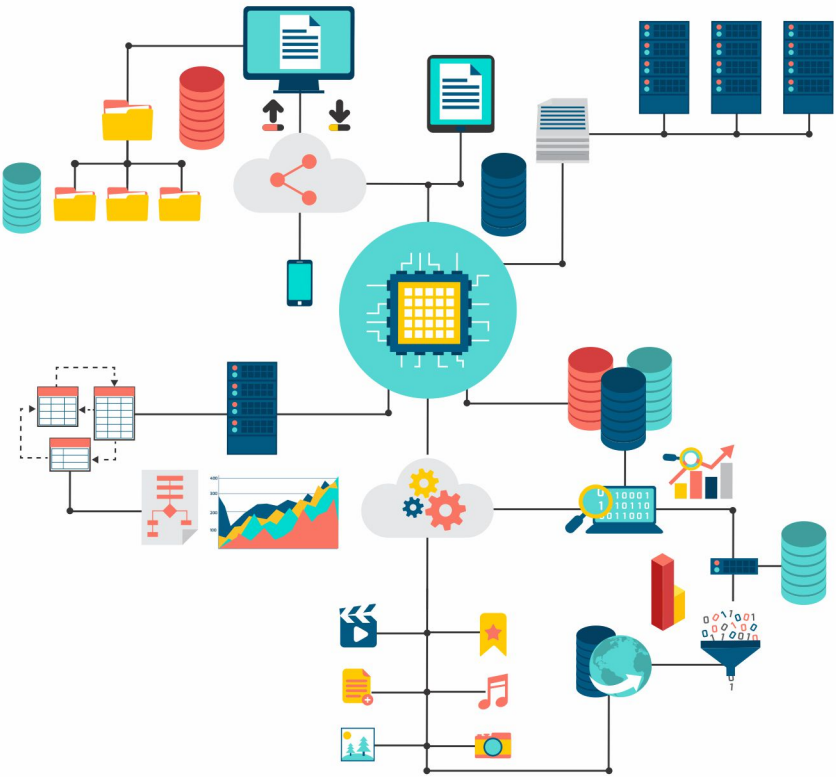
Job Interview Questions Series

SAS PROGRAMMING GUIDELINES

INTERVIEW QUESTIONS YOU'LL MOST LIKELY BE ASKED



293

Interview Questions

VIBRANT PUBLISHERS

Job Interview Questions Series

SAS PROGRAMMING GUIDELINES

INTERVIEW QUESTIONS YOU'LL MOST LIKELY BE ASKED

293
Interview Questions

VIBRANT
PUBLISHERS

SAS Programming Guidelines

Interview Questions You'll Most Likely Be Asked

ISBN-10: 1-946383-76-7
ISBN-13: 978-1-946383-76-1

Library of Congress Control Number: 2012920665

Table of Contents

Dear Reader,

Thank you for purchasing **SAS Programming Guidelines Interview Questions You'll Most Likely Be Asked.** We are committed to publishing books that are content-rich, concise and approachable enabling more readers to read and make the fullest use of them. We hope this book provides the most enriching learning experience as you prepare for your interview.

Should you have any questions or suggestions, feel free to email us at **reachus@vibrantpublishers.com**

Thanks again for your purchase. Good luck with your interview!

– Vibrant Publishers Team

SAS **Programming Guidelines** Interview Questions

Review these typical interview questions and think about how you would answer them. Read the answers listed; you will find best possible answers along with strategies and suggestions.

www.vibrantpublishers.com

This page is intentionally left blank

Chapter 1

Efficient SAS Programming

1: How do you achieve scalability in SAS programming?

Answer:

SAS program scalability can be achieved in 2 ways - by scaling up and scaling out. Scalability is ensuring the lowest time to solution, especially for the most vital tasks. Typically, when you want to speed up the task completion, you either try to complete multiple processes at the same time or distribute the task across various processors and do parallel processing. This, sometimes, involve overlapping of certain processes. Scaling up requires better hardware that is capable of multiprocessing which is known as symmetric multiprocessing or SMP. Scaling out requires more servers that can handle distributed processing.

2: How do SQL Views help better efficiency?

Answer:

A View typically consists of a subset of the entire table and hence is more efficient as it accesses a smaller set of data which is required. View also lets you hide the sensitive columns and complex queries from the user by choosing only what needs to be shown. Views always fetch fresh data from the table as they do not store any data.

3: What do you know about the SPD Engine?

