Reporting and Tabulation
Module - 6

Learn to Evolve

Proc Report

Enables to

- Create custom reports
- Request separate subtotals and grand totals
- Calculate columns
- Create and store report definitions.

Syntax

PROC REPORT <DATA=*SAS-data-set*> <*options*>*;*
RUN;

where

- *SAS-data-set* is the name of the SAS data set that is used for the report

options includes

- **WD**, which invokes the procedure in a windowing mode. Your report appears in the REPORT window. This is the default for the SAS windowing environment.
- **NOWD**, which displays a listing of the report in the OUTPUT window.
- **HEADLINE**, which underlines all column headings and the spaces between them (This option have no effect on HTML output)
- **HEADSKIP**, which writes a blank line beneath all column headings or after the underline (This option have no effect on HTML output)

Example

- In this program, PROC REPORT reads the Flights.Europe data set and creates a report in a nonwindowing mode.

```
proc report data=flights.europe nowd;
  run;
```

- all observations and variables in the data set are printed
- Variables appear in the order in which they occur in the data set.

Selecting Variables

- To select and order the variables that appear in your list report, you can use the COLUMN statement.

Syntax

PROC REPORT <DATA=*SAS-data-set*> <*options*>*;*
 COLUMN *variable(s);*
RUN;

- *variable(s)* is one or more variable names, separated by blanks.

Example

- proc report data=flights.europe nowd;
  column flight orig dest mail freight revenue;
  run;

Flight	Orig	Dest	Mail	Freight	Revenue
821	LGA	LON	403	209	150634
271	LGA	PAR	492	308	156804
271	LGA	PAR	366	498	190098
821	LGA	LON	345	243	150634
821	LGA	LON	248	307	193930
271	LGA	PAR	353	205	166470
821	LGA	LON	391	395	167772
271	LGA	PAR	366	279	163248
821	LGA	LON	219	368	183106
271	LGA	PAR	357	282	170766
821	LGA	LON	389	479	169576
271	LGA	PAR	415	463	195468

Selecting Observations

- Use the WHERE statement to select observations based on conditions

Syntax

PROC REPORT <DATA=*SAS-data-set*> <*options*>*;*
 COLUMN *variable(s);*
 where *condition ;*
RUN;

- *variable(s)* is one or more variable names, separated by blanks.

Example

```
proc report data=flights.europe nowd;
 column flight orig dest mail freight revenue;
 where dest in ('LON','PAR');
run;
```

Flight	Orig	Dest	Mail	Freight	Revenue
821	LGA	LON	403	209	150634
271	LGA	PAR	492	308	156804
271	LGA	PAR	366	498	190098
821	LGA	LON	345	243	150634
821	LGA	LON	248	307	193930
271	LGA	PAR	353	205	166470
821	LGA	LON	391	395	167772

DEFINE Statements

- Describes how to use and display variables in the report
- Can use one or more DEFINE statements
- Can list options (usages, attributes, and so on) in any order
- If PROC REPORT can't create groups, it displays group variables as order variables.

Syntax

PROC REPORT <DATA=*SAS-data-set*> <*options*>;
  DEFINE *variable* / <*usage*> <*attribute(s)*> <*option(s)*>
  <*justification*> <*'column-heading'*> ;
RUN;

Where,

- *variable* is the name of the variable that you want to define.
- *usage* specifies how to use the variable. Valid options are ACROSS, ANALYSIS, COMPUTED, DISPLAY, GROUP and ORDER.
- *attribute(s)* specifies attributes for the variable, including FORMAT=, WIDTH=, and SPACING=.
- *option(s)* specifies formatting options, including DESCENDING, NOPRINT, NOZERO, and PAGE.
- <*justification*> specifies column justification (CENTER, LEFT, or RIGHT).
- *'column-heading'* specifies a label for the column heading.

