**1. Difference between INPUT and INFILE**  
  
The **INFILE statement** is used to identify an external file while the **INPUT statment** is used to describe your variables.

FILENAME TEST 'C:\DEEP\File1.xls';

DATA READIN;

**INFILE TEST;**

LENGTH NAME $25;

**INPUT ID NAME$ SEX;**

RUN;

Note : The variable name, followed by **$** (dollar sign), idenfities the variable type as **character**.  
  
In the example shown above, ID and SEX are numeric variables and Name a character variable.

**2. Difference between Informat and Format**  
  
**Informats** read the data while **Formats** write the data.  
  
**Informat -** To tell SAS that a number should be read in a particular format.  
  
For example: the informat mmddyy6. tells SAS to read the number 121713 as the date December 17, 2013.  
  
**Format -** To tell SAS how to print the variables.

**3. Difference between Missover and Truncover**

**Infile ‘c://abcd/a.txt ;**

**Test.txt**

**1 raman male**

**2 jatin**

**3**

**4 chahat female**

**Missover** - When the MISSOVER option is used on the INFILE statement, the INPUT statement does not jump to the next line when reading a short line. Instead, MISSOVER sets variables to missing.  
  
**Truncover** - It assigns the raw data value to the variable even if the value is shorter than the length that is expected by the INPUT statement.  
  
**The following is an example of an external file that contains data:**  
  
1  
22  
333  
4444  
  
This DATA step uses the numeric informat 4. to read a single field in each record of raw data and to assign values to the variable ID.

data readin;

infile 'external-file' **missover**;

input ID4.;

run;

proc print data=readin;

run;

**The output is shown below :**

Obs ID

1 .

2 .

3 .

4 4444

**Truncover**

data readin;

infile 'external-file' **truncover**;

input ID4.;

run;

proc print data=readin;

run;

**The output is shown below :**

Obs ID

1 1

2 22

3 333

4 4444

**4. Purpose of double trailing @@ in Input Statement ?**  
  
The double trailing sign (@@) tells SAS rather than advancing to a new record, hold the current input record for the execution of the next INPUT statement.

DATA Readin;

Input Name $ Score @@;

cards;

Sam 25 David 30 Ram 35

Deeps 20 Daniel 47 Pars 84

;

RUN;

The output is shown below :

|  |
| --- |
| <https://4.bp.blogspot.com/-Uq6WSfwDKL0/WJ7U8bsMKmI/AAAAAAAAF4Y/YSRPCJ0qiEIPl_o1Gpa8MX3WvRmuTWZhQCLcB/s1600/Sascert2.png> |
| Double Trailing |

**5. How to include or exclude specific variables in a data set?**  
  
- DROP, KEEP Statements and Data set Options  
  
**DROP, KEEP Statement**  
  
The **DROP statement** specifies the names of the variables that you want to remove from the data set.

data readin1;

set readin;

**drop score;**

run;

The **KEEP statement** specifies the names of the variables that you want to retain from the data set.

data readin1;

set readin;

**keep var1;**

run;

**DROP, KEEP Data set Options**

The main difference between **DROP/ KEEP statement** and **DROP=/ KEEP=data set option** is that you can not use DROP/KEEP statement in procedures.

data readin1 (**drop=score**);

set readin;

run;

data readin1 (**keep=var1**);

set readin;

run;

**6. How to print observations 5 through 10 from a data set?**  
  
The **FIRSTOBS=** and **OBS=** data set options would tell SAS to print observations 5 through 10 from the data set READIN.

proc print data = readin **(firstobs=5 obs=10)**;

run;

**7. What are the default statistics that PROC MEANS produce?**  
  
PROC MEANS produce the “default” statistics of N, MIN, MAX, MEAN and STD DEV.

**8. Name and describe functions that you have used for data cleaning?**

|  |
| --- |
| <https://1.bp.blogspot.com/--qnV-GyBrnI/Vxyb3moK0DI/AAAAAAAAEN8/HaHVJGxc7kk3SJDhbDXTA5ebCKLST5hWgCKgB/s1600/Functions_6960f6474527c3159903491005b5c8ed.png> |
| SAS Character Functions |

**9. Difference between FUNCTION and PROC**  
  
Example : MEAN function and PROC MEANS  
  
  
The MEAN function is an average of the value of several variables in one observation.  
  
