| Steps for Creating a SAS Datasets | | | Scope |
|---|---|---|---|
| 1 | **LIBNAME** libref '<Path>'; | Reference a SAS data library | Global |
| 2 | **FILENAME** fileref '<Path>'; | Reference (Temp) an external file | Global |
| 3 | **DATA** 'SASDataSetName'; | Name a SAS data set | |
| 4 | **INFILE** 'file name/fileref' *OBS*=10; **FIRSTOBS=2;** **Dlm=',' DSD;** | Identify an external file using INFILE statement **OBS** mention the ***range till which data needs to be read.*** Can be used in data and proc print. Used to ***verify Data*** reading without affecting RAM space much. ***FIRSTOBS*** will start to read data from row2 of raw dataset, ***DLM/DSD*** is Delimiter and Delimiter sensitive data. | |
| 5 | **INPUT** <informats>; | Describe data | |
| 6 | Sum_var **+** var2; | + is called accumulator variable. ***Defaults to zero initially*** and in case if values are ***missing***. Values get summed as dataset is read. ***+ will automatically retain*** its value | |
| 7 | **Retain** <Sum_var> <val>; | ***Used to initializes Accumulator variable*** which is otherwise 0 by default. | |
| 8 | **IF** <condition> then Vari=Val; | Condition can use any conditional operator: ***=/eq,~=/^=/ne, >=/ge, <=/le, >/gt, </lt, in, &, \|*** Character ***values need to be of same case*** in condition statements, enclosed in " Condition inside ***parenthesis is given high importance***. ***BODMAS*** rule apply here. **0/. = False**, that is 0 or missing is false **1 = True** | |
| 9 | **LENGTH** Var1 $ 10 Var2 20; | By default, SAS ***allocates the space of first value*** it encounters. Numeric variables have ***default size 8***. This ***should be declared before value is set*** | |
| 10 | If <condition> then <stmt>; **Else** if <condi2> then <stmt>; Else <final condition>; | Used for code optimization Better to arrange else-if operation in ***decreasing probability*** to increase performance. | |
| 11 | If <condition> then **DELETE**; | This is used to ***delete an observation using condition***. Used mostly ***along with IF*** | |
| 12 | **DROP =** Var1 / **KEEP =** Var1; | This ***can be used in Data Step as well SAS procedures***. Doesn't apply to all output dataset that are named in Data statement. | |
| 13 | **DROP** Var1 / **Keep** Var1; | ***Cannot be used in proc steps*** ***Applies to all o/p data sets*** Based on the count of variables use Drop and Keep wisely. | |
| 14 | **LABEL** Var='Label Detail'; **FORMAT** Var1 DOLLAR12; | Used to provide a ***permanent label/format*** to a variable. However, when ***used in Proc*** statement ***can override this behaviour***. | |
| 15 | **SELECT** <Var>; **WHEN** ("Val") stmt; **otherwise** <stmt>; **end**; | This is like a Switch-Case statement, this will use ***select – when – otherwise - end*** | |
| 16 | **DO**; <SAS Statements>; **END**; | If loop or when can handle only one stmt, do can handle many statement in its block | |

# Steps for Reading & Combining SAS Datasets

***DATA*** NEWSASDATASET (***DROP***=COL4 COL5); * Col4-5 participate in any data manipulation but not available in the final datasets.

**SET** <Data Set Name> (**...**); * Used to read SAS dataset;

(***DROP =*** COL1 COL2 COL3) * COL1-3 will not participate in any data manipulations.

*Use* ***DROP/KEEP*** *complimentarily based on the number of variables involved*

*POINT – Used for direct access of an observation, should be used along with* ***STOP***

