

PROC SQL FOR DATA STEP DIE-HARDS

Christianna Williams, PhD
BASUG
November 9, 2010

INTRODUCTION

- Virtually every SAS programmer uses the DATA step
- PROC SQL is another valuable tool for data manipulation
- Structure of this talk
 - DATA step examples followed by SQL code to achieve same result; start simple...
 - Four logically linked files
- Read the paper!

2

DESCRIPTION OF DATA SETS

Admissions	
Variable	Description
pt_id	Patient identifier
admdate	Date of admission
disdate	Date of discharge
hosp	Hospital identifier
bp_sys	Systolic blood pressure
bp_dia	Diastolic blood pressure
dest	Discharge destination
primdx	Primary diagnosis (ICD-10)
md	Admitting physician ID

3

PROC SQL for DATA Step Die-Hards

DESCRIPTION OF DATA SETS

Patients	
Variable	Description
id	Patient identifier
lastname	Patient last name
firstname	Patient first name
sex	Patient gender (1=M, 2=F)
birthdate	Patient date of birth
primmd	Primary physician identifier

4

PROC SQL for DATA Step Die-Hards

DESCRIPTION OF DATA SETS

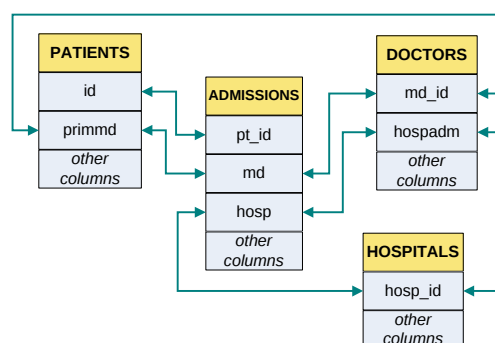
Hospital	
Variable	Description
hosp_id	Hospital identifier
hospname	Hospital name
town	Hospital location
nbeds	Number of beds

Doctors	
Variable	Description
md_id	Physician identifier
hospadm	Hospital at which MD has admit privileges
lastname	Physician last name

5

PROC SQL for DATA Step Die-Hards

LINKAGES AMONG THE DATA SETS



6

EXAMPLE 1
SELECTING VARIABLESDATA step code:

```
DATA selvar1 ;
SET ex.admissions
(KEEP = pt_id admdate
disdate);
RUN;
```

SQL code:

```
PROC SQL;
CREATE TABLE selvar2
AS
SELECT pt_id,
admdate, disdate
FROM ex.admissions ;
QUIT;
```

SAS keywords
in ALL CAPS

NOTE: In SQL items in lists are separated by commas!

7

EXAMPLE 1
SAS LOGDATA step code:

```
DATA selvar1 ;
SET ex.admissions
(KEEP = pt_id
admdate disdate);
RUN;
NOTE: The data set
WORK.SELVAR1 has 22
observations and 3
variables.
```

SQL code:

```
PROC SQL;
CREATE TABLE selvar2
AS
SELECT pt_id,
admdate, disdate
FROM
ex.admissions;
QUIT;
NOTE: Table
WORK.SELVAR2
created, with 22
rows and 3 columns.
```

8

SQL SYNONYMS

DATA step	PROC SQL
data set	table
observation	row
variable	column

PROC SQL for DATA Step Die-Hards

9

EXAMPLE 2
SELECTING OBSERVATIONS

- Objective – Select all admissions to the VA hospital
 - i.e. hosp = 3

PROC SQL for DATA Step Die-Hards

10

EXAMPLE 2
SELECTING OBSERVATIONSDATA step code:

```
DATA vahosp1 ;
SET ex.admissions (WHERE = (hosp EQ 3));
RUN;
```

SQL code:

```
PROC SQL FEEDBACK;
CREATE TABLE vahosp2 AS
SELECT *
FROM ex.admissions
WHERE hosp EQ 3;
QUIT;
```

* = wild card -- all
columns (variables)

11

EXAMPLE 2
SELECTING OBSERVATIONSexpansion of query in logSQL code:

```
PROC SQL FEEDBACK;
CREATE TABLE
vahosp2 AS
SELECT *
FROM
ex.admissions
WHERE hosp EQ 3;
QUIT;
```

