Top 10 SAS best programming practices

Charu Shankar
SAS Education
SCSUG, Dallas Forth Worth
November 7, 2011

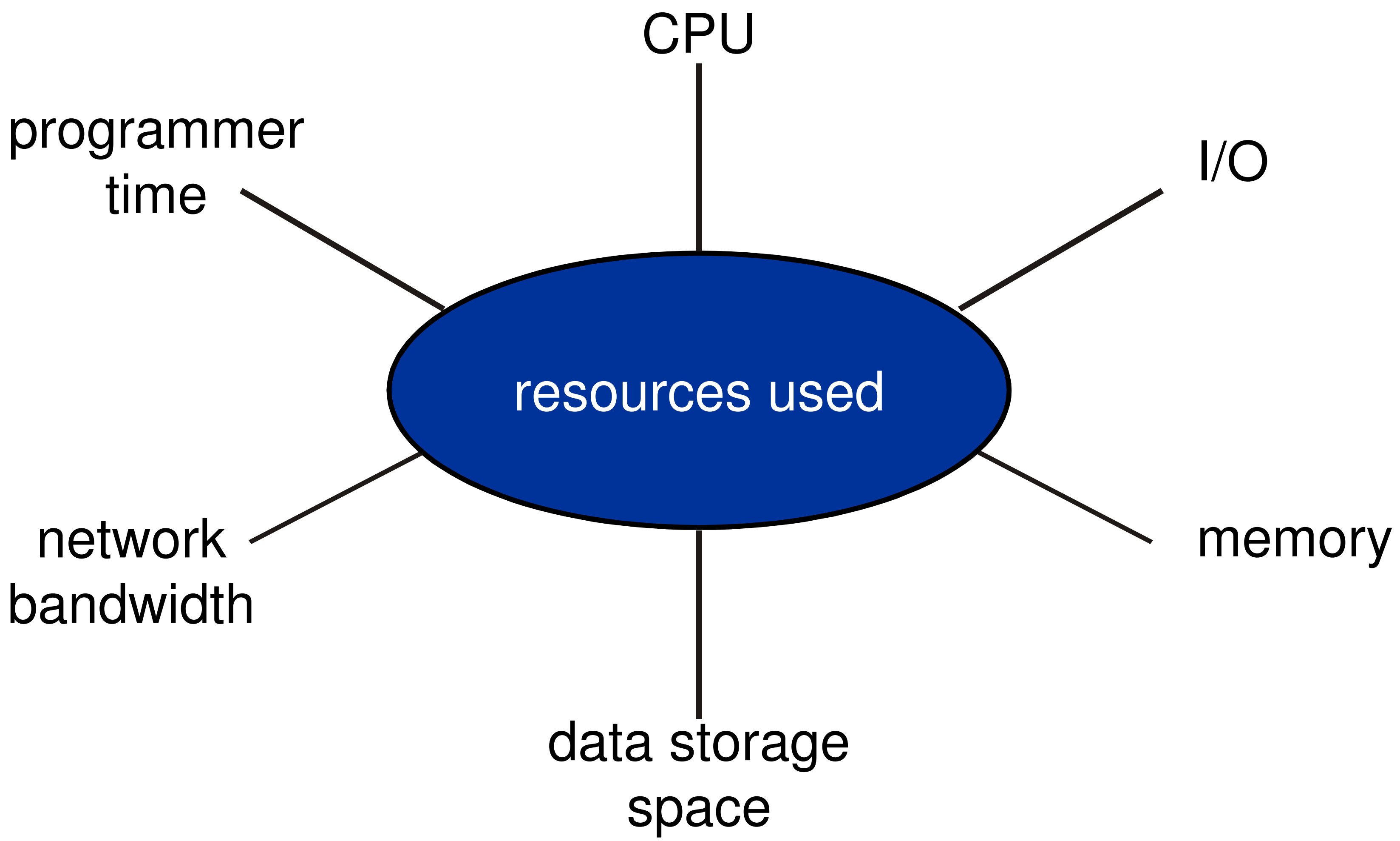
What can you expect to learn in this session

- Data Worker Rule #1
- Top 3 questions you need answered before you start your data work
- The only answer to your “What’s the best way to do this”?
- Reduce CPU time
- Reduce I/O
- Reduce Memory
- Reduce Space

