**PROCEDURES EXAMPLES**

**PROC PRINT Procedure**

**/\***

This customized report

* selects variables to include in the report and controls their order
* selects observations to include in the report
* groups the selected observations by JobCode
* sums the salaries for each job code and for all job codes
* displays numeric data with commas and dollar signs.

**Var - Selects variables that appear in the report and determines their order.**

The ID and BY statements work together to produce this layout. The ID variable is listed only once for each BY group. The BY lines are suppressed. Instead, the value of the ID variable, JobCode, identifies each BY group.

**\*/**

**data** a;

input name $ sal;

cards;

gg 1000

ff 2000

;

**run**;

**proc** **print** data=a label;

label name='empname';

**run**;

**proc** **print** data=a noobs;

**by sal;**

**run**;

**proc** **print** data=a noobs;

**var sal name;**

**run**;

**proc** **print** data=a noobs;

**id sal name;**

**run**;

**data** a;

input name $ sal @@;

cards;

gg 1000 ff 5000

;

**run**;

**proc** **print** data=a label;

label name='empname';

**run**;

**data** qtr2;

input Month qtr SalesRep $ type $ Units AmountSold;

cards;

01 1 Hollingsworth Deluxe 260 49.50

01 1 Garcia Standard 41 30.97

01 1 Hollingsworth Deluxe 330 49.50

01 1 Jensen Standard 1110 30.97

01 1 Garcia Standard 715 30.97

01 1 Jensen Deluxe 675 49.50

02 1 Jensen Standard 45 30.97

02 1 Garcia Deluxe 10 49.50

12 4 Hollingsworth Deluxe 125 49.50

12 4 Jensen Standard 1254 30.97

12 4 Hollingsworth Deluxe 175 49.50

;

**run**;

**proc** **sort** data=qtr2;

by SalesRep;

**run**;

**proc** **print** data=qtr2;

var Units AmountSold;

where Units>**500** or AmountSold>**20000**;

format Units comma7. AmountSold dollar14.2;

sum Units AmountSold;

by SalesRep Month;

id SalesRep Month;

sumby SalesRep;

title1 'Sales Rep Quarterly Totals for Sales above 500 Units or $20,000';

**run**;

**data** empdata;

input IdNumber $ **1**-**4** LastName $ **9**-**19** FirstName $ **20**-**29**

City $ **30**-**42** State $ **43**-**44** /

Gender $ **1** JobCode $ **9**-**11** Salary **20**-**29** @**30** Birth date7.

@**43** Hired date7. HomePhone $ **54**-**65**;

format birth hired date7.;

datalines;

1919 Adams Gerald Stamford CT

M TA2 34376 15SEP48 07JUN75 203/781-1255

1653 Alexander Susan Bridgeport CT

F ME2 35108 18OCT52 12AUG78 203/675-7715

1407 Grant Daniel Mt. Vernon NY

M PT1 68096 26MAR57 21MAR78 914/468-1616

1114 Green Janice New York NY

F TA2 32928 21SEP57 30JUN75 212/588-1092

;

**run**;

options nodate pageno=**1** linesize=**64** pagesize=**60**;

**proc** **sort** data=empdata out=tempemp;

by jobcode gender;

**run**;

**proc** **print** data=tempemp split='\*';

id jobcode;

by jobcode;

var gender salary;

sum salary;

label jobcode='Job Code\*========'

gender='Gender\*======'

salary='Annual Salary\*=============';

format salary dollar11.2;

where jobcode contains 'FA' or jobcode contains 'ME';

title 'Expenses Incurred for';

title2 'Salaries for Flight Attendants and Mechanics';

**run**;

**PROC IMPORT**

/\*

The IMPORT procedure reads data from an external data source and writes it to a SAS data set.

Base SAS can import delimited files. In delimited files, a delimiter--such as a blank, comma,

or tab--separates columns of data values. If you license SAS/ACCESS Interface to PC Files,

additional external data sources can include such files as Microsoft Access Database, Excel files,

and Lotus spreadsheets.

**PROC IMPORT**

**DATAFILE="filename" | TABLE="tablename"**

**OUT=<libref.>SAS-data-set <(SAS-data-set-options)>**

**<DBMS=identifier><REPLACE> ; <data-source-statement(s);>**

**\*/**

**proc** **import** datafile="D:\test\test1\abc.xls" out=test2

dbms=excel

replace;

getnames=yes;

**run**;

**proc** **import**

datafile="D:\test\test1\abc.xls" out=test3

dbms=excel

replace ;

sheet=emp;

getnames=yes;

**run**;

**proc** **import** table="sample" out=test1

dbms=access ;

database="D:\test\test1\sam1.mdb";

**run**;

**proc** **import** datafile="D:\test\test1\kk.txt"

out=mydata

dbms=dlm

replace;

delimiter='&';

getnames=no;

**run**;

**proc** **print** data=mydata;

**run**;

**proc** **import** datafile="D:\test\test1\kk.txt"

out=mydata

dbms=dlm

replace;

delimiter='&';

getnames=YES;

**run**;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*hexadecimal representation of a tab is '09'x.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**proc** **import** datafile='E:\Office\test\tab.txt'

out=class

dbms=dlm

replace;

delimiter='09'x;

datarow=**5**;

**run**;

**proc** **import** datafile="D:\test\test1\abc.csv" out=test2

dbms=dlm

replace;

delimiter=',';

getnames=yes;

**run**;

**/\***

**proc** **import** out=test1

datatable="D:\test\test1\sam1.mdb"

dbms=access

replace;

SCANMEMO=YES;

USEDATE=NO;

SCANTIME=YES;

**run**;

\*/

**EXPORT Procedure**

/\* The EXPORT procedure reads data from a SAS data set and writes it to an external data source.

External data sources can include such files as Microsoft Access Database, Excel files, Lotus spreadsheets,

and delimited external files. In delimited files, a delimiter--such as a blank, comma, or tab--separates

columns of data values.

PROC EXPORT DATA=<libref.>SAS-data-set <(SAS-data-set-options)>

OUTFILE="filename" | OUTTABLE="tablename"

<DBMS=identifier> <REPLACE><LABEL>;

\*/

**proc** **export** data=test2 outfile="D:\test\test1\ample.xls"

dbms=excel replace;

**run**;

**proc** **export** data=sasuser.cust

outtable="customers"

dbms=access97

replace;

database="E:\Office\test\mydata.mdb";

**run**;

**proc** **export** data=sashelp.class

dbms=excelcs

outfile='c:\Myfiles\demo.xls';

sheet='Class';

server='sales';

port= **4632**;

version='2000';

**run**;

**proc** **export** data=test2 outfile="D:\test\test1\ample.xls"

dbms=excel replace;

sheet="sheet2";

**run**;

**PROC SORT Procedure**

/\*The SORT procedure orders SAS data set observations by the values of one or more character or numeric variables.

