(7) LABELING AND GROUPING

Variable ***labels*** created with a LABEL statement can be used to associate more information with the ***names of variables***. For example, you know that a variable named PDX in a data set contains the value of a principal diagnosis. Adding a label to the variable PDX and using that label in place of or together with the variable name will also let others know that information.

Formats can be used to associate more information with the ***values of variables***. Formats can also be used to ***group*** observations based on the ***values of variables***. You have already seen some SAS-supplied formats, e.g. MMDDYY10. used when printing the value of a variable that contains a date or 4.1 when printing the value of a variable younded to one decimal place. PROC FORMAT allows you to create your own formats (or 'rules') to control the appearance of the values of variables and to create rules to group observations based on the values of variables. The process of either changing the values of variables or changing their appearance is sometimes referred to as recoding.

...LABELING VALUES/GROUPING OBSERVATIONS...

There are two methods to add more information to variable values. One of them is to create new variables based on variable values. The other is to create formats to label variable values. Creating new variables is done within a data step.

...Example 7.1...

```
data c99;
set x.cancer99;

if gender eq '1' then newgender = 'MALE  ';  ①
else                 newgender = 'FEMALE';

if cause eq : 'C34' then newcause = 'BRONCHUS & LUNG'; ②
else
if cause eq : 'C18' then newcause = 'COLON';
else
if cause eq : 'C50' then newcause = 'BREAST';
else
if cause eq : 'C25' then newcause = 'PANCREAS';
else
if cause eq : 'C61' then newcause = 'PROSTATE';
else                     newcause = 'OTHER';
;
run;

title
'RECODED DATA - NEW VARIABLES USING IF-THEN-ELSE';
proc freq data=c99;
table newcause*newgender / norow nopercent; ③
run;
```

A data step is used to add two new variables to the cancer data set. The variable NEWGENDER contains labels for the two values of the variable GENDER ①. The variable NEEWCAUSE contains labels for ranges of the value of the variable CAUSE ②. The new variables are used in PROC FREQ to create a table showing the distribution of cancer deaths by cause within gender ③. Only the column percentages are left in the table, showing the percentage of deaths from each cause within gender. Both functions, labels for variable values and forming groups based on variable values, can also be done with formats.

RECODED DATA - NEW VARIABLES USING IF-THEN-ELSE

Table of newcause by newgender

newcause Frequency Col Pct	FEMALE	MALE	Total
BREAST	3113 16.35	41 0.22	3154
BRONCHUS & LUNG	4319 22.68	5317 28.68	9636
COLON	1940 10.19	1705 9.20	3645
OTHER	8514 44.71	8364 45.11	16878
PANCREAS	1155 6.07	1065 5.74	2220
PROSTATE	0 0.00	2050 11.06	2050
Total	19041	18542	37583

...Example 7.2...

```
proc format; ①
value $gen  ②
'1' = 'MALE'   ③
'2' = 'FEMALE'
;
value $cau ④
'C34'-'C349' = 'BRONCHUS & LUNG' ⑤
'C18'-'C189' = 'COLON'
'C50'-'C509' = 'BREAST'
'C25'-'C259' = 'PANCREAS'
'C61'        = 'PROSTATE'
;
run;

title 'DEATHS BY CAUSE WITHIN GENDER USING FORMATS';
proc freq data=c99;
tables cause*gender/norow nopercent; ⑥
where cause in : ('C34' 'C18' 'C50' 'C25' 'C61'); ⑦
format cause $cau. gender $gen.; ⑧
run;
```

DEATHS BY CAUSE WITHIN GENDER USING FORMATS

Table of CAUSE by GENDER

CAUSE(ICD-10 CAUSE OF DEATH)

Frequency Col Pct	GENDER MALE	FEMALE	Total
COLON	1705 16.75	1940 18.43	3645
PANCREAS	1065 10.46	1155 10.97	2220
BRONCHUS & LUNG	5317 52.24	4319 41.03	9636
BREAST	41 0.40	3113 29.57	3154
PROSTATE	2050 20.14	0 0.00	2050
Total	10178	10527	20705

PROC FORMAT is used to create user-defined formats, or rules that can be used when displaying the values of variables ①. One or more VALUE statements can be used within PROC FORMAT. The first value statement creates the format $GEN ②. Values on the left of the '=' are values found in your data, while labels on the right of the '=' is text to be substituted for those values ③. A second VALUE statement creates another format, $CAU ④. In this format, ranges of values on the left of the '=' are values found in your data, while labels on the right of the '=' is text to be used in place of range of values rather than a single value ⑤. PROC FREQ is used to create a table of cause of death groups with values of the variable GENDER ⑥. The WHERE statement limits causes used in the table to those causes found in the format $CAU ⑦. A FORMAT statement instructs SAS to use a specific format for each variable ⑧. In a FORMAT statement SAS first looks for one or more variable names followed by a format name. In this example, you tell SAS to use the format $CAU with the variable CAUSE and the format $GEN with the variable GENDER. In the resulting table, all you see are the formatted values of the variables GENDER and CAUSE. Values of GENDER are displayed using the labels assigned in the format $GEN while values of CAUSE are displayed with labels found in the format $CAU. Notice that format names need not be the same names as the variables with which they are used. However, you should develop a habit of creating format names that allow you to remember the variable(s) with which you will use the format(s) you create.