Example

```
proc report data=flights.europe nowd;
   where dest in ('LON','PAR');
   column flight orig dest mail freight revenue;
   define flight/order descending 'Flight Number'   center width=6 spacing=5;
   define orig/'Flight Origin' center width=6;
run;
```

Defining Column Attributes

- Easy to change the appearance of the output by specifying attributes for variables
- Can select a format for data values
- Can specify the column width
- Specify the spacing between columns

Example

```
proc report data=flights.europe nowd;
     where dest in ('LON','PAR');
     column flight orig dest mail freight revenue;
     define revenue/format=dollar15.2;
run;
```

Partial PROC REPORT Output, HTML

Flight	Orig	Dest	Mail	Freight	Revenue
821	LGA	LON	403	209	$150,634.00
271	LGA	PAR	492	308	$156,804.00
271	LGA	PAR	366	498	$190,098.00
821	LGA	LON	345	243	$150,634.00
821	LGA	LON	248	307	$193,930.00

Example : Specifying Column Widths

- To specify a width for columns, use the WIDTH= attribute in the DEFINE statement.
- The WIDTH= attribute has no effect on HTML output.

```
proc report data=flights.europe nowd;
    where dest in ('LON','PAR');
    column flight orig dest mail freight revenue;
    define revenue / format=dollar15.2;
    define flight/width=6;
    define orig/width=4;
    define dest/width=4;
run;
```

Example : Specifying Column Spacing

- Specify the number of blank characters to be put between a selected column and the column immediately to its left.
- Default column spacing is 2
- To specify a different column spacing, use the SPACING= attribute in the DEFINE statement.
- The SPACING= attribute has no effect on HTML output.

```
proc report data=flights.europe nowd;
      where dest in ('LON','PAR');
      column flight orig dest mail freight revenue;
      define revenue / format=dollar15.2;
      define flight / width=6;
      define orig / width=4 spacing=5;
      define dest / width=4 spacing=5;
 run;
```

Example : Defining Column Headings

- To define a column heading, enclose the heading text in quotation marks in the DEFINE statement.

```
proc report data=flights.europe nowd;
    where dest in ('LON','PAR');
    column flight orig dest mail freight revenue;
    define revenue / format=dollar15.2;
    define flight / width=13'Flight Number';
    define orig / width=13 spacing=5 'Flight Origin';
    define dest / width=18 spacing=5 'Flight Destination';
run;
```

Example : Splitting Column Headings across Multiple Lines

To use a split character:

- Use the default slash (/) as the split character.
- Define a split character by using the SPLIT= option in the PROC REPORT statement

```
proc report data=flights.europe nowd;
     where dest in ('LON','PAR');
     column flight orig dest mail freight revenue;
     define revenue / format=dollar15.2;
     define flight / width=6 'Flight/Number';
    define orig / width=6 spacing=5 'Flight/Origin';
    define dest / width=11 spacing=5 'Flight/Destination';
run;

proc report data=flights.europe nowd split='*';
    where dest in ('LON','PAR');
    column flight orig dest mail freight revenue;
   define revenue / format=dollar15.2;
   define flight / width=6 'Flight*Number';
   define orig / width=6 spacing=5 'Flight*Origin';
   define dest / width=11 spacing=5 'Flight*Destination';
run;
```

Using Order Variables

- An order variable orders the detail rows in a report

```
proc report data=flights.europe nowd headline
headskip;
  where dest in ('LON','PAR');
 column flight orig dest mail freight revenue;
 define revenue / format=dollar15.2;
 define flight / order 'Flight/Number' width=6
center;
 define orig / width=6 spacing=5 'Flight/Origin'
center;
 define dest / width=11 spacing=5
'Flight/Destination'  center;
run;
```

Flight Number	Flight Origin	Flight Destination	Mail	Freigt	Revenue
219	LGA	LON	331	376	$189,065.00
	LGA	LON	485	267	$197,456.00
	LGA	LON	388	298	$162,343.00
	LGA	LON	421	356	$134,520.00
	LGA	LON	447	299	$106,753.00
	LGA	LON	356	547	$122,766.00
	LGA	LON	272	370	$198,744.00
271	LGA	PAR	492	308	$156,804.00
	LGA	PAR	366	498	$190,098.00
	LGA	PAR	353	205	$166,470.00
	LGA	PAR	366	279	$163,248.00
	LGA	PAR	357	282	$170,766.00
	LGA	PAR	415	463	$195,468.00
	LGA	PAR	352	351	$123,456.00
	LGA	PAR	492	308	$125,632.00
	LGA	PAR	366	498	$128,972.00

Example : Using Group Variables

- Can define one or more **group** variables, to summarize your data using PROC REPORT.
- Specify the **GROUP** usage option in the DEFINE statement.

```
Proc report data=flights.europe nowd headline headskip;
      where dest in ('LON','PAR');
      column flight orig dest mail freight revenue;
      define revenue / format=dollar15.2;
      define flight / group 'Flight/Number' width=6 center;
      define orig / group width=6 spacing=5 'Flight/Origin' center;
      define dest / group width=11 spacing=5 'Flight/Destination' center;
run;
```

