Paper 64-26

The Essential Meaning of PROC MEANS: A Beginner's Guide to Summarizing Data Using SAS® Software

Andrew H. Karp
Sierra Information Services, Inc.
Sonoma, California USA

Introduction

Learning how to use PROC MEANS is often a difficult task for new users of the SAS System. It is an extremely powerful procedure with numerous options, statements and capabilities. In Version 8 the functionalities of this procedure, long a staple of SAS programs and applications, were significantly enhanced. This Beginning Tutorial will guide you through the basics of how to apply this BASE SAS procedure to summarize and analyze the values of numeric variables in your data sets. By the end of the paper you should have a grasp of what PROC MEANS can do for you, how to avoid common pitfalls in using it, and some of the most important enhancements to it in Version 8 of the SAS System. Once you have mastered the concepts presented in the paper you will be able to use the procedure with more confidence and to learn about additional functionalities in its documentation.

Core Concepts

PROC MEANS is found in BASE SAS software, so every SAS site has it. Since it is a procedure, it operates on the variables in a SAS data set, or in a SAS view to another relational data base management system product using SAS/ACCESS™ software. In this paper, we will work with existing SAS data sets to demonstrate various capabilities of PROC MEANS. PROC MEANS, like most other SAS Procedures, therefore "works down the columns" or the variables data set. The SAS Programming Language, also in BASE SAS Software, is used to "operate on the rows" or observations.

The core function of PROC MEANS is to analyze the values of variables that are defined as numeric variables. Its analyses can be portrayed in the SAS Output Window (the default), or, with some additional statements within the PROC MEANS 'unit of work,' stored in SAS data sets. As we will soon see, PROC MEANS has a powerful range of tools to analyze numeric variables and then store those analyses in new SAS data sets.

PROC MEANS vs. PROC SUMMARY

Prior to Version 6 of the SAS System, released in 1990, these two procedures had many different features and capabilities. In Version 6 they were standardized and with some very small differences, essentially became identical procedures. With Version 8 of the SAS System they remain essentially identical procedures. The key difference between PROC MEANS and PROC SUMMARY is that the default action of PROC MEANS is to place the analyses it performs in to your Output Window and in PROC SUMMARY the default is to create an output data set. Chapter 36 ("The Summary Procedure") of the Version 8 BASE SAS documentation contains additional details.

PROC MEANS vs. PROC UNIVARIATE

PROC UNIVARIATE is another BASE SAS procedure that is often used to analyze numeric values in a SAS data set. In Version 6 and earlier releases of SAS Software there were some key differences in the types of analyses that each procedure could perform. The median (50th percentile) could only be calculated in PROC UNIVARIATE, for example. With Version 8 of the SAS System PROCs MEANS, SUMMARY, and UNIVARIATE all calculate the same analytical statistics. Statisticians and others interested in looking at the distributional properties of a numeric variable, or who are interested in graphical displays of a numeric variable's distribution should consider using the BOXPLOT and HISTOGRAM options in PROC UNIVARIATE, as these are not available in PROC MEANS.

PROC MEANS vs. PROC FREQ

PROC FREQ is another powerful BASE SAS procedure that can analyze both numeric and character variables in SAS data sets or SAS views to other RDBMS files. As its name implies, the core function of PROC FREQ is to generate frequency tables that portray counts of how many observations have particular values of the variable or variables listed in its TABLES statement. PROC FREQ also calculates statistics, such as the Pearson Chi-Square Test of Independence and Kendall's tau measures, to assess the association between the values of two or more variables. Although PROC FREQ can create output SAS data sets, its functionalities are vastly different from those available from PROC MEANS.

Getting Started with PROC MEANS

When using PROC MEANS, remember that there are two types of variables you will use with it. Analysis Variables are numeric variables that you tell the procedure you want analyzed. Classification (or "by-group") variables are either numeric or character variables. When you specify one or more classification variables in your PROC MEANS task, the procedure conducts the desired analyses for each separate value of the variable(s) you tell it are the classification variables.