The average that is calculated for a PROC MEANS is the sum of all of the values of a variable divided by the number of observations in the variable.  
  
In other words, The MEAN function will SUM across the row and a procedure will SUM down a column.  
  
**MEAN Function**

*AVG=MEAN (of Q1 - Q3);*

**See the output below :**

|  |
| --- |
| <https://4.bp.blogspot.com/-b3gZe8q4toQ/WJ7V8NirVQI/AAAAAAAAF4k/dSGeOeFQ884H6nGm6wPWYqwM7aKeTKmrgCLcB/s1600/sascert9.png> |
| MEAN Function Output |

**PROC MEANS**

*PROC MEANS DATA=READIN* ***MEAN****;  
RUN;*

**The output is shown below :**

|  |
| --- |
| <https://2.bp.blogspot.com/-YyHRRzuef0I/WJ7Vcxpuu0I/AAAAAAAAF4c/eN3SbVv4HDo6XquKPGFdGQRQDQgYKdKYQCLcB/s1600/sascert11.png> |
| PROC MEANS Output |

**10. Differences between WHERE and IF statement?**  
  
**WHERE statement** can be used in procedures to subset data while **IF statement** cannot be used in procedures.

1. **WHERE** can be used as a data set option while **IF** cannot be used as a data set option.
2. **WHERE statement** is more efficient than **IF statement**. It tells SAS not to read all observations from the data set
3. **WHERE statement**can be used to search for all similar character values that sound alike while **IF statement** cannot be used.
4. **WHERE statement** can not be used when reading data using INPUT statement whereas**IF statement**can be used.
5. Multiple**IF statements**can be used to execute multiple conditional statements
6. 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**.**

**11. What is Program Data Vector (PDV)?**

PDV is a logical area in the memory.  
  
**How PDV is created?**

SAS creates a dataset one observation at a time. Input buffer is created at the time of compilation, for holding a record from external file.***PDV is created followed by the creation of input buffer***. SAS builds dataset in the PDV area of memory.

**12. What is DATA \_NULL\_?**

The DATA \_NULL\_ is mainly used to create macro variables. It can also be used to write output without creating a dataset. The idea of "null" here is that we have a data step that actually doesn't create a data set.

**13. What is the difference between '+' operator and SUM function?**

 SUM function returns the sum of non-missing arguments whereas “+” operator returns a missing value if any of the arguments are missing.  
  
Suppose we have a data set containing three variables - X, Y and Z. They all have missing values. We wish to compute sum of all the variables.

*data mydata2;  
set mydata;****a=sum(x,y,z)****;****p=x+y+z****;  
run;*

The output is shown in the image below :

|  |
| --- |
| <https://2.bp.blogspot.com/-TxWqpxKGZtU/WJ7REIXZ-HI/AAAAAAAAF4I/tyh0kJMnLCgQMxLhcjE5PytVqdDicpBhgCLcB/s1600/dataset2.png> |
| SAS : SUM Function vs Plus Operator |

In the output, value of p is missing for 4th, 5th and 6th observations.

**14. How to identify and remove unique and duplicate values?**

1. Use **PROC SORT** with **NODUPKEY** and **NODUP** Options.  
2. Use **First. and Last.**   
  
The detailed explanation is shown below :  
  
**SAMPLE DATA SET**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Score** |
| 1 | David | 45 |
| 1 | David | 74 |
| 2 | Sam | 45 |
| 2 | Ram | 54 |
| 3 | Bane | 87 |
| 3 | Mary | 92 |
| 3 | Bane | 87 |
| 4 | Dane | 23 |
| 5 | Jenny | 87 |
| 5 | Ken | 87 |
| 6 | Simran | 63 |
| 8 | Priya | 72 |

**Create this data set in SAS**

*data readin;  
input ID Name $ Score;  
cards;  
1 David 45  
1 David 74  
2 Sam 45  
2 Ram 54  
3 Bane 87  
3 Mary 92  
3 Bane 87  
4 Dane 23  
5 Jenny 87  
5 Ken 87  
6 Simran 63  
8 Priya 72;  
run;*

There are several ways to identify and remove unique and duplicate values:  
  
**PROC SORT**  
  
In **PROC SORT**, there are two options by which we can remove duplicates.  
  
