How Many Ways Can You Join SAS® Tables

HSBC

Charu Shankar
Adoption Services

30 July 2025

Baking With Arrays Vs. Cooking With Hash In-Memory Look Up Techniques

Charu Shankar, SAS® Institute

With a background in computer systems management. SAS Instructor Charu Shankar engages with logic, visuals, and analogies to spark critical thinking since 2007.

Charu curates and delivers unique content on SAS, SQL, Viya, etc. to support users in the adoption of SAS software.

When not coding, Charu teaches yoga and loves to explore Canadian trails with her service dog & husky Miko.

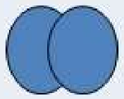
Agenda

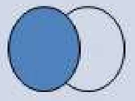
- Nuts and Bolts
- Data Step Merge
- PROC SQL Inner and outer joins
- Handy Links

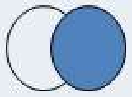
Nuts and Bolts

Merging -SQL vs. Data Step

VISUAL	SQL	DATA STEP

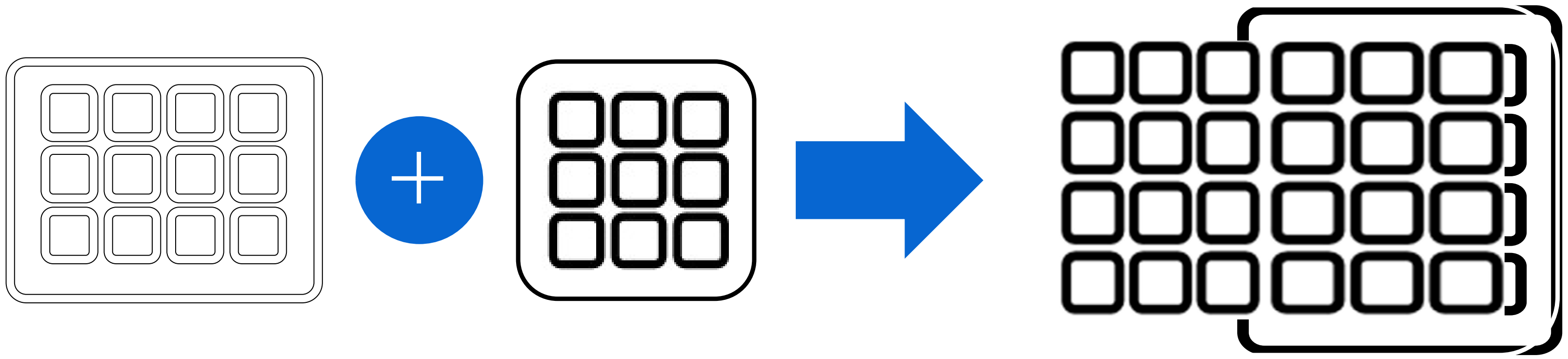
All rows from both tables 	**Full Outer Join**
	Select * from tableA
	Full outer Join
	tableB
	On tableA.id=tableB.id;
		Match Merge
		Data tableC;
		Merge tableA tableB;
		By id;
		Run;

All rows from left table & matching rows from right table 	**Left Join**
	Select * from tableA
	Left Join tableB
	On tableA.id=tableB.id;
		Data Step Merge
		Use IN=data set option
		Data tableD;
		Merge tableA(in=INA) tableB;
		By id;
		If INA;
		Run;

All rows from right table & matching rows from left table 	**Right Join**
	Select * from tableA
	Right Join tableB
	On tableA.id=tableB.id;
		Data Step Merge
		Data tableE;
		Merge tableA tableB(in=INB);
		By id;
		If INB;
		Run;

Data Step Merge

Merging Tables



Merging Tables

one-to-one

A	B	C
		1
		2
		3

C	D	E
1		
2		
3		

nonmatching rows

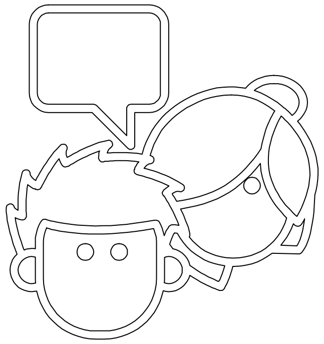
A	B	C
		1
		2
		4

C	D	E
2		
3		
4		

one-to-many

A	B	C
		1
		2

C	D	E
1		
1		
2		



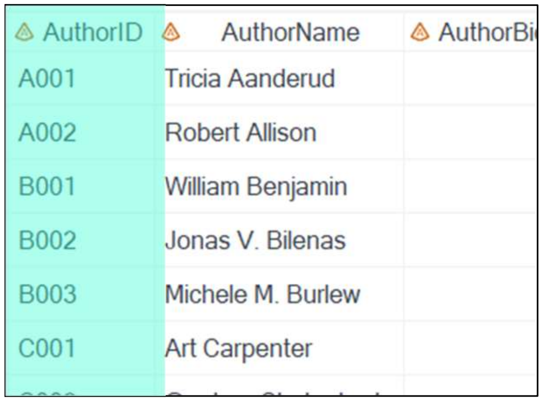
Discussion

Invited	Attending
Drew	Caroline
Lani	Drew
Mansfield	Michael
Caroline	Lani
Kristin	Kristin
Michael	
George	

Invited	Attending
Caroline	Caroline
Drew	Drew
George	Kristin
Kristin	Lani
Lani	Michael
Mansfield	
Michael	

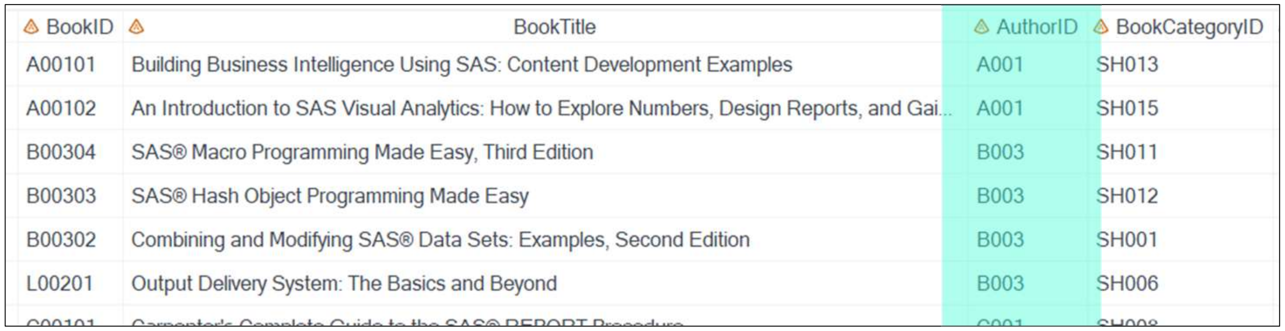
Merging Tables

Authors

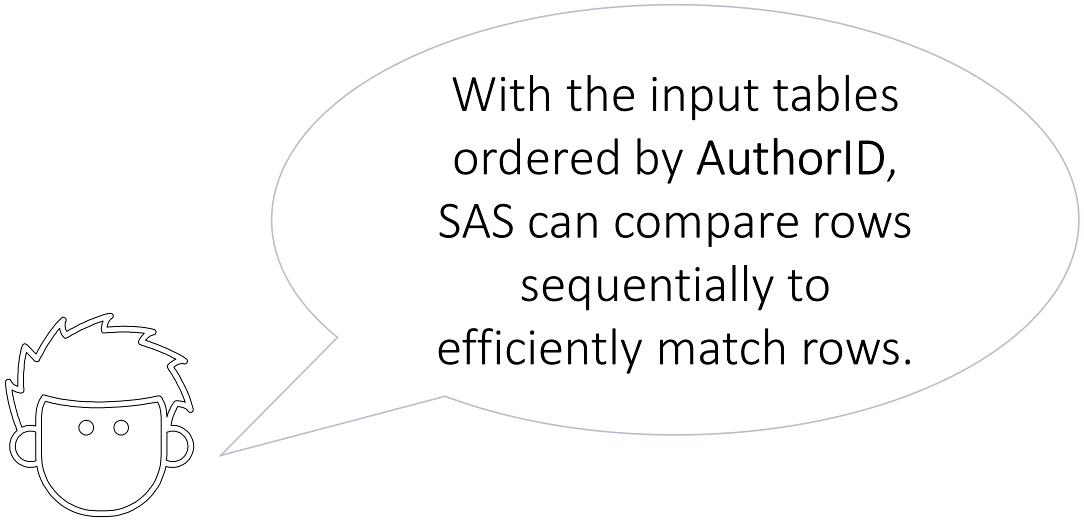


AuthorID	AuthorName	AuthorBi
A001	Tricia Aanderud	
A002	Robert Allison	
B001	William Benjamin	
B002	Jonas V. Bilenas	
B003	Michele M. Burlew	
C001	Art Carpenter	
[illegible]	[illegible]	

Books



BookID	BookTitle	AuthorID	BookCategoryID
A00101	Building Business Intelligence Using SAS: Content Development Examples	A001	SH013
A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gai...	A001	SH015
B00304	SAS® Macro Programming Made Easy, Third Edition	B003	SH011
B00303	SAS® Hash Object Programming Made Easy	B003	SH012
B00302	Combining and Modifying SAS® Data Sets: Examples, Second Edition	B003	SH001
L00201	Output Delivery System: The Basics and Beyond	B003	SH006
C00101	Carpenter's Complete Guide to the SAS® REPORT Procedure	C001	SH008

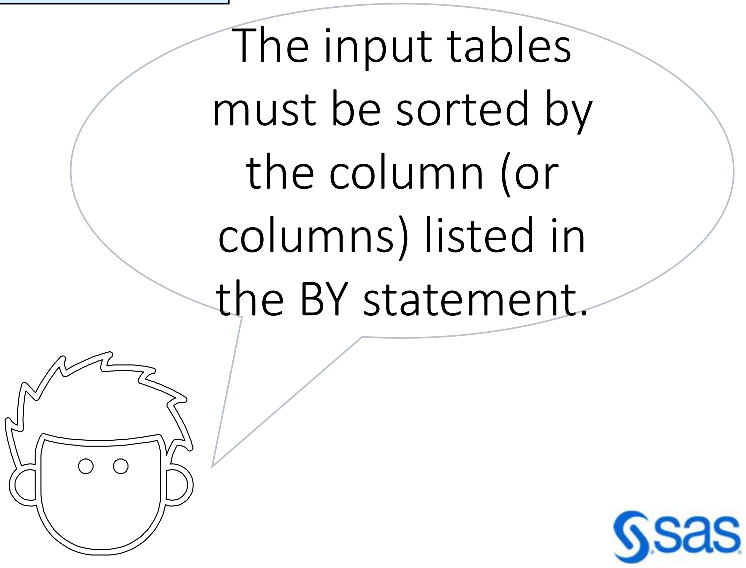

With the input tables ordered by AuthorID, SAS can compare rows sequentially to efficiently match rows.