The analysis variables go in the VAR statement. They MUST be stored as numeric variables in your SAS data set. If you forget and put one or more character variables

in the VAR statement then the PROC MEANS task will not run for ANY of the variables and you will see an error message in your SAS log.

By default, PROC MEANS will analyze ALL the numeric variables in your SAS data set unless you give it specific instructions, via the VAR statement, as to which variable to analyze. As a general rule, it is a good idea to get in the habit of using the VAR statement to explicitly list all the numeric variables for which you want analyses performed. In many data sets there are numeric variables whose values cannot be analyzed statistically and return a meaningful result. The mean of customer telephone number, or the sum of zip code does not "tell you" anything worthwhile, so there is no reason to waste computing and other resources having PROC MEANS calculate analyses of them. A fancier way of putting this that in many data sets there are numeric variables that do not "admit of a meaningful arithmetic operation," which means that the result of applying an arithmetic operation (e.g., summation) does not give you a result you can use.

Here is an example data set we will use to show some of the core features of PROC MEANS. Later we will use a larger data set to show some more advanced features of the procedure. This data set, called ELECTRICITY, contains 12 observations showing electric consumption and revenue billed by a public utility in January 2001. The unit of measurement for electricity consumption is the Kilowatt Hour, abbreviated KwH.

The variables REV1 and KWH1 show the amount of revenue billed to each customer and their KwH consumption for January. Most of the other variables are self-explanatory with the possible exception of SCHEDULE, which indicates the rate schedule under which that customer was charged and SERIAL, which identifies the day of the week on which the customer meter is to be read.

Here is a very simple use of PROC MEANS with these data:

```
PROC MEANS Data=SUGI26.Electricity;
VAR KWH1 REV1;
Title3 'Default PROC MEANS Results';
Run;
```

SAS places the output shown in Figure 1 in the Output Window

This example shows what you will obtain, by default, from PROC MEANS. Five statistical measures are computed for each variable in the VAR statement (or for ALL numeric variables in the SAS data set, if the VAR statement is omitted. Looking back at Figure 1, it's easy to understand why we did not include DIVISION as an analysis variable. There is no meaningful arithmetic operation to be applied to that variable. Later, we will use PROC FORMAT with DIVISION and use it as a classification variable.

Now let's get a bit fancier. In the next example, we will use two options in the PROC MEANS Statement. The WHERE clause data set option will exclude from the analysis all observations where the value of the variable SERIAL is equal to X and the MAXDEC option will round the SAS-generated output to two decimal places. Here we go:

```
PROC MEANS
DATA=SUGI26.Electricity(where=(SERIAL NOT
IN('X'))) MAXDEC=2;
VAR KWH1 REV1;
Title3 'PROC MEANS with a WHERE Clause
Data Set Option and the MAXDEC=2 Option';
Run;
```

If we compare the Figure 2 to Figure 3, you will see that there are only ten (10) observations analyzed in the output shown in Figure 3 and twelve (12) in Figure 2. That's because the WHERE Clause Data Set option using to generate the output in Figure 3 instructed the SAS System to exclude the two observations with values of the variable SERIAL equal to the letter X. Those observations are still in the SAS data set, but they were not used by PROC MEANS to create the output in Figure 3.

Using a CLASSIFICATION Variable

The previous two examples show how PROC MEANS was used to analyze the values of two variables without requesting sub-group or separate analyses at each unique value of another variable. For the purposes of this tutorial, we will call the variables whose values will be used to sub-group and analyze the analysis variables the *classification* variables.

Classification variables are placed in either the BY Statement or CLASS Statement. If you use the BY Statement, then the observations in the input data set (that is, the data set upon which you are applying PROC MEANS) ***must be sorted*** by the values of the classification variables. Otherwise, PROC MEANS will not execute and you will get an error message in your SASLOG. You can overcome this problem by using the CLASS statement, in which case the SAS System will automatically assemble the observations by ascending value of the classification variables.