The SORT procedure either replaces the original data set or creates a new data set. PROC SORT produces only an output data set\*/

**data** account;

input Company $ **1**-**22** Debt **25**-**30** AccountNumber **33**-**36**

Town $ **39**-**51**;

datalines;

Paul's Pizza 83.00 1019 Apex

World Wide Electronics 119.95 1122 Garner

Strickland Industries 657.22 1675 Morrisville

Ice Cream Delight 299.98 2310 Holly Springs

Watson Tabor Travel 37.95 3131 Apex

Boyd & Sons Accounting 312.49 4762 Garner

Bob's Beds 119.95 4998 Morrisville

Tina's Pet Shop 37.95 5108 Apex

Elway Piano and Organ 65.79 5217 Garner

Peter's Auto Parts 65.79 7288 Apex

Deluxe Hardware 467.12 8941 Garner

Pauline's Antiques 302.05 9112 Morrisville

Apex Catering 37.95 9923 Apex

Paul's Pizza 83.00 1019 Apex

;

**proc** **sort** data=account out=bytown;

by town;

**run**;

**proc** **sort** data=account out=bytown1;

by town company;

**run**;

**proc** **sort** data=account out=bytown;

by descending town;

**run**;

**proc** **sort** data=account out=bytown1;

by \_all\_;

**run**;

**proc** **print** data=bytown;

var company town debt accountnumber;

title 'Customers with Past-Due Accounts';

title2 'Listed Alphabetically within Town';

**run**;

**proc** **sort** data=account out=towns nodupkey;

by town;

**run**;

**proc** **sort** data=account out=towns dupout=accdup nodupkey;

by town;

**run**;

**proc** **sort** data=account out=towns nodup;

by company;

**run**;

**proc** **sort** data=account out=towns noduprecs;

by company;

**run**;

**proc** **print** data=towns;

var town company debt accountnumber;

title 'Towns of Customers with Past-Due Accounts';

**run**;

**PROC PRINTTO Procedure**

/\*To route SAS log and procedure output directly to a required path\*/

**data** numbers;

input x y z;

datalines;

14.2 25.2 96.8

10.8 51.6 96.8

9.5 34.2 138.2

8.8 27.6 83.2

11.5 49.4 287.0

6.3 42.0 170.7

;

**proc** **printto** log='D:\test\a.log' ;

**run**;

**proc** **printto** print='D:\test\kkk.lst' new;/\*replace the file instead of appending to it\*/

**run**;

**proc** **print** data=numbers;

title 'Listing of NUMBERS Data Set';

**run**;

**proc** **printto**;

**run**;

**APPEND Procedure**

/\*The APPEND procedure adds the observations from one SAS data set to the end of another SAS data set.\*/

PROC APPEND BASE=base-SAS-data-set <DATA=SAS-data-set-to-append> <FORCE>; \*/

**data** a;

input name $ age ;

cards;

rr 45

uu 44

;

**run**;

**data** b;

input name $ age ;

cards;

ee 34

er 45

yu 23

;

**run**;

**data** c;

input name $ age salary ;

cards;

ee 34 1000

er 45 2000

yu 23 3000

;

**run**;

**proc** **append** base=a data=b ;

**run**;

**proc** **append** base=a data=c ;

**run**;

**proc** **append** base=a data=c force;

**run**;

**data** c;

set a;

set b;

**run**;

**data** d;

set a b;

**run**;

**CHART Procedure**

**data** shirts;

input Size $ @@;

datalines;

medium large

large large

large medium

medium small

small medium

medium large

small medium

large large

large small

medium medium

medium medium

medium large

small small

;

options nodate pageno=**1** linesize=**80** pagesize=**60**;

**proc** **chart** data=shirts;

hbar size / type=percent;

title ' Total Sales for Each Shirt Size';

**run**;

**proc** **chart** data=shirts;

hbar size ;

title ' of Total Sales for Each Shirt Size';

**run**;

/\*Example 2: Producing a Percentage Bar Chart\*/

/\*Example 3: Subdividing the Bars into Categories\*/

options nodate pageno=**1** linesize=**80** pagesize=**60**;

**data** piesales;

input Bakery $ Flavor $ Year Pies\_Sold;

datalines;

Samford apple 1995 234

Samford apple 1996 288

Samford blueberry 1995 103

Samford blueberry 1996 143

Samford cherry 1995 173

Samford cherry 1996 195

Samford rhubarb 1995 26

Samford rhubarb 1996 28

Oak apple 1995 319

Oak apple 1996 371

Oak blueberry 1995 174

Oak blueberry 1996 206

Oak cherry 1995 246

Oak cherry 1996 311

Oak rhubarb 1995 51

Oak rhubarb 1996 56

Clyde apple 1995 313

Clyde apple 1996 415

Clyde blueberry 1995 177

Clyde blueberry 1996 201

Clyde cherry 1995 250

Clyde cherry 1996 328

Clyde rhubarb 1995 60

Clyde rhubarb 1996 59

;

**proc** **chart** data=piesales;

vbar flavor / subgroup=bakery sumvar=pies\_sold;

title 'Pie Sales by Flavor Subdivided by Bakery Location';

**run**;

**PLOT Procedure**

**data** djia;

input Year @**7** HighDate date7. High @**24** LowDate date7. Low;

format highdate lowdate date7.;

datalines;

1954 31DEC54 404.39 11JAN54 279.87

1955 30DEC55 488.40 17JAN55 388.20

1956 06APR56 521.05 23JAN56 462.35

1957 12JUL57 520.77 22OCT57 419.79

1958 31DEC58 583.65 25FEB58 436.89

1959 31DEC59 679.36 09FEB59 574.46

1960 05JAN60 685.47 25OCT60 568.05

1961 13DEC61 734.91 03JAN61 610.25

1962 03JAN62 726.01 26JUN62 535.76

1963 18DEC63 767.21 02JAN63 646.79

1964 18NOV64 891.71 02JAN64 768.08

1965 31DEC65 969.26 28JUN65 840.59

1966 09FEB66 995.15 07OCT66 744.32

1967 25SEP67 943.08 03JAN67 786.41

1968 03DEC68 985.21 21MAR68 825.13

1969 14MAY69 968.85 17DEC69 769.93

1970 29DEC70 842.00 06MAY70 631.16

1971 28APR71 950.82 23NOV71 797.97

1972 11DEC72 1036.27 26JAN72 889.15

1973 11JAN73 1051.70 05DEC73 788.31

1974 13MAR74 891.66 06DEC74 577.60

1975 15JUL75 881.81 02JAN75 632.04

1976 21SEP76 1014.79 02JAN76 858.71

1977 03JAN77 999.75 02NOV77 800.85

1978 08SEP78 907.74 28FEB78 742.12

1979 05OCT79 897.61 07NOV79 796.67

1980 20NOV80 1000.17 21APR80 759.13

1981 27APR81 1024.05 25SEP81 824.01

1982 27DEC82 1070.55 12AUG82 776.92

1983 29NOV83 1287.20 03JAN83 1027.04

1984 06JAN84 1286.64 24JUL84 1086.57

1985 16DEC85 1553.10 04JAN85 1184.96

1986 02DEC86 1955.57 22JAN86 1502.29

1987 25AUG87 2722.42 19OCT87 1738.74

1988 21OCT88 2183.50 20JAN88 1879.14

1989 09OCT89 2791.41 03JAN89 2144.64

1990 16JUL90 2999.75 11OCT90 2365.10

1991 31DEC91 3168.83 09JAN91 2470.30

1992 01JUN92 3413.21 09OCT92 3136.58

1993 29DEC93 3794.33 20JAN93 3241.95

1994 31JAN94 3978.36 04APR94 3593.35

;

**proc** **plot** data=djia;

plot high\*year='\*';

title 'High Values of the Dow Jones Industrial Average';

title2 'from 1954 to 1994';

