2. USING BASE SAS PROCEDURES

Introduction to

- **SAS system options**
- **Common PROC Step Components**
- **SAS Output Delivery System (ODS)**
- **PROC SORT**
- **PROC CONTENTS**
- **PROC PRINT**
- **PROC FREQ**
- **PROC UNIVARIATE**
- **PROC MEANS**

SAS System Options

- The settings of SAS system options can affect the appearance of the output from procedures, so we'll discuss these before discussing individual procedures.
- SAS System options are parameters you can change that affect the SAS System: how it works, what the output looks like, error handling, and so forth.
- One way to view and change SAS system options is the OPTIONS statement.

OPTIONS Statement

- An OPTIONS statement is submitted as part of a SAS program and affects all steps that follow it.
- It starts with the keyword OPTIONS and continues with a list of options and their values.
- The OPTIONS statement is one of the special SAS statements known as *global* statements that do not belong to either a PROC or a DATA step.
- An OPTIONS statement can appear anywhere in a program and often appears as one of the first statements in a program.

Figure 1: Common Placement of an OPTIONS statement

```
*************************************************************************
*  TITLE :      A Simple SAS Program
*
*  DESCRIPTION: A basic program using the "class" dataset
*
*------------------------------------------------------------------------
*  JOB NAME:    simple-program.SAS
*  LANGUAGE:    SAS, VERSION 9.4
*
*  NAME:        Matthew Psioda
*  DATE:        2017-08-07
*************************************************************************;

FOOTNOTE "Job chapter1.SAS run by <your name> on &SYSDATE at &SYSTIME" ;
OPTIONS MERGENOBY=WARN NODATE NONUMBER LS=95 PS=54;
```

- Any subsequent OPTIONS settings in a program override previous ones.
- To check which options are currently in effect, use the simple PROC OPTIONS syntax below. This will print the system options currently in effect (long list) to the SAS log.

Figure 2: Use of PROC OPTIONS with Partial SAS Log Output

```
PROC OPTIONS; RUN;
```

```
1    PROC OPTIONS; RUN;

    SAS (r) Proprietary Software Release 9.4  TS1M1

Portable Options:

 ANIMATION=STOP Specifies whether to start or stop animation.
 ANIMDURATION=MIN Specifies the number of seconds that each animation frame displays.
 ANIMLOOP=YES Specifies the number of iterations that animated images repeat.
.
.
.
```

Commonly Used Options

Option	Description
CENTER | NOCENTER	This option works as a switch. CENTER centers output on a page, while NOCENTER left justifies output.
DATE | NODATE	This option is also a switch. DATE places the date and time the SAS system was started at the top of each page of output
NUMBER | NONUMBER	This switch controls whether a page number appears on each page of output.

LINESIZE = *n* Abbreviated LS	• Use LINESIZE to control the maximum width of output lines that are printed to the Output window or Listing destination. • This takes values from 64 to 256. • A line size <=98 tends to work well for BIOS 511 assignments. • This option has no effect on non-Listing output such as RTF, HTML, and PDF.
PAGESIZE = *n* Abbreviated PS	• Use PAGESIZE to control the maximum number of lines per page of output printed to the Output window or Listing destination. • This takes values from 15 to 32767. • A page size <=55 tends to work well for BIOS 511 assignments. • This option has no effect on non-Listing output such as RTF, HTML, and PDF.
PAGENO = *n*	Start numbering pages with *n*.
MERGENOBY = WARN	Asks SAS to write a warning to the log if no BY statement is included in a DATA step involving a MERGE.
ORIENTATION = PORTRAIT | LANDSCAPE	Specifies the paper orientation to use for printing.
LABEL | NOLABEL	This switch controls whether SAS procedures print the variable label when printing information about a variable.
FORMDLIM = “delimiting character”	• Specifies the string of characters to be written between pages. • The default is the null string “” which results in a page break after each page. • Specifying a character results in printing a horizontal line of repeating values of that character. • This option has no effect on non-Listing output such as RTF, HTML, and PDF.

NOTES | NONOTES	• This switch controls whether notes are written to the log. • Turning this off is useful for simulations or long programs where too many notes are written to the log.

The PROC Step

Once you have a SAS data set, you can use SAS procedures to analyze and process the data. SAS procedures are computer programs that:

- read SAS data sets
- compute statistics
- write results
- create SAS data sets

SAS has many procedures. In this section we will study some of the procedures used in base SAS to print reports or compute simple statistics. We will examine the following procedures:

Procedure Name	Procedure Description
SORT	Sorts observations by one or more variables.
CONTENTS	Provides information about a SAS data set.
PRINT	Prints the observations in a SAS data set using some or all of the variables.
FREQ	Produces one-way to n-way frequency and cross-tabulation tables as well as many measures of association.
UNIVARIATE	Provides data summarization tools and information on the distribution of numeric variables.
MEANS	Produces descriptive statistics for numeric variables.

Using SAS Procedures

- *Using a procedure is like filling out a form*.
    - SAS Developers designed the form.
    - You have to fill in the blanks and choose from a list of options.
- Each PROC has its own unique form, with its own list of statements and options, but there are similarities between procedures too.
    - Most procedures have printed output.
    - You can customize the general appearance of the output using the system options described previously along with other global statements.
    - Many procedures can also write results to an output SAS data set using an OUTPUT statement or an OUT= option.
    - All procedures begin with a PROC statement.
    - Several statements can be used with almost every procedure. These statements include BY, WHERE, TITLE, FOOTNOTE, LABEL, and FORMAT.
    - TITLE and FOOTNOTE are actually global statements (like OPTIONS) that can be placed anywhere in a program and affect all output until they are changed.
- All procedures can use the Output Delivery System (ODS) to produce results.
- The ODS can also be used to produce an output dataset containing *any* piece of information produced by a procedure.

PROC Statement

- Each PROC step begins with a PROC statement.
- A PROC statement consists of the word PROC followed by the name of the procedure you want to run, followed by any procedure options.
- The procedure options should always include DATA=<LIBREF.FILENAME> to specify the data set that you want to analyze.
    - If a DATA=option is not specified, then the procedure will use the most recently created SAS dataset.
- This example instructs PROC PRINT to print the WEIGHT_CLUB data set, display variable labels instead of names, and wrap the labels based on user-specified instructions.

Figure 3: PROC PRINT Example with HTML Output

```
data weight_club;
 set bios511.weight_club;

 label MemberNum   = 'Member Number'
       StartWeight = 'Starting^Weight'
       EndWeight   = 'Ending^Weight'
       Loss        = 'Weight^Loss';
run;

proc print data=weight_club label split='^'; run;
```

The SAS System

Obs	Member Number	Name	Team	Starting Weight	Ending Weight	Weight Loss
1	1023	David Shaw	red	189	165	24
2	1049	Amelia Serrano	yellow	145	124	21
3	1219	Alan Nance	red	210	192	18
4	1246	Ravi Sinha	yellow	194	177	17
5	1078	Ashley McKnight	red	127	118	9

BY Statement

- A BY statement tells the procedure to perform a specified task *separately* for each combination of values of the BY variables.
- All procedures except PROC SORT assume that your data are already sorted by the variables in your BY statement.
- If they are not, use PROC SORT to do so.

Figure 4: Use of PROC SORT and PROC MEANS with HTML Output

```
proc sort data = bios511.sales out = sales;
  by dept;
run;

proc means data=sales mean median std;
  by dept;
  var price;
run;
```

The SAS System

The MEANS Procedure

DEPT=FURS

Analysis Variable : PRICE		
Mean	Median	Std Dev
648.3166667	649.9750000	105.9273367

DEPT=SHOES

Analysis Variable : PRICE		
Mean	Median	Std Dev
74.6238636	69.9500000	21.5759084

WHERE Statement

- The WHERE statement instructs the procedure to use a subset of the observations in the data set.
- The basic form is: `WHERE condition;`
- Only observations satisfying the condition will be used by the PROC.
- Here are the operators most frequently used in a WHERE condition:

Symbolic	Mnemonic	Example
=	EQ	WHERE team = ‘red’;
^=	NE	WHERE team ^= 'yellow';
>	GT	WHERE loss > 20;
<	LT	WHERE startweight < endweight - 5;
>=	GE	WHERE startweight >= endweight;
<=	LE	WHERE loss <= 5; WHERE 3 < loss <= 5;
&	AND	WHERE loss < 5 & startweight > 150;
|	OR	WHERE endweight < 150 OR loss > 15;
^	NOT	WHERE NOT (3 < loss <= 5)

Other Commonly Used Operators	
Standard mathematical operators + - * / **	WHERE (startweight + 5) > 120;
IN (list)	WHERE team in (‘red’, ‘yellow’);
BETWEEN-AND for an inclusive range	WHERE loss between 5 and 20 Same thing as WHERE 5 <= loss <=20;
CONTAINS for specific string	WHERE team contains (‘y’);
IS MISSING to find missing values or IS NOT MISSING to find non-missing values	WHERE team IS MISSING; WHERE team IS NOT MISSING;

TITLE Statement

- Use TITLE statements to place descriptive information at the top of **every page** of output.
- The basic form of a TITLE statement is TITLE followed by the text of the title in quotes.

```
TITLE 'Average Weight Loss by Team';
```

- You can use either single or double quotes as long as they are the same on either end of the text.
    - If you want to put an apostrophe in a title, use double quotes.
        - `TITLE "Matt's Title Example";`
- You can specify up to ten titles.

```
TITLE1 'Average Weight Loss by Team';
TITLE2 "Spring 2001";
```

- TITLE statements can appear anywhere in a program, not just in a PROC step, but it makes sense to put them in or directly above PROC steps since they affect procedure output.
- Once you specify a title, it is used for all subsequent output until you cancel the title with a null statement or define another title for that line.
    - A null statement for TITLE2 would be the following: `TITLE2;`
- When you specify a new title, it replaces the old title with the same number **and cancels those with a higher number.**

FOOTNOTE Statement

- The FOOTNOTE statement works exactly the same way as the TITLE statement. However, footnotes print at the bottom of the page.

