

There may be times when you run a SAS procedure when you would like to direct the procedure output to a SAS data set rather than to the output window. There are a number of different ways to accomplish this task and some of them will be illustrated using the example of age adjusting of death rates.

Age adjusting a rate is a common task when using public health data. For example, you might be interested in comparing the death rates among counties in New York State. Since overall death rates are higher among the elderly, counties with older populations will have higher overall death rates. Here are some fake data showing how the overall death rate in two counties can differ due to a difference in the age distribution rather than difference in age-specific death rates (the RATE is the number of deaths per 1000 population).

AGE GROUP	COUNTY A			COUNTY B		
	POPULATION	DEATHS	RATE	POPULATION	DEATHS	RATE
<20	1000	10	10	500	5	10
20-44	1000	20	20	500	10	20
45-64	500	30	60	1000	60	60
65+	500	50	100	1000	100	100
TOTAL	3000	110	36.7	3000	175	58.3

Though the age-specific death rates in th two counties are identical, the TOTAL death rate in COUNTY B is much higher than that in COUNTY A since a higher proportion of its population experiences the higher age-specific death rates (or, "...who dies, old people die ...").

Imagine it is your task to compute the TOTAL death rate (sometimes referred to as the CRUDE death rate) and the AGE-ADJUSTED death rate for New York State and also for each county in New York State. You need data on both deaths and population so you can compute the rates. Age adjusting is commonly done by first computing the age-specific death rates in the following 11 age groups: <1, 1-4, 5-14, 15-24, 25-34, 35-44, 45-54, 55-64, 65-74, 75-84, 85+. Once the age-specific rates are computed, weights (a standard population) are applied to each rate and an age-adjusted overall death rate is computed. The purpose of the weights is to remove the effect of varying population distributions.

You are given a raw data file that contains information on each death that occurred among New York State residents in 2000. The only two variables of interest to you are age and county of residence. You know that you must group the deaths into the age groups shown above, so you decide that the best way to group the observations will be using a FORMAT. You create a numeric format named AGE.

```
proc format;
value age
0 = '1'
1-4 = '2'
5-14 = '3'
15-24 = '4'
25-34 = '5'
35-44 = '6'
45-54 = '7'
55-64 = '8'
65-74 = '9'
75-84 = '10'
85-hi gh = '11'
;
run;
```

Next, you must read the raw data file that contains the death data. The value of age in the data file may be in terms of years, months, days, hours, or minutes. There is a variable in the data that tells you the UNITS of the age variable. If UNITS are 2, 3, 4, or 5, the age at death is in terns of months, days, hours, or minutes respectively.

```

data deaths (drop=mcd units); ❶
if mod(_n_,10000) eq 0 then put _n_=; ❷
infile 'd:\vs\deaths\data\dthsta00.txt'; ❸
input
  @005 county    $2.
  @007 mcd       $2.
  @023 units     $1.
  @066 age      ?? 3.
;
if (county ge '01' and county le '61' or county eq '70'); ❹
if county eq '70' then county = mcd;
if units ge '2' and units le '5' then age = 0; ❺
run;

```

You create a data set named DEATHS ❶. You like to give yourself some information as to how much progress that data step is making ❷. Each time you read 10,000 records from the raw data file, the number of records read will be written to the LOG file. All deaths among New York City residents are given a county code of '70' ❸. If a death occurs to a New York City resident, you replace the county code with the borough code (MCD, minor civil division). Finally, if the UNITS of the age at death are less than years, you assign the deceased an age of 0 (the <1 age group needed for age adjusting).