When using large data sets it is usually more efficient to sort the data set by the desired classification variables and then use PROC MEANS with the BY Statement. Depending on the number of observations, number of classification variables and the number of unique combinations of the values of the classification variables, and your operating system, PROC MEANS may run out of memory before it can assemble all the unique values of the CLASS statement variables.

It is impossible, because of all the different factors involved, to provide a "magic number" of observations, classification variables, or combinations of classification variables where using PROC SORT and the BY statement is more efficient than using the CLASS statement. A few years ago I tested this with a data set containing about 2,755,000 observations on a mainframe computer. I was able to save about 25% of the central processing unit (CPU) time required to use PROC MEANS to create an output data set (see below) if I used PROC SORT and then a BY statement in PROC MEANS instead of using the CLASS statement. In the mainframe world, this is a significant savings. With our 12 observation test data set used to create examples for this tutorial, we don't need to worry about this issue. But, you should keep it in mind when working with your large data sets.

Now it is time to see how the CLASS statement works in PROC MEANS. The next PROC MEANS task requests an analysis of KwH classified by SCHEDULE. The MAXDEC option will be set to zero (0) as was demonstrated in the previous example. Here is the SAS code to implement this example.

```
proc means maxdec=0
data=sugi26.electricity;
class schedule;
var kwh1;
title3 'Using the CLASS Statement for
By-Group Analyses';
run;
```

The resulting output is shown in Figure 4. You'll notice that that there are two lines in the output, one for each unique value of the classification variable. Also, SAS has generated a column labeled "N" and another marked "N Obs." What's the difference? The "N Obs" column shows the total number of observations with a non missing value of the classification variable and the "N" column shows the number of observations with a non missing value of the *Analysis Variable* at that unique value of the classification variable. These columns allow you to determine if some observations have missing values of the analysis variables.

Using Two or More Classification Variables

You can create more complex analyses of your data by declaring two or more variables as classification variables. One of the most powerful and useful capabilities of PROC MEANS is its ability to rapidly calculate analyses at different combinations of the values of the classification variables. This will be discussed in detail shortly, when we learn how to create output SAS data sets with PROC MEANS. To fix ideas, the final example of using PROC MEANS to generate results in the Output Window shows the results of an analysis of

```
proc means maxdec=0 SUM
data=sugi26.electricity;
class region schedule;
var kwh1;
title3 'Using the CLASS Statement with Two
CLASS Variables';
title4 'Using the SUM Statistics Keyword';
run;
```

KWH1 when both SCHEDULE and REGION are placed in the CLASS Statement.

In addition, this example shows how a *Statistics Keyword* is added to the PROC MEANS task in order to limit the generated analysis to include only a specified statistic. In this task only the SUM of KWH1 is requested. The resulting output is shown in Figure 5.

Creating Output SAS Data Sets With PROC MEANS

Up to now we've looked at ways to have PROC MEANS analyze numeric variables and put the results in the Output Window. While this is certainly very useful, PROC MEANS also contains a number of tools that are used to create output SAS data sets. The remainder of this tutorial shows you ways you can use these tools, and points out key enhancements to these capabilities now available in Version 8 of the SAS System.

Basic Rules of the Road for Creating Output SAS Data Sets with PROC MEANS

Here are a few rules you need to follow when using PROC MEANS to create an output SAS data set

- The OUTPUT Statement is used to generate the desired output SAS data set.
- Within the OUTPUT Statement, you need to give PROC MEANS explicit instructions as to which statistical analyses you want performed
- You can request different analyses for different analysis variables placed in the VAR statement
- It is a very good idea to either declare the names of the variables PROC MEANS will place in the output data set, or use the new AUTONAME option in the OUTPUT Statement, or you may be very sorry.
- Multiple data sets can be created in one PROC MEANS task by using multiple OUTPUT Statements.

