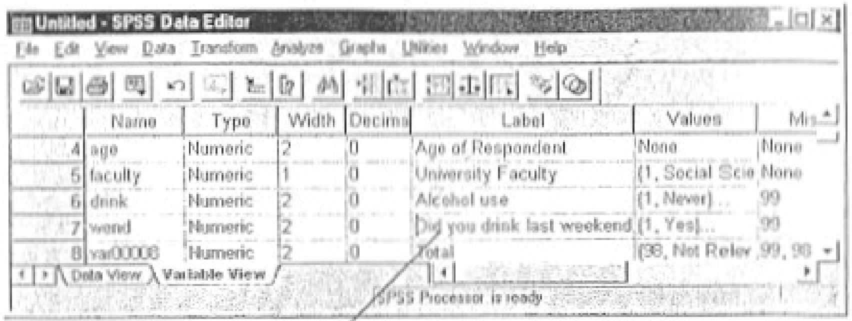
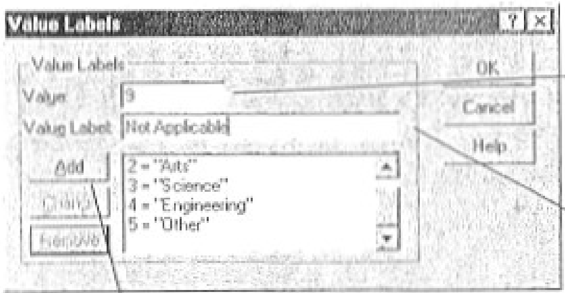
Figure 1.4e Labelling a variable


Untitled - SPSS Data Editor
File Edit View Data Transform Analyze Graphs Utilities Window Help
Data View
Variable View
SPSS Processor is ready

	Name	Type	Width	Decima	Label	Values	Mis
4	age	Numeric	2	0	Age of Respondent	None	None
5	faculty	Numeric	1	0	University Faculty	{1, Social Scie	None
6	drink	Numeric	2	0	Alcohol use	{1, Never}...	99
7	wend	Numeric	2	0	Did you drink last weekend	{1, Yes}...	99
8	var00008	Numeric	2	0	Total	{98, Not Relev	99, 98

To insert additional information on the variable, click on the **Label** cell, type in the details, and press **the Enter** or **Tab key**.

Figure 1.4f Defining **Values**



Enter the value for each *Faculty*:
1=Social science,
2=Arts,
3=Science.

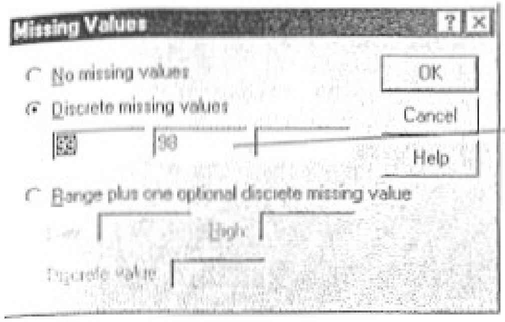
Enter the **Value Label** e.g.
9 = Not Applicable

Click on **Add** to save the labels.
Click on **OK** when complete.

It is also important to identify the *missing value* codes for SPSS (see the section below on missing values). Click on the **Missing Values** cell of the row of the required variable, move the cursor to the three dots in the shaded corner of the cell, and click once. A window like Figure 1.4g will come up. Enter the values which are consided missing, click on the **OK** button to go back to the Variable View window.