NOTE: Statement
transforms to: **select**
ADMISSIONS.PT_ID,
ADMISSIONS.ADMDATE,
ADMISSIONS.DISDATE,
ADMISSIONS.MD,
ADMISSIONS.HOSP,
ADMISSIONS.DEST,
ADMISSIONS.BP_SYS,
ADMISSIONS.BP_DIA,
ADMISSIONS.PRIMDX
from ex.admissions
where
ADMISSIONS.HOSP=3;

12

SELECTING OBSERVATIONS

OUTPUT

PT_ID	ADMDATE	DISDATE	HOSP
003	15MAR2006	15MAR2006	3
008	01OCT2006	15OCT2006	3
008	26NOV2006	28NOV2007	3
014	17JAN2007	20JAN2007	3
018	01NOV2006	15NOV2006	3
018	26DEC2006	08JAN2007	3

PROC SQL for DATA Step Die-Hards

13

EXAMPLE 3

CREATING A NEW VARIABLE

- Objective – create variable DXGRP to categorize primary diagnosis
 - 3 categories

PROC SQL for DATA Step Die-Hards

14

EXAMPLE 3

CREATING A NEW VARIABLE

DATA step:

```
DATA grouping ;
  SET ex.admissions ;
  LENGTH dxgrp $5 ;
  IF primdx EQ: '410' THEN dxgrp = 'MI';
  ELSE IF primdx EQ: '428' THEN dxgrp = 'CHF';
  ELSE dxgrp = 'other' ;
RUN;
```

“EQ:” syntax means “starts like”

15

EXAMPLE 3

CREATING A NEW VARIABLE

SQL code:

```
PROC SQL;
CREATE TABLE grouping2 AS
SELECT *,
CASE
  WHEN primdx LIKE '410%' THEN 'MI'
  WHEN primdx LIKE '428%' THEN 'CHF'
  ELSE 'other'
END AS dxgrp
FROM ex.admissions;
QUIT;
```

“LIKE” + %
equivalent to
EQ: syntax

CASE clause for
new variables
only

16

CREATING A NEW VARIABLE

OUTPUT

pt_id	admdate	primdx	dxgrp
001	07FEB2006	410.0	MI
001	12APR2006	428.2	CHF
001	10SEP2006	813.90	other
001	06JUN2007	428.4	CHF
003	15MAR2006	431	other
004	18JUN2006	434.1	other
005	19JAN2006	411.81	other
005	10MAR2006	410.9	MI

PROC SQL for DATA Step Die-Hards

17

EXAMPLE 4

SELECTING ROWS BASED ON A CREATED VARIABLE

- Objective – select admissions that are at least 2 weeks long
 - create LOS variable
 - assign attributes

PROC SQL for DATA Step Die-Hards

18

EXAMPLE 4

SELECTING ROWS BASED ON A CREATED VARIABLE

DATA step code:

```
DATA twowks1 ;
SET ex.admissions (KEEP = pt_id hosp
                    admdate disdate);
ATTRIB los LENGTH=4
        LABEL='Length of Stay';
los = (disdate - admdate) + 1;

IF los GE 14 ;
RUN;
```

19

EXAMPLE 4

SELECTING ROWS BASED ON A CREATED VARIABLE

SQL code:

```
PROC SQL;
CREATE TABLE twowks2 AS
SELECT pt_id, hosp, admdate, disdate,
       (disdate-admdate) + 1 AS los
       LENGTH=4 LABEL='Length of Stay'
FROM ex.admissions
WHERE CALCULATED los GE 14;
QUIT;
```



CALCULATED keyword is required

20

ROW SELECTION FOR CREATED VARIABLE

OUTPUT

PT_ID	HOSP	ADMDATE	DISDATE	Length of Stay
001	1	12APR2006	25APR2006	14
007	2	28JUL2006	10AUG2006	14
008	3	01OCT2006	15OCT2006	15
009	2	15DEC2006	04JAN2007	21
018	3	01NOV2006	15NOV2006	15
018	3	26DEC2006	08JAN2007	14
020	1	08OCT2007	01NOV2007	25

