CLASS 24

```
% let a = amit ;
% let b = kumar ;
% let c = % sysfunc (cat (&a, &b));
% put **&c** ;
```

Sysfunc → Used to call base SAS function in macros.

Explanation Amit → Here, space is not visible if there is space after amit.

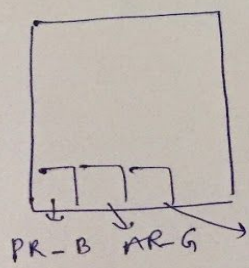
**Amit **

↓

But here space after amit is easily visible. Using ** will print Amit between "*" and the space will easily be seen.

Sysfunc → Bridge between macros and base SAS.

All the base sas functions cannot be called directly. So, in that case sysfunc is written before base SAS function.



→ Home work Question

```
%macro ffawsm;
proc sql noprint;

select distinct state into :n separated by "@ "
from sasuser.frequent_flyers;

select count (distinct state) into : n from
sasuser.frequentflyers ;

%do i=1 %to n;   → This loop is running 33 times.

%let st= %scan (&s ,&i ,"@");

proc sql;                    → This will give different
select distinct membertype into : mt separated      membertype
by "@" from sasuser.frequentflyers where
state = "&st"; → This will keep on changing states
                                like AR, AZ and so on.

select count (distinct membertype ) into :c from
sasuser.frequentflyers where state = "&st";
quit;
        ↓ This will give the count like in
AR we have all three bronze, gold, silver so
the count will be 3.
```

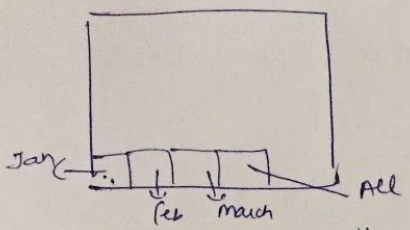
```
%do j=1 %to &c;        → This will run according to count.
%let m = %scan (&mt, &j, "@");
(scan mt, j, where @ is found)
data &st._&m;          → This will give AR_bronze (from st, from m)
                         and so on with diff. states name.
set sasuser.frequentflyers;
where state = "&st" and membertype = "&m";
run;

PROC EXPORT DATA = &st._&m OUTFILE = "
------path------\ff.xls"
    DBMS = EXCEL REPLACE;      → This data set will be exported using this command.
  SHEET = "&st._&m";}
RUN;

%end
%end
%mend ffawsm;
```

```
%ffawsm;
```

EXCEL ENGINE {Import/Export}



libname yo "C: ——————— sale.xls";

data yo.admit; → Import

set sasuser.admit; → Export

run; ② ① This will work as proc import.

---

libname yo clear; → SAS is using excel so ~~we~~ cannot not open the excel, so we need to clear the library.

Note: Without opening the excels we want all the sheets to append in "all" tab.

Problem: we don't know the sheets name and the number of sheets.

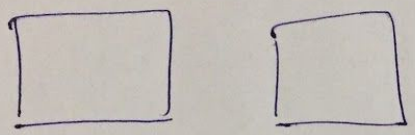
Running ① → will give print of all the sheets present like

→ sheets will print like datasets.

By this we will come to know the names and number of sheet.

$ → This sign indicates that the sheets are excel imported, not the actual data sets.

Running ② program will give two enteries of admit.

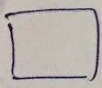


admit admit $ → Excel property

↓

Data imported

Note: If we delete  admit, then in excel,

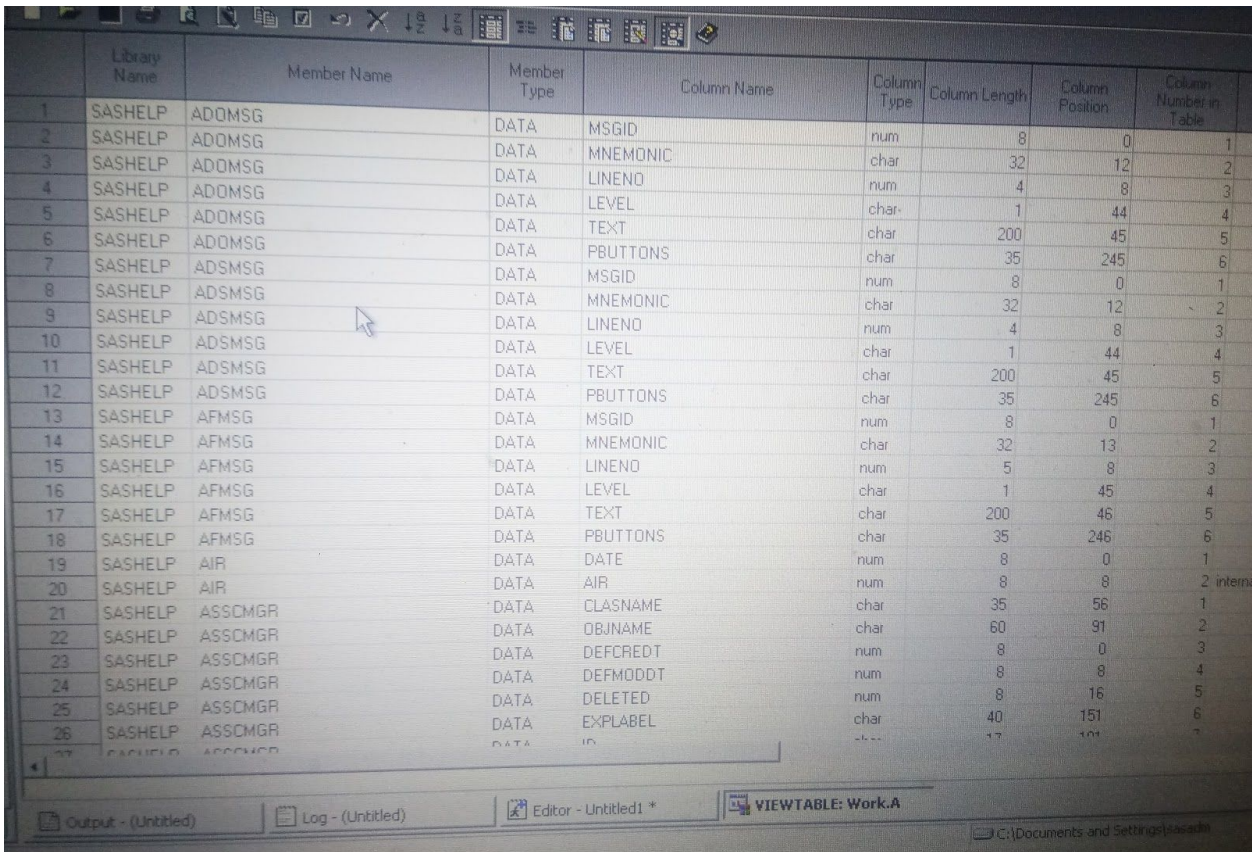
 tab will remain but data will be lost.

