

B7.2 - Joining Tables Using SQL in SAS

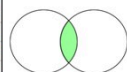
Creating Inner Joins in SQL

class_update

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
David	M	11	55.3	73
Henry	M	14	63.5	102.5

class_teachers

Name	Grade	Teacher
Alfred	8	Thomas
Alice	7	Evans
Barbara	6	Smith
Carol	8	Thomas
Henry	8	Thomas



class_combine

Name	Sex	Age	Height	Weight	Grade	Teacher
Alfred	M	14	69	112.5	8	Thomas
Alice	F	13	56.5	84	7	Evans
Barbara	F	13	65.3	98	6	Smith
Henry	M	14	63.5	102.5	8	Thomas

Only students in both input tables are included.

19

Copyright © SAS Institute Inc. All rights reserved.



Joining tables is a very common requirement when working with data. There are multiple methods available in SAS to join tables. The most common are SQL and the DATA step. In this course, we introduce the SQL inner join. In Module 12 of this course we will address the DATA step merge.

In this example, we have information about students in the **class_update** table, and each student's assigned grade and teacher in the **class_teachers** table. Notice that the **Name** column is common in both tables. We would like to join the tables so that all information for each student is contained in a single result. An inner join will create a new report or table that includes students found in both tables. Notice that *David* is in only **class_update**, and *Carol* is in only **class_teachers**, so they are not included in the inner join result.

Creating Inner Joins in SQL

```
FROM table1 INNER JOIN table2
ON table1.column = table2.column
```

```
proc sql;
select Grade, Age, Teacher
  from pg1.class_update inner join pg1.class_teachers
    on class_update.Name = class_teachers.Name;
quit;
```

20

Copyright © SAS Institute Inc. All rights reserved.

p107d02



What is the syntax required to combine the matching rows from two tables? We can modify the FROM clause to add INNER JOIN, followed by the second table.

Creating Inner Joins in SQL

```
FROM table1 INNER JOIN table2
ON table1.column = table2.column
```

```
proc sql;
select Grade, Age, Teacher
  from pg1.class_update inner join pg1.class_teachers
    on class_update.Name = class_teachers.Name;
quit;
```

matching
criteria

21

Copyright © SAS Institute Inc. All rights reserved.

p107d02



Following the table names, this join syntax requires an ON clause to describe the criteria for matching rows in the tables. Omitting the ON clause produces a syntax error.

The join in this example is an example of a specific type of inner join, referred to as an *equijoin*, where only rows with identical values in the **Name** column will produce a match. The ON condition could also use other comparison operators, such as greater than or less than.

Although not illustrated in this course, outer joins enable you to include nonmatching rows in the results. This is accomplished simply by changing the keyword INNER to OUTER (all nonmatching rows) or RIGHT or LEFT (all rows from one table).

Qualifying Table Names

```
proc sql;
select class_update.Name, Grade, Age, Teacher
  from pg1.class_update inner join pg1.class_teachers
    on class_update.Name = class_teachers.Name;
quit;
```

Because **Name** occurs in both tables, you must use the table prefix to indicate which column you want to select.



22

Copyright © SAS Institute Inc. All rights reserved.

p107d02



Note that the **Name** column is prefixed by one of the table names. This is known as *qualifying* the column names, and it is necessary when you have columns with the same name from more than one table. Qualifying the column name avoids creating an ambiguous column reference, where SAS doesn't know which **Name** column to read.

Demo: Joining Tables with PROC SQL

[7_2 - Demo - Joining Tables with PROC SQL.pdf](#)

<https://clemons.instructure.com/courses/237270/files/23073938/download?wrap=1> ↓

https://clemons.instructure.com/courses/237270/files/23073938/download?download_frd=1 ⓘ

Combining Tables with SQL

```
FROM table1 AS alias1 INNER JOIN table2 AS alias2
```

```
proc sql;  
select u.Name, Grade, Age, Teacher  
  from pg1.class_update as u  
       inner join pg1.class_teachers as t  
       on u.Name=t.Name;  
quit;
```

24

Copyright © SAS Institute Inc. All rights reserved.

sas

Typing the full table names to qualify columns can be tedious. SQL enables you to assign an alias (or nickname) to a table in the FROM clause by adding the keyword AS and the alias of your choice. Then you can use the alias in place of the full table name to qualify columns in the other clauses of a query. In this example, the alias for the two tables are the letters **U** and **T**.

Activity 7.03:

Open **p107a04.sas** from the **activities** folder and perform the following tasks:

1. Define aliases for **storm_summary** and **storm_basincodes** in the FROM clause.
2. Use one table alias to qualify **Basin** in the SELECT clause.
3. Complete the ON expression to match rows when **Basin** is equal in the two tables. Use the table aliases to qualify **Basin** in the expression. Run the step.

```
FROM table1 AS alias1 INNER JOIN table2 AS alias2
ON alias1.column = alias2.column
```

[Click here for Solution.](#)

Comparing SQL and DATA Step

Comparing SQL and the DATA Step


DATA step

- provides more control of reading, writing, and manipulating data
- can create multiple tables in one step
- includes looping and array processing

PROC SQL

- ANSI-standard language used by most databases
- code can be more streamlined
- can manipulate, summarize, and sort data in one step

The DATA step and PROC SQL each have unique strengths.



The DATA step and SQL each provide rich syntax designed to solve our data processing requirements. But each has its own strengths, and therefore it is helpful to know both and the situations where one might be easier or more efficient than the other. The DATA step provides very detailed and customizable control over how data is read, processed, and written. It includes the ability to create multiple tables simultaneously in a single DATA step, which requires reading the input table only once. It also includes syntax for creating loops and processing data in arrays. SQL has the distinct advantage of being a standardized language that is used in most databases. Some SQL syntax can be more streamlined than the equivalent statements in a DATA or PROC step. And as we have seen, SQL can sometimes do in one query what can require multiple steps in SAS, such as

creating a report in sorted order.

Ultimately, it is a great benefit to know both native SAS syntax and SQL and use them when appropriate in your SAS programs.