**1. NODUPKEY Option               2. NODUP Option**  
  
The **NODUPKEY** option removes duplicate observations where value of a variable listed in **BY** statement is repeated while **NODUP** option removes duplicate observations where values in **all the variables** are repeated (identical observations).

*PROC SORT DATA=readin* ***NODUPKEY****;  
BY ID;  
RUN;*

*PROC SORT DATA=readin* ***NODUP****;  
BY ID;  
RUN;*

The output is shown below :

|  |
| --- |
| <https://1.bp.blogspot.com/-_fAKht_aT48/Vxyf8uMfD6I/AAAAAAAAEOQ/1ywmUqPbKf8aeUAJQW7lw21kbJqyQkaCACLcB/s1600/nodup_b648493f5ff8b01e6456ce38f060e7f4.png> |
| SAS : NODUPKEY vs NODUP |

The **NODUPKEY** has deleted 5 observations with duplicate values whereas **NODUP** has not deleted any observations.  
 **Why no value has been deleted when NODUP option is used?**  
  
Although ID 3 has two identical records (See observation 5 and 7), NODUP option has not removed them. It is because they are not next to one another in the dataset and SAS only looks at one record back.  
  
To fix this issue, sort on all the variables in the dataset **READIN**.   
To sort by all the variables without having to list them all in the program, you can use the keywork ‘\_ALL\_’ in the BY statement (see below).  
  
  
The output is shown below :



|  |
| --- |
| <https://3.bp.blogspot.com/-Q8vYkr9ChBw/VxygJrSOpzI/AAAAAAAAEOU/vkR3ed7Xtdo1T2ZGiphHLBx_5SpsK6QDQCLcB/s1600/readin1.png-duplicate_d59159cac72021dd4941681b868fc4eb.png> |
| SAS NODUP Output |

**15. Difference between NODUP and NODUPKEY Options?**

The **NODUPKEY** option removes duplicate observations where value of a variable listed in BY statement is repeated while **NODUP** option removes duplicate observations where values in all the variables are repeated (identical observations).  
  
***See the detailed explanation for this question above (Q14).***

**16. What are \_numeric\_ and \_character\_ and what do they do?**

1. **\_NUMERIC\_** specifies all numeric variables that are already defined in the current DATA step.  
2. **\_CHARACTER\_** specifies all character variables that are currently defined in the current DATA step.  
3. **\_ALL\_** specifies all variables that are currently defined in the current DATA step.  
  
**Example :** To include all the numeric variables in PROC MEANS

*proc means;  
var \_numeric\_;  
run;*

**17. How to sort in descending order?**

Use **DESCENDING** keyword in PROC SORT code. The example below shows the use of the descending keyword.

PROC SORT DATA=auto;

  BY **DESCENDING** engine ;

RUN ;

**18. Under what circumstances would you code a SELECT construct instead of IF statements?**

When you have a long series of mutually exclusive conditions and the comparison is numeric, using a SELECT group is slightly more efficient than using IF-THEN or IF-THEN-ELSE statements because CPU time is reduced.  
  
**The syntax for SELECT WHEN is as follows :**  
  
SELECT (condition);  
   WHEN (1) x=x;  
   WHEN (2) x=x\*2;  
   OTHERWISE x=x-1;  
END;  
  
**Example :**  
  
SELECT (str);  
   WHEN ('Sun') wage=wage\*1.5;  
   WHEN ('Sat') wage=wage\*1.3;  
   OTHERWISE DO;  
wage=wage+1;  
bonus=0;  
END;  
END;

**19. How to convert a numeric variable to a character variable?**

You must create a differently-named variable using the **PUT** function.

The example below shows the use of the PUT function.

charvar=put(numvar, 7.) ;

**20. How to convert a character variable to a numeric variable?**

You must create a differently-named variable using the **INPUT**function.

The example below shows the use of the INPUT function.

numvar=input(charvar,4.0);

**21. What's the difference between VAR A1 - A3 and VAR A1 -- A3?**

**Single Dash :** It is used to specify consecutively numbered variables. A1-A3 implies A1, A2 and A3.

**Double-dash :**It is used to specify variables based on the order of the variables as they appear in the file, regardless of the name of the variable. A1--A3 implies all the variables from A1 to A3 in the order they appear in the data set.