What order is used for the rows and columns in the table? It is not alphabetic order of the format labels. It is the default order of tables created with PROC FREQ, the internal order of the variable values. Since the value of the variable GENDER for males is 1, males appear before females. The causes of death are in ICD-10 cause of death code order, colon cancer being all deaths starting with C18, pancreas starts with C25, through the last group, prostate, with a code of C61. You can use an option in PROC FREQ to change the order of the rows and columns so they are in the order of the format labels.

...Example 7.3...

```
proc format;
value $cau ①
'C34'-'C349' = 'BRONCHUS & LUNG'
'C18'-'C189' = 'COLON'
'C50'-'C509' = 'BREAST'
'C25'-'C259' = 'PANCREAS'
'C61'        = 'PROSTATE'
other        = 'OTHER CAUSE' ②
;
run;

title1 'DEATHS BY CAUSE WITHIN GENDER USING FORMATS';
title2 'ALL DEATHS IN FORMATTED ORDER';
proc freq data=c99 order=formatted; ③
tables cause*gender/norow nopercent;
format cause $cau. gender $gen.;
run;
```

```
NOTE: Format $CAU is already on the library.
NOTE: Format $CAU has been output.
```

PROC FORMAT is used again, this time to change the already existing format $CAU ①. An 'other' condition is added on the left of the '=' meaning that any value not found in the ranges of cause of death codes in the format is assigned a label of OTHER CAUSE ②. The word 'other' is one of three allowable words than can appear on the left of the '=' in PROC FORMAT (the others are 'low' and 'high'). When you revise an already existing format, you will see a message in the LOG, shown above the table on the right. That is not an error, PROC FORMAT is merely telling you that you have replaced an old format with a new format that has the same name as the old one.

DEATHS BY CAUSE WITHIN GENDER USING FORMATS
ALL DEATHS IN FORMATTED ORDER

Table of CAUSE by GENDER

CAUSE(ICD-10 CAUSE OF DEATH)

Frequency Col Pct	GENDER FEMALE	MALE	Total
BREAST	3113 16.35	41 0.22	3154
BRONCHUS & LUNG	4319 22.68	5317 28.68	9636
COLON	1940 10.19	1705 9.20	3645
OTHER CAUSE	8514 44.71	8364 45.11	16878
PANCREAS	1155 6.07	1065 5.74	2220
PROSTATE	0 0.00	2050 11.06	2050
Total	19041	18542	37583

An ORDER option is added to PROC FREQ ③. Using a value of FORMATTED changes the order of the rows and columns to that found in the format labels rather than the variable values. Notice that there is no WHERE statement in this example as there was in example 7.2 since there now is a rule for every value of the variable CAUSE. Any value of CAUSE nor found in the specified ranges is assigned a label of OTHER CAUSE. Also notice that the table is identical to that produced in example 7.1 using new variables. The new table is created using already existing variables and a set of rules created in PROC FORMAT.

The two formats in the previous two examples are used with character variables. You should have noticed that the names of the user-written formats $GEN and $CAU both begin with '$'. That is one of the rules you will have to remember ... the names of formats to be used with CHARACTER variables MUST BEGIN with a '$'. There is one numeric variable in the data set CANCER99. Another rule you must remember is that ... the names of formats to be used with a NUMERIC variable DO NOT BEGIN with a '$'. Both character and numeric formats are used in the following example.

...Example 7.4...

```
proc format;
value age ①
low - 44   = '<45' ②
45  - 54   = '45-54'
55  - 64   = '55-64'
65  - 74   = '65-74'
75  - high = '75+' ③
;
value $pla
'A'-'C'    ='HOSP/OTHER'
'D'        ='HOSP/INPATIENT'
'E'        ='OTHER INST'
'F'        ='RESIDENCE'
'G','H','N'='OTHER' ④
other      ='UNKNOWN'
;
run;
```

CHARACTER AND NUMERIC VARIABLES IN TABLE
REQUIRE CHARACTER AND NUMERIC FORMATS

Table of PLACE by AGE

PLACE(WHERE DEATH OCCURRED) AGE(AGE AT DEATH (YEARS))

Frequency Col Pct	.	<45	45-54	55-64	65-74	75+	Total
UNKNOWN	0 0.00	12 0.75	21 0.64	34 0.56	60 0.58	108 0.66	235
HOSP/OTHER	1 20.00	69 4.29	132 4.01	225 3.74	340 3.31	473 2.89	1240
HOSP/INPATIENT	2 40.00	1090 67.74	2028 61.55	3519 58.42	5623 54.68	7980 48.76	20242
OTHER INST	0 0.00	61 3.79	178 5.40	381 6.32	909 8.84	2838 17.34	4367
RESIDENCE	2 40.00	278 17.28	680 20.64	1401 23.26	2498 24.29	3398 20.76	8257
OTHER	0 0.00	99 6.15	256 7.77	464 7.70	853 8.30	1570 9.59	3242
Total	5	1609	3295	6024	10283	16367	37583

```
title1 'CHARACTER AND NUMERIC VARIABLES IN TABLE';
title2 'REQUIRE CHARACTER AND NUMERIC FORMATS';
proc freq data=x.cancer99;
tables place*age/norow nopercent missing; ⑤
format place $pla. age age.;
run;
```

