# 3 Redatam+ SP Process Module
## Samples for the Creation of Indicators

# Table of Contents

# I. Dictionary

The database dictionary (.dic file) is the main object from where the user can manipulate, analyze and browse the data contained in a Redatam database. The first thing to do in a session is to open a dictionary file from de main menu **File>Open>Dictionary**.

The right side of the dictionary window displays the hierarchical database structure as a directory-like tree of *entities*, each of which has individual elements or members, which are often geographical, like provinces, districts, or city blocks, but can also be individual persons, houses, schools, etc. The right-side box contains the list of *variables* describing the currently high-lighted entity on the left side. Click on another entity name to change the current entity. The dictionary can be laid out with the variables appearing to the right of the entity tree (by selecting **Layout Presentation>Vertical**) or below the entity tree (by selecting **Layout Presentation>Horizontal**).

Double click on a variable name (on the right side list) to open a window displaying the information about the variable. Double click on an entity name and the branches beneath this entity will collapsed or expanded.

# II. Programming: generating outputs

## THE BASIC COMMAND SET

R+SP has three levels of programming to get results from a database: a) Fast Results; b) Assisted Programming; and c) Free Editor. The first level is designed for you to get results using only the mouse, in a "point and click" fashion, requiring no programming skills at all. However, this level has some limitations, because it does not provide the functionality to define new derived variables. The second level is based on a Programming Assistant, which needs very little knowledge of the R+SP language, and builds full programs that use all the system's flexibility and power. The third level is for the programmers who are already familiar with the R+SP language, and will be more at home with this totally free format of programming. This level also provides an assisted wizard to verify the syntax and to write blocks of commands.

Each tool (Easy Frequency, Easy Crosstabs, etc.) is in fact a visual way to create a program in the REDATAM+SP language. The same result can be obtained through programming a command set. In a command set, the user tells REDATAM+SP to carry out processes to create new variables and produce specific tabulations and other results via a set of commands from the REDATAM+SP language.

The programs are stored in files with the extension .spc (Statistical Processor Commands). To create a new program, or open an existing program, choose the function File in the main menu.

Then you can select New for a new object, or Existing for an existing object. In each case (New or Existing), choose Command Set. In the case of a new program, the system opens directly the window for free editing. In the case of an existing program, the system opens a dialog box to navigate and select the desired program. To create a new program, or open a program, it is necessary to have an active database, in other

words, the user must already have opened a database dictionary. Otherwise, the File menu will not permit the user to select either New or Existing.

There are two ways of programming in R+SP:
a) using the Programming Assistant; or
b) writing the program directly in the Command Editor.

In reality, programs are often written using a mixture of these techniques. Programs can be written directly in the Command Editor. In addition, the Programming Assistants can be accessed by right clicking within this editor in such a way that they can be used to aid in the writing of the commands. A R+SP program can be created in one way (for example using Assistants) and later worked on in another way (by writing directly in the Command Editor).

The language has three basic commands:

**RUNDEF** to define the environment in which a process will occur, including the specific selection set to be used or/and the universe.

**DEFINE** to create new variables and indicate their nature and scope.

**TABLE** to indicate what the specific process and output should be.

Each of these main instructions can be qualified by subordinate clauses. For example, to DEFINE a new variable, the user can specify its nature via clauses including AS, TYPE, RANGE, VARLABEL, VALUELABEL, limit the cases to which it applies by a FOR clause (which is similar to an "if" in Redatam-Plus), and RECODE the values, QUANTIFY and COUNT elements, etc.

```
DEFINE housin.newvar
AS COUNT person
TYPE INTEGER
VARLABEL "Total persons in the house"
```

Similarly, the output can be specified with the TABLE command using AS <tabletype>, where tabletype may be FREQUENCY, AVERAGE, CROSSTABS, AREALIST, etc., OF a set of variables, and further specified by clauses beginning with FOR, etc.

```
TABLE tab1
AS FREQUENCY OF
housin.newvar
```

## Scheme

### A simple Program using RUNDEF and TABLE

Write a program to get the same results of frequencies, crosstabs and averages that are produced by the Wizards.

Procedure

1 Opening the Command Editor: To open the Command Editor screen (with a database already active), go to the main menu, select **File**, and then choose **New**.

2 Save: It is recommended to do a Save from time to time, before executing the program and when you finish your job. Click on the save icon in the top left-hand corner of the Command Editor and give the program a meaningful name, such as Prog1Freq. You can use also the save icon in the REDATAM+SP toolbar, or use CTRL + S.

3 RUNDEF, always the first command: A Program must start with a RUNDEF, specifying the Selection and other general filters, if needed. It is always the very first command, and it must be unique in the program. Redatam makes your job easier by writing a default RUNDEF command every time you start a new program. This default RUNDEF command can be edited through the RUNDEF Assistant. The other Assistants are the DEFINE Assistant, the TABLE Assistant and the EXECUTE Assistant.

4 Accessing the RUNDEF Assist: After opening the Command Editor, you can use the specific Assistants by means of the popup menu, with the right mouse button. Select the whole RUNDEF command by positioning the mouse before the R of RUNDEF, and holding the mouse left button, move over the two lines of the command until the end of the word ALL, and then press the right mouse button.

5 Fill in the spaces in the RUNDEF Assistant:

**Name**, accepts any text.
**Selection**, defines the area that you want to process. By default it will read all the database (ALL). If you want to use a specific selection, you should have already created the selection file (with extension .slw). Use the browse button find and select it.
**Universe**, optional. This clause defines the filter expression to select the cases to process.
Go back to the Command Editor with a click in the **OK button** in the bottom part of the Assistant screen, or use the Save button in the Toolbar.

**Specifying the output table** – the TABLE command. Generally, each result needs one separate TABLE command. So, the program will have one TABLE command for each set of frequencies, crosstabs and averages (FREQUENCY, CROSSTABS and AVERAGE).

6 Set the table command position in the program (TABLE): Click the mouse in the next free line after the RUNDEF command. If there is no free line, click the mouse at the end of the second line of the RUNDEF command and press the [ENTER] key, and then click the mouse in the new line. This will be the place where you will put the first TABLE command. Once you finish with the TABLE Assistant (see next step), the command will be inserted into the Command Editor in the same place where the cursor is—if you want, you can leave an empty line after the RUNDEF command to improve its readability.

7 Use the TABLE Assistant: with the Command Editor as the active window, that is, its title bar being a darker shade of blue, call the popup menu (mouse right button) and select the TABLE Assistant. In the first tab of the TABLE Assistant, the Table tab, the user chooses a name for the table and the kind of table they want, for example a frequency table (TABLE AS FREQUENCY). Select the Frequency tab and then select variables by clicking and dragging from the variable list contained in the dictionary window of your active database.

