

Innovative Ways to Enhance Proc SQL Output

Kalyani Chaturvedula, Health Management Corporation, Richmond, VA

ABSTRACT

SAS® programmers are always looking for new and different ways to format their output. PROC SQL, though not widely used to format output, provides the functionality necessary to allow the user a number of innovative ways to enhance the appearance of their output. SQL or Structured Query Language is a language primarily developed for retrieving data from relational databases, but is much more robust in application. These formatting methods are especially beneficial for SAS programmers who prefer Proc SQL to SAS data step and would like to get their output formatted within the PROC SQL procedure rather than do an additional data step to format output but are relevant for a general SAS audience. Examples were created with SAS 9.1.3 of SAS executing on a Windows XP platform but are not platform specific.

Keywords: SQL, CATX, RESET, FORMAT

INTRODUCTION

Just like data retrieval is simplified with SQL code, the code to determine data appearance can also be simplified. Some of the SQL output enhancements discussed by this paper include:

- o Double spacing the rows;
- o Inserting text and constant between selected columns;
- o Displaying column headers for derived fields;
- o Concatenating two or more columns;
- o Displaying Row Numbers
- o Labels and Formats
- o Titles and Footnotes

The objective of this paper is to demonstrate that SAS users can use output formats and options in PROC SQL without having to do an extra DATA STEP, thus making the code efficient.

DOUBLE SPACING ROWS

The 'DOUBLE' option is available as a procedure option in PROC SQL. Specifying 'DOUBLE' enables PROC SQL to insert a blank line between every row of output. ('NODOUBLE' is the default)

This is especially helpful in providing a visual separation in cases where one or more line flows or wraps in the output. The example below shows an output without the 'DOUBLE' option being turned on.

```
PROC SQL;
SELECT *
FROM SALES;
QUIT;
```

The SAS System						
FIRSTNAME	LASTNAME	CUSTOMER_ID	PRODUCT_ID	UNITS	REVUNIT	STATE
Mary	Williams	10000000	201	4	10	VA
John	Smith	10000001	202	6	20	CT
Kim	Wells	10000002	203	2	50	VA
Jennifer	Crown	10000003	201	9	10	VA
Charlotte	McKay	10000004	206	20	5	VA
Sean	Kingston	10000005	202	5	20	GA
Cary	Thomas	10000006	205	15	10	FL
Tom	King	10000007	203	4	50	NC
Joe	Harrison	10000008	201	3	10	NC
Jim	Wards	10000008	205	3	10	GA

When 'DOUBLE' is added as an option, the output below displays a blank line between each row.

```
PROC SQL DOUBLE;
SELECT *
FROM SALES
QUIT;
```

The SAS System						
FIRSTNAME	LASTNAME	CUSTOMER_ID	PRODUCT_ID	UNITS	REVUNIT	STATE
Mary	Williams	10000000	201	4	10	VA
John	Smith	10000001	202	6	20	CT
Kim	Wells	10000002	203	2	50	VA
Jennifer	Crown	10000003	201	9	10	VA
Charlotte	McKay	10000004	206	20	5	VA
Sean	Kingston	10000005	202	5	20	GA
Cary	Thomas	10000006	205	15	10	FL
Tom	King	10000007	203	4	50	NC
Joe	Harrison	10000008	201	3	10	NC
Jim	Wards	10000008	205	3	10	GA

DISPLAYING ROW NUMBERS

The SQL procedure option –NUMBER displays the row number for every line under the column header ROW. This is very helpful in knowing how many rows the output actually has.

```
PROC SQL NUMBER;
SELECT *
FROM SALES
QUIT;
```

The SAS System							
Row	FIRSTNAME	LASTNAME	CUSTOMER_ID	PRODUCT_ID	UNITS	REVUNIT	STATE
1	Mary	Williams	10000000	201	4	10	VA
2	John	Smith	10000001	202	6	20	CT
3	Kim	Wells	10000002	203	2	50	VA
4	Jennifer	Crown	10000003	201	9	10	VA
5	Charlotte	McKay	10000004	206	20	5	VA
6	Sean	Kingston	10000005	202	5	20	GA
7	Cary	Thomas	10000006	205	15	10	FL
8	Tom	King	10000007	203	4	50	NC
9	Joe	Harrison	10000008	201	3	10	NC
10	Jim	Wards	10000008	205	3	10	GA

RESET OPTION

To revert back to a single space output and no row numbers, the RESET option with NODOUBLE and NONUMBER can be specified. The RESET statement with or without options reestablishes each options original settings. As you can see in the example below, specifying the RESET option eliminates the row numbers as well as the double spacing between rows.

```
PROC SQL;
RESET NODOUBLE NONUMBER;
QUIT;
```

The SAS System

FIRSTNAME	LASTNAME	CUSTOMER_ID	PRODUCT_ID	UNITS	REVUNIT	STATE
Mary	Williams	10000000	201	4	10	VA
John	Smith	10000001	202	6	20	CT
Kim	Wells	10000002	203	2	50	VA
Jennifer	Crown	10000003	201	9	10	VA
Charlotte	McKay	10000004	206	20	5	VA
Sean	Kingston	10000005	202	5	20	GA
Cary	Thomas	10000006	205	15	10	FL
Tom	King	10000007	203	4	50	NC
Joe	Harrison	10000008	201	3	10	NC
Jim	Wards	10000008	205	3	10	GA

