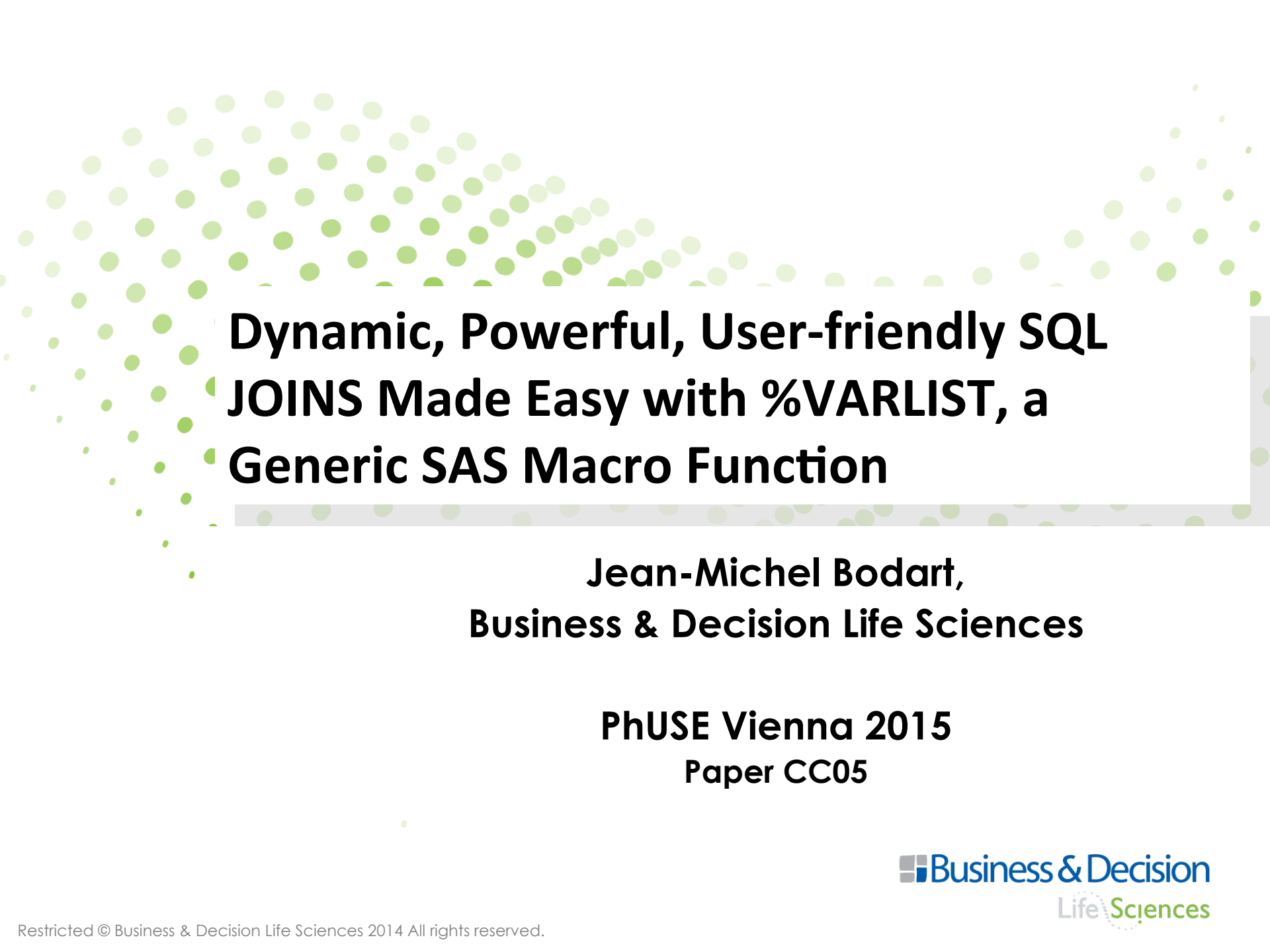




Dynamic, Powerful, User-friendly SQL JOINS Made Easy with %VARLIST, a Generic SAS Macro Function

**Jean-Michel Bodart,
Business & Decision Life Sciences**

**PhUSE Vienna 2015
Paper CC05**

Table of contents

	Introducing %VARLIST	• User-written Macro-Function • SAS Utility tool
	Inputs	• Data • Var
	Return Value Format	• Sep • Pattern • From
	SQL JOINS	• Basic Outer Join • Generalized Framework
	Conclusions & Questions	

INTRODUCTION: a Macro-Function is...

