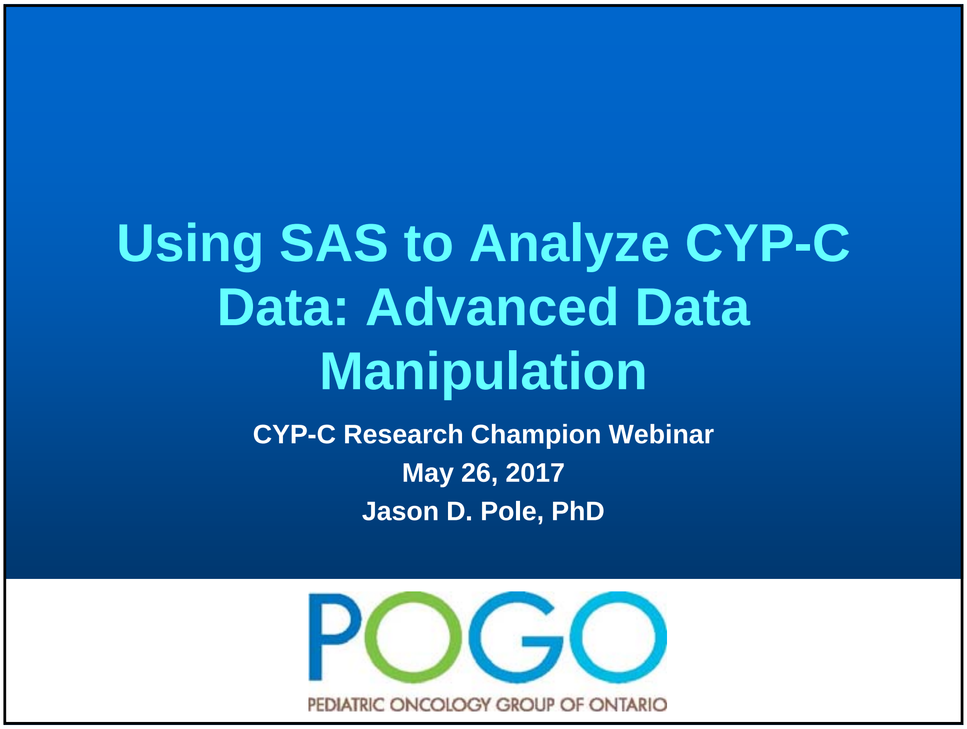


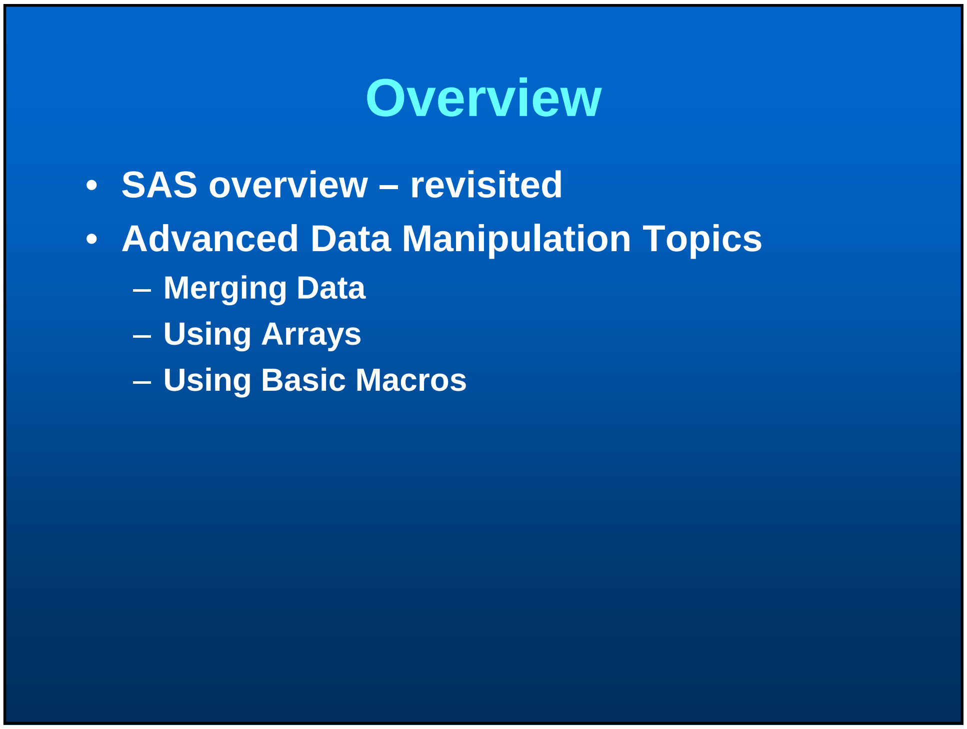
Using SAS to Analyze CYP-C Data: Advanced Data Manipulation

CYP-C Research Champion Webinar

May 26, 2017

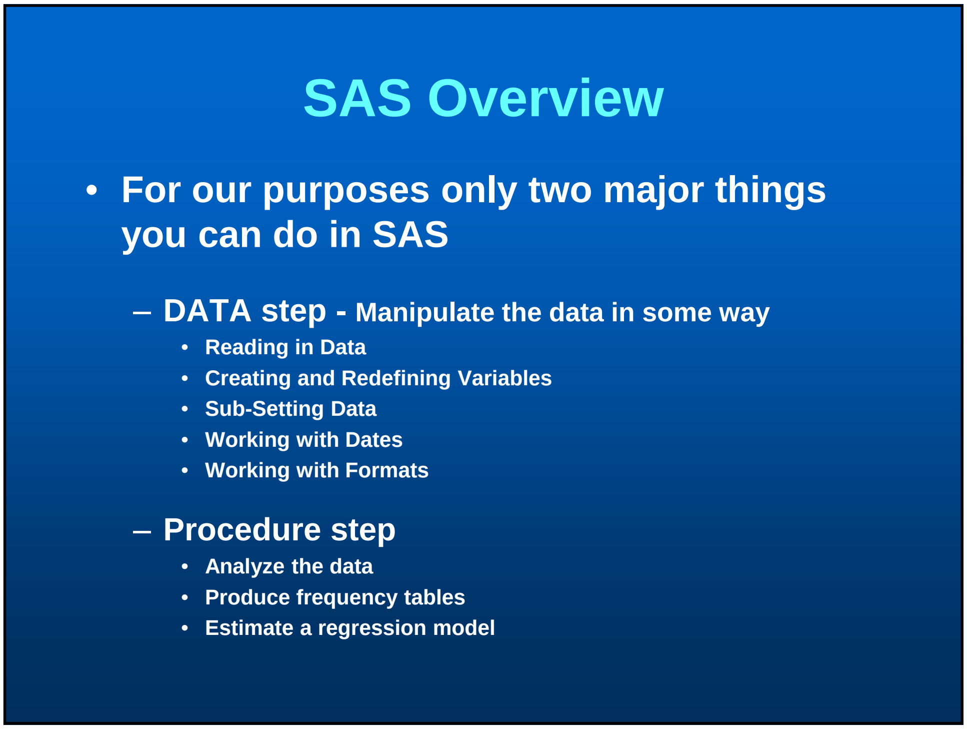
Jason D. Pole, PhD

POGO
PEDIATRIC ONCOLOGY GROUP OF ONTARIO



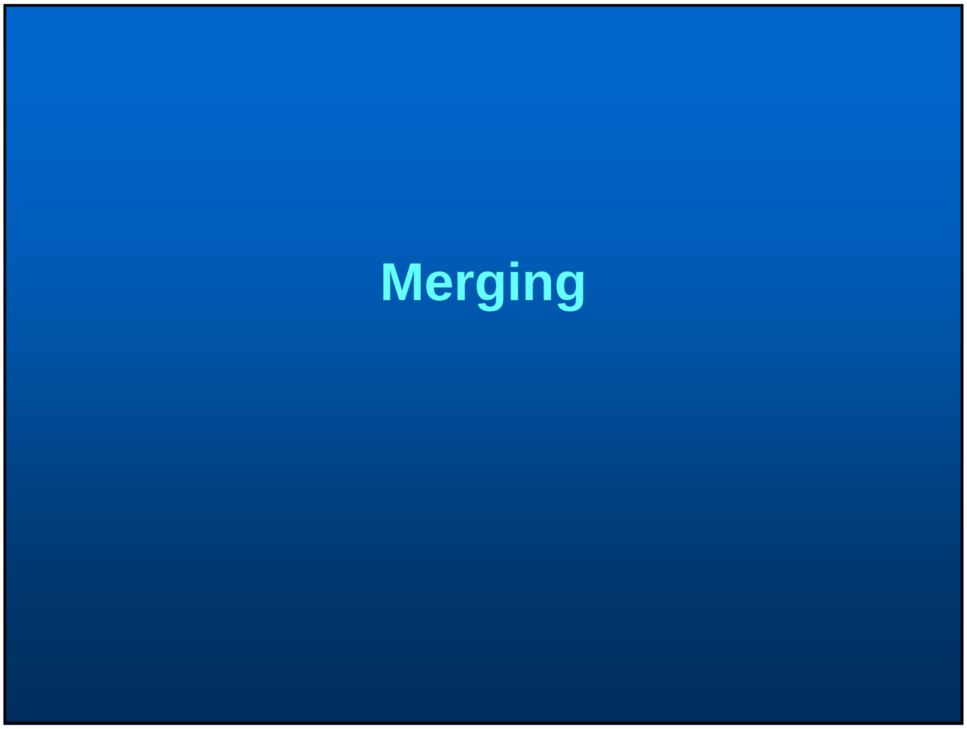
Overview

- SAS overview – revisited
- Advanced Data Manipulation Topics
  - Merging Data
  - Using Arrays
  - Using Basic Macros



