



# *Tips & Techinques for Large Files*

**Andrew H. Karp**  
*Sierra Information Services*  
Sonoma, California

[www.sierrainformation.com](http://www.sierrainformation.com)  
[andrew@sierrainformation.com](mailto:andrew@sierrainformation.com)

Note: SAS is a registered trademark of SAS Institute in the USA and other countries. ® indicates USA registration. This document copyright © 2010 Sierra Information Services. All rights reserved.

1



## **Tips & Techniques for Large Files**

- Suggestions for Working Efficiently With Large
  - Flat Files
  - SAS Data Sets
- Goals:
  - Identify approaches to reduce
    - Processing Time
    - Programming Steps
      - Required to obtain desired results

2



## Copies of this Presentation

---

- Available in PDF
- Download from
  - [www.SierraInformation.com](http://www.SierraInformation.com)
  - Click on "Free Downloads" link on the home page
- Please hold your question(s) until the end of the talk, thanks!

3



## One "Core" Strategy for Working With Large Files

---

- Test your code on an extract from the large file, not on the large file itself
  - Reduces costs to run many programs
  - Reduces time required to "see the results"
  - Eliminates the need for
    - repeated tape mounts
    - excessive I/O and CPU resource utilization
    - scheduling the jobs to run in an "overnight" or other "slow queue."

4



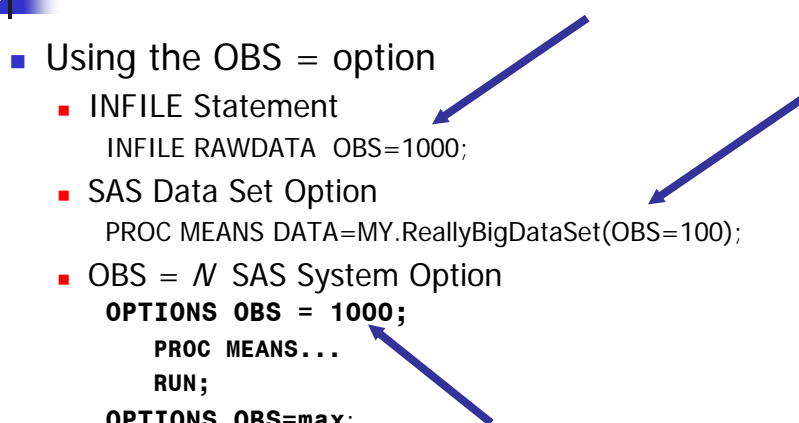
## SAS System Tools for Obtaining an Extract

- Two potential ways to obtain an extract from a large file are:
  - select the first  $n$  records from an existing data set or flat file
  - choose a random sample from an existing data set
    - depending on your needs, a stratified, or other more complex random sample, could be selected

5



## SAS System Tools for Obtaining an Extract

- Using the OBS = option
    - INFILE Statement  
`INFILE RAWDATA OBS=1000;`
    - SAS Data Set Option  
`PROC MEANS DATA=MY.ReallyBigDataSet(OBS=100);`
    - OBS =  $N$  SAS System Option  
`OPTIONS OBS = 1000;`  
`PROC MEANS...`  
`RUN;`  
`OPTIONS OBS=max;`
- 

6



## SAS System Tools for Obtaining an Extract

---

- Selecting a random sample
  - Several SAS Programming Language *functions* generate random numbers.
  - Example: The UNIFORM Function
    - Generates a uniform random number between zero and one
    - Seed argument "initializes the random number stream"

7



## Obtaining a Simple Random Sample

---

- The UNIFORM(*seed*) function is used in a DATA STEP to obtain a "pseudo random sample" of some desired proportion of observations in a SAS data set.
- The *seed* value is used to initialize the random number stream by the UNIFORM function.
  - The value of *seed* must be either zero or an integer. If you use zero, you get a different random sample each time the program runs.


8

## Obtaining a Simple Random Sample: Example 1

Take approximately ten per cent of the observations at random on a mainframe tape data set and write them to DASD (hard drives attached to a mainframe computer)

```
//DATA DD DSN=ANDREW.MY.BIGTAPE.DATASET,DISP=SHR,UNIT=TAPE
//DISK DD DSN=ANDREW.MY.DISK.FILES,DISP=OLD
```

```
DATA DISK.SAMPLE;
  SET DATA.TAPE;
  IF UNIFORM(94115) LE .10;
RUN;
```