admit $

METADATA

Dictionary.columns: It is a repository that stores column level information of every datastep present at the run time.

Proc sql;
Create table a as select * from dictionary.tables;
Quit;



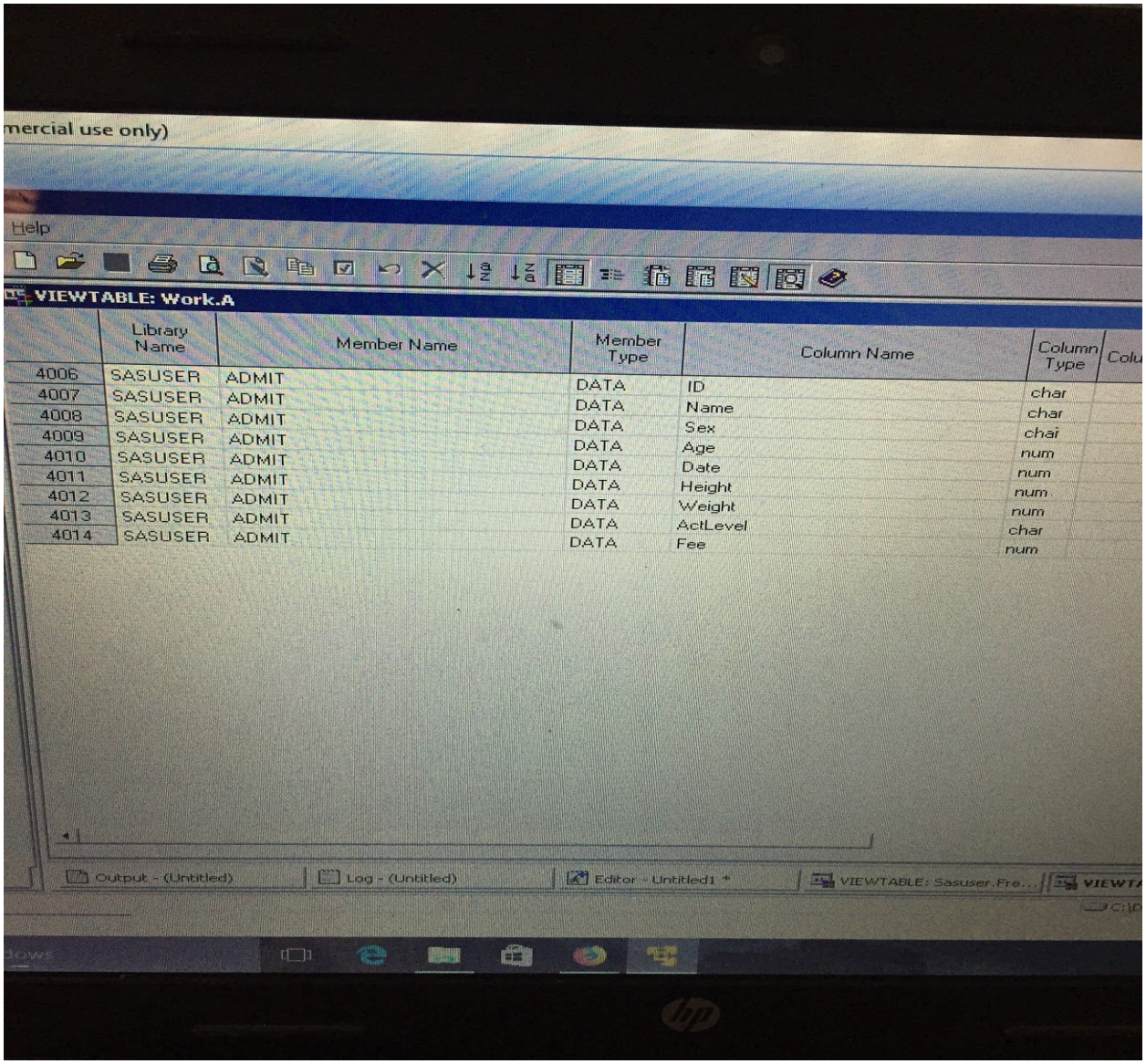
	Library Name	Member Name	Member Type	Column Name	Column Type	Column Length	Column Position	Column Number in Table
1	SASHELP	ADOMSG	DATA	MSGID	num	8	0	1
2	SASHELP	ADOMSG	DATA	MNEMONIC	char	32	12	2
3	SASHELP	ADOMSG	DATA	LINENO	num	4	8	3
4	SASHELP	ADOMSG	DATA	LEVEL	char	1	44	4
5	SASHELP	ADOMSG	DATA	TEXT	char	200	45	5
6	SASHELP	ADOMSG	DATA	PBUTTONS	char	35	245	6
7	SASHELP	ADSMSG	DATA	MSGID	num	8	0	1
8	SASHELP	ADSMSG	DATA	MNEMONIC	char	32	12	2
9	SASHELP	ADSMSG	DATA	LINENO	num	4	8	3
10	SASHELP	ADSMSG	DATA	LEVEL	char	1	44	4
11	SASHELP	ADSMSG	DATA	TEXT	char	200	45	5
12	SASHELP	ADSMSG	DATA	PBUTTONS	char	35	245	6
13	SASHELP	AFMSG	DATA	MSGID	num	8	0	1
14	SASHELP	AFMSG	DATA	MNEMONIC	char	32	13	2
15	SASHELP	AFMSG	DATA	LINENO	num	5	8	3
16	SASHELP	AFMSG	DATA	LEVEL	char	1	45	4
17	SASHELP	AFMSG	DATA	TEXT	char	200	46	5
18	SASHELP	AFMSG	DATA	PBUTTONS	char	35	246	6
19	SASHELP	AIR	DATA	DATE	num	8	0	1
20	SASHELP	AIR	DATA	AIR	num	8	8	2
21	SASHELP	ASSCMGR	DATA	CLASNAME	char	35	56	1
22	SASHELP	ASSCMGR	DATA	OBJNAME	char	60	91	2
23	SASHELP	ASSCMGR	DATA	DEFCREDT	num	8	0	3
24	SASHELP	ASSCMGR	DATA	DEFMODDT	num	8	8	4
25	SASHELP	ASSCMGR	DATA	DELETED	num	8	16	5
26	SASHELP	ASSCMGR	DATA	EXPLABEL	char	40	151	6