**run**;

options nodate pageno=**1** linesize=**64** pagesize=**30**;

**proc** **plot** data=djia;

plot high\*year='\*'

low\*year='o' / overlay box;

title 'Plot of Highs and Lows';

title2 'for the Dow Jones Industrial Average';

**run**;

options nodate pageno=**1** linesize=**120** pagesize=**60**;

**proc** **plot** data=djia vpercent=**50** hpercent=**50**;

plot high\*year='\*';

plot low\*year='o';

plot high\*year='\*' low\*year='o' / overlay box;

title 'Plots of the Dow Jones Industrial Average';

title2 'from 1954 to 1994';

**run**;

**CONTENTS Procedure**

/\*describes the structure of a SAS data set, including the name, type, and length of all variables in the data set.\*/

**data** a;

input name $ age;

cards;

kk 34

tt 23

tr 67

;

**run**;

**proc** **print** data=a;

**run**;

**proc** **contents** data=work.a;

**run**;

**proc** **contents** data=work.\_all\_;

**run**;

**COPY Procedure**

/\*

COPY OUT=libref <IN=libref> <MOVE>;

copies files from the procedure input library that is specified in the PROC DATASETS statement to the output library that is specified in the OUT= option. The IN= option specifies a different input library. The MOVE option deletes files from the input library after copying them to the output library.

You can use the following statements with the COPY statement:

EXCLUDE SAS-data-set;

specifies a SAS data set that you want to exclude from the copy process. Files that you do not list in this statement are copied to the output library.

SELECT SAS-data-set;

specifies a SAS data set that you want to copy to the output library.

DELETE SAS-data-set;

deletes only the SAS data set that you specify in this statement.

SAVE SAS-data-set;

deletes all members of the library except those that you specify in this statement.

\*/

libname aa "D:\test";

**data** aa.e;

input name $ age ;

cards;

rr 45

uu 44

;

**run**;

**proc** **copy** in=work out=aa;

**run**;

**proc** **copy** in=aa out =work;

select e;

**run**;

**DATASETS Procedure**

/\*

The DATASETS procedure is used to manage SAS datasets.

With this procedure, you can list, change, append, and repair

datasets and create and maintain indexes. Incorporated within

the DATASETS procedure are all of the capabilities of the

APPEND, CONTENTS, and COPY procedures. Procedure

commands execute with a RUN command or upon the entry

of a DATASETS command. The procedure remains active

until another procedure, dataset statement, or QUIT command

is executed.

\*/

libname mylib "D:\test";

**DATA** mylib.intial\_cl;

INPUT account credit\_limit;

DATALINES;

1002 2000

1003 4000

1004 3000

;

**DATA** mylib.new\_cl;

INPUT account credit\_limit;

DATALINES;

1002 3000

1004 5000

1005 2500

;

**proc** **datasets** library=mylib;

contents data=\_all\_;

**run**;

**proc** **datasets** library=mylib;

change new\_cl=brand\_new\_cl;

copy in=mylib out =work; select brand\_new\_cl;

**run**;

**proc** **datasets** library=mylib;

change new\_cl=brand\_new\_cl;

**run**;

**proc** **datasets** library=mylib;

change new\_cl=brand\_new\_cl;

copy in=mylib out =work; select brand\_new\_cl;

**run**;

**proc** **datasets** library=mylib;

modify intial\_cl;

rename account=EMP\_ACCOUNT;

**run**;

**proc** **datasets** library=mylib;

modify intial\_cl;

delete intial\_cl;

**run**;

/\*The KILL option deletes all members and member types from the library.\*/

**proc** **datasets** library=work kill;

**run**;

**PROC** **Delete** data = mylib.\_all\_;

**run**;

**FORMAT Procedure**

**data** staff;

input Name & $16. IdNumber $ Salary

Site $ HireDate date7.;

format hiredate date7.;

datalines;

Capalleti, Jimmy 2355 21163 BR1 30JAN79

Chen, Len 5889 20976 BR1 18JUN76

Davis, Brad 3878 19571 BR2 20MAR84

Leung, Brenda 4409 34321 BR2 18SEP74

Martinez, Maria 3985 49056 US2 10JAN93

Orfali, Philip 0740 50092 US2 16FEB83

Patel, Mary 2398 35182 BR3 02FEB90

Smith, Robert 5162 40100 BR5 15APR86

Sorrell, Joseph 4421 38760 US1 19JUN93

Zook, Carla 7385 22988 BR3 18DEC91

;

**proc** **format** library=work;

value $city 'BR1'='Birmingham UK'

'BR2'='Plymouth UK'

'BR3'='York UK'

'US1'='Denver USA'

'US2'='Miami USA'

other='INCORRECT CODE';

**run**;

**proc** **print** data=staff noobs label;

label salary='Salary in U.S. Dollars';

format salary uscurrency. site $city.;

title 'PROCLIB.STAFF with a Format for the Variables';

title2 'Salary and Site';

**run**;

/\*Example 3: Writing a Format for Dates Using a Standard SAS Format\*/

**proc** **format** library=work;

value benefit low-**'31DEC1979'd**=[worddate20.]

**'01JAN1980'd**-high=' \*\* Not Eligible \*\*';

**run**;

**proc** **print** data=staff noobs label;

label salary='Salary in U.S. Dollars';

format site $city. hiredate benefit.;

title 'PROCLIB.STAFF with a Format for the Variables';

title2 'Salary, Site, and HireDate';

**run**;

**proc** **format**;

picture aud low-<**0**='0,000,000,009.00'

(prefix='-AU$' mult=**100**)

**0**-high='0,000,00,009.00 '

(prefix='AU$' mult=**100**);

picture sfr low-<**0**='0,000,000,009.00'

(prefix='-SFr.' mult=**100**)

**0**-high='0,000,00,009.00 '

(prefix='-SFr.' mult=**100**);

picture bpd low-<**0**='0,000,000,009.00'

(prefix='-BPd.' mult=**100**)

**0**-high='0,000,00,009.00 '

(prefix='BPd.' mult=**100**);

**run**;

**data** currency;

input aud sfr bpd **12.2**;

datalines;

12345 12345 12345

0 0 0

-12345 -12345 -12345

;

**proc** **print** data=currency noobs;

var aud sfr bpd;

format aud aud. sfr sfr. bpd bpd.;

title 'Unique Currency Formats'

run;

**PROC** **FORMAT** LIBRARY=WORK CNTLOUT=psn;

**RUN**;

**PROC** **FORMAT** LIBRARY=WORK

CNTLOUT=psn(KEEP=FMTNAME START END LABEL);

**RUN**;

**PROC** **FORMAT** LIBRARY=WORK

CNTLOUT=psn

(KEEP=FMTNAME

START END

LABEL);

VALUE PERSON

**0**-**2**=íBabyí

**3**-**12**=íChildí

**13**-**19**=íTeení

**20**-**64**=íAdultí

**65**-**100**=íSeniorí

RUN;

libname proj1 'E:\Office\SAS\_Chakra\_Training\SAS Base\Sas\_practice\format';

options fmtsearch=(proj1.city);

**proc** **format** library=proj1;

value $city 'BR1'='Birmingham UK'

'BR2'='Plymouth UK'

'BR3'='York UK'

'US1'='Denver USA'