Figure 5: Example use of a WHERE, TITLE, and FOOTNOTE Statements.

```
title 'Everyone';
footnote;

proc print data=weight_club;
run;

title h=2 'The Yellow Team';
footnote h=1 j=c
         color=blue "Note: "
         color=VeryDarkRed "The red team was excluded.";

proc print data=weight_club;
 where team = 'yellow';
run;
```

Everyone

Obs	MemberNum	Name	Team	StartWeight	EndWeight	Loss
1	1023	David Shaw	red	189	165	24
2	1049	Amelia Serrano	yellow	145	124	21
3	1219	Alan Nance	red	210	192	18
4	1246	Ravi Sinha	yellow	194	177	17
5	1078	Ashley McKnight	red	127	118	9

The Yellow Team

Obs	MemberNum	Name	Team	StartWeight	EndWeight	Loss
2	1049	Amelia Serrano	yellow	145	124	21
4	1246	Ravi Sinha	yellow	194	177	17

Note: The red team was excluded.

LABEL Statement

- By default, SAS uses variable names to label your output, but with the LABEL statement you can create more descriptive labels, up to 256 characters long, for each variable.
- When a LABEL statement is used within a DATA step, the labels are *permanently* attached to the variables.
- When used within a PROC step, the labels stay in effect *only for the duration of that step*.
- Labels containing single quotes must be enclosed in double quotes (and vice versa). Otherwise, labels can be enclosed in either single or double quotes.

Figure 6: PROC PRINT w/ and w/o LABEL Statement

```
/* note the LABEL option on the PROC PRINT statement instructs SAS to
   display labels, if present */
proc print data=bios511.weight_club label noobs split='^';
   label MemberNum   = 'Member Number'
         StartWeight = 'Starting^Weight'
         EndWeight   = 'Ending^Weight'
         Loss        = 'Weight^Loss';
run;

proc print data=bios511.weight_club label noobs; run;
```

With LABEL Statement

Member Number	Name	Team	Starting Weight	Ending Weight	Weight Loss
1023	David Shaw	red	189	165	24
1049	Amelia Serrano	yellow	145	124	21
1219	Alan Nance	red	210	192	18

Without LABEL Statement

MemberNum	Name	Team	StartWeight	EndWeight	Loss
1023	David Shaw	red	189	165	24
1049	Amelia Serrano	yellow	145	124	21
1219	Alan Nance	red	210	192	18

FORMAT Statement

- Formats can be used to change the appearance of printed values – how many decimal places to print, to show a $ when a variable contains amounts of money, etc. Base SAS includes many formats for numeric, character, and date values.

- You use a FORMAT statement to associate formats with particular variables. This statement is mentioned here because it is commonly used in PROC steps. However, we will save a detailed discussion of formats for a later chapter.

Figure 7: PROC PRINT w/ FORMAT Statement

```
title "With FORMAT Statement";
proc print data=bios511.weight_club anoobs;
   Format StartWeight EndWeight Loss 5.1;
run;
```

With FORMAT Statement

MemberNum	Name	Team	StartWeight	EndWeight	Loss
1023	David Shaw	red	189.0	165.0	24.0
1049	Amelia Serrano	yellow	145.0	124.0	21.0
1219	Alan Nance	red	210.0	192.0	18.0

The Output Delivery System (ODS)

- The SAS Output Delivery System provides a method of delivering output in a variety of formats, and makes the formatted output easy to access.

- With ODS, one can do the following:

    - Create Listing (old-style SAS output), HTML, RTF, postScript, PDF, and Latex Files

        - Latex → https://support.sas.com/rnd/base/ods/odsmarkup/latex.html

    - Select from many SAS supplied styles to enhance reports

    - Create output objects (e.g. tables) from almost all procedures

    - Create SAS datasets from output objects

- Terminology**:**

    - **Destination:** Destinations are the locations to which ODS routes the output from SAS (e.g., the HTML destination)

    - **Objects:** Output objects are created by ODS to store the formatted results of most SAS procedures

        - An output object consists of the tabular data component from a SAS procedure and formatting instructions provided by a table template that is unique to that SAS procedure

        - Many SAS procedures produce several output objects by default. Additional output objects are available by request (through specification of procedure options)

    - **Styles:** Styles define the presentation attributes of a report, such as font and color. ODS uses style definitions, or templates, to enhance the visual appearance of the output in those destinations that support styles

Figure 8: Basic Structure of ODS Statements

```
ODS TRACE ON </<options>>;                                    /** (1) **/
ODS destination <FILE=filename>;                              /** (2) **/

ODS OUTPUT <output-object-name>=<sas-data-set-name>;          /** (3) **/

ODS destination SELECT  <output-object-name|all|none>;        /** (4) **/
ODS destination EXCLUDE <output-object-name|all|none>;        /** (4) **/

/** ...SAS procedure syntax... **/

ODS destination CLOSE;                                        /** (2) **/
ODS TRACE OFF;                                                /** (1) **/
```

(1) ODS TRACE ON/OFF;

- A useful tool for identifying output objects that are created from SAS procedures – (information is printed to the SAS log)
- You must place it before the first SAS procedure that you want to trace.

(2) ODS destination <FILE=filename>; ODS destination CLOSE;

- Statements open/close a destination for output.
- The destination value can be any of the destinations used for report generation, including LISTING, HTML, RTF, PS, and PDF.
- The FILE= option is required to name the output file. If omitted, SAS will generate a name in most cases (e.g., SASHTML1.html).

(3) ODS OUTPUT <output-object-name>=<sas-data-set-name>;

- Saves the results that are stored in the output objects from SAS procedures to a data set.
- The object and data set names are required for this destination.

(4) ODS destination SELECT/EXCLUDE <output-object-name|all|none>;

- Specifies objects to include in the destination.
- Using SELECT/EXCLUDE is helpful for selecting specific output objects from a large set.
- The object and data set names are required for this destination.

Figure 9: Example Use of Old-Style Listing Destinations

```
option ps = 50 ls = 80 nodate nonumber;
dm "output; clear;"; * command to clear listing window;

** SAS old-style listing destination;
** listing destination /w file name;
ods listing;
proc print data=sashelp.class; run;
ods listing close;

** listing destination w/o file name;
ods listing file = "weight_club.lst";
proc print data=sashelp.class; run;
ods listing close;
```

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Figure 10: Example Use of RTF Destinations w/ ODS TRACE

```
ods trace on;
ods rtf file="/folders/myshortcuts/myFolders/ch2notes.rtf";
  proc univariate data = sashelp.class;
   class sex;
   var height;
  run;
ods rtf close;
ods trace off;
```

SAS LOG

```
63         ods trace on;
64         ods rtf file="/folders/myshortcuts/myFolders/ch2notes.rtf";
NOTE: Writing RTF Body file: /folders/myshortcuts/myFolders/ch2notes.rtf
65          proc univariate data = sashelp.class;
66           class sex;
67           var height;
68          run;


Output Added:
-------------
Name:       Moments
Label:      Moments
Template:   base.univariate.Moments
Path:       Univariate.Height.F.Moments
-------------

Output Added:
-------------
Name:       BasicMeasures
Label:      Basic Measures of Location and Variability
Template:   base.univariate.Measures
Path:       Univariate.Height.F.BasicMeasures
-------------
```

EXCERPT FROM DEFAULT OUTPUT

The UNIVARIATE Procedure
Variable:
Height
Sex = F

Moments			
N	9	Sum Weights	9
Mean	60.5888889	Sum Observations	545.3
Std Deviation	5.01832752	Variance	25.1836111
Skewness	-0.7238643	Kurtosis	-0.3464949
Uncorrected SS	33240.59	Corrected SS	201.468889
Coeff Variation	8.28258714	Std Error Mean	1.67277584

Basic Statistical Measures			
Location		Variability	
Mean	60.58889	Std Deviation	5.01833
Median	62.50000	Variance	25.18361
Mode	.	Range	15.20000
		Interquartile Range	7.80000

Figure 11: Example Use of PDF Destination w/ SELECT and STYLE

```
ods pdf file="P:\Teaching\BIOS-511\lecture-notes\chapter-02\supporting-
program\output\ch2notes.pdf" style=sasweb;
ods pdf select BasicMeasures;
  proc univariate data = sashelp.class;
   class sex;
   var height;
  run;
ods pdf close;
```

