Statistical Analysis System : Class 16
Dated: 22-Apr-2018

::

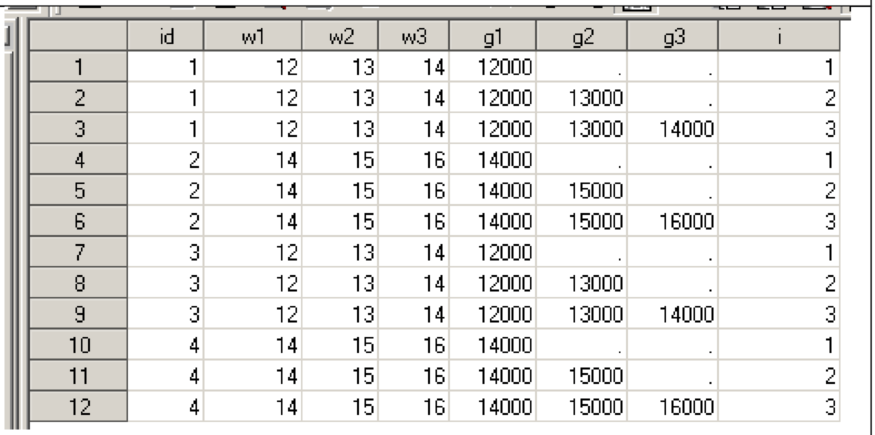
Array 1-D revised :

.

```
/* prog 1 */
```

```
data a;
input id w1 w2 w3;
array k(3) w1-w3;
array g(3) g1-g3;
do i=1 to 3;
g(i)=k(i)*1000;
output;
end;
cards;
1 12 13 14
2 14 15 16
3 12 13 14
4 14 15 16
;
run
;
```

Output



	id	w1	w2	w3	g1	g2	g3	i
1	1	12	13	14	12000	.	.	1
2	1	12	13	14	12000	13000	.	2
3	1	12	13	14	12000	13000	14000	3
4	2	14	15	16	14000	.	.	1
5	2	14	15	16	14000	15000	.	2
6	2	14	15	16	14000	15000	16000	3
7	3	12	13	14	12000	.	.	1
8	3	12	13	14	12000	13000	.	2
9	3	12	13	14	12000	13000	14000	3
10	4	14	15	16	14000	.	.	1
11	4	14	15	16	14000	15000	.	2
12	4	14	15	16	14000	15000	16000	3

Explained:

Here, array (g) is created by multiplying each array elements of array (k) with 1000.

Note: array (g) elements are stored column-wise as the value of "i" in the loop increases.

Dynamic array definition: **Syntax:** array array_name(*) _numeric_;

array : is the keyword

array_name(*) : asterisk in parenthesis implies dynamic dimension / size of the number of elements of array.

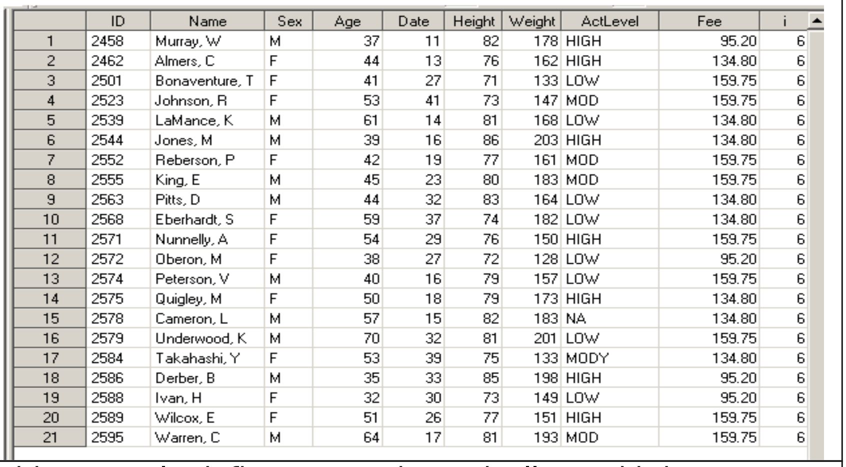
numeric : type of array elements on which the functions are performed

dim (array_name): is dimension function. It returns the count value of the total elements of array as the ending range for the loop.

/* prog 2 */

Output

```
data a;
set sasuser.admit;
array k(*) _numeric_;
do i=1 to dim (k);
k(i)=k(i)+10;
end;
run;
```



	ID	Name	Sex	Age	Date	Height	Weight	ActLevel	Fee	i
1	2458	Murray, W	M	37	11	82	178	HIGH	95.20	6
2	2462	Almers, C	F	44	13	76	162	HIGH	134.80	6
3	2501	Bonaventure, T	F	41	27	71	133	LOW	159.75	6
4	2523	Johnson, R	F	53	41	73	147	MOD	159.75	6
5	2539	LaMance, K	M	61	14	81	168	LOW	134.80	6
6	2544	Jones, M	M	39	16	86	203	HIGH	134.80	6
7	2552	Reberson, P	F	42	19	77	161	MOD	159.75	6
8	2555	King, E	M	45	23	80	183	MOD	159.75	6
9	2563	Pitts, D	M	44	32	83	164	LOW	134.80	6
10	2568	Eberhardt, S	F	59	37	74	182	LOW	134.80	6
11	2571	Nunnelly, A	F	54	29	76	150	HIGH	159.75	6
12	2572	Oberon, M	F	38	27	72	128	LOW	95.20	6
13	2574	Peterson, V	M	40	16	79	157	LOW	159.75	6
14	2575	Quigley, M	F	50	18	79	173	HIGH	134.80	6
15	2578	Cameron, L	M	57	15	82	183	NA	134.80	6
16	2579	Underwood, K	M	70	32	81	201	LOW	159.75	6
17	2584	Takahashi, Y	F	53	39	75	133	MODY	134.80	6
18	2586	Derber, B	M	35	33	85	198	HIGH	95.20	6
19	2588	Ivan, H	F	32	30	73	149	LOW	95.20	6
20	2589	Wilcox, E	F	51	26	77	151	HIGH	159.75	6
21	2595	Warren, C	M	64	17	81	193	MOD	159.75	6