CONCATENATING COLUMNS

It is possible to concatenate two columns in the PROC SQL SELECT Statement. The example below shows the concatenation of FIRSTNAME and LASTNAME into a new variable - NAME

```
PROC SQL;
SELECT FIRSTNAME || LASTNAME AS NAME
FROM SALES
```

The SAS System

NAME

Mary	Williams
John	Smith
Kim	Wells
Jennifer	Crown
Charlotte	McKay
Sean	Kingston
Cary	Thomas
Tom	King
Joe	Harrison
Jim	Wards

CONCATENATING CHARACTER STRINGS

Constants can be inserted within Character Strings as shown below. If you would like to space out FIRSTNAME and LASTNAME, then a blank space can be inserted between them. This makes the output look cleaner.

```
PROC SQL;
SELECT FIRSTNAME || " " || LASTNAME
FROM SALES
```

The SAS System

Mary	Williams
John	Smith
Kim	Wells
Jennifer	Crown
Charlotte	McKay
Sean	Kingston
Cary	Thomas
Tom	King
Joe	Harrison
Jim	Wards

Constants can be inserted within the two variables. In the example below, “,” is being inserted and used as a separator between FIRSTNAME and LASTNAME.

```
PROC SQL;
SELECT FIRSTNAME || " , " || LASTNAME
FROM CUSTOMER_SALES
```

The SAS System		
Mary	,	Williams
John	,	Smith
Kim	,	Wells
Jennifer	,	Crown
Charlotte	,	McKay
Sean	,	Kingston
Cary	,	Thomas
Tom	,	King
Joe	,	Harrison
Jim	,	Wards

REMOVING LEADING AND TRAILING BLANKS WHILE INSERTING TEXT AND CONSTANTS

The CATX function removes leading and trailing blanks. The following example shows how this function uses “,” as a separator between FIRSTNAME and LASTNAME and removes any trailing blanks.

```
PROC SQL;
SELECT CATX (",", FIRSTNAME, LASTNAME) AS NAME
FROM SALES
```

The SAS System	
NAME	
Mary	Williams
John	Smith
Kim	Wells
Jennifer	Crown
Charlotte	McKay
Sean	Kingston
Cary	Thomas
Tom	King
Joe	Harrison
Jim	Wards

DISPLAYING COLUMN HEADERS FOR DERIVED AND CALCULATED FIELDS

PROC SQL allows you to assign column headers or aliases to derived columns as well as calculated columns. In the example below, the SUM function is aggregating REVUNIT and the column header is TOTREV. The same applies to UNITS which are aggregated and named as TOTUNITS.

The 'Calculated' function is extremely helpful as it lets you use the calculated fields like TOTREV and TOTUNITS to calculate a third field- REVUNITS

```
PROC SQL;
SELECT PRODUCT_ID,
SUM (REVUNIT) AS TOTREV,
SUM (UNITS) AS TOTUNITS,
CALCULATED TOTREV/CALCULATED TOTUNITS AS REVUNIT
FROM SALES GROUP BY 1;
quit;
```

PRODUCT_ID	TOTREV	TOTUNITS	REVUNIT
201	30	16	1.875
202	40	11	3.636364
203	100	6	16.66667
205	20	18	1.111111
206	5	20	0.25

USING LABELS AND FORMATS

The output above can be further refined by adding labels and formats. In the example below, a title and footnote option has been added. The derived fields- REVUNIT is formatted using the Dollar8.2 shown below. By using the LABEL option, you can create a new column header- TOTAL REVENUE. Similarly for UNITS, a format of Comma8.0 is applied to add comma's to this field. By specifying TOTAL UNITS in the Label option, the column header is now changed.

```
TITLE 'PRODUCT REVENUE REPORT';
FOOTNOTE 'PREPARED BY MARKETING DEPT.';

PROC SQL;
SELECT PRODUCT_ID,
SUM (REVUNIT) AS TOTAL_REV FORMAT DOLLAR8.2 LABEL 'TOTAL REVENUE',
SUM (UNITS) AS TOTAL_UNITS FORMAT COMMA8.0 LABEL 'TOTAL UNITS',
CALCULATED TOTAL_REV/CALCULATED TOTAL_UNITS AS REVUNIT FORMAT DOLLAR8.2 LABEL '
REVENUE PER UNIT'
FROM SALES
GROUP BY 1
ORDER BY 1;
QUIT;
```

PRODUCT REVENUE REPORT			
PRODUCT_ID	TOTAL REVENUE	TOTAL UNITS	REVENUE PER UNIT
201	\$30.00	16	\$1.88
202	\$40.00	11	\$3.64
203	\$100.00	6	\$16.67
205	\$20.00	18	\$1.11
206	\$5.00	20	\$0.25
PREPARED BY MARKETING DEPT.			

CONCLUSION

These are some examples of how PROC SQL can aid in enhancing your output. SAS programmers being unfamiliar with PROC SQL's output capabilities resort to using DATA STEPS or reporting functions to create beautiful outputs. I hope that paper has been successful in illustrating the virtues of PROC SQL output formatting methods.

REFERENCES

PROC SQL- Beyond the Basics- Kirk Paul Lafler

CONTACT INFORMATION

Kalyani Chaturvedula
Health Management Corporation
8831 Park Central Drive, Suite 100
Richmond, VA 23227
Phone- 804-662-8788
Email- kchaturvedula@choosehmc.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.