common variables.



Types of match-merging

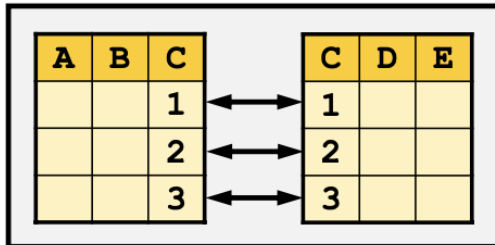
One-to-one: a single observation in one dataset is related to one and only one observation from another dataset based on the values of one or more selected variables

One-to-many or *Many-to-one*: a single observation in one dataset is related to more than one observation from another dataset based on the values of one or more selected variables (1-1); the opposite of this is a many-to-one

Nonmatches: at least one single observation in one dataset is unrelated to any observation from another dataset based on the values of one or more selected variables

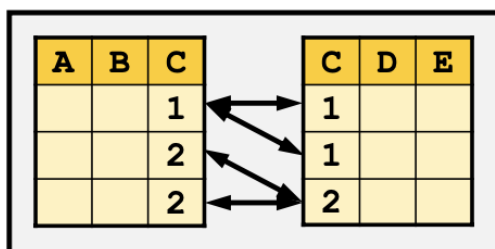
Merging types

Match-Merging



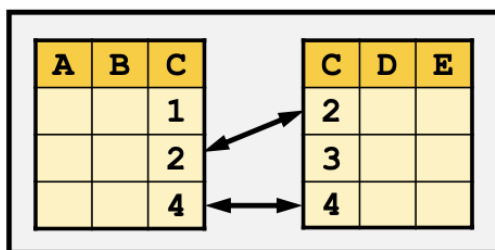
One-to-One

A single observation in one data set is related to one and only one observation from another data set based on the values of one or more selected variables.



One-to-Many or Many-to-One

A single observation in one data set is related to more than one observation from another data set based on the values of one or more selected variables and vice versa.



Nonmatches

At least one single observation in one data set is unrelated to any observation from another data set based on the values of one or more selected variables.

types.png

General form of DATA step with MERGE

Merging using the DATA step and a MERGE statement:

```
DATA output-dataset;  
MERGE SASdataset1 SASdataset2 ... ;  
BY <DESCENDING> by-variable(s);  
<additional SAS statements>  
RUN;
```

Requirements of MERGE statement

Requirements when 2 or more SAS datasets are specified in the MERGE statement:

- variables in the BY statement must be common to *ALL* datasets
- datasets listed in the MERGE statement *must be sorted* in the order of the values of the variables that are listed in the BY statement

Many-to many: If a MERGE statement in a DATA step is used to do a many-to-many merge, an error will most likely be printed to the log as follows:

“NOTE: MERGE statement has more than one dataset with repeats of BY values”

hercules.employee_payroll, employee_addresses data

```
libname hercules 's:\courses\stat-renaes\stat426\data1';
```

```
proc print data=hercules.employee_payroll;  
run;  
proc print data=hercules.employee_addresses;  
run;
```

hercules.employee_payroll print

The SAS System								
Obs	Employee_ID	Employee_Gender	Salary	Birth_Date	Employee_Hire_Date	Employee_Term_Date	Marital_Status	Dependents
1	120101	M	163040	6074	15887	.	S	0
2	120102	M	108255	3510	10744	.	O	2
3	120103	M	87975	-3996	5114	.	M	1
4	120104	F	46230	-2061	7671	.	M	1
5	120105	F	27110	5468	14365	.	S	0
6	120106	M	26960	-5487	5114	.	M	2
7	120107	F	30475	-3997	5145	.	M	2
8	120108	F	27660	8819	17014	.	S	0
9	120109	F	26495	9845	17075	.	M	3
10	120110	M	28615	-3694	7244	.	M	1

hercules.employee_addresses print

The SAS System									
Obs	Employee_ID	Employee_Name	Street_ID	Street_Number	Street_Name	City	State	Postal_Code	Country
1	121044	Abbott, Ray	9260116912	2267	Edwards Mill Rd	Miami-Dade	FL	33135	US
2	120145	Aisbitt, Sandy	1600101803	30	Bingera Street	Melbourne		2001	AU
3	120761	Akinfolarin, Tameaka	9260121030	5	Donnybrook Rd	Philadelphia	PA	19145	US
4	120656	Amos, Salley	9260123736	3524	Calico Ct	San Diego	CA	92116	US
5	121107	Anger, Rose	9260120989	744	Chapwith Rd	Philadelphia	PA	19142	US
6	121038	Anstey, David	9260116991	939	Hilltop Needmore Rd	Miami-Dade	FL	33157	US
7	120273	Antonini, Doris	9260116925	681	Ferguson Rd	Miami-Dade	FL	33141	US
8	120759	Apr, Nishan	9260123711	105	Brack Penny Rd	San Diego	CA	92071	US
9	120798	Ardskin, Elizabeth	9260116954	701	Glenridge Dr	Miami-Dade	FL	33177	US
10	121030	Areu, Jeryl	9260116937	265	Fyfe Ct	Miami-Dade	FL	33133	US

