Creating a Basic Report

To produce a simple list report, you first reference the library where your SAS data set is stored. You can also set system options to control the appearance of your reports. Then you submit a PROC PRINT step.

Syntax, PROC PRINT step:

PROC PRINT DATA=*SAS-data-set***;**
RUN;

SAS-data-set is the name of the SAS data set to be printed.

In the program below, the PROC PRINT statement invokes the PRINT procedure and specifies the data set Therapy in the SAS library to which the libref Cert has been assigned.

```
libname cert 'C:\Users\Student1\Cert';
proc print data=cert.therapy;
run;
```

Notice the layout of the resulting report below. These are the default behaviors:

- All observations and variables in the data set are printed.
- A column for observation numbers appears on the far left.
- Variables and observations appear in the order in which they occur in the data set.

Figure 6.1 *Cert.Therapy Data Set (partial output)*

Obs	Date	AerClass	WalkJogRun	Swim
1	JAN2012	56	78	14
2	FEB2012	32	109	19
3	MAR2012	35	106	22
4	APR2012	47	115	24
5	MAY2012	55	121	31

Selecting Variables

The VAR Statement

By default, PROC PRINT lists all the variables in a data set. You can select variables and control the order in which they appear by using a VAR statement.

Syntax, VAR statement:

VAR *variable(s)***;**

variable(s) is one or more variable names, separated by blanks.

For example, the following VAR statement specifies that only the variables Age, Height, Weight, and Fee be printed, in that order:

```
proc print data=cert.admit;
  var age height weight fee;
run;
```

The procedure output from the PROC PRINT step with the VAR statement lists only the values for those variables.

Figure 6.2 *PRINT Procedure Output*

The SAS System

Obs	Age	Height	Weight	Fee
1	27	72	168	85.20
2	34	66	152	124.80
3	31	61	123	149.75
4	43	63	137	149.75
5	51	71	158	124.80
6	29	76	193	124.80
7	32	67	151	149.75
8	35	70	173	149.75
9	34	73	154	124.80
10	49	64	172	124.80
11	44	66	140	149.75
12	28	62	118	85.20
13	30	69	147	149.75
14	40	69	163	124.80
15	47	72	173	124.80
16	60	71	191	149.75
17	43	65	123	124.80
18	25	75	188	85.20
19	22	63	139	85.20
20	41	67	141	149.75
21	54	71	183	149.75

Removing the OBS Column

In addition to selecting variables, you can suppress observation numbers.

To remove the Obs column, specify the NOOBS option in the PROC PRINT statement.

```
proc print data=work.example noobs;
  var age height weight fee;
run;
```

Figure 6.3 *PRINT Procedure Output with No Observation Numbers*

The SAS System

Age	Height	Weight	Fee
27	72	168	85.20
34	66	152	124.80
31	61	123	149.75
43	63	137	149.75
51	71	158	124.80
29	76	193	124.80
32	67	151	149.75
35	70	173	149.75
34	73	154	124.80
49	64	172	124.80
44	66	140	149.75
28	62	118	85.20
30	69	147	149.75
40	69	163	124.80
47	72	173	124.80
60	71	191	149.75
43	65	123	124.80
25	75	188	85.20
22	63	139	85.20
41	67	141	149.75
54	71	183	149.75

Identifying Observations

Using the ID Statement in PROC PRINT

The ID statement identifies observations using variable values, such as an identification number, instead of observation numbers.

Syntax, ID statement in the PRINT procedure:

ID *variable(s)*;

variable(s) specifies one or more variables to print whose value is used instead of the observation number at the beginning of each row of the report.

Example: ID Statement

In the following example, the OBS column in the output is replaced with the variable values for IDnum and LastName.

```
proc print data=cert.reps;
  id idnum lastname;
run;
```

Here is the output produced by PROC PRINT:

Figure 6.4 *PROC PRINT: ID Statement Output*

IDnum	LastName	FirstName	City	State	Sex	JobCode	Salary	Birth	Hired	HomePhone
1269	CASTON	FRANKLIN	STAMFORD	CT	M	NA1	41690.00	06MAY60	01DEC80	203/781-3335
1935	FERNANDEZ	KATRINA	BRIDGEPORT	CT		NA2	51081.00	31MAR42	19OCT69	203/675-2962
1417	NEWKIRK	WILLIAM	PATERSON	NJ	,	NA2	52270.00	30JUN52	10MAR77	201/732-6611
1839	NORRIS	DIANE	NEW YORK	YN	F	NA1	43433.00	02DEC58	06JUL81	718/384-1767
1111	RHODES	JEREMY	PRINCETON	NJ	M	NA1	40586.00	17JUL61	03NOV80	201/812-1837
1352	RIVERS	SIMON	NEW YORK	NY	M	NA2	5379.80	05DEC48	19OCT74	718/383-3345
1332	STEPHENSON	ADAM	BRIDGEPORT	CT	M	NA1	42178.00	20SEP58	07JUN79	203/675-1497
1443	WELLS	AGNES	STAMFORD	CT	F	NA1	422.74	20NOV56	01SEP79	203/781-5546

Example: ID and VAR Statement

You can use the ID and VAR statement together to control which variables are printed and in which order. If a variable in the ID statement also appears in the VAR statement, the output contains two columns for that variable.

```
proc print data=cert.reps;
  id idnum lastname;                /*#1*/
  var idnum sex jobcode salary;     /*#2*/
run;
```

1 The ID statement replaces the OBS column in the output with the IDnum and LastName variable values.

2 The VAR statement selects the variables that appear in the output and determines the order.

The variable IDnum appeared in both the ID statement and the VAR statement. Therefore, IDnum appears twice in the output.

Output 6.1 *PROC PRINT: ID and VAR Statement Output*

IDnum	LastName	IDnum	Sex	JobCode	Salary
1269	CASTON	1269	M	NA1	41690.00
1935	FERNANDEZ	1935		NA2	51081.00
1417	NEWKIRK	1417	,	NA2	52270.00
1839	NORRIS	1839	F	NA1	43433.00
1111	RHODES	1111	M	NA1	40586.00
1352	RIVERS	1352	M	NA2	5379.80
1332	STEPHENSON	1332	M	NA1	42178.00
1443	WELLS	1443	F	NA1	422.74

Selecting Observations

By default, a PROC PRINT step lists all the observations in a data set. You can control which observations are printed by adding a WHERE statement to your PROC PRINT

step. There should be only one WHERE statement in a step. If multiple WHERE statements are issued, only the last statement is processed.

Syntax, WHERE statement:

WHERE *where-expression*;

where-expression specifies a condition for selecting observations. The *where-expression* can be any valid SAS expression.

Example Code 1 *Using the WHERE Statement in PROC PRINT*

```
proc print data=cert.admit;
  var age height weight fee;          /*#1*/
  where age>30;                       /*#2*/
run;
```

1 The VAR statement selects the variables Age, Height, Weight, and Fee and displays them in the output in that order.

2 The WHERE statement selects only the observations for which the value of Age is greater than 30 and prints them in the output.

The following output displays only the observations where the value of Age is greater than 30.

Figure 6.5 *PROC PRINT Output with a WHERE Statement*

The SAS System

Obs	Age	Height	Weight	Fee
2	34	66	152	124.80
3	31	61	123	149.75
4	43	63	137	149.75
5	51	71	158	124.80
7	32	67	151	149.75
8	35	70	173	149.75
9	34	73	154	124.80
10	49	64	172	124.80
11	44	66	140	149.75
14	40	69	163	124.80
15	47	72	173	124.80
16	60	71	191	149.75
17	43	65	123	124.80
20	41	67	141	149.75
21	54	71	183	149.75

Specifying WHERE Expressions

In the WHERE statement, you can specify any variable in the SAS data set, not just the variables that are specified in the VAR statement. The WHERE statement works for both character and numeric variables. To specify a condition based on the value of a character variable, follow these rules:

- Enclose the value in quotation marks.

