Lec 3
Efficiency

Measuring I/O

- I/O is measured when input data is copied to the buffer in memory and when it is copied from the output buffer to the output data set.

- Improvement in I/O can come at the cost of increased memory consumption.

- When you create a SAS data set using a DATA step, the following actions occur:

- SAS copies a page of data from the input data set to a buffer in memory.
- One observation at a time is loaded from the buffer into the program data vector.
- Each observation is written from the PDV to an output buffer.
- The contents of the output buffer are written to the disk when the buffer is full.

Page Size

- Buffer holds exactly one page of data.

- A page is described as follows:

- It is the unit of data transfer between the storage device and memory.
- It is fixed in size when the data set is created, either to a default value or to a user-specified value.
- A larger page size can reduce execution time by reducing the number of times SAS has to read from or write to the storage medium. However, the improvement in execution time comes at the cost of increased memory consumption.

Reporting Page Size

- You can use the CONTENTS procedure or the CONTENTS statement in the DATASETS procedure to report the page size and the number of pages.

- The total number of bytes that a data file occupies equals the page size multiplied by the number of pages.

Using the BUFSIZE= Option

- In certain cases, choosing a page size or buffer size that is larger than the default can speed up execution time by reducing the number of times that SAS must read from or write to the storage medium.

- You can use the BUFSIZE= system option or data set option to control the page size of an output SAS data set.

- The new buffer size is a permanent attribute of the data set. After it is specified, it is used whenever the data set is processed.

  In the following program, the BUFSIZE= system option specifies a page size of 30720 bytes.

```
options bufsize=30720;
data overview_new;
set overview;
run;
```

Using the BUFNO= Option

- You can use the BUFNO= system or data set option to control the number of buffers that are available for reading or writing a SAS data set.

- By increasing the number of buffers, you can control how many pages of data are loaded into memory with each I/O transfer.

- When you work with a small data set, allocate as many buffers as there are pages in the data set so that the entire data set can be loaded into memory. This technique is most effective if you read the same observations several times during processing.

  In the following program, the BUFNO= system option specifies that 4 buffers are available.

```
options bufno=4;
filename orders 'c:\orders.dat';
data company.orders_fact;
infile orders;
<more SAS code>
run;
proc print data=company.orders_fact;
run;
```

- The product of BUFNO= and BUFSIZE=, determines how much data can be transferred in one I/O operation.
- Increasing the value of either option increases the amount of data that can be transferred in one I/O operation.

BUFSIZE	BUFNO	Bytes Transferred in One I/O Operation
6144	2	12,288
6144	10	61,440
30,720	2	61,440
30,720	10	307,200

Example 38:
The following SAS program is submitted:

```
data WORK.NEW(bufno=4);
  set WORK.OLD(bufno=3);
run;
```

Why are the BUFNO options used?
A. to reduce memory usage
B. to reduce CPU time usage
C. to reduce the amount of data read
D. to reduce the number of I/O operations

SASFILE statement

- SASFILE statement is used to hold a SAS data file in memory so that the data is available to multiple program steps.

- Keeping the data file open, reduces I/O processing and open/close operations, including the allocation and freeing of memory for buffers.

SASFILE *SAS-data-file* <(*password-option(s)*)> **OPEN | LOAD | CLOSE;**

- The SASFILE statement opens a SAS data file and allocates enough buffers to hold the entire file in memory.

- Once the data file is read, the data is held in memory, and it is available to subsequent DATA and PROC steps or applications until either of the following occurs:

- A SASFILE CLOSE statement frees the buffers and closes the file
- The SAS session ends, which automatically frees the buffers and closes the file

```
sasfile company.sales load;

proc print data=company.sales;
var Customer_Age_Group;
run;

proc tabulate data=company.sales;
class Customer_Age_Group;
var Customer_BirthDate;
table Customer_Age_Group,Customer_BirthDate*(mean median);
run;

sasfile company.sales close;
```

Note: The SASFILE statement can also be used to reduce CPU time and I/O in SAS programs that repeatedly read one or more SAS data views.

- It is important to note that I/O processing is reduced only if there is sufficient real memory. If there is not sufficient real memory, the operating environment might use the following:

- Virtual memory (If SAS uses virtual memory, there might be a degradation in performance)
- The default number of buffers

Example 39:
The SASFILE statement requests that a SAS data set be opened and loaded into memory:
A. one page at a time.
B. one variable at a time.
C. one observation at a time.
D. in its entirety, if possible.

SAS Benchmark Options

- System options STIMER, MEMRPT, FULLSTIMER, and STATS to track and report on resource use.

