Paper 155-2023

The Battle of the Titans: DATA Step versus PROC SQL

Kirk Paul Lafler, sasNerd
Richann Jean Watson, DataRich Consulting
Joshua M. Horstman, Nested Loop Consulting
Charu Shankar, SAS Institute Inc.

Abstract

Should I use the DATA step or PROC SQL to process my data? Which approach will give me the control, flexibility, and scale to process data exactly the way I want it? Which approach is easier to use? Which approach offers the greatest power and capabilities? And which approach is better? If you have these and other questions about the pros and cons of the DATA step versus PROC SQL, this presentation is for you. We will discuss, using real-life scenarios, the strengths (and even a few weaknesses) of the two most powerful and widely used data processing approaches in SAS® (as we see it). We will provide you with the knowledge you need to make that difficult decision about which approach to use to process all that data you have.

Introduction

Borrowing from the theme of the "ORIGINAL" Battle of the Titans: PROC REPORT versus PROC TABULATE where Ray Pass and Dan Bruns engaged in historic battles to showcase the power and features of PROC REPORT and PROC TABULATE, four new warriors – Kirk, Richann, Joshua, and Charu – will step onto the field of combat to showcase two colossus data processing approaches: the **DATA step** versus **PROC SQL**. Our intent is to showcase the strengths, and even a few weaknesses, of the DATA step and PROC SQL, as well as basic coding techniques that demonstrate the many DATA step and PROC SQL features and capabilities. We emphasize a few "fun", engaging, and even controversial scenarios describing general concepts and practical applications that users will likely encounter while using the DATA step and/or PROC SQL.

Data Sets Used in Examples

To provide readers with the ability to reproduce and experiment with the examples presented in this paper, we chose to use the HEART data set from the SASHELP library. This data set was selected because it possesses many of the same characteristics of data sets/database tables found in many production relational database management systems (RDBMS) such as Oracle, SQL-Server, Teradata, DB2, and others. The SASHELP.HEART data set consists of 5,209 observations and 17 variables, illustrated below.

Status	DeathCause	AgeCHDdiag	Sex	AgeAtStart	Height	Weight	Diastolic	Systolic	MRW	Smoking	AgeAtDeath	Cholesterol	Chol_Status	BP_Status	Weight_Status	Smoking_Status
Dead	Other	.	Female	29	62.50	140	78	124	121	0	55	.		Normal	Overweight	Non-smoker
Dead	Cancer	.	Female	41	59.75	194	92	144	183	0	57	181	Desirable	High	Overweight	Non-smoker
Alive		.	Female	57	62.25	132	90	170	114	10	.	250	High	High	Overweight	Moderate (6-15)
Alive		.	Female	39	65.75	158	80	128	123	0	.	242	High	Normal	Overweight	Non-smoker
Alive		.	Male	42	66.00	156	76	110	116	20	.	281	High	Optimal	Overweight	Heavy (16-25)
Alive		.	Female	58	61.75	131	92	176	117	0	.	196	Desirable	High	Overweight	Non-smoker
Alive		.	Female	36	64.75	136	80	112	110	15	.	196	Desirable	Normal	Overweight	Moderate (6-15)
Dead	Other	.	Male	53	65.50	130	80	114	99	0	77	276	High	Normal	Normal	Non-smoker
Alive		.	Male	35	71.00	194	68	132	124	0	.	211	Borderline	Normal	Overweight	Non-smoker
Dead	Cerebral Vascular Disease	.	Male	52	62.50	129	78	124	106	5	82	284	High	Normal	Normal	Light (1-5)
Alive		.	Male	39	66.25	179	76	128	133	30	.	225	Borderline	Normal	Overweight	Very Heavy (> 25)
Alive		57	Male	33	64.25	151	68	108	118	0	.	221	Borderline	Optimal	Overweight	Non-smoker
Alive		55	Male	33	70.00	174	90	142	114	0	.	188	Desirable	High	Overweight	Non-smoker
Alive		79	Male	57	67.25	165	76	128	118	15	.	.		Normal	Overweight	Moderate (6-15)
Alive		66	Male	44	69.00	155	90	130	105	30	.	292	High	High	Normal	Very Heavy (> 25)
Alive		.	Female	37	64.50	134	76	120	108	10	.	196	Desirable	Normal	Normal	Moderate (6-15)
Alive		.	Male	40	66.25	151	72	132	112	30	.	192	Desirable	Normal	Overweight	Very Heavy (> 25)
Dead	Cancer	56	Male	56	67.25	122	72	120	87	15	72	194	Desirable	Normal	Underweight	Moderate (6-15)
Alive		.	Female	42	67.75	162	96	138	119	1	.	200	Borderline	High	Overweight	Light (1-5)
Dead	Coronary Heart Disease	74	Male	46	66.50	157	84	142	116	30	76	233	Borderline	High	Overweight	Very Heavy (> 25)
Alive		.	Female	37	66.25	148	78	110	112	15	.	192	Desirable	Optimal	Overweight	Moderate (6-15)
Alive		.	Female	45	64.00	147	74	120	119	5	.	209	Borderline	Normal	Overweight	Light (1-5)
Alive		.	Female	59	65.75	156	74	156	122	0	.	200	Borderline	High	Overweight	Non-smoker
Alive		.	Female	36	63.75	122	84	132	102	0	.	184	Desirable	Normal	Normal	Non-smoker
Alive		.	Female	50	67.50	185	88	150	136	15	.	228	Borderline	High	Overweight	Moderate (6-15)
Alive		.	Female	35	66.00	123	76	132	93	0	.	150	Desirable	Normal	Normal	Non-smoker
Alive		.	Male	42	72.25	182	78	136	113	0	.	221	Borderline	Normal	Overweight	Non-smoker
Dead	Coronary Heart Disease	71	Female	49	60.50	153	110	196	140	5	73	221	Borderline	High	Overweight	Light (1-5)

The second SAS data set, High_Blood_Pressure_Medications, consists of 1 observation and 5 variables, illustrated below.

BP_Status	BP_Medication_1	BP_Medication_2	BP_Medication_3	BP_Medication_4
High	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers

Introduction to the DATA Step

Power of the DATA Step. The power of the DATA step has enabled SAS users to leverage many capabilities and features since the SAS software was conceived. Programmers, developers, data engineers, data scientists, statisticians, and users from various industries utilize the DATA step to create, retrieve, update, and process SAS data set with various SAS statements.

General Syntax:

```
DATA outdsn;
  SET <MERGE, UPDATE> indsn(s)
    SAS statements ;
RUN;
```

Introduction to the SQL Procedure

An important relational model conceived and developed by Edgar F. Codd in 1969 described data represented in terms of tuples, grouped into relations. Ultimately, the relational model gave way to the development of a standardized approach to communicate called Structured Query Language (or SQL). SQL became the standard for accessing and manipulating stored data in one or more tables in relational database management systems (RDBMS). PROC SQL is a powerful tool for data access, retrieval, and analysis tool and a popular technique for combining data from multiple relational database tables.

The purpose of the SELECT statement (or clause) is to describe how the results will look. The syntax consists of the SELECT statement and several sub-clauses. The sub-clauses name the input data set, select rows matching specific conditions (such as subsetting), grouping (or aggregating) the data, and ordering (or sorting) the data, as follows.

General SELECT Syntax:

```
PROC SQL options ;
  SELECT column(s)
    FROM table-name | view-name
      WHERE expression
        GROUP BY column(s)
          HAVING expression
            ORDER BY column(s) ;
QUIT;
```

Battle #1 – SAS Data Set Creation

In this first battle users will create SAS data sets (or tables) using the DATA step and PROC SQL. We will find out if one method has a distinct advantage over the other by demonstrating both methods below.

