Paper SAS 3344-2019

The Shape of SAS® Code

Charu Shankar, SAS® Institute Inc.

ABSTRACT

There are many languages that co-exist in the ecosystem of your SAS® toolbox. This Hands-On Workshop teaches you how to use four SAS languages - Base SAS, PROC SQL, Perl language elements, and the SAS® Macro Language - to help you manipulate and investigate your data. Learn to leverage these powerful languages to check your data with simple, yet elegant techniques such as Boolean logic in PROC SQL, operators such as the SOUNDS-LIKE operator in the DATA step and PROC step, functions such as the SCAN function in the DATA step, efficient checking of your data with Perl regular expressions, and last but not least, the amazing marriage between PROC SQL and the SAS Macro Language to hold data you just found in a variable that you can use over and over again. This workshop focuses on coding techniques for data investigation and manipulation using Base SAS.

DATA USED IN THIS PRESENTATION

TABLE NAME	DETAILS
Sashelp.demographics	The Sashelp.demographics data set provides the 2004 revision of data derived from world population prospects. The data set contains 197 observations.
Sashelp.mwelect	The Sashelp.mwelect data set provides midwest electrical supply monthly sales by product group. The data set contains 11,296 observations.
Pflugerville	Code to build the dataset is provided in the References section of this paper

Table 1. Details about the data sets used in this Hands-on workshop

1. INTRODUCTION

When the author set out to share the power of different languages in the SAS® toolkit, her initial targeted audience was the novice user. What she didn’t fully realize was the value of this topic to experienced users as well. For example, there is value to the user who came up to the author after her presentation to share that they were now able to view PROC SQL with a Boolean angle, and the advanced user also benefitted by learning about PERL regular expressions in SAS. Clearly, this paper won’t be able to teach you every single nuance of these majestic languages in the short timeframe of a hands-on workshop. The purpose of this paper is to synthesize & distill each language to its best strengths. Whether you are a novice user or an experienced SAS coder, the hope is for you to get something new out of this paper. Comments and feedback are always appreciated.

The goal of this Hands-on workshop is to understand the shape of 4 SAS® languages. Four business scenarios have been developed in this Hands-on workshop to practice the four SAS® languages:

Business Scenario 1 – Using the most misspelled city in Texas, this scenario will discuss the fundamental concepts of the SAS data step and how to filter data using the WHERE clause with the Sounds-like operator and the Contains Operator. It will also reveal where the sounds-like operator works well and where the contains operator may be a better fit.

Business Scenario 2 – Using the SASHELP. Mwelect dataset, this scenario will discuss how to find a pattern using the language of Perl.

Business Scenario 3 – Using the SASHELP. demographics dataset, this scenario will discuss how to create a macro variable to find the country with the highest population in 2004.

Business Scenario 4 – Using the SASHELP. demographics dataset, this scenario will discuss how to use Boolean logic in PROC SQL to obtain population range counts by region.

REVIEW OF SAS PROGRAMS

A SAS program is a sequence of one or more steps.

DATA steps typically build and manipulate SAS data sets. So, we can reference the DATA step as the **Builder**. Keeping the building capability in mind helps while writing data step code as the executable statements are, for the most part processed in sequence.

PROC steps typically process SAS data sets to generate reports and graphs, and to manage data. So, we can reference the PROC Step as the **Analyzer**. Keeping the analytic capability in mind helps while writing PROC steps as the order of statements is usually not important.

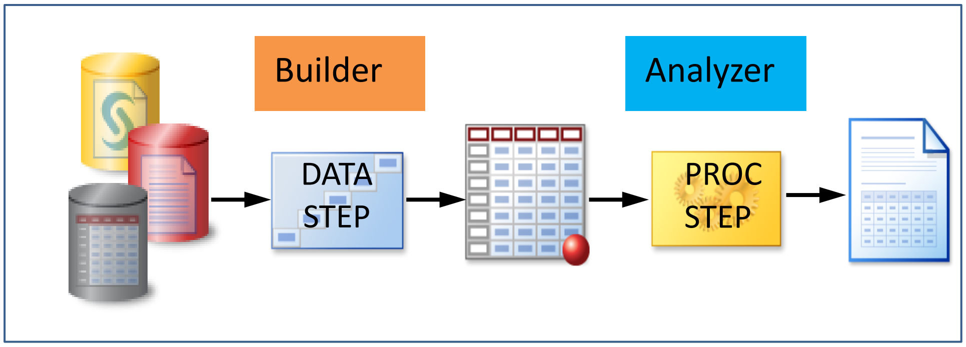


Figure 1. The role of the DATA STEP vs the PROC STEP

DATA EXTRACTION TOOL – LIBRARY TO READ SAS DATASETS

A SAS data library is a collection of one or more SAS files that are recognized by SAS and can be referenced and stored as a unit. A library is simply an alias, a pointer or a reference pointing to a physical location on your computer, e.g. a folder on your C drive.

