

SAS® Users Group International

27th Annual Meeting

Tips and Techniques for PROC MEANS

Andrew H. Karp

Sierra Information Services, Inc.

19229 Sonoma Highway PMB 264

Sonoma, California 95476 USA

707 996 7380

www.SierraInformation.com

SierraInfo@AOL.com



© Sierra Information Services, Inc. SAS is a registered trademark in the USA and other countries. ® indicates USA registration.

Tips and Techniques for PROC MEANS

- This presentation shows you ten hopefully useful tips for getting the most out of PROC MEANS
 - Many of the tips show features added to PROC MEANS in Version 8
 - This presentation will:
 - Show you ways to reduce the number of times you need to run PROC MEANS to accomplish a given task
 - Enhance the analytic value of the output the procedure produces
 - Add more information to output analytic data sets generated by PROC MEANS
 - Reduce processing “overhead” when using PROC MEANS with large data sets.

Tips and Techniques for PROC MEANS

- PROC MEANS vs. PROC SUMMARY
 - Very powerful: rapidly summarize and analyze the values of numeric variables in SAS data sets
 - Became identical procedures in Version 6
 - Create:
 - *Output SAS data sets* (default in PROC SUMMARY)
 - *Output in the SAS Output Window* (default in PROC MEANS)

3

Tips and Techniques for PROC MEANS

- A basic example of using PROC MEANS to create a SAS Data Set, then we will look at the tips and techniques.
- Task: from a customer order file data set, obtain the total (sum) dollars of sales by mail code and shipment status

4

Tips and Techniques for PROC MEANS

- Example Data Set :
 - Customer order file
 - 335,992 observations
 - Potential *Classification* Variables:
 - **Mailcode**
 - Mailing Code (Nine values)
 - **Dept_Nbr**
 - Department number for item (about 40 unique values)
 - **Segment**
 - Customer demographic segment (six values)
 - **Status**
 - 1= Shipped Immediately 2 = Deferred Shipment

5

Tips and Techniques for PROC MEANS

- Example Data Set (continued):
 - Customer order file
 - 335,992 observations
 - Potential **Analysis** Variables:
 - **ITM_QTY**
 - Number of items ordered
 - **ITM_PRICE**
 - Price (Cost) of Item

6

Tips and Techniques for PROC MEANS

- Task:
 - Obtain the sum of
 - items sold
 - item prices
 - By
 - MAILCODE
 - STATUS

7

```
title 'Changes and Enhancements to PROCs MEANS & SUMMARY';
title2 'In Version 8 of the SAS(r) System';
title3 'Simple Example of Using PROC MEANS';
```

```
Proc Means NOPRINT
```

```
data=orders.orderfile2(where=(mailcode in('A','B','C')));
```

```
CLASS Mailcode Status; ←
```

```
var itm_qty itmprice; ←
```

```
OUTPUT out = new sum=total_sold total_revenue;
```

```
RUN;
```

```
proc print data=new;
```

```
run;
```

***Note: longer variable and
data set names in V8***

8

Changes and Enhancements to PROCs MEANS & SUMMARY
In Version 8 of the SAS(r) System
Simple Example of Using PROC MEANS

Obs	mailcode	status	_TYPE_	_FREQ_	total_	total_
					sold	revenue
1		.	0	127742	159889	4250086
2		1	1	100505	125708	3340654
3		2	1	27237	34181	909432.4
4	A	.	2	48792	61385	1628543
5	B	.	2	23063	28611	763298.1
6	C	.	2	55887	69893	1858245
7	A	1	3	38373	48243	1276468
8	A	2	3	10419	13142	352074.8
9	B	1	3	18073	22430	599346.1
10	B	2	3	4990	6181	163952.0
11	C	1	3	44059	55035	1464840
12	C	2	3	11828	14858	393405.6

9

Tips and Techniques for PROC MEANS

- What Happened?
 - An output (temporary) SAS Data Set was created which contained analyses of the analysis variables at every possible combination (or “crossing”) of the classification variables
 - **_TYPE_** variable: ‘level’ of the aggregation
 - **_FREQ_** variable: number of observations in the input, or source, data set which contributed to that row/observation in the output data set
 - The output SAS Data Set was printed to the output window
 - Results shown on prior page

10

Tip #1: Use PROC MEANS to Calculate Quantile Statistics

- PROCs MEANS/SUMMARY now permits calculation of **quantile statistics**, such as the
 - Median (50th percentile), 90th percentile, interquartile range
 - Prior to V8, only PROC UNIVARIATE had this capability
 - (PROCs TABULATE and REPORT can also calculate quantile statistics in V8)
- You don't need to run both PROC UNIVARIATE and PROC MEANS and then merge the results
 - PROCs SUMMARY, MEANS, UNIVARIATE, TABULATE and REPORT all share a common set of analysis tools in Version 8.