The format AGE is to be used with the numeric variable AGE, so its name does not start with a '$' ①. Two of the three words allowed on the left of the '=' in PROC FORMAT are used in the ranges of values of the variable AGE, 'low' ② and 'high' ③. Yes, you could have used a 0 (zero) in place of the word 'low' and a very high number (maybe 199) in place of the word 'high'. Using 'low' and/or 'high' asks SAS to look for the lowest and highest values of a variable in your data when you use the format. In the format $PLA, individual values of a variable in the left of the '=' are separated by commas ④ ... that is yet another rule to remember when using PROC FORMAT. A table is created in PROC FREQ and missing data are included in the table by using the MISSING option ⑤. Notice, the range 'low-44' in the format AGE did not include missing values of AGE. There are 5 observations with missing values of AGE and they appear in their own column. Also notice that the group OTHER in the table for PLACE may contain missing values or any other value of PLACE not found in those listed in PROC FORMAT. Missing data can created problems when using formats and those problems are discussed after the rules for creating user-written formats are made explicit.

In addition to expressing ranges with values separated by a '-', there are instances when other symbols are required in a format. For example, what if we have data where one of the variables is a number of days. We want to create counts, but with observations grouped by weeks.

...Example 7.5...

```
proc format;
value weeks ①
low - 19   = "< 20 WEEKS"
20  - 49   = "20-49 WEEKS"
50  - high = "50+ WEEKS"
;
run;
```

```
data day_week;
input days @@;
weeks = days / 7; ②
datalines;
140 136 250 300 350 400 348 120
;
run;

title 'WEEKS GROUPED USING A FORMAT WITH RANGES';
proc freq data=day_week;
tables weeks;
format weeks weeks.; ③
run;
```

WEEKS GROUPED USING A FORMAT WITH RANGES

WEEKS	Frequency	Percent	Cumulative Frequency	Cumulative Percent
< 20 WEEKS	1	12.50	1	12.50
19.42857143	1	12.50	2	25.00
20-49 WEEKS	3	37.50	5	62.50
49.71428571	1	12.50	6	75.00
50+ WEEKS	2	25.00	8	100.00

The numeric format WEEKS is created and it will be used to group observations into week ranges ①. The variable WEEKS is created by dividing DAYS by 7 ②. PROC FREQ creates a table, grouping observations using the recently created format ③. In the table on the right, what are those entries with all the decimal places? Values for WEEKS fall 'between the cracks' when the format is used. The first range ends with 19 and the second starts with 20. There is no range assigned for values greater than 19 and less than 20. The same is true for values greater than 49 and less than 50. That is what causes the extra entries in the output from PROC FREQ. You can recreate the format using a new style of expressing a range.

...Example 7.6...

```
proc format;
value weeks
low -< 20  = "< 20 WEEKS" ①
20  -< 50  = "20-49 WEEKS"
50  - high = "50+ WEEKS"
;
run;

title 'WEEKS GROUPED USING A FORMAT WITH RANGES USING -<';
proc freq data=day_week;
tables weeks;
format weeks weeks.;
run;
```

WEEKS GROUPED USING A FORMAT WITH RANGES USING -<

WEEKS	Frequency	Percent	Cumulative Frequency	Cumulative Percent
< 20 WEEKS	2	25.00	2	25.00
20-49 WEEKS	4	50.00	6	75.00
50+ WEEKS	2	25.00	8	100.00

The format now indicates that you want all values that are less than 20 weeks included in the first range, and all values less than 50 weeks included in the second range ①. If you wanted fractional values treated differently, i.e. any value of weeks greater than 19 to be in the second range (20-49) and any value greater than 49 in the last range (50+), the format can be changed.

...Example 7.7...

```
proc format;
value weeks
low - 19   = "< 20 WEEKS" ①
19 <- 49   = "20-49 WEEKS"
49 <- high = "50+ WEEKS"
;
run;
```

```
title 'WEEKS GROUPED USING A FORMAT WITH RANGES USING <-';
proc freq data=day_week;
tables weeks;
format weeks weeks.;
run;
```

The format now indicates that you want all values greater than 19 weeks included in the second range, and all values greater than 49 weeks included in the third range ①. The results are different from the previous example.

WEEKS GROUPED USING A FORMAT WITH RANGES USING <-

WEEKS	Frequency	Percent	Cumulative Frequency	Cumulative Percent
< 20 WEEKS	1	12.50	1	12.50
20-49 WEEKS	4	50.00	5	62.50
50+ WEEKS	3	37.50	8	100.00

...CREATING AND USING FORMATS...

There are a number of rules for creating and using formats. One of the rules forces you to think ahead in that when you create a format, the format type determines the variable type with which the format can be used ...

FORMAT TYPES

- ***if you plan to use a format with a character variable, create a character format***
- ***if you plan to use a format with a numeric variable, create a numeric format***

How do you choose a format type? A format type is determined by the name you give a format. You give a format a name in a VALUE statement within PROC FORMAT. You can enter as many VALUE statements as you like More rules to remember are as follows ...

FORMAT NAMES

- ***the name of the format determines the format type***
- ***if a format name begins with a $, it is character format***
- ***if there is no $ at the start of the name, it is a numeric format***
- ***a format name can be up to thirty-two characters in length (a $ counts as part of the thirty-two characters)***
- ***a format name must start with a letter or underscore (the first character or the first character after the $)***
- ***a format name can only contain letters, numbers, and underscores***
- ***a format name cannot end in a number***

The rules about types and names all apply to the first text SAS sees in a VALUE statement within PROC FORMAT. Once you specify a format name, there are also rules for what appears of the left and right side of the equals sign, how you can write values of variables and labels for those values.

