```
DATA dads;
 INPUT famid name $  dadinc ;
DATALINES;
2 Art  22000
1 Bill 30000
3 Paul 25000
;
RUN;

DATA moms;
 INPUT famid name $  mominc ;
DATALINES;
1 Bess 15000
3 Pat  50000
2 Amy  18000
;
RUN;


DATA faminc;
   INPUT famid faminc1-faminc12 ;
CARDS;
1 3281 3413 3114 2500 2700 3500 3114 3319 3514 1282 2434 2818
2 4042 3084 3108 3150 3800 3100 1531 2914 3819 4124 4274 4471
3 6015 6123 6113 6100 6100 6200 6186 6132 3123 4231 6039 6215
;
RUN;



/* Class 4 Part II -- Data Set Appending and Merging and
Advanced Considerations */


proc print data=dads;
run;

proc print data=moms;
run;

/* Create a variable indicating mom vs. dad*/

data momsdads;
     set moms(in=mom) dads(in=dad);
          if mom=1 then momdad = "Mom";
          if dad=1 then momdad = "Dad";
```

```
            momflag = 0;
            if mom=1 then momflag=1;
run;

/*But what if we want to append the income variables "on top" of each
other*/

/*Option 1*/

data momsdads_alt (drop=dadinc mominc);
      set moms(in=mom) dads(in=dad);
            if mom=1 then do;
                  momdad = "Mom";
                  inc=mominc;
            end;
            if dad=1 then do;
                  momdad = "Dad";
                  inc=dadinc;
            end;

            momflag = 0;
            if mom=1 then momflag=1;
run;

/*Option 2 is to use the rename statement in the set line */

data momsdads_alt2;
      set moms(rename=(mominc=inc)) dads(rename=(dadinc=inc));
run;

/*Be very mindful of your syntax!!!*/

/*Remember that appending is different than merging!*/
/*Merging will generally keep the same number of observations as you
currently have
/*Appending will generally result in the sum of the two datasets'
number of observations */

proc sql;
      create table momsdads_merge
      as select moms.famid,
                  moms.name as mom_name,
                  dads.name as dad_name,
                  mominc,
                  dadinc
      from moms, dads
```

```
      where moms.famid = dads.famid
      order by moms.famid;
run;

/*ABBAQAYD*/

/*Appending is very different than merging*/
/*A Data Step with a set statement with more than one dataset on the
line will always result
in appending*/


/*Section 2: Arrays */


proc print data=faminc;
run;

/*Want to calculate the effect of a 15 percent tax on every family*/

data faminc_tax;
      set faminc;

      taxinc1 = faminc1 * 0.15;
      taxinc2 = faminc2 * 0.15;
      taxinc3 = faminc3 * 0.15;

      /*This is getting tedious, but you could do it.*/

run;

/*Let's use arrays!*/

data faminc_a (drop=i);
      set faminc;

            ARRAY Afaminc(12) faminc1-faminc12;
            ARRAY Ataxinc(12) taxinc1-taxinc12;

            DO i = 1 to 12;
                  Ataxinc(i) = Afaminc(i) * 0.15;
            END;
run;

/*You can move forwards and backwards in an array within the same
line*/
```

```
/*Say you want to calculate quarterly income*/

data faminc_q;
      set faminc;

            ARRAY Afaminc(12) faminc1-faminc12;
            ARRAY Aincqtr(4) incqtr1-incqtr4;

                  DO qtr = 1 to 4;
                        month3 = 3*qtr;
                        Aincqtr(qtr) = Afaminc(month3-2) +
Afaminc(month3-1) + Afaminc(month3);
                  END;
run;


/*One more advanced array example*/

/*Let's test for whether or not a family's income declined from one
month to the next,
and count the number of times its income declined*/

data faminc_dec;
      set faminc;

            ARRAY Afaminc(12) faminc1 - faminc12;
            ARRAY A_decinc(2:12) decinc2 - decinc12;

            DO month = 2 to 12;
                  if Afaminc(month) < Afaminc(month-1) then
A_decinc(month) = 1; /*This indicates if the income declined in any
given month*/
                  else A_decinc(month) =0;
            end;

            sum_declines=0;
            do month = 2 to 12;
                  sum_declines = sum_declines + A_decinc(month); /*This
sums all the months that a income declined*/
            end;
run;



/*Proc Transpose is a very powerful tool for data manipulation*/
```

```
/*Unfortunately the syntax is very difficult, and you usually need
to guess and check a couple times to ensure that your desired output
is being created*/

proc transpose data=faminc out=faminc_t prefix=faminc;
     by famid;
     var faminc1-faminc12;
run;

/* Above, variable is the info that is going from wide/long to
long/wide
/* By is the information that you "pivot around"
/*ID, if you have it, identifies the variable which can create a name
for the newly created variables
/*Prefix will identify the renamed variables*/

/* We usually need to do some clean-up after a transpose*/

data faminc_t_clean (drop=_NAME_ faminc1);
     set faminc_t;

      faminc = faminc1;

      format month 2.;

      month = substr((_NAME_),indexc(_NAME_,'c')+1,length(_NAME_));

/*     month =
put(substr((_NAME_),indexc(_NAME_,'c')+1,length(_NAME_)),2.); */

/*Put tells SAS to explicitly cast a variable as numeric
/*Input tells SAS to explicitly cast a variable as character*/

run;


/*You can continue to transpose your data in ways to "groom" your
dataset
in whatever format you want*/

/*Be mindful that you need to sort your data prior to a proc
transpose*/

proc sort data=faminc_t_clean;
     by month;
run;
```

```
proc transpose data=faminc_t_clean out=faminc_t2 prefix=inc_;
      var faminc;
      id famid;
      by month;
run;


/*Remember you don't necessarily need a transpose proc inorder to
transpose
data */

data faminc_t_alt (keep=famid month faminc);
      set faminc;

      month = 1;
      faminc = faminc1;
      output;

      month = 2;
      faminc = faminc2;
      output;

      month = 3;
      faminc = faminc3;
      output;

run;
```