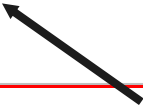
9

## Obtaining a Simple Random Sample: Example 2

- Select approximately 5 percent of the cases at random from a state social services agency payments PC SAS permanent SAS data set.

```
5 data payments_sample;
6   set sasclass.social_services_payments;
7   if uniform(999) < .05;
8   run;
```

```
NOTE: There were 376544 observations read from the data set SASCLASS.SOCIAL_SERVICES_PAYMENTS.
NOTE: The data set WORK.PAYMENTS_SAMPLE has 18713 observations and 19 variables.
NOTE: DATA statement used (Total process time):
      real time           1.32 seconds
      cpu time            0.48 seconds
```



10

## Obtaining a *Stratified* Random Sample

- PROC SURVEYSELECT
  - Added to SAS/STAT in Version 8
  - Many tools and capabilities
  - Creates SAS data sets
- Example: create a stratified random sample comprised of 50 cases at random within each value of single\_race in the social services payments data set.

11

## Obtaining a *Stratified* Random Sample

```
11  
12=proc freq data=sasclass.social_services_payments;  
13 tables single_race;  
14 title 'Tips & Techniques for Large Files';  
15 title2 'Frequency of Single_Race in State Social Services Payments File';  
16 run;
```

| single_race | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
|-------------|-----------|---------|----------------------|--------------------|
| A           | 18614     | 5.33    | 18614                | 5.33               |
| B           | 24661     | 7.07    | 43275                | 12.40              |
| I           | 11410     | 3.27    | 54685                | 15.67              |
| O           | 54732     | 15.65   | 109417               | 31.32              |
| W           | 239567    | 68.65   | 348984               | 100.00             |

Frequency Missing = 27560

12



## Obtaining a *Stratified* Random Sample

```
17
18 proc sort data=sasclass.social_services_payments out=payments;
19 by single_race;
20 run;
21
22 proc surveyselect seed=999 out=payments_sample_by_race
23 n=50 method=srs data=payments;
24 strata single_race;
25 run;
26
27 proc freq data=payments_sample_by_race;
28 tables single_race;
29 title2 'Frequency of Single_Race in Stratified Random Sample Created by';
30 title3 'PROC DATASETS';
31 run;
```

Note: Data set must be sorted by the values of the variables supplied in the STRATA Statement.

13



## Obtaining a *Stratified* Random Sample

```
Tips & Techniques for Large Files
Frequency of Single_Race in State Social Services Payments File

The SURVEYSELECT Procedure

Selection Method      Simple Random Sampling
Strata Variable       single_race

Input Data Set                PAYMENTS
Random Number Seed           999
Stratum Sample Size          50
Number of Strata              6
Total Sample Size            300
Output Data Set              PAYMENTS_SAMPLE_BY_RACE
```

PROC SURVEYSELECT generated output.

14

## Obtaining a Stratified Random Sample

Tips & Techniques for Large Files  
Frequency of Single\_Race in Stratified Random Sample Created by  
PROC DATASETS

The FREQ Procedure

| single_<br>race | Frequency | Percent | Cumulative<br>Frequency | Cumulative<br>Percent |
|-----------------|-----------|---------|-------------------------|-----------------------|
| A               | 50        | 20.00   | 50                      | 20.00                 |
| B               | 50        | 20.00   | 100                     | 40.00                 |
| I               | 50        | 20.00   | 150                     | 60.00                 |
| O               | 50        | 20.00   | 200                     | 80.00                 |
| W               | 50        | 20.00   | 250                     | 100.00                |

Frequency Missing = 50 ←

Notice that "missing" is treated as a separate "stratum" by PROC SURVEYSELECT when generating the desired stratified random sample

15

## Using the Random Sample

- Once the appropriate random sample has been selected and "written off" to storage (most likely on a hard drive), write and test your code against it, rather than the entire data set
- What are some other strategies to follow now that a sample of the large file is available for use?

16

## Programming Strategies and SAS System Software Features



- Here are some
    - programming strategies
    - SAS System tools/features
- you may want to consider when working on large data files with SAS System software.

