Proc sort data=sashelp.class out=mylib.sortdata;

by sex;

run;

proc rank data = mylib.sortdata descending out=mylib.temp(where=(weight\_r <= 4));

by sex;

var weight;

ranks weight\_r;

run;

Proc print data = mylib.temp;

run;

\* Horizontal Bar Chart;

\* Bar chart is one of the most commonly used charts. It's used to show comparison between two or more categories.

\* In the program below, HBAR represents horizontal bar, type = defines the type of statistics

we need to compute. SUMVAR refers to summary variable. It does not imply SUM/ Addition.;

proc gchart data=mylib.temp;

hbar name / type=sum sumvar=weight;

run;

quit;

\* Vertical Bar Chart

\* The only difference between the previous graph and this graph would be to change the option HBAR to VBAR.;

proc gchart data=mylib.temp;

vbar name / type=sum sumvar=weight;

run;

quit;

\* Add Labels in the Vertical Bar Chart;

proc gchart data=mylib.temp;

vbar name / type=sum sumvar=weight outside=sum;

run;

\* Note : You can use INSIDE= option to show data label at the end of the bars.;

\* Sort Bars - Use DESCENDING / ASCENDING option to sort the bars by their length.;

proc gchart data=mylib.temp;

vbar name / type=sum sumvar=weight descending outside=sum;

run;

/\* Add / Remove Axes Title

There are three axes to format in SAS.

Midpoint (Use MAXIS option)

Group (Use GAXIS option)

Response (Use RAXIS option)

In the following code, we are removing label from the X Axis and adding custom label in Y axis.

The goptions htext=13pt htitle=15pt refers to the font size of the default text and chart title.

\*/

goptions htext=13pt htitle=15pt;

axis1 label=none;

axis2 label=('Weight in Pounds');

proc gchart data=mylib.temp;

vbar name / type=sum sumvar=weight descending maxis=axis1 raxis=axis2;

run;

quit;

\* Change Width of the Chart - You can use WIDTH= option to increase width of the chart.;

\* Mean Chart - The type=mean option is used to generate mean statistics of the two categories of variable SEX.;

goptions htext=13pt htitle=15pt;

axis1 label=none value=(f="Arial/Bold" "Female" "Male"); ;

axis2 label=(a=90 f="Arial/Bold" 'Mean Weight in Pounds') order=(0 to 150 by 25) minor=none offset=(0,0);

proc gchart data=mylib.temp;

vbar sex / width= 25 type=mean sumvar=weight descending maxis=axis1 raxis=axis2 outside=mean;

run;

quit;

/\* Options

Assign Value Labels 'Male' 'Female' to X-axis. It is done via axis1 value = option.

Change the orientation of Y-axis title with 'a=90' in axis2 label.

Change the scale of Y axis with 'order=(0 to 150 by 25)'.

minor=none turns off minor tick marks.

offset=(0,0) remove space between the last number in the axis and the frame.

\*/

/\* Change color of the Chart - Set the color of the chart with PATTERN statement. The outline color of the

bar can be changed with COUTLINE= option. We can also custom (hex) colors in the graph.

See the custom color used in the following graph.

\*/

pattern1 v=solid color=red;

proc gchart data=mylib.temp;

vbar sex / width= 25 type=mean sumvar=weight descending maxis=axis1 raxis=axis2 outside=mean coutline=gray;

run;

quit;

\* Set Multiple Colors - We can use multiple PATTERN statements with SUBGROUP= option to fill multiple

colors in the graphs.;

pattern1 value=solid color=pink;

pattern2 value=solid color=cx42C0FB;

proc gchart data=mylib.temp;

vbar sex / width= 25 type=mean sumvar=weight descending maxis=axis1 raxis=axis2 outside=mean

coutline=gray subgroup=sex;

run;

quit;

\* Percentage Chart - We need to show the percentage of male and female students in a graph

(including the count number).;

goptions htext=13pt htitle=15pt;

axis1 label=none value=(f="Arial/Bold" "Female" "Male"); ;

axis2 label=(a=90 f="Arial/Bold" 'Percentage of Students') minor=none offset=(0,0);

proc gchart data=mylib.temp;

vbar sex / width= 25 type=pct descending maxis=axis1 raxis=axis2 inside= freq inside=pct;

run;

quit;

\* To show both frequent and %, use INSIDE= and OUTSIDE= options.