21

EXAMPLE 5

SELECTING ROWS USING DATA FROM ANOTHER TABLE

◦ **Objective** – select admissions to VA hospital

- Example 2 redux...
- This time we want to select based on *hospital name* ...which isn't on the admissions data set

PROC SQL for DATA Step Die-Hards

22

EXAMPLE 5

SELECTING ROWS USING DATA FROM ANOTHER TABLE

```
PROC SORT DATA = ex.admissions OUT=admits;
BY hosp ;
RUN;

DATA vahosp1d (DROP = hospname) ;
MERGE admits (IN=adm)
      ex.hospitals (IN=va KEEP = hosp_id
                    hospname RENAME = (hosp_id=hosp)
                    WHERE = (hospname EQ: 'Veteran'));
BY hosp ;
IF adm AND va;
RUN;

PROC SORT;
BY pt_id admdate;
RUN;
```

EXAMPLE 5

SELECTING ROWS USING DATA FROM ANOTHER TABLE

SQL code:

```
PROC SQL ;
CREATE TABLE vahosp5 AS
SELECT * FROM ex.admissions
WHERE hosp IN
      (SELECT hosp_id
       FROM ex.hospitals
       WHERE hospname LIKE "Veteran%")
ORDER BY pt_id, admdate ;
QUIT;
```

1st example of a **subquery** – query nested within a query

24

SELECTING ROWS IN ONE TABLE USING DATA FROM ANOTHER TABLE
OUTPUT

PT_ID	ADMDATE	DISDATE	HOSP
003	15MAR2006	15MAR2006	3
008	01OCT2006	15OCT2006	3
008	26NOV2006	28NOV2006	3
014	17JAN2007	20JAN2007	3
018	01NOV2006	15NOV2006	3
018	26DEC2006	08JAN2007	3

PROC SQL for DATA Step Die-Hards

25

EXAMPLE 6
USING SUMMARY FUNCTIONS

```
DATA admsum1;
  SET ex.admissions ;
  BY pt_id;
  IF FIRST.pt_id THEN DO;
    nstays = 0;
    maxlos = .;
  END;
  nstays = nstays + 1;
  los=(disdate-admdate)+1;
  maxlos=MAX(OF maxlos los);
  IF LAST.pt_id THEN OUTPUT;
  RETAIN nstays maxlos;
  KEEP pt_id nstays maxlos;
RUN;
```

Objectives:

- Count # of stays for each patient with a stay
- Get **longest** stay for each patient

Steps:

- Process BY pt_id
- For each BY group
 - Initialization
 - Accumulation
 - Output
- RETAIN is key

26

EXAMPLE 6
USING SUMMARY FUNCTIONS

SQL code:

```
PROC SQL;
  CREATE TABLE admsum2 AS
  SELECT pt_id,
         COUNT(*) AS nstays,
         MAX(disdate - admdate + 1) AS maxlos
  FROM ex.admissions
  GROUP BY pt_id ;
QUIT;
```

COUNT(*) means
count the rows for
each BY group

27

USING SUMMARY FUNCTIONS
OUTPUT

PT_ID	NSTAYS	MAXLOS
001	4	14
003	1	1
004	1	7
005	3	9
007	2	14
008	3	15

PROC SQL for DATA Step Die-Hards

28

EXAMPLE 7
SELECTION BASED ON SUMMARY FUNCTIONS

DATA STEP Code (part 1)

```
PROC SUMMARY DATA =
  ex.admissions ;
VAR bp_sys ;
OUTPUT OUT = bpstats
  MEAN(bp_sys)=mean_sys
  STD(bp_sys)=sd_sys;
RUN;
```

Objective:

Select stays with BP outliers

- PROC SUMMARY
- OUTPUT data set has 1 observation

29

EXAMPLE 7
SELECTION BASED ON SUMMARY FUNCTIONS

DATA STEP Code (part 2)

```
DATA hi_sys1 ;
SET bpstats (keep=mean_sys
  sd_sys)
  ex.admissions ;
IF _N_ EQ 1 THEN DO;
  high = mean_sys + 2*(sd_sys);
  low = mean_sys - 2*(sd_sys);
  DELETE;
END;
RETAIN high low;
IF (bp_sys GE high) OR
  (bp_sys LE low);
DROP mean_sys sd_sys high low;
RUN;
```