Flight Number	Flight Origin	Flight Destination	Mail	Freight	Revenue
219	LGA	LON	2700	2513	$1,111,647.00
271	LGA	PAR	5050	4421	$1,969,201.00
821	LGA	LON	4438	4284	$2,077,907.00

Example : Specifying Statistics

- Can display the average revenue for each flight number, by specifying MEAN in the DEFINE statement for Revenue

```
proc report data=flights.europe nowd headline headskip;
    where dest in ('LON','PAR');
    column flight orig dest mail freight revenue;
    define revenue / mean format=dollar15.2 'Average/Revenue';
    define flight / group 'Flight/Number' width=6 center;
    define orig / group width=6 spacing=5 'Flight/Origin'  center;
    define dest / group width=11 spacing=5  'Flight/Destination' center;
run;
```

Flight Number	Flight Origin	Flight Destination	Mail	Freight	Average Revenue
219	LGA	LON	2700	2513	$158,806.71
271	LGA	PAR	5050	4421	$151,477.00
821	LGA	LON	4438	4284	$159,839.00

Using Across Variables

- Can also define variables as across variables, which are functionally similar to group variables.
- Displays the groups that it creates for an across variable horizontally rather than vertically.

```
proc report data=flights.europe nowd headline headskip;
  where dest in ('LON','PAR');
     column flight dest mail freight revenue;
     define revenue / format=dollar15.2;
     define flight / across 'Flight/Number' width=6 center;
     define dest / across width=11 spacing=5  'Flight/Destination' center;
run;
```

Flight Number			Flight Destination				
219	271	821	LON	PAR	Mail	Freight	Revenue
7	13	13	20	13	12188	11218	$5,158,755.00

Using Computed Variables

- Computed variables are numeric or character variables defined for the report, but not in the input data set
- Doesn't add them to the input data set
- Include the computed variable in the COLUMN statement.
- Define the variable's usage as COMPUTED in the DEFINE statement.
- Compute the value of the variable in a compute block that is associated with the variable.

```
proc report data=flights.europe nowd;
   where dest in ('LON','PAR');
   column flight capacity deplaned emptyseats;
   define flight / width=6;
   define emptyseats/computed 'Empty Seats';
   Compute emptyseats;
      emptyseats=capacity.sum-deplaned.sum;
   endcomp;
run;
```

Flight	Capacity	Deplaned	Empty Seats
821	250	222	28
271	250	163	87
271	250	227	23
821	250	222	28
821	250	158	92

HEADLINE and HEADSKIP Options

- The HEADLINE option underlines all column headers and the spaces between them at the top of each page of the report.
- The HEADSKIP option writes a blank line beneath all column headers (or beneath the underlining that the HEADLINE option writes) at the top of each page of the report.

```
proc report data=sashelp.shoes nowd headline headskip ;
run;
```

Without HEADLINE and HEADSKIP

Partial Output

Shoe Report

Province	Shoes	Total Inventory	Total Sales
Calgary	Men's Dress	39,663	12,775
	Women's Dress	54,677	12,601
Montreal	Men's Casual	187,155	53,929

With HEADLINE and HEADSKIP

Partial Output

Shoe Report

Province	Shoes	Total Inventory	Total Sales
Calgary	Men's Dress	39,663	12,775
	Women's Dress	54,677	12,601
Montreal	Men's Casual	187,155	53,929

RBREAK Statement

- The RBREAK *statement* produces a default summary at the beginning or end of a report.
- AFTER (end of report) or BEFORE (beginning of report) keywords are used in the RBREAK statement.
- Options must be specified after a forward slash.

Options	Definition
SUMMARIZE	includes a summary.
OL	overlines each value.
DOL	double-overlines each value.
UL	underlines each value.
DUL	double-underlines each value.