Demonstration

Compute the distribution of population by age and sex and the distribution of dwellings by type in the Nueva Miranda database: Open a new Commands Editor and type the following program:

```
RUNDEF program1
        SELECTION ALL

TABLE Freq1
        AS FREQUENCY
        OF PERSON.EDQUINQ, PERSON.SEX, HOUSIN.TYPHOU
```

Run the program and check the results

Notes:

1. The name of the program “program 1”could be any.
2. “Freq1” is the name of the table. It could be any user defined name.
3. Both the RUNDEF and TABLE require a name.
4. The clause AS is followed by the type of table required.
5. The clause OF is followed by the list of variable identifiers to tabulate.
6. A variable identifier is always built from the entity name (PERSON) followed by a dot immediately followed by the variable name (EDQUINQ). NO BLANK (space) must appear in the variable (PERSON.EDQUINQ).
7. A command set must include one (**and only once**) a RUNDEF command, one DEFINE command for each derived variable (optionally and could be more than one) and one TABLE command for each output required (could be more than one TABLE command). Several table commands may appear in the same command set.
8. The RUNDEF/SELECTION clause allows you to select a different geographical area to process.
9. Optionally use the Assists windows to build the table command.

Exercises

1. Create a frequency table of the variables OWNERS, ROOF, TROOMS, and TOILCO.
2. Create a frequency of any PERSON variable.
3. Compute the sex distribution by type of family (1).
4. Compute the sex distribution by kinship relationship
5. Compute the sex distribution by kinship relationship and marital status.
6. Compute the average age by sex and marital status. Comment on the results.

# III. INDICATORS

The examples in this document were written originally by Serge Poulard, as part of a tutorial in a workshop with the Barbados Statistical Service following the 2002 Census.

They were later adapted to become a set of REDATAM programs applying to the NMIR database, the English version, whose dictionary is NMIR_Eng.dic.

For the sake of simplicity, and to avoid having to change directory and file directions, we assume that:

1. This database was installed in the C:\Program Files\Redatam\NMIR\ directory,
2. The dictionary is at C:\Program Files\Redatam\NMIR\BaseR\NMIR_Eng.dic
3. The database itself is at C:\Program Files\Redatam\NMIR\BaseR\

## SECTION 1: COUNTING SELECTED INDIVIDUALS OF A HOUSEHOLD

Program description:
A household variable is derived by counting the number of persons (entity PERSON) complying with a given criteria. In the example, the program counts the number of heads of the household (PERSON.RELAT=1). A simple frequency of the variable displays the results.

Note:
This program also checks the consistency of the database. There should be one person and only one person declared as head of household.

```
RUNDEF Section01
     SELECTION ALL

DEFINE HOUSIN.NHEADS AS COUNT PERSON
  FOR PERSON.RELAT = 1
  TYPE INTEGER

TABLE VERIF AS FREQUENCY OF HOUSIN.NHEADS
```

Programming Notes:
1. In the RUNDEF command, the clause SELECTION ALL might be omitted since it is ALL by default.
2. In the DEFINE command, the clause TYPE INTEGER might also be omitted, since TYPE INTEGER is the default.
3. The commands could have been written in a single line, such as (it is just a matter of programming style):
RUNDEF Section01 selection all define HOUSIN.NHEADS as count PERSON for PERSON.RELAT=1 type integer table t1 as frequency of HOUSIN.NHEADS
4. Although we recommend to separate the commands (RUNDEF, DEFINE and TABLE) in different lines
5. As you already have guessed, you can use lower or capital letters in all the programming clauses, BUT for the entities and variables names, which MUST be the same way they were defined in the dictionary. So, PERSON.RELAT is not the same as person.relat, or PERSON.relat
6. There must be at least one blank to separate the clauses of a command.

7. Blanks might be omitted in the logical (or arithmetic) expressions.

PERSON.RELAT = 1 is the same as PERSON.RELAT=1

8. Comments can be anywhere in a program. They are written as a block, starting with /* and finishing with */ or comments which only occupy a single line can be written using a double slash //, such as: //this is a comment

Results:

| Categories | Counts |
|---|---|
| 0 | 2544 |
| 1 | 11375 |
| Total | 13919 |

Comments:
1. There are no households with more than 1 head.
2. However, there are households with no heads, which could mean that they are collective households, or another explanation that we will leave for the time being.

## SECTION 2: COUNTING SELECTED INDIVIDUALS OF A HOUSEHOLD

Program description:
A household variable is derived by counting the number of spouses of the head of the household (PERSON.RELAT=2). A simple frequency of the variable displays the results.

Note:
This program also checks the consistency of the database. The head of the household should have either 0 (zero) or no more than one spouse.

```
RUNDEF Section02

DEFINE HOUSIN.NSPOUSES AS COUNT PERSON
  FOR PERSON.RELAT = 2

TABLE VERIF AS FREQUENCY OF HOUSIN.NSPOUSES
```

Results:

| Categories | Counts |
|---|---|
| 0 | 7009 |
| 1 | 6910 |
| Total | 13919 |

Comments:
1. There are no households with more than 1 spouse.
2. However, it might be the case where there exists a spouse but no head in the household. In order to check that, see program in Section03.

## Section 3: CHECKING THE EXISTENCY OF SPOUSES BUT NO HEADS

Program description:
The example uses the two defines of the previous examples to count heads and spouses. A crosstabs of the variables might display the results.

```
RUNDEF Section03

DEFINE HOUSIN.NHEADS AS COUNT PERSON
  FOR PERSON.RELAT = 1

DEFINE HOUSIN.NSPOUSES AS COUNT PERSON
  FOR PERSON.RELAT = 2
  RANGE 0-10

TABLE VERIF2 AS CROSSTABS OF HOUSIN.NHEADS BY HOUSIN.NSPOUSES
```

Programming notes:
1. An arbitrary range of 0-10 has been defined for the variable. The range will be necessary for the "column" variable in the table VERIF2. It is possible to generate a FREQUENCY of a variable without any range. However, in a Crosstabs (two-way tabulation), only the first variable may be without range.

2. The clauses FREQUENCY and CROSSTABS can be used interchangeably, which means that what really defines if the result is a frequency or a cross tabulation is the number of BY (at most 4)

Results:

| NHEADS by NSPOUSES | 0 | 1 | Total |
|---|---|---|---|
| 0 | 2544 | 0 | 2544 |
| 1 | 4465 | 6910 | 11375 |
| Total | 7009 | 6910 | 13919 |

Comments:
1. There are no households with no heads and 1 spouse.