SAS Overview

- For our purposes only two major things you can do in SAS
  - DATA step - Manipulate the data in some way
    - Reading in Data
    - Creating and Redefining Variables
    - Sub-Setting Data
    - Working with Dates
    - Working with Formats
  - Procedure step
    - Analyze the data
    - Produce frequency tables
    - Estimate a regression model



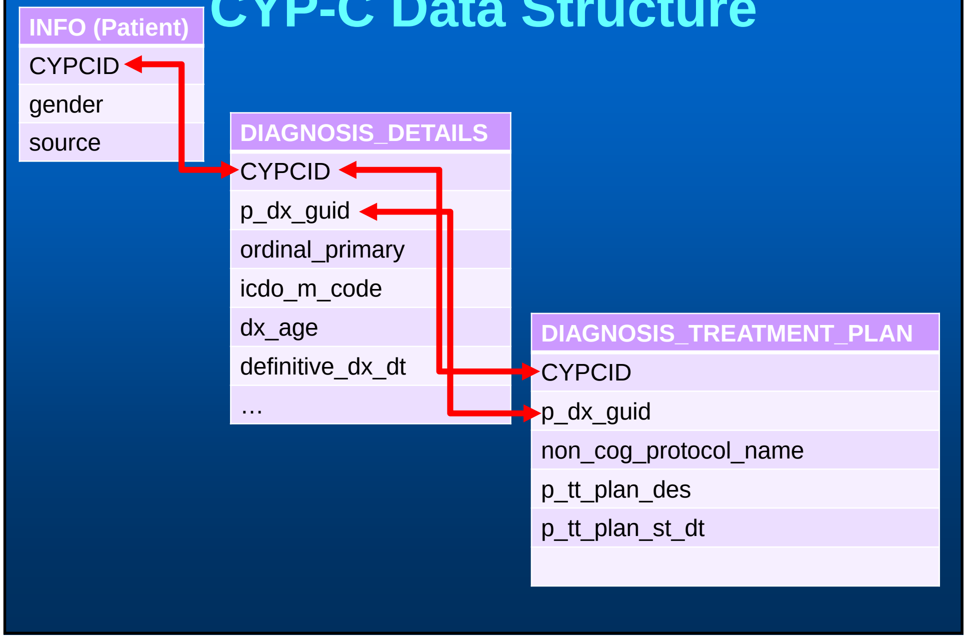
Merging

Merging Datasets in SAS

- Merging is the term used to weave two or more datasets together
- One-to-one merging is simple
- One-to-many or many-to-many is a little more complicated.

- Remember: CYP-C data is relational
  - Given this, you know you are in a many-to-many merging situation
  - More than one record of particular event (ie. more than one diagnosis per patient) that can be linked to other events/patient

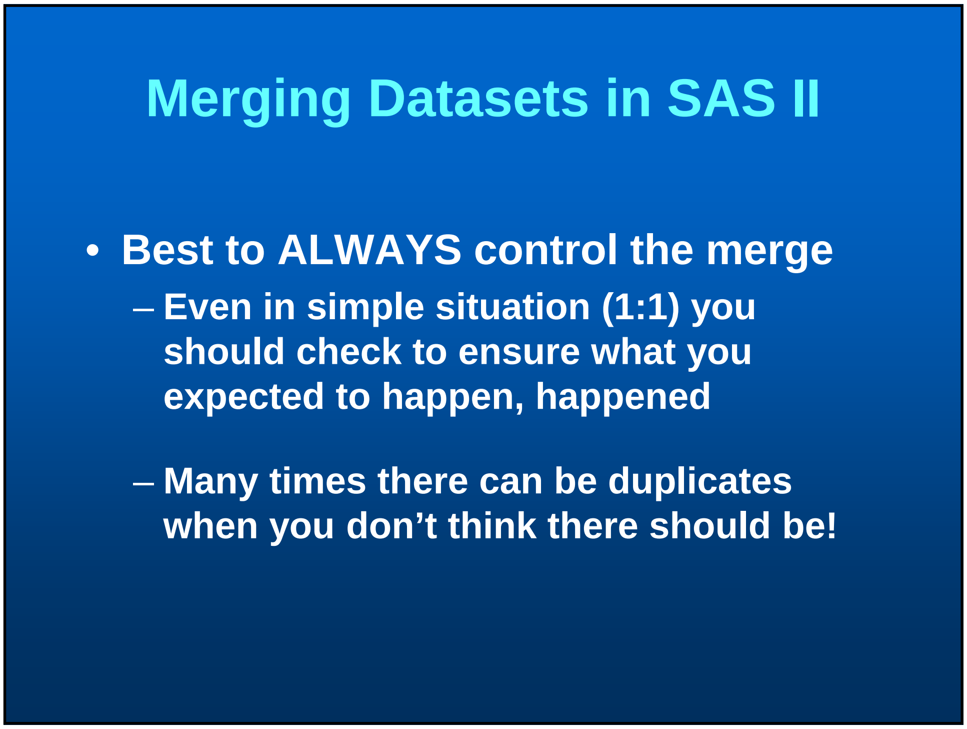
CYP-C Data Structure



INFO (Patient)
- CYPCID
- gender
- source

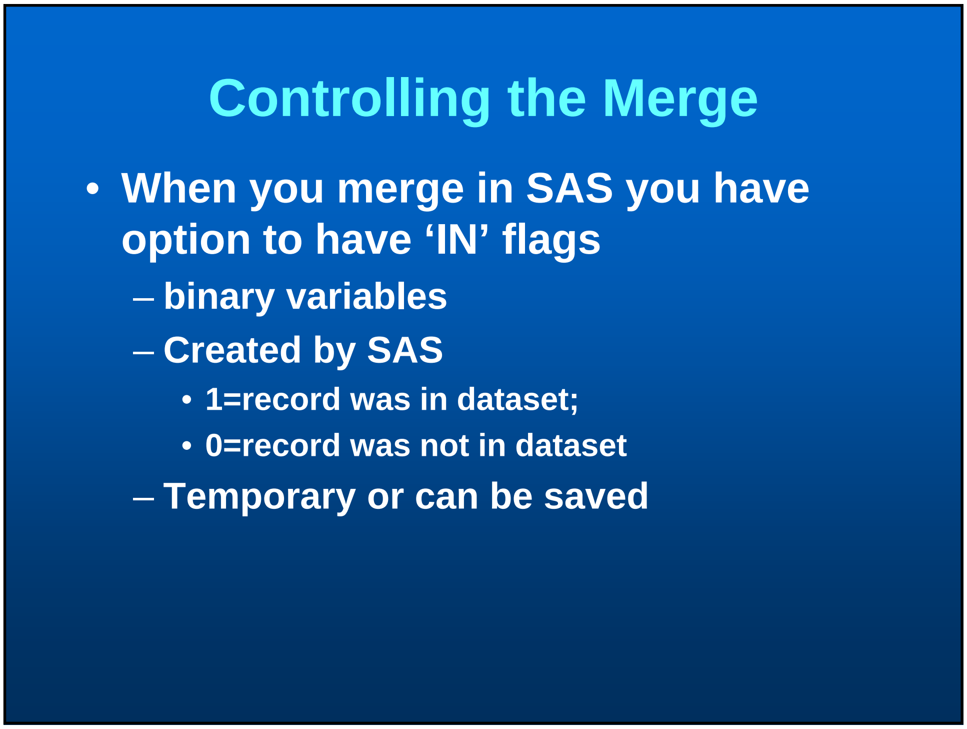
DIAGNOSIS_DETAILS
- CYPCID
- p_dx_guid
- ordinal_primary
- icdo_m_code
- dx_age
- definitive_dx_dt
- …

DIAGNOSIS_TREATMENT_PLAN
- CYPCID
- p_dx_guid
- non_cog_protocol_name
- p_tt_plan_des
- p_tt_plan_st_dt