17

## Programming Strategies

- Important core concepts to remember:
  - **Data Steps** Create Data Sets
  - **Procedures** Operate on Data Sets
    - and sometimes create new Data Sets
  - **Data Sets** are composed of
    - descriptor portion
    - data portion
      - Variables (columns)
      - Observations (rows)

18

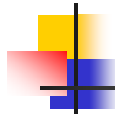


## Programming Strategies

---

- Core concepts, continued:
  - If you are working on *observations*, use a **Data Step**
  - If you are working on *variables*, use a **Procedure**
  - In many situations using a Procedure will be easier than duplicating the functionality of the Procedure by writing your own Data Step
    - example: PROC MEANS, PROC REPORT
  - Understand the descriptor portion of a SAS data set

19



## Programming Strategies

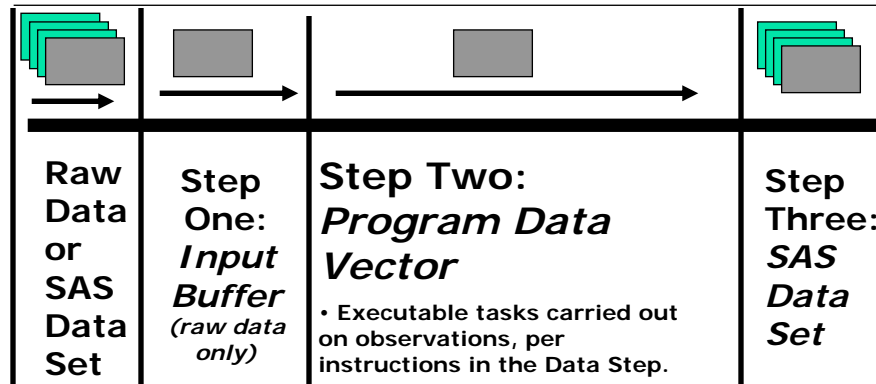
---

- The SAS System creates a Data Set, one observation at a time, from either raw data or from an existing SAS data set
- The SAS System executes statements in the *exact order* they are encountered in your program
- **Maximize** the number of *operations* performed on the **minimum** number of *observations* required to perform the desired tasks

20

## How the SAS System Creates a SAS Data Set

### Overview of the Data Step Execution Phase



21

## Read Raw Data Selectively

- When creating SAS data sets from a flat files, test the values of some of the variables before writing ALL variables in to the PDV
  - Useful when creating a SAS data set from a subset of records in a raw data file

22

## Reading Raw Data Selectively

```
DATA PRIMEPAT(DROP=CNT1-CNT4);
  INFILE PRIMEPAT;
  INPUT @008 PPAU1 $7.
        @050 HHTYPE $1.
        @139 CNT1 PD4.
        @150 CNT2 PD4.
        @161 CNT3 PD4.
        @172 CNT4 PD4.
        @725 TOTDEP PD7.2
        @723 MVCODE 2.;

  NUMTRANS=SUM(OF CNT1-CNT4);
  IF SUBSTR(PPAU1,3,1) IN ('A','T')
  THEN PPAU1='WF0' || SUBSTR(PPAU1,4,4);
  IF HHTYPE = 'S';
  RUN;
```

Approach 1

23

## Reading Raw Data Selectively

Approach 2

```
DATA PRIMEPAT(DROP=CNT1-CNT4);
  INFILE PRIMEPAT;
  INPUT @008 PPAU1 $7.
        @050 HHTYPE $1.
        @139 CNT1 PD4.
        @150 CNT2 PD4.
        @161 CNT3 PD4.
        @172 CNT4 PD4.
        @725 TOTDEP PD7.2
        @723 MVCODE 2.;

  IF HHTYPE = 'S';
  NUMTRANS=SUM(OF CNT1-CNT4);
  IF SUBSTR(PPAU1,3,1) IN ('A','T')
  THEN PPAU1='WF0' || SUBSTR(PPAU1,4,4);
  RUN;
```

24



## Reading Raw Data Selectively

```

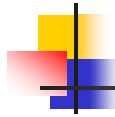
DATA PRIMEPAT(DROP=CNT1-CNT4);
  INFILE PRIMEPAT;
  INPUT @050 HHTYPE $1. @;
  IF HHTYPE NE 'S' THEN DELETE;
  ELSE
  INPUT @008 PPAU1 $7.
        @139 CNT1 PD4.
        @150 CNT2 PD4.
        @161 CNT3 PD4.
        @172 CNT4 PD4.
        @725 TOTDEP PD7.2
        @723 MVCODE 2.;
  NUMTRANS=SUM(OF CNT1-CNT4);
  IF SUBSTR(PPAU1,3,1) IN ('A','T')
  THEN PPAU1='WF0' || SUBSTR(PPAU1,4,4);
  DROP MVCODE ;
  RUN;

```



Approach 3

25



## Reading Raw Data Selectively

| <i>Metric</i> | <i>Approach</i> | <i>Mean of Five Trials</i> | <i>Max</i> | <i>Min</i> | <i>Range</i> |
|---------------|-----------------|----------------------------|------------|------------|--------------|
| CPU           | 1               | <b>59.23</b>               | 61.21      | 58.50      | 2.71         |
| CPU           | 2               | <b>47.56</b>               | 49.43      | 46.83      | 2.60         |
| CPU           | 3               | <b>22.26</b>               | 23.15      | 21.73      | 1.42         |