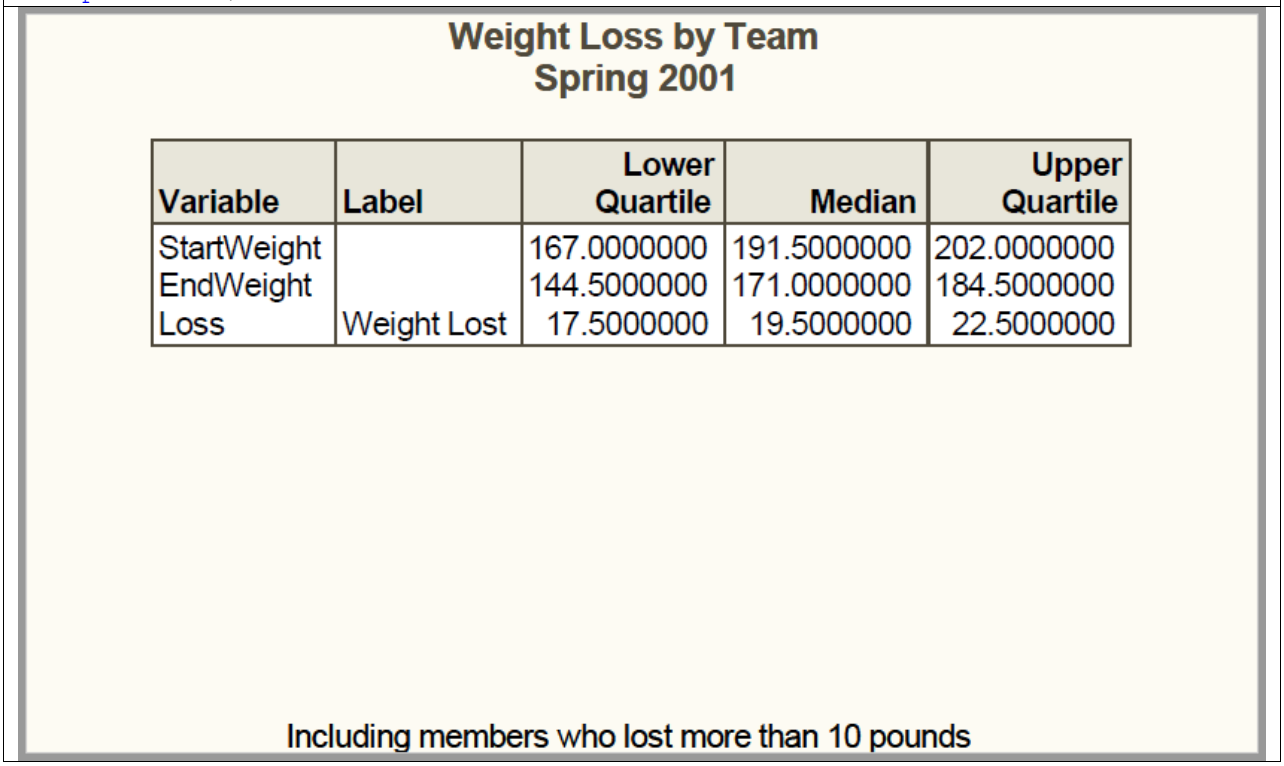
The UNIVARIATE Procedure
Variable: Height
Sex = F

Basic Statistical Measures			
Location		Variability	
Mean	60.58889	Std Deviation	5.01833
Median	62.50000	Variance	25.18361
Mode	.	Range	15.20000
		Interquartile Range	7.80000

ALL SAS OUTPUT

Figure 12: A SAS Program Illustrating the Statements Common to Most Procedures with PDF output

```
options center nodate nonumber orientation=portrait papersize=("5in","3in");

ods noptitle; ** removes "The XXX Procedure" title;

title1 'Weight Loss by Team';
title2 'Spring 2001';
footnote1 'Including members who lost more than 10 pounds';

ods pdf file="weight_club.pdf" style=analysis;
 proc means data=weight_club q1 median q3;
  where loss > 10;
  var startweight endweight loss;
  label loss = 'Weight Lost';
 run;
ods pdf close;
```



Weight Loss by Team
Spring 2001

Variable	Label	Lower Quartile	Median	Upper Quartile
StartWeight		167.0000000	191.5000000	202.0000000
EndWeight		144.5000000	171.0000000	184.5000000
Loss	Weight Lost	17.5000000	19.5000000	22.5000000

Including members who lost more than 10 pounds

The SORT Procedure

- The SORT procedure sorts observations in a SAS data set by one or more variables, storing the sorted observations in a new SAS data set or replacing the original data set.

- Windows and UNIX use the ASCII collating sequence for character variables, shown below.
    - Sorting character variables:

      blank ! " # $ % & ' ( ) * + , - / 0 1 2 3 4 5 6 7 8 9
      : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T
      U V W X Y Z [\ ] ^ _ ` a b c d e f g h I g h I j k l
      m n o p q r s t u v w x y z { | } ~

    - Sorting numeric variables:

      missing values (._ . .A to .Z), numeric values

- Minimal form of the PROC SORT statement:

```
PROC SORT DATA=<libref.filename> OUT=<libref.filename>;
```

- Selected options:

Option	Description
NODUPKEY	Checks for and eliminates observations with duplicate BY values.
NODUPRECS	Checks for and eliminates duplicate observations (all values equal).
DUPOUT=	Specifies a data set to hold any duplicate observations.

- Statements used with PROC SORT:

    - `BY <list of variables>;`

    - `BY <variable list> DESCENDING <variable name> <variable list>;`

        - The `DESCENDING` option sorts by the values of the immediately following variable in descending order (largest to smallest).

        - Use this keyword before the name of each variable whose values you want sorted in descending order.

Figure 13: Use of DESCENDING option with PROC SORT

```
proc sort data=bios511.sales out=sales ;
  by dept descending clerk ;
run;

proc print data=sales noobs ;
  title 'Printing a Sorted SAS Data Set' ;
run;
```

Printing a Sorted SAS Data Set

DEPT	CLERK	PRICE	COST	WEEKDAY	DAY
FURS	BURLEY	599.95	180.01	THR	5
FURS	BURLEY	800.00	240.00	MON	9
FURS	BURLEY	499.95	200.01	SAT	14
FURS	BURLEY	700.00	210.00	THR	19
FURS	AGILE	590.00	182.00	SAT	14
FURS	AGILE	700.00	210.00	WED	25
SHOES	CLEVER	99.95	41.21	TUE	3
SHOES	CLEVER	65.00	33.44	WED	4
SHOES	CLEVER	65.00	33.44	WED	4
SHOES	CLEVER	84.95	38.65	SAT	7
SHOES	CLEVER	54.95	30.00	SAT	7

.
.
.

Figure 14: Use of NODUPKEY option with PROC SORT

```
proc sort data=bios511.sales
           out=sales_nodup nodupkey ;
  by clerk ;
run;

proc print data=sales_nodup noobs ;
  title 'Clerk Names';
  var clerk;
run;
```

Clerk Names

CLERK
AGILE
BURLEY
CLEVER

BY Group Processing

- The BY statement can be used with procedures to perform subgroup processing.
- The PROC will be executed separately for each of the subsets of observations defined by the values of the BY variables.
- Notes:
    - All variables in the BY statement must be in the data set.
    - The data set must be sorted as specified in the BY variable list.
    - DESCENDING may be used before any of the sort variables, but the data set sort order must match this specification.

Figure 15: Example of BY Group Processing

```
proc sort data=bios511.sales out=sales ;
 by dept ;
run;

ods html style=meadow;
title 'PROC MEANS by DEPT' ;
proc means data=sales ;
 by dept ;
 var cost price;
run;
```

PROC MEANS by DEPT

The MEANS Procedure

DEPT=FURS

Variable	N	Mean	Std Dev	Minimum	Maximum
COST	6	203.6700000	22.1033337	180.0100000	240.0000000
PRICE	6	648.3166667	105.9273367	499.9500000	800.0000000

DEPT=SHOES

Variable	N	Mean	Std Dev	Minimum	Maximum
COST	44	35.0363636	4.9674752	23.7600000	43.1800000
PRICE	44	74.6238636	21.5759084	39.9500000	139.9500000

The CONTENTS Procedure

- PROC CONTENTS prints the descriptor information from a SAS data set.
- The output lists data set attributes followed by all variables and their attributes in alphabetical order.
    - Variable attributes displayed include type, length (number of bytes used to store the variable), label (if present), and format (if present).
- PROC CONTENTS is especially useful for documenting a permanent data set since it displays the data set creation date and date of last modification.
- Basic form of the PROC statement:

```
PROC CONTENTS DATA = data  < options > ;
```

- Selected procedure statement options:

OUT=	Produces a SAS data set containing information about the variables in the input data set.
NOPRINT	Suppresses the printed output (useful with the OUT= option).
ORDER= IGNORECASE	Prints the list of variables in alphabetical order ignoring the case of the letters.
VARNUM	Prints variables by their order in the dataset instead of alphabetically.