Output - (Untitled) | Log - (Untitled) | Editor - Untitled1 * | VIEWTABLE: Work.A

Note : Table work.A is created with 4674 rows and 18 columns.
All the metadata will be copied.

From this table if we want to look for the sasuser.admit data.

Go to Data -- Where -- libname-- EQ-- <look for ditinct values>--SASUSER-- AND-- memname-- EQ-- <look for distinct values>--ADMIT



Code:

Proc sql;

Select memname,name,varnum from dictionary.columns where lowcase(libname)=”sasuser” and lowcase(memname)=”admit” and varnum in (3 5);

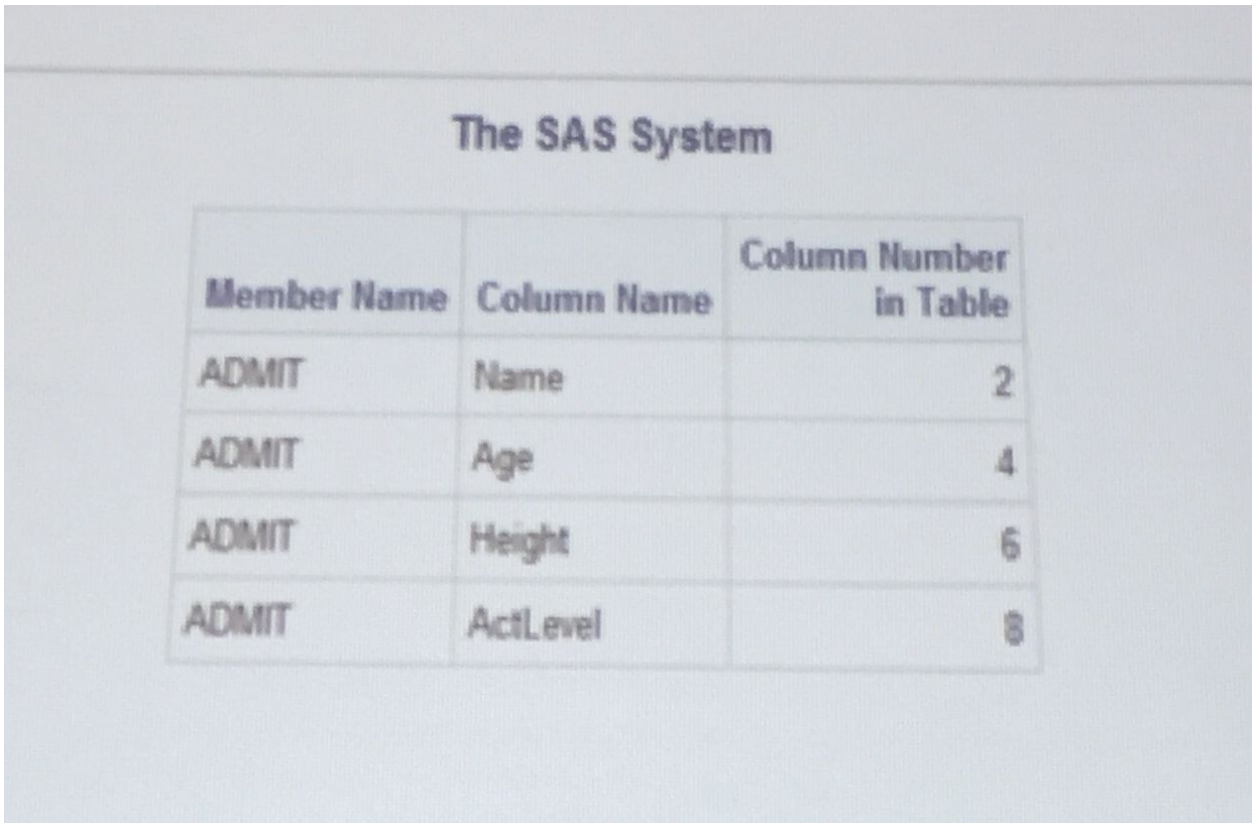
Quit;

Member Name	Column name	Column Number in Table
ADMIT	Sex	3
ADMIT	Date	5

(3 5) : Variables at the position 3rd and 5th in the table will be printed.

