Specifying Titles and Footnotes in Procedure Output

TITLE and FOOTNOTE Statements

To make your report more meaningful and self-explanatory, you can assign up to 10 titles with procedure output by using TITLE statements before the PROC step. Likewise, you can specify up to 10 footnotes by using FOOTNOTE statements before the PROC step.

TIP Because TITLE and FOOTNOTE statements are global statements, place them anywhere within or before the PRINT procedure. Titles and footnotes are assigned as soon as TITLE or FOOTNOTE statements are read; they apply to all subsequent output.

Syntax, TITLE, and FOOTNOTE statements:

TITLE<*n*> *'text'***;**

FOOTNOTE<*n*> *'text'***;**

n is a number from 1 to 10 that specifies the title or footnote line, and *'text'* is the actual title or footnote to be displayed. The maximum title or footnote length depends on your operating environment and on the value of the LINESIZE= option.

The keyword TITLE is equivalent to TITLE1. Likewise, FOOTNOTE is equivalent to FOOTNOTE1. If you do not specify a title, the default title is The SAS System. No footnote is printed unless you specify one.

As a best practice be sure to match quotation marks that enclose the title or footnote text.

Example: Creating Titles

In the following example, the two TITLE statements are specified for lines 1 and 3. These two TITLE statements define titles for the PROC PRINT output. You can create a blank line between two titles by skipping a number in the TITLE statement.

```
title1 'Heart Rates for Patients with';
title3 'Increased Stress Tolerance Levels';
proc print data=cert.stress;
  var resthr maxhr rechr;
  where tolerance='I';
run;
```

Output 6.4 *PROC PRINT Output with Titles*

Heart Rates for Patients with

Increased Stress Tolerance Levels

Obs	RestHR	MaxHR	RecHR
2	68	171	133
3	78	177	139
8	70	167	122
11	65	181	141
14	74	152	113
15	75	158	108
20	78	189	138

Example: Creating Footnotes

In the following example, the two FOOTNOTE statements are specified for lines 1 and 3. These two FOOTNOTE statements define footnotes for the PROC PRINT output. Since there is no FOOTNOTE2, a blank line is inserted between FOOTNOTE1 and FOOTNOTE3 in the output.

```
footnote1 'Data from Treadmill Tests';
footnote3 '1st Quarter Admissions';
proc print data=cert.stress;
  var resthr maxhr rechr;
  where tolerance='I';
run;
```

Footnotes appear at the bottom of each page of procedure output. Notice that footnote lines are pushed up from the bottom. The FOOTNOTE statement that has the largest number appears on the bottom line.

Output 6.5 *PROC PRINT Output with Footnotes*

Heart Rates for Patients with

Increased Stress Tolerance Levels

Obs	RestHR	MaxHR	RecHR
2	68	171	133
3	78	177	139
8	70	167	122
11	65	181	141
14	74	152	113
15	75	158	108
20	78	189	138

Data from Treadmill Tests

1st Quarter Admissions

Modifying and Canceling Titles and Footnotes

As global statements, the TITLE and FOOTNOTE statements remain in effect until you modify the statements, cancel the statements, or end your SAS session. In the following example, the titles and footnotes that are assigned in the PROC PRINT step also appear in the output for the PROC MEANS step.

```
title1 'Heart Rates for Patients with';
title3 'Increased Stress Tolerance Levels';
footnote1 'Data from Treadmill Tests';
footnote3 '1st Quarter Admissions';
proc print data=cert.stress;
  var resthr maxhr rechr;
  where tolerance='I';
run;
proc means data=cert.stress;
  where tolerance='I';
  var resthr maxhr;
run;
```

Output 6.6 *PROC PRINT Output with Titles and Footnotes*

Heart Rates for Patients with

Increased Stress Tolerance Levels

Obs	RestHR	MaxHR	RecHR
2	68	171	133
3	78	177	139
8	70	167	122
11	65	181	141
14	74	152	113
15	75	158	108
20	78	189	138