## Section 4: THE COUNTY WITH THE HIGHEST PROPORTION OF CHILDREN (0-5)

```
RUNDEF Section04

DEFINE COUNTY.TOTPOP AS COUNT PERSON

DEFINE COUNTY.TOTKIDS AS COUNT PERSON
  FOR PERSON.AGE < 6

DEFINE COUNTY.PROPKIDS
  AS 100 * (COUNTY.TOTKIDS / COUNTY.TOTPOP)
  FOR COUNTY.TOTPOP > 0
  TYPE REAL

TABLE VERIF AS AREALIST OF COUNTY,
  COUNTY.NCOUNTY, COUNTY.TOTPOP, COUNTY.TOTKIDS, COUNTY.PROPKIDS
```

Programming notes:

1. The required variables are not readily available in the database; they must be calculated within the command set. The total population is calculated with a COUNT (see the definition of variable COUNTY.TOTPOP) and the number of children is calculated in a similar way applying a filter on the individuals to be counted (see the FOR clause in the definition of variable COUNTY.TOTKIDS).
2. The proportion (or percentage) is calculated as the ratio of the number of children aged 0-5 divided by the total population, normalized to 100 (see the definition of the variable COUNTY.PROPKIDS).
3. The number of decimals is established in Preferences, or defined directly as a clause in the TABLE command, such as

```
TABLE .....
  DECIMALS 3
```

4. The list is produced with the command TABLE/AREALIST. Please note the AS and OF clauses MUST be included in the command. Furthermore, the first parameter of the OF clause is the output level (a selectable Entity name) of the AREALIST (COUNTY in the example).
5. To enhance the table, the list shows also the name of each County (variable COUNTY.NCOUNTY)

Results:

| Code | County Name | TOTPOP | TOTKIDS | PROPKIDS |
|---|---|---|---|---|
| 5 | Santa Maria | 21728 | 2525 | 11.62 |
| 6 | Santiago | 8969 | 1606 | 17.91 |
| 7 | Bolivar | 14281 | 2050 | 14.36 |
| 8 | Marbella | 3818 | 624 | 16.34 |
| 9 | Puerto Nuevo | 3393 | 552 | 16.27 |

Comments:

1. Searching the list visually, the highest would be the County code 6 (Santiago).
2. If the list is big, you can see it sorted by double clicking in the results entry in the output tree, then click in PROPKIDs column, and then click in the Z-A button in the toolbar.

# Section 5: PROMOTING THE VALUE OF A SELECTED INDIVIDUAL SAVING PERMANENTLY A VARIABLE IN THE DATABASE

Program description:
This program selects the age of the household head and save it as a household variable. This operation is valid since only one person will qualify (the head of household is unique). It illustrates the way a person variable may be "promoted" to the household level.

A simple frequency can be used to check households having heads of an age lesser than the acceptable limit.

```
RUNDEF Section05

DEFINE HOUSIN.AGEHEAD AS PERSON.AGE
  FOR PERSON.RELAT = 1
  VARLABEL "Age of Head of Household"
  RANGE 0 - 100
  SAVE "C:\Program Files\Redatam\NMIR\BASER\Housin_AGEHEAD.rbf" OVERWRITE

TABLE VERIF AS FREQUENCY OF HOUSIN.AGEHEAD
  FOR HOUSIN.AGEHEAD <= 15
```

Programming notes:
1. The DEFINE command includes the SAVE clause that triggers the permanent addition of the variable to the database. The file name containing the data must include the full path, otherwise it will be saved in the working directory defined in the database dictionary or in Preferences. The example assumes the Redatam installation default.
2. Only the low ages (<=15) are tabulated. The clause FOR HOUSIN.AGEHEAD <= 15 restricts the tabulation for those head of household lesser than or equal to 15 years of age.
3. The RANGE clause must be specified. A variable cannot be saved without this clause. When the range of the variable to be saved is not known a priori, a dry run (without the SAVE clause) can be performed with a non-restricted tabulation (no FOR clause) in order to establish the range of the variable.
4. There can be no SELECTION file when a variable is saved in the database, that is, SELECTION ALL must be selected (which is the default).
5. No filters can be applied in the RUNDEF command when saving a variable.
6. The VARLABEL clause specifies the label for the new variable.
7. The OVERWRITE parameter in the SAVE clause allows the output file (Housin_AGEHEAD.rbf) to be rewritten.

Results:

| Categories | Counts |
|---|---|
| 15 | 171 |
| Total | 171 |

Comments:
1. After executing the program, the updated dictionary shows a new variable (AGEHEAD) in the HOUSIN entity. The dictionary must be saved in order for this change to be permanent.
2. When, for any reason, the variable must be recreated, the variable name should be deleted from the dictionary prior to executing the program. Remember that there are two different things, the variable name in

the dictionary (which must be deleted), and the file storing the variable data, which can be written automatically using the OVERWRITE clause.
3. It is good programming practice to name the file using the entity name and the variable name, separated by an underscore. The file extension is unnecessary, although it tells us that this is a Redatam Binary File (.rbf).
4. It is useful to run an Easy frequency in the newly created variable. By doing this you will see that there are 2544 households with a Not applicable value for the AGEHEAD. They correspond to the households having no person being the household head (see Section01). If you wanted to, you could have used the DEFAULT 0 (or any other value) clause in the DEFINE command, to establish the values for those not applicable cases.

## Section 6: DERIVING A VARIABLE USING AN ARITHMETIC EXPRESSION

Program description:
The program calculates the difference in age between the child of household head and his declared father. It also performs a consistency check: the difference between child and parent should be at least 15 years.

```
RUNDEF Section06

DEFINE PERSON.DIFAGENE AS HOUSIN.AGEHEAD - PERSON.AGE
  FOR PERSON.RELAT = 4
  VARLABEL "Number of years between child and parent"

TABLE VERIF AS FREQUENCY OF PERSON.DIFAGENE
  FOR PERSON.DIFAGENE < 15
```

Programming notes:
1. The program requires the age of household head (HOUSIN.AGEHEAD). This variable must have been calculated previously and stored permanently in the database (see Section05).
2. The tabulation has been restricted to the differences lesser that 15, using the FOR clause. The clause may be suppressed to find the maximum difference of age between the head of the household and his youngest child.

Results:

| Categories | Counts |
|---|---|
| 3 | 1 |
| 11 | 1 |
| 12 | 2 |
| 13 | 10 |
| 14 | 16 |
| -69 | 1 |
| Total | 31 |

Comments:
1. The table suggests that 30 persons have a difference lesser than 15 to the head of the household, and one person has a negative difference (-69), meaning that the child is older than his father (or mother).