With these rules in mind, let's create an output SAS data set containing the sum and mean of both KWH1 and REV1 classified by REGION. The new data set, called NEW1, is Figure 6 and was displayed using PROC PRINT. The NOPRINT option is added to the PROC MEANS statement, telling it that not to place any analyses in the Output Window (remember, we want to create a SAS data set and are not interested in having anything placed in the Output Window.

Take a close look at the OUTPUT Statement, which follows the VAR Statement. This statement gives PROC MEANS all the information it needs to create an output data set. To the right of "OUT=" is the name of the data

set to be created (in this example, temporary data set NEW1) which will be placed in the WORK data library.

```
proc means NOPRINT
data=sugi26.electricity;
class region ;
var kwh1 rev1;
output out=new1 sum=sumkwh1 sumrev1
mean=meankwh1 meanrev1;
run;

proc print data=new1;
title3 'SAS Data Set NEW1 Created by PROC
MEANS';
run;
```

Two Statistics Keywords are used in the Output Statement, SUM and MEAN. To the right of their respective equals signs are the names of the variables that PROC MEANS creates and places in temporary data set NEW1. So, the MEAN of KWH1 is given variable name MEANKWH1 and the SUM of REV1 is given variable name SUMREV1, and so forth.

You'll notice that the PROC PRINT output in Figure 6 shows that two other variables were created by PROC MEANS and placed in Data Set NEW1. These variables _TYPE_ and _FREQ_ are automatically created by PROC MEANS every time it creates and output data set. Some new users of PROC MEANS either ignore them or can't quite figure out what to do with them. The next several examples show you what they are and their usefulness when generating analyses of numeric variables in your SAS data sets.

Understanding _TYPE_ and _FREQ_

Understanding what these two variables represent, and then how to use their values, is essential to your becoming a power user of PROC MEANS. _TYPE_ shows the combination of classification variables used to create each observation in the output SAS data set, and _FREQ_ gives the number of observations in the source, or input, SAS data set used by PROC MEANS to create that observation in the output SAS data set.

Since there was only one variable placed in the CLASS statement, there are two unique values of the variable _TYPE_ in output data set NEW1 shown in Figure 6. _TYPE_ = 0 is shows the values of the analyses *without regard to the values of the classification variables.* In other words, the first observation in data set NEW1, with _TYPE_ = 0, shows the grand totals (sums) and the grand means of the two analysis variables without regard to the values of the classification variable REGION. The four observations with _TYPE_ = 1 are the means and sums of KWH1 and REV1 at each unique value of the variable REGION. Values of _FREQ_ show how many observations in the input SAS data set were used by PROC MEANS to generate each observation in the output data set. Since there are 12 observations in the input data set, the _FREQ_ for the observation with _TYPE_ = 0, is 12.

The next example shows gives a bit more detail about using _TYPE_ and _FREQ_ when there are two classification variables. In the this and subsequent PROC MEANS tasks you'll see why _TYPE_ and _FREQ_ two variables are very important, and how you can take advantage of them when you apply PROC MEANS to summarize your data sets.

The following PROC MEANS task generates temporary data set NEW4, which analyzes KWH1 and REV1 at all possible combinations of the values of two classification variables, DIVISION and SERIAL. As before, _TYPE_ = 0 shows the requested statistics without regard to the values of the classification variables. _TYPE_ = 1 contains analyses for SERIAL without regard to DIVISION, _TYPE_= 2 for DIVISION without regard to SERIAL and _TYPE_ = 3 is the analysis at every possible combination of DIVISION *and* SERIAL.

```
proc means noprint data=sugi26.electricity;
class division serial;
var kwh1 rev1;
output out=new4 mean(kwh1) =
                sum(rev1) =/autoname;
run;

proc print data=new4;
title1 'SUGI 26 Beginning Tutorials';
title2 'Electricity Data Set';
title3 'Understanding _TYPE_ and _FREQ_';
run;
```

