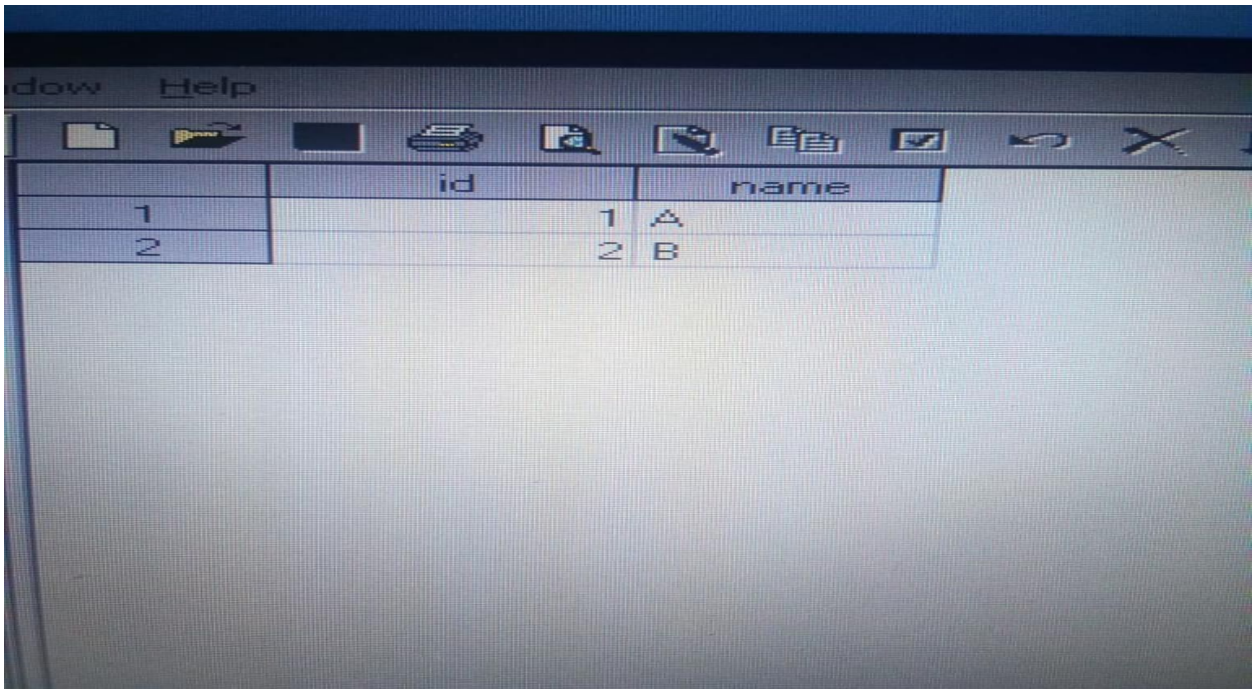
CLASS-34

Reading raw data

```
Data a;
Input id name $; // input statement used to declare variables //
cards;
1 A
2 B
;
Run;
```

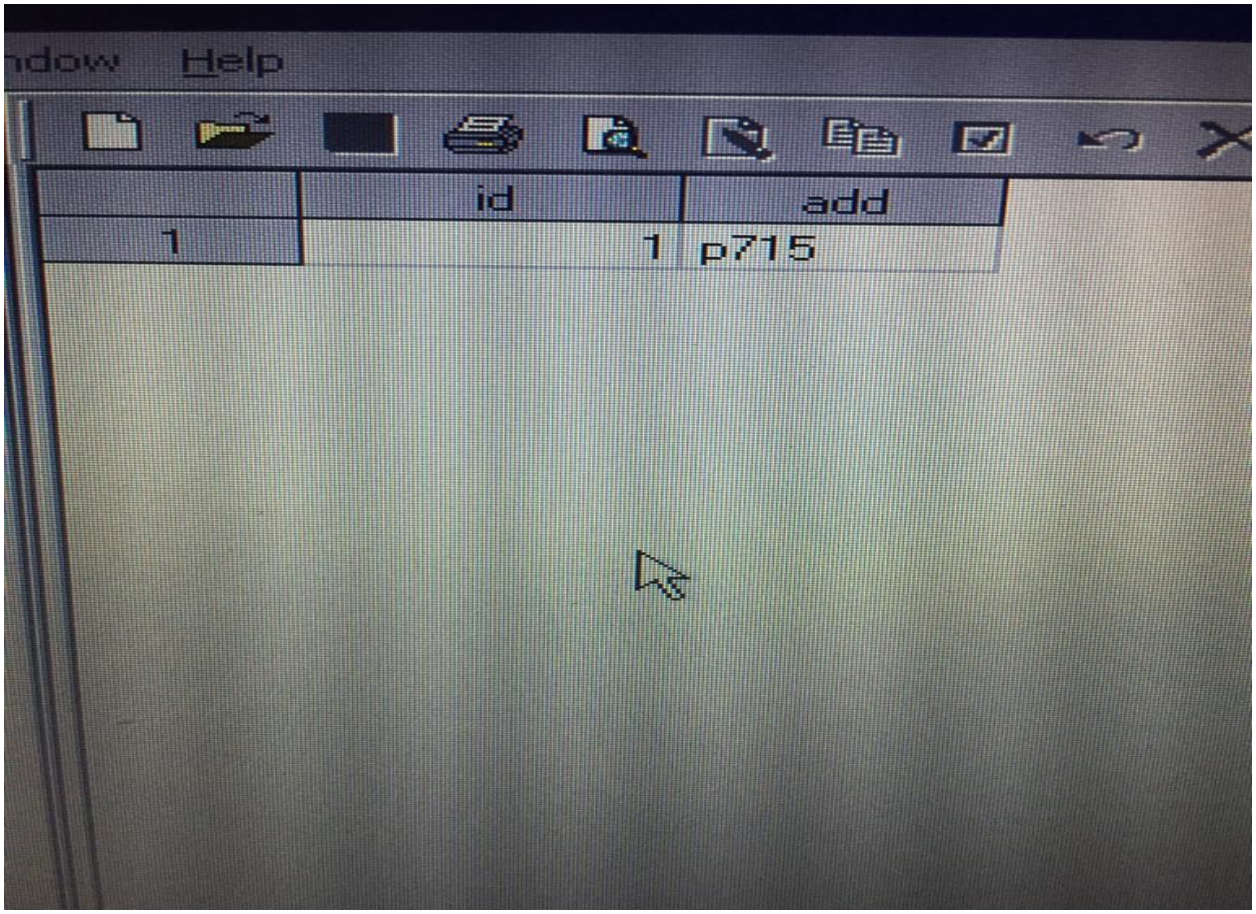


DATALINES4 : used to read in-built semi-colons in data.

Code: **without datalines4**

```
Data a;
Input id add $;
datalines4;
1 p715
2 B;312  // semi-colon between raw values. In this case, SAS will consider it as a terminator and will not read this value.//
```

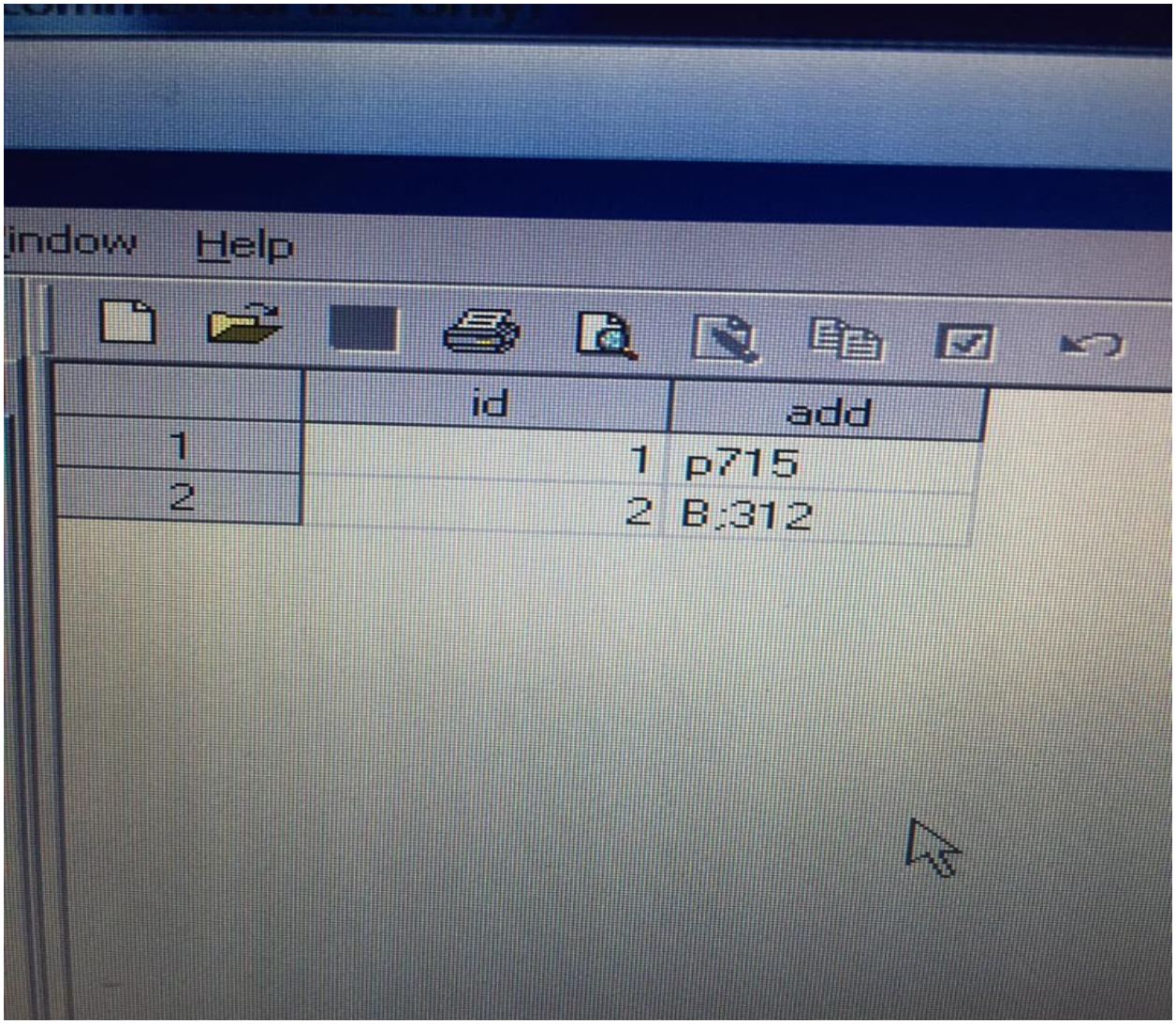
;;
Run;



**********;
Code: **with datalines4**

Data a;
Input id add $;
datalines4;
1 p715
2 B;312 // semi-colon between raw values. So, in this case we use the option datalines4 //
;;;; // after using datalines4 , we will terminate with 4 semi-colons. Now, even if a semi-colon comes between values, it will read it as single value.//

Run;



Data a;
Infile” …….. Folder path ……..a.txt”; // to read flat files //
Input id name $;
Run;

**********;

Filename sa “ …….Folder path ……..a.txt”; // in case we want to use the same files multiples times, then everytime we have to give the path in case of infile. But here we have created alias like “sa”.