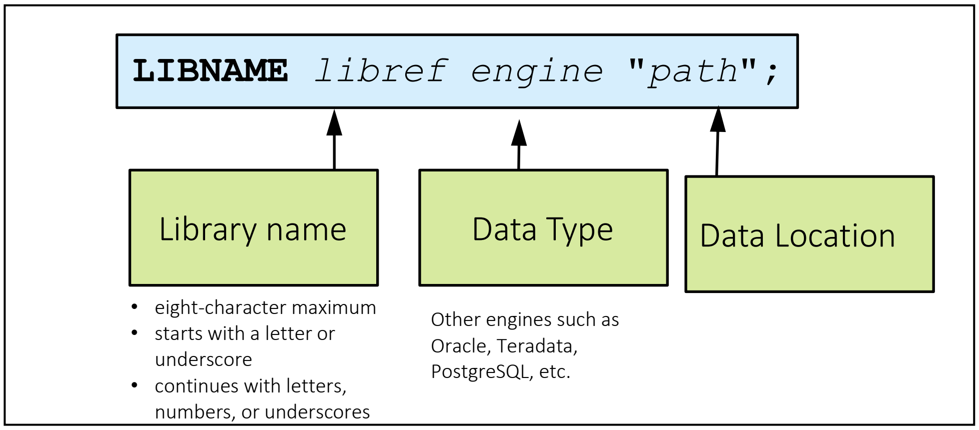

LIBNAME libref engine "path";
Library name
Data Type
Data Location
• eight-character maximum
• starts with a letter or underscore
• continues with letters, numbers, or underscores
Other engines such as Oracle, Teradata, PostgreSQL, etc.

Figure 2. The Libname Statement

Libname Code

```
libname sgf "c:\HOW\shankar\data’;
```

2. SHAPE 1 - THE SAS DATA STEP – THE MANIPULATOR

BUSINESS SCENARIO 1 – FIND ALL PFLUGERVILLE RECORDS

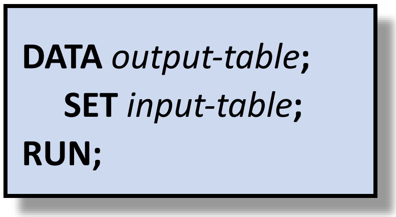
You have been tasked to find all records for the city of Pflugerville in Texas. However, there’s a small problem. The city’s name is so misspelled. How can you filter your data?

Figure 3. Map of Texas with Pflugerville

USING THE DATA STEP TO CREATE A SAS DATASET

The Data Step is a powerful tool to create, clean, and prepare your data. These are some of the tasks it can perform:

- Filter rows and columns
- Compute new columns
- Conditionally process data

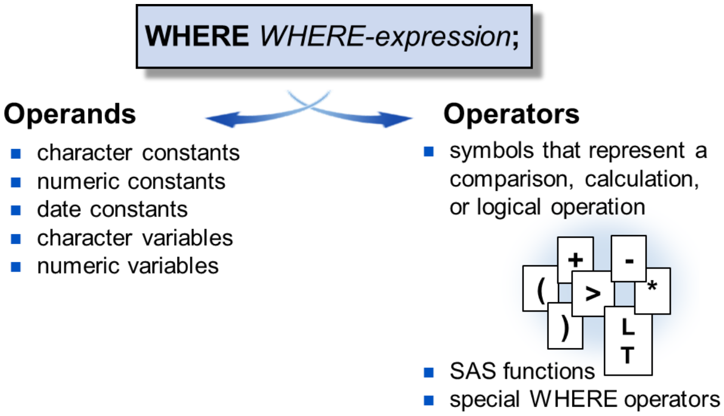


```
DATA output-table;
   SET input-table;
RUN;
```

Figure 4. Data Step Code

FILTER DATA WITH THE WHERE CLAUSE

The WHERE expression defines the condition (or conditions) for selecting observations.