Select stays with BP outliers

- Concatenate (SET) summary stats w/ admits
- Specify & RETAIN limits
- Select extremes

30

EXAMPLE 7

SELECTION BASED ON SUMMARY FUNCTIONS

SQL code:

```
PROC SQL ;
CREATE TABLE hi_sys2 AS
  SELECT * FROM ex.admissions
WHERE (bp_sys GE
      (SELECT MEAN(bp_sys)+ 2*STD(bp_sys)
      FROM ex.admissions))
OR (bp_sys LE
    (SELECT MEAN(bp_sys) - 2*STD(bp_sys)
    FROM ex.admissions));
QUIT;
```

No "GROUP BY" -
- so summary
functions apply to
entire table

31

SELECTION BASED ON SUMMARY FUNCTIONS

OUTPUT

pt_id	admdate	bp_sys	bp_dia
001	12APR2006	230	101
003	15MAR2006	74	40
009	15DEC2006	228	92

32

EXAMPLE 8

SELECTION BASED ON SUMMARY FUNCTIONS & "RE-MERGE"

DATA step code (part 1):

```
PROC SUMMARY DATA =
  ex.admissions NWAY;
CLASS dest ;
VAR bp_sys ;
OUTPUT OUT=bpstats
  MEAN(bp_sys)=mean_sys
  STD(bp_sys)=sd_sys;
RUN;
PROC SORT
  DATA=ex.admissions
  OUT=admits;
BY dest;
RUN;
```

Objective: Select stays
with BP outliers by
discharge destination

- PROC SUMMARY w/
CLASS & NWAY
 - Output has 1 observation
per DEST
- Sort admits file

33

EXAMPLE 8

SELECTION BASED ON SUMMARY FUNCTIONS & "RE-MERGE"

Select stays with BP outliers by discharge destination

- MERGE in destination-specific limits (mean & SD)
- Select with subsetting IF

DATA step code (part 2):

```
DATA hi_sys ;
  MERGE admits (KEEP=pt_id bp_sys bp_dia dest)
    bpstats (KEEP = dest mean_sys sd_sys);
BY dest ;
IF bp_sys GE mean_sys + 2*(sd_sys) OR
   bp_sys LE mean_sys - 2*(sd_sys) ;
FORMAT mean_sys sd_sys 6.2;
RUN;
```

EXAMPLE 8

SELECTION BASED ON SUMMARY FUNCTIONS & "RE-MERGE"

PROC SQL;

```
CREATE TABLE hi_sys4 AS
  SELECT pt_id, bp_sys, bp_dia, dest,
    MEAN(bp_sys) AS mean_sys FORMAT=6.2,
    STD(bp_sys) AS sd_sys    FORMAT=6.2
  FROM ex.admissions
  GROUP BY dest
  HAVING bp_sys GE (mean_sys + 2*sd_sys)
    OR bp_sys LE (mean_sys - 2*sd_sys);
QUIT;
```

NOTE: The query requires remerging summary
statistics back with the original data.

35

SELECTION BASED ON SUMMARY FUNCTIONS & "RE-MERGE"

OUTPUT

PT_ID	BP_SYS	BP_DIA	DEST	MEAN_SYS	SD_SYS
001	230	101	1	165.8	30.5
018	199	93	2	151.1	21.3

36

EXAMPLE 10

CREATING TWO DATA SETS FROM ONE

Objective – Separate admissions into two data sets based on year

37

EXAMPLE 10

CREATING TWO DATA SETS FROM ONE

DATA step code:

```
DATA adm06_1 adm07_1;
  SET ex.admissions ;
  IF YEAR(admdate)=2006
    THEN OUTPUT adm06_1;
  ELSE
    IF YEAR(admdate)=2007
      THEN OUTPUT adm07_1;
  RUN;
```