DATA Step Code:

```
LIBNAME MYDATA "/home/username/Data Sources" ;

DATA MYDATA.High_Blood_Pressure_Medications ;
  LENGTH BP_Status     $7.
         BP_Medication_1
         BP_Medication_2
         BP_Medication_3
         BP_Medication_4 $50. ;
  BP_Status = "High" ;
```

```
  BP_Medication_1 = "Thiazide Diuretics" ;
  BP_Medication_2 = "ACE (angiotensin-converting enzyme) inhibitors" ;
  BP_Medication_3 = "Angiotensin receptor blockers (ARBs)" ;
  BP_Medication_4 = "Calcium channel blockers" ;
RUN ;

PROC PRINT DATA=MYDATA.High_Blood_Pressure_Medications NOOBS ;
RUN ;
```

Results:

BP_Status	BP_Medication_1	BP_Medication_2	BP_Medication_3	BP_Medication_4
High	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers

PROC SQL Code:

```
LIBNAME MYDATA "/home/username/Data Sources" ;

PROC SQL ;
  CREATE TABLE MYDATA.High_Blood_Pressure_Medications
    (BP_Status CHAR(7)
    ,BP_Medication_1 CHAR(50)
    ,BP_Medication_2 CHAR(50)
    ,BP_Medication_3 CHAR(50)
    ,BP_Medication_4 CHAR(50)) ;
  INSERT INTO MYDATA.High_Blood_Pressure_Medications
    VALUES("High"
          ,"Thiazide Diuretics"
          ,"ACE (angiotensin-converting enzyme) inhibitors"
          ,"Angiotensin receptor blockers (ARBs)"
          ,"Calcium channel blockers") ;

  SELECT * FROM MYDATA.High_Blood_Pressure_Medications ;
QUIT ;
```

Results:

BP_Status	BP_Medication_1	BP_Medication_2	BP_Medication_3	BP_Medication_4
High	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers

While both techniques yield the same result, notice that to display the data set to the results window for the DATA step, the PRINT procedure is used. However, with PROC SQL, the SELECT statement will display the data set in the results window. Thus, with PROC SQL, the data set is created, populated, and displayed in one procedure.

Multiple Data Sets

In the previous battles, at the outset, it appears that both the DATA step and PROC SQL result in the same outcome and indeed that is correct. However, it is important to point out one edge the DATA step has. That is the ability to create multiple output data sets in one read of the input table, thereby saving valuable input/output resources.

Let's take the weight-status column to do some conditional processing. First, a FREQ procedure is used to identify unique values of Weight_status.

```
PROC FREQ DATA=SASHELP.Heart ORDER=FREQ ;
     TABLES weight_status ;
RUN ;
```

Weight Status				
Weight_Status	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Overweight	3550	68.23	3550	68.23
Normal	1472	28.29	5022	96.52
Underweight	181	3.48	5203	100.00
Frequency Missing = 6				

Now a DATA step can be used to create separate data sets for the unique values of weight_status. We do this by listing the table names on the DATA statement and then sending rows of data to the appropriate tables by using IF-THEN-ELSE conditional logic. Notice that the DATA step reads the HEART data set one time, and in that one read of the input table it can be made to divert data to multiple output tables.

DATA Step Code:

```
DATA WORK.overWt WORK.normalWt WORK.underWt ;
     SET SASHELP.Heart ;

     IF weight_status="Overweight" THEN
           OUTPUT WORK.overWt ;
     ELSE IF weight_status="Normal" THEN
           OUTPUT WORK.normalWt ;
     ELSE IF weight_status="Underweight" THEN
           OUTPUT WORK.underWt ;
RUN ;
```

How would PROC SQL handle the same challenge? See below how SQL innately can build just one output table for each read. Thus, we would need to read the HEART data set 3 times to build 3 output tables.

PROC SQL Code (Two Techniques):

```
PROC SQL ;
  CREATE TABLE WORK.overWt AS
  SELECT * FROM SASHELP.Heart
  WHERE weight_status="Overweight" ;
QUIT ;

PROC SQL ;
  CREATE TABLE WORK.normalwt AS
  SELECT * FROM SASHELP.Heart
  WHERE weight_status="Normal" ;
QUIT ;

PROC SQL ;
  CREATE TABLE WORK.underWt AS
  SELECT * FROM SASHELP.Heart
  WHERE weight_status="Underweight" ;
QUIT ;
```

```
PROC SQL ;
  CREATE TABLE WORK.overWt AS
   SELECT * FROM SASHELP.Heart
    WHERE weight_status="Overweight" ;

  CREATE TABLE WORK.normalwt AS
   SELECT * FROM SASHELP.Heart
    WHERE weight_status="Normal" ;

  CREATE TABLE WORK.underWt AS
   SELECT * FROM SASHELP.Heart
    WHERE weight_status="Underweight" ;
QUIT ;
```

Battle #2 – Data Access / Retrieval / Data Manipulation

In this second battle we turn our attention to data access, data retrieval, and data manipulation techniques. In the next example, we'll access and read an Excel spreadsheet by specifying a LIBNAME statement with the XLSX engine in a DATA step.

DATA Step Code:

```
LIBNAME MYDATA XLSX "/home/Data Sources/Heart.xlsx" ;

DATA WORK.Heart_DATA_Step ;
  SET MYDATA.Heart ;
RUN ;
```

SAS Log:

```
LIBNAME MYDATA XLSX "/home/Data Sources/Heart.xlsx" ;
NOTE: Libref MYDATA was successfully assigned as follows:
      Engine:        XLSX
      Physical Name: /home/Data Sources/Heart.xlsx

DATA WORK.Heart_DATA_Step ;
  SET MYDATA.Heart ;
RUN ;

NOTE: The import data set has 5209 observations and 17 variables.
NOTE: There were 5209 observations read from the data set MYDATA.Heart.
NOTE: The data set WORK.HEART_DATA_Step has 5209 observations and 17 variables.
```

In the next example, we'll access and read an Excel spreadsheet by specifying a LIBNAME statement with the XLSX engine and PROC SQL.

PROC SQL Code:

```
LIBNAME MYDATA XLSX "/home/Data Sources/Heart.xlsx" ;

PROC SQL ;
  CREATE TABLE WORK.Heart_PROC_SQL AS
    SELECT *
      FROM MYDATA.Heart ;
QUIT ;
```

SAS Log:

```
LIBNAME MYDATA XLSX "/home/Data Sources/Heart.xlsx" ;
NOTE: Libref MYDATA was successfully assigned as follows:
      Engine:        XLSX
      Physical Name: /home/Data Sources/Heart.xlsx

PROC SQL ;
  CREATE TABLE WORK.Heart_PROC_SQL AS
    SELECT *
      FROM MYDATA.Heart ;
QUIT ;

NOTE: The import data set has 5209 observations and 17 variables.
NOTE: There were 5209 observations read from the data set MYDATA.Heart.
NOTE: The data set WORK.HEART_PROC_SQL has 5209 observations and 17 variables.
```

The result of running both approaches is that there is no appreciable difference between the DATA step and PROC SQL approach.

Battle #3 – Logic Scenarios

Both the DATA step and PROC SQL provide us with the ability to apply logic scenarios in our programs so they can conditionally do or perform the operations we desire – "if one condition is true, then do X but if another condition is true, then do Y." But before engaging in this battle of logic scenarios we'll introduce comparison and logical operators.

Comparison Operators

Comparison operators are used in the DATA step and PROC SQL to compare one character or numeric value to another. As in the DATA step, SQL comparison operators, mnemonics, and their descriptions appear in the following table.

SAS Operator	Mnemonic Operator	Description
=	EQ	Equal to
^=	NE	Not equal to
<	LT	Less than
<=	LE	Less than or equal to
>	GT	Greater than
>=	GE	Greater than or equal to

Logical Operators

Logical operators are used to connect two or more expressions together. The three operators include AND, OR, and NOT as is displayed in the following table.