Figure 16: Use of PROC CONTENTS with HTML Output

```
PROC CONTENTS DATA=bios511.weight_club; RUN;
```

The CONTENTS Procedure

Data Set Name	BIOS511.WEIGHT_CLUB	Observations	5
Member Type	DATA	Variables	6
Engine	V9	Indexes	0
Created	09/27/2010 15:36:31	Observation Length	64
Last Modified	09/27/2010 15:36:31	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
5	EndWeight	Num	8
6	Loss	Num	8
1	MemberNum	Num	8
2	Name	Char	19
4	StartWeight	Num	8
3	Team	Char	8

The PRINT Procedure

- The PRINT procedure lists the observations in a SAS data set, using all or some of the variables.
- Features include:
    - Automatic formatting
    - Columns labeled with variable names or labels
    - Special handling of section and page breaks
    - Accumulation and printing of subtotals and totals
    - Special BY/ID formatting

Figure 17: PROC PRINT Syntax

```
PROC PRINT DATA = <libref.filename> <option(s)>;
      BY <DESCENDING> variable-1 <...<DESCENDING> variable-n><NOTSORTED>;
      PAGEBY BY-variable;
      SUMBY BY-variable;
      ID variable(s) <option>;
      SUM variable(s) <option>;
      VAR variable(s) <option>;
RUN;
```

- Description of internal PROC statements:

Statement	Description

VAR	• selects the variables to appear in the printed output
	• If omitted, all variables are included
	• Example uses:
	  ○ VAR x1 x2 x3;
	  ○ VAR name -- age; *(using positional order)*
	  ○ VAR x1-x10; *(using suffix ranges)*
	  ○ VAR _NUMERIC_; *(all numeric variables)*

ID	• If an ID statement is included in a PROC PRINT step, observations in the listing are identified by the value of the ID variable(s) rather than by observation number. • The variables in the ID statement list appear to the left of any variables in the VAR statement list. • Even without an ID statement, the NOOBS option of the PROC PRINT statement can be used to turn off observation numbers.
PAGEBY	• The PAGEBY statement can be used to identify a variable appearing in the BY statement in the same PROC PRINT step. • If the value of this BY variable changes, or if the value of any BY variable that precedes it in the BY statement changes, PROC PRINT begins printing a new page.
SUM	• The SUM statement identifies any numeric variables to total in the report.
SUMBY	• The SUMBY statement can be used to identify a variable appearing in the BY statement in the same PROC PRINT step. • If the value of this BY variable changes, or if the value of any BY variable that precedes it in the BY statement changes, PROC PRINT prints the sums of all variables listed in the SUM statement.

- Selected procedure statement options:

Option	**Description**
LABEL	Uses variable labels as column headings. If this option is not specified, variable names are used as column headings.
N	Prints the number of observations at the end of the printed output.
NOOBS	Suppresses the observation number column in the printed output.

SPLIT = “split character”	Splits column labels onto a new line after a particular character.

Figure 18: Example 1 -- Use of PROC PRINT

```
ods html newfile=proc style=dtree;
options center nodate nonumber;
proc print data=bios511.weight_club n noobs;
  sum loss;

  title1 h=2.0 'Listing of the Weight_Club Data Set';
  title2 h=0.5 'Using the N and NOOBS Options';
  title3 h=0.5 'Also, Using a SUM Statement But No VAR Statement';
run;
```

Listing of the Weight_Club Data Set
Using the N and NOOBS Options
Also, Using a SUM Statement But No VAR Statement

MemberNum	Name	Team	StartWeight	EndWeight	Loss
1023	David Shaw	red	189	165	24
1049	Amelia Serrano	yellow	145	124	21
1219	Alan Nance	red	210	192	18
1246	Ravi Sinha	yellow	194	177	17
1078	Ashley McKnight	red	127	118	9
					89
N = 5					

Figure 19: Example 2 -- Use of PROC PRINT to display first 8 Observations

```
ods html newfile=proc style=sasweb;
options  center nodate nonumber;

title1 'Listing of the First 8 Observations of the SALES Data Set';
title2 'Only Character Variables';
proc print data=bios511.sales(obs=8);
  var _character_;
run;

title1 'Listing of the Observation 5-8 of the SALES Data Set';
title2 'Only Character Variables';
proc print data=bios511.sales(firstobs=5 obs=8);
  var _character_;
run;
```

Listing of the First 8 Observations of the SALES Data Set
Only Character Variables

Obs	DEPT	CLERK	WEEKDAY
1	SHOES	CLEVER	TUE
2	SHOES	AGILE	WED
3	SHOES	CLEVER	WED
4	SHOES	CLEVER	WED
5	FURS	BURLEY	THR
6	SHOES	AGILE	THR
7	SHOES	AGILE	THR
8	SHOES	BURLEY	THR

Listing of the Observation 5-8 of the SALES Data Set
Only Character Variables

Obs	DEPT	CLERK	WEEKDAY
5	FURS	BURLEY	THR
6	SHOES	AGILE	THR
7	SHOES	AGILE	THR
8	SHOES	BURLEY	THR

Figure 20: Example 4 – Use of SUMBY Statement

```
ods html newfile=proc style=journal2;

proc sort data=bios511.sales out=sales;
  by dept;
run;

title1 'Partial Listing of Sales Data Set';
title2 'Using Selected Options and Statements';

proc print data=sales label;
  by dept;

  var clerk price cost;
  sum price cost;
  sumby dept;

  label cost  = 'Cost of Item'
        price = 'Price of Item'
        dept  = 'Department'
        clerk = 'Clerk';
run;
```

Partial Listing of Sales Data Set
Using Selected Options and Statements

Department=FURS

Obs	Clerk	Price of Item	Cost of Item
1	BURLEY	599.95	180.01
2	BURLEY	800.00	240.00
3	AGILE	590.00	182.00
4	BURLEY	499.95	200.01
5	BURLEY	700.00	210.00
6	AGILE	700.00	210.00
DEPT		*3889.90*	*1222.02*

Department=SHOES

Obs	Clerk	Price of Item	Cost of Item
7	CLEVER	99.95	41.21
8	AGILE	95.00	40.49
9	CLEVER	65.00	33.44
10	CLEVER	65.00	33.44

.
.
.

43	BURLEY	59.95	31.78
44	CLEVER	84.95	38.65
45	CLEVER	54.95	30.00
46	AGILE	64.95	33.43
47	BURLEY	54.95	30.00
48	BURLEY	55.00	30.02
49	CLEVER	64.95	33.43
50	CLEVER	129.95	43.18
DEPT		*3283.45*	*1541.60*
		7173.35	*2763.62*

The FREQ Procedure

- The FREQ procedure produces one-way to n-way frequency and cross-tabulation tables and is used to compute counts and percents.
- You can use it to answer these questions:
    - What are the different values of a certain variable in my data set?
    - How many times does each value occur?
- Cross-tabulation tables, also known as contingency tables, show the number of observations for each combination of variable values of two or more variables.
- Features include:
    - frequency counts of variable values along with related percentages
    - tests for proportions for one-way tables
    - combined frequencies for two or more variables
    - weighted frequencies
    - measures of association and tests (chi-square and exact) for n-way tables
    - ability to output frequencies to a SAS data set
    - stratified analysis, within and across strata

Figure 21: PROC FREQ Syntax (Restricted to Basic Statements);

```
PROC FREQ DATA = <libref.filename> <options> ;
    BY variables;
    TABLES requests </ options> ;
    WEIGHT variable </ options> ;
RUN;
```

- Description of internal PROC statements:

Statement	Description
TABLE	• The TABLES statement requests one-way to n-way frequency and cross-tabulation tables and statistics for those tables. • If you omit the TABLES statement, PROC FREQ generates one-way frequency tables for all data set variables that are not listed in the other statements. • One can use any number of TABLE statements in a single PROC FREQ step • By default, missing levels of each variable are excluded from the table, but the total frequency of missing observations is printed below each table.
WEIGHT	• The WEIGHT statement names a numeric variable that provides a weight for each observation in the input data set. • The WEIGHT statement is most commonly used to input cell count data (e.g., pre-summarized data).

- Selected PROC statement options:

Option	Description
NOPRINT	Suppress all printed output
NLEVELS	Display a table of the number of levels of each variable in any TABLES
ORDER = DATA | FORMATTED | FREQ | INTERNAL	Control the order in which values appear in the tables; for details, see the PROC FREQ documentation

- Selected TABLE statement options:

Option	Description
NOFREQ	Suppresses printing of cell frequencies
NOCUM	Suppresses printing of cumulative frequencies and percentages
NOPERCENT	Suppresses printing of cell percentages
NOROW	Suppresses printing of row percentages
NOCOL	Suppresses printing of column percentages
MISSPRINT	Prints missing value frequencies for two- to n-way tables; the frequencies are not used in the calculation of statistics
MISSING	Interprets missing values as non-missing and includes them in calculations of percentages and other statistics
LIST	Presents two- to n-way tables in list format rather than as cross-tabulation tables
OUT=	Names an output data set to contain variable values and frequency counts
OUTPCT	Adds percentages to OUT= data set
SPARSE	Lists all possible combinations of the variable values for an n-way table, even if a combination does not occur in the data; only has an effect with the LIST or OUT= option
ALL	Requests tests and measures of association produced by CHISQ, MEASURES, and CMH
CHISQ	Requests chi-square tests and measures of association based on chi-square
MEASURES	Requests several measures of association, including odds ratios and relative risks for 2x2 tables
CMH	Computes Cochran-Mantel-Haenzel statistics

Many other statistics can also be requested, including Fisher's exact test, binomial statistics for one-way tables, kappa coefficients, and polychoric correlation coefficients. See the PROC FREQ documentation for details.