WHERE *WHERE-expression*;

Operands

- character constants
- numeric constants
- date constants
- character variables
- numeric variables

Operators

- symbols that represent a comparison, calculation, or logical operation
- SAS functions
- special WHERE operators

Figure 5. The Where Expression

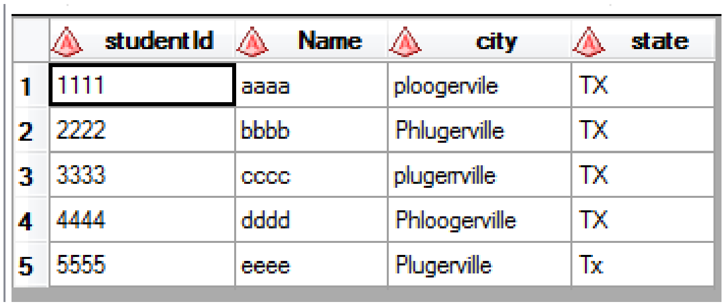
THE SOUNDS LIKE OPERATOR

The sounds-like operator =* is very useful when fuzzy matching of character values is needed. It matches character strings based on their phonetic values. The sounds-like operator is based on the SOUNDEX algorithm for identifying words that sound alike.

<u>Data step Code</u>

```
data citytypo;
       set sgf.citysoundslike;
            where city=*'Pflugerville';
 run;
```

Data step Results



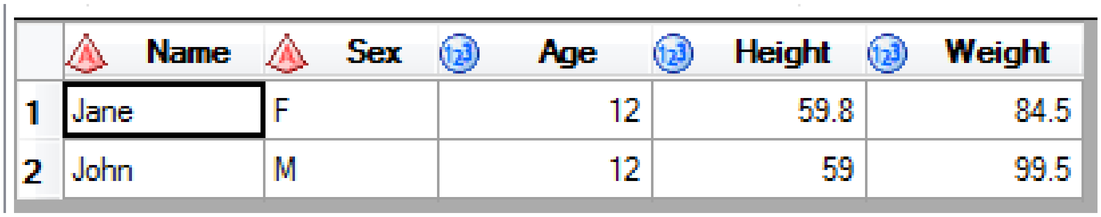
	studentId	Name	city	state
1	1111	aaaa	ploogervile	TX
2	2222	bbbb	Phlugerville	TX
3	3333	cccc	plugerrville	TX
4	4444	dddd	Phloogerville	TX
5	5555	eeee	Plugerville	Tx

However, the sounds like operator does not work in every instance. Observe what happens when we try to find all students named John in the SASHELP.CLASS data set.

Data Step Code

```
data sgf.john;
      set sashelp.class;
            where name =*'John';
run;
```

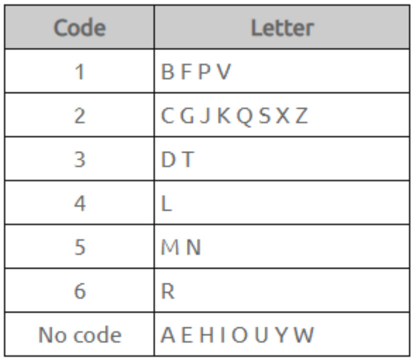
Data Step Results



	Name	Sex	Age	Height	Weight
1	Jane	F	12	59.8	84.5
2	John	M	12	59	99.5

THE SOUNDEX ALGORITHM

Why did our search for John return Jane as well? Consider the Soundex Algorithm. Soundex is an indexing system that translates a name into a 4-digit code. The advantage of Soundex is its ability to locate names by the way they sound, rather than by exact spelling.



Code	Letter
1	B F P V
2	C G J K Q S X Z
3	D T
4	L
5	M N
6	R
No code	A E H I O U Y W

Figure 6. The Soundex Algorithm Chart

The steps used by SOUNDEX to derive the phonetic equivalent of a character string are as follows:

a) Retain the first letter of the character string.

b) Discard the letters A E H I O U W Y.

c) Assign a numeric value to the following consonants:

   1. B F P V
   2. C G J K Q S Z
   3. D T
   4. L
   5. M N
   6. R

d) Discard all duplicate classification values if they are adjacent. That is, DT results in a single value of 3, and NN results in a single value of 5).

Here is the answer to our puzzle as to why a search for John pulled up Jane as well

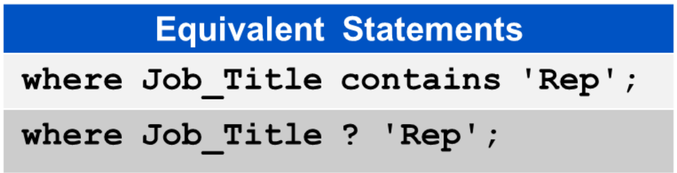
JOHN	Soundex	JANE	Soundex
J	J	J	J
O	discarded	A	discarded
H	discarded	N	5
N	5	E	discarded

Figure 7. John sounds like Jane

THE CONTAINS OPERATOR

The *CONTAINS operator* selects observations that include the specified substring.

