Combining Data Horizontally
Terminology

- Table Lookup
- Base table
- Lookup tables
- Lookup values
Working with Lookup Values Outside of SAS Data Sets

- Lookup tables are not necessarily SAS data sets.
- The following techniques can be used to hard-code lookup values into programs:
  - IF-THEN/ELSE statements
  - SAS arrays
  - User-defined SAS formats
Advantages: easy to use and to understand, versatile

Disadvantages: Code requires maintenance. Lookup values might change. Number of statements might be very large and create inefficiencies both in execution and maintenance.
IF-THEN/ELSE Statement Example

data new;
set old;
if id=1 then x=4;
else if id=2 then x=5;
else if id=3 then x=6;

<table>
<thead>
<tr>
<th>ID</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>
SAS Arrays

- Lookup values can be hard-coded into the program or read into the array from a data set
- Array elements are referenced positionally
- Potential disadvantages: system memory requirements, only returns a single value per lookup operation, dimensions of the array must be known at compile time
### Scoring Example with 1-Dimensional SAS Array

<table>
<thead>
<tr>
<th></th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response Variable</td>
<td>r1</td>
<td>r2</td>
<td>r3</td>
</tr>
<tr>
<td>Answer Key</td>
<td>B</td>
<td>D</td>
<td>C</td>
</tr>
</tbody>
</table>

```sas
data one;
input name $4. +1 (r1-r3) ($1.);
array answer {3} $1 _temporary_ ('B','D','C');
array response r1-r3;
score=0;
do _i_=1 to 3;
   if answer{*_i_}=response{*_i_} then score+1;
end;
```
DATA Step match-merge

- Familiar technique from STAT 540
- Typically introduced as
  - a one-to-one Outer Join
  - A many-to-one match merge of summary data
- Not appropriate for a many-to-many match
DATA Step match-merge

BY variables should match, but matching can be done during execution.

```sas
proc sort data=a; by student;
proc sort data=b; by name;
data gradebook;
merge a(in=in_a) b(in=in_b rename=(name=student));
by student;
if in_a and in_b; run;
```
DATA Step match-merge vs. PROC SQL

- **Match-merge**
  - Unlimited data sets
  - More complex data management

- **PROC SQL**
  - No pre-sorting
  - No common variables
DATA Step match-merge vs. PROC SQL

- **Match-merge**
  - Portable Data Vector (PDV) used to hold information while DATA step executes
  - Outputs first observation from each data set for each level of the BY group variable

- **PROC SQL**
  - Creates Cartesian product
  - Eliminates ineligible cases in WHERE clause
DATA Step match-merge

- The DATA step can be used for many-to-one match merges
  - By exporting calculation of summary measures
  - By computing summary measures within the DATA step itself
  - STAT 540 example
The DATA step tends to over-match on many-to-many match merges

The text introduces a fix, but it’s cumbersome
Using an Index to Combine Data

- Useful when
  - One of the data sets is much larger than the other
  - The smaller data set contains all the cases of interest (e.g., a left/right join)
- Appropriate for one-to-one matches only
Using an Index to Combine Data

Example

- SAS uses the noobs index in Fall08 to find lookup values in Fall10ms to match values of the index.
- The smaller data set has to be included first so that lookup values are available in the PDV for use by the index.
- _IORC_ (Input/Output Return Code) indicates whether a match for each record in the smaller data set was found.
Using an Index to Combine Data

Example

- Full Fall08 data set
- Fall10 Marine Science majors

```sql
proc sql;
create index noobs on fall08(noobs);
quit;

data msretro;
set fall10ms;
set fall08 key=noobs;
run;
```
Using a Transactional Data Set

- The Base data set can be updated from a lookup table
- Both data sets have to be sorted
- The lookup table can have missing values for variables that are unchanged
- Be careful about “mixed” information (see example)