**Example :** The order of variables in a data set  :  ID Name A1 A2 C1 A3

So using A1-A3 would return A1 A2 A3.

A1--A3 would return A1 A2 C1 A3.

**22. Difference between PROC MEANS and PROC SUMMARY?**

1. **Proc MEANS** by default produces printed output in the OUTPUT window whereas **Proc SUMMARY** does not. Inclusion of the PRINT option on the Proc SUMMARY statement will output results to the output window.

2. Omitting the var statement in **PROC MEANS** analyses all the numeric variable whereas Omitting the variable statement in **PROC SUMMARY** produces a simple count of observation.

**How to produce output in the OUTPUT window using PROC SUMMARY?**

*Use PRINT option.*

proc summary data=retail **print**;

  class services;

  var investment;

run;

**23. Can PROC MEANS analyze ONLY the character variables?**

No, Proc Means requires at least one numeric variable.

**24. How SUBSTR function works?**

The SUBSTR function is used to extract substring from a character variable.

***The SUBSTR function has three arguments:***

**SUBSTR ( character variable**, **starting point to begin reading the variable**, **number of characters to read from the starting point)**

**There are two basic applications of the SUBSTR function:**

**RIGHT SIDE APPLICATION**

data \_null\_ ;

 phone='(312) 555-1212' ;

 area\_cd=**substr(phone, 2, 3)** ;

put area\_cd=;

run;

**Result :** In the log window, it writes **area\_cd=312 .**

**LEFT SIDE APPLICATION**

It is used to change just a few characters of a variable.

data \_null\_ ;   
phone='(312) 555-1212' ;   
**substr(phone, 2, 3)='773' ;**   
put phone=;   
run ;

**Result :** The variable PHONE has been changed from (312) 555-1212 to **(773) 555-1212**.

**25. Difference between CEIL and FLOOR functions?**

The ceil function returns the smallest integer greater than/equal to the argument whereas the floor returns the greatest integer less than/equal to the argument.

**For example :** ceil(4.4) returns 5 whereas floor(4.4) returns 4.

**26. Difference between SET and MERGE?**

SET concatenates the data sets where as MERGE matches the observations of the data sets.  
  
**SET**

|  |
| --- |
| <https://1.bp.blogspot.com/-w0ZJIXNH7Ak/WJ7QmIoLLFI/AAAAAAAAF4A/Hz1u4dvcUR8F4d33G1ITzZ0WvQFwm795gCLcB/s1600/Set.png> |
| SAS SET Statement |

**MERGE**

|  |
| --- |
| [https://2.bp.blogspot.com/-Lo6FZgOiurQ/WJ7QS-r1pRI/AAAAAAAAF38/HPA_meDc_YUXgWaufwsqW2J6rDmCa80YwCLcB/s320/merge.png](https://2.bp.blogspot.com/-Lo6FZgOiurQ/WJ7QS-r1pRI/AAAAAAAAF38/HPA_meDc_YUXgWaufwsqW2J6rDmCa80YwCLcB/s1600/merge.png) |
| Join Horizontally |

**27. How to do Matched Merge and output only consisting of observations from both files?**  
  
Use **IN=variable** in MERGE statements. It is used for matched merge to track and select which observations in the data set from the merge statement will go to a new data set.  
  
data readin;  
merge file1**(in=infile1)** file2**(in=infile2)**;  
           by id;  
          **if infile1=infile2;**  
run;

**28. How to do Matched Merge and output consisting of observations in file1 but not in file2, or in file2 but not in file1?**  
  
data readin;  
merge file1**(in=infile1)** file2**(in=infile2)**;  
           by id;  
          **if infile1 ne infile2;**  
run;

|  |
| --- |
| [https://4.bp.blogspot.com/-dDAp7seLQHA/WJ7P-JvXW0I/AAAAAAAAF34/dol1x2u0aM0FJRtEFh4gvRIPg8bsn2xQACLcB/s1600/Merged%2BDataset.png](https://4.bp.blogspot.com/-dDAp7seLQHA/WJ7P-JvXW0I/AAAAAAAAF34/dol1x2u0aM0FJRtEFh4gvRIPg8bsn2xQACLcB/s1600/Merged+Dataset.png) |
| SAS MERGE |