SQL code:

```
PROC SQL;
  CREATE TABLE adm06_2 AS
  SELECT * FROM
    ex.admissions WHERE
      YEAR(admdate)=2006;
  CREATE TABLE adm07_2 AS
  SELECT * FROM
    ex.admissions
  WHERE
    YEAR(admdate)=2007;
  QUIT;
```

single PROC SQL step can have multiple queries

38

TWO DATA SETS FROM ONE

OUTPUT

pt_id	admdate	md	hosp

001	07FEB2006	3274	1
001	12APR2006	1972	1
001	10SEP2006	3274	2
003	15MAR2006	2322	3

pt_id	admdate	md	hosp

001	06JUN2007	3274	2
010	30NOV2007	2322	1
014	17JAN2007	7803	3
015	25MAY2007	4003	5

39

EXAMPLE 11

CONCATENATION

DATA step code:

```
DATA alladm1 ;
  SET admit06_1 admit07_1;
  BY pt_id ;
  RUN;
```

Objective:

Put data sets back together again

- Keep in ID order

SQL code:

```
PROC SQL ;
  CREATE TABLE alladm2 AS
  SELECT * FROM admit06_2
  UNION ALL CORRESPONDING
  SELECT * FROM admit07_2
  ORDER BY pt_id;
  QUIT;
```

UNION is the concatenation operator

CORRESPONDING ensures that like columns align

CONCATENATION

OUTPUT

pt_id	admdate	md	hosp

001	07FEB2006	3274	1
001	12APR2006	1972	1
001	10SEP2006	3274	2
001	06JUN2007	3274	2
003	15MAR2006	2322	3
004	18JUN2006	7803	2
005	19JAN2006	1972	1
005	10MAR2006	1972	1
005	10APR2006	1972	2
007	28JUL2006	3274	2
007	08SEP2006	3274	2

41

EXAMPLE 12

SELECTING ROWS UNIQUE TO ONE TABLE

Objective:

Identify patients with admits in 2006 but not 2007

DATA step code:

```
DATA only2006 ;
  MERGE admit06(IN=in06 keep = pt_id)
        admit07(IN=in07 keep = pt_id);
  BY pt_id ;
  IF in06 AND NOT in07 ;
  IF FIRST.pt_id ;
  RUN;
```

42

EXAMPLE 12

SELECTING ROWS UNIQUE TO ONE TABLE

Objective:

Identify patients with admits in 2006 but not 2007

SQL code:

```
PROC SQL ;
CREATE TABLE only2006 AS
  SELECT pt_id
    FROM admit06
EXCEPT
  SELECT pt_id
    FROM admit07;
QUIT;
```

By default EXCEPT
operator eliminates
duplicates

43

SELECTING ROWS UNIQUE TO ONE TABLE
OUTPUT

Obs	pt_id
1	003
2	004
3	005
4	007
5	008
6	009
7	012
8	018

PROC SQL for DATA Step Die-Hards

44

JOIN TERMINOLOGY

	KEYS				
Table 1:	A	B	C	E	F
Table 2:	A		C	D	F
Inner Join	→ A C F				
Full or Outer Join	→ A B C D E F				
Left Join	→ A B C E F				
Right Join	→ A C D F				

45

EXAMPLE 13

INNER JOIN OF TWO TABLES

Objective – Add additional patient-level information to each admission record

- patient last name
 - patient gender
 - patient's primary MD name
- These fields are all on the PATIENTS table

46

EXAMPLE 13

INNER JOIN OF TWO TABLES

Objective: Add patient variables to each admissionDATA step code:

```
DATA admits1 ;
MERGE
  ex.admissions (IN=adm KEEP=pt_id admdate
                  disdate hosp md)
  ex.patients (IN=pts KEEP=id lastname sex
                primmd
                RENAME = (id=pt_id));
BY pt_id ;
IF adm AND pts;
RUN;
```

EXAMPLE 13

INNER JOIN OF TWO TABLES

SQL code:

```
PROC SQL ;
CREATE TABLE admits2 AS
  SELECT pt_id, admdate, disdate, hosp,
         md, lastname, sex, primmd
    FROM ex.admissions AS a,
         ex.patients AS b
   WHERE a.pt_id = b.id
   ORDER BY a.pt_id, admdate;
QUIT;
```

"a.pt_id" is short for "ex.admissions.pt_id"