***END = var -*** *Used to read only the last observation in a dataset. Do not use with POINT*

| | | |
|---|---|---|
| 1 | **If** (condition);<br>**If** (condition) **then delete**; | 1. IF statement is used to subset a data<br>2. IF – then – delete is used to drop unnecessary data based on a condition |
| 2 | **If** (condition) **then** Var1=''<br>**Else** var1 = ''; | IF – then – else can be used to create a new column in a SAS data file. Also, called as conditional execution. |
| 3 | **Length** var2 $ 5; | We cannot set length for already existing variable at this stage as they would be already defined. This is used when we need to create a new variable and set explicit length for it. |
| 4 | **Label** var1 = "Variable1"; | Label is used to set the label; this can be seem using proc print with label as its argument. |
| 5 | **Format** var1 COMMA6.; | Used to define the format of the variable |
| 6 | **By** COL1 COL2; | When By is used, data set ***must be sorted*** based on that BY variable before. Use ***PROC SORT DATA=<DS> out = <New DS>; BY Var;*** command for the same.<br>When BY is used, SAS produce ***FIRST.variable*** and ***LAST.variable*** *to keep track on sorted variables data.* Used to fetch first and last observations in Subgroups.<br>***BY can carry more than one variable***; but again, both needs to be sorted before accordingly. |
| 7 | Varname = 5;<br>**SET** <DS> **POINT**=<Var***name***>;<br>**OUTPUT;**<br>**STOP**; | This is used to read an observation using direct access and not sequentially using point and observation number accordingly. Remember***, POINT cannot carry a numeric constant, it can only carry a variable name.*** So, define a variable with an observation number and then use it in POINT. More complex way of using it is in merging the dataset.<br>Because there is no EoF (end of file) just using POINT will create an infinite loop. So, it ***needs to be used with STOP statement***.<br>Again, this will only write data to PDV, to write the observation to a target dataset, we ***need to explicitly OUTPUT the data obtained as part of POINT***. |
| 8 | DATA ***<DS1> <DS2>***;<br>SET <DS>; | We can create one or more dataset like this. Data in <DS> is written to both <DS1> and <DS2>. |
| 9 | **END** = <Variable Name> | Variable name will carry 1 or zero for the last observation. Variable contains the EoF marker. |
| 10 | SET, RETAIN, SUM, _TEMPORARY_ | Will retain its values in PDV for each iteration. Other variables are set to missing values accordingly in each iteration.<br>Difference in reading the data from SAS is that ***for each iteration variables are not assigned to missing*** but values are retained with respective older values. |
| 11 | **_N_, _ERROR_** | _N_ = Initial value is 1 and increment as observations are read<br>_ERROR_ = Initial value is 0 and is set to 1 if error found |

# Steps for Combining SAS Datasets

***DATA*** NEWSASDATASET (***DROP***=COL4 COL5); * Col4-5 participate in any data manipulation but not available in the final datasets.

**SET/MERGE** <Data Set Name> (**…**); * Used to read SAS dataset;

(***DROP =*** COL1 COL2 COL3) * COL1-3 will not participate in any data manipulations.

*Use **DROP/KEEP** complimentarily based on the number of variables involved*

| | | |
|---|---|---|
| 1 | **SET** A;<br>**SET** B; | 1. **One to One Mapping**<br>2. ***Multiple SET statement – No Missing Values – Values skipped***<br>3. Number of observation in new dataset is equal to the number of observation in the smallest original dataset; |
| 2 | **SET** A B C; | 1. **Concatenation**<br>2. ***Single SET statement - Missing Values - No Values skipped***<br>3. Like a sanwidge, one data set sit below the other in a stacked fashion<br>4. Type of common variables should be the same, else SAS throw error<br>5. If no explicit mention of Type, Label, format or informats are made, SAS will automatically derive them from first occurring dataset |
| 3 | **SET** A B C;<br>**BY** ID; | 1. **Interleaving**<br>2. ***Single SET + BY statement - No Missing Value - No Values Skipped;***<br>3. Multiple matching observation for a single observation in BY statement<br>4. Data read based on the order of By Variables defined |
| 4 | **MERGE** A B; | 1. **One to One Match merging**<br>2. ***Single MERGE statement – Missing Values - Values skipped***<br>3. Diff between Concatenation and Simple Merge: Doesn't stops its iteration with the smaller dataset, loop extends to the maximum observations |
| 5 | **MERGE**<br>A (in=inA RENAME=(VarA=VariableA))<br>B(in=inB RENAME=(VarA=VariableB)**;**<br>**BY** DESCENDING ID;<br>If inA = 1 and inB=1; | 1. Simple Match Merging<br>2. ***Single MERGE + BY statement - Missing Values - No Values skipped***<br>3. PDV will retain its value until the value for all BY variables changes;<br>4. Order of Sorting can be changed to descending by mentioning **DECENDING** after BY Statement;<br>5. It must ***also be done in all PROC SORT steps*** and as well in merge statement accordingly;<br>6. In case any two datasets has same column name, SAS will overwrite the data with the latest data it encounters, to prevent this we can rename the matching variables using **RENAME**.<br>7. **IN** is a temporary variable, used to select only the observations that appear in both dataset<br>8. **DROP/KEEP** in ***DATA statement means*** drop those variable as part of DROP in target dataset<br>9. **DROP/KEEP** in ***merge statement means*** don't even consider while merging, drop them even before PDV is completely formed |