\* Grouped Bar Chart - Use GROUP= option to create hierarchical / grouped bar chart in SAS.;

axis1 label=('MSRP') minor=none offset=(0,0);

axis2 label=none;

axis3 label=none offset=(7,7);

proc gchart data=sashelp.cars;

vbar type / discrete type=mean sumvar=msrp group=origin coutline=gray raxis=axis1 maxis=axis2

gaxis=axis3 noframe;

run;

quit;

\* Stacked Bar Chart - The DISCRETE option is used to show each data value in a separate bar.

The SUBGROUP= option tells SAS to fill different color in each of the groups.;

axis1 label=('MSRP') minor=none offset=(0,0);

axis2 label=none offset=(7,7);

proc gchart data=sashelp.cars;

vbar type / discrete type=mean sumvar=msrp subgroup=origin coutline=gray width=10 raxis=axis1

maxis=axis2 noframe;

run;

quit;

\* Histogram - Histogram is used to show distribution of continuous values in a graph. The LEVELS=4

implies we need to show only 4 bars in the histogram.;

goptions htext=10pt htitle=12pt;

proc gchart data=mylib.temp;

vbar weight / space=1 width=10 outside=freq levels=4 range;

run;

quit;

\* Scatterplot - Scatterplot is used to see the relationship between two variables. In the graph below,

we are looking for relationship between height and weight by male and female.;

\* PROC GPLOT is used to create scatter plot, box plot and line chart in SAS.;

symbol1 value=circle height=3 interpol=none color=blue;

symbol2 value=circle height=3 interpol=none color=red;

proc gplot data=mylib.temp;

plot height\*weight=sex;

run;

quit;

\* Box Plot - The following graph is a special case of Box plot in which we are displaying the

distribution of continuous variable (height) by a categorical variable (sex).;

symbol1 value=circle height=3 interpol=none color=blue;

axis1 minor=none offset=(0,0);

axis2 offset=(20,20);

proc gplot data=mylib.temp;

plot height\*sex=1 / vaxis=axis1 haxis=axis2 noframe;

run;

quit;

\* To show 25th, 50th and 75th percentile in the box-plot, use the code below. The only change in the

code is to modify the symbol statement.;

symbol1 interpol=boxt bwidth=4 color=blue;

axis1 minor=none offset=(0,0);

axis2 offset=(20,20);

proc gplot data=mylib.temp;

plot height\*sex=1 / vaxis=axis1 haxis=axis2 noframe;

run;

\* Line Chart - Line chart is used to show trends in data over some time periods.;

symbol1 value=none interpol=sm color=blue;

proc gplot data=sashelp.failure;

where cause ="Contamination" and Process = "Process A";

plot count\*day;

run;

quit;

\* The INTERPOL=SM in SYMBOL statement tells SAS to connect the dots/arrows and create a line.

/\* Export Data using Proc export;

PROC EXPORT DATA= SASDataSet OUTFILE="Path/fileanme" DBMS=identifier REPLACE;

delimiter = 'Specify-Delimiter';

RUN;

\*/

\* Export excel file;

proc export data=sashelp.cars dbms=xlsx outfile="/home/u1048896/sampletest1.xlsx" replace;

run;

\* Export csv file;

proc export data=sashelp.cars dbms=csv outfile="/home/u1048896/sampletest1.csv" replace;

run;

\* Export delimited file;

proc export data=sashelp.cars dbms=dlm outfile="/home/u1048896/sampletest1.txt" replace;

delimiter="|" ;

run;

/\* ODS - Output Delivery System

\* The output from a SAS program can be converted to more user friendly forms like .html or PDF.

This is done by using the ODS statement available in SAS. ODS stands for output delivery system.

It is mostly used to format the output data of a SAS program to nice reports which are good

to look at and understand. That also helps sharing the output with other platforms and

softwares. It can also combine the results from multiple PROC statements in one single file.

The basic syntax for using the ODS statement in SAS is −

ODS outputtype PATH path name FILE = Filename and Path STYLE = StyleName;

PROC some proc

;

ODS outputtype CLOSE;

a) PATH represents the statement used in case of HTML output. In other types of output

we include the path in the filename.

b) Style represents one of the in-built styles available in the SAS environment.