So, in this case we don't have to give the path everytime. We can just change name of the file and path will remain same. It will work as global statement //

```
Data a;
Infile sa; // like here , we can just change the name of the file and path we don't have to copy everytime //
Input id name $;
Run;

Filename sa clear;
```

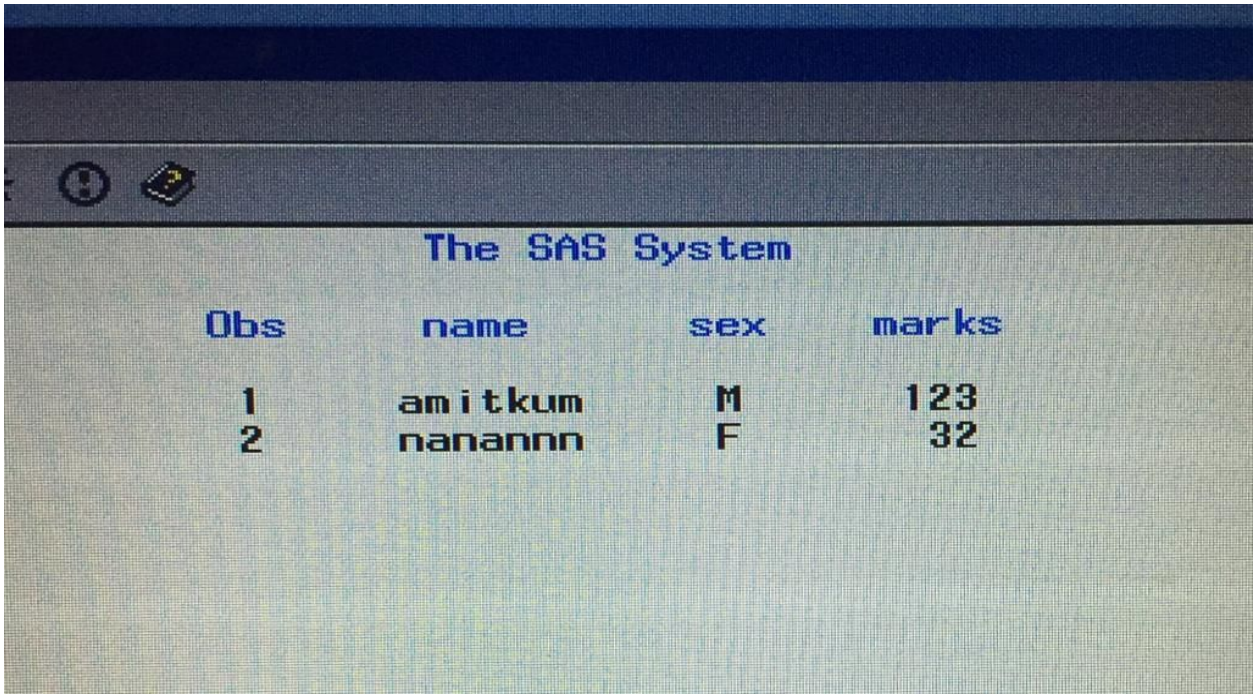
COLUMN POINTERS : Read the data column wise. Helps in flexible reading and can read in any order.

*The two basic column pointer controls are;

1.@n: Helps in the reading of the source data at a specified number of column
2. +n: Helps in the reading of the source data at plus n columns.

**********column pointers, help in flexible reading;

```
data amit;
input @1name $7. @9 sex $1. @10 marks;    // @1 name $7 : start from 1st column, read the name variable of length 7 …  @9 means come to 9th column, 'sex $1' : read the sex variable of length 1…
 @10 marks : come to 10th column, read the variable marks…….
Note :column pointer is always written before variable name i.e. @10 is column pointer and it is written before variable 'marks'  //
datalines;
amitkum M123
nanannn F32  // even if we do not give space between the values i.e. nanannF32, then also with the help of column pointer we can read the values separately , we just need to change the position in input statement i.e. @8sex $1. @9 marks  //
;
run;
proc print data=amit;
run;
```



The SAS System

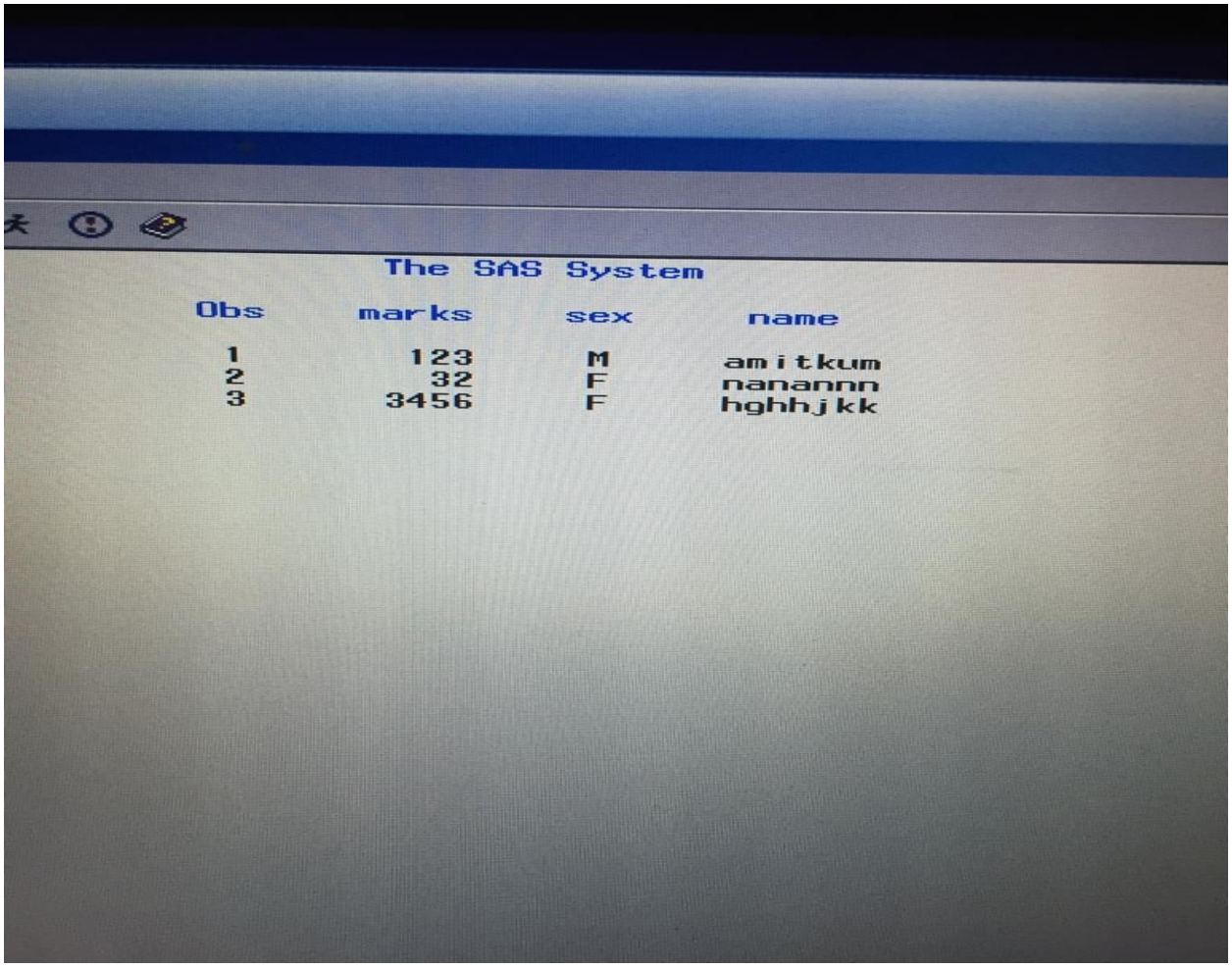
Obs	name	sex	marks
1	amitkum	M	123
2	nanannn	F	32

---

* **The column pointer control @ can be used to read the variables in any order**:(but we should know the starting and end point of the variable)

```
data amit;
input @10 marks  @9 sex $1. @1name $7. ;
datalines;
amitkum M123
nanannn F32
hghhjkk F3456
;
run;

proc print data=amit;
```

run;