SAS Operator	Mnemonic Operator	Description
AND	&	All expressions (conditions) must be true.
OR	|	Any of the expressions (conditions) can be true.
NOT	^ or ~	Negate or reverse the logic of a comparison.

In this battle we'll explore the application of "IF-THEN-ELSE", "SELECT-WHEN-OTHERWISE" and WHERE conditional logic scenarios used in the DATA step and WHERE-clauses and Case expressions used in PROC SQL.

IF-THEN-ELSE Syntax in the DATA Step:

```
IF condition-1 THEN condition1-result
< ELSE IF condition-2 THEN condition2-result ; >
< ELSE IF condition-3 THEN condition3-result ; >
<    …         …          …     >
< ELSE condition-n ; >
```

Two types of select (DATA step) or case (PROC SQL) expression constructs exist:

- ✓ DATA Step Logic
  - IF-THEN-ELSE condition
  - WHEN-THEN-ELSE condition
  - Does not support AND / OR between Boolean expressions
- ✓ PROC SQL Case Expression
  - Simple Case Expression - Compares an expression against multiple values
  - Searched Case Expression - Specifies an equality check
    - ❖ Supports one or more WHEN-THEN-ELSE conditions
    - ❖ Permits the use of comparison operators
    - ❖ Supports the use of AND / OR between Boolean expressions

General SELECT Expression Syntax in the DATA Step:

```
SELECT (<select-expression>)
    WHEN (when-condition) statement ;
   <WHEN (when-condition) statement ;>
   <    …         …          …     >
   <OTHERWISE statement ;n>
END ;
```

DATA Step Code:

```
DATA WORK.Smoke ;
   SET SASHELP.Heart ;
   SELECT (Smoking_Status) ;
      WHEN ("Non-smoker")        My_Smoking_Status = "Non-smoker" ;
      WHEN ("Light (1-5)")       My_Smoking_Status = "Light-smoker" ;
      WHEN ("Moderate (6-15)")   My_Smoking_Status = "Moderate-smoker" ;
      WHEN ("Heavy (16-25)")     My_Smoking_Status = "Heavy-smoker" ;
      WHEN ("Very Heavy (> 25)") My_Smoking_Status = "Very Heavy-smoker" ;
      OTHERWISE                  My_Smoking_Status = "Unknown" ;
   END ;
   KEEP Sex Status Smoking_Status My_Smoking_Status ;
RUN ;

PROC PRINT DATA = WORK.Smoke NOOBS ;
RUN ;
```

A Select with case expressions in PROC SQL provides a method of reclassifying or regrouping data into separate and unique groups. A simple SELECT or CASE expression reduces the number of keystrokes required in constructing logic scenarios.

CASE Expression Syntax in PROC SQL:

```
CASE <column-name>
    WHEN when-condition THEN result
   <WHEN when-condition THEN result>
   <    …         …          …     >
   <ELSE result-expression>
END <AS user-defined-column-name>
```

PROC SQL Code:

```
PROC SQL ;
  SELECT Sex
       , Status
       , Smoking_Status
       , CASE Smoking_Status
           WHEN "Non-smoker"        THEN "Non-smoker"
           WHEN "Light (1-5)"       THEN "Light-smoker"
           WHEN "Moderate (6-15)"   THEN "Moderate-smoker"
           WHEN "Heavy (16-25)"     THEN "Heavy-smoker"
           WHEN "Very Heavy (> 25)" THEN "Very Heavy-smoker"
           ELSE "Unknown"
         END AS My_Smoking_Status
    FROM SASHELP.Heart ;
QUIT ;
```

Results:

Sex	Status	Smoking Status	My_Smoking_Status
Female	Dead	Non-smoker	Non-smoker
Female	Dead	Non-smoker	Non-smoker
Female	Alive	Moderate (6-15)	Moderate-smoker
Female	Alive	Non-smoker	Non-smoker
Male	Alive	Heavy (16-25)	Heavy-smoker
Female	Alive	Non-smoker	Non-smoker
Female	Alive	Moderate (6-15)	Moderate-smoker
Male	Dead	Non-smoker	Non-smoker
Male	Alive	Non-smoker	Non-smoker
Male	Dead	Light (1-5)	Light-smoker
Male	Alive	Very Heavy (> 25)	Very Heavy-smoker
Male	Alive	Non-smoker	Non-smoker
Male	Alive	Non-smoker	Non-smoker
Male	Alive	Moderate (6-15)	Moderate-smoker
Male	Alive	Very Heavy (> 25)	Very Heavy-smoker
Female	Alive	Moderate (6-15)	Moderate-smoker
Male	Alive	Very Heavy (> 25)	Very Heavy-smoker
Male	Dead	Moderate (6-15)	Moderate-smoker
Female	Alive	Light (1-5)	Light-smoker
Male	Dead	Very Heavy (> 25)	Very Heavy-smoker
Female	Alive	Moderate (6-15)	Moderate-smoker
Female	Alive	Light (1-5)	Light-smoker
Female	Alive	Non-smoker	Non-smoker
Female	Alive	Non-smoker	Non-smoker
Female	Alive	Moderate (6-15)	Moderate-smoker

A DATA step SELECT-WHEN-OTHERWISE expression supports the application of logic scenarios for conditional processing with comparison and logical operators.

DATA Step Code:

```
DATA WORK.Smoke_BP ;
   SET SASHELP.Heart ;
   LENGTH My_Smoking_Status $50;
   SELECT ;
      WHEN (Smoking_Status = “Non-smoker”)      My_Smoking_Status = “Non-smoker” ;
      WHEN (Smoking_Status = “Light (1-5)”)     My_Smoking_Status = “Light-smoker” ;
      WHEN (Smoking_Status = “Moderate (6-15)”) My_Smoking_Status = “Moderate-smoker” ;
      WHEN (Smoking_Status = “Heavy (16-25)” and BP_Status = “High”)
                        My_Smoking_Status = “Heavy-smoker with High Blood Pressure” ;
      WHEN (Smoking_Status = “Very Heavy (> 25)” and BP_Status = “High”)
                        My_Smoking_Status = “Very Heavy-smoker with High Blood Pressure” ;
      OTHERWISE                                 My_Smoking_Status = “Unknown” ;
   END ;
   KEEP Sex Status Smoking_Status My_Smoking_Status ;
RUN ;
```

A PROC SQL searched CASE expression supports the application of logic scenarios for conditional processing with comparison and logical operators.

PROC SQL Code:

```
PROC SQL ;
  SELECT Sex
       , Status
       , BP_Status
       , Smoking_Status
       , CASE
```

```
            WHEN Smoking_Status = "Non-smoker"        THEN "Non-smoker"
            WHEN Smoking_Status = "Light (1-5)"       THEN "Light-smoker"
            WHEN Smoking_Status = "Moderate (6-15)"   THEN "Moderate-smoker"
            WHEN Smoking_Status = "Heavy (16-25)"
             AND BP_Status = "High" THEN "Heavy-smoker with High Blood Pressure"
            WHEN Smoking_Status = "Very Heavy (> 25)"
             AND BP_Status = "High" THEN "Very Heavy-smoker with High Blood Pressure"
            ELSE "Unknown"
          END AS My_Smoking_Status
    FROM SASHELP.Heart ;
QUIT ;
```

Results:

Sex	Status	Blood Pressure Status	Smoking Status	My_Smoking_Status
Female	Dead	Normal	Non-smoker	Non-smoker
Female	Dead	High	Non-smoker	Non-smoker
Female	Alive	High	Moderate (6-15)	Moderate-smoker
Female	Alive	Normal	Non-smoker	Non-smoker
Male	Alive	Optimal	Heavy (16-25)	Unknown
Female	Alive	High	Non-smoker	Non-smoker
Female	Alive	Normal	Moderate (6-15)	Moderate-smoker
Male	Dead	Normal	Non-smoker	Non-smoker
Male	Alive	Normal	Non-smoker	Non-smoker
Male	Dead	Normal	Light (1-5)	Light-smoker
Male	Alive	Normal	Very Heavy (> 25)	Unknown
Male	Alive	Optimal	Non-smoker	Non-smoker
Male	Alive	High	Non-smoker	Non-smoker
Male	Alive	Normal	Moderate (6-15)	Moderate-smoker
Male	Alive	High	Very Heavy (> 25)	Very Heavy-smoker with High Blood Pressure
Female	Alive	Normal	Moderate (6-15)	Moderate-smoker
Male	Alive	Normal	Very Heavy (> 25)	Unknown
Male	Dead	Normal	Moderate (6-15)	Moderate-smoker
Female	Alive	High	Light (1-5)	Light-smoker
Male	Dead	High	Very Heavy (> 25)	Very Heavy-smoker with High Blood Pressure
Female	Alive	Optimal	Moderate (6-15)	Moderate-smoker
Female	Alive	Normal	Light (1-5)	Light-smoker
Female	Alive	High	Non-smoker	Non-smoker
Female	Alive	Normal	Non-smoker	Non-smoker
Female	Alive	High	Moderate (6-15)	Moderate-smoker

In this scenario, we may have to call it a draw. There is no clear 'winner' in this battle. It comes down to which syntax you are most comfortable with.

Battle #4 – BY-group Processing

SAS users often need the ability to identify the first (beginning) and last (ending) observation as well as the between observation(s) in a by-group. The DATA step is the "go-to" approach used by many but PROC SQL can also be used to emulate this stalwart DATA step approach. In the next example, a PROC SORT and DATA step illustrates a popular BY-group processing technique where two temporary variables: FIRST(dot) and LAST(dot) are automatically created to identify and select the first and last observation in a BY-group.

Identify the Maximum Value in BY-Groups

SAS provides users with several ways to identify the maximum value in a group. In the next example, we illustrate how to identify the maximum value in a group using PROC SORT, a DATA step, and PROC PRINT.

DATA Step Code:

```
PROC SORT DATA=SASHELP.Heart(KEEP=Sex Status Weight Smoking_Status)
          OUT=WORK.Heart_Sorted ;
  BY Smoking_Status DESCENDING Weight ;
  WHERE Smoking_Status NE "" AND Weight NE . ;
RUN ;

DATA WORK.BY_Group_Obs ;
  SET WORK.Heart_Sorted ;
  BY Smoking_Status ;
  IF FIRST.Smoking_Status THEN OUTPUT ;
RUN ;

PROC PRINT DATA=WORK.BY_Group_Obs NOOBS ;
  VAR Sex Status Weight Smoking_Status ;
RUN ;
```

Results:

Sex	Status	Weight	Smoking_Status
Female	Dead	300	Heavy (16-25)
Female	Alive	239	Light (1-5)
Male	Alive	237	Moderate (6-15)
Female	Alive	300	Non-smoker
Male	Dead	256	Very Heavy (> 25)

In the next example, we illustrate how to identify the maximum value in a group using PROC SQL.

PROC SQL Code:

```
PROC SQL ;
  SELECT Sex
       , Status
       , Weight
       , Smoking_Status
    FROM SASHELP.Heart(KEEP=Sex Status Weight Smoking_Status)
      GROUP BY Smoking_Status
        HAVING Weight = MAX(Weight) AND Smoking_Status NE "" ;
QUIT ;
```

Results:

Sex	Status	Weight	Smoking Status
Female	Dead	300	Heavy (16-25)
Male	Dead	239	Light (1-5)
Female	Alive	239	Light (1-5)
Male	Alive	237	Moderate (6-15)
Female	Alive	300	Non-smoker
Male	Dead	256	Very Heavy (> 25)

As seen both approaches yield the same results, but PROC SQL is more concise and does not require the pre-sorting that is needed in the DATA step or the display of the results.

Count the Number of Observations by Group

SAS provides users with several ways to count the number of observations (or rows) in a group. We will illustrate two methods to count the number of observations contained in a group: a DATA step and PROC SQL. By default, the results produced in both examples are ordered in ascending order by the value contained in the Smoking_Status variable.

DATA Step Code:

```
PROC SORT DATA=SASHELP.Heart(KEEP=Smoking_Status)
          OUT=WORK.Heart_Sorted ;
  BY Smoking_Status ;
RUN ;

DATA WORK.BY_Group_Obs ;
  SET WORK.Heart_Sorted ;
  BY Smoking_Status ;
  IF FIRST.Smoking_Status THEN CTR_Smoking_Status = 0 ;
  CTR_Smoking_Status + 1 ;
  IF LAST.Smoking_Status THEN OUTPUT ;
  FORMAT CTR_Smoking_Status COMMA10. ;
RUN ;

PROC PRINT DATA=WORK.BY_Group_Obs NOOBS ;
RUN ;
```

Results:

Smoking Status	CTR_BY_Group
	36
Heavy (16-25)	1,046
Light (1-5)	579
Moderate (6-15)	576
Non-smoker	2,501
Very Heavy (> 25)	471

PROC SQL Code:

```
PROC SQL ;
  SELECT Smoking_Status
       , COUNT(*) AS CTR_BY_Group FORMAT=COMMA10.
    FROM SASHELP.Heart
      GROUP BY Smoking_Status ;
QUIT ;
```

Results:

Smoking Status	CTR_BY_Group
	36
Heavy (16-25)	1,046
Light (1-5)	579
Moderate (6-15)	576
Non-smoker	2,501
Very Heavy (> 25)	471

While PROC SQL may have won this skirmish, the DATA step is not giving up this dogfight.

An added feature provided with the FIRST(dot) and LAST(dot) automatic variables is the ability to identify and select the BETWEEN observations contained in a BY-group.

DATA Step Code:

```
PROC SORT DATA=SASHELP.Heart(KEEP=Smoking_Status Weight_Status)
          OUT=WORK.Heart_Sorted ;
  BY Smoking_Status Weight_Status ;
RUN ;

DATA WORK.Weight_Status_Groups ;
  RETAIN Smoking_Status Weight_Status ;
  SET WORK.Heart_Sorted ;
  BY Smoking_Status Weight_Status ;
  WHERE Smoking_Status NE "" ;
  WHERE Weight_Status  NE "" ;
  IF Weight_Status="Normal"      THEN CTR_Normal + 1 ;
  IF Weight_Status="Underweight" THEN CTR_Underweight + 1 ;
  IF Weight_Status="Overweight"  THEN CTR_Overweight + 1 ;
  Total_Weight_Status = CTR_Normal + CTR_Underweight + CTR_Overweight ;
  IF LAST.Smoking_Status THEN DO ;
     OUTPUT ;
     CTR_Normal = 0 ;
     CTR_Underweight = 0 ;
     CTR_Overweight  = 0 ;
     Total_Weight_Status = 0 ;
  END ;
RUN ;

PROC REPORT DATA=WORK.Weight_Status_Groups ;
  COLUMNS Smoking_Status CTR_Normal CTR_Underweight CTR_Overweight
          Total_Weight_Status ;
  FORMAT CTR_Normal CTR_Underweight CTR_Overweight Total_Weight_Status COMMA10.0 ;
  DEFINE Smoking_Status / ORDER ;
RUN ;
```

Results:

Smoking Status	CTR_Normal	CTR_Underweight	CTR_Overweight	Total_Weight_Status
Heavy (16-25)	381	47	618	1,046
Light (1-5)	171	32	375	578
Moderate (6-15)	234	35	306	575
Non-smoker	543	53	1,903	2,499
Very Heavy (> 25)	133	11	325	469

PROC SQL Code:

```
PROC SQL ;
  SELECT Smoking_Status
       , Weight_Status
       , COUNT(Weight_Status) AS CTR_Weight_Status FORMAT=COMMA10.0
    FROM SASHELP.HEART
      WHERE Smoking_Status NE ""
        AND Weight_Status  NE ""
        GROUP BY Smoking_Status, Weight_Status
          HAVING COUNT(Weight_Status) > 0 ;
QUIT ;
```

Results:

Smoking Status	Weight Status	CTR_Weight_Status
Heavy (16-25)	Normal	381
Heavy (16-25)	Overweight	618
Heavy (16-25)	Underweight	47
Light (1-5)	Normal	171
Light (1-5)	Overweight	375
Light (1-5)	Underweight	32
Moderate (6-15)	Normal	234
Moderate (6-15)	Overweight	306
Moderate (6-15)	Underweight	35
Non-smoker	Normal	543
Non-smoker	Overweight	1,903
Non-smoker	Underweight	53
Very Heavy (> 25)	Normal	133
Very Heavy (> 25)	Overweight	325

Notice that with PROC SQL it does not produce the overall Total_Weight_Status. To incorporate the total counts with the in-between counts, an additional PROC SQL statement is needed. Furthermore, to put the data from PROC SQL in the same format as the data from the DATA step, additional processing is needed and the data from PROC SQL would need to be transposed.

Nearest Neighbor Processing by Emulating LAG and LEAD Functionality

As a rule, SQL queries are designed to perform operations on a row-by-row basis. But sometimes a problem comes along where row-by-row processing will not work. For example, nearest neighbor problems identify content that is close to another value. In the DATA step, functions such as LAG are often used to perform operations. Since PROC SQL does not support the use of the LAG (or LEAD) functions, another approach must be used.

DATA Step Code:

```
DATA WORK.Nearest_Neighbor(KEEP=Sex Status Smoking_Status Previous_Smoking_Status
                               Next_Smoking_Status) ;
  RETAIN Sex Status Smoking_Status Previous_Smoking_Status Next_Smoking_Status ;
  ARRAY FLAG {10000} $1 _TEMPORARY_;
  SET SASHELP.Heart(KEEP=Sex Status Smoking_Status) ;
  Current_Obs+1 ;
  Loop=0 ;
  DO POINT=(Current_Obs-1) to (Current_Obs+1) ;
     Loop+1 ;
     SET SASHELP.Heart(KEEP=Sex Status Smoking_Status
                    RENAME=(Smoking_Status=Smoking_Status1)) POINT=POINT NOBS=NOOBS ;
     IF Loop=1 THEN Previous_Smoking_Status=Smoking_Status1 ;
     IF Loop=2 THEN Smoking_Status=Smoking_Status ;
     IF Loop=3 THEN Next_Smoking_Status=Smoking_Status1 ;
  END ;
  OUTPUT WORK.Nearest_Neighbor ;
RUN;
```

An alternative to using a DO loop in the DATA step is to utilize the double SET statement within the DATA step.

```
DATA WORK.Nearest_Neighbor;
   SET SASHELP.Heart (KEEP=Sex Status Smoking_Status) END = EOF;
   Previous_Smoking_Status=LAG(Smoking_Status) ;
   IF NOT(EOF) THEN
     SET SASHELP.Heart (KEEP=Sex Status Smoking_Status
                        RENAME=(Smoking_Status=Next_Smoking_Status)
                        FIRSTOBS=2) ;
```

```
   ELSE IF EOF ;
RUN;

PROC PRINT DATA=WORK.Nearest_Neighbor NOOBS ;
RUN ;
```

Results:

Sex	Status	Smoking Status	Previous_Smoking_Status	Next_Smoking_Status
Female	Dead	Non-smoker		Non-smoker
Female	Dead	Non-smoker	Non-smoker	Moderate (6-15)
Female	Alive	Moderate (6-15)	Non-smoker	Non-smoker
Female	Alive	Non-smoker	Moderate (6-15)	Heavy (16-25)
Male	Alive	Heavy (16-25)	Non-smoker	Non-smoker
Female	Alive	Non-smoker	Heavy (16-25)	Moderate (6-15)
Female	Alive	Moderate (6-15)	Non-smoker	Non-smoker
Male	Dead	Non-smoker	Moderate (6-15)	Non-smoker
Male	Alive	Non-smoker	Non-smoker	Light (1-5)
Male	Dead	Light (1-5)	Non-smoker	Very Heavy (> 25)
Male	Alive	Very Heavy (> 25)	Light (1-5)	Non-smoker
Male	Alive	Non-smoker	Very Heavy (> 25)	Non-smoker
Male	Alive	Non-smoker	Non-smoker	Moderate (6-15)
Male	Alive	Moderate (6-15)	Non-smoker	Very Heavy (> 25)
Male	Alive	Very Heavy (> 25)	Moderate (6-15)	Moderate (6-15)
Female	Alive	Moderate (6-15)	Very Heavy (> 25)	Very Heavy (> 25)
Male	Alive	Very Heavy (> 25)	Moderate (6-15)	Moderate (6-15)
Male	Dead	Moderate (6-15)	Very Heavy (> 25)	Light (1-5)

PROC SQL Code:

```
ODS OUTPUT SQL_RESULTS = Heart_with_Row_Numbers ;

PROC SQL NUMBER ;
  SELECT Sex, Status, Smoking_Status
    FROM SASHELP.Heart ;
QUIT ;

PROC SQL NONUMBER ;
  CREATE TABLE WORK.Nearest_Neighbor(drop=Row) as
    SELECT *,
      (SELECT Smoking_Status
         FROM Heart_with_Row_Numbers
           WHERE Row = H.Row - 1) AS Previous_Smoking_Status,
      (SELECT Smoking_Status
         FROM Heart_with_Row_Numbers
           WHERE Row = H.Row + 1) AS Next_Smoking_Status
      FROM Heart_with_Row_Numbers H ;
  SELECT * FROM WORK.Nearest_Neighbor ;
QUIT ;
```

Results:

Sex	Status	Smoking Status	Previous_Smoking_Status	Next_Smoking_Status
Female	Dead	Non-smoker		Non-smoker
Female	Dead	Non-smoker	Non-smoker	Moderate (6-15)
Female	Alive	Moderate (6-15)	Non-smoker	Non-smoker
Female	Alive	Non-smoker	Moderate (6-15)	Heavy (16-25)
Male	Alive	Heavy (16-25)	Non-smoker	Non-smoker
Female	Alive	Non-smoker	Heavy (16-25)	Moderate (6-15)
Female	Alive	Moderate (6-15)	Non-smoker	Non-smoker
Male	Dead	Non-smoker	Moderate (6-15)	Non-smoker
Male	Alive	Non-smoker	Non-smoker	Light (1-5)
Male	Dead	Light (1-5)	Non-smoker	Very Heavy (> 25)
Male	Alive	Very Heavy (> 25)	Light (1-5)	Non-smoker
Male	Alive	Non-smoker	Very Heavy (> 25)	Non-smoker
Male	Alive	Non-smoker	Non-smoker	Moderate (6-15)
Male	Alive	Moderate (6-15)	Non-smoker	Very Heavy (> 25)
Male	Alive	Very Heavy (> 25)	Moderate (6-15)	Moderate (6-15)
Female	Alive	Moderate (6-15)	Very Heavy (> 25)	Very Heavy (> 25)
Male	Alive	Very Heavy (> 25)	Moderate (6-15)	Moderate (6-15)
Male	Dead	Moderate (6-15)	Very Heavy (> 25)	Light (1-5)

When it comes to “look behind” or “look ahead”, the DATA step utilizes the double SET within the one DATA step which makes the code more concise.

