

## **CLASS 24**

% let a = amit ;

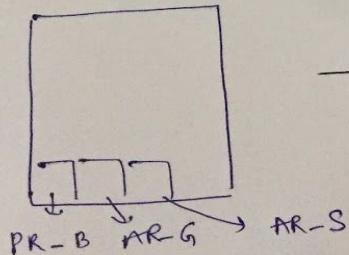
% let b = kumar ;

% let c = %sysfunc(cat(&a, &b));

% put \*\*&c\*\* ;

Explanation Amit → Here if there is space after  
                          xx Amit xx  
                          ↓  
                          space is not visible  
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  
~~to macros~~ directly. So, in that case sysfunc  
is written before base SAS function.



→ Home Work  
Question

% macro ffawsm;

proc sql noint;

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

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;

select distinct membertype into :mt separated  
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  
set sasuser.frequentflyers; and so on with  
where state = "&st" and membertype = "&m";  
run;

Proc EXPORT DATA = &st. - &m OUTFILE = " —

— path — \ff.xls" → This data set  
DBMS = EXCEL REPLACE;  
SHEET = "&st. - &m";} will be exported  
using this command.

RUN;

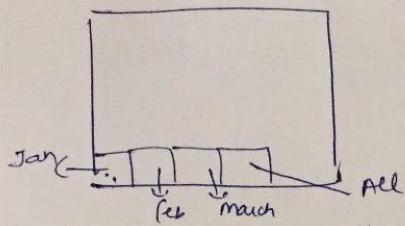
% end

% end

% mend ffawsm;

---

% ffawsm;



## EXCEL ENGINE { Import/Export }

libname yo "c: \_\_\_\_\_ sale.xls";

data yo.admit; → Import

set sasuser.admit; → Export

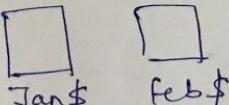
run; → ②

This will work  
as per import.

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

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

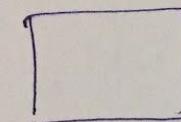
Problem: we don't know the sheet name and the  
number of sheets.

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

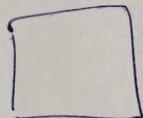
By this we will come to know the names and  
number of sheet.

\$ → This sign indicates that the sheets are  
excel imported, not the actual datasets.

Running ② program will give two entries 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 \$