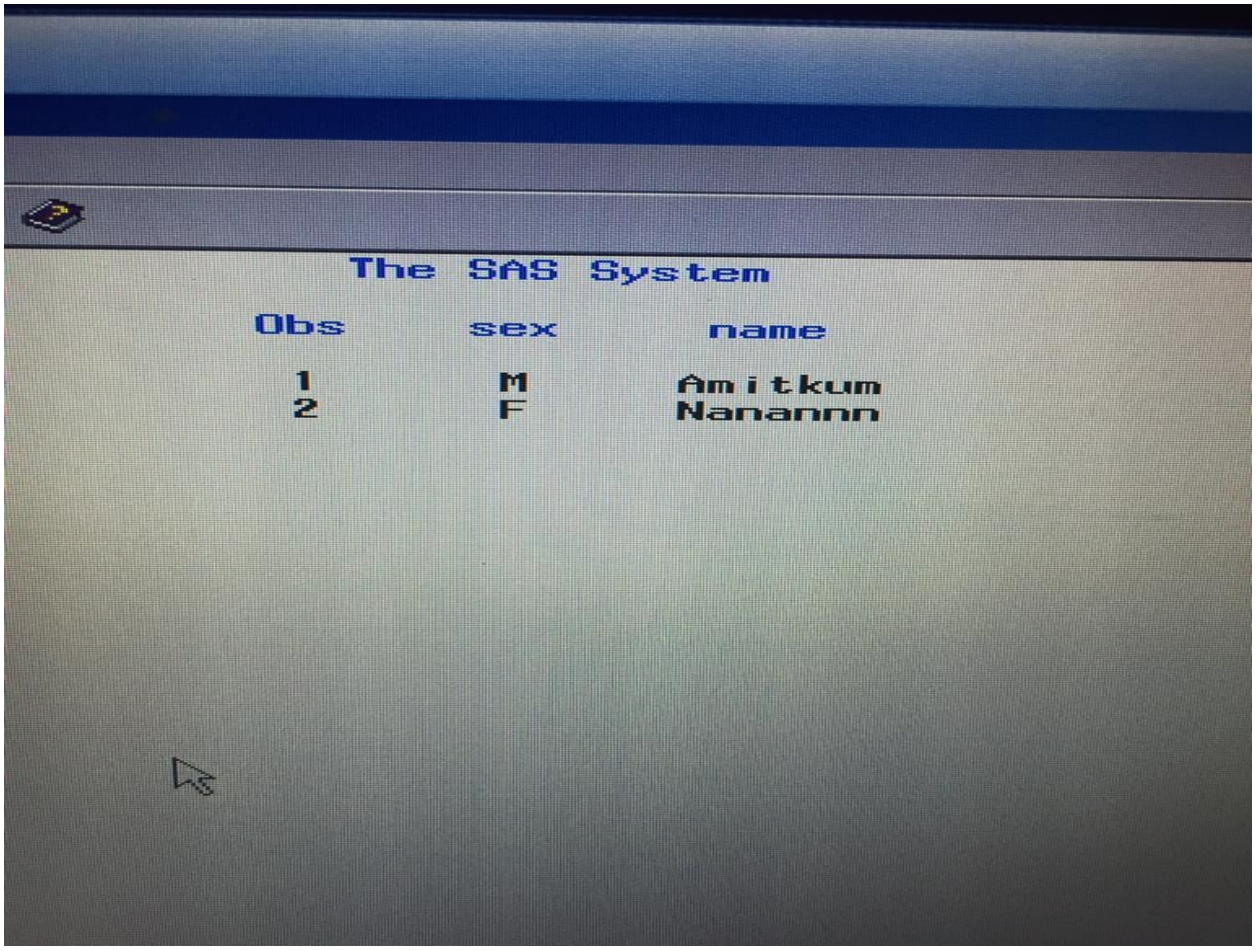


The SAS System

Obs	marks	sex	name
1	123	M	amitkum
2	32	F	nanannn
3	3456	F	hghhjkk

********** **column pointers, you can read any variable, just like keep and drop;**

```
Data amit;
Input @9 sex $1. @1name $7. ;  // suppose we want to read only sex and name from the given values
//
Datalines;
Amitkum M123
Nanannn F32
‘
Run;

Proc print data=amit;
Run;
```

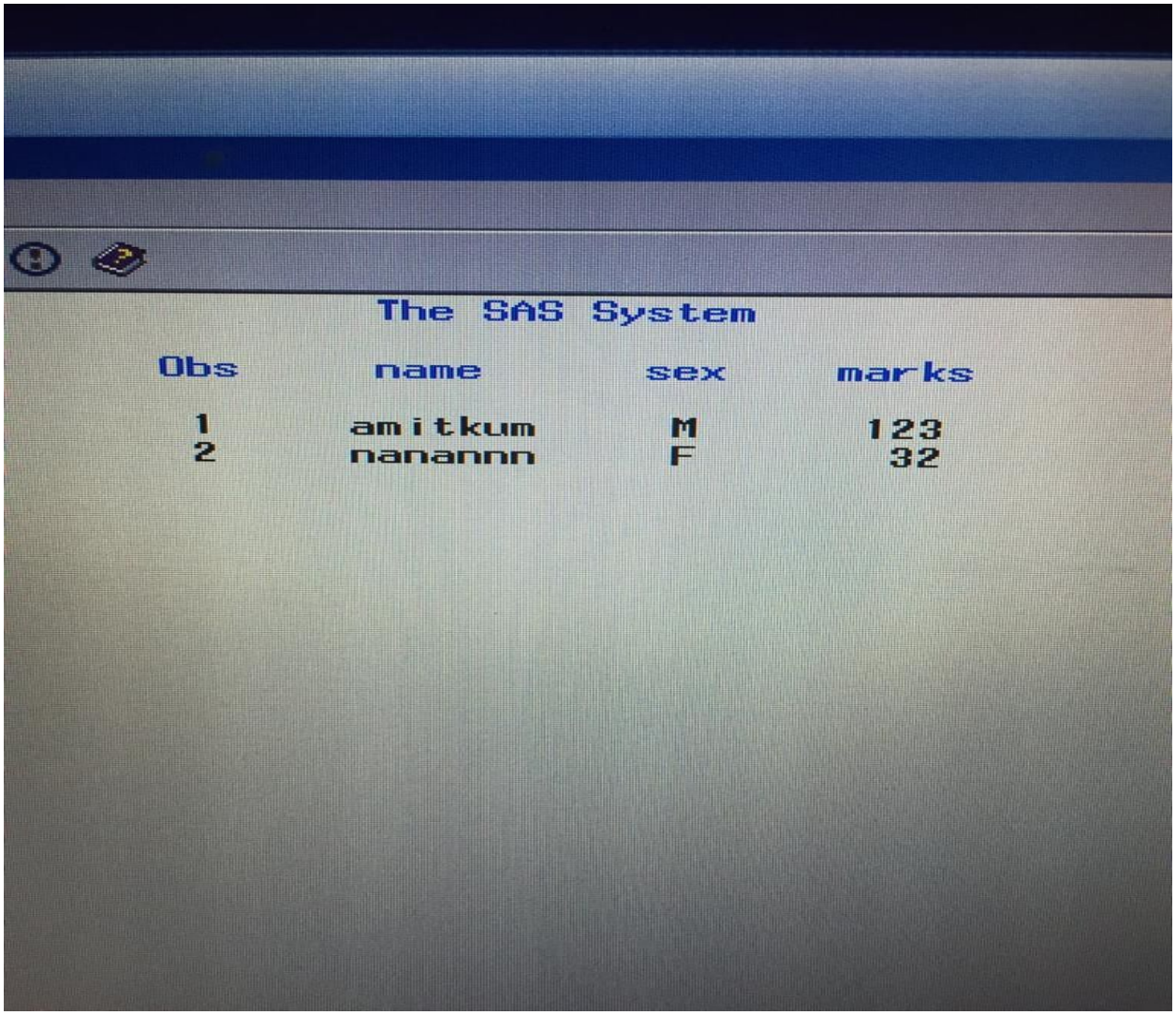


---

* **The +n pointer**

```
data amit;
input @1name $7. +1 sex $1. @10marks;
```
// we can jump the columns with "'+" pointer. Like here, after reading the value of name till 7th column, the pointer is now at the beginning of 8th column.. +1 will move the pointer to 9th column and will read the sex value. So, by this we don't have to use @ pointer, we can jump the columns with +n pointer. //

```
datalines;
amitkum M123
nanannn F32
;
run;
proc print data=amit;
run;
```