- Write the value with lowercase, uppercase, or mixed case letters exactly as it appears in the data set.

You use the following comparison operators to express a condition in the WHERE statement:

Table 6.1 *Comparison Operators in a WHERE Statement*

Symbol	Meaning	Sample Program Code
= or eq	equal to	`where name='Jones, C.';`
^= or ne	not equal to	`where temp ne 212;`
> or gt	greater than	`where income>20000;`
< or lt	less than	`where partno lt "BG05";`
>= or ge	greater than or equal to	`where id>='1543';`
<= or le	less than or equal to	`where pulse le 85;`

For more information about valid SAS expressions, see Chapter 4, “Creating SAS Data Sets,” on page 31.

Using the CONTAINS Operator

The CONTAINS operator selects observations that include the specified substring. The symbol for the CONTAINS operator is **?**. You can use either the CONTAINS keyword or the symbol in your code, as shown below.

```
where firstname CONTAINS 'Jon';
where firstname ? 'Jon';
```

Specifying Compound WHERE Expressions

You can also use WHERE statements to select observations that meet multiple conditions. To link a sequence of expressions into compound expressions, you use logical operators, including the following:

Table 6.2 *Compound WHERE Expression Operators*

Operator, Symbol		Description
AND	&	and, both. If both expressions are true, then the compound expression is true.
OR	|	or, either. If either expression is true, then the compound expression is true.

Examples of WHERE Statements

- You can use compound expressions like these in your WHERE statements:

```
where age<=55 and pulse>75;
where area='A' or region='S';
where ID>'1050' and state='NC';
```

- When you test for multiple values of the same variable, you specify the variable name in each expression:

```
where actlevel='LOW' or actlevel='MOD';
where fee=124.80 or fee=178.20;
```

- You can use the IN operator as a convenient alternative:

```
where actlevel in ('LOW','MOD');
where fee in (124.80,178.20);
```

- To control how compound expressions are evaluated, you can use parentheses (expressions in parentheses are evaluated first):

```
where (age<=55 and pulse>75) or area='A';
where age<=55 and (pulse>75 or area='A');
```

Using System Options to Specify Observations

SAS system options set the preferences for a SAS session. You can use the FIRSTOBS= and OBS= options in an OPTIONS statement to specify the observations to process from SAS data sets.

Specify either or both of these options as needed:

- FIRSTOBS= starts processing at a specific observation.
- OBS= stops processing after a specific observation.

Note: Using FIRSTOBS= and OBS= together processes a specific group of observations.

Syntax, FIRSTOBS=, and OBS= options in an OPTIONS statement:

FIRSTOBS=*n*

OBS=*n*

n is a positive integer. For FIRSTOBS=, *n* specifies the number of the *first* observation to process. For OBS=, *n* specifies the number of the *last* observation to process. By default, FIRSTOBS=1. The default value for OBS= is MAX, which is the largest signed, 8-byte integer that is representable in your operating environment. The number can vary depending on your operating system.

To reset the number of the last observation to process, you can specify OBS=MAX in the OPTIONS statement.

```
options obs=max;
```

This instructs any subsequent SAS programs in the SAS session to process through the last observation in the data set that is being read.

CAUTION:

Each of these options applies to every input data set that is used in a program or a SAS process because a system option sets the preference for the SAS session.

Examples: FIRSTOBS= and OBS= Options

The following examples use the data set Cert.Heart, which contains 20 observations and 8 variables.

Example Code 2 *Using the FIRSTOBS= Option*

```
options firstobs=10;          /*#1*/
proc print data=cert.heart;   /*#2*/
run;
```

1 Use the OPTIONS statement to specify the FIRSTOBS= option. In this example, the FIRSTOBS=10 option enables SAS to read the 10th observation of the data set first and read through the last observation.

2 A total of 11 observations are printed using the PROC PRINT step.

Here is the output:

Figure 6.6 *PROC PRINT Output with FIRSTOBS=10*

Obs	Patient	Sex	Survive	Shock	Arterial	Heart	Cardiac	Urinary
10	509	2	SURV	OTHER	79	84	256	90
11	742	1	DIED	HYPOVOL	100	54	135	0
12	609	2	DIED	NONSHOCK	93	101	260	90
13	318	2	DIED	OTHER	72	81	410	405
14	412	1	SURV	BACTER	61	87	296	44
15	601	1	DIED	BACTER	84	101	260	377
16	402	1	SURV	CARDIO	88	137	312	75
17	98	2	SURV	CARDIO	84	87	260	377
18	4	1	SURV	HYPOVOL	81	149	406	200
19	50	2	SURV	HYPOVOL	72	111	332	12
20	2	2	DIED	OTHER	101	114	424	97

You can specify the FIRSTOBS= and OBS= options together. In the following example, SAS reads only through the 10th observation.

Example Code 3 *Using the FIRSTOBS= and OBS= Options*

```
options firstobs=1 obs=10;      /*#1*/
proc print data=cert.heart;     /*#2*/
run;
```

1 The FIRSTOBS=1 option resets the FIRSTOBS= option to the default value. The default value reads the first observation in the data set. When you specify OBS=10 in the OPTIONS statement, SAS reads through the 10th observation.

2 A total of 10 observations are printed using the PROC PRINT step.

Here is the output:

Figure 6.7 *PROC PRINT Output with FIRSTOBS=1 and Obs=10*

Obs	Patient	Sex	Survive	Shock	Arterial	Heart	Cardiac	Urinary
1	203	1	SURV	NONSHOCK	88	95	66	110
2	54	1	DIED	HYPOVOL	83	183	95	0
3	664	2	SURV	CARDIO	72	111	332	12
4	210	2	DIED	BACTER	74	97	369	0
5	101	2	DIED	NEURO	80	130	291	0
6	102	2	SURV	OTHER	87	107	471	65
7	529	1	DIED	CARDIO	103	106	217	15
8	524	2	DIED	CARDIO	145	99	156	10
9	426	1	SURV	OTHER	68	77	410	75
10	509	2	SURV	OTHER	79	84	256	90

You can also combine FIRSTOBS= and OBS= to process observations in the middle of the data set.

Example Code 4 *Processing Middle Observations of a Data Set*

```
options firstobs=10 obs=15;     /*#1*/
proc print data=cert.heart;     /*#2*/
run;
```

1 When you set FIRSTOBS=10 and OBS=15, the program processes only observations 10 through 15.

2 A total of six observations are printed using the PROC PRINT step.

Here is the output:

Figure 6.8 *PROC PRINT Output with FIRSTOBS=10 and Obs=15*

Obs	Patient	Sex	Survive	Shock	Arterial	Heart	Cardiac	Urinary
10	509	2	SURV	OTHER	79	84	256	90
11	742	1	DIED	HYPOVOL	100	54	135	0
12	609	2	DIED	NONSHOCK	93	101	260	90
13	318	2	DIED	OTHER	72	81	410	405
14	412	1	SURV	BACTER	61	87	296	44
15	601	1	DIED	BACTER	84	101	260	377

Using FIRSTOBS= and OBS= for Specific Data Sets

Using the FIRSTOBS= or OBS= system options determines the first or last observation, respectively, that is read for all steps for the duration of your current SAS session or until you change the setting. However, you can still do the following:

- override these options for a given data set
- apply these options to a specific data set only

To affect any single file, use FIRSTOBS= or OBS= as data set options instead of using them as system options. You specify data set options in parentheses immediately following the input data set name.