'US2'='Miami USA'

other='INCORRECT CODE';

**run**;

title1 Display Format names;

**proc** **catalog** cat=proj1.formats;

contents;

**run**;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Another Example\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

LIBNAME tutorial "'E:\Office\SAS\_Chakra\_Training\SAS Base\Sas\_practice\format";

**PROC** **FORMAT** LIBRARY=tutorial;

VALUE GENDER

**1** = "Male"

**2** = "Female";

VALUE YN

**1** = "Yes"

**2** = "No";

**RUN**;

LIBNAME tutorial "'E:\Office\SAS\_Chakra\_Training\SAS Base\Sas\_practice\format";

OPTIONS FMTSEARCH=(tutorial);

LIBNAME tutorial "'E:\Office\SAS\_Chakra\_Training\SAS Base\Sas\_practice\format";

LIBNAME school "'E:\Office\SAS\_Chakra\_Training\SAS Base\Sas\_practice\format/format1";

OPTIONS FMTSEARCH=(tutorial school work);

**FREQ Procedure**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The TABLES statement requests one-way to n-way frequency and crosstabulation tables and statistics for those tables.

If you omit the TABLES statement, PROC FREQ generates one-way frequency tables for all data set variables

that are not listed in the other statements

OUTEXPECT

includes expected cell frequencies in the OUT= data set for crosstabulation tables.

The variable EXPECTED contains the expected cell frequencies. The EXPECTED option has no effect for one-way tables.

SPARSE

When you specify the SPARSE and LIST options, PROC FREQ displays all combinations

of variable values in the table listing, including those with a frequency count of zero.

By default, without the SPARSE option, PROC FREQ does not display zero-frequency levels in LIST output.

When you use the SPARSE and OUT= options, PROC FREQ includes empty crosstabulation table cells

in the output data set. By default, PROC FREQ does not include zero-frequency table cells in the output data set.

Control Displayed Output

CONTENTS=

specifies the HTML contents link for crosstabulation tables

CROSSLIST

displays crosstabulation tables in ODS column format

FORMAT=

formats the frequencies in crosstabulation tables

LIST

displays two-way to n-way tables in list format

NOCOL

suppresses display of column percentages

NOCUM

suppresses display of cumulative frequencies and percentages

NOFREQ

suppresses display of frequencies

NOPERCENT

suppresses display of percentages

NOPRINT

suppresses display of crosstabulation tables but displays statistics

NOROW

suppresses display of row percentages

NOSPARSE

suppresses zero frequency levels in CROSSLIST, LIST and OUT=

NOWARN

suppresses log warning message for the chi-square test

PRINTKWT

displays kappa coefficient weights

SCOROUT

displays row and column scores

Produce Statistical Graphics

PLOTS=

requests plots from ODS Graphics

Create an Output Data Set

OUT=

names an output data set to contain frequency counts

OUTCUM

includes cumulative frequencies and percentages in the output data set for one-way tables

OUTEXPECT

includes expected frequencies in the output data set

OUTPCT

includes row, column, and two-way table percentages in the output data set

You can specify the following options in a TABLES statement.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**data** Color;

input Region Eyes $ Hair $ Count @@;

label Eyes ='Eye Color'

Hair ='Hair Color'

Region='Geographic Region';

datalines;

1 blue fair 23 1 blue red 7 1 blue medium 24

1 blue dark 11 1 green fair 19 1 green red 7

1 green medium 18 1 green dark 14 1 brown fair 34

1 brown red 5 1 brown medium 41 1 brown dark 40

1 brown black 3 2 blue fair 46 2 blue red 21

2 blue medium 44 2 blue dark 40 2 blue black 6

2 green fair 50 2 green red 31 2 green medium 37

2 green dark 23 2 brown fair 56 2 brown red 42

2 brown medium 53 2 brown dark 54 2 brown black 13

;

**proc** **freq** data=Color;

**run**;

**proc** **freq** data=Color;

tables Eyes Hair;

**run**;

**proc** **freq** data=Color;

weight Count;

tables Eyes Hair ;

**run**;

**proc** **freq** data=Color;

weight Count;

tables Eyes Hair / nocum;

**run**;

**proc** **freq** data=Color;

weight Count;

tables Eyes Hair Eyes\*Hair / out=FreqCnt outexpect sparse;

title1 'Eye and Hair Color of European Children';

**run**;

**proc** **print** data=FreqCnt ;

title2 'Output Data Set from PROC FREQ';

**run**;

**data** SummerSchool;

input Gender $ Internship $ Enrollment $ Count @@;

datalines;

boys yes yes 35 boys yes no 29

boys no yes 14 boys no no 27

girls yes yes 32 girls yes no 10

girls no yes 53 girls no no 23

;

**proc** **freq** data=SummerSchool ;

**run**;

**proc** **freq** data=SummerSchool order=data ;

tables Internship\*Enrollment / nocum chisq;

weight Count;

**run**;

/\*A chi-square test is used to examine the association between two categorical variables.

It can be used to test both extent of dependence and extent of independence between Variables.

SAS uses PROC FREQ along with the option chisq to determine the result of Chi-Square test.\*/

/\*This variable has six levels and we assign percentage to each level as per the design of the test.\*/

/\*Percentage Values in the TESTP statement represent the percentage of levels of the variable.\*/

**proc** **freq** data = sashelp.cars;

tables type

/chisq

testp = (**0.20** **0.12** **0.18** **0.10** **0.25** **0.15**);

**run**;

/\*Two Way chi-square

Two way Chi-Square test is used when we apply the tests to two variables of the dataset.

he result shows the tabular form of all combinations of these two variables.\*/

**proc** **freq** data = sashelp.cars;

tables type\*origin

/chisq

;

**run**;

**MEANS Procedure**

PROC MEANS

calculates descriptive statistics based on moments

estimates quantiles, which includes the median

calculates confidence limits for the mean

identifies extreme values

performs a t test.

By default, PROC MEANS displays output. You can also use the OUTPUT statement to store the statistics in a SAS data set.

PROC MEANS and PROC SUMMARY are very similar

specifies whether PROC SUMMARY displays the descriptive statistics.

By default, PROC SUMMARY produces no display output, but PROC MEANS does produce display output.

Compute descriptive statistics for variables --PROC MEANS

Calculate separate statistics for each BY group --BY

Identify variables whose values define subgroups for the analysis --CLASS

Identify a variable whose values represent the frequency of each observation --FREQ

Include additional identification variables in the output data set --ID

Create an output data set that contains specified statistics and identification variables --OUTPUT

Identify specific combinations of class variables to use to subdivide the data --TYPES

Identify the analysis variables and their order in the results --VAR

Specify the number of ways to make unique combinations of class variables --WAYS

Identify a variable whose values weight each observation in the statistical calculations --WEIGHT

MLF

enables PROC MEANS to use the primary and secondary format labels for a given range or overlapping ranges to create subgroup combinations when a multilabel format is assigned to a class variable.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Example 1: Computing Specific Descriptive Statistics\*/

**data** cake;

input LastName $ **1**-**12** Age **13**-**14** PresentScore **16**-**17**

TasteScore **19**-**20** Flavor $ **23**-**32** Layers **34** ;

datalines;

Orlando 27 93 80 Vanilla 1

Ramey 32 84 72 Rum 2

Goldston 46 68 75 Vanilla 1

Roe 38 79 73 Vanilla 2

Larsen 23 77 84 Chocolate .

