**ASSESSMENT 2**

INITIAL INSTRUCTIONS

**Import raw data into a SAS data set**

The first stage in analytics programming is to analyse the problem and to create a list of requirements. The 'problem' for this practical session assignment is to import raw data into a SAS data set. The steps are:

1. look at the data set description
2. create a minimal program
3. add test data lines
4. input numeric and character variables
5. give each variable a suitable label, input format and output format
6. fix any errors, warnings or notes that indicate a problem

**A minimal SAS program for importing raw data should have the following features:**

1. a data step naming the SAS data set
2. an infile statement to read the data
3. an input statement that inputs the first variable of the data set
4. a test data line for which the inputs are known

**QUESTION 1**

Using the description of the credit raw data in the folder for week 2, construct a minimal program. The template for the data step is:

data **DATASET**;  
 infile  
 **source** ;  
 input  
 **variables** ;  
 datalines;  
 **data** */\* notice the alignment to the left margin \*/* ;  
 run;

The important elements of the code are:

* *data****DATASET***: introduces the data step and is partnered by a '*run*' statement.
* *infile****source***: states the source of the data.
* *input****variables***: introduces the list of variables to be imported.
* *datalines*: introduces the raw data to be read.

You should replace the items in bold with suitable code. It is a good idea to use capitals for data-set names in order to make them stand out.

Recall that the source of data - the *datalines* - will be replaced by a file later in the development of the code. At this stage, you might like to download the '*credit.dat*' data file and to put it in a suitable such as an 'IMAT5168' folder that you create for the work of this module.

You can either construct a data line using the specification or copy a line from the downloaded data file, using Notepad++. (If you use Linux or Apple's iOS, you will need to find an alternative.

Run and test your code and copy it into the box below, when you judge that the program is running correctly.

**ANSWER 1**

**data** CREDIT;

infile

datalines

;

input

ID\_Number /\* ID Number\*/

;

datalines;

1 1 42 2 2 7882 1 4 2 3 3 4 2 45 3 3 1 3 2 1

;

**run**;

**QUESTION 2**

##### Data set analysis

The data set description is made up of 3 columns, the variable, the column location of the variable and a description of how its values are encoded.

This means that each variable for a data subject starts and ends in a particular location in the line of text. This format is known as 'column' format.

Typically, if a column spans more than one character the locations are given as a range such as: '*5-7*'. If the column is only one character wide, it may be specified using just the starting location: '*8*'.

###### **Self-assessment question**

Download the credit data set from the folder for week 2 and open it using Notepad++. In Notepad++, switch on 'Show all characters' in the 'View|Show Symbol' pop-up menu. (If you are using Linux, you will need to find an alternative to Notepad++ with similar capabilities.) You should now be able to see each character in the data set.

An orange '·' means: 'space'. The white on black 'CR' and 'LF' mean characters which signal a carriage return and a line-feed. 'CR' and 'LF' are the Microsoft Windows line termination characters. (Linux and Apple iOS systems use different combinations of characters.)

Notepad++ places the vertical bar of the cursor to the left of the character and begins in column 1. One right-arrow press moves to column 2 and a further press to column 3.

Using Notepad++'s view of the data and the information in the data set description, add the location of the first variable to your minimal program. Run the program and paste the *input statement* in the box below, when you judge that the program is running correctly.

*Note*: being able to view all the characters in raw data and to see where columns start and end is a valuable aid to fixing problems with reading data arranged in columns.

**ANSWER 2**

**data** CREDIT;

infile

datalines

;

input

customer **1**-**3** /\* ID Number\*/

;

datalines;

1 1 42 2 2 7882 1 4 2 3 3 4 2 45 3 3 1 3 2 1

;

**run**;

**proc** **print** data=credit;

**run**;

**QUESTION 3**

##### Obtain rapid feedback

Recall that before running the program, you should:

* clear the log
* delete all results
* close all open data sets

and that after running the program, you should:

* check the log
* review the results
* review the dataset

The latter process takes a number of steps with the SAS program's user interface and so a quicker solution may be to have the relevant data printed in the results window.

###### **Self-assessment question**

Using SAS help, write a 'procedure' (proc) step to print the *contents* of the data set. Apart from summarizing the size of the SAS data set, this should list each variable, read from the raw data. Also, write a second procedure to print the value of each variable in the data set.