# DO Loop – Generating Data with DO loop

***DATA*** NEWSASDATASET (***DROP***=Var); * BY default SAS will print iterating variable too, to avoid it in target dataset explicit DROP needed;

```
 DO Var = 2 TO 10 BY 2; * Default increment is 1, can also use -1 to decrement;
        <Statements>;
 END;
```

| | | |
|---|---|---|
| 1. | DO Var = 1, 2, 3, 4, 5;<br>   **OUTPUT**;<br>END; | 1. This is used to specify the series of items as part of iteration.<br>2. We will not have start, stop, increment or decrement values.<br>3. **OUTPUT** will force SAS to write data from PDV to Target dataset and print it as result during execution.<br>4. Difference between ***Out*** and ***Output*** is, ***out*** is used to create a new dataset itself, generally used in PROC SORT; However, ***output*** is like a print statement. |
| 2. | DO **Var1** = 1 to 5;<br>   DO **Var2** = 1 to 3;<br>      <Statements>;<br>   END;<br>END; | 1. This is called nested DO loop<br>2. While using nested DO loop be careful in using the ***increment variable***, ***it should be different with variable used in outer loop***, else value will get overwritten in PDV and will cause undesired output |
| 3. | **DO UNTIL** (Expression);<br>   <Statements>;<br> **END;** | 1. **Executes** the statements mentioned with in the do loop **at least once**. |
| 4. | **DO WHILE** (Expression);<br>   <Statements>;<br> **END;** | 1. **Executes *only when the expression is true*** at the first stage, else loop will not even execute. |
| 5. | **DO** sample=10 to 50 by 10;<br> SET Clinic.Cap2000 **POINT**=sample;<br> **OUTPUT**;<br>end;<br>**STOP**; | 2. This is used to create a sample out of a dataset, which can be generally used during model building<br>3. We will use ***Do loop + POINT + OUTPUT + STOP*** to derive this<br>4. However, these are not random samples<br>5. In this example, we are trying to create a sample by picking observations with observation number 10, 20, 30, 40 and 50 |
| 6 | **END**; | 1. End will terminate the loop |