TIP A FIRSTOBS= or OBS= specification from a data set option overrides the corresponding FIRSTOBS= or OBS= system option, but only for that DATA step.

Example: FIRSTOBS= and OBS= as Data Set Options

As shown in the following example, this program processes only observations 10 through 15, for a total of 6 observations:

```
options firstobs=10 obs=15;
proc print data=clinic.heart;
run;
```

You can create the same output by specifying FIRSTOBS= and OBS= as data set options, as follows. The data set options override the system options for this instance only.

```
options firstobs=10 obs=15;
proc print data=clinic.heart(firstobs=20 obs=30);
run;
```

To specify FIRSTOBS= or OBS= for this program only, you could omit the OPTIONS statement altogether and simply use the data set options.

Sorting Data

The SORT Procedure

By default, PROC PRINT lists observations in the order in which they appear in your data set. To sort your report based on values of a variable, you must use PROC SORT to sort your data before using the PRINT procedure to create reports from the data.

The SORT procedure does the following:

- rearranges the observations in a SAS data set
- creates a new SAS data set that contains the rearranged observations
- replaces the original SAS data set by default
- can sort on multiple variables
- can sort in ascending or descending order
- treats missing values as the smallest possible values

Note: PROC SORT does not generate printed output.

Syntax, PROC SORT step:

PROC SORT DATA=*SAS-data-set* <OUT=*SAS-data-set*>;

BY <DESCENDING> *BY-variable(s)*;

RUN;

- The DATA= option specifies the data set to be read.
- The OUT= option creates an output data set that contains the data in sorted order.
- *BY-variable(s)* in the required BY statement specifies one or more variables whose values are used to sort the data.
- The DESCENDING option in the BY statement sorts observations in descending order. If you have more that one variable in the BY statement, DESCENDING applies only to the variable that immediately follows it.

CAUTION:

If you do not use the OUT= option, PROC SORT overwrites the data set that is specified in the DATA= option.

Example: PROC SORT

```
proc sort data=cert.admit out=work.wgtadmit;     /* #1 */
  by weight age;
run;
proc print data=work.wgtadmit;                   /* #2 */
  var weight age height fee;                     /* #3 */
  where age>30;                                  /* #4 */
run;
```

1 The PROC SORT step sorts the permanent SAS data set Cert.Admit by the values of the variable Age within the values of the variable Weight. The OUT= option creates the temporary SAS data set Wgtadmit.

2 The PROC PRINT step prints a subset of the Wgtadmit data set.

3 The VAR statement selects only the variables Weight, Age, Height, and Fee to print in the output.

4 The WHERE statement subsets the data by printing only those observations where the values of Age are greater than 30.

The report displays observations in ascending order of Age within Weight.

Figure 6.9 *Observations Displayed in Ascending Order of Age within Weight*

The SAS System

Obs	Weight	Age	Height	Fee
2	123	31	61	149.75
3	123	43	65	124.80
4	137	43	63	149.75
6	140	44	66	149.75
7	141	41	67	149.75
9	151	32	67	149.75
10	152	34	66	124.80
11	154	34	73	124.80
12	158	51	71	124.80
13	163	40	69	124.80
15	172	49	64	124.80
16	173	35	70	149.75
17	173	47	72	124.80
18	183	54	71	149.75
20	191	60	71	149.75

Adding the DESCENDING option to the BY statement sorts observations in ascending order of age within descending order of weight. Notice that DESCENDING applies only to the variable Weight.

```
proc sort data=cert.admit out=work.wgtadmit;
   by descending weight age;
run;
proc print data=work.wgtadmit;
   var weight age height fee;
   where age>30;
run;
```

Figure 6.10 *Observations Displayed in Descending Order*

The SAS System

Obs	Weight	Age	Height	Fee
2	191	60	71	149.75
4	183	54	71	149.75
5	173	35	70	149.75
6	173	47	72	124.80
7	172	49	64	124.80
9	163	40	69	124.80
10	158	51	71	124.80
11	154	34	73	124.80
12	152	34	66	124.80
13	151	32	67	149.75
15	141	41	67	149.75
16	140	44	66	149.75
18	137	43	63	149.75
19	123	31	61	149.75
20	123	43	65	124.80

Applying SAS Formats and Informats

Temporarily Assigning Formats to Variables

In your SAS reports, formats control how the data values are displayed. To make data values more understandable when they are displayed in your procedure output, you can use the FORMAT statement, which associates formats with variables.

Formats affect only how the data values appear in output, not the actual data values as they are stored in the SAS data set.

Syntax, FORMAT statement:

FORMAT *variable(s) format-name*;

- *variable(s)* is the name of one or more variables whose values are to be written according to a particular pattern
- *format-name* specifies a SAS format or a user-defined format that is used to write out the values.

Tip: The FORMAT statement applies only to the PROC step in which it appears.

You can use a separate FORMAT statement for each variable, or you can format several variables (using either the same format or different formats) in a single FORMAT statement.

Table 12.1 *Formats That Are Used to Format Data*

FORMAT Statement	Description	Example
`format date mmddyy8.;`	associates the format MMDDYY8. with the variable Date	01/06/17
`format net comma5.0` `gross comma8.2;`	associates the format COMMA5.0 with the variable Net and the format COMMA8.2 with the variable Gross	1,234 5,678.90
`format net gross dollar9.2;`	associates the format DOLLAR9.2 with both variables, Net, and Gross	$1,234.00 $5,678.90

For example, the FORMAT statement below writes values of the variable Fee using dollar signs, commas, and no decimal places.

```
proc print data=cert.admit;
  var actlevel fee;
  where actlevel='HIGH';
  format fee dollar4.;
run;
```

Figure 12.1 *FORMAT Statement Output*

Obs	ActLevel	Fee
1	HIGH	$85
2	HIGH	$125
6	HIGH	$125
11	HIGH	$150
14	HIGH	$125
18	HIGH	$85
20	HIGH	$150

Specifying SAS Formats

The table below describes some SAS formats that are commonly used in reports.

Table 12.2 *Commonly Used SAS Formats*

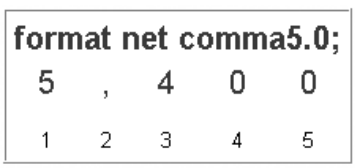
Format	Description	Example
COMMA*w.d*	specifies values that contain commas and decimal places	comma8.2
DOLLAR*w.d*	specifies values that contain dollar signs, commas, and decimal places	dollar6.2
MMDDYY*w*.	specifies values as date values of the form 09/12/17 (MMDDYY8.) or 09/12/2017 (MMDDYY10.)	mmddyy10.
w.	specifies values that are rounded to the nearest integer in *w* spaces	7.
w.d	specifies values that are rounded to *d* decimal places in *w* spaces	8.2
$w.	specifies values as character values in *w* spaces	$12.
DATE*w*.	specifies values as date values of the form 16OCT17 (DATE7.) or 16OCT2017 (DATE9.)	date9.

Field Widths

All SAS formats specify the total field width (*w*) that is used for displaying the values in the output. For example, suppose the longest value for the variable Net is a four-digit number, such as 5400. To specify the COMMA*w.d* format for Net, you specify a field width of 5 or more. You must count the comma, because it occupies a position in the output.

Note: When you use a SAS format, specify a field width (*w*) that is wide enough for the largest possible value. Otherwise, values might not be displayed properly.