Battle #5 – Combining Data Sets (or Tables)

In the fifth, and final, battle we illustrate various DATA step and PROC SQL techniques to combine two data sets (or tables) together. The following table illustrates merge / join techniques using Venn diagrams, see below.

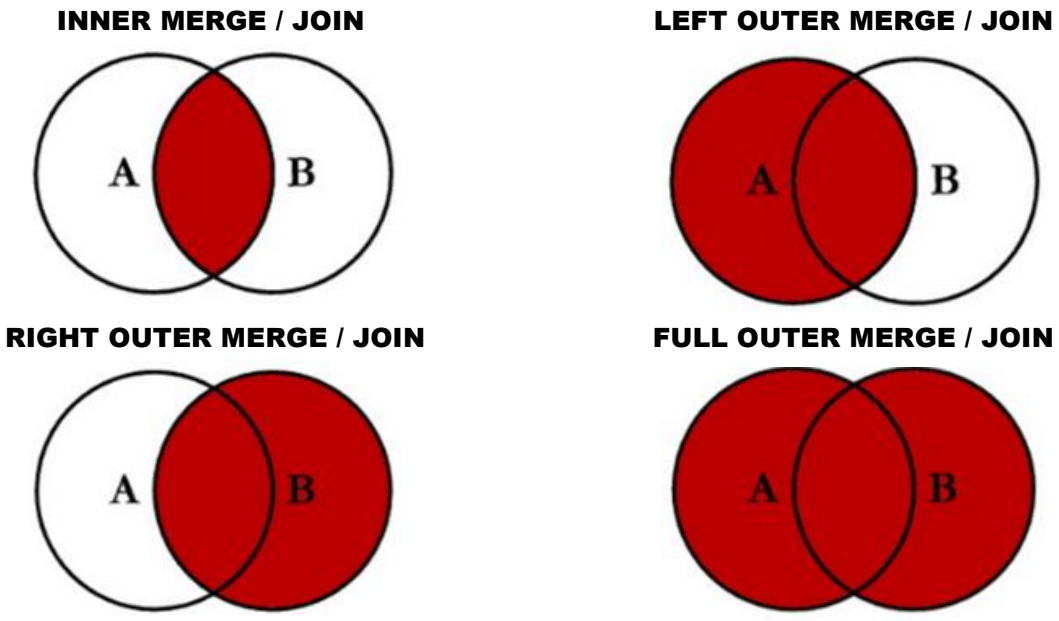

Illustration of Merges / Joins using Venn Diagrams

MATCH-MERGE / INNER JOIN (Intersect)

In the next example we illustrate how to combine tables together by constructing a MATCH-MERGE (or Intersect) construct using a DATA step.

DATA Step Code:

```
PROC SORT DATA=SASHELP.Heart
          OUT=WORK.Heart_Sorted ;
  BY BP_Status ;
  WHERE BP_Status = "High" ;
RUN ;

DATA WORK.Heart_HBP_Medications_MM ;
  MERGE WORK.Heart_Sorted(IN=H)
        MYDATA.High_Blood_Pressure_Medications(IN=HBP) ;
  BY BP_Status ;
  IF H AND HBP ;
RUN ;

PROC PRINT DATA=WORK.Heart_HBP_Medications_MM NOOBS ;
RUN ;
```

Results:

Status	DeathCause	AgeCHDdiag	Sex	AgeAtStart	Height	Weight	Diastolic	Systolic	MRW	Smoking	AgeAtDeath	Cholesterol	Chol_Status	BP_Status	Weight_Status	Smoking_Status	BP_Medication_1	BP_Medication_2	BP_Medication_3	BP_Medication_4
Dead	Cancer	.	Female	41	59.75	194	92	144	183	0	57	181	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	57	62.25	132	90	170	114	10	.	250	High	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	58	61.75	131	92	176	117	0	.	196	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		55	Male	33	70.00	174	90	142	114	0	.	188	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		66	Male	44	69.00	155	90	130	105	30	.	292	High	High	Normal	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	42	67.75	162	96	138	119	1	.	200	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	74	Male	46	66.50	157	84	142	116	30	76	233	Borderline	High	Overweight	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	59	65.75	156	74	156	122	0	.	200	Borderline	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	50	67.50	185	88	150	136	15	.	228	Borderline	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	71	Female	49	60.50	153	110	196	140	5	73	221	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Unknown	.	Female	59	67.75	153	82	172	113	0	79	263	High	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers

2,267 Observations and 21 Variables

In the next example we illustrate how to combine tables together by constructing an INNER JOIN construct (or Intersect) using PROC SQL.

PROC SQL Code:

```
LIBNAME MYDATA "/home/username/Data Sources" ;

PROC SQL NONUMBER ;
  CREATE TABLE WORK.Heart_HBP_Medications_IJ AS
    SELECT *
      FROM SASHELP.HEART H
           INNER JOIN
           MYDATA.High_Blood_Pressure_Medications HBP
        ON H.BP_Status = HBP.BP_Status ;

  SELECT *
    FROM WORK.Heart_HBP_Medications_IJ ;
QUIT ;
```

Results:

Status	DeathCause	AgeCHDdiag	Sex	AgeAtStart	Height	Weight	Diastolic	Systolic	MRW	Smoking	AgeAtDeath	Cholesterol	Chol_Status	BP_Status	Weight_Status	Smoking_Status	BP_Medication_1	BP_Medication_2	BP_Medication_3	BP_Medication_4
Dead	Cancer	.	Female	41	59.75	194	92	144	183	0	57	181	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	57	62.25	132	90	170	114	10	.	250	High	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	58	61.75	131	92	176	117	0	.	196	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		55	Male	33	70.00	174	90	142	114	0	.	188	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		66	Male	44	69.00	155	90	130	105	30	.	292	High	High	Normal	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	42	67.75	162	96	138	119	1	.	200	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	74	Male	46	66.50	157	84	142	116	30	76	233	Borderline	High	Overweight	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	59	65.75	156	74	156	122	0	.	200	Borderline	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	50	67.50	185	88	150	136	15	.	228	Borderline	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	71	Female	49	60.50	153	110	196	140	5	73	221	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Unknown	.	Female	59	67.75	153	82	172	113	0	79	263	High	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers

2,267 Observations and 21 Variables

LEFT OUTER MERGE / JOIN (Intersect + Unmatched Rows from Left Table)

In the next example we illustrate the process of combining two tables together using a Left Outer Merge in a DATA step.

DATA Step Code:

```
PROC SORT DATA=SASHELP.Heart
          OUT=WORK.Heart_Sorted ;
  BY BP_Status ;
RUN ;

DATA WORK.Heart_HBP_Medications_LOM ;
  MERGE WORK.Heart_Sorted(IN=H)
        MYDATA.High_Blood_Pressure_Medications(IN=HBP) ;
  BY BP_Status ;
  IF H ;
RUN ;

PROC PRINT DATA=WORK.Heart_HBP_Medications_LOM NOOBS ;
RUN ;
```

Results:

Status	DeathCause	AgeCHDdiag	Sex	AgeAtStart	Height	Weight	Diastolic	Systolic	MRW	Smoking	AgeAtDeath	Cholesterol	Chol_Status	BP_Status	Weight_Status	Smoking_Status	BP_Medication_1	BP_Medication_2	BP_Medication_3	BP_Medication_4
Dead	Cancer	.	Female	41	59.75	194	92	144	183	0	57	181	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	57	62.25	132	90	170	114	10	.	250	High	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	58	61.75	131	92	176	117	0	.	196	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		55	Male	33	70.00	174	90	142	114	0	.	188	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		66	Male	44	69.00	155	90	130	105	30	.	292	High	High	Normal	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	42	67.75	162	96	138	119	1	.	200	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	74	Male	46	66.50	157	84	142	116	30	76	233	Borderline	High	Overweight	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	59	65.75	156	74	156	122	0	.	200	Borderline	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	50	67.50	185	88	150	136	15	.	228	Borderline	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	71	Female	49	60.50	153	110	196	140	5	73	221	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Unknown	.	Female	59	67.75	153	82	172	113	0	79	263	High	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers

5,209 Observations and 21 Variables

In the next example we illustrate the process of combining two tables together using a Left Outer Join in PROC SQL.