Figure 22: Example 1 – Use of PROC FREQ (Basic Use)

```
proc freq data=bios511.sales;
  tables dept clerk dept*clerk;
  title1 "Frequency Distributions and Cross-tabulations";
run;

/** equivalent code **/
proc freq data=bios511.sales;
  tables dept;
  tables clerk;
  tables dept*clerk;
run;
```

Frequency Distributions and Cross-tabulations

The FREQ Procedure

DEPT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
FURS	6	12.00	6	12.00
SHOES	44	88.00	50	100.00

CLERK	Frequency	Percent	Cumulative Frequency	Cumulative Percent
AGILE	13	26.00	13	26.00
BURLEY	9	18.00	22	44.00
CLEVER	28	56.00	50	100.00

Table of DEPT by CLERK

DEPT Frequency Percent Row Pct Col Pct	AGILE	BURLEY	CLEVER	Total
FURS	2 4.00 33.33 15.38	4 8.00 66.67 44.44	0 0.00 0.00 0.00	6 12.00
SHOES	11 22.00 25.00 84.62	5 10.00 11.36 55.56	28 56.00 63.64 100.00	44 88.00
Total	13 26.00	9 18.00	28 56.00	50 100.00

Figure 23: Example 2 – Use of PROC FREQ (Suppressed Percentages)

```
proc freq data=bios511.sales;
   tables dept*clerk / norow nocol nopercent;
   title1 "Table with Percentages Suppressed";
run;
```

Table with Percents Suppressed

The FREQ Procedure

Table of DEPT by CLERK

DEPT CLERK

Frequency	AGILE	BURLEY	CLEVER	Total
FURS	2	4	0	6
SHOES	11	5	28	44
Total	13	9	28	50

Figure 24: Use of PROC FREQ (Output Dataset)

```
title1 "Table underlying output data sets";
proc freq data=bios511.sales noprint;
  tables dept*clerk / out=no_outpct;
  tables dept*clerk / out=with_outpct outpct;
run;

title1 "Data set without OUTPCT";
proc print data=no_outpct; run;

title1 "Data set with OUTPCT";
proc print data=with_outpct; run;
```

Data set without OUTPCT

Obs	DEPT	CLERK	COUNT	PERCENT
1	FURS	AGILE	2	4
2	FURS	BURLEY	4	8
3	SHOES	AGILE	11	22
4	SHOES	BURLEY	5	10
5	SHOES	CLEVER	28	56

Data set with OUTPCT

Obs	DEPT	CLERK	COUNT	PERCENT	PCT_ROW	PCT_COL
1	FURS	AGILE	2	4	33.3333	15.385
2	FURS	BURLEY	4	8	66.6667	44.444
3	SHOES	AGILE	11	22	25.0000	84.615
4	SHOES	BURLEY	5	10	11.3636	55.556
5	SHOES	CLEVER	28	56	63.6364	100.000

Note: Output has been reorganized for display.

Figure 25: Use of PROC FREQ (Missing Data)

```
title1 "Missing Data Example";
proc freq data=bios511.clin ;
  where visit in (1,2);
   tables evdnk*visit / norow nocol;
   tables evdnk*visit / norow nocol missprint ;
   tables evdnk*visit / norow nocol missing ;
run;
```

Frequency
Percent

Table of EVDNK by VISIT

EVDNK(EVER DRINK(1=NO 2=YES))	VISIT 1	VISIT 2	Total
1	0 0.00	16 14.29	16 14.29
2	0 0.00	96 85.71	96 85.71
Total	0 0.00	112 100.00	112 100.00

Frequency Missing = 184

Table of EVDNK by VISIT

EVDNK(EVER DRINK(1=NO 2=YES))	VISIT 1	VISIT 2	Total
.	148 .	36 .	. .
1	0 0.00	16 14.29	16 14.29
2	0 0.00	96 85.71	96 85.71
Total	0 0.00	112 100.00	112 100.00

Frequency Missing = 184

Table of EVDNK by VISIT

EVDNK(EVER DRINK(1=NO 2=YES))	VISIT 1	VISIT 2	Total
.	148 50.00	36 12.16	184 62.16
1	0 0.00	16 5.41	16 5.41
2	0 0.00	96 32.43	96 32.43
Total	148 50.00	148 50.00	296 100.00

Note: Titles were cropped out of screenshot.

Figure 26: Use of PROC FREQ with ALL option

```
title 'Example with the ALL Option';
ods html newfile=proc style=default;
proc freq data=bios511.sales2 ;
  tables sex*dept / norow nopercent nocol all ;
run;
```

Example with the ALL Option

Frequency

Table of SEX by DEPT

SEX	DEPT FURS	DEPT SHOES	Total
FEMALE	1	11	12
MALE	3	31	34
Total	4	42	46

Frequency Missing = 4

Statistics for Table of SEX by DEPT

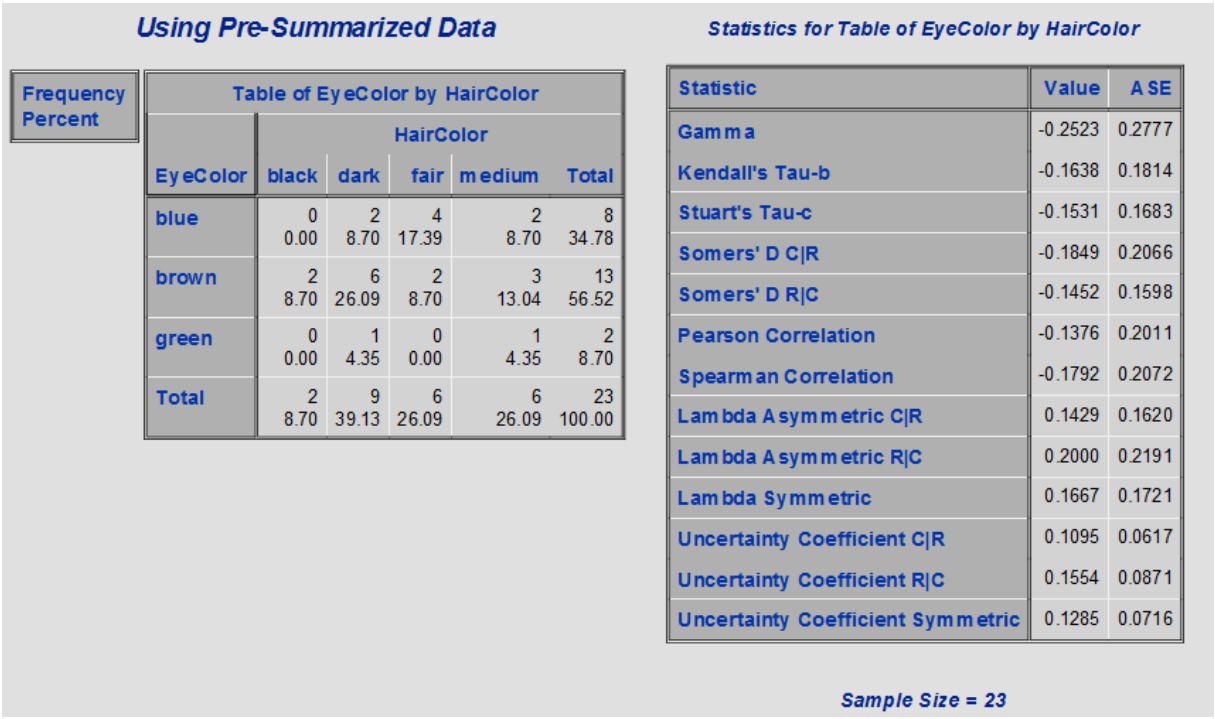
Statistic	DF	Value	Prob
Chi-Square	1	0.0027	0.9587
Likelihood Ratio Chi-Square	1	0.0027	0.9585
Continuity Adj. Chi-Square	1	0.0000	1.0000
Mantel-Haenszel Chi-Square	1	0.0026	0.9591
Phi Coefficient		-0.0076	
Contingency Coefficient		0.0076	
Cramer's V		-0.0076	

WARNING: 50% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Notes: (1) Output reorganized for display in figure; (2) Partial output displayed;

Figure 27: Analyzing Pre-Summarized Data

```
data colors;
 input EyeColor $ HairColor $ inputCount;
 datalines;
blue       fair       4
blue       medium     2
blue       dark       2
brown      fair       2
brown      medium     3
brown      dark       6
brown      black      2
green      medium     1
green      dark       1
;
run;
**dm "viewtable work.colors";

Title "Using Pre-Summarized Data";
proc freq data=colors;
  weight inputCount;
  tables Eyecolor * Haircolor /  nocol norow measures;
run;
```



Using Pre-Summarized Data

Frequency Percent	Table of EyeColor by HairColor				
	HairColor				
EyeColor	black	dark	fair	medium	Total
blue	0 0.00	2 8.70	4 17.39	2 8.70	8 34.78
brown	2 8.70	6 26.09	2 8.70	3 13.04	13 56.52
green	0 0.00	1 4.35	0 0.00	1 4.35	2 8.70
Total	2 8.70	9 39.13	6 26.09	6 26.09	23 100.00

Statistics for Table of EyeColor by HairColor

Statistic	Value	ASE
Gamma	-0.2523	0.2777
Kendall's Tau-b	-0.1638	0.1814
Stuart's Tau-c	-0.1531	0.1683
Somers' D C|R	-0.1849	0.2066
Somers' D R|C	-0.1452	0.1598
Pearson Correlation	-0.1376	0.2011
Spearman Correlation	-0.1792	0.2072
Lambda Asymmetric C|R	0.1429	0.1620
Lambda Asymmetric R|C	0.2000	0.2191
Lambda Symmetric	0.1667	0.1721
Uncertainty Coefficient C|R	0.1095	0.0617
Uncertainty Coefficient R|C	0.1554	0.0871
Uncertainty Coefficient Symmetric	0.1285	0.0716

Sample Size = 23

Note: Output reorganized for display.

