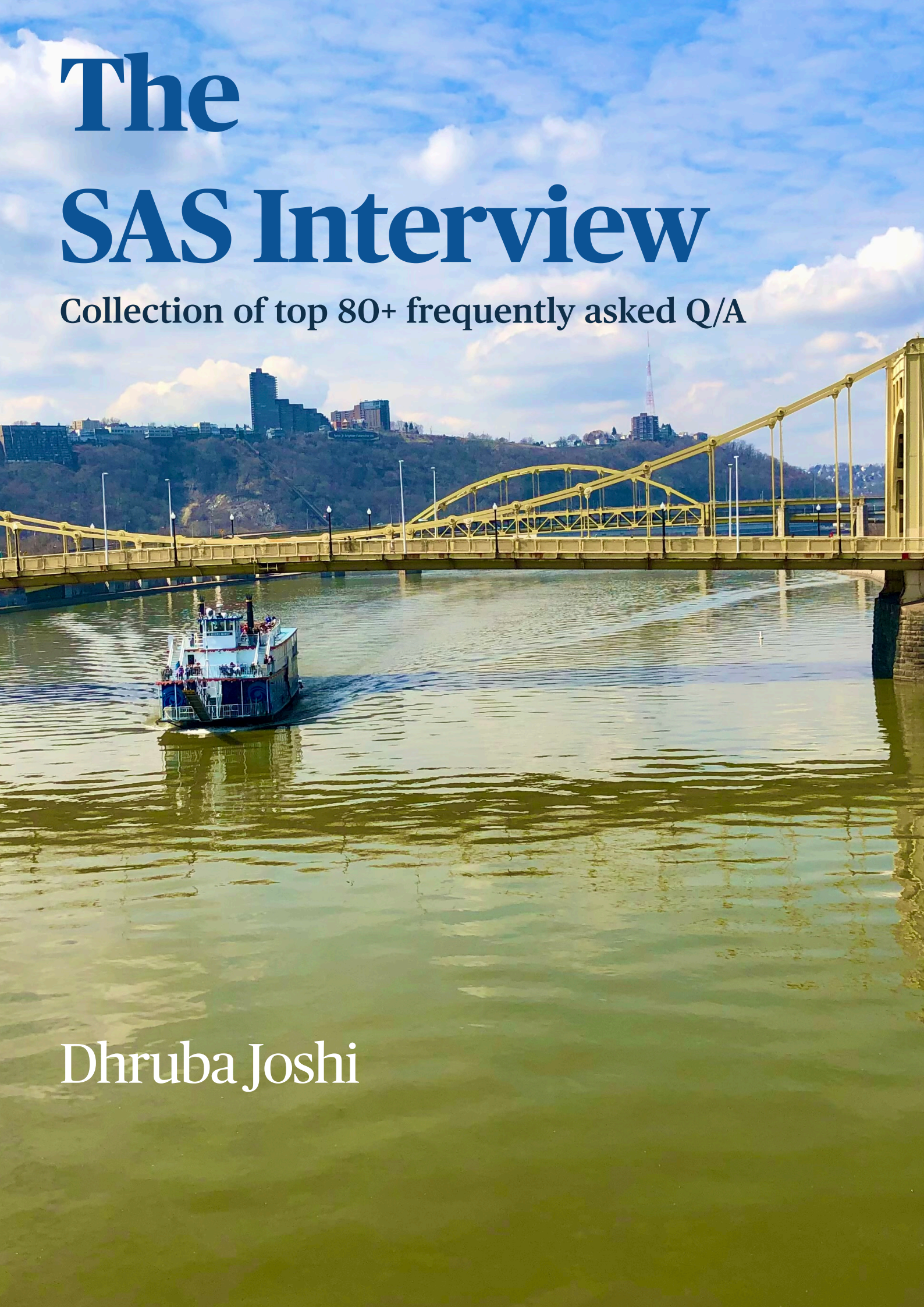
# The SAS Interview

## Collection of top 80+ frequently asked Q/A



Dhruba Joshi

# About the Author

Dhruba Joshi – I have been working as a Senior SAS developer for the past decade in world's top financial institutions including Wells Fargo, PNC and US Bank. Originally, my background is in physics. While I pursued my PhD, I grew fond of computational physics using mathematics and statistics to draw insights on real world problems. I decided to go into industry and use SAS/SQL primarily in my work.

I wrote this book to provide a resource for interview candidates learning SAS. These are questions I have personally encountered in my interviews from base SAS developer roles to advanced senior roles. As you go through this book, I encourage you to follow along with the YouTube video links provided for each problem.

## Q01: What are the data types in SAS?

⇨ **SAS has two data types (Numeric & Character).**

**1. Numeric:**
Integer, Decimal values, Date

**2. Character:**
Alphanumeric, Special character

***Let's check data type for a Dataset: CARS.***

```
PROC CONTENTS DATA =SASHELP.CARS;
RUN;
```

**Alphabetic List of Variables and Attributes**

| # | Variable | Type | Len | Format | Label |
|---|---|---|---|---|---|
| 9 | Cylinders | Num | 8 | | |
| 5 | DriveTrain | Char | 5 | | |
| 8 | EngineSize | Num | 8 | | Engine Size (L) |
| 10 | Horsepower | Num | 8 | | |
| 7 | Invoice | Num | 8 | DOLLAR8. | |
| 15 | Length | Num | 8 | | Length (IN) |
| 11 | MPG_City | Num | 8 | | MPG (City) |
| 12 | MPG_Highway | Num | 8 | | MPG (Highway) |
| 6 | MSRP | Num | 8 | DOLLAR8. | |
| 1 | Make | Char | 13 | | |
| 2 | Model | Char | 40 | | |
| 4 | Origin | Char | 6 | | |
| 3 | Type | Char | 8 | | |
| 13 | Weight | Num | 8 | | Weight (LBS) |
| 14 | Wheelbase | Num | 8 | | Wheelbase (IN) |

**Video link: Youtube Video**

## Q02: How do you copy SAS dataset to a new SAS Dataset?

⇨ **SAS data set can be copied to another SAS dataset name using set statement.**

**Sample Code:**

```
DATA NEWDATA;
SET SASHELP.CLASS;
RUN;
```

**Output data in Newdata:**

| | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69 | 112.5 |
| 2 | Alice | F | 13 | 56.5 | 84 |
| 3 | Barbara | F | 13 | 65.3 | 98 |
| 4 | Carol | F | 14 | 62.8 | 102.5 |
| 5 | Henry | M | 14 | 63.5 | 102.5 |
| 6 | James | M | 12 | 57.3 | 83 |
| 7 | Jane | F | 12 | 59.8 | 84.5 |
| 8 | Janet | F | 15 | 62.5 | 112.5 |
| 9 | Jeffrey | M | 13 | 62.5 | 84 |
| 10 | John | M | 12 | 59 | 99.5 |
| 11 | Joyce | F | 11 | 51.3 | 50.5 |
| 12 | Judy | F | 14 | 64.3 | 90 |
| 13 | Louise | F | 12 | 56.3 | 77 |
| 14 | Mary | F | 15 | 66.5 | 112 |
| 15 | Philip | M | 16 | 72 | 150 |
| 16 | Robert | M | 12 | 64.8 | 128 |
| 17 | Ronald | M | 15 | 67 | 133 |
| 18 | Thomas | M | 11 | 57.5 | 85 |
| 19 | William | M | 15 | 66.5 | 112 |

**Video link: Youtube Video**

## Q03: What is the difference between single and multiple set statement?

⇨ **Single and multiple set statement in SAS data step result differently. We have following three datasets TEST1, TEST2 TEST3.**

```
DATA TEST1;
INPUT NAME $ SEX $ AGE HEIGHT WEIGHT;
DATALINES;
Jess F 30 65 122
Bob M 33 67 157
Philip M 55 70 170
Krish M 32 62 166
;
RUN;
DATA TEST2;
INPUT NAME $ SEX $ AGE HEIGHT WEIGHT;
DATALINES;
GAYLE F 38 60 125
MIKE M 48 67 153
;
RUN;
DATA TEST3;
INPUT NAME $ SEX $ AGE HEIGHT WEIGHT;
DATALINES;
Jaferry M 14 67 153
;
RUN;
```

**Output: TEST1**

Total rows: 4 Total columns: 5

| | NAME | SEX | AGE | HEIGHT | WEIGHT |
|---|---|---|---|---|---|
| 1 | Jess | F | 30 | 65 | 122 |
| 2 | Bob | M | 33 | 67 | 157 |
| 3 | Philip | M | 55 | 70 | 170 |
| 4 | Krish | M | 32 | 62 | 166 |

**Output: TEST2**

Total rows: 2 Total columns: 5

| | NAME | SEX | AGE | HEIGHT | WEIGHT |
|---|---|---|---|---|---|
| 1 | GAYLE | F | 38 | 60 | 125 |
| 2 | MIKE | M | 48 | 67 | 153 |

**Output: TEST3**

Total rows: 1 Total columns: 5

| | NAME | SEX | AGE | HEIGHT | WEIGHT |
|---|---|---|---|---|---|
| 1 | Jaferry | M | 14 | 67 | 153 |

***Generating following two data sets COMBINE1 COMBINE2 with single set statement.***

**Combining data without sorting:**

```
DATA COMBINE1;
SET TEST1 TEST2 TEST3;
RUN;
```

***Result: Data will append TEST1 TEST2 TEST3 accordingly (Without sorting).***

**Combining data with sorting by sex variable.**

```
DATA COMBINE2;
SET TEST1 TEST2 TEST3;
BY Sex;
RUN;
```

***Result: Data will append TEST1 TEST2 TEST3 accordingly with sorting by sex variable.***

***Output: COMBINE1***

Total rows: 7 Total columns: 5

| | NAME | SEX | AGE | HEIGHT | WEIGHT |
|---|---|---|---|---|---|
| 1 | Jess | F | 30 | 65 | 122 |
| 2 | Bob | M | 33 | 67 | 157 |
| 3 | Philip | M | 55 | 70 | 170 |
| 4 | Krish | M | 32 | 62 | 166 |
| 5 | GAYLE | F | 38 | 60 | 125 |
| 6 | MIKE | M | 48 | 67 | 153 |
| 7 | Jaferry | M | 14 | 67 | 153 |

***Output: COMBINE2***

Total rows: 7 Total columns: 5

| | NAME | SEX | AGE | HEIGHT | WEIGHT |
|---|---|---|---|---|---|
| 1 | Jess | F | 30 | 65 | 122 |
| 2 | GAYLE | F | 38 | 60 | 125 |
| 3 | Bob | M | 33 | 67 | 157 |
| 4 | Philip | M | 55 | 70 | 170 |
| 5 | Krish | M | 32 | 62 | 166 |
| 6 | MIKE | M | 48 | 67 | 153 |
| 7 | Jaferry | M | 14 | 67 | 153 |

***Generating following data set COMBINE3 with multiple set statement.***

**Using multiple set statement.**

```
DATA COMBINE3;
SET TEST3;
SET TEST2;
SET TEST1;
RUN;
```

***Result: It displays records from the last set statement data where the number of the record is controlled by the minimum records of the provided data sets.***

**[Displays only first one record from Last set statement data (Here from TEST1)]**

***Output: COMBINE3***

Total rows: 1 Total columns: 5

| | NAME | SEX | AGE | HEIGHT | WEIGHT |
|---|---|---|---|---|---|
| 1 | Jess | F | 30 | 65 | 122 |

**Video link: Youtube Video**

## Q04. What are the differences between data step and proc step in SAS?

⇨ **1. Data step starts with DATA statement and procedure starts with PROC statement.**

**2. Data step reads input data into a SAS dataset, edit, merge, update, rearrange, and create data set. Proc step is used to perform specific analysis on data and produce results or report.**

**Data Step:**

```
DATA TEST;
SET SASHELP.CLASS;
BMI=703*WEIGHT/HEIGHT**2;
WHERE AGE =14;
RUN;
```

***Output: TEST***

| | Name | Sex | Age | Height | Weight | BMI |
|---|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69 | 112.5 | 16.611531191 |
| 2 | Carol | F | 14 | 62.8 | 102.5 | 18.270898414 |
| 3 | Henry | M | 14 | 63.5 | 102.5 | 17.870295741 |
| 4 | Judy | F | 14 | 64.3 | 90 | 15.3029757 |

**Proc Step:**

```
TITLE "PRINT REPORT";
PROC PRINT DATA = SASHELP.CLASS;
VAR NAME SEX AGE;
WHERE AGE IN (15,16);
RUN;
```

***Result:***

**Print Report**

| Obs | Name | Sex | Age |
|---|---|---|---|
| 8 | Janet | F | 15 |
| 14 | Mary | F | 15 |
| 15 | Philip | M | 16 |
| 17 | Ronald | M | 15 |
| 19 | William | M | 15 |

**Video link: Youtube Video**

## Q05: What do you mean by informat and format statement in SAS?

⇨ **1. Format can be used in both data step and proc step but informat is used only in data step.**

**2. Format is a layout specification for how a variable be printed or displayed where informat is a specification for how the raw data should read.**

***Sample code:***

```
DATA TEST;
INPUT ID $ DOB;
INFORMAT DOB MMDDYY10.;
FORMAT DOB DATE9.;
DATALINES;
001 03/12/1990
002 09/22/1994
003 10/19/1996
005 23/12/2002
;
RUN;
```

***Output: TEST***

Total rows: 4 Total columns: 2

| | ID | DOB |
|---|---|---|
| 1 | 001 | 12MAR1990 |
| 2 | 002 | 22SEP1994 |
| 3 | 003 | 19OCT1996 |
| 4 | 005 | . |

**Video link: Youtube Video**

## Q06: How to print specific number of records from a SAS dataset?

⇨ **Following code can be used to print specific number of records:**

**Print given number of records from the starting.**

```
Title 'Printing first 6 records from Class';
PROC PRINT DATA=SASHELP.CLASS(OBS=6);
RUN;
```

**Print given number of records from specific records (Any starting and ending record number):**

```
Title 'Printing 6 records starting row 5 without obs column';
PROC PRINT DATA=SASHELP.CLASS(FIRSTOBS=5 OBS=10) noobs;
RUN;
```

***Result:***

**Printing first 6 records from Class**

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69.0 | 112.5 |
| 2 | Alice | F | 13 | 56.5 | 84.0 |
| 3 | Barbara | F | 13 | 65.3 | 98.0 |
| 4 | Carol | F | 14 | 62.8 | 102.5 |
| 5 | Henry | M | 14 | 63.5 | 102.5 |
| 6 | James | M | 12 | 57.3 | 83.0 |

**Printing 6 records starting row 5 without obs column**

| Name | Sex | Age | Height | Weight |
|---|---|---|---|---|
| Henry | M | 14 | 63.5 | 102.5 |
| James | M | 12 | 57.3 | 83.0 |
| Jane | F | 12 | 59.8 | 84.5 |
| Janet | F | 15 | 62.5 | 112.5 |
| Jeffrey | M | 13 | 62.5 | 84.0 |
| John | M | 12 | 59.0 | 99.5 |

## Video link: Youtube Video

## Q07: How do you rename existing SAS dataset name?

⇨ ***Using change statement with datasets procedure we can rename one or more than one dataset name.***

```
Libname ABCD 'SAS-library path';
proc datasets library=ABCD;
 change Existing_Name1=New_Name1 Existing_Name2=New_Name2;
Run;
```

***Changing country table from DRJ library to New name Country_Data.***

```
Libname DRJ 'SAS-library path';
proc datasets library=DRJ;
 change Country=Country_Data;
Run;
```

## Video link: Youtube Video

## Q08: If no variable lengths are specified in a SAS program, what will be the default length?

⇨ ***For each undefined variable length, each variable is given a default length 8 bytes.***

***For Character Variable:***

***In Character variable 8 bytes means we can store values with up to eight characters.***

```
Data Character;
Input Name $ Address $;
Datalines;
Kingjuni 679WesternIllinis
Stefenkingjuni Virginia
;
Run;
```

***For Numeric Variable:***

***In Numeric variable 8 bytes means we can store numbers with approximately 14 or 15 significant figures.***

***Important point is- 8 Bytes in numeric variable does not mean to limit numbers with eight digits.***

```
Data Numeric;
Input ACNum1 ACNum2 ;
Datalines;
12675344 8987765443
567876543456 78765678765666
;
Run;
```

***Output: Character***

| | Name | Address |
|---|---|---|
| 1 | Kingjuni | 679Weste |
| 2 | Stefenki | Virginia |

***Output: Numeric***

| | ACNum1 | ACNum2 |
|---|---|---|
| 1 | 12675344 | 8987765443 |
| 2 | 567876543456 | 7.8765679E13 |

***Video link: Youtube_Video***

## Q09: What do you mean by PDV (Program Data Vector)?

⇨ ***PDV (Program Data Vector) object is created at the end of the compilation phase of SAS data step which can be explained as follow.***

1. ***During the data step processing, logical area of the memory is created which is called PDV(Program Data Vector).***
2. ***SAS builds a SAS data set by reading one observation at a time into PDV.***
3. ***PDV contains two types of variables***
    1. ***Permanent variables (Variables from data set and Computed variable)***
    2. ***Temporary variable (Automatic variable and Option defined variable)***

***i. Automatic Variables:-***

***_N_ : It is incremented by 1***

***_ERROR_: It's value either 0 or 1***

***ii. Option Defined Variables:-***

***First.Variable Last.Variable***

***IN=Variable***

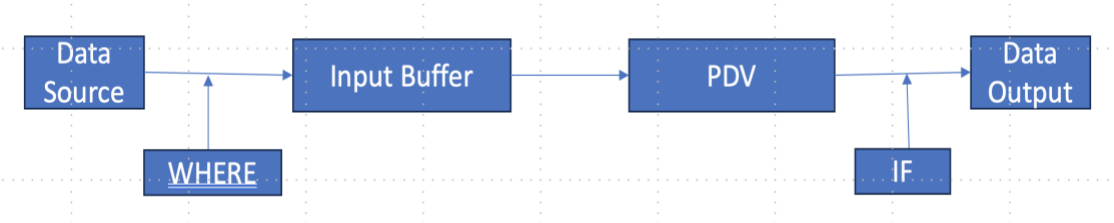
***End=Variable***

***Automatic Variables _N_ and _ERROR_ drop before writing output dataset.***

*Video link: Youtube_Video*

**Q10: While sub-setting data using WHERE and IF statement, which one is more efficient and why?**

***Use of where statement is more efficient than if statement, which can be explain as follow: While processing SAS data step, It follow steps as given below figure.***




***Here WHERE condition is applied before the data enters the input buffer while IF condition is applied after the data enters the program data vector.***
***When we apply WHERE condition, sub-setting applies before entering PDV that means SAS doesn't need to read all records but when we apply IF condition, SAS need to read all records first then sub-setting apply.***
***This is the reason why the WHERE condition more efficient than IF condition.***

## Use of Where and If statement [We have used HEART data set (Total records=5209) from SASHELP library]

### Using Where:

```
DATA USE_WHERE;
SET SASHELP.HEART;
WHERE DEATHCAUSE='Cancer'; RUN;
```

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         DATA USE_WHERE;
70         SET SASHELP.HEART;
71         WHERE DEATHCAUSE='Cancer';
72         RUN;

NOTE: There were 539 observations read from the data set SASHELP.HEART.
      WHERE DEATHCAUSE='Cancer';
NOTE: The data set WORK.USE_WHERE has 539 observations and 17 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      user cpu time       0.00 seconds
      system cpu time     0.00 seconds
```

### Using If:

```
DATA USE_IF;
SET SASHELP.HEART;
IF DEATHCAUSE='Cancer'; RUN;
```

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         DATA USE_IF;
70         SET SASHELP.HEART;
71         IF DEATHCAUSE='Cancer';
72         RUN;

NOTE: There were 5209 observations read from the data set SASHELP.HEART.
NOTE: The data set WORK.USE_IF has 539 observations and 17 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      user cpu time       0.00 seconds
      system cpu time     0.00 seconds
```

***Conclusion: From above example, while using where statement only 539 records read store those records to use_where table. But while using if statement it reads 5209 records and after sub-setting only 539 records stored to use_if table.***

***Video link: Youtube_Video***

## Q11: Which statement (where or if) can be applied on temporary variables and why?

⇨ ***If statement is used with temporary variables [automatic variables such as _N_, _ERROR_ and option variables such as first.variable, last.variable].***
***Automatic variables and option variables are generated at the PDV. As we know, where statement is applied before PDV and if statement is applied after PDV. From here it is clear that only if statement can be used with automatic and option defined variables.***

***Let's check with where statement:***

```
DATA TEST_WHERE;
SET SASHELP.CLASS;
WHERE _N_=5;
RUN;
```

***Log:***

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         DATA TEST_WHERE;
70         SET SASHELP.CLASS;
71         WHERE  _N_=5;
ERROR: Variable _N_ is not on file SASHELP.CLASS.
72         RUN;

NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.TEST_WHERE may be incomplete.  When this step was stopped there were 0 observations and
WARNING: Data set WORK.TEST_WHERE was not replaced because this step was stopped.
```

***Let's check with if statement:***

```
DATA TEST_IF;
SET SASHELP.CLASS;
IF _N_=5;
RUN;
```

***Log:***

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         DATA TEST_IF;
70         SET SASHELP.CLASS;
71         IF  _N_=5;
72         RUN;

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.TEST_IF has 1 observations and 5 variables.
```

Total rows: 1 Total columns: 5 Rows 1-1

| | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Henry | M | 14 | 63.5 | 102.5 |

***Similar explanation is applied with option variables first.variable, last.variable. But in this situation source data should be properly sorted with the corresponding variable.***

Total rows: 4 Total columns: 3

| | ID | NAME | SALARY |
|---|---|---|---|
| 1 | 101 | KRI%%%SH | 66669 |
| 2 | 102 | $$$MA@GAN# | 89000 |
| 3 | 103 | JAFFERY&! | 90000 |
| 4 | 105 | A#NNAYA*** | 75900 |

**Video link: Youtube Video**

## Q12: How do you check if there are duplicate records present in a SAS dataset?

⇨ ***First count each record using group by all field and print those records whose count is greater than 1. If it gives output result, then it confirms that dataset contains duplicate records.***

**Sample data set.**

```
DATA EMPLOYEE;
INPUT E_ID NAME $ SALARY;
DATALINES;
1 ROBERT 8750
2 SAMI 8000
2 SAMI 8000
3 JESS 8700
1 MARK 8000
3 JESS 8700
5 JENNY 7000
5 KEN 8700
;
RUN;
```

**Output:**

| | E_ID | NAME | SALARY |
|---|---|---|---|
| 1 | 1 | ROBERT | 8750 |
| 2 | 1 | MARK | 8000 |
| 3 | 2 | SAMI | 8000 |
| 4 | 2 | SAMI | 8000 |
| 5 | 3 | JESS | 8700 |
| 6 | 3 | JESS | 8700 |
| 7 | 5 | JENNY | 7000 |
| 8 | 5 | KEN | 8700 |

**Proc SQL used to identify duplicate records in a SAS dataset.**

```
PROC SQL;
SELECT *, COUNT(*) AS REC_COUNT
FROM EMPLOYEE
GROUP BY E_ID, NAME, SALARY
HAVING REC_COUNT >1;
QUIT;
```

**Result:**

| E_ID | NAME | SALARY | REC_COUNT |
|---|---|---|---|
| 2 | SAMI | 8000 | 2 |
| 3 | JESS | 8700 | 2 |

**Video link: Youtube_Video**

**Q13: How to include and exclude specific variables in a SAS dataset?**

⇨ **To include and exclude specific variables, we can use either keep/drop option or keep/drop statement.**

**Using Keep/Drop Option:**

1. Keep/Drop option can be used in both input and output data.
2. Keep/Drop option is used with both data step and proc step.