The template for **proc** steps is:

proc **name** data=**source**;  
 */\* options are specific to the named procedure \*/*  
 run;  
   
Run the program and paste your procedure steps into the box below, when you judge that the program is running correctly

**ANSWER 3**

**data** CREDIT;

infile

datalines

;

input

ID\_Number **1**-**3** /\* ID Number\*/

;

datalines;

1 1 42 2 2 7882 1 4 2 3 3 4 2 45 3 3 1 3 2 1

;

**run**;

**proc** **contents** data =CREDIT;

**run**;

**proc** **print** data =CREDIT;

**run**;

**QUESTION 4**

##### Give each variable a suitable label.

A programmer can use any names that are meaningful within the context of the program. This allows the use of short names that describe the data to the programmer. By default, SAS uses these names as the label for variables when printing results. Since short names are not always meaningful to the client, we can add labels to each variable to print more useful text.

###### **Self-assessment question**

Use the *label* statement within the data step to label the variable you read in your minimal program by means of the following steps:

1. Using appropriate sources of help, write a template for the label statement that uses a similar format to earlier questions in this assignment.
2. Add a label statement to your code that labels the first variable.

Run the program, and paste the label statement into the box below, when you judge that the program is running correctly.

|  |
| --- |
| For the toolbar, press ALT+F10 (PC) or ALT+FN+F10 (Mac). |

**ANSWER 4**

**data** CREDIT;

infile

datalines;

/\* creating one variable \*/

input

ID\_Number **1**-**3** /\* 1-3 is the column number represented in notepad++ \*/

;

Label

ID\_Number = ‘Customer ID’;

datalines;

1 1 42 2 2 7882 1 4 2 3 3 4 2 45 3 3 1 3 2 1

;

**run**;

**proc** **contents** data =CREDIT;

**run**;

**proc** **print** data =CREDIT;

**run**;

**QUESTION 5**

##### Give each variable suitable format.

SAS views raw data as a sequence of characters ending in one or more-line termination characters.

When the raw characters are read from a file they can be translated into other characters or numbers using an 'input format'. (SAS often abbreviate this term to 'informat'.)

As with labels, the numbers or characters stored has the value of a data subject's variable can be translated before printing within a set of results. This is done using an 'output format' or 'outformat'.

Formats are first defined in a *format* procedure statement and then applied to a variable in the data step.

###### **Self-assessment question**

The template for a format procedure statement is:

proc format;  
 **type** **name**  
 **mappings**  
 ;  
 */\* repeat as necessary \*/*  
 run;

label

**1**='...<£0'

**2**= '£0 <=...<200'

**3**='...>=200'

**4**= 'no cheque amount'

The **type** may be '*value*' for output formats and '*invalue*' for input formats. **name** is the name of the format. **mappings** are the list of mappings: for '*invalues*' raw data is mapped to what should be stored in the variable; for *'values'* values stored in the variable are mapped onto the text to be printed in results. You may add as many formats as you need to the *proc format* statement.

Your task is to:

1. use a proc step to define the output format for the cheque account variable (a suggested name is: '*cr\_account*')
2. define a template for the list of mappings to be used in an output format
3. add the list of mappings for the cheque account variable to the *proc format* statement
4. add the account variable to the list of variables in the input statement
5. add a suitable label for the variable
6. add an output format to the variable in the data step

When your code is complete, and the log shows no errors, paste your data statement into the box below.

*Hints:* the data-step format statement has the following template:

format  
 **variable** **format**.  
 */\* repeat as necessary \*/*  
 ;

**variable** is the variable to which the format applies. the **format** is the name of the format**.** the full stop terminates the format.

You will not need to add any in formats for this assignment. They are added to the input statement and an illustration will follow in a future assignment.

**ANSWER 5**

data CREDIT;

infile

datalines

;

input

Cheque\_Account 5 /\* numeric variable\*/

;

label

Cheque\_Account = *'Cheque Account'* /\* data on VIEWTABLE: Work.Credit \*/

;

datalines;

5 2 36 2 1 6948 1 3 2 3 1 2 3 35 3 1 1 4 1 2

;

proc print data = CREDIT;

run;

proc format;

value cr\_account 1 = *'<£0'*

2 = *'£0<=...<£200'*

3 = *'>=£200'*

4 = *'no cheque ammount'*

;

run;

proc print data = CREDIT;

format Cheque\_Account cr\_account.

;

run;

**QUESTION 6**

##### Systematically build up the data set

Until now, the questions demonstrate how a data set can be imported step-by-step. Provided that each new variable is added only after all errors relating to previous ones have been fixed, the number of lines to review for mistakes is small.