The UNIVARIATE Procedure

- The UNIVARIATE procedure provides data summarization tools, graphics displays, and information on the distribution of numeric variables.
- Features:
    - Descriptive statistics
    - Details on extreme values and extreme observations
    - Median, mode, range, and quantiles
    - Tests for location and normality
    - Confidence limits
    - Low-resolution plots to picture the distribution
    - High-resolution histograms, quantile-quantile plots, and probability plots
    - Frequency tables
    - Output data sets

Figure 28: PROC UNIVARIATE Syntax (Restricted to Basic Statements);

```
PROC UNIVARIATE DATA = <libref.filename> <options> ;
    BY variables ;
    CLASS variable-1 <(v-options)> <variable-2 <(v-options)>> ;
    HISTOGRAM <variables> < / options> ;
    ID variables ;
    INSET keyword-list </ options> ;
    OUTPUT <OUT=SAS-data-set> <keyword1=names ...keywordk=names>;
    VAR variables ;
    WEIGHT variable ;
RUN;
```

- Description of commonly used internal PROC statements:

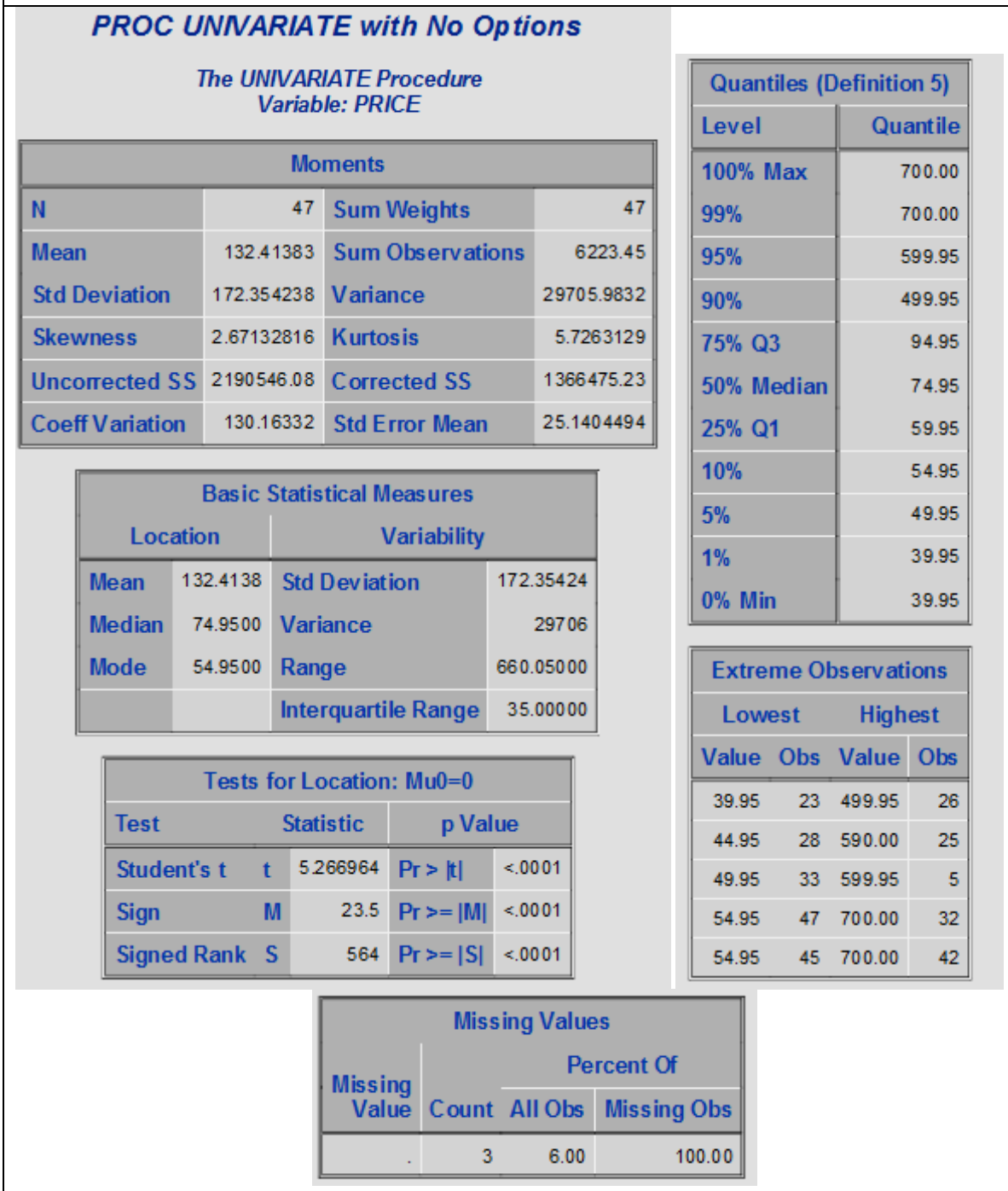
Statement	**Description**
CLASS	• The CLASS statement specifies **at most two variables** used to group the data into classification levels. • Variables in a CLASS statement are referred to as *CLASS variables*. • CLASS variables can be numeric or character. • You do not have to sort the data by CLASS variables. • PROC UNIVARIATE uses the formatted values of the CLASS variables to determine the classification levels.
WEIGHT	• The WEIGHT statement names a numeric variable that provides a weight for each observation in the input data set.
ID	• Identifies extreme values (used for display)
OUTPUT OUT=	• The OUTPUT statement saves statistics and BY variables in an output data set. • When you use a BY statement, each observation in the OUT= data set corresponds to one of the BY groups. Otherwise, the OUT= data set contains only one observation.
HISTOGRAM	• The HISTOGRAM statement creates histograms and optionally superimposes estimated parametric and nonparametric probability density curves. • You cannot use the WEIGHT statement with the HISTOGRAM statement.
INSET	• An INSET statement places a box or table of summary statistics, called an *inset*, directly in a graph created with a HISTOGRAM statement (or CDFPLOT, PPPLOT, PROBPLOT, or QQPLOT statement).

- Selected PROC statement options:

Option	Description
NOPRINT	Suppress all printed output
FREQ	• Requests a frequency table that consists of the variable values, frequencies, cell percentages, and cumulative percentages.
NORMAL	• Requests tests for normality that include a series of goodness-of-fit tests based on the empirical distribution function
PLOT/ PLOTS	• Produces a stem-and-leaf plot (or a horizontal bar chart), a box plot, and a normal probability plot in line printer output. • If you use a BY statement, side-by-side box plots that are labeled "Schematic Plots" appear after the univariate analysis for the last BY group.
CIBASIC	• Requests confidence limits for the mean, standard deviation, and variance based on the assumption that the data are normally distributed.
NEXTROBS=n	• Specifies the number of extreme observations that PROC UNIVARIATE lists in the table of extreme observations. • The table lists the lowest observations and the highest observations. • The default value is 5.

Figure 29: Example 1 -- Basic Use of PROC UNIVARIATE

```
proc univariate data=bios511.sales2;
  var price;
  title1 'PROC UNIVARIATE with No Options';
run;
```



PROC UNIVARIATE with No Options

The UNIVARIATE Procedure
Variable: PRICE

Moments			
N	47	Sum Weights	47
Mean	132.41383	Sum Observations	6223.45
Std Deviation	172.354238	Variance	29705.9832
Skewness	2.67132816	Kurtosis	5.7263129
Uncorrected SS	2190546.08	Corrected SS	1366475.23
Coeff Variation	130.16332	Std Error Mean	25.1404494

Basic Statistical Measures			
Location		Variability	
Mean	132.4138	Std Deviation	172.35424
Median	74.9500	Variance	29706
Mode	54.9500	Range	660.05000
		Interquartile Range	35.00000

Tests for Location: Mu0=0				
Test		Statistic	p Value	
Student's t	t	5.266964	Pr > |t|	<.0001
Sign	M	23.5	Pr >= |M|	<.0001
Signed Rank	S	564	Pr >= |S|	<.0001

Quantiles (Definition 5)	
Level	Quantile
100% Max	700.00
99%	700.00
95%	599.95
90%	499.95
75% Q3	94.95
50% Median	74.95
25% Q1	59.95
10%	54.95
5%	49.95
1%	39.95
0% Min	39.95

