\* SAS : FIRST. AND LAST. VARIABLES

First. and Last. Variables

FIRST.VARIABLE assigns the value of 1 for the first observation in a BY group and

the value of 0 for all other observations in the BY group.

LAST.VARIABLE assigns the value of 1 for the last observation in a BY group and

the value of 0 for all other observations in the BY group.

Note : Data set must be sorted BY group before applying FIRST. and LAST. Variables.;

DATA mylib.Cust;

INPUT ID Product$ DATE DATE10.;

FORMAT DATE date9.;

DATALINES;

1001 P1 01JAN2011

1002 P1 03JAN2011

1003 P2 05JAN2011

1001 P3 03JAN2011

1002 P4 05JAN2011

1003 P1 07JAN2011

1001 P2 05JAN2011

1002 P3 07JAN2011

1003 P4 09JAN2011

1004 P1 07JAN2011

;

Run;

PROC PRINT data=mylib.Cust;

RUN;

PROC SORT DATA=mylib.Cust;

BY ID date;

RUN;

DATA mylib.Cust1;

SET mylib.Cust;

BY ID;

Fst=First.ID;

lst=last.ID;

RUN;

PROC PRINT data=mylib.Cust1;

RUN;

\* Note : FIRST./LAST. variables are temporary variables. That means they are not visible in

the newly created data set. To make them visible, we need to create two new variables. In

the program above, i have created Fst and lst variables.

FIRST.variable = 1 when an observation is the first observation in each group values

of variable ID.

FIRST.variable = 0 when an observation is not the first observation in each group values

of variable ID.

LAST.variable = 1 when an observation is the last observation in each group values

of variable ID.

LAST.variable = 0 when an observation is not the last observation in each group values

of variable ID.

When FIRST.variable = 1 and LAST.VARIABLE = 1, it means there is only a single value

in the group. (See ID = 1004 in the above data for reference)

Selecting First Observation within a Group;

DATA mylib.Cust1;

SET mylib.Cust;

BY ID;

IF FIRST.ID=1;

RUN;

PROC PRINT data=mylib.Cust1;

RUN;

\* Selecting Last Observation within a Group;

DATA mylib.Cust1;

SET mylib.Cust;

BY ID;

IF LAST.ID=1;

RUN;

PROC PRINT data=mylib.Cust1;

RUN;

\* Retain Statement

Advantages of Retain Statement

1) Initialize Variable

2) Preserve a variable’s value from the previous iteration of the DATA step ;

Data mylib.Sales;

Input Day$ Sales;

Datalines;

D1 120

D2 300

D3 678

D4 789

D5 123

D6 234

D7 345

;

run;

Proc Print Data=mylib.Sales;

Run;

Data mylib.CumulativeSales;

set mylib.Sales;

retain Cum\_Sales 0;

Cum\_Sales=Cum\_Sales+Sales;

run;

Proc Print Data=mylib.CumulativeSales;

Run;

\* SAS MERGING TUTORIAL;

DATA mylib.Emp;

INPUT EmpID AGE GENDER $;

DATALINES;

1001 22 M

1002 25 F

1003 28 M

1004 22 F

1005 34 M

1007 30 M

;

RUN;

DATA mylib.Dept;

INPUT EmpID DeptID DeptName$;

DATALINES;

1001 21 Computer

1002 23 Civil

1004 21 Computer

1008 23 Civil

;

Run;

\* Important Steps when using MERGE Statement

Step 1 : Both the data sets must be SORTED by the variable you want to use for merging

Step 2 : The variable you want to use for merging must have same name in both the datasets;

DATA mylib.BOTH;

MERGE mylib.Emp mylib.Dept;

BY EmpID;

RUN;

Proc print data = mylib.emp;

run;

Proc print data = mylib.dept;

run;

Proc print data = mylib.both;

run;

\* Merge two datasets - Join Operations with IN option;

DATA mylib.BOTH;

MERGE mylib.Emp(in=a) mylib.Dept(in=b);

BY EmpID;

x=a;

y=b;

RUN;

Proc print data = mylib.both;

run;

\* The IN= option tells SAS to create a flag that has either the value 0 or 1. If the

observation does not come from the dataset, then the flag returns 0. If the observation

comes from the data set, then the flag returns 1. Since the IN= option creates temporary

variables, we need to create permanent variables so that we can see the flag in the dataset.