26



## Programming Strategies

- Avoid the Data Step wherever possible.
  - Avoid Unnecessary Data Steps
    - Remember: Maximize *Operations* and Minimize *Observations*
  - Use Procedures Instead of Data Steps
  - Use PROC DATASETS to manage the descriptor portion of an existing data set, or to manage libraries of SAS data sets.
- Avoid "extra" PROC SORT Steps

27



## Programming Strategies: Examples

Create  
multiple data  
sets in one  
Data Step

```
2
3 data eastern;
4   set sasclass.electricity;
5   if region = 'EASTERN';
6   run;
7
8 data western;
9   set sasclass.electricity;
10  if region = 'WESTERN';
11  run;
12
13 data eastern western;
14  set sasclass.electricity;
15  if region = 'EASTERN' then output eastern;
16  else
17  if region = 'WESTERN' then output western;
18  run;
```

28

## Programming Strategies: Examples

```
20 data eastern western;  
21 set sasclass.electricity(where=(region in('EASTERN','WESTERN')));  
22 if region = 'EASTERN' then output eastern;  
23 else if  
24     region = 'WESTERN' then output western;  
25 run;
```

*Use the WHERE Clause Data Set Option to limit entry in to the PDV from the source data set just the observations you need to complete the task.*

29

## Programming Strategies: Examples

- The SAS System executes statements in the exact order they are written in the Data Step

```
27 data low_income;  
28 set sasclass.electricity;  
29 total_rev = sum(of rev1-rev12);  
30 mean_rev = mean(of rev1-rev12);  
31 total_kwh = sum(of kwh1-kwh12);  
32 mean_kwh = mean(of kwh1-kwh12);  
33 * low income rates have the letter L in the second position;  
34 if substr(rate_schedule,2,1) = 'L';  
35 run;
```

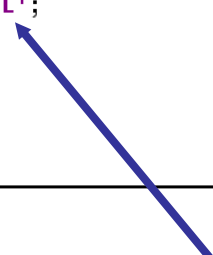
30



## Programming Strategies: Examples

- Careful placement of the “Subsetting IF” results in reduced execution time.

```
37 data low_income;
38   set sasclass.electricity;
39   * low income rates have the letter L in the second position;
40   if substr(rate_schedule,2,1) = 'L';
41   total_rev = sum(of rev1-rev12);
42   mean_rev = mean(of rev1-rev12);
43   total_kwh = sum(of kwh1-kwh12);
44   mean_kwh = mean(of kwh1-kwh12);
45 run;
```



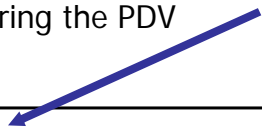
31



## Programming Strategies: Examples

- Use the WHERE Clause Data Step Option to Limit the Number of Observations Entering the PDV

```
47 data low_income;
48   set sasclass.electricity(where=(substr(rate_schedule,2,1) = 'L'));
49   total_rev = sum(of rev1-rev12);
50   mean_rev = mean(of rev1-rev12);
51   total_kwh = sum(of kwh1-kwh12);
52   mean_kwh = mean(of kwh1-kwh12);
53 run;
```



32



## Programming Strategies: Examples

- Avoid the Data Step whenever possible

*HARD WAY:*

```
3 DATA HOSPITAL.DISCHG;  
4     SET HOSPITAL.DISCHG;  
5     FORMAT DISDATE MMDDYY10.;  
6     LABEL DISDATE = 'DISCHARGE DATE';  
7     RUN;  
~
```

*In this example, the data set is recreated when the only thing we want to do is modify the descriptor portion of the data set.*

33



## Programming Strategies: Examples

- Avoid the Data Step whenever possible

*EASY WAY*

```
8  
9 PROC DATASETS LIBRARY = HOSPITAL;  
10 MODIFY DISCHG;  
11 FORMAT DISDATE MMDDYY8.;  
12 LABEL DISDATE = 'DISCHARGE DATE';  
13 QUIT;
```

*PROC DATASETS provides many features to manage libraries of SAS data sets. The MODIFY Statement is used to modify the descriptor portion of an existing SAS data set without having to run its observations through the PDV.*

34



## Programming Strategies: Examples


---

- Avoid the Data Step whenever possible

*HARD WAY*

- \* append today's file to an historical data set;

```
15 DATA HOSPITAL.DISCHG;  
16 SET HOSPITAL.DISCHG HOSPITAL.TODAY;  
17 RUN;
```