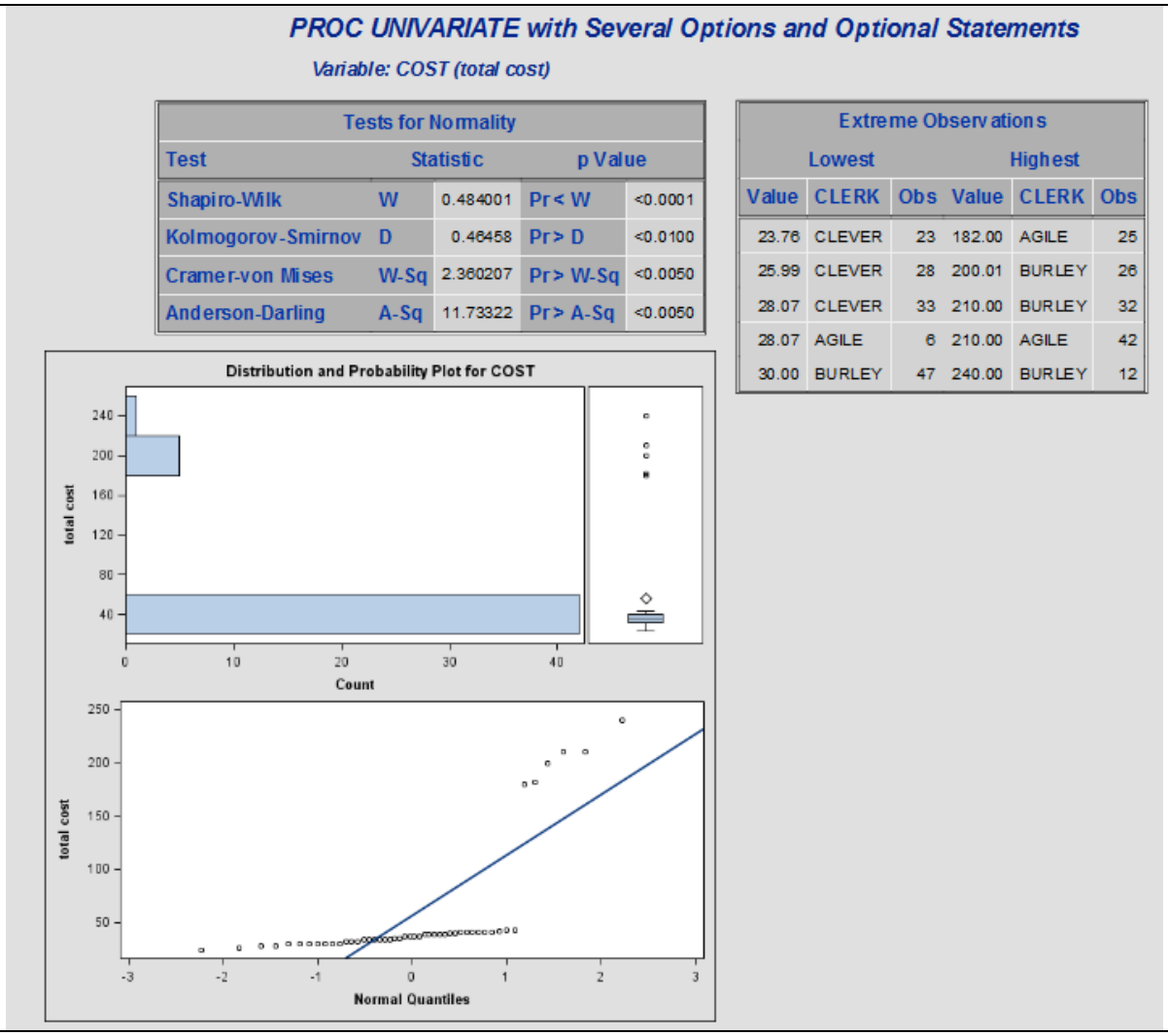
Extreme Observations			
Lowest		Highest	
Value	Obs	Value	Obs
39.95	23	499.95	26
44.95	28	590.00	25
49.95	33	599.95	5
54.95	47	700.00	32
54.95	45	700.00	42

Missing Values			
Missing Value	Count	Percent Of All Obs	Percent Of Missing Obs
.	3	6.00	100.00

Note: Output reformatted for display.

Figure 30: Example 2 – Basic Use of PROC UNIVARIATE

```
ods html select plots ExtremeObs testsForNormality ;
proc univariate data=bios511.sales2 plot normal ;
  var cost ;
  id clerk ;
  label cost='total cost' ;
  title1 'PROC UNIVARIATE with Several Options and Optional
          Statements';
run;
```



PROC UNIVARIATE with Several Options and Optional Statements

Variable: COST (total cost)

Tests for Normality

Test		Statistic	p Value	
Shapiro-Wilk	W	0.484001	Pr< W	<0.0001
Kolmogorov-Smirnov	D	0.46458	Pr> D	<0.0100
Cramer-von Mises	W-Sq	2.360207	Pr> W-Sq	<0.0050
Anderson-Darling	A-Sq	11.73322	Pr> A-Sq	<0.0050

Extreme Observations

Lowest			Highest		
Value	CLERK	Obs	Value	CLERK	Obs
23.76	CLEVER	23	182.00	AGILE	25
25.99	CLEVER	28	200.01	BURLEY	26
28.07	CLEVER	33	210.00	BURLEY	32
28.07	AGILE	6	210.00	AGILE	42
30.00	BURLEY	47	240.00	BURLEY	12

Figure 31: Example 3 -- Use of PROC UNIVARIATE (OUTPUT Statement)

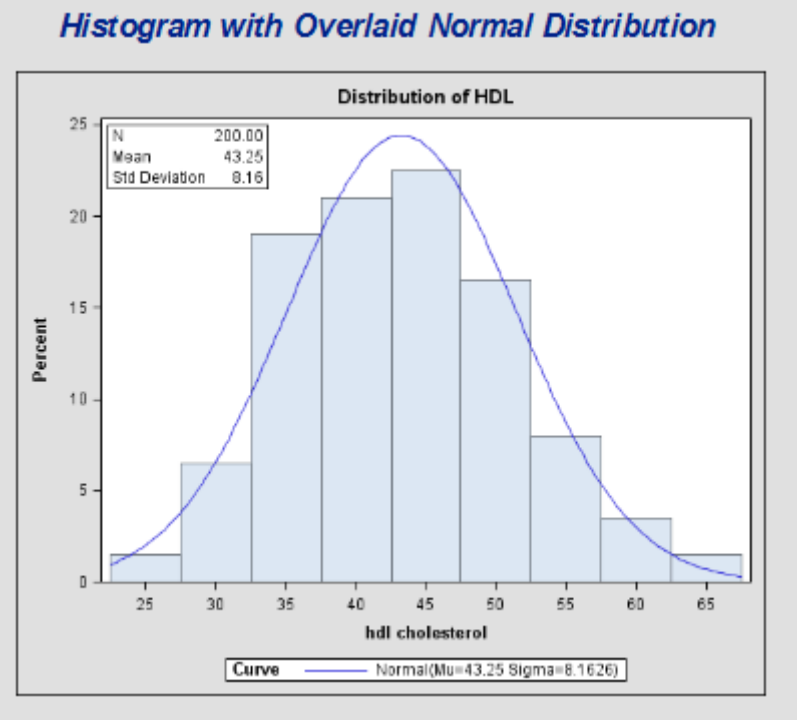
```
proc sort data=bios511.sales2 out=sales;
    by sex;
run;

proc univariate data=sales noprint;
    by sex;
    var cost price;              /** two input variables */
    output out=stats
           n    =ncost   nprice   /** two output variables per stat. */
           nmiss=nmcost  nmprice
           mean =mcost   mprice
           max  =maxcost maxprice;
run;

proc print data=stats;
    title1 'Output Data Set from PROC UNIVARIATE';
run;
```

Output Data Set from PROC UNIVARIATE

Obs	SEX	ncost	nprice	mcost	mprice	nmcost	nmprice	maxcost	maxprice
1		2	2	111.240	342.475	0	0	182	590
2	FEMALE	11	11	51.825	133.150	1	1	210	700
3	MALE	35	34	54.281	119.819	1	2	240	700

Figure 32: Example 4 -- Use of PROC UNIVARIATE (HISTOGRAM Statement)

```
title1 'Histogram with Overlaid Normal Distribution';
ods graphics / height=5in width=6in; ** changes the size of graph;
ods html select histogram;
proc univariate data=bios511.hw4 noprint;
  histogram hdl / normal;
  inset n mean std / format=6.2;
run;
```



Note: One can fit many other distributions, see documentation.

The MEANS Procedure

- The MEANS procedure produces simple univariate descriptive statistics for numeric variables.
- Selected univariate statistics
- Special BY group processing: using a BY or CLASS statement will cause MEANS to calculate descriptive statistics separately for groups of observations
- Printed output is optional
- Output data set of summary statistics is optional
- For the set of capabilities that PROC MEANS shares with PROC UNIVARIATE, most users prefer PROC MEANS.