- ? can be used instead of the mnemonic.
- The position of the substring within the variable's values is not important.
- Comparisons made with the CONTAINS operator are case sensitive.



Equivalent Statements

```
where Job_Title contains 'Rep';
where Job_Title ? 'Rep';
```

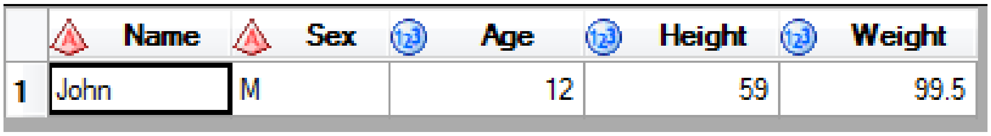
Figure 8. The Contains Operator

Let's see what happens when we use the contains operator.

Data Step Code

```
data sgf.thisisJohn;
        set sashelp.class;
              where name contains "John";
run;
```

Data Step Results

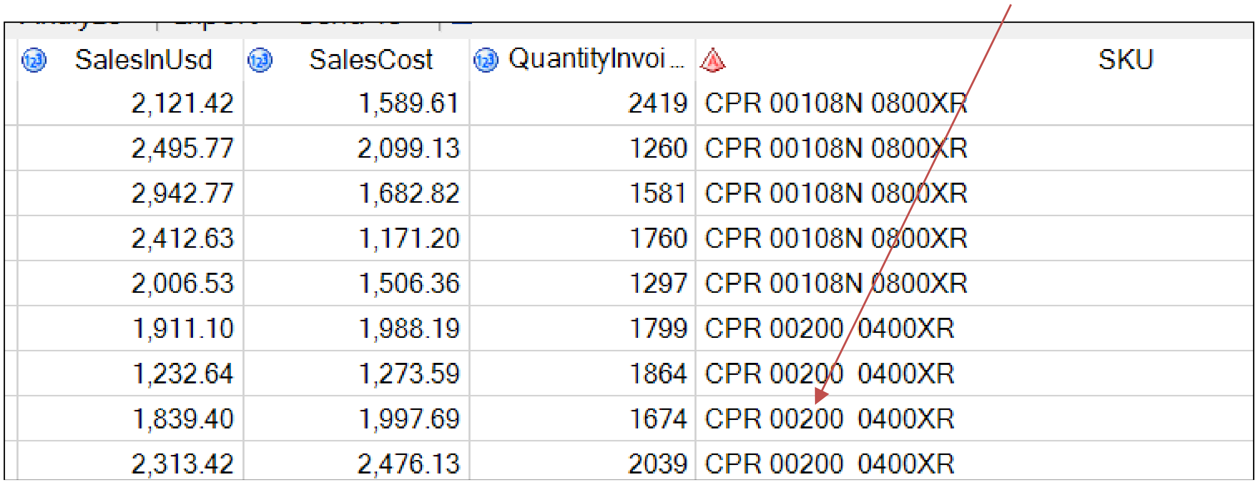


	Name	Sex	Age	Height	Weight
1	John	M	12	59	99.5

3. SHAPE 2 - PERL – MATCH A PATTERN

BUSINESS SCENARIO 2 - MATCH A PATTERN

We have been tasked to isolate all SKUs that match this pattern 'DDD ddddd '.



SalesInUsd	SalesCost	QuantityInvoi...	SKU
2,121.42	1,589.61	2419	CPR 00108N 0800XR
2,495.77	2,099.13	1260	CPR 00108N 0800XR
2,942.77	1,682.82	1581	CPR 00108N 0800XR
2,412.63	1,171.20	1760	CPR 00108N 0800XR
2,006.53	1,506.36	1297	CPR 00108N 0800XR
1,911.10	1,988.19	1799	CPR 00200 0400XR
1,232.64	1,273.59	1864	CPR 00200 0400XR
1,839.40	1,997.69	1674	CPR 00200 0400XR
2,313.42	2,476.13	2039	CPR 00200 0400XR

Figure 9. Matching a pattern

USING PERL TO MATCH A PATTERN

Perl was designed specifically for text processing. The 1990s saw the growth of the World Wide Web. It also saw the rise of text-based information during that period. As one of the languages very capable of text manipulation and undergoing rapid development, Perl was suited to the task at hand. As a result, it became a very popular web programming language, even being referred to as the ‘duct-tape of the Web’.

