

Efficient SAS Coding with Proc SQL

When Proc SQL is Easier than
Traditional SAS Approaches

Mike Atkinson, May 4, 2005

Note on Efficiency

There are at least two kinds of efficiency: SAS program efficiency and SAS programmer efficiency. This talk addresses the latter.

I think Proc SQL can reduce coding effort, once you become familiar with SQL.

Uses of Proc SQL

Proc SQL is the usual method to select data from an external DBMS, such as Oracle, but don't overlook that...

Proc SQL is very useful with SAS datasets:

- Summarising (alternative to Proc Summary)
- Select totals into a macro variables
- Joining (alternative to Merge in Data Step)
- Sorting
- Creating indexes for efficiency

Selecting from an External DBMS

- Proc SQL is the usual method to select data from an external DBMS, such as Oracle
- Can use pass-through SQL queries to allow DBMS to optimise the query

However, this particular talk focuses on how Proc SQL can be used with SAS datasets.

Summarising Data

- Proc SQL can be used as an alternative to Proc Summary

Advantages of Proc SQL:

- Can use functions (like put), removing need for an additional data step
- Can join two or more datasets in the same step
- Don't need to sort input prior to Proc SQL
- Can sort results in the same step

Traditional SAS Code

```
proc sort data=sasdata.pracds  
    out=my_pracds;  
    by praclha;  
run;
```

```
proc summary data=my_pracds;  
    by praclha;  
    output out=prac_lha_counts  
    (drop=_type_ rename=( _freq_ =prac_cnt ) );  
run;
```

Proc SQL Code

```
proc sql;  
    create table prac_lha_counts as  
    select praclha,  
           count(*) as prac_cnt  
    from sasdata.pracds  
    group by praclha  
    order by praclha;  
quit;
```

Results (portion)

Obs	PRACLHA	prac_cnt
1	1	85
2	2	144
3	3	34
4	4	48
5	5	60
6	6	33
7	7	190
8	9	50
9	10	25
10	11	171
11	12	30

Traditional SAS Code

```
proc sort data=sasdata.fitmserv  
    out=my_fitmserv;  
    by servcd;  
run;
```

```
proc summary data=my_fitmserv;  
    by servcd;  
    output out=servcd_fitm_cnts_0  
        (drop=_type_ rename=(_freq_=fitm_cnt));  
run;
```

```
data servcd_fitm_cnts;  
    set servcd_fitm_cnts_0;  
    servcd_descrip = put(servcd, svcd2ds.);  
run;
```

Proc SQL Code

```
proc sql;  
  create table servcd_fitm_cnts as  
  select servcd,  
         put(servcd, svcd2ds.) as servcd_descrip,  
         count(*) as fitm_cnt  
  from sasdata.fitmserv  
  group by servcd, servcd_descrip  
  order by servcd;  
quit;
```

Results (portion)

Obs	SERVCD	servcd_descrip	fitm_cnt
1	1	1 - REGIONAL EXAMINATIONS (0100,0107)	79
2	2	2 - CONSULTATION (0110)	21
3	3	3 - COMPLETE EXAMINATIONS (0101)	20
4	4	4 - COUNSELLING (0120)	22
5	5	5 - HOME VISITS	13
6	6	6 - EMERGENCY VISITS	26
7	7	7 - INSTITUTIONAL VISITS	22
8	8	8 - MISCELLANEOUS AND OTHER VISITS (GP)	202
9	9	9 - VISIT PREMIUMS (NOT INCLUDED IN SERVICE COUNT	20
10	11	11 - PROLONGED OR EXTENDED VISIT	6

Select Values into Macro Variable

```
proc sql noprint;  
  select count(distinct pracnum) into :prac_cnt  
  from sasdata.pracds;  
quit;  
  
%put prac_cnt = &prac_cnt;
```

Results (In Log):

```
prac_cnt =      24793
```

Select More Values into Macro Variables

```
proc sql noprint;
  select sum(popn), count(*)
  into :pop_lha61, :rec_cnt_lha61
  from sasdata.people
  where lfsclyr = 20042005
  and clntlha = 61;
quit;

%put pop_lha61=&pop_lha61;
%put rec_cnt_lha61=&rec_cnt_lha61;
```

Results (in log)

pop_lha61= 208372

rec_cnt_lha61= 40

Aside: Formatting Macro Variables for Footnotes

```
%let pop_fmt = %sysfunc(putn(&pop_lha61, comma9.));
```

```
footnote1 j=1 "Note: LHA 61 had population of  
&pop_fmt in 2004/2005";
```

Resulting footnote:

```
Note: LHA 61 had population of 208,372 in 2004/2005
```

Joining SAS datasets

- Proc SQL can be an alternative to using a data step merge

Advantages of Proc SQL:

- Datasets do not need to be sorted first
- Can use functions (like substr) in the join, removing need for an additional data step
- Can perform many to many joins
- Can sort results in the same step

Traditional SAS Code

```
proc sort data=my_pracs;  
  by pracnum;  
run;
```

```
proc sort data=sasdata.pracds out=my_pracds;  
  by pracnum;  
run;
```

```
data prac_info;  
  merge my_pracs (in=a keep=pracnum)  
        my_pracds;  
  by pracnum;  
  if (a);  
run;
```

Proc SQL Code

```
proc sql;  
  create table prac_info as  
  select a.pracnum, b.*  
  from my_pracs          a  
  left join sasdata.pracds b  
  on a.pracnum = b.pracnum  
  order by a.pracnum;  
quit;
```

Alternative Proc SQL Code (not quite the same)

```
proc sql;  
  create table prac_info as  
  select a.*  
  from sasdata.pracds a,  
       my_pracs      b  
  where a.pracnum = b.pracnum  
  order by a.pracnum;  
quit;
```

Proc SQL Join With Index for Efficiency

```
proc sql;  
  create unique index phnnum on my_phns;  
quit;
```

```
proc sql;  
  create table hosp_ar_days as  
  select "2004/2005" as fiscal_year,  
         p.phnnum,  
         sum(h.ar_days) as ar_days  
  from sasdata.hosp04 h,  
         my_phns p  
  where h.phn = p.phnnum  
  group by fiscal_year, p.phnnum  
  order by p.phnnum;  
quit;
```

Results (portion)

Obs	fiscal_ year	phnnum	ar_days
1	2004/2005	9013461036	15
2	2004/2005	9013510873	6
3	2004/2005	9013751586	4
4	2004/2005	9013931499	2
5	2004/2005	9013948747	22
6	2004/2005	9014182899	5
7	2004/2005	9015106078	7
8	2004/2005	9015259421	3
9	2004/2005	9015371037	4
10	2004/2005	9015670239	1
11	2004/2005	9015834749	2
12	2004/2005	9016265127	33
13	2004/2005	9016750346	1
14	2004/2005	9017397422	10
15	2004/2005	9017503546	2

NOTE: MERGE statement has more than one data set with repeats of BY values.

```
proc sort data=sasdata.fitmserv
          out=my_fitmserv;
  by fitm;
run;
```

```
proc sort data=sasdata.fitmspec
          out=my_fitmspec;
  by fitm;
run;
```

```
data spec_serv;
  merge my_fitmserv (in=a)
        my_fitmspec (in=b);
  by fitm;
run;
```

Many to Many Joins Are Possible with Proc SQL

```
proc sql;  
  create table spec_serv as  
  select a.servcd,  
         b.speccode,  
         count(*) as fitm_count  
  from sasdata.fitmserv a,  
       sasdata.fitmspec b  
  where a.fitm = b.fitm  
  group by a.servcd, b.speccode;  
quit;
```

Results (portion)

Obs	SERVCD	SPECCODE	fitm_ count
1	1	30	6
2	1	31	3
3	1	32	17
4	1	34	4
5	1	38	7
6	1	39	4
7	1	42	21
8	1	43	10
9	2	00	1
10	2	32	1
11	2	34	4
12	2	38	5
13	2	42	5
14	2	43	1

To Find Out, Are Any Fee Items Related to More than One Service Code?

```
proc sql;  
    create table fitm_mults as  
        select fitm,  
               count(*) as rec_cnt  
        from sasdata.fitmserv  
        group by fitm  
        having count(*) > 1;  
quit;
```

Results (portion)

Obs	FITM	rec_cnt
1	41	2
2	116	2
3	122	2
4	127	2
5	141	2
6	222	2
7	223	2
8	224	2
9	333	2
10	381	2
11	613	2
12	614	2
13	615	2

Building a Format

```
proc sql noprint;
  create table mypracnm as
  select a.pracnum as start,
         a.pracnum as end,
         case when b.pracnum is missing then
              put(a.pracnum, z5.) || ' - **No Name Found**'
            else
              put(a.pracnum, z5.) || ' - ' || b.pracnm
            end as label,
         'mypracnm' as fmtname,
         'N' as type
  from my_pracs          a
       left join sasdata.pracds b
  on a.pracnum = b.pracnum;
quit;

proc format cntlin=mypracnm;
run;
```

Sample Usage of Format

```
data two_small_pracs;  
    pracnum = 2600; output;  
    pracnum = 2624; output;  
run;  
  
proc print data=two_small_pracs;  
    format pracnum mypracnm.;  
run;
```

Obs	pracnum
1	02600 - **No Name Found**
2	02624 - SMITH JOHN M