Figure 33: PROC MEANS Syntax (Restricted to Basic Statements)

```
PROC MEANS DATA = <libref.filename> <option(s)> <statistic-keyword(s)>;
      BY <DESCENDING> variable-1 <... <DESCENDING> variable-n><NOTSORTED>;
      CLASS variable(s) </ option(s)>;
      ID variable(s);
      OUTPUT <OUT=SAS-data-set> <output-statistic-specification(s)>
             <id-group-specification(s)> <maximum-id-specification(s)>
             <minimum-id-specification(s)> </ option(s)> ;
      TYPES request(s);
      VAR variable(s) < / WEIGHT=weight-variable>;
      WAYS list;
      WEIGHT variable;
RUN;
```

- Description of internal PROC statements:

Statement	Description
CLASS	• The CLASS statement specifies variables used to group the data into classification levels. • Variables in a CLASS statement are referred to as *CLASS variables*. • CLASS variables can be numeric or character. • You do not have to sort the data by CLASS variables.
WEIGHT	• The WEIGHT statement names a numeric variable that provides a weight for each observation in the input data set.
ID	• Identifies extreme values (used for display)
OUTPUT OUT=	• The OUTPUT statement saves statistics and BY variables in an output data set. • When you use a BY statement, each observation in the OUT= data set corresponds to one of the BY groups. Otherwise, the OUT= data set contains only one observation. • You can use any number of OUTPUT statements;
WAYS	• Specifies the number of ways to make unique combinations of the class variables
TYPES	• Specifies which of the 2^k combinations of class variables are used

- Selected PROC statement options:

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>FW</td><td>• Specify the field width for the statistics</td></tr>
<tr><td>MISSING</td><td>• Use missing values as valid values to create combinations of class variables</td></tr>
<tr><td>NOPRINT</td><td>• Suppresses printed output</td></tr>
<tr><td>MAXDEC</td><td>• Specifies the maximum number of decimal places</td></tr>
<tr><td>NWAY</td><td>• Specifies that only the observations with the highest _TYPE_ value be put into the output data set.
• <u>This results in an output dataset containing no summary rows.</u></td></tr>
<tr><td>Selected statistic-keywords</td><td>(Default are N, MEAN, STD, MIN, and MAX):
<table>
<tr><td>N</td><td>Number of non-missing values</td></tr>
<tr><td>NMISS</td><td>Number of missing values</td></tr>
<tr><td>MEAN</td><td>The mean</td></tr>
<tr><td>SUM</td><td>The sum of all values for a given variable</td></tr>
<tr><td>STD</td><td>Standard deviation</td></tr>
<tr><td>STDERR</td><td>Standard error</td></tr>
<tr><td>VAR</td><td>Variance</td></tr>
<tr><td>MAX</td><td>Maximum</td></tr>
<tr><td>MIN</td><td>Minimum</td></tr>
<tr><td>CLM</td><td>Confidence interval for the mean</td></tr>
<tr><td>MEDIAN</td><td>The median (50th percentile)</td></tr>
<tr><td>P1</td><td>1st percentile</td></tr>
<tr><td>P5</td><td>5th percentile</td></tr>
<tr><td>P10</td><td>10th percentile</td></tr>
<tr><td>P90</td><td>90th percentile</td></tr>
<tr><td>P95</td><td>95th percentile</td></tr>
<tr><td>P99</td><td>99th percentile</td></tr>
<tr><td>Q1</td><td>25th percentile</td></tr>
<tr><td>Q3</td><td>75th percentile</td></tr>
<tr><td>T</td><td>t-test</td></tr>
<tr><td>PROBT</td><td>P-value based on t</td></tr>
</table></td></tr>
</table>

- Note: If you have one or more CLASS variables and you are using an OUTPUT statement to produce an output data set, you almost certainly should use the NWAY option on the PROC MEANS statement.

Figure 34: Example 1 -- Use of PROC MEANS

```
proc means data=bios511.sales2;
  title1 'PROC MEANS with No Options';
run;
```

PROC MEANS with No Options

Variable	N	Mean	Std Dev	Minimum	Maximum
PRICE	47	132.4138298	172.3542376	39.9500000	700.0000000
COST	48	56.0918750	57.0151396	23.7600000	240.0000000
DAY	50	15.1000000	7.8642825	3.0000000	27.0000000

Figure 35: Example 2 -- Use of PROC MEANS with Select Statistics

```
proc means data=bios511.sales2 mean min max fw=7 maxdec=2;
  var cost price ;
  title1 h=2 'PROC MEANS: Request Specific Statistics for
           Selected Variables';
run;
```

PROC MEANS: Request Specific Statistics for Selected Variables

Variable	Mean	Minimum	Maximum
COST	56.09	23.76	240.00
PRICE	132.41	39.95	700.00

Figure 36: Example 3 – Use of BY Statement

```
proc sort data=bios511.sales2 out=sales ;
  by sex;
run;

proc means data=sales maxdec=2 n mean max ;
  by sex ;
  title1 color=black 'PROC MEANS: Summary by sex using '
         color=red   'BY '
         color=black 'Statement' ;
run;
```

PROC MEANS: Summary by sex using BY Statement

SEX=' '

Variable	N	Mean	Maximum
PRICE	2	342.48	590.00
COST	2	111.24	182.00
DAY	2	17.00	20.00

SEX=FEMALE

Variable	N	Mean	Maximum
PRICE	11	133.15	700.00
COST	11	51.83	210.00
DAY	12	15.50	27.00

SEX=MALE

Variable	N	Mean	Maximum
PRICE	34	119.82	700.00
COST	35	54.28	240.00
DAY	36	14.86	27.00

Note: Output has been reorganized for display (tables actually display vertically).

Figure 37: Example 4 -- Use of CLASS Statement

```
proc means data=bios511.sales2 maxdec=2 n mean max;
  class sex ;
  title1 color=black 'PROC MEANS: Summary by sex using '
         color=blue  'CLASS '
         color=black 'Statement' ;
run;
```

PROC MEANS: Summary by sex using CLASS Statement

SEX	N Obs	Variable	N	Mean	Maximum
FEMALE	12	PRICE	11	133.15	700.00
		COST	11	51.83	210.00
		DAY	12	15.50	27.00
MALE	36	PRICE	34	119.82	700.00
		COST	35	54.28	240.00
		DAY	36	14.86	27.00

Figure 38: **Example 5 -- Using the OUTPUT and By Statements**

```
proc sort data=bios511.sales2 out=sales;
  by sex;
run;

proc means data=sales noprint;
  by sex;
  where not missing(sex);
  var cost price;
  output out=summary n=cost_n price_n mean= cost_mean price_mean;
run;

proc print data=summary;
  title1 h=2 'Output SAS Data Set from PROC MEANS';
  title2 h=1.5 'with the BY Statement';
run;
```

Output SAS Data Set from PROC MEANS
with the BY Statement

Obs	SEX	_TYPE_	_FREQ_	cost_n	price_n	cost_mean	price_mean
1	FEMALE	0	12	11	11	51.8255	133.150
2	MALE	0	36	35	34	54.2814	119.819

Figure 39: Example 5 -- Using the OUTPUT, BY, and CLASS Statements (Using AUTONAME and NWAY)

```
proc means data=bios511.sales2 noprint;
  where not missing(sex);
  class dept sex;
  var cost price;
  output out=summary n= mean= / autoname;
run;

proc print data=summary;
  title1 h=2 'Output SAS Data Set from PROC MEANS';
  title2 h=1.5 'with the CLASS Statements and AUTONAME Option';
  format cost_mean price_mean 6.2;
run;
```

Output SAS Data Set from PROC MEANS
with the CLASS Statements and AUTONAME Option

Obs	DEPT	SEX	_TYPE_	_FREQ_	COST_N	PRICE_N	COST_Mean	PRICE_Mean
1			0	46	45	45	49.55	123.08
2		FEMALE	1	12	11	11	51.83	133.15
3		MALE	1	34	34	34	48.82	119.82
4	FURS		2	4	4	4	200.01	624.98
5	SHOES		2	42	41	41	34.88	74.11
6	FURS	FEMALE	3	1	1	1	210.00	700.00
7	FURS	MALE	3	3	3	3	196.67	599.97
8	SHOES	FEMALE	3	11	10	10	36.01	76.47
9	SHOES	MALE	3	31	31	31	34.51	73.35

Summary of SAS Descriptive Procedures

SORT
- Used to rearrange observations by one or more sort fields.
- Must be used to sort data before a BY statement is used with other procedures.
- Can sort data in ascending or descending order.
- Sorts data in ASCII sorting sequence.
- Can be used to check for and remove duplicate observations.

CONTENTS
- Displays descriptor part of a SAS data set.
- Can be used to get meta-information on existing or new SAS data sets.
- To see information similar to PROC CONTENTS output, view the properties of a data set in the SAS Explorer window.

PRINT
- Displays data part of a SAS data set.
- Can be used to check a data set or to produce simple listing reports.
- Another way to visually inspect the data portion of a data set is to use the ViewTable window.

FREQ
- Displays the distribution of variable values in tabular format.
- Displays the combined frequency for two or more variables.
- Displays measures of association and test statistics for two-way tables.
- More appropriate for discrete variables.
- Should be used to print out all of the possible values a discrete variable can have and to get a frequency distribution of each discrete variable.

UNIVARIATE
- Gives detailed information about the distribution of a numeric variable.
- More appropriate for continuous variables.
- Should be run on any variables to be used in more analytic procedures.
- Can be used to output statistics of interest.

MEANS
- Provides simple univariate statistics for numeric variables.
- More appropriate for continuous variables.
- More condensed output than UNIVARIATE.
- Can be used to output statistics of interest.