Perl is a very high-level language. That means that the code is quite dense. A Perl program might be around 30% to 70% as long as the corresponding program in C.

PERL IN SAS

Perl regular expressions were added to SAS in Version 9. SAS regular expressions (similar to Perl regular expressions but using a different syntax to indicate text patterns) have been around since version 6.12, but many SAS users are unfamiliar with either SAS or Perl regular expressions.

Because SAS already has such a powerful set of string functions, you might wonder why you need regular expressions. Many of the string processing tasks can be performed either with the traditional character functions or regular expressions. However, regular expressions can sometimes provide a much more compact solution to a complicated string manipulation task.

MATCHING A PATTERN

Since the backslash, forward slash, parentheses and several other characters have special meaning in a regular expression, you may wonder, how do you search a string for a \ character or a left or right parenthesis? You do this by preceding any of these special characters with a backslash character (in Perl jargon called an escape character). So, to match a \ in a string, you code two backslashes like this: \\. To match an open parenthesis, you use \(.

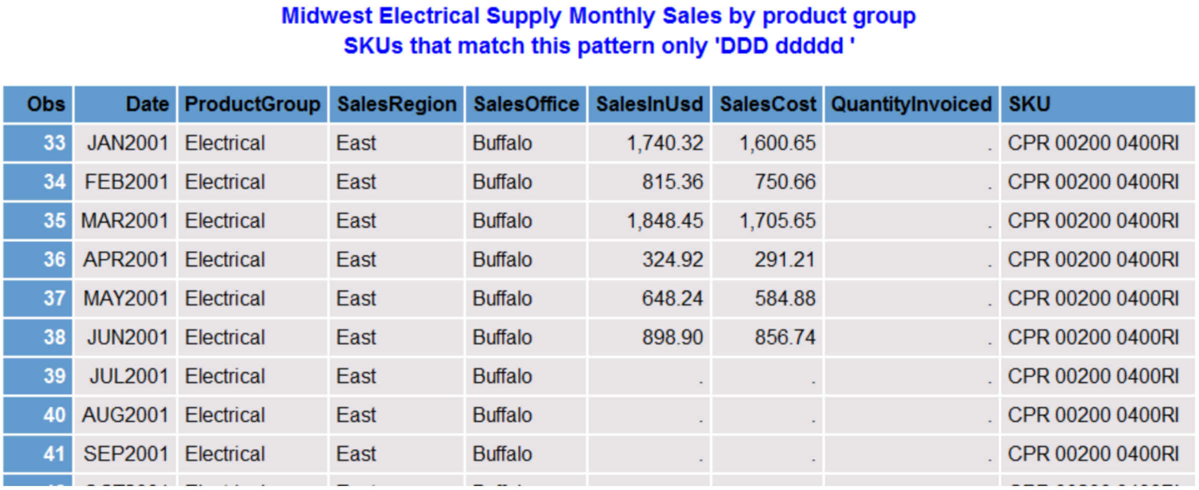
/ delimiters
\(matches an open paranthesis
\D matches a non-digit
\d matches a digit
\s matches a space
{n,m} Matches the previous subexpression n or more times, but no more than m
\) matches a closed paranthesis

<u>Perl Code</u>

```
title 'Midwest Electrical Supply Monthly Sales by product group';
title2 "SKUs that match this pattern only 'DDD ddddd ";

proc print data=sashelp.mwelect;
        where prxmatch(("/\D{3}\s\d{5}\s/"), SKU) > 0;
run;
```

Perl Results



Midwest Electrical Supply Monthly Sales by product group
SKUs that match this pattern only 'DDD ddddd '

Obs	Date	ProductGroup	SalesRegion	SalesOffice	SalesInUsd	SalesCost	QuantityInvoiced	SKU
33	JAN2001	Electrical	East	Buffalo	1,740.32	1,600.65	.	CPR 00200 0400RI
34	FEB2001	Electrical	East	Buffalo	815.36	750.66	.	CPR 00200 0400RI
35	MAR2001	Electrical	East	Buffalo	1,848.45	1,705.65	.	CPR 00200 0400RI
36	APR2001	Electrical	East	Buffalo	324.92	291.21	.	CPR 00200 0400RI
37	MAY2001	Electrical	East	Buffalo	648.24	584.88	.	CPR 00200 0400RI
38	JUN2001	Electrical	East	Buffalo	898.90	856.74	.	CPR 00200 0400RI
39	JUL2001	Electrical	East	Buffalo	.	.	.	CPR 00200 0400RI
40	AUG2001	Electrical	East	Buffalo	.	.	.	CPR 00200 0400RI
41	SEP2001	Electrical	East	Buffalo	.	.	.	CPR 00200 0400RI