11

Tip #1: Use PROC MEANS to Calculate Quantile Statistics

- Task:
 - Obtain the
 - Total revenue (sum)
 - Average revenue (mean)
 - Median revenue (50th percentile)
 - From the Customer Electric Rate Data Set
 - By Rate Schedule

12

Customer Electric Rate Data Set

- Approximately 16,000 observations
 - Premise-level data
 - Monthly Kwh (Kilowatt Hour) Consumption
 - Monthly Billed Revenue
 - Total Revenue
 - Quarterly Revenue
 - Rate Schedule
 - Transformer
 - Region
 - Local Office

13

Tip #1: Use PROC MEANS to Calculate Quantile Statistics

```
proc MEANS NOPRINT
data=electric.elec_V8;
class rate_schedule;
var total_revenue;
output out=new2    sum=median_REV
                   mean=total_REV
                   p50=mean_REV;

run;
proc print data=new2;
run;
```

14

Tip #1: Use PROC MEANS to Calculate Quantile Statistics

The SAS System

**What's "Wrong" With
These Results?**

Improved Analytic Capabilities: Quantile Statistics in Version 8

Obs	rate_schedule	_TYPE_	_FREQ_	median_REV	total_REV	mean_REV
1		0	15847	11326177.68	714.72	590.51
2	E1	1	14466	10227564.97	707.01	602.07
3	E1L	1	1225	622299.86	508.00	440.65
4	E1M	1	156	476312.85	3053.29	1040.97

15

Tip #2: Use the AUTONAME and AUTOLABEL Options

- What happened?
 - The desired analyses were carried out, and the data set was created, BUT the variable names are incorrect.
 - This is a common problem when creating new variables in the OUTPUT Statement in PROCs MEANS and SUMMARY
 - Solution: the **AUTONAME** option

16

Tip #2: Use the AUTONAME and AUTOLABEL Options

- These options are used in the OUTPUT Statement to have the SAS System automatically name the variables it creates while making an output SAS data set
- The statistics keyword is appended to the analysis variable's name
 - In the variable created by PROC MEANS which “hold” the desired analysis in the output data set
 - In the variable label placed in the output data set's descriptor portion

17

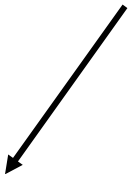
Tip #2: Use the AUTONAME and AUTOLABEL Options

```
proc MEANS NOPRINT NWAY data=electric.elec_v8;  
label total_revenue = 'Total Billed Rev. ($)';  
class rate_schedule;  
var total_revenue;  
output out=new3 sum= mean= p50= /autoname autolabel;  
run;
```

18

Tip #2: Use the AUTONAME and AUTOLABEL Options

Tips & Techniques for PROC MEANS
Using the AUTONAME and AUTOLABEL Options
Example 1: Using Variable Names as Column Headers



Obs	rate_ schedule	_TYPE_	_FREQ_	total_ revenue_Sum	total_ revenue_ Mean	total_ revenue_ P50
1	E1	1	14466	10227564.97	707.01	602.07
2	E1L	1	1225	622299.86	508.00	440.65
3	E1M	1	156	476312.85	3053.29	1040.97

19

Tip #2: Use the AUTONAME and AUTOLABEL Options

```
proc contents data=new3 NOPRINT OUT=new4(keep = name label);  
run;
```

```
proc print data=new4;  
title3 'Effect of Using the AUTOLABEL Option';  
title4 'PROC CONTENTS Excerpt Stored in SAS Data Set';  
run;
```

```
proc print data=new3 label;  
title3 'Using the PROC MEANS-Created Labels';  
title4 'As Column-Headers in PROC PRINT Output';  
run;
```

20

Tip #2: Use the AUTONAME and AUTOLABEL Options

Tips & Techniques for PROC MEANS
 Using the AUTONAME and AUTOLABEL Options
 Effect of Using the AUTOLABEL Option
 PROC CONTENTS Excerpt Stored in SAS Data Set

Obs	NAME	LABEL
1	_FREQ_	
2	_TYPE_	
3	rate_schedule	
4	total_revenue_Mean	Total Billed Rev. (\$) _Mean
5	total_revenue_P50	Total Billed Rev. (\$) _P50
6	total_revenue_Sum	Total Billed Rev. (\$) _Sum