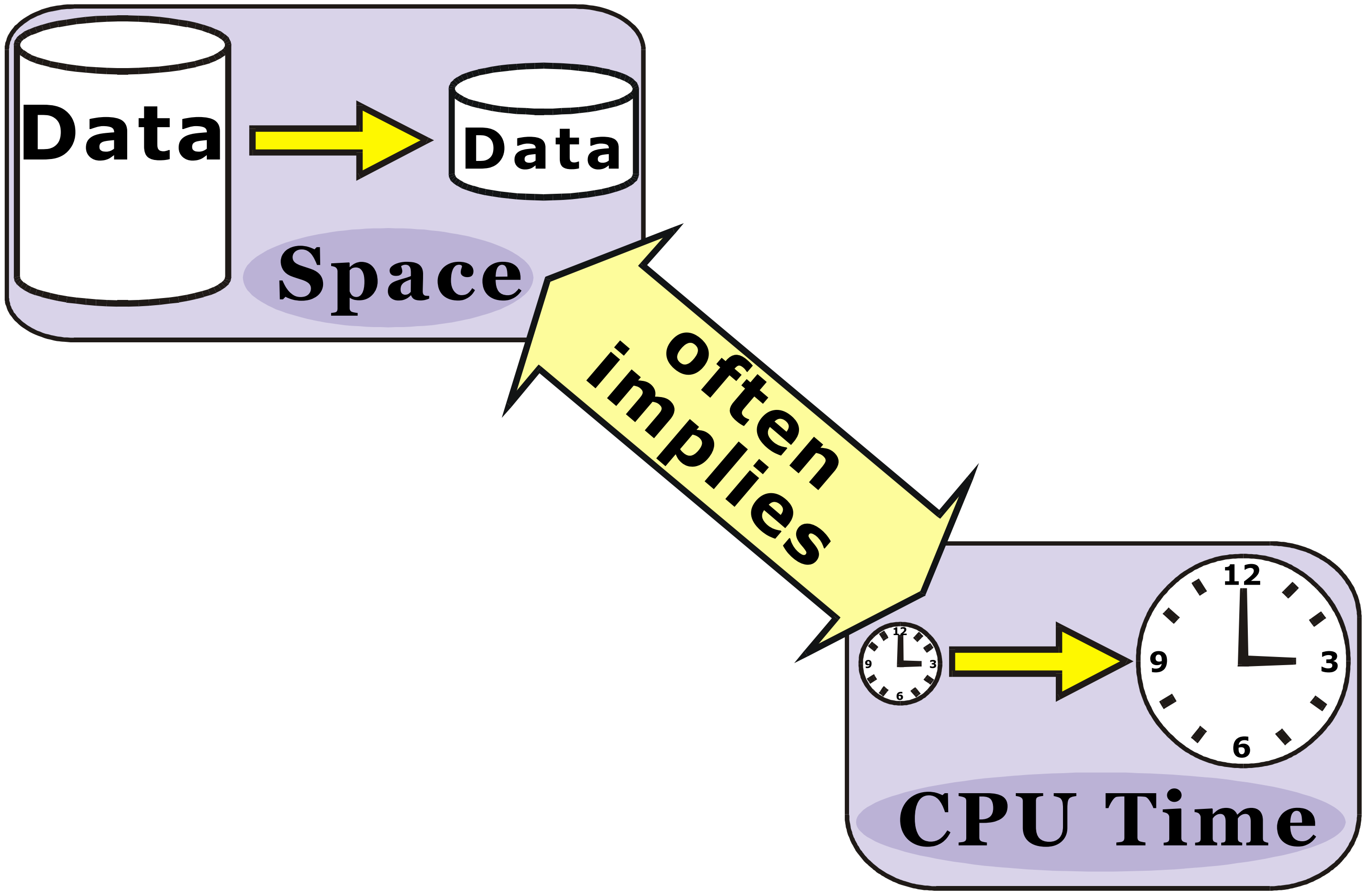
And last but not least

- Reduce your programing time

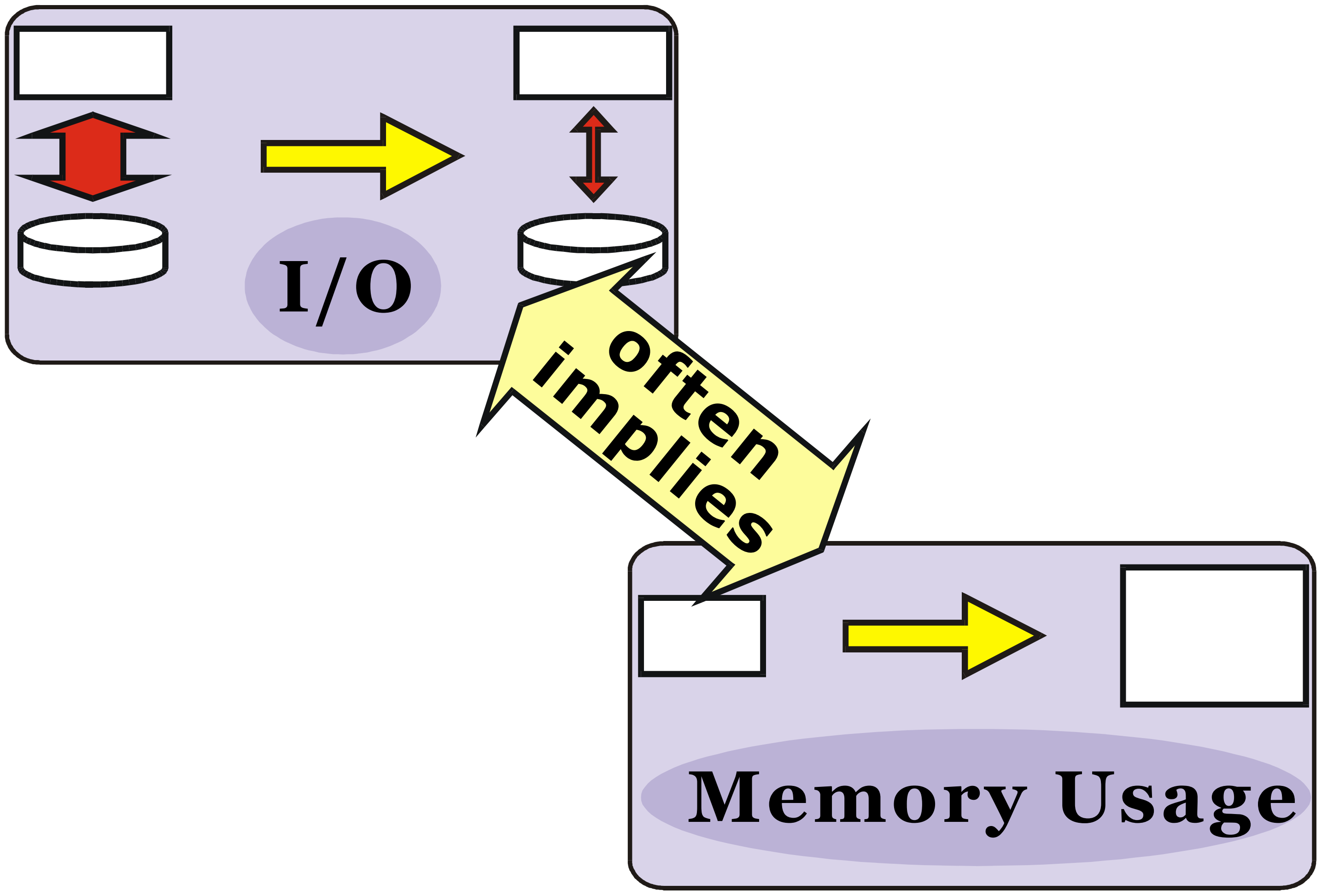
What 6 Resources Are Used?



Understanding Efficiency Trade-offs

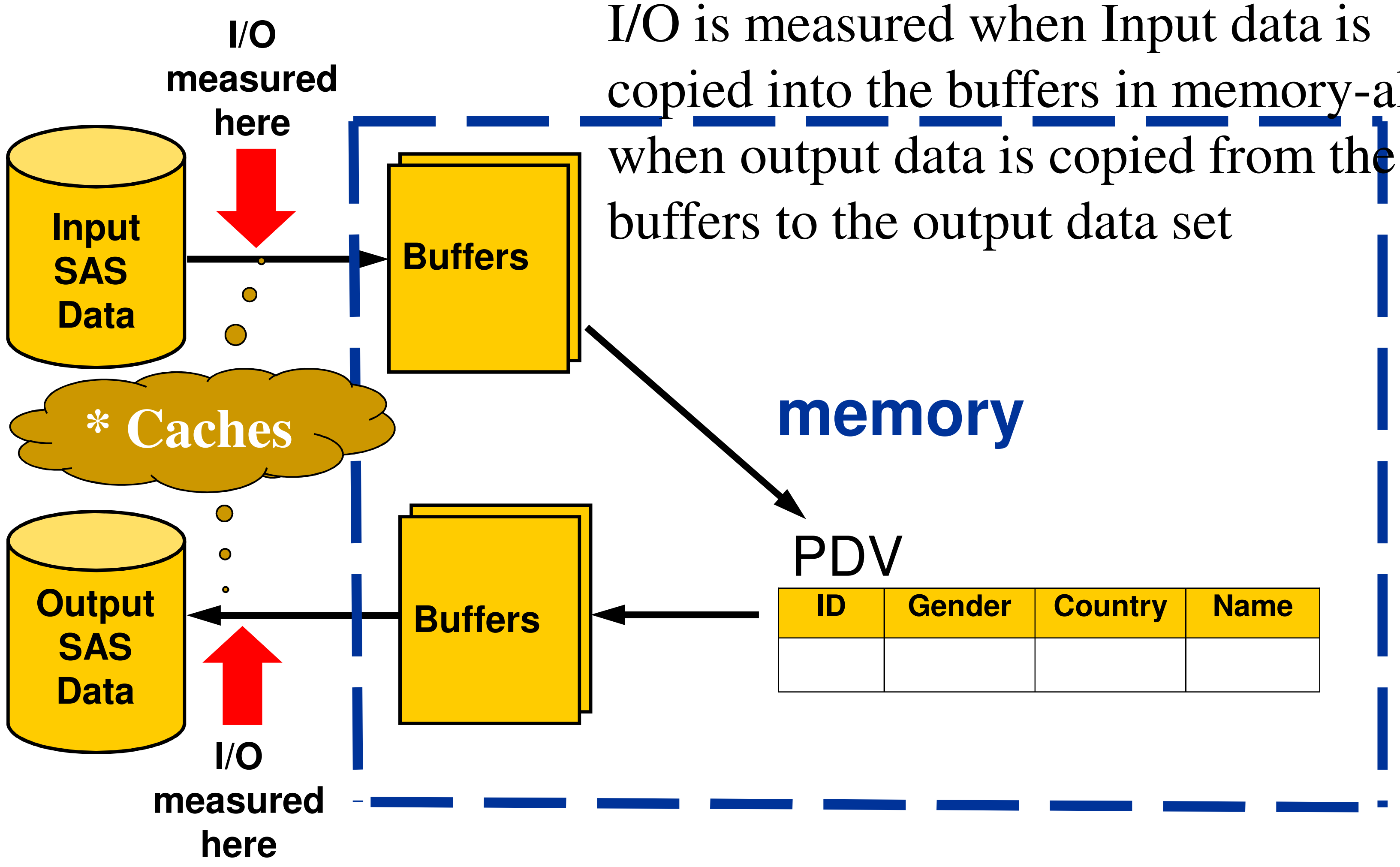


Understanding Efficiency Trade-offs



Where Is I/O Measured? (Review)

I/O is measured when Input data is copied into the buffers in memory-also when output data is copied from the buffers to the output data set



* Windows and UNIX Only

What’s the only answer to “What’s the best way to do this?”