Examples:

```
proc report data=sashelp.shoes nowd headline;
   column subsidiary inventory sales;
   define subsidiary / group 'Subsidiary';
   define inventory / analysis mean 'Inventory';
   define sales / analysis mean 'Sales';
   rbreak after / dol summarize;
run;
```

Subsidiary	Inventory	Sales
Calgary	$94,340	$25,376
Montreal	$1,060,770	$323,073
Ottawa	$159,691	$71,746
Toronto	$420,451	$170,354
Vancouver	$5,688,106	$2,171,612
	$7,423,358	$2,762,161

```
proc report data=sashelp.shoes nowd headline;
   column subsidiary inventory sales;
   define subsidiary / group 'Subsidiary';
   define inventory / analysis mean 'Inventory';
   define sales / analysis mean 'Sales';
   rbreak before / dol summarize;
run;
```

Subsidiary	Inventory	Sales
	$7,423,358	$2,762,161
Calgary	$94,340	$25,376
Montreal	$1,060,770	$323,073
Ottawa	$159,691	$71,746
Toronto	$420,451	$170,354
Vancouver	$5,688,106	$2,171,612

Proc Tabulate

- Displays descriptive statistics in tabular format.
- Produce tables in up to three dimensions and allows, within each dimension, multiple variables to be reported one after another hierarchically.

Syntax:

```
PROC TABULATE <options>;
        CLASS variables < / options>;
        VAR variables < / options>;
        TABLE <page> ,
                <row> ,
                column
                < / options> ;
                 … other statements … ;
 RUN;
```

CLASS STATEMENT
Classification variables allow us to get statistics by category. We will get one column or row for each value of the CLASS variable.

VAR STATEMENT
The VAR statement is used to list the variables we intend to use to create summary statistics on. As such, they must be numeric.

TABLE STATEMENT

- To identify different dimensions, use commas. The order of the dimensions is page, row, and column. If we specify only one dimension, then it is assumed to be column. If two are specified, row, then column.

- Options appear at the end after a '/'. We can have multiple table statements in one PROC TABULATE. This will generate one table for each statement. All variables listed in the table statement must also be listed in either the VAR or CLASS statements.

- In the table expressions, there are many statistics that can be specified. Among them are row and column percents, counts, means, and percentiles.

Important points regarding Proc Tabulate

- A comma specifies to add a new dimension.
- The asterisk is used to produce a cross tabulation of one variable with another.
- A blank is used to represent concatenation.
- Parenthesis will group elements and associate an operator with each element in the group.
- Angle brackets specify a denominator definition for use in percentage calculations.

Example:

```
PROC TABULATE data =c.cust_seg;
        CLASS region sex;
        VAR post_usage_1month;
        TABLE region*mean, post_usage_1month*sex;
 RUN;
```

Adding Statistics

If a statistic name is not provided, the default statistic produced will be ‘N’ for the CLASS variables and ‘SUM’ for the VAR variables.

Descriptive Statistics	Quantile Statistics
COLPCTN	MEDIAN | P50
PCTSUM	P1
COLPCTSUM	Q3 | P75
MAX	P90
ROWPCTN	P95
MEAN	P5
ROWPCTSUM	P10
MIN	P99
STDDEV / STD	Q1 | P25
N	QRANGE
STDERR	
NMISS	
SUM	Hypothesis Testing
PAGEPCTSUM	ProbT
PCTN	T
VAR	

Adding and Hiding Row and Column Labels

There are two ways to add labels to variables.

- The first is by adding the text =‘label’ after the variable name. This will work for both variables and statistics.
- The second way is to add a LABEL statement for variables and/or a KEYLABEL statement for statistics to your code. For example:

LABEL var=‘label’;
KEYLABEL stat=‘label’;

In order to hide variable or statistic labels, leave the label specification blank (i.e. =‘ ‘).

Example:

```
PROC TABULATE data=one;
CLASS educ gender fulltime;
VAR income;
TABLE educ ,
Fulltime=’Employment Status’ * gender = ’ ’ *
 income * mean = ’ ’ ;
RUN;
```

Alternatively, we can also use this code to produce same table:

```
PROC TABULATE data=one;
CLASS educ gender fulltime;
VAR income;
TABLE educ,
Fulltime * gender * income * mean ;
LABEL gender=' ' Fulltime='Employment Status';
KEYLABEL mean=' ';
RUN;
```

	Employment Status			
	Fulltime		Parttime	
	Female	Male	Female	Male
	Income	Income	Income	Income
Educ				
High School		52506.67	44788.67	35600.00
Bachelors	42771.20		36947.00	
Masters		50729.25		62258.00
Doctorate	74589.60	56466.00		

Adding Totals and Subtotals

- To get marginal statistics in the table, use the ‘ALL’ keyword.
- Can place the keyword on the row or the column dimensions or on both.

```
PROC TABULATE data=one;
        CLASS gender fulltime educ;
        TABLE (fulltime ALL) * gender ALL,
                educ * N;
RUN;
```