21

Tip #2: Use the AUTONAME and AUTOLABEL Options

Tips & Techniques for PROC MEANS
 Using the AUTONAME and AUTOLABEL Options
 Using the PROC MEANS-Created Labels
 As Column-Headers in PROC PRINT Output

Obs	rate_ schedule	_TYPE_	_FREQ_	Total Billed Rev. (\$)_Sum	Total Billed Rev. (\$)_Mean	Total Billed Rev. (\$)_P50
1	E1	1	14466	10227564.97	707.01	602.07
2	E1L	1	1225	622299.86	508.00	440.65
3	E1M	1	156	476312.85	3053.29	1040.97

22

Tip #3: Create Multiple Output SAS Data Sets in One Use of PROC MEANS

- An often overlooked, but very powerful, feature in PROC MEANS is its ability to create multiple SAS data sets can be created in a single “run” of the PROC.
- This is often very useful (and more efficient) when multiple output data sets, continuing analyses at different combinations of the CLASS variables, are required.



23

Tip #3: Create Multiple Output SAS Data Sets in One Use of PROC MEANS

- A review of the `_TYPE_` variable
 - Represents the level of aggregation among the variables in the CLASS statement.
 - If there are four CLASS variables, the total number of unique combinations of the variables is 16
 - $2^N = 2^4 = 2 \times 2 \times 2 \times 2 = 16$ (zero to 15)
 - `_TYPE_` can be used to select just a few analyses from all possible combinations of the CLASS variables.
 - In Version 6 you needed to know the desired value of `_TYPE_` or create a big data set and then use the bit-testing facility in a Data Step



24

Tip #3: Create Multiple Output SAS Data Sets in One Use of PROC MEANS

- Example:

- Create a 'lightly summarized' data set from the order file. The analysis variables of interest are the sum of
 - ITM_QTY (Quantity Ordered)
 - ITMPRICE (Item Price)

```
proc means noprint data=order.orderfile2;  
class mailcode dept_nbr segment status;  
var itmprice itm_qty;  
output out=new sum=;  
run;
```

**6,723
Observations**

25

Tip #3: Create Multiple Output SAS Data Sets in One Use of PROC MEANS

- The resulting data set contains the sum of ITMPRICE and ITM_QTY at every possible combination of the four variables in the CLASS Statement

26

Tip #3: Create Multiple Output SAS Data Sets in One Use of PROC MEANS

- Out of the 15 different combinations of the CLASS variables, suppose a subset data set containing the analysis of ITM_QTY and ITMPRICE at each combination of MAILCODE and STATUS is desired
 - Version 6 approach 1: figure out the appropriate value of _TYPE_ and use it in a WHERE Clause
 - _TYPE_ = 9
 - Version 6 Approach 2: Use the bit-testing facility in a data step

27

Tip #3: Create Multiple Output SAS Data Sets in One Use of PROC MEANS

```
proc means noprint data=order.orderfile2;  
class mailcode dept_nbr segment status;  
var itmprice itm_qty;  
output out=new sum=;
```

Version 6 Approach 1

```
Output out=new2(where=(_TYPE_ = 9)) sum=;  
run;
```

Data Set 'new' contains a 'complete analysis' of the variables at all possible combinations of the classification variables, while Data Set 'new2' contains only those observations where _TYPE_ = 9

28

Tip #3: Create Multiple Output SAS Data Sets in One Use of PROC MEANS

```
data subset;  
  set new;  
  if _type_ = '1001'b;  
run;
```

What are the drawbacks to this approach?

Version 6 Approach 2

Using the Bit Testing Facility in the Data Step

29

Tip #3: Create Multiple Output SAS Data Sets in One Use of PROC MEANS


- New to Version 8 the **CHARTYPE** option simplifies creation of multiple output SAS data sets from a single use of PROC MEANS.
- This option converts the numeric values of `_TYPE_` to a character variable indicating the combination of CLASS Statement variables
 - Although this variable is composed only of zeros and ones, it is a **character variable**

30

Tip #3: Create Multiple Output SAS Data Sets in One Use of PROC MEANS

- With the CHARTYPE Option, making subset data sets with PROC MEANS in Version 8 is very simple!

```
proc means noprint data=order.orderfile2 chartype;  
class mailcode dept_nbr segment status;  
var itmprice itm_qty;  
output out=one(where=( _type_ = '1001')) sum=;  
output out=two(where=( _type_ = '1011')) sum=;  
output out=three(where=( _type_ = '0110')) sum = ;  
run;
```