With this lines of code "x = a" and "y = b", we tell SAS to create two variables named x, y and

put the same values as stored in variables x and y. You can assign any name you want, not

just x or y;

\* The value 1 in variable x implies these rows come from dataset emp and 0 implies these

rows do not come from dataset emp. The same logic holds for variable y. When variable y has 1,

it means these rows come from dataset dept.

\* Merge two datasets - Inner Join;

DATA mylib.BOTH;

MERGE mylib.Emp(in=a) mylib.Dept(in=b);

BY EmpID;

if a=1 and b=1;

RUN;

Proc print data = mylib.both;

run;

\* Merge two datasets - Left Join;

DATA mylib.BOTH;

MERGE mylib.Emp(in=a) mylib.Dept(in=b);

BY EmpID;

if a=1;

RUN;

Proc print data = mylib.both;

run;

\* Merge two datasets - Right Join;

DATA mylib.BOTH;

MERGE mylib.Emp(in=a) mylib.Dept(in=b);

BY EmpID;

if b=1;

RUN;

Proc print data = mylib.both;

run;

\* Proc Statements;

\* Proc Contents - Listing the Contents of a SAS Data Set;

\* PROC CONTENTS DATA = Datasetname;

\* RUN;

\* The output starts with information about your data set and then describes each variable.

For the data set

\_ Data set name

\_ Number of observations

\_ Number of variables

\_ Date created

\_ Data Sorted or Not

\_ Data Indexed or Not

For each variable

\_ Type (numeric or character)

\_ Length (storage size in bytes)

\_ Format for printing (if any)

\_ Informat for input (if any)

;

DATA mylib.Transaction;

INPUT CustId CustName $ DoB Sales Profit;

Informat DoB MMDDYY8. Sales Dollar11.2 Profit Dollar8.2;

FORMAT DoB Date9. Sales Comma10.2 Profit Comma7.2;

CARDS;

53 Susie 07-11-81 $12,254.45 $190.34

54 Charlie 10-26-54 $10,294.31 $169.87

55 Calvin 01-10-81 $13,125.65 $124.98

56 Lucy 01-13-55 $11,254.56 $109.12

;

RUN;

Proc Print Data=mylib.Transaction;

run;

\* Use PROC CONTENTS to describe data set Transaction;

PROC CONTENTS Data=mylib.Transaction out=abc;

RUN;

\* Proc Freq - list of counts;

\* PROC FREQ Data = Input\_Dataset;

\* TABLES Variable-Combitaionoptions;

\* RUN ;

\* Tables - Specify frequency or cross-tabulation tables and request tests and measures of

association

Varibale-Combitaion

When you have counts for one variable, they are called one-way frequencies.

When you combine two or more variables, the counts are called two-way, threeway,

and so on up to n-way frequencies or simply cross-tabulations.;

data mylib.store;

input storeid$ custid$ month sales;

cards;

s1 c1 1 100

s1 c1 1 260

s1 c1 2 150

s1 c1 2 300

s1 c1 2 250

s1 c2 1 160

s1 c2 1 290

s1 c2 1 190

s1 c2 2 100

s1 c2 2 289

s1 c2 2 280

s2 c1 1 190

s2 c1 1 290

s2 c1 2 140

s2 c1 2 130

s2 c1 2 220

s2 c2 1 180

s2 c2 1 110

s2 c2 1 530

s2 c2 2 120

s2 c2 2 210

;

run;

Proc Print data=mylib.store;

run;

\* One Way Frequency;

proc freq data=mylib.store;

tables custid;

run;

\* Two Way Frequency;

proc freq data=mylib.store;

tables month\*custid;

run;

\* Three Way Frequency;

proc freq data=mylib.store;

tables storeid\*month\*custid;

run;

\* Options - Check all options one by one;

\* The option NOCUM tells SAS to not to return cumulative scores

\* The option nopercent tells SAS to not to return percent distribution

\* The NOROW option hides row percentage in cross tabulation.