*Implied concatenation in a Data Step*

35



## Programming Strategies: Examples

---

- Avoid the Data Step whenever possible

*EASY WAY*

- \* append today's file to an historical data set;

```
PROC DATASETS LIBRARY = HOSPITAL;  
APPEND BASE = DISCHG DATA=TODAY;  
RUN;  
QUIT;
```

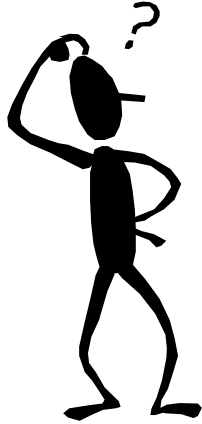
PROC APPEND and the APPEND Statement in PROC DATASETS are identical approaches to the same task.

36



## Avoid Confusion !

---



- Use consistent naming conventions
- Adopt common prefixes for groups of variables
- Sequentially number related groups of variables

37



## Avoid Confusion !

---

- Example: orders placed at a mail order catalog firm: up to four orders stored on a single customer record.
- HARD WAY: **FI\_ORDER,S\_ORDER,T\_ORDER,FO\_ORDER**
- EASY WAY:  
**ORDER1, ORDER2, ORDER3, ORDER4**

38



## Some Storage Hints



- Don't put unnecessary variables in the output data set
- Compress observations (maybe)
- Reduce the byte length of variables

39



## Some Storage Hints

- Limit the number of variables in the Data Set to those which you really need in subsequent steps in the process
  - Use KEEP and/or DROP Data Set *options* on existing Data Sets referenced in the SET Statement to reduce the number of variables in the Program Data Vector
  - Use KEEP and/or DROP Data Set Options on the new data set you are creating to avoid having variables you don't need output to the new file.

40



## Example:Using KEEP and DROP Statements

- Sum the Kilowatt Hour (KwH) consumption for 12 months by region, for only the Eastern and Southern Regions of an electrical utility.
- The only variables required at the end of the analysis are: REGION and SUM of the 12 monthly values.
  - Over 100 variables in the data set.

41



## Example:Using KEEP and DROP Statements

*HARD (and wrong) WAY:*

```
DATA A;  
  SET SASCLASS.ELECTRICITY;  
  TOTKWH = KWH1 + KWH2 + KWH3 + KWH4 + KWH5+ KWH6 +  
    KWH7 + KWH8 + KWH9 + KWH10 + KWH11 + KWH12;  
  IF REGION IN('SOUTHERN', 'EASTERN');  
  RUN;  
  
PROC MEANS DATA=A SUM;  
  CLASS REGION;  
  VAR TOTKWH;  
  TITLE 'Electricity Consumption Report';  
  RUN;
```

42

## Example: Using KEEP and DROP Statements

*EASY (and correct) WAY:*

```
DATA A(KEEP= REGION TOTKWH);  
SET SASCLASS.ELECTRICITY(WHERE=(REGION  
    IN('EASTERN','SOUTHERN'))  
    KEEP = REGION KWH1-KWH12);  
TOTKWH = SUM(OF KWH1-KWH12);  
RUN;
```

```
PROC MEANS DATA=A SUM;  
CLASS REGION;  
VAR TOTKWH;  
TITLE 'Electricity Consumption';  
RUN;
```

*Why is the VAR  
Statement not  
strictly required?*

43

## Compress Observations

- The COMPRESS Data Set option *may* reduce the size of SAS data sets.
  - Removes blank spaces and repeating values, which MAY reduce storage requirements
  - In V8, SAS can compress both character (default) and numeric (optional) variables
  - Can increase the data set size if used on data sets (less likely in V8, but it can still occur).

44



## Compress Observations

- Compress Data Set option, continued
  - additional CPU resources required for the SAS System to “uncompress” the data set before it is read by a Data Step or used by a PROC
  - SAS tasks applied to compressed data sets take longer to process.
  - **Suggestion:** use sparingly

45



## Reduce Variable Byte Length

- Probably the most common and effective way to reduce the storage requirements of a SAS Data Set
  - LENGTH statement in a Data Step
- Very useful for numeric variables that have small values and for character variables with just a few letters or symbols.
- Reducing byte length increases the number of variables stored per page of memory, reducing the size of the data set and the number of input/output operations required to move it in to and out of the CPU (central processing unit)

