SAS Procedures

The basic syntax of PROC step

PROC <proc_name> **DATA**=<dataset_name> ***Options****;*

…

…

…

RUN;

IMPORTANT NOTES:

1. PROCs are used for operations on data sets.
2. Many PROCs do not modify data itself.

PROC CONTENTS

PROC CONTENTS DATA = ABC ***VARNUM***;

RUN;

IMPORTANT NOTES:

1. **PROC CONTENTS** is used to get a summary of the variables
2. It gives the variable Name, Position, Length, Type, Format, Label.
3. By default i.e. without the ***VARNUM*** option the variables will be listed in alphabetical order.
4. If the ***VARNUM*** option is specified, then the variables are listed in ascending order of their position.

*

PROC SORT

PROC SORT DATA = <dataset_name> *OPTIONS;*
BY *DESCENDING* key1 key2 …
RUN;

OPTIONS AVAILABLE:

1. ***OUT =*** <dataset_name> :A new dataset is created which is the sorted version of the original dataset. If ***OUT*** is not specified then the original dataset is overwritten. Use this option almost always.
2. ***NODUPKEY*** :Duplicate keys are removed.
3. ***NODUPREC*** :Duplicate records are removed.

IMPORTANT NOTE:

1. By default, the sorting is done in ascending order hence there is no need to specify ASCENDING option. There is no ASCENDING keyword.

*

Exercise

- Sort the ‘Card_data’ dataset by ascending ‘Card_Limit’ and descending ‘Card_Expiry_Date’

PROC MEANS—What does it do?

- Provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations.
- E.g. the Means procedure computes
  - descriptive statistics
  - quantiles, which includes the median
  - confidence limits for the mean
  - extreme values
  - performs a **t** test.

PROC MEANS

PROC MEANS <DATA=*SAS-data-set*><*statistic-keyword(s)*>
<*option(s)*> **MAXDEC=***n;*

VAR *variable(s);*

BY *variable(s);*

CLASS *variable(s)/ MISSING;*

OUTPUT OUT=*SAS-data-set <statistic-keyword=variable-name(s)>;*

RUN;

Example :

```
proc means data=clinic.diabetes maxdec= 2;
   var age height weight;
   class gender;
   output out=work.sum_gender
     mean=AvgAge AvgHeight AvgWeight
     min=MinAge MinHeight MinWeight;
run;
```

Computing Statistics Using PROC MEANS

Keyword	Description	Keyword	Description
CLM	Two-sided confidence limit for the mean	MEDIAN / P50	Median or 50th percentile
CSS	Corrected sum of squares	P1	1st percentile
CV	Coefficient of variation	P5	5th percentile
KURTOSIS / KURT	Kurtosis	P10	10th percentile
LCLM	One-sided confidence limit below the mean	Q1 / P25	Lower quartile or 25th percentile
MAX	Maximum value	Q3 / P75	Upper quartile or 75th percentile
MEAN	Average	P90	90th percentile
MIN	Minimum value	P95	95th percentile
N	Number of observations with non-missing values	P99	99th percentile
NMISS	Number of observations with missing values	QRANGE	Difference between upper and lower quartiles: Q3-Q1
RANGE	Range		
SKEWNESS / SKEW	Skewness	PROBT	Probability of a greater absolute value for the t value
STDDEV / STD	Standard deviation	T	Student's t for testing the hypothesis that the population mean is 0
STDERR / STDMEAN	Standard error of the mean		
SUM	Sum		
SUMWGT	Sum of the Weight variable values		
UCLM	One-sided confidence limit above the mean		
USS	Uncorrected sum of squares		
VAR	Variance		

PROC MEANS

- Limiting Decimal Places

  By default, PROC MEANS output uses the BEST. format to display values in the report. To limit decimal places, use the **MAXDEC=** option in the PROC MEANS statement, and set it equal to the length that you prefer.

  e.g. maxdec= 2;