2. Of course this is just an example, since in the real situation, the relationship "son/daugther" could define also adopted children, or the head might me the stepfather.
3. Negative values could have been omitted using the range clause (RANGE 0-100), for example.

## Section 7: DERIVING A VARIABLE USING THE RECODE FUNCTION

Program description:
This program illustrates the derivation of a variable using the RECODE technique. The age of individuals are recoded into 3 categories, children, adults and elders. The categories reflect a standard practice for age regrouping: 0 – 14, Children; 15 64, Adults; 64+, Elders.

```
RUNDEF Section07
    COMPLETENAME

DEFINE PERSON.GRP3 AS RECODE PERSON.AGE
   (0 - 14=1)  (15 - 64=2)  (65 - HIGHEST =3)
   RANGE 1 - 3
   VARLABEL "Age groups 3"
   VALUELABELS 1 "Children" 2 "Adults" 3 "Elders"

TABLE VERIF AS FREQUENCY OF PERSON.GRP3
```

Programming notes:
1. Each recode item is specified in between parenthesis.
2. The RECODE last range is specified by the keyword HIGHEST to insure the highest value of the variable AGE is included. Similarly the keyword “LOWEST” might be used for specifying the lowest value of the database.
3. The ELSE clause could also be used to specify the last range, as in (0 - 14=1) (15 - 64=2) else 3 but it would comprise ALL the values not defined in previous items.
4. The VALUELABELS clause documents the 3 categories. The entries are specified by the category value immediately followed by the category label between quotes (and at least a blank between them).
5. The COMPLETENAME clause in the RUNDEF command forces both the category value and label to be shown i.e. “1. Children” (without this clause only the label “Children” would be shown). This clause could also be placed directly in the TABLE command with identical results.

Results:

| Categories | Counts | % |
|---|---|---|
| 1. Children | 17965 | 34.4 |
| 2. Adults | 30886 | 59.2 |
| 3. Elders | 3338 | 6.4 |
| Total | 52189 | 100 |

Comments:
1. The adults represent the 59.2% of the population.

# Section 8: DERIVING A BINARY VARIABLE USING A LOGICAL EXPRESSION

Program description:
The program derives a binary variable (a variable having only 2 possible values). The values are assimilated to true or false, yes or no etc… In the syntax of REDATAM, a logical expression within or in place of an arithmetic expression is evaluated to 0 (if false) or to 1 (if true). For example the expression (PERSON.AGE < 15) will be evaluated to INTEGER 1 if the expression is true and INTEGER 0 if it is false. This feature is very useful for evaluating a set of conditions based on an original variable. It is a special case of RECODING when the new variable has only two categories.

The following example derives a binary variable from the water origin of the household. If the origin is "public network" the variable value will be 1, otherwise 0.

```
RUNDEF Section08

DEFINE HOUSIN.PUBL AS HOUSIN.WATERO = 1
VARLABEL "Water availability by Public network"
  TYPE BOOL

TABLE VERIF1 AS FREQUENCY OF HOUSIN.PUBL
TABLE VERIF2 AS FREQUENCY OF HOUSIN.WATERO BY HOUSIN.PUBL
```

Programming notes:
1. By defining the new variable as type BOOL, the system automatically establishes the range of 0-1 and the valuelabels of False and True.
2. The second table is presented to show how the categories of the original variable are transformed to the new one. Only the line "public network" becomes "true".

Results:

Table VERIF1 - Simple frequency

| Categories | Counts | % |
|---|---|---|
| False | 6354 | 45.6 |
| True | 7565 | 54.4 |
| Total | 13919 | 100.0 |

TABLE VERIF2 - Crosstabs
Water Origin  Water availability by Public network

| | False | True | Total |
|---|---|---|---|
| No response | 2544 | - | 2544 |
| Public network | - | 7565 | 7565 |
| Well or noria | 2124 | - | 2124 |
| River creek | 1467 | - | 1467 |
| Other | 219 | - | 219 |
| Total | 6354 | 7565 | 13919 |

Comments:

1. According to the first table, 54.4% of the households are connected to the public network, but this result is misleading, since, as you can see in the second table, there are 2544 cases that do not respond to this question (collective households, unoccupied households, and others). So, to be exact, they should be filtered from the define, as

```
DEFINE HOUSIN.PUBL1 AS HOUSIN.WATERO = 1
...
FOR HOUSIN.WATERO <> 0
```

2. By doing that the percentage of serviced households would increase to 66.5%.

## Section 9 CREATION OF A PERMANENT VARIABLE (Number of children)

Program description:
The program creates a variable at the household level (Number of Children in Household). It is obtained by counting the number of children of the household, a child being defined as a person of age less than 15 (PERSON.AGE < 15) and declared as the child of the head of household (PERSON.RELAT = 4).

```
RUNDEF Section09

DEFINE HOUSIN.NCHILDREN AS COUNT PERSON
  FOR PERSON.RELAT = 4 AND PERSON.AGE < 15
  VARLABEL "Number of Children in Household"
  RANGE 0 - 20
  SAVE "C:\Program Files\Redatam\NMIR\BASER\HOUSIN_NCHILDREN.rbf"
  OVERWRITE

TABLE VERIF AS FREQUENCY OF HOUSIN.NCHILDREN
```

Programming notes:
1. There is no difference between this and the program in Section05, but we need to create this variable to be used in later examples.

Results:

| Categories | Counts |
|---|---|
| 0 | 7936 |
| 1 | 2137 |
| 2 | 1944 |
| 3 | 1103 |
| 4 | 489 |
| 5 | 207 |
| 6 | 51 |
| 7 | 34 |
| 8 | 10 |
| 9 | 4 |
| 10 | 4 |
| Total | 13919 |

Comments:
1. 7936 households have no children, and 4 households have 10 children.
2. Do not forget to save the dictionary after executing the program.

## Section 10 CREATION OF A PERMANENT VARIABLE

Program description: Same as Section09, but for Sex of household head.

```
RUNDEF Section10

DEFINE HOUSIN.SEXHEAD AS PERSON.SEX
  FOR PERSON.RELAT = 1
  VARLABEL "Sex of Head of Household"
  LIKE PERSON.SEX
  SAVE "C:\Program Files\Redatam\NMIR\BASER\HOUSIN_SEXHEAD.rbf" OVERWRITE

TABLE VERIF AS FREQUENCY OF HOUSIN.SEXHEAD
```

Programming notes:
1. The LIKE clause copies all the attributes of the "liked" variable to the new one, with the exception of the ones explicitly defined (VARLABEL in this case). (Note: In the Technical Specifications tab of the Properties Dialogue box for the variable PERSON.SEX, you may need to change the Missing Value to 3. This dialogue box is accessed by right clicking on the variable in the dictionary window and selecting Properties).