48

EXAMPLE 13
INNER JOIN OF TWO TABLES

SQL code:

```
PROC SQL ;
CREATE TABLE admits2 AS
SELECT pt_id, admdate,
       disdate, hosp, md,
       lastname, sex, primmd
FROM
  ex.admissions AS a,
  ex.patients AS b
WHERE a.pt_id = b.id
ORDER BY
  a.pt_id, admdate;
```

SQL code (alternative):

```
PROC SQL ;
CREATE TABLE admits3 AS
SELECT pt_id, admdate,
       disdate, hosp, md,
       lastname, sex, primmd
FROM
  ex.admissions
  INNER JOIN
  ex.patients
  ON pt_id = id
ORDER BY
  pt_id, admdate ;
```

INNER JOIN OF TWO TABLES
OUTPUT

PT_ID	ADMDATE	HOSP	MD	LASTNAME	PRIMMD
001	07FEB2006	1	3274	Williams	1972
001	12APR2006	1	1972	Williams	1972
001	10SEP2006	2	3274	Williams	1972
001	06JUN2007	2	3274	Williams	1972
003	15MAR2006	3	2322	Gillette	.
004	18JUN2006	2	7803	Wallace	4003
005	19JAN2006	1	1972	Abbott	1972
005	10MAR2006	1	1972	Abbott	1972
005	10APR2006	2	1972	Abbott	1972
007	28JUL2006	2	3274	Nickelby	3274
007	08SEP2006	2	3274	Nickelby	3274

50

EXAMPLE 14
JOIN OF 3 TABLES WITH ROW SELECTION

Objectives –

- identify patients who died in the hospital
- include age at death and the number of beds in the hospital
- requires information from three tables, with differing key fields

PROC SQL for DATA Step Die-Hards

51

EXAMPLE 14
JOIN OF 3 TABLES WITH ROW SELECTION

DATA step code (part 1):

```
DATA died1 (RENAME = (disdate=dthdate)) ;
MERGE
  ex.admissions (IN=dth KEEP=pt_id disdate hosp
                 dest WHERE = (dest=9))
  ex.patients (IN=pts KEEP=id birthdate
               RENAME = (id=pt_id));

BY pt_id ;
IF dth AND pts ;
agedth = FLOOR((disdate-birthdate)/365.25) ;
DROP dest birthdate ;
RUN;
```

EXAMPLE 14
JOIN OF 3 TABLES WITH ROW SELECTION

DATA step code (part 2):

```
PROC SORT DATA=died1;
BY hosp; RUN;

DATA died1b ;
MERGE died1 (IN=dth RENAME=(hosp=hosp_id))
  ex.hospitals (IN=hsp KEEP=hosp_id nbeds);
BY hosp_id ;
IF dth AND hsp ;
DROP hosp_id;
RUN;

PROC SORT; BY pt_id ; RUN;
```

DATA step
MERGES must be
“equi-joins”

53

EXAMPLE 14
JOIN OF 3 TABLES WITH ROW SELECTION

SQL code:

```
PROC SQL ;
CREATE TABLE died2 AS
SELECT pt_id, nbeds, disdate AS dthdate,
       INT((disdate-birthdate)/365.25) AS agedth
FROM ex.admissions AS a,
     ex.hospitals AS b,
     ex.patients AS c
WHERE (a.pt_id = c.id) AND
      (a.hosp = b.hosp_id) AND dest EQ 9
ORDER BY pt_id ;
QUIT;
```

The join is ALL
one statement!

54

JOIN OF 3 TABLES WITH ROW SELECTION

OUTPUT

PT_ID	DTHDATE	AGEDTH	NBEDS
001	12JUN2007	75	645
003	15MAR2006	78	1176
009	04JAN2007	88	645

PROC SQL for DATA Step Die-Hards

55

EXAMPLE 15

LEFT OUTER JOIN

Objective

- create a table that has a row for each hospital with indicator of whether there were any admissions at that hospital
- HOSPITALS will be *left* table
- Augment with info from ADMISSIONS as *right* table (if available)