Proc sql;

Select memname,name,varnum from dictionary.columns where lowcase(libname)=”sasuser” and lowcase(memname)=”admit” and mod (varnum,2)=0;
Quit;



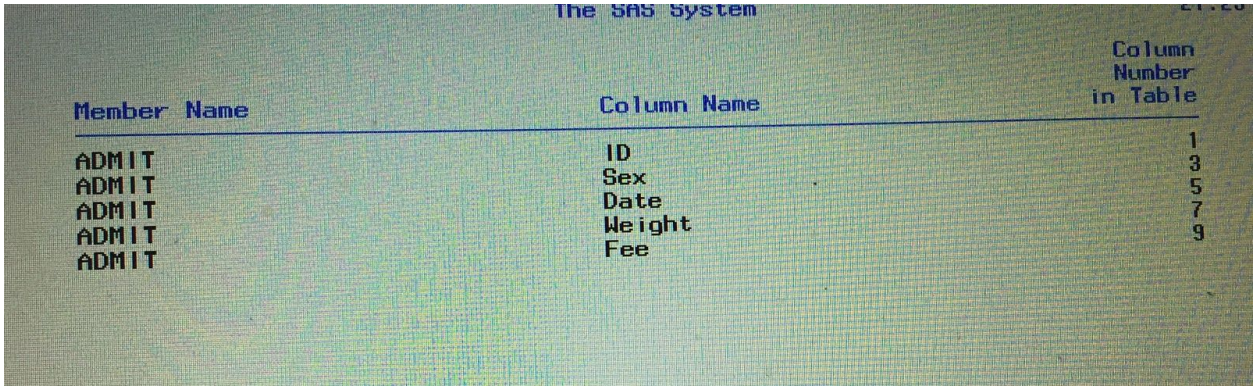
The SAS System

Member Name	Column Name	Column Number in Table
ADMIT	Name	2
ADMIT	Age	4
ADMIT	Height	6
ADMIT	ActLevel	8

mod (varnum,2)=0 : Varibles at even position will be printed.

Proc sql;

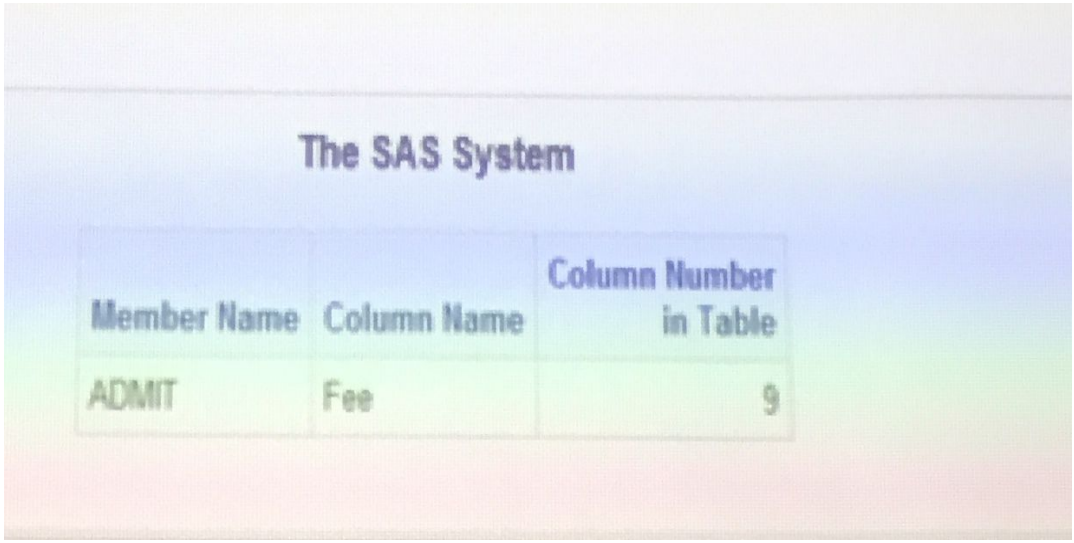
Select memname,name,varnum from dictionary.columns where lowcase(libname)=”sasuser” and lowcase(memname)=”admit” and mod (varnum,2)=1;
Quit;



mod (varnum,2)=1 : Varibles at odd position will be printed.

Proc sql;

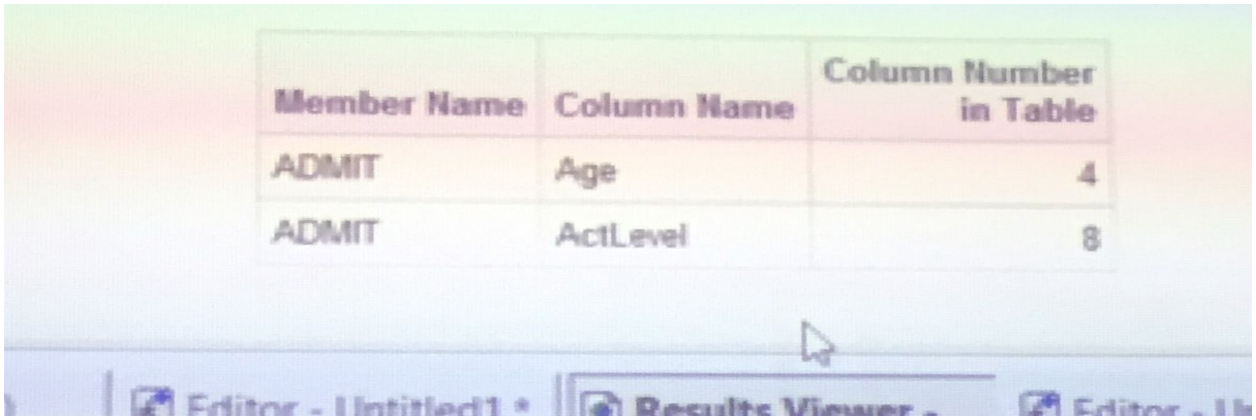
Select memname,name,varnum from dictionary.columns where lowcase(libname)=”sasuser” and lowcase(memname)=”admit” having varnum=max(varnum);
Quit;



varnum=max(varnum) : varnum with maximum value will be printed from the table.