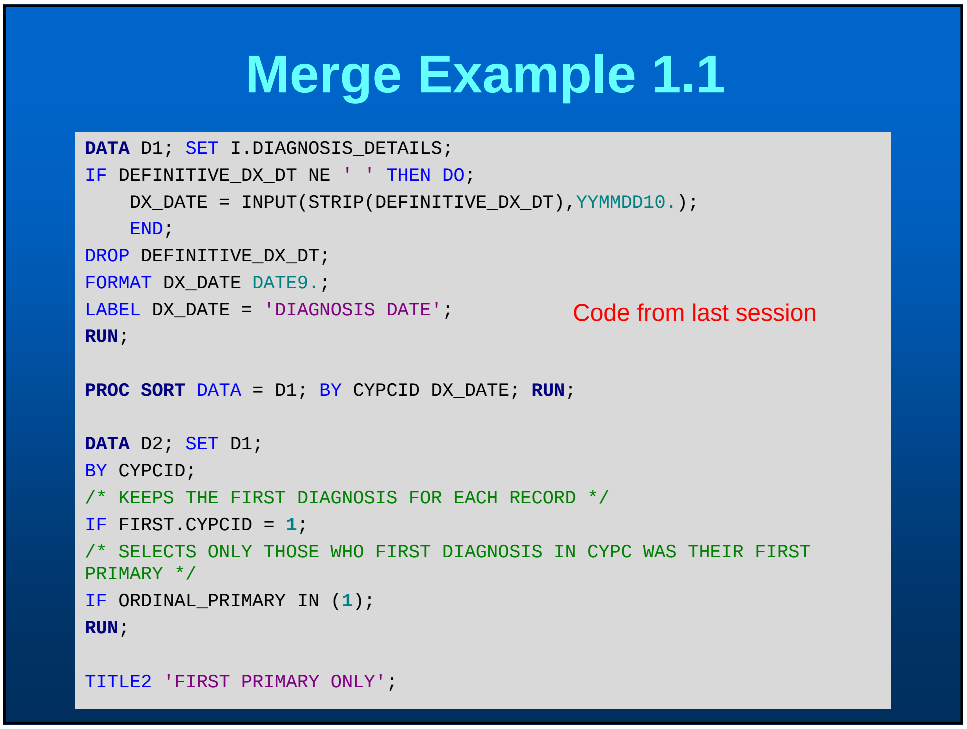
Merging Datasets in SAS II

- Best to ALWAYS control the merge
  - Even in simple situation (1:1) you should check to ensure what you expected to happen, happened
  - Many times there can be duplicates when you don't think there should be!



Controlling the Merge

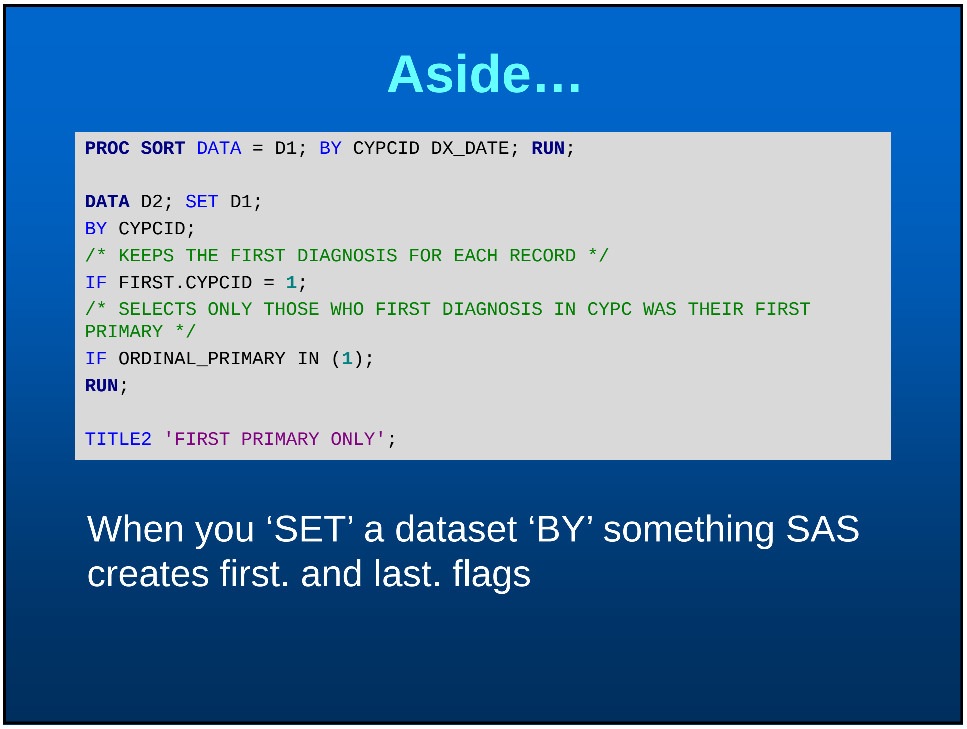
- When you merge in SAS you have option to have 'IN' flags
  - binary variables
  - Created by SAS
    - 1=record was in dataset;
    - 0=record was not in dataset
  - Temporary or can be saved



Merge Example 1.1

```
DATA D1; SET I.DIAGNOSIS_DETAILS;
IF DEFINITIVE_DX_DT NE ' ' THEN DO;
    DX_DATE = INPUT(STRIP(DEFINITIVE_DX_DT),YYMMDD10.);
    END;
DROP DEFINITIVE_DX_DT;
FORMAT DX_DATE DATE9.;
LABEL DX_DATE = 'DIAGNOSIS DATE';
RUN;

PROC SORT DATA = D1; BY CYPCID DX_DATE; RUN;

DATA D2; SET D1;
BY CYPCID;
/* KEEPS THE FIRST DIAGNOSIS FOR EACH RECORD */
IF FIRST.CYPCID = 1;
/* SELECTS ONLY THOSE WHO FIRST DIAGNOSIS IN CYPC WAS THEIR FIRST
PRIMARY */
IF ORDINAL_PRIMARY IN (1);
RUN;

TITLE2 'FIRST PRIMARY ONLY';
```