It is possible to change any of the characteristics (name, width, type, label, columns, measure (scale, ordinal or nominal)) in the Variable View window by clicking on the appropriate cell. It is also possible to copy a format from one variable to another using **Copy** and **Paste**. Highlight the cell which has been formatted, copy the format using **Edit Copy**. Move to the cell of the

Figure 1.4g Defining **Missing Values**

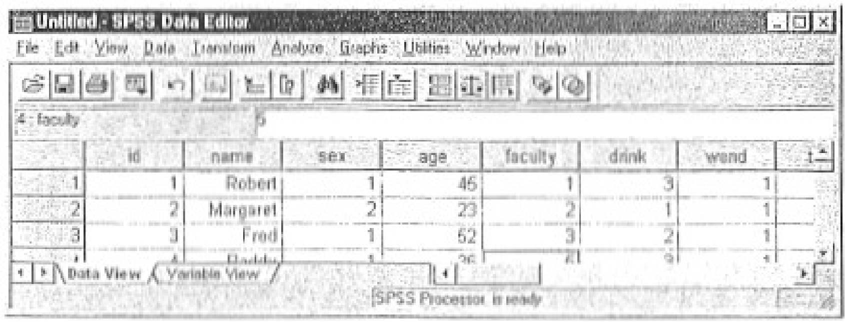


For this variable – Total number of drinks last weekend – the value **99** is allocated for **missing** and **98** is allocated for **not relevant** (not a drinker).

variable you want to copy the format to, and paste the format using **Edit Paste**. This is particularly useful for entering labels from a questionnaire with the same values and labels such as Likert scales.

Once you have entered the data and defined the variables it is possible to view the data with the new labels instead of the numeric values. To view the value labels of the dataset, go to the **Data View** format in the Data Editor, open the pull-down menu **View**, and click on **Value Labels**. The appearance of the grid will change from that showing the numeric codes (as in Figure 1.5a) to a grid showing the alphanumeric labels (as in Figure 1.5b).

Figure 1.5a Drink survey data in **Value** format

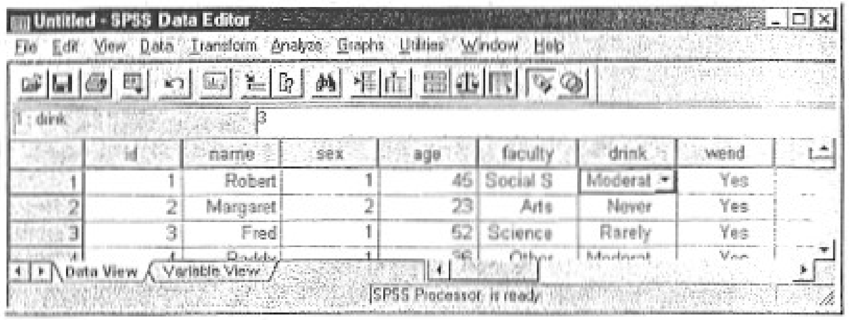

Untitled - SPSS Data Editor
File Edit View Data Transform Analyze Graphs Utilities Window Help
4 : faculty
Data View
Variable View
SPSS Processor is ready

	id	name	sex	age	faculty	drink	wend
1	1	Robert	1	45	1	3	1
2	2	Margaret	2	23	2	1	1
3	3	Fred	1	52	3	2	1

Validating

Validation of a dataset is conducted using *consistency checks* to remove or control errors and missing information. It is easy for errors in codings to creep into a dataset from a variety of sources. Before commencing the first real analyses, it is good practice to try to remove as many errors as practicable. One efficient means of removing errors is to use the *computer package itself* to check for errors. Errors can be identified by checking for *invalid* or *inconsistent* codes.

Figure 1.5b Drink survey data in Label format


Untitled - SPSS Data Editor
File Edit View Data Transform Analyze Graphs Utilities Window Help
1 : drink
3
Data View
Variable View

	id	name	sex	age	faculty	drink	wend
1	1	Robert	1	45	Social S	Moderat	Yes
2	2	Margaret	2	23	Arts	Never	Yes
3	3	Fred	1	52	Science	Rarely	Yes
4	4	Paddy	1	36	Other	Moderat	Yes

SPSS Processor is ready

Checking for invalid codes

The most obvious validation is to check for incorrect codes by seeking out impossible or invalid values. All possible *correct* code values for a variable are specified. Then, if any codes exist other than the possible values, they must be errors. For instance, in our **fac** example from the 'Drinking questionnaire', the only possible values are within the range 1 to 5 and 9. So, if you found a zero or a 6, 7 or 8 (or any other invalid value), it must be a coding error. These coding errors should be corrected before any analysis:

Variable: *fac, Faculty*

0	*3*	*X*
1 *Social sciences*	55	
2 *Arts*	186	
3 *Science*	97	
4 *Engineering*	131	
5 *Other*	4	
6	*1*	*X*
9 *Not applicable*	23	
Total	500	

Checking for inconsistent codes

Sometimes, a particular code or codes for one variable will mean that the codes for another variable must, or must not, fall within a certain range of values – that is, the codings of the two variables, while both are within the range of possible codes, may be *inconsistent* with each other.

For instance, you might do a consistency check with the 'Drinking questionnaire' data to see whether everyone coded as a 1 on the variable **drink** (that is, everyone who said they were a non-drinker) *also* has codings of zero for the variables **beer** (units of beer/cider drank over the weekend), **wine** (units of wine) and **spirit** (units of spirit). Non-drinkers should not have drunk any units. If it appears that a claimed non-drinker does have some units of alcohol consumed, an inconsistency exists which should be resolved. (This might require going back to the original source of the information in order to trace the reason for the apparent error.) If the

correct code can be established, the data grid can be edited to replace the error. If a correct code cannot be established, it may be necessary to declare the inconsistent or incorrect code as a *missing value*.

Software checks on invalid codes

You will note that tracking down the original sources of invalid codings or inconsistencies and then correcting them within the dataset can be a tedious and time-consuming operation, perhaps requiring going all the way back to the original source of the data. It is possible to get around this by using a specialist data entry package which is capable of performing checks for valid values and for inconsistencies between variables *at the point of coding the data*. The data is coded directly in and the package is continually checking for out-of-range and/or inconsistent values as they are keyed in. If an invalid or inconsistent value is entered, the error is highlighted on the spot. This allows the coder to check the error against the original source while that original source is in front of them and to repair the damage immediately.

Dealing with missing values

The expression 'missing values' may sound a bit odd to you, but there are many instances where, for only some variables for only some cases, data may be missing. For instance, our 'Drinking questionnaire' might have a second page where people are asked to tell us about the ill effects they suffer from drinking (hangovers). People who do not drink would not need to answer that section. That is, often a variable will not be coded for some cases because the variable *Does not apply* to those cases. Another example could be where only people who live in rented accommodation would answer the question about *'How much rent do you pay?'* or where questions about a person's children would not apply to people who have no children.

Information can of course be missing for other reasons. The information may not be available simply because it is *not known*. A person may *refuse* to answer some questions on a questionnaire or in an interview. Respondents sometimes write in illegible answers on a questionnaire or make simple mistakes like inadvertently skipping over some questions or even circling *two* answers when only one is required. *Errors* may have occurred in the coding or transcription of data so that you know the code is incorrect but you cannot find out the correct code.

Regardless of cause, all types of missing data are normally dealt with in the same manner; a special code or codes is applied to each instance of missing information. For example, with **fac** on the 'Drinking questionnaire', the code 9 was used to indicate those cases where the information on *'Faculty or division'* was not known or did not apply for some reason.

While any codes can be used to signify missing values, there are some rules of good practice with missing values which are advisable.

Strategies for coding missing values

It is tempting to avoid the problem of missing values by leaving blank the space where the code should go. After all, one does not know the code, so why not just put nothing? This, however, is not advisable. A blank space can easily end up being converted into a zero by accident ('_' → 0). If zero can be a genuine code for the variable, the result is an error in the dataset.

Secondly, blanks can occur for other reasons, such as someone accidentally skipping a space when typing in the data. Therefore, one can not be sure that the blank is a genuine missing value, or just a mistake.

Similarly, people often choose to use zeros as a *missing value code*. While better than blanks, this is still not advisable. As before, an accidentally entered blank space can end up being read as zero; introducing an error. Also, the value zero can often be a genuine code. For example, if people are asked to state how many children they have; many will legitimately say '0'.