- a **macro**, with
- some **arguments**, and
- a **return a value** which can
 - be printed
 - assigned to a macro-variable
 - passed to another function
 - **become part of non-macro code**

```
%SUBSTR
%SUBSTR(&text, 3)
%put %SUBSTR(&text, 3);
%let word=%SUBSTR(&text, 3);
%UPCASE(%SUBSTR(&text, 3));
TITLE "The
%UPCASE(%SUBSTR(&text, 3)).";
```

We can **create our own** macro-functions:

```
%macro nicedate(date);
  %local datenum;
  %let datenum = %sysfunc(inputn(&date, ANYDTDTE.)) ;
  first thing on %sysfunc(putn(&datenum, WEEKDATX.))
%mend;
```

```
%put I ran this example %nicedate(&sysdate9);
```

```
I ran this example first thing on Thursday, 10
September 2015
```

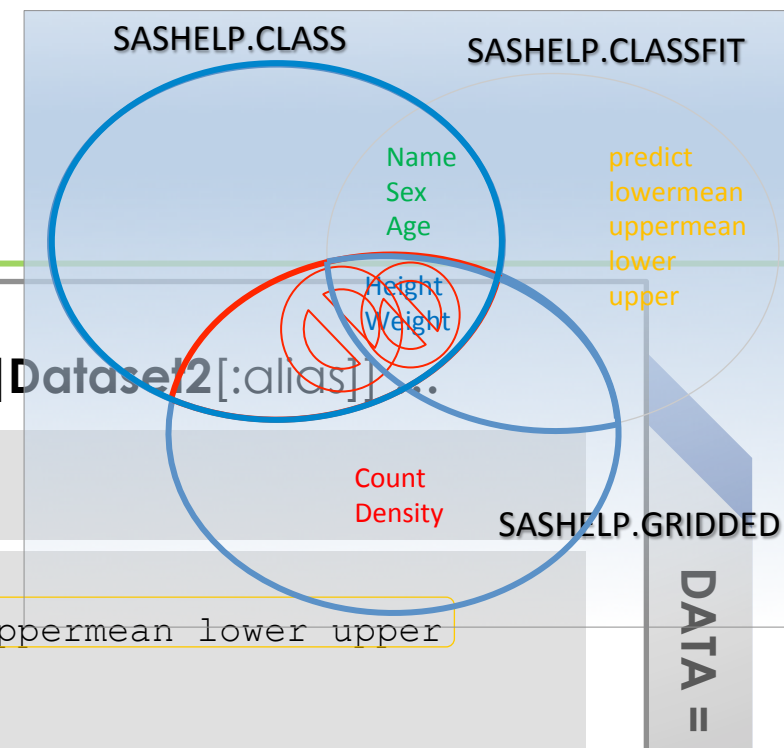
INTRODUCTION: the Macro-Function %VARLIST()

A **utility tool**

- to process **lists of variables**
 - passed as **literals**
 - retrieved dynamically from **dataset(s)** / **macro-variables**
 - and combine them using **operators**: union, intersection, exclusion, exclusive OR
- and **return** a combination of:
 - variable **name** (+ dataset name) / **attribute(s)**
 - **arbitrary characters or symbols**
 - **separated** by: space, comma, semicolon, custom separators
- **Code, FAQ, Examples** : see PhUSE wiki page

http://www.phusewiki.org/wiki/index.php?title=SAS_macro-function_%25VARLIST

%VARLIST(): Parameters « in »



I. **Data** = [[Lib.]**Dataset1**[:alias] [[Op] [Lib.]**Dataset2**[:alias]] ...

```
1568 %put %VARLIST(data=SASHELP.CLASS);
Name Sex Age Height Weight
```

```
1569 %put %VARLIST(data=SASHELP.CLASSFIT);
Name Sex Age Height Weight predict lowermean uppermean lower upper
```

```
1570 %put %VARLIST(data=SASHELP.GRIDDED);
Height Weight Count Density
```

```
1571 %put %VARLIST(data=SASHELP.CLASS #union# SASHELP.GRIDDED);
Name Sex Age Height Weight Count Density
```

```
1572 %put %VARLIST(data=SASHELP.CLASS #not# SASHELP.GRIDDED);
Name Sex Age
```

```
1573 %put %VARLIST(data=SASHELP.CLASS #xor# SASHELP.GRIDDED);
Name Sex Age Count Density
```

```
1574 %put %VARLIST(data=SASHELP.CLASSFIT #intersect# SASHELP.GRIDDED); ;
Height Weight
```

```
1575 %put %VARLIST(data=SASHELP.CLASSFIT #intersect# SASHELP.GRIDDED
#xor# SASHELP.CLASS);
Name Sex Age
```