PROC SQL Code:

```
PROC SQL NONUMBER ;
  CREATE TABLE WORK.Heart_HBP_Medications_LJ AS
    SELECT *
      FROM SASHELP.HEART H
             LEFT JOIN
           MYDATA.High_Blood_Pressure_Medications HBP
        ON H.BP_Status = HBP.BP_Status ;

  SELECT *
    FROM WORK.Heart_HBP_Medications_LJ ;
QUIT ;
```

Results:

Status	DeathCause	AgeCHDdiag	Sex	AgeAtStart	Height	Weight	Diastolic	Systolic	MRW	Smoking	AgeAtDeath	Cholesterol	Chol_Status	BP_Status	Weight_Status	Smoking_Status	BP_Medication_1	BP_Medication_2	BP_Medication_3	BP_Medication_4
Dead	Cancer	.	Female	41	59.75	194	92	144	183	0	57	181	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	57	62.25	132	90	170	114	10	.	250	High	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	58	61.75	131	92	176	117	0	.	196	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		55	Male	33	70.00	174	90	142	114	0	.	188	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		66	Male	44	69.00	155	90	130	105	30	.	292	High	High	Normal	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	42	67.75	162	96	138	119	1	.	200	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	74	Male	46	66.50	157	84	142	116	30	76	233	Borderline	High	Overweight	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	59	65.75	156	74	156	122	0	.	200	Borderline	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	50	67.50	185	88	150	136	15	.	228	Borderline	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	71	Female	49	60.50	153	110	196	140	5	73	221	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Unknown	.	Female	59	67.75	153	82	172	113	0	79	263	High	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers

5,209 Observations and 21 Variables

RIGHT OUTER MERGE / JOIN (Intersect + Unmatched Rows from Right Table)

In the next example we illustrate the process of combining two tables together using a right outer merge in a DATA step.

DATA Step Code:

```
PROC SORT DATA=SASHELP.Heart
          OUT=WORK.Heart_Sorted ;
  BY BP_Status ;
RUN ;

DATA WORK.Heart_HBP_Medications_ROM ;
  MERGE WORK.Heart_Sorted(IN=H)
        MYDATA.High_Blood_Pressure_Medications(IN=HBP) ;
  BY BP_Status ;
  IF HBP ;
RUN ;

PROC PRINT DATA=WORK.Heart_HBP_Medications_ROM NOOBS ;
RUN ;
```

Results:

Status	DeathCause	AgeCHDdiag	Sex	AgeAtStart	Height	Weight	Diastolic	Systolic	MRW	Smoking	AgeAtDeath	Cholesterol	Chol_Status	BP_Status	Weight_Status	Smoking_Status	BP_Medication_1	BP_Medication_2	BP_Medication_3	BP_Medication_4
Dead	Cancer	.	Female	41	59.75	194	92	144	183	0	57	181	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	57	62.25	132	90	170	114	10	.	250	High	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	58	61.75	131	92	176	117	0	.	196	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		55	Male	33	70.00	174	90	142	114	0	.	188	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		66	Male	44	69.00	155	90	130	105	30	.	292	High	High	Normal	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	42	67.75	162	96	138	119	1	.	200	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	74	Male	46	66.50	157	84	142	116	30	76	233	Borderline	High	Overweight	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	59	65.75	156	74	156	122	0	.	200	Borderline	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	50	67.50	185	88	150	136	15	.	228	Borderline	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	71	Female	49	60.50	153	110	196	140	5	73	221	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Unknown	.	Female	59	67.75	153	82	172	113	0	79	263	High	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers

2,267 Observations and 21 Variables

In the next example we illustrate the process of combining two tables together using a right outer join in PROC SQL.

PROC SQL Code:

```
PROC SQL NONUMBER ;
  CREATE TABLE WORK.Heart_HBP_Medications_RJ AS
    SELECT *
      FROM SASHELP.HEART H
             RIGHT JOIN
           MYDATA.High_Blood_Pressure_Medications HBP
        ON H.BP_Status = HBP.BP_Status ;

  SELECT *
    FROM WORK.Heart_HBP_Medications_RJ ;
QUIT ;
```

Results:

Status	DeathCause	AgeCHDdiag	Sex	AgeAtStart	Height	Weight	Diastolic	Systolic	MRW	Smoking	AgeAtDeath	Cholesterol	Chol_Status	BP_Status	Weight_Status	Smoking_Status	BP_Medication_1	BP_Medication_2	BP_Medication_3	BP_Medication_4
Dead	Cancer	.	Female	41	59.75	194	92	144	183	0	57	181	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	57	62.25	132	90	170	114	10	.	250	High	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	58	61.75	131	92	176	117	0	.	196	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		55	Male	33	70.00	174	90	142	114	0	.	188	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		66	Male	44	69.00	155	90	130	105	30	.	292	High	High	Normal	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	42	67.75	162	96	138	119	1	.	200	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	74	Male	46	66.50	157	84	142	116	30	76	233	Borderline	High	Overweight	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	59	65.75	156	74	156	122	0	.	200	Borderline	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	50	67.50	185	88	150	136	15	.	228	Borderline	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	71	Female	49	60.50	153	110	196	140	5	73	221	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Unknown	.	Female	59	67.75	153	82	172	113	0	79	263	High	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers

2,267 Observations and 21 Variables

FULL OUTER MERGE / JOIN (Intersect + Unmatched Rows from Left and Right Tables)

In the next example we illustrate the process of combining two tables together using a full outer merge in a DATA step.

DATA Step Code:

```
PROC SORT DATA=SASHELP.Heart
          OUT=WORK.Heart_Sorted ;
  BY BP_Status ;
RUN ;

DATA WORK.Heart_HBP_Medications_FOM ;  /* FULL OUTER MERGE */
  MERGE WORK.Heart_Sorted(IN=H)
        MYDATA.High_Blood_Pressure_Medications(IN=HBP) ;
  BY BP_Status ;
  IF H OR HBP ;
RUN ;

PROC PRINT DATA=WORK.Heart_HBP_Medications_FOM NOOBS ;
RUN ;
```

Results:

Status	DeathCause	AgeCHDdiag	Sex	AgeAtStart	Height	Weight	Diastolic	Systolic	MRW	Smoking	AgeAtDeath	Cholesterol	Chol_Status	BP_Status	Weight_Status	Smoking_Status	BP_Medication_1	BP_Medication_2	BP_Medication_3	BP_Medication_4
Dead	Cancer	.	Female	41	59.75	194	92	144	183	0	57	181	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	57	62.25	132	90	170	114	10	.	250	High	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	58	61.75	131	92	176	117	0	.	196	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		55	Male	33	70.00	174	90	142	114	0	.	188	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		66	Male	44	69.00	155	90	130	105	30	.	292	High	High	Normal	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	42	67.75	162	96	138	119	1	.	200	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	74	Male	46	66.50	157	84	142	116	30	76	233	Borderline	High	Overweight	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	59	65.75	156	74	156	122	0	.	200	Borderline	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	50	67.50	185	88	150	136	15	.	228	Borderline	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	71	Female	49	60.50	153	110	196	140	5	73	221	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Unknown	.	Female	59	67.75	153	82	172	113	0	79	263	High	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers

5,209 Observations and 21 Variables

In the next example we illustrate the process of combining two tables together using a full outer join in PROC SQL.