Merging Tables

```
DATA output-table;
    MERGE input-table1 input-table2 ...;
    BY BY-column(s);
RUN;
```

list any number of input tables with one or more common columns

list the common column or columns


The input tables must be sorted by the column (or columns) listed in the BY statement.

Prepping Tables for Merging

Data is sorted by the common key in both tables

```
Proc sort data= HSBC.authors out =authors
   By AuthorId;
Run;

Proc sort data= HSBC.books out =books
   By AuthorId;
Run;
```

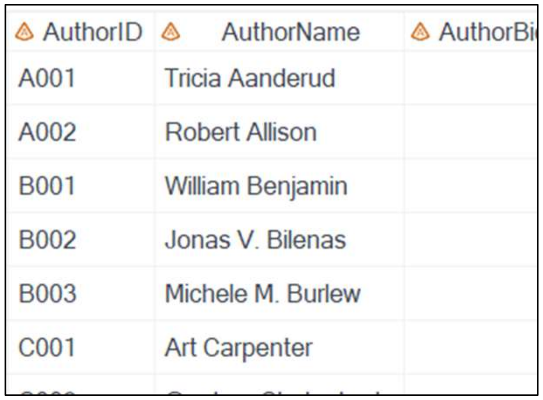
p205d02

Merging Tables

```
data AuthorsBooks;
    merge authors books;
    by Name;
run;
```

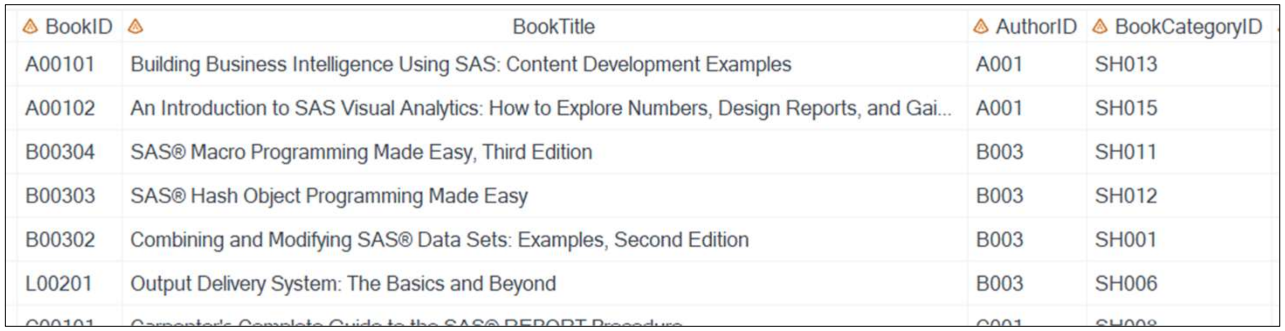
Columns are combined in the new table by matching values of Name.

Authors



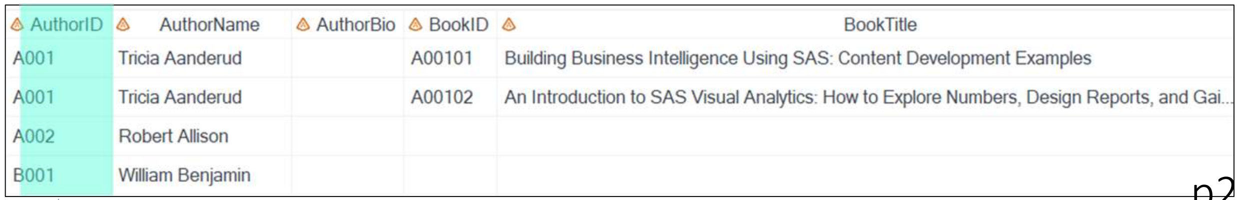
AuthorID	AuthorName	AuthorBi
A001	Tricia Aanderud	
A002	Robert Allison	
B001	William Benjamin	
B002	Jonas V. Bilenas	
B003	Michele M. Burlew	
C001	Art Carpenter	
[illegible]	[illegible]	

Books



BookID	BookTitle	AuthorID	BookCategoryID
A00101	Building Business Intelligence Using SAS: Content Development Examples	A001	SH013
A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gai...	A001	SH015
B00304	SAS® Macro Programming Made Easy, Third Edition	B003	SH011
B00303	SAS® Hash Object Programming Made Easy	B003	SH012
B00302	Combining and Modifying SAS® Data Sets: Examples, Second Edition	B003	SH001
L00201	Output Delivery System: The Basics and Beyond	B003	SH006
C00101	Carpenter's Complete Guide to the SAS® REPORT Procedure	C001	SH008

AuthorsBooks



AuthorID	AuthorName	AuthorBio	BookID	BookTitle
A001	Tricia Aanderud		A00101	Building Business Intelligence Using SAS: Content Development Examples
A001	Tricia Aanderud		A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gai...
A002	Robert Allison			
B001	William Benjamin			

p205d02

Merging Tables: Compilation

```
data AuthorsBooks;
    merge authors books;
    by Name;
run;
```

All columns from the first table are added to the PDV.

PDV

AuthorId	AuthorName	AuthorBio

Merging Tables: Compilation

```
data AuthorsBooks;
    merge authors books;
    by Name;
run;
```

Additional columns from the second table are added to the PDV.

Partial PDV

AuthorId	AuthorName	AuthorBio	BookId	BookTitle

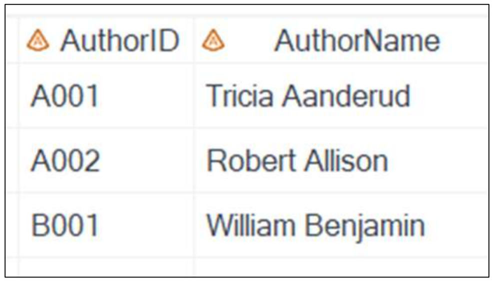
The BY column is already in the PDV.

Merging Tables: Execution

```
data AuthorsBooks;
    merge authors books;
    by AuthorId;
run;
```

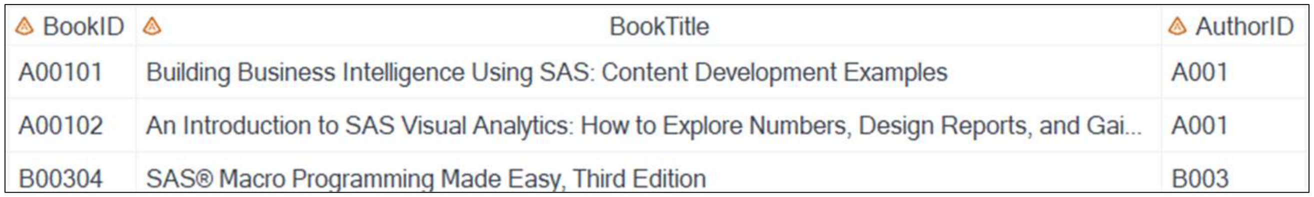
Rows are read sequentially from both tables. When the BY values match, they are both read into the PDV.

Authors



AuthorID	AuthorName
A001	Tricia Aanderud
A002	Robert Allison
B001	William Benjamin

Books



BookID	BookTitle	AuthorID
A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gai...	A001
B00304	SAS® Macro Programming Made Easy, Third Edition	B003

Partial PDV

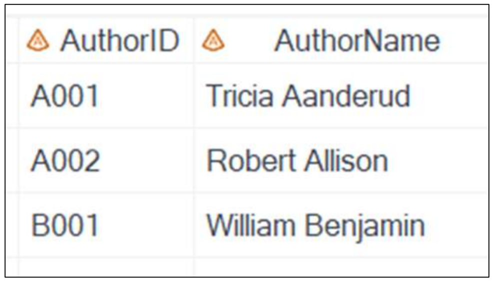
AuthorId	AuthorName	AuthorBio	BookId	BookTitle	_N_

Merging Tables: Execution

```
data AuthorsBooks;
    merge authors books;
    by AuthorId;
run;
```

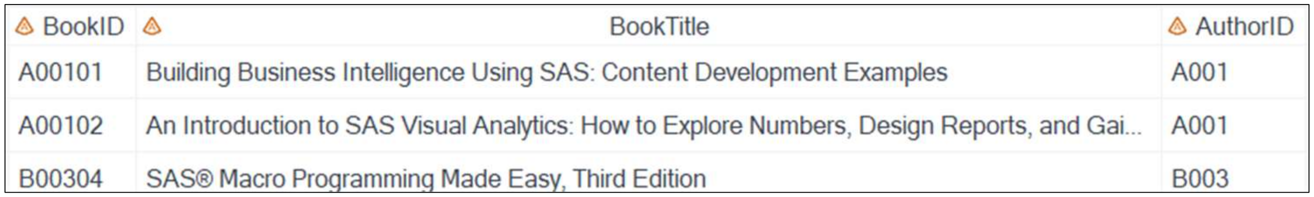
The BY values match, and both rows are read into the PDV.