Now you have to count the number of death within each age group, within each county. However, you do not want to print the results. You want to direct them to a SAS data set so you can use them later in the SAS job to compute deaths rates. There are a number of different ways to do this. First, you can use PROC FREQ.

```

*** count deaths within each county and age group;
proc freq data=deaths;
table county*age / noprint out=dtab1; ❶
where age ne .; ❷
format age age.; ❸
run;

proc print data=dtab1; ❹
run;

```

You create a table (counts of deaths by COUNTY and AGE), but you do not print the output ❶. The NOPRINT option suppresses the output normally sent to the output window, while the OUT= option specifies the name of the data set to be created, in this case DTAB1. You do not want to use any observations that do not have a value for age ❷. You use the format AGE (created earlier) to create the counts of deaths in age groups rather than by individual age ❸. You use PROC PRINT to examine the data set DTAB1 ❹. A portion of it looks as follows:

Obs	county	age	COUNT	PERCENT
1	01	1	28	0.01779
2	01	2	2	0.00127
3	01	3	7	0.00445
4	01	4	23	0.01461
5	01	5	23	0.01461
6	01	6	74	0.04702
7	01	7	159	0.10102
8	01	8	247	0.15693
9	01	9	470	0.29861
10	01	10	954	0.60612
11	01	11	984	0.62518
12	02	1	6	0.00381
13	02	2	1	0.00064
14	02	3	2	0.00127
15	02	4	5	0.00318
16	02	5	2	0.00127
17	02	6	18	0.01144
18	02	7	26	0.01652
19	02	8	49	0.03113
20	02	9	108	0.06862
21	02	10	136	0.08641
22	02	11	134	0.08514

There are four variables in the data set: COUNTY, AGE, COUNT, and PERCENT. Remember that you used a FORMAT (AGE) to group the observations into age groups. Therefore, the value of AGE that you see using PROC PRINT is FORMATTED. If you print the data set removing the format as follows...

```
proc print data=dtab1;
format _all_ ;
run;
```

you will see the actual values of the variable AGE in data set DTAB1. Here are the same 22 observations.

Obs	county	age	COUNT	PERCENT
1	01	0	28	0.01779
2	01	1	2	0.00127
3	01	5	7	0.00445
4	01	15	23	0.01461
5	01	25	23	0.01461
6	01	35	74	0.04702
7	01	45	159	0.10102
8	01	55	247	0.15693
9	01	65	470	0.29861
10	01	75	954	0.60612
11	01	85	984	0.62518
12	02	0	6	0.00381
13	02	1	1	0.00064
14	02	5	2	0.00127
15	02	15	5	0.00318
16	02	25	2	0.00127
17	02	35	18	0.01144
18	02	45	26	0.01652
19	02	55	49	0.03113
20	02	65	108	0.06862
21	02	75	136	0.08641
22	02	85	134	0.08514

Notice that the actual values are the lower bound of each age group in the format AGE. If there had been a county with no deaths at an age corresponding to the lower bound of the age group, the data set DTAB1 would contain the first value of age within that age group.

Another way to create counts of deaths by county and age group is to use PROC SUMMARY.

```
proc summary data=deaths; ❶
class county age; ❷
output out=dtab2; ❸
format age age.; ❹
run;

proc print data=dtab2; ❺
run;
```

PROC SUMMARY is the same procedure as PROC MEANS ❶. However, PROC SUMMARY does not create any printed output (as if you had used PROC MEANS with a NOPRINT option). A CLASS statement is used to group observations by COUNTY and AGE ❷. An OUTPUT statement and OUT= option are used to create a SAS data set named DTAB2 ❸. As with PROC FREQ, the AGE format creates counts in age groups ❹. PROC PRINT displays the data set DTAB2 ❺ and here is a portion of the output.

Obs	county	age	_TYPE_	FREQ
1		.	0	157394
2		1	1	1632
3		2	1	262
4		3	1	351
5		4	1	1485
6		5	1	2343
7		6	1	5481
8		7	1	10424
9		8	1	15596
10		9	1	28471
11		10	1	45501
12		11	1	45848
13	01	.	2	2971
14	02	.	2	487
15	03	.	2	2223
16	04	.	2	836
17	05	.	2	732
18	06	.	2	1535
19	07	.	2	959
20	08	.	2	529
<one observation per county, all _TYPE_ 2>				
70	93	.	2	11469
71	94	.	2	10138
72	95	.	2	18452
73	96	.	2	16220
74	97	.	2	3457