		Educ			
		High School	Bachelors	Masters	Doctorate
		N	N	N	N
Fulltime	Gender				
Fulltime	Female		5.00		5.00
	Male	3.00	1.00	4.00	2.00
Parttime	Female	3.00	4.00	2.00	
	Male	3.00	1.00	3.00	
All	Female	3.00	9.00	2.00	5.00
	Male	6.00	2.00	7.00	2.00
All		9.00	11.00	9.00	7.00

Adding Formats to Statistics and removing Horizontal Separators

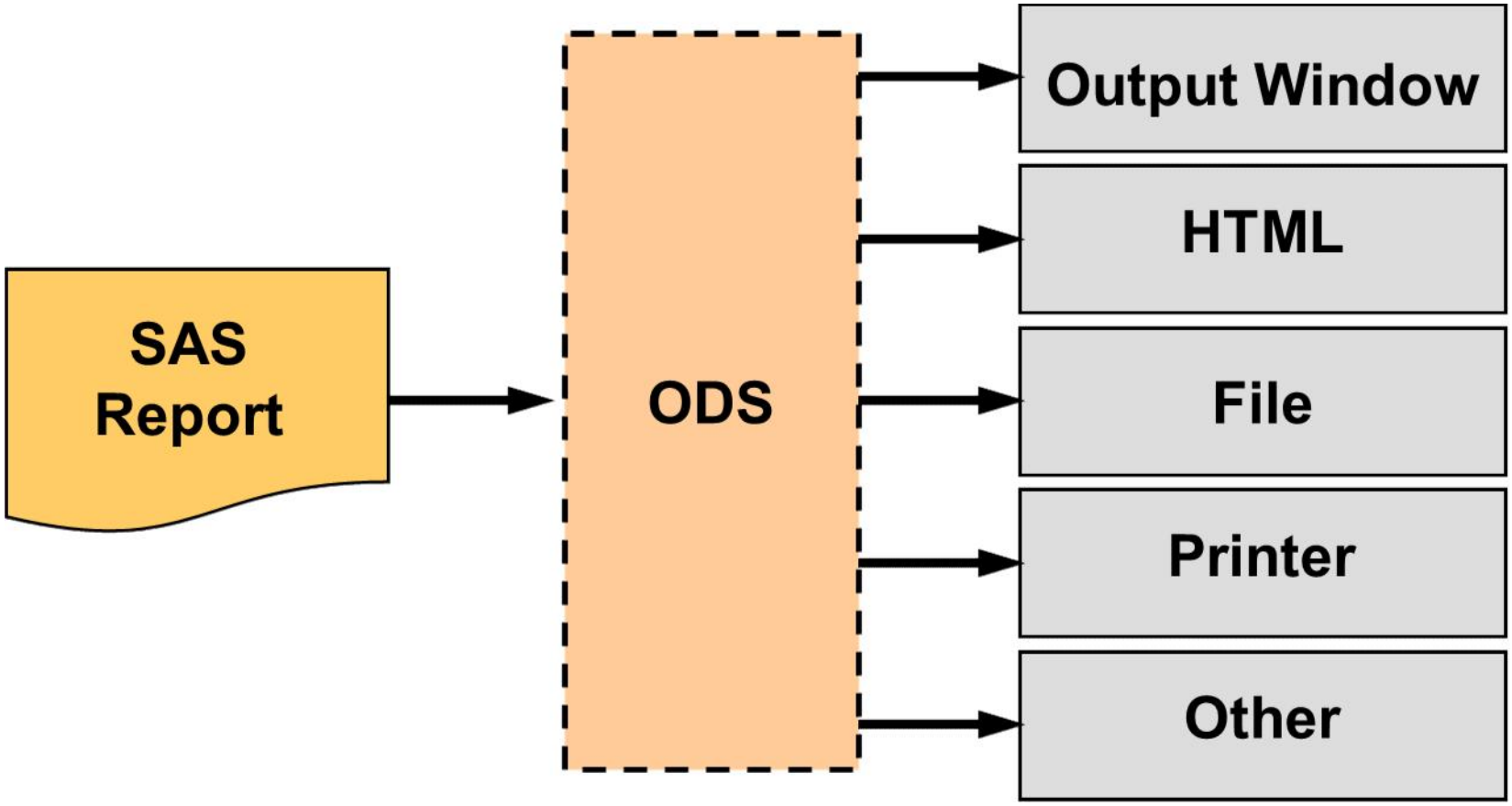
- We can specify formats for the numbers in the cells of the table using the *F=fmt. expression.
- NOSEPS option is used to remove the horizontal dividers between the row values from the table.

```
PROC TABULATE data=one NOSEPS;
        CLASS educ gender fulltime / missing;
        VAR income;
        TABLE educ=' ' * (fulltime=' ' ALL),
                gender=' ' * Income=' ' * ( N*F=6. MEAN*F=Dollar8. );
RUN;
```