If raw data consists of millions of lines, it is usually possible to use SAS to select a sample and to put them into a data file. This can later be copied into a *datalines* statement or used directly so simplifying development and avoiding having to wait a long time for millions of records to be processed.

###### Self-assessment question

Using the following steps:

1. Add the next variable with its location information
2. Add the label
3. Create the output format
4. Add the output format

complete reading all the raw data from the *datalines*. When your program works correctly, copy the code into the box below.

**ANSWER 6**

**data** CREDIT;

infile

datalines

;

input

ID\_Number /\*1-3 \*/

Cheque\_Account /\*5 numeric variable\*/

Duration\_in\_months /\*7-8 non standard data \*/

Credit\_History /\*10 numeric variable \*/

Purpose $ /\* 12 numeric variable \*/

Credit\_Amount /\* 14-18 numeric variable \*/

Savings\_accounts\_bonds /\* numeric variable \*/

Present\_empolyment\_since /\*22 numeric variable \*/

Instalment\_rate\_income /\*24 \*/

Personal\_status\_and\_sex /\* 26 numeric variable\*/

Other\_debtors\_guarantors /\* 28 numeric variable\*/

Present\_residence\_since /\* 30 numeric and character variable \*/

Property /\* 32 numeric variable \*/

Age\_in\_years /\*34-36 numeric data \*/

Other\_instalment\_plans /\* numeric data \*/

Housing /\* numeric data \*/

Number\_of\_existing\_credits /\* 42 \*/

Job /\* 44 numeric data \*/

Number\_of\_dependents /\*46\*/

Telephone /\*48 numeric data \*/

Foreign\_worker $ /\*50 numeric\*/

;

label

ID\_Number = 'ID Number'

Cheque\_Account = 'Cheque Account'

Duration\_in\_months = ' Duration in months'

Credit\_History = ' Credit History'

Purpose $ = 'Purpose'

Credit\_Amount = ' Credit Amount'

Savings\_accounts\_bonds = 'Saving accounts/bonds'

Present\_empolyment\_since = 'Present empolyment since'

Instalment\_rate\_income = 'Instalment rate % income'

Personal\_status\_and\_sex = 'Personal stauts and sex'

Other\_debtors\_guarantors = 'Other debtors/guarantors'

Present\_residence\_since = 'Present residence since'

Property 'Property'

Age\_in\_years 'Age in years'

Other\_instalment\_plans = 'Other instalment plans'

Housing = 'Housing'

Number\_of\_existing\_credits = ' Number of existing credits'

Job = 'Job'

Number\_of\_dependents = 'Number of dependents'

Telephone = 'Telephone'

Foreign\_worker $ = 'Foreign worker'

;

datalines;

15 1 10 4 0 2241 1 2 1 3 1 3 1 48 3 1 2 2 2 1 1

;

**proc** **print** data = CREDIT;

**run**;

**proc** **format** library = sasuser

;

value cr\_account **1** = '<£0'

**2** = '£0<=<£200'

**3** = '>=£200'

**4** = 'no cheque amount'

;

value cr\_history **0** ='no credits taken/all credits paid back duly'

**1** ='all credits at this bank paid back duly'

**2**='existing credits paid back duly till now'

**3**='delay in paying off in the past'

**4**='critical account/other credits existing (not at this bank)'

**9**='missing'

;

value $pr **0**='car (new)'

**1**='car (used)'

**2**='furniture/equipment'

**3**='radio/television'

**4**='domestic appliances'

**5**='repairs'

**6**='education'

**7**='vacation'

**8**='retraining'

**9**='business'

X='others'

;

value sab **1**= '< £100'

**2**= '£100 <= < £500'

**3**='£500 <= < £1000'

**4**='>= £1000'

**5**='unknown/no savings account'

;

value pes **1**='unemployed'

**2**='< 1 year'

**3**='1 <= < 4 years'

**4**='4 <= < 7 years'

**5**='>= 7 years'

;

value pse **1**='male :divorced/separated'

**2**='female:divorced/separated/married'

**3**='male :single'

**4**='male :married/widowed'

**5**='female:single'

**9**='missing'

;

value odg **1**='none'

**2**='co-applicant'

**3**='guarantor'

**9**='missing'

;

value prs

**4**='4 years or more'

**9**='missing'

;

value prp **1**='real estate'

**2**='if not 1: building society savings agreement/life insurance'

**3**='if not 1/2: car or other'

**4**='unknown /no property'

;

value aiy **999**=' missing'