Proc sql;

Select memname,name,varnum from dictionary.columns where lowcase(libname)=”sasuser” and lowcase(memname)=”admit” and name like “A%”;
Quit;

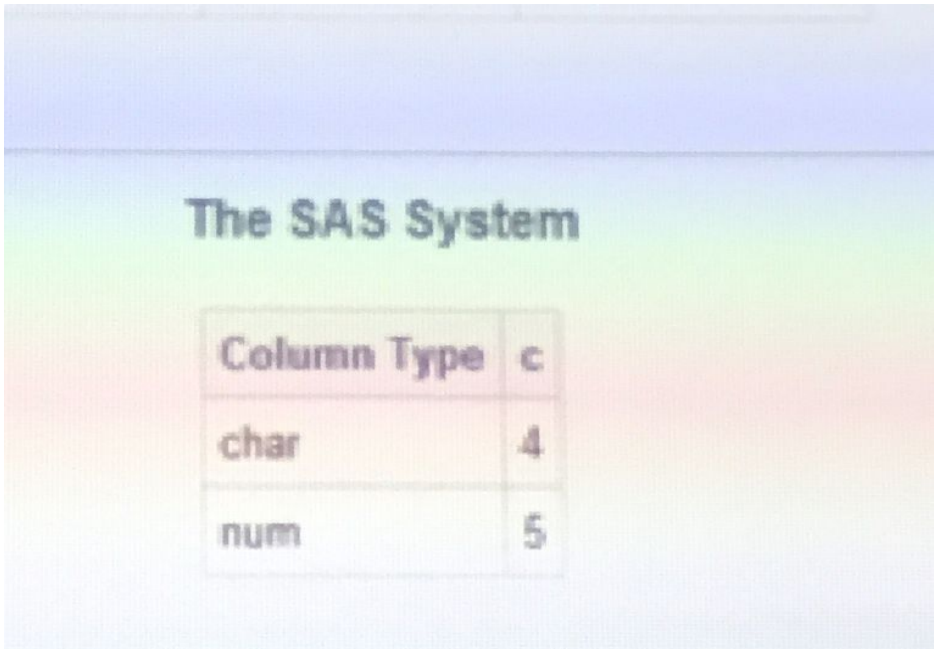


Member Name	Column Name	Column Number in Table
ADMIT	Age	4
ADMIT	ActLevel	8

“A%” : Variables name starting with A will be printed.

Proc sql;

Select type,count(*) as c from dictionary.columns where lowcase(libname)=”sasuser” and lowcase(memname)=”admit” group by type;
Quit;



The SAS System

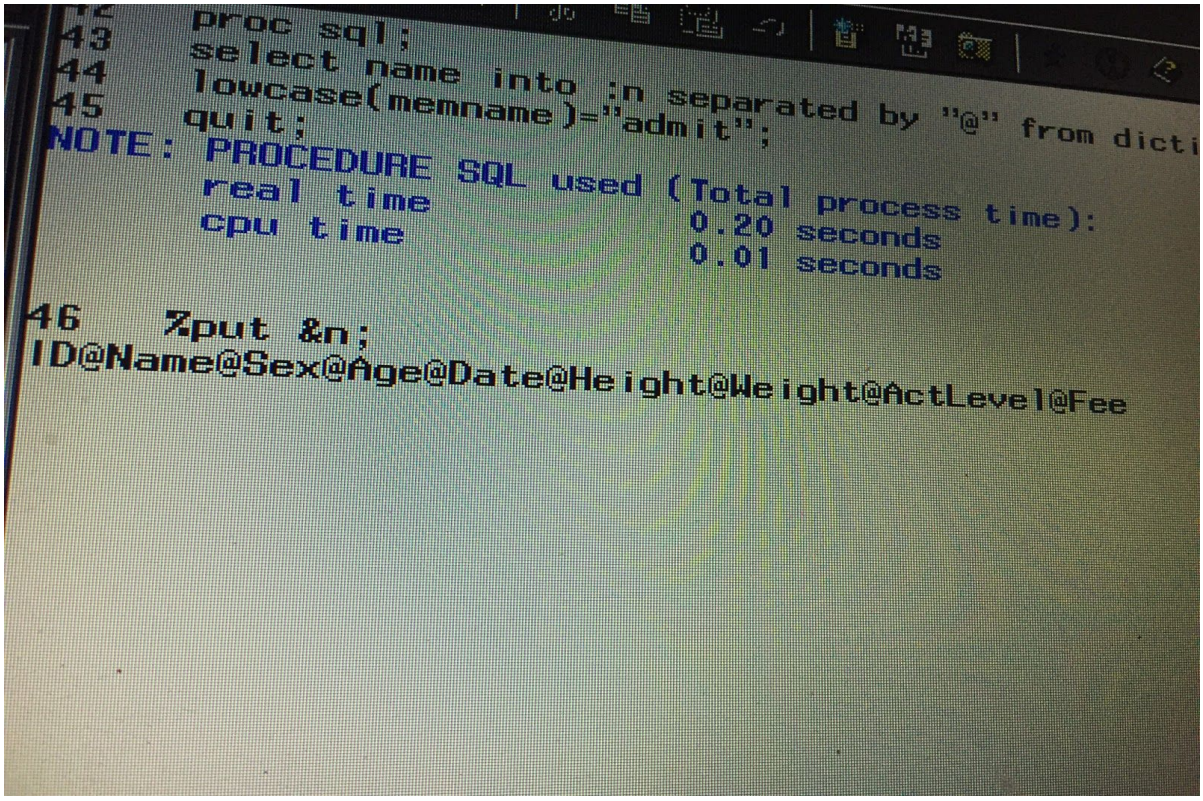
Column Type	c
char	4
num	5

This will give the count of character and numeric type.