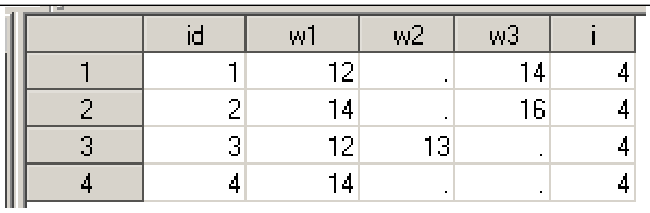
Explained:

this example defines array dynamically, and it increases all numeric values by 10 for all numeric variables.

/* prog 3 */

Output

```
data a;
input id w1 w2 w3;
array k(3) w1-w3;
do i=1 to 3;

end;
cards;
1 12 . 14
2 14 . 16
3 12 13 .
4 14 . .
;
run
;
```



	id	w1	w2	w3	i
1	1	12	.	14	4
2	2	14	.	16	4
3	3	12	13	.	4
4	4	14	.	.	4

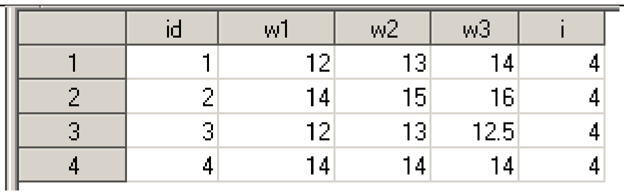
Explained:

Defines an array “k” with variables w1,w2,w3 with values read from the datalines including missing numeric values implied by “.”

/* prog 4 */

```
data a;
input id w1 w2 w3;
array k(3) w1-w3;
do i=1 to 3;

if k(i)=. then
k(i)=mean(of k(*));
end;
cards;
1 12 . 14
2 14 . 16
3 12 13 .
4 14 . .
;
run
;
```

Output



	id	w1	w2	w3	i
1	1	12	13	14	4
2	2	14	15	16	4
3	3	12	13	12.5	4
4	4	14	14	14	4

Explained:

Defines an array “k” with variables w1,w2,w3 with values read from the datalines including missing numeric values implied by “.”
Also, in the do-loop for any missing value encountered at any position, average/mean of the rest of the elements in that row is stored at the missing location.

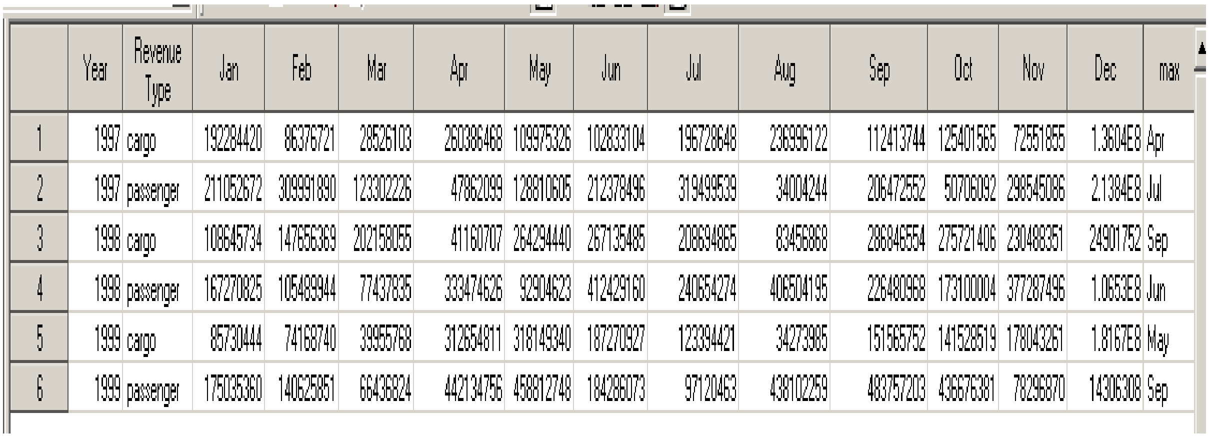
Whichn(a,b): whichn function searches the 2nd and the subsequent arguments for a value equal to the 1st argument and returns the position / index of the 1st matching value from the elements.

Vname: It returns the variable-name of the index / position given out by whichn.

