**Important Notes (17 July 2017):**

1. When used ID variable, it will act as your first column in the output dataset. If ***ID + VAR*** had same variable defined in them, then it will print that variable twice. So use ID and VAR carefully.

2. At times column name can be changed using Label, so observe carefully the usage of ***Var and Label*** properly.

3. In ***PROC REPORT***, Define is like selecting a column in a excel sheet and parameters after ‘/’ are enhancement you make to that particular column. These affect the o/p window only & not HTML window. Enhancement includes:

* **Usage**: Across, Analysis, Computed, Display, Group and Order
* **Options**: Decending, Noprint, Nozero, PAGE
* **Attribute**: Format, Width, Spacing
* **Justification**: Left, Center, Right

Remember while defining attributes above ***(‘=’)*** must be used. E.g. Define Dob / format***=***DDMMYY10. Width=2;

Be careful while using ***format and width option***, length in both for a variable must be like display the data correctly.

4. ***END=*** is used to point the last record. It holds value 0 or 1. If end of file is reached it will be set to 1. Same can be used with IF condition for further operation.

5. Generally ***||*** is used for concatenation, but this will concatenate with a space in between. To join without space, characters must be trimmed using TRIM function and Numbers must be type casted to Numeric using PUT.

6. ***Variable name in SAS*** can start with \_ too. SAS variable name cannot start with numbers or any other special char.

7. While Merging datasets using ***MERGE command***, we can rename a column using RENAME=(Col1=Col\_1) and define IN parameter to check the presence of observation in all datasets involved in merging. Both these RENAME and IN should be placed in a same parenthesis like: Merge dataset1**(**Rename=(col1=col\_1) IN=in\_a)

8. While using ***DROP=*** separate each variable with a space and not comma. For e.g. DROP= Var1 Var2 Var3; When used in DATA line you can use only DROP= and not just DROP. DATA Sample1 (DROP=Var1 Var2 Var3);

9. Pointers created through ***Arrays*** are not available outside the DATA step. Can be declared using ARRAY statement inside the DATA step like ARRAY Ques{3,75} q1-q75; and same can be access using the pointer Ques only within the Data step like Ques{2,25}

10. ***OBS*** in Print statement is used to define the last number of observation until which a data needs to be read. So when we use firstobs along with OBS like PROC PRINT DATA=DATA1 (FIRSTOBS=5 OBS=15); it means start to read from 5th observation and read until observation 15 and get it printed. So, ***it would print 11 records***. Be careful, it is 11 record and not 10 because observation 5 is included. OBS can only be used on SAS Dataset as Raw file do not have observation numbers. So OBS= can be used only SET or MERGE or PROC statements.

11. While using ***Format*** command, can include multiple variables in them in a single format command.

12. In ***input*** @1 height 2. @4 weight 2; when we print this weight will be printed with value 2.

13. ***input*** weight1-weight3 / height1-height3; used to read data which is spread sequentially in two rows.

14. Values of ***Retain*** variables will be assigned one time. However, value for simple assignment statement will be assigned for each observation in the dataset being read or set.

15. values returned by ***day***(x), ***month***(x) and ***year***(x) are of type numeric.

16. ***\*\**** is used for exponentiation. 3\*\*2 is 9

17. In ***Proc Print***, which using the where condition, strings are enclosed inside “ ” while numeric are not.

18. Standard numeric values include -3.7, ***+3.7***, 3.7 and 37

19. ***Date*** is a numeric value to SAS when read. *X="01Jan1960" D;*

*20.* ***@@*** *is used to hold a row for one DATA Step iteration,* ***@*** *hold a row until next input, used with do loop mostly.*

**Important Notes (18 July 2017):**

21. ***PROC SORT*** must always be sorted using with at least one ‘BY’ column. Without out it will affect original dataset.

22. ***Length*** statement must always be declared before any SET or assignment statement. Else it has no effect on var.

23. If ***DO-Until*** condition has > then same in ***DO-While*** will be <=; similarly, if DO-Until has < DO-while will have >= that is do while (count**<10**); is same as do until (count**>=10**); and do while (count**<=10**); is same as do until (count**>=11**); Similarly do while (count>**10**); is equal to do until (count<=**10**); and do while (count>=**10**); is equal to do until (count<=**9**);

24. Other than ***strings***, we cannot enclose any parameters or style inside a ***‘ ‘ quotation*** mark.

25. ***WHERE*** can be used on an existing variable in dataset, while IF can be used for both existing and assigned var.

26. When you use a ***parameter*** in a DATA or Proc step, if it must be defined you use ***= sign***, e.g PROC Print Data=<DS> obs=15 NOOBS; Here you see when we define OBS which need to be defined with a variable we use = and assign value 15 to it, this is same for OUT=, KEEP= and DROP=. However, when we define a standalone parameter it is just NOOBS.