\*/

\* Creating HTML Output;

ODS HTML PATH = '/home/u1048896/' FILE = 'AIRLINE.html' STYLE = EGDefault;

Proc Print data = sashelp.airline(obs=10);

run;

ODS HTML CLOSE;

\* Creating PDF Output;

ODS PDF file = '/home/u1048896/AIRLINE.pdf';

Proc Print data = sashelp.airline(obs=10);

run;

ODS PDF CLOSE;

\* Creating TRF(Word) Output;

ODS RTF file = '/home/u1048896/AIRLINE.rtf';

Proc Print data = sashelp.airline(obs=10);

run;

ODS RTF CLOSE;

\* Graph in PDF;

ods pdf file="/home/u1048896/Chart.pdf";

proc gchart data=sashelp.class;

vbar age / type=percent;

run;

quit;

ods pdf close;

\* Creating Excel File;

/\* Example 1 - Create a new sheet for each unique value in the grouping variable (By Group) \*/

ods tagsets.excelxp file="/home/u1048896/xlexport.xls"

options(embedded\_titles="yes" autofilter="1-3" frozen\_headers="3" frozen\_rowheaders="1"

absolute\_column\_width="8.5,11,7,9,8,8" autofit\_height="yes" sheet\_interval="bygroup"

sheet\_label=" " suppress\_bylines="yes") style=normal;

proc print data=sashelp.shoes noobs;

title "Detail of Region #byval(region)";

by region;

run;

ods tagsets.excelxp close;

/\* The SHEET\_INTERVAL= option is used to define the interval in which to create new

worksheets. \*/

/\* Example 2 : Define names of sheets manually \*/

ods tagsets.excelxp file='/home/u1048896/multitable.xls' style=STATISTICAL

options(sheet\_name='Summary' skip\_space='1,0,0,0,1' EMBEDDED\_TITLES='yes' sheet\_interval='none');

Title " First File";

proc freq data = sashelp.class;

table sex;

run;

Title " Second File";

proc print data = sashelp.cars;

run;

ods tagsets.excelxp options(sheet\_name='FREQ' skip\_space='1,0,0,0,1' EMBEDDED\_TITLES='yes'

sheet\_interval='none');

Title " Third File";

proc freq data = sashelp.cars;

table make;

run;

ods tagsets.excelxp close;

/\* Example 3 : Apply Custom Format of Excel \*/

data temp;

pct= 0.75;

number= -45;

run;

ods tagsets.excelxp file="/home/u1048896/customxl.xls";

proc print data=temp noobs;

var pct;

var number / style(data)={tagattr="format:$#,##0\_);[Red]($#,##0)"};

format pct percent5.2;

run;

ods tagsets.excelxp close;

/\* ODS TAGSETS.EXCELXP does not support graphs (charts). From SAS 9.4, SAS added new ODS

called ODS EXCEL that supports both graphs and tables. \*/

/\* ODS EXCEL \*/

ods excel file="/home/u1048896/graph.xlsx" options(start\_at="B5" tab\_color="red"

absolute\_row\_height="15" embedded\_titles="yes");

proc print data=sashelp.orsales;

title "Sample title showing new features";

run;

ods excel close;

/\* ODS Excel- Graph \*/

ods excel file="/home/u1048896/graph\_2.xlsx";

proc gchart data=sashelp.class;

vbar age / type=percent;

run;

quit;

ods excel close;

\* SAS : VARIABLE NAME HAVING SPACES OR SPECIAL CHARACTERS;

data mylib.temp;

input var1;

cards;

1

2

;

run;

\* Rename the variable 'var1' to 'variable one';

options validvarname=any;

data mylib.temp2;

set mylib.temp;

rename var1 = 'variable one'n;

run;

/\* The option VALIDMEMNAME= EXTEND allows you to read or access dataset (table) whose name having spaces

or special characters. In addition, we also need to put name of variable in quotes followed by the letter n.\*/

options VALIDMEMNAME=EXTEND;

proc print data= 'price data'n;

run;

\* The YEARCUTOFF = system option eliminates this ambiguity by telling SAS the first year of a 100-year

span to be used by date informats and functions when SAS encounters a two-digit year. The default value

of YEARCUTOFF is 1920. In the default case, if SAS encounters a two-digit year in your program between

20 and 99, SAS assumes the date has a prefix of 19. And, if SAS encounters a two-digit year in your program

between 00 and 19, SAS assumes the date has a prefix of 20;

data mylib.yearoff;

informat varname ddmmyy8.;

format varname worddatx20.;

input varname;

cards;

12-01-99

13-02-01

01-12-60

;

run;

Proc print data= mylib.yearoff;

run;

options yearcutoff=1820;

data mylib.yearoff;

informat varname ddmmyy8.;

format varname worddatx20.;

input varname;

cards;

12-01-99

13-02-01

01-12-60

;

run;

Proc print data= mylib.yearoff;

run;

/\* SAS ARRAYS AND DO LOOP

Array array-name {number-of-elements} list-of-variables;

You can use [ ] or { } or ( ) for defining number of elements in the ARRAY statement.

Examples

1. ARRAY ABC[5] a b c d e;

In the example above, ABC is an array-name, 5 implies the number of variables in array

and "a b c d e" are the fields that make up the array.