31

Tip #3: Create Multiple Output SAS Data Sets in One Use of PROC MEANS

- The same value of _TYPE_ can be “re-used” when specifying the creation of multiple SAS data sets.
- Example: Add the “grand total” (observation where _TYPE_ = '0000' to each output SAS data set created in the previous example



32

Tip #3: Create Multiple Output SAS Data Sets in One Use of PROC MEANS

```
output out=one(where=
  (_type_ IN('0000','1001')) sum=;
output out=two(where=
  (_type_ IN('0000','1011')) sum=;
output out=three(where=
  (_type_ IN('0000','0110')) sum = ;
```

In this example, the “grand sum”
(_TYPE_ = '0000') is output to all three data sets

33

Tip #4: Use the DESCENDTYPES Option

- By default, observations in output data sets created by PROC MEANS are stored in ascending (lowest to highest) value of _TYPE_
- In many analytic/reporting situations, however, it would “save a PROC SORT step” if PROC MEANS would put the observation whose value of _TYPE_ is zero at the bottom of the data set.
- Using the DESCENDTYPES option, now available in SAS Version 8, does this for you.

34

Tip #4: Use the DESCENDTYPES Option

- The DESCENDTYPES option
 - Placed in the PROC MEANS Statement
 - Applies to ALL output data sets created in the OUTPUT Statements included in the task
 - Observations in the output data set are stored in descending order (highest to lowest) of the values of `_TYPE_`
 - Effect: puts the “grand total” at the “bottom” of the output data set

35

Tip #4: Use the DESCENDTYPES Option

```
proc print data=one;  
title1 'Tips for PROC MEANS';  
title2 'Data Set ONE w/DESCENDING Option';  
run;
```

36

Tip #4: Use the DESCENDTYPES Option

Tips for PROC MEANS

Data Set ONE w/DESCENDING Option

Obs	mailcode	DEPT_NBR	SEGMENT	status	_TYPE_	_FREQ_	ITMPRICE	ITM_QTY
1	A			1	1001	38373	1276468	48243
2	A			2	1001	10419	352074.8	13142
3	B			1	1001	18073	599346.1	22430
4	B			2	1001	4990	163952.0	6181
5	C			1	1001	44059	1464840	55035
6	C			2	1001	11828	393405.6	14858
7	D			1	1001	23154	773338.5	28719
8	D			2	1001	6270	208887.5	7943
9	E			1	1001	17039	563429.0	21305
10	E			2	1001	4404	148922.4	5491
11	F			1	1001	36504	1219881	45501
12	F			2	1001	9953	338758.9	12427
13	G			1	1001	16644	551935.7	20723
14	G			2	1001	4407	146974.1	5555
15	H			1	1001	21831	729721.2	27223
16	H			2	1001	5754	187638.1	7238
17	I			1	1001	49078	1637716	61109
18	I			2	1001	13212	438955.3	16382
19	.			.	0000	335992	11196244	419505



37

Tip #5: Use the TYPES Statement to limit the combinations of CLASS variables

- Another enhancement to PROC MEANS is the **TYPES** Statement (not to be confused with the variable _TYPE_)
 - This statement permits creation of specified analyses among the CLASS variables
 - If you have many class variables and only some analyses are required, this approach might be preferable

38

Tip #5: Use the TYPES Statement

- Example: Use the TYPES Statement with the catalog order file to analyze ITMPRICE and ITM_QTY at the following levels of aggregation:
 - Overall (_TYPE_ = 0)
 - Segment by Status (_TYPE_ = 3)
 - Mailcode by Segment (_TYPE_ = 10)
 - Mailcode by Dept_Nbr by Segment (_TYPE_ = 14)

39

Tip #5: Use the TYPES Statement

```
proc means noprint data=order.orderfile2 ;  
class mailcode dept_nbr segment status;  
types ()  
    segment * status  
    mailcode * segment  
    mailcode * dept_nbr * segment;  
var itmprice itm_qty;  
output out=a sum = ;  
run;
```

TYPES () requests the overall total _TYPE_ = 0

40

Tip #6: Use the DESCENDING and other ordering options in the CLASS Statement

- Using the **DESCENDING** option and multiple **CLASS** statements
 - In Version 6, values of variables placed in the CLASS statement were portrayed in “sort order”
 - The new **DESCENDING** option overrides this default
 - Multiple CLASS statements are allowed in a single use of PROC MEANS



41

Tip #7: Using the IDGROUP Option to Create Observations Containing Extreme Values from the Input Data Set

- Using the new **IDGROUP** option
 - This option allows output of new variables to a data set created by PROC MEANS.
 - Combines the ID and IDMIN options in the PROC MEANS statement and the MAXID and MINID options in the OUTPUT Statement
 - Allows identification of extreme values of the analysis variables
 - The **OUT[N]** option controls the number of extreme observations to be output as variable in the new data set.