4. SHAPE 3 - MACRO – AUTOMATE FIND AND REPLACE

BUSINESS SCENARIO 3 - USING MACRO VARIABLES TO AUTOMATE

We have been asked to find the country with the highest population in 2004. Since we would like to feed this data to another report, we will store the country name and the population count in 2 macro variables that we can reuse over and over again.

Figure 10. Using macro variables to automate results

CREATING AND REFERENCING MACRO VARIABLES

The process of creating macro variables is simple. First, we create the macro variables and then submit code to create a report using the defined macro variables.

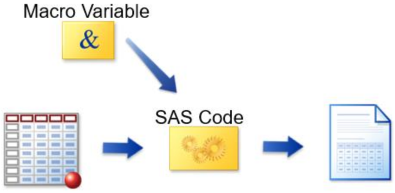
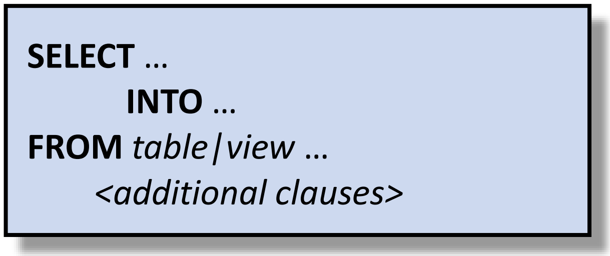


Figure 11. Creating and referencing macro variables

CREATING MACRO VARIABLES WITH PROC SQL

In PROC SQL, use an INTO clause to create macro variables and assign a value to them or to update existing macro variable values. The INTO clause must follow the SELECT clause.

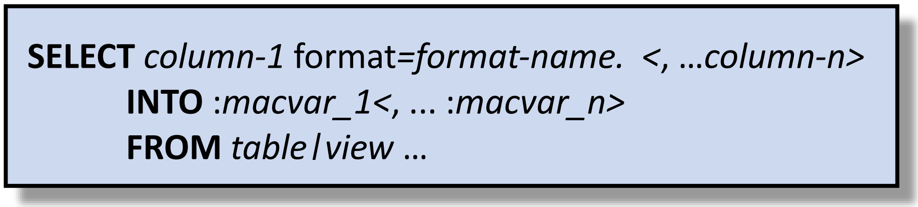


```
SELECT …
        INTO …
FROM table|view …
     <additional clauses>
```

Figure 12. Creating macro variables with PROC SQL

CREATING MACRO VARIABLES: SYNTAX 1

Syntax 1 places values from the *first row* returned by an SQL query into one or more macro variables. Data from additional rows returned by the query is ignored. The value from the first column in the SELECT clause is placed in the first macro variable listed in the INTO clause, and so on.



```
SELECT column-1 format=format-name.  <, …column-n>
        INTO :macvar_1<, … :macvar_n>
        FROM table|view …
```

Figure 13. Creating macro variables: Syntax 1

MACRO TO GRAB COUNTRY WITH THE HIGHEST POPULATION

Macro Code

```
proc sql noprint;
        select pop format comma13. , name
                into : Maxpop , : country
                        from sashelp.demographics
                                order by 1 descending;
        %put &=country;
        %put &=maxpop;
```

Macro Results in the log

```
45              %put &=country;
COUNTRY=CHINA
46              %put &=maxpop;

MAXPOP=1,323,344,591
```

5. SHAPE 4 – PROC SQL – SPEAK ELEGANT BOOLEAN

BUSINESS SCENARIO 4 - USING BOOLEAN LOGIC TO COUNT ROWS

We would like to obtain population range counts by region.