**Important Note:**

Using keep/drop option with input data source is more efficient than use of keep/drop option with output data. Because when we use keep/drop option with set statement (input data set), it reads only required field and results to output dataset.

**Using Keep/Drop Statement:**

1. Keep/Drop statement can be used in output data.
2. Keep/Drop statement is used with data step only.

**Video link: Youtube_Video**

**Q40: How can we save SAS log into txt format and reprint into SAS log window?**

⇨ **One of the important use of put statement is to print information into SAS log window. Put statement is used to redirect txt log format to SAS log window. We follow following steps:**

**Step1: Defining fileref where we store log.**

```
filename fileref "/home/joshid700/Interview/Log_printing.txt";
```

**Step2: Sending log to fileref:**

```
proc printto log=fileref ;
run;
```

**Step3: SAS Program/s:**

```
Title 'Printing  first 5 records from class dataset';
Proc print data =sashelp.class(obs=5);
run;
```

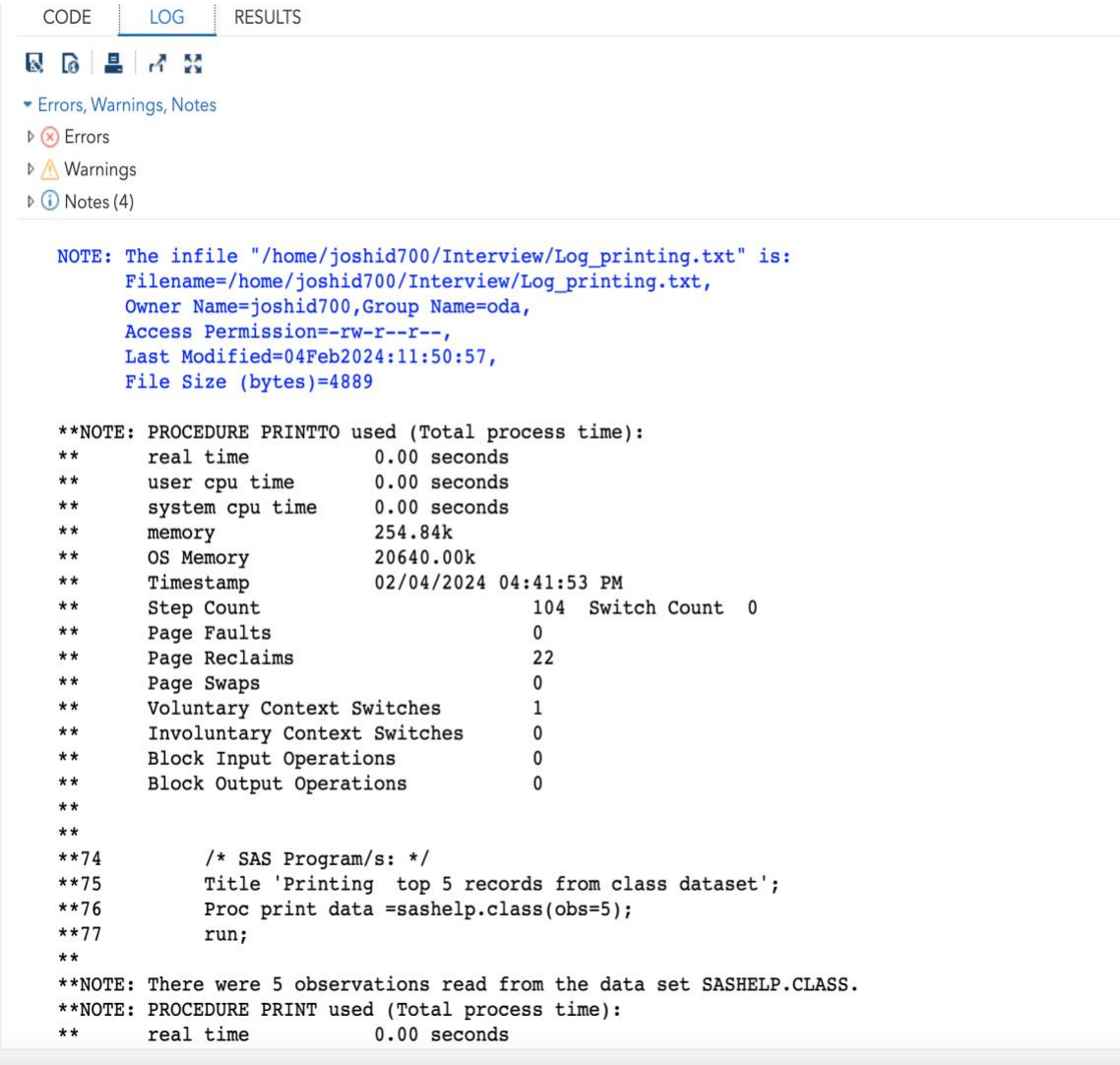
**Step4: Redirecting txt log from fileref to log window:**

```
proc printto; run;
data _null_;
infile "/home/joshid700/Interview/Log_printing.txt";
 input;
 putlog "**" _infile_;
 run;
```

**Result:**

**Printing first 5 records from class dataset**

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69.0 | 112.5 |
| 2 | Alice | F | 13 | 56.5 | 84.0 |
| 3 | Barbara | F | 13 | 65.3 | 98.0 |
| 4 | Carol | F | 14 | 62.8 | 102.5 |
| 5 | Henry | M | 14 | 63.5 | 102.5 |




**Video link: Youtube_Video**

## Q14: What are the commonly used character function in SAS?

⇨ ***Commonly used character function in SAS are as follow:***

1. Length of Character Value
   Length, Lengthn, Lengthc
2. Changing case:
   Upcase , Lowcase, Propcase,
3. Removing blank and Characters:
   Compbl, Compress
4. Searching Character and word
   Find, Findw
5. Extracting part of string and word:
   Scan, Substr(left of= ), Substr(right of =)
6. Removing leading and trealing blanks:
   Trim, Left, Strip
7. Changing variable type:
   Input, Put
8. Counting letter and word:
   Count, Countw

9. Joining two or more strings:
   Cat, Catx, Cats
10. Searching for Character Classes:
   Anyalpha, Anydigit, Anypunct, Anyspace

**Video link: Youtube Video**

**Q15.Accidently unwanted special characters were attached to customer name. How do you remove those unwanted special characters?**

⇨ ***Unwanted special characters can be removed using SAS character function compress with modifier.:***
***Var_name = COMPRESS( Var_name, 'modifier');***
***Sample dataset:***

```
DATA JUNK_DATA;
INPUT ID NAME $16. SALARY;
DATALINES;
101 KRI%%%SH        66669
102 $$$MA@GAN#      89000
103 JAFFERY&!       90000
105 A#NNAYA***      75900
;
RUN;
```

Total rows: 4 Total columns: 3

| | ID | NAME | SALARY |
|---|---|---|---|
| 1 | 101 | KRI%%%SH | 66669 |
| 2 | 102 | $$$MA@GAN# | 89000 |
| 3 | 103 | JAFFERY&! | 90000 |
| 4 | 105 | A#NNAYA*** | 75900 |

***Clean data after using SAS character function:***

```
DATA CLEAN_DATA;
SET JUNK_DATA;
NAME=COMPRESS(NAME,'$,@,%,#,!,*,&');
RUN;
```

Total rows: 4 Total columns: 3

| | ID | NAME | SALARY |
|---|---|---|---|
| 1 | 101 | KRISH | 66669 |
| 2 | 102 | MAGAN | 89000 |
| 3 | 103 | JAFFERY | 90000 |
| 4 | 105 | ANNAYA | 75900 |

**Video link: Youtube Video**

**Q16. How to convert character variable to numeric variable in SAS?**

**=> In data step INPUT function is used to change character to numeric variable with different name.Expression used to convert character to numeric variable is as below:**

**Numvar=input(Charvar, Informat.);**
Sample dataset:

```
DATA CHAR;
INPUT X $10.;
DATALINES;
2323456789
5433333
66
;
RUN;
```

**Output:**

Total rows: 3 Total columns: 1

| | X |
|---|---|
| 1 | 2323456789 |
| 2 | 5433333 |
| 3 | 66 |

**New Dataset and new var with numeric data type.**

```
DATA NUM
SET CHAR;
NEW_VAR=INPUT(X, 10.);
RUN;
```

**Output:**

Total rows: 3 Total columns: 2

| | X | NEW_VAR |
|---|---|---|
| 1 | 2323456789 | 2323456789 |
| 2 | 5433333 | 5433333 |
| 3 | 66 | 66 |

**Validating New_Var data type:**

```
PROC CONTENTS DATA = NUM;
RUN;
```

**Result:**

Alphabetic List of Variables and Attributes

| # | Variable | Type | Len |
|---|---|---|---|
| 2 | NEW_VAR | Num | 8 |
| 1 | X | Char | 10 |

## Video link: Youtube_Video

### Q17. How to convert numeric variable to character variable in SAS?

*=> In data step PUT function is used to change numeric to character variable with different name. Expression used to convert numeric to character variable is as below:*

**Charvar=put(Numvar, Format.);**

**Dataset creation:**

```
DATA NUM;
INPUT Y;
DATALINES;
2323456789
5433333
66
;
RUN;
```

**Output:**

Total rows: 3 Total columns: 1

| | Y |
|---|---|
| 1 | 2323456789 |
| 2 | 5433333 |
| 3 | 66 |

**New Dataset and new_var with character data type.**

```
DATA CHAR;
SET NUM;
NEW_VAR=PUT(Y,10.);
RUN;
```

**Output:**

Total rows: 3 Total columns: 2

| | Y | NEW_VAR |
|---|---|---|
| 1 | 2323456789 | 2323456789 |
| 2 | 5433333 | 5433333 |
| 3 | 66 | 66 |

**Validating New_Var data type:**

```
PROC CONTENTS DATA = CHAR;
RUN;
```

**Result:**

Alphabetic List of Variables and Attributes

| # | Variable | Type | Len |
|---|---|---|---|
| 2 | NEW_VAR | Char | 10 |
| 1 | Y | Num | 8 |

**Video link: Youtube_Video**

## Q18: What are the commonly used numeric function in SAS?

⇨ **Commonly used numeric function in SAS are as follow:**

**1. Round and Truncate:**

Round, Int

**2. Descriptive Statistics:**

Min, Max, Sum, Mean, Median

**3. Mathematical Function:**
ABS, SQRT, Exp, log

**4. Function to get previous observation:**
Lag, lag2, dif

**5. Missing and Non missing value:**
N, Miss

**6. Getting first non-missing value:**
Coalesce

**7. Getting smallest integer which is greater than or equal to argument.**
CILL

**8. Getting largest integer which is smaller than or equal to argument.**
FLOOR

**Video link: Youtube_Video**

**Q19: What are the commonly used Date functions in SAS?**

⇨ **Following are Date functions and their use.**

**1. DATE():**
converts today's date into days from JAN01 1960.

**2. DATETIME ():**
Converts today's instant into second calculating from JAN01 1960.

**3. TODAY ():**
Converts today's date into days from JAN01 1960.

**4. DAY (Date):**
Converts day of the month.

**5. WEEK(Date):**
Converts week of year, 1 to 52

**6. WEEKDAY (Date):**
Converts day of the week, SUN-1, Mon-2,…

**7. QTR (Date):**
Converts into quarter of the year. 1 to 4.

**8. MONTH (Date):**
Convert into nth month of the year. 1 to 12.

**9. Intck ('Interval' , 'From Date' , 'To Date')**= Number
Counts the number of interval boundaries between two dates.

**10. Intnx ('Interval' , 'Date' , Number)= Date**
Increment date by intervals.

**11. Datepart (Datetime):**
Extract date from SAS datetime value returns SAS date value.

**12. Timepart (Datetime):**
Extract time from SAS datetime value returns SAS datetime value.

**Video link: Youtube_Video**

**Q20: How can you pull four characters starting from third position of the given character variable?**

⇨ **We can extract certain part of the character variable using substr function.**

**Exract_Part = Substr(Charvar, Starting position, Required length);**

**Data creation:**

```
DATA PRODUCT;
```

```
INPUT PRODUCTID $20.;
DATALINES;
ABBSA1238Z1X253
ABQA123Q1X253
AEBSA1234Z2X252
ABGSA131Y253
ABBSD123Q1X253
ABBSW123004Z1X253
AQBSA1234Z2X2152
;
RUN;
```

**Output:**

Total rows: 7 Total columns: 1

| | PRODUCTID |
|---|---|
| 1 | ABBSA1238Z1X253 |
| 2 | ABQA123Q1X253 |
| 3 | AEBSA1234Z2X252 |
| 4 | ABGSA131Y253 |
| 5 | ABBSD123Q1X253 |
| 6 | ABBSW123004Z1X253 |
| 7 | AQBSA1234Z2X2152 |

**Four characters extract:**

```
DATA COMPANY_CODE;
SET PRODUCT;
C_CODE=SUBSTR(PRODUCTID, 3, 4);
RUN;
```

**Output:**

Total rows: 7 Total columns: 2

| | PRODUCTID | C_CODE |
|---|---|---|
| 1 | ABBSA1238Z1X253 | BSA1 |
| 2 | ABQA123Q1X253 | QA12 |
| 3 | AEBSA1234Z2X252 | BSA1 |
| 4 | ABGSA131Y253 | GSA1 |
| 5 | ABBSD123Q1X253 | BSD1 |
| 6 | ABBSW123004Z1X253 | BSW1 |
| 7 | AQBSA1234Z2X2152 | BSA1 |

**Video link: Youtube Video**

**Q21: How can you create first and last name column separately from customer's full name?**

⇨ **To extract certain word from character variable we use scan function with delimiters.**
**Word_extract=scan(Charvar, Nth word, 'Delimiter');**

```
DATA CUSTOMER1;
NPUT ID CUSTOMBER_NAME $30. AGE;
DATALINES;
101 Junier King Johnson          22
208 Gambir Bikram Marathi        90
333 Jimmy Oakson                 89
679 Krish S Dodge                77
876 Jani Jakson Junior Kakali    65
;
RUN;
```

**Output:**

Total rows: 5 Total columns: 3

| | ID | Customber_name | Age |
|---|---|---|---|
| 1 | 101 | Junier King Johnson | 22 |
| 2 | 208 | Gambir Bikram Marathi | 90 |
| 3 | 333 | Jimmy Oakson | 89 |
| 4 | 679 | Krish S Dodge | 77 |
| 5 | 876 | Jani Jakson Junior Kakali | 65 |

```
DATA CUSTOMER2;
SET CUSTOMER1;
FIRST_NAME=SCAN(CUSTOMBER_NAME,1,' ');
LAST_NAME=SCAN(CUSTOMBER_NAME,-1,' ');
RUN;
```

**Output:**

Total rows: 5 Total columns: 5

| | ID | Customber_name | Age | First_Name | Last_Name |
|---|---|---|---|---|---|
| 1 | 101 | Junier King Johnson | 22 | Junier | Johnson |
| 2 | 208 | Gambir Bikram Marathi | 90 | Gambir | Marathi |
| 3 | 333 | Jimmy Oakson | 89 | Jimmy | Oakson |
| 4 | 679 | Krish S Dodge | 77 | Krish | Dodge |
| 5 | 876 | Jani Jakson Junior Kakali | 65 | Jani | Kakali |

**Video link: Youtube Video**

**Q22: How can you save odd and even records of a dataset separately in SAS?**

⇨ **By using mod function with _N_, we can save odd and even records separately.**

**MOD(First argument, Second argument)**

```
DATA ODD_RECORD EVEN_RECORD;
SET SASHELP.CLASS;
IF MOD(_N_, 2)=1 THEN
    OUTPUT ODD_RECORD;
  ELSE
     OUTPUT EVEN_RECORD;
RUN;
```

**Odd Records:**

Total rows: 10 Total columns: 5 Rows 1-10

| | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69 | 112.5 |
| 2 | Barbara | F | 13 | 65.3 | 98 |
| 3 | Henry | M | 14 | 63.5 | 102.5 |
| 4 | Jane | F | 12 | 59.8 | 84.5 |
| 5 | Jeffrey | M | 13 | 62.5 | 84 |
| 6 | Joyce | F | 11 | 51.3 | 50.5 |
| 7 | Louise | F | 12 | 56.3 | 77 |
| 8 | Philip | M | 16 | 72 | 150 |
| 9 | Ronald | M | 15 | 67 | 133 |
| 10 | William | M | 15 | 66.5 | 112 |

**Even Records:**

Total rows: 9 Total columns: 5 Rows 1-9

| | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alice | F | 13 | 56.5 | 84 |
| 2 | Carol | F | 14 | 62.8 | 102.5 |
| 3 | James | M | 12 | 57.3 | 83 |
| 4 | Janet | F | 15 | 62.5 | 112.5 |
| 5 | John | M | 12 | 59 | 99.5 |
| 6 | Judy | F | 14 | 64.3 | 90 |
| 7 | Mary | F | 15 | 66.5 | 112 |
| 8 | Robert | M | 12 | 64.8 | 128 |
| 9 | Thomas | M | 11 | 57.5 | 85 |

**Video link: Youtube_Video**

## Q23: How can you extract last five characters from a character variable?

⇨ ***We can extract certain part of the character variable using substr function (Functions: Reverse, Substr, Strip).***

**Exract_Part = Substr(Charvar, Starting position, Required length);**

Data creation:

```
DATA PRODUCT;
INPUT PRODUCTID $20.;
DATALINES;
ABBSA1238Z1X253
ABQA123Q1X253
AEBSA1234Z2X252
ABGSA131Y253
ABBSD123Q1X253
ABBSW123004Z1X253
;
RUN;
```

**Output:**

Total rows: 6 Total columns: 1

| | PRODUCTID |
|---|---|
| 1 | ABBSA1238Z1X253 |
| 2 | ABQA123Q1X253 |
| 3 | AEBSA1234Z2X252 |
| 4 | ABGSA131Y253 |
| 5 | ABBSD123Q1X253 |
| 6 | ABBSW123004Z1X253 |

**Five characters extractction:**

```
DATA LAST_FIVE;
SET PRODUCT;
LAST_FIVE = REVERSE(SUBSTR(REVERSE(STRIP(PRODUCTID)), 1, 5));
RUN;
```

**Output:**

Total rows: 6 Total columns: 2

| | PRODUCTID | Last_Five |
|---|---|---|
| 1 | ABBSA1238Z1X253 | 1X253 |
| 2 | ABQA123Q1X253 | 1X253 |
| 3 | AEBSA1234Z2X252 | 2X252 |
| 4 | ABGSA131Y253 | 1Y253 |
| 5 | ABBSD123Q1X253 | 1X253 |
| 6 | ABBSW123004Z1X253 | 1X253 |

**Video link: Youtube_Video**

**Q24: Due to covid, loan pay date is extended by 18 months. How do you apply extension to every customer?**

⇨ **Increment date by certain interval is obtained by using date function Intnx.**
**Intnx ('Interval' , Date, Number)= Date**

```
DATA CUSTOMER;
INPUT ID LOANPAY_DATE;
INFORMAT LOANPAY_DATE DATE9.;
FORMAT LOANPAY_DATE DATE9.;
DATALINES;
223 23DEC2024
345 12JUL2025
778 09JAN2022
;
RUN;
```

Total rows: 3 Total columns: 2

| | ID | Loanpay_Date |
|---|---|---|
| 1 | 223 | 23DEC2024 |
| 2 | 345 | 12JUL2025 |
| 3 | 778 | 09JAN2022 |

```
DATA CUSTOMER;
SET CUSTOMER;
EXTN_PAY_DATE=INTNX('MONTH', LOANPAY_DATE, 18, 'SAMEDAY');
FORMAT EXTN_PAY_DATE DATE9.;
RUN;
```

Total rows: 3 Total columns: 3

| | ID | Loanpay_Date | EXTN_PAY_DATE |
|---|---|---|---|
| 1 | 223 | 23DEC2024 | 23JUN2026 |
| 2 | 345 | 12JUL2025 | 12JAN2027 |
| 3 | 778 | 09JAN2022 | 09JUL2023 |

**Video link: Youtube_Video**

**Q25: What are the commonly used SQL statements in SAS proc SQL ?**

***Proc SQL is used to process SQL in SAS. It is used to get back result from SQL query, create table and variables.***

***Most commonly used statements to fetch data from a table are as follow:***

```
Proc SQL;
      Select (All column(*)/ Column name)
            From (Table /Views name)
          Where (Expression)/Filters rows before grouping --Optional
          Group by (Column name)/Optional
          Having (Expression)/ Filters rows after grouping --Optional
          Order by (Column name)/ Sorting --Optional
;
Quit;
```

Sample code:

```
PROC SQL;
        SELECT NAME, AGE, WEIGHT
            FROM SASHELP.CLASS
            WHERE AGE GT 13
      ;
QUIT;
```

**Output:**

| Name | Age | Weight |
|---|---|---|
| Alfred | 14 | 112.5 |
| Carol | 14 | 102.5 |
| Henry | 14 | 102.5 |
| Janet | 15 | 112.5 |
| Judy | 14 | 90 |
| Mary | 15 | 112 |
| Philip | 16 | 150 |
| Ronald | 15 | 133 |
| William | 15 | 112 |