Results:

| Categories | Counts | % |
|---|---|---|
| Male | 8975 | 78.9 |
| Female | 2400 | 21.1 |
| Total | 11375 | 100 |

NotApp : 2544

Comments:
1. There are 2544 not applicable values, corresponding to the same households that have no head.
2. In New Miranda, roughly 80% of the households are headed by a male.
3. Do not forget to save the dictionary after executing the program.

## Section 11: CREATION OF A PERMANENT VARIABLE (Household head has partner)

Program description:
This program establishes if the head of the household has either a spouse or is in a consenting union. The new variable has two categories: 0 if the head has no partner or 1 if the head has a partner. Verification of results is obtained by 2 tables.

```
RUNDEF Section11
DEFINE HOUSIN.HASSPOUSE AS COUNT PERSON
  FOR PERSON.RELAT = 2
  RANGE 0-1

DEFINE HOUSIN.HASCMLAW AS COUNT PERSON
  FOR PERSON.RELAT = 3
  TYPE INTEGER
  RANGE 0-1

DEFINE HOUSIN.HASPARTNER
  AS ( HOUSIN.HASSPOUSE <> 0 ) OR ( HOUSIN.HASCMLAW <> 0 )
  VARLABEL "Head has a Partner"
  TYPE BOOL
  SAVE "C:\Program Files\Redatam\NMIR\BASER\HOUSIN_HEADHASPATNER.rbf" OVERWRITE

TABLE VERIF1 AS FREQUENCY OF HOUSIN.HASCMLAW BY HOUSIN.HASSPOUSE
TABLE VERIF2 AS FREQUENCY OF HOUSIN.HASPARTNER
```

Programming notes:
1. The program uses two intermediate variables (HOUSIN.HASSPOUSE and HOUSIN.HASCMLAW) that respectively count the number of individuals with variable PERSON.RELAT = 2 (count of spouses) and PERSON.RELAT = 3 (count of consenting union partners).
2. The final variable is obtained using a logical expression (HOUSIN.HASSPOUSE <> 0) OR (HOUSIN.HASCMLAW <> 0) that is false (0) when no spouse or consenting union partner are part of the household.

Results:

Table Verif1

HASCMLAW by HASSPOUSE

| | 0 | 1 | Total |
|---|---|---|---|
| 0 | 6372 | 6910 | 13282 |
| 1 | 637 | - | 637 |
| Total | 7009 | 6910 | 13919 |

Table Verif2

| Categories | Counts | % |
|---|---|---|
| False | 6372 | 45.8 |
| True | 7547 | 54.2 |
| Total | 13919 | 100.0 |

Comments:
1. The first table shows that 6372 households are single headed (no partner). This figure is confirmed by the second table (category False of the variable), showing also the percentage of single headed households (45.8%).
2. Again, this figure is misleading, since we did not take into account the number of households with no heads. By using a filter in the second table to select households with heads we will get a different result. The filter could be the sex of the household created in Section10. So, the second table should be

```
TABLE VERIF3 AS FREQUENCY OF HOUSIN.HASPARTNER
   for HOUSIN.SEXHEAD = 1 or HOUSIN.SEXHEAD = 2
```

3. Using this filter the percentage of single headed households drops to 33.7%
4. Do not forget to save the dictionary after executing the program.

## Section 12: DERIVING A VARIABLE USING THE SWITCH FUNCTION (Type of household)

Program description:
The use of SWITCH function is recommended when the definition of a variable is based on several other variables. While more complicated to implement, use of this technique is more likely to prevent the introduction of errors. The SWITCH function may be seen as the implementation of a decision tree.

The following example illustrates the classification of a household based on the number of children in the household, the sex of the head and the existence or not of a spouse or partner in a consenting union with the head. It implements the following decision tree:

- The household is not single headed
  - The household does not include children — Type 1
  - The household includes children — Type 2
- The household is single headed
  - The household does not include children
    - The head is male — Type 3
    - The head is female — Type 4
  - The household does include children
    - The head is male — Type 5
    - The head is female — Type 6

Please, note that this program uses derived variables permanently saved in the database by previous sections.

```
RUNDEF Section12

DEFINE HOUSIN.HHTYPE AS SWITCH
    INCASE HOUSIN.HASPARTNER = 1 AND HOUSIN.NCHILDREN = 0
```

```
    ASSIGN 1
  INCASE HOUSIN.HASPARTNER = 1 AND HOUSIN.NCHILDREN > 0
    ASSIGN 2
  INCASE HOUSIN.HASPARTNER = 0 AND HOUSIN.SEXHEAD = 1 AND HOUSIN.NCHILDREN = 0
    ASSIGN 3
  INCASE HOUSIN.HASPARTNER = 0 AND HOUSIN.SEXHEAD = 2 AND HOUSIN.NCHILDREN = 0
    ASSIGN 4
  INCASE HOUSIN.HASPARTNER = 0 AND HOUSIN.SEXHEAD = 1 AND HOUSIN.NCHILDREN > 0
    ASSIGN 5
  INCASE HOUSIN.HASPARTNER = 0 AND HOUSIN.SEXHEAD = 2 AND HOUSIN.NCHILDREN > 0
    ASSIGN 6
VARLABEL "Head Partner"
VALUELABELS
1 "Household with Children"
2 "Household without Children"
3 "Single Headed by Male no Children"
4 "Single Headed by Female no Children"
5 "Single Headed by Male with Children"
6 "Single Headed by Female with Children"
RANGE 1-6
SAVE "C:\Program Files\Redatam\NMIR\BASER\HOUSIN_HOUSEHOLDTYPE.rbf" OVERWRITE

TABLE VERIF AS FREQUENCY OF HOUSIN.HHTYPE
```

Programming notes:

1. It executes in pairs of INCASE and ASSIGN instructions. The INCASE defines the condition for the next ASSIGN to be executed.
2. The ASSIGN could have been written in the same line as the previous INCASE.
3. As in the RECODE type of defining, the SWITCH accepts an ELSE instruction, to cope with all the cases not stated by previous INCASEs.
4. As in the RECODE, values that do not fall into any INCASE expressions, will stay the same, provided there is no ELSE expression.

Results:

| Categories | Counts | % |
|---|---|---|
| Household with Children | 2459 | 21.6 |
| Household without Children | 5088 | 44.7 |
| Single Headed by Male no Children | 1504 | 13.2 |
| Single Headed by Female no Children | 1429 | 12.6 |
| Single Headed by Male with Children | 104 | 9.1 |
| Single Headed by Female with Children | 791 | 7.0 |
| Total | 11375 | 100.0 |

Missing : 2544

Comments:

1. Households having no heads will receive a Missing value.
2. Do not forget to save the dictionary after executing the program.