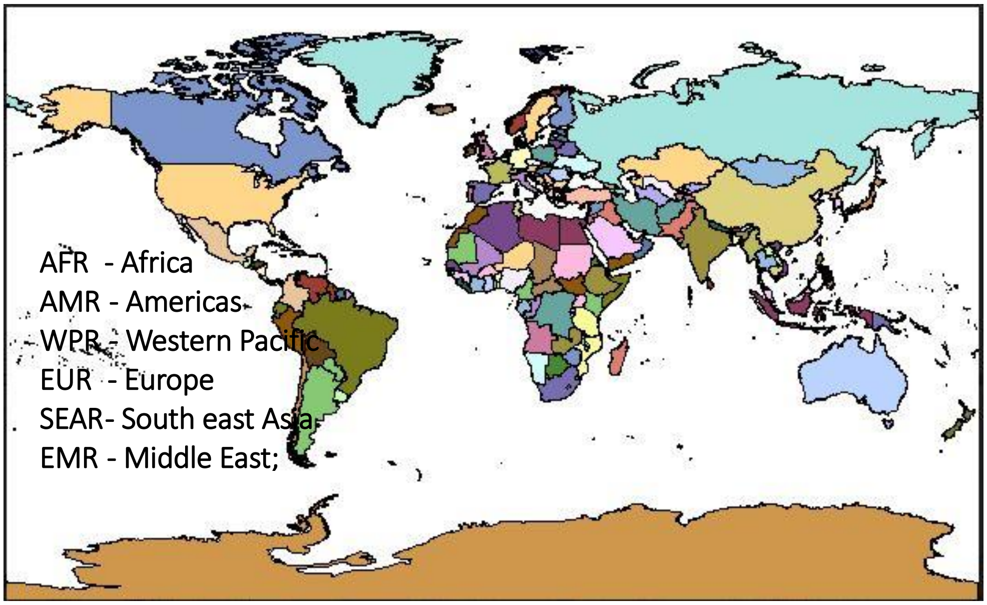


Figure 14. World map with region names

BOOLEAN LOGIC IN PROC SQL

Everything in the digital world can be broken down into 1 or 0, or rather Yes or No. We will take advantage of this Boolean capacity to get population counts by each region. First, it would be useful to understand the syntax order of PROC SQL.

PROC SQL SYNTAX ORDER

The specified order of the clauses below, within the SELECT statement is required.

```
PROC SQL;
SELECT object-item <, ...object-item>
            FROM from-list
      <WHERE sql-expression>
      <GROUP BY object-item <, … object-item >>
      <HAVING sql-expression>
      <ORDER BY order-by-item <DESC> <, …order-by-
item>>;
QUIT;
```

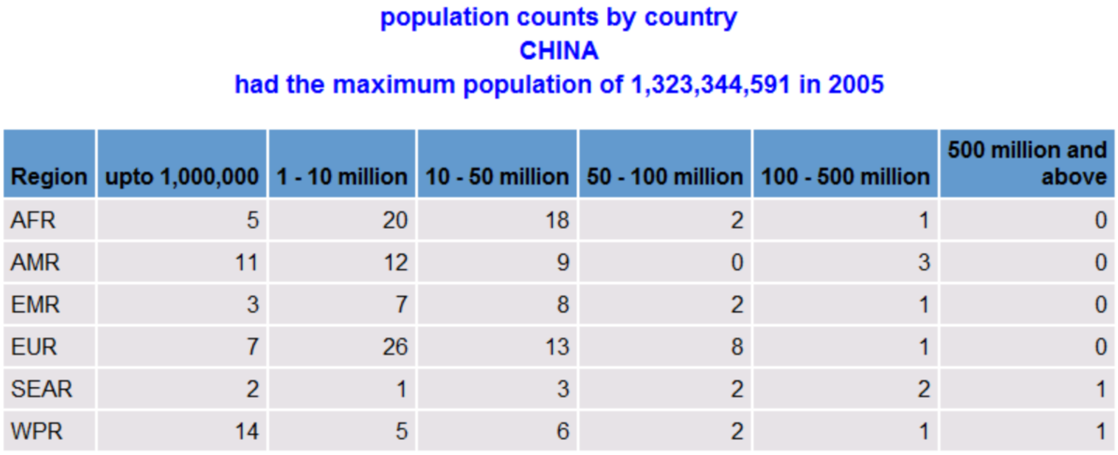
Figure 15. PROC SQL syntax order

USING PROC SQL TO GET COUNTS

Proc Sql Code

```
title "population counts by country";
title2 "&country";
title3 "had the maximum population of &maxpop in 2005";
proc sql;
        select  region,
               sum(pop <= 1000000) 'upto 1,000,000',
               sum(pop between   1000001 and    10000000)   '1 -  10 million',
               sum(pop between   10000001 and    50000000)   '10 -  50 million',
               sum(pop between   50000001 and   100000000)   '50 - 100 million',
               sum(pop between  100000001 and   500000000)  '100 - 500 million',
               sum(pop >  500000001 )  '500 million and above'
               from sashelp.demographics
                  group by 1
        ;
quit;
```