75	01	1	3	28
76	01	2	3	2
77	01	3	3	7
78	01	4	3	23
79	01	5	3	23
80	01	6	3	74
81	01	7	3	159
82	01	8	3	247
83	01	9	3	470
84	01	10	3	954
85	01	11	3	984
86	02	1	3	6

<one _TYPE_ 3 observation for each county/age combination in the data set>

You will notice that you get more counts than you did with PROC FREQ. Before you get to the counts in county and age groups, you get an overall count of deaths (the value of both COUNTY and AGE are MISSING), age-specific counts for the entire state (the value of county is MISSING), and county-specific counts (the value of AGE is MISSING). You also get a new variable, _TYPE_, and the variable _FREQ_ (not COUNT as in PROC FREQ) contains the death counts. Once again, the values of the variable AGE are formatted. The real values are the same as those you saw in data set DTAB1. Here's a look at the first 12 observations UNFORMATTED.

Obs	county	age	_TYPE_	FREQ
1		.	0	157394
2		0	1	1632
3		1	1	262
4		5	1	351
5		15	1	1485
6		25	1	2343
7		35	1	5481
8		45	1	10424
9		55	1	15596
10		65	1	28471
11		75	1	45501
12		85	1	45848

If you use either PROC FREQ or PROC SUMMARY as just shown and you have a count in your data with no deaths in one or more age groups, that age group will not appear in the output data set (either DTAB1 or DTAB2). For example, in the 2000 death data, county '54' has no deaths in age group 2.

585	54	1	3	11
586	54	3	3	2
587	54	4	3	9
588	54	5	3	7
589	54	6	3	7
590	54	7	3	39
591	54	8	3	45
592	54	9	3	78
593	54	10	3	185
594	54	11	3	189

As of version 8.2 of SAS, some PROCs allow you to force the procedure to create all the groups specified in a format. PROC SUMMARY is one of those PROCs.

```
proc summary data=deaths complete types; ❶
class county age / preloadfmt; ❷
output out=dtab3;
format age age.;
run;
```

There are additions to the SAS job you just ran. First, the COMPLETETYPES option is specified ❶. Then, the PRELOADFMT option is used on the CLASS statement. The observations for county '54' now contain an observation for age group 2 with a count of zero fir deaths.

614	54	1	3	11
615	54	2	3	0
616	54	3	3	2
617	54	4	3	9
618	54	5	3	7
619	54	6	3	7
620	54	7	3	39
621	54	8	3	45
622	54	9	3	78
623	54	10	3	185
624	54	11	3	189

The last step with the death counts is to create one observation per county that contains the death counts in each age group.

```
proc transpose data=dtab3 out=dtrans3 (drop=_) prefix=d; ❶
var _freq_; ❷
id age; ❸
by county; ❹
where age ne .; ❺
run;

proc print data=dtrans3 noobs; ❻
run;
```

PROC TRANSPOSE is used to rearrange the data and create a data set named DTRANS3 ❶. The variable whose values will occur in the data set is specified in a VAR statement, _FREQ_ ❷. The variable that will control the what variable will contain the value of _FREQ_ is specified in an ID statement ❸. A BY statement specifies the variable that controls what observation contains the death counts, COUNTY ❹. A WHERE statement eliminates any observation with no value for the variable AGE (eliminating the overall count, and the county-specific overall counts) ❺. PROC PRINT displays the data, and here are the first five observations.

county	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
01	1632	262	351	1485	2343	5481	10424	15596	28471	45501	45848
02	28	2	7	23	23	74	159	247	470	954	984
03	6	1	2	5	2	18	26	49	108	136	134
04	21	1	2	9	11	42	102	189	364	747	735
04	6	1	1	11	9	25	63	87	158	264	211

Notice the names of the variables. They are a combination of the value of the PREFIX option in PROC TRANSPOSE and the formatted value of the ID variable AGE. The first observation with no value for the variable county is the state total.