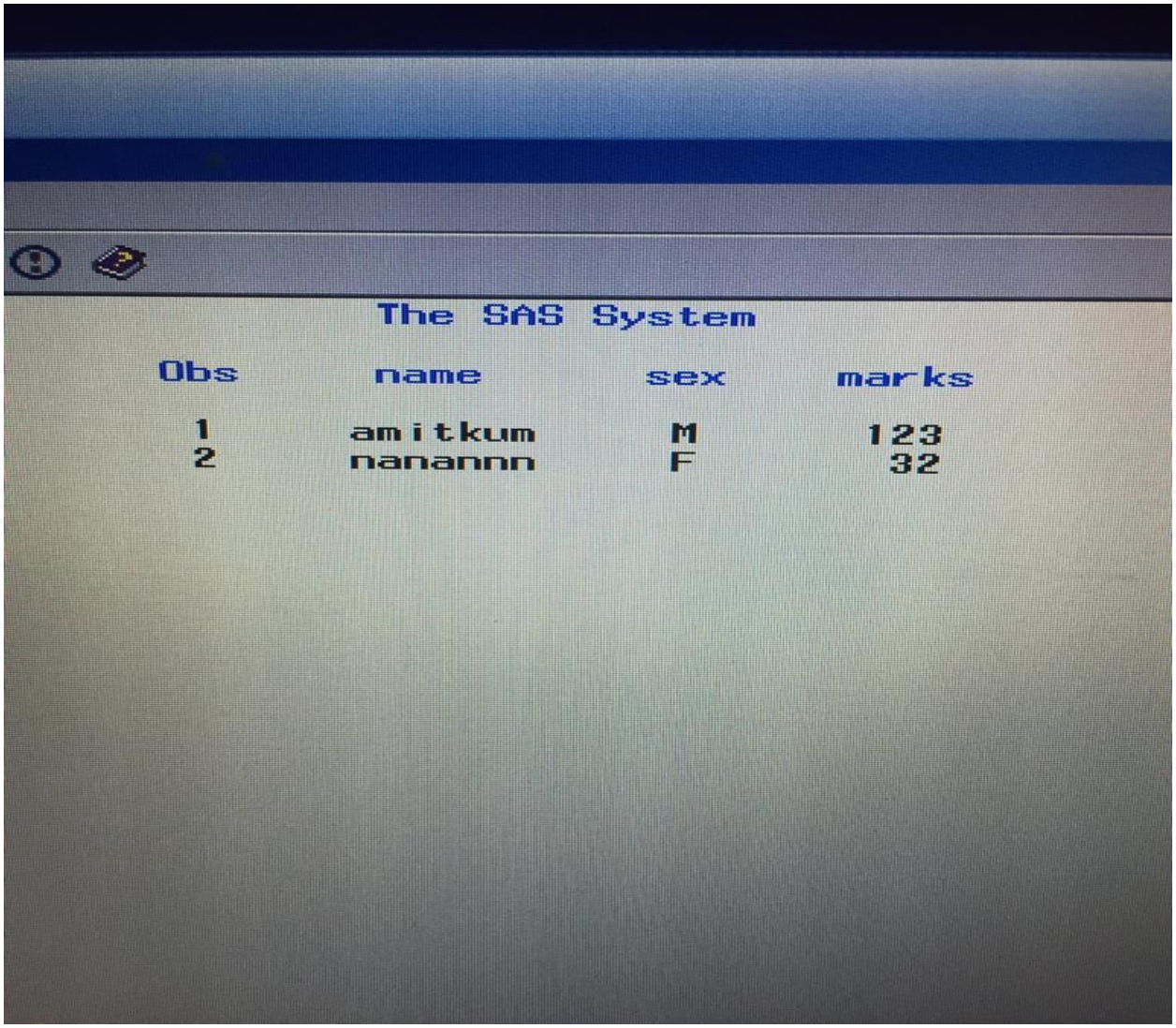


The SAS System

Obs	name	sex	marks
1	amitkum	M	123
2	nanannn	F	32

******** +(-n) pointer can be used to skip columns in backward direction*******;**

```
data amit;
input @1name $7. +2 marks 3. +(-4) sex$ 1. ;  // with +(-n) pointer we can jump the columns in backward direction //
datalines;
amitkum M123
nanannn F321
;
run;
proc print data=amit;
run;
```



Double Trailing (@@) :* The @@ trailing helps in the reading of multiple observation in single line;

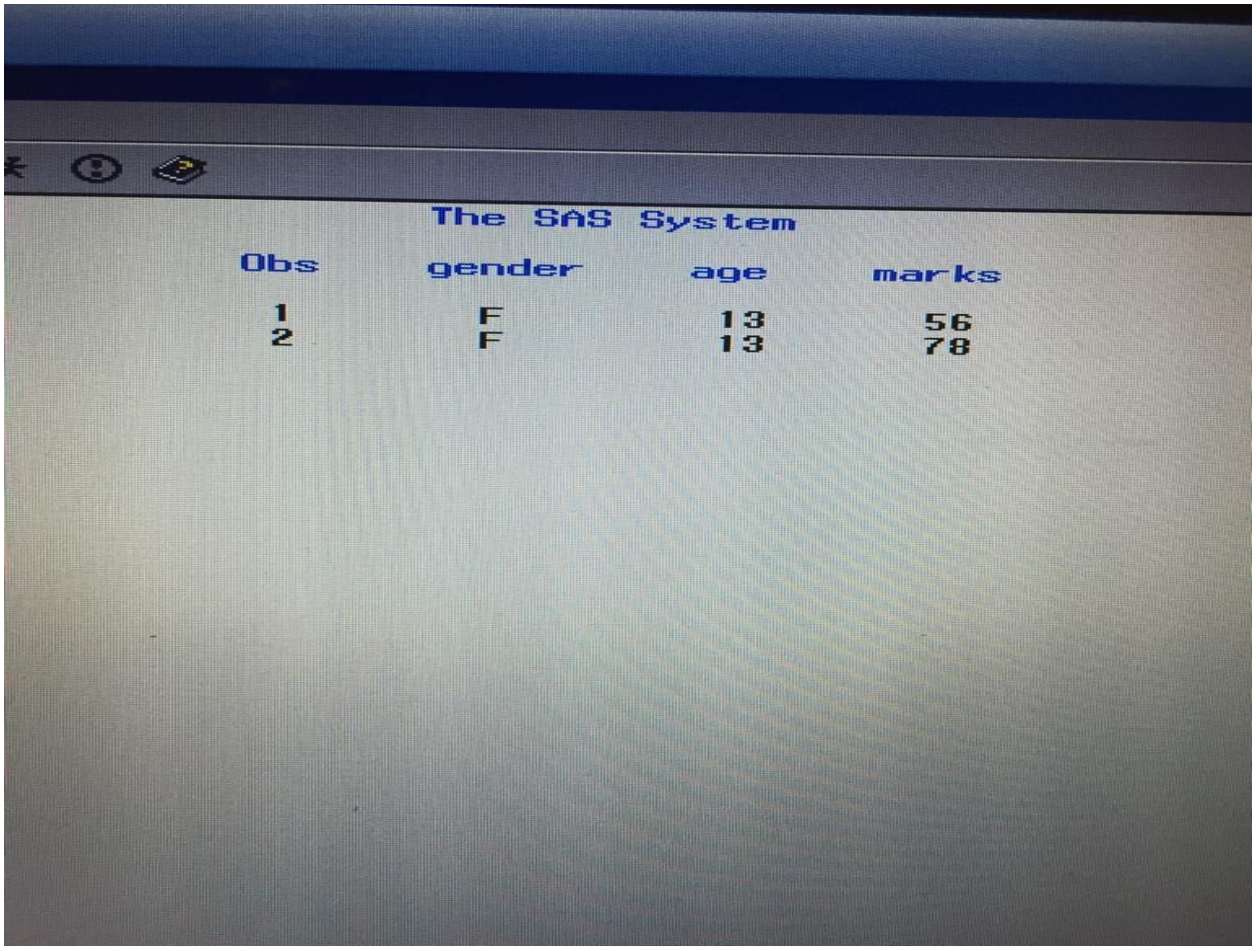
```
data amit;
input name $ age building $ @@;   // if we do not use @@ , it will read only one observation i.e. amit 23 jvt  corresponding to name , age and building //
datalines;
amit 23 jvt preeti 24 suncty kana 56 kendriya  // multiple records in single line //
;
Run;

proc print data=amit;
run;
```

The SAS System

Obs	name	age	building
1	amit	23	jvt
2	preeti	24	suncty
3	kana	56	kendriya

SINGLE TRAILING (@) : The use of single @ is to hold the line, it is used in more than one input statement,the pointer basically hold the line after reading the values, it holds the value and checks the condition. Single trailing is used at the end of input statement.

```
data amit;
input @1 gender $1. @;  // @1 is column pointer , gender$1. is char informat , @ is single trailing //
if gender ne 'F' then delete;  // single trailing will read data in memory, if gender is not equal to female, it will delete the value at that point only and will not read the full data, means if we see the program, it will not read the value of age and marks if gender condition is not satisfied . this is called efficient reading //
input @3 age @5 marks; // if condition is true then delete, if condition is false then at position @3 and @5 print the values of age and marks respectively //
datalines;

F 13 56
M 12 78
F 13 78
M 56 90
;
run;
proc print data=amit;
run;
```