Answer:

The SPD Engine or the SAS Scalable Performance Data Engine is developed for SAS 9 to speed up the processing of large data sets by splitting them into smaller physical files called partitions. There are several parallel processors that have exclusive access to each partition and process them in parallel using threads. Partitions are created when the SAS data sets are created. When a Where clause is mentioned it is split across the partitions and processed in parallel. Data blocks are also read in parallel. Multiple connections are created based on the partitions which further reduces the I/O bottlenecks. The SPD Engine also does an implicit sort if the query contains a by clause.

4: What resources are used to run a SAS program?

Answer:

The six resources used to run a SAS program are:

a) **Programmer time**: The amount of time taken by the programmer for writing, testing and maintaining the program

b) **Real time**: The time elapsed while executing a job

c) **CPU time**: The amount of time the CPU takes to perform a task. The task can be reading data, writing data, calculations or implementation of a logic

d) **Memory**: The work area memory space used for holding executable programs, data, etc

e) **Data storage space**: The disk space for storing the data. This is measured in terms of bytes, kilobytes, gigabytes etc.

f) **I/O**: The read and write operations performed to movie data from the memory to any output device, and vice versa

5: List the factors that need to be considered while assessing the technical environment.

Answer:

The four factors that need to be considered while assessing a technical environment are:

a) **Hardware**: Available memory, number of CPU's, number of devices connected, network bandwidth, I/O bandwidth, and capability to upgrade

b) **Operating environment**: The resource allocation & I/O methods

c) **System load**: This includes the number of users sharing the system, the network traffic, and the predicted increase in load

d) **SAS environment**: includes all SAS software products installed, number of CPU's, and memory allocated for SAS programming

6: Explain the functionality of the system option STIMER in the Windows environment.

Answer:

STIMER option in the Windows environment specifies that CPU time and real time statistics are tracked and written to the SAS log throughout the SAS session.

Example: The following line of code turns on the *STIMER* option.

options STIMER;

7: What is the function of the option FULLSTIMER in the Windows operating environment?

Answer:

FULLSTIMER option in the Windows environment specifies that all the available resource usage statistics needs to be tracked and written to SAS log throughout the SAS session.

Example:

options FULLSTIMER;

8: Explain the MEMRPT option.

Answer:

The *MEMRPT* option in the z/OS environment specifies that the memory usage statistics are tracked and written to SAS log throughout the SAS session. This is not available as a separate option in the Windows operating environment.

9: While benchmarking the programming techniques in SAS, why is it necessary to execute each programming technique in separate sessions?

Answer:

It is always necessary to execute each programming technique in separate SAS sessions while benchmarking them the first time a program is read because the operating system might load the code into the cache and retrieve it from the cache when it is referenced. This takes less time. The resource usage necessary to perform this action is referred to as *overhead*. Using separate sessions minimize the effect of overhead on resource statistics.

10: While doing benchmark tests, when is it advisable to run the code for each programming technique several times?

Answer:

It is advised to run the code for each programming technique several times while benchmarking tests if the system is executing other jobs at the same time. Running the code several times reduces variations in the resource consumption associated with the task and so the average resource usage is known.

11: How do you turn off the FULLSTIMER option?

Answer:

The *FULLSTIMER* option can be turned off with the following line of code.

options nofullstimer;

12: What steps can be taken to reduce the programmer time?

Answer:

Programmer time is the amount of time required for the programmer to determine the specifications, write, submit, test and maintain the program. It is difficult to calculate the exact time, but it can be reduced by the use of well-documented programming practices and reuse of SAS code modules.

Chapter 2

Memory Usage

13: What is PDV? How does it work?

Answer:

PDV or Program Data Vector is a memory area created after the input buffer is created. Two extra variables _N_ and _Error_ are created by the SAS engine during compilation. These variables are used for processing but never written into the data set. SAS creates a PDV for each observation.

14: How would you choose between Data Step and Proc SQL?

Answer:

With small data sets, Proc SQL works better since it loads the entire data set into the memory and works with the data. So there's less need to go back and forth into the database. But with large data sets Data Step will work better as loading the entire data set with Proc SQL will block a huge chunk of memory. Data

Step will always take one record at a time and hence, the number of records or large volume of data will not matter as long as the database connectivity remains good.

15: Explain memory management in SAS.