**Video link: Youtube_Video**

## Q26: How can you control variables and observation number in Proc Sql?

⇨ **Variables can be controlled by select statement and observation number is controlled by outobs option with proc sql statement.**

**Sample Code:**

```
PROC SQL FEEDBACK OUTOBS=5;
SELECT NAME, AGE, WEIGHT
FROM SASHELP.CLASS;
QUIT;
```

**Log:**

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         PROC SQL FEEDBACK OUTOBS=5;
70             SELECT NAME, AGE, WEIGHT
71             FROM SASHELP.CLASS;
NOTE: Statement transforms to:

        select CLASS.Name, CLASS.Age, CLASS.Weight
          from SASHELP.CLASS;

WARNING: Statement terminated early due to OUTOBS=5 option.
72             RUN;
NOTE: PROC SQL statements are executed immediately; The RUN statement has no effect.
73
74         OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
84
```

**Result:**

| Name | Age | Weight |
|---|---|---|
| Alfred | 14 | 112.5 |
| Alice | 13 | 84 |
| Barbara | 13 | 98 |
| Carol | 14 | 102.5 |
| Henry | 14 | 102.5 |

**Video link: Youtube Video**

## Q27: How can we select unique/ distinct rows (Or remove duplicate rows) from a dataset?

⇨ **To get distinct/unique records we use distinct word with select statement.**

**Sample data set.**

```
DATA EMPLOYEE;
INPUT E_ID NAME $ AGE SALARY;
DATALINES;
1 ROBERT  45 8750
7 SAM 49 8500
3 JESS  23 8700
4 MARK 65 8000
5 JENNY 51 7000
3 JESS  23 8700
4 MARK 65 8000
6 KEN 28 8700
;
RUN;
```

**EMPLOYEE Dataset:**

Total rows: 8 Total columns: 4

| | E_ID | NAME | AGE | SALARY |
|---|---|---|---|---|
| 1 | 1 | ROBERT | 45 | 8750 |
| 2 | 3 | JESS | 33 | 8700 |
| 3 | 3 | JESS | 23 | 8700 |
| 4 | 4 | MARK | 65 | 8000 |
| 5 | 4 | MARK | 65 | 8000 |
| 6 | 5 | JENNY | 51 | 7000 |
| 7 | 6 | KEN | 28 | 8700 |
| 8 | 7 | SAM | 49 | 8500 |

**Selecting distinct records:**

```
PROC SQL ;
SELECT DISTINCT E_ID, NAME, AGE, SALARY
FROM EMPLOYEE;
QUIT;
```

**Distinct Result:**

| E_ID | NAME | AGE | SALARY |
|---|---|---|---|
| 1 | ROBERT | 45 | 8750 |
| 3 | JESS | 23 | 8700 |
| 3 | JESS | 33 | 8700 |
| 4 | MARK | 65 | 8000 |
| 5 | JENNY | 51 | 7000 |
| 6 | KEN | 28 | 8700 |
| 7 | SAM | 49 | 8500 |

## Video link: Youtube Video

## Q28: How can you display Proc SQL result into SAS dataset table?

⇨ **Using create table statement before select statement in Proc SQL , result can be displayed into SAS Dataset.**

**Sample data set.**

```
DATA EMPLOYEE;
INPUT E_ID NAME $  AGE SALARY;
DATALINES;
1 ROBERT  45 8750
7 SAM 49 8500
3 JESS  23 8700
4 MARK 65 8000
5 JENNY 51 7000
6 KEN 28 8700
;
RUN;
```

**Sample data set.**

Total rows: 6 Total columns: 4 Rows 1-6

| | E_ID | NAME | AGE | SALARY |
|---|---|---|---|---|
| 1 | 1 | ROBERT | 45 | 8750 |
| 2 | 3 | JESS | 23 | 8700 |
| 3 | 4 | MARK | 65 | 8000 |
| 4 | 5 | JENNY | 51 | 7000 |
| 5 | 6 | KEN | 28 | 8700 |
| 6 | 7 | SAM | 49 | 8500 |

**Creating SAS Dataset (Table).**

```
PROC SQL ;
CREATE TABLE EMP_DATA AS
SELECT  *
FROM EMPLOYEE
WHERE AGE LT 50;
QUIT;
```

**Output:**

Total rows: 4 Total columns: 4 Rows 1-4

| | E_ID | NAME | AGE | SALARY |
|---|---|---|---|---|
| 1 | 1 | ROBERT | 45 | 8750 |
| 2 | 3 | JESS | 23 | 8700 |
| 3 | 6 | KEN | 28 | 8700 |
| 4 | 7 | SAM | 49 | 8500 |

**Video link: Youtube Video**

## 29: How can you rename and create new variable using Proc SQL?

⇨ **Variable (Column) can be renamed by using as between old column name and renamed column name. Similarly new column can be created by assigning new column name or expression in select statement.**

**Sample data set.**

```
DATA EMPLOYEE;
INPUT E_ID NAME $  AGE SALARY;
DATALINES;
1 ROBERT  45 8750
7 SAM 49 8500
3 JESS  23 8700
4 MARK 65 8000
5 JENNY 51 7000
6 KEN 28 8700
;
RUN;
```

**Output:**

Total rows: 6 Total columns: 4 Rows 1-6

| | E_ID | NAME | AGE | SALARY |
|---|---|---|---|---|
| 1 | 1 | ROBERT | 45 | 8750 |
| 2 | 7 | SAM | 49 | 8500 |
| 3 | 3 | JESS | 23 | 8700 |
| 4 | 4 | MARK | 65 | 8000 |
| 5 | 5 | JENNY | 51 | 7000 |
| 6 | 6 | KEN | 28 | 8700 |

**Creating SAS Dataset(Table).**

```
PROC SQL ;
CREATE TABLE EMP_DATA AS
SELECT  E_ID AS EMP_ID, AGE, SALARY, SALARY*.1 AS BONUS
FROM EMPLOYEE
WHERE AGE LT 50;
QUIT;
```

**Output:**

Total rows: 4 Total columns: 4 Rows 1-4

| | EMP_ID ▲ | AGE | SALARY | BONUS |
|---|---|---|---|---|
| 1 | 1 | 45 | 8750 | 875 |
| 2 | 3 | 23 | 8700 | 870 |
| 3 | 6 | 28 | 8700 | 870 |
| 4 | 7 | 49 | 8500 | 850 |

**Video link: Youtube_Video**

## Q30: How do you count number of employee per department using proc SQL?

⇨ **Employee count by department can can calculated by using count function and group by statement with department.**

**Sample data set.**

```
DATA NEW_HIRE;
INPUT EMPLOYEE_ID FIRST_NAME $ DEPARTMENT $12.;
DATALINES;
103 ALEXANDER IT
104 BRUCE MECHANICAL
105 DAVID IT
106 VALLI ELECTRICAL
107 DIANA MECHANICAL
;
RUN;
```

Total rows: 5 Total columns: 3

| | EMPLOYEE_ID | FIRST_NAME | DEPARTMENT |
|---|---|---|---|
| 1 | 103 | ALEXANDE | IT |
| 2 | 104 | BRUCE | MECHANICAL |
| 3 | 105 | DAVID | IT |
| 4 | 106 | VALLI | ELECTRICAL |
| 5 | 107 | DIANA | MECHANICAL |

**Department wise count:**

```
PROC SQL;
CREATE TABLE DEPT_COUNT AS
SELECT DEPARTMENT, COUNT(*) AS CNT
FROM NEW_HIRE
GROUP BY DEPARTMENT;
QUIT;
```

**Output:**

Total rows: 3 Total columns: 2 Rows 1-3

| | DEPARTMENT | CNT |
|---|---|---|
| 1 | ELECTRICAL | 1 |
| 2 | IT | 2 |
| 3 | MECHANICAL | 2 |

**Video link: Youtube_Video**

**Q31: How do you create a report showing all name starting with 'A' followed by any four characters?**

⇨ **We can use like operator to select pattern.**
**Where Variable_Name like ' _' or Variable_Name Like '%';**
Underscore (_) -> Any single character.
Percentage (%) -> Any number of characters

**Sample Code:**
**Title "Name starting with 'A' followed by any 4 Characters";**

```
PROC SQL;
SELECT * FROM SASHELP.CLASS
WHERE NAME LIKE 'A____' ;
QUIT;
```

**Extra Bonus:**
**Title "Starting any character with second letter 'a' followed by any number of characters";**

```
PROC SQL;
SELECT * FROM SASHELP.CLASS
WHERE NAME LIKE '_a%' ;
QUIT;
```

**Name starting with 'A' followed by any 4 Characters**

| Name | Sex | Age | Height | Weight |
|---|---|---|---|---|
| Alice | F | 13 | 56.5 | 84 |

**Starting any character with second letter 'a' followed by any number of characters**

| Name | Sex | Age | Height | Weight |
|---|---|---|---|---|
| Barbara | F | 13 | 65.3 | 98 |
| Carol | F | 14 | 62.8 | 102.5 |
| James | M | 12 | 57.3 | 83 |
| Jane | F | 12 | 59.8 | 84.5 |
| Janet | F | 15 | 62.5 | 112.5 |
| Mary | F | 15 | 66.5 | 112 |

**Video link: Youtube_Video**

## Q32: What do you mean by subquery or inner query in proc SQL?

⇨ **Query inside a main query is called subquery. It is also known as inner query or nested query. If the final result depends upon the result of other table then we use subquery. Subquery executes only one time.**

```
DATA EMPLOYEE;
  INPUT ID NAME $ SALARY DEPT_ID MANAGER_ID;
   DATALINES;
101 KEVIN 8900 20 103
102 ALEX 10000 23 103
103 KELLY 9800 20 104
104 NEENA 9000 25 109
105 JESS 9000 33 109
109 VANCE 10500 20 .
;
RUN;
```

**Output:**

Total rows: 6 Total columns: 5

| | ID | NAME | SALARY | DEPT_ID | MANAGER_ID |
|---|---|---|---|---|---|
| 1 | 101 | KEVIN | 8900 | 20 | 103 |
| 2 | 102 | ALEX | 10000 | 23 | 103 |
| 3 | 103 | KELLY | 9800 | 20 | 104 |
| 4 | 104 | NEENA | 9000 | 25 | 109 |
| 5 | 105 | JESS | 9000 | 33 | 109 |
| 6 | 109 | VANCE | 10500 | 20 | . |

```
DATA DEPARTMENT;
 INPUT DEPT_ID NAME $ 15.;
   DATALINES;
16 FINANCE
20 IT
23 MECHANICAL
```

```
25 ELECTRICAL
30 HR
;
RUN;
```

**Output:**

Total rows: 5 Total columns: 2

| | DEPT_ID | NAME |
|---|---|---|
| 1 | 16 | FINANCE |
| 2 | 20 | IT |
| 3 | 23 | MECHANICAL |
| 4 | 25 | ELECTRICAL |
| 5 | 30 | HR |

**Using Employee and Department table, lets extract all fields from employee table where employees work for IT department.**
**Sample code for Subquery/Inner query:**

```
PROC SQL;
CREATE TABLE EMPLOYEE_DEPARTMENT AS
SELECT * FROM EMPLOYEE
WHERE DEPT_ID =
        (
        SELECT DEPT_ID FROM DEPARTMENT
        WHERE NAME ='IT'
        );
QUIT;
```

**Output:**

Total rows: 3 Total columns: 5

| | ID | NAME | SALARY | DEPT_ID | MANAGER_ID |
|---|---|---|---|---|---|
| 1 | 101 | KEVIN | 8900 | 20 | 103 |
| 2 | 103 | KELLY | 9800 | 20 | 104 |
| 3 | 109 | VANCE | 10500 | 20 | . |

## Video link: Youtube_Video

## Q33: What are the different types of join used in proc SQL?

⇨ **Join is used to combine two or more than two tables horizontally. There are six types of commonly used joins:**

1. **Self Join**:
   Given table is joined with the same table -Example: Self join is used to identify manager's name of each employee from employee table.
2. **Inner Join**:
   This join is used to identify (Extract) all rows/records which are common to the two or more than two tables.
3. **Left Join**:
   Left join is used to select all the records from left table and records from right table which are common with the left table.
4. **Right Join**:

Right join is used to select all the records from right table and records from left table which are common with right table.

5. **Full outer Join**:

Full outer join is used to get all records from two or more than two tables.

6. **Cross Join:**

If we need to display all possible combination of information from two or more than two tables then cross join is used. In cross join it returns number of rows equal to the product of rows of the two or more tables.

**Video link: Youtube Video**

## Q34: How do you get manager's name of each employee using employee table?

⇨ **Employee table with self join is used to get manager's name for each employee. Referring Question # 32: Dataset Employee table is used as source tables.**

**Output:**

Total rows: 6 Total columns: 5

| | ID | NAME | SALARY | DEPT_ID | MANAGER_ID |
|---|---|---|---|---|---|
| 1 | 101 | KEVIN | 8900 | 20 | 103 |
| 2 | 102 | ALEX | 10000 | 23 | 103 |
| 3 | 103 | KELLY | 9800 | 20 | 104 |
| 4 | 104 | NEENA | 9000 | 25 | 109 |
| 5 | 105 | JESS | 9000 | 33 | 109 |
| 6 | 109 | VANCE | 10500 | 20 | . |

**Sample code using self join:**

```
PROC SQL;
   SELECT E.ID AS EMPLOYEE_ID
   , E.NAME AS EMPLOYEE_NAME
   , M.NAME AS MANAGER_NAME FROM EMPLOYEE E
   , EMPLOYEE M
 WHERE E.MANAGER_ID=M.ID;
QUIT;
```

**Output Result:**

| EMPLOYEE_ID | EMPLOYEE_NAME | MANAGER_NAME |
|---|---|---|
| 101 | KEVIN | KELLY |
| 102 | ALEX | KELLY |
| 103 | KELLY | NEENA |
| 104 | NEENA | VANCE |
| 105 | JESS | VANCE |

**Video link: Youtube Video**

## Q35: How do you get all employee information which exist in both employee and department table?

⇨ **Common information which exist in both tables can be obtained by using inner join with applying common variable on condition.**

**Output:**

Total rows: 6 Total columns: 5

| | ID | NAME | SALARY | DEPT_ID | MANAGER_ID |
|---|---|---|---|---|---|
| 1 | 101 | KEVIN | 8900 | 20 | 103 |
| 2 | 102 | ALEX | 10000 | 23 | 103 |
| 3 | 103 | KELLY | 9800 | 20 | 104 |
| 4 | 104 | NEENA | 9000 | 25 | 109 |
| 5 | 105 | JESS | 9000 | 33 | 109 |
| 6 | 109 | VANCE | 10500 | 20 | . |

**Output:**

Total rows: 5 Total columns: 2

| | DEPT_ID | NAME |
|---|---|---|
| 1 | 16 | FINANCE |
| 2 | 20 | IT |
| 3 | 23 | MECHANICAL |
| 4 | 25 | ELECTRICAL |
| 5 | 30 | HR |

**Referring Question # 32: Two dataset Employee and Department are used as two source tables.**

```
PROC SQL;
   SELECT E.*, D.NAME AS DEPT_NAME
   FROM EMPLOYEE E
   INNER JOIN DEPARTMENT D
   ON E.DEPT_ID=D.DEPT_ID;
QUIT;
```

**Result:**

| ID | NAME | SALARY | DEPT_ID | MANAGER_ID | DEPT_NAME |
|---|---|---|---|---|---|
| 101 | KEVIN | 8900 | 20 | 103 | IT |
| 102 | ALEX | 10000 | 23 | 103 | MECHANICAL |
| 103 | KELLY | 9800 | 20 | 104 | IT |
| 104 | NEENA | 9000 | 25 | 109 | ELECTRICAL |
| 109 | VANCE | 10500 | 20 | . | IT |

## Video link: Youtube Video

## Q36: How do you import excel data into SAS dataset?

⇨ **Excel data can be imported to SAS data set by using following procedure.**

**PROC IMPORT METHOD:**

```
PROC IMPORT DATAFILE=”Data file location with folder and file name”
OUT=     Output SAS dataset name
DBMS=    Database name
REPLACE; (It will replace if existed before)
SHEET="Sheet Name";(If multiple sheets are there)
```

GETNAMES=YES; (It will pickup field name from provided input data source.)

**SAMPLE CODE:**

```
PROC IMPORT DATAFILE="/home/joshid700/Interview/EMPLOYEE.XLSX"
OUT= EMP_DATA
DBMS= XLSX
REPLACE;
SHEET="EMPLOYEE";
GETNAMES=YES;
RUN;
```

**Output:**

Total rows: 6 Total columns: 5

| | ID | NA... | SALARY | DEPT_ID | MANAGER_ID |
|---|---|---|---|---|---|
| 1 | 101 | KEVIN | 8900 | 20 | 103 |
| 2 | 102 | ALEX | 10000 | 23 | 103 |
| 3 | 103 | KELLY | 9800 | 20 | 104 |
| 4 | 104 | NEENA | 9000 | 25 | 109 |
| 5 | 105 | JESS | 9000 | 33 | 109 |
| 6 | 109 | VANCE | 10500 | 20 | . |

**Importing excel data using Libname method.**

**LIBNAME METHOD:**

```
LIBNAME MYDATA XLSX “Data file location with folder and file name”;
DATA OUTPUT_DATA;
SET LIBNAME.SHEET_NAME;
RUN;
```

**SAMPLE CODE:**

```
LIBNAME MYDATA XLSX "/home/joshid700/Interview/EMPLOYEE.XLSX";
DATA DEPT_DATA;
SET MYDATA.DEPARTMENT;
RUN;
```

**Excel Data:**

| A | B | C | D | E |
|---|---|---|---|---|
| DEPT_ID | NAME | | | |
| 16 | FINANCE | | | |
| 20 | IT | | | |
| 23 | MECHANICAL | | | |
| 25 | ELECTRICAL | | | |
| 30 | HR | | | |
| | | | | |

EMPLOYEE DEPARTMENT

**Output:**

Total rows: 5 Total columns: 2

| | DEPT_ID ▲ | NAME |
|---|---|---|
| 1 | 16 | FINANCE |
| 2 | 20 | IT |
| 3 | 23 | MECHANICA |
| 4 | 25 | ELECTRICAL |
| 5 | 30 | HR |

## Video link: Youtube_Video

## Q37: How do you import text files into SAS dataset?

⇨ **By providing delimited file in dbms and providing respective delimiter value which is used to separate the variables.**

**Txt file with separating single space between variables can be imported by using dbms= dlm.**

Sample Class_A.txt

```
Name Sex Age Height Weight
John M 12 59 99.5
Joyce F 11 51.3 50.5
Judy F 14 64.3 90
Louise F 12 56.3 77
Mary F 15 66.5 112
Philip M 16 72 150
Robert M 12 64.8 128
```

**Importing .txt data(Separated by space):**

**Sample code :**

```
PROC IMPORT DATAFILE='/home/joshid700/Interview/Class_A.txt'
DBMS= DLM
OUT= CLASS_A
REPLACE;
GETNAMES=YES;
RUN;
```

| Name | Sex | Age ▲ | Height | Weight |
|---|---|---|---|---|
| John | M | 12 | 59 | 99.5 |
| Joyce | F | 11 | 51.3 | 50.5 |
| Judy | F | 14 | 64.3 | 90 |
| Louise | F | 12 | 56.3 | 77 |
| Mary | F | 15 | 66.5 | 112 |
| Philip | M | 16 | 72 | 150 |
| Robert | M | 12 | 64.8 | 128 |

**Txt file with separating special character or ‘,’ or ‘;’ or ‘.’ between variables can be imported by using dbms=dlm and delimiter=‘Special character’ or ‘,’ or ‘;’ or ‘.’ .**

**Class_B.txt:**

```
Name;Sex;Age;Height;Weight
Alfred;M;14;69;112.5
Alice;F;13;56.5;84
Barbara;F;13;65.3;98
Carol;F;14;62.8;102.5
Henry;M;14;63.5;102.5
James;M;12;57.3;83
Jane;F;12;59.8;84.5
Janet;F;15;62.5;112.5
```

**Importing .txt data(Separated by ';'):**
**Sample code :**

```
PROC IMPORT DATAFILE='/home/joshid700/Interview/Class_B.txt'
DBMS= DLM
OUT= CLASS_B
REPLACE;
GETNAMES=YES;
DELIMITER=';';
RUN;
```

Output:

| Name | Sex | Age ▲ | Height | Weight |
|---|---|---|---|---|
| Alfred | M | 14 | 69 | 112.5 |
| Alice | F | 13 | 56.5 | 84 |
| Barbara | F | 13 | 65.3 | 98 |
| Carol | F | 14 | 62.8 | 102.5 |
| Henry | M | 14 | 63.5 | 102.5 |
| James | M | 12 | 57.3 | 83 |
| Jane | F | 12 | 59.8 | 84.5 |
| Janet | F | 15 | 62.5 | 112.5 |

**Video link: Youtube Video**

## Q38: How do you connect Teradata DB with SAS studio/SAS EG?

⇨ **Following two methods are used to connect Teradata database:**

**Libname method**

LIBNAME ABCD Teradata server='Server name' User='User_Id' password='Password' Database='Database Name';

**Pulling data from Teradata:**

```
Data New_data;
Set ABCD.Teradata_Table_Name;
Where Var3='Value';
Run;
```

*In this method we do not touch Teradata table, it will pull whole data set into SAS then filter applies as a result it will take long time and uses more resources.*

**SQL pass-through facility:**

```
Proc sql;
Connect to Teradata (User='User_ID' password='password' Server='Server_Name'
connection=global mode=Teradata);
```

```
Create table New_Data as select * from connection to Teradata
        (select Var1, var2, var3
                From database.Teradata_Table_Name
                Where Var3='value');
Disconnect from Teradata;
Quit;
```

**(It has created volatile table inside server and can be reuse inside and outside the server)**

*In this method we connect Teradata database and touch Teradata table. We can select only require fields and apply row filter condition. In this method we don't need to extract all data to SAS and manipulate on it, so this method is more efficient and uses less resources.*

Note: Some SAS function like PUT,INPUT,INTCK can not be used.

**Video link: Youtube_Video**

## Q39: How do you create Permanent Table in Teradata using SAS?

⇨ **We can create permanent table in Teradata as follow:**

**Blank Table creation(No Data):**

