Selecting All Observations When Any Observation Is of Interest

Christopher Bost

What does that mean?

- Some data have more than one obs/person
- We want to select all obs for a person
  - If at least one obs for that person meets criteria
- Examples?

What's in it for you?

- Review how to select obs with the DATA step
- Review how to select obs with PROC SQL
- Useful techniques to have in your SAS "toolkit"

Sample data set: COURSES

studentid	course	ap
1	BIOL101	0
2	BIOL102	1
2	CHEM102	1
3	CALC101	0
3	STAT101	0
4	PSYC201	1
4	HIST102	0
4	PHYS101	1
5	CHEM101	0
5	CALC102	1

Desired data set

studentid	course	ap
2	BIOL102	1
2	CHEM102	1
4	PSYC201	1
4	HIST102	0
4	PHYS101	1
5	CHEM101	0
5	CALC102	1

Select observations with DATA step

1. Subset observations of interest
2. Keep one observation per person
3. Match-merge with original data set

1. Subset observations of interest

```
data ap;
set courses;
where ap=1;
run;
```

Data set AP

studentid	course	ap
2	BIOL102	1
2	CHEM102	1
4	PSYC201	1
4	PHYS101	1
5	CALC102	1

2. Keep one observation per person

```
proc sort data=ap out=ap_sort;
by studentid;
run;

data ap2;
set ap_sort;
by studentid;
if first.studentid;
keep studentid;
run;
```

Data set AP2

studentid
2
4
5

3. Match-merge observations

```
proc sort data=courses out=courses_sort;
by studentid;
run;

data anyap;
merge courses_sort ap2(in=inap2);
by studentid;
if inap2;
run;
```

Desired data set: ANYAP

studentid	course	ap
2	BIOL102	1
2	CHEM102	1
4	PSYC201	1
4	HIST102	0
4	PHYS101	1
5	CHEM101	0
5	CALC102	1

Full DATA step program

```
data ap;
set courses;
where ap=1;
run;

proc sort data=ap
          out=ap_sort;
by studentid;
run;

data ap2;
set ap_sort;
by studentid;
if first.studentid;
keep studentid;
run;
```

```
proc sort data=courses
          out=courses_sort;
by studentid;
run;

data anyap;
merge courses_sort
      ap2(in=inap2);
by studentid;
if inap2;
run;
```

Pros and cons

- It works
- Three DATA steps
- Two PROC SORT steps

Select observations with PROC SQL

1. Use a subquery
2. Use GROUP BY and HAVING clauses

Terminology

DATA Step	PROC SQL
Variable	Column
Observation	Row
SAS data set	Table

SQL clauses: required order

	`PROC SQL;`	*starts procedure*
1	**`SELECT`**	**selects variables**
2	**`FROM`**	**opens data sets**
3	**`WHERE`**	**restricts observations**
4	**`GROUP BY`**	**groups observations**
5	**`HAVING`**	**restricts groups**
6	**`ORDER BY ;`**	**sorts results**
	`QUIT;`	*ends procedure*

Use a subquery

```
  proc sql;
  create table anyap2 as
3 select *
1 from courses
2 where studentid in (select distinct studentid
                       from courses
                       where ap=1)
4 order by studentid;
  quit;
```

Desired data set: ANYAP2

studentid	course	ap
2	BIOL102	1
2	CHEM102	1
4	PSYC201	1
4	PHYS101	1
4	HIST102	0
5	CALC102	1
5	CHEM101	0

Pros and cons

- Single step
- Two passes through data set
  - One for the subquery
  - One for the outer query

SQL clauses: order of execution

1	**FROM**	**opens data sets**
2	**WHERE**	**restricts observations**
3	**GROUP BY**	**groups observations**
4	**HAVING**	**restricts groups**
5	**SELECT**	**selects variables**
6	**ORDER BY**	**sorts results**

Use GROUP BY and HAVING clauses

```
  proc sql;
  create table anyap3 as
4 select *
1 from courses
2 group by studentid
3 having sum(ap=1) > 0
5 order by studentid;
  quit;
```


Desired data set: ANYAP3

studentid	course	ap
2	CHEM102	1
2	BIOL102	1
4	HIST102	0
4	PSYC2101	1
4	PHYS101	1
5	CALC102	1
5	CHEM101	0

Pros and cons

- Single step
- **Flexibility**
  - Any condition(s) in parentheses after SUM
- **NOTE: The query requires remerging summary statistics back with the original data.**

Conclusion

- Use either the DATA step or PROC SQL
- PROC SQL requires significantly less coding
- PROC SQL is not necessarily more efficient
  - Test on your own data
- Use on **any data** with groups of observations

Contact information

Comments and questions are valued and encouraged.

Christopher J. Bost
MDRC
16 East 34th Street
New York, NY 10016
(212) 340-8613
christopher.bost@mdrc.org
chrisbost@gmail.com

Alternate DATA step method

```
proc sort data=courses(where=(ap=1) keep=studentid ap)
          out=lookup(keep=studentid)
          nodupkey;
by studentid;
run;

proc sort data=courses out=courses_sort;
by studentid;
run;

data anyap;
merge courses_sort lookup(in=inlookup);
by studentid;
if inlookup;
run;
```

Alternate PROC SQL method

```
proc sql;
create table anyap4 as
select courses.*
from courses inner join
     (select distinct studentid as studentid2
      from courses
      where ap=1)
on studentid=studentid2
order by studentid;
quit;
```