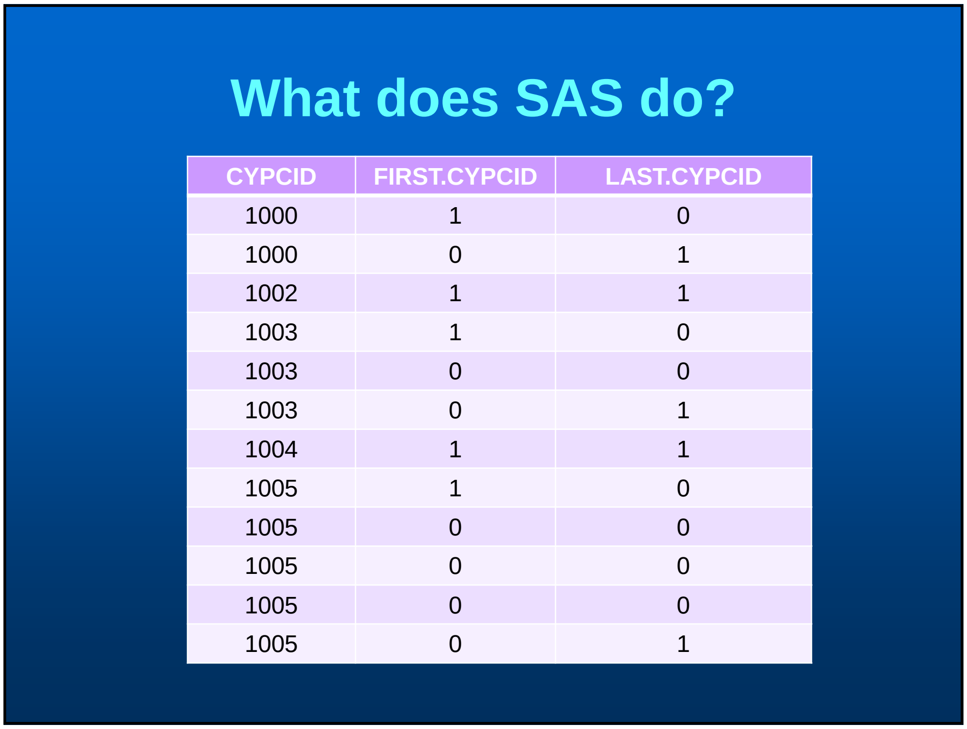
Code from last session



Aside…

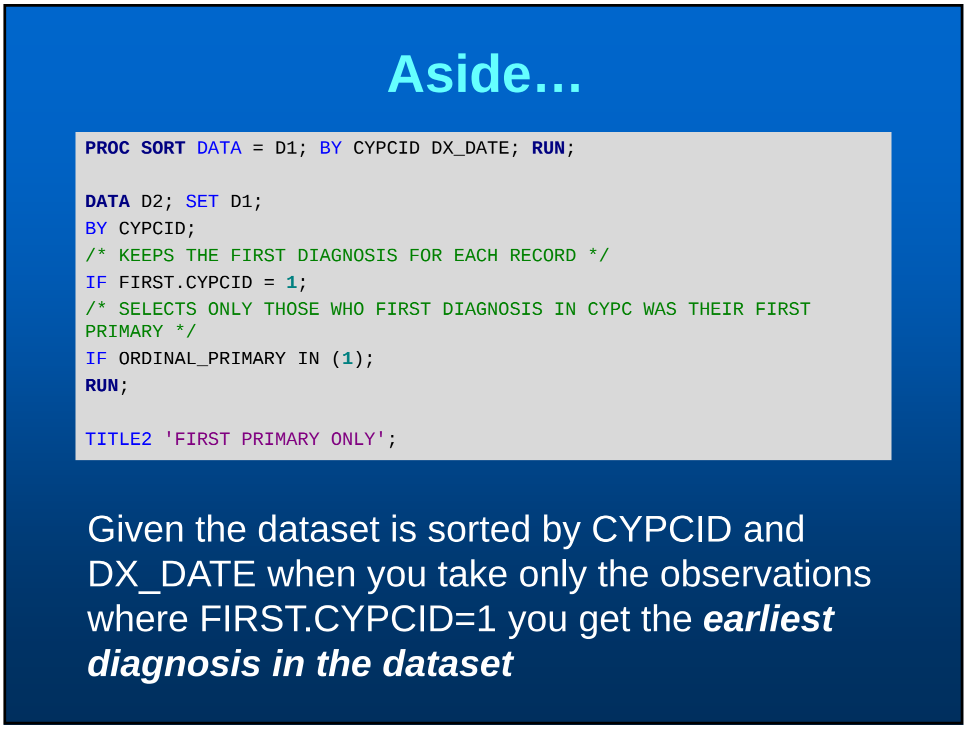
```
PROC SORT DATA = D1; BY CYPCID DX_DATE; RUN;

DATA D2; SET D1;
BY CYPCID;
/* KEEPS THE FIRST DIAGNOSIS FOR EACH RECORD */
IF FIRST.CYPCID = 1;
/* SELECTS ONLY THOSE WHO FIRST DIAGNOSIS IN CYPC WAS THEIR FIRST
PRIMARY */
IF ORDINAL_PRIMARY IN (1);
RUN;

TITLE2 'FIRST PRIMARY ONLY';
```

When you ‘SET’ a dataset ‘BY’ something SAS creates first. and last. flags



What does SAS do?

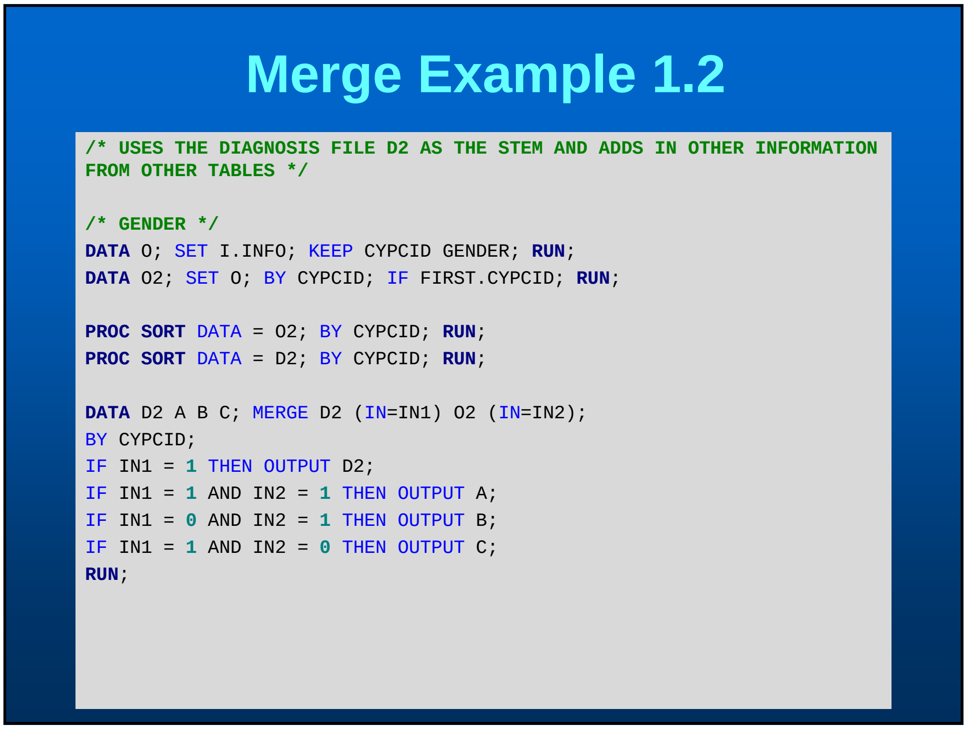
CYPCID	FIRST.CYPCID	LAST.CYPCID
1000	1	0
1000	0	1
1002	1	1
1003	1	0
1003	0	0
1003	0	1
1004	1	1
1005	1	0
1005	0	0
1005	0	0
1005	0	0
1005	0	1



Aside…

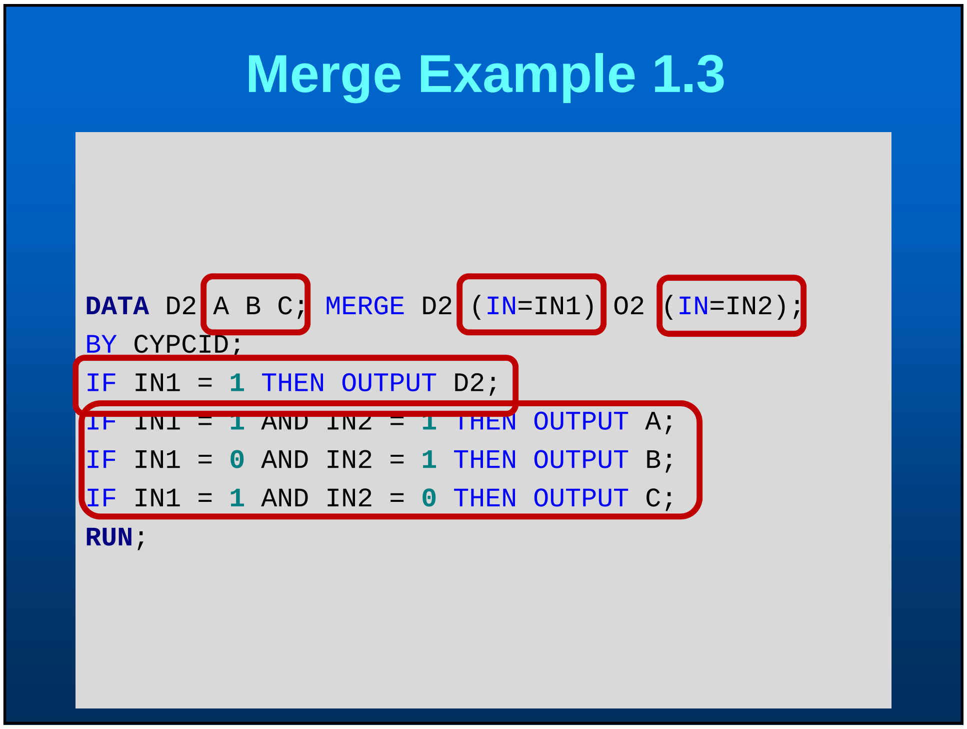
```
PROC SORT DATA = D1; BY CYPCID DX_DATE; RUN;

DATA D2; SET D1;
BY CYPCID;
/* KEEPS THE FIRST DIAGNOSIS FOR EACH RECORD */
IF FIRST.CYPCID = 1;
/* SELECTS ONLY THOSE WHO FIRST DIAGNOSIS IN CYPC WAS THEIR FIRST
PRIMARY */
IF ORDINAL_PRIMARY IN (1);
RUN;

TITLE2 'FIRST PRIMARY ONLY';
```

Given the dataset is sorted by CYPCID and DX_DATE when you take only the observations where FIRST.CYPCID=1 you get the ***earliest diagnosis in the dataset***