42

Tip #7: Use the IDGROUP Option to Create Observations Containing Extreme Values from the Input Data Set

- Tasks:
- Using the Electric Consumption Data Set
 - Perform the analysis in descending order of the values of REGION
 - Obtain the mean and sum of total revenue
 - Obtain the two largest and two smallest values of total revenue
 - Output to a new SAS data set

43

```
ods html path= 'c:\' (url=none) body = 'means4.htm'
  style = styles.andrew9;
proc means
data=electric.elec_v8(
where=(transformer in('R2448Y','A4356C','B2348X','D8976V')))
  noprint nway;
class region/descending; ←
class transformer;
var total_revenue ;
output out=c(rename=(_freq_ = Customer_Count) drop=_type_)
  idgroup (max(total_revenue) out[2] (total_revenue)=maxrev)
  idgroup (min(total_revenue) out[2] (total_revenue)=minrev)
  sum= mean= /autoname; ←
run;
```

44

Tip #7: Use the IDGROUP Option to Create Observations Containing Extreme Values from the Input Data Set

```
proc print data=c split = '_';  
title 'Changes & Enhancements to PROCs MEANS &  
SUMMARY';  
title2 'in Version 8 of the SAS System';  
title3 'Using the DESCENDING and IDGROUP Options';  
run;  
ods html close;  
ods listing;
```

45

Tip #7: Use the IDGROUP Option to Create Observations Containing Extreme Values from the Input Data Set

- More on the IDGROUP statement

```
idgroup (max(total_revenue) out[2] (total_revenue)=maxrev)
```

output the two largest values of TOTAL_REVENUE at each combination of the values of the CLASS variables. The variable prefixes are MAXREV. The variable names are MAXREV_1 and MAXREV_2

```
idgroup (min(total_revenue) out[2] (total_revenue)=minrev)
```

output the two smallest values of TOTAL_REVENUE at each combination of the values of the CLASS variables. The variable prefixes are MINREV. The variable names are MINREV_1 and MINREV_2

46

<p><i>Changes & Enhancements to PROCs MEANS & SUMMARY</i> <i>in Version 8 of the SAS System</i> <i>Using the DESCENDING and IDGROUP Options</i></p>									
Obs	REGION	transformer	Customer Count	maxrev 1	maxrev 2	minrev 1	minrev 2	total revenue Sum	total revenue Mean
1	WESTERN	A4356C	493	2482.42	2482.42	54.75	54.75	337698.48	684.99
2	WESTERN	B2348X	510	2671.17	2671.17	60.00	60.00	325224.96	637.70
3	WESTERN	D8976V	207	2205.09	2205.09	60.00	60.00	153963.98	743.79
4	WESTERN	R2448Y	301	4943.85	4943.85	113.44	113.44	190438.76	632.69
5	SOUTHERN	A4356C	72	2433.38	2433.38	148.42	148.42	73709.35	1023.74
6	SOUTHERN	B2348X	52	1496.53	1496.53	228.80	228.80	36504.67	702.01
7	SOUTHERN	D8976V	12	877.01	877.01	239.06	239.06	7881.57	656.80
8	SOUTHERN	R2448Y	47	1599.85	1599.85	206.38	206.38	32027.50	681.44
9	NORTHERN	A4356C	581	5850.54	5850.54	60.00	60.00	468485.68	806.34
10	NORTHERN	B2348X	490	2722.55	2722.55	130.08	130.08	427256.15	871.95
11	NORTHERN	D8976V	237	2274.44	2274.44	198.86	198.86	189297.28	798.72
12	NORTHERN	R2448Y	332	3158.60	3158.60	232.31	232.31	258361.37	778.20
13	EASTERN	A4356C	481	3101.83	3101.83	60.00	60.00	294379.91	612.02
14	EASTERN	B2348X	432	16152.28	16152.28	50.00	50.00	342410.95	792.62
15	EASTERN	D8976V	182	2683.75	2683.75	88.59	88.59	113840.69	625.50
16	EASTERN	R2448Y	339	1883.67	1883.67	60.00	60.00	198000.63	584.07