Figure 12.2 *Specifying a Field Width (w) with the FORMAT Statement*

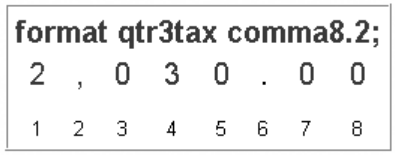


Decimal Places

For numeric variables, you can also specify the number of decimal places (*d*), if any, to be displayed in the output. Numbers are rounded to the specified number of decimal places. In the example above, no decimal places are displayed.

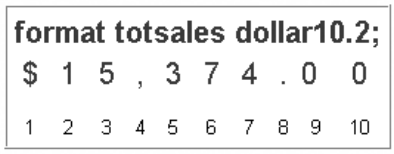
Writing the whole number 2030 as 2,030.00 requires eight print positions, including two decimal places and the decimal point.

Figure 12.3 *Whole Number Decimal Places*



Formatting 15374 with a dollar sign, commas, and two decimal places requires 10 print positions.

Figure 12.4 *Specifying 10 Decimal Places*



Examples: Data Values and Formats

This table shows you how data values are displayed when different format, field width, and decimal place specifications are used.

Table 12.3 *Displaying Data Values with Formats*

Stored Value	Format	Displayed Value
38245.3975	COMMA9.2	38,245.40
38245.3975	8.2	38245.40
38245.3975	DOLLAR10.2	$38,245.40
38245.3975	DOLLAR9.2	$38245.40
38245.3975	DOLLAR8.2	38245.40
0	MMDDYY8.	01/01/60
0	MMDDYY10.	01/01/1960
0	DATE7.	01JAN60

Stored Value	Format	Displayed Value
0	DATE9.	01JAN1960

TIP If a format is too small, the following message is written to the SAS log:

```
NOTE: At least one W.D format was too small for the number to be
printed. The decimal might be shifted by the 'BEST' format.
```

The MEANS Procedure

What Does the MEANS Procedure Do?

The MEANS procedure provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations. For example, PROC MEANS does the following:

- calculates descriptive statistics based on moments
- estimates quantiles, which includes the median
- calculates confidence limits for the mean
- identifies extreme values
- performs a *t* test

By default, PROC MEANS displays output.

MEANS Procedure Syntax

The MEANS procedure can include many statements and options for specifying statistics.

Syntax, MEANS procedure:

PROC MEANS <DATA=*SAS-data-set*>
<*statistic-keyword(s)*> <*option(s)*>**;**

RUN;

- *SAS-data-set* is the name of the data set to be analyzed.
- *statistic-keyword(s)* specify the statistics to compute.
- *option(s)* control the content, analysis, and appearance of output.

Example: Default PROC MEANS Output

In its simplest form, PROC MEANS prints the *n*-count (number of non missing values), the mean, the standard deviation, and the minimum and maximum values of every numeric variable in a data set.

```
proc means data=cert.survey;
run;
```

Output 15.1 *PROC MEANS Output of Cert.Survey*

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Item1	4	3.7500000	1.2583057	2.0000000	5.0000000
Item2	4	3.0000000	1.6329932	1.0000000	5.0000000
Item3	4	4.2500000	0.5000000	4.0000000	5.0000000
Item4	4	3.5000000	1.2909944	2.0000000	5.0000000
Item5	4	3.0000000	1.6329932	1.0000000	5.0000000
Item6	4	3.7500000	1.2583057	2.0000000	5.0000000
Item7	4	3.0000000	1.8257419	1.0000000	5.0000000
Item8	4	2.7500000	1.5000000	1.0000000	4.0000000
Item9	4	3.0000000	1.4142136	2.0000000	5.0000000
Item10	4	3.2500000	1.2583057	2.0000000	5.0000000
Item11	4	3.0000000	1.8257419	1.0000000	5.0000000
Item12	4	2.7500000	0.5000000	2.0000000	3.0000000
Item13	4	2.7500000	1.5000000	1.0000000	4.0000000
Item14	4	3.0000000	1.4142136	2.0000000	5.0000000
Item15	4	3.0000000	1.6329932	1.0000000	5.0000000
Item16	4	2.5000000	1.9148542	1.0000000	5.0000000
Item17	4	3.0000000	1.1547005	2.0000000	4.0000000
Item18	4	3.2500000	1.2583057	2.0000000	5.0000000

Specifying Descriptive Statistics Keywords

The default statistics in the MEANS procedure are *n*-count (number of nonmissing values), the mean, the standard deviation, and the minimum and maximum values of every numeric variable in a data set. However, you might need to compute a different statistic such as median or range of the values. Use the statistic keyword option in the PROC MEANS statement to specify one or more statistics to display in the output.

Here are the available keywords in the PROC statement:

Table 15.1 *Descriptive Statistics Keywords*

Keyword	Description
CLM	The two-sided confidence limit for the mean.
CSS	The sum of squares corrected for the mean.
CV	The percent coefficient of variation.
KURTOSIS | KURT	Measures the heaviness of tails.
LCLM	The one-sided confidence limit below the mean.
MAX	The maximum value.
MEAN	The arithmetic mean or average of all the values.
MIN	The minimum value.
MODE	The value that occurs most frequently.
N	The number of observations with nonmissing values.
NMISS	The number of observations with missing values.
RANGE	Calculated as the difference between the maximum value and the minimum value.
SKEWNESS | SKEW	Measures the tendency of the deviations to be larger in one direction than in the other.
STDDEV | STD	Is the standard deviation s and is computed as the square root of the variance.
STDERR | STDMEAN	The standard error of the mean.
SUM	Sum
SUMWGT	The sum of the weights.

Keyword	Description
UCLM	The one-sided confidence limit above the mean
USS	The value of the uncorrected sum of squares.
VAR	Variance.

Table 15.2 *Quantile Statistic Keywords*

Keyword	Description
MEDIAN | P50	The middle value or the 50th percentile.
P1	1st percentile.
P5	5th percentile.
P10	10th percentile.
Q1 | P25	The lower quartile or 25th percentile.
Q3 | P75	The upper quartile or 75th percentile.
P90	90th percentile.
P95	95th percentile.
P99	99th percentile.
QRANGE	The interquartile range and is calculated as the difference between the upper and lower quartile, Q3– Q1.

Table 15.3 *Hypothesis Testing Keywords*

Keyword	Description
PROBT | PRT	The two-tailed *p*-value for Student's *t* statistic, T, with $n - 1$ degrees of freedom. This value is the probability under the null hypothesis of obtaining a more extreme value of T than is observed in this sample.
T	The Student's *t* statistic to test the null hypothesis that the population mean is equal to μ_0 and is calculated as $\frac{\overline{X} - \mu_0}{8/\sqrt{\sum w_i}}$

Example: Specifying Statistic Keywords

To determine the median and range of Cert.Survey numeric values, add the MEDIAN and RANGE keywords as options.

```
proc means data=cert.survey median range;
run;
```

Output 15.2 *PROC MEANS Output of Cert.Survey Displays Only Median and Range*

Variable	Median	Range
Item1	4.0000000	3.0000000
Item2	3.0000000	4.0000000
Item3	4.0000000	1.0000000
Item4	3.5000000	3.0000000
Item5	3.0000000	4.0000000
Item6	4.0000000	3.0000000
Item7	3.0000000	4.0000000
Item8	3.0000000	3.0000000
Item9	2.5000000	3.0000000
Item10	3.0000000	3.0000000
Item11	3.0000000	4.0000000
Item12	3.0000000	1.0000000
Item13	3.0000000	3.0000000
Item14	2.5000000	3.0000000
Item15	3.0000000	4.0000000
Item16	2.0000000	4.0000000
Item17	3.0000000	2.0000000
Item18	3.0000000	3.0000000

Limiting Decimal Places with MAXDEC= Option

By default, PROC MEANS uses the BEST*w*. format to display numeric values in the report.