46

## Reduce the Byte Length of Observations

- Very effective way to reduce the size of your data sets IF.....
  - You know the System-dependent minimum byte length for numeric variables
    - In SAS for Windows, it is 3 bytes
    - In SAS running under z/OS (IBM Mainframe), it is 2 bytes
  - You know the largest possible value of the variables for which the LENGTHs will be changed

47

## Integers and Byte Lengths in V8 and SAS 9

*Largest Integer Represented Exactly in a Given Byte Length*

| Byte Length | Windows And UNIX      | Z/OS (Mainframe)       |
|-------------|-----------------------|------------------------|
| 2           | N/A                   | 256                    |
| 3           | 8,192                 | 65,536                 |
| 4           | 2,097,152             | 16,777,216             |
| 5           | 536,870,912           | 4,294,967,296          |
| 6           | 137,438,953,472       | 1,099,511,627,776      |
| 7           | 35,184,372,088,832    | 281,474,946,710,656    |
| 8           | 9,007,199,254,740,992 | 72,057,594,037,927,936 |



## Using an INDEX with a SAS Data Set

---

- Creating an INDEX for a SAS Data Set is suitable when:
  - the data set does not change often
    - is not updated frequently
    - does not have records added/removed regularly
  - the values of the variable(s) for which the INDEX will be applied discriminate among the observations
  - the data set is frequently analyzed using selection or sort variables

49



## Using an INDEX with a SAS Data Set

---

- An INDEX *may*:
  - reduce the time required to find observations meeting conditions in a WHERE clause
  - eliminate the need to sort a Data Set upon which BY group processing will be applied
- Drawbacks:
  - Additional CPU utilization and storage are required to build/store the “index portion”
  - The INDEX will need updating if the values of the indexed variables change

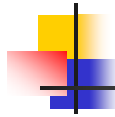
50



## Be Efficient With PROCS

- Use PROCS instead of the Data Step
  - **Example 1:** Avoid a Subset Data Set with the WHERE Clause SAS Data Set Option
  - **Example 2:** Use PROC RANK to select observations from a data set
  - **Example 3:** Create multiple data sets in a single use of PROC MEANS, and use the V8 CHARTYPE Option
  - **Example 4:** Use FORMATS instead of creating “recodes” in a Data Step

51



## Avoid a Subset SAS Data Set with the WHERE Clause Data Set Option

- **Example 1:**
  - From the Electricity Data Set, create a frequency table of the values of TRANS (Transformer) for low-income customers in the EASTERN Region
  - **Hard Way:** Create a Subset Data Set and then apply PROC FREQ to it
  - **Easy Way:** Use the WHERE Clause Data Set Option

52

## Avoid a Subset SAS Data Set with the WHERE Clause Data Set Option

```
34 data hardware;
35   set sasclass.electricity;
36   if region = 'EASTERN' and substr(CESCHED,2,1) = 'L';
37   run;
38
39 proc freq data=hardware;
40   tables trans;
41   title2 'Frequency of Transformer Serving Low Income Customers';
42   title3 'in the Eastern Region: the HARD Way';
43   run;
44
45
46 proc freq data=sasclass.electricity where=(region = 'EASTERN' and
47   substr(cesched,2,1) = 'L')
48   keep=region cesched trans;
49   title2 'in the Eastern Region: the EASY Way';
50   run;
```

53

## Use PROC RANK to Obtain Subsets of SAS Data Sets

- Example 2: Using PROC RANK to select a subset of observations
- PROC RANK
  - Collapses, or categorizes the values of numeric variables in a SAS data set
  - Creates new SAS data sets (no output to the Output Window)
    - new variables created by PROC RANK indicate observation "membership" in the ranking, or grouping variable
  - GROUPS=  $N$ , where  $N$  = number of groups to create

54



## Use PROC RANK to Obtain Subsets of SAS Data Sets

---

- PROC RANK, continued
  - By default, the procedure ranks all numeric variables in the source data set
  - The same grouping/ranking is applied to all numeric variables on which the procedure is applied.
  - GROUPS = 2           *above/below median*
  - GROUPS = 4           *quartiles*
  - GROUPS = 5           *quintiles*
  - GROUPS = 100       *percentiles*

55



## Use PROC RANK to Obtain Subsets of SAS Data Sets

---

- Task: From an electrical utility revenue data set, obtain the “top five percent” of customers based on the annual revenue.
  - Approach 1: Use PROC UNIVARIATE or PROC MEANS to obtain the value of annual revenue at the 95th percentile, then use that value in a Data Step.
  - Approach 2: Use **PROC RANK**