/* prog 5 using whichn function. */

```
data a;
set sasuser.target;
array k(*) jan--dec;
max= vname (k (whichn(max(of k(*)), of k(*))));
run;
```

Output



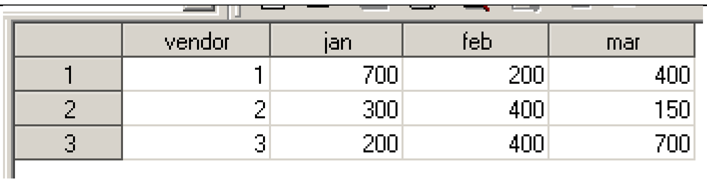
	Year	Revenue Type	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	max
1	1997	cargo	192284420	86376721	28526103	260386468	109975326	102833104	196728648	236996122	112413744	125401565	72551855	1.3604E8	Apr
2	1997	passenger	211052672	309991890	123302226	47862099	128810605	212378496	319499539	34004244	206472552	50706092	298545086	2.1384E8	Jul
3	1998	cargo	108645734	147656369	202158055	41160707	264294440	267135485	208694865	83456868	286846554	275721406	230488351	24901752	Sep
4	1998	passenger	167270825	105489944	77437835	333474626	92904623	412429160	240654274	406504195	226480968	173100004	377287496	1.0653E8	Jun
5	1999	cargo	85730444	74168740	39955768	312654811	318149340	187270927	123394421	34273985	151565752	141528519	178043261	1.8167E8	May
6	1999	passenger	175035360	140625851	66436824	442134756	458812748	184286073	97120463	438102259	483757203	436676381	78296870	14306308	Sep

Explained: The month with maximum sale by revenue type throughout the year is generated in the "max " variable.

/* prog 6 */

Output

```
data a;
input vendor jan feb
mar;
cards;
1 700 200 400
2 300 400 150
3 200 400 700
;
run;
```



	vendor	jan	feb	mar
1	1	700	200	400
2	2	300	400	150
3	3	200	400	700

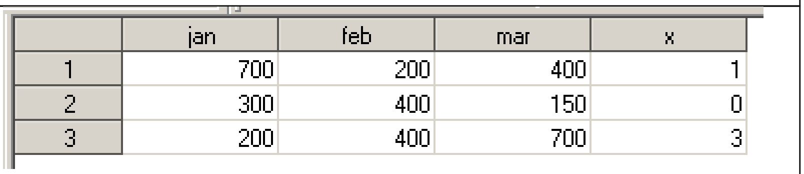
Explained:

Dataset "a" is created with variables vendor, jan, feb, mar and the values read by datalines.

/* prog 7 */

Output

```
data c;
set a;
x= whichn(700,of jan feb
mar);
run;
```



	jan	feb	mar	x
1	700	200	400	1
2	300	400	150	0
3	200	400	700	3

Explained:

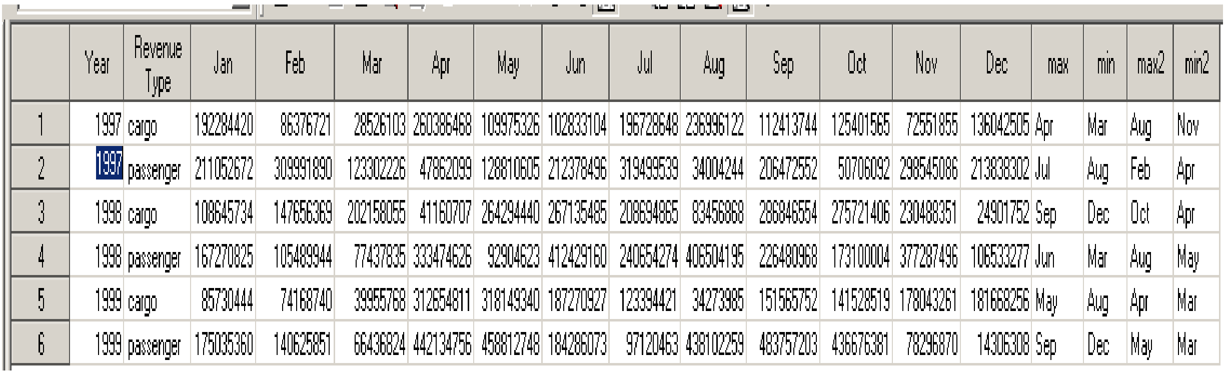
Whichn function looks for 1st argument i.e "700" into 2nd argument with variables JAN, FEB, MAR and returns the index of the matching value form among the variables in the 2nd arguments.

/* prog 8 */

```
data a;
set sasuser.target;
array k(*) jan--dec;
```

```
max=vname(k(whichn(max(of k(*)), of k(*))));
min=vname(k(whichn(min(of k(*)), of k(*))));
max2=vname(k(whichn(largest(2, of k(*)), of k(*))));
min2=vname(k(whichn(smallest(2, of k(*)), of k(*))));
run;
```

Output



	Year	Revenue Type	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	max	min	max2	min2
1	1997	cargo	192284420	86376721	28526103	260386468	109975326	102833104	196728648	236996122	112413744	125401565	72551855	136042505	Apr	Mar	Aug	Nov
2	1997	passenger	211052672	309991890	123302226	47862099	128810605	212378496	319499539	34004244	206472552	50706092	298545086	213838302	Jul	Aug	Feb	Apr
3	1998	cargo	108645734	147656369	202158055	41160707	264294440	267135485	208694865	83456868	286846554	275721406	230488351	24901752	Sep	Dec	Oct	Apr
4	1998	passenger	167270825	105489944	77437835	333474626	92904623	412429160	240654274	406504195	226480968	173100004	377287496	106533277	Jun	Mar	Aug	May
5	1999	cargo	85730444	74168740	39955768	312654811	318149340	187270927	123394421	34273985	151565752	141528519	178043261	181668256	May	Aug	Apr	Mar
6	1999	passenger	175035360	140625851	66436824	442134756	458812748	184286073	97120463	438102259	483757203	436676381	78296870	14306308	Sep	Dec	May	Mar

Explained:
using vname and whichn functions max, min, max2, min2 variables are calculated.