\* Similarly, NOCOL option suppresses column percentage.

\* List optio is used to show Table in List Form

\* OUT option is used to store result in a data file;

proc freq data=mylib.store;

tables month\*custid / nocum nopercent norow nocol List out=mylib.Freqout;

run;

proc print data= mylib.Freqout;

run;

\* Include Missing Values in Calculation;

\* By default, PROC FREQ does not consider missing values while calculating percent and

cumulative percent. The number of missing values are shown separately (below the table);

Proc freq data=sashelp.heart;

Tables deathcause;

Run;

\* By adding MISSING option, it includes missing value as a separate category and all

the respective statistics are generated based on it.;

Proc freq data=sashelp.heart;

Tables deathcause/missing;

Run;

/\* PROC MEANS is one of the most common SAS procedure used for analyzing data.

It is mainly used to calculate descriptive statistics such as mean, median, count,

sum etc. It can also be used to calculate several other metrics such as percentiles,

quartiles, standard deviation, variance.;

PROC MEANS Data = datasetname options;

BY variable-list;

VAR variable-list;

RUN ;

BY variable-list - The BY statement performs separate analyses for each level of the

variables in the list.

Note:- The data must first be sorted in the same order as the BY variable-list.

VAR variable-list - The VAR statement specifies which numeric variables to use

in the analysis. If it is absent then SAS uses all numeric variables.

\*/

DATA mylib.sales;

INPUT CustomerID $ SaleDate MMDDYY10. ProductA ProductB ProductC;

format SaleDate date9.;

Month = MONTH(SaleDate);

Cards;

756-01 05/04/2001 120 80 110

756-01 05/05/2001 100 180 120

756-01 05/06/2001 10 18 12

834-01 05/12/2001 90 160 60

834-01 05/18/2001 50 100 75

834-01 06/01/2001 80 60 100

756-01 06/11/2001 100 160 75

834-01 06/19/2001 60 60 60

756-01 06/25/2001 85 110 100

;

run;

proc print data=mylib.sales;

run;

\* If PROC MEANS is used with no other option it gives number of non-missing values,

mean, std, min and max for all variables;

PROC MEANS DATA =mylib.sales;

VAR ProductA ProductB ProductC;

RUN;

\* By default, PROC MEANS generates N, Mean, Standard Deviation, Minimum and

Maximum statistics.

The most frequent statistical options used in PROC MEANS are listed below against

their description.

Statistical Option Description

N Number of observations

NMISS Number of missing observations

MEAN Arithmetic average

STD Standard Deviation

MIN Minimum

MAX Maximum

SUM Sum of observations

MEDIAN 50th percentile

P1 1st percentile

P5 5th percentile

P10 10th percentile

P90 90th percentile

P95 95th percentile

P99 99th percentile

Q1 First Quartile

Q3 Third Quartile

VAR Variance

RANGE Range

QRANGE Quartile range

;

PROC MEANS DATA =mylib.sales sum max n;

VAR ProductA ;

RUN;

\* Group the analysis - can use the CLASS statement to accomplish this task.

It is equivalent to GROUP BY in SQL.;

PROC MEANS DATA =mylib.sales;

Class Month;

VAR ProductA ProductB ProductC;

RUN;

\* Grouping and Output in Separate Tables - Use by Statement in Proc Means

Note:- When we use by statement in any Proc Step, data should be sorted based on

variables mentioned in By statement;

PROC SORT DATA = mylib.sales;

BY Month;

run;

PROC MEANS DATA =mylib.sales;

BY Month;

VAR ProductA ProductB ProductC;

RUN;

\* Difference between CLASS and BY statement

The CLASS statement returns analysis for a grouping (classification) variable in a single

table whereas BY statement returns the analysis for a grouping variable in separate tables.