Tip #8: Use the PRELOADFMT and COMPLETETYPES Options

- By default, a format is applied to the values of a classification variable *after* PROC MEANS has performed the desired analyses
 - If no observations in the input data set have a formatted value of the classification variable, that formatted value is not portrayed in the PROC MEANS-generated output



Tip #8: Use the PRELOADFMT and COMPLETETYPES Options

- The **PRELOADFMT** Option is placed in the CLASS statement and instructs PROC MEANS to load the format in to memory before conducting the analysis
- The **COMPLETETYPES** Option, in the PROC MEANS statement, results in having all the formatted values of the classification variable portrayed, even if there are no observations in the input data set have that formatted value.



49

Tip #8: Use the PRELOADFMT and COMPLETETYPES Options

- Example:
 - From the Customer Order File
 - Analyze sales by Demographic Segment (SEGMENT)
 - First, a format is applied to the one-byte character variable to 'alter the external representation of the value of the variable'



50

```
proc format
library = orders;
value $segfmt
'A' = '1) Paycheck to Paycheck (A)'
'B' = '2) Rural Retirees (B)'
'C' = '3) Suburban Strivers (C)'
'I' = '4) Soccer Moms & Dads (I)'
'J' = '5) Young Families (J)'
'L' = '6) Urban Pioneers (L)'
'N' = '7) Stable Golden Years (N)'
'O' = '8) Solidly Suburban (O)'
'R' = '9) Dot Com Dandies (R)';
RUN;
```

51

Tip #8: Use the PRELOADFMT and COMPLETETYPES Options

- Now that this format is available, we can use it to improve the visual appeal of a data set created by PROC MEANS which analyzed the customer order file
 - Tasks:
 - Obtain the average number of items ordered
 - Obtain the mean price of items ordered
 - By SEGMENT

52

Tip #8: Use the PRELOADFMT and COMPLETETYPES Options

```
options fmtsearch = (orders) nocenter nodate;  
ods html path= 'c:\' (url=none)  
body = 'means4.htm' style = styles.andrew8;  
  
proc means noprint data=orders.orderfile2;  
format segment $segfmt.;  
  
class segment/PRELOADFMT;  
  
var itmprice itm_qty;  
  
output out=x mean = /autoname;  
  
run;
```

53

Tip #8: Use the PRELOADFMT and COMPLETETYPES Options

Obs	SEGMENT	_TYPE_	_FREQ_	ITMPRICE_Mean	ITM_QTY_Mean
1		0	335992	\$33.3229	1.24856
2	3) Suburban Strivers (C)	1	268704	\$32.6979	1.24544
3	4) Soccer Moms & Dads (I)	1	10592	\$35.3829	1.23688
4	6) Urban Pioneers (L)	1	22250	\$29.2650	1.20548
5	7) Stable Golden Years (N)	1	32828	\$39.8366	1.23626
6	8) Solidly Suburban (O)	1	1194	\$40.0112	3.17588
7	9) Dot Com Dandies (R)	1	424	\$67.7493	1.29953

Where are the observations for customers in
Segments 1, 2 and 5?

54

Tip #8: Use the PRELOADFMT and COMPLETETYPES Options

- Although there were nine values in the format, only six values of SEGMENT were portrayed in the preceding output
 - That's because no customer records exist in the data set for customers in those three demographic categories.
- By default, PROC MEANS ordered the observations in the output data set using the formatted value of the classification variable.



55

Tip #8: Use the PRELOADFMT and COMPLETETYPES Options

- Using the **PRELOADFMT** and **COMPLETETYPES** options generates a data set with ALL formatted values of the variable.
- Also, we can re-arrange the ordering of the CLASS variables using the **ORDER=** option.



56

Tip #8: Use the PRELOADFMT and COMPLETETYPES Options

- In the next example, the **PRELOADFMT** and **COMPLETETYPES** Options are used to generate a data set that includes values of the classification variable SEGMENT for which there are no observations in the analysis data set
- The **ORDER=FREQ** option in the **CLASS** statement instructs PROC MEANS to order the observations from highest to lowest frequency in the output data set it creates.