# ARRAYS – Processing Variables with ARRAYS

***DATA*** NEWSASDATASET (*DROP*=Array Name); * Array Incrementor can be dropped

**ARRAY** *<Array Name> {**Size**} Element1 Element2 Element3…ElementN;*

| | | |
|---|---|---|
| 1 | ARRAY Quizs[2] Quiz1 Quiz2;<br>ARRAY Quizs{5} Quiz6 - Quiz10;<br>ARRAY NUMS{6:10} Num6 - Num10;<br>ARRAY Sales[3] Sale1 - Sale3;<br>ARRAY Days(7) Day1 - Day7; | 1. This is a one-dimensional array, all variables in array must be either Number or Character<br>2. Default array size is 1<br>3. Array elements must be of same type<br>4. Array lives only within data step, outside data step it will expire<br>5. Array size can be mentioned inside **[], {} and ()** |
| 2 | ARRAY Nums(*****) **_NUMERIC_**;<br>ARRAY Chars{*} **_CHARACTER_**;<br>ARRAY Alls[*] **_ALL_**; | 1. One dimensional array can be created with *<br>***2.*** _NUMERIC_ implies ***numeric variables***<br>3. _CHARCTER_ implies ***character variable***<br>4. _ALL_ implies ***all variable type*** |
| 3 | ARRAY Scores[2] Score1 Score2;<br>Scores[**1**] = 89; | 1. Array element can be referenced using array name and element number.<br>2. Scores[1] refer the first element in array variable scores.<br>3. Remember ***SAS starts its indexing from 1*** |
| 4 | array weights[4] weight1-weight4;<br>**DO** i = 1 to **DIM**(weights);<br>weights[i] = weights[i] * 2.24;<br>**END**; | 1. Array elements are generally ***accessed through DO loop***<br>2. **DIM** is used to get the dimension size of an array.<br>3. ***Default*** array dimension size is **1** |
| 5 | array sizes[2] **$** 32;<br>sizes[1]="PRADEEPSATHYAM"; | 1. Use **$** to declare a character variable;<br>2. **Default** Character length **is 8**;<br>3. If you need to increase the character element size, it needs to be mentioned after $; |
| 6 | array Nums[3] (1,2,3);<br>array Digts[4] (1 2 3 4);<br>array Names[2] **$** (**'Prady'**,**'Srut'**);<br>array Temp[2] **_TEMPORARY_** (6,7); | 1. There are some of the ways to initialize values to the arrays.<br>2. ***_TEMPORARY_*** is used to ***initialize an array temporarily*** inside SAS.<br>3. Values can be initialized with a space or comma separator, for Char $ is used.<br>4. One dimensional array is used to do ***column wise manipulation*** for a ***single observation***.<br>5. One dimensional array without any elements will create default variables in the SAS. |
| 7 | array Temps[**3,4**] Temp1-Temp12; | 1. Multi-dimensional array is created by ***mentioning the dimension size of Row and Column*** while declaring array.<br>2. ***[3,4]*** implies 3 rows and 4 columns, thus totally 3*4 = 12 elements.<br>3. These are ***accessed with nested DO loops*** by referencing individual element at Row and Column level respectively.<br>4. Two dimensional arrays can be used to do ***row wise manipulation for multiple observations.*** |

## Column Style: [Standard Data + ***Well Ordered in Column]***

```
1--------10---------20---------30--------40---------50---------60---------70---------80--------90
124      61        Mod      Male       Pradeep                United States
123      76        Ded      Female     Sruthi                 India
142      89        Reg      Male       Sathyamurthy           United Kingdom
```

| Special SAS Constants | |
|---|---|
| **Example** | **Description** |
| 3.<br>Input() | Numeric |
| "PRADY"\|\|""<br>Put() | String |
| `'25dec2012'd` | Date |
| `'25dec2012:3:45:12pm'dt` | Date Time |
| `'3:45:12pm't` | Time |
| `'09'x` (tab)<br>`'0c'x` (form feed) | Hex Character |