27. ***Label*** command can be issues anywhere in the program where the dataset have more than 1 row. This is because this label command will be picked by SAS when it try to run through the Data step for the second record and accordingly change the PDV accordingly.

28. ***Input*** statement in DATA step is used to read data record by record. After the end of each input stamen, record pointer moves to the next line of the dataset/raw data. If you need to hold the record in same line use ***@***

29. There is great difference between ***if level = 2 or 3*** command and else if ***level = 2 or level = 3.*** Level = 2 or 3 means if level is equal to 2 or ***any non-missing values*** (3 denotes this) then if condition is true, while the later means if level is equal to 2 or level = 3 then if condition is true.

30. ***CAT*** command is used to concatenate data, while ***CATX*** is used to concatenate a data with a delimiter. CAT(var1,var2) and CATX(‘-‘,Var1,Var2)

31. ***DROP =*** Var1 is used mainly in DATA step while ***DROP*** Var1 Var2 is used inside the data step. You cannot use ‘,’ to separate the Variables, just the space would do. Do not declare it inside () because it becomes array to SAS.

**Important Notes (18 July 2017):**

32. When it comes to ***Arrays***, you define it using ARRAY keyword followed by Array Pointer Name and its element inside {}. E.g. array prady{**3**}; this will by default create 3 variable Prady1-prady3 if the element names are not explicitly mentioned. Array should be either a Numeric or character array, array elements cannot have data of mixed data types. If it’s a character array after the array element $ should be present, else not needed. We can even explicitly mention the variable name and also assign the element values for array like array prady{**3**} $ Name1-Name3 ('Prady','Sruthi','Sathya'); Multi-dimentional arrays are created by defining the array elements correctly like {2,3} implies array with 2 rows and 3 columns totally 2\*3=6 elements.

33. You can even define the ***length of an array*** before assigning the values like array prady{**3**} $ **8** ('Prady','123456789','Sathya');

34. Any ***Syntax mistakes*** will lead to an error, still SAS compiles remaining codes and stops without executing them. However, when an issue is with data, SAS compiles and even execute the code but create mismatching values to the data being stored to SAS Dataset.

35. ***POINT*** is used to access any specific observation directly from a SAS Dataset only and not raw data. Generally, we create a variable with numeric constant and then assign that to POINT like VAR1=5; POINT=VAR1; and POINT will be always be in SET command only. set SASHELP.CARS POINT=PT\_VAR; Remember you cannot use END; when using POINT. You need to use only the ***STOP*** keyword, else it will be crate an infinite loop. Explicit ***OUTPUT***; is needed to write the data from PDV to SAS Dataset. Else, we dataset will be empty.

36. ***RETAIN*** will initially hold no values unless initialized. It is ***generally used to initialize a sum variable*** to consider some defined value, else sum variable will start with default value ‘0’. Thus, when RETAIN is used for a Sum variable its default value is 0, for Char variable it needs to be defined properly like RETAIN CITY=’CALI’; and value for CITY will be retained as Cali for all the dataset. ***Retain is just one time assignment,*** while any actual assignment statement is a multiple time assignment based on the number of observations.

37.

***Read through these on 18 July 2017:***

1. *Revise Proc Report - Done*
2. *Major problem in Data reading (that uses & and :) - Done*
3. *See how to access the end of the dataset and usage of END*
4. *Read Chapter-14 - Done*
5. *Use of @ and @@ - Done*
6. *When to use DSD and when to use DLM, See how MISSOVER, TRUNCOVER is used - Done*
7. *See how computed is used in PROC REPORT - Done*
8. *Difference between Keep and Keep= and its usage - Done*
9. *How & and : is used in a unstructure list input - Done*
10. *See how sum variable is used internally by SAS*

***Read through these on 18 July 2017:***

1. *See the DO-WHile and DO-Untill equalent - Done*
2. *Difference between CAT and CATX - Done*
3. *How Sum variable works, do they get automatically initialized when used inside data step and how it gets initialized when used inside a Do loop?*
4. *Revise chapter-14, you surely get atleast 2 straight question from this*
5. *See how Keep= and Keep work inside a data step, we know in data step*
6. *Be clear in all default values hold by retain Var; and Sum Variables in a Datastep.*
7. *See how to use Data step Debugger? DATA Test / DEBUG; Key word is 'DEBUG' and not 'Debugger'*
8. *See how DO i=1 to 10 by 1; works in background in PDV, what would be the final stored value for i in PDV? Actual stored value would be 10. However, value of i would stop with 11. Be clear on what is stored value and what is actual value*
9. *expense values within each ascending IDNUMBER value implies IDUMBER will be sorted first over which Expenses will be sorted*