```
Proc Sql;
Connect to Teradata (User='User_ID' password='password' Server='Server_Name'
connection=global mode=Teradata);
Execute (create multiset table Teradata_DB.Teradata_Table
        (
        ID              Varchar(20),
        Emp_Name        Varchar(50),
        Salary          Decimal(20,2),
        Date            Date
        )
Primary index(ID)) by Teradata ;
Disconnect from Teradata;
Quit;
```

***It will create empty table name Teradata_Table inside database (Teradata_DB) with primary index(ID).***

**Table creation using existing table(With Data):**

```
Proc sql;
Connect to Teradata (User='User_ID' password='password' Server='Server_Name'
connection=global mode=Teradata);
Execute (create table Teradata_DB1.Teradata_Table1 as select *
From connection to Teradata
        (
        Select *
        From Teradata_DB2.Teradata_Table2
        );
Disconnect from Teradata;
Quit;
```

***It will create Teradata table name Teradata_Table1 inside database (Teradata_DB1) which includes same fields and records as Teradata_Table2).***

**Video link: Youtube_Video**

## Q41: How do you update existing Teradata table using SAS Studio/SAS EG?

⇨ **To update Teradata table information (Data) we follow the following two steps process:**

**Step 1: Delete existing Table information:**

```
Proc Sql;
```

Connect to Teradata (User='User_ID' password='password' Server='Server_Name' connection=global mode=Teradata);
Execute (Delete from Teradata_DB1.Teradata_Table1 ) by Teradata ;
Disconnect from Teradata;
Quit;
**Note:**
**Let us consider Teradata_DB1.Teradata_Table1 has four variables: Var1,Var2,Var3 and Var4**
***In Step 1- It deletes all records from the table and remains empty table.***

**Step 2: Insert new records into the empty table:**
Proc sql;
Connect to Teradata (User='User_ID' password='password' Server='Server_Name' connection=global mode=Teradata);
Execute(
Insert into Teradata_DB1.Teradata_Table1
(Var1,Var2,Var3,Var4)
select Var1, var2, var3,Var4
From Teradata_DB2.Teradata_Table2) by Teradata;
Disconnect from Teradata;
Quit;
***Note:***
***Var1,Var2,Var3 and Var4 variable values from Teradata_Table2 are inserted into the table Teradata_Table1.***
**Video link: Youtube_Video**

## Q42: How do you connect Oracle DB with SAS Studio/SAS EG?

⇨ **Following two methods are used to connect Oracle database:**
**Libname method**
LIBNAME ABC Oracle path='Path name' Schema='Schema Name' User='User_Id' password='Password';
**Pulling data from Oracle:**
**Data New_Table2;**
**Set ABC.Oracle_Table1;**
**Where Var3='Value';**
**Run;**
*In this method we do not touch Oracle Database, it will pull all data set from schema into SAS then filter applies on it, as a result it will take long time and uses more resources.*

**SQL pass-through facility:**
**Proc sql;**
**Connect to Oracle(User='User_ID' password='password' Path='Path_Name');**
**Create table New_Table2 as select * from connection to Oracle**
**(select Var1, var2, var3**
**From Schema.Oracle_Table1**
**Where Var3='value');**
**Disconnect from Oracle;**
**Quit;**
***(It has created table in work library and can be use inside and outside the Oracle DB connection)***
*In this method we connect Oracle database and touch Oracle table. We can select only require fields and apply row filter condition. In this method we don't need to extract all data to SAS and manipulate on it, so this method is faster and uses less resources.*
Note: Some SAS function like PUT,INPUT,INTCK can not be used.
**Video link: Youtube_Video**

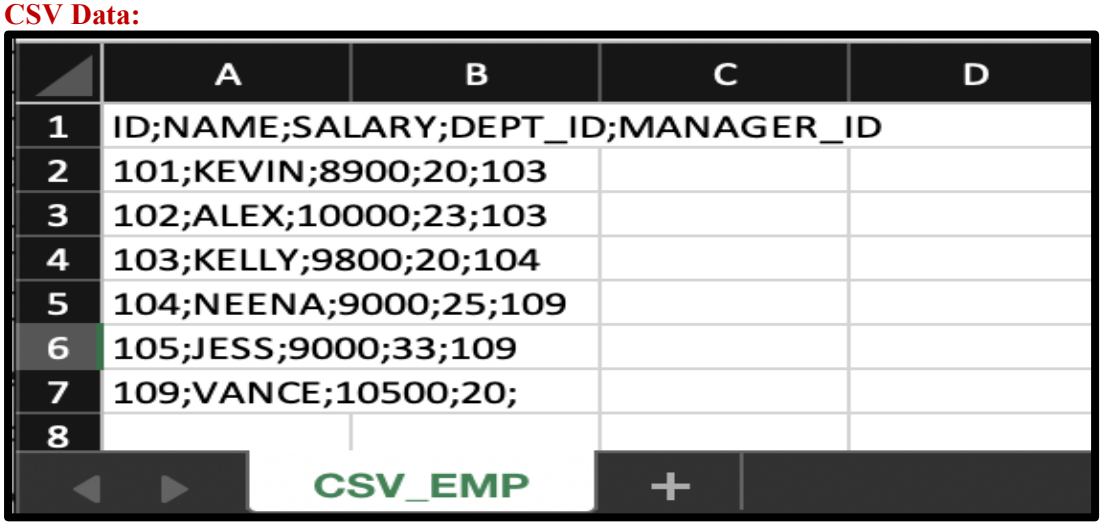
## Q43: How do you import csv data into SAS dataset?

⇨ **CSV data can be imported into SAS data set by using following procedure.**

**Importing csv file without delimiter:**

```
PROC IMPORT DATAFILE="/home/joshid700/Interview/class_data.csv"
OUT=Out_Class
DBMS=csv
REPLACE;
getnames=yes;
RUN;
```

**CSV Data:**




**Importing csv file with custom delimiter:**

```
proc import datafile="/home/joshid700/Interview/CSV_EMP.csv"
out= EMP_DATA
dbms=csv
replace;
delimiter=';';
getnames=yes;
run;
```

**Note:**

1. If the csv data is delimited by (;*,# $ % .@ !) then dbms=csv and delimiter='respective special character or symbol'
2. Similarly, if the csv data is delimited by single space then dbms=csv and delimiter=' ';

**Output EMP_Data:**

| ID | NAME | SALARY | DEPT_ID | MANAGER_ID |
|---|---|---|---|---|
| 101 | KEVIN | 8900 | 20 | 103 |
| 102 | ALEX | 10000 | 23 | 103 |
| 103 | KELLY | 9800 | 20 | 104 |
| 104 | NEENA | 9000 | 25 | 109 |
| 105 | JESS | 9000 | 33 | 109 |
| 109 | VANCE | 10500 | 20 | . |

**Video link**: [Youtube Video]

## Q44: How do you export SAS data into excel file format?

⇨ **One of the most frequently used procedure to export SAS data set to excel is proc export which is explained as follow.**
**Sample code:**

```
Proc export data = SAS_Table
outfile='file location /file_name.xlsx'
dbms=xlsx
replace;
sheet="Sheet_Name";
run;
```

**Partial class dataset:**

| | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69 | 112.5 |
| 2 | Alice | F | 13 | 56.5 | 84 |
| 3 | Barbara | F | 13 | 65.3 | 98 |
| 4 | Carol | F | 14 | 62.8 | 102.5 |
| 5 | Henry | M | 14 | 63.5 | 102.5 |
| 6 | James | M | 12 | 57.3 | 83 |

**Actual Sample Code:**
**Sample code:**

```
proc export data =sashelp.class
outfile='/home/joshid700/Interview/Export_Data.xlsx'
dbms=xlsx
replace;
sheet='Class_Data';
run;
```

**Excel Output:**

| A | B | C | D | E |
|---|---|---|---|---|
| Name | Sex | Age | Height | Weight |
| Alfred | M | 14 | 69 | 112.5 |
| Alice | F | 13 | 56.5 | 84 |
| Barbara | F | 13 | 65.3 | 98 |
| Carol | F | 14 | 62.8 | 102.5 |
| Henry | M | 14 | 63.5 | 102.5 |
| James | M | 12 | 57.3 | 83 |

**Video link: Youtube_Video**

## Q45: How do you export SAS data into excel workbook using libname method?

⇨ **We can export SAS data set into excel workbook using libname method only when SAS version is 9.4 M2 and higher.**
**Syntax to check version: %put my version of SAS is &sysvlong; (**my version of SAS is 9.04.01M7P080620)

**LIBNAME METHOD:**

```
LIBNAME abc XLSX “Data file location with folder and file name”;
DATA  abc.Sheet_Name;
SET SAS_Dataset;
RUN;
```

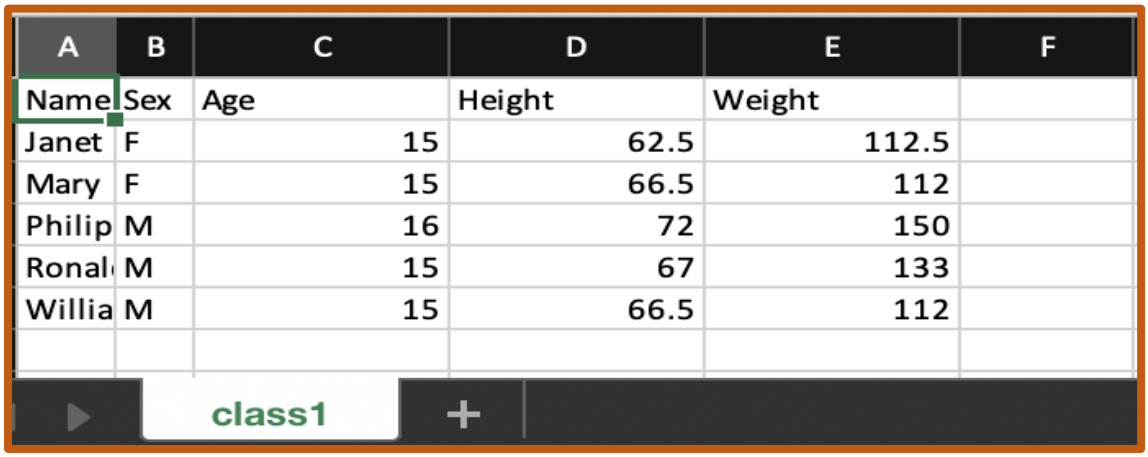
**Dataset:**

| | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Janet | F | 15 | 62.5 | 112.5 |
| 2 | Mary | F | 15 | 66.5 | 112 |
| 3 | Philip | M | 16 | 72 | 150 |
| 4 | Ronald | M | 15 | 67 | 133 |
| 5 | William | M | 15 | 66.5 | 112 |

**SAMPLE CODE:**

```
LIBNAME MYDATA XLSX "/home/joshid700/Interview/Class_Data.XLSX";
DATA Mydata.class1;
SET sashelp.class;
Where age gt 14;
RUN;
```

**Excel Sheet:**



| A | B | C | D | E | F |
|---|---|---|---|---|---|
| Name | Sex | Age | Height | Weight | |
| Janet | F | 15 | 62.5 | 112.5 | |
| Mary | F | 15 | 66.5 | 112 | |
| Philip | M | 16 | 72 | 150 | |
| Ronal | M | 15 | 67 | 133 | |
| Willia | M | 15 | 66.5 | 112 | |
| | | | | | |

class1

**Video link:** [Youtube Video]

## Q46: How do you export SAS report into excel worksheet?

⇨ **We can export SAS report into excel workbook using ODS(Output Delivery System) excel statement. Using this method, we can export SAS print result, frequency result, summary result, …**

Ods excel file =”Output file location with file name”;
SAS procedure steps
---
---
Ods excel close;

### Sample code:

```
ods excel file="/home/joshid700/Interview/Freq_Tbl.xlsx";
proc freq data =sashelp.shoes;
Table region*product/nopercent nocum norow nocol;
run;
ods excel close;
```

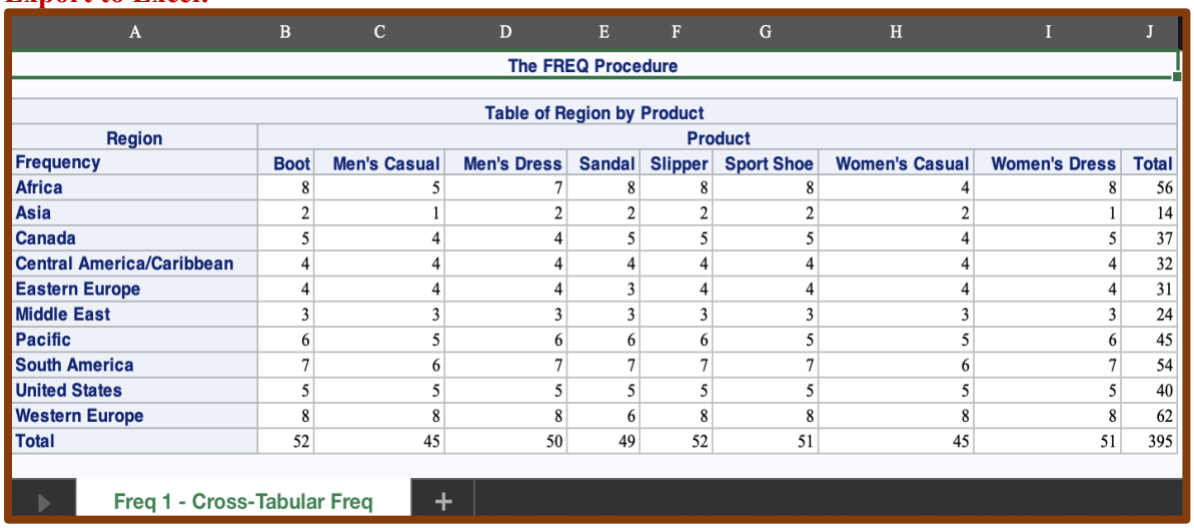
### Proc Freq Result:

The FREQ Procedure

Table of Region by Product

| Region | Product | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Boot | Men's Casual | Men's Dress | Sandal | Slipper | Sport Shoe | Women's Casual | Women's Dress | Total |
| Africa | 8 | 5 | 7 | 8 | 8 | 8 | 4 | 8 | 56 |
| Asia | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 14 |
| Canada | 5 | 4 | 4 | 5 | 5 | 5 | 4 | 5 | 37 |
| Central America/Caribbean | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 32 |
| Eastern Europe | 4 | 4 | 4 | 3 | 4 | 4 | 4 | 4 | 31 |
| Middle East | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 24 |
| Pacific | 6 | 5 | 6 | 6 | 6 | 5 | 5 | 6 | 45 |
| South America | 7 | 6 | 7 | 7 | 7 | 7 | 6 | 7 | 54 |
| United States | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 40 |
| Western Europe | 8 | 8 | 8 | 6 | 8 | 8 | 8 | 8 | 62 |
| Total | 52 | 45 | 50 | 49 | 52 | 51 | 45 | 51 | 395 |

**Export to Excel:**



The FREQ Procedure

Table of Region by Product

| Region | Product | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Frequency | Boot | Men's Casual | Men's Dress | Sandal | Slipper | Sport Shoe | Women's Casual | Women's Dress | Total |
| Africa | 8 | 5 | 7 | 8 | 8 | 8 | 4 | 8 | 56 |
| Asia | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 14 |
| Canada | 5 | 4 | 4 | 5 | 5 | 5 | 4 | 5 | 37 |
| Central America/Caribbean | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 32 |
| Eastern Europe | 4 | 4 | 4 | 3 | 4 | 4 | 4 | 4 | 31 |
| Middle East | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 24 |
| Pacific | 6 | 5 | 6 | 6 | 6 | 5 | 5 | 6 | 45 |
| South America | 7 | 6 | 7 | 7 | 7 | 7 | 6 | 7 | 54 |
| United States | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 40 |
| Western Europe | 8 | 8 | 8 | 6 | 8 | 8 | 8 | 8 | 62 |
| Total | 52 | 45 | 50 | 49 | 52 | 51 | 45 | 51 | 395 |

Freq 1 - Cross-Tabular Freq

**Video link: Youtube Video**

## Q47: What do you mean by SAS macro and what is its use?

⇨ **SAS MACROS are compiled programs that we can call in a submitted SAS programs or from command line.**

**SAS Macros are powerful feature of the SAS Programming which allow to avoid repetition of code and allow to use them again and again as needed. It decreases the lines of code drastically and code works faster and efficiently.**

**Sample Scenario 1:**
Suppose we have 1000 lines of SAS code and Start_Date is included in the code more than 200 places and which changes every week. It is tedious go to code every time and change date in each place. In such a scenario we define macro variable first and invoke that value for every Start_Date.
%let stdate='12FEB2023'd;
Then replace value of Start_Date by &stdate.
Start_Date=&stdate.
Then every week, just change stdate at one place then it will replace every Start_Date value.

**Sample Scenario 2:**
Suppose we need to calculate descriptive statistics of numeric fields of 10 different tables. (N, Min, Max, Mean Std Dev, Nmiss)
In this scenario, coding proc means procedure for 10 different table is not efficient way because it takes more time and code becomes very long and not efficient. Then one macro programming is sufficient and we pass parameter into it as required.

**Sample Scenario 3:**
Suppose you have monthly snapshot data in your database from Jan 2002 to Dec 2022 and your job is to compile all month and year data without duplicate records.
In this scenario selecting all fields and using union operator in each month is very tedious job because we have to write the same code 240 times which is time consuming task and not efficient. So, in such a scenario we use SAS macro with parameter (starting and ending data) which is sufficient to combine the data.

**Video link: Youtube Video**

## Q48: What do you mean by SAS Macro Variables and what are their types.

⇨ **SAS macro variable in SAS is a string variable that allows to modify the text in the SAS program. It is used to store a value in SAS which is always character and can hold character, number or text. Macro variables defined by the SAS programmer are called *user-defined macro variables* and those defined by the macro processor are called *automatic macro variables which are global macro variables*. We can define and use macro variables anywhere in SAS program, except within data lines.**

**User defined macro variables are two types: 1. Local Macro Variable: 2. Global Macro Variable:**

**Local Macro Variable:**
When the macro variable is defined inside the SAS macro code then it can be applicable only in that macro and value available only until macro stops executing .Such type of macro variable is called local macro variable.

```
%MACRO ABCD(DSN); /*ABCD is Macro and DSN is user defined local macro variable*/
/* DEFINING VARIABLE XYZ  INSIDE MACRO CODE SO IT IS LOCAL MACRO VARIABLE*/
%LET XYZ=12;              /*-> (USER DEFINED LOCAL MACRO VARIABLE.)*/
TITLE "REPORT PRINT LOCAL MACRO VARIABLE.";
PROC PRINT DATA =&DSN.;
WHERE AGE =&XYZ.;
RUN;
%MEND;
%ABCD(SASHELP.CLASS);
```

**Output Result:**

**REPORT PRINT LOCAL MACRO VARIABLE.**

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 6 | James | M | 12 | 57.3 | 83.0 |
| 7 | Jane | F | 12 | 59.8 | 84.5 |
| 10 | John | M | 12 | 59.0 | 99.5 |
| 13 | Louise | F | 12 | 56.3 | 77.0 |
| 16 | Robert | M | 12 | 64.8 | 128.0 |

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         %MACRO ABCD(DSN);              /*ABCD is Macro and DSN is user defined local macro variable*/
70         /* DEFINING VARIABLE XYZ  INSIDE MACRO CODE SO IT IS LOCAL MACRO VARIABLE*/
71         %LET XYZ=12;                   /*-> (USER DEFINED LOCAL MACRO VARIABLE.)*/
72         TITLE "REPORT PRINT LOCAL MACRO VARIABLE.";
73         PROC PRINT DATA =&DSN.;
74         WHERE AGE =&XYZ.;
75         RUN;
76         %MEND;
77         %ABCD(SASHELP.CLASS);

NOTE: There were 5 observations read from the data set SASHELP.CLASS.
      WHERE AGE=12;
```

**Global Macro Variable:**

When a macro variable is defined in a statement outside the macro definition or automatically created by the sas session then the macro variable can be used any where in the program and gets removed at the end of the session. Such type of macro variable is called global macro variable.