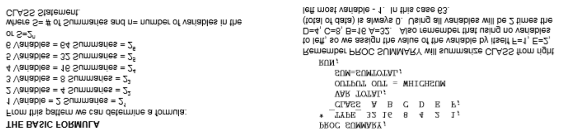
| **PROC PRINT DATA=DATASETNAME**<br>NOOBS *used to avoid printing observation column while printing;<br>DOUBLE *print double spacing in SAS Output and not in SAS Report;<br>(OBS=3) * Print only the first 3 observation of the dataset in print; | | **Scope** |
|---|---|---|
| **Sum** <Col Name>; | Calculate the sum of the column | Local |
| **VAR** <Col Name>; | Mention the variable and its ***order*** of printing | Local |
| **Label** <Col Name>=’’; | Define label name for a column<br>Can mention up to ***256 char***<br>***Can be defined in single or multiple lines*** | **Local** |
| **Where** <Col/col condi><br>**CONTAINS** ‘str’;<br>**?** ‘str’;<br>**IN**(‘str1’,’str2’); | Defines the column condition<br>***=, ^=, >, <, >=, <=***<br>***CONTAINS*** *is string comparison*<br>***AND, OR*** *operator used along with col name each time*<br>***IN*** *operator is used as SQL style in comparison.* | Local |
| **ID** <Col Names>; | Act as a primary key, ***replace OBS*** column without explicitly mention of NOOBS.<br>ID used along with ***Var*** will ***display*** a ***column twice***. | Local |
| **SUM** <Col Name>; | Will provide the total of the column specified. | Local |
| **BY** <Col Name>; | Col Name should be ***same as one that is sorted before*** using this. ***Subset results***. | Local |
| **BY** <Col Name1>;<br>**ID** <Col Name1>; | When ID used along with BY it will:<br>1. ***Supress OBS*** column<br>2. ID/BY variable name is printed in left col<br>3. Each ***ID/BY value is printed only once*** at the start of each by group and on the line, that has group sub-total. | Local |
| **By** <Col Name1>;<br>**PAGEBY**<Col Name1>; | Mostly used along with sum-by-id.<br>***Column used in PAGEBY should be same as one used in BY***.<br>Used to ***print each sub-total on a separate page***. | Local |
| **FORMAT** <Col Name>; | When defined ***inside PROC it scopes within it***. To make it ***permanent FORMAT or Labels*** need ***to be defined in DATA step*** | Local/Global |
| **TITLE** ‘str1’; | Generally, ***need to be defined outside a PROC step***.<br>However, it can be used inside PROC too<br>***TITLE is global.*** Once defined will stay forever until title statement is modified, cancelled or end SAS session.<br>***Cancel of title*** is done by ***title;*** | Global |
| **FOOTNOTE** ‘str2’; | Used to print note below a table/graph<br>It is same as TITLE function, up to ***10 footnotes can be defined*** in SAS.<br>***Cancel of footnote*** is done by:<br>***Footnote;*** | Global |

| **PROC SORT DATA=DATASETNAME**<br>**OUT=DATASETNAME** ***o/p SAS dataset** | | |
|---|---|---|
| **by** <Col Name>;<br>**by descending** <col1> | Sorted by the column mentioned, sort takes place from right to left columns mentioned.<br>If used with descending it will apply to ***column which is immediately after it***, rest of the other columns will be sorted in ascending order. | Local |
| **NOTSORTED**; | To explicitly mention not to sort if the ***values are equal*** based on by condition. | Local |
| | | |

| **PROC FORMAT LIB=library**<br>**LIBRARY/LIB** ***Defines the SAS library that needs to be referred;**<br>**FMTLIB** ***print all the user defined format present in the Library mentioned;** | | | **Scope** |
|---|---|---|---|
| 1 | **LIBNAME *library*** '<Path>'; | Reference a SAS data library | Permanent |
| 2 | **PROC FORMAT LIB**=library<br>**FMTLIB**; | Library can be the SAS library referred above or it can be a catalog like ***library.catalog.***<br>***FMTLIB*** will list all the user defined format present in the library. ***formats.sas7bcat*** file is created in the path mentioned in library. | Permanent |
| 3 | **Value** <format-name> | Format name must begin with ***$ for Char*** var<br>Cannot be > 8 char in length<br>Cannot be the name of existing SAS format<br>Cannot end with a number<br>Does not end with a period when defined | Permanent |
| | Range1='label1' | Range1= Actual Column Data<br>Label1= Description of Range1<br>***Numeric*** => 102='Manager'<br>***Character*** => 'A'='Good Performance'<br>***Range*** => low-<12='Not Teen Age' | Permanent |
| | Range2='label2'; | Always only the ***last Range must be ended with;*** which implies SAS that PROC FORMAT statement ends. | Permanent |
| 4 | **PROC FORMAT;** | This format will be created in the ***work directory*** which ***means temporary***. | Temporary |
| | Value <format-name> | Scope within that SAS session only | Temporary |
| | Range1='label1' | Scope within that SAS session only | Temporary |
| | Range2='label2'; | Scope within that SAS session only | Temporary |
| 5 | **PROC CATALOG;** | You can delete the user defined format | Permanent |