The best practice is to use a value that is completely *out of the range of real possible values* as a missing value code. For instance, with fac above, the genuine code values were 1 to 5; 9 is used as a missing value code. It is completely out of the range of possible values and, being an actual number, has to be entered in as a deliberate code (so it cannot appear by accident as a zero or blank could).

Sometimes people use more than one missing value code so that they can keep track of *why* the value is missing. For instance, one might use different missing value codes to distinguish between: *'refused to answer'*; *'does not apply'*; *'answer unknown'*, etc. Whether one chooses to use more than one missing value code depends upon whether knowing the reason for data being missing is of importance. (For instance, it may be important to know if people refuse to answer a certain question.)

The convention many people follow with missing values is to use the highest possible values available.

For instance, with our example of fac where 1 to 5 are legitimate codes, one might use: 9 to mean *'Refused'*; 8 to mean *'Does not apply'*; and 7 to mean *'Missing for some other reason'*:

Variable: fac, *Faculty*

1 *Social sciences*	55
2 Arts	186
3 Science	97
4 Engineering	131
5 Other	4
7 *Missing, other reason*	*3*
8 *Does not apply*	*1*
9 *Refused*	23
Total	500

If the variable takes up two columns, say, 'Number of children', you could maintain this practice of using the highest values to signify missing values; for example, 97, 98 and 99 or 77, 88 and 99. People might have 0, 1, 2, 3, 4 ... maybe even some with 10, 11, 12 children; but no one would have 77, 88, 97, 98 or 99 children!

For three-digit variables, one could use 777, 888, 997, 998 or 999; 7777, 8888, 9997, 9998, 9999 for four-digit variables; and so on for five digit variables, etc.

The importance of missing values

The importance of specifying missing value codes goes beyond just the cosmetic reasons of having them clearly displayed as missing values in tabulations. When certain codes for a variable are identified for SPSS as 'missing values', these codes will be excluded automatically from any mathematical calculations that are carried out on the variables. If this was not the case, completely misleading results would occur. For instance, taking the example of a code for 'Number of children' where the missing values codes of 97, 98 and 99 have been used. If the average number of children was calculated for the dataset with even a *small* proportion of missing value codes 97, 98 and 99 being averaged in with the true codes of 0, 1, 2, 3, etc. the overall average would be grossly inflated. Once these missing values have been specified to SPSS, however, they will play no part in the calculation and a misleading result will be avoided. A similar example of how SPSS would exclude missing values can be seen if we look at the tabulation of our variable fac with the numbers in each legitimate category being given as a percentage of the whole:

Variable: fac, *Faculty*

	N	%
1 Social sciences	55	11.6
2 Arts	186	39.3
3 Science	97	20.5
4 Engineering	131	27.7
5 Other	4	0.1
7 Missing, other reason	3	–
8 Does not apply	1	–
9 Refused	23	–
Total	500	100.0

27 cases are missing values

Conclusion

In the end, once:

- the data have been entered
- variables and their values have been 'labelled' and defined fully
- missing values are specified fully
- and the data have been 'cleaned' by checking for valid ranges of values and internal consistency

you have a *fully operational and self-supporting dataset*, ready for analysis.

Once the SPSS dataset has been created and data has been put into it, the resulting grid will look like Figure 1.5a, which shows the cases from our 'Drinking questionnaire'.

Some tips

- *Save your file frequently* as you enter the data. Do not wait until you have entered all of the data – save your file every 15 minutes. That way, you will have a permanent copy of most of your data even if something goes wrong while you're working.
- Make a *backup copy* (an extra copy) of the data to use in case your original data file is somehow lost or destroyed. Since disks can be damaged, put the backup copy on a *different physical disk*. If the data are important, be sure to label the backup floppy diskettes clearly and put them in a safe place
- Use the pull-down **Help** menu on the toolbar and the tutorial for *further advice and help*.