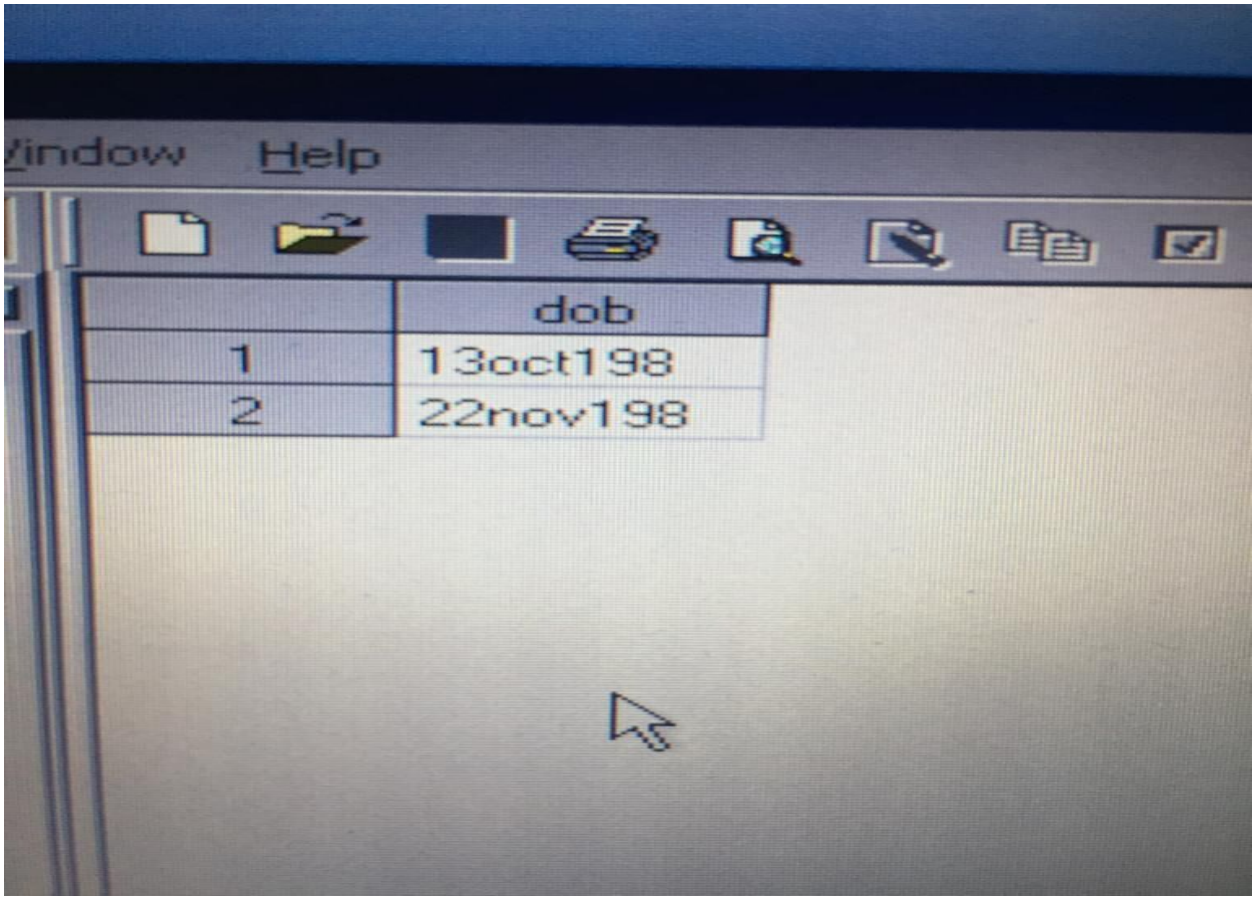


The SAS System

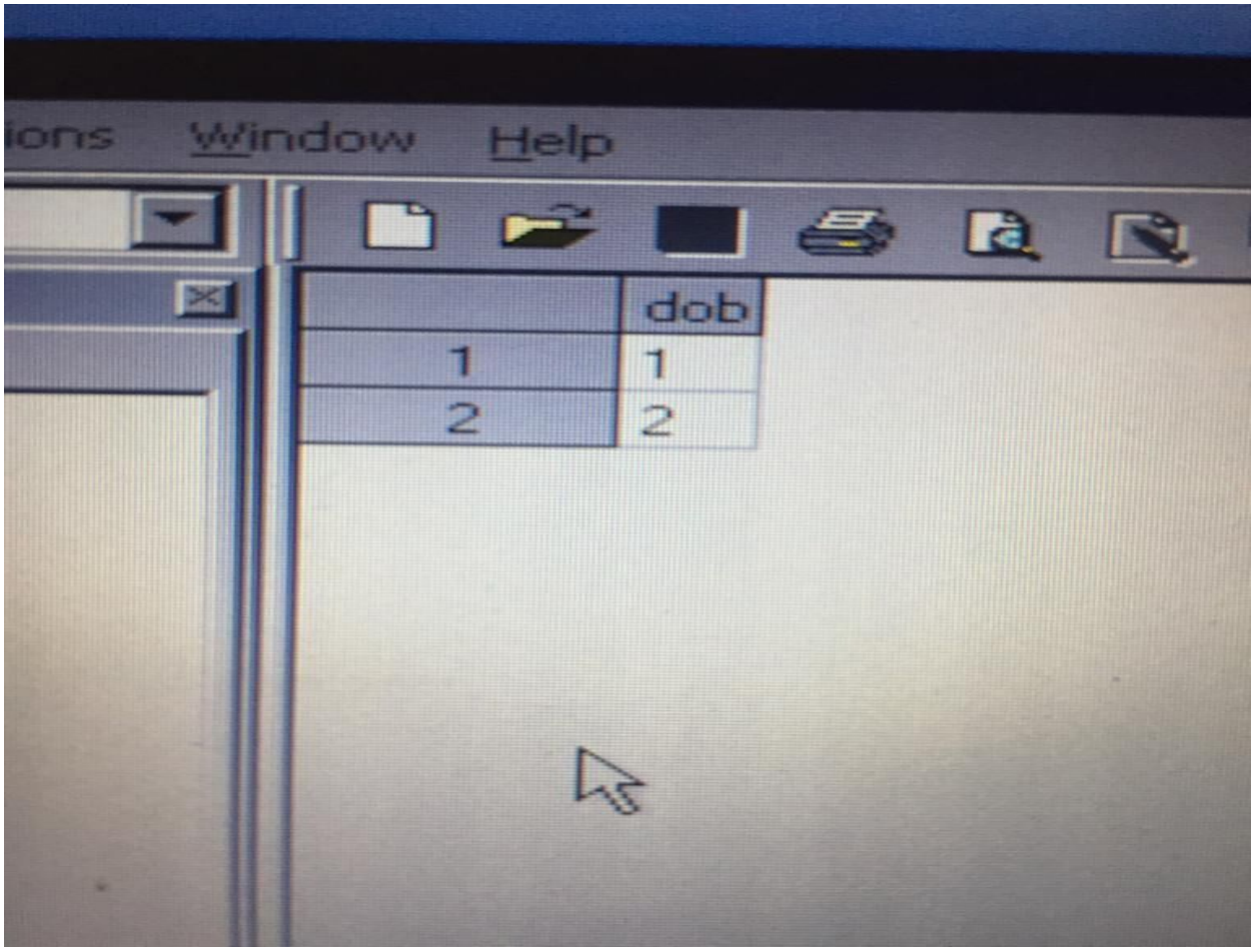
Obs	gender	age	marks
1	F	13	56
2	F	13	78

Revision of formats and informats :

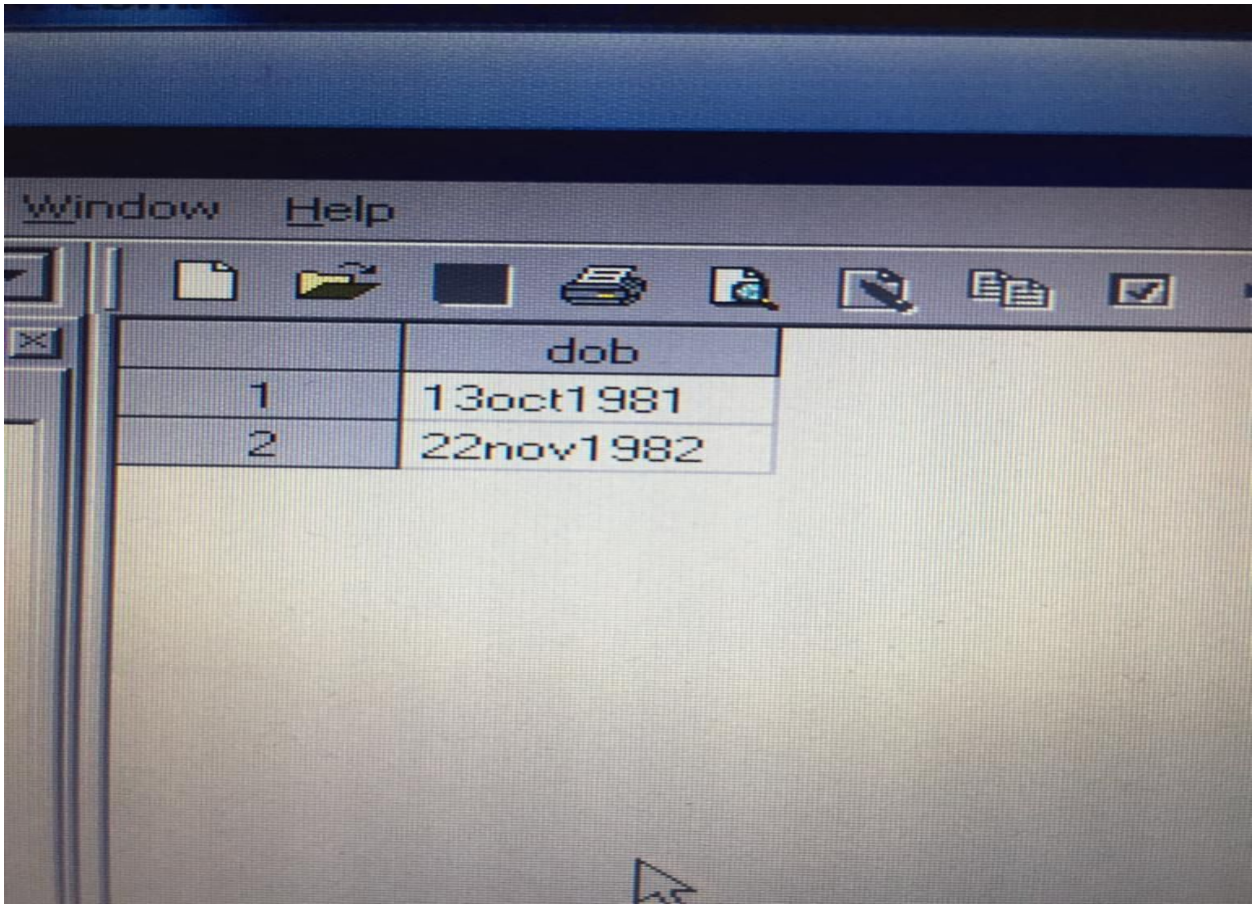
```
Data a;
Input dob $;  // character variable date of birth is declared and will read default 8 bytes and will truncate the value if the value will exceed 8 bytes.//
Cards;
13oct1981
22nov1982
;
Run;
```




```
************************;
Data a;
Input dob $9;  // $9 means read the 9th column, its as good as @9 //
Cards;
13oct1981
22nov1982
;
Run;
```




```
*************************;
Data a;
Input dob $1-9;  // $1-9 means read from 1 to 9 bytes //
Cards;
13oct1981
22nov1982
;
Run;
```