Proc sql;

Select name into :n separated by “@” from dictionary.columns where lowcase(libname)=”sasuser” and lowcase(memname)=”admit” ;

Quit;
%put &n;



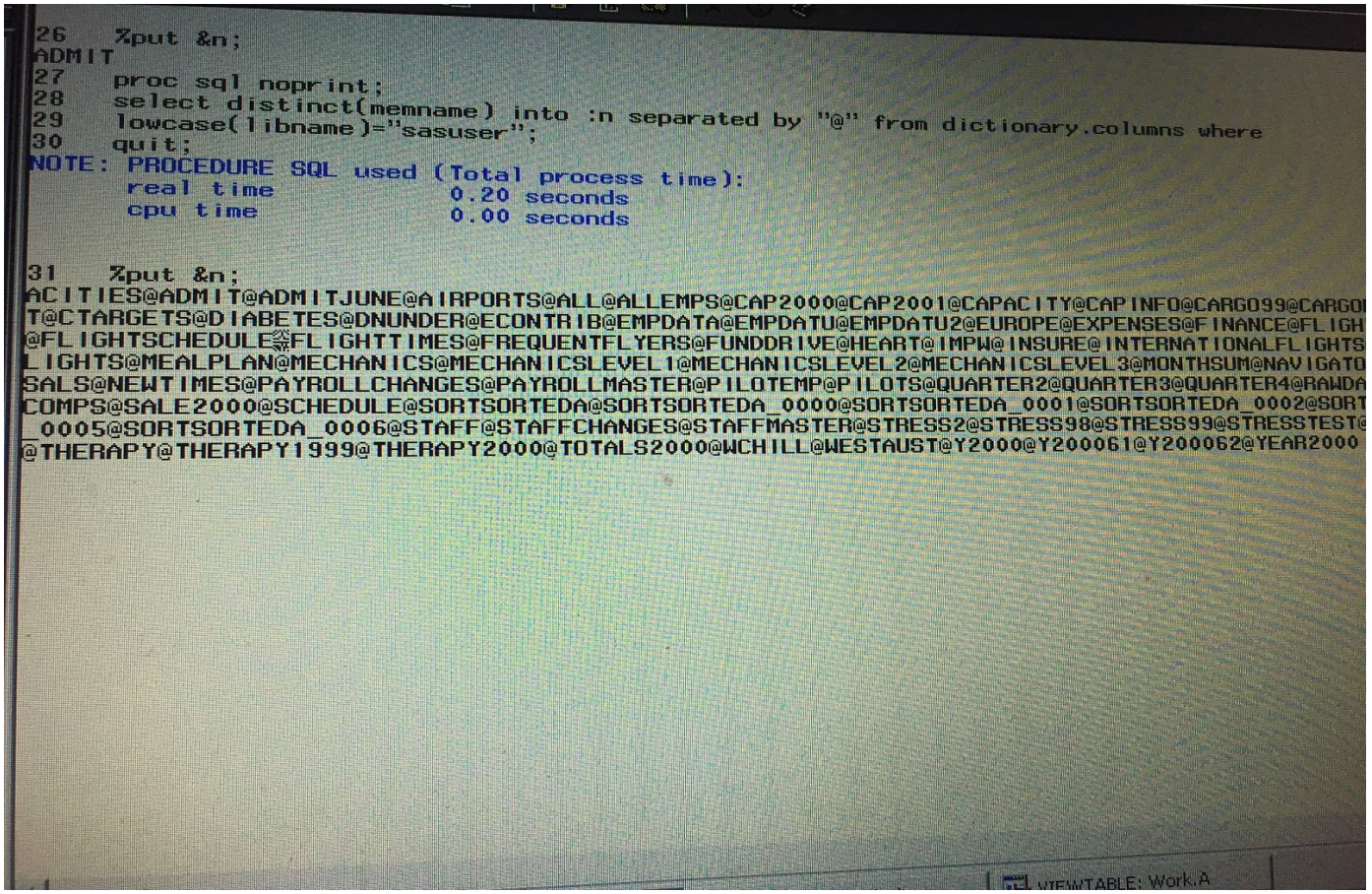
name into :n separated by "@" : "n" name macro variable will be created containing all the variables name seperated by @ where library name is sasuser and member name is admit.

Proc sql noprint;

Select distinct (memname) into :n separated by "@" from dictionary.columns where lowcase(libname)="sasuser";
Quit;

%put &n;

Note : All the files from sasuser whether it is a data set or view , will put in "n" separated by @.

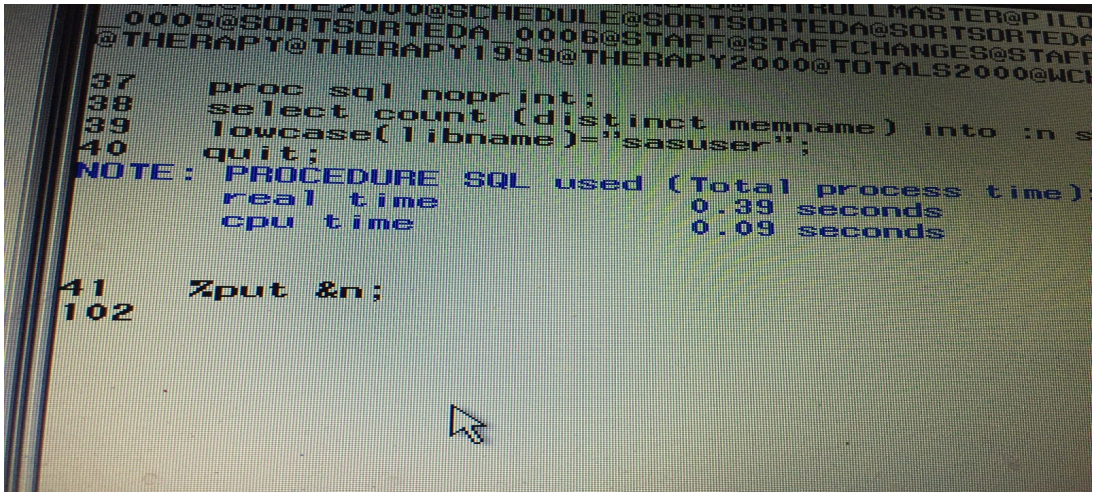


Proc sql noprint;