**Sample code:**

```
%LET PQR=14;                /*->(USER DEFINED GLOBAL MACRO VARIABLE.)*/
%MACRO ABCD(DSN);   /*ABCD is Macro and DSN is user defined local macro
variable*/
TITLE "REPORT PRINT GLOBAL MACRO VARIABLE.";
TITLE2 "PRINTED ON &SYSDATE9."; /*->(SYSTEM DEFINED GLOBAL VARIABLE)
*/
PROC PRINT DATA =&DSN.;
WHERE AGE =&PQR.;
RUN;
%MEND;
%ABCD(SASHELP.CLASS);
```

**Checking local & global macro variable outside macro:**

```
PROC PRINT DATA =SASHELP.CLASS;
WHERE AGE =&Xyz.;
/WHERE AGE=&Pqr.;*/
RUN;
```

**Output:**

REPORT PRINT GLOBAL MACRO VARIABLE.
"PRINTED ON 19FEB2024"

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69.0 | 112.5 |
| 4 | Carol | F | 14 | 62.8 | 102.5 |
| 5 | Henry | M | 14 | 63.5 | 102.5 |
| 12 | Judy | F | 14 | 64.3 | 90.0 |

**Checking Local Macro variable outside macro:**

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         /*Checking local macro variable outside macro"*/
70         Proc print data =sashelp.class;
71         where age =&xyz.;
                       _
                       22
                       76
WARNING: Apparent symbolic reference XYZ not resolved.
ERROR: Syntax error while parsing WHERE clause.
ERROR 22-322: Syntax error, expecting one of the following: a name, a
              a missing value, (, *, +, -, :, INPUT, NOT, PUT, ^, ~.
ERROR 76-322: Syntax error, statement will be ignored.
72         run;

NOTE: The SAS System stopped processing this step because of errors.
```

**Video link:** Youtube_Video

## Q49: How can you check the types of Macro Variable in your SAS program?

⇨ **To check Local Macro Variable, we write %put _local_ ; inside the macro code where local macro variable is defined and to check global macro variable we write %put _global_; any where in the code.**
**Alternatively, we can write %put _all_; inside the macro that displays local macro variable defined in that macro and displays all automatic and global macro variables.**
**Checking types of macro variable in SAS program.**

```
%let A ='Toyota';
%macro Test(dsn, var);
   %let B ='Sedan';
   proc freq data=&dsn;
      Table &var;
   where Type=&B.;
   run;
  %put _local_;
/*%put _all_;*/
%mend;
%Test(sashelp.cars, Origin);
%put _global_;
```

**Output Result:**

The FREQ Procedure

| Origin | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
|---|---|---|---|---|
| Asia | 94 | 35.88 | 94 | 35.88 |
| Europe | 78 | 29.77 | 172 | 65.65 |
| USA | 90 | 34.35 | 262 | 100.00 |

**Log :**

```
TEST B 'Sedan'
TEST DSN sashelp.cars
TEST VAR Origin
78          %put _global_;
GLOBAL A 'Toyota'
GLOBAL CLIENTMACHINE IP68100136140.DC.DC.COX.NET
GLOBAL GRAPHINIT GOPTIONS RESET=ALL GSFNAME=_GSFNAME;
GLOBAL GRAPHTERM GOPTIONS NOACCESSIBLE;
GLOBAL OLDPREFS homejoshid700/.wepreferences
```

**Video link:** Youtube_Video

## Q50: What are the different ways to create SAS Macro variables?

⇨ **There are five different ways to create the SAS Macro variables. Briefly defined as follow:**

**1. Using %let statement:**

By using %let statement we can define macro variable inside and outside the SAS MACRO code.

Syntax:

%let Macro_Var_Name=value;

**2. Call SYMPUT Routine:**

**This is a procedure (Routine) that can accept arguments but does not return any values.**

By using Call SYMPUT we can change data step variable into MACRO Variable.

Syntax:

Call SYMPUT('Macro_Var_Name', Value);

**3. Iterative %Do:**

For dynamic programming we use macro %DO loop. The iterative %DO loop is similar to DO statement used in data step but %DO can be used anywhere inside macro and used for increment macro variable.

Syntax:

%DO Macro_Var_Name=Start_Value %TO Stop_Value [%BY Increment-Which is optional];

SAS code

%END;

**4. Macro Parameters:**

This is a way to create a piece of code that can be run multiple times. A macro needs to be started %macro and ended %mend and you can use parameters to change values for each run.

%MACRO Macro_Name(Macro_var1_Name, Macro_var2_Name);

SAS code

---

---

%MEND;

%Macro_Name(Macro_var1_Value, Macro_Var2_Value);

**5. SAS Macro variable in Proc Sql using INTO clause:**

Syntax:

PROC SQL;

SELECT Macro_Specification(Variable selection)

INTO: Macro_Var_Name

```
FROM Source_Table;
Quit;
```

**Video link: Youtube_Video**

**Q51: Describe a scenario where you had used a macro. How did you go about building the macro?**

⇨ **When a business needs to get a result in which same SAS code is needed to run multiple times then SAS Macro is applicable. SAS MACRO feature avoids repetition of the code and allow them to use again and again as needed**.

Suppose we need to calculate descriptive statistics of numeric fields of 10 different tables. (N, Nmiss, Min, Max, Mean Median Std Dev)
In this scenario instead of writing SAS code to get descriptive statistics 10 times we use following macro and provide tables name as parameter.

**Creating Macro:**

```
%MACRO STAT_MACRO(DSN);
PROC MEANS DATA =&DSN. N Nmiss Min Max Mean Median Stddev;
RUN;
%MEND;
```

**Calling macro:**

```
%STAT_MACRO(SASHELP.CLASS);
%STAT_MACRO(SASHELP.FISH);
%STAT_MACRO(SASHELP.SHOES);
%STAT_MACRO(SASHELP.CARS);
```

The MEANS Procedure

| Variable | N | N Miss | Minimum | Maximum | Mean | Median | Std Dev |
|---|---|---|---|---|---|---|---|
| Age | 19 | 0 | 11.0000000 | 16.0000000 | 13.3157895 | 13.0000000 | 1.4926722 |
| Height | 19 | 0 | 51.3000000 | 72.0000000 | 62.3368421 | 62.8000000 | 5.1270752 |
| Weight | 19 | 0 | 50.5000000 | 150.0000000 | 100.0263158 | 99.5000000 | 22.7739335 |

The MEANS Procedure

| Variable | N | N Miss | Minimum | Maximum | Mean | Median | Std Dev |
|---|---|---|---|---|---|---|---|
| Weight | 158 | 1 | 0 | 1650.00 | 398.6955696 | 272.5000000 | 359.0862037 |
| Length1 | 159 | 0 | 7.5000000 | 59.0000000 | 26.2471698 | 25.2000000 | 9.9964412 |
| Length2 | 159 | 0 | 8.4000000 | 63.4000000 | 28.4157233 | 27.3000000 | 10.7163281 |
| Length3 | 159 | 0 | 8.8000000 | 68.0000000 | 31.2270440 | 29.4000000 | 11.6102458 |
| Height | 159 | 0 | 1.7284000 | 18.9570000 | 8.9709937 | 7.7860000 | 4.2862076 |
| Width | 159 | 0 | 1.0476000 | 8.1420000 | 4.4174855 | 4.2485000 | 1.6858039 |

The MEANS Procedure

| Variable | Label | N | N Miss | Minimum | Maximum | Mean | Median | Std Dev |
|---|---|---|---|---|---|---|---|---|
| Stores | Number of Stores | 395 | 0 | 1.0000000 | 41.0000000 | 11.6481013 | 10.0000000 | 8.8736315 |
| Sales | Total Sales | 395 | 0 | 325.0000000 | 1298717.00 | 85700.17 | 38912.00 | 129107.23 |
| Inventory | Total Inventory | 395 | 0 | 374.0000000 | 2881005.00 | 250898.86 | 118849.00 | 351514.63 |
| Returns | Total Returns | 395 | 0 | 10.0000000 | 57362.00 | 2967.32 | 1438.00 | 4611.74 |

The MEANS Procedure

| Variable | Label | N | N Miss | Minimum | Maximum | Mean | Median | Std Dev |
|---|---|---|---|---|---|---|---|---|
| MSRP | | 428 | 0 | 10280.00 | 192465.00 | 32774.86 | 27635.00 | 19431.72 |
| Invoice | | 428 | 0 | 9875.00 | 173560.00 | 30014.70 | 25294.50 | 17642.12 |
| EngineSize | Engine Size (L) | 428 | 0 | 1.3000000 | 8.3000000 | 3.1967290 | 3.0000000 | 1.1085947 |
| Cylinders | | 426 | 2 | 3.0000000 | 12.0000000 | 5.8075117 | 6.0000000 | 1.5584426 |
| Horsepower | | 428 | 0 | 73.0000000 | 500.0000000 | 215.8855140 | 210.0000000 | 71.8360316 |
| MPG_City | MPG (City) | 428 | 0 | 10.0000000 | 60.0000000 | 20.0607477 | 19.0000000 | 5.2382176 |
| MPG_Highway | MPG (Highway) | 428 | 0 | 12.0000000 | 66.0000000 | 26.8434579 | 26.0000000 | 5.7412007 |
| Weight | Weight (LBS) | 428 | 0 | 1850.00 | 7190.00 | 3577.95 | 3474.50 | 758.9832146 |
| Wheelbase | Wheelbase (IN) | 428 | 0 | 89.0000000 | 144.0000000 | 108.1542056 | 107.0000000 | 8.3118130 |
| Length | Length (IN) | 428 | 0 | 143.0000000 | 238.0000000 | 186.3621495 | 187.0000000 | 14.3579913 |

**Video link:** [Youtube Video]

**Q52: Can you explain a situation where you have used proc sql select into clause to create macro variable?**

⇨ **If we need to use certain value or values from the variable of a table to further processing steps than we can change that value to macro variable value and use it back where necessary.**

While updating MDT, if the code runs successfully then process_Status_CD changes to dot(.). If the program throws error than process_Status_CD change to corresponding number. In such scenario, after fixing error we need to change number to 0 that means it is ready to new run. Sometimes it is not efficient to go to check MDT_Process_Status table, identify corresponding id and change the status code. In this situation, it is better to run just below code that changes the status.

**Output:**

Total rows: 12 Total columns: 2

| | ID | process_Status_CD |
|---|---|---|
| 1 | 231 | 44 |
| 2 | 230 | . |
| 3 | 229 | . |
| 4 | 228 | . |
| 5 | 227 | . |
| 6 | 226 | . |
| 7 | 225 | . |
| 8 | 224 | . |
| 9 | 223 | . |

**Sending corresponding ID to macro variable value:**

```
proc sql;
select ID into: Proc_ID
from DRJ.MDT_Process_Status
where process_Status_CD not in (.,0);
Quit;
%put &Proc_ID.;
```

**Calling and using macro variable value:**

```
proc sql;
update DRJ.MDT_Process_Status
set process_Status_CD =0
where id=&Proc_ID.;
quit;
```

**Output:**

Total rows: 12 Total columns: 2

| | ID | process_Status_CD |
|---|---|---|
| 1 | 231 | 0 |
| 2 | 230 | . |
| 3 | 229 | . |
| 4 | 228 | . |
| 5 | 227 | . |
| 6 | 226 | . |
| 7 | 225 | . |
| 8 | 224 | . |
| 9 | 223 | . |

**Video link: Youtube_Video**

**Q53: How do you combine 20 years monthly data to the history table?**

⇨ **There are multiple ways to combined monthly data to history table but one of the efficient way is using SAS macro code. In this process we use macro code with proc append procedure so that every month data will append one after other.**

**Sample code to combined monthly data.**

```
/*Define libname*/
LIBNAME ABC "/home/joshid700/Interview/Monthly_History";
/*SAS MACRO FOR MONTHLY TO HISTORY */
%MACRO STUDENT_LOOP(START=, END=, BASE=);
  PROC DELETE DATA=&BASE.;
    RUN;
/*Set variable used in the loop to the inputs*/
%DO %UNTIL (&START.>&END.);
      %PUT CERATING DATA FOR: &START.;
      PROC SQL;
            CREATE TABLE STUDENT_LIST AS SELECT * FROM
            ABC.STUDENT_LIST_&START.;
      QUIT;
      PROC APPEND BASE=&BASE. DATA=STUDENT_LIST FORCE;
      RUN;
 /*While jumping from December to January, difference between 201001-200912 or 201101-
201012 ..=89,
 so following trick is applicable-  Example substr(201008,5,2)=08*/
      %IF %SUBSTR(&START, 5, 2)=12 %THEN
        %LET START=%EVAL(&START.+89);
      %ELSE
        %LET START=%EVAL(&START.+1);
   %END;
```

```
%MEND;
%STUDENT_LOOP(START=200906, END=201006, BASE=COMBINED_OUTPUT);
```

**Sample Output:**

Total rows: 247 Total columns: 5

| | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69 | 112.5 |
| 2 | Alice | F | 13 | 56.5 | 84 |
| 3 | Barbara | F | 13 | 65.3 | 98 |
| 4 | Carol | F | 14 | 62.8 | 102.5 |
| 5 | Henry | M | 14 | 63.5 | 102.5 |
| 6 | James | M | 12 | 57.3 | 83 |
| 7 | Jane | F | 12 | 59.8 | 84.5 |
| 8 | Janet | F | 15 | 62.5 | 112.5 |

**Log:**

```
NOTE: Appending WORK.STUDENT_LIST to WORK.COMBINED_OUTPUT.
NOTE: There were 19 observations read from the data set WORK.STUDENT_LIST.
NOTE: 19 observations added.
NOTE: The data set WORK.COMBINED_OUTPUT has 247 observations and 5 variables.
NOTE: PROCEDURE APPEND used (Total process time):
      real time               0.00 seconds
      user cpu time           0.00 seconds
      system cpu time         0.00 seconds
      memory                  1111.21k
      OS Memory               21932.00k
      Timestamp               03/04/2024 12:35:10 AM
      Step Count                        403  Switch Count  0
```

**Video link: Youtube_Video**

## Q54: What is the use of %SYSFUNC Function in SAS Macro and give an example how to use it?

⇨ **There are number of Base SAS functions which are not directly available in SAS MACRO, %Sysfunc is used to enable those functions to make them work in the SAS macro code. But following functions are not available with %Sysfunc: Dif, input, put, dim, lag, resolve, symget, lbound, hbound**

**Sample code1:**

```
DATA _NULL_;
%LET TODAY_DATE=%SYSFUNC(TODAY(), DATE9.);
RUN;
%PUT &TODAY_DATE;
```

**Sample code2:**

**Checking the existence of SAS Data:**

```
%MAcro Data_check(DSN);
%if %sysfunc(exist(&DSN)) %then
   %do;
   Title "Report printed on %sysfunc(Today(), Date9.)";
      proc print data=&dsn;
      run;
    %end;
```

```
    %else %put The data &DSN does not exist.;
%mend;
%Data_check(sashelp.Class);
%Data_check(sashelp.ABC);
```

**Log Sample Code1:**

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         %PUT &TODAY_DATE;
05MAR2024
70
71         OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
81
```

**Output(Sample code2- DSN: SASHELP.CLASS)**

Report printed on 05MAR2024

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Alfred | M | 14 | 69.0 | 112.5 |
| 2 | Alice | F | 13 | 56.5 | 84.0 |
| 3 | Barbara | F | 13 | 65.3 | 98.0 |
| 4 | Carol | F | 14 | 62.8 | 102.5 |
| 5 | Henry | M | 14 | 63.5 | 102.5 |

**Log: (Sample code2- DSN: SASHELP.ABC)**

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         %Data_check(sashelp.abc);
The data sashelp.abc does not exist.
70
71
72         OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
82
```

**Video link: Youtube Video**

## Q55: What are the commonly used SAS Macro functions?

⇨ ***Most commonly used SAS MACRO Functions are as follows:***

%EVAL, %SYSEVALF, %INDEX, %LENGTH, %SUBSTR, %UPCASE, %LOWCASE, %SCAN, %SYSFUNC, %STR, %NRSTR, %QUOTE, %NRQUOTE, %UNQUOTE.

**1. %EVAL:**

*Evaluates arithmetic and logical expressions using integer value.*

*Data _Null_;*
*%let x=7;*
*%let y=8;*
*%let z=%eval(&x+&y);*
*Result: Z=15  Without %Eval function Z=(7+8)*

2. **%SYSEVALF:**

*Evaluates arithmetic and logical expression using decimal values(Floating points).*

*Data _null_;*
*%let l=7.2;*
*%let m=3.25 ; %let n=%sysevalf(&l-&m);*
*Result =3.95  Without %sysevalf n=(7.2-3.25)*

3. **%INDEX:**

*Returns the positions of the first character of a string.*

*%let a=SAS Macro function;*
*%let b=%index(&a,c);*
*%put c position at &b;*
*Result: c position at 7.*

**4. %LENGTH:**

*Returns the length of a string.*

*%let x=Maryland;*
*%put The length of &x is %length(&x);*
*Result: Length of Maryland is 8.*

**5. %SUBSTR:**

*It will extract subcomponent of string variable.*

*%let x=Maryland;*
*%let y=%substr(&x,5,4);*
*%put value=&y;*
*Result: value=land.*

6. **%UPCASE**

*Converts resolved macro string value to upper case.*

*%let x=Maryland;*
*%let y=%upcase(&x);*
*%put value=&y;* *Result: value=MARYLAND*

**7. %LOWCASE:**

*Converts resolved macro string value to lower case*

*%let x=Maryland;*
*%let y=%lowcase(&x);*
*%put value=&y;* *Result: value=Maryland*

## Video link: Youtube Video

## Q56: What are the commonly used SAS Macro functions?

***(Remaining 7 Functions)***

⇨ ***Most commonly used MACRO Functions are as follows(Part-2 of 2)***

**8. %SCAN:**

*It extracts nth word of the string.*

*%let Var= Var1 Var2 Var3;*
*%let Final_Var =%scan(&Var,2);*
*%put &Final_Var;* *Result: Var2*

**9. %SYSFUNC:**

*Executes base SAS functions or user defined functions.*

*%let string1=X01-X20;*
*%let string1=%sysfunc(translate(&string1, Y,X));*
*%put translated result=&string1; Result: X01-X20 changes to Y01-Y20*

**10. %STR :**

*During compilation of macro definition %STR mask(removes the normal meaning ) special character and mnemonic operators(+ − * / < > = ¬ ^ ~ ; , # blank AND OR NOT EQ NE LE LT GE GT IN, with pair of ‘ ,”,( )*

*%let program1=proc print ; run;*
*%put &program1;* *Result: proc print*
*%let program2=%str(proc print ; run;);*
*%put &program2;* *Result: proc print; run;*

**11. %NRSTR:**

*In addition to above mention special character and mnemonic operators(With %STR), %NRSTR masks & and %*

*%put "Without using result: &SYSDATE9 and %sysfunc(Today(),Date9.)";*
*%put "With using result: %nrstr(&SYSDATE9) and %nrstr(%sysfunc(Today(),Date9.))";*

"Without using result: 16MAR2024 and 16MAR2024"
"With using result: &SYSDATE9 and %sysfunc(Today(),Date9.)"

**12. %QUOTE**

*Functions work same as %STR but it works during the execution of a macro language.*

```
%macro dept1(state);
/* without %quote -- problems might occur  but with %quote(&state) resolves problem */
%if &state=va %then %put Virginia DMV;
%else %put Other DMV;
 %mend dept1;
%dept1(or)
```

**13. %NRQUOTE:**

*Functions work same as %NRSTR but it works during the execution of a macro language.*

**14. %UNQUOTE:**

*It helps restore the meaning of any special characters and mnemonics masked by quoting during the macro execution*

```
%let val = UseofUnquote;
%let testval = %str(%'&val%');
data _null_;
 val = &testval;
  put 'VAL =' val;
run;
----------------------------------------
%let val = UseofUnquote;
%let testval = %str(%'&val%');
data _null_;
  val = %unquote(&testval);
  put 'VAL =' val;
run;
```

## Video link: Youtube_Video

## Q57: What are the differences between MPRINT, MLOGIC and SYMBOLGEN?

⇨ ***MPRINT, SYMBOLGEN & MLOGIC all three options are used for SAS MACRO debugging.***

**MPRINT:**

**Mprint displays all the SAS statements of the resolved macro code and used for debugging macros.**

```
OPTIONS MPRINT NOMLOGIC NOSYMBOLGEN;
%MACRO TEST (INPUT,OUTPUT);
PROC MEANS DATA = &INPUT NOPRINT;
VAR MSRP;
OUTPUT OUT = &OUTPUT MEAN= ;
RUN;
%MEND;
%TEST(SASHELP.CARS,TEST);
```

***Log Window Info*:**

```
70          OPTIONS MPRINT NOMLOGIC NOSYMBOLGEN;
71          %MACRO TEST (INPUT,OUTPUT);
72          PROC MEANS DATA = &INPUT NOPRINT;
73          VAR MSRP;
74          OUTPUT OUT = &OUTPUT MEAN= ;
75          RUN;
76          %MEND;
77          %TEST(SASHELP.CARS,TEST);
MPRINT(TEST):   PROC MEANS DATA = SASHELP.CARS NOPRINT;
MPRINT(TEST):   VAR MSRP;
MPRINT(TEST):   OUTPUT OUT = TEST MEAN= ;
MPRINT(TEST):   RUN;

NOTE: There were 428 observations read from the data set SASHELP.CARS.
```

**SYMBOLGEN:**
**Symbolgen displays the result of resolving macro variable reference to the SAS log and used for debugging macros.**

```
OPTIONS SYMBOLGEN NOMPRINT NOMLOGIC;
%MACRO TEST (INPUT =,OUTPUT=);
PROC MEANS DATA = &INPUT NOPRINT;
VAR MSRP;
OUTPUT OUT = &OUTPUT MEAN= ;
RUN;
%MEND;
%TEST(INPUT=SASHELP.CARS,OUTPUT=TEST);
```

**Log Window Info:**

```
82          OPTIONS SYMBOLGEN NOMPRINT NOMLOGIC;
83          %MACRO TEST (INPUT =,OUTPUT=);
84          PROC MEANS DATA = &INPUT NOPRINT;
85          VAR MSRP;
86          OUTPUT OUT = &OUTPUT MEAN= ;
87          RUN;
88          %MEND;
89          %TEST(INPUT=SASHELP.CARS,OUTPUT=TEST);
SYMBOLGEN:  Macro variable INPUT resolves to SASHELP.CARS
SYMBOLGEN:  Macro variable OUTPUT resolves to TEST

NOTE: There were 428 observations read from the data set SASHELP.CARS.
```

**MLOGIC/NOMLOGIC: Traces execution information in SAS log/ Does not trace execution.**

**MLOGIC:**
**When we use MLOGIC option, it causes the macro processor to trace its execution and to write the trace information to the SAS log.**
**It is used to debug macro. Each line starts with MLOGIC and micro processor displays following messages:**
**1.Beginning execution.**
**2. Value of macro parameter at invocation.**
**3. Execution of each macro program statement.**
**4. Displays whether each %IF condition is True or False.**
**5. Ending Execution.**

```
OPTIONS MLOGIC NOMPRINT NOSYMBOLGEN;
%macro dept1(state);
```

```
%if %quote(&state)=va %then %put Virginia DMV;
%else %put Other DMV;
%mend dept1;
%dept1(or);
```

***Log Window info:***

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
68
69         OPTIONS MLOGIC;
70         %macro dept1(state);
71         %if %quote(&state)=va %then %put Virginia DMV;
72         %else %put Other DMV;
73         %mend dept1;
74         %dept1(or);
MLOGIC(DEPT1):  Beginning execution.
MLOGIC(DEPT1):  Parameter STATE has value or
MLOGIC(DEPT1):  %IF condition %quote(&state)=va is FALSE
MLOGIC(DEPT1):  %PUT Other DMV
Other DMV
MLOGIC(DEPT1):  Ending execution.
```

***Conclusion:***

**MPRINT** option prints all SAS code with resolved macro in the Log window.
**SYMBOLGEN** option prints resolved macro variable message in the Log window.
**MLOGIC** option identifies and prints the macro logic information in the Log window

**Video link: Youtube_Video**

## Q58: How does multiple ampersand (&) macro variable resolve? Where do you use it?

⇨ **Single ampersand (&) is used to invoke/resolve the value of a macro variable. Sometimes multiple ampersands come in application, which is used to allow the value of a macro variable to become another macro variable reference. The macro variable reference will be rescanned until the macro variable is resolved.**

**Following rules are applicable to resolve multiple ampersand:**
Rule1: The macro processor reads a macro variable from left to right(Resolves left ampersand first then resolves right ampersand)
%put value=&BOOK&STORY.; => STORYFUNNY

```
74          %put value=&book&story.;
SYMBOLGEN:  Macro variable BOOK resolves to STORY
SYMBOLGEN:  Macro variable STORY resolves to FUNNY
value=STORYFUNNY
```

Rule2: For multiple ampersand, two ampersand(&&) resolves to one(&) during the macro resolution process.