FORMAT RULES-LEFT SIDE OF '='

- ***values on the left side of the ='s are variable values***
- ***individual values, multiple values separated by commas, ranges of values separated by a dash are allowed (for ranges: -< includes all values greater than or equal to the lower range value and less than the upper range value; <- includes all values greater than the lower range value and less than or equal to the upper range value)***
- ***combinations of individual values, multiple values, and ranges are allowed***
- ***numeric values are not enclosed in quotes***
- ***character values are enclosed in quotes***
- ***the words LOW, HIGH, and OTHER may appear in addition to variable values***
- ***values of variables associated with different labels (the right side of '=') should not overlap or be repeated***

FORMAT RULES-RIGHT SIDE OF '='

- ***a format label may be up to 32,767 characters in length (the content to the right of the '=' sign)***
- ***use quotation marks around format labels (the content to the right of the '=' sign)***

The use of quotation marks around all entries (value labels) on the right side of the '=' for all formats is a convention rather than a rule, as is quotation marks around entries (variable values) on the left side of the '=' in character formats. You can create formats without the quotation marks on either side. However, it is easier to just follow the rules you see in ***bold italics***.

Creating a user-written format in PROC FORMAT is only one step in the process of labeling or grouping variable values based on that format. You must use a FORMAT statement to explicitly tell SAS to use a format with a variable. A FORMAT statement can be used in either a data step or a procedure and the effect of that statement differs based on where it is used.

ASSOCIATING FORMATS WITH VARIABLES

- ***if a FORMAT statement is used in a DATA STEP to associate a format with a variable, that association is permanent and that information added to the header of the data set containing that variable (you will see that information is you use PROC CONTENTS) ... if you want to remove that association permanently, you must use PROC DATASETS or another data step***
- ***if a FORMAT statement is used in a PROCEDURE to associate a format with a variable, that association is temporary and only lasts for the duration of the procedure ... the exception to this rule is PROC DATASETS which can be used in place of a data step for permanent association variables and formats***

Finally, the last rule involves the format name and how it differs between when it is created and when it is used.

FORMAT NAMES-USING FORMATS, CREATING FORMATS

- ***when a format is USED in a data step or procedure, ADD a PERIOD at the end of the name***
- ***when a format is CREATED in PROC FORMAT, DO NOT ADD a PERIOD at the end of the name***

...MORE ABOUT ASSOCIATING FORMATS WITH VARIABLES...

PROC FORMAT creates rules for the display of the values of variables. It does not apply these rules to any variables in any data sets. Application of a format to a variable in a data set is done with a FORMAT statement. You can also use an ATTRIB statement to associate a format with a variable. However, the ATTRIB statement is not described in these notes.

A FORMAT statement can occur either in a data step or in a procedure, but its effect on a data set is different in a data step than in a procedure.

...Example 7.8...

```
proc format;
value $gender '1' = 'MALE' '2' = 'FEMALE'; ①
value $yesno  '0' = 'NO'   '1' = 'YES';
run;

data test;
input
@01 gender   $1.
@02 age       2.
@04 (q1-q5) ($1.)
;
format gender $gender. q1-q5 $yesno.; ②
```

```
datalines;
11001011
24511111
12800011
;
run;

proc contents data=test;
run;

title 'DATA SET TEST WITH FORMATTED VARIABLES';
proc print data=test;
run;

title
'DATA SET TEST WITH FORMATS TEMPORARILY REMOVED';
proc print data=test;
format gender q1-q5; ③
run;
```

Two character formats are created ①. When data set TEST is created. the FORMAT statement tells SAS to associate the format $GENDER. with the variable GENDER and the format $YESNO. with the five variables Q1 through Q5 ②. The output from PROC CONTENTS shows that six variables in data set TEST are formatted and whenever values of those variables are displayed, you will se the format labels, not the original data values. You can see that in the output of PROC PRINT. If you want to display the original data values rather than format labels, you can use a FORMAT statement that specifies one or more variables without using any format names ③. This temporarily removes the association of a variable with a format. It does not modify the data set.

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format
2	age	Num	8	
1	gender	Char	1	$GENDER.
3	q1	Char	1	$YESNO.
4	q2	Char	1	$YESNO.
5	q3	Char	1	$YESNO.
6	q4	Char	1	$YESNO.
7	q5	Char	1	$YESNO.

DATA SET TEST WITH FORMATTED VARIABLES

Obs	gender	age	q1	q2	q3	q4	q5
1	MALE	10	NO	YES	NO	YES	YES
2	FEMALE	45	YES	YES	YES	YES	YES
3	MALE	28	NO	NO	NO	YES	YES

DATA SET TEST WITH FORMATS TEMPORARILY REMOVED

Obs	gender	age	q1	q2	q3	q4	q5
1	1	10	0	1	0	1	1
2	2	45	1	1	1	1	1
3	1	28	0	0	0	1	1

You can use a SAS supplied variable name, _ALL_, in place of a list of variable names, for example ...

```
format _all_;          ... can be used to replace...        format gender q1-q5;
```

to remove the association of all formats with all variables.

The effect of a FORMAT statement in a data step is permanent in that the data set is actually modified. If you use a FORMAT statement in a procedure, the effect is temporary and the data set is not modified.