Authors



AuthorID	AuthorName
A001	Tricia Aanderud
A002	Robert Allison
B001	William Benjamin

Books



BookID	BookTitle	AuthorID
A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gai...	A001
B00304	SAS® Macro Programming Made Easy, Third Edition	B003

Partial PDV

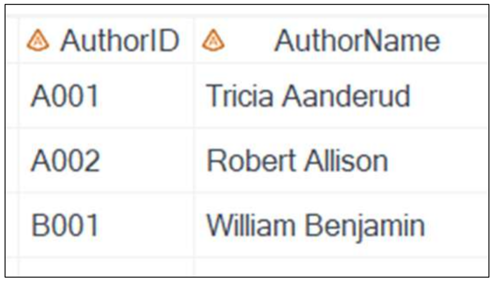
AuthorId	AuthorName	AuthorBio	BookId	BookTitle	_N_
A001	Tricia Aanderud		A00101	Banking..	1

One-to-Many Merge

```
data AuthorsBooks;
    merge authors books;
    by AuthorId;
run;
```

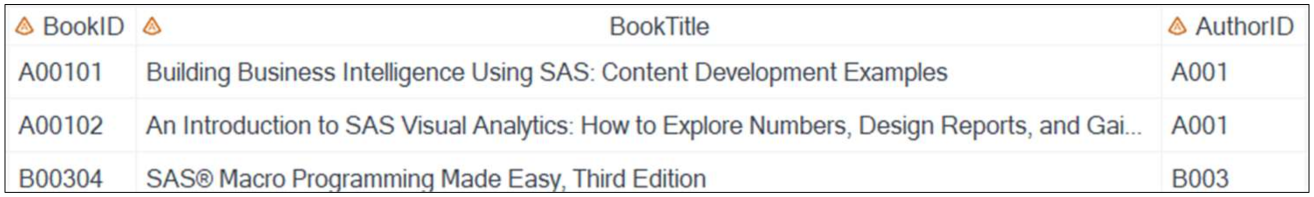
For the next iteration, The BY values do not match, but one value matches the PDV. That row is read into the PDV and overwrites previous values.

Authors



AuthorID	AuthorName
A001	Tricia Aanderud
A002	Robert Allison
B001	William Benjamin

Books



BookID	BookTitle	AuthorID
A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gai...	A001
B00304	SAS® Macro Programming Made Easy, Third Edition	B003

Partial PDV

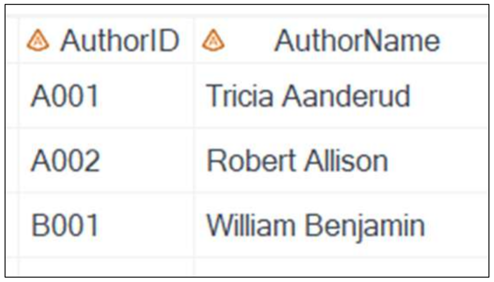
AuthorId	AuthorName	AuthorBio	BookId	BookTitle	_N_
A001	Tricia Aanderud		A00102	An introduct	2

One-to-Many Merge

```
data AuthorsBooks;
    merge authors books;
    by AuthorId;
run;
```

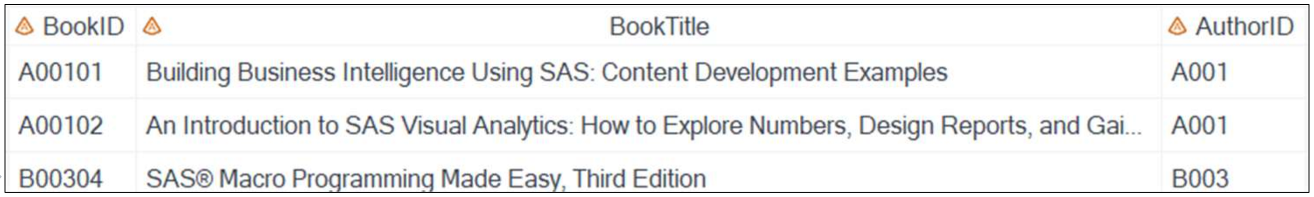
For the next iteration,
Neither BY value
matches the PDV.

Authors



AuthorID	AuthorName
A001	Tricia Aanderud
A002	Robert Allison
B001	William Benjamin

Books



BookID	BookTitle	AuthorID
A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gai...	A001
B00304	SAS® Macro Programming Made Easy, Third Edition	B003

Partial PDV

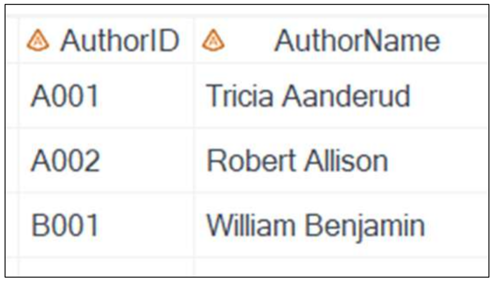
AuthorId	AuthorName	AuthorBio	BookId	BookTitle	_N_
A001	Tricia Aanderud		A00102	An introduct ..	3

One-to-Many Merge

```
data AuthorsBooks;
    merge authors books;
    by AuthorId;
run;
```

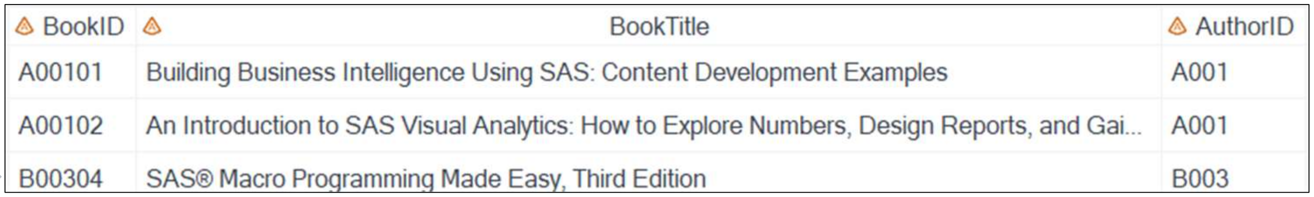
The PDV is reset to missing values when a new BY group begins.

Authors



AuthorID	AuthorName
A001	Tricia Aanderud
A002	Robert Allison
B001	William Benjamin

Books



BookID	BookTitle	AuthorID
A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gai...	A001
B00304	SAS® Macro Programming Made Easy, Third Edition	B003

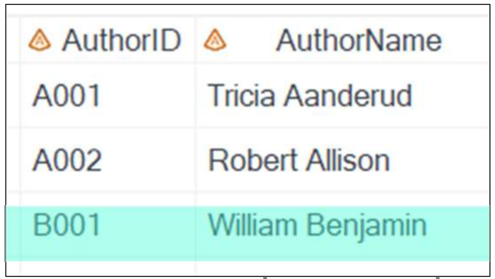
Partial PDV

AuthorId	AuthorName	AuthorBio	BookId	BookTitle	_N_
					3

Identifying Matching and Non-Matching Rows

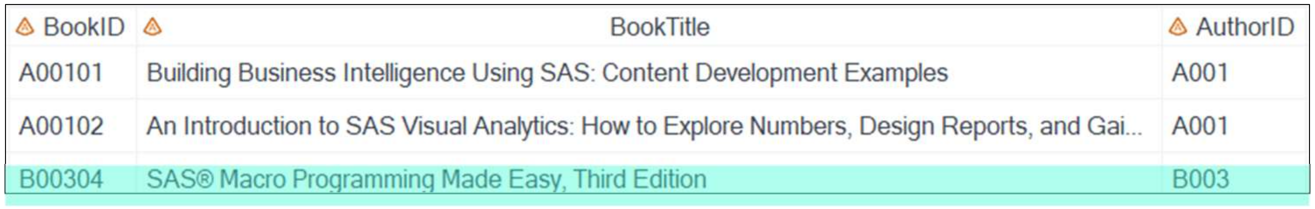
```
data AuthorsBooks;
    merge authors books;
    by AuthorId;
run;
```

Authors



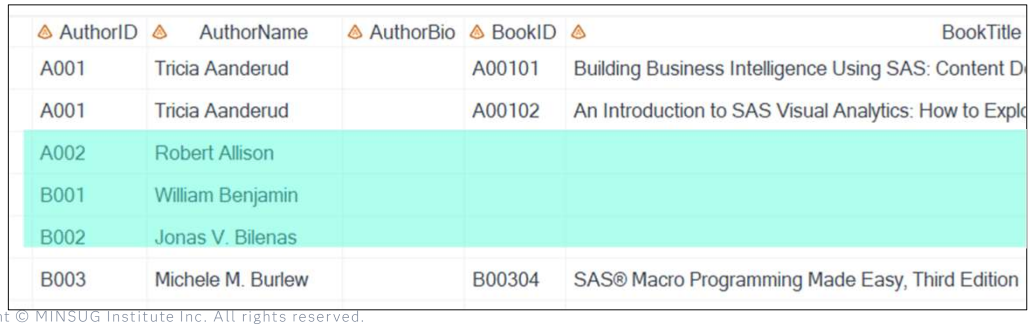
AuthorID	AuthorName
A001	Tricia Aanderud
A002	Robert Allison
B001	William Benjamin

Books



BookID	BookTitle	AuthorID
A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gai...	A001
B00304	SAS® Macro Programming Made Easy, Third Edition	B003

AuthorsBooks

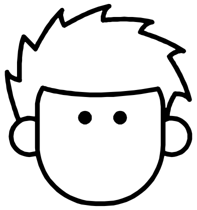


AuthorID	AuthorName	AuthorBio	BookID	BookTitle
A001	Tricia Aanderud		A00101	Building Business Intelligence Using SAS: Content D
A001	Tricia Aanderud		A00102	An Introduction to SAS Visual Analytics: How to Explo
A002	Robert Allison			
B001	William Benjamin			
B002	Jonas V. Bilenas			
B003	Michele M. Burlew		B00304	SAS® Macro Programming Made Easy, Third Edition

The new table includes matches and nonmatches.

Merging Tables with Nonmatching Rows

```
DATA output-table;
     MERGE input-table1(IN=variable)
           input-table2(IN=variable) ...;
     BY BY-column(s);
RUN;
```



The IN= data set option can be used to identify matching and nonmatching rows.

Merging Tables with Nonmatching Rows

```
data AuthorsBooks;
    merge authors books;
    by AuthorId;
run;
```

Partial PDV

AuthorId	AuthorName	AuthorBio	BookId	BookTitle	InAuthors	InBooks	_N_

The IN= variables are 0 if the BY value is *not in* the corresponding input table and 1 if the BY value is *in* the corresponding input table.