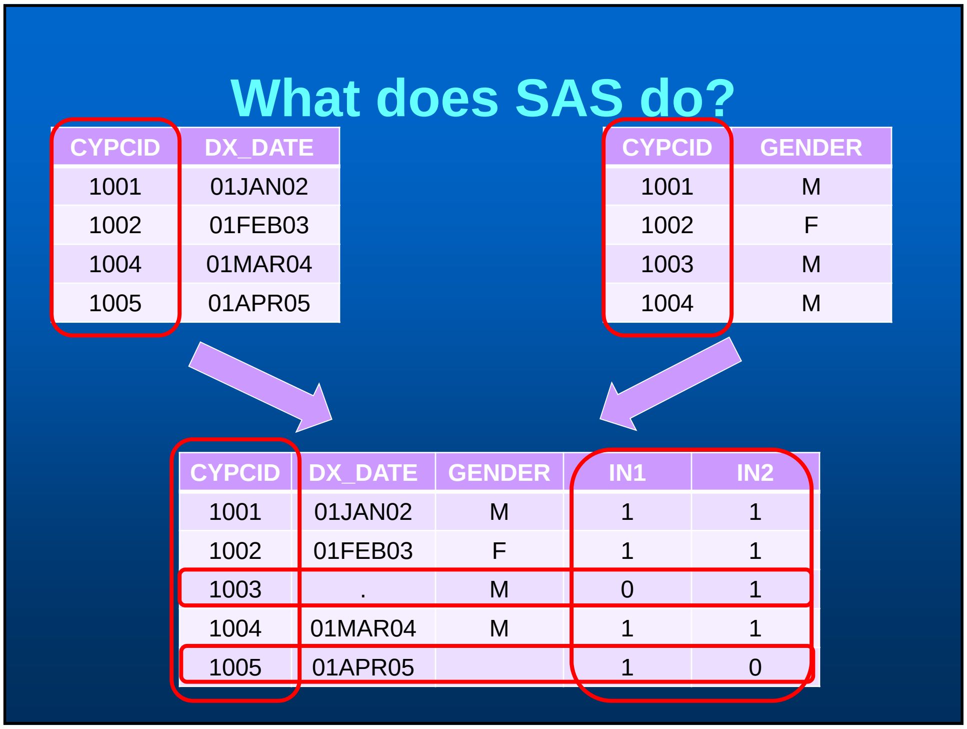
Merge Example 1.2

```
/* USES THE DIAGNOSIS FILE D2 AS THE STEM AND ADDS IN OTHER INFORMATION
FROM OTHER TABLES */

/* GENDER */
DATA O; SET I.INFO; KEEP CYPCID GENDER; RUN;
DATA O2; SET O; BY CYPCID; IF FIRST.CYPCID; RUN;

PROC SORT DATA = O2; BY CYPCID; RUN;
PROC SORT DATA = D2; BY CYPCID; RUN;

DATA D2 A B C; MERGE D2 (IN=IN1) O2 (IN=IN2);
BY CYPCID;
IF IN1 = 1 THEN OUTPUT D2;
IF IN1 = 1 AND IN2 = 1 THEN OUTPUT A;
IF IN1 = 0 AND IN2 = 1 THEN OUTPUT B;
IF IN1 = 1 AND IN2 = 0 THEN OUTPUT C;
RUN;
```



Merge Example 1.3

```
DATA D2 A B C; MERGE D2 (IN=IN1) O2 (IN=IN2);
BY CYPCID;
IF IN1 = 1 THEN OUTPUT D2;
IF IN1 = 1 AND IN2 = 1 THEN OUTPUT A;
IF IN1 = 0 AND IN2 = 1 THEN OUTPUT B;
IF IN1 = 1 AND IN2 = 0 THEN OUTPUT C;
RUN;
```

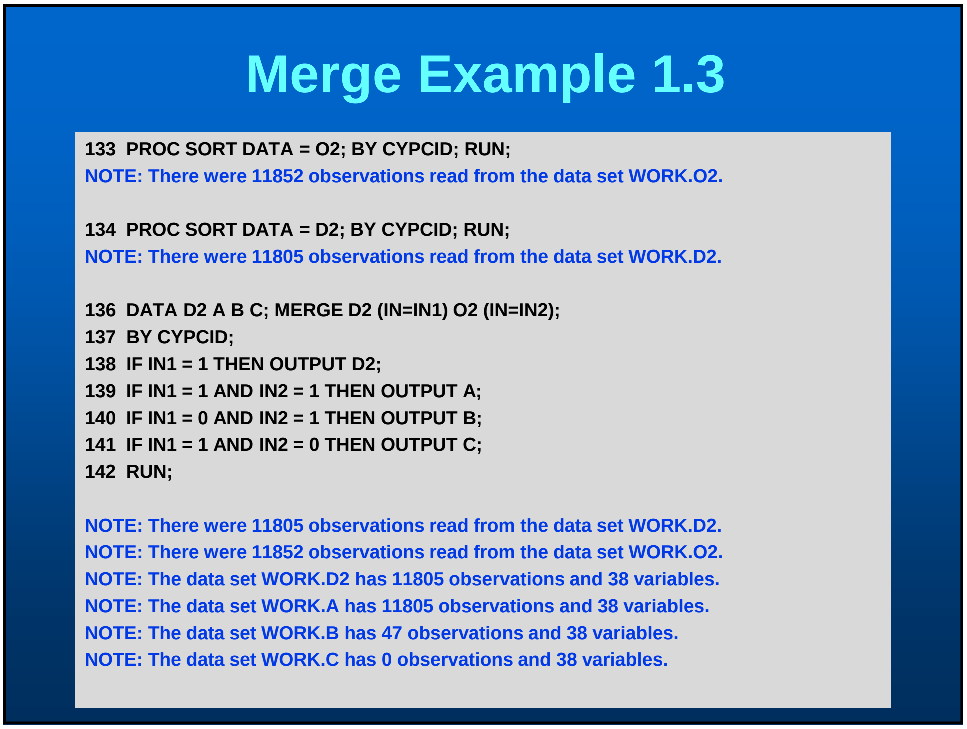


What does SAS do?

CYPCID	DX_DATE
1001	01JAN02
1002	01FEB03
1004	01MAR04
1005	01APR05

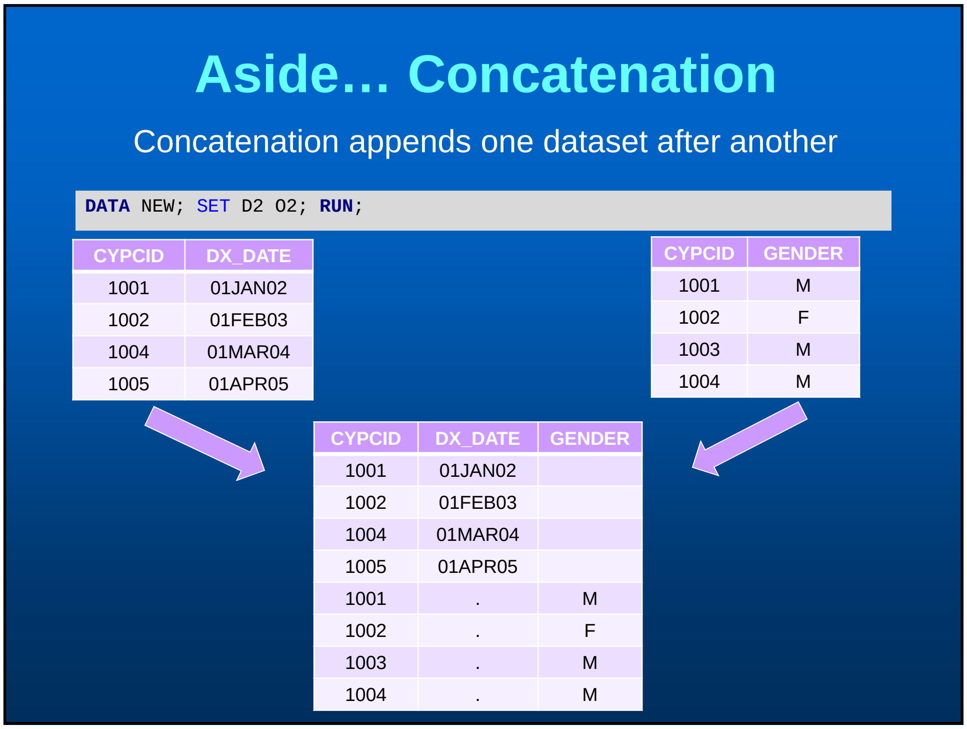
CYPCID	GENDER
1001	M
1002	F
1003	M
1004	M

CYPCID	DX_DATE	GENDER	IN1	IN2
1001	01JAN02	M	1	1
1002	01FEB03	F	1	1
1003	.	M	0	1
1004	01MAR04	M	1	1
1005	01APR05		1	0



Merge Example 1.3

```
133  PROC SORT DATA = O2; BY CYPCID; RUN;
NOTE: There were 11852 observations read from the data set WORK.O2.

134  PROC SORT DATA = D2; BY CYPCID; RUN;
NOTE: There were 11805 observations read from the data set WORK.D2.

136  DATA D2 A B C; MERGE D2 (IN=IN1) O2 (IN=IN2);
137  BY CYPCID;
138  IF IN1 = 1 THEN OUTPUT D2;
139  IF IN1 = 1 AND IN2 = 1 THEN OUTPUT A;
140  IF IN1 = 0 AND IN2 = 1 THEN OUTPUT B;
141  IF IN1 = 1 AND IN2 = 0 THEN OUTPUT C;
142  RUN;

NOTE: There were 11805 observations read from the data set WORK.D2.
NOTE: There were 11852 observations read from the data set WORK.O2.
NOTE: The data set WORK.D2 has 11805 observations and 38 variables.
NOTE: The data set WORK.A has 11805 observations and 38 variables.
NOTE: The data set WORK.B has 47 observations and 38 variables.
NOTE: The data set WORK.C has 0 observations and 38 variables.
```