Data from Treadmill Tests

1st Quarter Admissions

Output 6.7 *PROC MEANS Output with Titles and Footnotes*

Heart Rates for Patients with

Increased Stress Tolerance Levels

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
RestHR	7	72.5714286	5.0284903	65.0000000	78.0000000
MaxHR	7	170.7142857	12.9449383	152.0000000	189.0000000

Data from Treadmill Tests

1st Quarter Admissions

Redefining a title or footnote line cancels any higher numbered title or footnote lines, respectively. In the example below, defining a title for line 2 in the second report automatically cancels title line 3.

```
title1 'Heart Rates for Patients with';
title3 'Participation in Exercise Therapy';
footnote1 'Data from Treadmill Tests';
footnote3 '1st Quarter Admissions';
proc print data=cert.therapy;
  var swim walkjogrun aerclass;
run;
title2 'Report for March';
proc print data=cert.therapy;
run;
```

Output 6.8 *PROC PRINT Output of Cert.Therapy with Title 1 and Title 3 (partial output)*

Heart Rates for Patients with

Participation in Exercise Therapy

Obs	Swim	WalkJogRun	AerClass
1	14	78	56
2	19	109	32
...more observations...			
20	53	65	63
21	68	49	60
22	41	70	78
23	58	44	82
24	47	57	93

Data from Treadmill Tests

1st Quarter Admissions

Output 6.9 *PROC PRINT Output of Cert.Therapy with Title 1 and Title 2 (partial output)*

Heart Rates for Patients with
Report for March

Obs	Date	AerClass	WalkJogRun	Swim
1	JAN2012	56	78	14
2	FEB2012	32	109	19
	. . . more observations. . .			
20	AUG2013	63	65	53
21	SEP2013	60	49	68
22	OCT2013	78	70	41
23	NOV2013	82	44	58
24	DEC2013	93	57	47

Data from Treadmill Tests

1st Quarter Admissions

To cancel all previous titles or footnotes, specify a null TITLE or FOOTNOTE statement. A null TITLE or FOOTNOTE statement does not contain any number or text and cancels all footnotes and titles that are in effect.

```
title1;                                  /* #1 */
footnote1 'Data from Treadmill Tests';   /* #2 */
footnote3 '1st Quarter Admissions';
proc print data=cert.stress;
  var resthr maxhr rechr;
  where tolerance='I';
run;
footnote;                                /* #3 */
proc means data=cert.stress;
  where tolerance='I';
  var resthr maxhr;
run;
```

1 Specifying the TITLE1 statement cancels all previous titles and cancels the default title **`The SAS System`**. The PRINT procedure and the MEANS procedure do not contain any titles in the output.

2 Specifying the FOOTNOTE1 and FOOTNOTE3 statements before the PRINT procedure results in footnotes in the PROC PRINT output.

3 Specifying a null FOOTNOTE statement cancels the previously defined footnotes that are in effect.

Output 6.10 *PROC PRINT Output with Footnotes and No Titles*

Obs	RestHR	MaxHR	RecHR
2	68	171	133
3	78	177	139
8	70	167	122
11	65	181	141
14	74	152	113
15	75	158	108
20	78	189	138

Data from Treadmill Tests

1st Quarter Admissions

Output 6.11 *PROC MEANS Output with No Footnotes and No Titles*

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
RestHR	7	72.5714286	5.0284903	65.0000000	78.0000000
MaxHR	7	170.7142857	12.9449383	152.0000000	189.0000000

Assigning Descriptive Labels

Temporarily Assigning Labels to Variables

To enhance your PROC PRINT by labeling columns:

- Use the LABEL statement to assign a descriptive label to a variable.
- Use the LABEL option in the PROC PRINT statement to specify that the labels be displayed.

Syntax, LABEL statement:

LABEL *variable1='label1'*
variable2='label2'
... ;

Labels can be up to 256 characters long. Enclose the label in quotation marks.