...Example 7.9...

```
proc format;
value $gender '1' = 'MALE' '2' = 'FEMALE';
value $yesno  '0' = 'NO'   '1' = 'YES';
run;

data test;
input
@01 gender   $1.
@02 age       2.
@04 (q1-q5) ($1.)
;
```

```
datalines;
11001011
24511111
12800011
;
run;

title 'DATA SET TEST WITH FORMATS APPLIED IN A PROCEDURE';
proc print data=test;
format gender $gender. q1-q5 $yesno.; ①
run;

proc contents data=test;
run;
```

The only difference between this example and example 7.14 is that the FORMAT statement is moved into the procedure ①. There is no FORMAT statement in the data step. The output from PROC PRINT displays formatted values. When PROC CONTENTS is run after PROC PRINT, you can see in the results on the right that the data set has not been modified in that the variables are not formatted.

DATA SET TEST WITH FORMATS APPLIED IN A PROCEDURE

Obs	gender	age	q1	q2	q3	q4	q5
1	MALE	10	NO	YES	NO	YES	YES
2	FEMALE	45	YES	YES	YES	YES	YES
3	MALE	28	NO	NO	NO	YES	YES

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
2	age	Num	8
1	gender	Char	1
3	q1	Char	1
4	q2	Char	1
5	q3	Char	1
6	q4	Char	1
7	q5	Char	1

Example 7.14 shows how make a permanent association of formats with variables in a data step. You can also make a permanent disassociation in a data step. Assume that you create data set TEST in example 7.14 ... six variables are formatted. If you use this data step ...

```
data test;
set  test;
format _all_;
run;
```

the formats are removed form all variables. That is an inefficient method of removing formats since in addition to removing the formats, the data step reads and rewrites all the observations in the data set. A more efficient way to remove (or add) format is using PROC DATASETS (used already in examples 2.4 and 6.4 to modify SAS data sets).

...Example 7.10...

```
proc format;
value $gender '1' = 'MALE' '2' = 'FEMALE';
value $yesno  '0' = 'NO'   '1' = 'YES';
run;

data test; ①
input
@01 gender   $1.
@02 age       2.
@04 (q1-q5) ($1.)
;
datalines;
11001011
24511111
12800011
;
run;
```

```
proc datasets lib=work nolist; ②
modify test;
format gender $gender. q1-q5 $yesno.;
run;

proc contents data=test; ③
run;

proc datasets lib=work nolist;
modify test;
format gender q1-q5; ④
run;

proc contents data=test;
run;
```

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format
2	age	Num	8	
1	gender	Char	1	$GENDER.
3	q1	Char	1	$YESNO.
4	q2	Char	1	$YESNO.
5	q3	Char	1	$YESNO.
6	q4	Char	1	$YESNO.
7	q5	Char	1	$YESNO.

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
2	age	Num	8
1	gender	Char	1
3	q1	Char	1
4	q2	Char	1
5	q3	Char	1
6	q4	Char	1
7	q5	Char	1

The data set test is created without associating formats with variables in the data step ①. PROC DATASETS ② is used to add the formats and the results of PROC CONTENTS ③ in the top box show that the formats are permanently associated with variables. PROC DATASETS requires a library name (LIB=) and a data set name (MODIFY <data set name>). You can also remove formats with PROC DATASETS by specifying variable names and no format names ④. Once again, you can use a SAS supplied variable name, _ALL_, in place of a list of variable names, for example ...

```
format _all_;          ... can be used to replace...        format gender q1-q5;
```

to remove the formats from all variables.

...USING SAS-SUPPLIED FORMATS TO GROUP OBSERVATIONS...

There are several SAS-supplied formats that allow you to group observations. When working with medical data such as diagnosis codes, procedure codes, cause of death codes, etc., the codes usually have a hierarchy. For example, ICD-9-CM diagnosis codes have five digits. The first three digits for those related to diabetes mellitus are always '250' while the 4th digit adds more detail to the diagnosis ...

```
2500 DIABETES MELLITUS UNCOMP                2501 DIABETES W KETOACIDOSIS
2502 DM W HYPEROSMOLARITY                    2503 DIABETES W COMA NEC
2504 DM W RENAL MANIFESTATION                2505 DM W OPHTHALMIC MANIFEST
2506 DM2 NEUROLOGIC MANIFEST                 2507 DM W CIRC DISORDER
2508 DIABETES W MANIFEST NEC                 2509 DIABETES W COMP NOS
```

It is not important to understand all the abbreviations, only that the 4th digit adds detail to a diagnosis that starts with '250'. Within each of the above 4-digit categories, a 5th digit adds even more detail, for example, within '2500' ...

```
25000   DIABETES MELLITUS WITHOUT MENTION OF COMPLICATION, TYPE II OR UNSPECIFIED TYPE,
        NOT STATED AS UNCONTROLLED
25001   DIABETES MELLITUS WITHOUT MENTION OF COMPLICATION, TYPE I [JUVENILE TYPE],
        NOT STATED AS UNCONTROLLED
25002   DIABETES MELLITUS WITHOUT MENTION OF COMPLICATION, TYPE II OR UNSPECIFIED TYPE, UNCONTROLLED
25003   DIABETES MELLITUS WITHOUT MENTION OF COMPLICATION, TYPE I [JUVENILE TYPE], UNCONTROLLED
```