%put value1=&&&STORY&BOOK.;
=> &FUNNYSTORY=>FICTION

```
76          %put value1=&&&story&book.;
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable STORY resolves to FUNNY
SYMBOLGEN:  Macro variable BOOK resolves to STORY
SYMBOLGEN:  Macro variable FUNNYSTORY resolves to FICTION
value1=FICTION
```

Rule3: While working for multiple ampersand, the re-scan rule applies until no more ampersands are left to resolve.

%put value4=&&&&&&STORY&&BOOK;
=> &&&STORY&BOOK => &FUNNYSTORY => FICTION

```
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable STORY resolves to FUNNY
SYMBOLGEN:  Macro variable BOOK resolves to STORY
SYMBOLGEN:  Macro variable FUNNYSTORY resolves to FICTION
value4=FICTION
```

**Sample Code:**

```
OPTIONS SYMBOLGEN;
%LET BOOK = STORY;
%LET STORY = FUNNY;
%LET FUNNYSTORY = FICTION;
```

In above example, value of one macro variable is another macro variable reference.

**Video link: Youtube_Video**

## Q59: What are the differences between call symput and symget in SAS MACRO ?

⇨ **1. CALL SYMPUT is routine(A procedure that can accept arguments but does not return any value) but SYMGET is function.**

**2. CALL SYMPUT is used to assign a value produced in a data step to a macro variable, where SYMGET returns the value of a macro variable to the data step during data step execution.**

**CALL SYMPUT:**

Syntex: Call symput('m_var', m_value);

**Sample code:**

```
option symbolgen;
Data Test;
Month5='MAY';
Call symput('mvar', Month5);
%put macro_var=&mvar.;
Run;
```

**Log info:**

```
69          option symbolgen;
70          Data Test;
71          Month5='MAY';
72          Call symput('mvar', Month5);
73          %put macro_var=&mvar.;
SYMBOLGEN:  Macro variable MVAR resolves to MAY
macro_var=MAY
74          Run;
```

---

**SYMGET:**

Syntex: symget('m_var')= value;

**Sample code:**

```
option symbolgen;
Data Test1;
Month4='APRIL';
```

```
Month5=symget('mvar');
Run;
```

Log info:

```
69          option symbolgen nomprint;
70          Data Test1;
71          Month4='APRIL';
72          Month5=symget('mvar');
73          Run;
```

Output Data:

Columns

| | |
|---|---|
| ☑ | Select all |
| ☑ | Month4 |
| ☑ | Month5 |

Total rows: 1 Total columns: 2

| | Month4 | Month5 |
|---|---|---|
| 1 | APRIL | MAY |

**Video link: Youtube Video**

## Q60: How do you Call SAS MACRO from the external location?

⇨ **Following two methods are used to *Call SAS MACRO from the external location.***

***1. Using %include***

***2. Using AUTOCALL Macro Facility***

**1. Using % Include:**

```
%include '/home/joshid700/Interview/SAS_Macro/Macrostat.sas';
%frequency;
```

**In this method we need to provide the file name of the SAS MACRO(Macrostat.sas is file name). Frequency is macro name.**

**Folder location and file name all are case sensitive.**

**Result:**

The FREQ Procedure

| Status | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
|---|---|---|---|---|
| Alive | 3218 | 61.78 | 3218 | 61.78 |
| Dead | 1991 | 38.22 | 5209 | 100.00 |

---

**2. Using AUTOCALL Macro Facility:**

```
options mautosource sasautos = (sasautos'~/SAS_Macro/');
%stat1(sashelp.cars);
```

**In this method, we use SAS MACRO options mautosource sasautos and provide the folder name with tilde(~) in which the code file is saved.**

**Result:**

The MEANS Procedure

| Variable | Label | N | Mean | Std Dev | Minimum | Maximum |
|---|---|---|---|---|---|---|
| MSRP | | 428 | 32774.86 | 19431.72 | 10280.00 | 192465.00 |
| Invoice | | 428 | 30014.70 | 17642.12 | 9875.00 | 173560.00 |
| EngineSize | Engine Size (L) | 428 | 3.1967290 | 1.1085947 | 1.3000000 | 8.3000000 |
| Cylinders | | 426 | 5.8075117 | 1.5584426 | 3.0000000 | 12.0000000 |
| Horsepower | | 428 | 215.8855140 | 71.8360316 | 73.0000000 | 500.0000000 |
| MPG_City | MPG (City) | 428 | 20.0607477 | 5.2382176 | 10.0000000 | 60.0000000 |
| MPG_Highway | MPG (Highway) | 428 | 26.8434579 | 5.7412007 | 12.0000000 | 66.0000000 |
| Weight | Weight (LBS) | 428 | 3577.95 | 758.9832146 | 1850.00 | 7190.00 |
| Wheelbase | Wheelbase (IN) | 428 | 108.1542056 | 8.3118130 | 89.0000000 | 144.0000000 |
| Length | Length (IN) | 428 | 186.3621495 | 14.3579913 | 143.0000000 | 238.0000000 |

**Video link: Youtube Video**

**Q61: You have given a credit card customer data. Manager is asking the list of customer name in a row with comma separation. How do you solve the task?**

⇨ **We can use SAS MACRO - Select INTO clause in Proc SQL to collect all the name list with as defined separation.**

**Sample Code:**

```
PROC SQL;
SELECT NAME INTO: MACRO_VAR SEPARATED BY ','
FROM SASHELP.CLASS
WHERE AGE GT 13;
QUIT;
%PUT &MACRO_VAR;
```

**Log:**

```
70          PROC SQL;
71          SELECT NAME INTO: MACRO_VAR SEPARATED BY ','
72          FROM SASHELP.CLASS
73          WHERE AGE GT 13;
74          QUIT;
NOTE: PROCEDURE SQL used (Total process time):
      real time               0.00 seconds
      user cpu time           0.01 seconds
      system cpu time         0.00 seconds
      memory                  5493.71k
      OS Memory               26020.00k
      Timestamp               04/06/2024 07:59:49 PM
      Step Count                        68  Switch Count  0
      Page Faults                       0
      Page Reclaims                     74
      Page Swaps                        0
      Voluntary Context Switches        0
      Involuntary Context Switches      0
      Block Input Operations            0
      Block Output Operations           8


75          %PUT &MACRO_VAR;
Alfred,Carol,Henry,Janet,Judy,Mary,Philip,Ronald,William
```

```
DATA NAME_LIST;
NAME="&MACRO_VAR.";
RUN;
```

**Output:**


**Video link: Youtube_Video**

**Q62: How can you export categorical data in to a single excel worksheet with the category in the different tab ?**

⇨ ***To solve this task we can use a data set which contains the categorical data –Example : Lets take Demographics dataset from SASHELP library and we are going to export data according to region into a single worksheet with the different tab.***
***Before going through the SAS program, following steps are useful to be familiar with the data.***

***1. Identify data location(Sometimes we need to use code to extract data) In this example Demographics dataset is in SASHELP library which is permanent library.***
Libname= SASHELP, Data=Demographics
***2. Identify number of variables, variable type.***
Proc contents data =sashelp.demographics varnum;
Run;

Variables in Creation Order

| # | Variable | Type | Len | Format | Label |
|---|---|---|---|---|---|
| 1 | CONT | Num | 8 | | Numeric Rep. for Continent |
| 2 | ID | Num | 8 | | GLC Country ID Number |
| 3 | ISO | Num | 8 | Z3. | ISO Country Number: 900+ Undefined |
| 4 | NAME | Char | 45 | | GLC Country Name |
| 5 | ISONAME | Char | 45 | | ISO Name for Country |
| 6 | region | Char | 6 | | Region |
| 7 | pop | Num | 8 | COMMA15. | Population (2005) |
| 8 | popAGR | Num | 8 | PERCENTN9.2 | Population Annual Growth Rate Percentage (1995-2005) |
| 9 | popUrban | Num | 8 | PERCENTN9.2 | Population in Urban Areas Percentage (2005) |
| 10 | totalFR | Num | 8 | | Total Fertility Rate (per woman 2004) |
| 11 | AdolescentFPpct | Num | 8 | PERCENTN9.2 | Adolescent Fertility Proportion Percentage |
| 12 | AdolescentFPyear | Num | 8 | | Adolescent Fertility Proportion Year |
| 13 | AdultLiteracypct | Num | 8 | PERCENTN9.2 | Adult Literacy Rate Percentage (2000-2004) |
| 14 | MaleSchoolpct | Num | 8 | PERCENTN9.2 | Net Primary School Enrollment Ratio - Male Percentage (1998-2004) |
| 15 | FemaleSchoolpct | Num | 8 | PERCENTN9.2 | Net Primary School Enrollment Ratio - Female Percentage (1998-2004) |
| 16 | GNI | Num | 8 | | Gross National Income per Capita (PPP Int.$ 2004) |
| 17 | PopPovertypct | Num | 8 | PERCENTN9.2 | Population Living Below the Poverty Line (% with <$1 a day) |
| 18 | PopPovertyYear | Num | 8 | | Population Living Below the Poverty Line (Year) |

***3. Identify distinct count of required variable(Region) and corresponding counts.***
Proc freq data =sashelp.demographics;
Tables region;
Run;

| Region | | | | |
|---|---|---|---|---|
| region | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
| AFR | 46 | 23.35 | 46 | 23.35 |
| AMR | 35 | 17.77 | 81 | 41.12 |
| EMR | 21 | 10.66 | 102 | 51.78 |
| EUR | 55 | 27.92 | 157 | 79.70 |
| SEAR | 11 | 5.58 | 168 | 85.28 |
| WPR | 29 | 14.72 | 197 | 100.00 |

4. ***Following macro code can be used to export region wise demographic information to a single excel worksheet with the different tab. This is a SAS program where we use different ways of SAS Macro variable creation procedure and its use.***

```
proc sql;
   select distinct region into :region_list1-
  from sashelp.demographics;
quit;
```

**A: Proc sql; Select into clause macro variable creation.**

It creates list of macro variables region_list1 to region_list(Total). It stores each region into the corresponding macro variable.
[region_list1=AFR, region_list2=AMR, region_list3=EMR, region_list4=EUR, region_list5=SEAR, region_list6=WPR]

```
%let Folderpath= /home/joshid700/Interview/SAS_Macro/;
```

**B: %let method to create macro variable:**

It creates macro variable Folderpath where its value is
/home/joshid700/Interview/SAS_Macro/

```
%MACRO DEMO_Print;
 %do i=1 %to &sqlobs;
       Data test_&&region_list&i.;
             set sashelp.demographics;
   where region="&&region_list&i.";
            proc export data=test_&&region_list&i.
 outfile="&folderpath./Demographics.xlsx"
dbms=xlsx replace;
 sheet="test_&&region_list&i.";
%end;
%MEND;
```

***C: Iterative %do macro variable creation:***

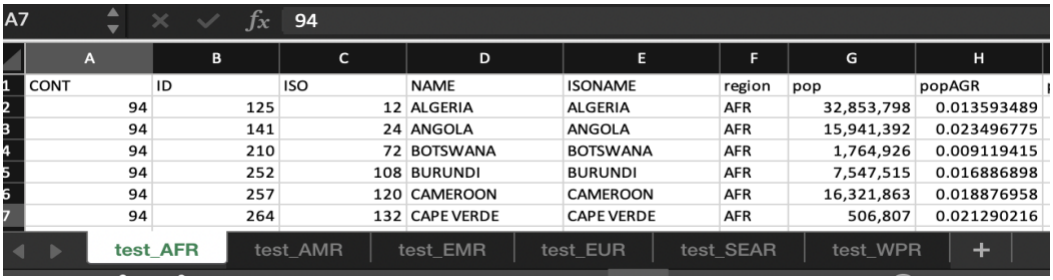
*This step creates macro variable i where value starts from 1 to Total count. It creates region wise six dataset*
*[test_AFR, test_AMR, test_EMR, test_EUR, test_SEAR, test_WPR] and exports to the different tab with corresponding dataset name.*

```
%MEND;
 %DEMO_Print;
```

***D: SAS MACRO:***

*It has SAS MACRO name DEMO_Print.*

**Exported Excel output:**



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | CONT | ID | ISO | NAME | ISONAME | region | pop | popAGR |
| 2 | 94 | 125 | 12 | ALGERIA | ALGERIA | AFR | 32,853,798 | 0.013593489 |
| 3 | 94 | 141 | 24 | ANGOLA | ANGOLA | AFR | 15,941,392 | 0.023496775 |
| 4 | 94 | 210 | 72 | BOTSWANA | BOTSWANA | AFR | 1,764,926 | 0.009119415 |
| 5 | 94 | 252 | 108 | BURUNDI | BURUNDI | AFR | 7,547,515 | 0.016886898 |
| 6 | 94 | 257 | 120 | CAMEROON | CAMEROON | AFR | 16,321,863 | 0.018876958 |
| 7 | 94 | 264 | 132 | CAPE VERDE | CAPE VERDE | AFR | 506,807 | 0.021290216 |



**Video link: Youtube Video**

## Q63: How do you remove duplicate records from a SAS dataset?

⇨ **There are different ways to remove duplicate records from sas dataset.**

**1. proc sort with nodup and nodupkey options.**

**2. first.variable and last.variable**

**Sample Dataset: Employee**

Total rows: 13 Total columns: 4

| | E_ID | NAME | SALARY | DEPARTMENT |
|---|---|---|---|---|
| 1 | 1 | JEFF | 4500 | IT |
| 2 | 1 | ROBERT | 8750 | HR |
| 3 | 2 | SAMI | 8000 | IT |
| 4 | 2 | SAMI | 8000 | IT |
| 5 | 3 | JESS | 8700 | IT |
| 6 | 1 | MARK | 8000 | ELECTRIC |
| 7 | 3 | MARY | 6200 | ELECTRIC |
| 8 | 3 | JESS | 8700 | IT |
| 9 | 4 | BOB | 6500 | HR |
| 10 | 5 | JENNY | 7000 | IT |
| 11 | 7 | JENNY | 7050 | HR |
| 12 | 5 | KEN | 8700 | HR |
| 13 | 6 | DAN | 6500 | IT |

**Proc sort with nodup option:This method sorts the data by all variable and displays output without duplication:Tbl-1**

```
PROC SORT DATA=EMPLOYEE NODUP OUT=NODUP_ALL;
BY _ALL_;
RUN;
```

**Output from NODUP option:Tbl-1**

| | E_ID | NAME | SALARY | DEPARTM... |
|---|---|---|---|---|
| 1 | 1 | JEFF | 4500 | IT |
| 2 | 1 | MARK | 8000 | ELECTRIC |
| 3 | 1 | ROBERT | 8750 | HR |
| 4 | 2 | SAMI | 8000 | IT |
| 5 | 3 | JESS | 8700 | IT |
| 6 | 3 | MARY | 6200 | ELECTRIC |
| 7 | 4 | BOB | 6500 | HR |
| 8 | 5 | JENNY | 7000 | IT |
| 9 | 5 | KEN | 8700 | HR |
| 10 | 6 | DAN | 6500 | IT |
| 11 | 7 | JENNY | 7050 | HR |

**Proc sort with nodupkey option:** This method sorts the data by a certain variable and displays output without duplication by that variable:Tbl-2

```
PROC SORT DATA=EMPLOYEE NODUPKEY OUT=NODUP_EID;
BY E_ID;
RUN;
```

**Output from NODUPKEY option: Tbl-2**

| | E_ID | NAME | SALARY | DEPARTMENT |
|---|---|---|---|---|
| 1 | 1 | JEFF | 4500 | IT |
| 2 | 2 | SAMI | 8000 | IT |
| 3 | 3 | JESS | 8700 | IT |
| 4 | 4 | BOB | 6500 | HR |
| 5 | 5 | JENNY | 7000 | IT |
| 6 | 6 | DAN | 6500 | IT |
| 7 | 7 | JENNY | 7050 | HR |

**This is data step method where first.variable and last.variable is used. If we need to keep distinct records and records from a duplicate variable then before using first.variable or last.variable, we need to sort the data by the given variable.**

**Keeping distinct records and first records from each duplicate variable group(if exist):**

```
PROC SORT DATA=EMPLOYEE OUT=EMP_SORT;
BY E_ID;
RUN;
DATA FIRST_REC;
SET EMP_SORT;
BY E_ID;
IF FIRST.E_ID THEN OUTPUT ;
RUN;
```

**Output:**

Total rows: 7 Total columns: 4

| | E_ID | NAME | SALARY | DEPARTMENT |
|---|---|---|---|---|
| 1 | 1 | JEFF | 4500 | IT |
| 2 | 2 | SAMI | 8000 | IT |
| 3 | 3 | JESS | 8700 | IT |
| 4 | 4 | BOB | 6500 | HR |
| 5 | 5 | JENNY | 7000 | IT |
| 6 | 6 | DAN | 6500 | IT |
| 7 | 7 | JENNY | 7050 | HR |

**Keeping distinct records and last records from each duplicate variable group(if exist):**

```
DATA LAST_REC;
SET EMP_SORT;
BY E_ID;
IF LAST.E_ID THEN OUTPUT;
RUN;
```

**Output:**

Total rows: 7 Total columns: 4

| | E_ID | NAME | SALARY | DEPARTMENT |
|---|---|---|---|---|
| 1 | 1 | MARK | 8000 | ELECTRIC |
| 2 | 2 | SAMI | 8000 | IT |
| 3 | 3 | JESS | 8700 | IT |
| 4 | 4 | BOB | 6500 | HR |
| 5 | 5 | KEN | 8700 | HR |
| 6 | 6 | DAN | 6500 | IT |
| 7 | 7 | JENNY | 7050 | HR |

**Video link: Youtube Video**

## Q64: How do you extract records having max and min salary from each department?

⇨ **There are different ways to solve this task(Using data step and procedure step). Using Employee data set from QN:63. Following Data step code can be used to extract records corresponding max and min salary from each department.**

**/*Sorting data by department and salary by descending*/**

```
PROC SORT DATA=EMPLOYEE OUT=EMP_SORT;
BY DEPARTMENT DESCENDING SALARY;
RUN;
```

**/*Max Salary from each Department*/**

```
DATA MAX_SALARY_LIST;
SET EMP_SORT;
BY DEPARTMENT;
```

```
IF FIRST.DEPARTMENT THEN OUTPUT;
RUN;
```

**MAX_SALARY_LIST**

Total rows: 3 Total columns: 4

| | E_ID | NAME | SALARY | DEPARTMENT |
|---|---|---|---|---|
| 1 | 1 | MARK | 8000 | ELECTRIC |
| 2 | 1 | ROBERT | 8750 | HR |
| 3 | 3 | JESS | 8700 | IT |

**/*Min Salary from each Department*/**

```
DATA MIN_SALARY_LIST;
SET EMP_SORT;
BY DEPARTMENT;
IF LAST.DEPARTMENT THEN OUTPUT ;
RUN;
```

**MIN_SALARY_LIST:**

Total rows: 3 Total columns: 4

| | E_ID | NAME | SALARY | DEPARTMENT |
|---|---|---|---|---|
| 1 | 3 | MARY | 6200 | ELECTRIC |
| 2 | 4 | BOB | 6500 | HR |
| 3 | 1 | JEFF | 4500 | IT |

**Video link: Youtube_Video**

**Q65: How can you create report having groupwise customers name separated by comma.**

⇨ **Target output can be obtained by following two step process:**
**1. Sorting the source data by grouping variable.**
**2.Using do loop until last.grouping variable with catx function. Syntax for catx:**
**CATX(delimiter, Item1 <,…Itemn>)**
***Sorting source data by grouping variable*:**

```
PROC SORT DATA=SASHELP.CLASS OUT=CLASSDATA;
BY AGE;
RUN;
```

**Data source(Partial Data):**

Total rows: 19 Total columns: 5

| | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Joyce | F | 11 | 51.3 | 50.5 |
| 2 | Thomas | M | 11 | 57.5 | 85 |
| 3 | James | M | 12 | 57.3 | 83 |
| 4 | Jane | F | 12 | 59.8 | 84.5 |
| 5 | John | M | 12 | 59 | 99.5 |
| 6 | Louise | F | 12 | 56.3 | 77 |
| 7 | Robert | M | 12 | 64.8 | 128 |
| 8 | Alice | F | 13 | 56.5 | 84 |
| 9 | Barbara | F | 13 | 65.3 | 98 |

***New data keeping Age & NameList:***

```
DATA TARGET_OUTPUT(KEEP=AGE NAMELIST);
 DO UNTIL (LAST.AGE);
  SET CLASSDATA;
  BY AGE;
  LENGTH NAMELIST $100;
  NAMELIST = CATX(', ',NAMELIST, NAME);
 END;
RUN;
```

**Target_Output:**

Total rows: 6 Total columns: 2

| | Age | NameList |
|---|---|---|
| 1 | 11 | Joyce, Thomas |
| 2 | 12 | James, Jane, John, Louise, Rob |
| 3 | 13 | Alice, Barbara, Jeffrey |
| 4 | 14 | Alfred, Carol, Henry, Judy |
| 5 | 15 | Janet, Mary, Ronald, William |
| 6 | 16 | Philip |

**Video link: Youtube_Video**

## Q66: How can you save distinct and duplicated records by a variable into two different datasets?

⇨ **By using data step method (first.variable and last.variable), *we can save distinct and duplicated records by a variable into two different datasets?***

**/*Sorting employee data by E_ID to output SORT_OUT*/**

```
PROC SORT DATA =EMPLOYEE OUT=SORT_OUT;
BY E_ID;
RUN;
```

**/*Saving records having distinct and duplicate E_ID into two different data sets*/**

```
DATA DISTINCT_ID DUP_ID;
```

```
SET SORT_OUT;
BY E_ID;
IF FIRST.E_ID and LAST.E_ID THEN OUTPUT DISTINCT_ID;
ELSE OUTPUT DUP_ID;
RUN;
```

**OUTPUT :DISTINCT_ID**

Total rows: 3 Total columns: 4

| | E_ID | NAME | SALARY | DEPARTMENT |
|---|---|---|---|---|
| 1 | 4 | BOB | 6500 | HR |
| 2 | 6 | DAN | 6500 | IT |
| 3 | 7 | JENNY | 7050 | HR |