%VARLIST(): Parameters « in »

II. **Var** = [**Var1** [[**Op**] **Var2**] ... (default: **#all#**)

Variable References use:

- names: **Age Name Weight**
- num ranges: **XXX07-12 T1-20**
- numbers (positions): **#3# #2-4#**
- classes: **#all# #num# #char#**
- pre/suffixes: **Na: :ght CM:DT**
- ranges:
 - alphanum: **EXDOSE-EXDUR**
 - positional: **Sex--Height**

Variable Operators:

- **#not#** (or **#except#**)
- **#intersect#** (or **#and#**)
- **#xor#**

➔ especially useful to **dynamically process** variables list stored in **macro-variables**

Require Data=<dataset> for lookup

• PFERI Rep Exn: /[^]TRT\d\dN?\$/

```
1574 %put %VARLIST(data=SASHELP.CLASS, var=Weight Height BMI);  
Weight Height
```

```
1575 %put %VARLIST(data=SASHELP.CLASS, var=#num# #not# :ght);  
Age
```

VAR =

%VARLIST(): Parameters « out » / SQL

I. Sep = **Keyword(s)** | **Litteral(s)** [...] (default: **#space#**)

Keywords:

- **#space#** (or **#s#**): space character ()
- **#comma#** (or **#c#**): comma (,)
- **#commaspace#** (or **#cs#**, **#c##s#**): comma + space (,)
- **#semicolon#** (or **#scol#**, **#sc#**): semicolon (;)
- **#semicolons#** (or **#scols#**, **#scs#**, **#sc##s#**): semicolon + space (;)
- **#and#**, **#or#**: " and ", " or " (without quotes)

Literals (e.g.):

- **%str()**
- **%bquote(,)**
- **%bquote(,)**
- **%bquote(;)**
- **%bquote(;)**
- **%str(and), %str(or)**

```
1546 %let keyvars = STUDYID USUBJID AVISITN;
1547 proc sql;
1548     create table SV as
1549     select %VARLIST(data=ADSV, var=#1-6# AVIS:, sep=#cs#)
MPRINT(VARLIST):    STUDYID, USUBJID, SUBJID, SITEID, VISITNUM, VISIT,
AVISITN, AVISIT
1550     from ADSV(obs=200)
1551     order by %VARLIST(var=&keyvars, sep=#cs#);
MPRINT(VARLIST):    STUDYID, USUBJID, AVISITN
NOTE: Table WORK.SV created, with 200 rows and 8 columns.
```

SEP =

SQL

%VARLIST(): Parameters « out »

II. Pattern = Keyword(s) | Literal(s) [...] (default: #var#)

Keywords:

- #var#, #vtyp#, #vlen#, #vfmt#, #vinfmt#, #vlab#
- #vfmtdef#, #vinfmtdef#
- #vlabq#
- #vlenlabfmt#

Replaced (for each variable) respectively by:

- Variable **name**, attribute: **type** (N or C), **length**, **format**, **informat**, **label** (unquoted)
- Variable (**in**)**format** if defined, otherwise default (in)formats \$<length>. (char) and **Best12.** (num)
- **Quoted** version of variable **label**, or " " if empty
- Variable name and attributes (if defined) using syntax: <variable> [LENGTH=<length>] LABEL=<quoted label> [FORMAT=<format>] [INFORMAT=<informat>]

```
1578 %put %VARLIST (data=SASHELP.CLASS, var=Sex, pattern=#vtyp#);  
C
```

```
1579 %put %VARLIST (data=SASHELP.CLASS, var=Sex, pattern=#vlabq#);  
" "
```

```
1580 %put ATTRIB %VARLIST (data=SASHELP.ELECTRIC, var=AllPower,  
pattern=#vlenlabfmt#);  
ATTRIB AllPower length=8 label="All" format=COMMA10.
```