		Female		Male	
		N	Mean	N	Mean
High School	Fulltime			3	$52,507
	Parttime	3	$44,789	2	$35,600
	All	3	$44,789	5	$45,744
Bachelors	Fulltime	5	$42,771	0	
	Parttime	3	$36,947	0	
	All	8	$40,587	0	
Masters	Fulltime			4	$50,729
	Parttime	0		3	$62,258
	All	0		7	$55,670
Doctorate	Fulltime	5	$74,590	2	$56,466
	All	5	$74,590	2	$56,466

Output Delivery System (ODS)

- Use ODS statements to specify destinations for your output
- Create output in a variety of formats
- the Listing destination is open by default



Syntax:

ODS *open-destination;*

ODS *close-destination* CLOSE;

Where,

- ***open-destination*** is a keyword and any required options for the type of output that is to be created, such as
  – HTML FILE=*'html-file-pathname'*
  – LISTING

- ***close-destination*** is a keyword for the type of output

Example:

```
ods html body = ‘ c:\mydata.html’;
    proc print data = sasuser.mydata;
    run;
 ods html close;
```

Here,

– The ods html statement creates an HTML output of the name mydata.html in the path specified.

ODS Destinations

- The table that follows lists the ODS destinations that are supported.

This destination…	Produces…
HTML	output that is formatted in HyperText Markup Language (HTML)
Listing	output that is formatted like traditional SAS procedure (listing) output
Markup Language Family	output that is formatted using markup languages such as Extensible Markup Language (XML)
ODS Document	a hierarchy of output objects that enables you to render multiple ODS output without re-running procedures
Output	SAS data sets
Printer Family	output that is formatted for a high-resolution printer, such as PostScript (PS), Portable Document Format (PDF), or Printer Control Language (PCL) files
RTF	Rich Text Format output for use with Microsoft Word

Closing Multiple ODS Destinations Concurrently

- Produce output in multiple formats concurrently by opening each ODS destination at the beginning of the program
- The keyword _**ALL**_ is used in the ODS CLOSE statement to close all open destinations concurrently

Syntax:

ODS *open-destination1;*
ODS *open-destination2;*

ODS _*all*_ CLOSE;

Where,

– ***open-destination1*** is a keyword and any required options for the first type of output that is to be created

– ***open-destination2*** is a keyword and any required options for the second type of output that is to be created

– _**all**_ keyword close all open destinations concurrently

Example:

```
ods html file = 'c:\admit.html' ;
ods pdf file = 'c:\admit.pdf' ;
      proc print data = sasuser.admit;
      run;
ods _all_ close;
```

Here,

- The ods html statement creates an HTML output of the name admit.html in the path specified
- The ods pdf statement creates a PDF output of the name admit.pdf in the path specified
- **FILE=** can also be used to specify the file that contains the HTML output. FILE= is an alias for BODY=.

Creating HTML Output from Multiple Procedures:

- Can also use the ODS HTML statement to direct the results from multiple procedures to the same HTML file

Syntax:

ODS *open-destination;*
Procedure1
Procedure2
ODS *close-destination* CLOSE;

Where,

– ***open-destination*** is a keyword and any required options for the type of output that is to be created

– ***Procedure1*** is the proc step for first procedure

– ***Procedure2*** is the proc step for second procedure

– ***close-destination*** is a keyword for the type of output

Example:

```
ods html body = ‘ c:\records\data.html’;
     proc print data =  clinic.admit label;
        var id sex age height weight actlevel;
        label actlevel = ’Activity Level’;
    run;
    proc tabulate data = clinic.stress2;
         var resthr maxhr rechr;
          table min mean, resthr maxhr rechr;
     run;
ods html close;
```

Here,

– The program above generates HTML output for the PRINT and TABULATE procedures