/* prog 9 */

```
proc freq
data= a;
tables max;
run;
```

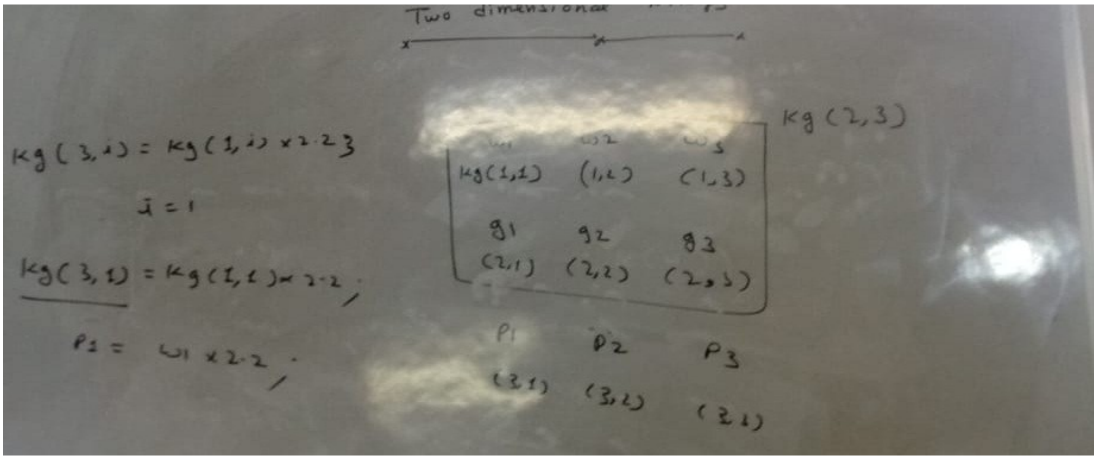
Output

```
                    The SAS System             02:26 Friday, M

                   The FREQ Procedure

                                      Cumulative    Cumulative
max     Frequency     Percent         Frequency      Percent
--------------------------------------------------------------
Apr            1        16.67                1         16.67
Jul            1        16.67                2         33.33
Jun            1        16.67                3         50.00
May            1        16.67                4         66.67
Sep            2        33.33                6        100.00
```

Explained:

Here, Proc freq is applied on the dataset “a” applied only on the variable “max”, it helps in calculating attributes as listed above.

2-D Array:

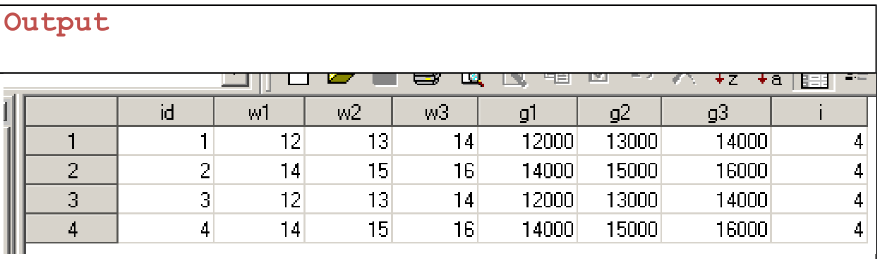


Above picture describes a 2-D array’s formation.

/* prog 10 */

Output

```
/* 2D array */

data a;
input id w1 w2 w3;
array kg(2,3) w1-w3
g1-g3;
do i=1 to 3;
kg(2,i)=kg(1,i)*1000
;
end;
cards;
1 12 13 14
2 14 15 16
3 12 13 14
4 14 15 16
;
run
;
```



	id	w1	w2	w3	g1	g2	g3	i
1	1	12	13	14	12000	13000	14000	4
2	2	14	15	16	14000	15000	16000	4
3	3	12	13	14	12000	13000	14000	4
4	4	14	15	16	14000	15000	16000	4

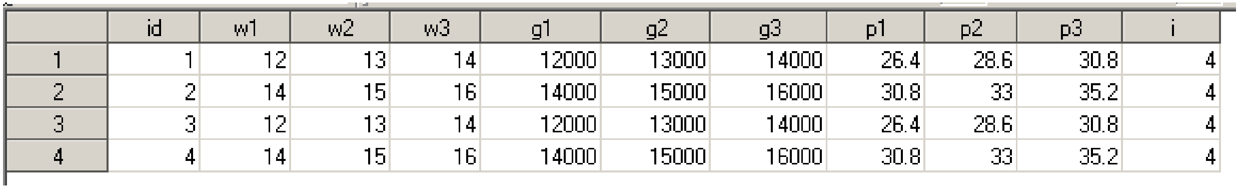
Explained:

Creates “kg” array which implies 2 arrays with 3 variables.
kg(2,i)=kg(1,i)*1000--- creates 2nd array with 3 variables w.r.t i’s iteration in the loop (g1,g2,g3) as per the arithmetic operation applied.