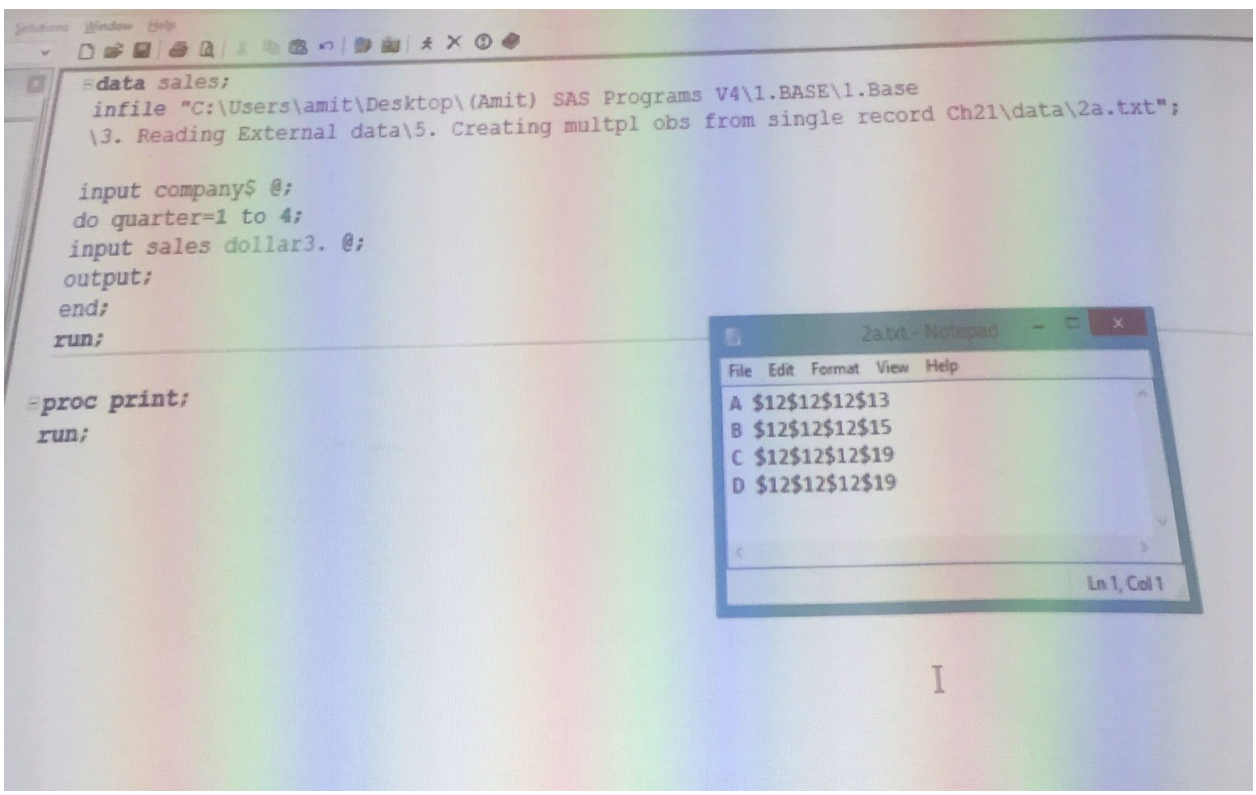



```
*************************;
Data a;
Input dob $9.;  // 9. Is an informat. It helps in the reading of the data //
Cards;
13oct1981
22nov1982
;
Run;

*************************;
Data a;
Input dob $date9.;  // string is read with date9. informat  and informat always read the value after
removing format that is applied on it. //
Cards;
13oct1981
22nov1982
;
Run;

*************************;
```

```
Data a;
Input dob $ date9.;  // with this informat date9. value is read and value will comes in numbers //
Format dob ddmmyy10.;  // with format we are writing the value and the value will come out to be separated by slash //
Cards;
13oct1981
22nov1982
;
Run;
```

Example of single trailing:

In the raw file we have the data of four companies of their four quaters. $12 means in 1st quater the revenue was of 12 million.
2nd quater the revenue was of 12 million.
3rd quater the revenue was of 12 million.
4th quater the revenue was of 13 million.



From program : If we will read $12 with dollar3. format, it will remove the $ sign with 12.

Loop will run 4 times. From single trailing we will make 4 records from 1 record and total 16 records will become.

```
Data sales;
Infile " …. Folder path ….."
Input company$ @;  // A value will come in company, '@' will hold the line //
Do quater=1 to 4 ; // Loop will run 4 times //
Input sales dollar3. @; // If we will read $12 with dollar3. format, it will remove the $ sign with 12.  And '@' will hold the line till the last value. //
Output;
End;
Run;
```

The SAS System

Obs	company	quarter	sales
1	A	1	12
2	A	2	12
3	A	3	12
4	A	4	13
5	B	1	12
6	B	2	12
7	B	3	12
8	B	4	15
9	C	1	12
10	C	2	12
11	C	3	12
12	C	4	19
13	D	1	12
14	D	2	12
15	D	3	12
16	D	4	19

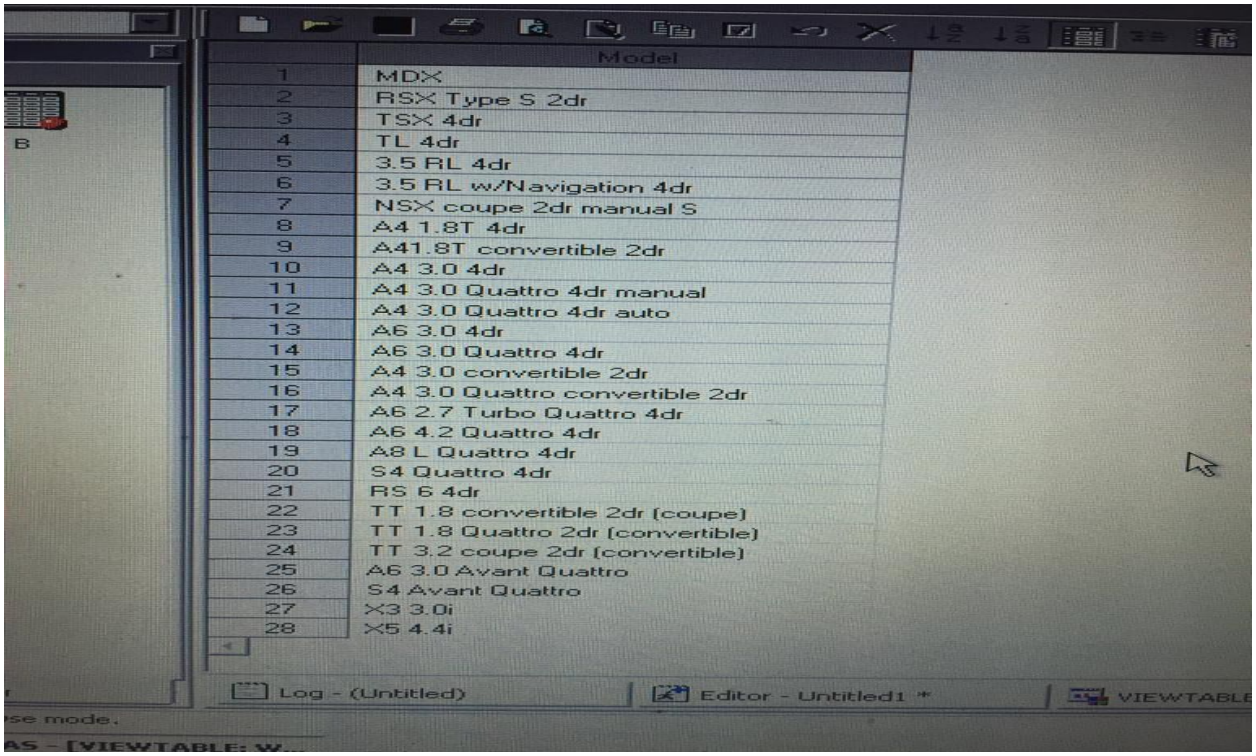
Note: If we see the data, it is getting transposed.

Syslast Macro-variable

Data a b c (keep=model);
Set sashelp.cars;
Run;

%put &syslast; // syslast macro-variable will show the last dataset created //

Data new;
Set &syslast; // here syslast will put the value from the last dataset created i.e. “c” , therefore in new only model variable will come //
Run;



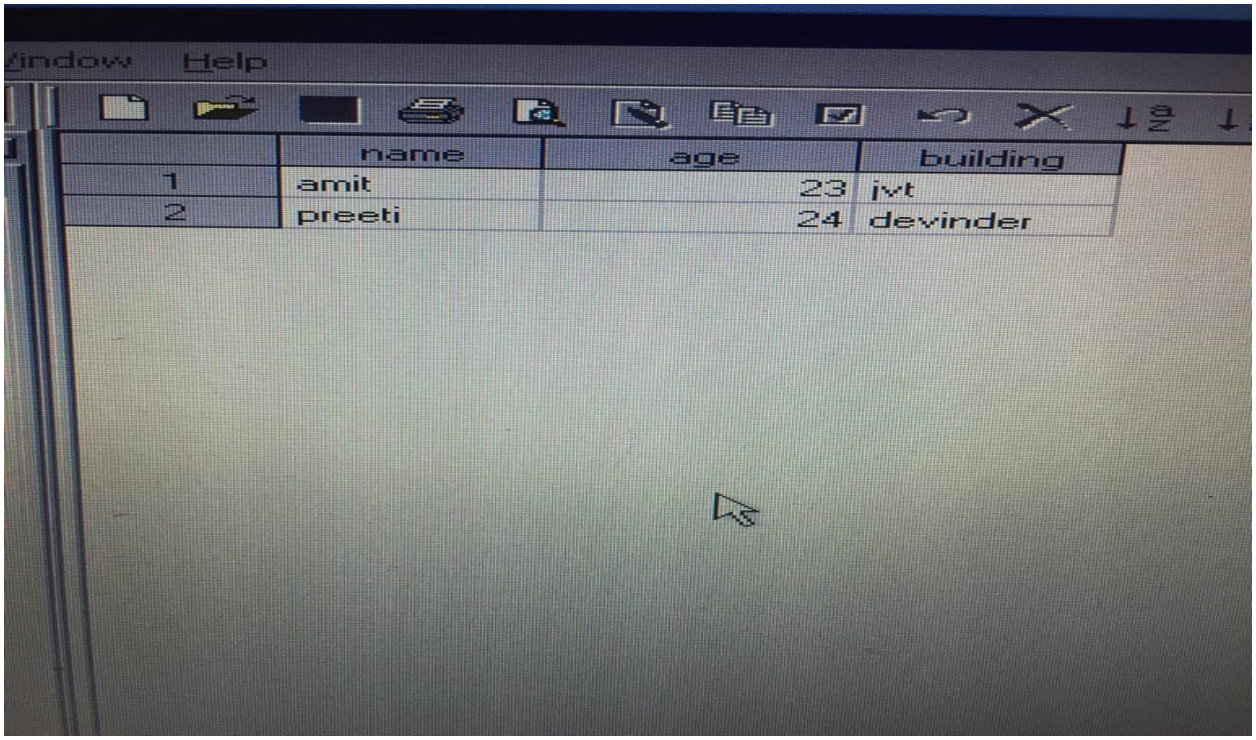
	Model
1	MDX
2	RSX Type S 2dr
3	TSX 4dr
4	TL 4dr
5	3.5 RL 4dr
6	3.5 RL w/Navigation 4dr
7	NSX coupe 2dr manual S
8	A4 1.8T 4dr
9	A4 1.8T convertible 2dr
10	A4 3.0 4dr
11	A4 3.0 Quattro 4dr manual
12	A4 3.0 Quattro 4dr auto
13	A6 3.0 4dr
14	A6 3.0 Quattro 4dr
15	A4 3.0 convertible 2dr
16	A4 3.0 Quattro convertible 2dr
17	A6 2.7 Turbo Quattro 4dr
18	A6 4.2 Quattro 4dr
19	A8 L Quattro 4dr
20	S4 Quattro 4dr
21	RS 6 4dr
22	TT 1.8 convertible 2dr (coupe)
23	TT 1.8 Quattro 2dr (convertible)
24	TT 3.2 coupe 2dr (convertible)
25	A6 3.0 Avant Quattro
26	S4 Avant Quattro
27	X3 3.0i
28	X5 4.4i