Select count (distinct memname) into :n separated by “@” from dictionary.columns where lowcase(libname)=”sasuser”;
Quit;

%put &n;

Note: This will give the count of distinct memname i.e 102 (from the output)



Proc sort data = sasuser .admit;
By age;
Run;

Options nolabel;

Proc sql;

Select memname,name,varnum sorted by from dictionary.columns where lowcase(libname)=”sasuser” and lowcase(memname)=”admit”;
Quit;

memname	name	varnum	sortedby
ADMIT	ID	1	0
ADMIT	Name	2	0
ADMIT	Sex	3	0
ADMIT	Age	4	1
ADMIT	Date	5	0
ADMIT	Height	6	0
ADMIT	Weight	7	0
ADMIT	ActLevel	8	0
ADMIT	Fee	9	0

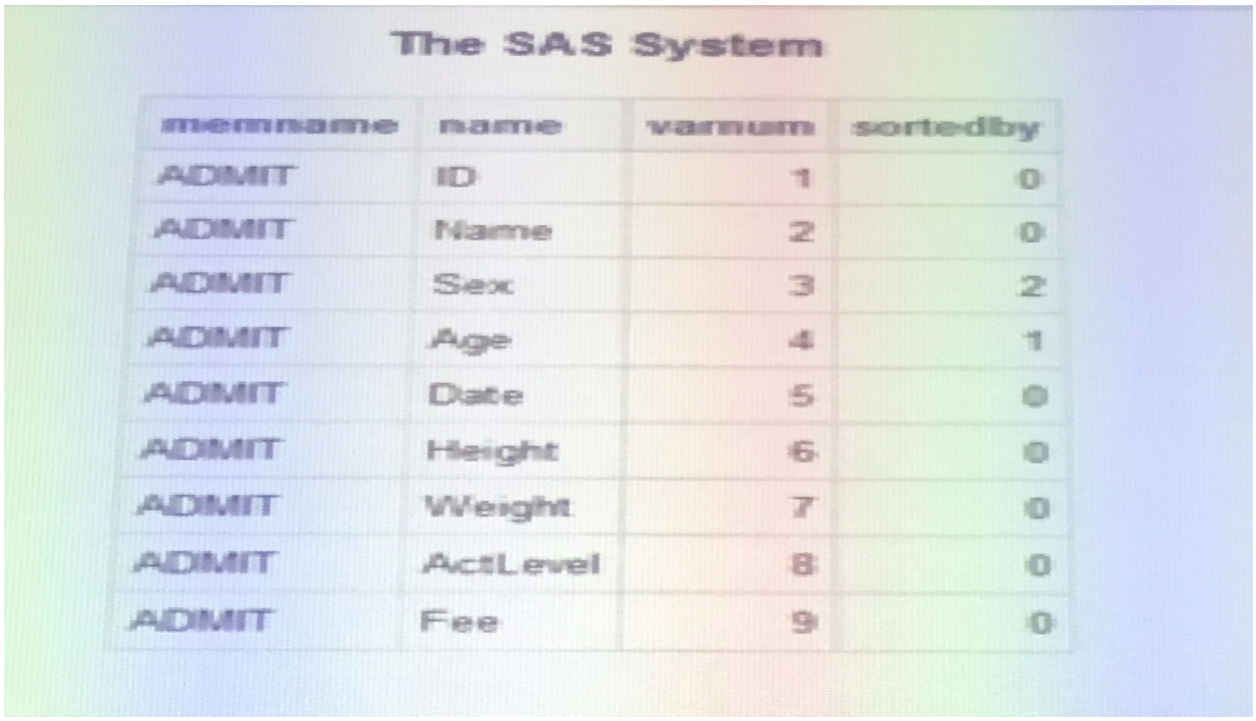
Admit is sorted by age , so in front of age in sorted by column the value is 1.

```
Proc sort data = sasuser .admit;
By age sex;
Run;

Options nolabel;

Proc sql;

Select memname,name,varnum sorted by  from dictionary.columns where
lowcase(libname)=”sasuser” and lowcase(memname)=”admit”;
Quit;
```