**29. How to do Matched Merge and output consisting of observations from only file1?**  
  
data readin;  
merge file1**(in=infile1)** file2**(in=infile2)**;  
           by id;  
          **if infile1;**  
run;

**30. How do I create a data set with observations=100, mean 0 and standard deviation 1?**

data readin;

do i=1 to 100;

           temp=0 + rannor(1) \* 1;

           output;

end;

run;

proc means data=readin mean stddev;

var temp;

run;

**31. How to label values and use it in PROC FREQ?**

Use **PROC FORMAT** to set up a format.  
  
proc format;  
value score 0 - 100=‘100-‘  
101 - 200=‘101+’  
other=‘others’  
;  
  
proc freq data=readin;  
tables outdata;  
format outdata **score.** ;  
run;

**32. How to use arrays to recode set of variables?**

Recode the set of questions:  Q1,Q2,Q3...Q20 in the same way: if the variable has a value of 6 recode it to SAS missing.

data readin;

set outdata;

array Q(20) Q1-Q20;

do i=1 to 20;

 if Q(i)=6 then Q(i)=.;

end;

run;

**33. How to use arrays to recode all the numeric variables?**

Use \_numeric\_ and dim functions in array.

data readin;

set outdata;

array Q(\*) **\_numeric\_**;

do i=1 to **dim(Q)**;

 if Q(i)=6 then Q(i)=.;

end;

run;

**Note :** DIM returns a total count of the number of elements in array dimension **Q**.

**34. How to calculate mean for a variable by group?**

Suppose Q1 is a numeric variable and Age a grouping variable. You wish to compute mean for Q1 by Age.

PROC MEANS DATA=READIN;

VAR Q1;

**CLASS** AGE;  
RUN;

**35. How to generate cross tabulation?**

Use **PROC FREQ** code.

PROC FREQ DATA=auto;

**TABLES A\*B** ;

RUN;

SAS will produce table of  **A** by **B**.

**36. How to generate detailed summary statistics?**

Use **PROC UNIVARIATE** code.

PROC UNIVARIATE DATA=READIN;

  CLASS Age;

  VAR Q1;

RUN;

**Note :** Q1 is a numeric variable and Age a grouping variable.

**37. How to count missing values for numeric variables?**

Use **PROC MEANS with NMISS** option.

**38. How to count missing values for all variables?**  
  
proc format;  
value $missfmt ' '='Missing' other='Not Missing';  
value missfmt .='Missing' other='Not Missing';  
run;  
  
  
proc freq data=one;   
format \_CHAR\_ $missfmt.;  
tables \_CHAR\_ / missing missprint nocum nopercent;  
format \_NUMERIC\_ missfmt.;  
tables \_NUMERIC\_ / missing missprint nocum nopercent;  
run;

**39. Describe the ways in which you can create macro variables**

There are 5 ways to create macro variables:

1. %Let
2. Iterative %DO statement
3. Call Symput
4. Proc SQl into clause
5. Macro Parameters.

**40. Use of CALL SYMPUT**

CALL SYMPUT puts the value from a dataset into a macro variable.

proc means data=test;  
var x;  
output out=testmean mean=xbar;  
run;  
  
data \_null\_;  
set testmean;  
**call symput("xbarmac",xbar);**  
run;  
  
%put mean of x is &xbarmac;

**41. What are SYMGET and SYMPUT?**

**SYMPUT** puts the value from a dataset into a macro variable where as   
**SYMGET** gets the value from the macro variable to the dataset.

**42. Which date function advances a date, time or datetime value by a given interval?**

INTNX function advances a date, time, or datetime value by a given interval, and returns a date, time, or datetime value. Ex: INTNX(interval,start-from,number-of-increments,alignment).

**43. How to count the number of intervals between two given SAS dates?**

**INTCK**(interval,start-of-period,end-of-period) is an interval function that counts the number of intervals between two give SAS dates, Time and/or datetime.