**OUTPUT :DUP_ID**

Total rows: 10 Total columns: 4

| | E_ID | NAME | SALARY | DEPARTMENT |
|---|---|---|---|---|
| 1 | 1 | JEFF | 4500 | IT |
| 2 | 1 | ROBERT | 8750 | HR |
| 3 | 1 | MARK | 8000 | ELECTRIC |
| 4 | 2 | SAMI | 8000 | IT |
| 5 | 2 | SAMI | 8000 | IT |
| 6 | 3 | JESS | 8700 | IT |
| 7 | 3 | MARY | 6200 | ELECTRIC |
| 8 | 3 | JESS | 8700 | IT |
| 9 | 5 | JENNY | 7000 | IT |
| 10 | 5 | KEN | 8700 | HR |

**Video link: Youtube_Video**

## Q67: What is SAS Array and why do we use it?

⇨ ***The repeated structure in the SAS data step is called an array which is similar to a dimension, matrix or table in mathematics.***
***Array is used to simplify the SAS processing and it helps to read and analyze repetitive data with a minimum of coding.***

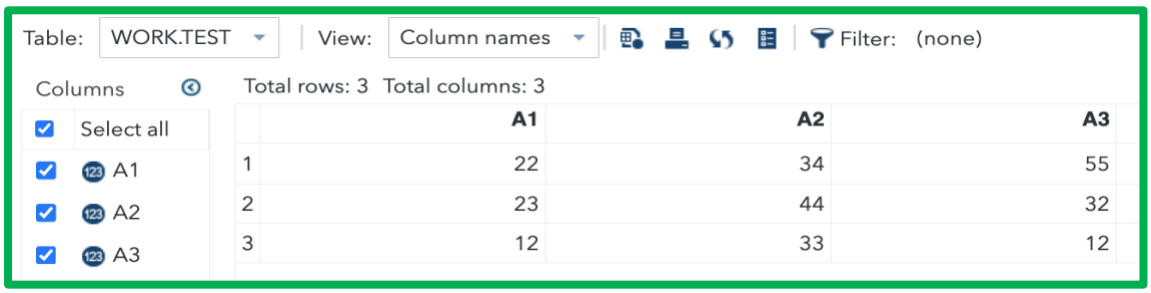
**Further Information:**

1. **When all the numeric variables are used as element then _NUMERIC_ can be used as array-element.**
2. **When all the character variables are used as element then _CHARACTER_ can be used as array-element.**
3. **When all the variables used are same type then _ALL_ Can be used as array-element.**

**Array Statement:**
**Array Array_Name{n} array-element.**
If you are creating array for character variable then:
**Array Array_Name{n} $ <length> array-element**.

```
DATA TEST
INPUT A1 A2 A3;
DATALINES;
22 34 55
23 44 32
12 33 12
;
RUN;
```

**Output:**

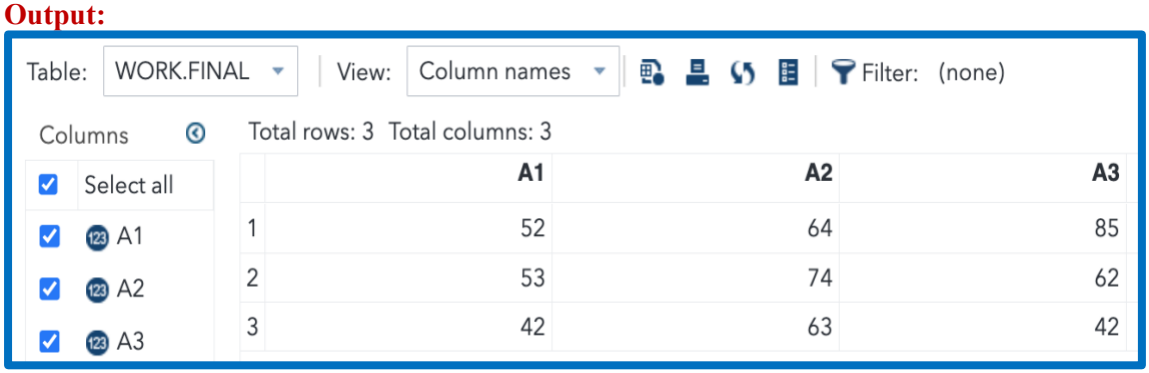



| | A1 | A2 | A3 |
|---|---|---|---|
| 1 | 22 | 34 | 55 |
| 2 | 23 | 44 | 32 |
| 3 | 12 | 33 | 12 |

**/* We are adding 30 with each record of all fields*/**

```
DATA FINAL;
SET TEST;
ARRAY NEW_VALUE(3) A1 A2 A3;
DO I=1 TO 3;
NEW_VALUE(I)=NEW_VALUE(I) +30;
END;
DROP I;
RUN;
```

**Output:**




| | A1 | A2 | A3 |
|---|---|---|---|
| 1 | 52 | 64 | 85 |
| 2 | 53 | 74 | 62 |
| 3 | 42 | 63 | 42 |

**Video link: Youtube Video**

## Q68: What is the best approach to replace all missing character variable value by the word 'Miss' in SAS data step?

⇨ **SAS array can be used to replace all missing character variable value by 'Miss' where we use _character_ in place of array element and do loop is used to apply change to all missing character variable. Similarly, to replace all missing numeric variable value by 9999, we use _numeric_ in place of array element.**

**Sample Data:**

| Status | DeathCause | AgeCHDdiag | Sex | AgeAtStart |
|---|---|---|---|---|
| Alive | | . | Male | 31 |
| Dead | Other | 64 | Male | 42 |
| Dead | Coronary Heart Disease | 75 | Male | 55 |
| Alive | | . | Male | 37 |
| Dead | Coronary Heart Disease | 62 | Female | 58 |
| Alive | | 43 | Male | 35 |
| Alive | | 67 | Male | 45 |
| Alive | | 63 | Female | 37 |
| Alive | | . | Male | 40 |
| Dead | Coronary Heart Disease | 55 | Male | 47 |

**/*Replacing all missing character value by 'Miss'*/**

```
DATA HEART_FINAL1;
SET SASHELP.HEART;
ARRAY CHARACTER(*) _CHARACTER_;
DO I=1 TO DIM(CHARACTER);
IF CHARACTER(I) =' ' THEN CHARACTER(I)='MISS ';
WHERE WEIGHT GT 195;
END;
DROP I;
RUN;
```

**Output:**

Total rows: 395 Total columns: 17

| | Status | DeathCause | AgeCHDdiag | Sex | AgeAtStart |
|---|---|---|---|---|---|
| 1 | Alive | MISS | . | Male | 31 |
| 2 | Dead | Other | 64 | Male | 42 |
| 3 | Dead | Coronary Heart Disease | 75 | Male | 55 |
| 4 | Alive | MISS | . | Male | 37 |
| 5 | Dead | Coronary Heart Disease | 62 | Female | 58 |
| 6 | Alive | MISS | 43 | Male | 35 |

**/*Replacing all missing numeric value by 9999*/**

```
DATA HEART_FINAL2;
SET SASHELP.HEART;
ARRAY NUMERIC(*) _NUMERIC_;
DO I=1 TO DIM(NUMERIC);
IF NUMERIC(I) =.  THEN NUMERIC(I)= 9999;
WHERE WEIGHT GT 195;
END;
DROP I;
RUN
```

**Output:**

| | Status | DeathCause | AgeCHDdiag | Sex | AgeAtStart |
|---|---|---|---|---|---|
| 1 | Alive | | 9999 | Male | 31 |
| 2 | Dead | Other | 64 | Male | 42 |
| 3 | Dead | Coronary Heart Disease | 75 | Male | 55 |
| 4 | Alive | | 9999 | Male | 37 |
| 5 | Dead | Coronary Heart Disease | 62 | Female | 58 |
| 6 | Alive | | 43 | Male | 35 |

**Video link: Youtube Video**

## Q69: What are the differences between Join and Merge in SAS? Which one is more efficient?

⇨ ***There are following notable points between proc sql join and data step merge:***
***1. Both methods are used to combine the two or more than two data sets. Join doesn't need to sort the data, but merge need to sort the data by index or by variable.***
***2. Multiple data sets can be joined in one step without having common variables in all the data sets.***
***3. Proc SQL join can handle many to many relationships whereas Data step merge do not.***

***Which one is more efficient?***
***Answer is neither because these are two different facilities and cannot be easily compared. One is not better than other because two facilities perform significantly different functions and both are equally valuable.***

**Sample code:**
*(SQL IN SAS-PROC SQL JOIN)*

```
PROC SQL;
CREATE TABLE COMBINED1 AS
SELECT X.*, Y.* FROM
TBL1 X
INNER JOIN
TBL2 Y
ON X.ID=Y.ID
WHERE X.SALARY GT 5000;
QUIT;
```

**Sample code:**
*(DATA STEP MERGE)*
*PROC SORT DATA =TBL1 OUT=TBL3; BY ID; RUN;*
*PROC SORT DATA =TBL2 OUT=TBL4; BY ID; RUN;*

```
DATA COMBINED1;
MERGE TBL3(IN=A) TBL4(IN=B);
BY ID;
IF A AND B THEN OUTPUT;
WHERE SALARY GT 5000;
RUN;
```

**Video link: Youtube Video**

## Q70: How do you use data step merge to get same result as inner join (from proc sql)?

⇨ **Common information which exist in both tables can be obtained by using following data step merge which results exactly same result as inner join with proc sql.**

**Employee Table:**

Total rows: 6 Total columns: 5 Rows 1-6

| | ID | NAME | SALARY | DEPT_ID | MANAGER_ID |
|---|---|---|---|---|---|
| 1 | 101 | KEVIN | 8900 | 20 | 103 |
| 2 | 102 | ALEX | 10000 | 23 | 103 |
| 3 | 103 | KELLY | 9800 | 20 | 104 |
| 4 | 104 | NEENA | 9000 | 25 | 109 |
| 5 | 105 | JESS | 9000 | 33 | 109 |
| 6 | 109 | VANCE | 10500 | 20 | . |

**Department Table:**

| | DEPT_ID | DEPT_NAME |
|---|---|---|
| 1 | 16 | FINANCE |
| 2 | 20 | IT |
| 3 | 23 | MECHANICAL |
| 4 | 25 | ELECTRICAL |
| 5 | 30 | HR |

**Using above two dataset: Employee and Department as two source tables.** ***Proc SQL-Inner Join:***

```
PROC SQL;
CREATE TABLE INNER_JOIN AS
   SELECT E.*, D.DEPT_NAME
   FROM EMPLOYEE E
   INNER JOIN DEPARTMENT D
   ON E.DEPT_ID=D.DEPT_ID
ORDER BY E.DEPT_ID;
QUIT;
```

**Output Result: From proc sql-Join**

| ID | NAME | SALARY | DEPT_ID | MANAGER_ID | DEPT_NAME |
|---|---|---|---|---|---|
| 103 | KELLY | 9800 | 20 | 104 | IT |
| 109 | VANCE | 10500 | 20 | . | IT |
| 101 | KEVIN | 8900 | 20 | 103 | IT |
| 102 | ALEX | 10000 | 23 | 103 | MECHANICAL |
| 104 | NEENA | 9000 | 25 | 109 | ELECTRICAL |

**Using lefthand side two dataset: Employee and Department as two source tables.** ***Data step-Merge***

```
PROC SORT DATA =EMPLOYEE; BY DEPT_ID; RUN;
PROC SORT DATA =DEPARTMENT; BY DEPT_ID; RUN;
DATA FINAL_DATA;
MERGE EMPLOYEE(IN=A) DEPARTMENT(IN=B);
BY DEPT_ID;
IF A and B THEN OUTPUT;
RUN;
```

**Output Result: From Data step Merge**

| ID | NAME | SALARY | DEPT_ID | MANAGER_ID | DEPT_NAME |
|---|---|---|---|---|---|
| 101 | KEVIN | 8900 | 20 | 103 | IT |
| 103 | KELLY | 9800 | 20 | 104 | IT |
| 109 | VANCE | 10500 | 20 | . | IT |
| 102 | ALEX | 10000 | 23 | 103 | MECHANICAL |
| 104 | NEENA | 9000 | 25 | 109 | ELECTRICAL |

**Video link: Youtube Video**

## Q71: How do you use data step merge to get same result as left join (from proc sql)?

⇨ **Following code is used *to get same result from data step merge and proc sql -left join.***
**Referring Question # 70: Two dataset Employee and Department are used as two source tables.**

***Data tep: Merge***

```
PRC SORT DATA =EMPLOYEE; BY DEPT_ID; RUN;
PROC SORT DATA =DEPARTMENT; BY DEPT_ID; RUN;
DATA FINAL_DATA;
MERGE EMPLOYEE(IN=A) DEPARTMENT(IN=B);
BY DEPT_ID;
IF A THEN OUTPUT;
RUN;
```

**Output from Data step merge:**

| ID | NAME | SALARY | DEPT_ID | MANAGER_ID | DEPT_NAME |
|---|---|---|---|---|---|
| 101 | KEVIN | 8900 | 20 | 103 | IT |
| 103 | KELLY | 9800 | 20 | 104 | IT |
| 109 | VANCE | 10500 | 20 | . | IT |
| 102 | ALEX | 10000 | 23 | 103 | MECHANICAL |
| 104 | NEENA | 9000 | 25 | 109 | ELECTRICAL |
| 105 | JESS | 9000 | 33 | 109 | |

***Proc Sql: Join***

```
PROC SQL;
CREATE TABLE LEFT_JOIN AS
   SELECT E.*, D.DEPT_NAME
   FROM EMPLOYEE E
   LEFT JOIN DEPARTMENT D
   ON E.DEPT_ID=D.DEPT_ID
ORDER BY E.DEPT_ID;
QUIT;
```

***Output from left join (Proc SQL)***

| ID | NAME | SALARY | DEPT_ID | MANAGER_ID | DEPT_NAME |
|---|---|---|---|---|---|
| 101 | KEVIN | 8900 | 20 | 103 | IT |
| 103 | KELLY | 9800 | 20 | 104 | IT |
| 109 | VANCE | 10500 | 20 | . | IT |
| 102 | ALEX | 10000 | 23 | 103 | MECHANICAL |
| 104 | NEENA | 9000 | 25 | 109 | ELECTRICAL |
| 105 | JESS | 9000 | 33 | 109 | |

**Video link: Youtube Video**

## Q72: How do you use data step merge to get same result as right join (from proc sql)?

⇨ **Following code is used *to get same result from data step merge and proc sql -right join.***
**Referring Question # 70: Two dataset Employee and Department are used as two source tables.**
**Data step: Merge**

```
PROC SORT DATA =EMPLOYEE; BY DEPT_ID; RUN;
PROC SORT DATA =DEPARTMENT; BY DEPT_ID; RUN;
DATA FINAL_DATA;
MERGE EMPLOYEE(IN=A) DEPARTMENT(IN=B);
BY DEPT_ID;
IF B THEN OUTPUT;
RUN;
```

**Output:**

| ID | NAME | SALARY | DEPT_ID | MANAGER_ID | DEPT_NAME |
|---|---|---|---|---|---|
| . | | . | 16 | . | FINANCE |
| 101 | KEVIN | 8900 | 20 | 103 | IT |
| 103 | KELLY | 9800 | 20 | 104 | IT |
| 109 | VANCE | 10500 | 20 | . | IT |
| 102 | ALEX | 10000 | 23 | 103 | MECHANICAL |
| 104 | NEENA | 9000 | 25 | 109 | ELECTRICAL |
| . | | . | 30 | . | HR |

***Proc sql-Right join:***

```
PROC SQL;
CREATE TABLE RIGHT_JOIN AS
 SELECT E.ID, E.NAME,E.SALARY,COALESCE(E.DEPT_ID,D.DEPT_ID) AS  DEPT_ID,
E.MANAGER_ID, D.DEPT_NAME
   FROM EMPLOYEE E
   RIGHT JOIN DEPARTMENT D
   ON E.DEPT_ID=D.DEPT_ID
ORDER BY DEPT_ID;
QUIT;
```

**Output:**

| ID | NAME | SALARY | DEPT_ID | MANAGER_ID | DEPT_NAME |
|---|---|---|---|---|---|
| . | | . | 16 | . | FINANCE |
| 101 | KEVIN | 8900 | 20 | 103 | IT |
| 103 | KELLY | 9800 | 20 | 104 | IT |
| 109 | VANCE | 10500 | 20 | . | IT |
| 102 | ALEX | 10000 | 23 | 103 | MECHANICAL |
| 104 | NEENA | 9000 | 25 | 109 | ELECTRICAL |
| . | | . | 30 | . | HR |

**Video link: Youtube Video**

## Q73: How do you use data step merge to get same result as full join (from proc sql)?

⇨ **Following code is used *to get same result from data step merge and proc sql –FULL JOIN.* Referring Question # 70: Two dataset Employee and Department are used as two source tables.**

*DATA STEP MERGE:*

```
PROC SORT DATA =EMPLOYEE; BY DEPT_ID; RUN;
PROC SORT DATA =DEPARTMENT; BY DEPT_ID; RUN;
DATA FINAL_DATA;
MERGE EMPLOYEE(IN=A) DEPARTMENT(IN=B);
BY DEPT_ID;
IF A OR B THEN OUTPUT;
RUN;
```

**Output:**

| ID | NAME | SALARY | DEPT_ID | MANAGER_ID | DEPT_NAME |
|---|---|---|---|---|---|
| . | | . | 16 | . | FINANCE |
| 101 | KEVIN | 8900 | 20 | 103 | IT |
| 103 | KELLY | 9800 | 20 | 104 | IT |
| 109 | VANCE | 10500 | 20 | . | IT |
| 102 | ALEX | 10000 | 23 | 103 | MECHANICAL |
| 104 | NEENA | 9000 | 25 | 109 | ELECTRICAL |
| . | | . | 30 | . | HR |
| 105 | JESS | 9000 | 33 | 109 | |

*PROC SQL-FULL JOIN:*

```
PROC SQL;
CREATE TABLE FULL_JOIN AS
   SELECT E.ID,E.NAME,E.SALARY,COALESCE(E.DEPT_ID,
   D.DEPT_ID) AS DEPT_ID, E.MANAGER_ID,D.DEPT_NAME
   FROM EMPLOYEE E
   Full JOIN DEPARTMENT D
   ON E.DEPT_ID=D.DEPT_ID
ORDER BY DEPT_ID;
QUIT;
```

**Output:**

| ID | NAME | SALARY | DEPT_ID | MANAGER_ID | DEPT_NAME |
|---|---|---|---|---|---|
| . | | . | 16 | . | FINANCE |
| 101 | KEVIN | 8900 | 20 | 103 | IT |
| 103 | KELLY | 9800 | 20 | 104 | IT |
| 109 | VANCE | 10500 | 20 | . | IT |
| 102 | ALEX | 10000 | 23 | 103 | MECHANICAL |
| 104 | NEENA | 9000 | 25 | 109 | ELECTRICAL |
| . | | . | 30 | . | HR |
| 105 | JESS | 9000 | 33 | 109 | |

**Video link: Youtube Video**

## 74: How does union operator work with proc sql?

⇨ ***Union operator is used to join two or more tables vertically by column position not by column name, so common column in each select statement must be in the same order. It joins table vertically removing duplicate records.***

**DATA: ONE**

Total rows: 4 Total columns: 3

| | ID | SALARY | DEPARTMENT |
|---|---|---|---|
| 1 | 102 | 6754 | IT |
| 2 | 103 | 8890 | HR |
| 3 | 109 | 6543 | ELECTRIC |
| 4 | 205 | 8977 | CS |

**DATA: TWO**

Total rows: 5 Total columns: 3

| | ID | AMOUNT | DEPARTMENT |
|---|---|---|---|
| 1 | 102 | 6754 | IT |
| 2 | 103 | 8890 | HR |
| 3 | 115 | 9000 | CS |
| 4 | 209 | 8977 | CS |
| 5 | 304 | 9950 | HR |

```
PROC SQL;
CREATE TABLE OUT_TBL AS
SELECT *
FROM ONE
UNION
SELECT *
FROM TWO;
QUIT;
```

**DATA: OUT_UNION_TBL**

Total rows: 7 Total columns: 3

| | ID | SALARY | DEPARTMENT |
|---|---|---|---|
| 1 | 102 | 6754 | IT |
| 2 | 103 | 8890 | HR |
| 3 | 109 | 6543 | ELECTRIC |
| 4 | 115 | 9000 | CS |
| 5 | 205 | 8977 | CS |
| 6 | 209 | 8977 | CS |
| 7 | 304 | 9950 | HR |

**Video link: Youtube_Video**

## 75: How does union all operator work with proc sql?

⇨ **Union all operator is used to join two or more tables vertically by column position not by column name, so common column in each select statement must be in the same order. It joins tables vertically keeping all duplicate records.**

**DATA: ONE**

Total rows: 4 Total columns: 3

| | ID | SALARY | DEPARTMENT |
|---|---|---|---|
| 1 | 102 | 6754 | IT |
| 2 | 103 | 8890 | HR |
| 3 | 109 | 6543 | ELECTRIC |
| 4 | 205 | 8977 | CS |

**DATA: TWO:**

Total rows: 5 Total columns: 3

| | ID | AMOUNT | DEPARTMENT |
|---|---|---|---|
| 1 | 102 | 6754 | IT |
| 2 | 103 | 8890 | HR |
| 3 | 115 | 9000 | CS |
| 4 | 209 | 8977 | CS |
| 5 | 304 | 9950 | HR |

```
PROC SQL;
CREATE TABLE OUT_TBL_All AS
SELECT * FROM ONE
UNION ALL
SELECT * FROM TWO
ORDER  BY ID;
QUIT;
```

**DATA: OUT_TBL_All**

Total rows: 9 Total columns: 3

| | ID | SALARY | DEPARTM... |
|---|---|---|---|
| 1 | 102 | 6754 | IT |
| 2 | 102 | 6754 | IT |
| 3 | 103 | 8890 | HR |
| 4 | 103 | 8890 | HR |
| 5 | 109 | 6543 | ELECTRIC |
| 6 | 115 | 9000 | CS |
| 7 | 205 | 8977 | CS |
| 8 | 209 | 8977 | CS |
| 9 | 304 | 9950 | HR |

**Video link: Youtube_Video**

## Q76: How does except operator work with proc sql?

⇨ **Except operator is used to get records from the one table which do not exist in another table. In this example we are going to pull all the records exist in dataset one which do not exist in dataset two.**