Option	z/OS	UNIX and Windows
STIMER	Specifies that the CPU time is to be tracked throughout the SAS session.	Specifies that CPU time and real-time statistics are to be tracked and written to the SAS log throughout the SAS session.
	Can be set at invocation only.	Can be set either at invocation or by using an OPTIONS statement.
	Is the default setting.	Is the default setting.
MEMRPT	Specifies that memory usage statistics are to be tracked throughout the SAS session.	Not available as a separate option; this functionality is part of the FULLSTIMER option.
	Can be set either at invocation or by using an OPTIONS statement.	
	Is the default setting.	
FULLSTIMER	Specifies that all available resource usage statistics are to be tracked and written to the SAS log throughout the SAS session.	Specifies that all available resource usage statistics are to be tracked and written to the SAS log throughout the SAS session.
	Can be set either at invocation or by using an OPTIONS statement.	Can be set either at invocation or by using an OPTIONS statement.
	In the z/OS operating environment, FULLSTIMER is an alias for the FULLSTATS option.	In Windows operating environments, some statistics are not calculated accurately unless FULLSTIMER is specified at invocation.
	This option is ignored unless STIMER or MEMRPT is in effect.	
STATS	Tells SAS to write statistics that are tracked by any combination of the preceding options to the SAS log.	Not available as a separate option.
	Can be set either at invocation or by using an OPTIONS statement.	
	Is the default setting.	

- To decide which SAS programming technique is most efficient for a task, you can benchmark (measure and compare) the resource usage for each technique that you are comparing.

Example 40:
Which of the following SAS System options can aid in benchmarking?
A. BUFSIZE= and BUFNO=
B. FULLSTIMER
C. IOBLOCKSIZE=
D. SYSTIMER

Example 41:
When attempting to minimize memory usage, the most efficient way to do group processing when using the MEANS procedure is to use:
A. the BY statement.
B. GROUPBY with the NOTSORTED specification.
C. the CLASS statement.
D. multiple WHERE statements.

- Use BY-group processing instead of CLASS statements in those procedures that support both, especially where you have pre-sorted data or can use an existing index.

Example 42:
A quick rule of thumb for the space required to run PROC SORT is:
A. two times the size of the SAS data set being sorted.
B. three times the size of the SAS data set being sorted.
C. four times the size of the SAS data set being sorted.
D. five times the size of the SAS data set being sorted.

- When you use the SAS sort, a quick rule of thumb for sort space is four times the size of the SAS data set.

- When you sort in place (sort a data set back to the same name), you need enough space in the library for two copies of the data.

Example 43:
Multi-threaded processing for PROC SORT will affect which of these system resources?
A. CPU time will decrease, wall clock time will decrease
B. CPU time will increase, wall clock time will decrease
C. CPU time will decrease, wall clock time will increase
D. CPU time will increase, wall clock time will increase

- Some procedures enable you to use threaded processing to reduce real time. Threaded processing can increase CPU time.

Compressed File

- By default, a SAS data file is uncompressed. You can compress your data files to conserve disk space, although some files are not good candidates for compression.

- The file structure of a compressed data file is different from the structure of an uncompressed file.

- You use the COMPRESS= data set option or system option to compress a data file.
- You use the POINTOBS= data set option to enable SAS to access observations in compressed files directly rather than sequentially.
- You use the REUSE= data set option or system option to specify that SAS should reuse space in a compressed file when observations are added or updated.

Deciding Whether to Compress a Data File

- When SAS need to read a compressed file, each observation must be uncompressed. This requires more CPU resources than reading an uncompressed file.

- However, compression can be beneficial when the data file has one or more of the following properties:

- It is large.
- Many long character values.
- Many values that have repeated characters or binary zeros.
- Many missing values.
- Repeated values in variables that are physically stored next to one another (blanks).

➢ A data file is not a good candidate for compression if it has any of the following characteristics:
- Few repeated characters
- Small physical size
- Few missing values
- Short text strings

The COMPRESS= System Option and the COMPRESS= Data Set Option

➢ To compress a data file, you use either the COMPRESS= data set option or the COMPRESS= system option.
OPTIONS COMPRESS= NO | YES | CHAR | BINARY;

➢ NO-default setting, which does not compress the data set.
Note: The COMPRESS= data set option overrides the COMPRESS= system option.

(1) The YES or CHAR setting for the COMPRESS= option uses the **RLE** compression algorithm.

RLE compresses observations by reducing repeated consecutive characters (including blanks) to two-byte or three-byte representations.

Therefore, RLE is most often useful for character data that contains repeated blanks.

The YES or CHAR setting is also good for compressing numeric data in which most of the values are zero.

(2) The BINARY setting for the COMPRESS= option uses **RDC**, which combines run-length encoding and sliding-window compression.

This method is highly effective for compressing medium to large blocks of binary data (numeric variables).

A file that has been compressed using the BINARY setting of the COMPRESS= option takes significantly more CPU time to uncompress than a file that was compressed with the YES or CHAR setting.

BINARY is more efficient with observations that are several hundred bytes or more in length.
BINARY can also be very effective with character data that contains patterns rather than simple repetitions.

Example 44:
Which of the following is true about the COMPRESS=YES data set option?
A. It uses the Ross Data Compression method to compress numeric data.
B. It is most effective with character data that contains repeated characters.
C. It is most effective with numeric data that represents large numeric values.
D. It is most effective with character data that contains patterns, rather than simple repetitions.