57

Tip #8: Use the PRELOADFMT and COMPLETETYPES Options

```
proc means noprint data=orders.orderfile2
      COMPLETETYPES;
format segment $segfmt.;
class segment/order=freq PRELOADFMT;
var itmprice itm_qty;
output out=x mean = /autoname;
run;
```

58

Tip #8: Use the PRELOADFMT and COMPLETETYPES Options

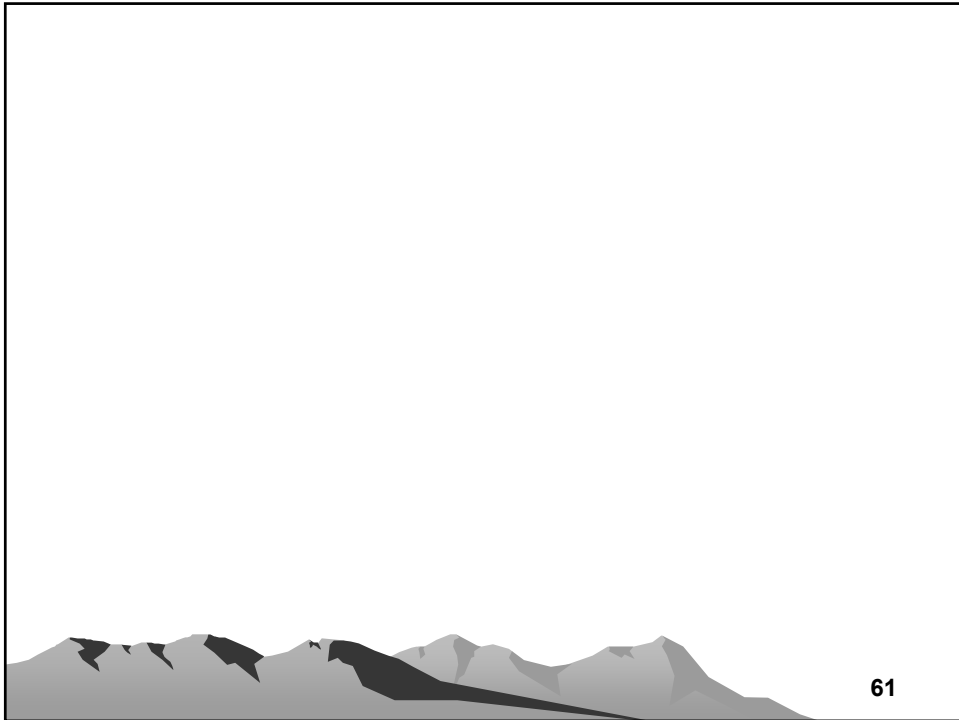
Obs	SEGMENT	_TYPE_	_FREQ_	ITMPRICE_Mean	ITM_QTY_Mean
1		0	335992	\$33.3229	1.24856
2	3) Suburban Strivers (C)	1	268704	\$32.6979	1.24544
3	7) Stable Golden Years (N)	1	32828	\$39.8366	1.23626
4	6) Urban Pioneers (L)	1	22250	\$29.2650	1.20548
5	4) Soccer Moms & Dads (I)	1	10592	\$35.3829	1.23688
6	8) Solidly Suburban (O)	1	1194	\$40.0112	3.17588
7	9) Dot Com Dandies (R)	1	424	\$67.7493	1.29953
8	1) Paycheck to Paycheck (A)	1	0	.	.
9	2) Rural Retirees (B)	1	0	.	.
10	5) Young Families (J)	1	0	.	.