The same type of hierarchy exists for other diagnoses with the first three digits always defining a major diagnostic category. ICD-9-CM procedure codes have four digits, with the first two digits defing mjor procedure groups and the 3rd and 4th digits used to add detail. For example, procedure codes for operations performed on heart vessels all begin with '36'. The 3rd digits adds more detail to the description of the operation ...

```
360 RMVL COR ART OBSTR/STENT                361 HRT REVASC BYPASS ANAST
362 ARTERIAL IMPLANT REVASC                 363 OTHER HEART REVASC
369 OTHER HEART VESSEL OPS
```

As with the diabetes codes, it is not important to understand all the abbreviations, only that the 3rd digit adds detail to a procedure that starts with '36'. Within each of the above 3-digit categories, a 4th digit adds even more detail ...

```
3610   AORTOCORONARY BYPASS FOR HEART REVASCULARIZATION, NOT OTHERWISE SPECIFIED
3611   (AORTO)CORONARY BYPASS OF ONE CORONARY ARTERY
3612   (AORTO)CORONARY BYPASS OF TWO CORONARY ARTERIES
3613   (AORTO)CORONARY BYPASS OF THREE CORONARY ARTERIES
3614   (AORTO)CORONARY BYPASS OF FOUR OR MORE CORONARY ARTERIES
3615   SINGLE INTERNAL MAMMARY-CORONARY ARTERY BYPASS
3616   DOUBLE INTERNAL MAMMARY-CORONARY ARTERY BYPASS
3617   ABDOMINAL-CORONARY ARTERY BYPASS
3619   OTHER BYPASS ANASTOMOSIS FOR HEART REVASCULARIZATION
```

In some of the examples used thus far, you have also seen the same type of hierarchy in ICD-10 cause of death codes. Given these types of hierarchies, SAS-supplied formats can be used to group observations based on a portion of the value of a variable, for example, the first two digits of a procedure code.

...Example 7.11...

```
data procedures;
input pr : $4. @@; ①
datalines;
3734 3791 3751 740 7499 412 4101
;
run;

title 'PROCEDURE CODES GROUPED USING FORMAT $2.';
proc freq data=procedures;
table pr;
format pr $2.; ②
run;
```

Procedure codes are read from a DATALINES file ①. The variable PR has a length of 4. PROC FREQ is used to count observations within procedure codes and the codes are grouped based on the first two characters using the SAS-supplied character format $2. ②. The results are shown on the right.

PROCEDURE CODES GROUPED USING FORMAT $2.

PR	Frequency	Percent	Cumulative Frequency	Cumulative Percent
37	3	42.86	3	42.86
41	2	28.57	5	71.43
74	2	28.57	7	100.00

In the data just used, the first three procedures are ...

```
3734  EXCISION OR DESTRUCTION OF OTHER LESION OR TISSUE OF HEART
3791  OPEN CHEST CARDIAC MASSAGE
3751  HEART TRANSPLANTATION
```

The major subgroup they all belong to is ...

```
37  OTHER OPERATIONS ON HEART AND PERICARDIUM
```

There is no need to create a new variable to count procedures using only the first two characters in the procedure code. The format $2. will group observations based on the first two characters of any character variable. If you were using ICD-9-CM diagnosis codes, the format $3. can be used to count diagnoses in major subgroups. In chapter 8, there are examples that show how to use SAS-supplied formats to group observations based on the values of variables that are dates. For example, observations can be grouped by year using the format YEAR. or month using the format MONTH.

...CREATING NEW VARIABLES USING FORMATS...

Though formats can avoid the necessity of creating new variables for performing grouped analyses, formats also allow you to create new variables within a data step. The PUT function can create a new variable based on the value of an already existing variable according to rules specified in a format. Though the emphasis in this chapter is to use formats instead of creating new variables, there are circumstances where a new variable is necessary. For example, the SAS data set CLASS used in chapter 1 has three numeric variables: age in years; height in inches; weight in pounds. If you wanted to determine the mean weight for observations in two groups, those less than 5 feet tall and those 5 feet tall or taller, you could use a format.

...Example 7.12...

```
proc format;
value age2group ①
low - 44   = "1 - <45"
45  - 64   = "2 - 45-64"
65  - high = "3 - 65+"
other      = "4 - UNKNOWN"
;
run;

data c99;
set x.cancer99;
age_group = put(age,age2group.); ②
keep age_group gender;
run;

title 'TABLE USING THE NEW VARIABLE AGE_GROUP
CREATED USING A PUT FUNCTION';
proc freq data=c99;
table age_group*gender / nopercent;
run;
```

TABLE WITH NEW VARIABLE AGE_GROUP CREATED USING A PUT FUNCTION

Table of age_group by GENDER

age_group	GENDER		
Frequency Row Pct Col Pct	1	2	Total
1 - <45	731 45.43 3.94	878 54.57 4.61	1609
2 - 45-64	4741 50.87 25.57	4578 49.13 24.04	9319
3 - 65+	13069 49.04 70.48	13581 50.96 71.33	26650
4 - UNKNOWN	1 20.00 0.01	4 80.00 0.02	5
Total	18542	19041	37583

PROC FORMAT creates a numeric format named AGE2GROUP ①. The PUT function creates a new variable based on the value if the variable AGE and the format AGE2GROUP ②. The PUT function requires two arguments (arguments are the information within the parentheses that follow the name of the function): a variable name; a format name. The new variable that results from the use of the PUT function is ALWAYS a CHARACTER variable regardless of the type of variable being used to create the new variable.