- Specifying Variables in PROC MEANS

  By default, the MEANS procedure generates statistics for **every numeric variable** in a data set. To specify the variables that PROC MEANS analyzes, add a **VAR statement** and list the variable names.

  e.g. var age height weight;

- Group Processing Using the CLASS Statement

  You will often want statistics for grouped observations, instead of for observations as a whole. To produce separate analyses of grouped observations, add a **CLASS statement** to the MEANS procedure.

  e.g. class gender;

PROC MEANS

- **Creating a Summarized Data Set Using PROC MEANS**

  You might want to create an output SAS data set that contains only the summarized variable. You can do this by using the **OUTPUT statement** in PROC MEANS.

  e.g. output out=work.sum_gender

Points to Remember

- In PROC MEANS, use a VAR statement to limit output to relevant variables. Exclude statistics for nominal variables such as ID or ProductCode.
- By default, PROC MEANS prints the full width of each numeric variable. Use the MAXDEC= option to limit decimal places and to improve legibility.
- Data must be sorted for BY-group processing. You might need to run PROC SORT before using PROC MEANS with a BY statement.
- PROC MEANS and PROC SUMMARY produce the same results; however, the default output is different. PROC MEANS produces a report, whereas PROC SUMMARY produces an output data set.

PROC MEANS OUTPUT

- Default Output of PROC MEANS (if you do not give any statistics keywords after data = xxxxx option

Variable	N	Mean	Std Dev	Minimum	Maximum
chq_ret_count_6	2839	1.5656921	1.5501368	1.0000000	29.0000000
chq_ret_count_7	3327	1.6654644	1.6149726	1.0000000	26.0000000
chq_ret_count_8	3312	1.5800121	1.3038517	1.0000000	19.0000000
chq_ret_count_9	4059	1.7285046	1.6595982	1.0000000	27.0000000
chq_ret_count_10	4603	1.7119270	1.6150648	1.0000000	27.0000000
chq_ret_count_11	4449	1.6486851	1.4254476	1.0000000	31.0000000
chq_ret_count	13101	2.8585604	4.3050203	1.0000000	93.0000000

PROC MEANS output dataset

- E.g.

```
PROC MEANS DATA = pl.cust_card;
VAR card_amt_6 card_amt_7 card_amt_8 card_amt_9 card_amt_10 card_amt_11;
CLASS cust_type;
OUTPUT
        OUT = cust_means;
RUN;
```

- The dataset cust_means will be as follows:

Cust_Type	_TYPE_	_FREQ_	card_amt_6	card_amt_7	card_amt_8	card_amt_9	card_amt_10	card_amt_11
	0	16422	6459.032678	4021.393383	3806.996041	4961.314129	3603.109271	3038.086759
CAA	1	576	67791.42429	17221.54149	11137.4206	18038.97761	10124.53345	51713.81154
SBA	1	15846	3740.762808	3594.857985	3581.856859	4348.89143	3279.967128	1562.227286

Exercise

- Calculate the average age for males and females
- Calculate the 95th percentile transaction amount where transaction date is in the year 2010

PROC FREQ—What does it do?

- The FREQ procedure produces
  - 1-way to **n**-way frequency and
  - cross tabulation tables.

PROC FREQ

PROC FREQ <DATA=*SAS-data-set*>;

TABLES *variable(s)* ***/*** **LIST;**

RUN;

By default, PROC FREQ creates a one-way table with the **frequency**, **percent**, **cumulative frequency**, and **cumulative percent** of every value of all variables in a data set.

Variable	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Value	Number of observations with the value	Frequency of the value divided by the total number of observations	Sum of the frequency counts of the value and all other values listed above it in the table	Cumulative frequency of the value divided by the total number of observations

PROC FREQ

Creating Two-Way Tables

- The simplest cross-tabulation is a **two-way** table. To create a two-way table, join two variables with an asterisk (*****) in the TABLES statement of a PROC FREQ step.