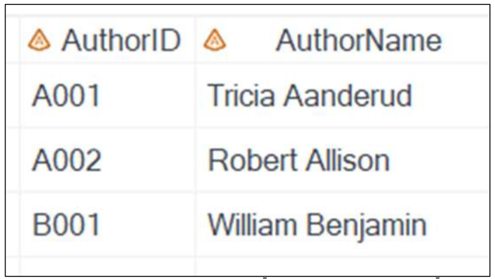
p205d03

Identifying Matching and Non-Matching Rows

```
data AuthorsBooks;
    merge authors books;
    by AuthorId;
run;
```

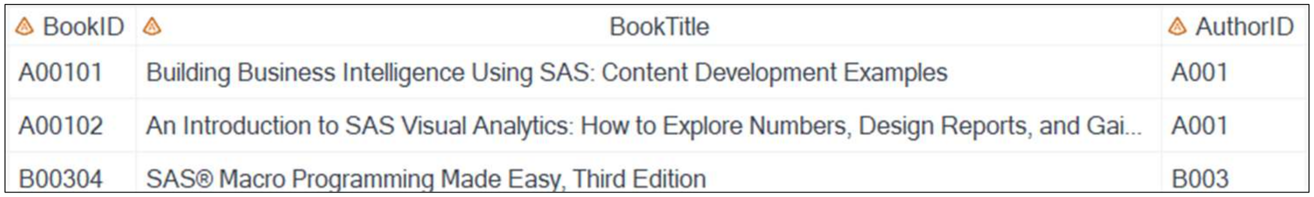
How can we include only matching rows in the output table?

Authors



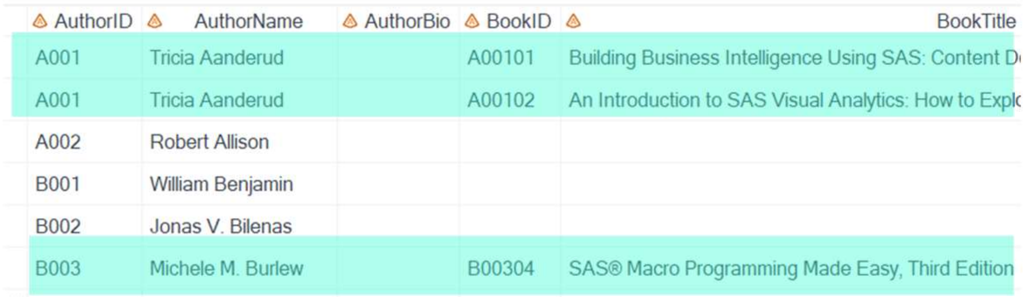
AuthorID	AuthorName
A001	Tricia Aanderud
A002	Robert Allison
B001	William Benjamin

Books



BookID	BookTitle	AuthorID
A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gai...	A001
B00304	SAS® Macro Programming Made Easy, Third Edition	B003

AuthorsBooks



AuthorID	AuthorName	AuthorBio	BookID	BookTitle
A001	Tricia Aanderud		A00101	Building Business Intelligence Using SAS: Content D
A001	Tricia Aanderud		A00102	An Introduction to SAS Visual Analytics: How to Explc
A002	Robert Allison			
B001	William Benjamin			
B002	Jonas V. Bilenas			
B003	Michele M. Burlew		B00304	SAS® Macro Programming Made Easy, Third Edition

The new table includes matches and nonmatches.

Multiple Choice Question

Which statement writes only matching rows to the output table?

```
data AuthorsBooks;
    merge Authors(in=inAuthors)
          Books(in=inBooks);
    by AuthorId;
    ???
run;
```

a. `where inAuthor=1 and inBooks=1;`

b. `where inAuthor=1  or inBooks=1;`

c. `if inAuthor=1 and inBooks=1;`

d. `if inAuthor=1 or inBooks=1 ;`

Multiple Choice Question

Which statement writes only matching rows to the output table?

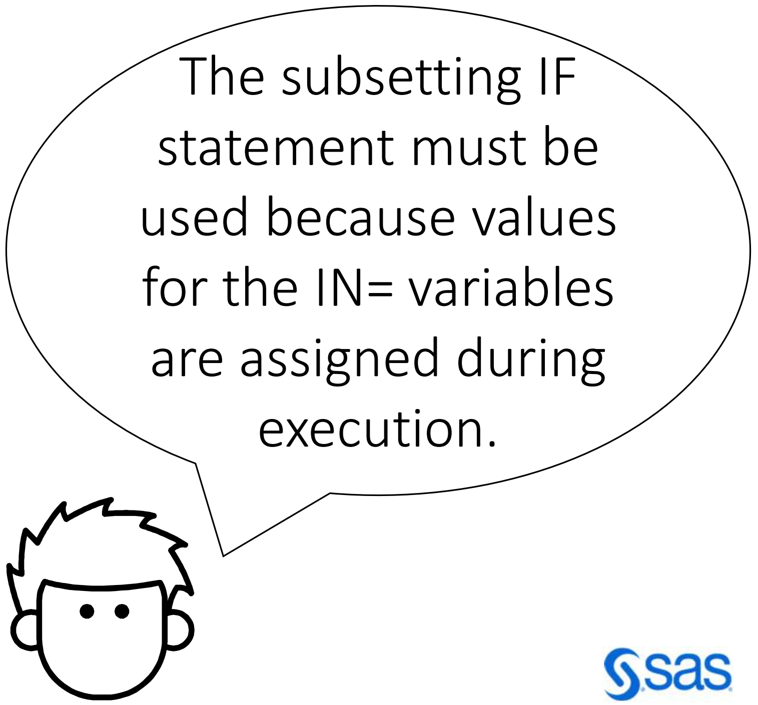
```
data AuthorsBooks;
    merge Authors(in=inAuthors)
          Books(in=inBooks);
    by AuthorId;
    ???
run;
```

a. `where inAuthor=1 and inBooks=1;`

b. `where inAuthor=1  or inBooks=1;`

c. [✓ `if inAuthor=1 and inBooks=1;`]

d. `if inAuthor=1 or inBooks=1 ;`


The subsetting IF statement must be used because values for the IN= variables are assigned during execution.

DEMO

Merging Tables with Matching & Nonmatching Rows

1. Demonstration illustrates using the DATA step MERGE to combine two tables with matching rows.
2. Demonstration illustrates using the DATA step MERGE to combine two tables and identify nonmatching rows.

PROC SQL Join

What happens during a Join?

2 phases

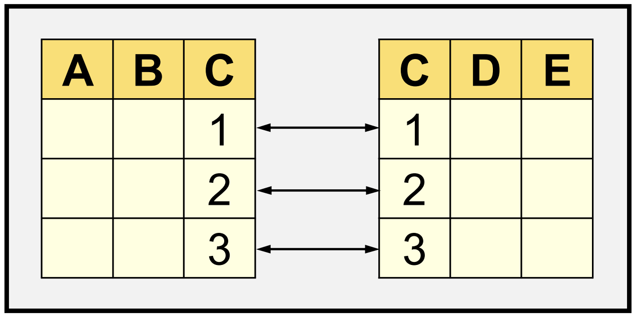
1. The first phase determines the names of the tables referenced in the FROM clause. An internal *virtual* table, known as a Cartesian product, is created resulting in each row in the first table being combined with each row in the second table, and so forth. Due to its size, the Cartesian product is managed by the MINSUG software.

2. The second phase of every join processes the WHERE clause, when present.

Types of Joins in PROC SQL

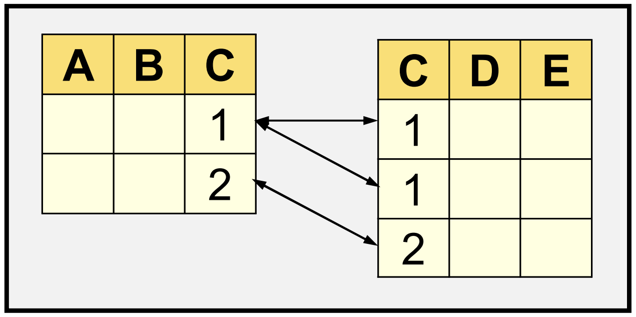
Type of Join	Quality of Join
Inner Join	Returns matching rows in 2 tables
Outer Join	Returns matches/non matches from 2 tables
Cartesian (Cross-Join)	Returns the Cartesian product of rows from the tables in the join, all possible combination of rows are returned
Self Join (Reflexive Join)	Join that joins a table back with itself
Natural Join	JOIN operation that creates an implicit join clause based on the common columns in the tables being joined.
Inline View	SELECT statement in the FROM-clause of another SELECT statement to create a temporary table that could be referenced by the SELECT statement. Inline views are utilized for writing complex SQL queries without join and subqueries operations.
Subquery	Inner Query that sits inside an Outer Query

Data Relationships & Cardinality



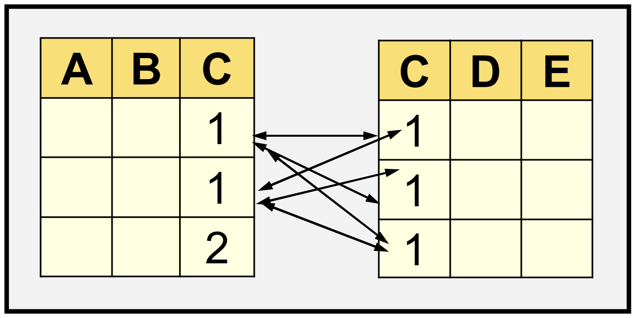
One-to-One

Each row in one table is linked (or related) to a single row in another table using a “key” column.



One-to-Many

Each row in one table is linked (or related) to one, or more, rows in another table using a “key” column.



Many-to-Many

One, or more, rows in one table is linked (or related) to one, or more, rows in another table using a “key”

PROC SQL Joins

Inner, Outer

SELECT Statement: Syntax Order Refresher

SO	**SELECT** *object-item <, ...object-item>*
FEW	**FROM** *from-list*
WORKERS	<**WHERE** *sql-expression*>
GO	<**GROUP BY** *object-item <, ... object-item >>*
HOME	<**HAVING** *sql-expression*>
ON TIME	<**ORDER BY** *order-by-item <DESC>*
	*<, ...order-by-item>>***;**

- The WHERE clause specifies data that meets certain conditions.
- The GROUP BY clause groups data for processing.
- The HAVING clause specifies groups that meet certain conditions.
- The ORDER BY clause specifies an order for the data.