Answer:

SAS, unlike Java and .Net, does not have garbage collection for memory management. But it does accomplish the job with a series of instructions called steps. Memory is allocated when the step begins and released when the step completes. This way, there's no memory loosely allocated during the runtime. When dealing with large volumes of data, there may be cases when ample memory is not available. In such cases, SAS pushes an error message that memory not available, which is logged for reference. The hash objects in SAS lets you handle considerable amount of objects quickly. The Data Step is also efficient in memory management as it takes only one record at a time. Since most of the SAS programs depend upon a Work Area which they use to store objects temporarily, this area typically runs out of memory which needs to be handled efficiently.

16: What is the sequence of action performed in the background while trying to create a data set from another data set?

Answer:

While creating a data set from another data set the following actions take place in the background

a) The data gets copied from the input data set to a buffer in memory
b) From the input buffer an observation at a time is written to PDV (Program Data Vector)

c) Each observation from PDV is written to output buffer when processing is complete

d) The contents of output buffer are written to disk when the buffer is full.

17: Define PAGE and PAGESIZE.

Answer:

A *PAGE* is a unit that indicates the data transfer between a storage device and *PAGESIZE* is the amount of data that can be transferred to one buffer in a single I/O operation.

18: What procedure is used to indicate the PAGESIZE of a data set?

Answer:

The *Contents* procedure is used to know the PAGESIZE associated with a data set.

Example: The following Contents procedure issues a message to SAS log indicating the PAGESIZE associated with the data set *exam.clinic1*. This also gives the number of data set pages.

Proc contents data = exam.clinic1;

run;

19: Is it possible to control the PAGESIZE of an output data set?

Answer:

It is possible to control the PAGESIZE of an output data set by using *BUFSIZE*= option, which specifies the PAGESIZE in bytes

Example: The following program creates a data set *exam.clinic1* from the data set *exam.clinic2*. In the following program the *BUFSIZE*= option specifies a PAGESIZE of 30720 bytes.

```
options bufsize=30720;
libname exam 'c:\myprog';
data exam.clinic1
   set exam.clinic2;
run;
```

20: What is the default value of the BUFSIZE= option?

Answer:

The default value of the *BUFSIZE*= option is 0. If *BUFSIZE*= option is set to zero SAS uses the optimal PAGESIZE determined by SAS for that operating environment.

21: Is it necessary to specify the BUFSIZE= option every time a data set is processed?

Answer:

No. The *BUFSIZE*= option is set at the time of creation of data set, and that value of becomes a permanent attribute of the data set. Once it is specified it is used every time the data set is processed.

22: What does the BUFNO= option signify?

Answer:

The *BUFNO* = option is used along with an SAS data set to lay down how many buffers are available for reading, writing, or updating. The larger the value of BUFNO = the faster the input/output function would be since more values will be stored in the buffer which avoids an actual input/output function. You can specify larger number of pages to include in the *BUFNO* = and accordingly that many pages will be loaded into the memory.

For Example, the following program creates a data set

MyExam.MyClinic from the data set *MyExam.MyClinic2* in the following program, the BUFNO = option is given the value 6, that denotes 6 buffers.

```
options bufno=6;
libname exam 'D:\MyProgram';
data MyExam.MyClinic
  set MyExam.MyClinic2;
run;
```

23: How do you set the BUFNO= option to the maximum possible number?

Answer:

To set the maximum value to *BUFNO*= option, you can set BUFNO= MAX which sets the maximum buffer value available in the current operating environment. The largest possible value of max would be approximately 2 billion (231-1).

For Example, the following program creates a data set *MyExam.MyClinic1* from the data set *MyExam.MyClinic2*. In the following program, the *BUFNO* = option is given the value max, that denotes the maximum buffer available in the current environment.

```
options bufno=max;
libname exam 'D:\MyProgram';
data MyExam.MyClinic
  set MyExam.MyClinic2;
run;
```

24: Is it necessary to specify the BUFNO= option every time **a data set is processed?**

Answer:

It is mandatory to specify the BUFNO = option every time a data set is processed. This is required since the buffer varies every time a data set is opened and closed. Moreover, the BUFNO = value set is valid only while a data set is open in the current session.

25: What are the general guidelines for specifying the buffer size and buffer number in the case of small data sets?

Answer:

The main objective behind specifying the buffer size and buffer number is to reduce the number of I/O operations. In the case of small data sets, care must always be taken to allocate as many buffers as there are pages in the data set. This ensures that the entire data set can be loaded into the memory using a single I/O operation.