Let's take a closer look at the OUTPUT Statement that created data set NEW4. There are two new features of PROC MEANS shown in it. First, we have requested different statistics for the two different analysis variables. We asked for the mean of KWH1 (that is, the average), and the sum (or total) of REV1. PROC MEANS allows you to select different statistical analyses of the analysis variables by first writing the statistics keyword and then putting the specific analysis variables in parentheses adjacent to the keyword. The other new feature shown is the AUTONAME option, which was added in Version 8 of the SAS System. In previous examples we've either declared the names of variables in the output statement or allowed the names of the analysis variables in the source, or input, data set to be the names of the variables in the output data set. With the AUTONAME option SAS automatically appends, or places, the statistics keyword after the name of the analysis variable and an underscore. This is a very useful feature that you should consider using when requesting multiple analyses of several analysis variables. If you are not careful, or forget to use the AUTONAME statement, PROC MEANS will give you incomplete results.

The output from this example was printed using PROC PRINT and is shown as Figure 7 below.
With only two classification variables we obtain four values of the variable _TYPE_. How many would there

be if we had ten variables in the CLASS or BY statement? Unless you use the NWAY option in the PROC MEANS Statement, there will be 1,024 unique values of this variable in the output data set! Computing this value is easy: there will be 2^n values of _TYPE_, where n is the number of variables in the CLASS statement.

With this knowledge in hand, we can now demonstrate another very powerful feature available in PROC MEANS. We can create different output SAS data sets, in a single use of PROC MEANS, containing both different levels of analyses and different statistical measures of the analysis variables.

Let's see how this is done by using the twelve observation electrical consumption data set.

```
proc means data=sugi26.electricity
noprint;
class office schedule serial;
var kwh1;
output out=new6 mean=  sum=/autoname;
run;

proc print data = new6;
title3 'PROC MEANS Output with three
CLASS  Variables';
run;
```

Data set NEW6, shown as Figure 8 below, contains the sum and mean of analysis variable KWH1 for all possible combinations of the three class variables. Suppose, however, that the sum of KWH1 was required only the analysis variable SCHEDULE and the mean of KWH1 was required at all combinations of OFFICE and SERIAL. Finally, a third data set, with the largest and smallest KWH1 values, is required at all combinations of the three classification variables.

At first you might be tempted to run PROC MEANS three separate times to create the required output data sets. That's not necessary, and starting in Version 8 it is even easier to select which values of _TYPE_ will be output to different data sets created in a single invocation of PROC MEANS.

Multiple OUTPUT statements have been permitted in previous releases of SAS System software, but the new option added in Release 8 makes simplifies this process. Before Version 8, you had to figure out the numeric value of the variable _TYPE_ in order to use this feature, which was often difficult and time consuming when you have many classification variables. The CHARTYPE option makes it easy.

Here's how it works: the CHARTYPE option converts the *numeric* value of _TYPE_ to a *character variable* composed of a series of zeros and ones corresponding to the variables in the CLASS Statement. Remember, this is a character variable, even though it is composed of zeros and ones. If you don't believe me, use the CHARTYPE option in PROC MEANS to create an output SAS data set and then use PROC CONTENTS to read the descriptor portion of the data set you created.

You can create multiple output SAS data sets from a single use of PROC MEANS by using the WHERE clause data set option, in conjunction with the CHARTYPE option, to test the character values of _TYPE_ and thus direct observations with desired values of _TYPE_ to the output data sets. Here's an example:

```
proc means noprint
data=sugi26.electricity CHARTYPE;
class office schedule serial;
var kwh1;
output  out=new7(where=(_type_= 010'))
sum=;
output out=new8(where=(_type_='101'))
sum=;

output out=new9(where=(_type_='111'))
sum=;
run;
```

(To save space, these output datasets are not included in the appendix below.)