Davis 51 86 91 Spice 3

Strickland 19 82 79 Chocolate 1

Nguyen 57 77 84 Vanilla .

Hildenbrand 33 81 83 Chocolate 1

Byron 62 72 87 Vanilla 2

Sanders 26 56 79 Chocolate 1

Jaeger 43 66 74 1

Davis 28 69 75 Chocolate 2

Conrad 69 85 94 Vanilla 1

Walters 55 67 72 Chocolate 2

Rossburger 28 78 81 Spice 2

Matthew 42 81 92 Chocolate 2

Becker 36 62 83 Spice 2

Anderson 27 87 85 Chocolate 1

Merritt 62 73 84 Chocolate 1

;

**proc** **means** data=cake;

\*var PresentScore TasteScore;

**run**;

**proc** **means** data=cake n mean min max range std;

var PresentScore TasteScore;

**run**;

**proc** **means** data=cake n mean max min range std;

var PresentScore TasteScore;

title 'Summary of Presentation and Taste Scores';

**run**;

**proc** **means** data=cake n mean max min range std fw=**8**;

var PresentScore TasteScore;

title 'Summary of Presentation and Taste Scores';

**run**;

**data** pets;

input Pet $ Gender $;

datalines;

dog m

dog f

dog f

dog f

cat m

cat m

cat f

;

**proc** **means** data=pets ;

class pet gender;

**run**;

**proc** **means** data=pets order=freq;

class pet gender;

**run**;

**proc** **means** data=pets order=data;

class pet gender;

**run**;

/\*Example 2: Computing Descriptive Statistics with Class Variables\*/

**data** grade;

input Name $ **1**-**8** Gender $ **11** Status $**13** Year $ **15**-**16**

Section $ **18** Score **20**-**21** FinalGrade **23**-**24**;

datalines;

Abbott F 2 97 A 90 87

Branford M 1 98 A 92 97

Crandell M 2 98 B 81 71

Dennison M 1 97 A 85 72

Edgar F 1 98 B 89 80

Faust M 1 97 B 78 73

Greeley F 2 97 A 82 91

Hart F 1 98 B 84 80

Isley M 2 97 A 88 86

Jasper M 1 97 B 91 93

;

**proc** **means** data=grade maxdec=**3**;

var Score;

class Status Year;

types () status\*year;

title 'Final Exam Grades for Student Status and Year of Graduation';

**run**;

/\*Example 3: Using the BY Statement with Class Variables\*/

**proc** **sort** data=Grade out=GradeBySection;

by section;

**run**;

**proc** **means** data=GradeBySection min max median;

by Section;

var Score;

class Status Year;

title1 'Final Exam Scores for Student Status and Year of Graduation';

title2 ' Within Each Section';

**run**;

/\*Example 4: Using a CLASSDATA= Data Set with Class Variables\*/

**data** caketype;

input Flavor $ **1**-**10** Layers **12**;

datalines;

Vanilla 1

Vanilla 2

Vanilla 3

Chocolate 1

Chocolate 2

Chocolate 3

;

**proc** **means** data=cake range median min max fw=**7** maxdec=**0**

classdata=caketype exclusive printalltypes;

var TasteScore;

class flavor layers;

title 'Taste Score For Number of Layers and Cake Flavor';

**run**;

/\*Example 5: Using Multilabel Value Formats with Class Variables\*/

**proc** **format**;

value $flvrfmt

'Chocolate'='Chocolate'

'Vanilla'='Vanilla'

'Rum','Spice'='Other Flavor';

value agefmt (multilabel)

**15** - **29**='below 30 years'

**30** - **50**='between 30 and 50'

**51** - high='over 50 years'

**15** - **19**='15 to 19'

**20** - **25**='20 to 25'

**25** - **39**='25 to 39'

**40** - **55**='40 to 55'

**56** - high='56 and above';

**run**;

**proc** **means** data=cake fw=**6** n min max median nonobs;

class flavor/order=freq;

class age /mlf order=fmt; /\*multilabel format\*/

types flavor flavor\*age;

var TasteScore;

format age agefmt. flavor $flvrfmt.;

title 'Taste Score for Cake Flavors and Participant''s Age';

**run**;

/\*Example 6: Using Preloaded Formats with Class \*/

options nodate pageno=**1** linesize=**80** pagesize=**64**;

**proc** **format**;

value layerfmt **1**='single layer'

**2**-**3**='multi-layer'

**.**='unknown';

value $flvrfmt (notsorted)

'Vanilla'='Vanilla'

'Orange','Lemon'='Citrus'

'Spice'='Spice'

'Rum','Mint','Almond'='Other Flavor';

**run**;

**proc** **means** data=cake fw=**7** completetypes missing nonobs;

class flavor layers/preloadfmt exclusive order=data;

ways **1** **2**;

var TasteScore;

format layers layerfmt. flavor $flvrfmt.;

title 'Taste Score For Number of Layers and Cake Flavors';

**run**;

**COMPARE Procedure**

PROC COMPARE  base= dsname

BY group varname

ID identifier varname

VAR varlist; (selected variables)

WITH varlist; (matching variables

RUN;

**Data** A B;

Do i=**1** To **10**;

Output;

End;

**Run**;

**Data** C;

Do i=**2** To **12**;

Output;

End;

**Run**;

**Proc** **Compare** Base=A Comp=B; **Run**;

**Proc** **Compare** Base=A Comp=C; **Run**;

**TRANSPOSE Procedure**

**data** score;

input Student $9. +**1** StudentID $ Section $ Test1 Test2 Final;

datalines;

Capalleti 0545 1 94 91 87

Dubose 1252 2 51 65 91

Engles 1167 1 95 97 97

Grant 1230 2 63 75 80

Krupski 2527 2 80 76 71

Lundsford 4860 1 92 40 86

McBane 0674 1 75 78 72

;

**proc** **transpose** data=score out=score\_transposed;

**run**;

**proc** **print** data=score\_transposed noobs;

title 'Student Test Scores in Variables';

**run**;

/\*Example 2: Naming Transposed Variables\*/

options nodate pageno=**1** linesize=**80** pagesize=**40**;

**proc** **transpose** data=score out=idnumber name=Test

prefix=sn ;

id student;

**run**;

**proc** **transpose** data=score out=idnumber name=Test prefix=sn;

by Student ;

id studentid;

**run**;

**proc** **print** data=idnumber noobs;

title 'Student Test Scores';

**run**;

/\*Example 3: Labeling Transposed Variables\*/

options nodate pageno=**1** linesize=**80** pagesize=**40**;

**proc** **transpose** data=score out=idlabel name=Test

prefix=sn;

id studentid;

idlabel student;

**run**;

**proc** **print** data=idlabel label noobs;

title 'Student Test Scores';

**run**;

/\*Example 4: Transposing BY Groups\*/

options nodate pageno=**1** linesize=**80** pagesize=**40**;

**data** fishdata;

infile datalines missover;

input Location & $10. Date date7.