When the format AGE2GROUP was created, we planned to use it with the numeric variable age in either a subsequent data step or procedure. Therefore, the name (or VALUE) of the format did NOT begin with a $. Also, all the values on the left side of the '=' in PROC FORMAT were expressed without quote marks. As with all formats, the values on the right side of the '=' do have quote marks (remember that is the safe method of creating a format, but not always required). A PUT function was used within the data step to create the new variable AGE_GROUP. Though AGE is a numeric variable and AGE2GROUP is a

numeric format, the variable AGE_GROUP is a character variable since the PUT function always results in the creation of a character variable. Remember that a format (or VALUE) is merely a set of rules that you set up someplace in your SAS job. The rule is invoked within the data step with a PUT function. The rule AGE2GROUP tells SAS to: compare the value of the variable AGE to values on the left side of the '=' in the format AGE2GROUP; when an appropriate value or range of values is found, use the values on the right side of the '=' to create the value of the new variable.

...FORMATS AND MISSING DATA...

Creating formats to use with variables that contain missing values can cause problems. You must understand that PROC FORMAT treats missing data differently in numeric and character formats and that the meaning of the 'low' differs in the two types of formats. The first examples examine what happens when a numeric variable with missing data is used with a numeric format.

...Example 7.13...

```
proc format; ①
value age2grp
low - 19   = "TEEN" ②
20  - 54   = "MIDDLE-AGE"
55  - high = "SENIOR"
;
run;

data ages; ③
length age_group $10;
input age @@;
if age le 19 then age_group = 'TEEN'; ④
else
if age le 54 then age_group = 'MIDDLE-AGE';
else              age_group = 'SENIOR';
datalines;
10 20 56 32 78 34 65 . 86 . 45 185 32 100 56 12
10 0 199 87 54 13 ⑤
;
run;

title 'MISSING NUMERIC DATA - FORMAT VERSUS IF-
THEN-ELSE';
proc freq data=ages order=formatted;
table age_group age / missing; ⑥
format age age2grp.;  ⑦
run;
```

MISSING NUMERIC DATA - FORMAT VERSUS IF-THEN-ELSE

age_group	Frequency	Percent	Cumulative Frequency	Cumulative Percent
MIDDLE-AGE	6	27.27	6	27.27
SENIOR	9	40.91	15	68.18
TEEN	7	31.82	22	100.00

age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
.	2	9.09	2	9.09
MIDDLE-AGE	6	27.27	8	36.36
SENIOR	9	40.91	17	77.27
TEEN	5	22.73	22	100.00

PROC FORMAT is used to create a format that can be used to group observations based on the values of the numeric variable AGE ①. The word 'low' is used in the range for the observations that will be assigned the label 'TEEN' ②. A data step creates the data set AGES ③. Within that data step, the variable AGE-GROUP is created using an IF-THEN-ELSE statement ④. The group 'TEEN' is to include all observations in which the AGE is less than or equal to 19. Two of the values for AGE are missing ⑤. PROC FREQ creates two tables and missing values are to appear in the tables ⑥. The first table uses the variable AGE_GROUP (created with an IF-THEN-ELSE statement). The second table uses the variable AGE and observations are grouped based on the format AGE2GRP ⑦.

The output on the right shows the two tables. If you look at the top table created with the variable AGE_GROUP, the observations with a missing value for AGE are included in the TEEN group. If you write a statement in a data step that asks if a variable is less than a given value (in this case less than or equal to 19), SAS will recognize a missing value as being less than the given value. If you look at the lower table, there is a separate missing entry in the table. Even though the first range in the format AGE2GRP starts with the word 'low', missing values are not recognized as real values and are left out of the TEEN group.

If you modify the format AGE2GRP created in example 7.5, you will see another problem caused by missing data when you use a format.

...Example 7.14...

```
proc format;
value age2grp ①
10  - 19   = "TEEN"
20  - 54   = "MIDDLE AGE"
55  - 110  = "SENIOR"
other      = "TOO LOW/TOO HIGH"
;
run;

title "MISSING NUMERIC DATA - 'OTHER' ADDED TO FORMAT";
proc freq data=ages;
table age; ②
table age / missing;
format age age2grp.;
run;
```

MISSING NUMERIC DATA - 'OTHER' ADDED TO FORMAT

age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
TEEN	4	23.53	4	23.53
MIDDLE AGE	6	35.29	10	58.82
SENIOR	7	41.18	17	100.00

Frequency Missing = 5

age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
TOO LOW/TOO HIGH	5	22.73	5	22.73
TEEN	4	18.18	9	40.91
MIDDLE AGE	6	27.27	15	68.18
SENIOR	7	31.82	22	100.00

The format AGE2GRP is changed ①. The 'low' and 'high' are removed from the range specifications, replaced by real values. An 'other' condition is added to account for all values that are not in the specified age ranges. Two tables are created by PROC FREQ ②. In the first, missing data are left out of the table, while in the second they are added to the table using the MISSING option. If you look at the top table on the right, it appears as if there are five observations with missing values for age, not the two that you can see in the DATALINES file in example 7.5. Why does PROC FREQ tell you that there are five missing values, not two? This is a result of what you can call the 'bad apple rule'. The format AGE2GRP instructs SAS to form a group comprising all ages that are not between 10 and 110 (the limits of the ranges in the format). Given the data in data set AGES, that includes the following ages: 0, 185, 199, and the two missing values. You have formed a group that includes some missing data and PROC FREQ treats all observations in that group as missing ... the missing data are 'bad apples' that have spoiled the entire group. When the MISSING option is added to the TABLE statement, you can see in the lower table that the missing data are in the group labeled 'TOO LOW/TOO LOW', grouped with real values that are too low or too high. There is a solution to this problem.