56

EXAMPLE 15

LEFT OUTER JOIN

DATA step code:

```
PROC SORT DATA=ex.admissions
  (KEEP=hosp)
  OUT=admits (RENAME=(hosp=hosp_id))
  NODUPKEY;
BY hosp ;
RUN;
DATA HOSPS1 ;
MERGE ex.hospitals (IN=hosp)
      admits (IN=adm);
  BY hosp_id ;
IF hosp ;
hasadmit = adm ;
RUN;
```

57

EXAMPLE 15

LEFT OUTER JOIN

SQL code:

```
PROC SQL ;
CREATE TABLE hosps2 AS
  SELECT DISTINCT a.*,
                 hosp IS NOT NULL AS hasadmit
  FROM ex.hospitals a
  LEFT JOIN
        ex.admissions b
  ON a.hosp_id = b.hosp;
QUIT;
```

DISTINCT eliminates
duplicates from
resulting table"hosp IS NOT
NULL" true if
match in
admissions data

58

LEFT OUTER JOIN

OUTPUT

HOSP_ID	HOSPNAME	HASADMIT
1	Big University Hospital	1
2	Our Lady of Charity	1
3	Veteran's Administration	1
4	Community Hospital	1
5	City Hospital	1
6	Children's Hospital	0

59

EXAMPLE 16

INNER JOIN WITH A SUBQUERY

Objective

- identify and select admissions where patients were treated by their primary physicians
- include the doctor's name and the patient's name

PROC SQL for DATA Step Die-Hards

60

EXAMPLE 16

INNER JOIN WITH A SUBQUERY

DATA step code (part 1):

```
DATA primdoc (DROP = primmd);
MERGE ex.admissions (IN=adm
      KEEP=pt_id admdate disdate hosp md)
      ex.patients (IN=pts
      KEEP = id lastname primmd
      RENAME = (id=pt_id));
BY pt_id ;
IF adm AND pts AND (md EQ primmd) ;
RUN;
PROC SORT DATA=primdoc; BY md; RUN;
DATA doctors ;
SET ex.doctors (KEEP = md_id lastname);
BY md_id ;
IF FIRST.md_id ;
RUN;
```

61

EXAMPLE 16

INNER JOIN WITH A SUBQUERY

DATA step code, continued:

```
DATA primdoc1a ;
MERGE primdoc (IN=p
      RENAME=(lastname=ptname md=md_id))
      doctors
      (RENAME = (lastname=mdname));
BY md_id ;
IF p ;
RUN;
PROC SORT DATA=primdoc1a ;
BY pt_id admdate;
RUN;
```

62

EXAMPLE 16

INNER JOIN WITH A SUBQUERY

SQL code:

```
PROC SQL ;
CREATE TABLE primdoc2 AS
SELECT pt_id, admdate, disdate, hosp,
md_id,
      b.lastname AS ptname,
      c.lastname AS mdname
FROM ex.admissions a, ex.patients b,
      (SELECT DISTINCT md_id, lastname
      FROM ex.doctors) c
WHERE (a.pt_id EQ b.id) AND
      (a.md EQ b.primmd) AND (a.md EQ c.md_id)
ORDER BY a.pt_id, admdate ;
```

INNER JOIN WITH A SUBQUERY

OUTPUT

PT_ID	ADMDATE	PTNAME	MDNAME
001	12APR2006	Williams	Fitzhugh
005	19JAN2006	Abbott	Fitzhugh
005	10MAR2006	Abbott	Fitzhugh
005	10APR2006	Abbott	Fitzhugh
007	28JUL2006	Nickelby	Hanratty
007	08SEP2006	Nickelby	Hanratty
010	30NOV2007	Alberts	MacArthur
018	01NOV2006	Baker	Fitzhugh
018	26DEC2006	Baker	Fitzhugh

64

CONCLUSIONS

- PROC SQL is a very versatile tool for the manipulation of SAS data sets
 - especially useful when one needs to obtain information from multiple input files
 - can often perform selection, summarization, and ordering with a single statement
- Compatibility with external information systems (e.g. MS Access, Oracle, SQL server)

65

ENCOURAGING WORDS

- Use your next challenging data management task as an opportunity to learn a new method
- You don't have to do it all in one statement!
- So *that's* why they call them "relational databases"

66