Sort then merge

```
proc sort data=hercules.employee_payroll  
out=payroll;  
by Employee_ID;  
run;  
proc sort data=hercules.employee_addresses  
out=addresses;  
by Employee_ID;  
run;
```

```

data payadd;
  merge payroll addresses;
  by Employee_ID;
run;
proc print data=payadd;
  format Birth_Date weekdate. Salary dollar10.;
run;

```

Payadd print

The SAS System										
Obs	Employee_ID	Employee_Gender	Salary	Birth_Date	Employee_Hire_Date	Employee_Term_Date	Marital_Status	Dependents	Employee_Name	Street
1	120101	M	\$163,040	Wednesday, August 18, 1976	15887	.	S	0	Lu, Patrick	16001025
2	120102	M	\$108,255	Monday, August 11, 1969	10744	.	O	2	Zhou, Tom	16001017
3	120103	M	\$87,975	Saturday, January 22, 1949	5114	.	M	1	Dawes, Wilson	16001030
4	120104	F	\$46,230	Tuesday, May 11, 1954	7671	.	M	1	Billington, Kareen	16001030
5	120105	F	\$27,110	Saturday, December 21, 1974	14365	.	S	0	Povey, Liz	16001018

Dealing with duplicates

Eliminating duplicate values with SORT:

- NODUPKEY option deletes observations with duplicate BY values
- EQUALS option maintains the relative order of the observations with the input dataset in the output dataset for observations with identical BY values
- NODUPRECS is specified in the SORT procedure, duplicate observations are removed based on all of the variables in the input data set

EmpsDUP creation

```

data EmpsDUP;
  input First $ Gender $ EmpID;
  cards;
Matt M 121160
Julie F 121161
Brett M 121162
Julie F 121161
Chris F 121161
Julie F 121163
;
run;
proc print data=EmpsDUP;
run;

```

EmpsDUP print

The SAS System

Obs	First	Gender	EmpID
1	Matt	M	121160
2	Julie	F	121161
3	Brett	M	121162
4	Julie	F	121161
5	Chris	F	121161
6	Julie	F	121163

Eliminating duplicates of the BY statement

```
proc sort data=EmpsDUP
          out=EmpsDUP1 nodupkey equals;
  by EmpID;
run;
proc print data=EmpsDUP1;
run;
```

EmpsDUP1 print

The SAS System

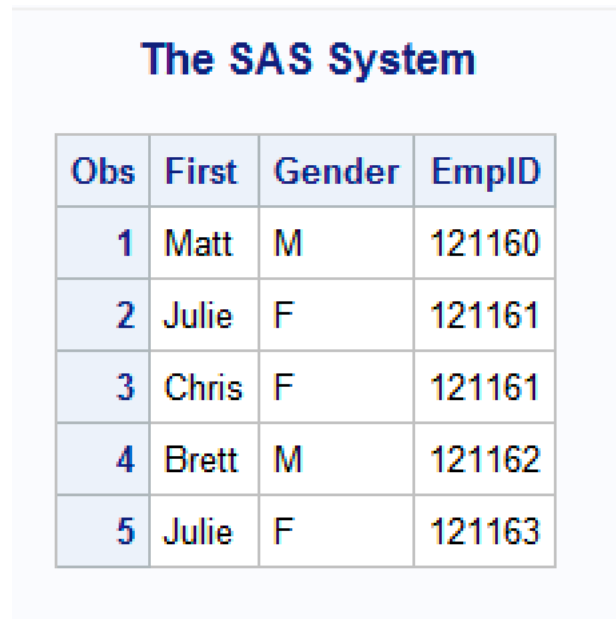
Obs	First	Gender	EmpID
1	Matt	M	121160
2	Julie	F	121161
3	Brett	M	121162
4	Julie	F	121163

Eliminating duplicates based on observation

This will include Chris but with his incorrect employee ID number

```
proc sort data=EmpsDUP
      out=EmpsDUP2 noduprecs;
  by EmpID;
run;
proc print data=EmpsDUP2;
run;
```

EmpsDUP2 print



The SAS System

Obs	First	Gender	EmpID
1	Matt	M	121160
2	Julie	F	121161
3	Chris	F	121161
4	Brett	M	121162
5	Julie	F	121163

Alternative method for combining datasets

The SQL procedure creates different results than the DATA step for a many-to-many merge

PROC SQL can:

- generate reports
- generate summary statistics
- retrieve data from tables (like a SAS dataset)
- combine data from tables
- create tables, views and indexes
- update the data values in PROC SQL tables
- update and retrieve data from database management system (DBMS) tables
- modify PROC SQL tables by adding, modifying or dropping columns (variables)

Often PROC SQL can be an alternative to other SAS procedures or the DATA step