/* prog 11 */

```
data a;
input id w1 w2 w3;
array kg(3,3)
w1-w3 g1-g3 p1-p3;
do i=1 to 3;
kg(2,i)=kg(1,i)*1000;
kg(3,i)=kg(1,i)*2.2;
end;
cards;
1 12 13 14
2 14 15 16
3 12 13 14
4 14 15 16
;
run
;
```

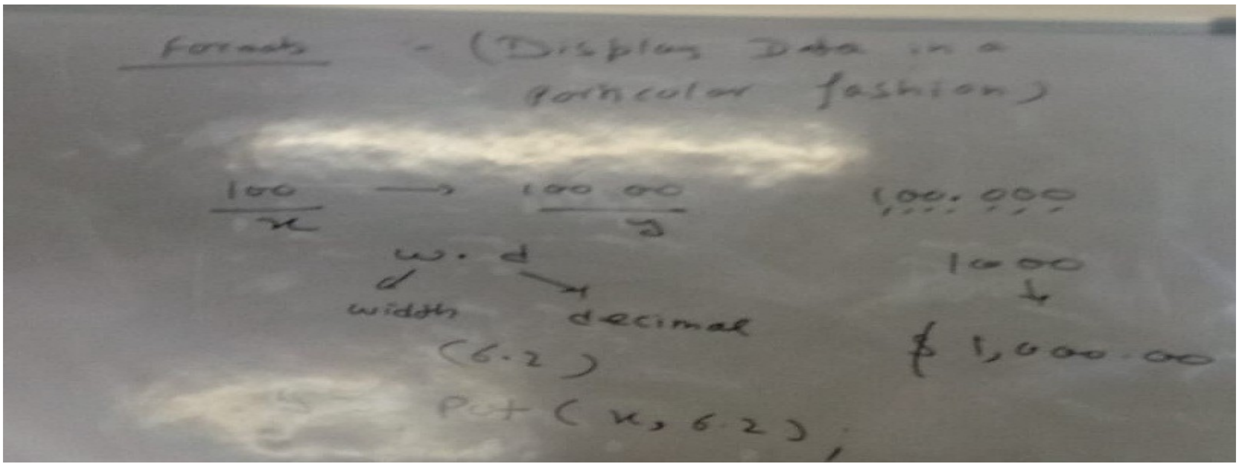
Output



	id	w1	w2	w3	g1	g2	g3	p1	p2	p3	i
1	1	12	13	14	12000	13000	14000	26.4	28.6	30.8	4
2	2	14	15	16	14000	15000	16000	30.8	33	35.2	4
3	3	12	13	14	12000	13000	14000	26.4	28.6	30.8	4
4	4	14	15	16	14000	15000	16000	30.8	33	35.2	4

Explained: kg(2,i)=kg(1,i)*1000;--- creates 2nd array with 3 variables w.r.t i's iteration in the loop (g1,g2,g3) as per the arithmetic operation applied.
Similarly, kg(3,i)=kg(1,i)*2.2;---- creates 3rd array with 3 variables same way w.r.t operation applied.

Formats:



Displays data in a particular fashion.
Syntax: Put (x,y) , x implies variable and y implies decimal value.
It converts numeric type to character type.

Example:
x=100; /* implies numeric*/
y=put (x,6.2); /* applying format*/

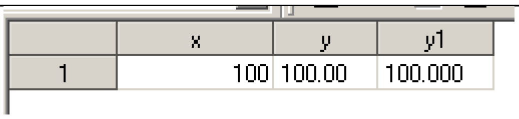
Output:

Y=100.00 /* implies 6.2 format on x =100 with total 6 characters including decimal point and 2 decimal values*/

/* prog 12 */

Output

```
/* formats  */

data a;
x=100;
y=put (x,6.2);
y1=put(x,7.3);
run;
```



	x	y	y1
1	100	100.00	100.000

Explained:

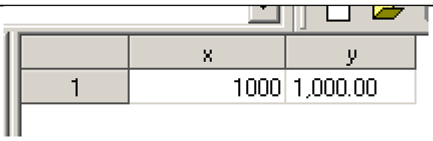
PUT FORMAT

Y = 100.00, in the output has total 6 characters including decimal point (.) and 2 decimal places implying 6.2 format.
Similarly, y1 has 7.3 format applied.

/* prog 13 */

Output

```
data a;
x=1000;
y=put(x,comma8.2);
run;
```



	x	y
1	1000	1,000.00

Explained:

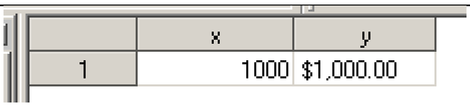
COMMA FORMAT

Y = 1,000.00 implies 8.2 format (total 8 characters & 2 decimal values) with comma separation preceding every 3 digit to the left of decimal point.

/* prog 14 */

Output

```
data a;
x=1000;
y=put (x,dollar9.2);
run;
```



	x	y
1	1000	$1,000.00

Explained:

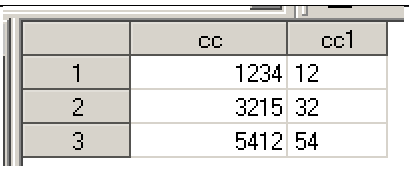
DOLLAR FORMAT

Y = $1,000.00 implies 9.2 format (total 9 characters & 2 decimal values) with comma separation preceding every 3 digit to the left of decimal point and $ applied at the leftmost place in the number.

/* prog 15 */

Output

```
data a;
input cc;
cc1=substr
(put(cc,4.),1,2);
cards;
1234
3215
```



	cc	cc1
1	1234	12
2	3215	32
3	5412	54

```
5412
;
run;
```

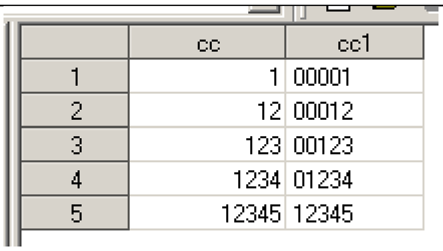
Explained:

This code extracts first 2 numbers from the original number by converting numeric variable “cc” into string by put format stored in variable cc1 (here, cc1 generated is character type).