By specifying the CHARTYPE option, we are able to easily test the value of _TYPE_ for each output data set created. Since there are three classification variables, the values of _TYPE_ have three 'positions' corresponding to the ordering of the variable names in the CLASS statement. The first position is for OFFICE, the second for SCHEDULE and the third for SERIAL. A zero in the appropriate position mean and analysis "without that classification variable" and a one means "with that classification variable." For example, _TYPE_ equal to '101' tells SAS to output just analyses just at the combination of values of OFFICE and SERIAL. When using the CHARTYPE option remember that you are working with a character variable, so you will have to enclose it in single quotes when referring to it.

The previous example showed you how to create separate output SAS data sets with different combinations of the classification variables. You can also create a single output data set with multiple analyses. Here are two ways to do it.

You could easily put several conditions in the WHERE clause in the Output Statement. For example

```
OUTPUT OUT=new10(WHERE=(_TYPE_
IN('010','101','111')) sum=;
```

Will put all observations with the appropriate values of _TYPE_ in to temporary data set NEW10. Another way to approach this task is to use the new TYPES statement, which was added to PROC MEANS in Version 8. This

statement limits the number of combinations of the CLASS variables output to a new SAS data set to just those you specify. Here is an how the TYPES statement would be used to generate the same data set that would have been created from the previous output statement.

```
Proc means noprint
data=sugi26.electricity;
class office schedule serial;
var kwh1;
types schedule
      office * serial
      office * schedule  * serial;
output out=new11 sum=;
run;
```

Additional Tools in PROC MEANS

There are several other useful tools you can use in PROC MEANS, several of which are new to Version 8 of the SAS System. These include:

- The NWAY option, which instructs PROC MEANS to include in the output data set only those observations with the highest possible value of _TYPE_. This is a very useful option if you don't need all the "intermediate" analyses in your output data set.
- Multiple CLASS statements
- Reordering of values of the classification variables using the ORDER=FREQ, ORDER=DESCENDING, ORDER=INTERNAL options in the CLASS statement
- The WAYS statement, to restrict the number of ways that the classification variables are combined
- The DESCENDTYPES option, which reverses the order of observations in the output data set with the highest values of _TYPE_ at the top of the data set and the lowest at the bottom
- Use of Multilabel Formats (created by PROC FORMAT) by specifying the MLF option in the CLASS statement
- Alternative ways to calculate the median (50th percentile), which are useful when requesting PROC MEANS to calculate quantile statistics on large data sets.
- The PRELOADFMT and COMPLETETYPES options, which are used to include observations in output data sets where one or more of the classification variables have missing values.

Summary and Conclusions

PROC MEANS is an very valuable tool for SAS users who need to analyze and summarize observations in their data sets. You can send your analyses to the Output Window or create output SAS data sets. Multiple output SAS data sets can be created in a single invocation of PROC MEANS, thus saving processing resources. The new CHARTYPE option simplifies creating multiple output SAS data sets. New features added to PROC MEANS in Version 8, including the CHARTYPE option, include capabilities to calculate quantile statistics, identification and output of extreme observations to new data sets, the TYPES and WAYS statements, and the ability to use multiple CLASS statements.

Hopefully, this tutorial will get you started with PROC MEANS and make it easier for you to use it. After you've mastered the concepts presented here you'll be able to apply more advanced functions that are described in the PROC MEANS documentation.

Acknowledgements

Thanks to Robert Ray of SAS Institute's BASE Information Technology group for his insights in to PROC MEANS and many of the enhancements added to it in Version 8. Also, many people who have attended my "Summarizing and Reporting Data Using the SAS System" seminar have made comments or asked questions that have challenged me to learn more about PROC MEANS. Thanks are also due to Deb Buck, my good friend and the SUGI 26 Beginning Tutorials Section Chair, for her support and encouragement in the development and presentation of this paper.