Another difference is CLASS statement does not require the classification variable to be

pre-sorted whereas BY statement demands sorting.;

\* Writing summary statistic into a SAS dataset

OUTPUT OUT = data-set statistic(variable-list) = name-list;

PROC MEANS DATA = mylib.sales;

by Month;

VAR ProductA ProductB ProductC;

OUTPUT OUT = mylib.salesout MEAN(ProductA ProductB) = MproductA MProductB

sum(ProductA)=SproductA;

RUN;

Proc Print Data=mylib.salesout;

run;

/\* Proc Print - Print your data

PROC PRINT DATA = dataset;

By Variable-list;

Id Variable-list;

Sum Variable-list;

Var Variable-list;

Run;

BY variable-list - The BY statement starts a new section in the output for each new value

of the BY variables.

Note:- The data must first be sorted in the same order as the BY variable-list.

ID variable-list - When you use the ID statement, the observation numbers are not printed.

Instead, the variables in the ID variable list appear on the left-hand side of the page.

SUM variable-list - The SUM statement prints sums for the variables in the list.

VAR variable-list - The VAR statement specifies which variables to print and the order.

Without a VAR statement, all variables in the SAS data set are printed in the order that

they occur in the data set.

\*/

Data mylib.Customer;

informat Tranactiondate Date9.;

format Tranactiondate Date9.;

Input CustomerName $ CustomerID

Tranactiondate Sales Profit;

Cards;

Cust1 1 20Jan2009 100 20

Cust1 1 21Jan2010 200 80

Cust1 1 02Feb2009 170 28

Cust1 1 01Mar2010 500 85

Cust1 1 23Apr2013 170 29

Cust1 1 24Jan2012 260 56

Cust2 2 23Dec2009 340 78

Cust2 2 18Sep2010 780 90

Cust2 2 13Oct2009 100 30

Cust2 2 18Jun2011 250 40

Cust2 2 16Jul2012 800 98

Cust2 2 01Mar2013 300 80

;

run;

Proc Print Data=mylib.Customer;

run;

Proc Print Data=mylib.Customer;

var CustomerName Sales CustomerID;

run;

Proc Print Data=mylib.Customer;

var CustomerName Sales;

Id CustomerID;

run;

Proc Print Data=mylib.Customer;

By CustomerName;

var CustomerId Sales Profit;

run;

Proc Print Data=mylib.Customer;

By CustomerName;

var CustomerId Sales Profit;

sum sales Profit;

run;

/\*

Proc Sort - IDENTIFYING AND STORING UNIQUE AND DUPLICATE VALUES

Default sorting is ascending

PROC SORT Data = Input\_DataSet Out=Output\_DataSet;

BY variable\_list;

RUN;

BY variable-list - Variable names use for ordering your data

\*/

data mylib.Marine;

input NAME $ FAMILY $ length ;

Cards;

beluga whale 15

whale shark 40

basking shark 30

gray whale 50

mako shark 12

sperm whale 60

dwarf shark .

whale shark 40

humpback whale 50

blue whale 100

killer whale 30

;

run;

Proc Print Data=mylib.Marine;

run;

PROC SORT data = mylib.marine out =mylib.seasort;

BY length;

run;

PROC PRINT data = mylib.seasort;

run;

/\* Sorting in descending - Use Descensing Keyword before Variable name

PROC SORT Data = data-name Out = data-name NODUPKEY ;

BY variable-1 DESCENDING variable-2 DESCENDING variable-3;

RUN;

\*/

PROC SORT data = mylib.marine out = mylib.seasort;

BY descending length;

run;

PROC PRINT data = mylib.seasort;

run;

/\*In PROC SORT, there are two options by which we can remove duplicates.