Forward slash “/” and hash “#”

Data amit;

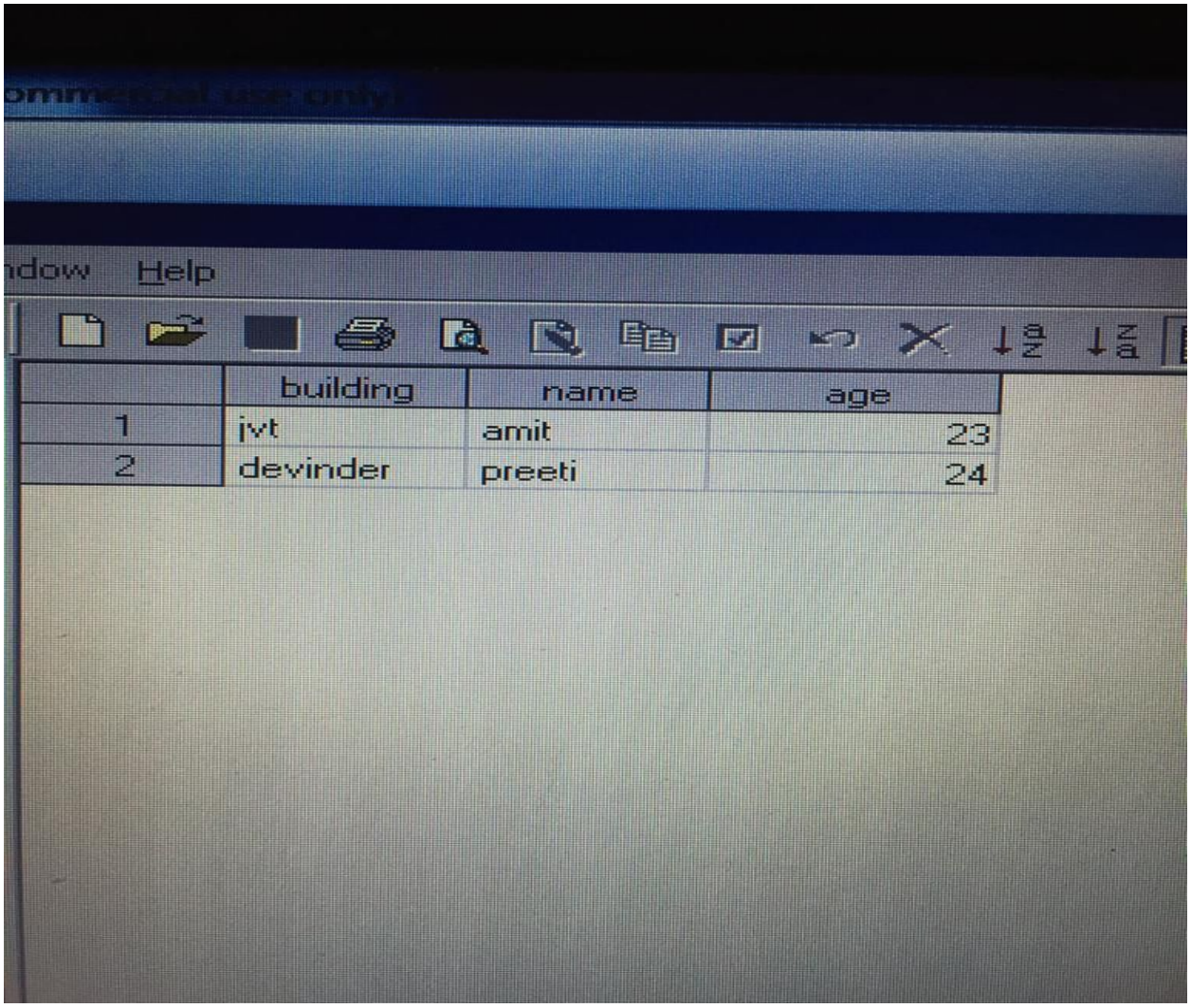
Input name $ age / building $; // forward slash helps SAS to move to the next line , code shows that name and age are in the 1st line and building is in the next line //
Datalines;
Amit 23

Jvt
Preeti 24
Devinder
;
Run;



Data amit;
Input #2 building $ #1 name $ age ; // #n helps SAS to read the values per line of the raw data , code will read 2nd line first i.e. jvt and after tht name and age //
Datalines;
Amit 23
Jvt
Preeti 24
Devinder
;
Run;

Proc print data = amit ;
run;



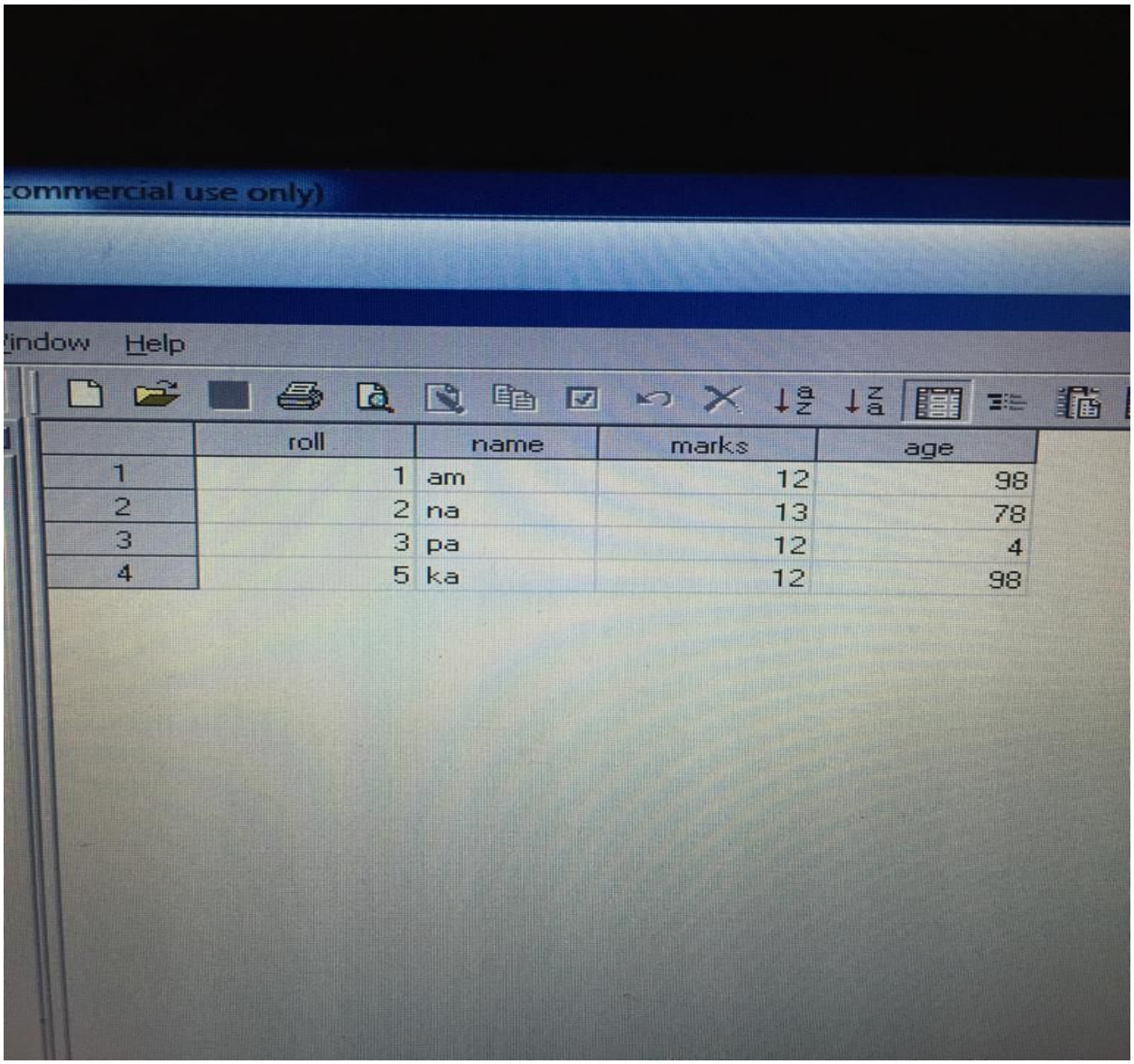
Missover Option

* The missover option with infile statement helps in the reading of data values of a line ,
if the values of a variable is missing then the pointer jumps to next line after setting the value to missing. It will only work if the last value is missing not in between values .