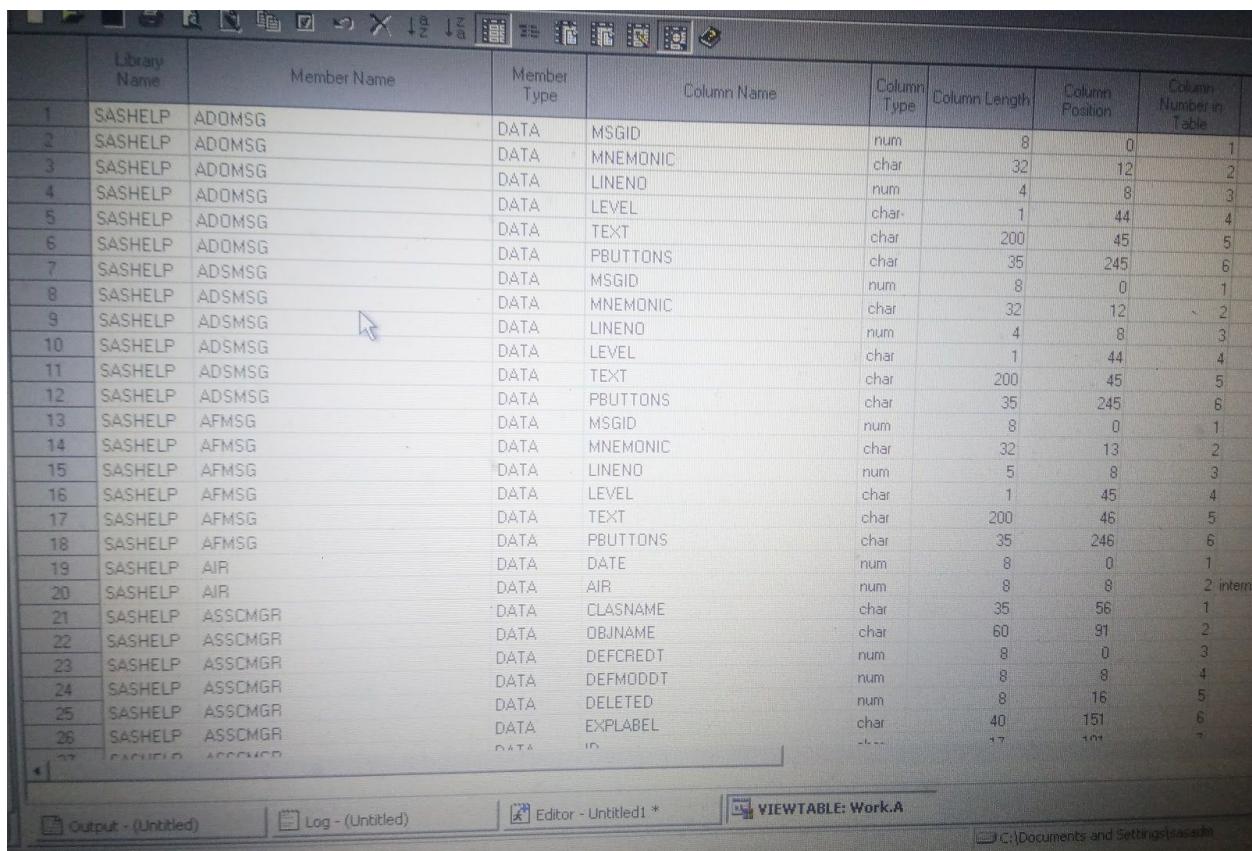
# METADATA

**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	ADSMMSG	DATA	MSGID	num	8	0	1
8	SASHELP	ADSMMSG	DATA	MNEMONIC	char	32	12	2
9	SASHELP	ADSMMSG	DATA	LINENO	num	4	8	3
10	SASHELP	ADSMMSG	DATA	LEVEL	char	1	44	4
11	SASHELP	ADSMMSG	DATA	TEXT	char	200	45	5
12	SASHELP	ADSMMSG	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 intern
21	SASHELP	ASSCMGR	DATA	CLASNAME	char	35	56	1
22	SASHELP	ASSCMGR	DATA	OBJNAME	char	60	91	2
23	SASHELP	ASSCMGR	DATA	DEFREDIT	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
27	SASHELP	ASSCMGR	DATA	ID	char	17	101	7

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 distinct values>--SASUSER-- AND-- memname-- EQ-- <look for distinct values>--ADMIT

mercial use only)

Help

VIEWTABLE: Work.A

	Library Name	Member Name	Member Type	Column Name	Column Type	Colu
4006	SASUSER	ADMIT	DATA	ID	char	
4007	SASUSER	ADMIT	DATA	Name	char	
4008	SASUSER	ADMIT	DATA	Sex	char	
4009	SASUSER	ADMIT	DATA	Age	num	
4010	SASUSER	ADMIT	DATA	Date	num	
4011	SASUSER	ADMIT	DATA	Height	num	
4012	SASUSER	ADMIT	DATA	Weight	num	
4013	SASUSER	ADMIT	DATA	ActLevel	char	
4014	SASUSER	ADMIT	DATA	Fee	num	

Output - (Untitled) Log - (Untitled) Editor - Untitled1 \* VIEWTABLE: Sasuser.Pre... VIEWTA C:\D

hp

---

Code:

Proc sql;

Select memname, name, varnum from dictionary.columns where lowercase(libname)="sasuser" and lowercase(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 lowercase(libname)="sasuser" and lowercase(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** : Variables at even position will be printed.

---

Proc sql;

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

Member Name	Column Name	Column Number in Table
ADMIT	ID	1
ADMIT	Sex	3
ADMIT	Date	5
ADMIT	Weight	7
ADMIT	Fee	9

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

---

Proc sql;

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

Member Name	Column Name	Column Number in Table
ADMIT	Fee	9

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

---

Proc sql;

Select memname,name,varnum from dictionary.columns where lowercase(libname)="sasuser" and lowercase(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 lowercase(libname)='sasuser' and lowercase(memname)='admit' group by type;  
Quit;