%VARLIST(): Parameters « out » / SQL Outer Joins

II. Pattern = Keyword(s) | Literal(s) [...] (default: #var#)

Keywords:

- #dsn#
- #dsa#

Replaced (for each variable) respectively by:

- Dataset name (first one that contains the variable), including Libname (if specified in *data=* parameter)
- Dataset alias if defined in parameter *data= [lib.]<dataset>:<alias>* (otherwise dataset name without Libname)

```
3091 proc sql;
3092     create table class2 as
3093     SELECT %VARLIST(data=SASHELP.CLASS #intersect# SASHELP.GRIDDED,
sep=#cs#, pattern=COALESCE(CLASS.#var#, GRIDDED.#var#) as #vlenlabfmt#),
MPRINT(VARLIST):    COALESCE(CLASS.Height, GRIDDED.Height) as Height length=8
label=" ", COALESCE(CLASS.Weight, GRIDDED.Weight) as Weight length=8 label=" "
3094     %VARLIST(data=SASHELP.CLASS #xor# SASHELP.GRIDDED, sep=#cs#,
pattern=#dsa#.#var#)
MPRINT(VARLIST):    CLASS.Name, CLASS.Sex, CLASS.Age, GRIDDED.Count,
GRIDDED.Density
3095     from SASHELP.CLASS FULL JOIN SASHELP.GRIDDED
3096     ON %VARLIST(data=SASHELP.CLASS #intersect# SASHELP.GRIDDED,
sep=#and#, pattern=CLASS.#var#=GRIDDED.#var#);
MPRINT(VARLIST):    CLASS.Height=GRIDDED.Height and CLASS.Weight=GRIDDED.Weight
NOTE: Table WORK.CLASS2 created, with 3619 rows and 7 columns.
```

PATTERN =

SQL Outer Joins

%VARLIST(): Parameters « out » / SQL Outer Joins

II. Pattern = Keyword(s) | Literal(s) [...] (default: #var#)

Keywords:

- #autoCOAL#
- #autoEQ#

Replaced (for each variable) respectively by:

- Full **COALESCE** () function call for variables in multiple datasets (specified in data= parameter), otherwise equivalent to pattern=#dsa#.#var#
- Equality condition of variables in multiple datasets (specified in data= parameter) as in SQL ON clause

```
1644 proc sql; create table class2 as
1645     select %VARLIST(data=SASHELP.CLASS SASHELP.GRIDDED, sep=#cs#,
pattern=#autoCOAL#)
MPRINT(VARLIST): CLASS.Name, CLASS.Sex, CLASS.Age,
coalesce(CLASS.Height, GRIDDED.Height) as Height length=8 label=" ",
coalesce(CLASS.Weight, GRIDDED.Weight) as Weight length=8 label=" ",
GRIDDED.Count, GRIDDED.Density
1646     from SASHELP.CLASS FULL JOIN SASHELP.GRIDDED
1647     ON %VARLIST(data=SASHELP.CLASS #intersect# SASHELP.GRIDDED,
sep=#and#, pattern=#autoEO#)
MPRINT(VARLIST): (CLASS.Height = GRIDDED.Height) and (CLASS.Weight =
GRIDDED.Weight)
1648     having 56 <= height <= 60 and 75 <= weight <=85;
NOTE: Table WORK.CLASS2 created, with 25 rows and 7 columns.
```

PATTERN =

SQL Outer Joins

%VARLIST(): Parameters « out » / SQL Outer Joins

II. Pattern = Single Special Keyword (alternate parameter form)

Special Keyword:

Replaced respectively by:

- **#autoFULLEQ#**, ➤ Full **FROM** clause contents including respectively the code needed for **FULL**, **LEFT**, **RIGHT** or **INNER JOIN(s)** successively for each pair of datasets (specified in data= parameter), with full **ON** clause requiring as matching criterion the **equality between all variables in common** to the pair of datasets being joined
- **#autoLEFTEQ#**,
- **#autoRIGHTEQ#**,
- **#autoINNEREQ#**

```
3119 proc sql;
3120     create table class2 as
3121     select %VARLIST(data=SASHELP.CLASS:a SASHELP.GRIDDED:b,
sep=#cs#, pattern=#autoCOAL#)
MPRINT(VARLIST):    a.Name, a.Sex, a.Age, coalesce(a.Height, b.Height) as
Height length=8 label=" ", coalesce(a.Weight, b.Weight) as Weight
length=8 label=" ", b.Count, b.Density
3122     from %VARLIST(data=SASHELP.CLASS:a SASHELP.GRIDDED:b, sep=#cs#,
pattern=#autoFULLEQ#)
MPRINT(VARLIST):    SASHELP.CLASS as a FULL JOIN SASHELP.GRIDDED as b ON
(a.Height = b.Height) and (a.Weight = b.Weight)
1654     having 56 <= height <= 60 and 75 <= weight <=85;
NOTE: Table WORK.CLASS2 created, with 25 rows and 7 columns.
```

PATTERN =

SQL Outer Joins

%VARLIST(): Parameters « out » / SQL Joins Generalized

III. From = [Single Special Keyword] (default: empty)

Special Keywords: Replaced respectively by:

#autoFULLEQ#, ➤ Appends to the usual return value a full **FROM** clause including
#autoLEFTEQ#, respectively the code needed for **FULL, LEFT, RIGHT** or **INNER JOIN(s)**
#autoRIGHTEQ#, successively for each pair of datasets (specified in data= parameter),
#autoINNEREQ# with full **ON** clause requiring as matching criterion the **equality between all variables in common** to the pair of datasets being joined

```
3144 proc sql;
3145     create table class3 as
3146     select %VARLIST(data=SASHELP.CLASS:a SASHELP.GRIDDED:b SASHELP.CLASSFIT:c,
3147                    sep=#cs#, pattern=#autoCOAL#, from=#autoLEFTEQ#);
MPRINT(VARLIST):  coalesce(a.Name, c.Name) as Name length=8 label=" ",
coalesce(a.Sex, c.Sex) as Sex length=1, coalesce(a.Age, c.Age) as Age length=8,
coalesce(a.Height, b.Height, c.Height) as Height length=8, coalesce(a.Weight,
b.Weight, c.Weight) as Weight length=8, b.Count, b.Density, c.predict,
c.lowermean, c.uppermean, c.lower, c.upper
FROM SASHELP.CLASS as a
LEFT JOIN SASHELP.GRIDDED as b ON (a.Height = b.Height) and (a.Weight = b.Weight)
LEFT JOIN SASHELP.CLASSFIT as c ON (a.Name = c.Name) and (a.Sex = c.Sex) and
(a.Age = c.Age) and (coalesce(a.Height, b.Height ) = c.Height) and
(coalesce(a.Weight, b.Weight ) = c.Weight)
```

NOTE: Table WORK.CLASS2 created, with 19 rows and 12 columns.

FROM =

SQL Joins Generalized

Conclusions – the macro-function %VARLIST():

Allows:

- **Straightforward access** to DATASET VARIABLES, and VARIABLE attributes
- Automated **PROCESSING of LISTS** of VARIABLES.
- **Reduced amount of code typing** for complex PROC SQL JOINS; FULL, LEFT, RIGHT and INNER joins (easily interchangeable)

Can be used:

- in “**OPEN CODE**”, avoids one-off macro definitions.
- with **DYNAMICALLY SPECIFIED lists**

Additionally:

- Extends the **STRUCTURED and DECLARATIVE aspect** of SQL syntax. E.g. you can specify: “select all variables that exist in dataset A but not in dataset B, except for key variables which need a special COALESCE() processing”.
- **OPEN-SOURCE** (on PhUSE Wiki), adaptable for additional needs.
COLLABORATIVE development, validation and documentation welcome.

QUESTIONS & ANSWERS

1. %VARLIST looks **a bit too cryptic** to be used by the average programmer e.g. in Analysis Dataset programs, since one **has to look at the log** to find out what it is actually doing Maybe it would be more suited to build standards macros?
2. What would be the **advantage** of using **PROC SQL+%VARLIST** as opposed to **merges within DATA steps**?
3. The last example (Generalized SQL Join) looks a lot **like SQL Natural Join**, in that you **only specify the datasets to be merged**, nothing more. Why then go to the trouble of using %VARLIST?
4. Why make %VARLIST **available on the PhUSE WIKI**? (
http://www.phusewiki.org/wiki/index.php?title=SAS_macro-function_%25VARLIST)
5. Ideas for **improvements**?