Length1 Weight1 Length2 Weight2 Length3 Weight3

Length4 Weight4;

format date date7.;

datalines;

Cole Pond 2JUN95 31 .25 32 .3 32 .25 33 .3

Cole Pond 3JUL95 33 .32 34 .41 37 .48 32 .28

Cole Pond 4AUG95 29 .23 30 .25 34 .47 32 .3

Eagle Lake 2JUN95 32 .35 32 .25 33 .30

Eagle Lake 3JUL95 30 .20 36 .45

Eagle Lake 4AUG95 33 .30 33 .28 34 .42

;

**proc** **transpose** data=fishdata

out=fishlength(rename=(col1=Measurement));

var length1-length4;

by location date;

**run**;

**proc** **print** data=fishlength noobs;

title 'Fish Length Data for Each Location and Date';

**run**;

**data** stocks;

input Company $14. Date $ Time $ Price;

datalines;

Horizon Kites jun11 opening 29

Horizon Kites jun11 noon 27

Horizon Kites jun11 closing 27

Horizon Kites jun12 opening 27

Horizon Kites jun12 noon 28

Horizon Kites jun12 closing 30

SkyHi Kites jun11 opening 43

SkyHi Kites jun11 noon 43

SkyHi Kites jun11 closing 44

SkyHi Kites jun12 opening 44

SkyHi Kites jun12 noon 45

SkyHi Kites jun12 closing 45

;

**proc** **transpose** data=stocks out=close let; /\*LET transposes only the last observation for each BY group \*/

by company;

id date;

**run**;

**proc** **print** data=close noobs;

title 'Closing Prices for Horizon Kites and SkyHi Kites';

**run**;

**REPORT Procedure**

**data** grocery;

input Sector $ Manager $ Department $ Sales @@;

datalines;

se 1 np1 50 se 1 p1 100 se 1 np2 120 se 1 p2 80

se 2 np1 40 se 2 p1 300 se 2 np2 220 se 2 p2 70

nw 3 np1 60 nw 3 p1 600 nw 3 np2 420 nw 3 p2 30

nw 4 np1 45 nw 4 p1 250 nw 4 np2 230 nw 4 p2 73

nw 9 np1 45 nw 9 p1 205 nw 9 np2 420 nw 9 p2 76

sw 5 np1 53 sw 5 p1 130 sw 5 np2 120 sw 5 p2 50

sw 6 np1 40 sw 6 p1 350 sw 6 np2 225 sw 6 p2 80

ne 7 np1 90 ne 7 p1 190 ne 7 np2 420 ne 7 p2 86

ne 8 np1 200 ne 8 p1 300 ne 8 np2 420 ne 8 p2 125

;

**proc** **format** ;

value $sctrfmt 'se' = 'Southeast'

'ne' = 'Northeast'

'nw' = 'Northwest'

'sw' = 'Southwest';

value $mgrfmt '1' = 'Smith' '2' = 'Jones'

'3' = 'Reveiz' '4' = 'Brown'

'5' = 'Taylor' '6' = 'Adams'

'7' = 'Alomar' '8' = 'Andrews'

'9' = 'Pelfrey';

value $deptfmt 'np1' = 'Paper'

'np2' = 'Canned'

'p1' = 'Meat/Dairy'

'p2' = 'Produce';

**run**;

**proc** **report** data=grocery

colwidth=**10**

spacing=**5**

headline headskip;

column sector manager department sales;

define manager / order order=formatted format=$mgrfmt.;

define department / order order=internal format=$deptfmt.;

define sales / analysis sum format=dollar7.2;

break before manager / dul

summarize

skip;

format sector $sctrfmt. manager $mgrfmt. department $deptfmt.

sales dollar11.2;

**run**;

\*rbreak before / dol dul summarize; /\*ul, dul,ol\*/\*/

/\* where sector='se'; \*/

run;

options nodate pageno=**1** linesize=**64** pagesize=**60**;

/\*fmtsearch=(proclib);\*/

**proc** **report** data=grocery nowindows;

column sector manager sales;

define sector / group format=$sctrfmt.;

define sales / analysis sum

format=dollar9.2;

define manager / group format=$mgrfmt.;

break after sector / summarize skip ol;

rbreak after / summarize dol dul;

compute after;

sector='Total:';

endcomp;

**run**;

**proc** **report** data=grocery nowd;

column sector manager department sales;

rbreak after / dol summarize; /\*ul, dul,ol\*/

where sector='se';

format sector $sctrfmt. manager $mgrfmt. department $deptfmt.

sales dollar11.2;

title 'Sales for the Southeast Sector';

title2 "for &sysdate";

**run**;

/\*Example 2: Ordering the Rows in a Report\*/

**proc** **report** data=grocery nowd

colwidth=**10**

spacing=**5**

headline headskip;

column manager department sales;

define manager / order order=formatted format=$mgrfmt.;

define department / order order=internal format=$deptfmt.;

define sales / analysis sum format=dollar7.2;

break after manager / ol

summarize

skip;

compute after;

line '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*';

line 'Total sales for these stores were: '

sales.sum dollar9.2;

line '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*';

endcomp;

where sector='se';

title 'Sales for the Southeast Sector';

**run**;

ods html body='external-HTML-file';

ods pdf file='external-PDF-file';

ods rtf file='external-RTF-file';

**proc** **report** data=grocery nowd headline headskip

style(report)=[cellspacing=**5** borderwidth=**10** bordercolor=blue]

style(header)=[foreground=yellow

font\_style=italic font\_size=**6**]

style(column)=[foreground=moderate brown

font\_face=helvetica font\_size=**4**]

style(lines)=[foreground=white background=black

font\_style=italic font\_weight=bold font\_size=**5**]

style(summary)=[foreground=cx3e3d73 background=cxaeadd9

font\_face=helvetica font\_size=**3** just=r];

column manager department sales;

define manager / order

order=formatted

format=$mgrfmt.

'Manager';

define department / order

order=internal

format=$deptfmt.

'Department';

break after manager / summarize;

compute after manager;

line 'Subtotal for ' manager $mgrfmt. 'is '

sales.sum dollar7.2 '.';

endcomp;

compute after;

line 'Total for all departments is: '

sales.sum dollar7.2 '.';

endcomp;

where sector='se';

title 'Sales for the Southeast Sector';

**run**;

ods html close;

ods pdf close;

ods rtf close;

**TABULATE Procedure**

The TABULATE procedure displays descriptive statistics in tabular format, using some or all of the variables in a data set. You can create a variety of tables ranging from simple to highly customized.

PROC TABULATE computes many of the same statistics that are computed by other descriptive

statistical procedures such as MEANS, FREQ, and REPORT. PROC TABULATE provides

-simple but powerful methods to create tabular reports

-flexibility in classifying the values of variables and establishing hierarchical relationships between the variables

-mechanisms for labeling and formatting variables and procedure-generated statistics.

PROC TABULATE <option(s)>; BY <DESCENDING> variable-1

<...<DESCENDING> variable-n>

<NOTSORTED>;

CLASS variable(s) </ options>;

CLASSLEV variable(s) / STYLE=<style-element-name | PARENT> <[style-attribute-specification(s)] >;

FREQ variable;

KEYLABEL keyword-1='description-1'

<...keyword-n='description-n'>;

KEYWORD keyword(s) / STYLE=<style-element-name | PARENT> <[style-attribute-specification(s)] >;

TABLE <<page-expression,> row-expression,> column-expression</ table-option(s)>;

VAR analysis-variable(s)</ options>;

WEIGHT variable;

Display descriptive statistics in tabular format- PROC TABULATE

Create a separate table for each BY group -BY

Identify variables in the input data set as class variables- CLASS

Specify a style for class variable level value headings- CLASSLEV

Identify a variable in the input data set whose values represent the frequency of each observation-- FREQ

Specify a label for a keyword -KEYLABEL

Specify a style for keyword headings --KEYWORD

Describe the table to create -TABLE

Identify variables in the input data set as analysis variables -VAR

Identify a variable in the input data set whose values weight each observation in the statistical calculations -WEIGHT

RTS(row title space)= provides 25 characters per line for row headings.