**44. Difference between SCAN and SUBSTR?**

**SCAN** extracts words within a value that is marked by delimiters. **SUBSTR** extracts a portion of the value by stating the specific location. It is best used when we know the exact position of the sub string to extract from a character value.

**45.**The following data step executes:

Data strings;  
Text1=“MICKEY MOUSE & DONALD DUCK”;  
Text=scan(text1,2,’&’);  
Run;  
  
What will the value of the variable Text be?

**\* DONALD DUCK [(Leading blanks are displayed using an asterisk \*]**

**46. For what purpose would you use the RETAIN statement?**

A **RETAIN** statement tells SAS not to set variables to missing when going from the current iteration of the DATA step to the next. Instead, SAS retains the values.

**47. When grouping is in effect, can the WHERE clause be used in PROC SQL to subset data?**

**No.** In order to subset data when grouping is in effect, the HAVING clause must be used. The variable specified in having clause must contain summary statistics.

**48. How to use IF THEN ELSE in PROC SQL?**

PROC SQL;  
SELECT WEIGHT,  
**CASE  
WHEN WEIGHT BETWEEN 0 AND 50 THEN ’LOW’  
WHEN WEIGHT BETWEEN 51 AND 70 THEN ’MEDIUM’  
WHEN WEIGHT BETWEEN 71 AND 100 THEN ’HIGH’  
ELSE ’VERY HIGH’  
END AS NEWWEIGHT** FROM HEALTH;  
QUIT;

**49. How to remove duplicates using PROC SQL?**

Proc SQL noprint;  
Create Table inter.Merged1 as  
**Select distinct \* from inter.readin ;**  
Quit;

**50. How to count unique values by a grouping variable?**

You can use **PROC SQL** with **COUNT(DISTINCT variable\_name)** to determine the number of unique values for a column.

**51. How to merge two data sets using PROC SQL?**

**PROC SQL Merging**

**52. Difference between %EVAL and %SYSEVALF**

%EVAL cannot perform arithmetic calculations with operands that have the floating point values. It is when the %SYSEVALF function comes into picture.

*%let last= %eval (4.5+3.2);  
%let last2=%sysevalf(4.5+3.2);  
%put &last2;*

**53. How to debug SAS Macros**

There are some system options that can be used to debug SAS Macros:  
MPRINT, MLOGIC, SYMBOLGEN.

**54.**  
  
%let x=temp;  
%let n=3;  
%let x3=result;  
%let temp3=result2;  
Difference between &x&n , &&x&n ,  &&&x&n ?

**55. How to save log in an external file**

Use **PROC PRINTTO**

*proc printto log="C:\Users\Deepanshu\Downloads\LOG2.txt" new;  
run;*

**56. How Data Step Merge and PROC SQL handle many-to-many relationship?**

Data Step MERGE does not create a cartesian product incase of a many-to-many relationship. Whereas, Proc SQL produces a cartesian product.

**57. What is the use of 'BY statement' in Data Step Merge?**

Without 'BY' statement, Data Step Merge performs merging without **matching**. In other words, the records are combined based on their **relative position** in the data set. The second data set gets placed to the "right" of the first data set (no matching based on the unique identifier - if data is not sorted based on unique identifier, wrong records can be merged).  
  
When you use 'BY' statement, it matches observations according to the values of the BY variables that you specify.

**58. Use of Multiple SET Statments**

**59. How to combine tables vertically with PROC SQL**

**PROC SQL : Combine tables vertically**

**60. Two ways to reverse order of data**

**61. Which is more faster- Data Step / Proc SQL**

The SQL procedure performed better with the smaller datasets (less than approx. 100 MB) whereas the data step performed better with the larger ones (more than approx. 100 MB).

It is because the DATA step handles each record sequentially so it never uses a lot of memory, however, it takes time to process one at a time. So with a smaller dataset, the DATA step is going to take more time sending each record through.

With the SQL procedure, everything is loaded up into memory at once. By doing this, the SQL procedure can process small datasets rather quickly since everything is available in memory. Conversely, when you move to larger datasets, your memory can get bogged down which then leads to the SQL procedure being a little bit slower compared to the DATA step which will never take up too much memory space.

*If you need to connect directly to a database and pull tables from there, then use PROC SQL.*