1. NODUPKEY Option - The NODUPKEY option removes duplicate observations where value

of a variable listed in BY statement is repeated

2. NODUP Option - The NODUP option removes duplicate observations where values in

all the variables are repeated (identical observations).

\*/

data mylib.readin;

input ID Name $ Score;

cards;

1 David 45

1 David 74

2 Sam 45

2 Ram 54

3 Bane 87

3 Mary 92

3 Bane 87

4 Dane 23

5 Ken 87

5 Ken 87

6 Simran 63

8 Simran 72

;

run;

Proc Print Data=mylib.readin;

run;

PROC SORT data = mylib.readin out = mylib.readout nodupkey;

BY Name;

run;

PROC PRINT data = mylib.readout;

run;

PROC SORT data = mylib.readin out = mylib.readout nodup;

BY Name;

run;

PROC PRINT data = mylib.readout;

run;

\* STORING DUPLICATES - Use the DUPOUT= option with NODUPKEY (or NODUP) to output

duplicates to the specified SAS data set;

PROC SORT data = mylib.readin out = mylib.readout nodupkey dupout=mylib.readdup;

BY Name;

run;

PROC PRINT data = mylib.readdup;

run;

/\*

Proc Tabulate - Analysis data and show output in Table format.

PROC TABULATE Data=Input\_Dataset;

VAR analysis-variable-list;

CLASS classification-variable-list;

TABLE page-dimension, row-dimension, column-dimension;

Run;

Var - The VAR statement is used to list all analysis variables to be used within PROC TABULATE.

Class - The Class statement is used to list all Classification variables by which you want

to summarize or group your data.

Table - The Table statement is used to define how you want your table to appear; which

variables are in the Page, row, and column dimension.

When you have one variable in Table statement then it shows column dimension

When you have two variable in Table statement then it shows row and column dimension

When you have three variable in Table statement then it shows page, row and column dimension

\*/

data mylib.sales;

input storeid$ custid$ month sales;

cards;

s1 c1 1 100

s1 c1 1 260

s1 c1 2 150

s1 c1 2 300

s1 c1 2 250

s1 c2 1 160

s1 c2 1 290

s1 c2 1 190

s1 c2 2 100

s1 c2 2 289

s1 c2 2 280

s2 c1 1 190

s2 c1 1 290

s2 c1 2 140

s2 c1 2 130

s2 c1 2 220

s2 c2 1 180

s2 c2 1 110

s2 c2 1 530

s2 c2 2 120

s2 c3 1 210

s2 c3 1 .

;

run;

Proc Print data=mylib.sales;

run;

PROC TABULATE DATA = mylib.sales;

VAR sales;

TABLE sales;

RUN;

\* By default, it calculates SUM for variables.;

\* The asterisk \* is used to add statistical keywords. Suppose you want to calculate

COUNT for T1 variable;

PROC TABULATE DATA = mylib.sales;

VAR sales;

TABLE sales\*N;

RUN;

\* Suppose you want to calculate both COUNT and SUM for T1 variable;

PROC TABULATE DATA = mylib.sales;

VAR sales;

TABLE sales\*(N Sum);

RUN;

\* Use of class statement;

PROC TABULATE DATA = mylib.sales;

CLASS custid;

TABLE custid;

RUN;

PROC TABULATE DATA = mylib.sales;

CLASS custid month;

TABLE custid, month;

RUN;

PROC TABULATE DATA = mylib.sales;

CLASS storeid custid month;

TABLE storeid, custid, month;

RUN;

\* Adding Statistics with anlysis variable;

PROC TABULATE DATA = mylib.sales;

Var sales;

CLASS custid month;

TABLE custid, (Mean sum)\*month\*Sales;

RUN;

PROC TABULATE DATA = mylib.sales;

Var sales;

CLASS storeid custid month;

TABLE storeid, custid, Mean\*month\*Sales;

RUN;

\* Proc Tabulate Options - Format, Box, Misstext, Change variable Name;