## Section 13: CREATION OF PERMANENT VARIABLE (Partner of household head)

Program description:
This program creates a variable at household level to state if the household head has no spouse (category 0), a spouse (category 1) or a partner in a consenting union (category 2).

```
RUNDEF Section13

DEFINE HOUSIN.HASSPOUSE
  AS COUNT PERSON
  FOR PERSON.RELAT = 2
  RANGE 0-1

DEFINE HOUSIN.HASCMLAW
  AS COUNT PERSON
  FOR PERSON.RELAT = 3
  RANGE 0-1

DEFINE HOUSIN.PARTNER AS SWITCH
    INCASE (HOUSIN.HASSPOUSE <> 0) ASSIGN 1
    INCASE (HOUSIN.HASCMLAW <> 0) ASSIGN 2
    ELSE 0
    VARLABEL "Head Partner"
    VALUELABELS
    0 "No Partner"
    1 "Spouse"
    2 "Common Law"
    RANGE 0-2
    SAVE "C:\Program Files\Redatam\NMIR\BASER\HOUSIN_HEADPARTNER.rbf" OVERWRITE

TABLE VERIF AS FREQUENCY OF HOUSIN.PARTNER
```

Programming notes:
1. The define uses an ELSE instruction for the follow thru cases (the ones that were not filtered by any previous INCASEs)

Results:

| Categories | Counts | % |
|---|---|---|
| No Partner | 6372 | 45.8 |
| Spouse | 6910 | 49.6 |
| Common Law | 637 | 4.6 |
| Total | 13919 | 100.0 |

Comments:
1. The results can be compared with a frequency of the variable HOUSIN.HASPARTNER.
2. Do not forget to save the dictionary after executing the program.

## Section14: CREATION OF AN INDICATOR (Transportation availability)

Program description:
This program creates a variable at the household level (entity HOUSIN) that contains the number of transportation types available in the household. This number varies from 0 to 3, based on the category 1 of variables BICYCL, CAR and MCYCLE.

```
RUNDEF Section14

DEFINE HOUSIN.TRANSP
  AS HOUSIN.BICYCL + HOUSIN.CAR + HOUSIN.MCYCLE
  VARLABEL "Number of available transportation types"

TABLE VERIF AS FREQUENCY OF HOUSIN.TRANSP
```

Programming notes:
1. The direct sum of the variables is possible because none of them have the not applicable or the missing categories. If that was not the case, we would need to define an intermediate variable to eliminate those values, otherwise the arithmetic sum would become invalid as well.

```
  Define HOUSIN.V1 as HOUSIN.BICYCL
     Default 0
```

Results:

| Categories | Counts | % |
|---|---|---|
| 0 | 8966 | 64.4 |
| 1 | 4067 | 29.2 |
| 2 | 852 | 6.1 |
| 3 | 34 | 0.2 |
| Total | 13919 | 100.0 |

Comments:
1. Almost 65% of the households have no private transportation.

## Section 15: CREATION OF AN INDICATOR (Household transportation by County)

Program description:
This program creates a variable for each type of transportation at County level, by counting the number of households declaring the transport.

```
RUNDEF Section15

DEFINE COUNTY.BICYCL AS COUNT HOUSIN
  FOR HOUSIN.BICYCL = 1
  VARLABEL "Houses having bicycles"
```

```
DEFINE COUNTY.CAR AS COUNT HOUSIN
  FOR HOUSIN.CAR = 1
  VARLABEL "Houses having cars"

DEFINE COUNTY.MCYCLE AS COUNT HOUSIN
  FOR HOUSIN.MCYCLE = 1
  VARLABEL "Houses having motorcycles"

DEFINE COUNTY.NHOUSES AS COUNT HOUSIN
  VARLABEL "Total houses"

TABLE VERIF AS AREALIST OF COUNTY, COUNTY.NCOUNTY,
  COUNTY.BICYCL, COUNTY.CAR, COUNTY.MCYCLE, COUNTY.NHOUSES
  TOTAL
```

Programming notes:
1. The TOTAL clause produces a total line in the AREALIST

Results:

| Code | County Name | Houses h/bicycles | Houses h/cars | Houses h/motorcycles | Total houses |
|---|---|---|---|---|---|
| 5 | Santa Maria | 1926 | 747 | 85 | 5311 |
| 6 | Santiago | 627 | 126 | 14 | 1975 |
| 7 | Bolivar | 1087 | 375 | 24 | 3299 |
| 8 | Marbella | 342 | 142 | 26 | 1959 |
| 9 | Puerto Nuevo | 270 | 74 | 8 | 1375 |
| | TOTAL | 4252 | 1464 | 157 | 13919 |

Comments:

1. Which county has, proportionally, more households with cars? Just divide the second column by the fourth. Or see the next Section16.

## Section 16: CREATION OF AN INDICATOR (Proportion of Households with cars by County)

Program description:
This program creates a variable for car availability at County level by counting the number of households declaring to have cars, and dividing by the number of households in the County.

```
RUNDEF Section16

DEFINE COUNTY.NHOUSES AS COUNT HOUSIN
   VARLABEL "Total Houses"

DEFINE COUNTY.NCARS AS COUNT HOUSIN
  FOR HOUSIN.CAR = 1
  VARLABEL "Houses having cars"

DEFINE COUNTY.PERC
```

```
  AS 100 * (COUNTY.NCARS / COUNTY.NHOUSES)
  TYPE REAL
  VARLABEL "Percent having cars"

TABLE VERIF AS AREALIST OF COUNTY, COUNTY.NCOUNTY,
  COUNTY.NCARS, COUNTY.NHOUSES, COUNTY.PERC
```

Programming notes:
1. The TOTAL option should not be used here, since there is a percentage (last column) that cannot be summed. It might be used if the value obtained in the last column is disregarded.

Results:

| Cod | County Name | Houses h/cars | Total Houses | Percent having cars |
|---|---|---|---|---|
| 5 | Santa Maria | 747 | 5311 | 14.1 |
| 6 | Santiago | 126 | 1975 | 6.4 |
| 7 | Bolivar | 375 | 3299 | 11.4 |
| 8 | Marbella | 142 | 1959 | 7.2 |
| 9 | Puerto Nuevo | 74 | 1375 | 5.4 |

Comments:
1. The County better served by this indicator is Santa María.

