**SAS Competency Test**

**Part A:**

***18 Questions - Total Marks – 50 (Duration – 45 minutes)***

1. How SAS processes the data at the back end? **(2 Marks)**

**Ans -**

a.Compilation Phase

Step I : Syntax Checking

SAS scans each statement in the DATA step and check syntax errors, such as missing semicolons and invalid statements.

Step II : Creating Input Buffer

If you read in a raw data set such as txt or csv file, the input buffer is created. The input buffer is used to hold raw data. If you read in a SAS data set, the input buffer will not be created.

Step III : Creating Program Data Vector (PDV)

1. SAS creates a program data vector (memory on your system) containing the automatic variables \_N\_ and \_ERROR\_.

How PDV works

\_N\_ = 1 indicates the first observation is being processed, \_N\_ = 2 indicates the second observation is being processed, and so on.

The automatic variable \_ERROR\_ with values of 1 or 0, if it is equal to 1 signals the data error of the currently-processed observation, such as reading the data with an incorrect data type.

2. In addition to the two automatic variables, there is one space allocated for each of the variables in the input statement (reading data).

3. SAS also adds a position to the program data vector for any variables that are created in the DATA step.

4. If the variables specified in the DROP statement, it will never be written to the output data set.

5. At the end of the compilation phase, SAS makes the descriptor portion of the SAS data set which includes the data set name, the number of observations, and the number, names, and attributes of variables.

b.Execution Phase

Sequential Processing (Iterative)

1. At the beginning of the execution phase, SAS sets all of the data set variables in the program data vector to missing:

2. The SET statement reads the first observation from the input data set and writes the values to the program data vector.

3. Compute the first value for the derived variable.

4. At the end of the first iteration of the DATA step, the values in the program data vector are written to the output data set as the first observation.

1. What are the automatic variables created in PDV? And when their values get updated for the respective automatic variable? **(2 Marks)**

Ans -SAS creates a program data vector (memory on your system) containing the automatic variables \_N\_ and \_ERROR\_.

How PDV works

* \_N\_ = 1 indicates the first observation is being processed, \_N\_ = 2 indicates the second observation is being processed, and so on.
* The automatic variable \_ERROR\_ with values of 1 or 0, if it is equal to 1 signals the data error of the currently-processed observation, such as reading the data with an incorrect data type.

1. The below data is saved in a text file:

|  |  |  |
| --- | --- | --- |
| Name | Age | Gender |
| Amit | 18 |  |
| Kushal |  | M |
| Nakul | 20 | F |
| Sukhpreet | 27 | F |

Please write a SAS code to convert the text data (list data) into SAS data file. (The data values are separated with blanks) **(4 Marks)**

Ans –

filename two 'C:\Users\sasuser\Desktop\CIA\sample1.txt'; /\* this is raw data file \*/

**data** new2;

infile two DSD delimiter='09'x ;

input name$ age gender$ ;

**run**;

1. Difference between IF and WHERE statement. Which is more efficient and why? **(3 Marks)**

**Ans -**

* WHERE statement can be used in procedures to subset data while IF statement cannot be used in procedures.
* WHERE can be used as a data set option while IF cannot be used as a data set option.
* WHERE statement is more efficient than IF statement. It tells SAS not to read all observations from the data set
* WHERE statement can be used to search for all similar character values that sound alike while IF statement cannot be used.
* WHERE statement can not be used when reading data using INPUT statement whereas IF statement can be used.
* Multiple IF statements can be used to execute multiple conditional statements
* When it is required to use newly created variables, use IF statement as it doesn't require variables to exist in the READIN data set.

The WHERE statement is more efficient than IF statement. It tells SAS not to read all observations from the data set.