**DATA: ONE**

Total rows: 4 Total columns: 3

| | ID | SALARY | DEPARTMENT |
|---|---|---|---|
| 1 | 102 | 6754 | IT |
| 2 | 103 | 8890 | HR |
| 3 | 109 | 6543 | ELECTRIC |
| 4 | 205 | 8977 | CS |

**DATA: TWO:**

Total rows: 5 Total columns: 3

| | ID | AMOUNT | DEPARTMENT |
|---|---|---|---|
| 1 | 102 | 6754 | IT |
| 2 | 103 | 8890 | HR |
| 3 | 115 | 9000 | CS |
| 4 | 209 | 8977 | CS |
| 5 | 304 | 9950 | HR |

```
PROC SQL;
CREATE TABLE OUT_EXCEPT_TBL AS
SELECT * FROM ONE
EXCEPT
SELECT * FROM TWO;
QUIT;
```

**DATA: OUT_TBL**

Total rows: 2 Total columns: 3

| | ID | SALARY | DEPARTM... |
|---|---|---|---|
| 1 | 109 | 6543 | ELECTRIC |
| 2 | 205 | 8977 | CS |

**Video link: Youtube_Video**

## Q77: How do you reshape data structure from vertical to horizontal & vice versa?

⇨ **For reshaping purpose, proc transpose procedure is used with different statement(ID,BY,VAR..)as required. A simple proc transpose scenario is explained as below where one record(One row) per id has converted to multiple records per ID. Before using proc transpose, data set must be sorted by a variable which does not transpose.**

```
DATA ONE_REC_PER_ID;
INPUT STD_ID $ SUBJ1  SUBJ2  SUBJ3;
DATALINES;
001 90 88 56
```

```
201 80 78 85
039 87 45 76
130 57 84 57
200 67 45 89
;
RUN;

PROC SORT DATA =ONE_REC_PER_ID;
BY STD_ID;
RUN;

PROC TRANSPOSE DATA=ONE_REC_PER_ID OUT=MANY_REC_PER_ID_ONE;
BY STD_ID;
RUN;
```

**Output: Many_REC_PER_ID_ONE**

Total rows: 15 Total columns: 3

| | STD_ID | _NAME_ | COL1 |
|---|---|---|---|
| 1 | 001 | SUBJ2 | 88 |
| 2 | 001 | SUBJ3 | 56 |
| 3 | 001 | SUBJ1 | 90 |
| 4 | 039 | SUBJ3 | 76 |
| 5 | 039 | SUBJ2 | 45 |
| 6 | 039 | SUBJ1 | 87 |
| 7 | 130 | SUBJ1 | 57 |
| 8 | 130 | SUBJ3 | 57 |
| 9 | 130 | SUBJ2 | 84 |
| 10 | 200 | SUBJ3 | 89 |
| 11 | 200 | SUBJ2 | 45 |
| 12 | 200 | SUBJ1 | 67 |
| 13 | 201 | SUBJ3 | 85 |
| 14 | 201 | SUBJ2 | 78 |
| 15 | 201 | SUBJ1 | 80 |

**Renaming column name meaningful (Changing COL1 TO MARKS & _NAME_ TO SUBJECT).**

```
DATA ONE_REC_PER_ID;
INPUT STD_ID $ SUBJ1  SUBJ2  SUBJ3;
DATALINES;
001 90 88 56
201 80 78 85
039 87 45 76
130 57 84 57
200 67 45 89
;
RUN;

PROC SORT DATA =ONE_REC_PER_ID;
BY STD_ID;
RUN;
```

```
PROC TRANSPOSE DATA=ONE_REC_PER_ID
OUT=MANY_REC_PER_ID_TWO(RENAME=(COL1=MARKS
_NAME_=SUBJECT));
BY STD_ID;
RUN;
```

**Output: MANY_REC_PER_ID_TWO**

Total rows: 15 Total columns: 3

| | STD_ID | SUBJECT | MARKS |
|---|---|---|---|
| 1 | 001 | SUBJ1 | 90 |
| 2 | 001 | SUBJ2 | 88 |
| 3 | 001 | SUBJ3 | 56 |
| 4 | 039 | SUBJ1 | 87 |
| 5 | 039 | SUBJ2 | 45 |
| 6 | 039 | SUBJ3 | 76 |
| 7 | 130 | SUBJ1 | 57 |
| 8 | 130 | SUBJ2 | 84 |
| 9 | 130 | SUBJ3 | 57 |
| 10 | 200 | SUBJ1 | 67 |
| 11 | 200 | SUBJ2 | 45 |
| 12 | 200 | SUBJ3 | 89 |
| 13 | 201 | SUBJ1 | 80 |
| 14 | 201 | SUBJ2 | 78 |
| 15 | 201 | SUBJ3 | 85 |

**By using proc transpose we reshape data structure. A simple proc transpose scenario explained as below where MANY_REC_PER_ID has converted to ONE_REC_PER_ID.(Vertical data structure has converted to horizontal data structure).**

Total rows: 15 Total columns: 3

| | STD_ID | SUBJECT | MARKS |
|---|---|---|---|
| 1 | 001 | SUBJ1 | 90 |
| 2 | 001 | SUBJ2 | 88 |
| 3 | 001 | SUBJ3 | 56 |
| 4 | 039 | SUBJ1 | 87 |
| 5 | 039 | SUBJ2 | 45 |
| 6 | 039 | SUBJ3 | 76 |
| 7 | 130 | SUBJ1 | 57 |
| 8 | 130 | SUBJ2 | 84 |
| 9 | 130 | SUBJ3 | 57 |
| 10 | 200 | SUBJ1 | 67 |
| 11 | 200 | SUBJ2 | 45 |
| 12 | 200 | SUBJ3 | 89 |
| 13 | 201 | SUBJ1 | 80 |
| 14 | 201 | SUBJ2 | 78 |
| 15 | 201 | SUBJ3 | 85 |

```
PROC TRANSPOSE DATA=MANY_REC_PER_ID_TWO PREFIX=SUBJ
OUT=ONE_REC_PER_ID_OUT(DROP=_NAME_);
BY STD_ID;
RUN;
```

**Output: ONE_REC_PER_ID_OUT**

Total rows: 5 Total columns: 4

| | Std... | SUBJ1 | SUBJ2 | SUBJ3 |
|---|---|---|---|---|
| 1 | 001 | 90 | 88 | 56 |
| 2 | 039 | 87 | 45 | 76 |
| 3 | 130 | 57 | 84 | 57 |
| 4 | 200 | 67 | 45 | 89 |
| 5 | 201 | 80 | 78 | 85 |

## Video link: Youtube_Video

### Q78: How does ID Statement work while transposing data?

⇨ **ID Statement is used to move column values to column name. It can't be used for variable having duplicate value. PREFIX option helps to clarify transposed column name it will be added Infront of the value.**

```
DATA ONE_REC_PER_ID;
INPUT STD_ID $ Physics Chemistry Biology Mathematics;
DATALINES;
001 90 88 56 98
201 80 78 85 89
039 87 45 76 85
130 57 84 57 92
200 67 45 89 80
;
```

```
RUN;
PROC TRANSPOSE DATA=ONE_REC_PER_ID NAME=SUBJECT
OUT=Transposed_Data1;
ID STD_ID;
RUN;
PROC TRANSPOSE DATA=ONE_REC_PER_ID NAME=SUBJECT
OUT=Transposed_Data2 Prefix=SID_;
ID STD_ID;
RUN;
```

**Output: Without prefix option**

Total rows: 4 Total columns: 6

| | SUBJECT | 001 | 201 | 039 | 130 | 200 |
|---|---|---|---|---|---|---|
| 1 | Physics | 90 | 80 | 87 | 57 | 67 |
| 2 | Chemistry | 88 | 78 | 45 | 84 | 45 |
| 3 | Biology | 56 | 85 | 76 | 57 | 89 |
| 4 | Mathematics | 98 | 89 | 85 | 92 | 80 |

**Output: With prefix='SID_' option**

Total rows: 4 Total columns: 6

| | SUBJECT | SID_001 | SID_201 | SID_039 | SID_130 | SID_200 |
|---|---|---|---|---|---|---|
| 1 | Physics | 90 | 80 | 87 | 57 | 67 |
| 2 | Chemistry | 88 | 78 | 45 | 84 | 45 |
| 3 | Biology | 56 | 85 | 76 | 57 | 89 |
| 4 | Mathematics | 98 | 89 | 85 | 92 | 80 |

**Video link: Youtube_Video**

## Q79: How do you transpose given specific variable from column to row?

⇨ **VAR Statement is used to select which variable/s need to transpose column to row in the proc transpose procedure. It helps to keep only required variable to the transposed output. In this example, suppose we need to keep only Physics & Mathematics marks.**

```
DATA ONE_REC_PER_ID;
INPUT STD_ID $ Physics Chemistry Biology Mathematics;
DATALINES;
001 90 88 56 98
201 80 78 85 89
039 87 45 76 85
130 57 84 57 92
200 67 45 89 80
;
RUN;

PROC TRANSPOSE DATA=ONE_REC_PER_ID NAME=SUBJECT
```

```
OUT=Transposed_Data2 Prefix=SID_;
ID STD_ID;
RUN;
PROC TRANSPOSE DATA=ONE_REC_PER_ID NAME=SUBJECT
OUT=Transposed_Data3 Prefix=SID_;
ID STD_ID;
VAR PHYSICS MATHEMATICS;
RUN;
```

**Output: Without VAR Statement:**

Total rows: 4 Total columns: 6

| | SUBJECT | SID_001 | SID_201 | SID_039 | SID_130 | SID_200 |
|---|---|---|---|---|---|---|
| 1 | Physics | 90 | 80 | 87 | 57 | 67 |
| 2 | Chemistry | 88 | 78 | 45 | 84 | 45 |
| 3 | Biology | 56 | 85 | 76 | 57 | 89 |
| 4 | Mathematics | 98 | 89 | 85 | 92 | 80 |

**Output: With VAR Statement:**

Total rows: 2 Total columns: 6

| | SUBJECT | SID_001 | SID_201 | SID_039 | SID_130 | SID_200 |
|---|---|---|---|---|---|---|
| 1 | Physics | 90 | 80 | 87 | 57 | 67 |
| 2 | Mathematics | 98 | 89 | 85 | 92 | 80 |

**Video link: Youtube_Video**

## Q80: What are the default statistics that Proc means procedure produce?

⇨ **Default statistics that proc means produce are N, MIN, MAX, MEAN and STD DEV. Proc means procedure is used to get summary statistics of numeric variable present in the dataset.**

```
PROC MEANS DATA=SASHELP.SHOES;
RUN;
```

**Group wise summary statistics:**

```
PROC SORT DATA =SASHELP.SHOES OUT=SHOES_SORT;
BY PRODUCT;
RUN;
PROC MEANS DATA= SHOES_SORT;
BY PRODUCT;
RUN;
```

**Result:**

The MEANS Procedure

| Variable | Label | N | Mean | Std Dev | Minimum | Maximum |
|---|---|---|---|---|---|---|
| Stores | Number of Stores | 395 | 11.6481013 | 8.8736315 | 1.0000000 | 41.0000000 |
| Sales | Total Sales | 395 | 85700.17 | 129107.23 | 325.0000000 | 1298717.00 |
| Inventory | Total Inventory | 395 | 250898.86 | 351514.63 | 374.0000000 | 2881005.00 |
| Returns | Total Returns | 395 | 2967.32 | 4611.74 | 10.0000000 | 57362.00 |

**Groupwise Statistics(Partial Result):**

The MEANS Procedure

Product=Boot

| Variable | Label | N | Mean | Std Dev | Minimum | Maximum |
|---|---|---|---|---|---|---|
| Stores | Number of Stores | 52 | 16.6153846 | 7.6007065 | 1.0000000 | 33.0000000 |
| Sales | Total Sales | 52 | 45202.75 | 45705.28 | 1179.00 | 286497.00 |
| Inventory | Total Inventory | 52 | 187012.90 | 172214.37 | 374.0000000 | 882080.00 |
| Returns | Total Returns | 52 | 1896.58 | 1652.97 | 80.0000000 | 9160.00 |

Product=Men's Casual

| Variable | Label | N | Mean | Std Dev | Minimum | Maximum |
|---|---|---|---|---|---|---|
| Stores | Number of Stores | 45 | 8.8666667 | 8.4975932 | 1.0000000 | 29.0000000 |
| Sales | Total Sales | 45 | 176304.60 | 224859.90 | 9244.00 | 1298717.00 |
| Inventory | Total Inventory | 45 | 379672.29 | 504818.08 | 2176.00 | 2881005.00 |
| Returns | Total Returns | 45 | 6911.89 | 9505.67 | 478.0000000 | 57362.00 |

**Video link:** Youtube Video

## Q81: What are the differences between proc means and proc summary?

⇨ **There are following differences between proc means and proc summary.**

**By default Proc MEANS produces printed result at the result window.**

```
PROC MEANS DATA=SASHELP.CLASS;
VAR AGE;
RUN;
```

**Result:**

The MEANS Procedure

| Analysis Variable : Age | | | | |
|---|---|---|---|---|
| N | Mean | Std Dev | Minimum | Maximum |
| 19 | 13.3157895 | 1.4926722 | 11.0000000 | 16.0000000 |

**By default proc summary neither produces printed result nor output dataset. We need to provide either print option or output statement.**

```
PROC SUMMARY DATA =SASHELP.CLASS PRINT;
VAR AGE;
RUN;
```

**Result:**

The SUMMARY Procedure

| Analysis Variable : Age | | | | |
|---|---|---|---|---|
| N | Mean | Std Dev | Minimum | Maximum |
| 19 | 13.3157895 | 1.4926722 | 11.0000000 | 16.0000000 |

**Output statement in proc summary produces dataset.**

**With output statement:**

```
PROC SUMMARY DATA =SASHELP.CLASS ;
VAR AGE;
OUTPUT;
RUN;
```

**Result:**

Total rows: 5 Total columns: 4

| | _TYPE_ | _FREQ_ | _STAT_ | Age |
|---|---|---|---|---|
| 1 | 0 | 19 | MAX | 16 |
| 2 | 0 | 19 | STD | 1.4926721594 |
| 3 | 0 | 19 | N | 19 |
| 4 | 0 | 19 | MIN | 11 |
| 5 | 0 | 19 | MEAN | 13.315789474 |

**Without var statement:**

```
PROC MEANS DATA =SASHELP.CLASS;
RUN;
```

**Result:**

The MEANS Procedure

| Variable | N | Mean | Std Dev | Minimum | Maximum |
|---|---|---|---|---|---|
| Age | 19 | 13.3157895 | 1.4926722 | 11.0000000 | 16.0000000 |
| Height | 19 | 62.3368421 | 5.1270752 | 51.3000000 | 72.0000000 |
| Weight | 19 | 100.0263158 | 22.7739335 | 50.5000000 | 150.0000000 |

**Without var statement:**

```
PROC SUMMARY DATA =SASHELP.CLASS PRINT;
RUN;
```

**Result:**

The SUMMARY Procedure

| N Obs |
|---|
| 19 |

**Video link: Youtube_Video**

**Q82: How do you get selected statistics, missing, and most repeated value for numeric variables?**

⇨ **We can use required options to get selected statistics, missing count by using NMISS, and highest repeated value by using MODE option with proc means procedure.**

**Proc means procedure is used to get required statistics by providing appropriate options.**

```
PROC MEANS DATA =SASHELP.HEART N NMISS MIN MAX MEAN MEDIAN MODE;
RUN;
```

**Result:**

The MEANS Procedure

| Variable | Label | N | N Miss | Minimum | Maximum | Mean | Median | Mode |
|---|---|---|---|---|---|---|---|---|
| AgeCHDdiag | Age CHD Diagnosed | 1449 | 3760 | 32.0000000 | 90.0000000 | 63.3029676 | 63.0000000 | 59.0000000 |
| AgeAtStart | Age at Start | 5209 | 0 | 28.0000000 | 62.0000000 | 44.0687272 | 43.0000000 | 36.0000000 |
| Height | | 5203 | 6 | 51.5000000 | 76.5000000 | 64.8131847 | 64.5000000 | 62.5000000 |
| Weight | | 5203 | 6 | 67.0000000 | 300.0000000 | 153.0866808 | 150.0000000 | 138.0000000 |
| Diastolic | | 5209 | 0 | 50.0000000 | 160.0000000 | 85.3586101 | 84.0000000 | 80.0000000 |
| Systolic | | 5209 | 0 | 82.0000000 | 300.0000000 | 136.9095796 | 132.0000000 | 120.0000000 |
| MRW | Metropolitan Relative Weight | 5203 | 6 | 67.0000000 | 268.0000000 | 119.9575245 | 118.0000000 | 113.0000000 |
| Smoking | | 5173 | 36 | 0 | 60.0000000 | 9.3665185 | 1.0000000 | 0 |
| AgeAtDeath | Age at Death | 1991 | 3218 | 36.0000000 | 93.0000000 | 70.5364139 | 71.0000000 | 68.0000000 |
| Cholesterol | | 5057 | 152 | 96.0000000 | 568.0000000 | 227.4174412 | 223.0000000 | 200.0000000 |

**For selected variable and controlling two decimals:**

```
PROC MEANS DATA =SASHELP.HEART N NMISS MIN MAX MEAN MEDIAN MODE MAXDEC=2;
VAR HEIGHT WEIGHT DIASTOLIC SYSTOLIC;
RUN
```

**Result:**

The MEANS Procedure

| Variable | N | N Miss | Minimum | Maximum | Mean | Median | Mode |
|---|---|---|---|---|---|---|---|
| Height | 5203 | 6 | 51.50 | 76.50 | 64.81 | 64.50 | 62.50 |
| Weight | 5203 | 6 | 67.00 | 300.00 | 153.09 | 150.00 | 138.00 |
| Diastolic | 5209 | 0 | 50.00 | 160.00 | 85.36 | 84.00 | 80.00 |
| Systolic | 5209 | 0 | 82.00 | 300.00 | 136.91 | 132.00 | 120.00 |

**Video link: Youtube_Video**

## Q83: How do you count non missing and missing record count for categorical variables ?

⇨ **Non missing and missing categorical variable can be obtained by using proc freq procedure by providing missing option with table statement.**

***It will produce frequency table for non-missing & missing categories:**

```
PROC FREQ DATA=SASHELP.HEART;
TABLE DEATHCAUSE/MISSING;
RUN;
```

**Result:**

The FREQ Procedure

| Cause of Death | | | | |
|---|---|---|---|---|
| DeathCause | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
| | 3218 | 61.78 | 3218 | 61.78 |
| Cancer | 539 | 10.35 | 3757 | 72.13 |
| Cerebral Vascular Disease | 378 | 7.26 | 4135 | 79.38 |
| Coronary Heart Disease | 605 | 11.61 | 4740 | 91.00 |
| Other | 357 | 6.85 | 5097 | 97.85 |
| Unknown | 112 | 2.15 | 5209 | 100.00 |

***It will produce frequency table for non-missing category but gives count for missing category:**

```
PROC FREQ DATA=SASHELP.HEART;
TABLE DEATHCAUSE;
RUN;
```

**Result:**

The FREQ Procedure

| Cause of Death | | | | |
|---|---|---|---|---|
| DeathCause | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
| Cancer | 539 | 27.07 | 539 | 27.07 |
| Cerebral Vascular Disease | 378 | 18.99 | 917 | 46.06 |
| Coronary Heart Disease | 605 | 30.39 | 1522 | 76.44 |
| Other | 357 | 17.93 | 1879 | 94.37 |
| Unknown | 112 | 5.63 | 1991 | 100.00 |
| Frequency Missing = 3218 | | | | |

***It will produce frequency table for non missing & missing categories controlling number of category levels;**

```
PROC FREQ DATA=SASHELP.HEART;
TABLE DEATHCAUSE/MISSING MAXLEVELS=3;
RUN;
```

**Result:**

The FREQ Procedure

| Cause of Death | | | | |
|---|---|---|---|---|
| DeathCause | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
| | 3218 | 61.78 | 3218 | 61.78 |
| Cancer | 539 | 10.35 | 3757 | 72.13 |
| Cerebral Vascular Disease | 378 | 7.26 | 4135 | 79.38 |
| The first 3 levels are displayed. | | | | |

**Video link: Youtube_Video**

**Q84: What happens when proc SQL processes a summary function without a group by clause?**

- **Without group by clause for sum function, it displays all rows, and each row shows total record count. With group by clause, it displays summary statistics of grouping variable.**

***Without group by clause;**

```
PROC SQL;
CREATE TABLE WITHOUT_GROUP_BY AS SELECT REGION,
SUM(RETURNS) AS TOTAL_RETURNS,
MEAN(RETURNS) AS MEAN_RETURNS
FROM SASHELP.SHOES RUN;
```

**Output:**

Total rows: 395 Total columns: 3

| | Region | TOTAL_RETURNS | MEAN_RETURNS |
|---|---|---|---|
| 1 | Africa | 1172092 | 2967.321519 |
| 2 | Africa | 1172092 | 2967.321519 |
| 3 | Africa | 1172092 | 2967.321519 |
| 4 | Africa | 1172092 | 2967.321519 |
| 5 | Africa | 1172092 | 2967.321519 |
| 6 | Africa | 1172092 | 2967.321519 |

***With group by clause;**

```
PROC SQL;
CREATE TABLE WITH_GROUP_BY AS SELECT REGION,
SUM(RETURNS) AS TOTAL_RETURNS,
MEAN(RETURNS) AS MEAN_RETURNS
FROM SASHELP.SHOES
GROUP BY REGION;
RUN;
```

**Output:**

Total rows: 10 Total columns: 3

| | Region | TOTAL_RETURNS | MEAN_RETURNS |
|---|---|---|---|
| 1 | Africa | 74087 | 1322.9821429 |
| 2 | Asia | 10895 | 778.21428571 |
| 3 | Canada | 129394 | 3497.1351351 |
| 4 | Central America/Caribbean | 126898 | 3965.5625 |
| 5 | Eastern Europe | 86701 | 2796.8064516 |
| 6 | Middle East | 206880 | 8620 |
| 7 | Pacific | 77129 | 1713.9777778 |
| 8 | South America | 102851 | 1904.6481481 |
| 9 | United States | 187502 | 4687.55 |
| 10 | Western Europe | 169755 | 2737.983871 |

**Video link**: [Youtube_Video]