* **without missover;**

```
data amit;

input roll name$ marks age;
datalines;
1 am 12 98
2 na 13 78
3 pa 12
4 sa 78
```

```
5 ka 12 98
;
run;
proc print data=amit;
run;
```

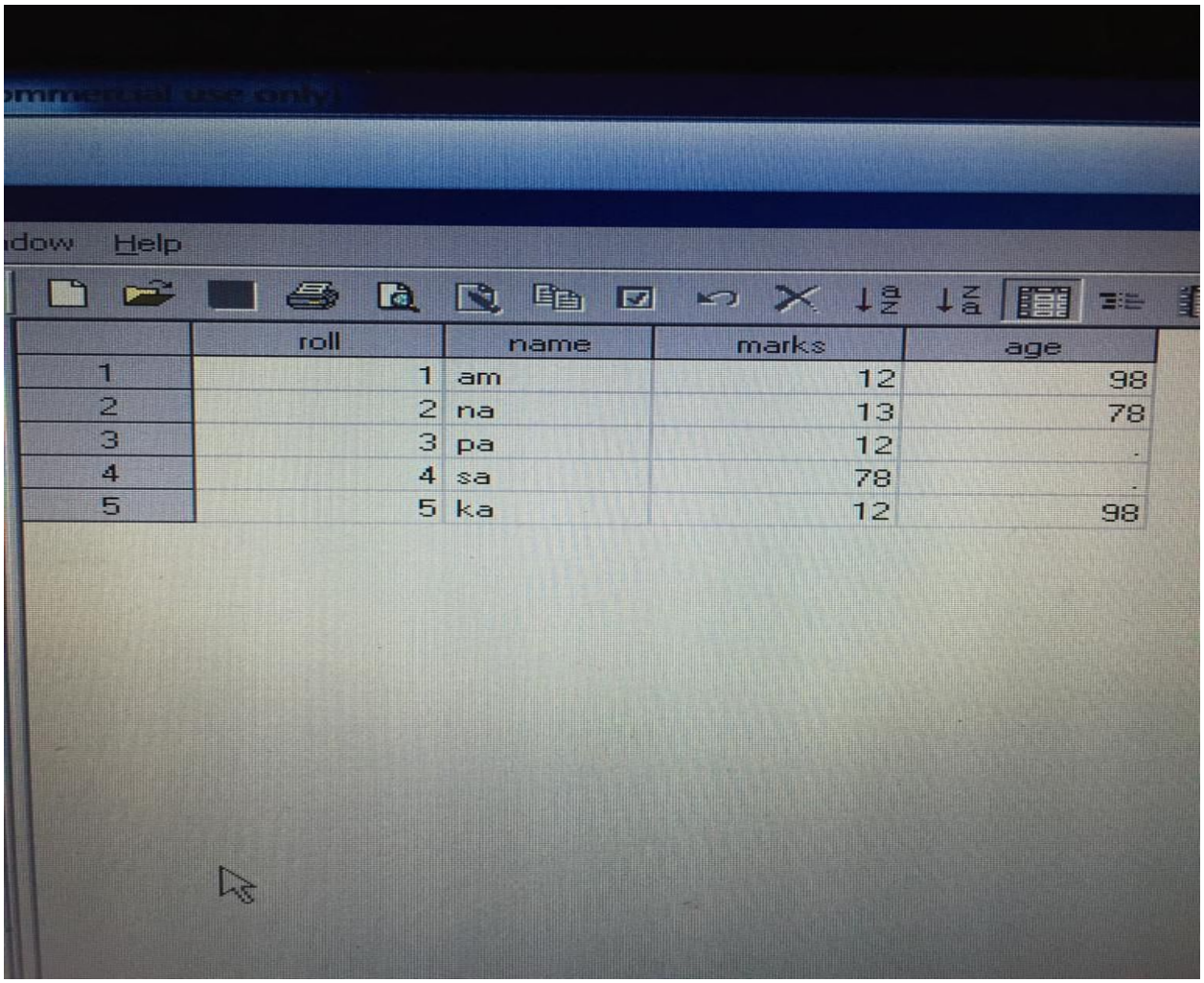


	roll	name	marks	age
1	1	am	12	98
2	2	na	13	78
3	3	pa	12	4
4	5	ka	12	98

*** with missover;**

```
data amit;
infile datalines missover;
```

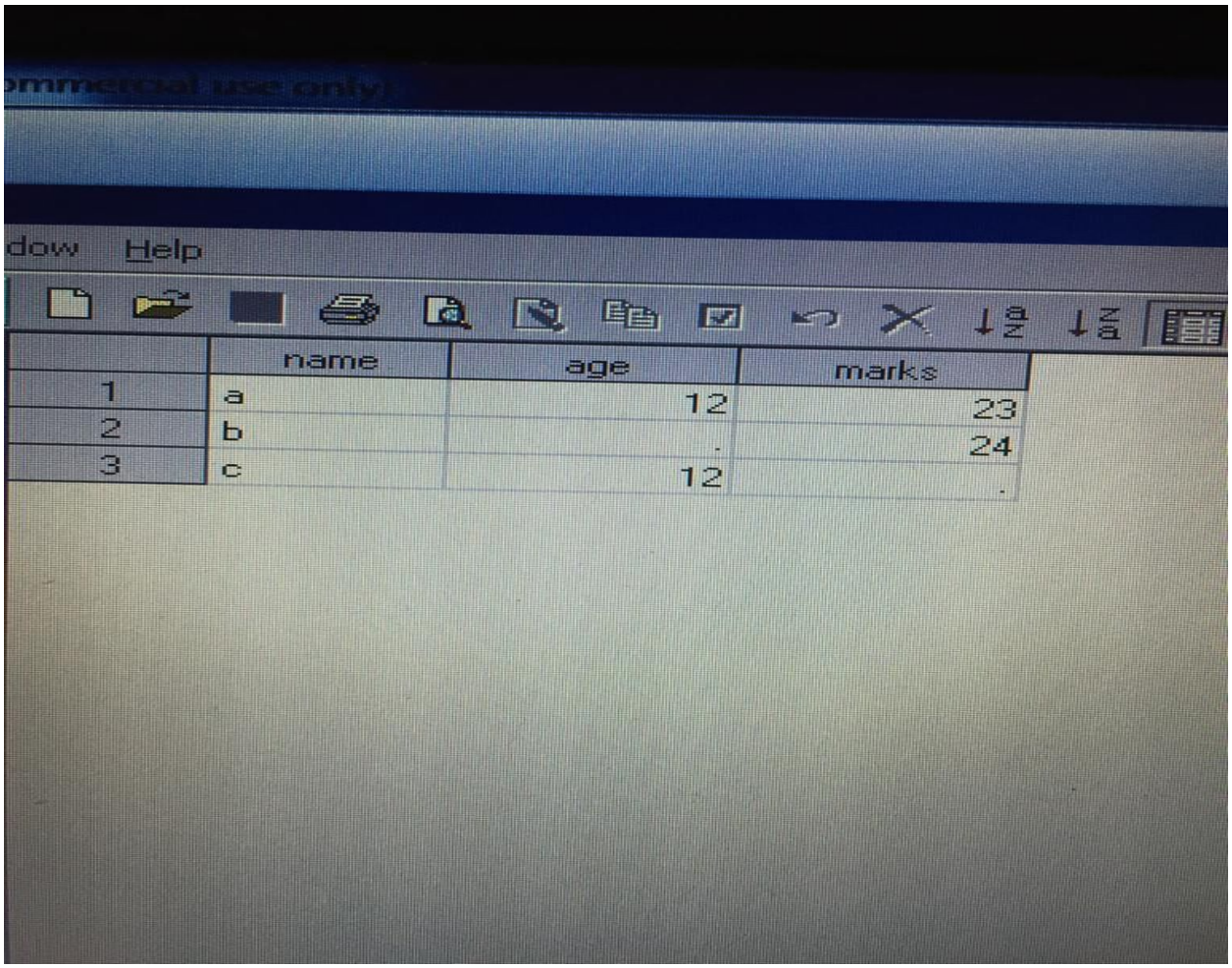
```
input roll name$ marks age;
datalines;
1 am 12 98
2 na 13 78
3 pa 12
4 sa 78
5 ka 12 98
;
run;
proc print data=amit;
run;
```



	roll	name	marks	age
1	1	am	12	98
2	2	na	13	78
3	3	pa	12	.
4	4	sa	78	.
5	5	ka	12	98

```
data x;
infile datalines dsd dlm= "," missover ;
```
// dsd: delimitor sensitive data, it will work on the missing values in between and for that it requires a delimitor and if the last value is missing that will be handled by missover//

```
input name $ age marks;
datalines;
A,12,23
B, ,24
C, 12
;
run;

proc print data=x ;
run;
```



```
Data amit;
Length name $25;
Input name & $ age;  // & will help in reading the data and the delimitor it requires is double space //
Datalines;
```

```
Amt  kumar  singh  29  // double spaces are there between the values //
Preeti  78
;
Run;

Proc print data=amit;
Run;
```

To write raw data : File and Put are used

```
Data _null_ ;
Set sasuser.admit;
File  ‘ D:\amit\cheeku.txt ’ ;
Put name$ 1-3  age  5  sex  $  7; // 3 bytes of name will come //
Run;

Data _null_ ;
Set sasuser.admit;
File  ‘ c:\users\amit\desktop\B82\cheeku.txt ’ ;
Put name$ 1-3  age  5-6  sex  $  8 ;
Run;

Data _null_ ;
Set sasuser.admit;
File  ‘  c:\users\amit\desktop\B82\cheeku.txt ’ ;
Put name$ 1-14  age  19-20;
Run;
```