TABLES *variable-1*variable-2 ;*

When cross-tabulations are specified, PROC FREQ produces tables with cells that contain

- cell frequency
- cell percentage of total frequency
- cell percentage of row frequency
- cell percentage of column frequency.

PROC FREQ

Creating *N*-Way Tables

```
tables sex*weight*height /norow nopercent;
```

- Determining the Table Layout
  - Height goes to column, Weight goes to rows, Sex goes to higher level say page
  - It can be generalized as Page x rows x Column

- Suppressing Table Information :

  To control the depth of cross-tabulation results, add a slash (/) and any combination of the following options to the TABLES statement:
  - **NOFREQ** suppresses cell frequencies.
  - **NOPERCENT** suppresses cell percentages
  - **NOROW** suppresses row percentages.
  - **NOCOL** suppresses column percentages.

Proc Freq single var output

```
PROC FREQ DATA=ABC;
TABLES chq_ret_count_6 /OUT=XYZ;
RUN;
```

The dataset XYZ will contain the following data

chq_ret_count_6	COUNT	PERCENT
.	177783	
1	2042	71.92673
2	487	17.15393
3	163	5.741458
4	54	1.902078
5	27	0.951039
6	24	0.845368
7	14	0.493131
8	5	0.176118
9	4	0.140895
10	6	0.211342
11	3	0.105671
12	4	0.140895
13	1	0.035224
14	1	0.035224
22	1	0.035224
24	1	0.035224
28	1	0.035224
29	1	0.035224

Proc Freq 2 var output

Output of PROC FREQ with cross tab

Frequency Percent Row Pct Col Pct	Table of NO_OF_ADD_ONS by npur6							
	npur6							
NO_OF_ADD_ONS	0	1	2	3	4	5	6	Total
NO	870 7.54 13.34 60.37	1420 12.31 21.77 57.19	1141 9.89 17.49 57.05	1032 8.95 15.82 58.04	811 7.03 12.43 54.25	638 5.53 9.78 53.30	611 5.30 9.37 53.46	6523 56.54
YES	571 4.95 11.39 39.63	1063 9.21 21.20 42.81	859 7.45 17.13 42.95	746 6.47 14.88 41.96	684 5.93 13.64 45.75	559 4.85 11.15 46.70	532 4.61 10.61 46.54	5014 43.46
Total	1441 12.49	2483 21.52	2000 17.34	1778 15.41	1495 12.96	1197 10.38	1143 9.91	11537 100.00

Exercise

- Create a two-way table of ‘Card_type’ and ‘Card_Limit’ using “Card_data” dataset

PROC UNIVARIATE-What does it do?

- Provides data summarization methods that produce
  - univariate statistics
  - information on the distribution of numeric variables.
- E.g. it computes
  - calculates descriptive statistics based on moments
  - calculates the median, mode, range, and quantiles
  - calculates confidence limits
  - tabulates extreme observations and extreme values
  - generates frequency tables
  - performs tests for location and normality
  - creates output data sets with requested statistics.

PROC UNIVARIATE-- Syntax

- **PROC UNIVARIATE DATA**=<dataset_name> ***OPTIONS***;
  CLASS variable1…….variablen/ ***OPTIONS***;
  VAR variable1…….variablen;
  OUTPUT OUT=<dataset_name> ***OPTIONS***;
  RUN;

Options in the output statement:

- All the descriptive statistics can be computed with the help of keywords, viz., “pctlpts”, “mean”, “median” etc.
- Output out =<dataset_name> pctlpre = <Any prefix> pctlpts = <1 to 100>

PROC RANK -- Syntax

- **PROC RANK DATA**=<dataset_name> ***OPTIONS***;

  VAR <dataset variables>;

  RANKS <new variables>;

  By <dataset variable>; (Requires sorting)

  RUN;

Options:

By default the smallest value will receive the rank of 1.