## Section 17: DERIVING AGE GROUPS

Program description:
This program demonstrates the creation of an age group variable using a RECODE function. The example includes the age grouping in 4 categories.

```
RUNDEF Section17

DEFINE PERSON.AGEGROUP4 AS RECODE PERSON.AGE
   (0 - 14=1) (15 - 24=2) (25 - 64=3) (65 - HIGHEST =4)
   RANGE 1-4
   VARLABEL "Age groups 3"
   VALUELABELS 1 "Children" 2 "Young Adults" 3 "Adults" 4 "Elders"
   SAVE "C:\Program Files\Redatam\NMIR\BASER\PERSON_AGEGROUP4.rbf" OVERWRITE

TABLE VERIF AS FREQUENCY OF PERSON.AGEGROUP4
```

Programming notes:
1. A useful classification is the 4 way grouping. It classifies the population into children (age 0-14), young adults (15-24), adults (25-64) and elders (64+). This categorization is used to calculate the dependency ratio based on age.

Results:

| Categories | Counts | % |
|---|---|---|
| Children | 7965 | 34.4 |
| Young Adults | 1200 | 21.5 |
| Adults | 9686 | 37.7 |
| Elders | 3338 | 6.4 |
| Total | 52189 | 100.0 |

Comments:
1. Do not forget to save the dictionary after executing the program.

## Section 18: CALCULATING A SEX RATIO BY AREA

Program description:
This program calculates the sex ratio per area. The sex ratio is established as the ratio of the number of males divided by the number of females in each area.

```
RUNDEF Section18

DEFINE COUNTY.NMALES
  AS COUNT PERSON
  FOR PERSON.SEX = 1
  VARLABEL "# of Males"

DEFINE COUNTY.NFEMALES
  AS COUNT PERSON
  FOR PERSON.SEX = 2
  VARLABEL "# of Females"

DEFINE COUNTY.SEXRATIO AS COUNTY.NMALES / COUNTY.NFEMALES
  TYPE REAL
  FOR COUNTY.NFEMALES > 0
  VARLABEL "Sex Ratio per County"
  DECIMALS 2

TABLE VERIF AS AREALIST OF COUNTY, COUNTY.NCOUNTY,
  COUNTY.NMALES, COUNTY.NFEMALES, COUNTY.SEXRATIO
  TITLE "Sex Ratio in New Miranda"
  DECIMALS 2
```

Programming notes:
1. The clause FOR COUNTY.NFEMALES > 0 guarantees no division by zero is to occur. Even if the precaution is not necessary in this obvious case, it is a good practice to include such a clause in every division.
2. When a variable is included in an AREALIST, no range needs to be specified in the variable definition.
3. A title clause can be used to identify the list.

Results:

| Cod | County Name | # of Males | # of Females | Sex Ratio per County |
|---|---|---|---|---|
| 5 | Santa Maria | 10767 | 10961 | 0.98 |
| 6 | Santiago | 4931 | 4038 | 1.22 |
| 7 | Bolivar | 7300 | 6981 | 1.05 |
| 8 | Marbella | 2004 | 1814 | 1.10 |
| 9 | Puerto Nuevo | 1832 | 1561 | 1.17 |

Comments:
1. The highest ratios are encountered in Santiago and Puerto Nuevo

## Section 19: CALCULATING THE AVERAGE NUMBER OF PERSONS PER HOUSEHOLD

Program description:
This program produces a list by area of the average number of persons per household.

```
RUNDEF Section19
  FOR HOUSIN.COLPRI = 1

DEFINE COUNTY.NPERS AS COUNT PERSON

DEFINE COUNTY.NHH AS COUNT HOUSIN

DEFINE COUNTY.RATIO AS COUNTY.NPERS / COUNTY.NHH
 TYPE REAL
 FOR COUNTY.NHH <> 0
 DECIMALS 2

TABLE VERIF AS AREALIST OF COUNTY, COUNTY.NCOUNTY,
 COUNTY.NPERS, COUNTY.NHH, COUNTY.RATIO
 DECIMALS 2
```

Programming notes:
1. This type of indicator makes sense only for private households, and that is the reason for the filter in the RUNDEF command.

Results:

| Cod | County Name | NPERS | NHH | RATIO |
|---|---|---|---|---|
| 5 | Santa Maria | 21432 | 5290 | 4.05 |
| 6 | Santiago | 8581 | 1959 | 4.38 |
| 7 | Bolivar | 14135 | 3285 | 4.30 |
| 8 | Marbella | 3771 | 1941 | 1.94 |
| 9 | Puerto Nuevo | 3376 | 1366 | 2.47 |

Comments:
1. The lowest ratio is in Marbella.

# Section 20: CALCULATING AN INDEX ON BUILDING CONDITIONS

Program description:
This program establishes a score for each living accommodation. The score is defined as the arithmetic sum of 6 binary variables that characterize the physical conditions of the building: water origin, cooking fuel, toilet connection, and types of walls, floor and roof.

The choice of the variables is arbitrary and may be adapted to more realistic conditions. Each binary variable reflects a good condition by a 1 value.

```
RUNDEF Section20
   FOR HOUSIN.SEXHEAD = 1 OR HOUSIN.SEXHEAD = 2

//Accepts only public network
DEFINE HOUSIN.BIN1
  AS HOUSIN.WATERO = 1
  TYPE BOOL
  DEFAULT 0

//Accepts gas or electricity
DEFINE HOUSIN.BIN2
  AS HOUSIN.FUEL = 1 OR HOUSIN.FUEL = 4
  TYPE BOOL
  DEFAULT 0

//Accepts only sewage system
DEFINE HOUSIN.BIN3
  AS HOUSIN.TOILCO = 1
  TYPE BOOL
  DEFAULT 0

//Accepts concrete or clay brick
DEFINE HOUSIN.BIN4
  AS HOUSIN.WALLS = 1 OR HOUSIN.WALLS = 3
  TYPE BOOL
  DEFAULT 0

//Accepts slate or tiles
DEFINE HOUSIN.BIN5
  AS HOUSIN.ROOF = 2 OR HOUSIN.ROOF = 3
  TYPE BOOL
  DEFAULT 0

//Accepts parquet or carpeting
DEFINE HOUSIN.BIN6
  AS HOUSIN.FLOOR = 1 OR HOUSIN.FLOOR = 3
  TYPE BOOL
  DEFAULT 0
```

```
DEFINE HOUSIN.CONDITION
  AS HOUSIN.BIN1 + HOUSIN.BIN2 + HOUSIN.BIN3 + HOUSIN.BIN4 + HOUSIN.BIN5 + HOUSIN.BIN6
  RANGE 0-6
  TYPE INTEGER
  VARLABEL "Dwelling Conditions"
  VALUELABELS 0 "Very bad" 1 "Bad" 2 "Medium bad"
              3 "Medium good" 4 "Good" 5 "Very Good"
              6 "Excellent"

TABLE VERIF AS FREQUENCY OF HOUSIN.CONDITION
```

Programming notes:

1. As this type of indicator applies only to private and occupied households, we decided to filter the cases using the SEXHEAD (sex of the household). If we look at the frequency of this variable and the frequency of the ones being used for the indicators, we will see that all of them has the "magic" 2544 number for the non response, so, it is fairly safe to use the SEXHEAD as our filter in the RUNDEF command.
2. The specific filters for each binary indicator is documented in the program but commented out using //.
3. The range of the CONDITION variable varies from 0-6. The quality index thus has 7 values.
4. All DEFINEs that create the binary indicators include the DEFAULT 0 clause, so that any non-applicable or missing value does not invalidate the variable and thus the final sum. Any missing or non-applicable cases are evaluated as 0.