56

## Use PROC RANK to Obtain Subsets of SAS Data Sets

```

69 data totrev;
70   set sasclass.electricity(keep=premise rev1-rev12);
71   totrev=round(sum(of rev1-rev12));
72   run;
73
74 proc means noprint data=totrev;
75   var totrev;
76   output out=pct95  p95 = p95;
77   run;
78
79 proc print data=pct95;
80   title2 'Value of Totrev at the 95th Percentile';
81   run;
82
83 * pass value of p95 in to the macro symbol table;
84 data _null_;
85   set pct95;
86   call symputx('p95',p95); * note: SAS 9 call symputx left-justifies value
87                               of variable in the Macro Symbol Table;
88   run;
89
90 options symbolgen; * display resolved values of macro variables in SASLOG;
91 data big_spenders1;
92   set totrev(when=(totrev ge &p95));
93   run;

```

Tips & Techniques for Large Files  
Value of Totrev at the 95th Percentile

| Obs | _TYPE_ | _FREQ_ | p95  |
|-----|--------|--------|------|
| 1   | 0      | 16381  | 1668 |

## Use PROC RANK to Obtain Subsets of SAS Data Sets

```

20
27 options symbolgen; * display resolved values of macro variables in SASLOG;
28 data big_spenders1;
29   set totrev(when=(totrev ge &p95));
30   run;

```

SYMBOLGEN: Macro variable P95 resolves to 1668

```

NOTE: There were 822 observations read from the data set WORK.TOTREV.
      WHERE totrev >= 1668;
NOTE: The data set WORK.BIG_SPENDERS1 has 822 observations and 14 variables.
NOTE: DATA statement used (Total process time):
      real time          0.12 seconds
      cpu time           0.04 seconds

```

58

## Use PROC RANK to Obtain Subsets of SAS Data Sets

```

110 * Proof of Results: Data Step vs. PROC RANK;
111 data prove_the_boss_wrong;
112 merge big_spenders1(in=a) big_spenders2(in=b);
113 by premise;
114 * if both approaches generated the same list of customers there will be zero (0) obs in this data set;
115 if a and b then delete;
116 run;
117

```

```

98 * Proof of Results: Data Step vs. PROC RANK;
99 data prove_the_boss_wrong;
100 merge big_spenders1(in=a) big_spenders2(in=b);
101 by premise;
102 * if both approaches generated the same list of customers there will be zero (0) obs in this
102! data set;
103 if a and b then delete;
104 run;

NOTE: There were 822 observations read from the data set WORK.BIG_SPENDERS1.
NOTE: There were 822 observations read from the data set WORK.BIG_SPENDERS2.
NOTE: The data set WORK.PROVE_THE_BOSS_WRONG has 0 observations and 15 variables.
NOTE: DATA statement used (Total process time):
      real time           0.70 seconds
      cpu time            0.07 seconds

```

59

## Use PROC RANK to Obtain Subsets of SAS Data Sets

```

95 proc rank data=totrev groups=100
96 out=big_spenders2(where=(percentile GE 95));
97 ranks percentile; * name of variable assigning percentile to each obs in totrev;
98 var totrev; * analysis variable;
99 run;

```

```

35 proc rank data=totrev groups=100
36 out=big_spenders2(where=(percentile GE 95));
37 ranks percentile; * name of variable assigning percentile to each obs in totrev;
38 var totrev; * analysis variable;
39 run;

NOTE: The data set WORK.BIG_SPENDERS2 has 822 observations and 15 variables.
NOTE: PROCEDURE RANK used (Total process time):
      real time           0.40 seconds
      cpu time            0.12 seconds

```

60



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

- Example 3
- 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 default numeric values of `_TYPE_` to a **character** variable indicating the combination of CLASS Statement variables
  - A string of zeros and ones, corresponding to the order of the variables in the CLASS Statement

61



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

- Example: Calculate the mean and sum of total revenue and total kilowatts grouped/classified by
  - Region
  - Region and office
  - Office
  - Region and rate schedule
  - Office and rate schedule
  - Region, office and rate schedule
    - Results are to be in six separate data sets.