When there is no format specification, SAS chooses the format that provides the most information about the value according to the available field width. At times, this can result in unnecessary decimal places, making your output hard to read. To limit decimal places, use the MAXDEC= option in the PROC MEANS statement, and set it equal to the length that you prefer.

Syntax, PROC MEANS statement with MAXDEC= option:

PROC MEANS <DATA=*SAS-data-set*>
<*statistic-keyword(s)*> **MAXDEC=***n***;**

n specifies the maximum number of decimal places.

```
proc means data=cert.diabetes min max maxdec=0;
run;
```

Output 15.3 *PROC MEANS Output of Cert.Diabetes with the MAXDEC= Option*

Variable	Minimum	Maximum
ID	1128	9723
Age	15	63
Height	61	75
Weight	102	240
Pulse	65	100
FastGluc	152	568
PostGluc	206	625

Specifying Variables Using the VAR Statement

By default, the MEANS procedure generates statistics for every numeric variable in a data set. But the typical focus is on just a few variables, particularly if the data set is large. It also makes sense to exclude certain types of variables. The values of a numeric identifier variable ID, for example, are unlikely to yield useful statistics.

To specify the variables that PROC MEANS analyzes, add a VAR statement and list the variable names.

Syntax, VAR statement:

VAR *variable(s)*;

variable(s) lists numeric variables for which to calculate statistics.

```
proc means data=cert.diabetes min max maxdec=0;
  var age height weight;
run;
```

Output 15.4 *Specifying Variables in the PROC MEANS Output of Cert.Diabetes*

Variable	Minimum	Maximum
Age	15	63
Height	61	75
Weight	102	240

In addition to listing variables separately, you can use a numbered range of variables.

```
proc means data=cert.survey mean stderr maxdec=2;
  var item1-item5;
run;
```

Output 15.5 *PROC MEANS Output of Cert.Survey with Variable Range*

Variable	Mean	Std Error
Item1	3.75	0.63
Item2	3.00	0.82
Item3	4.25	0.25
Item4	3.50	0.65
Item5	3.00	0.82

Group Processing Using the CLASS Statement

You often want statistics for groups of observations, rather than for the entire data set. For example, census numbers are more useful when grouped by region than when viewed as a national total. To produce separate analyses of grouped observations, add a CLASS statement to the MEANS procedure.

Syntax, CLASS statement:

CLASS *variable(s)*;

variable(s) specifies category variables for group processing.

CLASS variables are used to categorize data. CLASS variables can be either character or numeric, but they should contain a limited number of discrete values that represent meaningful groupings. If a CLASS statement is used, then the N Obs statistic is calculated. The N Obs statistic is based on the CLASS variables, as shown in the output below.

The output of the program shown below is grouped by values of the variables Survive and Sex. The order of the variables in the CLASS statement determines their order in the output table.

```
proc means data=cert.heart maxdec=1;
  var arterial heart cardiac urinary;
  class survive sex;
run;
```

Output 15.6 *PROC MEANS Output Grouped by Values of Variables*

Survive	Sex	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
DIED	1	4	Arterial	4	92.5	10.5	83.0	103.0
			Heart	4	111.0	53.4	54.0	183.0
			Cardiac	4	176.8	75.2	95.0	260.0
			Urinary	4	98.0	186.1	0.0	377.0
	2	6	Arterial	6	94.2	27.3	72.0	145.0
			Heart	6	103.7	16.7	81.0	130.0
			Cardiac	6	318.3	102.6	156.0	424.0
			Urinary	6	100.3	155.7	0.0	405.0
SURV	1	5	Arterial	5	77.2	12.2	61.0	88.0
			Heart	5	109.0	32.0	77.0	149.0
			Cardiac	5	298.0	139.8	66.0	410.0
			Urinary	5	100.8	60.2	44.0	200.0
	2	5	Arterial	5	78.8	6.8	72.0	87.0
			Heart	5	100.0	13.4	84.0	111.0
			Cardiac	5	330.2	87.0	256.0	471.0
			Urinary	5	111.2	152.4	12.0	377.0

Group Processing Using the BY Statement

Like the CLASS statement, the BY statement specifies variables to use for categorizing observations.

Syntax, BY statement:

BY *variable(s)***;**

variable(s) specifies category variables for group processing.

But BY and CLASS differ in two key ways:

- Unlike CLASS processing, BY-group processing requires that your data already be sorted or indexed in the order of the BY variables. Unless data set observations are already sorted, you must run the SORT procedure before using PROC MEANS with any BY group.

  CAUTION:
  If you do not specify an output data set by using the OUT= option, PROC SORT overwrites the initial data set with newly sorted observations.

- The layout of BY-group results differs from the layout of CLASS group results. Note that the BY statement in the program below creates four small tables; a CLASS statement would produce a single large table.

```
proc sort data=cert.heart out=work.heartsort;
  by survive sex;
run;
proc means data=work.heartsort maxdec=1;
  var arterial heart cardiac urinary;
  by survive sex;
run;
```

Figure 15.1 *BY Groups Created by PROC MEANS*

Survive=DIED Sex=1

Variable	N	Mean	Std Dev	Minimum	Maximum
Arterial	4	92.5	10.5	83.0	103.0
Heart	4	111.0	53.4	54.0	183.0
Cardiac	4	176.8	75.2	95.0	260.0
Urinary	4	98.0	186.1	0.0	377.0

Survive=DIED Sex=2

Variable	N	Mean	Std Dev	Minimum	Maximum
Arterial	6	94.2	27.3	72.0	145.0
Heart	6	103.7	16.7	81.0	130.0
Cardiac	6	318.3	102.6	156.0	424.0
Urinary	6	100.3	155.7	0.0	405.0

Survive=SURV Sex=1

Variable	N	Mean	Std Dev	Minimum	Maximum
Arterial	5	77.2	12.2	61.0	88.0
Heart	5	109.0	32.0	77.0	149.0
Cardiac	5	298.0	139.8	66.0	410.0
Urinary	5	100.8	60.2	44.0	200.0

Survive=SURV Sex=2

Variable	N	Mean	Std Dev	Minimum	Maximum
Arterial	5	78.8	6.8	72.0	87.0
Heart	5	100.0	13.4	84.0	111.0
Cardiac	5	330.2	87.0	256.0	471.0
Urinary	5	111.2	152.4	12.0	377.0

TIP The CLASS statement is easier to use than the BY statement because it does not require a sorting step. However, BY-group processing can be more efficient when your categories might contain many levels.

Creating a Summarized Data Set Using the OUTPUT Statement

To write summary statistics to a new data set, use the OUTPUT statement in the MEANS procedure.

Syntax, OUTPUT statement:

OUTPUT OUT=*SAS-data-set statistic=variable(s)***;**

- OUT= specifies the name of the output data set.
- *statistic*= specifies which statistic to store in the output data set.
- *variable(s)* specifies the names of the variables to create. These variables represent the statistics for the analysis variables that are listed in the VAR statement.

Tip: You can use multiple OUTPUT statements to create several OUT= data sets.

The OUTPUT statement writes statistics to a new SAS data set. By default, the default summary statistics are produced for all numeric variables or for the variables specified in the VAR statement. To specify specific statistics to be produced in the new SAS data set, specify *output-statistic-specification= variable-name* in the OUTPUT statement.