1. Can we calculate the sum of a Salary variable stored in WORK.XYZ dataset using data step? If so, please write an appropriate code to get the output. **(2 Marks)**

Data xyz;

input employee\_id$ month $ salary ;

cards;

a jan 14000

b jan 30000

b feb 45000

b mar 20000

b apr 18000

a feb 16000

a mar 40000

;

proc sort data=xyz;

by employee\_id ;

run;

data xyz2;

set xyz;

retain salcum;

by employee\_id ;

if first.employee\_id then salcum = salary;

else salcum = salcum + salary;

if last.employee\_id then output;

drop salary;

run;

1. Write a sample code to get the cumulative SUM of salary variable in WORK.XYZ dataset. **(3 Marks)**

Data xyz;

input employee\_id$ month $ salary ;

cards;

a jan 14000

b jan 30000

b feb 45000

b mar 20000

b apr 18000

a feb 16000

a mar 40000

;

proc sort data=xyz;

by employee\_id ;

run;

data c;

set xyz;

by employee\_id;

if first.employee\_id then cumsal = 0;

cumsal+salary;

run;

data xyz1;

set xyz;

retain salcum;

by employee\_id ;

if first.employee\_id then salcum = salary;

else salcum = salcum + salary;

run;

1. Difference between PROC Freq, PROC Means and PROC Summary. **(3 Marks)**

Proc SUMMARY and Proc MEANS are essentially the same procedure. Both procedures compute descriptive statistics. The main difference concerns the default type of output they produce. Proc MEANS by default produces printed output in the LISTING window or other open destination whereas Proc SUMMARY does not. Inclusion of the print option on the Proc SUMMARY statement will output results to the output window.

The second difference between the two procedures is reflected in the omission of the VAR statement. When all variables in the data set are character the same output: a simple count of observations, is produced for each procedure. However, when some variables in the dataset are numeric, Proc MEANS analyses all numeric variables not listed in any of the other statements and produces default statistics for these variables (N, Mean, Standard Deviation, Minimum and Maximum).

But in case of Proc Freq in can only produce frequency distributions .

1. There are four variables (A, B, C & D) in PQR dataset. How can we get the cross-tabulation of A & D variables? **(2 Marks)**

Ans –

Proc freq data=pqr;

Tables a\*b;

Run;

1. There is a variables Age in SAS dataset i.e. “Demo”. Please create a new variable Age\_group to create different buckets for different age groups as shown below. Please do this task using data step as well as PROC SQL. **(4 Marks)**

|  |
| --- |
| Age |
| 18 |
| 19 |
| 26 |
| 32 |
| 45 |

|  |  |
| --- | --- |
| Age | Age\_Group |
| 18 | 0-18 |
| 19 | 19-25 |
| 26 | 26-35 |
| 32 | 26-35 |
| 45 | 35+ |

Ans –

Data demo;

input age;

cards;

18

19

26

32

45

;

Proc sql;

create table abc as select \*,

case

when 0<=age<=18 then "0-18"

when 19<=age<=25 then "19-25"

when 26<=age<=35 then "26-35"

when age>35 then "35+"

end as age\_group format=$10.

from demo;

run;

Data avc;

set demo;

format age\_group $10.;

if 0<=age<=18 then age\_group="0-18";

else if 19<=age<=25 then age\_group="19-25";

else if 26<=age<=35 then age\_group="26-35";

else age\_group= "35+";

run;

1. Difference between DATA merge and PROC SQL joins **(5 Marks)**

Ans -There is difference in data step and sql processing. Data merging works sequentially checks each observation of each data set to match these by BY value to write it to combined dataset and same name variables in the first data set are overwritten with the liked named variable in subsequent data sets . SQL processing is based on cartesian product where processing moves horizontally and same name variables coming from different data sets can not be overwritten.

1. There is a SAS dataset i.e. “Demo”. We want to know about total IDs and unique IDs in this particular table. Please write a sample code to get this information. **(3 Marks)**

proc sql;

select count(id) as total\_id,count(distinct id) as unq\_id

from demo;

quit;

1. Difference between Keyword and Positional parameters in SAS Macros. **(2 Marks)**

Ans –

When we define the parameters to our macro we can make one or more of the parameters positional.  All positional parameters must come at the beginning of the parameter list, preceding any keyword parameters. We should only define parameters as positional when there usage is obvious from the purpose of the macro. When we call the macro we can specify the value for the positional parameters without using their names. The advantage of the keyword parameters is that we can specify a default value in the macro definition.

