Chapter 3 COMBINING TABLES HORIZONTALLY

USING PROC SQL

A **join** in PROC SQL is a query that combines tables horizontally (that is, side by side) by combining rows. There are two types of joins:

* Inner join (only the rows that match across all tables are output)

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*

* Outer join (rows that match across tables plus nonmatching rows from one or more tables)

\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Left-join right-join full-join

USING INNER JOINS

An **inner join** combines and displays only the rows from the first table that match rows from the second table, based on the matching criteria (also known as **join conditions**) that are specified in the WHERE clause.

Example. Consider two sets

Two

|  |  |
| --- | --- |
| id | varB |
| 2 | x |
| 3 | y |
| 5 | v |

One

|  |  |
| --- | --- |
| id | varA |
| 1 | a |
| 2 | b |
| 4 | d |

The code below displays the observations with ids common to both tables.

data one;

input id varA $ @@;

cards;

1 a 2 b 4 d

;

data two;

input id varB $ @@;

cards;

2 x 3 y 5 v

;

proc sql;

select \*

from one, two

where one.id=two.id;

quit;

The output is

id varA id varB

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

2 b 2 x

ELIMINATING DUPLICATE COLUMNS

If the SELECT clause in the query contains an asterisk, the output displays *all* columns from *both* tables. To eliminate a duplicate column, specify just one of the duplicate columns in the SELECT statement.

proc sql;

select one.id, varA, varB

from one, two

where one.id=two.id;

quit;

The output is

id varA varB

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

2 b x

RENAMING A COLUMN BY USING A COLUMN ALIAS

If there are several tables that have a column with a common name but slightly different data and both columns should appear on the output, then one or several of them may be renamed by specifying a column alias in the SELECT statement.

proc sql;

select one.id as id1, varA, two.id as id2, varB

from one, two

where one.id=two.id; /\* or where id1=id2 \*/

quit;

The output is

id1 varA id2 varB

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

2 b 2 x

JOINING TABLES THAT HAVE ROWS WITH MATCHING VALUES

When two tables are joined that have multiple rows with duplicate values of the column on which tables are being joined, then all possible combinations of the duplicate rows are displayed.

Example. Consider two sets

Four

|  |  |
| --- | --- |
| id | varB |
| 2 | x1 |
| 2 | x2 |
| 3 | y |
| 5 | v |

Three

|  |  |
| --- | --- |
| id | varA |
| 1 | a1 |
| 1 | a2 |
| 2 | b1 |
| 2 | b2 |
| 4 | d |

The following inner join query matches rows from the two tables based on the common column id:

data three;

input id varA $ @@;

cards;

1 a1 1 a2 2 b1 2 b2 4 d

;

data four;

input id varB $ @@;

cards;

2 x1 2 x2 3 y 5 v

;

proc sql;

select \*

from three, four

where three.id=four.id;

quit;

The output is

id varA id varB

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

2 b1 2 x1

2 b1 2 x2

2 b2 2 x1

2 b2 2 x2

SPECIFYING A TABLE ALIAS

To avoid the use of a lengthy data set name in an inner join query, a short temporary alternate name (called **table alias**) may be specified after the table name in the FROM clause.

proc sql;

select \*

from three as a, four as b

where a.id=b.id;

quit;

USING OUTER JOINS

\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Left Outer Join Right Outer Join Full Outer Join

* The output of the **left outer join** consists of all matching rows plus nonmatching rows from the first (the **left**) table specified in the FROM clause.
* The output of the **right outer join** consists of all matching rows plus nonmatching rows from the second (the **right**) table specified in the FROM clause.
* The output of the **full outer join** consists of both matching and nonmatching rows from both tables.
* The syntax is

select *column1*, *column2*, …

from *table1*

left join | right join | full join

*table2*

on *join-condition*

<other clauses>;

quit;

Example.

1. For the data sets one and two, the **left outer join** is

proc sql;

select \*

from one

left join

two

on one.id=two.id;

quit;

The output is

id varA id varB

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

1 a .

2 b 2 x

4 d .

1. For the data sets one and two, the **right outer join** is

proc sql;

select \*

from one

right join

two

on one.id=two.id;

quit;

The output is

id varA id varB

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

2 b 2 x

. 3 y

. 5 v

1. For the data sets one and two, the **full outer join** is

proc sql;

select \*

from one

full join

two

on one.id=two.id;

quit;

The output is

id varA id varB

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

1 a .

2 b 2 x

. 3 y

4 d .

. 5 v

CREATING AN INNER JOIN WITH OUTER JOIN-STYLE SYNTAX

An inner join may be queried using the same style of syntax that is used for an outer join. The syntax is

select *column1*, *column2*, …

from *table1*

inner join

*table2*

on *join-condition*;

<other clauses>;

quit;

* An inner join that uses this syntax can be performed on only two tables.
* An inner join that uses the syntax we studied earlier can combine up to 32 tables at once.

Example. In our example,

proc sql;

select \*

from one

inner join

two

on one.id=two.id;

quit;

The output is

id varA id varB ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

2 b 2 x