Results:

| Categories | Counts | % |
|---|---|---|
| Very bad | 704 | 6.19 |
| Bad | 1833 | 16.11 |
| Medium bad | 3069 | 26.98 |
| Medium good | 2782 | 24.46 |
| Good | 2182 | 19.18 |
| Very Good | 752 | 6.61 |
| Excellent | 53 | 0.47 |
| Total | 11375 | 100.00 |

Comments:

1. The majority of dwellings are considered in medium bad or medium good conditions, according to the specific choice of options.
2. This score may be saved permanently in the base in order to produce a list by area.

## Section 21: CALCULATING THE DEPENDENCY RATIO BASED ON AGE GROUPS

Program description:

This program calculates the dependency ratio indicator by area. The dependency ratio of a given area is defined as the ratio of dependants (children and elders) by the independents (adults). The variable PERSON.GRP3 (programmed earlier in Section07) is used for establishing the respective counts.

The program illustrates the production of a list at COUNTY level.

```
RUNDEF Section21
DEFINE PERSON.GRP3 AS RECODE PERSON.AGE
   (0 - 14=1)  (15 - 64=2)  (65 - HIGHEST =3)
   RANGE 1 - 3
   VARLABEL "Age groups 3"
   VALUELABELS 1 "Children" 2 "Adults" 3 "Elders"

DEFINE COUNTY.DEPEND AS COUNT PERSON
  FOR PERSON.GRP3 <> 2
  VARLABEL "# of Dependents"

DEFINE COUNTY.INDEP AS COUNT PERSON
  FOR PERSON.GRP3 = 2
  VARLABEL "# of Actives"

DEFINE COUNTY.DEPRATIO AS 100 * (COUNTY.DEPEND / COUNTY.INDEP)
  TYPE REAL
  FOR COUNTY.INDEP > 0
  VARLABEL "Dependency Ratio"
  DECIMALS 2

TABLE VERIF AS AREALIST OF COUNTY, COUNTY.NCOUNTY,
  COUNTY.DEPEND, COUNTY.INDEP, COUNTY.DEPRATIO
  TITLE "Dependency Ratio in New Miranda"
  DECIMALS 2
```

Programming notes:

1. Every time we use a division operation we need to:
   - 1.1 define the variable as type real
   - 1.2 check to prevent a division by 0.

Results:

| Cod | County Name | # of Dependents | # of Actives | Dependency Ratio |
|---|---|---|---|---|
| 5 | Santa Maria | 8293 | 13435 | 61.73 |
| 6 | Santiago | 4191 | 4778 | 87.71 |
| 7 | Bolivar | 6053 | 8228 | 73.57 |
| 8 | Marbella | 1455 | 2363 | 61.57 |
| 9 | Puerto Nuevo | 1311 | 2082 | 62.97 |

Comments:

1. The county with the highest dependency ratio is Santiago.

## Section 22: NOT APPLICABLE VALUES

Program description:
This section illustrates the concept of Not Applicable value. A non-applicable value reflects the non-existence of the definition of a concept (variable) for an entity. This program runs different calculations of the same variable in order to demonstrate how to control the non-applicable value.

```
RUNDEF Section22

//Case 1
DEFINE PERSON.WFERT1 AS PERSON.SEX = 2 AND PERSON.AGE > 14
  TYPE BOOL
  VARLABEL "Females over fertile age 1"

TABLE VERIF1 AS FREQUENCY OF PERSON.WFERT1

/*
Results:
Categories  Counts
 False       35345
 True        16844
 Total       52189

Comments:
1. The calculation of PERSON.WFERT1 considers the entire population.
2. The category “False” includes males as well as females under 15.
*/

//Case 2
DEFINE PERSON.WFERT2 AS 1
  FOR PERSON.SEX = 2 AND PERSON.AGE > 14
  TYPE BOOL
  VARLABEL "Females over fertile age 2"

TABLE VERIF2 AS FREQUENCY OF PERSON.WFERT2

/*
Results:
Categories  Counts
 True        16844
 Total       16844

NotApp :     35345

Comments:
1. The calculation of PERSON.WFERT2 considers only female over fertile age.
2. All other individuals are not considered in the calculation (clause FOR) and are set to non-applicable. Only one category (“True”) is tabulated. The other category is at the bottom, in the not applicable line.
*/
```

```
//Case 3
DEFINE PERSON.WFERT3 AS PERSON.AGE > 14
  FOR PERSON.SEX = 2
  TYPE BOOL
  VARLABEL "Females over fertile age 3"

TABLE VERIF3 AS FREQUENCY OF PERSON.WFERT3

/*
Results:
Categories  Counts
 False       8511
 True       16844
 Total      25355

NotApp :     26834

Comments:
1. The calculation of PERSON.WFERT3 considers only female individuals.
2. All males have a non-applicable value. Category “False” includes females under 15.
*/

//Case 4
DEFINE PERSON.WFERT4 AS PERSON.AGE > 14
  FOR PERSON.SEX = 2
  VARLABEL "Females over fertile age 4"
  DEFAULT 2
  VALUELABELS 1 "True" 0 "False" 2 "Males"
  RANGE 0-2

TABLE VERIF4 AS FREQUENCY OF PERSON.WFERT4

/*
Results:
Categories   Counts
 False        8511
 True        16844
 Males       26834
 Total       52189

Comments:
1. The calculation of PERSON.WFERT4 considers only females, however forces a category 2 for
unconsidered cases (males).Thus, the males are included in the tabulation (note the DEFAULT 2 as well as
the RANGE 0-2).
2. In this case we cannot use a BOOL variable.
3. This is better than PERSON.WFERT1 since it has all possible combinations, female under 15, female
over 14, and males.
*/
```