62

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

```

118= data new(drop=kwh1-kwh12 rev1-rev12); * drop unneeded vars from output data set;
119   set sasclass.electricity(keep=region office cesched rev1-rev12 kwh1-kwh12);
120 * sum revenue and kwh for each customer;
121   totrev = sum(of rev1-rev12);
122   totkwh = sum(of kwh1-kwh12);
123   run;
124
125=proc means noprint chartype data=new;
126   class region office cesched;
127   var totrev totkwh;
128 * by region;
129   output out=a(where=( type = '100')) sum= mean=/autoname;
130 * by region and office;
131   output out=b(where=( _type_ = '110')) sum= mean=/autoname;
132 * by office;
133   output out=c(where=( _type_ = '010')) sum= mean=/autoname;
134 * by region and rate schedule;
135   output out=d(where=( _type_ = '101')) sum= mean=/autoname;
136 * by office and rate sched.;
137   output out=e(where=( _type_ = '011')) sum= mean=/autoname;|
138 * by region, office and rate sched.;
139   output out=f(where=( _type_ = '111')) sum= mean=/autoname;
140 run;
  
```

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

Tips & Techniques for Large Files  
 Creating Multiple Output Data Sets in a Single PROC MEANS Step  
 Results with AUTONAME Option

| Obs | REGION   | OFFICE | CESCHED | _TYPE_ | _FREQ_ | totrev_Sum | totkwh_Sum | totrev_Mean | totkwh_Mean |
|-----|----------|--------|---------|--------|--------|------------|------------|-------------|-------------|
| 1   | EASTERN  |        |         | 100    | 5124   | 3729942.73 | 30809144   | 727.936     | 6030.37     |
| 2   | NORTHERN |        |         | 100    | 5457   | 4553269.16 | 37869828   | 834.391     | 6951.14     |
| 3   | SOUTHERN |        |         | 100    | 720    | 580492.41  | 4887053    | 806.239     | 6787.57     |
| 4   | WESTERN  |        |         | 100    | 5068   | 3481429.22 | 28284251   | 686.943     | 5599.73     |



## Summary and Conclusions



- I've presented information about approaches to and tools you can use to apply the SAS System to large files
- Remember:
  - **Plan your Work, and Work your Plan !**

65



## Thank You

- Thank you for attending this presentation!
- Questions ?
- Comments ?
- Copies of the Presentation Slides
  - [www.SierraInformation.com](http://www.SierraInformation.com)

66

Copies of this Presentation:  
[www.SierraInformation.com](http://www.SierraInformation.com)

**SIERRA INFORMATION SERVICES**

HOME  
 ABOUT SIERRA  
 PRO  
 CONTACT US

**TRAINING SOLUTIONS**

- Current Public Training Schedule
- Online Training Seminars
- Detailed Seminar Descriptions/Outlines
- About Our Public Seminars
- Technical Details for Online Seminars
- Customized Training Seminars

**CONSULTING**

- Data Mining and Database Marketing
- Customer Segmentation
- Forecasting
- SAS Application
- Efficiency Improvement

**FREE DOWNLOADS**

**Maximizing your investment in the SAS® System...  
 ...and those in your organization who use it.**

Sierra Information Services is a leading independent provider of SAS consulting and training services. We offer a wide range of public and on-site training solutions showing you how to effectively utilize both SAS programming and Enterprise Guide tools. And, Sierra provides cost-effective consulting services when the intelligent application of SAS tools for business intelligence, analytics and statistical modeling can yield actionable insights from your organization's data. We look forward to helping you and your organization get the most from your investment in the SAS System!

Choosing Sierra as your consulting partner means actionable results from consultants with over 25 years of in-depth, hands-on, experience applying—and teaching others how to apply—SAS' capabilities to manage, report, analyze and display data. Exploring our site shows you how we can work with your organization to maximize your investment in the SAS System... and those in your organization who use it. We look forward to [hearing from you](#) soon!

- » [Andrew Karp Featured Speaker at 2010 Institute on Research and Statistics](#)
- » [New Online Classes and Schedule!](#)
- » [Seminars in Sacramento April/May 2010](#)
- » [Seminars in Washington, DC, Jun/Jul 2010](#)
- » [Download SAS-to-Excel via ExcelXP Target Tutorial Slides](#)

**Intensive Public Training Seminars**      **Customized SAS Training Solutions**

Our affordable public SAS training curriculum provides cost-effective instruction on a wide range of programming, reporting, data management and statistical analysis topics. Focus on just the SAS training you need with our full- and half-day sessions. We offer both classroom and internet-based sessions for SAS users at all experience levels. See [our current schedule of classes and seminars](#). Our financial strength is

Our flexible pricing plans and in-depth expertise makes Sierra the right choice for design and delivery of customized SAS training solutions at your workplace. We develop and present tailored on-site sessions showing how to apply SAS programming and/or Enterprise Guide tools to your organization's data, projects and [operations](#). Our financial strength is

67

Thanks for Attending This  
 Presentation!

**Virtual SAS®  
 Users Group**

[www.VirtualSUG.org](http://www.VirtualSUG.org)

68