59

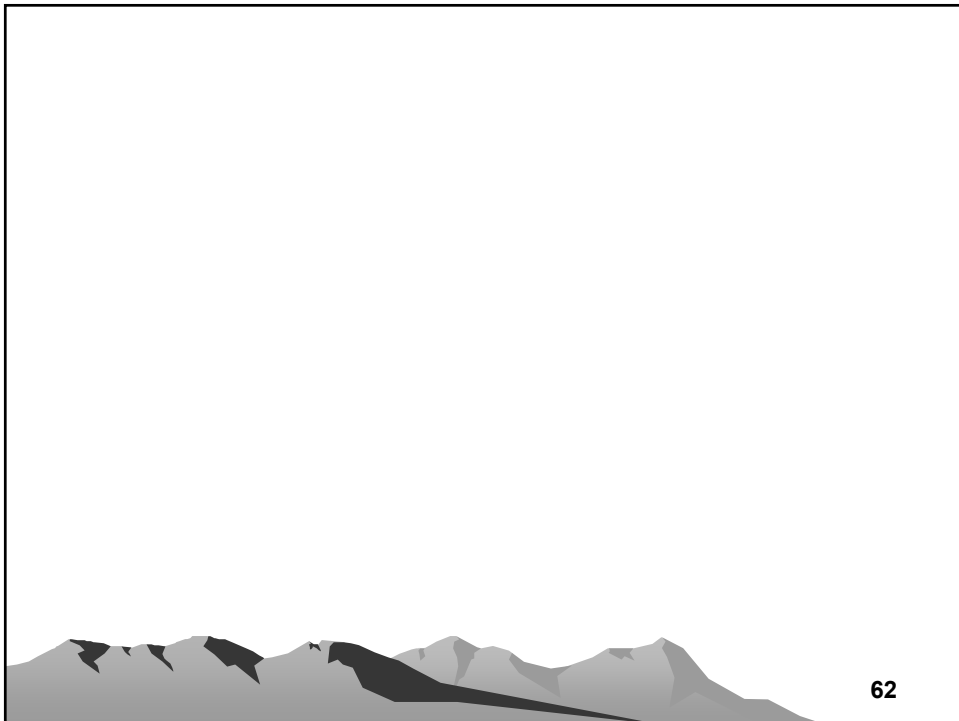
Tip #9: Use the WAYS Statement

- The WAYS Statement can save you
 - Coding time
 - Processing time
 - When you want PROC MEANS to generate analyses at certain combinations of the classification variables
 - WAYS 3; Requests all three-way combinations
 - WAYS 2 3; Requests all two and all three way combinations

60



61



62

Tip #10: Use Multilabel Formats

- A new feature available in PROC FORMAT allows creation of multilabel formats.
 - MLFs are used by PROCs MEANS/SUMMARY, REPORT and TABULATE
- The next example shows how a multilabel format can be used in PROC MEANS
- The MLF option in the CLASS statement instructs PROC MEANS to create subgroup combinations when a multilabel format is assigned to a classification variable.

63

```
proc format library = orders;  
value totalf (multilabel)
```

```
1 - 1000 = '1 to 1,000'  
1001 - 2000 = '1,001 to 2,000'  
2001 - 3000 = '2,001 to 3,000'  
3001 - high = '3,001 or more'
```

*Primary
Format
Label*

```
1000 - 1500 = 'a) 1,000 to 1,500'  
1501 - 2000 = 'b) 1,501 to 2,000'  
2001 - 2500 = 'c) 2,001 to 2,500';
```

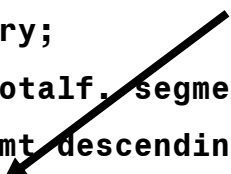
*Secondary
Format Label*

```
run;
```

64


```
ods html path= 'c:\' (url=None) body ='means4.htm'  
style = styles.andrew8;
```

```
proc means noprint completetypes NWAY  
data=orders.orderhistory;  
format tot_sales_amt totalf. segment $segfmt.;  
class segment/preloadfmt descending;  
class tot_sales_amt/MLF preloadfmt  
order=formatted;  
var tot_sales_amt;  
output out=new(drop=_type_ _freq_)  
                                    n = Customer_Count;  
run;
```



65



```
proc print data=new noobs;  
title1 'Changes and Enhancements to PROCs  
MEANS & SUMMARY';  
title2 'in Version 8 of the SAS System';  
title3 'Using MULTILABEL Formats';  
  
ods html close;  
ods listing;
```

66



Tip #10: Use Multilabel Formats

SEGMENT	tot_sales_amt	Customer_Count
9) Dot Com Dandies (R)	1 to 1,000	8
9) Dot Com Dandies (R)	1,001 to 2,000	2
9) Dot Com Dandies (R)	2,001 to 3,000	2
9) Dot Com Dandies (R)	3,001 or more	3
9) Dot Com Dandies (R)	a) 1,000 to 1,500	1
9) Dot Com Dandies (R)	b) 1,501 to 2,000	1
9) Dot Com Dandies (R)	c) 2,001 to 2,500	1
8) Solidly Suburban (O)	1 to 1,000	37
8) Solidly Suburban (O)	1,001 to 2,000	8
8) Solidly Suburban (O)	2,001 to 3,000	2
8) Solidly Suburban (O)	3,001 or more	2
8) Solidly Suburban (O)	a) 1,000 to 1,500	7
8) Solidly Suburban (O)	b) 1,501 to 2,000	1
8) Solidly Suburban (O)	c) 2,001 to 2,500	2

67

Learning More...

www.forecon.com

68

Thanks for Inviting Me
to Portland!