1. There is a variables Age in SAS dataset i.e. “Demo”. We want to store all the Age values in different macro variables. Write a SAS code to store these values. **(5 Marks)**

|  |
| --- |
| Age |
| 18 |
| 19 |
| 26 |
| 32 |
| 45 |

Ans -

proc sql;

select distinct age into : age1 - : age5 from demo;

quit;

1. For what purpose would you use the RETAIN statement? **(2 Marks)**

Ans –

At the beginning of each iteration of the DATA step, SAS replaces variables values under program data vector with missing values. A RETAIN statement overrides this default values. RETAIN statement tells SAS not to set variables with missing values when going from the current iteration of the DATA step to the next i.e. SAS retains the values.

1. How do you control the number of observations read or written? **(2 Marks)**

Ans – Using “Obs” and ” firstobs” option in sas

1. Which function is used to remove the following blanks: **(2 Marks)**
   1. Leading
   2. Trailing
   3. Leading and trailing both

Ans –

1. Leading - LEFT
2. Trailing - TRIM
3. Leading and trailing both - STRIP
4. What are the different purposes of SUBSTR function? **(2 Marks)**

**Ans -**

SUBSTR:

It extracts strings based on character position and length.

= substr(old\_var, starting\_position, number of characters to keep);

Examples :

t="AFHood Analytics Group";

new\_var=substr(t,8,9);

Result : Analytics

SUBSTR(Left of =) Function

It replaces a portion of string with new string

substr("old\_variable",1,8) = new\_data;

Result : New\_dataable

1. Purpose of ROUND function and its syntax. **(2 Marks)**

Ans - Round function can round a number up, round it down, or round it to the nearest integer

Data ROUND;

INPUT NUM;

FORMAT NUM 15.5;

d1 = round(NUM,100);

d2 = round(NUM,10);

d3 = round(NUM,1);

d4 = round(NUM,.1);

d5 = round(NUM,.01);

datalines;

12345.67890

;

proc print data=round;run;

Gives the following SAS output results:

Obs NUM d1 d2 d3 d4 d5

1 12345.67890 12300 12350 12346 12345.7 12345.68

**Part B:**

***3 Questions - Total Marks – 25 (Duration – 1 Hr 15 minutes)***

**Instructions**

* Candidate needs to create the SAS datasets for the tables given in all the exercises below.

1. In the below tables, information about owners and their field information is given. Use DATA STEP to do the following:
   * 1. Generate a separate data set containing owner name, place, location, Yield1 and Yield2, and sum of Yield1 and Yield2 **(10 Marks)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Yield Data** | | | |  |  | **Ownership Data** | | |
| **Place** | **Location** | **Yield1** | **Yield2** |  |  | **Owner** | **Place** | **Location** |
| A | A | 2.95 | 51.30 |  |  | Hrshikesh | a | g |
| B | D | 39.17 | 8.09 |  |  | Teddy | d | k |
| C | F | 26.43 | 97.51 |  |  | Srinu | g | t |
| D | G | 56.53 | 64.05 |  |  | Rajpal | e | l |
| G | T | 52.25 | 35.33 |  |  | Raman | a | a |
| E | H | 80.43 | 66.39 |  |  | Kapil | a | n |
| A | G | 15.70 | 11.76 |  |  | Rakesh | b | d |
| B | F | 63.66 | 25.44 |  |  | Yuvi | c | f |
| C | R | 78.84 | 0.04 |  |  | Sibel | c | r |
| D | D | 12.75 | 0.14 |  |  | Yukta | d | t |
| G | T | 64.67 | 68.04 |  |  | Srinivas | d | g |
| E | Z | 41.20 | 73.02 |  |  | Dev | g | t |
| D | K | 5.17 | 23.51 |  |  | Vishwam | e | z |
| G | J | 92.16 | 11.43 |  |  | Mayank | b | m |
| E | L | 62.50 | 73.60 |  |  | Vivek | b | f |
| A | N | 3.17 | 84.54 |  |  | Vaibhav | c | r |
| B | M | 52.31 | 4.15 |  |  | Pawan | d | d |
| C | R | 39.22 | 16.89 |  |  | Mukul | g | j |
| D | T | 26.93 | 33.12 |  |  | Baljeet | e | H |