...Example 7.15...

```
proc format;
value age2grp
10  - 19   = "TEEN"
20  - 54   = "MIDDLE AGE"
55  - 110  = "SENIOR"
other      = 'TOO LOW/TOO HIGH'
.          = 'MISSING'  ①
;
run;
```

```
title 'MISSING NUMERIC DATA - ASSIGN A LABEL IN THE FORMAT TO MISSING VALUES';
proc freq data=ages;
table age;
format age age2grp.;
run;
```

The best way to handle missing data in a numeric format is to create a category that is exclusive to missing data ①. This separates missing values from other ranges and avoids the 'bad apple' behavior seen in example 7.6. The output from PROC FREQ on the right shows that you now appear to have only two missing values in your data, plus three that are too low or too high ... that's correct.

MISSING NUMERIC DATA - ASSIGN A LABEL IN THE FORMAT TO MISSING VALUES

age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
TOO LOW/TOO HIGH	3	15.00	3	15.00
TEEN	4	20.00	7	35.00
MIDDLE AGE	6	30.00	13	65.00
SENIOR	7	35.00	20	100.00

Frequency Missing = 2

Complicating the missing data problem is the fact that missing character data are treated differently by PROC FORMAT. Missing values are treated as real values, they just happen to blank.

...Example 7.`6...

```
proc format;
value $score ①
low -'6'   = 'NOT HEALTHY'
'7' - high = 'HEALTHY'
;
run;

data births;
input score $1.;
datalines; ②
1
X

3
7

9
6
;
run;

title 'MISSING CHARACTER DATA AND PROC FORMAT';
proc freq data=births;
table score; ③
table score / missing;
format score $score.;
run;
```

MISSING CHARACTER DATA AND PROC FORMAT

score	Frequency	Percent	Cumulative Frequency	Cumulative Percent
HEALTHY	3	100.00	3	100.00

Frequency Missing = 5

score	Frequency	Percent	Cumulative Frequency	Cumulative Percent
NOT HEALTHY	5	62.50	5	62.50
HEALTHY	3	37.50	8	100.00

The character format $SCORE contains only two ranges, one starting with 'low' and the other ending with 'high' ①. In example 7.1, you saw a similar set of ranges in a numeric format did NOT include missing values. The data set BIRTHS contains two observations with missing values for SCORE ②. Two tables are created using PROC FREQ, the first will ignore missing values while the second will include missing data since the MISSING option is used ③. If you look at the top table output on the above left, notice that it appears as if there are five observations with missing data, not the two you see in the DATALINES file. This is another instance of the 'bad apple rule' except this time is caused by the fact that the word 'low' in

a range in PROC FORMAT does include missing values in a character format. All observations grouped with those two missing values are treated as having missing values.

With a numeric, one solution to handling missing values was to give them their own label within a format. That solution can also be tried with a character format.

...Example 7.17...

```
proc format;
value $score
low -'6'    = 'NOT HEALTHY'
'7' - high = 'HEALTHY'
' '         = 'UNKNOWN' ①
;
run;

proc format;
value $score
'0' -'6'    = 'NOT HEALTHY' ②
'7' - high = 'HEALTHY'
' '         = 'UNKNOWN'
;
run;

title 'MISSING CHARACTER DATA - ASSIGN A LABEL IN THE FORMAT TO MISSING VALUES';
proc freq data=births;
table score;
format score $score.;  ③
run;
```

PROC FORMAT is used twice in this example. The LOG resulting from the first use is shown on the upper right. Missing values are given their own label ①. However, since 'low' is still used in a range and since 'low' includes missing character data, there is an error message that showing that ranges overlap. Should missing data be included in the range 'low-6' or in its own group? That error can be eliminated by changing the range on the group of values with the label 'NOT HEALTHY' from low-'6' to '0'-'6' ②. Now, missing values are only in one group. When the format is used in PROC FREQ ③, the output is correct and there only two observations with missing data.

```
150  proc format;
151  value $score
152  low -'6'    = 'NOT HEALTHY'
153  '7' - high = 'HEALTHY'
154  ' '         = 'UNKNOWN'
155  ;
ERROR: These two ranges overlap: LOW-6 and - (fuzz=0).
NOTE: The previous statement has been deleted.
156  run;
```

MISSING CHARACTER DATA - ASSIGN A LABEL IN THE FORMAT TO MISSING VALUES

score	Frequency	Percent	Cumulative Frequency	Cumulative Percent
NOT HEALTHY	3	50.00	3	50.00
HEALTHY	3	50.00	6	100.00

Frequency Missing = 2

In summary, there are some new rules about formats to add to those shown earlier in the chapter.

FORMATS AND MISSING DATA

- ***the LOW range value in a NUMERIC format DOES NOT INCLUDE includes MISSING values***
- ***the LOW range value in a CHARACTER format DOES INCLUDE includes MISSING values***
- ***any values grouped with missing data by a format are considered as missing data***