- DESCENDING: Reverses the direction of the ranks.
  - The largest value receives a rank of 1, the next largest value receives a rank of 2, and so on.
  - Otherwise, values are ranked from smallest to largest.
- GROUPS: assigns group values ranging from 0 to number-of-groups minus 1
  - Common specifications are
    - GROUPS=100 for percentiles
    - GROUPS=10 for deciles
    - GROUPS=4 for quartiles.

PROC RANK—Sample Output

- ```
  proc rank data=golf out=rankings;
  var strokes;
  ranks Finish;
  run;

  proc print data=rankings;
  run;
  ```

Assignment of the Lowest Rank Value to the Lowest Variable Value

```
                        The SAS System                          1

                OBS    Player    Strokes    Finish

                 1     Jack        279         2
                 2     Jerry       283         3
                 3     Mike        274         1
                 4     Randy       296         4
                 5     Tito        302         5
```

IMP: **1.** If one omits the OUT= statement, the data set is named using the “DATAn” naming convention.

2. Missing values are not ranked
```

Exercise

- Rank the transaction amount at “Cust_id” level in “Transaction_data” dataset

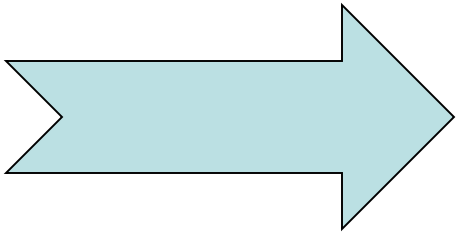
PROC TRANSPOSE

PROC TRANSPOSE DATA=<dataset_name> *OUT*=<dataset_name>;
VAR x1 x2 x3;
RUN;

ABC

x1	x2	X3
1	2	3
4	5	6
7	8	9

PROC TRANSPOSE
DATA=ABC *OUT*=XYZ; RUN;



XYZ

NAME	COL1	COL2	COL3
x1	1	4	7
x2	2	5	8
x3	3	6	9

Note

By default, PROC TRANSPOSE works only on num variables. For character variables use VAR _ALL_

To use values of x1 as column names use ID x1; before RUN;

PROC FORMAT

```
proc format;
  value popfmt
    (low) 0 - 300 = 'Below 300'
    301 - 500 = 'Between 301 and 500'
    500 - high = 'Over 500‘;
run;
```

PROC TABULATE

```
proc tabulate data = data1 ;
class       classification-variable-list ;
var         analysis-variable-list ;
Table       page-dimension  ,
            row dimension   ,
            column-dimension ;
run ;
```

Variable Name	Type	Values taken
REGION	Char	NC, NE, SO, WE
CITYSIZE	Char	L, M, S
POP	Num	
PRODUCT	Char	A100, A200, A300
SALETYPE	Char	R, W
Quantity	Num	
Amount	Num	

PROC TABULATE

```
PROC TABULATE data = data1 F=12.     F=dollar12. ;
CLASS REGION CITYSIZE PRODUCT          ;
var pop ;
TABLE REGION,                          / misstext = "Missing" ;
                citysize ,
                PRODUCT * pop ;
run ;
```

REGION NC	PRODUCT		
	A100	A200	A300
	POP	POP	POP
	Sum	Sum	Sum
CITYSIZE			
L	**$1,250,000**	**$1,250,000**	**$1,350,000**
M	**$250,000**	**$250,000**	**Missing**
S	**$110,000**	**$50,000**	**$52,000**

REGION NC	PRODUCT		
	A100	A200	A300
	POP	POP	POP
	Sum	Sum	Sum
CITYSIZE			
L	**1250000**	**1250000**	**1350000**
M	**250000**	**250000**	**.**
S	**110000**	**50000**	**52000**

REGION NE	PRODUCT		
	A100	A200	A300
	POP	POP	POP
	Sum	Sum	Sum
CITYSIZE			
L	**1674000**	**1674000**	**1674000**
M	**274000**	**492000**	**474000**
S	**75000**	**75000**	**74000**