The following example creates a PROC MEANS report.

```
proc means data=cert.diabetes;
  var age height weight;                                  /*#1*/
  class sex;                                              /*#2*/
  output out=work.diabetes_by_gender                      /*#3*/
    mean=AvgAge AvgHeight AvgWeight
    min=MinAge MinHeight MinWeight;
run;
proc print data=work.diabetes_by_gender noobs;            /*#4*/
  title1 'Diabetes Results by Gender';
run;
```

1 Specify the analysis variables. The VAR statement specifies that PROC MEANS calculate the default statistics on the Age, Height, and Weight variables.

2 Specify subgroups for the analysis. The CLASS statement separates the analysis by the values of Sex.

3 Specify the output data set options. The OUTPUT statement creates the Work.Diabetes_By_Gender data set and writes the mean value to the new variables AvgAge, AvgHeight, and AvgWeight. The statement also writes the min value to the new variables, MinAge, MinHeight, and MinWeight.

4 Print the output data set Work.Diabetes_By_Gender. The NOOBS option suppresses the observation numbers.

The following output is of Cert.Diabetes from the MEANS procedure.

Output 15.7 *PROC MEANS Output of Cert.Diabetes*

Sex	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
F	11	Age	11	48.9090909	13.3075508	16.0000000	63.0000000
		Height	11	63.9090909	2.1191765	61.0000000	68.0000000
		Weight	11	150.4545455	18.4464828	102.0000000	168.0000000
M	9	Age	9	44.0000000	12.3895117	15.0000000	54.0000000
		Height	9	70.6666667	2.6457513	66.0000000	75.0000000
		Weight	9	204.2222222	30.2893454	140.0000000	240.0000000

In addition to the variables that you specify, PROC MEANS adds the following variables to the output set.

FREQ
: contains the number of observations that a given output level represents.

STAT
: contains the names of the default statistics if you omit statistic keywords.

TYPE
: contains information about the class variables. By default _TYPE_ is a numeric variable. If you specify CHARTYPE in the PROC statement, then _TYPE_ is a character variable. When you use more than 32 class variables, _TYPE_ is automatically a character variable.

The following output is of Work.Diabetes_By_Gender from the PRINT procedure.

Output 15.8 *PROC PRINT Output of Work.Diabetes_By_Gender*

Diabetes Results By Gender

Sex	_TYPE_	_FREQ_	AvgAge	AvgHeight	AvgWeight	MinAge	MinHeight	MinWeight
	0	20	46.7000	66.9500	174.650	15	61	102
F	1	11	48.9091	63.9091	150.455	16	61	102
M	1	9	44.0000	70.6667	204.222	15	66	140

TIP You can use the NOPRINT option in the PROC MEANS statement to suppress the default report.

The FREQ Procedure

What Does the FREQ Procedure Do?

PROC FREQ is a procedure that is used give descriptive statistics about a SAS data set. The procedure creates one-way, two-way, and *n*-way frequency tables. It also describes data by reporting the distribution of variable values. The FREQ procedure creates crosstabulation tables to summarize data for two or more categorical values by displaying the number of observations for each combination of variable values.

TIP It is a best practice that you use the TABLES statement with PROC FREQ.

FREQ Procedure Syntax

The FREQ procedure can include many statements and options for controlling frequency output.

Syntax, FREQ procedure:

PROC FREQ <*options*>;

RUN;

The following table lists the options that are available in the PROC FREQ statement.

Table 15.4 *PROC FREQ Statement Options*

Option	Description
COMPRESS	Begins the display of the next one-way frequency table on the same page as the preceding one-way table if there is enough space to begin the table. By default, the next one-way table begins on the current page only if the entire table fits on that page. *Note:* The COMPRESS option is not valid with the PAGE option.
DATA=*SAS-data-set*	Names the *SAS-data-set* to be analyzed by PROC FREQ. If you omit the DATA= option, the procedure uses the most recently created SAS data set.
FORMCHAR(1,2,7)='*formchar-string*'	Defines the characters to be used for constructing the outlines and dividers for the cells of crosstabulation table displays. The *formchar-string* should be three characters long. The characters are used to draw the vertical separators (position 1), the horizontal separators (position 2), and the vertical-horizontal intersections (position 7). If you do not specify the FORMCHAR= option, PROC FREQ uses FORMCHAR(1,2,7)='|-+' by default. Position 1 Default: | The characters are used to draw vertical separators. Position 2 Default: — The characters are used to draw horizontal separators. Position 7 Default: + The characters are used to draw intersections of vertical and horizontal separators. Specifying all blanks for *formchar-string* produces crosstabulation tables with no outlines or dividers—for example, FORMCHAR(1,2,7)=' '. You can use any character in *formchar-string*, including hexadecimal characters. If you use hexadecimal characters, you must put an x after the closing quotation mark.
NLEVELS	Displays the "Number of Variable Levels" table, which provides the number of levels for each variable named in the TABLES statements.

Option	Description
NOPRINT	Suppresses the display of all output. You can use the NOPRINT option when you want to create only an output data set.
<ORDER=DATA | FORMATTED | FREQ | INTERNAL>=	Specifies the order of the variable levels in the frequency and crosstabulation tables, which you request in the TABLES statement. The ORDER= option can take the following values: DATA order of appearance in the input data set FORMATTED external formatted value, except for numeric variables with no explicit format, which are sorted by their unformatted (internal) value FREQ descending frequency count; levels with the most observations come first in the order INTERNAL unformatted value *Note:* The ORDER= option does not apply to missing values, which are always ordered first.
PAGE	Displays only one table per page. Otherwise, PROC FREQ displays multiple tables per page as space permits. *Note:* The PAGE option is not valid with the COMPRESS option.

Example: Creating a One-Way Frequency Table (Default)

By default, the FREQ procedure creates a one-way table that contains the frequency, percent, cumulative frequency, and cumulative percent of every value of every variable in the input data set. In the following example, the FREQ procedure creates crosstabulation tables for each of the variables.

```
proc freq data=cert.usa;
run;
```

Output 15.9 *PROC FREQ Output of Cert.Usa*

Dept	Frequency	Percent	Cumulative Frequency	Cumulative Percent
ADM10	5	33.33	5	33.33
ADM20	4	26.67	9	60.00
ADM30	2	13.33	11	73.33
CAM10	3	20.00	14	93.33
CAM20	1	6.67	15	100.00

WageCat	Frequency	Percent	Cumulative Frequency	Cumulative Percent
H	1	6.67	1	6.67
S	14	93.33	15	100.00

WageRate	Frequency	Percent	Cumulative Frequency	Cumulative Percent
13.65	1	6.67	1	6.67
1572.5	1	6.67	2	13.33
1813.3	1	6.67	3	20.00
2960	1	6.67	4	26.67
3392.5	1	6.67	5	33.33
3420	1	6.67	6	40.00
3819.2	1	6.67	7	46.67
4045.8	1	6.67	8	53.33
4480.5	1	6.67	9	60.00
4522.5	1	6.67	10	66.67
5260	1	6.67	11	73.33
5910.8	1	6.67	12	80.00
6855.9	1	6.67	13	86.67
6862.5	1	6.67	14	93.33
9073.8	1	6.67	15	100.00

Manager	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Coxe	5	33.33	5	33.33
Delgado	5	33.33	10	66.67
Overby	5	33.33	15	100.00

JobType	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	1	6.67	1	6.67
3	1	6.67	2	13.33
5	1	6.67	3	20.00
10	1	6.67	4	26.67
20	2	13.33	6	40.00
50	2	13.33	8	53.33
240	4	26.67	12	80.00
420	2	13.33	14	93.33
440	1	6.67	15	100.00

Specifying Variables Using the TABLES Statement

By default, the FREQ procedure creates frequency tables for every variable in a data set. But this is not always what you want. A variable that has continuous numeric values (such as DateTime) can result in a lengthy and meaningless table. Likewise, a variable that has a unique value for each observation (such as FullName) is unsuitable for PROC FREQ processing. Frequency distributions work best with variables whose values are categorical, and whose values are better summarized by counts rather than by averages.