Now that we have a data set with counts of deaths, we need populations to calculate rates. Population data set from the 2000 census are contained in a SAS data set named T12.

```
data pop1 (keep=county p0-p11); ❶
set sf1.t12; ❷
where sumlev eq '050'; ❸

p0 = t012001; ❹
p1 = sum (of t012003 t012107); ❺
p2 = sum (of t012004-t012007 t012108-t012111);
p3 = sum (of t012008-t012017 t012112-t012121);
p4 = sum (of t012018-t012027 t012122-t012131);
p5 = sum (of t012028-t012037 t012132-t012141);
p6 = sum (of t012038-t012047 t012142-t012151);
p7 = sum (of t012048-t012057 t012152-t012161);
p8 = sum (of t012058-t012067 t012162-t012171);
p9 = sum (of t012068-t012077 t012172-t012181);
p10 = sum (of t012078-t012087 t012182-t012191);
p11 = sum (of t012088-t012105 t012192-t012209);
run;

proc print data=pop1 noobs; ❻
var county p0-p11;
run;
```

There are many variables in the data set T12. All we need to calculate rates are the COUNTY, the overall population (P0) and the age-specific populations (P1-P11) ❶. The data set T12 is a permanent data set located in a library referred to with LIBNAME SF1 (the data are from 2000 census summary file one) ❷. The data set T12 contains populations for the state, counties, census tracts, etc. County-specific populations are selected using a WHERE statement ❸. The overall population is placed in variable P0 (from variable T012001 in data set T12) ❹. The SUM function is used to create the 11 age-specific populations ❺. PROC PRINT displays the data ❻ and here are the first two observations.

COUNTY	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11
001	294565	3192	13457	38376	44836	39421	45513	42201	24975	20783	15826	5985
003	49927	528	2270	7133	9987	5040	6902	6505	4562	3748	2328	924

If you look at the entire data set, you will notice that there is no observation for the overall state population. You will also notice that the values for the variable county are different from those in the death data (FIPS, or federal county numbers, in the population data and New York State vital statistics county numbers, VS, in the death data). An observation for the overall state population can be added using PROC SUMMARY.

```
proc summary data=pop1; ❶
var p0-p11; ❷
class county; ❸
output out=tpop1 (drop=_) sum=; ❹
run;

proc print data=tpop1 noobs; ❺
var county;
run;
```

PROC SUMMARY is used to create a data set, TPOP1 ❶. When PROC SUMMARY was used with the death data, there was no VAR statement. This time, we want to sum the populations over counties within each age group ❷. The CLASS COUNTY statement produces an observation for the overall state plus one observation per county ❸. The SUM= option in the OUTPUT statement requests that all variables (P0-P11) in the VAR statement be summed over all levels of the variable (COUNTY) in the CLASS statement ❹. PROC PRINT displays the new data set, and here is portion of the output showing the overall state populations, plus some of the new county numbers, matching those in the death data ❺.

COUNTY	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11
001	18976457	243891	995526	2684290	2531853	2757324	3074298	2552936	1687987	1276046	860818	311488
003	294565	3192	13457	38376	44836	39421	45513	42201	24975	20783	15826	5985
005	49927	528	2270	7133	9987	5040	6902	6505	4562	3748	2328	924
	1332650	21416	88316	227583	201560	208458	200321	148710	102338	70823	44636	18489

Though we already have a population data set that can be used to calculate death rates, we will look at one other way to produce the same data set. When the age-specific populations were created using the SUM function, the data set variable names were used. It might be more convenient if we could use variable names or array elements with names that more closely resembled.

```
data pop2 (keep=county p0-p11); ❶
array m(0:102) t012003-t012105; ❷
array f(0:102) t012107-t012209;
array p(11); ❸
array start(11) _temporary_ (0 1 5 15 25 35 45 55 65 75 85); ❹
array stop(11) _temporary_ (0 4 14 24 34 44 54 64 74 84 102);
set sf1.t12; ❺
where sumlev eq '050';
do i=1 to 11; ❻
  p(i) = 0;
do j=start(i) to stop(i); ❼
  p(i) + m(j) + f(j);
end;
end;
p0 = sum (of p1-p11); ❸
run;

proc print data=pop2 noobs; ❾
var county;
sum p0-p11;
run;
```