;

value oip **1**='bank'

**2**='stores'

**3**='none'

**9**='missing'

;

value hsg **1**='rent'

**2**='own'

**3**='for free'

**9**='missing'

;

value job **1**='unemployed'

**2**='unskilled'

**3**='skilled employee/official'

**4**='management/self-employed/highly qualified employee/officer'

**9**='missing'

;

value tel **1**='yes, registered under the customers name'

**2**='no'

**9**='missing'

;

value $fw **1**='yes'

other = ' '

;

**run**;

**proc** **print** data = CREDIT;

format

Cheque\_Account cr\_account. Credit\_History cr\_history. Purpose $pr. Savings\_accounts\_bonds sab. Present\_empolyment\_since pes.

Personal\_status\_and\_sex pse. Other\_debtors\_guarantors odg. Present\_residence\_since prs. Property prp. Age\_in\_years aiy.

Other\_instalment\_plans oip. Housing hsg. Job job. Telephone tel. Foreign\_worker $fw.

;

**run**;

**QUESTION 7**

##### Read the data file

The next step is to see if the code reads the credit raw data file.

###### Self-assessment question

Carry out the following tasks:

1. Add the following statement to your program file:

filename pwd '/folders/myfolders/imat5168/';

and alter the path statement to the folder where the 'credit.dat' raw data file is located.

1. Change the *infile* statement so that '*datalines*' is replaced by:

pwd(credit.dat)

1. Without making further changes, run the program.   
   (... Although you may find it useful to add:

(obs=10)

to the *print* procedure after *data=CUSTOMER* to limit the number of lines printed.)

Record in the box below your observations of what happened. Has the data been read correctly? If there are problems, suggest and justify a possible solution.

**ANSWER 7**

/\* pwd = Present Working Directory\*/

/\* The location for the data file \*/

**data** CREDIT;

infile

'D:\Sas\credit.dat'

;

input

ID\_Number /\*1-3 \*/

Cheque\_Account /\*5 numeric variable\*/

Duration\_in\_months /\*7-8 non standard data \*/

Credit\_History /\*10 numeric variable \*/

Purpose $ /\* 12 numeric variable \*/

Credit\_Amount /\* 14-18 numeric variable \*/

Savings\_accounts\_bonds /\* numeric variable \*/

Present\_empolyment\_since /\*22 numeric variable \*/

Instalment\_rate\_income /\*24 \*/

Personal\_status\_and\_sex /\* 26 numeric variable\*/

Other\_debtors\_guarantors /\* 28 numeric variable\*/

Present\_residence\_since /\* 30 numeric and character variable \*/

Property /\* 32 numeric variable \*/

Age\_in\_years /\*34-36 numeric data \*/

Other\_instalment\_plans /\* numeric data \*/

Housing /\* numeric data \*/

Number\_of\_existing\_credits /\* 42 \*/

Job /\* 44 numeric data \*/

Number\_of\_dependents /\*46\*/

Telephone /\*48 numeric data \*/

Foreign\_worker $ /\*50 numeric\*/

;

label

ID\_Number = 'ID Number'

Cheque\_Account = 'Cheque Account'

Duration\_in\_months = ' Duration in months'

Credit\_History = ' Credit History'

Purpose = 'Purpose'

Credit\_Amount = ' Credit Amount'

Savings\_accounts\_bonds = 'Saving accounts/bonds'

Present\_empolyment\_since = 'Present empolyment since'

Instalment\_rate\_income = 'Instalment rate % income'

Personal\_status\_and\_sex = 'Personal stauts and sex'

Other\_debtors\_guarantors = 'Other debtors/guarantors'

Present\_residence\_since = 'Present residence since'

Property 'Property'

Age\_in\_years 'Age in years'

Other\_instalment\_plans = 'Other instalment plans'

Housing = 'Housing'

Number\_of\_existing\_credits = ' Number of existing credits'

Job = 'Job'

Number\_of\_dependents = 'Number of dependents'

Telephone = 'Telephone'

Foreign\_worker = 'Foreign worker'

;

**proc** **print** data = CREDIT (obs=**10**);

**run**;

**proc** **format** library = sasuser

;

value cr\_account **1** = '<£0'

**2** = '£0<=<£200'

**3** = '>=£200'

**4** = 'no cheque amount'

;

value cr\_history **0** ='no credits taken/all credits paid back duly'