Aside… Concatenation

Concatenation appends one dataset after another

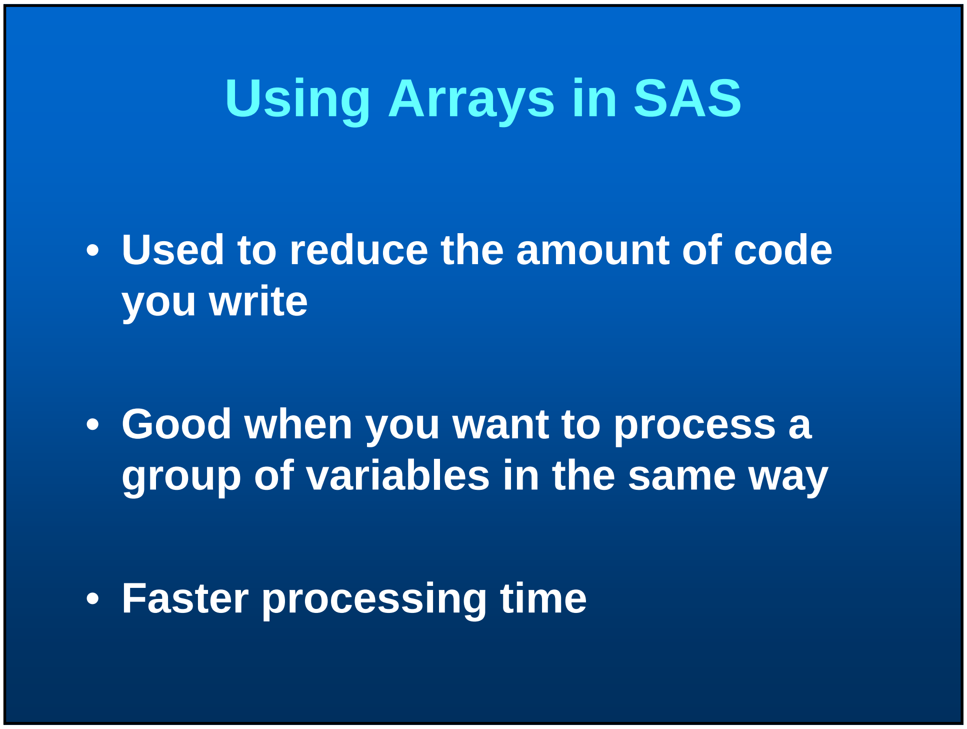
```
DATA NEW; SET D2 O2; RUN;
```

CYPCID	DX_DATE
1001	01JAN02
1002	01FEB03
1004	01MAR04
1005	01APR05

CYPCID	GENDER
1001	M
1002	F
1003	M
1004	M

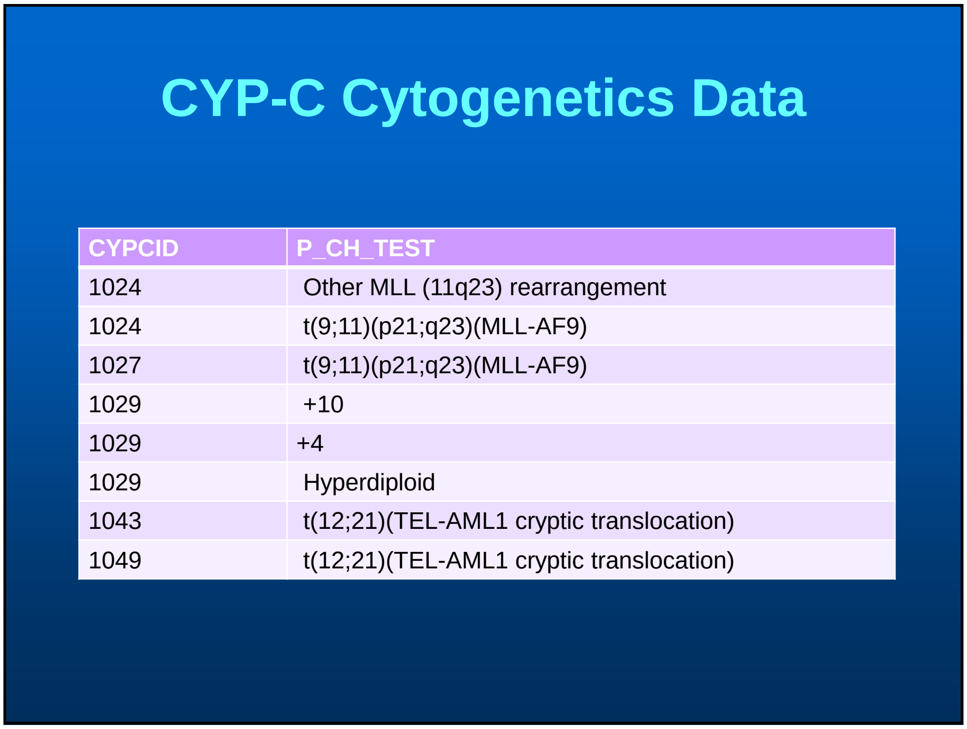
CYPCID	DX_DATE	GENDER
1001	01JAN02	
1002	01FEB03	
1004	01MAR04	
1005	01APR05	
1001	.	M
1002	.	F
1003	.	M
1004	.	M

Arrays



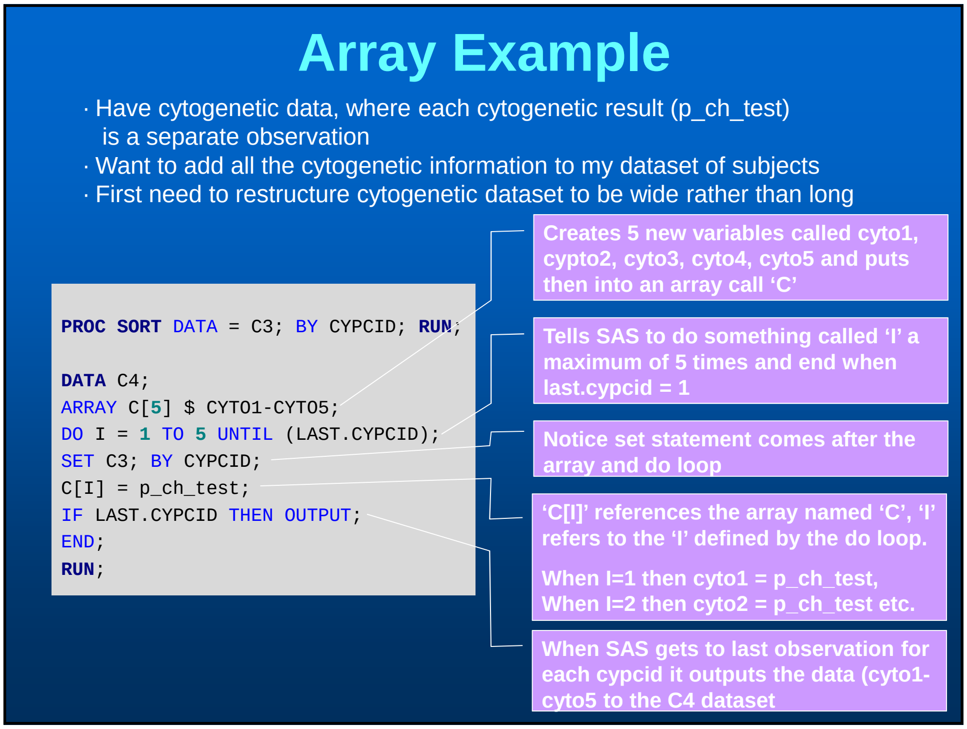
Using Arrays in SAS

- Used to reduce the amount of code you write
- Good when you want to process a group of variables in the same way
- Faster processing time



CYP-C Cytogenetics Data

CYPCID	P_CH_TEST
1024	Other MLL (11q23) rearrangement
1024	t(9;11)(p21;q23)(MLL-AF9)
1027	t(9;11)(p21;q23)(MLL-AF9)
1029	+10
1029	+4
1029	Hyperdiploid
1043	t(12;21)(TEL-AML1 cryptic translocation)
1049	t(12;21)(TEL-AML1 cryptic translocation)