It Depends!!

Best Practices for CPU savings

#1. Boiling down or reducing your data

#2. Doing conditional processing

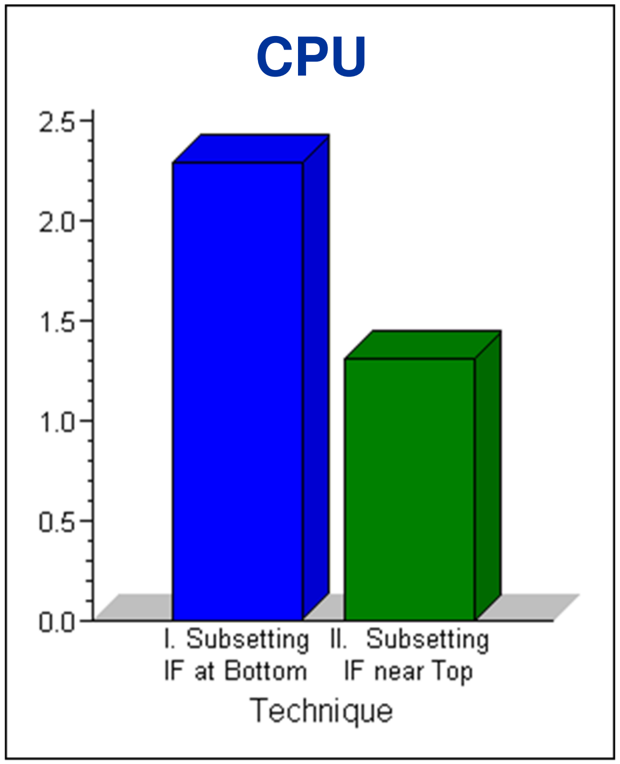
#3. Do not reduce the length of numeric variables.

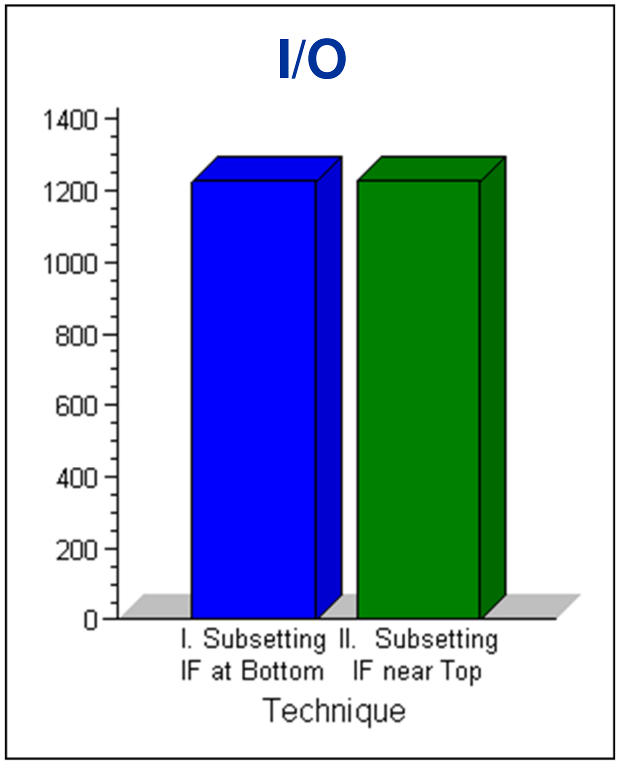
#1 Boiling down your data

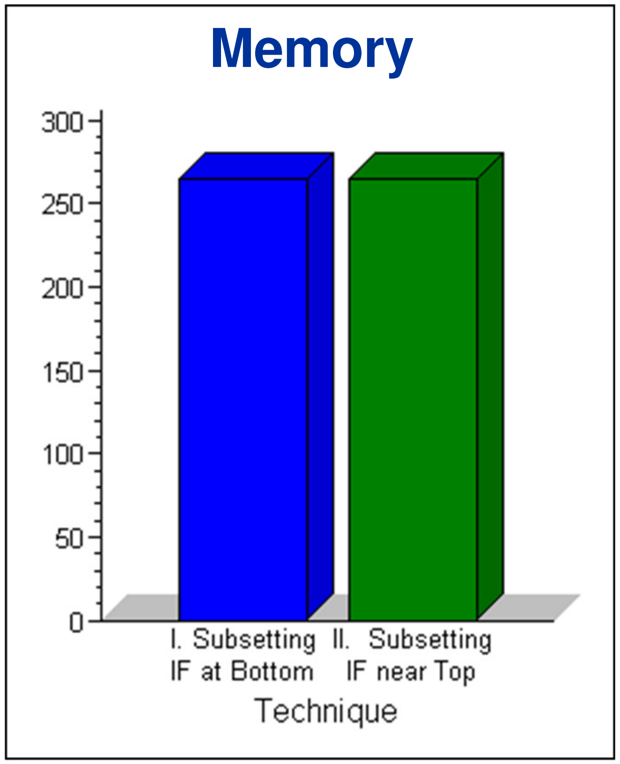
Where we take a look at the placement of statements in the Datastep

The Result- Let's compare techniques

Technique	CPU	I/O	Memory
I. Subsetting IF at Bottom	2.3	1226.0	265.0
II. Subsetting IF near Top	1.3	1226.0	265.0
Percent Difference	42.8	0.0	0.0