/* prog 16 */

Output

```
data a;
input cc;
cc1=put(cc,z5.);
cards;
1
12
123
1234
12345
;
run;
```



	cc	cc1
1	1	00001
2	12	00012
3	123	00123
4	1234	01234
5	12345	12345

Explained:

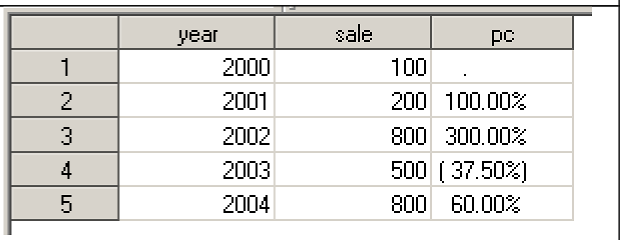
Z FORMAT

This code uses z5. indicating total 5 characters in the output variable, which simply adds 0’s in the leading places of any number if having less than 5 digit (here 5, can be any length).

/* prog 17 */

Output

```
data a;
input year sale;
pc = put((sale-
lag(sale))/lag(sale),percent9.2);
cards;
2000 100
2001 200
2002 800
```

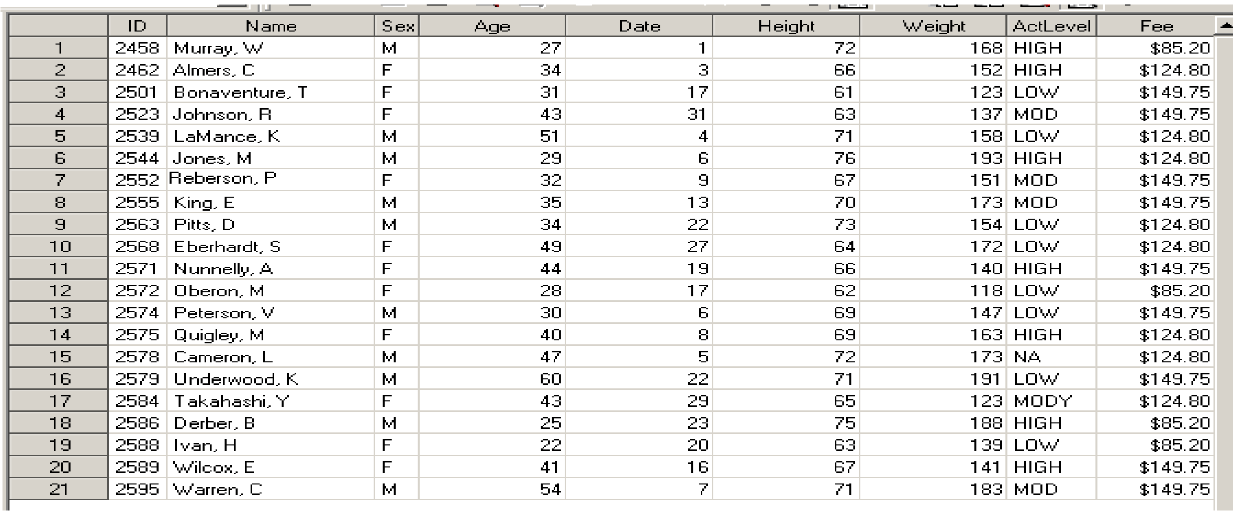


	year	sale	pc
1	2000	100	.
2	2001	200	100.00%
3	2002	800	300.00%
4	2003	500	(37.50%)
5	2004	800	60.00%

2003 500 2004 800 ; run;	
Explained: PERCENT FORMAT	Percent, used with put indicates percent format, it calculates percentage alongwith "%" symbol (% takes 3 bits). For any negative value in the variable where percent format is applied, value appears in parenthesis (like 4th row above).

```
/* prog 18 */
data a;
set sasuser.admit;
format fee dollar9.2;
run;
```

Output



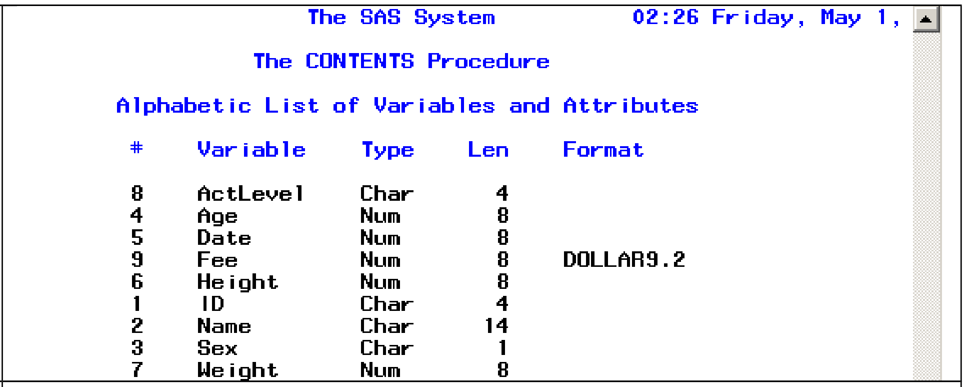
	ID	Name	Sex	Age	Date	Height	Weight	ActLevel	Fee
1	2458	Murray, W	M	27	1	72	168	HIGH	$85.20
2	2462	Almers, C	F	34	3	66	152	HIGH	$124.80
3	2501	Bonaventure, T	F	31	17	61	123	LOW	$149.75
4	2523	Johnson, R	F	43	31	63	137	MOD	$149.75
5	2539	LaMance, K	M	51	4	71	158	LOW	$124.80
6	2544	Jones, M	M	29	6	76	193	HIGH	$124.80
7	2552	Reberson, P	F	32	9	67	151	MOD	$149.75
8	2555	King, E	M	35	13	70	173	MOD	$149.75
9	2563	Pitts, D	M	34	22	73	154	LOW	$124.80
10	2568	Eberhardt, S	F	49	27	64	172	LOW	$124.80
11	2571	Nunnelly, A	F	44	19	66	140	HIGH	$149.75
12	2572	Oberon, M	F	28	17	62	118	LOW	$85.20
13	2574	Peterson, V	M	30	6	69	147	LOW	$149.75
14	2575	Quigley, M	F	40	8	69	163	HIGH	$124.80
15	2578	Cameron, L	M	47	5	72	173	NA	$124.80
16	2579	Underwood, K	M	60	22	71	191	LOW	$149.75
17	2584	Takahashi, Y	F	43	29	65	123	MODY	$124.80
18	2586	Derber, B	M	25	23	75	188	HIGH	$85.20
19	2588	Ivan, H	F	22	20	63	139	LOW	$85.20
20	2589	Wilcox, E	F	41	16	67	141	HIGH	$149.75
21	2595	Warren, C	M	54	7	71	183	MOD	$149.75