26: How does the BUFSIZE= and BUFNO= impact the following program?

data exam.clinic1 (BUFSIZE=12288 BUFNO=10);

set exam.clinic2;

run;

Answer:

The above program reads the data set *exam.clinic2* and creates *exam.clinic1*. The *BUFSIZE*= option specifies that *exam.clinic1* is created with a buffer size of 12288 bytes. The *BUFNO*= option specifies that 10 pages of data are loaded into memory with each I/O transfer.

27: Explain the SASFILE statement.

Answer:

The *SASFILE statement* loads the SAS data file into the memory to be available further to the program. With SAS File you can free the buffers. Instead, the file is loaded and kept in the system memory with a pointer in the program to access it.

The following example explains the use of *SASFILE* in a simple way. The *SASFILE* statement opens the data set *MyExam.MyClinic* and allocates the buffer. It reads the file and loads it into the memory so that it is available to both the *PROC print* as well as the *PROC MEANS* step. Finally, the *SASFILE* data file is closed with the close statement and the buffer is cleared.

```
SASFILE MyExam.MyClinic load;
proc print data= MyExam.MyClinic
   var. Serial No result;
run;
proc means data= MyExam.MyClinic;
run;
SASFILE MyExam.MyClinic close;
```

28: What happens if the size of the file in the memory increases during the execution of SASFILE statement?

Answer:

When the *SASFILE statement* is executed, SAS assigns some buffer to the DATAFILE based on the number of pages to be loaded and the size of the index file. Once this is done, the file data is loaded into the memory for updates. The buffer size is automatically increased as the file size to be saved increases. The initial buffer memory size allocated is only the minimum memory allocated to

load the file. It automatically increases provided there's ample memory left in the current operating system.

29: Mention the guidelines to be followed while using SASFILE statement.

Answer:

While using the *SASFILE* statement, the following procedures are to be followed:

a) There should be sufficient real memory to load the file

b) In case, there's not enough memory to load the entire file into one SAS data set, the data step should be used to create a subset of the file which will fit into the available memory. Since one part of the file is already loaded into the memory, the rest of the file data can also be easily accessed by the program. This reduces the CPU time significantly.

30: When is the buffer allocated by the SASFILE statement freed?

Answer:

The buffer allocated by the *SASFILE* statement to load the data file is freed in two instances:

a) When the *SASFILE Close* statement is executed, the file is closed, and the buffer allocated for the DATAFILE is closed. For *example*, In the following program the SASFILE statement opens the data set. *MyExam.MyClinic* and allocates the buffer. It reads the data into the memory which is available to the PROC print and PROC MEANS steps. The last SASFILE statement closes the SAS DATAFILE and frees the buffer allocated for the file.

SASFILE MyExam.MyClinic load;

proc print data= MyExam.MyClinic

var Serial No result;

run;

proc means data= MyExam.MyClinic;

run;

SASFILE MyExam.MyClinic close;

b) The SASFILE buffer is allocated only as long as the session is open. When SAS session ends, it frees the buffer and closes the DATAFILE.

31: Which operations are not allowed in a file opened with SASFILE statement?

Answer:

There are certain operations that cannot be performed on a file opened with *SASFILE statement, such as* replacing the file and renaming the variables.

32: How do you calculate the total number of bytes occupied by a data file if you know the PAGESIZE?

Answer:

The total number of bytes that a data file occupies can be calculated by multiplying the PAGESIZE by the number of pages.

Example: If the data file *exam.clinic1* has a PAGESIZE of 8192 and number of pages is 900, then the data file occupies 7372800 bytes (8192* 9423).

This page is intentionally left blank

Job Interview Questions Series

SAS PROGRAMMING GUIDELINES

INTERVIEW QUESTIONS YOU'LL MOST LIKELY BE ASKED

SAS Programming Guidelines Interview Questions You'll Most Likely Be Asked is a perfect companion to stand ahead above the rest in today's competitive job market. Rather than going through comprehensive, textbook-sized reference guides, this book includes only the information required immediately for job search to build an IT career. This book puts the interviewee in the driver's seat and helps them steer their way to impress the interviewer.

The following is included in this book:

- 215 SAS Programming Guidelines Interview Questions, Answers and proven strategies for getting hired as an IT professional
- Dozens of examples to respond to interview questions
- 78 HR Questions with Answers and proven strategies to give specific, impressive, answers that help nail the interviews
- 2 Aptitude Tests download available on www.vibrantpublishers.com

VIBRANT
PUBLISHERS

www.vibrantpublishers.com

ISBN 978-1-946383-76-1
9 781946 383761
90000>