To specify the variables to be processed by the FREQ procedure, include a TABLES statement.

Syntax, TABLES statement:

TABLES *variable(s)***;**

variable(s) lists the variables to include.

Example: Creating a One-Way Table for One Variable

The TABLES statement tells SAS the specific frequency tables that you want to create. The following example creates only one frequency table for the variable Sex as specified in the TABLES statement. The other variables are suppressed.

```
proc freq data=cert.diabetes;
  tables sex;
run;
```

Output 15.10 *One-Way Table for the Variable Sex*

Sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	11	55.00	11	55.00
M	9	45.00	20	100.00

Example: Determining the Report Layout

The order in which the variables appear in the TABLES statement determines the order in which they are listed in the PROC FREQ report.

Consider the SAS data set Cert.Loans. The variables Rate and Months are categorical variables, so they are the best choices for frequency tables.

```
proc freq data=cert.loans;
  tables rate months;
run;
```

Output 15.11 *Frequency Tables for Rate and Months*

Rate	Frequency	Percent	Cumulative Frequency	Cumulative Percent
9.50%	2	22.22	2	22.22
9.75%	1	11.11	3	33.33
10.00%	2	22.22	5	55.56
10.50%	4	44.44	9	100.00

Months	Frequency	Percent	Cumulative Frequency	Cumulative Percent
12	1	11.11	1	11.11
24	1	11.11	2	22.22
36	1	11.11	3	33.33
48	1	11.11	4	44.44
60	2	22.22	6	66.67
360	3	33.33	9	100.00

In addition to listing variables separately, you can use a numbered range of variables.

```
proc freq data=cert.survey;
  tables item1-item3;
run;
```

Output 15.12 *Frequency Tables for Item1–Item3*

Item1	Frequency	Percent	Cumulative Frequency	Cumulative Percent
2	1	25.00	1	25.00
4	2	50.00	3	75.00
5	1	25.00	4	100.00

Item2	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	1	25.00	1	25.00
3	2	50.00	3	75.00
5	1	25.00	4	100.00

Item3	Frequency	Percent	Cumulative Frequency	Cumulative Percent
4	3	75.00	3	75.00
5	1	25.00	4	100.00

TIP To suppress the display of cumulative frequencies and cumulative percentages in one-way frequency tables and in list output, add the NOCUM option to your TABLES statement. Here is the syntax:

TABLES *variable(s)* / **NOCUM**;

Create Two-Way and N-Way Tables

The simplest crosstabulation is a two-way table. To create a two-way table or *n*-way table, join the variables with an asterisk (*) in the TABLES statement in a PROC FREQ step. For a two-way table, one table is created. For *n*-way tables, a series of tables are produced with a table for each level of the variables.

Syntax, TABLES statement for crosstabulation:

TABLES *variable-1 *variable-2 <* ... variable-n>***;**

Here are the options for two-way tables:

- *variable-1* specifies table rows.
- *variable-2* specifies table columns.

Tip: You can include up to 50 variables in a single multi-way table request.

When crosstabulations are specified, PROC FREQ produces tables with cells that contain the following frequencies:

- cell frequency
- cell percentage of total frequency
- cell percentage of row frequency
- cell percentage of column frequency

Example: Creating Two-Way Tables

In the following example, you can create a two-way table to see the frequency of fasting glucose levels for each value for the variable Sex.

```
proc freq data=cert.diabetes;
  tables sex*fastgluc;
run;
```

Output 15.13 *Two-Way Table Output Cert.Diabetes*

Frequency Percent Row Pct Col Pct	Table of Sex by FastGluc											
	FastGluc											
Sex	152	155	156	166	177	193		447	486	492	568	Total
F	1 5.00 9.09 100.00	1 5.00 9.09 100.00	0 0.00 0.00 0.00	1 5.00 9.09 100.00	1 5.00 9.09 100.00	1 5.00 9.09 100.00	. . . more variables . . .	0 0.00 0.00 0.00	0 0.00 0.00 0.00	0 0.00 0.00 0.00	1 5.00 9.09 100.00	11 55.00
M	0 0.00 0.00 0.00	0 0.00 0.00 0.00	1 5.00 11.11 100.00	0 0.00 0.00 0.00	0 0.00 0.00 0.00	0 0.00 0.00 0.00		1 5.00 11.11 100.00	1 5.00 11.11 100.00	1 5.00 11.11 100.00	0 0.00 0.00 0.00	9 45.00
Total	1 5.00	1 5.00	1 5.00	1 5.00	1 5.00	1 5.00		1 5.00	1 5.00	1 5.00	1 5.00	20 100.00

Note that the first variable, Sex, forms the table rows, and the second variable, FastGluc, forms the columns. Reversing the order of the variables in the TABLES statement would reverse their positions in the table. Note also that the statistics are listed in the legend box.

Examples: Creating N-Way Tables

The following example creates a series of two-way tables with a table for each level of the other variables. The variables WhiteCells and AG are the rows and columns that are crosstabulated by the variable Survived.

```
proc format;
  value Survive 0='Dead'
                1='Alive';
run;
proc freq data=cert.leukemia;
  tables Survived*AG*WhiteCells;
  format Survived survive.;
run;
```

Output 15.14 *N-Way Tables*

Frequency
Percent
Row Pct
Col Pct

Table 1 of AG by WhiteCells

Controlling for Survived=Dead

AG	WhiteCells											
	750	1500	2300	2600	3000	. . . more variables . . .	31000	32000	35000	52000	79000	1000000	Total
Absent	0 0.00 0.00 .	1 5.56 8.33 100.00	0 0.00 0.00 .	0 0.00 0.00 .	0 0.00 0.00 .		1 5.56 8.33 100.00	0 0.00 0.00 .	0 0.00 0.00 0.00	0 0.00 0.00 0.00	0 0.00 0.00 .	1 5.56 8.33 33.33	12 66.67
Present	0 0.00 0.00 .	0 0.00 0.00 0.00	0 0.00 0.00 .	0 0.00 0.00 .	0 0.00 0.00 .		0 0.00 0.00 0.00	0 0.00 0.00 .	1 5.56 16.67 100.00	1 5.56 16.67 100.00	0 0.00 0.00 .	2 11.11 33.33 66.67	6 33.33
Total	0 0.00	1 5.56	0 0.00	0 0.00	0 0.00		1 5.56	0 0.00	1 5.56	1 5.56	0 0.00	3 16.67	18 100.00

Frequency
Percent
Row Pct
Col Pct

Table 2 of AG by WhiteCells

Controlling for Survived=Alive

AG	WhiteCells											
	750	1500	2300	2600	3000	. . . more variables . . .	31000	32000	35000	52000	79000	1000000	Total
Absent	0 0.00 0.00 0.00	0 0.00 0.00 .	0 0.00 0.00 0.00	0 0.00 0.00 0.00	1 6.67 25.00 100.00		0 0.00 0.00 .	0 0.00 0.00 0.00	0 0.00 0.00 .	0 0.00 0.00 .	1 6.67 25.00 100.00	1 6.67 25.00 50.00	4 26.67
Present	1 6.67 9.09 100.00	0 0.00 0.00 .	1 6.67 9.09 100.00	1 6.67 9.09 100.00	0 0.00 0.00 0.00		0 0.00 0.00 .	1 6.67 9.09 100.00	0 0.00 0.00 .	0 0.00 0.00 .	0 0.00 0.00 0.00	1 6.67 9.09 50.00	11 73.33
Total	1 6.67	0 0.00	1 6.67	1 6.67	1 6.67		0 0.00	1 6.67	0 0.00	0 0.00	1 6.67	2 13.33	15 100.00

Creating Tables Using the LIST Option

When three or more variables are specified, the multiple levels of *n*-way tables can produce considerable output. Such bulky, often complex crosstabulations are often easier to read when they are arranged as a continuous list. Although this arrangement eliminates row and column frequencies and percentages, the results are compact and clear.