Explained:

This code here applies dollar format on the fee variable from S.A dataset but only with values having maximum range of 9 characters including $ symbol, comma (if any), digits, decimal point, decimal values.

/* prog 19 */	**Output**

```
proc contents
data=a;
run;
```



```
                    The SAS System          02:26 Friday, May 1,

                 The CONTENTS Procedure

         Alphabetic List of Variables and Attributes

          #    Variable    Type    Len    Format

          8    ActLevel    Char      4
          4    Age         Num       8
          5    Date        Num       8
          9    Fee         Num       8    DOLLAR9.2
          6    Height      Num       8
          1    ID          Char      4
          2    Name        Char     14
          3    Sex         Char      1
          7    Weight      Num       8
```

Explained:

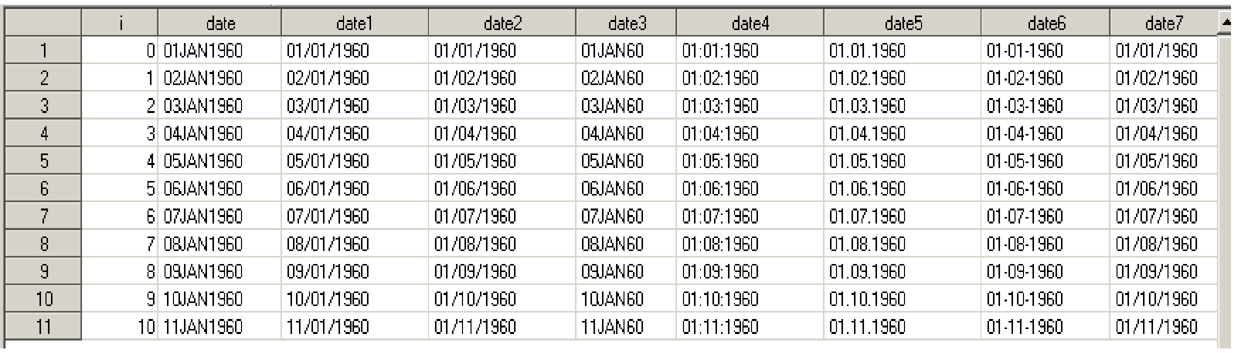
PROC CONTENTS used here displays the content and their attributes as seen above.

Dates:

Dates are stored in numeric format always.
In SAS date has starting point set to 01-JAN-1960.

```
/* prog 20 */

data a;
do i=0 to 10;
date=put(i,date9.);          /* Statement 1      */
date1=put(i,ddmmyy10.);      /* Statement 2      */
date2=put(i,mmddyy10.);      /* Statement 3      */
date3=put(i,date7.);         /* Statement 4      */
date4=put(i,mmddyyc10.);     /* Statement 5      */
date5=put(i,mmddyyp10.);     /* Statement 6      */
date6=put(i,mmddyyd10.);     /* Statement 7      */
date7=put(i,mmddyys10.);     /* Statement 8      */
output;
end;
run;
```