Proc Sql Results



population counts by country
CHINA
had the maximum population of 1,323,344,591 in 2005

Region	upto 1,000,000	1 - 10 million	10 - 50 million	50 - 100 million	100 - 500 million	500 million and above
AFR	5	20	18	2	1	0
AMR	11	12	9	0	3	0
EMR	3	7	8	2	1	0
EUR	7	26	13	8	1	0
SEAR	2	1	3	2	2	1
WPR	14	5	6	2	1	1

6. REFERENCES

"How the data step works: A Basic Introduction". Support.sas.com website. http://support.sas.com/documentation/cdl/en/basess/58133/HTML/default/viewer.htm#a001290590.htm

Kuligowski, Andrew T.; Shankar, Charu. "Know thy data: Techniques for data exploration". Proceedings of SAS Global Forum 2013, San Francisco, CA. http://support.sas.com/resources/papers/proceedings13/145-2013.pdf

SAS processing: compile & execute phase
Shankar, Charu. November 2011. "Retail therapy the SAS way". Blogs.sas.com website. https://blogs.sas.com/content/sastraining/2011/11/03/retail-therapy-the-sas-way/

Sounds Like Operator.
Shankar, Charu. January 2011. "A new year's resolution that sounds like more fun than a spinning class". Blogs.sas.com website. https://blogs.sas.com/content/sastraining/2011/01/12/a-new-years-resolution-that-sounds-like-more-fun-than-a-spinning-class/

Shankar, Charu. January 2011. "Find your data pattern with PERL". Blogs.sas.com website. https://blogs.sas.com/content/sastraining/2011/01/24/find-your-data-pattern-with-perl/

"SAS® 9 PERL regular expression cheat sheet". Support.sas.com website. https://support.sas.com/rnd/base/datastep/perl_regexp/regexp-tip-sheet.pdf

"SEARCHING USING Soundex codes". The Spreadsheet Page http://spreadsheetpage.com/index.php/tip/searching_using_soundex_codes/

Hadden, Louise S. "Wow! You Did That Map with SAS/GRAPH®?". Proceedings of SAS Global Forum 2009, Washington, DC. https://support.sas.com/resources/papers/proceedings09/215-2009.pdf

SAS Macro INTO clause.
"SAS® 9.4 Macro Language: Reference, Fifth Edition". Support.sas.com website. https://go.documentation.sas.com/?docsetId=mcrolref&docsetTarget=n1y2jszlvs4hugn14nooftfrxhp3.htm&docsetVersion=9.4&locale=en

"SAS® 9.4 SQL Procedure User's Guide, Fourth Edition". Support.sas.com website. http://support.sas.com/documentation/cdl//en/sqlproc/69822/HTML/default/viewer.htm#titlepage.htm

Boolean: #1 SAS programing tip for 2012
Shankar, Charu. May 2012. "#1 SAS programming tip for 2012". Blogs.sas.com website. https://blogs.sas.com/content/sastraining/2012/05/10/1-sas-programming-tip-for-2012/

Shankar, Charu. April 2012. Go home on time with these 5 PROC SQL tips ". Blogs.sas.com website. https://blogs.sas.com/content/sastraining/2012/04/24/go-home-on-time-with-these-5-proc-sql-tips/

Code for building the Pflugerville dataset

```
data sgf.misspelled;
        length studentId $4 Name $7 city $20 state $2;
        input studentid $ name $ city $ state $;
        datalines;
1111 aaaa ploogervile TX
2222 bbbb Phlugerville TX
3333 cccc plugerrville TX
4444 dddd Phloogerville TX
5555 eeee Plugerville Tx
;

run;
```

7. ACKNOWLEDGMENTS

The author is grateful to the many SAS users that have entered her life. Each User has either asked or answered a question. This in turn gave the author the impetus to research and study new ways to express the wonderful shape of SAS code. Sometimes a user appeared in the form of a teacher to show her the many ways in which to express the Shape of SAS code. The users are too many to thank individually so this is a thank you to every single user who has touched her life. She is grateful to the SAS global forum committee for inviting her to present a paper. She would also like to express her gratitude to her manager, Stephen Keelan without whose support and permission, this paper would not be possible.

8. CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Charu Shankar
SAS Institute Canada, Inc.
Charu.shankar@sas.com
https://blogs.sas.com/content/author/charushankar/