Once again, a data set (POP2) is created containing a county number, plus a total population and eleven age-specific populations ❶. Two arrays are defined, one for males (m) and one for females (f) ❷. Notice that the array subscripts are defined as starting at 0 and ending at 102. There are 103 variables referenced by both arrays, t012003 through t012105 for males and t012107 through t012209 for females. The next array will be used to hold the age-specific populations ❸. Two temporary arrays (start and stop) are defined ❹. The values of the array elements are the age ranges for the eleven age-specific population groups. The population data set is read, limited to those observations with county-specific data ❺. Two DO LOOPS are used to sum the appropriate populations for each age group. The first loop iterates 11 times, once per age group ❻. Prior to the next loop, the population is set to 0 (why is this necessary?). The next loop use the start and stop values to sum the appropriate populations ❼. After the two loops are complete, the total population is calculated ❸. The populations are displayed with PROC PRINT ❾.

COUNTY	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11
001	18976457	243891	995526	2684290	2531853	2757324	3074298	2552936	1687987	1276046	860818	311488
003	294565	3192	13457	38376	44836	39421	45513	42201	24975	20783	15826	5985
005	49927	528	2270	7133	9987	5040	6902	6505	4562	3748	2328	924
005	1332650	21416	88316	227583	201560	208458	200321	148710	102338	70823	44636	18489

The results are identical to that of the data step used previously.

Before we can merge the death and population data sets, we will convert the county numbers in the death data set from VS to FIPS county numbers. A format will be used to do the conversion.

```
data dtrans3;
set dtrans3 (rename=(county=temp)); ❶
county = put(temp,$cog2fip.); ❷
if county eq '999' then county = ''; ❸
d0 = sum(of d1-d11); ❹
drop temp; ❺
run;
```

```
proc print data=dtrans3 noobs; ❻
var county d1-d11;
run;
```

The data set DTRANS3 is read and the variable COUNTY is renamed TEMP ❶. A PUT function and format are used to recreate the variable COUNTY from the value of the variable TEMP ❷. The RENAME option used with the SET statement allows us to end up with the correct variable name, COUNTY, matching the variable name in the population data set. The format will convert unknown VS county numbers to the value '999'. However, the total state observation in the population data set has a blank county number. So, the '999' is changed to a blank ❸. The total number of deaths is calculated from the age-specific death counts ❹. We no longer need the variable TEMP and it is dropped from the data set ❺. The death data are displayed with PROC PRINT, but notice the last six observations in the data set.

123	3	2	0	2	1	8	9	28	65	81	79
061	108	10	28	81	229	528	989	1246	2011	2903	3336
005	169	23	45	150	289	662	995	1205	1755	2363	2482
047	279	46	51	260	437	957	1659	2131	3296	4716	4620
081	176	28	31	145	265	578	1093	1559	2969	4668	4708
085	36	6	9	33	50	117	262	385	668	984	907

The county numbers are out of order since in the VS county numbering system, county numbers 93 through 97 are New York City counties, while the FIPS numbering system is alphabetic. For example, in the VS numbering system, the Bronx is county number 94, while its FIPS number is 005.

```
proc sort data=dtrans3; ❶
by county;
run;

data dth_pop; ❷
merge dtrans3 tpop1; ❸
by county;
run;
```

Prior to merging the death and population data sets, the death data are sorted ❶. A new data set (DTH_POP) is created ❷. The deaths and populations are merged, combining observations with like county numbers ❸. Now that we have a data set with death and population data in the same observation(s), we can calculate rates.

```
data rates (keep=county d0 p0 r_); ❶
array adj (11) _temporary_ (.013818 .055317 .145565 .138646 .135573 .162613
.134834 .087247 .066037 .044842 .015508); ❷

array d(11); ❸
array p(11);
set dth_pop1;
r_crude = 1000 * d0 / p0; ❹
r_adj_usted = 0; ❺
do j=1 to 11; ❻
r_adj_usted + adj(j)*d(j)/p(j);
end;
r_adj_usted = 1000*r_adj_usted; ❼
label
d0 = 'DEATHS'
p0 = 'POPULATION'
;
run;
```