RTSpace = Number of positions to allow for the row headings.

Box = Text and style for the empty box in the upper left corner.

Condense Print multiple pages to the same physical page.

NoContinued Suppress the continuation message

MissText If a cell is blank, this text will print instead

PrintMiss Print CLASS variable values, even if there is not data for them

(this only works if somewhere there is at least one observation

with that value.

Indent= Number of spaces to indent nested row headings.

Style=[options] Specify ODS style elements for various parts of the table.

Style= ODS style element definitions. Example might be to change the justification or the font.

Weight= specify another variable that will weight the values of the variable with the following exceptions: (0 or <**0** = counts observation in total number of observations,

blank= exclude observation entirely)

**data** energy;

length State $**2**;

input Region Division state $ Type Expenditures;

datalines;

1 1 ME 1 708

1 1 ME 2 379

1 1 NH 1 597

1 1 NH 2 301

1 1 VT 1 353

1 1 VT 2 188

1 1 MA 1 3264

1 1 MA 2 2498

1 1 RI 1 531

1 1 RI 2 358

1 1 CT 1 2024

1 1 CT 2 1405

1 2 NY 1 8786

1 2 NY 2 7825

1 2 NJ 1 4115

1 2 NJ 2 3558

1 2 PA 1 6478

1 2 PA 2 3695

4 3 MT 1 322

4 3 MT 2 232

4 3 ID 1 392

4 3 ID 2 298

4 3 WY 1 194

4 3 WY 2 184

4 3 CO 1 1215

4 3 CO 2 1173

4 3 NM 1 545

4 3 NM 2 578

4 3 AZ 1 1694

4 3 AZ 2 1448

4 3 UT 1 621

4 3 UT 2 438

4 3 NV 1 493

4 3 NV 2 378

4 4 WA 1 1680

4 4 WA 2 1122

4 4 OR 1 1014

4 4 OR 2 756

4 4 CA 1 10643

4 4 CA 2 10114

4 4 AK 1 349

4 4 AK 2 329

4 4 HI 1 273

4 4 HI 2 298

;

/\*One Dimention \*/

**proc** **tabulate** data=energy;

class region division ;

table region\*division;

**run**;

/\*Two Dimention \*/

**proc** **tabulate** data=energy;

class region division ;

table region,division;

**run**;

**proc** **format**;

value regfmt **1**='Northeast'

**2**='South'

**3**='Midwest'

**4**='West';

value divfmt **1**='New England'

**2**='Middle Atlantic'

**3**='Mountain'

**4**='Pacific';

value usetype **1**='Residential Customers'

**2**='Business Customers';

**run**;

options nodate pageno=**1** linesize=**80** pagesize=**60**;

**proc** **tabulate** data=energy format=dollar12.;

class region division type;

var expenditures;

table region\*division,

type\*expenditures

/ rts=**25**;

format region regfmt. division divfmt. type usetype.;

title 'Energy Expenditures for Each Region';

title2 '(millions of dollars)';

**run**;

ods html body=”pth\ccc.html”;

/\*ods pdf file='external-PDF-file';\*/

/\*ods rtf file='external-RTF-file';\*/

**proc** **tabulate** data=energy style=[font\_weight=bold];

class region division type / style=[just=center];

classlev region division type / style=[just=left];

var expenditures / style=[font\_size=**3**];

keyword all sum / style=[font\_width=wide];

keylabel all="Total" sum= "dgfh";

table (region all)\*(division all\*[style=[background=yellow]]),

(type all)\*(expenditures\*f=dollar10.) / style=[bordercolor=blue]

misstext=[label="Missing" style=[font\_weight=light]]

box=[label="Region by Division by Type"

style=[font\_style=italic]];

format region regfmt. division divfmt. type usetype.;

title 'Energy Expenditures';

title2 '(millions of dollars)';

**run**;

ods \_all\_ close;

SETINIT procedure

The **SETINIT Procedure** is an [undocumented procedure](http://www.sascommunity.org/wiki/Undocumented_procedure) that identifies which SAS components or products are installed and what the license expiration date is for the system and each component. It also gives the site name and number as well as the system birthday, operating system and any grace or warning period for license expiration.

Syntax

**PROC** **SETINIT** *<options>* **;**

**RUN** **;**

**Proc product\_status;**

**Run;**

For Base SAS Software ...

Custom version information: 9.4\_M3

Image version information: 9.04.01M3D041815

For SAS/STAT ...

Custom version information: 14.1

For SAS/GRAPH ...

Custom version information: 9.4\_M3

For SAS/ETS ...

Custom version information: 14.1

For SAS/FSP ...

Custom version information: 9.4\_M3

For SAS/OR ...

Custom version information: 14.1

For SAS/AF ...

Custom version information: 9.4\_M3

**PROC PWENCODE Statement**

|  |
| --- |
| **PROC PWENCODE** IN='*password*' <OUT=*fileref*> <METHOD=*encoding-method*>; |

|  |
| --- |
|  |
|  |

**IN='*password*'**

specifies the password to encode. The password can contain up to a maximum of 512 characters, which include alphanumeric characters, spaces, and special characters. If the password contains embedded single or double quotation marks, use the standard SAS rules for quoting character constants (see ["SAS Constants in Expressions"](http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/a000780334.htm) in ***SAS Language Reference: Concepts*** for details).

**OUT=*fileref***

specifies a fileref to which the output string is to be written. If the OUT= option is not specified, the output string is written to the SAS log.

**METHOD=*encoding-method***

specifies the encoding method. Here are the supported values for ***encoding-method***:

If the METHOD= option is omitted, the default encoding method, **sas002**, is used automatically.

**PROC** **PWENCODE** IN='pril123' ;

**run**;

filename pwfile

'external-filename'

proc pwencode in='mypass1' out=pwfile;

run;

filename pwfile 'external-filename';

options symbolgen;

data \_null\_;

infile pwfile truncover;

input line :$50.;

call symputx('dbpass',line);

run;

libname x odbc dsn=SQLServer user=testuser password="&dbpass";

PROC DELETE

PROC Delete **data** = Libref.DataName;

**PROC SQL**;

drop table Libref.DataName;

**quit**;

**PROC Datasets** library = Libref;

delete DataName;

**run**;

**DATA** \_Null\_;

length dd $8;

rc = filename(dd,cats(pathname('Libref'),'**\D**ataName.sas7bdat'));

rc = fdelete(dd);

put \_all\_;

**run**;

proc delete lib=MyLib data=MyFile (memtype=catalog);

run;

The GENNUM= option deletes all the historical versions for each of the data sets.

proc delete data=MyLib.A MyLib.B MyLib.C (gennum=all);

run;