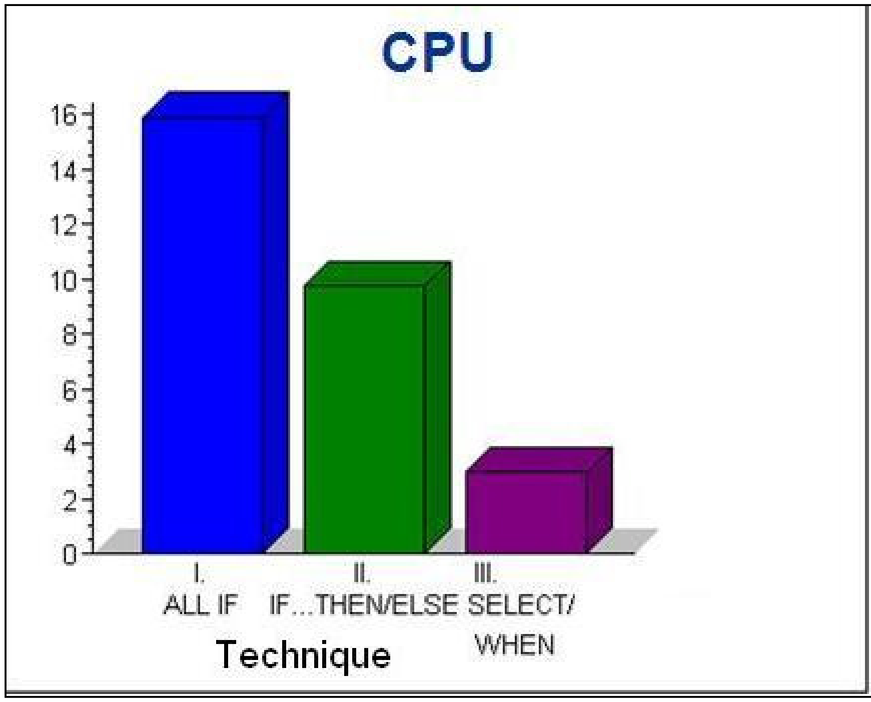


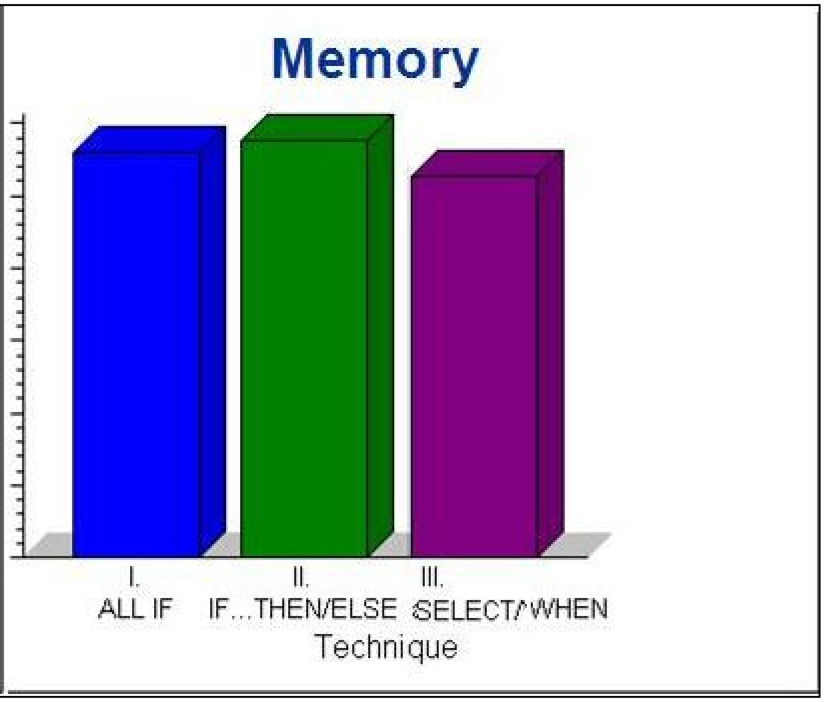
#2 Use conditional Logic

IF-THEN/ELSE	Executes a SAS statement for observations that meet a specific condition
SELECT	Executes one of several statements or groups of statements

The result-Let's compare Techniques

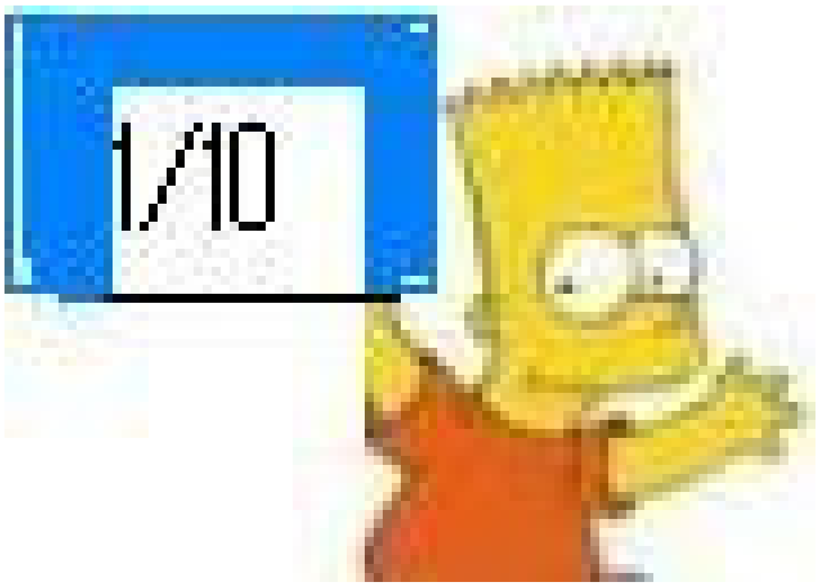
Technique	CPU	I/O	Memory
I. ALL IF Statements	15.9	6797.0	280.0
II. ELSE-IF Statements	9.7	6797.0	288.0
III. SELECT/WHEN Block	3.0	6795.0	263.0





The I/O for each technique is the same.

#3 Do not reduce the length of numeric data



Characteristics of Numeric Variables

Numeric variables have the following characteristics:

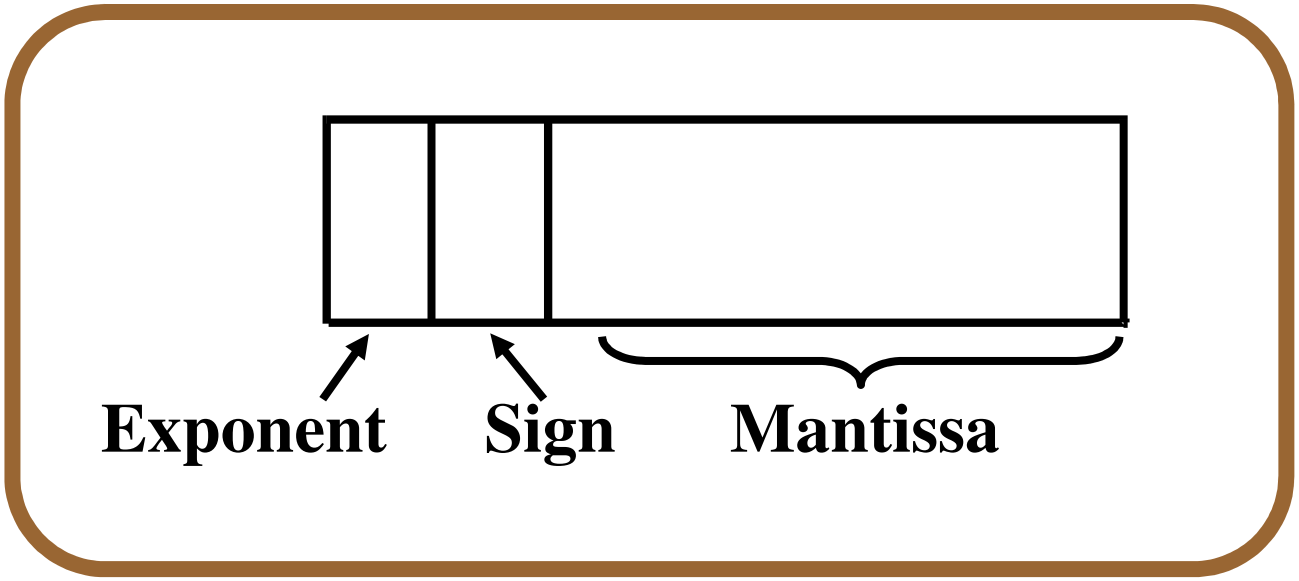
- are stored as floating-point numbers in real-binary representation
  - store multiple digits per byte
  - use a minimum of one byte to store the sign and exponent of the value (depending on the operating environment) and use the remaining bytes to store the mantissa of the value
- take 8 bytes of storage per variable, by default, but can be reduced in size
- always have a length of 8 bytes in the PDV

Default Length of Numeric Variables

The number 35,298 can be written as follows:

$$\underset{\text{Sign}}{+}\underbrace{0.35298}_{\text{Mantissa}}*(\underset{\text{Base}}{10}**\underset{\text{Exponent}}{5})$$

SAS stores numeric variables in floating-point form:



Possible Storage Lengths for Integer Values

Windows and UNIX

Length (bytes)	Largest Integer Represented Exactly
3	8,192
4	2,097,152
5	536,870,912
6	137,438,953,472
7	35,184,372,088,832
8	9,007,199,254,740,992

Possible Storage Lengths for Integer Values

z/OS

Length (bytes)	Largest Integer Represented Exactly
2	256
3	65,536
4	16,777,216
5	4,294,967,296
6	1,099,511,627,776
7	281,474,946,710,656
8	72,057,594,037,927,936

Assigning the Length of Numeric Variables

The use of a numeric length less than 8 bytes does the following:

- causes the number to be truncated to the specified length when the value is written to the SAS data set

⚠ This reduces the number of bytes available for the mantissa, which reduces the precision of the number that can be accurately stored.