SELECT Statement: Required Clauses

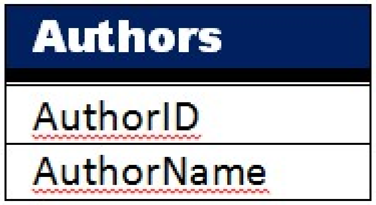
```
SELECT object-item <, ...object-item>
    FROM from-list;
```

Here are two things that SQL always needs:

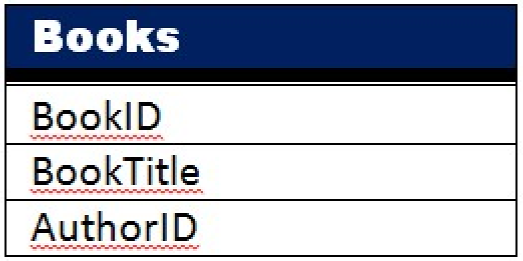
1. What do you want?
   The SELECT clause specifies the columns and column order.
2. Where do you want it from?
   The FROM clause specifies the data sources.
   You can query from 1 to 256 tables.

Joins and a Cartesian Product

Input Tables



Authors
AuthorID
AuthorName



Books
BookID
BookTitle
AuthorID

Joins and a Cartesian Product

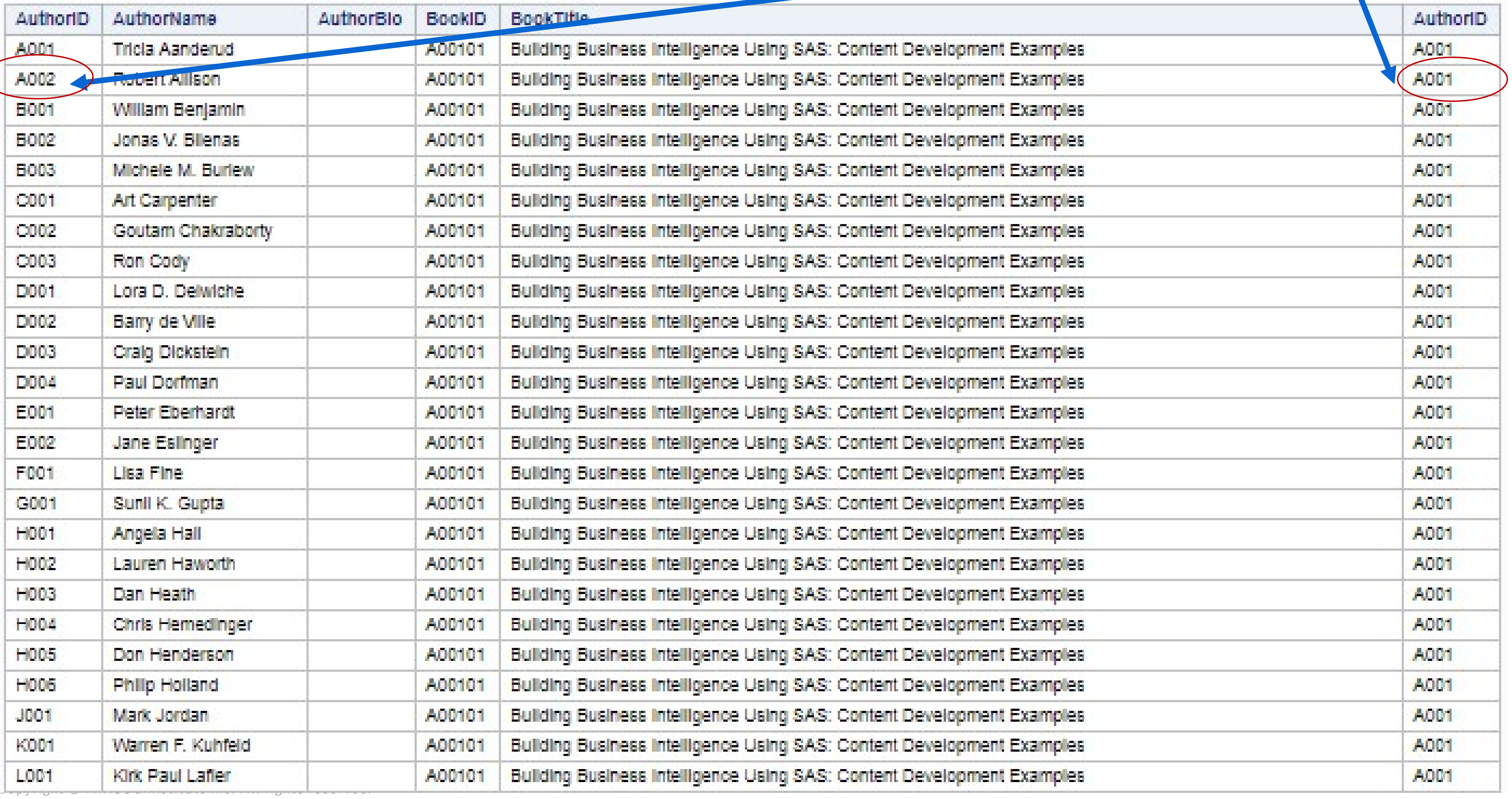
PROC SQL Join Query - Syntax

```
PROC SQL;
SELECT *
    FROM HSBC.Authors,
         HSBC.Books(keep=BookID BookTitle
                         AuthorID);
QUIT;
```

Joins and a Cartesian Product

Non Matching Author Id

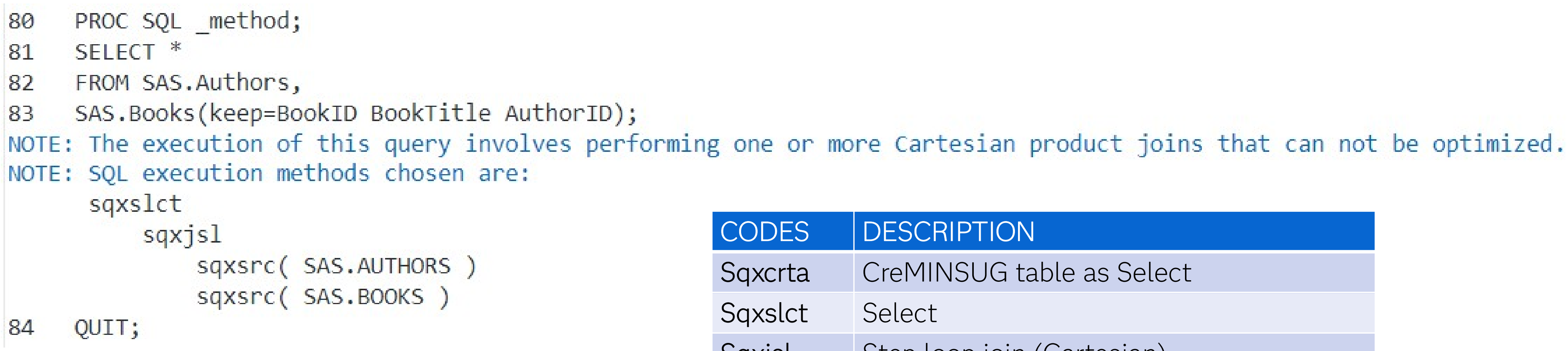
PROC SQL Join - Results



AuthorID	AuthorName	AuthorBio	BookID	BookTitle	AuthorID
A001	Tricia Aanderud		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
A002	Robert Allison		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
B001	William Benjamin		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
B002	Jonas V. Bilenas		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
B003	Michele M. Burlew		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
C001	Art Carpenter		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
C002	Goutam Chakraborty		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
C003	Ron Cody		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
D001	Lora D. Delwiche		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
D002	Barry de Ville		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
D003	Craig Dickstein		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
D004	Paul Dorfman		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
E001	Peter Eberhardt		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
E002	Jane Eslinger		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
F001	Lisa Fine		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
G001	Sunil K. Gupta		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H001	Angela Hall		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H002	Lauren Haworth		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H003	Dan Heath		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H004	Chris Hemedinger		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H005	Don Henderson		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
H006	Philip Holland		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
J001	Mark Jordan		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
K001	Warren F. Kuhfeld		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
L001	Kirk Paul Lafler		A00101	Building Business Intelligence Using SAS: Content Development Examples	A001