**1** ='all credits at this bank paid back duly'

**2**='existing credits paid back duly till now'

**3**='delay in paying off in the past'

**4**='critical account/other credits existing (not at this bank)'

**9**='missing'

;

value $pr **0**='car (new)'

**1**='car (used)'

**2**='furniture/equipment'

**3**='radio/television'

**4**='domestic appliances'

**5**='repairs'

**6**='education'

**7**='vacation'

**8**='retraining'

**9**='business'

X='others'

;

value sab **1**= '< £100'

**2**= '£100 <= < £500'

**3**='£500 <= < £1000'

**4**='>= £1000'

**5**='unknown/no savings account'

;

value pes **1**='unemployed'

**2**='< 1 year'

**3**='1 <= < 4 years'

**4**='4 <= < 7 years'

**5**='>= 7 years'

;

value pse **1**='male :divorced/separated'

**2**='female:divorced/separated/married'

**3**='male :single'

**4**='male :married/widowed'

**5**='female:single'

**9**='missing'

;

value odg **1**='none'

**2**='co-applicant'

**3**='guarantor'

**9**='missing'

;

value prs

**4**='4 years or more'

**9**='missing'

;

value prp **1**='real estate'

**2**='if not 1: building society savings agreement/life insurance'

**3**='if not 1/2: car or other'

**4**='unknown /no property'

;

value aiy **999**=' missing'

;

value oip **1**='bank'

**2**='stores'

**3**='none'

**9**='missing'

;

value hsg **1**='rent'

**2**='own'

**3**='for free'

**9**='missing'

;

value job **1**='unemployed'

**2**='unskilled'

**3**='skilled employee/official'

**4**='management/self-employed/highly qualified employee/officer'

**9**='missing'

;

value tel **1**='yes, registered under the customers name'

**2**='no'

**9**='missing'

;

value $fw **1**='yes'

other = ' '

;

**run**;

**proc** **print** data = CREDIT;

format

Cheque\_Account cr\_account. Credit\_History cr\_history. Purpose $pr. Savings\_accounts\_bonds sab. Present\_empolyment\_since pes.

Personal\_status\_and\_sex pse. Other\_debtors\_guarantors odg. Present\_residence\_since prs. Property prp. Age\_in\_years aiy.

Other\_instalment\_plans oip. Housing hsg. Job job. Telephone tel. Foreign\_worker $fw.

;

**run**;

**QUESTION 8**

##### Fix the program

The results should demonstrate that 254 observations were read, where 500 were expected.

The *print* procedure should show that only odd-numbered observations have been read. The log should report that the last data subject observation is a 'lost card'.

The credit data description states that the last variable is blank if the answer is 'no' to the subject being a 'foreign worker'.

In your code, you should have included a 1 character wide column for this variable. This means that the SAS data step is 1 character short of those needed to read all the data for a subject (a customer).

The default response of the SAS processor is to read the next line of raw data and to use the first character of this to complete the data set. It then to discards the rest.

###### Self-assessment question

Update the infile statement to correct the problem. When your code runs correctly, paste the entire program in the box below.

**ANSWER 8**

/\* pwd = Present Working Directory\*/

/\* The location for the data file \*/

filename pwd 'D:\Sas\';

**proc** **format**;

value cr\_account **1** = '<£0'

**2** = '£0<=<£200'

**3** = '>=£200'

**4** = 'no cheque amount'

;

value cr\_history **0** ='no credits taken/all credits paid back duly'

**1** ='all credits at this bank paid back duly'

**2**='existing credits paid back duly till now'

**3**='delay in paying off in the past'

**4**='critical account/other credits existing (not at this bank)'

**9**='missing'

;

value $purpose **0**='car (new)'

**1**='car (used)'

**2**='furniture/equipment'

**3**='radio/television'

**4**='domestic appliances'

**5**='repairs'

**6**='education'

**7**='vacation'

**8**='retraining'

**9**='business'

X='others'

;

value savings\_accounts\_bonds **1**= '< £100'

**2**= '£100 <= < £500'

**3**='£500 <= < £1000'

**4**='>= £1000'

**5**='unknown/no savings account'

;

value present\_employment\_since **1**='unemployed'

**2**='< 1 year'

**3**='1 <= < 4 years'

**4**='4 <= < 7 years'

**5**='>= 7 years'

;

value personal\_status\_and\_sex **1**='male :divorced/separated'

**2**='female:divorced/separated/married'

**3**='male :single'

**4**='male :married/widowed'

**5**='female:single'

**9**='missing'

;

value other\_debtors\_guarantors **1**='none'