| **PROC REPORT DATA=<DATASETNAME>**<br>**WD/NOWD *Decides should the o/p be printed in a dedicated report window;**<br>**DOUBLE *print double spacing in SAS Output and not in SAS Report;**<br>**SPLIT='<symbol>' * Symbol can be *, # $ etc., Used to define the label split in reporting;** | | | **Scope** |
|---|---|---|---|
| 1 | **COLUMN** <Col Names> | Used ***to subset the column*** that is needed to be displayed in the report. | Local |
| 2 | **WHERE** <Col Condi/Name><br>**In** ('value1','value2') | Used to ***filter out the data*** required<br>In ***used along with where to filter the data*** based on values provided, SQL style usage. | Local |
| 3 | **DEFINE** <Col1>/<***usage***><br>DEFINE <Col2>/<***attribute***><br>DEFINE <Col3>/<***options***><br>DEFINE <Col4>/<**Justify**><br>DEFINE <Col5>/<***Col*** Heading><br><br>`* Column definition;`<br>`PROC REPORT DATA=CARS_SAMPLE`<br>`NOWD SPLIT='*' HEADLINE`<br>`HEADSKIP;`<br>`      define Make/format=$CHAR8.`<br>`width=3 spacing=10;`<br>`      define Type/'Car*Type';`<br>`      define Model/center;`<br>`      define Cylinders/order`<br>`DESCENDING;`<br>`      define Cylinders/group;`<br>`RUN;`<br><br>`* Column definition - usage of`<br>`group definition;`<br>`PROC REPORT DATA=CARS_SAMPLE`<br>`NOWD SPLIT='*' HEADLINE`<br>`HEADSKIP;`<br>`      column cylinders MSRP;`<br>`      define cylinders/group;`<br>`RUN;`<br><br>`* Specifying statistics;`<br>`PROC REPORT DATA=CARS_SAMPLE`<br>`NOWD SPLIT='*' HEADLINE`<br>`HEADSKIP;`<br>`      column cylinders MSRP;`<br>`      define cylinders/group;`<br>`      define MSRP/mean 'Average`<br>`of MSRP';`<br>`RUN;`<br><br>`* Column definition - usage of`<br>`across definition;`<br>`PROC REPORT DATA=CARS_SAMPLE`<br>`NOWD SPLIT='*' HEADLINE`<br>`HEADSKIP;`<br>`      column cylinders type`<br>`MSRP;`<br>`      define cylinders/across;`<br>`      define type/across;`<br>`RUN;` | Used to ***build column definitions*** in report like column space and width, etc.,<br>Let to ***define more than one column attribute*** at a time.<br>Column can be defined ***in any order*** and list ***options within it in any order as well***.<br><br>***Usage*** specifies ***how to use the variables***:<br>By ***default***, **Char** Variable defined as ***Display***<br>And ***Numeric*** variables defined as ***Analysis***<br>1. **Across** – Displays variable ***horizontally*** rather vertically<br>2. **Analysis** - Default ***SUM*** analysis.<br>3. **Computed** – ***position*** of compute variable is ***very important. Use compute*** and ***endcomp*** and ***derive the value*** with some formula<br>4. ***Display*** – This is for ***Char*** variables<br>5. ***Group*** – to create ***summary report***. To get a proper result, display/character variables need to be grouped properly.<br>6. **Order** – This is like Grouping and Order, by ***default it is ordered in ascending***, if needed we need explicit mention of value ***DESCENDING***.<br><br>***Attributes*** specifies the ***look*** of each column:<br>Width and spacing has its ***effect only in o/p window*** and doesn't affect HTML window.<br>1. **Format** – define SAS/user format, default is ***its variable type***<br>2. **Width** – width of col, default is ***Max***<br>3. **Spacing** – No of blank char, default is ***2***<br><br>***Options*** specifies the ***further formatting*** option:<br>**1. DESCENDING**<br>**2. NOPRINT**<br>**3. NOZERO**<br>**4. PAGE**<br><br>***Justification*** specifies ***arrangements*** of column:<br>1. **Center** – Justify the char in centre<br>2. **Left** – ***default for chars*** n left justify<br>3. **Right** – ***default for num*** n right justify | Local |
| ***Column Heading*** is the ***label definition***. ***Split*** in report definition is used to ***split the column label*** as needed. (e.g. ***SPLIT='*';***) define col/c*t; | | | |