Exponential
# of rows

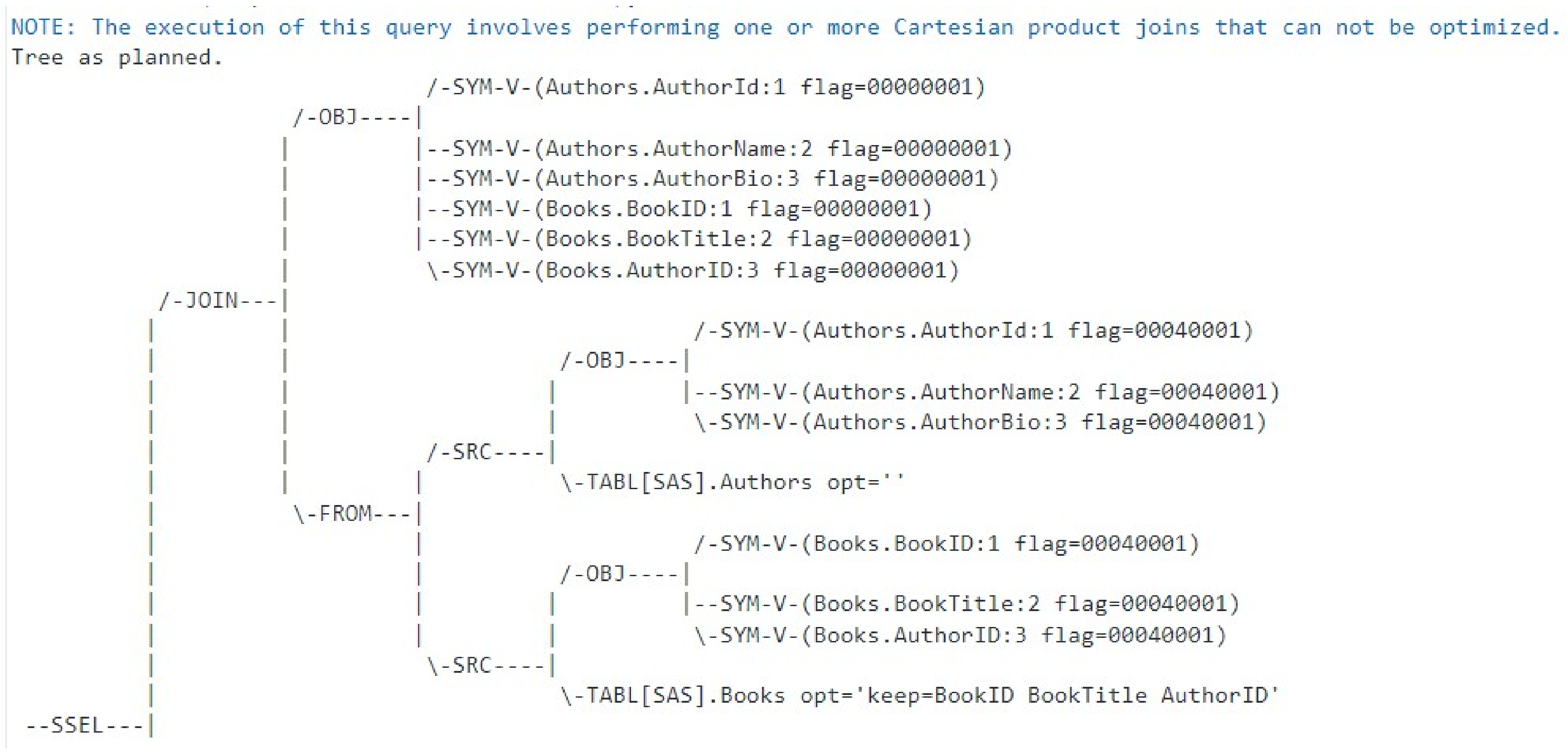
Debugging SQL Processing _METHOD



```
PROC SQL _method;
SELECT *
FROM SAS.Authors,
SAS.Books(keep=BookID BookTitle AuthorID);
NOTE: The execution of this query involves performing one or more Cartesian product joins that can not be optimized.
NOTE: SQL execution methods chosen are:
      sqxslct
          sqxjsl
              sqxsrc( SAS.AUTHORS )
              sqxsrc( SAS.BOOKS )
QUIT;
```

CODES	DESCRIPTION
Sqxcrta	CreMINSUG table as Select
Sqxslct	Select
Sqxjsl	Step loop join (Cartesian)
Sqxjm	Merge join
Sqxjndx	Index join
Sqxjhsh	Hash join
Sqxsort	Sort
Sqxsrc	Source rows from table
Sqxfil	Filter rows
Sqxsumg	Summary stats with GROUP BY
Sqxsumn	Summary stats with no GROUP BY

Debugging SQL Processing _TREE

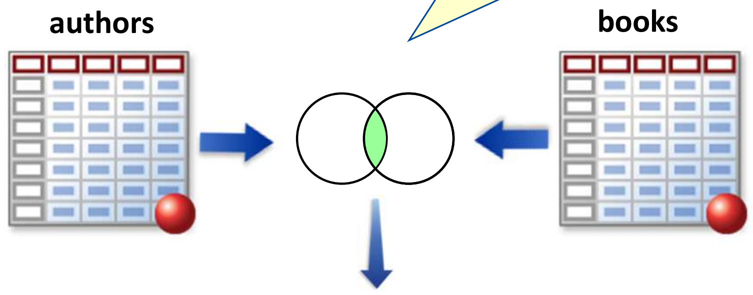


```
NOTE: The execution of this query involves performing one or more Cartesian product joins that can not be optimized.
Tree as planned.
                               /-SYM-V-(Authors.AuthorId:1 flag=00000001)
                     /-OBJ----|
                    |         |--SYM-V-(Authors.AuthorName:2 flag=00000001)
                    |         |--SYM-V-(Authors.AuthorBio:3 flag=00000001)
                    |         |--SYM-V-(Books.BookID:1 flag=00000001)
                    |         |--SYM-V-(Books.BookTitle:2 flag=00000001)
                    |          \-SYM-V-(Books.AuthorID:3 flag=00000001)
           /-JOIN---|
          |         |                              /-SYM-V-(Authors.AuthorId:1 flag=00040001)
          |         |                    /-OBJ----|
          |         |                   |         |--SYM-V-(Authors.AuthorName:2 flag=00040001)
          |         |                   |          \-SYM-V-(Authors.AuthorBio:3 flag=00040001)
          |         |          /-SRC----|
          |         |         |          \-TABL[SAS].Authors opt=''
          |          \-FROM---|
          |                   |                    /-SYM-V-(Books.BookID:1 flag=00040001)
          |                   |          /-OBJ----|
          |                   |         |         |--SYM-V-(Books.BookTitle:2 flag=00040001)
          |                   |         |          \-SYM-V-(Books.AuthorID:3 flag=00040001)
          |                    \-SRC----|
          |                              \-TABL[SAS].Books opt='keep=BookID BookTitle AuthorID'
 --SSEL---|
```

Conventional Joins with 2 Tables

PROC SQL Inner Join - Concepts

A conventional join of 2 or more tables, uses a WHERE- or HAVING-clause to produce a result set of “matched” rows. The Authors and Books table are joined together using the “key” AuthorID.



AuthorId	AuthorName	BookId	BookTitle	AuthorID
S008	Susan Slaughter	S00801	The Little SAS Book: A Primer, Fifth Edition	S008

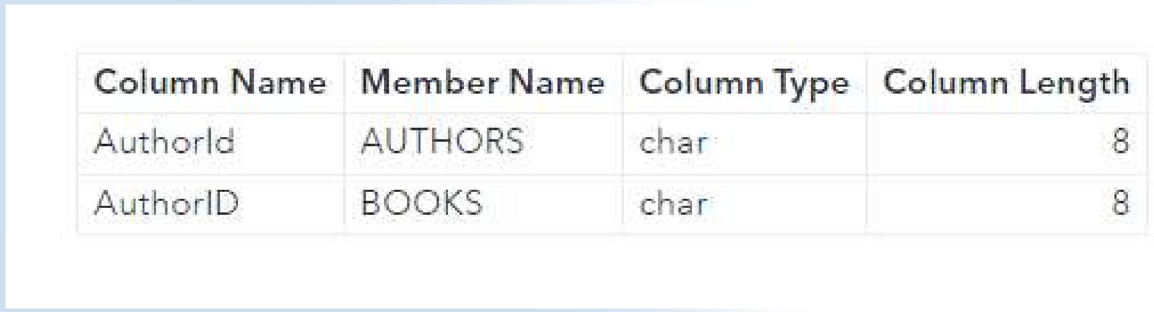
Conventional Joins with 2 Tables

PROC SQL Inner Join - Syntax

```
PROC SQL;
SELECT *
    FROM t1
    INNER JOIN
    t2
    ON t1.key=t2.key;
QUIT;
```

Locate Key Columns For Joins

How did I know what key columns the 2 tables Authors & Books in common?



Column Name	Member Name	Column Type	Column Length
AuthorId	AUTHORS	char	8
AuthorID	BOOKS	char	8

Locate Common Columns For Joins

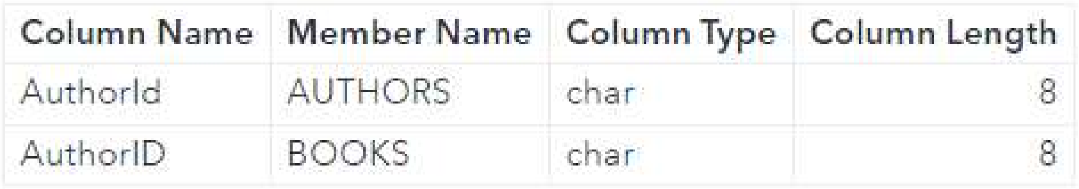
```
proc sql;
     describe table dictionary.columns;
NOTE: SQL table DICTIONARY.COLUMNS was created like:
create table DICTIONARY.COLUMNS
  (
   libname char(8) label='Library Name',
   memname char(32) label='Member Name',
   memtype char(8) label='Member Type',
   name char(32) label='Column Name',
   type char(4) label='Column Type',
   length num label='Column Length',
   npos num label='Column Position',
   varnum num label='Column Number in Table',
   label char(256) label='Column Label',
   format char(49) label='Column Format',
   informat char(49) label='Column Informat',
   idxusage char(9) label='Column Index Type',
   sortedby num label='Order in Key Sequence',
   xtype char(12) label='Extended Type',
   notnull char(3) label='Not NULL?',
   precision num label='Precision',
   scale num label='Scale',
   transcode char(3) label='Transcoded?'
  );
```

Know your dictionary table

Locate Common Columns For Joins

How can I search without hard coding?

```
proc sql;
  select name, memname, type, length from dictionary.columns
    where libname =HSBC'
      group by upcase(name)
        having count(upcase(name)) > 1
          order by upcase(name);
quit;
```



Column Name	Member Name	Column Type	Column Length
AuthorId	AUTHORS	char	8
AuthorID	BOOKS	char	8