- causes the number to be expanded to 8 bytes in the PDV when the data set is read by padding the mantissa with binary zeros

✎ Numbers are always 8 bytes in length in the PDV.

Dangers of Reduced-Length Numeric Variables

It is **not** recommended that you reduce the length of integer numeric variables inappropriately or that you reduce the length of variables that hold large integer numeric values. This example illustrates the effect of inappropriately reducing integer values.

```
data test;
   length X 3;
   X=8193;
run;

data _null_;
   set test;
   put X=;
run;
```

p202d07

Saving I/O

#4 Reduce multiple & unnecessary passes through data.
Create multiple output datasets from one pass of the input data, rather than processing the input data each time that you create an output data set.-use the data step over PROC SQL
Creating sorted subsets with the SORT procedure.

#5 Modify variable attributes.

#5 Manage your data with PROC DATASETS

Business task- Rename & format variable attributes in **choc.cesales_analysis** to be consistent with those in other datasets

	Var Name	Var Format
Ceorder_info	Prod_id	$7.
	Total_sales	.
Ceesales_analysis	Product_id	$7.
	Total_sales	Comma9.2.

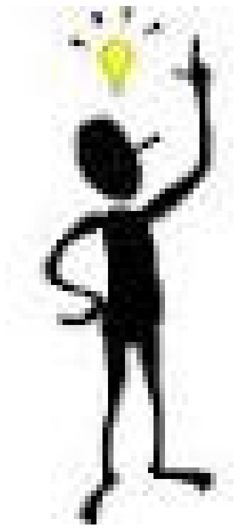
DATA Step / PROC DATASETS

```
data choc.ceorder_info;
set choc.ceorder_info;
rename prod_id=product_id;
format total_cases comma9.2;
run;
```

```
proc datasets library=choc;
modify ceorder_info
rename prod_id=product_id;
format total_cases comma9.2;
run;
```

DATA Step / PROC DATASETS

**So Which one is better for data management?
The Data Step or PROC Datasets?**



Did you know ? PROC Datasets needs a QUIT statement otherwise it just sits in memory waiting for you to submit another request.. So remember to end it with a QUIT statement

Techniques affecting CPU and/or IO

If you process fewer variables and observations,
CPU and/or I/O operations can be affected significantly.

# 6 Process only necessary variables & observations

I've only worked at a nucluer power plant. How do I make that look good on a teaching CV?

…

#6 Process only necessary variables

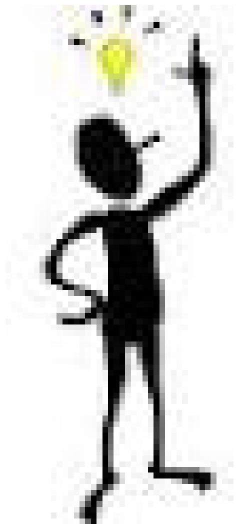
Simple techniques can conserve I/O. The amount of I/O saved depends on the size of the subset being processed.

6.1 Reduce the number of observations - WHERE in the In the Data step or WHERE in the PROC step

6.2. WHERE statement or IF statement

Consider- Which one is more efficient?

The Data Step & then subsetting in PROC MEANS
or
subsetting directly in the Datastep



Did you know ? The data step is a builder – that's why you had to use the data step here because you were creating a new variable. Otherwise PROC MEANS alone would have been enough!

6.2 Reduce Observations

Where or IF – that is the question?

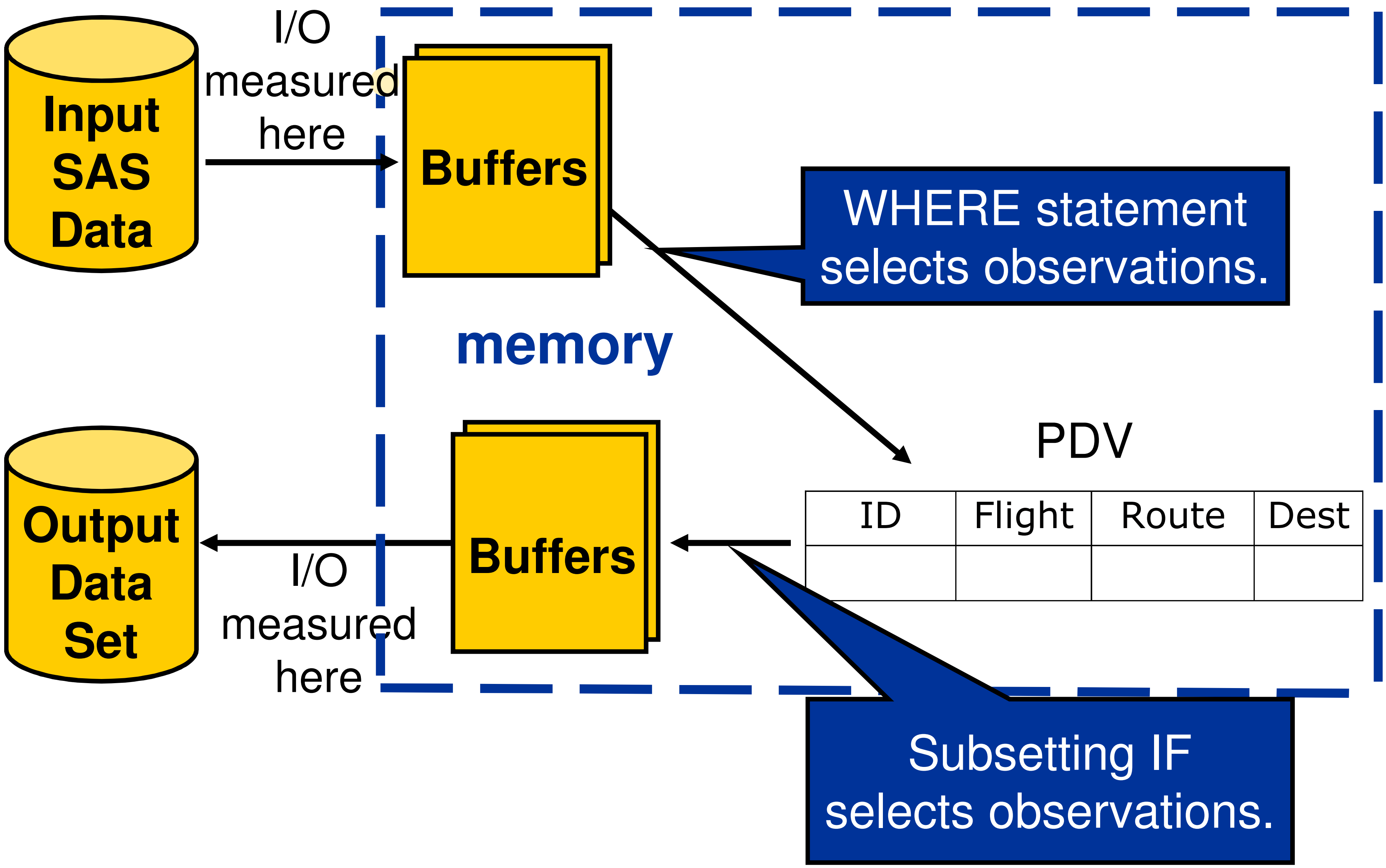
Subsetting IF or the Where clause?