PROC SQL Code:

```
PROC SQL NONUMBER ;
  CREATE TABLE WORK.Heart_HBP_Medications_FJ AS
    SELECT *
      FROM SASHELP.HEART H
             FULL JOIN
           MYDATA.High_Blood_Pressure_Medications HBP
        ON H.BP_Status = HBP.BP_Status ;

  SELECT *
    FROM WORK.Heart_HBP_Medications_FJ ;
QUIT ;
```

Results:

Status	DeathCause	AgeCHDdiag	Sex	AgeAtStart	Height	Weight	Diastolic	Systolic	MRW	Smoking	AgeAtDeath	Cholesterol	Chol_Status	BP_Status	Weight_Status	Smoking_Status	BP_Medication_1	BP_Medication_2	BP_Medication_3	BP_Medication_4
Dead	Cancer	.	Female	41	59.75	194	92	144	183	0	57	181	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	57	62.25	132	90	170	114	10	.	250	High	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	58	61.75	131	92	176	117	0	.	196	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		55	Male	33	70.00	174	90	142	114	0	.	188	Desirable	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		66	Male	44	69.00	155	90	130	105	30	.	292	High	High	Normal	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	42	67.75	162	96	138	119	1	.	200	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	74	Male	46	66.50	157	84	142	116	30	76	233	Borderline	High	Overweight	Very Heavy (> 25)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	59	65.75	156	74	156	122	0	.	200	Borderline	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Alive		.	Female	50	67.50	185	88	150	136	15	.	228	Borderline	High	Overweight	Moderate (6-15)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Coronary Heart Disease	71	Female	49	60.50	153	110	196	140	5	73	221	Borderline	High	Overweight	Light (1-5)	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers
Dead	Unknown	.	Female	59	67.75	153	82	172	113	0	79	263	High	High	Overweight	Non-smoker	Thiazide Diuretics	ACE (angiotensin-converting enzyme) inhibitors	Angiotensin receptor blockers (ARBs)	Calcium channel blockers

5,209 Observations and 21 Variables

Conclusion

So, "Which colossus approach is better?", you might ask. Well, our task was not to declare a winner or a loser, but rather to show that the DATA step and PROC SQL could both be used to meet your data processing needs. This paper is intended to illustrate the differences, and the similarities, between the DATA step and PROC SQL. What we have attempted to do is give you an insightful way, and hopefully ignite a curiosity for further discovery about these two SAS "TITANS" (To be clear, we're talking about the DATA step and SQL procedure here, not the authors.) It is also worth mentioning that each technique has a definite set of strengths over the other depending on the desired result. The tools are available, and they serve as powerful foundations to valuable and productive processing.

References

Lafler, Kirk Paul, Joshua Horstman, Ben Cochran, Ray Pass, and Dan Bruns (2023). "Battle of the Titans (Part II): PROC REPORT versus PROC TABULATE," Proceedings of the 2023 PharmaSUG Conference.

Lafler, Kirk Paul (2019). PROC SQL: Beyond the Basics Using SAS, Third Edition, SAS Institute Inc., Cary, NC, USA.

Lafler, Kirk Paul, Joshua Horstman, Ben Cochran, Ray Pass, and Dan Bruns (2022). "Battle of the Titans (Part II): PROC REPORT versus PROC TABULATE," Proceedings of the 2022 SouthEast SAS Users Group (SESUG) Conference.

Lafler, Kirk Paul, Joshua Horstman, Ben Cochran, Ray Pass, and Dan Bruns (2022). "Battle of the Titans (Part II): PROC REPORT versus PROC TABULATE," Proceedings of the 2022 Western Users of SAS Software (WUSS) Conference.

Lafler, Kirk Paul, Ben Cochran, Josh Horstman, and Ray Pass (2019). "Battle of the Titans (Part II): PROC TABULATE versus PROC REPORT," Proceedings of the 2019 SouthEast SAS Users Group (SESUG) Conference.

Lafler, Kirk Paul, Ben Cochran, and Ray Pass (2017). "Battle of the Titans (Part II): PROC TABULATE versus PROC REPORT," Proceedings of the 2017 MidWest SAS Users Group (MWSUG) Conference.

Acknowledgments

The authors thank the WUSS 2023 Conference Committee, particularly the Hands-On Workshops (HOW) Section Chair, Jenny Antunez, for accepting our paper; the WUSS 2023 Academic Chair, Lida Gharibvand, and the Operation Chair, Julie Kilburn, for organizing and supporting a great "live" conference event; SAS Institute Inc. for providing SAS users with wonderful software; and SAS users everywhere for being the nicest people anywhere!

Trademark Citations

About the Authors

Kirk Paul Lafler is an educator, developer, programmer, consultant, and data analyst; currently working as a lecturer and adjunct professor at San Diego State University and the University of California San Diego Extension; and teaching SAS, SQL, Python, Excel, and cloud-based technology courses to users around the world. Kirk has decades of programming experience and specializes in SAS software, SQL, RDBMS technologies (Oracle, SQL-Server, Teradata, DB2), Python, and other languages and productivity tools. Kirk is the author of the popular PROC SQL: Beyond the Basics Using SAS, Third Edition (SAS Press. 2019) and is actively involved with SAS, SQL, Python, R, ML, and cloud-computing user groups, conferences, and blogs as an Invited speaker, educator, keynote, and leader; and is the recipient of 27 "Best" contributed paper, hands-on workshop (HOW), and poster awards.

Richann Jean Watson is an independent statistical programmer and CDISC consultant based in Ohio who loves to code and is very active in the SAS User Group community. When Richann is not busy coding or volunteering in the SAS User Group community, she is spending time with her family and cute but psycho puppy, Loki, or doing some of her favorite crafts such as crocheting or sewing.

Joshua Horstman is an independent statistical programming consultant and trainer based in Indianapolis with over 25 years of experience using SAS, primarily in the life sciences industry. Josh is a SAS Certified Advanced Programmer who loves coding and presenting at SAS user group conferences and other industry conferences. Josh also enjoys travelling and hiking with his family and has been to 47 states and 27 national parks.

Charu Shankar has been teaching SAS for over 15 years. She teaches by engaging with logic, visuals, and analogies to spark critical thinking. In addition to teaching public classes, and academia, Charu has presented at over 150 SAS international conferences on SAS, SAS Viya, SQL, Macro, Hadoop, Python, DS2, tips and tricks, efficiencies When she's not teaching SAS, Charu is a food blogger, yoga teacher & loves to hike Canadian trails with Miko, her husky.

Comments and suggestions can be sent to:

Kirk Paul Lafler
Data Scientist, Developer, Programmer, Consultant, Educator, Data Analyst, and Author
sasNerd
E-mail: KirkLafler@cs.com
LinkedIn: http://www.linkedin.com/in/KirkPaulLafler
Twitter: @sasNerd

Richann Jean Watson
Statistical Programmer and CDISC Consultant
DataRich Consulting
E-mail: richann.watson@datarichconsulting.com
LinkedIn: https://www.linkedin.com/in/richann-watson-31435422/
Website: https://datarichconsulting.com/

Joshua M. Horstman
Statistical Programming Consultant and Trainer
Nested Loop Consulting
E-mail: josh@nestedloopconsulting.com
LinkedIn: https://www.linkedin.com/in/joshuahorstman

Charu Shankar
SAS® Educator
SAS Institute Inc.
E-mail: Charu.shankar@sas.com
LinkedIn: https://www.linkedin.com/in/charushankar/