Author contact

Andrew H. Karp
President
Sierra Information Services, Inc.
19229 Sonoma Highway PMB 264
Sonoma, California 94115 USA
707 996 7380
SierraInfo@AOL.COM
www.SierraInformation.com

Copyright

SUGI 26 Beginning Tutorials
Electricity Data Set

Obs	sche	div	PRICE	REGION	OFFICE	REV1	KWH1	SALE
1	E1	1	311164	WESTERN	SANTA CRUZ	15.21	133	B
2	E1L	1	352144	EASTERN	SONORA	15.72	162	B
3	E1	1	308311	EASTERN	BISHOP	60.78	505	B
4	E1	1	865208	EASTERN	RIPON	33.75	295	B
5	E1	1	226577	EASTERN	JACKSON	23.91	209	B
6	E1L	1	1017790	WESTERN	HALF MOON BAY	10.38	107	B
7	E1	1	546963	EASTERN	SONORA	28.14	246	F
8	E1	1	806884	WESTERN	SANTA CRUZ	95.99	767	X
9	E1	2	1455859	SOUTHERN	FRESNO	212.54	1658	W
10	E1	2	895807	SOUTHERN	HANFORD	134.09	1078	C
11	E1	2	445268	NORTHERN	RED BLUFF	30.89	270	X
12	E1L	2	1255175	NORTHERN	RED BLUFF	22.76	199	N

Figure 1: Data Set Electricity

Figure 2: Default PROC MEANS Output in the Output Window

SUGI 26 Beginning Tutorials
Electricity Data Set
Default PROC MEANS Results

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
KWH1	12	469.0833333	474.1496615	107.0000000	1658.00
REV1	12	57.0133333	61.5086490	10.3800000	212.5400000

Figure 3: Default PROC MEANS with a WHERE Clause and MAXDEC=2 Options

SUGI 26 Beginning Tutorials
Electricity Data Set
PROC MEANS with a WHERE Clause Data Set Option and the MAXDEC=2 Option

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
KWH1	10	459.20	510.30	107.00	1658.00
REV1	10	55.73	66.16	10.38	212.54

Figure 4: Using the CLASS Statement for BY-Group Analyses

SUGI 26 Beginning Tutorials
Electricity Data Set
Using the CLASS Statement for By-Group Analyses

The MEANS Procedure

Analysis Variable : KWH1

schedule	N Obs	N	Mean	Std Dev	Minimum	Maximum
E1	9	9	573	509	133	1658
E1L	3	3	156	46	107	199

Figure 5: Using the CLASS Statement with Two CLASS Variables and Placing the SUM Statistics Keyword in the PROC MEANS Statement

SUGI 26 Beginning Tutorials
Electricity Data Set
Using the CLASS Statement with Two CLASS Variables
Using the SUM Statistics Keyword
The MEANS Procedure

Analysis Variable : KWH1

REGION	schedule	N Obs	Sum
EASTERN	E1	4	1255
	E1L	1	162
NORTHERN	E1	1	270
	E1L	1	199
SOUTHERN	E1	2	2736
WESTERN	E1	2	900
	E1L	1	107

Figure 6: Output Data Set Created by PROC MEANS' OUTPUT Statement

SUGI 26 Beginning Tutorials
Electricity Data Set
SAS Data Set NEW1 Created by PROC MEANS

Obs	REGION	_TYPE_	_FREQ_	sumkwh1	sumrev1	meankwh1	meanrev1
1		0	12	5629	684.16	469.08333333	57.013333333
2	EASTERN	1	5	1417	162.3	283.4	32.46
3	NORTHERN	1	2	469	53.65	234.5	26.825
4	SOUTHERN	1	2	2736	346.63	1368	173.315
5	WESTERN	1	3	1007	121.58	335.66666667	40.526666667