Create a subset of the cesales_analysis dataset that contains data for Chocolate.

```
data chocolate;
set choc.cesales_analysis;
if category='Chocolate' ;
Run;

NOTE: There were 115928
observations read from the
data set
CHOC.CESALES_ANALYSIS.
NOTE: The data set
WORK.CHOCOLATE has 50368
observations and 11
variables.
NOTE: DATA statement used
(Total process time):
real time          2.84 seconds
cpu time           0.12 seconds
```

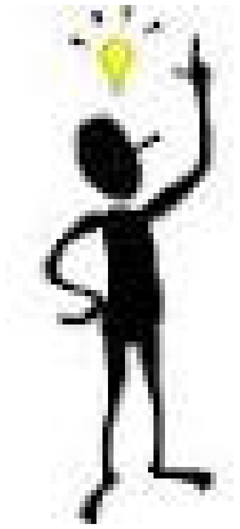
```
data chocolate;
set choc.cesales_analysis;
where category='Chocolate' ;
Run;

NOTE: There were 50368
observations read from the data
set CHOC.CESALES_ANALYSIS.
WHERE category='Chocolate';
NOTE: The data set
WORK.CHOCOLATE has 50368
observations and 11 variables.
NOTE: DATA statement used
(Total process time):
real time          2.26 seconds
cpu time           0.06 seconds
```

The Subsetting IF and the WHERE Statements



Consider- When to use which one?

The WHERE clause
Or
The Subsetting IF
The answer lies in this question - do you want to subset existing obs or newly created obs?

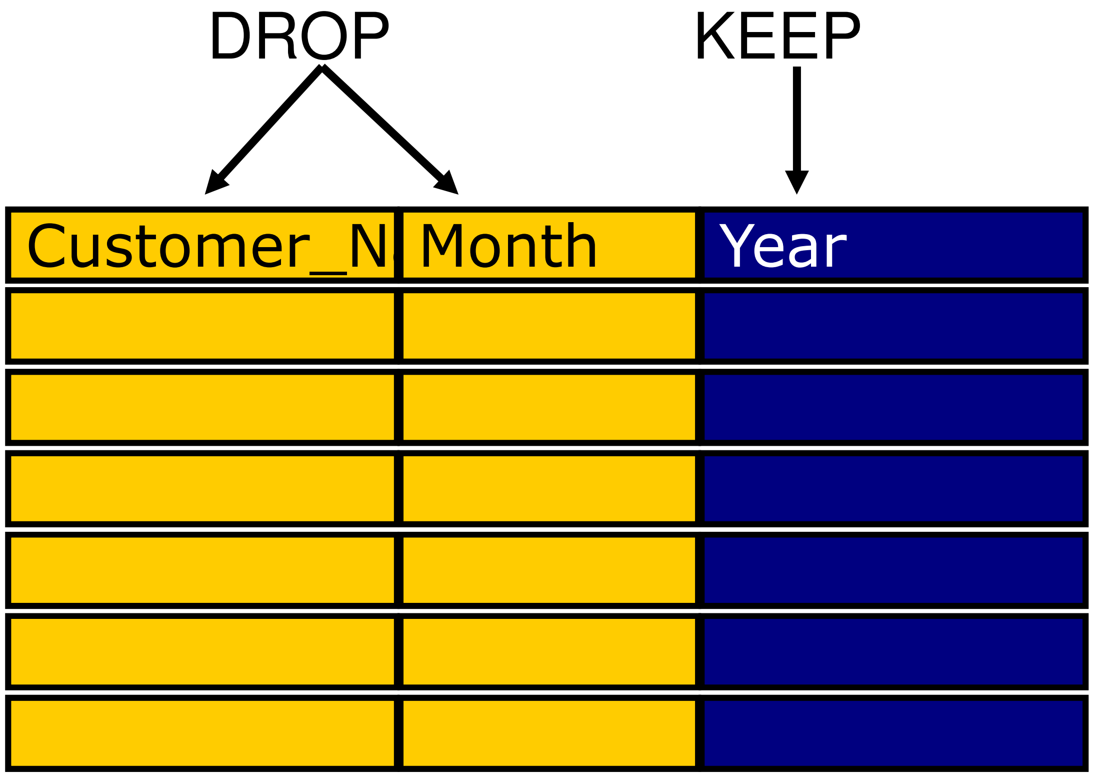


Did you know ? The WHERE clause is the same one used in SQL. If you want to subset existing obs use the WHERE. The powerful WHERE acts on obs before moving it to the PDV. The IF statement works on newly created var but has to read in row by row into the PDV thus slower in comparison

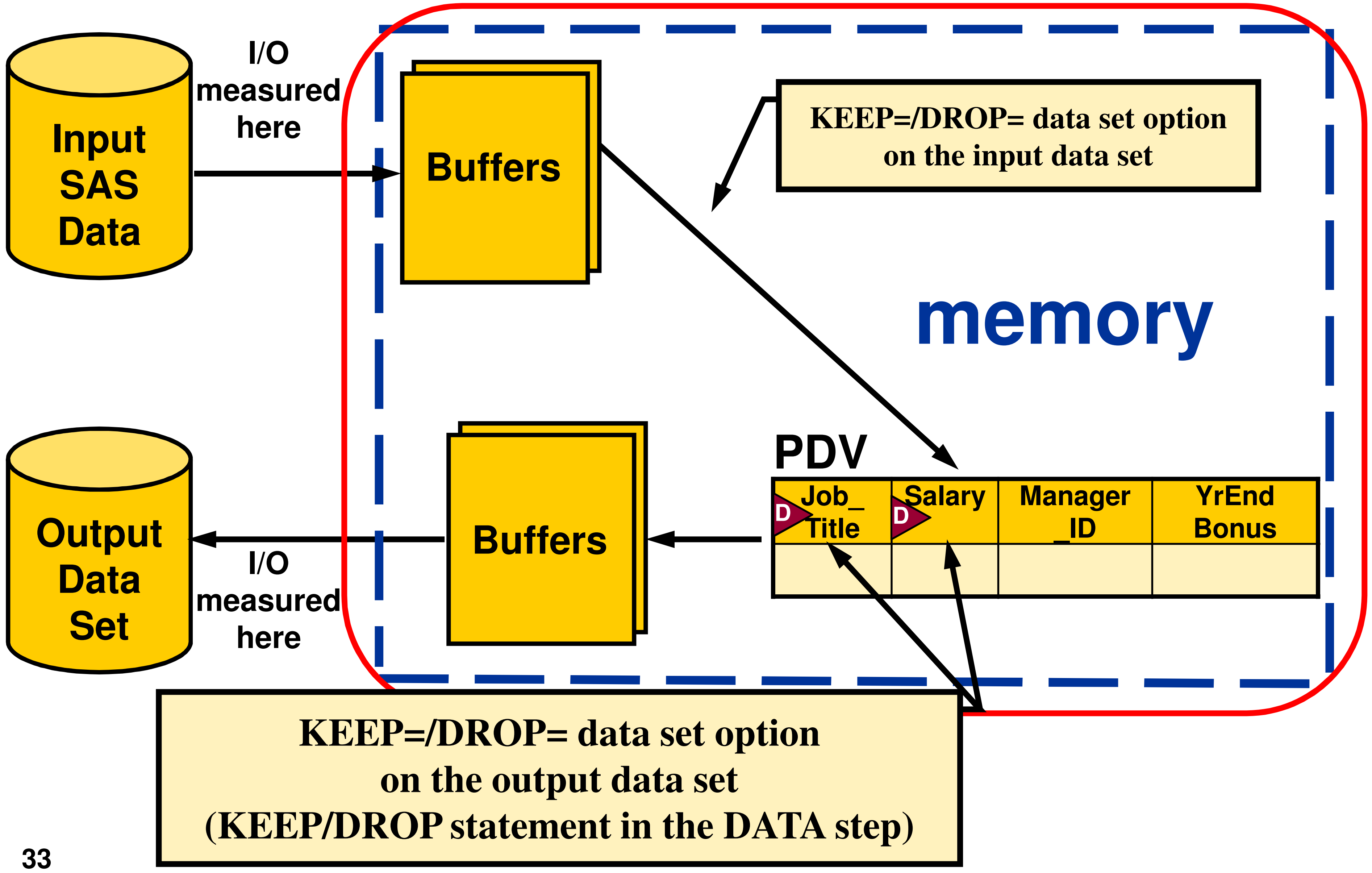
#7 Process only the necessary variables

To subset variables, you can use the following:

- DROP and KEEP statements
- DROP= and KEEP= data set options



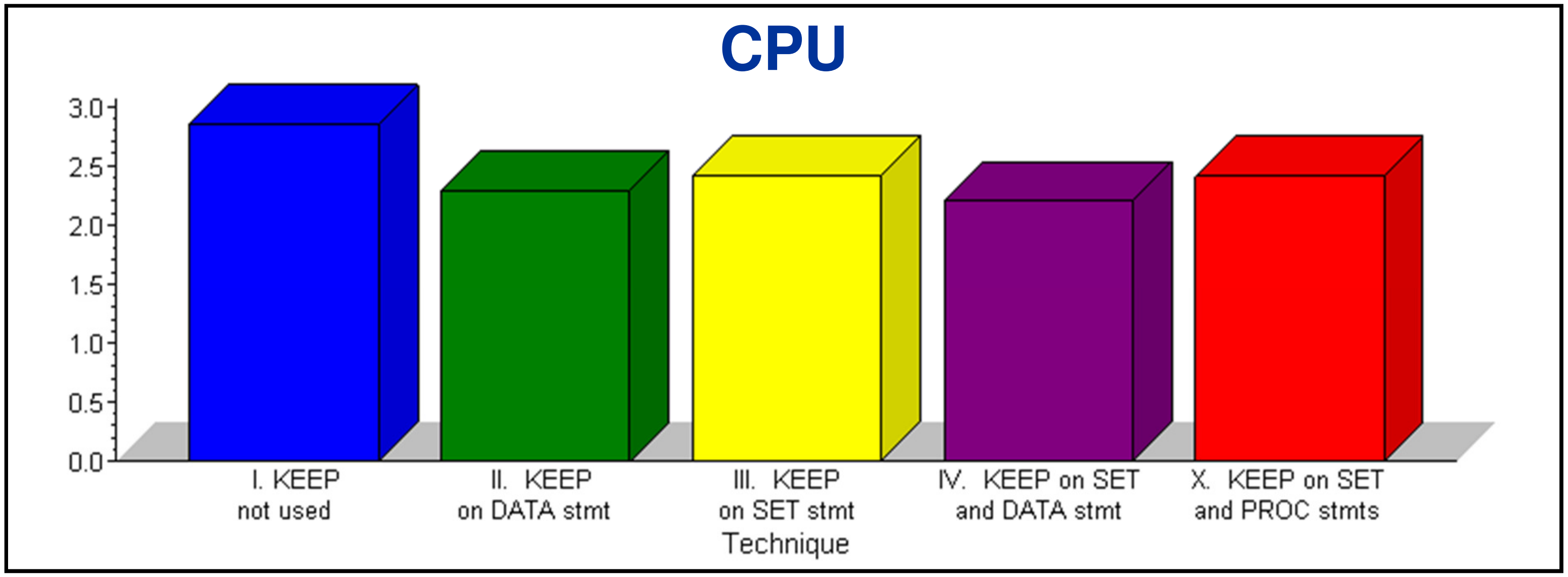
Using the KEEP=/DROP= Options



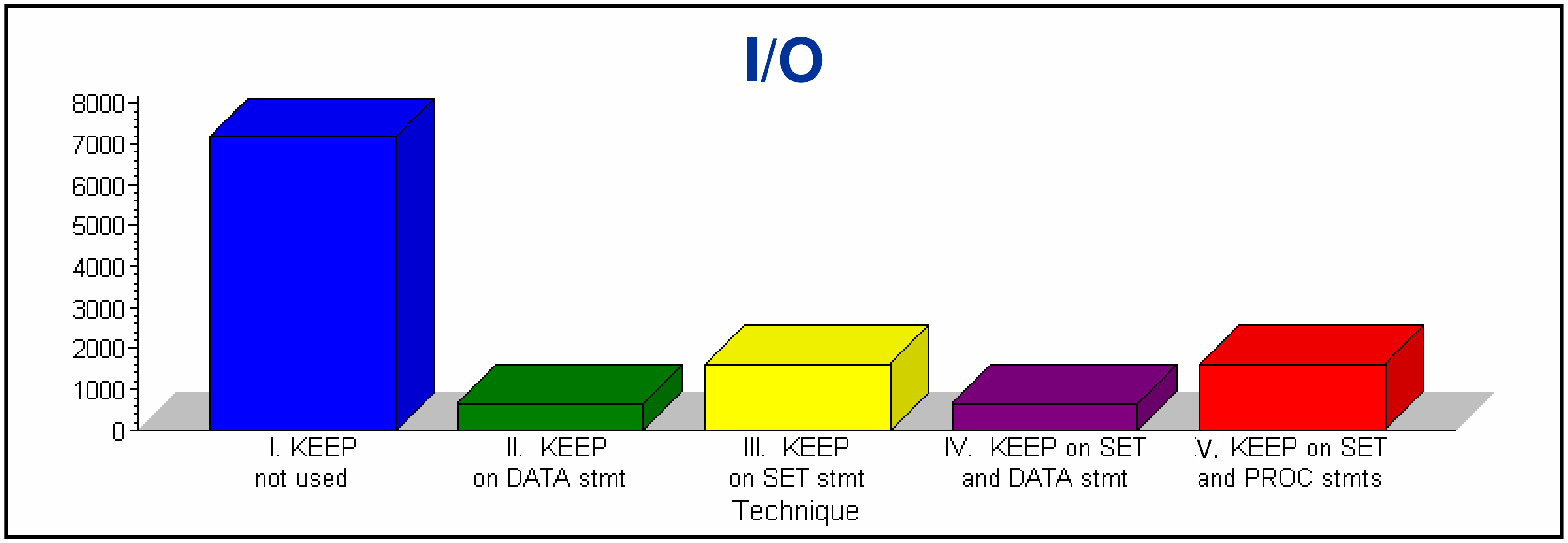
Input SAS Data
I/O measured here
Buffers
KEEP=/DROP= data set option on the input data set
memory
PDV
D
Job_ Title
D
Salary
Manager _ID
YrEnd Bonus
Output Data Set
I/O measured here
Buffers
KEEP=/DROP= data set option on the output data set (KEEP/DROP statement in the DATA step)