Another new data set (RATES) will be created, containing a county number, total deaths and total population, plus two rates (the only two variables in the data set that begin with the two characters R_) ❶. An array (ADJ) is used to hold the eleven factors used direct age-adjust the county-specific death rates ❷. These factors are the fraction of the population in each of eleven age groups and are often referred to as a standard population. A standard population is no more than an agreed upon set of factors used to assign the same age distribution to each county (or any other area). The factors used in this example represent the age distribution of the United States in the 2000.

The crude death rate per 1,000 population is calculated using the total deaths and total population ❸. The adjusted death rate is initialized at zero prior to using a DO LOOP to calculate the rate (why is this necessary?) ❹. A loop is used to calculate eleven age-specific death rates, multiplying each rate by the appropriate factor from the standard population ❺. The results of these eleven calculations are accumulate in the variable R_ADJUSTED. Once the loop is complete, the rate is multiplied by 1,000 ❻.

```
proc format; ❶
value $county
' ' = 'NEW YORK STATE'
other = [$fip2nam40. ];
run;

proc print data=rates noobs label; ❷
var county d0 p0 r_crude r_adjusted;
format county $county. d0 p0 comma10. r_: 6.1; ❸
run;
```

A format (\$FIP2NAM) is available that will convert FIPS county numbers to county names ❶. However, if the county number is blank, the format will produce the text 'UNKNOWN', though that observation in our data set is that for New York State. A new format (\$COUNTY) is created that will convert a blank entry to the text 'NEW YORK STATE', but use the stored format to convert all other values of the variable county to text. PROC PRINT displays the rates ❷ with the new format \$COUNTY used to dispoaly county names ❸.

COUNTY	DEATHS	POPULATION	R_CRUDE	R_ADJUSTED
NEW YORK STATE	157,394	18,976,457	8.3	8.1
ALBANY	2,971	294,565	10.1	8.7
ALLEGANY	487	49,927	9.8	9.0
BRONX	10,138	1,332,650	7.6	9.0
BROOME	2,223	200,536	11.1	8.6
CATTARAUGUS	836	83,955	10.0	8.9

We have completed the task, but it might be interesting to see what counties have the highest (or lowest) crude and age-adjusted death rates.

```
proc rank data=rates out=ranks descending ties=low; ❶
where county ne ' '; ❷
var r_crude r_adjusted; ❸
ranks rank_c rank_a; ❹
run;

proc print data=ranks noobs label; ❺
var county d0 p0 r_crude r_adjusted rank_c rank_a;
format county $county. d0 p0 comma10. r_: 6.1;
run;
```

PROC RANK is used to assign a rank to the two death rates in each county ❶. Normally, PROC RANK will assign ranks in ascending order, the lowest rate gets the lowest rank of 1. The DESCENDING option produces ranks in the reverse order, the highest rate gets the lowest rank of 1. The TIES options pecifies how to treat like ranks. TIES=LOW will assign the same rank to counties with like rates, and that rate will be a value closer to the lowest rank of 1. Since we do not want the overall state rank involved in the ranking, a WHERE statement is used to eliminate that observation ❷. Only variables in the VAR statement will have ranks assigned ❸. Otherwise, all numeric variables are ranked. A RANKS statement specifies that the ranks are stored in two new variables ❹. Without a RANKS statement, the values of the ranks would replace the values of the rates in the variables R_CRUDE and R_ADJUSTED.