The SAS System	
Column Type	c
char	4
num	5

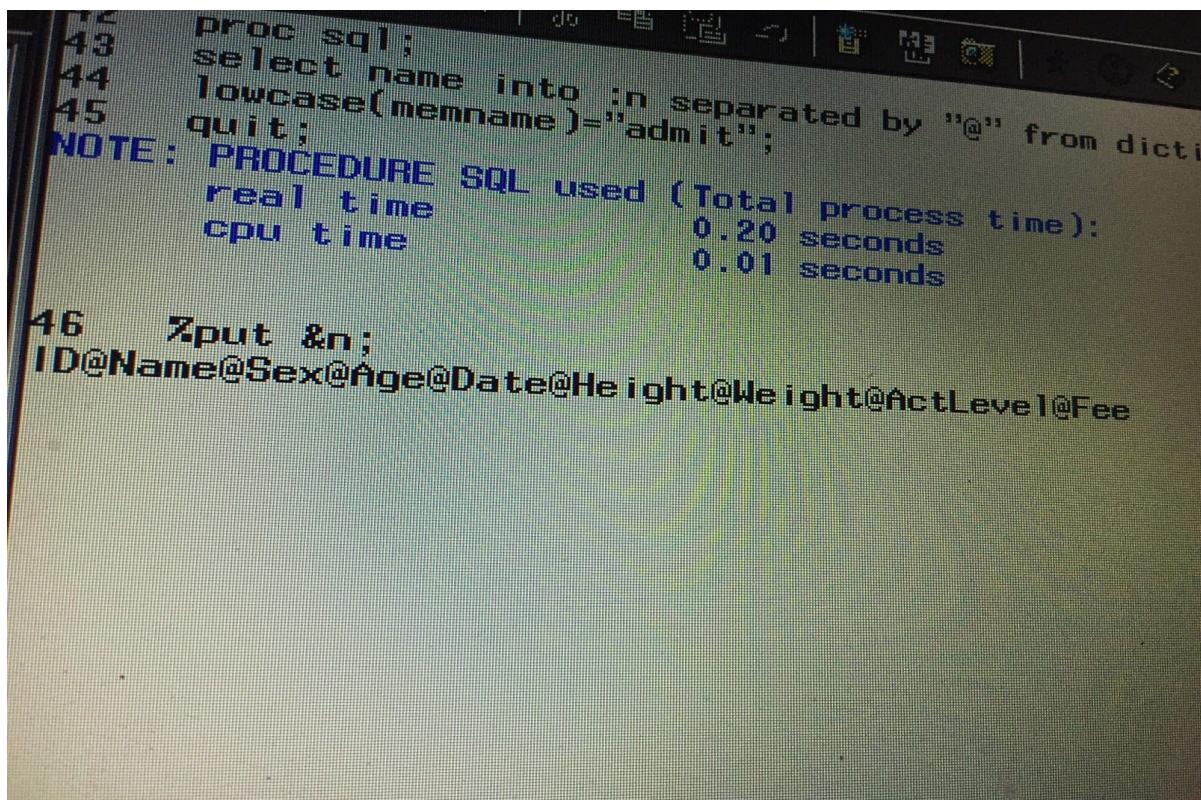
This will give the count of character and numeric type.

---

Proc sql;

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

```
Quit;  
%put &n;
```



42  
43 Proc sql;  
44 select name into :n separated by "@" from dictionary.columns  
45 where  
46 lowcase(memname)="admit";  
quit;  
NOTE: PROCEDURE SQL used (Total process time):  
real time 0.20 seconds  
cpu time 0.01 seconds

```
46 %put &n;  
ID@Name@Sex@Age@Date@Height@Weight@ActLevel@Fee
```

**name into :n separated by “@” :** “n” name macro variable will be created containing all the variables name separated 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 @.

```

26  %put &n;
ADMIT
27  proc sql noprint;
28  select distinct(memname) into :n separated by "@" from dictionary.columns where
29  lowercase(libname)="sasuser";
30  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time            0.20 seconds
      cpu time             0.00 seconds

31  %put &n;
ACITIES@ADMIT@ADMITJUNE@AIRPORTS@ALL@ALLEMP@CAP2000@CAP2001@CAPACITY@CAPINFO@CARGO99@CARGOT
@CTARGETS@DIAOTES@DNUNDER@ECONTRIB@EMPDATA@EMPDATU@EMPDATU2@EUROPE@EXPENSES@FINANCE@FLIGH
@FLIGHTSCHEDULE@FLIGHTTIMES@FREQUENTFLYERS@FUNDRIVE@HEART@IMPW@INSURE@INTERNATIONALFLIGHTS
LIGHTS@MEALPLAN@MECHANICS@MECHANICSLEVEL1@MECHANICSLEVEL2@MECHANICSLEVEL3@MONTHSUM@NAVIGATO
SALS@NEWTIMES@PAYROLLCHANGES@PAYROLLMASTER@PILOTTEMP@PILOTS@QUARTER2@QUARTER3@QUARTER4@RAWDA
COMPS@SALE2000@SCHEDULE@SORTSORTEDA@SORTSORTEDA_0000@SORTSORTEDA_0001@SORTSORTEDA_0002@SORT
_0005@SORTSORTEDA_0006@STAFF@STAFFCHANGES@STAFFMASTER@STRESS2@STRESS98@STRESS99@STRESSTEST
@THERAPY@THERAPY1999@THERAPY2000@TOTALS2000@WCHILL@WESTAUST@Y2000@Y200061@Y200062@YEAR200

```

Proc sql noprint;

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

```
%put &n;
```

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

```

0005@SORTSORTED 0006@SORTSORTED 0007@SORTSORTED 0008@SORTSORTED 0009@SORTSORTED 0010@MASTER@P10
@THERAPY@THERAPY1 999@THERAPY2 000@TOTALS 2000@WCI
37  proc sql noprint;
38  select count (distinct memname) into :n
39  lowercase(libname)="sasuser";
40  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.39 seconds
      cpu time            0.09 seconds

41  %put &n;
102

```

---

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

Options nolabel;

Proc sql;

Select memname,name,varnum sorted by from dictionary.columns where  
 lowercase(libname)="sasuser" and lowercase(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.