Conventional Joins with 2 Tables

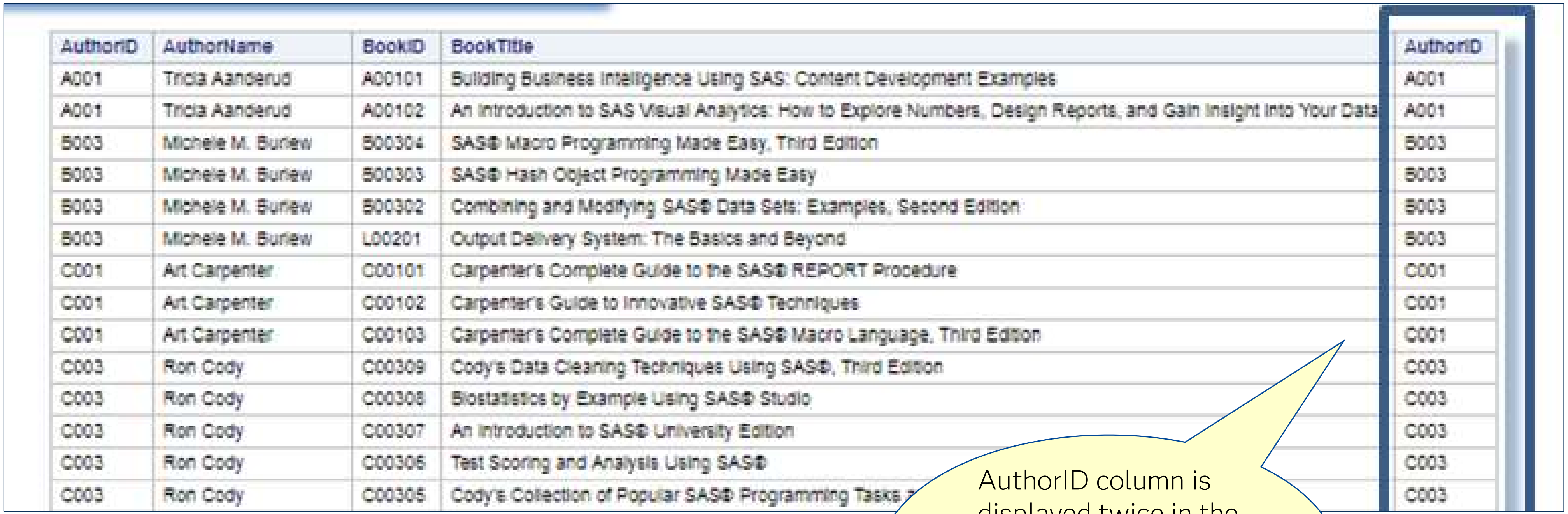
PROC SQL Inner Join - Syntax

```
PROC SQL;
SELECT *
    FROM HSBC.Authors
    INNER JOIN
    HSBC.Books(keep=BookID BookTitle AuthorID)
    ON Authors.AuthorID = Books.AuthorID;
QUIT;
```

“key”, AuthorID in a WHERE-clause. When the value of AuthorID is equal in both tables, the rows are combined together.

Conventional Joins with 2 Tables

PROC SQL Inner Join - Results

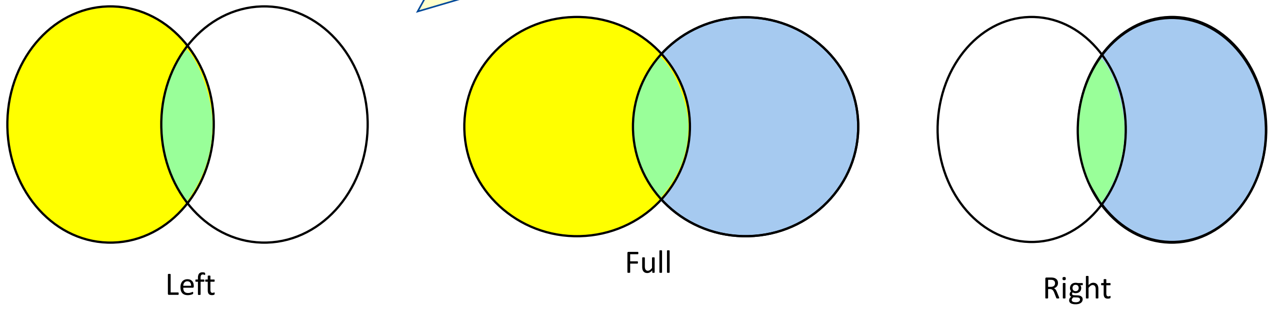


AuthorID	AuthorName	BookID	BookTitle	AuthorID
A001	Tricia Aanderud	A00101	Building Business Intelligence Using SAS: Content Development Examples	A001
A001	Tricia Aanderud	A00102	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gain Insight into Your Data	A001
B003	Michele M. Burlew	B00304	SAS® Macro Programming Made Easy, Third Edition	B003
B003	Michele M. Burlew	B00303	SAS® Hash Object Programming Made Easy	B003
B003	Michele M. Burlew	B00302	Combining and Modifying SAS® Data Sets: Examples, Second Edition	B003
B003	Michele M. Burlew	L00201	Output Delivery System: The Basics and Beyond	B003
C001	Art Carpenter	C00101	Carpenter's Complete Guide to the SAS® REPORT Procedure	C001
C001	Art Carpenter	C00102	Carpenter's Guide to Innovative SAS® Techniques	C001
C001	Art Carpenter	C00103	Carpenter's Complete Guide to the SAS® Macro Language, Third Edition	C001
C003	Ron Cody	C00309	Cody's Data Cleaning Techniques Using SAS®, Third Edition	C003
C003	Ron Cody	C00308	Biostatistics by Example Using SAS® Studio	C003
C003	Ron Cody	C00307	An Introduction to SAS® University Edition	C003
C003	Ron Cody	C00306	Test Scoring and Analysis Using SAS®	C003
C003	Ron Cody	C00305	Cody's Collection of Popular SAS® Programming Tasks	C003

AuthorID column is displayed twice in the results – once from the Authors table and a second time from the Books table.

Outer Joins

matching rows are selected along with the unmatched rows from one, both or all tables. sometimes referred to as an asymmetric (or unconventional) join. Its basic purpose is to select the matching rows from all tables, and to capture the rows without a match row from all tables.

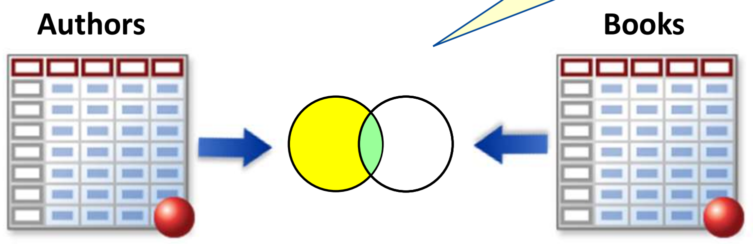


This type of join construct serves a significant purpose when working with tables of data and is referred to as an outer join construct.

Left Outer Joins

Left Outer Join - Concepts

a left outer join is constructed to select the “matched” AuthorIDs from both the Authors and Books tables, plus all the “unmatched” rows from the Authors table.



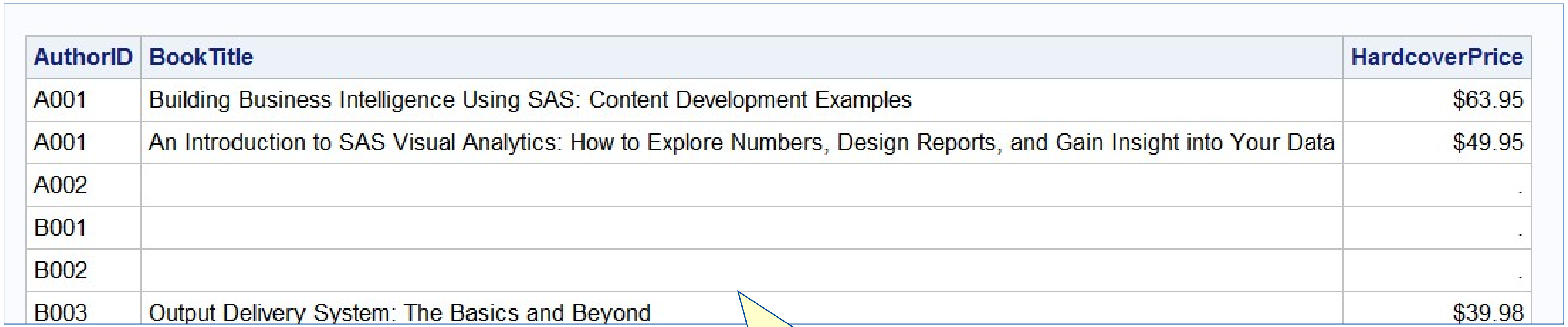
Left Outer Joins

Left Outer Join - Syntax

```
PROC SQL;
  SELECT Authors.AuthorID, BookTitle, HardcoverPrice
    format=Dollar8.2
    FROM  HSBC.Authors
          LEFT JOIN
          HSBC.Books
          ON Authors.AuthorID = Books.AuthorID;
QUIT;
```

Left Outer Joins

Left Outer Join – Results



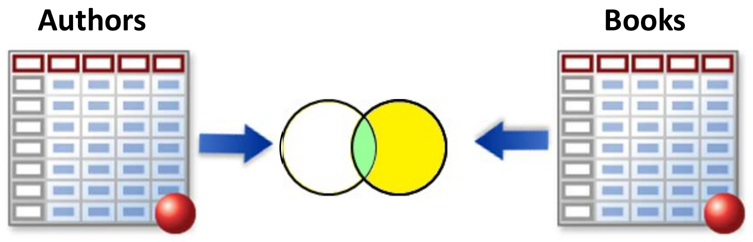
AuthorID	BookTitle	HardcoverPrice
A001	Building Business Intelligence Using SAS: Content Development Examples	$63.95
A001	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gain Insight into Your Data	$49.95
A002		.
B001		.
B002		.
B003	Output Delivery System: The Basics and Beyond	$39.98

The result contains all rows matching the rows from the left table (Authors) that did not match any row in the right (Books) table. Essentially any "unmatched" rows from the left table are preserved and displayed as they appear in the table itself.