COUNTY	DEATHS	POPULATION	R_CRUDE	R_ADJUSTED	Rank for Variable R_CRUDE	Rank for Variable R_ADJUSTED
ALBANY	2,971	294,565	10.1	8.7	21	26
ALLEGANY	487	49,927	9.8	9.0	26	11
BRONX	10,138	1,332,650	7.6	9.0	53	13
BROOME	2,223	200,536	11.1	8.6	8	29
CATTARAUGUS	836	83,955	10.0	8.9	24	16
CAYUGA	732	81,963	8.9	7.8	35	53
CHAUTAUQUA	1,535	139,750	11.0	8.7	9	25
CHEMUNG	959	91,070	10.5	8.8	15	18
CHENANGO	529	51,401	10.3	8.8	17	20
CLINTON	620	79,894	7.8	8.4	52	35
COLUMBIA	745	63,094	11.8	9.1	3	8
CORTLAND	427	48,599	8.8	8.7	36	24
DELAWARE	565	48,055	11.8	8.4	4	39
DUTCHESS	2,203	280,150	7.9	8.2	49	48
ERIE	10,069	950,265	10.6	8.8	13	19
ESSEX	397	38,851	10.2	8.4	18	38
FRANKLIN	466	51,134	9.1	9.0	32	12

Look at the ranks, for example Delaware County. The crude death rate is the 4th highest in the state. After adjustment, the county rate is ranked 34th. Franklin County moved in the other direction. What does this say about the age distribution in the two counties?

The following is another method of producing a SAS data set from a procedure. The method can be used with any procedure.

```
ods trace on; ❶
proc freq data=deaths; ❷
table county*age;
where age ne .;
format age age.;
run;
ods trace off; ❸
```

ODS (the output delivery system) is used. ODS is asked to produce a TRACE of all procedure output ❶. The output of each portion of each SAS PROC (almost 100% true) has a name and the TRACE option results in these names being written to the SAS LOG. PROC FREQ is used to produce counts of deaths by county and age group ❷. Another ODS statements turn off the tracing ❸.

In the LOG, we can see that the name of the PROC FREQ output is CROSSTABFREQS.

```
ods listing close; ❶
ods output crosstabfreqs=dtab4; ❷
proc freq data=deaths; ❸
table county*age;
where age ne .;
format age age.;
run;
ods listing; ❹
proc print data=dtab4; ❺
run;
```

ODS is used to close the LISTING destination and no output is written to the OUTPUT window ❶. An OUTPUT option on an ODS statement requests that CROSSTABSFREQ data be placed in data set DTAB4 ❷. PROC FREQ is run ❸. No table is produced in the OUTPUT window, but a SAS data set (DTAB4) is created. The ODS LISTING destination is reopened ❹, and data set DTAB4 is displayed ❺.