2. ARRAY ABC[] a b c d e;

In the example above, SAS would automatically calculate the number of variables in array.

3. ARRAY ABC[] X1-X10;

Where the X1 variable contains the X1 value, X2 contains the X2 value, etc.

4. ARRAY ABC[] $ X1-X10;

If the variables are of character type then use $ sign before specifying list of variables.

\*/

data mylib.temp;

input x1 x2 x3 x4$ x5$;

cards;

1 2 3 AA BB

2 3 4 AB CC

3 4 5 AC DD

4 5 6 AD EE

5 6 7 AE FF

6 7 8 AF GG

;

run;

Proc Print data = mylib.temp;

run;

\* Example I : Numeric variables having value greater than 3 need to be replaced with missing value.;

data mylib.temp1;

set mylib.temp;

array nvars {3} x1-x3;

do i = 1 to 3;

if nvars{i} > 3 then nvars{i} =.;

end;

drop i

run;

Proc Print data = mylib.temp1;

run;

/\* The first time the loop processes, the value of count is 1; the second time, 2; and the third time, 3.

At the beginning of the fourth iteration, the value of count is 4, which is found to be greater than the

stop value of 3 so the loop stops. However, the value of i is now 4 and not 3, the last value before

it would be greater than 3 as the stop value. \*/

\* Improvised version of the above code ;

data mylib.temp1;

set mylib.temp;

array nvars (\*) \_numeric\_;

do i = 1 to dim(nvars);

if nvars{i} > 3 then nvars{i} =.;

end;

drop i;

run;

Proc Print data = mylib.temp1;

run;

\* a) The "\_numeric\_" is used to specify all the numeric variables.

b) The DIM function returns the number of elements of array (variables).;

\* Example: Extract first letter of all the character variables;

data mylib.temp1;

set mylib.temp;

array cvars (\*) \_character\_;

do i = 1 to dim(cvars);

cvars{i} = substr(cvars{i},1,1);

end;

drop i;

run;

Proc Print data = mylib.temp1;

run;

\* The "\_character\_" is used to specify all the character variables.

\* Example: Extract first letter and fill in the new character variables ;

data mylib.temp1;

set mylib.temp;

array cvars (\*) \_character\_;

array dvars (\*) $ x6 X7;

do i = 1 to dim(cvars);

dvars{i} = substr(cvars{i},1,1) ;

end;

drop i;

run;

Proc Print data = mylib.temp1;

run;

\* Example: Assign Initial Values in a SAS Array;

data mylib.temp1;

set mylib.temp;

array nvars (\*) \_numeric\_;

array pvars (\*) px1 px2 px3;

array pctinc {3} \_temporary\_ (1.1 , 1.2 ,1.3);

do i = 1 to dim(nvars);

pvars{i} = nvars{i}\*pctinc{i};

end;

drop i;

run;

Proc Print data = mylib.temp1;

run;

/\*a) In the above example, we are multiplying variables' values with different numbers.

b) When the key word \_TEMPORARY\_ is used in a ARRAY statement, data elements are created

but are not stored in the data file. \*/

\* Example: Calculate Percentage Growth;

data mylib.temp1;

set mylib.temp;

array nvars(\*) \_numeric\_;

array diff{2} \_temporary\_;

array percent{2};

do i = 1 to 2;

diff{i} = nvars{i+1} - nvars{i};

percent{i} = diff{i}/nvars{i} ;

end;

drop i;

run;

Proc Print data = mylib.temp1;

run;

/\* Using the OF Operator in a SAS Array

The following two codes are equivalent :

array gnp (\*) x y z;

sumgnp = sum(of gnp(\*));

OR

sumgnp = sum(x,y,z);

Calculate the mean

mean\_score = mean(of gnp(\*));

Calculate the minimum;

min\_score = min(of gnp(\*));

\*/

data mylib.temp1;

set mylib.temp;

array nvars (\*) x1-x3;

if sum(of nvars(\*)) > 10 then flag =1;

else flag=0;

run;

Proc Print data = mylib.temp1;

run;

\* DO OVER LOOP - The DO OVER loop is one of the most useful DO loops. It can be used with an array

when indexing of the array is not needed.;

data mylib.temp1;

set mylib.temp;

array nvars \_numeric\_;

do over nvars;

if nvars > 3 then nvars = .;

end;

run;

Proc Print data = mylib.temp1;

run;

\* Below code is alternative of above code;

data mylib.temp1;

set mylib.temp;

array nvars {\*} \_numeric\_;

do i = 1 to 3;

if nvars{i} > 3 then nvars{i} =.;

end;

run;

Proc Print data = mylib.temp1;

run;