Array Example

- Have cytogenetic data, where each cytogenetic result (p_ch_test) is a separate observation
- Want to add all the cytogenetic information to my dataset of subjects
- First need to restructure cytogenetic dataset to be wide rather than long

```
PROC SORT DATA = C3; BY CYPCID; RUN;

DATA C4;
ARRAY C[5] $ CYTO1-CYTO5;
DO I = 1 TO 5 UNTIL (LAST.CYPCID);
SET C3; BY CYPCID;
C[I] = p_ch_test;
IF LAST.CYPCID THEN OUTPUT;
END;
RUN;
```

Creates 5 new variables called cyto1, cypto2, cyto3, cyto4, cyto5 and puts then into an array call ‘C’

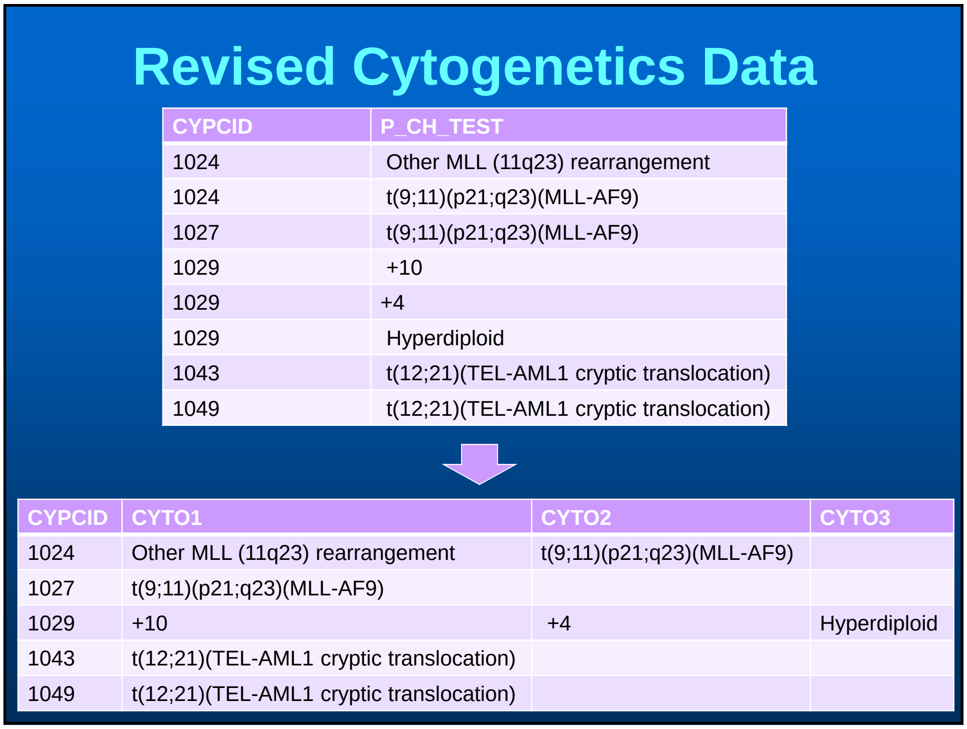
Tells SAS to do something called ‘I’ a maximum of 5 times and end when last.cypcid = 1

Notice set statement comes after the array and do loop

‘C[I]’ references the array named ‘C’, ‘I’ refers to the ‘I’ defined by the do loop.

When I=1 then cyto1 = p_ch_test,
When I=2 then cyto2 = p_ch_test etc.

When SAS gets to last observation for each cypcid it outputs the data (cyto1-cyto5 to the C4 dataset



Revised Cytogenetics Data

CYPCID	P_CH_TEST
1024	Other MLL (11q23) rearrangement
1024	t(9;11)(p21;q23)(MLL-AF9)
1027	t(9;11)(p21;q23)(MLL-AF9)
1029	+10
1029	+4
1029	Hyperdiploid
1043	t(12;21)(TEL-AML1 cryptic translocation)
1049	t(12;21)(TEL-AML1 cryptic translocation)

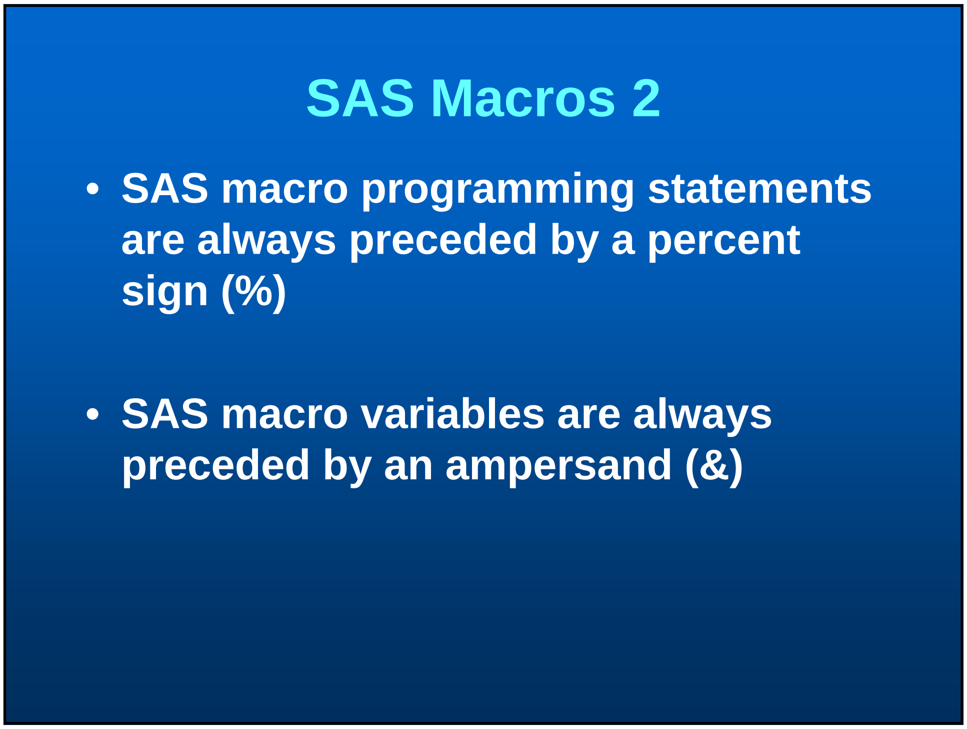
CYPCID	CYTO1	CYTO2	CYTO3
1024	Other MLL (11q23) rearrangement	t(9;11)(p21;q23)(MLL-AF9)	
1027	t(9;11)(p21;q23)(MLL-AF9)		
1029	+10	+4	Hyperdiploid
1043	t(12;21)(TEL-AML1 cryptic translocation)		
1049	t(12;21)(TEL-AML1 cryptic translocation)		

Macros

Basic Macros in SAS

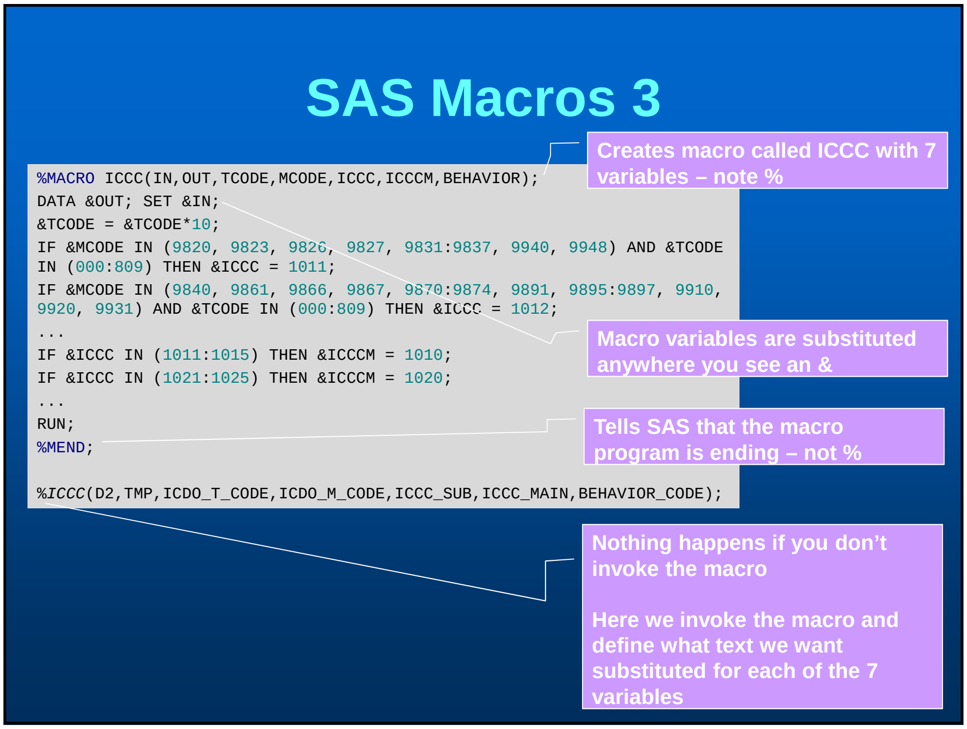
- Helps accomplish repetitive tasks efficiently
- Allows you to assign a string (characters or words) to a variable then be able to use that variable anywhere in the program
- This presentation focuses on the most simple use of the macro language

```
%LET SAVE_TIME = "IF YOU WERE GOING TO TYPE THIS ALL THE TIME";
PROC PRINT; VAR X Y Z; TITLE &SAVE_TIME; RUN;
/************************/
%LET LIST = ICDO_M_CODE p_tt_plan_des source  DumReg DumTumorType
p_tt_plan_tt_non_reg_rsn_des protocol_type P_DX_CENTRE_CODE gender
p_ethnicity_des DX_AGEgrp QAIPPE;
DATA TMP; SET D2; KEEP CYCPCID &LIST; RUN;
```