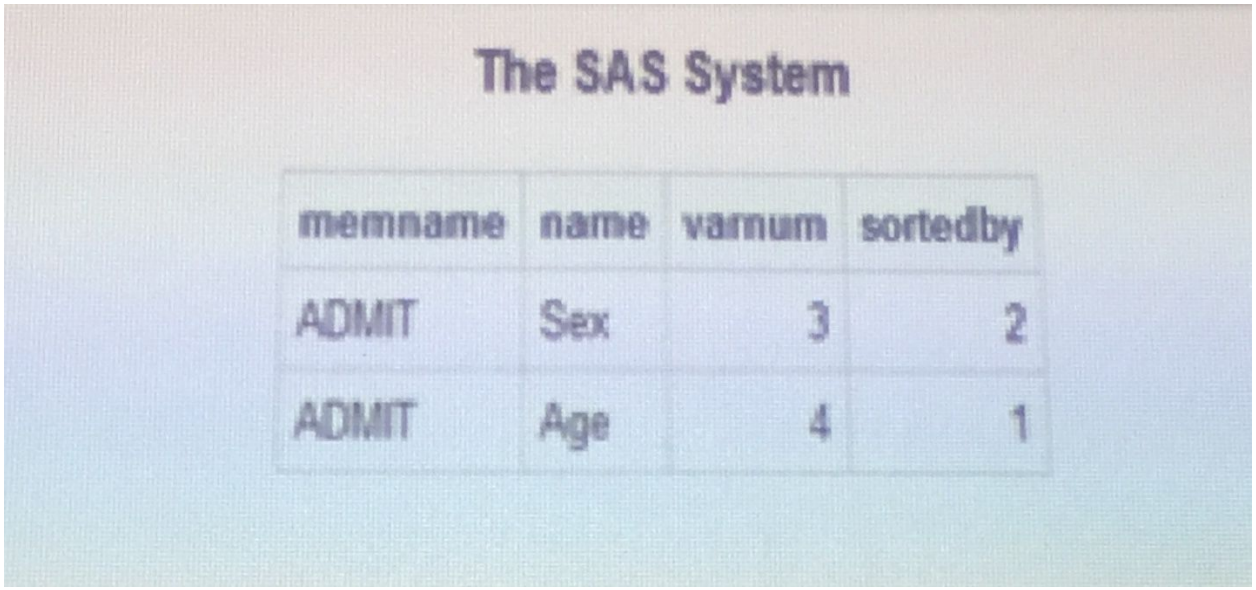
The SAS System

memname	name	varnum	sortedby
ADMIT	ID	1	0
ADMIT	Name	2	0
ADMIT	Sex	3	2
ADMIT	Age	4	1
ADMIT	Date	5	0
ADMIT	Height	6	0
ADMIT	Weight	7	0
ADMIT	ActLevel	8	0
ADMIT	Fee	9	0

Admit is sorted by age and sex.

```
Proc sql;

Select memname,name,varnum sorted by  from dictionary.columns where
lowcase(libname)=”sasuser” and lowcase(memname)=”admit” and sorted by gt 0;;
Quit;
```



The SAS System

memname	name	varnum	sortedby
ADMIT	Sex	3	2
ADMIT	Age	4	1

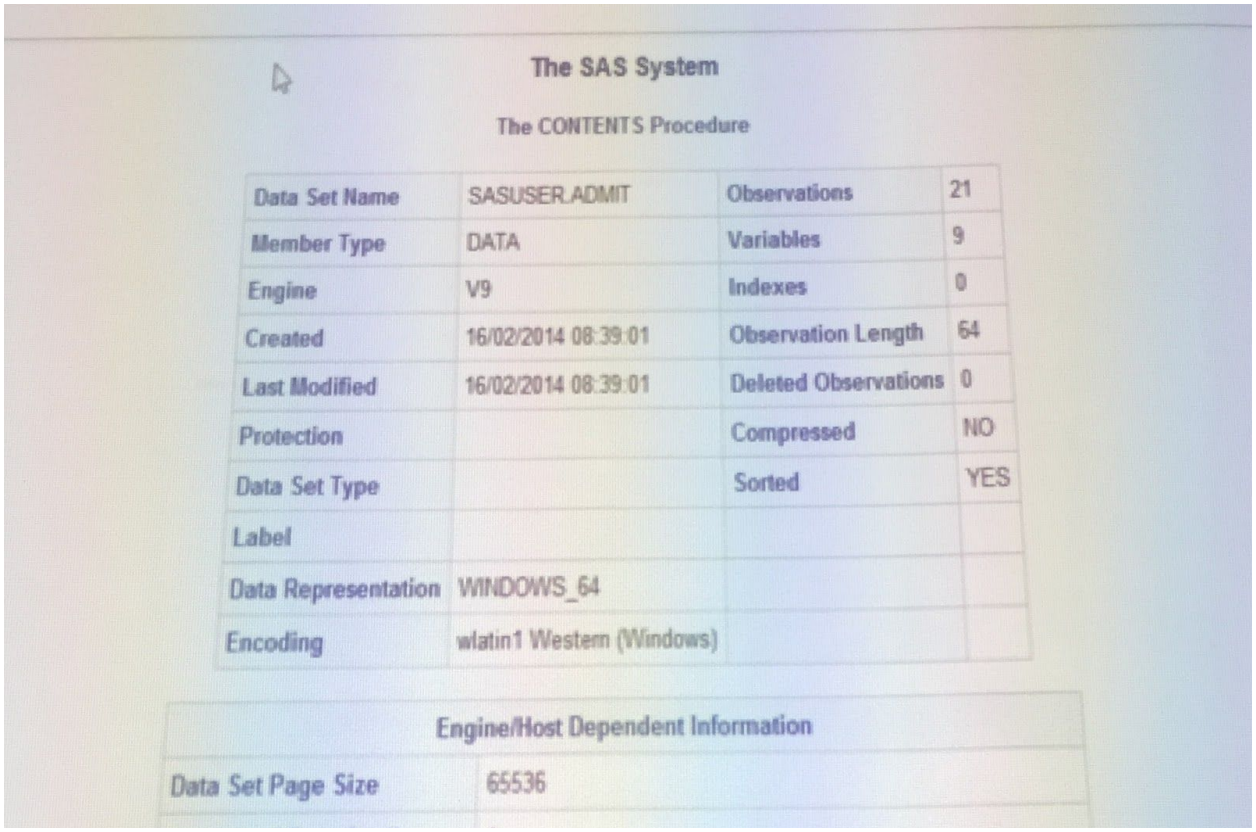
This will give only those variables that are sorted in the dataset i.e. age and sex.

Dictionary Tables: Stores metadata of tables.

```
Proc sql;
Create table a as select * from dictionary.tables;
Quit;
```

Note : Table work.A is created with 549 rows and 39 columns.

```
Proc contents data=sasuser.admit;
Run;
```



The SAS System

The CONTENTS Procedure

Data Set Name	SASUSER.ADMIT	Observations	21
Member Type	DATA	Variables	9
Engine	V9	Indexes	0
Created	16/02/2014 08:39:01	Observation Length	64
Last Modified	16/02/2014 08:39:01	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	65536

Note : proc contents are used to see SAS metadata. Here proc content gives the data from dictionary.columns and dictionary.tables of sasuser.admit.