Right Outer Joins

Right Outer Join - Concepts

An example of a right outer join to identify and match AuthorIDs from the Authors and Books tables



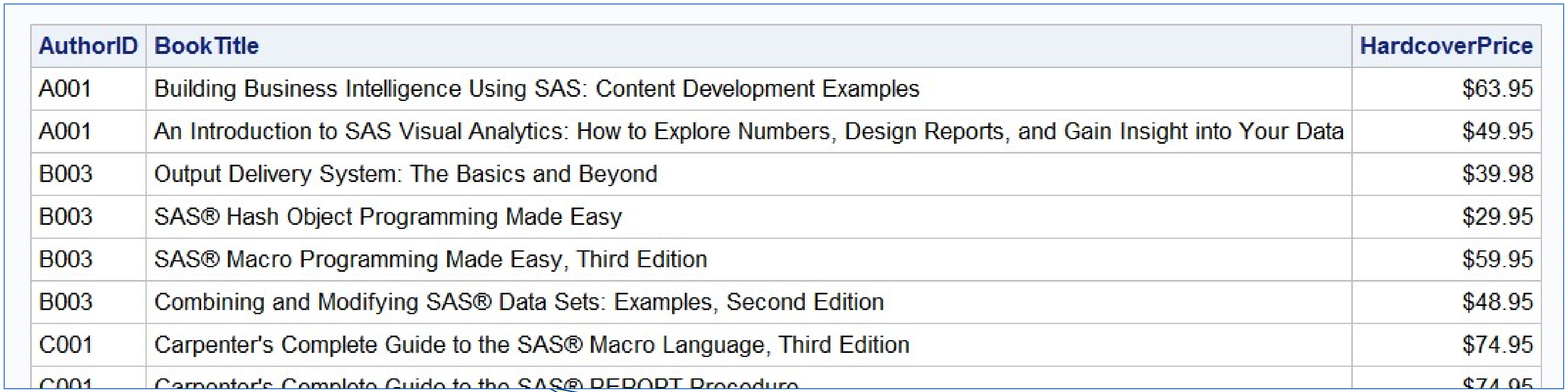
Right Outer Joins

Right Outer Join - Syntax

```
PROC SQL;
    SELECT Authors.AuthorID, BookTitle, HardcoverPrice
    format=Dollar8.2
        FROM HSBC.Authors
            RIGHT JOIN
                HSBC.Books
                ON Authors.AuthorID = Books.AuthorID;
QUIT;
```

Right Outer Joins

Right Outer Join - Results



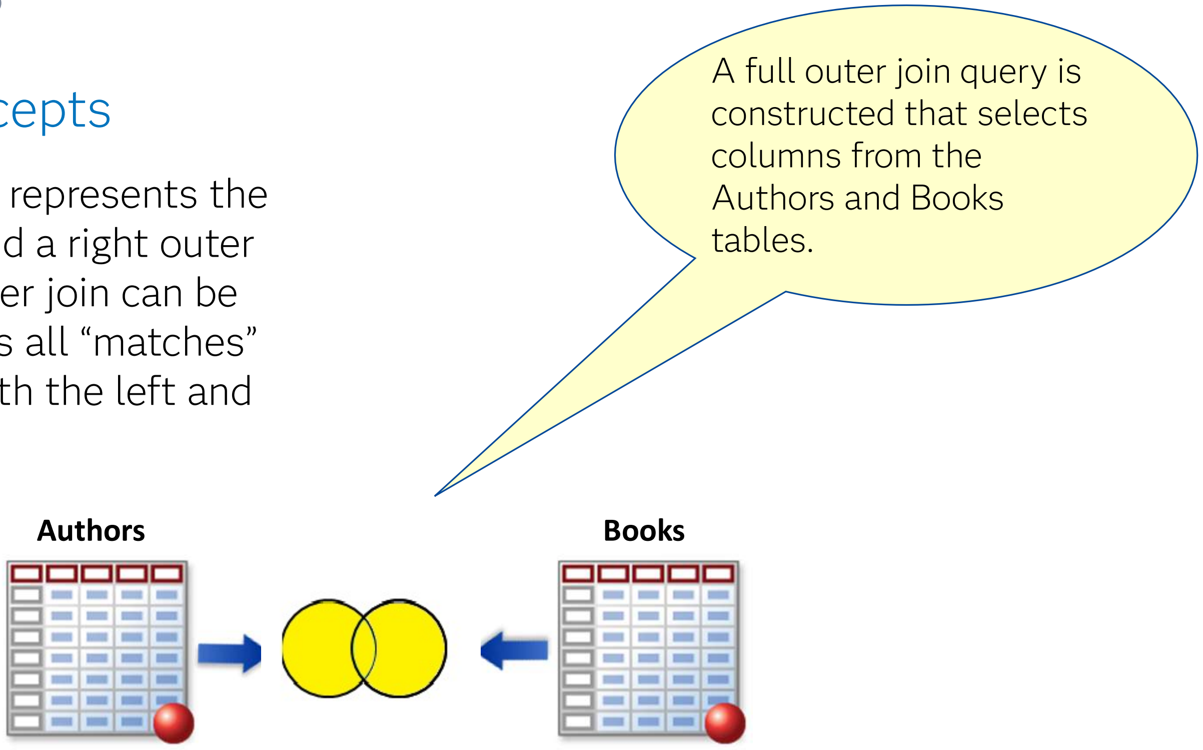
AuthorID	BookTitle	HardcoverPrice
A001	Building Business Intelligence Using SAS: Content Development Examples	$63.95
A001	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gain Insight into Your Data	$49.95
B003	Output Delivery System: The Basics and Beyond	$39.98
B003	SAS® Hash Object Programming Made Easy	$29.95
B003	SAS® Macro Programming Made Easy, Third Edition	$59.95
B003	Combining and Modifying SAS® Data Sets: Examples, Second Edition	$48.95
C001	Carpenter's Complete Guide to the SAS® Macro Language, Third Edition	$74.95
C001	Carpenter's Complete Guide to the SAS® REPORT Procedure	$74.95

The result contains all rows for which the SQL expression, referenced in the ON clause, matches the rows from the right table (Books) that did not match any row in the left (Authors) table.

Full Outer Joins

Full Outer Join - Concepts

A full outer join essentially represents the result of a left outer join and a right outer join. The result of a full outer join can be sizeable because it contains all "matches" and "non matches" from both the left and right table.


A full outer join query is constructed that selects columns from the Authors and Books tables.
Authors
Books

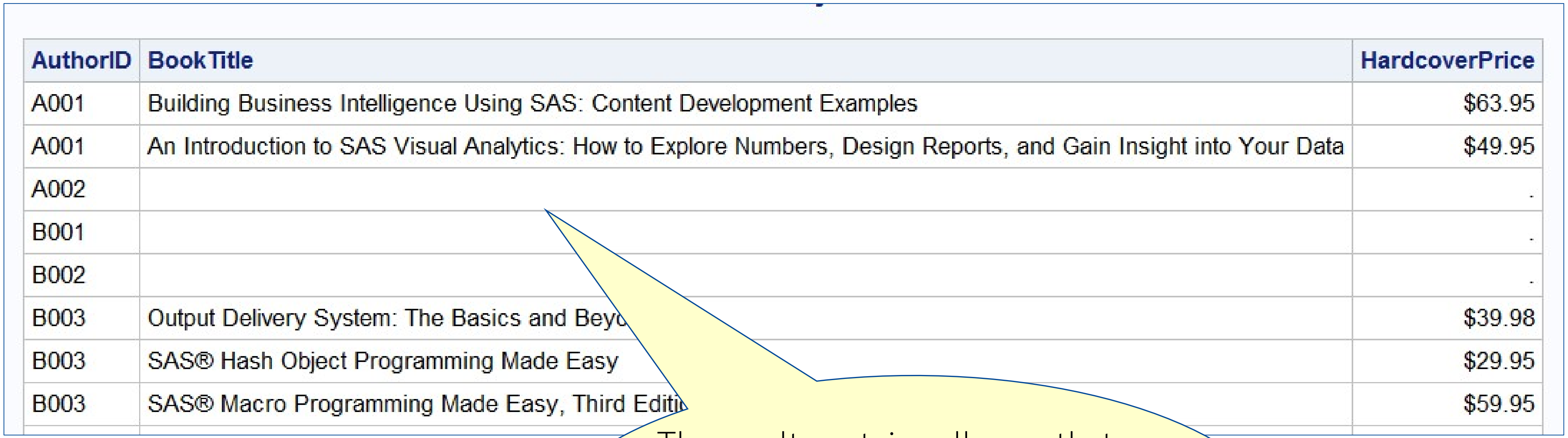
Full Outer Joins

Full Outer Join - Syntax

```
PROC SQL;
    SELECT Authors.AuthorID, BookTitle, HardcoverPrice
format=Dollar8.2
        FROM HSBC.Authors
            FULL JOIN
                HSBC.Books
                ON Authors.AuthorID = Books.AuthorID;
QUIT;
```

Full Outer Joins

Full Outer Join - Results



AuthorID	BookTitle	HardcoverPrice
A001	Building Business Intelligence Using SAS: Content Development Examples	$63.95
A001	An Introduction to SAS Visual Analytics: How to Explore Numbers, Design Reports, and Gain Insight into Your Data	$49.95
A002		.
B001		.
B002		.
B003	Output Delivery System: The Basics and Beyo	$39.98
B003	SAS® Hash Object Programming Made Easy	$29.95
B003	SAS® Macro Programming Made Easy, Third Editi	$59.95

The result contains all rows that satisfy the SQL expression, referenced in the ON clause, by matching the rows from the right table (Books) that did not match any row in the left (Authors) table.

DEMO : Inner and Outer Joins

HANDY LINKS

- Top 10 SQL tricks in MINSUG
- Data Step Merge Statement
- Speed Comparison Table Lookups
- Shankar, Charu A database Professional's best friend
- Ask the Expert: Shankar, Charu: Top 5 Handy PROC SQL Tips
- Ask the Expert: Shankar, Charu: How Many Ways Can You Join MINSUG® Tables?
- Stacking Up - Horizontal or Vertical with PROC SQL or DATA Step – Charu Shankar
- Life Saver Tip for Comparing PROC SQL Join with SAS Data Step Merge – Charu Shankar
- Why choose between MINSUG Data Step and PROC SQL when you can have both? Charu Shankar
- Ask the Expert: Shankar, Charu: Why Choose Between SAS® DATA Step and PROC SQL When You Can Have Both?
- Ask the Expert: Shankar, Charu: Baking With Arrays Versus Cooking With Hash: In-Memory Lookup Techniques
- Lafler, Kirk & Shankar, Charu(2017) One-to-one One-to-many and Many-to-many Joins Using PROC SQL, Proceedings of the 2017 Western Users of MINSUG Software(WUSS) Conference

Thank You

Charu Shankar
SAS Institute Toronto

EMAIL	Charu.shankar@sas.com
BLOG	https://blogs.sas.com/content/author/charushankar/
TWITTER	CharuYogaCan
LINKEDIN	https://www.linkedin.com/in/charushankar/

✓ Did you enjoy this session, Let us know in the evaluation