**2**='co-applicant'

**3**='guarantor'

**9**='missing'

;

value present\_residence\_since **4**='4 years or more'

**9**='missing'

;

value property **1**='real estate'

**2**='if not 1: building society savings agreement/life insurance'

**3**='if not 1/2: car or other'

**4**='unknown /no property'

;

value age\_in\_years **999**=' missing'

;

value other\_instalments **1**='bank'

**2**='stores'

**3**='none'

**9**='missing'

;

value hhousing **1**='rent'

**2**='own'

**3**='for free'

**9**='missing'

;

value job **1**='unemployed'

**2**='unskilled'

**3**='skilled employee/official'

**4**='management/self-employed/highly qualified employee/officer'

**9**='missing'

;

value tel **1**='yes, registered under the customers name'

**2**='no'

**9**='missing'

;

value $foreign\_worker **1**='yes'

other = ' '

;

**run**;

**data** CREDIT;

infile

pwd(credit.dat)

missover

;

input

ID\_Number **1**-**3** /\*1-3 \*/

Cheque\_Account **5** /\*5 numeric variable\*/

Duration\_in\_months **7**-**8** /\*7-8 non standard data \*/

Credit\_History **10** /\*10 numeric variable \*/

Purpose $ **12** /\* 12 numeric variable \*/

Credit\_Amount **14**-**18** /\* 14-18 numeric variable \*/

Savings\_accounts\_bonds **20** /\* numeric variable \*/

Present\_empolyment\_since **22** /\*22 numeric variable \*/

Instalment\_rate\_income **24** /\*24 \*/

Personal\_status\_and\_sex **26** /\* 26 numeric variable\*/

Other\_debtors\_guarantors **28** /\* 28 numeric variable\*/

Present\_residence\_since **30** /\* 30 numeric and character variable \*/

Property **32** /\* 32 numeric variable \*/

Age\_in\_years **34**-**36** /\*34-36 numeric data \*/

Other\_instalment\_plans **38** /\* numeric data \*/

Housing **40** /\* numeric data \*/

Number\_of\_existing\_credits **42** /\* 42 \*/

Job **44** /\* 44 numeric data \*/

Number\_of\_dependents **46** /\*46\*/

Telephone **48** /\*48 numeric data \*/

Foreign\_worker $ **50** /\*50 numeric\*/

;

label

ID\_Number = 'ID Number'

Cheque\_Account = 'Cheque Account'

Duration\_in\_months = ' Duration in months'

Credit\_History = 'Credit History'

Purpose = 'Purpose'

Credit\_Amount = ' Credit Amount'

Savings\_accounts\_bonds = 'Saving accounts/bonds'

Present\_empolyment\_since = 'Present empolyment since'

Instalment\_rate\_income = 'Instalment rate % income'

Personal\_status\_and\_sex = 'Personal stauts and sex'

Other\_debtors\_guarantors = 'Other debtors/guarantors'

Present\_residence\_since = 'Present residence since'

Property 'Property'

Age\_in\_years 'Age in years'

Other\_instalment\_plans = 'Other instalment plans'

Housing = 'Housing'

Number\_of\_existing\_credits = ' Number of existing credits'

Job = 'Job'

Number\_of\_dependents = 'Number of dependents'

Telephone = 'Telephone'

Foreign\_worker = 'Foreign worker'

;

/\* format \*/

format

Cheque\_Account cr\_account.

Credit\_History cr\_history.

Purpose $purpose.

Savings\_accounts\_bonds savings\_accounts\_bonds.

Present\_empolyment\_since present\_employment\_since.

Personal\_status\_and\_sex personal\_status\_and\_sex.

Other\_debtors\_guarantors other\_debtors\_guarantors.

Present\_residence\_since present\_residence\_since.

Property property.

Age\_in\_years age\_in\_years.

Other\_instalment\_plans other\_instalments.

Housing housing.

Job job.

Telephone tel.

Foreign\_worker $foreign\_worker.

;

**run**;

**proc** **print** data = CREDIT label noobs;

**run**;

**proc** **print** data = CREDIT(obs=**10**);

format

Cheque\_Account

Credit\_History

Purpose

Saving\_accounts\_bonds

Present\_employment\_since

Personal\_status\_and\_sex

Other\_debtors\_guarantors

Present\_residence\_since

Property

Age\_in\_years

Other\_instalment\_plans

Housing

Job

Telephone

Foreign\_worker

;

**run**;