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

Example: Using the LABEL Option in the PROC PRINT Statement

In the PROC PRINT step below, the variable name WalkJogRun is displayed with the label Walk/Jog/Run. Note that the LABEL option is in the PROC PRINT statement.

```
proc print data=cert.therapy label;
  label walkjogrun='Walk/Jog/Run';
run;
```

Output 6.12 *PROC PRINT Output with LABEL Option (partial output)*

Obs	Date	AerClass	Walk/Jog/Run	Swim
1	JAN2012	56	78	14
2	FEB2012	32	109	19
	. . . more observations. . .			
20	AUG2013	63	65	53
21	SEP2013	60	49	68
22	OCT2013	78	70	41
23	NOV2013	82	44	58
24	DEC2013	93	57	47

If you omit the LABEL option in the PROC PRINT statement, PROC PRINT uses the name of the column heading, **`walkjogrun`**, even though you specified a value for the variable.

Example: Using Multiple LABEL Statements

The following example illustrates the use of multiple LABEL statements.

```
proc print data=cert.admit label;       /*#1*/
  var age height;
  label age='Age of Patient';           /*#2*/
  label height='Height in Inches';      /*#3*/
run;
```

1 Use the LABEL option with the PROC PRINT statement. If you omit the LABEL option in the PROC PRINT statement, PROC PRINT uses the variable name.

2 You can assign labels in separate LABEL statements. In this example, label the variable Age as Age of Patients.

3 This is the second LABEL statement in this example. Label the variable Height as Height in Inches.

Output 6.13 *PROC PRINT Output with Multiple LABEL Statements (partial output)*

Obs	Age of Patient	Height in Inches
1	27	72
2	34	66
...more observations...		
17	43	65
18	25	75
19	22	63
20	41	67
21	54	71

Example: Using a Single LABEL Statement to Assign Multiple Labels

You can also assign multiple labels using a single LABEL statement.

```
proc print data=cert.admit label;          /*#1*/
  var actlevel height weight;
  label actlevel='Activity Level'          /*#2*/
        height='Height in Inches'
        weight='Weight in Pounds';
run;
```

1 Use the LABEL option with the PROC PRINT statement.

2 A single LABEL statement assigns three labels to three different variables. Note that you do not need a semicolon at the end of your label until you are ready to close your LABEL statement. In this example, the semicolon is at the end of the label for Weight.

Output 6.14 *PROC PRINT Output with a Single LABEL Statement (partial output)*

Obs	Activity Level	Height in Inches	Weight in Pounds
1	HIGH	72	168
2	HIGH	66	152
. . . more observations. . .			
17	MOD	65	123
18	HIGH	75	188
19	LOW	63	139
20	HIGH	67	141
21	MOD	71	183

Using Permanently Assigned Labels

When you use a LABEL statement within a PROC step, the label applies only to the output from that step.

However, in PROC steps, you can also use permanently assigned labels. Permanent labels can be assigned in the DATA step. These labels are saved with the data set, and they can be reused by procedures that reference the data set.

For example, the DATA step below creates the data set Cert.Paris and defines the label for the variable Date. Because the LABEL statement is inside the DATA step, the labels are written to the Cert.Paris data set and are available to the subsequent PRINT procedure.

```
data cert.paris;
  set cert.laguardia;
  where dest='PAR' and (boarded=155 or boarded=146);
  label date='Departure Date';
run;
proc print data=cert.paris label;
  var date dest boarded;
run;
```

Output 6.15 *Using Permanent Labels*

Obs	Departure Date	Dest	Boarded
1	04MAR2012	PAR	146
2	07MAR2012	PAR	155
3	04MAR2012	PAR	146
4	07MAR2012	PAR	155

Notice that the PROC PRINT statement still requires the LABEL option in order to display the permanent labels. Other SAS procedures display permanently assigned labels without additional statements or options.

For more information about permanently assigning labels, see Lesson 9, “Creating and Managing Variables,” on page 137.

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