Output

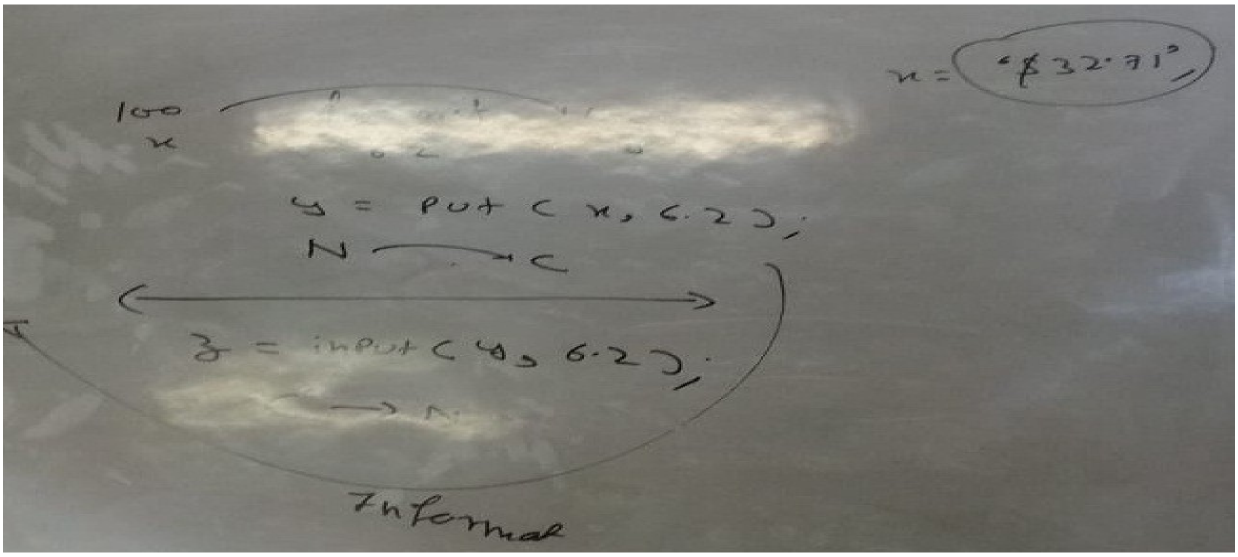


	i	date	date1	date2	date3	date4	date5	date6	date7
1	0	01JAN1960	01/01/1960	01/01/1960	01JAN60	01:01:1960	01.01.1960	01-01-1960	01/01/1960
2	1	02JAN1960	02/01/1960	01/02/1960	02JAN60	01:02:1960	01.02.1960	01-02-1960	01/02/1960
3	2	03JAN1960	03/01/1960	01/03/1960	03JAN60	01:03:1960	01.03.1960	01-03-1960	01/03/1960
4	3	04JAN1960	04/01/1960	01/04/1960	04JAN60	01:04:1960	01.04.1960	01-04-1960	01/04/1960
5	4	05JAN1960	05/01/1960	01/05/1960	05JAN60	01:05:1960	01.05.1960	01-05-1960	01/05/1960
6	5	06JAN1960	06/01/1960	01/06/1960	06JAN60	01:06:1960	01.06.1960	01-06-1960	01/06/1960
7	6	07JAN1960	07/01/1960	01/07/1960	07JAN60	01:07:1960	01.07.1960	01-07-1960	01/07/1960
8	7	08JAN1960	08/01/1960	01/08/1960	08JAN60	01:08:1960	01.08.1960	01-08-1960	01/08/1960
9	8	09JAN1960	09/01/1960	01/09/1960	09JAN60	01:09:1960	01.09.1960	01-09-1960	01/09/1960
10	9	10JAN1960	10/01/1960	01/10/1960	10JAN60	01:10:1960	01.10.1960	01-10-1960	01/10/1960
11	10	11JAN1960	11/01/1960	01/11/1960	11JAN60	01:11:1960	01.11.1960	01-11-1960	01/11/1960

Explained:

statement 1: gives date output in the default 9. Format like: 01JAN1960
statement 2: gives output date in the ddmmyy with 10.format, separated by slash (/) like 01/01/1960
Statement 3: gives output date in the mmddyy with 10.format separated by slash (/) like 01/14/1960
Statement 4: gives output in the 7. Format similar to 9. but with trimming year's 1st 2-digit. like `01JAN60`
Statement 5 :gives in the mmddyy with 10. Format separated by colon (:). like `01:01:1960`
Statement 6: gives in the mmddyy with 10. Format separated by period (.). like `01.01.1960`
Statement 7: gives in the mmddyy with 10. Format separated by dash (-). like `01-01-1960`
Statement 8: gives in the mmddyy with 10. Format separated by slash (/). like `01/01/1960`

INFORMAT

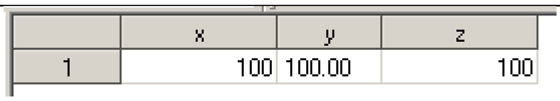


To remove the format applied on any variable value, informat helps.
Keyword used is INPUT (a,b), where

“a” is any character type variable and “b” is the applied format type.

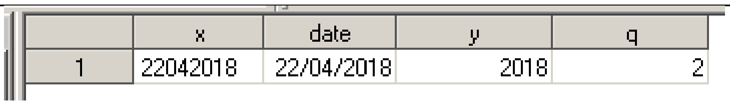
Example below:

/* prog 21 */	Output

/* Informat */
 	
data a;
x=100;
y=put (x,6.2);
z=input(y,6.2);
run;
	
	x y z
	1 100 100.00 100

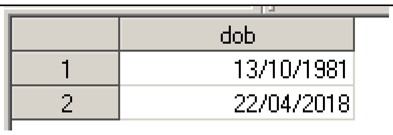
Explained:	“x”= 100, becomes a character type variable “y = 100.00” by applying put (x,6.2)format. Applying input (y,6.2) does the reverse by making it numeric again and removing all formats applied on x, hence output z = 100.

/* prog 22 */	Output

data a;
x="22042018";
date =
mdy(substr(x,3,2),substr
(x,1,2),substr(x,5));
y=year(date);
q=qtr(date);
format date ddmmyy10.;
run;
	
	x date y q
	1 22042018 22/04/2018 2018 2

Explained: MDY function: gets month date, year separated all numeric type Year (date) gets a value representing year of the date. Qtr (date) returns a value 1,2,3 or 4 according to the quarter in which the date falls.	X is character type, mdy(), is used to get month, date, year and a numeric type value for date. Year(), to get the year of the date is used. And similarly quarter(), to get the quarter in which this date falls (here, q = 2).

/* prog 23 */	Output

data a; input dob date9.; format dob ddmmyy10.; cards; 13oct1981 22apr2018 ; run;	 dob 1 13/10/1981 2 22/04/2018
Explained:	This code takes date in the 9. Format like 13oct1981 gives output in the 10.format like 22/04/2018.

Interview question: what does 730 implies in SAS?
Answer: 365*2 days from the base date 01-JAN-1960.