Ans –

Proc sort data=ownership;

by location place;

run;

data ownership;

set ownership;

place=lowcase(place);

location=lowcase(location);

run;

Proc sort data=field\_yield;

by location Place;

run;

data field\_yield;

set field\_yield;

place=lowcase(place);

location=lowcase(location);

run;

data merged;

merge ownership(in=a) field\_yield(in=b);

by location place;

if a;

sum\_field\_yield=sum(field\_yield1+field\_yield2);

run;

1. Names.sas file has 2 fields – name\_id and earning from different resources. Please note that each row corresponds to earning by person from a unique resource. Therefore a name\_id can have multiple earnings.
2. Write a query to store a list of unique name\_id in file unique.csv from names.sas **(3 Marks)**
3. Find out the average earning per person (name\_id) **(2 Marks)**

**Ans – Assuming names.sas file have been kep in abc library**

**a)**

**Proc sql;**

**create table uniq as select distinct name\_id from abc.names ;**

**quit;**

**data uniq1(keep=name\_id);**

**set abc.names;**

**by name\_id;**

**if first.name\_id;**

**run;**

**proc export data=uniq1**

**outfile='C:\Users\sasuser\Desktop\Anirban\sas test\unique.csv'**

**dbms=dlm;**

**delimiter=' ';**

**run;**

**b)**

**Proc sql;**

**create table average as select name\_id,avg(earning)as avg\_earning**

**from abc.names**

**group by name\_id;**

**quit;**

1. There are two files - File One contains information about id, name and File Two contains information about id, age and sex as shown in Table 1 and Table 2 below. Write a program to merge the two files to obtain the desired result as shown in Table 3 **(10 Marks)**

**Table1 FILE ONE**

|  |  |
| --- | --- |
| ID | NAME |
| A01 | SUE |
| A02 | TOM |
| A05 | KAY |
| A10 | JIM |

**Table2 FILE TWO**

|  |  |  |
| --- | --- | --- |
| ID | AGE | SEX |
| A01 | 58 | F |
| A02 | 20 | M |
| A04 | 47 | F |
| A10 | 11 | M |

**Table3 DESIRED OUTPUT**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | AGE | SEX |
| A01 | SUE | 58 | F |
| A02 | TOM | 20 | M |
| A04 |  | 47 | F |
| A10 | JIM | 11 | M |

**Table4 DESIRED OUTPUT**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | AGE | SEX |
| A01 | SUE | 58 | F |
| A02 | TOM | 20 | M |
| A10 | JIM | 11 | M |

**Table 5 DESIRED OUTPUT**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | AGE | SEX |
| A04 |  | 47 | F |

**Table 6 DESIRED OUTPUT**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | AGE | SEX |
| A05 | KAY | . | . |

Ans –

proc sort data=one;

by id;

run;

proc sort data=two;

by id;

run;

\*table 3;

data merged1;

merge one(in=a) two(in=b);

by id;

if b;

run;

\*table 4;

data merged2;

merge one(in=a) two(in=b);

by id;

if a and b;

run;

\*table 5;

data merged3;

merge one(in=a) two(in=b);

by id;

if b and not a;

run;

\*table 6;

data merged4;

merge one(in=a) two(in=b);

by id;

if a and not b;

run;