| SI.NO | Statistics | Definition |
|---|---|---|
| 1 | CSS | Corrected sum of squares |
| 2 | USS | Uncorrected sum of squares |
| 3 | CV | Coefficient of variation |
| 4 | MAX | Maximum value |
| 5 | MEAN | Average |
| 6 | MIN | Minimum Value |
| 7 | N | Number of observations with non-missing values |
| 8 | NMISS | Number of observations with missing values |
| 9 | RANGE | Range |
| 10 | STD | Standard deviation |
| 11 | STDERR | Standard error of the mean |
| 12 | SUM | Sum |
| 13 | SUMWGT | Sum of the `Weight` variable values |
| 14 | PCTN | Percentage of a cell or row frequency to a total frequency |
| 15 | PCTSUM | Percentage of a cell or row sum to a total sum |
| 16 | VAR | Variance |
| 17 | T | Student's *t* for testing the hypothesis that the population mean is 0 |
| 18 | PRT | Probability of a greater absolute value of student's *t* |

| *Computing Statistics for Numeric Variable* | | | |
|---|---|---|---|
| **PROC MEANS DATA=<DATASETNAME>**<br>*By default gives descriptive statistics, with n-count of all non-missing values;<br><STATS KEYWORDS> *To suppress default o/p and choose what stats is required for o/p;<br>MAXDEC=2 *To set the decimal point;<br>NOPRINT *Supress the result being printed; | | | **Scope** |
| 1 | **VAR** <Col Names>; | Used to display the ***variables for which the statistics are required*** | Local |
| 2 | **CLASS** <Col Names>; | Specifies categorical variables which needed ***group processing*** | Local |
| 3 | **OUTPUT**<br><***STATS***>=<Col Names><br>***OUT*** = <O/p dataset> | ***Output*** is used to ***structure the final output of the PORC MEAN*** above the segregation done based on a class variable.<br><***STATS***> can be any ***statistic key-word*** and col name specifies on which columns it needs to be applied.<br>If <STATS> keywords are ***not mentioned***, then ***SAS will produce*** whole statistics and add ***_STAT_ variable along with _TYPE_ and _FREQ_***<br><br>***_TYPE_*** is a ***simple binary pattern to summarise the CLASS variable***.<br><br>THE BASIC FORMULA<br>From this pattern we can determine a formula:<br>1 Variable = 2 Summaries = $2^1$<br>2 Variables = 4 Summaries = $2^2$<br>3 Variables = 8 Summaries = $2^3$<br>4 Variables = 16 Summaries = $2^4$<br>5 Variables = 32 Summaries = $2^5$<br>6 Variables = 64 Summaries = $2^6$<br>or $S=2^n$<br>where S= # of Summaries and n= number of variables in the CLASS Statement.<br>`PROC SUMMARY;`<br>`* _TYPE_ 32 16 8 4 2 1;`<br>`CLASS A B C D E F;`<br>`VAR TOTAL;`<br>`OUTPUT OUT = WHICHSUM`<br>`SUM=SUMTOTAL;`<br>`RUN;`<br>Remember PROC SUMMARY will summarize CLASS from right to left, so we assign the value of the variable by itself F=1, E=2, D=4, C=8, B=16 A=32. Also remember that using no variables (total of data) is always 0. Using all variables will be 2 times the left most variable - 1. In this case 63.<br><br>***_FREQ_*** is the count of class variable occurrence<br><br>***OUT*** specifies the ***output dataset*** in which the final ***statistic result needs to be stored***. | |

| *Computing Statistics for Numeric Variable* | | | |
|---|---|---|---|
| **PROC SUMMARY DATA=<DATASETNAME> PRINT;** | | | **Scope** |
| 1 | **VAR** <Col Names>; | Used to display the ***variables for which the statistics are required*** | |
| 2 | **CLASS** <Col Names>; | Specifies categorical variables which needed ***group processing*** | |
| 3 | **OUTPUT**<br><***STATS***>=<Col Names><br>***OUT*** = <O/p dataset> | ***Output*** is used to ***structure the final output of the PORC MEAN*** above the segregation done based on a class variable. | |