TABLE	COUNTY	AGE	_TYPE_	_TABLE_	FREQUENCY	PERCENT	ROWPERCENT	COLPERCENT	MISSING
Table COUNTY * AGE	01	1	11	1	28	0.01781	0.9447	1.72096	.
Table COUNTY * AGE	01	2	11	1	2	0.00127	0.0675	0.76628	.
Table COUNTY * AGE	01	3	11	1	7	0.00445	0.2362	2.00000	.
Table COUNTY * AGE	01	4	11	1	23	0.01463	0.7760	1.55510	.
Table COUNTY * AGE	01	5	11	1	23	0.01463	0.7760	0.98417	.
Table COUNTY * AGE	01	6	11	1	74	0.04706	2.4966	1.35259	.
Table COUNTY * AGE	01	7	11	1	159	0.10111	5.3644	1.52708	.
Table COUNTY * AGE	01	8	11	1	245	0.15580	8.2659	1.57223	.
Table COUNTY * AGE	01	9	11	1	468	0.29761	15.7895	1.64511	.
Table COUNTY * AGE	01	10	11	1	951	0.60476	32.0850	2.09158	.
Table COUNTY * AGE	01	11	11	1	984	0.62574	33.1984	2.14767	.

```

Table COUNTY * AGE 01 . 10 1 2964 1.88486
Table COUNTY * AGE 02 1 11 1 6 0.00382 1.2320 0.36878 <more
output>
Table COUNTY * AGE 19 10 11 1 135 0.08585 24.6801 0.29691 .
Table COUNTY * AGE 19 11 11 1 163 0.10365 29.7989 0.35576 .
Table COUNTY * AGE 19 . 10 1 547 0.34785 . . .
Table COUNTY * AGE 20 1 11 1 0 0.00000 0.0000 0.00000 .
Table COUNTY * AGE 20 2 11 1 0 0.00000 0.0000 0.00000 .
Table COUNTY * AGE 20 3 11 1 0 0.00000 0.0000 0.00000 .
Table COUNTY * AGE 20 4 11 1 0 0.00000 0.0000 0.00000 .
Table COUNTY * AGE 20 5 11 1 0 0.00000 0.0000 0.00000 .
Table COUNTY * AGE 20 6 11 1 2 0.00127 2.7778 0.03656 .
Table COUNTY * AGE 20 7 11 1 4 0.00254 5.5556 0.03842 .
<more output>
Table COUNTY * AGE 97 8 11 1 385 0.2448 11.1433 2.4706 .
Table COUNTY * AGE 97 9 11 1 668 0.4248 19.3343 2.3481 .
Table COUNTY * AGE 97 10 11 1 983 0.6251 28.4515 2.1620 .
Table COUNTY * AGE 97 11 11 1 907 0.5768 26.2518 1.9796 .
Table COUNTY * AGE 97 . 10 1 3455 2.1971 . .
Table COUNTY * AGE 1 01 1 1627 1.0346 . .
Table COUNTY * AGE 2 01 1 261 0.1660 . .
Table COUNTY * AGE 3 01 1 350 0.2226 . .
Table COUNTY * AGE 4 01 1 1479 0.9405 . .
Table COUNTY * AGE 5 01 1 2337 1.4861 . .
Table COUNTY * AGE 6 01 1 5471 3.4791 . .
Table COUNTY * AGE 7 01 1 10412 6.6212 . .
Table COUNTY * AGE 8 01 1 15583 9.9095 . .
Table COUNTY * AGE 9 01 1 28448 18.091 . .
Table COUNTY * AGE 10 01 1 45468 28.914 . .
Table COUNTY * AGE 11 01 1 45817 29.136 . .
Table COUNTY * AGE . 00 1 157253 100.000 . . 0
    
```

There are many more variables in this data set than were produced using the OUT= option in PROC FREQ when producing death counts by COUNTY and AGE group. There are also more observations. The extra observations contain county-specific totals, age-specific totals, and an overall state total (similar to the output in PROC SUMMARY when CLASS variables are used). Also notice that there are observations that have FREQUENCY=0. In previous examples, PROC SUMMARY with the COMPLETETYPES and PRELOADFMT options and PROC FREQ with a SPARSE option were used to add zero frequencies to output data sets.

```

ods listing close;
ods output crosstabfreqs=dtab5 (keep=county age frequency); ❶

proc freq data=deaths;
table county*age;
where age ne .;
format age age.;
run;

proc sort data=dtab5; ❷
by county;
run;

proc transpose data=dtab5 out=dtrans5 (drop=_) prefix=d; ❸
var frequency;
id age;
by county;
where age ne .; ❹
run;

ods listing;

proc print data=dtrans5 noobs;
run;
    
```

We can limit the variables placed in the new data set using a KEEP option ❶. After PROC FREQ is used to produce the counts, the new data set is sorted by county ❷. Remember that the observations with blank county numbers (state totals) appeared at the end of the data set. PROC TRANSPOSE is used to create a data set with one observation per county plus an observation with a state total ❸. A WHERE statement eliminates the observations with no value for the variable AGE (the county totals) ❹. Once again, we have a data set that can be used to calculate rates (after merging with populations).

COUNTY	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11
01	1632	262	351	1485	2343	5481	10424	15596	28471	45501	45848
02	28	2	7	23	23	74	159	247	470	954	984
03	6	1	2	5	2	18	26	49	108	136	134
04	21	1	2	9	11	42	102	189	364	747	735
04	6	1	1	11	9	25	63	87	158	264	211