The REUSE= System Option and the REUSE= Data Set Option

- SAS appends new observations to the end of all data sets by default.
- If you delete an observation within the data set, empty disk space remains in its place.
- However, in compressed data sets only, it is possible to track and reuse free space when you add or update observations.
- By reusing space within a data set, you can conserve data storage space.
- The REUSE= system option and the REUSE= data set option specify whether SAS reuses space when observations are added to a compressed data set.

- If you set the REUSE= data set option to YES in a DATA statement, SAS tracks and reuses space in the compressed data set that is created in that DATA step.
- If you set the REUSE= system option to YES, SAS tracks and reuses free space in all compressed data sets that are created for the remainder of the current SAS session.

Note: The REUSE= data set option overrides the REUSE= system option.

The following program creates a compressed data set named Company.Customer_Compressed from the Company.Customer data set.

- Because the REUSE= option is set to YES, SAS tracks and reuses any empty space within the compressed data set.

```
data company.customer_compressed (compress=yes reuse=yes);
set company.customer;
run;
```

Example 45:
The following SAS program is submitted:

```
options reuse=YES;
data SASUSER.REALESTATE(compress=CHAR);
   set SASUSER.HOUSES;
run;
```

What is the effect of the reuse=YES SAS system option?

A. It allows updates in place.
B. It tracks and recycles free space.
C. It allows a permanently stored SAS data set to be replaced.
D. It allows users to access the same SAS data set concurrently.

Using SAS DATA Step Views to Conserve Data Storage Space

- Another way to save disk space is to leave your data in its original location and use a SAS data view to access it.

- A SAS data file and a SAS data view are both types of SAS data sets.

- A SAS data file contains both descriptor information and the data.
- A SAS data view, contains only descriptor information and instructions on how to retrieve data stored elsewhere.

- The main difference between SAS data files and SAS data views is where the data is stored. Data views can be particularly useful if you are working with data that changes often.

- In most cases, you can use a SAS data view as if it were a SAS data file.

DATA Step Views

- A DATA step view can be created only in a DATA step and it cannot be indexed or compressed.

- It is useful when you need to do the following:
  - Always access the most current data in changing files
  - Avoid storing a copy of a large data file
  - Combine data from multiple sources

Example 46:
Which of the following is an advantage of SAS views?
A. SAS views can access the most current data in files that are frequently updated.
B. SAS views can avoid storing a SAS copy of a large data file.
C. SAS views can decrease programming time.
D. both A and B are true

- To create a DATA step view, specify the VIEW= option after the final DATA set name in the DATA statement.
  DATA *SAS-data-view* <*SAS-data-file-1 ... SAS data-file-n*> **/ VIEW=***SAS-data-view***;**
  <*SAS statements*>
  RUN;

- The VIEW= option tells SAS to compile, but not to execute, the source program and to store the compiled code in the DATA step view that is named in the option.

Note: If you specify additional data files in the DATA statement, SAS creates these data files when the view is processed in a subsequent DATA or PROC step. Therefore, you need to reference the data view before you attempt to reference the data files in later steps.

```
data company.newdata / view=company.newdata;
infile <fileref>;
<DATA step statements>
run;
```

Example 47:
The following SAS code is submitted:

```
   data WORK.TEMP WORK.ERRORS / view=WORK.TEMP;
     infile RAWDATA;
     input Xa Xb Xc;
     if Xa=. then output WORK.ERRORS;
     else output WORK.TEMP;
  run;
```

Which of the following is true of the WORK.ERRORS data set?
A. The data set is created when the DATA step is submitted.
B. The data set is created when the view TEMP is used in another SAS step.
C. The data set is not created because the DATA statement contains a syntax error.
D. The descriptor portion of WORK.ERRORS is created when the DATA step is submitted.

Example 48:
The following program is submitted to check the variables Xa, Xb, and Xc in the SASUSER.LOOK data set:

```
   data _null_ WORK.BAD_DATA / view=WORK.BAD_DATA ;
      set SASUSER.LOOK(keep=Xa Xb Xc);
      length _Check_ $ 10 ;
      if Xa=. then _check_=trim(_Check_)!!" Xa" ;
      if Xb=. then _check_=trim(_Check_)!!" Xb" ;
      if Xc=. then _check_=trim(_Check_)!!" Xc" ;
      put Xa= Xb= Xc= _check_= ;
   run ;
```

When is the PUT statement executed?
A. when the code is submitted
B. only when the WORK.BAD_DATA view is used
C. both when the code is submitted and the view is used
D. never, the use of _null_ in a view is a syntax error

The DESCRIBE Statement

- DATA step views retain source statements. You can retrieve these statements by using the DESCRIBE statement.

- The following example uses the DESCRIBE statement in a DATA step to write a copy of the source code for the data view Company.Newdata to the SAS log:

```
data view=company.newdata;
describe;
run;
```

- When you create a DATA step view, the following actions occur:

- The DATA step is partially compiled.
- The intermediate code is stored in the specified SAS library with a member type of VIEW.

Referencing a Data View Multiple Times in One Program

- SAS executes a view each time it is referenced, even within one program.
- If data is used many times in one program, it is more efficient to create and reference a temporary SAS data file than to create and reference a view.