Figure 7: Data Set NEW4

SUGI 26 Beginning Tutorials
Electricity Data Set
Understanding _TYPE_ and _FREQ_

Obs	division	SERIAL	_TYPE_	_FREQ_	KWH1_Mean	REV1_Sum
1	.		0	12	469.08333333	684.16
2	.	B	1	6	235.16666667	159.75
3	.	C	1	1	1078	134.09
4	.	F	1	1	246	28.14
5	.	N	1	1	199	22.76
6	.	W	1	1	1658	212.54
7	.	X	1	2	518.5	126.88
8	1		2	8	303	283.88
9	2		2	4	801.25	400.28
10	1	B	3	6	235.16666667	159.75
11	1	F	3	1	246	28.14
12	1	X	3	1	767	95.99
13	2	C	3	1	1078	134.09
14	2	N	3	1	199	22.76
15	2	W	3	1	1658	212.54
16	2	X	3	1	270	30.89

Figure 8: Data Set New6 (page one)

SUGI 26 Beginning Tutorials
Electricy Data Set
PROC MEANS Output with three CLASS Variables

Obs	OFFICE	schedule	SERIAL	_TYPE_	_FREQ_	KWH1_Mean	KWH1_Sum
1				0	12	469.08333333	5629
2			B	1	6	235.16666667	1411
3			C	1	1	1078	1078
4			F	1	1	246	246
5			N	1	1	199	199
6			W	1	1	1658	1658
7			X	1	2	518.5	1037
8		E1		2	9	573.44444444	5161
9		E1L		2	3	156	468
10		E1	B	3	4	285.5	1142
11		E1	C	3	1	1078	1078
12		E1	F	3	1	246	246
13		E1	W	3	1	1658	1658
14		E1	X	3	2	518.5	1037
15		E1L	B	3	2	134.5	269
16		E1L	N	3	1	199	199
17	BISHOP			4	1	505	505
18	FRESNO			4	1	1658	1658
19	HALF MOON BAY			4	1	107	107
20	HANFORD			4	1	1078	1078
21	JACKSON			4	1	209	209
22	RED BLUFF			4	2	234.5	469
23	RIPON			4	1	295	295
24	SANTA CRUZ			4	2	450	900
25	SONORA			4	2	204	408

Figure 8: Data Set New6 (page two)

26	BISHOP		B	5	1	505	505
27	FRESNO		W	5	1	1658	1658
28	HALF MOON BAY		B	5	1	107	107
29	HANFORD		C	5	1	1078	1078
30	JACKSON		B	5	1	209	209
31	RED BLUFF		N	5	1	199	199
32	RED BLUFF		X	5	1	270	270
33	RIPON		B	5	1	295	295
34	SANTA CRUZ		B	5	1	133	133
35	SANTA CRUZ		X	5	1	767	767
36	SONORA		B	5	1	162	162
37	SONORA		F	5	1	246	246
38	BISHOP	E1		6	1	505	505
39	FRESNO	E1		6	1	1658	1658
40	HALF MOON BAY	E1L		6	1	107	107

SUGI 26 Beginning Tutorials
Electricy Data Set
PROC MEANS Output with three CLASS Variables

Obs	OFFICE	schedule	SERIAL	_TYPE_	_FREQ_	KWH1_Mean	KWH1_Sum
41	HANFORD	E1		6	1	1078	1078
42	JACKSON	E1		6	1	209	209
43	RED BLUFF	E1		6	1	270	270
44	RED BLUFF	E1L		6	1	199	199
45	RIPON	E1		6	1	295	295
46	SANTA CRUZ	E1		6	2	450	900
47	SONORA	E1		6	1	246	246
48	SONORA	E1L		6	1	162	162
49	BISHOP	E1	B	7	1	505	505
50	FRESNO	E1	W	7	1	1658	1658
51	HALF MOON BAY	E1L	B	7	1	107	107
52	HANFORD	E1	C	7	1	1078	1078
53	JACKSON	E1	B	7	1	209	209
54	RED BLUFF	E1	X	7	1	270	270
55	RED BLUFF	E1L	N	7	1	199	199
56	RIPON	E1	B	7	1	295	295
57	SANTA CRUZ	E1	B	7	1	133	133
58	SANTA CRUZ	E1	X	7	1	767	767