PROC TABULATE DATA = mylib.sales Format=dollar9.2;

Var sales;

CLASS custid month;

TABLE custid, Mean\*month\*Sales / misstext='NoData';

RUN;

PROC TABULATE DATA = mylib.sales Format=dollar9.2;

Var sales;

CLASS custid month;

TABLE custid, Mean\*month\*Sales / box='Sales Analysis';

RUN;

PROC TABULATE DATA = mylib.sales Format=dollar9.2;

Var sales;

CLASS custid month;

TABLE custid='Customer ID', Mean='Average'\*month\*Sales;

RUN;

\* Save output in a dataset;

PROC TABULATE DATA = mylib.sales Format=dollar9.2 out= mylib.tabout;

Var sales;

CLASS custid month;

TABLE custid='Customer ID', Mean='Average'\*month\*Sales;

RUN;

Proc print data = mylib.tabout;

run;

/\* Proc Transpose - Changing observations to variables

Used to transpose SAS datasets (turning observations into variables or variables

into observations) / (Rows into Columns or Columns into Rows)

PROC TRANSPOSE DATA = Input-Dataset OUT = Output-Dataset;

BY variable-list;

ID variable;

VAR variable-list;

BY - Used if you have any grouping variables that you want to retain as variables.

Note:- The data must first be sorted in the same order as the BY variable-list.

ID - Names the variables whose formatted values will become new variable names.

In absence of an ID statement, the new variables will be named COL1, COL2, and so on

Var - Names the variables whose values you want to transpose

\*/

data mylib.store;

input storeid$ dayid$ sales;

cards;

s1 d1 100

s1 d2 200

s1 d3 100

s1 d4 300

s1 d5 500

s1 d6 100

s1 d7 200

s2 d1 100

s2 d2 200

s2 d3 100

s2 d4 300

s2 d5 500

s2 d6 100

s2 d7 200

;

run;

Proc Print Data=mylib.store;

run;

proc sort data=mylib.store;

by storeid;

run;

proc transpose data=mylib.store out=mylib.transposedata;

by storeid;

id dayid;

var sales;

run;

proc print data=mylib.transposedata;

run;

\* SAS : PROC RANK;

data mylib.temp;

input ID Gender $ Score;

cards;

1 M 33

2 M 94

3 M 66

4 M 46

5 F 92

6 F 95

7 F 18

8 F 11

;

run;

Proc print data=mylib.temp;

run;

proc rank data= mylib.temp out = mylib.result;

var Score;

ranks ranking;

run;

Proc print data=mylib.result;

run;

/\*

1) The OUT option is used to store output of the rank procedure.

2) The VAR option is used to specify numeric variable (s) for which you want to

calculate rank

3) The RANKS option tells SAS to name the rank variable

4) By default, it calculates rank in ascending order.

\*/

\* Reverse order of ranking (Descending) - The descending keyword tells SAS to

sort the data in descending order and assign rank to the variable accordingly.;

proc rank data= mylib.temp descending out = mylib.result;

var Score;

ranks ranking;

run;

Proc print data=mylib.result;

run;

\* Percentile Ranking (Quartile Rank) - Suppose you need to split the variable into

four parts, you can use the groups option in PROC RANK. It means you are telling

SAS to assign only 4 ranks to a variable.;

proc rank data= mylib.temp descending groups = 4 out = mylib.result;

var Score;

ranks ranking;

run;

Proc print data=mylib.result;

run;

\* GROUPS=4 for quartile ranks, and GROUPS=10 for decile ranks, GROUPS = 100 for

percentile ranks.;

\* Ranking within BY group - Use the by statement in proc rank. It is required to sort

the data before using by statement.;

proc sort data = mylib.temp;

by gender;

run;

Proc print data=mylib.temp;

run;

proc rank data= mylib.temp descending out = mylib.result;

var Score;

ranks ranking;

by Gender;

run;

Proc print data=mylib.result;

run;