**PROC CATALOG Procedure**

## Syntax

**PROC CATALOG** [CATALOG=<*libref*.>*catalog*](http://support.sas.com/documentation/cdl/en/proc/70377/HTML/default/p1il9pe2plu3bpn0zza8igp6xue4.htm#n1k9d01i6kgr71n1okkq0aquw4du)<[ENTRYTYPE=*entry-type*](http://support.sas.com/documentation/cdl/en/proc/70377/HTML/default/p1il9pe2plu3bpn0zza8igp6xue4.htm#p1pi776ihb5et8n1rqmz2o0duy0e)>   
<[FORCE](http://support.sas.com/documentation/cdl/en/proc/70377/HTML/default/p1il9pe2plu3bpn0zza8igp6xue4.htm#p0usmwz71bq2ehn1ar6vuucm4pdc)> <[KILL](http://support.sas.com/documentation/cdl/en/proc/70377/HTML/default/p1il9pe2plu3bpn0zza8igp6xue4.htm#n0sgpobb2b1hu6n129tjejbfsumd)>;

### Summary of Optional Arguments

##### [**ENTRYTYPE=*entry-type***](http://support.sas.com/documentation/cdl/en/proc/70377/HTML/default/p1il9pe2plu3bpn0zza8igp6xue4.htm#p1pi776ihb5et8n1rqmz2o0duy0e)

restricts processing of the current PROC CATALOG step to one entry type.

##### [**FORCE**](http://support.sas.com/documentation/cdl/en/proc/70377/HTML/default/p1il9pe2plu3bpn0zza8igp6xue4.htm#p0usmwz71bq2ehn1ar6vuucm4pdc)

forces statements to execute on a catalog that is opened by another resource environment.

##### [**KILL**](http://support.sas.com/documentation/cdl/en/proc/70377/HTML/default/p1il9pe2plu3bpn0zza8igp6xue4.htm#n0sgpobb2b1hu6n129tjejbfsumd)

deletes all entries in a SAS catalog.

**Assign a library reference to a SAS library.**The LIBNAME statement assigns the libref Perm to the SAS library that contains a permanent SAS catalog.

libname perm '*SAS-library*';

**Delete two entries from the Perm.Sample catalog.**

proc catalog cat=perm.sample;

delete credit.program credit.log;

run;

**Copy all entries in the Perm.Sample catalog to the Work.TCatAll catalog.**

copy out=tcatall;

run;

**Copy everything except three LOG entries and Passist.Slist from Perm.Sample to Work.TestCat.**The EXCLUDE statement specifies which entries not to copy. ET= specifies a default type. (ET=) specifies an exception to the default type.

copy out=testcat;

exclude test1 test2 test3 passist (et=slist) / et=log;

run;

**Move three LOG entries from Perm.Sample to Work.LogCat.** The SELECT statement specifies which entries to move. ET= restricts processing to LOG entries.

copy out=logcat move;

select test1 test2 test3 / et=log;

run;

**Copy five SAS/AF software entries from Perm.Sample to Perm.Finance.** The NOEDIT option protects these entries in Perm.Finance from further editing with PROC BUILD.

copy out=perm.finance noedit;

select loan.frame loan.help loan.keys loan.pmenu;

run;

**Copy two formats from Perm.Formats to Perm.Finance.**The IN= option enables you to copy from a different catalog than the one specified in the PROC CATALOG statement. Note the entry types for numeric and character formats: REVENUE.FORMAT is a numeric format and DEPT.FORMATC is a character format. The COPY and SELECT statements execute before the QUIT statement ends the PROC CATALOG step.

copy in=perm.formats out=perm.finance;

select revenue.format dept.formatc;

run;

quit;

**The SAS Log**

***Copying, Protecting, Removing, Deleting, and Processing Entries Using PROC CATALOG***

1 libname perm '*SAS-library*';

NOTE: Libref PERM was successfully assigned as follows:

Engine: V9

Physical Name: *SAS-library*\perm

2 proc catalog cat=perm.sample;

NOTE: Writing HTML Body file: sashtml.htm

3 delete credit.program credit.log;

4 run;

NOTE: Deleting entry CREDIT.PROGRAM in catalog PERM.SAMPLE.

NOTE: Deleting entry CREDIT.LOG in catalog PERM.SAMPLE.

5 copy out=tcatall;

6 run;

NOTE: Copying entry DEFAULT.FORM from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry FSLETTER.FORM from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry LOAN.FRAME from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry LOAN.HELP from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry BUILD.KEYS from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry LOAN.KEYS from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry TEST1.LOG from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry TEST2.LOG from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry TEST3.LOG from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry TEST1.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry TEST2.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry TEST3.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry LOAN.SCL from catalog PERM.SAMPLE to catalog WORK.TCATALL.

NOTE: Copying entry PASSIST.SLIST from catalog PERM.SAMPLE to catalog WORK.TCATALL.

7 copy out=testcat;

8 exclude test1 test2 test3 passist (et=slist) / et=log;

9 run;

NOTE: Copying entry DEFAULT.FORM from catalog PERM.SAMPLE to catalog WORK.TESTCAT.

NOTE: Copying entry FSLETTER.FORM from catalog PERM.SAMPLE to catalog WORK.TESTCAT.

NOTE: Copying entry LOAN.FRAME from catalog PERM.SAMPLE to catalog WORK.TESTCAT.

NOTE: Copying entry LOAN.HELP from catalog PERM.SAMPLE to catalog WORK.TESTCAT.

NOTE: Copying entry BUILD.KEYS from catalog PERM.SAMPLE to catalog WORK.TESTCAT.

NOTE: Copying entry LOAN.KEYS from catalog PERM.SAMPLE to catalog WORK.TESTCAT.

NOTE: Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog WORK.TESTCAT.

NOTE: Copying entry TEST1.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TESTCAT.

NOTE: Copying entry TEST2.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TESTCAT.

NOTE: Copying entry TEST3.PROGRAM from catalog PERM.SAMPLE to catalog WORK.TESTCAT.

NOTE: Copying entry LOAN.SCL from catalog PERM.SAMPLE to catalog WORK.TESTCAT.

10 copy out=logcat move;

11 select test1 test2 test3 / et=log;

12 run;

NOTE: Moving entry TEST1.LOG from catalog PERM.SAMPLE to catalog WORK.LOGCAT.

NOTE: Moving entry TEST2.LOG from catalog PERM.SAMPLE to catalog WORK.LOGCAT.

NOTE: Moving entry TEST3.LOG from catalog PERM.SAMPLE to catalog WORK.LOGCAT.

13 copy out=perm.finance noedit;

14 select loan.frame loan.help loan.keys loan.pmenu;

15 run;

NOTE: Copying entry LOAN.FRAME from catalog PERM.SAMPLE to catalog PERM.FINANCE.

NOTE: Copying entry LOAN.HELP from catalog PERM.SAMPLE to catalog PERM.FINANCE.

NOTE: Copying entry LOAN.KEYS from catalog PERM.SAMPLE to catalog PERM.FINANCE.

NOTE: Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog PERM.FINANCE.

16 copy in=perm.formats out=perm.finance;

17 select revenue.format dept.formatc;

18 quit;

NOTE: Copying entry REVENUE.FORMAT from catalog PERM.FORMATS to catalog PERM.FINANCE.

NOTE: Copying entry DEPT.FORMATC from catalog PERM.FORMATS to catalog PERM.FINANCE.