TIP The LIST option is not available when you also specify statistical options.

To generate list output for crosstabulations, add a slash (/) and the LIST option to the TABLES statement in your PROC FREQ step.

Syntax, TABLES statement:

TABLES *variable-1 *variable-2 <* ... variable-n>* **/ LIST;**

Here are the options for two-way tables:

- *variable-1* specifies table rows.
- *variable-2* specifies table columns.

Tip: You can include up to 50 variables in a single multi-way table request.

Example: Using the LIST Option

As in the previous example, the following example creates a series of two-way tables with a table for each level of the other variables. The variables WhiteCells and AG are the rows and columns that are crosstabulated by the variable Survived. Use the LIST option in the TABLES statement to make the PROC FREQ output easier to read. The output is generated in a continuous list.

```
proc format;
  value survive 0='Dead'
                1='Alive';
run;
proc freq data=cert.leukemia;
  tables Survived*AG*WhiteCells / list;
  format Survived survive.;
run;
```

Output 15.15 *PROC FREQ Output in List Format*

Survived	AG	WhiteCells	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Dead	Absent	1500	1	3.03	1	3.03
Dead	Absent	4000	1	3.03	2	6.06
Dead	Absent	5300	1	3.03	3	9.09
Dead	Absent	9000	1	3.03	4	12.12
Dead	Absent	10000	1	3.03	5	15.15
. . . more observations. . .						
Alive	Present	9400	1	3.03	29	87.88
Alive	Present	10000	1	3.03	30	90.91
Alive	Present	10500	1	3.03	31	93.94
Alive	Present	32000	1	3.03	32	96.97
Alive	Present	1000000	1	3.03	33	100.00

Example: Using the CROSSLIST Option

The CROSSLIST option displays crosstabulation tables in ODS column format instead of the default crosstabulation cell format. In a CROSSLIST table display, the rows correspond to the crosstabulation table cells, and the columns correspond to descriptive statistics such as Frequency and Percent. The CROSSLIST table displays the same information as the default crosstabulation table, but uses an ODS column format instead of the table cell format

```
proc format;
  value survive 0='Dead'
                1='Alive';
run;
proc freq data=cert.leukemia;
  tables Survived*AG*whitecells / crosslist;
  format Survived survive.;
run;
```

Output 15.16 *Table Created by the CROSSLIST Option Survived=Dead*

Table of AG by WhiteCells					
Controlling for Survived=Dead					
AG	WhiteCells	Frequency	Percent	Row Percent	Column Percent
Absent	750	0	0.00	0.00	.
	1500	1	5.56	8.33	100.00
	2300	0	0.00	0.00	.
	2600	0	0.00	0.00	.
	3000	0	0.00	0.00	.
. . . *more observations*. . .					
	Total	12	66.67	100.00	
Present	750	0	0.00	0.00	.
	1500	0	0.00	0.00	0.00
	2300	0	0.00	0.00	.
	2600	0	0.00	0.00	.
	3000	0	0.00	0.00	.
. . . *more observations*. . .					
	Total	6	33.33	100.00	
Total	750	0	0.00		.
	1500	1	5.56		100.00
	2300	0	0.00		.
	2600	0	0.00		.
	3000	0	0.00		.
. . . *more observations*. . .					
	35000	1	5.56		100.00
	52000	1	5.56		100.00
	79000	0	0.00		.
	1000000	3	16.67		100.00
	Total	18	100.00		

Output 15.17 *Table Created by the CROSSLIST Option Survived=Alive*

Table of AG by WhiteCells					
Controlling for Survived=Alive					
AG	WhiteCells	Frequency	Percent	Row Percent	Column Percent
Absent	750	0	0.00	0.00	0.00
	1500	0	0.00	0.00	.
	2300	0	0.00	0.00	0.00
	2600	0	0.00	0.00	0.00
	3000	1	6.67	25.00	100.00
. . . more observations. . .					
	Total	4	26.67	100.00	
Present	750	1	6.67	9.09	100.00
	1500	0	0.00	0.00	.
	2300	1	6.67	9.09	100.00
	2600	1	6.67	9.09	100.00
	3000	0	0.00	0.00	0.00
. . . more observations. . .					
	Total	11	73.33	100.00	
Total	750	1	6.67		100.00
	1500	0	0.00		.
	2300	1	6.67		100.00
	2600	1	6.67		100.00
	3000	1	6.67		100.00
. . . more observations. . .					
	35000	0	0.00		.
	52000	0	0.00		.
	79000	1	6.67		100.00
	1000000	2	13.33		100.00
	Total	15	100.00		

Suppressing Table Information

Another way to control the format of crosstabulations is to limit the output of the FREQ procedure to a few specific statistics. Remember that when crosstabulations are run, PROC FREQ produces tables with cells that contain these frequencies:

- cell frequency
- cell percentage of total frequency
- cell percentage of row frequency
- cell percentage of column frequency

You can use options to suppress any of these statistics. To control the depth of crosstabulation results, add any combination of the following options to the TABLES statement:

- NOFREQ suppresses cell frequencies
- NOPERCENT suppresses cell percentages
- NOROW suppresses row percentages
- NOCOL suppresses column percentages

Example: Suppressing Percentages

You can suppress frequency counts, rows, and column percentages by using the NOFREQ, NOROW, and NOCOL options in the TABLES statement.

```
proc format;
  value survive  0='Dead'
                 1='Alive';
run;
proc freq data=cert.leukemia;
  tables Survived*AG*whitecells / nofreq norow nocol;
  format Survived survive.;
run;
```

Output 15.18 *Suppressing Percentage Information*

Percent

Table 1 of AG by WhiteCells

Controlling for Survived=Dead

AG	WhiteCells												
	750	1500	2300	2600	3000	. . . more variables . . .	31000	32000	35000	52000	79000	1000000	Total
Absent	0.00	5.56	0.00	0.00	0.00		5.56	0.00	0.00	0.00	0.00	5.56	66.67
Present	0.00	0.00	0.00	0.00	0.00		0.00	0.00	5.56	5.56	0.00	11.11	33.33
Total	0 0.00	1 5.56	0 0.00	0 0.00	0 0.00		1 5.56	0 0.00	1 5.56	1 5.56	0 0.00	3 16.67	18 100.00

Percent

Table 2 of AG by WhiteCells

Controlling for Survived=Alive

AG	WhiteCells												
	750	1500	2300	2600	3000	. . . more variables . . .	31000	32000	35000	52000	79000	1000000	Total
Absent	0.00	0.00	0.00	0.00	6.67		0.00	0.00	0.00	0.00	6.67	6.67	26.67
Present	6.67	0.00	6.67	6.67	0.00		0.00	6.67	0.00	0.00	0.00	6.67	73.33
Total	1 6.67	0 0.00	1 6.67	1 6.67	1 6.67		0 0.00	1 6.67	0 0.00	0 0.00	1 6.67	2 13.33	15 100.00