SAS Macros 2

- SAS macro programming statements are always preceded by a percent sign (%)
- SAS macro variables are always preceded by an ampersand (&)



SAS Macros 3

```
%MACRO ICCC(IN,OUT,TCODE,MCODE,ICCC,ICCCM,BEHAVIOR);
DATA &OUT; SET &IN;
&TCODE = &TCODE*10;
IF &MCODE IN (9820, 9823, 9826, 9827, 9831:9837, 9940, 9948) AND &TCODE
IN (000:809) THEN &ICCC = 1011;
IF &MCODE IN (9840, 9861, 9866, 9867, 9870:9874, 9891, 9895:9897, 9910,
9920, 9931) AND &TCODE IN (000:809) THEN &ICCC = 1012;
...
IF &ICCC IN (1011:1015) THEN &ICCCM = 1010;
IF &ICCC IN (1021:1025) THEN &ICCCM = 1020;
...
RUN;
%MEND;

%ICCC(D2,TMP,ICDO_T_CODE,ICDO_M_CODE,ICCC_SUB,ICCC_MAIN,BEHAVIOR_CODE);
```

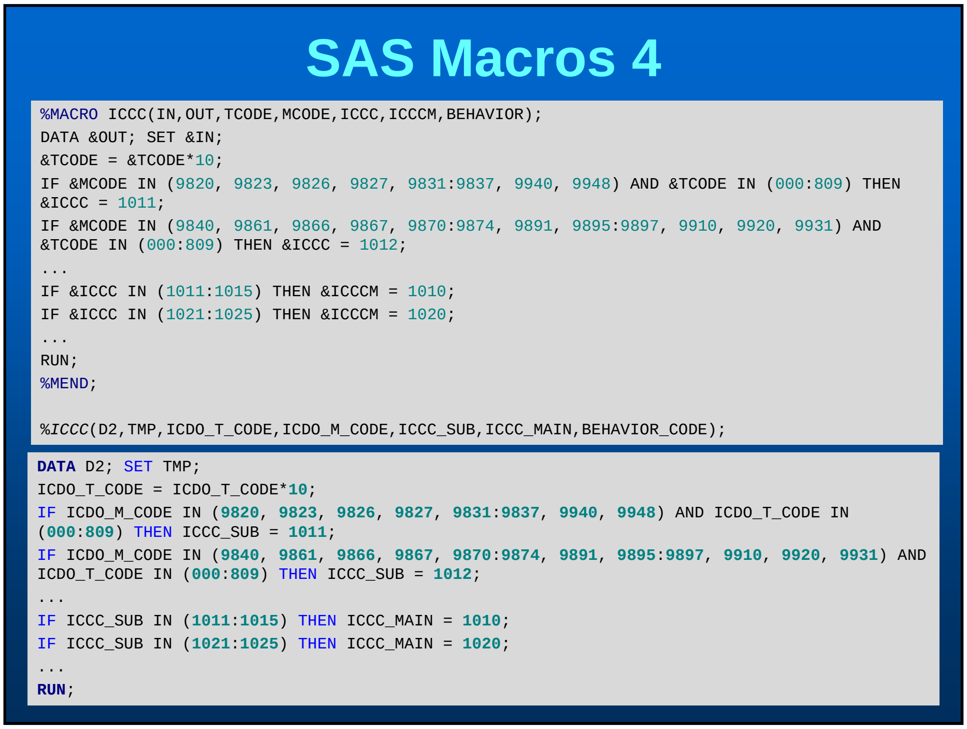
Creates macro called ICCC with 7 variables – note %

Macro variables are substituted anywhere you see an &

Tells SAS that the macro program is ending – not %

Nothing happens if you don't invoke the macro

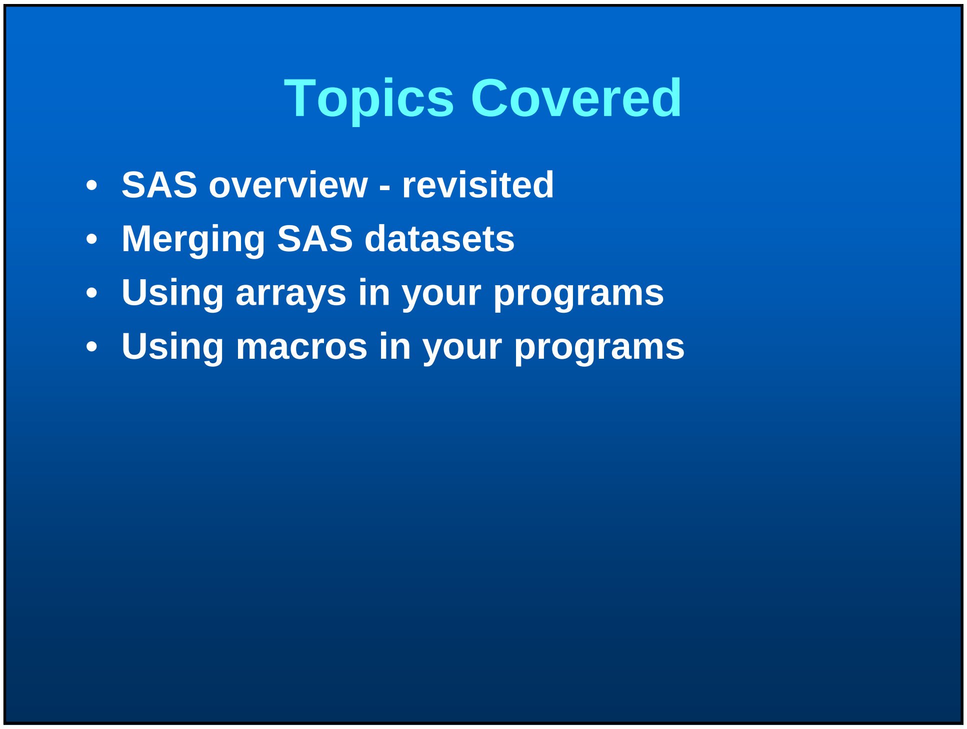
Here we invoke the macro and define what text we want substituted for each of the 7 variables



SAS Macros 4

```
%MACRO ICCC(IN,OUT,TCODE,MCODE,ICCC,ICCCM,BEHAVIOR);
DATA &OUT; SET &IN;
&TCODE = &TCODE*10;
IF &MCODE IN (9820, 9823, 9826, 9827, 9831:9837, 9940, 9948) AND &TCODE IN (000:809) THEN
&ICCC = 1011;
IF &MCODE IN (9840, 9861, 9866, 9867, 9870:9874, 9891, 9895:9897, 9910, 9920, 9931) AND
&TCODE IN (000:809) THEN &ICCC = 1012;
...
IF &ICCC IN (1011:1015) THEN &ICCCM = 1010;
IF &ICCC IN (1021:1025) THEN &ICCCM = 1020;
...
RUN;
%MEND;

%ICCC(D2,TMP,ICDO_T_CODE,ICDO_M_CODE,ICCC_SUB,ICCC_MAIN,BEHAVIOR_CODE);
```

```
DATA D2; SET TMP;
ICDO_T_CODE = ICDO_T_CODE*10;
IF ICDO_M_CODE IN (9820, 9823, 9826, 9827, 9831:9837, 9940, 9948) AND ICDO_T_CODE IN
(000:809) THEN ICCC_SUB = 1011;
IF ICDO_M_CODE IN (9840, 9861, 9866, 9867, 9870:9874, 9891, 9895:9897, 9910, 9920, 9931) AND
ICDO_T_CODE IN (000:809) THEN ICCC_SUB = 1012;
...
IF ICCC_SUB IN (1011:1015) THEN ICCC_MAIN = 1010;
IF ICCC_SUB IN (1021:1025) THEN ICCC_MAIN = 1020;
...
RUN;
```



Topics Covered

- SAS overview - revisited
- Merging SAS datasets
- Using arrays in your programs
- Using macros in your programs