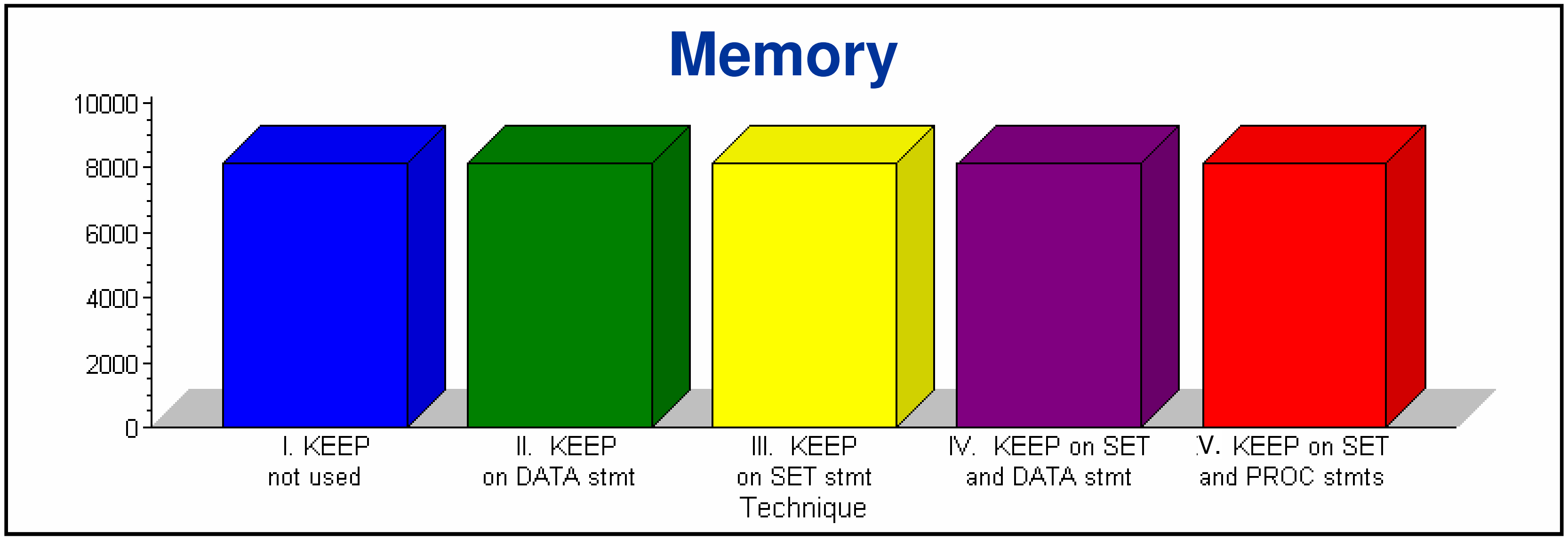
Comparing Techniques

Technique	CPU	I/O	Memory
I. KEEP not used	2.9	7177	8140
II. KEEP on DATA statement	2.3	656	8138
III. KEEP on SET statement	2.4	1625	8138
IV. KEEP on SET and DATA statements	2.2	662	8138
V. KEEP on SET and PROC statements	2.4	1625	8139

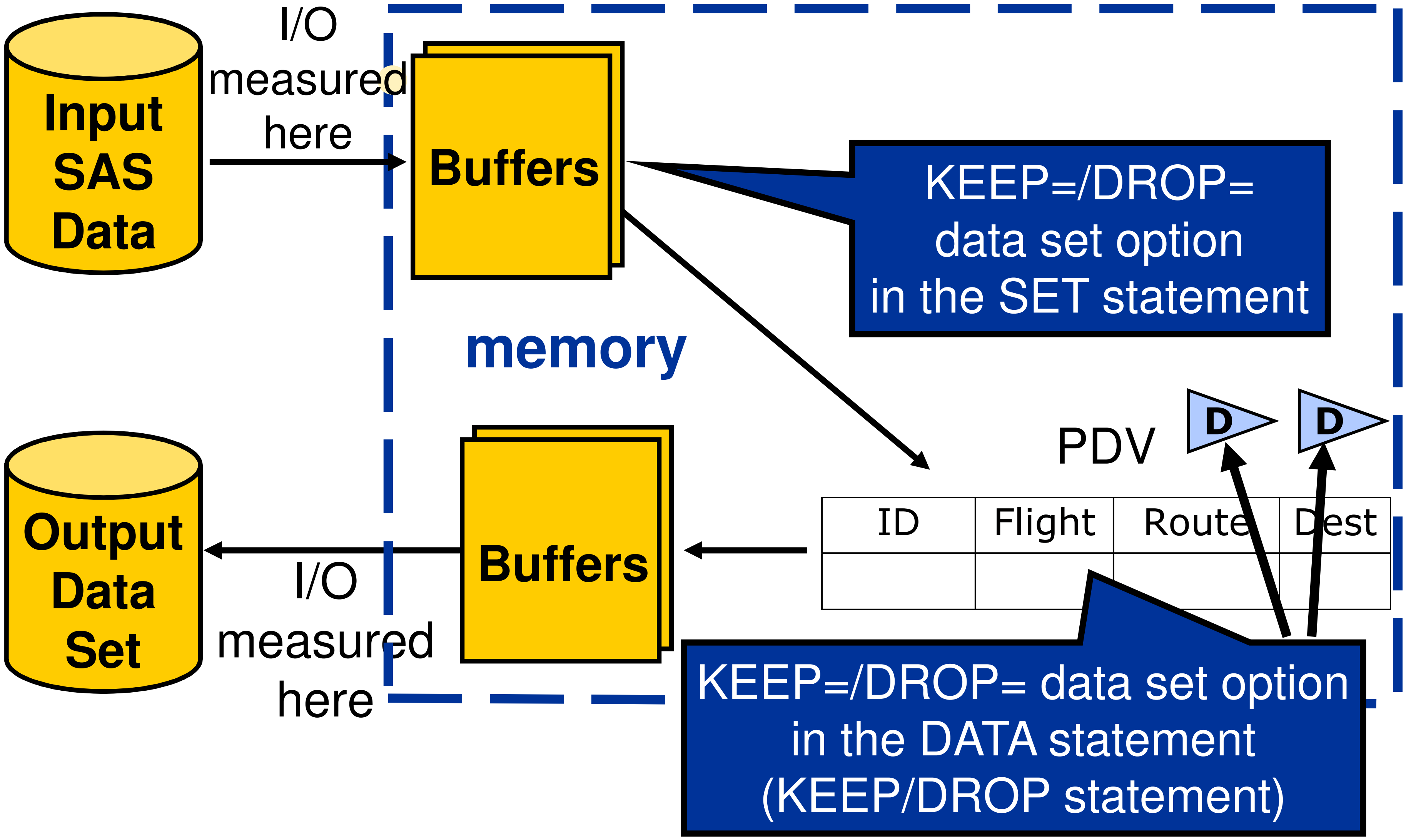


Comparing Techniques





Using the KEEP=/DROP= Options


Input SAS Data
I/O measured here
Buffers
KEEP=/DROP= data set option in the SET statement
memory
PDV
D
D
ID
Flight
Route
Dest
Output Data Set
I/O measured here
Buffers
KEEP=/DROP= data set option in the DATA statement (KEEP/DROP statement)

Best practice - Saving Space

#8 Store data as character to manage space

What type should my data be—Character or numeric?

Saving memory

I always have trouble remembering three things: faces, names, and -- I can't remember what the third thing is.

Fred A. Allen

#10. Programmer’s time saving

10.1 Getting intimate with the SAS display manager

- The log-
- Shortcuts-keys, comments
- Using macros to understand your recent log

10.2 Getting to know your data-Enter the PROCS

- Dictionary Tables
- Which proc gives you duplicates with a special option &_No its not PROC SORT
- Which proc lets you look at the highest value

10.3 Variable shortcuts

10.4 Stealing code from SAS

10.5 When does a function work better than an Operator

10.6 What options keep me from accidentally overwriting source data

Tips & tricks to manage the SAS display manager

Last Word

What is the data worker’s rule #1?

What are 3 questions to ask before jumping to data work

Top 10 SAS best programming practices:

#1. Boiling down or reducing your data

#2. Do conditional processing

#3. Do not reduce the length of numeric variables

#4 Reduce multiple passes of your data

#5 Manage your data with PROC Datasets

#6 Process only necessary observations

#7 Process only necessary variables

#8 Store data as character type to save space

#9 Use the BY statement instead of CLASS to save space

#10 Finally its all about YOU & your time-many tips

Thanks for your time

Questions

Contact

Charu Shankar

Technical Training Specialist

SAS Institute, Toronto

Charu.shankar@sas.com