## Descriptive Statistics

| SI.NO | Keywords | Definition |
|---|---|---|
| 1 | **CLM** | Two-sided confidence limit for the mean |
| 2 | **CSS** | Corrected sum of squares |
| 3 | **CV** | Coefficient of variation |
| 4 | **KURTOSIS / KURT** | Kurtosis |
| 5 | **LCLM** | One-sided confidence limit below the mean |
| 6 | **MAX** | Maximum value |
| 7 | **MEAN** | Average |
| 8 | **MIN** | Minimum value |
| 9 | **N** | Number of observations with non-missing values |
| 10 | **NMISS** | Number of observations with missing values |
| 11 | **RANGE** | Range |
| 12 | **SKEWNESS / SKEW** | Skewness |
| 13 | **STDDEV / STD** | Standard deviation |
| 14 | **STDERR / STDMEAN** | Standard error of the mean |
| 15 | **SUM** | Sum |
| 16 | **SUMWGT** | Sum of the Weight variable values |
| 17 | **UCLM** | One-sided confidence limit above the mean |
| 18 | **USS** | Uncorrected sum of squares |
| 19 | **VAR** | Variance |

## Quantile Statistics

| SI.NO | Keywords | Definition |
|---|---|---|
| 1 | **MEDIAN / P50** | Median or 50th percentile |
| 2 | **P1** | 1st percentile |
| 3 | **P5** | 5th percentile |
| 4 | **P10** | 10th percentile |
| 5 | **Q1 / P25** | Lower quartile or 25th percentile |
| 6 | **Q3 / P75** | Upper quartile or 75th percentile |
| 7 | **P90** | 90th percentile |
| 8 | **P95** | 95th percentile |
| 9 | **P99** | 99th percentile |
| 10 | **QRANGE** | Difference between upper and lower quartiles: Q3-Q1 |

## Hypothesis Testing

| SI.NO | Keywords | Definition |
|---|---|---|
| 1 | **PROBT** | Probability of a greater absolute value for the *t* value |
| 2 | **T** | Student's *t* for testing the hypothesis that the population mean is 0 |

| *Computing Statistics for Categorical Variable* | | | |
|---|---|---|---|
| **PROC FREQ DATA=<DATASETNAME>** **WD/NOWD *Decides should the o/p be printed in a dedicated report window;** | | | **Scope** |
| 1 | **TABLE** <Col Names> / **NOCUM**; | Used to ***mention the column names based on which a frequency table*** needs to be constructed. ***One column name*** in TABLE will construct a ***simple frequency table with frequency and cumulative frequency and percentage***, totally 4 outputs. ***NOCUM*** will ***supress the display of cumulative frequency and percentage*** from the output. | Local |
| 2 | **TABLE** <COL1> - <COL5> | This will again create simple frequency table for columns-1 to column-5 | Local |
| 3 | PROC FORMAT;<br>Value <frmt_name> range1 'label-1'<br>Range2 'label-2'<br>Range3 'label-3'<br>RUN;<br><br>**PORC FREQ** data=<datasetnames>;<br>**Tables** <cat_col_name>;<br>***Format weight <frmt_name>.***; | | |
| 4 | **TABLE** <COL1> * <COL2>; | This will ***create two-way table***. This will ***cross tabulate*** 2 different categorical variables. | Local |
| 5 | **TABLE** <COL1> * <COL2> * <COL3>; | This will ***create N-way table***. This will ***cross tabulate*** N different categorical variables. | Local |
| 6 | **TABLE** <COL1> * <COL2> / ***CROSSLIST***; | ***CROSSLIST*** will ***display cross tabulation in a ODS format***. This ***ODS output can be customized using*** the ***TEMPLATE*** procedure. | Local |
| 7 | **TABLE** <COL1> * <COL2> / ***LIST***; | Produce list output for crosstabulation. Puts frequency table in a simple and short table. | Local |
| | **TABLE** <COL1> * <COL2> / ***nofreq nopercent norow nocol***; | ***Nofreq*** will ***supress the cell frequency***<br>***Nopercent*** will supress the ***cell percentage***<br>***Norow*** will supress ***row percentages***<br>***Nocol*** will supress ***column percentage*** | Local |